Oracle® Database
SQL 言語リファレンス
19c

F16180-14(原本部品番号:E96310-23)

2023年11月

タイトルおよび著作権情報

Oracle Database SQL言語リファレンス, 19c

F16180-14

原本著者: Usha Krishnamurthy

原本協力者: Mary Beth Roeser、Drew Adams、Lance Ashdown、Vikas Arora、Thomas Baby、Hermann Baer、Yasin Baskan、Nigel Bayliss、Eric Belden、Rajesh Bhatiya、Atif Chaudhry、Nelson Corcoran、Dinesh Das、Mark Dilman、Sudha Duraiswamy、Yanfei Fan、Mike Gleeson、Laura Solis Gomez、Naveen Gopal、Beda Hammerschmidt、Patricia Huey、Peter Knaggs、Sriram Krishnamurthy、Andre Kruglikov、Praveen Kumar、Bryn Llewellyn、Yunrui Li、Roger MacNicol、Darshan Maniyani、David McDermid、Jan Michels、Rahil Mir、Gopal Mulagund、Ian Neall、Padmaja Potineni、Hanlin Qian、Giridhar Ravipati、Prashanth Shantaveerappa、Wayne Smith、Samarjeet Tomar、Nirav Vyas、Alan Williams、Andy Witkowski、Sergiusz Wolicki、Tsae-Feng Yu、Weiran Zhang

目次

- APPEND_VALUESヒント
- CACHEヒント
- CHANGE_DUPKEY_ERROR_INDEXヒント
- CLUSTERヒント
- CLUSTERINGヒント
- CONTAINERSヒント
- CURSOR_SHARING_EXACTヒント
- DISABLE_PARALLEL_DMLヒント
- DRIVING_SITEヒント
- DYNAMIC_SAMPLINGヒント
- ENABLE_PARALLEL_DMLヒント
- FACTヒント
- FIRST_ROWSヒント
- FRESH_MVヒント
- FULLヒント
- GATHER_OPTIMIZER_STATISTICSヒント
- GROUPINGヒント
- HASHヒント
- IGNORE_ROW_ON_DUPKEY_INDEXヒント
- INDEXヒント
- INDEX_ASCヒント
- INDEX_COMBINEヒント
- INDEX_DESCヒント
- INDEX_FFSヒント
- INDEX_JOINヒント
- INDEX_SSヒント
- INDEX_SS_ASCヒント
- INDEX_SS_DESCヒント
- INMEMORYヒント
- INMEMORY_PRUNINGヒント
- LEADINGヒント
- MERGEヒント
- MODEL_MIN_ANALYSISヒント
- MONITORヒント
- NATIVE_FULL_OUTER_JOINヒント
- NOAPPENDヒント
- NOCACHEヒント
- NO_CLUSTERINGヒント
- NO_EXPANDヒント
- NO_FACTヒント
- NO_GATHER_OPTIMIZER_STATISTICSヒント
- NO_INDEXヒント
- NO_INDEX_FFSヒント

- NO_INDEX_SSヒント
- NO_INMEMORYヒント
- NO_INMEMORY_PRUNINGヒント
- NO_MERGEヒント
- NO_MONITORヒント
- NO_NATIVE_FULL_OUTER_JOINヒント
- NO_PARALLELヒント
- NOPARALLELヒント
- NO_PARALLEL_INDEXヒント
- NOPARALLEL_INDEXヒント
- NO_PQ_CONCURRENT_UNIONヒント
- NO_PQ_SKEWヒント
- NO_PUSH_PREDヒント
- NO_PUSH_SUBQヒント
- NO_PX_JOIN_FILTERヒント
- NO_QUERY_TRANSFORMATIONヒント
- NO_RESULT_CACHEヒント
- NO_REWRITEヒント
- NOREWRITEヒント
- NO_STAR_TRANSFORMATIONヒント
- NO_STATEMENT_QUEUINGヒント
- NO_UNNESTヒント
- NO_USE_BANDヒント
- NO_USE_CUBEヒント
- NO_USE_HASHヒント
- NO_USE_MERGEヒント
- NO_USE_NLヒント
- NO_XML_QUERY_REWRITEヒント
- NO_XMLINDEX_REWRITEヒント
- NO_ZONEMAPヒント
- OPTIMIZER_FEATURES_ENABLE Hint
- OPT_PARAMヒント
- ORDEREDヒント
- PARALLELヒント
- PARALLEL_INDEXヒント
- PQ_CONCURRENT_UNIONヒント
- PQ_DISTRIBUTEヒント
- PQ_FILTERヒント
- PQ_SKEWヒント
- PUSH_PREDヒント
- PUSH_SUBQヒント
- PX_JOIN_FILTERヒント
- QB_NAMEヒント

- CHARTOROWID
- CHR
- CLUSTER_DETAILS
- CLUSTER_DISTANCE
- CLUSTER_ID
- CLUSTER_PROBABILITY
- CLUSTER_SET
- COALESCE
- COLLATION
- COLLECT
- COMPOSE
- CON_DBID_TO_ID
- CON_GUID_TO_ID
- CON_ID_TO_CON_NAME
- CON_ID_TO_DBID
- CON_NAME_TO_ID
- CON_UID_TO_ID
- CONCAT
- CONVERT
- CORR
- CORR_*
    - CORR_S
    - CORR_K
- COS
- COSH
- COUNT
- COVAR_POP
- COVAR_SAMP
- CUBE_TABLE
- CUME_DIST
- CURRENT_DATE
- CURRENT_TIMESTAMP
- CV
- DATAOBJ_TO_MAT_PARTITION
- DATAOBJ_TO_PARTITION
- DBTIMEZONE
- DECODE
- DECOMPOSE
- DENSE_RANK
- DEPTH
- DEREF
- DUMP
- EMPTY_BLOB、EMPTY_CLOB

- EXISTSNODE
- EXP
- EXTRACT (日時)
- EXTRACT (XML)
- EXTRACTVALUE
- FEATURE_COMPARE
- FEATURE_DETAILS
- FEATURE_ID
- FEATURE_SET
- FEATURE_VALUE
- FIRST
- FIRST_VALUE
- FLOOR
- FROM_TZ
- GREATEST
- GROUP_ID
- GROUPING
- GROUPING_ID
- HEXTORAW
- INITCAP
- INSTR
- ITERATION_NUMBER
- JSON_ARRAY
- JSON_ARRAYAGG
- JSON_DATAGUIDE
- JSON_MERGEPATCH
- JSON_OBJECT
- JSON_OBJECTAGG
- JSON_QUERY
- JSON_SERIALIZE
- JSON_TABLE
- JSON_TRANSFORM
- JSON_VALUE
- LAG
- LAST
- LAST_DAY
- LAST_VALUE
- LEAD
- LEAST
- LENGTH
- LISTAGG
- LN
- LNNVL

- LOCALTIMESTAMP
- LOG
- LOWER
- LPAD
- LTRIM
- MAKE_REF
- MAX
- MEDIAN
- MIN
- MOD
- MONTHS_BETWEEN
- NANVL
- NCHR
- NEW_TIME
- NEXT_DAY
- NLS_CHARSET_DECL_LEN
- NLS_CHARSET_ID
- NLS_CHARSET_NAME
- NLS_COLLATION_ID
- NLS_COLLATION_NAME
- NLS_INITCAP
- NLS_LOWER
- NLS_UPPER
- NLSSORT
- NTH_VALUE
- NTILE
- NULLIF
- NUMTODSINTERVAL
- NUMTOYMINTERVAL
- NVL
- NVL2
- ORA_DM_PARTITION_NAME
- ORA_DST_AFFECTED
- ORA_DST_CONVERT
- ORA_DST_ERROR
- ORA_HASH
- ORA_INVOKING_USER
- ORA_INVOKING_USERID
- PATH
- PERCENT_RANK
- PERCENTILE_CONT
- PERCENTILE_DISC
- POWER

- POWERMULTISET
- POWERMULTISET_BY_CARDINALITY
- PREDICTION
- PREDICTION_BOUNDS
- PREDICTION_COST
- PREDICTION_DETAILS
- PREDICTION_PROBABILITY
- PREDICTION_SET
- PRESENTNNV
- PRESENTV
- PREVIOUS
- RANK
- RATIO_TO_REPORT
- RAWTOHEX
- RAWTONHEX
- REF
- REFTOHEX
- REGEXP_COUNT
- REGEXP_INSTR
- REGEXP_REPLACE
- REGEXP_SUBSTR
- REGR_(線形回帰)ファンクション
- REMAINDER
- REPLACE
- ROUND (日付)
- ROUND (数値)
- ROUND_TIES_TO_EVEN (数値)
- ROW_NUMBER
- ROWIDTOCHAR
- ROWIDTONCHAR
- RPAD
- RTRIM
- SCN_TO_TIMESTAMP
- SESSIONTIMEZONE
- SET
- SIGN
- SIN
- SINH
- SOUNDEX
- SQRT
- STANDARD_HASH
- STATS_BINOMIAL_TEST
- STATS_CROSSTAB

- STATS_F_TEST
- STATS_KS_TEST
- STATS_MODE
- STATS_MW_TEST
- STATS_ONE_WAY_ANOVA
- STATS_T_TEST_*
    - STATS_T_TEST_ONE
    - STATS_T_TEST_PAIRED
    - STATS_T_TEST_INDEPおよびSTATS_T_TEST_INDEPU
- STATS_WSR_TEST
- STDDEV
- STDDEV_POP
- STDDEV_SAMP
- SUBSTR
- SUM
- SYS_CONNECT_BY_PATH
- SYS_CONTEXT
- SYS_DBURIGEN
- SYS_EXTRACT_UTC
- SYS_GUID
- SYS_OP_ZONE_ID
- SYS_TYPEID
- SYS_XMLAGG
- SYS_XMLGEN
- SYSDATE
- SYSTIMESTAMP
- TAN
- TANH
- TIMESTAMP_TO_SCN
- TO_APPROX_COUNT_DISTINCT
- TO_APPROX_PERCENTILE
- TO_BINARY_DOUBLE
- TO_BINARY_FLOAT
- TO_BLOB (bfile)
- TO_BLOB (raw)
- TO_CHAR (bfile|blob)
- TO_CHAR (文字)
- TO_CHAR (日時)
- TO_CHAR (数値)
- TO_CLOB (bfile|blob)
- TO_CLOB (文字)
- TO_DATE
- TO_DSINTERVAL

- TO_LOB
- TO_MULTI_BYTE
- TO_NCHAR (文字)
- TO_NCHAR (日時)
- TO_NCHAR (数値)
- TO_NCLOB
- TO_NUMBER
- TO_SINGLE_BYTE
- TO_TIMESTAMP
- TO_TIMESTAMP_TZ
- TO_UTC_TIMESTAMP_TZ
- TO_YMINTERVAL
- TRANSLATE
- TRANSLATE ... USING
- TREAT
- TRIM
- TRUNC (日付)
- TRUNC (数値)
- TZ_OFFSET
- UID
- UNISTR
- UPPER
- USER
- USERENV
- VALIDATE_CONVERSION
- VALUE
- VAR_POP
- VAR_SAMP
- VARIANCE
- VSIZE
- WIDTH_BUCKET
- XMLAGG
- XMLCAST
- XMLCDATA
- XMLCOLATTVAL
- XMLCOMMENT
- XMLCONCAT
- XMLDIFF
- XMLELEMENT
- XMLEXISTS
- XMLFOREST
- XMLISVALID
- XMLPARSE

はじめに

このマニュアルでは、Oracle Databaseの情報を管理するために使用されるStructured Query Language(SQL)について説明します。Oracle SQLは、ANSI(米国規格協会)およびISO(国際標準化機構)SQLの規格にさらに多くの内容を盛り込んでいます。

ここでは、次の項目について説明します。

対象読者

Oracle Database SQL言語リファレンスは、すべてのOracle SQLユーザーを対象としています。

ドキュメントのアクセシビリティ

オラクルのアクセシビリティについての詳細情報は、Oracle Accessibility ProgramのWebサイト(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc)を参照してください。

Oracleサポートへのアクセス

サポートをご契約のお客様には、My Oracle Supportを通して電子支援サービスを提供しています。詳細情報は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info)か、聴覚に障害のあるお客様は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs)を参照してください。

関連ドキュメント

詳細は、次のOracleドキュメントを参照してください。

- PL/SQL、Oracle SQLに対する手続き型言語の拡張の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。
- Oracle埋込みSQLの詳細は、『Pro*C/C++プログラマーズ・ガイド』および『Pro*COBOLプログラマーズ・ガイド』を参照してください。

このマニュアルの例の多くは、Oracle Databaseのインストール時に基本インストール・オプションを選択した場合にデフォルトでインストールされるサンプル・スキーマを使用しています。これらのスキーマがどのように作成されているか、およびその使用方法については、『Oracle Databaseサンプル・スキーマ』を参照してください。

表記規則

このドキュメントでは、次のテキスト表記規則が使用されます:

規則	意味
boldface	太字体は、アクションに関連付けられたグラフィカル・ユーザー・インタフェース要素や、本文または用語集で定義されている用語を示します。
italic	イタリック体は、ブック・タイトル、強調、またはユーザーが特定の値を指定するプレースホルダー変数を示します。
`monospace`	等幅体は、段落内のコマンド、URL、サンプル内のコード、画面に表示されるテキスト、またはユーザーが入力するテキストを示します。

『Oracle Database SQL言語リファレンス』のこのリリースでの変更点

内容は次のとおりです。

Oracle Databaseリリース19cにおける変更点

Oracle Database 19cの『Oracle Database SQL言語リファレンス』で変更された点を次に示します。

新機能

リリース19cの新機能は次のとおりです。

Microsoft Azure ADユーザーへのOracle Databaseスキーマおよびロールのマッピング

CREATE USERの拡張されたGLOBALLY句を使用して、Oracle DatabaseスキーマをMicrosoft Azure ADユーザーに排他的にマップできます。

CREATE ROLEの拡張されたGLOBALLY句を使用して、共有OracleスキーマまたはOracle Databaseグローバル・ロールをAzure ADアプリケーション・ロールにマップできます。

SQLマクロ

SQLマクロ(SQM)を作成して、共通のSQL式および文を、他のSQL文で使用できる再利用可能なパラメータ化された構造体にくくり出すことができます。

Oracle Databaseリリース19c, バージョン19.7以降、SQL表マクロがサポートされています。SQL表マクロは、通常FROM句で使用される式で、多相(パラメータ化)ビューのように機能します。

SQL表マクロにより、開発者の生産性が高まり、コラボレーション開発が簡略化され、コード品質が向上します。

粒度の高いサプリメンタル・ロギング

粒度の高いサプリメンタル・ロギングでは、データベース・レベルまたはスキーマ・レベルでサプリメンタル・ロギングが有効になっている場合でも、部分データベース・レプリケーション・ユーザーは、関心のない表に対するサプリメンタル・ロギングを無効にできます。

この機能は、データベース内の一部の表に対してのみサプリメンタル・ロギングが必要な場合に使用し、リソース使用率およびREDO生成のオーバーヘッドを大幅に削減します。

この機能を使用するには、ALTER DATABASEおよびALTER PLUGGABLE DATABASE文の supplemental_db_logging句に追加された新しいsupplemental_subset_replication_clauseを構成します。

LOBロケータの署名ベース・セキュリティ

このリリース以降、ラージ・オブジェクト(LOB)ロケータの署名ベース・セキュリティを構成できます。LOB署名キーは、マルチテナントPDBまたはスタンドアロンの非マルチテナント・データベースのどちらでも使用できます。

ALTER DATABASE DICTIONARY ENCRYPT CREDENTIALS SQL文を実行することで、LOB署名キー資格証明の暗号化を有効にできます。それ以外の場合、資格証明は不明瞭化された形式で格納されます。LOB署名キーを暗号化され

た形式で格納する場合、データベースまたはPDBにオープンTDEキーストアが必要です。

REST APIを使用したクラウドのオブジェクト・ストアのサポート

Oracle Data Pumpは、サポートされているオブジェクト・ストアにあるファイルからデータをインポートできます。

ALTER DATABASE文の新しいproperty_clauseを使用して、コマンドラインでデータベース資格証明を指定し、デフォルトの資格証明をデータベースに格納できます。

システム管理シャーディングに対する複数表ファミリのサポート

この機能は、システム管理シャード・データベースにのみ適用されます。CREATE SHARDED TABLE文を使用して、1つのシャード・データベースで異なる表ファミリにアクセスする様々なアプリケーションをホストできます。

システム・シャーディングを使用する表ファミリは複数作成できますが、ユーザー定義および複合シャーディングでは1つの表ファミリのみがサポートされます。

シャード間の一意の連番の生成

一意制約を持つ主キー以外の列のシャード間でグローバルに一意の連番を生成でき、自分で管理する必要がありません。シャード・データベースによってこれらの連番が管理されます。

CREATE SEQUENCE文またはALTER SEQUENCE文のSHARD句を使用して、シャード間で一意の連番を生成します。

インメモリー外部表のビッグ・データおよびパフォーマンスの向上

CREATE TABLEおよびALTER TABLE文のinmemory_clauseでは、ORACLE_HIVEおよびORACLE_BIGDATAドライバ・タイプの指定をサポートしています。

非パーティション表のinmemory_clause句でINMEMORYを指定して、これらのドライバ・タイプをサポートできます。

ビットマップ・ベースの固有カウントのSQLファンクション

5つの新しいビット・ベクトル・ファンクションを使用して、SQL問合せ内でCOUNT DISTINCT操作を高速化できます。

- BITMAP_BUCKET_NUMBER
- BITMAP_BIT_POSITION
- BITMAP_CONSTRUCT_AGG
- BITMAP_OR_AGG
- BITMAP_COUNT

Memoptimized Rowstore - 高速インジェスト

高速収集を有効にするには、CREATE TABLEまたはALTER TABLEのmemoptimize_write_clauseを使用します。高速収集は、ディスクに書き込む前に挿入を格納するバッファ・プールを利用して、モノのインターネット(IoT)アプリケーションから1行データを挿入する頻繁なメモリー処理を最適化します。

JSONとオブジェクトのマッピング

この機能により、JSONデータと、ユーザー定義のSQLオブジェクト型およびコレクションとの間のマッピングが可能になります。

SQL/JSONファンクションjson_valueを使用して、JSONデータをSQLオブジェクト型のインスタンスに変換できます。反対の方向では、SQL/JSONファンクションjson_objectまたはjson_arrayを使用して、SQLオブジェクト型のインスタンスからJSONデータを生成できます。

JSON_MERGEPATCHファンクション

新しいSQLファンクションjson_mergepatchを使用して、JSONドキュメントを宣言的に更新できるようになりました。1つの文を使用して複数のドキュメントに1つ以上の変更を適用できます。

この機能により、JSON更新操作の柔軟性が向上します。

JSON構文の簡略化

構文の簡略化は、SQL/JSONパス式およびファンクションjson_objectによるSQL/JSON生成用に提供されます。新しいSQL問合せ句NESTEDは、json_tableをLEFT OUTER JOINとともに使用する方法の単純な代替方法を提供します。

GeoJSONデータに対するJSON_SERIALIZEおよびJSONデータ・ガイドのサポート

新しいSQLファンクションjson_serializeを使用して、JSONデータをテキストまたはUTFエンコードBLOBデータにシリアライズできます。

SQL集計ファンクションjson_dataguideでは、GeoJSON地理データをドキュメント内で検出できるようになりました。これを使用して、このようなデータをSQLデータ型SDO_GEOMETRYとして投影するビューを作成できます。

ハイブリッド・パーティション表

一部のパーティションがOracleデータベース・セグメント内に存在し、一部のパーティションが外部ファイルおよびソース内に存在するハイブリッド・パーティション表を作成できます。内部パーティションおよび外部パーティションは、必要に応じて単一のパーティション表に統合できます。

CREATE TABLEまたはALTER TABLEのtable_partition_description句でINTERNALまたはEXTERNALを指定します。

パリティ保護されたファイル

高レベルの冗長性を必要としないアーカイブ・ログやバックアップ・セットなどのライトワンス・ファイルの単一のパリティを構成し、領域を節約できます。

ALTER DISKGROUP文のredundancy_clauseにPARITYを指定します。

LISTAGG集計のDISTINCTオプション

LISTAGG集計ファンクションでは、新しいDISTINCTキーワードを使用した重複削除がサポートされるようになりました。

統合監査のトップ・レベル文

ユーザーが直接発行したSQL文を監査する場合は、CREATE AUDIT POLICY文(統合監査)にONLY TOPLEVEL句を指定します。

by_users_with_roles_clauseを使用して、指定したロールを直接的または間接的に付与されたユーザーのポリシーを有効化できるようになりました。

OMF以外のモードでの自動名前変更の透過的オンライン変換のサポート

ALTER TABLESPACE ENCRYPTION文でFILE_NAME_CONVERTを省略すると、Oracleは内部的に補助ファイルの名前を選択し、後でその名前を元の名前に戻します。

ALTER SYSTEM句のFLUSH PASSWORDFILE_METADATA_CACHE

コマンドALTER SYSTEM FLUSH PASSWORDFILE_METADATA_CACHEは、SGAに格納されているパスワード・ファイルのメタデータ・キャッシュをフラッシュし、変更があったことをデータベースに通知します。

非推奨となった機能

次の機能は、このリリースでは非推奨であり、将来のリリースではサポートされなくなる可能性があります。

SQLNET.ENCRYPTION_WALLET_LOCATIONパラメータは非推奨になりつつあります。

サポート対象外機能

次の機能は、Oracle Database Release 19cではサポート対象外となります。

- Oracle Multimediaはサポート対象外となります。
- Oracle Streamsはサポート対象外となります。

リリース19cでサポート対象外となる機能の詳細なリストは、『Oracle Databaseアップグレード・ガイド』を参照してください。

1 Oracle SQLの概要

Structured Query Language(SQL)とは、プログラムおよびユーザーが、Oracle Databaseのデータにアクセスするために使用する一連の文です。アプリケーション・プログラムやOracleのツール製品を使用すると、SQLを直接使用せずにデータベースにアクセスできます。ただし、アプリケーションがユーザーの要求を実行するときには、必ずSQLを使用します。この章では、ほとんどのデータベース・システムで使用されるSQLの背景について説明します。

この章には次のトピックが含まれます:

- SQLの歴史
- SQL規格
- 字句規則
- ツール製品のサポート

SQLの歴史

博士E. F.Coddは、1970年6月、Association of Computer Machinery(ACM)が刊行した「Communications of the ACM」誌で、論文「大型共用データ・バンク用のデータのリレーショナル・モデル」を発表しました。Codd博士のモデルは、現在ではリレーショナル・データベース管理システム(RDBMS)の完成したモデルとして認められています。Structured English Query Language(SEQUEL)は、IBM社がCodd博士のモデルを使用するために開発したものです。このSEQUELが後のSQLです。1979年、Relational Software, Inc.(現在のオラクル社)は、商業的に利用可能な最初のSQLの処理系を導入しました。今日、SQLは標準のRDBMS言語として認められています。

SQL規格

Oracleでは業界に受け入れられた標準への準拠に努め、SQL標準調査会に積極的に参加しています。業界で認知されている委員会には、ANSI(米国規格協会)、およびIEC(国際電気標準会議)が電気・電子部門を担当しているISO(国際標準化機構)があります。ANSIおよびISO/IECはいずれも、リレーショナル・データベースの標準言語としてSQLを認可しています。新しいSQL規格がこの両機関から同時に発表された場合、その規格の名前は、各機関の規則に従って付けられますが、技術的な詳細は同じです。

関連項目:

SQL規格へのOracle Databaseの規格準拠は、「Oracleと標準SQL」を参照してください。

SQLの特長

SQLは、アプリケーション・プログラマ、データベース管理者、マネージャ、エンド・ユーザーなど、あらゆる分野のユーザーに利益をもたらします。技術的ないい方をすると、SQLはデータ副言語です。SQLの目的は、Oracle Databaseのようなリレーショナル・データベースとのインタフェースを提供することであり、すべてのSQL文はデータベースに対する命令です。この点において、SQLは、CやBASICのような汎用プログラミング言語と異なります。SQLには、次のような特長があります。

- 個々の単位としてではなく、グループとして一連のデータを処理します。
- データへの自動的なナビゲーション・アクセス(経路設定)を実行します。
- 使用される文は、それぞれが複雑、強力で、スタンドアロン型の文です。当初、begin-end、if-then-else、ループ、例外状態処理などのフロー制御文は、SQLおよびSQL標準の一部ではありませんでしたが、現在はISO/IEC 9075-4 - Persistent Stored Modules (SQL/PSM)に存在します。Oracle SQLに対するPL/SQLの拡張は、PSMに似ています。

SQLでは、データを論理的なレベルで処理できます。処理系について考えることは、データの細部を操作する場合のみで済みます。たとえば、表から一連の行を検索するには、行をフィルタ処理するための条件を定義します。この条件を満たすすべての行が1つのステップで検索され、ユーザー、別のSQL文またはアプリケーションに1つの単位として渡されます。行単位で処理する必要がなく、行の物理的な格納方法や検索方法を気にする必要もありません。SQL文を実行すると、Oracle Databaseの問合せオプティマイザが働きます。この機能によって、指定したデータに最も速くアクセスする方法が決定されます。Oracleには、オプティマイザの性能を向上させる方法も用意されています。

SQL文を使用して、次の処理を行うことができます。

- データの問合せ
- 表の中の行の挿入、更新、削除
- オブジェクトの作成、置換、変更および削除
- データベースおよびそのオブジェクトへのアクセスの制御
- データベースの整合性と一貫性の保証

SQLでは、前述のすべてのタスクを1つの一貫性のある言語に統一しました。

すべてのリレーショナル・データベースに共通の言語

すべての主なリレーショナル・データベース管理システムは、SQLをサポートしているため、SQLで得た技術的な知識を他のデータベースでも生かすことができます。さらに、SQLで記述されたプログラムは移植性に優れています。わずかな変更のみで他のデータベースに移行できます。

Enterprise Managerの使用方法

SQL構文を使用して実行できる操作の多くは、Enterprise Managerを使用するとさらに簡単に実行できます。詳細は、Oracle Enterprise Managerのドキュメント・セット、『Oracle Database 2日でデータベース管理者』またはOracle Databaseの「2日で」シリーズのいずれかのマニュアルを参照してください。

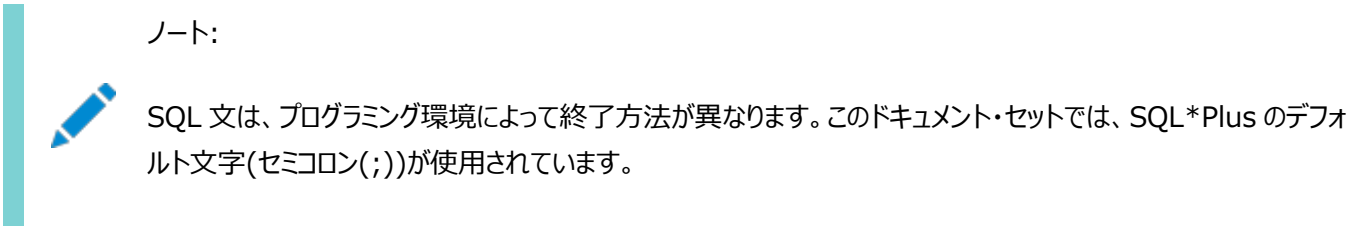
字句規則

SQL文の記述に関する次の字句規則は、Oracle DatabaseのSQL実装に対してのみ適用されますが、他のすべてのSQL実装にも一般的に適用されます。

SQL文では、文の定義の中で空白が入る可能性がある任意の位置に、1つ以上のタブ、改行文字、空白またはコメントを記述できます。したがって、Oracle Databaseは、次の2つの文を同一と解釈します。

```
SELECT last_name,salary*12,MONTHS_BETWEEN(SYSDATE,hire_date)
  FROM employees
  WHERE department_id = 30
  ORDER BY last_name;
SELECT last_name,
  salary * 12,
        MONTHS_BETWEEN( SYSDATE, hire_date )
FROM employees
WHERE department_id=30
ORDER BY last_name;
```

予約語、キーワード、識別子およびパラメータは、大文字と小文字を区別せずに記述できます。ただし、テキスト・リテラルと引用名では大文字と小文字が区別されます。テキスト・リテラルの構文の詳細は、テキスト・リテラルを参照してください。



ノート:

SQL 文は、プログラミング環境によって終了方法が異なります。このドキュメント・セットでは、SQL*Plus のデフォルト文字(セミコロン(;))が使用されています。

ツールのサポート

Oracleには、SQL開発プロセスを容易にする多くのユーティリティが提供されています。

- Oracle SQL Developerは、データベース・オブジェクトの参照、作成、編集および削除、PL/SQLコードの編集およびデバッグ、SQL文およびスクリプトの実行、データの操作およびエクスポート、レポートの作成および表示を行うためのグラフィカル・ツールです。SQL Developerを使用すると、標準のOracle Database認証を使用してOracle Databaseの任意のターゲット・スキーマに接続することができます。接続後に、データベース内のオブジェクトに対する操作を行うことができます。また、MySQL、Microsoft SQL Server、Microsoft Accessなど、サード・パーティ(Oracle以外)のデータベースのスキーマに接続して、このデータベースのメタデータを表示し、Oracleにこのデータベースを移行することもできます。
- SQL*Plusは、対話形式のバッチ問合せツールです。すべてのOracle Databaseサーバーまたはクライアントの使用環境にインストールされています。このツールには、コマンドライン・ユーザー・インタフェースおよびiSQL*PlusというWebベースのユーザー・インタフェースが備わっています。
- Oracle JDeveloperは、マルチプラットフォームの統合開発環境であり、Java、WebサービスおよびSQLの開発のすべての工程をサポートします。この環境では、SQL文の実行およびチューニング用のグラフィカル・インタフェースと、ビジュアル・スキーマ・ダイアグラマ(データベース・モデラー)を使用できます。また、PL/SQLアプリケーションの編集、コンパイルおよびデバッグもサポートします。
- Oracle Application Expressは、データベース関連のWebアプリケーションの開発およびデプロイ用のホストされた環境です。Oracle Application Expressのコンポーネントであ るSQL Workshopを使用すると、Webブラウザからデータベース・オブジェクトを表示および管理できます。SQL Workshopでは、SQLコマンド・プロセッサおよびSQLスクリプト・リポジトリに素早くアクセスできます。

関連項目:

これらの製品の詳細は、『SQL*Plusユーザーズ・ガイドおよびリファレンス』および『Oracle Application Expressアプリケーション・ビルダー・ユーザーズ・ガイド』を参照してください。

Oracle Call InterfaceおよびOracleプリコンパイラを使用すると、標準SQL文をプロシージャ・プログラミング言語に埋め込むことができます。

- Oracle Call Interface(OCI)を使用すると、CプログラムにSQL文を埋め込むことができます。
- OracleプリコンパイラであるPro*C/C++およびPro*COBOLによって、埋込みSQL文が解析され、それぞれC/C++コンパイラとCOBOLコンパイラが処理できる文に変換されます。

関連項目:

各製品で使用可能な埋込みSQL文の詳細は、『Oracle C++ Call Interfaceプログラマーズ・ガイド』、『Pro*COBOLプログラマーズ・ガイド』および『Oracle Call Interfaceプログラマーズ・ガイド』を参照してください。

ほとんどのOracleツール製品は、Oracle SQLのすべての機能もサポートしています。このマニュアルでは、SQLのすべての機能について説明しています。ご使用のOracleのツール製品でサポートしていない機能がある場合は、『SQL*Plusユーザーズ・ガイドおよびリファレンス』など、そのOracleのツール製品について記述しているマニュアルで、制限事項を確認してください。

2 Oracle SQLの基本要素

この章では、Oracle SQLの基本要素に関する参照情報を説明します。これらの要素は、SQL文の最も単純な構成ブロックです。したがって、このマニュアルで説明されているSQL文を使用する前に、この章で説明する概念を理解しておく必要があります。

この章では、次の内容を説明します。

- データ型
- データ型の比較規則
- リテラル
- 書式モデル
- NULL
- 説明
- データベース・オブジェクト
- データベース・オブジェクト名および修飾子
- スキーマ・オブジェクトの構文およびSQL文の構成要素

データ型

Oracle Databaseが処理する値は、それぞれデータ型を持ちます。値のデータ型により、固定された一連のプロパティが値に関連付けられます。このプロパティに応じて、Oracleは、あるデータ型の値を別のデータ型の値と区別して扱います。たとえば、NUMBERデータ型の値は加算できますが、RAWデータ型の値は加算できません。

表またはクラスタを作成する場合、各列にデータ型を指定する必要があります。プロシージャまたはストアド・ファンクションを作成する場合は、その各引数にデータ型を指定する必要があります。データ型によって、各列が含むことができる値のドメイン、または各引数が持つことができる値のドメインが決まります。たとえば、DATE列は、2月29日(うるう年を除く)、2または'SHOE'という値を格納できません。列に入る値は、その列のデータ型を受け継ぎます。たとえば、DATE列に'01-JAN-98'を挿入すると、Oracleはそれが有効な日付に変換されることを確認してから、文字列'01-JAN-98'をDATE値として扱います。

Oracle Databaseには、多くの組込みデータ型、およびデータ型として使用できるいくつかのユーザー定義型のカテゴリがあります。Oracleのデータ型の構文については、次の構文図で示します。この項は、次の6つの項にわかれています。

- Oracleの組込みデータ型
- ANSI、DB2、SQL/DSのデータ型
- ユーザー定義型
- Oracleが提供する型
- データ型の比較規則
- データ変換

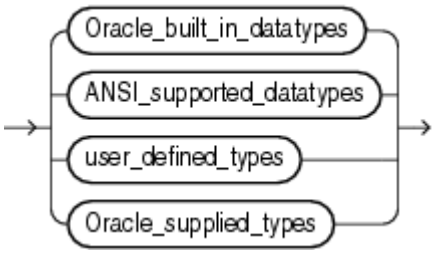
データ型はスカラーまたは非スカラーです。スカラー型は、アトム値を持ちますが、非スカラー型(コレクションともいう)は一連の値を持ちます。ラージ・オブジェクト(LOB)は、スカラー型の特別な形で、バイナリまたは文字データのラージ・スカラー値を表しています。LOBはサイズが大きいため、他のスカラー型には影響しないいくつかの制限事項があります。これらの制限事項については、関連するSQL構文を参照してください。

関連項目:

LOB列の制限事項

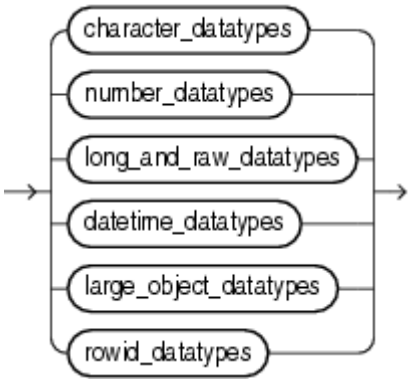
Oracleプリコンパイラでは、埋込みSQLプログラム内の他のデータ型が認識されます。このようなデータ型は、外部データ型と呼ばれ、ホスト変数に対応付けられています。組込みデータ型およびユーザー定義型を外部データ型と混同しないでください。Oracleによる外部データ型と組込み型またはユーザー定義データ型の間の変換など、外部データ型の詳細は、『Pro*COBOLプログラマーズ・ガイド』および『Pro*C/C++プログラマーズ・ガイド』を参照してください。

datatypes::=

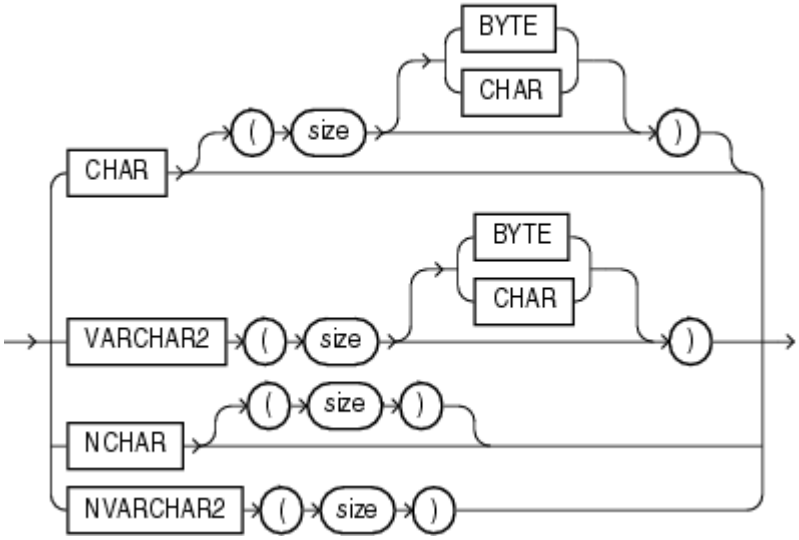


次の図に、Oracleの組込みデータ型について示します。詳細は、Oracleの組込みデータ型を参照してください。

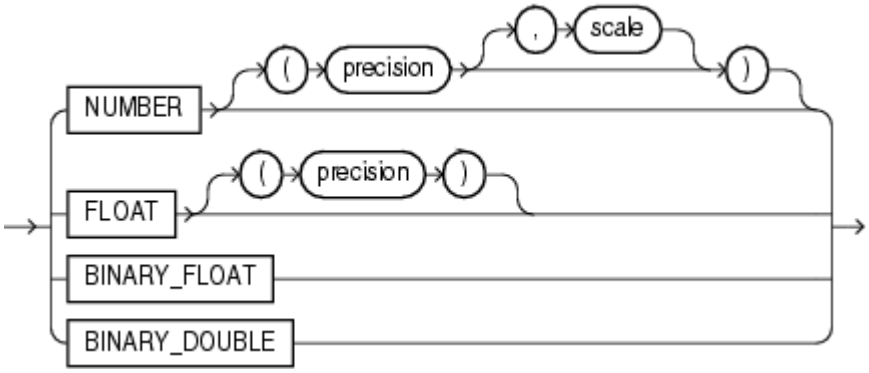
Oracle_built_in_datatypes::=



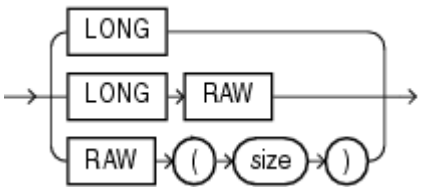
character_datatypes::=



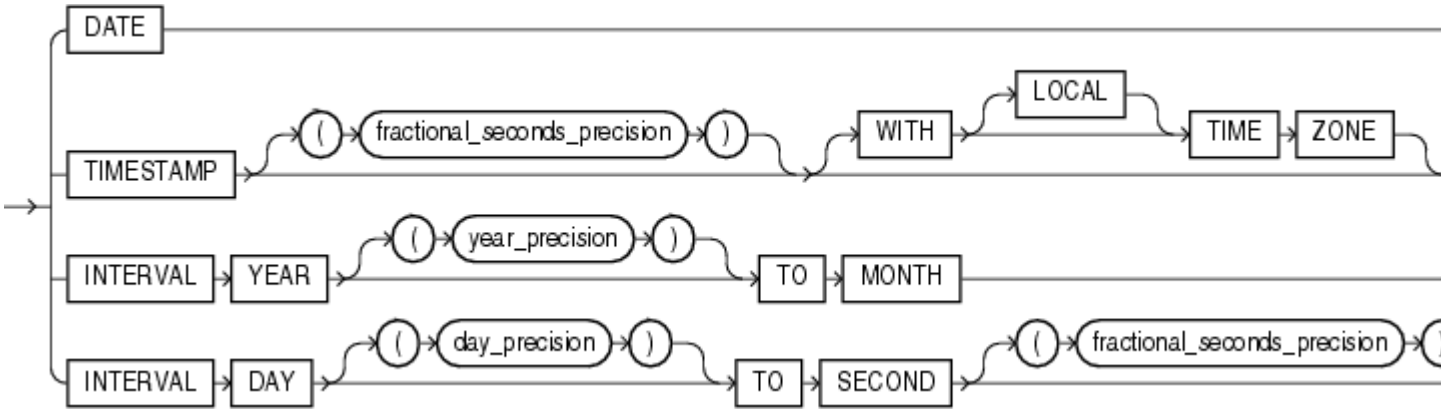
number_datatypes::=



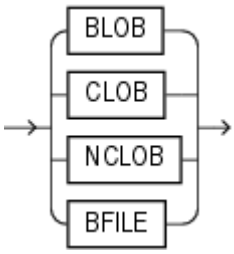
long_and_raw_datatypes::=



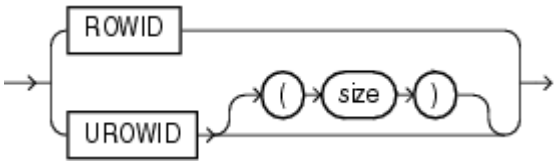
datetime_datatypes::=



large_object_datatypes::=

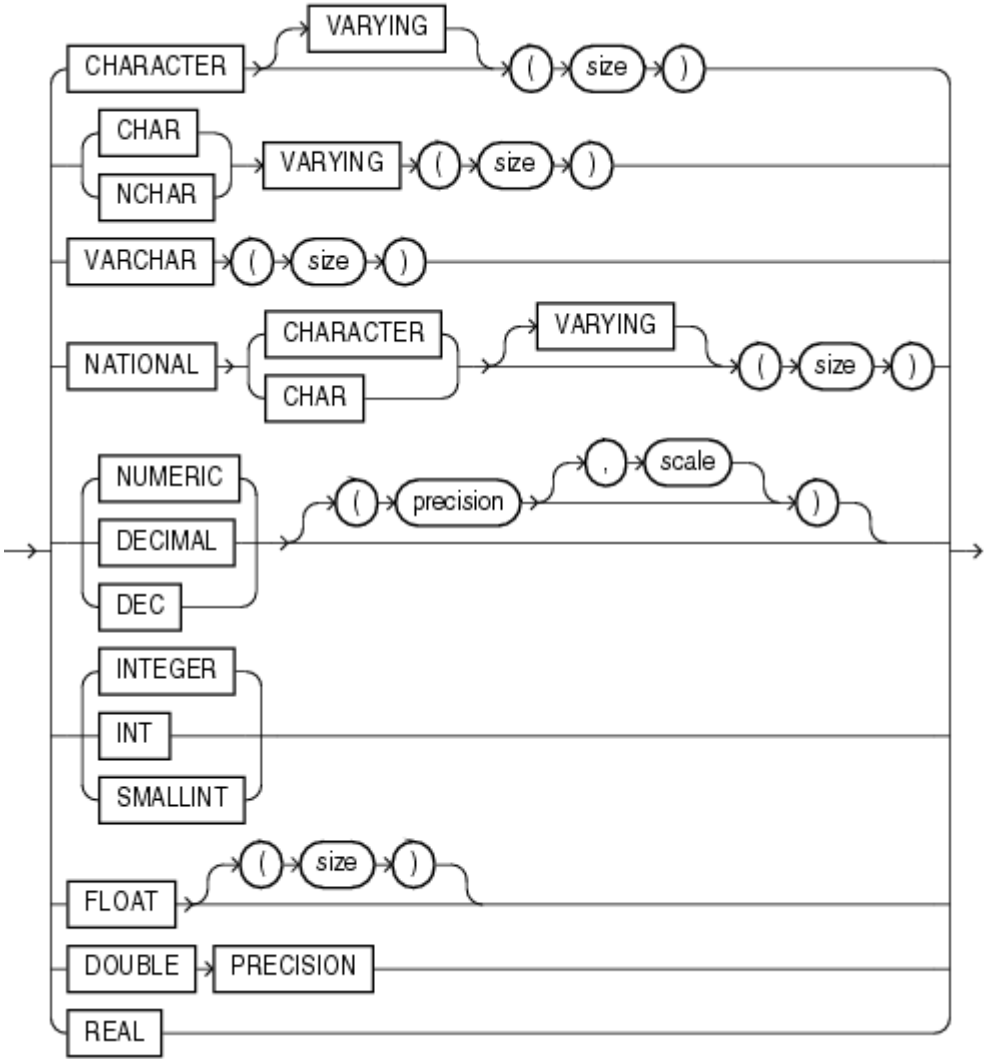


rowid_datatypes::=



次に、ANSIがサポートしているデータ型について示します。また、ANSI、DB2、SQL/DSのデータ型に、ANSIがサポートしているデータ型をOracle組込みデータ型にマップする方法を示します。

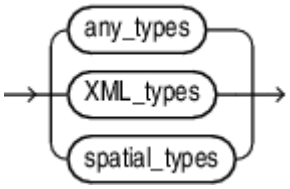
ANSI_supported_datatypes::=



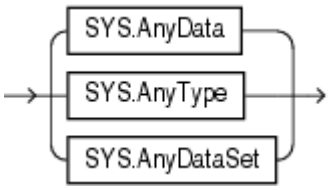
ユーザー定義型の詳細は、ユーザー定義型を参照してください。

次の図に、Oracleが提供するデータ型について示します。詳細は、Oracleが提供する型を参照してください。

Oracle_supplied_types::=

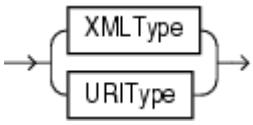


any_types::=



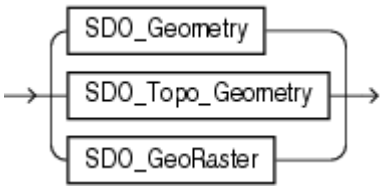
任意型の詳細は、任意型を参照してください。

XML_types::=



XML型の詳細は、XML型を参照してください。

spatial_types::=



Spatial型の詳細は、Spatial型を参照してください。

Oracleの組込みデータ型

組込みデータ型の概要表には、使用可能な組込みデータ型がリストされています。Oracle Databaseでは、データ型を内部的に識別するためにコードが使用されます。これは、組込みデータ型の概要表の「コード」列の数字です。表のコードは、DUMP関数を使用して検証できます。

組込みデータ型の概要表に示されている組込みデータ型に加えて、Oracle Databaseでは、DUMPファンクションによって表示される多くのデータ型が内部的に使用されます。

表2-1 組込みデータ型の概要

コード	データ型	説明
1	VARCHAR2(size [BYTE | CHAR])	最大長が size バイトまたは文字の可変長文字列。VARCHAR2 には、size を指定する必要があります。最小の size は 1 バイトまたは 1 文字です。最大の size は次のとおりです。 ● MAX_STRING_SIZE = EXTENDED の場合は、32,767 バイト(文字) ● MAX_STRING_SIZE = STANDARD の場合は、4,000 バイト(文字) MAX_STRING_SIZE 初期化パラメータの詳細は、拡張データ型を参照してください。 BYTE は、列がバイト長セマンティクスを持つことを示します。CHAR は、列がキャラクタ・セマンティクスを持つことを示します。
1	NVARCHAR2(size)	最大長が size 文字の可変長 Unicode 文字列。NVARCHAR2 には、size を指定する必要があります。バイト数は、AL16UTF16 エンコードの場合は size の 2 倍まで、UTF8 エンコードの場合は size

コード	データ型	説明
		の 3 倍までです。最大の size は、各国語文字セット定義によって決定されます。上限は次のとおりです。 ● 32767 バイト(MAX_STRING_SIZE = EXTENDED の場合) ● 4000 バイト(MAX_STRING_SIZE = STANDARD の場合) MAX_STRING_SIZE 初期化パラメータの詳細は、拡張データ型を参照してください。
2	NUMBER [(p [, s])]	精度 p、位取り s を持つ数。精度 p には 1 から 38 の値を指定できます。位取り s には-84 から 127 の値を指定できます。精度とスケールは両方とも 10 進数です。NUMBER の値は 1 から 22 バイトである必要があります。
2	FLOAT [(p)]	精度 p を持つ NUMBER データ型のサブタイプ。FLOAT の値は、内部的に NUMBER と表されます。精度 p には 1 から 126 の 2 進数を指定できます。FLOAT の値は 1 から 22 バイトである必要があります。
8	LONG	最大 2GB(2^{31} から 1 を引いたバイト数)の可変長文字データ。下位互換性を維持するために提供されています。
12	DATE	紀元前 4712 年 1 月 1 日から紀元 9999 年 12 月 31 日までの日付を指定します。デフォルトの書式は、NLS_DATE_FORMAT パラメータを使用すると明示的に、NLS_TERRITORY パラメータを使用すると暗黙的に決定される。サイズは 7 バイトに固定されています。このデータ型には、YEAR、MONTH、DAY、HOUR、MINUTE および SECOND の日時フィールドが含まれます。小数部の秒数およびタイムゾーンは含まれません。
100	BINARY_FLOAT	32 ビットの浮動小数点数。このデータ型には 4 バイトが必要です。
101	BINARY_DOUBLE	64 ビットの浮動小数点数。このデータ型には 8 バイトが必要です。
180	TIMESTAMP [(fractional_seconds_precision)]	日付の年、月、日および時刻の時、分、秒の値。fractional_seconds_precision は、SECOND 日時フィールドの小数部の桁数です。fractional_seconds_precision の有効範囲は 0 から 9 です。デフォルトは 6 です。デフォルトの書式は、

コード	データ型	説明
		NLS_TIMESTAMP_FORMAT パラメータによって明示的に決まるか、または NLS_TERRITORY パラメータによって暗黙的に決まります。サイズは、精度に応じて 7 または 11 バイトです。このデータ型には、YEAR、MONTH、DAY、HOUR、MINUTE および SECOND の日時フィールドが含まれます。小数部の秒数は含まれますが、タイムゾーンは含まれません。
181	TIMESTAMP [(fractional_seconds_precision)] WITH TIME ZONE	タイムゾーンによる時差などのすべての TIMESTAMP の値。fractional_seconds_precision は、SECOND 日時フィールドの小数部の桁数です。0 から 9 までの値を使用できます。デフォルトは 6 です。TIMESTAMP WITH TIME ZONE データ型のデフォルトの日付書式は、NLS_TIMESTAMP_TZ_FORMAT 初期化パラメータによって決定されます。サイズは 13 バイトに固定されています。このデータ型には、YEAR、MONTH、DAY、HOUR、MINUTE、SECOND、TIMEZONE_HOUR および TIMEZONE_MINUTE の日時フィールドが含まれます。小数部の秒数および明示的なタイムゾーンが含まれます。
231	TIMESTAMP [(fractional_seconds_precision)] WITH LOCAL TIME ZONE	TIMESTAMP WITH TIME ZONE のすべての値。ただし、次に示す例外があります。 ● データベースへの格納時、データがデータベースのタイムゾーンに正規化されている場合。 ● データの検索時、ユーザーがセッションのタイムゾーンでデータを検索する場合。 デフォルトの書式は、NLS_TIMESTAMP_FORMAT パラメータによって明示的に決まるか、または NLS_TERRITORY パラメータによって暗黙的に決まります。サイズは、精度に応じて 7 または 11 バイトです。
182	INTERVAL YEAR [(year_precision)] TO MONTH	年および月で期間を格納。year_precision は、YEAR 日時フィールドの桁数です。0 から 9 までの値を使用できます。デフォルトは 2 です。サイズは 5 バイトに固定されています。
183	INTERVAL DAY [(day_precision)] TO SECOND [(fractional_seconds_precision)]	日、時、分および秒で期間を格納。 ● day_precision は、DAY 日時フィールドの最大桁数です。0 から 9 までの値を使用できます。デフォルトは 2 です。

コード	データ型	説明
		● fractional_seconds_precision は、SECOND フィールドの小数部の桁数です。0 から 9 までの値を使用できます。デフォルトは 6 です。 サイズは 11 バイトに固定されています。
23	RAW(size)	長さ size バイトのバイナリ・データ。RAW 値には、size を指定する必要があります。最大の size は次のとおりです。 ● 32767 バイト(MAX_STRING_SIZE = EXTENDED の場合) ● 2000 バイト(MAX_STRING_SIZE = STANDARD の場合) MAX_STRING_SIZE 初期化パラメータの詳細は、拡張データ型を参照してください。
24	LONG RAW	最大 2GB の可変長バイナリ・データ。
69	ROWID	表の行のアドレスを一意に表す BASE64 文字列。主に、ROWID 疑似列によって戻される値のためのデータ型です。
208	UROWID [(size)]	索引構成表の行の論理アドレスを表す BASE64 文字列。オプションの size は、UROWID 型の列のサイズです。最大サイズおよびデフォルトは 4000 バイトです。
96	CHAR [(size [BYTE | CHAR])]	長さ size バイトまたは size 文字の固定長文字データ。最大の size は 2000 バイトまたは文字です。デフォルトは size の最小値の 1 バイトです。 BYTE および CHAR は、VARCHAR2 と同じセマンティクスを持ちます。
96	NCHAR[(size)]	長さ size 文字の固定長文字データ。バイト数は、AL16UTF16 エンコードの場合は size の 2 倍まで、UTF8 エンコードの場合は size の 3 倍までです。最大の size は、各国語文字セット定義(上限 2000 バイト)によって決定されます。デフォルトは size の最小値の 1 文字です。
112	CLOB	シングルバイト文字またはマルチバイト文字を含むキャラクタ・ラージ・オブジェクト。固定幅および可変幅の文字セットがサポートされ、両方と

コード	データ型	説明
		もデータベース文字セットで使用されます。最大サイズは、4GB から 1 を引いたバイト数にデータベース・ブロック・サイズを掛けた値です。
112	NCLOB	Unicode キャラクタを含むキャラクタ・ラージ・オブジェクト。固定幅および可変幅の文字セットがサポートされます。両方の文字セットでデータベースの各国語文字セットを使用します。最大サイズは、4GB から 1 を引いたバイト数にデータベース・ブロック・サイズを掛けた値です。各国語文字セット・データが格納されます。
113	BLOB	バイナリ・ラージ・オブジェクト。最大サイズは、4GB から 1 を引いたバイト数にデータベース・ブロック・サイズを掛けた値です。
114	BFILE	データベース外に保存された大きなバイナリ・ファイルへロケータを格納。データベース・サーバー上に存在する外部 LOB へのバイト・ストリーム I/O アクセスを可能にします。最大サイズは 4GB です。

次の項では、Oracle Databaseに格納されるOracleデータ型について説明します。これらのデータ型をリテラルとして指定する方法については、リテラルを参照してください。

文字データ型

文字データ型を使用すると、単語や自由形式のテキストなど、データベース文字セットまたは各国語文字セットの文字(英数字)を格納できます。文字データ型は、他のデータ型より制限が少ないため、プロパティも少なくなります。たとえば、文字データ型の列は、すべての英数字の値を格納できますが、NUMBER型の列が格納できるのは数値のみです。

文字データは、7ビットASCIIやEBCDICなど、データベース作成時に指定された文字セットの1つに対応しているバイト値で文字列に格納されます。Oracle Databaseは、シングルバイトの文字セットとマルチバイトの文字セットの両方をサポートします。

次のデータ型が、文字データに対して使用されます。

- CHARデータ型
- NCHARデータ型
- VARCHAR2データ型
- NVARCHAR2データ型

文字データ型をリテラルとして指定する方法については、テキスト・リテラルを参照してください。

CHARデータ型

CHARデータ型は、データベース文字セットの固定長の文字列を指定します。データベースの作成時には、データベース文字セットを指定します。

CHARの列を含む表の作成時には、sizeとして列長を指定し、その後にオプションで長さの修飾子を指定します。修飾子

BYTEはバイト長セマンティクスを表し、修飾子CHARは文字長セマンティクスを表します。バイト長セマンティクスでは、sizeは列に格納するバイト数です。文字長セマンティクスでは、sizeは列に格納するデータベース文字セットのコード・ポイント数です。コード・ポイントは、データベース文字セットおよびコード・ポイントによってエンコードされた特定の文字に応じて1から4バイトです。いずれかの長さ修飾子を指定して、列の目的の長さセマンティクスを明示的に指定することをお薦めします。修飾子を指定しない場合は、列を作成しているセッションのNLS_LENGTH_SEMANTICSパラメータの値によって長さセマンティクスが定義されますが、デフォルト・セマンティクスがBYTEであるスキーマSYSに表が属している場合を除きます。

Oracleでは、CHAR列に格納されているすべての値は、選択した長さセマンティクスのsizeによって指定された長さになります。列の長さより短い値を挿入すると、列の長さにあわせるため、その値の後に空白が埋め込まれます。列に対して長すぎる値を挿入しようとすると、Oracleはエラーを戻します。列長が文字(コード・ポイント)で表現されている場合、空白埋めでは、すべての列値が同じバイト長であることは保証されません。

sizeは列定義から省略できます。デフォルト値は1です。

sizeの最大値は2000で、選択した長さセマンティクスに応じて2000バイトまたは文字(コード・ポイント)を意味します。ただし、これとは関係なく、CHAR列に格納できる文字値の絶対最大長は2000バイトです。たとえば、列長を2000文字に定義した場合でも、1つ以上のコード・ポイントの幅が1バイトより大きい2000文字の値を挿入しようとすると、エラーが戻されます。文字のsizeの値は長さの制約であり、容量を保証するものではありません。CHAR列に常にデータベース文字セットのsize文字を格納できるようにするには、500以下のsizeの値を使用します。

異なる文字セットを持つデータベースおよびクライアント間で適切にデータを変換するには、CHARデータが正しい書式の文字列で構成されていることを確認してください。

関連項目:

文字セット・サポートの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。比較セマンティクスについては、データ型の比較規則を参照してください。

NCHARデータ型

NCHARデータ型は、各国語文字セットの固定長の文字列を指定します。データベースの作成時に、各国語文字セットをAL16UTF16またはUTF8として指定します。AL16UTF16およびUTF8は、Unicode文字セットの2つのエンコード方式(それぞれUTF-16およびCESU-8)であるため、NCHARはUnicodeのみのデータ型です。

NCHAR列を使用して表を作成するときは、列長をsize文字として指定するか、より正確に各国語文字セットのコード・ポイントとして指定します。1つのコード・ポイントは、コード・ポイントによってエンコードされた特定の文字に応じて、AL16UTF16では常に2バイト、UTF8では1から3バイトです。

Oracleでは、NCHAR列に格納されているすべての値は、size文字の長さになります。列の長さより短い値を挿入すると、列の長さにあわせるため、その値の後に空白が埋め込まれます。列に対して長すぎる値を挿入しようとすると、Oracleはエラーを戻します。各国語文字セットがUTF8の場合、空白埋めでは、すべての列値が同じバイト長であることは保証されません。

sizeは列定義から省略できます。デフォルト値は1です。

sizeの最大値は、各国語文字セットがAL16UTF16の場合は1000文字、UTF8の場合は2000文字です。ただし、これとは関係なく、NCHAR列に格納できる文字値の絶対最大長は2000バイトです。たとえば、列長を1000文字に定義した場合でも、各国語文字セットがUTF8ですべてのコード・ポイントが3バイト幅のときに1000文字の値を挿入しようとすると、エラーが戻されます。sizeの値は長さの制約であり、容量を保証するものではありません。NCHAR列に常に両方の各国語文字セットのsize文字を格納できるようにするには、666以下のsizeの値を使用します。

異なる文字セットを持つデータベースおよびクライアント間で適切にデータを変換するには、NCHARデータが正しい書式の文字列で構成されていることを確認してください。

CHAR値をNCHAR列に割り当てる場合、値はデータベース文字セットから各国語文字セットに暗黙的に変換されます。NCHAR値をCHAR列に割り当てる場合、値は各国語文字セットからデータベース文字セットに暗黙的に変換されます。NCHAR値の文字の一部をデータベース文字セットで表現できない場合、セッション・パラメータNLS_NCHAR_CONV_EXCPの値がTRUEのときは、エラーが戻されます。パラメータの値がFALSEの場合は、表現できない文字はデータベース文字セットのデフォルトの置換文字(通常は疑問符?または逆疑問符¿)に置き換えられます。

関連項目:

Unicodeデータ型のサポートについては、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください

VARCHAR2データ型

VARCHAR2データ型は、データベース文字セットの可変長の文字列を指定します。データベースの作成時には、データベース文字セットを指定します。

VARCHAR2の列を含む表の作成時には、sizeとして列長を指定し、その後にオプションで長さの修飾子を指定する必要があります。修飾子BYTEはバイト長セマンティクスを表し、修飾子CHARは文字長セマンティクスを表します。バイト長セマンティクスでは、sizeは列に格納できる最大バイト数です。文字長セマンティクスでは、sizeは列に格納できるデータベース文字セットの最大コード・ポイント数です。コード・ポイントは、データベース文字セットおよびコード・ポイントによってエンコードされた特定の文字に応じて1から4バイトです。いずれかの長さ修飾子を指定して、列の目的の長さセマンティクスを明示的に指定することをお薦めします。修飾子を指定しない場合は、列を作成しているセッションのNLS_LENGTH_SEMANTICSパラメータの値によって長さセマンティクスが定義されますが、デフォルト・セマンティクスがBYTEであるスキーマSYSに表が属している場合を除きます。

文字値は、列の長さを超えない場合、空白埋めなしに、指定したとおりにVARCHAR2列に格納されます。最大長を超える値を挿入しようとすると、Oracleはエラーを戻します。

sizeの最小値は1です。最大値は次のとおりです。

- 32767バイト(MAX_STRING_SIZE = EXTENDEDの場合)
- 4000バイト(MAX_STRING_SIZE = STANDARDの場合)

MAX_STRING_SIZE初期化パラメータと拡張データ型の内部記憶域メカニズムの詳細は、拡張データ型を参照してください。

sizeはバイトまたは文字(コード・ポイント)で表現できますが、VARCHAR2列に格納できる文字値の独立した絶対最大長は、MAX_STRING_SIZEに応じて32767または4000バイトです。たとえば、列長を32767文字に定義した場合でも、1つ以上のコード・ポイントの幅が1バイトより大きい32767文字の値を挿入しようとすると、エラーが戻されます。文字のsizeの値は長さの制約であり、容量を保証するものではありません。VARCHAR2列に常にデータベース文字セットのsize文字を格納できるようにするには、MAX_STRING_SIZE = EXTENDEDの場合は8191以下、MAX_STRING_SIZE = STANDARDの場合は1000以下のsizeの値を使用します。

Oracleは、非空白埋め比較セマンティクスを使用してVARCHAR2値を比較します。

異なる文字セットを持つデータベース間で適切にデータを変換するには、VARCHAR2データが正しい書式の文字列で構成されていることを確認してください。文字セット・サポートの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

関連項目:

比較セマンティクスについては、データ型の比較規則を参照してください。

VARCHARデータ型

VARCHARデータ型は、使用しないでください。かわりにVARCHAR2データ型を使用してください。現在、VARCHARデータ型はVARCHAR2データ型と同じ意味で使用されていますが、VARCHARデータ型は、異なる比較セマンティクスで比較される別の可変長文字列のデータ型に変更される予定です。

NVARCHAR2データ型

NVARCHAR2データ型は、各国語文字セットの可変長文字列を指定します。データベースの作成時に、各国語文字セットをAL16UTF16またはUTF8として指定します。AL16UTF16およびUTF8は、Unicode文字セットの2つのエンコード方式(それぞれUTF-16およびCESU-8)であるため、NVARCHAR2はUnicodeのみのデータ型です。

NVARCHAR2列を使用して表を作成するときは、列長を`size`文字として指定するか、より正確に各国語文字セットのコード・ポイントとして指定する必要があります。1つのコード・ポイントは、コード・ポイントによってエンコードされた特定の文字に応じて、AL16UTF16では常に2バイト、UTF8では1から3バイトです。

文字値は、列の長さを超えない場合、空白埋めなしに、指定したとおりにNVARCHAR2の列に格納されます。最大長を超える値を挿入しようとすると、Oracleはエラーを戻します。

`size`の最小値は1です。最大値は次のとおりです。

- `MAX_STRING_SIZE = EXTENDED`で各国語文字セットがAL16UTF16の場合は16383
- `MAX_STRING_SIZE = EXTENDED`で各国語文字セットがUTF8の場合は32767
- `MAX_STRING_SIZE = STANDARD`で各国語文字セットがAL16UTF16の場合は2000
- `MAX_STRING_SIZE = STANDARD`で各国語文字セットがUTF8の場合は4000

`MAX_STRING_SIZE`初期化パラメータと拡張データ型の内部記憶域メカニズムの詳細は、拡張データ型を参照してください。

文字の最大列長とは関係なく、NVARCHAR2列に格納できる値の絶対最大長は、`MAX_STRING_SIZE`に応じて32767または4000バイトです。たとえば、列長を16383文字に定義した場合でも、各国語文字セットがUTF8ですべてのコード・ポイントが3バイト幅のときに16383文字の値を挿入しようとすると、エラーが戻されます。`size`の値は長さの制約であり、容量を保証するものではありません。NVARCHAR2列に常に両方の各国語文字セットの`size`文字を格納できるようにするには、`MAX_STRING_SIZE = EXTENDED`の場合は10922以下、`MAX_STRING_SIZE = STANDARD`の場合は1333以下の`size`の値を使用します。

Oracleは、非空白埋め比較セマンティクスを使用してNVARCHAR2値を比較します。

異なる文字セットを持つデータベースおよびクライアント間で適切にデータを変換するには、NVARCHAR2データが正しい書式の文字列で構成されていることを確認してください。

VARCHAR2値をNVARCHAR2列に割り当てる場合、値はデータベース文字セットから各国語文字セットに暗黙的に変換されます。NVARCHAR2値をVARCHAR2列に割り当てる場合、値は各国語文字セットからデータベース文字セットに暗黙的に変換されます。NVARCHAR2値の文字の一部をデータベース文字セットで表現できない場合、セッション・パラメータ`NLS_NCHAR_CONV_EXCP`の値がTRUEのときは、エラーが戻されます。パラメータの値がFALSEの場合は、表現できない文字はデータベース文字セットのデフォルトの置換文字(通常は疑問符?または逆疑問符¿)に置き換えられます。

関連項目:

Unicodeデータ型のサポートについては、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

数値データ型

Oracle Databaseの数値データ型は、正と負の固定小数点数と浮動小数点数、0(ゼロ)、無限大、および操作の未定義の結果である値(非数値、つまりNAN)を格納します。数値データ型をリテラルとして指定する方法については、数値リテラルを参照してください。

NUMBERデータ型

NUMBERデータ型は、0(ゼロ)と、絶対値が1.0×10^{-130}以上1.0×10^{126}未満の範囲にある正と負の固定小数点数を格納します。1.0×10^{126}以上の絶対値を持つ算術式を指定した場合、Oracleはエラーを戻します。NUMBERの各値は1から22バイトである必要があります。

次の書式で固定小数点数を指定できます。

```
NUMBER(p,s)
```

説明:

- pは精度(precision)、つまり最大有効桁数(10進)です。最上位有効桁は最も左側のゼロ以外の桁で、最下位有効桁は最も右側の桁です。Oracleは、100進数で20桁までの精度で数値の移植性を保証します。これは小数点の位置によって39桁(10進)または40桁(10進)になります。
- sは位取り(scale)、つまり小数点から最下位有効桁までの桁数です。位取りの有効範囲は-84から127です。
    - 位取りが正の場合は、小数点の右側にある桁数で最下位有効桁を含んだ桁数になります。
    - 位取りが負の場合、小数点の左側にある桁数です。この場合は最下位有効桁を含みません。位取りが負の場合、実際のデータは整数部分の右から指定された桁数のみ丸められるため、最下位有効桁は小数点の左側になります。たとえば、(10,-2)と指定すると100の位まで丸められます。

e表記を使用する場合、位取りは通常、精度よりも大きくなります。位取りが精度より大きい場合、精度は小数点の右側にある最大有効桁数を示します。たとえば、NUMBER(4,5)として定義された列は、小数点の後の最初の桁が0(ゼロ)である必要があり、小数点以下5桁を超える値はすべて丸められます。

入力に対する特別な整合性チェックとして、固定小数点数列の位取りと精度を指定することをお薦めします。位取りと精度を指定しても、すべての値が固定長に強制されるわけではありません。値が精度の有効範囲を超えると、Oracleはエラーを戻します。値が位取りの有効範囲を超えると、Oracleはその値を丸めます。

次の書式で整数を指定できます。

```
NUMBER(p)
```

これは、精度がpで、位取りが0の固定小数点数です(NUMBER(p,0)と同じです)。

次の書式で浮動小数点を指定できます。

```
NUMBER
```

精度および位取りを指定しない場合、最大の範囲および精度をOracleの数値に指定したことになります。

関連項目:

浮動小数点数

表2-2に、異なる精度および位取りを使用したOracleのデータの格納方法を示します。

表2-2 精度および位取りの格納

実際のデータ	指定する精度と位取り	格納されるデータ
123.89	NUMBER	123.89
123.89	NUMBER(3)	124
123.89	NUMBER(3,2)	精度を超える
123.89	NUMBER(4,2)	精度を超える
123.89	NUMBER(5,2)	123.89
123.89	NUMBER(6,1)	123.9
123.89	NUMBER(6,-2)	100
.01234	NUMBER(4,5)	.01234
.00012	NUMBER(4,5)	.00012
.000127	NUMBER(4,5)	.00013
.0000012	NUMBER(2,7)	.0000012
.00000123	NUMBER(2,7)	.0000012
1.2e-4	NUMBER(2,5)	0.00012
1.2e-5	NUMBER(2,5)	0.00001

FLOATデータ型

FLOATデータ型は、NUMBERのサブタイプです。このデータ型は、精度とともに指定するか、または精度なしで指定でき、その定義はNUMBERの場合と同じであり、1から126の範囲で指定します。位取りは指定できませんが、データから解釈されます。FLOATの各値は1から22バイトである必要があります。

2進精度から10進精度に変換するには、nに0.30103を乗算します。10進精度から2進精度に変換するには、10進精度に3.32193を乗算します。2進精度の126桁は、10進精度の38桁とほぼ等しくなります。

NUMBERとFLOATの違いは、例を使用して説明します。次の例では、同じ値がNUMBER列とFLOAT列に挿入されます。

```
CREATE TABLE test (col1 NUMBER(5,2), col2 FLOAT(5));
INSERT INTO test VALUES (1.23, 1.23);
INSERT INTO test VALUES (7.89, 7.89);
INSERT INTO test VALUES (12.79, 12.79);
INSERT INTO test VALUES (123.45, 123.45);
SELECT * FROM test;
      COL1       COL2
---------- ----------
      1.23        1.2
      7.89        7.9
     12.79         13
    123.45        120
```

この例では、戻されるFLOATの値は、5桁(2進)を超えることはできません。5桁(2進)で表すことができる10進数の最大値は31です。最後の行には、31を超える10進値が含まれています。このため、FLOATの値は、その有効桁数が5桁(2進)を超えないように切り捨てられる必要があります。したがって、123.45は、2桁の有効桁数(10進)のみを持ち、4桁(2進)しか必要のない120に丸められます。

ANSIのFLOATデータを変換する際、Oracle Databaseは、OracleのFLOATデータ型を内部的に使用します。OracleのFLOATは使用可能ですが、BINARY_FLOATおよびBINARY_DOUBLEデータ型の方がより堅牢であるため、これらのデータ型を使用することをお薦めします。詳細は、浮動小数点数を参照してください。

浮動小数点数

浮動小数点数は、最初の桁から最後の桁までの任意の位置に小数点を置くことも、小数点を省略することもできます。指数は、数字の後に増加する桁数を表すときに、オプションで使用されます(たとえば、1.777 e^{-20})。小数点以下の桁数に制限はないため、浮動小数点数に対して位取りは指定できません。

2進浮動小数点数とNUMBERでは、Oracle Databaseによる内部的な値の格納方法が異なります。NUMBERの場合、値は10進精度を使用して格納されます。NUMBERでサポートされる範囲内および精度内のすべてのリテラルは、NUMBERとして正確に格納されます。これは、リテラルが10進精度(0から9)を使用して表されるためです。2進浮動小数点数は、2進精度(0および1)を使用して格納されます。このような格納スキームでは、10進精度を使用したすべての値を正確に表すことができません。値を10進精度から2進精度に変換するときに発生するエラーは、その値を2進精度から10進精度に戻すときには発生しない場合が多くあります。リテラル0.1はその一例です。

Oracle Databaseでは、浮動小数点数専用の2種類の数値データ型を提供しています。

BINARY_FLOAT

BINARY_FLOATは、32ビットの単精度浮動小数点数データ型です。BINARY_FLOATの各値には4バイトが必要です。

BINARY_DOUBLE

BINARY_DOUBLEは、64ビットの倍精度浮動小数点数データ型です。各BINARY_DOUBLEの値には8バイトが必要です。

NUMBER列では、浮動小数点数は10進精度を持ちます。BINARY_FLOAT列またはBINARY_DOUBLE列では、浮動小数点数は2進精度を持ちます。2進浮動小数点数では、特殊な値である無限大およびNaN(非数値)がサポートされます。

表2-3に示す制限内で、浮動小数点数を指定できます。浮動小数点数を指定する書式については、数値リテラルを参照してください。

表2-3 浮動小数点数の制限

値	BINARY_FLOAT	BINARY_DOUBLE
正の最大有限値	3.40282E+38F	1.79769313486231E+308
正の最小有限値	1.17549E-38F	2.22507485850720E-308

IEEE754への規格準拠

Oracleの浮動小数点データ型の実装は、2進浮動小数点算術の規格である電気電子学会(IEEE) 754-1985 (IEEE754)規格に準拠しています。浮動小数点データ型は次の点でIEEE754に準拠しています。

- SQLファンクションSQRTは平方根を実装します。「SQRT」を参照してください。
- SQLファンクションREMAINDERは余りを実装します。「REMAINDER」を参照してください。
- 算術演算子が準拠しています。「算術演算子」を参照してください。
- 比較演算子が準拠しています。ただし、NaNと比較する場合は除きます。Oracleは、NaNが他のすべての値より大きいとみなし、NaNとNaNは等しいと評価します。「浮動小数点条件」を参照してください。
- 変換演算子が準拠しています。「変換ファンクション」を参照してください。
- デフォルトの丸めモードがサポートされています。
- デフォルトの例外処理モードがサポートされています。
- 特殊な値であるINF、-INFおよびNaNがサポートされています。「浮動小数点条件」を参照してください。
- BINARY_FLOATおよびBINARY_DOUBLEの値を整数値のBINARY_FLOATおよびBINARY_DOUBLEに丸める方法として、SQLファンクションROUND、TRUNC、CEILおよびFLOORが提供されています。
- BINARY_FLOATおよびBINARY_DOUBLEを10進値に、10進値をBINARY_FLOATおよびBINARY_DOUBLEに丸める方法として、SQLファンクションTO_CHAR、TO_NUMBER、TO_NCHAR、TO_BINARY_FLOAT、TO_BINARY_DOUBLEおよびCASTが提供されています。

浮動小数点データ型は次の点ではIEEE754に準拠していません。

- -0は+0に強制変換されます。
- NaNとの比較はサポートされていません。
- すべてのNaN値は、BINARY_FLOAT_NANまたはBINARY_DOUBLE_NANのいずれかに強制変換されます。
- デフォルト以外の丸めモードはサポートされていません。
- デフォルト以外の例外処理モードはサポートされていません。

数値の優先順位

数値データ型をサポートする操作で、操作の引数が様々なデータ型を持つ場合、Oracleが使用するデータ型は、数値の優先順位によって判断されます。数値の優先順位が最も高いデータ型はBINARY_DOUBLEであり、その次がBINARY_FLOAT、最後がNUMBERとなります。したがって、複数の数値に対する操作では、次のようになります。

- オペランドのいずれかがBINARY_DOUBLEの場合、Oracleは操作の実行前に、すべてのオペランドを暗黙的にBINARY_DOUBLEに変換しようとします。
- オペランドにBINARY_DOUBLEはないが、いずれかがBINARY_FLOATの場合、Oracleは操作の実行前に、すべてのオペランドを暗黙的にBINARY_FLOATに変換しようとします。
- それ以外の場合は、Oracleは操作の実行前に、すべてのオペランドをNUMBERに変換しようとします。

暗黙的な変換が必要な場合に変換に失敗すると、操作は実行されません。暗黙的な変換の詳細は、表2-8を参照してください。

他のデータ型のコンテキストでは、数値データ型の優先順位は、日時データ型と期間データ型よりも低く、文字データ型とその他のすべてのデータ型よりも高くなります。

LONGデータ型

LONG列を使用した表を作成しないでください。かわりに、LOB列(CLOB、NCLOBまたはBLOB)を使用してください。LONG列は、下位互換性のためにサポートされています。

LONG列には、2GB(2^{31})から1を引いたバイト数までの可変長の文字列を格納できます。LONG列には、多くの点でVARCHAR2列と同じ特長があります。LONG列を使用すると、長いテキスト文字列を格納できます。LONG値の長さは、ご使用のコンピュータで利用できるメモリーによって制限される場合もあります。LONGリテラルは、テキスト・リテラルの説明のような形式になります。

既存のLONG列もLOB列に変換することをお薦めします。LOB列は、LONG列ほど制限は多くありません。LOB機能はリリースごとに拡張されていますが、LONG機能は最近のリリースでは変更されていません。LONG列からLOB列への変換については、「ALTER TABLE」のmodify_col_properties句および「TO_LOB」を参照してください。

SQL文の中の次の場所でLONG列を参照できます。

- SELECTリスト
- UPDATE文のSET句
- INSERT文のVALUES句

LONG値を使用する場合には、次の制限があります。

- 表には複数のLONG列を含めることはできません。
- LONG属性を持つオブジェクトは作成できません。
- LONG列は、WHERE句または整合性制約では指定できません(NULLおよびNOT NULL制約は除く)。
- LONG列に索引を付けることはできません。
- LONGデータは正規表現では指定できません。
- ストアド・ファンクションはLONG値を戻すことはできません。
- LONGデータ型を使用して、PL/SQLプログラム・ユニットの変数または引数を宣言できます。ただし、そのプログラム・ユニットは、SQLからコールできません。
- 単一のSQL文に指定する、すべてのLONG列、更新された表、ロックされた表は、同一データベース上にある必要があります。
- LONG列およびLONG RAW列は、分散型のSQL文で使用できません。また、レプリケートできません。

- LONGとLOBの両方の列を持つ表では、1つのSQL文でLONGとLOB列の両方に4000バイトより大きいデータをバインドすることはできません。ただし、LONG列またはLOB列のいずれか一方には、4,000バイトを超えるデータをバインドできます。

また、LONG列はSQL文の次のような部分では使用できません。

- GROUP BY句、ORDER BY句、CONNECT BY句またはSELECT文にあるDISTINCT演算子
- SELECT文のUNIQUE演算子
- CREATE CLUSTER文の列リスト
- CREATE MATERIALIZED VIEW文のCLUSTER句
- SQL組込みファンクション、式または条件
- GROUP BY句を含む問合せのSELECT構文のリスト
- UNION、INTERSECTまたはMINUS集合演算子によって結合されている副問合せまたは問合せのSELECT構文のリスト
- CREATE TABLE ... AS SELECT文のSELECT構文のリスト
- ALTER TABLE ... MOVE文
- INSERT文の副問合せのSELECT構文のリスト

トリガーでは、LONGデータ型は次のように使用されます。

- トリガー内のSQL文で、データをLONG列に挿入できます。
- LONG列のデータをCHARやVARCHAR2などの制約があるデータ型に変換できる場合は、トリガー内のSQL文でLONG列を参照できます。
- トリガー内の変数は、LONGデータ型を使用して宣言できません。
- :NEWと:OLDはLONG列で使用できません。

Oracle Call Interfaceを使用して、データベースからLONG値の一部を検索できます。

関連項目:

Oracle Call Interfaceプログラマーズ・ガイド

日時および間隔のデータ型

日時データ型には、DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONEおよびTIMESTAMP WITH LOCAL TIME ZONEがあります。日時データ型の値は、「日時」とも呼ばれます。期間データ型には、INTERVAL YEAR TO MONTHおよびINTERVAL DAY TO SECONDがあります。期間データ型の値は、「期間」とも呼ばれます。日時値と期間値をリテラルとして表す方法の詳細は、日時リテラルおよび期間リテラルを参照してください。

DatatimesとIntervalsは、どちらも複数のフィールドで構成されます。これらのフィールドの値は、データ型の値によって決まります。表2-4に、日時フィールド、および日時と期間の有効な値を示します。

日時データのDML操作で正しい結果を得るには、組込みSQLファンクションDBTIMEZONEおよびSESSIONTIMEZONEで問い合せることによって、データベースおよびセッションのタイムゾーンを確認します。タイムゾーンを手動で設定していない場合、

Oracle Databaseは、オペレーティング・システムのタイムゾーンをデフォルトで使用します。オペレーティング・システムのタイムゾーンがOracleで有効でない場合は、Oracleは、協定世界時(UTC)(以前のグリニッジ標準時)をデフォルトの値として使用します。

表2-4 日時フィールド、および日時と期間の値

日時フィールド	日時に有効な値	期間に有効な値
YEAR	-4712 から 9999 (0 を除きます)	正または負のすべての整数
MONTH	01 から 12	0 から 11
DAY	01 から 31(現在の NLS カレンダ・パラメータに従った MONTH および YEAR の値の範囲内)	正または負のすべての整数
HOUR	00 から 23	0 から 23
MINUTE	00 から 59	0 から 59
SECOND	00 から 59.9(n)(「9(n)」は秒の小数部の精度。)DATE には適用されません。	00 から 59.9(n)(「9(n)」は秒の小数部の期間の精度)
TIMEZONE_HOUR	-12 から 14(この範囲は夏時間の変更を保存します。)DATE または TIMESTAMP には適用されません。	該当なし
TIMEZONE_MINUTE (表の後のノートを参照)	00 から 59。DATE または TIMESTAMP には適用されません。	該当なし
TIMEZONE_REGION	V$TIMEZONE_NAMES データ・ディクショナリ・ビュー内の TZNAME 列を問い合せます。DATE または TIMESTAMP には適用されません。すべてのタイムゾーン地域名のリストは、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。	該当なし
TIMEZONE_ABBR	V$TIMEZONE_NAMES データ・ディクショナリ・ビュー内の TZABBREV 列を問い合せます。DATE または TIMESTAMP には適用されません。	該当なし

ノート:

TIMEZONE_HOUR および TIMEZONE_MINUTE は同時に指定され、+|- hh:mi の形式(-12:59 から +14:00 までの値を使用)で 1 つのエンティティとして解釈されます。API にタイムゾーン値を指定する方法の詳細は、『Oracle Data Provider for .NET 開発者ガイド for Microsoft Windows』を参照してください。

DATEデータ型

DATEデータ型は、日付および時刻の情報を格納するために使用します。日付および時刻の情報は、文字データ型および数値データ型で表現できますが、DATEデータ型には特別に対応付けられているプロパティがあります。各DATE値には、年、月、日、時、分および秒の情報が格納されます。

DATE値をリテラルに指定するか、文字値や数値をTO_DATEファンクションによって日付値に変換できます。これらの方法でのDATE値の表し方の例は、日時リテラルを参照してください。

ユリウス日の使用方法

ユリウス日は、紀元前4712年1月1日から経過した日数です。ユリウス日によって共通の基準で日付を算定できます。日付ファンクションTO_DATEとTO_CHARで日付書式モデル「J」を使用して、OracleのDATE値とユリウス日の間で変換を行うことができます。

ノート:

Oracle Database では、ユリウス日の計算に天文学方式を使用しています。この方式では、紀元前 4713 年は -4712 として計算されます。これに対し、歴史学方式では、紀元前 4713 年は-4713 として計算されます。Oracle のユリウス日を、歴史学方式で計算した値と比較する場合には、紀元前の日付に 365 日の違いがあることに注意してください。詳細は、http://aa.usno.navy.mil/faq/docs/millennium.php を参照してください。

デフォルトの日付値は次のように決まります。

- 年は現在の年で、SYSDATEで戻されます。
- 月は現在の月で、SYSDATEで戻されます。
- 日は01、つまり月の最初の日になります。
- 時、分、秒はすべて0です。

これらのデフォルト値は、次の例(5月に発行)のように、日付が指定されていない場合に日付値を要求する問合せで使用されます。

```
SELECT TO_DATE('2009', 'YYYY')
  FROM DUAL;
TO_DATE('
---------
01-MAY-09
```

例

次の文は、2009年1月1日に相当するユリウス日を戻します。

```
SELECT TO_CHAR(TO_DATE('01-01-2009', 'MM-DD-YYYY'),'J')
    FROM DUAL;
TO_CHAR
-------
2454833
```

関連項目:

DUAL表については、DUAL表からの選択を参照してください。

TIMESTAMPデータ型

TIMESTAMPデータ型は、DATEデータ型の拡張機能です。DATEデータ型の年、月および日に加えて、時、分および秒の値を格納します。このデータ型は、正確な時刻値の格納と、複数の地域にまたがる日時情報の収集および評価に有効です。TIMESTAMPデータ型は次のように指定します。

```
TIMESTAMP [(fractional_seconds_precision)]
```

fractional_seconds_precisionには、オプションで、Oracleが格納する桁数を、SECOND日時フィールドの小数部まで指定します。このデータ型の列を作成する場合、値は0から9までの範囲内の数に設定できます。デフォルトは6です。

関連項目:

文字データからTIMESTAMPデータへの変換については、「TO_TIMESTAMP」を参照してください。

TIMESTAMP WITH TIME ZONEデータ型

TIMESTAMP WITH TIME ZONEは、タイムゾーン地域名またはタイムゾーン・オフセットを値に含むTIMESTAMPの変形です。タイムゾーン・オフセットは、ローカルの時刻とUTC(時および分)との差異です。このデータ型は、ローカル・タイムゾーン情報を保持する場合に有効です。

TIMESTAMP WITH TIME ZONEデータ型は、次のように指定します。

```
TIMESTAMP [(fractional_seconds_precision)] WITH TIME ZONE
```

fractional_seconds_precisionには、オプションで、Oracleが格納する桁数を、SECOND日時フィールドの小数部まで指定します。このデータ型の列を作成する場合、値は0から9までの範囲内の数に設定できます。デフォルトは6です。

Oracleのタイムゾーン・データは、http://www.iana.org/time-zones/で入手できるパブリック・ドメイン情報から導出されます。Oracleのタイムゾーン・データは、このサイトで入手できる最新のデータを反映していない場合があります。

関連項目:

- Oracleのタイムゾーン・データの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。
- 夏時間のサポートについては、「夏時間のサポート」および表2-19を参照してください。
- 文字データからTIMESTAMP WITH TIME ZONEデータへの変換については、「TO_TIMESTAMP_TZ」を参照してください。
- ERROR_ON_OVERLAP_TIMEセッション・パラメータについては、「ALTER SESSION」を参照してください。

TIMESTAMP WITH LOCAL TIME ZONEデータ型

TIMESTAMP WITH LOCAL TIME ZONEは、TIMESTAMPのもう1つの変形であり、タイムゾーン情報を識別します。これは、TIMESTAMP WITH TIME ZONEとは異なり、データベースに格納されるデータはデータベース・タイムゾーンに対して正規

化され、タイムゾーン情報は列データの一部として格納されません。ユーザーがデータを検索すると、Oracleはユーザーのローカル・セッション・タイムゾーンのデータを戻します。このデータ型は、2層アプリケーションでクライアント・システムのタイムゾーンの日時情報を常に表示する場合に有効です。

TIMESTAMP WITH LOCAL TIME ZONEデータ型は、次のように指定します。

```
TIMESTAMP [(fractional_seconds_precision)] WITH LOCAL TIME ZONE
```

fractional_seconds_precisionには、オプションで、Oracleが格納する桁数を、SECOND日時フィールドの小数部まで指定します。このデータ型の列を作成する場合、値は0から9までの範囲内の数に設定できます。デフォルトは6です。

Oracleのタイムゾーン・データは、http://www.iana.org/time-zones/で入手できるパブリック・ドメイン情報から導出されます。Oracleのタイムゾーン・データは、このサイトで入手できる最新のデータを反映していない場合があります。

関連項目:

- Oracleのタイムゾーン・データの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。
- このデータ型の使用例は、『Oracle Database開発ガイド』を参照してください。文字データからTIMESTAMP WITH LOCAL TIME ZONEデータへの変換については、CASTを参照してください。

INTERVAL YEAR TO MONTHデータ型

INTERVAL YEAR TO MONTHは、YEARおよびMONTH日時フィールドを使用して期間を格納します。このデータ型は、年および月の値のみが重要な場合に、2つの日時の値の正確な違いを表す場合に有効です。

INTERVAL YEAR TO MONTHは、次のように指定します。

```
INTERVAL YEAR [(year_precision)] TO MONTH
```

year_precisionは、YEAR日時フィールドの桁数です。year_precisionのデフォルト値は2です。

期間値をリテラルとして指定すると、高い柔軟性が得られます。期間値をリテラルとして指定する方法の詳細は、期間リテラルを参照してください。期間の使用例は、日時および期間の例を参照してください。

INTERVAL DAY TO SECONDデータ型

INTERVAL DAY TO SECONDデータ型には、日、時間、分および秒による期間が格納されます。このデータ型は、2つの日時の値の正確な違いを表す場合に有効です。

このデータ型は次のように指定します。

```
INTERVAL DAY [(day_precision)]
   TO SECOND [(fractional_seconds_precision)]
```

説明:

- day_precisionは、DAY日時フィールドの桁数です。0から9までの値を使用できます。デフォルトは2です。
- fractional_seconds_precisionは、SECOND日時フィールドの小数部の桁数です。0から9までの値を使用できます。デフォルトは6です。

期間値をリテラルとして指定すると、高い柔軟性が得られます。期間値をリテラルとして指定する方法の詳細は、期間リテラルを参照してください。期間の使用例は、日時および期間の例を参照してください。

日時および期間の演算

日付(DATE)、タイムスタンプ(TIMESTAMP、TIMESTAMP WITH TIME ZONEおよびTIMESTAMP WITH LOCAL TIME ZONE)および期間(INTERVAL DAY TO SECONDおよびINTERVAL YEAR TO MONTH)データに対して、様々な演算処理を実行できます。Oracleは、次の規則に基づいて結果を計算します。

- 日付およびタイムスタンプ値の演算処理ではNUMBER定数を使用できますが、期間値では使用できません。Oracleは、タイムスタンプ値を内部的に日付値に変換し、日時の演算で使用されるNUMBER定数と期間式を日数として解析します。たとえば、SYSDATE+ 1は明日です。SYSDATE -7は1週間前です。SYSDATE + (10/1440)は10分後です。SYSDATEからemployeesサンプル表のhire_date列を引くと、各従業員が雇用されてから経過した日数が戻ります。日付値またはタイムスタンプ値の乗算や除算はできません。
- Oracleは、BINARY_FLOATオペランドおよびBINARY_DOUBLEオペランドを暗黙的にNUMBERに変換します。
- 各DATE値には時刻コンポーネントが含まれるため、多くの場合、日付操作の結果には小数部が含まれます。この小数部は、日を単位として表されています。たとえば、1.5日は36時間です。小数部は、DATEデータの一般的な操作を行うOracle組込みファンクションによっても戻されます。たとえば、MONTHS_BETWEENファンクションは、2つの日付の間の月数を戻します。結果の小数部は、月(1か月は31日)を単位として表されます。
- 1つのオペランドがDATE値または数値であり、タイムゾーンおよび小数部コンポーネントのいずれも含まない場合、次のようになります。
    - Oracleは、他方のオペランドを暗黙的にDATEデータに変換します。ただし、数値と期間を掛けて期間を戻す乗算を行う場合は除きます。
    - 他方のオペランドがタイムゾーン値を持つ場合、Oracleは戻り値でセッション・タイムゾーンを使用します。
    - 他方のオペランドが小数部の値を持つ場合、その小数部の値は失われます。
- DATEデータ型専用に設計された組込みファンクションにタイムスタンプ値、期間値または数値を渡すと、OracleはDATE値以外の値を暗黙的にDATE値に変換します。DATEへの暗黙的な変換を実行するファンクションの詳細は、日時ファンクションを参照してください。
- 期間の計算が日時の値を戻す場合、その結果は実際の日時の値である必要があります。実際の日時の値でない場合、データベースはエラーを戻します。たとえば、次の2つの文はエラーを戻します。

```
SELECT TO_DATE('31-AUG-2004','DD-MON-YYYY') + TO_YMINTERVAL('0-1')
  FROM DUAL;
SELECT TO_DATE('29-FEB-2004','DD-MON-YYYY') + TO_YMINTERVAL('1-0')
  FROM DUAL;
```

最初の文は、1か月31日の月に1か月が追加され、9月31日となります。これは有効な日付ではないため、この文は失敗します。2つ目の文は、4年に一度のみ存在する日付に1年を追加する計算は無効であるため、失敗します。ただし、2月29日に4年を追加する計算は有効であるため、次の文は成功します。

```
SELECT TO_DATE('29-FEB-2004', 'DD-MON-YYYY') + TO_YMINTERVAL('4-0')
  FROM DUAL;

TO_DATE('
---------
29-FEB-08
```

- Oracleは、すべてのタイムスタンプの演算をUTC時間で実行します。TIMESTAMP WITH LOCAL TIME ZONEでは、Oracleは、日時の値をデータベース・タイムゾーンからUTCに変換し、演算を実行した後にデータベース・タイムゾーンに変換しなおします。TIMESTAMP WITH TIME ZONEでは、日時の値は常にUTCであるため、変換は必要

ありません。

表2-5に、日時の演算処理のマトリックスを示します。ダッシュはサポートされていない処理を表します。

表2-5 日時の演算のマトリックス

オペランドおよび演算子	DATE	TIMESTAMP	INTERVAL	数値
DATE				
+	–	–	DATE	DATE
-	NUMBER	INTERVAL	DATE	DATE
*	–	–	–	–
/	–	–	–	–
TIMESTAMP				
+	–	–	TIMESTAMP	DATE
-	INTERVAL	INTERVAL	TIMESTAMP	DATE
*	–	–	–	–
/	–	–	–	–
INTERVAL				
+	DATE	TIMESTAMP	INTERVAL	–
-	–	–	INTERVAL	–
*	–	–	–	INTERVAL
/	–	–	–	INTERVAL
数値				
+	DATE	DATE	–	NA
-	–	–	–	NA
*	–	–	INTERVAL	NA
/	–	–	–	NA

例

期間値の式を開始時間に追加できます。order_date列を持つサンプル表oe.ordersについて考えます。次の文は、order_date列の値に30日を加算します。

```
SELECT order_id, order_date + INTERVAL '30' DAY AS "Due Date"
  FROM orders
  ORDER BY order_id, "Due Date";
```

夏時間のサポート

Oracle Databaseは、指定したタイムゾーン地域に、夏時間が適用されているかを自動的に判断し、それに応じてローカル時刻の値を戻します。日時の値は、境界を除いたすべての指定した地域において、夏時間が適用されているかをOracleが判断するために有効です。夏時間の開始または終了時に、境界が発生します。たとえば、米国の太平洋地域では、夏時間の開始時、時刻は午前2時から午前3時に変更されます。午前2時と午前3時の間の1時間は存在しません。夏時間の終了時、時刻は午前2時から午前1時に変更されます。午前1時と午前2時の間の1時間は繰り返されます。

このような境界を解決するために、OracleはTZRおよびTZD書式要素を使用します。詳細は、表2-19を参照してください。TZRは、日時の入力文字列でタイムゾーン地域名を表します。たとえば、'Australia/North'、'UTC'および'Singapore'などです。TZDは、夏時間情報を含むタイムゾーン地域名の略称書式です。たとえば、米国/太平洋標準時はPST、米国/太平洋夏時間はPDTなどです。TZRおよびTZD書式要素の値を表示するには、V$TIMEZONE_NAMES動的パフォーマンス・ビューのTZNAMEおよびTZABBREV列に問合せを実行してください。

> ノート:
>
> 夏時間機能には、タイムゾーン地域名が必要です。この名前は、大小 2 つのタイムゾーン・ファイルに格納されます。これらのファイルのうち、使用する環境および使用する Oracle Database のリリースに応じて、いずれか一方がデフォルトのファイルになります。タイムゾーン・ファイルおよびタイムゾーン名の詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

両方のファイル内のすべてのタイムゾーン地域名のリストは、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

Oracleのタイムゾーン・データは、http://www.iana.org/time-zones/で入手できるパブリック・ドメイン情報から導出されます。Oracleのタイムゾーン・データは、このサイトで入手できる最新のデータを反映していない場合があります。

関連項目:

- 書式要素の詳細は、日時書式モデルを参照してください。また、セッション・パラメータERROR_ON_OVERLAP_TIMEの説明も参照してください。
- Oracleのタイムゾーン・データの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。
- 動的パフォーマンス・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

日時および期間の例

次の例では、日時および期間データ型の指定方法を示します。

```
CREATE TABLE time_table
```

```
  (start_time    TIMESTAMP,
   duration_1    INTERVAL DAY (6) TO SECOND (5),
   duration_2    INTERVAL YEAR TO MONTH);
```

start_time列は、TIMESTAMP型です。TIMESTAMPの暗黙的な小数部の精度は6です。

duration_1列は、INTERVAL DAY TO SECOND型です。DAYフィールドの最大桁数は6です。また、小数部の最大桁数は5です(その他のすべての日時フィールドの最大桁数は2です)。

duration_2列は、INTERVAL YEAR TO MONTH型です。各フィールド(YEARおよびMONTH)の値の最大桁数は2です。

期間データ型には書式モデルはありません。そのため、これらの表示を調整するには、EXTRACTなどの文字ファンクションを結合させて要素を連結させる必要があります。たとえば、次の例では、hr.employeesとoe.orders表にそれぞれ問合せを行い、期間出力の書式を"yy-mm"から"yy years mm months"に、および"dd-hh"から"dddd days hh hours"に変更します。

```
SELECT last_name, EXTRACT(YEAR FROM (SYSDATE - hire_date) YEAR TO MONTH)
       || ' years '
       || EXTRACT(MONTH FROM (SYSDATE - hire_date) YEAR TO MONTH)
       || ' months'  "Interval"
  FROM employees;
LAST_NAME                 Interval
------------------------- --------------------
OConnell                  2 years 3 months
Grant                     1 years 9 months
Whalen                    6 years 1 months
Hartstein                 5 years 8 months
Fay                       4 years 2 months
Mavris                    7 years 4 months
Baer                      7 years 4 months
Higgins                   7 years 4 months
Gietz                     7 years 4 months
. . .
SELECT order_id, EXTRACT(DAY FROM (SYSDATE - order_date) DAY TO SECOND)
       || ' days '
       || EXTRACT(HOUR FROM (SYSDATE - order_date) DAY TO SECOND)
       || ' hours' "Interval"
  FROM orders;
  ORDER_ID Interval
---------- --------------------
      2458 780 days 23 hours
      2397 685 days 22 hours
      2454 733 days 21 hours
      2354 447 days 20 hours
      2358 635 days 20 hours
      2381 508 days 18 hours
      2440 765 days 17 hours
      2357 1365 days 16 hours
      2394 602 days 15 hours
      2435 763 days 15 hours
. . .
```

RAWデータ型とLONG RAWデータ型

RAWデータ型とLONG RAWデータ型には、異なるシステム間でデータを移動する際にOracle Databaseによって明示的に変換されないデータが格納されます。これらのデータ型は、2進データおよびバイト列のために用意されています。たとえば、LONG RAWは、図形、音声、文書、またはバイナリ・データの配列の格納に使用できますが、解析方法は用途によって異なります。

LONG RAW列をバイナリLOB(BLOB)へ変換することをお薦めします。LOB列は、LONG列ほど制限は多くありません。詳細は、「TO_LOB」を参照してください。

RAWは、VARCHAR2と同様に可変長データ型ですが、Oracle Net(クライアント・ソフトウェアとデータベース、またはデータベースとデータベースを接続します)およびOracleのインポート/エクスポート・ユーティリティは、RAWまたはLONG RAWデータの転送時に文字変換を行いません。これに対し、Oracle NetおよびOracleのインポート/エクスポート・ユーティリティは、自動的にCHAR、VARCHAR2およびLONGデータをあるデータベース文字セットから別のデータベース文字セットに変換するか(データがデータベース間で送信される場合)、またはデータベース文字セットからクライアント文字セットに変換します(データがデータベースとクライアントの間で送信される場合)。クライアント文字セットは、クライアント・インタフェースのタイプ(OCIやJDBCなど)およびクライアント構成(たとえば、NLS_LANG環境変数)によって決定されます。

OracleがRAWまたは LONG RAWデータを文字データに暗黙的に変換する場合、結果として生成される文字値は、バイナリ入力を16進表記で表したものになり、各文字はRAWデータの連続した4つのビットを表す16進数(0-9、A-F)になります。たとえば、ビット列が11001011で表示される1バイトのRAWデータは、CBという値になります。

Oracleは、文字データをRAWまたはLONG RAWに暗黙的に変換する場合、連続した入力文字をバイナリ・データの4つの連続したビットを16進表記で表したものとして解釈し、これらのビットを連結してRAWまたはLONG RAW値を生成します。入力文字の中に16進数(0-9、A-F、a-f)以外の文字がある場合は、エラーが報告されます。文字数が奇数の場合、結果は未定義になります。

SQLファンクションRAWTOHEXおよびHEXTORAWは、前述の暗黙的な変換と同等の明示的な変換を行います。オラクル社が提供するPL/SQLパッケージ(UTL_RAWおよびUTL_I18N)のファンクションを使用すると、RAWおよび文字データ間の他のタイプの変換を行うことができます。

ラージ・オブジェクト(LOB)・データ型

組込みLOBデータ型のBLOB、CLOB、NCLOB(内部ファイルに格納)およびBFILE(外部ファイルに格納)には、構造化されていない大きいデータ(text、image、video、spatial dataなど)を格納できます。BLOB、CLOBおよびNCLOBデータの最大サイズは、2^{32}から1を引いたバイト数にLOB記憶域のCHUNKパラメータの値を掛けた値です。データベースの表領域が標準のブロック・サイズで、LOB列を作成したときにLOB記憶域のCHUNKパラメータのデフォルト値を使用した場合には、前述の値は2^{32}バイトから1を引いた値にデータベース・ブロック・サイズを掛けた値に等しくなります。BFILEデータの最大サイズは、2^{64}バイトから1を引いた値ですが、この最大サイズはオペレーティング・システムによって制限される場合があります。

表を作成するときに、LOB列またはLOBオブジェクト属性に、オプションで表に指定したものとは異なる表領域および記憶特性を指定できます。

LOB列の作成時に行の記憶域を使用可能にしている場合、CLOB、NCLOBおよびBLOBの値は、約4000バイトを上限として表内に格納されます。4000バイトを超えるLOBは、常に外部ファイルに格納されます。詳細は、「ENABLE STORAGE IN ROW」を参照してください。

LOB列には、内部のLOB値(データベース内)または外部のLOB値(データベース外)を参照できるLOBロケータが含まれています。表からLOBを選択すると、実際にはLOBのロケータが戻され、LOB値全体は戻されません。LOBに対するDBMS_LOBパッケージとOracle Call Interface(OCI)の操作は、これらのロケータを介して行われます。

LOBは、LONG型およびLONG RAW型と似ていますが、次の点で異なります。

- LOBは、オブジェクト型(ユーザー定義のデータ型)の属性に指定できます。
- LOBロケータは、表の列に格納されます。実際のLOB値は、表の列に格納される場合と格納されない場合があります。BLOB、NCLOBおよびCLOBの値は、別々の表領域に格納されます。BFILEデータは、サーバー上の外部ファイルに格納されます。
- LOB列にアクセスしたときに戻されるのはロケータです。

- LOBの最大サイズは、2^{32}から1を引いたバイト数にデータベース・ブロック・サイズを掛けた値です。BFILEデータの最大サイズは、2^{64}バイトから1を引いた値ですが、この最大サイズはオペレーティング・システムによって制限される場合があります。
- LOBでは、効果的かつランダムなピース単位のデータ・アクセスおよび操作が可能です。
- 1つの表内に2つ以上のLOB列を定義できます。
- NCLOBの例外を除いて、1つのオブジェクトに1つ以上のLOB属性を定義できます。
- LOBバインド変数を宣言できます。
- LOB列とLOB属性を選択できます。
- 1つ以上のLOB列または1つ以上のLOB属性を持つオブジェクトが含まれている新しい行を挿入したり、既存の行を更新できます。更新操作では、内部LOB値をNULL、つまり空に設定したり、LOB全体をデータに置き換えられます。BFILEは、NULLに設定したり、別のファイルを指すように設定できます。
- LOB行と列の交差部またはLOB属性を、別のLOB行と列の交差部またはLOB属性を使用して更新できます。
- LOB列またはLOB属性が含まれている行を削除できます(これによってLOB値も削除されます)。BFILEの場合、実際のオペレーティング・システム・ファイルは削除されません。

INSERT文またはUPDATE文を発行するだけで、表内LOB列(データベースに格納されているLOB列)またはLOB属性(データベースに格納されているオブジェクト型列の属性)の行にアクセスして移入することができます。

LOB列の制限事項

LOB列には次のルールと制限があります。詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。

関連項目:

- これらのインタフェースおよびLOBの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』および『Oracle Call Interfaceプログラマーズ・ガイド』を参照してください。
- LONG列からLOB列への変換については、「ALTER TABLE」のmodify_col_properties句および「TO_LOB」を参照してください。

BFILEデータ型

BFILEデータ型を使用すると、Oracle Database外のファイル・システムに格納されているバイナリ・ファイルLOBにアクセスできます。BFILE列または属性には、サーバーのファイル・システム上のバイナリ・ファイルに対するポインタとして機能する、BFILEロケータが格納されます。ロケータには、ディレクトリ名とファイル名が保持されます。

BFILENAMEファンクションを使用すると、実表のデータに影響を与えずにBFILEのファイル名およびパスを変更できます。この組込みSQLファンクションの詳細は、「BFILENAME」を参照してください。

バイナリ・ファイルLOBは、トランザクションには関係なく、リカバリができません。ファイルの統合性と耐久性を提供しているのは基本にあるオペレーティング・システムです。BFILEデータの最大サイズは、2^{64}バイトから1を引いた値ですが、この最大サイズはオペレーティング・システムによって制限される場合があります。

データベース管理者は、外部ファイルが存在し、Oracleのプロセスがファイルに対するオペレーティング・システムの読取り権限を持っていることを確認する必要があります。

BFILEデータ型を使用すると、サイズが大きいバイナリ・ファイルの読取り専用のサポートが有効になります。この場合、ファイルを修正またはレプリケートすることはできません。Oracleでは、ファイル・データにアクセスするためのAPIが提供されています。ファイル・データにアクセスするために使用する主なインタフェースは、DBMS_LOBパッケージとOracle Call Interface(OCI)です。

関連項目:

LOBの詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』および『Oracle Call Interfaceプログラマーズ・ガイド』を参照してください。また、「CREATE DIRECTORY」も参照してください。

BLOBデータ型

BLOBデータ型は、構造化されていないバイナリ・ラージ・オブジェクトを格納するために使用します。BLOBオブジェクトは、文字セットのセマンティクスを持たないビットストリームとして考えることができます。BLOBオブジェクトには、4GBから1を引いたバイト数にLOB記憶域のCHUNKパラメータの値を掛けた値のサイズまでのバイナリ・データを格納できます。データベースの表領域が標準のブロック・サイズで、LOB列を作成したときにLOB記憶域のCHUNKパラメータのデフォルト値を使用した場合には、前述の値は4GBから1を引いた値にデータベース・ブロック・サイズを掛けた値に等しくなります。

BLOBオブジェクトでは、トランザクションが完全にサポートされます。SQL、DBMS_LOBパッケージまたはOracle Call Interface(OCI)を介して行った変更は、すべてトランザクションに反映されます。BLOB値の操作は、コミットおよびロールバックできます。ただし、1つのトランザクションのPL/SQLまたはOCI変数をBLOBロケータに保存し、そのロケータを別のトランザクションまたはセッションで使用することはできません。

CLOBデータ型

CLOBデータ型は、シングルバイトおよびマルチバイト文字データを格納するために使用します。固定幅および可変幅の文字セットがサポートされます。両方の文字セットでデータベース文字セットを使用します。CLOBオブジェクトには、4GBから1を引いたバイト数にLOB記憶域のCHUNKパラメータの値を掛けた値のサイズまでの文字データを格納できます。データベースの表領域が標準のブロック・サイズで、LOB列を作成したときにLOB記憶域のCHUNKパラメータのデフォルト値を使用した場合には、前述の値は4GBから1を引いた値にデータベース・ブロック・サイズを掛けた値に等しくなります。

CLOBオブジェクトでは、トランザクションが完全にサポートされます。SQL、DBMS_LOBパッケージまたはOracle Call Interface(OCI)を介して行った変更は、すべてトランザクションに反映されます。CLOB値の操作は、コミットおよびロールバックできます。ただし、1つのトランザクションのPL/SQLまたはOCI変数をCLOBロケータに保存し、そのロケータを別のトランザクションまたはセッションで使用することはできません。

NCLOBデータ型

NCLOBデータ型は、Unicodeデータを格納するために使用します。固定幅および可変幅の文字セットがサポートされます。両方の文字セットで各国語文字セットを使用します。NCLOBオブジェクトには、4GBから1を引いたバイト数にLOB記憶域のCHUNKパラメータの値を掛けた値のサイズまでの文字テキスト・データを格納できます。データベースの表領域が標準のブロック・サイズで、LOB列を作成したときにLOB記憶域のCHUNKパラメータのデフォルト値を使用した場合には、前述の値は4GBから1を引いた値にデータベース・ブロック・サイズを掛けた値に等しくなります。

NCLOBオブジェクトでは、トランザクションが完全にサポートされます。SQL、DBMS_LOBパッケージまたはOCIを介して行った変更は、すべてトランザクションに反映されます。NCLOB値の操作は、コミットおよびロールバックできます。ただし、1つのトランザクションのPL/SQLまたはOCI変数をNCLOBロケータに保存し、そのロケータを別のトランザクションまたはセッションで使用することはできません。

関連項目:

Unicodeデータ型のサポートについては、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

拡張データ型

Oracle Database 12cからは、VARCHAR2、NVARCHAR2およびRAWデータ型の最大サイズの32767バイトを指定できます。この新しい最大サイズをデータベースでサポートするかどうかは、初期化パラメータMAX_STRING_SIZEを次のように設定することで制御できます。

- MAX_STRING_SIZE = STANDARDの場合、Oracle Database 12cより前のリリースのサイズ制限が適用されます。このサイズ制限は、VARCHAR2データ型とNVARCHAR2データ型については4,000バイト、およびRAWデータ型については2,000バイトになります。これはデフォルトです。
- MAX_STRING_SIZE = EXTENDEDの場合、VARCHAR2、NVARCHAR2、およびRAWの各データ型について、サイズ制限は32,767バイトになります。

関連項目:

MAX_STRING_SIZE = EXTENDEDを設定すると、データベース・オブジェクトを更新することになる場合があり、そのオブジェクトを無効化してしまう可能性があります。このパラメータの影響と、この新機能を設定して有効化する方法の詳細は、『Oracle Databaseリファレンス』を参照してください。

4,000バイト以上のサイズを宣言したVARCHAR2データ型またはNVARCHAR2データ型、または2,000バイト以上のサイズを宣言したRAWデータ型は、拡張データ型になります。拡張データ型の列は、OracleのLOBテクノロジを利用して表外に格納されます。LOB記憶域は、常に表と連動します。自動セグメント領域管理(ASSM)で管理された表領域の場合、拡張データ型の列はSecureFiles LOBとして格納されます。それ以外の場合は、BasicFiles LOBとして格納されます。記憶域メカニズムとしてのLOBは、内部使用に限定されています。そのため、これに該当するLOBは、DBMS_LOBを使用して操作することはできません。

ノート:

- SecureFiles LOB を記憶域メカニズムとして使用することをお薦めします。BasicFiles LOB では、拡張データ型の列の機能が制限されます。
- 拡張データ型には、LOB と同じルールと制限が適用されます。詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』を参照してください。

RAWデータ型に2,000バイトを超えるサイズを設定するには、MAX_STRING_SIZE = EXTENDEDを設定する必要がありますが、RAWデータ型は、そのサイズが4,000バイトを超える場合にのみ、表外LOBとして格納されることに注意してください。たとえば、RAW(3000)のデータ型を宣言するには、MAX_STRING_SIZE = EXTENDEDを設定する必要があります。ただし、その列は表内に格納されます。

拡張データ型は、標準のデータ型と同様に使用できますが、次のような考慮事項があります。

- 拡張データ型の列に対して索引を作成するときや、索引で主キー制約または一意制約を適用する必要があるときの特別な考慮事項は、拡張データ型の列に対する索引の作成を参照してください。
- リスト・パーティションのパーティション化キー列が拡張データ型の列の場合、パーティションに対して指定しようとする値の

リストがパーティション・バウンドの制限(4KB)を超えていることがあります。この問題を回避する方法の詳細は、CREATE TABLEの「list_partitions」句を参照してください。

- 初期化パラメータMAX_STRING_SIZEの値が影響する項目は、次のとおりです。
  - テキスト・リテラルの最大長。詳細は、テキスト・リテラルを参照してください。
  - 2つの文字列を連結した場合のサイズ制限。詳細は、連結演算子を参照してください。
  - NLSSORTファンクションから返される照合キーの長さ。「NLSSORT」を参照してください。
  - XMLFormatオブジェクトの一部の属性のサイズ。詳細は、XML書式モデルを参照してください。
  - XMLファンクションのXMLCOLATTVAL、XMLELEMENT、XMLFOREST、XMLPI、およびXMLTABLEの一部の式のサイズ。

ROWIDデータ型

データベース内の各行にはアドレスがあります。次の項では、Oracle Databaseにおける行のアドレスの2つの書式について説明します。

ROWIDデータ型

Oracle Database固有のヒープ構成表内の行は、ROWIDという行のアドレスを持ちます。疑似列ROWIDを問い合せることによって、ROWIDの行のアドレスを調べることができます。この疑似列の値は、各行のアドレスを表す文字列です。これらの文字列は、ROWIDデータ型です。ROWID疑似列の詳細は、疑似列を参照してください。

ROWIDには、次の情報が含まれています。

- 行を含むデータファイルのデータ・ブロック。この文字列の長さは、オペレーティング・システムによって異なります。
- データ・ブロック内の行。
- 行を含むデータベース・ファイル。最初のデータファイルは1になります。この文字列の長さは、オペレーティング・システムによって異なります。
- すべてのデータベース・セグメントに割り当てられる識別番号であるデータ・オブジェクト番号。データ・オブジェクト番号は、データ・ディクショナリ・ビューのUSER_OBJECTS、DBA_OBJECTSおよびALL_OBJECTSから取り出すことができます。同じセグメントを共有するオブジェクト(たとえば、同じクラスタ内のクラスタ化された表など)には、同じオブジェクト番号が付けられます。

ROWIDは、BASE 64の値で格納され、文字AからZ、aからz、0から9、プラス記号(+)およびスラッシュ(/)を含めることができます。ROWIDは、直接は使用できません。ROWIDの内容を解析するには、提供されているパッケージDBMS_ROWIDを使用します。パッケージ・ファンクションを使用すると、前述の4つのROWIDの要素に関する情報を抽出して利用できます。

関連項目:

DBMS_ROWIDパッケージで使用できるファンクション、およびこのファンクションの使用方法については、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

UROWIDデータ型

物理アドレスまたは永続アドレス以外のアドレス、またはOracle Databaseが生成したものではないアドレスがある行を持つ表

もあります。たとえば、索引編成表の行のアドレスは、移動可能な索引リーフに格納されます。外部表(ゲートウェイを介してアクセスされるDB2表など)のROWIDは、Oracle標準のROWIDではありません。

Oracleでは、ユニバーサルROWID(UROWID)を使用して索引構成表および外部キー表のアドレスを格納します。索引構成表には、論理UROWIDがあり、外部キー表には外部キーUROWIDがあります。いずれのタイプのUROWIDも、(ヒープ構成表の物理ROWIDのように)ROWID疑似列に格納されます。

Oracleは、表の主キーに基づいて論理ROWIDを作成します。論理ROWIDは、主キーが変更されないかぎり、変更されません。索引構成表のROWID疑似列のデータ型は、UROWIDです。この疑似列には、ヒープ構成表のROWID疑似列と同様に(SELECT ... ROWID文を使用して)アクセスできます。索引構成表のROWIDを格納する場合、表にUROWID型の列を定義し、この列にROWID疑似列の値を取り込みます。

ANSI、DB2、SQL/DSのデータ型

表とクラスタを作成するSQL文では、ANSIデータ型、およびIBM社の製品SQL/DSとDB2のデータ型も使用できます。Oracleでは、Oracle Databaseのデータ型の名前と異なるANSIまたはIBMのデータ型の名前を認識します。データ型をOracleの同等のデータ型に変換して、Oracleのデータ型を列のデータ型の名前として記録し、次の表に示す変換に基づいて、Oracleのデータ型で列データを格納します。

表2-6 Oracleデータ型に変換されるANSIデータ型

ANSI SQLデータ型	Oracleデータ型
CHARACTER(n) CHAR(n)	CHAR(n)
CHARACTER VARYING(n) CHAR VARYING(n)	VARCHAR2(n)
NATIONAL CHARACTER(n) NATIONAL CHAR(n) NCHAR(n)	NCHAR(n)
NATIONAL CHARACTER VARYING(n)、 NATIONAL CHAR VARYING(n)、 NCHAR VARYING(n)	NVARCHAR2(n)
NUMERIC[(p,s)] DECIMAL[(p,s)](ノート1)	NUMBER(p,s)
INTEGER INT	NUMBER(38)

ANSI SQLデータ型	Oracleデータ型
SMALLINT	
FLOAT(ノート 2)	FLOAT(126)
DOUBLE PRECISION(ノート 3)	FLOAT(126)
REAL(ノート 4)	FLOAT(63)

ノート:

1. NUMERICデータ型およびDECIMALデータ型では、固定小数点数のみを指定できます。これらのデータ型では、位取りsのデフォルトは0です。
2. FLOATデータ型は、2進精度bを持つ浮動小数点数です。このデータ型のデフォルト精度は、126桁の2進精度または38桁の10進精度です。
3. DOUBLE PRECISIONデータ型は126桁の2進精度を持つ浮動小数点数です。
4. REALデータ型は63桁の2進精度(18桁の10進精度)を持つ浮動小数点数です。

次のSQL/DSとDB2のデータ型には、対応するOracleデータ型がありません。次のデータ型を持つ列は定義しないでください。

- GRAPHIC
- LONG VARGRAPHIC
- VARGRAPHIC
- TIME

データ型がTIMEであるデータは、Oracleの日時データとしても表現できます。

関連項目:

日時および間隔のデータ型

表2-7 Oracleデータ型に変換されるSQL/DSとDB2のデータ型

SQL/DSとDB2データ型	Oracleデータ型
CHARACTER(n)	CHAR(n)
VARCHAR(n)	VARCHAR(n)
LONG VARCHAR	LONG
DECIMAL(p,s)(ノート 1)	NUMBER(p,s)
INTEGER	NUMBER(p,0)

SQL/DSとDB2データ型	Oracleデータ型
SMALLINT	
FLOAT(ノート 2)	NUMBER

ノート:

1. DECIMALデータ型では、固定小数点数のみを指定できます。このデータ型では、sのデフォルトは0です。
2. FLOATデータ型はbの2進精度を持つ浮動小数点数です。このデータ型のデフォルト精度は126桁の2進精度(38桁の10進精度)です。

ユーザー定義型

ユーザー定義のデータ型は、Oracle組込みデータ型とその他のユーザー定義のデータ型を、アプリケーション内のデータの構造と動作をモデル化するオブジェクト型の構築ブロックとして使用します。次の項で、ユーザー定義型の様々なカテゴリを説明します。

関連項目:

- Oracle組込みデータ型の詳細は、『Oracle Database概要』を参照してください。
- ユーザー定義型の作成方法については、「CREATE TYPE」および「CREATE TYPE BODY」を参照してください。
- ユーザー定義型の使用方法については、『Oracle Databaseオブジェクト・リレーショナル開発者ガイド』を参照してください。

オブジェクト型

オブジェクト型とは、実社会エンティティ(たとえば、発注書など)を抽象化し、アプリケーション・プログラムで処理できるようにしたものです。1つのオブジェクト型は、次の3種類のコンポーネントを持つスキーマ・オブジェクトです。

- 名前 - スキーマ内でオブジェクト型を一意に識別するためのものです。
- 属性 - 組込み型またはその他のユーザー定義型です。属性は、実社会エンティティの構造をモデル化します。
- メソッド - PL/SQLで記述され、データベースに格納されるファンクションまたはプロシージャ、あるいはCやJavaなどの言語で記述され、外部に格納されるファンクションまたはプロシージャのことです。メソッドは、アプリケーションが実社会エンティティに対して実行できる操作を実装します。

REFデータ型

オブジェクト識別子(キーワードOIDで表される)を使用することによって、オブジェクトを一意に識別し、他のオブジェクトまたはリレーショナル表からそのオブジェクトを参照できます。REFと呼ばれるデータ型のカテゴリが、そのような参照を表します。REFデータ型は、オブジェクト識別子のコンテナです。REF値は、オブジェクトへのポインタとなります。

REF値が、存在しないオブジェクトを指している場合、そのREFはDANGLING(参照先がない)状態であるといいます。DANGLING状態のREFは、NULLであるREFと異なります。REFがDANGLING状態であるかどうかを確認するには、条件

IS [NOT] DANGLINGを使用します。たとえば、列型がcustomer_typ(属性cust_emailを持つ)を指しているREFであるcustomer_ref列があるとします。このcustomer_ref列を含むオブジェクト・ビューoc_orders(サンプル・スキーマoe内)は、次のように指定します。

```
SELECT o.customer_ref.cust_email
  FROM oc_orders o
  WHERE o.customer_ref IS NOT DANGLING;
```

VARRAY

配列とは、順序付けられたデータ要素の集合です。ある特定の配列のすべての要素は、同じデータ型です。各要素には索引があります。索引は、各要素の配列内での位置に対応する番号です。

配列内の要素数は、その配列のサイズを表します。Oracleの配列は可変サイズであるため、VARRAYと呼ばれます。VARRAYを宣言する場合は、最大サイズを指定する必要があります。

VARRAYの宣言時に領域は割り当てられません。ここでは次のような型を定義します。

- リレーショナル表の列のデータ型
- オブジェクト型属性
- PL/SQLの変数、パラメータ、またはファンクションの戻り型

通常、Oracleは、1つの配列オブジェクトを表内に(その行の他のデータと同じ表領域に)、または表外に(LOBに)格納します。この形式は配列オブジェクトのサイズによって決まります。ただし、VARRAYに個別の記憶特性を指定する場合、Oracleはこれをサイズに関係なく表外に格納します。VARRAYの格納方法の詳細は、「CREATE TABLE」の「varray_col_properties」を参照してください。

ネストした表

ネストした表は、順序付けられていない要素の集合を表現します。要素には組込み型またはユーザー定義型を指定できます。ネストした表は、単一列の表として表示できます。ネストした表がオブジェクト型の場合、オブジェクト型のそれぞれの属性を表す複数列の表としても表示できます。

ネストした表の定義では、領域は割り当てられません。ネストした表では、次のものを宣言するための型を定義します。

- リレーショナル表の列のデータ型
- オブジェクト型属性
- PL/SQLの変数、パラメータ、またはファンクションの戻り型

ネストした表が、リレーショナル表内の列型として使用される場合、またはオブジェクト表の基礎となるオブジェクト型の属性として使用される場合、Oracleは、ネストした表のすべてのデータを単一表に格納し、その単一表を、ネストした表を囲むリレーショナル表またはオブジェクト表に対応付けます。

Oracleが提供する型

Oracleは、組込み型またはANSIがサポートする型が不十分な場合に、新しい型を定義するためのSQLに基づくインタフェースを提供します。これらの型の動作は、C/C++、JavaまたはPL/SQLで実装されます。Oracle Databaseは、入出力、異機種間でのクライアント側の新しいデータ型へのアクセス、およびアプリケーションとデータベース間のデータ転送のための最適化で必要な下位レベルのインフラストラクチャ・サービスを自動的に提供します。

これらのインタフェースは、ユーザー定義(またはオブジェクト)型の作成に使用できます。また、Oracleが有効なデータ型を作成するときに使用します。このようなデータ型のいくつかはサーバーで提供され、横方向の幅広いアプリケーション領域(任意型など)および縦方向の固有アプリケーション領域(Spatial型など)の両方に役立ちます。

Oracleが提供する型については、次の項で説明します。また、それらの型の実装および使用に関するドキュメントの参照先も示します。

- 任意型
- XML型
- 空間型

任意型

任意型は、実際の型が不明なプロシージャ・パラメータおよび表の列の柔軟性が高いモデリングを提供します。これらのデータ型によって、型の記述、データ・インスタンスおよびその他のSQL型の一連のデータ・インスタンスを動的にカプセル化し、アクセスできます。これらの型を構成およびアクセスするには、OCIおよびPL/SQLインタフェースを使用します。

ANYTYPE

この型には、任意の名前のあるSQL型または名前のない一時型の型の記述を含めることができます。

ANYDATA

この型には、指定した型のインスタンスをデータおよび型の記述とともに含めることができます。ANYDATAは表の列データ型として使用でき、単一の列に異なる型の値を格納できます。値には、SQL組込み型またはユーザー定義型を使用できます。

ANYDATASET

この型には、指定した型の記述およびその型の一連のデータ・インスタンスを含めることができます。ANYDATASETは、柔軟性が必要なプロシージャ・パラメータのデータ型として使用できます。ユーザー定義型と同様に、SQL組込み型のデータ・インスタンスの値も格納できます。

関連項目:

ANYTYPE、ANYDATAおよびANYDATASET型の詳細は、『*Oracle Database PL/SQL*パッケージおよびタイプ・リファレンス』を参照してください。

XML型

eXtensible Markup Language(XML)は、World Wide Web Consortium(W3C)によって開発された、構造化されたデータおよび構造化されていないデータをWebで表示するための標準書式です。Universal Resource Identifiers(URI)は、WebページなどのWeb上のリソースを識別します。Oracleでは、データベース自体に格納されるデータにアクセスするために`DBUriRef`型と呼ばれるURIのクラスと同様に、XMLおよびURIデータを処理するための型が用意されています。また、データベースから外部URIおよび内部URIの両方に格納およびアクセスする型セットを利用できます。

XMLType

Oracleが提供するこの型は、データベースでのXMLデータの格納および問合せに使用します。XMLTypeは、XPath式を使用したXMLデータへのアクセス、抽出および問合せに使用するメンバー・ファンクションを持ちます。XPathは、XML文書をトラバースするためにW3Cによって開発された別の規格です。OracleのXMLTypeファンクションは、W3Cの多数のXPath式をサポートします。また、Oracleは、既存のリレーショナルまたはオブジェクト・リレーショナル・データからXMLType値を作成するためのSQLファンクションおよびPL/SQLパッケージのセットを提供します。

XMLTypeはシステム定義型であるため、ファンクションの引数として、あるいは表またはビューの列のデータ型として使用できます。また、XMLTypeの表およびビューも作成できます。表にXMLType列を作成する際は、XMLデータをCLOB列に(内部的にはCLOBとして格納されるバイナリXMLとして)格納するか、またはオブジェクトと関連付けて格納するかを選択できます。

また、スキーマを登録し(DBMS_XMLSCHEMAパッケージを使用)、登録したスキーマに適合する表または列を作成できます。この場合、XMLデータは、デフォルトでは基礎となるオブジェクト・リレーショナル列に格納されますが、スキーマ・ベースのデータであっても、CLOB列またはバイナリXML列に格納するように指定できます。

XMLType列の問合せおよびDMLは、格納メカニズムとは関係なく、同様に動作します。

関連項目:

XMLType列の使用方法については、『Oracle XML DB開発者ガイド』を参照してください。

URIデータ型

Oracleは、継承階層で関連するURI型のファミリ(URIType、DBURIType、XDBURITypeおよびHTTPURIType)を提供します。URITypeはオブジェクト型であり、その他はURITypeのサブタイプです。URITypeはスーパータイプであるため、この型の列を作成し、DBURITypeまたはHTTPURIType型のインスタンスをこの列に格納できます。

HTTPURIType

HTTPUriTypeを使用すると、外部のWebページまたはファイルのURLを格納できます。Oracleは、Hypertext Transfer Protocol(HTTP)を使用して、これらのファイルにアクセスします。

XDBURIType

XDBURITypeを使用すると、表の任意のURIType列に埋込み可能なURIとして、ドキュメントをXMLデータベース階層に展開できます。XDBUriTypeは、URLで構成されています。URLは、参照先のXML文書の階層名と、XPath構文を表すオプションのフラグメントで構成されています。フラグメントとURL部分は、ポンド記号(#)で区切ります。次に、XDBUriTypeの例を示します。

```
/home/oe/doc1.xml
/home/oe/doc1.xml#/orders/order_item
```

DBURIType

DBURITypeを使用すると、データベース内のデータを参照するDBURIRef値を格納できます。DBURIRefを格納することで、データベースの内部または外部に格納されたデータを参照し、常にデータにアクセスできます。

DBUriRef値は、データベース内のデータを参照するために、XPathのような表現を使用します。データベースをXMLツリーに想定すると、表、行および列がXML文書の要素です。たとえば、サンプルの人事部門のユーザーhrは次のようにXMLツリーで表します。

```
<HR>
```

```
  <EMPLOYEES>
    <ROW>
      <EMPLOYEE_ID>205</EMPLOYEE_ID>
      <LAST_NAME>Higgins</LAST_NAME>
      <SALARY>12008</SALARY>
      .. <!-- other columns -->
    </ROW>
    ... <!-- other rows -->
  </EMPLOYEES>
  <!-- other tables..-->
</HR>
<!-- other user schemas on which you have some privilege on..-->
```

DBUriRefは、この仮想XML文書のXPath式です。したがって、従業員番号205の従業員のSALARY値をEMPLOYEES表で参照するには、DBURIRefを次のように記述します。

```
/HR/EMPLOYEES/ROW[EMPLOYEE_ID=205]/SALARY
```

このモデルを使用すると、CLOB列またはその他の列に格納されたデータを参照し、それらのデータを外部のURLとして外部の世界へ公開できます。

URIFactoryパッケージ

Oracleは、URITypeの様々なサブタイプのインスタンスを作成し、戻すことができるURIFactoryパッケージも提供します。このパッケージは、URL文字列を分析し、URLの種類(HTTP、DBUriなど)を識別し、サブタイプのインスタンスを作成します。DBURIインスタンスを作成する場合は、URLの先頭に接頭辞/oradbを付ける必要があります。たとえば、URIFactory.getURI('/oradb/HR/EMPLOYEES')によってDBURITypeインスタンスが作成され、URIFactory.getUri('/sys/schema')によってXDBURITypeインスタンスが作成されます。

関連項目:

- オブジェクト型および型の継承については、『Oracle Databaseオブジェクト・リレーショナル開発者ガイド』を参照してください。
- 提供される型およびその実装については、『Oracle XML DB開発者ガイド』を参照してください。
- Oracle Advanced QueuingでのXMLTypeの使用については、『Oracle Databaseアドバンスト・キューイング・ユーザーズ・ガイド』を参照してください。

Spatial型

Oracle Spatial and Graphは、位置情報アプリケーション、地理情報システム(GIS)・アプリケーションおよびジオイメージング・アプリケーションのユーザーが、空間データ管理をより簡単かつ自然に行えるように設計されています。一度空間データをOracle Databaseに格納すると、そのデータの操作や取得を簡単に行うことができ、データベースに格納された他のすべてのデータと関連付けることもできます。次のデータ型は、Spatial and Graphをインストールしている場合にのみ使用できます。

SDO_GEOMETRY

空間オブジェクトのジオメトリの記述は、ユーザー定義の表の単一行およびオブジェクト型SDO_GEOMETRYの単一列に格納されます。SDO_GEOMETRY型の列を含むすべての表は、表に対して一意の主キーを定義する別の列を持つ必要があります。このような表は、ジオメトリ表と呼ばれる場合があります。

SDO_GEOMETRYオブジェクト型の定義は、次のとおりです。

```
CREATE TYPE SDO_GEOMETRY AS OBJECT
  (sgo_gtype         NUMBER,
   sdo_srid          NUMBER,
   sdo_point         SDO_POINT_TYPE,
   sdo_elem_info     SDO_ELEM_INFO_ARRAY,
   sdo_ordinates     SDO_ORDINATE_ARRAY);
/
```

SDO_TOPO_GEOMETRY

この型はトポロジ・ジオメトリを記述します。この記述は、ユーザー定義の表の単一行およびオブジェクト型SDO_TOPO_GEOMETRYの単一列に格納されます。

SDO_TOPO_GEOMETRYオブジェクト型の定義は、次のとおりです。

```
CREATE TYPE SDO_TOPO_GEOMETRY AS OBJECT
  (tg_type         NUMBER,
   tg_id           NUMBER,
   tg_layer_id     NUMBER,
   topology_id     NUMBER);
/
```

SDO_GEORASTER

GeoRasterオブジェクト・リレーショナル・モデルでは、ラスター・グリッドまたはイメージ・オブジェクトは、ユーザー定義の表の単一行およびオブジェクト型SDO_GEORASTERの単一列に格納されます。このような表は、GeoRaster表と呼ばれます。

SDO_GEORASTERオブジェクト型の定義は、次のとおりです。

```
CREATE TYPE SDO_GEORASTER AS OBJECT
  (rasterType          NUMBER,
   spatialExtent       SDO_GEOMETRY,
   rasterDataTable     VARCHAR2(32),
   rasterID            NUMBER,
   metadata            XMLType);
/
```

関連項目:

空間データ型の完全な実装および使用のガイドラインは、Oracle Spatial and Graph開発者ガイド、Oracle Spatial and Graphトポロジ・データ・モデルおよびネットワーク・データ・モデル・グラフ開発者ガイドおよびOracle Spatial and Graph GeoRaster開発者ガイドを参照してください。

データ型の比較規則

ここでは、Oracle Databaseが各データ型の値を比較する方法について説明します。

数値

多い値は少ない値よりも大きいと見なされます。すべての負の数は0および正の数よりも小さいと見なされます。このため、-1は100より小さく、-100は-1より小さい数です。

浮動小数点値NaN(非数値)は、その他の数値よりも大きく、NaNとは等しいとみなされます。

関連項目:

比較セマンティクスの詳細は、数値の優先順位および浮動小数点数を参照してください。

日時値

後の日付またはタイムスタンプは前の日付またはタイムスタンプよりも大きいとみなされます。たとえば、29-MAR-2005 (2005年3月29日)に相当する日付は05-JAN-2006 (2006年1月5日)に相当する日付よりも小さく、05-JAN-2006 1:35pm (2006年1月5日午後1時35分)に相当するタイムスタンプは05-JAN-2005 10:09am (2005年1月5日午前10時9分)に相当するタイムスタンプよりも大きいとみなされます。

タイムゾーン付きの2つのタイムスタンプを比較する場合、これらのタイムスタンプは、まずUTC、つまりタイムゾーン・オフセット+00:00に正規化されます。たとえば、16-OCT-2016 05:59am Europe/Warsaw (2016年10月16日午前5時59分、ヨーロッパ/ワルシャワ)に相当するタイムゾーン付きタイムスタンプは、15-OCT-2016 08:59pm US/Pacific (2016年10月15日午前8時59分、米国/太平洋)に相当するタイムスタンプと同じです。両方とも同じ絶対時点を表し、UTCでは2016年10月16日午前3時59分と表されます。

バイナリ値

データ型RAWまたはBLOBのバイナリ値は、バイト・シーケンスです。2つのバイナリ値を比較する場合、2つのバイト・シーケンスの対応する連続したバイトが順番に比較されます。両方の比較対象の値の最初のバイトが異なる場合は、数値が小さいバイトを含むバイナリ値の方が小さいとみなされます。最初のバイトが等しい場合は2つ目のバイトが比較され、以降、比較対象のバイトが異なるか比較処理がいずれかの値の最後に到達するまで同様に比較されます。後者の場合、短い値の方が小さいとみなされます。

データ型BLOBのバイナリ値は、比較条件では直接比較できません。ただし、PL/SQLファンクションDBMS_LOB.COMPAREを使用して比較できます。

関連項目:

DBMS_LOB.COMPAREファンクションの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

文字値

文字値は、2つのメジャーに基づいて比較されます。

- バイナリ照合または言語照合
- 空白埋め比較セマンティクスまたは非空白埋め比較セマンティクス

次の項で、2つのメジャーについて説明します。

バイナリ照合および言語照合

デフォルトであるバイナリ照合では、文字値はバイナリ値と同様に比較されます。記憶域文字セットで2つの文字値のエンコーディングを形成する2つのバイト・シーケンスは、バイナリ値として処理され、「バイナリ値」で説明されているように比較されます。この比較の結果は、ソース文字値のバイナリ比較の結果として戻されます。

関連項目:

文字セットの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

多くの言語では、バイナリ照合によって、文字値の順序が言語的に正しくなくなる場合があります。たとえば、最も一般的な文字セットでは、すべてのラテン大文字の文字コードは、ラテン小文字よりも小さい値です。そのため、バイナリ照合では、順序は次のようになります。

```
MacDonald
MacIntosh
Macdonald
Macintosh
```

しかし、ほとんどのユーザーは、これらの4つの値が次の順序で表示されると想定します。

```
MacDonald
Macdonald
MacIntosh
Macintosh
```

これは、バイナリ照合が英語の文字値にも適さない場合があることを示しています。

Oracle Databaseでは、様々な話し言葉の規則に従って文字列を順序付ける言語照合がサポートされています。文字値を大/小文字またはアクセントを区別せずに照合する照合の変形もサポートされています。言語照合の方がコストは高くなりますが、ユーザー操作に優れています。

関連項目:

言語ソートについては、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

言語照合の制限事項

言語照合を使用する場合は、比較条件、ORDER BY、GROUP BYおよびMATCH_RECOGNIZE問合せ句、COUNT(DISTINCT)および統計集計ファンクション、LIKE条件、ORDER BYおよびPARTITION BY分析句により、照合キーが生成されます。照合キーは、NLSSORTファンクションによって戻されるものと同じ値であり、「NLSSORT」で説明されているものと同じ制限事項があります。

空白埋め比較セマンティクスおよび非空白埋め比較セマンティクス

空白埋め比較セマンティクスでは、2つの値の長さが異なる場合、Oracleはまず短い方の値の最後に空白を追加して、2つの値が同じ長さになるようにします。次に、その2つの値を、最初に異なる文字まで1文字ずつ比較します。最初に異なる文字の位置で、大きい方の文字を持つ値の方が大きいとみなされます。2つの値に異なる文字がない場合、その2つの値は等しいとみ

なされます。この規則では、2つの値の後続空白数のみが異なる場合、その2つの値は等しいとみなされます。Oracleでは、比較する両方の値が、CHARデータ型、NCHARデータ型、テキスト・リテラルのいずれかの式の場合、またはUSERファンクションの戻り値の場合のみ空白埋め比較セマンティクスを使用します。

非空白埋め比較セマンティクスでは、Oracleは、2つの値を、最初に異なる文字まで1文字ずつ比較します。最初に異なる文字の位置で、大きい方の文字を持つ値の方が大きいとみなされます。長さが異なる2つの値を短い方の値の最後まで比較して、すべて同じ文字だった場合、長い方の値が大きいとみなされます。同じ長さの2つの値に異なる文字がない場合、その2つの値は等しいとみなされます。Oracleでは、比較する片方または両方の値がVARCHAR2データ型またはNVARCHAR2データ型の場合、非空白埋め比較セマンティクスを使用します。

これらの異なる比較セマンティクスを使用して2つの文字値を比較した場合、その結果が異なることもあります。次の表に、それぞれの比較セマンティクスを使用して5組の文字を比較した結果を示します。通常、空白埋め比較と非空白埋め比較の結果は同じです。表に示されている最後の比較では、空白埋め比較と非空白埋め比較の違いが明確になっています。

空白埋め比較	**非空白埋め比較**
`'ac' > 'ab'`	`'ac' > 'ab'`
`'ab' > 'a '`	`'ab' > 'a '`
`'ab' > 'a'`	`'ab' > 'a'`
`'ab' = 'ab'`	`'ab' = 'ab'`
`'a ' = 'a'`	`'a ' > 'a'`

データ・バインドされた照合

Oracle Database 12cリリース2 (12.2)以降、特定の文字値を比較または照合するときに使用する照合は、値自体に関連付けられます。これをデータ・バインドされた照合といいます。データ・バインドされた照合は、値のデータ型の属性として表示できます。

以前のリリースのOracle Databaseでは、データベース・セッションのすべての照合依存のSQL操作の照合は、セッション・パラメータNLS_COMPおよびNLS_SORTによって大まかに決定されていました。データ・バインドされた照合アーキテクチャを使用すると、アプリケーションによって、言語固有の比較規則を、これらの規則を必要とするデータに常に正確に適用できます。

Oracle Database 12cリリース2 (12.2)では、表の列の照合を宣言できます。列を照合依存のSQL操作に引数として渡すと、そのSQL操作は、列の宣言された照合を使用して列の値を処理します。SQL操作に複数の文字引数があり、それらが相互に比較される場合、使用する照合は照合決定ルールによって決定されます。

データ・バインドされた照合には次の2つのタイプがあります。

- 名前付き照合: この照合は、照合名で指定された照合ルールの特定のセットです。名前付き照合は、NLS_SORTパラメータの値として指定される照合と同じです。名前付き照合として、バイナリ照合または言語照合を使用できます。
- 疑似照合: この照合では、SQL操作の照合ルールを直接指定しません。かわりに、使用する実際の名前付き照合のセッション・パラメータNLS_SORTおよびNLS_COMPの値を確認するよう操作に指示します。疑似照合は、新しい宣言的な照合指定方法とセッション・パラメータを使用する古い方法の橋渡しとなります。特に、疑似照合USING_NLS_COMPは、Oracle Database 12cリリース2より前の動作とまったく同じように動作するようSQL操作

に指示します。

列の名前付き照合を宣言する場合は、列値の比較方法を静的に決定します。疑似照合を宣言する場合は、セッション・パラメータNLS_COMPおよびNLS_SORTを使用して比較動作を動的に制御できます。ただし、疑似照合で宣言された列に定義されている索引や制約などの静的オブジェクトでは、バイナリ照合が使用されます。動的に設定可能な照合ルールを使用して静的オブジェクトの値を比較することはできません。

式で使用される文字リテラルまたはバイナリ変数の照合は、その式が含まれるデータベース・オブジェクト(ビューまたはマテリアライズド・ビュー問合せ、PL/SQLストアド・ユニット・コード、ユーザー定義型のメソッド・コード、スタンドアロンのDMLまたは問合せ文など)のデフォルト照合から導出されます。Oracle Database 12cリリース2では、PL/SQLストアド・ユニット、ユーザー定義型メソッドおよびスタンドアロンのSQL文のデフォルトの照合は、常に疑似照合のUSING_NLS_COMPです。ビューおよびマテリアライズド・ビューのデフォルト照合は、CREATE VIEWおよびCREATE MATERIALIZED VIEW文のDEFAULT COLLATION句で指定できます。

SQL操作によって文字値が戻される場合、照合導出ルールによってその結果の導出照合が決定されるため、結果が式ツリー内の別の照合依存のSQL操作またはトップレベルのコンシューマ(SELECT文のSQL文の句など)の引数として渡されるとき、照合は既知です。SQL操作の対象が文字引数値の場合、その文字結果の導出照合は、引数の照合に基づきます。それ以外の場合、導出ルールは文字リテラルの場合と同じです。

単純式や演算子結果などの式ノードの導出照合は、COLLATE演算子を使用して上書きできます。

Oracle Databaseでは、列、表またはスキーマに対して大/小文字を区別しない照合を宣言できるため、表またはスキーマ内の列またはすべての文字の列は、常に大/小文字を区別しない方法で比較できます。

関連項目:

- 詳細な照合導出ルールや照合決定ルールなどのデータ・バインドされた照合アーキテクチャの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。
- COLLATE演算子

オブジェクト値

オブジェクト値は、MAPとORDERの2つの比較ファンクションのいずれかを使用して比較されます。どちらのファンクションでもオブジェクト型インスタンスは比較されますが、両者は別のものです。これらのファンクションは、他のオブジェクト型と比較するオブジェクト型の一部として指定される必要があります。

関連項目:

MAPメソッドとORDERメソッド、およびこれらのメソッドが戻す値の詳細は、「CREATE TYPE」を参照してください。

VARRAYとネストした表

ネストした表の比較の詳細は、比較条件を参照してください。

データ型の優先順位

Oracleでは、データ型の優先順位によって、暗黙的にデータ型を変換するかどうかを判断します(次の項を参照)。Oracleデー

タ型の優先順位は、次のとおりです。

- 日時および期間データ型
- BINARY_DOUBLE
- BINARY_FLOAT
- NUMBER
- 文字データ型
- その他のすべての組込みデータ型

データ変換

通常、式では異なるデータ型の値を使用できません。たとえば、式では10に5を掛けた値に'JAMES'を加えることはできません。ただし、Oracleでは値をあるデータ型から別のデータ型へ変換する場合の、暗黙的な変換と明示的な変換をサポートしています。

暗黙的なデータ変換と明示的なデータ変換

暗黙的または自動的な変換に依存するかわりに、Oracleでは明示的な変換の指定をお薦めします。これは、次の理由によります。

- 明示的なデータ型変換ファンクションを使用すると、SQL文がわかりやすくなります。
- 暗黙的なデータ型変換(特に列値のデータ型が定数に変換される場合)は、パフォーマンスに悪影響を及ぼす可能性があります。
- 暗黙的な変換はその変換が行われるコンテキストに依存し、どんな場合でも同様に機能するとはかぎりません。たとえば、日時値からVARCHAR2型へ値を暗黙的に変換すると、NLS_DATE_FORMATパラメータに指定されている値によっては、予期しない年が戻される場合があります。
- 暗黙的な変換のアルゴリズムはソフトウェア・リリース間やOracle製品間で変更される場合があります。明示的な変更では、動作がより安定しています。
- 索引式で暗黙的なデータ型変換が発生した場合、その索引が変換前のデータ型に対して定義されているために、Oracle Databaseがその索引を使用しないことがあります。これは、パフォーマンスに悪影響を与えるおそれがあります。

暗黙的なデータ変換

あるデータ型から別のデータ型への変換が意味を持つ場合、Oracle Databaseは値を自動的に変換します。

表2-8に、Oracleの暗黙的な変換のマトリックスについて示します。この表は、変換の方向または変換されるコンテキストにかかわらず、すべての可能な変換を示します。

セルの'X'は、最初の列とヘッダー行に示されたデータ型の暗黙的な変換を示します。

表2-8 暗黙的な型変換のマトリックス

DataType	CHAR	VARCHAR2	NCHAR	NVARCHAR2	DATE	DATETIME/INTERVAL	NUMBER	BINFLO
CHAR	--	X	X	X	X	X	X	X
VARCHAR2	X	--	X	X	X	X	X	X
NCHAR	X	X	--	X	X	X	X	X
NVARCHAR2	X	X	X	--	X	X	X	X
DATE	X	X	X	X	--	--	--	--
DATETIME/INTERVAL	X	X	X	X	--	--	--	--
NUMBER	X	X	X	X	--	--	--	X
BINARY_FLOAT	X	X	X	X	--	--	X	--
BINARY_DOUBLE	X	X	X	X	--	--	X	X
LONG	X	X	X	X	--	X 脚注 1	--	--
RAW	X	X	X	X	--	--	--	--
ROWID	X	X	X	X	--	--	--	--
CLOB	X	X	X	X	--	--	--	--
BLOB	--	--	--	--	--	--	--	--
NCLOB	X	X	X	X	--	--	--	--

脚注 1

LONGをINTERVALに直接は変換できませんが、TO_CHAR(interval)を使用してLONGをVARCHAR2に変換し、そのVARCHAR2の値をINTERVALに変換できます。

暗黙的なデータ型変換ルール

- INSERTおよびUPDATE操作中に、Oracleは変更する列のデータ型に値を変換します。

- SELECT FROM操作中に、Oracleは列からターゲット変数の型にデータを変換します。
- 数値を操作する際、Oracleは、通常、最大容量を確保するために精度および位取りを調整します。この場合、このような操作によって変換された数値データ型は、基礎となる表に含まれる数値データ型と異なることがあります。
- 数値と文字値を比較する場合、Oracleは文字データを数値に変換します。
- 文字値またはNUMBERの値と浮動小数点数の値の間では、変換が正確に行われない場合があります。これは、文字型およびNUMBERでは10進精度で数値が示されるのに対し、浮動小数点数では2進精度が使用されているためです。
- CLOB値をVARCHAR2などの文字データ型に変換する場合、またはBLOBをRAWデータに変換する場合、変換するデータがターゲットのデータ型より大きいと、データベースはエラーを戻します。
- タイムスタンプ値からDATE値への変換では、タイムスタンプ値の秒の小数部は切り捨てられます。この動作は、タイムスタンプ値の秒の小数部が丸められていた、以前のリリースのOracle Databaseの動作とは異なります。
- BINARY_FLOATからBINARY_DOUBLEへの変換は正確に行われます。
- BINARY_DOUBLEの値がBINARY_FLOATでサポートされている精度のビット数よりも多いビット数を使用している場合、BINARY_DOUBLEからBINARY_FLOATへの変換は正確に行われません。
- DATEの値と文字の値を比較する場合、Oracleは文字データをDATEに変換します。
- SQLファンクションまたは演算子に不当なデータ型の引数を指定して使用する場合、Oracleはその引数を正当なデータ型に変換します。
- 代入を実行する場合、Oracleは等号(=)の右側の値を左側の代入ターゲットのデータ型に変換します。
- 連結中に、Oracleは非文字データ型をCHARまたはNCHARに変換します。
- 文字データ型と非文字データ型に対する演算処理および比較中に、Oracleはすべての文字データ型を数値、日付またはROWIDのいずれかの適切なデータ型に変換します。CHAR/VARCHAR2とNCHAR/NVARCHAR2の演算処理では、OracleはNUMBERに変換します。
- ほとんどのSQL文字ファンクションは、CLOBをパラメータとして指定できます。また、OracleはCLOBと文字型間で暗黙的な変換を実行します。このため、CLOBを使用できないファンクションは、暗黙的な変換を使用してCLOBを受け入れます。このような場合、Oracleはファンクションが起動される前にCLOBをCHARまたはVARCHAR2に変換します。CLOBが4000バイトより大きい場合、Oracleは最初の4000バイトのみをCHARに変換します。
- RAWまたはLONGRAWデータを文字データとの間で変換する場合、バイナリ・データは16進形式で表され、16進文字1文字ごとに4ビットのRAWデータを表します。詳細は、「RAWデータ型とLONG RAWデータ型」を参照してください。
- CHARとVARCHAR2、NCHARとNVARCHAR2の比較では、異なる文字セットが必要な場合があります。このような場合のデフォルトの変換の方向は、データベース文字セットから各国語文字セットです。表2-9に、異なるキャラクタ・タイプ間での暗黙的な変換の方向を示します。

表2-9 異なるキャラクタ・タイプの変換方向

SourceData型	CHARへ	VARCHAR2へ	NCHARへ	NVARCHAR2へ
CHAR から	--	VARCHAR2	NCHAR	NVARCHAR2

SourceData型	CHARへ	VARCHAR2へ	NCHARへ	NVARCHAR2へ
VARCHAR2 から	VARCHAR2	--	NVARCHAR2	NVARCHAR2
NCHAR から	NCHAR	NCHAR	--	NVARCHAR2
NVARCHAR2 から	NVARCHAR2	NVARCHAR2	NVARCHAR2	--

コレクションなどのユーザー定義型は暗黙的に変換できないため、CAST ... MULTISETを使用して明示的に変換する必要があります。

暗黙的なデータ変換の例

テキスト・リテラルの例

テキスト・リテラル10のデータ型はCHARです。次の文のように数式で使用すると暗黙的にNUMBERデータ型に変換されます。

```
SELECT salary + '10'
  FROM employees;
```

文字値および数値の例

条件で文字値とNUMBER型の値を比較する場合、NUMBER型の値は文字値に変換されず、文字値が暗黙的にNUMBER型の値に変換されます。次の文では、'200'が暗黙的に200に変換されます。

```
SELECT last_name
  FROM employees
  WHERE employee_id = '200';
```

日付の例

次の文では、デフォルトの日付書式DD-MON-YYを使用して、24-JUN-06がDATE値に暗黙的に変換されます。

```
SELECT last_name
  FROM employees
  WHERE hire_date = '24-JUN-06';
```

明示的なデータ変換

SQL変換ファンクションを使用すると、データ型の変換を明示的に指定できます。表2-10に、値をあるデータ型から別のデータ型に明示的に変換するSQLファンクションを示します。

Oracleが暗黙的なデータ型変換を行うことができる場合には、LONGおよびLONG RAWの値を指定できません。たとえば、ファンクションや演算子を含む式では、LONGとLONG RAWの値を使用できません。LONGデータ型およびLONG RAWデータ型の制限については、「LONGデータ型」を参照してください。

表2-10 明示的な型の変換

SourceData 型	CHAR、VARCHAR2、NCHAR、NVARCHAR2へ	NUMBER へ	Datetime/Interval へ	RAW へ	ROWID へ	LONG、LONG RAW へ	CLOB、NCLOB、BLOB へ	BI FL
CHAR、VARCHAR2、NCHAR、NVARCHAR2 から	TO_CHAR (char.) TO_NCHAR (char.)	TO_NUMBER	TO_DATE TO_TIMESTAMP TO_TIMESTAMP_TZ TO_YMINTERVAL TO_DSINTERVAL	HEXTORAW	CHARTO-=ROWID	--	TO_CLOB TO_NCLOB	TO Y_
NUMBER から	TO_CHAR (number) TO_NCHAR (number)	--	TO_DATE NUMTOYM-INTERVAL NUMTODS-INTERVAL	--	--	--	--	TO Y_
Datetime Interval から	TO_CHAR (date) TO_NCHAR (datetime)	--	--	--	--	--	--	--
RAW から	RAWTOHEX RAWTONHEX	--	--	--	--	--	TO_BLOB	--
ROWID から	ROWIDTOCHAR	--	--	--	--	--	--	--
LONG/LONG RAW から	--	--	--	--	--	--	TO_LOB	--
CLOB、NCLOB、BLOB から	TO_CHAR TO_NCHAR	--	--	--	--	--	TO_CLOB TO_NCLOB	--
CLOB、NCLOB、BLOB から	TO_CHAR TO_NCHAR	--	--	--	--	--	TO_CLOB TO_NCLOB	--

SourceData型	CHAR、VARCHAR2、NCHAR、NVARCHAR2へ	NUMBERへ	Datetime/Intervalへ	RAWへ	ROWIDへ	LONG、LONG RAWへ	CLOB、NCLOB、BLOBへ	BIFL
BINARY_FLOAT から	TO_CHAR (char.) TO_NCHAR (char.)	TO_NUMBER	--	--	--	--	--	TOY_
BINARY_DOUBLE から	TO_CHAR (char.) TO_NCHAR (char.)	TO_NUMBER	--	--	--	--	--	TOY_

関連項目:

すべての明示的な変換ファンクションの詳細は、変換ファンクションを参照してください。

データ変換のセキュリティ上の考慮事項

暗黙的な変換、または書式モデルを指定しない明示的な変換のいずれかによって日時値がテキストに変換される場合、書式モデルはグローバリゼーション・セッション・パラメータの1つによって定義されます。ソースのデータ型に応じて、パラメータ名はNLS_DATE_FORMAT、NLS_TIMESTAMP_FORMATまたはNLS_TIMESTAMP_TZ_FORMATです。これらのパラメータの値は、クライアント環境で、またはALTER SESSION文で指定できます。

書式モデルがセッション・パラメータに依存している場合、明示的な書式モデルが指定されていない変換が動的SQL文のテキストに連結される日時値に適用されると、データベースのセキュリティに悪影響を及ぼす可能性があります。動的SQL文は、実行のためにデータベースに渡される前にそのテキストがフラグメントから連結される文です。動的SQLは、組込みPL/SQLパッケージDBMS_SQLまたはPL/SQL文EXECUTE IMMEDIATEに関連付けられる場合が多いですが、動的に構成されたSQLテキストを引数として渡すことができる場所はこれらのみではありません。たとえば:

```
EXECUTE IMMEDIATE
'SELECT last_name FROM employees WHERE hire_date > ''' || start_date || '''';
```

start_dateのデータ型はDATEです。

前述の例では、start_dateの値はセッション・パラメータNLS_DATE_FORMATで指定された書式モデルを使用してテキストに変換されます。結果は、連結されてSQLテキストになります。日時書式モデルは、単純に二重引用符で囲んだリテラル・テキストで構成できます。したがって、セッションのグローバリゼーション・パラメータを明示的に設定できるすべてのユーザーが、前述の変換で生成されるテキストを決定できます。SQL文がPL/SQLプロシージャによって実行される場合、そのプロシージャはセッション・パラメータによるSQLインジェクションに対して脆弱になります。セッション自体より強い権限である定義者の権限でプロシージャが実行されると、ユーザーは機密データに不正にアクセスすることができます。

関連項目:

他の例およびこのセキュリティ・リスクを回避する場合の推奨事項については、『Oracle Database PL/SQL言語リファレンス』を参照してください。

ノート:

このセキュリティ・リスクは、データベースまたは OCI 日時ファンクションによってテキストに変換された日時値から SQL テキストを構成する中間層アプリケーションにも該当します。セッションのグローバリゼーション・パラメータがユーザー・プリファレンスから取得される場合、それらのアプリケーションは脆弱になります。

数値の暗黙的および明示的な変換でも、変換結果がセッション・パラメータNLS_NUMERIC_CHARACTERSに依存する場合があるため、類似した問題が発生する可能性があります。このパラメータは、小数点区切り文字および桁区切り文字を定義します。小数点区切りを一重引用符または二重引用符と定義すると、SQLインジェクションが発生する可能性があります。

関連項目:

- セッションのグローバリゼーション・パラメータの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。
- 書式モデルの詳細は、書式モデルを参照してください。

リテラル

「リテラル」と「定数値」という用語は同義語であり、固定されたデータ値を表します。たとえば、'JACK'、'BLUE ISLAND'および'101'はすべて文字リテラルです。5001は数値リテラルです。文字リテラルは、一重引用符で囲みます。一重引用符を付けることで、Oracleは文字リテラルとスキーマ・オブジェクト名を区別します。

この項の内容は次のとおりです。

- テキスト・リテラル
- 数値リテラル
- 日時リテラル
- 期間リテラル

多くのSQL文とファンクションでは、文字リテラルと数値リテラルを指定する必要があります。式と条件の一部として、リテラルを指定できます。文字リテラルは'text'の表記法を、各国文字リテラルはN'text'の表記法を、数値リテラルはリテラルのコンテキストに応じてintegerまたはnumberの表記法を使用して指定できます。これらの表記法の構文については、次の項で説明します。

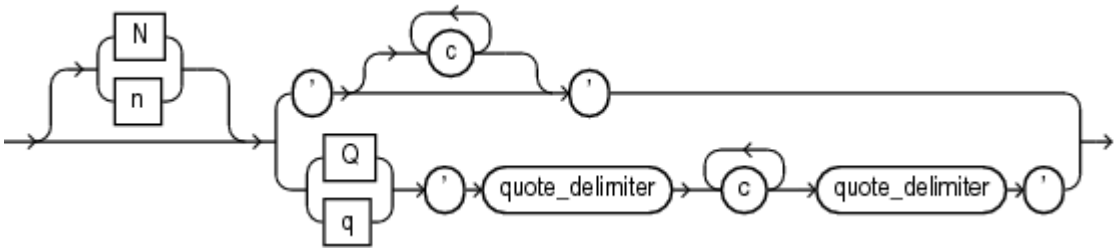
Datetime(日時)またはInterval(期間)のデータ型をリテラルとして指定する場合は、データ型に含まれているオプションの精度を考慮する必要があります。Datetime (日時)およびInterval (期間)のデータ型をリテラルとして指定する場合の例は、データ型の関連する項を参照してください。

テキスト・リテラル

このマニュアルの他の箇所で、式、条件、SQLファンクションおよびSQL文の各構文に示されているstringに値を指定するときには、必ずこのテキスト・リテラルの表記法を使用してください。このマニュアルでは、 テキスト・リテラル、文字リテラルおよび文字列は同じ用語として使用しています。テキスト、文字、文字列リテラルは必ず一重引用符で囲まれています。構文にcharが使用されている場合、テキスト・リテラルを指定するか、または文字データに変換する他の式(たとえば、hr.employees表のlast_name列)を指定します。構文にcharがある場合、一重引用符で囲む必要はありません。

テキスト・リテラルまたは文字列の構文は次のとおりです。

string::=



ここで、Nまたはnは、各国語文字セット(NCHARまたはNVARCHAR2データ)を使用してリテラルを指定します。デフォルトでは、この表記法を使用して入力したテキストは、サーバーで使用するときにデータベースの文字セットによって各国語文字セットに変換されます。テキスト・リテラルをデータベースの文字セットに変換しているときにデータの消失を避けるためには、環境変数ORA_NCHAR_LITERAL_REPLACEにTRUEを設定してください。このように設定することで、n'を透過的に内部で置き換え、SQLの処理中にテキスト・リテラルを保持します。

関連項目:

N付き引用符で表されたリテラルの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

構文の上の方のブランチでは、次のようになります。

- cは、データベース文字セットの任意の要素です。リテラル内の一重引用符(')の前には、エスケープ文字を付ける必要があります。リテラル内で一重引用符を表すには、一重引用符を2つ使用します。
- ' 'は、テキスト・リテラルの始まりと終わりを示す2つの一重引用符です。

構文の下の方のブランチでは、次のようになります。

- QまたはqIは、代替引用メカニズムが使用されることを示します。このメカニズムを使用すると、様々なデリミタをテキスト文字列に使用できます。
- 一番外側の' 'は、開始と終了のquote_delimiterの前後に付ける2つの一重引用符です。
- cは、データベース文字セットの任意の要素です。cで構成されるテキスト・リテラル内に引用符(")を含めることができます。また、一重引用符が直後に付かないquote_delimiterも含めることができます。
- quote_delimiterは、空白、タブおよび改行文字を除く、任意のシングルバイト文字またはマルチバイト文字です。quote_delimiterには一重引用符も使用できます。ただし、quote_delimiterがテキスト・リテラル自体に使用されている場合は、一重引用符を直後に付けないようにしてください。

  開始のquote_delimiterが[、{、<または(のいずれかである場合、終了のquote_delimiterも対応する]、}、>または)である必要があります。それ以外の場合は常に、開始および終了のquote_delimiterは同じ文字である必要があります。

テキスト・リテラルは、次のようにCHARデータ型とVARCHAR2データ型の両方のプロパティを持ちます。

- 式と条件の中のテキスト・リテラルは、OracleによってCHARデータ型として扱われ、空白埋め比較セマンティクスで比較されます。
- テキスト・リテラルの最大長は、初期化パラメータがMAX_STRING_SIZE = STANDARDの場合は4,000バイトになり、MAX_STRING_SIZE = EXTENDEDの場合は32,767になります。詳細は、拡張データ型を参照してください。

有効なテキスト・リテラルの例を次に示します。

```
'Hello'
'ORACLE.dbs'
'Jackie''s raincoat'
'09-MAR-98'
N'nchar literal'
```

代替引用メカニズムを使用した場合の、有効なテキスト・リテラルの例を次に示します。

```
q'!name LIKE '%DBMS_%%'!'
q'<'So,' she said, 'It's finished.'>'
q'{SELECT * FROM employees WHERE last_name = 'Smith';}'
nq'ï Ÿ1234 ï'
q'"name like '['"'
```

関連項目:

空白埋め比較セマンティクスおよび非空白埋め比較セマンティクス

数値リテラル

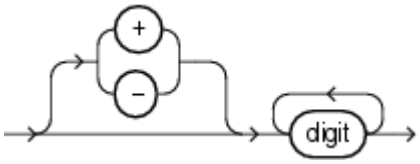
固定長の数値または浮動小数点数値を指定する場合は、数値リテラルの表記を使用します。

整数リテラル

このマニュアルの他の箇所で、式、条件、SQLファンクションおよびSQL文に示されているinteger(整数)に値を指定するときには、必ずこの表記法を使用してください。

integerの構文は次のとおりです。

integer::=



digitは0、1、2、3、4、5、6、7、8、9のいずれかです。

整数は最大38桁の精度を記憶できます。

次に、有効な整数を示します。

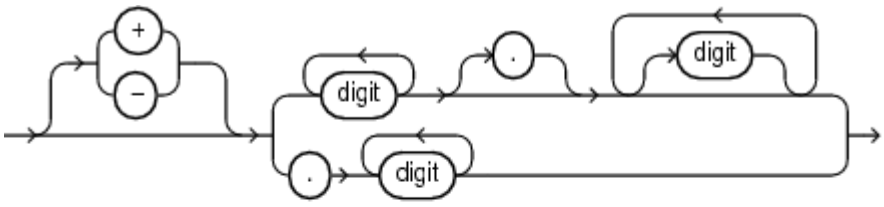
```
7
+255
```

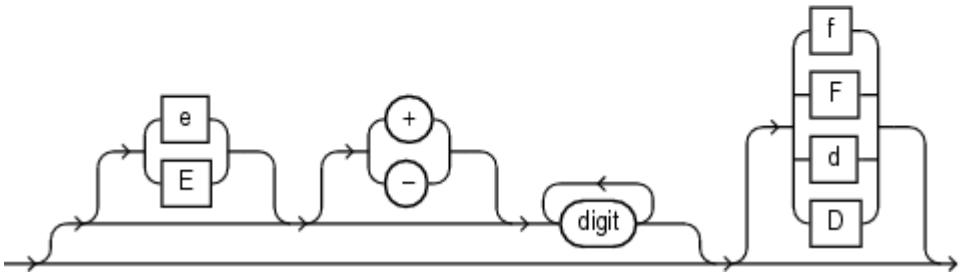
NUMBERおよび浮動小数点リテラル

このマニュアルの他の箇所で、式、条件、SQLファンクションおよびSQL文に示されているnumberまたはn(数)に値を指定するときには、必ずこれらの表記法を使用してください。

numberの構文は次のとおりです。

number::=





説明:

- +は正、-は負の値を示します。符号を省略すると、デフォルトで正の値になります。

- digitは0、1、2、3、4、5、6、7、8、9のいずれかです。
- eまたはEは、数が科学表記法で指定されることを示します。Eの後の数字が指数を示します。指数は-130から125の範囲で指定します。
- fまたはFは、数がBINARY_FLOAT型の32ビットの2進浮動小数点数であることを示します。
- dまたはDは、数がBINARY_DOUBLE型の64ビットの2進浮動小数点数であることを示します。

  f(F)およびd(D)を省略すると、数はNUMBER型になります。

  接尾辞f(F)およびd(D)は、浮動小数点数リテラルでのみサポートされます。文字列ではNUMBERに変換されるため、サポートされません。たとえば、OracleがNUMBERを想定している場合に、文字列'9'を使用すると、文字列は数字の9に変換されます。ただし、文字列'9f'を使用すると、変換は行われず、エラーが戻されます。

NUMBER型の数値は、最大38桁の精度を記憶できます。NUMBER、BINARY_FLOATまたはBINARY_DOUBLEで提供される精度以上の精度がリテラルに必要な場合、値は切り捨てられます。リテラルの範囲がNUMBER、BINARY_FLOATまたはBINARY_DOUBLEでサポートされる範囲を超える場合、エラーが発生します。

数値リテラルはSQL構文の要素であり、NLS設定には従いません。数値リテラルの小数点区切り文字は、常にピリオド(.)です。ただし、数値が必要な箇所でテキスト・リテラルが指定されている場合、テキスト・リテラルはNLSに従って暗黙的に数値に変換されます。テキスト・リテラルに含まれる小数点区切りは、初期化パラメータNLS_NUMERIC_CHARACTERSで設定されたものである必要があります。NLS環境に関係なく動作するように、SQLスクリプトでは数値リテラルを使用することが推奨されます。

次の例は、数値リテラルとテキスト・リテラルでの小数点区切りの動作を示しています。これらの例では、次の文で現在のセッションに、カンマ(,)がNLS小数点区切りとして設定されていると想定しています。

```
ALTER SESSION SET NLS_NUMERIC_CHARACTERS=',.';
```

前の文では、ピリオド(.)もNLSグループ・セパレータとして設定されていますが、それはこれらの例とは関係ありません。

この例では、数値リテラル1.23では必要な小数点区切り(.)が使用されており、テキスト・リテラル'2,34'では設定されたNLS小数点区切り(,)が使用されています。テキスト・リテラルは数値2.34に変換され、出力は小数点区切りのカンマを使用して表示されます。

```
SELECT 2 * 1.23, 3 * '2,34' FROM DUAL;
    2*1.23   3*'2,34'
---------- ----------
      2,46       7,02
```

次の例では、カンマは数値リテラルの一部として扱われていません。かわりに、カンマは2つの数式2*1と23のリストのデリミタとして扱われています。

```
SELECT 2 * 1,23 FROM DUAL;
       2*1         23
---------- ----------
         2         23
```

次の例は、テキストから数値への暗黙的な変換を成功させるには、テキスト・リテラルの小数点区切りとNLS小数点区切りが一致する必要があることを示しています。 次の文は、小数点区切り(.)が設定されたNLS小数点区切り(,)と一致しないため失敗します。

```
SELECT 3 * '2.34' FROM DUAL;
           *
ERROR at line 1:
ORA-01722: invalid number
```

関連項目:

ALTER SESSIONおよび『Oracle Databaseリファレンス』を参照してください。

次に、有効なNUMBERリテラルを示します。

```
25
+6.34
0.5
25e-03
-1
```

次に、有効な浮動小数点数リテラルを示します。

```
25f
+6.34F
0.5d
-1D
```

値を数値リテラルとして表現できない場合は、次の浮動小数点リテラルを使用することもできます。

表2-11 浮動小数点リテラル

リテラル	意味	例
binary_float_nan	IS NAN 条件が true である BINARY_FLOAT 型の値	SELECT COUNT(*) FROM employees WHERE TO_BINARY_FLOAT(commission_pct) != BINARY_FLOAT_NAN;
binary_float_infinity	単精度の正の無限大	SELECT COUNT(*) FROM employees WHERE salary < BINARY_FLOAT_INFINITY;
binary_double_nan	IS NAN 条件が true である BINARY_DOUBLE 型の値	SELECT COUNT(*) FROM employees WHERE TO_BINARY_FLOAT(commission_pct) != BINARY_FLOAT_NAN;
binary_double_infinity	倍精度の正の無限大	SELECT COUNT(*) FROM employees WHERE salary < BINARY_DOUBLE_INFINITY;

日時リテラル

Oracle Databaseは、DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONEおよびTIMESTAMP WITH LOCAL TIME ZONEの4つの日時データ型をサポートしています。

日付リテラル

DATE値を文字列リテラルに指定するか、文字値や数値をTO_DATEファンクションによって日付値に変換できます。Oracle Databaseが文字列リテラルのかわりにTO_DATE式を受け入れるのは、DATEリテラルの場合だけです。

DATE値をリテラルに指定する場合は、グレゴリオ暦を使用する必要があります。次の例に示すように、ANSIのリテラルを指定できます。

```
DATE '1998-12-25'
```

ANSIの日付リテラルには、時刻部分を含めず、書式'YYYY-MM-DD'で指定する必要があります。また、次のように、Oracleの日付値を指定できます。

```
TO_DATE('98-DEC-25 17:30','YY-MON-DD HH24:MI')
```

OracleのDATE値のデフォルトの日付書式は、初期化パラメータNLS_DATE_FORMATで指定します。この例は、日付としての2桁の数、月の名前の省略形、年の下2桁および24時間表記の時刻を含む日付書式です。

デフォルト日付書式の文字値が日付式で使用されると、Oracleは自動的にそれらを日付値に変換します。

日付値を指定する場合に時刻コンポーネントを指定しないと、デフォルト時刻の真夜中(24時間表記では00:00:00、12時間表記では12:00:00)が採用されます。日付値を指定する場合に日付を指定しないと、デフォルト日付である現在の月の最初の日が採用されます。

OracleのDATE列には、常に、日付フィールドと時刻フィールドが含まれます。したがって、DATE列を問い合せる場合は、問合せで時刻フィールドを指定するか、またはDATE列の時刻フィールドが真夜中に設定されていることを確認する必要があります。そうでない場合、Oracleは、正しい結果を戻さない場合があります。時刻フィールドを真夜中に設定するには、TRUNC日付ファンクションを使用します。また、問合せに、等価性や非等価性の条件のかわりに大/小条件を含めることもできます。

次の例では、数値列row_numおよびDATE列datecolを持つ表my_tableがあると想定します。

```
INSERT INTO my_table VALUES (1, SYSDATE);
INSERT INTO my_table VALUES (2, TRUNC(SYSDATE));
SELECT *
  FROM my_table;
   ROW_NUM DATECOL
---------- ---------
         1 03-OCT-02
         2 03-OCT-02
SELECT *
  FROM my_table
  WHERE datecol > TO_DATE('02-OCT-02', 'DD-MON-YY');
   ROW_NUM DATECOL
---------- ---------
         1 03-OCT-02
         2 03-OCT-02
SELECT *
  FROM my_table
  WHERE datecol = TO_DATE('03-OCT-02','DD-MON-YY');
   ROW_NUM DATECOL
---------- ---------
         2 03-OCT-02
```

DATE列の時刻フィールドが真夜中に設定されている場合は、前述の例に示すように、DATE列に対して問い合せるか、DATEリテラルを使用して問い合せることができます。

```
SELECT *
  FROM my_table
  WHERE datecol = DATE '2002-10-03';
   ROW_NUM DATECOL
---------- ---------
         2 03-OCT-02
```

ただし、DATE列が真夜中以外の値を含む場合、正しい結果を得るためには、問合せで時刻フィールドを排除する必要があり

ます。たとえば:

```
SELECT *
  FROM my_table
  WHERE TRUNC(datecol) = DATE '2002-10-03';
   ROW_NUM DATECOL
---------- ---------
         1 03-OCT-02
         2 03-OCT-02
```

Oracleは、問合せの各行にTRUNCファンクションを適用します。これによって、データの時刻フィールドが真夜中である場合、パフォーマンスが向上します。時刻フィールドを真夜中に設定するには、挿入および更新時に次のいずれかの操作を行います。

- TO_DATEファンクションを使用して、時刻フィールドをマスクします。

  ```
  INSERT INTO my_table
    VALUES (3, TO_DATE('3-OCT-2002','DD-MON-YYYY'));
  ```

- DATEリテラルを使用します。

  ```
  INSERT INTO my_table
    VALUES (4, '03-OCT-02');
  ```

- TRUNCファンクションを使用します。

  ```
  INSERT INTO my_table
    VALUES (5, TRUNC(SYSDATE));
  ```

SYSDATE日付ファンクションは、現在のシステムの日付および時刻を返します。CURRENT_DATEファンクションは、現行のセッションの日付を返します。SYSDATE、TO_*日時ファンクションおよびデフォルト日付書式の詳細は、日時ファンクションを参照してください。

TIMESTAMPリテラル

TIMESTAMPデータ型には、年、月、日、時、分、秒および小数秒の値が格納されます。TIMESTAMPをリテラルに指定する場合、fractional_seconds_precision値には最大9桁を指定できます。次に例を示します。

```
TIMESTAMP '1997-01-31 09:26:50.124'
```

TIMESTAMP WITH TIME ZONEリテラル

TIMESTAMP WITH TIME ZONEデータ型はTIMESTAMPの変形で、値にはタイムゾーン地域名またはタイムゾーン・オフセットが含まれます。TIMESTAMP WITH TIME ZONEをリテラルに指定する場合、fractional_seconds_precision値には最大9桁を指定できます。たとえば:

```
TIMESTAMP '1997-01-31 09:26:56.66 +02:00'
```

2つのTIMESTAMP WITH TIME ZONE値がUTCで同じ時刻を表す場合は、データに格納されたTIME ZONEオフセットにかかわらず、同一であるとみなされます。たとえば、次のようになります。

```
TIMESTAMP '1999-04-15 8:00:00 -8:00'
```

前述の例文は次の例文と同じです。

```
TIMESTAMP '1999-04-15 11:00:00 -5:00'
```

太平洋標準時の午前8時は、東部標準時の午前11時と同じです。

UTCオフセットをTZR(タイムゾーン地域名)書式要素に置換できます。次の例では、前述の例と同じ値を持ちます。

```
TIMESTAMP '1999-04-15 8:00:00 US/Pacific'
```

夏時間に切り替えられる境界のあいまいさを排除するには、TZRおよび対応するTZD書式要素の両方を使用します。次の例では、前述の例が確実に夏時間の値を戻します。

```
TIMESTAMP '1999-10-29 01:30:00 US/Pacific PDT'
```

タイムゾーン・オフセットは日時式を使用して表記することもできます。

```
SELECT TIMESTAMP '2009-10-29 01:30:00' AT TIME ZONE 'US/Pacific'
  FROM DUAL;
```

関連項目:

詳細は、日時式を参照してください。

ERROR_ON_OVERLAP_TIMEセッション・パラメータをTRUEに設定しておくと、TZD書式要素を追加しなかったために日時値があいまいな場合、Oracleはエラーを戻します。そのパラメータがFALSEに設定されている場合、不明確な日時は、指定されている地域の標準時間として解釈されます。

TIMESTAMP WITH LOCAL TIME ZONEリテラル

TIMESTAMP WITH LOCAL TIME ZONEデータ型は、TIMESTAMP WITH TIME ZONEとは異なり、データベースに格納されるデータはデータベースのタイムゾーンに対して正規化されます。タイムゾーン・オフセットは、列データの一部として格納されません。TIMESTAMP WITH LOCAL TIME ZONEにはリテラルはありません。他の有効な日時リテラルを使用してこのデータ型の値を示してください。次の表には、TIMESTAMP WITH LOCAL TIME ZONE列に値を挿入するときに使用できる一部の書式と、問合せによって戻される対応する値を示します。

表2-12 TIMESTAMP WITH LOCAL TIME ZONEリテラル

INSERT文の中で指定する値	問合せによって戻される値
'19-FEB-2004'	19-FEB-2004.00.00.000000 AM
SYSTIMESTAMP	19-FEB-04 02.54.36.497659 PM
TO_TIMESTAMP('19-FEB-2004', 'DD-MON-YYYY')	19-FEB-04 12.00.00.000000 AM
SYSDATE	19-FEB-04 02.55.29.000000 PM
TO_DATE('19-FEB-2004', 'DD-MON-YYYY')	19-FEB-04 12.00.00.000000 AM
TIMESTAMP'2004-02-19 8:00:00 US/Pacific'	19-FEB-04 08.00.00.000000 AM

指定した値に時刻コンポーネントがない場合(明示的または暗黙的に)、デフォルトの午前0時の値が戻されます。

期間リテラル

期間リテラルは期間を指定します。年および月、または日付、時間、分および秒の違いを指定できます。Oracle Databaseは、YEAR TO MONTHおよびDAY TO SECONDの2種類の期間リテラルをサポートします。各リテラルは先行フィールドを含み、後続フィールドを含むこともあります。先頭フィールドは、測定する日付または時間の基本単位を定義します。末尾フィールドは、該当する基本単位の最小の増分を定義します。たとえば、YEAR TO MONTH期間では、最も近い月に対する年との期間が考慮されます。DAY TO MINUTE期間では、最も近い分に対する日との期間が考慮されます。

数値形式の日付データがある場合、NUMTOYMINTERVALまたはNUMTODSINTERVAL変換ファンクションを使用して、数値データを期間値へ変換できます。

期間リテラルは、主に分析ファンクションとともに使用します。

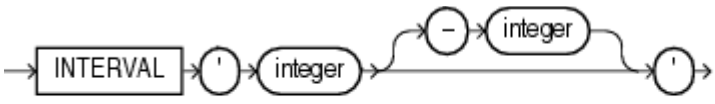
関連項目:

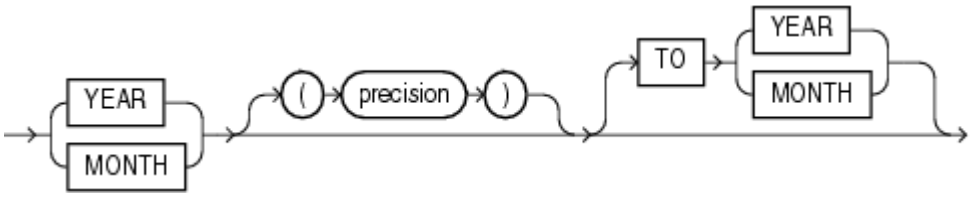
分析ファンクション、NUMTODSINTERVAL、NUMTOYMINTERVAL

INTERVALYEARTOMONTH

次の構文を使用して、YEAR TO MONTH期間リテラルを指定します。

interval_year_to_month::=





説明:

- 'integer [-integer]'には、リテラルの先行フィールドおよびオプションの後続フィールドの整数値を指定します。先行フィールドがYEARで、後続フィールドがMONTHの場合、MONTHフィールドの整数値の範囲は0から11です。
- precisionは、先行フィールドの桁数です。先行フィールド精度の有効範囲は0から9で、デフォルトは2です。

先行フィールドに対する制限

後続のフィールドを指定する場合、先行のフィールドよりも下位の単位を指定する必要があります。たとえば、INTERVAL '0-1' MONTH TO YEARは無効です。

次のINTERVAL YEAR TO MONTHリテラルは、intervalが123年2か月であることを示しています。

```
INTERVAL '123-2' YEAR(3) TO MONTH
```

このリテラルの他の書式の例を次に示します。省略バージョンも含みます。

表2-13 INTERVAL YEAR TO MONTHリテラルの書式

期間リテラルの形式	解釈
`INTERVAL '123-2' YEAR(3) TO MONTH`	123 年 2 か月の interval を示します。先行フィールド精度がデフォルトの 2 桁より大きい場合、その精度を指定してください。
`INTERVAL '123' YEAR(3)`	123 年 0 か月の interval を示します。
`INTERVAL '300' MONTH(3)`	300 か月の interval を示します。
`INTERVAL '4' YEAR`	`INTERVAL '4-0' YEAR TO MONTH` へマップし、4 年を示します。
`INTERVAL '50' MONTH`	`INTERVAL '4-2' YEAR TO MONTH` へマップし、50 か月(4 年 2 か月)を示します。
`INTERVAL '123' YEAR`	デフォルト精度は 2 ですが、123 が 3 桁のため、エラーを戻します。

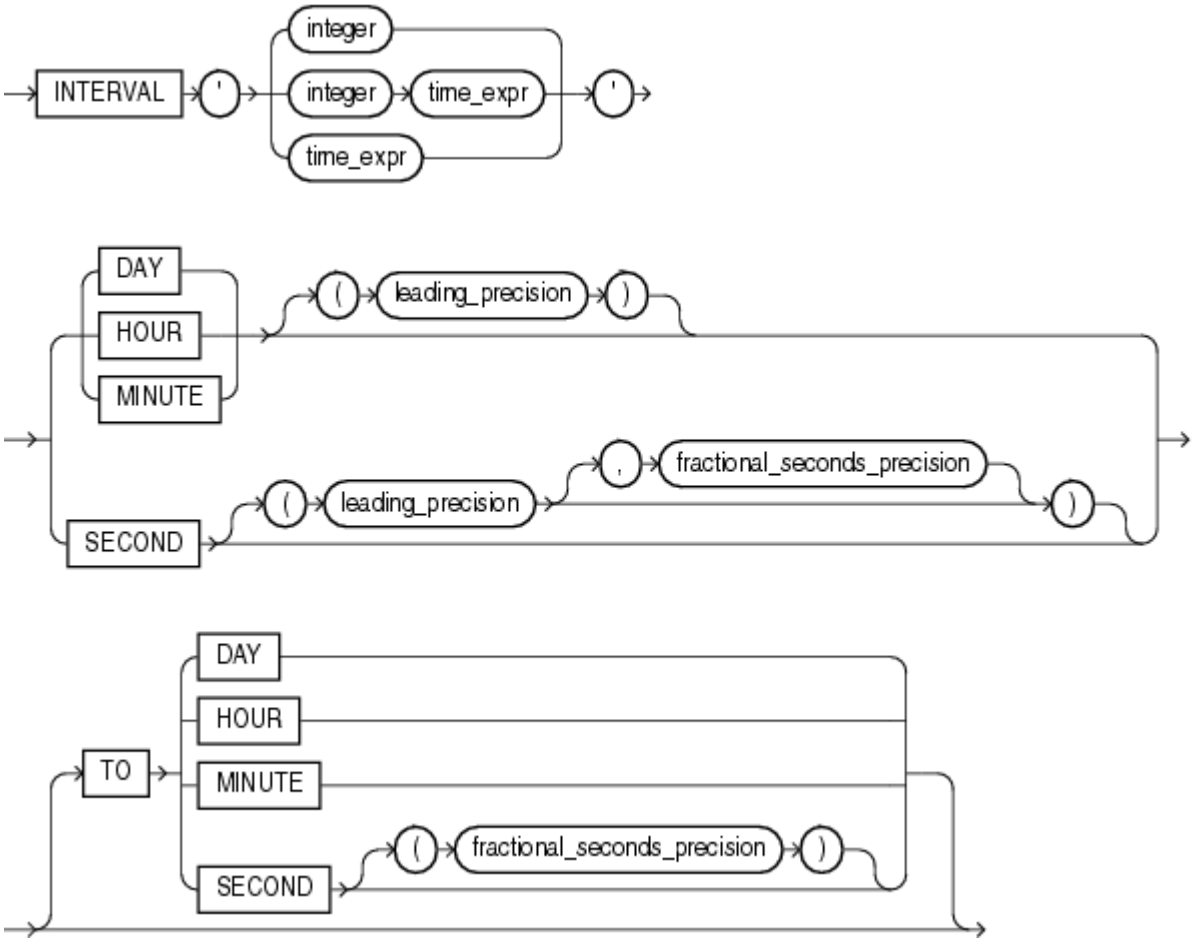
あるINTERVAL YEAR TO MONTHリテラルを別のリテラルに加算または減算して、新しいINTERVAL YEAR TO MONTHリテラルを作成できます。たとえば:

```
INTERVAL '5-3' YEAR TO MONTH + INTERVAL'20' MONTH =
INTERVAL '6-11' YEAR TO MONTH
```

INTERVALDAYTOSECOND

次の構文を使用して、DAY TO SECOND期間リテラルを指定します。

interval_day_to_second::=



説明:

- integerは日数を指定します。ここで指定した値の桁数が先行精度で指定した桁数より大きい場合、Oracleはエラーを戻します。
- time_exprには、HH[:MI[:SS[.n]]]、MI[:SS[.n]]またはSS[.n]の形式で時間を指定します(ここで、nは秒の小数部を指定します)。nの桁数が、fractional_seconds_precisionで指定した数より多い場合、nはfractional_seconds_precision値で指定した数に丸められます。先行フィールドがDAYの場合にのみ、1桁の整数および1つの空白の後にtime_exprを指定できます。
- leading_precisionは、先行フィールドの桁数です。0から9までの値を使用できます。デフォルトは2です。
- fractional_seconds_precisionは、SECOND日時フィールドの小数部の桁数です。有効範囲は1から9です。デフォルトは6です。

先行フィールドに対する制限:

後続のフィールドを指定する場合、先行のフィールドよりも下位の単位を指定する必要があります。たとえば、INTERVAL MINUTE TO DAYは無効です。このため、SECONDが先行フィールドの場合、期間リテラルは後続フィールドを持つことができません。

後続フィールドの有効範囲は次のとおりです。

- HOUR: 0から23
- MINUTE: 0から59
- SECOND: 0から59.999999999

様々なINTERVAL DAY TO SECONDリテラルの書式の例を次に示します。省略バージョンも含みます。

表2-14 INTERVAL DAY TO SECONDリテラルの書式

期間リテラルの形式	解釈
INTERVAL '4 5:12:10.222' DAY TO SECOND(3)	4 日、5 時間、12 分、10 秒、222 ミリ秒。
INTERVAL '4 5:12' DAY TO MINUTE	4 日 5 時間 12 分を示します。
INTERVAL '400 5' DAY(3) TO HOUR	400 日 5 時間を示します。
INTERVAL '400' DAY(3)	400 日を示します。
INTERVAL '11:12:10.2222222' HOUR TO SECOND(7)	11 時間 12 分 10.2222222 秒を示します。
INTERVAL '11:20' HOUR TO MINUTE	11 時間 20 分を示します。
INTERVAL '10' HOUR	10 時間を示します。
INTERVAL '10:22' MINUTE TO SECOND	10 分 22 秒を示します。
INTERVAL '10' MINUTE	10 分を示します。
INTERVAL '4' DAY	4 日を示します。
INTERVAL '25' HOUR	25 時間を示します。
INTERVAL '40' MINUTE	40 分を示します。
INTERVAL '120' HOUR(3)	120 時間を示します。
INTERVAL '30.12345' SECOND(2,4)	30.1235 秒を示します。精度は 4 のため、秒の小数部 '12345'は'1235'に丸められます。

DAY TO SECOND期間リテラルを別のDAY TO SECOND期間リテラルに加算または減算できます。たとえば。

```
INTERVAL'20' DAY - INTERVAL'240' HOUR = INTERVAL'10-0' DAY TO SECOND
```

書式モデル

書式モデルは、格納された日時または数値データの書式を文字列で記述する文字リテラルです。書式モデルによってデータベース内の値の内部表現が変更されることはありません。文字列を日付または数値に変換する場合、書式モデルによって、Oracle Databaseによる文字列の変換方法が決まります。SQL文では、書式モデルをTO_CHARファンクションやTO_DATEファンクションの引数として使用して、次の書式を指定できます。

- Oracleがデータベースから値を戻す場合に使用する書式
- Oracleがデータベースに格納するために指定した値の書式

たとえば:

- '17:45:29'の日時書式モデルは、'HH24:MI:SS'です。
- '11-Nov-1999'の日時書式モデルは、'DD-Mon-YYYY'です。
- '$2,304.25'の数値書式モデルは、'$9,999.99'です。

日時および数値書式モデルの要素のリストは、表2-15および表2-17を参照してください。

いくつかの書式の値は、初期化パラメータの値によって決まります。これらの書式要素によって戻される文字は、初期化パラメータNLS_TERRITORYを使用して暗黙的に指定することもできます。デフォルト日付書式をセッションごとに変更するには、ALTER SESSION文を使用します。

関連項目:

- これらのパラメータの値の変更については、「ALTER SESSION」を参照してください。書式モデルの使用例は、「書式モデルの例」を参照してください。
- 「TO_CHAR(日時)」、「TO_CHAR(数値)」および「TO_DATE」を参照してください。
- これらのパラメータの詳細は、『Oracle Databaseリファレンス』および『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

この項の後半では、次の書式モデルの使用方法について説明します。

- 数値書式モデル
- 日時書式モデル
- 書式モデル修飾子

数値書式モデル

次の関数で、数値書式モデルを使用できます。

- NUMBER、BINARY_FLOATまたはBINARY_DOUBLEデータ型の値をVARCHAR2データ型の値に変換するTO_CHARファンクション
- CHARまたはVARCHAR2データ型の値をNUMBERデータ型に変換するTO_NUMBERファンクション
- CHARおよびVARCHAR2式をBINARY_FLOATまたはBINARY_DOUBLEの値に変換するTO_BINARY_FLOATファンクションおよびTO_BINARY_DOUBLEファンクション

すべての数値書式モデルでは、数値が指定された有効桁数に丸められます。小数点左の有効桁数が書式で指定された桁数より多い場合、ポンド記号(#)が値のかわりに戻されます。通常、このイベントは、TO_CHARファンクションを制限的な数値書式文字列で使用して、丸め処理が行われた場合に発生します。

- 正のNUMBERの値が非常に大きく、指定の書式で表せない場合、無限大記号(~)が値のかわりに戻されます。同様に、負のNUMBERの値が非常に小さく、指定の書式で表せない場合、負の無限大記号(-~)が戻されます。
- BINARY_FLOATまたはBINARY_DOUBLEの値がCHARまたはNCHARに変換される場合、無限大またはNaN(非数値)のいずれかが入力されると、Oracleは常に値のかわりにポンド記号を戻します。ただし、書式モデルを省略すると、OracleはInfまたはNanを文字列として戻します。

数値書式の要素

数値書式モデルは、1つ以上の数値書式要素で構成されます。次の表に、数値書式モデルの要素とその例を示します。

数値書式モデルに書式要素MI、SまたはPRが指定されないかぎり、負の戻り値の先頭には自動的に負の符号が付けられ、正の戻り値の先頭には空白が付けられます。

表2-15 数値書式の要素

要素	例	説明
, (カンマ)	9,999	指定した位置にカンマを戻します。1 つの数値書式モデルに複数のカンマを指定できます。 制限事項: ● 数値書式モデルは、カンマ要素で始めることはできません。 ● 数値書式モデルでは、カンマを小数点文字やピリオドの右側に指定できません。
. (ピリオド)	99.99	指定した位置に小数点(ピリオド(.))を戻します。 制限事項: 1 つの数値書式モデルには 1 つのピリオドのみ指定できます。
$	$9999	値の前にドル記号を付けて戻します。
0	0999 9990	先行 0 (ゼロ)を戻します。 後続 0 (ゼロ)を戻します。
9	9999	指定された桁数の値を戻します。正の場合は先行空白が含まれ、負の場合は先行マイナスが含まれます。先行 0(ゼロ)は空白ですが、0(ゼロ)値の場合は固定小数点数の整数部分に 0(ゼロ)を戻します。
B	B9999	整数部が 0(ゼロ)の場合、書式モデル内の 0(ゼロ)にかかわらず、固定小数点数の整数部に対して空白を戻します。

要素	例	説明
C	C999	指定した位置に ISO 通貨記号(NLS_ISO_CURRENCY パラメータの現在の値)を戻します。
D	99D99	指定した位置に小数点文字(NLS_NUMERIC_CHARACTER パラメータの現在の値)を戻します。デフォルトはピリオド(.)です。 制限事項: 1 つの数値書式モデルには 1 つの小数点文字のみ指定できます。
EEEE	9.9EEEE	科学表記法で値を戻します。
G	9G999	指定した位置に桁区切り(NLS_NUMERIC_CHARACTER パラメータの現在の値)を戻します。単一の数値書式モデルに複数の桁区切りを指定できます。 制限事項: 数値書式モデルにおいて、桁区切りは小数点文字やピリオドの右側に指定できません。
L	L999	指定した位置にローカル通貨記号(NLS_CURRENCY パラメータの現在の値)を戻します。
MI	9999MI	負の値の後に負の符号(-)を戻します。 正の値の後に空白を戻します。 制限事項: 書式要素 MI は、数値書式モデルの最後の位置にのみ指定できます。
PR	9999PR	山カッコ<>の中に負の値を戻します。 正の値の前後に空白を付けて戻します。 制限事項: 書式要素 PR は、数値書式モデルの最後の位置にのみ指定できます。
RN rn	RN rn	大文字のローマ数字で値を戻します。 小文字のローマ数字で値を戻します。 値は 1 から 3999 の整数となります。
S	S9999 9999S	負の値の前に負の符号(-)を戻します。 正の値の前に正の符号(+)を戻します。 負の値の後に負の符号(-)を戻します。

要素	例	説明
		正の値の後に正の符号(+)を戻します。 制限事項: 書式要素 S は、数値書式モデルの最初または最後の位置にのみ指定できます。
TM	TM	テキストの最小数値書式モデルは、文字の最小数を(10 進出力で)戻します。この要素の大文字と小文字は区別されません。 デフォルトは TM9 で、出力が 64 文字を超えないかぎり、固定表記法で数値を戻します。戻り値が 64 文字を超える場合、Oracle Database は自動的に科学表記法で文字数を戻します。 制限事項: ● この要素の前に他の要素を指定することはできません。 ● この要素の後には、9、E(1 つ)または e(1 つ)のみを指定できます。次の文は、エラーを戻します。 `SELECT TO_CHAR(1234, 'TM9e') FROM DUAL;`
U	U9999	指定した位置にユーロ(または他の)第 2 通貨記号(NLS_DUAL_CURRENCY パラメータの現在の値によって決定される)を戻します。
V	999V99	値を 10 の n 乗にして戻します(必要に応じて数値を丸めます)。ここで、n は V の後の 9 の数です。
X	XXXX xxxx	指定の桁数の 16 進数値を戻します。指定した値が整数でない場合、Oracle Database はその値を整数に丸めます。 制限事項: ● この要素は、正の値または 0(ゼロ)のみを受け入れます。負の値の場合、エラーを戻します。 ● この要素の前には、0(先行 0(ゼロ)を戻す)または FM のみを指定できます。その他の要素の場合、エラーを戻します。0(ゼロ)または FM を X と同時に指定しないと、戻り値の前に常に空白が 1 つ追加されます。詳細は、書式モデル修飾子「FM」を参照してください。

表2-16に、異なる値のnumberと'fmt'に対して次の問合せを行った場合の結果を示します。

```
SELECT TO_CHAR(number, 'fmt')
  FROM DUAL;
```

表2-16 数値変換の結果

number	'fmt'	結果
-1234567890	9999999999S	'1234567890-'
0	99.99	' .00'
+0.1	99.99	' .10'
-0.2	99.99	' -.20'
0	90.99	' 0.00'
+0.1	90.99	' 0.10'
-0.2	90.99	' -0.20'
0	9999	' 0'
1	9999	' 1'
0	B9999	' '
1	B9999	' 1'
0	B90.99	' '
+123.456	999.999	' 123.456'
-123.456	999.999	'-123.456'
+123.456	FM999.009	'123.456'
+123.456	9.9EEEE	' 1.2E+02'
+1E+123	9.9EEEE	' 1.0E+123'
+123.456	FM9.9EEEE	'1.2E+02'
+123.45	FM999.009	'123.45'
+123.0	FM999.009	'123.00'

number	'fmt'	結果
+123.45	L999.99	' $123.45'
+123.45	FML999.99	'$123.45'
+1234567890	9999999999S	'1234567890+'

日時書式モデル

次のファンクションで日時書式モデルを使用できます。

- デフォルト書式以外の書式の文字値を日時値に変換するTO_*日時ファンクション(TO_*日時ファンクションとは、TO_DATE、TO_TIMESTAMPおよびTO_TIMESTAMP_TZです。)
- デフォルト書式以外の書式の日時値を文字値に変換する(たとえば、アプリケーションから日付を出力する)TO_CHARファンクション

日時書式モデルの合計長は最大22文字です。

デフォルト日時書式は、NLSセッション・パラメータNLS_DATE_FORMAT、NLS_TIMESTAMP_FORMATおよびNLS_TIMESTAMP_TZ_FORMATで明示的に指定することも、NLSセッション・パラメータNLS_TERRITORYで暗黙的に指定することもできます。セッションのデフォルトの日時書式は、ALTER SESSION文で変更できます。

関連項目:

NLSパラメータの詳細は、ALTER SESSIONおよび『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

日時書式要素

日時書式モデルは、表2-17に示す、1つ以上の日時書式要素で構成されます。

- 入力書式モデルの場合、同じ書式項目は2回指定できません。また、類似した情報を表す書式項目を組み合せることもできません。たとえば、'SYYYY'と'BC'を同一の書式文字列内で使用することはできません。
- 2番目の列は、書式要素がTO_*日付ファンクションで使用可能かどうかを示します。TO_CHARファンクションでは、すべての書式要素が使用できます。
- 日時書式要素FF、TZD、TZH、TZMおよびTZRは、タイムスタンプおよび期間書式モデルでは使用できますが、元のDATE書式モデルでは使用できません。
- 多くの場合、日時書式要素は一定の長さになるまで空白または先行0(ゼロ)が埋め込まれます。詳細は、書式モデル修飾子「FM」を参照してください。

ノート:

年要素は、短いものではなく 4 桁のもの(YYYY)を使用することをお薦めしますが、その理由は次のとおりです。

- 4 桁の年要素ならば、あいまいさが排除されます。
- 短い年要素は、問合せ最適化に影響を及ぼすことがあります(問合せコンパイルの時点では年が未知であり、実行時にならないと判明しないことがあるため)。

日付書式要素における大文字

スペルアウトした単語、省略形またはローマ数字での大文字の使用方法は、対応する書式要素での大文字の使用方法に従います。たとえば、日付書式モデル「DAY」は「MONDAY」、「Day」は「Monday」、「day」は「monday」を生成します。

日時書式モデルにおける句読点と文字リテラル

日付書式モデルでは、次の文字を指定できます。

- ハイフン、スラッシュ、カンマ、ピリオドおよびコロンなどの句読点
- 二重引用符で囲んだ文字リテラル

これらの文字は、書式モデル内と同じ位置で戻り値に表示されます。

表2-17 日時書式要素

要素	TO_*日時ファンクションで指定可能かどうか	説明

`-`
`/ , . ; :`
`"text"`
	可	句読点と引用符付きテキストは、結果で複製されます。

`AD` `A.D.`	可	ピリオド付き/なしで西暦を示します。
`AM` `A.M.`	可	ピリオド付き/なしで子午線インジケータを示します。
`BC` `B.C.`	可	ピリオド付き/なしで紀元前を示します。
`CC` `SCC`		世紀。 ● 4 桁の年の下 2 桁が 01 から 99(01 および 99 を含む)の場合、世紀はその年の上 2 桁に 1 を加えた値になります。 ● 4 桁の年の下 2 桁が 00 の場合、世紀はその年の上 2 桁と等しい値になります。 たとえば、2002 は 21 を戻し、2000 は 20 を戻します。

要素	TO_*日時ファンクションで指定可能かどうか	説明
D	可	曜日(1 から 7)。この要素はセッションの NLS 地域に依存します。
DAY	可	曜日。
DD	可	月における日(1 から 31)。
DDD	可	年における日(1 から 366)。
DL	可	Oracle Database の DATE 書式の拡張である長い日付書式の値(NLS_DATE_FORMAT パラメータの現在の値によって決定される)を戻します。日付の構成要素(曜日や月など)の表示が、NLS_TERRITORY パラメータおよび NLS_LANGUAGE パラメータによって決まるようにします。たとえば、AMERICAN_AMERICA ロケールでは、これは書式 'fmDay, Month dd, yyyy' を指定することと同じです。GERMAN_GERMANY ロケールでは、書式 'fmDay, dd.Month yyyy'。 制限: この書式は、TS 要素でのみ、空白で区切って指定可能です。
DS	可	短い日付書式の値を戻します。日付の構成要素(曜日や月など)の表示が、NLS_TERRITORY パラメータおよび NLS_LANGUAGE パラメータによって決まるようにします。たとえば、AMERICAN_AMERICA ロケールでは、'MM/DD/RRRR' 形式で指定したのと同じです。ENGLISH_UNITED_KINGDOM ロケールでは、'DD/MM/RRRR'形式で指定したのと同じです。 制限: この書式は、TS 要素でのみ、空白で区切って指定可能です。
DY	可	曜日の省略形。
E	可	年号(日本の元号暦、台湾暦およびタイ仏教暦)の省略形。
EE	可	年号(日本の元号暦、台湾暦およびタイ仏教暦)の完全形。
FF [1..9]	可	小数部。基数は出力されません。基数文字の追加には、X 書式要素を使用します。FF の後に 1 から 9 の数字を設定して、戻される日時値のミリ秒部分の桁数を指定します。数字を指定しない場合は、日時データ型に指定された精度またはデータ型のデフォルトの精度が使用されます。タイムスタンプ書式および期間書式では有効ですが、DATE 書式では有効ではありません。

要素	TO_*日時ファンクションで指定可能かどうか	説明
		例: 'HH:MI:SS.FF' SELECT TO_CHAR(SYSTIMESTAMP, 'SS.FF3') from DUAL;
FM	可	先行および後続する空白を持たない値を返します。 関連項目: FM
FX	可	文字データと書式モデルの完全一致を必要とします。 関連項目: FX
HH HH12	可	時間(1 から 12)。
HH24	可	時間(0 から 23)。
IW		ISO 8601 標準で定義されている年の暦週(1-52 または 1-53)。 ● 暦週は月曜日に始まります。 ● 年の最初の暦集は、1 月 4 日を含みます。 ● 年の最初の暦集は、12 月 19、30、31 日を含む場合があります。 ● 年の最後の暦集は、1 月 1、2、3 日を含む場合があります。
IYYY		ISO 8601 標準で定義されている暦週を含む 4 桁で表される年。
IYY IY I		ISO 8601 標準で定義されている暦週を含む、下 3 桁、2 桁、1 桁で表される年。
J	可	ユリウス日(紀元前 4712 年 1 月 1 日からの経過日数)。J で指定する数字は、整数である必要があります。
MI	可	分(0 から 59)。
MM	可	月(01 から 12。1 月は 01)。

要素	TO_*日時ファンクションで指定可能かどうか	説明
MON	可	月の名前の省略形。
MONTH	可	月の名前。
PM P.M.	可	ピリオド付き/なしで子午線インジケータを示します。
Q		年の四半期(1、2、3、4。1 月から 3 月は 1)。
RM	可	ローマ数字による月(I から XII。1 月は I)。
RR	可	2 桁のみを使用して、21 世紀に 20 世紀の日付を格納できます。 関連項目: RR 日時書式要素
RRRR	可	年。4 桁または 2 桁で入力できます。2 桁の場合、RR の場合と同様の結果が戻ります。この機能を使用しない場合は、4 桁の年を入力します。
SS	可	秒(0 から 59)。
SSSSS	可	午前 0 時から経過した秒(0 から 86399)。
TS	可	短い時間書式で値を戻します。時間の構成要素(時間や分など)の表示が、NLS_TERRITORY 初期化パラメータおよび NLS_LANGUAGE 初期化パラメータによって決まるようにします。 制限: この書式は、DL または DS 要素でのみ、空白で区切って指定可能です。
TZD	可	夏時間の情報。TZD の値は、夏時間の情報を持つタイムゾーン文字列の省略形です。TZR で指定した地域と対応している必要があります。タイムスタンプ書式および期間書式では有効ですが、DATE 書式では有効ではありません。 例: PST(アメリカ太平洋標準時)、PDT(アメリカ太平洋夏時間)。
TZH	可	タイムゾーンの時間。(TZM 書式要素を参照。)タイムスタンプ書式および期間書式では有効ですが、DATE 書式では有効ではありません。

要素	TO_*日時ファンクションで指定可能かどうか	説明
		例: 'HH:MI:SS.FFTZH:TZM'
TZM	可	タイムゾーンの分。(TZH 書式要素を参照。)タイムスタンプ書式および期間書式では有効ですが、DATE 書式では有効ではありません。 例: 'HH:MI:SS.FFTZH:TZM'
TZR	可	タイムゾーン地域の情報。値は、データベースでサポートされるタイムゾーン地域名である必要があります。タイムスタンプ書式および期間書式では有効ですが、DATE 書式では有効ではありません。 例: US/Pacific
WW		年における週(1 から 53)。第 1 週はその年の 1 月 1 日で始まり、1 月 7 日で終了します。
W		月における週(1 から 5)。第 1 週はその月の 1 日で始まり、7 日で終了します。
X	可	ローカル基数文字。 例: 'HH:MI:SSXFF'
Y,YYY	可	指定の位置にカンマを付けた年。
YEAR SYEAR		スペルアウトした年。S の場合、紀元前の日付の先頭にマイナス記号(-)が付加されます。
YYYY SYYYY	可	4 桁の年。S の場合、紀元前の日付の先頭にマイナス記号が付加されます。
YYY YY Y	可	年の最後の 3 桁、2 桁または 1 桁。

Oracle Databaseは、一定の柔軟性によって文字列を日付に変換します。たとえば、TO_DATEファンクションの使用時、句読点文字を含む書式モデルと、句読点文字がないか、一部を使用した入力文字列は同じになり、入力文字列の各数値要素に最大許容桁数が含まれます(たとえば、'MM'に2桁の'05'、'YYYY'に4桁の'2007')。次の文は、エラーを戻しません。

```
SELECT TO_CHAR(TO_DATE('0207','MM/YY'), 'MM/YY') FROM DUAL;
TO_CH
-----
02/07
```

次の書式文字列はエラーを戻しませんが、FX(厳密な書式一致)書式修飾子では、式および書式文字列が完全に一致する必要があります。

```
SELECT TO_CHAR(TO_DATE('0207', 'fxmm/yy'), 'mm/yy') FROM DUAL;
SELECT TO_CHAR(TO_DATE('0207', 'fxmm/yy'), 'mm/yy') FROM DUAL;
                       *
ERROR at line 1:
ORA-01861: literal does not match format string
```

この書式モデルでは、英数字以外の任意の1文字を句読点文字に一致させることができます。たとえば、次の文は、エラーを戻しません。

```
SELECT TO_CHAR (TO_DATE('02#07','MM/YY'), 'MM/YY') FROM DUAL;
TO_CH
-----
02/07
```

関連項目:

詳細は、書式モデルの修飾子および文字列から日付への変換に関する規則を参照してください。

日時書式要素およびグローバリゼーション・サポート

いくつかの日時書式要素の機能は、Oracle Databaseを使用している国および言語に依存します。たとえば、次の日時書式要素は、フルスペルで値が戻されます。

- MONTH
- MON
- DAY
- DY
- BC、AD、B.C.またはA.D.
- AM、PM、A.M.またはP.M.

これらの値を戻す言語は、初期化パラメータNLS_DATE_LANGUAGEによって明示的に指定することも、初期化パラメータNLS_LANGUAGEによって暗黙的に指定することもできます。日時書式要素YEARとSYEARによって戻される値は常に英語です。

日時書式要素Dは、週の何日目であるか(1から7)を返します。週の第1日目は、初期化パラメータNLS_TERRITORYで暗黙的に指定されます。

関連項目:

グローバリゼーション・サポートの初期化パラメータの詳細は、『Oracle Databaseリファレンス』および『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

ISO標準日付書式要素

Oracleでは、ISO規格に従って、日時書式要素IYYY、IYY、IY、IおよびIWによって返された値が計算されます。これらの値と、日時書式要素YYYY、YYY、YY、YおよびWWによって返された値との違いについては、『Oracle Databaseグローバリゼーション・サポート・ガイド』のグローバリゼーション・サポートの説明を参照してください。

RR日時書式要素

RR日時書式要素は、YY日時書式要素に似ていますが、より柔軟に、他の世紀に日付の値を格納できます。RR日時書式要素では、年の下2桁のみを指定することにより、20世紀の日付を21世紀に格納できます。

TO_DATEファンクションとYY日時書式要素を使用する場合、返される年の上2桁は常に、現在の年と同一になります。かわりにRR日時書式要素を使用する場合、戻り値の世紀は、指定した2桁の年および現在の年の下2桁によって異なります。

つまり、次のようになります。

- 指定した2桁の年が00から49である場合、次のようになります。
    - 現在の年の下2桁が00から49の場合、返される年の上2桁は、現在の年と同一になる。
    - 現在の年の下2桁が50から99の場合、返される年の上2桁は、現在の年の上2桁に1を加えた値になる。
- 指定した2桁の年が50から99である場合、次のようになります。
    - 現在の年の下2桁が00から49の場合、返される年の上2桁は、現在の年の上2桁から1を引いた値になる。
    - 現在の年の下2桁が50から99の場合、返される年の上2桁は、現在の年と同一になる。

次に、RR日時書式要素の特長を具体的に説明します。

RR日時書式の例

次の問合せが、1950年から1999年の間に発行されるとします。

```
SELECT TO_CHAR(TO_DATE('27-OCT-98', 'DD-MON-RR'), 'YYYY') "Year" FROM DUAL;
Year
----
1998
SELECT TO_CHAR(TO_DATE('27-OCT-17', 'DD-MON-RR'), 'YYYY') "Year" FROM DUAL;
Year
----
2017
```

次の問合せが、2000年から2049年の間に発行されるとします。

```
SELECT TO_CHAR(TO_DATE('27-OCT-98', 'DD-MON-RR'), 'YYYY') "Year" FROM DUAL;
Year
----
1998
SELECT TO_CHAR(TO_DATE('27-OCT-17', 'DD-MON-RR'), 'YYYY') "Year" FROM DUAL;
Year
----
2017
```

発行される年(2000年の前後)にかかわらず、問合せが同じ値を戻していることに注目してください。RR日時書式要素によって、年の上2桁が異なっても同じ値を戻すSQL文を記述できます。

日時書式要素の接尾辞

表2-18に、日時書式要素に付加できる接尾辞を示します。

表2-18 日時書式要素の接尾辞

接尾辞	意味	要素の例	値の例
TH	序数	DDTH	4TH
SP	フルスペルで表した数	DDSP	FOUR
SPTH または THSP	フルスペルで表した序数	DDSPTH	FOURTH

日時書式要素の接尾辞のノート:

- 前述の接尾辞のいずれかを日時書式要素に追加する場合、戻り値は必ず英語になります。
- 日時の接尾辞は、書式の出力にのみ有効です。データベースに日付を挿入するためには使用できません。

書式モデルの修飾子

TO_CHARファンクションの書式モデルで修飾子FMとFXを使用して、空白の埋め方および書式検査を制御できます。

同じ修飾子が1つの書式モデルに2回以上出現してもかまいません。その場合は、2回目以降の出現のたびにその修飾子の効果は逆になります。その効果は、モデルの中の最初の出現に続く部分に対しては有効になり、2番目の出現に続く部分に対しては無効になり、3番目の出現に続く部分に対しては再び有効になり、以降も同様です。

FM

Fill mode (埋込みモード)です。書式要素が固定幅になるまで後続空白文字と先行0(ゼロ)が埋め込まれます。幅は、関連する書式モデルの最大要素の表示幅と等しくなります。

- 数値要素には、その要素に指定できる最大値の幅まで先行0(ゼロ)が埋め込まれます。たとえば、YYYY要素は4桁(9999の長さ)まで、HH24は2桁(23の長さ)まで、DDDは3桁(366の長さ)まで埋め込まれます。
- 文字要素MONTH、MON、DAYおよびDYには、NLS_DATE_LANGUAGEパラメータおよびNLS_CALENDARパラメータの値によって指定される有効な名前の中で、それぞれ最長のフルスペルの月名、最長の省略形の月名、最長のフルスペルの曜日または最長の省略形の曜日の幅まで、後続空白が埋め込まれます。たとえば、NLS_DATE_LANGUAGEがAMERICANでNLS_CALENDARがGREGORIAN(デフォルト)の場合、MONTHの最大要素はSEPTEMBERであるため、MONTH書式要素のすべての値は表示文字が9文字になるまで埋め込まれます。NLS_DATE_LANGUAGEパラメータおよびNLS_CALENDARパラメータの値は、TO_CHARおよびTO_*日時ファンクションの3番目の引数で指定されるか、または現行セッションのNLS環境から取得されます。
- 文字要素RMには、長さ4(viiiの長さ)まで後続空白が埋め込まれます。
- その他の文字要素とフルスペルで表した数(SP、SPTHおよびTHSP接尾辞)には埋め込まれません。

FM修飾子を使用すると、TO_CHARファンクションの戻り値で前述の埋込みは行われなくなります。

FX

Format exact (厳密な書式一致)です。この修飾子は、TO_DATEファンクションの文字引数と日時書式モデルに対して、厳密な一致を指定します。

- 文字引数における句読点と引用符で囲まれたテキストは、書式モデルの対応する部分と(大/小文字の違いを除いて)厳密に一致する必要があります。
- 文字引数には余分な空白を含めることはできません。FXを指定していない場合、Oracleは余分な空白を無視します。
- 文字引数における数値データの桁数は、書式モデルの対応する要素と同じ桁数である必要があります。FXを指定していない場合、文字引数における数値によっては先行0(ゼロ)が省略されます。

  FXが使用可能になっているとき、FM修飾子を指定して、この先行0(ゼロ)のチェックを使用禁止にできます。

文字引数の位置がこれらの条件に違反する場合、Oracleはエラー・メッセージを戻します。

書式モデルの例

次の文は、日付書式モデルを使用して文字式を戻します。

```
SELECT TO_CHAR(SYSDATE, 'fmDDTH') || ' of ' ||
       TO_CHAR(SYSDATE, 'fmMonth') || ', ' ||
       TO_CHAR(SYSDATE, 'YYYY') "Ides"
  FROM DUAL;
Ides
------------------
3RD of April, 2008
```

この文はFM修飾子も使用していることに注目してください。FMを指定しないと、月は、次のように空白を埋め込んで9文字にして戻されます。

```
SELECT TO_CHAR(SYSDATE, 'DDTH') || ' of ' ||
   TO_CHAR(SYSDATE, 'Month') || ', ' ||
   TO_CHAR(SYSDATE, 'YYYY') "Ides"
  FROM DUAL;
Ides
-----------------------
03RD of April    , 2008
```

次の文は、2つの連続した一重引用符を含む日付書式モデルを使用することによって、戻り値に一重引用符を含めます。

```
SELECT TO_CHAR(SYSDATE, 'fmDay') || '''s Special' "Menu"
  FROM DUAL;
Menu
-----------------
Tuesday's Special
```

書式モデル内の文字リテラルにおいても、同じ目的で一重引用符を2つ連続して使用できます。

表2-19に、char値と'fmt'の様々な組合せに対し、次の文がFXを使用した一致条件を満たしているかどうかを示します(tableという名前の表にはDATEデータ型の列date_columnがあります)。

```
UPDATE table
  SET date_column = TO_DATE(char, 'fmt');
```

表2-19 FX書式モデル修飾子による文字データと書式モデルの一致

char	'fmt'	一致とエラー
'15/ JAN /1998'	'DD-MON-YYYY'	Match
' 15! JAN % /1998'	'DD-MON-YYYY'	Error
'15/JAN/1998'	'FXDD-MON-YYYY'	Error
'15-JAN-1998'	'FXDD-MON-YYYY'	Match
'1-JAN-1998'	'FXDD-MON-YYYY'	Error
'01-JAN-1998'	'FXDD-MON-YYYY'	Match
'1-JAN-1998'	'FXFMDD-MON-YYYY'	Match

戻り値の書式: 例

書式モデルを使用して、データベースから値を戻すために使用するOracleの書式を指定できます。

次の文は、部門80の従業員の給与を選択し、TO_CHARファンクションを使用して、その給与を数値書式モデル'$99,990.99'で指定した書式の文字値に変換します。

```
SELECT last_name employee, TO_CHAR(salary, '$99,990.99')
  FROM employees
  WHERE department_id = 80;
```

Oracleはこの書式モデルによって、ドル記号を先頭に付け、3桁ごとにカンマで区切り、小数点以下2桁を持つ給与を戻します。

次の文は、部門20の各従業員の入社した日付を選択し、TO_CHARファンクションを使用して、その日付を日付書式モデル'fmMonth DD, YYYY'で指定した書式の文字列に変換します。

```
SELECT last_name employee, TO_CHAR(hire_date,'fmMonth DD, YYYY') hiredate
  FROM employees
  WHERE department_id = 20;
```

Oracleはこの書式モデルによって、2桁の日、世紀も含めた4桁の年で示された入社日付(fmで指定)を空白で埋めないで戻します。

関連項目:

fm書式要素の詳細は、書式モデルの修飾子を参照してください。

正しい書式モデルの指定: 例

列の値を挿入または更新する場合、指定する値のデータ型が列の列データ型に対応する必要があります。書式モデルを使用して、あるデータ型の値を列が必要とする別のデータ型の値に変換する書式を指定できます。

たとえば、DATE列に挿入する値は、DATEデータ型の値か、またはデフォルト日付書式の文字列値である必要があります(Oracleは、暗黙的にデフォルト日付書式の文字列をDATEデータ型に変換します)。値が別の書式で与えられる場合、

TO_DATEファンクションを使用して値をDATEデータ型に変換する必要があります。また、文字列の書式を指定する場合にも、書式モデルを使用する必要があります。

次の文は、TO_DATEファンクションを使用してHunoldの入社日を更新します。文字列'2008 05 20'をDATE値に変換するために、書式マスク'YYYY MM DD'を指定します。

```
UPDATE employees
  SET hire_date = TO_DATE('2008 05 20','YYYY MM DD')
  WHERE last_name = 'Hunold';
```

文字列から日付への変換に関する規則

次の追加の書式化規則は、文字列値を日付値に変換する場合に適用されます(ただし、書式モデルで修飾子FXまたはFXFMを使用して書式検査を制御した場合は適用できません)。

- 先行0(ゼロ)を含む数値書式要素の桁がすべて指定されている場合は、日付文字列から書式文字列に含まれる句読点を省略できます。たとえば、MM、DD、YYなどの2桁の書式要素については、2のかわりに02を指定した場合です。
- 日付文字列から、書式文字列の最後にある時刻フィールドを省略できます。
- 日付文字列では、英数字以外の任意の1文字を使用して、書式文字列の句読点記号に一致させることができます。
- 日時書式要素と日付文字列内の対応する文字の一致に失敗した場合、表2-20に示すとおり、元の書式要素のかわりに、別の書式要素の適用が試みられます。

表2-20 Oracleの書式一致

元の書式要素	元の書式要素のかわりに試行する書式要素
'MM'	'MON'および'MONTH'
'MON	'MONTH'
'MONTH'	'MON'
'YY'	'YYYY'
'RR'	'RRRR'

XML書式モデル

SYS_XMLAggファンクションとSYS_XMLGenファンクション(非推奨)は、XML文書を格納するXMLType型のインスタンスを返します。Oracleには、これらのファンクションの出力を書式設定できるXMLFormatオブジェクトが用意されています。

表2-21に、XMLFormatオブジェクトの属性を示します。表に示すように、ファンクションはこの型を実装します。

関連項目:

- SYS_XMLAggファンクションの詳細は、「SYS_XMLAGG」を参照してください。
- SYS_XMLGenファンクションの詳細は、「SYS_XMLGEN」を参照してください。
- XMLFormatオブジェクトの実装およびその使用の詳細は、『Oracle XML DB開発者ガイド』を参照してください。

表2-21 XMLFormatオブジェクトの属性

属性	データ型	目的
enclTag	VARCHAR2(4000)またはVARCHAR2(32767)脚注 1	SYS_XMLAgg ファンクションまたは SYS_XMLGen ファンクション(非推奨)の結果の囲みタグの名前です。 SYS_XMLAgg: デフォルトは ROWSET です。 SYS_XMLGen: ファンクションへの入力が列名の場合、その列名がデフォルトになります。それ以外の場合、デフォルトは ROW です。schemaType を USE_GIVEN_SCHEMA に設定すると、この属性によって XML スキーマの要素名を指定することもできます。
schemaType	VARCHAR2(100)	出力された文書のスキーマ生成の型です。有効な値は、'NO_SCHEMA'および'USE_GIVEN_SCHEMA'です。デフォルトは、'NO_SCHEMA'です。
schemaName	VARCHAR2(4000)またはVARCHAR2(32767)脚注 1	schemaType の値が'USE_GIVEN_SCHEMA'のときに、Oracle が使用するターゲット・スキーマの名前です。schemaName を指定すると、囲みタグが要素名として使用されます。
targetNameSpace	VARCHAR2(4000)またはVARCHAR2(32767)脚注 1	スキーマが指定されている場合(つまり、schemaType が GEN_SCHEMA_*または USE_GIVEN_SCHEMA の場合)のターゲット・ネームスペースです。
dburlPrefix	VARCHAR2(4000)またはVARCHAR2(32767)脚注 1	WITH_SCHEMA が指定されている場合に使用するデータベースの URL です。この属性が指定されていない場合、Oracle はその型への URL を相対 URL 参照として宣言します。
processingIns	VARCHAR2(4000)またはVARCHAR2(32767)脚注 1	ファンクションの出力の最上位に、要素の前に追加されるユーザーの処理命令です。

脚注 1

この属性のデータ型は、初期化パラメータがMAX_STRING_SIZE = STANDARDの場合はVARCHAR2(4000)、

MAX_STRING_SIZE = EXTENDEDの場合はVARCHAR2(32767)です。詳細は、拡張データ型を参照してください。

XMLFormatオブジェクトを実装するファンクションは、次のとおりです。

```
STATIC FUNCTION createFormat(
     enclTag IN varchar2 := 'ROWSET',
     schemaType IN varchar2 := 'NO_SCHEMA',
     schemaName IN varchar2 := null,
     targetNameSpace IN varchar2 := null,
     dburlPrefix IN varchar2 := null,
     processingIns IN varchar2 := null) RETURN XMLGenFormatType
       deterministic parallel_enable,
  MEMBER PROCEDURE genSchema (spec IN varchar2),
  MEMBER PROCEDURE setSchemaName(schemaName IN varchar2),
  MEMBER PROCEDURE setTargetNameSpace(targetNameSpace IN varchar2),
  MEMBER PROCEDURE setEnclosingElementName(enclTag IN varchar2),
  MEMBER PROCEDURE setDbUrlPrefix(prefix IN varchar2),
  MEMBER PROCEDURE setProcessingIns(pi IN varchar2),
  CONSTRUCTOR FUNCTION XMLGenFormatType (
     enclTag IN varchar2 := 'ROWSET',
     schemaType IN varchar2 := 'NO_SCHEMA',
     schemaName IN varchar2 := null,
     targetNameSpace IN varchar2 := null,
     dbUrlPrefix IN varchar2 := null,
     processingIns IN varchar2 := null) RETURN SELF AS RESULT
      deterministic parallel_enable,
  STATIC function createFormat2(
      enclTag in varchar2 := 'ROWSET',
      flags in raw) return sys.xmlgenformattype
      deterministic parallel_enable
);
```

NULL

行のある列の値がない場合、その列はNULLである、またはNULLを含むといいます。NOT NULL整合性制約またはPRIMARY KEY整合性制約によって制限されていない列の場合は、どのデータ型の列でもNULLを含むことができます。実際のデータ値が不定または値に意味がない場合に、NULLを使用してください。

Oracle Databaseは、長さが0(ゼロ)の文字値をNULLとして処理します。ただし、数値0を表すのにnullを使用しないでください。この2つは等価ではありません。

ノート:

Oracle Database は現在、長さが 0(ゼロ)の文字値を NULL として処理します。ただし、将来のリリースではこの処理が変更される場合があるため、空の文字列を null と同じように処理しないことをお薦めします。

nullを含む算術式は常にnullに評価されます。たとえば、10にnullを加算したものはnullです。実際、nullオペランドを指定すると、すべての演算子ではnullが返されます(連結を除く)。

SQLファンクションでのNULL

SQLファンクションでのNULL処理の詳細は、SQLファンクションでのNULLを参照してください。

比較条件でのNull

NULLを検査するには、比較条件IS NULLおよびIS NOT NULLのみを使用します。それ以外の条件をnullで使用すると、結果がnull値に依存し、結果がUNKNOWNになります。NULLはデータの欠落を表すため、任意の値や別のNULLとの関係で等号や不等号は成り立ちません。ただし、OracleはDECODEファンクションを評価するときに2つのNULLを等しい値とみなします。構文および追加情報については、「DECODE」を参照してください。

コンポジット・キーの場合、2つのNULLは等しいと判断されます。NULLを含む2つのコンポジット・キーは、そのキーのNULL以外のコンポーネントのすべてが等しい場合、同一であると判断されます。

条件でのNULL

UNKNOWNに評価される条件はFALSEとほぼ同様に動作します。たとえば、UNKNOWNと評価される条件をWHERE句に持つSELECT文からは、行が戻されません。ただし、UNKNOWNに評価される条件は、UNKNOWN条件の評価でさらに操作を行う場合はUNKNOWNに評価されるのでFALSEとは異なります。このため、NOT FALSEはTRUEに評価されますが、NOT UNKNOWNはUNKNOWNに評価されます。

表2-22は、条件でnullを使用する場合の様々な評価の例を示しています。SELECT文のWHERE句でUNKNOWNに評価される条件を使用する場合、問合せで返される行はありません。

表2-22 Nullを使用する条件

条件	aの値	評価
a IS NULL	10	FALSE

条件	aの値	評価
a IS NOT NULL	10	TRUE
a IS NULL	NULL	TRUE
a IS NOT NULL	NULL	FALSE
a = NULL	10	UNKNOWN
a != NULL	10	UNKNOWN
a = NULL	NULL	UNKNOWN
a != NULL	NULL	UNKNOWN
a = 10	NULL	UNKNOWN
a != 10	NULL	UNKNOWN

NULLを含む論理条件の結果を示した真理値表は、表6-5、表6-6、および表6-7を参照してください。

コメント

次の2種類のコメントを作成できます。

- SQL文中のコメントは、SQL文を実行するアプリケーション・コードの一部として格納されます。
- 個別のスキーマ・オブジェクトまたは非スキーマ・オブジェクトに付けたコメントは、オブジェクト自体のメタデータとともにデータ・ディクショナリに格納されます。

SQL文中のコメント

コメントは、アプリケーションを読みやすく、メンテナンスしやすくします。たとえば、文にはアプリケーションでのその文の目的を記述したコメントを含めることができます。SQL文中のコメントは文の実行には影響しませんが、ヒントは例外です。この特殊なコメント形式を使用する場合の詳細は、ヒントを参照してください。

コメントは、文中のキーワード、パラメータまたは句読点の間に入れることができます。次のいずれかの方法を使用します。

- スラッシュとアスタリスク(/*)を使用してコメントを開始します。コメントのテキストを続けます。このテキストは複数行にまたがってもかまいません。アスタリスクとスラッシュ(*/)を使用してコメントを終了します。開始文字と終了文字は、空白や改行によってテキストから切り離す必要はありません。
- --(ハイフン2個)を使用してコメントを開始します。コメントのテキストを続けます。このテキストは複数行にまたがることはできません。改行によってコメントを終了します。

SQLの入力に使用するツール製品には、追加の制限事項があるものもあります。たとえば、SQL*Plusを使用している場合、デフォルトでは複数行のコメント内に空白行を入れることはできません。詳細は、データベースのインタフェースとして使用するツール製品のドキュメントを参照してください。

SQL文の中に両方のスタイルのコメントが複数あってもかまいません。コメントのテキストには、使用しているデータベース文字セットの印字可能文字を含めることができます。

例

次の文には多くのコメントが含まれています。

```
SELECT last_name, employee_id, salary + NVL(commission_pct, 0),
       job_id, e.department_id
  /* Select all employees whose compensation is
  greater than that of Pataballa.*/
  FROM employees e, departments d
  /*The DEPARTMENTS table is used to get the department name.*/
  WHERE e.department_id = d.department_id
    AND salary + NVL(commission_pct,0) >   /* Subquery:       */
      (SELECT salary + NVL(commission_pct,0)
        /* total compensation is salary + commission_pct */
        FROM employees
        WHERE last_name = 'Pataballa')
  ORDER BY last_name, employee_id;
SELECT last_name,                                    -- select the name
       employee_id                                   -- employee id
       salary + NVL(commission_pct, 0),              -- total compensation
       job_id,                                       -- job
       e.department_id                               -- and department
  FROM employees e,                                  -- of all employees
       departments d
  WHERE e.department_id = d.department_id
    AND salary + NVL(commission_pct, 0) >            -- whose compensation
                                                     -- is greater than
```

```
        (SELECT salary + NVL(commission_pct,0)      -- the compensation
          FROM employees
          WHERE last_name = 'Pataballa')            -- of Pataballa
  ORDER BY last_name                                -- and order by last name
           employee_id                              -- and employee id.
;
```

スキーマ・オブジェクトおよび非スキーマ・オブジェクトに関するコメント

COMMENTコマンドを使用して、スキーマ・オブジェクト(表、ビュー、マテリアライズド・ビュー、演算子、索引タイプ、マイニング・モデル)または非スキーマ・オブジェクト(エディション)にCOMMENTコマンドでコメントを付けることができます。表スキーマ・オブジェクトの一部である列にもコメントを作成できます。スキーマ・オブジェクトおよび非スキーマ・オブジェクトに付けたコメントは、データ・ディクショナリに格納されます。このコメントの形式の詳細は、「COMMENT」を参照してください。

ヒント

ヒントとは、Oracle Databaseのオプティマイザに指示を与えるためにSQL文中に記述するコメントのことです。オプティマイザは、オプティマイザの動作を阻止する条件が存在しないかぎり、これらのヒントを使用して文の実行計画を選択します。

ヒントが導入されたのはOracle7からですが、当時は、オプティマイザによって生成された計画が最善ではない場合にユーザーが取れる手段はほとんどありませんでした。現在では、オプティマイザでは解決されないパフォーマンスの問題の解決に役立つように、OracleはSQLチューニング・アドバイザ、SQL計画管理、SQLパフォーマンス・アナライザをはじめとする多数のツールを提供しています。ヒントではなく、これらのツールを使用することをお薦めします。これらのツールはヒントに比べてはるかに優れていますが、その理由は、これらのツールを継続的に使用すれば、データやデータベース環境の変化に合せた最新の解決策が得られるからです。

ヒントの多用は避けるとともに、使用する場合は必ず事前に、関係する表の統計情報を収集して、ヒントなしのオプティマイザ計画をEXPLAIN PLAN文を使用して評価してください。データベースの状態の変化や以降のリリースにおける問合せパフォーマンスの向上によって、コード内のヒントがパフォーマンスに及ぼす影響が大きく変化することもあります。

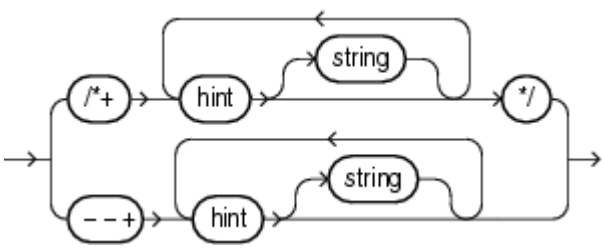
この後は、使用頻度の高い一部のヒントについて説明します。他の高度なチューニング・ツールではなくヒントを使用する場合は、ヒントの使用によって短期的な効果が得られても、長期にわたるパフォーマンスの向上にはつながらない場合があることに注意してください。

ヒントの使用方法

文ブロックはヒントを含むコメントを1つだけ持つことができ、SELECT、UPDATE、INSERT、MERGEまたはDELETEの各キーワードに続けてコメントを指定します。

次の構文図は、Oracleが文ブロック内でサポートする両方のスタイルのコメントに含まれるヒントの構文です。ヒント構文は、文ブロックを開始するINSERT、UPDATE、DELETE、SELECTまたはMERGEのいずれかのキーワードの直後でのみ指定できます。

hint::=



説明:

- +(プラス記号)は、コメントをヒントのリストとして、Oracleに解析させます。プラス記号は、コメント・デリミタの直後に置く必要があります。空白を入れてはいけません。
- hintは、この項で説明するヒントの1つです。プラス記号とヒントの間の空白は入れても入れなくてもかまいません。コメントに複数のヒントが含まれている場合は、1つ以上の空白で区切る必要があります。
- stringは、ヒントに含めることができるその他のコメント・テキストです。

--+構文では、コメント全体を単一行で指定する必要があります。

Oracle Databaseは、次の状況ではヒントを無視し、エラーを戻しません。

- ヒントにスペルの誤りまたは構文エラーがある場合。ただし、データベースは、同一コメント内に正しく指定された他のヒントがある場合はそれを採用します。
- ヒントを含むコメントが、DELETE、INSERT、MERGE、SELECTまたはUPDATEのいずれかのキーワードの後で指定していない場合。
- ヒントの組合せが互いに競合している場合。ただし、データベースは、同一コメント内に他のヒントがあればそれを採用します。
- データベース環境が、Formsバージョン3トリガー、Oracle Forms 4.5、Oracle Reports 2.5など、PL/SQLバージョン1を使用している場合。
- 1つのグローバル・ヒントが複数の問合せブロックを参照している場合。詳細は、1つのグローバル・ヒントでの複数問合せブロックの指定を参照してください。

19cでは、DBMS_XPLANを使用してヒントが使用されるかどうかを判定できます。詳細は、 Oracle Database SQLチューニング・ガイドを参照してください。

ヒントでの問合せブロックの指定

多くのヒントでオプションの問合せブロック名を指定して、ヒントが適用される問合せブロックを指定できます。この構文によって、インライン・ビューに適用されるヒントを外部問合せ内で指定できます。

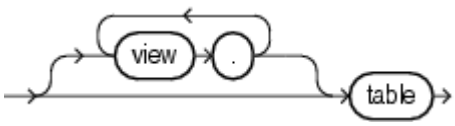
問合せブロックの引数の構文は、@queryblockの形式になります。ここでqueryblockは問合せ内の問合せブロックを指定する識別子です。queryblock識別子は、システムで生成されたものでも、ユーザーが指定したものでもかまいません。ヒントを、そのヒントが適用される問合せブロック自体の中で指定するときは、@queryblock構文を省略してください。

- システム生成の識別子は、問合せに対するEXPLAIN PLAN文を使用して取得できます。変換前の問合せブロック名は、NO_QUERY_TRANSFORMATIONヒントを使用している問合せに対してEXPLAIN PLANを実行することで調べることができます。NO_QUERY_TRANSFORMATIONヒントを参照してください。
- ユーザー指定の名前はQB_NAMEヒントで指定できます。QB_NAMEヒントを参照してください。

グローバル・ヒントの指定

多くのヒントは、特定の表や索引に適用することも、ビュー内の表や、索引の一部である列によりグローバルに適用することもできます。このようなグローバル・ヒントは、構文要素tablespecおよびindexspecによって定義します。

tablespec::=

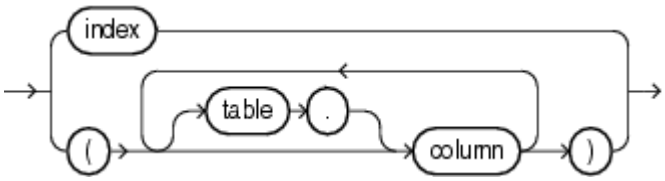


アクセスする表は、文で示されるとおり正確に指定する必要があります。文が表の別名を使用している場合は、ヒントでも表名を使用せずに別名を使用します。ただし、文でスキーマ名を指定している場合でも、ヒント内ではスキーマ名を表名に含めないでください。

ノート:

解析時にオプティマイザが追加のビューを生成するため、tablespec 句を使用したグローバル・ヒントの指定は、ANSI 結合を使用する問合せに対しては無効です。かわりに、@queryblock を指定してヒントが適用される問合せブロックを指定してください。

indexspec::=



ヒントの指定の中で、tablespecの後にindexspecがある場合は、表名と索引名を区切るカンマを使用できますが、このカンマは必須ではありません。indexspecが複数回出現するときも、区切るためのカンマを使用できますが、必須ではありません。

1つのグローバル・ヒントでの複数問合せブロックの指定

Oracle Databaseは、複数の問合せブロックを参照するグローバル・ヒントを無視します。この問題を回避するために、ヒントの中ではtablespecとindexspecを使用するかわりにオブジェクト別名を指定することをお薦めします。

たとえば、次のビューvと表tがあるとします。

```
CREATE VIEW v AS
  SELECT e.last_name, e.department_id, d.location_id
  FROM employees e, departments d
  WHERE e.department_id = d.department_id;

CREATE TABLE t AS
  SELECT * from employees
  WHERE employee_id < 200;
```

ノート:

次に示す例では EXPLAIN PLAN 文が使用されており、これによって実行計画が表示されるので、ヒントが遵守されているか無視されているかがわかります。詳細は、「EXPLAIN PLAN」を参照してください。

次の問合せのLEADINGヒントは無視されますが、これは複数の問合せブロック(表tが含まれるメイン問合せブロックと、ビュー問合せブロックv)を参照しているためです。

```
EXPLAIN PLAN
  SET STATEMENT_ID = 'Test 1'
  INTO plan_table FOR
    (SELECT /*+ LEADING(v.e v.d t) */ *
     FROM t, v
```

```
     WHERE t.department_id = v.department_id);
```

次のSELECT文を実行すると実行計画が戻され、これを見るとLEADINGヒントが無視されたことがわかります。

```
SELECT id, LPAD(' ',2*(LEVEL-1))||operation operation, options, object_name,
object_alias
  FROM plan_table
  START WITH id = 0 AND statement_id = 'Test 1'
  CONNECT BY PRIOR id = parent_id AND statement_id = 'Test 1'
  ORDER BY id;
 ID OPERATION            OPTIONS    OBJECT_NAME   OBJECT_ALIAS
--- -------------------- ---------- ------------- --------------------
  0 SELECT STATEMENT
  1   HASH JOIN
  2     HASH JOIN
  3       TABLE ACCESS   FULL       DEPARTMENTS   D@SEL$2
  4       TABLE ACCESS   FULL       EMPLOYEES     E@SEL$2
  5     TABLE ACCESS     FULL       T             T@SEL$1
```

次の問合せのLEADINGヒントは遵守されますが、これはオブジェクト別名を参照しているからであり、このことは前の問合せで戻された実行計画に現れています。

```
EXPLAIN PLAN
  SET STATEMENT_ID = 'Test 2'
  INTO plan_table FOR
    (SELECT /*+ LEADING(E@SEL$2 D@SEL$2 T@SEL$1) */ *
     FROM t, v
     WHERE t.department_id = v.department_id);
```

次のSELECT文から戻される実行計画は、LEADINGヒントが遵守されたことを示しています。

```
SELECT id, LPAD(' ',2*(LEVEL-1))||operation operation, options,
  object_name, object_alias
  FROM plan_table
  START WITH id = 0 AND statement_id = 'Test 2'
  CONNECT BY PRIOR id = parent_id AND statement_id = 'Test 2'
  ORDER BY id;
 ID OPERATION            OPTIONS    OBJECT_NAME   OBJECT_ALIAS
--- -------------------- ---------- ------------- --------------------
  0 SELECT STATEMENT
  1   HASH JOIN
  2     HASH JOIN
  3       TABLE ACCESS   FULL       EMPLOYEES     E@SEL$2
  4       TABLE ACCESS   FULL       DEPARTMENTS   D@SEL$2
  5     TABLE ACCESS     FULL       T             T@SEL$1
```

関連項目:

『Oracle Database SQLチューニング・ガイド』では、ヒントおよびEXPLAIN PLANについて説明します。

機能のカテゴリに分類したヒント

表2-23に、機能のカテゴリに分類したヒントと、各ヒントの構文とセマンティクスの参照先を示します。表の後には、ヒントをアルファベット順に説明します。

表2-23 機能のカテゴリに分類したヒント

ヒント	構文とセマンティクスの参照先
最適化目標と方法	ALL_ROWS ヒント FIRST_ROWS ヒント
アクセス・パスのヒント	CLUSTER ヒント
--	CLUSTERING ヒント NO_CLUSTERING ヒント
--	FULL ヒント
--	HASH ヒント
--	INDEX ヒント NO_INDEX ヒント
--	INDEX_ASC ヒント INDEX_DESC ヒント
--	INDEX_COMBINE ヒント
--	INDEX_JOIN ヒント
--	INDEX_FFS ヒント
--	INDEX_SS ヒント
--	INDEX_SS_ASC ヒント
--	INDEX_SS_DESC ヒント
--	NATIVE_FULL_OUTER_JOIN ヒント NO_NATIVE_FULL_OUTER_JOIN ヒント
--	NO_INDEX_FFS ヒント

ヒント	構文とセマンティクスの参照先
--	NO_INDEX_SS ヒント
--	NO_ZONEMAP ヒント
インメモリー列ストアヒント	INMEMORY ヒント NO_INMEMORY ヒント
--	INMEMORY_PRUNING ヒント NO_INMEMORY_PRUNING ヒント
結合順序のヒント	ORDERED ヒント
--	LEADING ヒント
結合操作のヒント	USE_BAND ヒント NO_USE_BAND ヒント
--	USE_CUBE ヒント NO_USE_CUBE ヒント
--	USE_HASH ヒント NO_USE_HASH ヒント
--	USE_MERGE ヒント NO_USE_MERGE ヒント
--	USE_NL ヒント USE_NL_WITH_INDEX ヒント NO_USE_NL ヒント
パラレル実行のヒント	ENABLE_PARALLEL_DML ヒント

ヒント	構文とセマンティクスの参照先
	DISABLE_PARALLEL_DML ヒント
--	PARALLEL ヒント NO_PARALLEL ヒント
--	PARALLEL_INDEX ヒント NO_PARALLEL_INDEX ヒント
--	PQ_CONCURRENT_UNION ヒント NO_PQ_CONCURRENT_UNION ヒント
--	PQ_DISTRIBUTE ヒント
--	PQ_FILTER ヒント
--	PQ_SKEW ヒント NO_PQ_SKEW ヒント
オンライン・アプリケーション・アップグレードのヒント	CHANGE_DUPKEY_ERROR_INDEX ヒント
--	IGNORE_ROW_ON_DUPKEY_INDEX ヒント
--	RETRY_ON_ROW_CHANGE ヒント
問合せ変換のヒント	FACT ヒント NO_FACT ヒント
--	MERGE ヒント NO_MERGE ヒント
--	NO_EXPAND ヒント USE_CONCAT ヒント

ヒント	構文とセマンティクスの参照先
--	REWRITE ヒント NO_REWRITE ヒント
--	UNNEST ヒント NO_UNNEST ヒント
--	STAR_TRANSFORMATION ヒント NO_STAR_TRANSFORMATION ヒント
--	NO_QUERY_TRANSFORMATION ヒント
XML のヒント	NO_XMLINDEX_REWRITE ヒント
--	NO_XML_QUERY_REWRITE ヒント
その他のヒント	APPEND ヒント APPEND_VALUES ヒント NOAPPEND ヒント
--	CACHE ヒント NOCACHE ヒント
--	CONTAINERS ヒント
--	CURSOR_SHARING_EXACT ヒント
--	DRIVING_SITE ヒント
--	DYNAMIC_SAMPLING ヒント
	FRESH_MV ヒント
--	GATHER_OPTIMIZER_STATISTICS ヒント

ヒント	構文とセマンティクスの参照先
	NO_GATHER_OPTIMIZER_STATISTICS ヒント
	GROUPING ヒント
--	MODEL_MIN_ANALYSIS ヒント
--	MONITOR ヒント
--	NO_MONITOR ヒント
--	OPT_PARAM ヒント
--	PUSH_PRED ヒント NO_PUSH_PRED ヒント
--	PUSH_SUBQ ヒント NO_PUSH_SUBQ ヒント
--	PX_JOIN_FILTER ヒント NO_PX_JOIN_FILTER ヒント
--	QB_NAME ヒント

ヒントのリスト(アルファベット順)

この項では、すべてのヒントの構文とセマンティクスをアルファベット順に説明します。

ALL_ROWSヒント

ALL_ROWSヒントは、文ブロックが最高のスループットになるよう(リソースの消費が最小になるよう)、オプティマイザに最適化の指示をします。たとえば、オプティマイザは問合せの最適化アプローチを使用して、この文を最高のスループットに最適化します。

```
SELECT /*+ ALL_ROWS */ employee_id, last_name, salary, job_id
  FROM employees
  WHERE employee_id = 107;
```

ALL_ROWSまたはFIRST_ROWSヒントのいずれかをSQL文で指定する場合に、この文でアクセスする表の統計情報がデータ・ディクショナリにないと、オプティマイザは該当する表に割り当てられている記憶域など、デフォルトの統計値を使用して欠落している統計値を推定し、実行計画を選択します。このような推定値は、DBMS_STATSパッケージで収集した値ほど正確でない場合があるため、統計情報の収集にはDBMS_STATSパッケージを使用する必要があります。

アクセス・パスまたは結合操作のヒントをALL_ROWSまたはFIRST_ROWSヒントとともに指定する場合、オプティマイザでは、ヒントで指定されたアクセス・パスまたは結合操作が優先されます。

APPENDヒント

APPENDヒントは、INSERT文の副問合せ構文とともにダイレクト・パスINSERT文を使用するようオプティマイザに指示します。

- 従来型のINSERTはシリアル・モードでのデフォルトです。シリアル・モードでは、ダイレクト・パスはAPPENDヒントを指定する場合にのみ使用できます。
- ダイレクト・パス・インサートはパラレル・モードでのデフォルトです。パラレル・モードでは、従来型INSERTはNOAPPENDヒントを指定する場合にのみ使用できます。

INSERTをパラレルにするかどうかの決定は、APPENDヒントとは関係ありません。

ダイレクト・パス・インサートでは、データは現在表に割り当てられている既存の空き領域を使用せず、表の最後に追加されます。その結果、ダイレクト・パスINSERTは、従来型INSERTよりも高速に処理されます。

APPENDヒントは、INSERT文の副問合せ構文でのみサポートされています。VALUES句ではサポートされていません。APPENDヒントをVALUES句とともに指定した場合、ヒントは無視され、従来型INSERTが使用されます。VALUES句とともにダイレクト・パスINSERTを使用する方法については、「APPEND_VALUESヒント」を参照してください。

関連項目:

NOAPPENDヒントの詳細は、NOAPPENDヒントを参照してください。ダイレクト・パス・インサートの詳細は、『Oracle Database管理者ガイド』を参照してください。

APPEND_VALUESヒント

APPEND_VALUESヒントは、VALUES句とともにダイレクト・パスINSERTを使用するようオプティマイザに指示します。このヒントを指定しない場合、従来型INSERTが使用されます。

ダイレクト・パス・インサートでは、データは現在表に割り当てられている既存の空き領域を使用せず、表の最後に追加されます。その結果、ダイレクト・パスINSERTは、従来型INSERTよりも高速に処理されます。

APPEND_VALUESヒントを使用すると、パフォーマンスを大幅に向上できます。次に使用方法の例を示します。

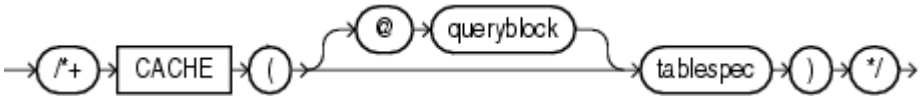
- Oracle Call Interface(OCI)を使用するプログラムで、大規模な配列バインドまたは行コールバックを伴う配列バインドを使用する場合
- PL/SQLで、VALUES句とともにINSERT文を使用するFORALLループを伴う多くの行をロードする場合

APPEND_VALUESヒントは、INSERT文のVALUES句でのみサポートされています。APPEND_VALUESヒントをINSERT文の副問合せ構文とともに指定した場合、ヒントは無視され、従来型INSERTが使用されます。副問合せとともにダイレクト・パスINSERTを使用する方法については、「APPENDヒント」を参照してください。

関連項目:

ダイレクト・パス・インサートの詳細は、『Oracle Database管理者ガイド』を参照してください。

CACHEヒント



(ヒントでの問合せブロックの指定、tablespec::=を参照)

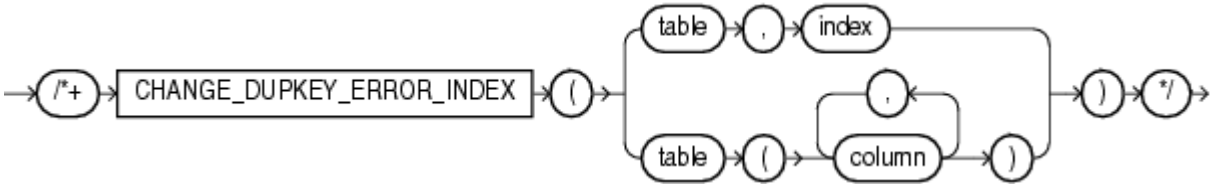
CACHEヒントは、全表スキャンの実行時に、この表に対して取り出されたブロックを、バッファ・キャッシュ内のLRUリストの最高使用頻度側に入れるようオプティマイザに指定します。このヒントは、小規模な参照表で有効です。

次の例では、CACHEヒントが表のデフォルト・キャッシュ仕様を上書きします。

```
SELECT /*+ FULL (hr_emp) CACHE(hr_emp) */ last_name
  FROM employees hr_emp;
```

CACHEおよびNOCACHEヒントは、V$SYSSTATデータ・ディクショナリ・ビューで示すように、システム統計情報table scans (long tables)およびtable scans (short tables)に影響を与えます。

CHANGE_DUPKEY_ERROR_INDEXヒント



ノート:

CHANGE_DUPKEY_ERROR_INDEX ヒント、IGNORE_ROW_ON_DUPKEY_INDEX ヒントおよびRETRY_ON_ROW_CHANGE ヒントは、セマンティクスに影響を与えるという点で他のヒントとは異なります。これら 3 つのヒントには、ヒントで説明されている一般的な原則は当てはまりません。

CHANGE_DUPKEY_ERROR_INDEXヒントは、指定した列セットまたは指定した索引に対する一意キー違反を明確に識別するメカニズムを提供します。指定した索引に対して一意キー違反が発生すると、ORA-001のかわりにORA-38911エラーが報告されます。

このヒントは、INSERT操作およびUPDATE操作に適用されます。索引を指定する場合は、その索引が存在するとともに一意であることが必要です。索引のかわりに列リストを指定する場合は、列の数と順序が列リストのものと一致する一意索引が存在する必要があります。

特定の規則に違反した場合、このヒントの使用によってエラー・メッセージが戻されます。詳細は、「IGNORE_ROW_ON_DUPKEY_INDEXヒント」を参照してください。

ノート:

このヒントを使用すると、APPEND モードおよびパラレル DML の両方が無効になります。

CLUSTERヒント

(ヒントでの問合せブロックの指定、tablespec::=を参照)

CLUSTERヒントは、クラスタ・スキャンを使用して、指定した表にアクセスするようオプティマイザに指示します。このヒントは、索引クラスタ内の表にのみ適用されます。

CLUSTERINGヒント

このヒントは、属性クラスタリングに対して有効な表のINSERTおよびMERGE操作にのみ有効です。CLUSTERINGヒントは、ダイレクト・パス・インサート(シリアルまたはパラレル)の属性クラスタリングを有効化します。これにより、部分的にクラスタリングされるデータになります。つまり、挿入またはマージ操作ごとにデータがクラスタリングされます。このヒントは、表を作成または変更したDDLのNO ON LOAD設定をオーバーライドします。このヒントは、属性クラスタリングに有効化されていない表には影響しません。

関連項目:

- NO ON LOAD設定の詳細は、CREATE TABLEのclustering_when句を参照してください。
- NO_CLUSTERINGヒント

CONTAINERSヒント

CONTAINERSヒントは、マルチテナント・コンテナ・データベース(CDB)で有益です。このヒントは、CONTAINERS( )句を含むSELECT文で指定できます。このような文では、CDBまたはアプリケーション・コンテナのすべてのコンテナにおいて、指定した表またはビューでデータを問い合せることができます。

- CDBのデータを問い合せるには、CDBルートに接続された共通ユーザーであり、表またはビューがルートおよびすべてのPDBに存在している必要があります。問合せは、CDBルートおよびすべてのオープン状態になっているPDBの表またはビューのすべての行を戻します。
- アプリケーション・コンテナのデータを問い合せるには、アプリケーション・ルートに接続された共通ユーザーであり、表また

はビューがアプリケーション・ルートおよびアプリケーション・コンテナ内のすべてのPDBに存在している必要があります。この問合せは、アプリケーション・ルートおよびアプリケーション・コンテナ内のオープン状態になっているすべてのPDBに存在する表またはビューのすべての行を戻します。

CONTAINERS( )句を含む文では、再帰的SQL文が生成され、問合せ対象の各PDBで実行されます。CONTAINERSヒントを使用して、デフォルトのPDBヒントを各再帰的SQL文に渡すことができます。hintには、SELECT文に適した任意のSQLヒントを指定できます。

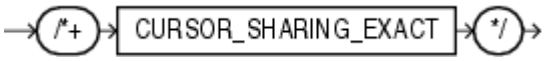
次の例では、CONTAINERS( )句の評価の一環として実行する各再帰的SQL文にNO_PARALLELヒントを渡します。

```
SELECT /*+ CONTAINERS(DEFAULT_PDB_HINT='NO_PARALLEL') */
  (CASE WHEN COUNT(*) < 10000
        THEN 'Less than 10,000'
        ELSE '10,000 or more' END) "Number of Tables"
  FROM CONTAINERS(DBA_TABLES);
```

関連項目:

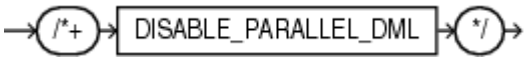
CONTAINERS( )句の詳細は、containers_clauseを参照してください

CURSOR_SHARING_EXACTヒント



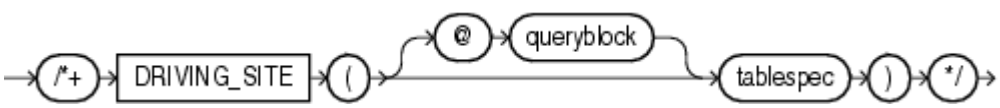
Oracleでは、置換しても安全な場合には、SQL文内のリテラルをバインド変数に置き換えることができます。この置換処理は、CURSOR_SHARING初期化パラメータを使用して制御します。CURSOR_SHARING_EXACTヒントは、この動作を行わないようオプティマイザに指示します。このヒントを指定すると、Oracleは、リテラルのバインド変数への置換を試行せずにSQL文を実行します。

DISABLE_PARALLEL_DMLヒント



DISABLE_PARALLEL_DMLヒントは、DELETE、INSERT、MERGEおよびUPDATE文のパラレルDMLを無効化します。ALTER SESSION ENABLE PARALLEL DML文を使用したセッションでパラレルDMLが有効化されている場合、このヒントを使用して、個別の文のパラレルDMLを無効化できます。

DRIVING_SITEヒント



(ヒントでの問合せブロックの指定、tablespec::=を参照)

DRIVING_SITEヒントは、データベースによって選択されたサイトとは異なるサイトで問合せを実行するようオプティマイザに指示します。このヒントは、分散化された問合せの最適化を使用している場合に有効です。

たとえば:

```
SELECT /*+ DRIVING_SITE(departments) */ *
```

```
  FROM employees, departments@rsite
  WHERE employees.department_id = departments.department_id;
```

この問合せがヒントなしで実行されている場合、departmentsからの行がローカル・サイトに送信され、そこで結合が実行されます。このヒントを使用している場合、employeesからの行がリモート・サイトに送信され、そこで問合せが実行されて結果セットがローカル・サイトに戻されます。

DYNAMIC_SAMPLINGヒント

(ヒントでの問合せブロックの指定、tablespec::=を参照)

DYNAMIC_SAMPLINGヒントは、動的なサンプリングを制御する方法をオプティマイザに指示し、より正確な述語の選択性と表および索引に対する統計情報を調べることでサーバーのパフォーマンスを改善します。

DYNAMIC_SAMPLINGの値は、0から10の範囲で設定できます。このレベルが高いほど、コンパイラは多くのリソースを動的なサンプリングに費やし、その適用範囲も広がります。tablespecを指定しない場合、サンプリングのデフォルトは、カーソルのレベルになります。

integerの値は0から10となり、サンプリングの度合いを示します。

カーディナリティ統計情報が表にすでに存在する場合、オプティマイザはその情報を使用します。存在しない場合、オプティマイザは動的なサンプリングによってカーディナリティ統計情報を推定します。

tablespecを指定しており、カーディナリティ統計情報がすでに存在している場合は、次のようになります。

- 単一表述語(1つの表のみを評価するWHERE句)がない場合、オプティマイザは既存の統計情報を信頼し、このヒントを無視します。たとえば、employeesが分析される場合、次の問合せは動的なサンプリングにはなりません。

  ```
  SELECT /*+ DYNAMIC_SAMPLING(e 1) */ count(*)
    FROM employees e;
  ```

- 単一表述語がある場合、オプティマイザは既存のカーディナリティ統計情報を使用し、この既存の統計情報を使用して述語の選択性を推定します。

動的なサンプリングを特定の表に適用するには、次の形式のヒントを使用します。

```
SELECT /*+ DYNAMIC_SAMPLING(employees 1) */ *
  FROM employees
  WHERE ...
```

関連項目:

動的なサンプリングおよび設定可能なサンプリング・レベルの詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

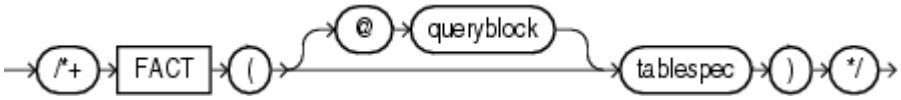
ENABLE_PARALLEL_DMLヒント

ENABLE_PARALLEL_DMLヒントは、DELETE、INSERT、MERGEおよびUPDATE文のパラレルDMLを有効化します。ALTER SESSION ENABLE PARALLEL DML文を使用したセッションでパラレルDMLが有効化されている場合、このヒントを使用して、個別の文のパラレルDMLを有効化できます。

関連項目:

パラレルDMLの有効化の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

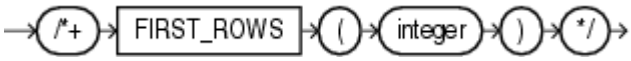
FACTヒント



(ヒントでの問合せブロックの指定、tablespec::=を参照)

FACTヒントは、スター型変換のコンテキストで使用されます。これは、tablespecで指定された表をファクト表とみなすようオプティマイザに指示するためのものです。

FIRST_ROWSヒント



FIRST_ROWSヒントは、最初のn行を最も効率的に戻す計画を選択し、個々のSQL文を最適化して応答時間を速くするようOracleに指示します。integerには、戻される行数を指定します。

たとえば、オプティマイザは問合せの最適化アプローチを使用して、次の文を最善の応答時間に最適化します。

```
SELECT /*+ FIRST_ROWS(10) */ employee_id, last_name, salary, job_id
  FROM employees
  WHERE department_id = 20;
```

この例では、各部門に多数の従業員がいます。ユーザーは、部門20の最初の10人の従業員を迅速に表示する必要があります。

オプティマイザは、DELETEおよびUPDATEの文ブロック、およびソートまたはグループ化などのブロッキング操作を含むSELECT文ブロックではこのヒントを無視します。Oracle Databaseでは最初の行を戻す前に、この文でアクセスされるすべての行を取り出す必要があるため、このような文は最善の応答時間に最適化することができません。このヒントをこのような文で指定する場合は、データベースを最善のスループットに最適化します。

関連項目:

FIRST_ROWSヒントと統計情報の詳細は、ALL_ROWSヒントを参照してください。

FRESH_MVヒント

FRESH_MVヒントは、リアルタイムのマテリアライズド・ビューを問い合せるときに適用されます。マテリアライズド・ビューが古い場合

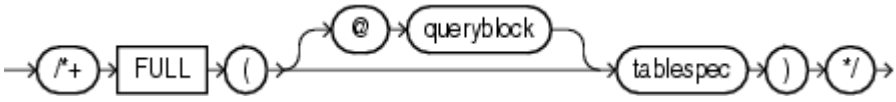
でも、このヒントはオプティマイザに問合せ時計算を使用してマテリアライズド・ビューから最新データを取得するよう指示します。

オプティマイザは、リアルタイムのマテリアライズド・ビューではないオブジェクトを問い合せるSELECT文ブロックおよびすべてのUPDATE、INSERT、MERGE、DELETE文ブロックではこのヒントを無視します。

関連項目:

リアルタイムのマテリアライズド・ビューの詳細は、CREATE MATERIALIZED VIEWの「{ ENABLE | DISABLE } ON QUERY COMPUTATION」句を参照してください。

FULLヒント



(ヒントでの問合せブロックの指定、tablespec::=を参照)

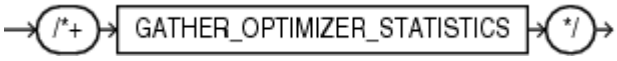
FULLヒントは、指定した表に対して全表スキャンを実行するようオプティマイザに指示します。たとえば:

```
SELECT /*+ FULL(e) */ employee_id, last_name
  FROM hr.employees e
  WHERE last_name LIKE :b1;
```

Oracle Databaseは、WHERE句内の条件によって使用可能になるlast_name列に索引がある場合でも、employees表に対して全表スキャンを実行し、この文を実行します。

employees表は、FROM句の中に別名eを持つため、ヒントは表名ではなく別名で表を参照する必要があります。スキーマ名がFROM句の中で指定されている場合でも、ヒントではスキーマ名を指定しないでください。

GATHER_OPTIMIZER_STATISTICSヒント



GATHER_OPTIMIZER_STATISTICSヒントは、次のタイプのバルク・ロード操作中に統計収集を有効にするようオプティマイザに指示します。

- CREATE TABLE ... AS SELECT
- ダイレクト・パス・インサートを使用した、空の表へのINSERT INTO ... SELECT

関連項目:

バルク・ロードに関する統計収集の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

GROUPINGヒント

GROUPINGヒントは、パーティション化されたモデルをスコアリングするときに、データ・マイニング・スコアリング・ファンクションに適用されます。このヒントによって、入力データ・セットが個別のデータ・スライスにパーティション化されるため、次のパーティションに進む

前に各パーティション全体がスコアリングされます。ただし、パーティションによるパラレル化も使用できます。データ・スライスは、モデルの構築時に使用されたパーティション化キー列によって決定されます。この方法は、パーティション化されたモデルに対するデータ・マイニング・ファンクションとともに使用できます。このヒントにより、多くのパーティションに関連付けられている大規模データをスコアリングするときは問合せパフォーマンスが向上する可能性がありますが、大規模システムでパーティションが少ない大規模データをスコアリングするときはパフォーマンスが低下する可能性があります。通常、このヒントを単一行問合せに使用する場合、パフォーマンスは向上しません。

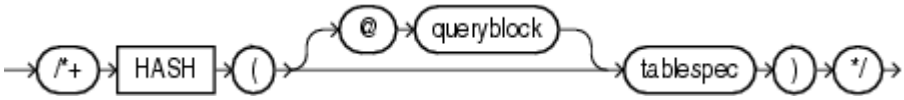
次の例では、PREDICTIONファンクションでGROUPINGヒントが使用されています。

```
SELECT PREDICTION(/*+ GROUPING */my_model USING *) pred FROM <input table>;
```

関連項目:

データ・マイニング・ファンクション

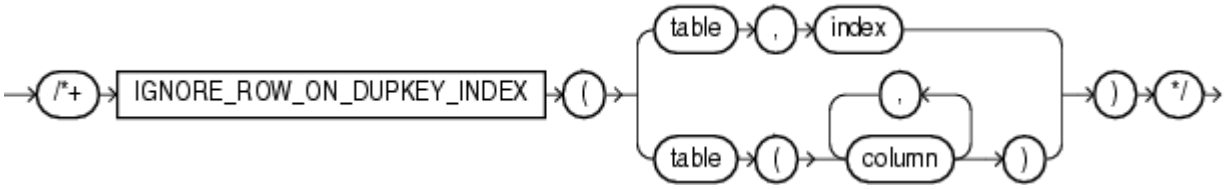
HASHヒント



(ヒントでの問合せブロックの指定、tablespec::=を参照)

HASHヒントは、ハッシュ・スキャンを使用して、指定した表にアクセスするようオプティマイザに指示します。このヒントは、ハッシュ・クラスタ内の表にのみ適用されます。

IGNORE_ROW_ON_DUPKEY_INDEXヒント



ノート:

CHANGE_DUPKEY_ERROR_INDEX ヒント、IGNORE_ROW_ON_DUPKEY_INDEX ヒントおよびRETRY_ON_ROW_CHANGE ヒントは、セマンティクスに影響を与えるという点で他のヒントとは異なります。これら 3 つのヒントには、ヒントで説明されている一般的な原則は当てはまりません。

IGNORE_ROW_ON_DUPKEY_INDEXヒントが適用されるのは、単一表INSERT操作のみです。UPDATE、DELETE、MERGEおよびマルチテーブル・インサート操作に対してはサポートされません。IGNORE_ROW_ON_DUPKEY_INDEXを指定すると、その文では、指定の列セットまたは指定の索引に対する一意キー違反が無視されます。一意キー違反が発生したときは、行レベルのロールバックが行われ、実行は次の入力行から再開します。データを挿入するときにこのヒントを指定した場合は、DMLエラー・ロギングが有効になっていても、一意キー違反はログに記録されず、文は終了しません。

特定の規則に違反した場合、このヒントによるセマンティクスへの影響によってエラー・メッセージが戻されます。

- 索引を指定する場合は、その索引が存在するとともに一意であることが必要です。そうでない場合は、ORA-38913が発生します。
- 索引は、1つのみ指定する必要があります。索引を指定しなかった場合、ORA-38912が発生します。複数の索引を指定した場合、ORA-38915が発生します。
- INSERT文にCHANGE_DUPKEY_ERROR_INDEXまたはIGNORE_ROW_ON_DUPKEY_INDEXのいずれかのヒントを指定できますが、両方を指定することはできません。両方を指定した場合、ORA-38915が発生します。

すべてのヒントと同様に、ヒント内の構文エラーは特に警告もなく無視されます。その結果、ヒントを使用しなかった場合と同じように、ORA-00001が発生します。

ノート:

このヒントを使用すると、APPEND モードおよびパラレル DML の両方が無効になります。

関連項目:

CHANGE_DUPKEY_ERROR_INDEXヒント

INDEXヒント

(ヒントでの問合せブロックの指定、tablespec::=、indexspec::=を参照)

INDEXヒントは、指定した表について索引スキャンを使用するようオプティマイザに指示します。ファンクション索引、ドメイン索引、Bツリー索引、ビットマップ索引およびビットマップ結合索引について、INDEXヒントを使用できます。

ヒントの動作は、indexspecの仕様によって異なります。

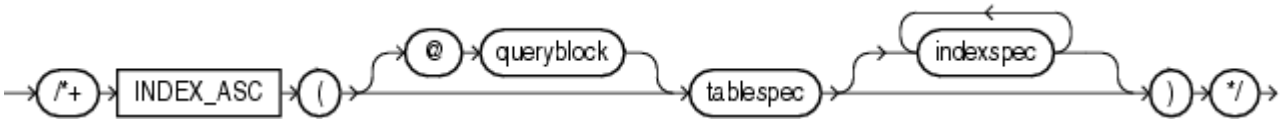
- INDEXヒントが使用可能な単一の索引を指定すると、データベースはこの索引でスキャンを実行します。オプティマイザは、全表スキャンおよび表上の別の索引のスキャンを考慮しません。
- 複数の索引の組合せに対してヒントを指定する場合は、INDEXではなく、より多目的なヒントであるINDEX_COMBINEを使用することをお薦めします。INDEXヒントで索引のリストが指定されている場合は、オプティマイザはこのリスト内の各索引のスキャンのコストを計算してから、コストが最低である索引スキャンを実行します。あるいは、このリストの複数の索引をスキャンしてその結果をマージするというアクセス・パスが選択されることもあります(そのようなアクセス・パスのコストが最低になる場合)。全表スキャンや、ヒントで指定されていない索引のスキャンが検討されることはありません。
- INDEXヒントが索引を指定しない場合、オプティマイザは、表にある使用可能な各索引でのスキャンのコストを検討し、最低のコストで索引スキャンを実行します。アクセス・パスのコストが最低となる場合、データベースは複数の索引をスキャンするように選択し、結果をマージすることもあります。オプティマイザは全表スキャンを検討しません。

たとえば:

```
SELECT /*+ INDEX (employees emp_department_ix)*/ employee_id, department_id
  FROM employees
```

```
  WHERE department_id > 50;
```

INDEX_ASCヒント

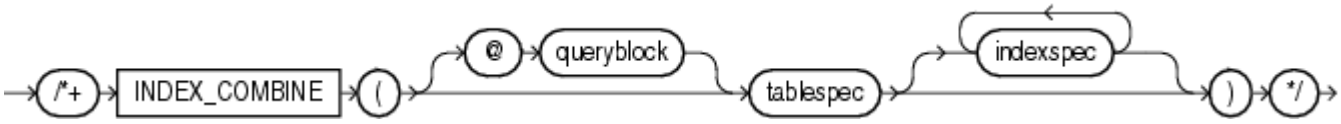


(ヒントでの問合せブロックの指定、tablespec::=、indexspec::=を参照)

INDEX_ASCヒントは、指定した表について索引スキャンを使用するようオプティマイザに指示します。文で索引レンジ・スキャンを使用する場合、Oracle Databaseは索引付きの値の昇順で索引エントリをスキャンします。各パラメータは、INDEXヒントと同じ目的で使用されます。

レンジ・スキャンのデフォルトの動作は、索引エントリを索引値の昇順で(降順索引の場合は降順で)スキャンするというものです。このヒントを指定しても、索引のデフォルトの順序が変わることはなく、したがってINDEXヒントを指定したのと同じことになります。ただし、INDEX_ASCヒントを使用すると、デフォルトの動作が変更された場合に昇順レンジ・スキャンを明示的に指定できます。

INDEX_COMBINEヒント

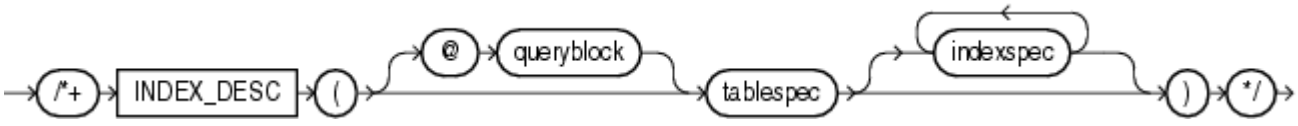


(ヒントでの問合せブロックの指定、tablespec::=、indexspec::=を参照)

INDEX_COMBINEヒントでは、ビットマップ、Bツリーまたはドメインのいずれかのタイプの索引を使用できます。INDEX_COMBINEヒントにindexspecを指定しない場合、オプティマイザは、可能な限り多くの索引を使用し、すべての索引にINDEXヒントを暗黙的に適用します。indexspecを指定すると、オプティマイザはコストを考慮せずに、ヒントが適用されている有効かつ使用可能な索引をすべて使用します。各パラメータは、INDEXヒントと同じ目的で使用されます。たとえば:

```
SELECT /*+ INDEX_COMBINE(e emp_manager_ix emp_department_ix) */ *
  FROM employees e
  WHERE manager_id = 108
     OR department_id = 110;
```

INDEX_DESCヒント



(ヒントでの問合せブロックの指定、tablespec::=、indexspec::=を参照)

INDEX_DESCヒントは、指定した表に降順索引スキャンを使用するようオプティマイザに指示します。文で索引レンジ・スキャンを使用しており、索引が昇順の場合、Oracleは索引付きの値の降順で索引エントリをスキャンします。パーティション索引では、結果は各パーティション内で降順になります。降順索引の場合、このヒントは降順を効果的に取り消すため、索引エントリは昇順でスキャンされます。各パラメータは、INDEXヒントと同じ目的で使用されます。たとえば:

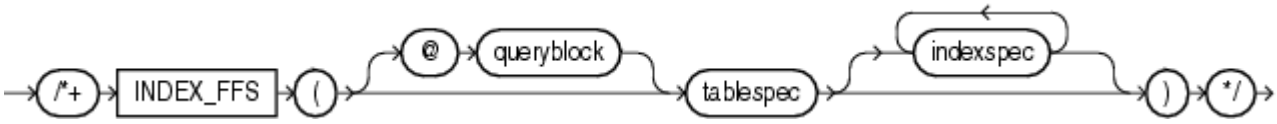
```
SELECT /*+ INDEX_DESC(e emp_name_ix) */ *
```

```
  FROM employees e;
```

関連項目:

全体スキャンの詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

INDEX_FFSヒント



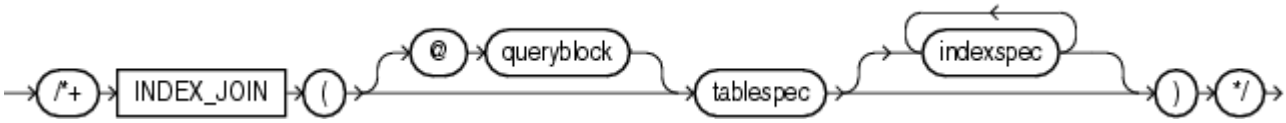
(ヒントでの問合せブロックの指定、tablespec::=、indexspec::=を参照)

INDEX_FFSヒントは、全表スキャンではなく高速全索引スキャンを実行するようオプティマイザに指示します。

各パラメータは、INDEXヒントと同じ目的で使用されます。たとえば:

```
SELECT /*+ INDEX_FFS(e emp_name_ix) */ first_name
  FROM employees e;
```

INDEX_JOINヒント



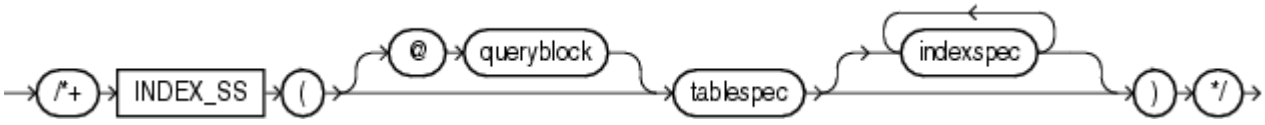
(ヒントでの問合せブロックの指定、tablespec::=、indexspec::=を参照)

INDEX_JOINヒントは、アクセス・パスとして索引結合を使用するようオプティマイザに指示します。ヒントが正しく機能するためには、問合せの解決に必要なすべての列を含む索引が最小限の数だけ存在している必要があります。

各パラメータは、INDEXヒントと同じ目的で使用されます。たとえば、次に示す問合せでは索引結合を使用してmanager_id列とdepartment_id列にアクセスしています(どちらの列もemployees表内で索引が付けられています)。

```
SELECT /*+ INDEX_JOIN(e emp_manager_ix emp_department_ix) */ department_id
  FROM employees e
  WHERE manager_id < 110
    AND department_id < 50;
```

INDEX_SSヒント



(ヒントでの問合せブロックの指定、tablespec::=、indexspec::=を参照)

INDEX_SSヒントは、指定した表について索引スキップ・スキャンを実行するようオプティマイザに指示します。文で索引レンジ・スキャンを使用する場合、Oracleは索引付きの値の昇順で索引エントリをスキャンします。パーティション索引では、結果は各パーティション内で昇順になります。

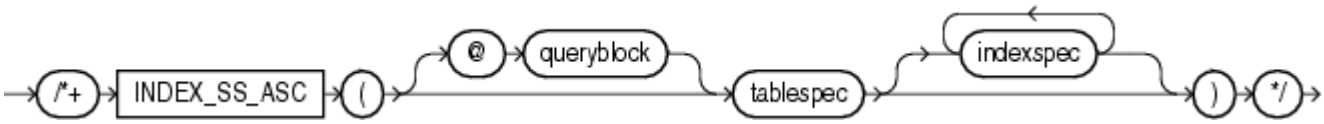
各パラメータは、INDEXヒントと同じ目的で使用されます。たとえば:

```
SELECT /*+ INDEX_SS(e emp_name_ix) */ last_name
  FROM employees e
  WHERE first_name = 'Steven';
```

関連項目:

索引スキップ・スキャンの詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

INDEX_SS_ASCヒント



(ヒントでの問合せブロックの指定、tablespec::=、indexspec::=を参照)

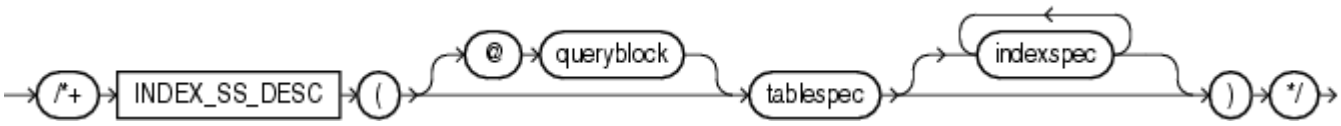
INDEX_SS_ASCヒントは、指定した表について索引スキップ・スキャンを実行するようオプティマイザに指示します。文で索引レンジ・スキャンを使用する場合、Oracle Databaseは索引付きの値の昇順で索引エントリをスキャンします。パーティション索引では、結果は各パーティション内で昇順になります。各パラメータは、INDEXヒントと同じ目的で使用されます。

レンジ・スキャンのデフォルトの動作は、索引エントリを索引値の昇順で(降順索引の場合は降順で)スキャンするというものです。このヒントを指定しても、索引のデフォルトの順序が変わることはなく、したがってINDEX_SSヒントを指定したのと同じことになります。ただし、INDEX_SS_ASCヒントを使用すると、デフォルトの動作が変更された場合に昇順レンジ・スキャンを明示的に指定できます。

関連項目:

索引スキップ・スキャンの詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

INDEX_SS_DESCヒント



(ヒントでの問合せブロックの指定、tablespec::=、indexspec::=を参照)

INDEX_SS_DESCヒントは、指定した表について索引スキップ・スキャンを実行するようオプティマイザに指示します。文で索引レンジ・スキャンを使用しており、索引が昇順の場合、Oracleは索引付きの値の降順で索引エントリをスキャンします。パーティション索引では、結果は各パーティション内で降順になります。降順索引の場合、このヒントは降順を効果的に取り消すため、索引エントリは昇順でスキャンされます。

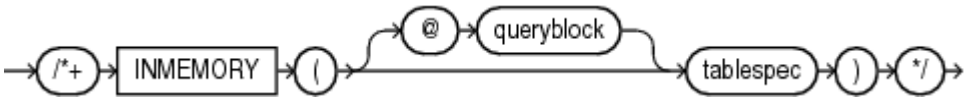
各パラメータは、INDEXヒントと同じ目的で使用されます。たとえば:

```
SELECT /*+ INDEX_SS_DESC(e emp_name_ix) */ last_name
  FROM employees e
  WHERE first_name = 'Steven';
```

関連項目:

索引スキップ・スキャンの詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

INMEMORYヒント

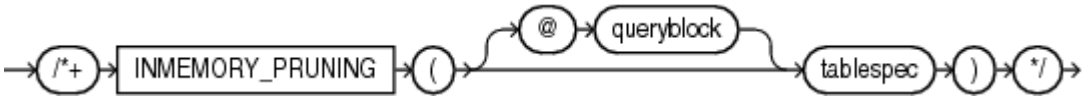


(ヒントでの問合せブロックの指定、tablespec::=を参照)

INMEMORYヒントは、インメモリー問合せを有効化します。

このヒントは、オプティマイザに全表スキャンを実行するようには指示しません。全表スキャンが必要な場合は、FULLヒントも指定してください。

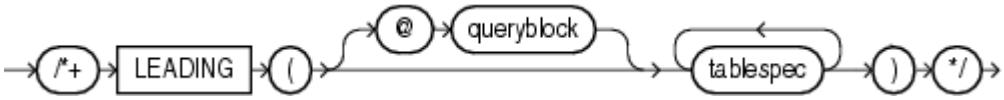
INMEMORY_PRUNINGヒント



(ヒントでの問合せブロックの指定、tablespec::=を参照)

INMEMORY_PRUNINGヒントは、インメモリー問合せのプルーニングを有効化します。

LEADINGヒント



(ヒントでの問合せブロックの指定、tablespec::=を参照)

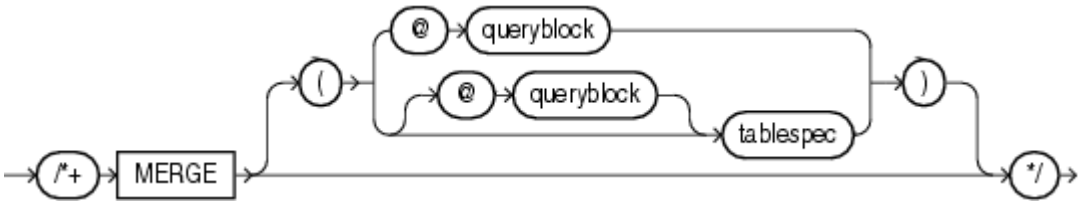
LEADINGヒントは、複数の表またはビューを指定できる複数表ヒントです。LEADINGは、実行計画において、指定した一連の表を接頭辞として使用するようオプティマイザに指示します。指定した最初の表が結合の開始に使用されます。

このヒントは、ORDEREDヒントより多目的なヒントです。たとえば:

```
SELECT /*+ LEADING(e j) */ *
    FROM employees e, departments d, job_history j
    WHERE e.department_id = d.department_id
      AND e.hire_date = j.start_date;
```

図形結合の依存性により、指定の表を指定の順序の最初に結合できない場合、LEADINGヒントは無視されます。2つ以上の競合するLEADINGヒントを指定すると、指定したすべてのヒントが無視されます。ORDEREDヒントを指定すると、このヒントがすべてのLEADINGヒントに優先します。

MERGEヒント



(ヒントでの問合せブロックの指定、tablespec::=を参照)

MERGEヒントを使用すると、問合せ内のビューをマージできます。

ビューの問合せブロックにGROUP BY句、またはSELECTリスト内のDISTINCT演算子が含まれている場合、複雑なビューのマージが可能であれば、オプティマイザはビューをアクセス文のみにマージできます。複雑なマージは、副問合せに相関関係がない場合、IN副問合せをアクセス文にマージする際に使用することもできます。

たとえば:

```
SELECT /*+ MERGE(v) */ e1.last_name, e1.salary, v.avg_salary
   FROM employees e1,
        (SELECT department_id, avg(salary) avg_salary
           FROM employees e2
           GROUP BY department_id) v
   WHERE e1.department_id = v.department_id
     AND e1.salary > v.avg_salary
   ORDER BY e1.last_name;
```

引数なしでMERGEヒントを使用する場合、ビューの問合せブロック内に配置する必要があります。ビュー名を引数としてMERGEヒントを使用する場合、周囲の問合せに挿入する必要があります。

MODEL_MIN_ANALYSISヒント

MODEL_MIN_ANALYSISヒントは、スプレッドシート・ルールのコンパイル時間の最適化(主に、詳細な依存グラフ分析)を省略するようオプティマイザに指示します。スプレッドシートのアクセス構造に選択的に移入するためのフィルタの作成や制限されたルールのプルーニングなど、他のスプレッドシートの最適化は、引き続きオプティマイザによって使用されます。

スプレッドシート・ルールの数が数百を超えると、スプレッドシート分析に長い時間がかかることがあるため、このヒントによってコンパイル時間を減らします。

MONITORヒント

MONITORヒントは、文の実行時間が長くない場合でも、問合せのリアルタイムのSQL監視を強制します。このヒントは、パラメータCONTROL_MANAGEMENT_PACK_ACCESSがDIAGNOSTIC+TUNINGに設定されている場合にのみ有効です。

関連項目:

リアルタイムSQL監視の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

NATIVE_FULL_OUTER_JOINヒント

NATIVE_FULL_OUTER_JOINヒントは、ネイティブ完全外部結合を使用するようオプティマイザに指示します。ネイティブ完全外部結合は、ハッシュ結合に基づくネイティブ実行メソッドです。

関連項目:

- NO_NATIVE_FULL_OUTER_JOINヒント
- ネイティブ完全外部結合の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

NOAPPENDヒント

NOAPPENDヒントは、INSERT文が有効な間、パラレル・モードを無効にして従来型のINSERTを使用するようにオプティマイザに指示します。従来型のINSERTはシリアル・モードでのデフォルトです。また、ダイレクト・パスINSERTはパラレル・モードでのデフォルトです。

NOCACHEヒント

(ヒントでの問合せブロックの指定、tablespec::=を参照)

NOCACHEヒントは、全表スキャンの実行時に、この表に対して取り出されたブロックを、バッファ・キャッシュ内のLRUリストの最低使用頻度側に入れるようオプティマイザに指定します。これは、バッファ・キャッシュ内のブロックの通常動作です。たとえば:

```
SELECT /*+ FULL(hr_emp) NOCACHE(hr_emp) */ last_name
  FROM employees hr_emp;
```

CACHEおよびNOCACHEヒントは、V$SYSSTATビューで示すように、システム統計情報table scans(long tables)およびtable scans(short tables)に影響を与えます。

NO_CLUSTERINGヒント

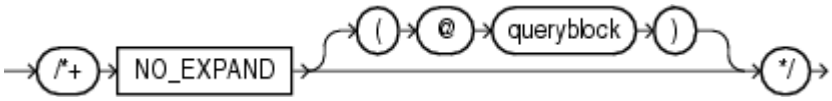
このヒントは、属性クラスタリングに対して有効な表のINSERTおよびMERGE操作にのみ有効です。NO_CLUSTERINGヒントは、ダイレクト・パス・インサート(シリアルまたはパラレル)の属性クラスタリングを無効化します。このヒントは、表を作成または変更したDDLのYES ON LOAD設定をオーバーライドします。このヒントは、属性クラスタリングに有効化されていない表には影響しません。

関連項目:

- YES ON LOAD設定の詳細は、CREATE TABLEのclustering_when句を参照してください。

- CLUSTERINGヒント

NO_EXPANDヒント



(ヒントでの問合せブロックの指定を参照)

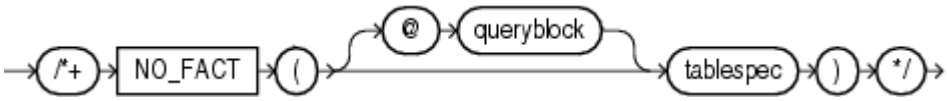
NO_EXPANDヒントは、OR条件を持つ問合せ用のOR拡張、またはWHERE句内にあるINリストを検討しないようオプティマイザに指示します。通常、オプティマイザは、OR拡張を使用しない場合よりもコストが低減できると判断すると、OR拡張の使用を検討します。たとえば:

```
SELECT /*+ NO_EXPAND */ *
  FROM employees e, departments d
  WHERE e.manager_id = 108
     OR d.department_id = 110;
```

関連項目:

USE_CONCATヒントを参照してください(このヒントの反対の機能です)。

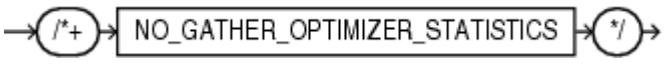
NO_FACTヒント



(ヒントでの問合せブロックの指定、tablespec::=を参照)

NO_FACTヒントは、スター型変換のコンテキストで使用されます。これは、問合せ対象の表をファクト表とみなさないようオプティマイザに指示するためのものです。

NO_GATHER_OPTIMIZER_STATISTICSヒント



NO_GATHER_OPTIMIZER_STATISTICSヒントは、次のタイプのバルク・ロード操作中に統計収集を無効にするようオプティマイザに指示します。

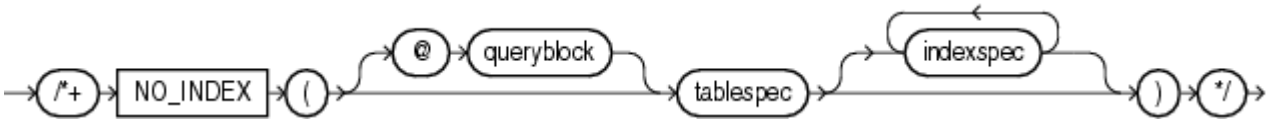
- CREATE TABLE AS SELECT
- ダイレクト・パス・インサートを使用した、空の表へのINSERT INTO ... SELECT

NO_GATHER_OPTIMIZER_STATISTICSヒントは、従来型ロードに適用されます。このヒントが従来型の挿入文で指定されている場合、Oracleはそのヒントに従い、リアルタイム統計を収集しません。

関連項目:

従来型ロードに関するオンライン統計収集の詳細は、Oracle Database SQLチューニング・ガイドを参照してください。

NO_INDEXヒント



(ヒントでの問合せブロックの指定、tablespec::=、indexspec::=を参照)

NO_INDEXヒントは、指定した表について1つ以上の索引を使用しないようオプティマイザに指示します。たとえば:

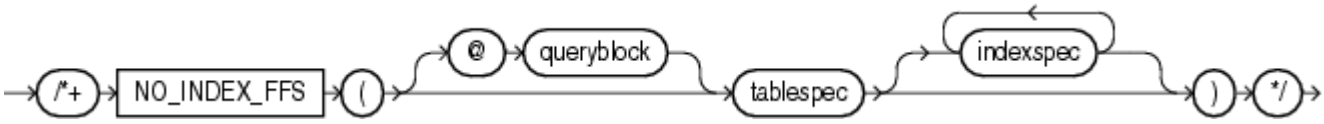
```
SELECT /*+ NO_INDEX(employees emp_empid) */ employee_id
  FROM employees
  WHERE employee_id > 200;
```

各パラメータはINDEXヒントと同じ目的で使用されますが、次の変更が加えられています。

- このヒントが使用可能な単一の索引を指定すると、オプティマイザはこの索引でのスキャンを検討しません。その他の指定されていない索引については、スキャンを検討します。
- このヒントが使用可能な索引のリストを指定すると、オプティマイザは指定された索引でのスキャンを検討しません。リスト内に指定されていないその他の索引については、スキャンを検討します。
- このヒントが索引を指定しない場合、オプティマイザは表のいずれの索引でのスキャンを考慮しません。この動作は、表について使用可能なすべての索引のリストを指定するNO_INDEXヒントと同様になります。

NO_INDEXヒントは、ファンクション索引、Bツリー索引、ビットマップ索引、クラスタ索引およびドメイン索引に適用されます。NO_INDEXヒントと索引ヒント(INDEX、INDEX_ASC、INDEX_DESC、INDEX_COMBINEまたはINDEX_FFS)の両方が同じ索引を指定する場合、データベースは、NO_INDEXヒントと、指定した索引の索引ヒントの両方を無視し、これらの索引を文の実行中に使用することを検討します。

NO_INDEX_FFSヒント

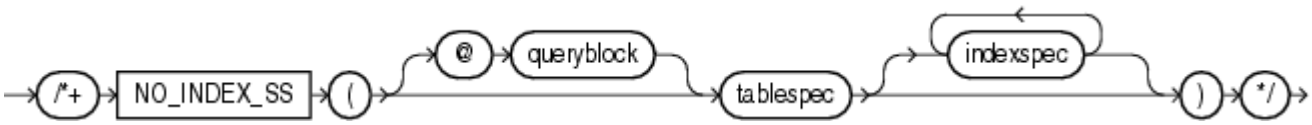


(ヒントでの問合せブロックの指定、tablespec::=、indexspec::=を参照)

NO_INDEX_FFSヒントは、指定された表の指定された索引の高速全索引スキャンを除外するようオプティマイザに指示します。各パラメータは、NO_INDEXヒントと同じ目的で使用されます。たとえば:

```
SELECT /*+ NO_INDEX_FFS(items item_order_ix) */ order_id
  FROM order_items items;
```

NO_INDEX_SSヒント



(ヒントでの問合せブロックの指定、tablespec::=、indexspec::=を参照)

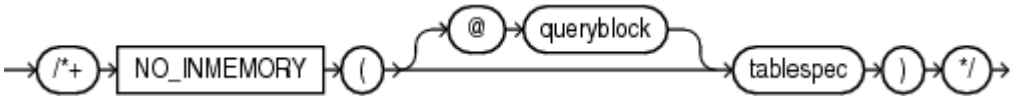
NO_INDEX_SSヒントは、指定された表の指定された索引のスキップ・スキャンを除外するようオプティマイザに指示します。各パ

ラメータは、NO_INDEXヒントと同じ目的で使用されます。

関連項目:

索引スキップ・スキャンの詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

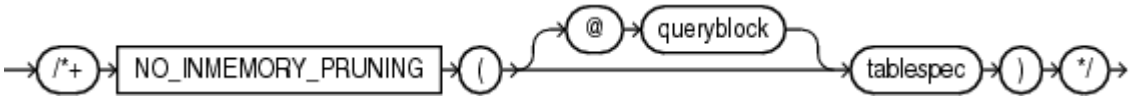
NO_INMEMORYヒント



(ヒントでの問合せブロックの指定、tablespec::=を参照)

NO_INMEMORYヒントは、インメモリー問合せを無効化します。

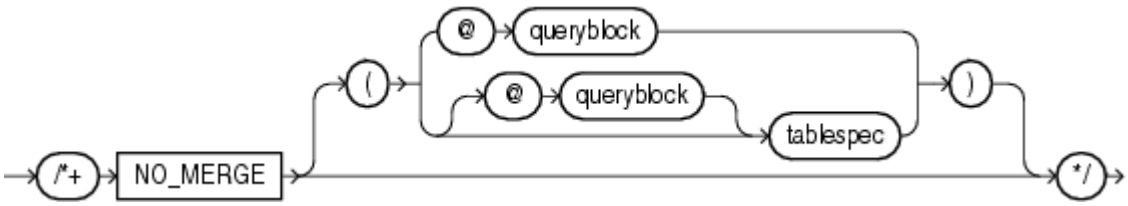
NO_INMEMORY_PRUNINGヒント



(ヒントでの問合せブロックの指定、tablespec::=を参照)

NO_INMEMORY_PRUNINGヒントは、インメモリー問合せのプルーニングを無効化します。

NO_MERGEヒント



(ヒントでの問合せブロックの指定、tablespec::=を参照)

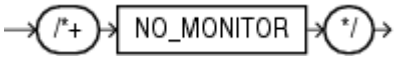
NO_MERGEヒントは、外部問合せとインライン・ビュー問合せを結合して単一の問合せにしないようオプティマイザに指示します。

このヒントを使用すると、ビューへのアクセス方法に強い影響を与えることができます。たとえば、次の文は、ビューseattle_deptがマージされないようにします。

```
SELECT /*+ NO_MERGE(seattle_dept) */ e1.last_name, seattle_dept.department_name
  FROM employees e1,
       (SELECT location_id, department_id, department_name
          FROM departments
          WHERE location_id = 1700) seattle_dept
  WHERE e1.department_id = seattle_dept.department_id;
```

ビューの問合せブロック内でNO_MERGEヒントを使用する場合は、引数なしで指定します。また、周囲の問合せでNO_MERGEを指定する場合には、ビュー名を引数として指定します。

NO_MONITORヒント



NO_MONITORヒントは、問合せの実行時間が長い場合でも、問合せのSQLのリアルタイム監視を無効にします。

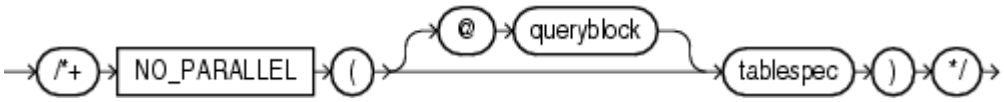
NO_NATIVE_FULL_OUTER_JOINヒント

NO_NATIVE_FULL_OUTER_JOINヒントでは、オプティマイザに対して、指定した各表を結合する際にネイティブの実行方法を除外するように指示します。かわりに、完全外部結合は、左側外部結合とアンチ結合の論理和として実行されます。

関連項目:

NATIVE_FULL_OUTER_JOINヒント

NO_PARALLELヒント



(ヒントでの問合せブロックの指定、tablespec::=を参照)

NO_PARALLELヒントは、文をシリアルで実行するようにオプティマイザに指示します。このヒントは、PARALLEL_DEGREE_POLICY初期化パラメータの値をオーバーライドします。また、表を作成するか変更したDDL内のPARALLELパラメータを上書きします。たとえば、次のSELECT文はシリアルで実行されます。

```
ALTER TABLE employees PARALLEL 8;
SELECT /*+ NO_PARALLEL(hr_emp) */ last_name
  FROM employees hr_emp;
```

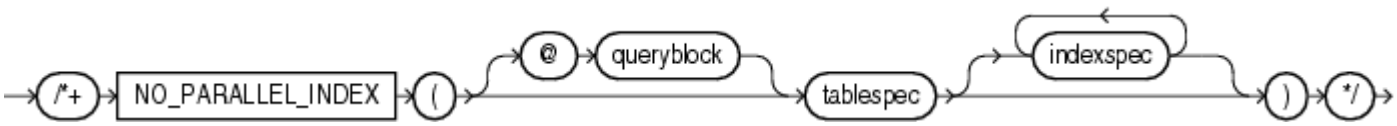
関連項目:

- パラレル・ヒントの詳細は、パラレル・ヒントのノートを参照してください。
- PARALLEL_DEGREE_POLICY初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

NOPARALLELヒント

NOPARALLELヒントは現在非推奨になっています。NO_PARALLELヒントをかわりに使用します。

NO_PARALLEL_INDEXヒント



(ヒントでの問合せブロックの指定、tablespec::=、indexspec::=を参照)

NO_PARALLEL_INDEXヒントは、索引を作成するか変更したDDL内のPARALLELパラメータを上書きし、パラレル索引スキャン操作を防止します。

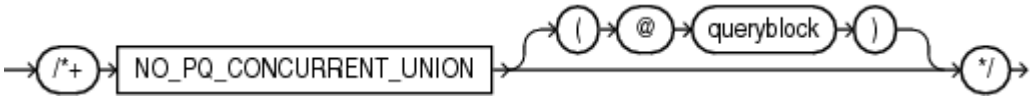
関連項目:

パラレル・ヒントの詳細は、パラレル・ヒントのノートを参照してください。

NOPARALLEL_INDEXヒント

NOPARALLEL_INDEXヒントは現在非推奨になっています。NO_PARALLEL_INDEXヒントをかわりに使用します。

NO_PQ_CONCURRENT_UNIONヒント



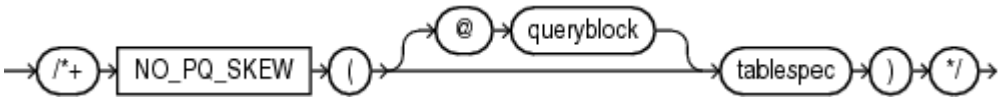
(ヒントでの問合せブロックの指定を参照)

NO_PQ_CONCURRENT_UNIONヒントは、UNION操作とUNION ALL操作の同時処理を無効にするようオプティマイザに指示します。

関連項目:

- PQ_CONCURRENT_UNIONヒント
- このヒントの使用方法の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

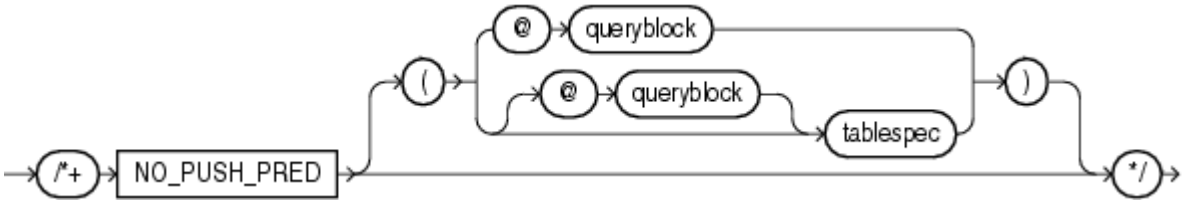
NO_PQ_SKEWヒント



(ヒントでの問合せブロックの指定、tablespec::=を参照)

NO_PQ_SKEWヒントは、パラレル結合の結合キーの値の分散が偏っていない(つまり、同じ結合キー値を持つ行の割合が大きくない)ことをオプティマイザに知らせます。tablespecで指定される表は、ハッシュ結合のプローブ表です。

NO_PUSH_PREDヒント

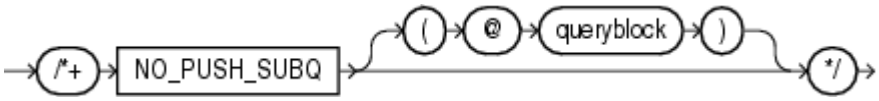


(ヒントでの問合せブロックの指定、tablespec::=を参照)

NO_PUSH_PREDヒントは、結合述語をビューにプッシュしないようオプティマイザに指示します。たとえば:

```
SELECT /*+ NO_MERGE(v) NO_PUSH_PRED(v) */ *
  FROM employees e,
       (SELECT manager_id
          FROM employees) v
  WHERE e.manager_id = v.manager_id(+)
    AND e.employee_id = 100;
```

NO_PUSH_SUBQヒント



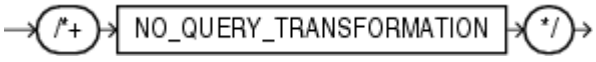
(ヒントでの問合せブロックの指定を参照)

NO_PUSH_SUBQヒントは、実行計画における最後のステップとして、マージされていない副問合せを評価するようオプティマイザに指示します。これにより、副問合せのコストが比較的高い場合や、副問合せによって行数が大幅に減少しない場合に、パフォーマンスが向上する場合があります。

NO_PX_JOIN_FILTERヒント

このヒントは、オプティマイザがパラレル結合のビットマップ・フィルタ処理を使用するのを防ぎます。

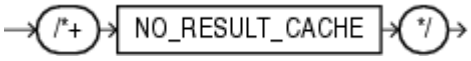
NO_QUERY_TRANSFORMATIONヒント



NO_QUERY_TRANSFORMATIONヒントは、すべての問合せ変換(OR拡張、ビュー・マージ、副問合せのネスト解除、スター型変換、マテリアライズド・ビュー・リライトなど)をスキップするようオプティマイザに指示するためのものです。たとえば:

```
SELECT /*+ NO_QUERY_TRANSFORMATION */ employee_id, last_name
  FROM (SELECT * FROM employees e) v
  WHERE v.last_name = 'Smith';
```

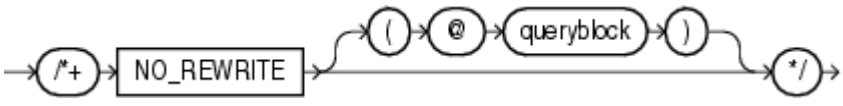
NO_RESULT_CACHEヒント



RESULT_CACHE_MODE初期化パラメータがFORCEに設定されていると、オプティマイザは、問合せ結果を結果キャッシュにキャッシュします。この場合、NO_RESULT_CACHEヒントにより、このような現行の問合せのキャッシュが無効になります。

問合せがOCIクライアントから実行され、OCIクライアントの結果キャッシュが有効になっている場合、NO_RESULT_CACHEヒントにより現行の問合せキャッシュが無効になります。

NO_REWRITEヒント



(ヒントでの問合せブロックの指定を参照)

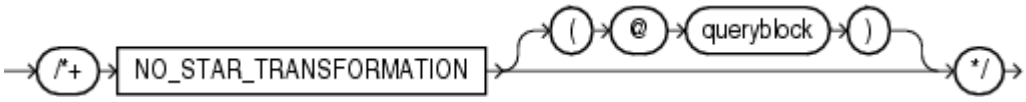
NO_REWRITEヒントは、パラメータQUERY_REWRITE_ENABLEDの設定を上書きして、問合せブロック用のクエリー・リライトを無効にするようオプティマイザに指示します。たとえば:

```
SELECT /*+ NO_REWRITE */ sum(s.amount_sold) AS dollars
  FROM sales s, times t
  WHERE s.time_id = t.time_id
  GROUP BY t.calendar_month_desc;
```

NOREWRITEヒント

NOREWRITEヒントは現在非推奨になっています。NO_REWRITEヒントをかわりに使用します。

NO_STAR_TRANSFORMATIONヒント



(ヒントでの問合せブロックの指定を参照)

NO_STAR_TRANSFORMATIONヒントは、問合せのスター型問合せ変換を実行しないようオプティマイザに指示します。

NO_STATEMENT_QUEUINGヒント

NO_STATEMENT_QUEUINGヒントは、パラレル文のキューイングによって文がキューに入れられるかどうかに影響します。

PARALLEL_DEGREE_POLICYをAUTOに設定すると、このヒントにより文がパラレル文キューに入らないようにすることができます。ただし、文のキューを回避する文は、パラレル文のキューイングを開始する制限を決定するPARALLEL_SERVERS_TARGET初期化パラメータの値で定義されるパラレル実行サーバーの最大数を超える可能性があります。

システムで使用できるパラレル実行サーバーの数がPARALLEL_MAX_SERVERS初期化パラメータの値までに制限されるため、パラレル文のキューイングを回避する文がリクエストされたパラレル実行サーバーの数を受け取る保証はありません。

たとえば:

```
SELECT /*+ NO_STATEMENT_QUEUING */ emp.last_name, dpt.department_name
  FROM employees emp, departments dpt
  WHERE emp.department_id = dpt.department_id;
```

関連項目:

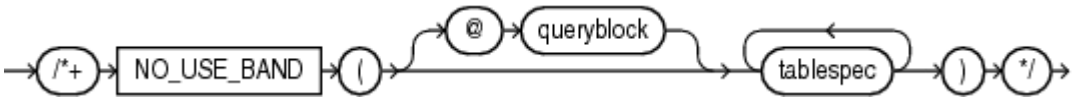
STATEMENT_QUEUINGヒント

NO_UNNESTヒント

(ヒントでの問合せブロックの指定を参照)

NO_UNNESTヒントを使用するとネスト解除をオフに切り替えます。

NO_USE_BANDヒント

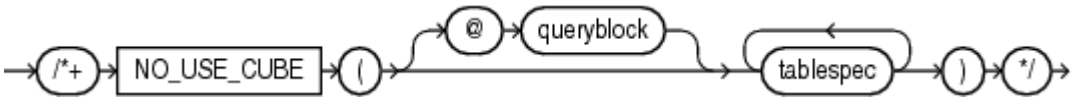


(ヒントでの問合せブロックの指定、tablespec::=を参照)

NO_USE_BANDヒントは、指定された各表を別の行のソースに結合する際にバンド結合を除外するようオプティマイザに指示します。たとえば:

```
SELECT /*+ NO_USE_BAND(e1 e2) */
  e1.last_name
  || ' has salary between 100 less and 100 more than '
  || e2.last_name AS "SALARY COMPARISON"
FROM employees e1, employees e2
WHERE e1.salary BETWEEN e2.salary - 100 AND e2.salary + 100;
```

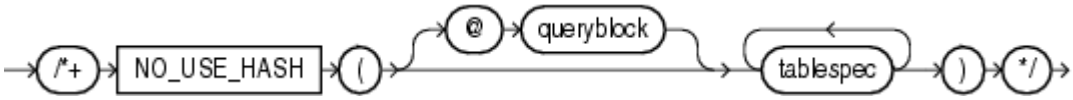
NO_USE_CUBEヒント



(ヒントでの問合せブロックの指定、tablespec::=を参照)

NO_USE_CUBEヒントは、指定された表を内部表として使用して、指定された各表を別の行のソースに結合する際にキューブ結合を除外するようオプティマイザに指示します。

NO_USE_HASHヒント

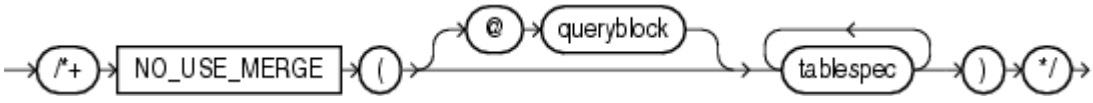


(ヒントでの問合せブロックの指定、tablespec::=を参照)

NO_USE_HASHヒントは、指定された表を内部表として使用して、指定された各表を別の行のソースに結合する際にハッシュ結合を除外するようオプティマイザに指示します。たとえば:

```
SELECT /*+ NO_USE_HASH(e d) */ *
  FROM employees e, departments d
  WHERE e.department_id = d.department_id;
```

NO_USE_MERGEヒント

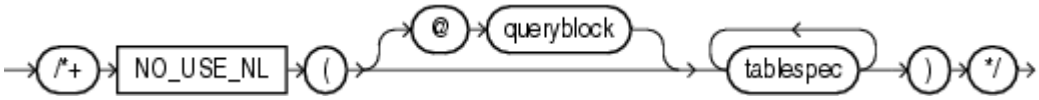


(ヒントでの問合せブロックの指定、tablespec::=を参照)

NO_USE_MERGEヒントは、指定された表を内部表として使用して、指定された各表を別の行のソースに結合する際にソート/マージ結合を除外するようオプティマイザに指示します。たとえば:

```
SELECT /*+ NO_USE_MERGE(e d) */ *
   FROM employees e, departments d
   WHERE e.department_id = d.department_id
   ORDER BY d.department_id;
```

NO_USE_NLヒント



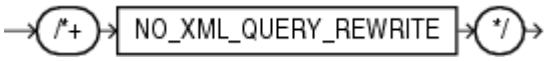
(ヒントでの問合せブロックの指定、tablespec::=を参照)

NO_USE_NLヒントは、指定された表を内部表として使用して、指定された各表を別の行のソースに結合する際に、ネストしたループ結合を除外するようオプティマイザに指示します。たとえば:

```
SELECT /*+ NO_USE_NL(l h) */ *
  FROM orders h, order_items l
  WHERE l.order_id = h.order_id
    AND l.order_id > 2400;
```

このヒントを指定すると、指定された表についてハッシュ結合とソート/マージ結合のみが検討されます。ただし、ネストしたループのみを使用して表を結合する場合もあります。この場合、オプティマイザは、それらの表に関するヒントを無視します。

NO_XML_QUERY_REWRITEヒント



NO_XML_QUERY_REWRITEヒントは、SQL文のXPath式のリライトを禁止するようオプティマイザに指示します。このヒントは、XPath式のリライトを禁止することで、現行の問合せでのXMLIndex索引の使用も禁止します。たとえば:

```
SELECT /*+NO_XML_QUERY_REWRITE*/ XMLQUERY('<A/>' RETURNING CONTENT)
  FROM DUAL;
```

関連項目:

NO_XMLINDEX_REWRITEヒント

NO_XMLINDEX_REWRITEヒント

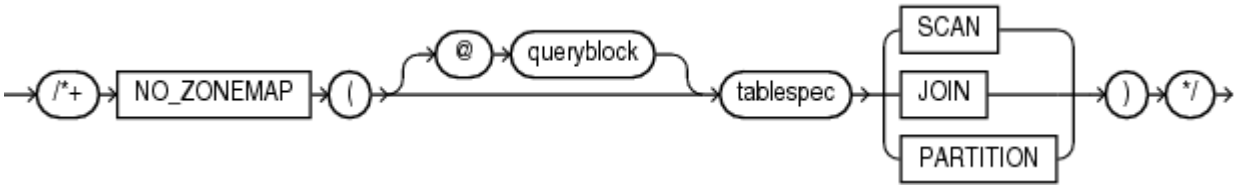
NO_XMLINDEX_REWRITEヒントは、現行の問合せにXMLIndex索引を使用しないようにオプティマイザに指示します。たとえば:

```
SELECT /*+NO_XMLINDEX_REWRITE*/ count(*)
  FROM warehouses
  WHERE existsNode(warehouse_spec, '/Warehouse/Building') = 1;
```

関連項目:

XMLIndexの使用を無効にするもう1つの方法については、NO_XML_QUERY_REWRITEヒントを参照してください。

NO_ZONEMAPヒント



(ヒントでの問合せブロックの指定、tablespec::=を参照)

NO_ZONEMAPヒントは、異なるタイプのプルーニングのゾーン・マップの使用を無効化します。このヒントは、ゾーン・マップを作成または変更したDDLのENABLE PRUNING設定をオーバーライドします。

次のオプションのいずれかを指定します。

- SCAN - スキャン・プルーニングのゾーン・マップの使用を無効化します。
- JOIN - 結合プルーニングのゾーン・マップの使用を無効化します。
- PARTITION - パーティション・プルーニングのゾーン・マップの使用を無効化します。

関連項目:

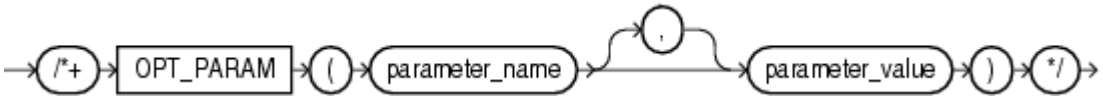
- CREATE MATERIALIZED ZONEMAPのENABLE | DISABLE PRUNING句
- ゾーン・マップのプルーニングに関する一般的な情報については、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

OPTIMIZER_FEATURES_ENABLEヒント

このヒントは、Databaseリファレンスのマニュアルに記載されています。

詳細は、Databaseリファレンスを参照してください。

OPT_PARAMヒント



OPT_PARAMヒントを使用すると、現行の問合せ中にのみ初期化パラメータを設定できます。このヒントは、パラメータAPPROX_FOR_AGGREGATION、APPROX_FOR_COUNT_DISTINCT、APPROX_FOR_PERCENTILE、OPTIMIZER_DYNAMIC_SAMPLING、OPTIMIZER_INDEX_CACHING、OPTIMIZER_INDEX_COST_ADJ、OPTIMIZER_SECURE_VIEW_MERGINGおよびSTAR_TRANSFORMATION_ENABLEDに対してのみ有効です。

たとえば、次のヒントは、ヒントを追加した文のパラメータSTAR_TRANSFORMATION_ENABLEDをTRUEに設定します。

```
SELECT /*+ OPT_PARAM('star_transformation_enabled' 'true') */ *
  FROM ... ;
```

文字列のパラメータ値は、一重引用符で囲まれます。数値のパラメータ値は、一重引用符で囲まずに指定されます。

ORDEREDヒント

ORDEREDヒントは、FROM句に現れる順序で表を結合するようOracleに指示します。ORDEREDヒントより多目的なLEADINGヒントを使用することをお薦めします。

結合を要求するSQL文からORDEREDヒントを削除すると、オプティマイザが表の結合順序を選択します。各表から選択した行数をオプティマイザが把握していないと考えられる場合、ORDEREDヒントを使用して結合順序を指定することがあります。この情報を使用すると、オプティマイザによる選択よりも効率良く内部表と外部表を選択できます。

次の問合せは、ORDEREDヒントの使用例です。

```
SELECT  /*+ ORDERED */ o.order_id, c.customer_id, l.unit_price * l.quantity
  FROM customers c, order_items l, orders o
  WHERE c.cust_last_name = 'Taylor'
    AND o.customer_id = c.customer_id
    AND o.order_id = l.order_id;
```

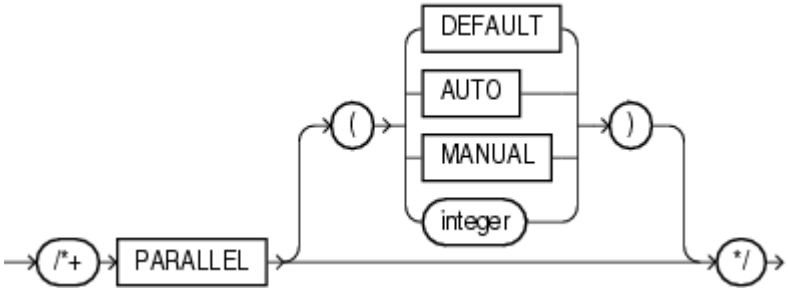
PARALLELヒント

パラレル・ヒントのノート

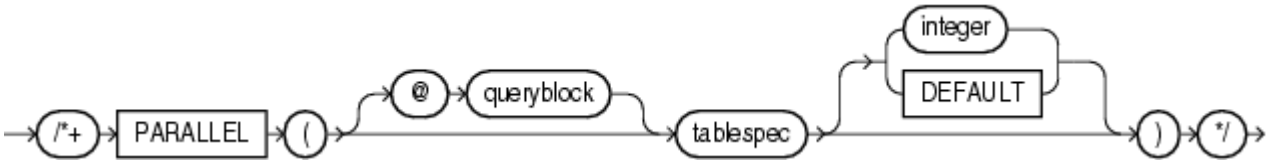
Oracle Database 11gリリース2以降では、PARALLELヒントとNO_PARALLELヒントは文レベルのヒントであり、以前のオブジェクト・レベルのPARALLEL_INDEXヒントとNO_PARALLEL_INDEXヒント、および以前に指定したPARALLELヒントとNO_PARALLELヒントより優先されます。PARALLELでintegerを指定すると、並列度は文に対して使用されます。integerを指定しない場合は、データベースで並列度が計算されます。パラレル化を使用できるすべてのアクセス・パスで、指定した並列度または計算された並列度が使用されます。

次の構文図で、parallel_hint_statementは文レベルのヒントの構文を示し、parallel_hint_objectはオブジェクト・レベルのヒントの構文を示します。オブジェクト・レベルのヒントは下位互換性のためにサポートされており、文レベルのヒントの方が優先されます。

parallel_hint_statement::=



parallel_hint_object::=



(ヒントでの問合せブロックの指定、tablespec::=を参照)

PARALLELヒントは、指定された数の同時サーバーをパラレル操作に使用するようオプティマイザに指示します。このヒントは、PARALLEL_DEGREE_POLICY初期化パラメータの値をオーバーライドします。このヒントは、文のSELECT、INSERT、MERGE、UPDATEおよびDELETE部分と表のスキャン部分に適用されます。パラレル制限に違反すると、ヒントは無視されます。

ノート:

ソート操作またはグループ操作も実行する場合、使用可能なサーバー数は PARALLEL ヒントの値の 2 倍となります。

文レベルのPARALLELヒントでは、次のように動作します。

- PARALLEL: 文は、最も低いコスト計画でパラレル化が実現できない場合を除き、計算された並列度以上になります。パラレル化が実現できない場合、文はシリアルで実行されます。
- PARALLEL (DEFAULT): オプティマイザは並列度を計算します。並列度は、すべての関係するインスタンスで使用可能なCPUの数に、初期化パラメータPARALLEL_THREADS_PER_CPUの値を掛けたものです。
- PARALLEL (AUTO): 文は、最も低いコスト計画でパラレル化が実現できない場合を除き、計算された並列度以上になります。パラレル化が実現できない場合、文はシリアルで実行されます。
- PARALLEL (MANUAL): オプティマイザは強制的に、文の中のオブジェクトのパラレル設定を使用します。
- PARALLEL (integer): オプティマイザは、integerに指定された並列度を使用します。

次の例では、オプティマイザは並列度を計算します。文は常にパラレルで実行されます。

```
SELECT /*+ PARALLEL */ last_name
  FROM employees;
```

次の例では、オプティマイザは並列度を計算しますが、並列度は1の場合があります。その場合、文はシリアルで実行されます。

```
SELECT /*+ PARALLEL (AUTO) */ last_name
  FROM employees;
```

次の例では、PARALLELヒントは、表自体に現在適用されている並列度(5)を使用するようオプティマイザに指示します。

```
CREATE TABLE parallel_table (col1 number, col2 VARCHAR2(10)) PARALLEL 5;
SELECT /*+ PARALLEL (MANUAL) */ col2
  FROM parallel_table;
```

オブジェクト・レベルのPARALLELヒントでは、次のように動作します。

- PARALLEL: 問合せコーディネータはデフォルトの並列度を決定するために初期化パラメータの設定を検証する必要があります。
- PARALLEL (integer): オプティマイザは、integerに指定された並列度を使用します。
- PARALLEL (DEFAULT): オプティマイザは並列度を計算します。並列度は、すべての関係するインスタンスで使用可能なCPUの数に、初期化パラメータPARALLEL_THREADS_PER_CPUの値を掛けたものです。

次の例では、PARALLELヒントが、employees表定義内で指定された並列度を上書きします。

```
SELECT /*+ FULL(hr_emp) PARALLEL(hr_emp, 5) */ last_name
  FROM employees hr_emp;
```

次の例では、PARALLELヒントが、employees表定義内で指定された並列度を上書きし、並列度を計算するようにオプティマイザに指示します。並列度は、すべての関係するインスタンスで使用可能なCPUの数に初期化パラメータPARALLEL_THREADS_PER_CPUの値を掛けたものです。

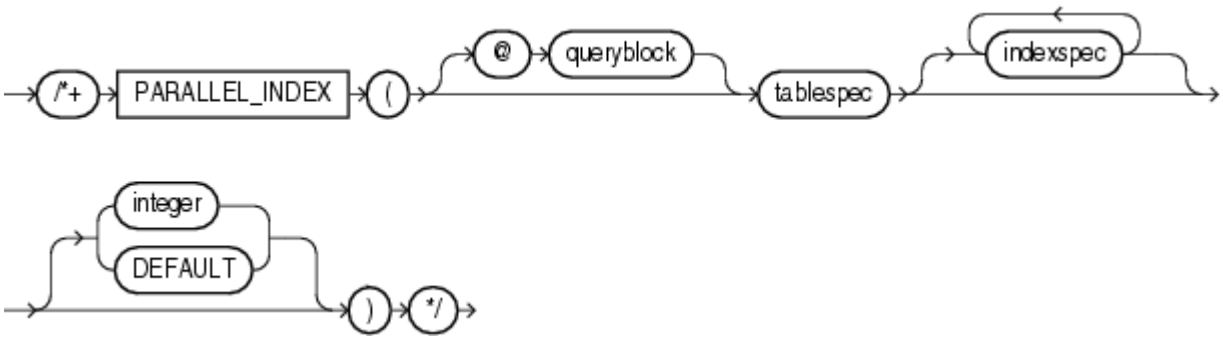
```
SELECT /*+ FULL(hr_emp) PARALLEL(hr_emp, DEFAULT) */ last_name
  FROM employees hr_emp;
```

パラレル実行の詳細は、CREATE TABLEおよび『Oracle Database概要』を参照してください。

関連項目:

- パラレル実行の詳細は、CREATE TABLEおよび『Oracle Database概要』を参照してください。
- DBMS_PARALLEL_EXECUTEパッケージ(行のチャンクの変更を表に適用するメソッドを含むパッケージ)の詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。エラーがない場合は、各チャンクに対する変更が個別にコミットされます。
- PARALLEL_DEGREE_POLICY初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。
- NO_PARALLELヒント

PARALLEL_INDEXヒント



(ヒントでの問合せブロックの指定、tablespec::=、indexspec::=を参照)

PARALLEL_INDEXヒントは、パーティション索引について索引レンジ・スキャン、全体スキャンおよび高速全体スキャンをパラレル化するために、指定された数の同時サーバーを使用するようオプティマイザに指示します。

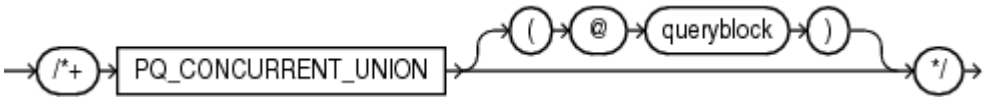
integer値は、指定された索引の並列度を示します。DEFAULTを指定するか、いかなる値も指定しない場合、問合せコーディネータはデフォルトの並列度を決定するために初期化パラメータの設定を検証する必要があります。たとえば、次のヒントは、3つのパラレル実行プロセスが使用されることを示します。

```
SELECT /*+ PARALLEL_INDEX(table1, index1, 3) */
```

関連項目:

パラレル・ヒントの詳細は、パラレル・ヒントのノートを参照してください。

PQ_CONCURRENT_UNIONヒント



(ヒントでの問合せブロックの指定を参照)

PQ_CONCURRENT_UNIONヒントは、UNION操作とUNION ALL操作の同時処理を有効にするようオプティマイザに指示します。

関連項目:

- NO_PQ_CONCURRENT_UNIONヒント
- このヒントの使用方法の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

PQ_DISTRIBUTEヒント

(ヒントでの問合せブロックの指定、tablespec::=を参照)

PQ_DISTRIBUTEヒントは、プロデューサおよびコンシューマ問合せサーバーに行を分散させる方法をオプティマイザに指示します。結合またはロードのいずれかで行の分散を制御できます。

ロードでの分散の制御

パラレルINSERT ... SELECT文およびパラレルCREATE TABLE ... AS SELECT文で行の分散を制御し、プロデューサ(問合せ)およびコンシューマ(ロード)サーバー間で行を分散する方法を指示できます。構文の上位ブランチを使用するには、1つの分散方法を指定します。表2-24に、分散方法の値とそのセマンティクスを示します。

表2-24 ロードでの分散処理の値

分散	説明
NONE	分散処理なし。したがって、問合せおよびロード操作は、問合せサーバーごとに組み合されます。すべてのサーバーがすべてのパーティションをロードします。分散を行わないことで、偏りがない場合に行の分散によって発生するオーバーヘッドを避けることができます。偏りは、空のセグメントによって発生したり、文の条件で問合せによって評価されるすべての行が排除される場合に発生することがあります。この方法を使用することで偏りが発生する場合、かわりに RANDOM または RANDOM_LOCAL のいずれかの分散を使用します。 ノート: この分散方法は、慎重に使用してください。ロードされる各パーティションに対し、プロセスごとに 512 KB 以上の PGA メモリーが必要です。圧縮も同時に使用している場合、サーバーごとに約 1.5 MB の PGA メモリーが消費されます。

分散	説明
PARTITION	この方法は、tablespec のパーティション情報を使用して、問合せサーバーからロード・サーバーへ行を分散します。この分散方法は、問合せおよびロード操作を組み合せることができない(または望ましくない)場合、すなわち、ロードされるパーティションの数がロード・サーバーの数以上で、ロードされるパーティション間で入力データが均等に分散される(偏りがない)場合に使用します。
RANDOM	この方法は、プロデューサの行をラウンドロビン法でコンシューマに分散します。この分散方法は、入力データの偏りが大きい場合に使用します。
RANDOM_LOCAL	この方法は、特定のパーティションの集合を保持する役割を果たすサーバーの集合に対して、プロデューサの行を分散します。2 つ以上のサーバーが同じパーティションをロードする可能性はありますが、すべてのパーティションをロードするサーバーは存在しません。この分散方法は、入力データが偏っており、メモリーの制約によって問合せおよびロード操作を組み合せることができない場合に使用します。

たとえば、次のダイレクト・パス・インサート操作の問合せおよびロード部分は、問合せサーバーごとに組み合されます。

```
INSERT /*+ APPEND PARALLEL(target_table, 16) PQ_DISTRIBUTE(target_table, NONE) */
  INTO target_table
  SELECT * FROM source_table;
```

次の表を作成する例では、オプティマイザは、target_tableのパーティション化を使用して行を分散します。

```
CREATE /*+ PQ_DISTRIBUTE(target_table, PARTITION) */ TABLE target_table
  NOLOGGING PARALLEL 16
  PARTITION BY HASH (l_orderkey) PARTITIONS 512
  AS SELECT * FROM source_table;
```

結合での分散の制御

構文図の下位ブランチに示すように、2つの分散方法(外部表への分散処理と内部表への分散処理)を指定することで、結合での分散方法を制御できます。

- outer_distributionは、外部表への分散処理です。
- inner_distributionは、内部表への分散処理です。

分散の値は、HASH、BROADCAST、PARTITIONおよびNONEです。表2-25で説明するとおり、6つの組合せの表分散のみが有効です。

表2-25 結合での分散処理の値

分散	説明
HASH、HASH	各表の行は、結合キーにハッシュ関数を使用して、コンシューマ問合せサーバーにマップされます。マッピングが完了すると、各問合せサーバーは、結果として生成されるパー

分散	説明
	ティションの組で結合を実行します。この分散は、表のサイズがほぼ等しく、結合操作がハッシュ結合またはソート/マージ結合で実施される場合にお薦めします。
BROADCAST、NONE	外部表のすべての行が各問合せサーバーにブロードキャストされます。内部表の行は、ランダムにパーティション化されます。この分散は、外部表のサイズが内部表よりもきわめて小さい場合にお薦めします。一般的に、内部表のサイズに問合せサーバーの数を乗じた数値が外部表のサイズよりも大きい場合、この分散を使用します。
NONE、BROADCAST	内部表のすべての行が各コンシューマ問合せサーバーにブロードキャストされます。外部表の行は、ランダムにパーティション化されます。この分散は、内部表のサイズが外部表よりもきわめて小さい場合にお薦めします。一般的に、内部表のサイズに問合せサーバーの数を乗じた数値が外部表のサイズより小さい場合、この分散を使用します。
PARTITION、NONE	外部表の行は、内部表のパーティション化を使用してマップされます。内部表は、結合キーでパーティション化されている必要があります。この分散は、外部表のパーティションの数が問合せサーバーの数と等しいかほぼ等しい(たとえば、パーティション数が 14 で問合せサーバー数が 15)場合にお薦めします。 ノート: オプティマイザは、内部表がパーティション化されていないか、パーティション化キーと等価結合関係にない場合、このヒントを無視します。
NONE、PARTITION	内部表の行は、外部表のパーティション化を使用してマップされます。外部表は、結合キーでパーティション化されている必要があります。この分散は、外部表のパーティションの数が問合せサーバーの数と等しいかほぼ等しい(たとえば、パーティション数が 14 で問合せサーバー数が 15)場合にお薦めします。 ノート: オプティマイザは、外部表がパーティション化されていないか、パーティション化キーと等価結合関係にない場合、このヒントを無視します。
NONE、NONE	各問合せサーバーは、各表から抽出した一致パーティションの組で結合操作を実行します。両方の表は、結合キーに基づいて同一レベルでパーティション化されている必要があります。

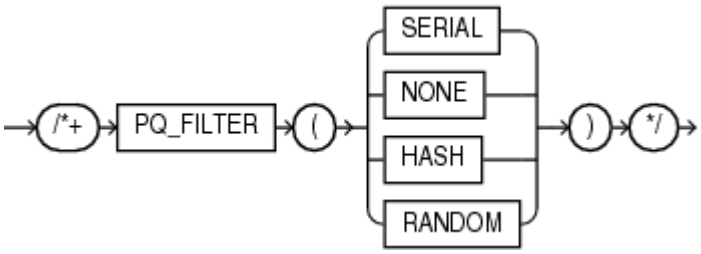
たとえば、rとsという2つの表がハッシュ結合によって結合されている場合、次の問合せには、ハッシュ分散を使用するためのヒントが含まれます。

```
SELECT /*+ORDERED PQ_DISTRIBUTE(s HASH, HASH) USE_HASH (s)*/ column_list
  FROM r,s
  WHERE r.c=s.c;
```

外部表rをブロードキャストするための問合せは、次のとおりです。

```
SELECT /*+ORDERED PQ_DISTRIBUTE(s BROADCAST, NONE) USE_HASH (s) */ column_list
  FROM r,s
  WHERE r.c=s.c;
```

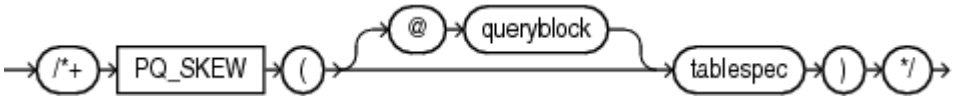
PQ_FILTERヒント



PQ_FILTERヒントは、相関副問合せをフィルタリングするときの行の処理方法についてオプティマイザに指示します。

- SERIAL: フィルタの左側と右側で行を逐次処理します。このオプションは、問合せにとってパラレル化のオーバーヘッドが大きすぎる場合(たとえば左側の行が非常に少ない場合など)に使用します。
- NONE: フィルタの左側と右側で行をパラレル処理します。このオプションは、フィルタの左側でデータの分散に偏りがなく、左側の分散を回避する(たとえば左側のサイズが大きいため)場合に使用します。
- HASH: フィルタの左側でハッシュ分散を使用して行をパラレル処理します。フィルタの右側では行を逐次処理します。このオプションは、フィルタの左側でデータの分散に偏りがない場合に使用します。
- RANDOM: フィルタの左側でランダム分散を使用して行をパラレル処理します。フィルタの右側では行を逐次処理します。このオプションは、フィルタの左側でデータの分散に偏りがある場合に使用します。

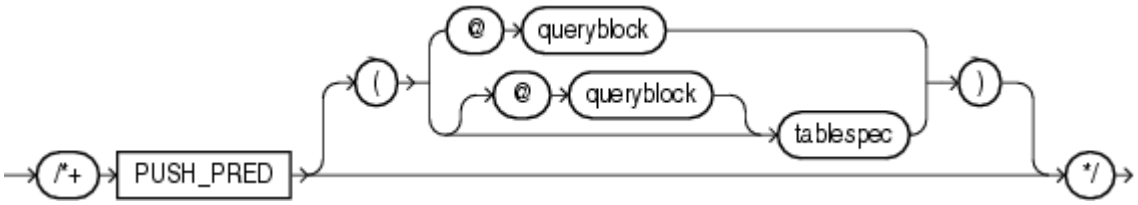
PQ_SKEWヒント



(ヒントでの問合せブロックの指定、tablespec::=を参照)

PQ_SKEWヒントは、パラレル結合の結合キーの値の分散が著しく偏っている(つまり、同じ結合キー値を持つ行の割合が大きい)ことをオプティマイザに知らせます。tablespecで指定される表は、ハッシュ結合のプローブ表です。

PUSH_PREDヒント



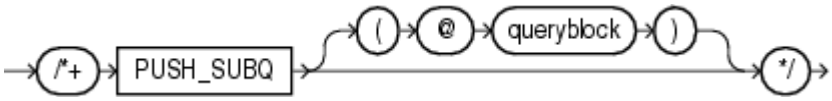
(ヒントでの問合せブロックの指定、tablespec::=を参照)

PUSH_PREDヒントは、結合述語をビューにプッシュするようオプティマイザに指示します。たとえば:

```
SELECT /*+ NO_MERGE(v) PUSH_PRED(v) */ *
  FROM employees e,
    (SELECT manager_id
      FROM employees) v
```

```
  WHERE e.manager_id = v.manager_id(+)
    AND e.employee_id = 100;
```

PUSH_SUBQヒント

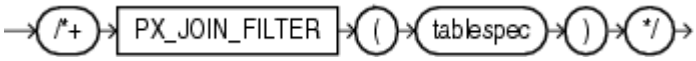


(ヒントでの問合せブロックの指定を参照)

PUSH_SUBQヒントは、実行計画の初期段階で実行可能なステップで、マージされていない副問合せを評価するようオプティマイザに指示します。通常、マージされていない副問合せは、実行計画の最終ステップで実行されます。副問合せのコストが比較的低く、副問合せによって行数が大幅に減少する場合、副問合せの早期評価によってパフォーマンスが向上する可能性があります。

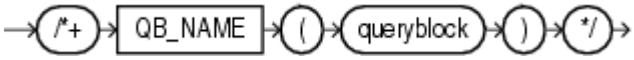
このヒントは、副問合せがリモート表か、マージ結合によって結合されている表に適用される場合には効果がありません。

PX_JOIN_FILTERヒント



このヒントは、パラレル結合のビットマップ・フィルタ処理の使用をオプティマイザに強制します。

QB_NAMEヒント



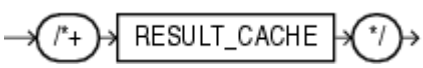
(ヒントでの問合せブロックの指定を参照)

QB_NAMEヒントを使用すると、問合せブロックの名前を定義できます。この名前は外部問合せ内のヒントまたはインライン・ビュー内のヒントで使用でき、名前が付けられた問合せブロックにある表で実行する問合せに影響をおよぼします。

2つ以上の問合せブロックに同じ名前が付いている場合や、同じ問合せブロックに対して2回、それぞれ異なる名前でヒントが適用されている場合は、オプティマイザはすべての名前を無視し、その問合せブロックを参照しているヒントも無視します。このヒントを使用して名前が付けられてはいない問合せブロックには、システムによって生成された一意の名前が付けられます。この名前は計画表に表示できます。また、問合せブロック内のヒントや問合せブロック・ヒントでも使用できます。たとえば:

```
SELECT /*+ QB_NAME(qb) FULL(@qb e) */ employee_id, last_name
  FROM employees e
  WHERE last_name = 'Smith';
```

RESULT_CACHEヒント



RESULT_CACHEヒントは、メモリー内の現行の問合せまたは問合せのフラグメントの結果をキャッシュし、今後の問合せまたは問合せのフラグメントの実行時にキャッシュした結果を使用するようデータベースに指示します。このヒントは、トップレベル問合せ、subquery_factoring_clauseまたはFROM句のインライン・ビューで認識されます。キャッシュ結果は、共有プールの

結果のキャッシュ・メモリー部分に保存されます。

作成に使用されたデータベース・オブジェクトが正常に修正されると、キャッシュ結果は自動的に無効化されます。このヒントは、RESULT_CACHE_MODE初期化パラメータの設定よりも優先されます。

問合せが結果キャッシュに使用できるのは、問合せで必要とされるすべてのファンクション(たとえば、組込みファンクション、ユーザー定義ファンクションまたは仮想列)が決定的である場合にかぎられます。

問合せがOCIクライアントから実行され、OCIクライアントの結果キャッシュが有効になっている場合、RESULT_CACHEヒントにより現行の問合せのクライアントのキャッシュが有効になります。

関連項目:

このヒントの使用方法については『Oracle Databaseパフォーマンス・チューニング・ガイド』、RESULT_CACHE_MODE初期化パラメータの詳細は『*Oracle Database*リファレンス』、OCI結果キャッシュの詳細および使用のガイドラインについては『Oracle Call Interfaceプログラマーズ・ガイド』を参照してください。

RETRY_ON_ROW_CHANGEヒント

ノート:

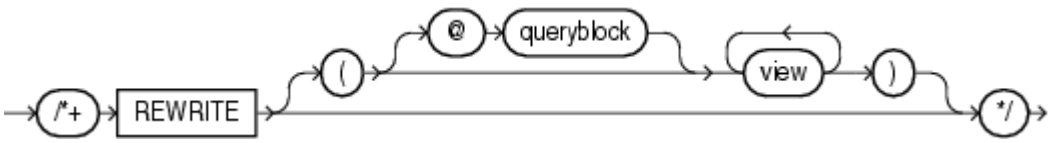
CHANGE_DUPKEY_ERROR_INDEX ヒント、IGNORE_ROW_ON_DUPKEY_INDEX ヒントおよびRETRY_ON_ROW_CHANGE ヒントは、セマンティクスに影響を与えるという点で他のヒントとは異なります。これら 3 つのヒントには、ヒントで説明されている一般的な原則は当てはまりません。

このヒントが有効になるのは、UPDATE操作およびDELETE操作に対してのみです。INSERT操作およびMERGE操作に対してはサポートされません。このヒントを指定すると、変更対象の行セットが決定された時点から、ブロックが実際に変更される時点までの間に、セット内の1つ以上の行のORA_ROWSCNが変更された場合に、操作が再試行されます。

関連項目:

「IGNORE_ROW_ON_DUPKEY_INDEXヒント」および「CHANGE_DUPKEY_ERROR_INDEXヒント」

REWRITEヒント



(ヒントでの問合せブロックの指定を参照)

REWRITEヒントは、可能な場合、コストを考慮することなく、マテリアライズド・ビューに関する問合せをリライトするようオプティマイザに指示します。REWRITEヒントは、ビュー・リストとともに、またはビュー・リストなしで使用します。ビュー・リストとともにREWRITEを使用し、リストに適切なマテリアライズド・ビューが含まれている場合、Oracleはコストを考慮せずにそのビューを使

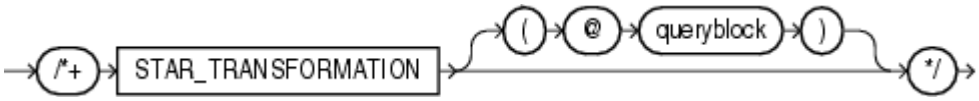
用します。

Oracleでは、リスト外のビューを検討しません。ビュー・リストを指定しない場合、Oracleは適切なマテリアライズド・ビューを検索し、最終計画のコストを考慮することなく常にそのビューを使用します。

関連項目:

- マテリアライズド・ビューの詳細は、『Oracle Database概要』を参照してください。
- マテリアライズド・ビューでREWRITEを使用する場合の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

STAR_TRANSFORMATIONヒント



(ヒントでの問合せブロックの指定を参照)

STAR_TRANSFORMATIONヒントは、変換を行う際に最適な計画を使用するようオプティマイザに指示します。このヒントを使用しない場合、オプティマイザは、変換された問合せ用の最適な計画のかわりに、変換なしで生成された最適な計画を使用するという、問合せの最適化に関する決定を行う場合があります。たとえば:

```
SELECT /*+ STAR_TRANSFORMATION */ s.time_id, s.prod_id, s.channel_id
  FROM sales s, times t, products p, channels c
  WHERE s.time_id = t.time_id
    AND s.prod_id = p.prod_id
    AND s.channel_id = c.channel_id
    AND c.channel_desc = 'Tele Sales';
```

ヒントが指定された場合でも、変換が実行される保証はありません。オプティマイザは、妥当と考えられる場合にかぎって副問合せを生成します。副問合せが生成されない場合には、変換された問合せが存在しないため、ヒントに関係なく、未変換の問合せに関する最適な計画が使用されます。

関連項目:

- スター型変換の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- STAR_TRANSFORMATION_ENABLED初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

STATEMENT_QUEUINGヒント

NO_STATEMENT_QUEUINGヒントは、パラレル文のキューイングによって文がキューに入れられるかどうかに影響します。

PARALLEL_DEGREE_POLICYをAUTOに設定しない場合は、このヒントにより文をパラレル文のキューイングに考慮し、リクエストされたDOPで十分なパラレル処理を実行できる場合にのみ実行させることができます。キューイングを有効にする前の使用できるパラレル実行サーバーの数は、使用するパラレル実行サーバーの数とPARALLEL_SERVERS_TARGET初期化パラメー

タで定義されるシステムで許可される最大数の違いと同じです。

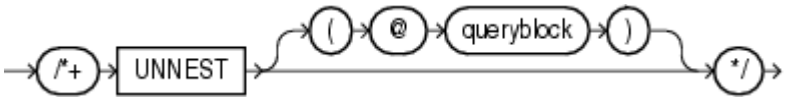
たとえば:

```
SELECT /*+ STATEMENT_QUEUING */ emp.last_name, dpt.department_name
  FROM employees emp, departments dpt
  WHERE emp.department_id = dpt.department_id;
```

関連項目:

NO_STATEMENT_QUEUINGヒント

UNNESTヒント



(ヒントでの問合せブロックの指定を参照)

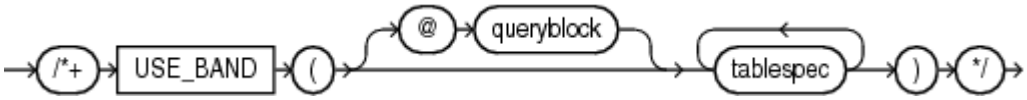
UNNESTヒントは、副問合せの本体のネストを解除し、その副問合せを含む問合せブロック本体にマージするようオプティマイザに指示します。これによって、アクセス・パスおよび結合の評価時に、オプティマイザが副問合せと問合せブロックを総合して考慮できるようになります。

副問合せのネストを解除する前に、オプティマイザは、文が有効かどうかをまず検討します。文は、経験則に基づくテストと問合せ最適化テストに合格する必要があります。UNNESTヒントは、副問合せブロックの有効性のみをチェックするようオプティマイザに指示します。副問合せブロックが有効な場合、経験則またはコストをチェックすることなく副問合せのネストを解除できます。

関連項目:

- ネスト化された副問合せのネスト解除、および副問合せブロックが有効となる条件については、コレクション・ネスト解除: 例を参照してください。
- 副問合せのネスト解除の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

USE_BANDヒント



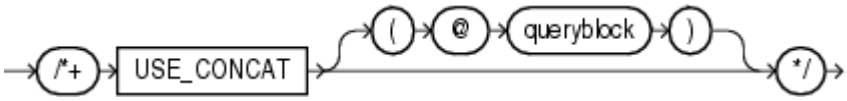
(ヒントでの問合せブロックの指定、tablespec::=を参照)

USE_BANDヒントは、指定された各表を、バンド結合を使用して別の行のソースに結合するようオプティマイザに指示します。たとえば:

```
SELECT /*+ USE_BAND(e1 e2) */
  e1.last_name
  || ' has salary between 100 less and 100 more than '
  || e2.last_name AS "SALARY COMPARISON"
FROM employees e1, employees e2
WHERE e1.salary BETWEEN e2.salary - 100 AND e2.salary + 100;
```

USE_BANDヒントに表が表示される順序によって結合順序は指定されません。特定の結合順序をヒントにするには、LEADINGヒントが必要です。

USE_CONCATヒント



(ヒントでの問合せブロックの指定を参照)

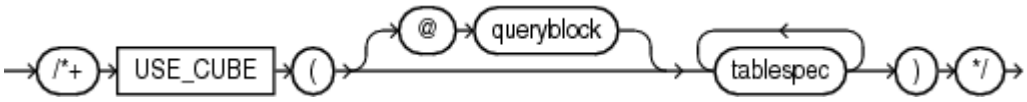
USE_CONCATヒントは、問合せのWHERE句内で組み合されたOR条件を、集合演算子UNION ALLを使用して複合問合せに変換するようオプティマイザに指示します。このヒントを使用しない場合、この変換は、連結を使用した問合せのコストが、使用しない場合よりも低い場合にのみ実行されます。USE_CONCATヒントは、コストより優先します。たとえば:

```
SELECT /*+ USE_CONCAT */ *
  FROM employees e
  WHERE manager_id = 108
     OR department_id = 110;
```

関連項目:

NO_EXPANDヒントを参照してください(このヒントの反対の機能です)。

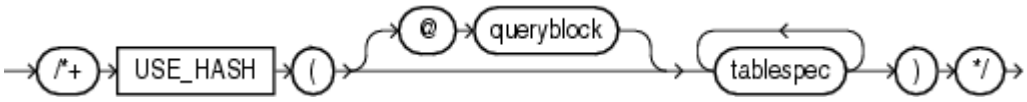
USE_CUBEヒント



(ヒントでの問合せブロックの指定、tablespec::=を参照)

結合の右側がキューブの場合、USE_CUBEヒントは、指定された各表を、キューブ結合を使用して別の行のソースに結合するようにオプティマイザに指示します。オプティマイザが統計分析に基づいてキューブ結合を使用しないことを決定したときには、USE_CUBEを使用するとオプティマイザの決定を無視できます。

USE_HASHヒント



(ヒントでの問合せブロックの指定、tablespec::=を参照)

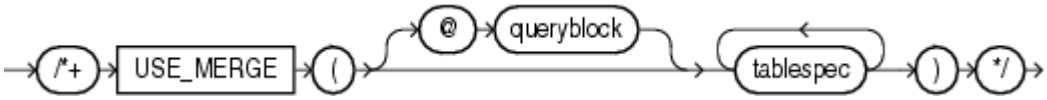
USE_HASHヒントは、指定された各表を、ハッシュ結合を使用して別の行のソースに結合するようオプティマイザに指示します。たとえば:

```
SELECT /*+ USE_HASH(l h) */ *
  FROM orders h, order_items l
  WHERE l.order_id = h.order_id
    AND l.order_id > 2400;
```

USE_HASHヒントに表が表示される順序によって結合順序は指定されません。特定の結合順序をヒントにするには、

LEADINGヒントが必要です。

USE_MERGEヒント



(ヒントでの問合せブロックの指定、tablespec::=を参照)

USE_MERGEヒントは、指定された各表を、ソート/マージ結合を使用して別の行のソースに結合するようオプティマイザに指示します。たとえば:

```
SELECT /*+ USE_MERGE(employees departments) */ *
  FROM employees, departments
  WHERE employees.department_id = departments.department_id;
```

LEADINGおよびORDEREDヒントとともに、USE_NLおよびUSE_MERGEヒントを使用することをお薦めします。オプティマイザは、参照表を結合の内部表にする必要がある場合に、これらのヒントを使用します。参照表が外部表の場合、ヒントは無視されます。

USE_NLヒント

USE_NLヒントは、指定された表を内部表として使用し、指定された各表をネストしたループ結合とともに別の行のソースに結合するようオプティマイザに指示します。

(ヒントでの問合せブロックの指定、tablespec::=を参照)

LEADINGおよびORDEREDヒントとともに、USE_NLおよびUSE_MERGEヒントを使用することをお薦めします。オプティマイザは、参照表を結合の内部表にする必要がある場合に、これらのヒントを使用します。参照表が外部表の場合、ヒントは無視されます。

次の例では、ネストしたループがヒントによって強制される場合、全表スキャンを介してordersがアクセスされ、各行にフィルタ条件l.order_id = h.order_idが適用されます。フィルタ条件を満たす各行については、索引order_idを介してorder_itemsがアクセスされます。

```
SELECT /*+ USE_NL(l h) */ h.customer_id, l.unit_price * l.quantity
  FROM orders h, order_items l
  WHERE l.order_id = h.order_id;
```

USE_NLヒントに表が表示される順序によって結合順序は指定されません。特定の結合順序をヒントにするには、LEADINGヒントが必要です。

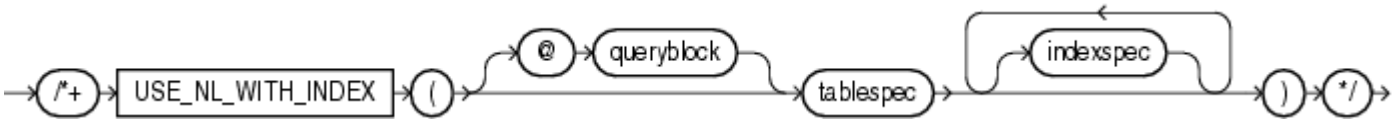
例

```
select /*+ LEADING(t2) USE_NL(t1) */ sum(t1.a),sum(t2.a)
from   t1 , t2
where   t1.b = t2.b;
select * from table(dbms_xplan.display_cursor()) ;
```

INDEXヒントを問合せに追加すると、ordersの全表スキャンを回避し、より大規模なシステムで使用される実行計画と同様

の実行計画を生成できる場合があります。ただし、ここでは特に効果的ではない場合があります。

USE_NL_WITH_INDEXヒント



(ヒントでの問合せブロックの指定、tablespec::=、indexspec::=を参照)

USE_NL_WITH_INDEXヒントは、指定された表を内部表として使用し、指定された表をネストしたループ結合とともに別の行のソースに結合するようオプティマイザに指示します。たとえば:

```
SELECT /*+ USE_NL_WITH_INDEX(l item_product_ix) */ *
  FROM orders h, order_items l
  WHERE l.order_id = h.order_id
    AND l.order_id > 2400;
```

次の条件が適用されます。

- 索引を指定しない場合、オプティマイザは、1つ以上の結合述語とともに、索引キーとして一部の索引を使用できる必要があります。
- 索引を指定する場合、オプティマイザは、1つ以上の結合述語とともに、索引キーとしてその索引を使用できる必要があります。

データベース・オブジェクト

次の項で説明するとおり、Oracle Databaseは、特定のスキーマに対応付けられたオブジェクトと、特定のスキーマに対応付けられていないオブジェクトを認識します。

スキーマ・オブジェクト

スキーマとは、データの論理構造(スキーマ・オブジェクト)の集まりです。スキーマはデータベース・ユーザーによって所有され、そのユーザーと同じ名前を持ちます。ユーザーはそれぞれ1つのスキーマを所有します。スキーマ・オブジェクトの作成と操作はSQLで行うことができ、オブジェクトのタイプには次のものがあります。

- 分析ビュー
- 属性ディメンション
- クラスタ
- 制約
- データベース・リンク
- データベース・トリガー
- ディメンション
- 外部プロシージャ・ライブラリ
- 階層
- 索引構成表
- 索引
- 索引タイプ
- Javaクラス
- Javaリソース
- Javaソース
- 結合グループ
- マテリアライズド・ビュー
- マテリアライズド・ビュー・ログ
- マイニング・モデル
- オブジェクト表
- オブジェクト型
- オブジェクト・ビュー
- 演算子
- パッケージ
- 順序
- ストアド・ファンクション
- ストアド・プロシージャ
- シノニム
- 表
- ビュー
- ゾーン・マップ

非スキーマ・オブジェクト

次のタイプのオブジェクトもデータベースに格納され、SQLで作成および操作されますが、スキーマには含まれません。

- コンテキスト
- ディレクトリ
- エディション
- フラッシュバック・アーカイブ
- ロックダウン・プロファイル
- プロファイル
- リストア・ポイント
- ロール
- ロールバック・セグメント
- 表領域
- 表領域セット
- 統合監査ポリシー
- ユーザー

このリファレンスでは、データベース・オブジェクトを作成する文に関する項で、オブジェクトの各タイプについて説明します。これらの文は、キーワードCREATEで始まります。たとえば、クラスタの定義については、「CREATE CLUSTER」を参照してください。

関連項目:

データベース・オブジェクトの概要については、『Oracle Database概要』を参照してください。

ほとんどのデータベース・オブジェクトでは、作成時に名前を指定する必要があります。名前は、この後の項に示す規則に従って付けてください。

データベース・オブジェクト名および修飾子

データベース・オブジェクトの中には、複数のコンポーネントから構成されていて、各コンポーネントに名前を付けることが可能または必須のものがあります(たとえば、表またはビュー内の列、索引および表のパーティションとサブパーティション、表の整合性制約、パッケージ内に格納されるオブジェクト(プロシージャやストアド・ファンクションなど))。この項の内容は、次のとおりです。

- データベース・オブジェクトとデータベース・オブジェクトの位置修飾子のネーミング規則
- データベース・オブジェクトと修飾子のネーミングのガイドライン

ノート:

Oracle では、暗黙的に生成されるデータベース・オブジェクトとサブオブジェクトには「SYS_」で始まるシステム生成名を使用し、Oracle が提供する一部のオブジェクトには、「ORA_」で始まる名前を使用します。名前解決での競合を避けるため、これらの接頭辞は明示的に指定するデータベース・オブジェクト名やサブオブジェクト名では使用しないでください。

データベース・オブジェクトのネーミング規則

すべてのデータベース・オブジェクトには、名前があります。SQL文では、引用識別子または非引用識別子を使用して、オブジェクトの名前を表します。

- 引用符付きの識別子は、先頭と末尾に二重引用符(")を付けます。引用符付きの識別子を使用してスキーマ・オブジェクトを命名した場合は、そのオブジェクトを参照するときに必ず二重引用符を使用する必要があります。
- 引用符なしの識別子は記号で囲みません。

データベース・オブジェクトを指定する場合、引用識別子または非引用識別子を使用できます。データベース名、グローバル・データベース名、データベース・リンク名、ディスク・グループ名およびプラガブル・データベース(PDB)名では大/小文字は区別されませんが、大文字として保存されます。このような名前を引用識別子として指定する場合、引用符は特に警告もなく無視されます。

関連項目:

ユーザー名とパスワードのネーミング規則の詳細は、「CREATE USER」を参照してください。

ノート:

データベース・オブジェクト名には引用識別子を使用しないことをお薦めします。SQL*Plus ではこのような引用識別子は許可されていますが、データベース・オブジェクトを管理する他のツールでは有効でない場合があります。

特に記載のある場合を除いて、次の規則が引用符付きと引用符なしの両方の識別子に適用されます。

1. 識別子名の最大長は、COMPATIBLE初期化パラメータの値によって異なります。
    - COMPATIBLE が12.2以上の値に設定されている場合、次の例外を除いて、名前は1から128バイトの長さにする必要があります。

- データベースの名前は、8バイトまでに制限されています。
- ディスク・グループ、プラガブル・データベース(PDB)、ロールバック・セグメント、表領域および表領域セットの名前は、30バイトまでに制限されています。

識別子にピリオドで区切られた複数の部分が含まれる場合、各属性の長さは最大128バイトにできます。セパレータのそれぞれのピリオド、および取り囲んでいる二重引用符は1バイトとしてカウントされます。たとえば、次のように列を識別するとします。

```
"schema"."table"."column"
```

スキーマ名、表名、列名の長さはそれぞれ128バイトです。各引用符やピリオドはシングルバイト文字のため、この例での識別子の総バイト長は、最大392バイトになります。

- COMPATIBLE が12.2未満の値に設定されている場合、次の例外を除いて、名前は1から30バイトの長さにする必要があります。
  - データベースの名前は、8バイトまでに制限されています。
  - データベース・リンクの名前は、128バイトまで指定できます。

識別子がピリオドで区切られた複数の部分から成る場合、各属性は30バイトまでの長さで指定できます。セパレータのそれぞれのピリオド、および取り囲んでいる二重引用符は1バイトとしてカウントされます。たとえば、次のように列を識別するとします。

```
"schema"."table"."column"
```

スキーマ名、表名、列名の長さはそれぞれ30バイトです。各引用符やピリオドはシングルバイト文字のため、この例での識別子の総バイト長は、最大98バイトになります。

2. 非引用識別子にOracle SQLの予約語は使用できません。引用識別子には、予約語を使用できますが、お薦めしません。

   名前は、データベース・オブジェクトにアクセスするために使用するOracle製品固有のその他の予約語によって、さらに制限されることもあります。

> ノート:
>
> 予約語 ROWID は、この規則の例外です。大文字の ROWID は、引用符の有無にかかわらず、列名としては使用できません。ただし、大文字の単語は列名以外の引用識別子として使用でき、小文字が含まれる単語(たとえば"Rowid"や"rowid")は、列名をはじめとする任意の引用識別子として使用できます。

関連項目:

- Oracle SQLの予約語のすべてのリストは、Oracle SQLの予約語を参照してください。
- 製品の予約語のリストについては、『Oracle Database PL/SQL言語リファレンス』などの各製品のマニュアルを参照してください。

3. OracleのSQL言語には、特別な意味を持つ文字が含まれています。これらの文字には、データ型、スキーマ名、ファンクション名、ダミーのシステム表DUALおよびキーワード(DIMENSION、SEGMENT、ALLOCATE、DISABLEなど、

SQL文中の大文字の単語)が含まれます。これらの文字は予約語ではありません。ただし、Oracleは固有の方法でこれらの文字を内部的に使用します。したがって、これらの文字をオブジェクトおよびオブジェクトの部分の名前として使用した場合、使用しているSQL文が読みにくくなり、予期しない結果になることがあります。

特に、SYS_または0RA_で始まる文字をスキーマ・オブジェクト名として使用しないでください。また、SQL組込みファンクションの名前を、スキーマ・オブジェクトまたはユーザー定義ファンクションの名前として使用しないでください。

関連項目:

- キーワードのリストを取得する方法の詳細は、Oracle SQLキーワードを参照してください。
- データ型、SQLファンクションおよびDUAL表からの選択を参照してください。

4. 異なるプラットフォームおよびオペレーティング・システム間では、ASCIIレパートリからの文字を使用することで最適な互換性を得ることができます。データベース名、グローバル・データベース名、データベース・リンク名には、これらの文字を使用してください。マルチテナント・コンテナ・データベース(CDB)内の共通ユーザー、共通ロールおよび共通プロファイルの名前には、ASCIIレパートリにある文字のみを使用してください。
5. パスワードにマルチバイト文字を含めることができます。
6. 引用符なしの識別子は、データベース文字セットのアルファベット文字で開始する必要があります。引用符付きの識別子は任意の文字で開始できます。
7. 非引用識別子には、データベース文字セットの英数字、アンダースコア(_)、ドル記号($)およびポンド記号(#)のみ含めることができます。データベース・リンクの名前には、ピリオド(.)とアットマーク(@)を含めることもできます。非引用識別子では、$と#はできるだけ使用しないでください。

   引用識別子には、すべての文字、句読点および空白を使用できます。ただし、二重引用符またはnull文字(¥0)は、引用符付きまたは引用符なしのどちらの識別子にも使用できません。
8. ネームスペース内では、2つのオブジェクトに同じ名前を付けることはできません。

   次のスキーマ・オブジェクトは、1つのネームスペースを共有します。

   - パッケージ
   - プライベート・シノニム
   - 順序
   - スタンドアロン・プロシージャ
   - スタンドアロン・ストアド・ファンクション
   - 表
   - ユーザー定義演算子
   - ユーザー定義型
   - ビュー

   次の各スキーマ・オブジェクトは、固有のネームスペースを持ちます。

   - クラスタ
   - 制約

- データベース・トリガー
- ディメンション
- 索引
- マテリアライズド・ビュー(マテリアライズド・ビューを作成すると、データベースで同じ名前の内部表が作成されます。この表は、スキーマ内の他の表と同じネームスペースを持ちます。このため、スキーマに同じ名前の表とマテリアライズド・ビューを含めることはできません。)
- プライベート・データベース・リンク

表および順序が同じネームスペースにあるため、同じスキーマの表および順序が同じ名前を持つことはできません。ただし、表と索引は異なるネームスペースに存在します。このため、同じスキーマ内の表と索引には、同じ名前を付けることができます。

データベース内の各スキーマには、その中のオブジェクトのために固有のネームスペースがあります。たとえば、異なるスキーマ内の2つの表は異なるネームスペースに存在し、同じ名前を付けることができます。

次の各非スキーマ・オブジェクトは、固有のネームスペースを持ちます。

- エディション
- パラメータ・ファイル(PFILE)およびサーバー・パラメータ・ファイル(SPFILE)
- プロファイル
- パブリック・データベース・リンク
- パブリック・シノニム
- 表領域
- ユーザー・ロール

これらのネームスペース内のオブジェクトはスキーマに含まれないため、これらのネームスペースはデータベース全体で使用されます。

9. 引用符なしの識別子では大文字と小文字が区別されません。引用符のない識別子は大文字として解析されます。引用符付きの識別子では大文字と小文字が区別されます。

   二重引用符で名前を囲むことによって、次の名前を同じネームスペースの異なるオブジェクトに割り当てることができます。

```
"employees"
"Employees"
"EMPLOYEES"
```

   ただし、Oracleは次の名前を同じ名前として解析するため、同じネームスペース内の異なるオブジェクトには、次の名前を使用できません。

```
employees
EMPLOYEES
"EMPLOYEES"
```

10. Oracleが識別子を大文字で格納または比較する場合、識別子の各文字の大文字形式は、データベース文字セットの大文字化規則を適用することによって決定されます。セッション設定NLS_SORTによって決定される言語固有の規則は考慮されません。この動作は、ファンクションNLS_UPPERではなく、SQLファンクションUPPERを識別子

に適用することに対応しています。

データベース文字セットの大文字化規則によって、特定の自然言語としては不適切な結果となる場合があります。たとえば、データベース文字セットの大文字化規則に従うと、ドイツ語で使用される小文字のシャープs("ß")に大文字形式はありません。この文字は、識別子が大文字に変換されても変更されませんが、期待されるドイツ語での大文字形式は、2つの大文字のSの連続("SS")です。同様に、小文字iの大文字形式は、データベース文字セットの大文字化規則に従うと大文字Iです。ただし、トルコ語およびアゼルバイジャン語で予期される大文字形式は、上に点が付いた大文字Iです。

データベース文字セットの大文字化規則では、識別子はセッションのすべての言語構成で同様に解釈されます。識別子が特定の自然言語で正しく表示されるようにする場合は、引用符で囲んで小文字形式を保持するか、またはその識別子を使用するときは常に言語的に正しい大文字形式を使用することができます。

11. 同じ表やビューでは、複数の列に同じ名前を付けることはできません。ただし、異なる表やビューでは、複数の列に同じ名前を付けることができます。

12. 引数の数およびデータ型が異なる場合、同じパッケージに含まれるプロシージャやファンクションに同じ名前を付けることができます。異なる引数を持ち、同じ名前のプロシージャやファンクションを同じパッケージ内に複数作成することを、オーバーロードといいます。

13. 表領域名は、30バイトに制限されている他の識別子とは異なり、大/小文字が区別されます。

スキーマ・オブジェクトのネーミング例

次に、有効なスキーマ・オブジェクト名の例を示します。

```
last_name
horse
hr.hire_date
"EVEN THIS & THAT!"
a_very_long_and_valid_name
```

これらのすべての例は、データベース・オブジェクトのネーミング規則に示す規則に従っています。

スキーマ・オブジェクトのネーミングのガイドライン

次に、オブジェクトおよび各部分の命名に役立つガイドラインを示します。

- 完全な、わかりやすい、発音可能な名前(またはよく知られた略語)を使用します。
- 一貫性のあるネーミング・ルールを使用します。
- 複数の表にまたがる同一のエンティティや属性を記述するためには、同一の名前を使用します。

オブジェクトに名前を付けるときは、名前を短く、使いやすくするという目標と、できるだけわかりやすい名前にするという目標のバランスをとる必要があります。迷ったときは、よりわかりやすい名前を選んでください(データベース内のオブジェクトは、長年の間に多数の人に使用される可能性があるため)。表の列に、`payment_due_date`ではなく`pmdd`のような名前を付けると、10年後のデータベース管理者が見たときに理解できない可能性があります。

一貫したネーミング規則を使用すると、アプリケーション上の各表の働きが理解しやすくなります。そのような規則の例として、`FINANCE`アプリケーションに属している表の名前をすべて`fin_`で始めるような場合が考えられます。

同一のエンティティや属性に対しては、複数の表にまたがっていても同じ名前を使用してください。たとえば、`employees`サンプル表と`departments`サンプル表の部門番号列には、どちらにも`department_id`という名前を付けます。

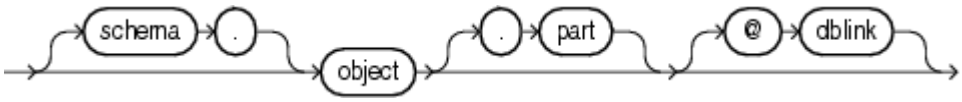
スキーマ・オブジェクトの構文およびSQL文の構成要素

SQL文のコンテキストでスキーマ・オブジェクトとそれらの部分を参照する方法について説明します。次の項目について説明します。

- オブジェクトを参照するための一般的な構文
- Oracleがオブジェクトへの参照を変換する方法
- 自分のスキーマ以外のスキーマ内のオブジェクトを参照する方法
- リモート・データベース内のオブジェクトを参照する方法
- 表と索引のパーティションおよびサブパーティションを参照する方法

次に、オブジェクトやそれらの部分を参照するための一般的な構文を示します。

database_object_or_part::=



(dblink::=)

説明:

- `object`は、オブジェクトの名前です。
- `schema`は、オブジェクトを含むスキーマです。この修飾子を指定することによって、自分のスキーマ以外のスキーマ内のオブジェクトを参照できます。その場合には、自分のスキーマ以外のスキーマ内のオブジェクトを参照するための権限が必要です。この修飾子を指定しないと、自分自身のスキーマ内のオブジェクトを参照するものとみなされます。

  スキーマ・オブジェクトのみが`schema`で修飾できます。スキーマ・オブジェクトについては、規則8を参照してください。規則8に示す非スキーマ・オブジェクトはスキーマ・オブジェクトではないため、`schema`では修飾できません。ただし、パブリック・シノニムは例外で、「`PUBLIC`」で修飾できます。この場合、引用符が必要です。
- `part`は、オブジェクトの部分です。この識別子によって、スキーマ・オブジェクトの部分(たとえば、表の列またはパーティション)を参照できます。なお、すべてのタイプのオブジェクトが部分を持っているとはかぎりません。
- `dblink`は、Oracle Databaseの分散オプションを使用している場合にのみ適用されます。オブジェクトを含むデータベースの名前です。この修飾子`dblink`を指定することによって、ローカル・データベース以外のデータベース内のオブジェクトを参照できます。この`dblink`を指定しないと、自分自身のローカル・データベース内のオブジェクトを参照するものとみなされます。なお、すべてのSQL文でリモート・データベースのオブジェクトにアクセスできるとはかぎりません。

オブジェクトを参照する際のコンポーネントを区切っているピリオドの前後には、空白を入れることができます。ただし、通常は入れません。

Oracle Databaseによるスキーマ・オブジェクト参照の変換方法

SQL文内のオブジェクトが参照される場合、OracleはそのSQL文のコンテキストを検討して、該当するネームスペース内でそのオブジェクトの位置を確認します。そのオブジェクトの位置を確認してから、そのオブジェクトに対して文が指定する操作を実行します。指定した名前のオブジェクトが適切なネームスペース内に存在しない場合、Oracleはエラーを戻します。

次の例で、OracleがSQL文内のオブジェクト参照を変換する方法について説明します。名前`departments`で識別される

表にデータ行を追加する次の文を考えます。

```
INSERT INTO departments
  VALUES (280, 'ENTERTAINMENT_CLERK', 206, 1700);
```

文のコンテキストに基づいて、Oracleは、departmentsが次のようなオブジェクトであると判断します。

- 自分のスキーマ内の表
- 自分のスキーマ内のビュー
- 表またはビューに対するプライベート・シノニム
- パブリック・シノニム

Oracleは、文を発行したユーザーのスキーマ外のネームスペースを考慮する前に、そのユーザーのスキーマ内のネームスペースからオブジェクト参照を変換しようとします。この例では、Oracleは次の方法で名前departmentsを変換しようとします。

1. Oracleは、最初に、表、ビューおよびプライベート・シノニムを含む文を発行したユーザーのスキーマ内のネームスペースで、オブジェクトの位置を確認しようとします。そのオブジェクトがプライベート・シノニムである場合は、Oracleはそのシノニムが表すオブジェクトの位置を確認します。このオブジェクトは、ユーザーのスキーマに存在することもあれば、別のスキーマまたは別のデータベースに存在することもあります。このオブジェクトが別のシノニムである場合もあります。その場合、Oracleはそのシノニムが表すオブジェクトの位置を確認します。
2. オブジェクトがネームスペース内に存在する場合、Oracleはそのオブジェクトに対して文を実行しようとします。この例では、Oracleはデータ行をdepartmentsに追加しようとします。オブジェクトがその処理にとって正しい型でない場合、Oracleはエラーを戻します。この場合、departmentsは、表またはビュー、あるいは表またはビューとなるプライベート・シノニムである必要があります。departmentsが順序である場合、Oracleはエラーを戻します。
3. 前述の処理で検索されたネームスペースにオブジェクトが存在しない場合、Oracleはパブリック・シノニムを含むネームスペースを検索します。オブジェクトがそのネームスペースに存在する場合、Oracleはそのオブジェクトに対して文を実行しようとします。オブジェクトがその処理にとって正しい型でない場合、Oracleはエラーを戻します。この例では、departmentsが順序のパブリック・シノニムである場合、Oracleはエラーを戻します。

パブリック・シノニムに、依存表またはユーザー定義型がある場合は、依存オブジェクトと同じスキーマに、シノニムと同じ名前でオブジェクトを作成することはできません。

シノニムに、依存表またはユーザー定義型がない場合は、依存オブジェクトと同じスキーマに、依存オブジェクトと同じ名前で、オブジェクトを作成できます。すべての依存オブジェクトが無効になり、次のアクセス時に妥当性チェックが再実行されます。

関連項目:

識別子名を解決するPL/SQLの詳細は、Oracle Database PL/SQL言語リファレンスを参照

他のスキーマ内のオブジェクトの参照

自分が所有するスキーマ以外のスキーマ内のオブジェクトを参照するには、次のように、オブジェクト名の前にスキーマ名を付けます。

```
schema.object
```

たとえば、次の文は、サンプル・スキーマhr内のemployees表を削除します。

```
DROP TABLE hr.employees;
```

リモート・データベース内のオブジェクトの参照

ローカル・データベース以外のデータベース内のオブジェクトを参照するには、オブジェクト名の後に、そのデータベースへのデータベース・リンクの名前を続けます。データベース・リンクはスキーマ・オブジェクトであり、これによってOracleがリモート・データベースに接続され、そこにあるオブジェクトにアクセスします。この項では、次の項目について説明します。

- データベース・リンクを作成する方法
- SQL文でデータベース・リンクを使用する方法

データベース・リンクの作成

「CREATE DATABASE LINK」を使用して、データベース・リンクを作成します。この文では、データベース・リンクに関する次の情報を指定できます。

- データベース・リンク名
- リモート・データベースにアクセスするためのデータベース接続文字列
- リモート・データベースに接続するためのユーザー名およびパスワード

これらの情報はデータ・ディクショナリに格納されます。

データベース・リンク名

データベース・リンクを作成するとき、データベース・リンク名を指定する必要があります。データベース・リンク名は、他のオブジェクト型の名前とは異なります。データベース・リンク名は128バイト以内の長さで指定し、ピリオド(.)とアットマーク(@)を使用できます。

データベース・リンクに付ける名前は、データベース・リンクが参照するデータベースの名前、およびデータベース名の階層内のそのデータベースの位置に一致している必要があります。次に、データベース・リンク名の書式を示します。

dblink::=

説明:

- `database`には、データベース・リンクの接続先であるリモート・データベースのグローバル名のうち、`name`の部分を指定します。このグローバル名は、リモート・データベースのデータ・ディクショナリに格納されます。この名前は、`GLOBAL_NAME`データ・ディクショナリ・ビューで確認できます。
- `domain`には、データベース・リンクの接続先であるリモート・データベースのグローバル名のうち、`domain`の部分を指定します。データベース・リンクの名前に`domain`を指定しないと、Oracleは、現在、データ・ディクショナリに存在しているローカル・データベースのドメインに、データベース・リンク名を付加します。
- `connection_qualifier`によって、データベース・リンクをさらに修飾できます。接続修飾子を使用する場合、同じデータベースに複数のデータベース・リンクを作成できます。たとえば、接続修飾子を使用して、同じデータベースにアクセスするOracle Real Application Clustersの異なるインスタンスに、複数のデータベース・リンクを作成できます。

関連項目:

接続修飾子の詳細は、『Oracle Database管理者ガイド』を参照してください。

database.domainの組合せは、サービス名と呼ばれることもあります。

関連項目:

Oracle Database Net Services管理者ガイド

ユーザー名およびパスワード

リモート・データベースに接続するために、ユーザー名およびパスワードを使用します。データベース・リンクでは、ユーザー名およびパスワードはオプションです。

データベース接続文字列

データベース接続文字列は、Oracle Netがリモート・データベースにアクセスするために使用する仕様です。データベース接続文字列の記述方法については、使用しているネットワーク・プロトコル用のOracle Netのドキュメントを参照してください。データベース・リンクのデータベース接続文字列はオプションです。

データベース・リンクの参照

データベース・リンクは、分散オプションを指定してOracleを使用している場合にのみ利用できます。データベース・リンクを含むSQL文の発行時に、次のいずれかの方法でデータベース・リンク名を指定します。

- データ・ディクショナリ内に格納される、database、domain、およびオプションのconnection_qualifierコンポーネントを含む完全なデータベース・リンク名を指定します。
- databaseとオプションのconnection_qualifierが含まれ、domainは含まれない、部分的データベース・リンク名を指定します。

Oracleは、リモート・データベースに接続する前に次のタスクを実行します。

1. 文中に指定されているデータベース・リンク名が部分指定の場合、Oracleは、データ・ディクショナリ内に格納されているグローバル・データベース名に見られるとおり、そのリンク名にローカル・データベースのドメイン名を付加します。現在のグローバル・データベース名は、GLOBAL_NAMEデータ・ディクショナリ・ビューで見ることができます。
2. Oracleは、最初に、文を発行したユーザーのスキーマ内で、文の中のデータベース・リンクと同じ名前を持つプライベート・データベース・リンクを検索します。必要に応じて、同じ名前を持つパブリック・データベース・リンクを検索します。
    - Oracleは、必ず最初に一致したデータベース・リンク(プライベートまたはパブリック)のユーザー名およびパスワードを採用します。最初に一致したデータベース・リンクに対応付けられているユーザー名およびパスワードがあると、Oracleはそれを使用します。対応付けられているユーザー名およびパスワードがない場合、Oracleは、現在のユーザー名およびパスワードを使用します。
    - 最初に一致したデータベース・リンクに対応付けられているデータベース文字列が存在する場合、Oracleは、そのデータベース文字列を使用します。データベース文字列がない場合、Oracleは一致する次の(パブリック)データベース・リンクを検索します。一致するデータベース・リンクが存在しない場合、または一致するリンクに対応付けられているデータベース文字列が存在しない場合、Oracleはエラーを戻します。
3. Oracleは、リモート・データベースにアクセスするためにデータベース文字列を使用します。リモート・データベースにアクセスした後で、GLOBAL_NAMESパラメータの値がtrueの場合は、Oracleは、データベース・リンク名の

database.domain部分がリモート・データベースの完全なグローバル名に一致しているかどうかを確認します。この条件が満たされている場合、Oracleはステップ2で選択したユーザー名とパスワードを使用して接続を続行します。それ以外の場合、Oracleはエラーを戻します。

4. データベース文字列、ユーザー名およびパスワードを使用した接続が成功した場合、Oracleは、リモート・データベース上の指定されたオブジェクトにアクセスしようとします。このとき、この項の前半で説明した、オブジェクト参照を変換するための規則、および他のスキーマ内のオブジェクトを参照するための規則が使用されます。

リモート・データベースの完全なグローバル名が、データベース・リンクのdatabase.domain部分と一致する必要があるという要件を無効にするには、初期化パラメータGLOBAL_NAMESか、ALTER SYSTEMまたはALTER SESSION文のGLOBAL_NAMESパラメータにFALSEを設定します。

関連項目:

リモート・データベースの名前の変換の詳細は、『Oracle Database管理者ガイド』を参照してください。

パーティション表と索引の参照

表および索引はパーティション化できます。パーティション化されたスキーマ・オブジェクトは、パーティションと呼ばれる多数の部分で構成され、各パーティションのすべての論理属性は同じです。たとえば、表のパーティションはすべて同じ列定義と制約定義を共有し、索引のパーティションはすべて同じ索引列を共有します。

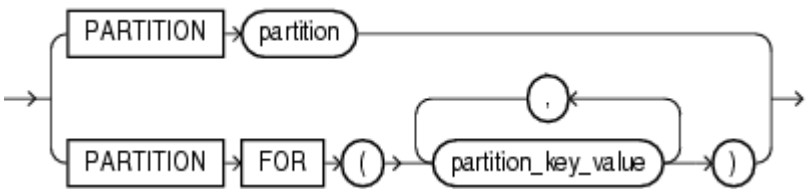
拡張パーティション名および拡張サブパーティション名を使用すると、パーティション・レベルやサブパーティション・レベルの操作(たとえば特定のパーティションまたはサブパーティションのすべての行の削除)を、1つのパーティションまたはサブパーティションのみに対して実行できます。拡張名を使用しないでそのような操作を実行する場合は、述語(WHERE句)の指定が必要になります。レンジ/リスト・パーティション表の場合は、パーティション・レベルの操作を述語で表現しようとすると、非常に複雑になることがあります(特に、レンジ・パーティション・キーで複数の列が使用されているとき)。ハッシュ・パーティション/サブパーティションの場合は、述語を使用するとさらに複雑になります(このようなパーティション/サブパーティションはシステム定義のハッシュ・ファンクションに基づいているため)。

拡張パーティション名を使用すると、パーティションを表と同じように扱うことができます。この方法の利点(レンジ・パーティション表に対して最も有益)は、これらのビューに対する権限を他のユーザーやロールに付与する(または権限を取り消す)ことによってパーティション・レベルのアクセス制御メカニズムを構築できることです。パーティションを表として使用するには、1つのパーティションからデータを選択してビューを作成し、このビューを表として使用します。

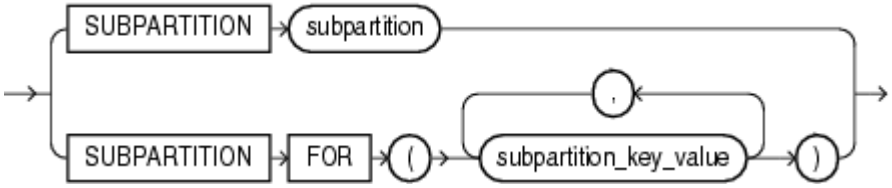
構文

構文にpartition_extended_name要素またはsubpartition_extended_name要素が指定されているSQL文には、拡張パーティション表名および拡張サブパーティション表名を指定できます。

partition_extended_name::=

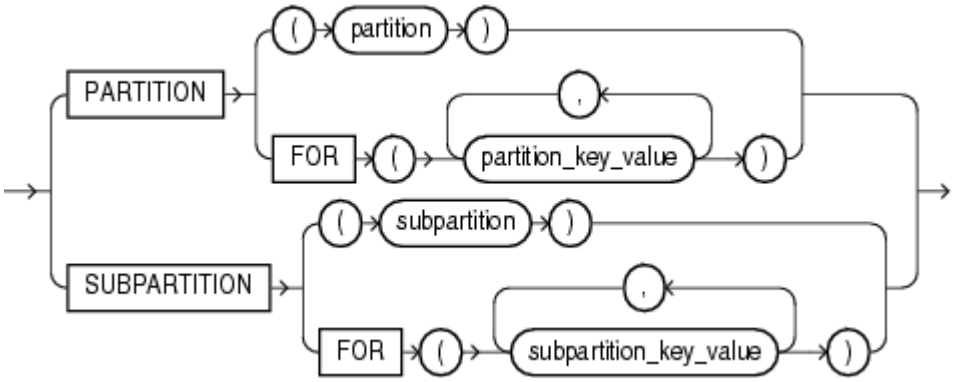


subpartition_extended_name::=



DML文INSERT、UPDATE、DELETE、およびANALYZE文では、パーティション名またはサブパーティション名をカッコで囲む必要があります。細かな違いですが、partition_extension_clauseに影響します。

partition_extension_clause::=



partition_extended_name、subpartition_extended_nameおよびpartition_extension_clauseでは、PARTITION FOR句およびSUBPARTITION FOR句を使用すると、名前がなくてもパーティションを参照できます。これらの句は、すべてのタイプのパーティション化に有効であり、特に時間隔パーティションで役立ちます。データが表に挿入されると、必要に応じて時間隔パーティションが自動的に作成されます。

partition_key_valueおよびsubpartition_key_valueには、パーティション化キー列1つにつき1つの値を指定します。複数列のパーティション化キーの場合は、パーティション化キーごとに1つの値を指定します。コンポジット・パーティションの場合は、パーティション化キーごとに1つの値を指定し、その後にサブパーティション化キーごとに1つの値を指定します。パーティション化キーの値はすべて、カンマで区切ります。時間隔パーティションには、1つのpartition_key_valueのみを指定できます。この値は、有効なNUMBERまたは日時値にする必要があります。指定した値が格納されているパーティションまたはサブパーティションが、SQL文による操作の対象となります。

関連項目:

時間隔パーティションの詳細は、CREATE TABLEのINTERVAL句を参照してください。

拡張名の制限事項

現在、拡張パーティション表名および拡張サブパーティション表名の使用には、次の制限があります。

- リモート表は使用できません。拡張パーティション表名または拡張サブパーティション表名には、データベース・リンク(dblink)、またはdblinkを使用して表に変換するシノニムを含めることはできません。リモートのパーティションやサブパーティションを使用するには、リモート・サイトで拡張表名の構文を使用してビューを作成し、そのリモート・ビューを参照します。
- シノニムは使用できません。拡張パーティションまたは拡張サブパーティションは、実表を使用して指定する必要があります。シノニムやビューなど、他のオブジェクトは使用できません。
- PARTITION FOR句およびSUBPARTITION FOR句は、ビューでのDDL操作に対しては無効になります。

- PARTITION FOR句およびSUBPARTITION FOR句では、キーワードDEFAULTまたはMAXVALUE、あるいはpartition_key_valueまたはsubpartition_key_valueのバインド変数を指定できません。
- PARTITIONおよびSUBPARTITION句では、パーティション名またはサブパーティション名にバインド変数を指定できません。

例

次の文のsalesは、パーティションsales_q1_2000を持つパーティション表です。単一パーティションsales_q1_2000のビューを作成でき、それを表のように使用できます。この例では、パーティションから行が削除されます。

```
CREATE VIEW Q1_2000_sales AS
  SELECT *
    FROM sales PARTITION (SALES_Q1_2000);
DELETE FROM Q1_2000_sales
  WHERE amount_sold < 0;
```

オブジェクト型の属性とメソッドの参照

SQL文のオブジェクト型の属性とメソッドを参照するには、参照を表の別名で完全に修飾する必要があります。次の例では、cust_address_typ型、およびcust_address_typに基づくcust_address列を持つ表customersを含むサンプル・スキーマoeについて考えます。

```
CREATE TYPE cust_address_typ
  OID '82A4AF6A4CD1656DE034080020E0EE3D'
  AS OBJECT
    (street_address    VARCHAR2(40),
     postal_code       VARCHAR2(10),
     city              VARCHAR2(30),
     state_province    VARCHAR2(10),
     country_id        CHAR(2));
/
CREATE TABLE customers
  (customer_id        NUMBER(6),
   cust_first_name    VARCHAR2(20) CONSTRAINT cust_fname_nn NOT NULL,
   cust_last_name     VARCHAR2(20) CONSTRAINT cust_lname_nn NOT NULL,
   cust_address       cust_address_typ,
. . .
```

次に示すとおり、SQL文では、postal_code属性への参照は表別名を使用して完全に修飾する必要があります。

```
SELECT c.cust_address.postal_code
  FROM customers c;
UPDATE customers c
  SET c.cust_address.postal_code = '14621-2604'
  WHERE c.cust_address.city = 'Rochester'
    AND c.cust_address.state_province = 'NY';
```

引数を取らないメンバー・メソッドを参照する場合は、空のカッコを付ける必要があります。たとえば、サンプル・スキーマoeには、メンバー・ファンクションgetCatalogNameを含むcatalog_typに基づくオブジェクト表categories_tabが含まれます。SQL文でこのメソッドをコールするには、次の例のように、空のカッコを付ける必要があります。

```
SELECT TREAT(VALUE(c) AS catalog_typ).getCatalogName() "Catalog Type"
  FROM categories_tab c
  WHERE category_id = 90;
Catalog Type
-----------------------------------
online catalog
```

3 疑似列

疑似列は表の列のように使用できますが、実際に表に格納されているわけではありません。疑似列から値を選択できますが、疑似列に対して値の挿入、更新、削除はできません。疑似列は、引数を指定しないファンクションとも似ています(「ファンクション」を参照)。ただし、引数を指定しないファンクションは、通常、結果セット内の各行に対して同じ値を戻しますが、疑似列では各行に対して異なる値を戻します。

この章の構成は、次のとおりです。

階層問合せ疑似列

階層問合せ疑似列は、階層問合せでのみ有効です。階層問合せ疑似列には、次のものがあります。

- CONNECT_BY_ISCYCLE疑似列
- CONNECT_BY_ISLEAF疑似列
- LEVEL疑似列

問合せの中で階層型の関連を定義するには、CONNECT BY句を使用する必要があります。

CONNECT_BY_ISCYCLE疑似列

CONNECT_BY_ISCYCLE疑似列は、現在の行に自身の祖先でもある子がある場合に1を戻します。それ以外の場合は、0を戻します。

CONNECT BY句のNOCYCLEパラメータを指定した場合のみ、CONNECT_BY_ISCYCLEを指定できます。NOCYCLEによって、Oracleは問合せの結果を戻すことができます。このパラメータを指定しないと、データ内のCONNECT BYループのため、問合せは失敗します。

関連項目:

NOCYCLEパラメータの詳細は、階層問合せを参照してください。CONNECT_BY_ISCYCLE疑似列を使用する例は、階層問合せの例を参照してください。

CONNECT_BY_ISLEAF疑似列

CONNECT_BY_ISLEAF疑似列は、現在の行がCONNECT BY条件によって定義されるツリーのリーフである場合に1を戻します。それ以外の場合は、0を戻します。この情報は、特定の行をさらに展開して階層の詳細を表示できるかどうかを示します。

CONNECT_BY_ISLEAFの例

次の例は、hr.employees表の最初の3レベルです。各行がリーフ行か(IsLeaf列が1)、子である行を持つか(IsLeaf列が0)を示しています。

```
SELECT last_name "Employee", CONNECT_BY_ISLEAF "IsLeaf",
       LEVEL, SYS_CONNECT_BY_PATH(last_name, '/') "Path"
  FROM employees
  WHERE LEVEL <= 3 AND department_id = 80
  START WITH employee_id = 100
  CONNECT BY PRIOR employee_id = manager_id AND LEVEL <= 4
  ORDER BY "Employee", "IsLeaf";
Employee                  IsLeaf      LEVEL Path
------------------------- ---------- ---------- -------------------------
Abel                               1          3 /King/Zlotkey/Abel
Ande                               1          3 /King/Errazuriz/Ande
Banda                              1          3 /King/Errazuriz/Banda
Bates                              1          3 /King/Cambrault/Bates
Bernstein                          1          3 /King/Russell/Bernstein
Bloom                              1          3 /King/Cambrault/Bloom
Cambrault                          0          2 /King/Cambrault
Cambrault                          1          3 /King/Russell/Cambrault
Doran                              1          3 /King/Partners/Doran
Errazuriz                          0          2 /King/Errazuriz
Fox                                1          3 /King/Cambrault/Fox
```

. . .

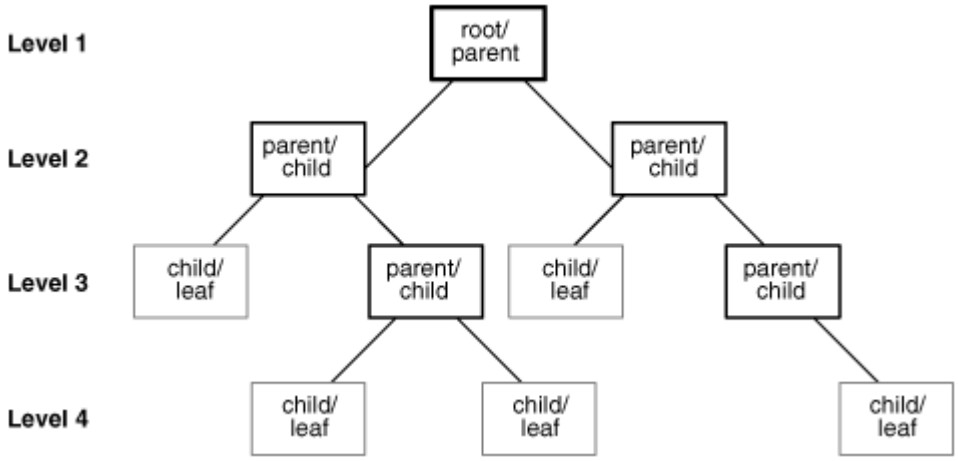
関連項目:

階層問合せおよびSYS_CONNECT_BY_PATHを参照してください。

LEVEL疑似列

階層問合せによって戻される各行について、LEVEL疑似列は、ルート行に1を戻し、ルートの子には2を戻します(以降同様に続きます)。ルート行は逆ツリー構造の最上位行です。子である行は任意の非ルート行です。親である行は子を持つ任意の行です。リーフ行は子を持たない任意の行です。図3-1に、逆ツリーのノードとそれらのLEVEL値を示します。

図3-1 階層ツリー



関連項目:

階層問合せの概要は、階層問合せを参照してください。LEVEL疑似列の使用に関する制限事項は、IN条件を参照してください。

順序疑似列

順序は、一意の連続値を生成できるスキーマ・オブジェクトです。これらの値は、主キーや一意のキーによく使用されます。次の疑似列を使用したSQL文で、順序値を参照できます。

- CURRVAL: 順序の現在の値を戻します。
- NEXTVAL: 順序を増加させて次の値を戻します。

CURRVALとNEXTVALは、順序の名前で修飾する必要があります。

```
sequence.CURRVAL
sequence.NEXTVAL
```

別のユーザーのスキーマ内での順序の現在の値または次の値を参照するには、その順序に対するSELECTオブジェクト権限またはSELECT ANY SEQUENCEシステム権限のどちらかが必要です。さらに、その順序は、次に示すとおり、順序を含むスキーマで修飾する必要があります。

```
schema.sequence.CURRVAL
schema.sequence.NEXTVAL
```

リモート・データベース上の順序の値を参照するには、次のように、データベース・リンクの完全な名前または部分的な名前で順序を修飾する必要があります。

```
schema.sequence.CURRVAL@dblink
schema.sequence.NEXTVAL@dblink
```

順序には、待機またはロックすることなく多数のユーザーが同時にアクセスできます。

関連項目:

データベース・リンクの参照方法の詳細は、「リモート・データベース内のオブジェクトの参照」を参照してください。

順序値の使用場所

次の場所でCURRVALとNEXTVALを使用できます。

- 副問合せ、マテリアライズド・ビューまたはビューに含まれていないSELECT文のSELECT構文のリスト
- INSERT文内の副問合せのSELECT構文のリスト
- INSERT文のVALUES句
- UPDATE文のSET句

順序値の制限事項

次の要素では、CURRVALとNEXTVALは使用できません。

- DELETE文、SELECT文またはUPDATE文内の副問合せ
- ビューの問合せ、またはマテリアライズド・ビューの問合せ
- DISTINCT演算子を指定したSELECT文
- GROUP BY句またはORDER BY句を持つSELECT文

- 集合演算子UNION、INTERSECTまたはMINUSによって別のSELECT文と結合されているSELECT文
- SELECT文のWHERE句
- CHECK制約の条件

CURRVALまたはNEXTVALを使用する単一のSQL文では、参照されたLONG列、更新された表、ロックされた表がすべて同じデータベース上にある必要があります。

順序値の使用方法

順序を作成するときに、初期値と増分値を定義できます。NEXTVALの最初の参照によって、順序の初期値が戻されます。その後の参照によって、定義されたNEXTVAL増分値で順序が増加され、その新しい値が戻されます。CURRVALを参照すると、NEXTVALへの最後の参照で戻された値である、順序の現在の値が常に戻されます。

セッションの順序に対してCURRVALを使用する前に、まずNEXTVALで順序を初期化してください。順序の詳細は、「CREATE SEQUENCE」を参照してください。

NEXTVALへの参照が含まれる単一のSQL文の中では、Oracleは、次の各行につき1回順序を増加させます。

- SELECT文の外部問合せブロックによって戻される行。このような問合せブロックは、次の場所に指定できます。
  - トップレベルのSELECT文
  - INSERT ... SELECT文(単一表または複数表)。マルチテーブル・インサートの場合、NEXTVALへの参照がVALUES句内に存在する必要があり、マルチテーブル・インサートの複数のブランチでNEXTVALが参照される場合でも、副問合せによって戻される行ごとに、順序が1回更新されます。
  - CREATE TABLE ... AS SELECT文
  - CREATE MATERIALIZED VIEW ... AS SELECT文
- UPDATE文で更新される行
- VALUES句が含まれるINSERT文
- INSERT ... [ALL | FIRST]文(マルチテーブル・インサート)。マルチテーブル・インサートは、単一のSQL文とみなされます。このため、順序のNEXTVALへの参照では、文のSELECT部分からの各入力レコードに対して順序が1回のみ増加されます。INSERT ... [ALL | FIRST]文のいずれかの部分でNEXTVALが複数回指定されている場合は、指定されたレコードが挿入される回数に関係なく、値はすべての挿入ブランチに対して同じになります。
- MERGE文でマージされる行。NEXTVALへの参照は、merge_insert_clauseまたはmerge_update_clauseあるいはその両方に指定できます。NEXTVALUE値は、更新操作または挿入操作に順序番号が使用されない場合でも、行が更新されるか挿入されるたびに増加されます。NEXTVALがこれらの場所のいずれかで複数回指定されている場合、順序は各行に対して1回増加され、その行のNEXTVALが検出されるたびにすべて同じ値を戻します。
- マルチテーブルINSERT ALL文の入力行。NEXTVALは、各行に対するinsert_into_clauseマップの回数に関係なく、副問合せによって戻される各行に対して1回のみ増加されます。

これらの場所のいずれかが、NEXTVALを複数回参照している場合、Oracleは1回のみ順序を増加させ、NEXTVALが検出されるたびにすべて同じ値を戻します。

これらの場所のいずれかが、CURRVALとNEXTVALの両方を参照している場合、OracleはCURRVALとNEXTVALの両方について順序を増加させ同じ値を戻します。

順序の次の値の確認: 例

この例では、サンプル・スキーマhrの従業員順序の次の値を選択します。

```
SELECT employees_seq.nextval
  FROM DUAL;
```

表への順序値の挿入: 例

次の例では、従業員順序を増加させ、サンプル表hr.employeesに挿入される新しい従業員のためにその値を使用します。

```
INSERT INTO employees
  VALUES (employees_seq.nextval, 'John', 'Doe', 'jdoe', '555-1212',
          TO_DATE(SYSDATE), 'PU_CLERK', 2500, null, null, 30);
```

順序の現在の値の再利用: 例

この例では、次の順序番号を持つ新しい順序をマスター順序表に追加します。その後、この番号を使用して関連する注文をディテール注文表に追加します。

```
INSERT INTO orders (order_id, order_date, customer_id)
  VALUES (orders_seq.nextval, TO_DATE(SYSDATE), 106);
INSERT INTO order_items (order_id, line_item_id, product_id)
  VALUES (orders_seq.currval, 1, 2359);
INSERT INTO order_items (order_id, line_item_id, product_id)
  VALUES (orders_seq.currval, 2, 3290);
INSERT INTO order_items (order_id, line_item_id, product_id)
  VALUES (orders_seq.currval, 3, 2381);
```

バージョン問合せ疑似列

バージョン問合せ疑似列は、Oracleフラッシュバック問合せの一形態であるOracle Flashback Version Queryでのみ有効です。バージョン問合せ疑似列には、次のものがあります。

- VERSIONS_STARTSCNおよびVERSIONS_STARTTIME: 行バージョンが作成されたときの最初のシステム変更番号(SCN)またはTIMESTAMP。この疑似列によって、データの値が最初に行バージョンに反映された時間が識別されます。この疑似列を使用して、Oracle Flashback TableまたはOracle Flashback Queryにおける過去のターゲット時間を指定できます。この疑似列がNULLの場合、行バージョンは開始前に作成されました。
- VERSIONS_ENDSCNおよびVERSIONS_ENDTIME: 行バージョンの期限が切れたときのSCNまたはTIMESTAMP。疑似列がNULLの場合、行バージョンが問合せの時点で現行のものであったか、行がDELETE操作に対応しているかのいずれかです。
- VERSIONS_XID: 行バージョンが作成されたトランザクションのID(RAW番号)。
- VERSIONS_OPERATION: トランザクションで実行された操作。挿入の場合はI、削除の場合はD、更新の場合はUです。バージョンは、挿入、削除または更新された行のバージョンです。つまり、INSERT操作後の行、DELETE操作前の行、またはUPDATE操作の影響を受ける行です。

  ユーザーによる索引キーの更新の場合、Oracle Flashback Version Queryでは、UPDATE操作がDELETEおよびINSERTの2つの操作として処理される場合があり、Dの後にIのVERSIONS_OPERATIONが続く、2つのバージョンの行で表されます。

関連項目:

- バージョン問合せの詳細は、「flashback_query_clause」を参照してください。
- Oracle Flashback Version Queryの詳細な使用方法については、『Oracle Database開発ガイド』を参照してください。
- VERSIONS_OPERATION疑似列の値の照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

COLUMN_VALUE疑似列

COLUMNS句を指定せずにXMLTable構造体を参照する場合、またはTABLEコレクション式を使用して表の型をネストしたスカラーを参照する場合、データベースは単一列を持つ仮想表を戻します。この疑似列の名前は、COLUMN_VALUEです。

XMLTableのコンテキストでは、戻り値のデータ型はXMLTypeです。たとえば、次の2つの文は同等で、どちらの出力も戻される列の名前としてCOLUMN_VALUEを示します。

```
SELECT *
  FROM XMLTABLE('<a>123</a>');
COLUMN_VALUE
----------------------------------------
<a>123</a>
SELECT COLUMN_VALUE
  FROM (XMLTable('<a>123</a>'));
COLUMN_VALUE
----------------------------------------
<a>123</a>
```

TABLEコレクション式のコンテキストでは、戻り値はコレクション要素のデータ型になります。次の文は、2つのレベルにネストした表を作成します。このコンテキストでのCOLUMN_VALUEの使用方法は、表の作成: マルチレベル・コレクションの例を参照してください。

```
CREATE TYPE phone AS TABLE OF NUMBER;
/
CREATE TYPE phone_list AS TABLE OF phone;
/
```

次の文ではCOLUMN_VALUEを使用してphone型から選択します。

```
SELECT t.COLUMN_VALUE
  FROM TABLE(phone(1,2,3)) t;
COLUMN_VALUE
------------
           1
           2
           3
```

ネストした型では、SELECT構文のリストとTABLEコレクション式の両方でCOLUMN_VALUE疑似列を使用できます。

```
SELECT t.COLUMN_VALUE
  FROM TABLE(phone_list(phone(1,2,3))) p, TABLE(p.COLUMN_VALUE) t;
COLUMN_VALUE
------------
           1
           2
           3
```

次の例に示すように、キーワードCOLUMN_VALUEは、列または属性名を持たない内部のネストした表のスカラー値に対してOracle Databaseが生成する名前です。このコンテキストでは、COLUMN_VALUEは疑似列ではなく、実際の列の名前です。

```
CREATE TABLE my_customers (
    cust_id       NUMBER,
    name          VARCHAR2(25),
    phone_numbers phone_list,
    credit_limit  NUMBER)
  NESTED TABLE phone_numbers STORE AS outer_ntab
  (NESTED TABLE COLUMN_VALUE STORE AS inner_ntab);
```

関連項目:

- このファンクションの詳細は、「XMLTABLE」を参照してください。
- TABLEコレクション式の詳細は、「table_collection_expression::=」を参照してください。
- ALTER TABLEの例は、「ネストした表: 例」を参照してください。
- COLUMN_VALUE疑似列の値の照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

OBJECT_ID疑似列

OBJECT_ID疑似列は、オブジェクト表またはビューの列のオブジェクト識別子を戻します。Oracleはこの疑似列をオブジェクト表の主キーとして使用します。OBJECT_IDは、ビューのINSTEAD OFトリガーや、オブジェクト表の置換可能行のIDの識別に便利です。

ノート:

以前のリリースでは、この疑似列は SYS_NC_OID$と呼ばれていました。下位互換性を保つため、この名称は引き続きサポートされます。ただし、より直感的な名前である OBJECT_ID を使用することをお薦めします。

関連項目:

この疑似列の使用例については、『Oracle Databaseオブジェクト・リレーショナル開発者ガイド』を参照してください。

OBJECT_VALUE疑似列

OBJECT_VALUE疑似列は、オブジェクト表、XMLType表、オブジェクト・ビューまたはXMLTypeビューの列のシステム生成名を戻します。この疑似列は、オブジェクト表の置換可能行の値の識別や、WITH OBJECT IDENTIFIER句を使用したオブジェクト・ビューの作成に便利です。

ノート:

以前のリリースでは、この疑似列は SYS_NC_ROWINFO$と呼ばれていました。下位互換性を保つため、この名称は引き続きサポートされます。ただし、より直感的な名前である OBJECT_VALUE を使用することをお薦めします。

関連項目:

- この疑似列の使用方法の詳細は、「object_table」および「object_view_clause」を参照してください。
- この疑似列の使用例については、『Oracle Databaseオブジェクト・リレーショナル開発者ガイド』を参照してください。

ORA_ROWSCN疑似列

ORA_ROWSCNは、行へ最も新しい変更のシステム変更番号(SCN)を表します。この変更には、ブロック・レベル(粗密)で行われるものと、行レベル(ファイングレイン)で行われるものがあります。後者は行レベル依存の追跡によって行われます。行レベル依存の追跡の詳細は、「CREATE TABLE」の「NOROWDEPENDENCIES | ROWDEPENDENCIES」を参照してください。行レベル依存がない場合、ORA_ROWSCNにはブロック・レベルの依存が反映されます。

ブロック・レベルの場合も行レベルの場合も、ORA_ROWSCNはSCNそのものであると考えないでください。たとえば、トランザクションによってあるブロック内のRという行が変更され、SCN 10でコミットされた場合、その行のORA_ROWSCNが常に10を返すとはかぎりません。10未満の値が返されることはありませんが、10以上のあらゆる値が返される可能性があります。つまり、行のORA_ROWSCNは、必ずその行を最後に変更したトランザクションのコミットSCNであるという保証はありません。しかし、ファイングレインORA_ROWSCNで、2つのトランザクションT1とT2によって同じ行Rが順番に変更され、コミットされた場合、T1のコミット後に行RのORA_ROWSCNに関する問合せで返される値と、T2のコミット後に返されるこの値では、T1のコミット後に返される値の方が小さくなります。同じブロックに関する問合せが2回行われた場合、最初の問合せと2番目の問合せの間にその行が更新されていなくても、ORA_ROWSCNの値は問合せ間で変わる可能性があります。唯一確実なことは、両方の問合せ内のORA_ROWSCNの値は、その行を最後に変更したトランザクションのコミットSCNよりも大きくなるということです。

ORA_ROWSCN疑似列はビューへの問合せでは使用できません。ただし、この疑似列を使用して、ビューの作成時に基礎となる表を参照することは可能です。また、UPDATE文またはDELETE文のWHERE句でこの疑似列を使用することもできます。

ORA_ROWSCNはFlashback Queryではサポートされていません。かわりに、Flashback Queryのために明示的に提供されているバージョン問合せの疑似列を使用します。フラッシュバック問合せの詳細は、SELECTのflashback_query_clauseを参照してください。バージョン問合せ疑似列の詳細は、バージョン問合せ疑似列を参照してください。

ORA_ROWSCNの制限事項: この疑似列は、外部表ではサポートされません。

例

次の最初の文では、ORA_ROWSCN疑似列を使用して、employees表に対する最後の操作のシステム変更番号を取得します。2番目の文では、この疑似列をSCN_TO_TIMESTAMPファンクションとともに使用して、操作のタイムスタンプを判別します。

```
SELECT ORA_ROWSCN, last_name
  FROM employees
  WHERE employee_id = 188;
SELECT SCN_TO_TIMESTAMP(ORA_ROWSCN), last_name
  FROM employees
  WHERE employee_id = 188;
```

関連項目:

SCN_TO_TIMESTAMP

ROWID疑似列

ROWID疑似列は、データベース内の各行について、行のアドレスを戻します。Oracle DatabaseのROWID値には、行を検索するために必要な次の情報が含まれています。

- オブジェクトのデータ・オブジェクト番号。
- 行が存在するデータファイルのデータ・ブロック。
- データ・ブロック内での行の位置(最初の行は0)。
- 行が存在するデータファイル(最初のファイルは1)。ファイル番号は表領域に対して相対的です。

ほとんどの場合、ROWID値ではデータベース内の行は一意に識別されます。ただし、同じクラスタに格納されている異なる表の行は、同じROWIDを持つことができます。

ROWID疑似列の値はROWIDまたはUROWIDデータ型を持ちます。詳細は、ROWIDデータ型およびUROWIDデータ型を参照してください。

ROWID値には、次の重要な用途があります。

- 単一の行に最も速くアクセスする方法です。
- 表の行が格納されている様子を示すことができます。
- これは、表の各行の一意識別子です。

表の主キーとしてROWIDを使用しないでください。たとえば、インポート・ユーティリティとエクスポート・ユーティリティで行を削除してから再挿入する場合、ROWIDが変わる場合があります。行を削除した場合、Oracleは、後から新しく挿入される行にそのROWIDを再度割り当てる可能性があります。

問合せのSELECT句とWHERE句でROWID疑似列を使用できますが、これらの疑似列の値が実際にデータベースに格納されるわけではありません。ROWID疑似列の値に対して挿入、更新および削除はできません。

例

次の文は、部門20の従業員のデータを含むすべての行のアドレスを選択します。

```
SELECT ROWID, last_name
  FROM employees
  WHERE department_id = 20;
```

ROWNUM疑似列

ノート:

- ROW_NUMBER 組込み SQL ファンクションは、問合せの結果の順序付けを強力にサポートします。詳細は、「ROW_NUMBER」を参照してください。
- SELECT 文の row_limiting_clause は、問合せから返される行数を制限するための強力なサポートを提供します。詳細は、「row_limiting_clause」を参照してください。

ROWNUM疑似列は、問合せによって戻される各行について、表や結合処理された行の集合からOracleが行を選択する順序を示す番号を戻します。つまり、選択される最初の行のROWNUMは1、2番目の行のROWNUMは2です(以降同様に続きます)。

次の例のように、ROWNUMを使用して問合せによって戻される行数を制限できます。

```
SELECT *
  FROM employees
  WHERE ROWNUM < 11;
```

同じ問合せでROWNUMにORDER BY句が続く場合、ORDER BY句によって行が再び順序付けられます。結果は、行がアクセスされる方法によって異なります。たとえば、ORDER BY句の指定によってOracleが索引を使用してデータにアクセスする場合、索引なしの場合とは異なる順序で行が取り出されることがあります。このため、次の文では、前述の例と同じ行が戻されるとはかぎりません。

```
SELECT *
  FROM employees
  WHERE ROWNUM < 11
  ORDER BY last_name;
```

ORDER BY句を副問合せに埋め込んでROWNUM条件をトップレベル問合せに置いた場合、行の順序付けの後でROWNUM条件を強制的に適用させることができます。たとえば、次の問合せは、小さい順に10個の従業員番号を持つ従業員を戻します。これは、上位N番のレポートと呼ばれることがあります。

```
SELECT *
  FROM (SELECT * FROM employees ORDER BY employee_id)
  WHERE ROWNUM < 11;
```

前述の例では、ROWNUM値はトップレベルのSELECT文の値です。これらの値は、副問合せ内のemployee_idによって行が順序付けられた後で生成されます。

比較条件「ROWNUM値> 正の整数」は、常に偽となるため注意してください。たとえば、次の問合せでは行は戻されません。

```
SELECT *
  FROM employees
  WHERE ROWNUM > 1;
```

最初にフェッチされる行のROWNUMには1が割り当てられるため、条件は偽と判断されます。2番目にフェッチされる予定だった行は最初の行になるため、このROWNUMにも1が割り当てられ、条件も偽と判断されます。このように、後続するすべての行が条件を満たさないため、行は戻されません。

また、次の例のように、ROWNUMを使用して表の各行に一意の値を割り当てることもできます。

```
UPDATE my_table
  SET column1 = ROWNUM;
```

行に一意の番号を割り当てる別の方法については、「ROW_NUMBER」を参照してください。

ノート:

問合せで ROWNUM を使用した場合、ビューの最適化に影響することがあります。

XMLDATA疑似列

Oracleは、XMLスキーマ情報およびSTORAGE句の指定方法に基づいて、XMLTypeデータをLOBまたはオブジェクト・リレーショナル列に格納します。XMLDATA疑似列を使用すると、基礎となるLOBまたはオブジェクト・リレーショナル列にアクセスし、追加のSTORAGE句のパラメータ、制約、索引などを指定できます。

例

次の文は、この疑似列を使用した例です。たとえば、CLOB列を1つ含む単純なXMLTypeの表を作成するとします。

```
CREATE TABLE xml_lob_tab of XMLTYPE
  XMLTYPE STORE AS CLOB;
```

基礎となるLOB列の記憶特性を変更する場合は、次の文を使用できます。

```
ALTER TABLE xml_lob_tab
  MODIFY LOB (XMLDATA) (STORAGE (MAXSIZE 2G) CACHE);
```

SQL文でのXMLの使用方法で作成したxwarehouses表のようなXMLスキーマベースの表を作成したとします。その後、次の文に示すように、XMLDATA列を使用して、基礎となる列のプロパティを設定できます。

```
ALTER TABLE xwarehouses
  ADD (UNIQUE(XMLDATA."WarehouseId"));
```

4 演算子

演算子は、データ項目を操作し、結果を戻すために使用します。構文上、演算子はオペランドの前後、または2つのオペランドの間に置かれます。

この章では、次の内容を説明します。

- SQL演算子
- 算術演算子
- COLLATE演算子
- 連結演算子
- 階層問合せ演算子
- 集合演算子
- MULTISET演算子
- ユーザー定義演算子

この章では、非論理(非ブール)演算子について説明します。これらの演算子は、問合せまたは副問合せ内でWHEREまたはHAVING句の条件として、単独では使用できません。条件として使用できる論理演算子の詳細は、「条件」を参照してください。

SQL演算子

演算子は、オペランドまたは引数と呼ばれる個々のデータ項目を操作します。演算子は特殊文字またはキーワードで表します。たとえば、乗算演算子は、アスタリスク(*)で表します。

Oracle Textがインストールされている場合は、Oracle Text問合せで、この製品に含まれるSCORE演算子を使用できます。また、CONTAINS、CATSEARCH、MATCHESなどの組込みText演算子を使用して条件を作成することもできます。Oracle Text要素の詳細は、『Oracle Textリファレンス』を参照してください。

単項演算子およびバイナリ演算子

一般に、演算子には次の2つのクラスがあります。

- 単項: 単項演算子は1つのオペランドのみを操作します。単項演算子は通常、オペランドとともに次の形式で使用します。

  ```
  operator operand
  ```

- バイナリ: 二項演算子は2つのオペランドを操作します。二項演算子は通常、オペランドとともに次の形式で使用します。

  ```
  operand1 operator operand2
  ```

特殊な形式を持つ他の演算子では3つ以上のオペランドを使用できます。オペランドにnullが指定された場合、結果は常にnullになります。この規則に従わない唯一の演算子は、連結(||)です。

演算子の優先順位

優先順位とは、同じ式の中の異なる演算子をOracle Databaseが評価する順序を意味します。複数の演算子を含む式を評価するとき、Oracleは優先順位の高い演算子を評価した後で、優先順位の低い演算子を評価します。優先順位の等しい演算子は、式の中で左から右に評価されます。

表4-1に、SQL演算子を優先順位の高い方から順に示します。同じ行にある演算子の優先順位は同じです。

表4-1 SQL演算子の優先順位

演算子	操作
+, - (単項演算子)、PRIOR、CONNECT_BY_ROOT、COLLATE	同一、否定、階層内の位置
*, /	乗算、除算
+, - (バイナリ演算子), ||	加算、減算、連結
SQL 条件は、SQL 演算子の後で評価されます。	「条件の優先順位」を参照してください。

優先順位の例

次の式では、乗算は加算よりも優先順位が高いため、2と3を掛けた結果に1が加算されます。

```
1+2*3
```

式の中でカッコを使用すると、演算子の優先順位をオーバーライドできます。カッコ内の式がカッコ外の式よりも先に評価されます。

SQLでは、集合演算子(UNION、UNION ALL、INTERSECTおよびMINUS)もサポートされます。集合演算子によって結合されるのは、問合せによって戻される行の集まりで、個々のデータ項目ではありません。集合演算子の優先順位はすべて同じです。

関連項目:

階層問合せでのみ使用するPRIOR演算子の詳細は、階層問合せ演算子および階層問合せを参照してください。

算術演算子

算術演算子を1つまたは2つの引数とともに使用して、数値を否定、加算、減算、乗算および除算できます。これらの演算子の一部は日時と期間の算術演算でも使用されます。演算子の引数は、数値データ型または数値データ型に暗黙的に変換できる任意のデータ型に解決される必要があります。

単項算術演算子は、引数の数値データ型と同じデータ型を戻します。バイナリ算術演算子の場合、Oracleは、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。表4-2に、算術演算子を示します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。数値の優先順位の詳細は、数値の優先順位を参照してください。また、日時および期間の演算を参照してください

表4-2 算術演算子

演算子	目的	例
+ -	正または負の式を表す場合は単項演算子です。	`SELECT * FROM order_items WHERE quantity = -1 ORDER BY order_id, line_item_id, product_id; SELECT * FROM employees WHERE -salary < 0 ORDER BY employee_id;`
+ -	加算または減算の場合は二項演算子です。	`SELECT hire_date FROM employees WHERE SYSDATE - hire_date > 365 ORDER BY hire_date;`
* /	乗算、除算を行います。これらは二項演算子です。	`UPDATE employees SET salary = salary * 1.1;`

二重否定または負の値の減算を表す場合に、連続する2つのマイナス記号(--)を算術式で使用しないでください。文字--は、SQL文ではコメントの開始を示す場合に使用します。スペースまたはカッコを使用して、連続するマイナス記号を区切る必要があります。SQL文中のコメントの詳細は、コメントを参照してください。

COLLATE演算子

COLLATE演算子により、式の照合が決定されます。この演算子を使用すると、式に対してデータベースが標準の照合導出ルールを使用して導出した照合を上書きできます。

COLLATEは後置単項演算子です。優先順位は他の単項演算子と同じになりますが、すべての前置単項演算子が評価された後に評価されます。

この演算子は、VARCHAR2、CHAR、LONG、NVARCHARまたはNCHAR型の式に適用できます。

COLLATE演算子では1つの引数collation_nameを取得し、この引数には、名前付き照合または疑似照合を指定できます。照合名に空白が含まれている場合は、名前を二重引用符で囲む必要があります。

表4-3に、COLLATE演算子を示します。

表4-3 COLLATE演算子

演算子	目的	例
COLLATE collation_name	式の照合を決定します。	SELECT last_name FROM employees ORDER BY last_name COLLATE GENERIC_M;

関連項目:

- 複合式でのCOLLATE演算子の使用方法の詳細は、複合式を参照してください。
- COLLATE演算子の詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

連結演算子

連結演算子は、文字列およびCLOBデータを操作する場合に使用します。表4-4は、連結演算子を示しています。

表4-4 連結演算子

演算子	目的	例
||	文字列および CLOB データを連結します。	`SELECT 'Name is ' || last_name FROM employees ORDER BY last_name;`

2つの文字列を連結した結果は、別の文字列になります。両方の文字列がCHARデータ型の場合、結果はCHARデータ型の文字列になり、その最大文字数は2000です。両方の文字列のデータ型がVARCHAR2の場合、その結果のデータ型はVARCHAR2になります。このデータ型は、初期化パラメータがMAX_STRING_SIZE = EXTENDEDの場合は32,767文字に制限され、MAX_STRING_SIZE = STANDARDの場合は4,000文字に制限されます。詳細は、拡張データ型を参照してください。どちらかの引数がCLOBデータ型の場合、結果は、一時CLOBになります。データ型が文字列型かCLOB型かにかかわらず、後続空白は連結後も文字列に残ります。

多くのプラットフォームでは、連結演算子は、表4-4に示すとおり2本の実線垂直バーで表されます。ただし、IBM社のプラットフォームの中には、この演算子として破線垂直バーを使用するものもあります。異なる文字セットを持つシステム間(たとえばASCIIとEBCDIC間)でSQLスクリプト・ファイルを移動する場合、垂直バーが、移動先のOracle Database環境で必要な垂直バーに変換されない場合があります。オペレーティング・システムまたはネットワーク・ユーティリティによる変換の制御が困難または不可能である場合に備えて、Oracleでは、垂直バー演算子にかわるものとしてCONCAT文字ファンクションが提供されています。異なる文字セットを持つ環境間でアプリケーションを移動する場合は、この文字ファンクションを使用することをお薦めします。

Oracleは、長さが0(ゼロ)の文字列をNULLとして処理しますが、長さが0(ゼロ)の文字列を別のオペランドと連結すると、その結果は常にもう一方のオペランドになります。結果がNULLになるのは、2つのNULL文字列を連結したときのみです。ただし、この処理はOracle Databaseの今後のバージョンでも継続されるとはかぎりません。nullになる可能性のある式を連結する場合は、NVL関数を使用してその式を長さ0の文字列に明示的に変換してください。

関連項目:

- CHARデータ型とVARCHAR2データ型の違いの詳細は、文字データ型を参照してください。
- CONCATファンクションおよびNVLファンクションを参照してください。
- CLOBの詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。
- 連結演算子の照合導出ルールの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

連結の例

次の例では、CHAR列およびVARCHAR2列を持つ表を作成し、後続空白のある値とない値を挿入してから、これらの値を選択し、連結します。なお、CHAR列およびVARCHAR2列では、ともに後続空白が保存されます。

```
CREATE TABLE tab1 (col1 VARCHAR2(6), col2 CHAR(6),
                   col3 VARCHAR2(6), col4 CHAR(6));
INSERT INTO tab1 (col1,  col2,     col3,     col4)
```

```
          VALUES ('abc', 'def   ', 'ghi   ', 'jkl');
SELECT col1 || col2 || col3 || col4 "Concatenation"
  FROM tab1;
Concatenation
------------------------
abcdef   ghi   jkl
```

階層問合せ演算子

PRIORおよびCONNECT_BY_ROOTの2つの演算子は、階層問合せでのみ有効です。

PRIOR

階層問合せでは、CONNECT BY condition内の1つの式をPRIOR演算子で修飾する必要があります。CONNECT BY conditionが複合条件の場合、1つの条件のみにPRIOR演算子が必要です(複数のPRIOR条件を使用することもできます)。PRIORは、階層問合せ内でカレント行の親である行の直後にある式を評価します。

PRIORは、等価演算子を使用して列の値を比較する場合によく使用されます。(PRIORキーワードは演算子のどちら側でもかまいません。)PRIORを指定すると、列の親である行の値が使用されます。等号(=)以外の演算子は、理論上はCONNECT BY句に指定できます。ただし、これらの他の演算子の組合せによっては、作成される条件は無限ループを発生させる場合があります。この場合、実行時にループが検出され、エラーが戻されます。この演算子の詳細および例は、階層問合せを参照してください。

CONNECT_BY_ROOT

CONNECT_BY_ROOTは、階層問合せでのみ有効な単項演算子です。この演算子を使用して列を修飾すると、ルート行のデータを使用して列の値が戻されます。この演算子は、階層問合せのCONNECT BY [PRIOR]条件の機能を拡張します。

CONNECT_BY_ROOTの制限事項

START WITH条件またはCONNECT BY条件では、この演算子は指定できません。

関連項目:

CONNECT_BY_ROOTの例

集合演算子

集合演算子は、2つのコンポーネントの問合せ結果を1つの結果にまとめます。集合演算子を含む問合せを複合問合せと呼びます。表4-5に、SQLの集合演算子を示します。これらの演算子の例や制限事項などの詳細は、UNION [ALL]、INTERSECTおよびMINUS演算子を参照してください。

表4-5 集合演算子

演算子	返される内容
UNION	各問合せによって戻るすべての行(重複行は含まない)
UNION ALL	各問合せによって戻るすべての行(重複行を含む)
INTERSECT	両方の問合せによって戻るすべての行(重複行は含まない)
MINUS	最初の問合せによって戻る行で、2 番目の問合せでは戻されない行(重複行は含まない)

MULTISET演算子

MULTISET演算子は、2つのネストした表の結果を1つのネストした表にまとめます。

MULTISET演算子に関する例では、次のように2つのネストした表を作成し、データをロードする必要があります。

まず、customers_demoという名前で、oe.customers表のコピーを作成します。

```
CREATE TABLE customers_demo AS
  SELECT * FROM customers;
```

次に、cust_address_tab_typという表型を作成します。この型はネストした表の列を作成する際に使用します。

```
CREATE TYPE cust_address_tab_typ AS
  TABLE OF cust_address_typ;
/
```

次に、customers_demo表に、ネストした表の列を2つ作成します。

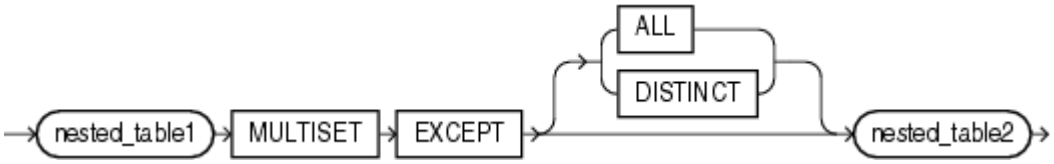
```
ALTER TABLE customers_demo
  ADD (cust_address_ntab cust_address_tab_typ,
       cust_address2_ntab cust_address_tab_typ)
    NESTED TABLE cust_address_ntab STORE AS cust_address_ntab_store
    NESTED TABLE cust_address2_ntab STORE AS cust_address2_ntab_store;
```

最後に、oe.customers表のcust_address列のデータを使用して、2つの新しいネストした表の列にデータをロードします。

```
UPDATE customers_demo cd
  SET cust_address_ntab =
    CAST(MULTISET(SELECT cust_address
                  FROM customers c
                  WHERE c.customer_id =
                        cd.customer_id) as cust_address_tab_typ);
UPDATE customers_demo cd
  SET cust_address2_ntab =
    CAST(MULTISET(SELECT cust_address
                  FROM customers c
                  WHERE c.customer_id =
                        cd.customer_id) as cust_address_tab_typ);
```

MULTISET EXCEPT

MULTISET EXCEPTは、引数として2つのネストした表を取り、1つ目のネストした表に存在し、2つ目のネストした表に存在しない要素を持つネストした表を戻します。入力する2つのネストした表は同じ型である必要があり、戻されるネストした表も同じ型です。



- ALLキーワードは、nested_table2に存在せず、nested_table1に存在するすべての要素を戻します。たとえば、ある特定の要素がnested_table1でm回、nested_table2でn回出現すると想定します。結果には、この要素は、m >nの場合は(m-n)回、m<=nの場合は0(ゼロ)回出現します。ALLはデフォルトです。

- DISTINCTキーワードは、出現回数にかかわらず、nested_table1とnested_table2の両方に存在するすべての要素を排除します。
- ネストした表の要素型は比較可能なものである必要があります。スカラー型以外の型の比較の可能性は、比較条件を参照してください。

例

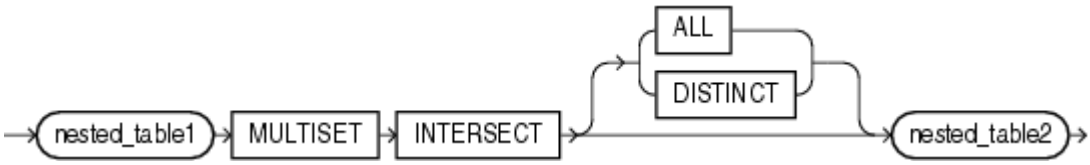
次の例では、2つのネストした表を比較し、1つ目のネストした表に存在し、2つ目のネストした表に存在しない要素を持つネストした表を戻します。

```
SELECT customer_id, cust_address_ntab
  MULTISET EXCEPT DISTINCT cust_address2_ntab multiset_except
  FROM customers_demo
  ORDER BY customer_id;
CUSTOMER_ID MULTISET_EXCEPT(STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE,
COUNTRY_ID)
----------- ------------------------------------------------------------------------
-------
        101 CUST_ADDRESS_TAB_TYP()
        102 CUST_ADDRESS_TAB_TYP()
        103 CUST_ADDRESS_TAB_TYP()
        104 CUST_ADDRESS_TAB_TYP()
        105 CUST_ADDRESS_TAB_TYP()
. . .
```

この例では、表customers_demoと、データを含むネストした表の列が2つ必要です。この表およびネストした表の列を作成する方法は、MULTISET演算子を参照してください。

MULTISET INTERSECT

MULTISET INTERSECTは、引数として2つのネストした表を取り、入力する2つのネストした表に共通する値を持つネストした表を戻します。入力する2つのネストした表は同じ型である必要があり、戻されるネストした表も同じ型です。



- ALLキーワードは、入力する2つのネストした表に存在するすべての共通要素を戻します(重複する共通の値や、重複する共通のNULLも含まれます)。たとえば、ある特定の値がnested_table1でm回、nested_table2でn回出現する場合、結果には、この要素がmin(m,n)回出現します。ALLはデフォルトです。
- DISTINCTキーワードは、戻されるネストした表から重複を排除します(重複するNULLが存在する場合、これも排除されます)。
- ネストした表の要素型は比較可能なものである必要があります。スカラー型以外の型の比較の可能性は、比較条件を参照してください。

例

次の例では、2つのネストした表を比較し、入力する両方のネストした表に存在する要素を持つネストした表を戻します。

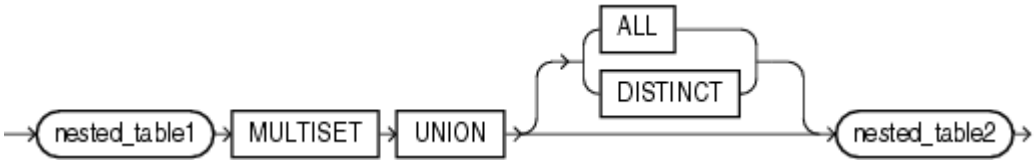
```
SELECT customer_id, cust_address_ntab
  MULTISET INTERSECT DISTINCT cust_address2_ntab multiset_intersect
  FROM customers_demo
  ORDER BY customer_id;
```

```
CUSTOMER_ID MULTISET_INTERSECT(STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE,
COUNTRY_ID
----------- ------------------------------------------------------------------------
----------
        101 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('514 W Superior St', '46901',
'Kokomo', 'IN', 'US'))
        102 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('2515 Bloyd Ave', '46218',
'Indianapolis', 'IN', 'US'))
        103 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('8768 N State Rd 37', '47404',
'Bloomington', 'IN', 'US'))
        104 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('6445 Bay Harbor Ln', '46254',
'Indianapolis', 'IN', 'US'))
        105 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('4019 W 3Rd St', '47404',
'Bloomington', 'IN', 'US'))
. . .
```

この例では、表customers_demoと、データを含むネストした表の列が2つ必要です。この表およびネストした表の列を作成する方法は、MULTISET演算子を参照してください。

MULTISET UNION

MULTISET UNIONは、引数として2つのネストした表を取り、入力する2つのネストした表に存在する値を持つネストした表を戻します。入力する2つのネストした表は同じ型である必要があり、戻されるネストした表も同じ型です。



- ALLキーワードは、入力する2つのネストした表に存在するすべての要素を戻します(重複する値や、重複するNULLも含まれます)。これはデフォルトです。
- DISTINCTキーワードは、戻されるネストした表から重複を排除します(重複するNULLが存在する場合、これも排除されます)。
- ネストした表の要素型は比較可能なものである必要があります。スカラー型以外の型の比較の可能性は、比較条件を参照してください。

例

次の例では、2つのネストした表を比較し、入力する両方のネストした表に存在する要素を持つネストした表を戻します。

```
SELECT customer_id, cust_address_ntab
  MULTISET UNION cust_address2_ntab multiset_union
  FROM customers_demo
  ORDER BY customer_id;
CUSTOMER_ID MULTISET_UNION(STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE,
COUNTRY_ID)
----------- ------------------------------------------------------------------------
------
        101 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('514 W Superior St', '46901',
'Kokomo', 'IN', 'US'),
                                 CUST_ADDRESS_TYP('514 W Superior St', '46901',
'Kokomo', 'IN', 'US'))
        102 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('2515 Bloyd Ave', '46218',
'Indianapolis', 'IN', 'US'),
                                 CUST_ADDRESS_TYP('2515 Bloyd Ave', '46218',
'Indianapolis', 'IN','US'))
        103 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('8768 N State Rd 37', '47404',
'Bloomington', 'IN', 'US'),
```

```
                                    CUST_ADDRESS_TYP('8768 N State Rd 37', '47404',
'Bloomington', 'IN', 'US'))
        104 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('6445 Bay Harbor Ln', '46254',
'Indianapolis', 'IN', 'US'),
                                    CUST_ADDRESS_TYP('6445 Bay Harbor Ln', '46254',
'Indianapolis', 'IN', 'US'))
        105 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('4019 W 3Rd St', '47404',
'Bloomington', 'IN', 'US'),
                                    CUST_ADDRESS_TYP('4019 W 3Rd St', '47404',
'Bloomington', 'IN', 'US'))
. . .
```

この例では、表customers_demoと、データを含むネストした表の列が2つ必要です。この表およびネストした表の列を作成する方法は、MULTISET演算子を参照してください。

ユーザー定義演算子

ユーザー定義演算子は、組込み演算子のように、一連のオペランドを入力として受け取り、結果を戻します。ユーザー定義演算子は、ユーザーがCREATE OPERATOR文で作成し、ユーザー定義の名前で識別されます。これらは、表、ビュー、型およびスタンドアロン・ファンクションと同じネームスペースに存在します。

新規の演算子を定義すると、他の組込み演算子のようにSQL文で使用できます。たとえば、SELECT文のSELECT構文のリスト、WHERE句の条件、ORDER BY句およびGROUP BY句でユーザー定義演算子を使用できます。ただし、これはユーザー定義オブジェクトであるため、演算子に対するEXECUTE権限が必要です。

関連項目:

演算子の作成例は、CREATE OPERATORを、ユーザー定義演算子の詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。

5 式

この章では、値、演算子およびファンクションを式の中で組み合せて使用する方法について説明します。

この章では、次の内容を説明します。

- SQL式
- 単純式
- 分析ビュー式
- 複合式
- CASE式
- 列式
- CURSOR式
- 日時式
- ファンクション式
- 期間式
- JSONオブジェクト・アクセス式
- モデル式
- オブジェクト・アクセス式
- プレースホルダ式
- スカラー副問合せ式
- 型コンストラクタ式
- 式のリスト

SQL式

式は、1つ以上の値、演算子、および値に評価されるSQLファンクションの組合せです。一般に、式のデータ型は、そのコンポーネントのデータ型になります。

この単純な式は4に評価され、データ型はNUMBER (構成要素と同じデータ型)です。

```
2*2
```

次の式は、関数と演算子を使用するより複雑な式の例です。この式は現在の日付に7日を加算し、合計から時間の要素を削除し、結果をCHARデータ型に変換します。

```
TO_CHAR(TRUNC(SYSDATE+7))
```

次の場所で式を使用できます。

- SELECT文のselectリスト
- WHERE句およびHAVING句の条件
- CONNECT BY句、START WITH句およびORDER BY句
- INSERT文のVALUES句
- UPDATE文のSET句

たとえば、次のUPDATE文のSET句で、引用符で囲まれた文字列'Smith'のかわりに式を使用することもできます。

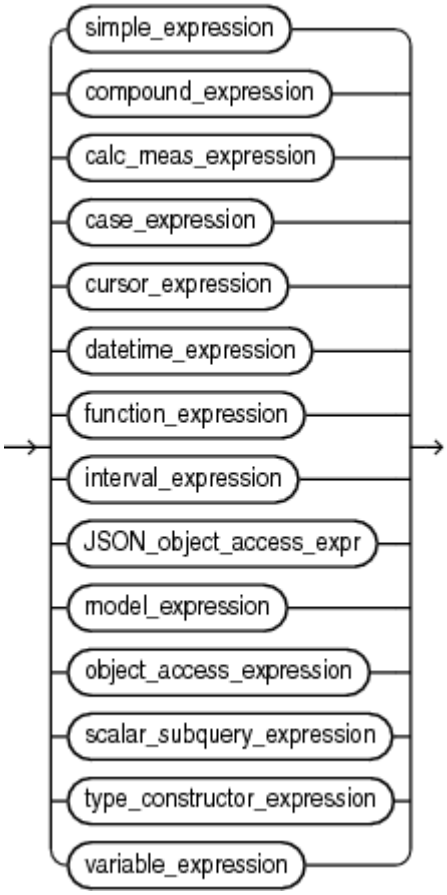
```
SET last_name = 'Smith';
```

このSET句では、引用符で囲まれた文字列'Smith'のかわりに、INITCAP(last_name)を使用しています。

```
SET last_name = INITCAP(last_name);
```

次の構文に示すとおり、式にはいくつかの書式があります。

expr::=



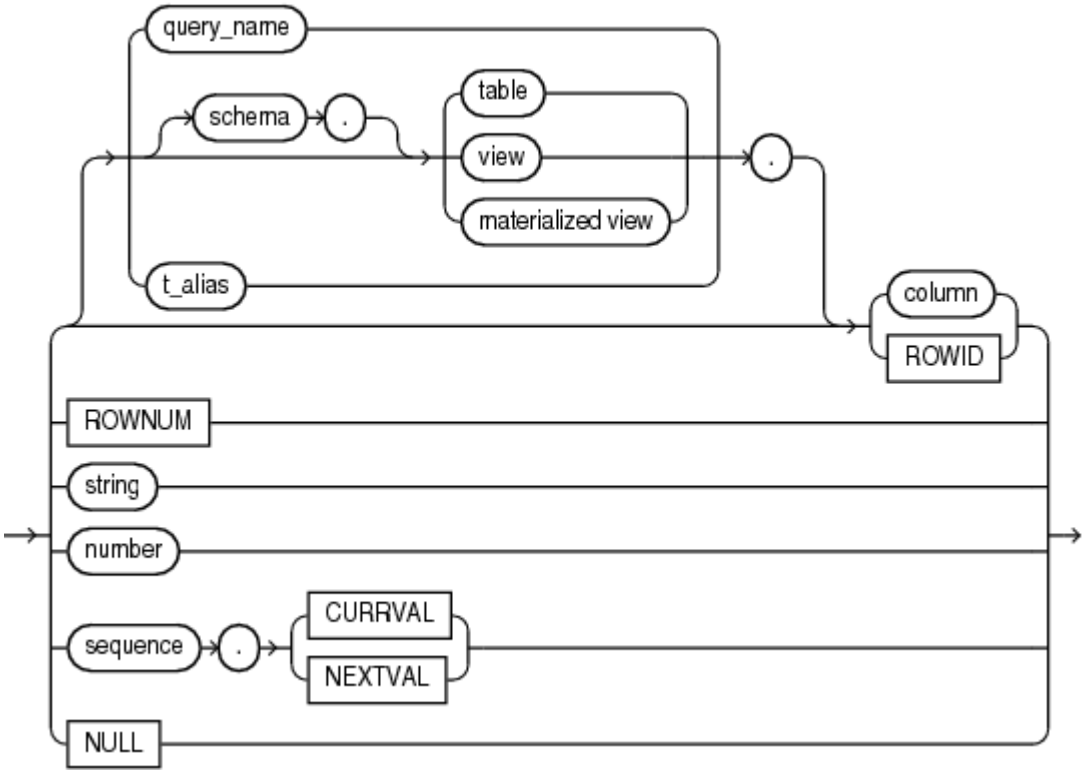
Oracle Databaseは、すべてのSQL文のすべての部分で、式のすべての書式を受け入れるわけではありません。該当する文における式の制限事項の詳細は、このマニュアルの特定のSQL文に関する項を参照してください。

このマニュアルの他の箇所で、条件、SQLファンクションまたはSQL文にexprが示されている場合は、必ず適切な式の表記法を使用してください。それに続く項では、様々な式の例をあげて説明しています。

単純式

単純式は、列、疑似列、定数、順序番号またはNULLを指定します。

simple_expression::=



スキーマは各ユーザー用の他に、"PUBLIC"(二重引用符が必要)にもなり得ます。その場合、スキーマは表、ビューまたはマテリアライズド・ビューのパブリック・シノニムを修飾する必要があります。"PUBLIC"でのパブリック・シノニムの修飾は、データ操作言語(DML)文でのみサポートされています。データ定義言語(DDL)文ではサポートされていません。

ROWIDは、表でのみ使用でき、ビューまたはマテリアライズド・ビューでは使用できません。NCHARおよびNVARCHAR2は、有効な疑似列データ型ではありません。

関連項目:

疑似列の詳細は、「疑似列」を参照してください。query_nameの詳細は、「subquery_factoring_clause」を参照してください。

有効な単純式の例を次に示します。

```
employees.last_name
'this is a text string'
10
N'this is an NCHAR string'
```

分析ビュー式

分析ビュー式を使用すると、分析ビューの定義内、または分析ビューから選択する問合せに計算済メジャーを作成できます。

分析ビュー式は、表や列ではなく、階層および分析ビューの要素を参照する点で他のタイプの式とは異なります。

分析ビュー式は次のいずれかです。

- `av_meas_expression` (これは分析ビューのメジャーに基づいています)
- `av_hier_expression` (これは関連メンバーの属性値を戻します)

分析ビュー式は、`CREATE ANALYTIC VIEW`文の`calc_measure_clause`、および`SELECT`文の`WITH`句または`FROM`句で、`calc_meas_expression`パラメータとして使用できます。

計算済メジャーを定義する場合は、次のタイプの式を使用することもできます。

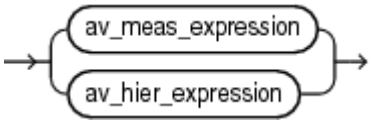
- 単純
- CASE
- 複合
- 日時
- 期間

ヒント:

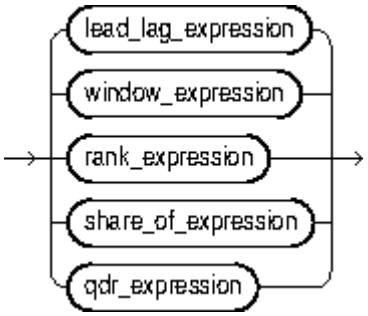
計算済メジャーを使用する分析ビューを作成する SQL スクリプトは、Oracle Live SQL の Web サイト (https://livesql.oracle.com/apex/livesql/file/index.html)で表示および実行できます。この Web サイトには、分析ビューの作成方法および使用方法を示すスクリプトおよびチュートリアルが用意されています。

構文

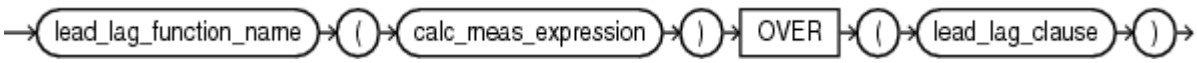
av_expression::=



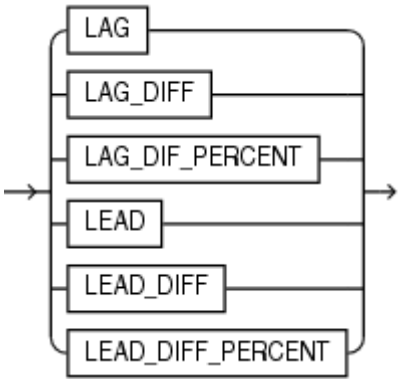
av_meas_expression::=



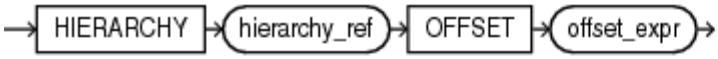
lead_lag_expression::=

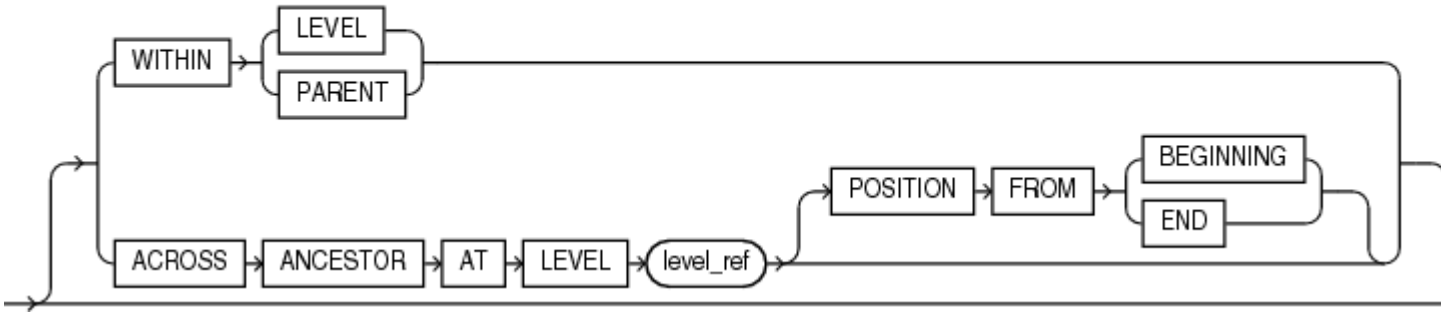


lead_lag_function_name::=

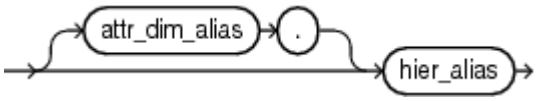


lead_lag_clause::=



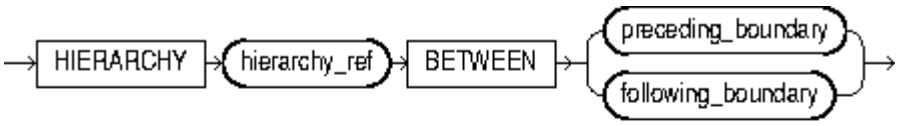


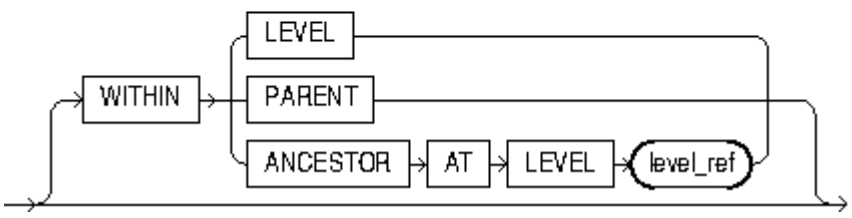
hierarchy_ref::=



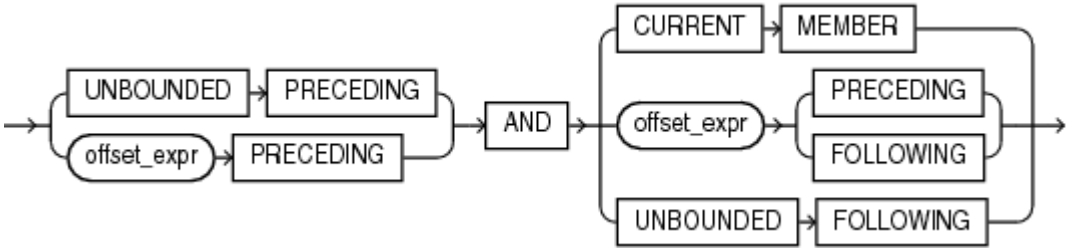
window_expression::=

window_clause::=

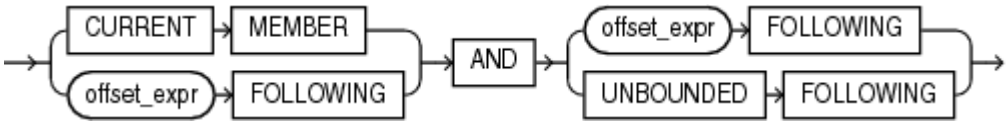




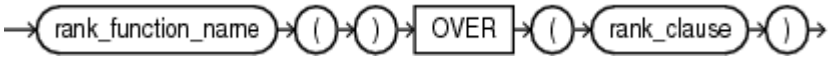
preceding_boundary ::=



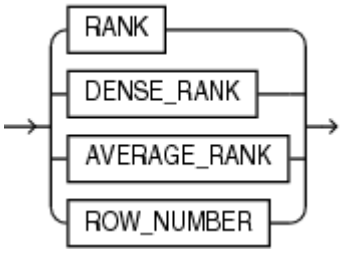
following_boundary::=



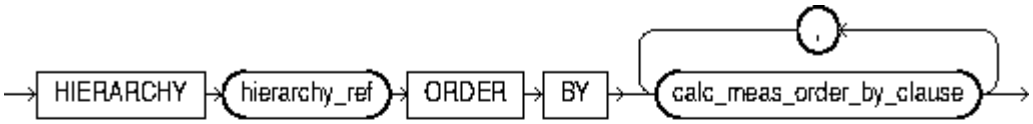
rank_expression::=

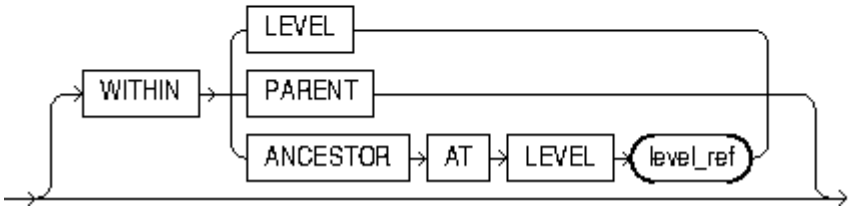


rank_function_name::=

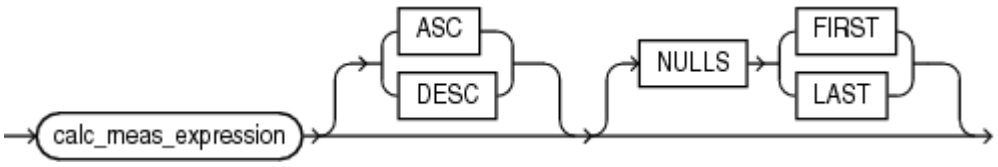


rank_clause::=

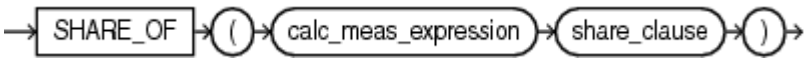




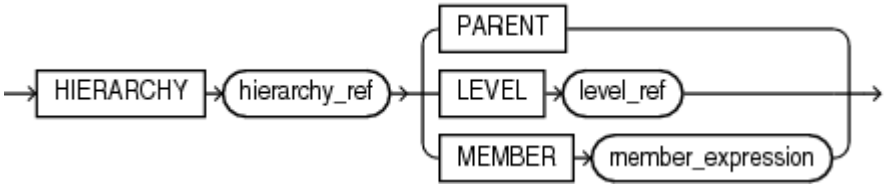
calc_meas_order_by_clause::=



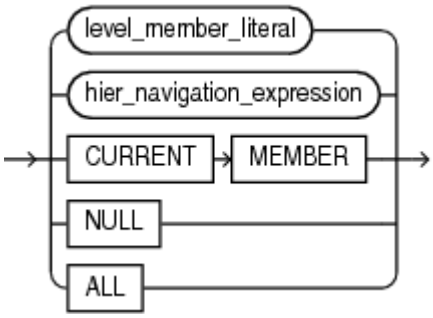
share_of_expression::=



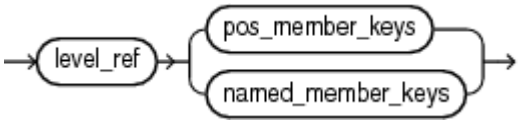
share_clause::=



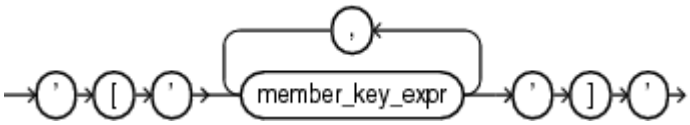
member_expression::=



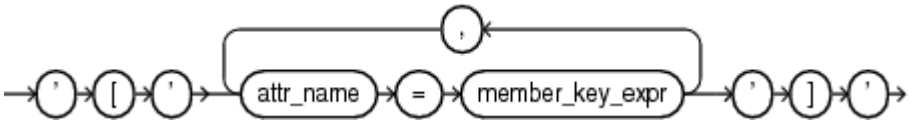
level_member_literal::=



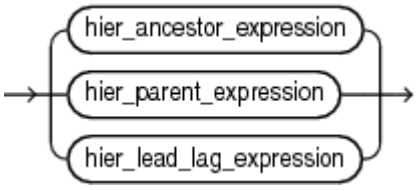
pos_member_keys::=



named_member_keys::=

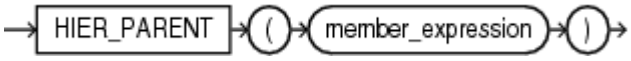


hier_navigation_expression::=

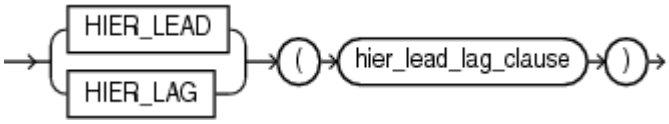


hier_ancestor_expression::=

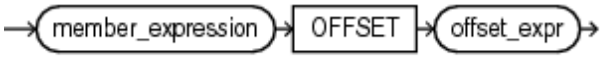
hier_parent_expression::=

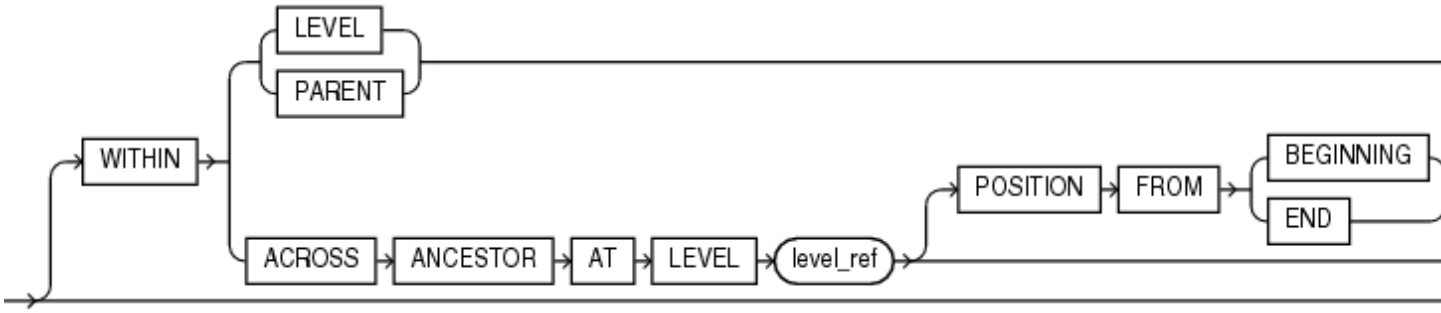


hier_lead_lag_expression::=

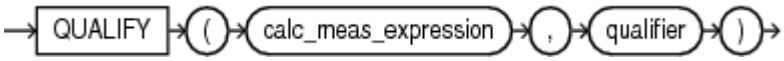


hier_lead_lag_clause::=

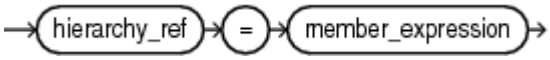




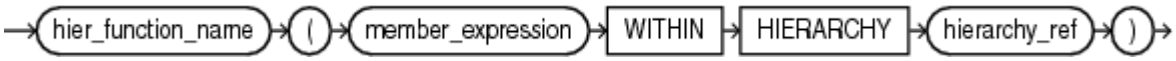
qdr_expression::=



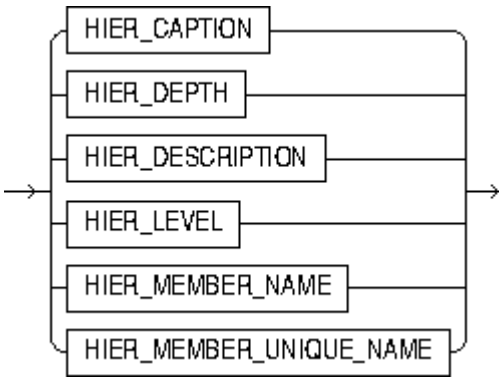
qualifier::=



av_hier_expression::=



hier_function_name::=



セマンティクス

av_meas_expression

関連するメジャー値を検索するために、階層ナビゲーションを実行する式。

lead_lag_expression

階層内で、LEAD操作またはLAG操作を指定して、ある数のメンバー分だけ前後にナビゲートして、関連するメジャー値を検索する式。

calc_meas_expressionパラメータは、lead_lag_expressionによって作成される新しいコンテキストで評価されます。このコンテキストには、指定される階層のメンバーが、LEAD操作またはLAG操作によって指定された関連するメンバーに変更される以外は、外部コンテキストと同じメンバーが含まれます。LEADファンクションまたはLAGファンクションは、lead_lag_clauseパラメータで指定された階層メンバーに対して実行されます。

lead_lag_function_name

LEADファンクションまたはLAGファンクションは、次のいずれかになります。

- LAGでは、先行するメンバーのメジャー値を戻します。
- LAG_DIFFでは、現在のメンバーのメジャー値と先行するメンバーのメジャー値との差を戻します。
- LAG_DIFF_PERCENTでは、現在のメンバーのメジャー値と先行するメンバーのメジャー値との相違率を戻します。
- LEADでは、後続のメンバーのメジャー値を戻します。
- LEAD_DIFFでは、現在のメンバーのメジャー値と後続のメンバーのメジャー値との差を戻します。
- LEAD_DIFF_PERCENTでは、現在のメンバーのメジャー値と後続のメンバーのメジャー値との相違率を戻します。

lead_lag_clause

評価する階層およびオフセット値を指定します。lead_lag_clauseのパラメータは次のとおりです。

- HIERARCHY hierarchy_refでは、分析ビューで定義されている階層の別名を指定します。
- OFFSET offset_exprでは、数値に変換するcalc_meas_expressionを指定します。数値は、現在のメンバーから前後に何メンバー分移動するかを指定します。レベル内のメンバーの順序は、階層で使用する属性ディメンションの定義によって決まります。
- WITHIN LEVELでは、現在のメンバーと深さのレベルが同じメンバー内で、メンバーのオフセット数に従って前後に移動して、関連するメンバーを検索することを指定します。レベル内のメンバーの順序は、階層で使用する属性ディメンションの定義によって決まります。

  WITHIN LEVELキーワードとACROSS ANCESTOR AT LEVELキーワードのどちらも指定されていない場合は、WITHIN LEVEL操作がデフォルトとなります。
- WITHIN PARENTでは、現在のメンバーと親が同じメンバー内で、メンバーのオフセット数に従って前後に移動して、関連するメンバーを検索することを指定します。
- ACROSS ANCESTOR AT LEVEL level_refでは、level_refで指定されたレベルの現在のメンバーの祖先(または祖先が存在しない場合はメンバー自体)までナビゲートして関連するメンバーを検索することを指定し、かつ、その親内の各祖先メンバー(メンバー自体を含む)の位置を示します。level_refパラメータは、指定された階層のレベルの名前です。

  祖先メンバーが見つかると、ナビゲーションは、祖先メンバーと深さが同じメンバー内で、メンバーのオフセット数だけ前後に移動します。関連する祖先が見つかった後、ナビゲーションはそのメンバーから階層を逆方向に進み、上に向かって記録された(逆順)親内の位置を照合します。親内での位置は、POSITION FROM BEGINNINGとPOSITION FROM ENDのいずれが指定されているかに従い、最初の子または最後の子からオフセットされます。デフォルト値はPOSITION FROM BEGINNINGです。レベル内のメンバーの順序は、階層で使用する属性ディメンションの定義に

よって決まります。

window_expression

window_expressionでは、現在のメンバーから始まる指定された範囲内にあり、かつ、現在のメンバーと同じ深さにある一連のメンバーを選択します。WITHIN句を使用して階層関係を指定することによって、このメンバーの選択をさらに制限できます。その後、選択したメジャー値で集計が実行され、式に対する1つの結果が生成されます。

window_expressionのパラメータは次のとおりです。

- aggregate_functionは、COLLECT、GROUP_ID、GROUPING、GROUPING_ID、SYS_XMLAGG、XMLAGGおよびすべての複数引数ファンクションを除く、任意の既存のSQL集計ファンクションです。ユーザー定義集計ファンクションも使用できます。集計ファンクションの引数はcalc_meas_expression式です。これらの式は、外部コンテキストを使用して評価され、指定された階層のメンバーは、関連する範囲の各メンバーに変更されます。このため、それぞれの式引数は関連するメンバーごとに1回評価されます。その後、結果が、aggregate_functionを使用して集計されます。
- OVER (window_clause)では、使用する階層および考慮するウィンドウの境界を指定します。

関連項目:

集計ファンクション

window_clause

window_clauseパラメータでは、現在のメンバーに関連するメンバーの範囲を選択します。この範囲は、preceding_boundaryパラメータまたはfollowing_boundaryパラメータで指定されたメンバーの間になります。範囲は、常に、現在のメンバーと同じレベルのメンバーに対して計算されます。

window_clauseのパラメータは次のとおりです。

- HIERARCHY hierarchy_refでは、分析ビューで定義されている階層の別名を指定します。
- BETWEEN preceding_boundaryまたはfollowing_boundaryでは、現在のメンバーに関連する一連のメンバーを定義します。
- WITHIN LEVELでは、現在のレベルのすべてのメンバーにboundary句を適用することによって、関連するメンバーを選択します。これは、WITHINキーワードが指定されていない場合にデフォルトとなります。
- WITHIN PARENTでは、現在のメンバーと親を共有するすべてのメンバーにboundary句を適用することによって、関連するメンバーを選択します。
- WITHIN ANCESTOR AT LEVELでは、現在のメンバーと指定されたレベルの祖先を共有する(またはメンバー自体である)現在の深さのすべてのメンバーにboundary句を適用することによって、関連するメンバーを選択します。現在のメンバーが指定されたレベルよりも上にある場合、ウィンドウ式の値はNULLとなります。指定された階層に、このレベルがない場合は、エラーが発生します。

preceding_boundary

preceding_boundaryパラメータではメンバーの範囲を定義し、その先頭は、現在のメンバーのレベルで、指定された数だけ戻ったメンバーとなり、末尾は指定された境界の端となります。次のパラメータでは、範囲を指定します。

- UNBOUNDED PRECEDINGを指定すると、レベルの最初のメンバーが範囲の先頭になります。

- offset_expr PRECEDINGを指定すると、現在のメンバーからoffset_exprの数だけ戻ったメンバーが範囲の先頭になります。offset_expr式は、数値に変換するcalc_meas_expressionです。オフセット数が、そのレベルの現在のメンバーから最初のメンバーまでのメンバーの数より大きい場合、最初のメンバーが範囲の開始点として使用されます。
- CURRENT MEMBERを指定すると、現在のメンバーが範囲の末尾となります。
- offset_expr PRECEDINGを指定すると、現在のメンバーからoffset_exprだけ戻ったメンバーが範囲の末尾になります。
- offset_expr FOLLOWINGを指定すると、現在のメンバーからoffset_expr分だけ進んだメンバーが範囲の末尾になります。
- UNBOUNDED FOLLOWINGを指定すると、そのレベルの最後のメンバーが範囲の末尾になります。

following_boundary

following_boundaryパラメータではメンバーの範囲を定義し、その先頭は、現在のメンバーから指定された数だけ進んだメンバーになり、末尾は、指定した範囲の終わりになります。次のパラメータでは、範囲を指定します。

- CURRENT MEMBERを指定すると、現在のメンバーが範囲の先頭になります。
- offset_expr FOLLOWINGを指定すると、現在のメンバーからoffset_exprだけ進んだメンバーが範囲の先頭になります。
- offset_expr FOLLOWINGを指定すると、現在のメンバーからoffset_expr分だけ進んだメンバーが範囲の末尾になります。
- UNBOUNDED FOLLOWINGを指定すると、そのレベルの最後のメンバーが範囲の末尾になります。

hierarchy_ref

分析ビューの階層への参照。hier_aliasパラメータでは、分析ビューの定義に階層の別名を指定します。特殊文字のエスケープまたはケースの保持、あるいはその両方を行うには、二重引用符を使用します。

オプションのattr_dim_aliasパラメータでは、分析ビューの定義に属性ディメンションの別名を指定します。指定された階層の別名が分析ビューの別の階層の別名と競合する場合、または分析ビューの定義で属性ディメンションが複数回使用されている場合は、attr_dim_aliasパラメータを使用して、あいまいさを解決できます。attr_dim_aliasパラメータは、名前の競合が存在しない場合も使用できます。

rank_expression

階層のランク計算では、指定されたメジャー値の順序に基づいて、指定された階層の関連メンバーがランク付けされ、その結果内で現在のメンバーのランクが戻されます。

階層のランク計算では、指定された階層内の一連の関連メンバーが検索され、指定されたメジャー値の順序に基づいてすべての関連メンバーがランク付けされ、その結果内で現在のメンバーのランクが戻されます。関連メンバーは、現在のメンバーと同じレベルのメンバーのセットです。オプションで階層関係によってセットを制限できますが、セットには必ず現在のメンバーが含まれます。メジャー値の順序付けはrank_clauseのcalc_meas_order_by_clauseによって決定されます。

rank_function_name

各階層ランク付けファンクションにより、calc_meas_order_by_clauseに基づき、1から始まる順序番号が各関連メンバーに割り当てられます。同じメジャー値の処理方法はファクションによって異なります。

各種ファンクションとそれぞれの相違点は次のとおりです。

- RANKは、同じメジャー値に同じランクを割り当てます。一連の結合された値の後のランクは、結合された値と結合された順序値を加算した数値です。そのため、順序付けは連続した番号にならないことがあります。
- DENSE_RANKは、同じメジャー値に同じ最小ランクを割り当てます。一連の結合された値の後のランクは、必ず結合された値に1を加算した数値になります。そのため、順序付けは必ず連続した番号になります。
- AVERAGE_RANK,では、同じ値には同じ平均ランクが割り当てられます。平均ランク値の後の次の値は、同じ値に1を加算して、その合計を2で除算し、平均ランク値を加算した数値です。たとえば、4、5、10、5、7という5つの一連の値の場合、AVERAGE_RANKは1、1.5、1.5、3、4を戻します。2、12、10、12、17、12という一連の値の場合は、戻されるランクは1、2、3、3、3、5です。
- ROW_NUMBERは、階層メンバー間で連続した一意の値を割り当てます。calc_meas_order_by_clauseの結果が等しい値になる場合、その結果は非決定的になります。

rank_clause

rank_clauseでは、現在のメンバーに関連する階層メンバーの範囲を特定します。範囲は、現在のメンバーと同じレベルにあるメンバーのサブセットです。サブセットはWITHIN句から決定されます。

WITHIN句の有効な値は次のとおりです。

- WITHIN LEVELは、関連メンバーが現在のレベルのすべてのメンバーであることを指定します。これは、WITHINキーワードが指定されない場合のデフォルトのサブセットです。
- WITHIN PARENTは、すべての関連メンバーが現在のメンバーと親を共有することを指定します。
- WITHIN ANCESTOR AT LEVELは、関連メンバーが、指定されたレベルで祖先(または自体)を現在のメンバーと共有する現在のレベルのすべてのメンバーであることを指定します。

share_of_expression

share_of_expression式では、現在のコンテキストの式の値と、関連するコンテキストの式の値の比率を計算します。この式は、現在のコンテキストおよび関連するコンテキストで評価されるcalc_meas_expressionです。share_clauseの指定により、使用する関連するコンテキストが決まります。

share_clause

share_clauseでは、指定された階層のメンバーを関連するメンバーに設定することによって、外部コンテキストを変更します。

share句のパラメータは次のとおりです。

- HIERARCHY hierarchy_refでは、share_of_expression計算の外部コンテキストである階層の名前を指定します。
- PARENTでは、関連するメンバーが現在のメンバーの親であることを指定します。
- LEVEL level_refでは、関連するメンバーが、階層の指定されたレベルの現在のメンバーの祖先(またはメンバー自体)であることを指定します。現在のメンバーが、指定されたレベルよりも上にある場合は、share式に対してNULLが戻されます。階層に、このレベルがない場合は、エラーが発生します。
- MEMBER member_expressionでは、関連するメンバーが、現在のコンテキストでmember_expressionを評価した後に戻されるメンバーであることを指定します。指定されたメンバーの値がNULLの場合、share式に対してNULLが戻されます。

member_expression

member_expressionは、指定された階層のメンバーと評価されます。階層は、常に、外部式(構文によって適用)から決定できます。member_expressionは、次のいずれかになります。

- level_member_literalは、階層メンバーと評価される式です。
- hier_navigation_exprは、階層のあるメンバーを別のメンバーに関連付ける式です。
- CURRENT MEMBERでは、外部コンテキストで決定された階層のメンバーを指定します。
- NULLは、存在しないメンバーを指定する方法です。
- ALLでは、すべての階層の最上位の単一メンバーを指定します。

level_member_literal

level_member_literalは、階層の単一メンバーに変換する式です。この式には、レベルの名前および1つ以上のメンバー・キーが含まれます。メンバー・キーは位置または名前で識別できます。コンテキストの階層に、指定されたレベルがない場合は、エラーが発生します。

pos_member_keys

member_key_expr式は、メンバーのキー値に解決されます。位置で指定する場合、キーのすべてのコンポーネントは、ALL_HIER_LEVEL_ID_ATTRSディクショナリ・ビュー内の順序で指定する必要があります。指定されたレベルが子レベルによって決定されない階層では、そのようなすべての子レベルのすべてのメンバー・キー値を、現在のレベルのメンバー・キーより前に指定する必要があります。重複するキー・コンポーネントは、初めて出現したときにのみ指定されます。

主キーは、level_member_literalがpos_member_keys句で指定された場合に使用されます。代替キーを参照するには、named_member_keys句を使用します。

named_member_keys

member_key_expr式は、メンバーのキー値に解決されます。attr_nameパラメータは、属性名の識別子です。すべての属性名が、指定されたレベルのキーまたは代替キーを構成していない場合は、エラーが発生します。

名前で指定する場合、キーのすべてのコンポーネントを指定し、すべてで属性の名前 = 値書式を任意の順序で使用する必要があります。指定されたレベルが子レベルによって決定されない階層では、そのようなすべての子レベルのすべてのメンバー・キー値を指定し、指定された書式も使用する必要があります。重複するキー・コンポーネントは、1回のみ指定されます。

hier_navigation_expression

hier_navigation_expression式は、指定されたメンバーから階層の別のメンバーにナビゲートします。

hier_ancestor_expression

指定されたメンバーから、指定されたレベルまたは深さの祖先メンバー(またはメンバー自体)にナビゲートします。深さは、数値に変換する必要がある式として指定されます。メンバーが、指定されたメンバーより上のレベルまたは深さの場合、またはメンバーがNULLの場合、式の値に対してNULLが戻されます。コンテキストの階層に、指定されたレベルがない場合は、エラーが発生します。

hier_parent_expression

指定されたメンバーから親メンバーにナビゲートします。

hier_lead_lag_expression

コンテキスト階層内で、一定のメンバーの数だけ前後に移動して、指定されたメンバーから関連するメンバーにナビゲートします。HIER_LEADキーワードを指定すると、後続のメンバーが戻されます。HIER_LAGキーワードを指定すると、先行するメンバーが

戻されます。

hier_lead_lag_clause

指定されたメンバーからoffset_exprの数だけ前後にあるメンバーを対象にナビゲートします。レベル内のメンバーの順序は、属性ディメンションの定義で指定されます。

hier_lead_lag_clauseのオプションのパラメータは次のとおりです。

- WITHIN LEVELでは、現在のメンバーと深さが同じメンバー内で、offset_exprメンバー分だけ前後に移動して、関連するメンバーが検索されます。レベル内のメンバーの順序は、属性ディメンションの定義によって決まります。WITHINキーワードとACROSSキーワードのどちらも使用されていない場合は、WITHIN LEVEL操作がデフォルトとなります。
- WITHIN PARENTでは、現在のメンバーと深さが同じメンバー内で、offset_exprメンバー分だけ前後に移動して、関連するメンバーが検索されますが、現在のメンバーと親を共有するメンバーのみが考慮されます。レベル内のメンバーの順序は、属性ディメンションの定義によって決まります。
- WITHIN ACROSS ANCESTOR AT LEVELは、指定されたレベルの現在のメンバーの祖先(またはメンバー自体)までナビゲートして関連するメンバーを検索し、その親内の各祖先メンバー(メンバー自体を含む)の位置を示します。祖先メンバーが見つかると、祖先メンバーと深さが同じメンバー内で、offset_exprメンバー分前後にナビゲーションが移動します。

  関連する祖先が見つかった後、ナビゲーションはそのメンバーから階層を逆方向に移動し、上に向かって記録された(逆順)親内の位置を照合します。親内での位置は、POSITION FROM BEGINNINGとPOSITION FROM ENDのいずれが指定されているかに従い、最初の子または最後の子からオフセットされます。デフォルトは、POSITION FROM BEGINNINGです。レベル内のメンバーの順序は、属性ディメンションの定義によって決まります。

qdr_expression

qdr_expressionは、新しいコンテキストで指定されたcalc_meas_expressionを評価し、階層メンバーを新しい値に設定する修飾データ参照です。

qualifier

修飾子は、指定された階層のメンバーを、member_expressionの評価の結果のメンバーに設定して、外部コンテキストを変更します。member_expressionがNULLの場合、qdr_expressionの選択の結果はNULLです。

av_hier_expression

av_hier_expressionでは、関連メンバーの属性値を特定するために階層ナビゲーションを実行します。av_hier_expressionは最上位の式にすることができますが、hier_navigation_expressionはmember_expression引数としてのみ使用できます。

たとえば次の問合せでは、HIER_MEMBER__NAMEはav_hier_expression、HIER_PARENTはhier_navigation_expressionです。

```
HIER_MEMBER_NAME(HIER_PARENT(CURRENT MEMBER) WITHIN HIERARCHY product_hier))
```

hier_function_name

hier_function_name値は次のとおりです。

- HIER_CAPTIONは、階層内の関連メンバーのキャプションを戻します。
- HIER_DEPTHは、階層内の関連メンバーとすべてのメンバーの間にある祖先の数から1を引いた数を戻します。すべて

のメンバーの深さは、0です。

- HIER_DESCRIPTIONは、階層内の関連メンバーの説明を戻します。
- HIER_LEVELは、階層で関連メンバーが属するレベルの名前を文字列値として戻します。
- HIER_MEMBER_NAMEは、階層内の関連メンバーのメンバー名を戻します。
- HIER_MEMBER_UNIQUE_NAMEは、階層内の関連メンバーの一意のメンバー名を戻します。

分析ビュー式の例

このトピックには、分析ビューのMEASURES句、およびSELECT文のADD MEASURES句で定義された計算済メジャーを示す例が記載されています。

次の例があります。

- LAG式の例
- ウィンドウ式の例
- SHARE OF式の例
- QDR式の例
- RANKファンクションを使用して追加されたメジャーの例

その他の例は、SQL LiveのWebサイト(https://livesql.oracle.com/apex/livesql/file/index.html)にある分析ビューのチュートリアルを参照してください。

LAG式の例

これらの計算済メジャーは、LAG操作とは異なります。

```
-- These calculated measures are from the measures_clause of the
-- sales_av analytic view.
MEASURES
 (sales FACT sales,                        -- A base measure
  units FACT units,                        -- A base measure
  sales_prior_period AS                    -- Calculated measures
    (LAG(sales) OVER (HIERARCHY time_hier OFFSET 1)),
  sales_year_ago AS
    (LAG(sales) OVER (HIERARCHY time_hier OFFSET 1
     ACROSS ANCESTOR AT LEVEL year)),
  chg_sales_year_ago AS
    (LAG_DIFF(sales) OVER (HIERARCHY time_hier OFFSET 1
     ACROSS ANCESTOR AT LEVEL year)),
  pct_chg_sales_year_ago AS
    (LAG_DIFF_PERCENT(sales) OVER (HIERARCHY time_hier OFFSET 1
     ACROSS ANCESTOR AT LEVEL year)),
  sales_qtr_ago AS
    (LAG(sales) OVER (HIERARCHY time_hier OFFSET 1
     ACROSS ANCESTOR AT LEVEL quarter)),
  chg_sales_qtr_ago AS
    (LAG_DIFF(sales) OVER (HIERARCHY time_hier OFFSET 1
     ACROSS ANCESTOR AT LEVEL quarter)),
  pct_chg_sales_qtr_ago AS
    (LAG_DIFF_PERCENT(sales) OVER (HIERARCHY time_hier OFFSET 1
     ACROSS ANCESTOR AT LEVEL quarter))
 )
```

ウィンドウ式の例

この計算済メジャーは、ウィンドウ操作を使用します。

```
MEASURES
 (sales FACT sales,
  units FACT units,
  sales_qtd AS
    (SUM(sales) OVER (HIERARCHY time_hier
     BETWEEN UNBOUNDED PRECEDING AND CURRENT MEMBER
     WITHIN ANCESTOR AT LEVEL QUARTER)),
  sales_ytd AS
    (SUM(sales) OVER (HIERARCHY time_hier
     BETWEEN UNBOUNDED PRECEDING AND CURRENT MEMBER
     WITHIN ANCESTOR AT LEVEL YEAR))
 )
```

SHARE OF式の例

これらの計算済メジャーは、SHARE OF式を使用します。

```
MEASURES
 (sales FACT sales,
  units FACT units,
 sales_shr_parent_prod AS
   (SHARE_OF(sales HIERARCHY product_hier PARENT)),
 sales_shr_parent_geog AS
   (SHARE_OF(sales HIERARCHY geography_hier PARENT)),
 sales_shr_region AS
   (SHARE_OF(sales HIERARCHY geography_hier LEVEL REGION))
 )
```

QDR式の例

これらの計算済メジャーは、QUALIFYキーワードを使用して、修飾データ参照式を指定します。

```
MEASURES
 (sales FACT sales,
  units FACT units,
  sales_2011 AS
    (QUALIFY (sales, time_hier = year['11'])),
  sales_pct_chg_2011 AS
    ((sales - (QUALIFY (sales, time_hier = year['11']))) /
    (QUALIFY (sales, time_hier = year['11'])))
 )
```

RANKファンクションを使用して追加されたメジャーの例

この例では、units_geog_rank_levelメジャーにRANKファンクションを使用して、レベル内の地理階層メンバーを単位に基づいてランク付けします。

```
SELECT geography_hier.member_name AS "Region",
       units AS "Units",
       units_geog_rank_level AS "Rank"
  FROM ANALYTIC VIEW (
    USING sales_av HIERARCHIES (geography_hier)
    ADD MEASURES (
      units_geog_rank_level AS (
        RANK() OVER (
          HIERARCHY geography_hier
          ORDER BY units desc nulls last
          WITHIN LEVEL))
    )
  )
  WHERE geography_hier.level_name IN ('REGION')
  ORDER BY units_geog_rank_level;
```

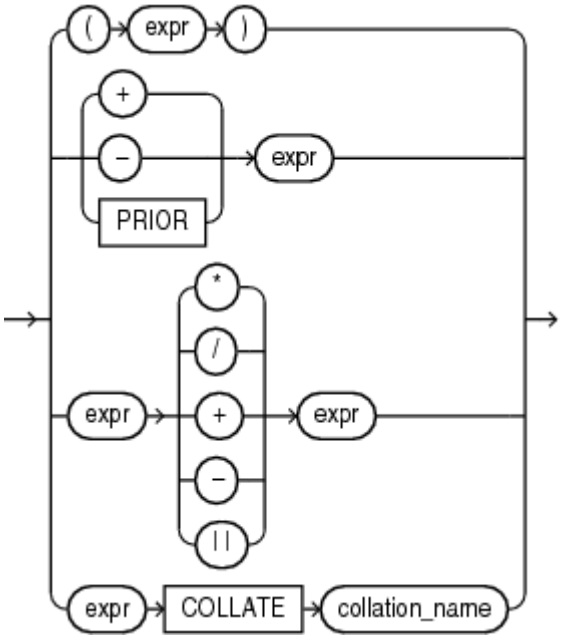
問合せの結果は次のとおりです。

```
Regions            Units  Rank
-------------  ---------  ----
Asia            56017849     1
South America   23904155     2
North America   20523698     3
Africa          12608308     4
Europe           8666520     5
Oceania           427664     6
```

複合式

複合式は、その他の式の組合せを指定します。

compound_expression::=



組込みファンクションを式として使用できます(ファンクション式を参照)。ただし、複合式では、ファンクションの組合せによっては、適切でないものや拒否されるものもあります。たとえば、LENGTHファンクションは集計ファンクション内では使用できません。

PRIOR演算子は、階層問合せのCONNECT BY句で使用されます。

COLLATE演算子により、式の照合が決定されます。この演算子は、データベースが標準の照合導出ルールを使用して式について導出した照合を上書きします。

関連項目:

- 演算子の優先順位
- 階層問合せ
- COLLATE演算子

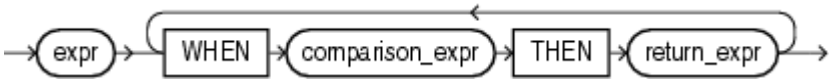
有効な複合式の例を次に示します。

```
('CLARK' || 'SMITH')
LENGTH('MOOSE') * 57
SQRT(144) + 72
my_fun(TO_CHAR(sysdate,'DD-MMM-YY'))
name COLLATE BINARY_CI
```

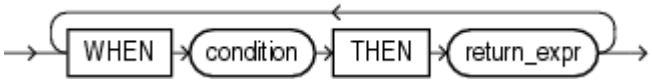
CASE式

CASE式を使用すると、プロシージャを起動せずに、SQL文でIF ... THEN ... ELSE論理を使用できます。構文は次のとおりです。

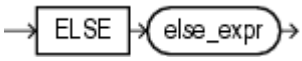
simple_case_expression::=



searched_case_expression::=



else_clause::=



単純CASE式では、Oracle Databaseは、exprとcomparison_exprが一致する最初のWHEN ... THENの組合せを検索し、return_exprを戻します。WHEN ... THENの組合せが条件に一致せず、ELSE句が存在する場合、Oracleはelse_exprを戻します。それ以外の場合、OracleはNULLを戻します。

検索CASE式では、Oracleは、conditionが真である項目を左から右へ検索し、return_exprを戻します。真であるconditionがなく、ELSE句が存在する場合、Oracleはelse_exprを戻します。それ以外の場合、OracleはNULLを戻します。

Oracle Databaseでは、短絡評価を使用します。単純CASE式では、データベースは、comparison_expr値のいずれかとexprを比較する前にすべてのcomparison_expr値を評価するのではなく、各comparison_expr値とexprを比較する前にのみ、各comparison_expr値を評価します。その結果、exprと等しいcomparison_exprが見つかると、Oracleはその後のcomparison_exprを評価しません。検索CASE式の場合、データベースは各conditionを評価してこれが真であるかどうかを判断します。ただし、あるconditionが真の場合は、次のconditionを評価しません。

単純CASE式では、exprおよびすべてのcomparison_expr値は、同じデータ型(CHAR、VARCHAR2、NCHAR、NVARCHAR2、NUMBER、BINARY_FLOATまたはBINARY_DOUBLE)であるか、またはすべて数値データ型である必要があります。すべての式が数値データ型の場合、Oracleは、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

単純および検索CASE式では、すべてのreturn_exprは、同じデータ型(CHAR、VARCHAR2、NCHAR、NVARCHAR2、NUMBER、BINARY_FLOATまたはBINARY_DOUBLE)であるか、またはすべて数値データ型である必要があります。すべての戻り式が数値データ型の場合、Oracleは、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

1つのCASE式では、引数の最大数は65535です。単純CASE式の初期式やオプションのELSE式を含むすべての式が、この

上限の対象となります。WHEN ... THENの各組は、2つの引数としてカウントします。この上限を超えないように、CASE式をネストし、return_expr自体をCASE式にできます。

単純なCASE式によって実行される比較は、比較される引数に文字データ型(CHAR、VARCHAR2、NCHARまたはNVARCHAR2)が含まれる場合、照合依存です。照合決定ルールによって、使用する照合が決まります。

関連項目:

- 暗黙的な変換の詳細は、表2-8を参照してください。
- CASE式の照合の導出および決定のルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。
- 数値の優先順位の詳細は、数値の優先順位を参照してください。
- その他のCASE式の書式については、「COALESCE」および「NULLIF」を参照してください。
- CASE式の様々な書式を使用した例は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

単純なCASEの例

次の文は、サンプル表oe.customersのそれぞれの顧客について、クレジット利用限度額を、$100の場合は「Low」、$5000の場合は「High」、それ以外の場合は「Medium」で表示します。

```
SELECT cust_last_name,
   CASE credit_limit WHEN 100 THEN 'Low'
   WHEN 5000 THEN 'High'
   ELSE 'Medium' END AS credit
   FROM customers
   ORDER BY cust_last_name, credit;
CUST_LAST_NAME       CREDIT
-------------------- ------
Adjani               Medium
Adjani               Medium
Alexander            Medium
Alexander            Medium
Altman               High
Altman               Medium
. . .
```

検索CASEの例

次の文は、$2000を最少額の給与として、サンプル表oe.employeesの従業員の給与の平均を検出します。

```
SELECT AVG(CASE WHEN e.salary > 2000 THEN e.salary
   ELSE 2000 END) "Average Salary" FROM employees e;
Average Salary
--------------
    6461.68224
```

列式

後述の構文図でcolumn_expressionで示される列式は、exprの書式の一部です。列式は単純式、複合式、ファンクション式または式リストにできますが、列式に含めることができるのは、次の書式の式にかぎられます。

- 対象とする表(作成、変更または索引作成される表)の列
- 定数(文字列または数値)
- DETERMINISTICファンクション ― SQL組込みファンクションまたはユーザー定義ファンクションのいずれか

これ以外に、この章で説明されている書式で使用できるものはありません。また、PRIORキーワードを使用した複合式および集計ファンクションはサポートされません。

列式は、次の目的で使用できます。

- ファンクション索引を作成する場合。
- 明示的または暗黙的に仮想列を定義する場合。仮想列を定義する際、column_expressionの定義では、現行の文または以前の文ですでに定義されている対象の表の列のみを参照する必要があります。

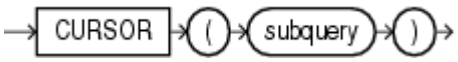
列式を組み合せた要素は、決定的である必要があります。つまり、同じ入力値のセットによって、同じ出力値のセットが戻される必要があります。

関連項目:

これらのexprの書式の詳細は、単純式、複合式、ファンクション式および式のリストを参照してください。

CURSOR式

CURSOR式は、ネストしたカーソルを戻します。この式の書式は、PL/SQLのREF CURSORと同じで、REF CURSOR引数としてファンクションに渡すことができます。



カーソル式が評価されるときに、ネステッド・カーソルは暗黙的にオープンされます。たとえば、カーソル式がSELECT構文のリストにある場合、問合せによってフェッチされた各行に対して、ネステッド・カーソルがオープンされます。ネステッド・カーソルは、次の場合にのみクローズされます。

- ユーザーによって明示的にクローズされたとき
- 親カーソルが再実行されたとき
- 親カーソルがクローズされたとき
- 親カーソルが取り消されたとき
- 親カーソルの1つでのフェッチ時にエラーが発生したとき(クリーンアップの一部としてクローズされる)

CURSOR式の制限事項

CURSOR式には、次の制限事項があります。

- 囲まれる文がSELECT文以外の文である場合、ネステッド・カーソルは、プロシージャのREF CURSOR引数としてのみ表示されます。
- 囲まれる文がSELECT文である場合、ネステッド・カーソルは、問合せ指定の一番外側のSELECT構文のリスト、または別のネステッド・カーソルの一番外側のSELECT構文のリストに表示されます。
- ネステッド・カーソルはビューに表示できません。
- ネステッド・カーソルに対して、BIND操作およびEXECUTE操作は実行できません。

例

次の例は、問合せの選択リストでのCURSOR式の使用方法を示しています。

```
SELECT department_name, CURSOR(SELECT salary, commission_pct
   FROM employees e
   WHERE e.department_id = d.department_id)
   FROM departments d
   ORDER BY department_name;
```

ファンクションの引数としてのCURSORの使用方法について、次に例を示します。例では、まず、サンプル・スキーマOEにREF CURSOR引数を受け入れるファンクションを作成します。(イタリック体は、PL/SQLファンクションの本体です。)

```
CREATE FUNCTION f(cur SYS_REFCURSOR, mgr_hiredate DATE)
   RETURN NUMBER IS
   emp_hiredate DATE;
   before number :=0;
   after number:=0;
begin
  loop
    fetch cur into emp_hiredate;
    exit when cur%NOTFOUND;
    if emp_hiredate > mgr_hiredate then
      after:=after+1;
```

```
    else
      before:=before+1;
    end if;
  end loop;
  close cur;
  if before > after then
    return 1;
  else
    return 0;
  end if;
end;
/
```

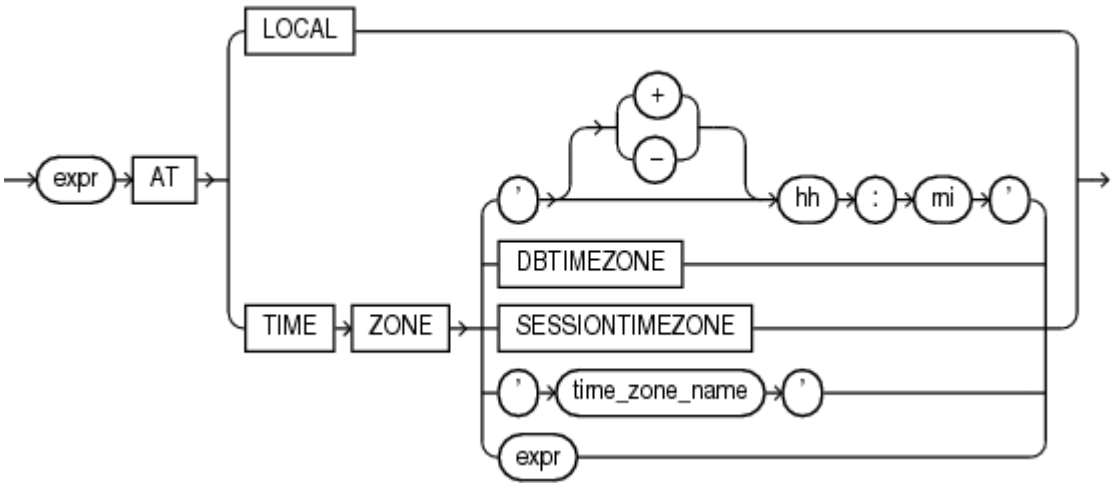
ファンクションには、カーソルおよび日付を指定できます。ファンクションは、カーソルが日付セットを戻す問合せであることを想定します。次の問合せでは、ファンクションを使用して、サンプル表employeesから、ほとんどの従業員がマネージャよりも前に雇用されているマネージャを検索します。

```
SELECT e1.last_name FROM employees e1
   WHERE f(
   CURSOR(SELECT e2.hire_date FROM employees e2
   WHERE e1.employee_id = e2.manager_id),
   e1.hire_date) = 1
   ORDER BY last_name;

LAST_NAME
-------------------------
Cambrault
Higgins
Hunold
Kochhar
Mourgos
Zlotkey
```

日時式

日時式は、日時データ型の値を戻します。

datetime_expression::=



初期のexprは、データ型TIMESTAMP、TIMESTAMP WITH TIME ZONEまたはTIMESTAMP WITH LOCAL TIME ZONEの値に評価される任意の式(スカラー副問合せ式を除く)です。DATEデータ型はサポートされません。このexpr自体がdatetime_expressionである場合は、カッコで囲む必要があります。

表2-5で定義される規則に従って、日時および期間を組み合せることができます。日時値を生成する3つの組合せは、日時式で有効です。

AT LOCALを指定すると、Oracleは現行のセッションのタイムゾーンを使用します。

AT TIME ZONEの設定は、次のように解析されます。

- 文字列'[+|-]hh:mi 'は、タイムゾーンをUTCからのオフセットとして指定します。hhには時間数を指定します。miには分数を指定します。
- DBTIMEZONE: Oracleは、データベースの作成中に(明示的またはデフォルトで)構築されたデータベース・タイムゾーンを使用します。
- SESSIONTIMEZONE: デフォルトまたは最新のALTER SESSION文で設定されたセッションのタイムゾーンが使用されます。
- time_zone_name: time_zone_nameで指定されたタイムゾーンのdatetime_value_exprが戻されます。有効なタイムゾーン地域名を表示するには、V$TIMEZONE_NAMES動的パフォーマンス・ビューに問合せを実行してください。

> ノート:
>
> 夏時間機能には、タイムゾーン地域名が必要です。この名前は、大小 2 つのタイムゾーン・ファイルに格納されます。これらのファイルのうち、使用する環境および使用する Oracle Database のリリースに応じて、いずれか一方がデフォルトのファイルになります。タイムゾーン・ファイルおよびタイムゾーン名の詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

関連項目:

- 両方のファイル内のすべてのタイムゾーン地域名のリストは、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。
- 動的パフォーマンス・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。
- `expr`: `expr`が有効なタイムゾーン書式で文字列を戻す場合、Oracleは、そのタイムゾーンで入力を戻します。そうでない場合は、エラーが戻ります。

例

次の例は、タイムゾーンの日時の値を別のタイムゾーンに変換します。

```
SELECT FROM_TZ(CAST(TO_DATE('1999-12-01 11:00:00',
      'YYYY-MM-DD HH:MI:SS') AS TIMESTAMP), 'America/New_York')
   AT TIME ZONE 'America/Los_Angeles' "West Coast Time"
   FROM DUAL;
West Coast Time
---------------------------------------------
01-DEC-99 08.00.00.000000 AM AMERICA/LOS_ANGELES
```

ファンクション式

組込みSQLファンクションまたはユーザー定義ファンクションを式として使用できます。有効な組込みファンクション式の例を次に示します。

```
LENGTH('BLAKE')
ROUND(1234.567*43)
SYSDATE
```

関連項目:

組込みファンクションの詳細は、SQLファンクションおよび集計ファンクションを参照してください。

ユーザー定義ファンクション式は、次のものへのコールを指定します。

- オラクル社が提供するパッケージにあるファンクション(『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照)
- ユーザー定義パッケージにあるファンクションまたはスタンドアロン・ユーザー定義ファンクション(ユーザー定義ファンクションを参照)
- ユーザー定義ファンクションまたは演算子(CREATE OPERATOR、CREATE FUNCTIONおよび『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照)

有効なユーザー定義ファンクション式の例を次に示します。

```
circle_area(radius)
payroll.tax_rate(empno)
hr.employees.comm_pct@remote(dependents, empno)
DBMS_LOB.getlength(column_name)
my_function(a_column)
```

式として使用されるユーザー定義ファンクションでは、位置表記法、名前付き表記法および複合表記法がサポートされています。たとえば、次の表記はすべて正しい表記です。

```
CALL my_function(arg1 => 3, arg2 => 4) ...

CALL my_function(3, 4) ...
CALL my_function(3, arg2 => 4) ...
```

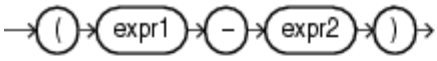
ユーザー定義ファンクション式の制限事項

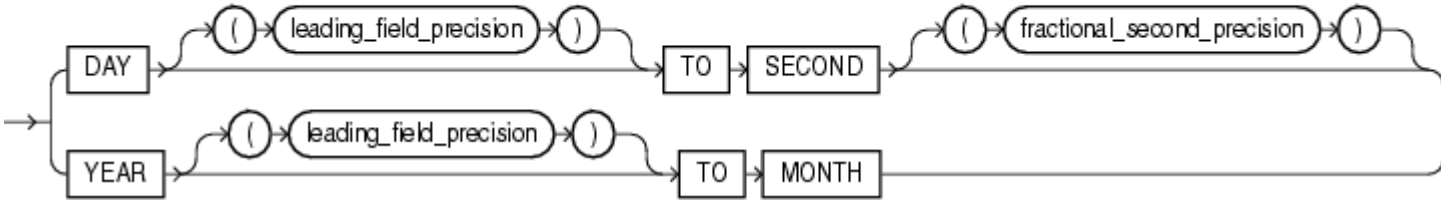
リモートのファンクションおよびプロシージャに、オブジェクト型またはXMLTypeの引数を渡すことはできません。

期間式

期間式は、INTERVAL YEAR TO MONTHまたはINTERVAL DAY TO SECONDの値を戻します。

interval_expression::=





式expr1およびexpr2は、データ型DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONEまたはTIMESTAMP WITH LOCAL TIME ZONEの値に評価される任意の式にすることが可能です。

表2-5で定義される規則に従って、日時および期間を組み合せることができます。期間の値を戻す6つの組合せは、期間式で有効です。

leading_field_precisionおよびfractional_second_precisionは、0から9の任意の整数です。DAYまたはYEARのいずれかでleading_field_precisionを省略すると、デフォルト値である2が使用されます。秒でfractional_second_precisionを省略すると、データベースはデフォルト値である6を使用します。問合せで戻された値にデフォルトの精度を超える桁数が含まれる場合、Oracle Databaseはエラーを戻します。したがって、問合せで戻されるとわかっている値以上の精度を指定することをお薦めします。

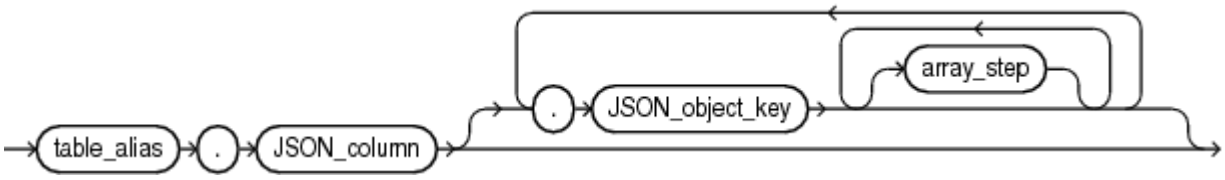
たとえば、次の文は、システム・タイムスタンプ(日時の値)からサンプル表ordersのorder_date列の値(別の日時の値)を減算して、期間値の式を戻します。一番古い注文が発注されたのが何日前かわからないため、DAYの先行フィールド精度に最大値である9を指定します。

```
SELECT (SYSTIMESTAMP - order_date) DAY(9) TO SECOND FROM orders
   WHERE order_id = 2458;
```

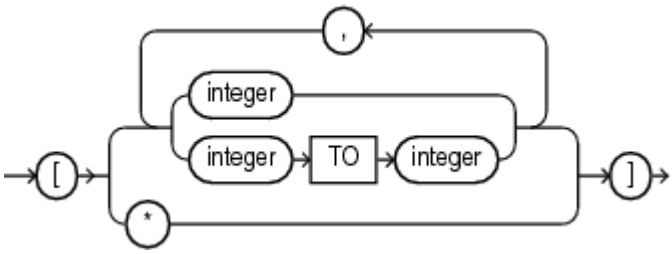
JSONオブジェクト・アクセス式

JavaScript Object Notation(JSON)データの列を問い合せる場合のみ、JSONオブジェクト・アクセス式を使用します。そのデータ内の1つ以上のJSON値を含む文字列を戻します。このタイプの式の構文は、ドット表記法構文といいます。

JSON_object_access_expr::=



array_step::=



- table_aliasでは、JSONデータの列を含む表の別名を指定します。この表の別名が必要であり、SQL文で表に割り当てる必要があります。
- JSON_columnでは、JSONデータの列の名前を指定します。列のデータ型はVARCHAR2、CLOBまたはBLOBである必要があり、IS JSON CHECK制約を列に定義する必要があります。
- 1つ以上のJSONオブジェクト・キーをオプションで指定できます。オブジェクト・キーによって、JSONデータの特定のJSON値を特定できます。最初のJSON_object_keyは、JSONデータのトップレベルのオブジェクト・メンバーのキー(プロパティ)名と大/小文字を区別して一致する必要があります。オブジェクト・メンバーの値が別のJSONオブジェクトである場合、オブジェクトのメンバーのキー名と一致する2番目のJSON_object_keyを指定でき、以降も同様です。これらの反復中にJSON配列が検出され、array_stepを指定しない場合、配列が暗黙的にアンラップされ、配列の要素がJSON_object_keyを使用して評価されます。
- JSON値が配列の場合、オプションで1つ以上のarray_step句を指定できます。これにより、JSON配列の特定の要素にアクセスできます。
  - integerを使用して、JSON配列の索引integerの要素を指定します。integer TO integerを使用して、2つのinteger索引値の間(これらの値を含む)の要素の範囲を指定します。指定された要素が評価されるJSON配列にある場合、配列ステップはそれらの要素と一致する結果になります。それ以外の場合、一致しない配列ステップになります。JSON配列の最初の要素は索引0です。
  - ワイルド・カード記号であるアスタリスク(*)を使用して、JSON配列のすべての要素を指定します。評価されるJSON配列に少なくとも1つの要素が含まれる場合、JSON配列のすべての要素が一致する配列ステップになります。それ以外の場合、一致しない配列ステップになります。

JSONオブジェクト・アクセス式は、次のようなターゲットJSON値を含むデータ型VARCHAR2(4000)の文字列を戻します。

- 単一のターゲット値の場合、文字列には、JSONスカラー値、オブジェクト、配列に関係なくその値が含まれます。
- 複数のターゲット値の場合、文字列には、要素がそれらの値であるJSON配列が含まれます。

JSON_object_keyを省略すると、この式は全体のJSONデータを含む文字列を戻します。この場合、文字列は、問い合せるJSONデータの列と同じデータ型です。

JSONオブジェクト・アクセス式は、4KBを超える値を返せません。値がこの制限を超えると、式はnullを返します。実際の値を取得するには、かわりにJSON_QUERY関数またはJSON_VALUE関数を使用して、RETURNING句に適切な戻り型を指定します。

JSONオブジェクト・アクセス式の照合導出ルールは、JSON_QUERYファンクションのものと同じです。

関連項目:

JSON_QUERYファンクションの照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、JSONドキュメントを含む表の作成: 例で作成されるj_purchaseorder表を使用します。この表には、po_documentと呼ばれるJSONデータの列が含まれます。これらの例では、列po_documentからJSON値を戻します。

次の文は、キー名PONumberを使用したプロパティの値を戻します。

```
SELECT po.po_document.PONumber
  FROM j_purchaseorder po;
PONumber
--------
1600
```

次の文は、値がJSONオブジェクトであるキー名ShippingInstructionsを使用したプロパティを最初に対象とします。この文は、そのオブジェクト内のキー名Phoneを使用したプロパティを対象とします。この文は、JSON配列であるPhoneの値を戻します。

```
SELECT po.po_document.ShippingInstructions.Phone
  FROM j_purchaseorder po;

SHIPPINGINSTRUCTIONS
--------------------------------------------------------------------------------
[{"type":"Office","number":"909-555-7307"},{"type":"Mobile","number":"415-555-1234"}]
```

次の文は、値がJSON配列であるキー名LineItemsを使用したプロパティを最初に対象とします。この式は配列を暗黙的にアンラップし、JSONオブジェクトである要素を評価します。次に、この文は、アンラップされたオブジェクト内のキー名Partを使用したプロパティを対象とし、2つのオブジェクトを確認します。この文は、それらのオブジェクト内のキー名Descriptionを使用したプロパティを対象とし、文字列値を確認します。複数の値が戻されるため、値はJSON配列の要素として戻されます。

```
SELECT po.po_document.LineItems.Part.Description
  FROM j_purchaseorder po;

LINEITEMS
----------------------------------
[One Magic Christmas,Lethal Weapon]
```

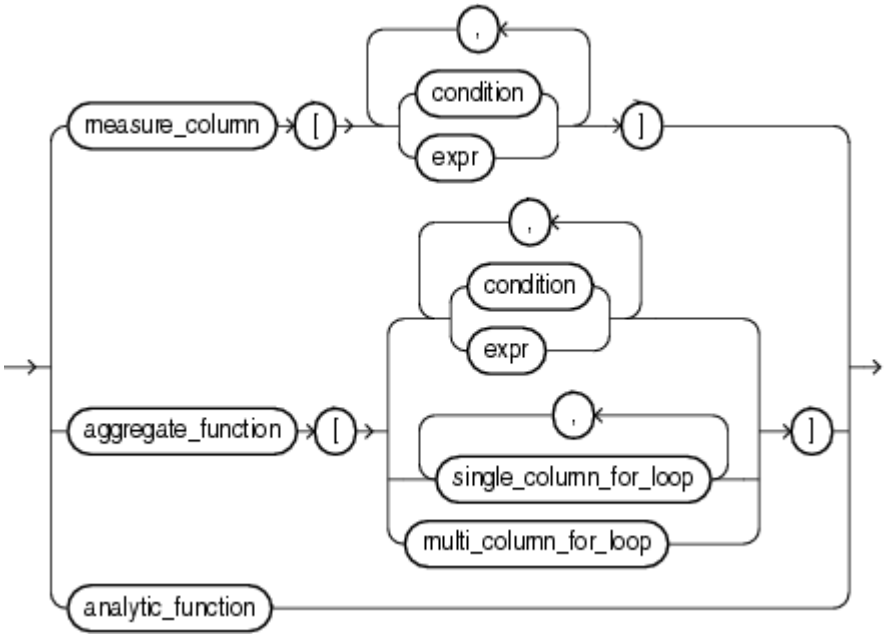
関連項目:

ドット表記法構文を使用したJSONデータの問合せの詳細は、Oracle Database JSON開発者ガイドを参照してください。

モデル式

モデル式は、SELECT文のmodel_clauseでのみ、さらにモデル・ルールの右側でのみ使用されます。モデル式は、model_clauseで事前定義されたメジャー列のセルに値を戻します。詳細は、「model_clause」を参照してください。

model_expression::=



モデル式でメジャー列を指定する場合、指定する任意の条件と式は単一の値に変換される必要があります。

モデル式で集計ファンクションを指定する場合、ファンクションの引数はmodel_clauseで事前定義したメジャー列です。集計ファンクションは、モデル・ルールの右側でのみ使用できます。

モデル・ルールの右側で分析ファンクションを指定すると、model_clauseに複雑な計算を直接表記することができます。モデル式で分析ファンクションを使用するときには、次の制限事項が適用されます。

- 分析ファンクションは、UPDATEルールのみで使用できます。
- モデル・ルールの左側にFORループまたはORDER BY句が含まれている場合は、モデル・ルールの右側で分析ファンクションを指定できません。
- 分析ファンクションのOVER句の引数に集計を含めることはできません。
- 分析ファンクションのOVER句の前の引数にセル参照を含めることはできません。

関連項目:

モデル・ルールの右側で分析ファンクションを使用する例は、MODEL句: 例を参照してください。

expr自体がモデル式である場合、ネストしたセル参照と呼ばれます。ネストしたセル参照には、次の制限事項が適用されます。

- ネストは1レベルのみ可能です。
- ネストしたセル参照は、単一セル参照である必要があります。
- model_rules_clauseにAUTOMATIC ORDERが指定されているとき、ネストしたセル参照で使用されるメジャー

が静的なままの場合のみ、ネストしたセル参照をモデル・ルールの左側で使用できます。

後述するモデル式は、次に示すSELECT文のmodel_clauseに基づいています。

```
SELECT country,prod,year,s
  FROM sales_view_ref
  MODEL
    PARTITION BY (country)
    DIMENSION BY (prod, year)
    MEASURES (sale s)
    IGNORE NAV
    UNIQUE DIMENSION
    RULES UPSERT SEQUENTIAL ORDER
    (
      s[prod='Mouse Pad', year=2000] =
        s['Mouse Pad', 1998] + s['Mouse Pad', 1999],
      s['Standard Mouse', 2001] = s['Standard Mouse', 2000]
    )
  ORDER BY country, prod, year;
```

次のモデル式は、記号表記法を使用して単一セル参照を表しています。この式は、2000年のマウス・パッドの売上を示します。

```
s[prod='Mouse Pad',year=2000]
```

次のモデル式は、CVファンクションを使用して、位置表記法を使用する複数セル参照を表しています。この式は、prodディメンション列の現在値の2001年における売上を示します。

```
s[CV(prod), 2001]
```

次のモデル式は、集計ファンクションを表しています。この式は、yearディメンション列の現在の値-2からyearディメンション列の現在の値-1までの年のマウス・パッドの売上合計を示します。

```
SUM(s)['Mouse Pad',year BETWEEN CV()-2 AND CV()-1]
```

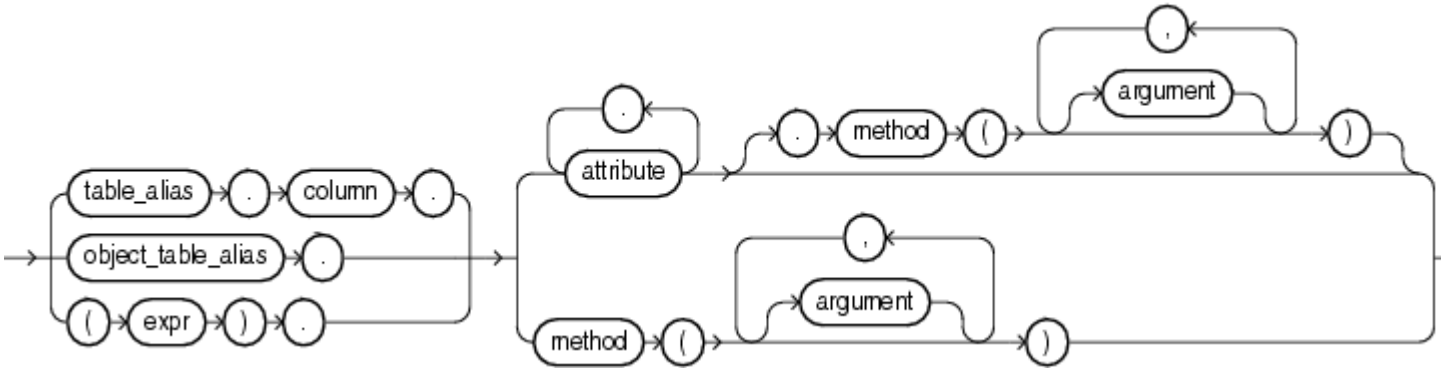
関連項目:

「CV」および「model_clause」を参照してください。

オブジェクト・アクセス式

オブジェクト・アクセス式は、属性の参照およびメソッドの起動を指定します。

object_access_expression::=



columnパラメータはオブジェクトまたはREF列です。exprを指定する場合、オブジェクト型に変換する必要があります。

ある型のメンバー・ファンクションがSQL文のコンテキストでコールされると、SELF引数がNULLの場合に、OracleはNULLを戻し、ファンクションは起動されません。

例

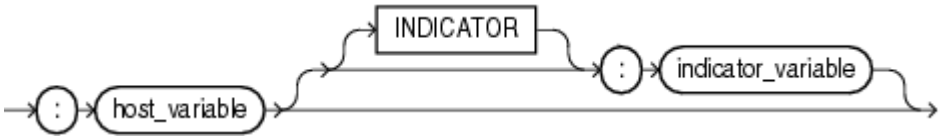
次の例では、サンプル・オブジェクト型oe.order_item_typに基づく表を作成し、オブジェクト列属性から更新および選択する方法を示します。

```
CREATE TABLE short_orders (
   sales_rep VARCHAR2(25), item order_item_typ);
UPDATE short_orders s SET sales_rep = 'Unassigned';
SELECT o.item.line_item_id, o.item.quantity FROM short_orders o;
```

プレースホルダ式

プレースホルダ式は、SQL文における位置を指定します。この位置に対して、第三世代言語のバインド変数によって値が指定されます。プレースホルダ式には、オプションで標識変数を指定できます。この書式の式は、埋込みSQL文またはOracle Call Interface(OCI)プログラムで処理されるSQL文のみで指定できます。

placeholder_expression::=



有効なプレースホルダ式の例を次に示します。

```
:employee_name INDICATOR :employee_name_indicator_var
:department_location
```

関連項目:

文字データ型を含むプレースホルダ式の照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

スカラー副問合せ式

スカラー副問合せ式は、1つの行から1つの列値のみを戻す副問合せです。スカラー副問合せ式の値は、副問合せのSELECT構文リスト項目の値です。副問合せが0行を戻す場合、スカラー副問合せ式の値はNULLです。副問合せが2つ以上の行を戻す場合、Oracleはエラーを戻します。

スカラー副問合せ式は、式(expr)をコールするほとんどの構文で使用できます。すべての場合、スカラー副問合せは、その構文の場所がすでにカッコ内であっても(組込みファンクションの引数として使用されている場合など)、独自のカッコで囲む必要があります。

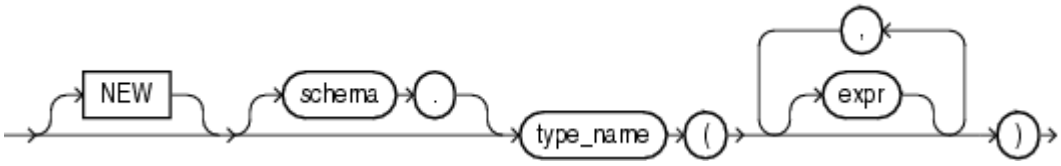
スカラー副問合せは、次の場所では無効です。

- 列のデフォルト値
- クラスタのハッシュ式
- DML文RETURNING句
- ファンクション索引の基礎
- CHECK制約
- GROUP BY句
- CREATE PROFILEなどの問合せに関連しない文

型コンストラクタ式

型コンストラクタ式は、コンストラクタ・メソッドへのコールを指定します。型コンストラクタの引数は、任意の式です。型コンストラクタは、ファンクションが起動されるすべての場所で起動できます。

type_constructor_expression::=



NEWキーワードは、コレクション型ではなくオブジェクト型のコンストラクタに適用されます。これによって、適切なコンストラクタを起動して新しいオブジェクトを作成するようにOracleに指示します。NEWキーワードはオプションですが、指定することをお薦めします。

type_nameがオブジェクト型の場合、式は順序リストで、その最初の引数の値の型がオブジェクト型の最初の属性と一致し、2番目の引数の値の型がオブジェクト型の2番目の属性と一致し、以降同様に続く必要があります。コンストラクタの引数の合計数は、オブジェクト型の属性の合計数と一致する必要があります。

type_nameがVARRAY型またはネストした表型の場合、式のリストには0個以上の引数を含めることができます。引数が0個の場合は、空コレクションの構造であることを示します。それ以外の場合は、各引数が、型がコレクション型の要素型である要素値に対応します。

型コンストラクタの起動の制限事項

型コンストラクタ・メソッドの起動では、オブジェクト型に1000以上の属性がある場合でも、指定できるパラメータの数(expr)は最大で999です。この制限は、コンストラクタがSQLからコールされる場合にのみ適用されます。PL/SQLからコールされる場合には、PL/SQLの制限が適用されます。

関連項目:

コンストラクタ・メソッドの詳細は、『Oracle Databaseオブジェクト・リレーショナル開発者ガイド』を参照してください。型コンストラクタへのコールに関するPL/SQLの制限の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

式の例

次の例では、サンプル・スキーマoeのcust_address_typ型を使用して、コンストラクタ・メソッドへのコールに含まれる式の使用方法を示します(PL/SQLはイタリック体で示しています)。

```
CREATE TYPE address_book_t AS TABLE OF cust_address_typ;
DECLARE
   myaddr cust_address_typ := cust_address_typ(
     '500 Oracle Parkway', 94065, 'Redwood Shores', 'CA','USA');
   alladdr address_book_t := address_book_t();
BEGIN
   INSERT INTO customers VALUES (
      666999, 'Joe', 'Smith', myaddr, NULL, NULL, NULL, NULL,
      NULL, NULL, NULL, NULL, NULL, NULL, NULL);
END;
/
```

副問合せの例

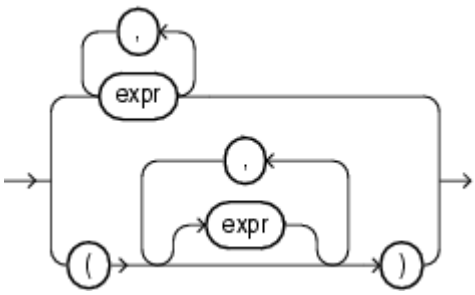
次の例では、サンプル・スキーマoeのwarehouse_typ型を使用して、コンストラクタ・メソッドへのコールに含まれる副問合せの使用方法を示します。

```
CREATE TABLE warehouse_tab OF warehouse_typ;
INSERT INTO warehouse_tab
   VALUES (warehouse_typ(101, 'new_wh', 201));
CREATE TYPE facility_typ AS OBJECT (
   facility_id NUMBER,
   warehouse_ref REF warehouse_typ);

CREATE TABLE buildings (b_id NUMBER, building facility_typ);
INSERT INTO buildings VALUES (10, facility_typ(102,
   (SELECT REF(w) FROM warehouse_tab w
      WHERE warehouse_name = 'new_wh')));
SELECT b.b_id, b.building.facility_id "FAC_ID",
   DEREF(b.building.warehouse_ref) "WH" FROM buildings b;
      B_ID     FAC_ID WH(WAREHOUSE_ID, WAREHOUSE_NAME, LOCATION_ID)
---------- ---------- -----------------------------------------------
        10        102 WAREHOUSE_TYP(101, 'new_wh', 201)
```

式のリスト

式のリストは、その他の式の組合せです。

expression_list::=



式のリストは、比較条件、メンバーシップ条件、および問合せと副問合せのGROUP BY句に指定できます。比較条件またはメンバーシップ条件内の式のリストは、行値コンストラクタまたは行コンストラクタと呼ばれる場合があります。

比較条件およびメンバーシップ条件は、WHERE句の条件として指定します。これらの条件には、カンマで区切られた1つ以上の式、または1つ以上の式の集合(各集合には、カンマで区切られた式が1つ以上含まれる)を含めることができます。式の集合が複数存在する場合は、次のように指定します。

- 各集合をカッコで区切ります。
- 各集合に含まれる式の数を同じにします。
- 各集合内の式の数を、比較条件の演算子の前またはメンバーシップ条件のINキーワードの前にある式の数と同じにします。

カンマで区切られた式のリストには、最大1000個の式を指定できます。カンマで区切られた式の集合のリストには、任意数の式の集合を含めることができますが、各集合に指定できる式は最大1000個です。

次に、条件に含める有効な式のリストを示します。

```
(10, 20, 40)
('SCOTT', 'BLAKE', 'TAYLOR')
( ('Guy', 'Himuro', 'GHIMURO'),('Karen', 'Colmenares', 'KCOLMENA') )
```

3番目の例の場合、各集合の式の数は、条件の最初の部分の式の数と同じである必要があります。たとえば:

```
SELECT * FROM employees
  WHERE (first_name, last_name, email) IN
  (('Guy', 'Himuro', 'GHIMURO'),('Karen', 'Colmenares', 'KCOLMENA'))
```

関連項目:

比較条件およびIN条件を参照してください。

単純なGROUP BY句では、式のリストの書式として、次のいずれかを使用できます。

```
SELECT department_id, MIN(salary) min, MAX(salary) max FROM employees
   GROUP BY department_id, salary
   ORDER BY department_id, min, max;
SELECT department_id, MIN(salary) min, MAX(salary) max FROM employees
   GROUP BY (department_id, salary)
   ORDER BY department_id, min, max;
```

ROLLUP、CUBE、およびGROUP BY句のGROUPING SETS句では、個々の式と同じリストにある式の集合を組み合せることができます。次に、1つのSQL文に含まれている有効なグルーピング・セットと式のリストを示します。

```
SELECT
prod_category, prod_subcategory, country_id, cust_city, count(*)
   FROM  products, sales, customers
   WHERE sales.prod_id = products.prod_id
   AND sales.cust_id=customers.cust_id
   AND sales.time_id = '01-oct-00'
   AND customers.cust_year_of_birth BETWEEN 1960 and 1970
GROUP BY GROUPING SETS
  (
   (prod_category, prod_subcategory, country_id, cust_city),
   (prod_category, prod_subcategory, country_id),
   (prod_category, prod_subcategory),
    country_id
  )
ORDER BY prod_category, prod_subcategory, country_id, cust_city;
```

関連項目:

SELECT

6 条件

条件は、1つ以上の式と論理(ブール)演算子の組合せで指定し、TRUE、FALSEまたはUNKNOWNのいずれかの値を戻します。

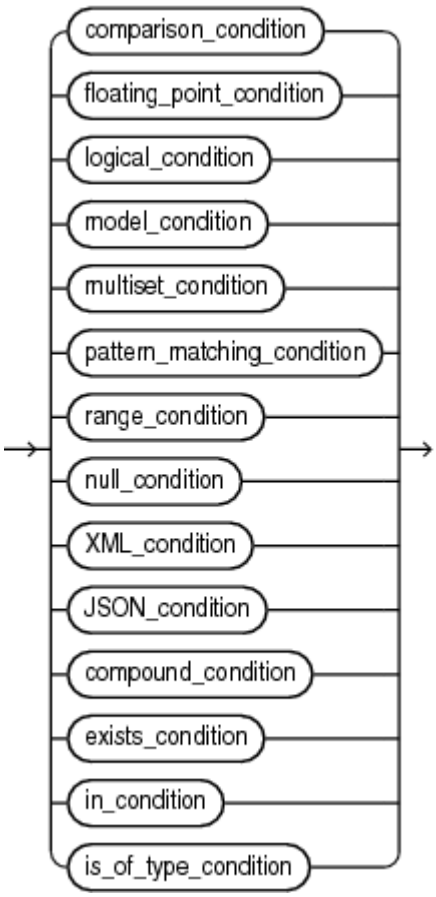
この章の構成は、次のとおりです。

- SQL条件
- 比較条件
- 浮動小数点条件
- 論理条件
- モデル条件
- 多重集合条件
- パターン一致条件
- NULL条件
- XML条件
- SQL For JSON条件
- 複合条件
- BETWEEN条件
- EXISTS条件
- IN条件
- IS OF type条件

SQL条件

条件には、次の構文で示すとおり、複数の書式があります。

condition::=



Oracle Textがインストールされている場合、CONTAINS、CATSEARCH、MATCHESなど、この製品に含まれる組込み演算子を使用して条件を作成できます。Oracle Text要素の詳細は、『Oracle Textリファレンス』を参照してください。

次の項では、様々な書式の条件を説明します。SQL文にconditionが含まれる場合は、適切な条件構文を使用する必要があります。

条件は、次の文のWHERE句で使用できます。

- DELETE
- SELECT
- UPDATE

条件は、SELECT文の次の句で使用できます。

- WHERE
- START WITH
- CONNECT BY
- HAVING

条件は、論理データ型であるともいえます。ただし、Oracle Databaseで、正式にこのようなデータ型をサポートしているわけではありません。

次の単純な条件は常にTRUEに評価されます。

```
1 = 1
```

次のやや複雑な条件は、salaryの値をsalary*commission_pctの値に加算し(NULLは0で置き換える)、その合計が定数25000より大きいかどうかを判断します。

```
NVL(salary, 0) + NVL(salary + (salary*commission_pct, 0) > 25000)
```

論理条件では、複数の条件を単一の条件に結合できます。たとえば、AND条件を使用して2つの条件を結合できます。

```
(1 = 1) AND (5 < 7)
```

次に、有効な条件を示します。

```
name = 'SMITH'
employees.department_id = departments.department_id
hire_date > '01-JAN-08'
job_id IN ('SA_MAN', 'SA_REP')
salary BETWEEN 5000 AND 10000
commission_pct IS NULL AND salary = 2100
```

すべてのSQL文のすべての部分ですべての条件が受け入れられるわけではありません。該当する文における条件の制限事項の詳細は、このマニュアルの特定のSQL文に関する項を参照してください。

条件の優先順位

優先順位とは、同じ式の中の異なる条件をOracle Databaseが評価する順序を意味します。複数の条件を含む式を評価するとき、Oracleは優先順位の高い条件を評価した後で、優先順位の低い条件を評価します。優先順位の等しい条件は、式の中で左から右に評価されます。ただし、次の例外があります。

- ANDを使用して接続された複数の条件については、左から右への評価は保証されません。
- ORを使用して接続された複数の条件については、左から右への評価は保証されません。

表6-1に、SQL条件を優先順位の高い方から順に示します。同じ行にある条件の優先順位は同じです。表に示されているとおり、演算子は条件の前に評価されます。

表6-1 SQL条件の優先順位

条件の種類	目的
SQL 演算子は、SQL 条件の前に評価されます。	「演算子の優先順位」を参照
=, !=, <, >, <=, >=,	比較
IS [NOT] NULL, LIKE, [NOT] BETWEEN, [NOT] IN, EXISTS, IS OF type	比較
NOT	指数、論理否定
AND	論理積
OR	論理和

比較条件

比較条件はある式を別の式と比較します。比較の結果は、TRUE、FALSEまたはUNKNOWNになります。

ラージ・オブジェクト(LOB)は、比較条件ではサポートされていません。ただし、CLOBデータの比較では、PL/SQLプログラムを使用できます。

数式を比較する場合、Oracleは数値の優先順位を使用して、その条件がNUMBER、BINARY_FLOATまたはBINARY_DOUBLEのうちどの値を比較するかを判断します。数値の優先順位の詳細は、数値の優先順位を参照してください。

文字式を比較する場合、Oracleはデータ型の比較規則で説明されている規則を使用します。この規則では、比較の前の式の文字セットの位置、バイナリ比較または言語比較(照合)の使用、空白埋め比較セマンティクスの使用および照合キーに課せられた制限による制約(エラーORA-12742: unable to create the collation keyのレポートなど)が定義されます。

非スカラー型の2つのオブジェクトが比較可能なのは、これらが同じ名前付きの型であり、要素が1対1で対応している場合です。また、ユーザー定義オブジェクト型のネストした表では、要素が比較可能な場合も、等式やIN条件で使用するMAPメソッドが定義されている必要があります。

関連項目:

MAPメソッドを使用したオブジェクトの比較の詳細は、『Oracle Databaseオブジェクト・リレーショナル開発者ガイド』を参照してください。

表6-2に、比較条件を示します。

表6-2 比較条件

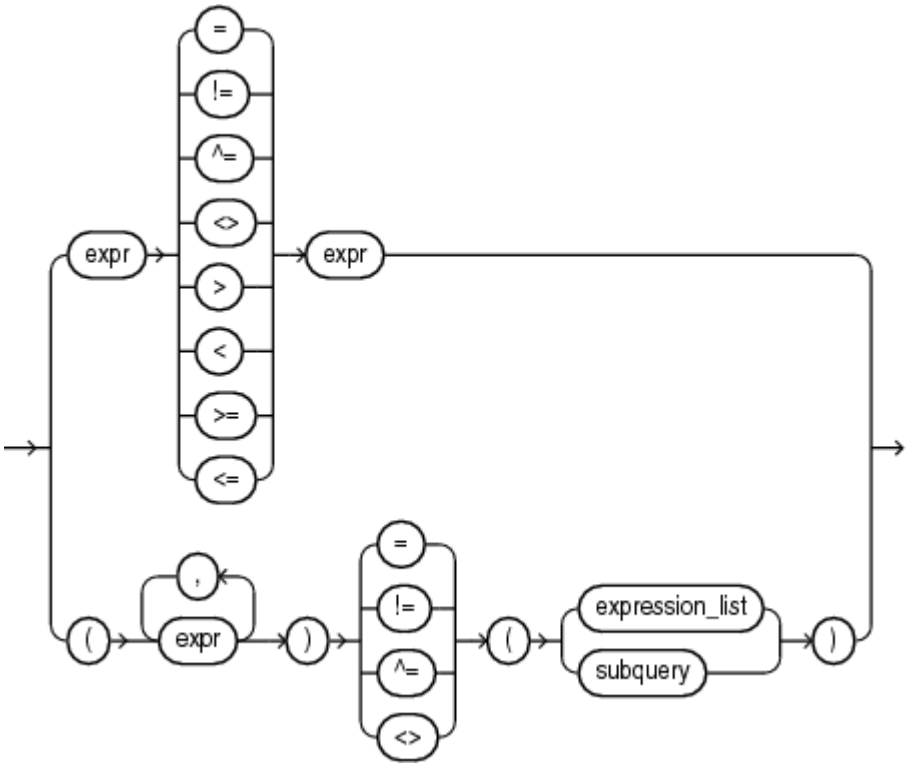
条件の種類	目的	例
=	等価性をテストします。	SELECT * FROM employees WHERE salary = 2500 ORDER BY employee_id;
!= ^= <>	非等価性をテストします。	SELECT * FROM employees WHERE salary != 2500 ORDER BY employee_id;
> <	大小のテスト。	SELECT * FROM employees WHERE salary > 2500 ORDER BY employee_id; SELECT * FROM employees WHERE salary < 2500 ORDER BY employee_id;
>=	以上または以下のテスト。	SELECT * FROM employees WHERE salary >= 2500 ORDER BY employee_id; SELECT *

条件の種類	目的	例
<=		FROM employees WHERE salary <= 2500 ORDER BY employee_id;
op ANY OP SOME	"op"は=、!=、>、<、<=、または>=のいずれかである必要があります。 op ANY は、左側の値をリスト内の各値、または問合せによって返される各値と比較します。どちらと比較するかは、条件 op を使用して、右側で指定されます。 これらの比較のいずれかが TRUE を返す場合、op ANY は TRUE を戻します。 これらの比較がすべて FALSE を返すか、右側の副問合せが行を返さない場合、op ANY は FALSE を戻します。それ以外の場合、戻り値は UNKNOWN です。 op ANY と op SOME は同義です。	SELECT * FROM employees WHERE salary = ANY (SELECT salary FROM employees WHERE department_id = 30) ORDER BY employee_id;
op ALL	"op"は=、!=、>、<、<=、または>=のいずれかである必要があります。 op ALL は、左側の値をリスト内の各値、または副問合せによって返される各値と比較します。どちらと比較するかは、条件 op を使用して、右側で指定されます。 これらの比較のいずれかが FALSE を戻す場合、op ALL は FALSE を戻します。 これらの比較がすべて TRUE を返すか、右側の副問合せが行を返さない場合、op ALL は TRUE を戻します。それ以外の場合、戻り値は UNKNOWN です。	SELECT * FROM employees WHERE salary >= ALL (1400, 3000) ORDER BY employee_id;

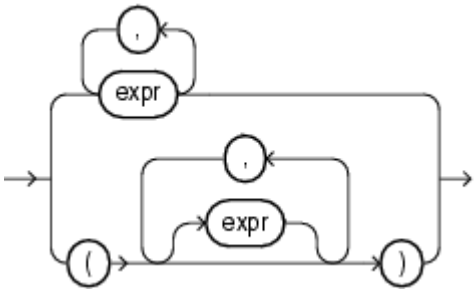
単純比較条件

単純比較条件は、式または副問合せの結果の比較方法を指定します。

simple_comparison_condition::=



expression_list::=



この条件の下の方の書式(演算子の左辺に1つの式を指定する書式)を使用する場合は、expression_listの上または下の方の書式を使用できます。この条件の下の方の書式(演算子の左辺に複数の式を指定する書式)を使用する場合は、expression_listの下の方の書式を使用する必要があります。どちらの場合も、expression_listの式は、演算子の左辺にある式と同じ数とデータ型で構成されている必要があります。副問合せを指定する場合、副問合せから戻される値は、演算子の左辺にある式と同じ数とデータで構成されている必要があります。

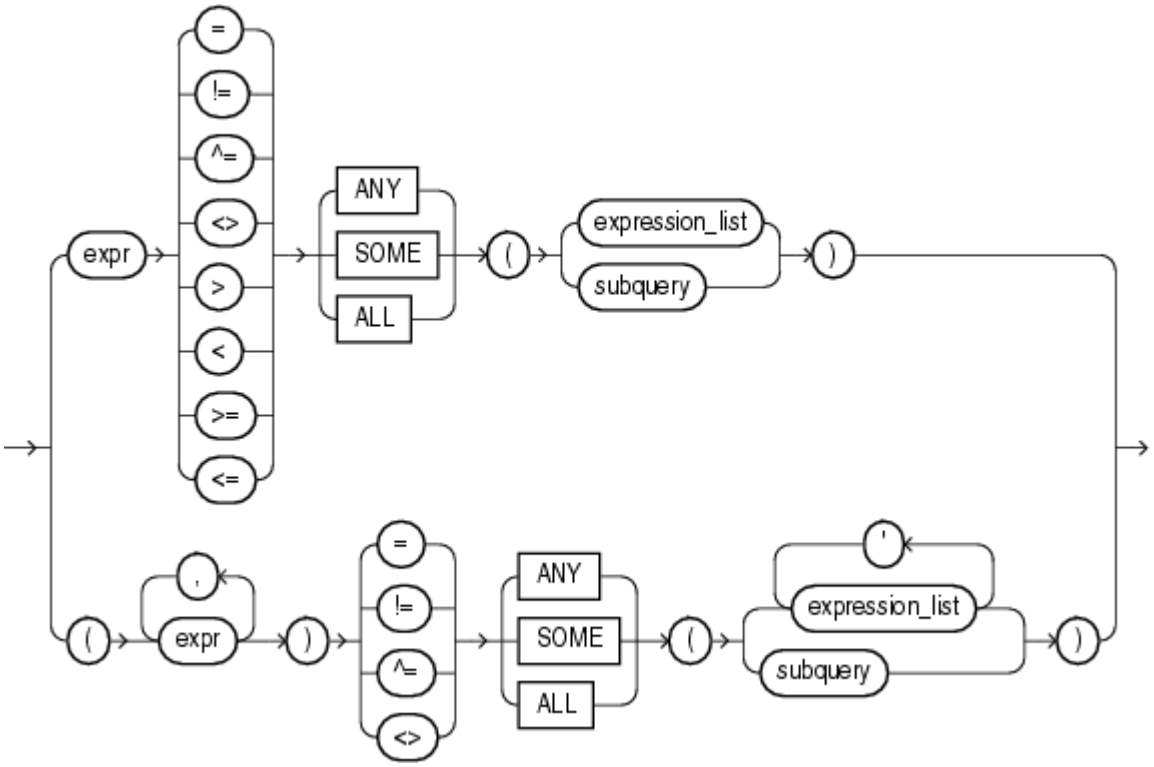
関連項目:

式の組合せの詳細は、式のリストを参照してください。副問合せの詳細は、SELECTを参照してください。

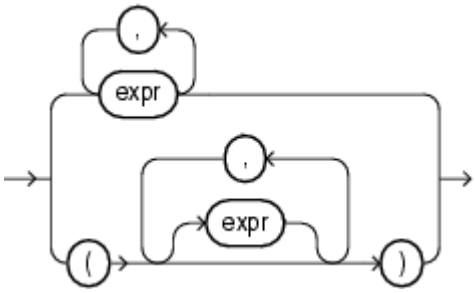
グループ比較条件

グループ比較条件は、リストまたは副問合せ内の任意またはすべてのメンバーの比較方法を指定します。

group_comparison_condition::=



expression_list::=



この条件の上の方の書式(演算子の左辺に1つの式を指定する書式)を使用する場合は、expression_listの上の方の書式を使用する必要があります。この条件の下の方の書式(演算子の左辺に複数の式を指定する書式)を使用する場合は、expression_listの下の方の書式を使用し、各expression_list内の式は、演算子の左辺にある式と同じ数とデータ型で構成されている必要があります。副問合せを指定する場合、副問合せから戻される値は、演算子の左辺にある式と同じ数とデータで構成されている必要があります。

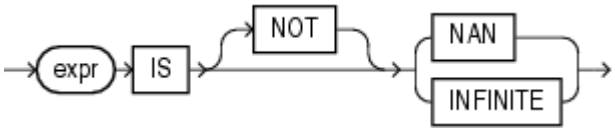
関連項目:

- 式のリスト
- SELECT
- 比較条件の照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

浮動小数点条件

浮動小数点条件では、式が無限か、または演算の未定義の結果(非数値(NaN))かを判断します。

floating_point_condition::=



どちらの浮動小数点条件でも、exprは、数値データ型、または数値データ型に暗黙的に変換可能な任意のデータ型に解決される必要があります。表6-3に、浮動小数点条件を示します。

表6-3 浮動小数点条件

条件の種類	操作	例
IS [NOT] NAN	NOT が指定されているときは、expr が特殊な値である NaN の場合に TRUE を戻します。NOT が指定されているときは、expr が特殊な値である NaN ではない場合に TRUE を戻します。	SELECT COUNT(*) FROM employees WHERE commission_pct IS NOT NAN;
IS [NOT] INFINITE	NOT が指定されていないときは、expr が特殊な値である+INF または-INF の場合に TRUE を戻します。NOT が指定されているときは、expr が特殊な値である+INF でも-INF でもない場合に TRUE を戻します。	SELECT last_name FROM employees WHERE salary IS NOT INFINITE;

関連項目:

- Oracleでの浮動小数点数の実装の詳細は、浮動小数点数を参照してください。
- Oracleでの浮動小数点データ型の変換方法の詳細は、暗黙的なデータ変換を参照してください。

論理条件

論理条件は、構成要素である2つの条件の結果を結合して、それらに基づく単一の結果を生成するか、または単一条件の結果を逆転します。表6-4に、論理条件を示します。

表6-4 論理条件

条件の種類	操作	例
NOT	以降の条件が FALSE の場合は TRUE を返します。条件が TRUE の場合は FALSE を返します。条件が UNKNOWN の場合は UNKNOWN のままです。	`SELECT * FROM employees WHERE NOT (job_id IS NULL) ORDER BY employee_id; SELECT * FROM employees WHERE NOT (salary BETWEEN 1000 AND 2000) ORDER BY employee_id;`
AND	両方のコンポーネント条件が TRUE の場合は TRUE を返します。一方が FALSE の場合は FALSE を返します。それ以外の場合は UNKNOWN を戻します。	`SELECT * FROM employees WHERE job_id = 'PU_CLERK' AND department_id = 30 ORDER BY employee_id;`
OR	一方のコンポーネント条件が TRUE の場合は TRUE を返します。両方が FALSE の場合は FALSE を返します。それ以外の場合は UNKNOWN を戻します。	`SELECT * FROM employees WHERE job_id = 'PU_CLERK' OR department_id = 10 ORDER BY employee_id;`

表6-5に、式にNOT条件を適用した結果を示します。

表6-5 NOT真理値表

--	TRUE	FALSE	UNKNOWN
NOT	FALSE	TRUE	UNKNOWN

表6-6に、2つの式にAND条件を組み合せた結果を示します。

表6-6 AND真理値表

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

たとえば、次のSELECT文のWHERE句は、AND論理条件を使用して、2004年より前に入社し、給与が$2500を超える従業員のみを戻します。

```
SELECT * FROM employees
WHERE hire_date < TO_DATE('01-JAN-2004', 'DD-MON-YYYY')
  AND salary > 2500
  ORDER BY employee_id;
```

表6-7に、2つの式にORを適用した結果を示します。

表6-7 OR真理値表

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

たとえば、次の問合せは、歩合率が40%または給与が$20,000を超える従業員を戻します。

```
SELECT employee_id FROM employees
   WHERE commission_pct = .4 OR salary > 20000
   ORDER BY employee_id;
```

モデル条件

モデル条件は、SELECT文のMODEL句でのみ使用できます。

IS ANY条件

IS ANY条件は、SELECT文のmodel_clauseでのみ使用できます。この条件を使用して、ディメンション列のすべての値(NULLを含む)を修飾します。

is_any_condition::=

この条件は、常にブール値のTRUEを戻し、列内のすべての値を修飾します。

関連項目:

詳細は、model_clauseおよびモデル式を参照してください。

例

次の例は、2000年の各製品の売上を0(ゼロ)に設定します。

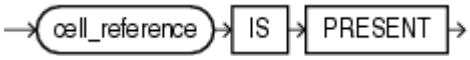
```
SELECT country, prod, year, s
  FROM sales_view_ref
  MODEL
    PARTITION BY (country)
    DIMENSION BY (prod, year)
    MEASURES (sale s)
    IGNORE NAV
    UNIQUE DIMENSION
    RULES UPSERT SEQUENTIAL ORDER
    (
      s[ANY, 2000] = 0
    )
  ORDER BY country, prod, year;
COUNTRY       PROD                                      YEAR          S
----------    ----------------------------------      --------   ---------
France        Mouse Pad                                   1998     2509.42
France        Mouse Pad                                   1999     3678.69
France        Mouse Pad                                   2000           0
France        Mouse Pad                                   2001     3269.09
France        Standard Mouse                              1998     2390.83
France        Standard Mouse                              1999     2280.45
France        Standard Mouse                              2000           0
France        Standard Mouse                              2001     2164.54
Germany       Mouse Pad                                   1998     5827.87
Germany       Mouse Pad                                   1999     8346.44
Germany       Mouse Pad                                   2000           0
Germany       Mouse Pad                                   2001     9535.08
Germany       Standard Mouse                              1998     7116.11
Germany       Standard Mouse                              1999     6263.14
Germany       Standard Mouse                              2000           0
Germany       Standard Mouse                              2001     6456.13

16 rows selected.
```

この例では、ビューsales_view_refが必要です。このビューを作成する方法は、MODEL句: 例を参照してください。

IS PRESENT条件

is_present_condition::=

IS PRESENT条件は、SELECT文のmodel_clauseでのみ使用できます。この条件を使用すると、model_clauseの実行前に、参照されるセルが存在するかどうかをテストできます。



この条件は、model_clauseの実行前にセルが存在する場合にTRUE、存在しない場合にFALSEを戻します。

関連項目:

詳細は、model_clauseおよびモデル式を参照してください。

例

次の例では、1999年のマウス・パッドの売上が存在する場合、2000年のマウス・パッドの売上が1999年のマウス・パッドの売上に設定されます。1999年のマウス・パッドの売上が存在しない場合は、2000年のマウス・パッドの売上は0(ゼロ)に設定されます。

```
SELECT country, prod, year, s
  FROM sales_view_ref
  MODEL
    PARTITION BY (country)
    DIMENSION BY (prod, year)
    MEASURES (sale s)
    IGNORE NAV
    UNIQUE DIMENSION
    RULES UPSERT SEQUENTIAL ORDER
    (
      s['Mouse Pad', 2000] =
        CASE WHEN s['Mouse Pad', 1999] IS PRESENT
             THEN s['Mouse Pad', 1999]
             ELSE 0
        END
    )
  ORDER BY country, prod, year;
COUNTRY       PROD                                    YEAR          S
----------    ----------------------------------    --------   ---------
France        Mouse Pad                                 1998     2509.42
France        Mouse Pad                                 1999     3678.69
France        Mouse Pad                                 2000     3678.69
France        Mouse Pad                                 2001     3269.09
France        Standard Mouse                            1998     2390.83
France        Standard Mouse                            1999     2280.45
France        Standard Mouse                            2000     1274.31
France        Standard Mouse                            2001     2164.54
Germany       Mouse Pad                                 1998     5827.87
Germany       Mouse Pad                                 1999     8346.44
Germany       Mouse Pad                                 2000     8346.44
Germany       Mouse Pad                                 2001     9535.08
Germany       Standard Mouse                            1998     7116.11
Germany       Standard Mouse                            1999     6263.14
Germany       Standard Mouse                            2000     2637.31
Germany       Standard Mouse                            2001     6456.13
16 rows selected.
```

この例では、ビュー`sales_view_ref`が必要です。このビューを作成する方法は、MODEL句：例を参照してください。

多重集合条件

多重集合条件は、ネストした表を様々な側面からテストします。

IS A SET条件

IS A SET条件を使用すると、指定されたネストした表が一意の要素で構成されているかどうかをテストできます。この条件は、ネストした表がNULLの場合に、UNKNOWNを戻します。それ以外では、ネストした表がセットである場合(ネストした表の長さが0(ゼロ)の場合も含む)にTRUEを戻し、その他の場合はFALSEを戻します。

is_a_set_condition::=

例

次の例は、表customers_demoから、cust_address_ntabというネストした表の列に一意の要素が含まれる行を選択します。

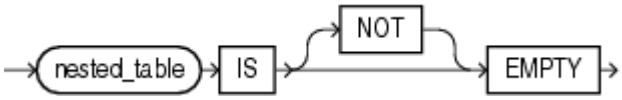
```
SELECT customer_id, cust_address_ntab
  FROM customers_demo
  WHERE cust_address_ntab IS A SET
  ORDER BY customer_id;
CUSTOMER_ID CUST_ADDRESS_NTAB(STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE,
COUNTRY_ID)
--------------------------------------------------------------------------------
---------
        101 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('514 W Superior St', '46901',
'Kokomo', 'IN', 'US'))
        102 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('2515 Bloyd Ave', '46218',
'Indianapolis', 'IN', 'US'))
        103 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('8768 N State Rd 37', '47404',
'Bloomington', 'IN', 'US'))
        104 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('6445 Bay Harbor Ln', '46254',
'Indianapolis', 'IN', 'US'))
        105 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('4019 W 3Rd St', '47404',
'Bloomington', 'IN', 'US'))
```

この例では、表customers_demoと、データを含むネストした表の列が1つ必要です。この表およびネストした表の列を作成する方法は、「MULTISET演算子」を参照してください。

IS EMPTY条件

IS [NOT] EMPTY条件を使用すると、指定されたネストした表が空かどうかをテストできます。単一の値(NULL)で構成されたネストした表は、空のネストした表とはみなされません。

is_empty_condition::=



この条件は、コレクションが空の場合にIS EMPTY条件にブール値TRUEを戻し、コレクションが空でない場合にIS NOT EMPTY条件にブール値TRUEを戻します。ネストした表またはVARRAYにNULLを指定すると、結果はNULLになります。

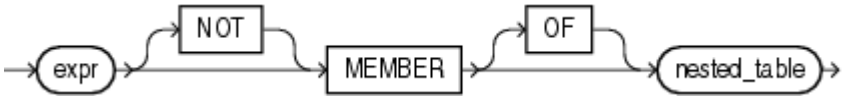
例

次の例は、サンプル表pm.print_mediaから、ad_textdocs_ntabというネストした表の列が空ではない行を選択します。

```
SELECT product_id, TO_CHAR(ad_finaltext) AS text
   FROM print_media
   WHERE ad_textdocs_ntab IS NOT EMPTY
   ORDER BY product_id, text;
```

MEMBER条件

member_condition::=



member_conditionは、要素がネストした表のメンバーかどうかをテストするメンバーシップ条件です。exprが指定されたネストした表またはVARRAYのメンバーと等しい場合、TRUEが戻されます。exprがNULLまたはネストした表が空の場合、NULLが戻されます。

- exprの型は、ネストした表の要素型と同じである必要があります。
- OFキーワードはオプションであり、条件の動作に影響しません。
- NOTキーワードはブール出力を逆にします。exprが指定されたネストした表のメンバーである場合、FALSEが戻されます。
- ネストした表の要素の型は、比較可能な型である必要があります。スカラー型以外の型の比較の可能性は、比較条件を参照してください。

例

次の例は、表customers_demoから、cust_address_ntabというネストした表の列に、WHERE句で指定された値が含まれる行を選択します。

```
SELECT customer_id, cust_address_ntab
  FROM customers_demo
  WHERE cust_address_typ('8768 N State Rd 37', 47404,
                         'Bloomington', 'IN', 'US')
  MEMBER OF cust_address_ntab
  ORDER BY customer_id;
CUSTOMER_ID CUST_ADDRESS_NTAB(STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE,
COUNTRY_ID)
------------ ------------------------------------------------------------------------
---------
        103 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('8768 N State Rd 37', '47404',
'Bloomington', 'IN', 'US'))
```

この例では、表customers_demoと、データを含むネストした表の列が1つ必要です。この表およびネストした表の列を作成する方法は、MULTISET演算子を参照してください。

SUBMULTISET条件

SUBMULTISET条件は、指定されたネストした表が他の指定されたネストした表のサブ多重集合かどうかをテストします。

この演算子はブール値を戻します。nested_table1がnested_table2のサブ多重集合である場合、TRUEが戻されま

す。次の条件のいずれかが発生する場合、nested_table1はnested_table2のサブ多重集合となります。

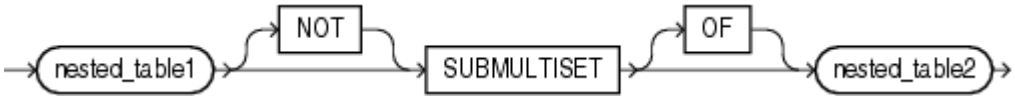
- nested_table1がNULLでなく、また行を含まない場合。空の多重集合はnested_table2に対してNULL以外の値を示すサブ多重集合であるため、nested_table2がNULLの場合でもTRUEが戻されます。
- nested_table1とnested_table2がともにNULLでなく、nested_table1にNULL要素が含まれず、またnested_table1内の各要素がnested_table2内の等しい要素と1対1でマップされている場合。

次の条件のいずれかが発生する場合、NULLが戻されます。

- nested_table1がNULLの場合。
- nested_table2がNULLで、またnested_table1がNULLでもなく空でもない場合。
- nested_table1およびnested_table2の各NULL要素をNULL以外の値に変更し、nested_table1内の各要素と等しいnested_table2内の要素の1対1マッピングを有効にすることで、nested_table1がnested_table2のサブ多重集合となっている場合。

前述の条件のいずれも発生していない場合は、FALSEが戻されます。

submultiset_condition::=



- OFキーワードはオプションであり、演算子の動作に影響しません。
- NOTキーワードはブール出力を逆にします。nested_table1がnested_table2のサブセットである場合、FALSEが戻されます。
- ネストした表の要素の型は、比較可能な型である必要があります。スカラー型以外の型の比較の可能性は、比較条件を参照してください。

例

次の例は、customers_demo表から、cust_address_ntabというネストした表がcust_address2_ntabというネストした表のサブ多重集合である行を選択します。

```
SELECT customer_id, cust_address_ntab
  FROM customers_demo
  WHERE cust_address_ntab SUBMULTISET OF cust_address2_ntab
  ORDER BY customer_id;
```

この例では、表customers_demoと、データを含むネストした表の列が2つ必要です。この表およびネストした表の列を作成する方法は、MULTISET演算子を参照してください。

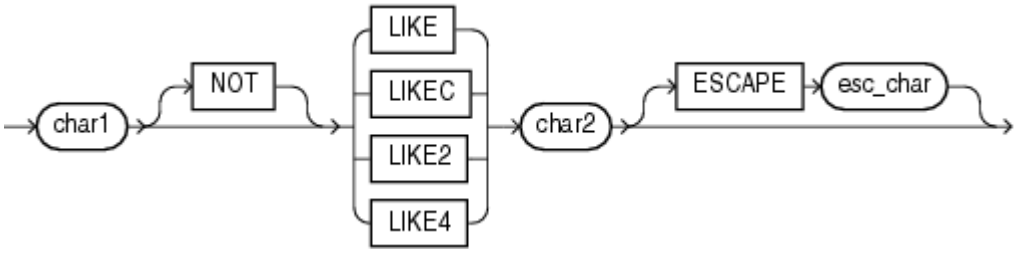
パターン一致条件

パターン一致条件は文字データを比較します。

LIKE条件

LIKE条件は、パターン一致を含むかどうかをテストします。等号演算子(=)は、ある文字値を別の文字値と一致させますが、LIKE条件は、ある文字値の一部を別の文字値と一致させます(ある値が指定したパターンの検索を、もう一方の値に対して行います)。LIKEは入力文字セットで定義された文字を使用して、文字列を計算します。LIKECは、完全なUnicodeキャラクタを使用します。LIKE2は、UCS2コードポイントを使用します。LIKE4は、UCS4コードポイントを使用します。

like_condition::=



この構文は、次の特長があります。

- char1は、キャラクタ列などの文字式で、検索値と呼ばれます。
- char2は、通常はリテラルの文字式で、パターンと呼ばれます。
- esc_charは、通常はリテラルの文字式で、エスケープ文字と呼ばれます。

LIKE条件は、ほぼすべての場合において最適な選択です。次のガイドラインを使用して、ご使用の環境に有効な例を判断してください。

- LIKE2を使用して、UCS-2セマンティクスで文字列を処理します。LIKE2は、Unicode補助文字を2文字として扱います。
- LIKE4を使用して、UCS-4セマンティクスで文字列を処理します。LIKE4は、Unicode補助文字を1文字として扱います。
- LIKECを使用して、完全なUnicodeキャラクタ・セマンティクスで文字列を処理します。LIKECは、複合文字を1文字として扱います。

文字長の詳細は、次を参照してください。

- Oracle Databaseグローバリゼーション・サポート・ガイド
- Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド

esc_charが指定されていない場合、デフォルトのエスケープ文字はありません。char1、char2またはesc_charのいずれかがNULLである場合、結果は不明になります。それ以外の場合は、エスケープ文字(指定されている場合)は、長さが1の文字列です。

すべての文字式(char1、char2およびesc_char)は、CHAR、VARCHAR2、NCHARまたはNVARCHAR2データ型です。文字式のデータ型が異なる場合、Oracleはすべての文字式をchar1のデータ型に変換します。

パターンは、特殊パターン一致文字を含むことができます。

- パターン中のアンダースコア(_)は、値の中の1文字(マルチバイトの文字セットでの1バイトとは異なる)に置き換えることができます。
- パターン中のパーセント記号(%)は、値の中の0(ゼロ)を含む任意の数の文字(マルチバイトの文字セットでの1バイトとは異なる)に置き換えることができます。ただし、パターン「%」は、NULLに置き換えることはできません。

エスケープ文字を識別するESCAPE句を使用すると、パターン中に%または_を実際の文字として含めることができます。エスケープ文字がパターンの中で文字%または_の前に指定されている場合、Oracleは、この文字を特殊パターン一致文字としてではなく、リテラル文字として解析します。また、エスケープ文字自体を検索する場合は、その文字を続けて入力してください。たとえば、アットマーク(@)がエスケープ文字である場合、@@を使用してアットマーク(@)を検索できます。

ノート:

ASCII 表記のアンダースコア(_)およびパーセント(%)文字のみが、パターン一致文字として認識されます。東アジア文字セットおよび Unicode で表示される全角文字は、通常の文字として扱われます。

表6-8に、LIKE条件を示します。

表6-8 LIKE条件

条件の種類	操作	例
x [NOT] LIKE y [ESCAPE 'z']	x がパターン y に一致する場合(NOT を指定すると一致しない場合)は TRUE と評価されます。y 内で、文字%は 0 以上の NULL 以外の文字を含む文字列と一致します。文字_は、任意の 1 文字に一致します。パーセント(%)およびアンダースコア(_)を除く任意の文字を ESCAPE の後に指定できます。ワイルド・カード文字は、エスケープ文字が前に付いている場合はリテラルとして扱われます。	SELECT last_name FROM employees WHERE last_name LIKE '%A¥_B%' ESCAPE '¥' ORDER BY last_name;

LIKE条件を処理するために、Oracleは、パターンを1つまたは2つの文字で構成されるサブパターンに分割します。2文字のサブパターンは、エスケープ文字で始まり、もう1つの文字はパーセント(%)、アンダースコア(_)またはエスケープ文字です。

P_1、P_2、...、P_nを、このようなサブパターンと想定します。1からnのすべてのiにおいて、次の条件を満たすように検索値を部分文字列S_1、S_2、...、S_nに分割できる場合、LIKE条件はTRUEです。

- P_iがアンダースコア(_)の場合、S_iは単一文字である。
- P_iがパーセント(%)の場合、S_iは任意の文字列である。
- P_iがエスケープ文字で始まる2文字の場合、S_iはP_iの2番目の文字である。
- それ以外の場合、P_iとS_iは一致する。

LIKE条件では、値を定数ではなくパターンと比較できます。必ずLIKEキーワードの直後に、パターンを指定してください。たとえば、次の問合せを発行することによって、名前がRで始まるすべての従業員の給与を検索できます。

```
SELECT salary
    FROM employees
```

```
    WHERE last_name LIKE 'R%'
    ORDER BY salary;
```

次の問合せは、LIKE条件ではなく=演算子を使用しているため、名前が「R%」のすべての従業員の給与が検索されます。

```
SELECT salary
    FROM employees
    WHERE last_name = 'R%'
    ORDER BY salary;
```

次の問合せでは、名前が「SM%」のすべての従業員の給与が検索されます。この場合、'SM%'がLIKEキーワードの前にあるため、Oracleは、'SM%'をパターンとしてではなく、テキスト・リテラルとして解析します。

```
SELECT salary
    FROM employees
    WHERE 'SM%' LIKE last_name
    ORDER BY salary;
```

照合および大/小文字の区別

LIKE条件は照合依存です。Oracle Databaseにおいて、前述の処理アルゴリズムでは、サブパターンPiがサブストリングSiと比較され、そのとき、char1とchar2の導出照合から決定された照合が使用されます。この照合で大/小文字が区別されない場合は、パターン一致でも大/小文字が区別されません。

関連項目:

大/小文字またはアクセントを区別しない照合およびLIKE条件の照合決定ルールの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

索引列でのパターン一致

LIKEを使用して索引列をパターン検索する場合、パターンの先頭文字が%または_でなければ、Oracleは索引を利用して問合せのパフォーマンスを向上させることができます。この場合、Oracleはこの先頭文字によって索引をスキャンできます。パターンの先頭文字が%または_の場合、Oracleは索引をスキャンできないため、パフォーマンスは向上しません。

LIKE条件: 一般的な例

次の条件は、Maで始まるすべてのlast_name値でtrueです。

```
last_name LIKE 'Ma%'
```

次のすべてのlast_name値では、条件がtrueになります。

```
Mallin, Markle, Marlow, Marvins, Mavris, Matos
```

大文字と小文字が区別されるため、MA、ma、mAで始まるlast_name値では条件がfalseになります。

次のような条件があるとします。

```
last_name LIKE 'SMITH_'
```

この条件は、次のlast_name値でtrueです。

```
SMITHE, SMITHY, SMITHS
```

特殊文字であるアンダースコア(_)は、last_name値の1つの文字に置き換えることができるため、この条件は「SMITH」についてFALSE(偽)となります。

ESCAPE句の例

次の例では、名前のパターンがA_Bである従業員を検索します。

```
SELECT last_name
    FROM employees
    WHERE last_name LIKE '%A¥_B%' ESCAPE '¥'
    ORDER BY last_name;
```

ESCAPE句は、エスケープ文字としてバックスラッシュ(¥)を識別します。パターンの中でエスケープ文字はアンダースコア(_)に先行します。これによって、Oracleは、アンダースコアを特殊なパターン一致文字としてではなく、リテラルとして解析します。

%なしのパターンの例

パターンに%文字が含まれていない場合、条件がtrueになるのは、両方のオペランドが同じ長さである場合のみです。次の表の定義および挿入される値について考えます。

```
CREATE TABLE ducks (f CHAR(6), v VARCHAR2(6));
INSERT INTO ducks VALUES ('DUCK', 'DUCK');
SELECT '*'||f||'*' "char",
   '*'||v||'*' "varchar"
   FROM ducks;
char     varchar
-------- --------
*DUCK  * *DUCK*
```

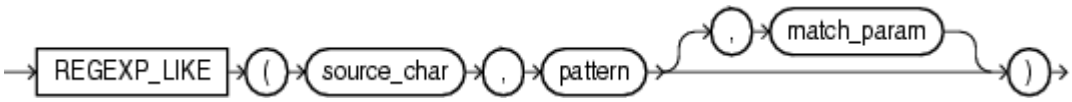
Oracleは、CHAR値に空白埋めを行うため、fの値は空白埋めによって6バイトになります。vの値は空白埋めされず、4文字長のままです。

REGEXP_LIKE条件

REGEXP_LIKEはLIKE条件と似ています。ただし、REGEXP_LIKEは、単純なパターン一致を実行するLIKEとは異なり、正規表現一致を実行します。この条件は、入力文字セットによって定義された文字を使用して、文字列を評価します。

この条件は、POSIX正規表現規格およびUnicode Regular Expression Guidelinesに準拠します。詳細は、「Oracleの正規表現のサポート」を参照してください。

regexp_like_condition::=



- source_charは、検索値として使用される文字式です。通常は文字列であり、データ型はCHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBのいずれかです。
- patternは正規表現です。通常はテキスト・リテラルであり、データ型はCHAR、VARCHAR2、NCHARまたはNVARCHAR2のいずれかになります。ここには、 512バイトまで入力できます。patternのデータ型がsource_charのデータ型と異なる場合、Oraclepatternをsource_charのデータ型に変換します。patternで指定できる演算子のリストは、「Oracleの正規表現のサポート」を参照してください。
- match_paramは、データ型がVARCHAR2またはCHARの文字式であり、条件のデフォルトの照合動作を変更できます。

  match_paramの値には、次の文字を1つ以上含めることができます。

  - 'i'を指定すると、条件に対して決定されている照合が大/小文字の区別ありであっても、大/小文字の区

別なしでの照合となります。

- 'c'を指定すると、条件に対して決定されている照合が大/小文字の区別なしまたはアクセントの区別なしの場合でも、大/小文字の区別あり、アクセントの区別ありでの照合となります。
- 'n': ピリオド(.)の使用を許可します。ピリオドは、改行文字を含む任意の文字と一致するワイルド・カード文字です。このパラメータを指定しない場合、ピリオドは改行文字には一致しません。
- 'm': ソース文字列を複数行として処理します。Oracleは、 ^および$を、それぞれ、ソース文字列全体の開始または終了としてのみではなく、ソース文字列内の任意の場所にある任意の行の開始または終了としてとして解釈します。このパラメータを指定しない場合、Oracleはソース文字列を単一行として処理します。
- 'x'は空白文字を無視します。デフォルトでは、空白文字は空白文字として一致します。

相反する文字がmatch_paramの値に複数含まれている場合は、最後の文字が使用されます。たとえば、'ic'を指定した場合は、大/小文字の区別あり、アクセントの区別ありでの照合が使用されます。値に前述以外の文字が含まれている場合は、エラーが戻されます。

match_paramを指定しない場合、次のようになります。

- 大/小文字およびアクセントを区別するかどうかのデフォルトは、REGEXP_LIKE条件に対して決定されている照合によって決まります。
- ピリオド(.)は改行文字に一致しません。
- ソース文字列は単一行として処理されます。

LIKE条件と同様、REGEXP_LIKE条件は照合依存です。

関連項目:

- LIKE条件
- 正規表現をサポートするファンクションについては、「REGEXP_INSTR」、「REGEXP_REPLACE」および「REGEXP_SUBSTR」を参照してください。
- REGEXP_LIKE条件の照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の問合せは、名前がStevenまたはStephenである(first_nameがSteで始まってenで終わり、間がvまたはphがある)従業員の姓と名を戻します。

```
SELECT first_name, last_name
FROM employees
WHERE REGEXP_LIKE (first_name, '^Ste(v|ph)en$')
ORDER BY first_name, last_name;
FIRST_NAME           LAST_NAME
-------------------- -------------------------
Steven               King
Steven               Markle
Stephen              Stiles
```

次の問合せは、姓に母音を2つ含む(大文字か小文字にかかわらず、last_nameでa、e、i、oまたはuのいずれかが2つ連続している)従業員の姓を戻します。

```
SELECT last_name
FROM employees
WHERE REGEXP_LIKE (last_name, '([aeiou])¥1', 'i')
ORDER BY last_name;
LAST_NAME
-------------------------
De Haan
Greenberg
Khoo
Gee
Greene
Lee
Bloom
Feeney
```

NULL条件

NULL条件はnullをテストします。これはnullのテストで使用される唯一の条件です。

null_condition::=

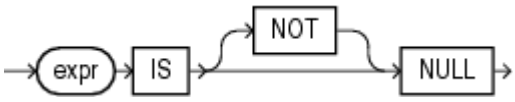


表6-9に、NULL条件を示します。

表6-9 NULL条件

条件の種類	操作	例
IS [NOT] NULL	NULL をテストします。 関連項目: NULL	SELECT last_name FROM employees WHERE commission_pct IS NULL ORDER BY last_name;

XML条件

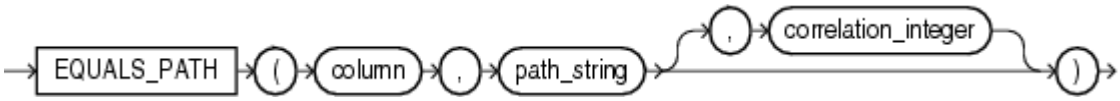
XML条件は、指定したXMLリソースが、指定されたパスにあるかどうかを判断します。

EQUALS_PATH条件

EQUALS_PATH条件は、Oracle XMLデータベース内のリソースが、指定されたパスのデータベースにあるかどうかを判断します。

この条件は、RESOURCE_VIEWおよびPATH_VIEWの問合せで使用します。これらのパブリック・ビューは、XMLデータベース・リポジトリ内に格納されているデータに対してSQLでアクセスするためのメカニズムを提供します。RESOURCE_VIEWには、リポジトリ内のリソースごとに行が1行あり、PATH_VIEWには、リポジトリ内の一意パスごとに行が1行あります。

equals_path_condition::=



この条件は、指定どおりパスにのみ適用されます。この条件は、UNDER_PATHと似ていますが、制限は多くなります。

path_stringには、変換する(絶対)パス名を指定します。このパス名には、ハード・リソース・リンクまたは弱いリソース・リンクが構成要素として含まれます。

オプションのcorrelation_integer引数は、EQUALS_PATH条件を補助ファンクションDEPTHおよびPATHと関連付けます。

関連項目:

「UNDER_PATH条件」、「DEPTH」および「PATH」を参照してください。

例

ビューRESOURCE_VIEWは、データベース・リポジトリ内の(res列にある)すべてのXMLリソースへの(any_path列にある)パスを計算します。次の例は、RESOURCE_VIEWビューに問い合せて、サンプル・スキーマoeにあるリソースへのパスを検索します。EQUALS_PATH条件によって、問合せは指定されたパスのみを戻します。

```
SELECT ANY_PATH FROM RESOURCE_VIEW
   WHERE EQUALS_PATH(res, '/sys/schemas/OE/www.example.com')=1;
ANY_PATH
--------------------------------------------
/sys/schemas/OE/www.example.com
```

この例と、「UNDER_PATH条件」の例を比較してください。

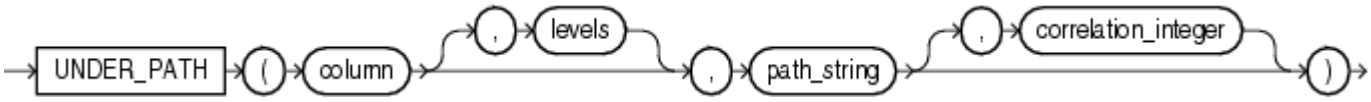
UNDER_PATH条件

UNDER_PATH条件は、列で指定されたリソースが、Oracle XMLデータベース・リポジトリのpath_stringで指定された特定のパスにあるかどうかを判断します。パス情報は、この条件を使用する場合に問い合せるRESOURCE_VIEWビューによって計算されます。

この条件は、RESOURCE_VIEWおよびPATH_VIEWの問合せで使用します。これらのパブリック・ビューは、XMLデータベース・

リポジトリ内に格納されているデータに対してSQLでアクセスするためのメカニズムを提供します。RESOURCE_VIEWには、リポジトリ内のリソースごとに行が1行あり、PATH_VIEWには、リポジトリ内の一意パスごとに行が1行あります。

under_path_condition::=



オプションのlevels引数は、検索対象となるpath_string以下のレベル数を示します。levelsには負ではない整数を指定します。

オプションのcorrelation_integer引数は、UNDER_PATH条件を補助ファンクションPATHおよびDEPTHと関連付けます。

関連項目:

関連する条件については、「EQUALS_PATH条件」を参照してください。補助ファンクションについては、「DEPTH」および「PATH」を参照してください。

例

ビューRESOURCE_VIEWは、データベース・リポジトリ内の(res列にある)すべてのXMLリソースへの(any_path列にある)パスを計算します。次の例は、RESOURCE_VIEWビューに問い合せて、サンプル・スキーマoeにあるリソースへのパスを検索します。この問合せは、XMLType表の例で作成されたXMLスキーマのパスを戻します。

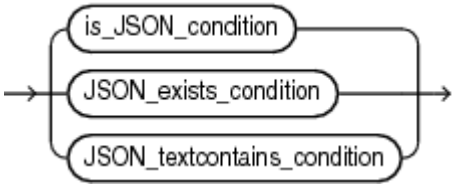
```
SELECT ANY_PATH FROM RESOURCE_VIEW
   WHERE UNDER_PATH(res, '/sys/schemas/OE/www.example.com')=1;
ANY_PATH
--------------------------------------------
/sys/schemas/OE/www.example.com/xwarehouses.xsd
```

SQL For JSON条件

SQL for JSON条件によって、次のようにJavaScript Object Notation (JSON)データをテストできます。

- IS JSON条件によって、式が構文的に正しいJSONデータかどうかをテストできます。
- JSON_EXISTS条件によって、指定されたJSON値がJSONデータにあるかどうかをテストできます。
- JSON_TEXTCONTAINS条件によって、指定された文字列がJSONプロパティ値にあるかどうかをテストできます。
- JSON_EQUAL条件は、2つのJSON値が同じかどうかをテストします。

JSON_condition::=

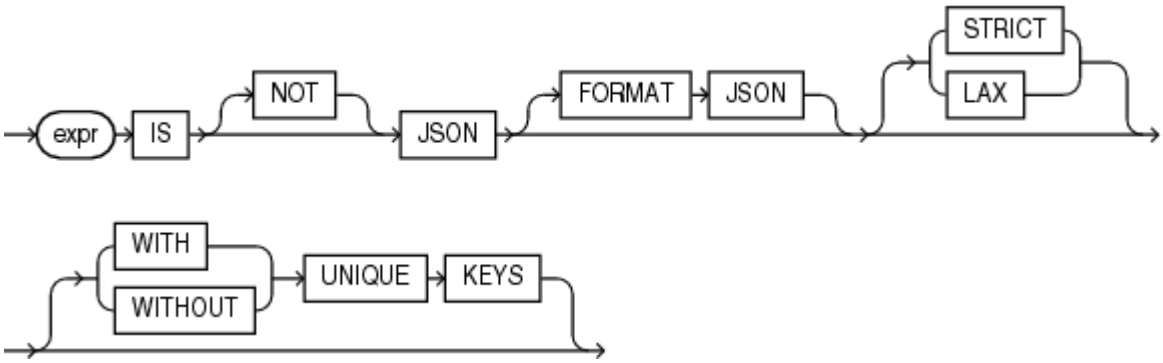


IS JSON条件

このSQL/JSON条件を使用して、式が構文的に正しい整形式のJSONデータかどうかをテストします。

- IS JSONを指定すると、この条件は式が整形式のJSONデータである場合にTRUEを戻し、式が整形式のJSONデータでない場合にFALSEを戻します。
- IS NOT JSONを指定すると、この条件は式が整形式のJSONデータでない場合にTRUEを戻し、式が整形式のJSONデータである場合にFALSEを戻します。

is_JSON_condition::=



- exprは、評価の対象となるJSONデータを指定するために使用します。テキスト・リテラルを評価する式を指定します。exprが列である場合、列のデータ型はVARCHAR2、CLOBまたはBLOBのいずれかである必要があります。exprがnullまたは長さゼロのテキスト・リテラルを評価する場合、この条件はUNKNOWNを戻します。
- exprがデータ・タイプBLOBの列の場合、FORMAT JSONを指定する必要があります。
- STRICTを指定すると、この条件は厳密なJSON構文のみを整形式のJSONデータとみなします。LAXを指定すると、この条件は緩いJSON構文のみを整形式のJSONデータとみなします。デフォルトはLAXです。厳密なJSON構文および緩いJSON構文の詳細は、Oracle Database JSON開発者ガイドを参照してください。
- WITH UNIQUE KEYSを指定すると、キー名が各オブジェクト内で一意である場合のみ、この条件は整形式のJSON

データとみなします。WITHOUT UNIQUE KEYSを指定すると、キー名がオブジェクト内で重複する場合、この条件は整形式のJSONデータとみなします。WITHOUT UNIQUE KEYSテストは、WITH UNIQUE KEYSテストより高速に実行されます。デフォルトはWITHOUT UNIQUE KEYSです。

例

厳密なJSON構文または緩いJSON構文のテスト: 例

次の文は、col1列を使用した表tを作成します。

```
CREATE TABLE t (col1 VARCHAR2(100));
```

次の文は、値を表tの列col1に挿入します。

```
INSERT INTO t VALUES ( '[ "LIT192", "CS141", "HIS160" ]' );
INSERT INTO t VALUES ( '{ "Name": "John" }' );
INSERT INTO t VALUES ( '{ "Grade Values" : { A : 4.0, B : 3.0, C : 2.0 } }');
INSERT INTO t VALUES ( '{ "isEnrolled" : true }' );
INSERT INTO t VALUES ( '{ "isMatriculated" : False }' );
INSERT INTO t VALUES (NULL);
INSERT INTO t VALUES ('This is not well-formed JSON data');
```

次の文は、表tを問い合せて、整形式のJSONデータであるcol1値を戻します。STRICTおよびLAXキーワードが指定されていないため、この例はデフォルトのLAX設定を使用します。このため、この問合せは、厳密なJSON構文または緩いJSON構文を使用する値を戻します。

```
SELECT col1
  FROM t
  WHERE col1 IS JSON;
COL1
--------------------------------------------------
[ "LIT192", "CS141", "HIS160" ]
{ "Name": "John" }
{ "Grade Values" : { A : 4.0, B : 3.0, C : 2.0 } }
{ "isEnrolled" : true }
{ "isMatriculated" : False }
```

次の文は、表tを問い合せて、整形式のJSONデータであるcol1値を戻します。この例は、STRICT設定を指定します。このため、この問合せは、厳密なJSON構文を使用する値のみ戻します。

```
SELECT col1
  FROM t
  WHERE col1 IS JSON STRICT;
COL1
--------------------------------------------------
[ "LIT192", "CS141", "HIS160" ]
{ "Name": "John" }
{ "isEnrolled" : true }
```

次の文は表tを問い合せて緩いJSON構文を使用するcol1値を戻しますが、厳密なJSON構文を使用するcol1値を省略します。このため、この問合せは、緩いJSON構文で使用できる例外を含む値のみ戻します。

```
SELECT col1
  FROM t
  WHERE col1 IS NOT JSON STRICT AND col1 IS JSON LAX;
COL1
--------------------------------------------------
{ "Grade Values" : { A : 4.0, B : 3.0, C : 2.0 } }
{ "isMatriculated" : False }
```

一意キーのテスト: 例

次の文は、col1列を使用した表tを作成します。

```
CREATE TABLE t (col1 VARCHAR2(100));
```

次の文は、値を表tの列col1に挿入します。

```
INSERT INTO t VALUES ('{a:100, b:200, c:300}');
INSERT INTO t VALUES ('{a:100, a:200, b:300}');
INSERT INTO t VALUES ('{a:100, b : {a:100, c:300}}');
```

次の文は、表tを問い合せて各オブジェクト内に一意のキー名を持つ整形式のJSONデータであるcol1値を戻します。

```
SELECT col1 FROM t
  WHERE col1 IS JSON WITH UNIQUE KEYS;
COL1
----------------------------
{a:100, b:200, c:300}
{a:100, b : {a:100, c:300}}
```

キー名aが2回表示され、2つの異なるオブジェクトにあるため、2番目の行が戻されます。

次の文は、表tを問い合せて、各オブジェクト内に一意のキー名があるかどうかに関係なく整形式のJSONデータであるcol1値を戻します。

```
SELECT col1 FROM t
  WHERE col1 IS JSON WITHOUT UNIQUE KEYS;
COL1
----------------------------
{a:100, b:200, c:300}
{a:100, a:200, b:300}
{a:100, b : {a:100, c:300}}
```

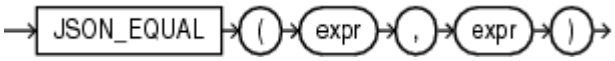
CHECK制約としてのIS JSONの使用方法: 例

次の文は、表j_purchaseorderを作成し、JSONデータを列po_documentに格納します。この文は、CHECK制約としてIS JSON条件を使用し、整形式のJSONのみ列po_documentに格納します。

```
CREATE TABLE j_purchaseorder
  (id RAW (16) NOT NULL,
   date_loaded TIMESTAMP(6) WITH TIME ZONE,
   po_document CLOB CONSTRAINT ensure_json CHECK (po_document IS JSON));
```

JSON_EQUAL条件

構文



目的

Oracle SQL条件JSON_EQUALは2つのJSON値を比較してtrueを戻します。2つの値が同じでない場合、falseを返します。入力値は有効なJSONデータである必要があります。

この比較では、意味のない空白と、意味のないオブジェクト・メンバーの順序は無視されます。たとえば、JSONオブジェクトに同じメンバーがある場合、その順序に関係なく、これらのオブジェクトは等しくなります。

比較された2つの入力のいずれかに1つ以上のフィールドの重複がある場合、JSON_EQUALによって戻される値は特定されません。

JSON_EQUALでは、ERROR ON ERROR、FALSE ON ERRORおよびTRUE ON ERRORがサポートされています。デフォルトはFALSE ON ERRORです。エラーの典型的な例は、入力式が有効なJSONでない場合です。

例

次の文はTRUEを戻します。

```
JSON_EQUAL('{}', '{ }')

JSON_EQUAL('{a:1, b:2}', '{b:2 , a:1 }')
```

次の文はFALSEを返します。

```
JSON_EQUAL('{a:"1"}', '{a:1 }') -> FALSE
```

次の文は、ORA-40441 JSON構文エラーになります

```
JSON_EQUAL('[1]', '[}' ERROR ON ERROR)
```

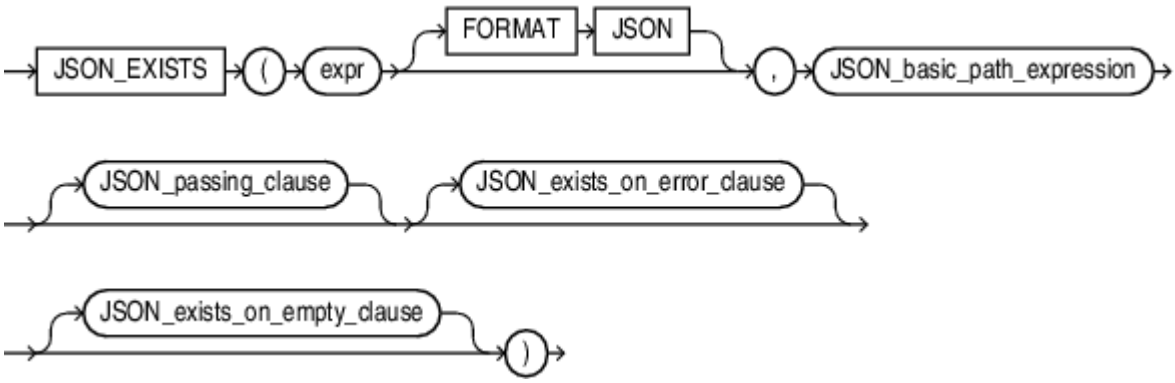
関連項目:

- 詳細は、Oracle Database JSON開発者ガイド を参照してください

JSON_EXISTS条件

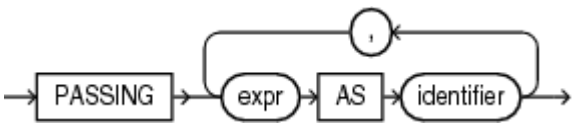
SQL/JSON条件JSON_EXISTSを使用して、指定されたJSON値がJSONデータにあるかどうかをテストします。この条件は、JSON値が存在する場合にTRUEを戻し、JSON値が存在しない場合にFALSEを戻します。

JSON_exists_condition::=

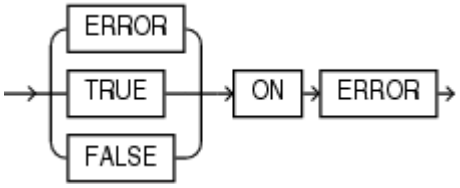


(JSON_basic_path_expression: Oracle Database JSON開発者ガイドを参照)

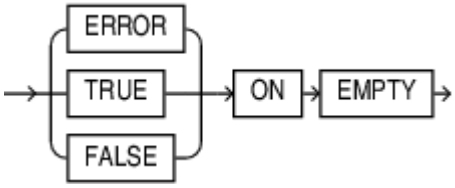
JSON_passing_clause::=



JSON_exists_on_error_clause::=



JSON_exists_on_empty_clause::=



expr

この句は、評価の対象となるJSONデータを指定するために使用します。exprでは、テキスト・リテラルを評価する式を指定します。exprが列である場合、列のデータ型はVARCHAR2、CLOBまたはBLOBのいずれかである必要があります。exprがnullまたは長さゼロのテキスト・リテラルを評価する場合、この条件はUNKNOWNを戻します。

exprが厳密なまたは緩い構文を使用した整形式のJSONデータのテキスト・リテラルでない場合、この条件はデフォルトでFALSEを戻します。JSON_exists_on_error_clauseを使用して、このデフォルトの動作をオーバーライドできます。JSON_exists_on_error_clauseを参照してください。

FORMAT JSON

exprがデータ・タイプBLOBの列の場合、FORMAT JSONを指定する必要があります。

JSON_basic_path_expression

この句を使用して、SQL/JSONパス式を指定します。この条件はパス式を使用してexprを評価し、パス式と一致する(パス式を満たす)JSON値があるかどうかを判断します。パス式はテキスト・リテラルである必要がありますが、パス式には、JSON_passing_clauseによって値がパス式に渡される変数を含めることができます。JSON_basic_path_expressionのセマンティクスの詳細は、Oracle Database JSON開発者ガイドを参照してください。

JSON_passing_clause

この句を使用して、値をパス式に渡します。exprには、VARCHAR2、NUMBER、BINARY_DOUBLE、DATE、TIMESTAMPまたはTIMESTAMP WITH TIME ZONEデータ型の値を指定します。exprの評価結果は、JSON_basic_path_expressionの対応する識別子にバインドされます。

JSON_exists_on_error_clause

この句を使用して、exprが整形式のJSONデータでない場合にこの条件で戻される値を指定します。

次の句を指定できます。

- ERROR ON ERROR - exprが整形式のJSONデータでない場合、適切なOracleエラーを戻します。
- TRUE ON ERROR - exprが整形式のJSONデータでない場合にTRUEを戻します。
- FALSE ON ERROR - exprが整形式のJSONデータでない場合にFALSEを戻します。これはデフォルトです。

JSON_exists_on_empty_clause

この句を使用して、JSONデータがSQL/JSONパス式を使用して評価されるときに一致が見つからない場合にこのファンクション

で戻される値を指定します。この句により、JSON_exists_on_error_clauseで指定された結果とは異なる、このタイプのエラーに対する結果を指定できます。

次の句を指定できます。

- NULL ON EMPTY - 一致が見つからない場合にNULLを戻します。
- ERROR ON EMPTY - 一致が見つからない場合に適切なOracleエラーを戻します。
- DEFAULT literal ON EMPTY - 一致が見つからない場合にliteralを戻します。literalのデータ型は、このファンクションにより戻される値のデータ型と一致する必要があります。

この句を省略すると、JSON_exists_on_error_clauseによって、一致が見つからない場合に戻される値が決まります。

例

次の文は、name列を使用した表tを作成します。

```
CREATE TABLE t (name VARCHAR2(100));
```

次の文は、値を表tの列nameに挿入します。

```
INSERT INTO t VALUES ('[{first:"John"}, {middle:"Mark"}, {last:"Smith"}]');
INSERT INTO t VALUES ('[{first:"Mary"}, {last:"Jones"}]');
INSERT INTO t VALUES ('[{first:"Jeff"}, {last:"Williams"}]');
INSERT INTO t VALUES ('[{first:"Jean"}, {middle:"Anne"}, {last:"Brown"}]');
INSERT INTO t VALUES (NULL);
INSERT INTO t VALUES ('This is not well-formed JSON data');
```

次の文は表tの列nameを問い合せて、最初の要素がプロパティ名firstのオブジェクトである配列で構成されるJSONデータを戻します。ON ERROR句は指定されません。このため、JSON_EXISTS条件は、整形式のJSONデータでない値に対してFALSEを戻します。

```
SELECT name FROM t
  WHERE JSON_EXISTS(name, '$[0].first');
NAME
--------------------------------------------------
[{first:"John"}, {middle:"Mark"}, {last:"Smith"}]
[{first:"Mary"}, {last:"Jones"}]
[{first:"Jeff"}, {last:"Williams"}]
[{first:"Jean"}, {middle:"Anne"}, {last:"Brown"}]
```

次の文は表tの列nameを問い合せて、2番目の要素がプロパティ名middleのオブジェクトである配列で構成されるJSONデータを戻します。ON ERROR句は指定されません。このため、JSON_EXISTS条件は、整形式のJSONデータでない値に対してFALSEを戻します。

```
SELECT name FROM t
  WHERE JSON_EXISTS(name, '$[1].middle');
NAME
--------------------------------------------------------------------------------
[{first:"John"}, {middle:"Mark"}, {last:"Smith"}]
[{first:"Jean"}, {middle:"Anne"}, {last:"Brown"}]
```

次の文は前の文と似ていますが、TRUE ON ERROR句が指定されている点が異なります。このため、JSON_EXISTS条件は、整形式のJSONデータでない値に対してTRUEを戻します。

```
SELECT name FROM t
  WHERE JSON_EXISTS(name, '$[1].middle' TRUE ON ERROR);
NAME
--------------------------------------------------------------------------------
[{first:"John"}, {middle:"Mark"}, {last:"Smith"}]
```

```
[{first:"Jean"}, {middle:"Anne"}, {last:"Brown"}]
This is not well-formed JSON data
```

次の文は表tの列nameを問い合せて、プロパティ名lastのオブジェクトである要素を含む配列で構成されるJSONデータを戻します。ワイルドカード記号(*)が配列索引に指定されます。このため、配列の索引番号に関係なく、問合せは、そのようなオブジェクトを含む配列を戻します。

```
SELECT name FROM t
  WHERE JSON_EXISTS(name, '$[*].last');
NAME
--------------------------------------------------
[{first:"John"}, {middle:"Mark"}, {last:"Smith"}]
[{first:"Mary"}, {last:"Jones"}]
[{first:"Jeff"}, {last:"Williams"}]
[{first:"Jean"}, {middle:"Anne"}, {last:"Brown"}]
```

次の文は、passing句を使用してフィルタ式を実行します。比較述語(`@.middle == $var1`)のSQL/JSON変数`$var1`は、PASSING句のバインド変数var1からその値を取得します。

値の比較にバインド変数を使用すると、問合せの再コンパイルが回避されます。

```
SELECT name FROM t
  WHERE JSON_EXISTS(name, '$[1]?(@.middle == $var1)' PASSING 'Anne' as "var1");
NAME
--------------------------------------------------------------------------------
[{first:"Jean"}, {middle:"Anne"}, {last:"Brown"}]
```

関連項目:

条件JSON_EXISTS

JSON_TEXTCONTAINS条件

SQL/JSON条件JSON_TEXTCONTAINSを使用して、指定された文字列がJSONプロパティ値にあるかどうかをテストします。この条件を使用して、特定のワードまたは数値でJSONデータをフィルタ処理できます。

この条件では次の引数を使用します。

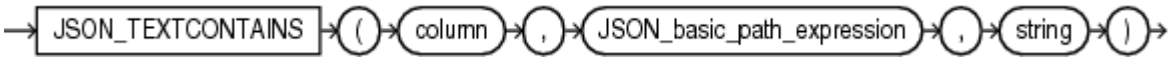
- JSONデータを含む表またはビューの列。JSONデータでの使用に特化して設計されているOracle Textデータである JSON検索索引を列に定義する必要があります。列のJSONデータの各行は、JSONドキュメントといいます。
- SQL/JSONパス式。ドキュメント内の特定のJSONオブジェクトと一致するかどうかの試行で、パス式が各JSONドキュメントに適用されます。パス式にはJSONオブジェクト・ステップのみ含むことができ、JSON配列ステップを含むことはできません。
- 文字列。この条件は、配列値を含む一致したJSONオブジェクトのすべての文字列および数値のプロパティ値内の文字列を検索します。文字列は、プロパティ値の個別のワードとして存在する必要があります。たとえば、'beth'を検索すると、文字列のプロパティ値"beth smith"と一致しますが、"elizabeth smith"とは一致しません。'10'を検索すると、数値のプロパティ値10または文字列のプロパティ値"10 main street"と一致しますが、数値のプロパティ値110または文字列のプロパティ値"102 main street"とは一致しません。

この条件は、一致する場合にTRUEを戻し、一致しない場合にFALSEを戻します。

関連項目:

JSON全文検索問合せ

JSON_textcontains_condition::=



(JSON_basic_path_expression: Oracle Database JSON開発者ガイドを参照)

column

テストするJSONデータを含む表またはビューの列の名前を指定します。列のデータ型はVARCHAR2、CLOBまたはBLOBである必要があります。JSONデータでの使用に特化して設計されているOracle TextデータであるJSON検索索引を列に定義する必要があります。列の値がnullまたは長さゼロのテキスト・リテラルである場合、この条件はUNKNOWNを戻します。

列の値が厳密な構文または緩い構文を使用した整形式のJSONデータのテキスト・リテラルでない場合、この条件はFALSEを戻します。

JSON_basic_path_expression

この句を使用して、SQL/JSONパス式を指定します。この条件はパス式を使用してcolumnを評価し、パス式と一致する(パス式を満たす)JSON値があるかどうかを判断します。パス式はテキスト・リテラルである必要があります。JSON_basic_path_expressionのセマンティクスの詳細は、Oracle Database JSON開発者ガイドを参照してください。

string

この条件は、stringで指定された文字列を検索します。文字列は一重引用符で囲みます。

例

次の文は、family_doc列を使用した表familiesを作成します。

```
CREATE TABLE families (family_doc VARCHAR2(200));
```

次の文は、列family_docのJSON検索索引を作成します。

```
CREATE INDEX ix
  ON families(family_doc)
  INDEXTYPE IS CTXSYS.CONTEXT
  PARAMETERS ('SECTION GROUP CTXSYS.JSON_SECTION_GROUP SYNC (ON COMMIT)');
```

次の文は、ファミリを示すJSONドキュメントを列family_docに挿入します。

```
INSERT INTO families
VALUES ('{family : {id:10, ages:[40,38,12], address : {street : "10 Main
Street"}}}');
INSERT INTO families
VALUES ('{family : {id:11, ages:[42,40,10,5], address : {street : "200 East Street",
apt : 20}}}');
INSERT INTO families
VALUES ('{family : {id:12, ages:[25,23], address : {street : "300 Oak Street", apt :
10}}}');
```

次の文は、トランザクションをコミットします。

```
COMMIT;
```

次の問合せは、ドキュメントのプロパティ値に10を含むJSONドキュメントを戻します。

```
SELECT family_doc FROM families
  WHERE JSON_TEXTCONTAINS(family_doc, '$', '10');
FAMILY_DOC
--------------------------------------------------------------------------------
{family : {id:10, ages:[40,38,12], address : {street : "10 Main Street"}}}
{family : {id:11, ages:[42,40,10,5], address : {street : "200 East Street", apt :
20}}}
{family : {id:12, ages:[25,23], address : {street : "300 Oak Street", apt : 10}}}
```

次の問合せは、idのプロパティ値に10を含むJSONドキュメントを戻します。

```
SELECT family_doc FROM families
  where json_textcontains(family_doc, '$.family.id', '10');
FAMILY_DOC
--------------------------------------------------------------------------------
{family : {id:10, ages:[40,38,12], address : {street : "10 Main Street"}}}
```

次の問合せは、agesプロパティの値の配列に10を含むJSONドキュメントを戻します。

```
SELECT family_doc FROM families
  WHERE JSON_TEXTCONTAINS(family_doc, '$.family.ages', '10');
FAMILY_DOC
--------------------------------------------------------------------------------
{family : {id:11, ages:[42,40,10,5], address : {street : "200 East Street", apt :
20}}}
```

次の問合せは、addressプロパティ値に10を含むJSONドキュメントを戻します。

```
SELECT family_doc FROM families
  WHERE JSON_TEXTCONTAINS(family_doc, '$.family.address', '10');
FAMILY_DOC
--------------------------------------------------------------------------------
{family : {id:10, ages:[40,38,12], address : {street : "10 Main Street"}}}
{family : {id:12, ages:[25,23], address : {street : "300 Oak Street", apt : 10}}}
```

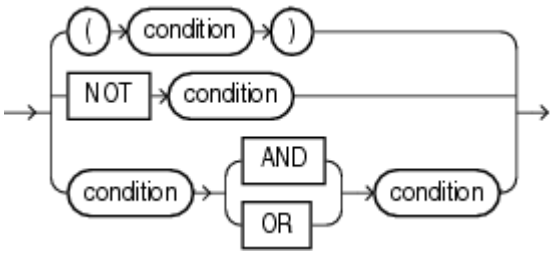
次の問合せは、aptプロパティ値に10を含むJSONドキュメントを戻します。

```
SELECT family_doc FROM families
  WHERE JSON_TEXTCONTAINS(family_doc, '$.family.address.apt', '10');
FAMILY_DOC
--------------------------------------------------------------------------------
{family : {id:12, ages:[25,23], address : {street : "300 Oak Street", apt : 10}}}
```

複合条件

複合条件は他の条件の組合せを指定します。

compound_condition::=



関連項目:

NOT、ANDおよびOR条件の詳細は、「論理条件」を参照してください

BETWEEN条件

BETWEEN条件は、1つの式の値が、他の2つの式によって定義された期間内に存在するかどうかを決定します。

between_condition::=

3つのすべての式は、数式、文字式または日時式である必要があります。SQLでは、expr1を複数回評価できます。BETWEEN式がPL/SQLで使用されている場合、expr1は1回のみ評価されることが保証されます。すべての式が同じデータ型ではない場合、式は共通のデータ型に暗黙的に変換されます。それが不可能な場合は、エラーが戻されます。

関連項目:

SQLのデータ型変換の詳細は、暗黙的なデータ変換を参照してください。

次の式の値について考えてみます。

```
expr1 NOT BETWEEN expr2 AND expr3
```

この式の値は次の式の値と同じです。

```
NOT (expr1 BETWEEN expr2 AND expr3)
```

さらに次の式の値について考えてみます。

```
expr1 BETWEEN expr2 AND expr3
```

この式の値は次のブール式の値と同じです。

```
expr2 <= expr1 AND expr1 <= expr3
```

expr3 < expr2の場合、この期間は空です。expr1がNULLの場合、結果はNULLとなります。expr1がNULLではない場合、値は通常はFALSEであり、キーワードNOTが使用された場合はTRUEとなります。

ブール演算子ANDを使用すると、予期しない結果となる場合があります。特に、式x AND yでは、条件x IS NULLは式の値を決定するのに十分ではありません。2番目のオペランドも評価する必要があります。2番目のオペランドの値がFALSEの場合、結果はFALSEであり、そうではない場合はNULLとなります。ANDの詳細は、論理条件を参照してください。

表6-10 BETWEEN条件

条件の種類	操作	例
[NOT] BETWEEN x AND y	[NOT](expr2 が expr1 以下、AND expr1 が expr3 以下)	SELECT * FROM employees WHERE salary BETWEEN 2000 AND 3000 ORDER BY employee_id;

EXISTS条件

EXISTS条件は、副問合せに行が存在するかどうかをテストします。

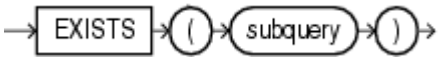


表6-11に、EXISTS条件を示します。

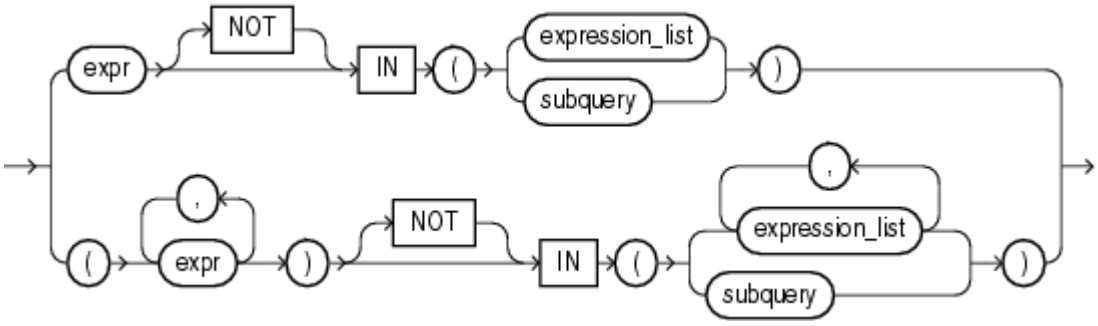
表6-11 EXISTS条件

条件の種類	操作	例
EXISTS	副問合せによって行が 1 行以上戻される場合には、TRUE と評価されます。	`SELECT department_id FROM departments d WHERE EXISTS (SELECT * FROM employees e WHERE d.department_id = e.department_id) ORDER BY department_id;`

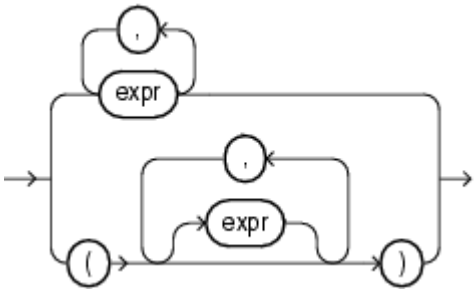
IN条件

in_conditionはメンバーシップ条件です。メンバーシップ条件は、値のリストまたは副問合せ内のメンバーシップをテストします。

in_condition::=



expression_list::=



in_condition条件の上の方の書式(演算子の左辺に1つの式を指定する書式)を使用する場合は、expression_listの上の方の書式を使用する必要があります。この条件の下の方の書式(演算子の左辺に複数の式を指定する書式)を使用する場合は、expression_listの下の方の書式を使用し、各expression_list内の式は、演算子の左辺にある式と同じ数とデータ型で構成されている必要があります。expression_listには、最大1000個の式を指定できます。

expression_listの式は、INリスト内の順序で評価されるとはかぎりません。ただし、副問合せのSELECT構文のリストにある式は、指定された順序で評価されます。

関連項目:

式のリスト

表6-12に、IN条件の書式を示します。

表6-12 IN条件

条件の種類	操作	例
IN	いずれかのメンバーとの等価テスト。=ANYと同じです。	SELECT * FROM employees WHERE job_id IN ('PU_CLERK','SH_CLERK') ORDER BY employee_id; SELECT * FROM employees WHERE salary IN (SELECT

条件の種類	操作	例
		salary FROM employees WHERE department_id =30) ORDER BY employee_id;
NOT IN	!=ALL と同じです。セット内のいずれかのメンバーが NULL の場合は FALSE に評価されます。	SELECT * FROM employees WHERE salary NOT IN (SELECT salary FROM employees WHERE department_id = 30) ORDER BY employee_id; SELECT * FROM employees WHERE job_id NOT IN ('PU_CLERK', 'SH_CLERK') ORDER BY employee_id;

NOT IN演算子に続くリストの中のいずれかの項目がNULLの場合は、すべての行はFALSEまたは不明(UNKNOWN)と評価されます(行は戻されません)。たとえば、次の文ではそれぞれの行に対して文字列'True'が戻されます。

```
SELECT 'True' FROM employees
   WHERE department_id NOT IN (10, 20);
```

ただし、次の文では行は戻されません。

```
SELECT 'True' FROM employees
    WHERE department_id NOT IN (10, 20, NULL);
```

この例で行が戻されないのは、WHERE句の条件が次のように評価されるためです。

```
department_id != 10 AND department_id != 20 AND department_id != null
```

3番目の条件でdepartment_idとNULLの比較が行われるため、この条件の結果がUNKNOWNとなり、式全体の結果がFALSE(department_idを持つ行が10または20と等しいため)となります。特に、NOT IN演算子が副問合せを参照するときは、このような動作を見逃してしまう可能性があることに注意してください。

また、NOT IN条件が、行を戻さない副問合せを参照する場合は、次のように、すべての行が戻されます。

```
SELECT 'True' FROM employees
   WHERE department_id NOT IN (SELECT 0 FROM DUAL WHERE 1=2);
```

文字引数の場合、IN条件は照合依存です。照合決定ルールによって、使用する照合が決まります。

関連項目:

IN条件の照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

WHERE句のLEVELの制限事項

WHERE句の[NOT] IN条件の右辺が副問合せになっている場合、条件の左辺でLEVELは使用できません。ただし、FROM句の副問合せでLEVELを指定すると、同じ結果が得られます。たとえば、次の文は無効です。

```
SELECT employee_id, last_name FROM employees
   WHERE (employee_id, LEVEL)
      IN (SELECT employee_id, 2 FROM employees)
   START WITH employee_id = 2
   CONNECT BY PRIOR employee_id = manager_id;
```

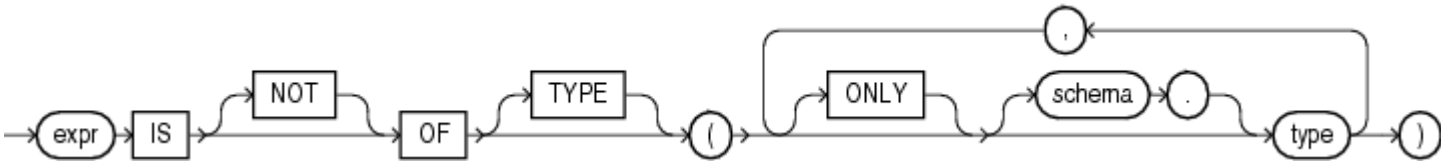
ただし、次の文は、LEVEL情報を含む問合せをFROM句にカプセル化するため有効です。

```
SELECT v.employee_id, v.last_name, v.lev FROM
      (SELECT employee_id, last_name, LEVEL lev
      FROM employees v
      START WITH employee_id = 100
      CONNECT BY PRIOR employee_id = manager_id) v
   WHERE (v.employee_id, v.lev) IN
      (SELECT employee_id, 2 FROM employees);
```

IS OF type条件

IS OF type条件を使用すると、固有の型情報に基づくオブジェクト・インスタンスをテストできます。

is_of_type_condition::=



typeによって参照されるすべての型に対するEXECUTE権限が必要です。また、すべてのtypeは、同じ型ファミリに属している必要があります。

exprがNULLである場合、この条件はNULLと評価されます。exprがNULLでない場合、次に示す環境では、この条件はTRUE(NOTキーワードを指定した場合はFALSE)と評価されます。

- exprの型が、typeリストで指定された型のサブタイプであり、ONLYを指定していない場合
- exprの型がtypeリストで明示的に指定されている場合。

exprは、相関変数を伴うVALUEファンクションの書式をとる場合があります。

次の例では、置換可能な表および列のサンプルの型階層に基づいて作成されたサンプル表oe.personsを使用します。この例は、IS OF type条件を使用して、問合せを特定のサブタイプに制限します。

```
SELECT * FROM persons p
   WHERE VALUE(p) IS OF TYPE (employee_t);
NAME                       SSN
--------------------------
Joe                      32456
Tim                       5678
SELECT * FROM persons p
   WHERE VALUE(p) IS OF (ONLY part_time_emp_t);
NAME                       SSN
--------------------------
Tim                       5678
```

7 ファンクション

ファンクションは、データ項目を操作し、結果を戻すという点で演算子に似ています。ファンクションと演算子は引数を指定する書式が異なります。次の書式によって、ファンクションでは0(ゼロ)以上の引数を操作できます。

```
function(argument, argument, ...)
```

引数を持たない関数は疑似列と同じです(「疑似列」を参照)。ただし、疑似列は、通常、結果セット内の各行に対して異なる値を戻しますが、引数を指定しないファンクションは、各行に対して同じ値を戻します。

この章では、次の内容を説明します。

- ユーザー定義ファンクション

SQLファンクション

SQLファンクションは、Oracle Databaseに組み込まれており、適切なSQL文で使用できます。SQLファンクションと、PL/SQLで記述されたユーザー定義ファンクションを混同しないでください。

SQLファンクションが戻す値のデータ型以外のデータ型の引数でSQLファンクションをコールすると、OracleはSQLファンクションを実行する前に、その引数を必要なデータ型に変換します。

関連項目:

ユーザー・ファンクションの詳細は、ユーザー定義ファンクションを参照してください。データ型の暗黙的な変換の詳細は、データ変換を参照してください。

SQLファンクションでのNULL

ほとんどのスカラー・ファンクションでは、引数としてNULLを指定するとNULLが戻されます。NVLファンクションを使用した場合、NULLが発生したときに値を戻すことができます。たとえば、式`NVL(commission_pct,0)`は、`commission_pct`がNULLの場合は0(ゼロ)を戻し、`commission_pct`がNULLでなければその値を戻します。

集計ファンクションによるNULLの処理の詳細は、集計ファンクションを参照してください。

SQLファンクションの構文

SQLファンクションの構文図では、データ型とともに引数が示されています。SQL構文にパラメータ`function`が指定されている場合は、この項で説明するファンクションの1つに置き換えます。ファンクションは、引数のデータ型および戻り値によってグループ化されています。

ノート:

LOB 列に SQL ファンクションを適用すると、Oracle Database は、SQL および PL/SQL の処理中に一時 LOB を作成します。ご使用のアプリケーションの一時 LOB を格納するために、十分な一時表領域が割り当てられていることを確認してください。

SQLファンクションは照合依存で、つまり、それによって実行される文字値の比較や一致は照合によって制御されます。ファンクションで使用する特定の照合は、ファンクションの引数の照合に基づいて決定されます。

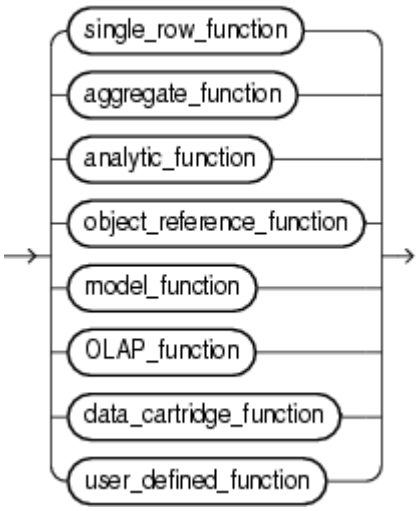
SQLファンクションの結果に文字データ型が含まれる場合、照合導出ルールによって、結果に関連付ける照合が定義されます。

関連項目:

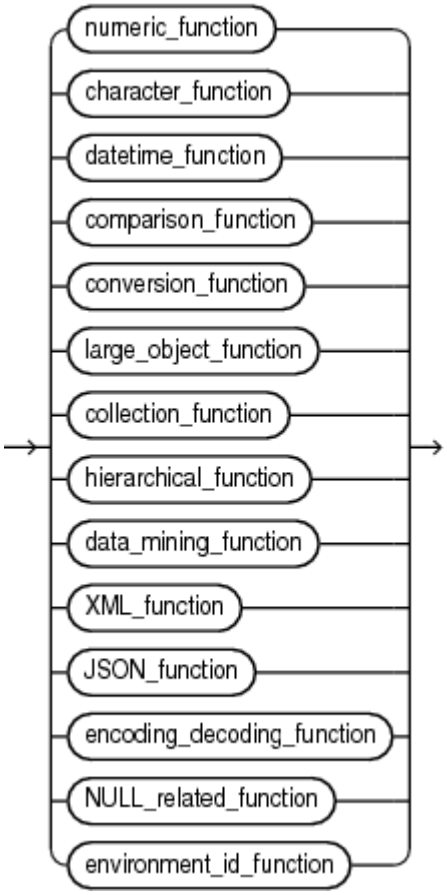
SQLファンクション用の照合の導出および決定のルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

次に、ファンクションのカテゴリを表す構文を示します。

function::=



single_row_function::=



次の項に、前述の図のユーザー定義ファンクション以外の各グループにおける組込みSQLファンクションを示します。すべての組込みSQLファンクションを、アルファベット順に説明します。

関連項目:

ユーザー定義ファンクションおよびCREATE FUNCTIONを参照してください。

単一行ファンクション

単一行ファンクションは、問合せ対象の表またはビューの各行に対して1つの結果行を戻します。これらのファンクションは、SELECT構文のリスト、WHERE句、START WITH句、CONNECT BY句およびHAVING句に指定できます。

数値ファンクション

数値ファンクションは入力として数値を受け取り、結果として数値を戻します。数値ファンクションのほとんどは、38桁(10進)のNUMBER値を戻します。超越関数(COS、COSH、EXP、LN、LOG、SIN、SINH、SQRT、TANおよびTANH)は、36桁(10進)の値を戻します。超越関数のACOS、ASIN、ATAN、ATAN2は、30桁(10進)の値を戻します。数値ファンクションを次に示します。

- ABS
- ACOS
- ASIN
- ATAN
- ATAN2
- BITAND
- CEIL
- COS
- COSH
- EXP
- FLOOR
- LN
- LOG
- MOD
- NANVL
- POWER
- REMAINDER
- ROUND (数値)
- SIGN
- SIN
- SINH
- SQRT
- TAN
- TANH
- TRUNC (数値)
- WIDTH_BUCKET

文字値を戻す文字ファンクション

文字値を戻す文字ファンクションは、特に指定がないかぎり、次のデータ型の値を戻します。

- 入力引数がCHARまたはVARCHAR2の場合、戻される値はVARCHAR2になります。
- 入力引数がNCHARまたはNVARCHAR2の場合、戻される値はNVARCHAR2になります。

ファンクションによって戻される値の長さは、戻されるデータ型の最大長によって制限されます。

- CHARまたはVARCHAR2型の値を戻すファンクションの戻り値の長さが制限を超えた場合、Oracle Databaseは戻り値から制限を超えた部分を切り捨てて、エラー・メッセージを表示せずにその結果を戻します。
- CLOB型の値を戻すファンクションの戻り値の長さが制限を超えた場合、Oracleはエラーを表示し、データを戻しません。

文字値を戻す文字ファンクションを次に示します。

- CHR
- CONCAT
- INITCAP
- LOWER
- LPAD
- LTRIM
- NCHR
- NLS_INITCAP
- NLS_LOWER
- NLS_UPPER
- NLSSORT
- REGEXP_REPLACE
- REGEXP_SUBSTR
- REPLACE
- RPAD
- RTRIM
- SOUNDEX
- SUBSTR
- TRANSLATE
- TRANSLATE ... USING
- TRIM
- UPPER

数値を戻す文字ファンクション

数値を戻す文字ファンクションの引数には、すべての文字データ型を指定できます。数値を戻す文字ファンクションを次に示します。

- ASCII
- INSTR
- LENGTH
- REGEXP_COUNT
- REGEXP_INSTR

文字セット・ファンクション

文字セット・ファンクションは、文字セットの情報を戻します。文字セット・ファンクションを次に示します。

- NLS_CHARSET_DECL_LEN
- NLS_CHARSET_ID
- NLS_CHARSET_NAME

照合ファンクション

照合ファンクションは、照合設定の情報を戻します。照合ファンクションを次に示します。

- COLLATION
- NLS_COLLATION_ID
- NLS_COLLATION_NAME

日時ファンクション

日時ファンクションは、日付(DATE)、タイムスタンプ(TIMESTAMP、TIMESTAMP WITH TIME ZONE、TIMESTAMP WITH LOCAL TIME ZONE)および期間(INTERVAL DAY TO SECOND、INTERVAL YEAR TO MONTH)の値を操作します。

一部の日付ファンクションは、OracleのDATEデータ型(ADD_MONTHS、CURRENT_DATE、LAST_DAY、NEW_TIMEおよびNEXT_DAY)用に設計されています。これらのファンクションの引数にタイムスタンプ値を指定すると、Oracle Databaseは入力された型をDATE値に内部的に変換し、DATE値を戻します。ただし、MONTHS_BETWEENファンクションは数値を戻し、ROUNDおよびTRUNCファンクションはタイムスタンプ値または期間値を受け入れません。

その他の日時ファンクションは、3種類のすべてのデータ型(日付、タイムスタンプ、期間)を受け入れ、それらのいずれかのデータ型の値を戻すように設計されています。

SYSDATE、SYSTIMESTAMP、CURRENT_TIMESTAMPなどの、現在のシステム日時情報を戻す日時ファンクションはすべて、SQL文で参照される回数に関係なく、SQL文ごとに1回評価されます。

日時ファンクションを次に示します。

- ADD_MONTHS
- CURRENT_DATE
- CURRENT_TIMESTAMP
- DBTIMEZONE
- EXTRACT (日時)
- FROM_TZ
- LAST_DAY
- LOCALTIMESTAMP
- MONTHS_BETWEEN
- NEW_TIME
- NEXT_DAY
- NUMTODSINTERVAL
- NUMTOYMINTERVAL
- ORA_DST_AFFECTED
- ORA_DST_CONVERT
- ORA_DST_ERROR
- ROUND (日付)
- SESSIONTIMEZONE

- SYS_EXTRACT_UTC
- SYSDATE
- SYSTIMESTAMP
- TO_CHAR (日時)
- TO_DSINTERVAL
- TO_TIMESTAMP
- TO_TIMESTAMP_TZ
- TO_YMINTERVAL
- TRUNC (日付)
- TZ_OFFSET

一般的な比較ファンクション

一般的な比較ファンクションは、値の集合から最大値または最小値(あるいはその両方)を決定します。一般的な比較ファンクションを次に示します。

- GREATEST
- LEAST

変換ファンクション

変換ファンクションは、あるデータ型から他のデータ型に値を変換します。一般に、ファンクション名は`datatype TO datatype`の書式で指定されます。最初のデータ型は入力データ型です。2番目のデータ型は出力データ型です。SQL変換ファンクションを次に示します。

- ASCIISTR
- BIN_TO_NUM
- CAST
- CHARTOROWID
- COMPOSE
- CONVERT
- DECOMPOSE
- HEXTORAW
- NUMTODSINTERVAL
- NUMTOYMINTERVAL
- RAWTOHEX
- RAWTONHEX
- ROWIDTOCHAR
- ROWIDTONCHAR
- SCN_TO_TIMESTAMP
- TIMESTAMP_TO_SCN
- TO_BINARY_DOUBLE
- TO_BINARY_FLOAT
- TO_BLOB (bfile)
- TO_BLOB (raw)

- TO_CHAR (bfile|blob)
- TO_CHAR (文字)
- TO_CHAR (日時)
- TO_CHAR (数値)
- TO_CLOB (bfile|blob)
- TO_CLOB (文字)
- TO_DATE
- TO_DSINTERVAL
- TO_LOB
- TO_MULTI_BYTE
- TO_NCHAR (文字)
- TO_NCHAR (日時)
- TO_NCHAR (数値)
- TO_NCLOB
- TO_NUMBER
- TO_SINGLE_BYTE
- TO_TIMESTAMP
- TO_TIMESTAMP_TZ
- TO_YMINTERVAL
- TREAT
- UNISTR
- VALIDATE_CONVERSION

ラージ・オブジェクト・ファンクション

ラージ・オブジェクト・ファンクションは、LOBを操作します。ラージ・オブジェクト・ファンクションを次に示します。

- BFILENAME
- EMPTY_BLOB、EMPTY_CLOB

収集ファンクション

収集ファンクションは、ネストした表およびVARRAYを操作します。SQL収集ファンクションを次に示します。

- CARDINALITY
- COLLECT
- POWERMULTISET
- POWERMULTISET_BY_CARDINALITY
- SET

階層ファンクション

階層ファンクションは、階層パス情報を結果セットに適用します。階層ファンクションを次に示します。

- SYS_CONNECT_BY_PATH

データ・マイニング・ファンクション

データ・マイニング・ファンクションでは、データのスコアリングにOracle Advanced Analyticsを使用します。これらの関数は、マイニング・モデルのスキーマ・オブジェクトをデータに適用することも、分析句の実行によって動的にデータをマイニングすることもできます。データ・マイニング・ファンクションは、Oracleのネイティブ・アルゴリズムを使用して構築されたモデル、およびOracle Advanced Analyticsの拡張性メカニズムによるRを使用して構築されたモデルに適用できます。

データ・マイニング・ファンクションを次に示します。

- CLUSTER_DETAILS
- CLUSTER_DISTANCE
- CLUSTER_ID
- CLUSTER_PROBABILITY
- CLUSTER_SET
- FEATURE_COMPARE
- FEATURE_DETAILS
- FEATURE_ID
- FEATURE_SET
- FEATURE_VALUE
- ORA_DM_PARTITION_NAME
- PREDICTION
- PREDICTION_BOUNDS
- PREDICTION_COST
- PREDICTION_DETAILS
- PREDICTION_PROBABILITY
- PREDICTION_SET

関連項目:

- Oracle Data Miningについて学習するには、『Oracle Data Mining概要』を参照してください。
- スコアリングの詳細は、Oracle Data Miningユーザーズ・ガイドを参照してください。

XMLファンクション

XMLファンクションは、XML文書またはフラグメントを操作または戻します。これらのファンクションで使用する引数はANSI/ISO/IEC SQL規格の中には定義されていませんが、World Wide Web Consortium(W3C)標準の中に定義されています。ファンクションで実行される処理および操作は、該当するW3C標準で定義されています。次の表に、これらのXML関連の各引数に適用されるルールとガイドラインについて、W3C標準の該当する項へのリンクを示します。これらのXMLファンクションのいずれかをSQL文で使用する場合、該当するW3Cの構文に準拠していない引数を使用するとエラーになります。データベース列の値として使用できる文字が、XMLでも有効であるとはかぎらないことに注意してください。

構文要素	W3C標準のURL
`value_expr`	http://www.w3.org/TR/2006/REC-xml-20060816

構文要素	W3C標準のURL
`Xpath_string`	http://www.w3.org/TR/1999/REC-xpath-19991116
`XQuery_string`	http://www.w3.org/TR/2007/REC-xquery-semantics-20070123/ http://www.w3.org/TR/xquery-update-10/
`namespace_string`	http://www.w3.org/TR/2006/REC-xml-names-20060816/
`identifier`	http://www.w3.org/TR/2006/REC-xml-20060816/#NT-Nmtoken

出力の書式設定など、これらのファンクションを使用したXMLデータの選択および問合せの詳細は、『Oracle XML DB開発者ガイド』を参照してください

SQL XMLファンクションを次に示します。

- DEPTH
- EXISTSNODE
- EXTRACT (XML)
- EXTRACTVALUE
- PATH
- SYS_DBURIGEN
- SYS_XMLAGG
- SYS_XMLGEN
- XMLAGG
- XMLCAST
- XMLCDATA
- XMLCOLATTVAL
- XMLCOMMENT
- XMLCONCAT
- XMLDIFF
- XMLELEMENT
- XMLEXISTS
- XMLFOREST
- XMLISVALID
- XMLPARSE
- XMLPATCH
- XMLPI
- XMLQUERY
- XMLROOT
- XMLSEQUENCE
- XMLSERIALIZE

- XMLTABLE
- XMLTRANSFORM

JSONファンクション

JavaScript Object Notation (JSON)ファンクションによって、JSONデータを問合せおよびテストできます。

次のSQL/JSONファンクションによって、JSONデータを問い合せることができます。

- JSON_QUERY
- JSON_TABLE
- JSON_VALUE

次のSQL/JSONファンクションによって、JSONデータを生成できます。

- JSON_ARRAY
- JSON_ARRAYAGG
- JSON_OBJECT
- JSON_OBJECTAGG

次のOracle SQLファンクションでは、JSONデータ・ガイドが作成されます。

- JSON_DATAGUIDE

エンコーディング・ファンクションおよびデコーディング・ファンクション

エンコーディング・ファンクションおよびデコーディング・ファンクションでは、データベース内のデータを調査およびデコードできます。エンコーディング・ファンクションおよびデコーディング・ファンクションを次に示します。

- DECODE
- DUMP
- ORA_HASH
- STANDARD_HASH
- VSIZE

NULL関連ファンクション

NULL関連ファンクションは、NULL処理を簡単にします。NULL関連ファンクションを次に示します。

- COALESCE
- LNNVL
- NANVL
- NULLIF
- NVL
- NVL2

環境および識別子関数

環境ファンクションおよび識別子ファンクションは、インスタンスとセッションの情報を提供します。環境ファンクションおよび識別子ファンクションを次に示します。

- CON_DBID_TO_ID
- CON_GUID_TO_ID
- CON_NAME_TO_ID
- CON_UID_TO_ID
- ORA_INVOKING_USER
- ORA_INVOKING_USERID
- SYS_CONTEXT
- SYS_GUID
- SYS_TYPEID
- UID
- USER
- USERENV

集計ファンクション

集計ファンクションは、単一行に基づく結果行を戻すのではなく、行のグループに基づく単一結果行を戻します。集計ファンクションは、SELECT構文のリスト、ORDER BYおよびHAVING句に指定できます。集計ファンクションは、通常、SELECT文のGROUP BY句で使用され、この句の中で問合せ対象の表またはビューの行がグループ化されます。GROUP BY句を含む問合せでは、SELECT構文のリストの要素は、集計ファンクション、GROUP BY式、定数またはこれらのいずれかを含む式になります。Oracleは、集計ファンクションを行の各グループに適用し、各グループに単一の結果行を戻します。

GROUP BY句を指定しないと、SELECT構文のリスト内の集計ファンクションは、問合せ対象の表またはビューのすべての行に適用されます。HAVING句で集計ファンクションを使用して、グループを出力しないこともできます。このとき、出力は、問合せ対象の表またはビューの各行の値ではなく、集計ファンクションの結果に基づきます。

関連項目:

- 問合せおよび副問合せにおけるGROUP BY句およびHAVING句の詳細は、「GROUP BY句の使用方法: 例」および「HAVING句」を参照してください
- 集計ファンクションのORDER BY句における式の照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

単一の引数を指定する多くの集計ファンクションには、次の句があります。

- DISTINCT句およびUNIQUE句(これらは同義)を指定すると、集計ファンクションは引数式の重複行を排除した値のみを考慮します。簡単に説明するために、この章の集計ファンクションの構文図ではキーワードDISTINCTを使用しています。
- ALL句を指定すると、集計ファンクションは重複行を含むすべての値を考慮します。

たとえば、1、1、1、3の平均値はDISTINCTでは2となります。ALLでの平均値は1.5となります。どの句も指定しない場合、デフォルトでALLが使用されます。

集計ファンクションには、分析ファンクションの構文の一部であるwindowing_clauseを使用できるものがあります。この句の詳細は、「windowing_clause」を参照してください。

COUNT(*)、GROUPING、およびGROUPING_IDを除くすべての集計ファンクションは、NULLを無視します。集計ファンクションに対する引数にNVLファンクションを使用して、NULLをある値で置き換えることができます。COUNTおよびREGR_COUNTはNULLを戻しません。数字または0(ゼロ)を戻します。その他の集計ファンクションでは、データ・セットに行がない場合、または集計ファンクションに対する引数としてNULLを持つ行のみがある場合はNULLを戻します。

集計ファンクションMIN、MAX、SUM、AVG、COUNT、VARIANCEおよびSTDDEVの後に、KEEPキーワードを指定すると、FIRSTまたはLASTファンクションと組み合せて使用することができ、与えられたソート指定に対して、FIRSTまたはLASTとしてランク付けされた一連の行の一連の値を操作できます。詳細は、「FIRST」を参照してください。

集計ファンクションはネストできます。たとえば、次の例では、サンプル・スキーマhrのすべての部門における最高額の給与の平均を計算します。

```
SELECT AVG(MAX(salary))
  FROM employees
  GROUP BY department_id;
AVG(MAX(SALARY))
----------------
      10926.3333
```

この計算では、GROUP BY句(department_id)で定義されているグループごとの内部集計(MAX(salary))を評価し、その結果をもう一度集計しています。

- APPROX_COUNT
- APPROX_COUNT_DISTINCT
- APPROX_COUNT_DISTINCT_AGG
- APPROX_COUNT_DISTINCT_DETAIL
- APPROX_MEDIAN
- APPROX_PERCENTILE
- APPROX_PERCENTILE_AGG
- APPROX_PERCENTILE_DETAIL
- APPROX_RANK
- APPROX_SUM
- AVG
- COLLECT
- CORR
- CORR_*
- COUNT
- COVAR_POP
- COVAR_SAMP
- CUME_DIST
- DENSE_RANK
- FIRST
- GROUP_ID
- GROUPING
- GROUPING_ID
- JSON_ARRAYAGG
- JSON_OBJECTAGG
- LAST
- LISTAGG
- MAX
- MEDIAN
- MIN
- PERCENT_RANK
- PERCENTILE_CONT
- PERCENTILE_DISC
- RANK
- REGR_(線形回帰)ファンクション
- STATS_BINOMIAL_TEST
- STATS_CROSSTAB
- STATS_F_TEST
- STATS_KS_TEST
- STATS_MODE
- STATS_MW_TEST

- STATS_ONE_WAY_ANOVA
- STATS_T_TEST_*
- STATS_WSR_TEST
- STDDEV
- STDDEV_POP
- STDDEV_SAMP
- SUM
- SYS_OP_ZONE_ID
- SYS_XMLAGG
- TO_APPROX_COUNT_DISTINCT
- TO_APPROX_PERCENTILE
- VAR_POP
- VAR_SAMP
- VARIANCE
- XMLAGG

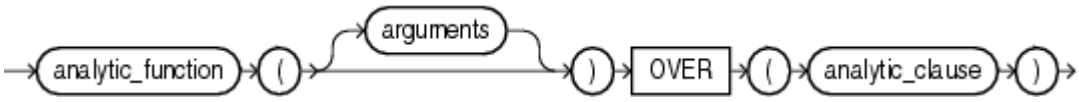
分析ファンクション

分析ファンクションは、行のグループに基づいて集計値を計算します。各グループに対して複数の行を戻す点で、集計ファンクションと異なります。行のグループをウィンドウといい、analytic_clauseで定義されます。各行に対して、行のスライディング・ウィンドウが定義されます。このウィンドウによって、カレント行の計算に使用される行の範囲が決定されます。ウィンドウの大きさは、行の物理数値または時間などのロジカル・インターバルに基づきます。

分析ファンクションは、問合せで最後に実行される演算(最後のORDER BY句を除く)の集合です。すべての結合およびすべてのWHERE、GROUP BYおよびHAVING句は、分析ファンクションが処理される前に実行されます。そのため、分析ファンクションは、SELECT構文のリストまたはORDER BY句のみに指定できます。

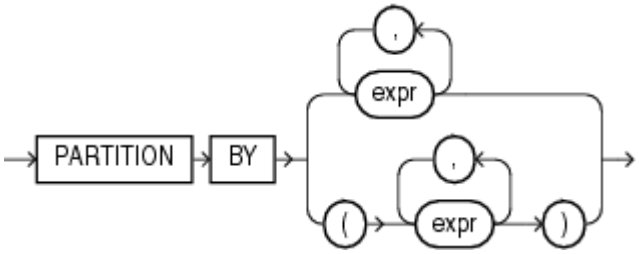
通常、分析ファンクションは、累積集計、移動集計、センター集計およびレポート集計の実行に使用されます。

analytic_function::=

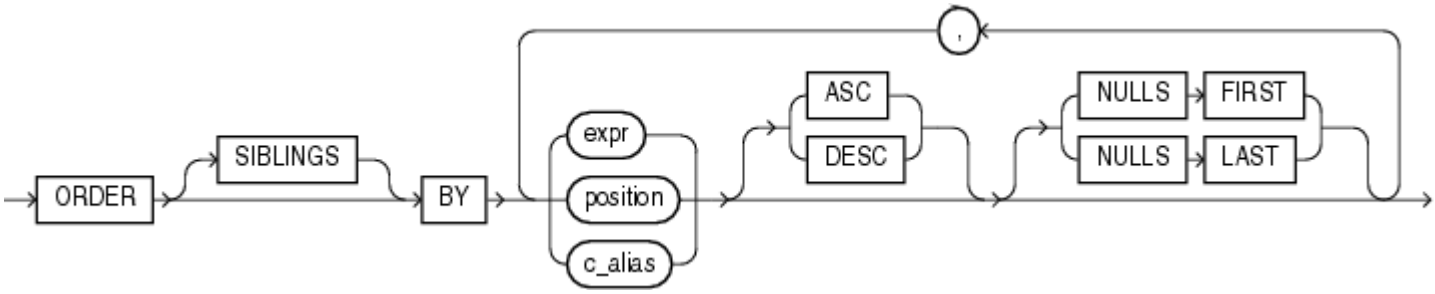


analytic_clause::=

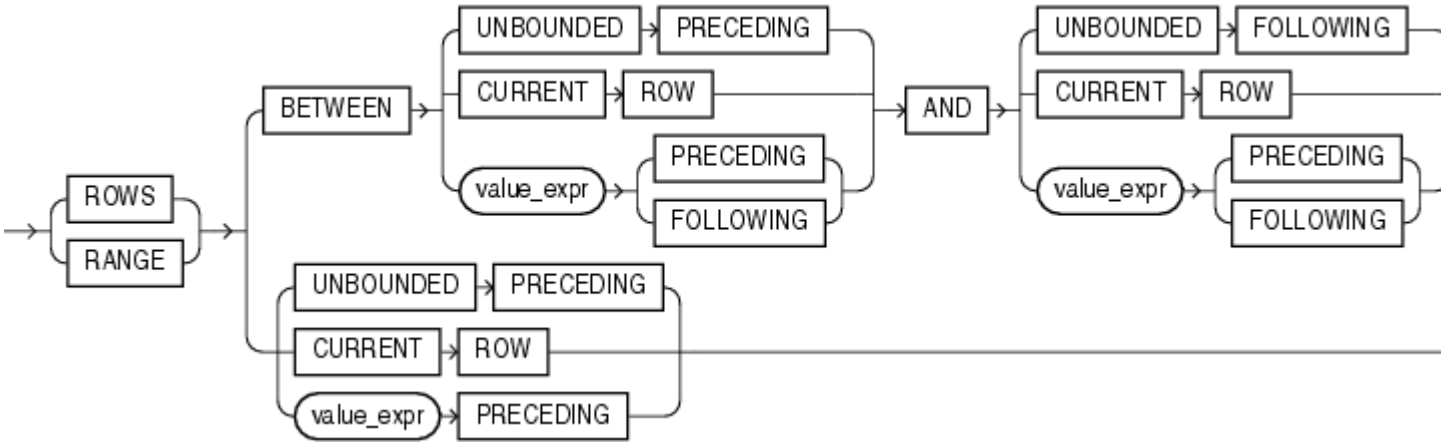
query_partition_clause::=



order_by_clause::=



windowing_clause::=



次に、この構文のセマンティクスを示します。

analytic_function

分析ファンクションの名前を指定します(セマンティクスの説明の後に示す分析ファンクションのリストを参照)。

arguments

分析ファンクションには引数を0から3個指定します。引数には、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を指定できます。Oracleは、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換します。個々のファンクションに特に指定がないかぎり、戻り型もその引数のデータ型となります。

関連項目:

数値の優先順位の詳細は、数値の優先順位を参照してください。暗黙的な変換の詳細は、表2-8を参照してください。

analytic_clause

OVER analytic_clause句は、ファンクションが問合せ結果セットを操作することを示します。この句は、FROM、WHERE、GROUP BYおよびHAVING句の後に計算されます。SELECT構文のリストのこの句またはORDER BY句に分析ファンクションを指定できます。分析ファンクションに基づいて、問合せの結果をフィルタするには、これらのファンクションを親問合せ内でネストした後、ネストされた副問合せの結果をフィルタします。

analytic_clauseのノート

analytic_clauseには、次のノートが適用されます。

- analytic_clauseのどの部分にも、分析ファンクションを指定して分析ファンクションをネストできません。ただし、副問合せで分析ファンクションを指定して、別の分析ファンクションを計算することはできます。
- OVER analytic_clauseには、組込み分析ファンクションと同様に、ユーザー定義の分析ファンクションを指定できます。「CREATE FUNCTION」を参照してください。
- analytic_clauseでのPARTITION BY句およびORDER BY句は照合依存です。

関連項目:

分析ファンクションのOVER (PARTITION BY ... ORDER BY ...)句の照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

query_partition_clause

PARTITION BY句を使用すると、1つ以上のvalue_exprに基づいて、問合せ結果セットをグループに分割できます。この句を省略すると、ファンクションは問合せ結果セットのすべての行を単一のグループとして扱います。

分析ファンクションでquery_partition_clauseを使用するには、構文の上位ブランチ(カッコなし)を使用します。この句をモデルの問合せ(model_column_clauses内)またはパーティション化された外部結合(outer_join_clause内)で使用するには、構文の下位ブランチ(カッコ付き)を使用します。

同じまたは異なるPARTITION BYキーで、同じ問合せに複数の分析ファンクションを指定できます。

問い合せているオブジェクトにパラレル属性があり、query_partition_clauseで分析ファンクションを指定する場合は、ファンクションの計算もパラレル化されます。

有効な値のvalue_exprは、定数、列、非分析ファンクション、ファンクション式、またはこれらのいずれかを含む式です。

order_by_clause

order_by_clauseを使用すると、パーティション内でのデータの順序付け方法を指定できます。すべての分析ファンクションに対して、各キーがvalue_exprで定義され、順番付け順序で修飾された複数キーのパーティション内での値を順番付けできます。

各ファンクションには、複数の順序式を指定できます。これは、2番目の式が最初の式にある同一値との間の関連性を変換できるため、値をランク付けするファンクションを使用する場合に特に有効です。

order_by_clauseの結果が複数行に対して同一値になる場合は常に、ファンクションは次のように動作します。

- CUME_DIST、DENSE_RANK、NTILE、PERCENT_RANKおよびRANKは、各行に対して同じ結果を戻します。
- ROW_NUMBERは、order_by_clauseに基づいた同順位がある場合でも、各行に異なる値を割り当てます。この値は行が処理される順序に基づくため、ORDER BYによって全体的な順序が保証されていない場合には非決定的になることがあります。
- 他のすべての分析ファンクションでは、ウィンドウの指定によって結果が異なります。RANGEキーワードを使用して論理ウィンドウを指定する場合、ファンクションは各行に同じ結果を戻します。ROWSキーワードを使用して物理ウィンドウを指定する場合、結果は非決定的になります。

ORDER BY句の制限事項

ORDER BY句には、次の制限事項が適用されます。

- order_by_clauseを分析ファンクションで使用する場合、式(expr)が必要です。SIBLINGSキーワードは無効です(これは、階層問合せでのみ有効です)。位置(position)および列別名(c_alias)も無効です。それ以外で使用する場合、このorder_by_clauseは、問合せまたは副問合せ全体を順序付ける場合に使用するものと同じです。
- RANGEキーワードを使用する分析ファンクションでは、次のいずれかのウィンドウを指定する場合に、ORDER BY句で複数のソート・キーを使用できます。
    - RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW。この短縮形は、RANGE UNBOUNDED PRECEDINGです。
    - RANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING。
    - RANGE BETWEEN CURRENT ROW AND CURRENT ROW。

- RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING。

この4つ以外のウィンドウ境界では、分析ファンクションのORDER BY句でソート・キーを1つしか持てません。この制限事項は、ROWキーワードで指定したウィンドウ境界には適用されません。

ASC | DESC

順番付け順序(昇順または降順)を指定します。デフォルトはASCです。

NULLS FIRST | NULLS LAST

NULL値を含む戻された行が、順番付け順序の最初にくるか、最後にくるかを指定します。

NULLS LASTは昇順のデフォルトで、NULLS FIRSTは降順のデフォルトです。

分析ファンクションは、常に、ファンクションのorder_by_clauseで指定された順序で行を操作します。ただし、ファンクションのorder_by_clauseは結果の順序を保証しません。最終結果の順序を保証するには、問合せのorder_by_clauseを使用してください。

関連項目:

この句の詳細は、「SELECT」の「order_by_clause」を参照してください。

windowing_clause

一部の分析ファンクションでは、windowing_clauseを使用できます。7-16ページに示す分析ファンクションのリストでは、windowing_clauseを使用できるファンクションにアスタリスク(*)が付いています。

ROWS | RANGE

これらのキーワードは、各行に対して、ファンクションの結果の計算に使用されるウィンドウ(行の物理集合または論理集合)を定義します。ファンクションは、ウィンドウのすべての行に適用されます。ウィンドウは、問合せ結果セット内またはパーティションの上から下まで移動します。

- ROWSは、物理単位(行)でウィンドウを指定します。
- RANGEは、論理オフセットとしてウィンドウを指定します。

order_by_clauseを指定しないと、この句を指定できません。RANGE句で定義したウィンドウ境界には、order_by_clauseで指定できる式が1つのみのものもあります。ORDER BY句の制限事項を参照してください。

分析ファンクションが論理オフセットで戻す値は、常に決定的なものです。ただし、分析ファンクションが物理オフセットで戻す値は、順序式の結果が一意の順序にならないかぎり、非決定的な結果を生成することがあります。order_by_clauseに複数の列を指定して、結果の順序を一意にする必要があります。

BETWEEN ... AND

BETWEEN ... AND句を使用すると、ウィンドウにスタート・ポイントおよびエンド・ポイントを指定できます。最初の式(ANDの前)はスタート・ポイントを定義し、2番目の式(ANDの後)はエンド・ポイントを定義します。

BETWEENを省略してエンド・ポイントを1つのみ指定すると、Oracleはそれをスタート・ポイントとみなし、デフォルトでカレント行をエンド・ポイントに指定します。

UNBOUNDED PRECEDING

UNBOUNDED PRECEDINGを指定すると、パーティションの最初の行で、ウィンドウが開始します。これはスタート・ポイントの指

定で、エンド・ポイントの指定としては使用できません。

UNBOUNDED FOLLOWING

UNBOUNDED FOLLOWINGを指定すると、パーティションの最後の行で、ウィンドウが終了します。これはエンド・ポイントの指定で、スタート・ポイントの指定としては使用できません。

CURRENT ROW

スタート・ポイントとして、CURRENT ROWでウィンドウがカレント行または値(それぞれROWまたはRANGEを指定したかどうかに基づく)で開始することを指定します。この場合、value_expr PRECEDINGをエンド・ポイントにできません。

エンド・ポイントとして、CURRENT ROWでウィンドウがカレント行または値(それぞれROWまたはRANGEを指定したかどうかに基づく)で終了することを指定します。この場合、value_expr FOLLOWINGをスタート・ポイントにできません。

value_expr PRECEDINGまたはvalue_expr FOLLOWING

RANGEまたはROWの場合は次のようになります。

- value_expr FOLLOWINGがスタート・ポイントの場合、エンド・ポイントはvalue_expr FOLLOWINGである必要があります。
- value_expr PRECEDINGがエンド・ポイントの場合、スタート・ポイントはvalue_expr PRECEDINGである必要があります。

数値形式の時間間隔で定義されている論理ウィンドウを定義する場合、変換ファンクションを使用する必要があります。

関連項目:

数値時間から間隔への変換の詳細は、「NUMTOYMINTERVAL」および「NUMTODSINTERVAL」を参照してください。

ROWSを指定した場合、次のことがいえます。

- value_exprは物理オフセットになります。これは定数または式であり、正数値に評価する必要があります。
- value_exprがスタート・ポイントの一部の場合、エンド・ポイントの前にある行に評価する必要があります。

RANGEを指定した場合、次のことがいえます。

- value_exprは論理オフセットになります。これは、正数値または期間リテラルに評価する定数または式である必要があります。期間リテラルの詳細は、リテラルを参照してください。
- order_by_clauseには、式を1つのみ指定できます。
- value_exprが数値に対して評価を行う場合、ORDER BY exprは数値データ型またはDATEデータ型である必要があります。
- value_exprが間隔値に対して評価を行う場合、ORDER BY exprはDATEデータ型である必要があります。

windowing_clauseを完全に省略した場合、デフォルトでRANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROWになります。

分析ファンクションは、通常、データ・ウェアハウス環境で使用されます。次に示す分析ファンクションのリストでは、windowing_clauseを含む完全な構文を使用できるファンクションには、アスタリスク(*)が付いています。

- AVG *
- CLUSTER_DETAILS

- CLUSTER_DISTANCE
- CLUSTER_ID
- CLUSTER_PROBABILITY
- CLUSTER_SET
- CORR *
- COUNT *
- COVAR_POP *
- COVAR_SAMP *
- CUME_DIST
- DENSE_RANK
- FEATURE_DETAILS
- FEATURE_ID
- FEATURE_SET
- FEATURE_VALUE
- FIRST
- FIRST_VALUE *
- LAG
- LAST
- LAST_VALUE *
- LEAD
- LISTAGG
- MAX *
- MIN *
- NTH_VALUE *
- NTILE
- PERCENT_RANK
- PERCENTILE_CONT
- PERCENTILE_DISC
- PREDICTION
- PREDICTION_COST
- PREDICTION_DETAILS
- PREDICTION_PROBABILITY
- PREDICTION_SET
- RANK
- RATIO_TO_REPORT
- REGR_(線形回帰)ファンクション *
- ROW_NUMBER
- STDDEV *
- STDDEV_POP *
- STDDEV_SAMP *
- SUM *
- VAR_POP *
- VAR_SAMP *

- VARIANCE *

関連項目:

これらのファンクションおよびその使用方法の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

オブジェクト参照ファンクション

オブジェクト参照ファンクションは、指定されたオブジェクト型のオブジェクトへの参照となるREF値を操作します。オブジェクト参照ファンクションを次に示します。

- DEREF
- MAKE_REF
- REF
- REFTOHEX
- VALUE

関連項目:

REFデータ型の詳細は、『Oracle Databaseオブジェクト・リレーショナル開発者ガイド』を参照してください。

モデル・ファンクション

モデル・ファンクションは、SELECT文のmodel_clauseでのみ使用できます。新しいモデル・ファンクションを次に示します。

- CV
- ITERATION_NUMBER
- PRESENTNNV
- PRESENTV
- PREVIOUS

OLAPファンクション

OLAPファンクションは、ディメンション・オブジェクトからのデータを2次元のリレーショナル形式で戻します。OLAPファンクションを次に示します。

- CUBE_TABLE

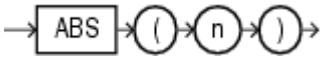
データ・カートリッジ・ファンクション

データ・カートリッジ・ファンクションは、データ・カートリッジ開発者にとって有用なものです。データ・カートリッジ・ファンクションを次に示します。

- DATAOBJ_TO_MAT_PARTITION
- DATAOBJ_TO_PARTITION

ABS

構文



目的

ABSは、nの絶対値を戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を返します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

例

次の例では、-15の絶対値を戻します。

```
SELECT ABS(-15) "Absolute"
  FROM DUAL;
  Absolute
----------
        15
```

ACOS

構文

目的

ACOSは、nのアーク・コサインを戻します。引数nは-1から1の範囲で、ファンクションによって戻される値は0からπ(ラジアン)の範囲です。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数がBINARY_FLOATの場合、このファンクションはBINARY_DOUBLEを戻します。それ以外の場合、引数と同じ数値データ型を戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

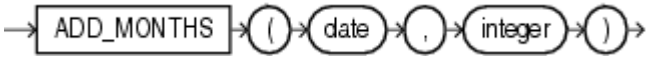
例

次の例では、.3のアーク・コサインを戻します。

```
SELECT ACOS(.3)"Arc_Cosine"
  FROM DUAL;
Arc_Cosine
----------
1.26610367
```

ADD_MONTHS

構文



目的

ADD_MONTHSは、日付dateに月数integerを加えて戻します。月は、セッション・パラメータNLS_CALENDARによって定義されます。引数dateには、日時値、または暗黙的にDATEに変換可能な任意の値を指定できます。引数integerには、整数、または暗黙的に整数に変換可能な任意の値を指定できます。戻り型は、dateのデータ型に関係なく常にDATEです。dateが月の最終日の場合、または結果の月の日数がdateの日付コンポーネントよりも少ない場合、戻される値は結果の月の最終日となります。それ以外の場合、結果にはdateと同じ日付コンポーネントが含まれます。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

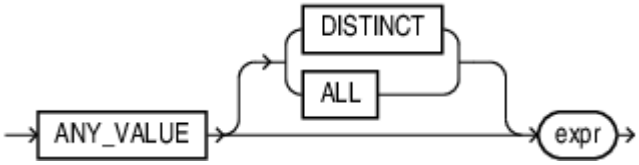
例

次の例では、サンプル表employeesのhire_date後の月を戻します。

```
SELECT TO_CHAR(ADD_MONTHS(hire_date, 1), 'DD-MON-YYYY') "Next month"
  FROM employees
  WHERE last_name = 'Baer';
Next Month
-----------
07-JUL-2002
```

ANY_VALUE

構文



目的

ANY_VALUEは、exprの非決定値を1つ返します。これは、集計ファンクションとして使用できます。

ANY_VALUEを使用すると、GROUP BY句を持つ問合せを最適化できます。ANY_VALUEは、グループ内の式の値を返します。これは、最初の値を返すように最適化されています。

これにより、入力行に対して比較が行われないようにし、GROUP BY句の一部としてすべての列を指定する必要もなくなります。値を比較しないため、ANY_VALUEは、GROUP BY問合せのMINまたはMAXよりも迅速に値を返します。

セマンティクス

ALL、DISTINCT: これらのキーワードは、ANY_VALUEによってサポートされますが、問合せ結果に影響はありません。

expr: 式には、列、定数、バインド変数またはこれらを含む式を指定できます。

式のNULL値は無視されます。

LONG、LOB、FILEまたはCOLLECTIONを除くすべてのデータ型をサポートします。

LONGを使用すると、ORA-00997が発生します。

LOB、FILEまたはCOLLECTIONデータ型を使用すると、ORA-00932が発生します。

ANY_VALUEは、MINおよびMAXと同じルールに従います。

GROUP BYの指定に基づいて、各グループ内のすべての値を返します。グループ内のすべての行にNULL式の値がある場合はNULLを返します。

ANY_VALUEの結果は決定的ではありません。

制限事項

XMLTypeおよびANYDATAはサポートされていません。

例7-1 集計ファンクションとしてのANY_VALUEの使用

この例では、ANY_VALUEをSHスキーマのGROUP BY問合せ内で集計ファンクションとして使用します。

```
SELECT c.cust_id, ANY_VALUE(cust_last_name), SUM(amount_sold)
  FROM customers c, sales s
  WHERE s.cust_id = c.cust_id
  GROUP BY c.cust_id;
```

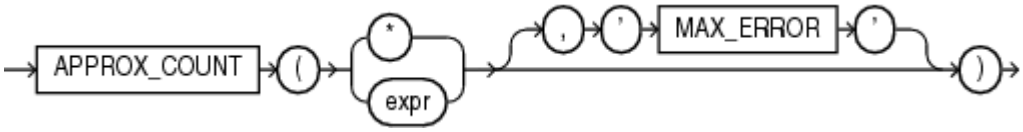
次の問合せ結果では、最初の11行のみが表示されています。

```
CUST_ID  ANY_VALUE(CUST_LAST_NAME) SUM(AMOUNT_SOLD)
------- ------------------------- ----------------
   6950 Sandburg                               78
  17920 Oliver                               3201
  66800 Case                                 2024
  37280 Edwards                              2256
```

```
 109850 Lindegreen                                757
   3910 Oddell                                    185
  84700 Marker                                  164.4
  26380 Remler                                   118
  11600 Oppy                                     158
  23030 Rothrock                                 533
  42780 Zanis                                    182
...
630 rows selected.
```

APPROX_COUNT

構文



目的

APPROX_COUNTは、式の近似カウントを戻します。MAX_ERRORを2番目の引数として指定すると、このファンクションは、実際のカウントと近似カウントの間の最大エラーを戻します。

このファンクションは、HAVING句で対応するAPPROX_RANKファンクションとともに使用する必要があります。問合せでAPPROX_COUNT、APPROX_SUMまたはAPPROX_RANKを使用する場合、その問合せでは他の集計ファンクションを使用しないでください。

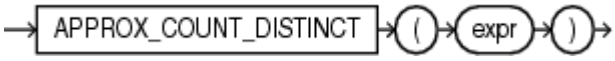
例

次の問合せでは、各部門内の10個の最も一般的なジョブが戻されます。

```
SELECT department_id, job_id,
       APPROX_COUNT(*)
FROM   employees
GROUP BY department_id, job_id
HAVING
  APPROX_RANK (
  PARTITION BY department_id
  ORDER BY APPROX_COUNT(*)
  DESC ) <= 10;
```

APPROX_COUNT_DISTINCT

構文



目的

APPROX_COUNT_DISTINCTは、exprの個別値を含む行の概数を戻します。

このファンクションは、exprの異なる値を含む正確な数の行を戻すCOUNT (DISTINCT expr)ファンクションの代替機能を提供します。APPROX_COUNT_DISTINCTは、正確な結果とわずかに誤差がありますが、COUNTよりはるかに高速に大量のデータを処理します。

exprでは、BFILE、BLOB、CLOB、LONG、LONG RAWまたはNCLOB以外のスカラー・データ型の列を指定できます。

APPROX_COUNT_DISTINCTは、exprのnull値を含む行を無視します。このファンクションは、NUMBERを戻します。

関連項目:

- COUNT (DISTINCT expr)ファンクションの詳細は、「COUNT」を参照してください。
- exprの文字値を比較するためにAPPROX_COUNT_DISTINCTで使用する照合を定義する照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の文は、manager_idの異なる値を使用した適切な数の行を戻します。

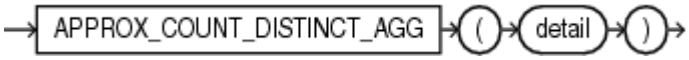
```
SELECT APPROX_COUNT_DISTINCT(manager_id) AS "Active Managers"
  FROM employees;
Active Managers
---------------
             18
```

次の文は、各製品の適切な数の異なる顧客を戻します。

```
SELECT prod_id, APPROX_COUNT_DISTINCT(cust_id) AS "Number of Customers"
  FROM sales
  GROUP BY prod_id
  ORDER BY prod_id;
   PROD_ID Number of Customers
---------- -------------------
        13                2516
        14                2030
        15                2105
        16                2367
        17                2093
        18                2975
        19                2630
        20                3791
. . .
```

APPROX_COUNT_DISTINCT_AGG

構文



目的

APPROX_COUNT_DISTINCT_AGGでは、その入力として、個別値の近似カウントに関する情報を含む詳細の列を取得し、これらのカウントの集計を実行できます。

detailには、APPROX_COUNT_DISTINCT_DETAILファンクションまたはAPPROX_COUNT_DISTINCT_AGGファンクションによって作成された詳細の列を指定します。この列のデータ型はBLOBです。

このファンクションをSELECT文でGROUP BY句とともに指定すると、行の各グループ内の詳細に含まれる情報を集計し、グループごとに単一の詳細を戻すことができます。

このファンクションは詳細と呼ばれるBLOB値を戻し、ここにはカウント集計に関する情報が特殊な形式で含まれます。このファンクションによって戻された詳細を表またはマテリアライズド・ビューに格納した後、再度APPROX_COUNT_DISTINCT_AGGファンクションを使用してこれらの詳細をさらに集計したり、TO_APPROX_COUNT_DISTINCTファンクションを使用して詳細値を判読可能なNUMBER値に変換できます。

関連項目:

- APPROX_COUNT_DISTINCT_DETAIL
- TO_APPROX_COUNT_DISTINCT

例

APPROX_COUNT_DISTINCT_DETAILおよびTO_APPROX_COUNT_DISTINCTファンクションと組み合せてAPPROX_COUNT_DISTINCT_AGGファンクションを使用する例は、「APPROX_COUNT_DISTINCT_AGG: 例」を参照してください。

APPROX_COUNT_DISTINCT_DETAIL

構文

目的

APPROX_COUNT_DISTINCT_DETAILは、exprに対して個別値を含む行の概数に関する情報を計算して、詳細と呼ばれるBLOB値を戻し、ここにはその情報が特殊な形式で含まれます。

exprでは、BFILE、BLOB、CLOB、LONG、LONG RAWまたはNCLOB以外のスカラー・データ型の列を指定できます。このファンクションでは、exprの値がNULLの行は無視されます。

通常、このファンクションはGROUP BY句とともにSELECT文で使用されます。このように使用すると、行の各グループ内のexprに対する個別値の近似カウント情報が計算され、グループごとに単一の詳細が戻されます。

APPROX_COUNT_DISTINCT_DETAILによって戻される詳細は、APPROX_COUNT_DISTINCT_AGGファンクションへの入力として使用して詳細の集計を実行したり、TO_APPROX_COUNT_DISTINCTファンクションへの入力として使用して詳細を判読可能な異なるカウント値に変換できます。これらの3つのファンクションを一緒に使用すると、リソースを大量に使用する近似カウント計算を1回実行し、生成された詳細を格納した後、これらの詳細に対して効率的に集計と問合せを実行できます。たとえば:

1. APPROX_COUNT_DISTINCT_DETAILファンクションを使用して、個別値の近似カウント情報を計算し、生成される詳細を表またはマテリアライズド・ビューに格納します。これらは、市区町村の人口統計カウントや日次販売カウントなど、粒度の高い詳細にできます。
2. APPROX_COUNT_DISTINCT_AGGファンクションを使用して、前のステップで取得した詳細を集計し、生成される詳細を表またはマテリアライズド・ビューに格納します。これらは、都道府県の人口統計カウントや月次販売カウントなど、粒度のより低い詳細にできます。
3. TO_APPROX_COUNT_DISTINCTファンクションを使用して、格納した詳細値を判読可能なNUMBER値に変換します。TO_APPROX_COUNT_DISTINCTファンクションを使用すると、APPROX_COUNT_DISTINCT_DETAILファンクションまたはAPPROX_COUNT_DISTNCT_AGGファンクションによって作成された詳細値を問い合せることができます。

関連項目:

- APPROX_COUNT_DISTINCT_AGG
- TO_APPROX_COUNT_DISTINCT

例

この項の例では、APPROX_COUNT_DISTINCT_DETAIL、APPROX_COUNT_DISTINCT_AGGおよびTO_APPROX_COUNT_DISTINCTファンクションを一緒に使用して、リソースを大量に使用する近似カウント計算を1回実行し、生成された詳細を格納した後、これらの詳細に対して効率的に集計と問合せを実行する方法を示します。

APPROX_COUNT_DISTINCT_DETAIL: 例

次の文は、表sh.timesおよびsh.salesに対して、各日に販売された異なる製品の概数を問い合せます。

APPROX_COUNT_DISTINCT_DETAILファンクションは、該当する製品が販売された各日について、daily_detailという詳細な情報を戻します。戻された詳細は、daily_prod_count_mvというマテリアライズド・ビューに格納されます。

```
CREATE MATERIALIZED VIEW daily_prod_count_mv AS
  SELECT t.calendar_year year,
         t.calendar_month_number month,
         t.day_number_in_month day,
         APPROX_COUNT_DISTINCT_DETAIL(s.prod_id) daily_detail
  FROM times t, sales s
  WHERE t.time_id = s.time_id
  GROUP BY t.calendar_year, t.calendar_month_number, t.day_number_in_month;
```

APPROX_COUNT_DISTINCT_AGG: 例

次の文は、APPROX_COUNT_DISTINCT_AGGファンクションを使用して、daily_prod_count_mvに格納されている日次詳細を読み取り、各月に販売された異なる製品の概数を含む集計された詳細を作成します。これらの集計された詳細は、monthly_prod_count_mvというマテリアライズド・ビューに格納されます。

```
CREATE MATERIALIZED VIEW monthly_prod_count_mv AS
  SELECT year,
         month,
         APPROX_COUNT_DISTINCT_AGG(daily_detail) monthly_detail
  FROM daily_prod_count_mv
  GROUP BY year, month;
```

次の文は、各年に販売された異なる製品の概数を含む、集計された詳細を作成することを除き、前の文と似ています。これらの集計された詳細は、annual_prod_count_mvというマテリアライズド・ビューに格納されます。

```
CREATE MATERIALIZED VIEW annual_prod_count_mv AS
  SELECT year,
         APPROX_COUNT_DISTINCT_AGG(daily_detail) annual_detail
  FROM daily_prod_count_mv
  GROUP BY year;
```

TO_APPROX_COUNT_DISTINCT: 例

次の文は、TO_APPROX_COUNT_DISTINCTファンクションを使用して、daily_prod_count_mvに格納されている日次詳細情報を問い合せて、各日に販売された異なる製品の概数を戻します。

```
SELECT year,
       month,
       day,
       TO_APPROX_COUNT_DISTINCT(daily_detail) "NUM PRODUCTS"
  FROM daily_prod_count_mv
  ORDER BY year, month, day;
      YEAR      MONTH        DAY NUM PRODUCTS
---------- ---------- ---------- ------------
      1998          1          1           24
      1998          1          2           25
      1998          1          3           11
      1998          1          4           34
      1998          1          5           10
      1998          1          6            8
      1998          1          7           37
      1998          1          8           26
      1998          1          9           25
      1998          1         10           38
. . .
```

次の文は、TO_APPROX_COUNT_DISTINCTファンクションを使用して、monthly_prod_count_mvに格納されている月次詳細情報を問い合せて、各月に販売された異なる製品の概数を戻します。

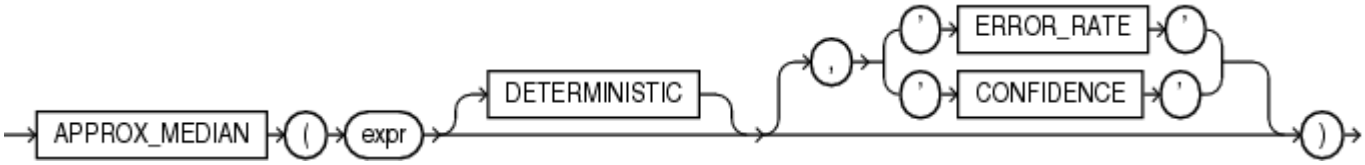
```
SELECT year,
       month,
       TO_APPROX_COUNT_DISTINCT(monthly_detail) "NUM PRODUCTS"
  FROM monthly_prod_count_mv
  ORDER BY year, month;
      YEAR      MONTH NUM PRODUCTS
---------- ---------- ------------
      1998          1           57
      1998          2           56
      1998          3           55
      1998          4           49
      1998          5           49
      1998          6           48
      1998          7           54
      1998          8           56
      1998          9           55
      1998         10           57
. . .
```

次の文は、TO_APPROX_COUNT_DISTINCTファンクションを使用して、annual_prod_count_mvに格納されている年次詳細情報を問い合せて、各年に販売された異なる製品の概数を戻します。

```
SELECT year,
       TO_APPROX_COUNT_DISTINCT(annual_detail) "NUM PRODUCTS"
  FROM annual_prod_count_mv
  ORDER BY year;
      YEAR NUM PRODUCTS
---------- ------------
      1998           60
      1999           72
      2000           72
      2001           71
```

APPROX_MEDIAN

構文



目的

APPROX_MEDIANは、連続分散モデルを想定する近似逆分散ファンクションです。数値または日時値を取得して、おおよその中央値、または値のソート後に中央値となる、おおよその補間された値を戻します。計算では、NULLは無視されます。

このファンクションは、MEDIANファンクションの代替機能として、正確な中央値または補間された値を戻します。APPROX_MEDIANは、正確な結果とわずかに誤差がありますが、MEDIANよりはるかに高速に大量のデータを処理します。

exprには、おおよその中央値を計算する式を指定します。exprで許容されるデータ型と、このファンクションの戻り値データ型は、DETERMINISTIC句で指定するアルゴリズムによって異なります。

DETERMINISTIC

この句を使用すると、このファンクションで使用するアルゴリズムのタイプを指定して、おおよその中央値を計算できます。

- DETERMINISTICを指定すると、このファンクションによって、決定的なおおよその中央値が計算されます。この場合、exprは数値または数値に暗黙的に変換可能な値に評価される必要があります。また、その引数の数値データ型と同じデータ型を戻します。
- DETERMINSTICを指定しないと、このファンクションによって、非決定的なおおよその中央値が計算されます。この場合、exprは、数値または日時値、あるいは数値または日時値に暗黙的に変換可能な値に評価される必要があります。また、その引数の数値または日時データ型と同じデータ型を戻します。

ERROR_RATE | CONFIDENCE

これらの句を使用すると、このファンクションによって計算される値の精度を決定できます。これらの句のいずれかを指定すると、exprのおおよその中央値が戻されるかわりに、次のいずれかの値を表す0から1までの10進値(それらの値を含む)が戻されます。

- ERROR_RATEを指定すると、戻り値は、exprのおおよその中央値の計算におけるエラー率を表します。
- CONFIDENCEを指定すると、戻り値は、ERROR_RATEを指定したときに戻されるエラー率の信頼水準を表します。

関連項目:

- MEDIAN
- 「APPROX_PERCENTILE」を参照してください。これは、指定されたパーセンタイルに対して、そのパーセンタイルに一致するように補間されたおおよその値を戻します。APPROX_MEDIANは、パーセンタイル値が0.5に指定される特別なAPPROX_PERCENTILEです。

例

次の問合せは、hr.employees表内の部門ごとに、決定的な給与のおおよその中央値を戻します。

```
SELECT department_id "Department",
       APPROX_MEDIAN(salary DETERMINISTIC) "Median Salary"
  FROM employees
  GROUP BY department_id
  ORDER BY department_id;

Department Median Salary
---------- -------------
        10          4400
        20          6000
        30          2765
        40          6500
        50          3100
        60          4800
        70         10000
        80          9003
        90         17000
       100          7739
       110          8300
                    7000
```

次の問合せは、前の問合せによって戻された給与のおおよその中央値におけるエラー率を戻します。

```
SELECT department_id "Department",
       APPROX_MEDIAN(salary DETERMINISTIC, 'ERROR_RATE') "Error Rate"
  FROM employees
  GROUP BY department_id
  ORDER BY department_id;
Department Error Rate
---------- ----------
        10 .002718282
        20 .021746255
        30 .021746255
        40 .002718282
        50 .019027973
        60 .019027973
        70 .002718282
        80 .021746255
        90 .021746255
       100 .019027973
       110 .019027973
           .002718282
```

次の問合せは、前の問合せによって戻されたエラー率について信頼水準を戻します。

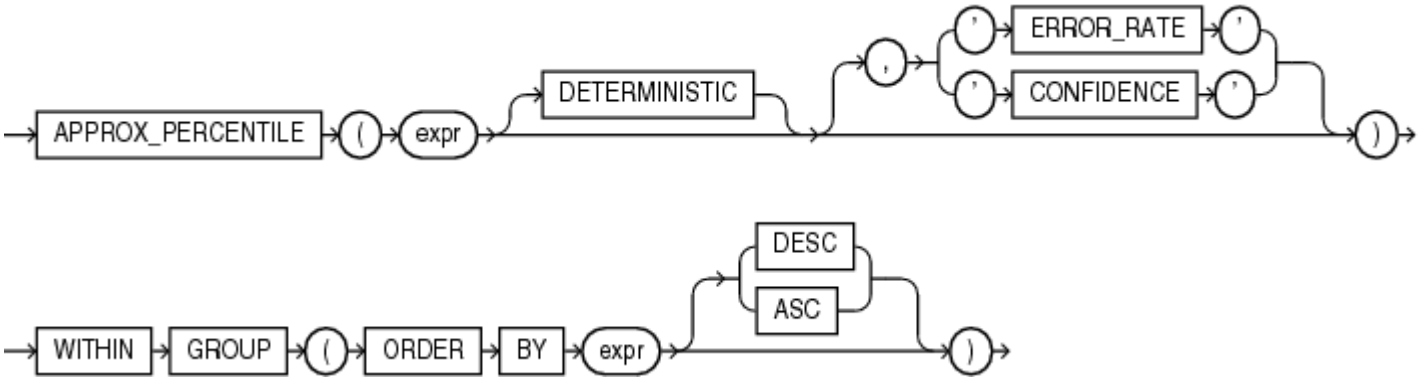
```
SELECT department_id "Department",
       APPROX_MEDIAN(salary DETERMINISTIC, 'CONFIDENCE') "Confidence Level"
  FROM employees
  GROUP BY department_id
  ORDER BY department_id;
Department Confidence Level
---------- ----------------
        10       .997281718
        20       .999660215
        30       .999660215
        40       .997281718
        50       .999611674
        60       .999611674
        70       .997281718
        80       .999660215
        90       .999660215
       100       .999611674
       110       .999611674
                 .997281718
```

次の問合せは、hr.employees表内の部門ごとに、非決定的な雇用開始日のおおよその中央値を戻します。

```
SELECT department_id "Department",
       APPROX_MEDIAN(hire_date) "Median Hire Date"
  FROM employees
  GROUP BY department_id
  ORDER BY department_id;
Department Median Hire Date
---------- ----------------
        10        17-SEP-03
        20        17-FEB-04
        30        24-JUL-05
        40        07-JUN-02
        50        15-MAR-06
        60        05-FEB-06
        70        07-JUN-02
        80        23-MAR-06
        90        17-JUN-03
       100        28-SEP-05
       110        07-JUN-02
                  24-MAY-07
```

APPROX_PERCENTILE

構文



目的

APPROX_PERCENTILEは、近似逆分散ファンクションです。これは、パーセンタイル値およびソート指定を取得し、そのソート指定に従ってそのパーセンタイル値に該当する値を戻します。計算では、NULLは無視されます

このファンクションは、PERCENTILE_CONTおよびPERCENTILE_DISCファンクションの代替機能として、正確な結果を戻します。APPROX_PERCENTILEは、正確な結果とわずかに誤差がありますが、PERCENTILE_CONTや PERCENTILE_DISCよりはるかに高速に大量のデータを処理します。

最初のexprはパーセンタイル値であり、0から1の数値に評価される必要があります。

ORDER BYに含まれる2番目のexprは単一式であり、このファンクションではこの式を使用して結果を計算します。exprで許容されるデータ型と、このファンクションの戻り値データ型は、DETERMINISTIC句で指定するアルゴリズムによって異なります。

DETERMINISTIC

この句を使用すると、このファンクションで使用するアルゴリズムのタイプを指定して、戻り値を計算できます。

- DETERMINISTICを指定すると、このファンクションによって、決定的な結果が計算されます。この場合、ORDER BY句式は、数値または-2,147,483,648から2,147,483,647の範囲の数値に暗黙的に変換可能な値に評価される必要があります。このファンクションは、数値入力を最も近い整数に丸めます。また、ORDER BY句式の数値データ型と同じデータ型を戻します。戻り値は、必ずしもexprの値のいずれかであるとはかぎりません。
- DETERMINSTICを指定しないと、このファンクションによって、非決定的な結果が計算されます。この場合、ORDER BY句式は、数値または日時値、あるいは数値または日時値に暗黙的に変換可能な値に評価される必要があります。また、ORDER BY句式の数値または日時データ型と同じデータ型を戻します。戻り値は、exprの値のいずれかとなります。

ERROR_RATE | CONFIDENCE

これらの句を使用すると、このファンクションによって計算される結果の精度を決定できます。これらの句のいずれかを指定すると、exprに指定したパーセンタイル値に該当する値が戻されるかわりに、次のいずれかの値を表す0から1までの10進値(それらの値を含む)が戻されます。

- ERROR_RATEを指定すると、戻り値は、exprに指定されたパーセンタイル値に該当する値の計算におけるエラー率を表します。
- CONFIDENCEを指定すると、戻り値は、ERROR_RATEを指定したときに戻されるエラー率の信頼水準を表します。

DESC | ASC

指定されたパーセンタイル値に該当する値を計算するためのソート指定を設定します。DESCを指定するとORDER BY句式の値を降順でソートでき、ASCを指定すると値を昇順でソートできます。デフォルトはASCです。

関連項目:

- 「PERCENTILE_CONT」および「PERCENTILE_DISC」を参照してください。
- 「APPROX_MEDIAN」を参照してください。これは、パーセンタイル値が0.5に指定される特別なAPPROX_PERCENTILEです。

例

次の問合せは、hr.employees表内の部門ごとに、決定的な近似の25パーセンタイル、50パーセンタイルおよび75パーセンタイルの給与を戻します。補間計算では、給与は昇順でソートされます。

```
SELECT department_id "Department",
       APPROX_PERCENTILE(0.25 DETERMINISTIC)
         WITHIN GROUP (ORDER BY salary ASC) "25th Percentile Salary",
       APPROX_PERCENTILE(0.50 DETERMINISTIC)
         WITHIN GROUP (ORDER BY salary ASC) "50th Percentile Salary",
       APPROX_PERCENTILE(0.75 DETERMINISTIC)
         WITHIN GROUP (ORDER BY salary ASC) "75th Percentile Salary"
  FROM employees
  GROUP BY department_id
  ORDER BY department_id;
Department 25th Percentile Salary 50th Percentile Salary 75th Percentile Salary
---------- ---------------------- ---------------------- ----------------------
        10                   4400                   4400                   4400
        20                   6000                   6000                  13000
        30                   2633                   2765                   3100
        40                   6500                   6500                   6500
        50                   2600                   3100                   3599
        60                   4800                   4800                   6000
        70                  10000                  10000                  10000
        80                   7400                   9003                  10291
        90                  17000                  17000                  24000
       100                   7698                   7739                   8976
       110                   8300                   8300                  12006
                             7000                   7000                   7000
```

次の問合せは、前の問合せで計算された近似の25パーセンタイル給与におけるエラー率を戻します。

```
SELECT department_id "Department",
       APPROX_PERCENTILE(0.25 DETERMINISTIC, 'ERROR_RATE')
         WITHIN GROUP (ORDER BY salary ASC) "Error Rate"
  FROM employees
  GROUP BY department_id
  ORDER BY department_id;
Department Error Rate
---------- ----------
        10 .002718282
        20 .021746255
        30 .021746255
        40 .002718282
        50 .019027973
        60 .019027973
        70 .002718282
        80 .021746255
        90 .021746255
       100 .019027973
```

```
       110 .019027973
           .002718282
```

次の問合せは、前の問合せで計算されたエラー率について信頼水準を戻します。

```
SELECT department_id "Department",
       APPROX_PERCENTILE(0.25 DETERMINISTIC, 'CONFIDENCE')
         WITHIN GROUP (ORDER BY salary ASC) "Confidence"
FROM employees
GROUP BY department_id
ORDER BY department_id;
Department Confidence
---------- ----------
        10 .997281718
        20 .999660215
        30 .999660215
        40 .997281718
        50 .999611674
        60 .999611674
        70 .997281718
        80 .999660215
        90 .999660215
       100 .999611674
       110 .999611674
           .997281718
```

次の問合せは、hr.employees表内の部門ごとに、非決定的な近似の25パーセンタイル、50パーセンタイルおよび75パーセンタイルの給与を戻します。補間計算では、給与は昇順でソートされます。

```
SELECT department_id "Department",
       APPROX_PERCENTILE(0.25)
         WITHIN GROUP (ORDER BY salary ASC) "25th Percentile Salary",
       APPROX_PERCENTILE(0.50)
         WITHIN GROUP (ORDER BY salary ASC) "50th Percentile Salary",
       APPROX_PERCENTILE(0.75)
         WITHIN GROUP (ORDER BY salary ASC) "75th Percentile Salary"
  FROM employees
  GROUP BY department_id
  ORDER BY department_id;
Department 25th Percentile Salary 50th Percentile Salary 75th Percentile Salary
---------- ---------------------- ---------------------- ----------------------
        10                   4400                   4400                   4400
        20                   6000                   6000                  13000
        30                   2600                   2800                   3100
        40                   6500                   6500                   6500
        50                   2600                   3100                   3600
        60                   4800                   4800                   6000
        70                  10000                  10000                  10000
        80                   7300                   8800                  10000
        90                  17000                  17000                  24000
       100                   7700                   7800                   9000
       110                   8300                   8300                  12008
                             7000                   7000                   7000
```

APPROX_PERCENTILE_AGG

構文

目的

APPROX_PERCENTILE_AGGでは、その入力として、近似パーセンタイル情報を含む詳細の列を取得し、該当する情報の集計を実行できます。

detailには、APPROX_PERCENT_DETAILファンクションまたはAPPROX_PERCENTILE_AGGファンクションによって作成された詳細の列を指定します。この列のデータ型はBLOBです。

このファンクションをSELECT文でGROUP BY句とともに指定すると、行の各グループ内の詳細に含まれる情報を集計し、グループごとに単一の詳細を戻すことができます。

このファンクションは詳細と呼ばれるBLOB値を戻し、ここには近似パーセンタイル情報が特殊な形式で含まれます。このファンクションによって戻された詳細を表またはマテリアライズド・ビューに格納した後、再度APPROX_PERCENTILE_AGGファンクションを使用してこれらの詳細をさらに集計したり、TO_APPROX_PERCENTILEファンクションを使用して詳細を指定されたパーセンタイル値に変換できます。

関連項目:

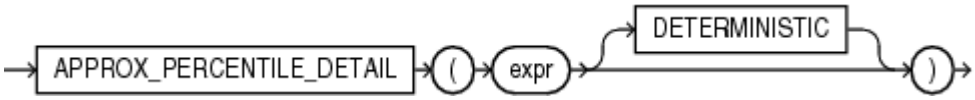
- APPROX_PERCENTILE_DETAIL
- TO_APPROX_PERCENTILE

例

APPROX_PERCENTILE_DETAILおよびTO_APPROX_PERCENTILEファンクションと組み合せてAPPROX_PERCENTILE_AGGファンクションを使用する例は、「APPROX_PERCENTILE_AGG: 例」を参照してください。

APPROX_PERCENTILE_DETAIL

構文



目的

APPROX_PERCENTILE_DETAILは、exprの値について近似パーセンタイル情報を計算して、詳細と呼ばれるBLOB値を戻し、ここにはその情報が特殊な形式で含まれます。

exprで許容されるデータ型は、DETERMINISTIC句で指定するアルゴリズムによって異なります。詳細は、「DETERMINISTIC」句を参照してください。

通常、このファンクションはGROUP BY句とともにSELECT文で使用されます。これにより、行の各グループ内のexprに対する近似パーセンタイル情報が計算され、グループごとに単一の詳細が戻されます。

APPROX_PERCENTILE_DETAILによって戻される詳細は、APPROX_PERCENTILE_AGGファンクションへの入力として使用して詳細の集計を実行したり、TO_APPROX_PERCENTILEファンクションへの入力として使用して詳細を指定されたパーセンタイル値に変換できます。これらの3つのファンクションを一緒に使用すると、リソースを大量に使用する近似パーセンタイル計算を1回実行し、生成された詳細を格納した後、これらの詳細に対して効率的に集計と問合せを実行できます。たとえば:

1. APPROX_PERCENTILE_DETAILファンクションを使用して、近似パーセンタイル計算を実行し、生成される詳細を表またはマテリアライズド・ビューに格納します。これらは、市区町村の収入パーセンタイル情報など、粒度の高い詳細にできます。
2. APPROX_PERCENTILE_AGGファンクションを使用して、前のステップで取得した詳細を集計し、生成される詳細を表またはマテリアライズド・ビューに格納します。これらは、都道府県の収入パーセンタイル情報など、粒度のより低い詳細にできます。
3. TO_APPROX_PERCENTILEファンクションを使用して、格納した詳細値をパーセンタイル値に変換します。TO_APPROX_PERCENTILEファンクションを使用すると、APPROX_PERCENTILE_DETAILファンクションまたはAPPROX_PERCENTILE_AGGファンクションによって作成された詳細値を問い合せることができます。

DETERMINISTIC

この句を使用すると、近似パーセンタイル値を計算するために使用するアルゴリズムのタイプを制御できます。

- DETERMINISTICを指定すると、このファンクションによって、決定的な近似パーセンタイル情報が計算されます。この場合、exprは数値または数値に暗黙的に変換可能な値に評価される必要があります。
- DETERMINSTICを指定しないと、このファンクションによって、非決定的な近似パーセンタイル情報が計算されます。この場合、exprは、数値または日時値、あるいは数値または日時値に暗黙的に変換可能な値に評価される必要があります。

関連項目:

- APPROX_PERCENTILE_AGG
- TO_APPROX_PERCENTILE

例

この項の例では、APPROX_PERCENTILE_DETAIL、APPROX_PERCENTILE_AGGおよびTO_APPROX_PERCENTILEファンクションを一緒に使用して、リソースを大量に使用する近似パーセンタイル計算を1回実行し、生成された詳細を格納した後、これらの詳細に対して効率的に集計と問合せを実行する方法を示します。

APPROX_PERCENTILE_DETAIL: 例

次の文は、表sh.customersおよびsh.salesに対して、各顧客に販売された製品の金額を問い合せます。APPROX_PERCENTILE_DETAIL関数は、顧客が居住する各都市のcity_detailと呼ばれる詳細の情報を戻します。戻された詳細は、amt_sold_by_city_mvというマテリアライズド・ビューに格納されます。

```
CREATE MATERIALIZED VIEW amt_sold_by_city_mv
ENABLE QUERY REWRITE AS
SELECT c.country_id country,
       c.cust_state_province state,
       c.cust_city city,
       APPROX_PERCENTILE_DETAIL(s.amount_sold) city_detail
FROM customers c, sales s
WHERE c.cust_id = s.cust_id
GROUP BY c.country_id, c.cust_state_province, c.cust_city;
```

APPROX_PERCENTILE_AGG: 例

次の文は、APPROX_PERCENTILE_AGGファンクションを使用して、amt_sold_by_city_mvに格納されている詳細を読み取り、各都道府県の顧客に販売された製品の金額を含む集計された詳細を作成します。これらの集計された詳細は、amt_sold_by_state_mvというマテリアライズド・ビューに格納されます。

```
CREATE MATERIALIZED VIEW amt_sold_by_state_mv AS
SELECT country,
       state,
       APPROX_PERCENTILE_AGG(city_detail) state_detail
FROM amt_sold_by_city_mv
GROUP BY country, state;
```

次の文は、各国の顧客に販売された製品のおおよその金額を含む、集計された詳細を作成することを除き、前の文と似ています。これらの集計された詳細は、amt_sold_by_country_mvというマテリアライズド・ビューに格納されます。

```
CREATE MATERIALIZED VIEW amt_sold_by_country_mv AS
  SELECT country,
         APPROX_PERCENTILE_AGG(city_detail) country_detail
  FROM amt_sold_by_city_mv
  GROUP BY country;
```

TO_APPROX_PERCENTILE: 例

次の文は、TO_APPROX_PERCENTILEファンクションを使用して、amt_sold_by_city_mvに格納されている詳細を問い合せ、各市区町村の顧客に販売された製品の金額について25パーセンタイル、50パーセンタイルおよび75パーセンタイルの近似値を戻します。

```
SELECT country,
       state,
       city,
       TO_APPROX_PERCENTILE(city_detail, .25, 'NUMBER') "25th Percentile",
       TO_APPROX_PERCENTILE(city_detail, .50, 'NUMBER') "50th Percentile",
       TO_APPROX_PERCENTILE(city_detail, .75, 'NUMBER') "75th Percentile"
FROM amt_sold_by_city_mv
ORDER BY country, state, city;
COUNTRY STATE        CITY           25th Percentile 50th Percentile 75th Percentile
------- ------------ -------------- --------------- --------------- ---------------
  52769 Kuala Lumpur Kuala Lumpur              19.29            38.1           53.84
```

```
  52769 Penang       Batu Ferringhi          21.51          42.09          57.26
  52769 Penang       Georgetown              19.15          33.25          56.12
  52769 Selangor     Klang                   18.08          32.06          51.29
  52769 Selangor     Petaling Jaya           19.29          35.43           60.2
. . .
```

次の文は、TO_APPROX_PERCENTILEファンクションを使用して、amt_sold_by_state_mvに格納されている詳細を問い合せ、各都道府県の顧客に販売された製品の金額について25パーセンタイル、50パーセンタイルおよび75パーセンタイルの近似値を戻します。

```
SELECT country,
       state,
       TO_APPROX_PERCENTILE(state_detail, .25, 'NUMBER') "25th Percentile",
       TO_APPROX_PERCENTILE(state_detail, .50, 'NUMBER') "50th Percentile",
       TO_APPROX_PERCENTILE(state_detail, .75, 'NUMBER') "75th Percentile"
FROM amt_sold_by_state_mv
ORDER BY country, state;
COUNTRY STATE        25th Percentile 50th Percentile 75th Percentile
------- ------------ --------------- --------------- ---------------
  52769 Kuala Lumpur           19.29            38.1           53.84
  52769 Penang                 20.19           36.84           56.12
  52769 Selangor               16.97           32.41           52.69
  52770 Drenthe                16.76            31.7           53.89
  52770 Flevopolder            20.38           39.73           61.81
. . .
```

次の文は、TO_APPROX_PERCENTILEファンクションを使用して、amt_sold_by_country_mvに格納されている詳細を問い合せ、各国の顧客に販売された製品の金額について25パーセンタイル、50パーセンタイルおよび75パーセンタイルの近似値を戻します。

```
SELECT country,
       TO_APPROX_PERCENTILE(country_detail, .25, 'NUMBER') "25th Percentile",
       TO_APPROX_PERCENTILE(country_detail, .50, 'NUMBER') "50th Percentile",
       TO_APPROX_PERCENTILE(country_detail, .75, 'NUMBER') "75th Percentile"
FROM amt_sold_by_country_mv
ORDER BY country;
  COUNTRY 25th Percentile 50th Percentile 75th Percentile
--------- --------------- --------------- ---------------
    52769            19.1           35.43           52.78
    52770           19.29           38.99           59.58
    52771           11.99           44.99          561.47
    52772           18.08           33.72           54.16
    52773           15.67           29.61           50.65
. . .
```

APPROX_PERCENTILE_AGGでは、その入力として、近似パーセンタイル情報を含む詳細の列を取得し、該当する情報の集計を実行できます。次の文は、近似パーセンタイル詳細をAPPROX_PERCENTILE_AGGによって解釈して、TO_APPROX_PERCENTILEファンクションへの入力を指定する方法を示します。前の例と同様、この問合せは、各国の顧客に販売された製品の金額について25パーセンタイル近似値を戻します。この結果は、前の例で25パーセンタイルに対して戻された結果と同じであることに注意してください。

```
SELECT country,
       TO_APPROX_PERCENTILE(APPROX_PERCENTILE_AGG(city_detail), .25, 'NUMBER') "25th
Percentile"
FROM amt_sold_by_city_mv
GROUP BY country
ORDER BY country;
  COUNTRY 25th Percentile
--------- ---------------
     52769            19.1
     52770           19.29
     52771           11.99
```

```
     52772           18.08
     52773           15.67
. . .
```

近似問合せに基づくクエリー・リライトおよびマテリアライズド・ビュー: 例

「APPROX_PERCENTILE_DETAIL: 例」では、ENABLE QUERY REWRITE句は、マテリアライズド・ビューamt_sold_by_city_mvの作成時に指定されます。これにより、マテリアライズド・ビューを使用して、APPROX_MEDIANやAPPROX_PERCENTILEなどの近似処理ファンクションを含む問合せをリライトできます。

たとえば、データベース・レベルか現行のセッションで、クエリー・リライトが有効になっていることを確認し、次の問合せを実行します。

```
SELECT c.country_id country,
       APPROX_MEDIAN(s.amount_sold) amount_median
FROM customers c, sales s
WHERE c.cust_id = s.cust_id
GROUP BY c.country_id;
```

DBMS_XPLANを問い合せて計画をEXPLAINします。

```
SET LINESIZE 300
SET PAGESIZE 0
COLUMN plan_table_output FORMAT A150
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(format=>'BASIC'));
```

次の計画に示すように、オプティマイザでは、問合せにマテリアライズド・ビューamt_sold_by_city_mvが使用されました。

```
EXPLAINED SQL STATEMENT:
------------------------
SELECT c.country_id country, APPROX_MEDIAN(s.amount_sold)
amount_median FROM customers c, sales s WHERE c.cust_id = s.cust_id
GROUP BY c.country_id
Plan hash value: 2232676046
-----------------------------------------------------------------
| Id  | Operation                     | Name                |
-----------------------------------------------------------------
|   0 | SELECT STATEMENT              |                     |
|   1 |  HASH GROUP BY APPROX         |                     |
|   2 |   MAT_VIEW REWRITE ACCESS FULL| AMT_SOLD_BY_CITY_MV |
-----------------------------------------------------------------
```

APPROX_RANK

構文

目的

APPROX_RANKは、値のグループでの近似値を戻します。

このファンクションは、オプションのPARTITION BY句と、その後に続く必須のORDER BY ... DESC 句を取ります。PARTITION BYキーは、GROUP BYキーのサブセットである必要があります。ORDER BY句には、APPROX_COUNTまたはAPPROX_SUMのいずれかを含める必要があります。

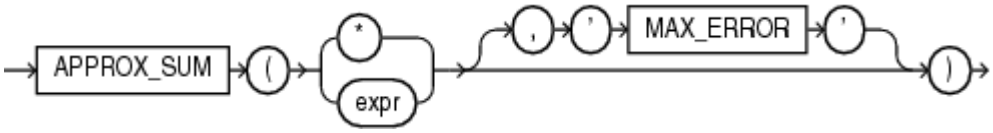
例

この問合せでは、部門ごとの上位10件の合計給与のジョブが戻されます。ジョブごとに、合計給与およびランクも示されます。

```
SELECT job_id,
                         APPROX_SUM(sal),
      APPROX_RANK(PARTITION BY department_id ORDER BY APPROX_SUM(salary) DESC)
FROM employees
GROUP BY department_id, job_id
HAVING
   APPROX_RANK(
   PARTITION BY department_id
   ORDER BY APPROX_SUM (salary)
   DESC) <= 10;
```

APPROX_SUM

構文



目的

APPROX_SUMは、式のおおよその合計を戻します。MAX_ERRORを2番目の引数として指定すると、このファンクションは、実際の合計とおおよその合計の間の最大エラーを戻します。

このファンクションは、HAVING句で対応するAPPROX_RANKファンクションとともに使用する必要があります。問合せでAPPROX_COUNT、APPROX_SUMまたはAPPROX_RANKを使用する場合、その問合せでは他の集計ファンクションを使用しないでください。

入力値が負の数値である場合、APPROX_SUMはエラーを戻すことに注意してください。

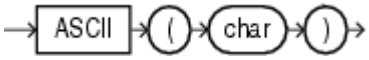
例

次の問合せでは、給与の集計が最も高い各部門内の10個のジョブ・タイプが戻されます。

```
SELECT department_id, job_id,
       APPROX_SUM(salary)
FROM   employees
GROUP BY department_id, job_id
HAVING
  APPROX_RANK (
  PARTITION BY department_id
  ORDER BY APPROX_SUM(salary)
  DESC ) <= 10;
```

ASCII

構文



目的

ASCIIは、charの最初の文字の、データベース文字セットでの10進表記を戻します。

charのデータ型は、CHAR、VARCHAR2、NCHARまたはNVARCHAR2です。戻り値のデータ型はNUMBERです。データベース文字セットが7ビットのASCIIの場合、このファンクションはASCII値を戻します。データベース文字セットがEBCDICコードの場合、このファンクションはEBCDIC値を戻します。このファンクションと一致するEBCDIC文字ファンクションは存在しません。

このファンクションは、CLOBデータを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用してCLOBを引数として渡すことはできます。

関連項目:

詳細は、データ型の比較規則を参照してください

例

次の例では、姓が文字「L」で始まり、ASCII表記が76の従業員を戻します。

```
SELECT last_name
  FROM employees
  WHERE ASCII(SUBSTR(last_name, 1, 1)) = 76
  ORDER BY last_name;

LAST_NAME
-------------------------
Ladwig
Landry
Lee
Livingston
Lorentz
```

ASCIISTR

構文

目的

ASCIISTRは、任意の文字セットの文字列、または文字列に変換する式を引数として取り、データベース文字セットのASCII文字列を戻します。ASCII文字以外は、¥xxxxという書式に変換されます。xxxxは、UTF-16のコード単位です。

関連項目:

- Unicode文字セットおよび文字セマンティクスの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。
- ASCIISTRの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

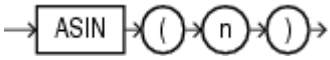
例

次の例では、テキスト文字列「ABÄCDE」をASCII文字列で戻します。

```
SELECT ASCIISTR('ABÄCDE')
  FROM DUAL;
ASCIISTR('
----------
AB¥00C4CDE
```

ASIN

構文



目的

ASINは、nのアーク・サインを戻します。引数nは-1から1の範囲で、ファンクションによって戻される値は-π/2からπ/2(ラジアン)の範囲です。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数がBINARY_FLOATの場合、このファンクションはBINARY_DOUBLEを戻します。それ以外の場合、引数と同じ数値データ型を戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

例

次の例では、.3のアーク・サインを戻します。

```
SELECT ASIN(.3) "Arc_Sine"
  FROM DUAL;
 Arc_Sine
----------
.304692654
```

ATAN

構文

目的

ATANは、nのアーク・タンジェントを戻します。引数nの範囲に制限はなく、ファンクションによって戻される値は-π/2からπ/2(ラジアン)の範囲です。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数がBINARY_FLOATの場合、このファンクションはBINARY_DOUBLEを戻します。それ以外の場合、引数と同じ数値データ型を戻します。

関連項目:

ATAN2ファンクションの詳細は、「ATAN2」を参照してください。暗黙的な変換の詳細は、表2-8を参照してください。

例

次の例では、.3のアーク・タンジェントを戻します。

```
SELECT ATAN(.3) "Arc_Tangent"
  FROM DUAL;
Arc_Tangent
----------
.291456794
```

ATAN2

構文

目的

ATAN2は、n1およびn2のアーク・タンジェントを戻します。引数n1の範囲に制限はなく、n1とn2の符号により、ファンクションによって戻される値は-piからpi (ラジアン)の範囲です。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。いずれかの引数がBINARY_FLOATまたはBINARY_DOUBLEの場合、このファンクションはBINARY_DOUBLEを戻します。それ以外の場合、NUMBERを戻します。

関連項目:

ATANファンクションの詳細は、「ATAN」を参照してください。暗黙的な変換の詳細は、表2-8を参照してください。

例

次の例では、.3および.2のアーク・タンジェントを戻します。

```
SELECT ATAN2(.3, .2) "Arc_Tangent2"
  FROM DUAL;

Arc_Tangent2
------------
  .982793723
```

AVG

構文

関連項目:

構文、セマンティクスおよび制限事項の詳細は、分析ファンクションを参照してください。

目的

AVGは、exprの平均値を戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を返します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

DISTINCTを指定する場合は、analytic_clauseのquery_partition_clauseのみを指定できます。order_by_clauseおよびwindowing_clauseは指定できません。

関連項目:

exprの有効な書式の詳細は、SQL式を参照してください。また、集計ファンクションを参照してください

集計の例

次の例では、hr.employees表にあるすべての従業員の平均給与を計算します。

```
SELECT AVG(salary) "Average"
  FROM employees;
       Average
--------------
    6461.83178
```

分析の例

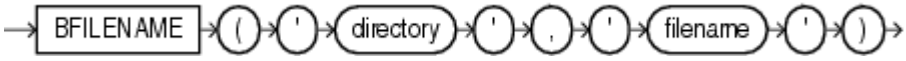
次の例では、employees表の各従業員について、ある期間内に雇用された従業員の平均給与を所属別に計算します。

```
SELECT manager_id, last_name, hire_date, salary,
       AVG(salary) OVER (PARTITION BY manager_id ORDER BY hire_date
  ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS c_mavg
  FROM employees
  ORDER BY manager_id, hire_date, salary;
MANAGER_ID LAST_NAME                 HIRE_DATE     SALARY     C_MAVG
---------- ------------------------- --------- ---------- ----------
       100 De Haan                   13-JAN-01      17000      14000
       100 Raphaely                  07-DEC-02      11000 11966.6667
       100 Kaufling                  01-MAY-03       7900 10633.3333
```

```
       100 Hartstein                 17-FEB-04      13000 9633.33333
       100 Weiss                     18-JUL-04       8000 11666.6667
       100 Russell                   01-OCT-04      14000 11833.3333
       100 Partners                  05-JAN-05      13500 13166.6667
       100 Errazuriz                 10-MAR-05      12000 11233.3333
. . .
```

BFILENAME

構文



目的

BFILENAMEは、サーバーのファイル・システムの物理LOBバイナリ・ファイルに対応付けられているBFILEロケータを戻します。

- 'directory'は、実際にファイルが存在するサーバーのファイル・システム上のフルパス名に対する別名となるデータベース・オブジェクトです。
- 'filename'は、サーバーのファイル・システムにあるファイルの名前です。

SQL文、PL/SQL文、DBMS_LOBパッケージまたはOCIの操作で、これらをBFILENAMEへの引数として使用するには、まずディレクトリ・オブジェクトを作成し、BFILE値を物理ファイルと対応付ける必要があります。

次の2つの方法で、このファンクションを使用できます。

- DML文でBFILE列を初期化する場合
- プログラム・インタフェースで、BFILEロケータに値を割り当てることによってBFILEデータにアクセスする場合

ディレクトリの引数の大/小文字は区別されます。データ・ディクショナリ内に存在する名前と同じ名前でディレクトリ・オブジェクト名を指定しているかを確認する必要があります。たとえば、CREATE DIRECTORY文で、大/小文字を組み合せた識別子を引用符で囲んで使用してAdminディレクトリ・オブジェクトを作成すると、BFILENAMEファンクションを使用する場合、ディレクトリ・オブジェクトを'Admin'として指定する必要があります。filename引数は、ご使用のオペレーティング・システムの大/小文字および記号の表記規則に従って指定する必要があります。

関連項目:

- LOBの詳細およびBFILEデータの検出例については、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』および『Oracle Call Interfaceプログラマーズ・ガイド』を参照してください。
- CREATE DIRECTORY

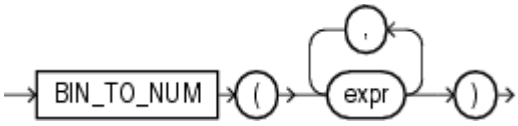
例

次の例では、サンプル表pm.print_mediaに行を挿入します。BFILENAMEファンクションを使用して、/demo/schema/product_mediaディレクトリにある、サーバーのファイル・システムのバイナリ・ファイルを識別します。次の例では、PMスキーマでのディレクトリのデータベース・オブジェクトmedia_dirの作成方法を示します。

```
CREATE DIRECTORY media_dir AS '/demo/schema/product_media';
INSERT INTO print_media (product_id, ad_id, ad_graphic)
  VALUES (3000, 31001, BFILENAME('MEDIA_DIR', 'modem_comp_ad.gif'));
```

BIN_TO_NUM

構文



目的

BIN_TO_NUMは、ビット・ベクトルを同等の数値に変換します。このファンクションの各引数は、ビット・ベクトルのビットを表します。このファンクションは、引数として、任意の数値データ型、または暗黙的にNUMBERに変換可能な数値以外のデータ型を取ります。各exprは、0または1に評価される必要があります。このファンクションはOracleのNUMBER値を戻します。

BIN_TO_NUMは、データ・ウェアハウスのアプリケーションで、グルーピング・セットを使用して、マテリアライズド・ビューから対象グループを検索する場合に有効です。

関連項目:

- GROUPING SETS構文の詳細は、「group_by_clause」を参照してください。
- 暗黙的な変換の詳細は、表2-8を参照してください。
- データ集計の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

例

次の例では、バイナリの値を数値に変換します。

```
SELECT BIN_TO_NUM(1,0,1,0)
  FROM DUAL;
BIN_TO_NUM(1,0,1,0)
-------------------
                 10
```

次の例では、3つの値を1つのバイナリの値に変換し、BIN_TO_NUMを使用してこのバイナリを数値に変換します。この例では、PL/SQL宣言を使用して元の値を指定します。これらの値は、通常は実際のデータソースから導出されます。

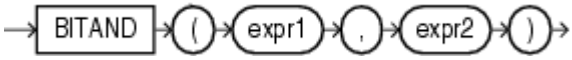
```
SELECT order_status
  FROM orders
  WHERE order_id = 2441;
ORDER_STATUS
------------
           5
DECLARE
  warehouse NUMBER := 1;
  ground    NUMBER := 1;
  insured   NUMBER := 1;
  result    NUMBER;
BEGIN
  SELECT BIN_TO_NUM(warehouse, ground, insured) INTO result FROM DUAL;
  UPDATE orders SET order_status = result WHERE order_id = 2441;
END;
/
PL/SQL procedure successfully completed.
SELECT order_status
  FROM orders
  WHERE order_id = 2441;
```

```
ORDER_STATUS
------------
           7
```

この逆の(1つの列値から複数の値を抽出する)プロセスの詳細は、「BITAND」の例を参照してください。

BITAND

構文



目的

BITANDファンクションは、その入力と出力をビットのベクトルとして扱います。出力は、入力のビット単位ANDになります。

expr1およびexpr2の型はNUMBERであり、結果はNUMBER型になります。BITANDのいずれかの引数がNULLの場合、結果はNULLになります。

引数は、$-(2^{(n-1)})$ .. $((2^{(n-1)})-1)$の範囲にする必要があります。引数をこの範囲外にすると、結果は未定義になります。

結果は、いくつかのステップで計算されます。まず、それぞれの引数Aが値SIGN(A)*FLOOR(ABS(A))に置き換えられます。この変換では、各引数の小数点以下が切り捨てられます。次に、整数値となったそれぞれの引数Aが、nビットの2の補数のバイナリ整数値に変換されます。ビット単位AND演算を使用して、2つのビット値が組み合せられます。最後に、生成されたnビットの2の補数値がNUMBERに変換されます。

BITANDファンクションのノート

- BITANDの現行の実装では、n = 128が定義されています。
- PL/SQLは、入力と結果の型がすべてBINARY_INTEGERで、n = 32のBITANDのオーバーロードをサポートしています。

例

次の例では、数値6(バイナリ1,1,0)と数値3(バイナリ0,1,1)にAND演算を実行します。

```
SELECT BITAND(6,3)
  FROM DUAL;
BITAND(6,3)
-----------
          2
```

これは、6と3のバイナリの値を示す次の例と同じです。BITANDファンクションは、バイナリの値の有効桁のみで演算を実行します。

```
SELECT BITAND(
    BIN_TO_NUM(1,1,0),
    BIN_TO_NUM(0,1,1)) "Binary"
  FROM DUAL;

    Binary
----------
         2
```

単一の列値における複数の値のエンコードの詳細は、「BIN_TO_NUM」の例を参照してください。

次の例では、サンプル表oe.ordersのorder_status列で複数の選択項目を1つの数値内の個別のビットとしてエンコードします。たとえば、まだ倉庫にある発注は、バイナリの値001(10進値1)で表されます。陸上輸送で発送される発注は、バイナリの値010(10進値2)で表されます。保険がかけられた荷物は、バイナリの値100(10進値4)で表されます。この例では、DECODEファンクションを使用して、order_status値の3つのビットのそれぞれに2つの値(ビットがオンの場合の値とビットがオフの場合の値)が示されています。

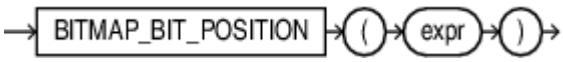
```
SELECT order_id, customer_id, order_status,
    DECODE(BITAND(order_status, 1), 1, 'Warehouse', 'PostOffice') "Location",
    DECODE(BITAND(order_status, 2), 2, 'Ground', 'Air') "Method",
    DECODE(BITAND(order_status, 4), 4, 'Insured', 'Certified') "Receipt"
  FROM orders
  WHERE sales_rep_id = 160
  ORDER BY order_id;
  ORDER_ID CUSTOMER_ID ORDER_STATUS Location   Method Receipt
---------- ----------- ------------ ---------- ------ ---------
      2416         104            6 PostOffice Ground Insured
      2419         107            3 Warehouse  Ground Certified
      2420         108            2 PostOffice Ground Certified
      2423         145            3 Warehouse  Ground Certified
      2441         106            5 Warehouse  Air    Insured
      2455         145            7 Warehouse  Ground Insured
```

Location列では、まずBITANDによってorder_statusと1(バイナリ001)が比較されます。比較されるのは重要なビット値のみです。したがって、右端のビットが1であるバイナリの値(奇数)は正と評価され、1が戻されます。偶数の場合は0(ゼロ)が戻されます。DECODEファンクションでは、BITANDによって戻された値と1が比較されます。両方とも1の場合、場所はWarehouse(倉庫)になります。これらの値が異なる場合、場所はPostOffice(郵便局)になります。

Method列とReceipt列は、同様に計算されます。Methodでは、BITANDによってorder_statusと2(バイナリ010)にAND演算が実行されます。Receiptでは、BITANDによってorder_statusと4(バイナリ100)にAND演算が実行されます。

BITMAP_BIT_POSITION

構文



目的

数値とビット位置の間で1対1のマッピングを作るには、BITMAP_BIT_POSITIONを使用します。

引数exprは、NUMBER型です。ビットマップにおける絶対ビット位置です。

BITMAP_BIT_POSITIONは、相対ビット位置NUMBERを返します。

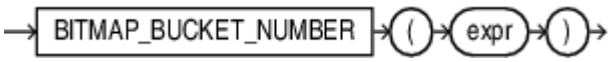
exprがNULLの場合、この関数はNULLを返します。

exprが整数でない場合には、次のエラー・メッセージが表示されます。

```
Invalid value has been passed to a BITMAP COUNT DISTINCT related operator.
```

BITMAP_BUCKET_NUMBER

構文



目的

数値とビットマップのビット位置との間で1対1のマッピングを作るには、BITMAP_BUCKET_NUMBERを使用します。

引数exprは、NUMBER型です。ビットマップにおける絶対ビット位置を表します。

BITMAP_BUCKET_NUMBERは、NUMBERを返します。相対ビット位置を表します。

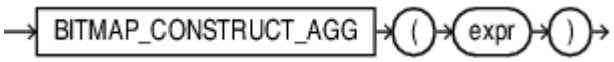
exprがNULLの場合、この関数はNULLを返します。

exprが整数でない場合には、次のエラー・メッセージが表示されます。

```
Invalid value has been passed to a BITMAP COUNT DISTINCT related operator.
```

BITMAP_CONSTRUCT_AGG

構文



目的

BITMAP_CONSTRUCT_AGG は、ビット位置を演算する集計ファンクションで、入力のビット位置すべて集合を表すビットマップ表記を返します。基本的にはビットマップを保持し、入力のビット位置すべてをそこに設定します。ビットマップ表記を返します。

引数exprは、NUMBER型です。

戻り型はBLOB型です。

exprがNULLの場合、この関数はNULLを返します。

制限事項

- 引数はNUMBER型です。入力値を自然数に変換できない場合は、ORA-62575エラーが発生します。

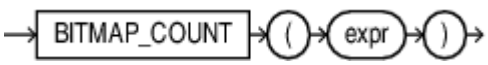
```
62575, 00000, "Invalid value has been passed to a BITMAP COUNT DISTINCT related
operator."
// *Cause: An attempt was made to pass an invalid value to a BITMAP COUNT
DISTINCT operator.
// *Action: Pass only natural number values to BITMAP_CONSTRUCT_AGG.
```

- ビットマップがBLOBの最大値を超える場合は、ORA-62577エラーが表示されます。

```
62577, 00000, "The bitmap size exceeds maximum size of its SQL data type."
// *Cause: An attempt was made to construct a bitmap larger than its maximum
SQL type size.
// *Action: Break the input to BITMAP_CONSTRUCT_AGG into smaller ranges.
```

BITMAP_COUNT

構文



目的

BITMAP_COUNTは、入力のビットマップに対して1ビットのカウントを返すスカラー・ファンクションです。

引数exprは、BLOB型です。

入力で設定されたビットのカウントを表すNUMBERを返します。

exprがNULLの場合は、0を返します。

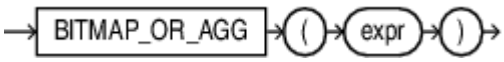
制限事項

引数はBLOB型である必要があります。引数は、BITMAP_CONSTRUCT_AGGによって、または再帰的にBITMAP_OR_AGGによって生成されるビットマップです。入力がそれ以外の場合は、ORA-62578エラーになります。

```
62578, 00000, "The input is not a valid bitmap produced by BITMAP COUNT DISTINCT
related operators."
// *Cause: An attempt was made to pass a bitmap that was not produced by one of the
BITMAP COUNT DISTINCT operators.
// *Action: Only pass bitmaps constructed via BITMAP_CONSTRUCT_AGG or BITMAP_OR_AGG
to BITMAP COUNT DISTINCT related operators.
```

BITMAP_OR_AGG

構文



目的

BITMAP_OR_AGGは、ビットマップを演算して入力のORを計算する集計ファンクションです。

引数exprは、BLOB型である必要があります。

戻り型はBLOB型です。集計したビットマップすべてのORを表すビットマップを返します。

BITMAP_OR_AGGの出力は、人が読める形ではありません。BITMAP_OR_AGGを介して、またはスカラー・ファンクションBITMAP_COUNTによってさらに集計処理されるものと想定されています。

exprがNULLの場合、この関数はNULLを返します。

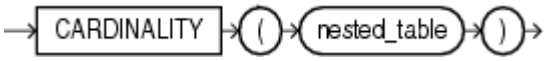
制限事項

引数はBLOB型である必要があります。引数は、BITMAP_CONSTRUCT_AGGによって、または再帰的にBITMAP_OR_AGGによって生成されるビットマップです。入力がそれ以外の場合は、ORA-62578エラーになります。

```
62578, 00000, "The input is not a valid bitmap produced by BITMAP COUNT DISTINCT
related operators."
// *Cause: An attempt was made to pass a bitmap that was not produced by one of the
BITMAP COUNT DISTINCT operators.
// *Action: Only pass bitmaps constructed via BITMAP_CONSTRUCT_AGG or BITMAP_OR_AGG
to BITMAP COUNT DISTINCT related operators.
```

CARDINALITY

構文



目的

CARDINALITYは、ネストした表内の要素数を戻します。戻り型は、NUMBERです。ネストした表が空であるかNULLの集合である場合、CARDINALITYはNULLを戻します。

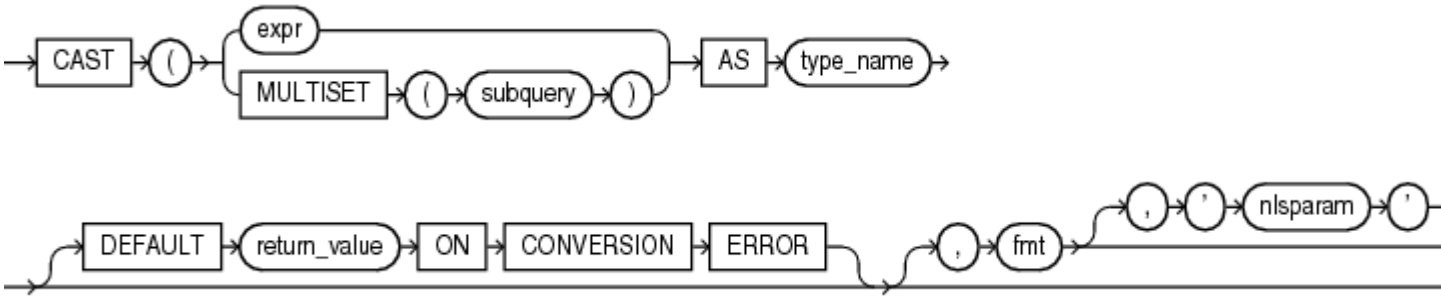
例

次の例では、pm.print_mediaサンプル表のネストした表の列ad_textdocs_ntab内の要素数を示します。

```
SELECT product_id, CARDINALITY(ad_textdocs_ntab) cardinality
  FROM print_media
  ORDER BY product_id;
PRODUCT_ID CARDINALITY
---------- -----------
      2056           3
      2268           3
      3060           3
      3106           3
```

CAST

構文



目的

CASTを使用すると、ある組込みデータ型またはコレクション型の値を、別の組込みデータ型またはコレクション型に変換できます。名前のないオペランド(日付や副問合せの結果セットなど)または名前付きのコレクション(VARRAYやネストした表など)を型互換のデータ型または名前付きコレクションにキャストできます。type_nameは、組込みデータ型またはコレクション型の名前である必要があり、オペランドは、組込みデータ型であるか、またはその値がコレクション値である必要があります。

オペランドでは、exprは組込みデータ型、コレクション型またはANYDATA型のインスタンスのいずれかです。exprがANYDATA型のインスタンスである場合、CASTはANYDATAインスタンスの値を抽出し、その値がキャスト対象の型と一致する場合はその値を戻し、一致しない場合はNULLを戻します。MULTISETは、副問合せの結果セットを取り、コレクション値を戻すようにOracle Databaseに通知します。表7-1に、どの組込みデータ型が、どの組込みデータ型にキャストできるかを示します。(CASTは、LONG型、LONG RAW型、またはOracleが提供する型をサポートしていません。)

CASTは、LOBデータ型のいずれも直接的にサポートしていません。CASTを使用して、CLOB値を文字データ型に変換するか、BLOB値をRAWデータ型に変換すると、データベースは暗黙的にLOB値を文字またはRAWデータに変換し、結果値を明示的にターゲットのデータ型にキャストします。結果値がターゲットの型より大きい場合、エラーが戻されます。

CAST ... MULTISETを使用してコレクション値を取得すると、CASTファンクションに渡される問合せ内のSELECT構文のリストのアイテムは、ターゲットのコレクション要素型に対応する属性型に変換されます。

表7-1 組込みデータ型のキャスト

宛先データ型	BINARY_FLOAT、BINARY_DOUBLEから	CHAR、VARCHAR2から	NUMBER/INTEGERから	DATETIME/INTERVALから(ノート1)	RAWから	ROWID、UROWIDから(ノート2)	NCHAR、NVARCHAR2から
BINARY_FLOAT、BINARY_DOUBLE へ	X (ノート 3)	X (ノート 3)	X (ノート 3)	--	--	--	X (ノート 3)
CHAR、VARCHAR2 へ	X	X	X	X	X	X	--

宛先データ型	BINARY_FLOAT、BINARY_DOUBLEから	CHAR、VARCHAR2から	NUMBER/INTEGERから	DATETIME/INTERVALから(ノート1)	RAWから	ROWID、UROWIDから(ノート2)	NCHAR、NVARCHAR2から
NUMBER/INTEGER へ	X (ノート 3)	X (ノート 3)	X (ノート 3)	--	--	--	X (ノート 3)
DATETIME/INTERVAL へ	--	X (ノート 3)	--	X (ノート 3)	--	--	--
RAW へ	--	X	--	--	X	--	--
ROWID、UROWID へ	--	X	--	--	--	X	--
NCHAR、NVARCHAR2 へ	X	--	X	X	X	X	X

ノート1: DATETIME/INTERVALには、DATE、TIMESTAMP、TIMESTAMP WITH TIMEZONE、TIMESTAMP WITH LOCAL TIME ZONE、INTERVAL DAY TO SECONDおよびINTERVAL YEAR TO MONTHが含まれます。

ノート2: UROWIDが索引構成表のROWIDの値を含んでいる場合、UROWIDをROWIDにキャストすることはできません。

ノート3: このタイプの変換には、DEFAULT return_value ON CONVERSION ERROR句を指定できます。このタイプの変換には、fmtおよびnlsparam句を指定できます。例外として、INTERVAL DAY TO SECONDに変換する際にはfmtを指定することはできず、INTERVAL YEAR TO MONTHに変換する際にはfmtまたはnlsparamを指定することはできません。

名前付きコレクション型を別の名前付きコレクション型にキャストするには、両方のコレクションの要素が同じ型である必要があります。

関連項目:

- Oracle Databaseで暗黙的にコレクション型データを文字データに変換する方法の詳細は、暗黙的なデータ変換の例を参照してください。また、データ変換のセキュリティ上の考慮事項を参照してください。
- CASTの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Database グローバリゼーション・サポート・ガイド』の付録Cを参照してください。

MULTISET

副問合せの結果セットが複数行に評価される可能性がある場合は、MULTISETキーワードを指定する必要があります。副問合せの結果である行は、それらの行がキャストされたコレクション値の要素を形成します。MULTISETキーワードを省略すると、副問合せはスカラー副問合せとして処理されます。

MULTISETの制限事項

MULTISETキーワードを指定する場合、DEFAULT return_value ON CONVERSION ERROR、fmtまたはnlsparam句を指定することはできません。

DEFAULT return_value ON CONVERSION ERROR

この句を使用すると、exprをtype_nameに変換するときにエラーが発生した場合、このファンクションによって戻される値を指定できます。exprを評価するときにエラーが発生した場合、この句による影響はありません。

この句は、exprがCHAR、VARCHAR2、NCHARまたはNVARCHAR2型の文字列に評価され、type_nameがBINARY_DOUBLE、BINARY_FLOAT、DATE、INTERVAL DAY TO SECOND、INTERVAL YEAR TO MONTH、NUMBER、TIMESTAMP、TIMESTAMP WITH TIME ZONEまたはTIMESTAMP WITH LOCAL TIME ZONEである場合に有効です。

return_valueは、文字リテラル、NULL、定数式またはバインド変数にすることができ、NULLに評価されるか、CHAR、VARCHAR2、NCHAR、またはNVARCHAR2型の文字列に評価される必要があります。return_valueをtype_nameに変換できない場合は、エラーが戻されます。

fmtおよびnlsparam

fmt引数を使用すると書式モデルを指定でき、nlsparam引数を使用するとNLSパラメータを指定できます。これらの引数を指定すると、exprとreturn_valueが指定されている場合に、それらがtype_nameに変換されるときに適用されます。

fmtおよびnlsparamは、type_nameが次のいずれかのデータ型である場合に指定できます。

- BINARY_DOUBLE

  BINARY_DOUBLEを指定した場合、オプションのfmtおよびnlsparam引数は、TO_BINARY_DOUBLE関数の場合と同じ役割を果たします。詳細は、「TO_BINARY_DOUBLE」を参照してください。

- BINARY_FLOAT

  BINARY_FLOATを指定した場合、オプションのfmtおよびnlsparam引数は、TO_BINARY_FLOAT関数の場合と同じ役割を果たします。詳細は、「TO_BINARY_FLOAT」を参照してください。

- DATE

  DATEを指定する場合、オプションのfmtおよびnlsparam引数の用途はTO_DATEファンクションと同じです。詳細は、「TO_DATE」を参照してください。

- NUMBER

  NUMBERを指定する場合、オプションのfmtおよびnlsparam引数の用途はTO_NUMBERファンクションと同じです。詳細は、「TO_NUMBER」を参照してください。

- TIMESTAMP

  TIMESTAMPを指定する場合、オプションのfmtおよびnlsparam引数の用途はTO_TIMESTAMPファンクションと同じです。fmtを指定しない場合、exprは、NLS_TIMESTAMP_FORMATパラメータで明示的に決定されるか、またはNLS_TERRITORYパラメータで暗黙的に決定されたTIMESTAMPデータ型のデフォルト書式である必要があります。詳細は、「TO_TIMESTAMP」を参照してください。

- TIME WITH TIME ZONE

  TIMESTAMP WITH TIME ZONEを指定する場合、オプションのfmtおよびnlsparam引数の用途はTO_TIMESTAMP_TZファンクションと同じです。fmtを指定しない場合、exprは、NLS_TIMESTAMP_TZ_FORMATパラメータで明示的に決定されるか、またはNLS_TERRITORYパラメータで暗黙

的に決定されたTIMESTAMP WITH TIME ZONEデータ型のデフォルト書式である必要があります。詳細は、「TO_TIMESTAMP_TZ」を参照してください。

- TIMESTAMP WITH LOCAL TIME ZONE

  TIMESTAMP WITH LOCAL TIME ZONEを指定する場合、オプションのfmtおよびnlsparam引数の用途はTO_TIMESTAMPファンクションと同じです。fmtを指定しない場合、exprは、NLS_TIMESTAMP_FORMATパラメータで明示的に決定されるか、またはNLS_TERRITORYパラメータで暗黙的に決定されたTIMESTAMPデータ型のデフォルト書式である必要があります。詳細は、「TO_TIMESTAMP」を参照してください。

組込みデータ型の例

次の例では、CASTファンクションをスカラー・データ型とともに使用します。最初の例は、セッション・パラメータNLS_TIMESTAMP_FORMATに指定された書式モデルを使用して、テキストをタイムスタンプ値に変換します。このNLSパラメータに依存しないようにする場合は、2番目の例のようにTO_DATEを使用できます。

```
SELECT CAST('22-OCT-1997'
       AS TIMESTAMP WITH LOCAL TIME ZONE)
  FROM DUAL;
SELECT CAST(TO_DATE('22-Oct-1997', 'DD-Mon-YYYY')
       AS TIMESTAMP WITH LOCAL TIME ZONE)
  FROM DUAL;
```

前述の例では、TO_DATEでテキストをDATEに変換し、CASTでDATEをTIMESTAMP WITH LOCAL TIME ZONEに変換して、セッション・タイムゾーン(SESSIONTIMEZONE)の日付を求めています。

```
SELECT product_id, CAST(ad_sourcetext AS VARCHAR2(30)) text
  FROM print_media
  ORDER BY product_id;
```

次の例では、指定された値を指定されたデータ型に変換するときにエラーが発生した場合、デフォルト値が戻されます。これらの例では、エラーが発生することなく変換が行われます。

```
SELECT CAST(200
       AS NUMBER
       DEFAULT 0 ON CONVERSION ERROR)
  FROM DUAL;

SELECT CAST('January 15, 1989, 11:00 A.M.'
       AS DATE
       DEFAULT NULL ON CONVERSION ERROR,
       'Month dd, YYYY, HH:MI A.M.')
  FROM DUAL;

SELECT CAST('1999-12-01 11:00:00 -8:00'
       AS TIMESTAMP WITH TIME ZONE
       DEFAULT '2000-01-01 01:00:00 -8:00' ON CONVERSION ERROR,
       'YYYY-MM-DD HH:MI:SS TZH:TZM',
       'NLS_DATE_LANGUAGE = American')
  FROM DUAL;
```

次の例では、'N/A'をNUMBER値に変換するときにエラーが発生します。したがって、CASTファンクションにより、デフォルト値である0が戻されます。

```
SELECT CAST('N/A'
       AS NUMBER
       DEFAULT '0' ON CONVERSION ERROR)
  FROM DUAL;
```

コレクションの例

後に続くCASTの例は、サンプルの注文入力スキーマoeで使用されているcust_address_typに基づいています。

```
CREATE TYPE address_book_t AS TABLE OF cust_address_typ;
/
CREATE TYPE address_array_t AS VARRAY(3) OF cust_address_typ;
/
CREATE TABLE cust_address (
  custno            NUMBER,
  street_address    VARCHAR2(40),
  postal_code       VARCHAR2(10),
  city              VARCHAR2(30),
  state_province    VARCHAR2(10),
  country_id        CHAR(2));
CREATE TABLE cust_short (custno NUMBER, name VARCHAR2(31));
CREATE TABLE states (state_id NUMBER, addresses address_array_t);
```

次の例では、副問合せをキャストします。

```
SELECT s.custno, s.name,
       CAST(MULTISET(SELECT ca.street_address,
                            ca.postal_code,
                            ca.city,
                            ca.state_province,
                            ca.country_id
                       FROM cust_address ca
                       WHERE s.custno = ca.custno)
       AS address_book_t)
  FROM cust_short s
  ORDER BY s.custno;
```

CASTでは、VARRAY型の列をネストした表に変換します。

```
SELECT CAST(s.addresses AS address_book_t)
  FROM states s
  WHERE s.state_id = 111;
```

次の例では、この後に示す例で使用するオブジェクトを作成します。

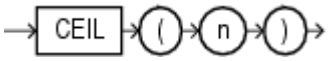
```
CREATE TABLE projects
  (employee_id NUMBER, project_name VARCHAR2(10));
CREATE TABLE emps_short
  (employee_id NUMBER, last_name VARCHAR2(10));
CREATE TYPE project_table_typ AS TABLE OF VARCHAR2(10);
/
```

次のMULTISET式の例は、前述のオブジェクトを使用します。

```
SELECT e.last_name,
       CAST(MULTISET(SELECT p.project_name
                       FROM projects p
                       WHERE p.employee_id = e.employee_id
                       ORDER BY p.project_name)
       AS project_table_typ)
  FROM emps_short e
  ORDER BY e.last_name;
```

CEIL

構文



目的

CEILは、n以上の最も小さい整数を戻します。数値nは常に、0 <= f < 1とn = k - fなどのように、整数kと正の小数部fの差として記述できます。CEILの値は整数kです。したがって、CEILの値がnそのものになるのは、nが小数部を持たない整数である場合にかぎられます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を返します。

関連項目:

暗黙的な変換およびFLOORの詳細は、表2-8を参照してください。

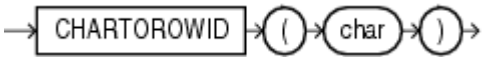
例

次の例では、指定した注文の合計以上である最小の整数を戻します。

```
SELECT order_total, CEIL(order_total)
  FROM orders
  WHERE order_id = 2434;
ORDER_TOTAL CEIL(ORDER_TOTAL)
----------- -----------------
   268651.8            268652
```

CHARTOROWID

構文



目的

CHARTOROWIDは、CHAR、VARCHAR2、NCHARまたはNVARCHAR2データ型の値をROWIDデータ型に変換します。

このファンクションは、CLOBデータを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用してCLOBを引数として渡すことはできます。

関連項目:

詳細は、データ型の比較規則を参照してください。

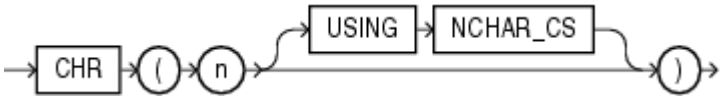
例

次の例では、文字表記のROWIDをROWIDに変換します。(実際のROWIDは、各データベース・インスタンスによって異なります。)

```
SELECT last_name
  FROM employees
  WHERE ROWID = CHARTOROWID('AAAFd1AAFAAAABSAA/');

LAST_NAME
-------------------------
Greene
```

CHR

構文



目的

CHRは、データベース文字セットまたは各国語文字セット(USING NCHAR_CSを指定している場合)の中のnに等しい2進数を持つ文字を、VARCHAR2値として戻します。

シングルバイト文字セットの場合、Oracle Databaseは、n > 256に対してn mod 256に等しい2進数を戻します。マルチバイト文字セットの場合、nは1つのコードポイント全体として解決される必要があります。無効なコードポイントは検証されないため、無効なコードポイントを指定した場合の結果は、予測不能です。

このファンクションは、引数として、NUMBER値、または暗黙的にNUMBER型に変換可能な任意の値を取り、文字を戻します。

ノート:

(オプションの USING NCHAR_CS 句の有無にかかわらず)CHR ファンクションを使用すると、ASCII および EBCDIC ベースのマシン・アーキテクチャ間で移植不可能なコードが戻されます。

関連項目:

- 暗黙的な変換の詳細は、「NCHR」および表2-8を参照してください。
- CHRの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例は、データベース文字セットがWE8ISO8859P1と定義されているASCIIベースのマシンで実行されています。

```
SELECT CHR(67)||CHR(65)||CHR(84) "Dog"
  FROM DUAL;
Dog
---
CAT
```

文字セットがWE8EBCDIC1047のEBCDICベースのマシンでも同じ結果を戻すには、前述の例を次のように修正する必要があります。

```
SELECT CHR(195)||CHR(193)||CHR(227) "Dog"
  FROM DUAL;
Dog
---
CAT
```

マルチバイト文字セットの場合、このように連結しても異なる結果となります。たとえば、a1a2(a1は1つ目のバイト、a2は2つ目のバイト)という16進数の値を持つマルチバイト文字の場合、nに対して「a1a2」に等しい10進数、つまり41378を指定する必要があります。

```
SELECT CHR(41378)
  FROM DUAL;
```

次の例のように、a1に等しい10進数とa2に等しい10進数を連結させて指定することはできません。

```
SELECT CHR(161)||CHR(162)
  FROM DUAL;
```

ただし、次の例のように、マルチバイト文字を連結するマルチバイトのコードポイント全体を連結することは可能です。この例では、a1a2とa1a3の16進数を持つマルチバイト文字を連結しています。

```
SELECT CHR(41378)||CHR(41379)
  FROM DUAL;
```

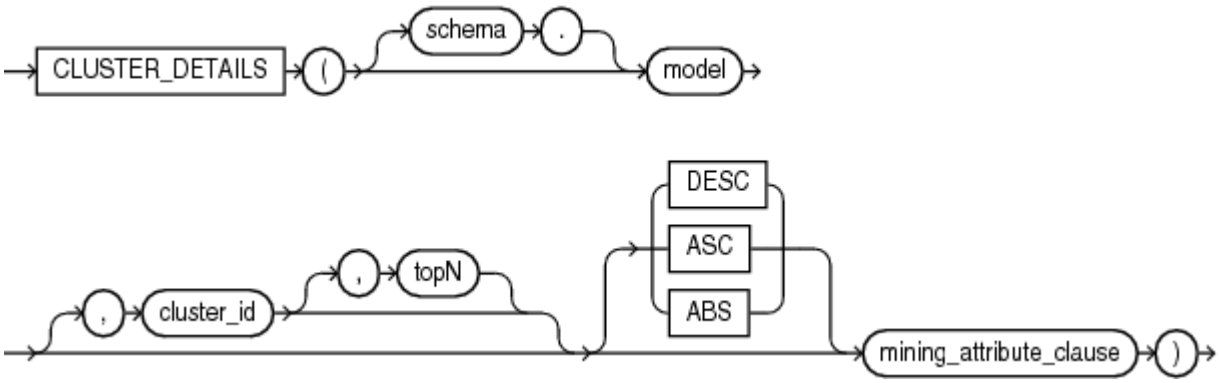
次の例では、各国語文字セットがUTF16であると仮定します。

```
SELECT CHR (196 USING NCHAR_CS)
  FROM DUAL;
CH
--
Ä
```

CLUSTER_DETAILS

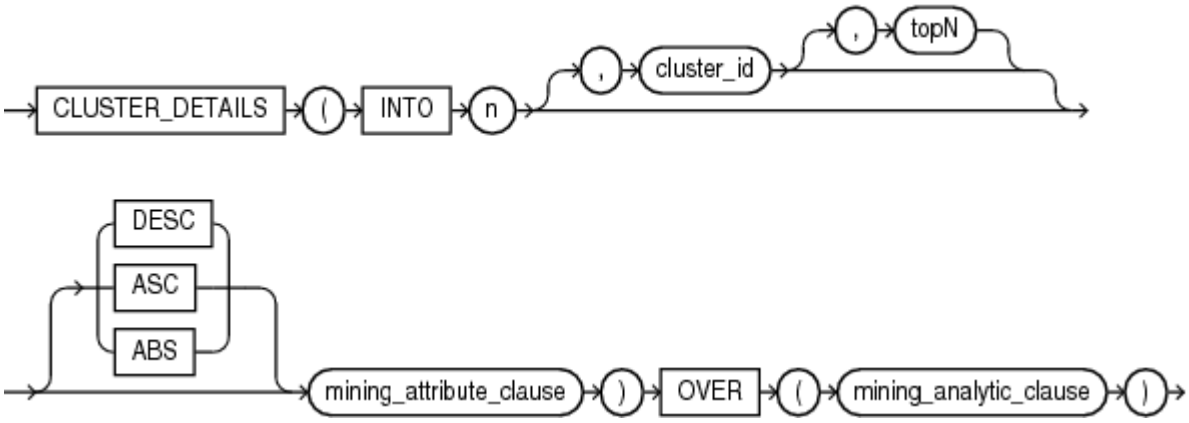
構文

cluster_details::=

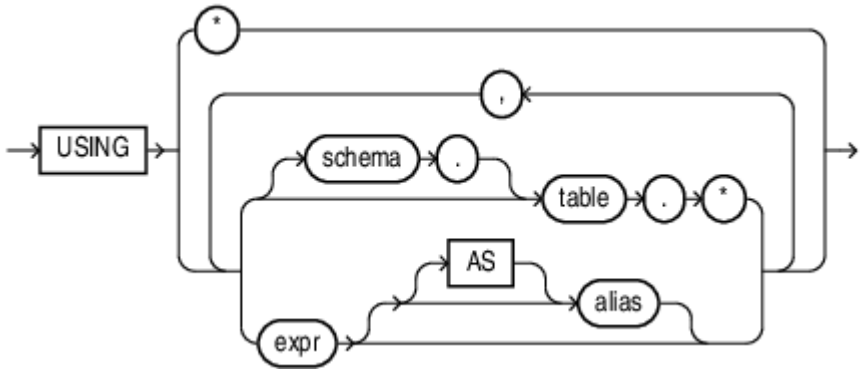


分析構文

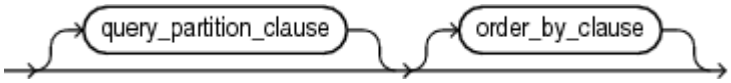
cluster_details_analytic::=



mining_attribute_clause::=



mining_analytic_clause::=



関連項目:

mining_analytic_clauseの構文、セマンティクスおよび制限事項の詳細は、分析ファンクションを参照してください。

目的

CLUSTER_DETAILSは、選択内に含まれる各行のクラスタ詳細を返します。戻り値は、最も高確率のクラスタまたは指定されたcluster_idの属性について記述したXML文字列です。

topN

topNに値を指定すると、このファンクションはクラスタ割当てに最も影響力のあるN個の属性(スコア)を返します。topNを指定しないと、このファンクションは最も影響力のある5つの属性を返します。

DESC、ASC、またはABS

返される属性が重みで順序付けされます。属性の重みは、その属性がクラスタ割当てに与える正の影響または負の影響を表します。正の重みは、割当ての可能性が増加することを示します。負の重みは、割当ての可能性が減少することを示します。

デフォルトでは、CLUSTER_DETAILSは、最大の正の重みを持つ属性を返します(DESC)。ASCを指定すると、最大の負の重みを持つ属性が返されます。ABSを指定すると、正負を問わずに、最大の重みを持つ属性が返されます。結果は、絶対値が高いほうから低いほうに順序付けされています。重みがゼロの属性は、出力に含まれません。

構文の選択

CLUSTER_DETAILSは、2つの方法のどちらかでデータにスコアを付けます。1つ目の方法では、データにマイニング・モデル・オブジェクトを適用します。もう1つの方法では、1つ以上の一時マイニング・モデルを作成してから適用する分析句を実行して、動的にデータをマイニングします。構文または分析構文を選択します。

- 構文 — 事前に定義されたモデルでデータにスコアを付ける場合は、最初の構文を使用します。クラスタリング・モデルの名前を指定します。
- 分析構文 — 事前定義されたモデルなしで、データにスコアを付ける場合は、分析構文を使用します。INTO n (nは、計算するクラスタの数)と、mining_analytic_clause (複数のモデル構築のためにデータをパーティション化する必要がある場合に指定します)を含めます。mining_analytic_clauseは、query_partition_clauseとorder_by_clauseをサポートします。(analytic_clause::=を参照。)

CLUSTER_DETAILSファンクションの構文では、パーティション化されたモデルをスコアリングするときに、オプションのGROUPINGヒントを使用できます。「GROUPINGヒント」を参照してください。

mining_attribute_clause

mining_attribute_clauseは、スコアの予測子として使用する列の属性を特定します。このファンクションが分析構文で起動された場合は、これらの予測子が一時モデルの構築にも使用されます。mining_attribute_clauseは、PREDICTIONファンクションと同様に動作します。(mining_attribute_clause::=を参照。)

関連項目:

- スコアリングの詳細は、Oracle Data Miningユーザーズ・ガイドを参照してください。
- クラスタリングの詳細は、Oracle Data Mining概要を参照してください。

ノート:

次に示す例は、Data Mining のサンプル・プログラムからの抜粋です。サンプル・プログラムの詳細は、Oracle Data Mining ユーザーズ・ガイドの「付録 A」を参照してください。

例

この例では、顧客ID 100955のクラスタ割当てに対して、最大の影響が(20%以上の確率で)ある属性が一覧表示されます。この問合せは、CLUSTER_DETAILSファンクションとCLUSTER_SETファンクションを起動します。これにより、クラスタリング・モデルem_sh_clus_sampleを適用します。

```
SELECT S.cluster_id, probability prob,
       CLUSTER_DETAILS(em_sh_clus_sample, S.cluster_id, 5 USING T.*) det
FROM
  (SELECT v.*, CLUSTER_SET(em_sh_clus_sample, NULL, 0.2 USING *) pset
    FROM mining_data_apply_v v
   WHERE cust_id = 100955) T,
  TABLE(T.pset) S
ORDER BY 2 DESC;

CLUSTER_ID  PROB DET
---------- ----- --------------------------------------------------------------------
-------------
        14 .6761 <Details algorithm="Expectation Maximization" cluster="14">
                 <Attribute name="AGE" actualValue="51" weight=".676" rank="1"/>
                 <Attribute name="HOME_THEATER_PACKAGE" actualValue="1" weight=".557"
rank="2"/>
                 <Attribute name="FLAT_PANEL_MONITOR" actualValue="0" weight=".412"
rank="3"/>
                 <Attribute name="Y_BOX_GAMES" actualValue="0" weight=".171"
rank="4"/>
                 <Attribute name="BOOKKEEPING_APPLICATION" actualValue="1"
weight="-.003"rank="5"/>
                 </Details>

         3 .3227 <Details algorithm="Expectation Maximization" cluster="3">
                 <Attribute name="YRS_RESIDENCE" actualValue="3" weight=".323"
rank="1"/>
                 <Attribute name="BULK_PACK_DISKETTES" actualValue="1" weight=".265"
rank="2"/>
                 <Attribute name="EDUCATION" actualValue="HS-grad" weight=".172"
rank="3"/>
                 <Attribute name="AFFINITY_CARD" actualValue="0" weight=".125"
rank="4"/>
                 <Attribute name="OCCUPATION" actualValue="Crafts" weight=".055"
rank="5"/>
                 </Details>
```

分析の例

この例では、共通の特徴に基づいて、顧客のデータベースを4つのセグメントに分割します。このクラスタリングのファンクションは、事前に定義されたクラスタリング・モデルなしでクラスタを計算してスコアを返します。

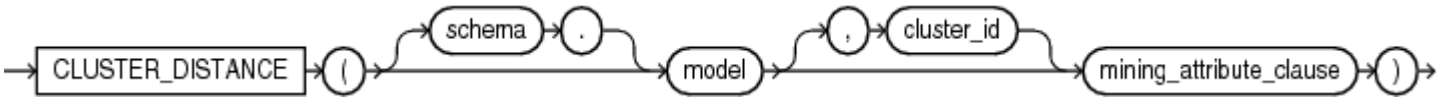
```
SELECT * FROM (
     SELECT cust_id,
          CLUSTER_ID(INTO 4 USING *) OVER () cls,
          CLUSTER_DETAILS(INTO 4 USING *) OVER () cls_details
     FROM mining_data_apply_v)
WHERE cust_id <= 100003
ORDER BY 1;

CUST_ID CLS CLS_DETAILS
------- --- --------------------------------------------------------------------
----------
 100001   5 <Details algorithm="K-Means Clustering" cluster="5">
```

```
            <Attribute name="FLAT_PANEL_MONITOR" actualValue="0" weight=".349"
rank="1"/>
            <Attribute name="BULK_PACK_DISKETTES" actualValue="0" weight=".33"
rank="2"/>
            <Attribute name="CUST_INCOME_LEVEL" actualValue="G: 130¥,000 - 149¥,999"
weight=".291"
             rank="3"/>
            <Attribute name="HOME_THEATER_PACKAGE" actualValue="1" weight=".268"
rank="4"/>
            <Attribute name="Y_BOX_GAMES" actualValue="0" weight=".179" rank="5"/>
            </Details>
 100002   6 <Details algorithm="K-Means Clustering" cluster="6">
            <Attribute name="CUST_GENDER" actualValue="F" weight=".945" rank="1"/>
            <Attribute name="CUST_MARITAL_STATUS" actualValue="NeverM" weight=".856"
rank="2"/>
            <Attribute name="HOUSEHOLD_SIZE" actualValue="2" weight=".468" rank="3"/>
            <Attribute name="AFFINITY_CARD" actualValue="0" weight=".012" rank="4"/>
            <Attribute name="CUST_INCOME_LEVEL" actualValue="L: 300¥,000 and above"
weight=".009"
             rank="5"/>
            </Details>

 100003   7 <Details algorithm="K-Means Clustering" cluster="7">
            <Attribute name="CUST_MARITAL_STATUS" actualValue="NeverM" weight=".862"
rank="1"/>
            <Attribute name="HOUSEHOLD_SIZE" actualValue="2" weight=".423" rank="2"/>
            <Attribute name="HOME_THEATER_PACKAGE" actualValue="0" weight=".113"
rank="3"/>
            <Attribute name="AFFINITY_CARD" actualValue="0" weight=".007" rank="4"/>
            <Attribute name="CUST_ID" actualValue="100003" weight=".006" rank="5"/>
            </Details>
```

CLUSTER_DISTANCE

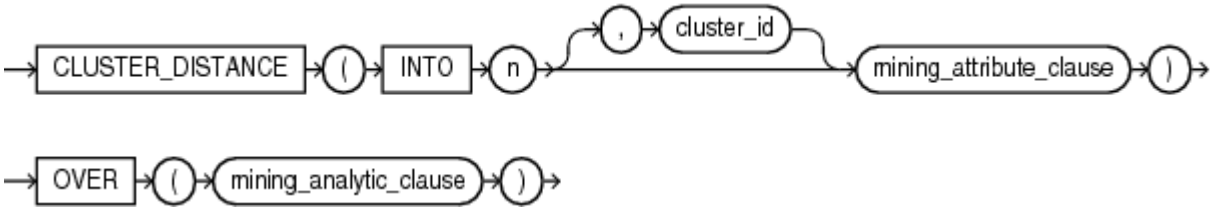
構文

cluster_distance::=

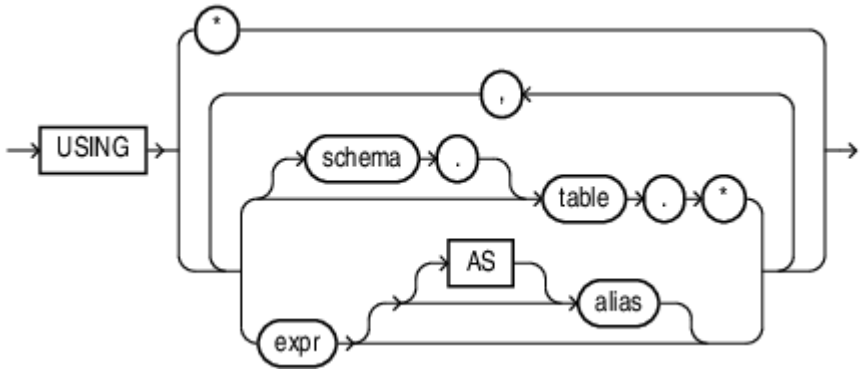


分析構文

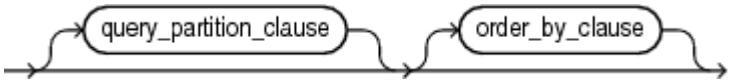
cluster_distance_analytic::=



mining_attribute_clause::=



mining_analytic_clause::=



関連項目:

`mining_analytic_clause`の構文、セマンティクスおよび制限事項の詳細は、分析ファンクションを参照してください。

目的

`CLUSTER_DISTANCE`は、選択内に含まれる各行のクラスタ距離を返します。クラスタ距離は、最も高確率のクラスタまたは指定された`cluster_id`の行と重心との間の距離です。この距離は、`BINARY_DOUBLE`として返されます。

構文の選択

`CLUSTER_DISTANCE`は、2つの方法のどちらかでデータにスコアを付けます。1つ目の方法では、データにマイニング・モデル・オブジェクトを適用します。もう1つの方法では、1つ以上の一時マイニング・モデルを作成してから適用する分析句を実行して、動的にデータをマイニングします。構文または分析構文を選択します。

- 構文 — 事前に定義されたモデルでデータにスコアを付ける場合は、最初の構文を使用します。クラスタリング・モデルの名前を指定します。
- 分析構文 — 事前定義されたモデルなしで、データにスコアを付ける場合は、分析構文を使用します。`INTO n` (nは、計算するクラスタの数)と、`mining_analytic_clause` (複数のモデル構築のためにデータをパーティション化する必要がある場合に指定します)を含めます。`mining_analytic_clause`は、`query_partition_clause`と`order_by_clause`をサポートします。(「analytic_clause::=」を参照。)

`CLUSTER_DISTANCE`ファンクションの構文では、パーティション化されたモデルをスコアリングするときに、オプションの`GROUPING`ヒントを使用できます。「GROUPINGヒント」を参照してください。

mining_attribute_clause

`mining_attribute_clause`は、スコアの予測子として使用する列の属性を特定します。分析構文でファンクションが起動されると、このデータは一時モデルの構築にも使用されます。`mining_attribute_clause`は、`PREDICTION`ファンクションと同様に動作します。(mining_attribute_clause::=を参照。)

関連項目:

- スコアリングの詳細は、Oracle Data Miningユーザーズ・ガイドを参照してください。
- クラスタリングの詳細は、Oracle Data Mining概要を参照してください。

ノート:

次に示す例は、Data Mining のサンプル・プログラムからの抜粋です。サンプル・プログラムの詳細は、Oracle Data Mining ユーザーズ・ガイドの「付録 A」を参照してください。

例

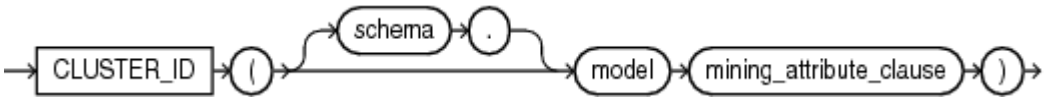
この例では、最近接のクラスタの重心からの距離で測定される、最も特異な10行の行を検出します。

```
SELECT cust_id
  FROM (
    SELECT cust_id,
           rank() over
             (order by CLUSTER_DISTANCE(km_sh_clus_sample USING *) desc) rnk
      FROM mining_data_apply_v)
  WHERE rnk <= 11
  ORDER BY rnk;

   CUST_ID
----------
    100579
    100050
    100329
    100962
    101251
    100179
    100382
    100713
    100629
    100787
    101478
```

CLUSTER_ID

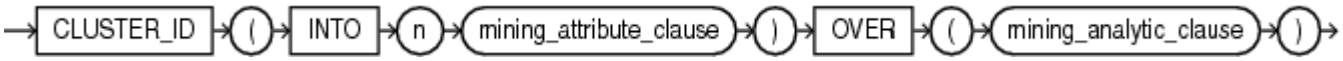
構文

cluster_id::=

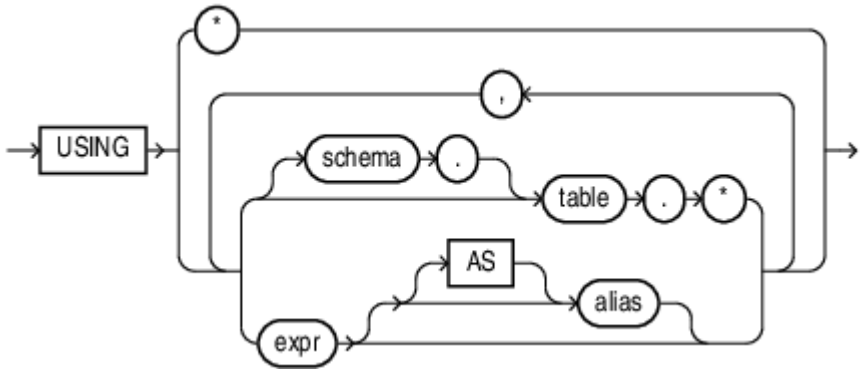


分析構文

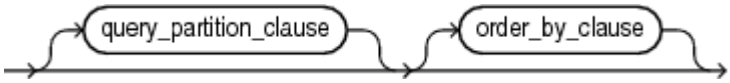
cluster_id_analytic::=



mining_attribute_clause::=



mining_analytic_clause::=



関連項目:

`mining_analytic_clause`の構文、セマンティクスおよび制限事項の詳細は、分析ファンクションを参照してください。

目的

`CLUSTER_ID`は、選択内に含まれる各行の最も高確率のクラスタの識別子(ID)を返します。このクラスタIDは、Oracle `NUMBER`として返されます。

構文の選択

`CLUSTER_ID`は、2つの方法のどちらかでデータにスコアを付けます。1つの方法では、データにマイニング・モデル・オブジェクトを適用します。もう1つの方法では、1つ以上の一時マイニング・モデルを作成して適用する分析句を実行して動的にデータをマイニングします。構文または分析構文を選択します。

- 構文 — 事前に定義されたモデルでデータにスコアを付ける場合は、最初の構文を使用します。クラスタリング・モデルの名前を指定します。

- 分析構文 — 事前定義されたモデルなしで、データにスコアを付ける場合は、分析構文を使用します。INTO n (nは、計算するクラスタの数)と、mining_analytic_clause (複数のモデル構築のためにデータをパーティション化する必要がある場合に指定します)を含めます。mining_analytic_clauseは、query_partition_clauseとorder_by_clauseをサポートします。(「analytic_clause::=」を参照。)

CLUSTER_IDファンクションの構文では、パーティション化されたモデルをスコアリングするときに、オプションのGROUPINGヒントを使用できます。「GROUPINGヒント」を参照してください。

mining_attribute_clause

mining_attribute_clauseは、スコアの予測子として使用する列の属性を特定します。このファンクションが分析構文で起動された場合は、これらの予測子が一時モデルの構築にも使用されます。mining_attribute_clauseは、PREDICTIONファンクションと同様に動作します。(mining_attribute_clause::=を参照。)

関連項目:

- スコアリングの詳細は、Oracle Data Miningユーザーズ・ガイドを参照してください。
- クラスタリングの詳細は、Oracle Data Mining概要を参照してください。

ノート:

次に示す例は、Data Mining のサンプル・プログラムからの抜粋です。サンプル・プログラムの詳細は、Oracle Data Mining ユーザーズ・ガイドの「付録 A」を参照してください。

例

次の例では、mining_data_apply_v内の顧客がグループ化されているクラスタを一覧表示します。

```
SELECT CLUSTER_ID(km_sh_clus_sample USING *) AS clus, COUNT(*) AS cnt
  FROM mining_data_apply_v
  GROUP BY CLUSTER_ID(km_sh_clus_sample USING *)
  ORDER BY cnt DESC;
      CLUS        CNT
---------- ----------
         2        580
        10        216
         6        186
         8        115
        19        110
        12        101
        18         81
        16         39
        17         38
        14         34
```

分析の例

この例では、共通の特徴に基づいて、顧客のデータベースを4つのセグメントに分割します。このクラスタリングのファンクションは、事前に定義されたクラスタリング・モデルなしでクラスタを計算してスコアを返します。

```
SELECT * FROM (
     SELECT cust_id,
          CLUSTER_ID(INTO 4 USING *) OVER () cls,
          CLUSTER_DETAILS(INTO 4 USING *) OVER () cls_details
     FROM mining_data_apply_v)
```

```
WHERE cust_id <= 100003
ORDER BY 1;

CUST_ID CLS CLS_DETAILS
------- --- --------------------------------------------------------------------
----
 100001   5 <Details algorithm="K-Means Clustering" cluster="5">
             <Attribute name="FLAT_PANEL_MONITOR" actualValue="0" weight=".349"
rank="1"/>
             <Attribute name="BULK_PACK_DISKETTES" actualValue="0" weight=".33"
rank="2"/>
             <Attribute name="CUST_INCOME_LEVEL" actualValue="G: 130¥,000 - 149¥,999"
                weight=".291" rank="3"/>
             <Attribute name="HOME_THEATER_PACKAGE" actualValue="1" weight=".268"
rank="4"/>
             <Attribute name="Y_BOX_GAMES" actualValue="0" weight=".179" rank="5"/>
             </Details>
 100002   6 <Details algorithm="K-Means Clustering" cluster="6">
             <Attribute name="CUST_GENDER" actualValue="F" weight=".945" rank="1"/>
             <Attribute name="CUST_MARITAL_STATUS" actualValue="NeverM" weight=".856"
rank="2"/>
             <Attribute name="HOUSEHOLD_SIZE" actualValue="2" weight=".468" rank="3"/>
             <Attribute name="AFFINITY_CARD" actualValue="0" weight=".012" rank="4"/>
             <Attribute name="CUST_INCOME_LEVEL" actualValue="L: 300¥,000 and above"
                weight=".009" rank="5"/>
             </Details>

 100003   7 <Details algorithm="K-Means Clustering" cluster="7">
             <Attribute name="CUST_MARITAL_STATUS" actualValue="NeverM" weight=".862"
rank="1"/>
             <Attribute name="HOUSEHOLD_SIZE" actualValue="2" weight=".423" rank="2"/>
             <Attribute name="HOME_THEATER_PACKAGE" actualValue="0" weight=".113"
rank="3"/>
             <Attribute name="AFFINITY_CARD" actualValue="0" weight=".007" rank="4"/>
             <Attribute name="CUST_ID" actualValue="100003" weight=".006" rank="5"/>
             </Details>
```

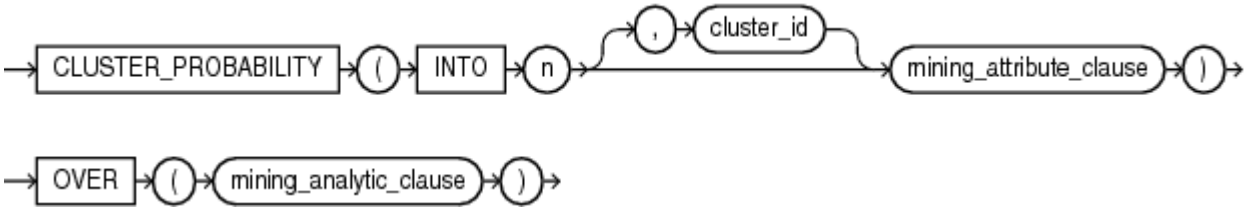
CLUSTER_PROBABILITY

構文

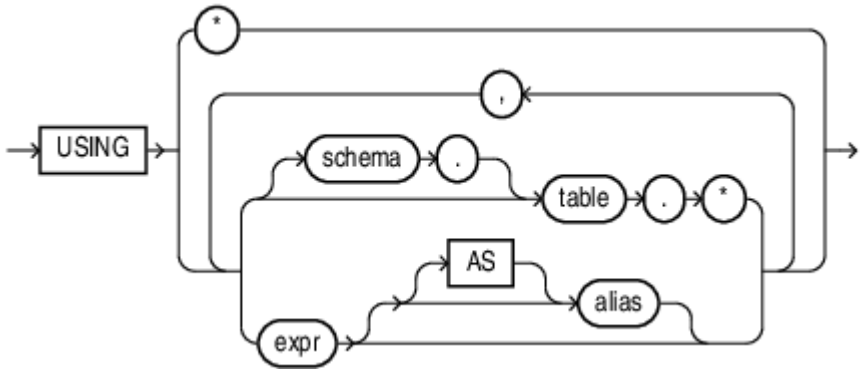
cluster_probability::=

分析構文

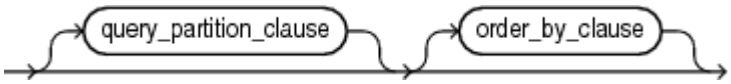
cluster_prob_analytic::=



mining_attribute_clause::=



mining_analytic_clause::=



関連項目:

`mining_analytic_clause`の構文、セマンティクスおよび制限事項の詳細は、分析ファンクションを参照してください。

目的

`CLUSTER_PROBABILITY`は、選択内に含まれる各行の確率を返します。確率は、最も高確率のクラスタまたは指定された`cluster_id`を参照します。クラスタの確率は、`BINARY_DOUBLE`として返されます。

構文の選択

`CLUSTER_PROBABILITY`は、2つの方法のどちらかでデータにスコアを付けます。1つの方法では、データにマイニング・モデル・オブジェクトを適用します。もう1つの方法では、1つ以上の一時マイニング・モデルを作成して適用する分析句を実行して動的にデータをマイニングします。構文または分析構文を選択します。

- 構文 — 事前に定義されたモデルでデータにスコアを付ける場合は、最初の構文を使用します。クラスタリング・モデルの名前を指定します。
- 分析構文 — 事前定義されたモデルなしで、データにスコアを付ける場合は、分析構文を使用します。INTO n (nは、計算するクラスタの数)と、mining_analytic_clause (複数のモデル構築のためにデータをパーティション化する必要がある場合に指定します)を含めます。mining_analytic_clauseは、query_partition_clauseとorder_by_clauseをサポートします。(「analytic_clause::=」を参照。)

CLUSTER_PROBABILITYファンクションの構文では、パーティション化されたモデルをスコアリングするときに、オプションのGROUPINGヒントを使用できます。「GROUPINGヒント」を参照してください。

mining_attribute_clause

mining_attribute_clauseは、スコアの予測子として使用する列の属性を特定します。このファンクションが分析構文で起動された場合は、これらの予測子が一時モデルの構築にも使用されます。mining_attribute_clauseは、PREDICTIONファンクションと同様に動作します。(mining_attribute_clause::=を参照。)

関連項目:

- スコアリングの詳細は、Oracle Data Miningユーザーズ・ガイドを参照してください。
- クラスタリングの詳細は、Oracle Data Mining概要を参照してください。

ノート:

次に示す例は、Data Mining のサンプル・プログラムからの抜粋です。サンプル・プログラムの詳細は、Oracle Data Mining ユーザーズ・ガイドの「付録 A」を参照してください。

例

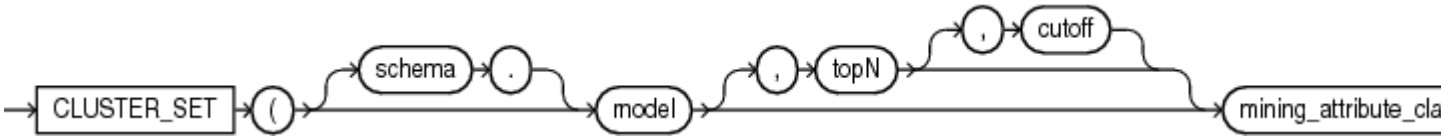
次の例では、可能性に基づいて、クラスタ2で最も代表的な顧客を10人一覧表示します。

```
SELECT cust_id
  FROM (SELECT cust_id, rank() OVER (ORDER BY prob DESC, cust_id) rnk_clus2
    FROM (SELECT cust_id, CLUSTER_PROBABILITY(km_sh_clus_sample, 2 USING *) prob
          FROM mining_data_apply_v))
WHERE rnk_clus2 <= 10
ORDER BY rnk_clus2;

   CUST_ID
----------
    100256
    100988
    100889
    101086
    101215
    100390
    100985
    101026
    100601
    100672
```

CLUSTER_SET

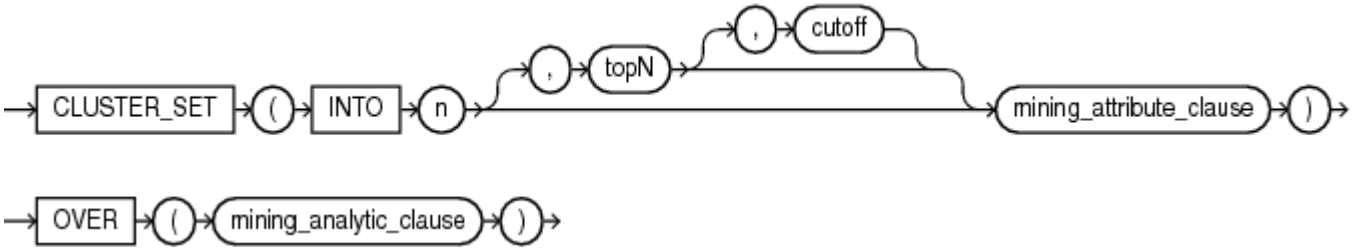
構文

cluster_set::=

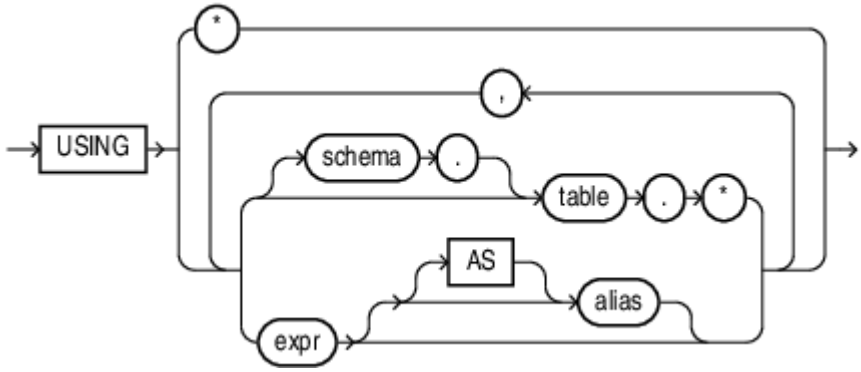


分析構文

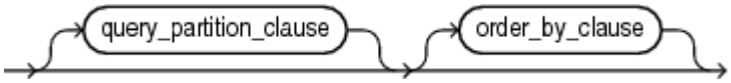
cluster_set_analytic::=



mining_attribute_clause::=



mining_analytic_clause::=



関連項目:

`mining_analytic_clause`の構文、セマンティクスおよび制限事項の詳細は、分析ファンクションを参照してください。

目的

`CLUSTER_SET`は、選択内に含まれる各行のクラスタIDと確率の組で構成されるセットを返します。戻り値は、フィールド名が`CLUSTER_ID`および`PROBABILITY`のオブジェクトのVARRAYです。クラスタIDはOracle `NUMBER`で、確率は`BINARY_DOUBLE`です。

topNおよびcutoff

topNとcutoffを指定すると、このファンクションから返されるクラスタの数を制限できます。デフォルトでは、topNおよびcutoffがnullであり、すべてのクラスタが戻されます。

- topNは、最も可能性の高いN個のクラスタになります。N番目の確率を持つクラスタが複数ある場合でも、ファンクションが選択するのはそのうち1つのみです。
- cutoffは、確率しきい値です。cutoff以上の確率を持つクラスタのみが返されます。cutoffのみでフィルタ処理するには、topNにNULLを指定します。

cutoff以上の最も高確率のクラスタを最大N個まで返すには、topNとcutoffの両方を指定します。

構文の選択

CLUSTER_SETは、2つの方法のどちらかでデータにスコアを付けます。1つの方法では、データにマイニング・モデル・オブジェクトを適用します。もう1つの方法では、1つ以上の一時マイニング・モデルを作成して適用する分析句を実行して動的にデータをマイニングします。構文または分析構文を選択します。

- 構文 — 事前に定義されたモデルでデータにスコアを付ける場合は、最初の構文を使用します。クラスタリング・モデルの名前を指定します。
- 分析構文 — 事前定義されたモデルなしで、データにスコアを付ける場合は、分析構文を使用します。INTO n (nは、計算するクラスタの数)と、mining_analytic_clause (複数のモデル構築のためにデータをパーティション化する必要がある場合に指定します)を含めます。mining_analytic_clauseは、query_partition_clauseとorder_by_clauseをサポートします。(「analytic_clause::=」を参照。)

CLUSTER_SETファンクションの構文では、パーティション化されたモデルをスコアリングするときに、オプションのGROUPINGヒントを使用できます。「GROUPINGヒント」を参照してください。

mining_attribute_clause

mining_attribute_clauseは、スコアの予測子として使用する列の属性を特定します。このファンクションが分析構文で起動された場合は、これらの予測子が一時モデルの構築にも使用されます。mining_attribute_clauseは、PREDICTIONファンクションと同様に動作します。(mining_attribute_clause::=を参照。)

関連項目:

- スコアリングの詳細は、Oracle Data Miningユーザーズ・ガイドを参照してください。
- クラスタリングの詳細は、Oracle Data Mining概要を参照してください。

ノート:

次に示す例は、Data Mining のサンプル・プログラムからの抜粋です。サンプル・プログラムの詳細は、Oracle Data Mining ユーザーズ・ガイドの「付録 A」を参照してください。

例

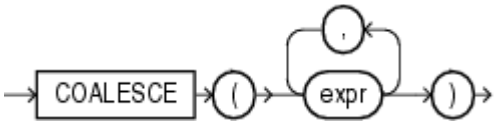
この例では、顧客ID 100955のクラスタ割当てに対して、最大の影響が(20%以上の確率で)ある属性が一覧表示されます。この問合せは、CLUSTER_DETAILSファンクションとCLUSTER_SETファンクションを起動します。これにより、クラスタリング・モデルem_sh_clus_sampleを適用します。

```
SELECT S.cluster_id, probability prob,
```

```
       CLUSTER_DETAILS(em_sh_clus_sample, S.cluster_id, 5 USING T.*) det
FROM
  (SELECT v.*, CLUSTER_SET(em_sh_clus_sample, NULL, 0.2 USING *) pset
    FROM mining_data_apply_v v
   WHERE cust_id = 100955) T,
  TABLE(T.pset) S
ORDER BY 2 DESC;

CLUSTER_ID  PROB DET
---------- ----- --------------------------------------------------------------------
----------
        14 .6761 <Details algorithm="Expectation Maximization" cluster="14">
                 <Attribute name="AGE" actualValue="51" weight=".676" rank="1"/>
                 <Attribute name="HOME_THEATER_PACKAGE" actualValue="1" weight=".557"
rank="2"/>
                 <Attribute name="FLAT_PANEL_MONITOR" actualValue="0" weight=".412"
rank="3"/>
                 <Attribute name="Y_BOX_GAMES" actualValue="0" weight=".171"
rank="4"/>
                 <Attribute name="BOOKKEEPING_APPLICATION" actualValue="1"
weight="-.003"rank="5"/>
                 </Details>

         3 .3227 <Details algorithm="Expectation Maximization" cluster="3">
                 <Attribute name="YRS_RESIDENCE" actualValue="3" weight=".323"
rank="1"/>
                 <Attribute name="BULK_PACK_DISKETTES" actualValue="1" weight=".265"
rank="2"/>
                 <Attribute name="EDUCATION" actualValue="HS-grad" weight=".172"
rank="3"/>
                 <Attribute name="AFFINITY_CARD" actualValue="0" weight=".125"
rank="4"/>
                 <Attribute name="OCCUPATION" actualValue="Crafts" weight=".055"
rank="5"/>
                 </Details>
```

COALESCE

構文



目的

COALESCEは、式のリストの最初のNULLでないexprを戻します。2つ以上の式を指定する必要があります。すべてのexprがNULLと評価された場合、このファンクションはNULLを戻します。

Oracle Databaseでは、短絡評価を使用します。データベースは、NULLかどうかを判断する前にexpr値のすべてを評価するのではなく、各expr値を評価して、NULLかどうかを判断します。

すべてのexprが数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型である場合、Oracle Databaseは、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

関連項目:

- 暗黙的な変換の詳細は、表2-8を参照してください。数値の優先順位の詳細は、数値の優先順位を参照してください。
- COALESCEの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

このファンクションはNVLファンクションを一般化したファンクションです。

COALESCEは、CASE式の変形として使用できます。たとえば、次のようになります。

```
COALESCE(expr1, expr2)
```

これは、次と同等です。

```
CASE WHEN expr1 IS NOT NULL THEN expr1 ELSE expr2 END
```

次も同様です。

```
COALESCE(expr1, expr2, ..., exprn)
```

n >= 3で、次と同じです。

```
CASE WHEN expr1 IS NOT NULL THEN expr1
   ELSE COALESCE (expr2, ..., exprn) END
```

関連項目:

NVLおよびCASE式を参照してください

例

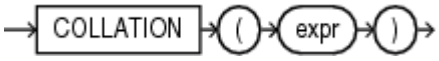
次の例では、サンプル表oe.product_informationを使用して、製品のクリアランス・セールを企画します。製品の表示

価格から10%値引きします。表示価格がない場合は、最小価格はセール価格となります。最小価格がない場合、セール価格は5となります。

```
SELECT product_id, list_price, min_price,
       COALESCE(0.9*list_price, min_price, 5) "Sale"
  FROM product_information
  WHERE supplier_id = 102050
  ORDER BY product_id;
PRODUCT_ID LIST_PRICE  MIN_PRICE       Sale
---------- ---------- ---------- ----------
      1769         48                 43.2
      1770                    73         73
      2378        305        247      274.5
      2382        850        731        765
      3355                                5
```

COLLATION

構文



目的

COLLATIONは、expr用に導出された照合の名前を戻します。このファンクションは、名前付き照合および疑似照合を戻します。導出された照合がUnicode照合アルゴリズム(UCA)に基づく照合である場合、ファンクションは長い形式の名前を戻します。このファンクションは、これを含むSQL文のコンパイル中に評価されます。exprの評価中の照合の競合により、導出された照合が未定義の場合、ファンクションはNULLを戻します。

exprは、CHAR、VARCHAR2、LONG、NCHARまたはNVARCHAR2型の文字列と評価される必要があります。

このファンクションは、VARCHAR2値を返します。

ノート:

COLLATION ファンクションは、NLS_SORT パラメータによって設定される動的な照合ではなく、データ・バインドされた照合のみを戻します。このため、COLLATE USING_NLS_SORT として宣言された列については、ファンクションは、セッション・パラメータ NLS_SORT の実際の値ではなく、文字値'USING_NLS_SORT'を戻します。組込みファンクション SYS_CONTEXT('USERENV','NLS_SORT')を使用すると、セッション・パラメータ NLS_SORT の実際の値を取得できます。

関連項目:

COLLATIONの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

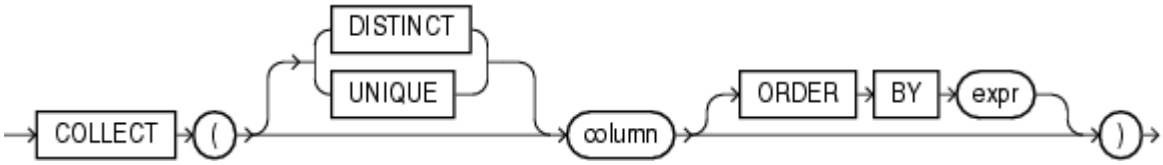
例

次の例では、表id_table内の列nameとidの導出された照合を戻します。

```
CREATE TABLE id_table
  (name VARCHAR2(64) COLLATE BINARY_AI,
   id VARCHAR2(8) COLLATE BINARY_CI);
INSERT INTO id_table VALUES('Christopher', 'ABCD1234');
SELECT COLLATION(name), COLLATION(id)
  FROM id_table;
COLLATION COLLATION
--------- ---------
BINARY_AI BINARY_CI
```

COLLECT

構文



目的

COLLECTは、任意の型の列を引数に取り、選択された行から、入力された型のネストした表を作成する集計ファンクションです。このファンクションから正確な結果を取得するには、このファンクションをCASTファンクション内で使用する必要があります。

column自体がコレクションである場合、COLLECTの出力はコレクションのネストした表になります。columnがユーザー定義型である場合は、オプションのDISTINCT、UNIQUEおよびORDER BY句を使用できるように、columnにMAPまたはORDERメソッドが定義されている必要があります。

関連項目:

- CASTおよび集計ファンクションを参照してください。
- DISTINCT句とORDER BY句の文字値を比較するためにCOLLECTで使用する照合を定義する照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、oe.customersサンプル表の電話番号のVARRAY列からネストした表を作成します。ネストした表には、収入水準がL: 300,000 and aboveである顧客の電話番号のみが含まれます。

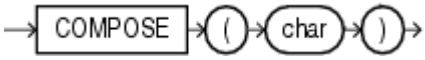
```
CREATE TYPE phone_book_t AS TABLE OF phone_list_typ;
/
SELECT CAST(COLLECT(phone_numbers) AS phone_book_t) "Income Level L Phone Book"
  FROM customers
  WHERE income_level = 'L: 300,000 and above';
Income Level L Phone Book
--------------------------------------------------------------------------------
PHONE_BOOK_T(PHONE_LIST_TYP('+1 414 123 4307'), PHONE_LIST_TYP('+1 608 123 4344'
), PHONE_LIST_TYP('+1 814 123 4696'), PHONE_LIST_TYP('+1 215 123 4721'), PHONE_L
IST_TYP('+1 814 123 4755'), PHONE_LIST_TYP('+91 11 012 4817', '+91 11 083 4817')
, PHONE_LIST_TYP('+91 172 012 4837'), PHONE_LIST_TYP('+41 31 012 3569', '+41 31
083 3569'))
```

次の例では、サンプル表oe.warehousesの倉庫名の列からネストした表を作成します。ORDER BYを使用して、倉庫名を順序付けます。

```
CREATE TYPE warehouse_name_t AS TABLE OF VARCHAR2(35);
/
SELECT CAST(COLLECT(warehouse_name ORDER BY warehouse_name)
       AS warehouse_name_t) "Warehouses"
   FROM warehouses;
Warehouses
--------------------------------------------------------------------------------
WAREHOUSE_NAME_TYP('Beijing', 'Bombay', 'Mexico City', 'New Jersey', 'San Franci
sco', 'Seattle, Washington', 'Southlake, Texas', 'Sydney', 'Toronto')
```

COMPOSE

構文



目的

COMPOSEは引数として文字値charを取り、Unicode規格の定義D117で説明されているように、Unicode正規構成を適用した結果を返します。引数の文字セットがUnicode文字セットのいずれでもない場合、COMPOSEは引数を変更せずに返します。

COMPOSEは、Unicode正規化フォームの文字列を直接返すことはありません。NFC形式の文字列を取得するには、最初にCANONICALを設定してDECOMPOSEをコールし、次にCOMPOSEをコールします。NFKC形式の文字列を取得するには、最初にCOMPATIBILITYを設定してDECOMPOSEをコールし、次にCOMPOSEをコールします。

charには、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のいずれかのデータ型を指定できます。他のデータ型は、VARCHAR2またはNVARCHAR2に暗黙的に変換できる場合に許可されます。COMPOSEの戻り値は、その引数と同じ文字セットで返されます。

暗黙的な変換を使用して、CLOBおよびNCLOBの値がサポートされます。charが文字のLOB値の場合、COMPOSE演算の前にVARCHAR2値に変換されます。特定の実行環境で、LOB値のサイズがVARCHAR2のサポートする長さを超える場合、この演算は失敗します。

関連項目:

- Unicode文字セットおよび文字セマンティクスの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。
- COMPOSEの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。
- DECOMPOSE

例

次の例は、oウムラウトのコードポイントを戻します。

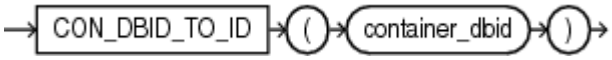
```
SELECT COMPOSE( 'o' || UNISTR('¥0308') )
  FROM DUAL;
CO
--
ö
```

関連項目:

UNISTR

CON_DBID_TO_ID

構文



目的

CON_DBID_TOは、コンテナDBIDを引数に取り、コンテナIDを返します。container_dbidには、NUMBERまたは暗黙的にNUMBERに変換可能な任意の値を指定します。戻り値はNUMBERです。

この関数は、マルチテナント・コンテナ・データベース(CDB)で有益です。CDB以外でこの関数を使用すると、0が返されます。

例

次の問合せでは、CDBのすべてのコンテナのIDおよびDBIDが表示されます。この例での出力例を示します。

```
SELECT CON_ID, DBID
  FROM V$CONTAINERS;
    CON_ID       DBID
---------- ----------
         1 1930093401
         2 4054529501
         4 2256797992
```

次の文は、DBIDが2256797992のコンテナのIDを返します。

```
SELECT CON_DBID_TO_ID(2256797992) "Container ID"
  FROM DUAL;
Container ID
------------
           4
```

CON_GUID_TO_ID

構文

目的

CON_GUID_TO_IDは、コンテナGUID(グローバル一意識別子)を引数に取り、コンテナIDを返します。container_guidには、RAW値を指定します。戻り値はNUMBERです。

この関数は、マルチテナント・コンテナ・データベース(CDB)で有益です。CDB以外でこの関数を使用すると、0が返されます。

例

次の問合せでは、CDBのすべてのコンテナのIDおよびGUIDが表示されます。GUIDは、16バイトのRAW値としてV$CONTAINERSビューに格納されます。この問合せは、32文字の16進表現のGUIDを返します。この例での出力例を示します。

```
SELECT CON_ID, GUID
  FROM V$CONTAINERS;
    CON_ID GUID
---------- --------------------------------
         1 DB0A9F33DF99567FE04305B4F00A667D
         2 D990C280C309591EE04305B4F00A593E
         4 D990F4BD938865C1E04305B4F00ACA18
```

次の文は、GUIDが16進数の値D990F4BD938865C1E04305B4F00ACA18で表現されたコンテナのIDを返します。HEXTORAW関数は、GUIDの16進表現をRAW値に変換します。

```
SELECT CON_GUID_TO_ID(HEXTORAW('D990F4BD938865C1E04305B4F00ACA18')) "Container ID"
  FROM DUAL;
Container ID
------------
           4
```

CON_ID_TO_CON_NAME

構文

目的

CON_ID_TO_CON_NAMEは、引数としてコンテナCON_IDを取り、コンテナNAMEを戻します。

CON_IDには、数値または数値に変換する式を指定する必要があります。戻り値はNUMBERです。

このファンクションは、マルチテナント・コンテナ・データベース(CDB)で有益です。CDB以外でこの関数を使用すると、0が返されます。

例

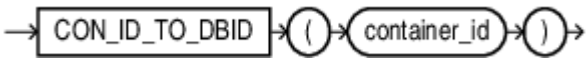
```
SELECT CON_ID, NAME FROM V$CONTAINERS;
    CON_ID          NAME
   ---------        -------------
       1            CDB$ROOT
       2            PDB$SEED
       3            CDB1_PDB1
       4            SALESPDB
```

次の文は、コンテナCON_IDが4のコンテナNAMEを戻します。

```
SELECT CON_ID_TO_CON_NAME(4) "CON_NAME" FROM DUAL;
     CON_NAME
     --------
     SALESDB
```

CON_ID_TO_DBID

構文



目的

CON_ID_TO_DBIDは、引数としてコンテナCON_IDを取り、コンテナDBIDを戻します。CON_IDには、数値または数値に変換する式を指定する必要があります。戻り値はNUMBERです。

このファンクションは、マルチテナント・コンテナ・データベース(CDB)で有益です。CDB以外でこの関数を使用すると、0が返されます。

例

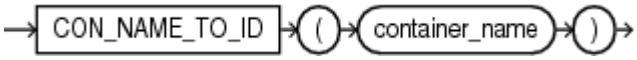
```
SELECT CON_ID, NAME, DBID FROM V$CONTAINERS;
CON_ID       NAME            DBID
-------      ------------    ---------------
   1         CDB$ROOT        2048400776
   2         PDB$SEED        2929762556
   3         CDB1_PDB1       3483444080
   4         SALESPDB        2221053340
```

次の文は、コンテナCON_IDが4のコンテナDBIDを戻します。

```
SELECT CON_ID_TO_DBID(4) FROM DUAL;
    DBID
    -------------
    2221053340
```

CON_NAME_TO_ID

構文



目的

CON_NAME_TO_IDは、コンテナ名を引数に取り、コンテナIDを返します。container_nameには、任意のデータ型の文字列、または文字列に変換する式を指定します。戻り値はNUMBERです。

この関数は、マルチテナント・コンテナ・データベース(CDB)で有益です。CDB以外でこの関数を使用すると、0が返されます。

例

次の問合せでは、CDBのすべてのコンテナのIDおよび名前が表示されます。この例での出力例を示します。

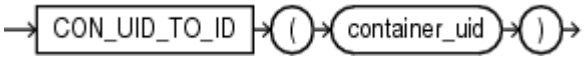
```
SELECT CON_ID, NAME
  FROM V$CONTAINERS;
    CON_ID NAME
---------- ----------
         1 CDB$ROOT
         2 PDB$SEED
         4 SALESPDB
```

次の文は、名前がSALESPDBのコンテナのIDを返します。

```
SELECT CON_NAME_TO_ID('SALESPDB') "Container ID"
  FROM DUAL;
Container ID
------------
           4
```

CON_UID_TO_ID

構文



目的

CON_UID_TO_IDは、コンテナUID(一意識別子)を引数に取り、コンテナIDを返します。container_uidには、NUMBERまたは暗黙的にNUMBERに変換可能な任意の値を指定します。戻り値はNUMBERです。

この関数は、マルチテナント・コンテナ・データベース(CDB)で有益です。CDB以外でこの関数を使用すると、0が返されます。

例

次の問合せでは、CDBのすべてのコンテナのIDおよびUIDが表示されます。この例での出力例を示します。

```
SELECT CON_ID, CON_UID
  FROM V$CONTAINERS;
    CON_ID    CON_UID
---------- ----------
         1          1
         2 4054529501
         4 2256797992
```

次の問合せは、UIDが2256797992のコンテナのIDを返します。

```
SELECT CON_UID_TO_ID(2256797992) "Container ID"
  FROM DUAL;
Container ID
------------
           4
```

CONCAT

構文

目的

CONCATは、char2に連結されているchar1を戻します。char1およびchar2は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBデータ型です。char1と同じ文字セットの文字列が戻されます。そのデータ型は、引数のデータ型によって決まります。

2つの異なるデータ型を連結すると、可逆式変換となるデータ型が戻されます。したがって、引数の1つがLOBの場合、戻り値はLOBとなります。引数の1つが各国語データ型の場合は、戻り値は各国語データ型となります。たとえば:

- CONCAT(CLOB, NCLOB)はNCLOBを戻します。
- CONCAT(NCLOB, NCHAR)はNCLOBを戻します。
- CONCAT(NCLOB, CHAR)はNCLOBを戻します。
- CONCAT(NCHAR, CLOB)はNCLOBを戻します。

この関数は、連結演算子(||)に相当します。

関連項目:

- CONCAT演算子の詳細は、連結演算子を参照してください。
- CONCATの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

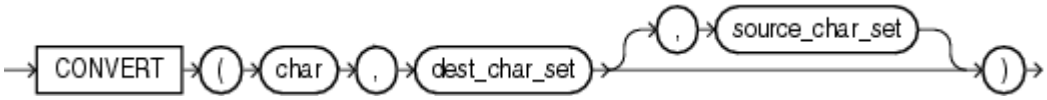
例

次の例では、ネストを使用して3つの文字列を連結します。

```
SELECT CONCAT(CONCAT(last_name, '''s job category is '), job_id) "Job"
  FROM employees
  WHERE employee_id = 152;

Job
-------------------------------------------------------
Hall's job category is SA_REP
```

CONVERT

構文



目的

CONVERTは、文字列を、ある文字セットから別の文字セットに変換します。

- 引数charは変換する値です。charは、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBデータ型です。
- 引数dest_char_setは、charが変換される文字セットの名前です。
- 引数source_char_setは、charをデータベースに格納している文字セットの名前です。デフォルト値はデータベース文字セットです。

CHARとVARCHAR2の戻り値は、VARCHAR2です。NCHARとNVARCHAR2の戻り値は、NVARCHAR2です。CLOBの戻り値はCLOB、NCLOBの戻り値はNCLOBです。

変換先文字セットと変換元文字セットの引数として、リテラルまたは文字セットの名前を含んでいる列を指定できます。

完全に文字を変換するには、変換先文字セットが変換元文字セットで定義されているすべての文字を表現できる必要があります。文字が変換先文字セットに存在しないと、置換文字が使用されます。置換文字は、文字セット定義の一部として定義できます。

ノート:

Oracle Database の現行のリリースでは、CONVERT ファンクションを使用しないことをお薦めします。CONVERT の戻り値は文字データ型であるため、そのデータ型に応じてデータベース文字セットまたは各国語文字セットのいずれかになります。この 2 つの文字セットのどちらでもない dest_char_set は、サポートされません。char 引数と source_char_set の要件は同じです。このため、ファンクションの実用的な用途は、誤った文字セットで格納されているデータの修正にかぎられます。

データベース文字セットと各国語文字セットのどちらでもない値は、RAW または BLOB として処理して格納する必要があります。PL/SQL パッケージ UTL_RAW および UTL_I18N のプロシージャ(たとえば、UTL_RAW.CONVERT)では、このような値の限定的な処理が可能です。パッケージ UTL_FILE、UTL_TCP、UTL_HTTP および UTL_SMTP 内で RAW 引数を受け入れるプロシージャを使用すると、処理されたデータを出力できます。

関連項目:

CONVERTの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、Latin-1文字列をASCIIに変換する文字セットの変換を示します。これは、同じ文字列をWE8ISO8859P1

データベースからUS7ASCIIデータベースへインポートした場合と同じ結果が得られます。

```
SELECT CONVERT('Ä Ê Í Õ Ø A B C D E ', 'US7ASCII', 'WE8ISO8859P1')
   FROM DUAL;
CONVERT('ÄÊÍÕØABCDE'
--------------------
A E I ? ? A B C D E ?
```

次のようにV$NLS_VALID_VALUESビューを問い合せて、有効な文字セットのリストを取得できます。

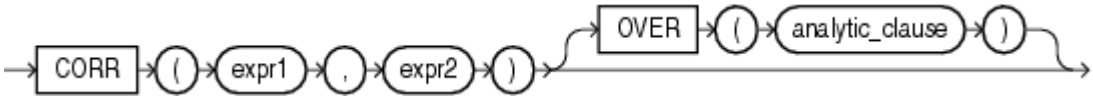
```
SELECT * FROM V$NLS_VALID_VALUES WHERE parameter = 'CHARACTERSET';
```

関連項目:

Oracle Databaseでサポートされている文字セットの一覧は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。V$NLS_VALID_VALUESビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

CORR

構文



関連項目:

構文、セマンティクスおよび制限事項の詳細は、分析ファンクションを参照してください。

目的

CORRは、数値の組の集合に対する相関係数を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。Oracleは、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。数値の優先順位の詳細は、数値の優先順位を参照してください。

Oracle Databaseは、expr1またはexpr2がNULLである組を排除した後、このファンクションを(expr1, expr2)の集合に適用します。その後、Oracleは次の計算を行います。

```
COVAR_POP(expr1, expr2) / (STDDEV_POP(expr1) * STDDEV_POP(expr2))
```

ファンクションは、NUMBER型の値を戻します。ファンクションが空の集合に適用されると、NULLを戻します。

ノート:

CORR ファンクションは、ピアソンの相関係数を計算します。この計算を行うには、数式を引数として指定する必要があります。Oracle は、ノンパラメトリックまたは順位相関をサポートするための CORR_S(スピアマンのロー係数)および CORR_K(ケンドールのタウ b 係数)ファンクションも提供します。

関連項目:

exprの有効な書式の詳細は、集計ファンクションおよびSQL式を参照してください。CORR_SおよびCORR_Kファンクションの詳細は、CORR_*を参照してください。

集計の例

次の例では、oe.product_informationサンプル表の重さクラスごとの製品の表示価格と最小価格の相関係数を計算します。

```
SELECT weight_class, CORR(list_price, min_price) "Correlation"
```

```
  FROM product_information
  GROUP BY weight_class
  ORDER BY weight_class, "Correlation";
WEIGHT_CLASS Correlation
------------ -----------
           1  .999149795
           2  .999022941
           3  .998484472
           4  .999359909
           5  .999536087
```

分析の例

次の例では、会社での勤務年数と給与の相関を従業員の役職別に示します。結果セットでは、指定した業務の従業員ごとに同じ相関を示します。

```
SELECT employee_id, job_id,
       TO_CHAR((SYSDATE - hire_date) YEAR TO MONTH ) "Yrs-Mns",     salary,
       CORR(SYSDATE-hire_date, salary)
       OVER(PARTITION BY job_id) AS "Correlation"
  FROM employees
  WHERE department_id in (50, 80)
  ORDER BY job_id, employee_id;
EMPLOYEE_ID JOB_ID     Yrs-Mns     SALARY Correlation
----------- ---------- ------- ---------- -----------
        145 SA_MAN     +04-09       14000  .912385598
        146 SA_MAN     +04-06       13500  .912385598
        147 SA_MAN     +04-04       12000  .912385598
        148 SA_MAN     +01-08       11000  .912385598
        149 SA_MAN     +01-05       10500  .912385598
        150 SA_REP     +04-05       10000   .80436755
        151 SA_REP     +04-03        9500   .80436755
        152 SA_REP     +03-10        9000   .80436755
        153 SA_REP     +03-03        8000   .80436755
        154 SA_REP     +02-07        7500   .80436755
        155 SA_REP     +01-07        7000   .80436755
. . .
```

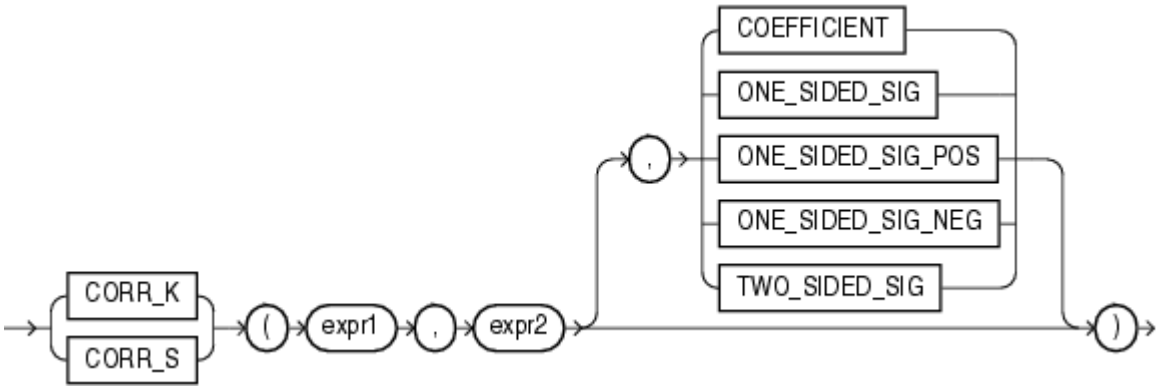
CORR_*

CORR_*ファンクションを次に示します。

- CORR_S
- CORR_K

構文

correlation::=



目的

CORRファンクション(「CORR」を参照)は、ピアソンの相関係数を計算し、入力として数式を必要とします。CORR_*ファンクションは、ノンパラメトリックまたは順位相関をサポートします。これらのファンクションを使用すると、式間の相関を順序尺度化して求めることができます(値の順序付けが可能な場合)。相関係数は-1から1の範囲の値となります。1は完全な正相関、-1は完全な逆相関(一方の変数が減少すると他方の変数が増加する)、および0(ゼロ)に近い値は無相関を表します。

これらのファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。Oracle Databaseは、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、計算を実行してNUMBERを戻します。

関連項目:

- 暗黙的な変換の詳細は、表2-8を参照してください。数値の優先順位の詳細は、数値の優先順位を参照してください。
- expr1からの文字をexpr2からの文字と比較するためにCORR_KおよびCORR_Sで使用する照合を定義する照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

expr1およびexpr2は、分析対象の2つの変数です。3つ目の引数はVARCHAR2型の戻り値です。3つ目の引数を指定しない場合、デフォルトでCOEFFICIENTが戻り値になります。戻り値の意味を次の表に示します。

表7-2 CORR_*の戻り値

戻り値	意味
COEFFICIENT	相関の係数

戻り値	意味
ONE_SIDED_SIG	相関の正の片側有意
ONE_SIDED_SIG_POS	ONE_SIDED_SIG に同じ
ONE_SIDED_SIG_NEG	相関の負の片側有意
TWO_SIDED_SIG	相関の両側有意

CORR_S

CORR_Sは、スピアマンの順位相関係数(ロー)を計算します。入力式は、観測値の組(x_i, y_i)の集合である必要があります。このファンクションは、最初に各値を順位に置き換えます。各x_iの値は、標本内の他のすべてのx_i中での順位に置き換えられ、各y_iの値は、他のすべてのy_i中での順位に置き換えられます。したがって、各x_iおよび各y_iは1からnの値となります。ここでnは、値の組の合計数です。同順位の場合は、値がわずかに異なっていた場合に付けられる順位の平均が割り当てられます。その後このファンクションは、順位の線形相関係数を計算します。

CORR_Sの例

次の例では、スピアマンの順位相関係数(ロー)を使用して、salaryとcommission_pct、salaryとemployee_idの2つの異なる比較ごとに相関係数を導出します。

```
SELECT COUNT(*) count,
       CORR_S(salary, commission_pct) commission,
       CORR_S(salary, employee_id) empid
  FROM employees;

     COUNT COMMISSION      EMPID
---------- ---------- ----------
       107 .735837022 -.04473016
```

CORR_K

CORR_Kは、ケンドールの順位相関係数(タウb)を計算します。CORR_Sと同様に、入力式は観測値の組(x_i, y_i)の集合です。係数を計算するために、このファンクションは一致した組と一致しない組の数をカウントします。xとyの値がいずれも大きい観測値の組は一致しています。xの値が大きくyの値が小さい観測値の組は一致していません。

タウbでの有意性は、タウbによって示される相関が偶然の結果である確率です(0から1の値)。この値が小さい場合、タウbが正の値であれば有意な相関が存在します(タウbが負の値の場合は逆相関)。

CORR_Kの例

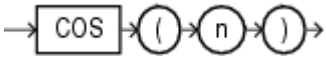
次の例では、ケンドールの順位相関係数(タウb)を使用して、従業員の給与と歩合の割合(パーセント)間に相関があるかどうかを判断します。

```
SELECT CORR_K(salary, commission_pct, 'COEFFICIENT') coefficient,
       CORR_K(salary, commission_pct, 'TWO_SIDED_SIG') two_sided_p_value
  FROM employees;
COEFFICIENT TWO_SIDED_P_VALUE
```

```
----------- -----------------
 .603079768        3.4702E-07
```

COS

構文



目的

COSは、n(ラジアンで表された角度)のコサインを戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数がBINARY_FLOATの場合、このファンクションはBINARY_DOUBLEを戻します。それ以外の場合、引数と同じ数値データ型を戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

例

次の例では、180度のコサインを戻します。

```
SELECT COS(180 * 3.14159265359/180) "Cosine of 180 degrees"
  FROM DUAL;
Cosine of 180 degrees
---------------------
                   -1
```

COSH

構文

目的

COSHは、nの双曲線コサインを戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数がBINARY_FLOATの場合、このファンクションはBINARY_DOUBLEを戻します。それ以外の場合、引数と同じ数値データ型を戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

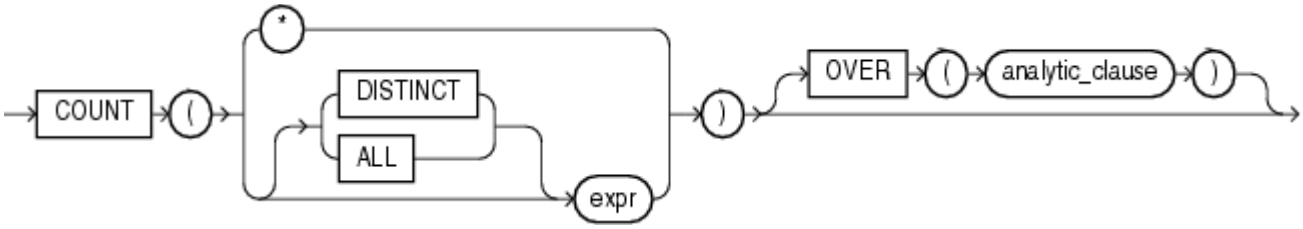
例

次の例では、0の双曲線コサインを戻します。

```
SELECT COSH(0) "Hyperbolic cosine of 0"
  FROM DUAL;
Hyperbolic cosine of 0
----------------------
                     1
```

COUNT

構文



関連項目:

構文、セマンティクスおよび制限事項の詳細は、分析ファンクションを参照してください。

目的

COUNTは、問合せによって戻された行の数を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

DISTINCTを指定する場合は、analytic_clauseのquery_partition_clauseのみを指定できます。order_by_clauseおよびwindowing_clauseは指定できません。

exprを指定すると、COUNTはexprがNULLでない行数を戻します。exprのすべての行を数えるか、または異なる値のみを数えることができます。

アスタリスク(*)を指定すると、このファンクションは重複値およびNULL値を含むすべての行を戻します。COUNTはNULLを戻しません。

ノート:

大量のデータに対して COUNT (DISTINCT expr)操作を実行する前に、次のいずれかのメソッドを使用して、正確な結果よりも速くおおよその結果を取得することを検討します。

- COUNT (DISTINCT expr)ファンクションを使用する前に、APPROX_FOR_COUNT_DISTINCT 初期化パラメータを true に設定します。このパラメータの詳細は、『Oracle Database リファレンス』を参照してください。
- COUNT (DISTINCT expr)ファンクションではなく、APPROX_COUNT_DISTINCT ファンクションを使用します。「APPROX_COUNT_DISTINCT」を参照してください。

関連項目:

- exprの有効な書式の詳細は、SQL式を参照してください。また、集計ファンクションを参照してください。
- DISTINCT句の文字値を比較するためにCOUNTで使用される照合を定義する照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

集計の例

次の例では、COUNTを集計ファンクションとして使用します。

```
SELECT COUNT(*) "Total"
  FROM employees;
     Total
----------
       107
SELECT COUNT(*) "Allstars"
  FROM employees
  WHERE commission_pct > 0;
 Allstars
---------
       35
SELECT COUNT(commission_pct) "Count"
  FROM employees;
     Count
----------
        35
SELECT COUNT(DISTINCT manager_id) "Managers"
  FROM employees;
  Managers
----------
        18
```

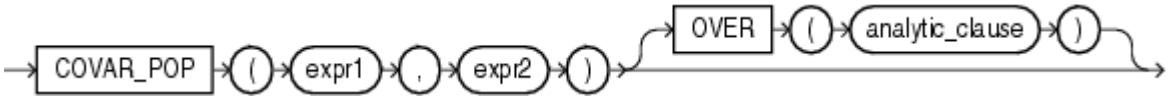
分析の例

次の例では、employees表の各従業員について、その従業員の給与より50ドル少ない金額から150ドル多い金額の範囲の給与を得ている従業員の数を計算します。

```
SELECT last_name, salary,
       COUNT(*) OVER (ORDER BY salary RANGE BETWEEN 50 PRECEDING AND
                      150 FOLLOWING) AS mov_count
  FROM employees
  ORDER BY salary, last_name;
LAST_NAME                      SALARY  MOV_COUNT
-------------------------- ---------- ----------
Olson                            2100          3
Markle                           2200          2
Philtanker                       2200          2
Gee                              2400          8
Landry                           2400          8
Colmenares                       2500         10
Marlow                           2500         10
Patel                            2500         10
. . .
```

COVAR_POP

構文



関連項目:

構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

COVAR_POPは、数値の組の集合に対する母集団共分散を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。Oracleは、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。数値の優先順位の詳細は、数値の優先順位を参照してください。

Oracle Databaseは、expr1またはexpr2がNULLであるすべての組を排除した後、このファンクションを(expr1, expr2)の組の集合に適用します。その後、Oracleは次の計算を行います。

```
(SUM(expr1 * expr2) - SUM(expr2) * SUM(expr1) / n) / n
```

ここで、nは(expr1, expr2)の組の数です(ただし、expr1およびexpr2の両方がNULLではない場合です)。

ファンクションは、NUMBER型の値を戻します。ファンクションが空の集合に適用されると、NULLを戻します。

関連項目:

exprの有効な書式の詳細は、SQL式を参照してください。また、集計ファンクションを参照してください

集計の例

次の例では、サンプル表hr.employeesを使用して、勤務時間(SYSDATE - hire_date)と給与の母集団共分散と標本共分散を計算します。

```
SELECT job_id,
       COVAR_POP(SYSDATE-hire_date, salary) AS covar_pop,
       COVAR_SAMP(SYSDATE-hire_date, salary) AS covar_samp
  FROM employees
  WHERE department_id in (50, 80)
  GROUP BY job_id
  ORDER BY job_id, covar_pop, covar_samp;
JOB_ID       COVAR_POP  COVAR_SAMP
---------- ----------- -----------
SA_MAN          660700      825875
SA_REP      579988.466   600702.34
```

```
SH_CLERK      212432.5  223613.158
ST_CLERK     176577.25  185870.789
ST_MAN          436092      545115
```

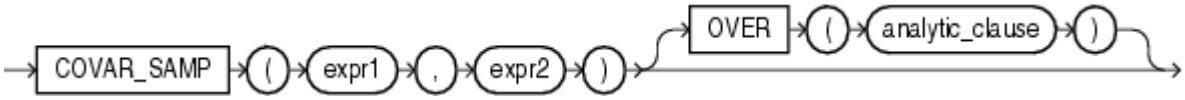
分析の例

次の例では、デモ・スキーマoeの製品の表示価格および最小価格の累積標本共分散を計算します。

```
SELECT product_id, supplier_id,
       COVAR_POP(list_price, min_price)
         OVER (ORDER BY product_id, supplier_id)
         AS CUM_COVP,
       COVAR_SAMP(list_price, min_price)
         OVER (ORDER BY product_id, supplier_id)
         AS CUM_COVS
  FROM product_information p
  WHERE category_id = 29
  ORDER BY product_id, supplier_id;
PRODUCT_ID SUPPLIER_ID   CUM_COVP   CUM_COVS
---------- ----------- ---------- ----------
      1774      103088          0
      1775      103087    1473.25     2946.5
      1794      103096 1702.77778 2554.16667
      1825      103093    1926.25 2568.33333
      2004      103086     1591.4    1989.25
      2005      103086     1512.5       1815
      2416      103088 1475.97959 1721.97619
. . .
```

COVAR_SAMP

構文



関連項目:

構文、セマンティクスおよび制限事項の詳細は、分析ファンクションを参照してください。

目的

COVAR_SAMPは、数値の組の集合の標本共分散を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。Oracleは、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。数値の優先順位の詳細は、数値の優先順位を参照してください。

Oracle Databaseは、`expr1`または`expr2`がNULLであるすべての組を排除した後、このファンクションを(`expr1, expr2`)の組の集合に適用します。その後、Oracleは次の計算を行います。

```
(SUM(expr1 * expr2) - SUM(expr1) * SUM(expr2) / n) / (n-1)
```

ここで、nは(`expr1, expr2`)の組の数です(ただし、`expr1`および`expr2`の両方がNULLではない場合です)。

ファンクションは、NUMBER型の値を戻します。ファンクションが空の集合に適用されると、NULLを戻します。

関連項目:

`expr`の有効な書式の詳細は、SQL式を参照してください。また、集計ファンクションを参照してください

集計の例

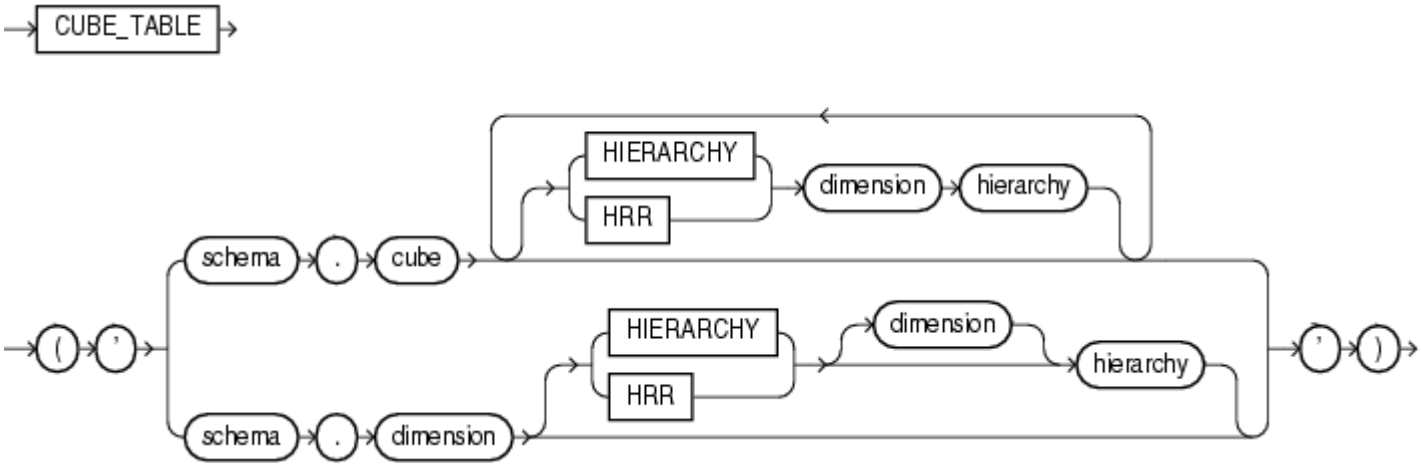
「COVAR_POP」の集計の例を参照してください。

分析の例

「COVAR_POP」の分析の例を参照してください。

CUBE_TABLE

構文



目的

CUBE_TABLEは、キューブまたはディメンションのデータを抽出し、SQLベースのアプリケーションで使用可能な2次元形式のリレーショナル表に戻します。

このファンクションは、1つのVARCHAR2引数を取ります。オプションのHIERARCHY句を使用すると、ディメンション階層を指定できます。キューブでは複数のHIERARCHY句(ディメンションごとに1つ)を使用できます。

次の様々なタイプの表を生成できます。

- キューブ表には、各ディメンションのキー列およびキューブ内の各メジャーと計算されたメジャーの列が含まれています。キューブ表を作成するには、キューブを指定します。キューブのHIERARCHY句は、使用することも使用しないこともできます。複数の階層のあるディメンションの場合は、この句によって戻り値を指定された階層内のディメンション・メンバーとレベルに制限します。HIERARCHY句を使用しない場合は、すべてのディメンション・メンバーとすべてのレベルが含められます。
- ディメンション表には、キー列、および各レベルと各属性の列が含まれています。次のいずれかのコードによりメンバーを識別するMEMBER_TYPE列も含まれます。
    - L - 表、ビュー、またはシノニムからロードされています。
    - A - ロードされたメンバーおよびディメンション内の全階層の単一ルート、つまりすべての集計メンバーです。
    - C - 計算済メンバー

  この表には、すべてのディメンション・メンバーとすべてのレベルが含められます。ディメンション表を作成するには、ディメンションのHIERARCHY句を使用しないでディメンションを指定します。
- 階層表には、ディメンション表のすべての列に加えて、親メンバーの列と各ソース・レベルの列が含まれています。ディメンション表と同様のMEMBER_TYPE列も含まれます。指定された階層に含まれないディメンション・メンバーとレベルはすべて、表から除外されます。階層表を作成する場合は、ディメンションのHIERARCHY句を使用してディメンションを指定します。

CUBE_TABLEは表ファンクションであり、SELECT文のコンテキストで常に次の構文で使用されます。

```
SELECT ... FROM TABLE(CUBE_TABLE('arg'));
```

関連項目:

- ディメンション・オブジェクトの詳細およびCUBE_TABLEで生成される表の詳細は、『Oracle OLAPユーザーズ・ガイド』を参照してください。
- CUBE_TABLEによって生成された表の各文字データ型列に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、OLAPオプションが指定されたOracle DatabaseおよびGLOBALサンプル・スキーマが必要です。GLOBALサンプル・スキーマのダウンロードおよびインストールの詳細は、『Oracle OLAPユーザーズ・ガイド』を参照してください。

次のSELECT文は、GLOBALスキーマのCHANNELのディメンション表を生成します。

```
SELECT dim_key, level_name, long_description, channel_total_id tot_id,
       channel_channel_id chan_id, channel_long_description chan_desc,
       total_long_description tot_desc
  FROM TABLE(CUBE_TABLE('global.channel'));
DIM_KEY      LEVEL_NAME LONG_DESCRIPTION TOT_ID CHAN_ID CHAN_DESC    TOT_DESC
-----------  ---------- ---------------- ------ ------- ------------ -------------
CHANNEL_CAT  CHANNEL    Catalog          TOTAL  CAT     Catalog      Total Channel
CHANNEL_DIR  CHANNEL    Direct Sales     TOTAL  DIR     Direct Sales Total Channel
CHANNEL_INT  CHANNEL    Internet         TOTAL  INT     Internet     Total Channel
TOTAL_TOTAL  TOTAL      Total Channel    TOTAL                       Total Channel
```

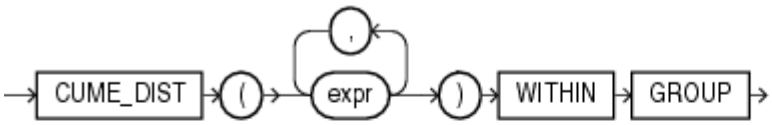
次の文は、UNITS_CUBEのキューブ表を生成します。この文は、表をMARKET階層およびCALENDAR階層に制限します。

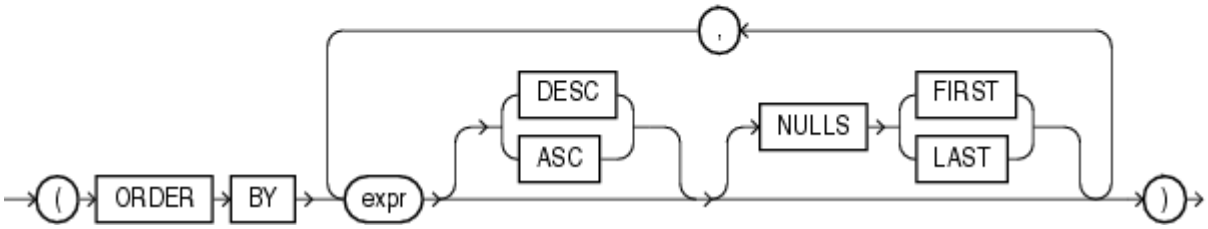
```
SELECT sales, units, cost, time, customer, product, channel
  FROM TABLE(CUBE_TABLE('global.units_cube HIERARCHY customer market HIERARCHY time
calendar'))
  WHERE rownum < 20;
     SALES       UNITS       COST TIME                        CUSTOMER       PRODUCT
CHANNEL
---------- ---------- ---------- ------------------------- ------------- ----------
- -----------
24538587.9      61109 22840853.7 CALENDAR_QUARTER_CY1998.Q1 TOTAL_TOTAL
TOTAL_TOTAL TOTAL_TOTAL
24993273.3      61320   23147171 CALENDAR_QUARTER_CY1998.Q2 TOTAL_TOTAL
TOTAL_TOTAL TOTAL_TOTAL
25080541.4      65265 23242535.4 CALENDAR_QUARTER_CY1998.Q3 TOTAL_TOTAL
TOTAL_TOTAL TOTAL_TOTAL
  26258474      66122 24391020.6 CALENDAR_QUARTER_CY1998.Q4 TOTAL_TOTAL
TOTAL_TOTAL TOTAL_TOTAL
  32785170      77589 30607218.1 CALENDAR_QUARTER_CY1999.Q1 TOTAL_TOTAL
TOTAL_TOTAL TOTAL_TOTAL
. . .
```

CUME_DIST

集計の構文

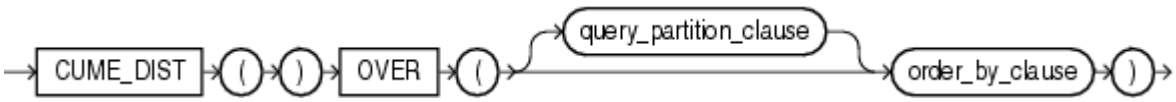
cume_dist_aggregate::=





分析構文

cume_dist_analytic::=



関連項目:

構文、セマンティクスおよび制限事項の詳細は、分析ファンクションを参照してください。

目的

CUME_DISTは、値のグループにある値の累積分布値を計算します。CUME_DISTが戻す値の範囲は、0より大きく1以下です。連結値は、常に同じ累積分布値に対して評価を行います。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。Oracle Databaseは、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、計算を実行してNUMBERを戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。数値の優先順位の詳細は、数値の優先順位を参照してください。

- 集計ファンクションとしてのCUME_DISTは、ファンクションの引数および対応するソート指定によって識別される不確定な行rに対して、集計グループ内の行の中での行rの相対位置を計算します。Oracleは、不確定な行rを集計される行のグループに挿入するように計算します。このファンクションの引数は、各集計グループ内の1つの不確定行を識別します。このため、すべての引数は、集計グループ内で定数式と評価される必要があります。定数引数式および集計のORDER BY句の式の位置は、一致します。このため、引数の数は同じであり、その型は互換性がある必要があります。
- 分析ファンクションとしてのCUME_DISTは、値のグループにある特定の値の相対位置を計算します。行rについて、昇順で順序付けられているとします。rのCUME_DISTは、rの値以下の値の行数を、評価される行の数(問合せ結果

セットまたはパーティション)で割った数です。

集計の例

次の例では、サンプル表oe.employeesの従業員の中から、給与が$15,500であり、歩合が5%の不確定な従業員の累積分布を計算します。

```
SELECT CUME_DIST(15500, .05) WITHIN GROUP
  (ORDER BY salary, commission_pct) "Cume-Dist of 15500"
  FROM employees;
Cume-Dist of 15500
------------------
         .972222222
```

分析の例

次の例では、購買部門の各従業員の給与のパーセンタイルを計算します。たとえば、事務員の40%が、Himuroの給与以下の給与を得ていることがわかります。

```
SELECT job_id, last_name, salary, CUME_DIST()
  OVER (PARTITION BY job_id ORDER BY salary) AS cume_dist
  FROM employees
  WHERE job_id LIKE 'PU%'
  ORDER BY job_id, last_name, salary, cume_dist;
JOB_ID     LAST_NAME                     SALARY  CUME_DIST
---------- ------------------------- ---------- ----------
PU_CLERK   Baida                           2900         .8
PU_CLERK   Colmenares                      2500         .2
PU_CLERK   Himuro                          2600         .4
PU_CLERK   Khoo                            3100          1
PU_CLERK   Tobias                          2800         .6
PU_MAN     Raphaely                       11000          1
```

CURRENT_DATE

構文

目的

CURRENT_DATEは、セッション・タイムゾーンの現在の日付をDATEデータ型のグレゴリオ暦の値で戻します。

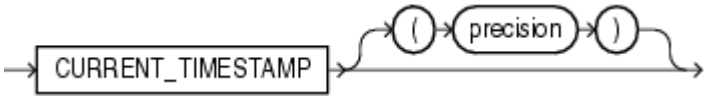
例

次の例では、CURRENT_DATEがセッション・タイムゾーンによって異なることを示します。

```
ALTER SESSION SET TIME_ZONE = '-5:0';
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;
SESSIONTIMEZONE CURRENT_DATE
--------------- --------------------
-05:00          29-MAY-2000 13:14:03
ALTER SESSION SET TIME_ZONE = '-8:0';
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;
SESSIONTIMEZONE CURRENT_DATE
--------------- --------------------
-08:00          29-MAY-2000 10:14:33
```

CURRENT_TIMESTAMP

構文



目的

CURRENT_TIMESTAMPは、セッション・タイムゾーンの現在の日付および時刻をTIMESTAMP WITH TIME ZONEデータ型の値で戻します。タイムゾーン・オフセットは、SQLセッションの現在のローカル時刻を反映します。精度の指定を省略した場合のデフォルトは6です。このファンクションとLOCALTIMESTAMPとの違いは、CURRENT_TIMESTAMPは、TIMESTAMP WITH TIME ZONEの値を戻し、LOCALTIMESTAMPはTIMESTAMPの値を戻す点です。

オプションの引数では、precisionは、戻される時刻の値の小数秒の精度を指定します。

関連項目:

LOCALTIMESTAMP

例

次の例では、CURRENT_TIMESTAMPがセッション・タイムゾーンによって異なることを示します。

```
ALTER SESSION SET TIME_ZONE = '-5:0';
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
SELECT SESSIONTIMEZONE, CURRENT_TIMESTAMP FROM DUAL;
SESSIONTIMEZONE CURRENT_TIMESTAMP
--------------- ---------------------------------------------------
-05:00          04-APR-00 01.17.56.917550 PM -05:00
ALTER SESSION SET TIME_ZONE = '-8:0';
SELECT SESSIONTIMEZONE, CURRENT_TIMESTAMP FROM DUAL;
SESSIONTIMEZONE CURRENT_TIMESTAMP
--------------- ---------------------------------------------------
-08:00          04-APR-00 10.18.21.366065 AM -08:00
```

CURRENT_TIMESTAMPで書式マスクを使用する場合は、ファンクションが戻す値と書式マスクを一致させてください。たとえば、次の表の場合を考えます。

```
CREATE TABLE current_test (col1 TIMESTAMP WITH TIME ZONE);
```

ファンクションが戻す型のTIME ZONEの部分がマスクに含まれていないため、次の文は正常に実行されません。

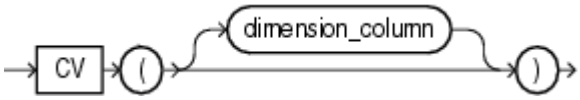
```
INSERT INTO current_test VALUES
  (TO_TIMESTAMP_TZ(CURRENT_TIMESTAMP, 'DD-MON-RR HH.MI.SSXFF PM'));
```

次の文では、CURRENT_TIMESTAMPの戻り値の型と一致する正しい書式マスクが使用されています。

```
INSERT INTO current_test VALUES
  (TO_TIMESTAMP_TZ(CURRENT_TIMESTAMP, 'DD-MON-RR HH.MI.SSXFF PM TZH:TZM'));
```

CV

構文



目的

CVファンクションは、SELECT文のmodel_clauseでのみ、かつモデル・ルールの右側でのみ使用できます。戻り値は、ルールの左側から右側に送られたディメンション列またはパーティション列の現在の値です。このファンクションをmodel_clauseで使用するとディメンション列に対する相対索引を作成できます。戻り型は、ディメンション列のデータ型です。引数を指定しない場合、セル参照内のファンクションの相対位置に対応付けられたディメンション列がデフォルトで使用されます。

CVファンクションは、セル参照外でも使用できます。その場合は、dimension_columnが必要です。

関連項目:

- 構文およびセマンティクスの詳細は、model_clauseおよびモデル式を参照してください。
- CVの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、ディメンション列(マウス・パッドまたはスタンダード・マウス)の1999年および2000年の現在の値が表す製品の売上の合計を、その製品の2001年の売上に割り当てます。

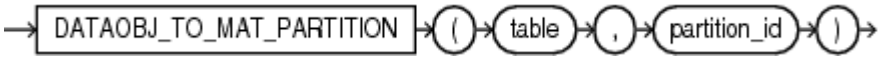
```
SELECT country, prod, year, s
  FROM sales_view_ref
  MODEL
    PARTITION BY (country)
    DIMENSION BY (prod, year)
    MEASURES (sale s)
    IGNORE NAV
    UNIQUE DIMENSION
    RULES UPSERT SEQUENTIAL ORDER
    (
      s[FOR prod IN ('Mouse Pad', 'Standard Mouse'), 2001] =
        s[CV( ), 1999] + s[CV( ), 2000]
    )
  ORDER BY country, prod, year;
COUNTRY       PROD                                      YEAR          S
----------    ----------------------------------      --------   ---------
France        Mouse Pad                                   1998     2509.42
France        Mouse Pad                                   1999     3678.69
France        Mouse Pad                                   2000     3000.72
France        Mouse Pad                                   2001     6679.41
France        Standard Mouse                              1998     2390.83
France        Standard Mouse                              1999     2280.45
France        Standard Mouse                              2000     1274.31
France        Standard Mouse                              2001     3554.76
Germany       Mouse Pad                                   1998     5827.87
Germany       Mouse Pad                                   1999     8346.44
Germany       Mouse Pad                                   2000     7375.46
Germany       Mouse Pad                                   2001     15721.9
Germany       Standard Mouse                              1998     7116.11
Germany       Standard Mouse                              1999     6263.14
```

```
Germany        Standard Mouse                                  2000     2637.31
Germany        Standard Mouse                                  2001     8900.45

16 rows selected.
```

この例では、ビューsales_view_refが必要です。このビューを作成する方法は、MODEL句：例を参照してください。

DATAOBJ_TO_MAT_PARTITION

構文



目的

DATAOBJ_TO_MAT_PARTITIONは、ドメイン索引データの格納に使用するシステム・パーティション表でデータ・メンテナンスまたは問合せ操作を実行するデータ・カートリッジ開発者にのみ役立ちます。DML操作や問合せ操作は、ドメイン索引の実表での対応する操作によってトリガーされます。

このファンクションは、引数として実表の名前と実表のパーティションのパーティションIDを取ります。これらはともに、適切なODCIIndexメソッドによってファンクションに渡されます。ファンクションは、対応するシステム・パーティション表のマテリアライズド・パーティション番号を戻します。この番号は、システム・パーティション表の該当パーティションで操作(DMLまたは問合せ)を実行する際に使用できます。

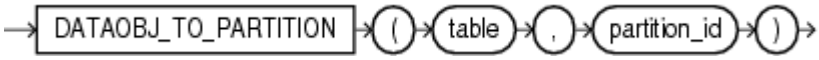
実表が時間隔パーティション表の場合、DATAOBJ_TO_PARTITION関数のかわりにこの関数を使用することをお薦めします。DATAOBJ_TO_PARTITION関数は、物理識別子を使用して絶対パーティション番号を判断します。ただし、実表が時間隔パーティション表の場合、対応する非マテリアライズド・パーティションのパーティション番号にホールがある可能性があります。これは、システム・パーティション表がマテリアライズド・パーティションのみを持ち、実表のパーティションと基礎となるシステム・パーティション索引の記憶域表のパーティションとの間で、DATAOBJ_TO_PARTITION番号で不一致が発生する場合があるためです。DATAOBJ_TO_MAT_PARTITION関数は、マテリアライズド・パーティション番号(絶対パーティション番号ではなく)を返し、2つの表が同期し続けるのを助けます。この関数を使用するには、時間隔パーティション表上のローカル・ドメイン索引をサポートする予定の索引型を移行する必要があります。

関連項目:

- DATAOBJ_TO_PARTITION
- DATAOBJ_TO_MAT_PARTITIONファンクションの使用の詳細および例は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください

DATAOBJ_TO_PARTITION

構文



目的

DATAOBJ_TO_PARTITIONは、ドメイン索引データの格納に使用するシステム・パーティション表でデータ・メンテナンスまたは問合せ操作を実行するデータ・カートリッジ開発者にのみ役立ちます。DML操作や問合せ操作は、ドメイン索引の実表での対応する操作によってトリガーされます。

このファンクションは、引数として実表の名前と実表のパーティションのパーティションIDを取ります。これらはともに、適切なODCIIndexメソッドによってファンクションに渡されます。ファンクションは、対応するシステム・パーティション表の絶対パーティション番号を戻します。この番号は、システム・パーティション表の該当パーティションで操作(DMLまたは問合せ)を実行する際に使用できます。

ノート:

実表が時間隔パーティション表の場合、かわりに DATAOBJ_TO_MAT_PARTITION 関数を使用することをお薦めします。詳細は、DATAOBJ_TO_MAT_PARTITION を参照してください。

関連項目:

DATAOBJ_TO_PARTITIONファンクションの使用の詳細および例は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください

DBTIMEZONE

構文

目的

DBTIMEZONEは、データベースのタイムゾーンの値を戻します。戻り型は、タイムゾーン・オフセット('[+|-]TZH:TZM'という書式の文字列型)またはタイムゾーン地域名です。これは、最近のCREATE DATABASEまたはALTER DATABASE文でユーザーが指定したデータベース・タイムゾーンの値によって異なります。

関連項目:

DBTIMEZONEの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

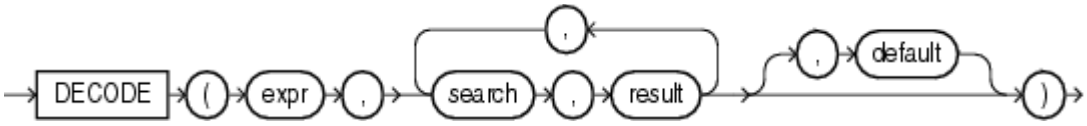
例

次の例では、データベース・タイムゾーンがUTCタイムゾーンに設定されていると想定します。

```
SELECT DBTIMEZONE
  FROM DUAL;
DBTIME
------
+00:00
```

DECODE

構文



目的

DECODEは、exprを各search値と1つずつ比較します。exprがsearchと等しい場合、Oracle Databaseは対応するresultを戻します。一致する値が見つからない場合は、defaultを戻します。defaultが省略されている場合は、NULLを戻します。

引数は、任意の数値型(NUMBER、BINARY_FLOAT、BINARY_DOUBLE)または文字列型です。

- exprおよびsearchが文字データである場合、Oracleは、非空白埋め比較セマンティクスを使用してそれらを比較します。expr、searchおよびresultのデータ型は、CHAR、VARCHAR2、NCHARまたはNVARCHAR2です。戻される文字列は、VARCHAR2データ型で、最初のresultパラメータと同じ文字セットの文字列です。
- 最初のsearch-resultの組が数値である場合、Oracleはすべてのsearch-resultの式と最初のexprを比較して、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

search、resultおよびdefaultの各値は式から派生できます。Oracle Databaseでは、短絡評価を使用します。データベースは、search値のいずれかとexprを比較する前にすべてのsearch値を評価するのではなく、各search値とexprを比較する前にのみ、各search値を評価します。その結果、exprと等しいsearchが見つかると、Oracleはその後のsearchを評価しません。

比較する前に、Oracleはexprと各search値を、最初のsearch値のデータ型に自動的に変換します。Oracleは、戻り値を最初のresultと同じデータ型に自動的に変換します。最初のresultのデータ型がCHARの場合、または最初のresultがNULLの場合、Oracleは戻り値をVARCHAR2データ型の値に変換します。

DECODEファンクションでは、Oracleは2つのNULLを同等とみなします。exprがNULLの場合、Oracleは最初のsearch値のresultもNULLとして戻します。

DECODEファンクションのコンポーネントの最大数は、expr、search、result、defaultを含めて255です。

関連項目:

- 比較セマンティクスについては、データ型の比較規則を参照してください。
- データ型変換の概要は、データ変換を参照してください。
- 浮動小数点の比較セマンティクスの詳細は、浮動小数点数を参照してください。
- 暗黙的な変換のデメリットの詳細は、暗黙的なデータ変換と明示的なデータ変換を参照してください。
- COALESCEおよびCASE式を参照してください。これらは、DECODEと同様の機能を提供します。
- expr式の文字をsearchの文字と比較するためにDECODEで使用する照合を定義する照合決定ルール、およびこのファンクションの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle

Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

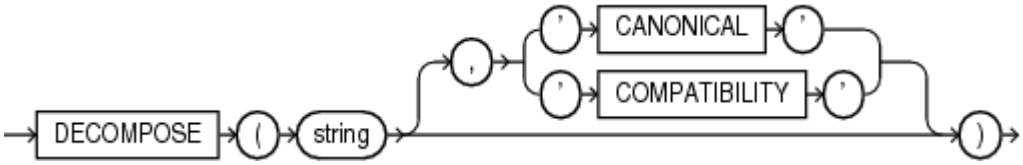
例

この例では、warehouse_idの値をデコードします。warehouse_idが1の場合「Southlake」を、warehouse_idが2の場合は「San Francisco」を戻します。warehouse_idが1、2、3、4のいずれでもない場合、ファンクションは「Non domestic」を戻します。

```
SELECT product_id,
       DECODE (warehouse_id, 1, 'Southlake',
                             2, 'San Francisco',
                             3, 'New Jersey',
                             4, 'Seattle',
                                'Non domestic') "Location"
  FROM inventories
  WHERE product_id < 1775
  ORDER BY product_id, "Location";
```

DECOMPOSE

構文



目的

DECOMPOSEは、1番目の引数として文字値stringを取り、Unicode分解の1つをこの値に適用した結果を返します。適用する分解は、2番目のオプション・パラメータによって決まります。1番目の引数の文字セットがUnicode文字セットのいずれでもない場合、DECOMPOSEは引数を変更せずに返します。

DECOMPOSEの2番目の引数が文字列CANONICAL (大/小文字区別なし)である場合、DECOMPOSEは、Unicode規格の定義D68で説明されているように正規分解を適用し、NFD正規化フォームで文字列を返します。2番目の引数が文字列COMPATIBILITYである場合、DECOMPOSEは、Unicode規格の定義D65で説明されているように互換性分解を適用し、NFKD正規化フォームで文字列を返します。デフォルトの動作では、正規分解が適用されます。

ペシミスティック・ケースでは、DECOMPOSEの戻り値がstringより数倍長くなる場合があります。特定の実行時環境において、返される文字列がVARCHAR2値の最大長より長い場合、値は警告なしにVARCHAR2の最大長に切り捨てられます。

DECOMPOSEの両方の引数には、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のいずれかのデータ型を指定できます。他のデータ型は、VARCHAR2またはNVARCHAR2に暗黙的に変換できる場合に許可されます。DECOMPOSEの戻り値は、その1番目の引数と同じ文字セットで返されます。

暗黙的な変換を使用して、CLOBおよびNCLOBの値がサポートされます。stringが文字のLOB値の場合、DECOMPOSE演算の前にVARCHAR2値に変換されます。特定の実行環境で、LOB値のサイズがVARCHAR2のサポートする長さを超える場合、この演算は失敗します。

関連項目:

- Unicode文字セットおよび文字セマンティクスの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。
- DECOMPOSEの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。
- COMPOSE

例

次の例では、文字列「Châteaux」をその要素のコードポイントに分解します。

```
SELECT DECOMPOSE ('Châteaux')
  FROM DUAL;
DECOMPOSE
---------
Châteaux
```

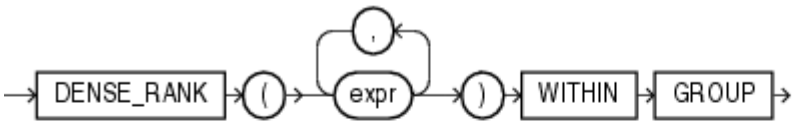
ノート:

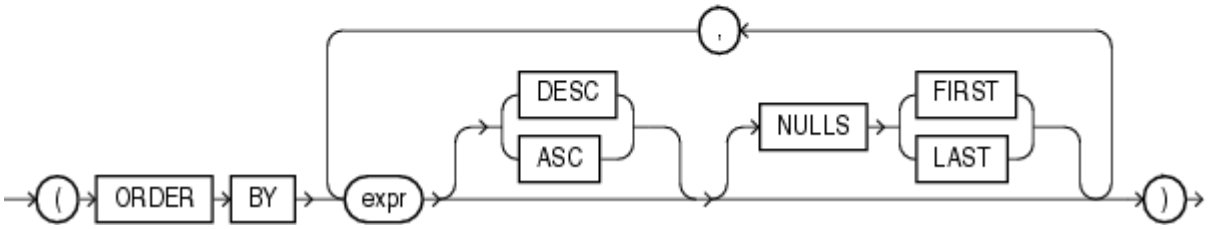
この例の結果は、ご使用のオペレーティング・システムの文字セットによって異なる場合があります。

DENSE_RANK

集計の構文

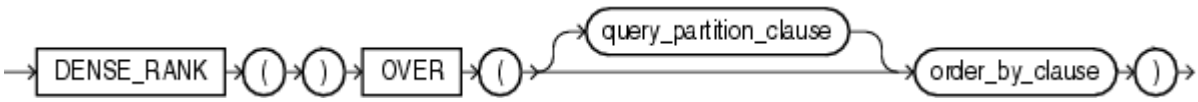
dense_rank_aggregate::=





分析構文

dense_rank_analytic::=



関連項目:

構文、セマンティクスおよび制限事項の詳細は、分析ファンクションを参照してください。

目的

DENSE_RANKは、順序付けされた行のグループ内の行のランクを計算し、そのランクをNUMBERとして戻します。ランクは1から始まる連続した整数です。ランクの最大値は、問合せが戻す一意の数値です。同ランクの場合、ランクの値はスキップされません。そのランク付け基準に関して同じ値を持つ行は、同じランクになります。このファンクションは、上位N番および下位N番のレポートに有効です。

このファンクションは、引数に任意の数値データ型を受け入れ、NUMBERを戻します。

- 集計ファンクションとして使用するDENSE_RANKは、ファンクションの引数によって識別される不確定な行の稠密ランクを、与えられたソート指定で計算します。ファンクションの引数は、各集計グループの単一行を識別するため、すべての引数は各集計グループ内で定数式に評価される必要があります。定数引数式および集計のorder_by_clauseの式の位置は、一致します。このため、引数の数は同じであり、型は互換性がある必要があります。
- 分析ファンクションとして使用するDENSE_RANKは、他の行について、問合せで戻される各行のランクを計算します。この計算は、order_by_clauseにあるvalue_exprsの値に基づいて行われます。

関連項目:

ORDER BY句の文字値を比較するためにDENSE_RANKで使用する照合を定義する照合決定ルールの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

集計の例

次の例では、サンプル表oe.employeesから、給与が$15,500であり、歩合が5%の仮想の従業員のランクを計算します。

```
SELECT DENSE_RANK(15500, .05) WITHIN GROUP
  (ORDER BY salary DESC, commission_pct) "Dense Rank"
  FROM employees;
Dense Rank
----------
         3
```

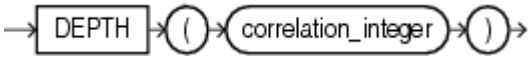
分析の例

次の文は、サンプル・スキーマhrの部門60の従業員を、その給与に基づいてランク付けします。給与の値が等しい従業員のランクは同じになります。ただし、ランクの値がスキップされることはありません。この例と、「RANK」の分析の例を比較してください。

```
SELECT department_id, last_name, salary,
       DENSE_RANK() OVER (PARTITION BY department_id ORDER BY salary) DENSE_RANK
  FROM employees WHERE department_id = 60
  ORDER BY DENSE_RANK, last_name;

DEPARTMENT_ID LAST_NAME                     SALARY DENSE_RANK
------------- ------------------------- ---------- ----------
           60 Lorentz                         4200          1
           60 Austin                          4800          2
           60 Pataballa                       4800          2
           60 Ernst                           6000          3
           60 Hunold                          9000          4
```

DEPTH

構文



目的

DEPTHは、UNDER_PATHおよびEQUALS_PATH条件でのみ使用される補助ファンクションです。このファンクションは、同じ相関変数を持つUNDER_PATH条件によって指定されるパスのレベル数を戻します。

correlation_integerは任意のNUMBER整数です。文に複数の一次条件が含まれている場合に、この補助ファンクションをその一次条件と関連付けるために使用します。1未満の値は1として扱われます。

関連項目:

「EQUALS_PATH条件」および「UNDER_PATH条件」を参照してください。関連するファンクションについては、「PATH」を参照してください。

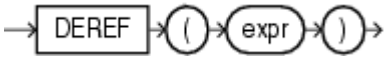
例

EQUALS_PATHおよびUNDER_PATH条件は、2つの補助ファンクションDEPTHおよびPATHを取ることができます。次に、その2つの補助ファンクションの使用方法を示します。この例では、XMLスキーマであるwarehouses.xsd (SQL文でのXMLの使用方法で作成)が存在することを前提としています。

```
SELECT PATH(1), DEPTH(2)
  FROM RESOURCE_VIEW
  WHERE UNDER_PATH(res, '/sys/schemas/OE', 1)=1
    AND UNDER_PATH(res, '/sys/schemas/OE', 2)=1;
PATH(1)                          DEPTH(2)
-------------------------------- --------
. . .
www.example.com                         1
www.example.com/xwarehouses.xsd         2
. . .
```

DEREF

構文



目的

DEREFは、引数exprのオブジェクト参照を戻します。この場合、exprはオブジェクトにREFを戻す必要があります。問合せでこのファンクションを使用しない場合、次の例で示すとおり、かわりにREFのオブジェクトIDを戻します。

関連項目:

MAKE_REF

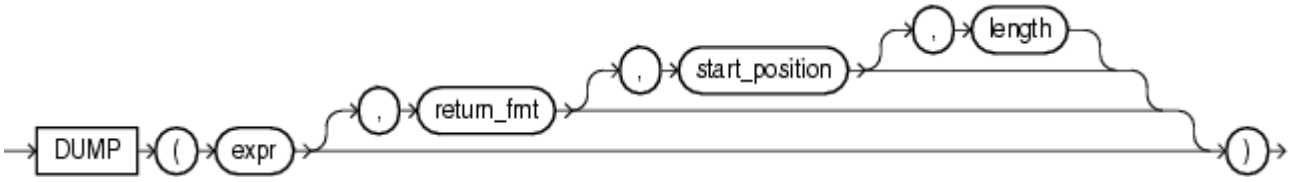
例

サンプル・スキーマoeには、cust_address_typというオブジェクト型が含まれます。「REF制約の例」では、類似する型cust_address_typ_new、およびその型へのREFである1つの列を含む表を作成します。次の例では、その列に挿入を行う方法、およびDEREFを使用して列から情報を抽出する方法を示します。

```
INSERT INTO address_table VALUES
  ('1 First', 'G45 EU8', 'Paris', 'CA', 'US');
INSERT INTO customer_addresses
  SELECT 999, REF(a) FROM address_table a;
SELECT address
  FROM customer_addresses
  ORDER BY address;
ADDRESS
--------------------------------------------------------------------------------
000022020876B2245DBE325C5FE03400400B40DCB176B2245DBE305C5FE03400400B40DCB1
SELECT DEREF(address)
  FROM customer_addresses;
DEREF(ADDRESS)(STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE, COUNTRY_ID)
--------------------------------------------------------------------------------
CUST_ADDRESS_TYP_NEW('1 First', 'G45 EU8', 'Paris', 'CA', 'US')
```

DUMP

構文



目的

DUMPは、exprのデータ型コード、長さ(バイト単位)および内部表現を含むVARCHAR2値を戻します。戻される結果は、常にデータベース文字セットの文字です。各コードに対応するデータ型については、表2-1を参照してください。

引数return_fmtには戻り値の書式として、次の値のいずれかを指定します。

- 8は、結果を8進表記で戻します。
- 10は、結果を10進表記で戻します。
- 16は、結果を16進表記で戻します。
- 17は、各バイトがコンパイラの文字セット内の出力可能な文字(通常はASCIIまたはEBCDIC)として解釈される場合にのみ、文字として出力された各バイトを戻します。一部のASCII制御文字は、^X形式で出力される場合もあります。それ以外の場合、文字は16進表記で出力されます。NLSパラメータはすべて無視されます。return_fmt 17を指定したDUMPの場合、特定の出力書式には依存しないでください。

デフォルトでは、戻り値に文字セット情報が含まれません。exprの文字セット名を取り出すには、前述の書式のいずれかの値に1000を加えます。たとえば、return_fmtに1008を指定すると、8進表記で結果が戻り、さらにexprの文字セット名が得られます。

引数start_positionとlengthを組み合せて、内部表現の戻す部分を指定します。デフォルトでは、10進表記で全体の内部表現が戻されます。

exprがNULLの場合はNULLを戻します。

このファンクションは、CLOBデータを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用してCLOBを引数として渡すことはできます。

関連項目:

- 詳細は、データ型の比較規則を参照してください
- DUMPの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、文字列式および列からダンプ情報を抽出する方法を示します。

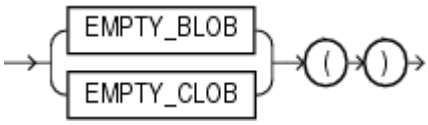
```
SELECT DUMP('abc', 1016)
  FROM DUAL;
DUMP('ABC',1016)
----------------------------------------
Typ=96 Len=3 CharacterSet=WE8DEC: 61,62,63
```

```
SELECT DUMP(last_name, 8, 3, 2) "OCTAL"
  FROM employees
  WHERE last_name = 'Hunold'
  ORDER BY employee_id;
OCTAL
--------------------------------------------------------------------
Typ=1 Len=6: 156,157
SELECT DUMP(last_name, 10, 3, 2) "ASCII"
  FROM employees
  WHERE last_name = 'Hunold'
  ORDER BY employee_id;
ASCII
--------------------------------------------------------------------
Typ=1 Len=6: 110,111
```

EMPTY_BLOB、EMPTY_CLOB

構文

empty_LOB::=



目的

EMPTY_BLOBおよびEMPTY_CLOBは、LOB変数を初期化したり、INSERTまたはUPDATE文でLOB列または属性をEMPTYに初期化できる空のLOBロケータを戻します。EMPTYとは、LOBは初期化されていても、データが移入されていない状態をいいます。

ノート:

空の LOB と、NULL の LOB は、同じではありません。また、空の CLOB と、長さが 0(ゼロ)の文字列を含む LOB は、同じではありません。詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』を参照してください。

関連項目:

EMPTY_CLOBの戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

LOBロケータの制限事項

このファンクションから戻されたロケータは、DBMS_LOBパッケージまたはOCIへのパラメータとして使用できません。

例

次の例では、サンプル表pm.print_mediaのad_photo列をEMPTYに初期化します。

```
UPDATE print_media
  SET ad_photo = EMPTY_BLOB();
```

EXISTSNODE

ノート:

EXISTSNODE ファンクションは非推奨です。これは、下位互換性を保つためにのみサポートされています。ただし、かわりに XMLEXISTS ファンクションを使用することをお薦めします。詳細は、「XMLEXISTS」を参照してください。

構文

目的

EXISTSNODEは、指定されたパスを使用するXML文書のトラバースが任意のノードとなるかどうかを決定します。このファンクションは、XML文書を含むXMLTypeインスタンスとパスを指定するVARCHAR2のXPath文字列を、引数として受け取ります。オプションのnamespace_stringは、接頭辞のデフォルト・マッピングまたはネームスペース・マッピング(Oracle DatabaseがXPath式を評価する場合に使用)を指定するVARCHAR2値に解決される必要があります。

namespace_string引数のデフォルトは、ルート要素のネームスペースになります。Xpath_stringのサブ要素を参照する場合は、namespace_stringを指定する必要があります。また、これらの引数の両方にwho接頭辞を指定する必要があります。

関連項目:

namespace_stringの指定例およびwho接頭辞の使用例は、SQL文でのXMLの使用方法を参照してください。

戻り値は、NUMBERです。

- ドキュメントにXPathトラバースを適用した後にノードが存在しない場合は、0(ゼロ)が戻ります。
- ノードが存在する場合は、1が戻ります。

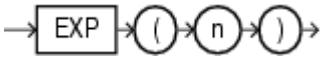
例

次の例では、サンプル表oe.warehousesのwarehouse_spec列のXMLパスに/Warehouse/Dockノードが存在するかを判断します。

```
SELECT warehouse_id, warehouse_name
  FROM warehouses
  WHERE EXISTSNODE(warehouse_spec, '/Warehouse/Docks') = 1
  ORDER BY warehouse_id;
WAREHOUSE_ID WAREHOUSE_NAME
------------ ----------------------------------
           1 Southlake, Texas
           2 San Francisco
           4 Seattle, Washington
```

EXP

構文



目的

EXPは、eのn乗を戻します(e = 2.71828183)。このファンクションは、引数と同じ型の値を戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数がBINARY_FLOATの場合、このファンクションはBINARY_DOUBLEを戻します。それ以外の場合、引数と同じ数値データ型を戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

例

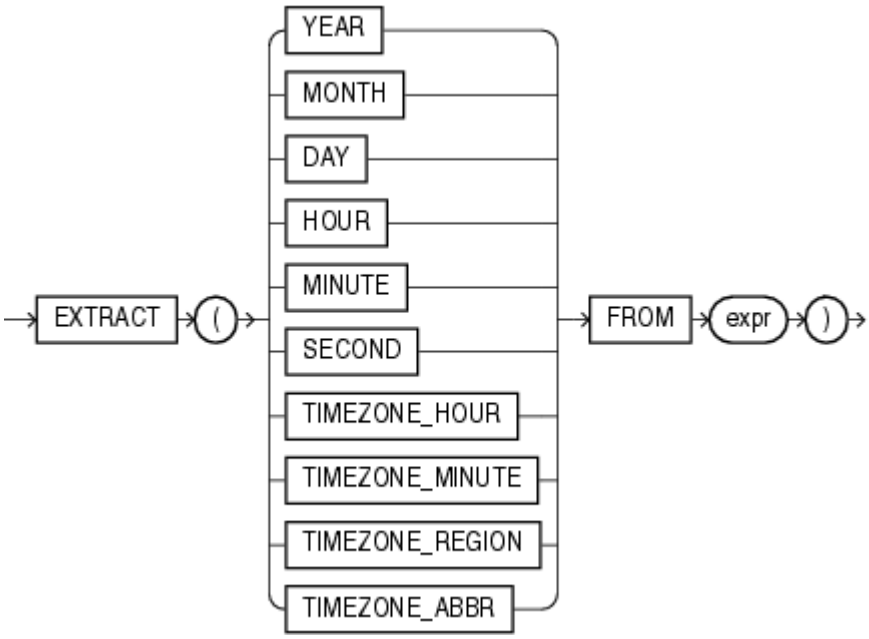
次の例では、eを4乗した値を戻します。

```
SELECT EXP(4) "e to the 4th power"
  FROM DUAL;
e to the 4th power
------------------
          54.59815
```

EXTRACT (日時)

構文

extract_datetime::=



目的

EXTRACTは、日時式または期間式から、指定された日時フィールドの値を抽出して戻します。exprには、要求されたフィールドと互換性のある日時または期間データ型に評価される任意の式を指定できます。

- YEARまたはMONTHが要求された場合、exprはデータ型DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONE、TIMESTAMP WITH LOCAL TIME ZONEまたはINTERVAL YEAR TO MONTHの式に評価される必要があります。
- DAYが要求された場合、exprはデータ型DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONE、TIMESTAMP WITH LOCAL TIME ZONEまたはINTERVAL DAY TO SECONDの式に評価される必要があります。
- HOUR、MINUTEまたはSECONDが要求された場合、exprはデータ型TIMESTAMP、TIMESTAMP WITH TIME ZONE、TIMESTAMP WITH LOCAL TIME ZONEまたはINTERVAL DAY TO SECONDの式に評価される必要があります。DATEは、Oracle Databaseによって時刻フィールドを持たないANSI DATEデータ型として処理されるため、ここでは有効ではありません。
- TIMEZONE_HOUR、TIMEZONE_MINUTE、TIMEZONE_ABBR、TIMEZONE_REGIONまたはTIMEZONE_OFFSETが要求された場合、exprはデータ型TIMESTAMP WITH TIME ZONEまたはTIMESTAMP WITH LOCAL TIME ZONEの式に評価される必要があります。

EXTRACTは、exprをANSI日時データ型として解釈します。たとえば、EXTRACTはDATEをレガシーOracle DATEではなく、時刻要素を持たないANSI DATEとして処理します。したがって、DATE値からは、YEAR、MONTHおよびDAYのみを抽出できます。同様に、TIMESTAMP WITH TIME ZONEデータ型からは、TIMEZONE_HOURおよびTIMEZONE_MINUTEのみを抽出できます。

TIMEZONE_REGIONまたはTIMEZONE_ABBR(略称)を指定する場合、戻り値は、適切なタイムゾーン地域名または略称を含むVARCHAR2文字列です。その他の日時フィールドを指定する場合、戻り値は、グレゴリオ暦での日時値を表すNUMBER

データ型の整数値です。タイムゾーン値を持つ日時から抽出する場合は、UTC形式の値が返されます。有効なタイムゾーン地域名およびそれに対する略称を表示するには、V$TIMEZONE_NAMES動的パフォーマンス・ビューに問合せを実行してください。

このファンクションは、次に示す最初の例のように、非常に大規模な表の日時フィールド値を操作する場合に特に有効です。

ノート:

夏時間機能には、タイムゾーン地域名が必要です。この名前は、大小 2 つのタイムゾーン・ファイルに格納されます。これらのファイルのうち、使用する環境および使用する Oracle Database のリリースに応じて、いずれか一方がデフォルトのファイルになります。タイムゾーン・ファイルおよびタイムゾーン名の詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

日時フィールドと日時または期間値の式を組み合せると、あいまいな結果になる場合があります。この場合、Oracle Databaseは、UNKNOWNを戻します(詳細は、次の例を参照してください)。

関連項目:

- 両方のファイル内のすべてのタイムゾーン地域名のリストは、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。
- EXTRACTの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。
- datetime_value_exprおよびinterval_value_exprの詳細は、日時および期間の演算を参照してください。
- 動的パフォーマンス・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

例

次の例では、oe.orders表から、各月の注文数を戻します。

```
SELECT EXTRACT(month FROM order_date) "Month", COUNT(order_date) "No. of Orders"
  FROM orders
  GROUP BY EXTRACT(month FROM order_date)
  ORDER BY "No. of Orders" DESC, "Month";
     Month No. of Orders
---------- -------------
        11            15
         6            14
         7            14
         3            11
         5            10
         2             9
         9             9
         8             7
        10             6
         1             5
        12             4
         4             1

12 rows selected.
```

次の例では、1998年を戻します。

```
SELECT EXTRACT(YEAR FROM DATE '1998-03-07')
  FROM DUAL;
EXTRACT(YEARFROMDATE'1998-03-07')
---------------------------------
                             1998
```

次の例では、サンプル表hr.employeesから、2007以降に雇用されたすべての従業員を選択します。

```
SELECT last_name, employee_id, hire_date
  FROM employees
  WHERE EXTRACT(YEAR FROM (hire_date, 'DD-MON-RR')) > 2007
  ORDER BY hire_date;
LAST_NAME                 EMPLOYEE_ID HIRE_DATE
------------------------- ----------- ---------
Johnson                           179 04-JAN-08
Grant                             199 13-JAN-08
Marvins                           164 24-JAN-08
. . .
```

次の例では、結果があいまいになるため、OracleはUNKNOWNを戻します。

```
SELECT EXTRACT(TIMEZONE_REGION FROM TIMESTAMP '1999-01-01 10:00:00 -08:00')
  FROM DUAL;
EXTRACT(TIMEZONE_REGIONFROMTIMESTAMP'1999-01-0110:00:00-08:00')
---------------------------------------------------------------
UNKNOWN
```

タイムゾーン数値オフセットが式に指定され、その数値オフセットが複数のタイムゾーン地域名をマップする場合、結果があいまいになります。

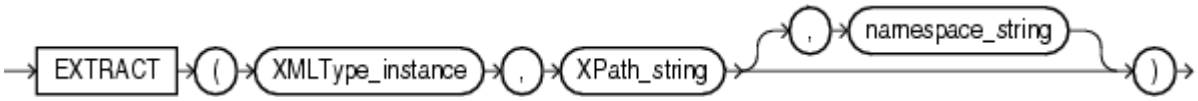
EXTRACT (XML)

ノート:

EXTRACT(XML)ファンクションは非推奨です。これは、下位互換性を保つためにのみサポートされています。ただし、かわりに XMLQUERY ファンクションを使用することをお薦めします。詳細は、「XMLQUERY」を参照してください。

構文

extract_xml::=



目的

EXTRACT(XML)は、EXISTSNODEファンクションに似ています。このファンクションは、VARCHAR2のXPath文字列に適用され、XMLの断片を含むXMLTypeインスタンスを戻します。先頭にスラッシュを付けて絶対XPath_stringを指定したり、先頭のスラッシュを省略して相対XPath_stringを指定できます。先頭のスラッシュを省略した場合、相対パスのコンテキストは、デフォルトでルート・ノードに設定されます。処理しているXMLでネームスペース接頭辞が使用される場合、オプションのnamespace_stringが必要です。この引数は、接頭辞のデフォルト・マッピングまたはネームスペース・マッピング(Oracle DatabaseがXPath式を評価する場合に使用)を指定するVARCHAR2値に解決される必要があります。

例

次の例では、サンプル表oe.warehousesのwarehouse_spec列のXMLパスの/Warehouse/Dockノードの値を抽出します。

```
SELECT warehouse_name,
       EXTRACT(warehouse_spec, '/Warehouse/Docks') "Number of Docks"
  FROM warehouses
  WHERE warehouse_spec IS NOT NULL
  ORDER BY warehouse_name;
WAREHOUSE_NAME            Number of Docks
------------------------- -------------------------
New Jersey
San Francisco             <Docks>1</Docks>
Seattle, Washington       <Docks>3</Docks>
Southlake, Texas          <Docks>2</Docks>
```

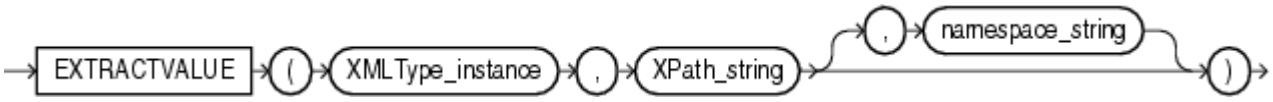
この例と、XMLのフラグメントのスカラー値を戻す、「EXTRACTVALUE」の例を比較してみてください。

EXTRACTVALUE

ノート:

EXTRACTVALUE ファンクションは非推奨です。これは、下位互換性を保つためにのみサポートされています。ただし、かわりに、XMLTABLE ファンクション、または XMLCAST および XMLQUERY ファンクションを使用することをお薦めします。詳細は、「XMLTABLE」、「XMLCAST」および「XMLQUERY」を参照してください。

構文



EXTRACTVALUEは、引数としてXMLTypeインスタンスとXPath式を取り、結果として得られるノードのスカラー値を戻します。結果はシングルノードであり、テキスト・ノード、属性または要素のいずれかである必要があります。結果が要素である場合、その要素はシングル・テキスト・ノードを子ノードとして持つ必要があり、このファンクションが戻す値は、その子ノードの値になります。先頭にスラッシュを付けて絶対XPath_stringを指定したり、先頭のスラッシュを省略して相対XPath_stringを指定できます。先頭のスラッシュを省略した場合、相対パスのコンテキストは、デフォルトでルート・ノードに設定されます。

指定されたXPathが複数の子ノードを持つノードを指す場合、または指されたノードが非テキスト・ノードの子を持つ場合は、エラーが戻されます。オプションのnamespace_stringは、接頭辞のデフォルト・マッピングまたはネームスペース・マッピング(OracleがXPath式を評価する場合に使用)を指定するVARCHAR2値に解決される必要があります。

XMLスキーマに基づくドキュメントで戻り値の型が推測できる場合は、適切な型のスカラー値が戻ります。その他の場合は、VARCHAR2型の結果が戻ります。XMLスキーマに基づかないドキュメントの場合、戻り値は常にVARCHAR2です。

関連項目:

EXTRACTVALUEの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

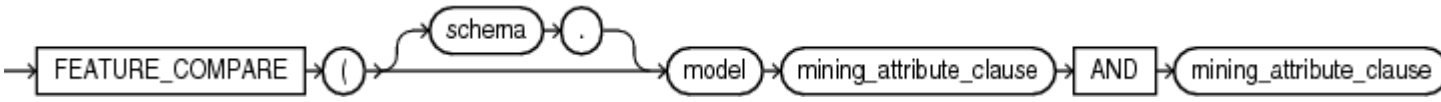
次の例では、入力として、「EXTRACT(XML)」の例と同じ引数を取ります。このファンクションは、EXTRACTファンクションとは異なり、XMLのフラグメントを戻すのではなく、XMLのフラグメントのスカラー値を戻します。

```
SELECT warehouse_name, EXTRACTVALUE(e.warehouse_spec, '/Warehouse/Docks') "Docks"
  FROM warehouses e
  WHERE warehouse_spec IS NOT NULL
  ORDER BY warehouse_name;
WAREHOUSE_NAME       Docks
-------------------- ------------
New Jersey
San Francisco        1
Seattle, Washington  3
Southlake, Texas     2
```

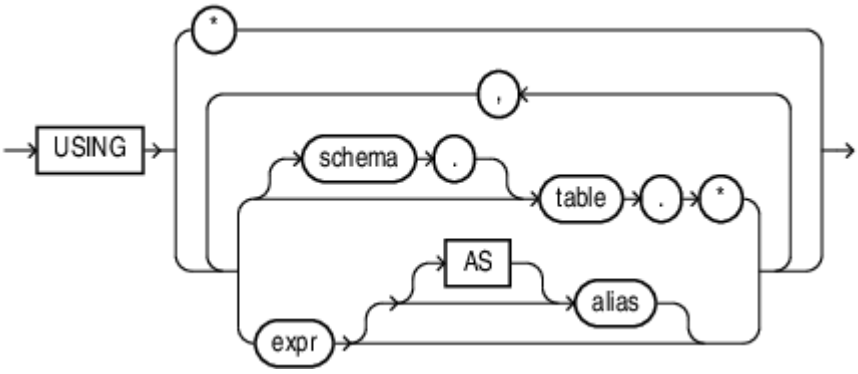
FEATURE_COMPARE

構文

feature_compare::=



mining_attribute_clause::=



目的

FEATURE_COMPAREファンクションでは、特徴抽出モデルを使用して、キーワード・フレーズなどの短いものや2つの属性リストを含む、2つの異なるドキュメントを比較して似ているかどうかを判別します。FEATURE_COMPAREファンクションは、特異値分解(SVD)、主成分分析(PCA)、Non-Negative Matrix Factorization (NMF)、明示的セマンティクス分析(ESA)などの特徴抽出アルゴリズムとともに使用できます。このファンクションは、ドキュメントだけでなく、数値データおよびカテゴリ・データにも適用できます。

FEATURE_COMPAREファンクションへの入力は、Oracle Data MiningのNMF、SVD、ESAなどの特徴抽出アルゴリズムを使用して構築された単一の機能モデルです。double USING句によって、モデルで抽出された特徴を使用して、2つの異なるドキュメントまたは一定のキーワード・フレーズ、あるいはその2つの組合せを比較して似ているかどうかを判別するメカニズムが提供されます。

FEATURE_COMPAREファンクションの構文では、パーティション化されたモデルをスコアリングするときに、オプションのGROUPINGヒントを使用できます。「GROUPINGヒント」を参照してください。

mining_attribute_clause

mining_attribute_clauseは、スコアの予測子として使用する列の属性を特定します。このファンクションが分析構文で起動された場合は、これらの予測子が一時モデルの構築にも使用されます。mining_attribute_clauseは、PREDICTIONファンクションと同様に動作します。「mining_attribute_clause」を参照してください。

関連項目:

- スコアリングの詳細は、Oracle Data Miningユーザーズ・ガイドを参照してください。
- クラスタリングの詳細は、Oracle Data Mining概要を参照してください。

ノート:

次に示す例は、Data Mining のサンプル・プログラムからの抜粋です。サンプル・プログラムの詳細は、Oracle Data Mining ユーザーズ・ガイドの「付録 A」を参照してください。

例

ESAモデルは、200,000を超える特徴をレンダリングする2005 Wikiデータセットに対して構築されています。ドキュメントはテキストとしてマイニングされ、ドキュメント・タイトルは特徴IDとみなされます。

次の例で、類似するテキストのセットを比較して、その後で類似しないテキストのセットを比較するESAアルゴリズムを使用したFEATURE_COMPAREファンクションを示します。

類似するテキスト

```
SELECT 1-FEATURE_COMPARE(esa_wiki_mod USING 'There are several PGA tour golfers from
South Africa' text AND USING 'Nick Price won the 2002 Mastercard Colonial Open' text)
similarity FROM DUAL;
SIMILARITY
----------
      .258
```

出力されるメトリックに、差異の計算結果が示されます。したがって、より小さい数値は、テキストがより類似していることを示します。そのため、問合せでは1から差異を引いた値で、ドキュメントの類似性メトリックが示されます。

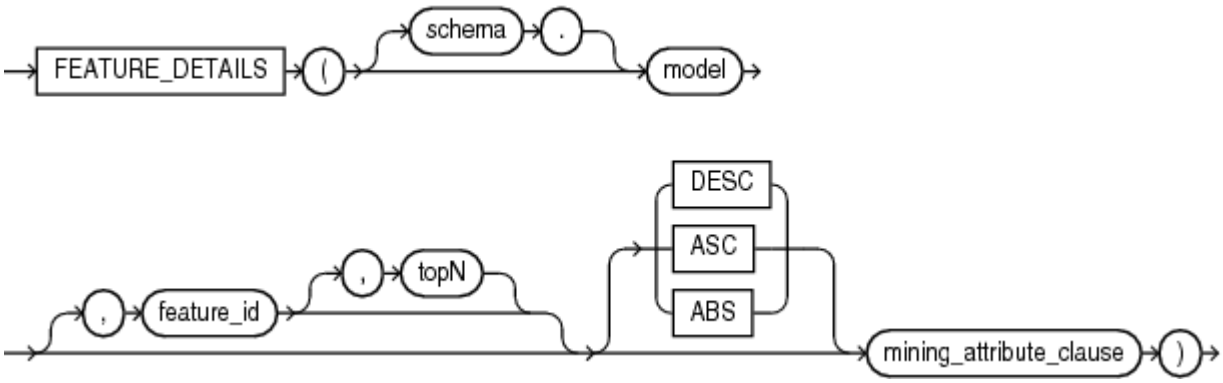
類似しないテキスト

```
SELECT 1-FEATURE_COMPARE(esa_wiki_mod USING 'There are several PGA tour golfers from
South Africa' text AND USING 'John Elway played quarterback for the Denver Broncos'
text) similarity FROM DUAL;
SIMILARITY
----------
      .007
```

FEATURE_DETAILS

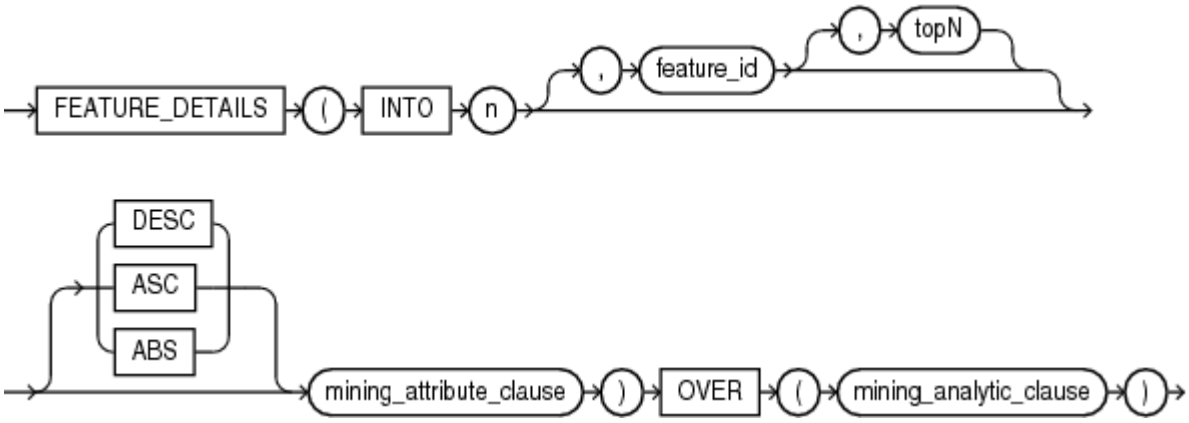
構文

feature_details::=

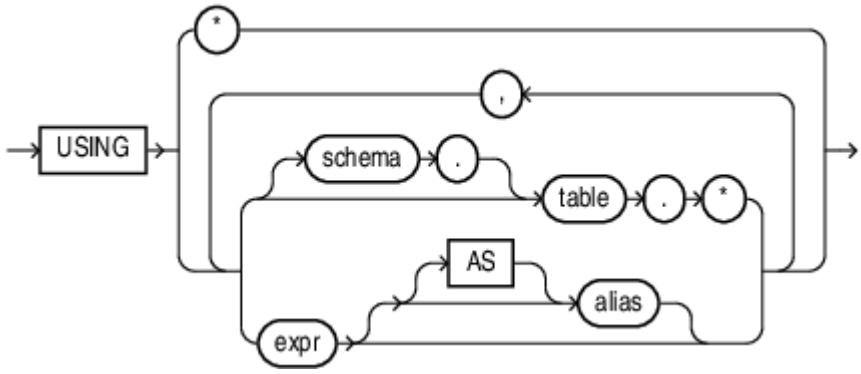


分析構文

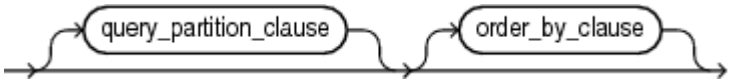
feature_details_analytic::=



mining_attribute_clause::=



mining_analytic_clause::=



関連項目:

mining_analytic_clauseの構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

FEATURE_DETAILSは、選択内に含まれる各行の特徴の詳細を返します。戻り値は、最も値が大きい特徴または指定されたfeature_idの属性について記述したXML文字列です。

topN

topNに値を指定すると、このファンクションは特徴の値に最も影響力のあるN個の属性を返します。topNを指定しないと、このファンクションは最も影響力のある5つの属性を返します。

DESC、ASC、またはABS

返される属性が重みで順序付けされます。属性の重みは、その属性が特徴値に与える正の影響または負の影響を表します。正の重みは、特徴の値が大きくなることを示します。負の重みは、特徴の値が小さくなることを示します。

デフォルトでは、FEATURE_DETAILSは、最大の正の重みを持つ属性を返します(DESC)。ASCを指定すると、最大の負の重みを持つ属性が返されます。ABSを指定すると、正負を問わずに、最大の重みを持つ属性が返されます。結果は、絶対値が高いほうから低いほうに順序付けされています。重みがゼロの属性は、出力に含まれません。

構文の選択

FEATURE_DETAILSは、2つの方法のどちらかでデータにスコアを付けます。1つの方法では、データにマイニング・モデル・オブジェクトを適用します。もう1つの方法では、1つ以上の一時マイニング・モデルを作成して適用する分析句を実行して動的にデータをマイニングします。構文または分析構文を選択します。

- 構文 — 事前に定義されたモデルでデータにスコアを付ける場合は、最初の構文を使用します。特徴抽出モデルの名前を指定します。
- 分析構文 — 事前定義されたモデルなしで、データにスコアを付ける場合は、分析構文を使用します。INTO n (nは、抽出する特徴の数)と、mining_analytic_clause (複数のモデル構築のためにデータをパーティション化する必要がある場合に指定します)を含めます。mining_analytic_clauseは、query_partition_clauseとorder_by_clauseをサポートします。(「analytic_clause::=」を参照。)

FEATURE_DETAILSファンクションの構文では、パーティション化されたモデルをスコアリングするときに、オプションのGROUPINGヒントを使用できます。「GROUPINGヒント」を参照してください。

mining_attribute_clause

mining_attribute_clauseは、スコアの予測子として使用する列の属性を特定します。このファンクションが分析構文で起動された場合は、これらの予測子が一時モデルの構築にも使用されます。mining_attribute_clauseは、PREDICTIONファンクションと同様に動作します。(「mining_attribute_clause::=」を参照)

関連項目:

- スコアリングの詳細は、Oracle Data Miningユーザーズ・ガイドを参照してください。
- 特徴抽出の詳細は、『Oracle Data Mining概要』を参照してください。

ノート:

次に示す例は、Data Mining のサンプル・プログラムからの抜粋です。サンプル・プログラムの詳細は、Oracle Data Mining ユーザーズ・ガイドの「付録 A」を参照してください。

例

この例では、特徴抽出モデルnmf_sh_sampleを使用して、データにスコアを付けます。この問合せは、顧客100002を代表する3つの特徴と、それらの特徴に最も影響を与える属性を返します。

```
SELECT S.feature_id fid, value val,
       FEATURE_DETAILS(nmf_sh_sample, S.feature_id, 5 using T.*) det
   FROM
     (SELECT v.*, FEATURE_SET(nmf_sh_sample, 3 USING *) fset
         FROM mining_data_apply_v v
         WHERE cust_id = 100002) T,
   TABLE(T.fset) S
ORDER BY 2 DESC;

 FID    VAL  DET
---- ------  --------------------------------------------------------------------------
------------
   5  3.492  <Details algorithm="Non-Negative Matrix Factorization" feature="5">
              <Attribute name="BULK_PACK_DISKETTES" actualValue="1" weight=".077"
rank="1"/>
              <Attribute name="OCCUPATION" actualValue="Prof." weight=".062"
rank="2"/>
              <Attribute name="BOOKKEEPING_APPLICATION" actualValue="1" weight=".001"
rank="3"/>
              <Attribute name="OS_DOC_SET_KANJI" actualValue="0" weight="0" rank="4"/>
              <Attribute name="YRS_RESIDENCE" actualValue="4" weight="0" rank="5"/>
              </Details>
   3  1.928  <Details algorithm="Non-Negative Matrix Factorization" feature="3">
              <Attribute name="HOUSEHOLD_SIZE" actualValue="2" weight=".239"
rank="1"/>
              <Attribute name="CUST_INCOME_LEVEL" actualValue="L: 300¥,000 and above"
               weight=".051" rank="2"/>
              <Attribute name="FLAT_PANEL_MONITOR" actualValue="1" weight=".02"
rank="3"/>
              <Attribute name="HOME_THEATER_PACKAGE" actualValue="1" weight=".006"
rank="4"/>
              <Attribute name="AGE" actualValue="41" weight=".004" rank="5"/>
              </Details>
   8   .816  <Details algorithm="Non-Negative Matrix Factorization" feature="8">
              <Attribute name="EDUCATION" actualValue="Bach." weight=".211" rank="1"/>
              <Attribute name="CUST_MARITAL_STATUS" actualValue="NeverM" weight=".143"
rank="2"/>
              <Attribute name="FLAT_PANEL_MONITOR" actualValue="1" weight=".137"
rank="3"/>
              <Attribute name="CUST_GENDER" actualValue="F" weight=".044" rank="4"/>
              <Attribute name="BULK_PACK_DISKETTES" actualValue="1" weight=".032"
rank="5"/>
              </Details>
```

分析の例

この例では、顧客の属性を6つの特徴に動的にマップして、顧客100001の特徴マッピングを返します。

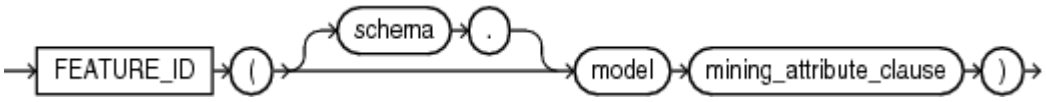
```
SELECT feature_id, value
  FROM (
     SELECT cust_id, feature_set(INTO 6 USING *) OVER () fset
        FROM mining_data_apply_v),
  TABLE (fset)
  WHERE cust_id = 100001
  ORDER BY feature_id;
```

```
FEATURE_ID    VALUE
---------- --------
         1    2.670
         2     .000
         3    1.792
         4     .000
         5     .000
         6    3.379
```

FEATURE_ID

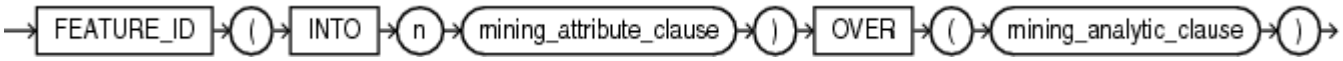
構文

feature_id::=

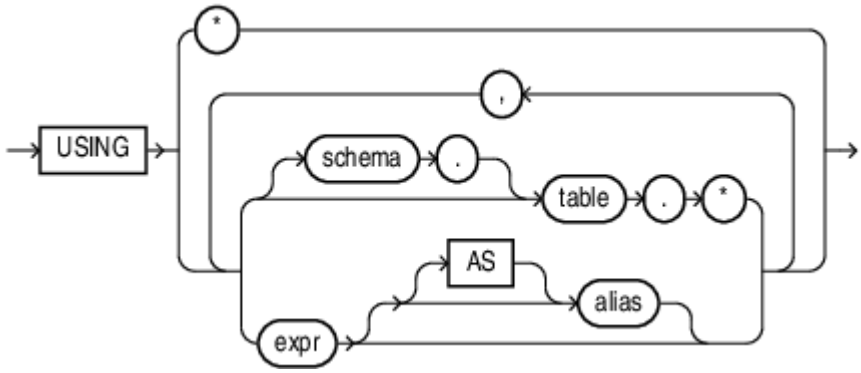


分析構文

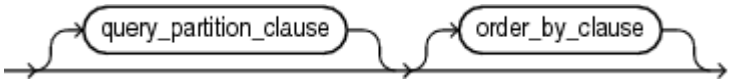
feature_id_analytic::=



mining_attribute_clause::=



mining_analytic_clause::=



関連項目:

`mining_analytic_clause`の構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

FEATURE_IDは、選択内に含まれる各行の最も値が大きい特徴の識別子(ID)を返します。この特徴IDは、Oracle NUMBERとして返されます。

構文の選択

FEATURE_IDは、2つの方法のどちらかでデータにスコアを付けます。1つの方法では、データにマイニング・モデル・オブジェクトを適用します。もう1つの方法では、1つ以上の一時マイニング・モデルを作成して適用する分析句を実行して動的にデータをマイニングします。構文または分析構文を選択します。

- 構文 — 事前に定義されたモデルでデータにスコアを付ける場合は、最初の構文を使用します。特徴抽出モデルの名前を指定します。

- 分析構文 — 事前定義されたモデルなしで、データにスコアを付ける場合は、分析構文を使用します。INTO n (nは、抽出する特徴の数)と、mining_analytic_clause (複数のモデル構築のためにデータをパーティション化する必要がある場合に指定します)を含めます。mining_analytic_clauseは、query_partition_clauseとorder_by_clauseをサポートします。(「analytic_clause::=」を参照。)

FEATURE_IDファンクションの構文では、パーティション化されたモデルをスコアリングするときに、オプションのGROUPINGヒントを使用できます。「GROUPINGヒント」を参照してください。

mining_attribute_clause

mining_attribute_clauseは、スコアの予測子として使用する列の属性を特定します。このファンクションが分析構文で起動された場合は、これらの予測子が一時モデルの構築にも使用されます。mining_attribute_clauseは、PREDICTIONファンクションと同様に動作します。(「mining_attribute_clause::=」を参照)

関連項目:

- スコアリングの詳細は、Oracle Data Miningユーザーズ・ガイドを参照してください。
- 特徴抽出の詳細は、『Oracle Data Mining概要』を参照してください。

ノート:

次に示す例は、Data Mining のサンプル・プログラムからの抜粋です。サンプル・プログラムの詳細は、Oracle Data Mining ユーザーズ・ガイドの「付録 A」を参照してください。

例

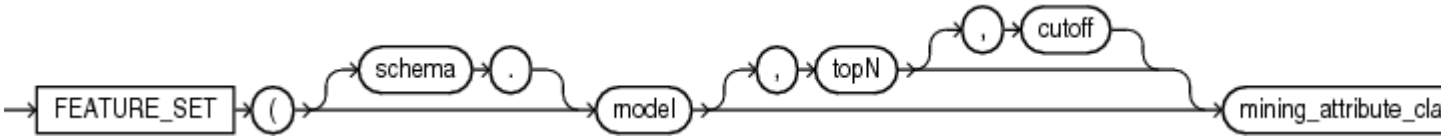
この例では、データ・セット内の特徴と、それに対応する顧客の件数を示します。

```
SELECT FEATURE_ID(nmf_sh_sample USING *) AS feat, COUNT(*) AS cnt
  FROM nmf_sh_sample_apply_prepared
  GROUP BY FEATURE_ID(nmf_sh_sample USING *)
  ORDER BY cnt DESC, feat DESC;
      FEAT        CNT
---------- ----------
         7       1443
         2         49
         3          6
         6          1
         1          1
```

FEATURE_SET

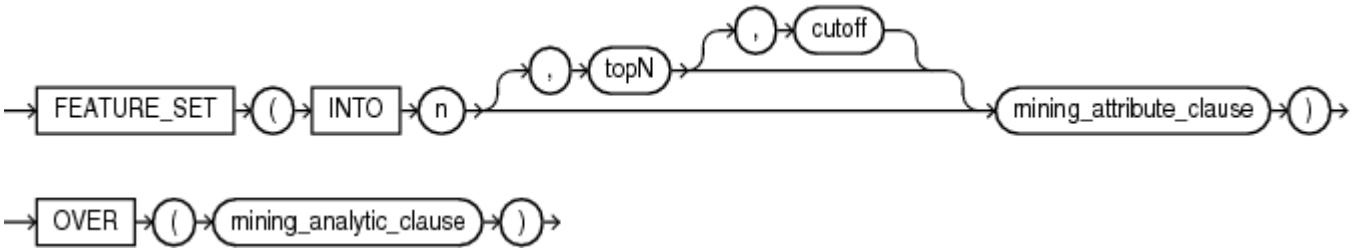
構文

feature_set::=

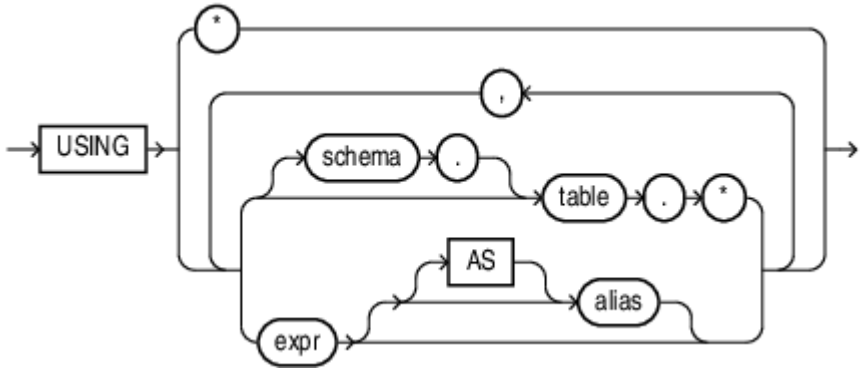


分析構文

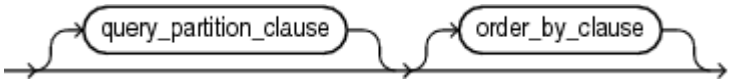
feature_set_analytic::=



mining_attribute_clause::=



mining_analytic_clause::=



関連項目:

`mining_analytic_clause`の構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

FEATURE_SETは、選択内に含まれる各行の特徴IDと特徴値の組で構成されるセットを返します。戻り値は、フィールド名がFEATURE_IDおよびVALUEのオブジェクトのVARRAYです。どちらのフィールドのデータ型もNUMBERです。

topNおよびcutoff

`topN`と`cutoff`を指定すると、このファンクションから返される特徴の数を制限できます。デフォルトでは、`topN`および`cutoff`

がnullであり、すべての特徴が戻されます。

- topNは、最も値が大きいほうからN個の特徴値です。N番目の値を持つ特徴が複数ある場合でも、ファンクションが選択するのはそのうち1つのみです。
- cutoffは、値のしきい値です。cutoff以上の特徴のみが返されます。cutoffのみでフィルタ処理するには、topNにNULLを指定します。

cutoff以上の特徴をN個まで返すようにするには、topNとcutoffの両方を指定します。

構文の選択

FEATURE_SETは、2つの方法のどちらかでデータにスコアを付けます。1つの方法では、データにマイニング・モデル・オブジェクトを適用します。もう1つの方法では、1つ以上の一時マイニング・モデルを作成して適用する分析句を実行して動的にデータをマイニングします。構文または分析構文を選択します。

- 構文 — 事前に定義されたモデルでデータにスコアを付ける場合は、最初の構文を使用します。特徴抽出モデルの名前を指定します。
- 分析構文 — 事前定義されたモデルなしで、データにスコアを付ける場合は、分析構文を使用します。INTO n (nは、抽出する特徴の数)と、mining_analytic_clause (複数のモデル構築のためにデータをパーティション化する必要がある場合に指定します)を含めます。mining_analytic_clauseは、query_partition_clauseとorder_by_clauseをサポートします。(「analytic_clause::=」を参照。)

FEATURE_SETファンクションの構文では、パーティション化されたモデルをスコアリングするときに、オプションのGROUPINGヒントを使用できます。「GROUPINGヒント」を参照してください。

mining_attribute_clause

mining_attribute_clauseは、スコアの予測子として使用する列の属性を特定します。このファンクションが分析構文で起動された場合は、これらの予測子が一時モデルの構築にも使用されます。mining_attribute_clauseは、PREDICTIONファンクションと同様に動作します。(「mining_attribute_clause::=」を参照)

関連項目:

- スコアリングの詳細は、Oracle Data Miningユーザーズ・ガイドを参照してください。
- 特徴抽出の詳細は、『Oracle Data Mining概要』を参照してください。

ノート:

次に示す例は、Data Mining のサンプル・プログラムからの抜粋です。サンプル・プログラムの詳細は、Oracle Data Mining ユーザーズ・ガイドの「付録 A」を参照してください。

例

この例では、特定の顧客レコードに対応する上位の特徴を示し、各特徴の上位の属性を(0.25を超える係数を基準にして)決定します。

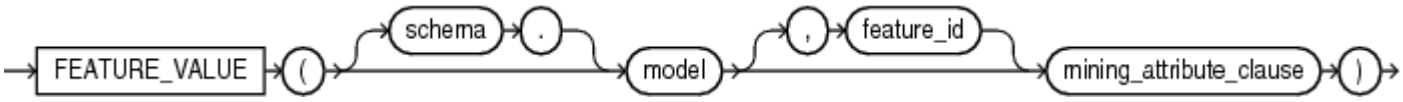
```
WITH
feat_tab AS (
SELECT F.feature_id fid,
       A.attribute_name attr,
```

```
       TO_CHAR(A.attribute_value) val,
       A.coefficient coeff
  FROM TABLE(DBMS_DATA_MINING.GET_MODEL_DETAILS_NMF('nmf_sh_sample')) F,
       TABLE(F.attribute_set) A
 WHERE A.coefficient > 0.25
),
feat AS (
SELECT fid,
       CAST(COLLECT(Featattr(attr, val, coeff))
         AS Featattrs) f_attrs
  FROM feat_tab
GROUP BY fid
),
cust_10_features AS (
SELECT T.cust_id, S.feature_id, S.value
  FROM (SELECT cust_id, FEATURE_SET(nmf_sh_sample, 10 USING *) pset
          FROM nmf_sh_sample_apply_prepared
         WHERE cust_id = 100002) T,
       TABLE(T.pset) S
)
SELECT A.value, A.feature_id fid,
       B.attr, B.val, B.coeff
  FROM cust_10_features A,
       (SELECT T.fid, F.*
          FROM feat T,
               TABLE(T.f_attrs) F) B
 WHERE A.feature_id = B.fid
ORDER BY A.value DESC, A.feature_id ASC, coeff DESC, attr ASC, val ASC;
   VALUE  FID ATTR                      VAL                       COEFF
-------- ---- ------------------------ ------------------------ -------
  6.8409    7 YRS_RESIDENCE                                    1.3879
  6.8409    7 BOOKKEEPING_APPLICATION                           .4388
  6.8409    7 CUST_GENDER              M                        .2956
  6.8409    7 COUNTRY_NAME             United States of America  .2848
  6.4975    3 YRS_RESIDENCE                                    1.2668
  6.4975    3 BOOKKEEPING_APPLICATION                           .3465
  6.4975    3 COUNTRY_NAME             United States of America  .2927
  6.4886    2 YRS_RESIDENCE                                    1.3285
  6.4886    2 CUST_GENDER              M                        .2819
  6.4886    2 PRINTER_SUPPLIES                                  .2704
  6.3953    4 YRS_RESIDENCE                                    1.2931
  5.9640    6 YRS_RESIDENCE                                    1.1585
  5.9640    6 HOME_THEATER_PACKAGE                              .2576
  5.2424    5 YRS_RESIDENCE                                    1.0067
  2.4714    8 YRS_RESIDENCE                                     .3297
  2.3559    1 YRS_RESIDENCE                                     .2768
  2.3559    1 FLAT_PANEL_MONITOR                                .2593
```

FEATURE_VALUE

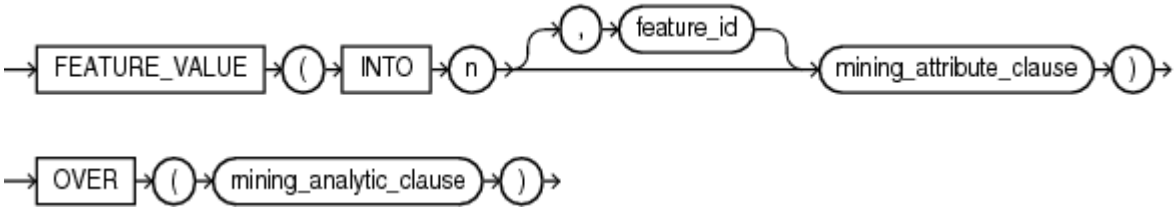
構文

feature_value::=

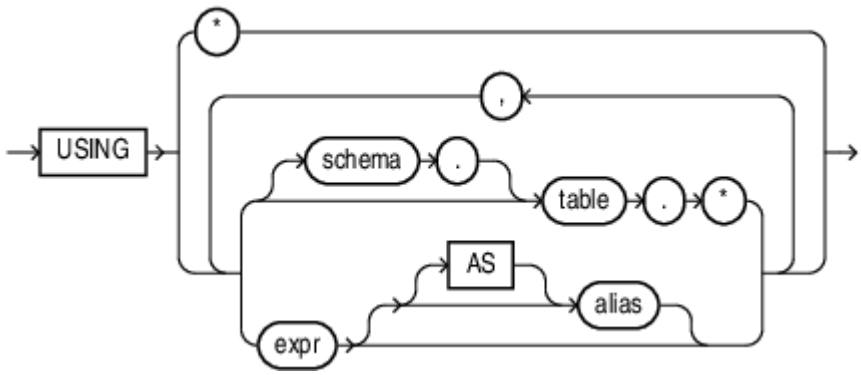


分析構文

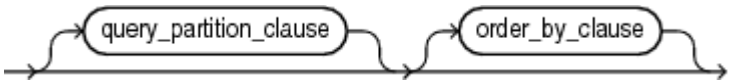
feature_value_analytic::=



mining_attribute_clause::=



mining_analytic_clause::=



関連項目:

`mining_analytic_clause`の構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

FEATURE_VALUEは、選択内に含まれる各行の特徴の値を返します。この値は、最も値が大きい特徴または指定された`feature_id`を参照します。特徴の値は、BINARY_DOUBLEとして返されます。

構文の選択

FEATURE_VALUEは、2つの方法のどちらかでデータにスコアを付けます。1つの方法では、データにマイニング・モデル・オブジェクトを適用します。もう1つの方法では、1つ以上の一時マイニング・モデルを作成して適用する分析句を実行して動的にデータをマイニングします。構文または分析構文を選択します。

- 構文 — 事前に定義されたモデルでデータにスコアを付ける場合は、最初の構文を使用します。特徴抽出モデルの名前を指定します。
- 分析構文 — 事前定義されたモデルなしで、データにスコアを付ける場合は、分析構文を使用します。INTO n (nは、抽出する特徴の数)と、mining_analytic_clause (複数のモデル構築のためにデータをパーティション化する必要がある場合に指定します)を含めます。mining_analytic_clauseは、query_partition_clauseとorder_by_clauseをサポートします。(「analytic_clause::=」を参照。)

FEATURE_VALUEファンクションの構文では、パーティション化されたモデルをスコアリングするときに、オプションのGROUPINGヒントを使用できます。「GROUPINGヒント」を参照してください。

mining_attribute_clause

mining_attribute_clauseは、スコアの予測子として使用する列の属性を特定します。分析構文でファンクションが起動されると、このデータは一時モデルの構築にも使用されます。mining_attribute_clauseは、PREDICTIONファンクションと同様に動作します。(「mining_attribute_clause::=」を参照)

関連項目:

- スコアリングの詳細は、Oracle Data Miningユーザーズ・ガイドを参照してください。
- 特徴抽出の詳細は、『Oracle Data Mining概要』を参照してください。

ノート:

次に示す例は、Data Mining のサンプル・プログラムからの抜粋です。サンプル・プログラムの詳細は、Oracle Data Mining ユーザーズ・ガイドの「付録 A」を参照してください。

例

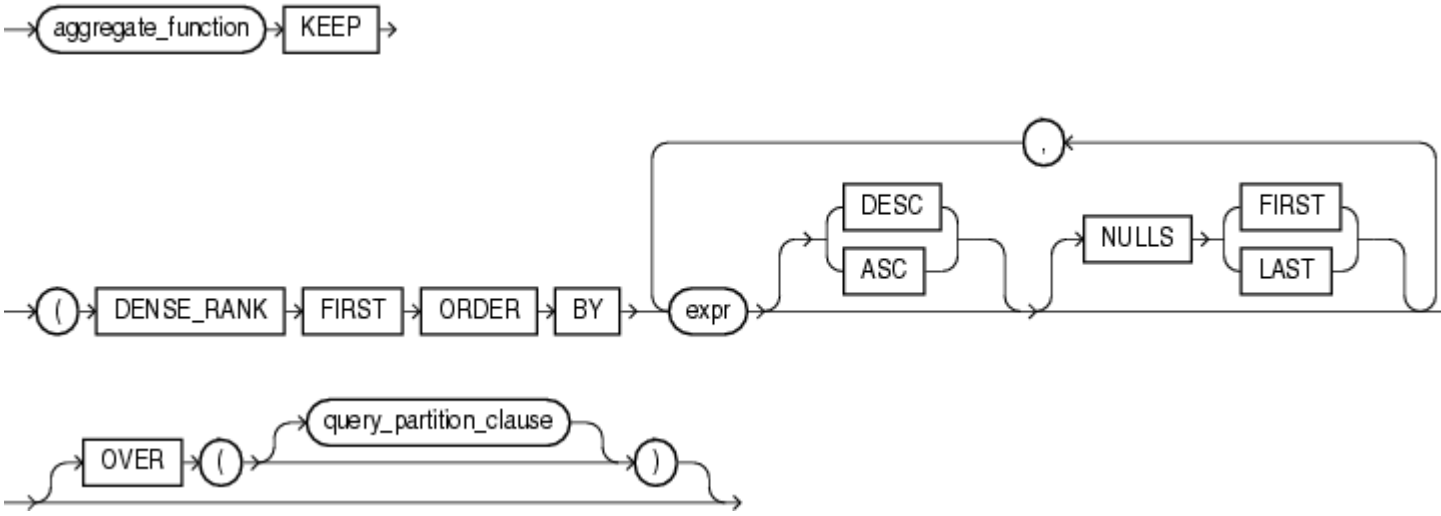
次の例では、特徴3に対応する顧客を、一致する品質の順序で示します。

```
SELECT *
  FROM (SELECT cust_id, FEATURE_VALUE(nmf_sh_sample, 3 USING *) match_quality
          FROM nmf_sh_sample_apply_prepared
          ORDER BY match_quality DESC)
  WHERE ROWNUM < 11;
   CUST_ID MATCH_QUALITY
---------- -------------
    100210    19.4101627
    100962    15.2482251
    101151    14.5685197
    101499    14.4186292
    100363    14.4037396
    100372    14.3335148
    100982    14.1716545
    101039    14.1079914
    100759    14.0913761
    100953    14.0799737
```

FIRST

構文

first::=



関連項目:

ORDER BY句およびOVER句の構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

FIRSTファンクションとLASTファンクションは類似しています。両方のファンクションとも、与えられたソート指定に対して、FIRSTまたはLASTとしてランク付けされた一連の行の一連の値を操作する集計ファンクションおよび分析ファンクションです。1つの行のみがFIRSTまたはLASTとしてランク付けされている場合、集計は、1つの要素のみを持つセットを操作します。

OVER句を省略すると、FIRSTファンクションおよびLASTファンクションは集計ファンクションとして処理されます。OVER句を指定すると、これらのファンクションを分析ファンクションとして使用できます。OVER句のうち、query_partition_clauseの部分のみがこれらのファンクションで有効です。OVER句を指定してquery_partition_clauseを省略すると、このファンクションは分析ファンクションとして処理されますが、分析対象のウィンドウは表全体になります。

これらのファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を返します。

ソートされたグループの最初または最後の行の値が必要であり、その値がソート・キーでない場合、FIRSTおよびLASTファンクションは、自己結合またはビューの必要性を排除し、パフォーマンスを向上させます。

- aggregate_function引数は、MIN、MAX、SUM、AVG、COUNT、VARIANCEまたはSTDDEVファンクションのいずれかです。FIRSTまたはLASTのいずれかにランク付けされた行の値を操作します。1つの行のみがFIRSTまたはLASTとしてランク付けされている場合、集計は、単一(集計ではない)のセットを操作します。
- KEEPキーワードはセマンティクスを明確にするためのものです。これはaggregate_functionを修飾し、戻されるaggregate_functionの値がFIRSTまたはLASTのみであることを示します。
- DENSE_RANK FIRSTまたはDENSE_RANK LASTは、最小(FIRST)または最大(LAST)稠密ランク(「オリンピック・ランク」)を持つ行のみを集計することを示します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。「LAST」も参照してください。

集計の例

次の例では、サンプル表hr.employeesの各部門で、歩合が最低である従業員の中での最低給与、および歩合が最高の従業員の中での最高給与を戻します。

```
SELECT department_id,
       MIN(salary) KEEP (DENSE_RANK FIRST ORDER BY commission_pct) "Worst",
       MAX(salary) KEEP (DENSE_RANK LAST ORDER BY commission_pct) "Best"
  FROM employees
  GROUP BY department_id
  ORDER BY department_id;
DEPARTMENT_ID      Worst       Best
------------- ---------- ----------
           10       4400       4400
           20       6000      13000
           30       2500      11000
           40       6500       6500
           50       2100       8200
           60       4200       9000
           70      10000      10000
           80       6100      14000
           90      17000      24000
          100       6900      12008
          110       8300      12008
                    7000       7000
```

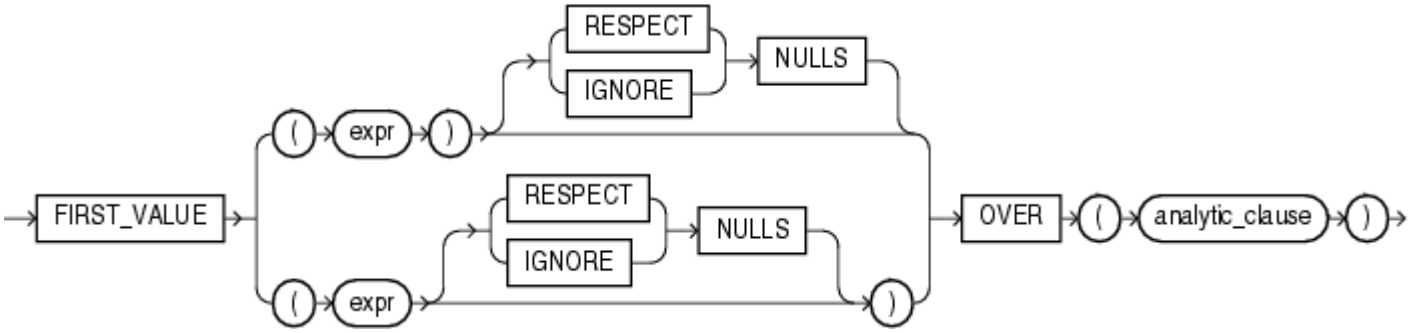
分析の例

次の例では、前述の例と同じ計算をしますが、当該部門の各従業員の結果を戻します。

```
SELECT last_name, department_id, salary,
       MIN(salary) KEEP (DENSE_RANK FIRST ORDER BY commission_pct)
         OVER (PARTITION BY department_id) "Worst",
       MAX(salary) KEEP (DENSE_RANK LAST ORDER BY commission_pct)
         OVER (PARTITION BY department_id) "Best"
   FROM employees
   ORDER BY department_id, salary, last_name;
LAST_NAME           DEPARTMENT_ID     SALARY      Worst       Best
------------------- ------------- ---------- ---------- ----------
Whalen                         10       4400       4400       4400
Fay                            20       6000       6000      13000
Hartstein                      20      13000       6000      13000
. . .
Gietz                         110       8300       8300      12008
Higgins                       110      12008       8300      12008
Grant                                   7000       7000       7000
```

FIRST_VALUE

構文



関連項目:

構文、セマンティクス、制限事項、およびexprの書式の詳細は、「分析ファンクション」を参照してください。

目的

FIRST_VALUEは分析ファンクションです。これは、順序付けられた値の集合にある最初の値を戻します。集合内の最初の値がNULLの場合、IGNORE NULLSを指定していないかぎり、ファンクションはNULLを戻します。この設定は、データの稠密化に役立ちます。

ノート:

この構文の 2 つの書式は、同じ動作になります。上のブランチは ANSI 形式です。ANSI との互換性を維持するために、こちらを使用することをお薦めします。

{RESPECT | IGNORE} NULLSは、exprのNULL値を計算に含めるか除外するかを指定します。デフォルトはRESPECT NULLSです。IGNORE NULLSを指定すると、FIRST_VALUEは集合内の最初のNULLではない値を戻します。すべての値がNULLの場合はNULLを戻します。データの稠密化の例は、「パーティション化された外部結合の使用方法: 例」を参照してください。

exprには、FIRST_VALUEまたは他の分析ファンクションを使用して分析ファンクションをネストできません。ただし、他の組込みファンクション式をexprで使用できます。exprの書式の詳細は、「SQL式」を参照してください。

関連項目:

FIRST_VALUEの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、部門90の各従業員について、その部門で給与が一番少ない従業員の名前を選択します。

```
SELECT employee_id, last_name, salary, hire_date,
       FIRST_VALUE(last_name)
         OVER (ORDER BY salary ASC ROWS UNBOUNDED PRECEDING) AS fv
  FROM (SELECT * FROM employees
```

```
          WHERE department_id = 90
          ORDER BY hire_date);
EMPLOYEE_ID LAST_NAME                     SALARY HIRE_DATE FV
----------- ------------------------- ---------- --------- -------
        102 De Haan                        17000 13-JAN-01 De Haan
        101 Kochhar                        17000 21-SEP-05 De Haan
        100 King                           24000 17-JUN-03 De Haan
```

例では、FIRST_VALUEファンクションの非決定的な性質が示されています。KochharとDe Haanの給与は同じであるため、Kochharの次の行にDe Haanがあります。Kochharが最初に表示されているのは、副問合せが戻す行がhire_dateで順序付けられているためです。ただし、副問合せが戻す行がhire_dateで降順に順序付けられている場合は、次の例に示すとおり、ファンクションは異なる値を戻します。

```
SELECT employee_id, last_name, salary, hire_date,
       FIRST_VALUE(last_name)
         OVER (ORDER BY salary ASC ROWS UNBOUNDED PRECEDING) AS fv
  FROM (SELECT * FROM employees
          WHERE department_id = 90
          ORDER by hire_date DESC);
EMPLOYEE_ID LAST_NAME                     SALARY HIRE_DATE FV
----------- ------------------------- ---------- --------- -------
        101 Kochhar                        17000 21-SEP-05 Kochhar
        102 De Haan                        17000 13-JAN-01 Kochhar
        100 King                           24000 17-JUN-03 Kochhar
```

次の2つの例では、一意キーで順序付けることによって、FIRST_VALUEファンクションを決定的にする方法を示しています。給与と一意キーemployee_idの両方でファンクション内を順序付けると、副問合せの順序付けにかかわらず、同じ結果が戻されます。

```
SELECT employee_id, last_name, salary, hire_date,
       FIRST_VALUE(last_name)
         OVER (ORDER BY salary ASC, employee_id ROWS UNBOUNDED PRECEDING) AS fv
  FROM (SELECT * FROM employees
          WHERE department_id = 90
          ORDER BY hire_date);

EMPLOYEE_ID LAST_NAME                     SALARY HIRE_DATE FV
----------- ------------------------- ---------- --------- -------
        101 Kochhar                        17000 21-SEP-05 Kochhar
        102 De Haan                        17000 13-JAN-01 Kochhar
        100 King                           24000 17-JUN-03 Kochhar


SELECT employee_id, last_name, salary, hire_date,
       FIRST_VALUE(last_name)
         OVER (ORDER BY salary ASC, employee_id ROWS UNBOUNDED PRECEDING) AS fv
   FROM (SELECT * FROM employees
           WHERE department_id = 90
           ORDER BY hire_date DESC);

EMPLOYEE_ID LAST_NAME                     SALARY HIRE_DATE FV
----------- ------------------------- ---------- --------- -------
        101 Kochhar                        17000 21-SEP-05 Kochhar
        102 De Haan                        17000 13-JAN-01 Kochhar
        100 King                           24000 17-JUN-03 Kochhar
```

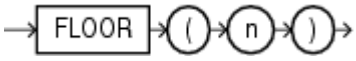
次の2つの例では、論理オフセット(ROWSのかわりにRANGE)を使用すると、FIRST_VALUEファンクションが決定的になることを示しています。ORDER BY式に重複値がある場合は、FIRST_VALUEがexprの最小値になります。

```
SELECT employee_id, last_name, salary, hire_date,
       FIRST_VALUE(last_name)
         OVER (ORDER BY salary ASC RANGE UNBOUNDED PRECEDING) AS fv
  FROM (SELECT * FROM employees
```

```
          WHERE department_id = 90
          ORDER BY hire_date);

EMPLOYEE_ID LAST_NAME                     SALARY HIRE_DATE FV
----------- ------------------------- ---------- --------- -------
        102 De Haan                        17000 13-JAN-01 De Haan
        101 Kochhar                        17000 21-SEP-05 De Haan
        100 King                           24000 17-JUN-03 De Haan


SELECT employee_id, last_name, salary, hire_date,
       FIRST_VALUE(last_name)
         OVER (ORDER BY salary ASC RANGE UNBOUNDED PRECEDING) AS fv
  FROM (SELECT * FROM employees
          WHERE department_id = 90
          ORDER BY hire_date DESC);

EMPLOYEE_ID LAST_NAME                     SALARY HIRE_DATE FV
----------- ------------------------- ---------- --------- -------
        102 De Haan                        17000 13-JAN-01 De Haan
        101 Kochhar                        17000 21-SEP-05 De Haan
        100 King                           24000 17-JUN-03 De Haan
```

FLOOR

構文



目的

FLOORはn以下の最大整数を戻します。数値nは常に、0 <= f < 1とn = k + fなどのように、整数kと正の小数部fの合計として記述できます。FLOORの値は整数kです。したがって、FLOORの値がnそのものになるのはnが小数部を持たない整数である場合にかぎられます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を返します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。「CEIL」も参照してください。

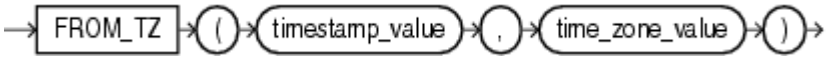
例

次の例では、15.7以下である最大の整数を戻します。

```
SELECT FLOOR(15.7) "Floor"
  FROM DUAL;
     Floor
----------
        15
```

FROM_TZ

構文



目的

FROM_TZファンクションは、タイムスタンプ値およびタイムゾーンをTIMESTAMP WITH TIME ZONE値に変換します。time_zone_valueは、'TZH:TZM'書式の文字列、またはオプションのTZD書式を持つTZR書式で文字列を戻す文字式です。

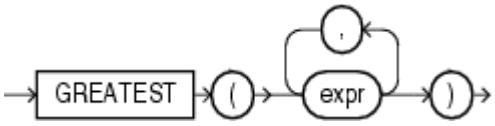
例

次の例では、タイムスタンプ値をTIMESTAMP WITH TIME ZONEに戻します。

```
SELECT FROM_TZ(TIMESTAMP '2000-03-28 08:00:00', '3:00')
  FROM DUAL;
FROM_TZ(TIMESTAMP'2000-03-2808:00:00','3:00')
---------------------------------------------------------------
28-MAR-00 08.00.000000000 AM +03:00
```

GREATEST

構文



目的

GREATESTは、1つ以上の式のリストの最大値を戻します。Oracle Databaseは、最初のexprを使用して戻り型を判断します。最初のexprが数値である場合、Oracleは、数値の優先順位が最も高い引数を判断し、比較の前に残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。最初のexprが数値ではない場合、比較の前に、2番目以降の各exprが最初のexprのデータ型に暗黙的に変換されます。

Oracle Databaseは、非空白埋め比較セマンティクスを使用して各exprを比較します。比較では、デフォルトの場合はバイナリが使用され、NLS_COMPパラメータがLINGUISTICに設定され、NLS_SORTパラメータにBINARY以外の設定がある場合は言語が使用されます。文字の比較は、データベース文字セットの文字の数値コードに基づいて行われます。また、文字ごとではなく、一連のバイトとして扱われる文字列全体で行われます。このファンクションによって戻される値が文字データの場合、そのデータ型は、最初のexprが文字データ型の場合はVARCHAR2、最初のexprが各国語キャラクタ・データ型の場合はNVARCHAR2になります。

関連項目:

- 文字の比較の詳細は、「データ型の比較規則」を参照してください。
- 暗黙的な変換の詳細は、表2-8を参照してください。2進浮動小数点の比較セマンティクスの詳細は、「浮動小数点数」を参照してください。
- 「LEAST」を参照してください(LEASTは1つ以上の式のリストの最小値を戻します)。
- exprの文字値を比較するためにGREATESTで使用される照合を定義する照合決定ルール、およびこのファンクションの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の文では、最大値を持つ文字列を検索します。

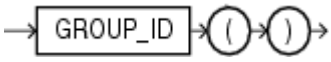
```
SELECT GREATEST('HARRY', 'HARRIOT', 'HAROLD') "Greatest"
  FROM DUAL;

Greatest
--------
HARRY
```

次の文では、最初の引数は数字です。Oracle Databaseは、数値の優先順位が最も高い引数は2番目の引数であると判断し、残りの引数を2番目の引数のデータ型に変換して、そのデータ型で最大値を戻します。

```
SELECT GREATEST (1, '3.925', '2.4') "Greatest"
  FROM DUAL;
Greatest
--------
   3.925
```

GROUP_ID

構文



目的

GROUP_IDは、GROUP BY指定の結果から、重複するグループを識別します。このファンクションは、問合せ結果から重複グループを除外する場合に有効です。このファンクションは、重複グループを一意に識別するために、OracleのNUMBER値を戻します。このファンクションは、1つのGROUP BY句を含むSELECT文のみに適用できます。

特定のグループ化でn個の重複が存在する場合、GROUP_IDは0からn-1の範囲の数値を戻します。

例

次の例では、サンプル表sh.countriesおよびsh.salesの問合せの結果、重複するco.country_regionグループ化に値1を割り当てます。

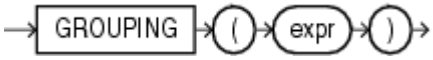
```
SELECT co.country_region, co.country_subregion,
       SUM(s.amount_sold) "Revenue", GROUP_ID() g
  FROM sales s, customers c, countries co
  WHERE s.cust_id = c.cust_id
    AND c.country_id = co.country_id
    AND s.time_id = '1-JAN-00'
    AND co.country_region IN ('Americas', 'Europe')
  GROUP BY GROUPING SETS ( (co.country_region, co.country_subregion),
                           (co.country_region, co.country_subregion) )
  ORDER BY co.country_region, co.country_subregion, "Revenue", g;
COUNTRY_REGION       COUNTRY_SUBREGION                 Revenue          G
-------------------- ------------------------------ ---------- ----------
Americas             Northern America                    944.6          0
Americas             Northern America                    944.6          1
Europe               Western Europe                     566.39          0
Europe               Western Europe                     566.39          1
```

GROUP_IDが1より小さい行のみを戻すには、文の最後に次のHAVING句を追加します。

```
HAVING GROUP_ID() < 1
```

GROUPING

構文



目的

GROUPINGは、通常のグループ化された行と超集合行を区別します。ROLLUPやCUBEなどのGROUP BYの拡張機能は、すべての値の集合がNULLで表される超集合行を生成します。GROUPINGファンクションを使用すると、通常の行に含まれるNULLと超集合行ですべての値の集合を表すNULLを区別できます。

GROUPINGファンクションのexprは、GROUP BY句の式のいずれかと一致する必要があります。行のexprの値がすべての値の集合を表すNULLの場合、このファンクションは1の値を戻します。それ以外の場合は、0(ゼロ)を戻します。GROUPINGファンクションは、OracleのNUMBERデータ型の値を戻します。これらの用語の詳細は、「SELECT」の「group_by_clause」を参照してください。

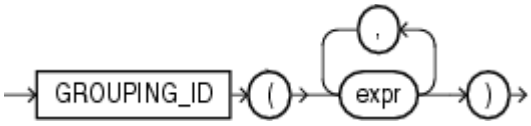
例

次の例では、サンプル表hr.departmentsおよびhr.employeesで、GROUPINGファンクションが1(表の通常の行ではなく超集合行)を戻す場合、それ以外の場合に表示されるNULLのかわりに、文字列「All Jobs」が「JOB」列に表示されます。

```
SELECT
    DECODE(GROUPING(department_name), 1, 'ALL DEPARTMENTS', department_name)
      AS department,
    DECODE(GROUPING(job_id), 1, 'All Jobs', job_id) AS job,
    COUNT(*) "Total Empl",
    AVG(salary) * 12 "Average Sal"
  FROM employees e, departments d
  WHERE d.department_id = e.department_id
  GROUP BY ROLLUP (department_name, job_id)
  ORDER BY department, job;
DEPARTMENT                     JOB        Total Empl Average Sal
------------------------------ ---------- ---------- -----------
ALL DEPARTMENTS                All Jobs          106  77481.0566
Accounting                     AC_ACCOUNT          1       99600
Accounting                     AC_MGR              1      144096
Accounting                     All Jobs            2      121848
Administration                 AD_ASST             1       52800
Administration                 All Jobs            1       52800
Executive                      AD_PRES             1      288000
Executive                      AD_VP               2      204000
Executive                      All Jobs            3      232000
Finance                        All Jobs            6      103216
Finance                        FI_ACCOUNT          5       95040
. . .
```

GROUPING_ID

構文



目的

GROUPING_IDは、行に関連するGROUPINGビット・ベクトルに対応する数値を戻します。GROUPING_IDは、ROLLUPやCUBEなど、GROUP BYの拡張機能およびGROUPINGファンクションを含むSELECT文にのみ適用できます。多くのGROUP BY式を含む問合せでは、多くのGROUPINGファンクションを必要とする特定の行のGROUP BYレベルを指定するため、非常に複雑なSQLになります。GROUPING_IDは、このような場合に有効です。

GROUPING_IDは、複数のGROUPINGファンクションの結果をビット・ベクトル(1と0を組み合せた文字列)に連結したものと同じです。GROUPING_IDを使用すると、複数のGROUPINGファンクションを使用する必要がなくなり、行のフィルタ条件の表記が簡単になります。GROUPING_IDを使用すると、要求する行が単一の条件(GROUPING_ID = n)によって識別されるため、行のフィルタ処理が簡単になります。このファンクションは、1つの表に複数のレベルの集計を格納する場合に、特に有効です。

例

次の例では、サンプル表sh.salesの問合せからグループ化IDを抽出する方法を示します。

```
SELECT channel_id, promo_id, sum(amount_sold) s_sales,
       GROUPING(channel_id) gc,
       GROUPING(promo_id) gp,
       GROUPING_ID(channel_id, promo_id) gcp,
       GROUPING_ID(promo_id, channel_id) gpc
  FROM sales
  WHERE promo_id > 496
  GROUP BY CUBE(channel_id, promo_id)
  ORDER BY channel_id, promo_id, s_sales, gc;

CHANNEL_ID   PROMO_ID    S_SALES         GC          GP         GCP         GPC
---------- ---------- ---------- ---------- ---------- ---------- ----------
         2        999 25797563.2          0          0          0          0
         2            25797563.2          0          1          1          2
         3        999 55336945.1          0          0          0          0
         3            55336945.1          0          1          1          2
         4        999 13370012.5          0          0          0          0
         4            13370012.5          0          1          1          2
                  999 94504520.8          1          0          2          1
                      94504520.8          1          1          3          3
```

HEXTORAW

構文

目的

HEXTORAWは、CHAR、VARCHAR2、NCHARまたはNVARCHAR2データ型で16進数を含むcharをRAW値に変換します。

このファンクションは、CLOBデータを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用してCLOBを引数として渡すことはできます。

関連項目:

詳細は、「データ型の比較規則」を参照してください。

例

次の例では、RAW列を含む簡単な表を作成し、RAWに変換された16進数値を挿入します。

```
CREATE TABLE test (raw_col RAW(10));
INSERT INTO test VALUES (HEXTORAW('7D'));
```

次の例では、16進数をRAW値に変換し、そのRAW値をVARCHAR2にキャストします。

```
SELECT UTL_RAW.CAST_TO_VARCHAR2(HEXTORAW('4041424344'))
  FROM DUAL;
UTL_RAW.CAST_TO_VARCHAR2(HEXTORAW('4041424344'))
------------------------------------------------
@ABCD
```

関連項目:

「RAWデータ型とLONG RAWデータ型」および「RAWTOHEX」を参照してください。

INITCAP

構文

目的

INITCAPは、各単語の最初の文字を大文字、残りの文字を小文字にしてcharを戻します。単語は空白または英数字以外の文字で区切ります。

charは、CHAR、VARCHAR2、NCHARまたはNVARCHAR2データ型です。戻り値は、charと同じデータ型です。データベースは、基礎となる文字セットに対して定義したバイナリ・マッピングに基づいて先頭文字の形式を設定します。大文字と小文字の区別については、「NLS_INITCAP」を参照してください。

このファンクションは、CLOBデータを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用してCLOBを引数として渡すことはできます。

関連項目:

- 詳細は、「データ型の比較規則」を参照してください。
- INITCAPの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

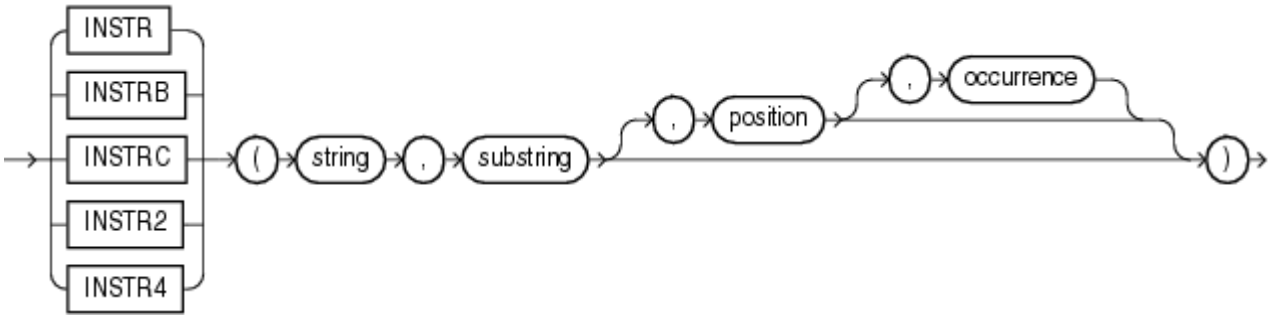
例

次の例では、文字列にある各単語の最初の文字を大文字にします。

```
SELECT INITCAP('the soap') "Capitals"
  FROM DUAL;
Capitals
---------
The Soap
```

INSTR

構文



目的

INSTRファンクションは、stringのsubstringを検索します。検索操作では、一致する値が見つかるか、または検索するサブストリングがなくなるまで、substring引数と、stringに含まれる同じ長さのサブストリングを比較して等しいものを検索します。stringの中で逐次比較される各サブストリングは、その前に比較したサブストリングの最初の文字の1つ右(順方向検索)または左(逆方向検索)から始まります。substringと等しいサブストリングが見つかった場合は、そのサブストリングの先頭文字の位置を示す整数が戻されます。該当するサブストリングが見つからない場合は0(ゼロ)が戻されます。

- positionは、Oracle Databaseが検索を開始するstringの文字の位置を示す0(ゼロ)以外の整数です。つまり、substringと比較する最初のサブストリングの先頭文字の位置です。positionが負の場合、Oracleはstringの終わりから逆方向にカウントし、結果位置から逆方向に検索します。
- occurrenceは、stringの中で何番目に現れるsubstringを検索するかを示す整数です。occurrenceの値は正である必要があります。occurrenceが1より大きい場合は最初の一致では位置を戻さずに、occurrence番目の一致が見つかるまで、前述した方法でstringに含まれるサブストリングの逐次比較を続行します。

INSTRは、文字列の先頭文字の位置を1として、入力文字セットによって定義された文字の位置を受け入れ、それに該当する位置を戻します。INSTRBには、文字でなくバイトを使用します。INSTRCは、完全なUnicodeキャラクタを使用します。INSTR2は、UCS2コードポイントを使用します。INSTR4には、UCS4コード・ポイントを使用します。

stringは、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBデータ型です。例外は、INSTRC、INSTR2およびINSTR4であり、これらでは、stringをCLOBおよびNCLOBのいずれにもできません。

substringは、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBデータ型です。

戻り値のデータ型はNUMBERです。

positionおよびoccurrenceは、NUMBERデータ型か、または暗黙的にNUMBERに変換可能な任意のデータ型で、整数に変換できる必要があります。positionおよびoccurrenceのデフォルト値は1です。この場合、Oracleはstringの最初の文字から検索を開始します。検索対象はsubstringが最初に現れる位置です。戻り値は、positionの値にかかわらず、stringの先頭を基準にします。

関連項目:

- 文字長の詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。
- 文字長の詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。

- 暗黙的な変換の詳細は、表2-8を参照してください。
- substring引数とstringのサブストリングを比較するためにINSTRファンクションで使用される照合を定義する照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、文字列CORPORATE FLOORで文字列ORの検索を3番目の文字から開始します。文字列CORPORATE FLOORで2回目に現れるORの開始位置を戻します。

```
SELECT INSTR('CORPORATE FLOOR','OR', 3, 2) "Instring"
  FROM DUAL;

  Instring
----------
        14
```

次の例では、最後の文字から、最後から3番目の文字まで、つまりFLOORの最初のOまで逆方向にカウントします。次に、2回目に現れる「OR」を逆方向に検索し、2回目に現れるこの文字列が、検索文字列の2番目の文字で始まることを認識します。

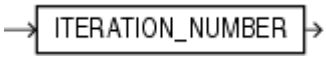
```
SELECT INSTR('CORPORATE FLOOR','OR', -3, 2) "Reversed Instring"
  FROM DUAL;

Reversed Instring
-----------------
                2
```

次の例では、データベース文字セットがダブルバイトの場合を想定しています。

```
SELECT INSTRB('CORPORATE FLOOR','OR',5,2) "Instring in bytes"
  FROM DUAL;
Instring in bytes
-----------------
               27
```

ITERATION_NUMBER

構文



目的

ITERATION_NUMBERファンクションは、SELECT文のmodel_clauseにのみ指定でき、model_rules_clauseにITERATE(number)が指定されている場合にのみ使用できます。このファンクションは、モデル・ルールに従って完了した反復を示す整数を戻します。ITERATION_NUMBERファンクションは、最初の反復では0(ゼロ)を戻します。2回目以降の各反復では、ITERATION_NUMBERファンクションは、iteration_numberに1を足した整数を戻します。

関連項目:

構文およびセマンティクスの詳細は、「model_clause」および「モデル式」を参照してください。

例

次の例では、1998年および1999年のマウス・パッドの売上を、それぞれ2001年および2002年のマウス・パッドの売上に割り当てます。

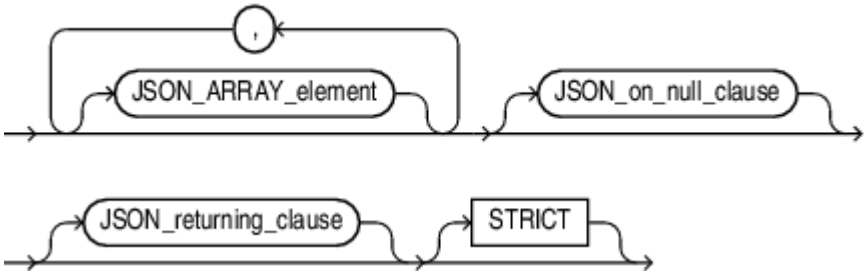
```
SELECT country, prod, year, s
  FROM sales_view_ref
  MODEL
    PARTITION BY (country)
    DIMENSION BY (prod, year)
    MEASURES (sale s)
    IGNORE NAV
    UNIQUE DIMENSION
    RULES UPSERT SEQUENTIAL ORDER ITERATE(2)
      (
        s['Mouse Pad', 2001 + ITERATION_NUMBER] =
          s['Mouse Pad', 1998 + ITERATION_NUMBER]
      )
  ORDER BY country, prod, year;
COUNTRY       PROD                                      YEAR          S
----------    ----------------------------------      --------   ---------
France        Mouse Pad                                   1998     2509.42
France        Mouse Pad                                   1999     3678.69
France        Mouse Pad                                   2000     3000.72
France        Mouse Pad                                   2001     2509.42
France        Mouse Pad                                   2002     3678.69
France        Standard Mouse                              1998     2390.83
France        Standard Mouse                              1999     2280.45
France        Standard Mouse                              2000     1274.31
France        Standard Mouse                              2001     2164.54
Germany       Mouse Pad                                   1998     5827.87
Germany       Mouse Pad                                   1999     8346.44
Germany       Mouse Pad                                   2000     7375.46
Germany       Mouse Pad                                   2001     5827.87
Germany       Mouse Pad                                   2002     8346.44
Germany       Standard Mouse                              1998     7116.11
Germany       Standard Mouse                              1999     6263.14
Germany       Standard Mouse                              2000     2637.31
Germany       Standard Mouse                              2001     6456.13

18 rows selected.
```

この例では、ビュー`sales_view_ref`が必要です。このビューを作成する方法は、「MODEL句：例」を参照してください。

JSON_ARRAY

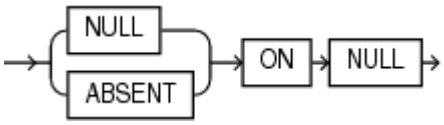
構文

JSON_ARRAY_content

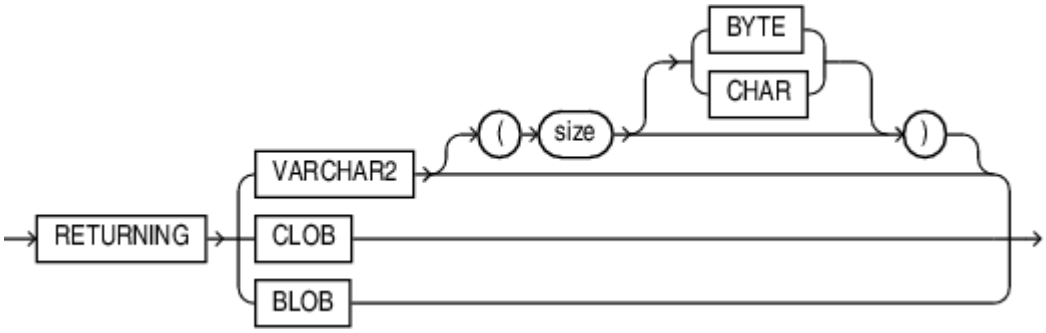


JSON_ARRAY_element

JSON_on_null_clause::=



JSON_returning_clause::=



目的

SQL/JSONファンクション JSON_ARRAYは、一連のSQLスカラー式、または1つのコレクション・タイプ・インスタンス、 VARRAYまたはNESTED TABLEを入力として取ります。

それぞれの式をJSON値に変換し、そのJSON値を含むJSON配列を返します。

ADTにコレクションのメンバーがある場合、型マッピングを実行すると、そのADTに対応するJSONオブジェクトと、コレクション・メンバーのネストしたJSON配列が作成されます。

コレクションにADTインスタンスが含まれている場合は、型マッピングによってJSONオブジェクトのJSON配列が作成されます。

JSON_ARRAY_content

JSON_ARRAYファンクションの入力を定義するとき、この句を使用します。

JSON_ARRAY_element

- expr

  exprには、JSONオブジェクト、JSON配列、数値リテラル、テキスト・リテラル、日付、タイムスタンプ、NULLのいずれかに評価される任意のSQL式を指定できます。このファンクションは、数値リテラルをJSON数値に、テキスト・リテラルをJSON文字値に変換します。日付およびタイムスタンプのデータ型は、生成されるJSONオブジェクトまたは配列に、ISO8601日付書式に従ったJSON文字列として出力されます。

- format_clause

  FORMAT JSONを指定すると、入力文字列がJSONであることを示すため、出力で引用符に囲まれません。

JSON_on_null_clause

この句を使用して、exprがNULLと評価される場合のこのファンクションの動作を指定します。

- NULL ON NULL - この句を指定した場合は、JSONのNULL値が戻されます。
- ABSENT ON NULL - この句を指定した場合は、JSON配列からこの値が省略されます。これはデフォルトです。

JSON_returning_clause

この句を使用して、戻り値のタイプを指定します。次のいずれかです。

- バイト数または文字数としてサイズを指定するVARCHAR2。デフォルトは、バイトです。この句を省略するか、size値を指定せずにこの句を指定した場合、JSON_ARRAYはVARCHAR2(4000)型の文字列を戻します。詳細は、「VARCHAR2データ型」を参照してください。SQLでVARCHAR2データ型を指定する場合、サイズを指定する必要があります。ただし、JSON_returning_clauseではサイズを省略できます。
- シングルバイト文字またはマルチバイト文字を含むキャラクタ・ラージ・オブジェクトを戻すCLOB。
- AL32UTF8文字セットのバイナリ・ラージ・オブジェクトを戻すBLOB。

STRICT

STRICT句を指定して、JSON生成ファンクションの出力が正しいJSONであることを確認します。チェックに失敗すると、構文エラーが発生します。

例については、JSON_OBJECTを参照してください。

例

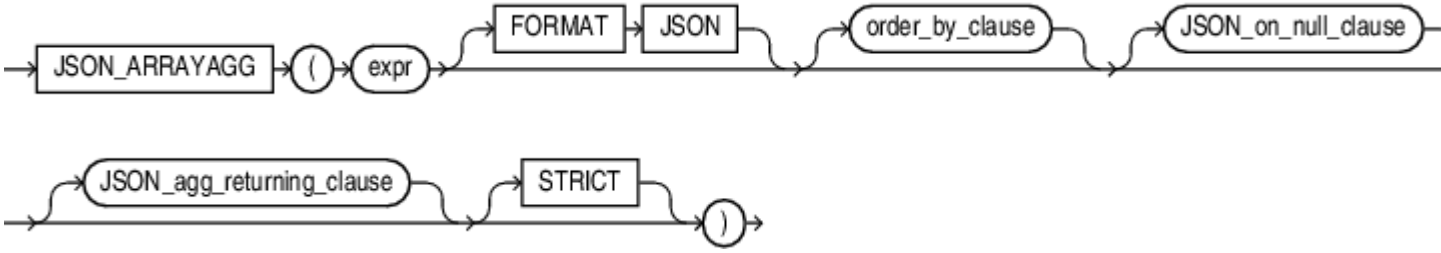
次の例では、JSONオブジェクト、JSON配列、数値リテラル、テキスト・リテラルおよびNULLからJSON配列を構成します。

```
SELECT JSON_ARRAY (
    JSON_OBJECT('percentage' VALUE .50),
    JSON_ARRAY(1,2,3),
    100,
    'California',
    null
    NULL ON NULL
    ) "JSON Array Example"
  FROM DUAL;

JSON Array Example
```

```
--------------------------------------------------------------------------------
[{"percentage":0.5},[1,2,3],100,"California",null]
```

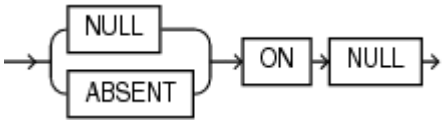
JSON_ARRAYAGG

構文

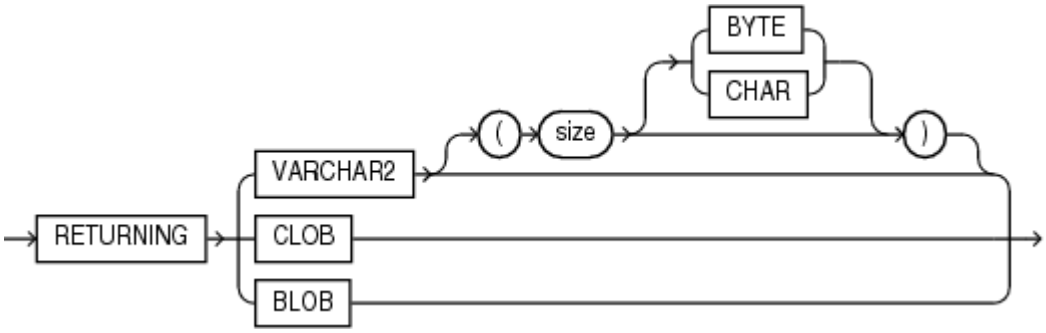


(この句の構文の詳細は、このドキュメントの「SELECT」の「order_by_clause::=」を参照)

JSON_on_null_clause::=



JSON_agg_returning_clause::=



目的

SQL/JSONファンクションJSON_ARRAYAGGは集計ファンクションです。これは、その入力として、SQL式の列を取得し、各式をJSON値に変換して、これらのJSON値を含む単一のJSON配列を戻します。

expr

exprでは、JSONオブジェクト、JSON配列、数値リテラル、テキスト・リテラルまたはNULLと評価される任意のSQL式を指定できます。このファンクションは、数値リテラルをJSON数値に、テキスト・リテラルをJSON文字値に変換します。

FORMAT JSON

このオプション句を使用すると、入力文字列がJSONであることを示すため、出力で引用符に囲まれません。

order_by_clause

この句により、文で戻されるJSON配列内のJSON値を順序付けることができます。この句のセマンティクスの詳細は、このドキュメントの「SELECT」の「order_by_clause」を参照してください。

JSON_on_null_clause

この句を使用して、exprがNULLと評価される場合のこのファンクションの動作を指定します。

- NULL ON NULL - この句を指定した場合は、JSONのNULL値が戻されます。
- ABSENT ON NULL - この句を指定した場合は、JSON配列からこの値が省略されます。これはデフォルトです。

JSON_agg_returning_clause

この句を使用して、このファンクションで戻される文字列のデータ型を指定します。次のデータ型を指定できます。

- VARCHAR2[(size [BYTE,CHAR])]

  SQLでVARCHAR2データ型を指定する場合、サイズを指定する必要があります。ただし、この句はサイズを省略できます。
- CLOB

この句を省略するか、VARCHAR2を指定してsize値を省略した場合、JSON_ARRAYAGGはVARCHAR2(4000)型の文字列を戻します。

前述のデータ型の詳細は、「データ型」を参照してください。

STRICT

STRICT句を指定して、JSON生成ファンクションの出力が正しいJSONであることを確認します。チェックに失敗すると、構文エラーが発生します。

例については、JSON_OBJECTを参照してください。

WITH UNIQUE KEYS

WITH UNIQUE KEYSを指定すると、生成されたJSONオブジェクトが一意キーを持つことが保証されます。

例

次の文では、ID番号を含む表id_tableを作成します。

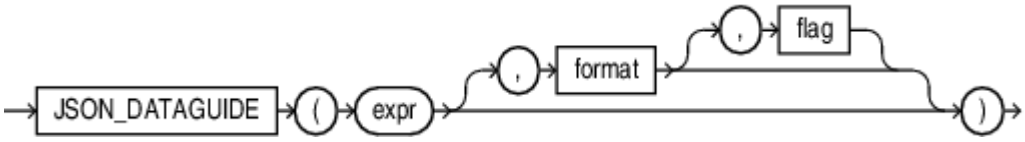
```
CREATE TABLE id_table (id NUMBER);
INSERT INTO id_table VALUES(624);
INSERT INTO id_table VALUES(null);
INSERT INTO id_table VALUES(925);
INSERT INTO id_table VALUES(585);
```

次の例では、表id_tableのID番号からJSON配列を構成します。

```
SELECT JSON_ARRAYAGG(id ORDER BY id RETURNING VARCHAR2(100)) ID_NUMBERS
  FROM id_table;
ID_NUMBERS
-------------
[585,624,925]
```

JSON_DATAGUIDE

構文



目的

集計ファンクションJSON_DATAGUIDEは、表のJSONデータの列を入力として取り、データ・ガイドをCLOBとして戻します。列の各行は、JSONドキュメントと呼ばれます。このファンクションは、列のJSONドキュメントごとに、そのJSONドキュメントのフラット・データ・ガイドを含むCLOB値を戻します。

JSON_DATAGUIDEで、GeoJSONタイプを検出できます。

expr

exprは、JSONオブジェクトまたはJSON配列に評価されるSQL式です。

書式オプション

返されるデータ・ガイドの書式を指定するには、書式オプションを使用します。次の値のうち1つを設定する必要があります。

- フラット構造の場合はdbms_json.format_flat。
- 階層構造の場合はdbms_json.format_hierarchical。

フラグのオプション

flagは、次の値を持つことができます。

- 戻されるデータ・ガイドを、適切なインデントで読みやすくするには、DBMS_JSON.PRETTYを指定します。
- データ・ガイドでGeoJSONタイプを自動的に検出するには、DBMS_JSON.GEOJSONを指定します。
- データ・ガイドでGeoJSONタイプを自動的に検出し、戻されたデータ・ガイドを読みやすくするには、DBMS_JSON.GEOJSON+DBMS_JSON.PRETTYを指定します。

データ・ガイドによって作成されたビューアには、sdo_geometryタイプに対応する列があります。

JSON_DATAGUIDEの制限事項

シャード・カタログ・サーバーではこのファンクションは実行できません。

関連項目:

データ・ガイドの詳細は、Oracle Database JSON開発者ガイドを参照してください。

例

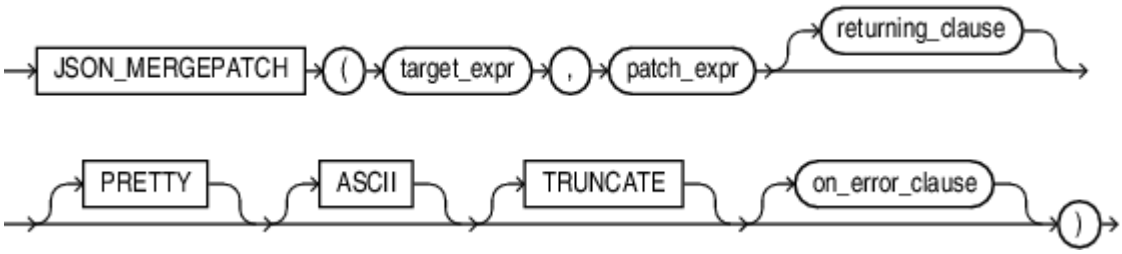
次の例では、「JSONドキュメントを含む表の作成: 例」で作成されるj_purchaseorder表を使用します。この表には、po_documentと呼ばれるJSONデータの列が含まれます。この例では、列po_documentのJSONドキュメントごとにフラット・データ・ガイドを戻します。

```
SELECT EXTRACT(YEAR FROM date_loaded) YEAR,
       JSON_DATAGUIDE(po_document) "DATA GUIDE"
```

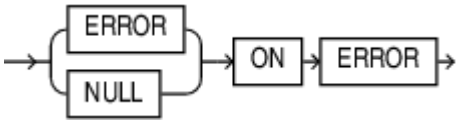
```
  FROM j_purchaseorder
  GROUP BY extract(YEAR FROM date_loaded)
  ORDER BY extract(YEAR FROM date_loaded) DESC;
YEAR DATA GUIDE
---- ----------------------------------------
2016 [
       {
         "o:path" : "$.PO_ID",
         "type" : "number",
         "o:length" : 4
       },
       {
         "o:path" : "$.PO_Ref",
         "type" : "string",
         "o:length" : 16
       },
       {
         "o:path" : "$.PO_Items",
         "type" : "array",
         "o:length" : 64
       },
       {
         "o:path" : "$.PO_Items.Part_No",
         "type" : "number",
         "o:length" : 16
       },
       {
         "o:path" : "$.PO_Items.Item_Quantity",
         "type" : "number",
         "o:length" : 2
       }
     ]
. . .
```

JSON_MERGEPATCH

構文



on_error_clause::=



目的

JSON_MERGEPATCHは標準化された機能のひとつで、マージ・パッチと呼ばれるJSONドキュメントを使用してターゲットJSONドキュメントを変更します。この機能についてはRFC 7396で説明されています。

target_exprは、JSON値のターゲット・ドキュメントに評価されます。

patch_exprは、JSON値のパッチ・ドキュメントに評価されます。

JSON_MERGEPATCHは、ターゲット・ドキュメントと照らしてパッチ・ドキュメントを評価し、その結果をドキュメントとして生成します。ターゲットまたはパッチ・ドキュメントがNULLの場合、結果もNULLです。

JSON_query_returning_clauseには、演算子の戻り型を指定します。デフォルトの戻り型はVARCHAR2(4000)です。

PRETTYキーワードは、結果を人が読める形式にするよう指定します。

ASCIIキーワードは、非ASCII文字を、JSONのエスケープ・シーケンスで出力することを指定します。

TRUNCATEキーワードは、最終的なドキュメントを特定の戻り型に適するように切り捨てるよう指定します。

on_error_clause (オプション)は、ターゲットおよびパッチ・ドキュメントの処理中に発生するエラーの扱い方を制御します。

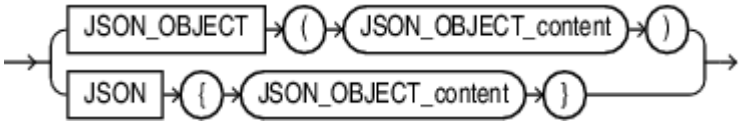
- NULL ON ERROR - エラーが発生した場合にnullを戻します。これはデフォルトです。
- ERROR ON ERROR - エラーが発生した場合に適切なOracleエラーを戻します。

関連項目:

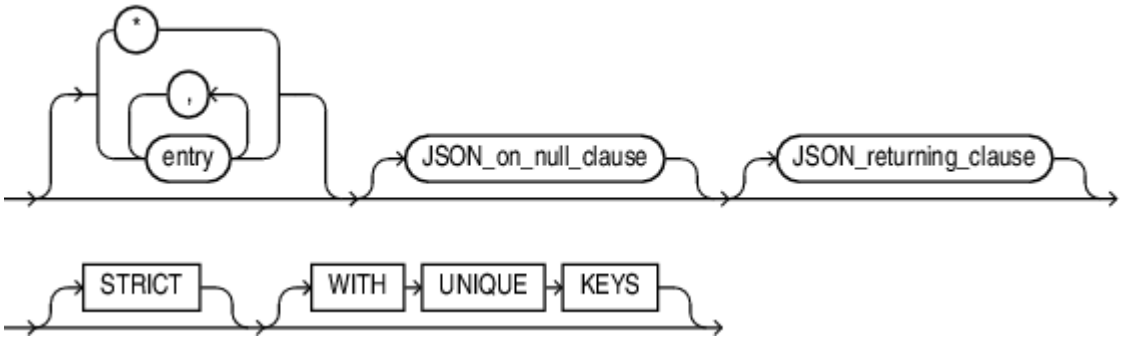
- RFC 7396 JSONマージ・パッチ
- JSONマージ・パッチによるドキュメントの更新

JSON_OBJECT

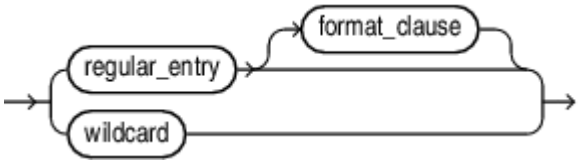
構文



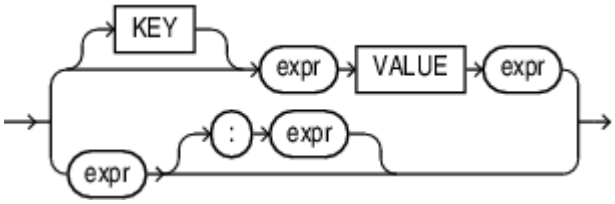
json_object_content::=



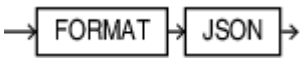
entry::=



regular_entry::=

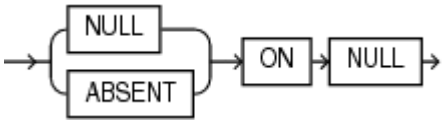


format_clause::=

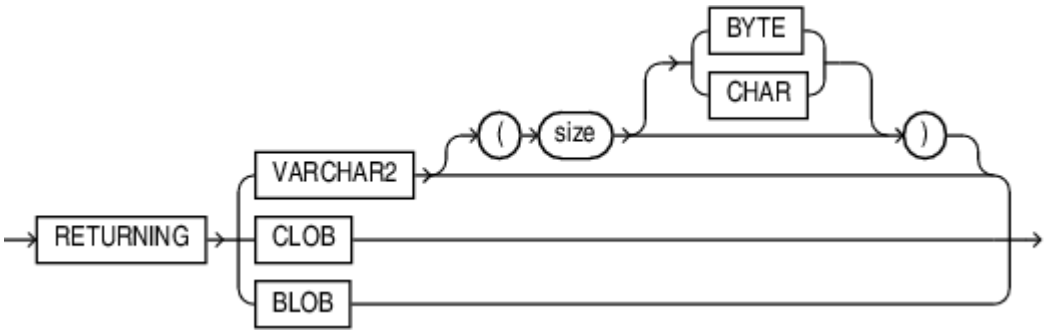


wildcard::=

JSON_on_null_clause::=



JSON_returning_clause::=



目的

SQL/JSONファンクションJSON_OBJECTは、キー/値ペアのシーケンスか、オブジェクト・タイプ・インスタンスのどちらかを入力に取ります。コレクション・タイプをJSON_OBJECTに渡すことはできません。

これは、それらのキーと値のペアごとのオブジェクト・メンバーを含むJSONオブジェクトを戻します。

エントリ

regular_entry: プロパティ・キー/ペアを指定するには、この句を使用します。

regular_entry

- KEYはオプションで、セマンティクスを明確にするために使用されます。
- オプションのexprを使用して、大/小文字を区別するテキスト・リテラルとしてプロパティ・キー名を指定します。
- exprを使用して、プロパティ値を指定します。exprには、SQL数値リテラル、テキスト・リテラル、日付、タイムスタンプのいずれかに評価される任意の式を指定できます。日付およびタイムスタンプのデータ型は、生成されるJSONオブジェクトまたは配列に、ISO日付書式に従ったJSON文字列として出力されます。exprが数値リテラルと評価される場合は結果のプロパティ値がJSON数値になり、そうでない場合は結果のプロパティ値が二重引用符で囲まれた大/小文字を区別するJSON文字列値になります。

  複数JSON_OBJECTエントリの区切りには、コロンを使用できます。

  例

```
SELECT JSON_OBJECT(
'name' : first_name || ' ' || last_name,
'email' : email,
'phone' : phone_number,
'hire_date' : hire_date
)
FROM employees
WHERE employee_id = 140;
```

format_clause

入力式の後にFORMAT JSONを指定すると、結果の値がJSONデータを表し、出力で引用符で囲まれないことを宣言できます。

ワイルドカード

ワイルドカード・エントリは複数の列を選択し、*、table.*、view.*、t_alias.*などの形を指定できます。問合せの列をすべて明示的に指定せず、表、副問合せ、ビューのすべての列をJSONオブジェクトにマップするには、ワイルドカード・エントリを使用します。ここでは、select_listで直接使用するときと同じようにワイルドカード・エントリを使用しています。

例1

得られるJSONオブジェクトでは、キー名が対応する列の名前に当たります。

```
SELECT JSON_OBJECT(*)
FROM employees
WHERE employee_id = 140;
```

出力 1

```
{"EMPLOYEE_ID":140,"FIRST_NAME":"Joshua","LAST_NAME":"Patel","EMAIL":"JPAT
EL","PHONE_NUMBER":"650.121.1834","HIRE_DATE":"2006-04-
06T00:00:00","JOB_ID":"ST_CLERK","SALARY":2500,"COMMISSION_PCT":null,"MAN
AGER_ID":123,"DEPARTMENT_ID":50}
```

例2

この問合せは、結合問合せの特定の表から列を選択します。

```
SELECT JSON_OBJECT('NAME' VALUE first_name, d.*)
FROM employees e, departments d
WHERE e.department_id = d.department_id
AND e.employee_id =140
```

例3

この問合せは、部門名を1つのJSON配列値に変換します。

```
SELECT JSON_ARRAYAGG(JSON_OBJECT(*))
FROM departments
```

JSON_on_null_clause

この句を使用して、exprがNULLと評価される場合のこのファンクションの動作を指定します。

- NULL ON NULL - NULL ON NULLが指定されている場合は、JSON NULL値は、指定したキーの値として使用されます。

  ```
  SELECT JSON_OBJECT('key1' VALUE NULL) evaluates to  {"key1" : null}
  ```

- ABSENT ON NULL - この句を指定した場合は、JSONオブジェクトからプロパティのキーと値のペアが省略されます。

JSON_returning_clause

この句を使用して、戻り値のタイプを指定します。次のいずれかです。

- バイト数または文字数としてサイズを指定するVARCHAR2。デフォルトは、バイトです。この句を省略するか、size値を指定せずにこの句を指定した場合、JSON_ARRAYはVARCHAR2(4000)型の文字列を戻します。詳細は、「VARCHAR2データ型」を参照してください。SQLでVARCHAR2データ型を指定する場合、サイズを指定する必要があります。ただし、JSON_returning_clauseではサイズを省略できます。

- シングルバイト文字またはマルチバイト文字を含むキャラクタ・ラージ・オブジェクトを戻すCLOB。
- AL32UTF8文字セットのバイナリ・ラージ・オブジェクトを戻すBLOB。
- WITH TYPENAME

STRICT

STRICT句を指定して、JSON生成ファンクションの出力が正しいJSONであることを確認します。チェックに失敗すると、構文エラーが発生します。

例1: FORMAT JSONが使用されていないため、出力文字列は引用符内に表示されます

```
SELECT JSON_OBJECT ('name' value 'Foo') FROM DUAL
Output:
JSON_OBJECT('NAME'VALUE'FOO'FORMATJSON)
-----------------------------------------------
{"name":"Foo"}
```

例2: FORMAT JSONが使用されている場合、出力文字列は引用符で囲まれません。

```
SELECT JSON_OBJECT ('name' value 'Foo' FORMAT JSON ) FROM DUAL
Output:
JSON_OBJECT('NAME'VALUE'FOO'FORMATJSON)
-----------------------------------------------
{"name":Foo}
```

例3: FORMAT JSON STRICTが使用されている場合、JSON構文エラーになります。

```
SELECT JSON_OBJECT ('name' value 'Foo' FORMAT JSON STRICT ) FROM DUAL
Output:
ORA-40441: JSON syntax error
```

WITH UNIQUE KEYS

WITH UNIQUE KEYSを指定すると、生成されたJSONオブジェクトが一意キーを持つことが保証されます。

例

次の例では、それぞれが2つのキーと値のペアを含むJSONオブジェクトを戻します。

```
SELECT JSON_OBJECT (
    KEY 'deptno' VALUE d.department_id,
    KEY 'deptname' VALUE d.department_name
    ) "Department Objects"
  FROM departments d
  ORDER BY d.department_id;
Department Objects
--------------------------------------
{"deptno":10,"deptname":"Administration"}
{"deptno":20,"deptname":"Marketing"}
{"deptno":30,"deptname":"Purchasing"}
{"deptno":40,"deptname":"Human Resources"}
{"deptno":50,"deptname":"Shipping"}
. . .
```

JSON_OBJECT列エントリ

場合によっては、キー値の式で列名を繰り返さないように、JSONオブジェクトのキー名を表の列と照合したいことがあります。たとえば:

```
SELECT JSON_OBJECT(
'first_name' VALUE first_name,
'last_name' VALUE last_name,
```

```
'email' VALUE email,
'hire_date' VALUE hire_date
)
FROM employees
WHERE employee_id = 140;
{"first_name":"Joshua","last_name":"Patel","email":"JPATEL","hire_date":"2006-04-
06T00:00:00"}
```

そうした場合に簡単な方法があり、1つの列値を入力として指定すると、対応するオブジェクトのエントリ・キーが、列の名前から推論されます。たとえば:

```
SELECT JSON_OBJECT(first_name, last_name, email, hire_date)
FROM employees
WHERE employee_id = 140;
{"first_name":"Joshua","last_name":"Patel","email":"JPATEL","hire_date":"2006-04-
06T00:00:00"}
```

列名には、引用符付き、または引用符なしの識別子を使用できます。引用符なしの識別子を使用する場合、問合せのように識別子の大文字小文字が区別され、対応するオブジェクト・キー値の生成に使用されます。ただし、列値を参照する目的上、識別子は大文字小文字が区別されます。たとえば:

```
SELECT JSON_OBJECT(eMail)
FROM employees
WHERE employee_id = 140
{"eMail":"JPATEL"}
```

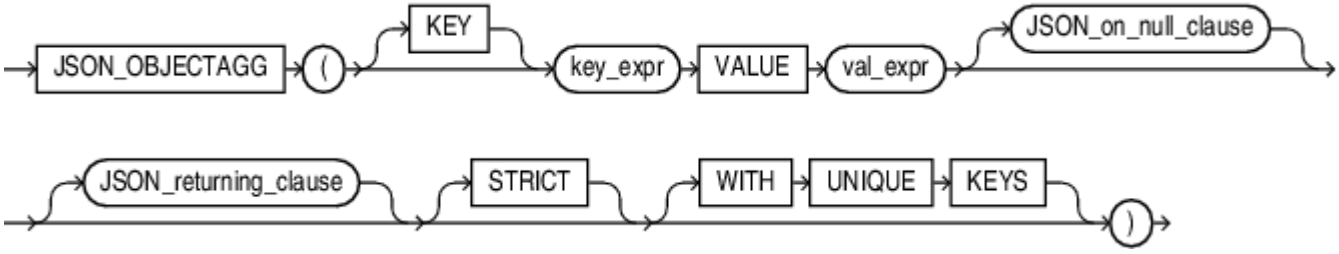
列名で入力したままなので、「M」が大文字です。

関連項目:

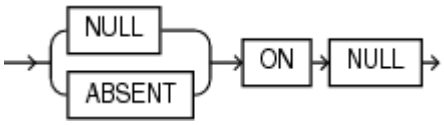
SQLを使用したJSONデータの生成

JSON_OBJECTAGG

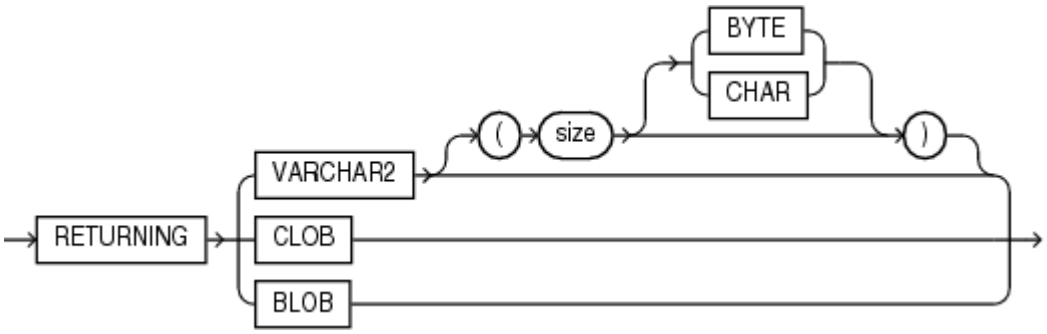
構文



JSON_on_null_clause::=



JSON_agg_returning_clause::=



目的

SQL/JSONファンクションJSON_OBJECTAGGは集計ファンクションです。これは、その入力として、プロパティのキーと値のペアを取得します。通常、プロパティ・キー、プロパティ値またはその両方はSQL式の列です。このファンクションは、キーと値のペアごとのオブジェクト・メンバーを構成し、それらのオブジェクト・メンバーを含む単一のJSONオブジェクトを戻します。

[KEY] string VALUE expr

この句を使用して、プロパティのキーと値のペアを指定します。

- KEYはオプションで、セマンティクスを明確にするために使用されます。
- stringを使用して、大/小文字を区別するテキスト・リテラルとしてプロパティ・キー名を指定します。
- exprを使用して、プロパティ値を指定します。exprには、SQL数値リテラル、テキスト・リテラル、日付、タイムスタンプのいずれかに評価される任意の式を指定できます。日付およびタイムスタンプのデータ型は、生成されるJSONオブジェクトまたは配列に、ISO8601日付書式に従ったJSON文字列として出力されます。exprが数値リテラルと評価される場合は結果のプロパティ値がJSON数値になり、そうでない場合は結果のプロパティ値が二重引用符で囲まれた大/小文字を区別するJSON文字列値になります。

FORMAT JSON

このオプション句を使用すると、入力文字列がJSONであることを示すため、出力で引用符に囲まれません。

JSON_on_null_clause

この句を使用して、exprがNULLと評価される場合のこのファンクションの動作を指定します。

- NULL ON NULL - NULL ON NULLが指定されている場合は、JSON NULL値は、指定したキーの値として使用されます。
- ABSENT ON NULL - この句を指定した場合は、JSONオブジェクトからプロパティのキーと値のペアが省略されます。

JSON_agg_returning_clause

この句を使用して、このファンクションで戻される文字列のデータ型を指定します。次のデータ型を指定できます。

- VARCHAR2[(size [BYTE,CHAR])]

  SQLでVARCHAR2データ型を指定する場合、サイズを指定する必要があります。ただし、この句はサイズを省略できます。

- シングルバイト文字またはマルチバイト文字を含むキャラクタ・ラージ・オブジェクトを戻すCLOB。
- AL32UTF8文字セットのバイナリ・ラージ・オブジェクトを戻すBLOB。

この句を省略するか、VARCHAR2を指定してsize値を省略した場合、JSON_OBJECTAGGはVARCHAR2(4000)型の文字列を戻します。

前述のデータ型の詳細は、「データ型」を参照してください。

STRICT

STRICT句を指定して、JSON生成ファンクションの出力が正しいJSONであることを確認します。チェックに失敗すると、構文エラーが発生します。

例については、JSON_OBJECTを参照してください。

WITH UNIQUE KEYS

WITH UNIQUE KEYSを指定すると、生成されたJSONオブジェクトが一意キーを持つことが保証されます。

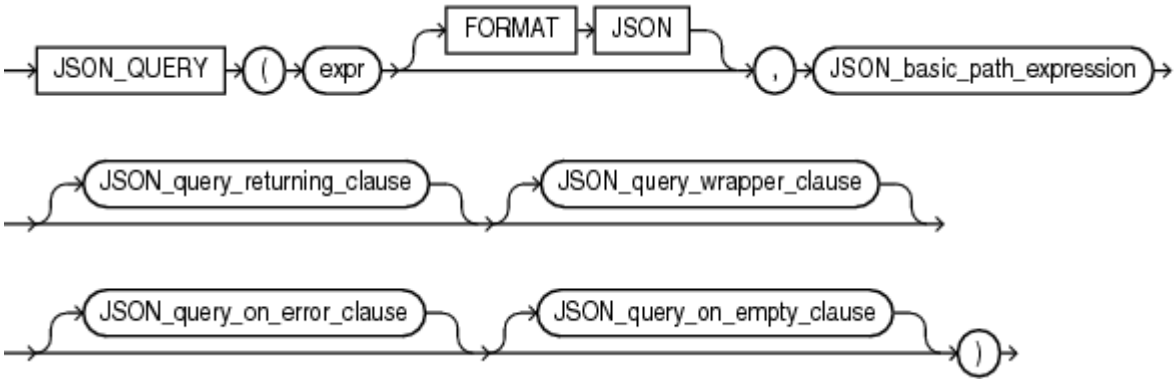
例

次の例では、メンバーに部門名と部門番号が含まれるJSONオブジェクトを構成します。

```
SELECT JSON_OBJECTAGG(KEY department_name VALUE department_id) "Department Numbers"
  FROM departments
  WHERE department_id <= 30;
Department Numbers
--------------------------------------------------
{"Administration":10,"Marketing":20,"Purchasing":30}
```

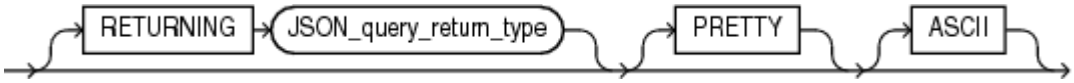
JSON_QUERY

構文

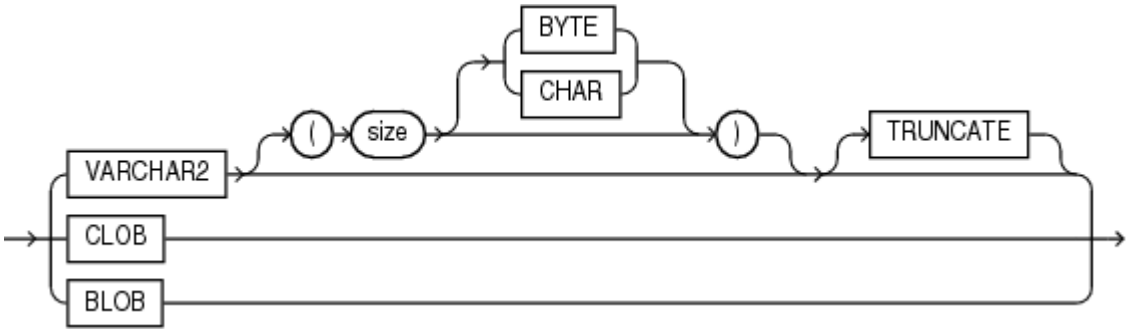


(JSON_basic_path_expression: Oracle Database JSON開発者ガイドを参照)

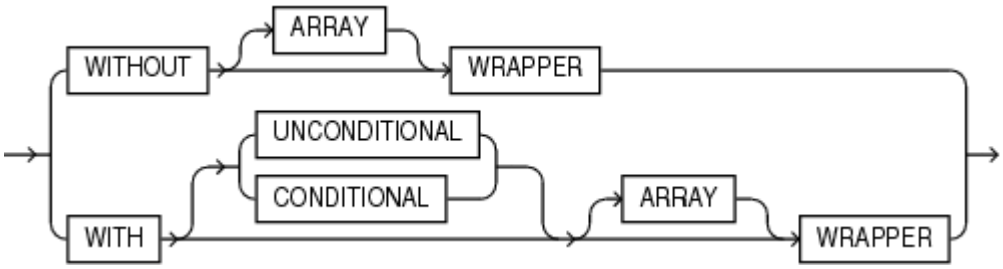
JSON_query_returning_clause::=



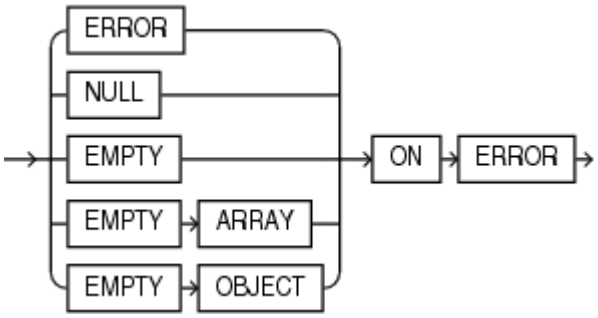
JSON_query_return_type::=



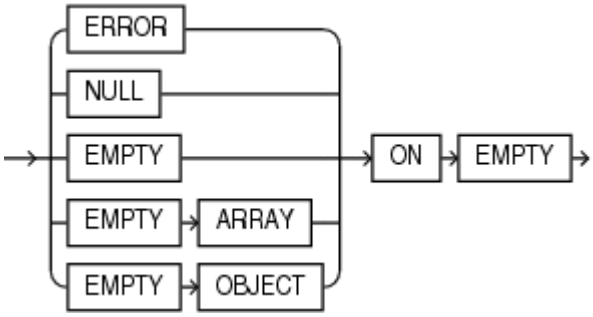
JSON_query_wrapper_clause::=



JSON_query_on_error_clause::=



JSON_query_on_empty_clause::=



目的

JSON_QUERYは、JSONデータから1つ以上の値を選択して返し、それらの値を返します。JSON_QUERYを使用してJSONドキュメントのフラグメントを取得できます。

関連項目:

- JSONデータの問合せ
- JSON_QUERYによって戻される文字値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

expr

exprを使用して、問い合せるJSONデータを指定します。

exprは、SQLデータ型のインスタンス(JSON、VARCHAR2、CLOBまたはBLOBのいずれか)を返すSQL式です。これは、表またはビューの列の値、PL/SQL変数、または適切にキャストされたバインド変数のいずれかになります。

exprがNULLの場合はNULLを戻します。

exprが厳密なまたは緩い構文を使用した整形式のJSONデータのテキスト・リテラルでない場合、このファンクションはデフォルトでnullを戻します。JSON_query_on_error_clauseを使用して、このデフォルトの動作をオーバーライドできます。JSON_query_on_error_clauseを参照してください。

FORMAT JSON

exprがデータ・タイプBLOBの列の場合、FORMAT JSONを指定する必要があります。

JSON_basic_path_expression

この句を使用して、SQL/JSONパス式を指定します。このファンクションはパス式を使用してexprを評価し、パス式と一致する(パス式を満たす)1つ以上のJSON値を確認します。パス式はテキスト・リテラルである必要があります。

JSON_basic_path_expressionのセマンティクスの詳細は、Oracle Database JSON開発者ガイドを参照してください。

JSON_query_returning_clause

この句を使用して、このファンクションで戻される文字列のデータ型および書式を指定します。

RETURNING

RETURNING句を使用して、返されるインスタンスのデータ型(VARCHAR2、CLOBまたはBLOBのいずれか)を指定できます。

デフォルトの戻り型は、すべての入力データ型でVARCHAR2(4000)です。

SQLでVARCHAR2データ型を指定する場合、サイズを指定する必要があります。ただし、この句はサイズを省略できます。この場合、JSON_QUERYはVARCHAR2(4000)型の文字列を戻します。

詳細は、「VARCHAR2データ型」を参照してください。

データ型が戻り文字列を保持できる十分な大きさでない場合、このファンクションはデフォルトでnullを戻します。JSON_query_on_error_clauseを使用して、このデフォルトの動作をオーバーライドできます。JSON_query_on_error_clauseを参照してください。

PRETTY

改行文字とインデントを挿入して戻り文字列を出力整形するには、PRETTYを指定します。

ASCII

標準のASCII Unicodeエスケープ・シーケンスを使用して戻り文字列の非ASCII Unicode文字を自動的にエスケープするには、ASCIIを指定します。

JSON_query_wrapper_clause

この句を使用して、このファンクションが配列ラッパーのパス式と一致する値をラップするかどうか、つまり大カッコ([])で値のシーケンスを囲むかどうかを制御します。

- WITHOUT WRAPPERを指定して、配列ラッパーを省略します。パス式が単一のJSONオブジェクトまたはJSON配列と一致する場合のみ、この句を指定できます。これはデフォルトです。
- WITH WRAPPERを指定して、配列ラッパーを含めます。パス式が単一のスカラー値(JSONオブジェクトまたはJSON配列でない値)または任意の型の複数の値と一致する場合、この句を指定する必要があります。
- WITH UNCONDITIONAL WRAPPER句の指定は、WITH WRAPPER句の指定と同じです。UNCONDITIONALキーワードはセマンティクスを明確にするためのものです。
- パス式が単一のスカラー値または任意の型の複数の値と一致する場合のみ配列ラッパーに含めるには、WITH CONDITIONAL WRAPPERを指定します。パス式が単一のJSONオブジェクトまたはJSON配列と一致する場合、配列ラッパーが省略されます。

ARRAYキーワードはオプションで、意味を明確化するために使用されます。

ファンクションが単一のスカラー値または任意の型の複数の値を戻し、WITH [UNCONDITIONAL | CONDITIONAL] WRAPPERを指定しない場合、ファンクションはデフォルトでnullを戻します。JSON_query_on_error_clauseを使用して、このデフォルトの動作をオーバーライドできます。JSON_query_on_error_clauseを参照してください。

JSON_query_on_error_clause

この句を使用して、次のエラーが発生した場合にこのファンクションで戻される値を指定します。

- exprが厳密なまたは緩いJSON構文を使用した整形式のJSONデータではありません。
- SQL/JSONパス式を使用してJSONデータを評価した場合に一致が見つかりません。JSON_query_on_empty_clauseを指定して、このタイプのエラーに対する動作を上書きできます。
- 戻り値のデータ型が戻り文字列を保持する十分な大きさではありません。
- ファンクションが単一のスカラー値または任意の型の複数の値と一致し、WITH [UNCONDITIONAL | CONDITIONAL] WRAPPER句が指定されていません。

次の句を指定できます。

- NULL ON ERROR - エラーが発生した場合にnullを戻します。これはデフォルトです。
- ERROR ON ERROR - エラーが発生した場合に適切なOracleエラーを戻します。
- EMPTY ON ERROR - この句を指定することは、EMPTY ARRAY ON ERRORを指定することと同じです。
- EMPTY ARRAY ON ERROR - エラーが発生した場合に空のJSON配列([])を戻します。
- EMPTY OBJECT ON ERROR - エラーが発生した場合に空のJSONオブジェクト({})を戻します。

JSON_query_on_empty_clause

この句を使用して、JSONデータがSQL/JSONパス式を使用して評価されるときに一致が見つからない場合にこのファンクションで戻される値を指定します。この句により、JSON_query_on_error_clauseで指定された結果とは異なる、このタイプのエラーに対する結果を指定できます。

次の句を指定できます。

- NULL ON EMPTY - 一致が見つからない場合にNULLを戻します。
- ERROR ON EMPTY - 一致が見つからない場合に適切なOracleエラーを戻します。
- EMPTY ON EMPTY - この句を指定することは、EMPTY ARRAY ON EMPTYを指定することと同じです。
- EMPTY ARRAY ON EMPTY - 一致が見つからない場合に空のJSON配列([])を戻します。
- EMPTY OBJECT ON EMPTY - 一致が見つからない場合に空のJSONオブジェクト({})を戻します。

この句を省略すると、JSON_query_on_error_clauseによって、一致が見つからない場合に戻される値が決まります。

例

次の問合せは、JSONデータの指定された文字列であるコンテキスト項目を戻します。パス式は、配列ラッパーを必要としない単一のJSONオブジェクトと一致します。JSONデータは戻り値で厳密なJSON構文に変換されません。つまり、オブジェクト・プロパティ名が二重引用符で囲まれます。

```
SELECT JSON_QUERY('{a:100, b:200, c:300}', '$') AS value
  FROM DUAL;
VALUE
--------------------------------------------------------------------------------
{"a":100,"b":200,"c":300}
```

次の問合せは、プロパティ名aのメンバーの値を戻します。パス式は、配列ラッパーで囲む必要があるスカラー値と一致します。このため、WITH WRAPPER句が指定されます。

```
SELECT JSON_QUERY('{a:100, b:200, c:300}', '$.a' WITH WRAPPER) AS value
  FROM DUAL;
VALUE
--------------------------------------------------------------------------------
```

```
[100]
```

次の問合せは、すべてのオブジェクト・メンバーの値を戻します。パス式は、配列ラッパーで囲む必要がある複数の値と一致します。このため、WITH WRAPPER句が指定されます。

```
SELECT JSON_QUERY('{a:100, b:200, c:300}', '$.*' WITH WRAPPER) AS value
  FROM DUAL;
VALUE
--------------------------------------------------------------------------------
[100,200,300]
```

次の問合せは、JSONデータの指定された文字列であるコンテキスト項目を戻します。パス式は、配列ラッパーを必要としない単一のJSON配列と一致します。

```
SELECT JSON_QUERY('[0,1,2,3,4]', '$') AS value
  FROM DUAL;
VALUE
--------------------------------------------------------------------------------
[0,1,2,3,4]
```

WITH WRAPPER句が指定されている点を除いて、次の問合せは前の問合せと似ています。このため、JSON配列が配列ラッパーでラップされます。

```
SELECT JSON_QUERY('[0,1,2,3,4]', '$' WITH WRAPPER) AS value
  FROM DUAL;
VALUE
--------------------------------------------------------------------------------
[[0,1,2,3,4]]
```

次の問合せは、JSON配列のすべての要素を戻します。パス式は、配列ラッパーで囲む必要がある複数の値と一致します。このため、WITH WRAPPER句が指定されます。

```
SELECT JSON_QUERY('[0,1,2,3,4]', '$[*]' WITH WRAPPER) AS value
  FROM DUAL;
VALUE
--------------------------------------------------------------------------------
[0,1,2,3,4]
```

次の問合せは、JSON配列の索引0、3から5、7の要素を戻します。パス式は、配列ラッパーで囲む必要がある複数の値と一致します。このため、WITH WRAPPER句が指定されます。

```
SELECT JSON_QUERY('[0,1,2,3,4,5,6,7,8]', '$[0, 3 to 5, 7]' WITH WRAPPER) AS value
  FROM DUAL;
VALUE
--------------------------------------------------------------------------------
[0,3,4,5,7]
```

次の問合せは、JSON配列の4番目の要素を戻します。パス式は、配列ラッパーで囲む必要があるスカラー値と一致します。このため、WITH WRAPPER句が指定されます。

```
SELECT JSON_QUERY('[0,1,2,3,4]', '$[3]' WITH WRAPPER) AS value
  FROM DUAL;
VALUE
--------------------------------------------------------------------------------
[3]
```

次の問合せは、JSON配列の最初の要素を戻します。WITH CONDITIONAL WRAPPER句が指定され、パス式が単一のJSONオブジェクトと一致します。このため、戻される値は配列でラップされます。JSONデータは戻り値で厳密なJSON構文に変換されません。つまり、オブジェクト・プロパティ名が二重引用符で囲まれます。

```
SELECT JSON_QUERY('[{a:100},{b:200},{c:300}]', '$[0]'
       WITH CONDITIONAL WRAPPER) AS value
  FROM DUAL;
VALUE
--------------------------------------------------------------------------------
{"a":100}
```

次の問合せは、JSON配列のすべての要素を戻します。WITH CONDITIONAL WRAPPER句が指定され、パス式が複数のJSONオブジェクトと一致します。このため、戻される値は配列でラップされます。

```
SELECT JSON_QUERY('[{"a":100},{"b":200},{"c":300}]', '$[*]'
       WITH CONDITIONAL WRAPPER) AS value
  FROM DUAL;
VALUE
--------------------------------------------------------------------------------
[{"a":100},{"b":200},{"c":300}]
```

戻される値のデータ型がVARCHAR2(100)である点を除いて、次の問合せは前の問合せと似ています。

```
SELECT JSON_QUERY('[{"a":100},{"b":200},{"c":300}]', '$[*]'
       RETURNING VARCHAR2(100) WITH CONDITIONAL WRAPPER) AS value
  FROM DUAL;
VALUE
--------------------------------------------------------------------------------
[{"a":100},{"b":200},{"c":300}]
```

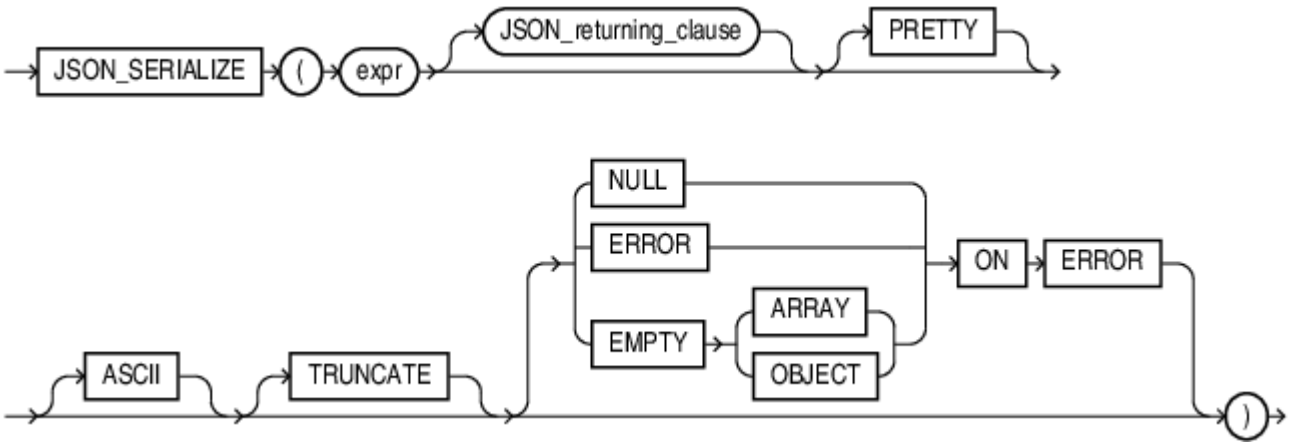
次の問合せは、JSON配列の4番目の要素を戻します。ただし、指定されたJSON配列に結果としてエラーとなる4番目の要素は含まれません。EMPTY ON ERROR句が指定されます。このため、問合せは空のJSON配列を戻します。

```
SELECT JSON_QUERY('[{"a":100},{"b":200},{"c":300}]', '$[3]'
       EMPTY ON ERROR) AS value
  FROM DUAL;
VALUE
--------------------------------------------------------------------------------
[]
```

JSON_SERIALIZE

構文

json_serialize



目的

json_serializeファンクションは、任意のSQLデータ型(VARCHAR2、CLOB、BLOB)のJSONデータを入力に取り、そのテキスト表記を返します。通常は、問合せの結果を変換するために使用します。

json_serializeを使用すると、バイナリのJSONデータをテキスト形式(VARCHAR2またはCLOB)に変換することや、出力整形やASCII Unicode以外の文字のエスケープによってテキストのJSONデータを変換することができます。

expr

exprは入力式です。VARCHAR2、CLOBまたはBLOBのいずれかの型になります。

JSON_returning_clause

JSON_returning_clauseを使用して、戻り型を指定します。戻り型は、VARCHAR2、CLOBまたはBLOBのいずれかにできます。

デフォルトの戻り型はVARCHAR2(4000)です。

戻り型がRAWまたはBLOBの場合は、 UTF8エンコードしたJSONテキストが含まれます。

PRETTY

結果を読みやすい形式にする場合は、PRETTYを指定します。

ASCII

JSONエスケープ・シーケンスを使用して非ASCII文字を出力する場合は、ASCIIを指定します。

TRUNCATE

結果ドキュメントのテキスト出力を、指定した戻り型のバッファに収める場合はTRUNCATEを指定します。

JSON_on_error_clause

JSON_on_error_clauseを指定すると、処理エラーの処理を制御できます。

ERROR ON ERRORがデフォルトです。

EMPTY ON ERRORはサポートされません。

JSON_on_error_clauseを指定してTRUNCATEを使用すると、エラーが発生することなく、戻り型に対して大きすぎる値が、バッファに収まるように切り捨てられます。

例

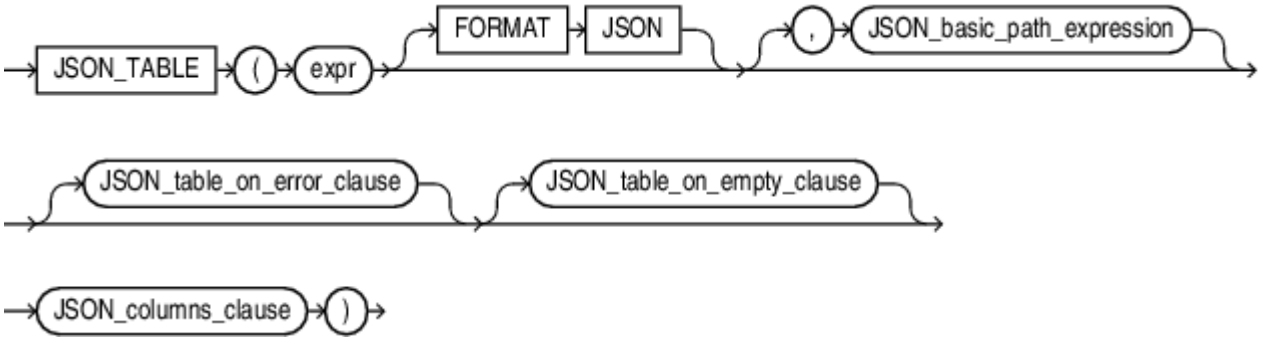
```
SELECT JSON_SERIALIZE ('{a:[1,2,3,4]}' RETURNING VARCHAR2(3) TRUNCATE ERROR ON ERROR)
from dual
--------
{"a
```

関連項目:

Oracle SQLファンクションJSON_SERIALIZE

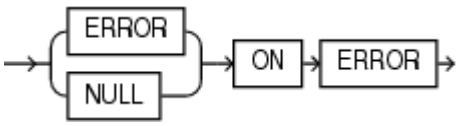
JSON_TABLE

構文

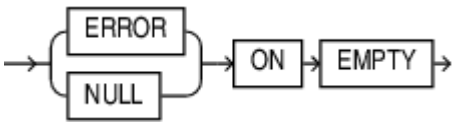


(JSON_basic_path_expression: Oracle Database JSON開発者ガイドを参照、JSON_table_on_error_clause::=、JSON_columns_clause::=)

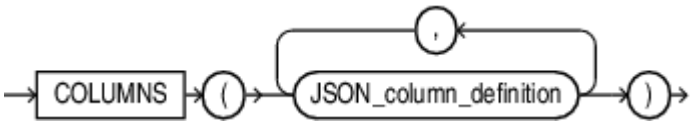
JSON_table_on_error_clause::=



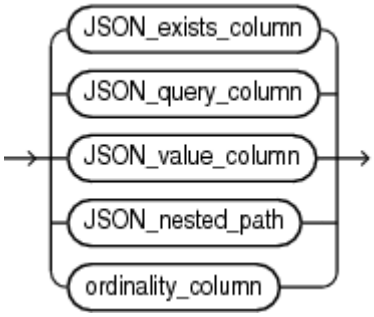
JSON_table_on_empty_clause::=



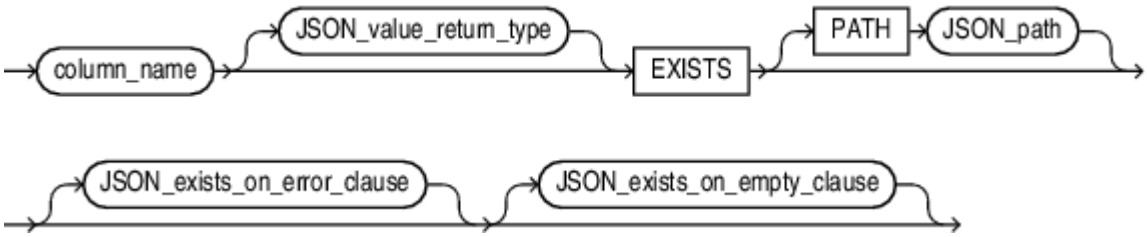
JSON_columns_clause::=



JSON_column_definition::=

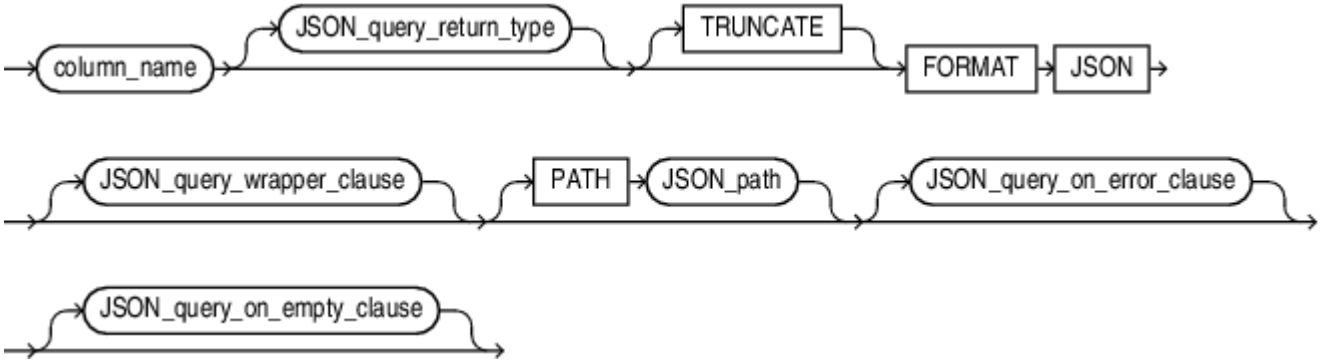


JSON_exists_column::=



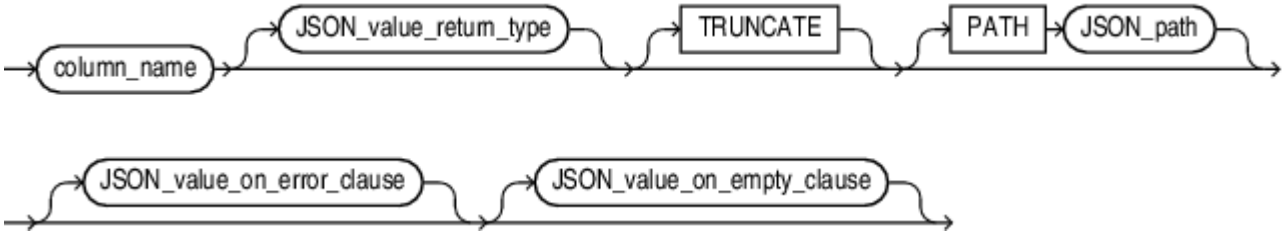
(JSON_value_return_type::=(JSON_VALUEの一部)、JSON_basic_path_expression: Oracle Database JSON開発者ガイドを参照、JSON_exists_on_error_clause::=(JSON_EXISTSの一部))

JSON_query_column::=



(JSON_query_return_type::=、JSON_query_wrapper_clause::=、JSON_query_on_error_clause::=(JSON_QUERYの一部)、JSON_basic_path_expression: Oracle Database JSON開発者ガイドを参照)

JSON_value_column::=



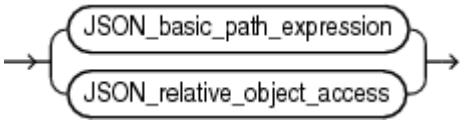
(JSON_value_return_type::=およびJSON_value_on_error_clause::=(JSON_VALUEの一部)、JSON_basic_path_expression: Oracle Database JSON開発者ガイドを参照)

JSON_nested_path::=

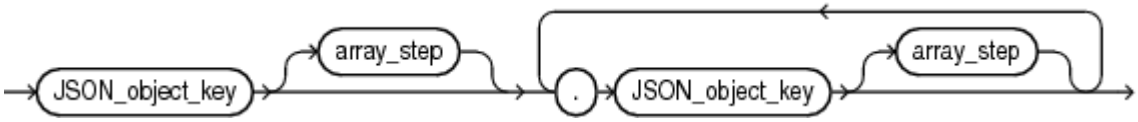
(JSON_basic_path_expression: Oracle Database JSON開発者ガイドを参照、JSON_columns_clause::=)

ordinality_column::=

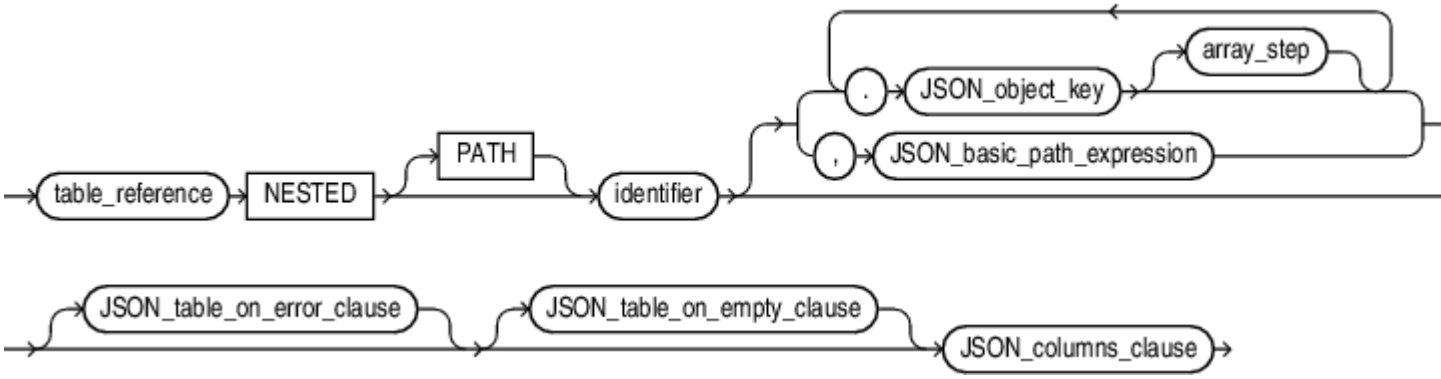
JSON_path ::=



JSON_relative_object_access ::=



nested_clause ::=



目的

SQL/JSONファンクションJSON_TABLEは、JSONデータのリレーショナル・ビューを作成します。JSONデータの評価結果をリレーショナル行および列にマップします。SQLを使用して、このファンクションで戻される結果を仮想リレーショナル表として問い合せることができます。JSON_TABLEの主な目的は、JSON配列内の各オブジェクトのリレーショナル・データの行を作成し、個々のSQL列値としてそのオブジェクト内からJSON値を出力することです。

SELECT文のFROM句にのみ、JSON_TABLEを指定する必要があります。ファンクションは、最初にSQL/JSON行パス式と呼ばれるパス式を、指定されたJSONデータに適用します。行パス式と一致するJSON値は、リレーショナル・データの行を生成するので行ソースと呼ばれます。COLUMNS句は行ソースを評価し、行ソース内の特定のJSON値を確認し、リレーショナル・データの行の個々の列のSQL値としてそれらのJSON値を戻します。

COLUMNS句によって、次の句を使用して別の方法でJSON値を検索できます。

- JSON_exists_column - JSON_EXISTS条件と同じ方法でJSONデータを評価します。つまり、指定されたJSONデータが存在するかどうかを確認し、値'true'または'false'のVARCHAR2列あるいは値1または0のNUMBER列を戻します。
- JSON_query_column - JSON_QUERYファンクションと同じ方法でJSONデータを評価します。つまり、1つ以上の指定されたJSON値を確認し、それらのJSON値を含む文字列の列を戻します。
- JSON_value_column - JSON_VALUEファンクションと同じ方法でJSONデータを評価します。つまり、指定されたスカラーJSON値を確認し、SQL値としてそれらのJSON値の列を戻します。
- JSON_nested_path - ネストされたJSONオブジェクトまたはJSON配列のJSON値を親のオブジェクトまたは配列のJSON値とともに単一の行の個々の列にフラット化できます。この句を再帰的に使用して、ネストされたオブジェクトま

たは配列の複数のレイヤーから単一の行にデータを投影できます。

- `ordinality_column` - 生成された行番号の列を戻します。

列定義句を使用すると、戻されるデータの列ごとに名前を指定できます。SELECTリストやWHERE句などのSELECT文でこれらの列名を参照できます。

関連項目:

JSON_TABLEによって生成される表の、文字データ型の列ごとに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

expr

この句は、評価の対象となるJSONデータを指定するために使用します。exprでは、テキスト・リテラルを評価する式を指定します。exprが列である場合、列のデータ型はVARCHAR2、CLOBまたはBLOBのいずれかである必要があります。exprがNULLの場合はNULLを戻します。

exprが厳密なまたは緩い構文を使用した整形式のJSONデータのテキスト・リテラルでない場合、このファンクションはデフォルトでnullを戻します。`JSON_table_on_error_clause`を使用して、このデフォルトの動作をオーバーライドできます。JSON_table_on_error_clauseを参照してください。

FORMAT JSON

exprがデータ・タイプBLOBの列の場合、FORMAT JSONを指定する必要があります。

PATH

PATH句を使用して、列の内容として使用する行の一部を記述します。PATH句がない場合、パス`'$.<column-name>'`(`<column-name>`は列名)での動作は変更されません。ターゲットであるオブジェクト・フィールド名が列名として暗黙的に取得されます。PATHの完全なセマンティクスについては、『Oracle Database JSON開発者ガイド』を参照してください。

JSON_basic_path_expression

`JSON_basic_path_expression`はテキスト・リテラルです。この句のセマンティクスの詳細は、Oracle Database JSON開発者ガイドを参照してください。

JSON_relative_object_access

単純なドット表記法を有効にするには、この行パス式を指定します。`JSON_relative_object_access`の値は、現在の行アイテムに関連したJSON/パス式として評価されます。

`JSON_object_key`句の詳細は、JSONオブジェクト・アクセス式を参照してください。

JSON_table_on_error_clause

この句を使用して、エラーが発生した場合にこのファンクションで戻される値を指定します。

- `NULL ON ERROR`
  - 入力が整形式のJSONテキストではない場合、エラーが検出されると即時にそれ以降の行が戻されなくなります。JSON_TABLEではストリーミング評価がサポートされているため、入力のエラーの部分が検出される前に行が戻される可能性があることに注意してください。
  - 行パス式の評価時に一致が見つからない場合は、行が返されません。

- すべての列式のデフォルト・エラー動作をNULL ON ERRORに設定します。

- ERROR ON ERROR
  - 入力が整形式のJSON テキストではない場合、エラーが発生します。
  - 行パス式の評価時に一致が見つからない場合は、エラーが発生します。
  - すべての列式のデフォルト・エラー動作をERROR ON ERRORに設定します。

JSON_table_on_empty_clause

この句を使用して、JSONデータがSQL/JSONパス式を使用して評価されるときに一致が見つからない場合にこのファンクションで戻される値を指定します。この句により、JSON_table_on_error_clauseで指定された結果とは異なる、このタイプのエラーに対する結果を指定できます。

次の句を指定できます。

- NULL ON EMPTY - 一致が見つからない場合にNULLを戻します。
- ERROR ON EMPTY - 一致が見つからない場合に適切なOracleエラーを戻します。
- DEFAULT literal ON EMPTY - 一致が見つからない場合にliteralを戻します。literalのデータ型は、このファンクションにより戻される値のデータ型と一致する必要があります。

この句を省略すると、JSON_table_on_error_clauseによって、一致が見つからない場合に戻される値が決まります。

JSON_columns_clause

COLUMNS句を使用して、JSON_TABLEファンクションで戻される仮想リレーショナル表の列を定義します。

列にJSON_VALUEセマンティクスがある場合は、TRUNCATEを指定します。

JSON_exists_column

この句は、JSON_EXISTS条件と同じ方法でJSONデータを評価します。つまり、指定されたJSON値が存在するかどうかを判断します。値'true'または'false'のVARCHAR2列あるいは値1または0のNUMBER列を戻します。

'true'または1の値はJSON値があることを示し、'false'または0の値はJSON値がないことを示します。

JSON_value_return_type句を使用して、戻される列のデータ型を制御できます。この句を省略すると、データ型はVARCHAR2(4000)です。column_nameを使用して、戻される列の名前を指定します。JSON_exists_columnの残りの句は、JSON_EXISTS条件と同じセマンティクスを持ちます。これらの句の詳細は、「JSON_EXISTS条件」を参照してください。「JSON_exists_columnの使用方法：例」の例も参照してください。

JSON_query_column

この句は、JSON_QUERYファンクションと同じ方法でJSONデータを評価します。つまり、1つ以上の指定されたJSON値を検索し、それらのJSON値を含む文字列の列を戻します。

column_nameを使用して、戻される列の名前を指定します。JSON_query_columnの残りの句は、JSON_QUERYファンクションと同じセマンティクスを持ちます。これらの句の詳細は、JSON_QUERYを参照してください。「JSON_query_columnの使用方法：例」の例も参照してください。

JSON_value_column

この句は、JSON_VALUEファンクションと同じ方法でJSONデータを評価します。つまり、指定されたスカラーJSON値を確認し、SQL値としてそれらのJSON値の列を戻します。

column_nameを使用して、戻される列の名前を指定します。JSON_value_columnの残りの句は、JSON_VALUEファンクションと同じセマンティクスを持ちます。これらの句の詳細は、JSON_VALUEを参照してください。「JSON_value_columnの使用方法：例」の例も参照してください。

JSON_nested_path

この句を使用して、ネストされたJSONオブジェクトまたはJSON配列のJSON値を親のオブジェクトまたは配列のJSON値とともに単一の行の個々の列にフラット化できます。この句を再帰的に使用して、ネストされたオブジェクトまたは配列の複数のレイヤーから単一の行にデータを投影できます。

JSON_basic_path_expression句を指定して、ネストされたオブジェクトまたは配列と照合します。このパス式は、JSON_TABLEファンクションで指定されたSQL/JSON行パス式に相対的になります。

COLUMNS句を使用して、戻されるネストされたオブジェクトまたは配列の列を定義します。この句は再帰的です。つまり、別のJSON_nested_path句内にJSON_nested_path句を指定できます。「JSON_nested_pathの使用方法：例」の例も参照してください。

ordinality_column

この句では、データ型がNUMBERの、生成された行番号の列を戻します。1つのordinality_columnのみ指定できます。ordinality_column句を使用した例は、「JSON_value_columnの使用方法：例」も参照してください。

nested_clause

nested_clauseは、JSON値をリレーショナルの列にマッピングする簡易構文として使用します。JSON_TABLE columns句の構文を再利用し、基本的にはJSON_TABLEでの左外部ANSI結合と等価です。

nested_clauseを使用する例1は、JSON_TABLEで左外部結合を使用する例2と等価です。

例1 Nested_Clause

```
SELECT t.*
FROM j_purchaseOrder
NESTED po_document COLUMNS(PONumber, Reference, Requestor) t;
PONUMBER REFERENCE REQUESTOR
-------------- ------------------------------ ----------------------------
1600 ABULL-20140421 Alexis Bull
```

例2 JSON_TABLEでの左外部結合

```
SELECT t.*
FROM j_purchaseOrder LEFT OUTER JOIN
JSON_TABLE(po_document COLUMNS(PONumber, Reference, Requestor)) t ON 1=1;
```

nested_clauseを使用するとき、NESTEDキーワードに従うJSONの列名はSELECT *拡張に含まれません。たとえば:

```
SELECT *
FROM j_purchaseOrder
NESTED po_document.LineItems[*]
COLUMNS(ItemNumber, Quantity NUMBER);
ID DATE_LOADED ITEMN QUANTITY
------------------------ ---------------------------------------- ------- -----------
6C5589E9A9156… 16-MAY-18 08.40.30.397688 AM -07:00 1 9
6C5589E9A9156… 16-MAY-18 08.40.30.397688 AM -07:00 2 5
```

結果に、JSON列内po_documentは、結果の列の1つとして含まれません。

JSON列データのネストを解除するときには、LEFT OUTER JOINセマンティクスを使用することをお薦めします。行を生成しないJSON列が、他の非JSONデータを結果から除外しないようにするためです。たとえば、NULLのpo_document列がある

j_purchaseOrder行は、NULLでない可能性があるリレーショナルのcolumns idとdate_loadedを結果から除外しません。

columns句は、列のネストも含め、JSON_TABLEで定義されているのと同じ機能をすべてサポートしています。たとえば:

```
SELECT t.*
FROM j_purchaseorder
NESTED po_document COLUMNS(PONumber, Reference,
NESTED LineItems[*] COLUMNS(ItemNumber, Quantity)
) t
PONUMBER REFERENCE ITEMN QUANTITY
--------------- ------------------------------ ----- ------------
1600 ABULL-20140421 1 9
1600 ABULL-20140421 2 5
```

例

JSONドキュメントを含む表の作成: 例

この例では、この項の残りのJSON_TABLEの例で使用される、表j_purchaseorderの作成方法と移入方法を示します。

次の文は、表j_purchaseorderを作成します。列po_documentはJSONデータを格納するために使用され、整形式のJSONのみを列に格納するためにIS JSON CHECK制約があります。

```
CREATE TABLE j_purchaseorder
  (id RAW (16) NOT NULL,
   date_loaded TIMESTAMP(6) WITH TIME ZONE,
   po_document CLOB CONSTRAINT ensure_json CHECK (po_document IS JSON));
```

次の文は、1つの行または1つのJSONドキュメントを表j_purchaseorderに挿入します。

```
INSERT INTO j_purchaseorder
  VALUES (
    SYS_GUID(),
    SYSTIMESTAMP,
    '{"PONumber"             : 1600,
      "Reference"            : "ABULL-20140421",
       "Requestor"           : "Alexis Bull",
       "User"                : "ABULL",
       "CostCenter"          : "A50",
       "ShippingInstructions" : {"name"   : "Alexis Bull",
                                 "Address": {"street"   : "200 Sporting Green",
                                             "city"     : "South San Francisco",
                                             "state"    : "CA",
                                             "zipCode" : 99236,
                                             "country" : "United States of
America"},
                                 "Phone" : [{"type" : "Office", "number" : "909-555-
7307"},
                                            {"type" : "Mobile", "number" : "415-555-
1234"}]},
       "Special Instructions" : null,
       "AllowPartialShipment" : true,
       "LineItems" : [{"ItemNumber" : 1,
                       "Part" : {"Description" : "One Magic Christmas",
                                 "UnitPrice"   : 19.95,
                                 "UPCCode"     : 13131092899},
                       "Quantity" : 9.0},
                      {"ItemNumber" : 2,
                       "Part" : {"Description" : "Lethal Weapon",
                                 "UnitPrice"   : 19.95,
                                 "UPCCode"     : 85391628927},
                       "Quantity" : 5.0}]}');
```

JSON_query_columnの使用方法: 例

この例の文は、JSON_query_column句を使用して特定のJSONプロパティのJSONデータを問い合せて、列のプロパティ値を戻します。

この文は、最初にSQL/JSON行パス式を列po_documentに適用し、これによってShippingInstructionsプロパティと一致します。COLUMNS句はJSON_query_column句を使用して、VARCHAR2(100)列のPhoneプロパティ値を戻します。

```
SELECT jt.phones
FROM j_purchaseorder,
JSON_TABLE(po_document, '$.ShippingInstructions'
COLUMNS
  (phones VARCHAR2(100) FORMAT JSON PATH '$.Phone')) AS jt;

PHONES
--------------------------------------------------------------------------------
[{"type":"Office","number":"909-555-7307"},{"type":"Mobile","number":"415-555-1234"}]
```

JSON_value_columnの使用方法: 例

この例の文は、JSON_value_column句を使用して特定のJSON値のJSONデータを問い合せて前の例の文を改良し、リレーショナル行および列のSQL値としてJSON値を戻します。

この文は、最初にSQL/JSON行パス式を列po_documentに適用し、これによってJSON配列Phoneの要素と一致します。これらの要素は、typeとnumberの2つのメンバーを含むJSONオブジェクトです。この文は、COLUMNS句を使用して、phone_typeと呼ばれるVARCHAR2(10)列の各オブジェクトのtype値およびphone_numと呼ばれるVARCHAR2(20)列の各オブジェクトのnumber値を戻します。この文は、row_numberと呼ばれる順序列も戻します。

```
SELECT jt.*
FROM j_purchaseorder,
JSON_TABLE(po_document, '$.ShippingInstructions.Phone[*]'
COLUMNS (row_number FOR ORDINALITY,
         phone_type VARCHAR2(10) PATH '$.type',
         phone_num VARCHAR2(20) PATH '$.number'))
AS jt;
ROW_NUMBER PHONE_TYPE PHONE_NUM
---------- ---------- --------------------
         1 Office     909-555-7307
         2 Mobile     415-555-1234
```

JSON_exists_columnの使用方法: 例

この例の文は、JSON_exists_column句を使用してJSON値がJSONデータにあるかどうかをテストします。最初の例は、列の'true'または'false'値としてテストの結果を戻します。2番目の例は、WHERE句のテストの結果を使用します。

次の文は、最初にSQL/JSON行パス式を列po_documentに適用し、これによってコンテキスト項目全体またはJSONドキュメントと一致します。次に、COLUMNS句を使用して、リクエスタの名前およびリクエスタのJSONデータに郵便番号が含まれるかどうかを示す'true'または'false'の文字列値を戻します。COLUMNS句は最初にJSON_value_column句を使用して、requestorと呼ばれるVARCHAR2(32)列のRequestor値を戻します。次に、JSON_exists_column句を使用して、zipCodeオブジェクトがあるかどうかを判断し、has_zipと呼ばれるVARCHAR2(5)列の結果を戻します。

```
SELECT requestor, has_zip
FROM j_purchaseorder,
JSON_TABLE(po_document, '$'
COLUMNS
  (requestor VARCHAR2(32) PATH '$.Requestor',
   has_zip VARCHAR2(5) EXISTS PATH '$.ShippingInstructions.Address.zipCode'));
REQUESTOR                        HAS_ZIP
```

```
------------------------------- -------
Alexis Bull                     true
```

次の文は、WHERE句のhas_zipの値を使用してRequestor値を戻すかどうかを判断している点を除いて、前の文と似ています。

```
SELECT requestor
FROM j_purchaseorder,
JSON_TABLE(po_document, '$'
COLUMNS
  (requestor VARCHAR2(32) PATH '$.Requestor',
   has_zip VARCHAR2(5) EXISTS PATH '$.ShippingInstructions.Address.zipCode'))
WHERE (has_zip = 'true');
REQUESTOR
--------------------------------
Alexis Bull
```

JSON_nested_pathの使用方法: 例

次の2つの単純な文で、JSON_nested_path句の機能を示します。3つの要素を含む簡単なJSON配列を操作します。最初の2つの要素は数値です。3つ目の要素は、2つの文字列値要素を含むネストされたJSON配列です。

次の文は、JSON_nested_path句を使用しません。単一の行の配列の3つの要素を戻します。ネストされた配列はすべて戻します。

```
SELECT *
FROM JSON_TABLE('[1,2,["a","b"]]', '$'
COLUMNS (outer_value_0 NUMBER PATH '$[0]',
         outer_value_1 NUMBER PATH '$[1]',
         outer_value_2 VARCHAR2(20) FORMAT JSON PATH '$[2]'));
OUTER_VALUE_0 OUTER_VALUE_1 OUTER_VALUE_2
------------- ------------- --------------------
            1             2 ["a","b"]
```

JSON_nested_path句を使用して親の配列要素とともに単一の行の個々の列にネストされた配列の個々の要素を戻しているため、次の文は前の文と異なります。

```
SELECT *
FROM JSON_TABLE('[1,2,["a","b"]]', '$'
COLUMNS (outer_value_0 NUMBER PATH '$[0]',
         outer_value_1 NUMBER PATH '$[1]',
         NESTED PATH '$[2]'
         COLUMNS (nested_value_0 VARCHAR2(1) PATH '$[0]',
                  nested_value_1 VARCHAR2(1) PATH '$[1]')));
OUTER_VALUE_0 OUTER_VALUE_1 NESTED_VALUE_0 NESTED_VALUE_1
------------- ------------- -------------- --------------
            1             2 a              b
```

前の例は、ネストされたJSON配列によるJSON_nested_pathの使用方法を示します。次の文は、親オブジェクト要素とともに単一の行の個々の列にネストされたオブジェクトの個々の要素を戻してネストされたJSONオブジェクトとともにJSON_nested_path句を使用する方法を示します。

```
SELECT *
FROM JSON_TABLE('{a:100, b:200, c:{d:300, e:400}}', '$'
COLUMNS (outer_value_0 NUMBER PATH '$.a',
         outer_value_1 NUMBER PATH '$.b',
         NESTED PATH '$.c'
         COLUMNS (nested_value_0 NUMBER PATH '$.d',
                  nested_value_1 NUMBER PATH '$.e')));
OUTER_VALUE_0 OUTER_VALUE_1 NESTED_VALUE_0 NESTED_VALUE_1
------------- ------------- -------------- --------------
          100           200            300            400
```

次の文は、j_purchaseorder表の問合せ時にJSON_nested_path句を使用します。最初に行パス式を列po_documentに適用し、これによってコンテキスト項目全体またはJSONドキュメントと一致します。COLUMNS句を使用して、requestorと呼ばれるVARCHAR2(32)列のRequestor値を戻します。その後、JSON_nested_path句を使用して、ネストされたPhone配列の各メンバーの個々のオブジェクトのプロパティ値を戻します。行がネストされた配列の各メンバーに生成され、各行に対応するRequestor値が含まれます。

```
SELECT jt.*
FROM j_purchaseorder,
JSON_TABLE(po_document, '$'
COLUMNS
  (requestor VARCHAR2(32) PATH '$.Requestor',
   NESTED PATH '$.ShippingInstructions.Phone[*]'
     COLUMNS (phone_type VARCHAR2(32) PATH '$.type',
              phone_num VARCHAR2(20) PATH '$.number')))
AS jt;


REQUESTOR            PHONE_TYPE           PHONE_NUM
-------------------- -------------------- --------------
Alexis Bull          Office               909-555-7307
Alexis Bull          Mobile               415-555-1234
```

次の例は、JSON_nested_pathで単純なドット表記法を使用したものと、ドット表記を使用しない同等の内容を示しています。

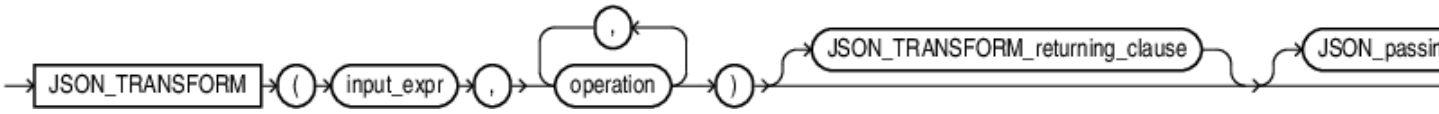
```
SELECT c.*
FROM customer t,
JSON_TABLE(t.json COLUMNS(
id, name, phone, address,
NESTED orders[*] COLUMNS(
updated, status,
NESTED lineitems[*] COLUMNS(
description, quantity NUMBER, price NUMBER
)
)
)) c;
```

ドット表記法の前述の文は、ドット表記法を使用しない次の文と同等です。

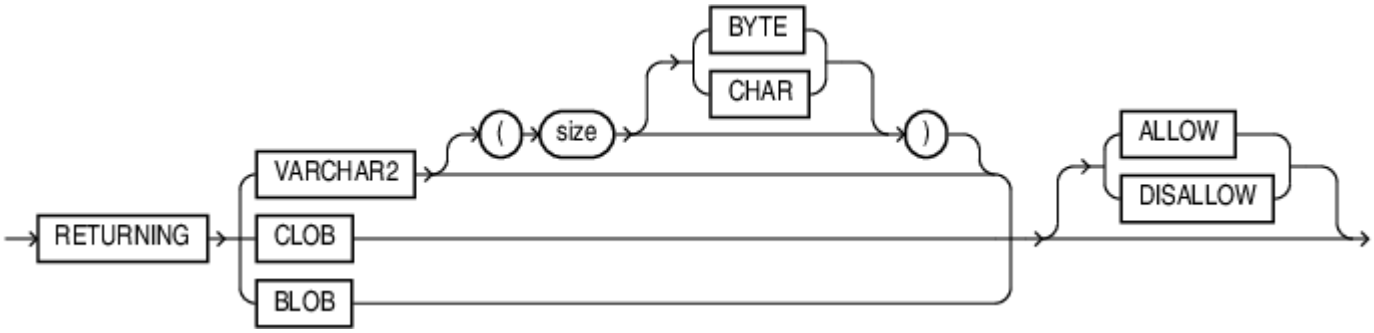
```
SELECT c.*
FROM customer t,
JSON_TABLE(t.json, '$' COLUMNS(
id PATH '$.id',
name PATH '$.name',
phone PATH '$.phone',
address PATH '$.address',
NESTED PATH '$.orders[*]' COLUMNS(
updated PATH '$.updated',
status PATH '$.status',
NESTED PATH '$.lineitems[*]' COLUMNS(
description PATH '$.description',
quantity NUMBER PATH '$.quantity',
price NUMBER PATH '$.price'
)
)
)) c;
```

JSON_TRANSFORM

構文



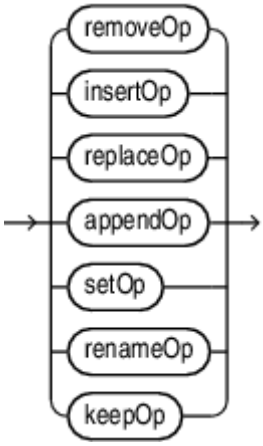
JSON_TRANSFORM_returning_clause::=



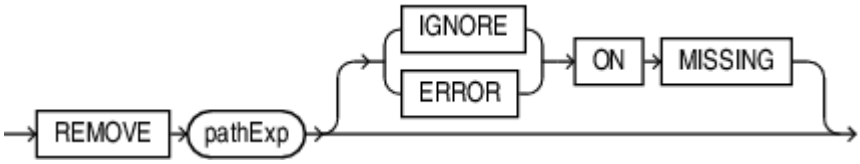
JSON_passing_clause::=

`JSON_passing_clause`の詳細は、「JSON_EXISTS条件」を参照してください。

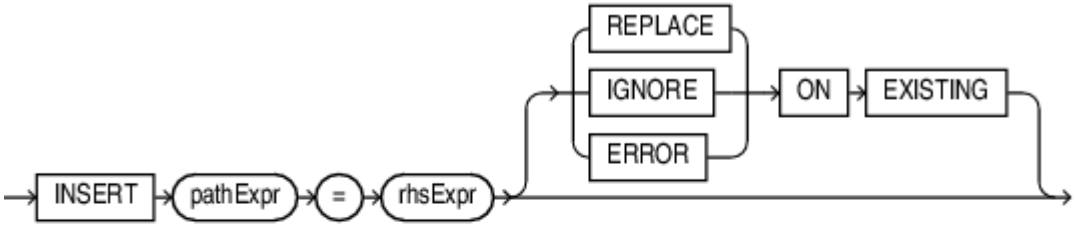
operation ::=

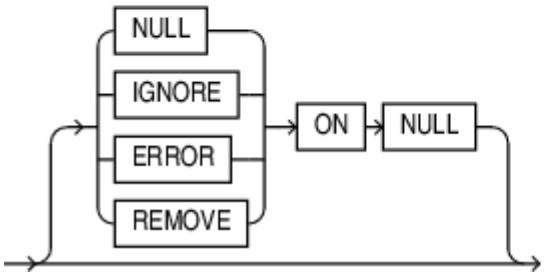


removeOp ::=

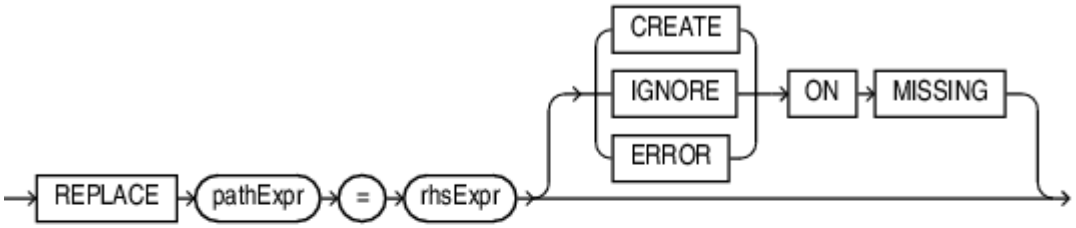


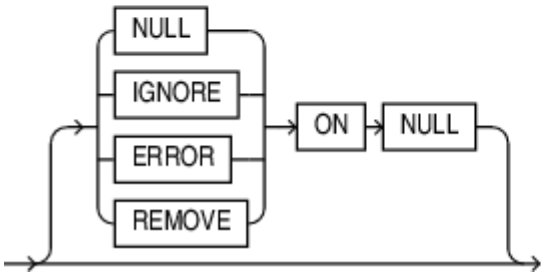
insertOp ::=





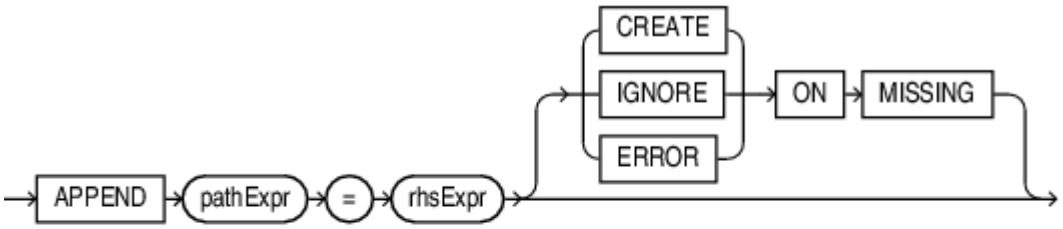
replaceOp ::=

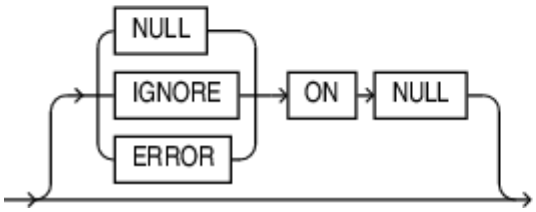




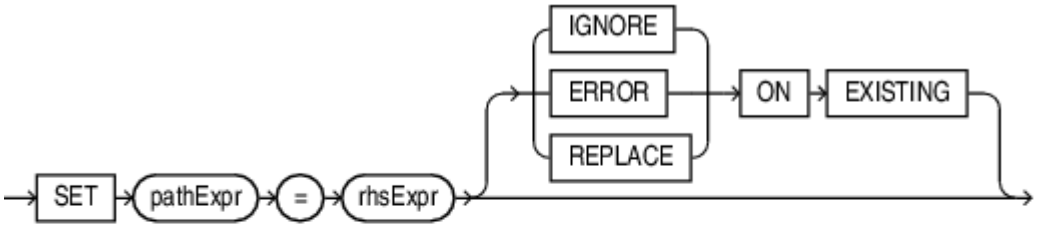
rhsExpr ::=

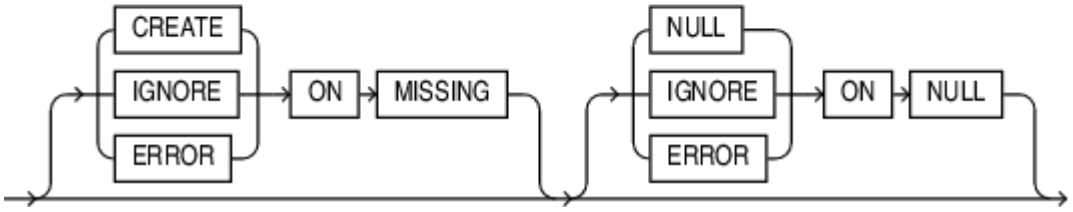
appendOp ::=



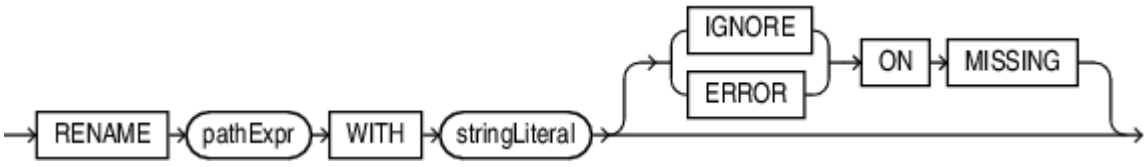


setOp ::=

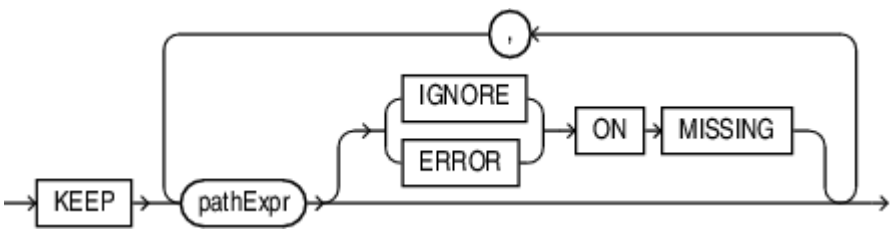




renameOp ::=



keepOp ::=



目的

JSON_TRANSFORM を使用して、ファンクションへのJSONドキュメント入力を変更します。JSONデータの変更を実行する1つ

以上の変更操作を指定することにより、JSONドキュメント(またはJSONドキュメントの一部)を変更できます。変更されたJSONドキュメントは出力として返されます。

JSONデータをサポートするSQLデータ型を入力できます。たとえば、入力は、IS JSONチェック制約の有無にかかわらずVARCHAR2列、またはJSONデータを返すファンクション・コールにできます。

例

例1: タイムスタンプを使用してJSON列を更新する

```
UPDATE t SET jcol = JSON_TRANSFORM(jcol, SET '$.lastUpdated' = SYSTIMESTAMP)
```

例2: クライアントにJSONを送信する前に、社会保障番号を削除する

```
SELECT JSON_TRANSFORM (jcol, REMOVE '$.ssn') FROM t WHERE …
```

JSON_TRANSFORM_returning_clause

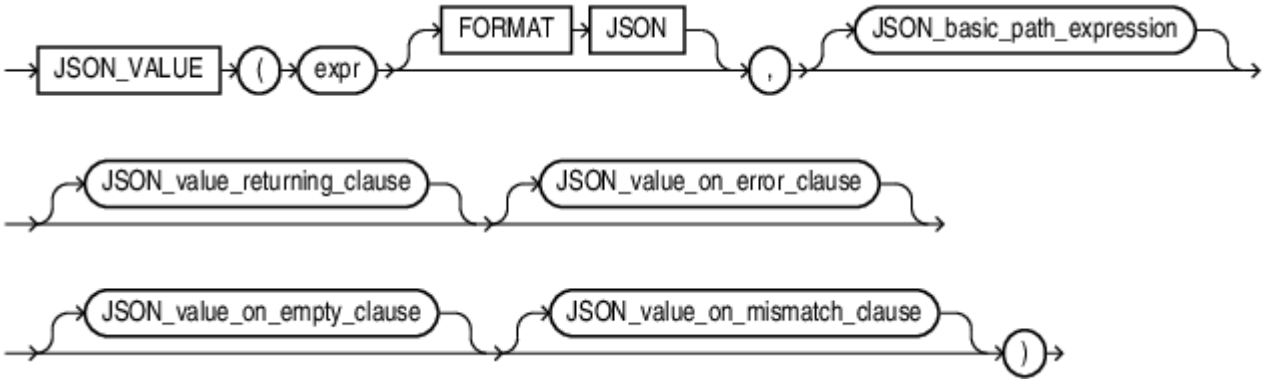
デフォルトの出力データ型は、入力のデータ型に一致します。

任意のサイズの入力データ型VARCHAR2の場合、デフォルトの出力データ型はVARCHAR2(4000)です。

例を含むJSON_TRANSFORMの詳細は、「Oracle SQLファンクションJSON_TRANSFORM」を参照してください。

JSON_VALUE

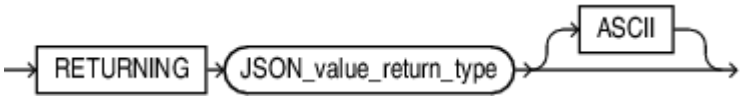
構文



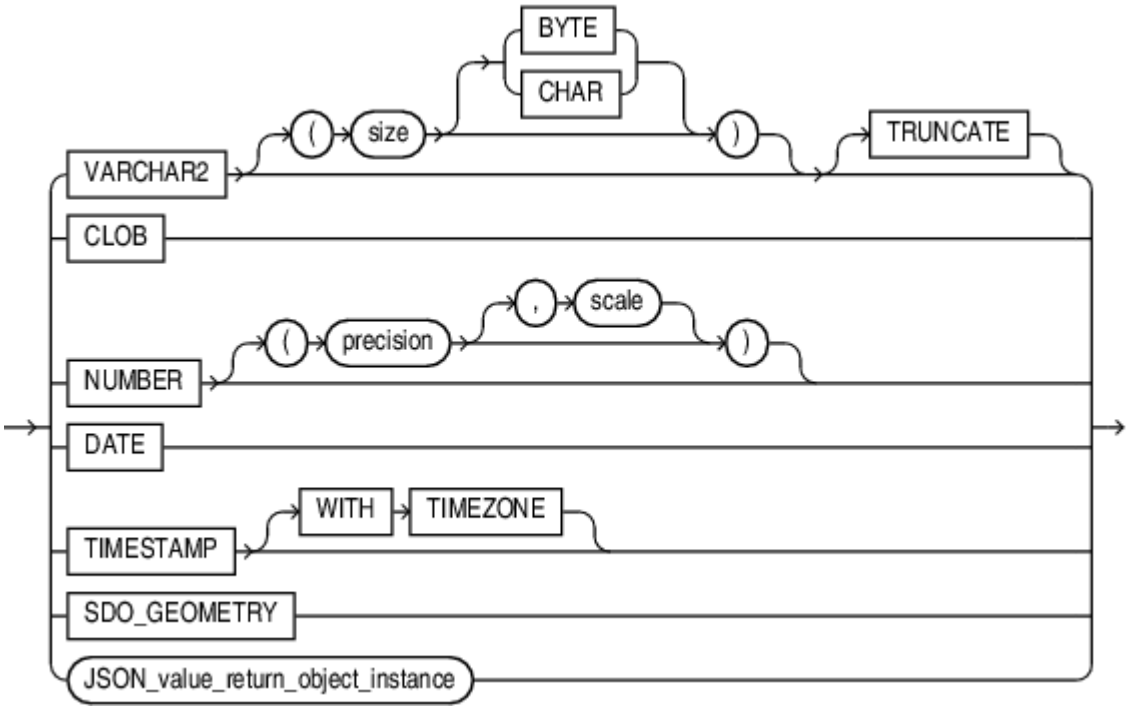
JSON_basic_path_expression::=

(`JSON_basic_path_expression`: SQL/JSONパス式を参照)

JSON_value_returning_clause::=

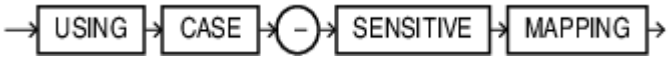


JSON_value_return_type::=

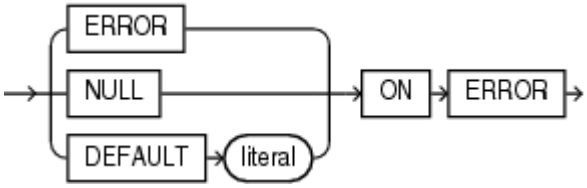


JSON_value_return_object_instance ::=

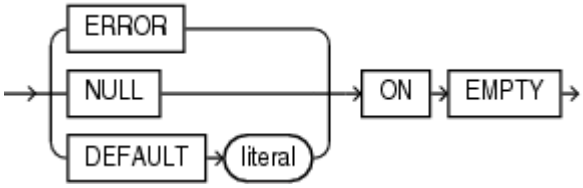
JSON_value_mapper_clause ::=



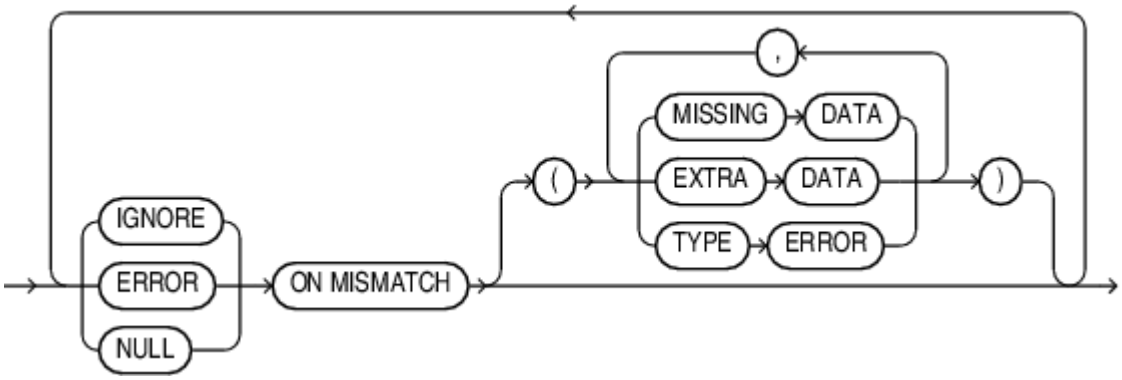
JSON_value_on_error_clause::=



JSON_value_on_empty_clause::=



JSON_value_on_mismatch_clause::=



目的

SQL/JSONファンクションJSON_VALUEは、JSONデータの指定されたスカラーJSON値を確認し、SQL値として戻します。

関連項目:

このファンクションによって戻される値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

expr

この句は、評価の対象となるJSONデータを指定するために使用します。exprでは、テキスト・リテラルを評価する式を指定します。exprが列である場合、列のデータ型はVARCHAR2、CLOBまたはBLOBのいずれかである必要があります。exprがNULLの場合はNULLを戻します。

exprが厳密なまたは緩い構文を使用した整形式のJSONデータのテキスト・リテラルでない場合、このファンクションはデフォルトでnullを戻します。JSON_value_on_error_clauseを使用して、このデフォルトの動作をオーバーライドできます。JSON_value_on_error_clauseを参照してください。

FORMAT JSON

exprがデータ・タイプBLOBの列の場合、FORMAT JSONを指定する必要があります。

JSON_basic_path_expression

この句を使用して、SQL/JSONパス式を指定します。このファンクションはパス式を使用してexprを評価し、パス式と一致する(パス式を満たす)スカラーJSON値を確認します。パス式はテキスト・リテラルである必要があります。
JSON_basic_path_expressionのセマンティクスの詳細は、Oracle Database JSON開発者ガイドを参照してください。

JSON_value_returning_clause

この句を使用して、このファンクションで戻される値のデータ型および書式を指定します。

RETURNING

RETURNING句を使用して、戻り値のデータ型を指定します。この句を省略すると、JSON_VALUEはVARCHAR2(4000)型の値を戻します。

JSON_value_return_type ::=

JSON_value_return_typeを使用して、次のデータ型を指定できます。

- VARCHAR2[(size [BYTE,CHAR])]

  このデータ型を指定すると、このファンクションで戻されるスカラー値は文字または数値になります。数値は暗黙的にVARCHAR2に変換されます。SQLでVARCHAR2データ型を指定する場合、サイズを指定する必要があります。ただし、この句はサイズを省略できます。この場合、JSON_VALUEはVARCHAR2(4000)型の値を戻します。

  戻り値がN文字よりも大きい場合は、VARCHAR2(N)の直後にオプションのTRUNCATE句を指定して、戻り値をN文字になるように切り捨てます。

  TRUNCATE句のノート:

  - 文字列値が長すぎる場合は、ORA-40478が発生します。
  - TRUNCATEが指定されており、戻り値がキャラクタ・タイプでない場合は、コンパイル時エラーが発生します。
  - TRUNCATEがFORMAT JSONとともに指定されている場合、構文的に正しくないJSONデータが戻り値に含まれる可能性があります。
  - TRUNCATEはEXISTSと一緒には機能しません。
- CLOB

  このデータ型を指定すると、シングルバイトまたはマルチバイト文字を含むキャラクタ・ラージ・オブジェクトが返されます。
- NUMBER[(precision [, scale])]

  このデータ型を指定すると、このファンクションで戻されるスカラー値は数値になります。戻されるスカラー値をJSONブール値にすることもできます。ただし、JSONブール値としてNUMBERを戻す操作は非推奨です。
- DATE

  このデータ型を指定すると、このファンクションで戻されるスカラー値は、暗黙的にDATEデータ型に変換できる文字値になります。
- TIMESTAMP

  このデータ型を指定すると、このファンクションで戻されるスカラー値は、暗黙的にTIMESTAMPデータ型に変換できる文字値になります。

- `TIME WITH TIME ZONE`

  このデータ型を指定すると、このファンクションで戻されるスカラー値は、暗黙的に`TIMESTAMP WITH TIME ZONE`データ型に変換できる文字値になります。

- `SDO_GEOMETRY`

  このデータ型は、Oracle Spatial and Graphデータで使用されます。このデータ型を指定すると、`expr`が、JSONの地理データをエンコードする書式であるGeoJSONデータを含むテキスト・リテラルに評価されます。このデータ型を指定すると、このファンクションで戻されるスカラー値は、`SDO_GEOMETRY`型のオブジェクトになります。

- JSON_value_return_object_instance

  `JSON_VALUE`がJSONオブジェクトをターゲットにしており、ユーザー定義のSQLオブジェクト型を戻り型として指定すると、`JSON_VALUE`はobject_type_nameのオブジェクト型のインスタンスを返します。

  たとえば、JSON_VALUEを使用してユーザー定義のオブジェクト型インスタンスをインスタンス化する例を参照

関連項目:

- 概念の理解には、SQL/JSON Function JSON_VALUE。
- 前述のデータ型の詳細は、「データ型」を参照してください。
- データ型が戻り値を保持できる十分な大きさでない場合、このファンクションはデフォルトでnullを戻します。`JSON_value_on_error_clause`を使用して、このデフォルトの動作をオーバーライドできます。JSON_value_on_error_clauseを参照してください。

ASCII

標準のASCII Unicodeエスケープ・シーケンスを使用して戻り値の非ASCII Unicode文字を自動的にエスケープするには、`ASCII`を指定します。

JSON_value_on_error_clause

この句を使用して、次のエラーが発生した場合にこのファンクションで戻される値を指定します。

- `expr`が厳密なまたは緩いJSON構文を使用した整形式のJSONデータではありません。
- JSONデータがSQL/JSONパス式を使用して評価される場合に非スカラー値が見つかります。
- SQL/JSONパス式を使用してJSONデータを評価した場合に一致が見つかりません。`JSON_value_on_empty_clause`を指定して、このタイプのエラーに対する動作を上書きできます。
- 戻り値のデータ型が戻り値を保持する十分な大きさではありません。

次の句を指定できます。

- `NULL ON ERROR` - エラーが発生した場合にnullを戻します。これはデフォルトです。
- `ERROR ON ERROR` - エラーが発生した場合に適切なOracleエラーを戻します。
- `DEFAULT literal ON ERROR` - エラーが発生した場合に`literal`を戻します。`literal`のデータ型は、このファンクションにより戻される値のデータ型と一致する必要があります。

JSON_value_on_empty_clause

この句を使用して、JSONデータがSQL/JSONパス式を使用して評価されるときに一致が見つからない場合にこのファンクション

で戻される値を指定します。この句により、JSON_value_on_error_clauseで指定された結果とは異なる、このタイプのエラーに対する結果を指定できます。

次の句を指定できます。

- NULL ON EMPTY - 一致が見つからない場合にNULLを戻します。
- ERROR ON EMPTY - 一致が見つからない場合に適切なOracleエラーを戻します。
- DEFAULT literal ON EMPTY - 一致が見つからない場合にliteralを戻します。literalのデータ型は、このファンクションにより戻される値のデータ型と一致する必要があります。

この句を省略すると、JSON_value_on_error_clauseによって、一致が見つからない場合に戻される値が決まります。

JSON_value_on_mismatch_clause

JSON_value_on_mismatch_clauseには、一般と個別の2つの使用方法があります。

追加データ、欠落データ、入力エラーなどのエラー・ケースすべてに適用する場合は一般の使用方法です。

ケースごとに異なるON MISMATCH句を指定するのが、個別の使用方法です。たとえば:

```
IGNORE ON MISMATCH (EXTRA DATA)

ERROR ON MISMATCH ( MISSING DATA, TYPE ERROR)
```

例

次の問合せは、プロパティ名aのメンバーの値を戻します。RETURNING句が指定されていないため、値がVARCHAR2(4000)データ型として戻されます。

```
SELECT JSON_VALUE('{a:100}', '$.a') AS value
  FROM DUAL;
VALUE
-----
100
```

次の問合せは、プロパティ名aのメンバーの値を戻します。RETURNING NUMBER句が指定されているため、値がNUMBERデータ型として戻されます。

```
SELECT JSON_VALUE('{a:100}', '$.a' RETURNING NUMBER) AS value
  FROM DUAL;
     VALUE
----------
       100
```

次の問合せは、プロパティ名aを使用したメンバーの値にあるプロパティ名bを使用したメンバーの値を戻します。

```
SELECT JSON_VALUE('{a:{b:100}}', '$.a.b') AS value
  FROM DUAL;
VALUE
-----
100
```

次の問合せは、任意のオブジェクトのプロパティ名dのメンバーの値を戻します。

```
SELECT JSON_VALUE('{a:{b:100}, c:{d:200}, e:{f:300}}', '$.*.d') AS value
  FROM DUAL;
VALUE
-----
200
```

次の問合せは、配列の最初の要素の値を戻します。

```
SELECT JSON_VALUE('[0, 1, 2, 3]', '$[0]') AS value
  FROM DUAL;
VALUE
-----
0
```

次の問合せは、配列の3番目の要素の値を戻します。配列は、プロパティ名aのメンバーの値です。

```
SELECT JSON_VALUE('{a:[5, 10, 15, 20]}', '$.a[2]') AS value
  FROM DUAL;
VALUE
-----
15
```

次の問合せは、配列の2番目のオブジェクトのプロパティ名aのメンバーの値を戻します。

```
SELECT JSON_VALUE('[{a:100}, {a:200}, {a:300}]', '$[1].a') AS value
  FROM DUAL;
VALUE
-----
200
```

次の問合せは、配列の任意のオブジェクトのプロパティ名cのメンバーの値を戻します。

```
SELECT JSON_VALUE('[{a:100}, {b:200}, {c:300}]', '$[*].c') AS value
  FROM DUAL;
VALUE
-----
300
```

次の問合せは、プロパティ名lastnameのメンバーの値を戻そうとします。ただし、そのメンバーは指定されたJSONデータに存在しないため、一致は見つかりません。ON ERROR句が指定されていないため、この文はデフォルトのNULL ON ERRORを使用し、nullを戻します。

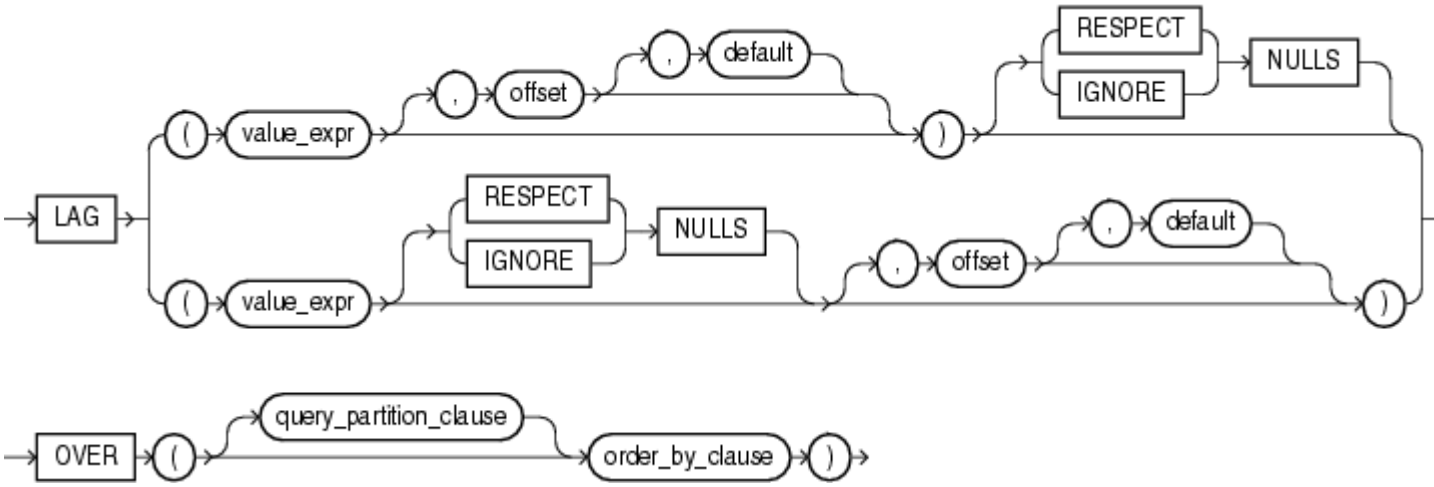
```
SELECT JSON_VALUE('{firstname:"John"}', '$.lastname') AS "Last Name"
  FROM DUAL;
Last Name
---------
```

指定されたJSONに存在しないプロパティ名lastnameのメンバーの値を戻そうとするため、次の問合せはエラーになります。ON ERROR句が指定されているため、この文は指定されたテキスト・リテラルを戻します。

```
SELECT JSON_VALUE('{firstname:"John"}', '$.lastname'
                  DEFAULT 'No last name found' ON ERROR) AS "Last Name"
  FROM DUAL;
Last Name
---------
No last name found
```

LAG

構文



関連項目:

構文、セマンティクス、制限事項、およびvalue_exprの書式の詳細は、「分析ファンクション」を参照してください。

目的

LAGは分析ファンクションです。これは、自己結合せずに、表の2つ以上の行へ同時アクセスを行います。問合せから戻される一連の行およびカーソル位置を指定すると、LAGは、その位置より前にある指定された物理オフセットにある行へアクセスします。

オプションのoffset引数には、0(ゼロ)より大きい整数を指定します。offsetを指定しない場合、デフォルト値は1です。オフセットがウィンドウの有効範囲を超えた場合、オプションのdefault値が戻されます。defaultを指定しない場合、デフォルトはNULLです。

{RESPECT | IGNORE} NULLSは、value_exprのNULL値を計算に含めるか除外するかを指定します。デフォルトはRESPECT NULLSです。

value_exprには、LAGまたは他の分析ファンクションを使用して分析ファンクションをネストできません。ただし、他の組込みファンクション式をvalue_exprで使用できます。

関連項目:

- exprの書式の詳細は、「SQL式」を参照してください。「LEAD」も参照してください。
- LAGの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、employees表の各購買係について、その購買係の直前に雇用された従業員の給与を示します。

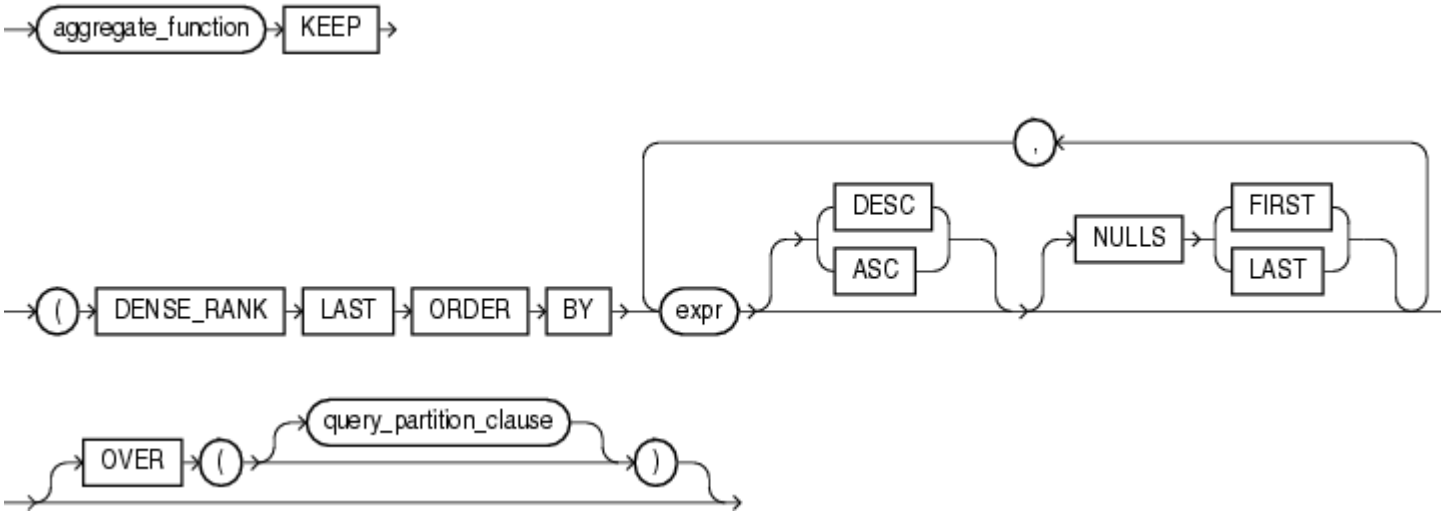
```
SELECT hire_date, last_name, salary,
       LAG(salary, 1, 0) OVER (ORDER BY hire_date) AS prev_sal
  FROM employees
  WHERE job_id = 'PU_CLERK'
  ORDER BY hire_date;
```

```
HIRE_DATE LAST_NAME                     SALARY   PREV_SAL
--------- ------------------------- ---------- ----------
18-MAY-03 Khoo                            3100          0
24-JUL-05 Tobias                          2800       3100
24-DEC-05 Baida                           2900       2800
15-NOV-06 Himuro                          2600       2900
10-AUG-07 Colmenares                      2500       2600
```

LAST

構文

last::=



関連項目:

query_partitioning_clauseの構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

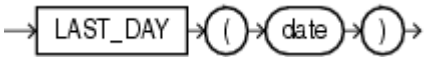
目的

FIRSTファンクションとLASTファンクションは類似しています。両方のファンクションとも、与えられたソート指定に対して、FIRSTまたはLASTとしてランク付けされた一連の行の一連の値を操作する集計ファンクションおよび分析ファンクションです。1つの行のみがFIRSTまたはLASTとしてランク付けされている場合、集計は、1つの要素のみを持つセットを操作します。

このファンクションの詳細および使用例は、「FIRST」を参照してください。

LAST_DAY

構文



目的

LAST_DAYは、dateを含む月の最終日の日付を戻します。月の最終日は、セッション・パラメータNLS_CALENDARによって定義されます。戻り型は、dateのデータ型に関係なく常にDATEです。

例

次の文は、現在の月の残りの日数を確認します。

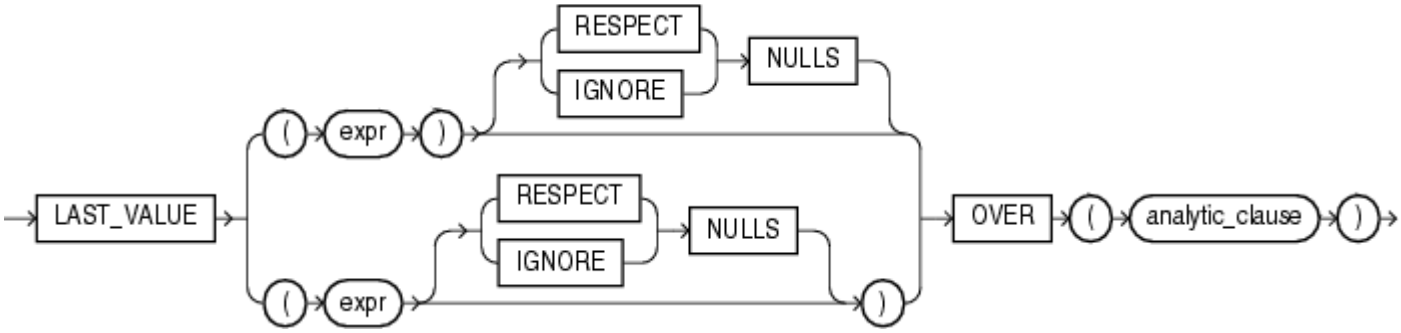
```
SELECT SYSDATE,
       LAST_DAY(SYSDATE) "Last",
       LAST_DAY(SYSDATE) - SYSDATE "Days Left"
  FROM DUAL;
SYSDATE   Last       Days Left
--------- --------- ----------
30-MAY-09 31-MAY-09          1
```

次の例では、各従業員の雇用開始日に5か月を加えて、評価日付を戻します。

```
SELECT last_name, hire_date,
       TO_CHAR(ADD_MONTHS(LAST_DAY(hire_date), 5)) "Eval Date"
  FROM employees
  ORDER BY last_name, hire_date;
LAST_NAME                 HIRE_DATE Eval Date
------------------------- --------- ---------
Abel                      11-MAY-04 31-OCT-04
Ande                      24-MAR-08 31-AUG-08
Atkinson                  30-OCT-05 31-MAR-06
Austin                    25-JUN-05 30-NOV-05
Baer                      07-JUN-02 30-NOV-02
Baida                     24-DEC-05 31-MAY-06
Banda                     21-APR-08 30-SEP-08
Bates                     24-MAR-07 31-AUG-07
. . .
```

LAST_VALUE

構文



関連項目:

構文、セマンティクス、制限事項、およびexprの書式の詳細は、「分析ファンクション」を参照してください。

目的

LAST_VALUEはデータの稠密化に役立つ分析ファンクションです。これは、順序付けられた値の集合にある最後の値を戻します。

ノート:

この構文の 2 つの書式は、同じ動作になります。上のブランチは ANSI 形式です。ANSI との互換性を維持するために、こちらを使用することをお薦めします。

{RESPECT | IGNORE} NULLSは、exprのNULL値を計算に含めるか除外するかを指定します。デフォルトはRESPECT NULLSです。集合内の最後の値がNULLの場合、IGNORE NULLSを指定していないかぎり、ファンクションはNULLを戻します。IGNORE NULLSを指定すると、LAST_VALUEは集合内の最後のNULLではない値を戻します。すべての値がNULLの場合はNULLを戻します。データの稠密化の例は、「パーティション化された外部結合の使用方法: 例」を参照してください。

exprには、LAST_VALUEまたは他の分析ファンクションを使用して分析ファンクションをネストできません。ただし、他の組込みファンクション式をexprで使用できます。exprの書式の詳細は、「SQL式」を参照してください。

analytic_clauseのwindowing_clauseを省略した場合、デフォルトでRANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROWになります。ウィンドウの最後の値はウィンドウの終端にあり、固定されていないため、このデフォルトは予期しない値を戻す場合があります。これは、現行の行が変化するに伴って変化します。正しい結果を得るには、windowing_clauseをRANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWINGとして指定します。または、windowing_clauseをRANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWINGとして指定することもできます。

関連項目:

このファンクションの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、給与が最も低い従業員の雇用開始日を各行に戻します。

```
SELECT employee_id, last_name, salary, hire_date,
       LAST_VALUE(hire_date)
         OVER (ORDER BY salary DESC ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED
               FOLLOWING) AS lv
  FROM (SELECT * FROM employees
          WHERE department_id = 90
          ORDER BY hire_date);
EMPLOYEE_ID LAST_NAME                 SALARY HIRE_DATE LV
----------- ------------------------- ---------- --------- ---------
        100 King                       24000 17-JUN-03 13-JAN-01
        101 Kochhar                    17000 21-SEP-05 13-JAN-01
        102 De Haan                    17000 13-JAN-01 13-JAN-01
```

この例では、LAST_VALUEファンクションの非決定的な性質を示しています。KochharとDe Haanの給与は同じであるため、Kochharの次の行にDe Haanがあります。Kochharが最初に表示されているのは、副問合せの行がhire_dateで順序付けられているためです。ただし、行がhire_dateで降順に順序付けられている場合は、次の例に示すとおり、ファンクションは異なる値を戻します。

```
SELECT employee_id, last_name, salary, hire_date,
       LAST_VALUE(hire_date)
         OVER (ORDER BY salary DESC ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED
               FOLLOWING) AS lv
  FROM (SELECT * FROM employees
          WHERE department_id = 90
          ORDER BY hire_date DESC);
EMPLOYEE_ID LAST_NAME                 SALARY HIRE_DATE LV
----------- ------------------------- ---------- --------- ---------
        100 King                       24000 17-JUN-03 21-SEP-05
        102 De Haan                    17000 13-JAN-01 21-SEP-05
        101 Kochhar                    17000 21-SEP-05 21-SEP-05
```

次の2つの例では、一意キーで順序付けることによって、LAST_VALUEファンクションを決定的にする方法を示しています。給与と一意キーemployee_idの両方でファンクション内を順序付けると、副問合せの順序付けにかかわらず、同じ結果が戻されます。

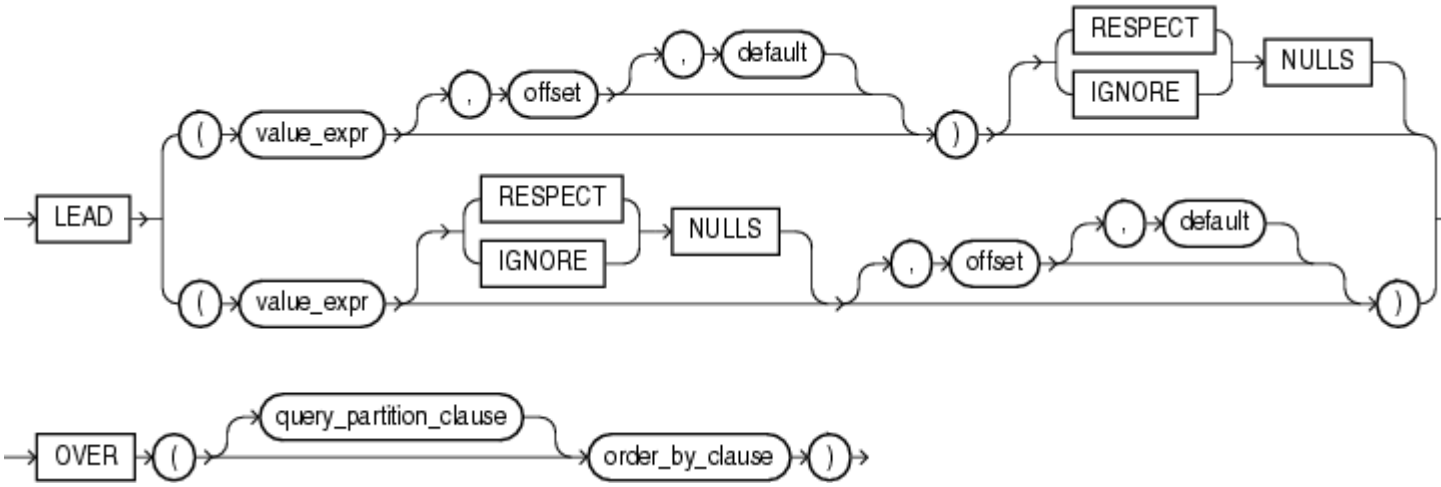
```
SELECT employee_id, last_name, salary, hire_date,
       LAST_VALUE(hire_date)
         OVER (ORDER BY salary DESC, employee_id ROWS BETWEEN UNBOUNDED PRECEDING
               AND UNBOUNDED FOLLOWING) AS lv
  FROM (SELECT * FROM employees
          WHERE department_id = 90
          ORDER BY hire_date);
EMPLOYEE_ID LAST_NAME                 SALARY HIRE_DATE LV
----------- ------------------------- ---------- --------- ---------
        100 King                       24000 17-JUN-03 13-JAN-01
        101 Kochhar                    17000 21-SEP-05 13-JAN-01
        102 De Haan                    17000 13-JAN-01 13-JAN-01
SELECT employee_id, last_name, salary, hire_date,
       LAST_VALUE(hire_date)
         OVER (ORDER BY salary DESC, employee_id ROWS BETWEEN UNBOUNDED PRECEDING
               AND UNBOUNDED FOLLOWING) AS lv
  FROM (SELECT * FROM employees
          WHERE department_id = 90
          ORDER BY hire_date DESC);
EMPLOYEE_ID LAST_NAME                 SALARY HIRE_DATE LV
----------- ------------------------- ---------- --------- ---------
        100 King                       24000 17-JUN-03 13-JAN-01
        101 Kochhar                    17000 21-SEP-05 13-JAN-01
        102 De Haan                    17000 13-JAN-01 13-JAN-01
```

次の2つの例では、論理オフセット(ROWSのかわりにRANGE)を使用すると、LAST_VALUEファンクションが決定的になることを示しています。ORDER BY式に重複値がある場合は、LAST_VALUEがexprの最大値になります。

```
SELECT employee_id, last_name, salary, hire_date,
       LAST_VALUE(hire_date)
         OVER (ORDER BY salary DESC RANGE BETWEEN UNBOUNDED PRECEDING AND
               UNBOUNDED FOLLOWING) AS lv
  FROM (SELECT * FROM employees
          WHERE department_id = 90
          ORDER BY hire_date);
EMPLOYEE_ID LAST_NAME                     SALARY HIRE_DATE LV
----------- ------------------------- ---------- --------- ---------
        100 King                           24000 17-JUN-03 21-SEP-05
        102 De Haan                        17000 13-JAN-01 21-SEP-05
        101 Kochhar                        17000 21-SEP-05 21-SEP-05
SELECT employee_id, last_name, salary, hire_date,
       LAST_VALUE(hire_date)
         OVER (ORDER BY salary DESC RANGE BETWEEN UNBOUNDED PRECEDING AND
               UNBOUNDED FOLLOWING) AS lv
  FROM (SELECT * FROM employees
          WHERE department_id = 90
          ORDER BY hire_date DESC);
EMPLOYEE_ID LAST_NAME                     SALARY HIRE_DATE LV
----------- ------------------------- ---------- --------- ---------
        100 King                           24000 17-JUN-03 21-SEP-05
        102 De Haan                        17000 13-JAN-01 21-SEP-05
        101 Kochhar                        17000 21-SEP-05 21-SEP-05
```

LEAD

構文



関連項目:

構文、セマンティクス、制限事項、およびvalue_exprの書式の詳細は、「分析ファンクション」を参照してください。

目的

LEADは分析ファンクションです。これは、自己結合せずに、表の2つ以上の行へ同時アクセスを行います。問合せから戻される一連の行およびカーソル位置を指定すると、LEADは、その位置より後にある指定された物理オフセットの行へアクセスします。

offsetを指定しない場合、デフォルト値は1です。オフセットが表の有効範囲を超えた場合、オプションのdefault値が戻されます。defaultを指定しない場合、デフォルト値はNULLです。

{RESPECT | IGNORE} NULLSは、value_exprのNULL値を計算に含めるか除外するかを指定します。デフォルトはRESPECT NULLSです。

value_exprには、LEADまたは他の分析ファンクションを使用して分析ファンクションをネストできません。ただし、他の組込みファンクション式をvalue_exprで使用できます。

関連項目:

- exprの書式の詳細は、「SQL式」を参照してください。「LAG」も参照してください。
- LEADの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Database グローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、employees表の部門30の各従業員について、その従業員の直後に雇用された従業員の雇用開始日を示します。

```
SELECT hire_date, last_name,
       LEAD(hire_date, 1) OVER (ORDER BY hire_date) AS "NextHired"
  FROM employees
  WHERE department_id = 30
  ORDER BY hire_date;
```

```
HIRE_DATE LAST_NAME                 Next Hired
--------- ------------------------- ----------
07-DEC-02 Raphaely                  18-MAY-03
18-MAY-03 Khoo                      24-JUL-05
24-JUL-05 Tobias                    24-DEC-05
24-DEC-05 Baida                     15-NOV-06
15-NOV-06 Himuro                    10-AUG-07
10-AUG-07 Colmenares
```

LEAST

構文

目的

LEASTは、1つ以上の式のリストの最小値を戻します。Oracle Databaseは、最初のexprを使用して戻り型を判断します。最初のexprが数値である場合、Oracleは、数値の優先順位が最も高い引数を判断し、比較の前に残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。最初のexprが数値ではない場合、比較の前に、2番目以降の各exprが最初のexprのデータ型に暗黙的に変換されます。

Oracle Databaseは、非空白埋め比較セマンティクスを使用して各exprを比較します。比較では、デフォルトの場合はバイナリが使用され、NLS_COMPパラメータがLINGUISTICに設定され、NLS_SORTパラメータにBINARY以外の設定がある場合は言語が使用されます。文字の比較は、データベース文字セットの文字の数値コードに基づいて行われます。また、文字ごとではなく、一連のバイトとして扱われる文字列全体で行われます。このファンクションによって戻される値が文字データの場合、そのデータ型は、最初のexprが文字データ型の場合はVARCHAR2、最初のexprが各国語キャラクタ・データ型の場合はNVARCHAR2になります。

関連項目:

- 文字の比較の詳細は、「データ型の比較規則」を参照してください。
- 暗黙的な変換の詳細は、表2-8を参照してください。2進浮動小数点の比較セマンティクスの詳細は、「浮動小数点数」を参照してください。
- 「GREATEST」を参照してください(GREATESTは1つ以上の式のリストの最大値を戻します)。
- exprの文字値を比較するためにLEASTで使用される照合を定義する照合決定ルール、およびこのファンクションの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の文では、最小値を持つ文字列を検索します。

```
SELECT LEAST('HARRY','HARRIOT','HAROLD') "Least"
  FROM DUAL;

Least
------
HAROLD
```

次の文では、最初の引数は数字です。Oracle Databaseは、数値の優先順位が最も高い引数は3番目の引数であると判断し、残りの引数を3番目の引数のデータ型に変換して、そのデータ型で最小値を戻します。

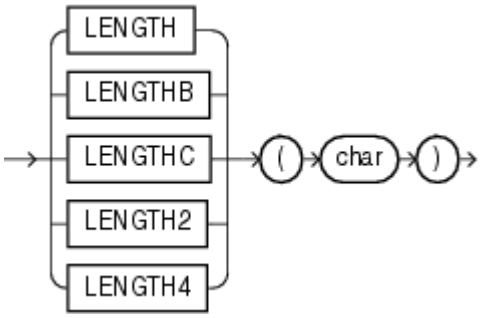
```
SELECT LEAST (1, '2.1', '.000832') "Least"
  FROM DUAL;

Least
-------
```

```
.000832
```

LENGTH

構文

length::=



目的

LENGTHの各ファンクションは、charの長さを戻します。LENGTHは、入力文字セットによって定義された文字を使用して、長さを算出します。LENGTHBは、文字のかわりにバイトを使用します。LENGTHCは、完全なUnicodeキャラクタを使用します。LENGTH2は、UCS2コードポイントを使用します。LENGTH4には、UCS4コード・ポイントを使用します。

charのデータ型は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBです。例外は、LENGTHC、LENGTH2およびLENGTH4であり、これらでは、charをCLOBおよびNCLOBのいずれにもできません。戻り値のデータ型はNUMBERです。charのデータ型がCHARの場合、その長さにはすべての後続空白が含まれます。charがNULLの場合、このファンクションはNULLを戻します。

文字長の詳細は、次を参照してください。

- Oracle Databaseグローバリゼーション・サポート・ガイド
- Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド

LENGTHBの制限事項

LENGTHBファンクションは、シングルバイトLOBのみでサポートされています。このファンクションは、マルチバイトの文字セットのCLOBとNCLOBデータでは使用できません。

例

次の例では、シングルバイトおよびマルチバイトのデータベース文字セットを使用するLENGTHファンクションを使用します。

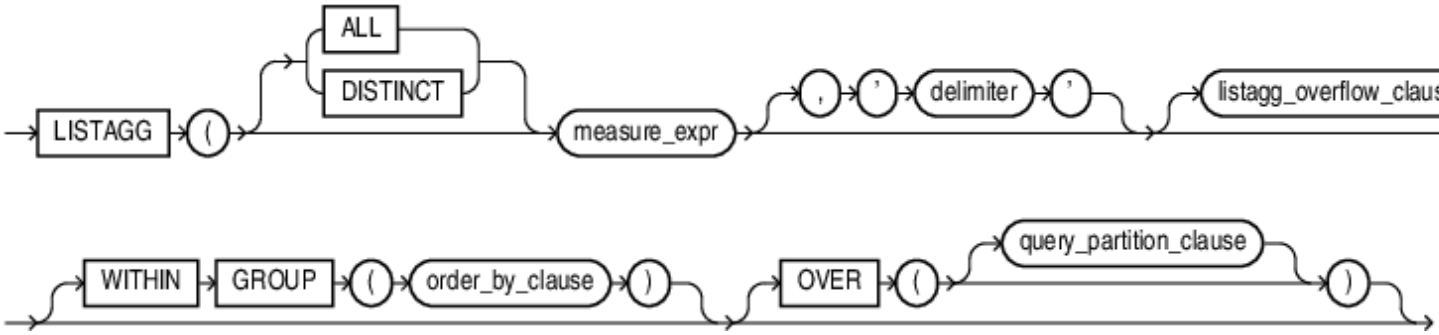
```
SELECT LENGTH('CANDIDE') "Length in characters"
  FROM DUAL;
Length in characters
--------------------
                   7
```

次の例では、データベース文字セットがダブルバイトの場合を想定しています。

```
SELECT LENGTHB ('CANDIDE') "Length in bytes"
  FROM DUAL;

Length in bytes
---------------
             14
```

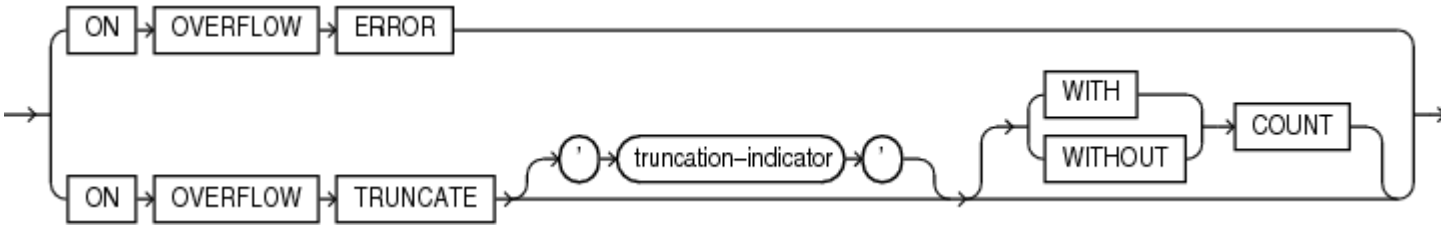
LISTAGG

構文



(listagg_overflow_clause::=、order_by_clause::=、query_partition_clause::=)

listagg_overflow_clause::=



関連項目:

ORDER BY句およびOVER句の構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

LISTAGGは、指定されたメジャーについて、ORDER BY句に指定された各グループ内でデータを順序付け、メジャー列の値を結合します。

- 単一セットの集計ファンクションとして使用すると、LISTAGGはすべての行に対して操作を行い、1つの出力行を戻します。
- グループ・セットの集計に使用すると、このファンクションはGROUP BY句に定義されたグループごとに操作を行い、そのグループごとに1つの出力行を戻します。
- 分析ファンクションとして使用すると、LISTAGGはquery_partition_clauseに指定された1つ以上の式に基づくグループに、問合せの結果セットを分割します。

ファンクションの引数には、次の規則があります。

- ALLキーワードはオプションで、セマンティクスを明確にするために使用されます。
- measure_exprはメジャー列であり、任意の式を指定できます。メジャー列のNULL値は無視されます。
- delimiterには、メジャー列の値の区切りに使用する文字列を指定します。この句はオプションであり、デフォルトはNULLです。

  measure_exprがRAW型である場合、デリミタはRAW型である必要があります。そのためには、暗黙的にRAWに変換

できる文字列としてデリミタを指定するか、UTL_RAW.CAST_TO_RAW関数を使用するなどして、デリミタを明示的にRAWに変換します。

- order_by_clauseには、結合した値を戻す順序を指定します。このファンクションが決定的となるのは、ORDER BYの列リストの順序が一意になった場合のみです。
- order_by_clauseを指定する場合は、WITHIN GROUPも指定する必要があります。その逆も同様です。これらの2つの句は、まとめて指定するか、まったく指定しないでください。

DISTINCTキーワードを使用すると、重複する値がリストから削除されます。

メジャー列がRAW型である場合、戻り値のデータ型はRAWです。そうでない場合、戻りデータ型はVARCHAR2です。

戻りデータ型の最大長は、MAX_STRING_SIZE初期化パラメータの値に応じて異なります。MAX_STRING_SIZE = EXTENDEDの場合、VARCHAR2データ型およびRAWデータ型の最大長は32767バイトです。MAX_STRING_SIZE = STANDARDの場合、VARCHAR2データ型の最大長は4000バイトで、RAWデータ型の最大長は2000バイトです。戻り値が戻りデータ型に収まるかどうかを確認する場合、最終デリミタは含まれません。

関連項目:

- MAX_STRING_SIZE初期化パラメータの詳細は、「拡張データ型」を参照してください。
- LISTAGGの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。
- 詳細は、Oracle Databaseデータ・ウェアハウス・ガイドを参照してください。

listagg_overflow_clause

この句は、戻り値が戻りデータ型の最大長を超える場合の、ファンクションの動作を制御します。

ON OVERFLOW ERROR この句を指定すると、ORA-01489エラーが戻されます。これはデフォルトです。

ON OVERFLOW TRUNCATE この句を指定すると、メジャー値の切り捨てられたリストが戻されます。

- truncation_indicatorには、メジャー値の切り捨てられたリストに追加する文字列を指定します。この句を省略すると、切り捨てインジケータは省略記号(...)になります。

  measure_exprがRAW型である場合、切捨てインジケータはRAW型である必要があります。そのためには、暗黙的にRAWに変換できる文字列として切捨てインジケータを指定するか、UTL_RAW.CAST_TO_RAW関数を使用するなどして、切捨てインジケータを明示的にRAWに変換します。

- WITH COUNTを指定した場合、切捨てインジケータの後に、切り捨てられた値の数がカッコで囲まれて追加されます。この場合、戻り値に最終デリミタ、切り捨てインジケータおよびカッコで囲まれた24文字の数値のスペースを確保するのに十分なメジャー値が切り捨てられます。
- WITHOUT COUNTを指定した場合、戻り値から切り捨てられた値の数が省略されます。この場合、戻り値に最終デリミタおよび切り捨てインジケータのスペースを確保するのに十分なメジャー値が切り捨てられます。

WITH COUNTまたはWITHOUT COUNTを指定しない場合は、デフォルトでWITH COUNTが設定されます。

集計の例

次の単一セットの集計例は、hr.employees表に含まれる部門30の全従業員を、雇用開始日と姓の順にリストします。

```
SELECT LISTAGG(last_name, '; ')
```

```
         WITHIN GROUP (ORDER BY hire_date, last_name) "Emp_list",
       MIN(hire_date) "Earliest"
  FROM employees
  WHERE department_id = 30;
Emp_list                                                    Earliest
----------------------------------------------------------- ---------
Raphaely; Khoo; Tobias; Baida; Himuro; Colmenares           07-DEC-02
```

次のグループ・セットの集計例は、hr.employees表の部門IDごとに、各部門の従業員を雇用開始日の順にリストします。

```
SELECT department_id "Dept.",
       LISTAGG(last_name, '; ') WITHIN GROUP (ORDER BY hire_date) "Employees"
  FROM employees
  GROUP BY department_id
  ORDER BY department_id;
Dept. Employees
------ --------------------------------------------------------------
    10 Whalen
    20 Hartstein; Fay
    30 Raphaely; Khoo; Tobias; Baida; Himuro; Colmenares
    40 Mavris
    50 Kaufling; Ladwig; Rajs; Sarchand; Bell; Mallin; Weiss; Davie
       s; Marlow; Bull; Everett; Fripp; Chung; Nayer; Dilly; Bissot
       ; Vollman; Stiles; Atkinson; Taylor; Seo; Fleaur; Matos; Pat
       el; Walsh; Feeney; Dellinger; McCain; Vargas; Gates; Rogers;
        Mikkilineni; Landry; Cabrio; Jones; Olson; OConnell; Sulliv
       an; Mourgos; Gee; Perkins; Grant; Geoni; Philtanker; Markle
    60 Austin; Hunold; Pataballa; Lorentz; Ernst
    70 Baer
. . .
```

次の例は、ON OVERFLOW TRUNCATE句が含まれることを除き、前の例と同じです。この例の目的のために、戻り値の最大長を、意図的に200バイトという小さい数にしています。部門50の従業員のリストは200バイトを超えるため、リストは切り捨てられ、最終デリミタ'; '、指定された切り捨てインジケータ'...'および切り捨てられた値の数'(23)'が追加されます。

```
SELECT department_id "Dept.",
       LISTAGG(last_name, '; ' ON OVERFLOW TRUNCATE '...')
               WITHIN GROUP (ORDER BY hire_date) "Employees"
  FROM employees
  GROUP BY department_id
  ORDER BY department_id;
Dept. Employees
------ --------------------------------------------------------------
    10 Whalen
    20 Hartstein; Fay
    30 Raphaely; Khoo; Tobias; Baida; Himuro; Colmenares
    40 Mavris
    50 Kaufling; Ladwig; Rajs; Sarchand; Bell; Mallin; Weiss; Davie
       s; Marlow; Bull; Everett; Fripp; Chung; Nayer; Dilly; Bissot
       ; Vollman; Stiles; Atkinson; Taylor; Seo; Fleaur; ... (23)
    70 Baer
. . .
```

分析の例

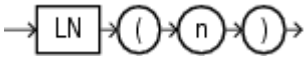
次の分析例は、2003年9月1日より前に雇用された各従業員について、所属部門、雇用開始日、および同じ部門で2003年9月1日より前に雇用された他の全従業員を表示します。

```
SELECT department_id "Dept", hire_date "Date", last_name "Name",
       LISTAGG(last_name, '; ') WITHIN GROUP (ORDER BY hire_date, last_name)
         OVER (PARTITION BY department_id) as "Emp_list"
  FROM employees
  WHERE hire_date < '01-SEP-2003'
  ORDER BY "Dept", "Date", "Name";
```

```
 Dept Date      Name            Emp_list
----- --------- --------------- ----------------------------------------------
   30 07-DEC-02 Raphaely        Raphaely; Khoo
   30 18-MAY-03 Khoo            Raphaely; Khoo
   40 07-JUN-02 Mavris          Mavris
   50 01-MAY-03 Kaufling        Kaufling; Ladwig
   50 14-JUL-03 Ladwig          Kaufling; Ladwig
   70 07-JUN-02 Baer            Baer
   90 13-JAN-01 De Haan         De Haan; King
   90 17-JUN-03 King            De Haan; King
  100 16-AUG-02 Faviet          Faviet; Greenberg
  100 17-AUG-02 Greenberg       Faviet; Greenberg
  110 07-JUN-02 Gietz           Gietz; Higgins
  110 07-JUN-02 Higgins         Gietz; Higgins
```

LN

構文



目的

LNは、nの自然対数を戻します(nは正の数)。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数がBINARY_FLOATの場合、このファンクションはBINARY_DOUBLEを戻します。それ以外の場合、引数と同じ数値データ型を戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

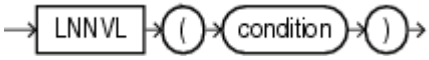
例

次の例では、95の自然対数を戻します。

```
SELECT LN(95) "Natural log of 95"
  FROM DUAL;
Natural log of 95
-----------------
       4.55387689
```

LNNVL

構文



目的

LNNVLは、条件のオペランドの1つまたは両方がNULLの可能性がある場合にその条件を簡単に評価する方法を提供します。このファンクションは問合せのWHERE句で、またはWHEN条件として検索CASE式で使用できます。このファンクションは、引数として条件を取り、その条件がFALSEまたはUNKNOWNの場合はTRUEを戻し、TRUEの場合はFALSEを戻します。LNNVLは、スカラー式を指定できる場所であればどこでも指定でき、有効ではないが、発生する可能性があるNULLを評価するためにIS [NOT] NULL、ANDまたはOR条件が必要であるようなコンテキストでも指定できます。

Oracle Databaseは、LNNVLファンクションをこの方法で内部的に使用して、NOT IN条件をNOT EXISTS条件として書き換える場合があります。この場合、EXPLAIN PLANからの出力によって、この操作がPLAN TABLEに出力されます。conditionでは、どのようなスカラー値でも評価できますが、AND、ORまたはBETWEENを含む複合条件は指定できません。

a = 2およびb=NULLの場合のLNNVLからの戻り値を次の表に示します。

条件	条件の真偽	LNNVLの戻り値
a = 1	FALSE	TRUE
a = 2	TRUE	FALSE
a IS NULL	FALSE	TRUE
b = 1	UNKNOWN	TRUE
b IS NULL	TRUE	FALSE
a = b	UNKNOWN	TRUE

例

歩合を受け取らない従業員を含めて、歩合率が20%未満の従業員数を調べるとします。次の問合せは、20%未満の歩合を実際に受け取る従業員のみを戻します。

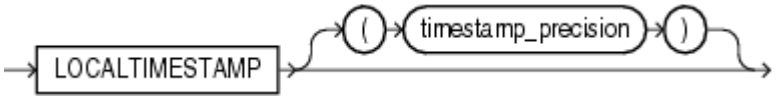
```
SELECT COUNT(*)
  FROM employees
  WHERE commission_pct < .2;
  COUNT(*)
----------
        11
```

歩合を受け取らない72人の従業員も含めるには、LNNVLファンクションを次のように使用して、この問合せを書き換えます。

```
SELECT COUNT(*)
  FROM employees
  WHERE LNNVL(commission_pct >= .2);
  COUNT(*)
----------
        83
```

LOCALTIMESTAMP

構文



目的

LOCALTIMESTAMPは、セッションのタイムゾーンの現在の日付および時刻をTIMESTAMPデータ型の値で戻します。このファンクションとCURRENT_TIMESTAMPとの違いは、CURRENT_TIMESTAMPは、TIMESTAMP WITH TIME ZONEの値を戻し、LOCALTIMESTAMPはTIMESTAMPの値を戻す点です。

オプションの引数timestamp_precisionには、戻される時刻値の秒の小数部の精度を指定します。

関連項目:

「CURRENT_TIMESTAMP」、「TIMESTAMPデータ型」、「TIMESTAMP WITH TIME ZONEデータ型」

例

次の例では、LOCALTIMESTAMPとCURRENT_TIMESTAMPの相違を示します。

```
ALTER SESSION SET TIME_ZONE = '-5:00';
SELECT CURRENT_TIMESTAMP, LOCALTIMESTAMP FROM DUAL;
CURRENT_TIMESTAMP                       LOCALTIMESTAMP
---------------------------------------------------------------------
04-APR-00 01.27.18.999220 PM -05:00  04-APR-00 01.27.19 PM
ALTER SESSION SET TIME_ZONE = '-8:00';
SELECT CURRENT_TIMESTAMP, LOCALTIMESTAMP FROM DUAL;
CURRENT_TIMESTAMP                       LOCALTIMESTAMP
---------------------------------  ----------------------------
04-APR-00 10.27.45.132474 AM -08:00  04-APR-00 10.27.451 AM
```

LOCALTIMESTAMPで書式マスクを使用する場合は、ファンクションが戻す値と書式マスクを一致させてください。たとえば、次の表の場合を考えます。

```
CREATE TABLE local_test (col1 TIMESTAMP WITH LOCAL TIME ZONE);
```

ファンクションが戻す型のTIME ZONEの部分がマスクに含まれていないため、次の文は正常に実行されません。

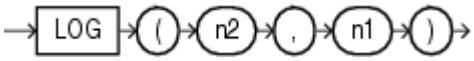
```
INSERT INTO local_test
  VALUES (TO_TIMESTAMP(LOCALTIMESTAMP, 'DD-MON-RR HH.MI.SSXFF'));
```

次の文では、LOCALTIMESTAMPの戻り値の型と一致する正しい書式マスクが使用されています。

```
INSERT INTO local_test
  VALUES (TO_TIMESTAMP(LOCALTIMESTAMP, 'DD-MON-RR HH.MI.SSXFF PM'));
```

LOG

構文



目的

LOGは、n2を底とするn1の対数を戻します。底n2は0(ゼロ)または1以外の任意の正の値で、n1は任意の正の値です。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。いずれかの引数がBINARY_FLOATまたはBINARY_DOUBLEの場合、このファンクションはBINARY_DOUBLEを戻します。それ以外の場合、NUMBERを戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

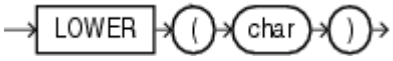
例

次の例では、100の対数を戻します。

```
SELECT LOG(10,100) "Log base 10 of 100"
  FROM DUAL;
Log base 10 of 100
------------------
                 2
```

LOWER

構文



目的

LOWERは、すべての文字を小文字にしてcharを戻します。charは、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBデータ型です。戻り値は、charと同じデータ型です。データベースは、基礎となる文字セットに対して定義したバイナリ・マッピングに基づいて文字の形式を設定します。小文字の区別については、「NLS_LOWER」を参照してください。

関連項目:

LOWERの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

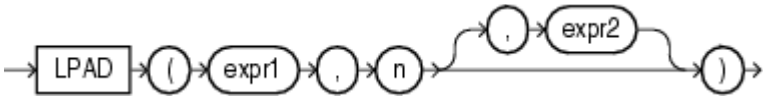
例

次の例では、文字列を小文字にして戻します。

```
SELECT LOWER('MR. SCOTT MCMILLAN') "Lowercase"
  FROM DUAL;
Lowercase
--------------------
mr. scott mcmillan
```

LPAD

構文



目的

LPADは、expr1の左側にexpr2に指定した文字を連続的に埋め込んでn桁にして戻します。このファンクションは、問合せの出力の書式設定に役立ちます。

expr1およびexpr2は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBデータ型です。戻される文字列は、expr1が文字データ型の場合はVARCHAR2データ型、expr1が各国語キャラクタ・データ型の場合はNVARCHAR2データ型、expr1がLOBデータ型の場合はLOBデータ型になります。expr1と同じ文字セットの文字列が戻されます。引数nは、NUMBER型の整数か、またはNUMBER型の整数に暗黙的に変換可能な値である必要があります。

expr2を指定しない場合、デフォルトで空白1個が指定されます。expr1がnより長い場合、このファンクションはnに収まるexpr1の一部を戻します。

引数nは、戻り値が画面に表示される場合の全体の長さです。多くの文字セットでは、これは戻り値の文字数でもあります。ただし、マルチバイトの文字セットでは、表示される文字列の長さが文字列の文字数と異なる場合もあります。

関連項目:

LPADの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

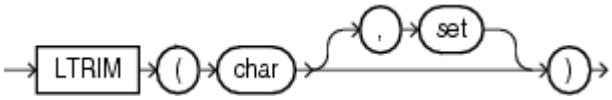
例

次の例では、文字列の左側にアスタリスクとピリオド(*.)を埋め込みます。

```
SELECT LPAD('Page 1',15,'*.') "LPAD example"
  FROM DUAL;
LPAD example
---------------
*.*.*.*.*Page 1
```

LTRIM

構文



目的

LTRIMは、charの左端から、setに指定されたすべての文字を削除します。setを指定しない場合、デフォルトで空白1個が指定されます。Oracle Databaseはcharの先頭文字からスキャンし始め、setに指定された文字をすべて削除します。setに指定された文字以外の文字が見つかった時点で結果を戻します。

charおよびsetは、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBデータ型です。戻される文字列は、charが文字データ型の場合はVARCHAR2データ型、charが各国語キャラクタ・データ型の場合はNVARCHAR2データ型、charがLOBデータ型の場合はLOBデータ型になります。

関連項目:

- RTRIM
- setの文字をcharの文字と比較するためにLTRIMで使用する照合を定義する照合決定ルール、およびこのファンクションの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

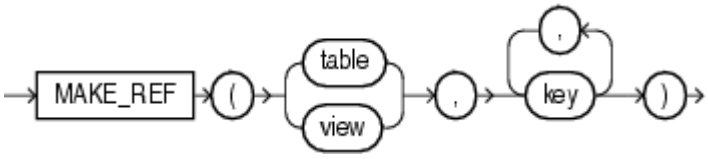
例

次の例では、文字列の左端にあるすべての小なり記号(<)、大なり記号(>)および等号(=)を文字列から削除します。

```
SELECT LTRIM('<=====>BROWNING<=====>', '<>=') "LTRIM Example"
  FROM DUAL;
LTRIM Example
---------------
BROWNING<=====>
```

MAKE_REF

構文



目的

MAKE_REFは、オブジェクト識別子が主キーに基づいている、オブジェクト・ビューの行またはオブジェクト表の行に対するREFを作成します。このファンクションは、オブジェクト・ビューを作成する場合などに有効です。

関連項目:

オブジェクト・ビューの詳細は、『Oracle Databaseオブジェクト・リレーショナル開発者ガイド』を参照してください。「DEREF」も参照してください。

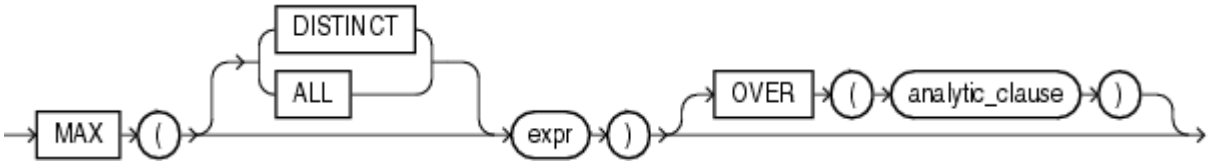
例

サンプル・スキーマoeには、inventory_typに基づくオブジェクト・ビューoc_inventoriesが格納されています。オブジェクト識別子はproduct_idです。次の例では、3003というproduct_idを含むオブジェクト・ビューoc_inventoriesにある行へのREFを作成します。

```
SELECT MAKE_REF (oc_inventories, 3003)
  FROM DUAL;
MAKE_REF(OC_INVENTORIES,3003)
--------------------------------------------------------------------------------
00004A038A0046857C14617141109EE03408002082543600000014260100010001
00290090606002A00078401FE0000000B03C21F040000000000000000000000000000
0000000000
```

MAX

構文



関連項目:

構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

MAXはexprの最大値を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

関連項目:

- exprの書式の詳細は、「SQL式」を参照してください。2進浮動小数点の比較セマンティクスの詳細は、「浮動小数点数」を参照してください。「集計ファンクション」も参照してください。
- exprの文字値を比較するためにMAXで使用する照合を定義する照合決定ルール、およびこのファンクションの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

集計の例

次の例では、hr.employees表の最高給与を検索します。

```
SELECT MAX(salary) "Maximum"
  FROM employees;

   Maximum
----------
     24000
```

分析の例

次の例では、各従業員について、その従業員と所属が同じ従業員のうち、一番高い給与を計算します。

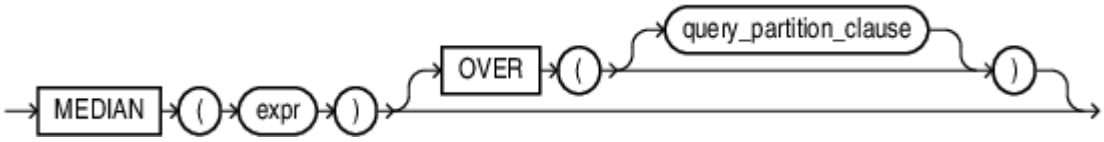
```
SELECT manager_id, last_name, salary,
       MAX(salary) OVER (PARTITION BY manager_id) AS mgr_max
  FROM employees
  ORDER BY manager_id, last_name, salary;
MANAGER_ID LAST_NAME                 SALARY    MGR_MAX
---------- ------------------------- ---------- ----------
       100 Cambrault                  11000      17000
       100 De Haan                    17000      17000
       100 Errazuriz                  12000      17000
       100 Fripp                       8200      17000
       100 Hartstein                  13000      17000
       100 Kaufling                    7900      17000
       100 Kochhar                    17000      17000
. . .
```

この問合せを述語のある親問合せで囲むと、各部門で給与の一番高い従業員がわかります。

```
SELECT manager_id, last_name, salary
  FROM (SELECT manager_id, last_name, salary,
               MAX(salary) OVER (PARTITION BY manager_id) AS rmax_sal
          FROM employees)
  WHERE salary = rmax_sal
  ORDER BY manager_id, last_name, salary;
MANAGER_ID LAST_NAME                 SALARY
---------- ------------------------ ----------
       100 De Haan                   17000
       100 Kochhar                   17000
       101 Greenberg                 12008
       101 Higgins                   12008
       102 Hunold                     9000
       103 Ernst                      6000
       108 Faviet                     9000
       114 Khoo                       3100
       120 Nayer                      3200
       120 Taylor                     3200
       121 Sarchand                   4200
       122 Chung                      3800
       123 Bell                       4000
       124 Rajs                       3500
       145 Tucker                    10000
       146 King                      10000
       147 Vishney                   10500
       148 Ozer                      11500
       149 Abel                      11000
       201 Fay                        6000
       205 Gietz                      8300
           King                      24000

22 rows selected.
```

MEDIAN

構文



関連項目:

構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

MEDIANは、連続分散モデルを想定する逆分散関数です。このファンクションは、数値または日時値を取り、その中央値、または値のソート後に中央値となる補間された値を戻します。計算では、NULLは無視されます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。exprのみを指定した場合、このファンクションは、引数の数値データ型と同じデータ型を戻します。OVER句を指定した場合、Oracle Databaseは、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。数値の優先順位の詳細は、「数値の優先順位の詳細」を参照してください。

MEDIANの結果の計算では、まず行が順序付けされます。グループ内の行数としてNが使用され、対象の行番号(RN)がRN = (1 + (0.5×(N-1))という式で計算されます。集計ファンクションの最終結果は、行番号がCRN = CEILING(RN)およびFRN = FLOOR(RN)の行の値間の直線補間によって計算されます。

最終結果は次のとおりです。

```
   if (CRN = FRN = RN) then
      (value of expression from row at RN)
   else
      (CRN - RN) * (value of expression for row at FRN) +
      (RN - FRN) * (value of expression for row at CRN)
```

MEDIANは、分析ファンクションとして使用できます。その場合、OVER句には、query_partition_clauseのみを指定できます。各行に対して、各パーティション内の一連の値の中央値に該当する値が戻されます。

このファンクションと次のファンクションを比較してください。

- PERCENTILE_CONT: 指定したパーセンタイルに一致するように補間された値が戻されます。MEDIANは、パーセンタイル値がデフォルトで0.5に指定される特別なPERCENTILE_CONTです。
- PERCENTILE_DISC: 指定したパーセンタイルに一致する値を補間なしで検索する場合に役立ちます。

集計の例

次の問合せは、hr.employees表内の部門ごとに給与の中央値を戻します。

```
SELECT department_id, MEDIAN(salary)
  FROM employees
  GROUP BY department_id
  ORDER BY department_id;
DEPARTMENT_ID MEDIAN(SALARY)
------------- --------------
           10           4400
           20           9500
           30           2850
           40           6500
           50           3100
           60           4800
           70          10000
           80           8900
           90          17000
          100           8000
          110          10154
                        7000
```

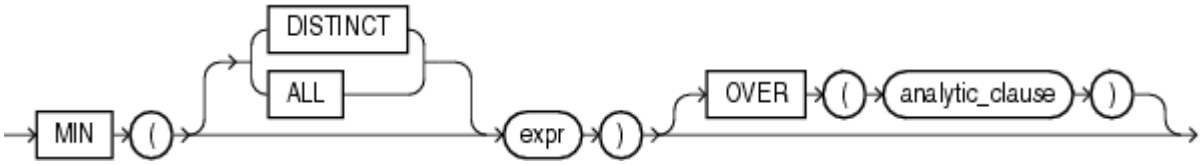
分析の例

次の問合せは、hr.employees表内の部門のサブセットのマネージャごとに給与の中央値を戻します。

```
SELECT manager_id, employee_id, salary,
       MEDIAN(salary) OVER (PARTITION BY manager_id) "Median by Mgr"
  FROM employees
  WHERE department_id > 60
  ORDER BY manager_id, employee_id;
MANAGER_ID EMPLOYEE_ID     SALARY Median by Mgr
---------- ----------- ---------- -------------
       100         101      17000         13500
       100         102      17000         13500
       100         145      14000         13500
       100         146      13500         13500
       100         147      12000         13500
       100         148      11000         13500
       100         149      10500         13500
       101         108      12008         12008
       101         204      10000         12008
       101         205      12008         12008
       108         109       9000          7800
       108         110       8200          7800
       108         111       7700          7800
       108         112       7800          7800
       108         113       6900          7800
       145         150      10000          8500
       145         151       9500          8500
       145         152       9000          8500
. . .
```

MIN

構文



関連項目:

構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

MINは、exprの最小値を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

関連項目:

- exprの書式の詳細は、「SQL式」を参照してください。2進浮動小数点の比較セマンティクスの詳細は、「浮動小数点数」を参照してください。「集計ファンクション」も参照してください。
- exprの文字値を比較するためにMINで使用する照合を定義する照合決定ルール、およびこのファンクションの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

集計の例

次の例では、hr.employees表の最初の雇用開始日を戻します。

```
SELECT MIN(hire_date) "Earliest"
  FROM employees;

Earliest
---------
13-JAN-01
```

分析の例

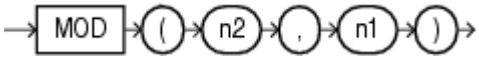
次の例では、各従業員について、その従業員が雇用された日以前に雇用された従業員を検索します。その従業員と所属が同じ従業員のサブセットを決定し、そのサブセット内で一番低い給与を戻します。

```
SELECT manager_id, last_name, hire_date, salary,
       MIN(salary) OVER(PARTITION BY manager_id ORDER BY hire_date
         RANGE UNBOUNDED PRECEDING) AS p_cmin
  FROM employees
  ORDER BY manager_id, last_name, hire_date, salary;
MANAGER_ID LAST_NAME                 HIRE_DATE     SALARY     P_CMIN
---------- ------------------------- --------- ---------- ----------
       100 Cambrault                 15-OCT-07      11000       6500
       100 De Haan                   13-JAN-01      17000      17000
       100 Errazuriz                 10-MAR-05      12000       7900
       100 Fripp                     10-APR-05       8200       7900
       100 Hartstein                 17-FEB-04      13000       7900
       100 Kaufling                  01-MAY-03       7900       7900
       100 Kochhar                   21-SEP-05      17000       7900
```

```
       100 Mourgos                   16-NOV-07       5800       5800
       100 Partners                  05-JAN-05      13500       7900
       100 Raphaely                  07-DEC-02      11000      11000
       100 Russell                   01-OCT-04      14000       7900
. . .
```

MOD

構文



目的

MODはn2をn1で割った余りを戻します。n1が0の場合は、n2を戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。Oracleは、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。数値の優先順位の詳細は、「数値の優先順位の詳細」を参照してください。

例

次の例では、11を4で割ったときの余りを戻します。

```
SELECT MOD(11,4) "Modulus"
  FROM DUAL;
   Modulus
----------
         3
```

このファンクションは、n1およびn2の積が負の値の場合には古典数学のモジュール・ファンクションとは異なる動作をします。古典数学のモジュール・ファンクションは、次の公式を使用したMODファンクションで表すことができます。

```
n2 - n1 * FLOOR(n2/n1)
```

次の表に、MODファンクションと古典数学のモジュール・ファンクションの相違を示します。

n2	n1	MOD(n2,n1)	古典数学のモジュール・ファンクション
11	4	3	3
11	-4	3	-1
-11	4	-3	1
-11	-4	-3	-3

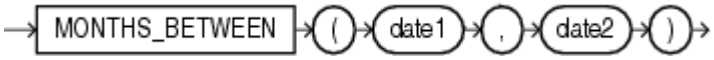
関連項目:

「FLOOR」および「REMAINDER」を参照してください。REMAINDERはMODと似ていますが、FLOORではなくROUNDを式に

使用します。

MONTHS_BETWEEN

構文



目的

MONTHS_BETWEENは、日付date1とdate2の間の月数を戻します。月および月の最終日は、パラメータNLS_CALENDARによって定義されます。date1がdate2より後の日付の場合、結果は正の値になります。date1がdate2より前の日付の場合、結果は負の値になります。date1およびdate2が、月の同じ日または月の最終日の場合、結果は常に整数になります。それ以外の場合、Oracle Databaseは結果の小数部を1か月31日として計算し、date1とdate2の差を割り出します。

例

次の例では、2つの日付間の月数を計算します。

```
SELECT MONTHS_BETWEEN
       (TO_DATE('02-02-1995','MM-DD-YYYY'),
        TO_DATE('01-01-1995','MM-DD-YYYY') ) "Months"
  FROM DUAL;
    Months
----------
1.03225806
```

NANVL

構文

目的

NANVLファンクションは、浮動小数点数のタイプがBINARY_FLOATまたはBINARY_DOUBLEの場合にのみ便利です。このファンクションは、入力値n2がNaN(非数値)の場合は代替値n1を戻すようにOracle Databaseに指示します。n2がNaNでない場合、Oracleはn2を戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。Oracleは、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。2進浮動小数点の比較セマンティクスの詳細は、「浮動小数点数」を参照してください。数値の優先順位の詳細は、「数値の優先順位」を参照してください。

例

「TO_BINARY_DOUBLE」で作成した表float_point_demoを使用して、表に2つ目のエントリを挿入します。

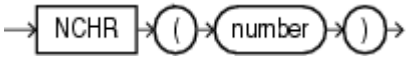
```
INSERT INTO float_point_demo
  VALUES (0,'NaN','NaN');
SELECT *
  FROM float_point_demo;
   DEC_NUM BIN_DOUBLE  BIN_FLOAT
---------- ---------- ----------
   1234.56 1.235E+003 1.235E+003
         0        Nan        Nan
```

次の例では、数値の場合はbin_floatを戻します。それ以外の場合は、0を戻します。

```
SELECT bin_float, NANVL(bin_float,0)
  FROM float_point_demo;
 BIN_FLOAT NANVL(BIN_FLOAT,0)
---------- ------------------
1.235E+003         1.235E+003
       Nan                  0
```

NCHR

構文



目的

NCHRは、各国語文字セットのnumberと同等のバイナリを持つ文字を戻します。戻り値は常にNVARCHAR2です。このファンクションは、CHRファンクションにUSING NCHAR_CS句を指定して使用した場合と同じ結果を戻します。

このファンクションは、引数として、NUMBER値、または暗黙的にNUMBER型に変換可能な任意の値を取り、文字を戻します。

関連項目:

- CHR
- NCHRの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

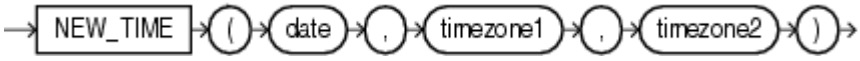
例

次の例では、NCHAR文字187を戻します。

```
SELECT NCHR(187)
  FROM DUAL;
N
-
>
SELECT CHR(187 USING NCHAR_CS)
  FROM DUAL;
C
-
>
```

NEW_TIME

構文



目的

NEW_TIMEは、タイムゾーンtimezone1の日時がdateの時点のタイムゾーンtimezone2の日時を戻します。このファンクションを使用する前に、24時間で表示されるように、NLS_DATE_FORMATパラメータを設定する必要があります。戻り型は、dateのデータ型に関係なく常にDATEです。

> ノート:
>
> このファンクションが入力として取ることのできるタイムゾーンの数には制限があります。FROM_TZ ファンクションと日時式を組み合せることによって、より多くのタイムゾーンにアクセスできます。「FROM_TZ」および「日時式」の例を参照してください。

引数timezone1およびtimezone2には、次のテキスト文字列のいずれかを指定できます。

- AST、ADT: 大西洋標準時および大西洋夏時間
- BST、BDT: ベーリング標準時およびベーリング夏時間
- CST、CDT: 中央標準時および中央夏時間
- EST、EDT: 東部標準時および東部夏時間
- GMT: グリニッジ標準時
- HST、HDT: アラスカ-ハワイ標準時およびアラスカ-ハワイ夏時間。
- MST、MDT: 山岳部標準時および山岳部夏時間
- NST: ニューファンドランド標準時
- PST、PDT: 太平洋標準時および太平洋夏時間
- YST、YDT: ユーコン標準時およびユーコン夏時間

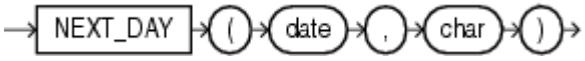
例

次の例では、指定された太平洋標準時と同等の大西洋標準時を戻します。

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
SELECT NEW_TIME(TO_DATE('11-10-09 01:23:45', 'MM-DD-YY HH24:MI:SS'), 'AST', 'PST')
          "New Date and Time"
  FROM DUAL;
New Date and Time
--------------------
09-NOV-2009 21:23:45
```

NEXT_DAY

構文



目的

NEXT_DAYは、charで指定した曜日で、日付dateより後の最初の日付を戻します。戻り型は、dateのデータ型に関係なく常にDATEです。引数charは、セッションの日付言語での曜日である必要があります(フルネームでも省略形でも可)。必要最小限の文字数は、省略形の文字数です。有効な省略形の後に続けて文字が入力されていても、それらの文字は無視されます。戻り値は、引数dateと同じ時、分および秒のコンポーネントを持っています。

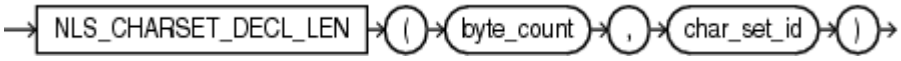
例

次の例では、2009年10月15日以降の最初の火曜日の日付を戻します。

```
SELECT NEXT_DAY('15-OCT-2009','TUESDAY') "NEXT DAY"
  FROM DUAL;
NEXT DAY
--------------------
20-OCT-2009 00:00:00
```

NLS_CHARSET_DECL_LEN

構文



目的

NLS_CHARSET_DECL_LENは、NCHAR列の宣言の長さを(文字数で)戻します。byte_count引数は、列の幅です。char_set_id引数は、列の文字セットIDです。

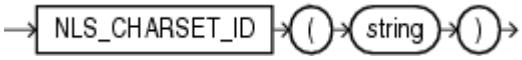
例

次の例では、マルチバイト文字セットを使用している場合に、列の幅が200バイトである文字の数を戻します。

```
SELECT NLS_CHARSET_DECL_LEN(200, nls_charset_id('ja16eucfixed'))
  FROM DUAL;
NLS_CHARSET_DECL_LEN(200,NLS_CHARSET_ID('JA16EUCFIXED'))
--------------------------------------------------------
                                                     100
```

NLS_CHARSET_ID

構文



目的

NLS_CHARSET_IDは、文字セット名stringに対応する文字セットのID番号を戻します。引数stringは、実行時のVARCHAR2値です。string値'CHAR_CS'は、サーバーのデータベース文字セットのID番号を戻します。string値'NCHAR_CS'は、サーバーの各国語文字セットのID番号を戻します。

無効な文字セット名を指定すると、NULLが戻されます。

関連項目:

文字セットのリストについては、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

例

次の例では、文字セットのIDを戻します。

```
SELECT NLS_CHARSET_ID('ja16euc')
  FROM DUAL;
NLS_CHARSET_ID('JA16EUC')
-------------------------
                      830
```

NLS_CHARSET_NAME

構文

目的

NLS_CHARSET_NAMEは、ID番号numberに対応する文字セット名を戻します。文字セット名は、データベース文字セットのVARCHAR2値として戻されます。numberが有効な文字セットIDとして認識されない場合は、このファンクションはNULLを戻します。

このファンクションは、VARCHAR2値を返します。

関連項目:

NLS_CHARSET_NAMEの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、文字セットのID番号2に対応する文字セットを戻します。

```
SELECT NLS_CHARSET_NAME(2)
  FROM DUAL;
NLS_CH
------
WE8DEC
```

NLS_COLLATION_ID

構文

目的

NLS_COLLATION_IDでは、引数として照合名を取得し、対応する照合ID番号を戻します。照合IDは、データ・ディクショナリ表およびOracle Call Interface (OCI)で使用されます。照合名は、SQL文およびデータ・ディクショナリ・ビューで使用されます。

exprには、照合名をVARCHAR2値として指定します。有効な名前付き照合を、大文字と小文字を任意に組み合せて指定できます。

このファンクションは、NUMBER値を返します。無効な照合名を指定した場合、ファンクションはNULLを戻します。

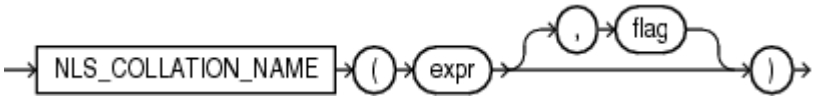
例

次の例では、照合BINARY_CIの照合IDを戻します。

```
SELECT NLS_COLLATION_ID('BINARY_CI')
  FROM DUAL;
NLS_COLLATION_ID('BINARY_CI')
-----------------------------
                       147455
```

NLS_COLLATION_NAME

構文



目的

NLS_COLLATION_NAMEでは、引数として照合ID番号を取得し、対応する照合名を戻します。照合IDは、データ・ディクショナリ表およびOracle Call Interface (OCI)で使用されます。照合名は、SQL文およびデータ・ディクショナリ・ビューで使用されます。

exprには、照合IDをNUMBER値として指定します。

このファンクションは、VARCHAR2値を返します。無効な照合IDを指定した場合、ファンクションはNULLを戻します。

オプションのflagパラメータは、Unicode Collation Algorithm (UCA)照合にのみ適用されます。このパラメータは、ファンクションが短い形式と長い形式のどちらの照合名を戻すかを指定します。パラメータは、次の意味を持つ'S'、's'、'L'または'l'に評価される文字式である必要があります。

- 'S'または's' – 短い形式の照合名を戻します。
- 'L'または'l' – 長い形式の照合名を戻します。

flagを指定しない場合、デフォルトは'L'です。

関連項目:

- UCA照合の詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。
- NLS_COLLATION_NAMEの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、照合ID番号81919に対応する照合の名前を戻します。

```
SELECT NLS_COLLATION_NAME(81919)
  FROM DUAL;
NLS_COLLA
---------
BINARY_AI
```

次の例では、照合ID番号208897に対応するUCA照合の短い形式の名前を戻します。

```
SELECT NLS_COLLATION_NAME(208897,'S')
  FROM DUAL;
NLS_COLLATION
-------------
UCA0610_DUCET
```

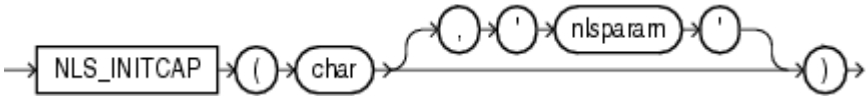
次の例では、照合ID番号208897に対応するUCA照合の長い形式の名前を戻します。

```
SELECT NLS_COLLATION_NAME(208897,'L')
  FROM DUAL;
NLS_COLLATION_NAME(208897,'L')
```

```
---------------------------------------
UCA0610_DUCET_S4_VS_BN_NY_EN_FN_HN_DN_MN
```

NLS_INITCAP

構文



目的

NLS_INITCAPは、各単語の最初の文字を大文字、残りの文字を小文字にしてcharを戻します。単語は空白または英数字以外の文字で区切ります。

charおよび'nlsparam'は、CHAR、VARCHAR2、NCHARまたはNVARCHAR2データ型です。戻される文字列は、VARCHAR2データ型であり、charと同じ文字セットです。

'nlsparam'の値は次の書式で指定します。

```
'NLS_SORT = sort'
```

sortは名前付き照合です。照合では、大文字と小文字の変換のために特別な言語要件を処理します。これらの要件によって、charと異なる長さの値が戻される場合があります。'nlsparam'を指定しない場合、ファンクションの決定済照合が使用されます。

このファンクションは、CLOBデータを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用してCLOBを引数として渡すことはできます。

関連項目:

- 詳細は、「データ型の比較規則」を参照してください。
- NLS_INITCAPの照合決定ルール、およびこのファンクションの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次に、ファンクションによって言語ソート基準がどのように異なる値を戻すかを示します。

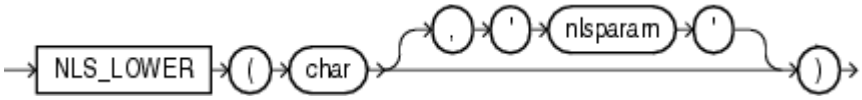
```
SELECT NLS_INITCAP('ijsland') "InitCap"
  FROM DUAL;
InitCap
-------
Ijsland
SELECT NLS_INITCAP('ijsland', 'NLS_SORT = XDutch') "InitCap"
  FROM DUAL;
InitCap
-------
IJsland
```

関連項目:

照合の詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

NLS_LOWER

構文



目的

NLS_LOWERは、すべての文字を小文字にしてcharを戻します。

charおよび'nlsparam'は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBデータ型です。戻される文字列は、charが文字データ型の場合はVARCHAR2データ型になり、charがLOBデータ型の場合はLOBになります。charと同じ文字セットの文字列が戻されます。

'nlsparam'の書式および用途は、NLS_INITCAPファンクションと同じです。

関連項目:

NLS_LOWERの照合決定ルール、およびこのファンクションの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

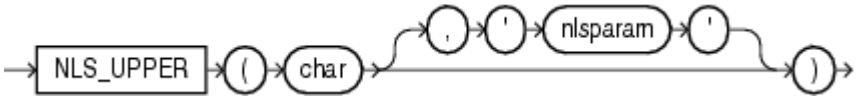
例

次の例では、XTurkish言語ソート順序を使用する文字列'NOKTASINDA'を小文字形式で戻します。トルコ語の大文字Iは、点のない小文字のiになります。

```
SELECT NLS_LOWER('NOKTASINDA', 'NLS_SORT = XTurkish') "Lowercase"
  FROM DUAL;
```

NLS_UPPER

構文



目的

NLS_UPPERは、すべての文字を大文字にしてcharを戻します。

charおよび'nlsparam'は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBデータ型です。戻される文字列は、charが文字データ型の場合はVARCHAR2データ型になり、charがLOBデータ型の場合はLOBになります。charと同じ文字セットの文字列が戻されます。

'nlsparam'の書式および用途は、NLS_INITCAPファンクションと同じです。

関連項目:

NLS_UPPERの照合決定ルール、およびこのファンクションの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、すべての文字を大文字に変換して文字列を戻します。

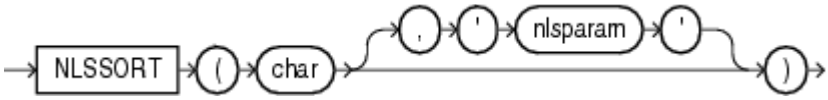
```
SELECT NLS_UPPER('große') "Uppercase"
  FROM DUAL;
Upper
-----
GROßE
SELECT NLS_UPPER('große', 'NLS_SORT = XGerman') "Uppercase"
  FROM DUAL;
Upperc
------
GROSSE
```

関連項目:

NLS_INITCAP

NLSSORT

構文



目的

NLSSORTは、文字値charと明示的または暗黙的に指定された照合に対する照合キーを返します。照合キーは、指定された照合に従ってcharをソートするために使用されるバイト文字列です。照合キーのプロパティでは、特定の照合に対して生成された2つの照合キーをバイナリ順に従って比較したときのそれらのキーの相互順が、その特定の照合に従って比較したソース文字値の相互順と等しくなります。

charおよび'nlsparam'は、CHAR、VARCHAR2、NCHARまたはNVARCHAR2データ型です。

'nlsparam'の値は次の書式で指定する必要があります。

```
'NLS_SORT = collation'
```

collationは、言語照合の名前またはBINARYです。NLSSORTは、指定された照合を使用して照合キーを生成します。'nlsparam'を指定しない場合、引数charの導出照合が使用されます。BINARYを指定すると、後述のようにRAWにキャストされて必要に応じて切り捨てられたchar値自体が返されます。

'nlsparam'を指定した場合、言語照合名に接尾辞_aiを追加してアクセント記号の有無を区別しない照合を行うか、または_ciを追加して大/小文字を区別しない照合を行うことができます。アクセント記号の有無および大/小文字を区別しないソートの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。ORDER BY問合せ句でアクセント記号の有無および大/小文字を区別しない照合を使用するとソート順が非決定的になるため、お薦めしません。

返される照合キーは、RAWデータ型です。特定の照合について特定のchar値から生成される照合キーの長さが、NLSSORTで返されるRAW値の最大長を超過することがあります。この場合、NLSSORTの動作は初期化パラメータMAX_STRING_SIZEの値に依存します。MAX_STRING_SIZE = EXTENDEDの場合、戻り値の最大長は32767バイトです。照合キーがこの制限を超えると、「ORA-12742: 照合キーを作成できません」エラーが発生してファンクションの実行が失敗します。入力文字列が短くても、その文字列に分解の比率が非常に高いUnicode文字が高い割合で含まれている場合には、このエラーが発生することがあります。

関連項目:

ORA-12742がいつ報告されるか、およびエラーが原因で発生する可能性があるアプリケーションの可用性の問題を回避する方法の詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

MAX_STRING_SIZE = STANDARDの場合、戻り値の最大長は2000バイトです。返される値がこの制限を超える場合、NLSSORTは、計算結果が最大長を超えないように、charの最長の接頭辞(先頭のサブストリング)についての照合キーを計算します。単一言語の照合(たとえばFRENCH)の場合、通常、接頭辞の長さは1000文字です。複数言語の照合(たとえばGENERIC_M)の場合、通常、接頭辞の長さは500文字です。Unicode照合アルゴリズム(UCA)に基づく照合(たとえばUCA0610_DUCET)の場合、通常、接頭辞の長さは285文字です。照合およびcharに含まれる文字によっては、正確な長さが短くなったり長くなる場合があります。

MAX_STRING_SIZE = STANDARDの場合の動作は、言語の順序付けを決定するためにその照合キー(NLSSORTの結

果)が比較される2つの文字値が、正確な文字位置は異なっていても、接頭辞が同じである場合に、等しいとみなされることを意味します。NLSSORTファンクションは、比較条件、BETWEEN条件、IN条件、ORDER BY、GROUP BYおよびCOUNT(DISTINCT)での言語の順序付けを決定するために暗黙的に使用されるため、長い文字値に対しては、これらの操作でおよその結果のみが返される場合があります。これらの操作から確実に正確な結果を得るには、MAX_STRING_SIZE = EXTENDEDを使用するようにデータベースを移行します。

MAX_STRING_SIZE初期化パラメータの詳細は、「拡張データ型」を参照してください。

このファンクションは、CLOBデータを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用してCLOBを引数として渡すことはできます。

関連項目:

- 詳細は、「データ型の比較規則」を参照してください。
- NLSSORTの照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

このファンクションを使用すると、文字列の2進値を基にしたソートおよび比較ではなく、言語ソート基準を基にしたソートおよび比較を指定できます。次の例では、2つの値を含むテスト表を作成し、戻される値がNLSSORTファンクションによってどのように順序付けられるかを示します。

```
CREATE TABLE test (name VARCHAR2(15));
INSERT INTO test VALUES ('Gaardiner');
INSERT INTO test VALUES ('Gaberd');
INSERT INTO test VALUES ('Gaasten');
SELECT *
  FROM test
  ORDER BY name;
NAME
---------------
Gaardiner
Gaasten
Gaberd
SELECT *
  FROM test
  ORDER BY NLSSORT(name, 'NLS_SORT = XDanish');
NAME
---------------
Gaberd
Gaardiner
Gaasten
```

次の例では、比較操作でのNLSSORTファンクションの使用方法を示します。

```
SELECT *
  FROM test
  WHERE name > 'Gaberd'
  ORDER BY name;
no rows selected
SELECT *
  FROM test
  WHERE NLSSORT(name, 'NLS_SORT = XDanish') >
        NLSSORT('Gaberd', 'NLS_SORT = XDanish')
  ORDER BY name;
NAME
---------------
Gaardiner
```

```
Gaasten
```

同一の言語ソート基準を使用する比較操作で頻繁にNLSSORTを使用する場合、NLS_COMPパラメータ(データベース用か現行のセッション用のいずれか)にLINGUISTICを設定し、セッションのNLS_SORTパラメータに必要なソート基準を設定するとより効率的です。これによって、現行のセッション中のすべてのソート操作および比較操作において、デフォルトでそのソート基準が使用されるようになります。

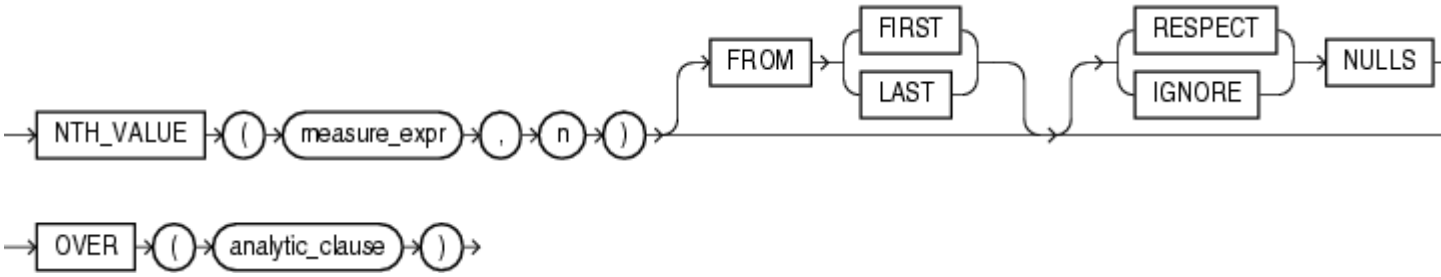
```
ALTER SESSION SET NLS_COMP = 'LINGUISTIC';
ALTER SESSION SET NLS_SORT = 'XDanish';
SELECT *
  FROM test
  WHERE name > 'Gaberd'
  ORDER BY name;
NAME
---------------
Gaardiner
Gaasten
```

関連項目:

ソート基準の詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

NTH_VALUE

構文



関連項目:

analytic_clauseの構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

NTH_VALUEは、analytic_clauseに定義したウィンドウ内でn番目の行のmeasure_expr値を戻します。戻り値のデータ型はmeasure_exprです。

- {RESPECT | IGNORE} NULLSは、measure_exprのNULL値を計算に含めるか除外するかを指定します。デフォルトはRESPECT NULLSです。
- nは、n番目の行についてメジャー値を戻すことを示します。nに指定できるのは、定数、バインド変数、列、またはこれらを含む式です。ただし、式の場合は正の整数に解決される必要があります。データソース・ウィンドウに含まれる行がnより少ない場合は、NULLが戻されます。nがNULLの場合は、エラーが戻されます。
- FROM {FIRST | LAST}では、計算をウィンドウ内の最初の行から開始するか、または最後の行から開始するかを指定します。デフォルトはFROM FIRSTです。

analytic_clauseのwindowing_clauseを省略した場合、デフォルトでRANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROWになります。このデフォルトは、NTH_VALUE ... FROM LAST ... に対して予期しない値を返すことがあります。これは、ウィンドウの最後の値は、ウィンドウの終端になり、その終端が固定されていないためです。これは、現行の行が変化するに伴って変化します。正しい結果を得るには、windowing_clauseをRANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWINGとして指定します。または、windowing_clauseをRANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWINGとして指定することもできます。

関連項目:

- このファンクションの使用の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- NTH_VALUEの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、2番目のchannel_idに対するamount_soldの最小値を、prod_id 13から16についてそれぞれ昇順で示します。

```
SELECT prod_id, channel_id, MIN(amount_sold),
```

```
    NTH_VALUE(MIN(amount_sold), 2) OVER (PARTITION BY prod_id ORDER BY channel_id
    ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) nv
  FROM sales
  WHERE prod_id BETWEEN 13 and 16
  GROUP BY prod_id, channel_id;
   PROD_ID CHANNEL_ID MIN(AMOUNT_SOLD)         NV
---------- ---------- ---------------- ----------
        13          2           907.34      906.2
        13          3            906.2      906.2
        13          4           842.21      906.2
        14          2          1015.94    1036.72
        14          3          1036.72    1036.72
        14          4           935.79    1036.72
        15          2           871.19     871.19
        15          3           871.19     871.19
        15          4           871.19     871.19
        16          2           266.84     266.84
        16          3           266.84     266.84
        16          4           266.84     266.84
        16          9            11.99     266.84
13 rows selected.
```

NTILE

構文

関連項目:

構文、セマンティクス、制限事項、およびexprの書式の詳細は、「分析ファンクション」を参照してください。

目的

NTILEは分析ファンクションです。これは、順序付けられたデータセットをexprに指定した数のバケットに分割し、適切なバケット番号を各行に割り当てます。バケットには1からexprの番号が付けられます。expr値は、パーティションごとに、正の定数に変換される必要があります。Oracle Databaseではexprは整数とみなされるため、この値が整数ではない定数の場合は整数に切り捨てられます。戻り値はNUMBERです。

バケット内の行数は、最大で1異なります。残りの値(バケットで割った行数の余り)は、バケット1から順に、1行ずつ分割されます。

exprが行数より大きい場合、行数と等しい数のバケットに行が入れられ、余りのバケットは空になります。

exprには、NTILEまたは他の分析ファンクションを使用して分析ファンクションをネストできません。ただし、他の組込みファンクション式をexprで使用できます。

関連項目:

exprの書式の詳細は、「SQL式」を参照してください。暗黙的な変換の詳細は、表2-8を参照してください。

例

次の例では、部門100のoe.employees表のsalary列の値を4つのバケットに分割します。この部門のsalary列には6つの値が存在するため、2つの余分な値(6を4で割った余り)は、バケット1および2に割り当てられます。そのため、バケット1および2は、バケット3または4より値が1つ多くなります。

```
SELECT last_name, salary, NTILE(4) OVER (ORDER BY salary DESC) AS quartile
  FROM employees
  WHERE department_id = 100
  ORDER BY last_name, salary, quartile;
LAST_NAME                 SALARY   QUARTILE
------------------------- ---------- ----------
Chen                            8200          2
Faviet                          9000          1
Greenberg                      12008          1
Popp                            6900          4
Sciarra                         7700          3
Urman                           7800          2
```

NULLIF

構文

目的

NULLIFは、expr1とexpr2を比較します。式が等しい場合、関数はNULLを返します。異なる場合は、expr1を戻します。expr1には、リテラルNULLを指定できません。

両方の引数が数値データ型である場合、Oracle Databaseは、数値の優先順位が高い方の引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。2つの引数が数値ではない場合、それらのデータ型が同じである必要があります。データ型が異なる場合、Oracleはエラーを戻します。

NULLIFファンクションは、次のCASE式と論理的に同じです。

```
CASE WHEN expr1 = expr2 THEN NULL ELSE expr1 END
```

関連項目:

- 「CASE式」
- exprの文字をexpr2の文字と比較するためにNULLIFで使用する照合を定義する照合決定ルール、およびこのファンクションの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

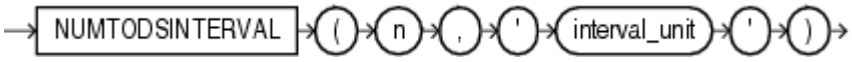
例

次の例では、サンプル・スキーマhrから、雇用後に職種が変更した従業員を検索します。これは、employees表の現行のjob_idがjob_history表のjob_idと異なるかどうかによって識別されます。

```
SELECT e.last_name, NULLIF(j.job_id, e.job_id) "Old Job ID"
  FROM employees e, job_history j
  WHERE e.employee_id = j.employee_id
  ORDER BY last_name, "Old Job ID";
LAST_NAME                 Old Job ID
------------------------- ----------
De Haan                   IT_PROG
Hartstein                 MK_REP
Kaufling                  ST_CLERK
Kochhar                   AC_ACCOUNT
Kochhar                   AC_MGR
Raphaely                  ST_CLERK
Taylor                    SA_MAN
Taylor
Whalen                    AC_ACCOUNT
Whalen
```

NUMTODSINTERVAL

構文



目的

NUMTODSINTERVALは、nをINTERVAL DAY TO SECONDリテラルに変換します。引数nには、任意のNUMBER値か、またはNUMBER値に暗黙的に変換可能な式を指定できます。引数interval_unitのデータ型は、CHAR、VARCHAR2、NCHARまたはNVARCHAR2です。interval_unitの値にはnの単位を指定します。この値は次の文字列値のいずれかである必要があります。

- 'DAY'
- 'HOUR'
- 'MINUTE'
- 'SECOND'

interval_unitでは、大/小文字は区別されません。カッコ内の先行値および後続値は無視されます。デフォルトでは、戻り値の精度は9です。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

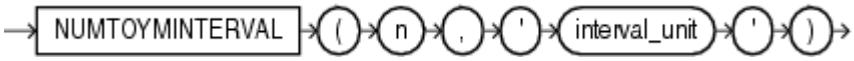
例

次の例では、COUNT分析ファンクションにNUMTODSINTERVALを使用し、各従業員について、雇用開始日から過去100日以内に同じマネージャの部下として配属された従業員の人数を計算します。分析ファンクションの構文の詳細は、「分析ファンクション」を参照してください。

```
SELECT manager_id, last_name, hire_date,
       COUNT(*) OVER (PARTITION BY manager_id ORDER BY hire_date
       RANGE NUMTODSINTERVAL(100, 'day') PRECEDING) AS t_count
  FROM employees
  ORDER BY last_name, hire_date;
MANAGER_ID LAST_NAME                 HIRE_DATE    T_COUNT
---------- ------------------------- --------- ----------
       149 Abel                      11-MAY-04          1
       147 Ande                      24-MAR-08          3
       121 Atkinson                  30-OCT-05          2
       103 Austin                    25-JUN-05          1
. . .
       124 Walsh                     24-APR-06          2
       100 Weiss                     18-JUL-04          1
       101 Whalen                    17-SEP-03          1
       100 Zlotkey                   29-JAN-08          2
```

NUMTOYMINTERVAL

構文



目的

NUMTOYMINTERVALは、数値nをINTERVAL YEAR TO MONTHリテラルに変換します。引数nには、任意のNUMBER値か、またはNUMBER値に暗黙的に変換可能な式を指定できます。引数interval_unitのデータ型は、CHAR、VARCHAR2、NCHARまたはNVARCHAR2です。interval_unitの値にはnの単位を指定します。この値は次の文字列値のいずれかである必要があります。

- 'YEAR'
- 'MONTH'

interval_unitでは、大/小文字は区別されません。カッコ内の先行値および後続値は無視されます。デフォルトでは、戻り値の精度は9です。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

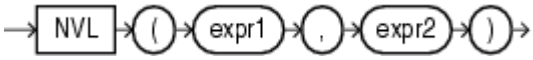
例

次の例では、SUM分析ファンクションにNUMTOYMINTERVALを使用し、各従業員について、その従業員の雇用日から過去1年の間に雇用された従業員の給与の合計を計算します。分析ファンクションの構文の詳細は、「分析ファンクション」を参照してください。

```
SELECT last_name, hire_date, salary,
       SUM(salary) OVER (ORDER BY hire_date
       RANGE NUMTOYMINTERVAL(1,'year') PRECEDING) AS t_sal
  FROM employees
  ORDER BY last_name, hire_date;
LAST_NAME                 HIRE_DATE     SALARY      T_SAL
------------------------- --------- ---------- ----------
Abel                      11-MAY-04      11000      90300
Ande                      24-MAR-08       6400     112500
Atkinson                  30-OCT-05       2800     177000
Austin                    25-JUN-05       4800     134700
. . .
Walsh                     24-APR-06       3100     186200
Weiss                     18-JUL-04       8000      70900
Whalen                    17-SEP-03       4400      54000
Zlotkey                   29-JAN-08      10500     119000
```

NVL

構文



目的

NVLを使用すると、NULL(空白として戻される)を文字列に置換して問合せの結果に含めることができます。expr1がnullの場合、NVLはexpr2を返します。expr1がnullでない場合、NVLはexpr1を返します。

引数expr1およびexpr2は任意のデータ型にすることができます。2つの引数のデータ型が異なる場合、一方のデータ型が他方のデータ型に暗黙的に変換されます。暗黙的に変換できない場合、データベースはエラーを戻します。暗黙的な変換は、次のように実行されます。

- expr1が文字データの場合、Oracle Databaseは2つの引数を比較する前にexpr2をexpr1のデータ型に変換し、expr1の文字セットでVARCHAR2を戻します。
- expr1が数値である場合、Oracle Databaseは数値の優先順位が最も高い引数を判断し、その引数のデータ型に他方の引数を暗黙的に変換して、そのデータ型を戻します。

関連項目:

- 暗黙的な変換の詳細は、表2-8を参照してください。数値の優先順位の詳細は、「数値の優先順位の詳細」を参照してください。
- 「COALESCE」および「CASE式」を参照してください。これらは、NVLと同様の機能を提供します。
- NVLの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

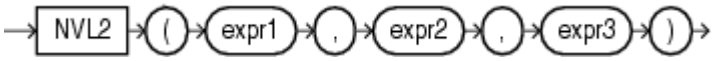
例

次の例では、従業員の名前と歩合のリストを戻し、従業員が歩合を受け取らない場合には「Not Applicable」を表示します。

```
SELECT last_name, NVL(TO_CHAR(commission_pct), 'Not Applicable') commission
  FROM employees
  WHERE last_name LIKE 'B%'
  ORDER BY last_name;

LAST_NAME                 COMMISSION
------------------------- ----------------------------------------
Baer                      Not Applicable
Baida                     Not Applicable
Banda                     .1
Bates                     .15
Bell                      Not Applicable
Bernstein                 .25
Bissot                    Not Applicable
Bloom                     .2
Bull                      Not Applicable
```

NVL2

構文



目的

NVL2を使用すると、指定された式がNULLかどうかに基づく問合せによって戻される値を判断できます。expr1がNULLでない場合、NVL2はexpr2を戻します。expr1がNULLの場合、NVL2はexpr3を戻します。

引数expr1は、任意のデータ型を持つことができます。引数expr2およびexpr3は、LONG以外の任意のデータ型を持つことができます。

expr2とexpr3のデータ型が異なる場合、一方のデータ型が他方のデータ型に暗黙的に変換されます。暗黙的に変換できない場合、データベースはエラーを戻します。expr2が文字または数値データの場合、暗黙的な変換は次のように実装されます。

- expr2が文字データの場合、Oracle Databaseは、expr3がNULLの定数ではないかぎり、値を戻す前にexpr3をexpr2のデータ型に変換します。その場合、データ型の変換は行われず、expr2の文字セットでVARCHAR2が戻されます。
- expr2が数値データである場合、Oracle Databaseは数値の優先順位が最も高い引数を判断し、その引数のデータ型に他方の引数を暗黙的に変換して、そのデータ型を戻します。

関連項目:

- 暗黙的な変換の詳細は、表2-8を参照してください。数値の優先順位の詳細は、「数値の優先順位の詳細」を参照してください。
- NVL2の戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

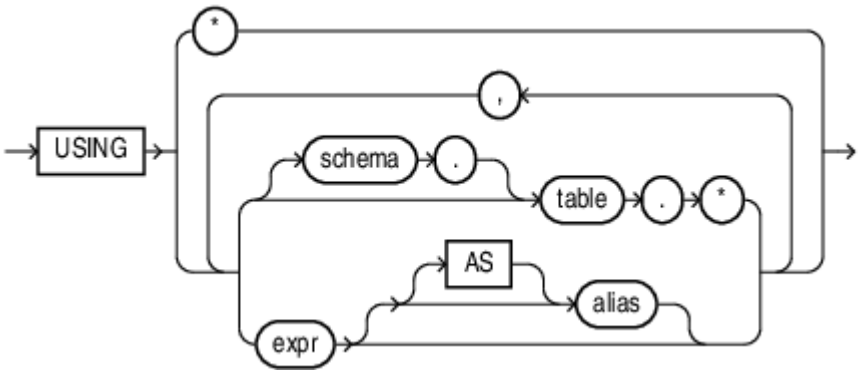
次の例では、employeesのcommission_pct列がNULLかどうかによって、従業員の収入が給与と歩合か、または給与のみかを示します。

```
SELECT last_name, salary,
       NVL2(commission_pct, salary + (salary * commission_pct), salary) income
  FROM employees
  WHERE last_name like 'B%'
  ORDER BY last_name;
LAST_NAME                      SALARY     INCOME
------------------------- ---------- ----------
Baer                            10000      10000
Baida                            2900       2900
Banda                            6200       6820
Bates                            7300       8395
Bell                             4000       4000
Bernstein                        9500      11875
Bissot                           3300       3300
Bloom                           10000      12000
Bull                             4100       4100
```

ORA_DM_PARTITION_NAME

構文

mining_attribute_clause::=



目的

ORA_DM_PARTITION_NAMEは、他の既存のファンクションと連携する単一行ファンクションです。このファンクションは、入力行に関連付けられているパーティションの名前を戻します。ORA_DM_PARTITION_NAMEをパーティション化されていないモデルで使用した場合の結果はNULLです。

ORA_DM_PARTITION_NAMEファンクションの構文では、パーティション化されたモデルをスコアリングするときに、オプションのGROUPINGヒントを使用できます。「GROUPINGヒント」を参照してください。

mining_attribute_clause

mining_attribute_clauseは、スコアの予測子として使用する列の属性を特定します。このファンクションが分析構文で起動された場合は、これらの予測子が一時モデルの構築にも使用されます。mining_attribute_clauseは、PREDICTIONファンクションと同様に動作します。「mining_attribute_clause」を参照してください。

関連項目:

- スコアリングの詳細は、Oracle Data Miningユーザーズ・ガイドを参照してください。
- クラスタリングの詳細は、『Oracle Data Mining概要』を参照してください。

ノート:

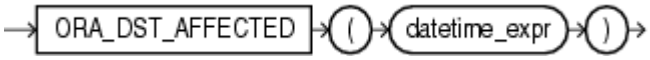
次に示す例は、Data Mining のサンプル・プログラムからの抜粋です。サンプル・プログラムの詳細は、Oracle Data Mining ユーザーズ・ガイドの「付録 A」を参照してください。

例

```
SELECT prediction(mymodel using *) pred, ora_dm_partition_name(mymodel USING *) pname
FROM customers;
```

ORA_DST_AFFECTED

構文



目的

ORA_DST_AFFECTEDは、データベースのタイムゾーン・データファイルを変更するときに役立ちます。このファンクションは、引数として、TIMESTAMP WITH TIME ZONE値、またはTIMESTAMP WITH TIME ZONE値を含むVARRAYオブジェクトに解決される日時式を取ります。このファンクションは、新しいタイムゾーン・データを使用したときに、日時値が、存在しない時刻または重複時刻のエラーの影響を受けたり、いずれかのエラーになる場合は1を戻します。それ以外の場合は0を戻します。

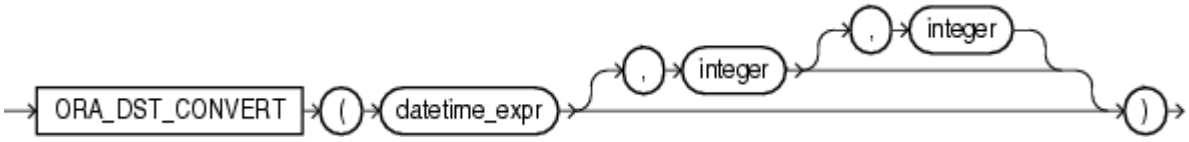
このファンクションを発行できるのは、データベースのタイムゾーン・データファイルを変更してタイムゾーン・データ付きのタイムスタンプをアップグレードする場合のみです。さらに、DBMS_DST.BEGIN_PREPAREプロシージャを実行してからDBMS_DST.END_PREPAREプロシージャを実行するまでの間か、またはDBMS_DST.BEGIN_UPGRADEプロシージャを実行してからDBMS_DST.END_UPGRADEプロシージャを実行するまでの間にのみ発行できます。

関連項目:

タイムゾーン・データファイルの詳細およびOracle Databaseでの夏時間の処理方法は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。DBMS_DSTパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

ORA_DST_CONVERT

構文



目的

ORA_DST_CONVERTは、データベースのタイムゾーン・データファイルを変更するときに役立ちます。このファンクションでは、指定した日時式のエラー処理方法を指定できます。

- datetime_exprには、TIMESTAMP WITH TIME ZONE値、またはTIMESTAMP WITH TIME ZONE値を含むVARRAYオブジェクトに解決される日時式を指定します。
- オプションの2番目の引数には、重複時刻エラーの処理方法を指定します。元の日時値を戻し、エラーを発生させないようにする場合は、0(偽)を指定します。これはデフォルトです。重複時刻エラーを戻すようにする場合は、1(真)を指定します。
- オプションの3番目の引数には、存在しない時刻エラーの処理方法を指定します。元の日時値を戻し、エラーを発生させないようにする場合は、0(偽)を指定します。これはデフォルトです。存在しない時刻エラーを戻すようにする場合は、1(真)を指定します。

エラーが発生しない場合、このファンクションは、datetime_exprと同じデータ型の値(TIMESTAMP WITH TIME ZONE値、またはTIMESTAMP WITH TIME ZONE値を含むVARRAYオブジェクト)を戻します。新しいタイムゾーン・ファイルで解析された場合、戻される日時値は、古いタイムゾーン・ファイルで解析されたdatetime_exprに対応します。

このファンクションを発行できるのは、データベースのタイムゾーン・データファイルを変更してタイムゾーン・データ付きのタイムスタンプをアップグレードする場合のみです。さらに、DBMS_DST.BEGIN_UPGRADEプロシージャを実行してからDBMS_DST.END_UPGRADEプロシージャを実行するまでの間にのみ発行できます。

関連項目:

タイムゾーン・データファイルの詳細およびOracle Databaseでの夏時間の処理方法は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。DBMS_DSTパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

ORA_DST_ERROR

構文

目的

ORA_DST_ERRORは、データベースのタイムゾーン・データファイルを変更するときに役立ちます。このファンクションは、引数として、TIMESTAMP WITH TIME ZONE値、またはTIMESTAMP WITH TIME ZONE値を含むVARRAYオブジェクトに解決される日時式を取り、新しいタイムゾーン・データを使用したときに日時値がエラーになるかどうかを示します。戻り値は次のとおりです。

- 0: 新しいタイムゾーン・データを使用しても、日時値はエラーになりません。
- 1878: 日時値は存在しない時刻エラーになります。
- 1883: 日時値は重複時刻エラーになります。

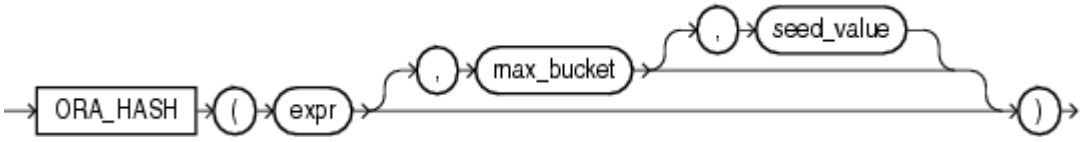
このファンクションを発行できるのは、データベースのタイムゾーン・データファイルを変更してタイムゾーン・データ付きのタイムスタンプをアップグレードする場合のみです。さらに、DBMS_DST.BEGIN_PREPAREプロシージャを実行してからDBMS_DST.END_PREPAREプロシージャを実行するまでの間か、またはDBMS_DST.BEGIN_UPGRADEプロシージャを実行してからDBMS_DST.END_UPGRADEプロシージャを実行するまでの間にのみ発行できます。

関連項目:

タイムゾーン・データファイルの詳細およびOracle Databaseでの夏時間の処理方法は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。DBMS_DSTパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

ORA_HASH

構文



目的

ORA_HASHファンクションは、指定された式のハッシュ値を計算します。このファンクションは、データのサブセットの分析や、ランダムな標本の生成などの操作に有効です。

- expr引数には、Oracle Databaseでハッシュ値を計算するデータを指定します。exprに指定できるデータの長さに制限はありません。このデータは通常、列名です。exprは、LONG型またはLOB型にすることはできません。また、ネストした表型でない場合は、ユーザー定義オブジェクト型にすることはできません。ネストした表型のハッシュ値は、コレクション内の要素の順序に依存しません。その他のデータ型はすべて、exprでサポートされています。
- オプションのmax_bucket引数には、ハッシュ・ファンクションから戻される最大バケット値を指定します。0(ゼロ)から4294967295の任意の値を指定できます。デフォルト値は4294967295です。
- オプションのseed_value引数を指定すると、同じデータ・セットに対して様々な結果を生成できます。Oracleは、ハッシュ・ファンクションをexprとseed_valueの組合せに適用します。0(ゼロ)から4294967295の任意の値を指定できます。デフォルト値は0です。

戻り値はNUMBERです。

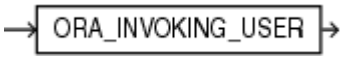
例

次の例では、sh.sales表内の顧客IDと製品IDの各組合せに対してハッシュ値を作成し、そのハッシュ値を最大100個のバケットに分割して、最初のバケット(バケット0(ゼロ))でamount_sold値の合計を戻します。3つ目の引数(5)には、ハッシュ・ファンクションのシード値を指定しています。このシード値を変更すると、同じ問合せで異なるハッシュ結果を得ることができます。

```
SELECT SUM(amount_sold)
  FROM sales
  WHERE ORA_HASH(CONCAT(cust_id, prod_id), 99, 5) = 0;
SUM(AMOUNT_SOLD)
----------------
       989431.14
```

ORA_INVOKING_USER

構文



目的

ORA_INVOKING_USERは、現在の文またはビューを起動したデータベース・ユーザーの名前を返します。このファンクションは、文で参照される中間ビューのBEQUEATHプロパティを考慮に入れます。このファンクションが定義者権限コンテキスト内で起動されたときには、その定義者権限オブジェクトの所有者名が返されます。このファンクションを起動したユーザーがReal Application Securityユーザーの場合は、ユーザーXS$NULLが返されます。

このファンクションは、VARCHAR2値を返します。

関連項目:

- CREATE VIEW文のBEQUEATH句
- ユーザーXS$NULLの詳細は、『Oracle Database 2日でセキュリティ・ガイド』を参照してください。
- ORA_INVOKING_USERの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

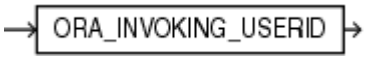
例

次の例では、文を起動したデータベース・ユーザーの名前が返されます。

```
SELECT ORA_INVOKING_USER FROM DUAL;
```

ORA_INVOKING_USERID

構文



目的

ORA_INVOKING_USERIDは、現在の文またはビューを起動したデータベース・ユーザーのIDを返します。このファンクションは、文で参照される中間ビューのBEQUEATHプロパティを考慮に入れます。

このファンクションは、NUMBER値を返します。

関連項目:

- 現在の文またはビューを起動したデータベース・ユーザーをOracle Databaseが判断する方法を学習するには、「ORA_INVOKING_USER」を参照してください。
- CREATE VIEW文のBEQUEATH句

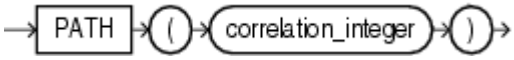
例

次の例では、文を起動したデータベース・ユーザーのIDが返されます。

```
SELECT ORA_INVOKING_USERID FROM DUAL;
```

PATH

構文



目的

PATHは、UNDER_PATHおよびEQUALS_PATH条件でのみ使用される補助ファンクションです。PATHは、親条件で指定されたリソースへの相対パスを戻します。

correlation_integerはNUMBER型の任意の整数で、この補助ファンクションをその一次条件と関連付けるために使用します。1未満の値は1として扱われます。

関連項目:

- 「EQUALS_PATH条件」および「UNDER_PATH条件」を参照してください。
- PATHの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

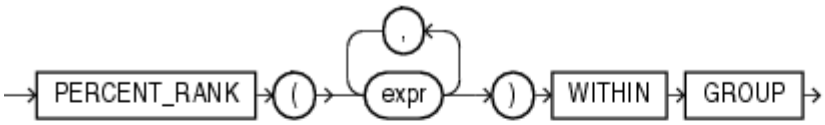
例

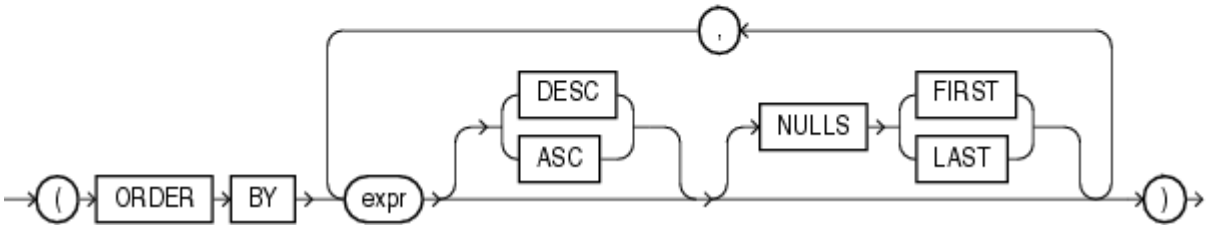
EQUALS_PATH条件およびUNDER_PATH条件でこの補助ファンクションを使用する例は、関連ファンクション「DEPTH」を参照してください。

PERCENT_RANK

集計の構文

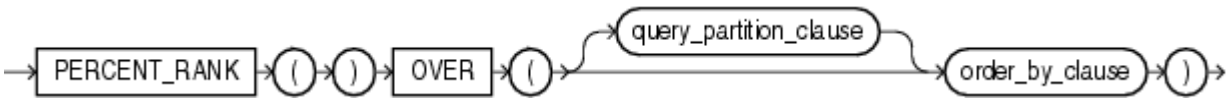
percent_rank_aggregate::=





分析構文

percent_rank_analytic::=



関連項目:

構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

PERCENT_RANKは、CUME_DIST(累積分布)ファンクションと似ています。PERCENT_RANKが戻す値の範囲は、0から1(0および1を含む)です。すべての集合の最初の行のPERCENT_RANKは0(ゼロ)になります。戻り値はNUMBERです。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

- 集計ファンクションとしてのPERCENT_RANKは、ファンクションの引数および対応するソート指定によって識別される不確定な行rを計算し、行rのランクから1を引いて、集計グループ内の行の数で割ります。不確定な行rをOracle Databaseが集計する行のグループに挿入するように計算します。

  このファンクションの引数は、各集計グループ内の1つの不確定行を識別します。このため、すべての引数は、集計グループ内で定数式と評価される必要があります。定数引数式および集計のORDER BY句の式の位置は、一致します。このため、引数の数は同じであり、その型は互換性がある必要があります。

- 分析ファンクションとしてのPERCENT_RANKは、行rに対して、rのランクから1を引いた数を、評価される行数(問合せ結果セット全体またはパーティション)より1少ない数で割ります。

関連項目:

ORDER BY句の文字値を比較するためにPERCENT_RANKで使用する照合を定義する照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

集計の例

次の例では、サンプル表hr.employeesから、給与が$15,500であり、歩合が5%である不確定な従業員のパーセント・ランクを計算します。

```
SELECT PERCENT_RANK(15000, .05) WITHIN GROUP
       (ORDER BY salary, commission_pct) "Percent-Rank"
  FROM employees;
Percent-Rank
------------
  .971962617
```

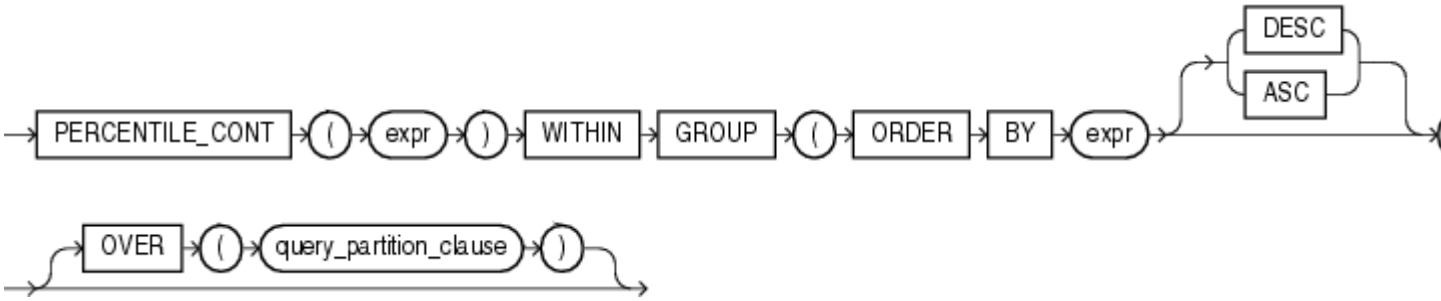
分析の例

次の例では、従業員ごとに、部門内での給与のパーセント・ランクを計算します。

```
SELECT department_id, last_name, salary, PERCENT_RANK()
       OVER (PARTITION BY department_id ORDER BY salary DESC) AS pr
  FROM employees
  ORDER BY pr, salary, last_name;
DEPARTMENT_ID LAST_NAME                     SALARY         PR
------------- ------------------------- ---------- ----------
           10 Whalen                          4400          0
           40 Mavris                          6500          0
              Grant                           7000          0
. . .
           80 Vishney                        10500 .181818182
           80 Zlotkey                        10500 .181818182
           30 Khoo                            3100         .2
. . .
           50 Markle                          2200 .954545455
           50 Philtanker                      2200 .954545455
           50 Olson                           2100          1
. . .
```

PERCENTILE_CONT

構文



関連項目:

OVER句の構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

PERCENTILE_CONTは、連続分散モデルを想定する逆分散関数です。このファンクションは、パーセンタイル値およびソート指定を使用し、そのソート指定に従ってそのパーセンタイル値に該当する補間された値を戻します。計算では、NULLは無視されます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を返します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

最初のexprは、パーセンタイル値であるため、0から1の数値で評価します。このexprは、各集計グループ内の定数である必要があります。ORDER BY句には、Oracleが補間を実行できる型である数値または日時値の単一式を指定します。

PERCENTILE_CONTの結果は、順序付けされた後の値間の直線補間によって計算されます。集計グループで、パーセンタイル値(P)および行数(N)を使用すると、ソート指定に従って行を順序付けた後の行数を計算できます。この行数(RN)は、計算式RN = (1+ (P*(N-1))に従って計算されます。集計ファンクションの最終結果は、行番号がCRN = CEILING(RN)およびFRN = FLOOR(RN)の行の値間の直線補間によって計算されます。

最終結果は次のとおりです。

```
  If (CRN = FRN = RN) then the result is
    (value of expression from row at RN)
  Otherwise the result is
    (CRN - RN) * (value of expression for row at FRN) +
    (RN - FRN) * (value of expression for row at CRN)
```

PERCENTILE_CONTファンクションは、分析ファンクションとしても使用できます。その場合、OVER句には、query_partitioning_clauseのみを指定できます。各行に対して、各パーティション内の一連の値から、指定されたパーセンタイルに該当する値を戻します。

MEDIANファンクションは、パーセンタイル値がデフォルトで0.5に指定される特別なPERCENTILE_CONTです。詳細は、「MEDIAN」を参照してください。

ノート:

PERCENTILE_CONT ファンクションで大量のデータを処理する前に、次のいずれかの方法を使用して、正確な結果よりも迅速におおよその結果を取得することを検討してください。

- PERCENTILE_CONT ファンクションを使用する前に、APPROX_FOR_PERCENTILE 初期化パラメータを PERCENTILE_CONT または ALL に設定します。このパラメータの詳細は、『Oracle Database リファレンス』を参照してください。
- PERCENTILE_CONT ファンクションではなく、APPROX_PERCENTILE ファンクションを使用します。「APPROX_PERCENTILE」を参照してください。

集計の例

次の例では、各部門の給与の中央値を計算します。

```
SELECT department_id,
       PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY salary DESC) "Median cont",
       PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY salary DESC) "Median disc"
  FROM employees
  GROUP BY department_id
  ORDER BY department_id;
DEPARTMENT_ID Median cont Median disc
------------- ----------- -----------
           10        4400        4400
           20        9500       13000
           30        2850        2900
           40        6500        6500
           50        3100        3100
           60        4800        4800
           70       10000       10000
           80        8900        9000
           90       17000       17000
          100        8000        8200
          110       10154       12008
                     7000        7000
```

PERCENTILE_CONTおよびPERCENTILE_DISCは、異なる結果を戻す場合があります。PERCENTILE_CONTは、直線補間後の計算結果を戻します。PERCENTILE_DISCは、集計された一連の値から値のみを戻します。この例に示すように、パーセンタイル値が0.5の場合、PERCENTILE_CONTは、偶数の要素を持つグループの中間の2つの値の平均を戻します。それに対して、PERCENTILE_DISCは、中間の2つの値の最初の値を戻します。奇数の要素を持つ集計グループの場合は、どちらの関数も中間要素の値を戻します。

分析の例

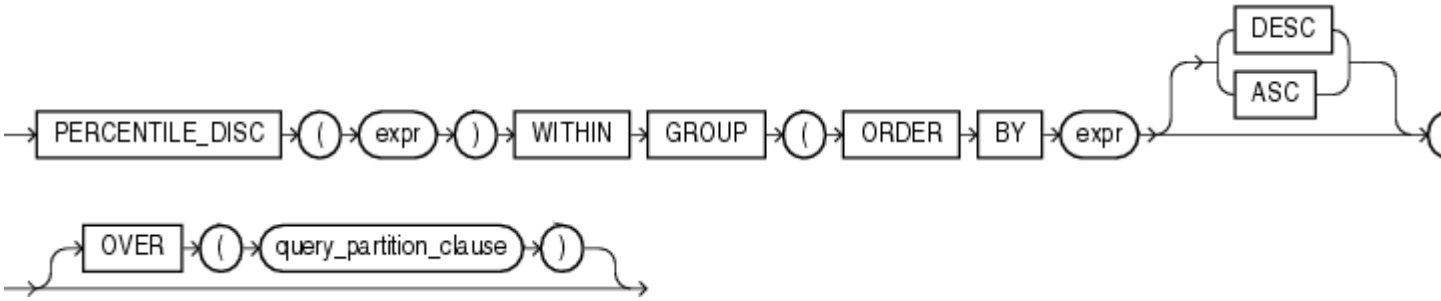
次の例では、0.5のパーセンタイル(Percent_Rank)に対応する部門60の中央値は4800です。部門30の給与にパーセンタイル0.5がないため、2900(パーセンタイル0.4)から2800(パーセンタイル0.6)の範囲で2850に評価される中央値が補間される必要があります。

```
SELECT last_name, salary, department_id,
       PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY salary DESC)
         OVER (PARTITION BY department_id) "Percentile_Cont",
       PERCENT_RANK()
        OVER (PARTITION BY department_id ORDER BY salary DESC) "Percent_Rank"
  FROM employees
  WHERE department_id IN (30, 60)
```

```
  ORDER BY last_name, salary, department_id;
LAST_NAME                    SALARY DEPARTMENT_ID Percentile_Cont Percent_Rank
------------------------- ---------- ------------- --------------- ------------
Austin                         4800            60            4800           .5
Baida                          2900            30            2850           .4
Colmenares                     2500            30            2850            1
Ernst                          6000            60            4800          .25
Himuro                         2600            30            2850           .8
Hunold                         9000            60            4800            0
Khoo                           3100            30            2850           .2
Lorentz                        4200            60            4800            1
Pataballa                      4800            60            4800           .5
Raphaely                      11000            30            2850            0
Tobias                         2800            30            2850           .6
```

PERCENTILE_DISC

構文



関連項目:

OVER句の構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

PERCENTILE_DISCは、不連続分散モデルを想定する逆分散関数です。このファンクションでは、パーセンタイル値およびソート指定を指定し、そのセットから要素を戻します。計算では、NULLは無視されます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を返します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

最初のexprは、パーセンタイル値であるため、0から1の数値で評価します。この式は、各集計グループ内の定数である必要があります。ORDER BY句には、ソート可能な型の単一式を指定します。

指定されたパーセンタイル値Pに対して、PERCENTILE_DISCは、ORDER BY句の式の値をソートし、P以上である(同じソート指定に従う)最小CUME_DIST値を持つ値を戻します。

ノート:

PERCENTILE_DISC ファンクションで大量のデータを処理する前に、次のいずれかの方法を使用して、正確な結果よりも迅速におおよその結果を取得することを検討してください。

- PERCENTILE_DISC ファンクションを使用する前に、APPROX_FOR_PERCENTILE 初期化パラメータを PERCENTILE_DISC または ALL に設定します。このパラメータの詳細は、『Oracle Database リファレンス』を参照してください。
- PERCENTILE_DISC ファンクションではなく、APPROX_PERCENTILE ファンクションを使用します。「APPROX_PERCENTILE」を参照してください。

集計の例

「PERCENTILE_CONT」の集計の例を参照してください。

分析の例

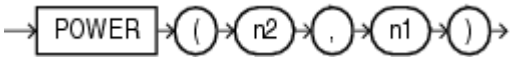
次の例では、サンプル表hr.employeesの各従業員の給与の中央値不連続パーセンタイルを計算します。

```
SELECT last_name, salary, department_id,
       PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY salary DESC)
         OVER (PARTITION BY department_id) "Percentile_Disc",
       CUME_DIST() OVER (PARTITION BY department_id
         ORDER BY salary DESC) "Cume_Dist"
  FROM employees
  WHERE department_id in (30, 60)
  ORDER BY last_name, salary, department_id;
LAST_NAME                 SALARY DEPARTMENT_ID Percentile_Disc  Cume_Dist
------------------------- ---------- ------------- --------------- ----------
Austin                          4800            60            4800         .8
Baida                           2900            30            2900         .5
Colmenares                      2500            30            2900          1
Ernst                           6000            60            4800         .4
Himuro                          2600            30            2900 .833333333
Hunold                          9000            60            4800         .2
Khoo                            3100            30            2900 .333333333
Lorentz                         4200            60            4800          1
Pataballa                       4800            60            4800         .8
Raphaely                       11000            30            2900 .166666667
Tobias                          2800            30            2900 .666666667
```

部門30の中央値の値は2900です。この値の対応するパーセンタイル(Cume_Dist)は、0.5以上の最小値です。部門60の中央値の値は4800です。この値の対応するパーセンタイルは、0.5以上の最小値です。

POWER

構文



目的

POWERは、n2をn1乗した値を戻します。底n2および指数n1は任意の数です。ただし、n2が負の場合、n1は整数である必要があります。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。いずれかの引数がBINARY_FLOATまたはBINARY_DOUBLEの場合、このファンクションはBINARY_DOUBLEを戻します。それ以外の場合、NUMBERを戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

例

次の例では、3の2乗を戻します。

```
SELECT POWER(3,2) "Raised"
  FROM DUAL;
    Raised
----------
         9
```

POWERMULTISET

構文

目的

POWERMULTISETは、入力としてネストした表を取り、入力されたネストした表のすべての空でないサブセットを含む、ネストした表のネストした表(サブ多重集合という)を戻します。

- exprには、ネストした表に評価される任意の式を指定できます。
- exprがNULLである場合、Oracle DatabaseはNULLを戻します。
- exprが空のネストした表である場合、Oracleはエラーを戻します。
- ネストした表の要素の型は、比較可能な型である必要があります。スカラー型以外の型の比較の可能性の詳細は、「比較条件」を参照してください。

ノート:

このファンクションは、PL/SQL ではサポートされていません。

例

まず、cust_address_tab_typeデータ型のネストした表であるデータ型を作成します。

```
CREATE TYPE cust_address_tab_tab_typ
  AS TABLE OF cust_address_tab_typ;
/
```

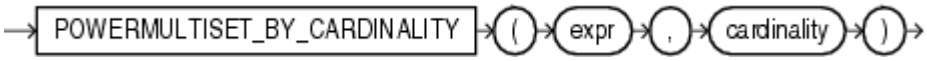
次に、POWERMULTISETファンクションを使用して、customers_demo表から、ネストした表の列cust_address_ntabを選択します。

```
SELECT CAST(POWERMULTISET(cust_address_ntab) AS cust_address_tab_tab_typ)
  FROM customers_demo;
CAST(POWERMULTISET(CUST_ADDRESS_NTAB) AS CUST_ADDRESS_TAB_TAB_TYP)
  (STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE, COUNTRY_ID)
--------------------------------------------------------------------
CUST_ADDRESS_TAB_TAB_TYP(CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP
  ('514 W Superior St', '46901', 'Kokomo', 'IN', 'US')))
CUST_ADDRESS_TAB_TAB_TYP(CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP
  ('2515 Bloyd Ave', '46218', 'Indianapolis', 'IN', 'US')))
CUST_ADDRESS_TAB_TAB_TYP(CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP
  ('8768 N State Rd 37', '47404', 'Bloomington', 'IN', 'US')))
CUST_ADDRESS_TAB_TAB_TYP(CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP
  ('6445 Bay Harbor Ln', '46254', 'Indianapolis', 'IN', 'US')))
. . .
```

前述の例では、customers_demo表、およびデータを含むネストした表の列が必要です。この表およびネストした表の列を作成する方法は、「MULTISET演算子」を参照してください。

POWERMULTISET_BY_CARDINALITY

構文



目的

POWERMULTISET_BY_CARDINALITYは、入力としてネストした表およびカーディナリティを取り、指定されたカーディナリティを持つネストした表のすべての空でないサブセットを含む、ネストした表のネストした表(サブ多重集合という)を戻します。

- exprには、ネストした表に評価される任意の式を指定できます。
- cardinalityには、任意の正の整数を指定できます。
- exprがNULLである場合、Oracle DatabaseはNULLを戻します。
- exprが空のネストした表である場合、Oracleはエラーを戻します。
- ネストした表の要素の型は、比較可能な型である必要があります。スカラー型以外の型の比較の可能性の詳細は、「比較条件」を参照してください。

ノート:

このファンクションは、PL/SQL ではサポートされていません。

例

まず、cust_address_tab_typeデータ型のネストした表であるデータ型を作成します。

```
CREATE TYPE cust_address_tab_tab_typ
  AS TABLE OF cust_address_tab_typ;
/
```

次に、ネストした表のすべての行内の要素を複製し、ネストした表の行のカーディナリティを2に増やします。

```
UPDATE customers_demo
  SET cust_address_ntab = cust_address_ntab MULTISET UNION cust_address_ntab;
```

次に、POWERMULTISET_BY_CARDINALITYファンクションを使用して、customers_demo表から、ネストした表の列cust_address_ntabを選択します。

```
SELECT CAST(POWERMULTISET_BY_CARDINALITY(cust_address_ntab, 2)
         AS cust_address_tab_tab_typ)
  FROM customers_demo;

CAST(POWERMULTISET_BY_CARDINALITY(CUST_ADDRESS_NTAB,2) AS CUST_ADDRESS_TAB_TAB_TYP)
  (STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE, COUNTRY_ID)
--------------------------------------------------------------------------------
---
CUST_ADDRESS_TAB_TAB_TYP(CUST_ADDRESS_TAB_TYP
  (CUST_ADDRESS_TYP('514 W Superior St', '46901', 'Kokomo', 'IN', 'US'),
   CUST_ADDRESS_TYP('514 W Superior St', '46901', 'Kokomo', 'IN', 'US')))
CUST_ADDRESS_TAB_TAB_TYP(CUST_ADDRESS_TAB_TYP
  (CUST_ADDRESS_TYP('2515 Bloyd Ave', '46218', 'Indianapolis', 'IN', 'US'),
   CUST_ADDRESS_TYP('2515 Bloyd Ave', '46218', 'Indianapolis', 'IN', 'US')))
CUST_ADDRESS_TAB_TAB_TYP(CUST_ADDRESS_TAB_TYP
```

```
  (CUST_ADDRESS_TYP('8768 N State Rd 37', '47404', 'Bloomington', 'IN', 'US'),
   CUST_ADDRESS_TYP('8768 N State Rd 37', '47404', 'Bloomington', 'IN', 'US')))
. . .
```

前述の例では、customers_demo表、およびデータを含むネストした表の列が必要です。この表およびネストした表の列を作成する方法は、「MULTISET演算子」を参照してください。

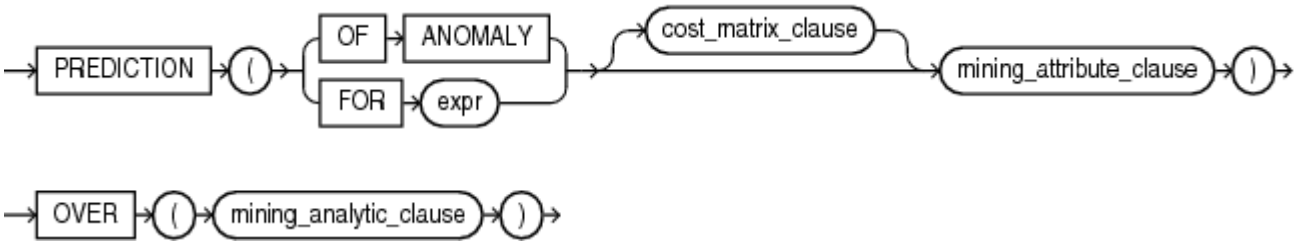
PREDICTION

構文

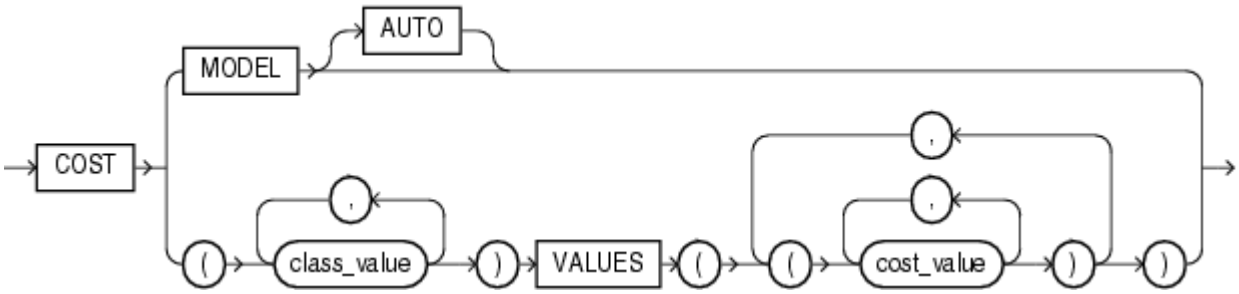
prediction::=

分析構文

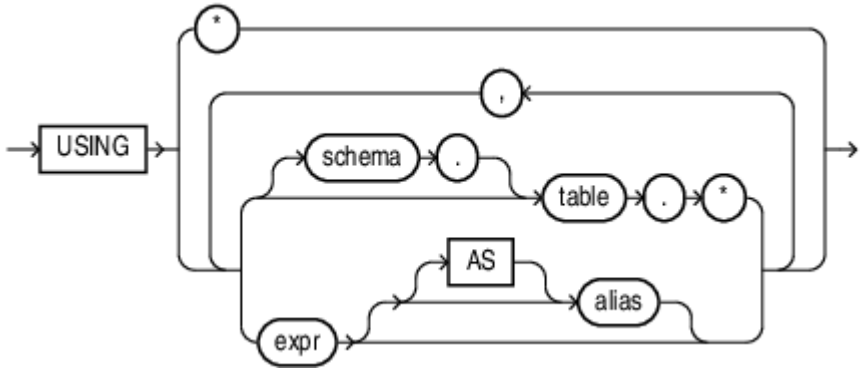
prediction_analytic::=



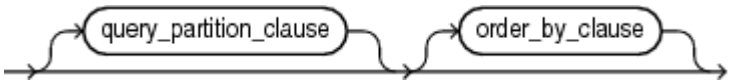
cost_matrix_clause::=



mining_attribute_clause::=



mining_analytic_clause::=



関連項目:

mining_analytic_clauseの構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

PREDICTIONは、選択内に含まれる各行の予測を返します。戻される予測のデータ型は、ファンクションが回帰、分類、異常検出のいずれを実行するかによって異なります。

- 回帰: 各行の予測されるターゲット値を戻します。戻り値のデータ型は、ターゲットのデータ型になります。
- 分類: 各行の最も確率の高いターゲット・クラス(または、コストが指定された場合は最もコストの低いターゲット・クラス)を戻します。戻り値のデータ型は、ターゲットのデータ型になります。
- 異常検出: 各行について1または0を戻します。通常の行は1に分類されます。残りのデータと著しく異なる行は0に分類されます。

cost_matrix_clause

コストは、最も悪影響を及ぼす誤分類を最小化するためのバイアス係数です。分類または異常検出には、cost_matrix_clauseを指定できます。コストは、回帰には適していません。cost_matrix_clauseは、「PREDICTION_COST」で説明されているように動作します。

構文の選択

PREDICTIONは、2つの方法のどちらかでデータにスコアを付けます。1つの方法では、データにマイニング・モデル・オブジェクトを適用します。もう1つの方法では、1つ以上の一時マイニング・モデルを作成して適用する分析句を実行して動的にデータにスコアを付けます。構文または分析構文を選択します。

- 構文: 事前に定義されたモデルでデータにスコアを付ける場合は、この構文を使用します。分類、回帰または異常検出を実行するモデルの名前を指定します。
- 分析構文: 事前定義されたモデルなしで、データにスコアを付ける場合は、分析構文を使用します。分析構文は、mining_analytic_clauseを使用します。これは、複数のモデル構築のためにデータをパーティション化する必要がある場合に指定します。mining_analytic_clauseは、query_partition_clauseとorder_by_clauseをサポートします。(「analytic_clause::=」を参照。)
    - 回帰の場合は、FOR exprを指定します。exprは、数値データ型のターゲット列を特定する式です。
    - 分類の場合は、FOR exprを指定します。exprは、文字データ型のターゲット列を特定する式です。
    - 異常検出の場合は、キーワードOF ANOMALYを指定します。

PREDICTIONファンクションの構文では、パーティション化されたモデルをスコアリングするときに、オプションのGROUPINGヒントを使用できます。「GROUPINGヒント」を参照してください。

mining_attribute_clause

mining_attribute_clauseは、スコアの予測子として使用する列の属性を特定します。

- USING *を指定すると、入力行に存在するすべての関連属性が使用されます。
- このファンクションを解析構文で起動すると、一時モデルの作成とスコアリングの両方にmining_attribute_clauseが使用されます。
- 事前に定義されたモデルを指定してこのファンクションを起動するときには、そのモデルの作成に使用したすべて(または一部)の属性をmining_attribute_clauseに含める必要があります。次の条件が適用されます。
    - mining_attribute_clauseに含まれている属性がモデルの作成に使用した属性と同じ名前でデータ

型が異なる場合、そのデータ型はモデルが期待するデータ型に変換されます。

- モデルの作成に使用した属性よりも多くの属性をスコアリングに指定すると、余分な属性は無視されます(エラーや警告は表示されません)。
- モデルの作成に使用した属性よりも少ない属性を指定すると、最善の方法でスコアリングが実行されます。

関連項目:

- スコアリングの詳細は、Oracle Data Miningユーザーズ・ガイドを参照してください。
- 予測データ・マイニングの詳細は、『Oracle Data Mining概要』を参照してください。
- PREDICTIONの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

ノート:

次に示す例は、Data Mining のサンプル・プログラムからの抜粋です。サンプル・プログラムの詳細は、Oracle Data Mining ユーザーズ・ガイドの「付録 A」を参照してください。

例

この例では、提携カードを使用することが最も多い顧客の性別と年齢を、モデルdt_sh_clas_sampleで予測します(ターゲット=1)。PREDICTIONファンクションは、コスト・マトリックスを考慮に入れ、婚姻区分、学歴、および世帯人数を予測子として使用します。

```
SELECT cust_gender, COUNT(*) AS cnt, ROUND(AVG(age)) AS avg_age
   FROM mining_data_apply_v
   WHERE PREDICTION(dt_sh_clas_sample COST MODEL
      USING cust_marital_status, education, household_size) = 1
   GROUP BY cust_gender
   ORDER BY cust_gender;

CUST_GENDER         CNT    AVG_AGE
------------ ---------- ----------
F                   170         38
M                   685         42
```

モデルdt_sh_clas_sampleに関連付けられたコスト・マトリックスは、表dt_sh_sample_costsに保存されます。次のコスト・マトリックス表では、1に誤分類すると0に誤分類するよりも8倍コストがかかることを示しています。

```
SQL> select * from dt_sh_sample_cost;

ACTUAL_TARGET_VALUE PREDICTED_TARGET_VALUE         COST
------------------- ----------------------- ------------
                  0                       0   .000000000
                  0                       1  1.000000000
                  1                       0  8.000000000
                  1                       1   .000000000
```

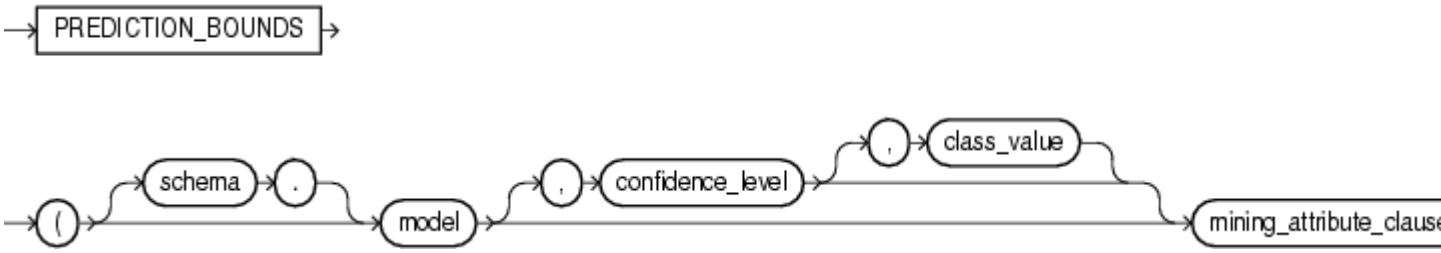
分析の例

この例では、動的回帰を使用して、提携カードを使用する可能性がある顧客の年齢を予測します。この問合せは、予測された年齢が実際とは最も異なる3件の顧客を返します。この問合せには、予測に最も影響する予測子の情報が含まれています。

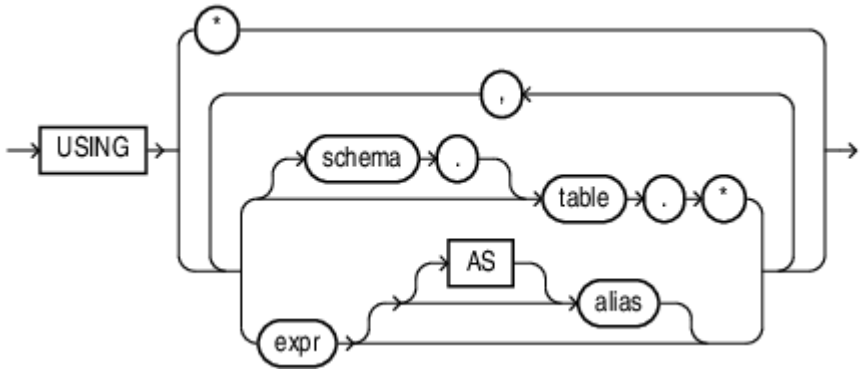
```
SELECT cust_id, age, pred_age, age-pred_age age_diff, pred_det FROM
   (SELECT cust_id, age, pred_age, pred_det,
          RANK() OVER (ORDER BY ABS(age-pred_age) desc) rnk FROM
   (SELECT cust_id, age,
           PREDICTION(FOR age USING *) OVER () pred_age,
           PREDICTION_DETAILS(FOR age ABS USING *) OVER () pred_det
    FROM mining_data_apply_v))
  WHERE rnk <= 3;

CUST_ID  AGE PRED_AGE AGE_DIFF PRED_DET
------- ---- -------- -------- -------- ----------------------------------------
-------------
 100910   80    40.67    39.33 <Details algorithm="Support Vector Machines">
                                <Attribute name="HOME_THEATER_PACKAGE" actualValue="1"
weight=".059"
                                 rank="1"/>
                                <Attribute name="Y_BOX_GAMES" actualValue="0"
weight=".059"
                                 rank="2"/>
                                <Attribute name="AFFINITY_CARD" actualValue="0"
weight=".059"
                                 rank="3"/>
                                <Attribute name="FLAT_PANEL_MONITOR" actualValue="1"
weight=".059"
                                 rank="4"/>
                                <Attribute name="YRS_RESIDENCE" actualValue="4"
weight=".059"
                                 rank="5"/>
                                </Details>

 101285    79   42.18    36.82 <Details algorithm="Support Vector Machines">
                                <Attribute name="HOME_THEATER_PACKAGE" actualValue="1"
weight=".059"
                                 rank="1"/>
                                <Attribute name="HOUSEHOLD_SIZE" actualValue="2"
weight=".059"
                                 rank="2"/>
                                <Attribute name="CUST_MARITAL_STATUS"
actualValue="Mabsent"
                                 weight=".059" rank="3"/>
                                <Attribute name="Y_BOX_GAMES" actualValue="0"
weight=".059"
                                 rank="4"/>
                                <Attribute name="OCCUPATION" actualValue="Prof."
weight=".059"
                                 rank="5"/>
                                </Details>

 100694     77  41.04    35.96 <Details algorithm="Support Vector Machines">
                                <Attribute name="HOME_THEATER_PACKAGE" actualValue="1"
weight=".059"
                                 rank="1"/>
                                <Attribute name="EDUCATION" actualValue="&lt; Bach."
weight=".059"
                                 rank="2"/>
                                <Attribute name="Y_BOX_GAMES" actualValue="0"
weight=".059"
                                 rank="3"/>
                                <Attribute name="CUST_ID" actualValue="100694"
weight=".059"
                                 rank="4"/>
                                <Attribute name="COUNTRY_NAME" actualValue="United
States of
                                 America" weight=".059" rank="5"/>
                                </Details>
```

PREDICTION_BOUNDS

構文



mining_attribute_clause::=



目的

PREDICTION_BOUNDSは、一般化線形モデル(GLM)を適用して、選択内に含まれる各行のクラスまたは値を予測します。この関数は、オブジェクトのVARRAYに含まれる各予測の上限と下限を、それぞれUPPERフィールドとLOWERフィールドに返します。

GLMは、回帰または2項分類を実行できます。

- 回帰の上限と下限は、予測されたターゲット値を参照します。UPPERとLOWERのデータ型は、ターゲットのデータ型になります。
- 2項分類での上限と下限は、予測されたターゲット・クラスまたは指定されたclass_valueの確率を参照します。UPPERとLOWERのデータ型は、BINARY_DOUBLEです。

リッジ回帰を使用してモデルが構築された場合や、構築中に共分散マトリックスの異常が検出された場合、PREDICTION_BOUNDSは上限と下限の両方についてNULLを返します。

confidence_levelは、(0,1)の範囲内の数値です。デフォルト値は0.95です。confidence_levelにNULLを指定すると、デフォルトでconfidence_levelを残したままclass_valueを指定できます。

PREDICTION_BOUNDSファンクションの構文では、パーティション化されたモデルをスコアリングするときに、オプションのGROUPINGヒントを使用できます。「GROUPINGヒント」を参照してください。

mining_attribute_clause

mining_attribute_clauseは、スコアの予測子として使用する列の属性を特定します。この句は、PREDICTIONファンクションの場合と同様に動作します。(分析構文への参照は適用されません。)「mining_attribute_clause::=」を参照してください。

関連項目:

- スコアリングの詳細は、Oracle Data Miningユーザーズ・ガイドを参照してください。
- 一般化線形モデルの詳細は、『Oracle Data Mining概要』を参照してください。

ノート:

次に示す例は、Data Mining のサンプル・プログラムからの抜粋です。サンプル・プログラムの詳細は、Oracle Data Mining ユーザーズ・ガイドの「付録 A」を参照してください。

例

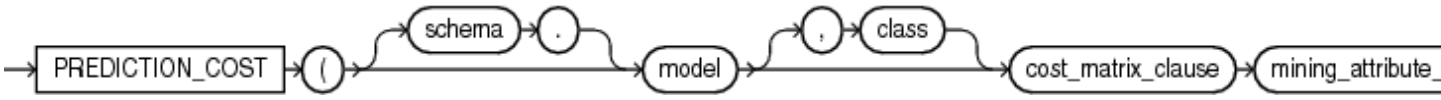
次の例では、98%の確度で年齢が25才から45才と予測される顧客の分布を返します。

```
SELECT count(cust_id) cust_count, cust_marital_status
  FROM (SELECT cust_id, cust_marital_status
    FROM mining_data_apply_v
    WHERE PREDICTION_BOUNDS(glmr_sh_regr_sample,0.98 USING *).LOWER > 24 AND
          PREDICTION_BOUNDS(glmr_sh_regr_sample,0.98 USING *).UPPER < 46)
    GROUP BY cust_marital_status;

    CUST_COUNT CUST_MARITAL_STATUS
-------------- --------------------
            46 NeverM
             7 Mabsent
             5 Separ.
            35 Divorc.
            72 Married
```

PREDICTION_COST

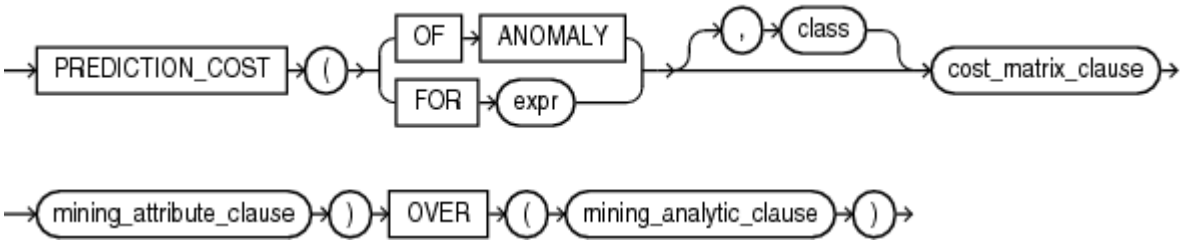
構文

prediction_cost::=

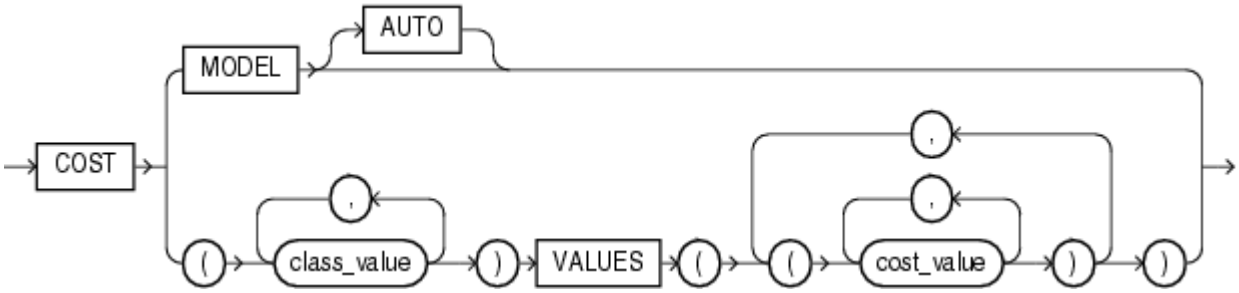


分析構文

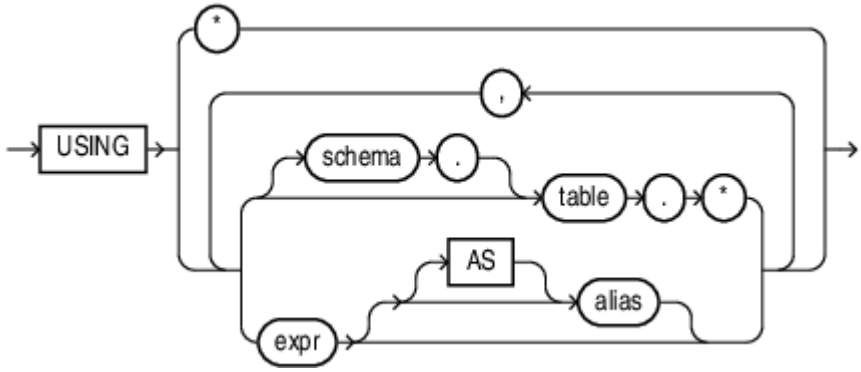
prediction_cost_analytic::=



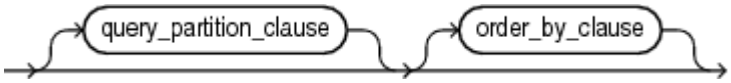
cost_matrix_clause::=



mining_attribute_clause::=



mining_analytic_clause::=



関連項目:

mining_analytic_clauseの構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

PREDICTION_COSTは、選択内に含まれる各行のコストを返します。コストは、最もコストが低いクラスまたは指定されたclassを参照します。コストは、BINARY_DOUBLEとして返されます。

PREDICTION_COSTでは、分類または異常検出を実行できます。分類の場合、返されるコストは予測されたターゲット・クラスを参照します。異常検出の場合、返されるコストは分類1(通常の行)または0(異常な行)を参照します。

PREDICTION_COSTとPREDICTIONファンクションを組み合せて使用すると、予測とその予測のコストを取得できます。

cost_matrix_clause

コストは、最も悪影響を及ぼす誤分類を最小化するためのバイアス係数です。たとえば、偽陽性は、偽陰性よりもコストが高いとみなされることがあります。コストは、モデルに関連付けられるコスト・マトリックスで指定するか、VALUES句にインラインで定義します。すべての分類アルゴリズムは、スコアリングに影響するようにコストを使用できます。

コストを使用してモデルの作成に影響を与えられるアルゴリズムは、デシジョン・ツリーのみです。デシジョン・ツリー・モデルの作成に使用したコスト・マトリックスは、そのモデルに対するデフォルトのスコアリング・マトリックスにもなります。

次のコスト・マトリックス表では、1に誤分類すると、0に誤分類するよりもコストが5倍かかることを示しています。

```
ACTUAL_TARGET_VALUE PREDICTED_TARGET_VALUE       COST
------------------- ----------------------- ----------
                  0                       0          0
                  0                       1          1
                  1                       0          5
                  1                       1          0
```

cost_matrix_clauseの指定内容は、次のとおりです。

- COST MODELは、モデルに関連付けられたコスト・マトリックスを考慮して、スコアリングする必要があることを示します。コスト・マトリックスが存在しない場合、このファンクションはエラーを返します。
- COST MODEL AUTOは、コスト・マトリックスが存在するかどうか不明なことを示します。コスト・マトリックスが存在する場合、このファンクションはコスト・マトリックスを使用して、コストが最小の予測を返します。それ以外の場合、このファンクションは、最も高確率の予測を返します。
- VALUES句では、class_valueにインラインのコスト・マトリックスを指定します。たとえば、1に誤分類すると、0に誤分類するよりも5倍コストがかかることを、次のように指定できます。

  ```
  PREDICTION (nb_model COST (0,1) VALUES ((0, 1),(1, 5)) USING *)
  ```

  スコアリングのコスト・マトリックスがあるモデルを起動するときに、インラインのコスト・マトリックスを指定すると、インラインのコスト・マトリックスが使用されます。

  関連項目:

  コスト考慮型予測の詳細は、Oracle Data Miningユーザーズ・ガイドを参照してください。

構文の選択

PREDICTION_COSTは、2つの方法のどちらかでデータにスコアを付けます。1つの方法では、データにマイニング・モデル・オブジェクトを適用します。もう1つの方法では、1つ以上の一時マイニング・モデルを作成して適用する分析句を実行して動的にデータをマイニングします。構文または分析構文を選択します。

- 構文 — 事前に定義されたモデルでデータにスコアを付ける場合は、最初の構文を使用します。分類または異常検出を実行するモデルの名前を指定します。
- 分析構文 — 事前定義されたモデルなしで、データにスコアを付ける場合は、分析構文を使用します。分析構文は、`mining_analytic_clause`を使用します。これは、複数のモデル構築のためにデータをパーティション化する必要がある場合に指定します。`mining_analytic_clause`は、`query_partition_clause`と`order_by_clause`をサポートします。(「analytic_clause::=」を参照)
    - 分類の場合は、`FOR expr`を指定します。`expr`は、文字データ型のターゲット列を特定する式です。
    - 異常検出の場合は、キーワード`OF ANOMALY`を指定します。

`PREDICTION_COST`ファンクションの構文では、パーティション化されたモデルをスコアリングするときに、オプションの`GROUPING`ヒントを使用できます。「GROUPINGヒント」を参照してください。

mining_attribute_clause

`mining_attribute_clause`は、スコアの予測子として使用する列の属性を特定します。このファンクションが分析構文で起動された場合は、これらの予測子が一時モデルの構築にも使用されます。`mining_attribute_clause`は、`PREDICTION`ファンクションと同様に動作します。(「mining_attribute_clause::=」を参照)

関連項目:

- スコアリングの詳細は、Oracle Data Miningユーザーズ・ガイドを参照してください。
- コストによる分類については、『Oracle Data Mining概要』を参照してください。

ノート:

次に示す例は、Data Mining のサンプル・プログラムからの抜粋です。サンプル・プログラムの詳細は、Oracle Data Mining ユーザーズ・ガイドの「付録 A」を参照してください。

例

この例では、最もコストの低い販売キャンペーン(提携カードの売込み)に反応する可能性があるイタリア在住の顧客を10人予測します。

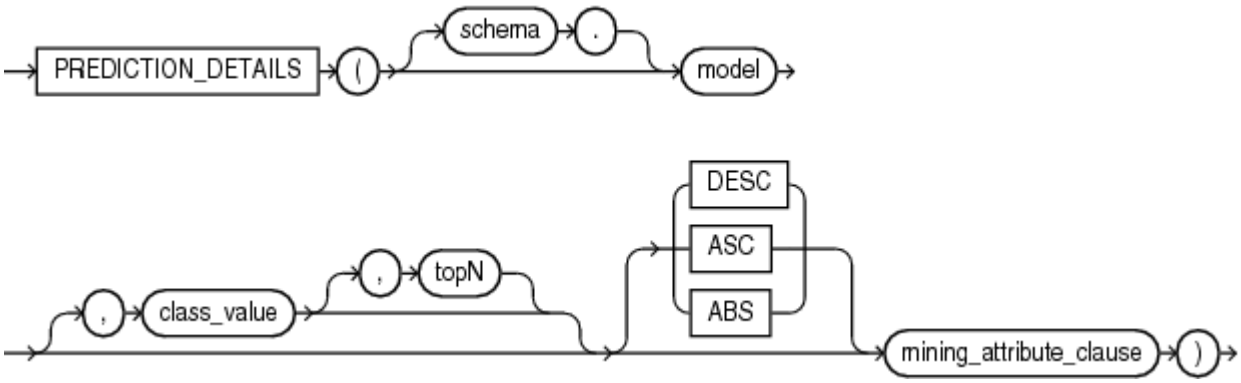
```
SELECT cust_id
FROM (SELECT cust_id,rank()
       OVER (ORDER BY PREDICTION_COST(DT_SH_Clas_sample, 1 COST MODEL USING *)
            ASC, cust_id) rnk
        FROM mining_data_apply_v
        WHERE country_name = 'Italy')
  WHERE rnk <= 10
  ORDER BY rnk;

   CUST_ID
----------
    100081
    100179
    100185
    100324
    100344
    100554
    100662
    100733
```

```
101250
101306
```

PREDICTION_DETAILS

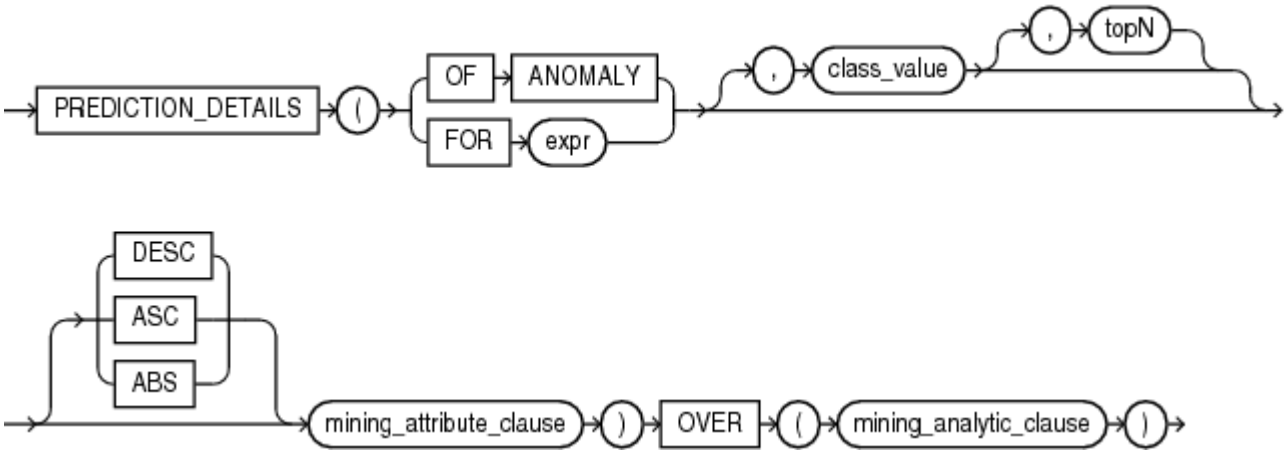
構文

prediction_details::=

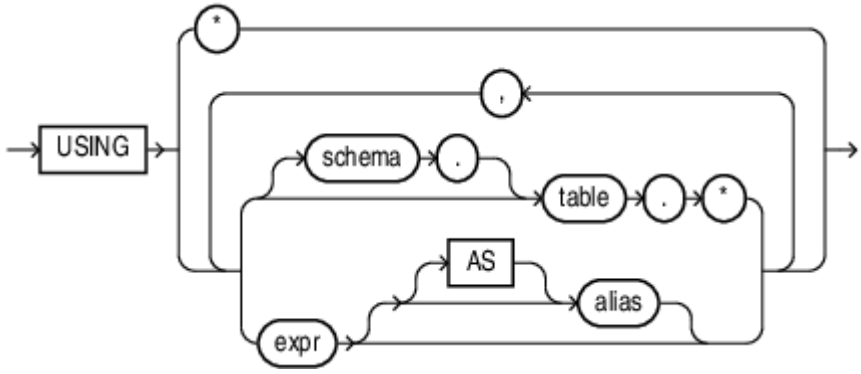


分析構文

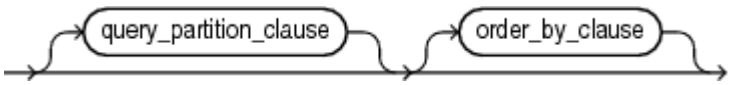
prediction_details_analytic::=



mining_attribute_clause::=



mining_analytic_clause::=



関連項目:

mining_analytic_clauseの構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

PREDICTION_DETAILSは、選択内に含まれる各行の予測の詳細を返します。戻り値は、予測の属性について記述したXML文字列です。

回帰の場合、返される詳細は予測されたターゲット値を参照します。分類と異常検出の場合、返される詳細は最も高確率のクラスまたは指定されたclass_valueを参照します。

topN

topNに値を指定すると、このファンクションは予測に最も影響力のあるN個の属性(スコア)を返します。topNを指定しないと、このファンクションは最も影響力のある5つの属性を返します。

DESC、ASC、またはABS

返される属性が重みで順序付けされます。属性の重みは、予測に対する正の影響または負の影響を表します。回帰の場合、正の重みは予測される値が大きくなることを示し、負の重みは予測される値が小さくなることを示します。分類と異常検出の場合、正の重みは予測の確率が高くなることを示し、負の重みは予測の確率が低くなることを示します。

デフォルトでは、PREDICTION_DETAILSは、最大の正の重みを持つ属性を返します(DESC)。ASCを指定すると、最大の負の重みを持つ属性が返されます。ABSを指定すると、正負を問わずに、最大の重みを持つ属性が返されます。結果は、絶対値が高いほうから低いほうに順序付けされています。重みがゼロの属性は、出力に含まれません。

構文の選択

PREDICTION_DETAILSは、2つの方法のどちらかでデータにスコアを付けます。1つの方法では、データにマイニング・モデル・オブジェクトを適用します。もう1つの方法では、1つ以上の一時マイニング・モデルを作成して適用する分析句を実行して動的にデータをマイニングします。構文または分析構文を選択します。

- 構文 — 事前に定義されたモデルでデータにスコアを付ける場合は、最初の構文を使用します。分類、回帰または異常検出を実行するモデルの名前を指定します。
- 分析構文 — 事前定義されたモデルなしで、データにスコアを付ける場合は、分析構文を使用します。分析構文は、mining_analytic_clauseを使用します。これは、複数のモデル構築のためにデータをパーティション化する必要がある場合に指定します。mining_analytic_clauseは、query_partition_clauseとorder_by_clauseをサポートします。(「analytic_clause::=」を参照)
    - 分類の場合は、FOR exprを指定します。exprは、文字データ型のターゲット列を特定する式です。
    - 回帰の場合は、FOR exprを指定します。exprは、数値データ型のターゲット列を特定する式です。
    - 異常検出の場合は、キーワードOF ANOMALYを指定します。

PREDICTION_DETAILSファンクションの構文では、パーティション化されたモデルをスコアリングするときに、オプションのGROUPINGヒントを使用できます。「GROUPINGヒント」を参照してください。

mining_attribute_clause

mining_attribute_clauseは、スコアの予測子として使用する列の属性を特定します。このファンクションが分析構文で起動された場合は、これらの予測子が一時モデルの構築にも使用されます。mining_attribute_clauseは、

PREDICTIONファンクションと同様に動作します。(「mining_attribute_clause::=」を参照)

関連項目:

- スコアリングの詳細は、Oracle Data Miningユーザーズ・ガイドを参照してください。
- 予測データ・マイニングの詳細は、『Oracle Data Mining概要』を参照してください。

ノート:

次に示す例は、Data Mining のサンプル・プログラムからの抜粋です。サンプル・プログラムの詳細は、Oracle Data Mining ユーザーズ・ガイドの「付録 A」を参照してください。

例

この例では、モデルsvmr_sh_regr_sampleを使用して、データにスコアを付けます。この問合せは、顧客の年齢の予測値が高くなるのに最も影響力がある3つの属性を返します。

```
SELECT PREDICTION_DETAILS(svmr_sh_regr_sample, null, 3 USING *) prediction_details
    FROM mining_data_apply_v
    WHERE cust_id = 100001;

PREDICTION_DETAILS
--------------------------------------------------------------------------------
--
<Details algorithm="Support Vector Machines">
<Attribute name="CUST_MARITAL_STATUS" actualValue="Widowed" weight=".361" rank="1"/>
<Attribute name="CUST_GENDER" actualValue="F" weight=".14" rank="2"/>
<Attribute name="HOME_THEATER_PACKAGE" actualValue="1" weight=".135" rank="3"/>
</Details>
```

分析構文

この例は、標準的ではない年齢の顧客を動的に特定します。この問合せは、予測される年齢や標準から逸脱した年齢の属性を返します。

```
SELECT cust_id, age, pred_age, age-pred_age age_diff, pred_det
    FROM (SELECT cust_id, age, pred_age, pred_det,
          RANK() OVER (ORDER BY ABS(age-pred_age) DESC) rnk
          FROM (SELECT cust_id, age,
             PREDICTION(FOR age USING *) OVER () pred_age,
             PREDICTION_DETAILS(FOR age ABS USING *) OVER () pred_det
             FROM mining_data_apply_v))
    WHERE rnk <= 5;

CUST_ID AGE PRED_AGE AGE_DIFF  PRED_DET
------- --- -------- -------- ----------------------------------------------------
-----------
 100910  80    40.67    39.33 <Details algorithm="Support Vector Machines">
                               <Attribute name="HOME_THEATER_PACKAGE" actualValue="1"
weight=".059"
                                rank="1"/>
                               <Attribute name="Y_BOX_GAMES" actualValue="0"
weight=".059"
                                rank="2"/>
                               <Attribute name="AFFINITY_CARD" actualValue="0"
weight=".059"
                                rank="3"/>
```

```
                                <Attribute name="FLAT_PANEL_MONITOR" actualValue="1"
weight=".059"
                                 rank="4"/>
                                <Attribute name="YRS_RESIDENCE" actualValue="4"
weight=".059"
                                 rank="5"/>
                                </Details>

 101285  79    42.18    36.82  <Details algorithm="Support Vector Machines">
                                <Attribute name="HOME_THEATER_PACKAGE" actualValue="1"
weight=".059"
                                 rank="1"/>
                                <Attribute name="HOUSEHOLD_SIZE" actualValue="2"
weight=".059"
                                 rank="2"/>
                                <Attribute name="CUST_MARITAL_STATUS"
actualValue="Mabsent"
                                 weight=".059" rank="3"/>
                                <Attribute name="Y_BOX_GAMES" actualValue="0"
weight=".059"
                                 rank="4"/>
                                <Attribute name="OCCUPATION" actualValue="Prof."
weight=".059"
                                 rank="5"/>
                                </Details>

 100694   77    41.04    35.96  <Details algorithm="Support Vector Machines">
                                 <Attribute name="HOME_THEATER_PACKAGE"
actualValue="1"
                                  weight=".059" rank="1"/>
                                 <Attribute name="EDUCATION" actualValue="&lt; Bach."
weight=".059"
                                  rank="2"/>
                                 <Attribute name="Y_BOX_GAMES" actualValue="0"
weight=".059"
                                  rank="3"/>
                                 <Attribute name="CUST_ID" actualValue="100694"
weight=".059"
                                  rank="4"/>
                                 <Attribute name="COUNTRY_NAME" actualValue="United
States of
                                  America" weight=".059" rank="5"/>
                                 </Details>

 100308  81    45.33    35.67  <Details algorithm="Support Vector Machines">
                                <Attribute name="HOME_THEATER_PACKAGE" actualValue="1"
weight=".059"
                                 rank="1"/>
                                <Attribute name="Y_BOX_GAMES" actualValue="0"
weight=".059"
                                 rank="2"/>
                                <Attribute name="HOUSEHOLD_SIZE" actualValue="2"
weight=".059"
                                 rank="3"/>
                                <Attribute name="FLAT_PANEL_MONITOR" actualValue="1"
weight=".059"
                                 rank="4"/>
                                <Attribute name="CUST_GENDER" actualValue="F"
weight=".059"
                                 rank="5"/>
                                </Details>

 101256  90    54.39    35.61  <Details algorithm="Support Vector Machines">
                                <Attribute name="YRS_RESIDENCE" actualValue="9"
weight=".059"
                                 rank="1"/>
```

```
                                <Attribute name="HOME_THEATER_PACKAGE" actualValue="1"
weight=".059"
                                 rank="2"/>
                                <Attribute name="EDUCATION" actualValue="&lt; Bach."
weight=".059"
                                 rank="3"/>
                                <Attribute name="Y_BOX_GAMES" actualValue="0"
weight=".059"
                                 rank="4"/>
                                <Attribute name="COUNTRY_NAME" actualValue="United
States of
                                 America" weight=".059" rank="5"/>
                                </Details>
```

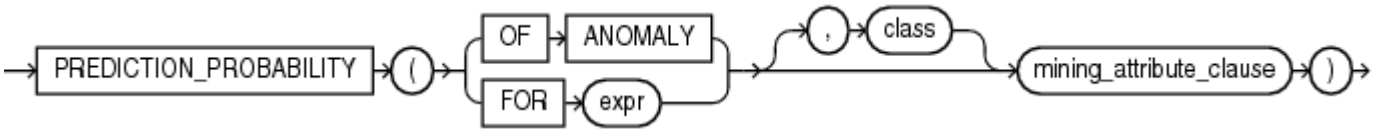
PREDICTION_PROBABILITY

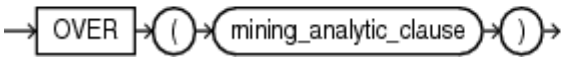
構文

prediction_probability::=

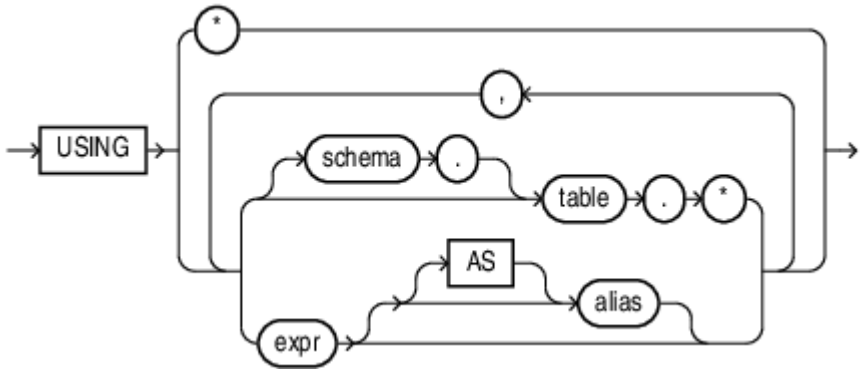
分析構文

prediction_prob_analytic::=

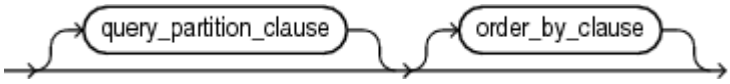




mining_attribute_clause::=



mining_analytic_clause::=



関連項目:

`mining_analytic_clause`の構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

`PREDICTION_PROBABILITY`は、選択内に含まれる各行の確率を返します。確率は、最も高確率のクラスまたは指定された`class`を参照します。返される確率のデータ型は、`BINARY_DOUBLE`です。

`PREDICTION_PROBABILITY`では、分類または異常検出を実行できます。分類の場合、返される確率は予測されたターゲット・クラスを参照します。異常検出の場合、返される確率は分類`1`(通常の行)または`0`(異常な行)を参照します。

`PREDICTION_PROBABILITY`と`PREDICTION`ファンクションを組み合せて使用すると、予測とその予測の確率を取得でき

ます。

構文の選択

PREDICTION_PROBABILITYは、2つの方法のどちらかでデータにスコアを付けます。1つの方法では、データにマイニング・モデル・オブジェクトを適用します。もう1つの方法では、1つ以上の一時マイニング・モデルを作成して適用する分析句を実行して動的にデータをマイニングします。構文または分析構文を選択します。

- 構文 — 事前に定義されたモデルでデータにスコアを付ける場合は、最初の構文を使用します。分類または異常検出を実行するモデルの名前を指定します。
- 分析構文 — 事前定義されたモデルなしで、データにスコアを付ける場合は、分析構文を使用します。分析構文は、mining_analytic_clauseを使用します。これは、複数のモデル構築のためにデータをパーティション化する必要がある場合に指定します。mining_analytic_clauseは、query_partition_clauseとorder_by_clauseをサポートします。(「analytic_clause::=」を参照)
  - 分類の場合は、FOR exprを指定します。exprは、文字データ型のターゲット列を特定する式です。
  - 異常検出の場合は、キーワードOF ANOMALYを指定します。

PREDICTION_PROBABILITYファンクションの構文では、パーティション化されたモデルをスコアリングするときに、オプションのGROUPINGヒントを使用できます。「GROUPINGヒント」を参照してください。

mining_attribute_clause

mining_attribute_clauseは、スコアの予測子として使用する列の属性を特定します。このファンクションが分析構文で起動された場合は、これらの予測子が一時モデルの構築にも使用されます。mining_attribute_clauseは、PREDICTIONファンクションと同様に動作します。(「mining_attribute_clause::=」を参照)

関連項目:

- スコアリングの詳細は、Oracle Data Miningユーザーズ・ガイドを参照してください。
- 予測データ・マイニングの詳細は、『Oracle Data Mining概要』を参照してください。

ノート:

次に示す例は、Data Mining のサンプル・プログラムからの抜粋です。サンプル・プログラムの詳細は、Oracle Data Mining ユーザーズ・ガイドの「付録 A」を参照してください。

例

次の例では、提携カードを使用している可能性が最も高いイタリア在住の顧客を10人戻します。

```
SELECT cust_id FROM (
   SELECT cust_id
   FROM mining_data_apply_v
   WHERE country_name = 'Italy'
   ORDER BY PREDICTION_PROBABILITY(DT_SH_Clas_sample, 1 USING *)
      DESC, cust_id)
   WHERE rownum < 11;

   CUST_ID
----------
    100081
```

```
    100179
    100185
    100324
    100344
    100554
    100662
    100733
    101250
    101306
```

分析の例

この例では、mining_data_one_class_v内のデータで最も標準的でない行を特定します。婚姻区分の各種別が個別に考慮されるため、婚姻区分グループごとに最も異常な行が返されるようになります。

この問合せは、異常な行の判断に最も影響を与える属性を返します。PARTITION BY句により、婚姻区分ごとに構築され適用される個別のモデルが生成されます。区分がMabsentのレコードは1つのみであるため、そのパーティションに作成されるモデルはありません(詳細も表示されません)。

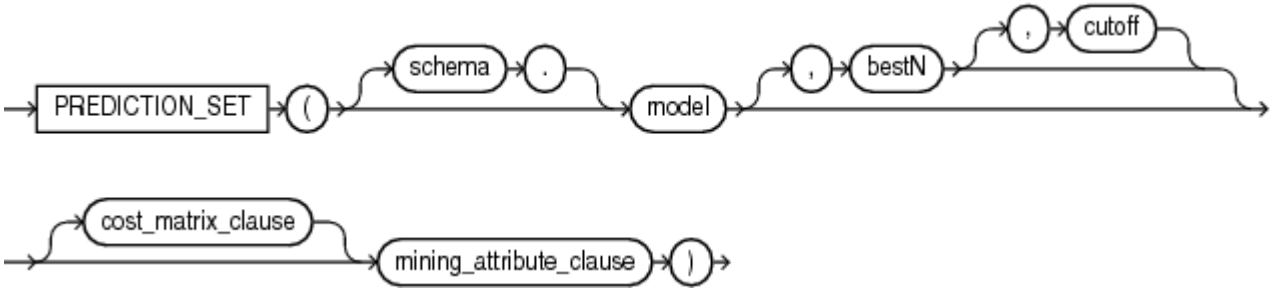
```
SELECT cust_id, cust_marital_status, rank_anom, anom_det FROM
    (SELECT cust_id, cust_marital_status, anom_det,
            rank() OVER (PARTITION BY CUST_MARITAL_STATUS
                         ORDER BY ANOM_PROB DESC,cust_id) rank_anom FROM
     (SELECT cust_id, cust_marital_status,
            PREDICTION_PROBABILITY(OF ANOMALY, 0 USING *)
              OVER (PARTITION BY CUST_MARITAL_STATUS) anom_prob,
            PREDICTION_DETAILS(OF ANOMALY, 0, 3 USING *)
              OVER (PARTITION BY CUST_MARITAL_STATUS) anom_det
     FROM mining_data_one_class_v
    ))
   WHERE rank_anom < 3 order by 2, 3;
CUST_ID CUST_MARITAL_STATUS RANK_ANOM  ANOM_DET
------- ------------------- ---------- ----------------------------------------------
-------------
102366  Divorc.             1          <Details algorithm="Support Vector Machines"
class="0">
                                       <Attribute name="COUNTRY_NAME"
actualValue="United Kingdom"
                                        weight=".069" rank="1"/>
                                       <Attribute name="AGE" actualValue="28"
weight=".013"
                                        rank="2"/>
                                       <Attribute name="YRS_RESIDENCE"
actualValue="4"
                                        weight=".006" rank="3"/>
                                       </Details>
101817  Divorc.             2          <Details algorithm="Support Vector Machines"
class="0">
                                       <Attribute name="YRS_RESIDENCE"
actualValue="8"
                                        weight=".018" rank="1"/>
                                       <Attribute name="EDUCATION" actualValue="PhD"
weight=".007"
                                        rank="2"/>
                                       <Attribute name="CUST_INCOME_LEVEL"
actualValue="K:
                                        250¥,000 - 299¥,999" weight=".006" rank="3"/>
                                       </Details>

101713  Mabsent             1

101790  Married             1          <Details algorithm="Support Vector Machines"
class="0">
                                       <Attribute name="COUNTRY_NAME"
actualValue="Canada"
```

```
                                        weight=".063" rank="1"/>
                                       <Attribute name="EDUCATION" actualValue="7th-
8th"
                                        weight=".011" rank="2"/>
                                       <Attribute name="HOUSEHOLD_SIZE"
actualValue="4-5"
                                        weight=".011" rank="3"/>
                                       </Details>
. . .
```

PREDICTION_SET

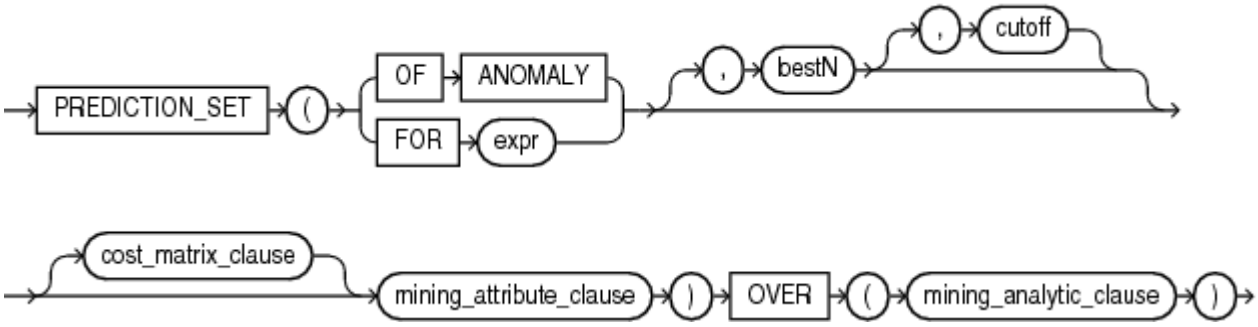
構文

prediction_set::=

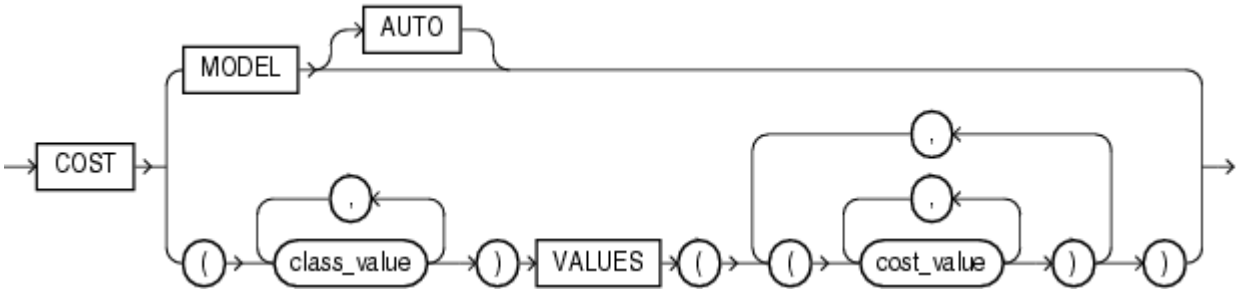


分析構文

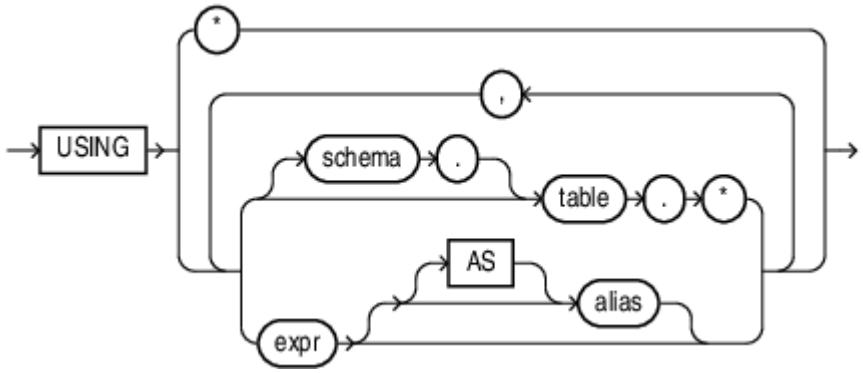
prediction_set_analytic::=



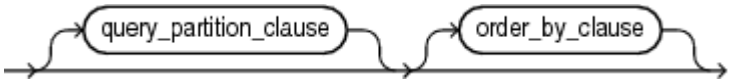
cost_matrix_clause::=



mining_attribute_clause::=



mining_analytic_clause::-



関連項目:

mining_analytic_clauseの構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

PREDICTION_SETは、選択内に含まれる各行の確率またはコストとともに一連の予測を戻します。戻り値は、フィールド名がPREDICTION_IDおよびPROBABILITYまたはCOSTのオブジェクトのVARRAYです。予測識別子はターゲットのデータ型を持ちます。確率およびコストのフィールドはBINARY_DOUBLEです。

PREDICTION_SETでは、分類または異常検出を実行できます。分類の場合、戻り値は予測されたターゲット・クラスを参照します。異常検出の場合、戻り値は分類1(通常の行)または0(異常な行)を参照します。

bestNおよびcutoff

bestNとcutoffを指定すると、このファンクションから返される予測の数を制限できます。デフォルトでは、bestNおよびcutoffがnullであり、すべての予測が戻されます。

- bestNは、最も確率が高いか、コストが低いN個の予測です。N番目の確率またはコストを持つ予測が複数ある場合でも、ファンクションが選択するのはそのうち1つのみです。
- cutoffは、値のしきい値です。cutoff以上の確率か、cutoff以下のコストの予測のみが返されます。cutoffのみでフィルタ処理するには、bestNにNULLを指定します。問合せに使用するcost_matrix_clauseにCOST MODEL AUTOを指定すると、cutoffは無視されます。

cutoffとともにbestNを指定して、cutoff以上の最も確率が高い最大N個の予測を戻すことができます。コストが使用される場合、cutoffとともにbestNを指定して、cutoff以下の最もコストが低い最大N個の予測を戻します。

cost_matrix_clause

最も悪影響を及ぼす誤分類を最小限に抑えるために、cost_matrix_clauseをバイアス係数として指定できます。cost_matrix_clauseは、「PREDICTION_COST」で説明されているように動作します。

構文の選択

PREDICTION_SETは、2つの方法のどちらかでデータにスコアを付けます。1つの方法では、データにマイニング・モデル・オブジェクトを適用します。もう1つの方法では、1つ以上の一時マイニング・モデルを作成して適用する分析句を実行して動的にデータをマイニングします。構文または分析構文を選択します。

- 構文 — 事前に定義されたモデルでデータにスコアを付ける場合は、最初の構文を使用します。分類または異常検出を実行するモデルの名前を指定します。
- 分析構文 — 事前定義されたモデルなしで、データにスコアを付ける場合は、分析構文を使用します。分析構文は、mining_analytic_clauseを使用します。これは、複数のモデル構築のためにデータをパーティション化する必要がある場合に指定します。mining_analytic_clauseは、query_partition_clauseとorder_by_clauseをサポートします。(「analytic_clause::=」を参照)
  - 分類の場合は、FOR exprを指定します。exprは、文字データ型のターゲット列を特定する式です。
  - 異常検出の場合は、キーワードOF ANOMALYを指定します。

PREDICTION_SETファンクションの構文では、パーティション化されたモデルをスコアリングするときに、オプションのGROUPINGヒントを使用できます。「GROUPINGヒント」を参照してください。

mining_attribute_clause

mining_attribute_clauseは、スコアの予測子として使用する列の属性を特定します。このファンクションが分析構文で起動された場合は、これらの予測子が一時モデルの構築にも使用されます。mining_attribute_clauseは、PREDICTIONファンクションと同様に動作します。(「mining_attribute_clause::=」を参照)

関連項目:

- スコアリングの詳細は、Oracle Data Miningユーザーズ・ガイドを参照してください。
- 予測データ・マイニングの詳細は、『Oracle Data Mining概要』を参照してください。

ノート:

次に示す例は、Data Mining のサンプル・プログラムからの抜粋です。サンプル・プログラムの詳細は、Oracle Data Mining ユーザーズ・ガイドの「付録 A」を参照してください。

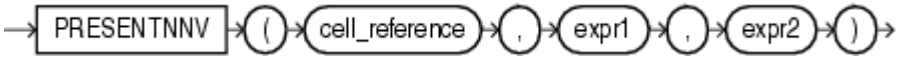
例

この例では、100006より小さいIDを持つ顧客が提携カードを使用する予測とコストを示します。この例のターゲットはバイナリですが、このような問合せは低、中、高などの複数クラスの分類にも有効です。

```
SELECT T.cust_id, S.prediction, S.probability, S.cost
  FROM (SELECT cust_id,
               PREDICTION_SET(dt_sh_clas_sample COST MODEL USING *) pset
          FROM mining_data_apply_v
         WHERE cust_id < 100006) T,
       TABLE(T.pset) S
ORDER BY cust_id, S.prediction;

   CUST_ID PREDICTION  PROBABILITY         COST
---------- ---------- ------------ ------------
    100001          0   .966183575   .270531401
    100001          1   .033816425   .966183575
    100002          0   .740384615  2.076923077
    100002          1   .259615385   .740384615
    100003          0   .909090909   .727272727
    100003          1   .090909091   .909090909
    100004          0   .909090909   .727272727
    100004          1   .090909091   .909090909
    100005          0   .272357724  5.821138211
    100005          1   .727642276   .272357724
```

PRESENTNNV

構文



目的

PRESENTNNVファンクションは、SELECT文のmodel_clauseでのみ、およびモデル・ルールの右側でのみ使用できます。このファンクションは、cell_referenceがmodel_clauseの実行前に存在し、PRESENTNNVの評価時にNULLではない場合にexpr1を戻します。それ以外の場合はexpr2を戻します。このファンクションは、NVL2とは異なります。NVL2は、model_clauseの実行前の状態のデータを評価するのではなく、実行時のデータを評価します。

関連項目:

- 構文およびセマンティクスの詳細は、「model_clause」および「モデル式」を参照してください。
- 比較については、「NVL2」を参照してください。
- PRESENTNNVの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、2002年のマウス・パッドの売上を含む行が存在し、その売上値がNULLではない場合、その売上値を変更しません。その行が存在し、売上値がNULLの場合、その売上値を10に設定します。その行が存在しない場合、売上値を10に設定して行を作成します。

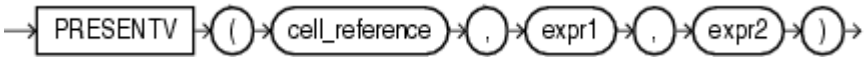
```
SELECT country, prod, year, s
  FROM sales_view_ref
  MODEL
    PARTITION BY (country)
    DIMENSION BY (prod, year)
    MEASURES (sale s)
    IGNORE NAV
    UNIQUE DIMENSION
    RULES UPSERT SEQUENTIAL ORDER
    ( s['Mouse Pad', 2002] =
        PRESENTNNV(s['Mouse Pad', 2002], s['Mouse Pad', 2002], 10)
    )
  ORDER BY country, prod, year;
COUNTRY       PROD                                       YEAR          S
----------    ----------------------------------      --------   ---------
France        Mouse Pad                                  1998     2509.42
France        Mouse Pad                                  1999     3678.69
France        Mouse Pad                                  2000     3000.72
France        Mouse Pad                                  2001     3269.09
France        Mouse Pad                                  2002          10
France        Standard Mouse                             1998     2390.83
France        Standard Mouse                             1999     2280.45
France        Standard Mouse                             2000     1274.31
France        Standard Mouse                             2001     2164.54
Germany       Mouse Pad                                  1998     5827.87
Germany       Mouse Pad                                  1999     8346.44
Germany       Mouse Pad                                  2000     7375.46
Germany       Mouse Pad                                  2001     9535.08
Germany       Mouse Pad                                  2002          10
Germany       Standard Mouse                             1998     7116.11
```

```
Germany       Standard Mouse                                  1999     6263.14
Germany       Standard Mouse                                  2000     2637.31
Germany       Standard Mouse                                  2001     6456.13
18 rows selected.
```

この例では、ビュー`sales_view_ref`が必要です。このビューを作成する方法は、「例」を参照してください。

PRESENTV

構文



目的

PRESENTVファンクションは、SELECT文のmodel_clauseでのみ、およびモデル・ルールの右側でのみ使用できます。このファンクションは、cell_referenceが存在する場合、model_clauseを実行する前にexpr1を戻します。それ以外の場合はexpr2を戻します。

関連項目:

- 構文およびセマンティクスの詳細は、「model_clause」および「モデル式」を参照してください。
- PRESENTVの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

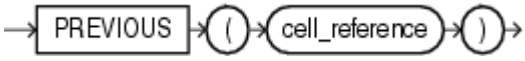
次の例では、2000年のマウス・パッドの売上を含む行が存在する場合、2001年のマウス・パッドの売上を2000年のマウス・パッドの売上値に設定します。その行が存在しない場合、2001年のマウス・パッドの売上値を0(ゼロ)に設定して行を作成します。

```
SELECT country, prod, year, s
  FROM sales_view_ref
  MODEL
    PARTITION BY (country)
    DIMENSION BY (prod, year)
    MEASURES (sale s)
    IGNORE NAV
    UNIQUE DIMENSION
    RULES UPSERT SEQUENTIAL ORDER
    (
      s['Mouse Pad', 2001] =
        PRESENTV(s['Mouse Pad', 2000], s['Mouse Pad', 2000], 0)
    )
  ORDER BY country, prod, year;
COUNTRY       PROD                                    YEAR          S
----------    ----------------------------------      --------   ---------
France        Mouse Pad                                  1998     2509.42
France        Mouse Pad                                  1999     3678.69
France        Mouse Pad                                  2000     3000.72
France        Mouse Pad                                  2001     3000.72
France        Standard Mouse                             1998     2390.83
France        Standard Mouse                             1999     2280.45
France        Standard Mouse                             2000     1274.31
France        Standard Mouse                             2001     2164.54
Germany       Mouse Pad                                  1998     5827.87
Germany       Mouse Pad                                  1999     8346.44
Germany       Mouse Pad                                  2000     7375.46
Germany       Mouse Pad                                  2001     7375.46
Germany       Standard Mouse                             1998     7116.11
Germany       Standard Mouse                             1999     6263.14
Germany       Standard Mouse                             2000     2637.31
Germany       Standard Mouse                             2001     6456.13
16 rows selected.
```

この例では、ビュー`sales_view_ref`が必要です。このビューを作成する方法は、「MODEL句：例」を参照してください。

PREVIOUS

構文



目的

PREVIOUSファンクションは、SELECT文のmodel_clauseでのみ、およびmodel_rules_clauseのITERATE ... [UNTIL]句でのみ使用できます。このファンクションは、各反復の最初にcell_referenceの値を戻します。

関連項目:

- 構文およびセマンティクスの詳細は、「model_clause」および「モデル式」を参照してください。
- PREVIOUSの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

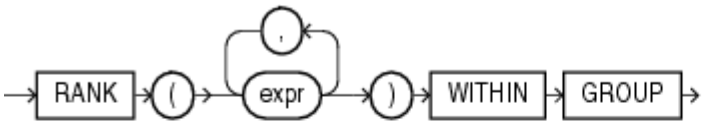
次の例では、反復の最初と最後にあるcur_valの値の差が1未満になるまで、最大1000回ルールを繰り返します。

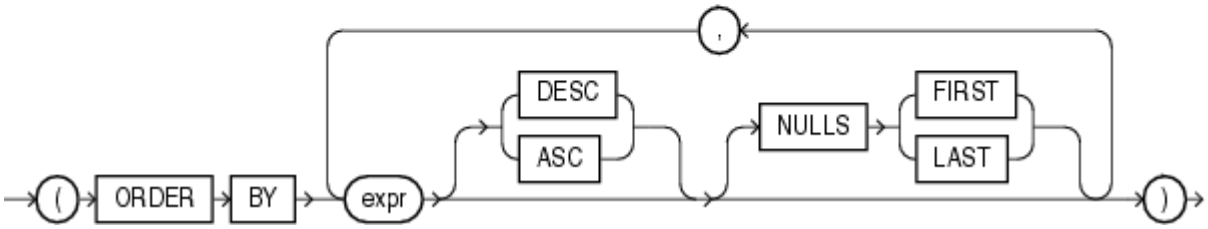
```
SELECT dim_col, cur_val, num_of_iterations
  FROM (SELECT 1 AS dim_col, 10 AS cur_val FROM dual)
  MODEL
    DIMENSION BY (dim_col)
    MEASURES (cur_val, 0 num_of_iterations)
    IGNORE NAV
    UNIQUE DIMENSION
    RULES ITERATE (1000) UNTIL (PREVIOUS(cur_val[1]) - cur_val[1] < 1)
    (
      cur_val[1] = cur_val[1]/2,
      num_of_iterations[1] = num_of_iterations[1] + 1
    );
   DIM_COL    CUR_VAL NUM_OF_ITERATIONS
---------- ---------- -----------------
         1       .625                 4
```

RANK

集計の構文

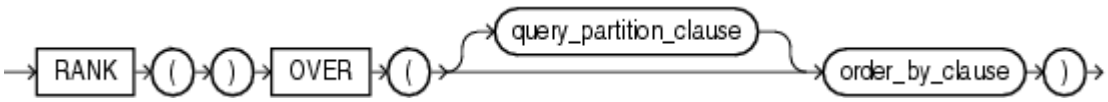
rank_aggregate::=





分析構文

rank_analytic::=



関連項目:

構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

RANKは、一連の値における値のランクを計算します。戻り型は、NUMBERです。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。数値の優先順位の詳細は、「数値の優先順位の詳細」を参照してください。

ランク付け基準と同じ値を持つ行は、同じランクになります。Oracle Databaseは連結行の数を連結ランクに追加して、次のランクを計算します。そのため、ランクは連続した数値でない場合があります。このファンクションは、上位N番および下位N番のレポートに有効です。

- 集計ファンクションとして使用するRANKは、ファンクションの引数として識別される不確定な行のランクを、指定されたソート指定に従って計算します。ファンクションの引数は、各集計グループの単一行を識別するため、すべての引数は各集計グループ内で定数式に評価される必要があります。定数引数式および集計のORDER BY句の式の位置は、一致します。このため、引数の数は同じであり、その型は互換性がある必要があります。
- 分析ファンクションとして使用するRANKは、ある問合せが戻す他の行について、その問合せが戻す各行のランクを計算します。この計算は、`order_by_clause`にある`value_exprs`の値に基づいて行われます。

関連項目:

ORDER BY句の文字値を比較するためにRANKで使用する照合を定義する照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

集計の例

次の例では、サンプル表hr.employeesから、給与が$15,500であり、歩合が5%である不確定な従業員のランクを計算します。

```
SELECT RANK(15500, .05) WITHIN GROUP
   (ORDER BY salary, commission_pct) "Rank"
   FROM employees;
      Rank
----------
       105
```

同様に、次の問合せは、従業員の給与から給与が$15,500のランクを戻します。

```
SELECT RANK(15500) WITHIN GROUP
   (ORDER BY salary DESC) "Rank of 15500"
   FROM employees;
Rank of 15500
--------------
             4
```

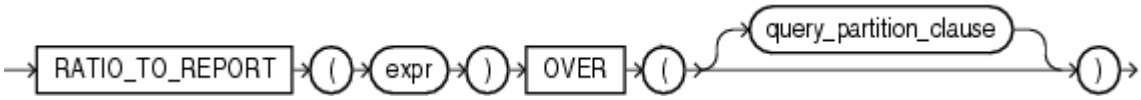
分析の例

次の文は、サンプル・スキーマhrの部門60の従業員を、その給与に基づいてランク付けします。給与が同じ場合は同じランクになるため、連続しないランクになります。この例と、「DENSE_RANK」の分析の例を比較してください。

```
SELECT department_id, last_name, salary,
       RANK() OVER (PARTITION BY department_id ORDER BY salary) RANK
  FROM employees WHERE department_id = 60
  ORDER BY RANK, last_name;
DEPARTMENT_ID LAST_NAME                      SALARY       RANK
------------- ------------------------- ---------- ----------
           60 Lorentz                         4200          1
           60 Austin                          4800          2
           60 Pataballa                       4800          2
           60 Ernst                           6000          4
           60 Hunold                          9000          5
```

RATIO_TO_REPORT

構文



関連項目:

構文、セマンティクス、制限事項、およびexprの書式の詳細は、「分析ファンクション」を参照してください。

目的

RATIO_TO_REPORTは分析ファンクションです。このファンクションは、ある値の集合の合計に対する、その値の比率を計算します。exprがNULLの場合は、値もNULLになります。

値の集合は、query_partition_clauseによって決まります。この句を省略すると、比率は、問合せによって戻されるすべての行で計算されます。

exprには、RATIO_TO_REPORTまたは他の分析ファンクションを使用して分析ファンクションをネストできません。ただし、他の組込みファンクション式をexprで使用できます。exprの書式の詳細は、「SQL式」を参照してください。

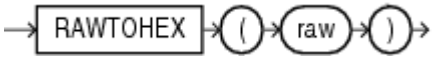
例

次の例では、すべての事務員の給与の合計に対する各事務員の給与の割合の値を計算します。

```
SELECT last_name, salary, RATIO_TO_REPORT(salary) OVER () AS rr
   FROM employees
   WHERE job_id = 'PU_CLERK'
   ORDER BY last_name, salary, rr;
LAST_NAME                 SALARY         RR
------------------------- ---------- ----------
Baida                           2900 .208633094
Colmenares                      2500 .179856115
Himuro                          2600  .18705036
Khoo                            3100 .223021583
Tobias                          2800 .201438849
```

RAWTOHEX

構文



目的

RAWTOHEXは、rawを16進表記で表した文字値に変換します。

SQL組込みファンクションとして使用される場合、RAWTOHEXは、LONG、LONG RAW、CLOB、NCLOB、BLOB, またはBFILE以外のすべてのスカラー・データ型の引数を取ります。引数がRAW以外のデータ型の場合、このファンクションは一定数のデータ・バイトを使用して表した引数値を同じ数のデータ・バイトのRAW値に変換します。データ自体は変更されませんが、そのデータ型はRAWデータ型に再キャストされます。

このファンクションは、rawの値を構成するバイトの16進表記でVARCHAR2値を戻します。各バイトは、2桁の16進数で表されます。

ノート:

RAWTOHEX は、PL/SQL 組込みファンクションとして使用される場合は異なる動作をします。詳細は、『Oracle Database 開発ガイド』を参照してください。

関連項目:

RAWTOHEXの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例は、RAW列の値に等しい16進を戻します。

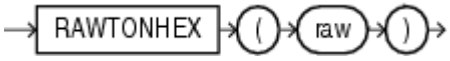
```
SELECT RAWTOHEX(raw_column) "Graphics"
   FROM graphics;
Graphics
--------
7D
```

関連項目:

「RAWデータ型とLONG RAWデータ型」および「HEXTORAW」

RAWTONHEX

構文



目的

RAWTONHEXは、rawを16進表記で表した文字値に変換します。RAWTONHEX(raw)は、TO_NCHAR(RAWTOHEX(raw))と同じです。戻り値は、常に各国語文字セットです。

ノート:

RAWTONHEX は、PL/SQL 組込みファンクションとして使用される場合は異なる動作をします。詳細は、『Oracle Database 開発ガイド』を参照してください。

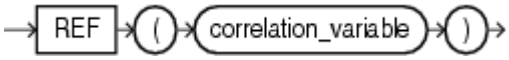
例

次の例は、RAW列の値に等しい16進を戻します。

```
SELECT RAWTONHEX(raw_column),
   DUMP ( RAWTONHEX (raw_column) ) "DUMP"
   FROM graphics;
RAWTONHEX(RA)           DUMP
----------------------- ------------------------------
7D                      Typ=1 Len=4: 0,55,0,68
```

REF

構文



目的

REFは、引数としてオブジェクト表またはオブジェクト・ビューの行に関連付けられた相関変数(表別名)を取ります。変数または行にバインドされたオブジェクト・インスタンスについてのREF値が戻ります。

例

サンプル・スキーマoeには、次のように定義されたcust_address_typという型が含まれます。

```
 Attribute                                 Type
 ----------------------------- ----------------
 STREET_ADDRESS                    VARCHAR2(40)
 POSTAL_CODE                       VARCHAR2(10)
 CITY                              VARCHAR2(30)
 STATE_PROVINCE                    VARCHAR2(10)
 COUNTRY_ID                             CHAR(2)
```

次の例では、サンプル型oe.cust_address_typに基づく表(addresses表)を作成した後、その表に行を挿入し、その表からその型のオブジェクト・インスタンスのREF値を取り出します。

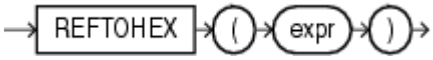
```
CREATE TABLE addresses OF cust_address_typ;
INSERT INTO addresses VALUES (
   '123 First Street', '4GF H1J', 'Our Town', 'Ourcounty', 'US');
SELECT REF(e) FROM addresses e;
REF(E)
--------------------------------------------------------------------------------
00002802097CD1261E51925B60E0340800208254367CD1261E51905B60E0340800208254360101018200
0
```

関連項目:

REFの詳細は、『Oracle Databaseオブジェクト・リレーショナル開発者ガイド』を参照してください。

REFTOHEX

構文



目的

REFTOHEXは、引数exprを16進で表した文字値に変換します。exprは、REFを戻す必要があります。

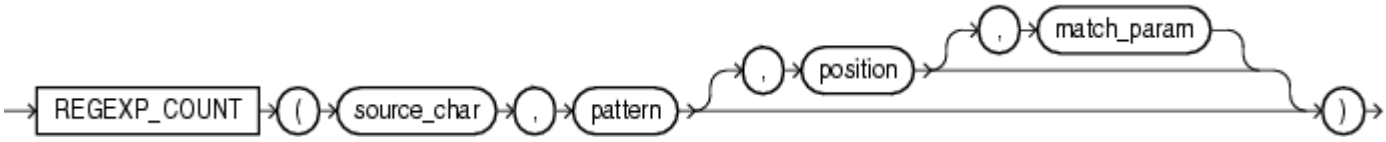
例

サンプル・スキーマoeにはwarehouse_typが含まれています。次の例では、その型を基に、ある列のREF値をそれに等しい16進を含む文字値に変換する方法を示します。

```
CREATE TABLE warehouse_table OF warehouse_typ
   (PRIMARY KEY (warehouse_id));
CREATE TABLE location_table
   (location_number NUMBER, building REF warehouse_typ
   SCOPE IS warehouse_table);
INSERT INTO warehouse_table VALUES (1, 'Downtown', 99);
INSERT INTO location_table SELECT 10, REF(w) FROM warehouse_table w;
SELECT REFTOHEX(building) FROM location_table;
REFTOHEX(BUILDING)
--------------------------------------------------------------------------------
0000220208859B5E9255C31760E034080020825436859B5E9255C21760E034080020825436
```

REGEXP_COUNT

構文



目的

REGEXP_COUNTは、ソース文字列でパターンが発生した回数を戻すことによって、REGEXP_INSTRファンクションの機能を補完します。このファンクションでは、入力文字セットによって定義された文字を使用して文字列を評価します。また、patternの発生回数を示す整数を戻します。一致する値が見つからない場合は0(ゼロ)を戻します。

- source_charは、検索値として使用される文字式です。通常は文字列であり、データ型はCHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBのいずれかです。
- patternは正規表現です。通常はテキスト・リテラルであり、データ型はCHAR、VARCHAR2、NCHARまたはNVARCHAR2のいずれかになります。ここには、 512バイトまで入力できます。patternのデータ型がsource_charのデータ型と異なる場合、Oracleはpatternをsource_charのデータ型に変換します。

  REGEXP_COUNTは、pattern内の部分正規表現のカッコを無視します。たとえば、パターン'(123(45))'は、'12345'と同じです。patternで指定できる演算子のリストは、「Oracleの正規表現のサポート」を参照してください。
- positionは、Oracleが検索を開始する文字source_charの位置を示す正の整数です。デフォルトは1で、source_charの最初の文字から検索が開始されます。1番目のpatternの出現が検出されると、その後に続く文字以降の2番目の出現が検索されます。
- match_paramは、データ型がVARCHAR2またはCHARの文字式であり、ファンクションのデフォルトの照合動作を変更できます。

  match_paramの値には、次の文字を1つ以上含めることができます。
  - 'i'を指定すると、条件に対して決定されている照合が大/小文字の区別ありであっても、大/小文字の区別なしでの照合となります。
  - 'c'を指定すると、条件に対して決定されている照合が大/小文字の区別なしまたはアクセントの区別なしの場合でも、大/小文字の区別あり、アクセントの区別ありでの照合となります。
  - 'n': 任意の文字に一致する文字であるピリオド(.)を、改行文字の一致に使用します。このパラメータを指定しない場合、ピリオドは改行文字には一致しません。
  - 'm': ソース文字列を複数行として処理します。Oracleは、キャレット(^)およびドル記号($)を、それぞれ、ソース文字列全体の開始または終了としてのみではなく、ソース文字列内の任意の場所にある任意の行の開始または終了としても解釈します。このパラメータを指定しない場合、Oracleはソース文字列を単一行として処理します。
  - 'x'は空白文字を無視します。デフォルトでは、空白文字は空白文字として一致します。

  相反する文字がmatch_paramの値に複数含まれている場合は、最後の文字が使用されます。たとえば、'ic'を指定した場合は、大/小文字の区別あり、アクセントの区別ありでの照合が使用されます。値に前述以外の文字が含

まれている場合は、エラーが戻されます。

match_paramを指定しない場合、次のようになります。

- 大/小文字およびアクセントを区別するかどうかのデフォルトは、REGEXP_COUNTファンクションに対して決定されている照合によって決まります。
- ピリオド(.)は改行文字に一致しません。
- ソース文字列は単一行として処理されます。

関連項目:

source_charの文字をpatternの文字と比較するためにREGEXP_COUNTで使用する照合を定義する照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、pattern内の部分正規表現のカッコが無視されることを示します。

```
SELECT REGEXP_COUNT('123123123123', '(12)3', 1, 'i') REGEXP_COUNT
   FROM DUAL;

REGEXP_COUNT
------------
           5
```

次の例では、ファンクションは3番目の文字でソース文字列の評価を開始するため、patternの最初の出現はスキップされます。

```
SELECT REGEXP_COUNT('123123123123', '123', 3, 'i') COUNT FROM DUAL;
     COUNT
----------
         3
```

REGEXP_COUNTの単純な一致: 例

次の例では、REGEXP_COUNTは指定されたパターンの入力文字列を検証し、アルファベット文字の数を戻します。

```
select regexp_count('ABC123', '[A-Z]'), regexp_count('A1B2C3', '[A-Z]') from dual;
REGEXP_COUNT('ABC123','[A-Z]') REGEXP_COUNT('A1B2C3','[A-Z]')
------------------------------ ------------------------------
                             3                              3
```

次の例では、REGEXP_COUNTは指定されたパターンの入力文字列を検証し、1桁の数字が続くアルファベット文字の数を戻します。

```
select regexp_count('ABC123', '[A-Z][0-9]'), regexp_count('A1B2C3', '[A-Z][0-9]')
from dual;
REGEXP_COUNT('ABC123','[A-Z][0-9]') REGEXP_COUNT('A1B2C3','[A-Z][0-9]')
----------------------------------- -----------------------------------
                                  1                                   3
```

次の例では、REGEXP_COUNTは指定されたパターンの入力文字列を検証し、文字列の先頭にあるものにかぎり、1桁の数字が続くアルファベット文字の数を戻します。

```
select regexp_count('ABC123', '[A-Z][0-9]'), regexp_count('A1B2C3', '[A-Z][0-9]')
from dual;
REGEXP_COUNT('ABC123','^[A-Z][0-9]') REGEXP_COUNT('A1B2C3','^[A-Z][0-9]')
------------------------------------ ------------------------------------
```

```
                                    0                                    1
```

次の例では、REGEXP_COUNTは指定されたパターンの入力文字列を検証し、文字列中に含まれるものにかぎり、2桁の数字が続くアルファベット文字の数を戻します。

```
select regexp_count('ABC123', '[A-Z][0-9]{2}'), regexp_count('A1B2C3', '[A-Z][0-
9]{2}') from dual;
REGEXP_COUNT('ABC123','[A-Z][0-9]{2}') REGEXP_COUNT('A1B2C3','[A-Z][0-9]{2}')
-------------------------------------- --------------------------------------
                                     1                                     0
```

次の例では、REGEXP_COUNTは指定されたパターンの入力文字列を検証し、文字列の先頭から最初の2回の出現の中で、1桁の数字が続くアルファベット文字の数を戻します。

```
select regexp_count('ABC123', '([A-Z][0-9]){2}'), regexp_count('A1B2C3', '([A-Z][0-
9]){2}') from dual;
REGEXP_COUNT('ABC123','([A-Z][0-9]){2}') REGEXP_COUNT('A1B2C3','([A-Z][0-9]){2}')
---------------------------------------- ----------------------------------------
                                       0                                      1
```

Live SQL:

REGEXP_COUNT の単純な一致で、Oracle Live SQL に関連する例を参照および実行します

REGEXP_COUNTの詳細な一致: 例

次の例では、REGEXP_COUNTは指定されたパターンの入力文字列を検証し、アルファベット文字の数を戻します。

```
select regexp_count('ABC123', '[A-Z]') Match_char_ABC_count,
regexp_count('A1B2C3', '[A-Z]') Match_char_ABC_count from dual;
MATCH_CHAR_ABC_COUNT MATCH_CHAR_ABC_COUNT
-------------------- --------------------
                   3                    3
```

次の例では、REGEXP_COUNTは指定されたパターンの入力文字列を検証し、1桁の数字が続くアルファベット文字の数を戻します。

```
select regexp_count('ABC123', '[A-Z][0-9]') Match_string_C1_count,
regexp_count('A1B2C3', '[A-Z][0-9]')  Match_strings_A1_B2_C3_count from dual;
MATCH_STRING_C1_COUNT MATCH_STRINGS_A1_B2_C3_COUNT
--------------------- ----------------------------
                    1                            3
```

次の例では、REGEXP_COUNTは指定されたパターンの入力文字列を検証し、文字列の先頭にあるものにかぎり、1桁の数字が続くアルファベット文字の数を戻します。

```
select regexp_count('ABC123A5', '^[A-Z][0-9]') Char_num_like_A1_at_start,
regexp_count('A1B2C3', '^[A-Z][0-9]') Char_num_like_A1_at_start from dual;
CHAR_NUM_LIKE_A1_AT_START CHAR_NUM_LIKE_A1_AT_START
------------------------- -------------------------
                        0                         1
```

次の例では、REGEXP_COUNTは指定されたパターンの入力文字列を検証し、文字列中に含まれるものにかぎり、2桁の数字が続くアルファベット文字の数を戻します。

```
select regexp_count('ABC123', '[A-Z][0-9]{2}') Char_num_like_A12_anywhere,
regexp_count('A1B2C34', '[A-Z][0-9]{2}') Char_num_like_A12_anywhere from dual;
CHAR_NUM_LIKE_A12_ANYWHERE CHAR_NUM_LIKE_A12_ANYWHERE
-------------------------- --------------------------
```

```
                         1                         1
```

次の例では、REGEXP_COUNTは指定されたパターンの入力文字列を検証し、文字列の先頭から最初の2回の出現の中で、1桁の数字が続くアルファベット文字の数を戻します。

```
select regexp_count('ABC12D3', '([A-Z][0-9]){2}') Char_num_within_2_places,
regexp_count('A1B2C3', '([A-Z][0-9]){2}') Char_num_within_2_places from dual;
CHAR_NUM_WITHIN_2_PLACES CHAR_NUM_WITHIN_2_PLACES
------------------------ ------------------------
                       0                        1
```

Live SQL:

REGEXP_COUNT の詳細な一致で、Oracle Live SQL に関連する例を参照および実行します

REGEXP_COUNTの大/小文字を区別した一致: 例

次の文は、表regexp_tempを作成し、そこに値を挿入します。

```
CREATE TABLE regexp_temp(empName varchar2(20));
INSERT INTO regexp_temp (empName) VALUES ('John Doe');
INSERT INTO regexp_temp (empName) VALUES ('Jane Doe');
```

次の例では、この文は従業員名の列を問合せ、文字「E」の小文字を検索します。

```
SELECT empName, REGEXP_COUNT(empName, 'e', 1, 'c') "CASE_SENSITIVE_E" From
regexp_temp;
EMPNAME              CASE_SENSITIVE_E
-------------------- ----------------
John Doe                            1
Jane Doe                            2
```

次の例では、この文は従業員名の列を問合せ、文字「O」の小文字を検索します。

```
SELECT empName, REGEXP_COUNT(empName, 'o', 1, 'c') "CASE_SENSITIVE_O" From
regexp_temp;
EMPNAME              CASE_SENSITIVE_O
-------------------- ----------------
John Doe                            2
Jane Doe                            1
```

次の例では、この文は従業員名の列を問合せ、文字「E」の小文字または大文字を検索します。

```
SELECT empName, REGEXP_COUNT(empName, 'E', 1, 'i') "CASE_INSENSITIVE_E" From
regexp_temp;
EMPNAME              CASE_INSENSITIVE_E
-------------------- ------------------
John Doe                              1
Jane Doe                              2
```

次の例では、この文は従業員名の列を問合せ、文字列「DO」の小文字を検索します。

```
SELECT empName, REGEXP_COUNT(empName, 'do', 1, 'i') "CASE_INSENSITIVE_STRING" From
regexp_temp;
EMPNAME              CASE_INSENSITIVE_STRING
-------------------- -----------------------
John Doe                                   1
Jane Doe                                   1
```

次の例では、この文は従業員名の列を問合せ、文字列「AN」の小文字または大文字を検索します。

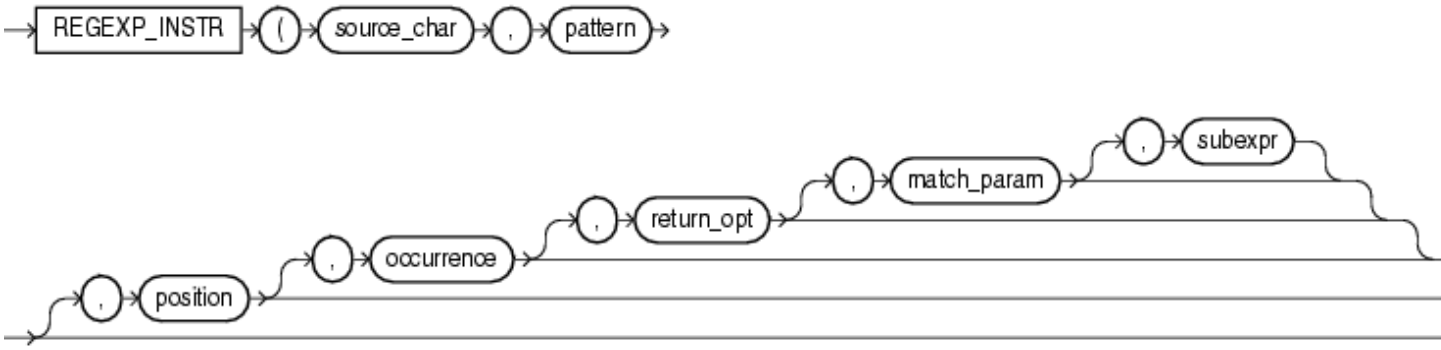
```
SELECT empName, REGEXP_COUNT(empName, 'an', 1, 'c') "CASE_SENSITIVE_STRING" From
regexp_temp;
EMPNAME              CASE_SENSITIVE_STRING
-------------------- ---------------------
John Doe                                0
Jane Doe                                1
```

Live SQL:

REGEXP_COUNT の大/小文字を区別した一致で、Oracle Live SQL に関連する例を参照および実行します

REGEXP_INSTR

構文



目的

REGEXP_INSTRは、INSTRファンクションの機能を拡張するものであり、正規表現パターンの文字列を検索できます。このファンクションでは、入力文字セットによって定義された文字を使用して文字列を評価します。また、return_option引数の値に基づいて、一致したサブストリングの開始位置または終了位置を示す整数を戻します。一致する値が見つからない場合は0(ゼロ)を戻します。

このファンクションは、POSIX正規表現規格およびUnicode Regular Expression Guidelinesに準拠しています。詳細は、「Oracleの正規表現のサポート」を参照してください。

- source_charは、検索値として使用される文字式です。通常は文字列であり、データ型はCHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBのいずれかです。
- patternは正規表現です。通常はテキスト・リテラルであり、データ型はCHAR、VARCHAR2、NCHARまたはNVARCHAR2のいずれかになります。ここには、 512バイトまで入力できます。patternのデータ型がsource_charのデータ型と異なる場合、Oracleはpatternをsource_charのデータ型に変換します。patternで指定できる演算子のリストは、「Oracleの正規表現のサポート」を参照してください。
- positionは、Oracleが検索を開始する文字source_charの位置を示す正の整数です。デフォルトは1で、source_charの最初の文字から検索が開始されます。
- occurrenceは、source_charに出現するpatternのうちのどれを検索するかを示す正の整数です。デフォルトは1で、最初に出現するpatternが検索されます。occurrenceが1より大きい場合、1番目のpatternが検出された後に続く1文字目から、2番目(以降)の出現を検索します。この動作は、最初の出現の2番目の文字から2番目の出現を検索するINSTRファンクションとは異なります。
- return_optionには、出現した文字と関連して戻すものを指定できます。
  - 0(ゼロ)を指定すると、出現した文字の最初の文字の位置が戻されます。これはデフォルトです。
  - 1を指定すると、出現した文字の次の文字の位置が戻されます。
- match_paramは、データ型がVARCHAR2またはCHARの文字式であり、ファンクションのデフォルトの照合動作を変更できます。このパラメータの動作は、REGEXP_COUNTのこのファンクションの動作と同じです。詳細は、「REGEXP_COUNT」を参照してください。
- 部分正規表現のあるpatternの場合、subexprは、ファンクションのターゲットとなるpattern内の部分正規表現を示す0から9までの整数になります。subexprは、カッコで囲まれたpatternのフラグメントです。部分正規表現は

ネストできます。部分正規表現は、その左側のカッコがpatternに出現する順に番号付けされます。たとえば、次の正規表現を考えます。

```
0123(((abc)(de)f)ghi)45(678)
```

この正規表現には、abcdefghi、abcdef、abc、de、678の順に5つの部分正規表現があります。

subexprが0(ゼロ)の場合、patternに一致するサブストリング全体の位置が戻されます。subexprが0(ゼロ)より大きい場合、一致したサブストリング内の部分正規表現番号subexprに対応するサブストリング・フラグメントの位置が戻されます。patternにsubexprの部分正規表現が1つもない場合、このファンクションは0(ゼロ)を戻します。NULLのsubexpr値は、NULLを戻します。subexprのデフォルト値は0(ゼロ)です。

関連項目:

5. 「INSTR」および「REGEXP_SUBSTR」を参照してください。
6. 「REGEXP_REPLACE」および「REGEXP_LIKE条件」を参照してください。
7. source_charの文字をpatternの文字と比較するためにREGEXP_INSTRで使用する照合を定義する照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、文字列を調べて、空白以外の文字を検索します。Oracleは、文字列の最初の文字から検索を開始し、空白以外の文字が6つ目に出現する開始位置(デフォルト)を戻します。

```
SELECT
  REGEXP_INSTR('500 Oracle Parkway, Redwood Shores, CA',
               '[^ ]+', 1, 6) "REGEXP_INSTR"
  FROM DUAL;
REGEXP_INSTR
------------
          37
```

次の例では、文字列を調べて、大/小文字を区別せずにs、rまたはpで始まり、任意の6つのアルファベット文字が続く単語を検索します。Oracleは、文字列の3つ目の文字から検索を開始し、大文字か小文字のs、rまたはpで始まる7文字の単語が2つ目に出現した後の文字の、文字列内の位置を戻します。

```
SELECT
  REGEXP_INSTR('500 Oracle Parkway, Redwood Shores, CA',
               '[s|r|p][[:alpha:]]{6}', 3, 2, 1, 'i') "REGEXP_INSTR"
  FROM DUAL;
REGEXP_INSTR
------------
          28
```

次の例では、subexpr引数を使用してpattern内の特定の部分正規表現を検索します。最初の文は、最初の部分正規表現にある最初の文字のソース文字列(123)の位置を戻します。

```
SELECT REGEXP_INSTR('1234567890', '(123)(4(56)(78))', 1, 1, 0, 'i', 1)
"REGEXP_INSTR" FROM DUAL;
REGEXP_INSTR
------------------
1
```

次の文は、2番目の部分正規表現にある最初の文字のソース文字列(45678)の位置を戻します。

```
SELECT REGEXP_INSTR('1234567890', '(123)(4(56)(78))', 1, 1, 0, 'i', 2)
```

```
"REGEXP_INSTR" FROM DUAL;
REGEXP_INSTR
------------------
4
```

次の文は、4番目の部分正規表現にある最初の文字のソース文字列(78)の位置を戻します。

```
SELECT REGEXP_INSTR('1234567890', '(123)(4(56)(78))', 1, 1, 0, 'i', 4)
"REGEXP_INSTR" FROM DUAL;
REGEXP_INSTR
------------------
7
```

REGEXP_INSTRのパターン一致: 例

次の文は、表regexp_tempを作成し、そこに値を挿入します。

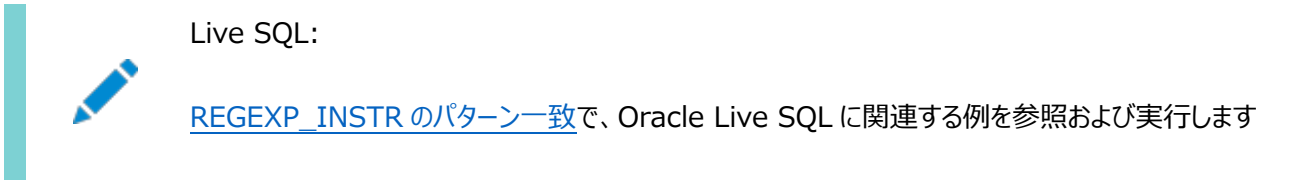
```
CREATE TABLE regexp_temp(empName varchar2(20), emailID varchar2(20));
INSERT INTO regexp_temp (empName, emailID) VALUES ('John Doe',
'johndoe@example.com');
INSERT INTO regexp_temp (empName, emailID) VALUES ('Jane Doe', 'janedoe');
```

次の例では、この文は電子メールの列を問合せ、有効な電子メール・アドレスを検索します。

```
SELECT emailID, REGEXP_INSTR(emailID, '¥w+@¥w+(¥.¥w+)+') "IS_A_VALID_EMAIL" FROM
regexp_temp;
EMAILID              IS_A_VALID_EMAIL
-------------------- ----------------
johndoe@example.com                1
example.com                        0
```

次の例では、この文は電子メールの列を問合せ、有効な電子メール・アドレスの数を戻します。

```
EMPNAME         Valid Email                    FIELD_WITH_VALID_EMAIL
--------        -------------------     ----------------------
John Doe        johndoe@example.com     1
Jane Doe
```

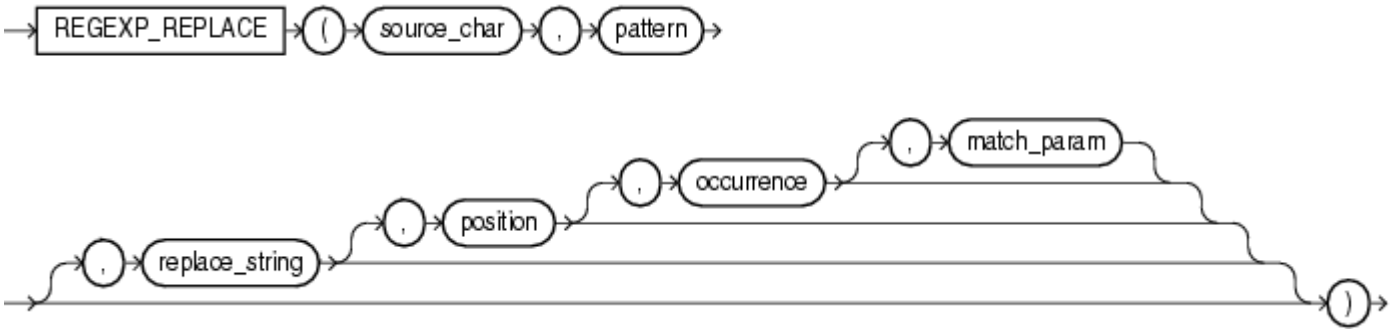


Live SQL:

REGEXP_INSTR のパターン一致で、Oracle Live SQL に関連する例を参照および実行します

REGEXP_REPLACE

構文



目的

REGEXP_REPLACEは、正規表現パターンで文字列を検索できるようにREPLACEの機能を拡張したものです。デフォルトでは、このファンクションは、正規表現パターンのすべての出現箇所を`replace_string`に置き換えて`source_char`を戻します。`source_char`と同じ文字セットの文字列が戻されます。このファンクションは、1つ目の引数がLOBではない場合はVARCHAR2を戻し、1つ目の引数がLOBの場合はCLOBを戻します。

このファンクションは、POSIX正規表現規格およびUnicode Regular Expression Guidelinesに準拠しています。詳細は、「Oracleの正規表現のサポート」を参照してください。

- `source_char`は、検索値として使用される文字式です。一般的には文字の列であり、データ型はCHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB、NCLOBのいずれかになります。
- `pattern`は正規表現です。通常はテキスト・リテラルであり、データ型はCHAR、VARCHAR2、NCHARまたはNVARCHAR2のいずれかになります。ここには、 512バイトまで入力できます。`pattern`のデータ型が`source_char`のデータ型と異なる場合、Oracleは`pattern`を`source_char`のデータ型に変換します。`pattern`で指定できる演算子のリストは、「Oracleの正規表現のサポート」を参照してください。
- `replace_string`は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBデータ型です。`replace_string`がCLOBまたはNCLOBの場合、`replace_string`は32KBに切り捨てられます。`replace_string`には、最大500個の部分正規表現への後方参照を¥nという書式で指定できます。nは、1から9の数値です。`replace_string`にバックスラッシュ(¥)を含める場合、その前にエスケープ文字(これもバックスラッシュ)を付ける必要があります。たとえば、¥2を置き換えるには、¥¥2を入力します。後方参照表現の詳細は、「Oracleの正規表現のサポート」の表D-1に示すノートを参照してください。
- `position`は、Oracleが検索を開始する文字`source_char`の位置を示す正の整数です。デフォルトは1で、`source_char`の最初の文字から検索が開始されます。
- `occurrence`は、置換操作の対象となる文字を示す、負ではない整数です。
  - 0(ゼロ)を指定すると、一致したすべての文字が置換されます。
  - 正の整数nを指定すると、n番目に一致した文字が置換されます。

  `occurrence`が1より大きい場合、1番目の`pattern`が検出された後に続く1文字目から、2番目(以降)の出現を検索します。この動作は、最初の出現の2番目の文字から2番目の出現を検索するINSTRファンクションとは異なります。
- `match_param`は、データ型VARCHAR2またはCHARの文字式で、ファンクションのデフォルトの一致動作を変更でき

ます。このパラメータの動作は、REGEXP_COUNTのこのファンクションの動作と同じです。詳細は、「REGEXP_COUNT」を参照してください。

関連項目:

3. REPLACE
4. 「REGEXP_INSTR」、「REGEXP_SUBSTR」および「REGEXP_LIKE条件」を参照してください。
5. source_charの文字をpatternの文字と比較するためにREGEXP_REPLACEで使用する照合を定義する照合決定ルール、およびこのファンクションの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、phone_numberを調べて、xxx.xxx.xxxxというパターンを検索します。その後、このパターンを(xxx) xxx-xxxxという書式に変更します。

```
SELECT
  REGEXP_REPLACE(phone_number,
                 '([[:digit:]]{3})¥.([[:digit:]]{3})¥.([[:digit:]]{4})',
                 '(¥1) ¥2-¥3') "REGEXP_REPLACE"
  FROM employees
  ORDER BY "REGEXP_REPLACE";
REGEXP_REPLACE
--------------------------------------------------------------------------------
(515) 123-4444
(515) 123-4567
(515) 123-4568
(515) 123-4569
(515) 123-5555
. . .
```

次の例では、country_nameを調べます。文字列内のNULLでない各文字の後に空白を挿入します。

```
SELECT
  REGEXP_REPLACE(country_name, '(.)', '¥1 ') "REGEXP_REPLACE"
  FROM countries;
REGEXP_REPLACE
--------------------------------------------------------------------------------
A r g e n t i n a
A u s t r a l i a
B e l g i u m
B r a z i l
C a n a d a
. . .
```

次の例では、文字列を調べて、2つ以上の空白を検索します。検索された2つ以上の空白を、それぞれ1つの空白に置換します。

```
SELECT
  REGEXP_REPLACE('500   Oracle     Parkway,    Redwood  Shores, CA',
                 '( ){2,}', ' ') "REGEXP_REPLACE"
  FROM DUAL;
REGEXP_REPLACE
-----------------------------------
500 Oracle Parkway, Redwood Shores, CA
```

REGEXP_REPLACEのパターン一致: 例

次の文は、表regexp_tempを作成し、そこに値を挿入します。

```
CREATE TABLE regexp_temp(empName varchar2(20), emailID varchar2(20));
INSERT INTO regexp_temp (empName, emailID) VALUES ('John Doe',
'johndoe@example.com');
INSERT INTO regexp_temp (empName, emailID) VALUES ('Jane Doe',
'janedoe@example.com');
```

次の文は、文字列'Jane'を'John'に置き換えます。

```
SELECT empName, REGEXP_REPLACE (empName, 'Jane', 'John') "STRING_REPLACE" FROM
regexp_temp;
EMPNAME         STRING_REPLACE
--------        --------------
John Doe        John Doe
Jane Doe        John Doe
```

次の文は、文字列'John'を'Jane'に置き換えます。

```
SELECT empName, REGEXP_REPLACE (empName, 'John', 'Jane' ) "STRING_REPLACE" FROM
regexp_temp;
EMPNAME         STRING_REPLACE
--------        --------------
John Doe        Jane Doe
Jane Doe        Jane Doe
```

REGEXP_REPLACE: 例

次の文は、文字列内のすべての数値を置換します。

```
WITH strings AS (
  SELECT 'abc123' s FROM dual union all
  SELECT '123abc' s FROM dual union all
  SELECT 'a1b2c3' s FROM dual
)
  SELECT s "STRING", regexp_replace(s, '[0-9]', '') "MODIFIED_STRING"
  FROM strings;
  STRING               MODIFIED_STRING
-------------------- --------------------
abc123               abc
123abc               abc
a1b2c3               abc
```

次の文は、文字列で最初に出現した数値を置き換えます。

```
WITH strings AS (
  SELECT 'abc123' s from DUAL union all
  SELECT '123abc' s from DUAL union all
  SELECT 'a1b2c3' s from DUAL
)
  SELECT s "STRING", REGEXP_REPLACE(s, '[0-9]', '', 1, 1) "MODIFIED_STRING"
  FROM   strings;
 STRING               MODIFIED_STRING
-------------------- --------------------
abc123               abc23
123abc               23abc
a1b2c3               ab2c3
```

次の文は、文字列で2番目に出現した数値を置き換えます。

```
WITH strings AS (
  SELECT 'abc123' s from DUAL union all
  SELECT '123abc' s from DUAL union all
  SELECT 'a1b2c3' s from DUAL
)
```

```
  SELECT s "STRING", REGEXP_REPLACE(s, '[0-9]', '', 1, 2) "MODIFIED_STRING"
  FROM   strings;
STRING               MODIFIED_STRING
-------------------- --------------------
abc123               abc13
123abc               13abc
a1b2c3               a1bc3
```

次の文は、文字列内の複数のスペースを1つのスペースに置換します。

```
WITH strings AS (
  SELECT 'Hello  World' s FROM dual union all
  SELECT 'Hello        World' s FROM dual union all
  SELECT 'Hello,   World  !' s FROM dual
)
  SELECT s "STRING", regexp_replace(s, ' {2,}', ' ') "MODIFIED_STRING"
  FROM   strings;
 STRING               MODIFIED_STRING
-------------------- --------------------
Hello  World         Hello World
Hello        World   Hello World
Hello,   World  !    Hello, World !
```

次の文は、キャメル・ケース表記の文字列を、アンダースコアで区切られた小文字を含む文字列に変換します。

```
WITH strings as (
  SELECT 'AddressLine1' s FROM dual union all
  SELECT 'ZipCode' s FROM dual union all
  SELECT 'Country' s FROM dual
)
  SELECT s "STRING",
         lower(regexp_replace(s, '([A-Z0-9])', '_¥1', 2)) "MODIFIED_STRING"
  FROM strings;
  STRING               MODIFIED_STRING
-------------------- --------------------
AddressLine1         address_line_1
ZipCode              zip_code
Country              country
```

次の文は、日付の書式を変換します。

```
WITH date_strings AS (
  SELECT  '2015-01-01' d from dual union all
  SELECT '2000-12-31' d from dual union all
  SELECT '900-01-01' d from dual
)
  SELECT d "STRING",
         regexp_replace(d, '([[:digit:]]+)-([[:digit:]]{2})-([[:digit:]]{2})',
'¥3.¥2.¥1') "MODIFIED_STRING"
  FROM date_strings;
  STRING               MODIFIED_STRING
-------------------- --------------------
2015-01-01           01.01.2015
2000-12-31           31.12.2000
900-01-01            01.01.900
```

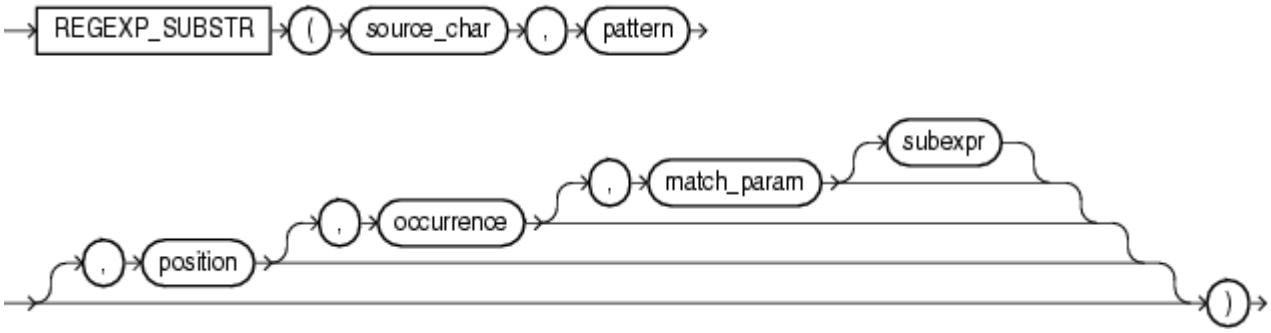
次の文は、文字列内のすべての文字を'1'に置き換えます。

```
WITH strings as (
  SELECT 'NEW YORK' s FROM dual union all
  SELECT 'New York' s FROM dual union all
  SELECT 'new york' s FROM dual
)
  SELECT s "STRING",
        regexp_replace(s, '[a-z]', '1', 1, 0, 'i') "CASE_INSENSITIVE",
```

```
       regexp_replace(s, '[a-z]', '1', 1, 0, 'c') "CASE_SENSITIVE",
       regexp_replace(s, '[a-zA-Z]', '1', 1, 0, 'c') "CASE_SENSITIVE_MATCHING"
  FROM  strings;
  STRING     CASE_INSEN CASE_SENSI CASE_SENSI
---------- ---------- ---------- ----------
NEW YORK   111 1111   NEW YORK   111 1111
New York   111 1111   N11 Y111   111 1111
new york   111 1111   111 1111   111 1111
```

REGEXP_SUBSTR

構文



目的

REGEXP_SUBSTRは、正規表現パターンで文字列を検索できるようにSUBSTRの機能を拡張したものです。REGEXP_INSTRと似ていますが、このファンクションはサブストリングの位置ではなくサブストリング自体を戻します。このファンクションは、一致文字列の内容のみが必要で、ソース文字列内での位置は必要ない場合に有効です。このファンクションは、文字列をVARCHAR2またはCLOBデータとして、source_charと同じ文字セットで戻します。

このファンクションは、POSIX正規表現規格およびUnicode Regular Expression Guidelinesに準拠しています。詳細は、「Oracleの正規表現のサポート」を参照してください。

- source_charは、検索値として使用される文字式です。通常は文字列であり、データ型はCHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBのいずれかです。
- patternは正規表現です。通常はテキスト・リテラルであり、データ型はCHAR、VARCHAR2、NCHARまたはNVARCHAR2のいずれかになります。ここには、 512バイトまで入力できます。patternのデータ型がsource_charのデータ型と異なる場合、Oracleはpatternをsource_charのデータ型に変換します。patternで指定できる演算子のリストは、「Oracleの正規表現のサポート」を参照してください。
- positionは、Oracleが検索を開始する文字source_charの位置を示す正の整数です。デフォルトは1で、source_charの最初の文字から検索が開始されます。
- occurrenceは、source_charに出現するpatternのうちのどれを検索するかを示す正の整数です。デフォルトは1で、最初に出現するpatternが検索されます。

  occurrenceが1より大きい場合、1番目のpatternが検出された後に続く1文字目から、2番目(以降)の出現を検索します。この動作は、最初の出現の2番目の文字から2番目の出現を検索するSUBSTRファンクションとは異なります。
- match_paramは、データ型VARCHAR2またはCHARの文字式で、ファンクションのデフォルトの一致動作を変更できます。このパラメータの動作は、REGEXP_COUNTのこのファンクションの動作と同じです。詳細は、「REGEXP_COUNT」を参照してください。
- 部分正規表現のあるpatternの場合、subexprは、ファンクションによって戻されるpattern内の部分正規表現を示す0から9までの負でない整数になります。このパラメータは、REGEXP_INSTRファンクションの場合と同じセマンティクスを持ちます。詳細は、「REGEXP_INSTR」を参照してください。

関連項目:

- 「SUBSTR」および「REGEXP_INSTR」を参照してください。
- 「REGEXP_REPLACE」および「REGEXP_LIKE条件」を参照してください。
- source_charの文字をpatternの文字と比較するためにREGEXP_SUBSTRで使用する照合を定義する照合決定ルール、およびこのファンクションの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、文字列を調べて、カンマで区切られた最初のサブストリングを検索します。Oracle Databaseは、後にカンマが付いているカンマでない1つ以上の文字の前にあるカンマを検索します。該当するサブストリングが、前後のカンマを含めて戻されます。

```
SELECT
  REGEXP_SUBSTR('500 Oracle Parkway, Redwood Shores, CA',
                ',[^,]+,') "REGEXPR_SUBSTR"
  FROM DUAL;
REGEXPR_SUBSTR
-----------------
, Redwood Shores,
```

次の例では、文字列を調べて、1つ以上の英数字を含むサブストリング、および任意でピリオド(.)が続くhttp://を検索します。Oracleは、http://と、スラッシュ(/)または文字列の末尾のいずれかとの間で、3つから4つのこのサブストリングを検索します。

```
SELECT
  REGEXP_SUBSTR('http://www.example.com/products',
                'http://([[:alnum:]]+¥.?){3,4}/?') "REGEXP_SUBSTR"
  FROM DUAL;
REGEXP_SUBSTR
----------------------
http://www.example.com/
```

次の2つの例では、subexpr引数を使用してpattern内の特定の部分正規表現を戻します。最初の文は、pattern内の最初の部分正規表現を戻します。

```
SELECT REGEXP_SUBSTR('1234567890', '(123)(4(56)(78))', 1, 1, 'i', 1)
"REGEXP_SUBSTR" FROM DUAL;
REGEXP_SUBSTR
-------------------
123
```

次の文は、pattern内の4番目の部分正規表現を戻します。

```
SELECT REGEXP_SUBSTR('1234567890', '(123)(4(56)(78))', 1, 1, 'i', 4)
"REGEXP_SUBSTR" FROM DUAL;
REGEXP_SUBSTR
-------------------
78
```

REGEXP_SUBSTRのパターン一致: 例

次の文は、表regexp_tempを作成し、そこに値を挿入します。

```
CREATE TABLE regexp_temp(empName varchar2(20), emailID varchar2(20));
INSERT INTO regexp_temp (empName, emailID) VALUES ('John Doe',
'johndoe@example.com');
INSERT INTO regexp_temp (empName, emailID) VALUES ('Jane Doe', 'janedoe');
```

次の例では、この文は電子メールの列を問合せ、有効な電子メール・アドレスを検索します。

```
SELECT empName, REGEXP_SUBSTR(emailID, '[[:alnum:]]+¥@[[:alnum:]]+¥.[[:alnum:]]+')
"Valid Email" FROM regexp_temp;
EMPNAME  Valid Email
-------- ------------------
John Doe johndoe@example.com
Jane Doe
```

次の例では、この文は電子メールの列を問合せ、有効な電子メール・アドレスの数を戻します。

```
SELECT empName, REGEXP_SUBSTR(emailID, '[[:alnum:]]+¥@[[:alnum:]]+¥.[[:alnum:]]+')
"Valid Email", REGEXP_INSTR(emailID, '¥w+@¥w+(¥.¥w+)+') "FIELD_WITH_VALID_EMAIL" FROM
regexp_temp;
EMPNAME         Valid Email                    FIELD_WITH_VALID_EMAIL
--------        -------------------     ----------------------
John Doe        johndoe@example.com     1
Jane Doe
```

次の例では、数字およびアルファベットが文字列から抽出されます。

```
with strings as (
  select 'ABC123' str from dual union all
  select 'A1B2C3' str from dual union all
  select '123ABC' str from dual union all
  select '1A2B3C' str from dual
)
  select regexp_substr(str, '[0-9]') First_Occurrence_of_Number,
         regexp_substr(str, '[0-9].*') Num_Followed_by_String,
         regexp_substr(str, '[A-Z][0-9]') Letter_Followed_by_String
  from strings;
FIRST_OCCURRENCE_OF_NUMB NUM_FOLLOWED_BY_STRING   LETTER_FOLLOWED_BY_STRIN
------------------------ ----------------------- ------------------------
1                        123                     C1
1                        1B2C3                   A1
1                        123ABC
1                        1A2B3C                  A2
```

次の例では、乗客名およびフライト情報が文字列から抽出されます。

```
with strings as (
  select 'LHRJFK/010315/JOHNDOE' str from dual union all
  select 'CDGLAX/050515/JANEDOE' str from dual union all
  select 'LAXCDG/220515/JOHNDOE' str from dual union all
  select 'SFOJFK/010615/JANEDOE' str from dual
)
  SELECT regexp_substr(str, '[A-Z]{6}') String_of_6_characters,
         regexp_substr(str, '[0-9]+') First_Matching_Numbers,
         regexp_substr(str, '[A-Z].*$') Letter_by_other_characters,
         regexp_substr(str, '/[A-Z].*$') Slash_letter_and_characters
  FROM strings;
STRING_OF_6_CHARACTERS  FIRST_MATCHING_NUMBERS  LETTER_BY_OTHER_CHARACTERS
SLASH_LETTER_AND_CHARACTERS
----------------------  ----------------------  -------------------------      -----
----------------------
LHRJFK                  010315                  LHRJFK/010315/JOHNDOE
/JOHNDOE
CDGLAX                  050515                  CDGLAX/050515/JANEDOE
/JANEDOE
LAXCDG                  220515                  LAXCDG/220515/JOHNDOE
/JOHNDOE
SFOJFK                  010615                  SFOJFK/010615/JANEDOE
/JANEDOE
```

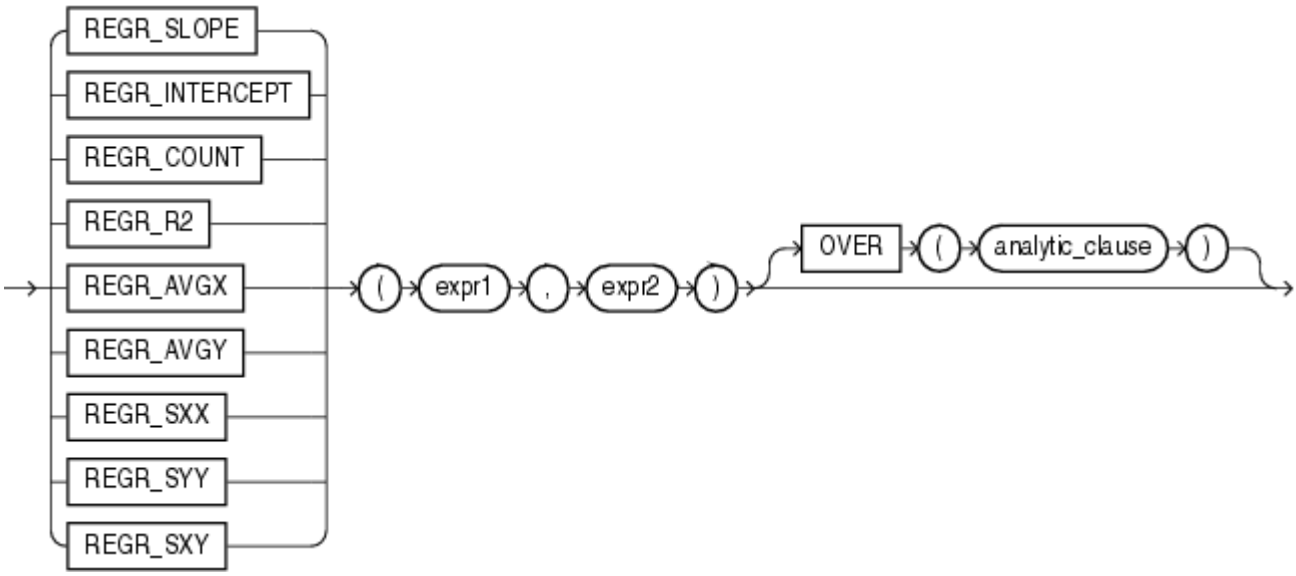
REGR_(線形回帰)ファンクション

線形回帰ファンクションは次のとおりです。

- REGR_SLOPE
- REGR_INTERCEPT
- REGR_COUNT
- REGR_R2
- REGR_AVGX
- REGR_AVGY
- REGR_SXX
- REGR_SYY
- REGR_SXY

構文

linear_regr::=



関連項目:

構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

線形回帰ファンクションは、微分最小2乗法で求めた回帰直線を数値の組の集合に対応付けます。これは、集計ファンクションまたは分析ファンクションとして使用できます。

関連項目:

exprの書式の詳細は、「集計ファンクション」および「SQL式」を参照してください。

これらのファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。Oracleは、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、その

データ型を戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。数値の優先順位の詳細は、「数値の優先順位の詳細」を参照してください。

Oracleは、expr1またはexpr2がNULLであるすべての組を排除した後、このファンクションを(expr1, expr2)の組の集合に適用します。Oracleは、データでの引渡し中、同時にすべての回帰ファンクションを計算します。

expr1は、従属変数の値(y値)として解析されます。expr2は、独立変数の値(x値)として解析されます。

- REGR_SLOPEは、直線の傾きを戻します。戻り値は数値データ型で、NULLになる場合もあります。NULL (expr1, expr2)の組を排除した後、このファンクションは次の計算を行います。

```
COVAR_POP(expr1, expr2) / VAR_POP(expr2)
```

- REGR_INTERCEPTは、回帰直線のy切片を戻します。戻り値は数値データ型で、NULLになる場合もあります。NULL (expr1, expr2)の組を排除した後、このファンクションは次の計算を行います。

```
AVG(expr1) - REGR_SLOPE(expr1, expr2) * AVG(expr2)
```

- REGR_COUNTは整数を戻します。この整数は、回帰直線に適応させるために使用されるNULLでない数値の組です。
- REGR_R2は、回帰に対する決定係数(Rの2乗または適合度とも呼ぶ)を戻します。戻り値は数値データ型で、NULLになる場合もあります。VAR_POP (expr1)およびVAR_POP (expr2)は、NULLの組が排除された後に評価されます。戻り値は次のとおりです。

```
                     NULL if VAR_POP(expr2)  = 0
                        1 if VAR_POP(expr1)  = 0 and
                             VAR_POP(expr2) != 0
POWER(CORR(expr1,expr),2) if VAR_POP(expr1)  > 0 and
                             VAR_POP(expr2  != 0
```

これ以外のすべての回帰ファンクションの戻り値は数値データ型で、NULLになる場合もあります。

- REGR_AVGXは、回帰直線の独立変数(expr2)の平均を求めます。NULL(expr1, expr2)の組を排除した後、このファンクションは次の計算を行います。

```
AVG(expr2)
```

- REGR_AVGYは、回帰直線の従属変数(expr1)の平均を求めます。NULL(expr1, expr2)の組を排除した後、このファンクションは次の計算を行います。

```
AVG(expr1)
```

REGR_SXY、REGR_SXX、REGR_SYYは補助ファンクションです。これらは、様々な診断統計の計算に使用されます。

- NULL (expr1, expr2)の組を排除した後、REGR_SXXは次の計算を行います。

```
REGR_COUNT(expr1, expr2) * VAR_POP(expr2)
```

- NULL (expr1, expr2)の組を排除した後、REGR_SYYは次の計算を行います。

```
REGR_COUNT(expr1, expr2) * VAR_POP(expr1)
```

- NULL (expr1, expr2)の組を排除した後、REGR_SXYは次の計算を行います。

```
REGR_COUNT(expr1, expr2) * COVAR_POP(expr1, expr2)
```

次の例は、サンプル表sh.salesおよびsh.productsに基づいています。

一般的な線形回帰の例

次の例では、分析書式で使用される様々な線形回帰ファンクションを比較します。これらのファンクションの分析書式は、モデル別で従業員ごとに予測される給与の検索などの計算で、回帰統計を使用する場合に有効です。次の個々の線形回帰ファンクションについての項では、これらのファンクションの集計書式の例を示します。

```
SELECT job_id, employee_id ID, salary,
REGR_SLOPE(SYSDATE-hire_date, salary)
   OVER (PARTITION BY job_id) slope,
REGR_INTERCEPT(SYSDATE-hire_date, salary)
   OVER (PARTITION BY job_id) intcpt,
REGR_R2(SYSDATE-hire_date, salary)
   OVER (PARTITION BY job_id) rsqr,
REGR_COUNT(SYSDATE-hire_date, salary)
   OVER (PARTITION BY job_id) count,
REGR_AVGX(SYSDATE-hire_date, salary)
   OVER (PARTITION BY job_id) avgx,
REGR_AVGY(SYSDATE-hire_date, salary)
   OVER (PARTITION BY job_id) avgy
   FROM employees
   WHERE department_id in (50, 80)
   ORDER BY job_id, employee_id;
JOB_ID        ID     SALARY SLOPE    INTCPT  RSQR  COUNT       AVGX      AVGY
---------- ----- ---------- ----- --------- ----- ------ ---------- ---------
SA_MAN       145      14000  .355 -1707.035  .832      5  12200.000  2626.589
SA_MAN       146      13500  .355 -1707.035  .832      5  12200.000  2626.589
SA_MAN       147      12000  .355 -1707.035  .832      5  12200.000  2626.589
SA_MAN       148      11000  .355 -1707.035  .832      5  12200.000  2626.589
SA_MAN       149      10500  .355 -1707.035  .832      5  12200.000  2626.589
SA_REP       150      10000  .257   404.763  .647     29   8396.552  2561.244
SA_REP       151       9500  .257   404.763  .647     29   8396.552  2561.244
SA_REP       152       9000  .257   404.763  .647     29   8396.552  2561.244
SA_REP       153       8000  .257   404.763  .647     29   8396.552  2561.244
SA_REP       154       7500  .257   404.763  .647     29   8396.552  2561.244
SA_REP       155       7000  .257   404.763  .647     29   8396.552  2561.244
SA_REP       156      10000  .257   404.763  .647     29   8396.552  2561.244
. . .
```

REGR_SLOPEおよびREGR_INTERCEPTの例

次の例では、サンプル表hr.employeesを使用して、勤務時間(SYSDATE - hire_date)と給与に関する線形回帰モデルの傾きと回帰を計算します。結果はjob_id別にグループ化されます。

```
SELECT job_id,
REGR_SLOPE(SYSDATE-hire_date, salary) slope,
REGR_INTERCEPT(SYSDATE-hire_date, salary) intercept
   FROM employees
   WHERE department_id in (50,80)
   GROUP BY job_id
   ORDER BY job_id;
JOB_ID     SLOPE    INTERCEPT
---------- ----- ------------
SA_MAN      .355 -1707.030762
SA_REP      .257   404.767151
SH_CLERK    .745   159.015293
ST_CLERK    .904   134.409050
ST_MAN      .479  -570.077291
```

REGR_COUNTの例

次の例では、サンプル表hr.employeesを使用して、勤務時間(SYSDATE - hire_date)と給与について、job_id別の数を計算します。結果はjob_id別にグループ化されます。

```
SELECT job_id,
REGR_COUNT(SYSDATE-hire_date, salary) count
   FROM employees
   WHERE department_id in (30, 50)
   GROUP BY job_id
   ORDER BY job_id, count;
JOB_ID          COUNT
---------- ----------
PU_CLERK             5
PU_MAN               1
SH_CLERK            20
ST_CLERK            20
ST_MAN               5
```

REGR_R2の例

次の例では、サンプル表hr.employeesを使用して、勤務時間(SYSDATE - hire_date)と給与の線形回帰に対する決定係数を計算します。

```
SELECT job_id,
REGR_R2(SYSDATE-hire_date, salary) Regr_R2
   FROM employees
   WHERE department_id in (80, 50)
   GROUP by job_id
  ORDER BY job_id, Regr_R2;
JOB_ID        REGR_R2
---------- ----------
SA_MAN       .83244748
SA_REP      .647007156
SH_CLERK    .879799698
ST_CLERK    .742808493
ST_MAN       .69418508
```

REGR_AVGYおよびREGR_AVGXの例

次の例では、サンプル表hr.employeesを使用して、勤務時間(SYSDATE - hire_date)と給与の平均値を計算します。結果はjob_id別にグループ化されます。

```
SELECT job_id,
REGR_AVGY(SYSDATE-hire_date, salary) avgy,
REGR_AVGX(SYSDATE-hire_date, salary) avgx
   FROM employees
   WHERE department_id in (30,50)
   GROUP BY job_id
   ORDER BY job_id, avgy, avgx;
JOB_ID           AVGY       AVGX
---------- ---------- ----------
PU_CLERK    2950.3778       2780
PU_MAN      4026.5778      11000
SH_CLERK    2773.0778       3215
ST_CLERK    2872.7278       2785
ST_MAN      3140.1778       7280
```

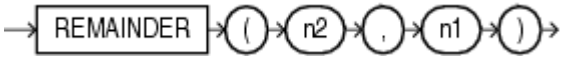
REGR_SXY、REGR_SXXおよびREGR_SYYの例

次の例では、サンプル表hr.employeesを使用して、勤務時間(SYSDATE - hire_date)と給与の線形回帰について、3種類の診断統計を計算します。

```
SELECT job_id,
REGR_SXY(SYSDATE-hire_date, salary) regr_sxy,
REGR_SXX(SYSDATE-hire_date, salary) regr_sxx,
REGR_SYY(SYSDATE-hire_date, salary) regr_syy
   FROM employees
   WHERE department_id in (80, 50)
   GROUP BY job_id
   ORDER BY job_id;

JOB_ID       REGR_SXY    REGR_SXX   REGR_SYY
---------- ---------- ----------- ----------
SA_MAN        3303500   9300000.0    1409642
SA_REP     16819665.5  65489655.2 6676562.55
SH_CLERK      4248650   5705500.0    3596039
ST_CLERK      3531545   3905500.0 4299084.55
ST_MAN        2180460   4548000.0  1505915.2
```

REMAINDER

構文



目的

REMAINDERはn2をn1で割った余りを戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。Oracleは、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

MODファンクションはREMAINDERと似ていますが、MODの式ではFLOORを使用します。これに対してREMAINDERではROUNDを使用します。詳細は、「MOD」を参照してください。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。数値の優先順位の詳細は、「数値の優先順位の詳細」を参照してください。

- n1が0(ゼロ)であるか、またはn2が無限大の場合、Oracleが返す内容は次のとおりです。
  - 両方の引数がNUMBER型の場合はエラー
  - 両方の引数がBINARY_FLOATまたはBINARY_DOUBLEの場合はNaN
- n1が0ではない場合、余りはn2 - (n1*N)となります。Nは、n2/n1に最も近い整数です。n2/n1がx.5と等しい場合、Nは最も近い偶数の整数です。
- n2が浮動小数点数で、余りが0(ゼロ)の場合、余りの符号はn2の符合になります。NUMBER値の場合、0の余りには符号は付きません。

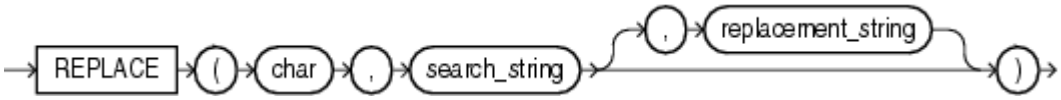
例

次の例では、TO_BINARY_DOUBLEの「例」で作成されたfloat_point_demo表を使用して、2つの浮動小数点数を割り、算出された余りを戻します。

```
SELECT bin_float, bin_double, REMAINDER(bin_float, bin_double)
  FROM float_point_demo;
 BIN_FLOAT BIN_DOUBLE REMAINDER(BIN_FLOAT,BIN_DOUBLE)
---------- ---------- ------------------------------
1.235E+003 1.235E+003                      5.859E-005
```

REPLACE

構文



目的

REPLACEは、replacement_stringでsearch_stringのすべての出現箇所を変換してcharを戻します。replacement_stringを指定しない場合またはNULLの場合、すべてのsearch_stringが削除されます。search_stringがNULLの場合、charが戻されます。

charと同様に、search_stringおよびreplacement_stringは、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBデータ型です。charと同じ文字セットの文字列が戻されます。このファンクションは、1つ目の引数がLOBではない場合はVARCHAR2を戻し、1つ目の引数がLOBの場合はCLOBを戻します。

REPLACEは、TRANSLATEファンクションに関連する機能を提供します。TRANSLATEは、単一文字を1対1で置き換えます。REPLACEファンクションでは、1つの文字列の置換および複数の文字列の削除を実行できます。

関連項目:

- TRANSLATE
- charの文字をsearch_stringの文字と比較するためにREPLACEで使用する照合を定義する照合決定ルール、およびこのファンクションの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

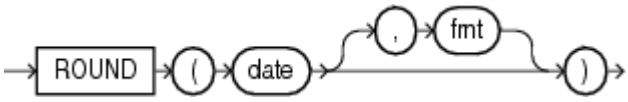
次の例では、JをBLに置換します。

```
SELECT REPLACE('JACK and JUE','J','BL') "Changes"
     FROM DUAL;
Changes
--------------
BLACK and BLUE
```

ROUND (日付)

構文

round_date::=



目的

ROUNDは、dateを書式モデルfmtで指定した単位に丸めた結果を戻します。このファンクションは、NLS_CALENDARセッション・パラメータの影響を受けません。このファンクションはグレゴリオ暦の規則に従って動作します。戻される値は、dateに異なる日時データ型を指定した場合でも、常にDATEデータ型です。fmtを省略すると、dateは最も近い日に丸められます。date式は、DATE値に変換される必要があります。

関連項目:

fmtで使用できる書式モデルは、「ROUNDおよびTRUNC日付ファンクション」を参照してください。

例

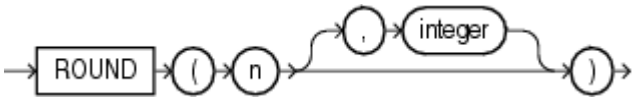
次の例では、次の年の最も近い日に丸めます。

```
SELECT ROUND (TO_DATE ('27-OCT-00'),'YEAR')
   "New Year" FROM DUAL;

New Year
---------
01-JAN-01
```

ROUND (数値)

構文

round_number::=



目的

ROUNDは、nを小数点以下integer桁に丸めた値を戻します。integerを指定しない場合、nは小数点以下が丸められます。integerが負の場合、nは小数点の左側に丸められます。

nには、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を指定できます。integerを指定しない場合、このファンクションは、nの数値データ型と同じデータ型で値ROUND(n, 0)を戻します。integerを指定すると、このファンクションはNUMBERを戻します。

ROUNDは、次の規則を使用して実行されます。

- nが0の場合、ROUNDはintegerに関係なく常に0を戻します。
- nが負の場合、ROUND(n, integer)は、-ROUND(-n, integer)を戻します。
- nが正の場合は、次のとおりです。

```
ROUND(n, integer) = FLOOR(n * POWER(10, integer) + 0.5) * POWER(10, -integer)
```

NUMBER値に適用されたROUNDが、浮動小数点で表現された同じ値に適用されたROUNDと少し異なる場合があります。この結果の相違は、NUMBERと浮動小数点値の内部表現の違いから発生します。相違が発生した場合、その相違は丸められた桁で1になります。

関連項目:

- 暗黙的な変換の詳細は、表2-8を参照してください。
- Oracle DatabaseによるBINARY_FLOAT値およびBINARY_DOUBLE値の処理方法の詳細は、「浮動小数点数」を参照してください。
- 関連する操作を実行するファンクションの詳細は、「FLOOR」、「CEIL」、「TRUNC(数値)」および「MOD」を参照してください。

例

次の例では、数値を小数点以下1桁に丸めます。

```
SELECT ROUND(15.193,1) "Round" FROM DUAL;
     Round
----------
      15.2
```

次の例では、数値の小数点の左1桁を丸めます。

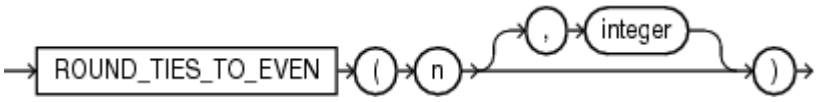
```
SELECT ROUND(15.193,-1) "Round" FROM DUAL;
     Round
----------
```

```
20
```

ROUND_TIES_TO_EVEN (数値)

構文

round_ties_to_even ::=



目的

ROUND_TIES_TO_EVENは2つのパラメータ(nおよびinteger)をとる丸めファンクションです。このファンクションは、次のルールに従ってinteger桁に丸めたnを返します。

- integerが正の場合、nは小数点以下integer桁に丸められます。
- integerが指定されていない場合、nは0桁に丸められます。
- integerが負の場合、nは小数点の左側integer桁に丸められます。

制限事項

このファンクションでは、BINARY_FLOAT型およびBINARY_DOUBLE型はサポートされていません。

例

次の例では、数値を小数点以下1桁に丸めます。

```
SELECT ROUND_TIES_TO_EVEN (0.05, 1) from DUAL
ROUND_TIES_TO_EVEN(0.05,1)
--------------------------
                         0
```

次の例では、数値を整数部の1桁に丸めます。

```
SELECT ROUND_TIES_TO_EVEN(45.177,-1) "ROUND_EVEN" FROM DUAL;
ROUND_TIES_TO_EVEN(45.177,-1)
-----------------------------
                                                50
```

ROW_NUMBER

構文

関連項目:

構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

ROW_NUMBERは分析ファンクションです。このファンクションは、order_by_clauseに指定された行の、1から始まる順番順序で、このファンクションが適用される各行(パーティションの各行、または問合せが戻す各行)に一意の数値を割り当てます。

指定された範囲のROW_NUMBER値を取得する問合せ内に、ROW_NUMBERを使用した副問合せをネストすることで、内側の問合せの結果から正確な行のサブセットを得ることができます。この方法でこのファンクションを使用すると、上位N番、下位N番および内部N番のレポートを実行できます。一貫した結果を戻すには、問合せで決定的なソート順序を使用する必要があります。

例

次の例では、hr.employees表で各部門の最も支給額の高い上位3人の従業員を検索します。従業員が3人未満の部門については、3行未満の行が戻されます。

```
SELECT department_id, first_name, last_name, salary
FROM
(
  SELECT
    department_id, first_name, last_name, salary,
    ROW_NUMBER() OVER (PARTITION BY department_id ORDER BY salary desc) rn
  FROM employees
)
WHERE rn <= 3
ORDER BY department_id, salary DESC, last_name;
```

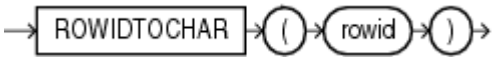
次の例は、sh.sales表の結合問合せです。1999年の売上高が上位5つの製品について2000年の売上高を検索し、2000年と1999年の差を比較します。各販売チャネルにおいて売上高上位の10製品が計算されます。

```
SELECT sales_2000.channel_desc, sales_2000.prod_name,
       sales_2000.amt amt_2000,  top_5_prods_1999_year.amt amt_1999,
       sales_2000.amt  - top_5_prods_1999_year.amt amt_diff
FROM
/* The first subquery finds the 5 top-selling products per channel in year 1999. */
  (SELECT channel_desc, prod_name, amt
   FROM
   (
     SELECT channel_desc, prod_name, sum(amount_sold) amt,
       ROW_NUMBER () OVER (PARTITION BY channel_desc
                           ORDER BY SUM(amount_sold) DESC) rn
     FROM sales, times, channels, products
     WHERE sales.time_id = times.time_id
       AND times.calendar_year = 1999
       AND channels.channel_id = sales.channel_id
       AND products.prod_id = sales.prod_id
     GROUP BY channel_desc, prod_name
```

```
   )
   WHERE rn <= 5
  ) top_5_prods_1999_year,
/* The next subquery finds sales per product and per channel in 2000. */
  (SELECT channel_desc, prod_name, sum(amount_sold) amt
     FROM sales, times, channels, products
     WHERE sales.time_id = times.time_id
       AND times.calendar_year = 2000
       AND channels.channel_id = sales.channel_id
       AND products.prod_id = sales.prod_id
     GROUP BY channel_desc, prod_name
  ) sales_2000
WHERE sales_2000.channel_desc = top_5_prods_1999_year.channel_desc
  AND sales_2000.prod_name = top_5_prods_1999_year.prod_name
ORDER BY sales_2000.channel_desc, sales_2000.prod_name
;
CHANNEL_DESC    PROD_NAME                                          AMT_2000
AMT_1999   AMT_DIFF
--------------- --------------==------------------------------ ---------- ---------
- ----------
Direct Sales     17" LCD w/built-in HDTV Tuner                     628855.7
1163645.78 -534790.08
Direct Sales     Envoy 256MB - 40GB                               502938.54
843377.88 -340439.34
Direct Sales     Envoy Ambassador                                2259566.96
1770349.25  489217.71
Direct Sales     Home Theatre Package with DVD-Audio/Video Play  1235674.15
1260791.44  -25117.29
Direct Sales     Mini DV Camcorder with 3.5" Swivel LCD           775851.87
1326302.51 -550450.64
Internet         17" LCD w/built-in HDTV Tuner                     31707.48
160974.7 -129267.22
Internet         8.3 Minitower Speaker                            404090.32
155235.25  248855.07
Internet         Envoy 256MB - 40GB                                28293.87
154072.02 -125778.15
Internet         Home Theatre Package with DVD-Audio/Video Play   155405.54
153175.04     2230.5
Internet         Mini DV Camcorder with 3.5" Swivel LCD            39726.23
189921.97 -150195.74
Partners         17" LCD w/built-in HDTV Tuner                    269973.97
325504.75  -55530.78
Partners         Envoy Ambassador                                1213063.59
614857.93  598205.66
Partners         Home Theatre Package with DVD-Audio/Video Play   700266.58
520166.26  180100.32
Partners         Mini DV Camcorder with 3.5" Swivel LCD           404265.85
520544.11 -116278.26
Partners         Unix/Windows 1-user pack                         374002.51
340123.02   33879.49
15 rows selected.
```

ROWIDTOCHAR

構文



目的

ROWIDTOCHARは、ROWIDの値をVARCHAR2データ型に変換します。この変換の結果は常に18文字です。

関連項目:

ROWIDTOCHARの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、employees表のROWID値を文字値に変換します。(結果はサンプル・データベースのビルドごとに異なります。)

```
SELECT ROWID FROM employees
   WHERE ROWIDTOCHAR(ROWID) LIKE '%JAAB%'
   ORDER BY ROWID;
ROWID
------------------
AAAFfIAAFAAAABSAAb
```

ROWIDTONCHAR

構文

目的

ROWIDTONCHARは、ROWID値をNVARCHAR2データ型に変換します。この変換の結果は、常に、各国語文字セットの文字で、18文字です。

関連項目:

ROWIDTONCHARの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

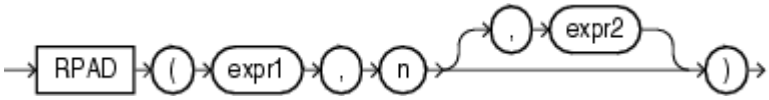
例

次の例では、ROWID値をNVARCHAR2文字列に変換します。

```
SELECT LENGTHB( ROWIDTONCHAR(ROWID) ) Length, ROWIDTONCHAR(ROWID)
   FROM employees
   ORDER BY length;
    LENGTH ROWIDTONCHAR(ROWID
---------- ------------------
        36 AAAL52AAFAAAABSABD
        36 AAAL52AAFAAAABSABV
. . .
```

RPAD

構文



目的

RPADは、expr1の右にexpr2で指定した文字を必要に応じて連続的に埋め込み、長さnにして戻します。このファンクションは、問合せの出力の書式設定に役立ちます。

expr1およびexpr2は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBデータ型です。戻される文字列は、expr1が文字データ型の場合はVARCHAR2データ型、expr1が各国語キャラクタ・データ型の場合はNVARCHAR2データ型、expr1がLOBデータ型の場合はLOBデータ型になります。expr1と同じ文字セットの文字列が戻されます。引数nは、NUMBER型の整数か、またはNUMBER型の整数に暗黙的に変換可能な値である必要があります。

expr1はNULL以外である必要があります。expr2を指定しない場合、デフォルトで空白1個が指定されます。expr1がnより長い場合、このファンクションはnに収まるexpr1の一部を戻します。

引数nは、戻り値が画面に表示される場合の全体の長さです。多くの文字セットでは、これは戻り値の文字数でもあります。ただし、マルチバイトの文字セットでは、表示される文字列の長さが文字列の文字数と異なる場合もあります。

関連項目:

RPADの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

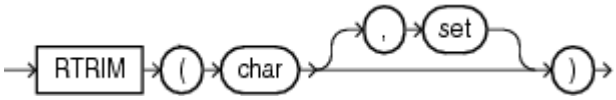
例

次の例では、空白にアスタリスクを埋め込んで、給与額の単純なチャートを作成します。

```
SELECT last_name, RPAD(' ', salary/1000/1, '*') "Salary"
   FROM employees
   WHERE department_id = 80
   ORDER BY last_name, "Salary";
LAST_NAME                 Salary
------------------------- ---------------
Abel                       **********
Ande                       *****
Banda                      *****
Bates                      ******
Bernstein                  ********
Bloom                      *********
Cambrault                  **********
Cambrault                  ******
Doran                      ******
Errazuriz                  ***********
Fox                        ********
Greene                     ********
Hall                       ********
Hutton                     *******
Johnson                    *****
King                       *********
. . .
```

RTRIM

構文



目的

RTRIMは、charの右端から、setに指定されたすべての文字を削除します。このファンクションは、問合せの出力の書式設定に役立ちます。

setを指定しない場合、デフォルトで空白1個が指定されます。RTRIMはLTRIMと同様の働きをします。

charおよびsetは、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBデータ型です。戻される文字列は、charが文字データ型の場合はVARCHAR2データ型、charが各国語キャラクタ・データ型の場合はNVARCHAR2データ型、charがLOBデータ型の場合はLOBデータ型になります。

関連項目:

- LTRIM
- setからの文字をcharからの文字と比較するためにRTRIMで使用する照合を定義する照合決定ルール、およびこのファンクションの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、文字列の右端にあるすべての小なり記号(<)、大なり記号(>)および等号(=)を文字列から削除します。

```
SELECT RTRIM('<=====>BROWNING<=====>', '<>=') "RTRIM Example"
  FROM DUAL;
RTRIM Example
---------------
<=====>BROWNING
```

SCN_TO_TIMESTAMP

構文

目的

SCN_TO_TIMESTAMPは、引数として、システム変更番号(SCN)と評価される数値を取り、そのSCNに関連付けられた概数のタイムスタンプを戻します。戻り値のデータ型はTIMESTAMPです。このファンクションは、SCNに関連付けられたタイムスタンプを調べる場合に有効です。たとえば、このファンクションをORA_ROWSCN疑似列とともに使用して、行への最新の変更にタイムスタンプを関連付けることができます。

ノート:

- 結果値の通常の精度は 3 秒です。
- SCN と SCN 生成時のタイムスタンプの関連は、一定期間データベースで記憶されます。この期間は、最大で自動調整された UNDO 保存期間(データベースが自動 UNDO 管理モードで実行されている場合)およびデータベース内のすべてのフラッシュバック・アーカイブの保存時間となりますが、120 時間以上になります。関連が不要となるまでの経過時間には、データベースが開かれているときの時間のみが加算されます。SCN_TO_TIMESTAMP の引数に対して指定された SCN が古すぎる場合は、エラーが戻されます。

関連項目:

「ORA_ROWSCN疑似列」および「TIMESTAMP_TO_SCN」を参照してください。

例

次の例では、ORA_ROWSCN疑似列を使用して行への最新の変更のシステム変更番号を判断し、SCN_TO_TIMESTAMPを使用してそのSCNをタイムスタンプに変換します。

```
SELECT SCN_TO_TIMESTAMP(ORA_ROWSCN) FROM employees
   WHERE employee_id = 188;
```

このような問合せを使用して、Oracleフラッシュバック問合せで使用するためにシステム変更番号をタイムスタンプに変換することもできます。

```
SELECT salary FROM employees WHERE employee_id = 188;
    SALARY
----------
      3800
UPDATE employees SET salary = salary*10 WHERE employee_id = 188;
COMMIT;
SELECT salary FROM employees WHERE employee_id = 188;
    SALARY
----------
     38000

SELECT SCN_TO_TIMESTAMP(ORA_ROWSCN) FROM employees
   WHERE employee_id = 188;
```

```
SCN_TO_TIMESTAMP(ORA_ROWSCN)
---------------------------------------------------------------------------
28-AUG-03 01.58.01.000000000 PM
FLASHBACK TABLE employees TO TIMESTAMP
   TO_TIMESTAMP('28-AUG-03 01.00.00.000000000 PM');
SELECT salary FROM employees WHERE employee_id = 188;
    SALARY
----------
      3800
```

SESSIONTIMEZONE

構文

目的

SESSIONTIMEZONEは、現行セッションのタイムゾーンを返します。戻り型は、タイムゾーン・オフセット('[+|-]TZH:TZM'という書式の文字列型)またはタイムゾーン地域名です。これは、最近のALTER SESSION文でユーザーが指定したセッション・タイムゾーンの値によって異なります。

ノート:

クライアントのオペレーティング・システムで特定のタイムゾーンを使用している場合でも、オフセットにはクライアント・セッションのデフォルトのタイムゾーンが使用されます。特定のタイムゾーンをデフォルトのセッション・タイムゾーンとして使用する場合は、クライアント環境の ORA_SDTZ 変数を Oracle のタイムゾーン地域名に設定します。この変数の詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

関連項目:

SESSIONTIMEZONEの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、現行のセッションのタイムゾーンを戻します。

```
SELECT SESSIONTIMEZONE FROM DUAL;
SESSION
-------
-08:00
```

SET

構文

目的

SETは、重複を排除することによってネストした表を単一の集合に変換します。このファンクションは、各要素が一意であるネストした表を戻します。戻されるネストした表のデータ型は、入力されたネストした表と同じです。

ネストした表の要素の型は、比較可能な型である必要があります。スカラー型以外の型の比較の可能性の詳細は、「比較条件」を参照してください。

例

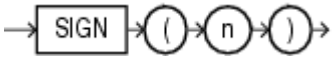
次の例では、customers_demo表から、ネストした表列cust_address_ntabの一意の要素を選択します。

```
SELECT customer_id, SET(cust_address_ntab) address
  FROM customers_demo
  ORDER BY customer_id;

CUSTOMER_ID ADDRESS(STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE, COUNTRY_ID)
----------- --------------------------------------------------------------------
        101 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('514 W Superior St', '46901',
'Kokomo', 'IN', 'US'))
        102 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('2515 Bloyd Ave', '46218',
'Indianapolis', 'IN', 'US'))
        103 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('8768 N State Rd 37', '47404',
'Bloomington', 'IN', 'US'))
        104 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('6445 Bay Harbor Ln', '46254',
'Indianapolis', 'IN', 'US'))
        105 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('4019 W 3Rd St', '47404',
'Bloomington', 'IN', 'US'))
. . .
```

この例では、表customers_demoと、データを含むネストした表の列が1つ必要です。この表およびネストした表の列を作成する方法は、「MULTISET演算子」を参照してください。

SIGN

構文



目的

SIGNは、nの符号を戻します。このファンクションは、引数として、任意の数値データ型、または暗黙的にNUMBERに変換可能な数値以外のデータ型を取り、NUMBERを戻します。

NUMBER型の値の場合、符号は次のとおりです。

- n<0の場合は-1
- n=0の場合は0
- n>0の場合は1

浮動小数点数(BINARY_FLOATおよびBINARY_DOUBLE)の場合、このファンクションは数値の符号ビットを戻します。符号ビットは次のとおりです。

- n<0の場合は-1
- n>=0またはn=NaNの場合、+1

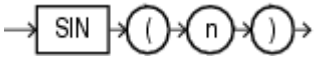
例

次の例では、ファンクションの引数(-15)が0より小さいことを示します。

```
SELECT SIGN(-15) "Sign" FROM DUAL;
      Sign
----------
        -1
```

SIN

構文



目的

SINは、n(ラジアンで表された角度)のサインを戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数がBINARY_FLOATの場合、このファンクションはBINARY_DOUBLEを戻します。それ以外の場合、引数と同じ数値データ型を戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

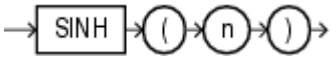
例

次の例では、30度のサインを戻します。

```
SELECT SIN(30 * 3.14159265359/180)
 "Sine of 30 degrees" FROM DUAL;
Sine of 30 degrees
------------------
                .5
```

SINH

構文



目的

SINHは、nの双曲線サインを戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数がBINARY_FLOATの場合、このファンクションはBINARY_DOUBLEを戻します。それ以外の場合、引数と同じ数値データ型を戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

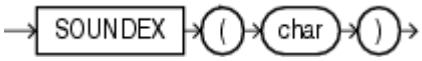
例

次の例では、1の双曲線サインを戻します。

```
SELECT SINH(1) "Hyperbolic sine of 1" FROM DUAL;
Hyperbolic sine of 1
--------------------
          1.17520119
```

SOUNDEX

構文



目的

SOUNDEXは、charと同じ音声表現を持つ文字列を戻します。このファンクションによって、綴りが異なっても発音が類似した英単語を比較できます。

音声表現については、『The Art of Computer Programming, Volume 3: Sorting and Searching』(Donald E. Knuth著)Knuthの『The Art of Computer Programming, Volume 3: Sorting and Searching』で次のように定義されています。

- 文字列の最初の文字を残し、a、e、h、i、o、u、w、yの文字が出てきた場合にはすべて削除します。
- 残った2文字目以降の文字に対し、次のように数値を割り当てます。

```
b, f, p, v = 1
c, g, j, k, q, s, x, z = 2
d, t = 3
l = 4
m, n = 5
r = 6
```

- 元の名前(最初のステップを行う前)で、同じ数値を持つ2つ以上の文字が並んでいるか、または挟まれているhとwを除外すると同じ数値を持つ2つ以上の文字が並んでしまう場合は、最初の文字を残してそれ以外の並んでいる同じ数値の文字をすべて削除します。
- 最初の4バイトを0(ゼロ)で埋めて戻します。

charは、CHAR、VARCHAR2、NCHARまたはNVARCHAR2データ型です。戻り値は、charと同じデータ型です。

このファンクションは、CLOBデータを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用してCLOBを引数として渡すことはできます。

関連項目:

- 詳細は、「データ型の比較規則」を参照してください。
- SOUNDEXの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、「Smyth」と同じ音声表現を持つ従業員を戻します。

```
SELECT last_name, first_name
     FROM hr.employees
     WHERE SOUNDEX(last_name)
         = SOUNDEX('SMYTHE')
     ORDER BY last_name, first_name;
LAST_NAME  FIRST_NAME
---------- ----------
Smith      Lindsey
```

```
Smith       William
```

SQRT

構文

目的

SQRTは、nの平方根を戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を返します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

- nがNUMBERになる場合、値nは負にはなりません。SQRTは実数を戻します。
- nが2進浮動小数点数(BINARY_FLOATまたはBINARY_DOUBLE)になる場合、結果は次のとおりです。
    - n >= 0の場合、結果は正。
    - n = -0の場合、結果は-0。
    - n < 0の場合、結果はNaN。

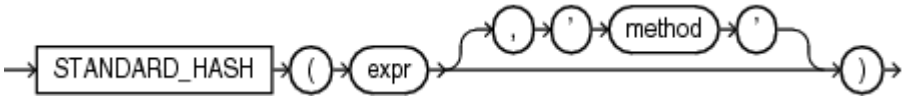
例

次の例では、26の平方根を戻します。

```
SELECT SQRT(26) "Square root" FROM DUAL;
Square root
-----------
5.09901951
```

STANDARD_HASH

構文



目的

STANDARD_HASHは、複数のハッシュ・アルゴリズムのいずれかを使用して、特定の式のハッシュ値を計算します。このハッシュ・アルゴリズムは、米国標準技術局で定義され、標準化されています。このファンクションは、セキュリティ・アプリケーション(デジタル署名、チェックサム、指紋認証など)で認証を実行して、データ整合性を維持するために役立ちます。

STANDARD_HASHファンクションを使用すると、拡張データ型の列に索引を作成できます。詳細は、「拡張データ型の列に対する索引の作成」を参照してください。

- expr引数には、Oracle Databaseでハッシュ値を計算するデータを指定します。exprに指定できるデータの長さに制限はありません。このデータは通常、列名です。exprは、LONG型またはLOB型にすることはできません。ユーザー定義のオブジェクト型にすることはできません。その他のデータ型はすべて、exprでサポートされています。
- オプションのmethod引数を使用すると、どのハッシュ・アルゴリズムを使用するかを指定できます。有効なアルゴリズムは、SHA1、SHA256、SHA384、SHA512、およびMD5です。この引数を省略すると、SHA1が使用されます。

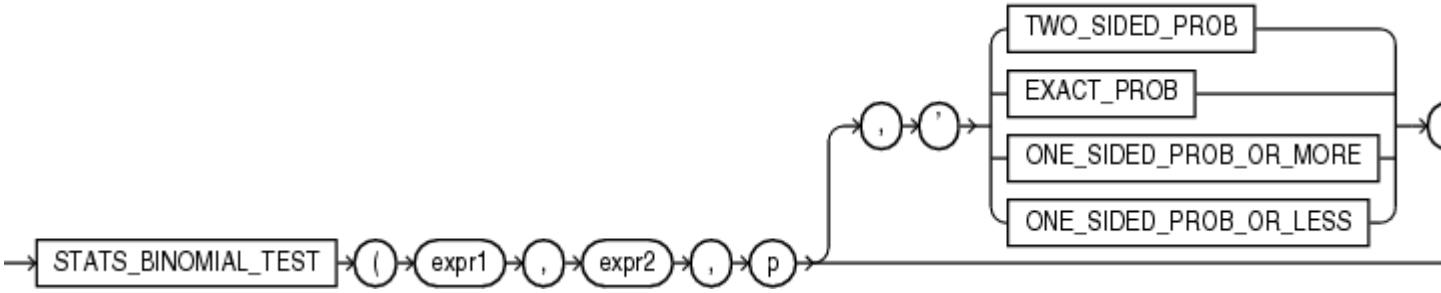
このファンクションは、RAW値を返します。

ノート:

STANDARD_HASH ファンクションは、Oracle Database の内部でハッシュ・パーティション化のために使用されるものと同じではありません。

STATS_BINOMIAL_TEST

構文



目的

STATS_BINOMIAL_TESTは、有効な値が2つのみである二値変数に使用する直接確立法です。このファンクションは、標本の割合と指定された割合との差をテストします。このテストでは通常、小さいサイズの標本が使用されます。

このファンクションは3つの必須の引数を取ります。expr1はテスト対象の標本、expr2は割合を求める値、pはテストの基準となる割合です。オプションの4つ目の引数を使用すると、表7-3に示すように、このファンクションによって戻されるNUMBER値の意味を指定できます。この引数には、テキスト・リテラル、バインド変数または定数の文字値に評価される式を指定できます。4つ目の引数を指定しない場合、デフォルトは'TWO_SIDED_PROB'です。

関連項目:

STATS_BINOMIAL_TESTの照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

表7-3 STATS_BINOMIALの戻り値

引数	戻り値の意味
'TWO_SIDED_PROB'	指定された母集団の割合 p が、観測された割合より外側になる確率。
'EXACT_PROB'	指定された母集団の割合 p が、観測された割合と正確に同じ値になる確率。
'ONE_SIDED_PROB_OR_MORE'	指定された母集団の割合 p が、観測された割合以上になる確率。
'ONE_SIDED_PROB_OR_LESS'	指定された母集団の割合 p が、観測された割合以下になる確率。

'EXACT_PROB'では、pと一致する割合が戻される確率が戻されます。標本における割合が、50%と大幅に異なるか(有意差があるか)どうかをテストする場合、通常、pを0.50に設定します。割合が異なるかどうかのみをテストする場合、戻り値に'TWO_SIDED_PROB'を使用します。割合がexpr2の値より大きいかどうかをテストする場合、戻り値に'ONE_SIDED_PROB_OR_MORE'を使用します。割合がexpr2より小さいかどうかをテストする場合、戻り値に'ONE_SIDED_PROB_OR_LESS'を使用します。

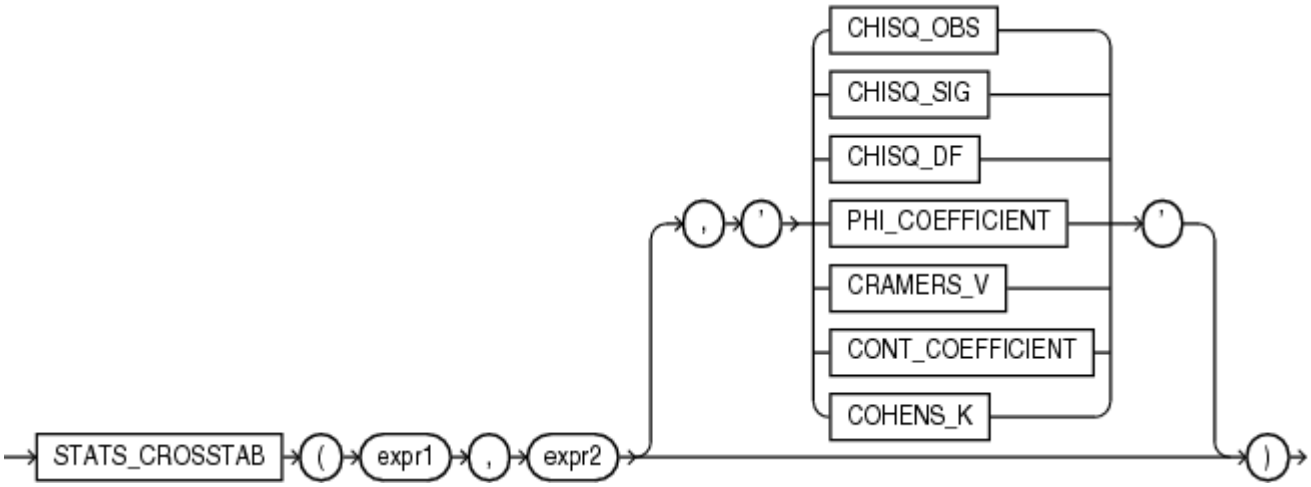
STATS_BINOMIAL_TESTの例

次の例では、母集団の69%が男性であるという仮定に基づいて観測された男性数が、実際の男性数と正確に同一となる確率を判断します。

```
SELECT AVG(DECODE(cust_gender, 'M', 1, 0)) real_proportion,
       STATS_BINOMIAL_TEST
         (cust_gender, 'M', 0.68, 'EXACT_PROB') exact,
       STATS_BINOMIAL_TEST
         (cust_gender, 'M', 0.68, 'ONE_SIDED_PROB_OR_LESS') prob_or_less
  FROM sh.customers;
```

STATS_CROSSTAB

構文



目的

クロス集計は、2つの名義変数の分析に使用する方法です。STATS_CROSSTABファンクションは2つの必須の引数を取ります。分析対象の2つの変数はexpr1とexpr2です。オプションの3つ目の引数を使用すると、表7-4に示すように、このファンクションによって戻されるNUMBER値の意味を指定できます。この引数には、テキスト・リテラル、バインド変数または定数の文字値に評価される式を指定できます。3つ目の引数を指定しない場合、デフォルトは'CHISQ_SIG'です。

関連項目:

STATS_CROSSTABの照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

表7-4 STATS_CROSSTABの戻り値

引数	戻り値の意味
'CHISQ_OBS'	カイ 2 乗の測定値
'CHISQ_SIG'	カイ 2 乗の測定値の有意性
'CHISQ_DF'	カイ 2 乗の自由度
'PHI_COEFFICIENT'	ファイ係数
'CRAMERS_V'	クラメールの V 統計
'CONT_COEFFICIENT'	一致係数

引数	戻り値の意味
'COHENS_K'	コーエンのカッパ

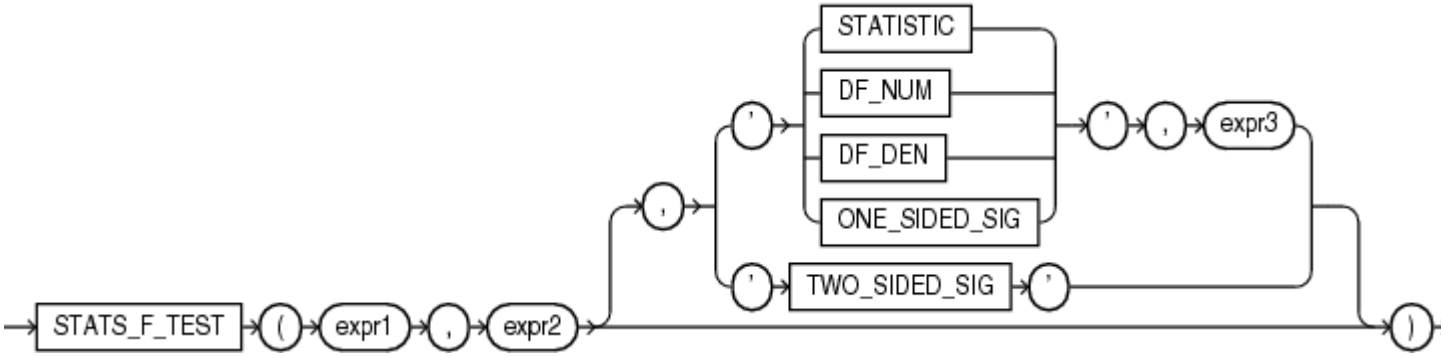
STATS_CROSSTABの例

次の例では、性別と収入水準の関連の強さを判断します。

```
SELECT STATS_CROSSTAB
         (cust_gender, cust_income_level, 'CHISQ_OBS') chi_squared,
       STATS_CROSSTAB
         (cust_gender, cust_income_level, 'CHISQ_SIG') p_value,
       STATS_CROSSTAB
         (cust_gender, cust_income_level, 'PHI_COEFFICIENT') phi_coefficient
  FROM sh.customers;
CHI_SQUARED    P_VALUE PHI_COEFFICIENT
----------- ---------- ---------------
 251.690705 1.2364E-47      .067367056
```

STATS_F_TEST

構文



目的

STATS_F_TESTは、2つの分散に有意差があるかどうかをテストします。fの観測値は、他方の分散に対する一方の分散の比率です。この値が1と大きく異なる場合は、通常、有意差があることを示しています。

このファンクションは2つの必須の引数を取ります。expr1はグループ変数または独立変数で、expr2は値の標本です。オプションの3つ目の引数を使用すると、表7-5に示すように、このファンクションによって戻されるNUMBER値の意味を指定できます。この引数には、テキスト・リテラル、バインド変数または定数の文字値に評価される式を指定できます。3つ目の引数を指定しない場合、デフォルトは'TWO_SIDED_SIG'です。

関連項目:

STATS_F_TESTの照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

表7-5 STATS_F_TESTの戻り値

引数	戻り値の意味
'STATISTIC'	f の観測値
'DF_NUM'	分子の自由度
'DF_DEN'	分母の自由度
'ONE_SIDED_SIG'	f の片側有意
'TWO_SIDED_SIG'	f の両側有意

片側有意は常に上側確率に関連します。最後の引数expr3は、expr1で指定した2つのグループのうち、大きい値または分子(棄却域が上側確率の値)のグループを示します。

fの観測値は、2つ目のグループの分散に対する1つ目のグループの分散の比率です。fの観測値の有意性は、2つの分散が偶然に異なる確率で、0(ゼロ)から1の数値です。この有意性の値が小さい場合、2つの分散に有意差があることを示しています。各分散の自由度は、標本における観測数から1を引いた値です。

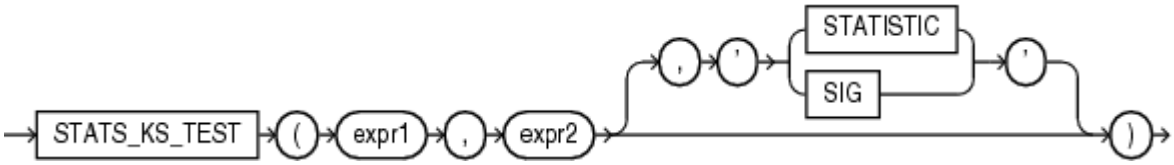
STATS_F_TESTの例

次の例では、男性と女性のクレジット利用限度額に有意差があるかどうかを判断します。p_valueが0(ゼロ)に近くなく、f_statisticが1に近いという結果は、男性と女性のクレジット利用限度額の差が有意でないことを示しています。

```
SELECT VARIANCE(DECODE(cust_gender, 'M', cust_credit_limit, null)) var_men,
       VARIANCE(DECODE(cust_gender, 'F', cust_credit_limit, null)) var_women,
       STATS_F_TEST(cust_gender, cust_credit_limit, 'STATISTIC', 'F') f_statistic,
       STATS_F_TEST(cust_gender, cust_credit_limit) two_sided_p_value
  FROM sh.customers;
   VAR_MEN  VAR_WOMEN F_STATISTIC TWO_SIDED_P_VALUE
---------- ---------- ----------- -----------------
12879896.7   13046865  1.01296348        .311928071
```

STATS_KS_TEST

構文



目的

STATS_KS_TESTは、2つの標本を比較して、それらが同じ母集団に属しているか、または同じ分布を持つ母集団に属しているかをテストするKolmogorov-Smirnovファンクションです。このファンクションでは、標本の母集団が正規分布に従うとは仮定されません。

このファンクションは2つの必須の引数を取ります。expr1には、データを2つの標本に分類する値を指定し、expr2には各標本の値を指定します。expr1に、データを1つのみか3つ以上の標本に分類する値を指定すると、エラーが発生します。オプションの3つ目の引数を使用すると、表7-6に示すように、このファンクションによって戻されるNUMBER値の意味を指定できます。この引数には、テキスト・リテラル、バインド変数または定数の文字値に評価される式を指定できます。3つ目の引数を指定しない場合、デフォルトは'SIG'です。

関連項目:

STATS_KS_TESTの照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

表7-6 STATS_KS_TESTの戻り値

引数	戻り値の意味
'STATISTIC'	D の観測値
'SIG'	D の有意性

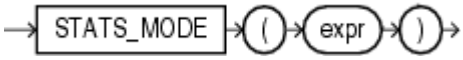
STATS_KS_TESTの例

次の例では、Kolmogorov-Smirnov検定を使用して、男性と女性に対する売上の分布が偶然であるかどうかを判断します。

```
SELECT stats_ks_test(cust_gender, amount_sold, 'STATISTIC') ks_statistic,
       stats_ks_test(cust_gender, amount_sold) p_value
  FROM sh.customers c, sh.sales s
  WHERE c.cust_id = s.cust_id;
KS_STATISTIC    P_VALUE
------------ ----------
  .003841396 .004080006
```

STATS_MODE

構文



目的

STATS_MODEは、引数として値の集合を取り、最も出現頻度の高い値を戻します。複数の最頻値が存在する場合、Oracle Databaseは1つの最頻値を選択し、その値のみを戻します。

複数の最頻値(存在する場合)を取得する場合は、次の不確定な問合せに示すとおり、他のファンクションの組合せを使用する必要があります。

```
SELECT x FROM (SELECT x, COUNT(x) AS cnt1
   FROM t GROUP BY x)
   WHERE cnt1 =
      (SELECT MAX(cnt2) FROM (SELECT COUNT(x) AS cnt2 FROM t GROUP BY x));
```

関連項目:

exprの文字値を比較するためにSTATS_MODEで使用される照合を定義する照合決定ルール、およびこのファンクションの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、hr.employees表の部門ごとの給与の最頻値を戻します。

```
SELECT department_id, STATS_MODE(salary) FROM employees
   GROUP BY department_id
   ORDER BY department_id, stats_mode(salary);
DEPARTMENT_ID STATS_MODE(SALARY)
------------- -----------------
           10              4400
           20              6000
           30              2500
           40              6500
           50              2500
           60              4800
           70             10000
           80              9500
           90             17000
          100              6900
          110              8300
                           7000
```

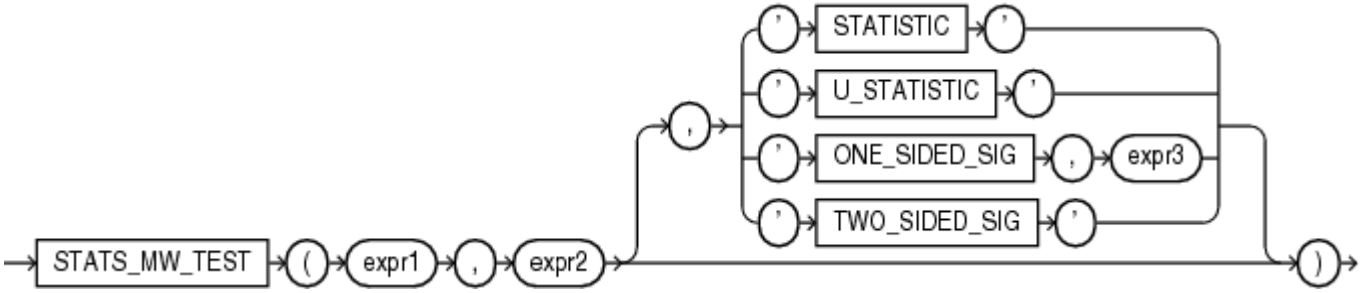
複数の最頻値が存在し、そのすべての最頻値を取得する必要がある場合は、次の例に示すとおり、他のファンクションの組合せを使用します。

```
SELECT commission_pct FROM
   (SELECT commission_pct, COUNT(commission_pct) AS cnt1 FROM employees
      GROUP BY commission_pct)
   WHERE cnt1 =
      (SELECT MAX (cnt2) FROM
         (SELECT COUNT(commission_pct) AS cnt2
         FROM employees GROUP BY commission_pct))
   ORDER BY commission_pct;
```

```
COMMISSION_PCT
--------------
            .2
            .3
```

STATS_MW_TEST

構文



目的

Mann-Whitney検定では、2つの独立した標本を比較して、2つの母集団が同じ分布ファンクションを持つという帰無仮説を、2つの分布ファンクションは異なるという対立仮説に対してテストします。

STATS_MW_TESTでは、STATS_T_TEST_*ファンクションとは異なり、標本間の差が正規分布するとは仮定されません。このファンクションは2つの必須の引数を取ります。expr1には、データをグループに分類する値を指定し、expr2には各グループの値を指定します。オプションの3つ目の引数を使用すると、表7-7に示すように、このファンクションによって戻されるNUMBER値の意味を指定できます。この引数には、テキスト・リテラル、バインド変数または定数の文字値に評価される式を指定できます。3つ目の引数を指定しない場合、デフォルトは'TWO_SIDED_SIG'です。

関連項目:

STATS_MW_TESTの照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

表7-7 STATS_MW_TESTの戻り値

引数	戻り値の意味
'STATISTIC'	Z の観測値
'U_STATISTIC'	U の観測値
'ONE_SIDED_SIG'	Z の片側有意
'TWO_SIDED_SIG'	Z の両側有意

ZまたはUの観測値の有意性は、2つの分散が偶然に異なる確率で、0(ゼロ)から1の数値です。この有意性の値が小さい場合、2つの分散に有意差があることを示しています。各分散の自由度は、標本における観測数から1を引いた値です。

片側有意は常に上側確率に関連します。最後の引数expr3は、expr1で指定した2つのグループのうち、大きい値(棄却域が上側確率の値)のグループを示します。

STATS_MW_TESTは、値の順位の合計における差を確認して、標本が同じ分布からのものである確率を計算します。標本が同じ分布に属している場合、各標本の合計値は近くなります。

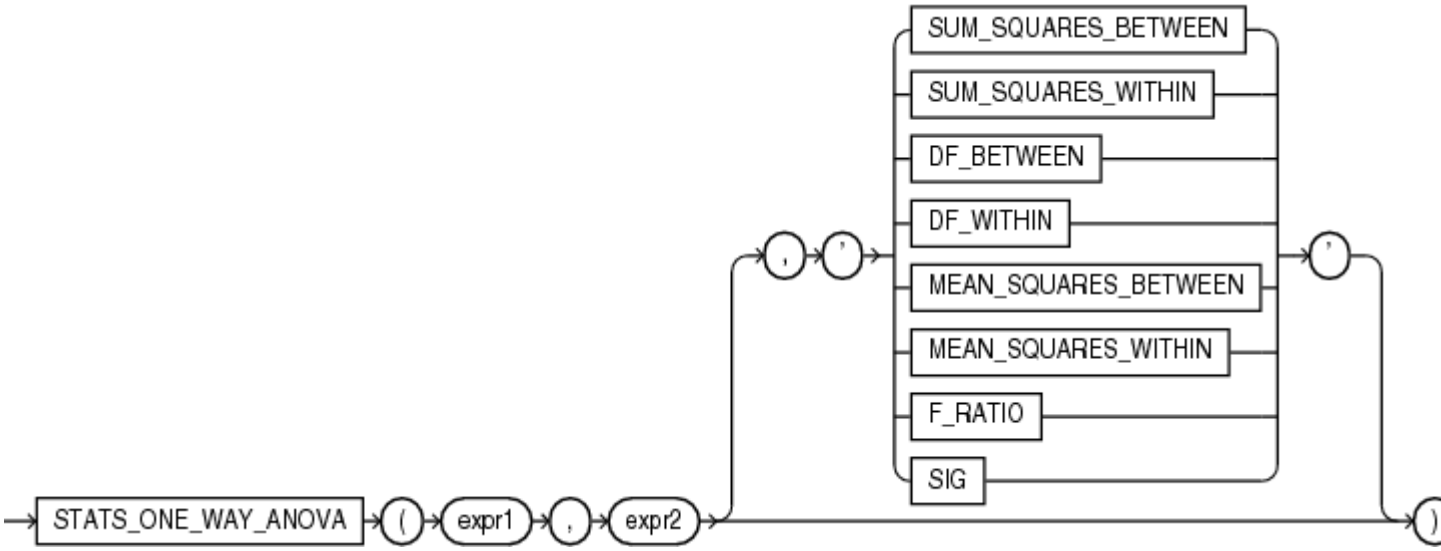
STATS_MW_TESTの例

次の例では、Mann-Whitney検定を使用して、男性と女性に対する売上の分布が偶然であるかどうかを判断します。

```
SELECT STATS_MW_TEST
         (cust_gender, amount_sold, 'STATISTIC') z_statistic,
       STATS_MW_TEST
         (cust_gender, amount_sold, 'ONE_SIDED_SIG', 'F') one_sided_p_value
  FROM sh.customers c, sh.sales s
  WHERE c.cust_id = s.cust_id;
Z_STATISTIC ONE_SIDED_P_VALUE
----------- -----------------
 -1.4011509        .080584471
```

STATS_ONE_WAY_ANOVA

構文



目的

STATS_ONE_WAY_ANOVAファンクションによる一元配置分散分析では、分散の2つの異なる推定値を比較して、(複数のグループまたは変数の)平均値の差をテストして統計学的有意差を求めます。1つ目の推定値は、各グループまたはカテゴリ内の分散に基づきます。これは、群内平均平方または平均平方誤差と呼ばれます。2つ目の推定値は、グループの平均値の分散に基づきます。これは、群間平均平方と呼ばれます。グループの平均値に有意差がある場合、その群間平均平方は期待値より大きくなり、群内平均平方に一致しません。グループの平均平方が一貫している場合、2つの分散の推定値はほぼ同じになります。

STATS_ONE_WAY_ANOVAは、2つの必須の引数を取ります。expr1には、データをグループの集合に分割する独立変数またはグループ変数を指定し、expr2には、グループの各メンバーに対応する値を含む従属変数(数式)を指定します。オプションの3つ目の引数を使用すると、表7-8に示すように、このファンクションによって戻されるNUMBER値の意味を指定できます。この引数には、テキスト・リテラル、バインド変数または定数の文字値に評価される式を指定できます。3つ目の引数を指定しない場合、デフォルトは'SIG'です。

関連項目:

STATS_ONE_WAY_ANOVAの照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

表7-8 STATS_ONE_WAY_ANOVAの戻り値

引数	戻り値の意味
'SUM_SQUARES_BETEEN'	グループ間平方和
'SUM_SQUARES_WITHIN'	グループ内平方和

引数	戻り値の意味
'DF_BETWEEN'	グループ間の自由度
'DF_WITHIN'	グループ内の自由度
'MEAN_SQUARES_BETWEEN'	グループ間平均平方
'MEAN_SQUARES_WITHIN'	グループ内平均平方
'F_RATIO'	群内平均平方に対する群間平均平方の比率(MSB(グループ間平均平方)/MSW(グループ内平均平方))
'SIG'	有意性

一元配置分散分析の有意性は群間平均平方と群内平均平方の比率に対するf検定の片側有意を求めることによって判断されます。f検定では片側有意を使用する必要があります。これは、群間平均平方は、群内平均平方以上のみになるためです。そのため、STATS_ONE_WAY_ANOVAが戻す有意性は、グループ間の差が偶然である確率で、0(ゼロ)から1の数値です。この数値が小さいほど、グループ間の有意差が大きくなります。f検定の実行の詳細は、「STATS_F_TEST」を参照してください。

STATS_ONE_WAY_ANOVAの例

次の例では、収入水準内での売上平均の差と、収入水準間での売上平均の差の有意性を判断します。p_valuesが0(ゼロ)に近いという結果は、男性と女性の両方に対して、様々な収入水準の人々に販売された商品の金額には有意差があることを示しています。

```
SELECT cust_gender,
       STATS_ONE_WAY_ANOVA(cust_income_level, amount_sold, 'F_RATIO') f_ratio,
       STATS_ONE_WAY_ANOVA(cust_income_level, amount_sold, 'SIG') p_value
  FROM sh.customers c, sh.sales s
  WHERE c.cust_id = s.cust_id
  GROUP BY cust_gender
  ORDER BY cust_gender;
C    F_RATIO    P_VALUE
- ---------- ----------
F 5.59536943 4.7840E-09
M  9.2865001 6.7139E-17
```

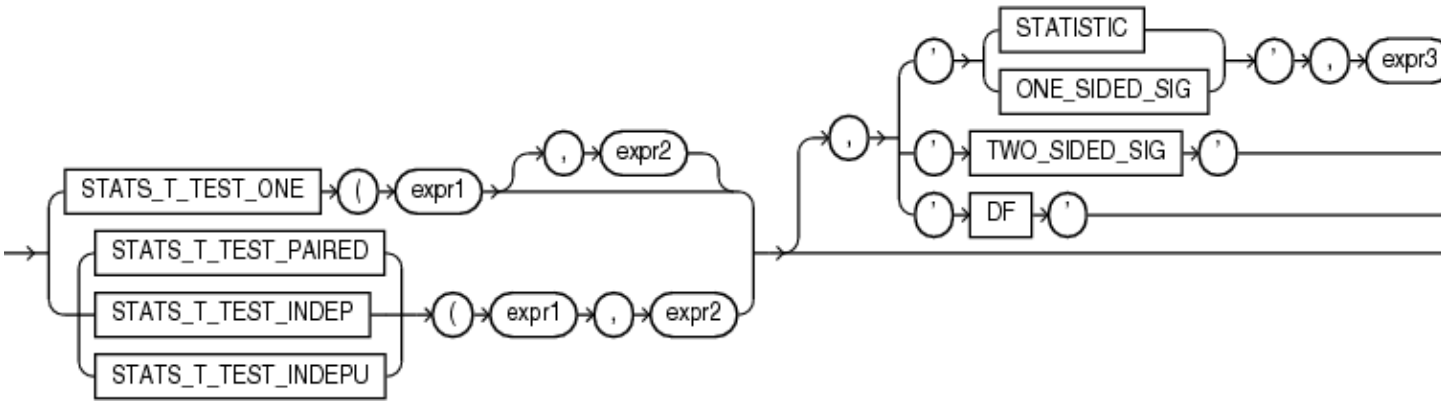
STATS_T_TEST_*

t検定ファンクションを次に示します。

- STATS_T_TEST_ONE: 1標本t検定
- STATS_T_TEST_PAIRED: 対応のある2標本t検定(クロスt検定とも呼ばれる)
- STATS_T_TEST_INDEP: 同じ分散(併合分散)を持つ独立した2つのグループのt検定
- STATS_T_TEST_INDEPU: 異なる分散(非併合分散)を持つ独立した2つのグループのt検定

構文

stats_t_test::=



目的

t検定では、平均値の差の有意性を測定します。この検定を使用して、2つのグループの平均値の比較、または1つのグループの平均値と定数の比較を行えます。t検定ファンクションは2つの式引数を取ります。ただし、2つ目の式は、1標本ファンクション(STATS_T_TEST_ONE)のオプションです。各t検定ファンクションでオプションの3つ目の引数を使用すると、表7-9に示すように、このファンクションによって戻されるNUMBER値の意味を指定できます。この引数には、テキスト・リテラル、バインド変数または定数の文字値に評価される式を指定できます。3つ目の引数を指定しない場合、デフォルトは'TWO_SIDED_SIG'です。

関連項目:

STATS_T_TEST_*ファンクションの照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

表7-9 STATS_T_TEST_*の戻り値

引数	戻り値の意味
'STATISTIC'	t の観測値
'DF'	自由度
'ONE_SIDED_SIG'	t の片側有意

引数	戻り値の意味
'TWO_SIDED_SIG'	t の両側有意

2つの独立したSTATS_T_TEST_*ファンクションは、3つ目の引数が'STATISTIC'または'ONE_SIDED_SIG'として指定されている場合、4つ目の引数(expr3)を取ることができます。この場合、expr3は、expr1のどちらの値が大きい値(棄却域が上側確率の値)であるかを示します。

tの観測値の有意性は、tの値が偶然得られた確率で、0(ゼロ)から1の数値です。この値が小さいほど、平均値間の有意差が大きくなります。片側有意は常に上側確率に関連します。1標本および対応のあるt検定の場合、最初の式が大きい値になります。独立したt検定の場合、expr3で指定した値が大きい値になります。

自由度は、tの観測値を求めるために使用したt検定の種類によって異なります。たとえば、1標本t検定(STATS_T_TEST_ONE)の場合、自由度は標本における観測数から1を引いた値です。

STATS_T_TEST_ONE

STATS_T_TEST_ONEファンクションでは、expr1には標本を指定し、expr2には標本平均値の比較対象となる定数平均値を指定します。t検定の場合にのみ、expr2はオプションとなります。定数平均値は、デフォルトで0になります。このファンクションは、標本平均値と既知の平均値の差を、平均値の標準誤差で割ってt値を求めます(STATS_T_TEST_PAIREDでは、平均値の差の標準誤差で割ります)。

STATS_T_TEST_ONEの例

次の例では、平均表示価格と定数値60の差の有意性を判断します。

```
SELECT AVG(prod_list_price) group_mean,
       STATS_T_TEST_ONE(prod_list_price, 60, 'STATISTIC') t_observed,
       STATS_T_TEST_ONE(prod_list_price, 60) two_sided_p_value
  FROM sh.products;
GROUP_MEAN T_OBSERVED TWO_SIDED_P_VALUE
---------- ---------- -----------------
139.545556 2.32107746        .023158537
```

STATS_T_TEST_PAIRED

STATS_T_TEST_PAIREDファンクションでは、expr1およびexpr2には、比較する平均値の計算元の2つの標本をそれぞれ指定します。このファンクションは、標本平均値の差を、平均値の差の標準誤差で割ってt値を求めます(STATS_T_TEST_ONEでは、平均値の標準誤差で割ります)。

STATS_T_TEST_INDEPおよびSTATS_T_TEST_INDEPU

STATS_T_TEST_INDEPおよびSTATS_T_TEST_INDEPUファンクションでは、expr1はグループ列で、expr2は値の標本です。併合分散用のファンクション(STATS_T_TEST_INDEP)は、同様の分散を持つ2つの分布において、平均値が同じか異なるかをテストします。非併合分散用のファンクション(STATS_T_TEST_INDEPU)は、既知の有意差のある分散を持つ2つの分布においても、平均値が同じか異なるかをテストします。

これらのファンクションを使用する前に、標本の分散に有意差があるかどうかを判断しておくことをお薦めします。有意差がある場合、そのデータは異なる形状の分布に属している可能性があり、平均値の差が有効でない場合があります。f検定を実行して、

分散の差を判断できます。分散に有意差がない場合、STATS_T_TEST_INDEPを使用します。有意差がある場合は、STATS_T_TEST_INDEPUを使用します。f検定の実行の詳細は、「STATS_F_TEST」を参照してください。

STATS_T_TEST_INDEPの例

次の例では、各分布の分散が同様である(併合分散)として、男性と女性に対する平均売上間の差の有意性を判断します。

```
SELECT SUBSTR(cust_income_level, 1, 22) income_level,
      AVG(DECODE(cust_gender, 'M', amount_sold, null)) sold_to_men,
      AVG(DECODE(cust_gender, 'F', amount_sold, null)) sold_to_women,
      STATS_T_TEST_INDEP(cust_gender, amount_sold, 'STATISTIC', 'F') t_observed,
      STATS_T_TEST_INDEP(cust_gender, amount_sold) two_sided_p_value
  FROM sh.customers c, sh.sales s
  WHERE c.cust_id = s.cust_id
  GROUP BY ROLLUP(cust_income_level)
  ORDER BY income_level, sold_to_men, sold_to_women, t_observed;
INCOME_LEVEL           SOLD_TO_MEN SOLD_TO_WOMEN T_OBSERVED TWO_SIDED_P_VALUE
---------------------- ----------- ------------- ---------- -----------------
A: Below 30,000          105.28349    99.4281447 -1.9880629        .046811482
B: 30,000 - 49,999       102.59651    109.829642 3.04330875        .002341053
C: 50,000 - 69,999      105.627588    110.127931 2.36148671        .018204221
D: 70,000 - 89,999      106.630299     110.47287 2.28496443        .022316997
E: 90,000 - 109,999     103.396741    101.610416 -1.2544577        .209677823
F: 110,000 - 129,999     106.76476    105.981312 -.60444998        .545545304
G: 130,000 - 149,999    108.877532     107.31377 -.85298245        .393671218
H: 150,000 - 169,999    110.987258    107.152191 -1.9062363        .056622983
I: 170,000 - 189,999    102.808238     107.43556 2.18477851        .028908566
J: 190,000 - 249,999    108.040564    115.343356 2.58313425        .009794516
K: 250,000 - 299,999    112.377993    108.196097 -1.4107871        .158316973
L: 300,000 and above    120.970235    112.216342 -2.0642868        .039003862
                        107.121845     113.80441 .686144393        .492670059
                        106.663769    107.276386 1.08013499        .280082357
14 rows selected.
```

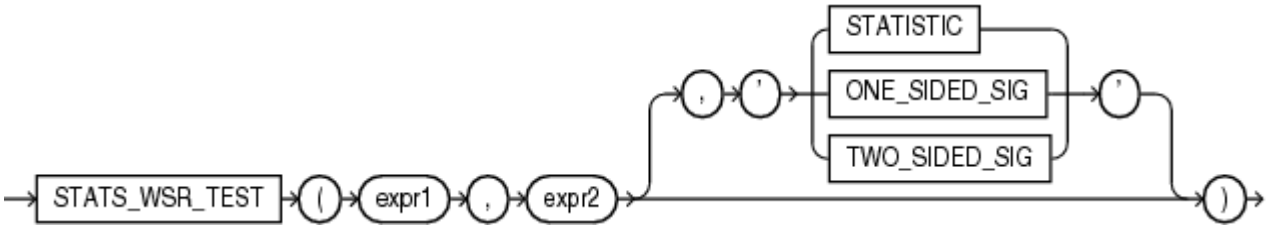
STATS_T_TEST_INDEPUの例

次の例では、各分布の分散に既知の有意差がある(非併合分散)として、男性と女性に対する平均売上間の差の有意性を判断します。

```
SELECT SUBSTR(cust_income_level, 1, 22) income_level,
       AVG(DECODE(cust_gender, 'M', amount_sold, null)) sold_to_men,
       AVG(DECODE(cust_gender, 'F', amount_sold, null)) sold_to_women,
       STATS_T_TEST_INDEPU(cust_gender, amount_sold, 'STATISTIC', 'F') t_observed,
       STATS_T_TEST_INDEPU(cust_gender, amount_sold) two_sided_p_value
  FROM sh.customers c, sh.sales s
  WHERE c.cust_id = s.cust_id
  GROUP BY ROLLUP(cust_income_level)
  ORDER BY income_level, sold_to_men, sold_to_women, t_observed;
INCOME_LEVEL           SOLD_TO_MEN SOLD_TO_WOMEN T_OBSERVED TWO_SIDED_P_VALUE
---------------------- ----------- ------------- ---------- -----------------
A: Below 30,000          105.28349    99.4281447 -2.0542592        .039964704
B: 30,000 - 49,999       102.59651    109.829642 2.96922332        .002987742
C: 50,000 - 69,999      105.627588    110.127931  2.3496854        .018792277
D: 70,000 - 89,999      106.630299     110.47287 2.26839281        .023307831
E: 90,000 - 109,999     103.396741    101.610416 -1.2603509        .207545662
F: 110,000 - 129,999     106.76476    105.981312 -.60580011        .544648553
G: 130,000 - 149,999    108.877532     107.31377 -.85219781        .394107755
H: 150,000 - 169,999    110.987258    107.152191 -1.9451486        .051762624
I: 170,000 - 189,999    102.808238     107.43556 2.14966921        .031587875
J: 190,000 - 249,999    108.040564    115.343356 2.54749867        .010854966
K: 250,000 - 299,999    112.377993    108.196097 -1.4115514        .158091676
L: 300,000 and above    120.970235    112.216342 -2.0726194        .038225611
                        107.121845     113.80441 .689462437        .490595765
                        106.663769    107.276386 1.07853782        .280794207
14 rows selected.
```

STATS_WSR_TEST

構文



目的

STATS_WSR_TESTは、対応のある標本のウィルコクソンの符号順位検定であり、標本間の差の中央値と0(ゼロ)に有意差があるかどうかを判断します。差の絶対値は、標本を順序付けし、順位に符号を付けて求められます。また、帰無仮説によって、正の差の順位の合計が、負の差の順位の合計に等しいと仮定されます。

このファンクションは2つの必須の引数を取ります。分析対象の2つの標本はexpr1とexpr2です。オプションの3つ目の引数を使用すると、表7-10に示すように、このファンクションによって戻されるNUMBER値の意味を指定できます。この引数には、テキスト・リテラル、バインド変数または定数の文字値に評価される式を指定できます。3つ目の引数を指定しない場合、デフォルトは'TWO_SIDED_SIG'です。

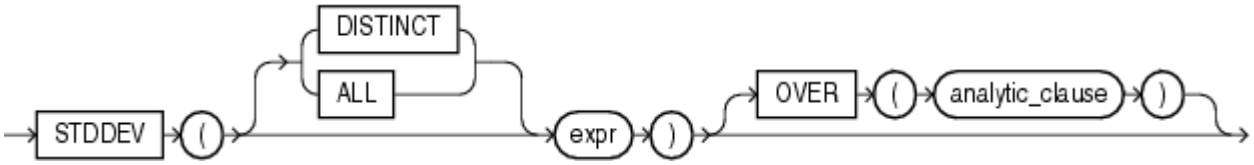
表7-10 STATS_WSR_TEST_*の戻り値

引数	戻り値の意味
'STATISTIC'	Z の観測値
'ONE_SIDED_SIG'	Z の片側有意
'TWO_SIDED_SIG'	Z の両側有意

片側有意は常に上側確率に関連します。expr1が大きい値(棄却域が上側確率の値)になります。

STDDEV

構文



関連項目:

構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

STDDEVは、数値の集合であるexprの標本標準偏差を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。このファンクションは、STDDEV_SAMPがNULLを戻すことに対して、入力データが1行のみの場合にSTDDEVが0(ゼロ)を戻すという点で、STDDEV_SAMPと異なります。

Oracle Databaseは、VARIANCE集計ファンクションに対して定義された分散の平方根として標準偏差を計算します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を返します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

DISTINCTを指定する場合は、analytic_clauseのquery_partition_clauseのみを指定できます。order_by_clauseおよびwindowing_clauseは指定できません。

関連項目:

- 「集計ファンクション」、「VARIANCE」および「STDDEV_SAMP」を参照してください。
- exprの有効な書式の詳細は、「SQL式」を参照してください。

集計の例

次の例では、サンプル表hr.employeesの給与値の標本標準偏差を戻します。

```
SELECT STDDEV(salary) "Deviation"
   FROM employees;

 Deviation
----------
3909.36575
```

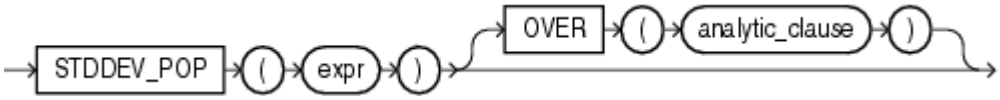
分析の例

次の例では、hire_dateで順序付けられたサンプル表hr.employeesの部門80の給与の値の累積標準偏差を戻します。

```
SELECT last_name, salary,
   STDDEV(salary) OVER (ORDER BY hire_date) "StdDev"
   FROM employees
   WHERE department_id = 30
   ORDER BY last_name, salary, "StdDev";

LAST_NAME                     SALARY     StdDev
------------------------- ---------- ----------
Baida                           2900 4035.26125
Colmenares                      2500 3362.58829
Himuro                          2600  3649.2465
Khoo                            3100 5586.14357
Raphaely                       11000          0
Tobias                          2800  4650.0896
```

STDDEV_POP

構文



関連項目:

構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

STDDEV_POPは母集団標準偏差を計算し、母集団分散の平方根を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を返します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

このファンクションは、VAR_POPファンクションの平方根と同じです。VAR_POPがNULLを戻す場合、このファンクションもNULLを戻します。

関連項目:

- 「集計ファンクション」および「VAR_POP」を参照してください。
- exprの有効な書式の詳細は、「SQL式」を参照してください。

集計の例

次の例では、サンプル表sh.salesにある売上高の母集団標準偏差および標本標準偏差を戻します。

```
SELECT STDDEV_POP(amount_sold) "Pop",
   STDDEV_SAMP(amount_sold) "Samp"
   FROM sales;
       Pop       Samp
---------- ----------
896.355151 896.355592
```

分析の例

次の例では、サンプル表hr.employeesの部門ごとの給与の母集団標準偏差を戻します。

```
SELECT department_id, last_name, salary,
   STDDEV_POP(salary) OVER (PARTITION BY department_id) AS pop_std
   FROM employees
   ORDER BY department_id, last_name, salary, pop_std;
DEPARTMENT_ID LAST_NAME                     SALARY    POP_STD
------------- ------------------------- ---------- ----------
           10 Whalen                          4400          0
```

```
          20 Fay                              6000       3500
          20 Hartstein                       13000       3500
          30 Baida                            2900  3069.6091
. . .
         100 Urman                            7800 1644.18166
         110 Gietz                            8300       1850
         110 Higgins                         12000       1850
             Grant                            7000          0
```

STDDEV_SAMP

構文

関連項目:

構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

STDDEV_SAMPは標本累積標準偏差を計算し、標本分散の平方根を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を返します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

このファンクションは、VAR_SAMPファンクションの平方根と同じです。VAR_SAMPがNULLを戻す場合、このファンクションもNULLを戻します。

関連項目:

- 「集計ファンクション」および「VAR_SAMP」を参照してください。
- exprの有効な書式の詳細は、「SQL式」を参照してください。

集計の例

「STDDEV_POP」の集計の例を参照してください。

分析の例

次の例では、employees表の部門ごとの給与の標本標準偏差を戻します。

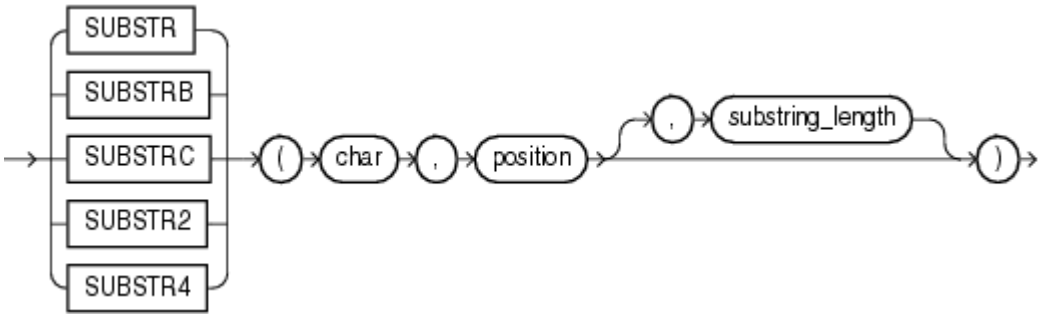
```
SELECT department_id, last_name, hire_date, salary,
   STDDEV_SAMP(salary) OVER (PARTITION BY department_id
      ORDER BY hire_date
      ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS cum_sdev
   FROM employees
   ORDER BY department_id, last_name, hire_date, salary, cum_sdev;
DEPARTMENT_ID LAST_NAME       HIRE_DATE     SALARY   CUM_SDEV
------------- --------------- --------- ---------- ----------
           10 Whalen          17-SEP-03       4400
           20 Fay             17-AUG-05       6000 4949.74747
           20 Hartstein       17-FEB-04      13000
           30 Baida           24-DEC-05       2900 4035.26125
           30 Colmenares      10-AUG-07       2500 3362.58829
           30 Himuro          15-NOV-06       2600  3649.2465
```

```
           30 Khoo            18-MAY-03       3100 5586.14357
           30 Raphaely        07-DEC-02      11000
. . .
          100 Greenberg       17-AUG-02      12008  2126.9772
          100 Popp            07-DEC-07       6900 1804.13155
          100 Sciarra         30-SEP-05       7700 1929.76233
          100 Urman           07-MAR-06       7800 1788.92504
          110 Gietz           07-JUN-02       8300 2621.95194
          110 Higgins         07-JUN-02      12008
              Grant           24-MAY-07       7000
```

SUBSTR

構文

substr::=



目的

SUBSTRの各ファンクションは、charのpositionの文字からsubstring_length文字分の文字列を抜き出して戻します。SUBSTRは、入力文字セットによって定義された文字を使用して、長さを計算します。SUBSTRBには、文字でなくバイトを使用します。SUBSTRCは、完全なUnicodeキャラクタを使用します。SUBSTR2は、UCS2コードポイントを使用します。SUBSTR4は、UCS4コードポイントを使用します。

- positionが0の場合、1として処理されます。
- positionが正の場合、Oracle Databaseはcharの始めから数えて最初の文字を検索します。
- positionが負の場合、Oracleはcharの終わりから逆方向に数えます。
- substring_lengthを指定しないと、Oracleはcharの終わりまでのすべての文字を戻します。substring_lengthが1より小さい場合、NULLを戻します。

charのデータ型は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBです。例外は、SUBSTRC、SUBSTR2およびSUBSTR4です。これらでは、charをCLOBおよびNCLOBのいずれにもできません。positionおよびsubstring_lengthは、NUMBERデータ型か、または暗黙的にNUMBERに変換可能な任意のデータ型で、整数である必要があります。戻り値はcharと同じデータ型ですが、例外として、CHAR引数の場合はVARCHAR2値が返され、NCHAR引数の場合はNVARCHAR2値が返されます。引数としてSUBSTRに渡された浮動小数点数は、自動的に整数に変換されます。

関連項目:

- 文字長の詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』および『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。
- 異なるロケールでのSUBSTRファンクションおよび長さセマンティクスの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。
- SUBSTRの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、「ABCDEFG」の指定されたサブストリングを戻します。

```
SELECT SUBSTR('ABCDEFG',3,4) "Substring"
```

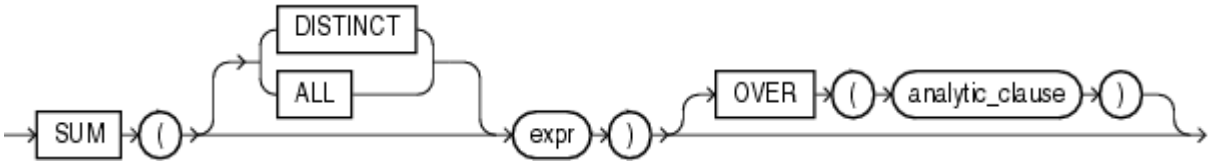
```
     FROM DUAL;

Substring
---------
CDEF
SELECT SUBSTR('ABCDEFG',-5,4) "Substring"
     FROM DUAL;
Substring
---------
CDEF
```

データベース文字セットがダブルバイトの場合を想定します。

```
SELECT SUBSTRB('ABCDEFG',5,4.2) "Substring with bytes"
     FROM DUAL;
Substring with bytes
--------------------
CD
```

SUM

構文



関連項目:

構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

SUMは、exprの値の合計を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を返します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

DISTINCTを指定する場合は、analytic_clauseのquery_partition_clauseのみを指定できます。order_by_clauseおよびwindowing_clauseは指定できません。

関連項目:

exprの有効な書式の詳細は、「SQL式」を参照してください。また、「集計ファンクション」を参照してください。

集計の例

次の例では、サンプル表hr.employeesにあるすべての給与の合計を計算します。

```
SELECT SUM(salary) "Total"
     FROM employees;

     Total
----------
    691400
```

分析の例

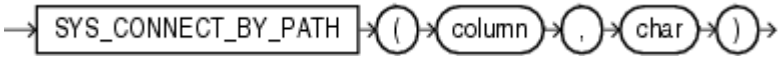
次の例では、サンプル表hr.employeesの各マネージャについて、そのマネージャの下で働く従業員の現在の給与以下の給与の累積合計を計算します。RaphaelyおよびCambraultが同じ累積を持っています。これは、RaphaelyおよびCambraultが同じ給与であるため、Oracle Databaseが給与の値を同時に追加し、同じ累積合計を両方の行に対して適用したためです。

```
SELECT manager_id, last_name, salary,
   SUM(salary) OVER (PARTITION BY manager_id ORDER BY salary
   RANGE UNBOUNDED PRECEDING) l_csum
   FROM employees
```

```
   ORDER BY manager_id, last_name, salary, l_csum;
MANAGER_ID LAST_NAME                 SALARY     L_CSUM
---------- ------------------------- ---------- ----------
       100 Cambrault                      11000      68900
       100 De Haan                        17000     155400
       100 Errazuriz                      12000      80900
       100 Fripp                           8200      36400
       100 Hartstein                      13000      93900
       100 Kaufling                        7900      20200
       100 Kochhar                        17000     155400
       100 Mourgos                         5800       5800
       100 Partners                       13500     107400
       100 Raphaely                       11000      68900
       100 Russell                        14000     121400
. . .
       149 Hutton                          8800      39000
       149 Johnson                         6200       6200
       149 Livingston                      8400      21600
       149 Taylor                          8600      30200
       201 Fay                             6000       6000
       205 Gietz                           8300       8300
           King                           24000      24000
```

SYS_CONNECT_BY_PATH

構文



目的

SYS_CONNECT_BY_PATHは、階層問合せのみで有効です。このファンクションは、ルートからノードへの列の値のパスを、CONNECT BY条件によって戻された各行をcharで区切った列の値とともに戻します。

columnおよびcharは、CHAR、VARCHAR2、NCHARまたはNVARCHAR2データ型です。戻される文字列は、VARCHAR2データ型であり、columnと同じ文字セットです。

関連項目:

- 階層問合せおよびCONNECT BY条件の詳細は、「階層問合せ」を参照してください。
- SYS_CONNECT_BY_PATHの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

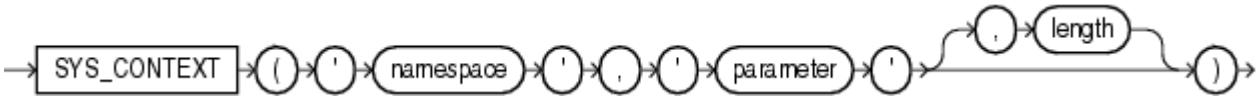
例

次の例では、従業員KochharからKochharのすべての従業員(およびそれらの従業員の従業員)への従業員名のパスを戻します。

```
SELECT LPAD(' ', 2*level-1)||SYS_CONNECT_BY_PATH(last_name, '/') "Path"
   FROM employees
   START WITH last_name = 'Kochhar'
   CONNECT BY PRIOR employee_id = manager_id;
Path
------------------------------
     /Kochhar/Greenberg/Chen
     /Kochhar/Greenberg/Faviet
     /Kochhar/Greenberg/Popp
     /Kochhar/Greenberg/Sciarra
     /Kochhar/Greenberg/Urman
     /Kochhar/Higgins/Gietz
   /Kochhar/Baer
   /Kochhar/Greenberg
   /Kochhar/Higgins
   /Kochhar/Mavris
   /Kochhar/Whalen
 /Kochhar
```

SYS_CONTEXT

構文



目的

SYS_CONTEXTは、現行のインスタンスのコンテキストnamespaceに関連付けられたparameterの値を戻します。この関数は、SQL文とPL/SQL文の両方で使用できます。SYS_CONTEXTはローカルに実行する必要があります。

namespaceおよびparameterには、文字列、またはネームスペースまたは属性を指定する文字列に変換する式を指定できます。namespaceおよびparameterにリテラル引数を指定し、(SQL文で指定されたPL/SQLファンクション内ではなく)SQL文で明示的にSYS_CONTEXTを使用した場合は、SYS_CONTEXTファンクションを起動するコール元の各サイトで、SQL文の実行ごとに1回のみSYS_CONTEXTが評価されます。

コンテキストnamespaceはすでに作成されている必要があり、関連付けられたparameterおよびその値は、DBMS_SESSION.set_contextプロシージャを使用して設定されている必要があります。namespaceは有効な識別子である必要があります。parameter名には、すべての文字列を指定できます。大/小文字は区別されず、長さは30バイト以下です。

戻り値のデータ型はVARCHAR2です。戻り値のデフォルトの最大サイズは256バイトです。オプションのlengthパラメータを指定して、このデフォルトを上書きできます。このパラメータは、NUMBERか、または暗黙的にNUMBERに変換可能な値である必要があります。有効な値の範囲は、1から4000バイトです。無効な値を指定すると、Oracle Databaseはその値を無視してデフォルト値を使用します。

Oracleは、次の組込みネームスペースを提供しています。

- USERENV: 現在のセッションを説明します。ネームスペースUSERENVの事前定義パラメータについては、表7-11を参照してください。
- SYS_SESSION_ROLES: 指定されたロールがセッションに対して現在有効かどうかを示します。Oracle Databaseでは、現在のユーザーのSYS_SESSION_ROLESのコンテキストが評価されます。また、定義者権限プロシージャまたはファンクション内でSYS_SESSION_ROLESが評価されるときには、定義しているユーザーのロールであると仮定されます。SYS_SESSION_ROLESを使用して定義者権限プロシージャ内のログイン・ユーザーで有効なロールを検索するかわりに、DBMS_SESSION:SESSION_IS_ROLE_ENABLED関数を使用できます。起動者権限プロシージャまたはファンクション、あるいはコード・ベースのアクセス制御(CBAC)をかわりに使用することもできます。

関連項目:

- アプリケーション開発でのアプリケーション・コンテキスト機能の使用方法については、『Oracle Databaseセキュリティ・ガイド』を参照してください。
- ユーザー定義のコンテキスト・ネームスペースの作成方法については、「CREATE CONTEXT」を参照してください。
- DBMS_SESSION.set_contextプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- SYS_CONTEXTの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリ

ゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、データベースにログインしたユーザー名を戻します。

```
CONNECT OE
Enter password: password
SELECT SYS_CONTEXT ('USERENV', 'SESSION_USER')
   FROM DUAL;
SYS_CONTEXT ('USERENV', 'SESSION_USER')
----------------------------------------
OE
```

次の例では、SESSION_ROLESデータ・ディクショナリ・ビューを問い合せて、セッションに対して現在有効なロールがRESOURCEのみであることを示します。その後で、SYS_CONTEXTファンクションを使用して、セッションに対してRESOURCEロールが現在有効であり、DBAロールが有効でないことを示します。

```
CONNECT OE
Enter password: password
SELECT role FROM session_roles;
ROLE
--------
RESOURCE
SELECT SYS_CONTEXT('SYS_SESSION_ROLES', 'RESOURCE')
  FROM DUAL
SYS_CONTEXT('SYS_SESSION_ROLES','RESOURCE')
----------------------------------------
TRUE
SELECT SYS_CONTEXT('SYS_SESSION_ROLES', 'DBA')
  FROM DUAL;
SYS_CONTEXT('SYS_SESSION_ROLES','DBA')
----------------------------------------
FALSE
```

ノート:

この機能を簡単に実演する場合、これらの例では、通常デプロイされたシステムが使用するパスワード管理技術を実行しません。本番環境では、Oracle Database のパスワード管理ガイドラインに従い、サンプル・アカウントを無効にしてください。パスワード管理ガイドラインおよびその他のセキュリティ上の推奨事項については、『Oracle Database セキュリティ・ガイド』を参照してください。

次の例では、hr_appsの作成時に、コンテキストhr_appsに関連付けられたPL/SQLパッケージの属性group_noに対する値として設定されたグループ番号を戻します。

```
SELECT SYS_CONTEXT ('hr_apps', 'group_no') "User Group"
   FROM DUAL;
```

表7-11 ネームスペースUSERENVの事前定義パラメータ

パラメータ	戻り値
ACTION	モジュール(アプリケーション名)内の位置を識別します。DBMS_APPLICATION_INFO パッケージまたは OCI を使用して設定します。

パラメータ	戻り値
IS_APPLICATION_ROOT	アプリケーションがアプリケーション・ルートであるかどうかを示します。
IS_APPLICATION_PDB	コンテナがアプリケーション PDB であるかどうかを示します。
AUDITED_CURSORID	監査をトリガーによって実行した SQL のカーソル ID を戻します。このパラメータは、ファイングレイン監査環境では無効です。このパラメータをファイングレイン監査環境で指定すると、Oracle Database は常に NULL を戻します。
AUTHENTICATED_IDENTITY	認証で使用する識別を戻します。次のリストでは、ユーザーのタイプの後に、戻される値を示します。 ● Kerberos 認証済のエンタープライズ・ユーザー: Kerberos プリンシパル名 ● Kerberos 認証済の外部ユーザー: Kerberos プリンシパル名(スキーマ名と同一) ● SSL 認証済のエンタープライズ・ユーザー: ユーザーの PKI 証明書の DN ● SSL 認証済の外部ユーザー: ユーザーの PKI 証明書の DN ● パスワード認証済のエンタープライズ・ユーザー: ニックネーム(ログイン名と同一) ● パスワード認証済のデータベース・ユーザー: データベースのユーザー名(スキーマ名と同一) ● OS 認証済の外部ユーザー: 外部オペレーティング・システムのユーザー名 ● RADIUS 認証済の外部ユーザー: スキーマ名 ● DN 付きプロキシ: クライアントの Oracle Internet Directory DN ● 証明書付きプロキシ: クライアントの証明書 DN ● ユーザー名付きプロキシ: クライアントがローカル・データベースのユーザーの場合はデータベース・ユーザー名、クライアントがエンタープライズ・ユーザーの場合はニックネーム。 ● パスワード・ファイルを使用する SYSDBA/SYSOPER: ログイン名 ● OS 認証を使用する SYSDBA/SYSOPER: オペレーティング・システムのユーザー名 ● パスワード認証済 OCI IAM ユーザー: IAM ユーザー名。ログイン名と同じです

パラメータ	戻り値
	● IAM トークン認証済エンタープライズ・ユーザー: IAM ユーザー名
AUTHENTICATION_DATA	ログイン・ユーザーの認証に使用されるデータを戻します。X.503 認証セッションでは、このフィールドは HEX2 形式での認証のコンテキストを戻します。 ノート: 構文の length パラメータを使用して、AUTHENTICATION_DATA 属性の戻り値を変更できます。最大 4000 までの値を指定できます。この属性は、Oracle Database がこのような変更を実行する USERENV の唯一の属性です。
AUTHENTICATION_METHOD	認証方式を戻します。次に、ユーザー・タイプの後に返される方式を続けて示します。 ● パスワード認証済のエンタープライズ・ユーザー、ローカル・データベースのユーザーまたはパスワード・ファイルを使用する SYSDBA/SYSOPER 管理者権限(パスワードを使用するユーザー名付きのプロキシ): PASSWORD ● パスワード認証済のエンタープライズ・ユーザー、OCI IAM ユーザー、ローカル・データベースのユーザー、またはパスワード・ファイルを使用する SYSDBA/SYSOPER 管理者権限(パスワードを使用するユーザー名付きのプロキシ): PASSWORD_GLOBAL ● OCI IAM トークン認証済エンタープライズ・ユーザー: TOKEN_GLOBAL ● Kerberos 認証済のエンタープライズ・ユーザーまたは外部ユーザー(管理者権限なし): KERBEROS ● Kerberos 認証済のエンタープライズ・ユーザー(管理者権限あり): KERBEROS_GLOBAL ● Kerberos 認証済の外部ユーザー(管理者権限あり): KERBEROS_EXTERNAL ● SSL 認証済のエンタープライズ・ユーザーまたは外部ユーザー(管理者権限なし): SSL ● SSL 認証済のエンタープライズ・ユーザー(管理者権限あり): SSL_GLOBAL ● SSL 認証済の外部ユーザー(管理者権限あり): SSL_EXTERNAL ● RADIUS 認証済の外部ユーザー: RADIUS ● OS 認証済の外部ユーザーまたは SYSDBA または SYSOPER 管理権限を持つユーザー: OS

パラメータ	戻り値
	● 証明書付きプロキシ、DN、またはパスワードを使用しないユーザー名: NONE ● バックグラウンド・プロセス(ジョブ・キュー・スレーブ・プロセス): JOB ● パラレル問合せスレーブ・プロセス: PQ_SLAVE 管理接続以外では、認証方式が PASSWORD、KERBEROS または SSL の場合、IDENTIFICATION_TYPE を使用して外部ユーザーとエンタープライズ・ユーザーを区別できます。管理接続の場合、AUTHENTICATION_METHOD は PASSWORD、SSL_EXTERNAL および SSL_GLOBAL 認証方式に有効です。
BG_JOB_ID	現行のセッションが Oracle Database のバックグラウンド・プロセスで確立された場合、そのセッションのジョブ ID を戻します。セッションがバックグラウンド・プロセスで確立されていない場合は、NULL を戻します。
CDB_DOMAIN	CDB_DOMAIN は CDB の DB_DOMAIN であり、それに関連付けられたすべての PDB で同じです。
CDB_NAME	マルチテナント・コンテナ・データベース(CDB)に接続しているときに問合せが実行されると、CDB の名前を戻します。それ以外の場合は、null を戻します。
CLIENT_IDENTIFIER	アプリケーションによって設定された識別子を、DBMS_SESSION.SET_IDENTIFIER プロシージャ、OCI 属性 OCI_ATTR_CLIENT_IDENTIFIER または Oracle Dynamic Monitoring Service (DMS)を使用して戻します。この属性は、同じデータベース・ユーザーとして認証される複数の軽量アプリケーション・ユーザーを識別するために、様々なデータベース・コンポーネントによって使用されます。
CLIENT_INFO	DBMS_APPLICATION_INFO パッケージを使用するアプリケーションが格納できる 64 バイトまでのユーザー・セッション情報を戻します。
CLIENT_PROGRAM_NAME	データベース・セッションに使用されるプログラムの名前。
CON_ID	CDB に接続しているときに問合せが実行されると、現在のコンテナ ID を戻します。それ以外の場合は 0 を戻します。
CON_NAME	CDB に接続しているときに問合せが実行されると、現在のコンテナ名を戻します。それ以外の場合は、DB_NAME 初期化パラメータで指定されたデータベースの名前を戻します。
CURRENT_BIND	ファイングレイン監査のバインド変数。この属性は、ファイングレイン監査機能のイベント・ハンドラ内のみで指定できます。

パラメータ	戻り値
CURRENT_EDITION_ID	現行のエディションの識別子。
CURRENT_EDITION_NAME	現行のエディションの名前。
CURRENT_SCHEMA	現在アクティブなデフォルトのスキーマの名前。この値は、セッションの存続期間中に ALTER SESSION SET CURRENT_SCHEMA 文を使用して変更できます。また、この値は、セッションの存続期間中に、アクティブな定義者権限オブジェクトの所有者を反映するために変更することもできます。この値をビュー定義の本体で直接使用すると、そのビューを使用しているカーソルを実行するときに使用されるデフォルトのスキーマが戻されます。定義者権限としてカーソルで使用されるビューは考慮されません。 ノート: すべての種類(ログオン・トリガーを除く)のストアド PL/SQL ユニット内から、SQL 文 ALTER SESSION SET CURRENT_SCHEMA を発行しないようにしてください。
CURRENT_SCHEMAID	現在アクティブなデフォルトのスキーマの識別子。
CURRENT_SQL CURRENT_SQLn	CURRENT_SQL は、ファイングレイン監査イベントをトリガーによって実行した現行の SQL の最初の 4KB を戻します。CURRENT_SQLn 属性は、後続の 4KB ずつを戻します。n は、1 から 7(1 および 7 を含む)の整数です。CURRENT_SQL1 は 4 から 8KB、CURRENT_SQL2 は 8 から 12KB のように戻します。これらの属性は、ファイングレイン監査機能のイベント・ハンドラ内のみで指定できます。
CURRENT_SQL_LENGTH	ファイングレイン監査または行レベルのセキュリティ(RLS)ポリシーのファンクションまたはイベント・ハンドラをトリガーする現行の SQL 文の長さ。この属性は、ファイングレイン監査機能のイベント・ハンドラ内のみで指定できます。
CURRENT_USER	現在アクティブな権限を持つデータベース・ユーザー名。これは、Real Application Security セッションがアタッチまたはデタッチされるときや、アクティブな定義者権限オブジェクトの所有者を反映するために、データベース・セッション中に変化することがあります。定義者権限オブジェクトがアクティブでない場合、CURRENT_USER は SESSION_USER と同じ値を戻します。この値をビュー定義の本体で直接使用すると、そのビューを使用しているカーソルを実行しているユーザーが戻されます。定義者権限としてカーソルで使用されるビューは考慮されません。エンタープライズ・ユーザーの場合、スキーマを返します。Real Application Security ユーザーが現在アクティブな場合は、ユーザーXS$NULL が返されます。 関連項目: ユーザーXS$NULL の詳細は、『Oracle Database 2 日でセキュリティ・ガイド』を参照してください。
CURRENT_USERID	権限が現在アクティブになっているデータベース・ユーザーの識別子。

パラメータ	戻り値
DATABASE_ROLE	USERENV ネームスペースで SYS_CONTEXT ファンクションを使用するデータベース・ロール。ロールは、PRIMARY、PHYSICAL STANDBY、LOGICAL STANDBY、SNAPSHOT STANDBY のいずれかです。
DB_DOMAIN	DB_DOMAIN 初期化パラメータで指定されたデータベースのドメインを戻します。
DB_NAME	DB_NAME 初期化パラメータで指定されたデータベース名を戻します。
DB_SUPPLEMENTAL_LOG_LEVEL	サプリメンタル・ロギングが有効になっている場合は、有効なサプリメンタル・ロギング・レベルのリストが含まれている文字列を返します。可能な値は、ALL_COLUMN、FOREIGN_KEY、MINIMAL、PRIMARY_KEY、PROCEDURAL および UNIQUE_INDEX です。サプリメンタル・ロギングが有効でない場合は、NULL を返します。
DB_UNIQUE_NAME	DB_UNIQUE_NAME 初期化パラメータで指定されたデータベース名を戻します。
DBLINK_INFO	データベース・リンク・セッションのソースを戻します。具体的には、次の書式の文字列を戻します。 SOURCE_GLOBAL_NAME=dblink_src_global_name, DBLINK_NAME=dblink_name, SOURCE_AUDIT_SESSIONID=dblink_src_audit_sessionid 説明: ● dblink_src_global_name: ソース・データベースの一意のグローバル名 ● dblink_name: ソース・データベースでのデータベース・リンクの名前 ● dblink_src_audit_sessionid: dblink_name を使用してリモート・データベースへの接続を開始したソース・データベースでのセッションの監査セッション ID
DRAIN_STATUS	現在のセッションのドレイン・ステータスが表示されます。セッションがドレインの候補である場合は、DRAINING を返し、それ以外の場合は NONE を返します。
ENTRYID	現行の監査エントリ番号を戻します。監査エントリ ID の順序は、ファイングレイン監査レコードと通常の監査レコードで共通です。この属性を分散 SQL 文で使用することはできま

パラメータ	戻り値
	せん。正しい監査エントリ識別子は、標準またはファイングレイン監査の監査ハンドラを介してのみ参照できます。
ENTERPRISE_IDENTITY	ユーザーのエンタープライズ全体の識別を戻します。 ● エンタープライズ・ユーザーの場合: Oracle Internet Directory DN。 ● 外部ユーザーの場合: 外部識別(Kerberos プリンシパル名、RADIUS スキーマ名、OS ユーザー名、証明書 DN)。 ● ローカル・ユーザーおよび SYSDBA/SYSOPER ログインの場合: NULL 属性の値はプロキシ方式によって異なります。 ● DN 付きプロキシの場合: クライアントの Oracle Internet Directory DN ● 証明書付きプロキシの場合: クライアントの証明書 DN(外部ユーザー)、Oracle Internet Directory DN(グローバル・ユーザー) ● ユーザー名付きプロキシの場合: クライアントがエンタープライズ・ユーザーの場合は Oracle Internet Directory DN、クライアントがローカル・データベースのユーザーの場合は NULL。 OCI IAM ユーザーの場合: Oracle Cloud Identifier (OCID)。
FG_JOB_ID	DBMS_JOB パッケージを使用して作成されたジョブ内から問い合せた場合: 現在のセッションがクライアントのフォアグラウンド・プロセスで確立されているときには、現在のセッションのジョブ ID が返されます。セッションがフォアグラウンド・プロセスで確立されていない場合は、NULL を戻します。 それ以外の場合は 0 を戻します。
GLOBAL_CONTEXT_MEMORY	コンテキストへのグローバルなアクセスによって、システム・グローバル領域で使用された数値を戻します。
GLOBAL_UID	一元管理ユーザー(CMU)ログインの場合は Active Directory から、エンタープライズ・ユーザー・セキュリティ(EUS)ログインの場合は Oracle Internet Directory からグローバル・ユーザーID (GUID)を戻します。他のすべてのログインについては null を戻します。
HOST	接続中のクライアントのホスト・マシン名を戻します。

パラメータ	戻り値
IDENTIFICATION_TYPE	データベースでユーザーのスキーマを作成した方法を戻します。特に、CREATE/ALTER USER 構文に、IDENTIFIED 句が反映されます。次のリストでは、スキーマの作成中に使用する構文の後に、戻される識別タイプが続きます。 ● IDENTIFIED BY パスワード: LOCAL ● IDENTIFIED EXTERNALLY: EXTERNAL ● IDENTIFIED GLOBALLY: GLOBAL SHARED ● IDENTIFIED GLOBALLY AS DN: GLOBAL PRIVATE ● GLOBAL EXCLUSIVE(排他的なグローバル・ユーザー・マッピング)。 ● GLOBAL SHARED(共有ユーザー・マッピング)。 ● NONE (認証なしでスキーマを作成する場合)
INSTANCE	現行のインスタンスのインスタンス識別番号を戻します。
INSTANCE_NAME	インスタンス名。
IP_ADDRESS	接続中のクライアントのマシンの IP アドレスを戻します。クライアントとサーバーが同じマシン上にあり、接続に IPv6 アドレスが使用されている場合は、::1 が戻されます。
IS_APPLY_SERVER	ロジカル・スタンバイ・データベースで SQL Apply サーバーから問合せが実行された場合は、TRUE を戻します。それ以外の場合は FALSE を戻します。
IS_DG_ROLLING_UPGRADE	DBMS_ROLLING パッケージを使用して開始された、Data Guard 構成内のデータベース・ソフトウェアのローリング・アップグレードがアクティブの場合には、TRUE を戻します。それ以外の場合は FALSE を戻します。
ISDBA	オペレーティング・システムまたはパスワード・ファイルによって、ユーザーが DBA 権限を持っていると認証された場合、TRUE を戻します。
LANG	言語名の略称を戻します。これは、既存の'LANGUAGE'パラメータを短縮したものです。
LANGUAGE	現行のセッションで使用している言語(language)および地域(territory)を、データベース文字セット(character set)も含めて次の書式で戻します。 language_territory.characterset

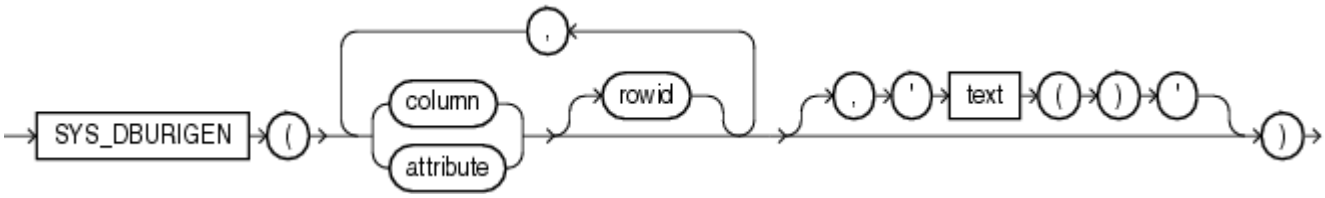
パラメータ	戻り値
LDAP_SERVER_TYPE	構成済 LDAP サーバー・タイプ(OID、AD(Active Directory)、OID_G、OPENLDAP のいずれか)を返します。
MODULE	DBMS_APPLICATION_INFO パッケージまたは OCI を使用して設定されたアプリケーション名(モジュール)を戻します。
NETWORK_PROTOCOL	接続文字列の'PROTOCOL=protocol'の部分で指定された、通信に使用されるネットワーク・プロトコルを戻します。
NLS_CALENDAR	現行のセッションの現行のカレンダを戻します。
NLS_CURRENCY	現行のセッションの通貨を戻します。
NLS_DATE_FORMAT	セッションの日付書式を戻します。
NLS_DATE_LANGUAGE	日付の表示に使用される言語を戻します。
NLS_SORT	BINARY または言語ソート基準を戻します。
NLS_TERRITORY	現行のセッションの地域を戻します。
ORACLE_HOME	Oracle ホーム・ディレクトリのフル・パス名。
OS_USER	データベース・セッションを開始するクライアント・プロセスのオペレーティング・システム・ユーザー名を戻します。
PLATFORM_SLASH	プラットフォームのパス区切り文字として使用されるスラッシュ(「/」)文字。
POLICY_INVOKER	行レベルのセキュリティ(RLS)・ポリシーのファンクションの実行者。
PROXY_ENTERPRISE_IDENTITY	プロキシ・ユーザーがエンタープライズ・ユーザーの場合に、Oracle Internet Directory DN を戻します。
PROXY_USER	SESSION_USER のかわりに現行のセッションを開いたデータベース・ユーザー名を戻します。
PROXY_USERID	SESSION_USER のかわりに現行のセッションを開いたデータベース・ユーザーの ID を戻します。

パラメータ	戻り値
SCHEDULER_JOB	現在のセッションがフォアグラウンド・ジョブまたはバックグラウンド・ジョブに属している場合は、Y を戻します。それ以外の場合は N を戻します。
SERVER_HOST	インスタンスを実行しているマシンのホスト名。
SERVICE_NAME	任意のセッションで接続しているサービスの名前を戻します。
SESSION_DEFAULT_COLLATION	セッションのデフォルトの照合。ALTER SESSION SET DEFAULT_COLLATION ...文によって設定されます。
SESSION_EDITION_ID	セッション・エディションの識別子。
SESSION_EDITION_NAME	セッション・エディションの名前。
SESSION_USER	セッション・ユーザー(ログオンしたユーザー)の名前。これは、Real Application Security セッションがアタッチまたはデタッチされると、データベース・セッション中に変化することがあります。エンタープライズ・ユーザーの場合、スキーマを戻します。その他のユーザーの場合、データベース・ユーザー名を戻します。現在のデータベース・セッションに Real Application Security がアタッチされている場合は、ユーザーXS$NULL が返されます。 関連項目: ユーザーXS$NULL の詳細は、『Oracle Database 2 日でセキュリティ・ガイド』を参照してください。
SESSION_USERID	セッション・ユーザー(ログオンしたユーザー)の ID。
SESSIONID	監査セッション識別子を戻します。この属性を分散 SQL 文で使用することはできません。
SID	セッション ID。
STATEMENTID	文の監査の識別子を戻します。STATEMENTID は、任意のセッションで監査された SQL 文の番号を示します。この属性を分散 SQL 文で使用することはできません。正しい文の監査の識別子は、標準またはファイングレイン監査の監査ハンドラを介してのみ参照できます。
TERMINAL	現行のセッションのクライアントに対するオペレーティング・システムの識別子を戻します。分散 SQL 文では、この属性はローカル・セッションの識別子を戻します。分散環境では、リモートの SELECT に対してのみこのオプションを使用でき、リモートの INSERT、UPDATE または DELETE には使用できません。(このパラメータの戻り値の長さはオペレーティング・システムによって異なります。)

パラメータ	戻り値
UNIFIED_AUDIT_SESSION ID	統合監査または混合モード監査を使用するデータベースに接続中に問い合せた場合、統合監査セッション ID を戻します。 従来の監査を使用するデータベースに接続中に問い合せた場合、null を戻します。

SYS_DBURIGEN

構文



目的

SYS_DBURIGenは、引数として1つ以上の列または属性、およびオプションでROWIDを取り、特定の列または行オブジェクトへのDBUriTypeデータ型のURLを生成します。これによって、データベースからXML文書を検索するためのURLを使用できるようになります。

参照するすべての列または属性は、同じ表内に存在する必要があります。これらは、主キーの役割を果たす必要があります。実際に表の主キーに一致する必要はありませんが、一意の値を参照する必要があります。複数の列を指定すると、最後の列以外のすべての列はデータベースの行を識別し、指定された最後の列は行にある列を識別します。

デフォルトでは、URLはフォーマットされたXML文書を指します。URLにドキュメントのテキストのみを指す場合は、オプションの'text()'を指定します

ノート:

この XML コンテキストでは、小文字の text はキーワードであり、構文のプレースホルダではありません。

列または属性を含む表またはビューが、問合せのコンテキストで指定されるスキーマを持たない場合、Oracle Databaseは、表名またはビュー名をパブリック・シノニムとして解析します。

関連項目:

データベースのDBURITypeデータ型およびXML文書の詳細は、『Oracle XML DB Developer's Guide』を参照してください。

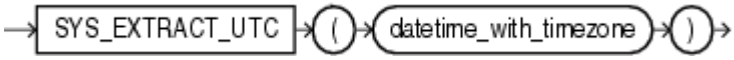
例

次の例では、SYS_DBURIGenファンクションを使用して、サンプル表hr.employeesのemployee_id = 206である行のemail列へのDBUriTypeデータ型のURLを生成します。

```
SELECT SYS_DBURIGEN(employee_id, email)
   FROM employees
   WHERE employee_id = 206;
SYS_DBURIGEN(EMPLOYEE_ID,EMAIL)(URL, SPARE)
--------------------------------------------------------------------
DBURITYPE('/PUBLIC/EMPLOYEES/ROW[EMPLOYEE_ID=''206'']/EMAIL', NULL)
```

SYS_EXTRACT_UTC

構文



目的

SYS_EXTRACT_UTCは、タイムゾーン・オフセットまたはタイムゾーン地域名を含む日時値から協定世界時(UTC)(以前のグリニッジ標準時)を抽出します。タイムゾーンを指定しないと、この日時はセッションのタイムゾーンに関連付けられます。

例

次の例では、指定された日時からUTCを抽出します。

```
SELECT SYS_EXTRACT_UTC(TIMESTAMP '2000-03-28 11:30:00.00 -08:00')
   FROM DUAL;
SYS_EXTRACT_UTC(TIMESTAMP'2000-03-2811:30:00.00-08:00')
-------------------------------------------------------------
28-MAR-00 07.30.00 PM
```

SYS_GUID

構文

目的

SYS_GUIDは、16バイトで構成されたグローバルな一意の識別子(RAW値)を生成して戻します。多くのプラットフォームでは、生成された識別子は、ホスト識別子、ファンクションをコールするプロセスやスレッドのプロセス識別子またはスレッド識別子、およびそのプロセスやスレッドに対する非反復値(バイトの順序)で構成されています。

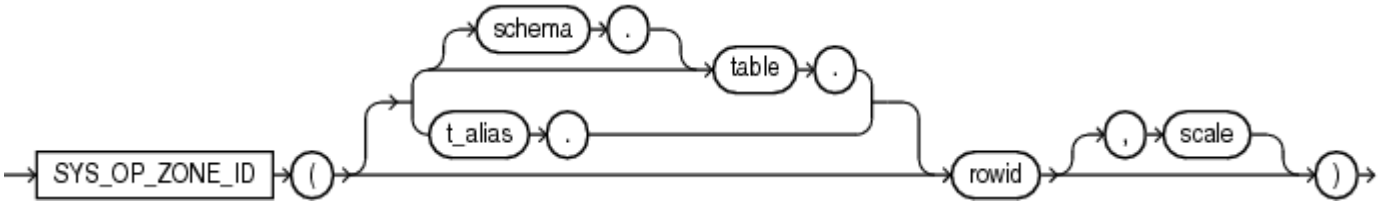
例

次の例では、サンプル表hr.locationsに列を追加後、一意の識別子を各行に挿入し、グローバルな一意識別子の16バイトのRAW値を32文字の16進表記で戻します。

```
ALTER TABLE locations ADD (uid_col RAW(16));
UPDATE locations SET uid_col = SYS_GUID();
SELECT location_id, uid_col FROM locations
   ORDER BY location_id, uid_col;
LOCATION_ID UID_COL
----------- --------------------------------------------------------------------
       1000 09F686761827CF8AE040578CB20B7491
       1100 09F686761828CF8AE040578CB20B7491
       1200 09F686761829CF8AE040578CB20B7491
       1300 09F68676182ACF8AE040578CB20B7491
       1400 09F68676182BCF8AE040578CB20B7491
       1500 09F68676182CCF8AE040578CB20B7491
. . .
```

SYS_OP_ZONE_ID

構文



目的

SYS_OP_ZONE_IDは、ROWIDを引数として、ゾーンIDを戻します。ROWIDは表の行を識別します。ゾーンIDは、行を含むゾーンと呼ばれる連続したディスク・ブロックのセットを識別します。戻り値はNUMBERです。

CREATE MATERIALIZED ZONEMAP文を使用してゾーン・マップを作成する場合、SYS_OP_ZONE_IDファンクションが使用されます。ゾーン・マップの定義する副問合せのSELECTおよびGROUP BY句にSYS_OP_ZONE_IDを指定する必要があります。

rowidの場合、ゾーン・マップのファクト表のROWID疑似列を指定します。

schemaおよびtableを使用してファクト表のスキーマと名前を指定するか、t_aliasを使用してファクト表の表の別名を指定します。これらのパラメータの指定は、ゾーン・マップの定義する副問合せのFROM句によって異なります。

- FROM句でファクト表の表の別名を指定する場合、SYS_OP_ZONE_IDに表の別名(t_alias)も指定する必要があります。
- FROM句でファクト表の表の別名を指定しない場合、tableを使用してファクト表の名前を指定します。他のスキーマ内にファクト表が存在している場合、schema修飾子を使用できます。schemaを省略する場合、ファクト表は自分のスキーマ内にあるとみなされます。FROM句で1つの表(ファクト表)のみを指定する場合、schemaまたはtableを指定する必要はありません。

オプションのscaleパラメータは、ゾーン・マップのスケールを表します。デフォルトでSYS_OP_ZONE_IDは作成されるゾーン・マップのスケールを使用するため、このパラメータを指定する必要はありません。scaleを指定する場合、作成されるゾーン・マップのスケールを照合する必要があります。ゾーン・マップのスケールの指定の詳細は、CREATE MATERIALIZED ZONEMAPのSCALE句を参照してください。

関連項目:

ゾーン・マップの作成の詳細は、CREATE MATERIALIZED ZONEMAPを参照してください。

例

次の例は、ファクト表salesの列time_idを追跡する基本的なゾーン・マップを作成する場合にSYS_OP_ZONE_IDファンクションを使用します。ゾーン・マップのスケールはデフォルト値10です。このため、SYS_OP_ZONE_IDファンクションのデフォルトのスケール値は10になります。

```
CREATE MATERIALIZED ZONEMAP sales_zmap
AS
  SELECT SYS_OP_ZONE_ID(rowid), MIN(time_id), MAX(time_id)
  FROM sales
  GROUP BY SYS_OP_ZONE_ID(rowid);
```

作成されるゾーン・マップのスケールに8が指定されている点を除いて、次の例は前の例と似ています。このため、SYS_OP_ZONE_IDファンクションのデフォルトのスケール値は8になります。

```
CREATE MATERIALIZED ZONEMAP sales_zmap
SCALE 8
AS
  SELECT SYS_OP_ZONE_ID(rowid), MIN(time_id), MAX(time_id)
  FROM sales
  GROUP BY SYS_OP_ZONE_ID(rowid);
```

作成されるゾーン・マップのスケールにSYS_OP_ZONE_IDファンクションで指定されているscale引数12と一致しない8が指定されているため、次の例はエラーを戻します。

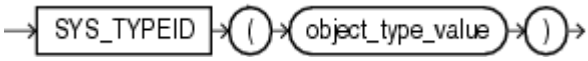
```
CREATE MATERIALIZED ZONEMAP sales_zmap
SCALE 8
AS
  SELECT SYS_OP_ZONE_ID(rowid,12), MIN(time_id), MAX(time_id)
  FROM sales
  GROUP BY SYS_OP_ZONE_ID(rowid,12);
```

次の例は、結合ゾーン・マップを作成します。ファクト表がsalesで、ディメンション表がproductsとcustomersです。表の別名sがFROM句のファクト表に指定されているため、表の別名sがSYS_OP_ZONE_IDファンクションにも指定されます。

```
CREATE MATERIALIZED ZONEMAP sales_zmap
AS
  SELECT SYS_OP_ZONE_ID(s.rowid),
         MIN(prod_category), MAX(prod_category),
         MIN(country_id), MAX(country_id)
  FROM sales s, products p, customers c
  WHERE s.prod_id = p.prod_id(+) AND
        s.cust_id = c.cust_id(+)
  GROUP BY SYS_OP_ZONE_ID(s.rowid);
```

SYS_TYPEID

構文



目的

SYS_TYPEIDは、オペランドで最も指定される型の型IDを戻します。この値は、主に、置換可能な列の基礎となる型判別式の列を識別するために使用されます。たとえば、型判別式の列に索引を構築するために、SYS_TYPEIDによって戻される値を使用できます。

このファンクションは、オブジェクト型のオペランドのみで使用してください。すべての最終ルート・オブジェクト型(型階層に属さない最終型)は、NULLの型IDを持ちます。Oracle Databaseは、型階層に属するすべての型に、NULL以外の一意の型IDを割り当てます。

関連項目:

型IDの詳細は、『Oracle Databaseオブジェクト・リレーショナル開発者ガイド』を参照してください。

例

次の例では、「置換可能な表および列のサンプル」で作成された表personsおよびbooksを使用します。最初の問合せは、persons表に格納されたオブジェクト・インスタンスの最も指定される型を戻します。

```
SELECT name, SYS_TYPEID(VALUE(p)) "Type_id" FROM persons p;
NAME                     Type_id
------------------------ --------------------------------
Bob                      01
Joe                      02
Tim                      03
```

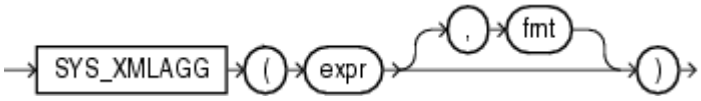
次の問合せは、books表に格納された作者の最も指定される型を戻します。

```
SELECT b.title, b.author.name, SYS_TYPEID(author)
   "Type_ID" FROM books b;
TITLE                    AUTHOR.NAME          Type_ID
------------------------ -------------------- ------------------
An Autobiography         Bob                  01
Business Rules           Joe                  02
Mixing School and Work   Tim                  03
```

SYS_TYPEIDファンクションを使用すると、表の型判別式の列に索引を作成できます。例は、「置換可能な列の索引の作成: 例」を参照してください。

SYS_XMLAGG

構文



目的

SYS_XMLAggは、exprによって表されるすべてのXML文書またはXMLフラグメントを集約し、単一のXML文書を生成します。このファンクションは、デフォルト名ROWSETの新しい囲み要素を追加します。XML文書を別の方法でフォーマットする場合は、XMLFormatオブジェクトのインスタンスであるfmtを指定します。

関連項目:

SYS_XMLAggの結果をフォーマットするためのXMLFormat型の属性の使用方法は、「SYS_XMLGEN」および「XML書式モデル」を参照してください。

例

次の例では、SYS_XMLGENファンクションを使用して、従業員名の最初の文字がRであるサンプル表employeesの各行に対して、XML文書を生成した後、デフォルトの囲み要素ROWSETの1つのXML文書にすべての行を集約します。

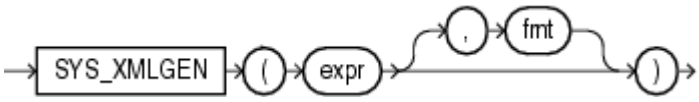
```
SELECT SYS_XMLAGG(SYS_XMLGEN(last_name)) XMLAGG
   FROM employees
   WHERE last_name LIKE 'R%'
   ORDER BY xmlagg;
XMLAGG
--------------------------------------------------------------------------------
<?xml version="1.0"?>
<ROWSET>
<LAST_NAME>Rajs</LAST_NAME>
<LAST_NAME>Raphaely</LAST_NAME>
<LAST_NAME>Rogers</LAST_NAME>
<LAST_NAME>Russell</LAST_NAME>
</ROWSET>
```

SYS_XMLGEN

ノート:

SYS_XMLGen ファンクションは、非推奨です。これは、下位互換性を保つためにのみサポートされています。そのかわりに、SQL/XML 生成ファンクションを使用してください。詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

構文



目的

SYS_XMLGENは、データベースの特定の行および列を評価する式を取り、XML文書を含むXMLType型のインスタンスを戻します。exprは、スカラー値、ユーザー定義型またはXMLTypeインスタンスです。

- exprがスカラー値である場合、ファンクションはスカラー値を含むXML要素を戻します。
- exprが型である場合、ファンクションはXML要素へユーザー定義型の属性をマップします。
- exprがXMLTypeインスタンスである場合、ファンクションはデフォルトのタグ名がROWであるXML要素でドキュメントを囲みます。

デフォルトでは、XML文書の要素はexprの要素と一致します。たとえば、exprが列名に変換される場合、XMLの囲み要素は同じ列名になります。XML文書を別の方法でフォーマットする場合は、XMLFormatオブジェクトのインスタンスであるfmtを指定します。

関連項目:

XMLFormat型の詳細およびSYS_XMLGenの結果をフォーマットするための属性の使用方法は、「XML書式モデル」を参照してください。

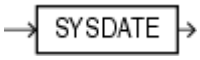
例

次の例では、サンプル表oe.employeesからemployee_id値が205の従業員の電子メールIDを検出し、EMAIL要素を持つXML文書を含むXMLTypeのインスタンスを生成します。

```
SELECT SYS_XMLGEN(email)
   FROM employees
   WHERE employee_id = 205;
SYS_XMLGEN(EMAIL)
--------------------------------------------------------------------
<?xml version="1.0"?>
<EMAIL>SHIGGINS</EMAIL>
```

SYSDATE

構文



目的

SYSDATEは、データベース・サーバーが存在するオペレーティング・システムの現在の日付と時刻のセットを戻します。戻り値のデータ型はDATEです。戻り値の書式は、NLS_DATE_FORMAT初期化パラメータの値によって異なります。この関数に引数を指定する必要はありません。分散SQL文では、このファンクションはローカル・データベースのオペレーティング・システムの日付と時刻のセットを戻します。このファンクションはCHECK制約の条件の中では使用できません。

ノート:

FIXED_DATE 初期化パラメータを使用して、SYSDATE が常に現在の日時のかわりに戻す変わらない日時を設定できます。このパラメータは、主にテストに役立ちます。FIXED_DATE 初期化パラメータの詳細は、Oracle Database Reference を参照してください。

例

次の例では、オペレーティング・システムの現在の日付および時刻を戻します。

```
SELECT TO_CHAR
    (SYSDATE, 'MM-DD-YYYY HH24:MI:SS') "NOW"
     FROM DUAL;
NOW
-------------------
04-13-2001 09:45:51
```

SYSTIMESTAMP

構文

目的

SYSTIMESTAMPは、データベースが存在するシステムの、秒の小数部とタイムゾーンを含む日付を戻します。戻り値の型は、TIMESTAMP WITH TIME ZONEです。

例

次の例では、システムのタイムスタンプを戻します。

```
SELECT SYSTIMESTAMP FROM DUAL;
SYSTIMESTAMP
---------------------------------------------------------------------
28-MAR-00 12.38.55.538741 PM -08:00
```

次の例では、秒の小数部を明示的に指定する方法を示します。

```
SELECT TO_CHAR(SYSTIMESTAMP, 'SSSSS.FF') FROM DUAL;
TO_CHAR(SYSTIME
---------------
55615.449255
```

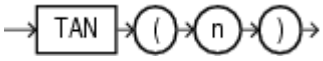
次の例では、指定したタイムゾーンでの現在のタイムスタンプを戻します。

```
SELECT SYSTIMESTAMP AT TIME ZONE 'UTC' FROM DUAL;

SYSTIMESTAMPATTIMEZONE'UTC'
---------------------------------------------------------------------------
08-07-21 20:39:52,743557 UTC
```

この例の出力書式は、セッションのNLS_TIMESTAMP_TZ_FORMATによって異なります。

TAN

構文



目的

TANは、n(ラジアンで表された角度)のタンジェントを戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数がBINARY_FLOATの場合、このファンクションはBINARY_DOUBLEを戻します。それ以外の場合、引数と同じ数値データ型を戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

例

次の例では、135度のタンジェントを戻します。

```
SELECT TAN(135 * 3.14159265359/180)
   "Tangent of 135 degrees"  FROM DUAL;
Tangent of 135 degrees
----------------------
                    - 1
```

TANH

構文

目的

TANHは、nの双曲線タンジェントを戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数がBINARY_FLOATの場合、このファンクションはBINARY_DOUBLEを戻します。それ以外の場合、引数と同じ数値データ型を戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

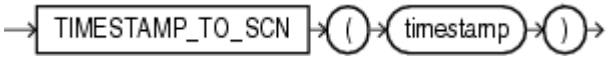
例

次の例では、5の双曲線タンジェントを戻します。

```
SELECT TANH(.5) "Hyperbolic tangent of .5"
   FROM DUAL;
Hyperbolic tangent of .5
------------------------
               .462117157
```

TIMESTAMP_TO_SCN

構文



目的

TIMESTAMP_TO_SCNは、引数としてタイムスタンプ値を取り、そのタイムスタンプに関連付けられたシステム変更番号(SCN)の概数を戻します。戻り値のデータ型はNUMBERです。このファンクションは、特定のタイムスタンプに関連付けられたSCNを調べる場合に有効です。

ノート:

SCN と SCN 生成時のタイムスタンプの関連は、一定期間データベースで記憶されます。この期間は、最大で自動調整された UNDO 保存期間(データベースが自動 UNDO 管理モードで実行されている場合)およびデータベース内のすべてのフラッシュバック・アーカイブの保存時間となりますが、120 時間以上になります。関連が不要となるまでの経過時間には、データベースが開かれているときの時間のみが加算されます。TIMESTAMP_TO_SCN の引数に対して指定されたタイムスタンプが古すぎる場合は、エラーが戻されます。

関連項目:

SCNをタイムスタンプへ変換する方法については、「SCN_TO_TIMESTAMP」を参照してください。

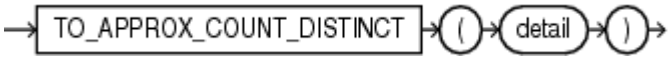
例

次の例では、行をoe.orders表に挿入し、TIMESTAMP_TO_SCNを使用して、挿入操作のシステム変更番号を判断します。(実際に戻されるSCNは、システムごとに異なります。)

```
INSERT INTO orders (order_id, order_date, customer_id, order_total)
   VALUES (5000, SYSTIMESTAMP, 188, 2345);
1 row created.
COMMIT;
Commit complete.
SELECT TIMESTAMP_TO_SCN(order_date) FROM orders
   WHERE order_id = 5000;
TIMESTAMP_TO_SCN(ORDER_DATE)
----------------------------
                      574100
```

TO_APPROX_COUNT_DISTINCT

構文



目的

TO_APPROX_COUNT_DISTINCTでは、その入力として、個別値の近似カウントに関する情報を含む詳細を取得し、これをNUMBER値に変換します。

detailには、APPROX_COUNT_DISTINCT_DETAILファンクションまたはAPPROX_COUNT_DISTINCT_AGGファンクションで作成されたBLOB型の詳細を指定します。

関連項目:

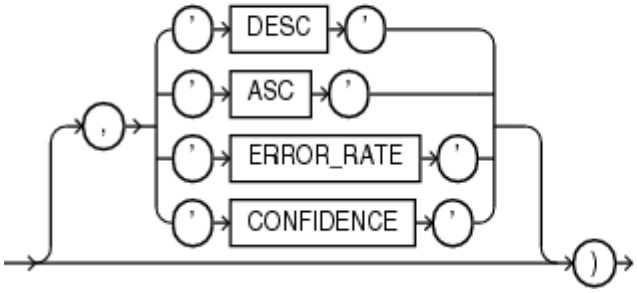
- APPROX_COUNT_DISTINCT_DETAIL
- TO_APPROX_COUNT_DISTINCT

例

APPROX_COUNT_DISTINCT_DETAILファンクションおよびAPPROX_COUNT_DISTINCT_AGGファンクションと組み合せてTO_APPROX_COUNT_DISTINCTファンクションを使用する例は、「TO_APPROX_COUNT_DISTINCT: 例」を参照してください。

TO_APPROX_PERCENTILE

構文



目的

TO_APPROX_PERCENTILEでは、その入力として、近似パーセンタイル情報を含む詳細、パーセンタイル値およびソート指定を取得し、そのソート指定に従ってそのパーセンタイル値に該当する近似値を戻します。

detailには、APPROX_PERCENTILE_DETAILファンクションまたはAPPROX_PERCENTLE_AGGファンクションで作成されたBLOB型の詳細を指定します。

exprには、0から1までの数値と評価されるパーセンタイル値を指定します。ERROR_RATE句またはCONFIDENCE句を指定した場合、パーセンタイル値は適用されません。この場合、exprには、NULLまたは0から1までの数値を指定する必要があります。ただし、値は無視されます。

datatypeには、詳細の近似パーセンタイル情報のデータ型を指定します。これは、詳細を生成したAPPROX_PERCENTILE_DETAILファンクションに指定した式のデータ型です。有効なデータ型は、NUMBER、BINARY_FLOAT、BINARY_DOUBLE、DATE、TIMESTAMP、INTERVAL YEAR TO MONTHおよびINTERVAL DAY TO SECONDです。

DESC | ASC

補間のためにソート指定を指定します。降順のソート順の場合はDESC、昇順のソート順の場合はASCを指定します。デフォルトはASCです。

ERROR_RATE | CONFIDENCE

これらの句により、詳細のパーセンタイル評価の精度を確認できます。これらの句のいずれかを指定すると、おおよその補間された値が戻されるかわりに、次のいずれかの値を表す0から1までの小数値(それらの値を含む)が戻されます。

- ERROR_RATEを指定すると、戻り値は、詳細のパーセンタイル評価のエラー率を表します。
- CONFIDENCEを指定すると、戻り値は、ERROR_RATEを指定したときに戻されるエラー率の信頼水準を表します。

ERROR_RATEまたはCONFIDENCEを指定した場合、パーセンタイル値exprは無視されます。

関連項目:

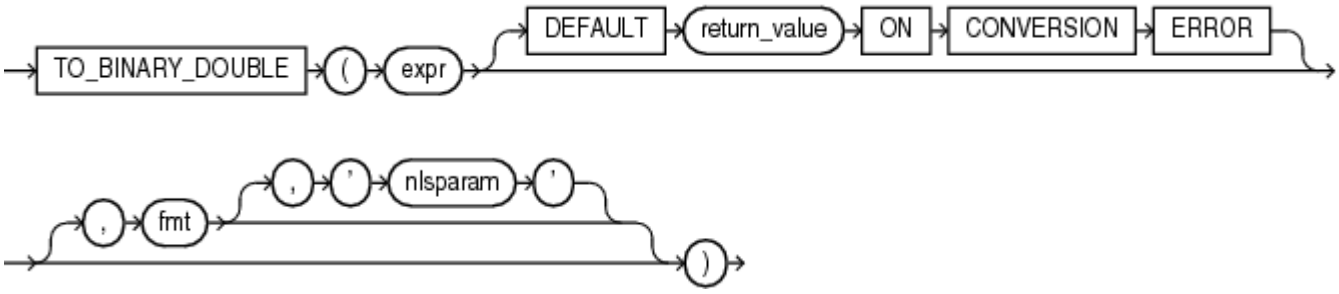
- APPROX_PERCENTILE_DETAIL
- APPROX_PERCENTILE_AGG

例

APPROX_PERCENTILE_DETAILファンクションおよびAPPROX_PERCENTILE_AGGファンクションと組み合せて
TO_APPROX_PERCENTILEファンクションを使用する例は、「APPROX_PERCENTILE_AGG: 例」を参照してください。

TO_BINARY_DOUBLE

構文



目的

TO_BINARY_DOUBLEは、exprを倍精度浮動小数点数に変換します。

- exprは、CHAR、VARCHAR2、NCHARまたはNVARCHAR2の型の文字列、NUMBER、BINARY_FLOATまたはBINARY_DOUBLEの型の数値、あるいはNULLと評価される任意の式にすることができます。exprがBINARY_DOUBLEの場合、このファンクションはexprを戻します。exprがNULLと評価されると、関数はNULLを返します。それ以外の場合は、exprがBINARY_DOUBLE値に変換されます。
- オプションのDEFAULT return_value ON CONVERSION ERROR句により、exprからBINARY_DOUBLEへの変換中にエラーが発生した場合にこのファンクションで戻される値を指定できます。exprを評価するときにエラーが発生した場合、この句による影響はありません。return_valueは式またはバインド変数にすることができ、CHAR、VARCHAR2、NCHARまたはNVARCHAR2の型の文字列、NUMBER、BINARY_FLOATまたはBINARY_DOUBLEの型の数値、あるいはNULLと評価される必要があります。exprをBINARY_DOUBLEに変換する場合と同じ方法で、return_valueがBINARY_DOUBLEに変換されます。return_valueをBINARY_DOUBLEに変換できない場合は、エラーが戻されます。
- オプションの'fmt'引数および'nlsparam'引数の用途は、TO_NUMBERファンクションと同じです。これらの引数を指定した場合は、exprとreturn_value (指定した場合)は、それぞれ文字列またはNULLにする必要があります。いずれかが文字列である場合は、fmt引数とnlsparam引数が使用されて、文字列がBINARY_DOUBLE値に変換されます。

exprまたはreturn_valueが次の文字列に評価される場合、これらは次のように変換されます。

- 文字列'INF'(大/小文字は区別されない)は、正の無限大に変換されます。
- 文字列'-INF'(大/小文字は区別されない)は、負の無限大に変換されます。
- 文字列'NaN'(大/小文字は区別されない)は、NaN(非数値)に変換されます。

expr文字列には、浮動小数点数の書式要素(F、f、Dまたはd)は使用できません。

文字列またはNUMBERからBINARY_DOUBLEへの変換は、正確に行われない場合があります。これは、NUMBERおよび文字列型では10進精度、BINARY_DOUBLEでは2進精度を使用して数値を表現するためです。

BINARY_FLOATからBINARY_DOUBLEへの変換は正確に行われます。

関連項目:

「TO_CHAR (数値)」および「浮動小数点数」を参照してください。

例

次の例では、それぞれ異なる数値データ型の3つの列を持つ次の表を使用します。

```
CREATE TABLE float_point_demo
  (dec_num NUMBER(10,2), bin_double BINARY_DOUBLE, bin_float BINARY_FLOAT);
INSERT INTO float_point_demo
  VALUES (1234.56,1234.56,1234.56);
SELECT * FROM float_point_demo;
   DEC_NUM BIN_DOUBLE  BIN_FLOAT
---------- ---------- ----------
   1234.56 1.235E+003 1.235E+003
```

次の例では、NUMBERデータ型の値をBINARY_DOUBLEデータ型の値に変換します。

```
SELECT dec_num, TO_BINARY_DOUBLE(dec_num)
  FROM float_point_demo;
   DEC_NUM TO_BINARY_DOUBLE(DEC_NUM)
---------- ------------------------
   1234.56                1.235E+003
```

次の例では、dec_num列およびbin_double列から抽出されたダンプ情報を比較します。

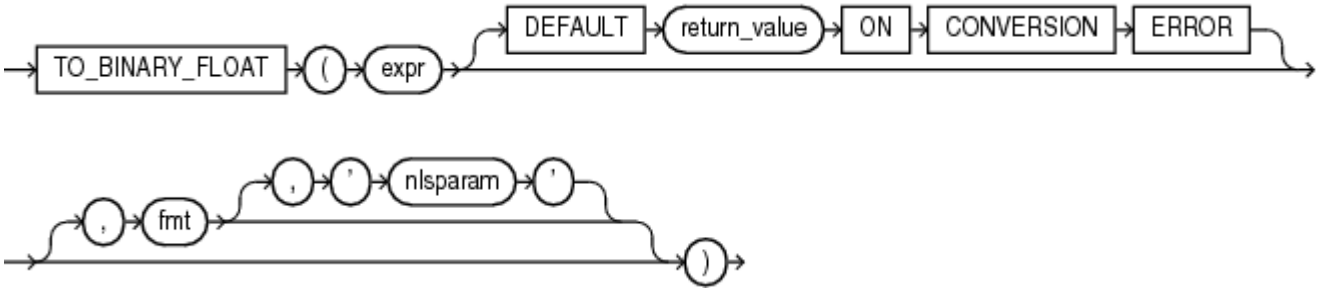
```
SELECT DUMP(dec_num) "Decimal",
   DUMP(bin_double) "Double"
   FROM float_point_demo;
Decimal                    Double
-------------------------- ----------------------------------------------
Typ=2 Len=4: 194,13,35,57   Typ=101 Len=8: 192,147,74,61,112,163,215,10
```

次の例では、指定した式をBINARY_DOUBLE値に変換できないため、デフォルト値の0が戻されます。

```
SELECT TO_BINARY_DOUBLE('2oo' DEFAULT 0 ON CONVERSION ERROR) "Value"
  FROM DUAL;
     Value
----------
         0
```

TO_BINARY_FLOAT

構文



目的

TO_BINARY_FLOATは、exprを単精度浮動小数点数に変換します。

- exprは、CHAR、VARCHAR2、NCHARまたはNVARCHAR2の型の文字列、NUMBER、BINARY_FLOATまたはBINARY_DOUBLEの型の数値、あるいはNULLと評価される任意の式にすることができます。exprがBINARY_FLOATの場合、このファンクションはexprを戻します。exprがNULLと評価されると、関数はNULLを返します。それ以外の場合は、exprがBINARY_FLOAT値に変換されます。
- オプションのDEFAULT return_value ON CONVERSION ERROR句により、exprからBINARY_FLOATへの変換中にエラーが発生した場合にこのファンクションで戻される値を指定できます。exprを評価するときにエラーが発生した場合、この句による影響はありません。return_valueは式またはバインド変数にすることができ、CHAR、VARCHAR2、NCHARまたはNVARCHAR2の型の文字列、NUMBER、BINARY_FLOATまたはBINARY_DOUBLEの型の数値、あるいはNULLと評価される必要があります。exprをBINARY_FLOATに変換する場合と同じ方法で、return_valueがBINARY_FLOATに変換されます。return_valueをBINARY_FLOATに変換できない場合は、エラーが戻されます。
- オプションの'fmt'引数および'nlsparam'引数の用途は、TO_NUMBERファンクションと同じです。これらの引数を指定した場合は、exprとreturn_value (指定した場合)は、それぞれ文字列またはNULLにする必要があります。いずれかが文字列である場合は、fmt引数とnlsparam引数が使用されて、文字列がBINARY_FLOAT値に変換されます。

exprまたはreturn_valueが次の文字列に評価される場合、これらは次のように変換されます。

- 文字列'INF'(大/小文字は区別されない)は、正の無限大に変換されます。
- 文字列'-INF'(大/小文字は区別されない)は、負の無限大に変換されます。
- 文字列'NaN'(大/小文字は区別されない)は、NaN(非数値)に変換されます。

expr文字列には、浮動小数点数の書式要素(F、f、Dまたはd)は使用できません。

文字列またはNUMBERからBINARY_FLOATへの変換は、正確に行われない場合があります。これは、NUMBERおよび文字列型では10進精度、BINARY_FLOATでは2進精度を使用して数値を表現するためです。

BINARY_DOUBLE値に、BINARY_FLOATがサポートする数を超える精度ビットが使用されている場合、BINARY_DOUBLEからBINARY_FLOATへの変換は正確に行われません。

関連項目:

「TO_CHAR (数値)」および「浮動小数点数」を参照してください。

例

次の例では、「TO_BINARY_DOUBLE」で作成したfloat_point_demo表を使用して、NUMBERデータ型の値をBINARY_FLOATデータ型の値に変換します。

```
SELECT dec_num, TO_BINARY_FLOAT(dec_num)
  FROM float_point_demo;
   DEC_NUM TO_BINARY_FLOAT(DEC_NUM)
---------- ------------------------
   1234.56                1.235E+003
```

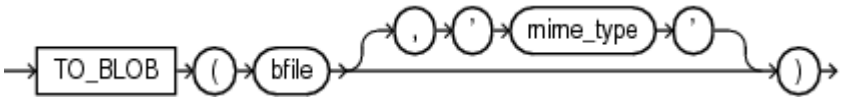
次の例では、指定した式をBINARY_FLOAT値に変換できないため、デフォルト値の0が戻されます。

```
SELECT TO_BINARY_FLOAT('2oo' DEFAULT 0 ON CONVERSION ERROR) "Value"
  FROM DUAL;
     Value
----------
         0
```

TO_BLOB (bfile)

構文

to_blob_bfile::=



目的

TO_BLOB (bfile)は、BFILE値をBLOB値に変換します。

mime_typeには、このファンクションで戻されるBLOB値に設定されるMIMEタイプを指定します。mime_typeを省略すると、BLOB値にMIMEタイプが設定されません。

例

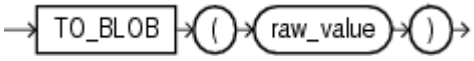
次の例では、表media_tabのBFILE列の値media_colのBLOBを戻します。これにより、結果のBLOBでMIMEタイプがJPEGに設定されます。

```
SELECT TO_BLOB(media_col, 'JPEG') FROM media_tab;
```

TO_BLOB (raw)

構文

to_blob::=



目的

TO_BLOB (raw)は、LONG RAWおよびRAW値をBLOB値に変換します。

PL/SQLパッケージ内からは、TO_BLOB (raw)を使用してRAW値およびBLOB値をBLOBに変換できます。

例

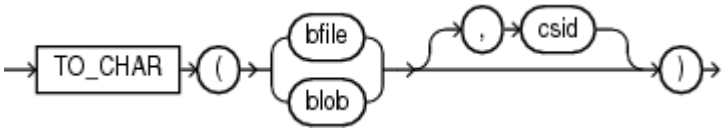
次の例は、RAW列の値としてBLOBを戻します。

```
SELECT TO_BLOB(raw_column) blob FROM raw_table;
BLOB
------------------------
00AADD343CDBBD
```

TO_CHAR (bfile|blob)

構文

to_char_bfile_blob::=



目的

TO_CHAR (bfile|blob)は、BFILEデータまたはBLOBデータをデータベース文字セットに変換します。戻り値は常にVARCHAR2です。戻される値が大きすぎてVARCHAR2データ型に収まらない場合は、データが切り捨てられます。

csidには、BFILEまたはBLOBデータの文字セットのIDを指定します。BFILEまたはBLOBデータの文字セットがデータベース文字セットの場合、csidに0の値を指定することも、csid自体を省略することもできます。

関連項目:

このファンクションの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

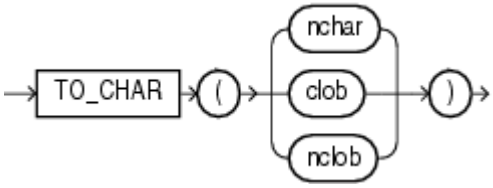
次の例では、その入力として、ID 873の文字セットを使用する、表media_tabのBFILE列のmedia_colを取得します。この例では、データベース文字セットを使用するVARCHAR2値が戻されます。

```
SELECT TO_CHAR(media_col, 873) FROM media_tab;
```

TO_CHAR (文字)

構文

to_char_char::=



目的

TO_CHAR(文字)は、NCHAR、NVARCHAR2、CLOBまたはNCLOBデータをデータベース文字セットに変換します。戻り値は常にVARCHAR2です。

このファンクションを使用して文字LOBをデータベース文字セットに変換する際、変換するLOB値がターゲットのデータ型よりも大きいと、データベースからエラーが返されます。

関連項目:

このファンクションの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、単純な文字列を文字データとして解析します。

```
SELECT TO_CHAR('01110') FROM DUAL;
TO_CH
-----
01110
```

この例と、「TO_CHAR(数値)」の最初の例を比較してください。

次の例では、pm.print_media表のCLOBデータをデータベース文字セットに変換します。

```
SELECT TO_CHAR(ad_sourcetext) FROM print_media
      WHERE product_id = 2268;
TO_CHAR(AD_SOURCETEXT)
--------------------------------------------------------------------------------
******************************
TIGER2 2268...Standard Hayes Compatible Modem
Product ID: 2268
The #1 selling modem in the universe! Tiger2's modem includes call management
and Internet voicing. Make real-time full duplex phone calls at the same time
you're online.
**********************************
```

TO_CHAR (character)ファンクション: 例

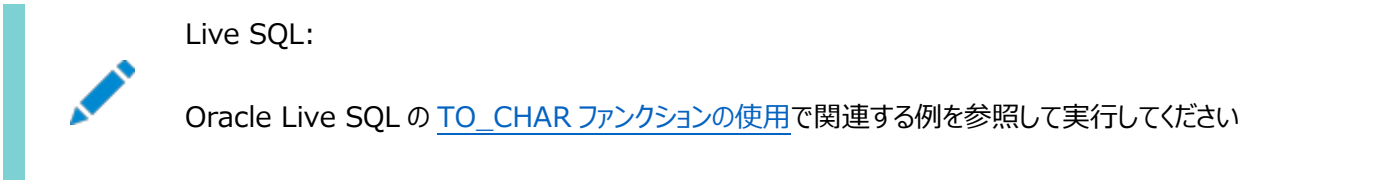
次の文は、empl_tempという名前の表を作成し、従業員の詳細を移入します。

```
CREATE TABLE empl_temp
  (
     employee_id NUMBER(6),
     first_name  VARCHAR2(20),
```

```
    last_name   VARCHAR2(25),
    email       VARCHAR2(25),
    hire_date   DATE DEFAULT SYSDATE,
    job_id      VARCHAR2(10),
    clob_column CLOB
  );
INSERT INTO empl_temp
VALUES(111,'John','Doe','example.com','10-JAN-2015','1001','Experienced Employee');
INSERT INTO empl_temp
VALUES(112,'John','Smith','example.com','12-JAN-2015','1002','Junior Employee');
INSERT INTO empl_temp
VALUES(113,'Johnnie','Smith','example.com','12-JAN-2014','1002','Mid-Career
Employee');
INSERT INTO empl_temp
VALUES(115,'Jane','Doe','example.com','15-JAN-2015','1005','Executive Employee');
```

次の文は、CLOBデータをデータベース文字セットに変換します。

```
SELECT To_char(clob_column) "CLOB_TO_CHAR"
FROM   empl_temp
WHERE  employee_id IN ( 111, 112, 115 );
CLOB_TO_CHAR
--------------------
Experienced Employee
Junior Employee
Executive Employee
```



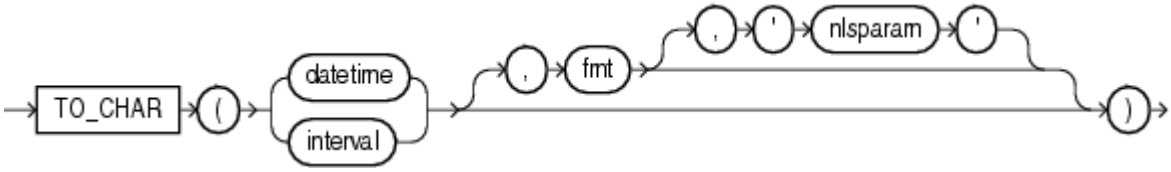
Live SQL:

Oracle Live SQL の TO_CHAR ファンクションの使用で関連する例を参照して実行してください

TO_CHAR (日時)

構文

to_char_date::=



目的

TO_CHAR(日時)は、DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONE、TIMESTAMP WITH LOCAL TIME ZONE、INTERVAL DAY TO SECONDまたはINTERVAL YEAR TO MONTHデータ型の日時値または期間値を日付書式fmtで指定された書式のVARCHAR2データ型の値に変換します。fmtを省略すると、次のように、dateはVARCHAR2値に変換されます。

- DATE値は、デフォルトの日付書式の値に変換されます。
- TIMESTAMPおよびTIMESTAMP WITH LOCAL TIME ZONE値は、デフォルトのタイムスタンプ書式の値に変換されます。
- TIMESTAMP WITH TIME ZONE値は、タイムゾーン書式のデフォルトのタイムスタンプの値に変換されます。
- 期間値は期間リテラルを数値で表現したものに変換されます。

日時書式の詳細は、「書式モデル」を参照してください。

'nlsparam'引数には、月と日の名前および略称が戻される言語を指定します。この引数は、次の書式で指定します。

```
'NLS_DATE_LANGUAGE = language'
```

'nlsparam'を指定しないと、このファンクションはセッションのデフォルト日付言語を使用します。

関連項目:

データ変換のセキュリティ上の考慮事項

このファンクションは任意のXMLファンクションと組み合せて使用できますが、生成される日付の書式は、XMLスキーマの標準書式ではなく、データベースの書式になります。

関連項目:

- XMLの日付およびタイムスタンプの書式設定とその例については、『Oracle XML DB開発者ガイド』を参照してください。
- XMLファンクションのリストは、「XMLファンクション」を参照してください。
- このファンクションの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例で使用する表は、次のとおりです。

```
CREATE TABLE date_tab (
   ts_col      TIMESTAMP,
   tsltz_col   TIMESTAMP WITH LOCAL TIME ZONE,
   tstz_col    TIMESTAMP WITH TIME ZONE);
```

次の例では、TO_CHARを別のTIMESTAMPデータ型に適用した結果を示します。TIMESTAMP WITH LOCAL TIME ZONE列の結果は、セッションのタイムゾーンを識別します。これに対して、TIMESTAMPおよびTIMESTAMP WITH TIME ZONE列の結果は、セッションのタイムゾーンを識別しません。

```
ALTER SESSION SET TIME_ZONE = '-8:00';
INSERT INTO date_tab VALUES (
   TIMESTAMP'1999-12-01 10:00:00',
   TIMESTAMP'1999-12-01 10:00:00',
   TIMESTAMP'1999-12-01 10:00:00');
INSERT INTO date_tab VALUES (
   TIMESTAMP'1999-12-02 10:00:00 -8:00',
   TIMESTAMP'1999-12-02 10:00:00 -8:00',
   TIMESTAMP'1999-12-02 10:00:00 -8:00');
SELECT TO_CHAR(ts_col, 'DD-MON-YYYY HH24:MI:SSxFF') AS ts_date,
   TO_CHAR(tstz_col, 'DD-MON-YYYY HH24:MI:SSxFF TZH:TZM') AS tstz_date
   FROM date_tab
   ORDER BY ts_date, tstz_date;

TS_DATE                        TSTZ_DATE
------------------------------ ------------------------------------
01-DEC-1999 10:00:00.000000    01-DEC-1999 10:00:00.000000 -08:00
02-DEC-1999 10:00:00.000000    02-DEC-1999 10:00:00.000000 -08:00
SELECT SESSIONTIMEZONE,
   TO_CHAR(tsltz_col, 'DD-MON-YYYY HH24:MI:SSxFF') AS tsltz
   FROM date_tab
   ORDER BY sessiontimezone, tsltz;
SESSIONTIM TSLTZ
---------- ------------------------------
-08:00     01-DEC-1999 10:00:00.000000
-08:00     02-DEC-1999 10:00:00.000000
ALTER SESSION SET TIME_ZONE = '-5:00';
SELECT TO_CHAR(ts_col, 'DD-MON-YYYY HH24:MI:SSxFF') AS ts_col,
   TO_CHAR(tstz_col, 'DD-MON-YYYY HH24:MI:SSxFF TZH:TZM') AS tstz_col
   FROM date_tab
   ORDER BY ts_col, tstz_col;

TS_COL                         TSTZ_COL
------------------------------ ------------------------------------
01-DEC-1999 10:00:00.000000    01-DEC-1999 10:00:00.000000 -08:00
02-DEC-1999 10:00:00.000000    02-DEC-1999 10:00:00.000000 -08:00
SELECT SESSIONTIMEZONE,
TO_CHAR(tsltz_col, 'DD-MON-YYYY HH24:MI:SSxFF') AS tsltz_col
   FROM date_tab
   ORDER BY sessiontimezone, tsltz_col;
  2    3    4
SESSIONTIM TSLTZ_COL
---------- ------------------------------
-05:00     01-DEC-1999 13:00:00.000000
-05:00     02-DEC-1999 13:00:00.000000
```

次の例では、期間リテラルをテキスト・リテラルに変換します。

```
SELECT TO_CHAR(INTERVAL '123-2' YEAR(3) TO MONTH) FROM DUAL;
TO_CHAR
-------
+123-02
```

日付および数値の書式を設定するためのTO_CHARの使用：例

次の文は、日付値をTO_CHARファンクションで指定された形式に変換します。

```
WITH dates AS (
  SELECT date'2015-01-01' d FROM dual union
  SELECT date'2015-01-10' d FROM dual union
  SELECT date'2015-02-01' d FROM dual
)
SELECT d "Original Date",
       to_char(d, 'dd-mm-yyyy') "Day-Month-Year",
       to_char(d, 'hh24:mi') "Time in 24-hr format",
       to_char(d, 'iw-iyyy') "ISO Year and Week of Year"
FROM dates;
```

次の文は、日付およびタイムスタンプの値をTO_CHARファンクションで指定された形式に変換します。

```
WITH dates AS (
  SELECT date'2015-01-01' d FROM dual union
  SELECT date'2015-01-10' d FROM dual union
  SELECT date'2015-02-01' d FROM dual union
  SELECT timestamp'2015-03-03 23:44:32' d FROM dual union
  SELECT timestamp'2015-04-11 12:34:56' d FROM dual
)
SELECT d "Original Date",
       to_char(d, 'dd-mm-yyyy') "Day-Month-Year",
       to_char(d, 'hh24:mi') "Time in 24-hr format",
       to_char(d, 'iw-iyyy') "ISO Year and Week of Year",
       to_char(d, 'Month') "Month Name",
       to_char(d, 'Year') "Year"
FROM dates;
```

次の文は、入力日時式から、EXTRACTファンクションで指定された日時フィールドを抽出します。

```
WITH dates AS (
  SELECT date'2015-01-01' d FROM dual union
  SELECT date'2015-01-10' d FROM dual union
  SELECT date'2015-02-01' d FROM dual union
  SELECT timestamp'2015-03-03 23:44:32' d FROM dual union
  SELECT timestamp'2015-04-11 12:34:56' d FROM dual
)
SELECT extract(minute from d) minutes,
       extract(hour from d) hours,
       extract(day from d) days,
       extract(month from d) months,
       extract(year from d) years
FROM dates;
```

次の文は、TO_CHARファンクションで指定された書式に従って入力番号を表示します。

```
WITH nums AS (
  SELECT 10 n FROM dual union
  SELECT 9.99 n FROM dual union
  SELECT 1000000 n FROM dual  --one million
)
SELECT n "Input Number N",
       to_char(n),
       to_char(n, '9,999,999.99') "Number with Commas",
       to_char(n, '0,000,000.000') "Zero-padded Number",
       to_char(n, '9.9EEEE') "Scientific Notation"
FROM nums;
```

次の文は、TO_CHARファンクションで指定された書式に従って入力番号を変換します。

```
WITH nums AS (
```

```
  SELECT 10 n FROM dual union
  SELECT 9.99 n FROM dual union
  SELECT .99 n FROM dual union
  SELECT 1000000 n FROM dual  --one million
)
SELECT n "Input Number N",
       to_char(n),
       to_char(n, '9,999,999.99') "Number with Commas",
       to_char(n, '0,000,000.000') "Zero_padded Number",
       to_char(n, '9.9EEEE') "Scientific Notation",
       to_char(n, '$9,999,990.00') Monetary,
       to_char(n, 'X') "Hexadecimal Value"
FROM nums;
```

次の文は、TO_CHARファンクションで指定された書式に従って入力番号を変換します。

```
WITH nums AS (
  SELECT 10 n FROM dual union
  SELECT 9.99 n FROM dual union
  SELECT .99 n FROM dual union
  SELECT 1000000 n FROM dual  --one million
)
SELECT n "Input Number N",
       to_char(n),
       to_char(n, '9,999,999.99') "Number with Commas",
       to_char(n, '0,000,000.000') "Zero_padded Number",
       to_char(n, '9.9EEEE') "Scientific Notation",
       to_char(n, '$9,999,990.00') Monetary,
       to_char(n, 'XXXXXX') "Hexadecimal Value"
FROM nums;
```

Live SQL:

日付および数値の書式を設定するための TO_CHAR の使用で、Oracle Live SQL に関連する例を参照および実行します

TO_CHAR (datetime)ファンクション: 例

次の文は、empl_tempという名前の表を作成し、従業員の詳細を移入します。

```
CREATE TABLE empl_temp
  (
     employee_id NUMBER(6),
     first_name  VARCHAR2(20),
     last_name   VARCHAR2(25),
     email       VARCHAR2(25),
     hire_date   DATE DEFAULT SYSDATE,
     job_id      VARCHAR2(10),
     clob_column CLOB
  );
INSERT INTO empl_temp
VALUES(111,'John','Doe','example.com','10-JAN-2015','1001','Experienced Employee');
INSERT INTO empl_temp
VALUES(112,'John','Smith','example.com','12-JAN-2015','1002','Junior Employee');
INSERT INTO empl_temp
VALUES(113,'Johnnie','Smith','example.com','12-JAN-2014','1002','Mid-Career
Employee');
INSERT INTO empl_temp
VALUES(115,'Jane','Doe','example.com','15-JAN-2015','1005','Executive Employee');
```

次の文は、短い書式と長い書式を使用して日付を表示します。

```
SELECT hire_date "Default",
       TO_CHAR(hire_date,'DS') "Short",
       TO_CHAR(hire_date,'DL') "Long"FROM empl_temp
WHERE employee_id IN (111, 112, 115);
Default    Short      Long
---------- ---------- -------------------------
10-JAN-15  1/10/2015  Saturday, January 10, 2015
12-JAN-15  1/12/2015  Monday, January 12, 2015
15-JAN-15  1/15/2015  Thursday, January 15, 2015
```

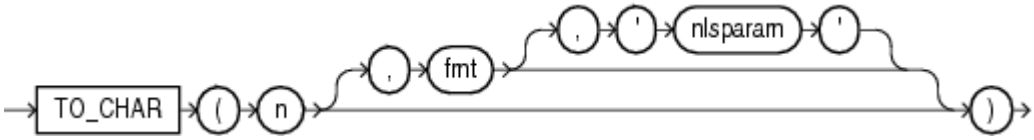
Live SQL:

Oracle Live SQL の TO_CHAR ファンクションの使用で関連する例を参照して実行してください

TO_CHAR (数値)

構文

to_char_number::=



目的

TO_CHAR(数値)は、nを、オプションの数値書式fmtを使用してVARCHAR2データ型の値に変換します。n値には、NUMBER、BINARY_FLOATまたはBINARY_DOUBLE型の値を指定できます。fmtを指定しないと、nの有効桁数を保持するために十分な長さのVARCHAR2値に変換されます。

nが負の場合、書式が適用された後で符号が適用されます。したがって、TO_CHAR(-1, '$9')は、$-1ではなく-$1を戻します。

数値書式の詳細は、「書式モデル」を参照してください。

'nlsparam'引数には、数値書式要素によって戻される次の文字を指定します。

- 小数点文字
- 桁区切り
- 各国通貨記号
- 国際通貨記号

この引数は、次の書式で指定します。

```
'NLS_NUMERIC_CHARACTERS = ''dg''
   NLS_CURRENCY = ''text''
   NLS_ISO_CURRENCY = territory '
```

文字dおよびgは、それぞれ小数点文字および桁区切りを表します。これらは、異なるシングルバイト文字である必要があります。引用符付き文字列の中では、パラメータ値を囲む一重引用符を2つ使用する必要があります。通貨記号には10文字使用できます。

'nlsparam'またはパラメータのいずれか1つを省略すると、このファンクションはセッションのデフォルト・パラメータ値を使用します。

関連項目:

- データ変換のセキュリティ上の考慮事項
- このファンクションの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の文は、暗黙的な変換を使用して、文字列と数値を数値に結合します。

```
SELECT TO_CHAR('01110' + 1) FROM DUAL;
```

```
TO_C
----
1111
```

この例と、「TO_CHAR(文字)」の最初の例を比較してください。

次の例では、出力で通貨記号の左側に空白埋めが行われます。オプションの数値書式fmtでは、Lは各国通貨記号を、MIは後に付くマイナス記号(-)を表します。数値書式要素の完全なリストは、表2-15を参照してください。例は、セッション・パラメータNLS_TERRITORYがAMERICAに設定されたセッションの出力を示します。

```
SELECT TO_CHAR(-10000,'L99G999D99MI') "Amount"
     FROM DUAL;
Amount
--------------
  $10,000.00-
```

次の例では、NLS_CURRENCYには、L数値書式要素について各国通貨記号として使用する文字列を指定します。NLS_NUMERIC_CHARACTERSは、カンマをD数値書式要素の小数点区切りとして使用する文字に指定し、ピリオドをG数値書式要素のグループ区切りとして使用する文字に指定します。これらの文字は、ドイツなどの多くの国で想定されます。

```
SELECT TO_CHAR(-10000,'L99G999D99MI',
   'NLS_NUMERIC_CHARACTERS = '',.''
   NLS_CURRENCY = ''AusDollars'' ') "Amount"
     FROM DUAL;
Amount
-------------------
AusDollars10.000,00-
```

次の例では、NLS_ISO_CURRENCYは、C数値書式要素のPOLAND地域に国際通貨記号を使用するようデータベースに指示します。

```
SELECT TO_CHAR(-10000,'99G999D99C',
   'NLS_NUMERIC_CHARACTERS = '',.''
   NLS_ISO_CURRENCY=POLAND') "Amount"
     FROM DUAL;
Amount
-----------------
    -10.000,00PLN
```

TO_CHAR (number)ファンクション: 例

次の文は、empl_tempという名前の表を作成し、従業員の詳細を移入します。

```
CREATE TABLE empl_temp
  (
     employee_id NUMBER(6),
     first_name  VARCHAR2(20),
     last_name   VARCHAR2(25),
     email       VARCHAR2(25),
     hire_date   DATE DEFAULT SYSDATE,
     job_id      VARCHAR2(10),
     clob_column CLOB
  );
INSERT INTO empl_temp
VALUES(111,'John','Doe','example.com','10-JAN-2015','1001','Experienced Employee');
INSERT INTO empl_temp
VALUES(112,'John','Smith','example.com','12-JAN-2015','1002','Junior Employee');
INSERT INTO empl_temp
VALUES(113,'Johnnie','Smith','example.com','12-JAN-2014','1002','Mid-Career
Employee');
INSERT INTO empl_temp
VALUES(115,'Jane','Doe','example.com','15-JAN-2015','1005','Executive Employee');
```

次の文は、数値データをデータベース文字セットに変換します。

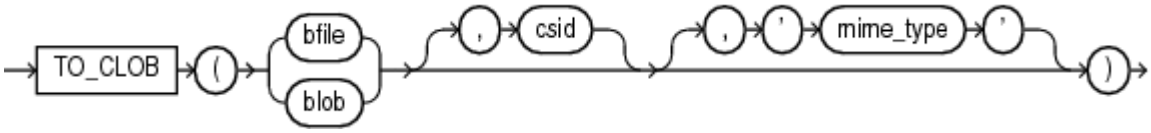
```
SELECT To_char(employee_id) "NUM_TO_CHAR"
FROM   empl_temp
WHERE  employee_id IN ( 111, 112, 113, 115 );
NUM_TO_CHAR
--------------------
111
112
113
115
```

Live SQL:

Oracle Live SQL の TO_CHAR ファンクションの使用で関連する例を参照して実行してください

TO_CLOB (bfile|blob)

構文



目的

TO_CLOB (bfile|blob)は、BFILEデータまたはBLOBデータをデータベース文字セットに変換し、CLOB値としてデータを戻します。

csidには、BFILEまたはBLOBデータの文字セットのIDを指定します。BFILEまたはBLOBデータの文字セットがデータベース文字セットの場合、csidに0の値を指定することも、csid自体を省略することもできます。

mime_typeには、このファンクションで戻されるCLOB値に設定されるMIMEタイプを指定します。mime_typeを省略すると、CLOB値にMIMEタイプが設定されません。

関連項目:

このファンクションの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

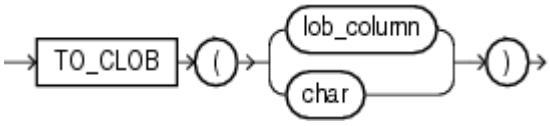
例

次の例では、ID 873の文字セットを使用する、表media_tabのBFILE列の値docuのCLOBを戻します。これにより、結果のCLOBでMIMEタイプがtext/xmlに設定されます。

```
SELECT TO_CLOB(docu, 873, 'text/xml') FROM media_tab;
```

TO_CLOB (文字)

構文



目的

TO_CLOB (文字)は、LOB列またはその他の文字列のNCLOB値をCLOB値に変換します。charは、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBデータ型です。Oracle Databaseは、基底LOBデータを各国語文字セットからデータベース文字セットに変換することによってこのファンクションを実行します。

PL/SQLパッケージ内からは、TO_CLOB (文字)ファンクションを使用してRAW、CHAR、VARCHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBの値をCLOBまたはNCLOBの値に変換できます。

関連項目:

このファンクションの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

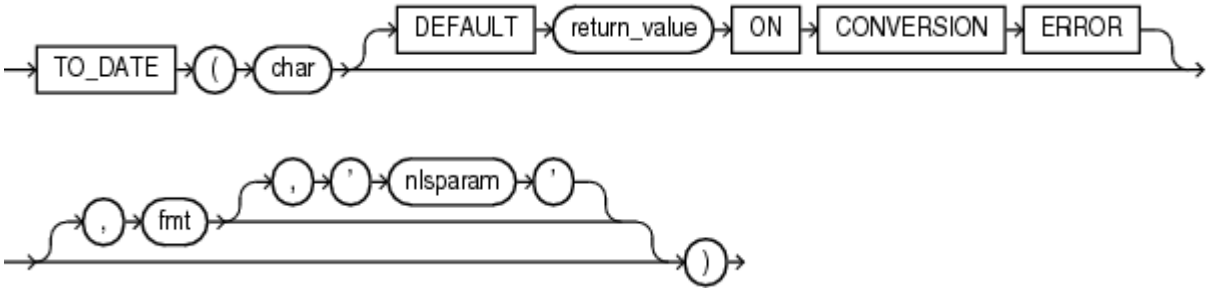
例

次の文は、サンプル表pm.print_mediaのNCLOBデータをCLOBに変換し、CLOB列に挿入して、その行の既存のデータを置換します。

```
UPDATE PRINT_MEDIA
   SET AD_FINALTEXT = TO_CLOB (AD_FLTEXTN);
```

TO_DATE

構文



目的

TO_DATEは、charをDATEデータ型の値に変換します。

charには、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のデータ型の文字列と評価される任意の式を指定できます。

ノート:

このファンクションは、データを別の日時データ型へは変換しません。他の日時変換の詳細は、「TO_TIMESTAMP」、「TO_TIMESTAMP_TZ」、「TO_DSINTERVAL」および「TO_YMINTERVAL」を参照してください。

オプションのDEFAULT return_value ON CONVERSION ERROR句により、charからDATEへの変換中にエラーが発生した場合にこのファンクションで戻される値を指定できます。charの評価中にエラーが発生した場合、この句による影響はありません。return_valueは式またはバインド変数にすることができ、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のデータ型の文字列、あるいはNULLと評価される必要があります。charをDATEに変換する場合と同じ方法を使用して、return_valueがDATEに変換されます。return_valueをDATEに変換できない場合は、エラーが戻されます。

fmtは、charの書式を指定する日時書式モデルです。fmtを指定しない場合、charはデフォルトの日付書式である必要があります。デフォルトの日付フォーマットは、NLS_TERRITORY初期化パラメータによって暗黙的に決まります。NLS_DATE_FORMATパラメータを使用して明示的に設定することもできます。fmtがJ(ユリウス日)の場合、charは整数である必要があります。

注意:

次の項の例に示すとおり、TO_DATE では、常に書式マスク(fmt)を指定することをお薦めします。書式マスクを指定しないと、このファンクションは、char が NLS_TERRITORY または NLS_DATE_FORMAT パラメータで指定されたものと同じ書式を使用している場合にのみ有効になります。さらに、依存性を回避するために明示的な書式マスクが指定されていない場合は、データベース間でファンクションが不安定になることがあります。

'nlsparam'引数には、日付に変換されるテキスト文字列の言語を指定します。この引数は、次の書式で指定します。

```
'NLS_DATE_LANGUAGE = language'
```

引数charにDATE値を持つTO_DATEファンクションは使用しないでください。戻されるDATE値は、fmtまたはデフォルトの日付

書式によって、元のcharとは異なる世紀の値を持つことがあります。

このファンクションは、CLOBデータを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用してCLOBを引数として渡すことはできます。

関連項目:

詳細は、「日時書式モデル」および「データ型の比較規則」を参照してください。

例

次の例では、文字列を日付に変換します。

```
SELECT TO_DATE(
    'January 15, 1989, 11:00 A.M.',
    'Month dd, YYYY, HH:MI A.M.',
     'NLS_DATE_LANGUAGE = American')
     FROM DUAL;
TO_DATE('
---------
15-JAN-89
```

戻された値は、NLS_TERRITORYパラメータが'AMERICA'に設定されている場合は、デフォルトの日付書式を反映します。異なるNLS_TERRITORYの値が設定されている場合は、異なるデフォルトの日付書式となります。

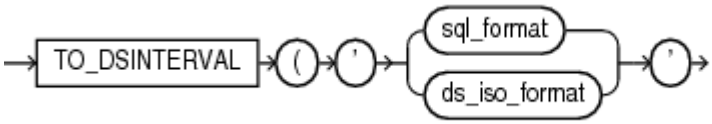
```
ALTER SESSION SET NLS_TERRITORY = 'KOREAN';
SELECT TO_DATE(
    'January 15, 1989, 11:00 A.M.',
    'Month dd, YYYY, HH:MI A.M.',
     'NLS_DATE_LANGUAGE = American')
     FROM DUAL;
TO_DATE(
--------
89/01/15
```

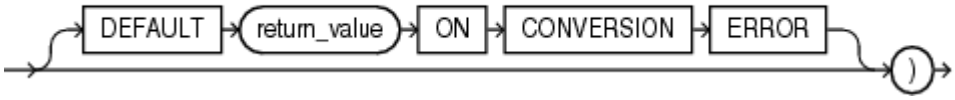
次の例では、月のスペルが誤っていることで、指定した式をDATE値に変換できないため、デフォルト値が戻されます。

```
SELECT TO_DATE('Febuary 15, 2016, 11:00 A.M.'
       DEFAULT 'January 01, 2016 12:00 A.M.' ON CONVERSION ERROR,
       'Month dd, YYYY, HH:MI A.M.') "Value"
  FROM DUAL;
Value
---------
01-JAN-16
```

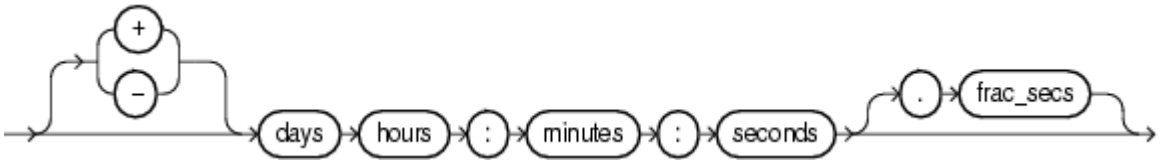
TO_DSINTERVAL

構文

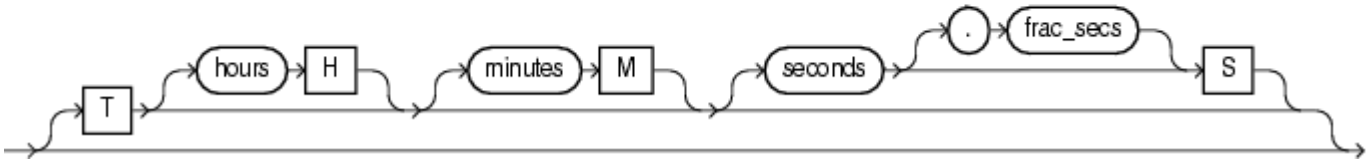




sql_format::=



ds_iso_format::=



ノート:

以前のリリースでは、TO_DSINTERVAL ファンクションはオプションの nlsparam 句を受け入れました。この句は下位互換性のために現在も受け入れられますが、効果はありません。

目的

TO_DSINTERVALは、その引数をINTERVAL DAY TO SECONDデータ型の値に変換します。

引数には、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のデータ型の文字列と評価される任意の式を指定できます。

TO_DSINTERVALは、2つの書式のいずれかの引数を取ります。

- SQL規格(ISO/IEC 9075:2003)に準拠したSQL期間書式(ISO/IEC 9075)
- ISO 8601:2004規格に準拠したISO存続期間書式

SQL書式では、daysは0(ゼロ)から999999999の整数、hoursは0(ゼロ)から23の整数、minutesとsecondsは0(ゼロ)から59の整数になります。frac_secsは秒の小数部であり、.0から.999999999になります。日付と時間は1つ以上の空白で区切ります。これ以外に、書式要素の間に空白を使用できます。

ISO書式では、days、hours、minutesおよびsecondsは、0(ゼロ)から999999999の整数になります。frac_secsは秒の小数部であり、.0から.999999999になります。値には空白を使用できません。Tを指定する場合は、hours、minutesまたはsecondsの値を少なくとも1つ指定する必要があります。

オプションのDEFAULT return_value ON CONVERSION ERROR句により、引数からINTERVAL DAY TO SECOND型への変換中にエラーが発生した場合にこのファンクションで戻される値を指定できます。引数の評価中にエラーが発生した場合、この句による影響はありません。return_valueは式またはバインド変数にすることができ、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のデータ型の文字列と評価される必要があります。これは、SQL書式とISO書式のいずれかにすることができ、ファンクションの引数と同じ書式にする必要はありません。return_valueをINTERVAL DAY TO SECOND型に変換できない場合は、エラーが戻されます。

例

次の例では、SQL書式を使用して、2002年11月1日から100日以上勤務している従業員をhr.employees表から検索します。

```
SELECT employee_id, last_name FROM employees
   WHERE hire_date + TO_DSINTERVAL('100 00:00:00')
   <= DATE '2002-11-01'
   ORDER BY employee_id;
EMPLOYEE_ID LAST_NAME
----------- ---------------
        102 De Haan
        203 Mavris
        204 Baer
        205 Higgins
        206 Giet
```

次の例では、ISO書式を使用して、2009年の始めから100日と5時間後のタイムスタンプを表示します。

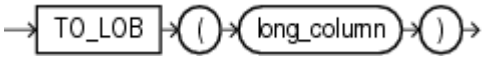
```
SELECT TO_CHAR(TIMESTAMP '2009-01-01 00:00:00' + TO_DSINTERVAL('P100DT05H'),
   'YYYY-MM-DD HH24:MI:SS') "Time Stamp"
     FROM DUAL;
Time Stamp
-------------------
2009-04-11 05:00:00
```

次の例では、指定した式をINTERVAL DAY TO SECOND値に変換できないため、デフォルト値が戻されます。

```
SELECT TO_DSINTERVAL('1o 1:02:10'
       DEFAULT '10 8:00:00' ON CONVERSION ERROR) "Value"
  FROM DUAL;
Value
------------------------------
+000000010 08:00:00.000000000
```

TO_LOB

構文



目的

TO_LOBは、long_column列のLONGまたはLONG RAW値をLOB値に変換します。このファンクションはLONGまたはLONG RAW列に対してのみ、およびINSERT文における副問合せのSELECT構文のリストにおいてのみ適用できます。

このファンクションを使用する前に、LOB列を作成して変換されたLONG値を受け取る必要があります。LONG値を変換する場合は、CLOB列を作成します。LONG RAW値を変換する場合は、BLOB列を作成します。

索引構成表を作成する場合は、CREATE TABLE ... AS SELECT文の副問合せで、TO_LOBファンクションを使用してLONG列をLOB列に変換することはできません。LONG列を含まない索引構成表を作成し、INSERT ...AS SELECT文でTO_LOBファンクションを使用してください。

このファンクションは、PL/SQLパッケージ内で使用できません。かわりに、TO_CLOB (文字)ファンクションまたはTO_BLOB (RAW)ファンクションを使用してください。

関連項目:

- LONG列をLOBに変換する別の方法については、「ALTER TABLE」のmodify_col_properties句を参照してください。
- INSERT文の副問合せについては、「INSERT」を参照してください。
- このファンクションの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、不確定な表old_tableのLONGデータに対するTO_LOBファンクションの使用方法を示します。

```
CREATE TABLE new_table (col1, col2, ... lob_col CLOB);
INSERT INTO new_table (select o.col1, o.col2, ... TO_LOB(o.old_long_col)
   FROM old_table o;
```

TO_MULTI_BYTE

構文

目的

TO_MULTI_BYTEは、シングルバイト文字を、対応するマルチバイト文字に変換してcharを戻します。charのデータ型は、CHAR、VARCHAR2、NCHARまたはNVARCHAR2です。戻り値は、charと同じデータ型です。

char内に同等のマルチバイト文字がないシングルバイト文字は、シングルバイト文字として出力されます。このファンクションは、ご使用のデータベース文字セットに、シングルバイト文字およびマルチバイト文字の両方が含まれている場合にのみ有効です。

このファンクションは、CLOBデータを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用してCLOBを引数として渡すことはできます。

関連項目:

- 詳細は、「データ型の比較規則」を参照してください。
- TO_MULTI_BYTEの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

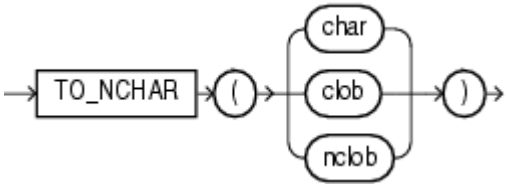
次の例では、シングルバイトAからUTF8のマルチバイトAへの変換を示します。

```
SELECT dump(TO_MULTI_BYTE( 'A')) FROM DUAL;
DUMP(TO_MULTI_BYTE('A'))
------------------------
Typ=1 Len=3: 239,188,161
```

TO_NCHAR (文字)

構文

to_nchar_char::=



目的

TO_NCHAR(文字)は、文字列、CHAR、VARCHAR2、CLOBまたはNCLOB値を各国語文字セットに変換します。戻り値は常にNVARCHAR2です。このファンクションは、各国語文字セットでUSING句を指定したTRANSLATE ... USINGファンクションと同じです。

関連項目:

- 「データ変換」および「TRANSLATE ... USING」を参照してください。
- このファンクションの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

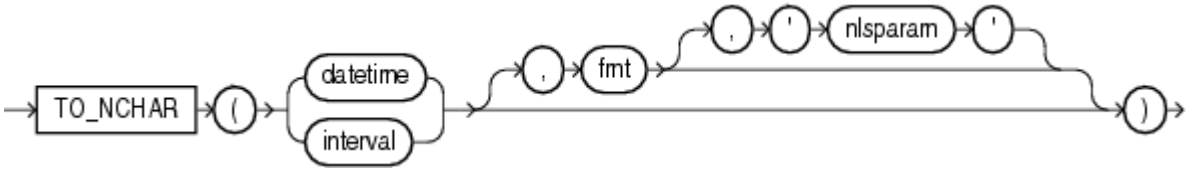
次の例では、oe.customers表のVARCHAR2データを各国語文字セットに変換します。

```
SELECT TO_NCHAR(cust_last_name) FROM customers
   WHERE customer_id=103;
TO_NCHAR(CUST_LAST_NAME)
-------------------------------------------------
Taylor
```

TO_NCHAR (日時)

構文

to_nchar_date::=



目的

TO_NCHAR(日時)は、DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONE、TIMESTAMP WITH LOCAL TIME ZONE、INTERVAL MONTH TO YEARまたはINTERVAL DAY TO SECONDデータ型の日付または期間値をデータベース文字セットから各国語文字セットに変換します。

関連項目:

- データ変換のセキュリティ上の考慮事項
- このファンクションの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

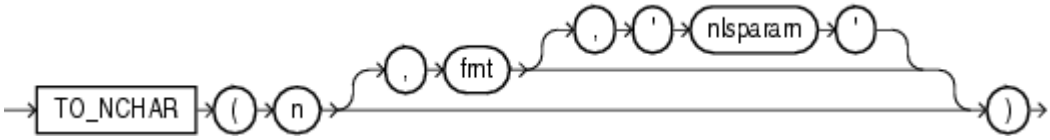
次の例では、状態が9であるすべての注文のorder_dateを各国語文字セットに変換します。

```
SELECT TO_NCHAR(ORDER_DATE) AS order_date
   FROM ORDERS
   WHERE ORDER_STATUS > 9
   ORDER BY order_date;
ORDER_DATE
---------------------------------------------------------------------------
06-DEC-99 02.22.34.225609 PM
13-SEP-99 10.19.00.654279 AM
14-SEP-99 09.53.40.223345 AM
26-JUN-00 10.19.43.190089 PM
27-JUN-00 09.53.32.335522 PM
```

TO_NCHAR (数値)

構文

to_nchar_number::=



目的

TO_NCHAR(数値)は、nを各国語文字セットの文字列に変換します。n値には、NUMBER、BINARY_FLOATまたはBINARY_DOUBLE型の値を指定できます。このファンクションは、引数と同じ型の値を戻します。オプションのfmt、nに対応する'nlsparam'は、DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONE、TIMESTAMP WITH LOCAL TIME ZONE、INTERVAL MONTH TO YEARまたはINTERVAL DAY TO SECONDデータ型です。

関連項目:

- データ変換のセキュリティ上の考慮事項
- このファンクションの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

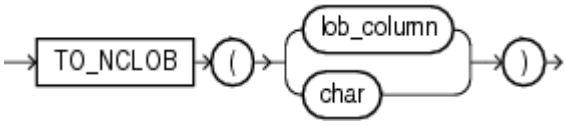
例

次の例では、oe.ordersサンプル表のcustomer_id値を各国語文字セットに変換します。

```
SELECT TO_NCHAR(customer_id) "NCHAR_Customer_ID"  FROM orders
   WHERE order_status > 9
   ORDER BY "NCHAR_Customer_ID";
NCHAR_Customer_ID
----------------------------------------
102
103
148
148
149
```

TO_NCLOB

構文



目的

TO_NCLOBは、LOB列またはその他の文字列のCLOB値をNCLOB値に変換します。charは、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBデータ型です。Oracle Databaseは、charの文字セットをデータベース文字セットから各国語文字セットに変換することによって、このファンクションを実行します。

関連項目:

このファンクションの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

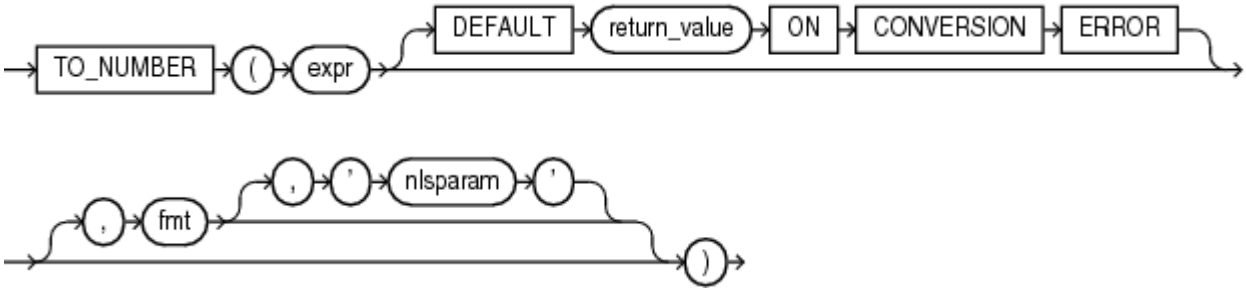
例

次の例では、TO_NCLOBファンクションでデータを変換後、pm.print_media表のNCLOB列に文字データを挿入します。

```
INSERT INTO print_media (product_id, ad_id, ad_fltextn)
   VALUES (3502, 31001,
      TO_NCLOB('Placeholder for new product description'));
```

TO_NUMBER

構文



目的

TO_NUMBERは、exprを、NUMBERデータ型の値に変換します。

exprは、CHAR、VARCHAR2、NCHARまたはNVARCHAR2の型の文字列、NUMBER、BINARY_FLOATまたはBINARY_DOUBLEの型の数値、あるいはNULLと評価される任意の式にすることができます。exprがNUMBERの場合、このファンクションはexprを戻します。exprがNULLと評価されると、関数はNULLを返します。それ以外の場合は、exprがNUMBER値に変換されます。

- CHAR、VARCHAR2、NCHARまたはNVARCHAR2データ型のexprを指定した場合、オプションの書式モデルfmtを指定できます。
- BINARY_FLOATまたはBINARY_DOUBLEのデータ型のexprを指定した場合、書式モデルを指定できません。これは、その内部表現によってのみ浮動小数点数を解釈できるためです。

数値書式の詳細は、「書式モデル」を参照してください。

このファンクションの'nlsparam'引数は、数値変換のTO_CHARファンクションの場合と同じ用途に使用されます。詳細は、「TO_CHAR(数値)」を参照してください。

このファンクションは、CLOBデータを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用してCLOBを引数として渡すことはできます。

関連項目:

詳細は、「データ型の比較規則」を参照してください。

例

次の例では、文字列データを数値に変換します。

```
UPDATE employees SET salary = salary +
   TO_NUMBER('100.00', '9G999D99')
   WHERE last_name = 'Perkins';

SELECT TO_NUMBER('-AusDollars100','L9G999D99',
   ' NLS_NUMERIC_CHARACTERS = '',.''
     NLS_CURRENCY            = ''AusDollars''
   ') "Amount"
     FROM DUAL;
    Amount
----------
      -100
```

次の例では、指定した式をNUMBER値に変換できないため、デフォルト値の0が戻されます。

```
SELECT TO_NUMBER('2,00' DEFAULT 0 ON CONVERSION ERROR) "Value"
  FROM DUAL;
   Value
--------
       0
```

TO_SINGLE_BYTE

構文

目的

TO_SINGLE_BYTEは、マルチバイト文字を、対応するシングルバイト文字に変換してcharを戻します。charのデータ型は、CHAR、VARCHAR2、NCHARまたはNVARCHAR2です。戻り値は、charと同じデータ型です。

char内に同等のシングルバイト文字がないマルチバイト文字は、マルチバイト文字として出力されます。このファンクションは、ご使用のデータベース文字セットに、シングルバイト文字およびマルチバイト文字の両方が含まれている場合にのみ有効です。

このファンクションは、CLOBデータを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用してCLOBを引数として渡すことはできます。

関連項目:

- 詳細は、「データ型の比較規則」を参照してください。
- TO_SINGLE_BYTEの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

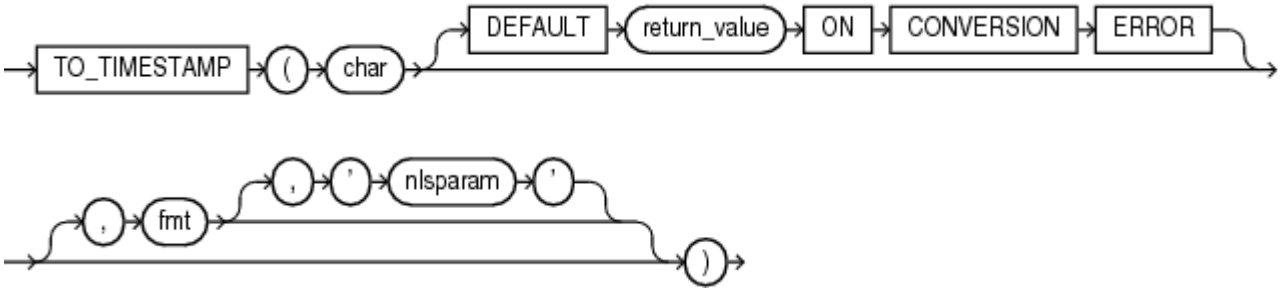
例

次の例では、UTF8のマルチバイトAからシングルバイトのASCIIのAへの変換を示します。

```
SELECT TO_SINGLE_BYTE( CHR(15711393)) FROM DUAL;
T
-
A
```

TO_TIMESTAMP

構文



目的

TO_TIMESTAMPは、charをTIMESTAMPデータ型の値に変換します。

charには、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のデータ型の文字列と評価される任意の式を指定できます。

オプションのDEFAULT return_value ON CONVERSION ERROR句により、charからTIMESTAMPへの変換中にエラーが発生した場合にこのファンクションで戻される値を指定できます。charの評価中にエラーが発生した場合、この句による影響はありません。return_valueは式またはバインド変数にすることができ、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のデータ型の文字列、あるいはNULLと評価される必要があります。charをTIMESTAMPに変換する場合と同じ方法を使用して、return_valueがTIMESTAMPに変換されます。return_valueをTIMESTAMPに変換できない場合は、エラーが戻されます。

オプションのfmtは、charの書式を指定します。fmtを指定しない場合、charはデフォルト書式のTIMESTAMPデータ型である必要があります。このデータ型は、NLS_TIMESTAMP_FORMAT初期化パラメータによって決まります。オプションの'nlsparam'引数は、日付変換のTO_CHARファンクションの場合と同じ用途で使用されます。

このファンクションは、CLOBデータを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用してCLOBを引数として渡すことはできます。

関連項目:

詳細は、「データ型の比較規則」を参照してください。

例

次の例では、文字列をタイムスタンプに変換します。文字列がデフォルトのTIMESTAMP書式ではないため、書式マスクを指定する必要があります。

```
SELECT TO_TIMESTAMP ('10-Sep-02 14:10:10.123000', 'DD-Mon-RR HH24:MI:SS.FF')
   FROM DUAL;
TO_TIMESTAMP('10-SEP-0214:10:10.123000','DD-MON-RRHH24:MI:SS.FF')
---------------------------------------------------------------------------
10-SEP-02 02.10.10.123000000 PM
```

次の例では、無効な月を指定していることで、指定した式をTIMESTAMP値に変換できないため、デフォルト値のNULLが戻されます。

```
SELECT TO_TIMESTAMP ('10-Sept-02 14:10:10.123000'
       DEFAULT NULL ON CONVERSION ERROR,
       'DD-Mon-RR HH24:MI:SS.FF',
```

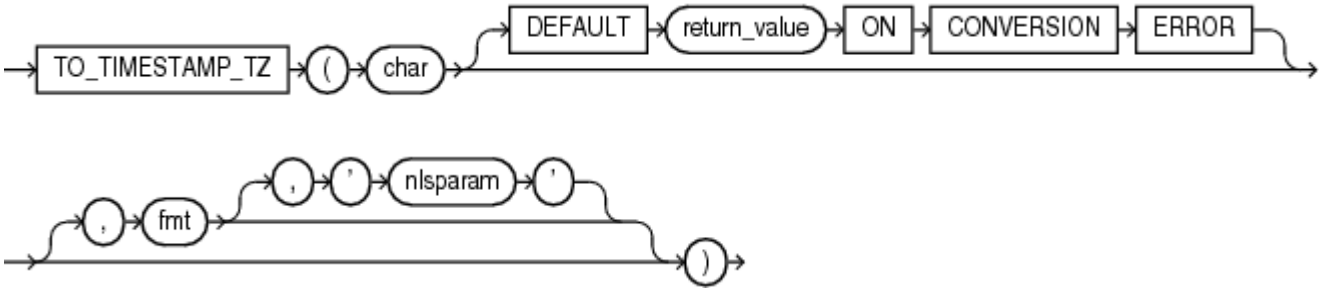
```
      'NLS_DATE_LANGUAGE = American') "Value"
FROM DUAL;
```

関連項目:

デフォルトのTIMESTAMP書式の詳細は、NLS_TIMESTAMP_FORMAT初期化パラメータを参照してください。書式マスクの指定の詳細は、「日時書式モデル」を参照してください

TO_TIMESTAMP_TZ

構文



目的

TO_TIMESTAMP_TZは、charをTIMESTAMP WITH TIME ZONEデータ型の値に変換します。

charには、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のデータ型の文字列と評価される任意の式を指定できます。

ノート:

このファンクションは、文字列を TIMESTAMP WITH LOCAL TIME ZONE に変換しません。変換するには、「CAST」で示すように、CAST ファンクションを使用してください。

オプションのDEFAULT return_value ON CONVERSION ERROR句により、charからTIMESTAMP WITH TIME ZONEへの変換中にエラーが発生した場合にこのファンクションで戻される値を指定できます。charの評価中にエラーが発生した場合、この句による影響はありません。return_valueは式またはバインド変数にすることができ、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のデータ型の文字列、あるいはNULLと評価される必要があります。charをTIMESTAMP WITH TIME ZONEに変換する場合と同じ方法を使用して、return_valueがTIMESTAMP WITH TIME ZONEに変換されます。return_valueをTIMESTAMP WITH TIME ZONEに変換できない場合は、エラーが戻されます。

オプションのfmtは、charの書式を指定します。fmtを指定しない場合、charはデフォルト書式のTIMESTAMP WITH TIME ZONEデータ型である必要があります。オプションの'nlsparam'は、日付変換のTO_CHARファンクションの場合と同じ用途で使用されます。

例

次の例では、文字列をTIMESTAMP WITH TIME ZONE値に変換します。

```
SELECT TO_TIMESTAMP_TZ('1999-12-01 11:00:00 -8:00',
   'YYYY-MM-DD HH:MI:SS TZH:TZM') FROM DUAL;
TO_TIMESTAMP_TZ('1999-12-0111:00:00-08:00','YYYY-MM-DDHH:MI:SSTZH:TZM')
---------------------------------------------------------------------
01-DEC-99 11.00.00.000000000 AM -08:00
```

次の例では、サンプル表oe.order_itemsおよびoe.ordersを使用して、UNION演算のNULL列をTIMESTAMP WITH LOCAL TIME ZONEとしてキャストします。

```
SELECT order_id, line_item_id,
   CAST(NULL AS TIMESTAMP WITH LOCAL TIME ZONE) order_date
   FROM order_items
UNION
SELECT order_id, to_number(null), order_date
   FROM orders;
```

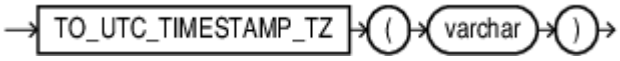
```
  ORDER_ID LINE_ITEM_ID ORDER_DATE
---------- ------------ ---------------------------------
      2354            1
      2354            2
      2354            3
      2354            4
      2354            5
      2354            6
      2354            7
      2354            8
      2354            9
      2354           10
      2354           11
      2354           12
      2354           13
      2354              14-JUL-00 05.18.23.234567 PM
      2355            1
      2355            2
. . .
```

次の例では、無効な月を指定していることで、指定した式をTIMESTAMP WITH TIME ZONE値に変換できないため、デフォルト値のNULLが戻されます。

```
SELECT TO_TIMESTAMP_TZ('1999-13-01 11:00:00 -8:00'
       DEFAULT NULL ON CONVERSION ERROR,
       'YYYY-MM-DD HH:MI:SS TZH:TZM') "Value"
  FROM DUAL;
```

TO_UTC_TIMESTAMP_TZ

構文



目的

SQLファンクションTO_UTC_TIMESTAMP_TZは、varchar入力としてISO 8601の日付フォーマット文字列を受け取り、SQLデータ型TIMESTAMP WITH TIMEZONEのインスタンスを戻します。入力はUTC時間(協定世界時、以前はグリニッジ標準時)に正規化されます。SQLファンクションTO_TIMESTAMP_TZとは異なり、新しいファンクションでは入力文字列がISO 8601日付形式を使用して、タイムゾーンをUTC 0にデフォルト設定すると想定しています。

このファンクションの一般的な使用方法として、SQLファンクションSYS_EXTRACT_UTCにその出力を提供し、UTC時間を取得してこれをSQLバインド変数としてSQL/JSON条件JSON_EXISTSに渡し、タイムスタンプ範囲の比較を実行します。

これは、日付と時刻に許可されている構文です。

- 日付(のみ):YYYY-MM-DD
- 日時: YYYY-MM-DDThh:mm:ss[.s[s[s[s[s]]]]][Z|(+|-)hh:mm]

説明:

- YYYYは年を4桁の10進数で指定します。
- MMは月を2桁の10進数の00から12で指定します。
- DDは日を2桁の10進数の00から31で指定します。
- hhは時を2桁の10進数の00から23で指定します。
- mmは分を2桁の10進数の00から59で指定します。
- ss[.s[s[s[s]]]]は秒を2桁の10進数の00から59で、必要に応じて小数点と1桁から6桁の10進数(秒の小数部を表す)を続けて指定します。
- ZはUTC時間(タイムゾーン0)を指定します。(+00:00で指定することもできますが、-00:00で指定することはできません。)
- (+|-)hh:mmはタイムゾーンをUTCとの差として指定します。(+または-のいずれか1つを指定する必要があります。)

時間値の場合、タイムゾーン部分はオプションです。これが指定されていない場合、UTC時間が仮定されます。

他のISO 8601日時構文はサポートされていません。具体的には、次のとおりです。

- ハイフンで始まるマイナスの日付(BCE 1年より前の日付) (たとえば-2018-10-26T21:32:52)はサポートされていません。
- ハイフンおよびコロン区切文字を入力する必要があります。いわゆる「基本」形式のYYYYMMDDThhmmssはサポートされていません。
- 序数日(年+通日、暦週+日数)はサポートされていません。
- 4桁より大きい年の使用はサポートされていません。

サポートされている日付と時刻は次のとおりです。

- 2018–10–26T21:32:52
- 2018-10-26T21:32:52+02:00
- 2018-10-26T19:32:52Z
- 2018-10-26T19:32:52+00:00
- 2018-10-26T21:32:52.12679

サポートされていない日付と時刻は次のとおりです。

- 2018-10-26T21:32 (時刻が指定されている場合は、そのすべての部分が存在する必要があります)
- 2018-10-26T25:32:52+02:00 (時の部分が25で、範囲外です)
- 18-10-26T21:32 (年が完全に指定されていません)

例

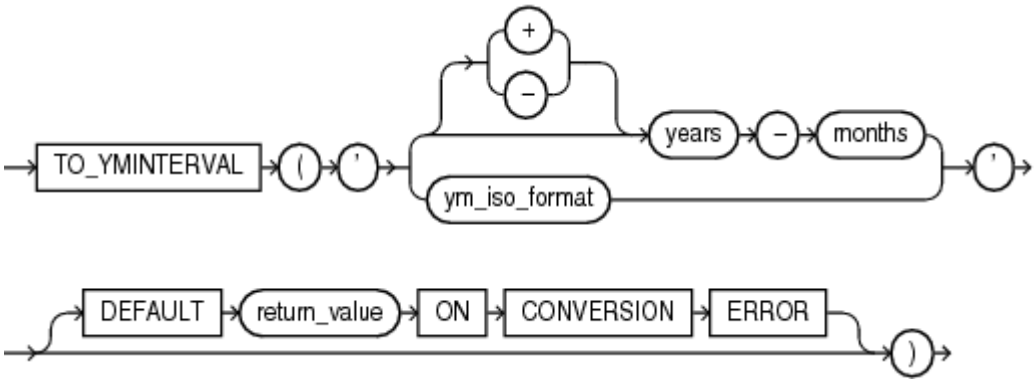
```
SELECT TO_UTC_TIMESTAMP_TZ('1998-01-01') FROM DUAL;
TO_UTC_TIMESTAMP_TZ('1998-01-01')
---------------------------------------------------------------------------
01-JAN-98 12.00.00.000000000 AM +00:00

SELECT TO_UTC_TIMESTAMP_TZ('2000-01-02T12:34:56.789') FROM DUAL;
TO_UTC_TIMESTAMP_TZ('2000-01-02T12:34:56.789')
---------------------------------------------------------------------------
02-JAN-00 12.34.56.789000000 PM +00:00

SELECT TO_UTC_TIMESTAMP_TZ('2016-05-05T00:00:00.000Z') FROM DUAL;
TO_UTC_TIMESTAMP_TZ('2016-05-05T00:00:00.000Z')
---------------------------------------------------------------------------
05-MAY-16 12.00.00.000000000 AM +00:00

SELECT TO_UTC_TIMESTAMP_TZ('2016-05-05T02:04:35.4678Z') FROM DUAL;
TO_UTC_TIMESTAMP_TZ('2016-05-05T02:04:35.4678Z')
---------------------------------------------------------------------------
05-MAY-16 02.04.35.467800000 AM +00:00
```

関連項目:

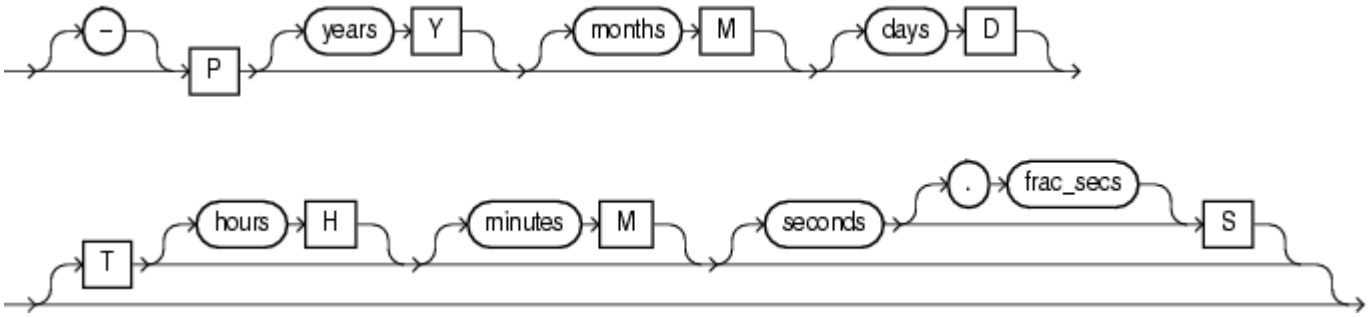
- ISO 8601標準
- WikipediaのISO 8601

TO_YMINTERVAL

構文



ym_iso_format::=



目的

TO_YMINTERVALは、その引数をINTERVAL MONTH TO YEARデータ型の値に変換します。

引数には、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のデータ型の文字列と評価される任意の式を指定できます。

TO_YMINTERVALは、2つの書式のいずれかの引数を取ります。

- SQL規格(ISO/IEC 9075:2003)に準拠したSQL期間書式(ISO/IEC 9075)
- ISO 8601:2004規格に準拠したISO存続期間書式

SQL書式では、yearsは0(ゼロ)から999999999の整数、monthsは0(ゼロ)から11の整数になります。これ以外に、書式要素の間に空白を使用できます。

ISO書式では、yearsとmonthsは、0(ゼロ)から999999999の整数になります。days、hours、minutes、secondsおよびfrac_secsは、負でない整数になります。負の整数を指定した場合は、無視されます。値には空白を使用できません。Tを指定する場合は、hours、minutesまたはsecondsの値を少なくとも1つ指定する必要があります。

オプションのDEFAULT return_value ON CONVERSION ERROR句により、引数からINTERVAL MONTH TO YEAR型への変換中にエラーが発生した場合にこのファンクションで戻される値を指定できます。引数の評価中にエラーが発生した場合、この句による影響はありません。return_valueは式またはバインド変数にすることができ、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のデータ型の文字列と評価される必要があります。これは、SQL書式とISO書式のいずれかにすることができ、ファンクションの引数と同じ書式にする必要はありません。return_valueをINTERVAL MONTH TO YEAR型に変換できない場合は、エラーが戻されます。

例

次の例では、サンプル表hr.employeesの各従業員の雇用された後から1年と2か月後の日付を計算します。

```
SELECT hire_date, hire_date + TO_YMINTERVAL('01-02') "14 months"
   FROM employees;
HIRE_DATE 14 months
--------- ---------
17-JUN-03 17-AUG-04
21-SEP-05 21-NOV-06
13-JAN-01 13-MAR-02
20-MAY-08 20-JUL-09
21-MAY-07 21-JUL-08
. . .
```

次の例では、ISO書式を使用して同じ計算を行います。

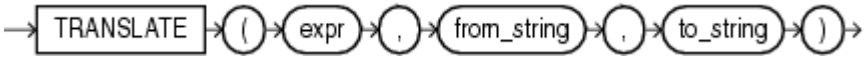
```
SELECT hire_date, hire_date + TO_YMINTERVAL('P1Y2M') FROM employees;
```

次の例では、指定した式をINTERVAL MONTH TO YEAR値に変換できないため、デフォルト値が戻されます。

```
SELECT TO_YMINTERVAL('1x-02'
       DEFAULT '00-00' ON CONVERSION ERROR) "Value"
  FROM DUAL;
Value
-------------
+000000000-00
```

TRANSLATE

構文



目的

TRANSLATEは、from_string内のすべての文字をto_string内の対応する文字に置換してexprを戻します。from_string内に存在しないexpr内の文字は置換されません。引数from_stringには、to_stringより多い文字を指定できます。この場合、from_stringの終わりにある余分な文字には、to_string内に対応する文字がありません。これらの余分な文字がexpr内にある場合、それらの文字は戻り値から削除されます。

from_string内に1つの文字が複数回現れた場合は、最初の出現に対応するto_stringマッピングが使用されます。

戻り値からfrom_string内のすべての文字を削除するために、to_stringに空の文字列を使用することはできません。Oracle Databaseは空の文字列をNULLと解析するため、このファンクションにNULLの引数がある場合、NULLが戻されます。from_string内の文字をすべて削除するには、from_stringの先頭に別の文字を結合し、この文字をto_stringとして指定します。たとえば、TRANSLATE(expr, 'x0123456789', 'x')では、すべての数字がexprから削除されます。

TRANSLATEは、REPLACEファンクションに関連する機能を提供します。REPLACEファンクションでは、単一の文字列から別の単一の文字列への置換、および文字列の削除を実行できます。TRANSLATEでは、1回の操作で複数の単一文字を1対1で置き換えることができます。

このファンクションは、CLOBデータを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用してCLOBを引数として渡すことはできます。

関連項目:

- 詳細は、「データ型の比較規則」および「REPLACE」を参照してください。
- exprからの文字とfrom_stringからの文字を比較するためにTRANSLATEで使用される照合を定義する照合決定ルール、およびTRANSLATEの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください

例

次の文は、本のタイトルを、ファイル名などに使用するために文字列に変換します。from_stringには、空白、アスタリスク、スラッシュおよびアポストロフィの4つの文字(およびエスケープ文字のアポストロフィ)が含まれています。to_stringには、3つのアンダースコアのみが含まれています。from_stringの4つ目の文字に対応する置換文字がないため、戻り値ではアポストロフィが削除されています。

```
SELECT TRANSLATE('SQL*Plus User''s Guide', ' */''', '___') FROM DUAL;
TRANSLATE('SQL*PLUSU
--------------------
SQL_Plus_Users_Guide
```

TRANSLATE ... USING

構文

目的

TRANSLATE ... USINGは、charをデータベース文字セットと各国語文字セット間の変換に指定された文字セットに変換します。

> ノート:
>
> TRANSLATE ... USING は、主に ANSI との互換性のためにサポートされているファンクションです。TO_CHAR および TO_NCHAR ファンクションを使用してデータをデータベース文字セットまたは各国語文字セットに適切に変換することをお薦めします。TO_CHAR および TO_NCHAR は、文字データのみを受け入れる TRANSLATE ... USING に比べ、より多くのデータ型を引数として取ることができます。

引数charは変換する式です。

- USING CHAR_CS引数を指定すると、charがデータベース文字セットに変換されます。出力データ型はVARCHAR2です。
- USING NCHAR_CS引数を指定すると、charが各国語文字セットに変換されます。出力データ型はNVARCHAR2です。

このファンクションは、OracleのCONVERTファンクションに似ていますが、入力または出力のデータ型にNCHARまたはNVARCHAR2を使用する場合は、CONVERTではなくこのファンクションを使用する必要があります。入力にUCS2コードポイントまたはバックスラッシュ(¥)が含まれる場合は、UNISTRファンクションを使用してください。

関連項目:

- 「CONVERT」および「UNISTR」
- TRANSLATE ... USINGの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

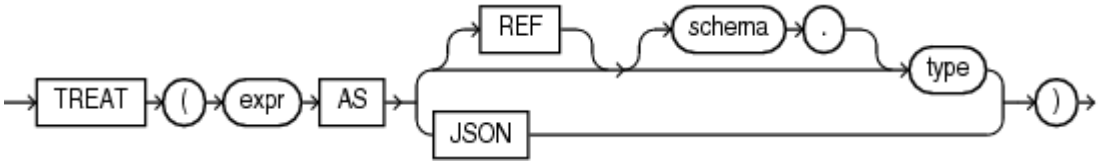
次の文は、サンプル表oe.product_descriptionsのデータを使用して、TRANSLATE ... USINGファンクションの使用方法を示しています。

```
CREATE TABLE translate_tab (char_col  VARCHAR2(100),
                            nchar_col NVARCHAR2(50));
INSERT INTO translate_tab
   SELECT NULL, translated_name
      FROM product_descriptions
      WHERE product_id = 3501;
SELECT * FROM translate_tab;
CHAR_COL              NCHAR_COL
```

```
-------------------- --------------------------------------------------
. . .
                     C pre SPNIX4.0 - Sys
                     C pro SPNIX4.0 - Sys
                     C til SPNIX4.0 - Sys
                     C voor SPNIX4.0 - Sys
. . .
UPDATE translate_tab
   SET char_col = TRANSLATE (nchar_col USING CHAR_CS);
SELECT * FROM translate_tab;
CHAR_COL                  NCHAR_COL
------------------------- -------------------------
. . .
C per a SPNIX4.0 - Sys    C per a SPNIX4.0 - Sys
C pro SPNIX4.0 - Sys      C pro SPNIX4.0 - Sys
C for SPNIX4.0 - Sys      C for SPNIX4.0 - Sys
C til SPNIX4.0 - Sys      C til SPNIX4.0 - Sys
. . .
```

TREAT

構文



目的

TREATファンクションを使用して、式の宣言された型を変更できます。

式がJSONデータを返すようにする場合は、キーワードAS JSONを使用します。これは、テキストを強制的にJSONデータとして解釈させる場合に便利です。たとえば、{}のVARCHAR2値を、文字列ではなく空のJSONオブジェクトとして解釈させる場合に使用できます。

このファンクションを使用する場合は、typeに対するEXECUTEオブジェクト権限が必要です。

- expr AS JSONのexprは、JSONを含むSQLデータ型(CLOBなど)です。
- expr AS typeのexprおよびtypeは、ユーザー定義のオブジェクト型である必要があります(トップレベルのコレクションを除く)。
- typeは、スーパータイプまたはexprの宣言型のサブタイプである必要があります。exprの最も指定される型がtype(またはtypeのサブタイプ)である場合、TREATはexprを戻します。exprの最も指定される型がtype(またはtypeのサブタイプ)ではない場合、TREATはNULLを戻します。
- exprの宣言型がREF型の場合にのみ、REFを指定できます。
- exprの宣言型がexprのソース型へのREFである場合、typeはサブタイプまたはexprのソース型のスーパータイプである必要があります。DEREF(expr)の最も指定される型がtype(またはtypeのサブタイプ)である場合、TREATはexprを戻します。DEREF(expr)の最も指定される型がtype(またはtypeのサブタイプ)ではない場合、TREATはNULLを戻します。

関連項目:

詳細は、「データ型の比較規則」を参照してください。

例

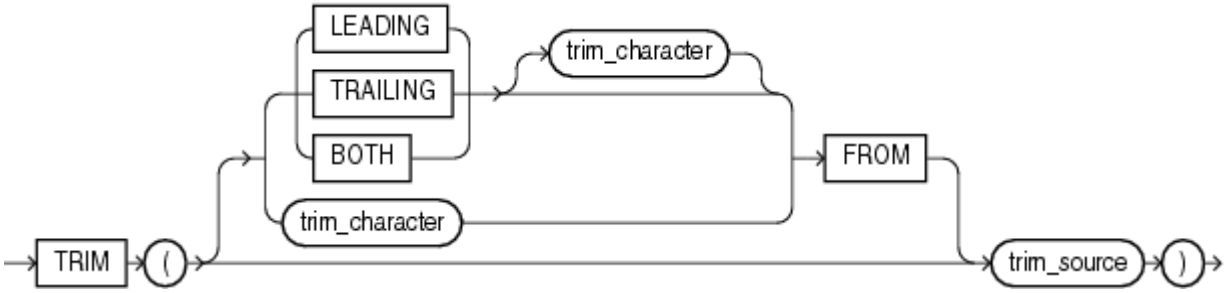
次の例では、「置換可能な表および列のサンプル」で作成された表oe.personsを使用します。次の例では、persons表のすべての人物の給与属性を検索します。従業員ではない人物のインスタンスの値はNULLです。

```
SELECT name, TREAT(VALUE(p) AS employee_t).salary salary
   FROM persons p;
NAME                           SALARY
------------------------- ----------
Bob
Joe                            100000
Tim                              1000
```

TREATファンクションを使用すると、置換可能列のサブタイプ属性に索引を作成できます。例は、「置換可能な列の索引の作成: 例」を参照してください。

TRIM

構文



目的

TRIMによって、文字列の先行または後続文字(またはその両方)を切り捨てることができます。trim_characterまたはtrim_sourceが文字リテラルの場合、一重引用符で囲む必要があります。

- LEADINGを指定すると、Oracle Databaseはtrim_characterと等しい先行文字を削除します。
- TRAILINGを指定すると、Oracleはtrim_characterと等しい後続文字を削除します。
- BOTHを指定するか、またはいずれも指定しない場合、Oracleはtrim_characterと等しい先行および後続文字を削除します。
- trim_characterを指定しないと、デフォルト値は空白になります。
- trim_sourceのみを指定すると、Oracleは先行および後続空白を削除します。
- このファンクションは、値をVARCHAR2データ型で戻します。値の最大長は、trim_sourceの長さです。
- trim_sourceまたはtrim_characterのいずれかがNULLの場合、TRIMはNULLを戻します。

trim_characterとtrim_sourceは両方とも、VARCHAR2またはVARCHAR2に暗黙的に変換できる任意のデータ型です。戻される文字列は、trim_sourceがCHARまたはVARCHAR2(NCHARまたはNVARCHAR2)データ型の場合はVARCHAR2(NVARCHAR2)データ型、trim_sourceがCLOBデータ型の場合はCLOBになります。trim_sourceと同じ文字セットの文字列が戻されます。

関連項目:

trim_characterからの文字とtrim_sourceからの文字を比較するためにTRIMで使用される照合を定義する照合決定ルール、およびこのファンクションの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、hrスキーマの従業員の雇用日から先行0(ゼロ)を切り捨てます。

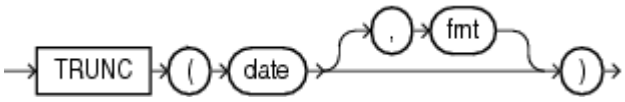
```
SELECT employee_id,
      TO_CHAR(TRIM(LEADING 0 FROM hire_date))
      FROM employees
      WHERE department_id = 60
      ORDER BY employee_id;
EMPLOYEE_ID TO_CHAR(T
----------- ---------
        103 20-MAY-08
```

```
104 21-MAY-07
105 25-JUN-05
106 5-FEB-06
107 7-FEB-07
```

TRUNC (日付)

構文

trunc_date::=



目的

TRUNC(日付)ファンクションは、時刻部分を書式モデルfmtで指定された単位まで切り捨てたdateを戻します。このファンクションは、NLS_CALENDARセッション・パラメータの影響を受けません。このファンクションはグレゴリオ暦の規則に従って動作します。戻される値は、dateに異なる日時データ型を指定した場合でも、常にDATEデータ型です。fmtを省略すると、デフォルトの書式モデル'DD'が使用され、午前0時(真夜中)を基準に切り捨てられたdateの値が戻されます。fmtで使用できる書式モデルは、「ROUNDおよびTRUNC日付ファンクション」を参照してください。

例

次の例では、日付を切り捨てます。

```
SELECT TRUNC(TO_DATE('27-OCT-92','DD-MON-YY'), 'YEAR')
  "New Year" FROM DUAL;

New Year
---------
01-JAN-92
```

TRUNCを使用した日付の書式設定: 例

次の例では、TRUNCファンクションは、書式モデルの指定に従って、日付の時刻部分で入力日付を返します。

```
WITH dates AS (
  SELECT date'2015-01-01' d FROM dual union
  SELECT date'2015-01-10' d FROM dual union
  SELECT date'2015-02-01' d FROM dual union
  SELECT timestamp'2015-03-03 23:45:00' d FROM dual union
  SELECT timestamp'2015-04-11 12:34:56' d FROM dual
)
SELECT d "Original Date",
       trunc(d) "Nearest Day, Time Removed",
       trunc(d, 'ww') "Nearest Week",
       trunc(d, 'iw') "Start of Week",
       trunc(d, 'mm') "Start of Month",
       trunc(d, 'year') "Start of Year"
FROM dates;
```

次の例では、入力日付の値が切り捨てられ、切り捨てられた日付値の分の部分を、TO_CHARファンクションを使用して取得します。

```
WITH dates AS (
  SELECT date'2015-01-01' d FROM dual union
  SELECT date'2015-01-10' d FROM dual union
  SELECT date'2015-02-01' d FROM dual union
  SELECT timestamp'2015-03-03 23:45:00' d FROM dual union
  SELECT timestamp'2015-04-11 12:34:56' d FROM dual
)
SELECT d "Original Date",
       trunc(d) "Date with Time Removed",
```

```
       to_char(trunc(d, 'mi'), 'dd-mon-yyyy hh24:mi') "Nearest Minute",
       trunc(d, 'iw') "Start of Week",
       trunc(d, 'mm') "Start of Month",
       trunc(d, 'year') "Start of Year"
FROM dates;
```

次の文は、現行のセッションの日付書式を変更します。

```
ALTER SESSION SET nls_date_format = 'dd-mon-yyyy hh24:mi';
```

次の例では、新しい日付書式でデータが表示されます。

```
WITH dates AS (
  SELECT date'2015-01-01' d FROM dual union
  SELECT date'2015-01-10' d FROM dual union
  SELECT date'2015-02-01' d FROM dual union
  SELECT timestamp'2015-03-03 23:44:32' d FROM dual union
  SELECT timestamp'2015-04-11 12:34:56' d FROM dual
)
SELECT d "Original Date",
       trunc(d) "Date, time removed",
       to_char(trunc(d, 'mi'), 'dd-mon-yyyy hh24:mi') "Nearest Minute",
       trunc(d, 'iw') "Start of Week",
       trunc(d, 'mm') "Start of Month",
       trunc(d, 'year') "Start of Year"
FROM dates;
```

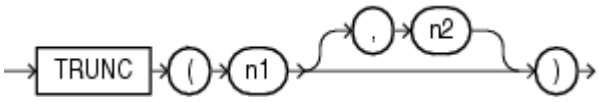
Live SQL:

TRUNC を使用した日付の書式設定で、Oracle Live SQL の関連例を表示し、実行します。

TRUNC (数値)

構文

trunc_number::=



目的

TRUNC(数値)ファンクションは、n1を小数第n2位までに切り捨てた値を戻します。n2を指定しない場合、n1の小数点以下を切り捨てます。n2が負の場合は、小数点の左n2桁を切り捨てて、0(ゼロ)にします。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。n2を指定しない場合、このファンクションは、引数の数値データ型と同じデータ型を戻します。n2を指定すると、このファンクションはNUMBERを戻します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

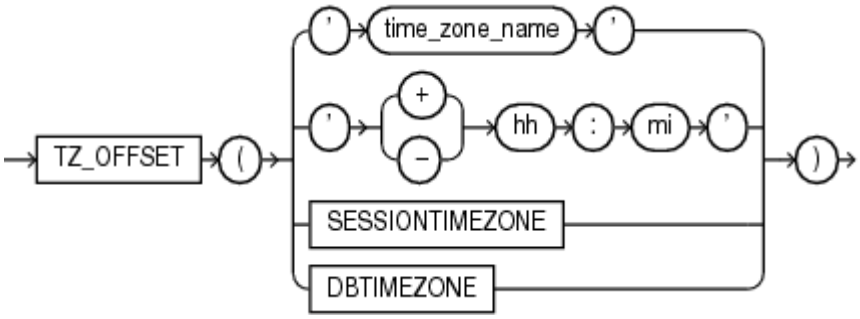
例

次の例では、数値を切り捨てます。

```
SELECT TRUNC(15.79,1) "Truncate" FROM DUAL;
  Truncate
----------
      15.7
SELECT TRUNC(15.79,-1) "Truncate" FROM DUAL;
  Truncate
----------
        10
```

TZ_OFFSET

構文



目的

TZ_OFFSETは、文が実行された日付に基づいた引数に対応するタイムゾーン・オフセットを戻します。有効なタイムゾーン地域名、UTCからのタイムゾーン・オフセット(それ自体を戻します)、またはキーワードSESSIONTIMEZONEまたはDBTIMEZONEを入力できます。time_zone_nameの有効な値を表示するには、V$TIMEZONE_NAMES動的パフォーマンス・ビューのTZNAME列を問い合せます。

ノート:

夏時間機能には、タイムゾーン地域名が必要です。この名前は、大小 2 つのタイムゾーン・ファイルに格納されます。これらのファイルのうち、使用する環境および使用する Oracle Database のリリースに応じて、いずれか一方がデフォルトのファイルになります。タイムゾーン・ファイルおよびタイムゾーン名の詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

関連項目:

- TZ_OFFSETの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、UTCからUS/東部タイムゾーンのタイムゾーン・オフセットを戻します。

```
SELECT TZ_OFFSET('US/Eastern') FROM DUAL;
TZ_OFFS
-------
-04:00
```

UID

構文

目的

UIDは、セッション・ユーザー(ログインしているユーザー)を一意に識別する整数を戻します。

関連項目:

セッション・ユーザーをOracle Databaseが判断する方法を学習するには、「USER」を参照してください。

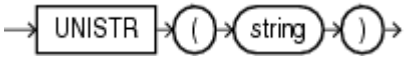
例

次の例では、セッション・ユーザーのUIDが返されます。

```
SELECT UID FROM DUAL;
```

UNISTR

構文



目的

UNISTRは、引数としてテキスト・リテラルまたは文字データに変換する式を取り、各国語文字セットで戻します。データベースの各国語文字セットは、AL16UTF16またはUTF8です。UNISTRでは、文字列への文字のUnicodeエンコーディング値の指定を可能にすることによって、Unicode文字列リテラルがサポートされます。これは、NCHAR列にデータを挿入する場合などに有効です。

Unicodeエンコーディング値は、¥xxxxという形式です。xxxxは、文字のUCS-2エンコーディング形式での16進数値です。補足文字は2つのコード・ユニットとしてエンコードされ、最初のコード・ユニットは上位サロゲート範囲(U+D800からU+DBFF)から、2番目のコード・ユニットは下位サロゲート範囲(U+DC00からU+DFFF)から取得されます。文字列そのものにバックスラッシュを含めるには、文字列の先頭に別のバックスラッシュ(¥¥)を付けます。

移植性およびデータ保護のために、UNISTR文字列の引数にはASCII文字およびUnicodeエンコーディング値のみを指定することをお薦めします。

関連項目:

- Unicodeおよび各国語文字セットの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。
- UNISTRの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

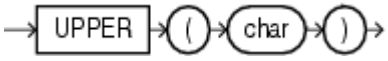
例

次の例では、ASCII文字およびUnicodeエンコーディング値をUNISTRファンクションに渡した後、そのUNISTRファンクションによって各国語文字セットで文字列が戻されます。

```
SELECT UNISTR('abc¥00e5¥00f1¥00f6') FROM DUAL;
UNISTR
------
abcåñö
```

UPPER

構文



目的

UPPERは、すべての文字を大文字にしてcharを戻します。charは、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOBまたはNCLOBデータ型です。戻り値は、charと同じデータ型です。データベースは、基礎となる文字セットに対して定義したバイナリ・マッピングに基づいて文字の形式を設定します。大文字の区別については、「NLS_UPPER」を参照してください。

関連項目:

UPPERの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

例

次の例では、各従業員の姓を大文字で戻します。

```
SELECT UPPER(last_name) "Uppercase"
   FROM employees;
```

USER

構文

目的

USERは、セッション・ユーザー(ログオンしているユーザー)の名前を返します。これは、Real Application Securityセッションがアタッチまたはデタッチされると、データベース・セッション中に変化することがあります。エンタープライズ・ユーザーの場合、このファンクションはスキーマを返します。その他のユーザーの場合、データベース・ユーザー名を返します。現在のデータベース・セッションにReal Application Securityがアタッチされている場合は、ユーザーXS$NULLを返します。

このファンクションは、VARCHAR2値を返します。

Oracle Databaseは、空白埋め比較セマンティクスでこのファンクションの値を比較します。

分散SQL文では、UIDファンクションおよびUSERファンクションは、ローカル・データベース上のユーザーを識別します。CHECK制約の条件でこれらのファンクションは使用できません。

関連項目:

- ユーザーXS$NULLの詳細は、『Oracle Database 2日でセキュリティ・ガイド』を参照してください。
- USERの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

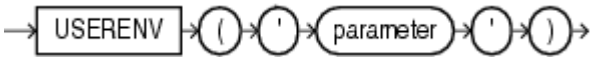
例

次の例では、セッション・ユーザーとユーザーのUIDが返されます。

```
SELECT USER, UID FROM DUAL;
```

USERENV

構文



目的

> ノート:
>
> USERENV は、下位互換用に保持されるレガシー・ファンクションです。現行の機能に対して組込み USERENV ネームスペースとともに SYS_CONTEXT ファンクションを使用することをお薦めします。詳細は、「SYS_CONTEXT」を参照してください。

USERENVは現行のセッションについての情報を戻します。この情報は、アプリケーション固有の監査証跡表を書き込む場合、またはセッションで現在使用されている言語固有の文字を判断する場合に有効です。CHECK制約の条件で、USERENVは使用できません。表7-12に、parameter引数の値を示します。

SESSIONID、SIDおよびENTRYIDパラメータを使用するコール(NUMBERを戻す)以外のUSERENVへのすべてのコールは、VARCHAR2データを戻します。

表7-12 USERENVファンクションのパラメータ

パラメータ	戻り値
CLIENT_INFO	CLIENT_INFO は、DBMS_APPLICATION_INFO パッケージを使用するアプリケーションが格納できる 64 バイトまでのユーザー・セッション情報を戻します。 注意: 商業用のアプリケーションによっては、このコンテキスト値を使用する可能性があります。このコンテキスト領域の使用に対する制限については、これらのアプリケーションのドキュメントを参照してください。 関連項目: アプリケーション・コンテキストの詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。「CREATE CONTEXT」および「SYS_CONTEXT」も参照してください。
ENTRYID	現行の監査エントリ番号を戻します。監査エントリ ID の順序は、ファイングレイン監査レコードと通常の監査レコードで共通です。この属性を分散 SQL 文で使用することはできません。
ISDBA	ISDBA は、オペレーティング・システムまたはパスワード・ファイルによって、ユーザーが DBA 権限を持っていると認証された場合、'TRUE'を戻します。
LANG	LANG は、言語名の ISO 略称を戻します。これは、既存の'LANGUAGE'パラメータを短縮したものです。
LANGUAGE	LANGUAGE は、現行のセッションで使用している言語(language)および地域(territory)を、データベース文字セット(character set)も含めた次の書式で戻します。

パラメータ	戻り値
	language_territory.characterset
SESSIONID	SESSIONID は、監査セッション識別子を戻します。このパラメータを分散 SQL 文で指定することはできません。
SID	SID は、セッション ID を戻します。
TERMINAL	TERMINAL は、現行のセッションの端末に対するオペレーティング・システム識別子を戻します。分散 SQL 文では、このパラメータはローカル・セッションの識別子を戻します。分散環境では、リモートの SELECT に対してのみこのパラメータを使用でき、リモートの INSERT、UPDATE または DELETE には使用できません。

関連項目:

USERENVの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

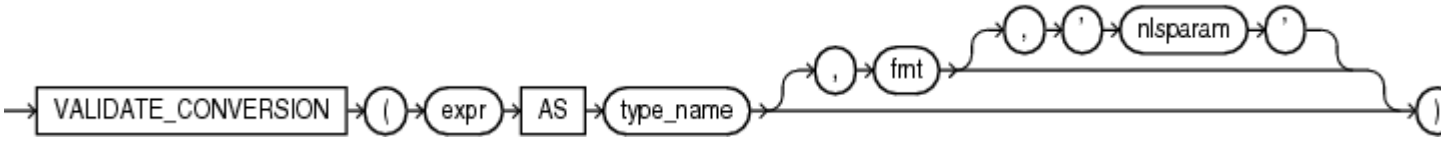
例

次の例では、現行のセッションのLANGUAGEパラメータを戻します。

```
SELECT USERENV('LANGUAGE') "Language" FROM DUAL;
Language
----------------------------------
AMERICAN_AMERICA.WE8ISO8859P1
```

VALIDATE_CONVERSION

構文



目的

VALIDATE_CONVERSIONは、exprが指定されたデータ型に変換できるかどうかを決定します。exprを正常に変換できる場合、このファンクションは1を戻し、それ以外の場合は0を戻します。exprがNULLと評価される場合、このファンクションは1を戻します。exprの評価中にエラーが発生した場合、このファンクションでエラーが戻されます。

exprには、SQL式を指定します。exprで許容されるデータ型と、オプションのfmtおよびnlsparamの引数の用途は、type_nameに指定したデータ型に応じて異なります。

type_nameには、exprの変換先のデータ型を指定します。次のデータ型を指定できます。

- BINARY_DOUBLE

  BINARY_DOUBLEを指定した場合、exprは、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のデータ型の文字列、あるいはNUMBER、BINARY_FLOATまたはBINARY_DOUBLEの型の数値と評価される任意の式にすることが可能です。オプションのfmt引数およびnlsparam引数の用途は、TO_BINARY_DOUBLEファンクションと同じです。詳細は、「TO_BINARY_DOUBLE」を参照してください。

- BINARY_FLOAT

  BINARY_FLOATを指定した場合、exprは、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のデータ型の文字列、あるいはNUMBER、BINARY_FLOATまたはBINARY_DOUBLEの型の数値と評価される任意の式にすることが可能です。オプションのfmt引数およびnlsparam引数の用途は、TO_BINARY_FLOATファンクションと同じです。詳細は、「TO_BINARY_FLOAT」を参照してください。

- DATE

  DATEを指定した場合、exprは、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のデータ型の文字列と評価される任意の式にすることができます。オプションのfmt引数およびnlsparam引数の用途は、TO_DATEファンクションと同じです。詳細は、「TO_DATE」を参照してください。

- INTERVAL DAY TO SECOND

  INTERVAL DAY TO SECONDを指定した場合、exprは、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のデータ型の文字列と評価される任意の式にすることができ、SQL期間書式またはISO存続期間書式の値を含める必要があります。このデータ型には、オプションのfmt引数およびnlsparam引数は適用されません。SQL期間書式およびISO存続期間書式の詳細は、「TO_DSINTERVAL」を参照してください。

- INTERVAL YEAR TO MONTH

  INTERVAL YEAR TO MONTHを指定した場合、exprは、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のデータ型の文字列と評価される任意の式にすることができ、SQL期間書式またはISO存続期間書式の値を含める必要があります。このデータ型には、オプションのfmt引数およびnlsparam引数は適用されません。SQL期間書式およびISO存続期間書式の詳細は、「TO_YMINTERVAL」を参照してください。

- NUMBER

NUMBERを指定した場合、exprは、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のデータ型の文字列、あるいはNUMBER、BINARY_FLOATまたはBINARY_DOUBLEの型の数値と評価される任意の式にすることが可能です。オプションのfmt引数およびnlsparam引数の用途は、TO_NUMBERファンクションと同じです。詳細は、「TO_NUMBER」を参照してください。

exprがNUMBER型の値の場合、VALIDATE_CONVERSIONファンクションでは、exprが有効な数値であることを検証します。exprが有効な数値でない場合は、0を戻します。これにより、データベース内の破損した数値を識別できます。

- TIMESTAMP

  TIMESTAMPを指定した場合、exprは、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のデータ型の文字列と評価される任意の式にすることができます。オプションのfmt引数およびnlsparam引数の用途は、TO_TIMESTAMPファンクションと同じです。fmtを省略すると、exprはNLS_TIMESTAMP_FORMAT初期化パラメータによって決定された、TIMESTAMPのデータ型のデフォルト形式である必要があります。詳細は、「TO_TIMESTAMP」を参照してください。

- TIME WITH TIME ZONE

  TIMESTAMP WITH TIME ZONEを指定した場合、exprは、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のデータ型の文字列と評価される任意の式にすることができます。オプションのfmt引数およびnlsparam引数の用途は、TO_TIMESTAMP_TZファンクションと同じです。fmtを省略すると、exprはNLS_TIMESTAMP_TZ_FORMAT初期化パラメータによって決定された、TIMESTAMP WITH TIME ZONEのデータ型のデフォルト形式である必要があります。詳細は、「TO_TIMESTAMP_TZ」を参照してください。

- TIMESTAMP WITH LOCAL TIME ZONE

  TIMESTAMPを指定した場合、exprは、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のデータ型の文字列と評価される任意の式にすることができます。オプションのfmt引数およびnlsparam引数の用途は、TO_TIMESTAMPファンクションと同じです。fmtを省略すると、exprはNLS_TIMESTAMP_FORMAT初期化パラメータによって決定された、TIMESTAMPのデータ型のデフォルト形式である必要があります。詳細は、「TO_TIMESTAMP」を参照してください。

例

次の各文では、指定した値を指定したデータ型に正常に変換できます。そのため、これらの各文では1の値が戻されます。

```
SELECT VALIDATE_CONVERSION(1000 AS BINARY_DOUBLE)
  FROM DUAL;
SELECT VALIDATE_CONVERSION('1234.56' AS BINARY_FLOAT)
  FROM DUAL;
SELECT VALIDATE_CONVERSION('July 20, 1969, 20:18' AS DATE,
    'Month dd, YYYY, HH24:MI', 'NLS_DATE_LANGUAGE = American')
  FROM DUAL;
SELECT VALIDATE_CONVERSION('200 00:00:00' AS INTERVAL DAY TO SECOND)
  FROM DUAL;
SELECT VALIDATE_CONVERSION('P1Y2M' AS INTERVAL YEAR TO MONTH)
  FROM DUAL;
SELECT VALIDATE_CONVERSION('$100,00' AS NUMBER,
    '$999D99', 'NLS_NUMERIC_CHARACTERS = '',.''')
  FROM DUAL;
SELECT VALIDATE_CONVERSION('29-Jan-02 17:24:00' AS TIMESTAMP,
    'DD-MON-YY HH24:MI:SS')
  FROM DUAL;
SELECT VALIDATE_CONVERSION('1999-12-01 11:00:00 -8:00'
    AS TIMESTAMP WITH TIME ZONE, 'YYYY-MM-DD HH:MI:SS TZH:TZM')
  FROM DUAL;
SELECT VALIDATE_CONVERSION('11-May-16 17:30:00'
```

```
    AS TIMESTAMP WITH LOCAL TIME ZONE, 'DD-MON-YY HH24:MI:SS')
  FROM DUAL;
```

次の文では、指定した値をBINARY_FLOATに変換できないため、0が戻されます。

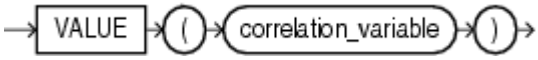
```
SELECT VALIDATE_CONVERSION('$29.99' AS BINARY_FLOAT)
  FROM DUAL;
```

次の文では、指定した数値書式モデルによって値をBINARY_FLOATに変換できるため、1が戻されます。

```
SELECT VALIDATE_CONVERSION('$29.99' AS BINARY_FLOAT, '$99D99')
  FROM DUAL;
```

VALUE

構文



目的

VALUEは、引数としてオブジェクト表の行に関連付けられた相関変数(表別名)を取り、オブジェクト表に格納されたオブジェクト・インスタンスを戻します。オブジェクト・インスタンスの型は、オブジェクト表と同じ型です。

例

次の例では、「置換可能な表および列のサンプル」で作成されたoe.personsサンプル表を使用します。

```
SELECT VALUE(p) FROM persons p;
VALUE(P)(NAME, SSN)
-----------------------------------------------------------
PERSON_T('Bob', 1234)
EMPLOYEE_T('Joe', 32456, 12, 100000)
PART_TIME_EMP_T('Tim', 5678, 13, 1000, 20)
```

関連項目:

VALUEファンクションでのIS OF type条件の使用方法の詳細は、「IS OF type条件」を参照してください

VAR_POP

構文

関連項目:

構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

VAR_POPは、数値の集合にあるNULLを削除した後、この集合の母集団分散を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を返します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

ファンクションが空の集合に適用されると、NULLを戻します。このファンクションは、次の計算を行います。

```
SUM((expr - (SUM(expr) / COUNT(expr)))2) / COUNT(expr)
```

関連項目:

exprの有効な書式の詳細は、「SQL式」を参照してください。また、「集計ファンクション」を参照してください。

集計の例

次の例では、employees表にある給与の母集団分散を戻します。

```
SELECT VAR_POP(salary) FROM employees;
VAR_POP(SALARY)
---------------
     15141964.9
```

分析の例

次の例では、sh.sales表での1998年における月ごとの売上について、累積の母集団および標本分散を計算します。

```
SELECT t.calendar_month_desc,
   VAR_POP(SUM(s.amount_sold))
      OVER (ORDER BY t.calendar_month_desc) "Var_Pop",
   VAR_SAMP(SUM(s.amount_sold))
      OVER (ORDER BY t.calendar_month_desc) "Var_Samp"
  FROM sales s, times t
  WHERE s.time_id = t.time_id AND t.calendar_year = 1998
  GROUP BY t.calendar_month_desc
  ORDER BY t.calendar_month_desc, "Var_Pop", "Var_Samp";
CALENDAR    Var_Pop   Var_Samp
```

```
-------- ---------- ----------
1998-01           0
1998-02  2269111326 4538222653
1998-03  5.5849E+10 8.3774E+10
1998-04  4.8252E+10 6.4336E+10
1998-05  6.0020E+10 7.5025E+10
1998-06  5.4091E+10 6.4909E+10
1998-07  4.7150E+10 5.5009E+10
1998-08  4.1345E+10 4.7252E+10
1998-09  3.9591E+10 4.4540E+10
1998-10  3.9995E+10 4.4439E+10
1998-11  3.6870E+10 4.0558E+10
1998-12  4.0216E+10 4.3872E+10
```

VAR_SAMP

構文

関連項目:

構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

VAR_SAMPは、数値の集合にあるNULLを削除した後、この集合の標本分散を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を返します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

ファンクションが空の集合に適用されると、NULLを戻します。このファンクションは、次の計算を行います。

```
(SUM(expr - (SUM(expr) / COUNT(expr)))2) / (COUNT(expr) - 1)
```

このファンクションは、1つの要素の集合を入力するとVARIANCEは0を戻し、VAR_SAMPはNULLを戻すという点を除いては、VARIANCEに似ています。

関連項目:

exprの有効な書式の詳細は、「SQL式」を参照してください。また、「集計ファンクション」を参照してください。

集計の例

次の例では、サンプル表employeesにある給与の標本分散を計算します。

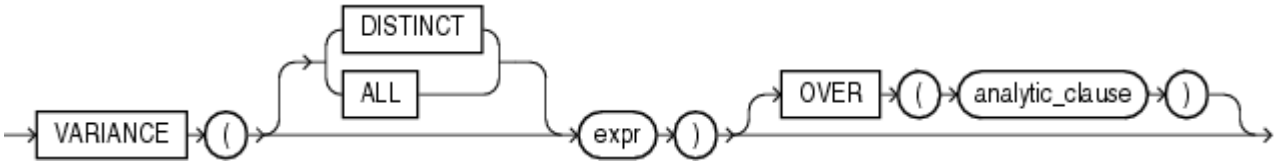
```
SELECT VAR_SAMP(salary) FROM employees;
VAR_SAMP(SALARY)
----------------
      15284813.7
```

分析の例

「VAR_POP」の分析の例を参照してください。

VARIANCE

構文



関連項目:

構文、セマンティクスおよび制限事項の詳細は、「分析ファンクション」を参照してください。

目的

VARIANCEはexprの分散を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

exprの分散の計算結果は、次のようになります。

- exprの行数が1の場合は0(ゼロ)
- exprの行数が1を超える場合はVAR_SAMP

DISTINCTを指定する場合は、analytic_clauseのquery_partition_clauseのみを指定できます。order_by_clauseおよびwindowing_clauseは指定できません。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を返します。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。exprの書式の詳細は、「SQL式」を参照してください。「集計ファンクション」も参照してください。

集計の例

次の例では、サンプル表employeesにあるすべての給与の分散を計算します。

```
SELECT VARIANCE(salary) "Variance"
   FROM employees;
  Variance
----------
15283140.5
```

分析の例

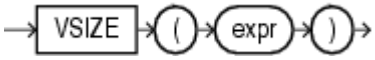
次の例では、雇用日で順序付けられた部門30の給与の値の累積分散を戻します。

```
SELECT last_name, salary, VARIANCE(salary)
      OVER (ORDER BY hire_date) "Variance"
   FROM employees
   WHERE department_id = 30
   ORDER BY last_name, salary, "Variance";
LAST_NAME                 SALARY   Variance
------------------------- ---------- ----------
Baida                           2900 16283333.3
```

Colmenares	2500	11307000
Himuro	2600	13317000
Khoo	3100	31205000
Raphaely	11000	0
Tobias	2800	21623333.3

VSIZE

構文



目的

VSIZEは、exprの内部表現でのバイト数を戻します。exprがNULLの場合はNULLを戻します。

このファンクションは、CLOBデータを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用してCLOBを引数として渡すことはできます。

関連項目:

詳細は、「データ型の比較規則」を参照してください。

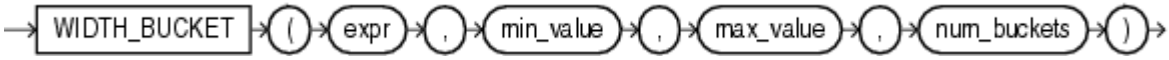
例

次の例では、部門10の従業員のlast_name列のバイト数を戻します。

```
SELECT last_name, VSIZE (last_name) "BYTES"
  FROM employees
  WHERE department_id = 10
  ORDER BY employee_id;

LAST_NAME            BYTES
-------------- ----------
Whalen                    6
```

WIDTH_BUCKET

構文



目的

WIDTH_BUCKETを使用すると、ヒストグラムの幅が同じサイズに分割された等幅ヒストグラムを作成できます。このファンクションと、等高ヒストグラムを作成するNTILEを比較してください。各バケットが実際の数値線幅のclosed-open間隔であることが理想です。たとえば、間隔に10が含まれ20が排除されることを示すために、バケットは10.00から19.999...のスコアに割り当てられます。これは、[10, 20]と表すこともできます。

指定された式に対して、WIDTH_BUCKETは、この式の値が評価された後に該当するバケット数を戻します。

- exprは、ヒストグラムが作成される式です。この式は、数値または日時値、あるいは数値または日時値に暗黙的に変換可能な値に評価される必要があります。exprがNULLと評価された場合、式はNULLを戻します。
- min_valueおよびmax_valueは、exprの許容域のエンド・ポイントに解決される式です。これらの式は両方とも数値または日時値に評価される必要があり、いずれもNULLに評価されません。
- num_bucketsは、バケット数を示す定数に解決される式です。この式は、正の整数に評価される必要があります。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。

Oracle Databaseは、必要に応じて、0(ゼロ)の下位バケットおよびnum_buckets+1の上位バケットを作成します。これらのバケットは、min_value未満およびmax_valueより大きい値を処理し、エンド・ポイントの妥当性チェックに有効です。

例

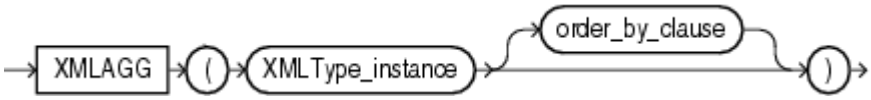
次の例では、サンプル表oe.customersのスイスの顧客のcredit_limit列に10バケットのヒストグラムを作成し、各顧客のバケット数(「クレジット・グループ」)を戻します。最大値以上のクレジット利用限度額を持つ顧客は、上位バケット11に割り当てられます。

```
SELECT customer_id, cust_last_name, credit_limit,
   WIDTH_BUCKET(credit_limit, 100, 5000, 10) "Credit Group"
   FROM customers WHERE nls_territory = 'SWITZERLAND'
   ORDER BY "Credit Group", customer_id, cust_last_name, credit_limit;
CUSTOMER_ID CUST_LAST_NAME       CREDIT_LIMIT Credit Group
----------- -------------------- ------------ ------------
        825 Dreyfuss                      500            1
        826 Barkin                        500            1
        827 Siegel                        500            1
        853 Palin                         400            1
        843 Oates                         700            2
        844 Julius                        700            2
        835 Eastwood                     1200            3
        836 Berenger                     1200            3
        837 Stanton                      1200            3
        840 Elliott                      1400            3
        841 Boyer                        1400            3
        842 Stern                        1400            3
        848 Olmos                        1800            4
        849 Kaurusmdki                   1800            4
```

828	Minnelli	2300	5
829	Hunter	2300	5
850	Finney	2300	5
851	Brown	2300	5
852	Tanner	2300	5
830	Dutt	3500	7
831	Bel Geddes	3500	7
832	Spacek	3500	7
833	Moranis	3500	7
834	Idle	3500	7
838	Nicholson	3500	7
839	Johnson	3500	7
845	Fawcett	5000	11
846	Brando	5000	11
847	Streep	5000	11

XMLAGG

構文



目的

XMLAggは集計ファンクションです。XMLフラグメントのコレクションを取り、集計されたXML文書を戻します。NULLを戻す引数は結果から排除されます。

XMLAggは、ノードのコレクションを戻す点を除いて、SYS_XMLAggに似ていますが、XMLFormatオブジェクトを使用した書式設定は受け入れません。また、XMLAGGは、SYS_XMLAGGとは異なり、出力を要素タグで囲みません。

このorder_by_clauseにかぎり、他の使用方法とは異なり、数値リテラルは列の位置として認識されず、単に数値リテラルとして解釈されます。

関連項目:

「XMLELEMENT」および「SYS_XMLAGG」を参照してください。

例

次の例では、従業員のジョブIDと姓を要素の内容とするEmployee要素を含むDepartment要素を生成します。

```
SELECT XMLELEMENT("Department",
   XMLAGG(XMLELEMENT("Employee",
   e.job_id||' '||e.last_name)
   ORDER BY last_name))
   as "Dept_list"
   FROM employees e
   WHERE e.department_id = 30;
Dept_list
----------------------------------------------------------------
<Department>
  <Employee>PU_CLERK Baida</Employee>
  <Employee>PU_CLERK Colmenares</Employee>
  <Employee>PU_CLERK Himuro</Employee>
  <Employee>PU_CLERK Khoo</Employee>
  <Employee>PU_MAN Raphaely</Employee>
  <Employee>PU_CLERK Tobias</Employee>
</Department>
```

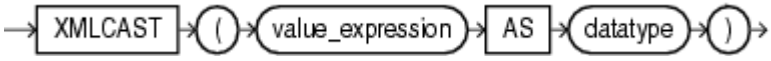
XMLAGGが行を集計するため、結果は単一行となります。GROUP BY句を使用して、戻された行の集合を複数のグループにまとめることができます。

```
SELECT XMLELEMENT("Department",
       XMLAGG(XMLELEMENT("Employee", e.job_id||' '||e.last_name)))
   AS "Dept_list"
   FROM employees e
   GROUP BY e.department_id;
Dept_list
-----------------------------------------------------------
<Department>
  <Employee>AD_ASST Whalen</Employee>
</Department>
<Department>
```

```
  <Employee>MK_MAN Hartstein</Employee>
  <Employee>MK_REP Fay</Employee>
</Department>
<Department>
  <Employee>PU_MAN Raphaely</Employee>
  <Employee>PU_CLERK Khoo</Employee>
  <Employee>PU_CLERK Tobias</Employee>
  <Employee>PU_CLERK Baida</Employee>
  <Employee>PU_CLERK Colmenares</Employee>
  <Employee>PU_CLERK Himuro</Employee>
</Department>
. . .
```

XMLCAST

構文



目的

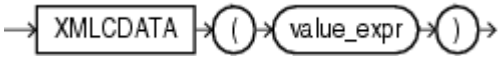
XMLCastは、value_expressionをdatatypeで指定されたスカラーSQLデータ型にキャストします。value_expression引数は、評価されるSQL式です。datatype引数にはNUMBER、VARCHAR2、CHAR、CLOB、BLOBおよびREF XMLTYPEデータ型、任意の日時データ型を指定できます。

関連項目:

- このファンクションの使用の詳細および例は、『Oracle XML DB開発者ガイド』を参照してください。
- XMLCASTの戻り値が文字値である場合に、それに割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

XMLCDATA

構文



目的

XMLCDataは、value_exprを評価してCDATAセクションを生成します。value_exprは文字列に変換する必要があります。このファンクションの戻り値は、次の書式になります。

```
<![CDATA[string]]>
```

結果値が無効なXML CDATAセクションの場合は、エラーが戻されます。次の条件がXMLCDataに適用されます。

- value_exprには、サブストリング]]>を含めることはできません。
- value_exprがNULLと評価された場合、このファンクションはNULLを戻します。

関連項目:

このファンクションの詳細は、『Oracle XML DB開発者ガイド』を参照してください。

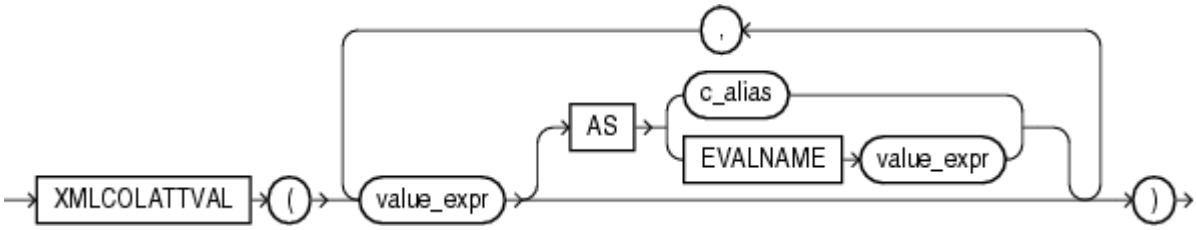
例

次の文は、DUAL表を使用して、XMLCDataの構文を示します。

```
SELECT XMLELEMENT("PurchaseOrder",
   XMLAttributes(dummy as "pono"),
   XMLCdata('<!DOCTYPE po_dom_group [
   <!ELEMENT po_dom_group(student_name)*>
   <!ELEMENT po_purch_name (#PCDATA)>
   <!ATTLIST po_name po_no ID #REQUIRED>
   <!ATTLIST po_name trust_1 IDREF #IMPLIED>
   <!ATTLIST po_name trust_2 IDREF #IMPLIED>
   ]>')) "XMLCData" FROM DUAL;

XMLCData
------------------------------------------------------------
<PurchaseOrder pono="X"><![CDATA[
<!DOCTYPE po_dom_group [
   <!ELEMENT po_dom_group(student_name)*>
   <!ELEMENT po_purch_name (#PCDATA)>
   <!ATTLIST po_name po_no ID #REQUIRED>
   <!ATTLIST po_name trust_1 IDREF #IMPLIED>
   <!ATTLIST po_name trust_2 IDREF #IMPLIED>
   ]>
  ]]>
</PurchaseOrder>
```

XMLCOLATTVAL

構文



目的

XMLCOLATTVALは、XMLフラグメントを作成し、各XMLフラグメントの名前がcolumn、属性がnameとなるように、結果としてできたXMLを展開します。

AS句を使用して、name属性の値を列名以外の値に変更できます。この場合、文字列リテラルのc_aliasを指定するか、またはEVALNAME value_exprを指定します。後者の場合は、値の式が評価され、その結果(文字列リテラル)が別名として使用されます。この別名は、初期化パラメータがMAX_STRING_SIZE = STANDARDの場合は最大4,000文字になります。また、MAX_STRING_SIZE = EXTENDEDの場合は最大32,767文字になります。詳細は、「拡張データ型」を参照してください。

value_exprの値を指定する必要があります。value_exprがNULLの場合、要素は戻されません。

XMLColAttValの制限事項

value_exprにはオブジェクト型の列を指定できません。

例

次の例では、従業員のサブセットに対してEmp要素を作成します。このとき、Empの内容として、employee_id、last_nameおよびsalaryの各要素がネストされます。ネストされた各要素にはcolumnという名前が付き、属性値として列名を持つname属性が与えられます。

```
SELECT XMLELEMENT("Emp",
   XMLCOLATTVAL(e.employee_id, e.last_name, e.salary)) "Emp Element"
   FROM employees e
   WHERE employee_id = 204;
Emp Element
--------------------------------------------------------------------
<Emp>
  <column name="EMPLOYEE_ID">204</column>
  <column name="LAST_NAME">Baer</column>
  <column name="SALARY">10000</column>
</Emp>
```

これらの2つのファンクションの出力の比較については、「XMLFOREST」の例を参照してください。

XMLCOMMENT

構文

目的

XMLCommentは、value_exprの評価結果を使用してXMLコメントを生成します。value_exprは文字列に変換する必要があります。このファンクションには、2つの連続したダッシュ(ハイフン)を含めることはできません。このファンクションの戻り値は、次の書式になります。

```
<!--string-->
```

value_exprがNULLである場合、このファンクションはNULLを戻します。

関連項目:

このファンクションの詳細は、『Oracle XML DB開発者ガイド』を参照してください。

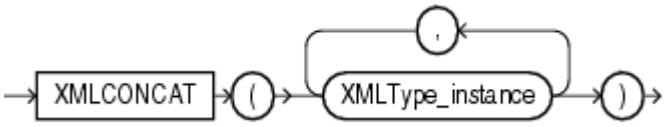
例

次の例は、DUAL表を使用して、XMLComment構文を示します。

```
SELECT XMLCOMMENT('OrderAnalysisComp imported, reconfigured, disassembled')
   AS "XMLCOMMENT" FROM DUAL;

XMLCOMMENT
--------------------------------------------------------------------------------
<!--OrderAnalysisComp imported, reconfigured, disassembled-->
```

XMLCONCAT

構文



目的

XMLCONCATは、入力として一連のXMLTypeインスタンスを取り、各行について一連の要素を連結し、連結した結果を戻します。XMLCONCATはXMLSEQUENCEの逆の処理をします。

NULL式は結果から排除されます。値のすべての式がNULLの場合、このファンクションはNULLを戻します。

関連項目:

XMLSEQUENCE

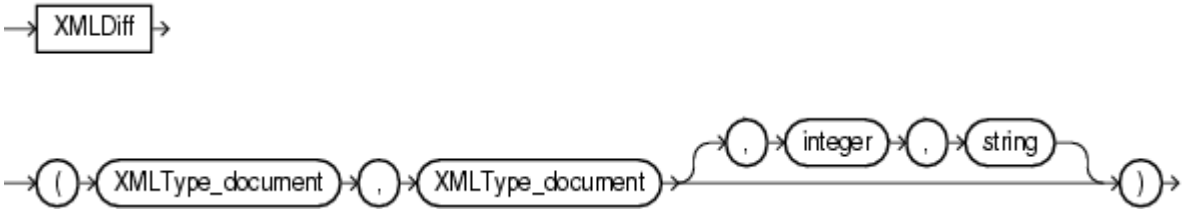
例

次の例では、従業員のサブセットの名前と姓に対してXML要素を作成し、作成した要素を連結して戻します。

```
SELECT XMLCONCAT(XMLELEMENT("First", e.first_name),
   XMLELEMENT("Last", e.last_name)) AS "Result"
   FROM employees e
   WHERE e.employee_id > 202;
Result
--------------------------------------------------------------------
<First>Susan</First>
<Last>Mavris</Last>
<First>Hermann</First>
<Last>Baer</Last>
<First>Shelley</First>
<Last>Higgins</Last>
<First>William</First>
<Last>Gietz</Last>
4 rows selected.
```

XMLDIFF

構文



目的

XMLDiffファンクションは、XmlDiff C APIのSQLインタフェースです。このファンクションは、2つのXML文書を比較し、Xdiffスキーマに基づいたXMLの差異を取得します。diff文書がXMLType文書として戻されます。

- 最初の2つの引数には、2つのXMLType文書の名前を指定します。
- integerには、CファンクションXmlDiffのhashLevelを表す数値を指定します。ハッシュを使用しない場合は、この引数を0(ゼロ)に設定するか、または引数全体を省略します。ハッシュを使用しないでフラグを指定する場合は、この引数を0(ゼロ)に設定する必要があります。
- stringには、ファンクションの動作を制御するフラグを指定します。これらのフラグは、セミコロンで区切られた1つ以上の名前で指定します。これらの名前は、XmlDiffファンクションの定数の名前と同じです。

関連項目:

このファンクションの使用方法と例については、『Oracle XML Developer's Kitプログラマーズ・ガイド』を参照してください。CのXML APIの詳細は、『Oracle Database XML C APIリファレンス』を参照してください

例

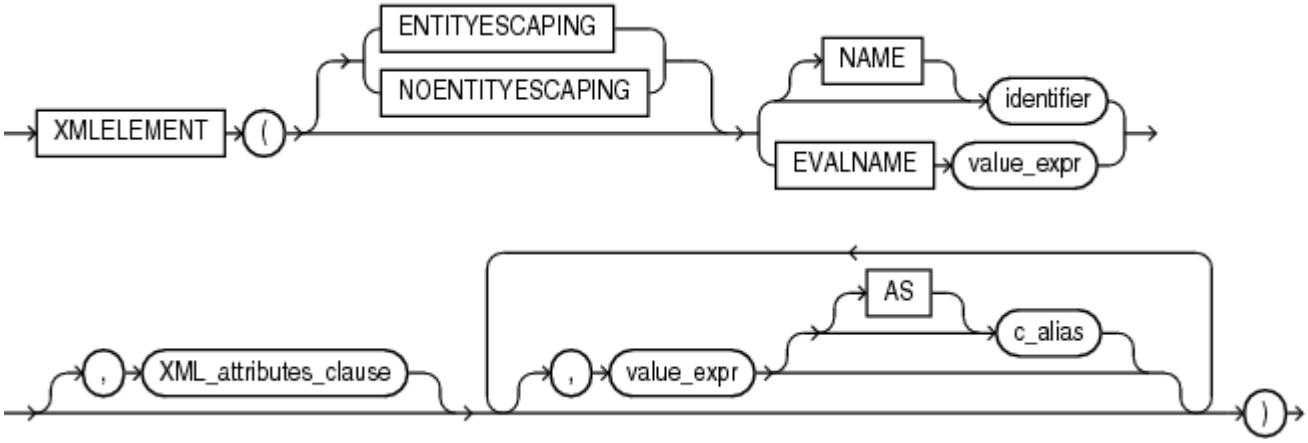
次の例では、2つのXML文書を比較し、差異をXMLType文書として戻します。

```
SELECT XMLDIFF(
XMLTYPE('<?xml version="1.0"?>
<bk:book xmlns:bk="http://example.com">
   <bk:tr>
        <bk:td>
                <bk:chapter>
                        Chapter 1.
                </bk:chapter>
        </bk:td>
        <bk:td>
                 <bk:chapter>
                        Chapter 2.
                </bk:chapter>
        </bk:td>
   </bk:tr>
</bk:book>'),
XMLTYPE('<?xml version="1.0"?>
<bk:book xmlns:bk="http://example.com">
   <bk:tr>
        <bk:td>
                <bk:chapter>
                        Chapter 1.
                </bk:chapter>
        </bk:td>
```

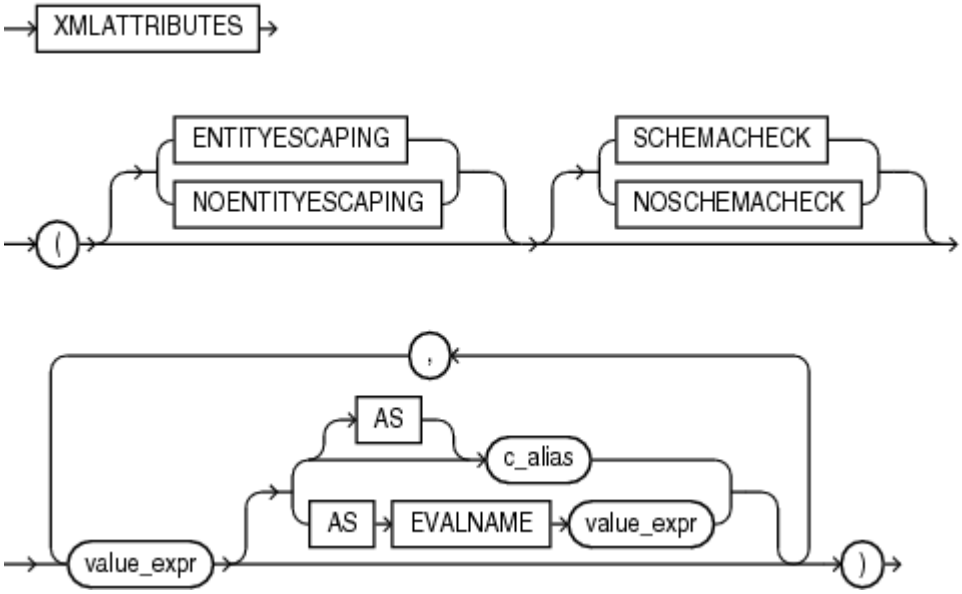
```
        <bk:td/>
   </bk:tr>
</bk:book>')
)
FROM DUAL;
```

XMLELEMENT

構文



XML_attributes_clause::=



目的

XMLElementは、identifierに対する要素名を取るか、またはEVALNAME value_exprの要素名、要素に対する属性のオプションのコレクション、およびその要素の内容を構成する引数を評価します。このファンクションはXMLType型のインスタンスを戻します。XMLElementは、戻されたXMLに属性を格納できる点を除いてSYS_XMLGenに似ていますが、XMLFormatオブジェクトを使用した書式設定は受け入れません。

次の項に示すとおり、通常は、XMLELEMENTファンクションはネストされており、ネスト構造のXML文書を生成します。

ENTITYESCAPINGおよびNONENTITYESCAPINGキーワードについては、『Oracle XML DB開発者ガイド』を参照してください。

囲みタグに使用するOracle Databaseの値を指定する必要があります。この場合、文字列リテラルのidentifierを指定するか、またはEVALNAME value_exprを指定します。後者の場合は、値の式が評価され、その結果(文字列リテラル)が識別子として使用されます。この識別子は、列名または列の参照である必要はありません。式またはNULLは指定できません。初期化パラメータがMAX_STRING_SIZE = STANDARDの場合は最大4,000文字になります。また、

MAX_STRING_SIZE = EXTENDEDの場合は最大32,767文字になります。

要素の内容を構成するオブジェクトは、XMLATTRIBUTESキーワードの後に指定します。XML_attributes_clauseでは、value_exprがNULLの場合、その値の式に対する属性は作成されません。value_exprの型に、オブジェクト型またはコレクションは指定できません。AS句を使用してvalue_exprに別名を指定すると、c_aliasまたは評価された値の式(EVALNAME value_expr)は、初期化パラメータがMAX_STRING_SIZE = STANDARDの場合は最大4,000文字まで指定可能になります。また、MAX_STRING_SIZE = EXTENDEDの場合は最大32,767文字まで指定可能になります。

関連項目:

MAX_STRING_SIZEの詳細は、「拡張データ型」を参照してください。

構文図のXML_attributes_clauseに続くオプションのvalue_exprは、次のようになります。

- value_exprがスカラー式である場合、AS句は省略できます。この場合、Oracleは列名を要素名として使用します。
- value_exprがオブジェクト型またはコレクションである場合、AS句は必須です。この場合、Oracleは指定されたc_aliasを囲みタグとして使用します。
- value_exprがNULLの場合、その値の式に対する要素は作成されません。

関連項目:

SYS_XMLGEN

例

次の例では、一連の従業員について、従業員の名前と雇用日を指定する、ネストされた要素を持つEmp要素を生成します。

```
SELECT XMLELEMENT("Emp", XMLELEMENT("Name",
   e.job_id||' '||e.last_name),
   XMLELEMENT("Hiredate", e.hire_date)) as "Result"
   FROM employees e WHERE employee_id > 200;
Result
--------------------------------------------------------------------
<Emp>
  <Name>MK_MAN Hartstein</Name>
  <Hiredate>2004-02-17</Hiredate>
</Emp>

<Emp>
  <Name>MK_REP Fay</Name>
  <Hiredate>2005-08-17</Hiredate>
</Emp>

<Emp>
  <Name>HR_REP Mavris</Name>
  <Hiredate>2002-06-07</Hiredate>
</Emp>

<Emp>
  <Name>PR_REP Baer</Name>
  <Hiredate>2002-06-07</Hiredate>
</Emp>

<Emp>
  <Name>AC_MGR Higgins</Name>
  <Hiredate>2002-06-07</Hiredate>
```

```
</Emp>

<Emp>
  <Name>AC_ACCOUNT Gietz</Name>
  <Hiredate>2002-06-07</Hiredate>
</Emp>
6 rows selected.
```

次の例では、XMLELEMENTファンクションにXML_attributes_clauseを使用して、トップレベル要素に対する属性値を持つ、ネストされたXML要素を作成します。

```
SELECT XMLELEMENT("Emp",
       XMLATTRIBUTES(e.employee_id AS "ID", e.last_name),
       XMLELEMENT("Dept", e.department_id),
       XMLELEMENT("Salary", e.salary)) AS "Emp Element"
   FROM employees e
   WHERE e.employee_id = 206;
Emp Element
--------------------------------------------------------------
<Emp ID="206" LAST_NAME="Gietz">
  <Dept>110</Dept>
  <Salary>8300</Salary>
</Emp>
```

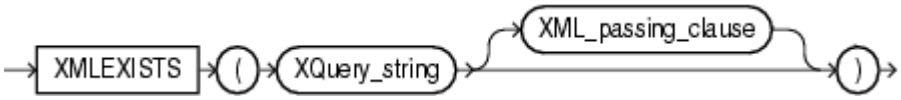
last_nameにはAS identifier句が指定されていません。その結果、戻されたXMLは、デフォルトで列名last_nameを使用します。

最後に、次の例では、XML_attributes_clauseに副問合せを使用して、別の表から要素の属性に情報を取り出します。

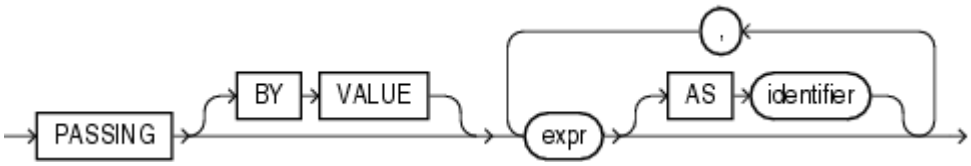
```
SELECT XMLELEMENT("Emp", XMLATTRIBUTES(e.employee_id, e.last_name),
   XMLELEMENT("Dept", XMLATTRIBUTES(e.department_id,
   (SELECT d.department_name FROM departments d
   WHERE d.department_id = e.department_id) as "Dept_name")),
   XMLELEMENT("salary", e.salary),
   XMLELEMENT("Hiredate", e.hire_date)) AS "Emp Element"
   FROM employees e
   WHERE employee_id = 205;
Emp Element
-----------------------------------------------------------------------
<Emp EMPLOYEE_ID="205" LAST_NAME="Higgins">
  <Dept DEPARTMENT_ID="110" Dept_name="Accounting"/>
  <salary>12008</salary>
  <Hiredate>2002-06-07</Hiredate>
</Emp>
```

XMLEXISTS

構文



XML_passing_clause::=



目的

XMLExistsは、指定されたXQuery式から空でないXQuery順序が戻されるかどうかをチェックします。戻される場合、このファンクションはTRUEを戻し、それ以外の場合はFALSEを戻します。引数XQuery_stringはリテラル文字列ですが、XML_passing_clauseを使用してバインドするXQuery変数を含めることができます。

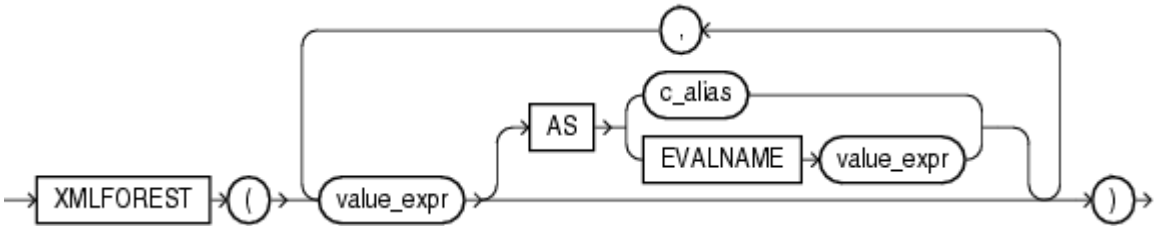
XML_passing_clauseのexprは、XMLTypeまたはSQLスカラー・データ型のインスタンスを戻し、XQuery式を評価するためのコンテキストとして使用されます。PASSING句には、識別子を指定せずに1つのexprのみを指定できます。各exprの評価結果は、XQuery_stringの対応する識別子にバインドされます。exprの後にAS句が続かない場合、式の評価結果はXQuery_stringの評価用のコンテキスト項目として使用されます。exprがリレーショナル列の場合、宣言された照合はOracle XML DBで無視されます。

関連項目:

このファンクションの使用の詳細および例は、『Oracle XML DB開発者ガイド』を参照してください。

XMLFOREST

構文



目的

XMLFORESTは、各引数パラメータをXMLに変換し、変換された引数を連結したXMLフラグメントを戻します。

- value_exprがスカラー式である場合、AS句は省略できます。この場合、Oracle Databaseは列名を要素名として使用します。
- value_exprがオブジェクト型またはコレクションである場合、AS句は必須です。この場合、Oracleは指定された式を囲みタグとして使用します。

  この場合、文字列リテラルのc_aliasを指定するか、またはEVALNAME value_exprを指定します。後者の場合は、値の式が評価され、その結果(文字列リテラル)が識別子として使用されます。この識別子は、列名または列の参照である必要はありません。式またはNULLは指定できません。初期化パラメータがMAX_STRING_SIZE = STANDARDの場合は最大4,000文字になります。また、MAX_STRING_SIZE = EXTENDEDの場合は最大32,767文字になります。詳細は、「拡張データ型」を参照してください。
- value_exprがNULLの場合、そのvalue_exprに対する要素は作成されません。

例

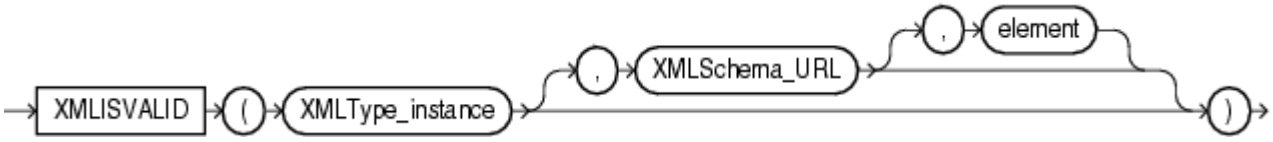
次の例では、従業員のサブセットに対してEmp要素を作成します。このとき、Empの内容として、employee_id、last_nameおよびsalaryの各要素がネストされます。

```
SELECT XMLELEMENT("Emp",
   XMLFOREST(e.employee_id, e.last_name, e.salary))
   "Emp Element"
   FROM employees e WHERE employee_id = 204;
Emp Element
--------------------------------------------------------------------
<Emp>
  <EMPLOYEE_ID>204</EMPLOYEE_ID>
  <LAST_NAME>Baer</LAST_NAME>
  <SALARY>10000</SALARY>
</Emp>
```

これらの2つのファンクションの出力の比較については、「XMLCOLATTVAL」の例を参照してください。

XMLISVALID

構文



目的

XMLISVALIDは、入力されたXMLType_instanceが、関連するXMLスキーマに準拠するかどうかをチェックします。XMLType_instanceに対して記録される検証ステータスは変更されません。

入力されたXMLドキュメントが有効であると判断された場合、XMLISVALIDは1を戻します。無効であると判断された場合は、0(ゼロ)を戻します。引数としてXMLSchema_URLを指定した場合、規格準拠のチェックに使用されます。指定しなかった場合、XMLドキュメントで指定されたXMLスキーマが規格準拠のチェックに使用されます。

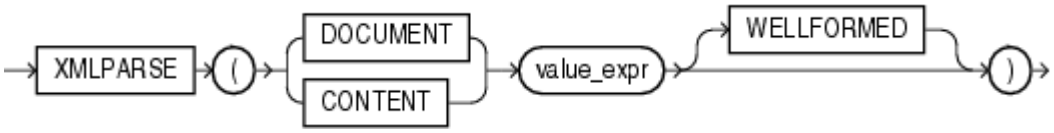
- XMLType_instanceは、検証されるXMLTypeインスタンスです。
- XMLSchema_URLは、規格準拠のチェックを行うXMLスキーマのURLです。
- elementは、規格準拠のチェックを行う指定されたスキーマの要素です。複数のトップレベル要素を定義するXMLスキーマを使用しており、これらの要素の特定の1つに対して規格準拠のチェックを行う場合に指定します。

関連項目:

このファンクションの使用の詳細および例は、『Oracle XML DB開発者ガイド』を参照してください

XMLPARSE

構文



目的

XMLParseは、value_exprの評価結果からXMLインスタンスを解析して生成します。value_exprは文字列に変換する必要があります。value_exprがNULLである場合、このファンクションはNULLを戻します。

14. DOCUMENTを指定する場合、value_exprは単一ルートのXML文書である必要があります。

15. CONTENTを指定する場合、value_exprは有効なXML値である必要があります。

16. WELLFORMEDを指定する場合、value_exprは整形式XML文書である必要があります。これは、入力が正しい書式であることを確認する、データベースによる妥当性チェックが実行されないためです。

関連項目:

このファンクションの詳細は、『Oracle XML DB開発者ガイド』を参照してください。

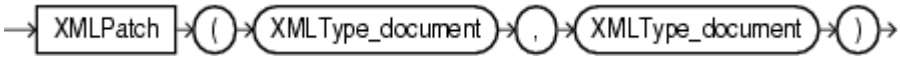
例

次の例は、DUAL表を使用して、XMLParseの構文を示します。

```
SELECT XMLPARSE(CONTENT '124 <purchaseOrder poNo="12435">
   <customerName> Acme Enterprises</customerName>
   <itemNo>32987457</itemNo>
   </purchaseOrder>'
WELLFORMED) AS PO FROM DUAL;

PO
--------------------------------------------------------------------
124 <purchaseOrder poNo="12435">
   <customerName> Acme Enterprises</customerName>
   <itemNo>32987457</itemNo>
   </purchaseOrder>
```

XMLPATCH

構文



目的

XMLPatchファンクションは、XmlPatch C APIのSQLインタフェースです。このファンクションは、指定された変更を適用してXML文書を修正します。修正されたXMLType文書が戻されます。

- 最初の引数には、入力XMLType文書の名前を指定します。
- 2番目の引数には、最初の文書に適用する変更を含むXMLType文書を指定します。変更は、Xdiff XMLスキーマに基づいている必要があります。Oracle XML Developer's KitのJavaメソッドdiff( )から、XML出力を指定できます。

関連項目:

このファンクションの使用方法と例については、『Oracle XML Developer's Kitプログラマーズ・ガイド』を参照してください。CのXML APIの詳細は、『Oracle Database XML C APIリファレンス』を参照してください

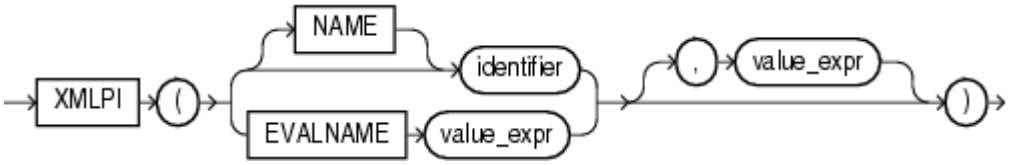
例

次の例では、別のXMLType文書に指定された変更を適用してXMLType文書を修正し、修正したXMLType文書を戻します。

```
SELECT XMLPATCH(
XMLTYPE('<?xml version="1.0"?>
<bk:book xmlns:bk="http://example.com">
   <bk:tr>
        <bk:td>
                <bk:chapter>
                        Chapter 1.
                </bk:chapter>
        </bk:td>
        <bk:td>
                 <bk:chapter>
                        Chapter 2.
                </bk:chapter>
        </bk:td>
   </bk:tr>
</bk:book>'),
XMLTYPE('<?xml version="1.0"?>
<xd:xdiff xsi:schemaLocation="http://xmlns.oracle.com/xdb/xdiff.xsd
  http://xmlns.oracle.com/xdb/xdiff.xsd"
  xmlns:xd="http://xmlns.oracle.com/xdb/xdiff.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:bk="http://example.com">
  <?oracle-xmldiff operations-in-docorder="true" output-model="snapshot"
    diff-algorithm="global"?>
  <xd:delete-node xd:node-type="element"
   xd:xpath="/bk:book[1]/bk:tr[1]/bk:td[2]/bk:chapter[1]"/>
</xd:xdiff>')
)
FROM DUAL;
```

XMLPI

構文



目的

XMLPIは、identifierとvalue_exprの評価結果(オプション)を使用して、XMLの処理命令を生成します。通常、処理命令は、XML文書のすべてまたは一部に関連付けられた情報をアプリケーションに提供するために使用されます。アプリケーションはこの処理命令を使用して、XML文書の最適な処理方法を決定します。

囲みタグに使用するOracle Databaseの値を指定する必要があります。この場合、文字列リテラルのidentifierを指定するか、またはEVALNAME value_exprを指定します。後者の場合は、値の式が評価され、その結果(文字列リテラル)が識別子として使用されます。この識別子は、列名または列の参照である必要はありません。式またはNULLは指定できません。初期化パラメータがMAX_STRING_SIZE = STANDARDの場合は最大4,000文字になります。また、MAX_STRING_SIZE = EXTENDEDの場合は最大32,767文字になります。詳細は、「拡張データ型」を参照してください。

オプションのvalue_exprは文字列に変換する必要があります。オプションのvalue_exprを指定しない場合、デフォルトは長さが0(ゼロ)の文字列です。このファンクションの戻り値は、次の書式になります。

```
<?identifier string?>
```

XMLPIには、次の制限事項があります。

- identifierは、処理命令の有効な対象である必要があります。
- xmlは、identifierと組み合せて指定することはできません。
- identifierには、連続文字?>を含めることはできません。

関連項目:

このファンクションの詳細は、『Oracle XML DB開発者ガイド』を参照してください。

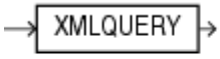
例

次の文は、DUAL表を使用して、XMLPIの構文の使用を示します。

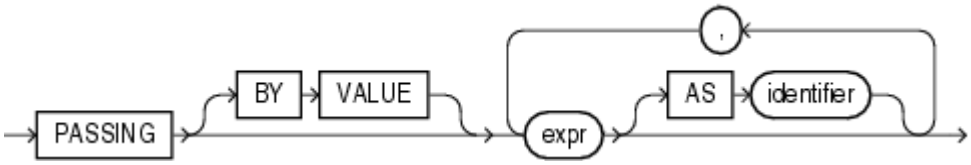
```
SELECT XMLPI(NAME "Order analysisComp", 'imported, reconfigured, disassembled')
   AS "XMLPI" FROM DUAL;

XMLPI
--------------------------------------------------------------------------------
<?Order analysisComp imported, reconfigured, disassembled?>
```

XMLQUERY

構文



XML_passing_clause::=



目的

XMLQUERYでは、SQL文のXMLデータを問い合せることができます。このファンクションは、文字列リテラル、オプションのコンテキスト項目、および他のバインド変数としてXQuery式を取り、これらの入力値を使用してXQuery式の評価結果を戻します。

- XQuery_stringは、プロローグを含む完全なXQuery式です。
- XML_passing_clauseのexprは、XMLTypeまたはSQLスカラー・データ型のインスタンスを戻し、XQuery式を評価するためのコンテキストとして使用されます。PASSING句には、識別子を指定せずに1つのexprのみを指定できます。各exprの評価結果は、XQuery_stringの対応する識別子にバインドされます。exprの後にAS句が続かない場合、式の評価結果はXQuery_stringの評価用のコンテキスト項目として使用されます。exprがリレーショナル列の場合、宣言された照合はOracle XML DBで無視されます。
- RETURNING CONTENTは、XQuery式の評価結果がXML 1.0文書か、またはXML 1.0セマンティクスに準拠したドキュメント・フラグメントのいずれかであることを示します。
- 結果セットが空の場合、このファンクションは、SQL NULL値を戻します。NULL ON EMPTYキーワードがデフォルトで実装されます。このキーワードは、意味を明確にするために示されます。

関連項目:

このファンクションの詳細は、『Oracle XML DB開発者ガイド』を参照してください。

例

次の文は、XML_passing_clauseのoe.warehouses表のwarehouse_spec列をコンテキスト項目として指定します。この文は、領域が50Kより大きいウェアハウスに関する特定の情報を戻します。

```
SELECT warehouse_name,
EXTRACTVALUE(warehouse_spec, '/Warehouse/Area'),
XMLQuery(
   'for $i in /Warehouse
   where $i/Area > 50000
   return <Details>
             <Docks num="{$i/Docks}"/>
             <Rail>
```

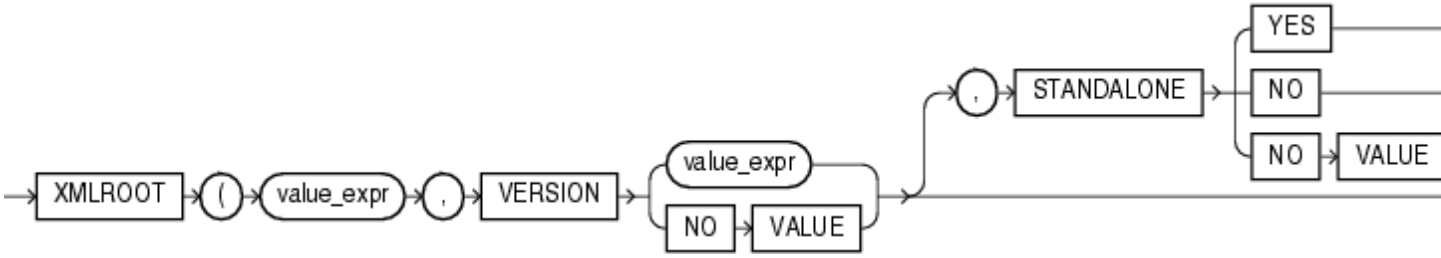
```
               {
               if ($i/RailAccess = "Y") then "true" else "false"
               }
            </Rail>
         </Details>' PASSING warehouse_spec RETURNING CONTENT) "Big_warehouses"
   FROM warehouses;
WAREHOUSE_ID Area      Big_warehouses
------------ --------- ------------------------------------------------------
           1     25000
           2     50000
           3     85700 <Details><Docks></Docks><Rail>false</Rail></Details>
           4    103000 <Details><Docks num="3"></Docks><Rail>true</Rail></Details>
 . . .
```

XMLROOT

ノート:

XMLROOT ファンクションは、非推奨です。これは、下位互換性を保つためにのみサポートされています。ただし、かわりに SQL/XML ファンクション XMLSERIALIZE を使用してバージョン番号を指定することをお薦めします。XMLSERIALIZE ファンクションの詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

構文



目的

XMLROOTでは、既存のXML値のXMLルート情報(プロローグ)のバージョンとスタンドアロンのプロパティを指定して、新しいXML値を作成できます。value_exprがすでにプロローグを持っている場合、エラーが戻されます。入力がNULLの場合、このファンクションはNULLを戻します。

戻り値は次の書式になります。

```
<?xml version = "version" [ STANDALONE = "{yes | no}" ]?>
```

- 最初のvalue_exprでは、XML値を指定します。この値に対してプロローグ情報を指定します。
- VERSION句のvalue_exprは、有効なXMLバージョンを表す文字列である必要があります。VERSIONにNO VALUEを指定すると、バージョンはデフォルトで1.0になります。
- オプションのSTANDALONE句を指定しない場合や、NO VALUEを指定した場合は、ファンクションの戻り値にスタンドアロンのプロパティは存在しません。

例

次の文は、DUAL表を使用して、XMLROOTの構文を示します。

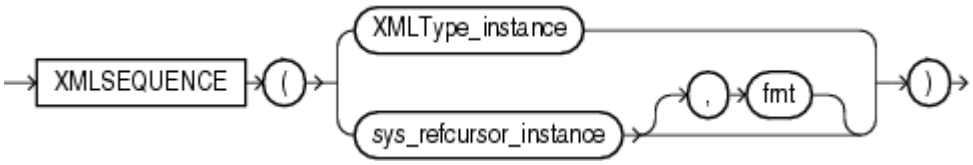
```
SELECT XMLROOT ( XMLType('<poid>143598</poid>'), VERSION '1.0', STANDALONE YES)
   AS "XMLROOT" FROM DUAL;
XMLROOT
--------------------------------------------------------------------------------
<?xml version="1.0" standalone="yes"?>
<poid>143598</poid>
```

XMLSEQUENCE

ノート:

XMLSEQUENCE ファンクションは、非推奨です。これは、下位互換性を保つためにのみサポートされています。ただし、かわりに XMLTABLE ファンクションを使用することをお薦めします。詳細は、「XMLTABLE」を参照してください。

構文



目的

XMLSEQUENCEには2つの書式があります。

4. 1つ目の書式は、入力としてXMLTypeインスタンスを取り、XMLTypeにあるトップレベルのノードのVARRAYを戻します。この書式のかわりに、SQL/XML標準ファンクションXMLTableを使用すると効果的です。このファンクションを使用すると、SQLコードがより読みやすくなります。Oracle Database 10gリリース2より前のリリースでは、このリリースのXMLTableファンクションでより効果的に実行されるいくつかの機能を、XMLSequenceがSQLファンクションTABLEを使用して実行していました。
5. 2つ目の書式は、入力としてREFCURSORインスタンスおよびオプションのXMLFormatオブジェクトのインスタンスを取り、カーソルの各行に対して、XMLSequence型としてXML文書を戻します。

XMLSEQUENCEはXMLTypeのコレクションを戻すため、このファンクションをTABLE句で使用して、コレクション値をネスト解除することで複数行にし、SQL問合せでの処理をさらに進めることができます。

関連項目:

このファンクションの詳細は、『Oracle XML DB開発者ガイド』を参照してください。また「XMLTABLE」も参照してください。

例

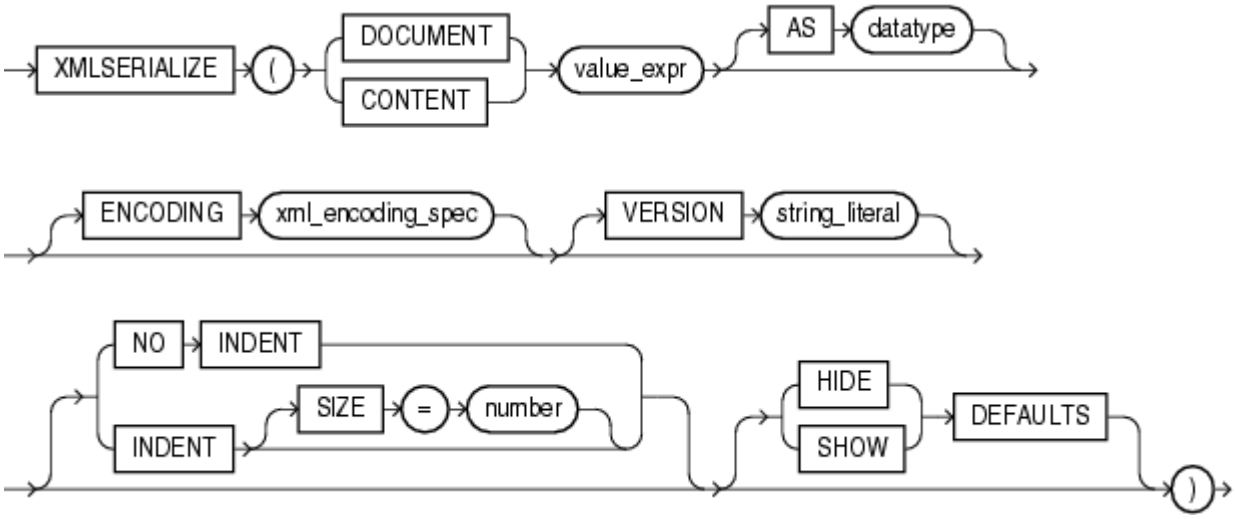
次の例では、XMLSEQUENCEが複数の要素を持つXML文書をVARRAY型の単一要素ドキュメントに分割する方法を示しています。この例では、TABLEキーワードが、コレクションを副問合せのFROM句で使用できる表の値とみなすように、Oracle Databaseに指示しています。

```
SELECT EXTRACT(warehouse_spec, '/Warehouse') as "Warehouse"
   FROM warehouses WHERE warehouse_name = 'San Francisco';
Warehouse
--------------------------------------------------------------
<Warehouse>
  <Building>Rented</Building>
  <Area>50000</Area>
  <Docks>1</Docks>
  <DockType>Side load</DockType>
  <WaterAccess>Y</WaterAccess>
  <RailAccess>N</RailAccess>
```

```
  <Parking>Lot</Parking>
  <VClearance>12 ft</VClearance>
</Warehouse>
1 row selected.
SELECT VALUE(p)
   FROM warehouses w,
   TABLE(XMLSEQUENCE(EXTRACT(warehouse_spec, '/Warehouse/*'))) p
   WHERE w.warehouse_name = 'San Francisco';
VALUE(P)
--------------------------------------------------------------------
<Building>Rented</Building>
<Area>50000</Area>
<Docks>1</Docks>
<DockType>Side load</DockType>
<WaterAccess>Y</WaterAccess>
<RailAccess>N</RailAccess>
<Parking>Lot</Parking>
<VClearance>12 ft</VClearance>
8 rows selected.
```

XMLSERIALIZE

構文



目的

XMLSerializeは、value_exprの内容を含む文字列またはLOBを作成します。

XMLSERIALIZEによって返されるすべてのLOBは読取り専用になります。

- DOCUMENTを指定する場合、value_exprは有効なXML文書である必要があります。
- CONTENTを指定する場合、value_exprは単一ルートのXML文書である必要はありません。ただし、有効なXMLコンテンツである必要があります。
- datatypeの指定には、文字列型(VARCHAR2またはVARCHARは使用可能、NVARCHAR2は使用不可)、BLOBまたはCLOBを使用できます。デフォルトはCLOBです。
- datatypeがBLOBの場合は、ENCODING句を指定して、指定したエンコーディングをプロローグで使用できます。xml_encoding_specは、XMLエンコーディング宣言(encoding="...")です。
- XML宣言でstring_literalとして指定したバージョン(<?xml version="..." ...?>)を使用するには、VERSION句を指定します。
- 意味のないすべての空白を出力から取り除くには、NO INDENTを指定します。出力をN個の空白の相対インデントを使用して出力整形するには、INDENT SIZE = Nを指定します。Nは整数です。Nが0の場合、出力整形によって各要素の後に改行文字が挿入され、各要素は独自の行に配置されますが、その他の意味のないすべての空白は出力で省略されます。SIZEを指定せずにINDENTを指定すると、2個の空白インデントが使用されます。この句を指定しない場合、動作(出力整形されるかどうか)は予測不能です。
- HIDE DEFAULTSおよびSHOW DEFAULTSは、XML Schemaに基づくデータにのみ適用されます。SHOW DEFAULTSを指定した場合、XML Schemaでデフォルト値が定義されているオプションの要素や属性が入力データに存在しないと、それらの要素や属性はそのデフォルト値とともに出力に含められます。HIDE DEFAULTSを指定した場合、このような要素や属性は出力に含められません。HIDE DEFAULTSがデフォルトの動作です。

関連項目:

- このファンクションの詳細は、『Oracle XML DB開発者ガイド』を参照してください。
- XMLSERIALIZEの文字の戻り値に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

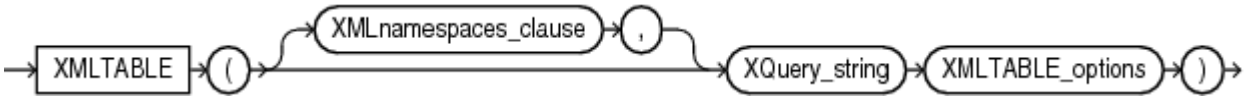
例

次の文は、DUAL表を使用して、XMLSerializeの構文を示します。

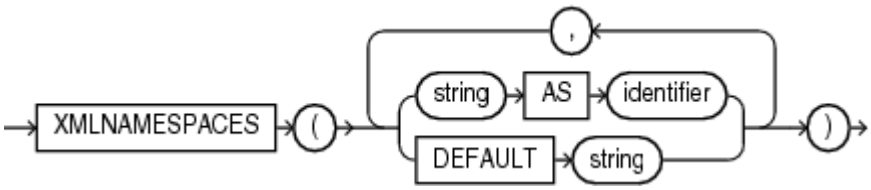
```
SELECT XMLSERIALIZE(CONTENT XMLTYPE('<Owner>Grandco</Owner>')) AS xmlserialize_doc
   FROM DUAL;
XMLSERIALIZE_DOC
----------------
<Owner>Grandco</Owner>
```

XMLTABLE

構文



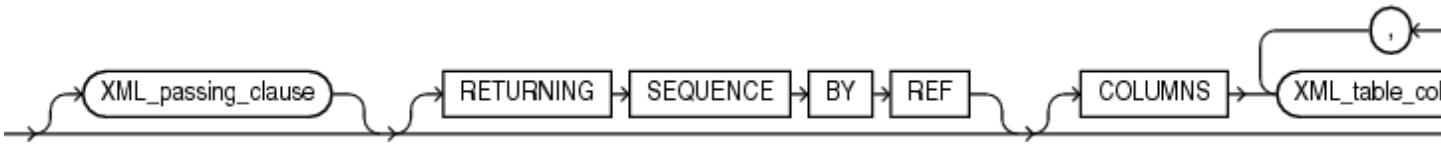
XMLnamespaces_clause::=



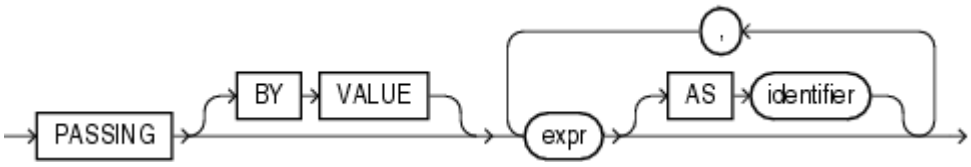
ノート:

`DEFAULT string` 句を 1 つのみ指定できます。

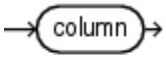
XMLTABLE_options::=

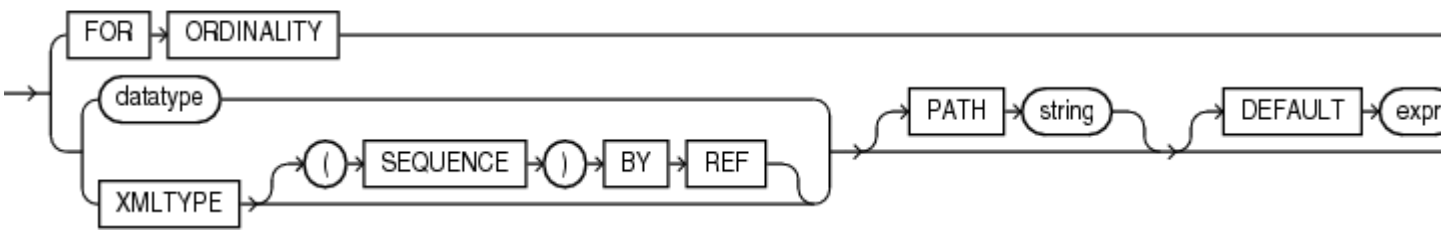


XML_passing_clause::=



XML_table_column::=





目的

`XMLTable`は、XQueryの評価結果をリレーショナル行および列にマップします。SQLを使用して、このファンクションで戻される結果を仮想リレーショナル表として問い合せることができます。

- XMLNAMESPACES句には、XMLネームスペースの宣言が含まれます。これらの宣言は、XQuery式(評価されたXQuery_string)と、XML_table_columnのPATH句のXPath式で参照されます。XQuery式は行を計算し、XPath式はXMLTableファンクション全体の列を計算します。COLUMNS句のPATH式に修飾名を使用する場合は、XMLNAMESPACES句を指定する必要があります。

- XQuery_stringはリテラル文字列です。この文字列は完全なXQuery式であり、プロローグ文字を含むことができます。XQuery_stringの値はXMLTableファンクションへの入力となります。つまり、このXQueryの結果は分解されてリレーショナル・データとして格納されます。

- XML_passing_clauseのexprは、XMLTypeまたはSQLスカラー・データ型のインスタンスを戻し、XQuery式を評価するためのコンテキストとして使用されます。PASSING句には、識別子を指定せずに1つのexprのみを指定できます。各exprの評価結果は、XQuery_stringの対応する識別子にバインドされます。exprの後にAS句が続かない場合、式の評価結果はXQuery_stringの評価用のコンテキスト項目として使用されます。この句がサポートするのは値渡しのみで、参照渡しはサポートしません。したがって、BY VALUEキーワードはオプションであり、意味を明確にするためのものです。

- オプションのRETURNING SEQUENCE BY REF句を指定すると、XQuery評価の結果が参照で戻されます。したがって、XML_table_column句の中でソース・データの任意の部分を参照できます。

  この句を指定しないと、XQuery評価の結果は値で戻されます。つまり、実際のノードへの参照のかわりに、ターゲット・ノードのコピーが戻されます。この場合、XML_table_column句では戻されたコピーの中に含まれないデータを参照できません。特に、ソース・データ内でターゲット・ノードより前にあるデータは参照できません。

- オプションのCOLUMNS句は、XMLTableにより作成される仮想表の列を定義します。

  - COLUMNS句を指定しない場合、XMLTableは、COLUMN_VALUEという名前の単一のXMLType疑似列を戻します。

  - FOR ORDINALITYは、columnが生成された行番号の列になるように指定します。FOR ORDINALITY句は、最大で1つにする必要があります。これは、NUMBER列として作成されます。

  - FOR ORDINALITY列以外の作成される各列について、列のデータ型を指定する必要があります。XMLTypeまたは他の任意のデータ型を指定できます。

    列のデータ型がXMLTypeの場合は、XMLTYPE句を指定します。オプションの(SEQUENCE) BY REF句を指定すると、PATH式のターゲットとなっているソース・データへの参照が列の内容として戻されます。それ以外の場合、columnにはそのターゲット・データのコピーが含まれます。

    XMLTypeデータを参照で戻すことにより、列のPATH式のターゲットの外にあるソース・データ内のノードをターゲットとするパスを持つ他の列を指定できるようになります。

    列のデータ型がそれ以外のデータ型の場合は、datatypeを指定します。

  - オプションのPATH句では、XQuery式の文字列でアドレス指定されるXQuery結果の部分を列の内容として使用するように指定します。

    PATHを指定しない場合、XQuery式のcolumnとみなされます。たとえば:

    ```
    XMLTable(... COLUMNS xyz)
    ```

    これは、次の式と同じです。

    ```
    XMLTable(... COLUMNS xyz PATH 'XYZ')
    ```

異なるPATH句を使用すると、XQuery結果を異なる仮想表の列に分割できます。

- オプションのDEFAULT句では、PATH式の結果が空の順序の場合に使用する値を指定します。exprは、デフォルト値の生成用に評価するXQuery式です。

関連項目:

- 追加の例を含むXMLTableファンクションおよびXQueryの概要については、『Oracle XML DB開発者ガイド』を参照してください。
- XMLTABLEによって生成される表の文字データ型の各列に割り当てる照合を定義する照合導出ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

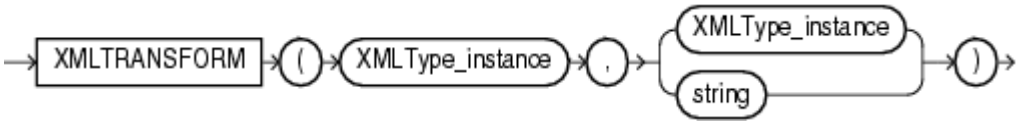
例

次の例では、warehouses表のwarehouse_spec列の各値にXQuery '/Warehouse'を適用した結果を、列WaterおよびRailのある仮想リレーショナル表に変換します。

```
SELECT warehouse_name warehouse,
   warehouse2."Water", warehouse2."Rail"
   FROM warehouses,
   XMLTABLE('/Warehouse'
      PASSING warehouses.warehouse_spec
      COLUMNS
         "Water" varchar2(6) PATH 'WaterAccess',
         "Rail" varchar2(6) PATH 'RailAccess')
      warehouse2;
WAREHOUSE                          Water  Rail
---------------------------------- ------ ------
Southlake, Texas                   Y      N
San Francisco                      Y      N
New Jersey                         N      N
Seattle, Washington                N      Y
```

XMLTRANSFORM

構文



目的

XMLTransformは、引数としてXMLTypeインスタンスおよびXSLスタイルシート(それ自体がXMLTypeインスタンス)を取ります。ファンクションは、スタイルシートをインスタンスに適用し、XMLTypeを戻します。

このファンクションは、データをデータベースから取得するように、スタイルシートに従ってデータを編成する場合に有効です。

関連項目:

このファンクションの詳細は、『Oracle XML DB開発者ガイド』を参照してください。

例

XMLTRANSFORMファンクションを使用するには、XSLスタイルシートが必要です。次に、ノード内の要素をアルファベット順に並べる単純なスタイルシートの例を示します。

```
CREATE TABLE xsl_tab (col1 XMLTYPE);
INSERT INTO xsl_tab VALUES (
   XMLTYPE.createxml(
   '<?xml version="1.0"?>
    <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
      <xsl:output encoding="utf-8"/>
      <!-- alphabetizes an xml tree -->
      <xsl:template match="*">
        <xsl:copy>
          <xsl:apply-templates select="*|text()">
            <xsl:sort select="name(.)" data-type="text" order="ascending"/>
          </xsl:apply-templates>
        </xsl:copy>
      </xsl:template>
      <xsl:template match="text()">
        <xsl:value-of select="normalize-space(.)"/>
      </xsl:template>
    </xsl:stylesheet>  '));
1 row created.
```

次の例では、XSLスタイルシートxsl_tabを使用して、サンプル表oe.warehousesにあるwarehouse_specの要素をアルファベット順に並べます。

```
SELECT XMLTRANSFORM(w.warehouse_spec, x.col1).GetClobVal()
   FROM warehouses w, xsl_tab x
   WHERE w.warehouse_name = 'San Francisco';
XMLTRANSFORM(W.WAREHOUSE_SPEC,X.COL1).GETCLOBVAL()
--------------------------------------------------------------------------------
<Warehouse>
  <Area>50000</Area>
  <Building>Rented</Building>
  <DockType>Side load</DockType>
  <Docks>1</Docks>
  <Parking>Lot</Parking>
  <RailAccess>N</RailAccess>
```

```
  <VClearance>12 ft</VClearance>
  <WaterAccess>Y</WaterAccess>
</Warehouse>
```

ROUNDおよびTRUNC日付ファンクション

表7-13は、ROUNDおよびTRUNC日付ファンクションで使用できる書式モデルと、日付の丸めや切捨ての対象となる単位をまとめたものです。デフォルトのモデルDDでは、午前0時(真夜中)を基準に丸めおよび切捨てを行い、日付を戻します。

表7-13 ROUNDおよびTRUNC日付ファンクションの日付書式モデル

書式モデル	丸め/切捨て対象の単位
CC SCC	4 桁の年の最初の 2 桁より 1 大きい数
SYYYY YYYY YEAR SYEAR YYY YY Y	年(7 月 1 日で切上げ)
IYYY IY IY I	ISO 8601 標準で定義されている暦週を含む年
Q	四半期(四半期の 2 番目の月の 16 日で切上げ)
MONTH MON MM RM	月(16 日で切上げ)
WW	年の最初の日と同じ曜日
IW	ISO 8601 標準で定義されている歴週の最初の日と同じ日、つまり月曜日
W	月の最初の日と同じ曜日
DDD DD J	曜日
DAY DY D	週の開始日
HH HH12 HH24	時間
MI	分

書式モデルDAY、DYおよびDで使用される週の開始日は、初期化パラメータNLS_TERRITORYで暗黙的に指定されます。

関連項目:

このパラメータの詳細は、『Oracle Databaseリファレンス』および『Oracle Databaseグローバリゼーション・サポート・ガイド』

を参照してください。

ユーザー定義ファンクション

PL/SQL、JavaまたはCでユーザー定義ファンクションを作成し、SQLまたはSQL組込みファンクションにはない機能を持たせることができます。ユーザー定義ファンクションは、式を指定できる場所であればどこでも、SQL文に指定できます。

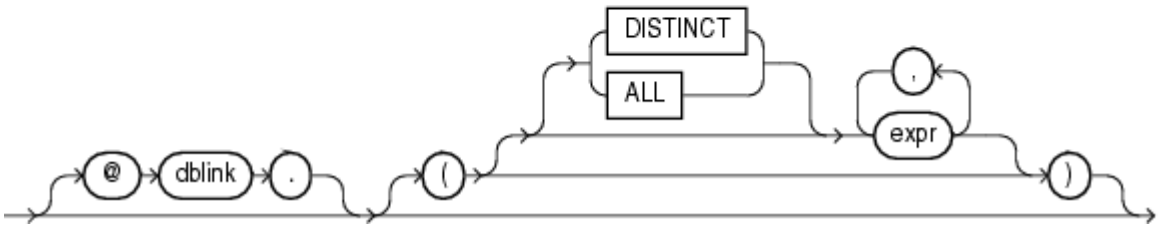
たとえば、次の場所でユーザー定義ファンクションを使用できます。

5. SELECT文のselectリスト
6. WHERE句の条件
7. CONNECT BY句、START WITH句、ORDER BY句およびGROUP BY句
8. INSERT文のVALUES句
9. UPDATE文のSET句

ノート:

Oracle SQL は、ブール値のパラメータまたは戻り値を持つファンクションのコールをサポートしていません。したがって、ユーザー定義ファンクションが SQL 文からコールされる場合は、数値(0 または 1)または文字列('TRUE'または'FALSE')を戻すように設計する必要があります。

user_defined_function::=



オプションの式のリストは、ファンクション、パッケージまたは演算子の属性と一致する必要があります。

ユーザー定義ファンクションの制限事項

DISTINCTおよびALLキーワードは、ユーザー定義の集計ファンクションでのみ有効です。

関連項目:

- ファンクションの作成(ユーザー定義ファンクションの制限を含む)の詳細は、「CREATE FUNCTION」を参照してください。
- ユーザー・ファンクションの作成および使用方法の詳細は、『Oracle Database開発ガイド』を参照してください。

前提条件

ユーザー定義ファンクションをSQL文で使用するには、トップレベル・ファンクションとして作成するか、またはパッケージ仕様部で宣言する必要があります。

SQLの式の中でユーザー・ファンクションを使用するには、ユーザーがユーザー・ファンクションのEXECUTE権限を持っている必要があります。ユーザー・ファンクションで定義したビューを問い合せるには、そのビューに対するREADまたはSELECT権限が必要です。ビューを検索するには個々のEXECUTE権限は必要ありません。

関連項目:

トップレベル・ファンクションの詳細は、「CREATE FUNCTION」を参照してください。パッケージ・ファンクションの詳細は、「CREATE PACKAGE」を参照してください。

名前の優先順位

SQL文内では、データベースの列名は、パラメータなしのファンクション名より優先順位が高くなります。たとえば、Human Resourcesのマネージャが、hrスキーマに次の2つのオブジェクトを作成する場合は、次のようにします。

```
CREATE TABLE new_emps (new_sal NUMBER, ...);
CREATE FUNCTION new_sal RETURN NUMBER IS BEGIN ... END;
```

その後、次の2つの文のように、new_salを参照するとnew_emps.new_sal列を参照することになります。

```
SELECT new_sal FROM new_emps;
SELECT new_emps.new_sal FROM new_emps;
```

new_salファンクションにアクセスするには、次のように入力します。

```
SELECT hr.new_sal FROM new_emps;
```

SQLの式内で使用できるユーザー・ファンクションのコール例を次に示します。

```
circle_area (radius)
payroll.tax_rate (empno)
hr.employees.tax_rate (dependent, empno)@remote
```

例

hrスキーマからtax_rateユーザー・ファンクションをコールし、tax_table内のss_no列とsal列に対してこのファンクションを実行するには、次のように指定します。

```
SELECT hr.tax_rate (ss_no, sal)
    INTO income_tax
    FROM tax_table WHERE ss_no = tax_id;
```

INTO句は、結果をincome_tax変数に配置するために使用できるPL/SQLです。

ネーミング規則

オプションのスキーマ名またはパッケージ名を1つのみ指定すると、最初の識別子はスキーマ名またはパッケージ名のいずれかになります。たとえば、PAYROLL.TAX_RATEという参照内のPAYROLLがスキーマ名かパッケージ名かを判断するには、Oracle Databaseは次の手順を実行します。

- カレント・スキーマ内のPAYROLLパッケージをチェックします。
- PAYROLLパッケージが検出されない場合は、トップレベルのTAX_RATEファンクションを含むスキーマ名PAYROLLを検索します。このようなファンクションが検出されない場合は、エラーを戻します。
- カレント・スキーマ内でPAYROLLパッケージが検出されると、PAYROLLパッケージの中でTAX_RATEファンクションを検索します。このようなファンクションが検出されない場合は、エラーを戻します。

また、ユーザーが定義したシノニムを使用して、ストアド・トップレベル・ファンクションを参照することもできます。

8 共通のSQL DDL句

この章では、複数のSQL文で使用されるSQLデータ定義句について説明します。

この章では、次の内容を説明します。

- allocate_extent_clause
- constraint
- deallocate_unused_clause
- file_specification
- logging_clause
- parallel_clause
- physical_attributes_clause
- size_clause
- storage_clause

allocate_extent_clause

目的

allocate_extent_clause句を使用すると、データベース・オブジェクトの新しいエクステントを明示的に割り当てることができます。

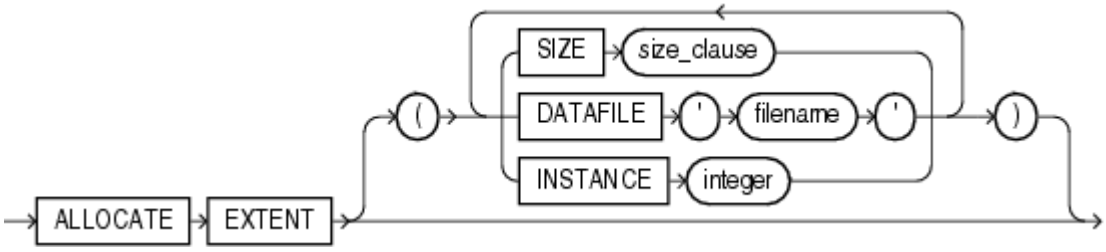
この句を使用してエクステントを明示的に割り当てた場合、NEXTおよびPCTINCREASE記憶域パラメータの値は変更されません。したがって、Oracle Databaseで暗黙的に割り当てられる次のエクステントのサイズには影響ありません。NEXTおよびPCTINCREASE記憶域パラメータの詳細は、「storage_clause」を参照してください。

エクステントは次のSQL文で割り当てることができます。

- ALTER CLUSTER(「ALTER CLUSTER」を参照)
- ALTER INDEX: エクステントを索引、索引パーティションまたは索引サブパーティションに割り当てる場合(「ALTER INDEX」を参照)
- ALTER MATERIALIZED VIEW: エクステントをマテリアライズド・ビュー、そのパーティションまたはサブパーティションの1つ、あるいは索引構成マテリアライズド・ビューのオーバーフロー・セグメントに割り当てる場合(「ALTER MATERIALIZED VIEW」を参照)
- ALTER MATERIALIZED VIEW LOG(「ALTER MATERIALIZED VIEW LOG」を参照)
- ALTER TABLE: エクステントを表、表パーティション、表サブパーティション、索引構成表のマッピング表、索引構成表のオーバーフロー・セグメントまたはLOB記憶域セグメントに割り当てる場合(「ALTER TABLE」を参照)

構文

allocate_extent_clause::=



(size_clause::=)

セマンティクス

この項では、allocate_extent_clauseのパラメータについて説明します。詳細は、特定のデータベース・オブジェクトに対してこれらのパラメータを設定または再設定するSQL文の説明を参照してください。

同じ文でallocate_extent_clauseとdeallocate_unused_clauseを指定することはできません。

SIZE

エクステント・サイズをバイト単位で指定します。integerの値は0から2147483647となります。これより大きいエクステント・サイズを指定するには、この範囲の整数とK、M、GまたはTを使用して、エクステント・サイズをKB、MB、GBまたはTB単位で指定します。

表、索引、マテリアライズド・ビューまたはマテリアライズド・ビュー・ログでは、SIZEを指定しないと、Oracle Databaseではオブジェクトの記憶域パラメータの値に基づいてサイズが決定されます。ただし、クラスタの場合、クラスタの記憶域パラメータは評価

されないため、デフォルト値を使用しない場合はSIZEを指定する必要があります。

DATAFILE 'filename'

新しいエクステントを割り当てるデータファイルを、表、クラスタ、索引、マテリアライズド・ビューまたはマテリアライズド・ビュー・ログの表領域から1つ指定します。DATAFILEを指定しないと、Oracleがデータファイルを選択します。

INSTANCE integer

Oracle Real Application Clustersを使用する場合のみ、このパラメータを使用します。

INSTANCE integerを指定すると、指定したインスタンスに対応付けられた空きリスト・グループが新しいエクステントを使用できるようになります。インスタンス数が空きリスト・グループの最大数を超えた場合、指定したインスタンス数/最大数という除算が行われ、その余りから、使用する空きリスト・グループが識別されます。インスタンスは初期化パラメータINSTANCE_NUMBERの値で識別されます。

このパラメータを指定しない場合、表、クラスタ、索引、マテリアライズド・ビューまたはマテリアライズド・ビュー・ログに領域が割り当てられますが、特定の空きリスト・グループからは割り当てられません。かわりにマスター空きリストが使用され、必要に応じて領域が割り当てられます。

ノート:

自動セグメント領域管理の使用時は、allocate_extent_clause の INSTANCE パラメータを設定しても、指定したインスタンスに新しく割り当てられた領域が確保されない場合があります。これは、自動セグメント領域管理では、エクステントとインスタンス間の固定したアフィニティが保持されないためです。

constraint

目的

constraintを使用すると、整合性制約(データベース内の値を制限する規則)を定義できます。Oracle Databaseでは、6つの制約を作成し、それを2つの方法で宣言することができます。

次に、6つの整合性制約について簡単に説明します。詳細は、「セマンティクス」を参照してください。

- NOT NULL 制約は、データベースの値がNULLになることを禁止します。
- 一意制約は、複数の行が同じ列または列の組合せで同じ値を持つことを禁止しますが、一部の値がNULLになることを許可します。
- 主キー制約は、1つの宣言でNOT NULL制約と一意制約を組み合せたものです。同じ列または列の組合せで、複数の行が同じ値を持つことを禁止し、値がNULLであることを禁止します。
- 外部キー制約は、ある表の値が別の表の値と一致することを必要とします。
- CHECK制約は、データベースの値が、指定された条件を満たすことを必要とします。
- REF列は、定義上は別のオブジェクト型またはリレーショナル表の中のオブジェクトを参照します。REF制約を使用すると、REF列と参照先のオブジェクトの関係をさらに詳しく指定できます。

制約は、次の2つの構文で定義できます。

- 個々の列または属性の定義の一部として定義できます。これを、表内指定といいます。
- 表定義の一部として定義できます。これを、表外指定といいます。

NOT NULL制約は、表内に宣言する必要があります。その他のすべての制約は、表内または表外に宣言できます。

制約句は、次の文に指定できます。

- CREATE TABLE(「CREATE TABLE」を参照)
- ALTER TABLE(「ALTER TABLE」を参照)
- CREATE VIEW(「CREATE VIEW」を参照)
- ALTER VIEW(「ALTER VIEW」を参照)

ビュー制約

Oracle Databaseでは、ビュー制約を適用していません。ただし、実表に対する制約によってビューに制約を適用できます。

ビューには、一意制約、主キー制約および外部キー制約のみを指定でき、これらの制約はDISABLE NOVALIDATEモードのみでサポートされています。オブジェクト列の属性にビュー制約を定義することはできません。

関連項目:

ビュー制約の詳細は、「ビュー制約」を参照してください。DISABLE NOVALIDATEモードの詳細は、「DISABLE句」を参照してください。

外部表の制約

外部表には、NOT NULL、一意制約、主キー制約および外部キー制約のみ指定できます。一意制約、主キー制約および外

部キーの制約は、RELY DISABLEモードでのみサポートされます。

関連項目:

RELYおよびDISABLEの詳細は、「DISABLE句」を参照してください。

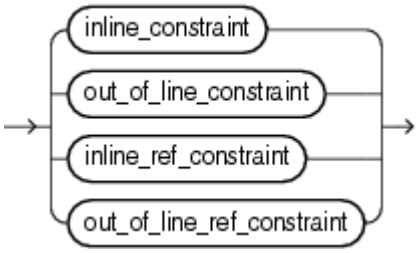
前提条件

制約を定義する文を発行できる権限が必要です。

外部キー制約を作成する場合は、この条件に加えて、親表またはビューが自分のスキーマ内に設定されている必要があります。設定されていないかぎり、親表またはビューの参照キー列に対するREFERENCES権限が必要です。

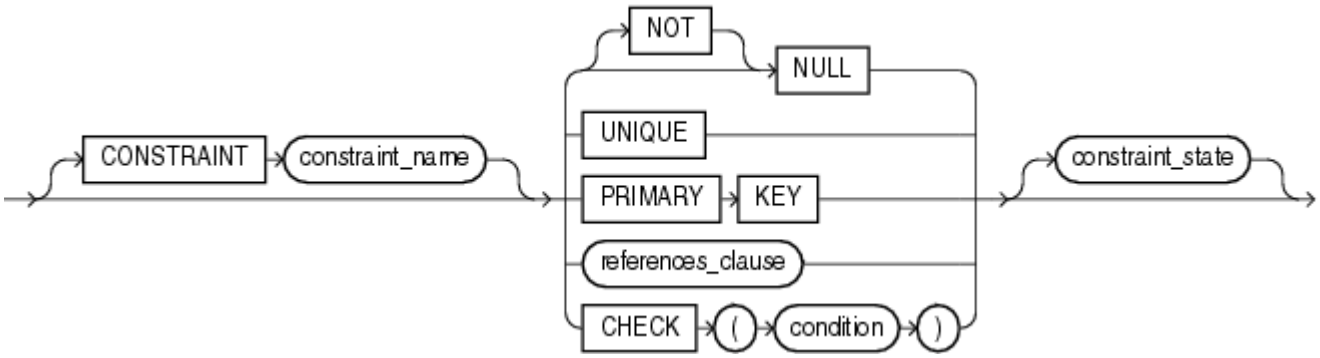
構文

constraint::=



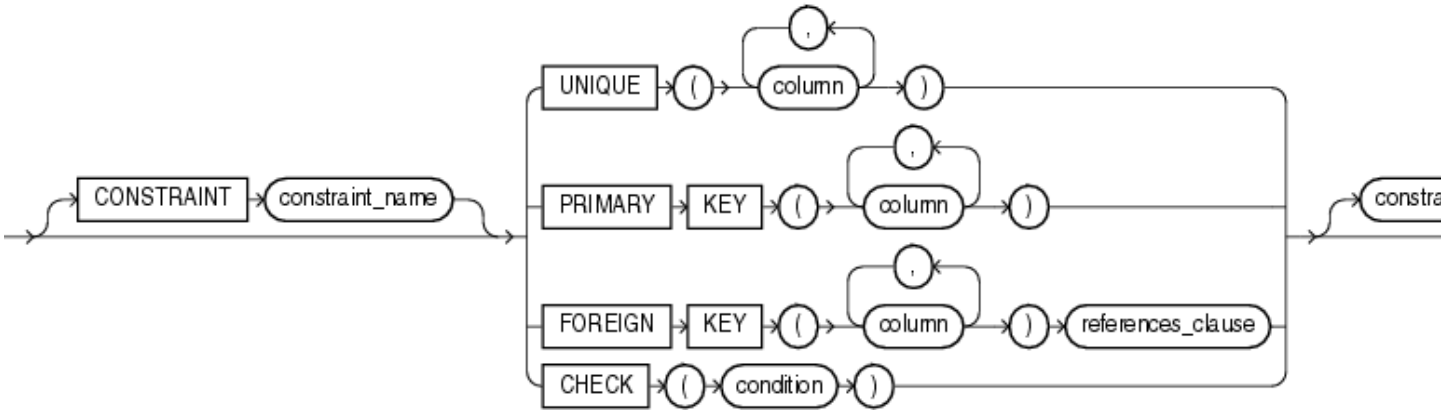
(inline_constraint::=、out_of_line_constraint::=、inline_ref_constraint::=、out_of_line_ref_constraint::=)

inline_constraint::=



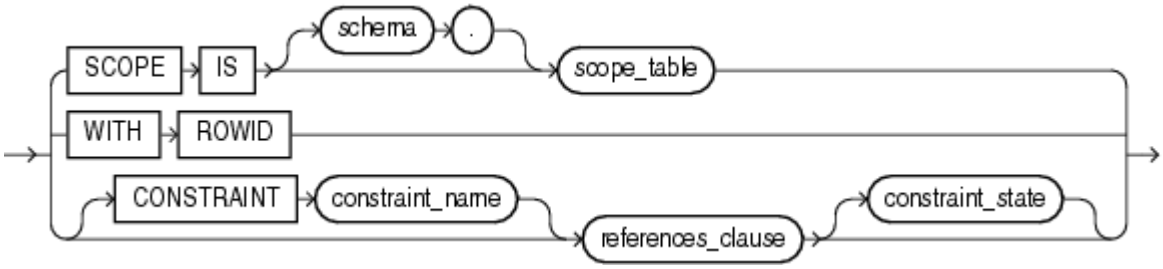
(references_clause::=)

out_of_line_constraint::=



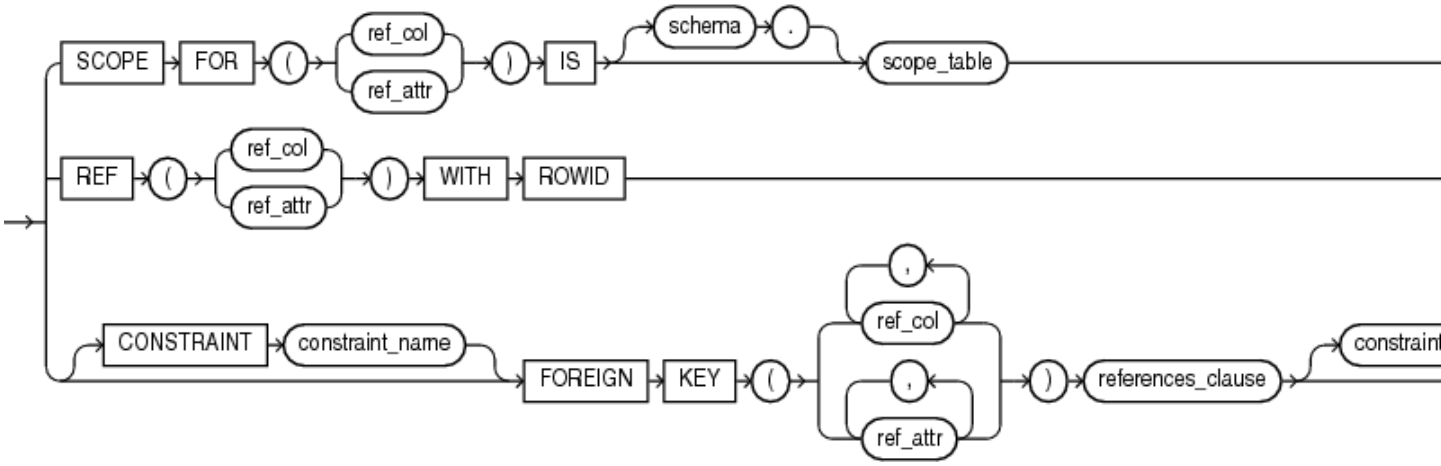
(references_clause::=、constraint_state::=)

inline_ref_constraint::=



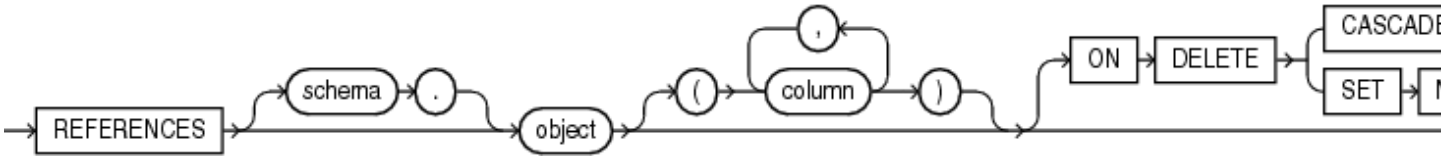
(references_clause::=、constraint_state::=)

out_of_line_ref_constraint::=

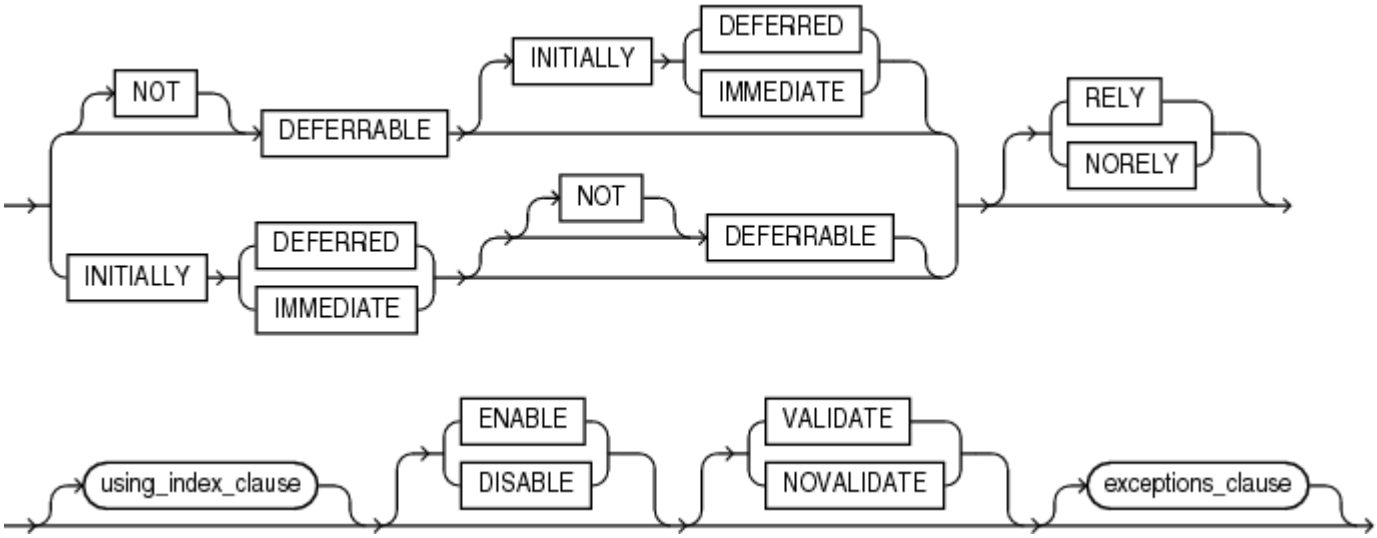


(references_clause::=、constraint_state::=)

references_clause::=

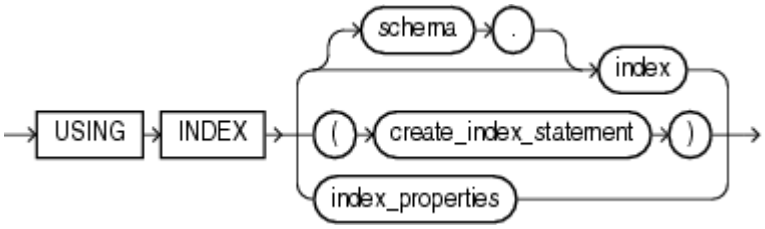


constraint_state::=



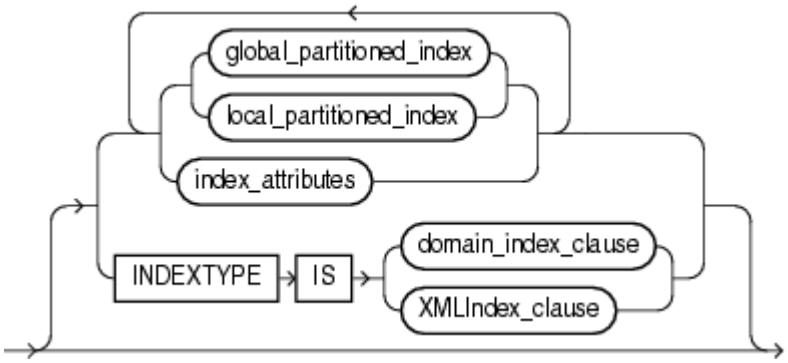
(using_index_clause::=、exceptions_clause::=)

using_index_clause::=



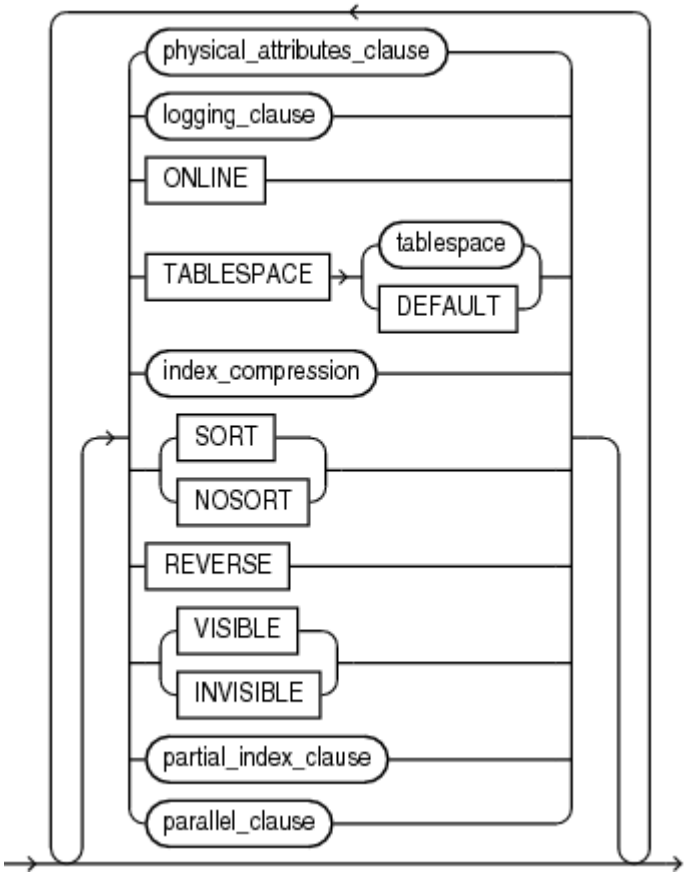
(create_index::=、index_properties::=)

index_properties::=



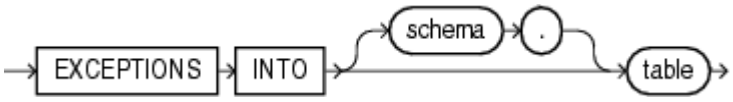
(global_partitioned_index::=、local_partitioned_index::=(CREATE INDEXの一部)、index_attributes::=::=。INDEXTYPE IS ...句は、制約を定義すると無効になります。)

index_attributes::=



(physical_attributes_clause::=、logging_clause::=、index_compression::=、partial_index_clause::=(すべてCREATE INDEXの一部)、parallel_clauseはusing_index_clauseではサポートされません。)

exceptions_clause::=



セマンティクス

この項では、constraintのセマンティクスについて説明します。詳細は、表またはビューの制約を定義または再定義するSQL文の説明を参照してください。

Oracle Databaseでは、ユーザー定義オブジェクト、ネストした表、VARRAY、REFまたはLOB型の列または属性に対して制約を使用することはできません。ただし、次の2つの例外があります。

- NOT NULL制約は、ユーザー定義オブジェクト、VARRAY、REFまたはLOB型の列や属性に使用できます。
- NOT NULL、外部キーおよびREF制約は、REF型の列に使用できます。

CONSTRAINT constraint_name

制約名を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。この識別子を指定しない場合、SYS_Cnの形式で名前が生成されます。整合性制約の名前と定義は、USER_、ALL_およびDBA_CONSTRAINTSデータ・ディクショナリ・ビュー(それぞれCONSTRAINT_NAME列およびSEARCH_CONDITION列)に格納されます。

関連項目:

データ・ディクショナリ・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

NOT NULL制約

NOT NULL制約は、列にNULLが含まれることを禁止します。NULLキーワード自体は、実際に整合性制約を定義するものではありませんが、これを指定すると、列にNULLが含まれることが許可されます。NOT NULLおよびNULLは、表内指定で定義する必要があります。NOT NULLまたはNULLを指定しない場合、NULLがデフォルトになります。

NOT NULL制約は、XMLType列およびVARRAY列で表内指定できる唯一の制約です。

NOT NULL制約を満たすには、表のすべての行がその列の値を持つ必要があります。

ノート:

Oracle Database では、すべてのキー列が NULL の表の行には索引を付けません(ただし、ビットマップ索引の場合を除きます)。このため、表のすべての行に索引を付けるには、1 つ以上の索引キー列に NOT NULL 制約を指定するか、ビットマップ索引を作成する必要があります。

NOT NULL制約の制限事項

NOT NULL制約には、次の制限事項があります。

- ビュー制約にはNULLまたはNOT NULLを指定できません。
- オブジェクト属性にはNULLまたはNOT NULLを指定できません。そのかわりに、IS [NOT] NULL条件でCHECK制約を使用してください。

関連項目:

「属性レベル制約の例」および「NOT NULLの例」を参照してください。

一意制約

一意制約は、列を一意キーとして指定します。複合一意キーは、列の組合せを一意キーとして指定します。一意制約を表内に定義する場合に必要となるのは、UNIQUEキーワードのみです。一意制約を表外に定義する場合は、1つ以上の列も指定する必要があります。複合一意キーは、表外に定義する必要があります。

一意制約を満たすには、表の中の2つの行が一意キーに対して同じ値を持たないようにする必要があります。ただし、単一の列で構成される一意キーの場合は、複数のNULLを持つことができます。複合一意キーを満たすには、表またはビューの2つの行がキー列に対して同じ組合せの値を持たないようにする必要があります。すべてのキー列に対してNULLを持つ行は、自動的にその制約を満たすことになります。ただし、1つ以上のキー列に対してNULLを持ち、その他のキー列に対して同じ組合せの値を持つ2つの行は、制約に違反します。

一意制約は、そのキー列の宣言された照合に依存します。詳細は、「制約の照合依存」を参照してください。

Oracleでは、1つ以上の列に一意制約を指定すると、暗黙的に一意キーに索引が作成されます。問合せのパフォーマンスを目的に一意性を定義する場合は、かわりにCREATE UNIQUE INDEX文を使用して明示的に一意索引を作成することをお薦めします。また、CREATE UNIQUE INDEX文で、条件付きの一意制約を定義する一意のファンクション索引を作成することもできます。詳細は、「条件付き一意性を定義するためのファンクション索引の使用方法: 例」を参照してください。

使用可能な一意制約を拡張データ型の列に指定すると、Oracleがその使用可能な制約の一意性を適用するための索引を作成しようとしたときに、「最大キー長を超過しました」というエラーが通知されることがあります。この問題を回避する方法の詳細は、「拡張データ型の列に対する索引の作成」を参照してください。

一意制約の制限事項

一意制約には、次の制限事項があります。

- 一意キー列は、LOB、LONG、LONG RAW、VARRAY、NESTED TABLE、OBJECT、REF、TIMESTAMP WITH TIME ZONEまたはユーザー定義型を含むことができません。ただし、一意キーはTIMESTAMP WITH LOCAL TIME ZONEの列を含むことができます。
- 1つの複合一意キーは、33以上の列を持つことはできません。
- 同一の列または列の組合せを一意キーと主キーの両方には指定できません。
- 継承階層でサブビューを作成する場合は、一意キーを指定できません。一意キーは、トップレベル(ルート)のビューのみに指定できます。
- 外部表の一意制約を指定する場合は、RELYおよびDISABLE制約状態を指定する必要があります。詳細は、「外部表の制約」を参照してください。

関連項目:

「一意キーの例」および「複合一意キーの例」を参照してください。

主キー制約

主キー制約は、列を表またはビューの主キーとして指定します。複合主キーは、列の組合せを主キーとして指定します。主キー制約を表内に定義する場合に必要となるのは、PRIMARY KEYキーワードのみです。主キー制約を表外に定義する場合は、1つ以上の列も指定する必要があります。複合主キーは表外に定義する必要があります。

主キー制約を満たすには:

- 主キーの値が、表の中の複数行に存在することはできません。
- 主キーを構成する列に、NULLを持たせることはできません。

主キー制約を作成する場合

- Oracle Databaseでは、主キー制約を適用する前に既存の索引に一意の値の集合が含まれる場合、その索引が使用されます。既存の索引は、一意であると定義されている場合も、一意でないと定義されている場合もあります。DML操作を実行すると、この既存の索引を使用して主キー制約が適用されます。
- 既存の索引を使用できない場合は、一意索引が生成されます。

主キー制約を削除する場合

- 既存の索引を使用して主キーが作成された場合、索引は削除されません。
- システム生成の索引を使用して主キーが作成された場合、索引は削除されます。

拡張データ型の列を使用可能な主キーとして指定すると、Oracleがその使用可能な制約の一意性を適用するための索引を作成しようとしたときに「最大キー長を超過しました」というエラーが通知されることがあります。この問題を回避する方法の詳細は、「拡張データ型の列に対する索引の作成」を参照してください。

主キー制約は、そのキー列の宣言された照合に依存します。詳細は、「制約の照合依存」を参照してください。

主キー制約の制限事項

主キー制約には、次の制限事項があります。

- 表またはビューには、主キーを1つのみ指定できます。
- 主キー列は、LOB、LONG、LONG RAW、VARRAY、NESTED TABLE、BFILE、REF、TIMESTAMP WITH TIME ZONEまたはユーザー定義型を含むことができません。ただし、主キーはTIMESTAMP WITH LOCAL TIME ZONEの列を含むことができます。
- 主キーのサイズは、1データベース・ブロックを超えることはできません。
- 1つの複合主キーは、33以上の列を持つことはできません。
- 同一の列または列の組合せを一意キーと主キーの両方には指定できません。
- 継承階層でサブビューを作成する場合は、主キーを指定できません。主キーは、トップレベル(ルート)のビューのみに指定できます。
- 外部表の主キー制約を指定する場合は、RELYおよびDISABLE制約状態を指定する必要があります。詳細は、「外部表の制約」を参照してください。

関連項目:

「主キーの例」および「複合主キーの例」を参照してください。

外部キー制約

外部キー制約(参照整合性制約ともいう)は、列を外部キーとして指定し、その外部キーと指定した主キーまたは一意キー(参照キー)との関係を設定します。複合外部キーは、列の組合せを外部キーとして指定します。

外部キーを持つ表またはビューを子オブジェクトといい、参照キーを持つ表またはビューを親オブジェクトといいます。外部キーと参照キーを、同一の表またはビューに設定することができます。この場合、親表と子表は同一の表になります。親表または親ビューのみを指定して、列名を指定しない場合、外部キーは自動的に親表または親ビューの主キーを参照します。外部キーと参照キーの対応する列は、同じ順序、データ型および宣言された照合で構成されている必要があります。

外部キー制約は、参照先の主キー列または一意キー列の宣言された照合に依存します。詳細は、「制約の照合依存」を参照してください。

単一キー列の外部キー制約は、表内または表外に定義できます。複合外部キーと属性の外部キーは、表外に指定します。

複合外部キー制約を満たすには、複合外部キーが親表または親ビューの複合一意キーまたは複合主キーを参照するか、外部キーの1つ以上の列の値がNULLである必要があります。

外部キーと主キーの両方、または外部キーと一意キーの両方に、同一の列または列の組合せを指定できます。また、外部キーとクラスタ・キーの両方にも同じ列または列の組合せを指定できます。

1つの表またはビューで複数の外部キーを定義できます。また、1つの列が複数の外部キーを構成することもできます。

外部キー制約の制限事項

外部キー制約には、次の制限事項があります。

- 外部キー列は、LOB、LONG、LONG RAW、VARRAY、NESTED TABLE、BFILE、REF、TIMESTAMP WITH

TIME ZONEまたはユーザー定義型を含むことができません。ただし、主キーはTIMESTAMP WITH LOCAL TIME ZONEの列を含むことができます。

- 親表または親ビューの参照一意制約または参照主キー制約が事前に定義されている必要があります。
- 1つの複合外部キーは、33以上の列を持つことはできません。
- 子表と親表は、同一データベース上に設定されている必要があります。分散データベースのノード間の参照整合性制約を使用可能にする場合、データベース・トリガーを使用する必要があります。「CREATE TRIGGER」を参照してください。
- 子オブジェクトと親オブジェクトのどちらかがビューの場合、ビュー制約のすべての制限事項が制約に適用されます。「ビュー制約」を参照してください。
- AS subquery句を含むCREATE TABLE文には、外部キー制約を定義できません。そのかわりに、制約を指定せずに表を作成し、後でALTER TABLE文を使用してその制約を追加できます。
- 表に外部キーが含まれ、外部キーの親が索引構成表の場合は、別のセッションが親表のキー以外の列を更新中のとき、外部キーを含む行を更新するセッションがハングすることがあります。
- 外部表の外部キー制約を指定する場合は、RELYおよびDISABLE制約状態を指定する必要があります。詳細は、「外部表の制約」を参照してください。

関連項目:

- 制約の使用の詳細は、『Oracle Database開発ガイド』を参照してください。
- 「外部キー制約の例」および「複合外部キー制約の例」を参照してください。

references_clause

外部キー制約はreferences_clause構文を使用します。外部キー制約を表内に指定する場合に必要となるのは、references_clauseのみです。外部キー制約を表外に指定する場合は、FOREIGN KEYキーワードと1つ以上の列も指定する必要があります。

ON DELETE句

ON DELETE句を指定すると、参照主キーまたは参照一意キーの値を削除した場合に参照整合性がどのように自動処理されるかを指定できます。この句を指定しない場合、子表に依存する行を持つ親表の中の参照キーの値は削除できません。

- 依存する外部キーの値を削除する場合は、CASCADEを指定します。
- 依存する外部キーの値をNULLに変換する場合は、SET NULLを指定します。仮想列の値は直接更新できないため、仮想列に対してこの句を指定することはできません。仮想列が導出される値を更新する必要があります。

ON DELETEの制限事項

この句は、ビュー制約に対して指定できません。

関連項目:

「ON DELETEの例」

CHECK制約

CHECK制約によって、表の各行に必要な条件を指定できます。制約を満たすためには、表のそれぞれの行が、その条件に対してTRUEまたは不明(NULLのため)のいずれかである必要があります。特定の行に対するCHECK制約条件が評価される場合、条件にある列名に、その行の列値が適用されます。

CHECK制約の表内指定と表外指定の構文は同じです。ただし、表内指定では現在定義されている列(または、オブジェクト列の場合は列の属性)のみを参照でき、表外指定では複数の列または属性を参照できます。

Oracleでは、CHECK制約の条件が相互に排他的かどうかは検証しません。このため、1つの列に対して複数のCHECK制約を作成する場合は、制約の用途が矛盾しないように注意する必要があります。また、条件の評価について特別な順序を想定しないでください。

CHECK制約の条件がNLS_DATE_FORMATなどのNLSパラメータに依存する場合、条件は、セッション値ではなくパラメータのデータベース値を使用して評価されます。NLSパラメータのデータベース値は、データ・ディクショナリ・ビューNLS_DATABASE_PARAMETERSにあります。これらの値はDDL文CREATE DATABASEによってデータベースに関連付けられ、その後変わることはありません。

関連項目:

- その他の詳細および構文は、「条件」を参照してください。
- 「CHECK制約の例」および「属性レベル制約の例」を参照してください。

CHECK制約の制限事項

CHECK制約には、次の制限事項があります。

- ビューに対しては、CHECK制約を指定できません。ただし、WITH CHECK OPTION句を使用してビューを定義することは可能です。これは、ビューにCHECK制約を指定することと同じです。
- CHECK制約の条件では、その表の中のすべての列を参照できますが、他の表の列は参照できません。
- CHECK制約の条件は、次の構造を持つことができません。
    - 副問合せおよびスカラー副問合せ式
    - 決定的でないファンクションへのコール(CURRENT_DATE、CURRENT_TIMESTAMP、DBTIMEZONE、LOCALTIMESTAMP、SESSIONTIMEZONE、SYSDATE、SYSTIMESTAMP、UID、USERおよびUSERENV)
    - ユーザー定義ファンクションへのコール
    - REF列の参照解除(DEREFファンクションを使用する場合など)
    - ネストした表の列または属性
    - 疑似列CURRVAL、NEXTVAL、LEVELまたはROWNUM
    - 完全に指定されていない日付定数
    - 外部表にはCHECK制約を指定できません。

REF制約

REF制約を使用すると、REF型の列とそれが参照するオブジェクトとの関係を指定できます。

ref_constraint

REF制約は、ref_constraint構文を使用します。REF制約は表内または表外に定義します。表外指定の場合は、指定しているREF列または属性を指定する必要があります。

- ref_columnには、オブジェクトまたはリレーショナル表のREF列の名前を指定します。
- ref_attributeには、リレーショナル表のオブジェクト列内の埋込みREF属性を指定します。

表内指定と表外指定のどちらでも、有効範囲制約、ROWID制約または参照整合性制約をREF列に定義できます。

REF列の有効範囲表または参照表が、主キーに基づくオブジェクト識別子を持っている場合、そのREF列はユーザー定義REF列です。

関連項目:

- REFデータ型の詳細は、『Oracle Databaseオブジェクト・リレーショナル開発者ガイド』を参照してください。
- 「外部キー制約」および「REF制約の例」を参照してください。

REF列の有効範囲制約

表がREF列を持つ場合は、この列のそれぞれのREF値が別のオブジェクト表内の行を参照する場合があります。SCOPE句は、参照の有効範囲を1つの表scope_tableに制限します。REF列または属性の値はscope_table内のオブジェクトを指し、その場所にREF列と同じ型のオブジェクト・インスタンスが格納されます。

REF列内の参照の有効範囲を1つの表に制限する場合に、SCOPE句を指定します。この句を指定するには、scope_tableが自分のスキーマ内にあるか、またはscope_tableに対するREADまたはSELECT権限あるいはREAD ANY TABLEまたはSELECT ANY TABLEシステム権限が必要です。REF列ごとに有効範囲表を1つのみ指定できます。

有効範囲制約の制限事項

有効範囲制約には、次の制限事項があります。

- 表が空でない場合は、有効範囲制約を既存の列に追加できません。
- VARRAY列のREF要素に対しては有効範囲制約を指定できません。
- AS subqueryを指定し、この副問合せがユーザー定義REFデータ型を戻す場合、この句を指定する必要があります。
- 有効範囲制約を、後でREF列から削除することはできません。
- 外部表には有効範囲制約を指定できません。

REF列のROWID制約

WITH ROWIDを指定して、ref_columnまたはref_attributeのREFの値とともにROWIDを格納します。ROWIDとともにREF値を格納した場合、参照解除操作のパフォーマンスは向上しますが、使用する領域も多くなります。デフォルトのREF値の記憶域では、ROWIDは格納されません。

関連項目:

参照解除の例は、「DEREF」を参照してください。

ROWID制約の制限事項

ROWID制約には、次の制限事項があります。

- VARRAY列のREF要素に対してはROWID制約を定義できません。
- ROWID制約を、後でREF列から削除することはできません。
- REF列または属性の範囲が限定される場合、この句は無視され、ROWIDはREFとともには格納されません。
- 外部表にはROWID制約を指定できません。

REF列の参照整合性制約

`ref_constraint`構文の`references_clause`を使用すると、外部キー制約をREF列に定義できます。また、この句はREF列や属性の有効範囲を参照表に暗黙的に制限します。ただし、REF列以外の外部キー制約が親表の実際の列を参照することに対し、REF列の外部キー制約は親表の暗黙のオブジェクト識別子を参照します。

制約名を指定しない場合、制約に対するシステム名がSYS_Cnという形式で生成されます。

範囲が限定されている既存のREF列に参照整合性制約を追加する場合、参照表は、REF列の有効範囲表と同じである必要があります。後で参照整合性制約を削除する場合、REF列の範囲が参照表に限定されたままになります。

他の型の列に対する外部キー制約と同様に、表内で宣言する場合に必要となるのは`references_clause`のみです。表外で宣言する場合は、`FOREIGN KEY`キーワード、および1つ以上のREF列または属性も指定する必要があります。

関連項目:

オブジェクト識別子の詳細は、『Oracle Databaseオブジェクト・リレーショナル開発者ガイド』を参照してください。

REF列の外部キー制約の制限事項

REF列における外部キー制約には、次の追加の制限事項があります。

- 範囲が限定されていない既存のREF列に参照整合性制約を追加する場合、有効範囲制約が暗黙的に追加されます。したがって、有効範囲制約に適用されるすべての制限事項は、この場合にも適用されます。
- `references_clause`内のオブジェクト名の後には列を指定できません。

制約の照合依存

Oracle Database 12cリリース2 (12.2)以降、主キー制約、一意制約および外部キー制約はそのキー列の宣言された照合に依存します。新しい行または更新された行の主キーまたは一意キーの文字列値は、キー列の宣言された照合を使用して既存の行の値と比較されます。たとえば、キー列の宣言された照合が大/小文字を区別しない照合`BINARY_CI`である場合、新しいキー列の値が既存のキー値と大/小文字のみ異なるときは、新しい行または更新された行が拒否される可能性があります。照合`BINARY_CI`では、大/小文字のみ異なる文字値は同じ値として処理されます。

外部キーの文字列値は、親キー列の宣言された照合を使用して親の主キー列または一意キー列の値と比較されます。たとえば、キー列の宣言された照合が大/小文字を区別しない照合`BINARY_CI`である場合、大/小文字のみ異なる値が存在するときは、対応する外部キー値について同じ親キー値がなくても、新しい子の行または更新された子の行が受け入れられる可能性があります。

外部キー列の宣言された照合は、対応する親キー列の照合と同じである必要があります。

制約のコンポジット・キーの列には、異なる宣言された照合を使用できます。

制約のキー列の宣言された照合が疑似照合の場合、制約では、照合`BINARY`の対応する変形が使用されます。制約は静的であり、疑似照合が依存するセッションNLSパラメータに依存できないため、疑似照合を直接使用して制約の値を比較することはできません。そのため、次のようになります。

- 疑似照合USING_NLS_COMP、USING_NLS_SORTおよびUSING_NLS_SORT_CSでは、照合BINARYが使用されます。
- 疑似照合USING_NLS_COMP_CIでは、照合BINARY_CIが使用されます。
- 疑似照合USING_NLS_COMP_AIでは、照合BINARY_AIが使用されます。

主キー列または一意キー列で使用される有効な照合がBINARYではない場合、この列に対して非表示の仮想列が作成されます。仮想列の式によって、元のキー列の文字値の照合キーが計算されます。主キー制約または一意制約は、元の列ではなく仮想列に内部的に作成されます。仮想列は、*_TAB_COLSファミリのデータ・ディクショナリ・ビューで参照できます。これらの非表示の仮想列のそれぞれについて、*_TAB_COLSビューのCOLLATED_COLUMN_IDに、対応する元のキー列を示す内部順序番号が含まれます。非表示の仮想列は、表の1000列の制限までカウントされます。

関連項目:

- 大/小文字を区別しない制約の例
- 照合の詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

制約状態の指定

制約を定義するとき、いつどのように制約を適用するかを指定できます。

constraint_state

constraint_stateは、表内指定と表外指定の両方に使用できます。DEFERRABLE句およびINITIALLY句は任意の順序で指定できますが、それを除く残りのコンポーネント句は、示されている順序で、各句を1回ずつだけ指定する必要があります。

DEFERRABLE句

DEFERRABLEおよびNOT DEFERRABLEパラメータは、後続のトランザクションで、SET CONSTRAINT(S)文を使用して、トランザクションが終了するまで制約のチェックを遅延できるようにするかどうかを指定します。この句を指定しない場合、NOT DEFERRABLEがデフォルトになります。

- NOT DEFERRABLEを指定すると、後続のトランザクションで、SET CONSTRAINT[S]句を使用して、トランザクションがコミットされるまでこの制約のチェックを遅延させることができません。NOT DEFERRABLE制約のチェックは、トランザクションの終わりまで遅延させることはできません。

  新しいNOT DEFERRABLE制約を宣言する場合、CREATE TABLEまたはALTER TABLE文のコミット時にその制約が有効である必要があります。有効でない場合、これらの文は正常に実行されません。
- DEFERRABLEを指定すると、後続のトランザクションで、SET CONSTRAINT[S]句を使用して、COMMIT文が発行されるまでこの制約のチェックを遅延させることができます。制約のチェックに失敗した場合、エラーが戻され、トランザクションはコミットされません。この設定によって、制約に違反する可能性のある変更をデータベースに行っている場合に、すべての変更が完了するまで、制約を一時的に使用禁止にすることが可能になります。

ノート:

オプティマイザは遅延可能制約の索引を使用可能として考慮しません。

制約の遅延可能状態は変更できません。これらのパラメータのいずれを指定するか、どちらも指定せずにNOT DEFERRABLE

制約を暗黙的に有効にするかに関係なく、この句をALTER TABLE文に指定することはできません。制約を削除してから再作成する必要があります。

関連項目:

- トランザクションに対する制約のチェックの設定の詳細は、「SET CONSTRAINT[S]」を参照してください。
- 遅延制約の詳細は、『Oracle Database管理者ガイド』および『Oracle Database概要』を参照してください。
- 「DEFERRABLE制約の例」

[NOT] DEFERRABLEの制限事項

ビュー制約には、これらのパラメータのいずれも指定できません。

INITIALLY句

INITIALLY句は、DEFERRABLE句が指定されている制約に対するデフォルトのチェック動作を指定します。INITIALLY設定は、後続のトランザクションにSET CONSTRAINT(S)文を指定することで上書きできます。

- INITIALLY IMMEDIATEを指定すると、この制約は後続の各SQL文の終わりでチェックされます。INITIALLYを指定しない場合、INITIALLY IMMEDIATEがデフォルトになります。

  新しいINITIALLY IMMEDIATE制約を宣言する場合、CREATE TABLEまたはALTER TABLE文のコミット時にその制約が有効である必要があります。有効でない場合、これらの文は正常に実行されません。

- INITIALLY DEFERREDを指定すると、この制約は後続のトランザクションの終わりでチェックされます。

制約をNOT DEFERRABLEとして宣言した場合、この句は無効です。NOT DEFERRABLE制約は自動的にINITIALLY IMMEDIATEになり、INITIALLY DEFERREDに変更することはできないためです。

RELY句

RELYおよびNORELYパラメータは、クエリー・リライトでNOVALIDATEモードの制約を考慮するかどうかを指定します。RELYを指定すると、適用されていないクエリー・リライトに対するNOVALIDATEモードの制約は、アクティブになります。その制約はNOVALIDATEモードであるため、適用されません。デフォルト値はNORELYです。

適用されない制約は、通常、マテリアライズド・ビューおよびクエリー・リライトにのみ有効です。QUERY_REWRITE_INTEGRITYモードに従って、クエリー・リライトは、VALIDATEモードの制約、またはRELYパラメータが設定されたNOVALIDATEモードの制約を使用して、結合情報を確認します。

RELY句の制限事項

遅延不可能なNOT NULL制約をRELYに設定することはできません。

関連項目:

マテリアライズド・ビューおよびクエリー・リライトの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

索引による制約の適用

一意制約または主キー制約の状態を定義する場合は、制約の適用に使用する索引を指定すること、または制約の適用に使用する索引をOracleで自動作成することが可能です。

using_index_clause

using_index_clauseは、一意制約または主キー制約を有効にしている場合にのみ指定できます。
using_index_clauseの句はどのような順序でも指定できますが、各句を指定できるのは1回のみです。

- schema.indexを指定すると、指定した索引を使用して制約が適用されます。索引がない、または制約の適用に索引が使用できない場合、エラーが戻されます。
- create_index_statementを指定すると、索引が作成され、制約の適用に使用されます。索引が作成できない、または制約の適用に索引が使用できない場合、エラーが戻されます。
- 既存の索引の指定も、新しい索引の作成も行わない場合は、索引が自動で作成されます。この場合は、次のようになります。
    - 索引には制約と同じ名前が割り当てられます。
    - 表がパーティション化されている場合、一意制約または主キー制約にローカル・パーティション索引またはグローバル・パーティション索引を指定できます。

using_index_clauseの制限事項

using_index_clauseには、次の制限事項が適用されます。

- この句は、ビュー制約に対して指定できません。
- この句は、NOT NULL、外部キーまたはCHECK制約には指定できません。
- 索引構成表の主キーが使用可能な場合は、索引の指定(schema.index)および作成(create_index_statement)はできません。
- index_attributesのparallel_clauseは指定できません。
- index_propertiesのINDEXTYPE IS ...句は、制約の定義では無効です。

関連項目:

- index_attributes、global_partitioned_index、local_partitioned_index句の詳細、および索引に関連するNOSORTとlogging_clauseの詳細は、「CREATE INDEX」を参照してください。
- 「physical_attributes_clause」と「PCTFREEパラメータ」および「storage_clause」を参照してください。
- 明示的な索引の制御例

ENABLE句

表のデータに制約を適用するには、ENABLEを指定します。

一意制約または主キー制約を使用可能にした場合、キーに索引が存在しないと、一意索引が作成されます。KEEP INDEXを指定しないかぎり、その後で制約が使用禁止になった場合にこの索引は削除されます。そのため、制約が使用可能になるたびに索引が再作成されます。

索引の再作成を避け、余分な索引を削除するには、最初に使用禁止にした主キー制約および一意制約を新しく作成します。その後、一意でない索引を作成して(または、既存の一意でない索引を使用して)制約を適用してください。制約が使用禁止の場合、一意でない索引は削除されず、後続のENABLE操作が容易になります。

- ENABLE VALIDATEを使用すると、すべての旧データと新データが制約に従うことを指定できます。制約が使用可能で、妥当性チェック済の場合は、すべてのデータが現在有効で、今後も有効であることが保証されます。

  整合性制約に違反する行が表にある場合、制約は使用禁止のままエラーを戻します。すべての行が制約に従ってい

る場合、Oracleは制約を使用可能にします。新規データが整合性制約に違反する場合、その後の文は実行されず、整合性制約の違反を示すエラーが戻されます。

主キー制約をENABLE VALIDATEモードに設定すると、妥当性チェック・プロセスによって主キー列にNULLが含まれないことが検証されます。これによるオーバーヘッドを回避するには、データを列に入力する前およびこの表の主キー制約を使用可能にする前に、主キーの各列にNOT NULLのマークを付けます。

- ENABLE NOVALIDATEを使用すると、制約データに対して新しく行うすべてのDML操作が制約に従うことが保証されます。表の既存データが制約に従っていることは保証されません。

VALIDATEもNOVALIDATEも指定しない場合、VALIDATEがデフォルトになります。

ENABLE NOVALIDATEからENABLE VALIDATEに単一制約状態を変更すると、パラレルで操作が実行できるため、読込み、書込みまたはその他のDDL操作が中断されません。

ENABLE句の制限事項

無効の一意キーまたは主キーを参照している外部キーを有効にすることはできません。

DISABLE句

整合性制約を無効にするには、DISABLEを指定します。データ・ディクショナリでは、使用禁止になっている整合性制約は、使用可能な制約とともに表示されます。この句を指定せずに制約を作成した場合は、その制約は自動的に使用可能になります。

- DISABLE VALIDATEは、制約を使用禁止にして制約の索引を削除しますが、制約は有効のままです。この機能を使用すると、大量のデータをロードでき、また索引用の領域が削減されるため、データ・ウェアハウスで非常に有効です。この設定を使用することで、ALTER TABLE文のexchange_partition_subpart句またはSQL*Loaderを使用して、データを非パーティション表からパーティション表にロードすることもできます。他のSQL文を使用した表に対するすべての変更(挿入、更新および削除)は禁止されます。

  関連項目:

  この設定の使用方法の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

- DISABLE NOVALIDATEは、Oracleによって制約がメンテナンスされないこと(使用禁止になっているため)、および制約が真であると保証されないこと(妥当性チェックが行われていないため)を示します。

  外部キー制約がDISABLE NOVALIDATE状態であっても、外部キーが参照している主キーを持つ表を削除できません。また、オプティマイザは、DISABLE NOVALIDATE状態でも制約を使用できます。

  関連項目:

  この設定を使用する状況の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

VALIDATEもNOVALIDATEも指定しない場合、NOVALIDATEがデフォルトになります。

一意索引を使用している一意制約または主キー制約を使用禁止にすると、一意索引は削除されます。その他のノートや制限事項は、「CREATE TABLE」の「enable_disable_clause」を参照してください。

VALIDATE | NOVALIDATE

VALIDATEおよびNOVALIDATEの動作は、制約が明示的にまたはデフォルトで有効/無効のどちらになっているかで異なりま

す。VALIDATEおよびNOVALIDATEキーワードは、「ENABLE句」および「DISABLE句」で説明されています。

NOVALIDATEモードでの外部キー制約のノート

外部キー制約がNOVALIDATEモードの場合、表内の既存のデータが制約に準拠せず、QUERY_REWRITE_INTEGRITYパラメータがENFORCEDに設定されていないと、オプティマイザが表の問合せ時に結合の絞込みを使用することがあります。この場合、準拠しない外部キー値を持つ表の行をフィルタで除外する結合条件が問合せに含まれている場合でも、問合せはこのような行を返すことがあります。

制約の例外の処理

制約の状態を定義する場合は、制約に違反するすべての行のROWIDを格納する表を指定できます。

exceptions_clause

exceptions_clause構文を使用すると、例外の処理を定義できます。schemaを指定しない場合、自分のスキーマ内に例外表があるとみなされます。この句自体を指定しない場合、表の名前はEXCEPTIONSになります。EXCEPTIONS表または指定した表は、ローカル・データベースに存在する必要があります。

次のいずれかのスクリプトを使用して、EXCEPTIONS表を作成できます。

- UTLEXCPT.SQLは、物理ROWIDを使用します。そのため、行は、索引構成表からではなく従来表から収集されます。(次の注意を参照。)
- UTLEXPT1.SQLは、ユニバーサルROWIDを使用します。そのため、行は、従来表と索引構成表の両方から収集されます。

独自の例外表を作成する場合、これら2つのスクリプトのいずれかで規定される形式に従う必要があります。

ユニバーサルROWIDではなく、主キーに基づく索引構成表から例外を収集する場合、索引構成表ごとに別の例外表を作成し、主キー記憶域を確保する必要があります。スクリプトを変更および再発行することによって、別の名前の例外表を複数作成できます。

exceptions_clauseの制限事項

exceptions_clauseには、次の制限事項が適用されます。

- この句は、ビュー制約に対して指定できません。
- この文が正常に終了されるまでROWIDは存在しないため、この句をCREATE TABLE文に指定することはできません。

関連項目:

- SQLスクリプトの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』のDBMS_IOTパッケージを参照してください。
- 移行行および連鎖行の削除については、『Oracle Databaseパフォーマンス・チューニング・ガイド』を参照してください。

ビュー制約

Oracleでは、ビュー制約を適用していません。ただし、ビューに対する操作には、基となる実表に定義されている整合性制約が適用されます。つまり、実表に対する制約によって、ビューに制約を適用できます。

ビュー制約のノート

ビュー制約は表制約のサブセットであり、次の制限事項があります。

- ビューには、一意制約、主キー制約および外部キー制約のみ指定できます。ただし、WITH CHECK OPTION句を使用してビューを定義することは可能です。これは、ビューにCHECK制約を指定することと同じです。
- ビュー制約は、DISABLE NOVALIDATEモードのみでサポートされています。他のモードは指定できません。ビュー制約を宣言する場合は、キーワードDISABLEを指定する必要があります。NOVALIDATEはデフォルトのため、明示的に指定する必要はありません。
- RELYおよびNORELYパラメータはオプションです。ビュー制約は適用されないため、通常はRELYパラメータで指定し、より有効に使用します。RELYまたはNORELYキーワードは、DISABLEキーワードより前に指定する必要があります。
- ビュー制約は直接適用されないため、INITIALLY DEFERREDまたはDEFERRABLEは指定できません。
- using_index_clause、exceptions_clause句、またはreferences_clauseのON DELETE句は指定できません。
- オブジェクト列の属性にビュー制約を定義することはできません。

外部表の制約

Oracle Database 12cリリース2 (12.2)以降、外部表には、NOT NULL、一意制約、主キー制約および外部キー制約を指定できます。

外部表に対するNOT NULL制約が適用され、列にNULLが含まれないようにします。

一意制約、主キー制約および外部キーの制約は、RELY DISABLEモードでのみ外部表に対してサポートされます。これらの制約を作成する場合は、キーワードRELYおよびDISABLEを指定する必要があります。これらの制約は宣言的であり、適用されません。より多くのオプティマイザ変換を考慮できるため、問合せパフォーマンスが向上し、リソース使用量が削減される可能性があります。オプティマイザがこれらのRELY DISABLE制約を利用できるようにするには、QUERY_REWRITE_INTEGRITY初期化パラメータをtrustedまたはstale_toleratedに設定する必要があります。

例

一意キーの例

次の文は、サンプル表sh.promotionsを作成する文の例です。この文は制約を表内に定義し、promo_id列で一意キーを暗黙的に使用可能にします(この例では、他の制約は省略されています)。

```
CREATE TABLE promotions_var1
    ( promo_id          NUMBER(6)
                        CONSTRAINT promo_id_u  UNIQUE
    , promo_name        VARCHAR2(20)
    , promo_category    VARCHAR2(15)
    , promo_cost        NUMBER(10,2)
    , promo_begin_date  DATE
    , promo_end_date    DATE
    ) ;
```

制約promo_id_uは、一意キーとしてpromo_id列を識別します。この制約によって、表に同じIDを持つ複数の販売促進の行が存在しないことが保証されます。ただし、識別子のない行は許可されます。

この制約を表外に定義して使用可能にすることもできます。

```
CREATE TABLE promotions_var2
    ( promo_id          NUMBER(6)
    , promo_name        VARCHAR2(20)
    , promo_category    VARCHAR2(15)
    , promo_cost        NUMBER(10,2)
```

```
    , promo_begin_date DATE
    , promo_end_date   DATE
    , CONSTRAINT promo_id_u UNIQUE (promo_id)
   USING INDEX PCTFREE 20
      TABLESPACE stocks
      STORAGE (INITIAL 8M) );
```

この文にはusing_index_clauseも含まれています。この句は、制約を使用可能にするために作成される索引の記憶特性を指定します。

複合一意キーの例

次の文は、oe.warehouses表のwarehouse_id列とwarehouse_name列を組み合せて複合一意キーを定義し、有効にします。

```
ALTER TABLE warehouses
   ADD CONSTRAINT wh_unq UNIQUE (warehouse_id, warehouse_name)
   USING INDEX PCTFREE 5
   EXCEPTIONS INTO wrong_id;
```

wh_unq制約によって、warehouse_idとwarehouse_nameを組み合せた値が表の中に複数存在しないことが保証されます。

このADD CONSTRAINT句には、制約以外のプロパティも指定できます。

- USING INDEX句は、制約を使用可能にするために作成される索引の記憶特性を指定します。
- EXCEPTIONS INTO句によって、制約に違反する行がwarehouses表に含まれている場合、その行に関する情報がwrong_id表に書き込まれます。wrong_id例外表が存在しない場合、この文は正常に実行されません。

主キーの例

次の文は、サンプル表hr.locationsを作成する文の例です。locations_demo表を作成し、location_id列に主キーを定義して使用可能にします(この例では、hr.locations表の他の制約は省略されています)。

```
CREATE TABLE locations_demo
    ( location_id    NUMBER(4) CONSTRAINT loc_id_pk PRIMARY KEY
    , street_address VARCHAR2(40)
    , postal_code    VARCHAR2(12)
    , city           VARCHAR2(30)
    , state_province VARCHAR2(25)
    , country_id     CHAR(2)
    ) ;
```

表内に指定されているloc_id_pk制約は、location_id列をlocations_demo表の主キーとして識別します。この制約によって、表の中の複数の所在地が同一の所在地識別子を持つことはなく、かつ所在地識別子がNULLにならないことが保証されます。

この制約を表外に定義して使用可能にすることもできます。

```
CREATE TABLE locations_demo
    ( location_id    NUMBER(4)
    , street_address VARCHAR2(40)
    , postal_code    VARCHAR2(12)
    , city           VARCHAR2(30)
    , state_province VARCHAR2(25)
    , country_id     CHAR(2)
    , CONSTRAINT loc_id_pk PRIMARY KEY (location_id));
```

NOT NULLの例

次の文は、(「主キーの例」で作成した) locations_demo表を変更し、country_id列にNOT NULL制約を定義して有効にします。

```
ALTER TABLE locations_demo
   MODIFY (country_id CONSTRAINT country_nn NOT NULL);
```

制約country_nnによって、country_idがNULLの所在地の行が表にないことが保証されます。

複合主キーの例

次の文は、sh.salesサンプル表のprod_id列およびcust_id列を組み合せて複合主キーを定義します。

```
ALTER TABLE sales
    ADD CONSTRAINT sales_pk PRIMARY KEY (prod_id, cust_id) DISABLE;
```

この制約は、sales表の主キーとしてprod_id列とcust_id列の組合せを識別します。この制約によって、表の中の複数の行がprod_id列とcust_id列に同じ組合せの値を持たないことが保証されます。

この制約句(PRIMARY KEY)では、次の制約のプロパティも指定しています。

- 制約定義によって制約名が指定されていないため、この制約に対する名前がOracleによって自動的に生成されます。
- DISABLE句によって制約が定義されますが、使用可能にはなりません。

外部キー制約の例

次の文は、dept_20表を作成し、departments表のdepartment_id列の主キーを参照するdepartment_id列に、外部キーを定義して有効にします。

```
CREATE TABLE dept_20
   (employee_id     NUMBER(4),
    last_name       VARCHAR2(10),
    job_id          VARCHAR2(9),
    manager_id      NUMBER(4),
    hire_date       DATE,
    salary          NUMBER(7,2),
    commission_pct  NUMBER(7,2),
    department_id   CONSTRAINT fk_deptno
                    REFERENCES departments(department_id) );
```

fk_deptno制約によって、dept_20表の従業員が属しているすべての部門がdepartments表に含まれることが保証されます。ただし、部門番号がNULL値の従業員(どの部門にも属さない従業員)がいてもかまいません。すべての従業員がいずれかの部門に割り当てられるようにするには、dept_20表のdepartment_id列に対して、REFERENCES制約の他にNOT NULL制約を作成します。

この制約を定義して使用可能にする前に、departments表のdepartment_id列を主キーまたは一意キーとして指定する制約を定義して使用可能にする必要があります。

制約が表内に定義されているため、この外部キー制約定義はFOREIGN KEY句を使用しません。また、この列には参照キーのデータ型が自動的に割り当てられるため、department_id列のデータ型は必要ありません。

制約定義によって親表と参照キーの列の両方が指定されます。参照キーは親表の主キーであるため、参照キーの列名の指定は任意です。

この外部キー制約を表外に定義することもできます。

```
CREATE TABLE dept_20
   (employee_id     NUMBER(4),
    last_name       VARCHAR2(10),
```

```
    job_id          VARCHAR2(9),
    manager_id      NUMBER(4),
    hire_date       DATE,
    salary          NUMBER(7,2),
    commission_pct  NUMBER(7,2),
    department_id,
   CONSTRAINT fk_deptno
      FOREIGN  KEY (department_id)
      REFERENCES  departments(department_id) );
```

これらの外部キー定義は、両方ともON DELETE句を指定していないため、従業員がいる部門は削除できません。

ON DELETEの例

次の文は、dept_20表を作成し、2つの参照整合性制約を定義して有効にした後、ON DELETE句を使用します。

```
CREATE TABLE dept_20
   (employee_id     NUMBER(4) PRIMARY KEY,
    last_name       VARCHAR2(10),
    job_id          VARCHAR2(9),
    manager_id      NUMBER(4) CONSTRAINT fk_mgr
                    REFERENCES employees ON DELETE SET NULL,
    hire_date       DATE,
    salary          NUMBER(7,2),
    commission_pct  NUMBER(7,2),
    department_id   NUMBER(2)   CONSTRAINT fk_deptno
                    REFERENCES departments(department_id)
                    ON DELETE CASCADE );
```

最初のON DELETE句によって、従業員番号2332の上司がemployees表から削除された場合、2332の上司の部下だったdept_20表のすべての従業員のmanager_id値がNULLになります。

次のON DELETE句によって、departments表のdepartment_id値が削除されると、これに依存するdept_20表の行のdepartment_id値も同時に削除されます。たとえば、部門番号20がdepartments表から削除されると、部門番号20の全従業員がdept_20表から削除されます。

複合外部キー制約の例

次の文は、dept_20表のemployee_id列およびhire_date列を組み合せて外部キーを定義し、有効にします。

```
ALTER TABLE dept_20
   ADD CONSTRAINT fk_empid_hiredate
   FOREIGN KEY (employee_id, hire_date)
   REFERENCES hr.job_history(employee_id, start_date)
   EXCEPTIONS INTO wrong_emp;
```

fk_empid_hiredate制約によって、dept_20表の中すべての従業員が、employees表に存在するemployee_idとhire_dateの組合せを持つことが保証されます。この制約を定義して使用可能にする前に、employees表のemployee_id列とhire_date列の組合せを主キーまたは一意キーとして指定する制約を定義して使用可能にする必要があります。

EXCEPTIONS INTO句によって、制約に違反する行がdept_20表に含まれている場合、その行に関する情報がwrong_emp表に書き込まれます。wrong_emp例外表が存在しない場合、この文は正常に実行されません。

CHECK制約の例

次の文ではdivisions表を作成し、表の各列のcheck制約を指定します。

```
CREATE TABLE divisions
   (div_no    NUMBER  CONSTRAINT check_divno
              CHECK (div_no BETWEEN 10 AND 99)
              DISABLE,
```

```
    div_name  VARCHAR2(9)  CONSTRAINT check_divname
              CHECK (div_name = UPPER(div_name))
              DISABLE,
    office    VARCHAR2(10)  CONSTRAINT check_office
              CHECK (office IN ('DALLAS','BOSTON',
              'PARIS','TOKYO'))
              DISABLE);
```

列に定義されている各制約によって、列の値が次のように制限されます。

- check_divnoによって、部門番号は必ず10から99の範囲内になります。
- check_divnameによって、部門名はすべて大文字になります。
- check_officeによって、事務所の所在地はDallas、Boston、ParisまたはTokyoのいずれかに制限されます。

それぞれのCONSTRAINT句にDISABLE句が指定されているため、これらの制約は定義されるのみで、使用可能にはされません。

次の文は、dept_20表を作成し、CHECK制約を表外に定義して暗黙的に使用可能にします。

```
CREATE TABLE dept_20
   (employee_id     NUMBER(4) PRIMARY KEY,
    last_name       VARCHAR2(10),
    job_id          VARCHAR2(9),
    manager_id      NUMBER(4),
    salary          NUMBER(7,2),
    commission_pct  NUMBER(7,2),
    department_id   NUMBER(2),
    CONSTRAINT check_sal CHECK (salary * commission_pct <= 5000));
```

この制約は、不等式の条件を使用して、従業員の歩合総額(salaryとcommission_pctを掛けた金額)を$5000に制限します。

- 従業員の給与と歩合がNULL以外の値の場合、制約を満たすには、この金額が$5000を超えてはいけません。
- 従業員の給与または歩合がNULL値の場合、条件の結果は不明となり、その従業員は自動的に制約を満たします。

この例の制約句には制約名が指定されていないため、制約の名前が自動的に生成されます。

次の文は、1つの主キー制約、2つの外部キー制約、1つのNOT NULL制約および2つのCHECK制約を定義して使用可能にします。

```
CREATE TABLE order_detail
  (CONSTRAINT pk_od PRIMARY KEY (order_id, part_no),
   order_id    NUMBER
      CONSTRAINT fk_oid
         REFERENCES oe.orders(order_id),
   part_no     NUMBER
      CONSTRAINT fk_pno
         REFERENCES oe.product_information(product_id),
   quantity    NUMBER
      CONSTRAINT nn_qty NOT NULL
      CONSTRAINT check_qty CHECK (quantity > 0),
   cost        NUMBER
      CONSTRAINT check_cost CHECK (cost > 0) );
```

この制約によって、表のデータに対して次の規則を適用できます。

- pk_odは、order_id列とpart_no列の組合せを表の主キーとして指定します。この制約を満たすためには、order_id列およびpart_no列の組合せが同じである行が、表内に2つあってはいけません。また、order_id列

およびpart_no列では、行にNULLを入れることはできません。

- fk_oidは、サンプル・スキーマoe内のorders表にあるorder_id列を参照する外部キーとして、order_id列を指定します。order_detail.order_id列に追加されるすべての新しい値は、oe.orders.order_id列にあらかじめ存在する必要があります。
- fk_pnoは、oeが所有するproduct_information表にあるproduct_id列を参照する外部キーとして、product_id列を指定します。order_detail.product_id列に追加されるすべての新しい値は、oe.product_information.product_id列にあらかじめ存在する必要があります。
- nn_qtyは、quantity列に対してNULLを禁止します。
- check_qtyによって、quantity列の値は必ず0(ゼロ)より大きくなります。
- check_costによって、cost列の値は必ず0(ゼロ)より大きくなります。

この例は、制約句と列定義について、次の点についても示しています。

- 表外制約定義は、列定義の前と後のどちらにも指定できます。この例では、pk_od制約の表外定義が、列定義の前にあります。
- 列定義には、複数の表内制約定義を含めることができます。この例では、quantity列の定義はnn_qty制約とcheck_qty制約の両方の定義を含んでいます。
- 表には、複数のCHECK制約を指定できます。複数のビジネス・ルールを適用する複雑な条件を持つ1つのCHECK制約よりも、それぞれ1つのビジネス・ルールのみを適用する単純な条件を持つ複数のCHECK制約を使用してください。矛盾している制約があると、その制約を識別するエラー・メッセージが戻されます。エラーが検出された制約が1つのビジネス・ルールのみを有効にする場合、このようなエラー・メッセージの方が、矛盾のあるビジネス・ルールを正確に識別できます。

大/小文字を区別しない制約の例

次の文は、親子関係の2つの表を作成します。親表は製品説明の表、子表は製品コンポーネント説明の表です。製品と説明の値が明確になるように、一意制約が定義されています。説明のため、製品とコンポーネントIDは大/小文字を区別しない文字値になっています。(現実世界のアプリケーションでは、通常、主キーIDは数値であるか、大/小文字が正規化されています。)

```
CREATE TABLE products
    ( product_id VARCHAR2(20) COLLATE BINARY_CI
           CONSTRAINT product_pk PRIMARY KEY
    , description VARCHAR2(1000) COLLATE BINARY_CI
           CONSTRAINT product_description_unq UNIQUE
    );
CREATE TABLE product_components
    ( component_id VARCHAR2(40) COLLATE BINARY_CI
           CONSTRAINT product_component_pk PRIMARY KEY
    , product_id CONSTRAINT product_component_fk REFERENCES products(product_id)
    , description VARCHAR2(1000) COLLATE BINARY_CI
           CONSTRAINT product_component_descr_unq UNIQUE
    );
```

外部キー列のデータ型または照合を指定しない場合は、親キー列から継承されます。

次の文は、製品とそのコンポーネントを表に追加します。

```
INSERT INTO products(product_id, description)
            VALUES('BICY0001', 'Men''s bicycle, fr 21", wh 24", gear 3x7');
INSERT INTO product_components(component_id, product_id, description)
            VALUES('BICY0001_FRAME01', 'BICY0001', 'Aluminium frame 21"');
```

```
INSERT INTO product_components(component_id, product_id, description)
            VALUES('BICY0001_WHEEL01', 'bicy0001', 'Wheels 24"');
INSERT INTO product_components(component_id, product_id, description)
            VALUES('BICY0001_GEAR01', 'Bicy0001', 'Front derailleur 3 chainrings');
INSERT INTO product_components(component_id, product_id, description)
            VALUES('BICY0001_gear02', 'BiCy0001', 'Rear derailleur 7 chainrings');
```

製品IDの大/小文字がコンポーネント行ごとに異なることに注意してください。製品IDの主キーは大/小文字を区別しないものとして宣言されているため、同じIDについて使用可能なすべての大/小文字の組合せは等しいものとみなされます。

次の文は、大/小文字のみ異なる同じ説明では別の製品を入力できないことを示しています。これは、エラーORA-00001: unique constraint (schema.PRODUCT_DESCRIPTION_UNQ) violatedで失敗します。

```
INSERT INTO products(product_id, description)
            VALUES('BICY0002', 'MEN''S BICYCLE, fr 21", wh 24", gear 3x7');
```

同様に、次の文は、製品表の主キー制約が大/小文字を区別せず、大/小文字のみ異なる値は使用できないことを示しています。これは、エラーORA-00001: unique constraint (schema.PRODUCT_PK) violatedで失敗します。

```
INSERT INTO products(component_id, product_id, description)
            VALUES('bicy0001', 'Women''s bicycle, fr 21", wh 24", gear 2x6');
```

次の文は、大/小文字のみ異なる同じ説明では別のコンポーネントを入力できないことを示しています。これは、エラーORA-00001: unique constraint (schema.PRODUCT_COMPONENT_DESCR_UNQ) violatedで失敗します。

```
INSERT INTO product_components(component_id, product_id, description)
            VALUES('BICY0001_gear03', 'BiCy0001', 'REAR DERAILLEUR 7 CHAINRINGS');
```

属性レベル制約の例

次の文は、students表のname列のfirst_name属性とlast_name属性の両方に対して値が存在することを保証します。

```
CREATE TYPE person_name AS OBJECT
   (first_name VARCHAR2(30), last_name VARCHAR2(30));
/
CREATE TABLE students (name person_name, age INTEGER,
   CHECK (name.first_name IS NOT NULL AND
          name.last_name IS NOT NULL));
```

REF制約の例

次の例は、サンプル・スキーマのオブジェクト型であるcust_address_typの複製を作成し、SCOPE制約が指定されたREF列を含む表を作成します。

```
CREATE TYPE cust_address_typ_new AS OBJECT
    ( street_address     VARCHAR2(40)
    , postal_code        VARCHAR2(10)
    , city               VARCHAR2(30)
    , state_province     VARCHAR2(10)
    , country_id         CHAR(2)
    );
/
CREATE TABLE address_table OF cust_address_typ_new;
CREATE TABLE customer_addresses (
   add_id NUMBER,
   address REF cust_address_typ_new
   SCOPE IS address_table);
```

次の例は、同じ表を作成しますが、親表のオブジェクト識別子列を参照するREF列に参照整合性制約が指定されています。

```
CREATE TABLE customer_addresses (
   add_id NUMBER,
   address REF cust_address_typ REFERENCES address_table);
```

次の例では、department_typ型およびdepartments_obj_t表(「オブジェクト表の作成: 例」で作成)を使用します。この文を実行すると、有効範囲付きREFを持つ表が作成されます。

```
CREATE TABLE employees_obj
   ( e_name   VARCHAR2(100),
     e_number NUMBER,
     e_dept   REF department_typ SCOPE IS departments_obj_t );
```

次の文は、参照整合性制約が定義されているREF列を含む表を作成します。

```
CREATE TABLE employees_obj
   ( e_name   VARCHAR2(100),
     e_number NUMBER,
     e_dept   REF department_typ REFERENCES departments_obj_t);
```

明示的な索引の制御例

次の文は、Oracleが制約を適用する場合に使用する索引を、明示的に制御する一意制約または主キー制約を作成する別の方法を示します。

```
CREATE TABLE promotions_var3
    ( promo_id          NUMBER(6)
    , promo_name        VARCHAR2(20)
    , promo_category    VARCHAR2(15)
    , promo_cost        NUMBER(10,2)
    , promo_begin_date DATE
    , promo_end_date    DATE
    , CONSTRAINT promo_id_u UNIQUE (promo_id, promo_cost)
         USING INDEX (CREATE UNIQUE INDEX promo_ix1
            ON promotions_var3 (promo_id, promo_cost))
    , CONSTRAINT promo_id_u2 UNIQUE (promo_cost, promo_id)
         USING INDEX promo_ix1);
```

この例は、1つの制約に対して1つの索引を作成し、その索引を使用して同じ文に別の制約を作成して使用可能にできることも示しています。

DEFERRABLE制約の例

次の文は、scores列にCHECK制約NOT DEFERRABLE INITIALLY IMMEDIATE(デフォルト)を使用したgames表を作成します。

```
CREATE TABLE games (scores NUMBER CHECK (scores >= 0));
```

次の文は、列に対する一意制約をINITIALLY DEFERRED DEFERRABLEとして定義します。

```
CREATE TABLE games
  (scores NUMBER, CONSTRAINT unq_num UNIQUE (scores)
   INITIALLY DEFERRED DEFERRABLE);
```

deallocate_unused_clause

目的

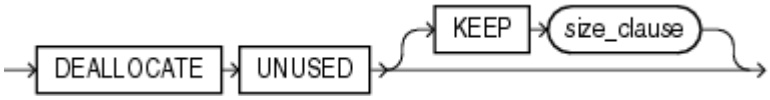
deallocate_unused_clauseを使用すると、データベース・オブジェクト・セグメントの終わりにある未使用領域の割当てを明示的に解除し、解放された領域を表領域の他のセグメントで使用できるようになります。

未使用領域の割当ては、次の文で解除できます。

- ALTER CLUSTER(「ALTER CLUSTER」を参照)
- ALTER INDEX: 未使用領域の割当てを索引、索引パーティションまたは索引サブパーティションから解除する場合(「ALTER INDEX」を参照)
- ALTER MATERIALIZED VIEW: 未使用領域の割当てを索引構成マテリアライズド・ビューのオーバーフロー・セグメントから解除する場合(「ALTER MATERIALIZED VIEW」を参照)
- ALTER TABLE: 未使用領域の割当てを表、表パーティション、表サブパーティション、索引構成表のマッピング表、索引構成表のオーバーフロー・セグメントまたはLOB記憶域セグメントから解除する場合(「ALTER TABLE」を参照)

構文

deallocate_unused_clause::=



(size_clause::=)

セマンティクス

この項では、deallocate_unused_clauseのセマンティクスについて説明します。詳細は、特定のデータベース・オブジェクトに対してこの句を設定または再設定するSQL文の説明を参照してください。

同じ文でdeallocate_unused_clauseとallocate_extent_clauseの両方を指定することはできません。

最高水位標を超える(データベース・ブロックがデータを受け取るためにフォーマットされていない)未使用領域のみ解放できます。未使用領域の割当て解除は、オブジェクトの終わりから開始し、オブジェクトの先頭にある最高水位標に向かって進行します。

エクステントが割当て解除の範囲内に完全に含まれる場合、エクステント全体が解放されて再利用可能となります。エクステントの一部が含まれる場合、最高水位標までのすでに使用されている部分がエクステントになり、残りの未使用領域が解放されて再利用可能になります。

割当てが解除される表領域のユーザー割当て制限は、解放される領域の量のみ増加します。

解放される領域の実際の量は、INITIAL、MINEXTENTSおよびNEXT記憶域パラメータによって異なります。これらのパラメータの詳細は、「storage_clause」を参照してください。

KEEP integer

割当て解除後に、データベース・オブジェクトのセグメントが保持する、最高水位標を超えるバイト数を指定します。

- KEEPを指定しなかった場合に、最高水位標がINITIALおよびMINEXTENTSのサイズを超えると、最高水位標を超えるすべての未使用領域が解放されます。最高水位標がINITIALまたはMINEXTENTSのサイズより小さい場合は、MINEXTENTSを超えるすべての未使用領域が解放されます。

- KEEPオプションを指定した場合、指定した量の領域が保持され、残りの領域のみが解放されます。このとき、残りのエクステント数がMINEXTENTSより小さくなった場合、MINEXTENTSはそのエクステント数に変更されます。また、初期エクステントがINITIALより小さくなった場合、INITIALがそのサイズに変更されます。
- どちらの場合にも、NEXT記憶域パラメータの値は、割当て解除された最後のエクステントのサイズに設定されます。

file_specification

目的

`file_specification`構文のいずれかを使用すると、ファイルをデータファイルまたは一時ファイルとして指定できます。また、1つ以上のファイルのグループを1つのREDOログ・ファイル・グループとして指定することもできます。Oracle Automatic Storage Management (Oracle ASM)ディスク・グループにファイルを格納する場合、そのファイルをディスク・グループ・ファイルとしても指定できます。

`file_specification`は次の文に指定できます。

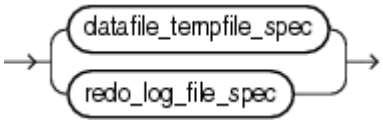
- `CREATE CONTROLFILE`(「CREATE CONTROLFILE」を参照)
- `CREATE DATABASE`(「CREATE DATABASE」を参照)
- `ALTER DATABASE`(「ALTER DATABASE」を参照)
- `CREATE TABLESPACE`(「CREATE TABLESPACE」を参照)
- `ALTER TABLESPACE`(「ALTER TABLESPACE」を参照)
- `ALTER DISKGROUP`(「ALTER DISKGROUP」を参照)

前提条件

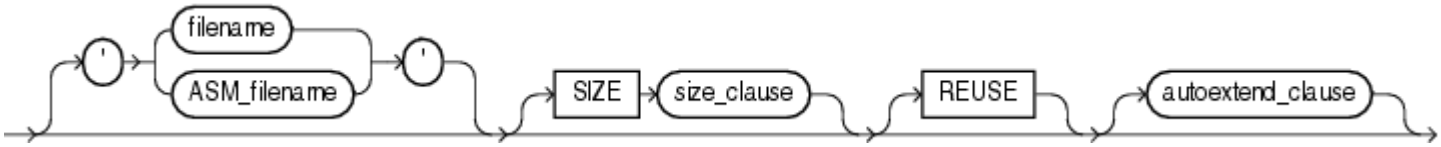
ファイルを指定する文を発行する権限が必要です。

構文

file_specification::=

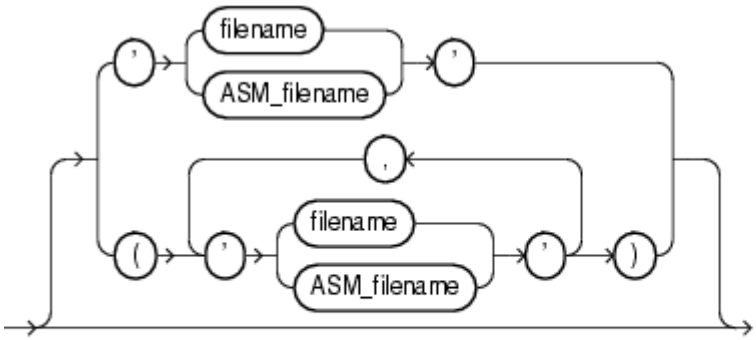


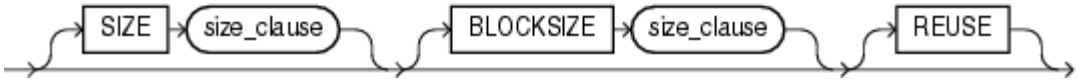
datafile_tempfile_spec::=



(size_clause::=)

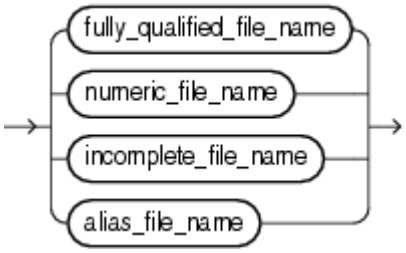
redo_log_file_spec::=



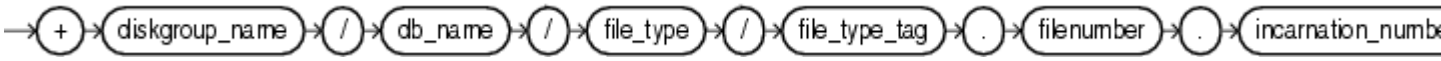


(size_clause::=)

ASM_filename::=

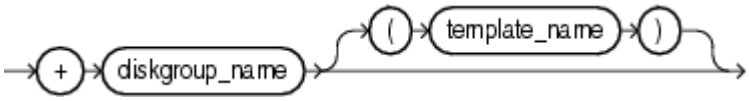


fully_qualified_file_name::=

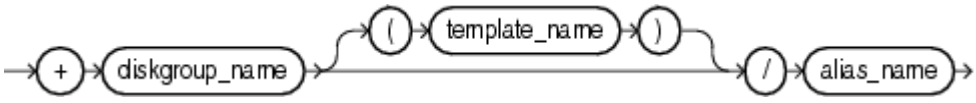


numeric_file_name::=

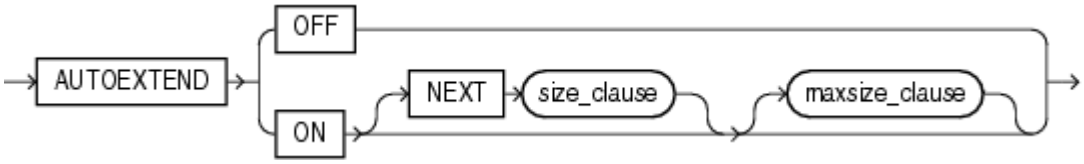
incomplete_file_name::=



alias_file_name::=

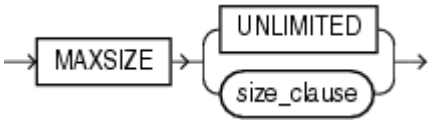


autoextend_clause::=



(size_clause::=)

maxsize_clause::=



(size_clause::=)

セマンティクス

この項では、file_specificationのセマンティクスについて説明します。詳細は、データファイル、一時ファイル、REDOログ・ファイルまたはOracle ASMディスク・グループやディスク・グループ・ファイルを指定するSQL文の説明を参照してください。

datafile_tempfile_spec

データベース記憶域がファイル・システム内またはOracle ASMディスク・グループ内に存在する場合、この句を使用してデータファイルと一時ファイルの属性を指定します。

redo_log_file_spec

データベース記憶域がファイル・システム内またはOracle ASMディスク・グループ内に存在する場合、この句を使用してREDOログ・ファイルの属性を指定します。

filename

filenameは、ファイル・システム内に格納されたファイルに使用します。filenameは、新しいファイルまたは既存のファイルを指定できます。新しいファイルを指定する場合、次のことに注意します。

- Oracle Managed Filesを使用していない場合は、filenameとSIZE句の両方を指定しないと、文は失敗します。サイズを指定せずにファイル名を指定すると、Oracleは既存のファイルを再利用し、ファイルが存在しない場合はエラーを戻そうとします。
- Oracle Managed Filesを使用している場合は、filenameや残りの句は必須ではありません。この場合、Oracle Databaseによってファイルに一意の名前が作成され、そのファイルが次のいずれかの初期化パラメータで指定されるディレクトリに保存されます。
    - DB_RECOVERY_FILE_DEST(ログ・ファイルと制御ファイルに有効)
    - DB_CREATE_FILE_DEST初期化パラメータ(すべてのファイル・タイプに有効)
    - DB_CREATE_ONLINE_LOG_DEST_n初期化パラメータ(ログ・ファイルに対してはDB_CREATE_FILE_DESTとDB_RECOVERY_FILE_DESTより優先される)。

既存のファイルを指定する場合は、データファイル、一時ファイルまたはREDOログ・ファイル・メンバーの名前を指定します。filenameには、7ビットASCII文字セットまたはEBCDIC文字セットのシングルバイト文字のみを使用できます。マルチバイト文字は無効です。

filenameには、パス接頭辞を含めることができます。パス接頭辞を指定しない場合は、データベースによってデフォルトの格納場所(プラットフォームによって異なる)のパス接頭辞が追加されます。

REDOログ・ファイル・グループは、1つ以上のメンバー(コピー)で構成されます。filenameは、使用するオペレーティング・システムの表記規則に従って、完全な名前で指定する必要があります。

データベースがfilenameを解析する方法は、SIZE句およびREUSE句の指定によっても異なります。

- filenameのみを指定する場合、またはREUSE句を指定してSIZE句を指定しない場合、そのファイルはすでに存在している必要があります。
- filenameの指定に、SIZE句を指定して、REUSE句を指定しない場合、そのファイルは新しいファイルである必要があります。
- filenameの指定に、SIZE句とREUSE句の両方を指定する場合、そのファイルは新しいファイル、既存のファイルのどちらでもかまいません。ファイルがすでに存在している場合は、そのファイルが新しいサイズで再利用されます。また、ファイルが存在しない場合は、データベースはREUSEキーワードを無視し、指定のサイズで新しいファイルを作成します。

関連項目:

Oracle Managed Filesの詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。また、「データファイルの指定: 例」および「ログ・ファイルの指定: 例」も参照してください。

ASM_filename

Oracle ASMディスク・グループに格納されたファイルには、ASM_filenameの形式を使用します。この構文でデータファイル、一時ファイルおよびREDOログ・ファイルを作成または参照できます。

すべてのASM_filename形式は、プラス記号(+)で始まりディスク・グループ名が続きます。すべてのOracle ASMディスク・グループの名前を確認するには、V$ASM_DISKGROUPビューを問い合せます。

関連項目:

Oracle ASMの使用については、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

fully_qualified_file_name

Oracle ASMディスク・グループにファイルを作成する場合、そのファイルにはシステム生成の完全修飾されたOracle ASMファイル名が付けられます。既存のOracle ASMファイルを参照する場合にのみ、この形式を使用できます。そのため、ファイル作成時にこの形式を使用する場合は、REUSEも指定する必要があります。

- db_nameは、DB_UNIQUE_NAME初期化パラメータの値です。この名前はファイルが存在するデータベースの名前と同じです。ただし、プライマリ・データベースとスタンバイ・データベースの両方が存在する場合は、パラメータはこれらを区別して異なる値になります。
- file_typeとfile_type_tagは、データベース・ファイル・タイプです。表8-1に、すべてのファイル・タイプとそれに対応するOracle ASMタグを示します。
- filenumberとincarnation_numberは、システム生成の識別子で、これによって一意性が保証されます。

完全修飾されたOracle ASMファイル名を確認するには、そのファイル・タイプに適切な動的パフォーマンス・ビュー(データファイルではV$DATAFILE、制御ファイルではV$CONTROLFILEなど)を問い合せます。また、V$ASM_FILEビューを問い合せることによって、完全修飾された名前のfilenumberとincarnation_numberの部分を取得できます。

表8-1 Oracleファイル・タイプとOracle ASMファイル・タイプ・タグ

Oracle ASM file_type	説明	Oracle ASM file_type_tag	コメント
CONTROLFILE	制御ファイルとバックアップ制御ファイル	Current Backup	—
DATAFILE	データファイルとデータファイル・コピー	tsname	ファイルが追加される表領域です。
ONLINELOG	オンライン・ログ	group_group#	—
ARCHIVELOG	アーカイブ・ログ	thread_thread#_seq_sequence#	—
TEMPFILE	一時ファイル	tsname	ファイルが追加される表領域です。
BACKUPSET	データファイルとアーカイブ・ログのバックアップ・ピース(データファイルの増分バックアップ・ピース)	hasspfile_timestamp	hasspfile の値は、s(バックアップ・セットが spfile を含む)または n(バックアップ・セットが spfile を含まない)のいずれかになります。
PARAMETERFILE	永続パラメータ・ファイル	spfile	—
DATAGUARDCONFIG	Data Guard 構成ファイル	db_unique_name	Data Guard では、DB_UNIQUE_NAME 初期化パラメータの値が使用されます。
FLASHBACK	フラッシュバック・ログ	log_log#	—
CHANGETRACKING	ブロック・チェンジ・トラッキング・データ	ctf	増分バックアップ中に使用されます。
DUMPSET	データ・ポンプ・ダンプセット	user_obj#_file#	ダンプ・セット・ファイルは、ユーザー名、ダンプ・セットが作成されたジョブ番号およびファイル番号をタグの一部としてエンコードします。

Oracle ASM file_type	説明	Oracle ASM file_type_tag	コメント
XTRANSPORT	データファイル変換	tsname	-
AUTOBACKUP	自動バックアップ・ファイル	hasspfile_timestamp	hasspfile の値は、s(バックアップ・セットが spfile を含む)または n(バックアップ・セットが spfile を含まない)のいずれかになります。

numeric_file_name

数値のOracle ASMファイル名は、一意のfilenumber.incarnation_number文字列のみが使用されることを除き、完全修飾されたファイル名に類似しています。既存のファイルを参照するためにのみ、この形式を使用できます。そのため、ファイル作成時にこの形式を使用する場合は、REUSEも指定する必要があります。

incomplete_file_name

不完全なOracle ASMファイル名は、ファイルの作成時にのみ使用されます。ディスク・グループ名のみを指定した場合、Oracle ASMではそのファイル・タイプに適したデフォルトのテンプレートが使用されます。たとえば、CREATE TABLESPACE文でデータファイルを作成する場合、Oracle ASMではデフォルトのDATAFILEテンプレートを使用してOracle ASMデータファイルが作成されます。ディスク・グループ名とテンプレートを指定した場合、Oracle ASMでは指定したテンプレートを使用してファイルが作成されます。いずれの場合も、Oracle ASMによって完全修飾されたファイル名が作成されます。

template_name

テンプレートは、属性の名前付きコレクションです。テンプレートを作成して、ディスク・グループのファイルに適用できます。すべてのOracle ASMテンプレートの名前を確認するには、V$ASM_TEMPLATEデータ・ディクショナリ・ビューを問い合せます。Oracle ASMテンプレートを作成する手順は、「diskgroup_template_clauses」を参照してください。

templateは、ファイルの作成時にのみ指定できます。これは、ASM_filenameの構文図の不完全なファイル名形式および別名形式に示されています。

- ディスク・グループ名の直後にtemplateを指定した場合、Oracle ASMでは指定したテンプレートを使用してファイルが作成され、そのファイルに完全修飾されたファイル名が付けられます。
- 別名を指定した後にtemplateを指定した場合、Oracle ASMでは指定したテンプレートを使用してファイルが作成され、そのファイルに完全修飾されたファイル名が付けられます。また、後でファイルの参照に使用できる別名も作成されます。指定した別名が既存のファイルを参照している場合は、REUSEを指定しないかぎり、Oracle ASMではテンプレートの指定が無視されます。

関連項目:

デフォルト・テンプレートについては、「diskgroup_template_clauses」を参照してください。

alias_file_name

別名は、Oracle ASMファイルのわかりやすい名前です。ファイルの別名は、ファイルの作成または参照時に使用できます。ファイルの作成時にのみ、別名とともにテンプレートを指定できます。Oracle ASMファイルの別名を確認するには、V$ASM_ALIASデータ・ディクショナリ・ビューを問い合せます。

ファイルの作成時に別名を指定する場合、完全な別名を指定する手順は、「diskgroup_directory_clauses」および「diskgroup_alias_clauses」を参照してください。

SIZE句

ファイルのサイズをバイト単位で指定します。K、M、GまたはTを使用して、KB、MB、GBまたはTB単位で指定することもできます。

- UNDO表領域の場合、各データファイルに対してSIZE句を指定する必要があります。その他の表領域の場合、ファイルがすでに存在するとき、またはOracle Managed Fileを作成するときには、このパラメータを省略できます。
- Oracle Managed Fileの作成時にこの句を指定しないと、100MBのファイルが作成されます。
- 表領域は、中に含まれるオブジェクトのサイズの合計より1ブロック大きいサイズである必要があります。

関連項目:

自動UNDO管理およびUNDO表領域の詳細は、『Oracle Database管理者ガイド』を参照してください。また、「ログ・ファイルの追加: 例」も参照してください。

BLOCKSIZE句

BLOCKSIZEを指定すると、オペレーティング・システム依存のセクター・サイズを上書きします。この句を省略した場合、オペレーティング・システム依存のセクター・サイズがブロック・サイズとして使用されます。

512バイト・セクターのディスク、または512バイトをエミュレートする4KBセクターのディスクにREDOログ・ファイルを追加する場合は、新しいファイルのブロック・サイズがプラットフォーム・ベースの元のサイズ(4KB)である必要があります。

- 512バイト・セクターのディスクにREDOログ・ファイルを追加する場合は、使用しているプラットフォームに応じて、ブロック・サイズを512または1024(1K)に指定する必要があります。
- 4KBセクターのディスク(ネイティブ)にREDOログ・ファイルを追加する場合は、ブロック・サイズを4096(4K)に指定する必要があります。
- 512バイトをエミュレートする4KBセクターのディスクにREDOログ・ファイルを追加する場合は、使用しているプラットフォームに応じて、ブロック・サイズを512、1024(1K)または4096(4K)のいずれかに指定できます。

ログ・グループ内のすべてのログが、同じブロック・サイズである必要があります。別々のディスクに2つのログ・グループを作成する場合は、ブロック・サイズが異なっていてもかまいません。ただし、構成が異なっていると、ログ・スイッチのたびにオーバーヘッドが発生します。すべてのログ・ファイルのブロック・サイズを同じにすることをお薦めします。

この句は、使用しているセクター・サイズが4Kの場合に、パフォーマンスよりもディスク使用量を重視して最適化する場合に有効です。この場合、BLOCKSIZE 512(HP-UXの場合はBLOCKSIZE 1024)を指定してオペレーティング・システムのセクター・サイズを上書きできます。

関連項目:

ログ・ファイルの追加: 例

REUSE

REUSEは、既存ファイルの再利用を可能にします。

- ファイルがすでに存在する場合、ファイル名が再利用され、新しいサイズが適用されるか(SIZEを指定する場合)、または元のサイズのままとなります。
- ファイルが存在していない場合、この句は無視され、ファイルが作成されます。

REUSE句の制限事項

filenameを指定していない場合は、REUSEを指定できません。

既存のファイルが使用される場合は、そのファイルに入っていた内容は失われます。

関連項目:

「データファイルの追加: 例」および「ログ・ファイルの追加: 例」を参照してください。

autoextend_clause

autoextend_clauseはデータファイルと一時ファイルに有効ですが、REDOログ・ファイルには有効ではありません。この句は、新しいデータファイル、既存のデータファイルまたは一時ファイルの自動拡張を使用可能または使用禁止にします。この句を指定しない場合、次のようになります。

- Oracle Managed Filesの場合
  - SIZEを指定すると、指定したサイズのファイルが作成され、AUTOEXTENDが使用禁止になります。
  - SIZEを指定しないと、100MBのファイルが作成され、AUTOEXTENDが使用可能になります。自動拡張が必要な場合、データベースは元のサイズまたは100MB(いずれか小さい方)のサイズ単位でファイルを拡張します。NEXT句を指定して、このデフォルトの動作を上書きできます。
- ユーザー管理ファイルの場合、SIZEの指定/未指定にかかわらず、AUTOEXTENDが使用禁止にされたファイルが作成されます。

ON

ONを指定すると、自動拡張を有効にします。

OFF

OFFを指定すると、自動拡張を使用禁止にします。自動拡張を使用禁止にする場合は、NEXTおよびMAXSIZEの値を0(ゼロ)に設定します。後続の文で自動拡張を使用可能に戻す場合は、これらの値を設定しなおす必要があります。

NEXT

NEXT句を使用すると、エクステントがさらに必要になった場合にデータファイルに自動的に割り当てられるディスク領域の増分サイズを、バイト単位で指定できます。デフォルトのサイズは1データ・ブロックです。

MAXSIZE

MAXSIZE句を使用すると、データファイルの自動拡張で使用されるディスク領域の最大サイズを指定できます。

UNLIMITED

データファイルまたは一時ファイルに割り当てられるディスク領域を制限しない場合は、UNLIMITED句を使用します。

autoextend_clauseの制限事項

この句は、CREATE CONTROLFILE文またはALTER DATABASE CREATE DATAFILE句で datafile_tempfile_specの一部として指定できません。

例

ログ・ファイルの指定: 例

次の文は、payableという名前のデータベースを作成します。このデータベースには各グループにメンバーを2つ持つREDOログ・ファイル・グループが2つと、データファイルが1つ設定されています。

```
CREATE DATABASE payable
   LOGFILE GROUP 1 ('diska:log1.log', 'diskb:log1.log') SIZE 50K,
           GROUP 2 ('diska:log2.log', 'diskb:log2.log') SIZE 50K
   DATAFILE 'diskc:dbone.dbf' SIZE 30M;
```

LOGFILE句の最初のファイル指定は、REDOログ・ファイル・グループにGROUP 1の値を指定します。このグループには、'diska:log1.log'および'diskb:log1.log'というメンバーがあり、サイズはそれぞれ50KBです。

LOGFILE句の2番目のファイル指定は、REDOログ・ファイル・グループにGROUP 2の値を指定します。このグループには、'diska:log2.log'および'diskb:log2.log'というメンバーがあり、サイズはそれぞれ50KBです。

DATAFILE句のファイル指定では、'diskc:dbone.dbf'という名前のデータファイルが指定されています。このファイルのサイズは30MBです。

各ファイル指定はSIZEパラメータの値を指定し、REUSE句は指定していないため、指定のファイルがすでに定義されていることはありません。Oracleが新しくファイルを作成します。

ログ・ファイルの追加: 例

次の文は、2つのメンバーで構成される新しいREDOログ・ファイル・グループを、payableデータベースに追加します。

```
ALTER DATABASE payable
   ADD LOGFILE GROUP 3 ('diska:log3.log', 'diskb:log3.log')
   SIZE 50K REUSE;
```

ADD LOGFILE句のファイル指定は、REDOログ・ファイル・グループにGROUP 3の値を指定します。このグループには、'diska:log3.log'および'diskb:log3.log'というメンバーがあり、サイズはそれぞれ50KBです。このファイル指定ではREUSE句が指定されているため、各メンバーはすでに存在している可能性があります(ただし、存在していなくてもかまいません)。

次の文は移行先のディスク4k_disk_aおよび4k_disk_b上にメンバー・ログ・ファイルを持つログ・ファイル・グループ5を追加します。この文の実行後は、switch_logfile_clauseを使用して、ブロック・サイズが512バイトのディスク上にある既存のログ・ファイルを、ブロック・サイズが4Kのログに切り替えることができます。

```
ALTER DATABASE ADD LOGFILE GROUP 5
     ('4k_disk_a:log5.log', '4k_disk_b:log5.log')
     SIZE 100M BLOCKSIZE 4096 REUSE;
```

データファイルの指定: 例

次の文は、3つのデータファイルを持つstocksという表領域を作成します。

```
CREATE TABLESPACE stocks
   DATAFILE 'stock1.dbf' SIZE 10M,
            'stock2.dbf' SIZE 10M,
            'stock3.dbf' SIZE 10M;
```

このデータファイルのファイル指定は、'diskc:stock1.dbf'、'diskc:stock2.dbf'および'diskc:stock3.dbf'という名前のファイルを指定します。

データファイルの追加: 例

次の文は、stocks表領域を変更し、新しいデータファイルを追加します。

```
ALTER TABLESPACE stocks
   ADD DATAFILE 'stock4.dbf' SIZE 10M REUSE;
```

このファイル指定は、'stock4.dbf'という名前のデータファイルを指定します。このファイル名が存在しない場合、REUSEキーワードは無視されます。

完全修飾Oracle ASMデータファイル名の使用方法: 例

次の文は、Oracle ASMを使用する際に、fully_qualified_file_name句を使用して、仮想データベースtestdbのデータファイルをオンラインにする方法を示します。

```
ALTER DATABASE testdb
   DATAFILE '+dgroup_01/testdb/datafile/system.261.1' ONLINE;
```

logging_clause

目的

logging_clauseを使用すると、特定のDML操作をREDOログ・ファイルに記録するか(LOGGING)、記録しないか(NOLOGGING)を指定できます。

logging_clauseは次の文に指定できます。

- CREATE TABLEおよびALTER TABLE: 表、表のパーティション、LOBセグメント、または索引構成表のオーバーフロー・セグメントのログを記録する場合(「CREATE TABLE」および「ALTER TABLE」を参照)。

  ノート:

  LOB 列に指定されたロギングは、表レベルで設定されたロギングとは異なる場合があります。表レベルで LOGGING を指定し、LOB 列に NOLOGGING を指定する場合は、実表の行に対する DML の変更は記録されますが、LOB データに対する DML の変更は記録されません。

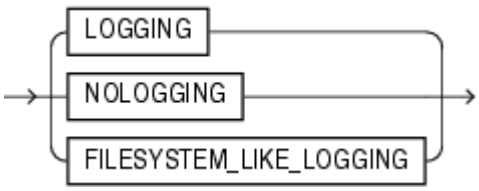
- CREATE INDEXおよびALTER INDEX: 索引または索引のパーティションのログを記録する場合(「CREATE INDEX」および「ALTER INDEX」を参照)。
- CREATE MATERIALIZED VIEWおよびALTER MATERIALIZED VIEW: マテリアライズド・ビュー、マテリアライズド・ビューのパーティションの1つまたはLOBセグメントのログを記録する場合(「CREATE MATERIALIZED VIEW」および「ALTER MATERIALIZED VIEW」を参照)。
- CREATE MATERIALIZED VIEW LOGおよびALTER MATERIALIZED VIEW LOG: マテリアライズド・ビュー・ログまたはマテリアライズド・ビュー・ログのパーティションの1つのログを記録する場合(「CREATE MATERIALIZED VIEW LOG」および「ALTER MATERIALIZED VIEW LOG」を参照)。
- CREATE TABLESPACEおよびALTER TABLESPACE: 表領域に作成されたすべてのオブジェクトに対するデフォルトのロギング特性を設定または変更する場合(「CREATE TABLESPACE」および「ALTER TABLESPACE」を参照)。
- CREATE PLUGGABLE DATABASEおよびALTER PLUGGABLE DATABASE: プラガブル・データベース(PDB)に作成されたすべての表領域のデフォルトのロギング特性を設定または変更する場合(CREATE PLUGGABLE DATABASEおよびALTER PLUGGABLE DATABASEを参照)。

次の操作にも、LOGGINGまたはNOLOGGINGを指定できます。

- 索引の再構築(CREATE INDEX ... REBUILDを使用)
- 表の移動(ALTER TABLE ... MOVEを使用)

構文

logging_clause::=



セマンティクス

この項では、logging_clauseのセマンティクスについて説明します。詳細は、特定のデータベース・オブジェクトに対してロギング特性を設定または再設定するSQL文の説明を参照してください。

- LOGGINGを指定すると、データベース・オブジェクトの作成、およびその後のオブジェクトへの挿入をREDOログ・ファイルに記録します。
- NOLOGGINGを指定すると、データベース・オブジェクトの作成、およびその後の従来型INSERTをREDOログ・ファイルに記録します。ダイレクト・パス・インサートは記録されません。
    - 非パーティション・オブジェクトの場合、この句に指定する値は、オブジェクトに関連付けられたセグメントの実際の物理属性となります。
    - パーティション・オブジェクトの場合、この句に指定する値は、PARTITION記述でロギング属性を指定しないかぎり、CREATE文(および後続のALTER ... ADD PARTITION文)で指定するすべてのパーティションに関連付けられたセグメントのデフォルトの物理属性となります。
    - SecureFiles LOBでは、NOLOGGING設定はFILESYSTEM_LIKE_LOGGINGに内部変換されます。
    - CACHE NOLOGGINGは、BasicFiles LOBに対しては許可されません。
- FILESYSTEM_LIKE_LOGGING句は、SecureFiles LOBセグメントのログを記録する場合にのみ有効です。この設定は、BasicFiles LOBには指定できません。メタデータ変更のログのみを記録する場合は、この設定を指定します。この設定は、障害からリカバリする平均時間を短縮するファイル・システムのメタデータ・ジャーナリングに似ています。SecureFiles LOBのLOGGING設定は、ファイル・システムのデータ・ジャーナルに似ています。LOGGINGとFILESYSTEM_LIKE_LOGGINGの両方の設定によって、SecureFileを使用した完全なトランザクション・ファイル・システムが実現します。

ノート:

LOB セグメントの場合は、NOLOGGING 設定と FILESYSTEM_LIKE_LOGGING 設定を使用すると、バックアップ操作中にディスク上でデータが変更され、バックアップの一貫性が損われる可能性があります。この状況を回避するには、LOB 記憶域に LOGGING を設定し、LOB セグメントに対する変更が REDO ログ・ファイルに保存されるようにします。または、データベースを FORCE LOGGING モードに変更し、すべての LOB セグメントに対する変更が REDO ログ・ファイルに保存されるようにします。

ロギング属性を指定しているオブジェクトが強制ロギング・モードのデータベースまたは表領域に存在している場合、そのデータベースまたは表領域が強制ロギング・モードから別のモードに変わるまで、NOLOGGING設定は無視されます。

データベースをARCHIVELOGモードで実行している場合、LOGGING操作の前に取ったバックアップからのメディア・リカバリによって、オブジェクトが再作成されます。ただし、NOLOGGING操作の前に取ったバックアップからのメディア・リカバリでは、オブジェクトは再作成されません。

NOLOGGINGモードでの操作で生成されるREDOログのサイズは、LOGGINGモードで生成されるログより非常に小さくなります。

NOLOGGINGモードでは、データの変更時に、(新しいエクステントをINVALIDとしてマーク設定し、ディクショナリの変更を記録するために)最小限のログが記録されます。メディア・リカバリ中にNOLOGGINGが適用された場合、REDOデータのログ記録が中断されるため、エクステント無効化レコードでは、一定のブロック範囲に「論理的に無効」というマークが付きます。このため、損

失してはならないデータベース・オブジェクトの場合は、NOLOGGING操作の後にバックアップを取る必要があります。

NOLOGGINGは、LOGGINGをサポートする場所のサブセットのみでサポートされます。次の操作でのみ、NOLOGGINGモードがサポートされます。

DML:

- INSERT文またはMERGE文のいずれかの結果として実行されるダイレクト・パスINSERT(シリアルまたはパラレル)。NOLOGGINGは、MERGE文の結果として実行されるUPDATE操作には適用できません。
- ダイレクト・ローダー(SQL*Loader)

DDL:

- CREATE TABLE ... AS SELECT(NOLOGGINGモードでは、表の作成は記録されますが、ダイレクト・パス・インサートは記録されません。)
- CREATE TABLE ... LOB_storage_clause ... LOB_parameters ... CACHE | NOCACHE | CACHE READS
- ALTER TABLE ... LOB_storage_clause ... LOB_parameters ... CACHE | NOCACHE | CACHE READS(新しく作成したLOB列のロギングを指定する場合)
- ALTER TABLE ... modify_LOB_storage_clause ... modify_LOB_parameters ... CACHE | NOCACHE | CACHE READS(既存のLOB列のロギングを変更する場合)
- ALTER TABLE ... MOVE
- ALTER TABLE ... (データ移動を伴うすべてのパーティション操作)
    - ALTER TABLE ... ADD PARTITION(ハッシュ・パーティションのみ)
    - ALTER TABLE ... MERGE PARTITIONS
    - ALTER TABLE ... SPLIT PARTITION
    - ALTER TABLE ... MOVE PARTITION
    - ALTER TABLE ... MODIFY PARTITION ... ADD SUBPARTITION
    - ALTER TABLE ... MODIFY PARTITION ... COALESCE SUBPARTITION
- CREATE INDEX
- ALTER INDEX ... REBUILD
- ALTER INDEX ... REBUILD [SUB]PARTITION
- ALTER INDEX ... SPLIT PARTITION

LOB以外のオブジェクトの場合、この句を指定しないと、オブジェクトが存在する表領域のロギング属性がオブジェクトのデフォルトのロギング属性になります。

LOBに対してこの句を省略した場合は、次のようになります。

- CACHEを指定した場合は、LOGGINGが使用されます(CACHE NOLOGGINGは指定できないため)。
- NOCACHEまたはCACHE READSを指定した場合は、表が存在する表領域の属性がデフォルトのロギング属性として使用されます。

NOLOGGINGは、内部に(行データとともに表に)格納されたLOBには適用されません。LOBに対するNOLOGGINGを4000バ

イト未満の値に指定し、STORAGE IN ROWを使用禁止にしていなかった場合、NOLOGGINGの指定は無視され、LOBデータは他の表データと同様に扱われます。

parallel_clause

目的

parallel_clauseを使用すると、データベース・オブジェクト作成のパラレル化、およびその後のオブジェクトに対する問合せのデフォルトの並列度と、オブジェクトに対するDML操作の設定が可能になります。

parallel_clauseは次の文に指定できます。

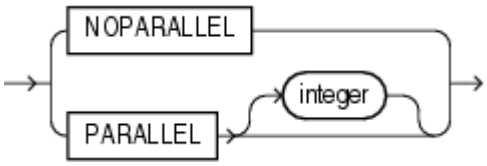
- CREATE TABLE: 表のパラレル化を設定する場合(「CREATE TABLE」を参照)
- ALTER TABLE(「ALTER TABLE」を参照)
    - 表のパラレル化を変更する場合
    - 表パーティションの追加、結合、交換、マージ、分割、切捨て、削除または移動の操作をパラレル化する場合
- CREATE CLUSTERおよびALTER CLUSTER: クラスタのパラレル化を設定または変更する場合(「CREATE CLUSTER」および「ALTER CLUSTER」を参照)。
- CREATE INDEX: 索引のパラレル化を設定する場合(「CREATE INDEX」を参照)
- ALTER INDEX(「ALTER INDEX」を参照)
    - 索引のパラレル化を変更する場合
    - 索引の再構築または索引パーティションの分割をパラレル化する場合
- CREATE MATERIALIZED VIEW: マテリアライズド・ビューのパラレル化を設定する場合(「CREATE MATERIALIZED VIEW」を参照)
- ALTER MATERIALIZED VIEW(「ALTER MATERIALIZED VIEW」を参照)
    - マテリアライズド・ビューのパラレル化を変更する場合
    - マテリアライズド・ビュー・パーティションの追加、結合、交換、マージ、分割、切捨て、削除または移動の操作をパラレル化する場合
    - マテリアライズド・ビュー・サブパーティションの追加または移動の操作をパラレル化する場合
- CREATE MATERIALIZED VIEW LOG: マテリアライズド・ビュー・ログのパラレル化を設定する場合(「CREATE MATERIALIZED VIEW LOG」を参照)
- ALTER MATERIALIZED VIEW LOG(「ALTER MATERIALIZED VIEW LOG」を参照)
    - マテリアライズド・ビュー・ログのパラレル化を変更する場合
    - マテリアライズド・ビュー・ログ・パーティションの追加、結合、交換、マージ、分割、切捨て、削除または移動の操作をパラレル化する場合
- ALTER DATABASE ... RECOVER: データベースをリカバリする場合(「ALTER DATABASE」を参照)
- ALTER DATABASE ... standby_database_clauses: スタンバイ・データベースに対する操作をパラレル化する場合(「ALTER DATABASE」を参照)。

関連項目:

DBMS_PARALLEL_EXECUTEパッケージ(行のチャンクの変更を表に適用するメソッドを含むパッケージ)の詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。エラーがない場合は、各チャンクに対する変更が個別にコミットされます。

構文

parallel_clause::=



セマンティクス

この項では、parallel_clauseのセマンティクスについて説明します。詳細は、特定のデータベース・オブジェクトまたは操作に対してパラレル化を設定または再設定するSQL文の説明を参照してください。

ノート:

parallel_clause 構文は、以前のリリースの構文にかわるものです。以前のリリースの構文は下位互換用にサポートされていますが、説明されている動作とわずかに異なることがあります。

parallel_clauseは、PARALLEL_DEGREE_POLICY初期化パラメータの設定に基づいて解釈されます。この初期化パラメータがAUTOに設定されている場合、parallel_clauseは完全に無視され、オプティマイザによってすべての文に対して最適な並列度が決定されます。PARALLEL_DEGREE_POLICYがMANUALまたはLIMITEDのいずれかに設定されている場合、parallel_clauseは次のように解釈されます。

NOPARALLEL

NOPARALLELを指定すると、シリアル実行が行われます。これはデフォルトです。

PARALLEL

PARALLELを指定すると、パラレル実行が行われます。

- PARALLEL_DEGREE_POLICYがMANUALに設定されている場合は、オプティマイザによって並列度が計算されます。並列度は、すべての関係するインスタンスで使用可能なCPUの数に、初期化パラメータPARALLEL_THREADS_PER_CPUの値を掛けたものです。
- PARALLEL_DEGREE_POLICYがLIMITEDに設定されている場合は、オプティマイザによって最適な並列度が決定されます。

PARALLEL integer

integerには、パラレル操作で使用するパラレル・スレッド数である並列度を指定します。各パラレル・スレッドは、1、2個のパラレル実行サーバーを使用します。

parallel_clauseのノート

parallel_clauseには、次のノートがあります。

- トリガーまたは参照整合性制約を定義した表に対するDML操作では、パラレル化を使用できません。

- 索引構成表でのUPDATEまたはDELETE操作に対して、パラレル化はサポートされません。
- 表の作成中にparallel_clauseを指定する際に、表にLOB型またはユーザー定義オブジェクト型の列が含まれている場合、このLOB型またはオブジェクト型の列を変更する後続のINSERT、UPDATE、DELETEおよびMERGE操作は、通知なしにシリアル実行されます。ただし、後続の問合せはパラレルで実行されます。
- parallel_clauseの効果はパラレル・ヒントによって上書きされます。
- リモート・オブジェクトを参照するDML文およびCREATE TABLE ... AS SELECT文は、パラレルで実行されます。ただし、リモート・オブジェクトはリモート・データベースにある必要があります。参照は、ローカル・データベースにあるオブジェクトにループバックできません。たとえば、ローカル・データベースのオブジェクトを指定するリモート・データベースのシノニムから参照することはできません。
- LOB列を含む表でのDML操作はパラレル化できます。ただし、パーティション内並列性はサポートされていません。

関連項目:

パラレル化操作の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。また「表の作成: パラレル化の例」も参照してください。

physical_attributes_clause

目的

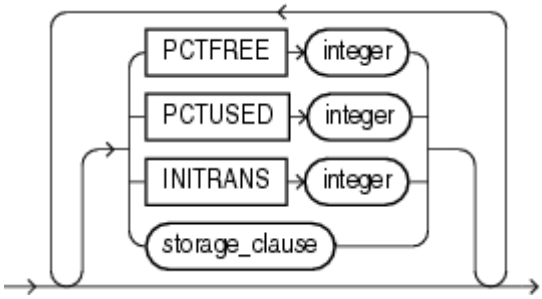
physical_attributes_clauseを使用すると、PCTFREE、PCTUSEDおよびINITRANSパラメータの値、および表、クラスタ、索引またはマテリアライズド・ビューの記憶特性を指定できます。

physical_attributes_clauseは次の文に指定できます。

- CREATE CLUSTERおよびALTER CLUSTER: クラスタおよびクラスタ内のすべての表の物理属性を設定または変更する場合(「CREATE CLUSTER」および「ALTER CLUSTER」を参照)。
- CREATE TABLE: 表、表パーティション、オブジェクト表のOIDINDEXまたは索引構成表のオーバーフロー・セグメントの物理属性を設定する場合(「CREATE TABLE」を参照)
- ALTER TABLE: 表の物理属性、将来追加される表パーティションのデフォルトの物理属性、または既存の表パーティションの物理属性を変更する場合(「ALTER TABLE」を参照)。次の制限があります。
  - 物理属性は、一時表には指定できません。
  - 物理属性は、クラスタ化表には指定できません。クラスタ内の表はクラスタの物理属性を継承します。
- CREATE INDEX: 索引または索引パーティションの物理属性を設定する場合(「CREATE INDEX」を参照)
- ALTER INDEX: 索引の物理属性、将来追加される索引パーティションのデフォルトの物理属性、または既存の索引パーティションの物理属性を変更する場合(「ALTER INDEX」を参照)
- CREATE MATERIALIZED VIEW: マテリアライズド・ビュー、そのパーティションの1つ、またはマテリアライズド・ビューを保持するために生成される索引の物理属性を設定する場合(「CREATE MATERIALIZED VIEW」を参照)
- ALTER MATERIALIZED VIEW: マテリアライズド・ビューの物理属性、将来追加されるパーティションのデフォルトの物理属性、および既存のパーティション、またはマテリアライズド・ビューを保持するために作成される索引の物理属性を変更する場合(「ALTER MATERIALIZED VIEW」を参照)
- CREATE MATERIALIZED VIEW LOGおよびALTER MATERIALIZED VIEW LOG: マテリアライズド・ビュー・ログの物理属性を設定または変更する場合(「CREATE MATERIALIZED VIEW LOG」および「ALTER MATERIALIZED VIEW LOG」を参照)。

構文

physical_attributes_clause::=



(storage_clause::=)

セマンティクス

この項では、physical_attributes_clauseのパラメータについて説明します。詳細は、特定のデータベース・オブジェク

トに対してこれらのパラメータを設定または再設定するSQL文の説明を参照してください。

PCTFREE integer

データベース・オブジェクトの各データ・ブロック内で、オブジェクトの行を将来更新するために確保しておく領域の割合を表す整数値を指定します。PCTFREEの値は、0から99の値にする必要があります。値に0を指定した場合は、ブロック全体が一杯になるまで新しい行を挿入できます。デフォルト値は10です。10を指定した場合、既存の行に対して行われる更新用に各ブロックの10%が確保されるため、各ブロックでは最大90%まで表に新しい行を挿入できます。

PCTFREEは、表、パーティション、クラスタ、索引、マテリアライズド・ビュー、マテリアライズド・ビュー・ログおよびゾーン・マップを作成および変更する文の中で同様に機能します。PCTFREEとPCTUSEDの組合せによって、新しい行を既存のデータ・ブロックと新しいデータ・ブロックのどちらに挿入するかが決まります。「PCTFREEとPCTUSEDの連携」を参照してください。

PCTFREE句の制限事項

索引を変更する場合は、このパラメータを`modify_index_default_attrs`句および`split_index_partition`句でのみ指定できます。

PCTUSED integer

使用済領域のうち、データベース・オブジェクトのデータ・ブロックごとに確保される最小限の割合を表す整数値を指定します。PCTUSEDは0から99までの正の整数で指定し、デフォルト値は40です。

PCTUSEDは、表、パーティション、クラスタ、マテリアライズド・ビュー、マテリアライズド・ビュー・ログおよびゾーン・マップを作成および変更する文の中で同様に機能します。

PCTUSEDは、索引構成表には無効な表記憶特性です。

PCTFREEおよびPCTUSEDの合計は100以下である必要があります。PCTFREEとPCTUSEDをともに使用して、データベース・オブジェクト内の領域を効果的に利用できます。「PCTFREEとPCTUSEDの連携」を参照してください。

PCTUSED句の制限事項

PCTUSEDパラメータには、次の制限事項があります。

- このパラメータは、索引または索引構成表の索引セグメントには指定できません。
- このパラメータは、自動セグメント領域管理のオブジェクトに対しては有効ではなく、無視されます。

関連項目:

PCTUSEDおよびPCTFREEの各値によるパフォーマンスへの効果については『Oracle Databaseパフォーマンス・チューニング・ガイド』を参照してください。自動セグメント領域管理については「CREATE TABLESPACE」の「segment_management_clause」を参照してください。

PCTFREEとPCTUSEDの連携

新しく割り当てられたデータ・ブロックでは、挿入に使用できる領域は、ブロック・サイズからブロック・オーバーヘッドと空き領域(PCTFREE)の合計を引いたものになります。既存のデータを更新する場合は、ブロック内の任意の利用可能な領域を使用できます。このため、更新によってブロックの利用可能な領域がPCTFREEより小さくなる可能性があります。

データ・ブロックがPCTFREEによって決定された制限に達すると、そのブロックの割合(パーセント)がパラメータPCTUSEDを下回るまで、ブロックに新しい行は挿入はできないとOracle Databaseによって判断されます。この値になるまで、Oracle Databaseでは、データ・ブロックにすでに含まれている行の更新にのみデータ・ブロックの空き領域が使用されます。ブロックは、

使用済領域がPCTUSEDの値を下回ると、行挿入の対象となります。

関連項目:

PCTUSEDおよびPCTFREEによる空きリストのセグメント領域管理処理の詳細は、「FREELISTS」を参照してください。

INITRANS integer

データベース・オブジェクトに割り当てられた各データ・ブロックに割り当てられる、同時実行トランザクション・エントリの初期数を指定します。この値の範囲は1から255で、デフォルト値は1ですが、次の例外があります。

- クラスタのINITRANSのデフォルト値は、クラスタが存在する表領域のINITRANSのデフォルト値と2のいずれか大きい方の値です。
- 索引のデフォルト値は2です。

通常、INITRANS値は、変更せずにデフォルトのまま使用してください。

ブロックを更新するトランザクションごとに、ブロックのトランザクション・エントリが必要です。このパラメータを指定した場合、最小数の同時実行トランザクションでブロックを更新できます。さらに、トランザクション・エントリを動的に割り当てるときのオーバーヘッドを回避できます。

INITRANSパラメータは、表、パーティション、クラスタ、索引、マテリアライズド・ビューおよびマテリアライズド・ビュー・ログを作成および変更する文の中で同様に機能します。

MAXTRANSパラメータ

MAXTRANSは、セグメント内の各データ・ブロックで実行可能な同時実行更新トランザクションの最大数を決定する以前のリリースのパラメータです。このパラメータは現在非推奨になっています。今回のリリースでは、Oracleは、ブロック内の未使用領域に応じて、任意のデータ・ブロックに対して最大255の同時実行更新トランザクションを自動的に許可します。同時更新トランザクションの最大数は、ブロックのサイズに基づいていることに注意してください

MAXTRANS値が設定された既存のオブジェクトは、その設定を保持します。ただし、MAXTRANS値を変更しようとすると、新しい指定は無視され、エラーを戻さずに値は255に置き換えられます。

storage_clause

storage_clauseによって、表、オブジェクト表OIDINDEX、パーティション、LOBデータ・セグメントまたは索引構成表のオーバーフロー・データ・セグメントの記憶特性を指定できます。この句は、大規模な表のパフォーマンスに影響します。記憶域は、追加領域の動的割当てを最小限に抑えるように割り当てられます。詳細は、「storage_clause」を参照してください。

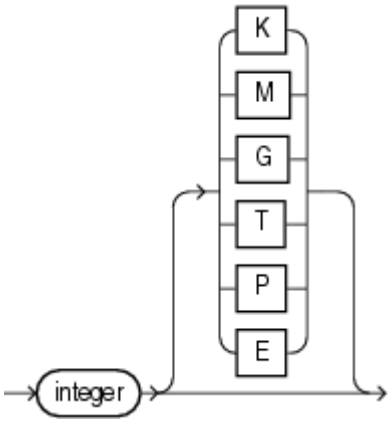
size_clause

目的

size_clauseを使用すると、すべての文にバイト、KB(K)、MB(M)、GB(G)、TB(T)、PB(P)、EB(E)の各単位でバイト数を指定して、ディスク容量またはメモリー容量を設定できます。

構文

size_clause::=



セマンティクス

size_clauseを使用すると、バイト数を様々な単位で指定できます。単位の略語を指定しない場合、integerはバイト単位で解釈されます。

ノート:

バイトのすべての単位が常に適切であるとはかぎりません。状況依存の制限が適用されます。後者の場合、Oracle はエラー・メッセージを発行します。

storage_clause

目的

storage_clauseを使用すると、Oracle Databaseによる永続的なデータベース・オブジェクトの格納方法を指定できます。一時セグメントの記憶域パラメータでは、常に、関連付けられた表領域のデフォルトの記憶域パラメータが使用されます。記憶域パラメータは、データベースのデータへのアクセス時間およびデータベース内での領域の効率的な利用に影響します。

関連項目:

記憶域パラメータの影響の詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

クラスタ、索引、マテリアライズド・ビュー、マテリアライズド・ビュー・ログ、ロールバック・セグメント、表、LOB、VARRAY、ネストした表またはパーティションを作成する場合、これらのオブジェクトに割り当てられるセグメントに、記憶域パラメータの値を指定できます。記憶域パラメータを指定しない場合、オブジェクトが存在する表領域に指定されている記憶域パラメータの値が使用されます。表領域に対して値を指定しなかった場合、データベースによってデフォルト値が使用されます。

ノート:

ローカル管理表領域内のオブジェクトへの記憶域パラメータの指定は、下位互換性のためにサポートされています。ローカル管理表領域を使用している場合、ローカル管理表領域にオブジェクトを作成する際にこれらの記憶域パラメータを省略できます。

クラスタ、索引、マテリアライズド・ビュー、マテリアライズド・ビュー・ログ、ロールバック・セグメント、表、VARRAY、ネストした表またはパーティションを変更する場合、記憶域パラメータの値を変更できます。この値の変更は、それ以降のエクステントの割当てにのみ影響します。

storage_clauseはphysical_attributes_clauseの一部であるため、この句は、物理属性句を指定できる任意の文に指定できます(「physical_attributes_clause」を参照)。storage_clauseは次の文にも指定できます。

- CREATE CLUSTERおよびALTER CLUSTER: クラスタおよびクラスタ内のすべての表の記憶特性を設定または変更する場合(「CREATE CLUSTER」および「ALTER CLUSTER」を参照)。
- CREATE INDEXおよびALTER INDEX: 表索引または索引パーティションに対して作成された索引セグメントまたは主キー制約または一意制約を適用するために使用される索引に対して作成された索引セグメントの記憶特性を設定または変更する場合(「CREATE INDEX」および「ALTER INDEX」を参照)。
- CREATE TABLEまたはALTER TABLEのENABLE ... USING INDEX句: 主キー制約または一意制約を適用するためにシステムによって作成された索引の記憶特性を設定または変更する場合
- CREATE MATERIALIZED VIEWおよびALTER MATERIALIZED VIEW: マテリアライズド・ビュー、そのパーティションの1つ、またはマテリアライズド・ビューを保持するために生成される索引の記憶特性を設定または変更する場合(「CREATE MATERIALIZED VIEW」および「ALTER MATERIALIZED VIEW」を参照)。
- CREATE MATERIALIZED VIEW LOGおよびALTER MATERIALIZED VIEW LOG: マテリアライズド・ビュー・ログの記憶特性を設定または変更する場合(「CREATE MATERIALIZED VIEW LOG」および「ALTER MATERIALIZED VIEW LOG」を参照)。
- CREATE ROLLBACK SEGMENTおよびALTER ROLLBACK SEGMENT: ロールバック・セグメントの記憶特性を

設定または変更する場合(「CREATE ROLLBACK SEGMENT」および「ALTER ROLLBACK SEGMENT」を参照)。

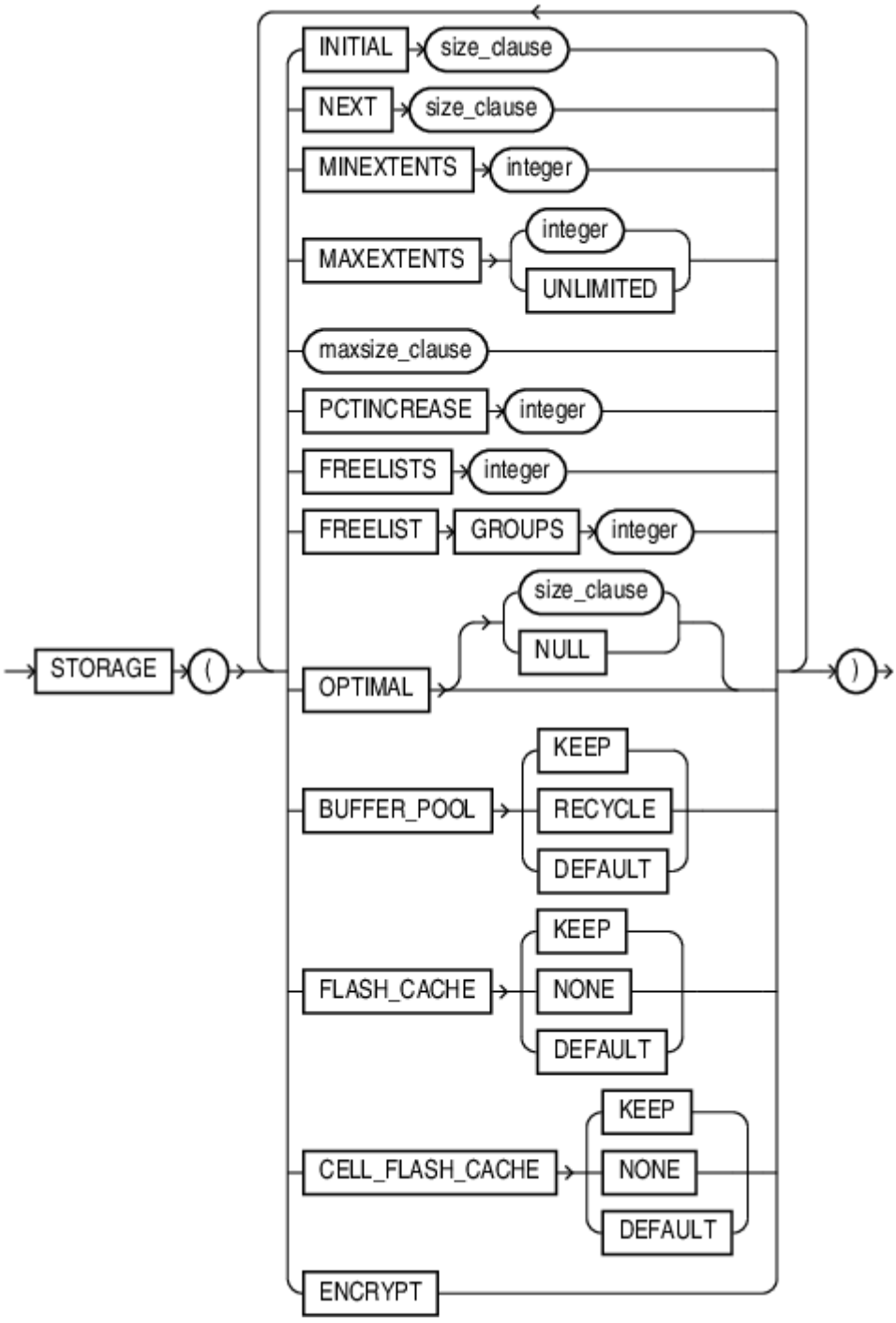
- CREATE TABLEおよびALTER TABLE: クラスタ化されていない表またはそのパーティションまたはサブパーティションの1つのLOBまたはVARRAYデータ・セグメント、またはネストした表の記憶表の記憶特性を設定する場合(「CREATE TABLE」および「ALTER TABLE」を参照)。
- CREATE TABLESPACEおよびALTER TABLESPACE: 表領域に作成されたすべてのオブジェクトに対するデフォルトの記憶特性を設定または変更する場合(「CREATE TABLESPACE」および「ALTER TABLESPACE」を参照)。表領域の記憶域パラメータに対する変更は、表領域に作成される新しいオブジェクトまたはセグメントに割り当てられる新しいエクステントにのみ影響します。
- `constraint`: 制約の適用に使用される索引(パーティション索引の場合は索引のパーティション)の記憶域を指定する場合(「constraint」を参照)。

前提条件

STORAGEパラメータの値を変更する場合は、適切なCREATE文またはALTER文を使用するための権限が必要です。

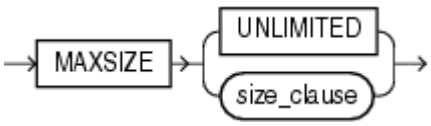
構文

storage_clause::=



(size_clause::=)

maxsize_clause::=



(size_clause::=)

セマンティクス

この項では、storage_clauseのパラメータについて説明します。詳細は、特定のデータベース・オブジェクトに対してこれらの記憶域パラメータを設定または再設定するSQL文の説明を参照してください。

ノート:

ローカル管理表領域の場合、storage_clause の解析方法は異なります。ローカル管理表領域の場合、Oracle Database では、オブジェクトの最初の作成時に割り当てられるエクステント数を計算するために、INITIAL、NEXT、PCTINCREASE および MINEXTENTS が使用されます。オブジェクトの作成後、これらのパラメータは無視されます。詳細は、「CREATE TABLESPACE」を参照してください。

関連項目:

表の記憶域属性の指定: 例

INITIAL

オブジェクトの第1エクステントのサイズを指定します。スキーマ・オブジェクトの作成時に、このエクステントに領域が割り当てられます。この句の詳細は、「size_clause」を参照してください。

ローカル管理表領域では、初期セグメント・サイズを決定するために、INITIALの値が、ローカル管理の種類(AUTOALLOCATEまたはUNIFORM)と、MINEXTENTS、NEXTおよびPCTINCREASEの値とともに使用されます。

- AUTOALLOCATEエクステント管理の場合、INITIAL設定を使用して、割り当てられるエクステント数を最適化します。エクステントには、64K、1M、8Mおよび64Mのサイズを割り当てることができます。セグメントの作成時に、これらの4つのサイズのうちINITIAL以下で最大のものがシステムによって選択され、そのサイズのエクステントがINITIAL設定に達するのに必要な数だけ割り当てられます。たとえば、INITIALを4Mに設定すると、1Mのエクステントが4つ作成されます。
- UNIFORMエクステント管理の場合、エクステント数は、初期セグメント・サイズと表領域作成時に指定した均一エクステント・サイズで決定されます。たとえば、エクステントが1Mの均一なローカル管理表領域の場合、INITIALの値を5Mに指定すると、1Mのエクステントが5つ作成されます。

  次の例を比較してみます。AUTOALLOCATEの場合、INITALを72Kに設定すると、初期セグメント・サイズは128K(INITIALを超えるサイズ)になります。64Kより小さいエクステントを割り当てることはできないため、必然的に64Kのエクステントが2つ割り当てられます。INITIALを72Kに設定し、24KのUNIFORMエクステント・サイズを使用する場合は、24Kのエクステント3つが割り当てられるため、72Kに等しくなります。

ディクショナリ管理表領域では、デフォルトの初期エクステント・サイズは5ブロックであり、後続のすべてのエクステントは5ブロックに丸められます。表領域作成時にMINIMUM EXTENTが指定された場合、エクステント・サイズはMINIMUM EXTENTの値に丸められます。

INITIALの制限事項

INITIALをALTER文で指定することはできません。

NEXT

オブジェクトに割り当てる次のエクステント・サイズをバイト単位で指定します。この句の詳細は、「size_clause」を参照してください。

ローカル管理表領域では、ユーザー指定のNEXTの値はすべて無視され、表領域が自動割当てエクステント管理用に設定されている場合、NEXTのサイズはOracleによって決定されます。UNIFORM表領域では、NEXTのサイズは表領域作成時に指定された均一エクステント・サイズです。

ディクショナリ管理表領域では、デフォルトのサイズは5データ・ブロックです。最小のサイズは1データ・ブロックです。最大値は、ご使用のオペレーティング・システムによって異なります。5データ・ブロックより小さい値が指定された場合、データ・ブロック・サイズの一番近い倍数に丸められます。5データ・ブロックを超える値の場合は、断片化を最小限に抑える値に切り上げられます。

関連項目:

断片化の最小化の詳細は、『Oracle Database概要』を参照してください。

PCTINCREASE

ローカル管理表領域では、初期セグメント・サイズを決定するためにセグメント作成時にPCTINCREASEの値が使用され、その後の領域割当て時にはこのパラメータは無視されます。

ディクショナリ管理表領域では、3番目以降の各エクステントが直前のエクステントに対して増加する割合(パーセント)を指定します。デフォルト値は50です。この場合、3番目以降のエクステントは、それぞれその直前のエクステントより50%ずつ大きくなります。最小値は0で、この場合、第2エクステント以降のエクステントのサイズはすべて同じになります。最大値は、ご使用のオペレーティング・システムによって異なります。計算された各新規エクステントのサイズは、データ・ブロック・サイズの一番近い倍数に丸められます。ALTER文にPCTINCREASEパラメータの値を指定してこの値を変更すると、この変更した値と直前に割り当てられたエクステントのサイズを使用して、次に割り当てるエクステントのサイズが計算されます。

PCTINCREASEの制限事項

PCTINCREASEをロールバック・セグメントに指定することはできません。ロールバック・セグメントでは、PCTINCREASEの値は常に0(ゼロ)です。

MINEXTENTS

ローカル管理表領域では、初期セグメント・サイズを決定するために、MINEXTENTSの値がPCTINCREASE、INITIALおよびNEXTの値とともに使用されます。

ディクショナリ管理表領域では、オブジェクトの作成時に割り当てられる合計エクステント数を指定します。最小値(デフォルト)は1です。この場合、第1エクステントのみが割り当てられます。ただし、ロールバック・セグメントの場合、最小値(デフォルト)は2です。最大値は、ご使用のオペレーティング・システムによって異なります。

- ローカル管理表領域では、領域の初期割当て量を計算するためにMINEXTENTSが使用されます。INITIAL * MINEXTENTSで算出されます。その後、この値は1に設定され、DBA_SEGMENTSビューに反映されます。
- ディクショナリ管理表領域では、MINEXTENTSは単純に、セグメントに割り当てる必要があるエクステントの最小数です。

MINEXTENTSの値が1より大きい場合、INITIAL、NEXTおよびPCTINCREASE記憶域パラメータの値に基づいて、次のエクステントのサイズが計算されます。

ALTER文にMINEXTENTSの値を指定して変更する際は、現行の値より小さくすることはできますが、大きくすることはできません。MINEXTENTSをより小さい値に再設定すると便利な場合があります。たとえば、TRUNCATE ... DROP STORAGE文の前で、TRUNCATE操作の後もセグメントがエクステントの最小数を変更しない場合です。

MINEXTENTSの制限事項

MINEXTENTS記憶域パラメータには、次の制限事項があります。

- MINEXTENTSは、表領域レベルでは適用できません。
- ALTER文のMINEXTENTSの値、またはローカル管理表領域に存在するオブジェクトに対するMINEXTENTSの値は

変更できません。

MAXEXTENTS

この記憶域パラメータは、ディクショナリ管理表領域のオブジェクトに対してのみ有効です。第1エクステントを含めて、Oracleがオブジェクトに割り当てることができるエクステントの総数を指定します。最小値は1です(最小値が常に2のロールバック・セグメントは除きます)。デフォルト値は、データ・ブロックのサイズによって異なります。

MAXEXTENTSの制限事項

MAXEXTENTSは、ローカル管理表領域に存在するオブジェクトに対しては無視されます。ただし、DBA_TABLESPACESデータ・ディクショナリ・ビューの表領域に対し、ALLOCATION_TYPEの値がUSERである場合を除きます。

関連項目:

DBA_TABLESPACESデータ・ディクショナリ・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

UNLIMITED

必要に応じてエクステントが自動的に割り当てられるようにする場合に、UNLIMITEDを指定します。断片化を最小限に抑えるため、この設定をお薦めします。

ロールバック・セグメントにこの句は使用しないでください。この句をロールバック・セグメントに使用すると、長時間実行される特殊なDMLトランザクションによって、ディスクが一杯になるまで新しいエクステントが作成され続ける可能性があります。

ノート:

storage_clause を指定せずに作成したロールバック・セグメントは、ロールバック・セグメントを作成した表領域の記憶域パラメータと同じ設定になります。MAXEXTENTS UNLIMITED を指定して表領域を作成する場合、ロールバック・セグメントのデフォルトは同じ設定になります。

MAXSIZE

MAXSIZE句を使用すると、記憶域要素の最大サイズを指定できます。LOB記憶域では、MAXSIZEを使用すると、次のようになります。

- LOB_parametersにRETENTION MAXを指定すると、指定されたサイズにLOBセグメントが拡張され、その後、UNDO領域から任意の領域を再利用できます。
- LOB_parametersにRETENTION AUTO、MINまたはNONEを指定すると、指定されたサイズはLOBセグメントのサイズの上限値となり、UNDOの保持においてその指定の影響はありません。

UNLIMITED

記憶域要素のディスク領域を制限しない場合は、UNLIMITED句を使用します。この句は、LOB_parametersでのRETENTION MAXの指定と同時には指定できません。両方を指定すると、RETENTION AUTOおよびMAXSIZE UNLIMITEDが使用されます。

FREELISTS

セグメントを手動で領域管理する表領域の場合、セグメント内の挿入ポイント数を増やしてOLTPシステムの領域管理のパフォーマンスを向上させるために、FREELISTS記憶域パラメータが使用されます。セグメントを自動的に領域管理する表領域

の場合、変化するワークロードにデータベースが適応するため、このパラメータは無視されます。

セグメントを手動で領域管理する表領域の場合、表領域およびロールバック・セグメント以外のオブジェクトに対して、表、パーティション、クラスタまたは索引の各空きリスト・グループの空きリスト数を指定します。このパラメータの最小値(デフォルト)は1です。この場合、空きリスト・グループごとに1つの空きリストが含まれます。最大値は、データ・ブロックのサイズによって異なります。FREELISTSに指定した値が大きすぎた場合、最大値を示すエラーが戻ります。

「LOB_parameters」のSECUREFILEパラメータを指定している場合、この句は無効です。SECUREFILEパラメータおよびFREELISTSの両方を指定した場合、FREELISTSの指定は無視されます。

FREELISTSの制限事項

表領域またはロールバック・セグメントを作成または変更するとき以外は、すべての文のstorage_clauseにFREELISTSを指定できます。

FREELIST GROUPS

セグメントを手動で領域管理する表領域の場合、Oracle Real Application Clusters環境でセグメント空き領域を静的にパーティション化するために、この記憶域パラメータの値が使用されます。このパーティション化によって、セグメント・メタデータがインスタンス間で転送されなくなり、領域の割当ておよび割当て解除のパフォーマンスが向上します。セグメントを自動的に領域管理する表領域の場合、インスタンス間ワークロードにOracleが動的に適応するため、このパラメータは無視されます。

セグメントを手動で領域管理する表領域の場合、作成するデータベース・オブジェクトに対する空きリスト・グループ数を指定します。このパラメータの最小値(デフォルト)は1です。Oracleは、Oracle Real Application Clusters(Oracle RAC)インスタンスのインスタンス番号を使用して、各インスタンスを空きリスト・グループにマップします。

1つの空きリスト・グループに、それぞれデータベース・ブロックを1つずつ使用します。そのため、次のようになります。

- 各空きリスト・グループの最小値に、1データ・ブロックを加えた値を格納できるだけの十分な大きさの値をINITIALの値に指定していない場合、INITIALの値は必要な分だけ引き上げられます。
- 均一なローカル管理表領域にオブジェクトを作成する場合、空きリスト・グループ数に適応するだけの十分なエクステント・サイズがないと、作成操作は正常に実行されません。

「LOB_parameters」のSECUREFILEパラメータを指定している場合、この句は無効です。SECUREFILEパラメータおよびFREELIST GROUPSの両方を指定した場合、FREELIST GROUPSの指定は無視されます。

FREELIST GROUPSの制限事項

FREELIST GROUPSパラメータは、CREATE TABLE、CREATE CLUSTER、CREATE MATERIALIZED VIEW、CREATE MATERIALIZED VIEW LOGおよびCREATE INDEX文でのみ指定できます。

OPTIMAL

OPTIMALキーワードは、ロールバック・セグメントのみに指定します。ロールバック・セグメントの最適なサイズをバイト単位で指定します。この句の詳細は、「size_clause」を参照してください。

エクステントのデータがアクティブ・トランザクションで不要になった場合、Oracleは、そのエクステントの割当てを動的に解除することによって、指定されたロールバック・セグメントのサイズを維持します。ロールバック・セグメントのサイズの合計をOPTIMAL値より小さくせずに、できるだけ多くのエクステントの割当てを解除します。

OPTIMALには、MINEXTENTS、INITIAL、NEXTおよびPCTINCREASEパラメータで最初に割り当てた領域より小さい値を指定できません。最大値は、ご使用のオペレーティング・システムによって異なります。値は、データ・ブロック・サイズの一番近い倍数に丸められます。

NULL

ロールバック・セグメントに対する最適なサイズがないことを示す場合にNULLを指定します。これは、ロールバック・セグメントのエクステントの割当てが解除されないことを示します。これはデフォルトの動作です。

BUFFER_POOL

BUFFER_POOL句では、スキーマ・オブジェクト用のデフォルトのバッファ・プール(キャッシュ)を定義します。オブジェクトのすべてのブロックは、指定されたキャッシュに格納されます。

- バッファ・プールがパーティション表またはパーティション索引用に定義されている場合は、パーティションは、パーティション・レベル定義で変更されないかぎり、表定義または索引定義のバッファ・プールを継承します。
- 索引構成表の場合、索引セグメントおよびオーバーフロー・セグメントに対して個々にバッファ・プールを指定できます。

BUFFER_POOLパラメータの制限事項

BUFFER_POOLには、次の制限事項があります。

- この句は、クラスタ化表には指定できません。ただし、クラスタには指定できます。
- この句は、表領域またはロールバック・セグメントには指定できません。

KEEP

KEEPを指定すると、ブロックがセグメントからKEEPバッファ・プールへ移されます。KEEPバッファ・プールに適切なサイズを維持すると、メモリーのスキーマ・オブジェクトが保持され、I/O操作を避けることができます。KEEPは、表、クラスタ、マテリアライズド・ビューまたはマテリアライズド・ビュー・ログに指定するNOCACHE句より優先されます。

RECYCLE

RECYCLEを指定すると、ブロックがセグメントからRECYCLEプールへ移されます。RECYCLEプールに適切なサイズを指定すると、不要なキャッシュ領域が利用されず、デフォルト・プールがRECYCLEプールであるオブジェクト数が削減されます。

DEFAULT

デフォルトのバッファ・プールを識別する場合に、DEFAULTを指定します。これは、KEEPもRECYCLEも指定しないオブジェクトのデフォルトです。

関連項目:

複数のバッファ・プールの使用方法については、『Oracle Databaseパフォーマンス・チューニング・ガイド』を参照してください。

FLASH_CACHE

FLASH_CACHE句を使用すると、自動バッファ・キャッシュ・ポリシーを上書きし、特定のスキーマ・オブジェクトをフラッシュ・メモリーにキャッシュする方法を指定できます。この句を使用するには、システムでデータベース・スマート・フラッシュ・キャッシュ(フラッシュ・キャッシュ)が構成されている必要があります。フラッシュ・キャッシュは、フラッシュ・ディスク(フラッシュ・メモリーを使用するストレージ・デバイス)に格納されるデータベース・バッファ・キャッシュの拡張機能です。フラッシュ・メモリーは磁気ディスクよりも高速であるため、データベースは、磁気ディスクから読み込むかわりに、フラッシュ・キャッシュにバッファをキャッシュすることによってパフォーマンスを向上させることができます。

KEEP

フラッシュ・キャッシュの大きさが十分にあり、スキーマ・オブジェクト・バッファを引き続きフラッシュ・キャッシュにキャッシュする場合は、KEEPを指定します。

NONE

スキーマ・オブジェクト・バッファをフラッシュ・キャッシュにキャッシュしないようにする場合は、NONEを指定します。これにより、より頻繁にアクセスするオブジェクト用にフラッシュ・キャッシュ領域を確保できます。

DEFAULT

メイン・メモリーからエージ・アウトされるときにスキーマ・オブジェクト・バッファをフラッシュ・キャッシュに書き込み、その後、標準のバッファ・キャッシュ置換アルゴリズムによりフラッシュ・キャッシュからエージ・アウトする場合は、DEFAULTを指定します。フラッシュ・キャッシュが構成されている場合に、KEEPまたはNONEを指定しないと、これがデフォルトになります。

ノート:

データベース・スマート・フラッシュ・キャッシュは、Solaris および Oracle Linux 環境でのみ使用できます。

関連項目:

- データベース・スマート・フラッシュ・キャッシュの詳細は、『Oracle Database概要』を参照してください。
- データベース・スマート・フラッシュ・キャッシュの構成方法を学習するには、『Oracle Database管理者ガイド』を参照してください。

ENCRYPT

この句は、表領域を作成する場合にのみ有効です。表領域全体を暗号化するには、ENCRYPTを指定します。CREATE TABLESPACE文にENCRYPTION句を指定する必要もあります。

ノート:

ENCRYPT 句は、下位互換性を保つためにサポートされています。ただし、Oracle Database 12c リリース 2 (12.2)以降は、かわりに tablespace_encryption_clause で ENCRYPT を指定できます。詳細は、「CREATE TABLESPACE」の「tablespace_encryption_clause」を参照してください。

例

表の記憶域属性の指定: 例

次の文は、表を作成し、記憶域パラメータ値を指定します。

```
CREATE TABLE divisions
    (div_no     NUMBER(2),
     div_name   VARCHAR2(14),
     location   VARCHAR2(13) )
     STORAGE  ( INITIAL 8M MAXSIZE 1G );
```

次の文は、表に対して第1エクステントのサイズを問い合せます。

```
SELECT INITIAL_EXTENT FROM USER_TABLES WHERE TABLE_NAME='DIVISIONS';
INITIAL_EXTENT
--------------
       8388608
```

STORAGEパラメータに指定した値に基づいて、次に示すとおり表に領域が割り当てられます。

- INITIALの値は8Mのため、第1エクステントのサイズは8MBになります。
- MAXSIZEの値は1Gのため、記憶域要素の最大サイズは1GBになります。

9 SQL問合せおよび副問合せ

この章では、SQL問合せおよび副問合せについて説明します。

この章では、次の内容を説明します。

- 問合せおよび副問合せ
- 単純な問合せの作成
- 階層問合せ
- UNION [ALL]、INTERSECTおよびMINUS演算子
- 問合せ結果のソート
- 結合
- 副問合せの使用方法
- ネストされた副問合せのネスト解除
- DUAL表からの選択
- 分散問合せ

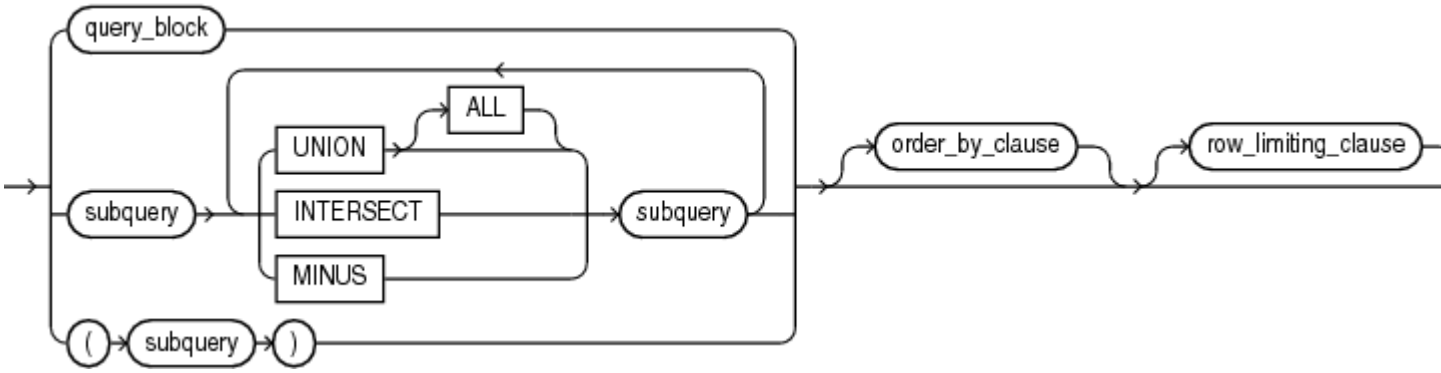
問合せおよび副問合せ

問合せとは、1つ以上の表またはビューからデータを検索する操作のことです。このマニュアルでは、トップレベルのSELECT文を問合せといい、他のSQL文の中でネストされた問合せを副問合せといいます。

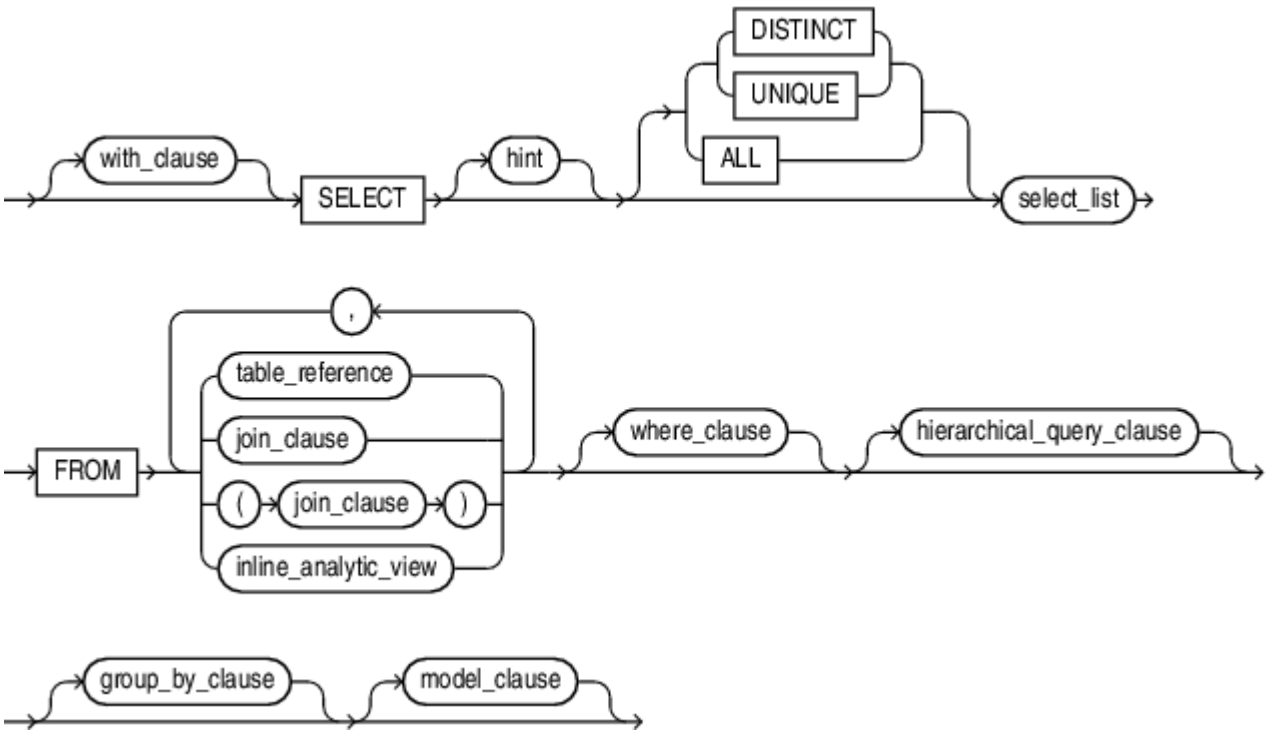
この項では、問合せおよび副問合せの種類およびその使用方法について説明します。この章では、トップレベルの構文について説明します。すべての句のすべての構文およびこの文のセマンティクスについては、「SELECT」を参照してください。

select::=

subquery::=



query_block::=



単純な問合せの作成

SELECTキーワードの後、FROM句の前にある式のリストを、SELECT構文のリストといいます。SELECT構文のリストに、1つ以上の表、ビューおよびマテリアライズド・ビューからOracle Databaseが戻す行に含まれる1つ以上の列を指定します。SELECT構文のリストの要素によって、列のデータ型、長さおよび数が決定されます。

複数の表に同じ名前の列がある場合、表の名前でその列名を修飾する必要があります。それ以外の場合は、完全に修飾した列名はオプションとなります。ただし、明示的に表および列の参照を修飾することをお薦めします。表および列名を完全に修飾することで、Oracleの作業が少なくなります。

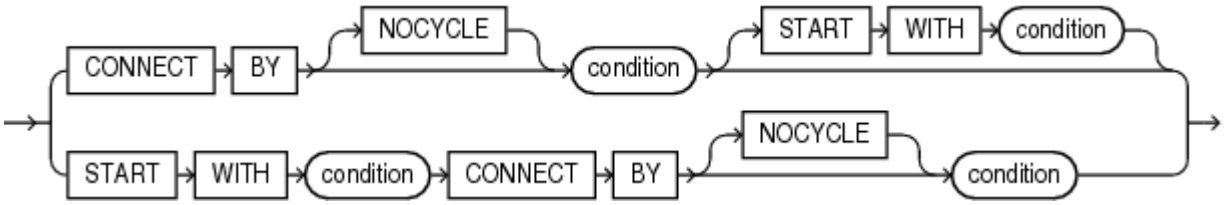
列の別名c_aliasを使用して、SELECT構文のリストの直前の式にラベルを付けると、列が新しい見出し付きで表示されます。別名によって、問合せ中にSELECT構文のリストの項目名を効果的に変更できます。別名はORDER BY句の中で使用できますが、問合せ内のその他の句には使用できません。

Oracle Databaseのオプティマイザに指示(ヒント)を与えるために、SELECT文中でコメントを使用できます。オプティマイザは、これらのヒントを使用して文の実行計画を選択します。ヒントの詳細は、「ヒント」を参照してください。

階層問合せ

表に階層データが含まれる場合、階層問合せ句を使用して階層順に行を選択することができます。

hierarchical_query_clause::=



conditionは、「条件」で説明されている任意の条件にすることができます。

START WITH句では、階層のルート行を指定します。

CONNECT BY句では、階層の親/子の行の関連を指定します。

- NOCYCLEパラメータは、データ内にCONNECT BYループが存在する場合でも問合せで行を戻すようにOracle Databaseに指示します。このパラメータをCONNECT_BY_ISCYCLE疑似列とともに使用すると、ループが含まれている行を確認できます。詳細は、「CONNECT_BY_ISCYCLE疑似列」を参照してください。
- 階層問合せでは、condition内の1つの式を、親である行を参照するためにPRIOR演算子で修飾する必要があります。たとえば、次のようになります。

  ```
  ... PRIOR expr = expr
  or
  ... expr = PRIOR expr
  ```

  CONNECT BY conditionが複合条件の場合、1つの条件のみにPRIOR演算子が必要です(複数のPRIOR条件を使用することもできます)。たとえば:

  ```
  CONNECT BY last_name != 'King' AND PRIOR employee_id = manager_id ...
  CONNECT BY PRIOR employee_id = manager_id and
             PRIOR account_mgr_id = customer_id ...
  ```

  PRIORは単項演算子であり、単項算術演算子の+および-と同じ優先順位を持っています。この演算子は、階層問合せ内でカレント行の親である行の直後にある式を評価します。

  PRIORは、等価演算子を使用して列の値を比較する場合によく使用されます。(PRIORキーワードは演算子のどちら側でもかまいません。)PRIORを指定すると、列の親である行の値が使用されます。等号(=)以外の演算子は、理論上はCONNECT BY句に指定できます。ただし、これらの他の演算子の組合せによっては、作成される条件は無限ループを発生させる場合があります。この場合、実行時にループが検出され、エラーが戻されます。

CONNECT BY条件とPRIOR式は、いずれも相関関係のない副問合せの形式で指定できます。ただし、CURRVALおよびNEXTVALは、無効なPRIOR式であるため、PRIOR式は順序を参照できません。

CONNECT_BY_ROOT演算子を使用してSELECT構文のリスト内の列を問い合せることによって、階層問合せをさらに向上できます。この演算子は、親の行のみでなく、階層内のすべての祖先の行を戻すことによって、階層問合せのCONNECT BY [PRIOR]条件の機能を拡張します。

関連項目:

この演算子の詳細は、「CONNECT_BY_ROOT」および「階層問合せの例」を参照してください。

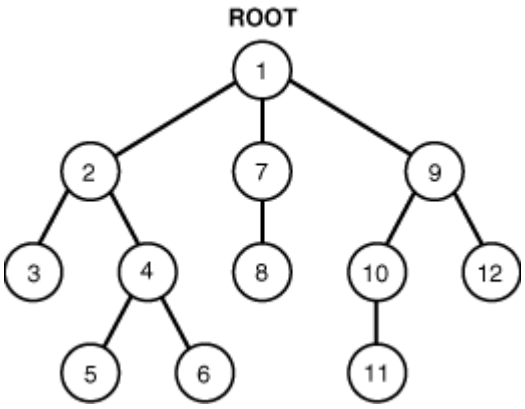
Oracleは次のように階層問合せを処理します。

- 最初に、結合(指定されている場合)が、FROM句で指定されているか、またはWHERE句述語で指定されているかが評価されます。
- CONNECT BY条件が評価されます。
- 残りのWHERE句述語が評価されます。

次に、Oracleはこれらの評価からの情報を使用して、次のステップで階層を形成します。

- Oracleは、階層のルート行を選択します。これらの行は、START WITH条件を満たすものです。
- Oracleは、各ルート行の子である行を選択します。子である各行は、1つのルート行に関してCONNECT BY条件を満たす必要があります。
- Oracleは、子である行の連続生成を選択します。まず、ステップ2で戻された子である行を選択し、その行にある子を選択します(以降同様に続きます)。現在の親である行に関するCONNECT BY条件を評価することによって、常に子を選択します。
- 問合せに結合を含まないWHERE句が含まれる場合、Oracleは、階層からWHERE句の条件を満たさないすべての行を排除します。条件を満たさない子である行をすべて排除するのではなく、各行に関してこの条件をそれぞれ評価します。
- Oracleは、図9-1に示す順序で行を戻します。この図では、親である行の下に子である行が表示されます。階層ツリーの詳細は、図3-1を参照してください。

図9-1 階層問合せ



親である行に対する子を検索するために、Oracleは、親である行のCONNECT BY条件のPRIOR式、および各行の他の式を表の中で評価します。条件がTRUEとなる行が、その親である行の子です。CONNECT BY条件に、問合せによって選択された行をさらにフィルタ処理するための他の条件を含めることができます。

CONNECT BY条件が階層のループになった場合、Oracleはエラーを戻します。1つの行が別の行の親(または親の親または祖先)および子(または子の子または子孫)の場合、ループが発生します。

ノート:

> 階層問合せでは、ORDER BY または GROUP BY を指定しないでください。指定すると、CONNECT BY の結果の階層順序が上書きされます。同じ親の兄弟である行を順序付ける場合は、ORDER SIBLINGS BY 句を使用します。「order_by_clause」を参照してください。

階層問合せの例

CONNECT BYの例

次の階層問合せは、CONNECT BY句を使用して従業員とマネージャの関係を定義しています。

```
SELECT employee_id, last_name, manager_id
   FROM employees
   CONNECT BY PRIOR employee_id = manager_id;
EMPLOYEE_ID LAST_NAME                 MANAGER_ID
----------- ------------------------- ----------
        101 Kochhar                          100
        108 Greenberg                        101
        109 Faviet                           108
        110 Chen                             108
        111 Sciarra                          108
        112 Urman                            108
        113 Popp                             108
        200 Whalen                           101
        203 Mavris                           101
        204 Baer                             101
. . .
```

LEVELの例

次の例は、前述の例と似ていますが、LEVEL疑似列を使用して、親および子である行を表示しています。

```
SELECT employee_id, last_name, manager_id, LEVEL
   FROM employees
   CONNECT BY PRIOR employee_id = manager_id;
EMPLOYEE_ID LAST_NAME                 MANAGER_ID      LEVEL
----------- ------------------------- ---------- ----------
        101 Kochhar                          100          1
        108 Greenberg                        101          2
        109 Faviet                           108          3
        110 Chen                             108          3
        111 Sciarra                          108          3
        112 Urman                            108          3
        113 Popp                             108          3
        200 Whalen                           101          2
        203 Mavris                           101          2
        204 Baer                             101          2
        205 Higgins                          101          2
        206 Gietz                            205          3
        102 De Haan                          100          1
...
```

START WITHの例

次の例は、START WITH句を追加して階層にルート行を指定し、SIBLINGSキーワードを使用したORDER BY句を追加して階層の順序を保持しています。

```
SELECT last_name, employee_id, manager_id, LEVEL
      FROM employees
      START WITH employee_id = 100
      CONNECT BY PRIOR employee_id = manager_id
      ORDER SIBLINGS BY last_name;
LAST_NAME                 EMPLOYEE_ID MANAGER_ID      LEVEL
```

```
------------------------ ---------- ---------- ----------
King                            100                     1
Cambrault                       148        100          2
Bates                           172        148          3
Bloom                           169        148          3
Fox                             170        148          3
Kumar                           173        148          3
Ozer                            168        148          3
Smith                           171        148          3
De Haan                         102        100          2
Hunold                          103        102          3
Austin                          105        103          4
Ernst                           104        103          4
Lorentz                         107        103          4
Pataballa                       106        103          4
Errazuriz                       147        100          2
Ande                            166        147          3
Banda                           167        147          3
...
```

hr.employees表で、Steven Kingは会社の最高責任者であるため、マネージャはいません。彼の従業員には、部門80のマネージャであるJohn Russellがいます。employees表を更新してRussellをKingのマネージャとして設定する場合は、データ内にループを作成します。

```
UPDATE employees SET manager_id = 145
   WHERE employee_id = 100;
SELECT last_name "Employee",
   LEVEL, SYS_CONNECT_BY_PATH(last_name, '/') "Path"
   FROM employees
   WHERE level <= 3 AND department_id = 80
   START WITH last_name = 'King'
   CONNECT BY PRIOR employee_id = manager_id AND LEVEL <= 4;
ERROR:
ORA-01436: CONNECT BY loop in user data
```

CONNECT BY条件にNOCYCLEパラメータを指定すると、Oracleはループでも行を戻します。CONNECT_BY_ISCYCLE疑似列には、サイクルを含む行が表示されます。

```
SELECT last_name "Employee", CONNECT_BY_ISCYCLE "Cycle",
   LEVEL, SYS_CONNECT_BY_PATH(last_name, '/') "Path"
   FROM employees
   WHERE level <= 3 AND department_id = 80
   START WITH last_name = 'King'
   CONNECT BY NOCYCLE PRIOR employee_id = manager_id AND LEVEL <= 4
   ORDER BY "Employee", "Cycle", LEVEL, "Path";
Employee                  Cycle      LEVEL Path
------------------------ ---------- ---------- -------------------------
Abel                              0          3 /King/Zlotkey/Abel
Ande                              0          3 /King/Errazuriz/Ande
Banda                             0          3 /King/Errazuriz/Banda
Bates                             0          3 /King/Cambrault/Bates
Bernstein                         0          3 /King/Russell/Bernstein
Bloom                             0          3 /King/Cambrault/Bloom
Cambrault                         0          2 /King/Cambrault
Cambrault                         0          3 /King/Russell/Cambrault
Doran                             0          3 /King/Partners/Doran
Errazuriz                         0          2 /King/Errazuriz
Fox                               0          3 /King/Cambrault/Fox
...
```

CONNECT_BY_ISLEAFの例

次の文は、階層問合せを使用して、列の値をカンマで区切られたリストにする方法を示しています。

```
SELECT LTRIM(SYS_CONNECT_BY_PATH (warehouse_id,','),',') FROM
   (SELECT ROWNUM r, warehouse_id FROM warehouses)
   WHERE CONNECT_BY_ISLEAF = 1
   START WITH r = 1
   CONNECT BY r = PRIOR r + 1
   ORDER BY warehouse_id;

LTRIM(SYS_CONNECT_BY_PATH(WAREHOUSE_ID,','),',')
--------------------------------------------------------------------------------
1,2,3,4,5,6,7,8,9
```

CONNECT_BY_ROOTの例

次の例では、部門110の各従業員の名字、階層内で各従業員の上に位置する最高レベルのマネージャ、マネージャと従業員との間のレベル数、および両者間のパスを戻します。

```
SELECT last_name "Employee", CONNECT_BY_ROOT last_name "Manager",
   LEVEL-1 "Pathlen", SYS_CONNECT_BY_PATH(last_name, '/') "Path"
   FROM employees
   WHERE LEVEL > 1 and department_id = 110
   CONNECT BY PRIOR employee_id = manager_id
   ORDER BY "Employee", "Manager", "Pathlen", "Path";
Employee        Manager            Pathlen Path
--------------- --------------- ---------- ------------------------------
Gietz           Higgins                  1 /Higgins/Gietz
Gietz           King                     3 /King/Kochhar/Higgins/Gietz
Gietz           Kochhar                  2 /Kochhar/Higgins/Gietz
Higgins         King                     2 /King/Kochhar/Higgins
Higgins         Kochhar                  1 /Kochhar/Higgins
```

次の例では、GROUP BY句を使用して、部門110の各従業員と階層内でその従業員の上位に位置するすべての従業員の合計の給与を戻します。

```
SELECT name, SUM(salary) "Total_Salary" FROM (
   SELECT CONNECT_BY_ROOT last_name as name, Salary
      FROM employees
      WHERE department_id = 110
      CONNECT BY PRIOR employee_id = manager_id)
      GROUP BY name
   ORDER BY name, "Total_Salary";
NAME                      Total_Salary
------------------------- ------------
Gietz                             8300
Higgins                          20300
King                             20300
Kochhar                          20300
```

関連項目:

- 階層問合せでのこれらの疑似列の処理方法については、「LEVEL疑似列」および「CONNECT_BY_ISCYCLE疑似列」を参照してください。
- ルートからノードへの列の値のパスの検索については、「SYS_CONNECT_BY_PATH」を参照してください。
- ORDER BY句のSIBLINGSキーワードについては、「order_by_clause」を参照してください。
- subquery_factoring_clauseでは、再帰的副問合せのファクタリング(再帰的WITH)がサポートされ、これによって、階層データの問合せを行うことができます。この機能は、深さ優先検索および幅優先検索を提供し、複数の再帰的ブランチをサポートするという点において、CONNECT BYよりも強力です。

UNION [ALL]、INTERSECTおよびMINUS演算子

集合演算子UNION、UNION ALL、INTERSECTおよびMINUSを使用して、複数の問合せを組み合せることができます。集合演算子の優先順位はすべて同じです。SQL文に複数の集合演算子がある場合、カッコによって明示的に別の順序が指定されないかぎり、Oracle Databaseは左から右の順に評価します。

複合問合せを構成する各問合せと、それに対応するSELECT構文のリスト内の各式は、数値が一致し、データ型グループ(数値や文字など)が同じである必要があります。

集合演算子によって結合された2つの問合せが文字データを選択する場合、戻される値のデータ型は次のようにして決定されます。

- 両方の問合せが同じ長さのCHARデータ型の値を選択する場合、戻される値のデータ型はその長さのCHARになります。両方の問合せが異なる長さのCHARデータ型の値を選択する場合、戻される値は、長い方のCHAR値の長さを使用したVARCHAR2になります。
- 問合せのどちらか一方または両方が、VARCHAR2データ型の値を選択する場合、戻される値のデータ型はVARCHAR2になります。

集合演算子によって結合された2つの問合せが数値データを選択する場合、戻される値のデータ型は数値の優先順位によって決定されます。

- すべての問合せがBINARY_DOUBLE型の値を選択する場合、戻される値のデータ型はBINARY_DOUBLEになります。
- いずれの問合せもBINARY_DOUBLE型の値を選択せず、BINARY_FLOAT型の値を選択する場合、戻される値のデータ型はBINARY_FLOATになります。
- すべての問合せがNUMBER型の値を選択する場合、戻される値のデータ型はNUMBERになります。

集合演算子を使用する問合せでは、データ型グループ間の暗黙的な変換は行われません。そのため、複合問合せの対応する式が文字データと数値データの両方になる場合は、エラーが戻されます。

関連項目:

暗黙的な変換の詳細は、表2-8を参照してください。数値の優先順位の詳細は、「数値の優先順位の詳細」を参照してください。

例

次の問合せは有効です。

```
SELECT 3 FROM DUAL
   INTERSECT
SELECT 3f FROM DUAL;
```

この問合せは、次の複合問合せに暗黙的に変換されます。

```
SELECT TO_BINARY_FLOAT(3) FROM DUAL
   INTERSECT
SELECT 3f FROM DUAL;
```

次の問合せはエラーを戻します。

```
SELECT '3' FROM DUAL
```

```
   INTERSECT
SELECT 3f FROM DUAL;
```

集合演算子の制限事項

集合演算子には、次の制限事項があります。

- 集合演算子は、データ型がBLOB、CLOB、BFILE、VARRAYまたはネストした表である列に対しては無効になります。
- UNION、INTERSECTおよびMINUS演算子は、LONG列に対しては無効になります。
- 集合演算子の前のSELECT構文のリストに式が含まれている場合、order_by_clauseでその式を参照するには、式に列の別名を指定する必要があります。
- for_update_clauseは、集合演算子とともに指定できません。
- これらの演算子の副問合せには、order_by_clauseを指定できません。
- TABLEコレクション式を含むSELECT文では、これらの演算子を使用できません。

ノート:

SQL 規格に準拠するために、Oracle の今後のリリースでは、他の集合演算子より優先順位の高い INTERSECT 演算子が提供されます。したがって、INTERSECT 演算子と他の集合演算子を使用する問合せでは、カッコを使用して評価順序を指定してください。

UNIONの例

次の文は、UNION演算子によって2つの問合せの結果を結合しています。結果に重複行は含まれません。次の文は、他の表に存在していない列がある場合に、(TO_CHARファンクションを使用して)データ型を一致させる必要があることを示しています。

```
SELECT location_id, department_name "Department",
   TO_CHAR(NULL) "Warehouse"  FROM departments
   UNION
   SELECT location_id, TO_CHAR(NULL) "Department", warehouse_name
   FROM warehouses;
LOCATION_ID Department                     Warehouse
----------- ------------------------------ --------------------------
       1400 IT
       1400                                Southlake, Texas
       1500 Shipping
       1500                                San Francisco
       1600                                New Jersey
       1700 Accounting
       1700 Administration
       1700 Benefits
       1700 Construction
       1700 Contracting
       1700 Control And Credit
...
```

UNION ALLの例

UNION ALL演算子がすべての行を戻すのに対して、UNION演算子は重複しない行のみを戻します。UNION ALL演算子は、重複行も対象に含めます。

```
SELECT product_id FROM order_items
UNION
```

```
SELECT product_id FROM inventories
ORDER BY product_id;
SELECT location_id  FROM locations
UNION ALL
SELECT location_id  FROM departments
ORDER BY location_id;
```

問合せで複数回戻されるlocation_id値(1700など)は、UNION演算子では1回のみ戻されますが、UNION ALL演算子では複数回戻されています。

INTERSECTの例

次の文は、INTERSECT演算子によって2つの結果を結合しています。この場合、両方の問合せによって共通に戻される一意の行のみが戻されます。

```
SELECT product_id FROM inventories
INTERSECT
SELECT product_id FROM order_items
ORDER BY product_id;
```

MINUSの例

次の文は、MINUS演算子を使用して2つの結果を結合します。この場合、最初の問合せでは戻されるが、2番目の問合せでは戻されない一意の行のみが戻されます。

```
SELECT product_id FROM inventories
MINUS
SELECT product_id FROM order_items
ORDER BY product_id;
```

問合せ結果のソート

問合せで選択された行を並べ替えるには、ORDER BY句を使用します。位置のソートは次のような場合に有効です。

- selectリストの式が長い場合に並べ替えを行うには、式全体を繰り返すかわりにORDER BY句で位置を指定できます。
- 集合演算子UNION、INTERSECT、MINUSまたはUNION ALLを含む複合問合せでは、ORDER BY句に明示的な式ではなく、位置または別名を指定する必要があります。また、ORDER BY句は最後のコンポーネント問合せでのみ使用できます。ORDER BY句は、複合問合せ全体で返されたすべての行の並べ替えを行います。

Oracle DatabaseがORDER BY句の文字値をソートする順序付け方法は、照合とも呼ばれ、照合導出ルールを使用してORDER BY句式ごとに個別に決定されます。

式に対して決定されている照合が照合BINARYでない場合、文字値は言語的に比較されます。この場合、最初に照合キーに変換されてから、RAW値のように比較されます。照合キーは、SQLファンクションNLSSORTで使用されるものと同じメソッドを使用して暗黙的に生成されます。生成される照合キーには、「NLSSORT」で説明されているものと同じ制限事項があります。これらの制限の結果、初期化パラメータMAX_STRING_SIZEがSTANDARDに設定されている場合、照合キーの生成に使用された接頭辞が異なっていなければ、残りの値が異なる場合でも、2つの値は言語的に等しいと比較されることがあります。パラメータの値がEXTENDEDの場合、特定の状況下でエラー"ORA-12742: unable to create the collation key"が報告されることがあります。制限事項の詳細は、次のリンクを参照してください。

関連項目:

- 照合導出
- 言語ソートと照合
- SQL関数のNLSパラメータのデフォルト値
- NLSSORT

結合

結合は2つ以上の表、ビューまたはマテリアライズド・ビューからの行を組み合せる問合せです。複数の表が問合せのFROM句に指定される場合、Oracle Databaseは結合を実行します。問合せのSELECT構文のリストは、これらの表のいずれかの任意の列を選択することができます。これらの表のいずれか2つに共通の列名を持つものがある場合、問合せの間、これらの列に対してすべての参照を明確にするために表の名前を付けて修飾する必要があります。

結合条件

ほとんどの結合問合せには、FROM句またはWHERE句のいずれかに1つ以上の結合条件が含まれます。結合条件によって、異なる表から2つの列が比較されます。結合を実行するために、Oracle Databaseは各表に1つずつ含まれている列を結合し、結合条件がTRUEになるようにします。結合条件の列をSELECT構文のリストに表示する必要はありません。

3つ以上の表を結合するために、Oracleはまず列を比較する結合条件に基づいて2つの表を結合し、結合された表と新規の表の列を含む結合条件に基づいて、さらにもう1つの表を結合します。すべての表が結果に結合されるまで、このプロセスを継続します。オプティマイザは、Oracleが結合条件に基づいて表を結合する順序、表の索引、および任意の使用可能な表の統計を決定します。

結合条件を含むWHERE句には、1つの表のみの列を参照する別の条件も含めることができます。これらの条件は、結合問合せによって戻された列をさらに制限することができます。

> ノート:
>
> WHERE 句に結合条件が含まれる場合、WHERE 句には LOB 列を指定できません。WHERE 句での LOB の使用については、他にも制限事項があります。詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』を参照してください。

等価結合

等価結合とは、等価演算子を含む結合条件での結合のことです。等価結合は、指定した列に同等の値を持つ行を結合します。オプティマイザが結合の実行を選択する内部アルゴリズムによって、1つの表の等価結合条件における列の合計サイズは、データ・ブロックのサイズ以下に制限される可能性があります。データ・ブロックのサイズは、初期化パラメータDB_BLOCK_SIZEによって指定されます。

関連項目:

結合問合せの使用方法: 例

バンド結合

バンド結合は、データ・セットのキー値が2番目のデータ・セットの指定された範囲(「バンド」)に収まる必要がある特殊なタイプの非等価結合です。最初のデータ・セットと2番目のデータ・セットの両方として同じ表が使用されます。

関連項目:

- バンド結合の詳細は、『Database SQLチューニング・ガイド』を参照してください。
- USE_BANDヒント
- NO_USE_BANDヒント

自己結合

自己結合とは、自己の表結合のことです。この表はFROM句に2回指定され、結合条件の列名を修飾する表の別名が続きます。自己結合を実行するために、Oracle Databaseは結合条件を満たす表の行を結合して戻します。

関連項目:

自己結合の使用方法: 例

デカルト積

結合問合せの2つの表に結合条件がない場合、Oracle Databaseはデカルト積を戻します。この場合、1つの表の各行が別の表の各行に結合されます。デカルト積は常に多数の行を生成するため、有効ではありません。たとえば、それぞれが100行を持つ2つの表のデカルト積は10,000行を生成します。特にデカルト積を必要としないかぎり、必ず結合条件を指定してください。問合せが3つ以上の表を結合し、特定の組に対して結合条件を指定しない場合、オプティマイザは、中間のデカルト積を生成しないように結合順序を選択する可能性があります。

内部結合

内部結合(単純結合)とは、結合条件を満たす行のみを戻す、複数の表の結合です。

外部結合

外部結合は、単純結合の結果を拡張します。外部結合では、結合条件を満たすすべての行に加えて、他の表に結合条件を満たす行が含まれていなくても表の一部またはすべての行が返されます。

- 表AおよびBの外部結合を行い、すべての行をAから戻す問合せ(左側外部結合)を記述するには、FROM句でLEFT [OUTER] JOIN構文を使用するか、WHERE句の結合条件で外部結合演算子(+)をBのすべての列に適用します。Bに一致する行のないAのすべての行に関して、Oracle Databaseは、Bの列を含む任意のSELECT構文のリストの式にNULLを戻します。
- 表AおよびBの外部結合を行い、すべての行をBから戻す問合せ(右側外部結合)を記述するには、FROM句でRIGHT [OUTER] JOIN構文を使用するか、WHERE句の結合条件で外部結合演算子(+)をAのすべての列に適用します。Aに一致する行のないBのすべての行に関して、Oracleは、Aの列を含む任意のSELECT構文のリストの式にNULLを戻します。
- 外部結合を行い、結合条件を満たさない場合にすべての行をAおよびBからNULLで拡張して戻す問合せ(完全外部結合)を記述するには、FROM句でFULL [OUTER] JOIN構文を使用します。

指定する書式にかかわらず、外部結合のWHERE句の副問合せでは列を比較できません。

外部結合を使用すると、疎データ内の欠損を補完できます。このような結合はパーティション化された外部結合と呼ばれ、join_clause構文のquery_partition_clauseを使用して形成されます。疎データとは、時刻や部門などのディメン

ションの一部の値に対する行を持たないデータです。たとえば、販売データの表には通常、売上のない任意の日付の製品に対する行は存在しません。データの欠損の補完は、データの欠損によって分析計算が複雑になる場合や、疎データを直接問い合せた場合に一部のデータを見逃す可能性がある際に役立ちます。

関連項目:

- 外部結合を使用した疎データの欠損の補完方法の詳細は、「join_clause」を参照してください。
- グループ外部結合および疎データの欠損補完の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

Oracleの結合演算子よりも、FROM句のOUTER JOIN構文を使用することをお薦めします。Oracleの結合演算子(+)を使用した外部結合問合せには、次の規則と制限事項があります(これらの規則や制限事項は、FROM句のOUTER JOIN構文にはありません)。

- FROM句の結合構文を含む問合せブロックには結合演算子(+)を指定できません。
- 結合演算子(+)は、WHERE句またはFROM句の左相関のコンテキスト(TABLE句を指定する場合)にのみ指定でき、表またはビューの列にのみ適用されます。
- AおよびBが複数の結合条件によって結合される場合、これらの条件のすべてにおいて結合演算子(+)を使用する必要があります。使用しない場合、Oracle Databaseは単純結合の結果である行のみを戻しますが、外部結合の結果がないことを示す警告やエラーは出力しません。
- ある表を外部問合せに指定して別の表を内部問合せに指定した場合、結合演算子(+)は外部結合を生成しません。
- 自己結合が有効であっても、結合演算子(+)を使用して表を自己に外部結合することはできません。たとえば、次の文は無効です。

```
-- The following statement is not valid:
SELECT employee_id, manager_id
   FROM employees
   WHERE employees.manager_id(+) = employees.employee_id;
```

ただし、次の自己結合は有効です。

```
SELECT e1.employee_id, e1.manager_id, e2.employee_id
   FROM employees e1, employees e2
   WHERE e1.manager_id(+) = e2.employee_id
   ORDER BY e1.employee_id, e1.manager_id, e2.employee_id;
```

- 結合演算子(+)は任意の式ではなく、列にのみ適用することができます。ただし、任意の式には結合演算子(+)でマークされた1つ以上の列を含めることができます。
- 結合演算子(+)を含むWHERE条件は、OR論理演算子を使用する他の条件と結合できません。
- WHERE条件は、IN比較条件を使用して、結合演算子(+)でマークされた列を式と比較できません。

WHERE句に表Bの列と定数を比較する条件が含まれる場合、Oracleがこの列に対してNULLを生成する表Aの列を戻すように、結合演算子(+)をこの列に適用する必要があります。それ以外の場合、Oracleは単純結合の結果のみを戻します。

以前のリリースのOracle Databaseでは、2組以上の表の外部結合を実行する問合せで、単一表は他の1つの表のみに対してNULL生成された表にすることができました。Oracle Database 12c以降では、単一表は複数の表に対してNULL生成された表にすることができます。たとえば、Oracle Database 12cでは、次の文が許容されています。

```
SELECT * FROM A, B, D
  WHERE A.c1 = B.c2(+) and D.c3 = B.c4(+);
```

この例では、NULL生成された表Bは、2つの表AとDに外部結合されます。外部結合の構文については、「SELECT」を参照してください。

アンチ結合

アンチ結合は、述語の右側に対応する行を持たない述語の左側の行を戻します。この結合は、右側の副問合せに一致しない(NOT IN)行を戻します。

関連項目:

「アンチ結合の使用方法：例」

セミ結合

セミ結合は、述語の右側の複数の行が副問合せの条件を満たす場合に、述語の左側から行を重複させずにEXISTS副問合せに一致する行を戻します。

副問合せがWHERE句のORブランチに指定されている場合、セミ結合およびアンチ結合変換は実行できません。

関連項目:

「セミ結合の使用方法：例」

副問合せの使用方法

副問合せは、複数部分の問合せに応答します。たとえば、Taylorの部門で働いている人を判断するには、まずTaylorが働く部門を判断する副問合せを使用できます。その後、親SELECT文で元の問合せに応答することができます。SELECT文のFROM句の副問合せは、インライン・ビューとも呼ばれます。任意の数の副問合せをインライン・ビュー内にネストできます。また、SELECT文のWHERE句の副問合せは、ネストした副問合せとも呼ばれます。ネストした副問合せには、最大255レベルの副問合せをネストできます。

副問合せは、別の副問合せを含むことができます。トップレベル問合せのFROM句内の副問合せレベルの数には、制限がありません。WHERE句には、最大255レベルの副問合せをネストできます。

副問合せにある列が、含まれる文の列と同じ名前を持つ場合、含まれる文の表の列に表名または別名で参照の接頭辞を付ける必要があります。文をさらに読みやすくするには、常に、表、ビューまたはマテリアライズド・ビューの名前または別名で副問合せの列を修飾します。

ネストした副問合せが、その副問合せ、またはネストした副問合せから1レベルまたは複数レベル上位の親である文で参照する表の列を参照する場合、Oracleは相関副問合せを行います。親である文は、副問合せがネストしているSELECT、UPDATEまたはDELETE文のいずれかです。概念的に、相関副問合せは、親である文によって処理される行ごとに1回評価されます。しかし、オプティマイザは、問合せを結合として書き換えるか、他のなんらかの手法を使用して意味的に同等の問合せを形成することを選択する可能性があります。Oracleは、副問合せで指定された表内を検索した後、親である文で指定された表内を検索することによって、副問合せ内の未修飾列を解決します。

相関副問合せは、応答が親である文によって処理された各列の値に依存する複数部分の問合せに応答します。たとえば、相関副問合せを使用して、部門内で給与が平均給与以上の従業員を判断することができます。この場合、相関副問合せは独自で各部門の平均給与を計算します。

関連項目:

相関副問合せの使用方法: 例

副問合せは、次の用途に使用します。

- INSERTまたはCREATE TABLE文のターゲット表に挿入する一連の行を定義します。
- CREATE VIEWまたはCREATE MATERIALIZED VIEW文のビューまたはマテリアライズド・ビューに含める一連の行を定義します。
- UPDATE文の既存の行に割り当てる1つ以上の値を定義します。
- SELECT、UPDATEおよびDELETE文のWHERE句、HAVING句またはSTART WITH句における条件に対する値を定義します。
- 含まれる問合せによって操作される表を定義します。

  表名を指定する場合と同様に、問合せを含むFROM句に副問合せを指定することによってこれらのことを行います。INSERT、UPDATEおよびDELETE文においても、このようにして表のかわりに副問合せを使用することができます。

  このように使用される副問合せでは、その副問合せ内で定義した相関変数と、その副問合せを含む問合せのブロック内で定義した相関変数が使用できます。詳細は、「table_collection_expression」を参照してください。

  1つの行から1つの列の値を戻すスカラー副問合せは、有効な書式の式です。構文でexprをコールするほとんどの場合に、スカラー副問合せ式を使用できます。詳細は、「スカラー副問合せ式」を参照してください。

ネストされた副問合せのネスト解除

副問合せは、親である文のWHERE句内にあるときはネストされています。ネストされた副問合せを持つ文を評価する場合、Oracle Databaseは、副問合せ部分を複数回評価する必要があり、効果的なアクセス・パスまたは結合を見逃してしまう可能性があります。

副問合せのネスト解除によって、副問合せの本体がネスト解除され、その副問合せを含む文の本体に結合されます。これによって、アクセス・パスおよび結合の評価時に、オプティマイザが副問合せと文を1つのものと判断します。オプティマイザは、ほぼすべての副問合せをネスト解除できますが、いくつか例外があります。これらの例外としては、階層副問合せ、およびROWNUM疑似列、集合演算子の1つ、ネストした集計ファンクション、副問合せの直接的な外部問合せブロックではない問合せブロックへの相関参照を含む副問合せなどがあります。

制約がない場合、オプティマイザは、次のネストされた副問合せを自動的にネスト解除します(ただし、ネスト解除しない場合もあります)。

- 相関関係のないIN副問合せ
- INおよびEXISTS相関副問合せ(集計ファンクションまたはGROUP BY句を含まない場合)

ネスト解除された拡張副問合せを行うには、次のタイプの副問合せをネスト解除するようにオプティマイザに指示します。

- 副問合せにHASH_AJまたはMERGE_AJヒントを指定して、NOT IN副問合せをネスト解除します。
- 副問合せにUNNESTヒントを指定して、その他の副問合せをネスト解除します。

関連項目:

ヒントの詳細は、「ヒント」を参照してください。

DUAL表からの選択

DUALは、データ・ディクショナリとともにOracle Databaseによって自動的に作成された表です。DUALは、ユーザーSYSのスキーマにありますが、すべてのユーザーがDUALという名前でアクセスすることができます。VARCHAR2(1)として定義されているDUMMY列を持ち、X値を持つ行を含みます。DUAL表から選択することは、定数式をSELECT文で計算する場合に便利です。DUALには行が1つしかないため、定数が返されるのは1回のみです。一方で、任意の表から定数、疑似列または式を選択できますが、値は表の行の数のみ戻されます。DUALから定数値を選択する例は、「SQLファンクション」を参照してください。

ノート:

Oracle Database 10g リリース 1 以降では、DUMMY 列を含まない式を計算する場合に、論理 I/O が DUAL 表で実行されません。この最適化は、実行計画で FAST DUAL としてリストされます。DUAL の DUMMY 列に対して SELECT を実行した場合、この最適化は行われず、論理 I/O が実行されます。

分散問合せ

Oracle分散データベース管理システム・アーキテクチャによって、Oracle NetおよびOracle Databaseサーバーを使用するリモート・データベースにアクセスできます。名前の最後に@dblinkを追加して、リモート表、ビューまたはマテリアライズド・ビューを識別できます。dblinkは、リモート表、ビューまたはマテリアライズド・ビューを含むデータベースへのデータベース・リンクの完全な名前または部分的な名前である必要があります。

関連項目:

データベース・リンクの参照方法の詳細は、「リモート・データベース内のオブジェクトの参照」を参照してください。

分散問合せの制限事項

現在、分散問合せには、FOR UPDATE句によってロックされたすべての表、および問合せによって選択されたLONG列を持つすべての表が、同じデータベース上に位置している必要があるという制限があります。また、Oracle Databaseは現在、リモート表にあるユーザー定義型またはオブジェクトREFのデータ型を選択する分散問合せをサポートしていません。

10 SQL文: ADMINISTER KEY MANAGEMENTからALTER JAVA

この章では、様々なSQL文をリストしてアルファベット順にSQL文を説明します。その他のSQL文については、後続の章でアルファベット順に説明します。

この章の構成は、次のとおりです。

様々な種類のSQL文

次の項にあるリストは、SQL文の機能の概要について、次のカテゴリに分類して説明しています。

- データ定義言語(DDL)文
- データ操作言語(DML)文
- トランザクション制御文
- セッション制御文
- システム制御文
- 埋込みSQL文

データ定義言語(DDL)文

データ定義言語(DDL)文によって、次のタスクを実行できます。

- スキーマ・オブジェクトの作成、変更および削除
- 権限およびロールの付与および取消し
- 表、索引またはクラスタ上の情報の分析
- 監査オプションの構築
- データ・ディクショナリへのコメントの追加

CREATE、ALTERおよびDROPコマンドは、特定のオブジェクトに対して排他的アクセスを必要とします。たとえば、別のユーザーが特定の表でトランザクションをオープンしている場合、ALTER TABLE文は実行できません。

GRANT、REVOKE、ANALYZE、AUDITおよびCOMMENTコマンドは、特定のオブジェクトに対する排他的アクセスを必要としません。たとえば、他のユーザーが表を更新しているときでも、その表を分析できます。

Oracle Databaseは、暗黙的にすべてのDDL文の前後で現在のトランザクションをコミットします。

DDL文はブロッキングまたは非ブロッキングのいずれかで、どちらのタイプのDDL文にも内部構造の排他ロックが必要です。

関連項目:

ブロック化DDLと非ブロック化DDLの相違について学習するには、『Oracle Database開発ガイド』を参照してください。

DDL文の多くは、Oracle Databaseにスキーマ・オブジェクトを再コンパイルまたは再認可させることができます。Oracle Databaseがスキーマ・オブジェクトを再コンパイルまたは再認可する方法、およびDDL文によってそれを実行する環境については、『Oracle Database概要』を参照してください。

DDL文は、DBMS_SQLパッケージを使用したPL/SQLによってサポートされます。

関連項目:

このパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

次に、DDL文を示します。

- ALTER (ALTER SESSIONおよびALTER SYSTEMを除くALTERで始まるすべての文。「セッション制御文」および「システム制御文」を参照)
- ANALYZE
- ASSOCIATE STATISTICS
- AUDIT
- COMMENT
- CREATE ... (CREATEで始まるすべての文)
- DISASSOCIATE STATISTICS
- DROP ... (DROPで始まるすべての文)
- FLASHBACK ... (FLASHBACKで始まるすべての文)
- GRANT
- NOAUDIT
- PURGE
- RENAME
- REVOKE
- TRUNCATE

データ操作言語(DML)文

データ操作言語(DML)文は、既存スキーマ・オブジェクトのデータにアクセスし、操作します。次の文は、現在のトランザクションを暗黙的にコミットしません。次に、データ操作言語文を示します。

- CALL
- DELETE
- EXPLAIN PLAN
- INSERT
- LOCK TABLE
- MERGE
- SELECT
- UPDATE

SELECT文は、DML文の制限された形式であり、データベース内のデータへのアクセスのみが可能です。アクセスしたデータを操作してから問合せの結果を戻すことはできますが、データベースに格納されたデータを操作することはできません。

SELECT文は、動的に実行されるときにのみPL/SQLでサポートされます。ただし、類似のPL/SQL文SELECT INTOをPL/SQLコード内で使用でき、この文は動的に実行する必要はありません。CALLおよびEXPLAIN PLAN文は、動的に実行されるときにのみPL/SQLでサポートされます。他のすべてのDML文は、PL/SQLで完全にサポートされます。

トランザクション制御文

トランザクション制御文は、DML文で行った変更を管理します。次に、トランザクション制御文を示します。

- COMMIT
- ROLLBACK
- SAVEPOINT
- SET TRANSACTION
- SET CONSTRAINT

COMMITおよびROLLBACKコマンドの特定書式以外のトランザクション制御文は、PL/SQLでサポートされます。制限については、「COMMIT」および「ROLLBACK」を参照してください。

セッション制御文

セッション制御文は、ユーザー・セッションのプロパティを動的に管理します。次の文は、現在のトランザクションを暗黙的にコミットしません。

PL/SQLは、セッション制御文をサポートしません。次に、セッション制御文を示します。

- `ALTER SESSION`
- `SET ROLE`

システム制御文

唯一のシステム制御文である`ALTER SYSTEM`は、Oracle Databaseインスタンスのプロパティを動的に管理します。この文によって、現行トランザクションが暗黙的にコミットされることはなく、この文はPL/SQLではサポートされません。

埋込みSQL文

埋込みSQL文は、DDL、DMLおよびトランザクション制御文を手続き型言語プログラム内に入れます。埋込みSQLは、Oracleプリコンパイラでサポートされており、次のマニュアルに記載されています。

- 『Pro*COBOLプログラマーズ・ガイド』
- 『Pro*C/C++プログラマーズ・ガイド』

SQL文に関する章の構成

このマニュアルのすべてのSQL文は、次の項に編成されています。

構文

構文図では、文を構成するキーワードとパラメータを示します。

ノート:

すべてのキーワードとパラメータがすべての状況で有効なわけではありません。各文および句の「セマンティクス」を参照し、構文上の制限を学習するようにしてください。

目的

「目的」では文の基本的な使用について示します。

前提条件

「前提条件」項では、必要な権限、および文を使用する前に実行する必要のあるステップを示します。特に指定がないかぎり、ご使用のインスタンスでデータベースがオープンされている必要があります。

セマンティクス

「セマンティクス」の項では、構文を構成するキーワード、パラメータおよび句の用途について説明します。また、制限事項およびその他のノートについても説明します。(この章で使用するキーワードおよびパラメータの表記規則の詳細は、「はじめに」を参照してください。)

例

「例」では文の様々な句とパラメータの使用例を示します。

ADMINISTER KEY MANAGEMENT

目的

ADMINISTER KEY MANAGEMENT文は、透過的データ暗号化のための統合されたキー管理インタフェースを提供します。この文は、次の目的に使用できます。

- ソフトウェア・キーストアおよびハードウェア・キーストアの管理
- 暗号化キーの管理
- シークレットの管理

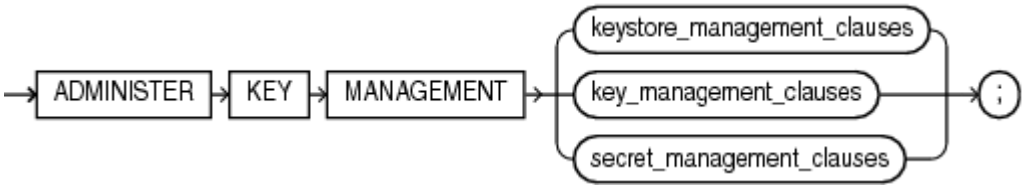
前提条件

ADMINISTER KEY MANAGEMENTシステム権限、またはSYSKMシステム権限が必要です。

CONTAINER句を指定する場合は、マルチテナント・コンテナ・データベース(CDB)に接続している必要があります。CONTAINER = ALLを指定するには、現在のコンテナがルートである必要があります。また、共通に付与されているADMINISTER KEY MANAGEMENTまたはSYSKM権限が必要です。

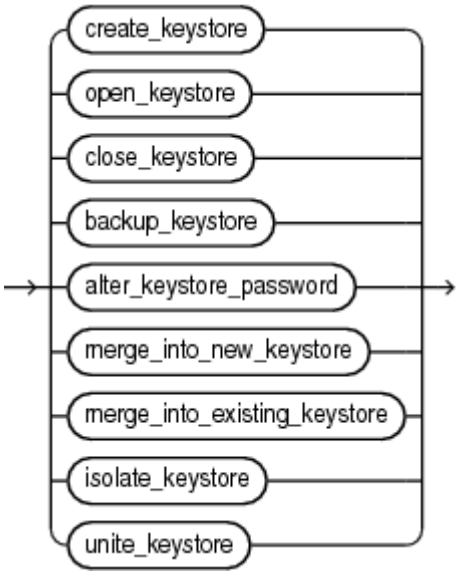
構文

administer_key_management::=



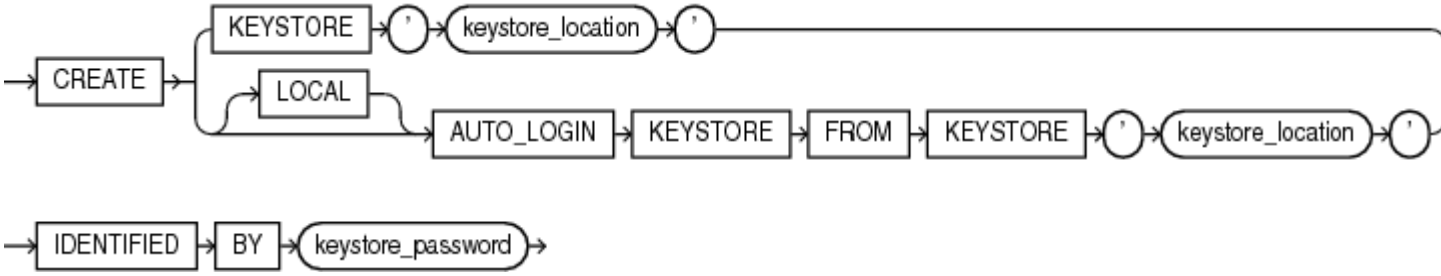
(keystore_management_clauses::=、key_management_clauses::=、secret_management_clauses::=)

keystore_management_clauses::=

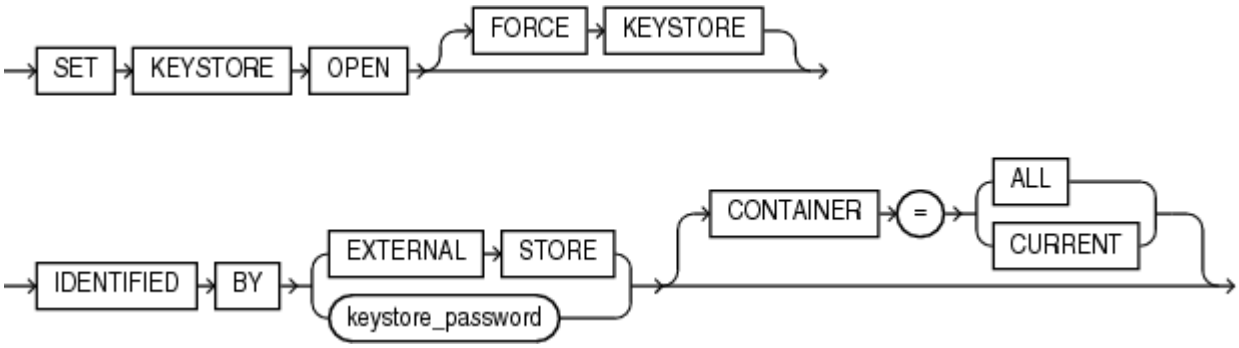


(create_keystore::=、open_keystore::=、close_keystore::=、backup_keystore::=、alter_keystore_password::=、merge_into_new_keystore::=、merge_into_existing_keystore::=)

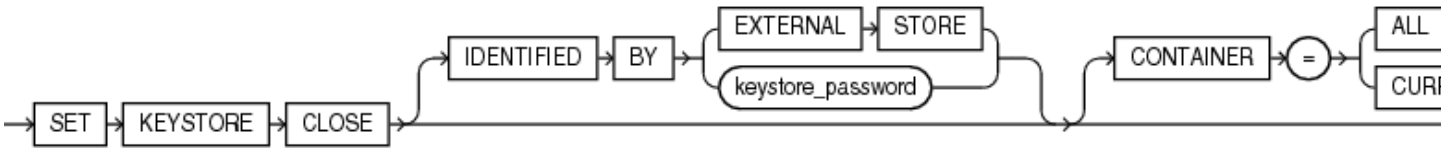
create_keystore::=



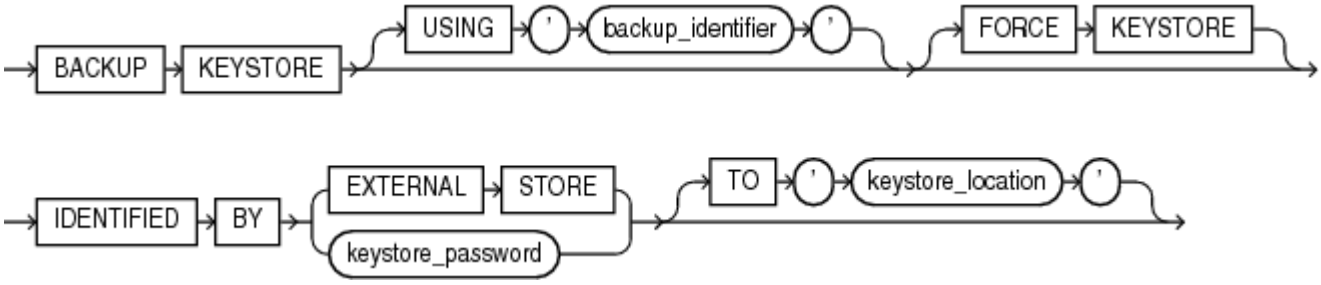
open_keystore::=



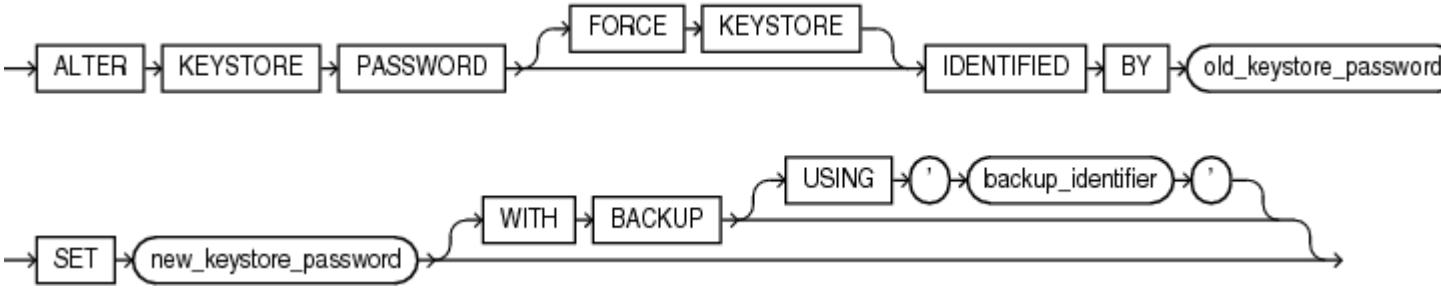
close_keystore::=



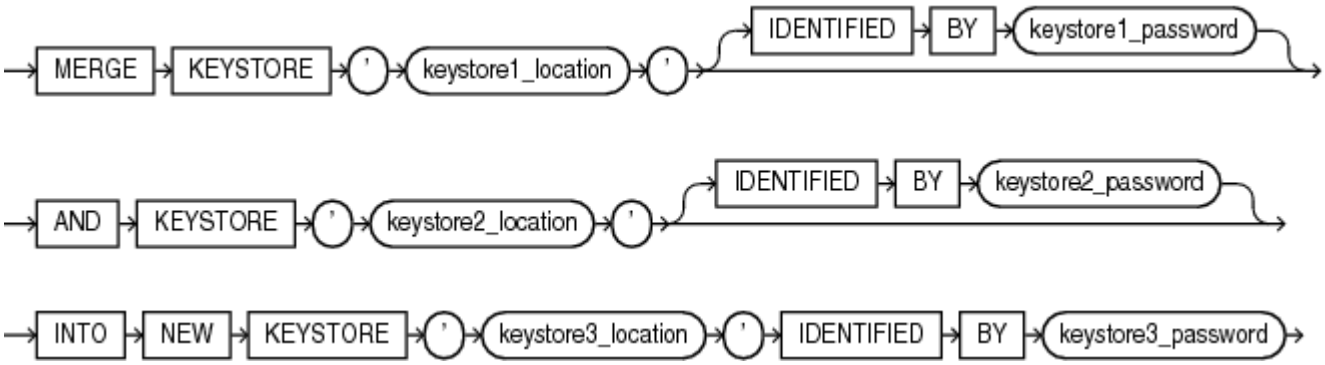
backup_keystore::=



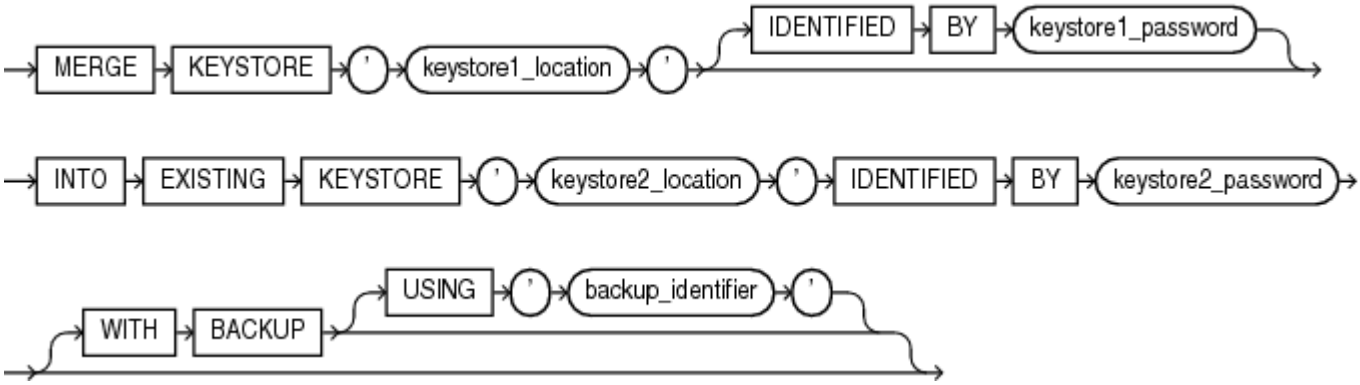
alter_keystore_password::=



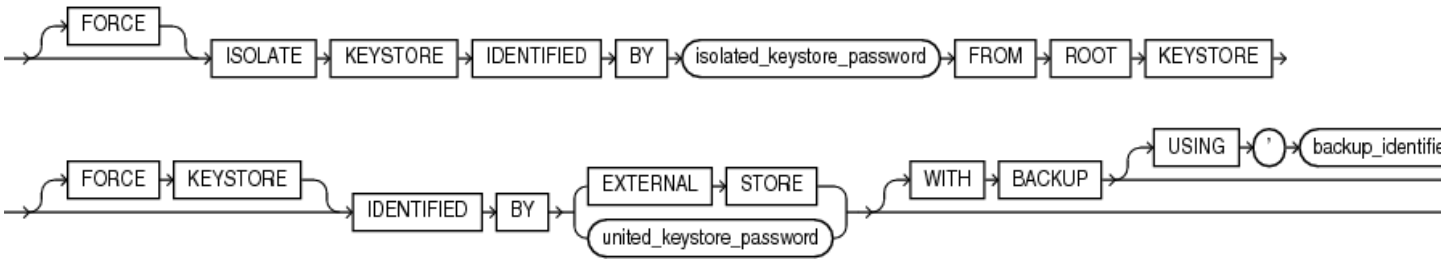
merge_into_new_keystore::=



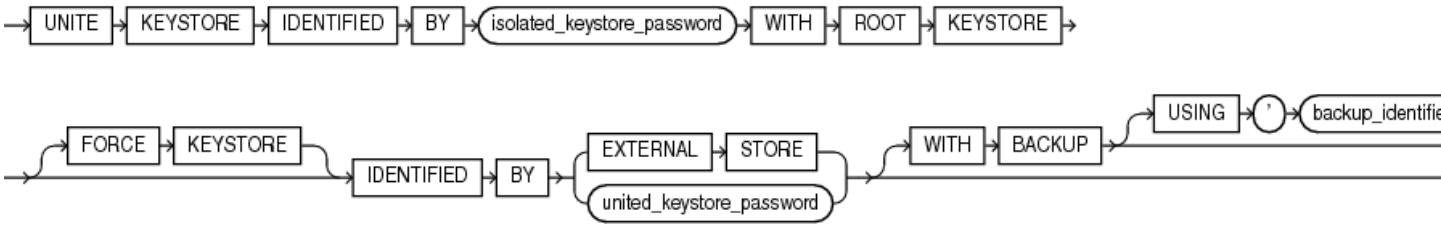
merge_into_existing_keystore::=



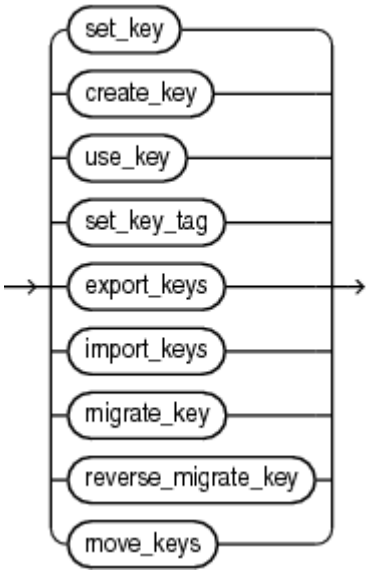
isolate_keystore::=



unite_keystore ::=

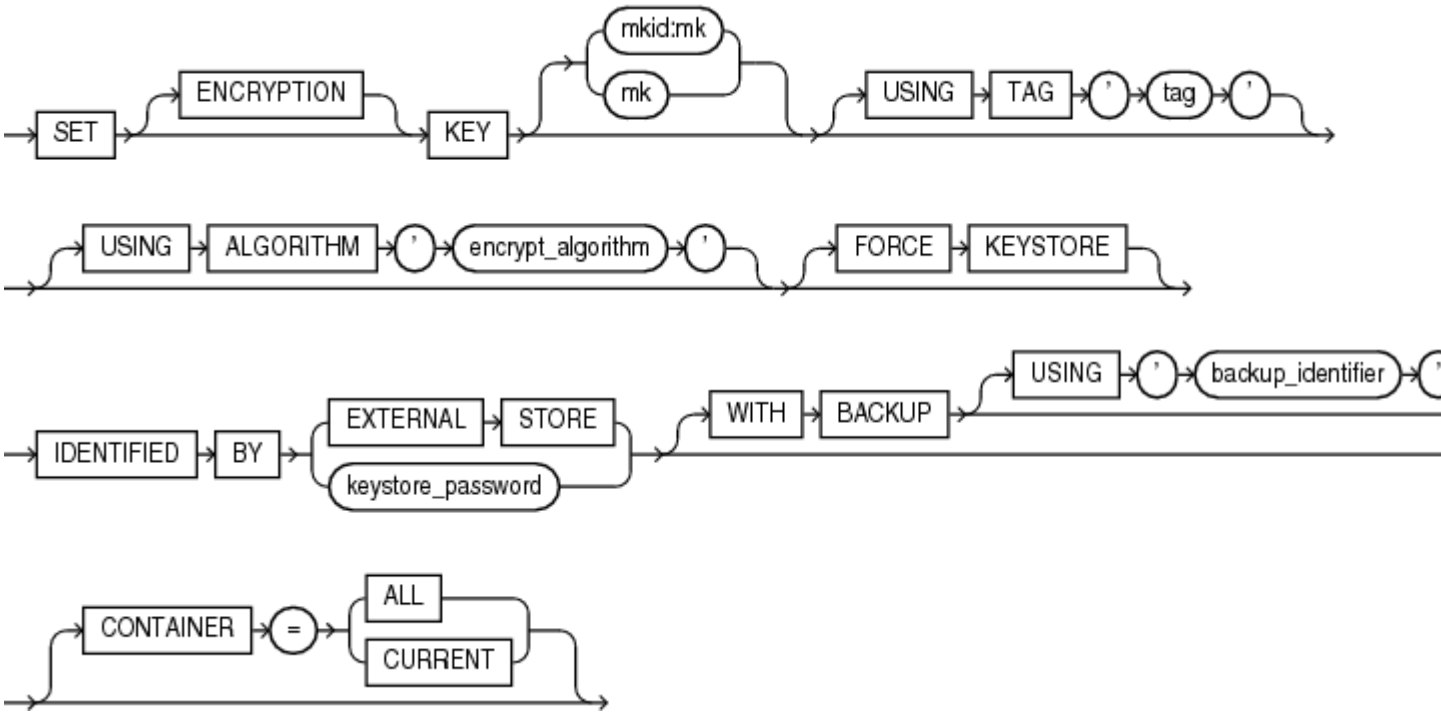


key_management_clauses::=

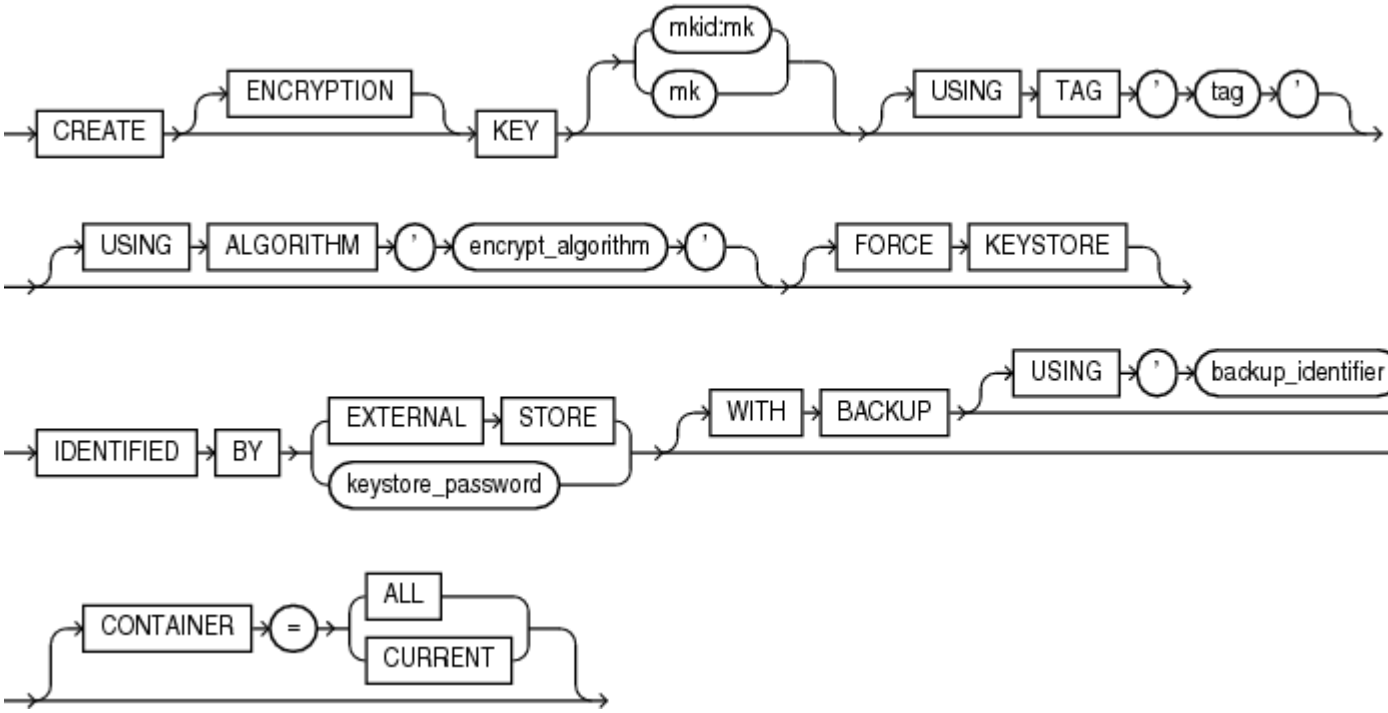


(set_key::=、create_key::=、use_key::=、set_key_tag::=、export_keys::=、import_keys::=、migrate_key::=、reverse_migrate_key::=)

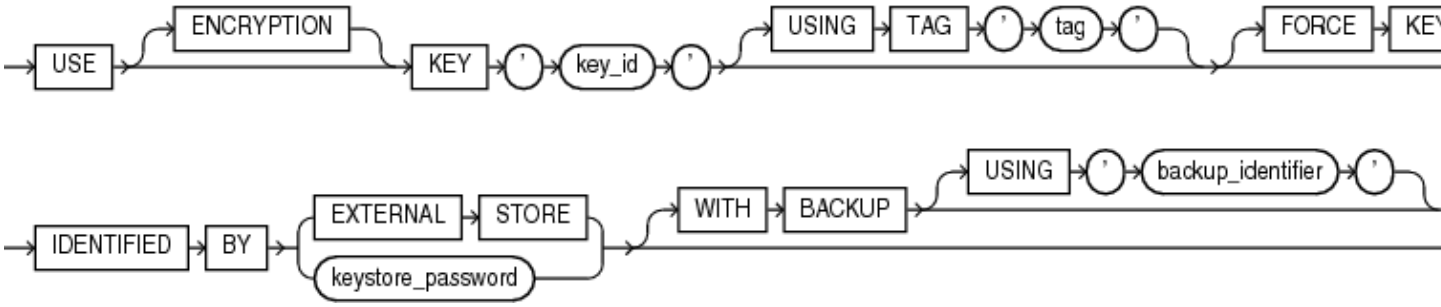
set_key::=



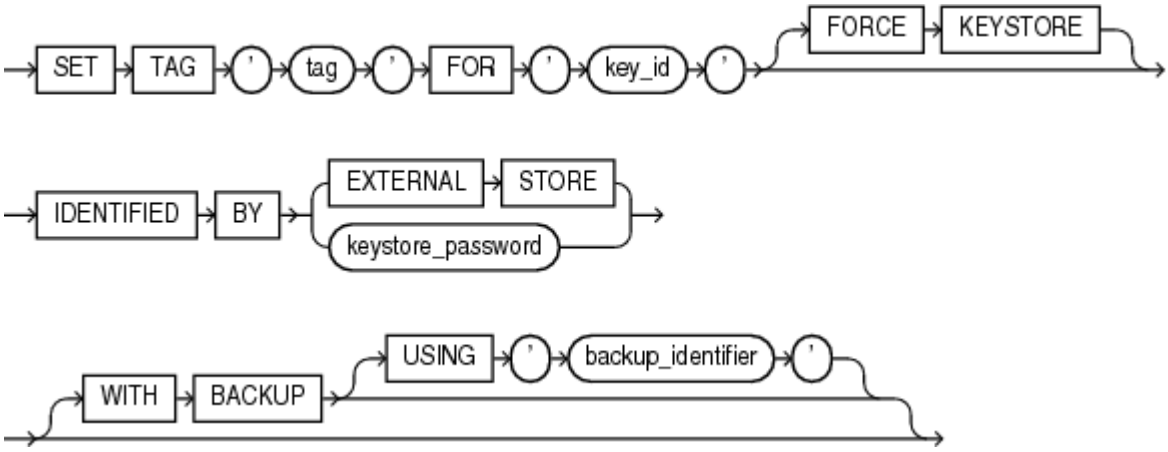
create_key::=



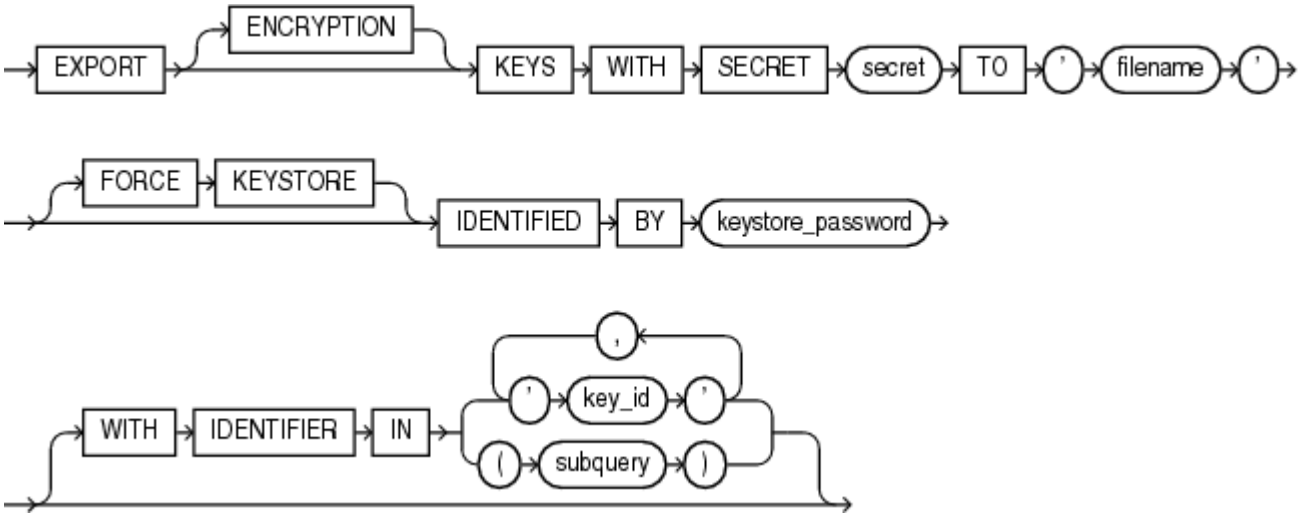
use_key::=



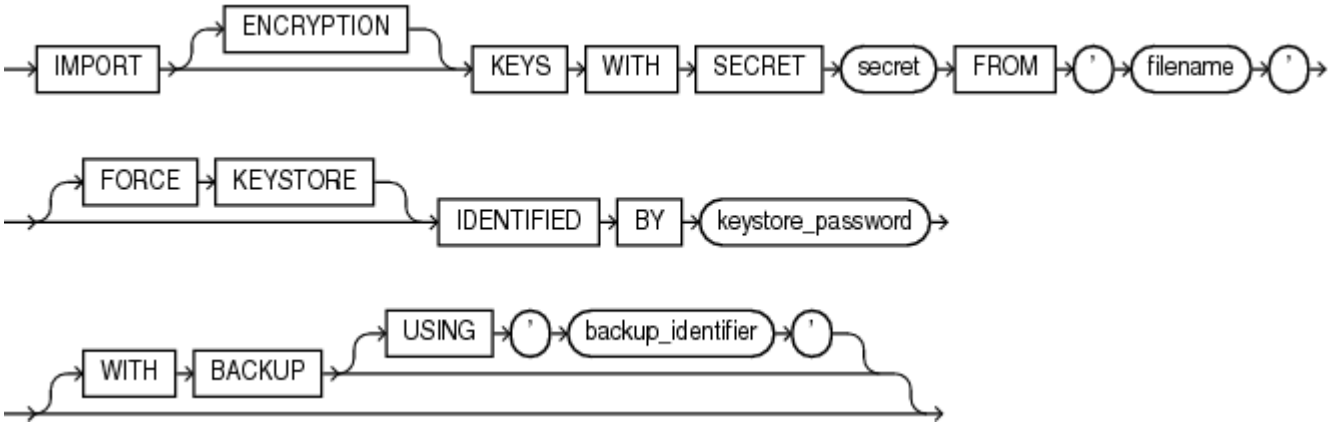
set_key_tag::=



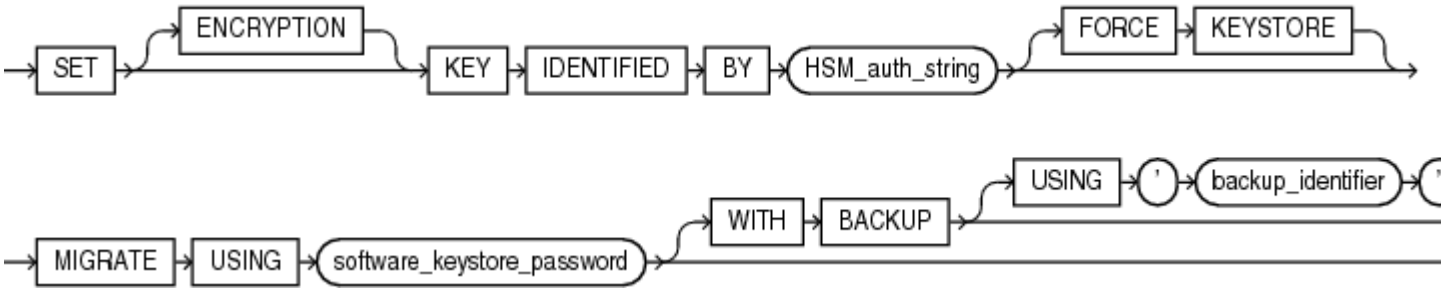
export_keys::=



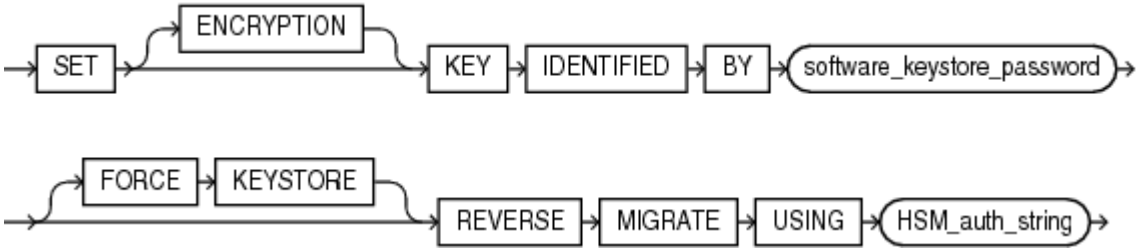
import_keys::=



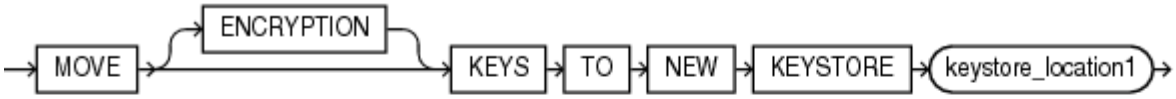
migrate_key::=

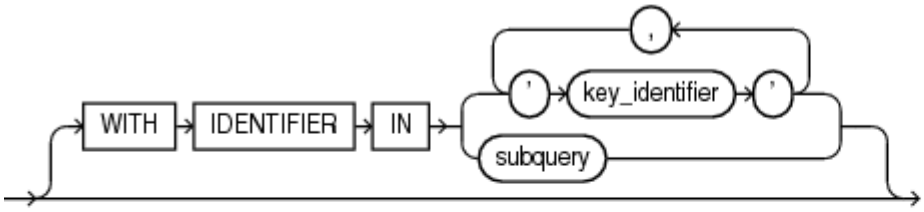


reverse_migrate_key::=



move_keys ::=

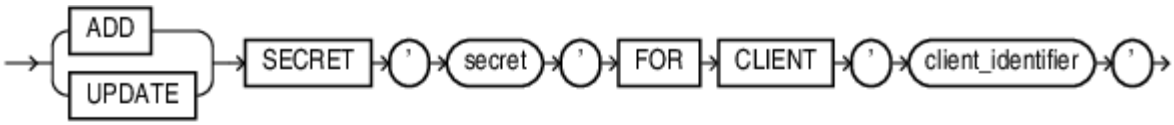


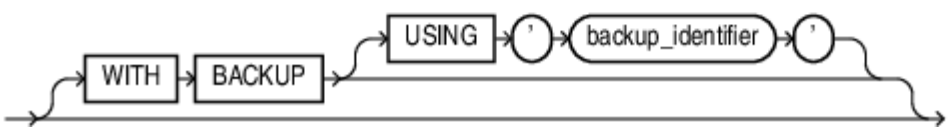


secret_management_clauses::=

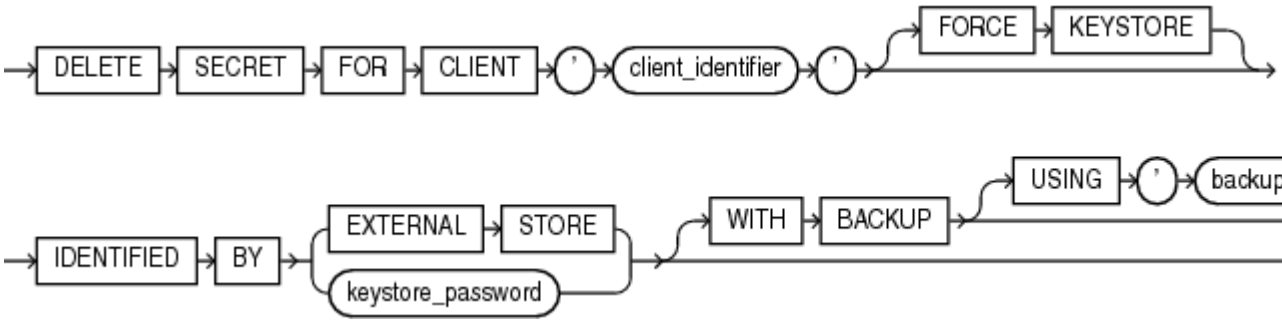
(add_update_secret::=、delete_secret::=)

add_update_secret::=

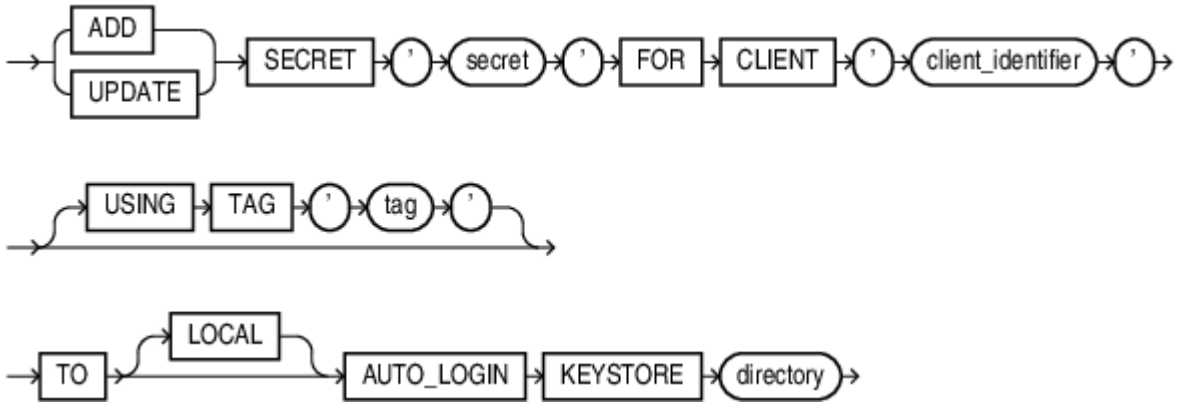




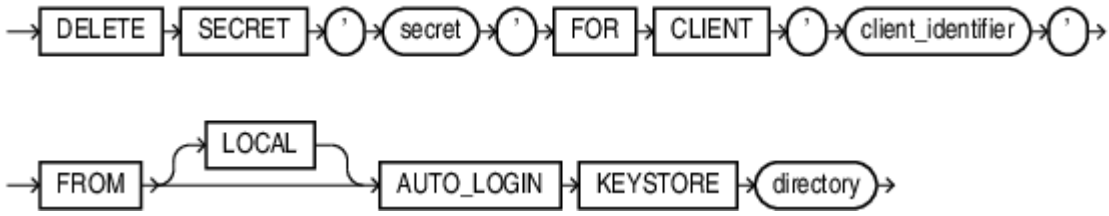
delete_secret::=



add_update_secret_seps::=



delete_secret_seps::=



セマンティクス

keystore_management_clauses

これらの句を使用して、次のキーストア管理操作を実行できます。

- ソフトウェア・キーストアの作成
- ソフトウェア・キーストアまたはハードウェア・キーストアのオープンとクローズ
- パスワードで保護されたソフトウェア・キーストアのバックアップ
- パスワードで保護されたソフトウェア・キーストアのパスワードの変更
- 2つの既存のソフトウェア・キーストアの、パスワードで保護された新しいソフトウェア・キーストアへのマージ
- 1つの既存のソフトウェア・キーストアの、パスワードで保護された既存のソフトウェア・キーストアへのマージ
- プラガブル・データベース(PDB)で固有のキーストアを管理できるようにするための、コンテナ・データベース(CDB)からのPDBのキーストアの分離

- CDBとのPDBのキーストアの統合

create_keystore

この句を使用して、パスワードで保護されたソフトウェア・キーストアと自動ログイン・ソフトウェア・キーストアの2種類のソフトウェア・キーストアを作成できます。この句をマルチテナント環境で発行するには、ルートに接続する必要があります。

CREATE KEYSTORE

パスワードで保護されたソフトウェア・キーストアを作成するには、この句を指定します。

- `keystore_location`には、ソフトウェア・キーストア・ディレクトリのフルパス名を指定します。キーストアは、このディレクトリに`ewallet.p12`という名前のファイルで作成されます。`WALLET_ROOT`パラメータが設定されている場合、この句はオプションです。ご使用のシステムのソフトウェア・キーストア・ディレクトリを確認する方法を学習するには、『Oracle Database Advanced Securityガイド』を参照してください。
- `IDENTIFIED BY`句を使用すると、キーストアのパスワードを設定できます。詳細は、「キーストア・パスワードを指定するときのノート」を参照してください。

CREATE [LOCAL] AUTO_LOGIN KEYSTORE

自動ログイン・ソフトウェア・キーストアを作成するには、この句を指定します。自動ログイン・ソフトウェア・キーストアは、パスワードで保護された既存のソフトウェア・キーストアから作成されます。自動ログイン・キーストアは、システム生成のパスワードを保持します。これは、パスワードで保護されたソフトウェア・キーストアと同じディレクトリにある、`cwallet.sso`という名前のPKCS#12ベースのファイルに格納されます。

- デフォルトでは、Oracleが作成する自動ログイン・キーストアは、このキーストアが格納されているコンピュータ以外のコンピュータからも開けるようになります。`LOCAL`キーワードを指定すると、Oracle Databaseによりローカルの自動ログイン・キーストアが作成されますが、そのキーストアはそれが格納されているコンピュータからのみ開けます。
- `keystore_location`には、パスワードで保護された既存のソフトウェア・キーストアが格納されているディレクトリのフルパス名を指定します。パスワードで保護されたソフトウェア・キーストアは、開いていても閉じていてもかまいません。
- `IDENTIFIED BY`句を使用すると、パスワードで保護された既存のソフトウェア・キーストアにパスワードを指定できます。詳細は、「キーストア・パスワードを指定するときのノート」を参照してください。

キーストアの作成の制限事項

ローカルであるかどうかにかかわらず、パスワードで保護されたソフトウェア・キーストアと自動ログイン・ソフトウェア・キーストアは、どちらも1つのディレクトリ内に1つまで作成できます。

関連項目:

ソフトウェア・キーストアの作成の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

open_keystore

この句を使用すると、パスワードで保護されたソフトウェア・キーストアまたはハードウェア・キーストアを開くことができます。

ノート:

自動ログイン・ソフトウェア・キーストアとローカルの自動ログイン・ソフトウェア・キーストアを開くために、この句を使用する必要はありません。これらのキーストアは、マスター暗号化キーへのアクセスにより自動的に開かれます。

- FORCE KEYSTORE句は、PDBでキーストアを開く場合に役立ちます。これにより、確実にPDBキーストアを開く前にCDBルート・キーストアが開きます。詳細は、「FORCE KEYSTORE句のノート」を参照してください。
- IDENTIFIED BY句を使用すると、キーストアのパスワードを指定できます。詳細は、「キーストア・パスワードを指定するときのノート」を参照してください。
- CONTAINER句は、CDBに接続しているときに適用されます。

  現在のコンテナがプラガブル・データベース(PDB)である場合は、CONTAINER = CURRENTを指定すると、PDBでキーストアが開きます。PDBでキーストアを開く前に、ルートでキーストアが開かれている必要があります。

  現在のコンテナがルートである場合は、CONTAINER = CURRENTを指定すると、ルートでキーストアが開き、CONTAINER = ALLと指定すると、ルートとすべてのPDBでキーストアが開きます。

  この句を省略すると、CONTAINER = CURRENTがデフォルトになります。

関連項目:

- Oracle Database Advanced Securityガイド 統一モードでのキーストアおよびTDEマスター暗号化キーの管理
- Oracle Database Advanced Securityガイド 分離モードでのキーストアおよびTDEマスター暗号化キーの管理
- パスワードベースのソフトウェア・キーストアとハードウェア・キーストアを開く方法の詳細は、Oracle Database Advanced Securityガイドを参照してください。

close_keystore

この句を使用すると、パスワードで保護されたソフトウェア・キーストア、自動ログイン・ソフトウェア・キーストアまたはハードウェア・キーストアを閉じることができます。キーストアを閉じると、すべての暗号化操作と復号化操作が無効になります。データを暗号化または復号化しようとしたり、暗号化データにアクセスしようとすると、エラーが発生します。

- パスワードで保護されたソフトウェア・キーストアまたはハードウェア・キーストアを閉じるには、IDENTIFIED BY句を指定します。詳細は、「キーストア・パスワードを指定するときのノート」を参照してください。
- 自動ログイン・キーストアを閉じる場合は、IDENTIFIED BY句を指定しないでください。自動ログイン・キーストアを閉じる前に、V$ENCRYPTION_WALLETビューのWALLET_TYPE列を確認してください。AUTOLOGINが戻されたら、キーストアを閉じることができます。そうしないと、キーストアを閉じようとするとエラーが発生します。
- CONTAINER句は、CDBに接続しているときに適用されます。

  現在のコンテナがPDBである場合は、CONTAINER = CURRENTを指定すると、PDBでキーストアが閉じます。

  現在のコンテナがルートである場合は、CONTAINER = CURRENT句とCONTAINER = ALL句は同じ結果になり、どちらの句でもルートとすべてのPDBでキーストアが閉じます。

  この句を省略すると、CONTAINER = CURRENTがデフォルトになります。

関連項目:

キーストアを閉じる方法の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

backup_keystore

この句を使用すると、パスワードで保護されたソフトウェア・キーストアをバックアップできます。このキーストアは、開いている必要があります。

- デフォルトでは、Oracle Databaseは、`ewallet_timestamp.p12`という形式の名前でバックアップ・ファイルを作成します。`timestamp`は、UTC形式でのファイル作成時のタイムスタンプです。オプションの`USING 'backup_identifier'`句を使用すると、バックアップ・ファイルの名前に追加するバックアップ識別子を指定できます。たとえば、バックアップ識別子に`'Backup1'`を指定すると、Oracle Databaseは、`ewallet_timestamp_Backup1.p12`という形式の名前でバックアップ・ファイルを作成します。
- `FORCE KEYSTORE`句を使用すると、キーストアが閉じている場合でも、この操作が可能になります。詳細は、「FORCE KEYSTORE句のノート」を参照してください。
- `IDENTIFIED BY`句を使用すると、キーストアのパスワードを指定できます。詳細は、「キーストア・パスワードを指定するときのノート」を参照してください。
- オプションの`TO 'keystore_location'`句を使用すると、バックアップ・ファイルの作成先ディレクトリを指定できます。この句を省略すると、バックアップは、キーストアがバックアップされているディレクトリに作成されます。

関連項目:

パスワードベースのソフトウェア・キーストアのバックアップ方法の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

alter_keystore_password

この句を使用すると、パスワードで保護されたソフトウェア・キーストアのパスワードを変更できます。このキーストアは、開いている必要があります。

- `FORCE KEYSTORE`句を使用すると、キーストアが閉じている場合でも、この操作が可能になります。詳細は、「FORCE KEYSTORE句のノート」を参照してください。
- `old_keystore_password`には、キーストアの古いパスワードを指定します。`new_keystore_password`には、キーストアの新しいパスワードを指定します。詳細は、「キーストア・パスワードを指定するときのノート」を参照してください。
- オプションの`WITH BACKUP`句を使用すると、パスワードを変更する前に、キーストアのバックアップを作成するようにデータベースに指示できます。詳細は、「WITH BACKUP句のノート」を参照してください。

関連項目:

パスワードベースのソフトウェア・キーストアのパスワード変更方法の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

merge_into_new_keystore

この句を使用すると、2つのソフトウェア・キーストアを新しい1つのキーストアにマージできます。2つの構成要素であるキーストアに含まれるキーと属性は、新しいキーストアに追加されます。構成要素であるキーストアには、パスワードベースのソフトウェア・キーストアまたは自動ログイン(ローカルの自動ログインを含む)ソフトウェア・キーストアのいずれもできます。また、それらは開いていても閉じていてもかまいません。新しいキーストアは、パスワードで保護されたソフトウェア・キーストアになります。マージが完了

された時点では、閉じられた状態になります。この句に指定するキーストアは、データベースで使用するように構成されたキーストアでも、それ以外のキーストアでもかまいません。

- `keystore1_location`には、最初のキーストアが格納されているディレクトリのフルパス名を指定します。
- `IDENTIFIED BY keystore1_password`は、最初のキーストアがパスワードベースのソフトウェア・キーストアの場合にかぎり指定できます。詳細は、「キーストア・パスワードを指定するときのノート」を参照してください。
- `keystore2_location`には、2番目のキーストアが格納されているディレクトリのフルパス名を指定します。
- `IDENTIFIED BY keystore2_password`は、2番目のキーストアがパスワードベースのソフトウェア・キーストアの場合にかぎり指定できます。
- `keystore3_location`には、新しいキーストアの作成先になるディレクトリのフルパス名を指定します。
- `keystore3_password`には、新しいキーストアのパスワードを指定します。

関連項目:

ソフトウェア・キーストアのマージ方法の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

merge_into_existing_keystore

この句を使用すると、ソフトウェア・キーストアを、それ以外の既存のソフトウェア・キーストアにマージできます。マージ元のキーストアに含まれるキーと属性は、マージ先のキーストアに追加されます。マージ元のキーストアには、パスワードで保護されたソフトウェア・キーストアまたは自動ログイン(ローカルの自動ログインを含む)ソフトウェア・キーストアのいずれも選択できます。また、それらは開いていても閉じていてもかまいません。マージ先になるキーストアは、パスワードベースのソフトウェア・キーストアにする必要があります。これは、マージの開始時に開いていても閉じられていてもかまいません。ただし、マージが完了した時点では、閉じられた状態になります。この句に指定するキーストアは、データベースで使用するように構成されたキーストアでも、それ以外のキーストアでもかまいません。

- `keystore1_location`には、マージ元のキーストアが格納されているディレクトリのフルパス名を指定します。
- `IDENTIFIED BY keystore1_password`は、マージ元のキーストアがパスワードベースのソフトウェア・キーストアの場合にかぎり、指定できます。
- `keystore2_location`には、マージ先になるキーストアが格納されているディレクトリのフルパス名を指定します。
- `keystore2_password`には、マージ先になるキーストアのパスワードを指定します。
- オプションの`WITH BACKUP`句を使用すると、マージを実行する前にマージ先になるキーストアのバックアップを作成するようにデータベースに指示できます。詳細は、「WITH BACKUP句のノート」を参照してください。

関連項目:

ソフトウェア・キーストアのマージ方法の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

isolate_keystore

コンテナ・データベース(CDB)内のプラガブル・データベース(PDB)では固有のキーストアを作成および管理できます。`isolate_keystore`句を使用すると、テナントで以下を行うことができます。

- CDBのキーから独立した透過的データ暗号化キーの管理

- 独立したキーストアのパスワードの作成

CDB環境では、特定のPDBのキーが保護される方法を選択できます。PDBでは、独立したパスワードでキーを保護するか、CDBの統合されたパスワードを使用できます。

- IDENTIFIED BY句を使用すると、キーストアのパスワードを指定できます。
- isolated_keystore_passwordは、PDBキーストアの独立したパスワードを示します。
- FORCE KEYSTORE句を使用すると、キーストアが閉じている場合でも、この操作が可能になります。詳細は、「FORCE KEYSTORE句のノート」を参照してください。
- united_keystore_passwordは、CDBキーストアのパスワードを示します。
- オプションのWITH BACKUP句を使用すると、パスワードを変更する前に、キーストアのバックアップを作成するようにデータベースに指示できます。詳細は、「WITH BACKUP句のノート」を参照してください。

isolate_keystoreを指定したFORCE句

ADMINISTER KEY MANAGEMENT FORCE ISOLATE KEYSTOREコマンドのFORCE句は、分離されているマスター・キーをPDBのクローンが使用している場合に使用されます。このコマンドにより、CDBキーストアから分離されたPDBキーストアにキーがコピーされます。たとえば:

```
ADMINISTER KEY MANAGEMENT
FORCE ISOLATE KEYSTORE
IDENTIFIED BY <isolated_keystore_password>
FROM ROOT KEYSTORE
[FORCE KEYSTORE]
IDENTIFIED BY [EXTERNAL STORE | <united_keystore_password>]
[WITH BACKUP [USING <backup_identifier>]
```

unite_keystore

unite_keystore句を使用すると、キーストアを独立して管理していたPDBのキーストア管理モードを統合モードに変更できます。統合モードでは、CDB内のPDBを管理するためにCDB$ROOTキーストア・パスワードが使用されます。

- IDENTIFIED BY句を使用すると、キーストアのパスワードを指定できます。
- isolated_keystore_passwordは、PDBキーストアの独立したパスワードを示します。
- FORCE KEYSTORE句を使用すると、キーストアが閉じている場合でも、この操作が可能になります。詳細は、「FORCE KEYSTORE句のノート」を参照してください。
- united_keystore_passwordは、CDBキーストアのパスワードを示します。
- オプションのWITH BACKUP句を使用すると、パスワードを変更する前に、キーストアのバックアップを作成するようにデータベースに指示できます。詳細は、「WITH BACKUP句のノート」を参照してください。

たとえば:

```
ADMINISTER KEY MANAGEMENT
UNITE KEYSTORE
IDENTIFIED BY <isolated_keystore_password>
WITH ROOT KEYSTORE
[FORCE KEYSTORE]
IDENTIFIED BY [EXTERNAL STORE | <united_keystore_password>]
[WITH BACKUP [USING <backup_identifier>]
```

key_management_clauses

これらの句を使用して次のキー管理操作を実行できます。

- マスター暗号化キーの作成とアクティブ化
- 暗号化キーのタグの設定
- キーストアからファイルへの暗号化キーのエクスポート
- ファイルからキーストアへの暗号化キーのインポート
- パスワードで保護されたソフトウェア・キーストアからハードウェア・キーストアへの移行
- ハードウェア・キーストアからパスワードで保護されたソフトウェア・キーストアへの移行

set_key

この句は新しいマスター暗号化キーを作成し、それをアクティブ化します。この句を使用すると、キーストア内に最初のマスター暗号化キーを作成することも、マスター暗号化キーを交換(変更)することもできます。この句を使用するときに、マスター暗号化キーがアクティブな場合は、そのキーが非アクティブ化されてから新しいマスター暗号化キーがアクティブ化されます。キーを含むキーストアは、パスワードで保護されたソフトウェア・キーストアであっても、ハードウェア・キーストアであってもかまいません。このキーストアは、開いている必要があります。

TDEマスター・キーID (MKID)およびTDEマスター暗号化キー(MK)に目的の値を指定して、固有のTDEマスター暗号化キーを作成してください。

- TDEで暗号化されたデータベースでは、どのTDEマスター暗号化キーが使用中であるかを追跡するためにTDEマスター・キーID(MKID)が使用されます。MKID:MKオプションを使用すると、MKIDおよびMKの両方を指定できます。
- MKのみを指定した場合、データベースによってMKIDが生成されるため、指定したMK値を持つTDEマスター暗号化キーを追跡できます。
- MKIDが無効な場合(たとえば、長さが誤っていた場合、または0(ゼロ)の文字列である場合)は、エラーORA-46685: invalid master key identifier or master key valueが表示されます。
- 指定したMKIDがキーストア内の既存のTDEマスター暗号化キーのMKIDと同じである場合は、エラーORA-46684: master key identifier exists in the keystoreが表示されます。
- MKIDおよびMKのいずれかが無効な場合は、エラーORA-46685: invalid master key identifier or master key valueが表示されます。
- set_key句およびcreate_key句にMKID:MKオプションを指定できます。
- ENCRYPTIONキーワードはオプションで、意味を明確化するために使用されます。
- オプションのUSING TAG句を指定すると、タグを新しいマスター暗号化キーに関連付けできます。詳細は、「USING TAG句のノート」を参照してください。
- USING ALGORITHM句を指定すると、指定された暗号化アルゴリズムに準拠するマスター暗号化キーが作成されます。encrypt_algorithmには、AES256、ARIA256、GOST256またはSEED128を指定できます。この句を指定するには、COMPATIBLE初期化パラメータが12.2以上に設定されている必要があります。この句を指定しない場合、AES256がデフォルトになります。

  ARIA、SEEDおよびGOSTアルゴリズムは、暗号化とハッシュについての国や政府の標準です。詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。
- FORCE KEYSTORE句を使用すると、キーストアが閉じている場合でも、この操作が可能になります。詳細は、

「FORCE KEYSTORE句のノート」を参照してください。

- IDENTIFIED BY句を使用すると、キーストアのパスワードを指定できます。詳細は、「キーストア・パスワードを指定するときのノート」を参照してください。
- WITH BACKUP句と、オプションのUSING 'backup_identifier'句を指定すると、新しいマスター暗号化キーが作成される前に、キーストアのバックアップが作成されます。詳細は、「WITH BACKUP句のノート」を参照してください。
- CONTAINER句は、CDBに接続しているときに適用されます。

  現在のコンテナがPDBである場合は、CONTAINER = CURRENTを指定すると、PDBで新しいマスター暗号化キーが作成およびアクティブ化されます。PDBでマスター暗号化キーを作成する前に、ルートにマスター暗号化キーが存在する必要があります。

  現在のコンテナがルートの場合は、CONTAINER = CURRENTを指定すると、ルートで新しいマスター暗号化キーが作成およびアクティブ化され、CONTAINER = ALLを指定すると、すべてのPDBで新しいマスター暗号化キーが作成およびアクティブ化されます。

  この句を省略すると、CONTAINER = CURRENTがデフォルトになります。

関連項目:

4. Oracle Database Advanced Securityガイド 統一モードでのキーストアおよびTDEマスター暗号化キーの管理
5. Oracle Database Advanced Securityガイド 分離モードでのキーストアおよびTDEマスター暗号化キーの管理
6. マスター暗号化キーの作成およびアクティブ化の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

create_key

MKID:MKオプションの指定の詳細は、set_key句のセマンティクスを参照してください。

この句を使用すると、後で使用するためのマスター暗号化キーを作成できます。その後で、use_key句を使用すると、このキーをアクティブ化できます。キーを含むキーストアは、パスワードで保護されたソフトウェア・キーストアであっても、ハードウェア・キーストアであってもかまいません。このキーストアは、開いている必要があります。

- ENCRYPTIONキーワードはオプションで、意味を明確化するために使用されます。
- オプションのUSING TAG句を指定すると、タグを暗号化キーに関連付けできます。詳細は、「USING TAG句のノート」を参照してください。
- USING ALGORITHM句を指定すると、指定された暗号化アルゴリズムに準拠するマスター暗号化キーが作成されます。encrypt_algorithmには、AES256、ARIA256、GOST256またはSEED128を指定できます。この句を指定するには、COMPATIBLE初期化パラメータが12.2以上に設定されている必要があります。この句を指定しない場合、AES256がデフォルトになります。

  ARIA、SEEDおよびGOSTアルゴリズムは、暗号化とハッシュについての国や政府の標準です。詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。
- FORCE KEYSTORE句を使用すると、キーストアが閉じている場合でも、この操作が可能になります。詳細は、「FORCE KEYSTORE句のノート」を参照してください。

- IDENTIFIED BY句を使用すると、キーを作成するキーストアのパスワードを指定できます。詳細は、「キーストア・パスワードを指定するときのノート」を参照してください。
- WITH BACKUP句とオプションのUSING 'backup_identifier'句を指定すると、キーが作成される前に、キーストアのバックアップが作成されるようになります。詳細は、「WITH BACKUP句のノート」を参照してください。
- CONTAINER句は、CDBに接続しているときに適用されます。

  現在のコンテナがPDBである場合は、CONTAINER = CURRENTを指定すると、PDBでマスター暗号化キーが作成されます。PDBでマスター暗号化キーを作成する前に、ルートにマスター暗号化キーが存在する必要があります。

  現在のコンテナがルートである場合は、CONTAINER = CURRENTを指定すると、ルートでマスター暗号化キーが作成され、CONTAINER = ALLを指定すると、ルートとすべてのPDBでマスター暗号化キーが作成されます。

  この句を省略すると、CONTAINER = CURRENTがデフォルトになります。

関連項目:

後で使用するためのマスター暗号化キーの作成方法の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

use_key

この句を使用して作成済のマスター暗号化キーをアクティブ化できます。この句を使用するときに、マスター暗号化キーがアクティブな場合は、そのキーが非アクティブ化されてから新しいマスター暗号化キーがアクティブ化されます。キーを含むキーストアは、パスワードベースのソフトウェア・キーストアであっても、ハードウェア・キーストアであってもかまいません。このキーストアは、開いている必要があります。

- ENCRYPTIONキーワードはオプションで、意味を明確化するために使用されます。
- key_idには、アクティブ化するキーの識別子を指定します。キー識別子は、V$ENCRYPTION_KEYSビューのKEY_ID列を問い合せることで確認できます。
- オプションのUSING TAG句を指定すると、タグを暗号化キーに関連付けできます。詳細は、「USING TAG句のノート」を参照してください。
- FORCE KEYSTORE句を使用すると、キーストアが閉じている場合でも、この操作が可能になります。詳細は、「FORCE KEYSTORE句のノート」を参照してください。
- IDENTIFIED BY句を使用すると、キーを含むキーストアのパスワードを指定できます。詳細は、「キーストア・パスワードを指定するときのノート」を参照してください。
- WITH BACKUP句とオプションのUSING 'backup_identifier'句を指定すると、キーがアクティブ化される前に、キーストアのバックアップを作成するようになります。詳細は、「WITH BACKUP句のノート」を参照してください。

関連項目:

マスター暗号化キーのアクティブ化の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

set_key_tag

この句を使用してに指定された暗号化キーにタグを設定できます。タグはオプションで、キーに関するユーザー定義の記述子です。キーにタグがない場合は、この句を使用してタグを作成できます。キーにタグがある場合は、この句を使用して置き換えること

ができます。暗号化キーのタグは、V$ENCRYPTION_KEYSビューのTAG列を問い合せることで表示できます。このキーストアは、開いている必要があります。

- tagには、英数字の文字列を指定します。tagは、一重引用符(' ')で囲みます。
- key_idには、暗号化キーの識別子を指定します。キー識別子は、V$ENCRYPTION_KEYSビューのKEY_ID列を問い合せることで確認できます。
- FORCE KEYSTORE句を使用すると、キーストアが閉じている場合でも、この操作が可能になります。詳細は、「FORCE KEYSTORE句のノート」を参照してください。
- IDENTIFIED BY句を使用すると、キーを含むキーストアのパスワードを指定できます。詳細は、「キーストア・パスワードを指定するときのノート」を参照してください。
- WITH BACKUP句とオプションのUSING 'backup_identifier'句を指定すると、キーがアクティブ化される前に、キーストアのバックアップが作成されます。詳細は、「WITH BACKUP句のノート」を参照してください。

関連項目:

キー・タグの設定の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

export_keys

この句を使用して、1つ以上の暗号化キーを、パスワードで保護されたソフトウェア・キーストアからファイルにエクスポートします。このキーストアは、開いている必要があります。各暗号化キーは、キー識別子およびキー属性とともにエクスポートされます。エクスポートされたキーは、ファイル内でパスワード(シークレット)によって保護されます。その後に、import_keys句を使用して1つ以上のキーをパスワードで保護されたソフトウェア・キーストアにインポートできます。

- ENCRYPTIONキーワードはオプションで、意味を明確化するために使用されます。
- secretを指定して、ファイル内のキーを保護するパスワード(シークレット)を設定します。シークレットは英数字の文字列です。シークレットは、二重引用符で囲むこともできます。引用符の有無にかかわらず、シークレットでは大文字と小文字が区別されます。
- filenameには、キーのエクスポート先となるファイルのフルパス名を指定します。filenameは一重引用符で囲みます。
- FORCE KEYSTORE句を使用すると、キーストアが閉じている場合でも、この操作が可能になります。詳細は、「FORCE KEYSTORE句のノート」を参照してください。
- IDENTIFIED BY句を使用すると、エクスポートするキーを含むキーストアのパスワードを指定できます。詳細は、「WITH BACKUP句のノート」を参照してください。
- 次のいずれかの方法により、WITH IDENTIFIER IN句を使用して、エクスポートする1つ以上の暗号化キーを指定します。
    - key_idを使用して、エクスポートする暗号化キーの識別子を指定します。key_idを複数指定する場合は、カンマ区切りのリストにします。キー識別子は、V$ENCRYPTION_KEYSビューのKEY_ID列を問い合せることで確認できます。
    - subqueryを使用して、エクスポートする暗号化キーのキー識別子のリストを戻す問合せを指定できます。たとえば、次のsubqueryは、タグがmytagという文字列で始まるデータベース内のすべての暗号化キーのキー識別子を戻します。

```
SELECT KEY_ID FROM V$ENCRYPTION_KEYS WHERE TAG LIKE 'mytag%'
```

Oracle Databaseは、subqueryを定義者の権限ではなく現在のユーザーの権限で実行することに注意してください。

- WITH IDENTIFIER IN句を省略すると、データベース内のすべての暗号化キーがエクスポートされます。

WITH IDENTIFIER IN句の制限事項

マルチテナント環境では、PDBからキーをエクスポートする際にWITH IDENTIFIER INを指定できません。これによって、PDB内のすべてのキーが、アクティブな暗号化キーに関するメタデータとともに確実にエクスポートされます。その後、PDBのクローンを作成する場合や、PDBを切断して接続する場合には、エクスポート・ファイルを使用して、クローンとして作成したPDBまたは新たに接続したPDBにキーをインポートし、アクティブな暗号化キーに関する情報を保存できます。

関連項目:

暗号化キーのエクスポートの詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

import_keys

この句を使用して、1つ以上の暗号化キーを、ファイルからパスワードベースのソフトウェア・キーストアにインポートします。このキーストアは、開いている必要があります。各暗号化キーは、キー識別子およびキー属性とともにインポートされます。キーは、事前にexport_keys句を使用して、ファイルにエクスポートしておく必要があります。キーストアにインポートされているキーは再インポートできません。

- ENCRYPTIONキーワードはオプションで、意味を明確化するために使用されます。
- secretには、ファイル内のキーを保護するパスワード(シークレット)を指定します。シークレットは英数字の文字列です。シークレットは、二重引用符で囲むこともできます。引用符の有無にかかわらず、シークレットでは大文字と小文字が区別されます。
- filenameには、キーのインポート元となるファイルのフルパス名を指定します。filenameは一重引用符で囲みます。
- FORCE KEYSTORE句を使用すると、キーストアが閉じている場合でも、この操作が可能になります。詳細は、「FORCE KEYSTORE句のノート」を参照してください。
- IDENTIFIED BY句を使用すると、キーをインポートするキーストアのパスワードを指定できます。詳細は、「WITH BACKUP句のノート」を参照してください。
- WITH BACKUP句とオプションのUSING 'backup_identifier'句を指定すると、キーがインポートされる前に、キーストアのバックアップが作成されます。詳細は、「WITH BACKUP句のノート」を参照してください。

関連項目:

暗号化キーのインポートの詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

migrate_key

この句を使用して、パスワードで保護されたソフトウェア・キーストアからハードウェア・キーストアへ移行します。この句によって、既存の表暗号化キーおよび表領域暗号化キーが、ソフトウェア・キーストア内のマスター暗号化キーを使用して復号化され、次にハードウェア・キーストア内の新規作成されたマスター暗号化キーを使用して再び暗号化されます。

ノート:

この句の使用は、パスワードで保護されたソフトウェア・キーストアからハードウェア・キーストアへの移行のための一連のステップのうちの 1 ステップにすぎません。この句を使用する前に、すべてのステップについて、『Oracle Database Advanced Security ガイド』を参照してください。

- ENCRYPTIONキーワードはオプションで、意味を明確化するために使用されます。
- HSM_auth_stringには、ハードウェア・キーストアのパスワードを指定します。詳細は、「キーストア・パスワードを指定するときのノート」を参照してください。
- FORCE KEYSTORE句を使用すると、キーストアが閉じている場合でも、この操作が可能になります。詳細は、「FORCE KEYSTORE句のノート」を参照してください。
- software_keystore_passwordには、パスワードベースのソフトウェア・キーストアのパスワードを指定します。詳細は、「キーストア・パスワードを指定するときのノート」を参照してください。
- WITH BACKUP句とオプションのUSING 'backup_identifier'句を指定すると、移行が行われる前に、キーストアのバックアップが作成されます。詳細は、「WITH BACKUP句のノート」を参照してください。

reverse_migrate_key

この句を使用して、ハードウェア・キーストアからパスワードで保護されたソフトウェア・キーストアへ移行します。この句によって、既存の表暗号化キーおよび表領域暗号化キーが、ハードウェア・キーストア内のマスター暗号化キーを使用して復号化され、次にパスワードで保護されたソフトウェア・キーストア内の新規作成されたマスター暗号化キーを使用して再び暗号化されます。

ノート:

この句の使用は、ハードウェア・キーストアからパスワードで保護されたソフトウェア・キーストアへの移行のための一連のステップのうちの 1 ステップにすぎません。この句を使用する前に、すべてのステップについて、『Oracle Database Advanced Security ガイド』を参照してください。

- ENCRYPTIONキーワードはオプションで、意味を明確化するために使用されます。
- software_keystore_passwordには、パスワードベースのソフトウェア・キーストアのパスワードを指定します。詳細は、「キーストア・パスワードを指定するときのノート」を参照してください。
- FORCE KEYSTORE句を使用すると、キーストアが閉じている場合でも、この操作が可能になります。詳細は、「FORCE KEYSTORE句のノート」を参照してください。
- HSM_auth_stringには、ハードウェア・キーストアのパスワードを指定します。詳細は、「キーストア・パスワードを指定するときのノート」を参照してください。

move_keys

move_keys句を使用して、暗号化キーを新しいキーストアに移動します。データベースにログインするには、ADMINISTER KEY MANAGEMENT権限またはSYSKM権限を持つユーザーである必要があります。キーの移動先にするキーストアのキー識別子を見つけるには、V$ENCRYPTION_KEYSビューのKEY_ID列を問い合せる必要があります。

keystore_location1は、新しいキーストア.p12ファイルを格納するウォレットのディレクトリへのパスです。デフォルトでは、

このディレクトリは$ORACLE_BASE/admin/db_unique_name/wallet内にあります。

keystore1_passwordは、新しいキーストアのパスワードです。

keystore_passwordは、キーの移動元のキーストアのパスワードです。

key_identifierは、V$ENCRYPTION_KEYSビューのKEY_ID列を問い合せて確認するキーの識別子です。この設定は一重引用符(' ')で囲みます。

subqueryは、目的の正確なキー識別子を検索するために使用できます。

backup_identifierはバックアップの説明(オプション)です。backup_identifierは、一重引用符(' ')で囲みます。

たとえば:

```
ADMINISTER KEY MANAGEMENT MOVE KEYS
TO NEW KEYSTORE $ORACLE_BASE/admin/orcl/wallet
IDENTIFIED BY keystore_password
FROM FORCE KEYSTORE
IDENTIFIED BY keystore_password
WITH IDENTIFIER IN
(SELECT KEY_ID FROM V$ENCRYPTION_KEYS WHERE ROWNUM < 2);
```

secret_management_clauses

この句を使用して、パスワードで保護されたソフトウェア・キーストアやハードウェア・キーストア内のシークレットを追加、更新および削除できます。

関連項目:

シークレットの追加、更新および削除の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

add_update_secret

この句を使用すると、キーストアにシークレットを追加したり、キーストア内の既存のシークレットを更新できます。このキーストアは、開いている必要があります。

- キーストアにシークレットを追加するには、ADD を指定します。
- キーストア内の既存のシークレットを更新するには、UPDATEを指定します。
- secretには、追加または更新するシークレットを指定します。シークレットは英数字の文字列です。シークレットは一重引用符で囲みます。
- client_identifierには、シークレットの識別に使用する英数字の文字列を指定します。client_identifierは一重引用符で囲みます。この値では大文字と小文字が区別されます。次の固定値のいずれかを入力できます。
  - キーストアがFILEとして構成されている場合は、TDE_WALLET
  - キーストアがOracle Key Vault HSM用の場合は、OKV_WALLET
  - キーストアがサード・パーティのHSM用の場合は、HSM_WALLET
- オプションのUSING TAG句を指定すると、secretにタグを関連付けできます。tagはオプションで、シークレットに関するユーザー定義の記述子です。タグは一重引用符で囲みます。シークレットのタグは、V$CLIENT_SECRETSビューのSECRET_TAG列を問い合せることで確認できます。
- FORCE KEYSTORE句を使用すると、キーストアが閉じている場合でも、この操作が可能になります。詳細は、

「FORCE KEYSTORE句のノート」を参照してください。

- IDENTIFIED BY句を使用すると、キーストアのパスワードを指定できます。詳細は、「キーストア・パスワードを指定するときのノート」を参照してください。
- WITH BACKUP句とオプションのUSING 'backup_identifier'句を指定すると、パスワードベースのソフトウェア・キーストアのシークレットが追加または更新される前に、キーストアのバックアップが作成されます。詳細は、「WITH BACKUP句のノート」を参照してください。

delete_secret

この句を使用すると、キーストアからシークレットを削除できます。このキーストアは、開いている必要があります。

4. client_identifierには、シークレットの識別に使用する英数字の文字列を指定します。client_identifierは一重引用符で囲みます。V$CLIENT_SECRETSビューのCLIENT列を問い合せることで、クライアント識別子を表示できます。
5. FORCE KEYSTORE句を使用すると、キーストアが閉じている場合でも、この操作が可能になります。詳細は、「FORCE KEYSTORE句のノート」を参照してください。
6. IDENTIFIED BY句を使用すると、キーストアのパスワードを指定できます。詳細は、「キーストア・パスワードを指定するときのノート」を参照してください。
7. WITH BACKUP句とオプションのUSING 'backup_identifier'句を指定すると、パスワードベースのソフトウェア・キーストアからシークレットが削除される前に、キーストアのバックアップが作成されます。詳細は、「WITH BACKUP句のノート」を参照してください。

USING TAG句のノート

多くのADMINISTER KEY MANAGEMENT操作にはUSING TAG句が含まれ、これを使用すると、タグを暗号化キーに関連付けることができます。tagはオプションで、キーに関するユーザー定義の記述子です。これは一重引用符で囲まれた文字列です。

暗号化キーのタグは、V$ENCRYPTION_KEYSビューのTAG列を問い合せることで表示できます。

FORCE KEYSTORE句のノート

自動ログイン・ウォレットが存在する場合、FORCE KEYSTORE句は、キーストアが閉じている場合でもキーストア操作を有効にします。この句の動作は、非CDB、CDBルートまたはPDBのいずれに接続されているかによって異なります。

- 非CDBに接続されている場合は、次のようになります。
  - パスワードで保護されたソフトウェア・キーストアまたはハードウェア・キーストアが閉じている場合は、操作の実行中にデータベースによってパスワードで保護されたソフトウェア・キーストアまたはハードウェア・キーストアが開かれ、開いた状態が維持されます。その後、自動ログイン・キーストアが存在する場合はそのキーストアが新しい情報で更新されます。
  - 自動ログイン・キーストアが開いている場合は、操作の実行中にデータベースによってパスワードで保護されたソフトウェア・キーストアまたはハードウェア・キーストアが一時的に開かれ、自動ログイン・キーストアから切り替わることなく、自動ログイン・キーストアが新しい情報で更新されます。
  - パスワードで保護されたソフトウェア・キーストアまたはハードウェア・キーストアが開いている場合、FORCE KEYSTORE句は必要なく、かつ影響を及ぼしません。
- CDBルートに接続されている場合は、次のようになります。

- CDBルート・キーストアに対する操作を実行するには(CONTAINER=CURRENT)、CDBルート・キーストアが開いている必要があります。したがって、非CDBについて説明した動作がCDBルートに適用されます。
- CDBルート・キーストアおよびすべてのPDBキーストアに対する操作を実行するには(CONTAINER=ALL)、CDBルート・キーストアとすべてのPDBキーストアが開いている必要があります。したがって、非CDBについて説明した動作がCDBルートおよび各PDBに適用されます。

- PDBに接続されている場合は、次のようになります。
  - PDBキーストアに対する操作を実行するには、CDBルート・キーストアと該当するPDBのキーストアが開いている必要があります。したがって、非CDBについて説明した動作がCDBルートおよび該当するPDBに適用されます。

キーストア・パスワードを指定するときのノート

キーストア・パスワードは、次のように指定します。

- パスワードで保護されたソフトウェア・キーストアの場合は、文字列としてパスワードを指定します。パスワードを二重引用符で囲むこともできます。引用符の有無にかかわらず、パスワードは大文字と小文字が区別されます。キーストアのパスワードはデータベース・ユーザーのパスワードと同じ規則に従います。詳細は、「CREATE USER」の「BY password」 句を参照してください。
- ハードウェア・キーストアの場合は、"user_id:password"の形式の文字列としてパスワードを指定します。
  - user_idは、HSM管理インタフェースを使用してデータベース用に作成されたユーザーIDです。
  - passwordは、HSM管理インタフェースを使用してユーザーIDに対して作成されたパスワードです。

  user_id:passwordの文字列は二重引用符(" ")囲み、user_idとpasswordはコロン(:)で区切ります。
- EXTERNAL STOREを指定する場合、データベースでは、外部ストアに格納されているキーストア・パスワードを使用して操作を実行します。この機能により、集中的に管理およびアクセスできる独立した場所にパスワードを格納できるようになります。この機能を使用するには、最初に、キーストア・パスワードを格納する場所にEXTERNAL_KEYSTORE_CREDENTIAL_LOCATION初期化パラメータを設定する必要があります。キーストア・パスワードに外部ストアを構成する方法の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

WITH BACKUP句のノート

多くのADMINISTER KEY MANAGEMENT操作には、WITH BACKUP句が含まれます。この句は、パスワードで保護されたソフトウェア・キーストアにのみ適用されます。これは、この操作を実行する前に、キーストアをバックアップする必要があることを意味しています。このため、操作を実行するときにWITH BACKUP句を指定するか、操作を実行する直前にADMINISTER KEY MANAGEMENT backup_clause文を発行する必要があります。

WITH BACKUP句を指定すると、Oracle Databaseは、ewallet_timestamp.p12という形式の名前でバックアップ・ファイルを作成します。timestampは、UTC形式でのファイル作成時のタイムスタンプです。バックアップ・ファイルは、キーストアをバックアップしているディレクトリに作成されます。

オプションのUSING 'backup_identifier'句を使用すると、バックアップ・ファイルの名前に追加するバックアップ識別子を指定できます。たとえば、バックアップ識別子に'Backup1'を指定すると、Oracle Databaseは、ewallet_timestamp_Backup1.p12という形式の名前でバックアップ・ファイルを作成します。

WITH BACKUPは、パスワードで保護されたソフトウェア・キーストアの場合は必須ですが、ハードウェア・キーストアの場合はオプションです。

add_update_secret_seps

SEPSウォレットとも呼ばれる安全性の高い外部パスワード・ストア(SEPS)でキーを管理するには、この句を指定します。この句のセマンティクスは、add_update_secret句と同じです。

delete_secret_seps

SEPSウォレットとも呼ばれる安全性の高い外部パスワード・ストア(SEPS)でキーを削除するには、この句を指定します。この句のセマンティクスは、delete_secret句と同じです。

例

キーストアの作成: 例

次の文は、/etc/ORACLE/WALLETS/orclディレクトリ内にパスワードで保護されたソフトウェア・キーストアを作成します。

```
ADMINISTER KEY MANAGEMENT
  CREATE KEYSTORE '/etc/ORACLE/WALLETS/orcl'
  IDENTIFIED BY password;
```

次の文は、前の文で作成されたキーストアから自動ログイン・ソフトウェア・キーストアを作成します。

```
ADMINISTER KEY MANAGEMENT
  CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE '/etc/ORACLE/WALLETS/orcl'
  IDENTIFIED BY password;
```

キーストアを開く: 例

次の文は、パスワードで保護されたソフトウェア・キーストアを開きます。

```
ADMINISTER KEY MANAGEMENT
  SET KEYSTORE OPEN
  IDENTIFIED BY password;
```

CDBに接続している場合、次の文は現在のコンテナ内のパスワードで保護されたソフトウェア・キーストアを開きます。

```
ADMINISTER KEY MANAGEMENT
  SET KEYSTORE OPEN
  IDENTIFIED BY password
  CONTAINER = CURRENT;
```

次の文はハードウェア・キーストアを開きます。

```
ADMINISTER KEY MANAGEMENT
  SET KEYSTORE OPEN
  IDENTIFIED BY "user_id:password";
```

次の文は、パスワードが外部ストアに格納されているキーストアを開きます。

```
ADMINISTER KEY MANAGEMENT
  SET KEYSTORE OPEN
  IDENTIFIED BY EXTERNAL STORE;
```

キーストアを閉じる: 例

次の文は、パスワードで保護されたソフトウェア・キーストアを閉じます。

```
ADMINISTER KEY MANAGEMENT
  SET KEYSTORE CLOSE
  IDENTIFIED BY password;
```

次の文は、自動ログイン・ソフトウェア・キーストアを閉じます。

```
ADMINISTER KEY MANAGEMENT
  SET KEYSTORE CLOSE;
```

次の文は、ハードウェア・キーストアを閉じます。

```
ADMINISTER KEY MANAGEMENT
  SET KEYSTORE CLOSE
  IDENTIFIED BY "user_id:password";
```

次の文は、パスワードが外部ストアに格納されているキーストアを閉じます。

```
ADMINISTER KEY MANAGEMENT
  SET KEYSTORE CLOSE
  IDENTIFIED BY EXTERNAL STORE;
```

キーストアのバックアップ: 例

次の文は、パスワードで保護されたソフトウェア・キーストアのバックアップを作成します。バックアップは、`/etc/ORACLE/KEYSTORE/DB1`ディレクトリに格納され、バックアップ・ファイル名にタグ`hr.emp_keystore`が含まれます。

```
ADMINISTER KEY MANAGEMENT
  BACKUP KEYSTORE USING 'hr.emp_keystore'
  IDENTIFIED BY password
  TO '/etc/ORACLE/KEYSTORE/DB1/';
```

キーストア・パスワードの変更: 例

次の文は、パスワードで保護されたソフトウェア・キーストアのパスワードを変更します。パスワードを変更する前に、キーストアのバックアップもタグ`pwd_change`付きで作成します。

```
ADMINISTER KEY MANAGEMENT
  ALTER KEYSTORE PASSWORD IDENTIFIED BY old_password
  SET new_password WITH BACKUP USING 'pwd_change';
```

2つのキーストアの新しいキーストアへのマージ: 例

次の文は、自動ログイン・ソフトウェア・キーストアをパスワードで保護されたソフトウェア・キーストアとマージして、パスワードで保護された新しいソフトウェア・キーストアを新しい場所に作成します。

```
ADMINISTER KEY MANAGEMENT
  MERGE KEYSTORE '/etc/ORACLE/KEYSTORE/DB1'
  AND KEYSTORE '/etc/ORACLE/KEYSTORE/DB2'
    IDENTIFIED BY existing_keystore_password
  INTO NEW KEYSTORE '/etc/ORACLE/KEYSTORE/DB3'
    IDENTIFIED BY new_keystore_password;
```

キーストアの既存のキーストアへのマージ: 例

次の文は、自動ログイン・ソフトウェア・キーストアをパスワードで保護されたソフトウェア・キーストアにマージします。マージを実行する前に、パスワードで保護されたソフトウェア・キーストアのバックアップも作成します。

```
ADMINISTER KEY MANAGEMENT
  MERGE KEYSTORE '/etc/ORACLE/KEYSTORE/DB1'
  INTO EXISTING KEYSTORE '/etc/ORACLE/KEYSTORE/DB2'
    IDENTIFIED BY existing_keystore_password
  WITH BACKUP;
```

マスター暗号化キーの作成およびアクティブ化: 例

次の文は、パスワードで保護されたソフトウェア・キーストア内のマスター暗号化キーを作成し、アクティブ化します。これにより、

SEED128アルゴリズムを使用してキーが暗号化されます。新しいマスター暗号化キーを作成する前に、キーストアのバックアップも作成します。

```
ADMINISTER KEY MANAGEMENT
  SET KEY USING ALGORITHM 'SEED128'
  IDENTIFIED BY password
  WITH BACKUP;
```

次の文は、パスワードで保護されたソフトウェア・キーストア内にマスター暗号化キーを作成しますが、そのキーをアクティブ化しません。新しいマスター暗号化キーを作成する前に、キーストアのバックアップも作成します。

```
ADMINISTER KEY MANAGEMENT
  CREATE KEY USING TAG 'mykey1'
  IDENTIFIED BY password
  WITH BACKUP;
```

次の問合せは、前の文で作成されたマスター暗号化キーのキー識別子を表示します。

```
SELECT TAG, KEY_ID
  FROM V$ENCRYPTION_KEYS
  WHERE TAG = 'mykey1';
TAG     KEY_ID
---     ------------------------------------------------------
mykey1  ARgEtzPxpE/Nv8WdPu8LJJUAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

次の文は、前の文で問合せのあったマスター暗号化キーをアクティブ化します。新しいマスター暗号化キーをアクティブ化する前に、キーストアのバックアップも作成します。

```
ADMINISTER KEY MANAGEMENT
  USE KEY 'ARgEtzPxpE/Nv8WdPu8LJJUAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
  IDENTIFIED BY password
  WITH BACKUP;
```

キー・タグの設定: 例

この例では、キーストアが閉じていることを想定しています。次の文は、キーストアを一時的に開き、前の例でアクティブ化されたマスター暗号化キーについて、タグをmykey2に変更します。タグを変更する前に、キーストアのバックアップも作成します。

```
ADMINISTER KEY MANAGEMENT
  SET TAG 'mykey2' FOR 'ARgEtzPxpE/Nv8WdPu8LJJUAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
  FORCE KEYSTORE
  IDENTIFIED BY password
  WITH BACKUP;
```

キーのエクスポート: 例

次の文は、パスワードで保護されたソフトウェア・キーストアから/etc/TDE/export.expファイルに2つのマスター暗号化キーをエクスポートします。この文は、シークレットmy_secretを使用してそのファイルの内のマスター暗号化キーを暗号化します。エクスポートされるマスター暗号化キーの識別子は、カンマ区切りのリストとして提供されます。

```
ADMINISTER KEY MANAGEMENT
  EXPORT KEYS WITH SECRET "my_secret"
  TO '/etc/TDE/export.exp'
  IDENTIFIED BY password
  WITH IDENTIFIER IN 'AdoxnJ0uH08cv7xkz83ovwsAAAAAAAAAAAAAAAAAAAAAAAAAAAA',
                     'AW5z3CoyKE/yv3cNT5CWCXUAAAAAAAAAAAAAAAAAAAAAAAAAAAA';
```

次の文は、パスワードで保護されたソフトウェア・キーストアから/etc/TDE/export.expファイルにマスター暗号化キーをエクスポートします。タグがmytag1またはmytag2であるキーのみがエクスポートされます。ファイル内のマスター・キーは、シークレットmy_secretを使用して暗号化されます。キー識別子は、V$ENCRYPTION_KEYSビューに問い合せて見つけます。

```
ADMINISTER KEY MANAGEMENT
  EXPORT KEYS WITH SECRET "my_secret"
  TO '/etc/TDE/export.exp'
  IDENTIFIED BY password
  WITH IDENTIFIER IN
    (SELECT KEY_ID FROM V$ENCRYPTION_KEYS WHERE TAG IN ('mytag1', 'mytag2'));
```

次の文は、データベースのすべてのマスター暗号化キーを/etc/TDE/export.expファイルにエクスポートします。ファイル内のマスター・キーは、シークレットmy_secretを使用して暗号化されます。

```
ADMINISTER KEY MANAGEMENT
  EXPORT KEYS WITH SECRET "my_secret"
  TO '/etc/TDE/export.exp'
  IDENTIFIED BY password;
```

マルチテナント環境で、次の文は、PDB salespdbのすべてのマスター暗号化キーをメタデータとともに/etc/TDE/salespdb.expファイルにエクスポートします。ファイル内のマスター・キーは、シークレットmy_secretを使用して暗号化されます。その後、PDBのクローンを作成する場合や、PDBを切断して再接続する場合には、この文で作成されたエクスポート・ファイルを使用して、クローンとして作成したPDB、または新たに接続したPDBにキーをインポートできます。

```
ALTER SESSION SET CONTAINER = salespdb;
ADMINISTER KEY MANAGEMENT
  EXPORT KEYS WITH SECRET "my_secret"
  TO '/etc/TDE/salespdb.exp'
  IDENTIFIED BY password;
```

キーのインポート: 例

次の文は、/etc/TDE/export.expファイルからパスワードで保護されたソフトウェア・キーストアに、シークレットmy_secretで暗号化されたマスター暗号化キーをインポートします。キーをインポートする前に、パスワードで保護されたソフトウェア・キーストアのバックアップも作成します。

```
ADMINISTER KEY MANAGEMENT
  IMPORT KEYS WITH SECRET "my_secret"
  FROM '/etc/TDE/export.exp'
  IDENTIFIED BY password
  WITH BACKUP;
```

キーストアの移行: 例

次の文は、パスワードで保護されたソフトウェア・キーストアからハードウェア・キーストアに移行します。移行を実行する前に、パスワードで保護されたソフトウェア・キーストアのバックアップも作成します。

```
ADMINISTER KEY MANAGEMENT
  SET ENCRYPTION KEY IDENTIFIED BY "user_id:password"
  MIGRATE USING software_keystore_password
  WITH BACKUP;
```

キーストアの逆移行: 例

次の文は、ハードウェア・キーストアからパスワードで保護されたソフトウェア・キーストアに逆移行します。

```
ADMINISTER KEY MANAGEMENT
  SET ENCRYPTION KEY IDENTIFIED BY software_keystore_password
  REVERSE MIGRATE USING "user_id:password";
```

キーストアへのシークレットの追加: 例

次の文は、クライアントclient1のシークレットsecret1をタグMy first secret付きでパスワードで保護されたソフトウェア・キーストアに追加します。シークレットを追加する前に、パスワードで保護されたソフトウェア・キーストアのバックアップも作成し

ます。

```
ADMINISTER KEY MANAGEMENT
  ADD SECRET 'secret1' FOR CLIENT 'client1'
  USING TAG 'My first secret'
  IDENTIFIED BY password
  WITH BACKUP;
```

次の文は、類似したシークレットをハードウェア・キーストアに追加します。

```
ADMINISTER KEY MANAGEMENT
  ADD SECRET 'secret2' FOR CLIENT 'client2'
  USING TAG 'My second secret'
  IDENTIFIED BY "user_id:password";
```

キーストア内のシークレットの更新: 例

次の文は、前の例でパスワードベースのソフトウェア・キーストア内に作成されたシークレットを更新します。シークレットを更新する前に、パスワードで保護されたソフトウェア・キーストアのバックアップも作成します。

```
ADMINISTER KEY MANAGEMENT
  UPDATE SECRET 'secret1' FOR CLIENT 'client1'
  USING TAG 'New Tag 1'
  IDENTIFIED BY password
  WITH BACKUP;
```

次の文は、前の例でハードウェア・キーストア内に作成されたシークレットを更新します。

```
ADMINISTER KEY MANAGEMENT
  UPDATE SECRET 'secret2' FOR CLIENT 'client2'
  USING TAG 'New Tag 2'
  IDENTIFIED BY "user_id:password";
```

キーストアからのシークレットの削除: 例

次の文は、前の例で更新されたシークレットをパスワードで保護されたソフトウェア・キーストアから削除します。シークレットを削除する前に、パスワードで保護されたソフトウェア・キーストアのバックアップも作成します。

```
ADMINISTER KEY MANAGEMENT
  DELETE SECRET FOR CLIENT 'client1'
  IDENTIFIED BY password
  WITH BACKUP;
```

次の文は、前の例で更新されたシークレットをハードウェア・キーストアから削除します。

```
ADMINISTER KEY MANAGEMENT
  DELETE SECRET FOR CLIENT 'client2'
  IDENTIFIED BY "user_id:password";
```

ALTER ANALYTIC VIEW

目的

ALTER ANALYTIC VIEW文を使用すると、分析ビューの名前を変更したり、分析ビューをコンパイルすることができます。その他の変更には、CREATE OR REPLACE ANALYTIC VIEWを使用してください。

前提条件

自分のスキーマ内で分析ビューを変更する場合は、ALTER ANALYTIC VIEWシステム権限が必要です。他のユーザーのスキーマ内で分析ビューを変更する場合は、ALTER ANY ANALYTIC VIEWシステム権限を持っているか、ALTER ANY TABLEが分析ビューに付与されている必要があります。

構文

alter_analytic_view::=

セマンティクス

schema

分析ビューが存在するスキーマを指定します。スキーマを指定しない場合、自分のスキーマ内で分析ビューが検索されます。

analytic_view_name

分析ビューの名前を指定します。

RENAME TO

RENAME TOを指定すると、分析ビューの名前を変更できます。new_av_nameの場合は、分析ビューの新しい名前を指定します。

COMPILE

COMPILEを指定すると、分析ビューをコンパイルできます。

例

次の文は、分析ビューの名前を変更します。

```
ALTER ANALYTIC VIEW sales_av RENAME TO mysales_av;
```

ALTER ATTRIBUTE DIMENSION

目的

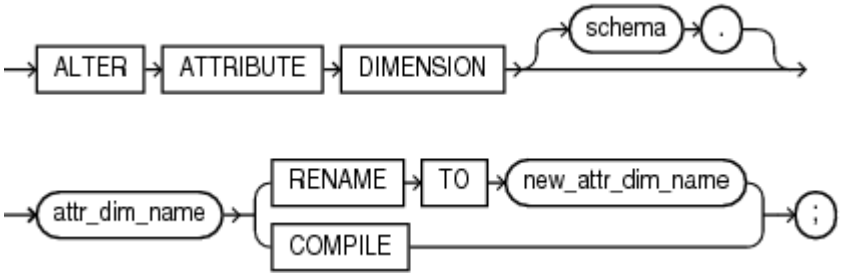
ALTER ATTRIBUTE DIMENSION文を使用して、属性ディメンションの名前を変更したり、属性ディメンションをコンパイルすることができます。その他の変更には、CREATE OR REPLACE ATTRIBUTE DIMENSIONを使用します。

前提条件

自分のスキーマ内で属性ディメンションを変更する場合は、ALTER ATTRIBUTE DIMENSIONシステム権限が必要です。別のユーザーのスキーマ内で属性ディメンションを変更する場合は、ALTER ANY ATTRIBUTE DIMENSIONシステム権限を持っているか、属性ディメンションに直接ALTERが付与されている必要があります。

構文

alter_attribute_dimension::=



セマンティクス

schema

属性ディメンションが存在するスキーマを指定します。スキーマを指定しない場合、自分のスキーマ内で属性ディメンションが検索されます。

attr_dim_name

属性ディメンションの名前を指定します。

RENAME TO

RENAME TOを指定すると、属性ディメンションの名前を変更できます。new_attr_dim_nameの場合は、属性ディメンションの新しい名前を指定します。

COMPILE

COMPILEを指定すると、属性ディメンションをコンパイルできます。

例

次の文は、属性ディメンションの名前を変更します。

```
ALTER ATTRIBUTE DIMENSION product_attr_dim RENAME TO my_product_attr_dim;
```

ALTER AUDIT POLICY (統合監査)

このセクションでは、統合監査のためのALTER AUDIT POLICY文について説明します。この種類の監査は、Oracle Database 12cで新たに導入されたもので、完全かつ高度な監査機能を提供します。統合監査の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

目的

ALTER AUDIT POLICY文を使用すると、統合監査ポリシーを変更できます。

関連項目:

- CREATE AUDIT POLICY (統合監査)
- DROP AUDIT POLICY (統合監査)
- AUDIT (統合監査)
- NOAUDIT (統合監査)

前提条件

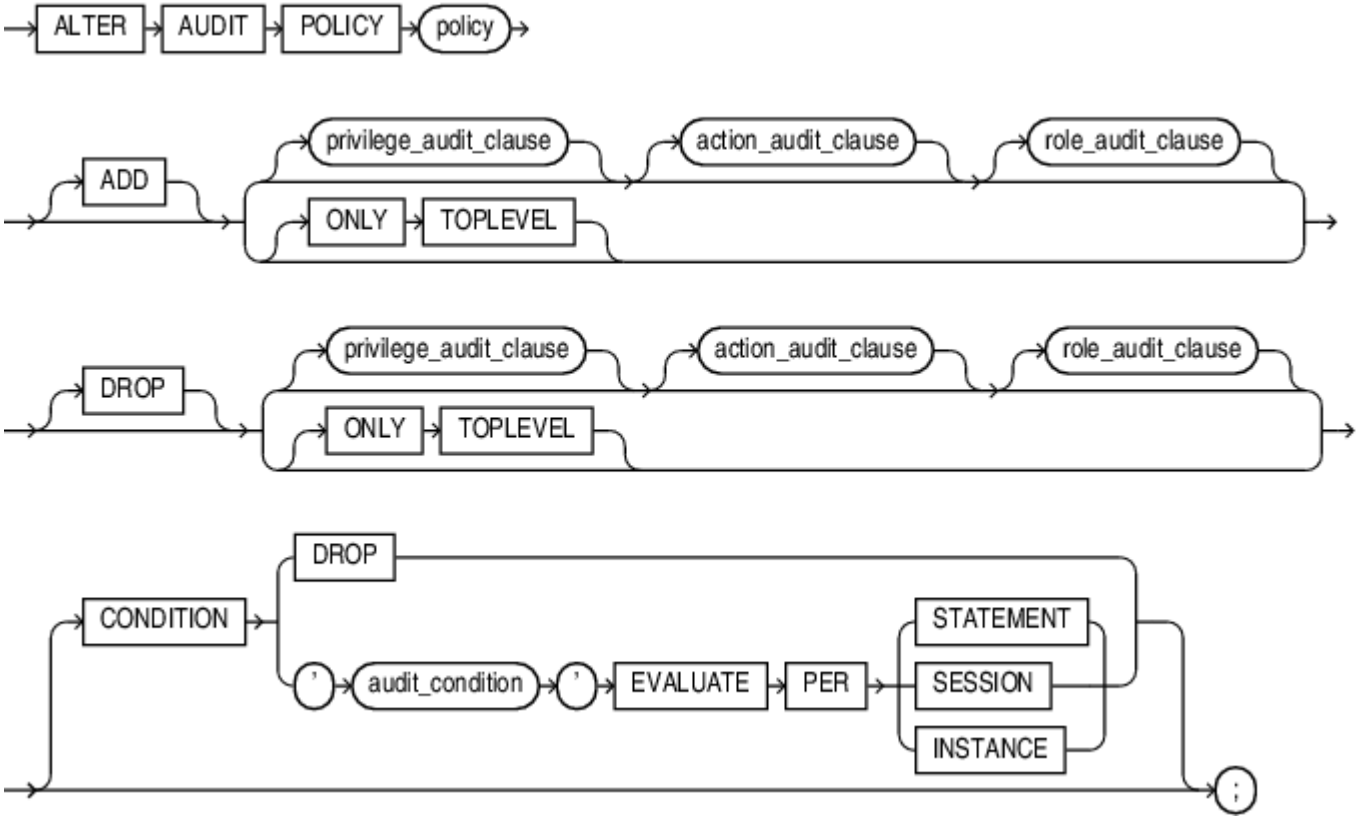
AUDIT SYSTEMシステム権限、またはAUDIT_ADMINロールが必要になります。

マルチテナント・コンテナ・データベース(CDB)に接続している場合、共通の統合監査ポリシーを変更するには、現在のコンテナがルートである必要があります。また、共通に付与されているAUDIT SYSTEM権限またはAUDIT_ADMIN共通ロールが必要です。ローカルの統合監査ポリシーを変更するには、現在のコンテナが、その監査ポリシーが作成されたコンテナである必要があります。また、共通に付与されているAUDIT SYSTEM権限またはAUDIT_ADMIN共通ロールを保有しているか、そのコンテナでローカルに付与されているAUDIT SYSTEM権限またはAUDIT_ADMINローカル・ロールを保有している必要があります。

オブジェクト監査オプションを使用して統合監査ポリシーを変更すると、アクティブおよび後続の両方のユーザー・セッションに対して、新しい監査設定が即時に実行されます。システム監査オプションまたは監査条件を使用して統合監査ポリシーを変更すると、現在のユーザー・セッションではなく、新しいユーザー・セッションに対してのみ有効になります。

構文

alter_audit_policy::=

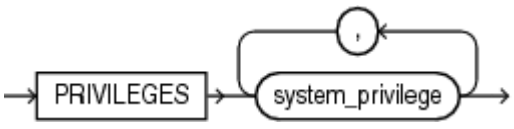


ノート:

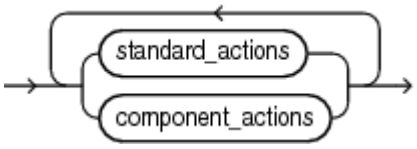
ADD 句または DROP 句を指定するときには、少なくとも `privilege_audit_clause` 句、`action_audit_clause` 句、または `role_audit_clause` 句のいずれか 1 つを指定する必要があります。

(privilege_audit_clause::=、action_audit_clause::=、role_audit_clause::=)

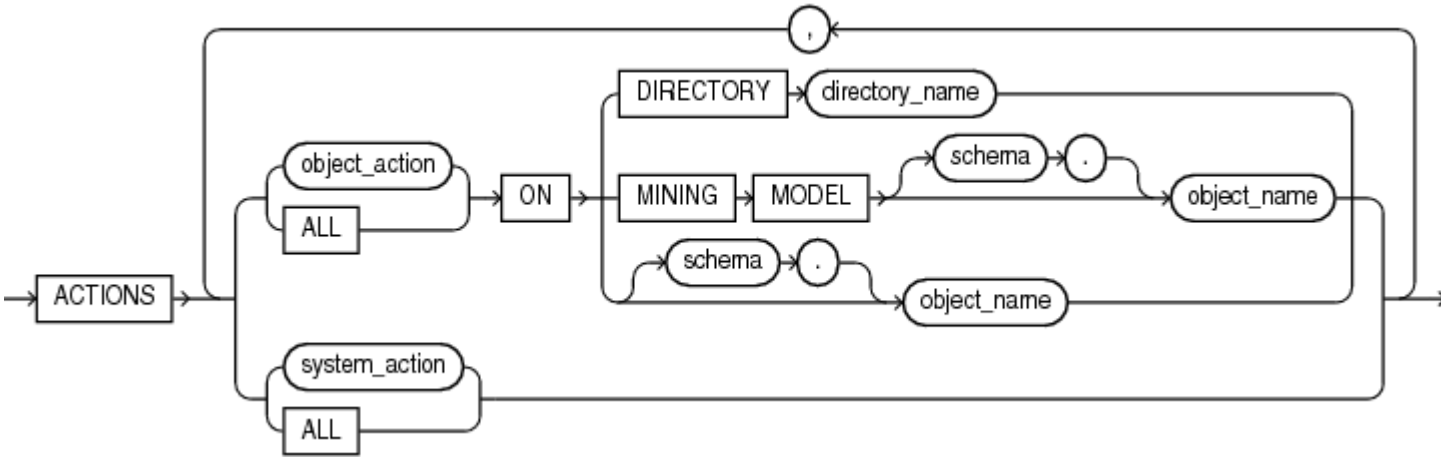
privilege_audit_clause::=



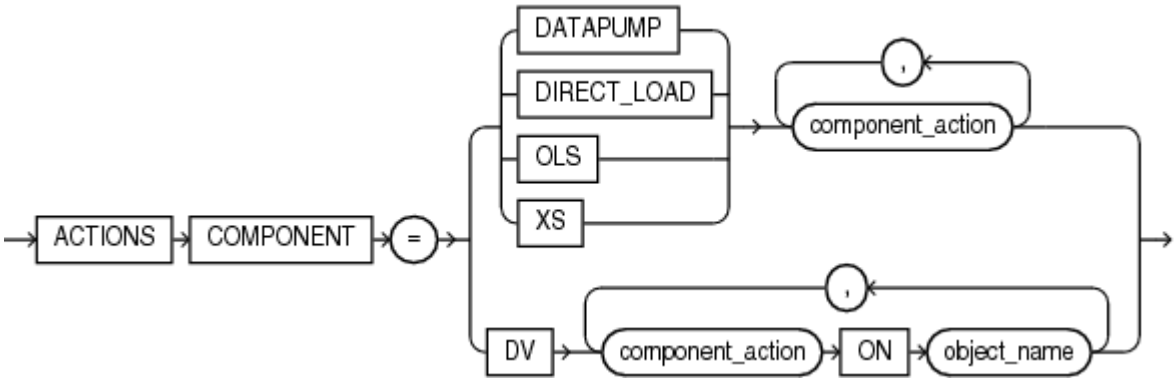
action_audit_clause::=



standard_actions::=



component_actions::=



role_audit_clause::=

セマンティクス

policy

変更する統合監査ポリシーの名前を指定します。このポリシーは、CREATE AUDIT POLICY文を使用して作成されている必要があります。すべての監査ポリシーの説明を確認するには、AUDIT_UNIFIED_POLICIESビューを問い合せます。

関連項目:

- CREATE AUDIT POLICY (統合監査)
- AUDIT_UNIFIED_POLICIESビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

ADD | DROP

ADD句を使用すると、監査対象になる権限をpolicyに追加できます。

DROP句を使用すると、監査対象になる権限をpolicyから削除できます。

これらの句のセマンティクスの詳細は、CREATE AUDIT POLICYの「privilege_audit_clause」、「action_audit_clause」、および「role_audit_clause」を参照してください。

CONDITION

この句を使用すると、policyの監査条件を削除、追加または置換できます。

DROPを指定すると、policyから監査条件を削除できます。

policyの監査条件を追加または置換するには、'audit_condition' ... を指定します。

これらの句のセマンティクスの詳細は、CREATE AUDIT POLICYの「audit_condition」、「EVALUATE PER STATEMENT」、「EVALUATE PER SESSION」、および「EVALUATE PER INSTANCE」を参照してください。

ONLY TOPLEVEL

既存の統合監査ポリシーを変更して、ユーザーが発行したトップ・レベルのSQL文のみを監査する場合に、この句を指定します。

例: トップ・レベル監査の追加

この例は、HR監査ポリシーhr_audit_policyを変更して、トップ・レベルの文のみを取得します。

```
ALTER AUDIT POLICY hr_audit_policy ADD ONLY TOPLEVEL
```

トップ・レベルSQL文を監査する既存の監査ポリシーから、トップ・レベル監査を削除できます。

例: トップ・レベル監査の削除

```
ALTER AUDIT POLICY hr_audit_policy DROP ONLY TOPLEVEL
```

詳細は、Oracle Databaseセキュリティ・ガイドを参照してください。

例

次の例では、CREATE AUDIT POLICYの「例」で作成した統合監査ポリシーを変更します。

統合監査ポリシーへの権限、アクションおよびロールの追加: 例

次の文では、システム権限CREATE ANY TABLEおよびDROP ANY TABLEを統合監査ポリシーdml_polに追加します。

```
ALTER AUDIT POLICY dml_pol
  ADD PRIVILEGES CREATE ANY TABLE, DROP ANY TABLE;
```

次の文では、システム・アクションのCREATE JAVA、ALTER JAVAおよびDROP JAVAを統合監査ポリシーjava_polに追加します。

```
ALTER AUDIT POLICY java_pol
  ADD ACTIONS CREATE JAVA, ALTER JAVA, DROP JAVA;
```

次の文では、ロールのdbaを統合監査ポリシーtable_polに追加します。

```
ALTER AUDIT POLICY table_pol
  ADD ROLES dba;
```

次の文では、複数のシステム権限、アクションおよびロールを統合監視ポリシーsecurity_polに追加します。

```
ALTER AUDIT POLICY security_pol
  ADD PRIVILEGES CREATE ANY LIBRARY, DROP ANY LIBRARY
      ACTIONS DELETE on hr.employees,
              INSERT on hr.employees,
              UPDATE on hr.employees,
              ALL on hr.departments
      ROLES dba, connect;
```

統合監査ポリシーからの権限、アクションおよびロールの削除: 例

次の文では、統合監査ポリシーtable_polからシステム権限CREATE ANY TABLEを削除します。

```
ALTER AUDIT POLICY table_pol
  DROP PRIVILEGES CREATE ANY TABLE;
```

次の文では、hr.employeesに対するINSERTおよびUPDATEアクションを統合監査ポリシーdml_polから削除します。

```
ALTER AUDIT POLICY dml_pol
  DROP ACTIONS INSERT on hr.employees,
               UPDATE on hr.employees;
```

次の文では、ロールのjava_deployを統合監査ポリシーjava_polから削除します。

```
ALTER AUDIT POLICY java_pol
  DROP ROLES java_deploy;
```

次の文では、システム権限、アクションおよびロールを統合監視ポリシーhr_admin_polから削除します。

```
ALTER AUDIT POLICY hr_admin_pol
  DROP PRIVILEGES CREATE ANY TABLE
       ACTIONS LOCK TABLE
       ROLES audit_viewer;
```

統合監査ポリシーのアクションの追加および削除: 例

次の文では、Oracle Data PumpのEXPORTアクションを統合監査ポリシーdp_actions_polに追加し、Oracle Data PumpのIMPORTアクションを削除します。

```
ALTER AUDIT POLICY dp_actions_pol
  ADD ACTIONS COMPONENT = datapump EXPORT
  DROP ACTIONS COMPONENT = datapump IMPORT;
```

統合監査ポリシーからの監査条件の削除: 例

次の文では、統合監査ポリシーorder_updates_polから監査条件を削除します。

```
ALTER AUDIT POLICY order_updates_pol
  CONDITION DROP;
```

統合監査ポリシーの監査条件の変更: 例

次の文では、統合監査ポリシーemp_updates_polの監査条件を変更して、UIDが102のユーザーが監査可能な文を発行したときにのみ、このポリシーを適用します。

```
ALTER AUDIT POLICY emp_updates_pol
  CONDITION 'UID = 102'
  EVALUATE PER STATEMENT;
```

ALTER CLUSTER

目的

ALTER CLUSTER文を使用すると、クラスタの記憶特性および並列特性を再定義できます。

ノート:

クラスタ・キーの列番号および列名を変更するためにこの文を使用することはできません。また、クラスタを格納する表領域を変更することはできません。

関連項目:

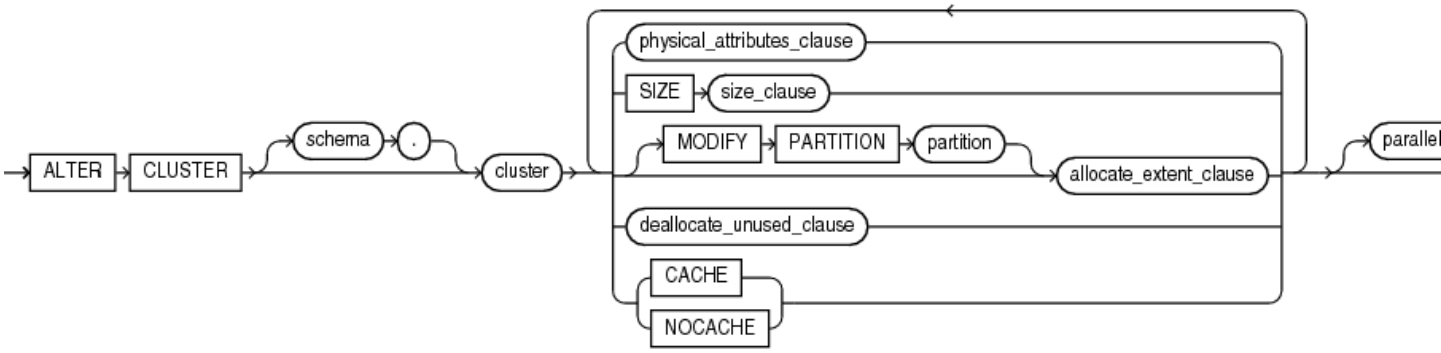
クラスタの作成については、「CREATE CLUSTER」を参照してください。クラスタからの表の削除については、「DROP CLUSTER」および「DROP TABLE」を参照してください。クラスタへの表の追加については、「CREATE TABLE」の「physical_properties」を参照してください。

前提条件

クラスタが自分のスキーマ内にあるか、またはALTER ANY CLUSTERシステム権限が必要です。

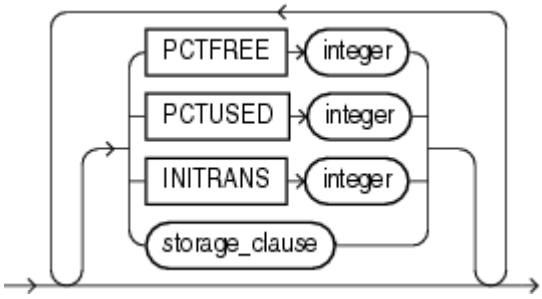
構文

alter_cluster::=



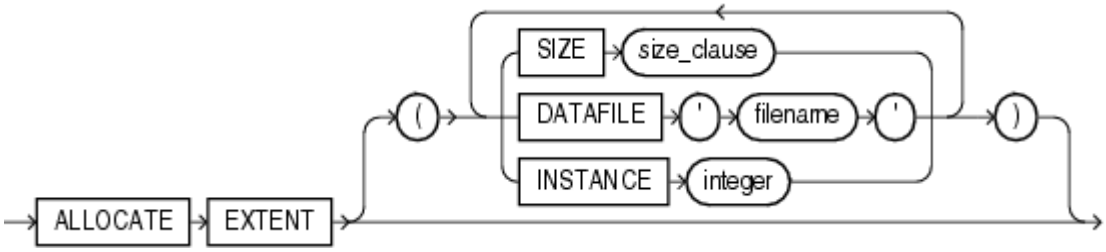
(physical_attributes_clause::、size_clause::=、MODIFY PARTITION、allocate_extent_clause::=、deallocate_unused_clause::=、parallel_clause::=)

physical_attributes_clause::



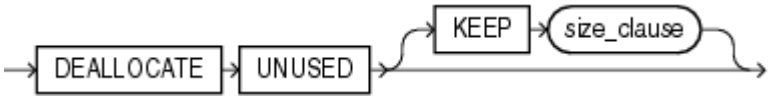
(storage_clause::=)

allocate_extent_clause::=



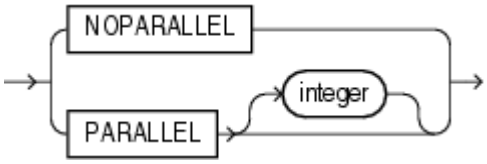
(size_clause::=)

deallocate_unused_clause::=



(size_clause::=)

parallel_clause::=



セマンティクス

schema

クラスタが含まれているスキーマを指定します。schemaを指定しない場合、クラスタは自分のスキーマ内にあるとみなされます。

cluster

変更するクラスタの名前を指定します。

physical_attributes_clause

クラスタのPCTUSEDパラメータ、PCTFREEパラメータおよびINITRANSパラメータの値を変更します。

クラスタの記憶特性を変更するには、STORAGE句を使用します。

関連項目:

- パラメータの詳細は、「physical_attributes_clause」を参照してください。
- その句の詳細は、「storage_clause」を参照してください。

物理属性の制限事項

クラスタの記憶域パラメータINITIALとMINEXTENTSの値は変更できません。

SIZE

integer

SIZE句を使用すると、クラスタに割り当てられたデータ・ブロック中に格納されるクラスタ・キーの数を指定できます。

SIZEの制限事項

ハッシュ・クラスタではなく、索引クラスタのSIZEパラメータのみを変更できます。

関連項目:

SIZEパラメータの詳細は、「CREATE CLUSTER」および「クラスタの変更: 例」を参照してください。

MODIFY PARTITION

MODIFY PARTITION partition allocate_extent_clauseを指定すると、クラスタ・パーティションの新しいエクステントを明示的に割り当てることができます。この操作は、レンジ・パーティション・ハッシュ・クラスタにのみ有効です。partitionには、クラスタ・パーティション名を指定します。

allocate_extent_clause

allocate_extent_clauseを指定すると、クラスタの新しいエクステントを明示的に割り当てることができます。この操作は、索引クラスタおよび非パーティション・ハッシュ・クラスタにのみ有効です。

allocate_extent_clauseを使用して明示的にエクステントを割り当てるときは、表を作成するときとは異なり、Oracle Databaseがクラスタの記憶域パラメータを評価して次に割り当てるエクステントの新しいサイズを決定することはありません。したがって、Oracle Databaseのデフォルト値が使用されるのを避けるにはSIZEを指定してください。

関連項目:

この句の詳細は、「allocate_extent_clause」を参照してください。

deallocate_unused_clause

deallocate_unused_clause句を使用すると、クラスタの終わりの未使用領域の割当てを明示的に解除し、解放された領域が他のセグメントで使用可能になります。

関連項目:

この句の詳細は、「deallocate_unused_clause」および「未使用領域の割当て解除: 例」を参照してください。

parallel_clause

parallel_clauseを指定すると、クラスタの問合せのデフォルト並列度を変更できます。

関連項目:

この句の詳細は、「CREATE TABLE」の「parallel_clause」を参照してください

例

次に、CREATE CLUSTERの「例」で作成したクラスタを変更する例を示します。

クラスタの変更: 例

次の文は、personnelクラスタを変更します。

```
ALTER CLUSTER personnel
   SIZE 1024 CACHE;
```

この結果、各クラスタ・キー値に1024バイトが割り当てられ、CACHE属性が有効になります。データ・ブロックのサイズを2KBと想定した場合、このクラスタ内の今後のデータ・ブロックには、各ブロックに2つのクラスタ・キー(2KBを1024バイトで割った値)が含まれます。

未使用領域の割当て解除: 例

次の文は、languageクラスタから未使用領域の割当てを解除し、後で使用できるように30KBの未使用領域を保持します。

```
ALTER CLUSTER language
   DEALLOCATE UNUSED KEEP 30 K;
```

クラスタの変更: 例

次の文は、デフォルトのキー・サイズ(600)を持つクラスタを作成します。

```
CREATE CLUSTER EMP_DEPT (DEPTNO NUMBER(3))
   SIZE 600
   TABLESPACE USERS
   STORAGE (INITIAL 200K
      NEXT 300K
      MINEXTENTS 2
      PCTINCREASE 33);
```

次の文は、USER_CLUSTERSを問い合せてクラスタ・メタデータを表示します。

```
SELECT CLUSTER_NAME, TABLESPACE_NAME, KEY_SIZE, CLUSTER_TYPE, AVG_BLOCKS_PER_KEY,
MIN_EXTENTS, MAX_EXTENTS FROM USER_CLUSTERS;
CLUSTER_NAME    TABLESPACE_NAME                  KEY_SIZE CLUST AVG_BLOCKS_PER_KEY
MIN_EXTENTS MAX_EXTENTS
--------------- ------------------------------ ---------- ----- ------------------ --
--------- -----------
EMP_DEPT        USERS                                 600 INDEX
1  2147483645
```

次の文は、クラスタ・キー・サイズを変更します。

```
ALTER CLUSTER EMP_DEPT SIZE 1024;
```

次の文は、変更されたクラスタのメタデータを表示します。

```
SELECT CLUSTER_NAME, TABLESPACE_NAME, KEY_SIZE, CLUSTER_TYPE, AVG_BLOCKS_PER_KEY,
MIN_EXTENTS, MAX_EXTENTS FROM USER_CLUSTERS;
CLUSTER_NAME    TABLESPACE_NAME                  KEY_SIZE CLUST AVG_BLOCKS_PER_KEY
MIN_EXTENTS MAX_EXTENTS
--------------- ------------------------------ ---------- ----- ------------------ --
--------- -----------
EMP_DEPT        USERS                                1024 INDEX
1  2147483645
```

次の文は、EMP_DEPTクラスタから未使用領域の割当てを解除し、後で使用できるように30KBの未使用領域を保持します。

```
ALTER CLUSTER EMP_DEPT DEALLOCATE UNUSED KEEP 30 K;
```

次の文は、変更されたクラスタのメタデータを表示します。

```
SELECT CLUSTER_NAME, TABLESPACE_NAME, KEY_SIZE, CLUSTER_TYPE, AVG_BLOCKS_PER_KEY,
MIN_EXTENTS, MAX_EXTENTS FROM USER_CLUSTERS;
```

```
CLUSTER_NAME    TABLESPACE_NAME                  KEY_SIZE CLUST AVG_BLOCKS_PER_KEY
MIN_EXTENTS MAX_EXTENTS
--------------- ------------------------------ ---------- ----- ------------------ --
--------- -----------
EMP_DEPT        USERS                                1024 INDEX
1  2147483645
```

Live SQL:

クラスタの作成と変更で Oracle Live SQL に関連する例を参照および実行します。

ALTER DATABASE

目的

ALTER DATABASE文を使用すると、既存のデータベースを変更、メンテナンスまたはリカバリできます。

関連項目:

- メディア・リカバリの実行例については、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- スタンバイ・データベースをメンテナンスするためにALTER DATABASE文を使用する場合の詳細は、『Oracle Data Guard概要および管理』を参照してください。
- データベースの作成の詳細は、「CREATE DATABASE」を参照してください。

前提条件

ALTER DATABASEシステム権限が必要です。

`startup_clauses`を指定するには、AS SYSDBA、AS SYSOPER、AS SYSBACKUPまたはAS SYSDGとして接続している必要もあります。

`general_recovery`句を指定する場合は、SYSDBAまたはSYSBACKUPシステム権限も必要です。

`DEFAULT EDITION`句を指定するには、指定されたエディションのUSEオブジェクト権限の`WITH GRANT OPTION`も必要です。

マルチテナント・コンテナ・データベース(CDB)に接続しているときには、次の条件が適用されます。

- CDB全体を変更する場合は、現在のコンテナがルートである必要があります。また、共通に付与されているALTER DATABASE権限が必要です。
- コンテナを変更する場合は、そのコンテナが現在のコンテナである必要があります。また、ALTER DATABASE権限(共通に付与されている権限か、そのコンテナでローカルに付与されている権限のいずれか)が必要です。

CDBでのALTER DATABASEの使用のノート

CDBに接続しているときにALTER DATABASE文を発行する場合、この文の動作は、現在のコンテナおよび指定した句によって変わります。

現在のコンテナがルートである場合、次の句を含むALTER DATABASE文を使用すると、CDB全体が変更されます。これらの句を指定するには、共通に付与されているALTER DATABASE権限が必要です。

- `startup_clauses`
- `recovery_clauses`

  ノート: 個々のプラガブル・データベース(PDB)のバックアップとリカバリ用に、`recovery_clauses`のサブセットがサポートされています。これらの句を指定するには、ALTER DATABASE権限(共通に付与されている権限か、そのPDBでローカルに付与されている権限のいずれか)が必要です。詳細は、「CDBでのrecovery_clausesの使用のノート」を参照してください。
- `logfile_clauses`
- `controlfile_clauses`
- `standby_database_clauses`

- instance_clauses
- security_clause
- RENAME GLOBAL_NAME TO
- ENABLE BLOCK CHANGE TRACKING
- DISABLE BLOCK CHANGE TRACKING
- undo_mode_clause

現在のコンテナがルートの場合、次の句を含むALTER DATABASE文を使用すると、ルートのみが変更されます。これらの句を指定するには、ALTER DATABASE権限(共通に付与されている権限か、そのルートでローカルに付与されている権限のいずれか)が必要です。

- database_file_clauses
- DEFAULT EDITION
- DEFAULT TABLESPACE

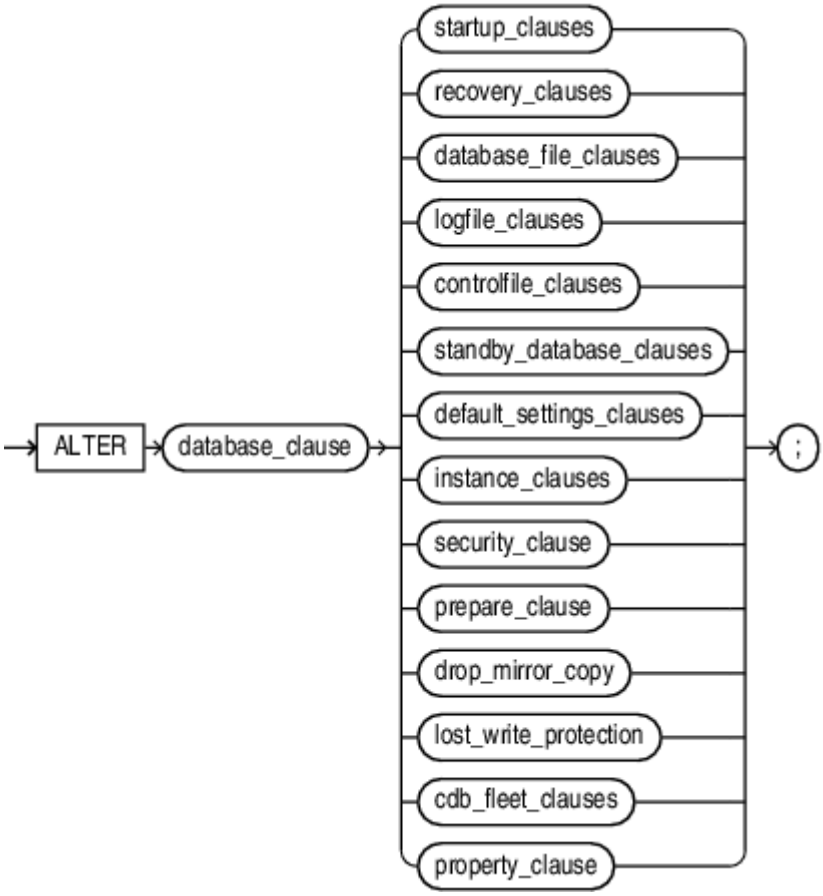
現在のコンテナがルートの場合、次の句を含むALTER DATABASE文を使用すると、ルートが変更され、PDBにデフォルト値が設定されます。これらの句を指定するには、共通に付与されているALTER DATABASE権限が必要です。

- DEFAULT [LOCAL] TEMPORARY TABLESPACE
- flashback_mode_clause
- SET DEFAULT { BIGFILE | SMALLFILE } TABLESPACE
- set_time_zone_clause

現在のコンテナがPDBである場合、ALTER DATABASE文を使用すると、そのPDBが変更されます。この場合、ALTER PLUGGABLE DATABASE文でもサポートされるALTER DATABASE句のみを発行できます。この機能は、CDBに移行したアプリケーションの下位互換性を維持するために用意されています。例外はPDBストレージの限度の変更です。これには、ALTER PLUGGABLE DATABASEのpdb_storage_clauseを使用する必要があります。これらの句の詳細は、ALTER PLUGGABLE DATABASEのドキュメントを参照してください。

構文

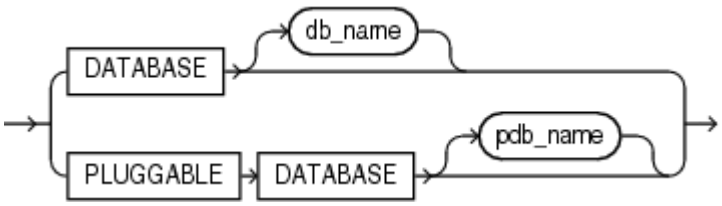
alter_database::=



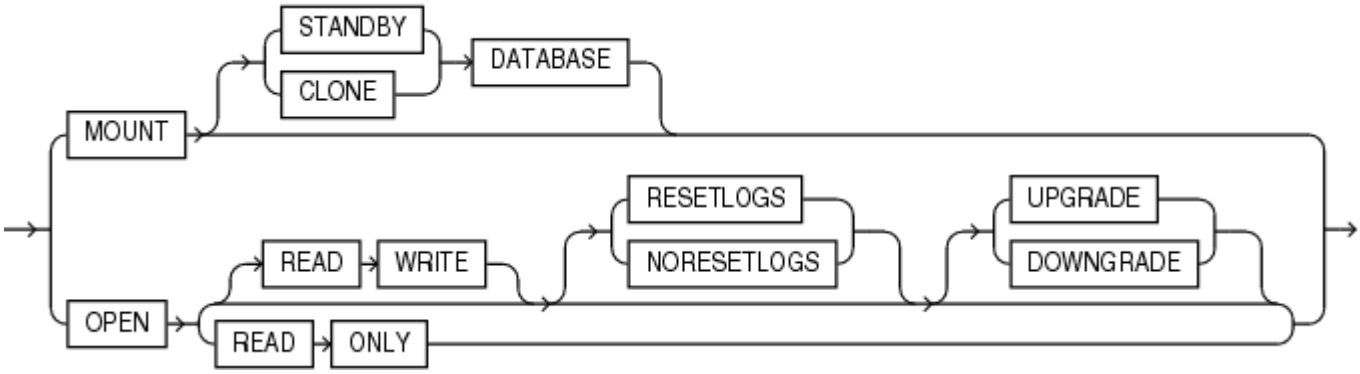
ALTER DATABASE構文のグループは、次のとおりです。

- startup_clauses::=
- recovery_clauses::=
- database_file_clauses::=
- logfile_clauses::=
- controlfile_clauses::=
- standby_database_clauses::=
- default_settings_clauses::=
- instance_clauses::=
- security_clause::=

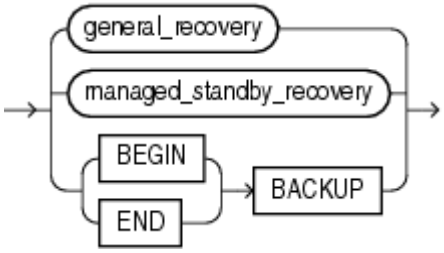
database_clause::=



startup_clauses::=

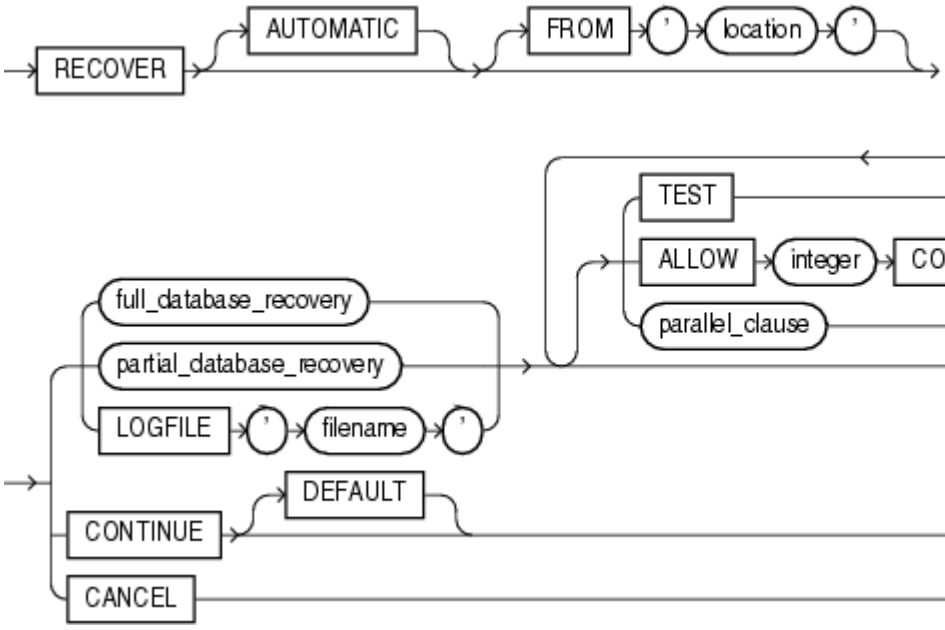


recovery_clauses::=



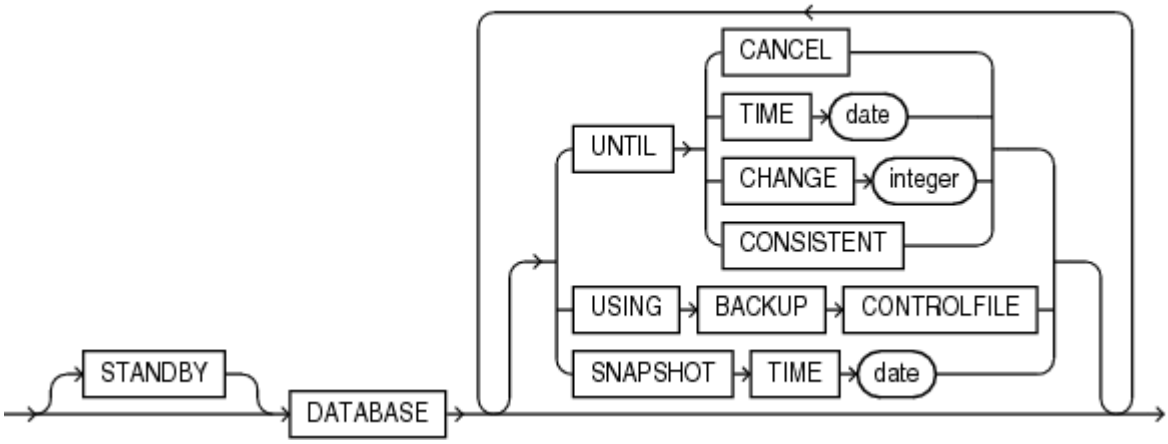
(general_recovery::=、managed_standby_recovery::=)

general_recovery::=

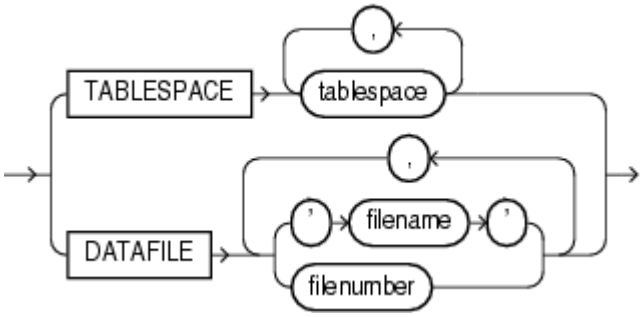


(full_database_recovery::=、partial_database_recovery::=、parallel_clause::=)

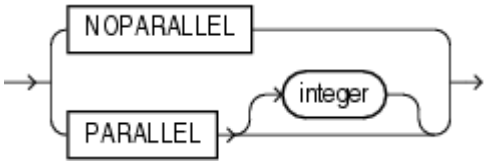
full_database_recovery::=



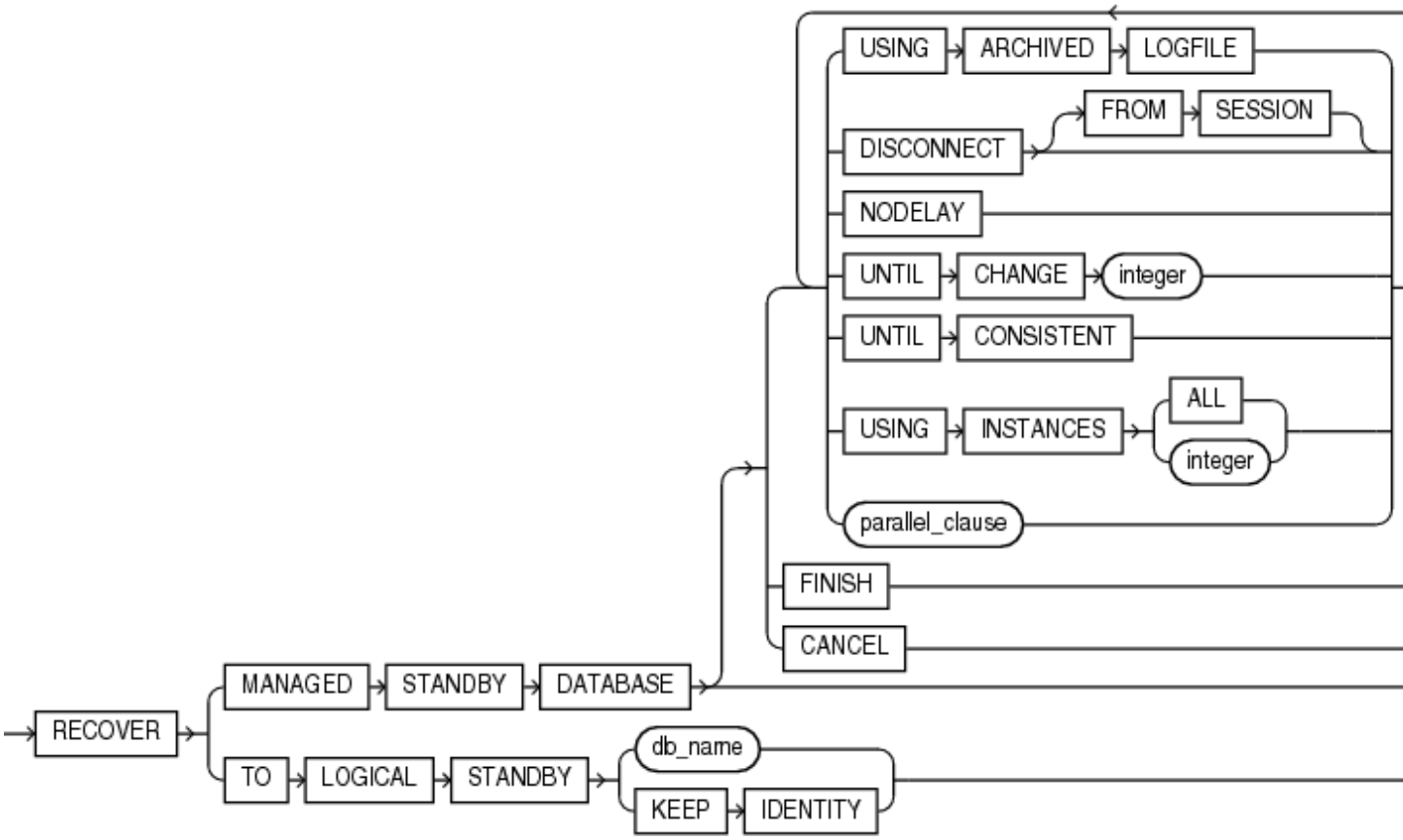
partial_database_recovery::=



parallel_clause::=



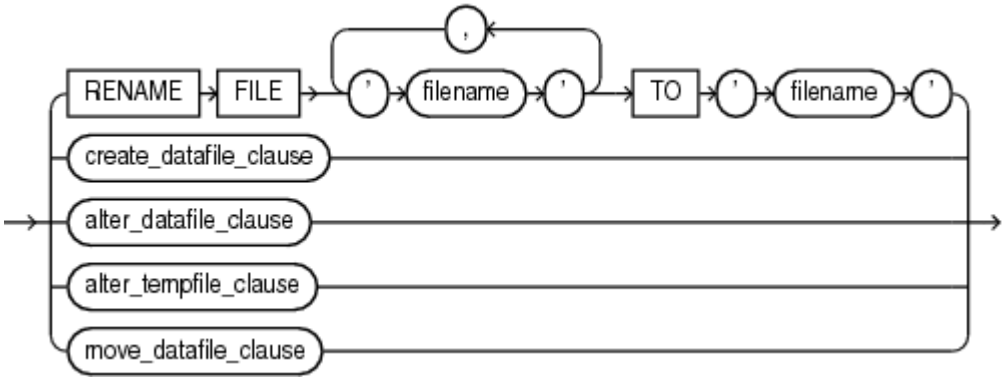
managed_standby_recovery::=



(parallel_clause::=)

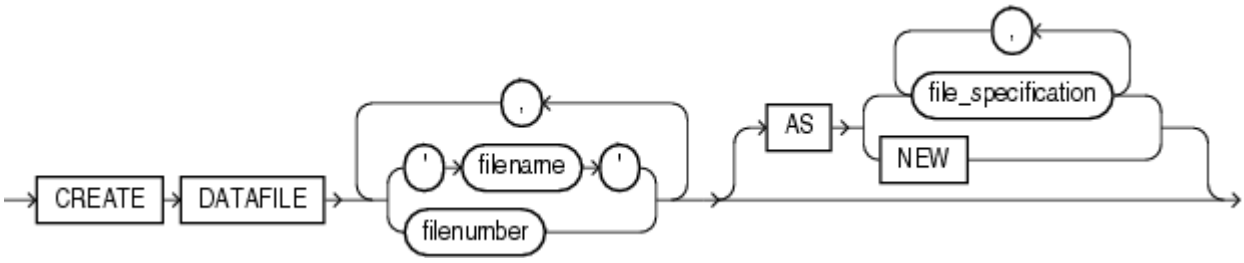
> ノート:
>
> managed_standby_recovery のいくつかの副次句は不要であり、その使用は非推奨になっています。これらの副次句は、構文図にも記載されていません。「managed_standby_recovery」のセマンティクスを参照してください。

database_file_clauses::=



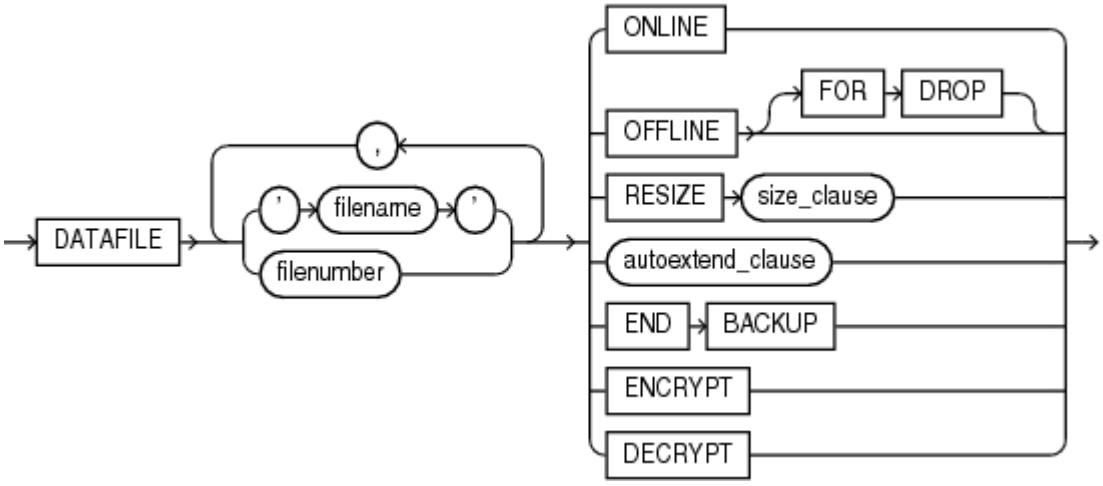
(create_datafile_clause::=、alter_datafile_clause::=、alter_tempfile_clause::=、move_datafile_clause::=)

create_datafile_clause::=



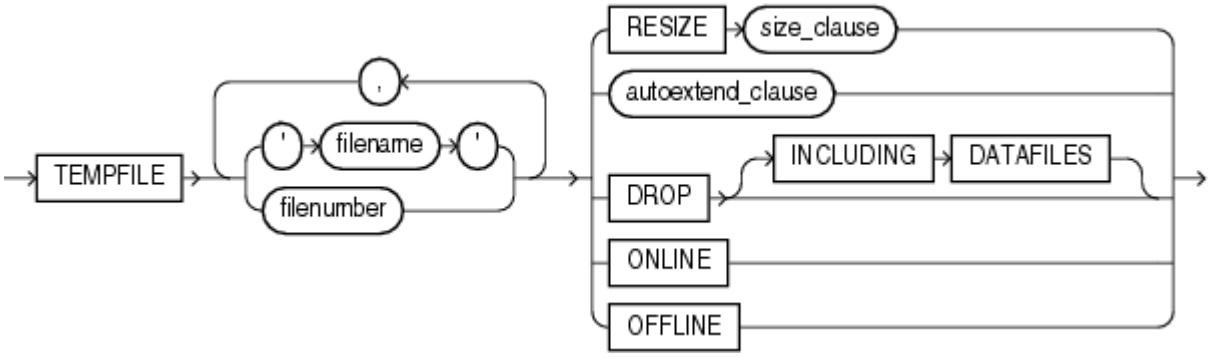
(file_specification::=)

alter_datafile_clause::=



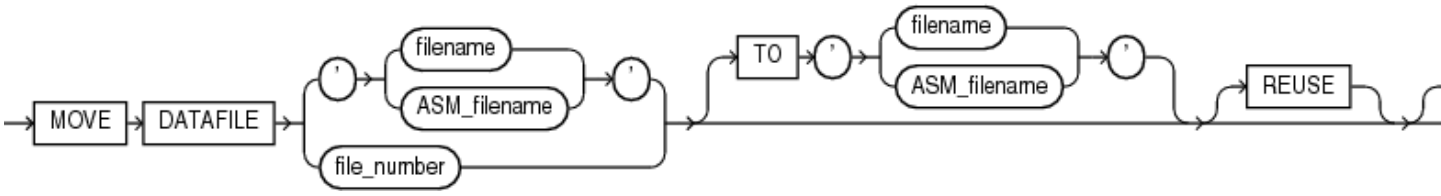
(autoextend_clause::=、size_clause::=)

alter_tempfile_clause::=

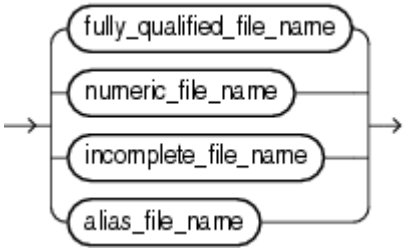


(autoextend_clause::=、size_clause::=)

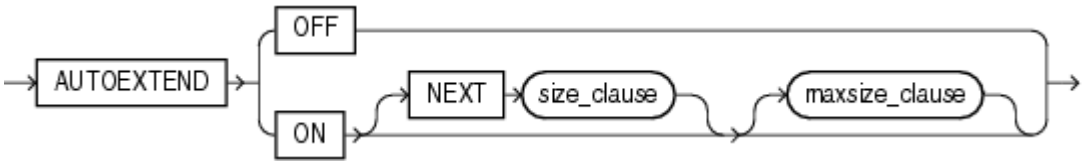
move_datafile_clause::=



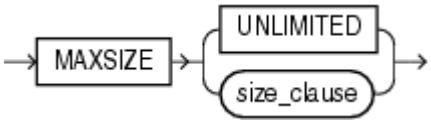
ASM_filename::=



autoextend_clause::=

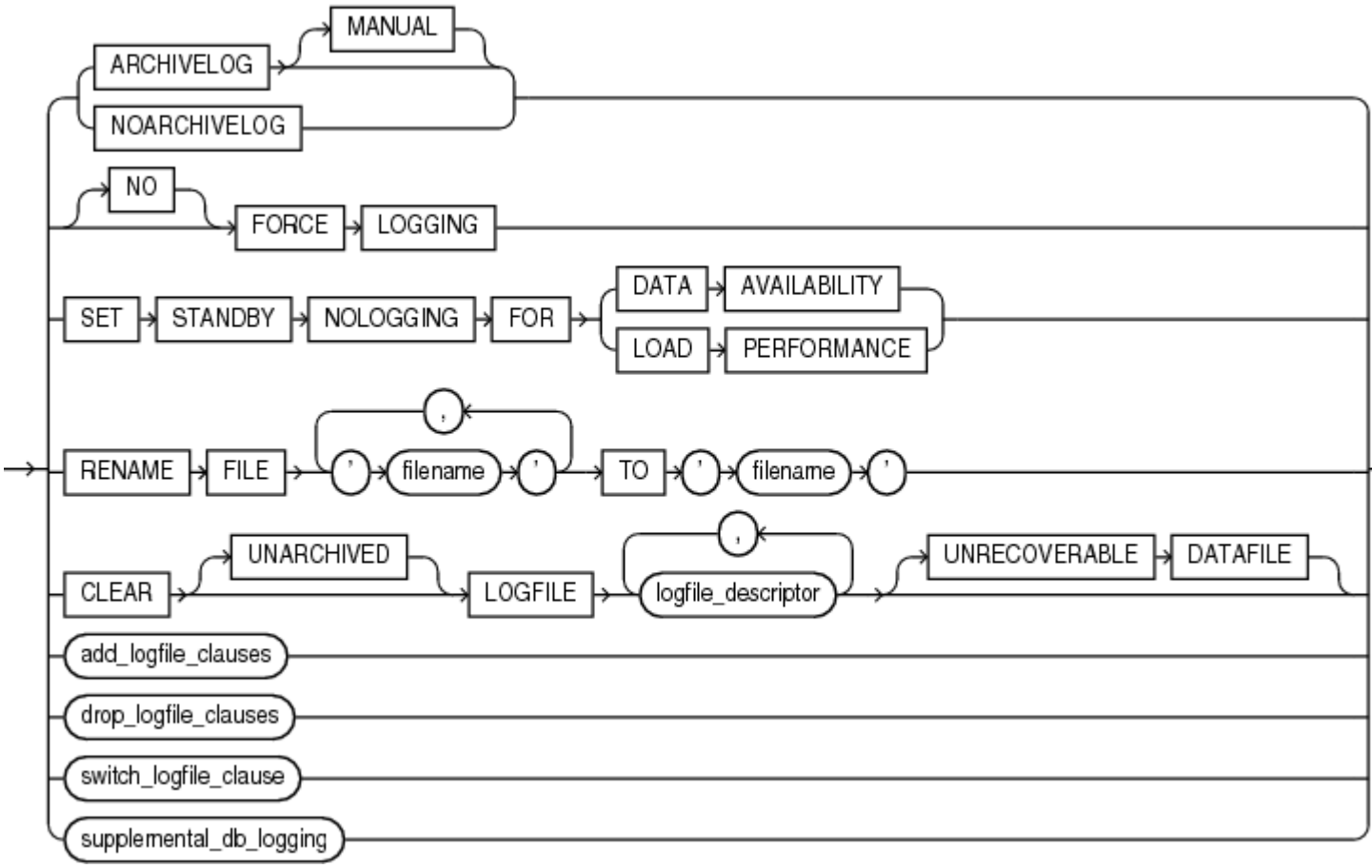


maxsize_clause::=



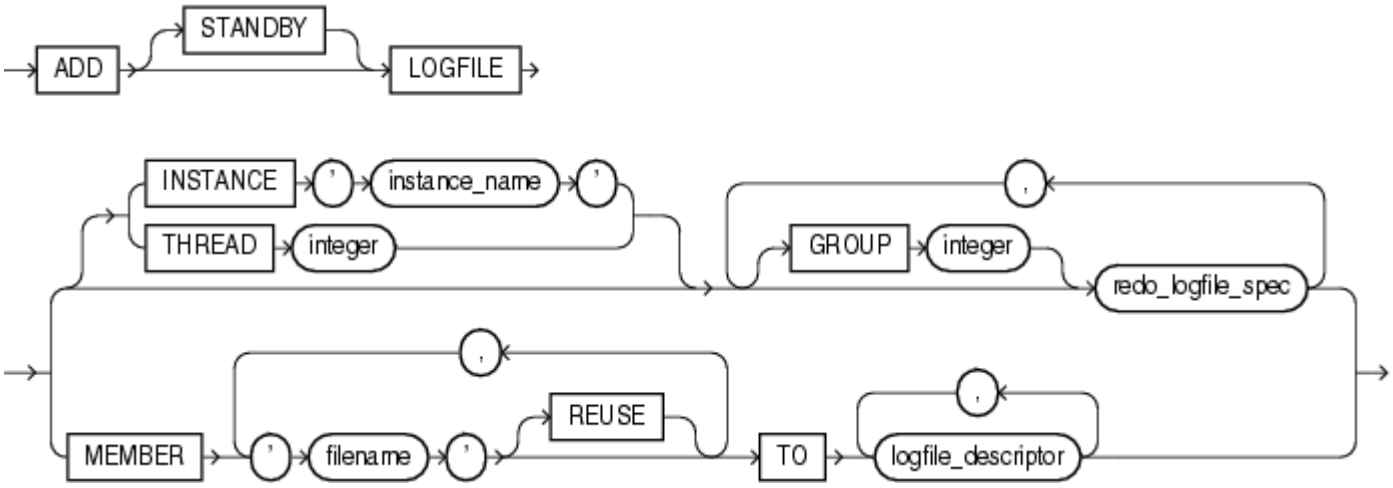
(size_clause::=)

logfile_clauses::=



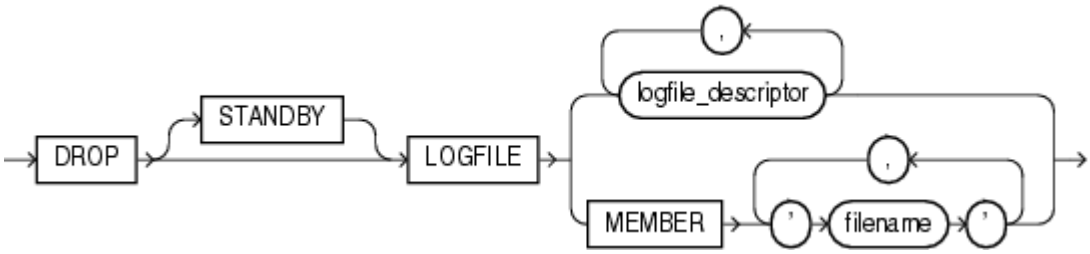
(logfile_descriptor::=、add_logfile_clauses::=、drop_logfile_clauses::=、switch_logfile_clause::=、supplemental_db_logging::=)

add_logfile_clauses::=



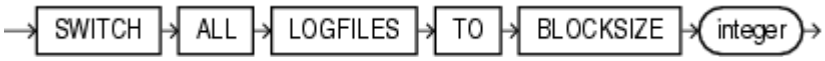
(redo_log_file_spec::=、logfile_descriptor::=)

drop_logfile_clauses::=

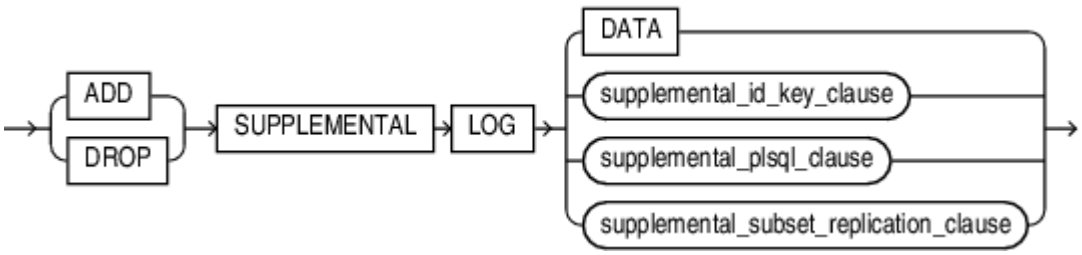


(logfile_descriptor::=)

switch_logfile_clause::=

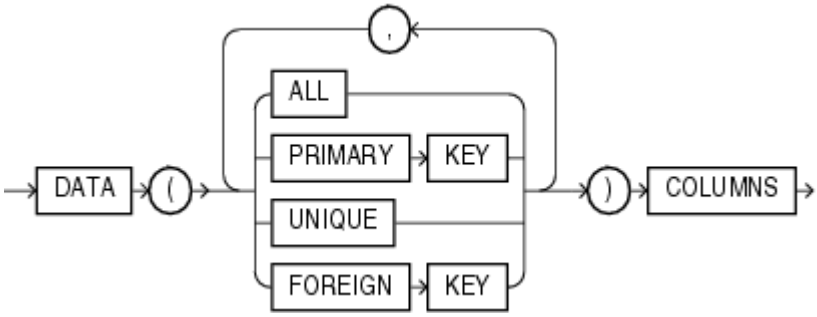


supplemental_db_logging::=



(supplemental_id_key_clause::=)

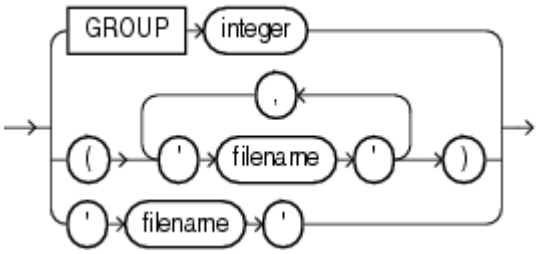
supplemental_id_key_clause::=を参照



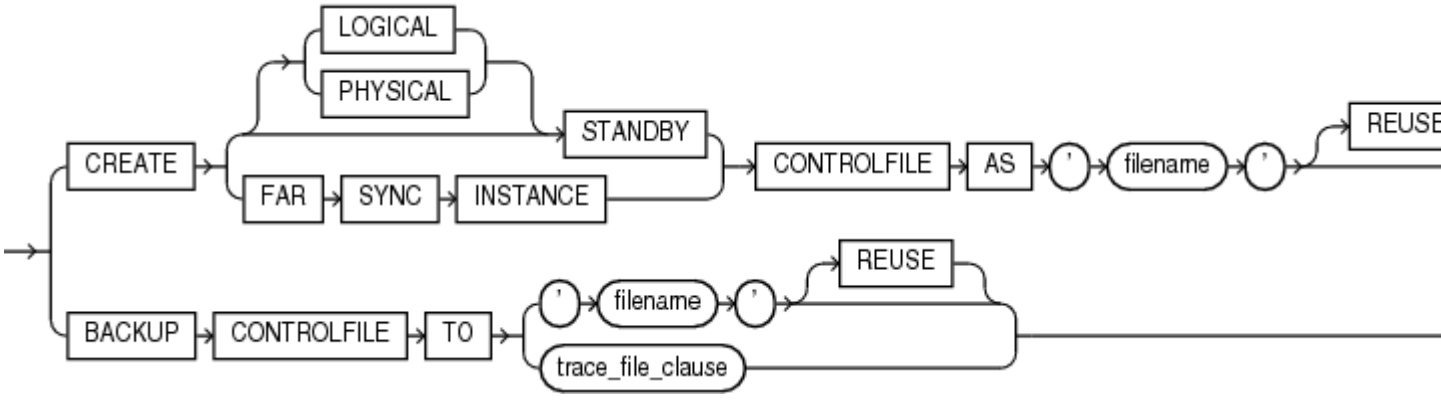
supplemental_plsql_clause::=

supplemental_subset_replication_clause

logfile_descriptor::=

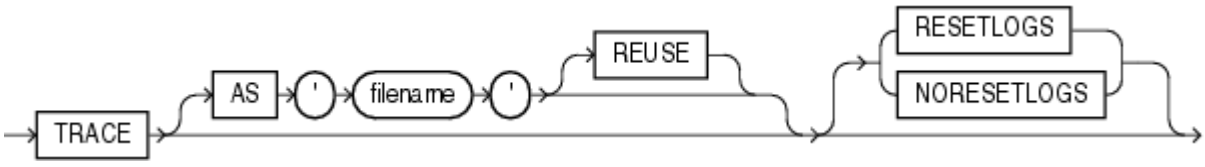


controlfile_clauses::=

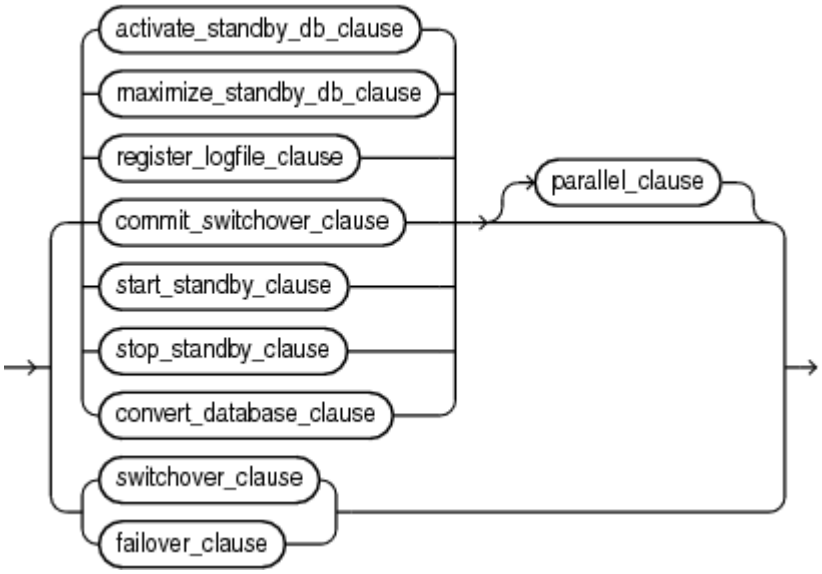


(trace_file_clause::=)

trace_file_clause::=

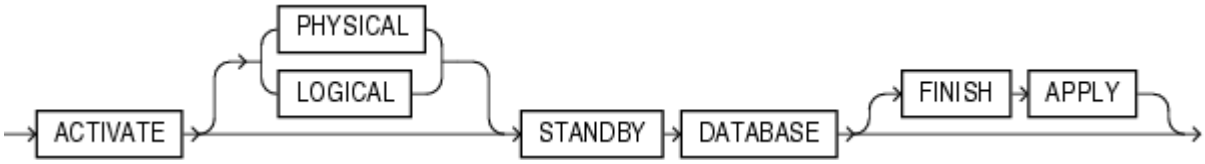


standby_database_clauses::=

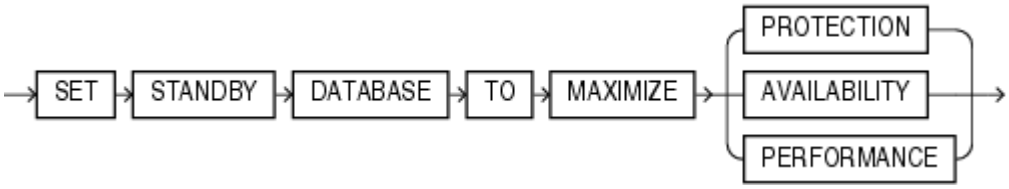


(activate_standby_db_clause::=、maximize_standby_db_clause::=、register_logfile_clause::=、commit_switchover_clause::=、start_standby_clause::=、stop_standby_clause::=、convert_database_clause::=、parallel_clause::=、switchover_clause::=、failover_clause::=)

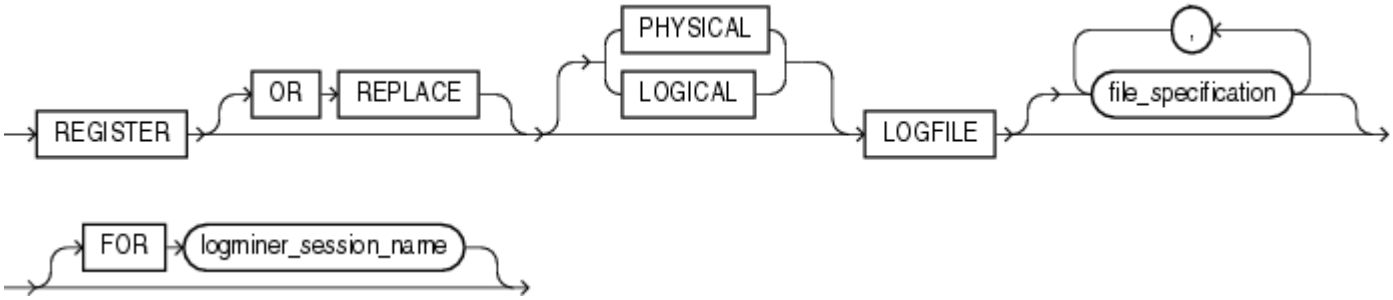
activate_standby_db_clause::=



maximize_standby_db_clause::=



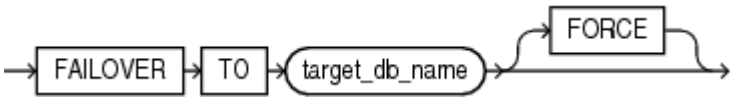
register_logfile_clause::=



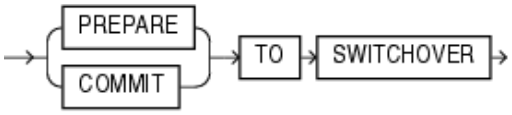
(file_specification::=)

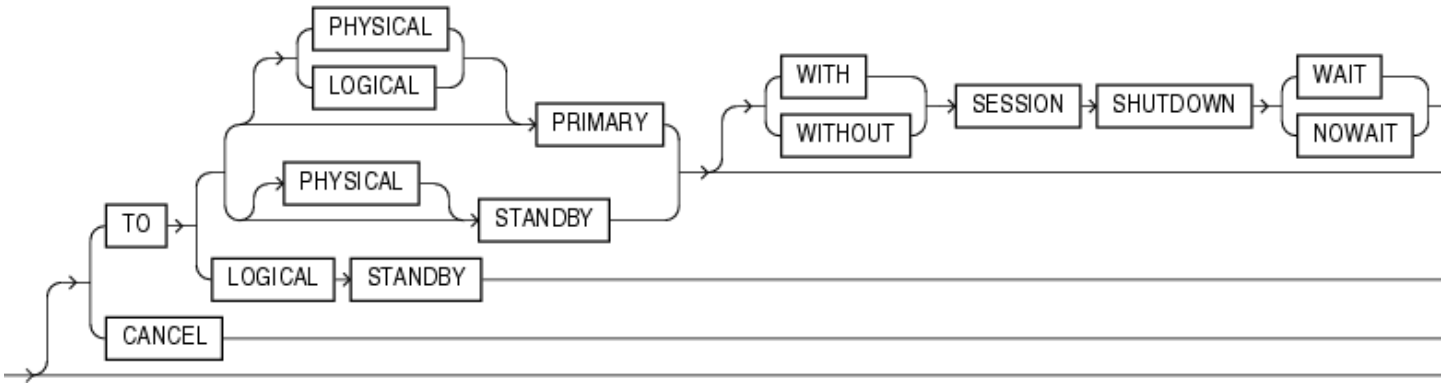
switchover_clause::=

failover_clause::=

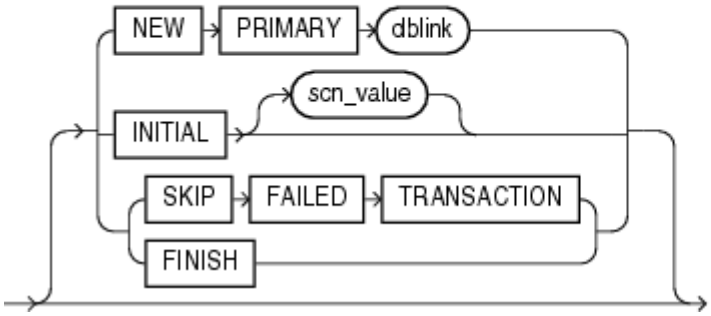


commit_switchover_clause::=



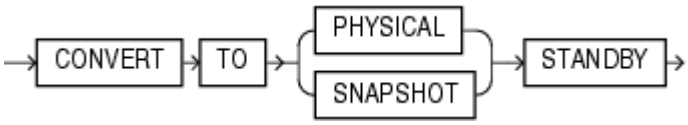


start_standby_clause::=

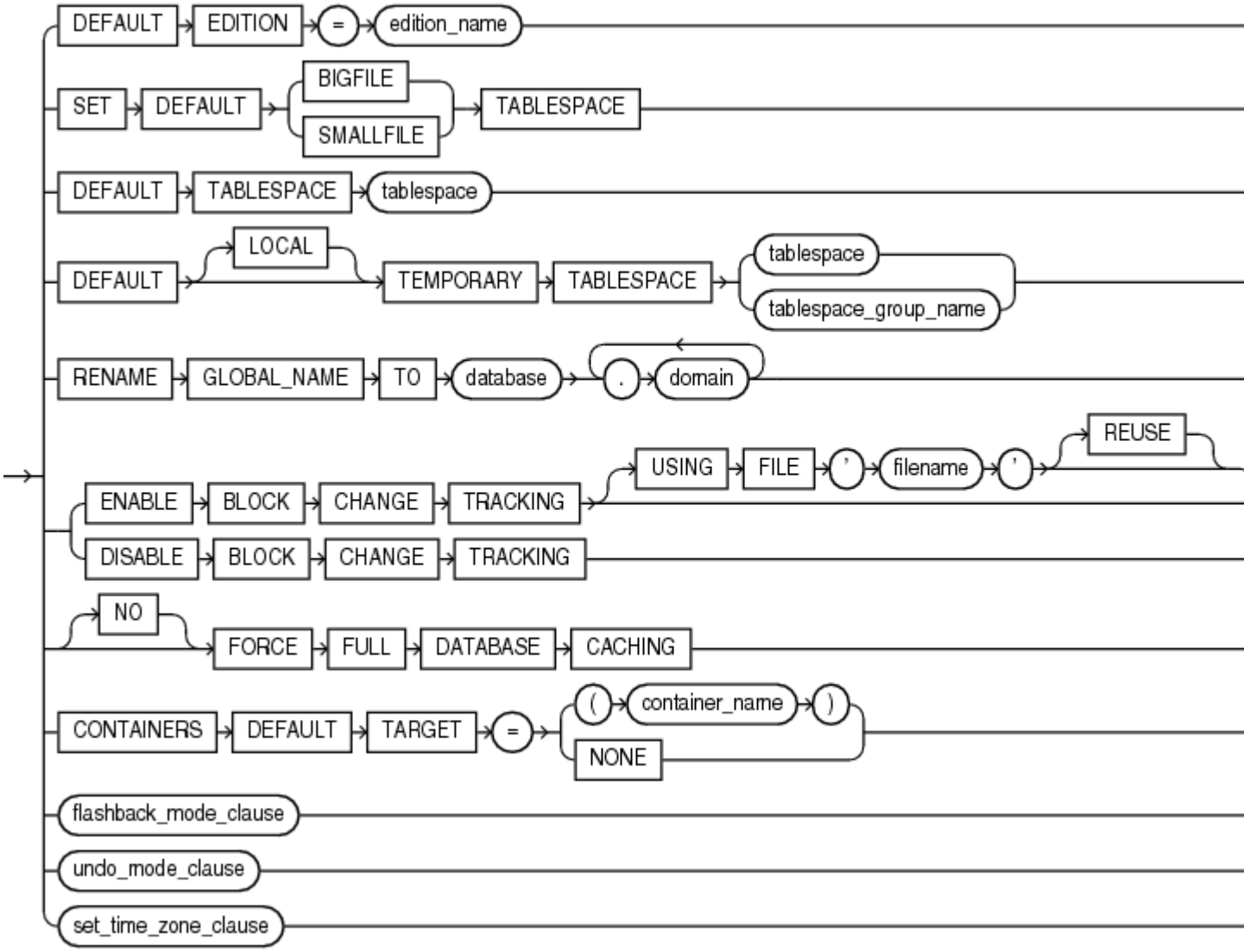


stop_standby_clause::=

convert_database_clause::=

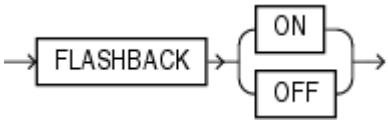


default_settings_clauses::=



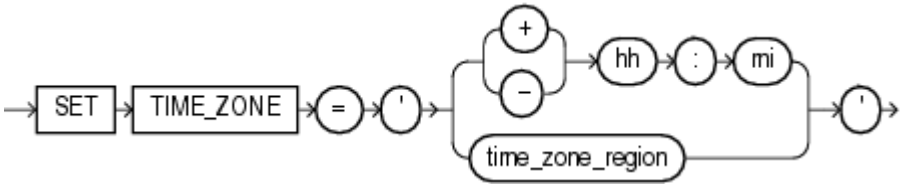
(flashback_mode_clause::=、undo_mode_clause::=、set_time_zone_clause::=)

flashback_mode_clause::=

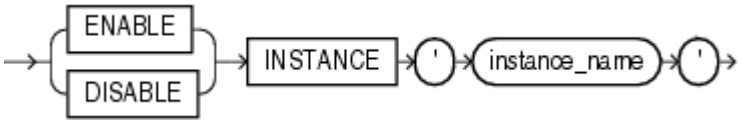


undo_mode_clause::=

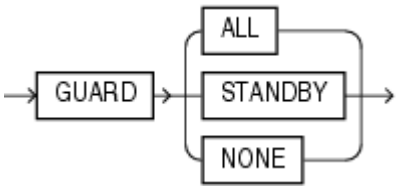
set_time_zone_clause::=



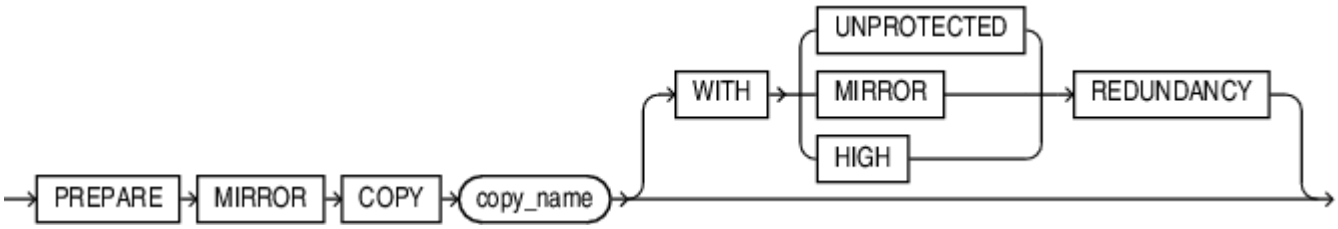
instance_clauses::=



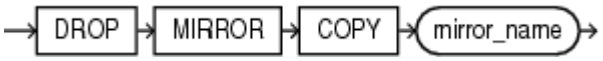
security_clause::=



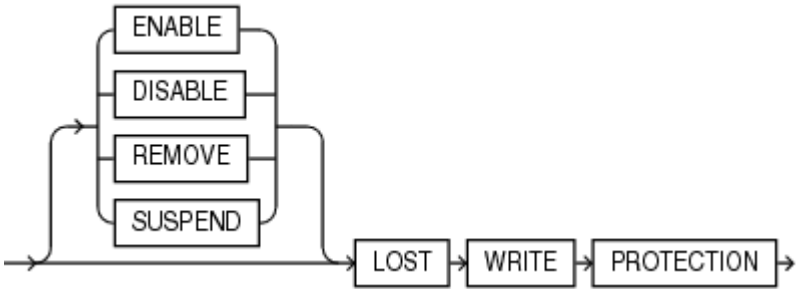
prepare_clause::=



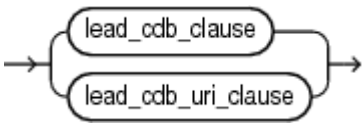
drop_mirror_copy::



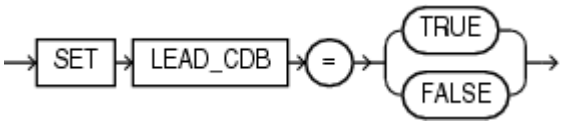
lost_write_protection ::=



cdb_fleet_clauses::=



lead_cdb_clause::=



lead_cdb_uri_clause::=

property_clause

セマンティクス

database_clause

非コンテナ・データベースのDATABASEオプションを指定します。

db_name

変更するデータベースの名前を指定します。db_nameを省略した場合は、初期化パラメータDB_NAMEの値によって特定されたデータベースが変更されます。なお、データベースの制御ファイルが初期化パラメータCONTROL_FILESに指定されている場合にのみ、そのデータベースを変更できます。データベース識別子は、Oracle Netのデータベース指定とは関係ありません。

startup_clauses

startup_clausesを使用すると、データベースをマウントおよびオープンして、アクセス可能にできます。

MOUNT句

MOUNT句を使用すると、データベースをマウントできます。データベースがすでにマウントされている場合、この句は使用できません。

MOUNT STANDBY DATABASE

MOUNT STANDBY DATABASEを指定すると、フィジカル・スタンバイ・データベースをマウントできます。キーワードSTANDBY DATABASEは省略可能です(マウントされるデータベースがプライマリとセカンダリのどちらであるかは自動的に判別されるため)。この文が実行されるとすぐに、スタンバイ・インスタンスはプライマリ・インスタンスからREDOデータを受信できるようになります。

関連項目:

スタンバイ・データベースの詳細は、『Oracle Data Guard概要および管理』を参照してください。

MOUNT CLONE DATABASE

MOUNT CLONE DATABASEを指定すると、クローン・データベースをマウントできます。

OPEN句

OPEN句を使用すると、データベースを使用可能な状態にできます。データベースをオープンするには、マウントしておく必要があります。

OPENを他のキーワードを指定せずに単独で指定した場合のデフォルトは、プライマリ・データベース、ロジカル・スタンバイ・データベースまたはスナップショット・スタンバイ・データベースではOPEN READ WRITE NORESETLOGS、フィジカル・スタンバイ・データベースではOPEN READ ONLYです。

OPEN READ WRITE

OPEN READ WRITEを指定すると、読取り/書込みモードでデータベースがオープンされ、ユーザーはREDOログを生成できるようになります。これは、プライマリ・データベースをオープンするときのデフォルトです。この句は、フィジカル・スタンバイ・データベースに対しては指定できません。

関連項目:

READ ONLY / READ WRITE: 例

RESETLOGS | NORESETLOGS

現行のログ順序番号を1にリセットし、アーカイブされていないログ(現行のログを含む)をアーカイブして、リカバリ時に適用されなかったREDO情報を今後適用されないように破棄するかどうかを指定します。Oracle Databaseは、この句の設定が必要な次の状況を除き、NORESETLOGSを自動的に使用します。

- 次の場合は、RESETLOGSを指定する必要があります。
    - 不完全メディア・リカバリ、またはバックアップ制御ファイルを使用してメディア・リカバリを実行した後
    - 前回の不完全なOPEN RESETLOGS操作の後
    - FLASHBACK DATABASE処理の後
- 作成した制御ファイルをマウントした場合、オンライン・ログが失われたときはRESETLOGSを、失われていないときはNORESETLOGSを指定する必要があります。

UPGRADE | DOWNGRADE

データベースをアップグレードまたはダウングレードする場合にかぎり、これらのOPEN句パラメータを使用します。この句は、Oracle Databaseに対して、アップグレードとダウングレードにそれぞれ必要なシステム・パラメータを動的に変更するように指示します。SQL*PlusのSTARTUP UPGRADEコマンドまたはSTARTUP DOWNGRADEコマンドでも、同じ結果が得られます。

CDBのUPGRADEパラメータまたはDOWNGRADEパラメータを使用すると、ルート・コンテナは指定したモードで開かれますが、それ以外のコンテナはすべてREAD WRITEモードで開かれます。

関連項目:

- データベースをあるリリースから別のリリースにアップグレードまたはダウングレードするステップについては、『Oracle Databaseアップグレード・ガイド』を参照してください。
- SQL*PlusのSTARTUPコマンドの詳細は、『SQL*Plusユーザーズ・ガイドおよびリファレンス』を参照してください。

OPEN READ ONLY

OPEN READ ONLYを指定すると、トランザクションが読取り専用に制限され、ユーザーはREDOログを生成できなくなります。この設定は、フィジカル・スタンバイ・データベースをオープンするときのデフォルトであり、このように設定すると、プライマリ・データベース・サイトからのアーカイブ・ログのコピー中でも、フィジカル・スタンバイ・データベースが問合せに使用可能になります。

データベースのオープンの制限事項

データベースをオープンする場合、次の制限事項が適用されます。

- データベースが現在他のインスタンスでREAD WRITEモードでオープンされている場合、READ ONLYモードではオープンできません。
- データベースをリカバリする必要がある場合、READ ONLYモードではオープンできません。
- データベースがREAD ONLYモードでオープンされている間は、表領域をオフラインに切り替えることはできません。ただし、データファイルのオフラインとオンラインの切替えは可能であり、データベースがREAD ONLYモードでオープンされている間にオフラインのデータファイルおよび表領域をリカバリすることも可能です。

関連項目:

フィジカル・スタンバイ・データベースのオープン方法の詳細は、『Oracle Data Guard概要および管理』を参照してください。

recovery_clauses

recovery_clausesを使用すると、バックアップ後の操作を指定できます。これらすべての句では、Oracle Databaseは、現行の制御ファイルが認識しているデータファイルおよびログ・ファイルのインカネーションを使用して、データベースをリカバリします。

関連項目:

データベースのバックアップの詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』および「データベースのリカバリ: 例」を参照してください。

CDBでのrecovery_clausesの使用のノート

現在のコンテナがルートである場合は、すべてのrecovery_clausesを指定してCDB全体のバックアップとリカバリを実行できます。

現在のコンテナがPDBである場合は、recovery_clausesの次の副次句を指定してそのPDBのバックアップとリカバリを実行できます。

- BEGIN BACKUP
- END BACKUP
- full_database_recovery: DATABASEキーワードのみを指定できます。
- partial_database_recovery
- general_recoveryのLOGFILE句とCONTINUE句

前述の副次句は、ALTER PLUGGABLE DATABASEのpdb_recovery_clausesを使用して指定することもできます。ALTER PLUGGABLE DATABASEの構文図pdb_recovery_clausesを参照してください。

general_recovery

general_recovery句を指定すると、データベース、スタンバイ・データベース、または指定した表領域やファイルのメディア・リカバリを制御できます。インスタンスで、データベースがマウント済(オープン状態またはクローズ状態)の場合、関連ファイルが使用中でなければ、この句を使用できます。

ノート:

全体的または部分的なデータベース・リカバリおよびログファイルのリカバリでは、デフォルトでパラレル化が有効になっています。並列度はデータベースによって計算されます。NOPARALLEL を指定してこれらの操作の並列化を無効にすることも、並列度を PARALLEL integer で指定することもできます(それぞれの構文図を参照してください)。

一般的なデータベース・リカバリの制限事項

一般的なリカバリには、次の制限事項があります。

- データベースがクローズ状態の場合にのみ、データベース全体をリカバリできます。
- インスタンスで、データベースが排他モードでマウントされている必要があります。
- リカバリ対象の表領域またはデータファイルがオフラインの場合、データベースがオープン状態でもクローズ状態でも、表領域またはデータファイルをリカバリできます。
- 共有サーバー・アーキテクチャでOracle Databaseに接続している場合、メディア・リカバリは実行できません。

関連項目:

- RMANメディア・リカバリおよびユーザー定義メディア・リカバリの詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- SQL*PlusのRECOVERコマンドの詳細は、『SQL*Plusユーザーズ・ガイドおよびリファレンス』を参照してください。

AUTOMATIC

AUTOMATICを指定すると、リカバリ操作を続けるために必要な、次のアーカイブREDOログ・ファイルの名前が自動的に生成されます。LOG_ARCHIVE_DEST_nパラメータが定義されている場合は、Oracle Databaseはこのパラメータのうち、有効で使用可能なものをスキャンして、最初のローカル・アーカイブ先を決定します。このアーカイブ先をLOG_ARCHIVE_FORMATと組み合せて、ターゲットのREDOログ・ファイル名が生成されます。LOG_ARCHIVE_DEST_nパラメータが定義されていない場合は、かわりにLOG_ARCHIVE_DESTパラメータ値が使用されます。

生成された名前のファイルが見つかった場合は、そのファイルに格納されているREDOが適用されます。ファイルが見つからない場合は、ファイル名の入力を求めるプロンプトが表示され、このときに、生成されたファイル名が候補として表示されます。

AUTOMATICもLOGFILEも指定しなかった場合は、ファイル名の入力を求めるプロンプトが表示され、このときに、生成されたファイル名が候補として表示されます。この生成されたファイル名をそのまま使用することも、別の完全修飾ファイル名を入力することもできます。アーカイブ済のファイル名が、Oracle Databaseによって生成されるファイル名とは異なることがわかっている場合は、LOGFILE句を使用すると時間を節約できます。

FROM 'location'

FROM 'location'を指定すると、アーカイブREDOログ・ファイル・グループを読み取る位置を指定できます。locationには、使用するオペレーティング・システムの表記規則に従って、ファイルの位置を完全に指定する必要があります。このパラメータを指定しないと、そのアーカイブREDOログ・グループは、初期化パラメータLOG_ARCHIVE_DESTまたはLOG_ARCHIVE_DEST_1に指定された位置にあるとみなされます。

full_database_recovery

full_database_recovery句を指定すると、データベース全体をリカバリできます。

DATABASE

DATABASE句を指定すると、データベース全体をリカバリできます。これはデフォルトです。データベースがクローズされている場合にのみ、このオプションを使用できます。

STANDBY DATABASE

STANDBY DATABASE句を指定すると、プライマリ・データベースからコピーされたアーカイブREDOログ・ファイルおよび制御ファイルを使用して、フィジカル・スタンバイ・データベースを手動でリカバリできます。スタンバイ・データベースは、マウントされているがオープンされていない状態である必要があります。

この句は、オンライン・データファイルのみをリカバリします。

- UNTIL句を使用すると、リカバリ操作の存続期間を指定できます。
  - CANCELは、取消しベースのリカバリを示します。RECOVER CANCEL句を指定したALTER DATABASE文を発行するまで、データベース・リカバリが続行されます。
  - TIMEは、時間ベースのリカバリを示します。このパラメータは、dateに指定した時点までデータベースをリカバリします。dateは、'YYYY-MM-DD:HH24:MI:SS'の書式の文字リテラルである必要があります。
  - CHANGEは、変更ベースのリカバリを示します。integerに指定したシステム変更番号の直前の、トランザクションの一貫性が保たれるところまでデータベースをリカバリします。
  - CONSISTENTは、データベースを読取り専用モードでオープンできるよう、すべてのオンライン・ファイルの一貫性が保たれるSCNポイントまでデータベースをリカバリします。この句では、制御ファイルがバックアップ制御ファイルである必要があります。
- 現行の制御ファイルのかわりにバックアップ制御ファイルを使用する場合、USING BACKUP CONTROLFILEを指定します。
- Storage Snapshot Optimizationを使用してストレージ・スナップショットでデータベースをリカバリするには、SNAPSHOT TIME句を指定します。この句はストレージ・スナップショットが作成されたときにデータベースがバックアップ・モードでなかったときに使用できます。
  - dateは、'YYYY-MM-DD:HH24:MI:SS'形式の文字リテラルである必要があります。これはスナップショットが完了した直後の時刻を示す必要があります。UNTIL TIME句を指定する場合は、SNAPSHOT TIME dateはUNTIL TIME date以前の時刻である必要があります。

関連項目:

ストレージ・スナップショットの最適化を使用したリカバリの詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

partial_database_recovery

partial_database_recovery句を指定すると、個々の表領域およびデータファイルをリカバリできます。

TABLESPACE

TABLESPACE句を指定すると、指定した表領域のみをリカバリできます。リカバリの対象となる表領域がオフラインの場合、データベースがオープン状態でもクローズ状態でも、この句を使用できます。

関連項目:

パラレル・リカバリ処理の使用方法: 例

DATAFILE

DATAFILE句を指定すると、指定したデータファイルをリカバリできます。リカバリ対象のデータファイルがオフラインの場合、データベースがオープン状態でもクローズ状態でも、この句を使用できます。

データファイルは、名前または番号で識別できます。番号で識別した場合、filenumberは、V$DATAFILE動的パフォーマンス・ビューのFILE#列、またはDBA_DATA_FILESデータ・ディクショナリ・ビューのFILE_ID列の数を表す整数です。

STANDBY {TABLESPACE | DATAFILE}

旧リリースでは、スタンバイ上の特定の表領域や特定のデータファイルの古いバックアップをリカバリして、残りのスタンバイ・データベースとの一貫性を確保するために、STANDBY TABLESPACEまたはSTANDBY DATAFILEを指定できました。これらの2つの句は、現在サポートされていません。かわりに、ALTER DATABASE RECOVER MANAGED STANDBY DATABASE UNTIL CONSISTENT文を使用して、一貫性のあるポイントまで(それ以上は超えずに)スタンバイ・データベースをリカバリします。

LOGFILE

LOGFILE 'filename'を指定すると、指定したREDOログ・ファイルを使用して、メディア・リカバリを続行できます。

TEST

TEST句を使用すると、試行リカバリを実行できます。試行リカバリが役立つのは、通常のリカバリ手順でなんらかの問題が発生した場合です。REDOストリームの内容をあらかじめ見ることができるので、その他に発生する可能性のある問題を検出できます。試行リカバリでは、標準リカバリに似た方法でREDOが適用されますが、ディスクへの変更書込みは行われず、リカバリによる変更は試行リカバリの終了時にロールバックされます。

この句を使用できるのは、最後のRESETLOGS操作以降に取ったバックアップをリストアする場合のみです。そうでない場合は、エラーが戻ります。

ALLOW ... CORRUPTION

ALLOW integer CORRUPTION句を指定すると、ログ・ファイルが破損した場合に、許容リカバリの続行で許容する破損ブロックの数を指定することができます。

関連項目:

- データベースのリカバリの概要は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- スタンバイ・データベースの管理リカバリの詳細は、『Oracle Data Guard概要および管理』を参照してください。

CONTINUE

CONTINUEを指定すると、スレッドを使用禁止にするために中断されていた複数インスタンス・リカバリを再開できます。

CONTINUE DEFAULTを指定すると、他に指定されたログ・ファイルがない場合、自動的に生成されたREDOログ・ファイルを使用して、リカバリが再開されます。ファイル名の入力を求めるプロンプトが表示されないこと以外は、AUTOMATICの指定と同じです。

CANCEL

CANCELを指定すると、取消しベースのリカバリを終了できます。

managed_standby_recovery

managed_standby_recovery句を使用すると、フィジカル・スタンバイ・データベースでのREDO Applyを開始および停止できます。REDO Applyは、プライマリ・データベースから受け取るREDOを継続的に適用することで、スタンバイ・データベースとプライマリ・データベースでのトランザクションの一貫性を確保します。

プライマリ・データベースは、そのREDOデータをスタンバイ・サイトに転送します。REDOデータが、フィジカル・スタンバイ・サイトのREDOログ・ファイルに書き込まれると、REDO Applyでそのログ・ファイルを使用できるようになります。
managed_standby_recovery句は、スタンバイ・インスタンスにデータベースがマウントされているか、またはスタンバイ・インスタンスが読取り専用でオープンされている場合にのみ使用できます。

ノート:

Oracle Database 12c 以降では、REDO Apply 時にデフォルトでリアルタイム適用が有効化されます。リアルタイム適用は、スタンバイ REDO ログ・ファイルが書き込まれると、このログ・ファイルからすぐに REDO をリカバリします。このとき、このログ・ファイルを最初に物理スタンバイ・データベースでアーカイブする必要はありません。リアルタイム適用は、USING ARCHIVED LOGFILE 句を使用して無効化できます。詳細は、次を参照してください。

- リアルタイム適用の詳細は、『Oracle Data Guard 概要および管理』を参照してください。
- USING ARCHIVED LOGFILE 句

ノート:

パラレル化は、REDO Apply の実行中、デフォルトで有効になります。並列度はデータベースによって計算されます。NOPARALLEL を指定してこれらの操作の並列化を無効にすることも、並列度を PARALLEL integer で指定することもできます(それぞれの構文図を参照してください)。

管理スタンバイ・リカバリの制限事項

この句には、「general_recovery」で示されているものと同じ制限事項が適用されます。

関連項目:

この句の使用の詳細は、『Oracle Data Guard概要および管理』を参照してください。

USING ARCHIVED LOGFILE句

USING ARCHIVED LOGFILEを指定すると、リアルタイム適用を有効化することなくREDO Applyを開始できます。

DISCONNECT

DISCONNECTを指定すると、現行のセッションを他のタスクで使用可能にしたまま、REDO Applyをバックグラウンドで実行するよう指示できます。FROM SESSIONキーワードはオプションであり、意味を明確にするためのものです。

NODELAY

NODELAY句は、プライマリ・データベースのLOG_ARCHIVE_DEST_nパラメータのDELAY属性より優先されます。NODELAY句を指定しない場合は、LOG_ARCHIVE_DEST_n設定のDELAY属性に従って、アーカイブREDOログ・ファイルの適用が遅延されます(属性が設定されている場合)。このパラメータにDELAY属性が指定されていない場合、アーカイブREDOログ・ファイルは、すぐにスタンバイ・データベースに適用されます。

リアルタイム適用をUSING CURRENT LOGFILE句とともに指定すると、このスタンバイのプライマリでLOG_ARCHIVE_DEST_nパラメータに対して指定されたDELAY値は無視されます。デフォルトはNODELAYです。

UNTIL CHANGE句

この句を使用すると、特定のシステム変更番号より前のREDOデータをリカバリするようREDO Applyに指示できます。

UNTIL CONSISTENT

この句を使用すると、スタンバイ・データベースを読取り専用モードでオープンできるよう、一貫性のあるSCNポイントまでスタンバイ・データベースをリカバリできます。

USING INSTANCES

この句は、Oracle Real Application Clusters (Oracle RAC)またはOracle RAC One Nodeデータベースにのみ適用可能で、これを使用すると、コマンドが実行されたインスタンスと同じモード(MOUNTEDまたはREAD ONLY)で開始されたスタンバイの、複数のインスタンスで適用プロセスを開始できます。USING INSTANCES ALLを指定すると、同じモードで開始されたOracle RACスタンバイ・データベースのすべてのインスタンスでRedo Applyが実行されます。USING INSTANCES integerを指定すると、同じモードで開始された、指定された数のインスタンスでRedo Applyが実行されます。integerには、1からスタンバイ・データベースのインスタンス数までの整数値を指定します。Redo Applyを実行するインスタンスはデータベースで選択され、ユーザーが特定のインスタンスを指定することはできません。たとえば、MOUNTEDのインスタンスから4つのインスタンスを指定し、スタンバイの3つのインスタンスのみがMOUNTEDモードで実行されている場合、Redo Applyはその3つのインスタンスでのみ開始されます。USING INSTANCES句を省略した場合、Oracle Databaseはコマンドが実行されたインスタンスでのみRedo Applyを実行します。

FINISH

FINISHを指定すると、フェイルオーバー用に準備されている使用可能なすべてのREDOデータの適用を完了できます。

FINISH句を使用するのは、プライマリ・データベースに障害が発生した場合のみです。この句は、設定済の遅延間隔よりも優先され、この句を指定すると、使用可能なすべてのREDOが即座に適用されます。FINISHコマンドが完了すると、このデータベースをスタンバイ・データベース・ロールで実行することができなくなります。このデータベースは、ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY文を発行してプライマリ・データベースに変換する必要があります。

CANCEL

CANCELを指定すると、REDO Applyをすぐに停止できます。REDO Applyが停止すると、すぐに制御が戻されます。

TO LOGICAL STANDBY句

この句を使用すると、フィジカル・スタンバイ・データベースをロジカル・スタンバイ・データベースに変換できます。

db_name

新しいロジカル・スタンバイ・データベースを識別するデータベース名を指定します。この文の発行時にサーバー・パラメータ・ファイル(spfile)を使用している場合、データベースは、新しいロジカル・スタンバイ・データベースに関する適切な情報でこのファイルを更新します。spfileを使用していない場合は、データベースの停止後にDB_NAMEパラメータの名前を設定するよう求めるメッセージが発行されます。さらに、この句をスタンバイ・データベースで使用する前に、プライマリ・データベースでDBMS_LOGSTDBY.BUILD PL/SQLプロシージャを起動する必要もあります。

関連項目:

DBMS_LOGSTDBY.BUILDプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

KEEP IDENTITY

ロジカル・スタンバイによって提供されるローリング・アップグレード機能を使用し、さらにプライマリ・データベースおよびフィジカル・スタンバイの元の構成に戻す場合は、この句を使用します。この句を使用して作成されたロジカル・スタンバイ・データベースでは、スイッチオーバーおよびフェイルオーバーのサポートは制限されます。そのため、この句を使用して汎用ロジカル・スタンバイ・データベースを作成しないでください。

関連項目:

ローリング・アップグレードの詳細は、『Oracle Data Guard概要および管理』を参照してください。

非推奨になる予定の管理スタンバイ・リカバリ句

以前のリリースの構文には、次の句が記載されていました。これらの句は非推奨になる予定であり、使用する必要はありません。これらの句は使用しないことをお薦めします。

FINISH FORCE、FINISH WAIT、FINISH NOWAIT

FINISH句のこれらのオプションの書式は、非推奨です。ここでは下位互換性のために、これらのセマンティクスについて説明します。

- FORCEを指定すると、FINISHプロセスを開始する妨げになる非アクティブなREDO転送セッションを終了できます。
- NOWAITを指定すると、リカバリが完了する前に、フォアグラウンド・プロセスに制御が戻ります。
- WAIT(デフォルト)を指定すると、リカバリが完了してから、フォアグラウンド・プロセスに制御が戻ります。

指定されている場合、これらの句は無視されます。ターミナル・リカバリはフォアグラウンドで実行され、常にすべてのREDO転送セッションを終了します。そのため、リカバリが完了するまでユーザーに制御は戻りません。

CANCEL IMMEDIATE、CANCEL WAIT、CANCEL NOWAIT

CANCEL句のこれらのオプションの書式は非推奨です。ここでは下位互換性のために、これらのセマンティクスについて説明します。

- IMMEDIATEキーワードを指定すると、現在のREDOログ・ファイルが完全に適用される前にREDO Applyを停止できます。セッション制御は、REDO Applyが実際に停止した時点で戻ります。
- NOWAITキーワードを含めると、CANCEL操作が終了するのを待たずに、セッション制御が戻ります。

指定されている場合、これらの句は無視されます。REDO Applyは常にすぐに取り消され、操作の完了後にのみ制御がセッションに戻されます。

USING CURRENT LOGFILE句

USING CURRENT LOGFILE句は非推奨です。これにより、REDO Apply時にリアルタイム適用が起動されます。ただし、現在、これはデフォルトの動作になっているため、この句を利用する意味はなくなりました。

BACKUP句

この句を使用すると、データベース内のすべてのデータファイルをオンライン・バックアップ・モード(ホット・バックアップ・モードともいう)にしたり、このモードから戻すことができます。

関連項目:

個々の表領域のすべてのデータファイルに対するオンライン・バックアップ・モードの設定または解除の詳細は、「ALTER TABLESPACE」を参照してください。

BEGIN BACKUP句

BEGIN BACKUPを指定すると、データベース内のすべてのデータファイルがオンライン・バックアップ・モードに移行します。データベースをマウントしてオープンしておく必要があります。また、メディア・リカバリを使用可能にする(データベースをARCHIVELOGモードにする)必要があります。

データベースがオンライン・バックアップ・モードになっている間は、インスタンスの通常停止、個々の表領域のバックアップの開始、表領域のオフラインへの切替え、または表領域の読取り専用設定は実行できません。

この句は、オフラインまたは読取り専用の表領域のデータファイルには影響しません。

END BACKUP句

END BACKUPを指定すると、オンライン・バックアップ・モードの現行のデータベースにあるすべてのデータファイルを、オンライン・バックアップ・モードから戻すことができます。この操作を実行するとき、データベースはマウント済(オープン状態またはクローズ状態)である必要があります。

システム障害、インスタンス障害、またはSHUTDOWN ABORT操作の後は、オンライン・バックアップ・モードのファイルがシステム・クラッシュ時のファイルと一致するかどうかは識別されません。ファイルに一貫性がある場合、個々のデータファイル、またはすべてのデータファイルをオンライン・バックアップ・モードから戻すことができます。これによって、起動時にファイルのメディア・リカバリを回避できます。

- ALTER DATABASE DATAFILE ... END BACKUP文を使用すると、個々のデータファイルをオンライン・バックアップ・モードから戻すことができます。「database_file_clauses」を参照してください。
- ALTER TABLESPACE ... END BACKUP文を使用すると、表領域内のすべてのデータファイルをオンライン・バックアップ・モードから戻すことができます。

database_file_clauses

database_file_clausesを指定すると、データファイルおよび一時ファイルを変更できます。インスタンスで、データベースがマウント済(オープン状態またはクローズ状態)の場合、関連ファイルが使用中でなければ、次の句はどれでも使用できます。ただし、move_datafile_clauseを使用すると、使用中のデータファイルを移動できます。

RENAME FILE句

RENAME FILE句を使用すると、データファイル、一時ファイルまたはREDOログ・ファイル・メンバーの名前を変更できます。この句を指定する前に、オペレーティング・システムのファイル名の表記規則に従って、各ファイル名を指定してください。

- データファイルまたは一時ファイルにこの句を使用するには、データベースをマウントしておく必要があります。データベースはオープンしていてもかまいませんが、名前を変更するデータファイルまたは一時ファイルはオフラインにしておく必要があります。また、最初にファイル・システム上のファイルを新しい名前に変更する必要があります。
- ログ・ファイル用にこの句を使用するには、データベースはマウントされているがオープンされていない状態である必要があります。

- ブロック・チェンジ・トラッキングを使用可能にしている場合、この句を使用してブロック・チェンジ・トラッキング・ファイルの名前を変更できます。ブロック・チェンジ・トラッキング・ファイルの名前を変更する場合、データベースはマウントされているがオープンされていない状態である必要があります。

この句によって名前が変更されるのは、制御ファイル内のファイルのみです。オペレーティング・システムのファイルの名前が実際に変更されることはありません。オペレーティング・システムのファイルは引き続き存在しますが、Oracle Databaseによって使用されることはなくなります。

関連項目:

- データファイルと一時ファイルのリカバリの詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- 「ログ・ファイル・メンバーの名前変更: 例」および「一時ファイルの操作: 例」

create_datafile_clause

CREATE DATAFILE句を使用すると、元のデータファイルのかわりに新しい空のデータファイルを作成できます。この句を使用すると、バックアップを取らずに失われたデータファイルを再作成できます。filenameまたはfilenumberには、データベースの一部であるファイルまたは以前一部であったファイルを指定します。番号で識別した場合、filenumberは、V$DATAFILE動的パフォーマンス・ビューのFILE#列、またはDBA_DATA_FILESデータ・ディクショナリ・ビューのFILE_ID列の数を表す整数です。

- AS NEWを指定すると、データファイル用のデフォルトのファイル・システム位置に、システムが生成するファイル名で、置き換えるファイルと同じサイズのOracle Managed Filesのデータファイルが作成されます。
- AS file_specificationを指定すると、新しいデータファイルにファイル名(およびオプションでサイズ)を割り当てることができます。オペレーティング・システムのファイル・システム内の標準データファイルと一時ファイル、またはOracle Automatic Storage Management (Oracle ASM)ディスク・グループのファイルを指定するには、file_specificationのdatafile_tempfile_spec書式(「file_specification」を参照)を使用します。

元のファイル(filenameまたはfilenumber)が既存のOracle Managed Filesのデータファイルの場合、Oracle Databaseは新しいファイルを作成した後、元のファイルを削除しようとします。元のファイルが既存のユーザー管理データファイルの場合、Oracle Databaseは元のファイルを削除しません。

AS句を指定しない場合、Oracle Databaseによって、filenameまたはfilenumberに指定したファイルと同じ名前およびサイズのファイルが新しく作成されます。

リカバリ時には、元のデータファイルの作成後に書き込まれたアーカイブREDOログを、失われたデータファイルにかわる新しい空のデータファイルに適用する必要があります。

新しいファイルは、元のファイルの作成時と同じ状態で作成されます。新しいファイルを元のファイルが失われた時点の状態に戻すには、メディア・リカバリを行ってください。

新規データファイルの作成の制限事項

新規データファイルの作成には、次の制限事項があります。

- SYSTEM表領域の最初のデータファイルに基づいて新しいファイルを作成することはできません。
- このCREATE DATAFILE句には、datafile_tempfile_specのautoextend_clauseは指定できません。

関連項目:

- 新しいデータファイル名を指定しない場合にこの句によって戻される結果の詳細は、「CREATE DATABASE」の「DATAFILE句」を参照してください。
- ファイル仕様(datafile_tempfile_spec)の詳細は、「file_specification」および「新規データファイルの作成: 例」を参照してください。

alter_datafile_clause

DATAFILE句を使用すると、名前または番号で識別するファイルを操作できます。番号で識別した場合、filenumberは、V$DATAFILE動的パフォーマンス・ビューのFILE#列、またはDBA_DATA_FILESデータ・ディクショナリ・ビューのFILE_ID列の数を表す整数です。DATAFILE句を使用すると、次のようにデータベース・ファイルを変更できます。

ONLINE

ONLINEを指定すると、データファイルをオンラインにできます。

OFFLINE

OFFLINEを指定すると、データファイルがオフラインになります。データベースがオープンされている場合は、データファイルをオンラインに戻す前にデータファイルのメディア・リカバリを行う必要があります(データファイルをオフラインにする前にチェックポイントが実行されないため)。

FOR DROP

データベースがNOARCHIVELOGモードの場合は、FOR DROP句を指定して、データファイルをオフラインにする必要があります。ただし、この句は、データベースからデータファイルを削除しません。そのため、オペレーティング・システムのコマンドを使用するか、またはデータファイルが存在する表領域を削除する必要があります。削除するまで、データファイルはRECOVERまたはOFFLINEの状態でデータ・ディクショナリに残ります。

データベースがARCHIVELOGモードの場合は、FOR DROP句は無視されます。

RESIZE

RESIZEを指定すると、データファイルのサイズを、指定した絶対サイズ(バイト単位)まで増やしたり減らすことができます。デフォルト値はないため、必ずサイズを指定してください。このコマンドを使用して、消失書込みデータを格納するシャドウ表領域のデータファイルのサイズを変更することもできます。

増やしたサイズに対して十分なディスク領域がない場合、または減らしたサイズを超えるデータがファイルに含まれる場合、エラー・メッセージが戻されます。

関連項目:

データファイルのサイズ変更: 例

END BACKUP

END BACKUPを指定すると、データファイルをオンライン・バックアップ・モードから戻すことができます。END BACKUP句の詳細は、ALTER DATABASE構文の最上位で説明しています。「END BACKUP句」を参照してください。

ENCRYPT | DECRYPT

これらの句を使用すると、透過的データ暗号化(TDE)を使用した、データファイルのオフラインでの暗号化または復号化を実行

できます。指定するすべての表領域で、すべてのデータファイルが暗号化されるか、すべてのデータファイルが復号化されている必要があります。

これらいずれかの句を発行する前に、データベースがマウントされている必要があります。データベースはオープンしておくこともできますが、暗号化または復号化されるデータファイルを含む表領域はオフラインになっている必要があります。TDEマスター・キーはデータベース・メモリーにロードされている必要があります。

- 暗号化されていないデータファイルを暗号化する場合は、ENCRYPTを指定します。データファイルはAES128アルゴリズムを使用して暗号化されます。
- データファイルを復号化する場合は、DECRYPTを指定します。データファイルは、ALTER DATABASE DATAFILE ... ENCRYPT文を使用して、事前に暗号化されている必要があります。

データファイルの暗号化および復号化の制限事項

次の制限事項は、ENCRYPTおよびDECRYPT句に適用されます。

- 一時表領域の一時データファイルを暗号化または復号化することはできません。かわりに、一時表領域を削除し、暗号化された表領域として再作成する必要があります。
- UNDO表領域のデータファイルは暗号化しないことをお薦めします。そのようにすると、キーストアが閉じることが妨げられ、データベースが機能しなくなります。また、暗号化された表領域に関連付けられているすべてのUNDOレコードは、UNDO表領域ですでに自動的に暗号化されているため、この操作は必要ありません。

> ノート:
>
> ENCRYPT または DECRYPT 句の使用は、データファイルのオフライン暗号化または復号化を実行する一連のステップにおける 1 つのステップにすぎません。これらのいずれかの句を使用する前に、すべてのステップについて、『Oracle Database Advanced Security ガイド』を参照してください。

alter_tempfile_clause

TEMPFILE句を使用すると一時データファイルのサイズを変更できますが、autoextend_clauseを指定して永続データファイルの場合と同じ効果を得ることもできます。データベースは、オープンされている必要があります。一時ファイルの指定は、名前と番号のどちらでもかまいません。番号で指定する場合は、filenumberに整数(V$TEMPFILE動的パフォーマンス・ビューのFILE#列に格納されている数値)を指定します。

> ノート:
>
> オペレーティング・システムによっては、一時ファイルのブロックが実際にアクセスされるまで、一時ファイル用の領域が割り当てられない場合があります。領域の割当ての遅延のため、一時ファイルの作成およびサイズ変更が速くなります。ただし、後で一時ファイルが使用されるときに、十分なディスク領域を使用可能にする必要があります。問題を回避するには、一時ファイルの作成またはサイズ変更の前に、ディスクの空き領域が、新しく作成する一時ファイルまたはサイズ変更後の一時ファイルのサイズよりも大きいことを確認してください。ディスク領域に余裕を持たせておくと、関連のない操作による、予期されるディスク使用量の増加にも対応できます。その後で、作成またはサイズ変更操作を実行してください。

DROP

DROPを指定すると、tempfileがデータベースから削除されます。表領域はそのまま残ります。

INCLUDING DATAFILESを指定すると、関連付けられたオペレーティング・システム・ファイルは削除され、削除された各ファイルのメッセージがアラート・ログに書き込まれます。同じ結果は、ALTER TABLESPACE ... DROP TEMPFILE文を使用しても得られます。詳細は、「ALTER TABLESPACE」の「DROP句」を参照してください。

move_datafile_clause

MOVE DATAFILE句を使用すると、オンライン・データファイルを新しい場所に移動できます。この操作を実行するときには、データベースが開かれていて、データファイルにアクセスしていてもかまいません。この処理の実行中には、データファイルのコピーが作成されます。この句を使用する前に、オリジナルのデータファイルとそのコピー用に適切なディスク領域があることを確認してください。

元のデータファイルは、file_name、ASM_filename、またはfile_numberを使用して指定できます。ASMファイル名の詳細は、ASM_filenameを参照してください。番号で識別した場合、file_numberは、V$DATAFILE動的パフォーマンス・ビューのFILE#列、またはDBA_DATA_FILESデータ・ディクショナリ・ビューのFILE_ID列の数を表す整数です。

TO句を使用すると、新しいfile_nameまたはASM_filenameを指定できます。Oracle Managed Filesを使用している場合は、TO句を省略できます。この場合、Oracle Databaseは、データファイル用に一意の名前を作成して、そのファイルをDB_CREATE_FILE_DEST初期化パラメータで指定したディレクトリに保存します。

REUSEを指定すると、すでにデータファイルが存在していても、新しいデータファイルが作成されます。

KEEPを指定すると、元のデータファイルがMOVE DATAFILE操作後に維持されるようになります。元のデータファイルがOracle Managed Fileの場合には、KEEPを指定できません。新しいデータファイルがOracle Managed Fileの場合には、KEEPを指定できます。

autoextend_clause

autoextend_clauseを使用すると、新規または既存のデータファイルまたは一時ファイルの自動拡張を使用可能または使用禁止にできます。この句の詳細は、「file_specification」を参照してください。

logfile_clauses

logfile_clausesを指定すると、ログ・ファイルを追加、削除または変更できます。

ARCHIVELOG

ARCHIVELOGを指定すると、REDOログ・ファイル・グループを再利用する前に、グループの内容をアーカイブできます。このモードでは、メディア・リカバリができるようになります。この句は、インスタンスを正常に停止したか、またはエラーなしで即時停止した後に、再起動してデータベースをマウントした後にのみ使用します。

MANUAL

MANUALを指定すると、Oracle Databaseによって作成されたREDOログ・ファイルのアーカイブを、ユーザーが制御できます。この句は、テープに直接アーカイブするユーザーなどに対して、下位互換性を保つために提供されています。MANUALを指定する場合、次のことに注意します。

- ログの切替えが発生した場合、Oracle DatabaseはREDOログ・ファイルをアーカイブしません。これは手動で行う必要があります。
- アーカイブ・ログの宛先として、スタンバイ・データベースを指定することはできません。したがって、データベースを、MAXIMUM PROTECTIONまたはMAXIMUM AVAILABILITYスタンバイ保護モードにすることはできません。

この句を指定しない場合、REDOログ・ファイルは、初期化パラメータLOG_ARCHIVE_DEST_nに指定された宛先に自動的にアーカイブされます。

NOARCHIVELOG

REDOログ・ファイル・グループを再利用する前に、内容をアーカイブする必要がない場合は、NOARCHIVELOGを指定します。このモードでは、メディア障害後のリカバリはできません。インスタンスでデータベースがマウントされているがオープンされていない場合にのみ、この句を使用します。

[NO] FORCE LOGGING

この句を使用すると、データベースをFORCE LOGGINGモードにしたり、このモードから戻すことができます。データベースは、マウントまたはオープンされている必要があります。

FORCE LOGGINGモードでは、一時表領域および一時セグメントの変更を除くデータベースのすべての変更が記録されます。この設定は、各表領域で指定するNOLOGGINGまたはFORCE LOGGING設定、および各データベース・オブジェクトで指定するNOLOGGING設定より優先され、これらの設定には影響されません。

FORCE LOGGINGを指定すると、Oracle Databaseは、まだ記録されていない実行中のすべての操作が終了するまで待機します。

関連項目:

FORCE LOGGINGモードの使用の詳細は、『Oracle Database管理者ガイド』を参照してください。

SET STANDBY NOLOGGING

STANDBY NOLOGGINGは、ロギングなしで実行できる操作をログに記録しないようにデータベースに対して指示します。データベースは、操作によって作成されたデータ・ブロックをData Guard構成内の資格を持つ各スタンバイ・データベースに送信し、スタンバイの欠落データを防止し、そのデータのプライマリとの同期を維持します。

この句を使用して、ログに記録されないタスクの処理方法を決定します。データベースの作成時にデータベースの2つのロギング・モードのいずれかを選択でき、データベースのロギング・モードを1つのモードからもう1つのモードに変更できます。

- データベースをロード・パフォーマンスのスタンバイ・ロギングなしモードにするには、SET STANDBY NOLOGGING FOR LOAD PERFORMANCEを指定します。このモードでは、ロード・プロセスの速度が低下しない場合には、ログに記録されないタスクの一部としてロードされたデータは、プライベート・ネットワーク接続を介して資格のあるスタンバイに送信されます。ロード・プロセスが低速になる場合は、データは送信されず、各スタンバイで無効化REDOが発生するとプライマリから自動的にフェッチされ、データ・ブロックが受信されるまで再試行されます。
- データベースをデータ可用性のスタンバイ・ロギングなしモードにするには、SET STANDBY NOLOGGING FOR DATA AVAILABILITYを指定します。このモードでは、ログに記録されないタスクの一部としてロードされたデータは、ネットワーク接続を介して、またはネットワーク接続が失敗する場合にはREDO内のブロック・イメージを介して、資格のあるスタンバイに送信されます。つまり、このモードでは、ネットワーク接続または関連するプロセスによってプライベート・ネットワーク接続を介したデータの送信が妨げられる場合には、ロードはログに記録する方法で行うように切り替わります。

スタンバイ・ロギングなしモードの場合、資格のあるスタンバイは、読取りのためにオープンされ、管理リカバリを実行しており、スタンバイREDOログでREDOを受信しているスタンバイです。

スタンバイ・ロギングなしの設定の制限事項

SET STANDBY NOLOGGING句はFORCE LOGGINGと同時に使用することはできません。

RENAME FILE句

ログ・ファイルに対するこの句の機能は、データファイルおよび一時ファイルの場合と同じです。「RENAME FILE句」を参照してください。

CLEAR LOGFILE句

CLEAR LOGFILE句を使用すると、オンラインREDOログを再初期化でき、このときにREDOログをアーカイブしないことも選択できます。CLEAR LOGFILEは、REDOログの追加および削除と似ていますが、スレッドのログが2つしかなくてもこの文を発行できる点や、現行のREDOログ・ファイルがクローズ状態のスレッドのものであっても発行できる点が異なります。

スタンバイ・データベースでは、STANDBY_FILE_MANAGEMENT初期化パラメータがAUTOに設定されており、いずれかのログ・ファイルがOracle Managed Filesである場合、Oracle Databaseによって、制御ファイル内のファイルと同数のOracle Managed Filesのログ・ファイルが作成されます。ログ・ファイル・メンバーは、ログ・ファイルの現行のデフォルト宛先に格納されます。

- アーカイブされていないREDOログを再利用する場合は、UNARCHIVEDを指定する必要があります。

  ノート:

  リカバリのために REDO ログが必要な場合に UNARCHIVED を指定すると、バックアップが使用できなくなります。

- ARCHIVELOGモードのデータベースでデータファイルをオフラインにする(DROPキーワードを使用せずにALTER DATABASE ... DATAFILE OFFLINEを指定する)場合、およびそのデータファイルをオンラインに戻す前に、クリアするアーカイブされていないログがデータファイルのリカバリに必要な場合は、UNRECOVERABLE DATAFILEを指定する必要があります。この場合、CLEAR LOGFILE文の完了後にデータファイルおよび表領域全体を削除する必要があります。

  メディア・リカバリで必要なログを、CLEAR LOGFILEを使用して消去しないでください。データベースのチェックポイント後のREDOを含むログをクリアする必要がある場合は、不完全メディア・リカバリを最初に実行する必要があります。オープンしているスレッドの現行のREDOログはクリアできます。クローズしているスレッドの現行のログは、そのスレッド内でログを切り替えればクリアできます。

  CLEAR LOGFILE文が、システム障害またはインスタンス障害による割込みを受けると、データベースがハングする場合があります。このような状況になった場合は、データベースの再起動後に、この文を再発行します。ログ・グループのあるメンバーにアクセスしようとした際、I/Oエラーによる障害が発生した場合は、そのメンバーを削除して他のメンバーを追加できます。

  関連項目:

  ログ・ファイルのクリア: 例

add_logfile_clauses

この句を使用すると、データベースにREDOログ・ファイル・グループを追加したり、既存のREDOログ・ファイル・グループに新しいメンバーを追加することができます。

ADD LOGFILE句

ADD LOGFILE句を使用すると、オンラインREDOログまたはスタンバイREDOログに1つ以上のREDOログ・ファイル・グループが追加されます。

関連項目:

- 新しいログ・ファイル・グループ名を指定しない場合にこの句によってOracle Managed Filesに戻される結果の詳細は、「CREATE DATABASE」の「LOGFILE句」を参照してください。
- REDOログ・ファイル・グループの追加: 例
- スタンバイREDOログの詳細は、『Oracle Data Guard概要および管理』を参照してください。

STANDBY

STANDBY句を使用すると、REDOログ・ファイル・グループがスタンバイREDOログに追加されます。この句を指定しないと、ログ・ファイル・グループがオンラインREDOログに追加されます。

INSTANCE

INSTANCE句は、Oracle Real Application Clusters (Oracle RAC)またはOracle RAC One Nodeデータベースのみに適用できます。REDOログ・ファイル・グループを追加するインスタンスの名前を指定します。インスタンス名は最大80文字の文字列です。Oracle Databaseは、指定されたインスタンスにマップされるスレッドを自動的に使用します。指定されたインスタンスにスレッドがマップされていない場合、Oracle Databaseはマップされていない使用可能なスレッドを自動的に取得して、そのインスタンスに割り当てます。この句を指定しないと、Oracle Databaseは、現行のインスタンスを指定した場合と同じコマンドを実行します。指定されたインスタンスにマップされた現行のスレッドが存在せず、マップされていない使用可能なスレッドも存在しない場合、エラーが戻されます。

THREAD

スタンバイREDOログにREDOログ・ファイル・グループを追加する場合は、THREAD句を使用して、ログ・ファイル・グループを特定のプライマリ・データベースREDOスレッドに割り当てます。どのREDOスレッドが開かれているかを判断するには、プライマリ・データベースでV$INSTANCEビューに対して問合せを行い、いずれか1つのスレッド番号を指定します。

オンラインREDOログにログ・ファイル・グループを追加する場合、THREAD句を使用して、ログ・ファイル・グループを特定のREDOスレッドに割り当てることもできます。この使用方法は現在非推奨になっています。INSTANCE句を使用すると、同じ結果になり、使用も簡単です。

GROUP

GROUP句を使用すると、すべてのスレッドのすべてのREDOログ・ファイル・グループの中でグループを一意に識別できます(この値の範囲は、1からCREATE DATABASE文のMAXLOGFILESで指定された値まで)。同一のGROUP値を持つREDOログ・ファイル・グループを複数追加することはできません。このパラメータを指定しない場合、値が自動的に生成されます。REDOログ・ファイル・グループのGROUP値は、動的パフォーマンス・ビューV$LOGで確認できます。

redo_log_file_spec

各redo_log_file_specには、1つ以上のメンバー(コピー)を含むREDOログ・ファイル・グループを指定します。新しいログ・ファイルにファイル名を指定しない場合、CREATE DATABASEの「LOGFILE句」に示すルールに従って、Oracle Managed Filesが作成されます。

関連項目:

- file_specification
- 動的パフォーマンス・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

ADD LOGFILE MEMBER句

ADD LOGFILE MEMBER句を使用すると、既存のREDOログ・ファイル・グループに新しいメンバーを追加できます。新しいメンバーをそれぞれ'filename'に指定します。すでにファイルが存在する場合、追加するメンバーはグループ内の他のメンバーと同じサイズである必要があり、REUSEを指定する必要があります。ファイルが存在しない場合、適切なサイズのファイルが作成されます。メディア障害によってグループのすべてのメンバーを失った場合は、そのグループにメンバーを追加することはできません。

STANDBY

スタンバイREDOログ・ファイル・グループにメンバーを追加する場合は、STANDBYを指定する必要があります。そうでない場合は、エラーが戻ります。

logfile_descriptor句を使用して、次のいずれかの方法で、既存のREDOログ・ファイル・グループを指定できます。

GROUP integer

REDOログ・ファイル・グループを識別するGROUPパラメータの値を指定します。

filename(s)

REDOログ・ファイル・グループのすべてのメンバーをリストします。ご使用のオペレーティング・システムの表記規則に従って、ファイル名を完全に指定する必要があります。

関連項目:

- 新しいログ・ファイル・グループ名を指定しない場合にこの句によってOracle Managed Filesに戻される結果の詳細は、「CREATE DATABASE」の「LOGFILE句」を参照してください。
- REDOログ・ファイル・グループ・メンバーの追加: 例

drop_logfile_clauses

この句を使用すると、REDOログ・ファイル・グループまたはREDOログ・ファイル・メンバーを削除できます。

DROP LOGFILE句

DROP LOGFILE句を使用すると、REDOログ・ファイル・グループのすべてのメンバーを削除できます。この句を使用してOracle Managed Filesを削除する場合、ディスクからもすべてのログ・ファイル・メンバーが削除されます。ADD LOGFILE MEMBER句と同様に、REDOログ・ファイル・グループを指定します。

- 現行のログ・ファイル・グループを削除する場合、最初にALTER SYSTEM SWITCH LOGFILE文を発行する必要があります。
- アーカイブが必要なREDOログ・ファイル・グループは、削除できません。
- REDOログ・ファイル・グループを削除すると、そのREDOスレッドのREDOログ・ファイル・グループが2つ未満になる場合は、削除できません。

関連項目:

「ALTER SYSTEM」および「ログ・ファイル・メンバーの削除: 例」

DROP LOGFILE MEMBER句

DROP LOGFILE MEMBER句を使用すると、1つ以上のREDOログ・ファイル・メンバーを削除できます。各'filename'には、ご使用のオペレーティング・システムのファイル名の表記規則に従って、メンバーを完全に指定する必要があります。

- 現行のログのログ・ファイルを削除する場合、最初にALTER SYSTEM SWITCH LOGFILE文を発行する必要があります。詳細は、「ALTER SYSTEM」を参照してください。
- この句では、有効なデータを含むREDOログ・ファイル・グループのすべてのメンバーを削除できません。このような操作には、DROP LOGFILE句を使用してください。

関連項目:

ログ・ファイル・メンバーの削除: 例

switch_logfile_clause

この句は、現在のデータベースのブロック・サイズと異なるブロック・サイズでデータベースをディスクに移行する場合に有効です。この句を使用すると、オープン状態とクローズ状態の両スレッドを含む外部で使用可能なすべてのスレッドに対して、異なるブロック・サイズにログ・ファイルを切り替えることができます。4KBセクターのディスクを使用するようデータベースを移行する場合は、integerに4096を指定する必要があります。データベースの移行を戻して512バイト・セクターのディスクを使用するようにする場合は、integerに512を指定する必要があります。

この句は、既存のALTER SYSTEM SWITCH LOGFILE文の拡張です。この文は、ログを単一スレッドに切り替えます。一方、句の方は、オープン状態とクローズ状態の両スレッドを含む外部で使用可能なすべてのスレッドに対して、ログ・ファイルを切り替えます。

この句を使用する前に、移行先のディスクと同じブロック・サイズで2つ以上のREDOログ・グループをあらかじめ作成しておく必要があります。

関連項目:

異なるブロック・サイズでのデータベースのディスクへの移行の詳細は、『Oracle Database管理者ガイド』および「ログ・ファイルの追加例」を参照してください。

supplemental_db_logging

この句を使用すると、ログ・ストリームへのサプリメンタル・データの追加を実行または停止するようにOracle Databaseに指示できます。

ADD SUPPLEMENTAL LOG句

ADD SUPPLEMENTAL LOG DATAを指定すると、最小限のサプリメンタル・ロギングを有効にできます。最小サプリメンタル・ロギングに加え、列データ・ロギングを使用可能にする場合、ADD SUPPLEMENTAL LOG supplemental_id_key_clauseを指定します。PL/SQLコールのサプリメンタル・ロギングを使用可能にする場合は、ADD SUPPLEMENTAL LOG supplemental_plsql_clauseを指定します。Oracle Databaseは、デフォルトでは最小サプリメンタル・ロギングまたはサプリメンタル・ロギングを使用可能にしません。

最小サプリメンタル・ロギングによって、LogMiner(およびLogMinerの技術に基づいた製品)には、連鎖行および様々な記憶域構成(クラスタ表など)をサポートするために十分な情報が確保されます。

あるデータベースで生成されたREDOが、別のデータベースでの変更の基となる(マイニングおよび適用される)場合(ロジカル・ス

タンバイ・データベースなど)、ROWIDではなく列データを使用して、影響を受ける行を識別する必要があります。この場合、supplemental_id_key_clauseを指定する必要があります。

サプリメンタル・ロギングが使用可能になっているかどうかを確認するには、V$DATABASEビューの適切な列を問い合せます。

この句は、データベースがオープンしているときに使用できます。ただし、パフォーマンスに影響するカーソル・キャッシュのすべてのDMLカーソルが、キャッシュが再移入されるまで無効になります。

CDBでこの句を使用する場合は、現在のコンテナがルートである必要があります。この場合、そのCDB全体に対して操作が実行されます。

supplemental_id_clauseの詳細は、「CREATE TABLE」の「supplemental_id_key_clause」を参照してください。

関連項目:

- ロジカル・スタンバイ・データベースをサポートするためのプライマリ・データベースでのサプリメンタル・ロギングの詳細は、『Oracle Data Guard概要および管理』を参照してください。
- supplemental_db_logging句の構文の使用例は、『Oracle Databaseユーティリティ』を参照してください。

DROP SUPPLEMENTAL LOG句

この句を使用すると、サプリメンタル・ロギングを停止できます。

- DROP SUPPLEMENTAL LOG DATAを指定すると、更新操作が発生するたびにREDOログ・ストリームに追加の最小ログ情報が置かれることを停止できます。Oracle Databaseがsupplemental_id_key_clauseで指定された列データ・サプリメンタル・ロギングを実行している場合、まずDROP SUPPLEMENTAL LOG supplemental_id_key_clauseで列データ・サプリメンタル・ロギングを停止してから、この句を指定する必要があります。
- DROP SUPPLEMENTAL LOG supplemental_id_key_clauseを指定すると、システム生成の補助ログ・グループの一部またはすべてを削除できます。削除する補助ログ・グループがsupplemental_id_key_clauseを使用して追加された場合、この句を指定する必要があります。
- PL/SQLコールのサプリメンタル・ロギングを使用禁止にする場合は、DROP SUPPLEMENTAL LOG supplemental_plsql_clauseを指定します。

CDBでこの句を使用する場合は、現在のコンテナがルートである必要があります。この場合、そのCDB全体に対して操作が実行されます。

ALTER DATABASEのADD SUPPLEMENTAL LOG DATA SUBSET DATABASE REPLICATIONは、影響の少ない最小サプリメンタル・ロギングを有効化します。

- このDDLを実行できるのは、enable_goldengate_replicationパラメータがTRUEで、データベースが19.0以上と互換のときだけです。
- このDDLは、DBレベルの他のサプリメンタル・ロギングDDLと同じように、DBレベルの最小サプリメンタル・ロギングを暗黙的に追加します。
- CDBの場合、このDDLはCDB$ROOTとプラガブル・データベースのどちらでも実行できます。
- CDB$ROOTで実行した場合、データベース全体で影響の少ない最小サプリメンタル・ロギングが有効になります。影響の少ない最小サプリメンタル・ロギングは、サブセット・データベース・レプリケーションのPDBレベル設定にかかわらず、すべ

てのプラガブル・データベースに対して有効になります。

- プラガブル・データベースで実行される場合、ALTER PLUGGABLE DATABASE ADD SUPPLEMENTAL LOG DATA SUBSET DATABASE REPLICATIONのときと同じです。詳細はALTER PLUGGABLE DATABASEを参照してください。

ALTER DATABASEのDROP SUPPLEMENTAL LOG DATA SUBSET DATABASE REPLICATIONは、影響の少ない最小サプリメンタル・ロギングを無効化します。

- このDDLを実行できるのは、enable_goldengate_replicationパラメータがTRUEで、データベースが19.0以上と互換のときだけです。
- 他のサプリメンタル・ログ・データは、明示的に有効化されている必要があります。このように制限すると、影響の少ない最小サプリメンタル・ロギングを無効にしても、最小サプリメンタル・ロギングが無効になることはありません。
- このDDLを実行すると、最小サプリメンタル・ロギングは現在の動作に戻ります。
- CDBの場合、このDDLはCDB$ROOTとプラガブル・データベースのどちらでも実行できます。
- CDB$ROOTで実行した場合、影響の少ない最小サプリメンタル・ロギングがデータベース・レベルで無効になります。プラガブル・データベースごとに、影響の少ないサプリメンタル・ロギングが有効になるかどうかは、サブセット・データベース・レプリケーションのPDBレベル設定によって決まります。
- プラガブル・データベースで実行される場合、動作はALTER PLUGGABLE DATABASE DROP SUPPLEMENTAL LOG DATA SUBSET DATABASE REPLICATIONのときと同じです。詳細はALTER PLUGGABLE DATABASEを参照してください。

関連項目:

サプリメンタル・ロギングの詳細は、『Oracle Data Guard概要および管理』を参照してください。

controlfile_clauses

controlfile_clausesを指定すると、制御ファイルを作成またはバックアップできます。

CREATE CONTROLFILE句

CREATE CONTROLFILE句を使用すると、制御ファイルを作成できます。

- PHYSICAL STANDBYを指定すると、物理データベースの管理に使用する制御ファイルを作成することになります。これは、STANDBYを指定していて、PHYSICALやLOGICALを指定していない場合のデフォルトです。
- LOGICAL STANDBYを指定すると、論理データベースの管理に使用する制御ファイルを作成することになります。
- FAR SYNC INSTANCEを指定すると、Data Guard遠隔同期インスタンスの管理に使用する制御ファイルを作成することになります。

ファイルがすでに存在している場合は、REUSEを指定してください。Oracle RAC環境では、制御ファイルを共有記憶域に配置する必要があります。

関連項目:

制御ファイル作成の詳細は、『Oracle Data Guard概要および管理』を参照してください。

BACKUP CONTROLFILE句

BACKUP CONTROLFILE句を使用すると、現行の制御ファイルのバックアップを取ることができます。この句を使用するとき、データベースをオープンまたはマウントしておく必要があります。

TO 'filename'

この句を使用して、制御ファイルのバイナリ・バックアップを指定します。filenameには、オペレーティング・システムの規則に従って完全なファイル名を指定する必要があります。指定したファイルがすでに存在している場合は、REUSEを指定します。Oracle RAC環境では、filenameを共有記憶域に配置する必要があります。

バイナリ・バックアップには、TO TRACEを指定した場合には取得されない情報が格納されています(アーカイブ・ログ履歴、読取り専用およびオフライン表領域のオフライン範囲、RMANを使用する場合のバックアップ・セットおよびコピーなど)。COMPATIBLE初期化パラメータが 10.2以上の場合は、バイナリの制御ファイル・バックアップに一時ファイルのエントリが含まれます。

TO TRACE

TO TRACEを指定すると、Oracle Databaseで、制御ファイルの物理バックアップが作成されるかわりに、トレース・ファイルにSQL文が書き込まれます。トレース・ファイルに記述されたSQL文を使用すると、データベースの起動、制御ファイルの再作成、データベースのリカバリやオープンなどの操作を、作成した制御ファイルに基づいて正しく実行できます。ブロック・チェンジ・トラッキングを使用可能にしているときにALTER DATABASE BACKUP CONTROLFILE TO TRACE文を発行すると、生成されるトレース・ファイルには、ブロック・チェンジ・トラッキングを再度使用可能にするコマンドが記述されます。

この文は暗黙的なALTER DATABASE REGISTER LOGFILE文を発行します。アーカイブ・ログ・ファイルが現行のアーカイブ・ログの宛先に存在する場合、この文によって、インカネーション・レコードが作成されます。

このトレース・ファイルには、アーカイブログの現在の宛先に存在しているログファイル用のALTER DATABASE REGISTER LOGFILE文も記述されています。これは、ログファイルを適用するREDOのブランチについて、暗黙的にデータベース・インカネーションのレコードを作成することになります。

制御ファイルのコピーがすべて失われた場合(または、制御ファイルのサイズを変更する場合)は、トレース・ファイルからスクリプト・ファイルに文をコピーし、必要に応じて文を編集できます。

- AS filenameを指定すると、標準トレース・ファイルではなくfilenameというファイルにトレース出力を格納できます。
- REUSEを指定すると、filenameと呼ばれる既存ファイルを上書きできます。
- RESETLOGSは、データベースの起動用としてトレース・ファイルに書き込まれたSQL文が、ALTER DATABASE OPEN RESETLOGSであることを示します。オンライン・ログが使用不可能な場合のみ、この設定は有効です。
- NORESETLOGSは、データベースの起動用としてトレース・ファイルに書き込まれたSQL文が、ALTER DATABASE OPEN NORESETLOGSであることを示します。オンライン・ログが使用可能な場合のみ、この設定は有効です。

オンライン・ログの今後の状態が予測できない場合は、RESETLOGSもNORESETLOGSも指定しないでください。この場合、両方のバージョンのスクリプトがトレース・ファイルに入れられるため、スクリプトが必要になった時点で、適切なバージョンを選択できます。

トレース・ファイルは、DIAGNOSTIC_DEST初期化パラメータによって決まるサブディレクトリに格納されます。CREATE CONTROLFILE文が書き込まれたトレース・ファイルの名前と場所は、アラート・ログで確認できます。V$DIAG_INFO動的パフォーマンス・ビューのNAMEおよびVALUE列を問い合せて、トレース・ファイルのディレクトリを検索することもできます。

関連項目:

アラート・ログの表示方法の詳細は、『Oracle Database管理者ガイド』を参照してください。

standby_database_clauses

この句を使用すると、スタンバイ・データベースをアクティブにする、または保護モードと非保護モードのいずれかを指定することができます。

関連項目:

フィジカルおよびロジカルのスタンバイ・データベースの説明およびスタンバイ・データベースの保守および使用に関する情報は、『Oracle Data Guard概要および管理』を参照してください。

activate_standby_db_clause

ACTIVATE STANDBY DATABASE句を使用すると、スタンバイ・データベースをプライマリ・データベースに変換できます。

ノート:

このコマンドを使用する前に、『Oracle Data Guard 概要および管理』を参照して使用方法を確認してください。

PHYSICAL

PHYSICALを指定すると、フィジカル・スタンバイ・データベースをアクティブにできます。これはデフォルトです。

LOGICAL

LOGICALを指定すると、ロジカル・スタンバイ・データベースをアクティブにできます。ロジカル・スタンバイ・データベースが複数ある場合は、すべてのスタンバイ・システムで同じログ・データが使用可能であることを確認する必要があります。

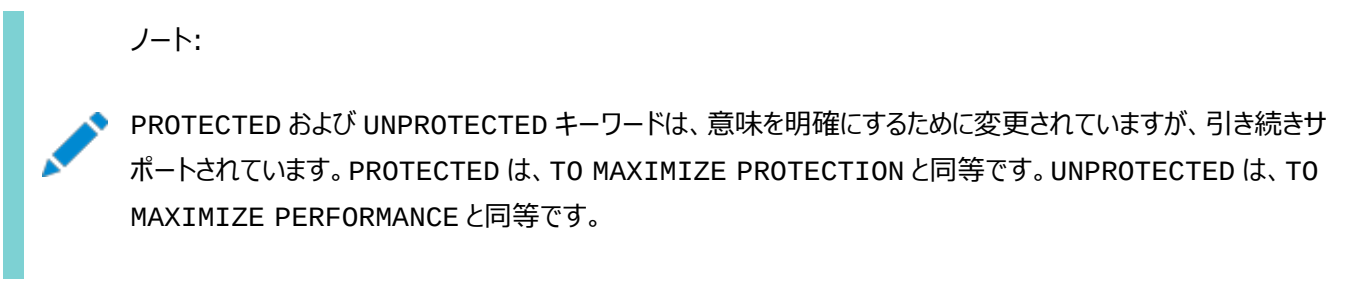
FINISH APPLY

この句は、ロジカル・スタンバイ・データベースのみに適用されます。この句は、ターミナル適用を開始するために使用します(ターミナル適用を実行すると、残りのREDOが適用され、ロジカル・スタンバイ・データベースがプライマリ・データベースと同じ状態になります)。ターミナル適用が完了すると、データベースは、ロジカル・スタンバイからプライマリ・データベースへのスイッチオーバーを完了します。

データの消失があってもデータベースをすぐにリストアする必要がある場合は、この句を指定しないでください。データベースは、ターミナル適用を行わずに、ロジカル・スタンバイからプライマリ・データベースへのスイッチオーバーを実行します。

maximize_standby_db_clause

この句では、このデータベース環境でのデータの保護レベルを指定します。この句をプライマリ・データベースから指定します。



ノート:

PROTECTED および UNPROTECTED キーワードは、意味を明確にするために変更されていますが、引き続きサポートされています。PROTECTED は、TO MAXIMIZE PROTECTION と同等です。UNPROTECTED は、TO MAXIMIZE PERFORMANCE と同等です。

TO MAXIMIZE PROTECTION

この設定は、最大保護モードを確立し、最高レベルのデータ保護を実現します。トランザクションのリカバリに必要なすべてのデータが、SYNCログ転送モードを使用するように構成された1つ以上のフィジカル・スタンバイ・データベースに書き込まれるまで、そのトランザクションはコミットされません。1つ以上のスタンバイ・データベースにREDOレコードを書き込めない場合、プライマリ・データベースは停止します。このモードでは、データ非消失は保証されますが、プライマリ・データベースのパフォーマンスおよび可用性に最大の影響を与える可能性があります。

最大保護モードの確立の制限事項

オープン状態のデータベースでTO MAXIMIZE PROTECTIONを指定できるのは、現在のデータ保護モードがMAXIMUM AVAILABILITYで、同期されているスタンバイ・データベースが少なくとも1つある場合のみです。

TO MAXIMIZE AVAILABILITY

この設定は、最大可用性モードを確立し、2番目に高いレベルのデータ保護を実現します。トランザクションのリカバリに必要なすべてのデータが、SYNCログ転送モードを使用するように構成された1つ以上のフィジカルまたはロジカル・スタンバイ・データベースに書き込まれるまで、そのトランザクションはコミットされません。最大保護モードとは異なり、1つ以上のスタンバイ・データベースにREDOレコードを書き込めない場合でも、プライマリ・データベースは停止しません。障害が修正され、スタンバイ・データベースがプライマリ・データベースと同一になるまで、データ保護レベルは最大パフォーマンス・モードまで下げられます。最大パフォーマンス・モードのプライマリ・データベースに障害が発生しないかぎり、このモードによってデータ非消失が保証されます。最大可用性モードは、プライマリ・データベースの可用性に影響を与えずに、最大レベルのデータ保護を実現します。

TO MAXIMIZE PERFORMANCE

この設定は、最大パフォーマンス・モードを確立します(これがデフォルトの設定です)。トランザクションがコミットされるのは、そのトランザクションのリカバリに必要なデータがスタンバイ・データベースに書き込まれる前です。したがって、プライマリ・データベースに障害が発生してREDOレコードをプライマリ・データベースからリカバリできなくなると、トランザクションが失われることがあります。このモードは、プライマリ・データベースのパフォーマンスに影響を与えないという条件の下で最大レベルのデータ保護を実現します。

データベースの現行モードを確認するには、V$DATABASE動的パフォーマンス・ビューのPROTECTION_MODE列を問い合せます。

関連項目:

スタンバイ・データベースのこれらの設定については、『Oracle Data Guard概要および管理』を参照してください。

register_logfile_clause

スタンバイ・データベースでREGISTER LOGFILE句を指定すると、障害が発生したプライマリ・データベースからログ・ファイルを手動で登録することができます。オペレーティング・システムのファイル・システム内の標準REDOログ・ファイル、またはOracle ASMディスク・グループのREDOログ・ファイルをリスト表示するには、file_specificationのredo_log_file_spec書式(「file_specification」を参照)を使用します。

ログ・ファイルのインカーネーションが不明である場合、REGISTER LOGFILE句は、インカネーション・レコードをV$DATABASE_INCARNATIONビューに追加します。新しく登録されたログ・ファイルのインカネーションが現行のRECOVERY_TARGET_INCARNATION#よりも高いRESETLOGS_TIMEを保持している場合、REGISTER LOGFILE句は、新しく追加されたインカネーション・レコードに対応するようにRECOVERY_TARGET_INCARNATION#も変更します。

OR REPLACE

OR REPLACEを指定すると、スタンバイ・データベースの既存のアーカイブ・ログ・エントリを更新できます(たとえばそのエントリの

位置やファイル指定が変更されたとき)。エントリのシステム変更番号は、正確に一致している必要があります。また、元のエントリは、管理スタンバイ・ログの送信メカニズムによって作成される必要があります。

FOR logminer_session_name

この句はStreams環境で役立ちます。LogMinerセッションを1つ指定して、そのセッションにログ・ファイルを登録できます。

switchover_clause

注意:

このコマンドを使用する前に、『Oracle Data Guard 概要および管理』を参照して使用方法を確認してください。

この句を使用すると、物理スタンバイ・データベースへのスイッチオーバーを実行できます。この句は、プライマリ・データベースから指定してください。target_db_nameには、スタンバイ・データベースのDB_UNIQUE_NAMEを指定します。

VERIFY

この句を使用すると、フィジカル・スタンバイ・データベースがスイッチオーバーできる状態になっていることを確認できます。この句は、プライマリ・データベースから指定してください。target_db_nameには、スタンバイ・データベースのDB_UNIQUE_NAMEを指定します。スタンバイ・データベースがスイッチオーバーできる状態のときには、「データベースが変更されました」のメッセージが返されます。それ以外の場合は、スタンバイ・データベースをスイッチオーバーするための準備に役立つエラー・メッセージが返されます。

FORCE

この句は、前回のスイッチオーバー・コマンドが失敗して、プライマリ・データベースなしで構成が作成された場合に使用します。この句は、プライマリ・データベースに変換する物理スタンバイ・データベースから指定します。target_db_nameには、プライマリ・データベースに変換するデータベースのDB_UNIQUE_NAMEを指定します。

failover_clause

注意:

このコマンドを使用する前に、『Oracle Data Guard 概要および管理』を参照して使用方法を確認してください。

この句を使用すると、物理スタンバイ・データベースへのフェイルオーバーを実行できます。この句は、スタンバイ・データベースから指定します。target_db_nameには、スタンバイ・データベースのDB_UNIQUE_NAMEを指定します。

FORCE

この句は、フェイルオーバー先がData Guard遠隔同期インスタンスでサービス提供されている場合にのみ意味を持ちます。この句は、前回のフェイルオーバー・コマンドが失敗して、失敗の理由が解決できなかったときに使用します。これにより、フェイルオーバーでData Guard遠隔同期インスタンスとの対話時に発生した失敗を無視して、可能ならばフェイルオーバーを継続するように指示します。

commit_switchover_clause

この句を使用すると、Data Guard構成でのデータベース・ロールの遷移を実行できます。

注意:

このコマンドを使用する前に、『Oracle Data Guard 概要および管理』を参照して使用方法を確認してください。

PREPARE TO SWITCHOVER

この句は、プライマリ・データベースがロジカル・スタンバイ・データベースになる準備またはロジカル・スタンバイ・データベースがプライマリ・データベースになる準備を行います。

- プライマリ・データベースでPREPARE TO SWITCHOVER TO LOGICAL STANDBYを指定します。
- ロジカル・スタンバイ・データベースでPREPARE TO SWITCHOVER TO PRIMARY DATABASEを指定します。

COMMIT TO SWITCHOVER

この句は、プライマリ・データベースをスタンバイ・データベース・ロールに切り替えるか、またはスタンバイ・データベースをプライマリ・データベース・ロールに切り替えます。

- プライマリ・データベースでCOMMIT TO SWITCHOVER TO PHYSICAL STANDBYまたはCOMMIT TO SWITCHOVER TO LOGICAL STANDBYを指定します。
- スタンバイ・データベースでCOMMIT TO SWITCHOVER TO PRIMARY DATABASEを指定します。

PHYSICAL

この句は常にオプションです。この句をCOMMIT TO SWITCHOVER TO PRIMARY句とともに使用することは、非推奨になりました。

LOGICAL

この句は、プライマリ・データベースをロジカル・スタンバイ・データベース・ロールに切り替えるときに、PREPARE TO SWITCHOVER句またはCOMMIT TO SWITCHOVER句とともに指定します。この句をCOMMIT TO SWITCHOVER TO PRIMARY句とともに使用することは、非推奨になりました。

WITH SESSION SHUTDOWN

この句は、データベース・ロールの遷移を実行する前に、すべてのデータベース・セッションをクローズし、コミットされていないトランザクションをロールバックします。

WITHOUT SESSION SHUTDOWN

この句は、データベース・セッションが存在する場合、要求されたロールの遷移が発生しないようにします。これはデフォルトです。

WAIT

この句を指定すると、ロールの遷移が完了するのを待機してから制御がユーザーに戻されます。

NOWAIT

この句を指定すると、ロールの遷移が完了するのを待機しないで制御がユーザーに戻されます。これはデフォルトです。

CANCEL

この句を指定すると、以前に指定したPREPARE TO SWITCHOVER文の影響が無効にされます。

関連項目:

プライマリ・データベースとスタンバイ・データベース間のスイッチオーバーの詳細は、『Oracle Data Guard概要および管理』を参照してください。

start_standby_clause

START LOGICAL STANDBY APPLY句を指定すると、ロジカル・スタンバイ・データベースへのREDOログの適用を開始できます。この句では、PL/SQLコールのロギング以外に、主キー、一意索引および一意制約のサプリメンタル・ロギングが使用可能になります。

- 現在のスタンバイREDOログ・ファイルのREDOデータを適用する場合は、IMMEDIATEを指定します。
- この適用において遅延を無視するようにOracle Databaseに指示する場合は、NODELAYを指定します。これは、プライマリ・データベースが存在せず、PL/SQLコールの実行が必要になる場合に便利です。
- 初めてスタンバイ・データベースにログを適用する場合は、INITIALを指定します。
- 次の2つの状況では、NEW PRIMARY句が必要となります。
    - ロジカル・スタンバイへのフェイルオーバー時に、フェイルオーバー操作に加わっていないロジカル・スタンバイ、およびロジカル・スタンバイ・データベースとして回復された後の古いデータベースでこの句を指定します。
    - (準備されていないスイッチオーバー操作を使用する)ロジカル・スタンバイ・データベースを使用してローリング・アップグレードを実行している場合は、元のプライマリ・データベースが新しいデータベース・ソフトウェアにアップグレードされた後でこの句を指定します。
- イベント表の最後のトランザクションをスキップして適用を再開する場合は、SKIP FAILED [TRANSACTION]を指定します。
- スタンバイREDOログ・ファイル情報をアーカイブ・ログに強制処理する場合は、FINISHを指定します。プライマリ・データベースが使用できなくなった場合、REDOログ・ファイルのデータを適用できます。

stop_standby_clause

この句を使用すると、ログ適用サービスを停止できます。この句は、フィジカル・スタンバイ・データベースではなく、ロジカル・スタンバイ・データベースのみに適用されます。STOP句を使用すると、適用が正常に停止されます。

convert_database_clause

この句を使用すると、データベースを別の形式に変換できます。

- CONVERT TO PHYSICAL STANDBYを使用すると、プライマリ・データベース、ロジカル・スタンバイ・データベースまたはスナップショット・スタンバイ・データベースをフィジカル・スタンバイ・データベースへ変換できます。

  この句を指定する前に、次のステップを実行します。

    - Oracle Real Application Clusters(Oracle RAC)データベースで、1つを除いてすべてのインスタンスを停止します。
    - データベースがマウントされ、オープンしていないことを確認します。

  データベースは変換後にディスマウントされ、再起動する必要があります。
- CONVERT TO SNAPSHOT STANDBYを使用すると、フィジカル・スタンバイ・データベースをスナップショット・スタンバイ・データベースへ変換できます。

  この句を指定する前に、REDO Applyを停止していることを確認してください。

ノート:

スナップショット・スタンバイ・データベースは、フィジカル・スタンバイ・データベースに変換される前に読取り/書込みモードで 1 回以上オープンされている必要があります。

関連項目:

スタンバイ・データベースの詳細は、『Oracle Data Guard概要および管理』を参照してください。

default_settings_clauses

データベースのデフォルト設定を変更できます。

DEFAULT EDITION句

この句を使用すると、指定したエディションをデータベースのデフォルト・エディションとして設定できます。指定するエディションをあらかじめ作成し、USABLEにしておく必要があります。変更内容は即座に反映され、Oracle RAC環境のすべてのノードから参照可能になります。指定したエディションで新しいデータベース・セッションが自動的に開始されます。新しい設定は、データベースを停止して起動した後も保持されます。

データベース・デフォルト・エディションとしてエディションを指定する場合、指定されたエディションに対してUSEオブジェクト権限がロールPUBLICに付与されていますが、すべてのユーザーがエディションを使用できます。

データベースの現在のデフォルト・エディションを確認するには、次の問合せを使用します。

```
SELECT PROPERTY_VALUE FROM DATABASE_PROPERTIES
  WHERE PROPERTY_NAME = 'DEFAULT_EDITION';
```

関連項目:

エディションの詳細は「CREATE EDITION」、エディションをUSABLEに指定する方法の詳細は『Oracle Database PL/SQL言語リファレンス』を参照してください。

CHARACTER SET、NATIONAL CHARACTER SET

ALTER DATABASE文を使用して、データベース文字セットまたは各国語文字セットを変更することはできなくなりました。データベース文字セットの移行の詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

SET DEFAULT TABLESPACE句

この句を使用すると、以降に作成する表領域のデフォルト・タイプを指定または変更できます。表領域がbigfileかsmallfileかを指定するには、BIGFILEまたはSMALLFILEを指定します。

- bigfile表領域に格納されるのは、1つのデータファイルまたは一時ファイルのみであり、このファイルには最大約40億(2^{32})ブロックを格納できます。データファイルまたは一時ファイル1つ当たりの最大サイズは、32Kブロックの表領域の場合は128TB、8Kブロックの表領域の場合は32TBです。
- smallfile表領域は、Oracleの従来の表領域であり、1022のデータファイルまたは一時ファイルを含めることができます。それぞれのファイルは、最大で約400万(2^{22})のブロックを格納できます。

関連項目:

- bigfile表領域の詳細は、『Oracle Database管理者ガイド』を参照してください。
- 表領域のデフォルト・タイプの設定：例

DEFAULT TABLESPACE句

この句を使用すると、データベースのデフォルトの永続表領域を構築または変更できます。作成済の表領域を指定する必要があります。この操作の完了後、すべての非SYSTEMユーザーに、新しいデフォルトの一時表領域が再度割り当てられます。これらのユーザーがこれ以降作成するすべてのオブジェクトは、デフォルトで、新しいデフォルトの表領域に格納されます。以前に指定したデフォルトの表領域を置き換える場合、以前作成したオブジェクトを古いデフォルトの表領域から新しいデフォルトの表領域に移動した後で、必要に応じて古いデフォルトの表領域を削除することができます。

DEFAULT [LOCAL] TEMPORARY TABLESPACE句

この句を指定すると、データベースのデフォルトの共有一時表領域を、新しい表領域または表領域グループに変更するか、デフォルトのローカル一時表領域を新しい表領域に変更できます。

- `tablespace`を指定すると、データベースの新しいデフォルトの一時表領域を指定できます。この操作の完了後、以前のデフォルトの一時表領域が割り当てられているすべてのユーザーに、新しいデフォルトの一時表領域が再度割り当てられます。その後、必要に応じて以前のデフォルトの一時表領域を削除することができます。デフォルトの共有一時表領域を変更するには、`DEFAULT TEMPORARY TABLESPACE`を指定します。デフォルトのローカル一時表領域を変更するには、`DEFAULT LOCAL TEMPORARY TABLESPACE`を指定します。
- `tablespace_group_name`を指定すると、`tablespace_group_name`で指定されている表領域グループのすべての表領域を、データベースのデフォルトの共有一時表領域として指定できます。この操作の完了後、デフォルトの一時表領域を明示的に割り当てられていないユーザーは、`tablespace_group_name`に含まれる任意の表領域に一時セグメントを作成できます。以前のデフォルトの一時表領域がデフォルトの一時表領域グループに含まれる場合、以前の一時表領域を削除することはできません。ローカル一時表領域を表領域グループに含めることはできません。

現行のデフォルトの一時表領域またはデフォルトの一時表領域グループの名前を確認するには、`ALL_`、`DBA_`または`USER_USERS`データ・ディクショナリ・ビューの`TEMPORARY_TABLESPACE`列を問い合せます。

デフォルトの一時表領域の制限事項

デフォルトの一時表領域には、次の制限事項があります。

- デフォルトの一時表領域として割り当てる表領域、または再度割り当てる表領域は、標準的なブロック・サイズである必要があります。
- SYSTEM表領域がローカル管理される場合、デフォルトの一時表領域として指定する表領域も、ローカル管理される必要があります。

関連項目:

- 表領域グループの詳細は、『Oracle Database管理者ガイド』を参照してください。
- デフォルトの一時表領域の変更：例

instance_clauses

Oracle Real Application Clusters環境では、`ENABLE INSTANCE`を指定すると、指定したデータベース・インスタンスに

マップされたスレッドが有効になります。使用可能にできるスレッドは、2つ以上のREDOログ・ファイル・グループを持つスレッドのみです。また、データベースはオープンされている必要があります。

DISABLE INSTANCEを指定すると、指定したデータベース・インスタンスにマップされたスレッドを使用禁止にできます。インスタンス名は最大80文字の文字列です。指定したインスタンスにマップされているスレッドがない場合、エラーが戻されます。データベースは、オープンされている必要がありますが、指定したスレッドを使用しているインスタンスでデータベースをマウント済の場合は、そのスレッドを使用禁止にできません。

関連項目:

インスタンスを使用可能または使用禁止にする場合の詳細は、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください。

RENAME GLOBAL_NAME句

RENAME GLOBAL_NAMEを指定すると、データベースのグローバル名を変更できます。データベースは、オープンされている必要があります。databaseには、データベースの新しい名前を8バイト以内の長さで指定します。オプションのdomainには、ネットワーク階層におけるデータベースの有効な位置を指定します。ドメイン名を指定する場合は、ドメイン名のコンポーネントを有効な識別子にする必要があります。有効な識別子の詳細は、「データベース・オブジェクトのネーミング規則」を参照してください。

ノート:

データベース名を変更しても、リモート・データベース内の既存のデータベース・リンク、シノニム、ストアド・プロシージャ、ストアド・ファンクションからのユーザーのデータベースに対するグローバル参照は変更されません。これらの参照を変更するのは、リモート・データベース管理者の責任です。

関連項目:

グローバル・データベース名の変更: 例

BLOCK CHANGE TRACKING句

ブロック・チェンジ・トラッキング機能を使用すると、Oracle Databaseでは、プライマリ・データベースおよび任意のフィジカル・スタンバイ・データベースの両方で、すべてのデータベース更新の物理的な場所を追跡できます。トラッキングを実行する各データベースで、ブロック・チェンジ・トラッキングを使用可能にする必要があります。トラッキング情報は、ブロック・チェンジ・トラッキング・ファイルという別のファイルに保持されます。Oracle Managed Filesを使用している場合、Oracle Databaseは、DB_CREATE_FILE_DESTで指定した場所にブロック・チェンジ・トラッキング・ファイルを自動的に作成します。Oracle Managed Filesを使用していない場合、チェンジ・トラッキング・ファイルの名前を指定する必要があります。Oracle Databaseは、増分バックアップのパフォーマンス向上などの内部タスクのために、チェンジ・トラッキング・データを使用します。ブロック・チェンジ・トラッキングを使用可能にする場合、データベースは、ARCHIVELOGモードまたはNOARCHIVELOGモードで、オープンされているかマウント済である必要があります。

ENABLE BLOCK CHANGE TRACKING

この句を指定すると、ブロック・チェンジ・トラッキングが有効になり、Oracle Databaseによってブロック・チェンジ・トラッキング・ファイルが作成されます。

- USING FILE 'filename'を指定すると、ブロック・チェンジ・トラッキング・ファイルの名前を(Oracle Databaseが生

成するのではなく)自分で指定できます。Oracle Managed Filesを使用していない場合、この句を指定する必要があります。

- REUSEを指定すると、同じ名前の既存のブロック・チェンジ・トラッキング・ファイルを上書きできます。

ノート:

スタンバイ・データベースでは、ブロック・チェンジ・トラッキングは、管理リカバリの開始時にのみ有効になります。コマンドの発行時にスタンバイ・データベースでリカバリがすでにアクティブになっている場合は、高速増分バックアップを利用するために、リカバリを停止してから再起動する必要があります。

DISABLE BLOCK CHANGE TRACKING

この句を指定すると、Oracle Databaseによるチェンジ・トラッキングを停止し、既存のブロック・チェンジ・トラッキング・ファイルを削除できます。

関連項目:

ブロック・チェンジ・トラッキングの設定の詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』および「ブロック・チェンジ・トラッキングの有効化および無効化: 例」を参照してください。

[NO] FORCE FULL DATABASE CACHING

この句を使用して、強制完全データベース・キャッシュ・モードを有効化または無効化します。自動で行われるデフォルト・モードとは異なり、全データベース・キャッシュ強制モードでは、NOCACHE LOBを含むデータベース全体がバッファ・キャッシュへのキャッシュの対象であると見なします。

データベースは、マウントされているがオープンされていない状態であることが必要です。Oracle RAC環境では、データベースをマウントする必要がありますが、現在のインスタンスでオープン状態ではなく、すべての他のインスタンスでアンマウントする必要があります。

- FORCE FULL DATABASE CACHINGを指定して、強制完全データベース・キャッシュ・モードを有効化します。
- NO FORCE FULL DATABASE CACHINGを指定して、強制完全データベース・キャッシュ・モードを無効化します。これがデフォルト・モードです。

V$DATABASE動的パフォーマンス・ビューのFORCE_FULL_DB_CACHING列を問い合せて強制完全データベース・キャッシュ・モードを有効化するかどうかを判断できます。

関連項目:

- 強制完全データベース・キャッシュ・モードの詳細は、『Oracle Database概要』を参照してください。
- 強制完全データベース・キャッシュ・モードを有効化する方法を学習するには、『Oracle Database管理者ガイド』を参照してください。
- V$DATABASE動的パフォーマンス・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

CONTAINERS DEFAULT TARGET

この句を使用して、CDB内のDML文のデフォルト・コンテナを指定します。CDBルートに接続している必要があります。

- container_nameには、デフォルト・コンテナの名前を指定します。デフォルト・コンテナには、CDBルート、PDB、アプリケーション・ルートまたはアプリケーションPDBを含む、CDB内の任意のコンテナを指定できます。指定できるデフォルト・コンテナは1つのみです。
- NONEを指定した場合は、デフォルト・コンテナはCDBルートです。これはデフォルトです。

WHERE句でコンテナを指定せずにDML文がCDBルートで発行されている場合、DML文はCDBのデフォルト・コンテナに影響します。

flashback_mode_clause

この句を使用すると、データベースをFLASHBACKモードにすることや、元のモードに戻すことができます。この句を指定できるのは、データベースがARCHIVELOGモードで、かつデータベースの高速リカバリ領域が準備済の場合のみです。この句は、データベースがマウントされているかオープンされているときに指定できます。フィジカル・スタンバイ・データベースに対しては、REDO Applyがアクティブな場合はこの句を指定できません。

関連項目:

フラッシュバック操作のための高速リカバリ領域の準備の詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

FLASHBACK ON

この句を使用すると、データベースはFLASHBACKモードになります。データベースがFLASHBACKモードのとき、Oracle Databaseは、高速リカバリ領域に自動的にフラッシュバック・データベース・ログを作成し、これを管理します。その後、SYSDBAシステム権限を持つユーザーは、FLASHBACK DATABASE文を発行できます。

FLASHBACK OFF

この句を使用すると、データベースはFLASHBACKモードから戻されます。Oracle Databaseは、フラッシュバック・データのロギングを停止し、既存のすべてのフラッシュバック・データベース・ログを削除します。FLASHBACK DATABASEを発行すると、エラーが発生して実行できません。

undo_mode_clause

この句は、CDBに接続している場合にのみ有効です。CDBのUNDOモードを変更できます。CDBはOPEN UPGRADEモードである必要があります。

- LOCAL UNDO ONを指定すると、ローカルUNDOモードを使用するようにCDBが変更されます。
- LOCAL UNDO OFFを指定すると、共有UNDOモードを使用するようにCDBが変更されます。

関連項目:

- この句のセマンティクスの詳細は、「CREATE DATABASE」の「undo_mode_clause」を参照してください。
- ローカルUNDOモードまたは共有UNDOモードを使用するようにCDBを構成するステップの詳細は、『Oracle Database管理者ガイド』を参照してください。

set_time_zone_clause

この句のセマンティクスは、CREATE DATABASE文およびALTER DATABASE文で同じです。この句をALTER DATABASEと併用すると、データベースのタイムゾーンがリセットされます。データベースのタイムゾーンを確認するには、組込み関数

「DBTIMEZONE」を問い合せます。この句を使用してタイムゾーンを設定または変更した後、新しいタイムゾーンを有効にするためにデータベースを再起動する必要があります。

すべての新しいTIMESTAMP WITH LOCAL TIME ZONEデータは、ディスクに格納されるときにデータベースのタイムゾーンに正規化されます。データベースの既存のデータは、自動的に新しいタイムゾーンに更新されません。したがって、データベースにTIMESTAMP WITH LOCAL TIME ZONEデータが存在する場合は、データベースのタイムゾーンをリセットすることはできません。TIMESTAMP WITH LOCAL TIME ZONEデータを削除またはエクスポートしてからデータベースのタイムゾーンをリセットする必要があります。このような理由から、データが格納されているデータベースのタイムゾーンは変更しないことをお薦めします。

この句の詳細は、「CREATE DATABASE」の「set_time_zone_clause」を参照してください。

security_clause

security_clause(GUARD)を使用すると、データベースのデータが変更されないように保護できます。現行のセッションでこの設定を上書きするには、ALTER SESSION DISABLE GUARD文を使用します。詳細は、「ALTER SESSION」を参照してください。

ALL

ALLを指定すると、SYS以外のすべてのユーザーが、データベースの内容を変更できなくなります。

STANDBY

STANDBYを指定すると、SYS以外のすべてのユーザーが、ロジカル・スタンバイで管理されるデータベース・オブジェクトを変更できなくなります。この設定は、ロジカル・スタンバイによってレプリケートされる前に、データをレポート操作によって変更できるようにする場合に便利です。

関連項目:

ロジカル・スタンバイの詳細は、『Oracle Data Guard概要および管理』を参照してください。

NONE

NONEを指定すると、データベースのすべてのデータについて、通常のセキュリティを設定できます。

ノート:

ロジカル・スタンバイ・データベースには、この設定をできるだけ使用しないでください。

prepare_clause

- この句を使用して、データベースのミラー・コピーを準備します。作成されるファイル・グループを識別するために mirror_nameを指定する必要があります。ファイル・グループには、準備されたファイルのすべてが含まれています。
- REDUNDANCYオプションのEXTERNAL、NORMALまたはHIGHによって準備されるコピーの数を指定します。
- ミラーの冗長性を指定しない場合、ソース・データベースの冗長性が使用されます。

データベースの準備: 例

```
ALTER DATABASE db_name PREPARE MIRROR COPY mirror_name WITH HIGH REDUNDANCY
```

drop_mirror_copy

この句を使用して、PREPARE文によって作成されたデータのミラー・コピーを破棄します。準備操作に使用したものと同じミラー名を指定する必要があります。

すでにデータベースがCREATE DATABASE 文またはCREATE PLUGGABLE DATABASE文によって分割されている場合は、この句を使用してデータベースを削除することはできません。

lost_write_protection

データファイルの消失書込み保護を有効にするには、この句を指定します。データファイルの消失書込み保護を有効化、削除および一時停止できます。

例: データファイルの消失書込み保護の有効化

この例では、データファイルtd_file.dfの消失書込み保護をオンにします。

```
ALTER DATABASE DATAFILE td_file.df ENABLE LOST WRITE PROTECTION
```

消失書込みデータベースが消失することに注意してください。現行のデータファイルの内容で初期化されません。

データファイルの消失書込み保護は、REMOVEオプションまたはSUSPENDオプションの2つの方法で無効化できます。

- REMOVEオプションを使用すると、データファイルの消失書込み保護が停止します。また、シャドウ表領域のトラッキング・データなど、消失書込み保護へのすべての参照が削除されます。

  例: データファイルの消失書込み保護の削除

  ```
  ALTER DATABASE DATAFILE td_file.df REMOVE LOST WRITE PROTECTION
  ```

- SUSPENDオプションを使用すると、更新および消失書込みチェックは無効になりますが、シャドウ表領域のトラッキング・データは残されます。消失書込み保護を短時間一時停止すると、一時停止期間中はデータファイルの消失書込み保護が停止されます。つまり、消失書込みデータは収集されず、ブロックはチェックされません。後でデータファイルの消失書込み保護を有効にすると、一時停止期間中にデータファイル内のブロックに対して行われたSCNの更新のレコードはありません。SUSPENDオプションでは、消失書込み記憶域の割当て解除は行われないことに注意してください。

  例: データファイルの消失書込み保護の一時停止

  ```
  ALTER DATABASE DATAFILE td_file.df SUSPEND LOST WRITE PROTECTION
  ```

コンテナ・データベースおよびプラガブル・データベースの消失書込み保護を有効化できます。

例: データベースの消失書込み保護の有効化

```
ALTER DATABASE ENABLE LOST WRITE PROTECTION
```

例: データベースの消失書込み保護の無効化

```
ALTER DATABASE DISABLE LOST WRITE PROTECTION
```

データベースの消失書込み保護を無効化しても、消失書込み記憶域の割当て解除は行われないことに注意してください。消失書込み記憶域の割当て解除を行うには、DROP TABLESPACE文を使用する必要があります。

cdb_fleet_clauses

様々なCDBのコレクションのリードCDBを設定するには、cdb_fleet_clausesを指定します。

lead_cdb_clause

この句を使用して、CDBをCDBフリート内のリードCDBとして指定します。データベース・プロパティのLEAD_CDBは、現行のCDBがリードCDBであることを示し、DATABASE_PROPERTIESビューで確認できます。

SYS_CONTEXTにはIS_LEAD_CDBという新しいパラメータがあり、これを使用して、現在のセッションがCDBフリート内のリードCDBに接続されているかどうかを判別できます。

lead_cdb_uri_clause

この句を使用して、CDBフリート内のリードCDBの接続URIを指定します。これは、フリートのリードCDBにメンバーCDBを登録するために使用されます。

dblinkで指定されるデータベース・リンク名は、CDBフリートに参加するメンバーCDBのCDBルートに存在している必要があります。これは、フリート内のリードCDBとPDBメタデータを同期化するために使用されます。

指定されたuri_stringはLEAD_CDB_URIというデータベース・プロパティとして保管され、DATABASE_PROPERTIESビューで確認できます。

SYS_CONTEXTにはIS_LEAD_CDBという新しいパラメータがあり、これを使用して、現在のセッションがCDBフリート内のメンバーCDBに接続されているかどうかを判別できます。

property_clause

DATABASE_PROPERTIESまたはCDB_PROPERTIESビューで公開されるデータベース・プロパティを設定または削除するには、この句を指定します。

例

READ ONLY / READ WRITE: 例

次の文は、データベースを読取り専用モードでオープンします。

```
ALTER DATABASE OPEN READ ONLY;
```

次の文は、データベースを読み書き両用モードでオープンし、オンラインREDOログをクリアします。

```
ALTER DATABASE OPEN READ WRITE RESETLOGS;
```

パラレル・リカバリ処理の使用方法: 例

次の文は、パラレル・リカバリ処理を使用して表領域リカバリを実行します。

```
ALTER DATABASE
   RECOVER TABLESPACE tbs_03
   PARALLEL;
```

REDOログ・ファイル・グループの追加: 例

次の文は、2つのメンバーを含むREDOログ・ファイル・グループを追加し、GROUPパラメータの値に3を指定してこのグループを識別します。

```
ALTER DATABASE
  ADD LOGFILE GROUP 3
    ('diska:log3.log' ,
     'diskb:log3.log') SIZE 50K;
```

次の文は、2つのメンバーを含むREDOログ・ファイル・グループをスレッド5(Real Application Clusters環境内)に追加して、このグループにGROUPパラメータ値4を割り当てます。

```
ALTER DATABASE
    ADD LOGFILE THREAD 5 GROUP 4
```

```
        ('diska:log4.log',
         'diskb:log4:log');
```

REDOログ・ファイル・グループ・メンバーの追加: 例

次の文は、前述の例で追加したREDOログ・ファイル・グループに1つのメンバーを追加します。

```
ALTER DATABASE
   ADD LOGFILE MEMBER 'diskc:log3.log'
   TO GROUP 3;
```

ログ・ファイル・メンバーの削除: 例

次の文は、前述の例で追加したREDOログ・ファイル・メンバーの1つを削除します。

```
ALTER DATABASE
    DROP LOGFILE MEMBER 'diskb:log3.log';
```

次の文は、REDOログ・ファイル・グループ3のすべてのメンバーを削除します。

```
ALTER DATABASE DROP LOGFILE GROUP 3;
```

ログ・ファイル・メンバーの名前変更: 例

次の文は、REDOログ・ファイル・メンバーの名前を変更します。

```
ALTER DATABASE
    RENAME FILE 'diskc:log3.log' TO 'diskb:log3.log';
```

この例では、REDOログ・グループ・メンバーのファイルが、別のファイルに変更されただけです。ファイル名が、実際に`diskc:log3.log`から`diskb:log3.log`に変更されたわけではありません。この文を発行する前に、オペレーティング・システムでこのファイル名を変更する必要があります。

表領域のデフォルト・タイプの設定: 例

次の文は、それ以降に作成される表領域が、デフォルトでbigfile表領域として作成されることを指定します。

```
ALTER DATABASE
    SET DEFAULT BIGFILE TABLESPACE;
```

デフォルトの一時表領域の変更: 例

次の文は、`tbs_05`表領域(「一時表領域の作成: 例」で作成)をデータベースのデフォルトの一時表領域にします。この文は、作成時に何も指定されていない場合にデフォルトの一時表領域を作成するか、既存のデフォルトの一時表領域を`tbs_05`に置き換えます。

```
ALTER DATABASE
   DEFAULT TEMPORARY TABLESPACE tbs_05;
```

または、表領域グループを使用して、表領域グループをデフォルトの一時表領域に定義することもできます。次の文は、表領域グループ`tbs_group_01`の表領域(「表領域グループへの一時表領域の追加: 例」で作成)をデータベースのデフォルトの一時表領域にします。

```
ALTER DATABASE
   DEFAULT TEMPORARY TABLESPACE tbs_grp_01;
```

新しいデータファイルの作成: 例

次の文は、ファイル`tbs_f03.dbf`に基づいて新しいデータファイル`tbs_f04.dbf`を作成します。新しいデータファイルを作成する前に、既存のデータファイル(またはこのデータファイルが存在する表領域)をオフラインにする必要があります。

```
ALTER DATABASE
    CREATE DATAFILE 'tbs_f03.dbf'
                 AS 'tbs_f04.dbf';
```

一時ファイルの操作: 例

次の例では、「データファイルおよび一時ファイルの追加と削除: 例」で作成された一時ファイルtemp02.dbfをオフラインにして、名前を変更します。

```
ALTER DATABASE TEMPFILE 'temp02.dbf' OFFLINE;
ALTER DATABASE RENAME FILE 'temp02.dbf' TO 'temp03.dbf';
```

一時ファイルの名前を変更する文では、まずファイル`temp03.dbf`をオペレーティング・システム上に作成する必要があります。

グローバル・データベース名の変更: 例

次の文は、データベースのグローバル名を変更し、データベース名とドメインの両方を含めます。

```
ALTER DATABASE
    RENAME GLOBAL_NAME TO demo.world.example.com;
```

ブロック・チェンジ・トラッキングの有効化および無効化: 例

次の文は、ブロック・チェンジ・トラッキングを有効にし、`tracking_file`というブロック・チェンジ・トラッキング・ファイルを作成(すでに存在する場合は上書き)します。

```
ALTER DATABASE
  ENABLE BLOCK CHANGE TRACKING
    USING FILE 'tracking_file' REUSE;
```

次の文は、ブロック・チェンジ・トラッキングを使用禁止にし、既存のブロック・チェンジ・トラッキング・ファイルを削除します。

```
ALTER DATABASE
  DISABLE BLOCK CHANGE TRACKING;
```

データファイルのサイズの変更: 例

次の文は、データファイル`diskb:tbs_f5.dbf`のサイズの変更を試行します。

```
ALTER DATABASE
    DATAFILE 'diskb:tbs_f5.dbf' RESIZE 10 M;
```

ログ・ファイルのクリア: 例

次の文は、ログ・ファイルをクリアします。

```
ALTER DATABASE
    CLEAR LOGFILE 'diskc:log3.log';
```

データベースのリカバリ: 例

次の文は、データベース全体を完全にリカバリし、必要な新しいアーカイブREDOログ・ファイル名を生成します。

```
ALTER DATABASE
  RECOVER AUTOMATIC DATABASE;
```

次の文は、Oracle Databaseが適用するREDOログ・ファイル名を明示的に指定します。

```
ALTER DATABASE
    RECOVER LOGFILE 'diskc:log3.log';
```

次の文は、時間ベースのデータベース・リカバリを実行します。

```
ALTER DATABASE
    RECOVER AUTOMATIC UNTIL TIME '2001-10-27:14:00:00';
```

データベースは、2001年10月27日の午後2時の状態にリカバリされます。

表領域のリカバリ例は、「パラレル・リカバリ処理の使用方法：例」を参照してください。

ALTER DATABASE DICTIONARY

目的

不明瞭化されたデータベース・リンクのパスワードを暗号化し、TDEフレームワークを使用して暗号化キーを管理します。

LOBロケータ(ラージ・オブジェクト(LOB)値の場所へのポインタ)に署名を割り当てて、LOBを保護できます。

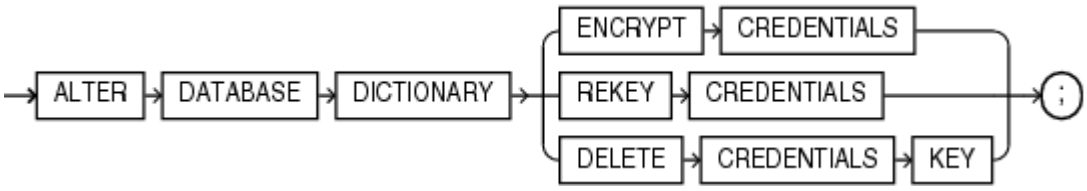
前提条件

- TDEキーストアが存在している必要があります。DDLは、最初にTDEについて次のことをチェックします。
  - キーストアが存在している。
  - キーストアがオープンしている。
  - マスター暗号化キーがTDEキーストア内に存在している。

    いずれかのチェックが失敗した場合、DDLは失敗します。その場合は、TDEキーストアを作成し、TDEマスター・キーをプロビジョニングする必要があります。詳細は、Oracle Databaseセキュリティ・ガイドを参照してください。
- インスタンス初期化パラメータCOMPATIBLEは12.2.0.2に設定する必要があります。
- このコマンドを実行するにはSYSKM権限が必要です。

構文

alter_database_dictionary::=



セマンティクス

alter_database_dictionary_encrypt_credentials::=

このDDLは、データ・ディクショナリ内の既存および将来の不明瞭化された機密情報(たとえば、SYS.LINKS$に格納されているデータベース・リンク・パスワード)を暗号化します。

次のアクションを実行します。

- SYS.LINK$に対応するENC$に新しいエントリを挿入します。
- SGA変数を作成および初期化します。
- SYS.LINK$内の不明瞭化されたパスワードを不明瞭化解除します。
- SYS.LINK$のENC$で生成された暗号化キーを使用して、不明瞭化解除されたパスワードを暗号化します。
- SYS.LINK$内の有効または使用可能なdblinkエントリを示すフラグを設定します。

LOBロケータ署名キーを使用してこのDDLを使用する場合、常に暗号化されます。LOBロケータ(ラージ・オブジェクト(LOB)値の場所へのポインタ)に署名を割り当てて、LOBを保護できます。

alter_database_dictionary_rekey_credentials::=

このDDLはデータ暗号化キーを変更するために使用します。これは、SYS.LINK$およびデータ・ディクショナリ暗号化フレームワークで扱われる他の表に適用されます。

このDDLを使用して、LOBロケータのLOBロケータ署名キーを再生成することもできます。データベースが制限モードの場合、Oracle Databaseは新しいLOB署名キーを再生成し、新しい暗号キーでそれを暗号化します。データベースが非制限モードの場合、新しい署名キーは再生成されず、Oracle Databaseは新しい暗号キーを使用して既存のLOB署名キーを暗号化します。

alter_database_dictionary_delete_credentials_key::=

このDDLは、暗号化されたパスワードにUnusableのマークを付けます。つまり、SYS.LINK$にある現行のパスワード・エントリにUnusableのマークが付けられます。資格証明の暗号化に使用されたENC$内のキーが削除され、それ以降の暗号化を防止するためにSGA変数がクリアされます。

このDDLを使用すると、暗号化されたロケータ署名キーを削除したうえで、不明瞭化した形で新しいLOB署名キーを再生成することもできます。

関連項目:

Oracle Databaseセキュリティ・ガイドのアプリケーション開発者のセキュリティの管理

ALTER DATABASE LINK

目的

接続または認証ユーザーのパスワードが変更された場合に固定ユーザー・データベース・リンクを変更するには、ALTER DATABASE LINK文を使用します。

ノート:

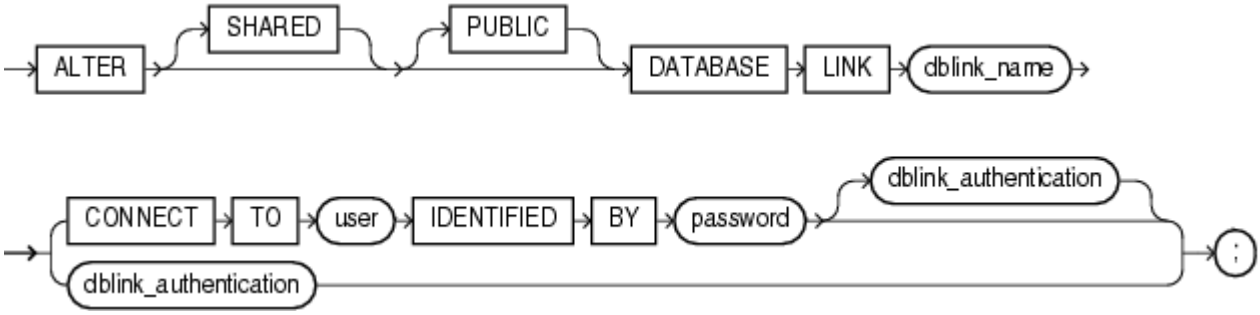
- この文を使用して、データベース・リンクに関連付けられた接続または認証ユーザーを変更することはできません。user を変更するには、データベース・リンクを再作成する必要があります。
- この文を使用して、接続または認証ユーザーのパスワードを変更することはできません。このためには、「ALTER USER」文を使用し、その後、ALTER DATABASE LINK 文でデータベース・リンクを変更する必要があります。
- この文は、固定ユーザー・データベース・リンクに対してのみ有効で、接続ユーザーまたは現行のユーザーのデータベース・リンクに対しては有効でありません。この 2 種類のデータベース・リンクの詳細は、「CREATE DATABASE LINK」を参照してください。

前提条件

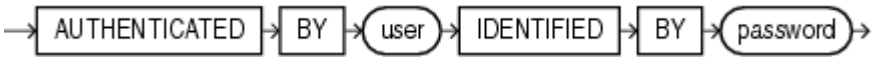
プライベート・データベース・リンクを変更するには、ALTER DATABASE LINKシステム権限が必要です。パブリック・データベース・リンクを変更するには、ALTER PUBLIC DATABASE LINKシステム権限が必要です。

構文

alter_database_link::=



dblink_authentication



セマンティクス

ALTER DATABASE LINK文は、接続および認証ユーザーの現在のパスワードによって固定ユーザー・データベース・リンクを更新するためにのみ使用します。したがって、上の構文図で表示していないCREATE DATABASE LINK文内の有効な句はすべて、ALTER DATABASE LINK文内では有効ではありません。この文で許可されているすべての句のセマンティクスは、CREATE DATABASE LINK内の同じ句のセマンティクスと同じです。詳細は、CREATE DATABASE LINKを参照してくだ

さい。

例

次の文は、ALTER DATABASE LINK文の有効な例を示しています。

```
ALTER DATABASE LINK private_link
  CONNECT TO hr IDENTIFIED BY hr_new_password;
ALTER PUBLIC DATABASE LINK public_link
  CONNECT TO scott IDENTIFIED BY scott_new_password;
ALTER SHARED PUBLIC DATABASE LINK shared_pub_link
  CONNECT TO scott IDENTIFIED BY scott_new_password
  AUTHENTICATED BY hr IDENTIFIED BY hr_new_password;
ALTER SHARED DATABASE LINK shared_pub_link
  CONNECT TO scott IDENTIFIED BY scott_new_password;
```

ALTER DIMENSION

目的

ALTER DIMENSION文を使用すると、ディメンションの階層関係またはディメンション属性を変更できます。

関連項目:

「CREATE DIMENSION」および「DROP DIMENSION」を参照してください。

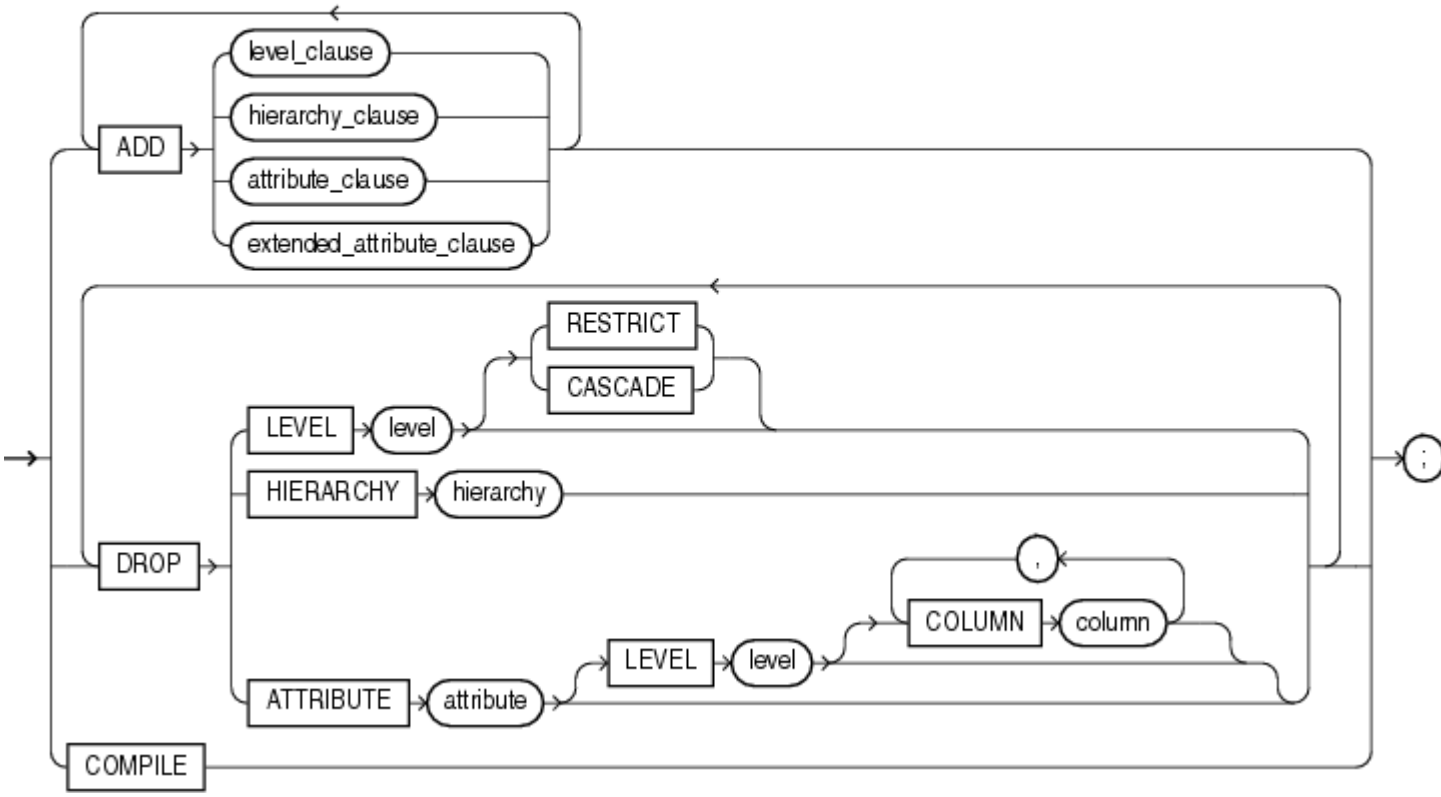
前提条件

ディメンションが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY DIMENSIONシステム権限が必要です。

所有者の権限があれば、ディメンションはいつでも変更されます。

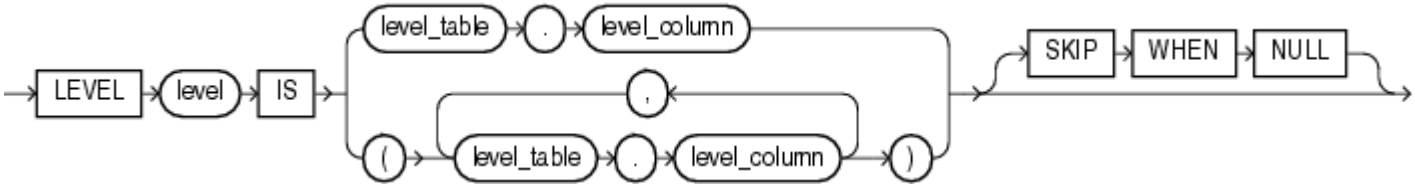
構文

alter_dimension::=

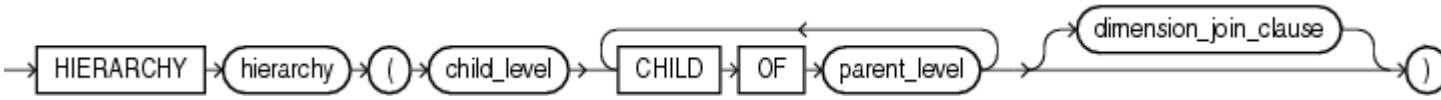


(level_clause::=、hierarchy_clause::=、attribute_clause::=、extended_attribute_clause::=)

level_clause::=

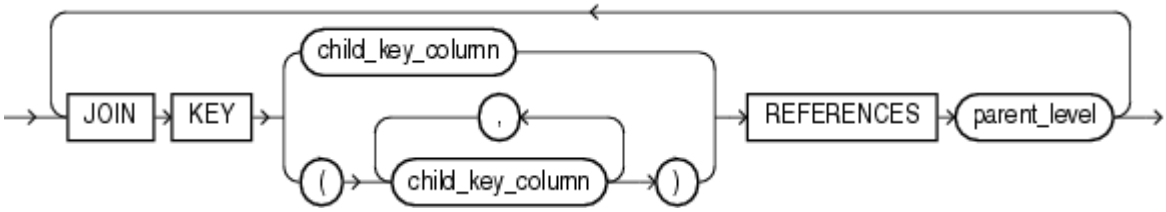


hierarchy_clause::=

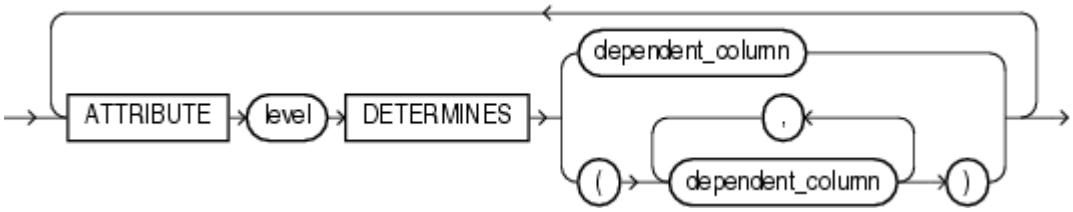


(dimension_join_clause::=)

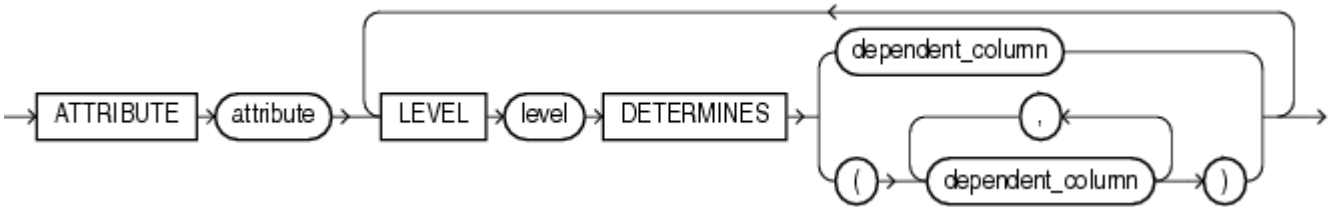
dimension_join_clause::=



attribute_clause::=



extended_attribute_clause::=



セマンティクス

次のキーワード、パラメータおよび句の意味は、ALTER DIMENSIONに対してのみ有効です。その他のキーワード、パラメータおよび句には、CREATE DIMENSION文での機能と同じ機能があります。詳細は、「CREATE DIMENSION」を参照してください。

schema

変更するディメンションのスキーマを指定します。schemaを指定しない場合、ディメンションは自分のスキーマ内にあるとみなされます。

dimension

ディメンション名を指定します。ディメンションは、すでに存在している必要があります。

ADD

ADD句を使用すると、ディメンションにレベル、階層または属性を追加できます。これらの要素の1つを追加しても、既存のマテリアライズド・ビューは無効になりません。

ADD LEVEL句は、他のADD句より前に処理されます。

DROP

DROP句を使用すると、ディメンションからレベル、階層または属性を削除できます。指定するすべてのレベル、階層または属性は、すでに存在している必要があります。

1つの属性で、1つのレベルに関連付けられている1つ以上のレベルと列の関係を削除できます。

DROPの制限事項

属性または階層がレベルを参照する場合は、すべての参照している属性および階層を削除するか、またはCASCADEを指定しないかぎり、このレベルを削除できません。

CASCADE

CASCADEを指定すると、レベルを参照する属性または階層をレベルとともに削除できます。

RESTRICT

RESTRICTを指定すると、属性または階層が参照するレベルを削除しないようにOracle Databaseに指示できます。これはデフォルトです。

COMPILE

COMPILEを指定すると、無効のディメンションを明示的に再コンパイルできます。ADD句またはDROP句が発行されると、ディメンションは自動的にコンパイルされます。ただし、ディメンションが参照するオブジェクトを変更する(たとえば、ディメンションで参照された表を削除した後再作成する)と、ディメンションが無効になるため、これを明示的に再コンパイルする必要があります。

例

ディメンションの変更: 例

次の例では、サンプル・スキーマsh内のcustomers_dimディメンションを変更します。

```
ALTER DIMENSION customers_dim
   DROP ATTRIBUTE country;
ALTER DIMENSION customers_dim
   ADD LEVEL zone IS customers.cust_postal_code
   ADD ATTRIBUTE zone DETERMINES (cust_city);
```

ALTER DISKGROUP

ノート:

この SQL 文は、Oracle ASM を使用しており、Oracle ASM インスタンスを起動している場合にのみ有効です。この文の発行は、通常のデータベース・インスタンスからではなく、Oracle ASM インスタンスから行う必要があります。Oracle ASM インスタンスの起動の詳細は、『Oracle Automatic Storage Management 管理者ガイド』を参照してください。

目的

ALTER DISKGROUP文を使用すると、ディスク・グループまたはディスク・グループ内のディスクに対して多数の操作を実行できます。

関連項目:

- ディスク・グループの作成については、「CREATE DISKGROUP」を参照してください。
- Oracle ASMおよびディスク・グループを使用してデータベース管理を簡略化する方法については、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

前提条件

この文を発行するOracle ASMインスタンスが起動されている必要があります。変更するディスク・グループはマウントされている必要があります。

ALTER DISKGROUPの句はすべて、発行するにはSYSASMシステム権限が必要です。個々の句の発行については、次のようになります。

- SYSOPER権限で実行可能なALTER DISKGROUP操作は、diskgroup_availability、rebalance_diskgroup_clause、check_diskgroup_clause(REPAIRオプションの指定なし)です。
- SYSDBAとして接続している場合は、この文を使用するための限定的な権限が与えられます。次の操作は常に、SYSDBAとして接続しているユーザーに許可されます。
  - ALTER DISKGROUP ... ADD DIRECTORY
  - ALTER DISKGROUP ... ADD/ALTER/DROP TEMPLATE(システム・テンプレート以外の場合のみ)
  - ALTER DISKGROUP ... ADD USERGROUP
  - SELECT
  - SHOW PARAMETER

表10-1に、SYSDBAとして接続しているユーザーに特定の条件下で付与されるその他の権限を示します。

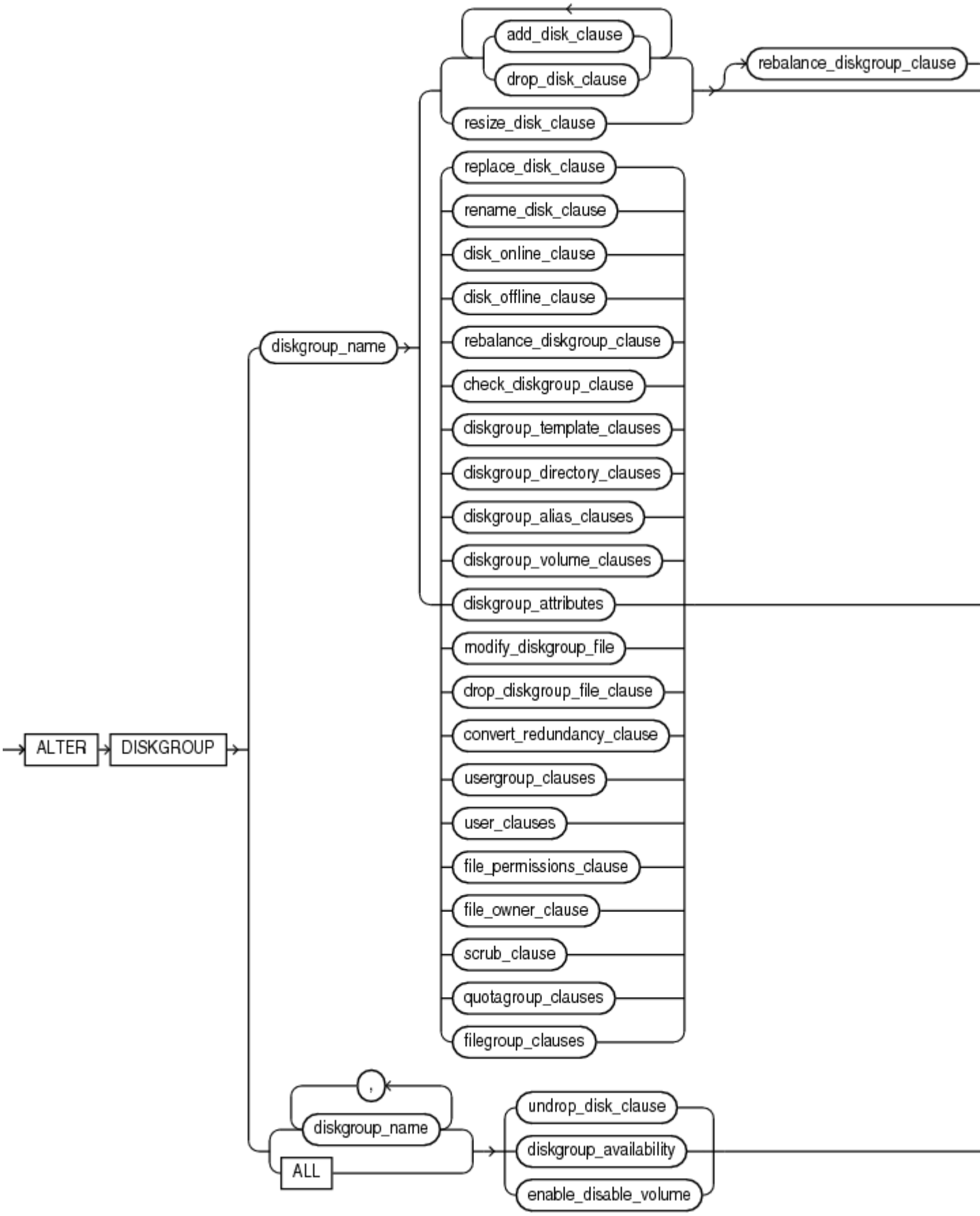
表10-1 SYSDBAへの条件付きディスク・グループ権限

ALTER DISKGROUP操作	条件
DROP FILE	ユーザーはファイルへの読取りおよび書込み権限を持っている必要があります。

ALTER DISKGROUP操作	条件
ADD ALIAS	ユーザーは関連のファイルへの読取りおよび書込み権限を持っている必要があります。
RENAME ALIAS	ユーザーは関連のファイルへの読取りおよび書込み権限を持っている必要があります。
DROP ALIAS	ユーザーは関連のファイルへの読取りおよび書込み権限を持っている必要があります。
RENAME DIRECTORY	ディレクトリにファイルが存在せず、別名のみが存在している必要があります。ユーザーはそのディレクトリ内のすべての別名への DROP ALIAS 権限を持っている必要があります。
DROP DIRECTORY	ディレクトリにファイルが存在せず、別名のみが存在している必要があります。ユーザーはそのディレクトリ内のすべての別名への DROP ALIAS 権限を持っている必要があります。
DROP USERGROUP	ユーザーはユーザー・グループの所有者である必要があります。
MODIFY FILE	ユーザーはファイルの所有者である必要があります。
MODIFY USERGROUP ADD MEMBER	ユーザーはユーザー・グループの所有者である必要があります。
MODIFY USERGROUP DROP MEMBER	ユーザーはユーザー・グループの所有者である必要があります。
SET PERMISSION	ユーザーはファイルの所有者である必要があります。
SET OWNER GROUP	ユーザーはファイルの所有者でユーザー・グループのメンバーである必要があります。

構文

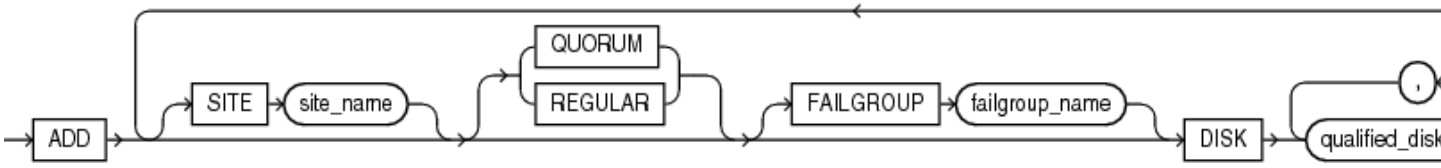
alter_diskgroup::=



(add_disk_clause::=、drop_disk_clause::=、resize_disk_clause::=、replace_disk_clause::=、rename_disk_clause::=、disk_online_clause::=、disk_offline_clause::=、rebalance_diskgroup_clause::=、check_diskgroup_clause::=、diskgroup_template_clauses::=、diskgroup_directory_clauses::=、diskgroup_alias_clauses::=、diskgroup_volume_clauses::=、

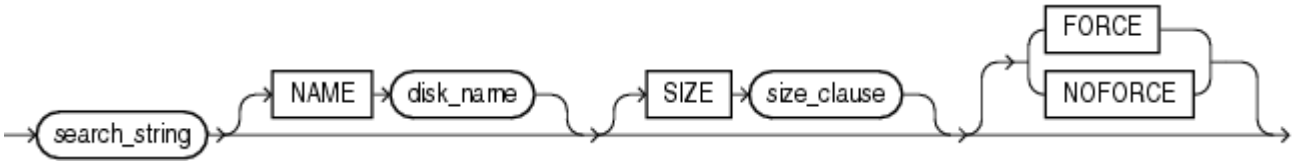
diskgroup_attributes::=、modify_diskgroup_file::=、drop_diskgroup_file_clause::=、
convert_redundancy_clause::=、usergroup_clauses::=、user_clauses::=、
file_permissions_clause::=、file_owner_clause::=、scrub_clause::=、quotagroup_clauses::=、
filegroup_clauses::=、undrop_disk_clause::=、diskgroup_availability::=、
enable_disable_volume::=)

add_disk_clause::=



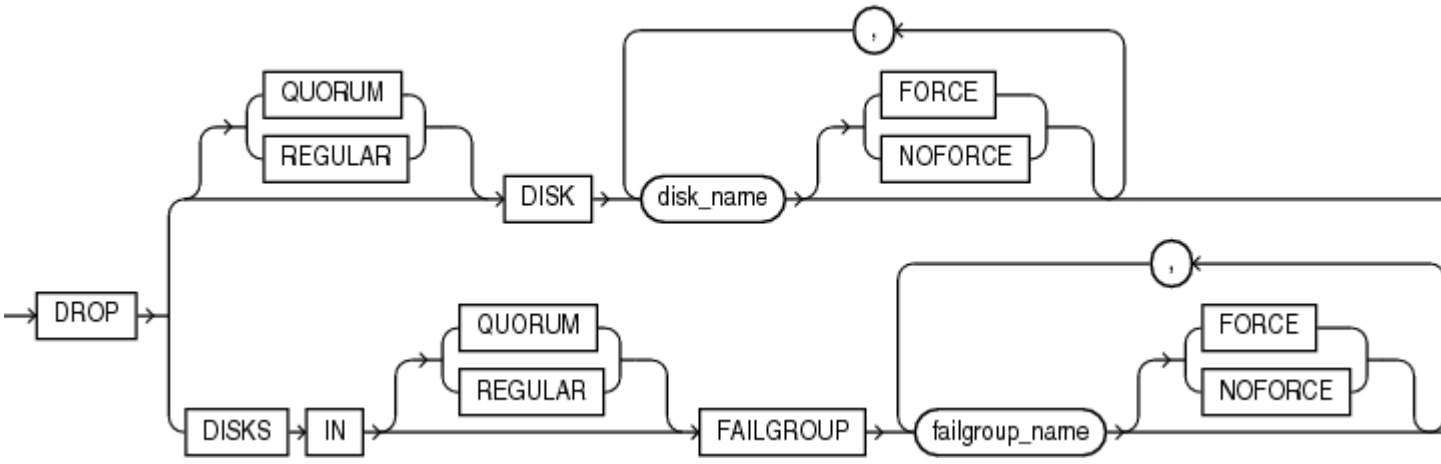
(qualified_disk_clause::=)

qualified_disk_clause::=

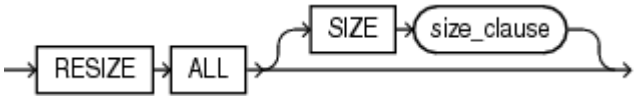


(size_clause::=)

drop_disk_clause::=

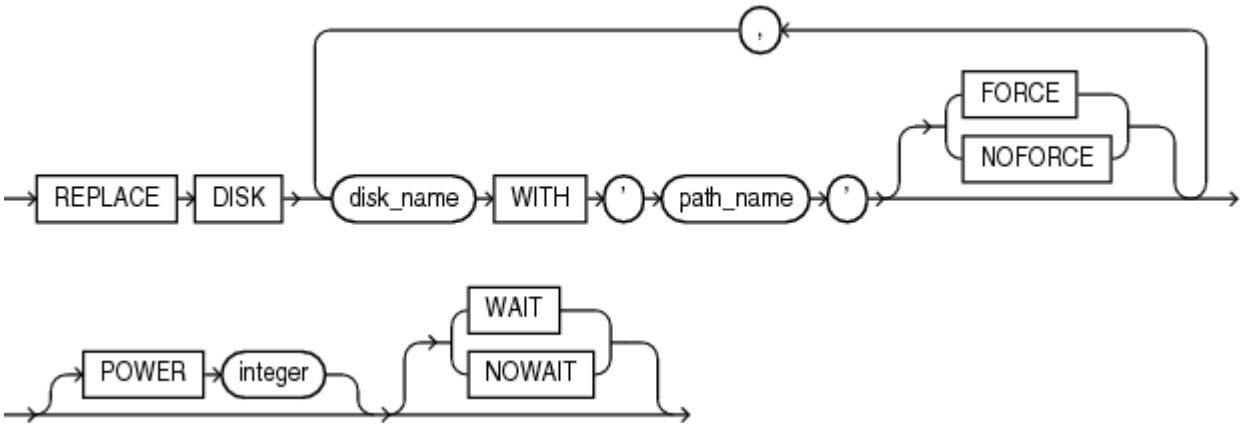


resize_disk_clause::=

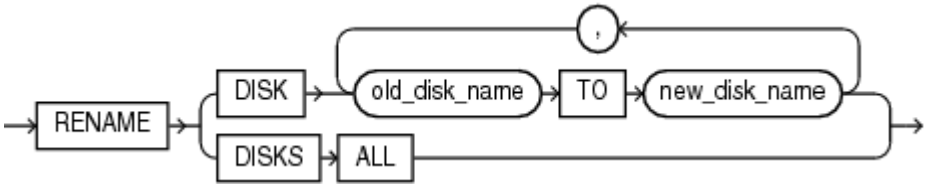


(size_clause::=)

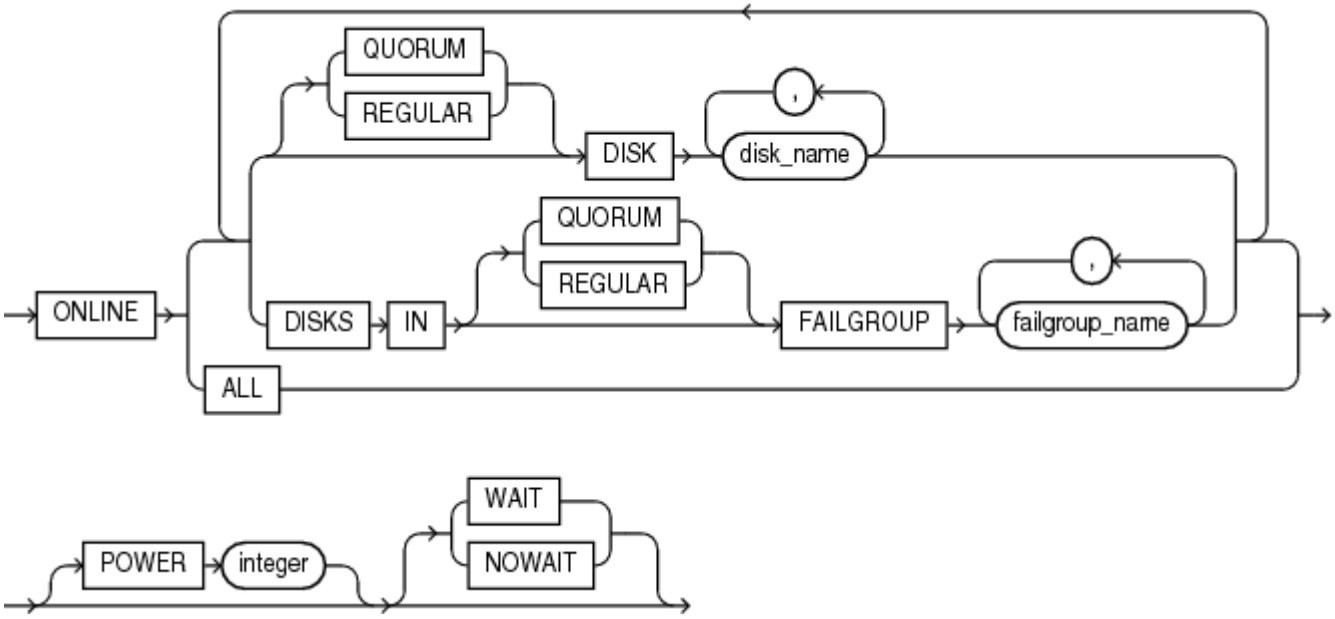
replace_disk_clause::=



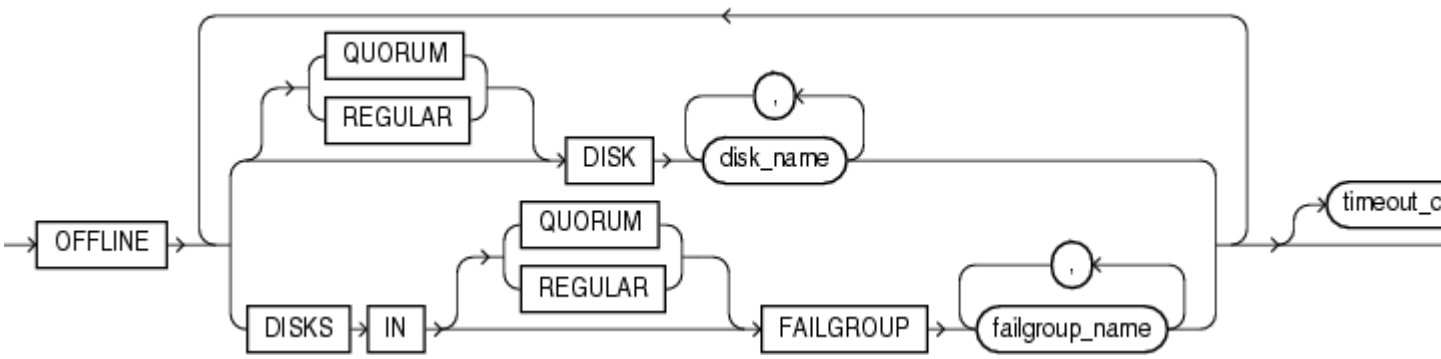
rename_disk_clause::=



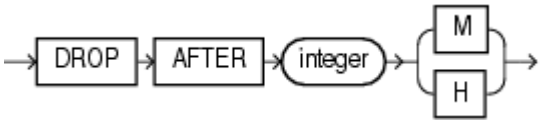
disk_online_clause::=



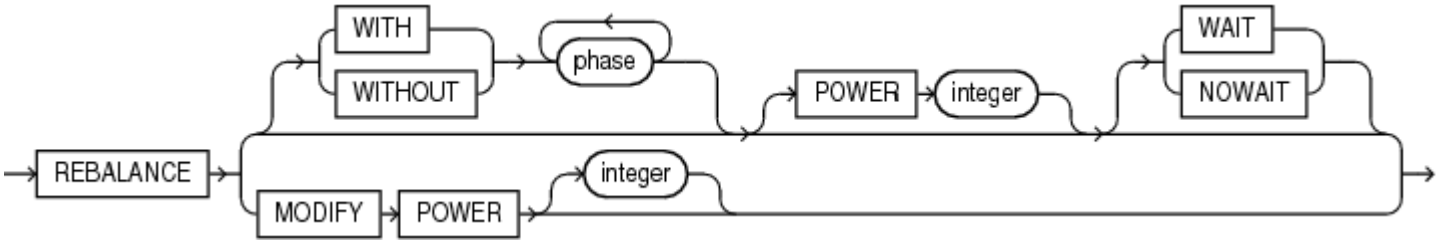
disk_offline_clause::=



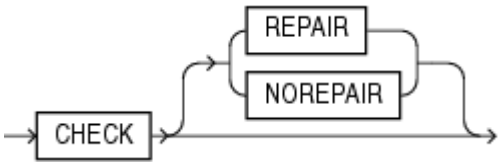
timeout_clause::=



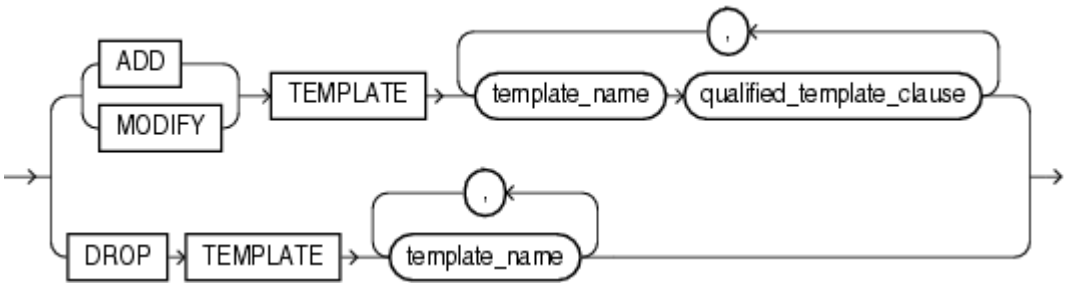
rebalance_diskgroup_clause::=



check_diskgroup_clause::=

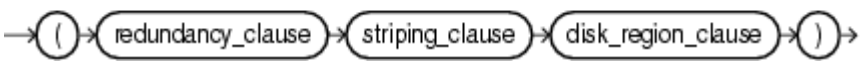


diskgroup_template_clauses::=

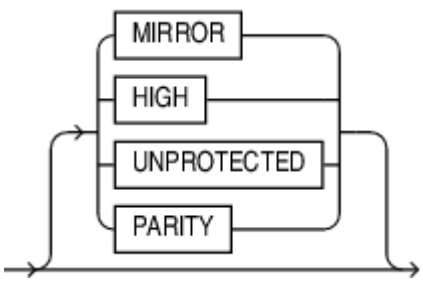


(qualified_template_clause::=)

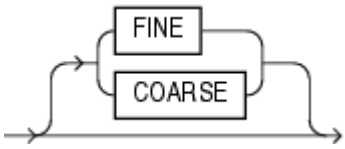
qualified_template_clause::=



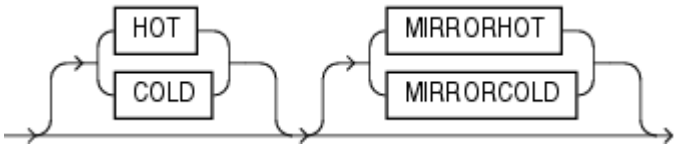
redundancy_clause::=



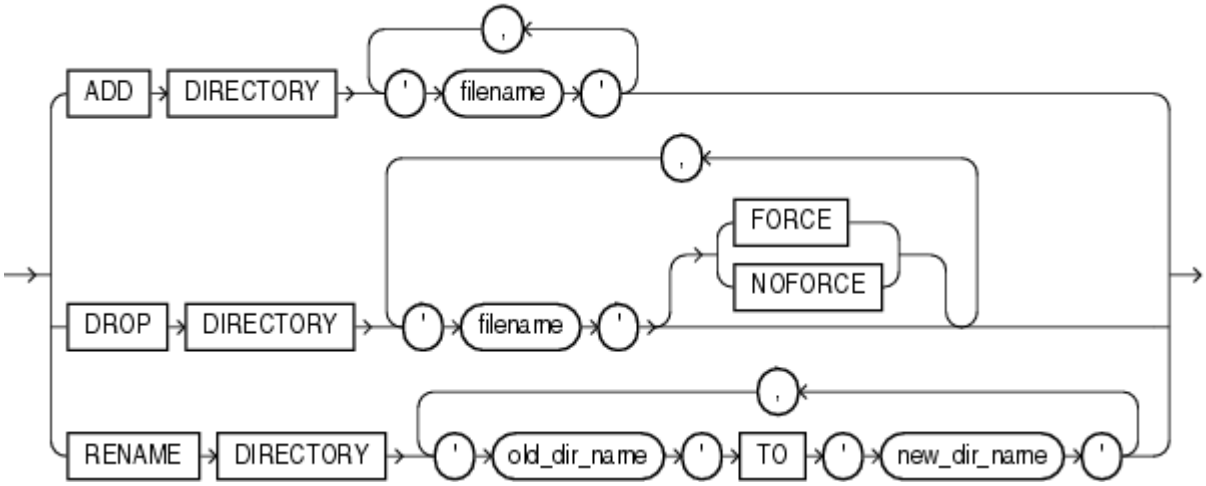
striping_clause::=



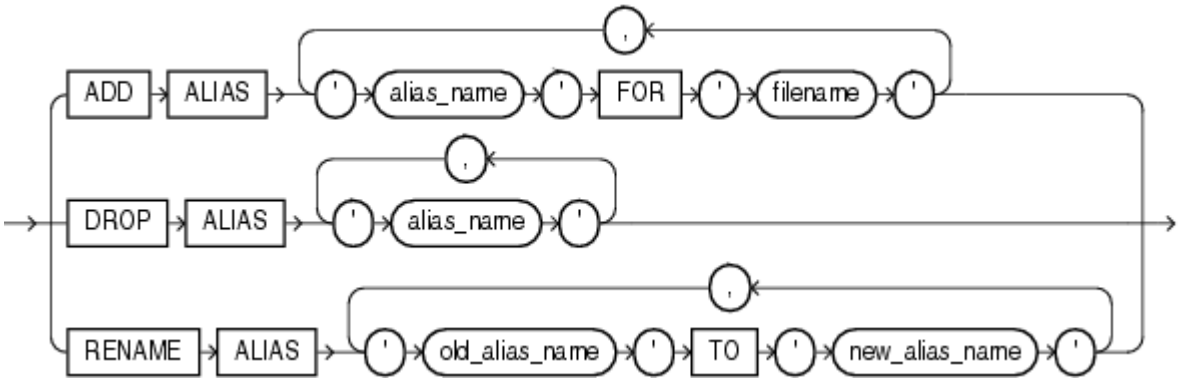
disk_region_clause::=



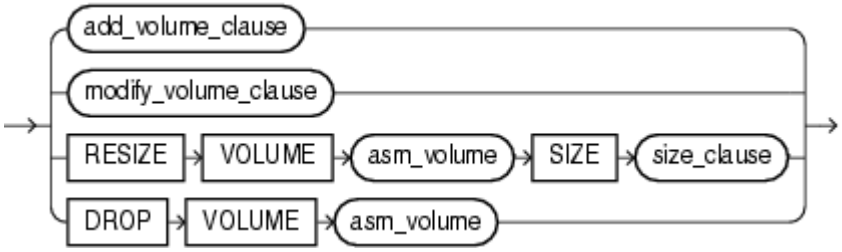
diskgroup_directory_clauses::=



diskgroup_alias_clauses::=

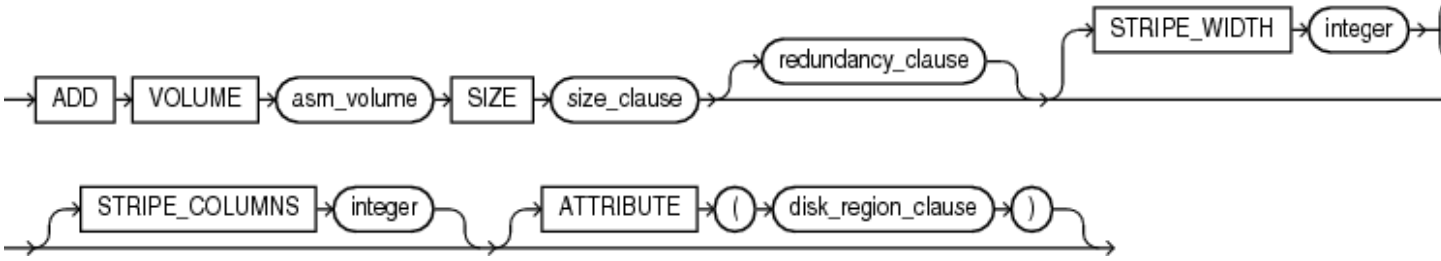


diskgroup_volume_clauses::=



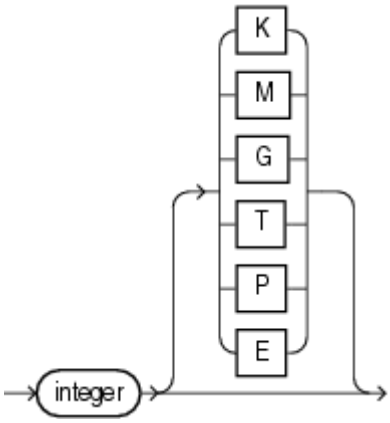
(add_volume_clause::=、modify_volume_clause::=)

add_volume_clause::=

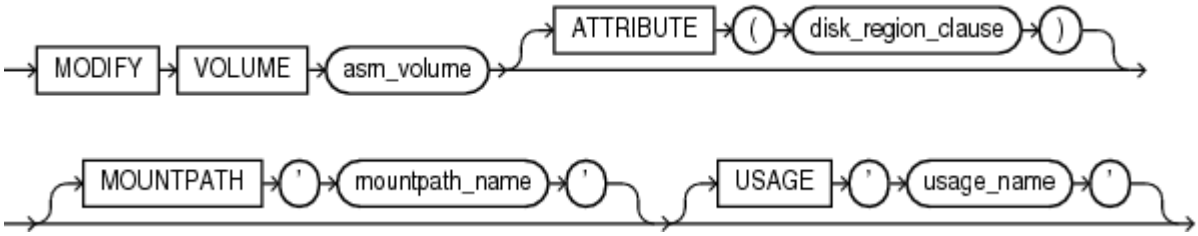


(size_clause::=、redundancy_clause::=、disk_region_clause::=)

size_clause::=

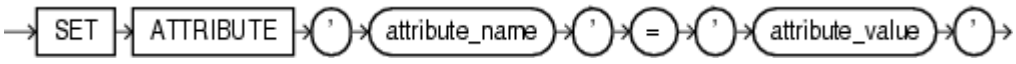


modify_volume_clause::=

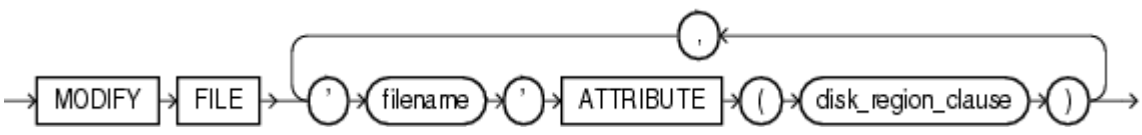


(disk_region_clause::=)

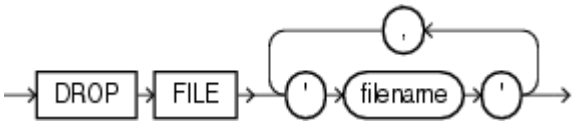
diskgroup_attributes::=



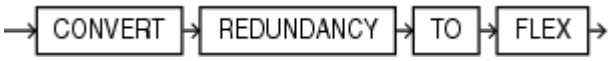
modify_diskgroup_file::=



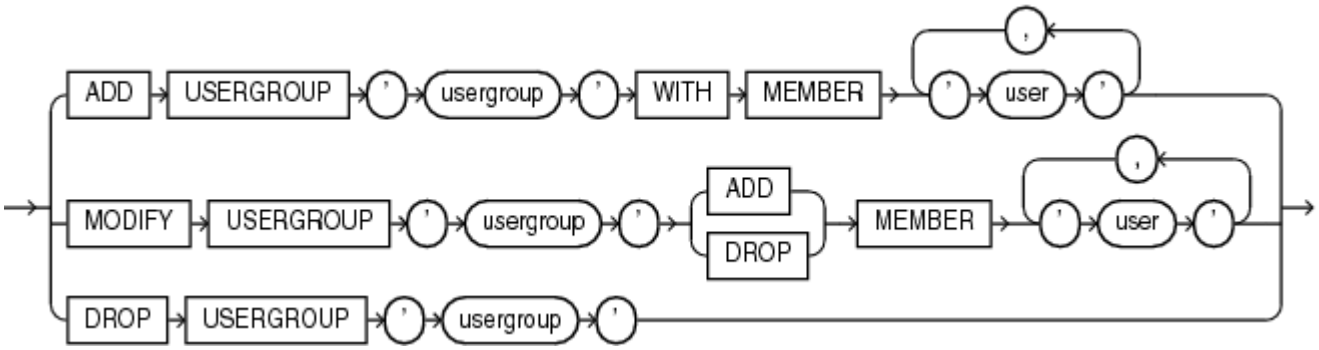
drop_diskgroup_file_clause::=



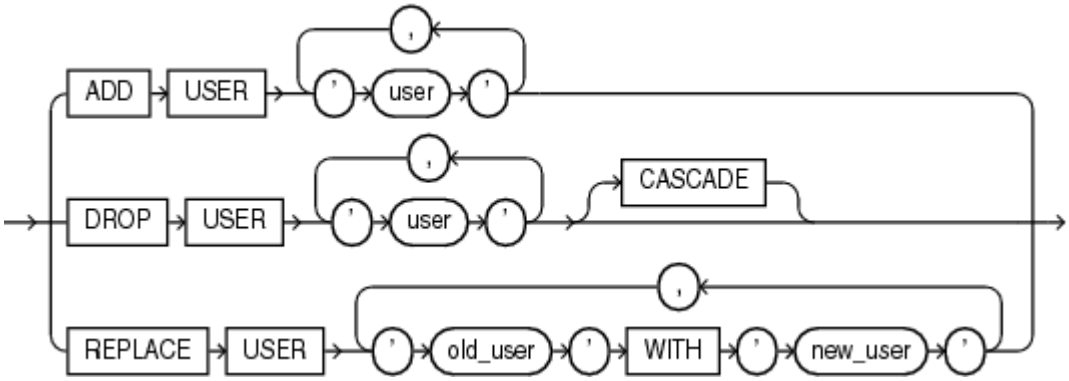
convert_redundancy_clause::=



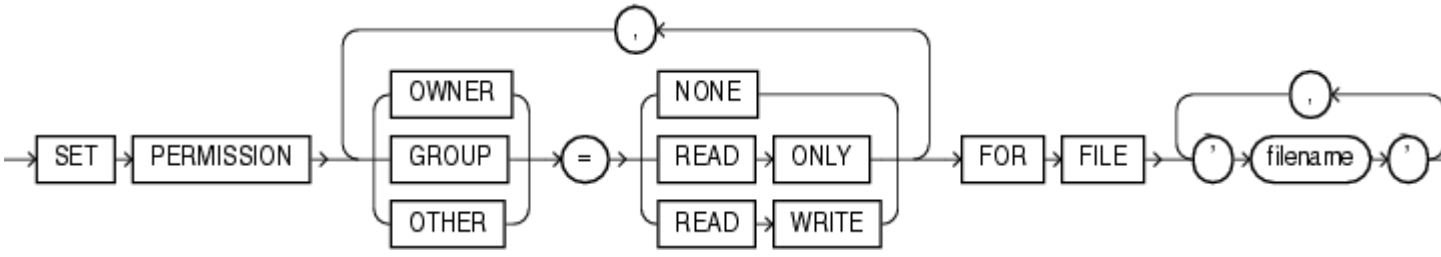
usergroup_clauses::=



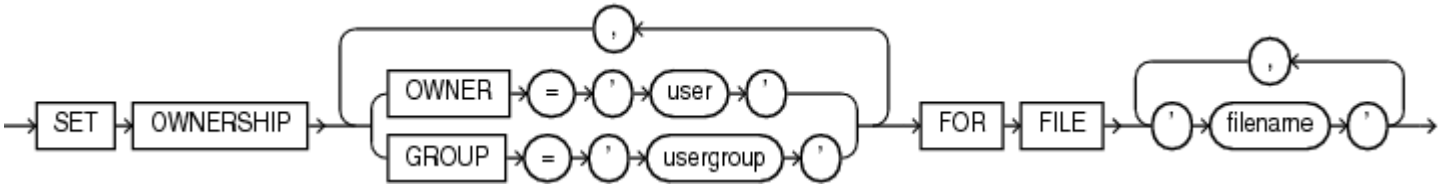
user_clauses::=



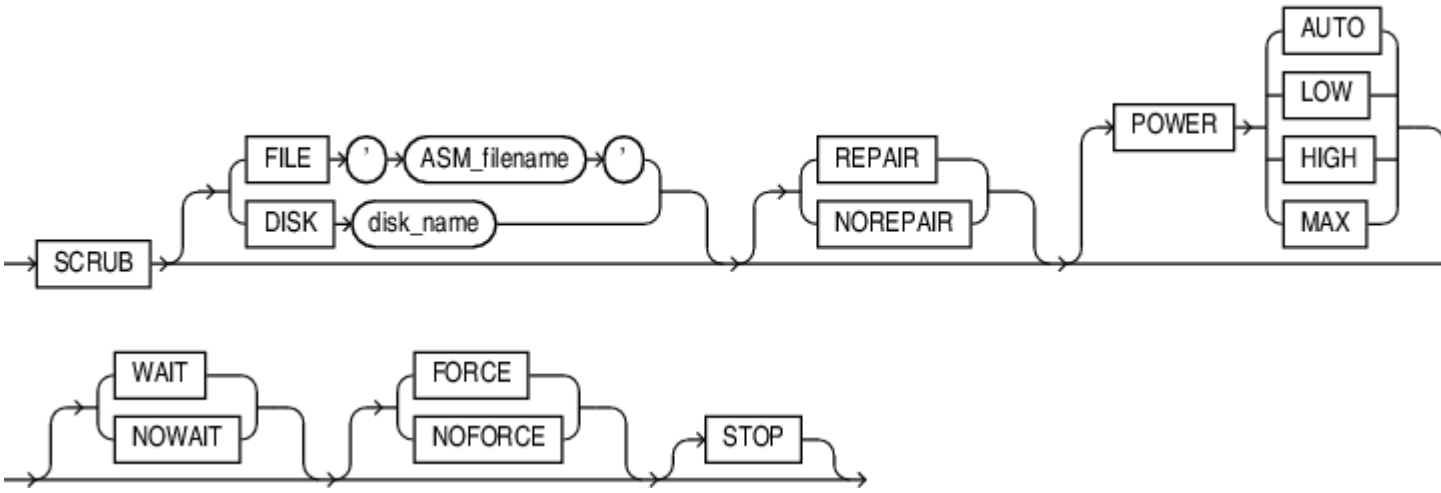
file_permissions_clause::=



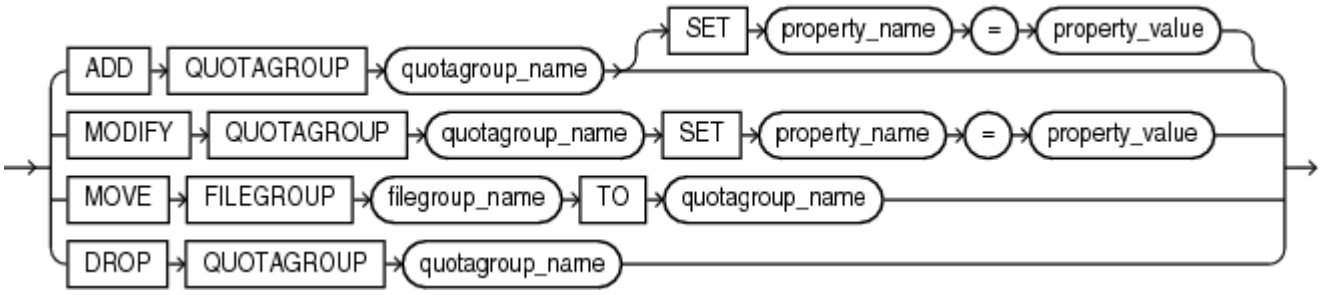
file_owner_clause::=



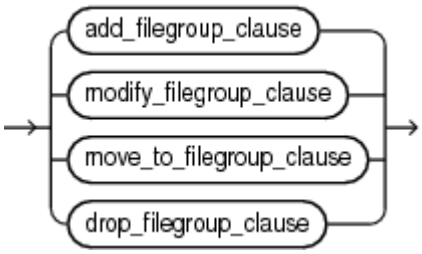
scrub_clause::=



quotagroup_clauses::=

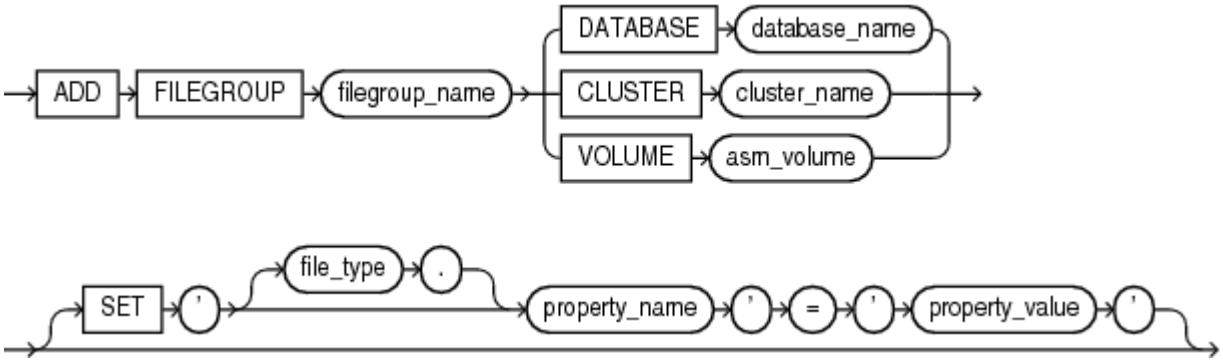


filegroup_clauses::=

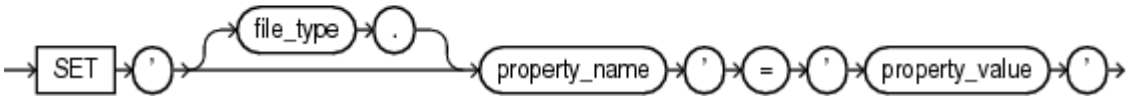


(add_filegroup_clause::=、modify_filegroup_clause::=、move_to_filegroup_clause::=、drop_filegroup_clause::=)

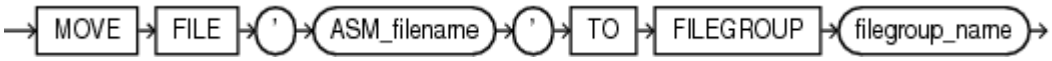
add_filegroup_clause::=



modify_filegroup_clause::=

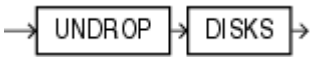


move_to_filegroup_clause::=

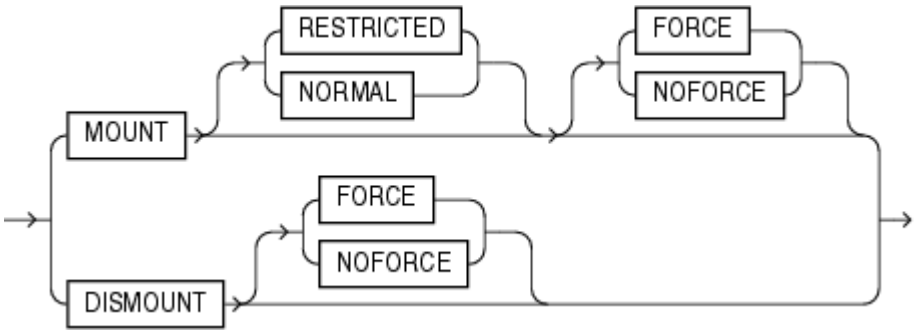


drop_filegroup_clause::=

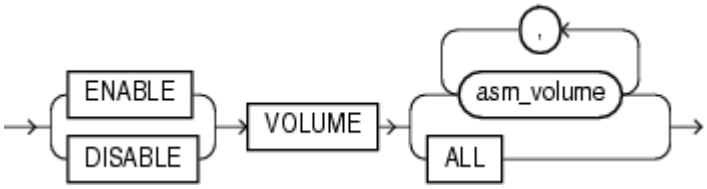
undrop_disk_clause::=



diskgroup_availability::=



enable_disable_volume::=



セマンティクス

diskgroup_name

変更するディスク・グループの名前を指定します。既存のディスク・グループの名前を特定するには、V$ASM_DISKGROUP動的パフォーマンス・ビューを問い合せます。

add_disk_clause

この句を使用すると、ディスク・グループに1つ以上のディスクを追加し、新しく追加したディスクの属性を指定できます。この操作を実行すると、Oracle ASMによって、自動的にディスク・グループの均衡が再調整されます。

この句を使用して、ディスクの障害グループを変更することはできません。これを行うには、ディスク・グループからディスクを削除した後で、新しい障害グループの一部として、ディスク・グループにディスクを再度追加します。

ディスク・グループ内の既存のディスク名を特定するには、V$ASM_DISK動的パフォーマンス・ビューを問い合せます。

QUORUM | REGULAR

これらのキーワードのセマンティクスは、CREATE DISKGROUP文内のセマンティクスと同じです。これらのキーワードの詳細は、「QUORUM | REGULAR」を参照してください。

既存のディスクまたはディスク・グループで、この修飾子を変更することはできません。したがって、ディスク・グループの作成時に指定されているキーワードと異なるキーワードをこの句内で指定することはできません。

関連項目:

これらのキーワードの使用方法の詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

FAILGROUP句

この句を使用すると、新しく追加したディスクを障害グループに割り当てることができます。この句を省略して、標準または高冗長性ディスク・グループにディスクを追加すると、Oracle Databaseは、新しく追加されたディスクを自動的に障害グループに追加します。障害グループの暗黙的な名前は、オペレーティング・システムに依存しないディスク名と同じです(「NAME句」を参照)。

外部冗長性ディスク・グループを作成している場合、この句は指定できません。

qualified_disk_clause

この句のセマンティクスは、CREATE DISKGROUP文およびALTER DISKGROUP文で同じです。この句の詳細は、CREATE DISKGROUPの「qualified_disk_clause」を参照してください。

drop_disk_clause

この句を使用すると、ディスク・グループから1つ以上のディスクを削除できます。

DROP DISK

DROP DISK句を使用すると、ディスク・グループから1つ以上のディスクを削除し、自動的にディスク・グループの均衡を再調整できます。ディスクを削除すると、Oracle ASMによって、ディスクのすべてのデータが再配置され、ディスクをディスク・グループから除外するためにディスク・ヘッダーがクリアされます。FORCEキーワードを指定すると、ディスク・ヘッダーは消去されません。

V$ASM_DISK動的パフォーマンス・ビューのNAME列のように、disk_nameを指定します。

削除するディスクがクォーラム・ディスクであるか、またはクォーラム障害グループに属している場合は、ディスクを削除するためにQUORUMを指定する必要があります。「QUORUM | REGULAR」を参照してください。

DROP DISKS IN FAILGROUP

DROP DISKS IN FAILGROUP句を使用すると、指定した障害グループからすべてのディスクを削除できます。他の動作は、DROP DISK句と同じです。

指定した障害グループがクォーラム障害グループである場合は、ディスクを削除するためにQUORUMキーワードを指定する必要があります。「QUORUM | REGULAR」を参照してください。

FORCE | NOFORCE

これらのキーワードを使用すると、ディスクがディスク・グループから除外されたとみなされるタイミングを指定できます。デフォルト設定であるNOFORCEを使用することをお薦めします。

- NOFORCEを指定すると、Oracle ASMによって、ディスクのすべてのエクステントが他のディスクに再配置された後で、ディスク・グループからそのディスクが除外され、ディスク・グループの均衡が再調整されます。

  ノート:

  DROP DISK ... NOFORCE を指定すると、ディスクの再利用やシステムからの削除を行っても問題ない状態になる前に制御がユーザーに戻ります。ディスクの除外操作が完了したことを確認するには、V$ASM_DISK ビューを問い合せて、HEADER_STATUS の値が FORMER であることを確認します。STATE の値が DROPPING の場合は、ディスクの削除や再利用はしないでください。V$ASM_OPERATION ビューを問い合せると、ディスク除外によって実行される均衡再調整に必要な時間の概算がわかります。REBALANCE ... WAIT (「rebalance_diskgroup_clause」を参照)も指定した場合は、この文から制御が戻るのは均衡再調整操作が完了してディスクがクリアされたときとなります。ただし、均衡の再調整が失敗することもあるため、V$ASM_DISK の HEADER_STATUS 列が FORMER になっていることを必ず確認してください。

- FORCEを指定すると、ディスクは、ディスク・グループからすぐに除外されます。次に、他のディスク上の冗長コピーからデータが再構築され、そのデータが他のディスクに再配置されて、ディスク・グループの均衡が再調整されます。

  FORCE句は、削除するディスクをOracle ASMで読み取れなくなった場合などに役立ちます。ただし、NOFORCEによる削除より長い時間がかかり、ファイルの一部の保護が低下することがあります。外部冗長性ディスク・グループに対してFORCEを指定することはできません(そのディスクの冗長データが存在しない場合は、Oracle ASMがそのディスクからデータを読み取ってからでなければそのディスクを除外できないため)。

FORCEやNOFORCEを指定したかどうかにかかわらず、ディスクを削除する際の均衡の再調整操作には時間がかかります。進捗を監視するには、V$ASM_OPERATION動的パフォーマンス・ビューを問い合せます。均衡の再調整操作の詳細は、「rebalance_diskgroup_clause」を参照してください。

resize_disk_clause

この句を使用して、ディスク・グループ内のそれぞれのディスクに新しいサイズを指定できます。この句は、オペレーティング・システムによって戻されるサイズや、以前にディスクに指定したサイズを上書きします。

SIZE

新しいサイズをKB、MB、GBまたはTB単位で指定します。ディスク容量を超えるサイズは指定できません。ディスク容量よりも小さいサイズを指定した場合、Oracle ASMで使用されるディスク容量が制限されます。この句を指定しない場合、Oracle ASMはプログラム的にディスクのサイズを決定します。

replace_disk_clause

この句を使用すると、ディスク・グループ内の1つ以上のディスクを置換できます。この句により、1回の操作でディスクが置換されます。これは各ディスクを削除して追加するより効率的です。

disk_nameには、置換するディスクの名前を指定します。この名前は置換用のディスクに割り当てられます。ディスク名は、V$ASM_DISK動的パフォーマンス・ビューのNAME列を問合せれば表示されます。

path_nameには、置換用ディスクのフルパス名を指定します。

FORCE

FORCEを指定すると、Oracle ASMで、ディスク・グループのメンバーである置換用ディスクをディスク・グループに追加できます。

ノート:

この方法で FORCE を使用すると、既存のディスク・グループが破棄される可能性があります。

NOFORCE

NOFORCEを指定すると、Oracle ASMで、置換用ディスクがディスク・グループのメンバーである場合にエラーを戻すことができます。デフォルトはNOFORCEです。

POWER

POWER句のセマンティクスは、POWER値を0に設定できないことを除き、ディスク・グループの均衡の手動再調整と同じです。「POWER」を参照してください。

WAIT | NOWAIT

WAITおよびNOWAITキーワードのセマンティクスは、ディスク・グループの均衡の手動再調整と同じです。「WAIT | NOWAIT」を参照してください。

rename_disk_clause

この句を使用すると、ディスク・グループ内の1つ以上のディスクの名前を変更できます。ディスク・グループはMOUNT RESTRICTED状態にあり、このディスク・グループ内のすべてのディスクがオンラインになっている必要があります。

RENAME DISK

この句を指定すると、1つ以上のディスクの名前を変更できます。ディスクごとに、old_disk_nameとnew_disk_nameを指定します。new_disk_nameがすでに存在する場合、この操作は失敗します。

RENAME DISKS ALL

この句を指定すると、ディスク・グループ内のすべてのディスクの名前を、diskgroupname_#### (####はディスク番号)という書式の名前に変更できます。すでにdiskgroupname_####書式になっているディスク名は変更されません。

disk_online_clause

この句を使用すると、1つ以上のディスクをオンラインにして、ディスク・グループの均衡を再調整できます。

ONLINE DISK

ONLINE DISK句を使用すると、指定した1つ以上のディスクをオンラインにして、ディスク・グループの均衡を再調整できます。

V$ASM_DISK動的パフォーマンス・ビューのNAME列のように、disk_nameを指定します。

ここでのQUORUMおよびREGULARキーワードのセマンティクスは、ディスク・グループへのディスクの追加時と同じセマンティクスとなります。「QUORUM | REGULAR」を参照してください。

ONLINE DISKS IN FAILGROUP

ONLINE DISKS IN FAILGROUP句を使用すると、指定した障害グループのすべてのディスクをオンラインにして、ディスク・グループの均衡を再調整できます。

指定した障害グループがクォーラム障害グループである場合は、ディスクをオンラインにするためにQUORUMキーワードを指定する

必要があります。「QUORUM | REGULAR」を参照してください。

ALL

ALL句を使用すると、ディスク・グループのすべてのディスクをオンラインにして、ディスク・グループの均衡を再調整できます。

POWER

POWER句のセマンティクスは、ディスク・グループの均衡の手動再調整と同じです。「POWER」を参照してください。

WAIT | NOWAIT

WAITおよびNOWAITキーワードのセマンティクスは、ディスク・グループの均衡の手動再調整と同じです。「WAIT | NOWAIT」を参照してください。

disk_offline_clause

disk_offline_clauseを使用すると、1つ以上のディスクをオフラインに切り替えることができます。指定したディスクをオフラインに切り替えることによってディスク・グループの冗長性レベルに違反する場合、この句は失敗します。

OFFLINE DISK

OFFLINE DISK句を使用すると、指定した1つ以上のディスクをオフラインにできます。

V$ASM_DISK動的パフォーマンス・ビューのNAME列のように、disk_nameを指定します。

ここでのQUORUMおよびREGULARキーワードのセマンティクスは、ディスク・グループへのディスクの追加時と同じセマンティクスとなります。「QUORUM | REGULAR」を参照してください。

OFFLINE DISKS IN FAILGROUP

OFFLINE DISKS IN FAILGROUP句を使用すると、指定した障害グループのすべてのディスクをオフラインにできます。

指定した障害グループがクォーラム障害グループである場合は、ディスクをオフラインにするためにQUORUMキーワードを指定する必要があります。「QUORUM | REGULAR」を参照してください。

timeout_clause

デフォルトでは、オフラインに切り替えられるとすぐに、ディスクはOracle ASMによって削除されます。timeout_clauseを指定してこの操作を遅らせることにより、ディスクを修復してオンラインに戻すことができます。分単位または時間単位でタイムアウト値を指定できます。単位を指定しない場合、デフォルトは時間です。

この句を複数回指定することによって、タイムアウトの期間を変更できます。指定するたびに、Oracle ASMによって、ディスク・グループがマウントされている間、直近のdisk_offline_clauseからの時間が測定されます。Oracle ASMによってオフライン・ディスクが削除されるまでの残り時間を知るには、V$ASM_DISKのREPAIR_TIMER列を問い合せます。

この句によって、disk_repair_time属性の以前の設定が上書きされます。ディスク・グループの属性の詳細は、表13-2を参照してください。

関連項目:

Oracle ASMディスクのオンラインおよびオフラインへの切替えの詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

rebalance_diskgroup_clause

この句を使用すると、ディスク・グループの均衡を手動で再調整できます。均衡の再調整操作中に、Oracle ASMによって、す

べてのドライブ間で均等にデータファイルが再分散されます。記憶域構成が変化すると、Oracle ASMによって、ファイルが均等に配置され、ディスク・グループの均衡が自動的に再調整されるため、この句が必要なことはほとんどありません。ただし、制御された均衡の再調整操作を実行する必要がある場合は、これが役立ちます。これにより、均衡の再調整操作の特定のフェーズを含めるか除外したり、再調整操作を停止および再開したり、再調整操作の能力を調整することができます。

WITH | WITHOUT

均衡の再調整操作は、RESTORE (RESYNC、RESILVERまたはREBUILDフェーズを含む)、BALANCE、PREPAREおよびCOMPACTの各フェーズで構成されます。

均衡の再調整操作の特定のフェーズを含めたり除外することをOracle ASMに指示するために、WITHまたはWITHOUT句を使用できます。たとえば、時間制約がある場合に、RESTOREフェーズのみを含めることができます。または、フラッシュ・ストレージ・ディスク・グループまたはフラッシュ・キャッシュを持つディスク・グループを使用している場合、そのようなディスク・グループにはメリットがないCOMPACTフェーズを除外できます。

- WITH句を使用すると、均衡の再調整操作の指定したフェーズのみを含めることができます。RESTORE、BALANCE、PREPAREおよびCOMPACTのうち、任意のフェーズを指定できます。RESTOREフェーズは常に発生するため、RESTOREを指定することは許容されますが、必要ではありません。
- WITHOUT句を使用すると、均衡の再調整操作の指定したフェーズを除外できます。BALANCE、PREPAREおよびCOMPACTのうち、任意のフェーズを指定できます。RESTOREフェーズは常に発生する必要があるため、RESTOREは指定できません。

WITHまたはWITHOUT句で複数のフェーズを指定する場合、指定順序は関係ありません。Oracle ASMによって、均衡の再調整操作のフェーズが適切な順序で実行されます。RESYNC、RESILVERまたはREBUILDフェーズはRESTOREフェーズに含まれるため、これらは指定できません。

WITHおよびWITHOUT句を指定しない場合、均衡の再調整操作のすべてのフェーズが実行されます。

均衡の再調整操作の進捗を監視するには、V$ASM_OPERATION動的パフォーマンス・ビューを問い合せます。

均衡の再調整操作のフェーズの詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

POWER

この句を使用すると、均衡の再調整操作の能力、つまり速度を指定できます。また、均衡の再調整操作を停止することもできます。

integerには、0から1024の値を指定します。

- 1から1024までの値で、Oracle ASMが均衡の再調整操作を実行する能力を示します。1が可能な最も低い能力を表し、1024が可能な最も高い能力を表します。
- 0の値を指定すると、アクティブな均衡の再調整操作が停止します。ディスク・グループで手動または自動の均衡の再調整操作が新たに開始されるまで、それ以上の均衡の再調整は発生せず、新たに開始された均衡の再調整操作は最初から実行されます。均衡の再調整操作を停止した時点から後で再開できるようにするには、かわりにMODIFY POWER 0を指定することによって、均衡の再調整操作を停止します。詳細は、「MODIFY POWER」句を参照してください。

POWER句を指定しない場合、デフォルトの能力は次のように決定されます。

- フレックス・ディスク・グループの場合は、そのPOWER_LIMITプロパティの値に従って、各ファイル・グループの均衡が再調整されます。ファイル・グループにPOWER_LIMITプロパティが設定されていない場合は、ファイル・グループの

ASM_POWER_LIMIT初期化パラメータの値が使用されます。

- その他すべてのタイプのディスク・グループでは、POWER句を指定しない場合、ASM_POWER_LIMIT初期化パラメータの値に従ってディスク・グループの均衡が再調整されます。

WAIT | NOWAIT

この句を使用すると、制御を、均衡の再調整操作のどの段階でユーザーに戻すか指定できます。

- 均衡の再調整操作が終了した後、ユーザーに制御を戻す場合は、WAITを指定します。WAITモードで実行されている均衡の再調整操作を明示的に終了させることができますが、操作を終了させても、同じ文の中ですでに完了したディスクの追加、削除またはサイズ変更の操作が元に戻ることはありません。
- NOWAITは、文の発行後すぐに制御をユーザーに戻す場合に指定します。これはデフォルトです。

MODIFY POWER

この句を使用すると、アクティブな均衡の再調整操作を停止または再開したり、操作の能力を変更することができます。

integerは次のように指定できます。

- 0を指定すると、均衡の再調整操作を停止できます。この方法で均衡の再調整操作を停止した場合は、ALTER DISKGROUP ... MODIFY POWER ...文を発行することで、停止したフェーズから操作を再開できます。POWER句を使用してディスク・グループで手動で均衡の再調整操作を開始した場合、またはディスク・グループで自動で均衡の再調整操作が発生した場合は、均衡の再調整操作は最初から開始されます。
- 1から1024を指定すると、均衡の再調整操作の能力を指定できます。1が可能な最も低い能力を表し、1024が可能な最も高い能力を表します。均衡の再調整操作が実行中の場合は、操作を中断することなく能力が変更されます。均衡の再調整操作がMODIFY POWER 0句を使用して停止された場合は、指定された能力で均衡の再調整操作が再開されます。
- integerを省略すると、デフォルトの能力を指定できます。均衡の再調整操作が実行中の場合は、操作を中断することなく能力がデフォルトの能力に変更されます。均衡の再調整操作がMODIFY POWER 0句を使用して停止された場合は、デフォルトの能力で均衡の再調整操作が再開されます。デフォルトの能力を決定する方法の詳細は、「POWER」句を参照してください。

関連項目:

- ASM_POWER_LIMIT初期化パラメータおよびV$ASM_OPERATION動的パフォーマンス・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。
- ディスク・グループの均衡の再調整: 例

check_diskgroup_clause

check_diskgroup_clauseを使用すると、Oracle ASMディスク・グループのメタデータの内部一貫性を検証できます。ディスク・グループは、マウントされている必要があります。Oracle ASMによってサマリー・エラーが表示され、検出されたエラーの詳細がアラート・ログに書き込まれます。

CHECKキーワードによって、次の処理が実行されます。

- ディスクの一貫性をチェックします。
- すべてのファイル・エクステント・マップおよび割当て表の一貫性をクロスチェックします。

- 別名メタデータ・ディレクトリおよびファイル・ディレクトリが正しくリンクされていることをチェックします。
- 別名ディレクトリ・ツリーが正しくリンクされていることをチェックします。
- Oracle ASMメタデータ・ディレクトリに、到達不可能なブロックが割り当てられていないことをチェックします。

REPAIR | NOREPAIR

この句を使用すると、一貫性チェックで検出されたエラーを修正するかどうかをOracle ASMに指示できます。デフォルトは`NOREPAIR`です。`NOREPAIR`設定は、非一貫性が検出された場合には警告を受け取るが、Oracle ASMによる修正処理を自動的に行わない場合に便利です。

非推奨となった句

以前のリリースでは、`ALL`、`DISK`、`DISKS IN FAILGROUP`または`FILE`に対して`CHECK`を指定できました。これらの句は不要になったため、非推奨になる予定です。指定すると、動作は以前のリリースと同じで、アラート・ログにメッセージが追加されます。ただし、これらの句はサポートされなくなる予定であるため、新しいコードには使用しないことをお薦めします。非推奨になる予定の句は、次のとおりです。

- `ALL`: ディスク・グループ内のすべてのディスクとファイルをチェックします。
- `DISK`: ディスク・グループ内の1つ以上の指定したディスクをチェックします。
- `DISKS IN FAILGROUP`: 指定した障害グループ内のすべてのディスクをチェックします。
- `FILE`: ディスク・グループ内の1つ以上の指定したファイルをチェックします。ファイル名の参照書式のいずれかを使用する必要があります。Oracle ASMのファイル名の参照書式の詳細は、「ASM_filename」を参照してください。

diskgroup_template_clauses

テンプレートは、属性の名前付きコレクションです。ディスク・グループを作成すると、Oracle ASMによって、システムの一連の初期デフォルト・テンプレートがそのディスク・グループに関連付けられます。テンプレートで定義される属性は、そのディスク・グループ内のすべてのファイルに適用されます。表10-2に、システムのデフォルト・テンプレートと、様々な種類のファイルに適用される属性を示します。表の後に説明する`diskgroup_template_clauses`を使用すると、テンプレートの属性を変更して、新しいテンプレートを作成できます。

ディスク・グループ・ファイルを作成した後は、この句を使用して属性を変更することはできません。これを行うには、Recovery Manager(RMAN)を使用して、新しい属性を持つ新しいファイルにそのファイルをコピーする必要があります。

表10-2 Oracle Automatic Storage Managementシステムのファイル・グループのデフォルト・テンプレート

テンプレート名	ファイル・タイプ	外部冗長性のディスク・グループでのミラー化レベル	標準冗長性のディスク・グループでのミラー化レベル	高冗長性のディスク・グループでのミラー化レベル	ストライプ化	リージョン
CONTROLFILE	制御ファイル	非保護	3 方向ミラー	3 方向ミラー	FINE	COLD

テンプレート名	ファイル・タイプ	外部冗長性のディスク・グループでのミラー化レベル	標準冗長性のディスク・グループでのミラー化レベル	高冗長性のディスク・グループでのミラー化レベル	ストライプ化	リージョン
DATAFILE	データファイルとコピー	非保護	双方向ミラー	3 方向ミラー	COARSE	COLD
ONLINELOG	オンライン・ログ	非保護	双方向ミラー	3 方向ミラー	COARSE	COLD
ARCHIVELOG	アーカイブ・ログ	非保護	双方向ミラー	3 方向ミラー	COARSE	COLD
TEMPFILE	一時ファイル	非保護	双方向ミラー	3 方向ミラー	COARSE	COLD
BACKUPSET	データファイルのバックアップ・ピース、データファイルの増分バックアップ・ピース、およびアーカイブ・ログのバックアップ・ピース	非保護	双方向ミラー	3 方向ミラー	COARSE	COLD
PARAMETERFILE	SPFILE	非保護	双方向ミラー	3 方向ミラー	COARSE	COLD
DATAGUARDCONFIG	障害回復構成(スタンバイ・データベースで使用)	非保護	双方向ミラー	3 方向ミラー	COARSE	COLD
FLASHBACK	フラッシュバック・ログ	非保護	双方向ミラー	3 方向ミラー	COARSE	COLD
CHANGETRACKING	ブロック・チェンジ・トラッキング・データ(増分バックアップで使用)	非保護	双方向ミラー	3 方向ミラー	COARSE	COLD

テンプレート名	ファイル・タイプ	外部冗長性のディスク・グループでのミラー化レベル	標準冗長性のディスク・グループでのミラー化レベル	高冗長性のディスク・グループでのミラー化レベル	ストライプ化	リージョン
DUMPSET	データ・ポンプ・ダンプセット	非保護	双方向ミラー	3 方向ミラー	COARSE	COLD
XTRANSPORT	クロス・プラットフォーム変換データファイル	非保護	双方向ミラー	3 方向ミラー	COARSE	COLD
AUTOBACKUP	自動バックアップ・ファイル	非保護	双方向ミラー	3 方向ミラー	COARSE	COLD
ASMPARAMETERFILE	SPFILE	非保護	双方向ミラー	3 方向ミラー	COARSE	COLD
OCRFILE	Oracle Cluster Registry ファイル	非保護	双方向ミラー	3 方向ミラー	COARSE	COLD

ADD TEMPLATE

この句を使用すると、ディスク・グループに1つ以上の名前付きテンプレートを追加できます。既存のテンプレートの名前を特定するには、V$ASM_TEMPLATE動的パフォーマンス・ビューを問い合せます。

MODIFY TEMPLATE

この句を使用すると、システムのデフォルト・ディスク・グループ・テンプレートまたはユーザー定義のディスク・グループ・テンプレートの属性を変更できます。指定した属性のみが変更されます。指定していないプロパティは、現在の値のまま変更されません。

ノート:

以前のリリースでは、キーワード ALTER TEMPLATE が MODIFY TEMPLATE のかわりに使用されました。ALTER キーワードは、下位互換性のためにまだサポートされていますが、他の Oracle SQL との一貫性のために MODIFY に置き換えられました。

template_name

追加または変更するテンプレートの名前を指定します。テンプレート名の長さは、最大30文字です。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

redundancy_clause

アーカイブ・ログのようなライトワンス・ファイル形式で単一のパリティを保護するために、PARITYを指定します。パリティ保護が指定されない場合、ライトワンス・ファイル・タイプのデフォルトの冗長性は引き続きシステム・テンプレートから導出されます。

ライトワンス・ファイル・タイプの冗長性を変更して、必要に応じて後からパリティ保護をサポートできます。

新しく追加したテンプレートまたは変更したテンプレートの冗長性レベルを指定します。

- MIRROR: このテンプレートが適用されるファイルは、データ・ブロックのミラー化によって保護されます。標準冗長性ディスク・グループでは、プライマリ・エクステントごとに1つのミラー・エクステントが存在します(双方向ミラー化)。高冗長性ディスク・グループでは、プライマリ・エクステントごとに2つのミラー・エクステントが存在します(3方向ミラー化)。外部冗長性ディスク・グループのテンプレートには、MIRRORを指定することはできません。
- HIGH: このテンプレートが適用されるファイルは、データ・ブロックのミラー化によって保護されます。標準冗長性および高冗長性ディスク・グループでは、プライマリ・エクステントごとに2つのミラー・エクステントが存在します(3方向ミラー化)。外部冗長性ディスク・グループのテンプレートには、HIGHを指定することはできません。
- UNPROTECTED: このテンプレートが適用されるファイルは、自動ストレージ管理によってメディア障害から保護されません。システムのアクションまたはユーザー・コマンドによってディスクがオフラインになると、保護されていないファイルが失われる可能性があります。外部冗長性ディスク・グループには、UNPROTECTEDのみが有効な設定です。高冗長性ディスク・グループのテンプレートには、UNPROTECTEDを指定することはできません。標準または高冗長性ディスク・グループでは、保護されていないファイルを使用しないことをお薦めします。
- PARITY: ライトワンス・ファイル形式の単一パリティに、プロパティPARITYを指定します。

REDUNDANCY句を省略した場合、デフォルト値は、標準冗長性ディスク・グループではMIRROR、高冗長性ディスク・グループではHIGH、外部冗長性ディスク・グループではUNPROTECTEDになります。

striping_clause

このテンプレートが適用されるファイルがどのようにストライプ化されるかを指定します。

- FINE: このテンプレートが適用されるファイルは、128KB単位でストライプ化されます。このストライプ化モードはOracle ASM spfileでは無効です。
- COARSE: このテンプレートが適用されるファイルは、1MB単位でストライプ化されます。これがデフォルト値です。

disk_region_clause

この句を使用すると、ディスク・グループ・ファイルのインテリジェント・データ配置属性を指定できます。Oracle ASMによってファイルに割り当てるディスク内の領域を次のように指定します。

- HOT: エクステントはスピンドルから最も離れたディスク領域に割り当てられます。ディスク上のこれらの外側のトラックは内側のトラックより長いため、より多くのセクターを含み、スループットもより向上します。
- COLD: エクステントはスピンドルに最も近いディスク領域に割り当てられます。
- MIRRORHOTおよびMIRRORCOLD: ファイルのミラー化されたデータ・ブロックに対する領域を指定します。

目的のディスク領域に使用できる領域がない場合は、Oracle ASMによって他の領域のエクステントが割り当てられますが、領域のサイズを調整するために均衡の再調整が開始されます。

関連項目:

インテリジェント・ディスク配置の詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

DROP TEMPLATE

この句を使用すると、ディスク・グループから1つ以上のテンプレートを削除できます。この句で削除できるのはユーザー定義テンプレートのみであり、システム・デフォルト・テンプレートは削除できません。

diskgroup_directory_clauses

Oracle ASMファイル名の別名(「diskgroup_alias_clauses」を参照)を作成する前に、別名が存在する場所の完全なディレクトリ構造を指定する必要があります。`diskgroup_directory_clauses`を使用すると、そのようなディレクトリ構造を作成および操作できます。

ADD DIRECTORY

この句を使用すると、階層的に名付けられた別名の新しいディレクトリ・パスを作成できます。ディレクトリの各コンポーネントを区切るには、スラッシュ(/)を使用します。各ディレクトリ・コンポーネントの最大長は48バイトであり、スラッシュ文字を含むことはできません。コンポーネントの最初または最後の文字に空白を使用することはできません。ディレクトリ・パス全体の長さは、256バイトから、このディレクトリに作成する別名(「diskgroup_alias_clauses」を参照)の長さを引いた長さを超えることはできません。

DROP DIRECTORY

この句を使用すると、階層的に名付けられた別名のディレクトリを削除できます。`FORCE`が指定されていないかぎり、Oracle ASMでは、別名の定義を含むディレクトリは削除されません。この句は、システム別名の一部として作成されたディレクトリの削除には無効です。そのようなディレクトリは、V$ASM_ALIAS動的パフォーマンス・ビューの`SYSTEM_CREATED`列の値がYとなっています。

RENAME DIRECTORY

この句を使用すると、階層的に名付けられた別名のディレクトリ名を変更できます。この句は、システム別名の一部として作成されたディレクトリ名の変更には無効です。そのようなディレクトリは、V$ASM_ALIAS動的パフォーマンス・ビューの`SYSTEM_CREATED`列の値がYとなっています。

diskgroup_alias_clauses

Oracle ASMファイルが作成されると(暗黙的かユーザー指定によるかにかかわらず)、そのファイルには、ドット付きの数値の組で終わる完全修飾名が割り当てられます(「file_specification」を参照)。`diskgroup_alias_clauses`を使用すると、Oracle ASMファイル名のわかりやすい別名を作成できます。ドット付きの数値の組で終わる別名は指定できません(このような書式を使用すると、Oracle ASMファイル名と区別できないため)。

この句を指定する前に、まずネーミング規則に従ってディレクトリ構造を作成する必要があります(「diskgroup_directory_clauses」を参照)。別名全体の長さは最大256バイトです(ディレクトリの接頭辞を含む)。別名では大/小文字は区別されませんが、大/小文字の区別は保持されます。

ADD ALIAS

この句を使用すると、Oracle ASMファイル名の別名を作成できます。`alias_name`には、ディレクトリのフルパスと別名を指定します。既存のOracle ASM別名の名前を特定するには、V$ASM_ALIAS動的パフォーマンス・ビューを問い合せます。Oracle ASMファイル名の詳細は、「ASM_filename」を参照してください。

DROP ALIAS

この句を使用すると、ディスク・グループ・ディレクトリから別名を削除できます。各別名には、ディレクトリのフルパスと別名を指定します。別名が参照する元のファイルは変更されません。

RENAME ALIAS

この句を使用すると、既存の別名を変更できます。alias_nameには、ディレクトリのフルパスと別名を指定します。

別名の削除および名前変更の制限事項

システム生成の別名を削除したり、名前を変更することはできません。別名がシステム生成されたものかどうかを確認するには、V$ASM_ALIAS動的パフォーマンス・ビューのSYSTEM_CREATED列を問い合せます。

diskgroup_volume_clauses

これらの句を使用して、物理ボリューム・デバイスに対応する論理Oracle ASM動的ボリューム・マネージャ(Oracle ADVM)のボリュームを操作できます。これらの句を使用するには、Oracle ASMが開始されていて、変更するディスク・グループがマウントされている必要があります。

関連項目:

例を含むディスク・グループ・ボリュームの詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

add_volume_clause

この句を使用すると、ディスク・グループにボリュームを追加できます。

asm_volumeでは、ボリュームの名前を指定します。名前に使用できるのは英数字のみで、先頭の文字は英字にする必要があります。名前の最大長はプラットフォームによって異なります。詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

size_clauseには、Oracle ADVMボリュームのサイズを指定します。Oracle ASMインスタンスによって、ボリュームを作成するために十分な領域が存在するかどうかが確認されます。十分な領域が存在しない場合は、Oracle ASMによりエラーが戻されます。十分な領域が存在する場合は、Oracle ASMインスタンスが実行されディスク・グループがマウントされているすべてのクラスタ内ノードに対し、追加が通知されます。Oracle ASMは、ファイル・システムの作成およびマウントに使用可能なボリューム・デバイスをそれらのノード上に作成して使用できるようにします。

次のオプションの設定も使用できます。

- redundancy_clauseで、Oracle ADVMボリュームの冗長性レベルを指定します。標準冗長ディスク・グループでボリュームを作成する場合のみ、この句を指定できます。次のボリューム冗長レベルを指定できます。
  - MIRROR: ボリュームの双方向ミラー化。これはデフォルトです。
  - HIGH: ボリュームの3方向ミラー化。
  - UNPROTECTED: ボリュームのミラー化なし。

  高冗長ディスク・グループまたは外部冗長ディスク・グループでボリュームを作成する場合は、redundancy_clauseを指定できません。その場合には、エラーが発生します。高冗長ディスク・グループでは、Oracle Databaseによりボリューム冗長が自動的にHIGH (3方向ミラー化)に設定されます。外部冗長ディスク・グループでは、Oracle Databaseによりボリューム冗長が自動的にUNPROTECTED (ミラー化なし)に設定されます。
- STRIPE_WIDTH句では、Oracle ADVMボリュームのストライプ幅を指定します。有効な値は、4KBから1MBまでの範囲の2の累乗です。デフォルト値は128Kです。
- STRIPE_COLUMNS句では、Oracle ADVMボリュームのストライプ・セット1つ当たりのストライプの数を指定します。有効範囲は1から8です。デフォルトは4です。STRIPE_COLUMNSを1に設定すると、ストライプ化は無効になります。この場合、ストライプ幅はボリュームのエクステント・サイズになります。このボリュームのエクステント・サイズは、ディスク・グ

ループの割当て単位(AU)の64倍です。

- disk_region_clause句では、ディスク・グループ・ボリュームのプライマリとプライマリ以外の両方のミラーのインテリジェント・データ配置属性を指定します。ともにデフォルトはCOLDです。この句の詳細は、「disk_region_clause」を参照してください。

modify_volume_clause

:この句を使用すると、既存のOracle ADVMボリュームの特性を変更できます。次の句を1つ以上指定する必要があります。

- disk_region_clause句では、ディスク・グループ・ボリュームのプライマリとプライマリ以外の両方のミラーのインテリジェント・データ配置属性を指定します。プライマリ・ミラーのデフォルトはCOLDです。ミラーおよび高冗長性のデフォルトはHOTです。この句の詳細は、「disk_region_clause」を参照してください。
- MOUNTPATH句では、ボリュームに関連付けられたマウントパスの名前を指定します。mountpath_nameは、最大1024文字まで指定可能です。
- USAGE句では、ボリュームに関連付けられた使用状況の名前を指定します。usage_nameは、最大30文字まで指定可能です。

RESIZE VOLUME句

この句を使用すると、既存のOracle ADVMボリュームのサイズを変更できます。Oracle ASMクラスタ内のすべてのノードに新しいサイズが伝播されます。ボリューム上にOracle Automatic Storage Managementファイル・システム(ACFS)が存在する場合は、ALTER DISKGROUP文のかわりにacfsutil sizeコマンドを使用する必要があります。

関連項目:

acfsutil sizeコマンドの使用方法の詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

DROP VOLUME句

この句を使用すると、既存のOracle ADVMボリュームの記憶域コンテナであるOracle ASMファイルを削除できます。Oracle ASMクラスタ内では、Oracle ASMインスタンスが実行されディスク・グループがオープンしているすべてのノードに対する削除操作が通知され、これにより、ボリューム・デバイスが削除されます。ボリューム・ファイルがオープンしている場合は、この句によりエラーが戻されます。

diskgroup_attributes

この句を使用すると、ディスク・グループの属性を指定できます。表13-2に、この句で設定できる属性を示します。この句の動作の詳細は、「CREATE DISKGROUP」の「ATTRIBUTE句」を参照してください。

modify_diskgroup_file

この句を使用すると、既存のディスク・グループ・ファイルのインテリジェント・データ配置属性を変更できます。ファイルのインテリジェント・データ配置を変更する場合、この処理はそのファイルの新しいエクステントには適用されますが、均衡の再調整操作を実行するまでは既存のファイルの内容には影響しません。均衡の再調整を手動で開始することで、新しいインテリジェント・データ配置ポリシーを既存のファイルの内容に適用できます。均衡の再調整操作では、最後に指定したポリシーがファイル・エクステントに対して使用されます。

関連項目:

- この句の詳細は、「disk_region_clause」を参照してください。
- ディスク・グループの均衡の手動による再調整の詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

drop_diskgroup_file_clause

この句を使用すると、ディスク・グループからファイルを削除できます。Oracle ASMによって、削除するファイルに関連付けられたすべての別名も削除されます。ファイル名の参照書式のいずれかを使用する必要があります。ほとんどのOracle ASMファイルはOracle Managed Filesであり、不要になると自動的に削除されるため、手動で削除する必要はありません。Oracle ASMのファイル名の参照書式の詳細は、「ASM_filename」を参照してください。

ディスク・グループ・ファイルが現在のインスタンスまたはOracle ASMクラスタ内のインスタンスの起動に使用されたspfileである場合は、そのディスク・グループ・ファイルを削除できません。

convert_redundancy_clause

この句を使用して、NORMAL REDUNDANCYまたはHIGH REDUNDANCYディスク・グループをFLEX REDUNDANCYディスク・グループに変換できます。変換を開始する前にディスク・グループには少なくとも3人の障害グループが必要です。

usergroup_clauses

これらの句を使用して、ディスク・グループにユーザー・グループを追加したり、ディスク・グループからユーザー・グループを削除したり、または既存のユーザー・グループにメンバーを追加したり、そこからメンバーを削除することができます。

関連項目:

例を含むユーザー・グループとメンバーの詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

ADD USERGROUP

この句を使用すると、ディスク・グループにユーザー・グループを追加できます。ユーザー・グループを作成するには、SYSASMまたはSYSDBA権限が必要です。ユーザー・グループ名の最大長は63バイトです。ユーザー名を指定する場合は、それがOSパスワード・ファイル内に含まれている必要があり、32文字を超えることはできません。

MODIFY USERGROUP

これらの句を使用すると、既存のユーザー・グループにメンバーを追加したり、そこからメンバーを削除できます。これらの句を使用するには、ユーザー・グループのOracle ASM管理者(SYSASM権限を所持)またはOracle ASM作成者(SYSDBA権限を所持)である必要があります。ユーザー名は、OSパスワード・ファイル内の既存のユーザーである必要があります。

DROP USERGROUP

この句を使用すると、ディスク・グループから既存のユーザー・グループを削除できます。この句を使用するには、ユーザー・グループのOracle ASM管理者(SYSASM権限を所持)またはOracle ASM作成者(SYSDBA権限を所持)である必要があります。ユーザー・グループを削除すると、有効なユーザー・グループを含まないディスク・グループ・ファイルが残る場合があります。この場合は、「file_permissions_clause」を使用して手動でディスク・グループ・ファイルを更新し、新しい有効なグループを追加できます。

user_clauses

これらの句を使用すると、ディスク・グループにユーザーを追加したり、このグループからユーザーを削除したり、このグループ内のユーザーを置換したりできます。

ノート:

SQL*Plus でユーザーを管理する場合、ユーザーは既存のオペレーティング・システム・ユーザーで、そのユーザー名には対応するオペレーティング・システム・ユーザーID が含まれている必要があります。ただし、Oracle ASM インスタンスと同じクラスタ内のユーザーのみを検証できます。

ADD USER

この句を使用すると、Oracle ASMディスク・グループに1人以上のオペレーティング・システム(OS)ユーザーを追加して、それらのユーザーにディスク・グループに対するアクセス権限を付与できます。ユーザー名はOSパスワード・ファイル内の既存のユーザーである必要があり、32文字を超えることはできません。指定したユーザーがすでにディスク・グループ内に存在する場合(V$ASM_USERで表示)は、コマンドによってエラーが記録され、他にもユーザーを指定している場合は、続けて他のユーザーの追加が行われます。データベース・インスタンスを実行しているOSユーザーは、インスタンスがディスク・グループにアクセスする際にディスク・グループに自動的に追加されるため、このコマンドを使用する必要はほとんどありません。ただし、この句は特定のデータベース・インスタンスに関連付けられていないユーザーを追加する場合に有効です。

DROP USER

この句を使用すると、ディスク・グループから1人以上のユーザーを削除できます。この句で指定したユーザーがディスク・グループに存在しない場合は、エラーが記録され、他にもユーザーを指定した場合は続けて他のユーザーの削除が行われます。そのユーザーが所有するファイルがある場合は、CASCADEキーワードを指定する必要があります(指定すると、ユーザーとそのユーザーのすべてのファイルが削除されます)。そのユーザーが所有しているファイルがオープンされている場合は、DROP USER CASCADEは失敗し、エラーが戻されます。

ユーザーが所有するファイルを削除せずにユーザーを削除するには、各ファイルの所有者を他のユーザーに変更し、そのユーザーに対してALTER DISKGROUP ... DROP USER文を発行します。また、ALTER DISKGROUP ... REPLACE USER文を発行して、ディスク・グループに存在していないユーザーで、削除するユーザーを置換することもできます。この操作には、それまで削除したユーザーが所有していたファイルを、新しいユーザーが所有するようになるという副作用があります。

REPLACE USER

この句を使用すると、ディスク・グループ内のold_userをnew_userに置換できます。現時点でold_userが所有しているすべてのファイルは、new_userが所有するようになり、old_userはディスク・グループから削除されます。old_userはディスク・グループに存在している必要があり、new_userはディスク・グループに存在していてはいけません。

file_permissions_clause

この句を使用して、ディスク・グループ・ファイルの権限の設定を変更できます。権限の3つのクラスは、所有者、ユーザー・グループおよびその他です。この句を使用するには、ファイルの所有者またはOracle ASM管理者である必要があります。

オープン・ファイルの権限設定を変更すると、その時点でファイルに対して実行している操作は、変更前の権限設定を使用して完了することになります。新しい権限設定は、再認証の要求時に反映されます。

file_owner_clause

この句を使用すると、指定したファイルに対して所有者またはユーザー・グループを設定できます。ファイルの所有者を変更するには、Oracle ASM管理者である必要があります。ファイルのユーザー・グループを変更するには、ファイルの所有者またはOracle ASM管理者である必要があります。さらに、ファイルに関連するユーザー・グループを変更する場合は、指定したユーザー・グループがディスク・グループ内にすでに存在し、ファイルの所有者がそのユーザー・グループのメンバーである必要があります。

オープン・ファイルに対してこの句を使用すると、次の条件が適用されます。

- オープン・ファイルの所有者またはユーザー・グループを変更すると、その時点でファイルに対して実行している操作は、変更前の所有者またはユーザー・グループを使用して完了することになります。新しい所有者またはユーザー・グループは、再認証の要求時に反映されます。
- オープン・ファイルの所有者を変更すると、そのファイルの新しい所有者は、インスタンスを再起動するまでディスク・グループから削除できなくなります。Oracle ASMクラスタでは、そのファイルの新しい所有者は、クラスタ内のすべてのインスタンスを再起動するまで削除できなくなります。
- オープン・ファイルの所有者を変更すると、ファイルの所有権が変更された後でも、そのファイルがオープンしている間は元の所有者を削除できません。

scrub_clause

この句を使用すると、ディスク・グループをスクラブできます。スクラブ操作は、論理データの破損を検査して、標準冗長性および高冗長性のディスク・グループ内で破損を自動的に修復します。

- FILE句を使用すると、ディスク・グループ内の指定したOracle ASMファイルをスクラブできます。ASM_filenameの参照書式のいずれか1つを使用する必要があります。Oracle ASMのファイル名の参照書式の詳細は、「ASM_filename」を参照してください。
- DISK句を使用すると、ディスク・グループ内の指定したディスクをスクラブできます。
- FILEまたはDISKを省略すると、ディスク・グループ内のすべてのファイルとディスクがスクラブされます。

REPAIR | NOREPAIR

REPAIRを指定すると、論理データの破損検査中に検出したエラーの修復を試行するようになります。NOREPAIRを指定すると、破損についての警告が通知されますが、Oracle ASMは修正処理を実行しません。デフォルトはNOREPAIRです。

POWER

POWER句を使用すると、スクラブ操作の機能レベルを指定できます。有効な値は、AUTO、LOW、HIGH、およびMAXです。この句を省略すると、機能レベルのデフォルトはAUTOになり、POWERはシステムの最適レベルに調整されます。

WAIT | NOWAIT

WAITを指定すると、スクラブ操作を完了してからユーザーに制御を戻します。NOWAITを指定すると、操作をスクラブ・キューに追加し、制御をただちにユーザーに戻します。デフォルトは、NOWAITです。

FORCE | NOFORCE

FORCEを指定すると、システムのI/O負荷が高い場合や、スクラブ処理がシステム・レベルで無効にされている場合でも、このコマンドが処理されます。NOFORCEを指定すると、コマンドは通常通りに処理されます。デフォルトは、NOFORCEです。

STOP

進行中のスクラブ操作を停止する場合は、STOPを指定します。

スクラブ操作の進行状況は、V$ASM_ OPERATION動的パフォーマンス・ビューを問い合せることで監視できます。

関連項目:

ディスク・グループのスクラブの詳細は、『Oracle Automatic Storage Management管理者ガイド』、および「ディスク・グループのスクラブ: 例」を参照してください。

quotagroup_clauses

これらの句を使用すると、ディスク・グループへの割当て制限グループの追加、割当て制限グループの変更、割当て制限グループへのファイル・グループの移動、割当て制限グループの削除を行うことができます。

割当て制限グループはファイル・グループのコレクションです。ファイル・グループは、1つのディスク・グループ内のデータベースのすべてのファイルのコンテナです。割当て制限グループには指定された割当て制限があり、これはファイル・グループが共同で使用できる記憶域の最大容量です。このため、割当て制限グループによって、ディスク・グループ内のデータベース・グループの割当て制限を指定できます。ディスク・グループ内のすべての割当て制限グループの割当て制限の合計は、ディスク・グループの記憶域容量を上回ることができます。

各ディスク・グループには、GENERICというデフォルトの割当て制限グループが含まれます。ファイル・グループを作成して、その割当て制限グループを指定しない場合、ファイル・グループはGENERIC割当て制限グループに属します。compatible.asm属性を12.2以上に設定してディスク・グループを作成すると、または既存のディスク・グループのcompatible.asmを12.2以上に設定すると、Oracle ASMでGENERIC割当て制限グループが自動的に作成されます。初期的には、GENERICの割当て制限はUNLIMITEDです。その後、MODIFY QUOTAGROUP句を使用して、この割当て制限を変更できます。

ADD QUOTAGROUP

この句を使用して割当て制限グループを作成し、ディスク・グループに追加します。quotagroup_nameに、新規割当て制限グループの名前を指定します。

SET句を使用すると、割当て制限グループの割当て制限を設定できます。

- property_nameにQUOTAを指定します。
- property_valueには、次のいずれかの句を指定します。
  - size_clauseを指定すると、割当て制限のバイト数を設定できます。指定可能な最小値は1バイトです。ディスク・グループの記憶域サイズよりも大きい値を指定できます。この場合、記憶域の使用量はディスク・グループの現在のサイズに制限されます。ただし、その後、ディスク・グループの記憶域容量を、割当て制限を超えるサイズに増やした場合、割当て制限が実施されます。この句の構文およびセマンティクスは、「size_clause」を参照してください。0バイトを指定することは、UNLIMITEDを指定することに相当することに注意してください。
  - 割当て制限を設定しない場合は、UNLIMITEDを指定します。この場合、記憶域の使用量はディスク・グループの記憶域サイズに制限されます。

SET句を指定しない場合、デフォルトはSET QUOTA=UNLIMITEDになります。

MODIFY QUOTAGROUP

この句を使用して、割当て制限グループの割当て制限を変更します。quotagroup_nameに、変更する割当て制限グループの名前を指定します。GENERIC割当て制限グループを含め、任意の割当て制限グループの割当て制限を変更できます。SET句のセマンティクスは、ADD QUOTAGROUP句と同じです。割当て制限には、割当て制限グループによって現在使用されている領域の量よりも小さい値を設定できます。このアクションによって、この割当て制限グループに関連付けられているファイル・グループによって記述されたファイルに、追加領域が割り当てられなくなります。

MOVE FILEGROUP

この句を使用して、割当て制限グループ間でファイル・グループを移動させることができます。filegroup_nameに、移動させるファイル・グループの名前を指定します。quotagroup_nameに、宛先割当て制限グループの名前を指定します。移動操作によって、宛先割当て制限グループの使用済記憶領域の容量が割当て制限を超える場合、操作は成功しますが、割当て

制限グループ内のファイル・グループで新しい記憶域割当てを実行できない可能性があります。この機能を使用すると、特定のファイル・グループによって記述されたすべてのファイルによる、追加領域の割当てを停止できます。

DROP QUOTAGROUP

この句を使用して、ディスク・グループから割当て制限グループを削除します。quotagroup_nameに、削除する割当て制限グループの名前を指定します。割当て制限グループにファイル・グループを含めることはできません。割当て制限グループGENERICは削除できません。

関連項目:

割当て制限グループの詳細は、『Automatic Storage Management管理者ガイド』を参照してください。

filegroup_clauses

filegroup_clausesは、フレックス・ディスク・グループでのみ有効です。これらの句を使用すると、ファイル・グループの作成、ファイル・グループの変更、ファイル・グループへのファイルの移動、ファイル・グループの削除を行うことができます。ファイル・グループは、1つのディスク・グループ内のデータベースのすべてのファイルのコンテナです。ファイル・グループは、割当て制限グループに属する必要があります。

各ディスク・グループには、FILEGROUP_NUMBER = 0のデフォルトのファイル・グループがあります。

add_filegroup_clause

この句を使用すると、ファイル・グループを作成できます。

filegroup_nameには、新規ファイル・グループの名前を指定します。ファイル・グループ名の最大長は127文字です。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。また、引用符で囲んだ場合も、大/小文字は区別されません。それらは常に内部的には大文字で格納されます。ファイル・グループ名は、ディスク・グループ内において一意である必要があります。

- DATABASE句を使用して、ファイル・グループが関連付けられるデータベース(非CDB、CDBまたはPDB)を指定します。
- CLUSTER句を使用して、ファイル・グループが関連付けられるクラスタを指定します。
- VOLUME句を使用して、ファイル・グループが関連付けられるボリュームを指定します。

同じディスク・グループ内の複数のファイル・グループに、同じデータベース、クラスタまたはボリュームを関連付けることはできません。ファイル・グループの作成時にデータベース、クラスタまたはボリュームが存在しない場合、その後それらが作成されるときに、自動的にファイル・グループに関連付けられます。データベース、クラスタおよびボリュームの名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

SET句を使用すると、ファイル・グループのプロパティを設定できます。プロパティにSET句を指定しない場合、デフォルト値が割り当てられます。ファイル・タイプを適用する任意のプロパティにfile_typeを指定できます。こうしたプロパティにfile_typeを指定しない場合、プロパティはすべてのファイル・タイプに適用されます。ファイル・グループ・プロパティおよびそのデフォルト値の詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

modify_filegroup_clause

この句を使用して、ファイル・グループ・プロパティを変更できます。filegroup_nameに、変更するファイル・グループの名前を指定します。デフォルトのファイル・グループを含め、任意のファイル・グループのプロパティを変更できます。この句で指定しないものは変更されません。SET句のセマンティクスは、add_filegroup_clauseと同じです。

move_to_filegroup_clause

この句を使用して、ファイル・グループにファイルを移動できます。ファイルが現在異なるファイル・グループに関連付けられている場合は、ファイルはそのファイル・グループからの関連付けを解除されます。ターゲット・ファイル・グループでは、ファイルを含めるのに十分な領域が使用可能である必要があります。ユーザーは、ファイルおよびターゲット・ファイル・グループの所有者である必要があります。

drop_filegroup_clause

この句を使用すると、空のファイル・グループを削除します。`filegroup_name`に、削除するファイル・グループの名前を指定します。

CASCADE

キーワードCASCADEを使用して、空ではないファイル・グループを削除します。ファイル・グループは、キーワードCASCADEが削除されると、ファイル・グループに関連付けられたすべてのファイルが自動的に削除されます。

関連項目:

ファイル・グループの詳細は、『Automatic Storage Management管理者ガイド』を参照してください。

undrop_disk_clause

この句を使用すると、ディスク・グループからのディスクの削除を取り消すことができます。1つ以上のディスク・グループ内のすべてのディスク(`diskgroup_name`を指定)、またはすべてのディスク・グループ内のすべてのディスク(`ALL`を指定)の保留中の削除を取り消すことができます。

ディスク・グループから完全に削除されたディスクや、完全に削除されたディスク・グループに対しては、この句は無効です。この句を指定すると、実行時間が長い操作が行われます。操作の状態を確認するには、`V$ASM_OPERATION`動的パフォーマンス・ビューを問い合せます。

関連項目:

実行時間が長いOracle ASM操作の詳細は、V$ASM_OPERATIONを参照してください。

diskgroup_availability

この句を使用すると、Oracle ASMインスタンスと同じノードで実行されているデータベース・インスタンスに対して、1つ以上のディスク・グループを使用可能または使用禁止にできます。この句は、クラスタ内の他のノードのディスク・グループの状態には影響しません。

MOUNT

`MOUNT`を指定すると、ローカルOracle ASMインスタンスのディスク・グループをマウントできます。`ALL MOUNT`を指定すると、`ASM_DISKGROUPS`初期化パラメータで指定されたすべてのディスク・グループがマウントされます。ファイル操作は、ディスク・グループがマウントされている場合のみ可能です。Oracle ASMがクラスタ内またはスタンドアロン・サーバー用のOracle Grid Infrastructureで管理されたスタンドアロン・サーバー内で実行されている場合、`MOUNT`句は対応するリソースを自動的にオンラインにします。

RESTRICTED | NORMAL

これらの句を使用すると、ディスク・グループがマウントされる方式を決定できます。

- RESTRICTEDモードでは、ディスク・グループは単一インスタンス排他モードでマウントされます。同じクラスタ内の他のOracle ASMインスタンスは、そのディスク・グループをマウントできません。このモードでは、Oracle ASMクライアントはディスク・グループを使用できません。
- NORMALモードでは、ディスク・グループは共有モードでマウントされるので、他のOracle ASMインスタンスおよびクライアントがディスク・グループにアクセスできます。これはデフォルトです。

FORCE | NOFORCE

これらの句を使用すると、ディスク・グループがマウントされる環境を決定できます。

- FORCEモードでは、Oracle ASMは、ディスク・グループに属するすべてのデバイスを検出できない場合でも、そのディスク・グループをマウントしようとします。この設定は、標準冗長性または高冗長性ディスク・グループのディスマウント中に、そのディスクの一部が使用不可になった場合に役立ちます。MOUNT FORCEが成功した場合、Oracle ASMは欠落しているディスクをオフラインにします。

  Oracle ASMがディスク・グループ内のすべてのディスクを検出した場合、MOUNT FORCEは失敗します。そのため、MOUNT FORCE設定は、一部のディスクが使用不可の場合にのみ使用します。それ以外の場合は、NOFORCEを使用します。

  標準冗長性および高冗長性ディスク・グループでは、1つの障害グループのディスクが使用不可になる場合があり、MOUNT FORCEは正常に実行されます。また、高冗長性ディスク・グループでは、2つの異なる障害グループの2つのディスクが使用不可になる場合があり、MOUNT FORCEは正常に実行されます。使用不可のディスクのその他の組合せでは、Oracle ASMはすべてのユーザー・データまたはメタデータの有効なコピーが使用可能なディスク上にあることを保証できないため、操作は失敗します。
- NOFORCEモードでは、Oracle ASMは、すべてのメンバー・ディスクが検出されないかぎり、ディスク・グループをマウントしようとしません。これはデフォルトです。

関連項目:

初期化パラメータ・ファイルへのディスク・グループ名の追加の詳細は、「ASM_DISKGROUPS」を参照してください。

DISMOUNT

DISMOUNTを指定すると、指定したディスク・グループをディスマウントできます。FORCEが指定されていないかぎり、ディスク・グループのいずれかのファイルがオープンされていると、Oracle ASMによりエラーが戻されます。ALL DISMOUNTを指定すると、現在マウントされているすべてのディスク・グループがディスマウントされます。ファイル操作は、ディスク・グループがマウントされている場合のみ可能です。Oracle ASMがクラスタ内またはスタンドアロン・サーバー用のOracle Grid Infrastructureで管理されたスタンドアロン・サーバー内で実行されている場合、DISMOUNT句は対応するリソースを自動的にオフラインにします。

FORCE

FORCEを指定すると、ディスク・グループのいずれかのファイルが開いていてもディスク・グループをディスマウントするように、Oracle ASMに指示できます。

enable_disable_volume

この句を使用すると、ディスク・グループ内の1つ以上のボリュームを有効または無効にできます。

- 有効化したボリュームごとに、Oracle ASMは、ファイル・システムの作成またはマウントに使用可能なボリューム・デバイス・ファイルをローカル・ノード上に作成します。

- 無効化したボリュームごとに、Oracle ASMは、ローカル・ノード上のデバイス・ファイルを削除します。ボリューム・ファイルがローカル・ノード上でオープンしている場合は、DISABLE句によりエラーが戻されます。

ALLキーワードを使用すると、ディスク・グループ内のすべてのボリュームを有効または無効にできます。ALTER DISKGROUP ALL ...を指定する場合は、この句内でALLキーワードも使用する必要があります。

関連項目:

ディスク・グループ・ボリュームの詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

例

次の例では、dgroup_01というディスク・グループが必要です。ASM_DISKSTRINGが/devices/disks/*に設定されていることが前提となります。またOracleユーザーが/devices/disks/d100への読取り/書込み権限を持っていると想定しています。dgroup_01を作成する方法は、「ディスク・グループの作成: 例」を参照してください。

ディスク・グループへのディスクの追加: 例

ディスクd100をディスク・グループdgroup_01に追加するには、次の文を発行します。

```
ALTER DISKGROUP dgroup_01
  ADD DISK '/devices/disks/d100';
```

ディスク・グループからのディスクの削除: 例

ディスクdgroup_01_0000をディスク・グループdgroup_01から削除するには、次の文を発行します。

```
ALTER DISKGROUP dgroup_01
  DROP DISK dgroup_01_0000;
```

ディスク・グループからのディスクの削除解除: 例

ディスク・グループdgroup_01からのディスクの削除を取り消すには、次の文を発行します。

```
ALTER DISKGROUP dgroup_01
  UNDROP DISKS;
```

ディスク・グループのサイズ変更: 例

ディスク・グループdgroup_01のすべてのディスクのサイズを変更するには、次の文を発行します。

```
ALTER DISKGROUP dgroup_01
  RESIZE ALL
  SIZE 36G;
```

ディスク・グループの均衡の再調整: 例

ディスク・グループdgroup_01の均衡を手動で再調整し、Oracle ASMに対して均衡の再調整を最高速度で実行することを許可するには、次の文を発行します。

```
ALTER DISKGROUP dgroup_01
  REBALANCE POWER 11 WAIT;
```

WAITキーワードを指定すると、データベースでは、ディスク・グループの均衡が再調整されるのを待機してからユーザーに制御を戻します。

ディスク・グループ・メタデータの内部整合性の検証: 例

Oracle ASMディスク・グループ・メタデータの内部整合性を検証し、検出されたエラーを修復するようOracle ASMに指示する

には、次の文を発行します。

```
ALTER DISKGROUP dgroup_01
  CHECK ALL
  REPAIR;
```

ディスク・グループへの名前付きテンプレートの追加: 例

名前付きテンプレートtemplate_01をディスク・グループdgroup_01に追加するには、次の文を発行します。

```
ALTER DISKGROUP dgroup_01
  ADD TEMPLATE template_01
    ATTRIBUTES (UNPROTECTED COARSE);
```

ディスク・グループ・テンプレートの属性の変更: 例

システム・デフォルトまたはユーザー定義のディスク・グループ・テンプレートtemplate_01の属性を変更するには、次の文を発行します。

```
ALTER DISKGROUP dgroup_01
  MODIFY TEMPLATE template_01
    ATTRIBUTES (FINE);
```

ディスク・グループからのユーザー定義のテンプレートの削除: 例

ユーザー定義のテンプレートtemplate_01をディスク・グループdgroup_01から削除するには、次の文を発行します。

```
ALTER DISKGROUP dgroup_01
  DROP TEMPLATE template_01;
```

階層的に名付けられた別名のディレクトリ・パスの作成: 例

別名が配置されるディレクトリ構造を指定するには、次の文を発行します。

```
ALTER DISKGROUP dgroup_01
  ADD DIRECTORY '+dgroup_01/alias_dir';
```

Oracle ASMファイル名の別名の作成: 例

Oracle ASMの数値ファイル名を指定してユーザーの別名を作成するには、次の文を発行します。

```
ALTER DISKGROUP dgroup_01
  ADD ALIAS '+dgroup_01/alias_dir/datafile.dbf'
    FOR '+dgroup_01.261.1';
```

ディスク・グループのスクラブ: 例

ディスク・グループdgroup_01をスクラブするには、次の文を発行します。この文では、論理データの破損検査中に検出したエラーの修復を試行して、スクラブ操作が完了してからユーザーに制御を返します。

```
ALTER DISKGROUP dgroup_01
  SCRUB REPAIR WAIT;
```

ディスク・グループのディスマウント: 例

ディスク・グループdgroup_01をディスマウントするには、次の文を発行します。この文は、アクティブなファイルが1つ以上ある場合でも、ディスク・グループをディスマウントします。

```
ALTER DISKGROUP dgroup_01
  DISMOUNT FORCE;
```

ディスク・グループのマウント: 例

ディスク・グループdgroup_01をマウントするには、次の文を発行します。

```
ALTER DISKGROUP dgroup_01
  MOUNT;
```

ALTER FLASHBACK ARCHIVE

目的

ALTER FLASHBACK ARCHIVE文を使用すると、次の操作を実行できます。

- システムのデフォルトのフラッシュバック・アーカイブとしてのフラッシュバック・アーカイブの指定
- フラッシュバック・アーカイブによって使用される表領域の追加
- フラッシュバック・アーカイブによって使用される表領域の割当て制限の変更
- フラッシュバック・アーカイブによる使用からの表領域の削除
- フラッシュバック・アーカイブの保存期間の変更
- 不要な古いデータのフラッシュバック・アーカイブの消去

関連項目:

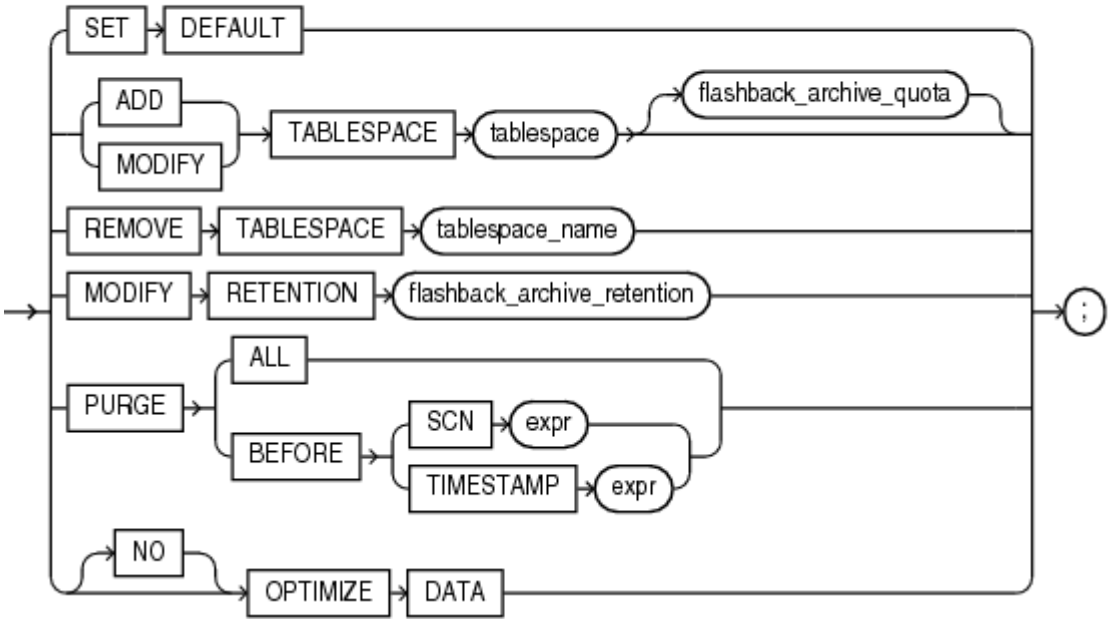
フラッシュバック・タイム・トラベルの使用についてのさらなる情報は、Oracle Database開発ガイドおよびCREATE FLASHBACK ARCHIVEを参照してください。

前提条件

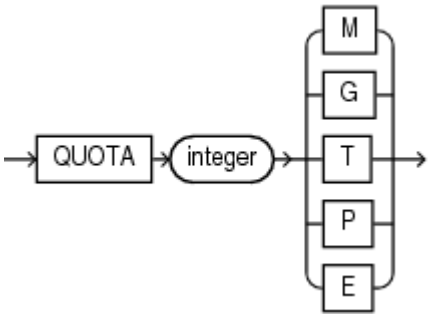
フラッシュバック・アーカイブを変更するには、FLASHBACK ARCHIVE ADMINISTERシステム権限が必要です。フラッシュバック・アーカイブ表領域を追加、変更または削除するには、影響を受ける表領域に対する適切な権限も必要です。

構文

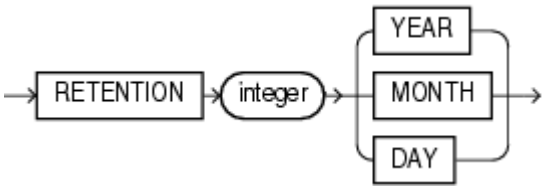
alter_flashback_archive::=



flashback_archive_quota::=



flashback_archive_retention::=



セマンティクス

flashback_archive

既存のフラッシュバック・アーカイブの名前を指定します。

SET DEFAULT

この句を指定するには、SYSDBAとしてログインしている必要があります。この句は、このフラッシュバック・アーカイブをシステムのデフォルトのフラッシュバック・アーカイブとして指定する場合に使用します。CREATE TABLE文またはALTER TABLE文に、フラッシュバック・アーカイブ名を指定しないで`flashback_archive_clause`が指定されている場合、データベースはデフォルトのフラッシュバック・アーカイブを使用して、その表のデータを格納します。

この文によって、異なるフラッシュバック・アーカイブの以前の指定はデフォルトとしてオーバーライドされます。

関連項目:

詳細は、「CREATE TABLE」の「flashback_archive_clause」を参照してください。

ADD TABLESPACE

この句は、フラッシュバック・アーカイブに表領域を追加する場合に使用します。`flashback_archive_quota`句を使用して、新規表領域内のフラッシュバック・アーカイブによって使用できる領域の量を指定できます。この句を省略すると、新しく追加された表領域内でフラッシュバック・アーカイブの領域は無制限になります。

MODIFY TABLESPACE

この句は、フラッシュバック・アーカイブによってすでに使用されている表領域の割当て制限を変更する場合に使用します。

REMOVE TABLESPACE

この句は、フラッシュバック・アーカイブによる使用から表領域を削除する場合に使用します。フラッシュバック・アーカイブによって使用される最後に残った表領域は削除できません。

削除する表領域にフラッシュバック・アーカイブの保存期間内のデータが含まれている場合は、そのデータも削除されます。そのため、この句を使用して表領域を削除する前に、データを別の表領域に移動する必要があります。

MODIFY RETENTION

この句は、フラッシュバック・アーカイブの保存期間を変更する場合に使用します。

PURGE

この句は、フラッシュバック・アーカイブからデータを消去する場合に使用します。

- PURGE ALLは、フラッシュバック・アーカイブからすべてのデータを削除する場合に指定します。この履歴情報は、フラッシュバック問合せで指定されたSCNまたはタイムスタンプがUNDO保存期間内である場合にのみ、フラッシュバック問合せを使用して取得できます。
- PURGE BEFORE SCNは、指定したシステム変更番号より前のフラッシュバック・アーカイブからすべてのデータを削除する場合に指定します。
- PURGE BEFORE TIMESTAMPは、指定したタイムスタンプより前のフラッシュバック・アーカイブからすべてのデータを削除する場合に指定します。

[NO] OPTIMIZE DATA

この句のセマンティクスは、CREATE FLASHBACK ARCHIVEの[NO] OPTIMIZE DATA句と同じです。

関連項目:

フラッシュバック・アーカイブの作成の詳細およびフラッシュバック・アーカイブの簡単な使用例は、CREATE FLASHBACK ARCHIVEを参照してください。

ALTER FUNCTION

目的

ファンクションはPL/SQLを使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

ALTER FUNCTION文を使用すると、無効なスタンドアロンのストアド・ファンクションを再コンパイルできます。明示的に再コンパイルすることによって、実行時に暗黙的に再コンパイルする必要がなくなり、また、実行時のコンパイル・エラーとパフォーマンス上のオーバーヘッドもなくなります。

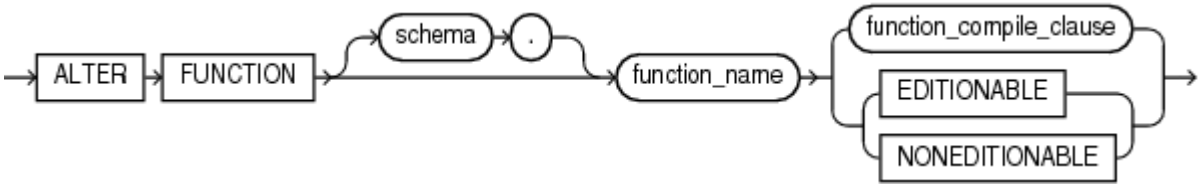
この文では、既存のファンクションの宣言または定義は変更されません。ファンクションを再宣言または再定義する場合は、OR REPLACE句を指定してCREATE FUNCTION文を使用します。「CREATE FUNCTION」を参照してください。

前提条件

ファンクションが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY PROCEDUREシステム権限が必要です。

構文

alter_function::=



(function_compile_clause: この句の構文の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。)

セマンティクス

schema

ファンクションが含まれているスキーマを指定します。schemaを指定しない場合、ファンクションは自分のスキーマ内にあるとみなされます。

function_name

再コンパイルするファンクション名を指定します。

function_compile_clause

この句の構文とセマンティクスの詳細およびファンクションの作成とコンパイルの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

EDITIONABLE | NONEDITIONABLE

この句を使用すると、schemaのスキーマ・オブジェクト・タイプFUNCTIONのエディショニングが後で有効化されたときに、そのファンクションをエディション・オブジェクトにするか非エディション・オブジェクトにするかを指定できます。デフォルトは、EDITIONABLEです。エディション・オブジェクトと非エディション・オブジェクトの変更の詳細は、『Oracle Database開発ガイド』を参照してください。

ALTER HIERARCHY

目的

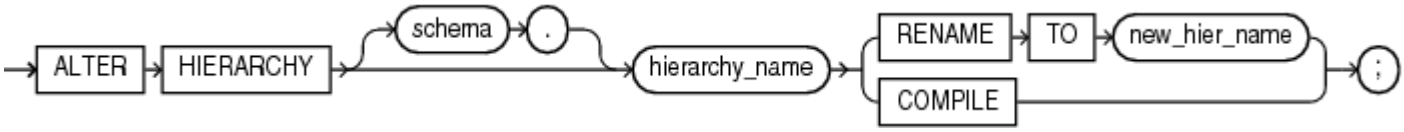
ALTER HIERARCHY文を使用して、階層の名前を変更したり、階層をコンパイルできます。その他の変更には、CREATE OR REPLACE HIERARCHYを使用してください。

前提条件

自分のスキーマ内で階層を変更する場合は、ALTER HIERARCHYシステム権限が必要です。別のユーザーのスキーマ内で階層を変更する場合は、ALTER ANY HIERARCHYシステム権限を持っているか、階層に直接ALTERが付与されている必要があります。

構文

alter_hierarchy::=



セマンティクス

schema

階層が存在するスキーマを指定します。スキーマを指定しない場合、自分のスキーマ内で階層が検索されます。

hierarchy_name

階層名を指定します。

RENAME TO

RENAME TOを指定すると、階層の名前を変更できます。

COMPILE

COMPILEを指定すると、階層をコンパイルできます。

new_hier_name

階層の新しい名前を指定します。

例

次の文は、階層の名前を変更します。

```
ALTER HIERARCHY product_hier RENAME TO myproduct_hier;
```

ALTER INDEX

目的

ALTER INDEX文を使用すると、既存の索引を変更または再作成できます。

関連項目:

索引の作成については、「CREATE INDEX」を参照してください。

前提条件

索引が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY INDEXシステム権限が必要です。

MONITORING USAGE句を実行する場合は、索引は自分のスキーマ内に存在する必要があります。

ドメイン索引を変更する場合は、索引の索引タイプに対してEXECUTEオブジェクト権限が必要です。

オブジェクト権限は、個々の索引パーティションまたはサブパーティションではなく、親索引に付与されている必要があります。

索引パーティションの変更、再作成または分割、索引サブパーティションの変更または再作成を行う場合は、表領域割当て制限が必要です。

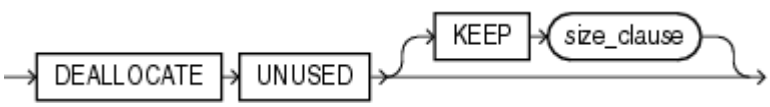
構文

alter_index::=

(deallocate_unused_clause::=、allocate_extent_clause::=、shrink_clause::=、parallel_clause::=、physical_attributes_clause::=、logging_clause::=、partial_index_clause::=、rebuild_clause::=、alter_index_partitioning::=)

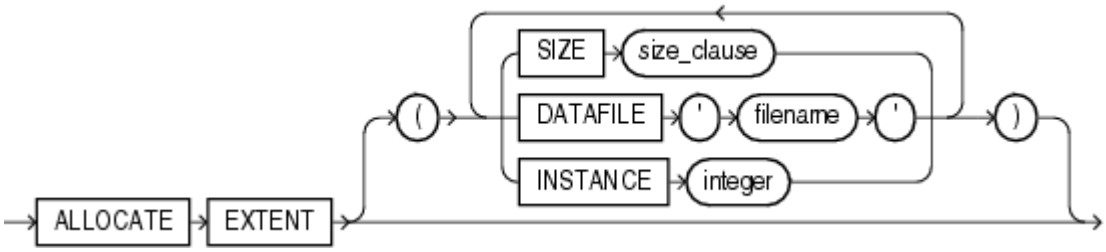
(ODCI_parametersについては、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。)

deallocate_unused_clause::=



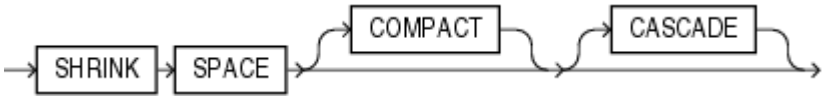
(size_clause::=)

allocate_extent_clause::=

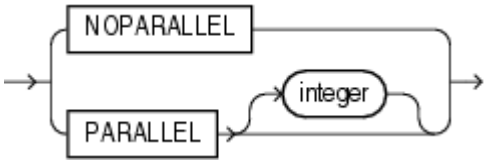


(size_clause::=)

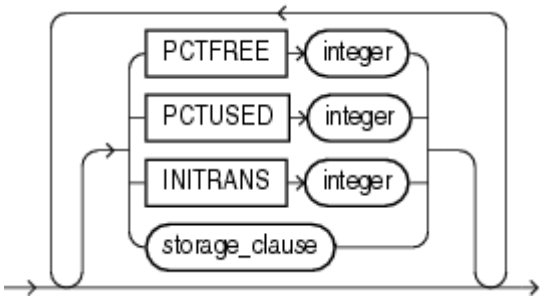
shrink_clause::=



parallel_clause::=

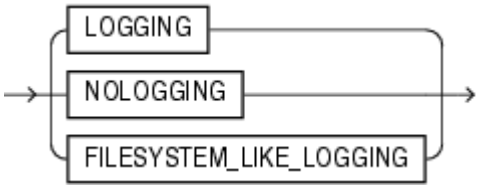


physical_attributes_clause::=

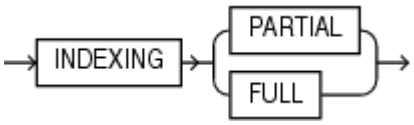


(storage_clause::=)

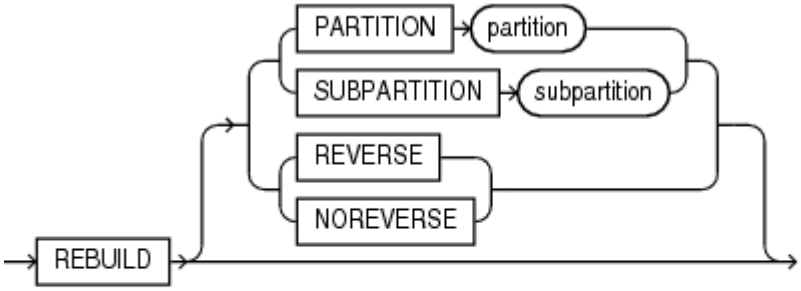
logging_clause::=

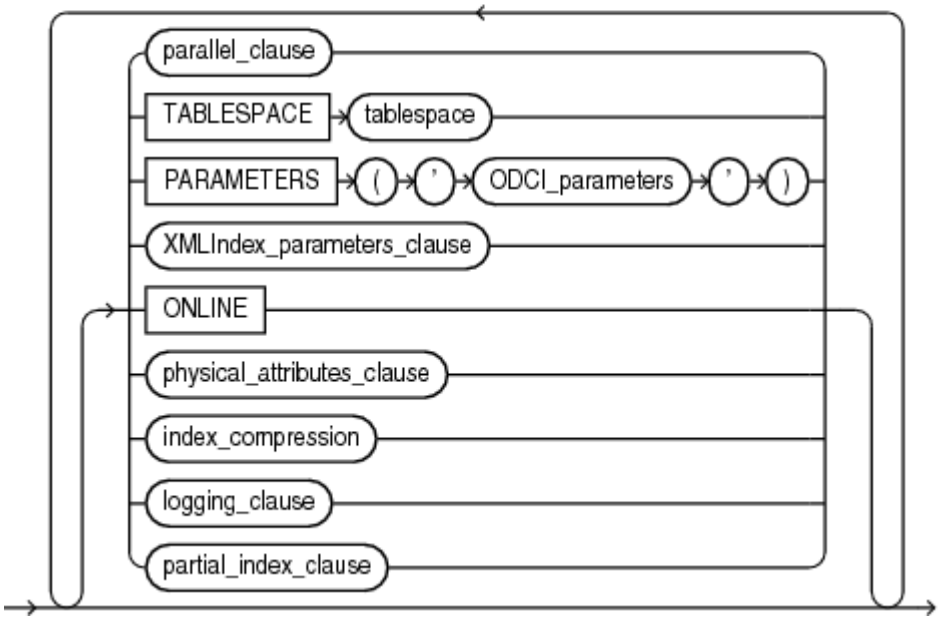


partial_index_clause::=



rebuild_clause::=

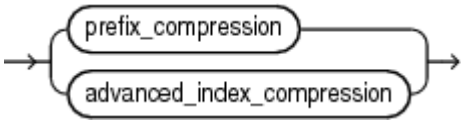




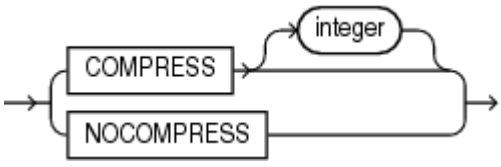
(parallel_clause::=、physical_attributes_clause::=、index_compression::=、logging_clause::=、partial_index_clause::=)

(`ODCI_parameters`については、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。
`XMLIndex_parameters_clause`については、『Oracle XML DB開発者ガイド』を参照してください。

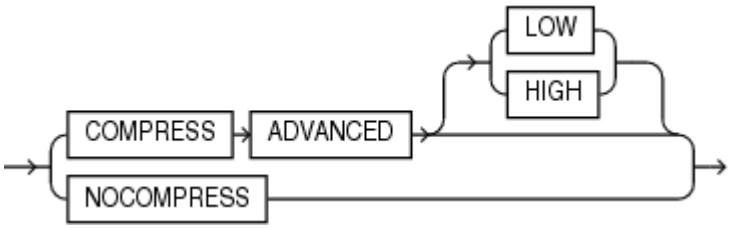
index_compression::=



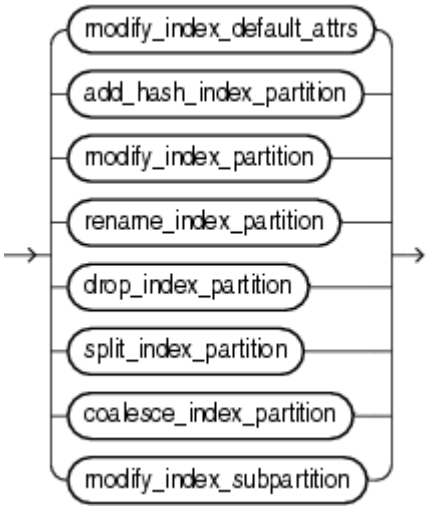
prefix_compression::=



advanced_index_compression::=

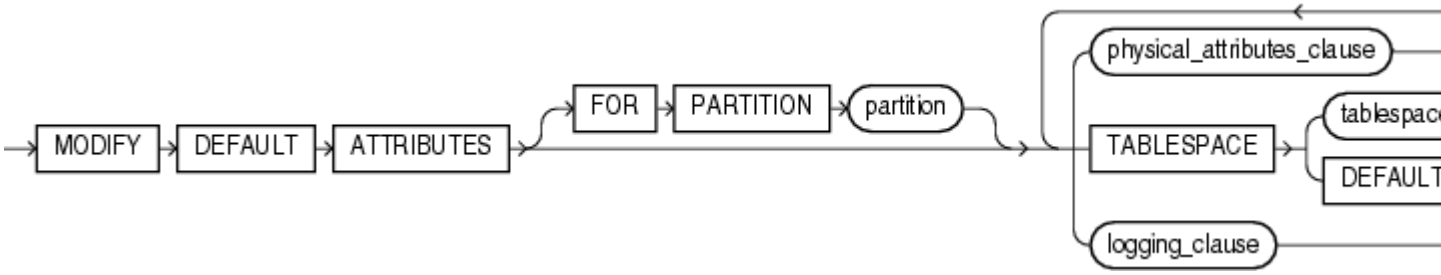


alter_index_partitioning::=



(modify_index_default_attrs::=、add_hash_index_partition::=、modify_index_partition::=、rename_index_partition::=、drop_index_partition::=、split_index_partition::=、coalesce_index_partition::=、modify_index_subpartition::=)

modify_index_default_attrs::=

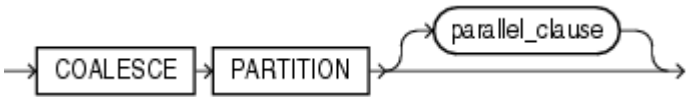


(physical_attributes_clause::=、logging_clause::=)

add_hash_index_partition::=

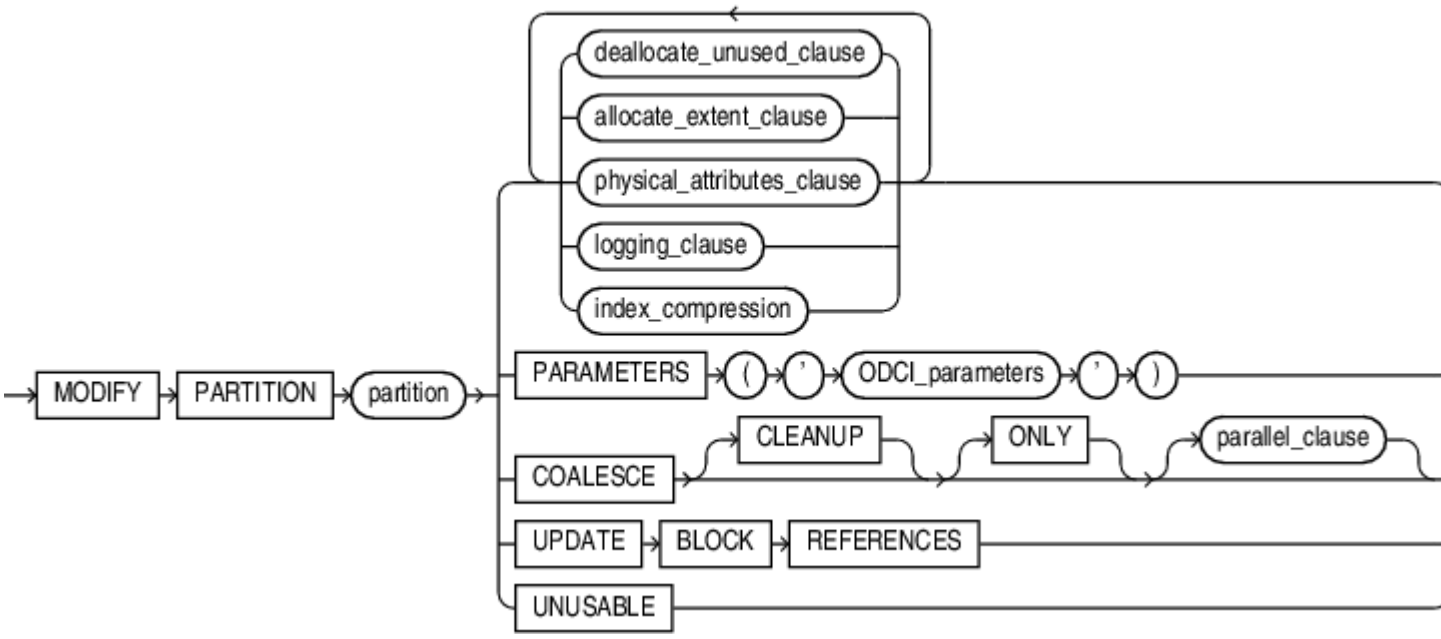
(index_compression::=、parallel_clause::=)

coalesce_index_partition::=



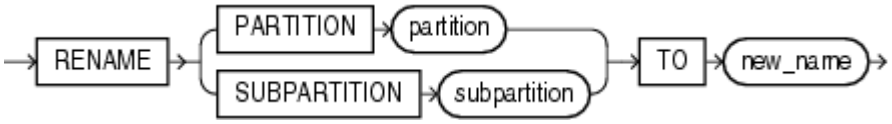
(parallel_clause::=)

modify_index_partition::=

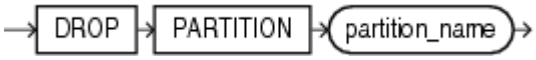


(deallocate_unused_clause::=、allocate_extent_clause::=、physical_attributes_clause::=、logging_clause::=、index_compression::=)

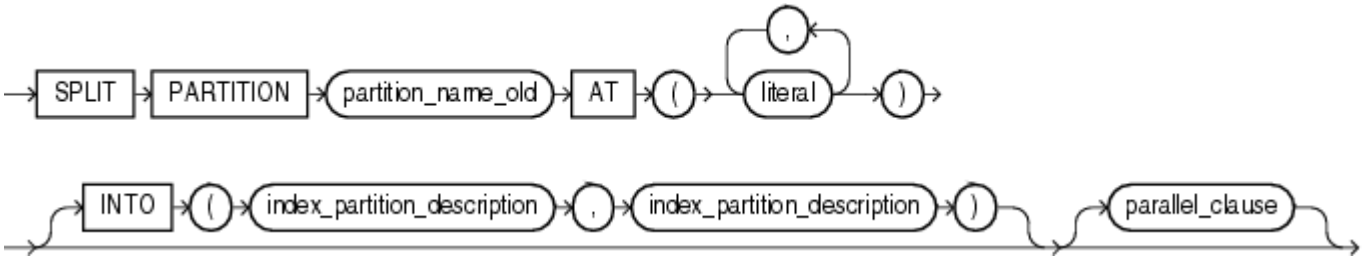
rename_index_partition::=



drop_index_partition::=

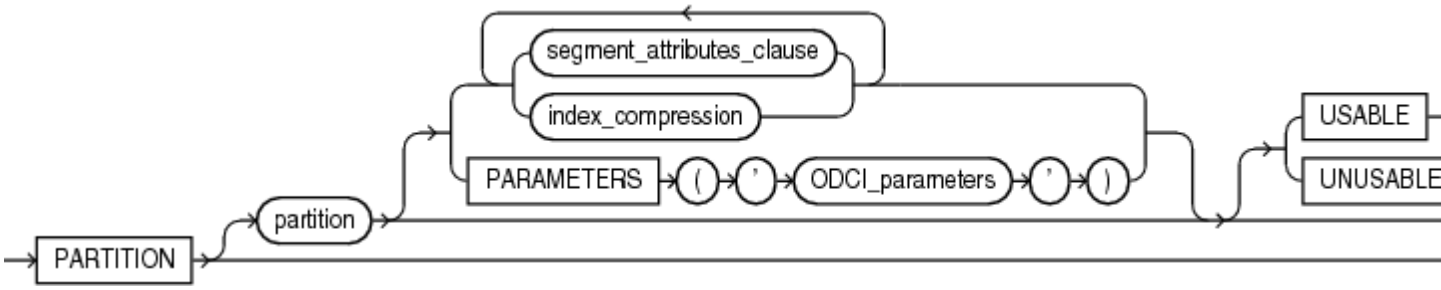


split_index_partition::=



(parallel_clause::=)

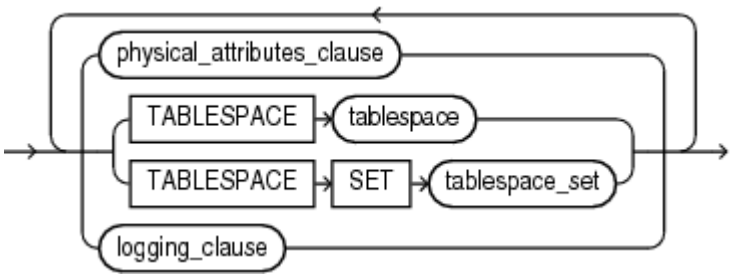
index_partition_description::=



(segment_attributes_clause::=、index_compression::=)

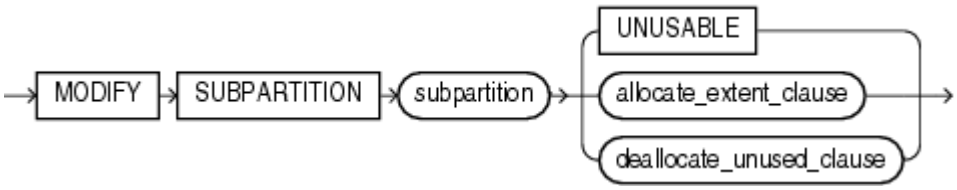
> ノート:
>
> USABLE キーワードと UNUSABLE キーワードは、split_index_partition 句に index_partition_description が指定されているときにはサポートされません。

segment_attributes_clause::=



(physical_attributes_clause::=、TABLESPACE SET: ALTER INDEXではサポートされていません、logging_clause::=)

modify_index_subpartition::=



(allocate_extent_clause::=、deallocate_unused_clause::=)

セマンティクス

schema

索引が含まれているスキーマを指定します。schemaを指定しない場合、索引は自分のスキーマ内にあるとみなされます。

index

変更する索引の名前を指定します。

索引の変更の制限事項

索引の変更には、次の制限事項があります。

- indexがドメイン索引である場合は、PARAMETERS句、RENAME句、rebuild_clause(PARAMETERS句の有無に関係なく)、parallel_clauseまたはUNUSABLE句のみ指定できます。その他のすべての句は無効です。
- LOADINGまたはFAILEDのマークが付いているドメイン索引は、変更または名前の変更ができません。索引にFAILEDのマークが付いている場合、REBUILD句のみ指定できます。

関連項目:

ドメイン索引のLOADINGおよびFAILED状態の詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。

deallocate_unused_clause

deallocate_unused_clause句を使用すると、索引の終わりの未使用領域の割当てを明示的に解除し、解放された領域が表領域内の他のセグメントで使用可能になります。

indexがレンジ・パーティションまたはハッシュ・パーティションである場合、各索引パーティションの未使用領域の割当てが解除されます。indexがコンポジット・パーティション表のローカル索引である場合、各索引サブパーティションの未使用領域の割当てが解除されます。

領域の割当て解除の制限事項

領域の割当て解除には、次の制限事項があります。

- この句は、一時表の索引に対して指定できません。
- この句およびrebuild_clauseは、指定できません。

この句の詳細は、「deallocate_unused_clause」を参照してください。

KEEP integer

KEEP句を使用すると、割当てを解除した後に索引に残す、最高水位標を超えるバイト数を指定できます。残りのエクステント数がMINEXTENTSより少ない場合、MINEXTENTSは現行のエクステント数に設定されます。初期エクステントがINITIALより小さくなると、INITIALは初期エクステントの現行の値に設定されます。KEEPを指定しないと、すべての未使用領域が解放されます。

この句の詳細は、「ALTER TABLE」を参照してください。

allocate_extent_clause

allocate_extent_clauseを使用すると、索引の新しいエクステントを明示的に割り当てることができます。ハッシュ・パーティション表のローカル索引に対して、新規エクステントが索引の各パーティションに割り当てられます。

エクステントの割当ての制限事項

この句は、一時表の索引、レンジ・パーティションまたはコンポジット・パーティション索引に対しては指定できません。

この句の詳細は、「allocate_extent_clause」を参照してください。

shrink_clause

この句を使用すると、索引セグメントを縮小化できます。ALTER INDEX ... SHRINK SPACE COMPACTを指定することは、ALTER INDEX ... COALESCEを指定することと同じです。

この句の詳細は、「CREATE TABLE」の「shrink_clause」を参照してください。

索引セグメントの縮小の制限事項

この句は、ビットマップ結合索引またはファンクション索引に対しては指定できません。

parallel_clause

PARALLEL句を使用すると、索引の問合せおよびDMLに対するデフォルトの並列度を変更できます。

索引のパラレル化の制限事項

この句は、一時表の索引に対して指定できません。

この句の詳細は、「CREATE TABLE」の「parallel_clause」を参照してください。

関連項目:

パラレル問合せの有効化: 例

physical_attributes_clause

`physical_attributes_clause`を使用すると、非パーティション索引、パーティション索引のすべてのパーティションおよびサブパーティション、指定されたパーティション、または指定されたパーティションのすべてのサブパーティションに対するパラメータの値を変更できます。

関連項目:

- この句のパラメータの詳細は、「CREATE TABLE」を参照してください。
- 「索引の実属性の変更: 例」および「MAXEXTENTSの変更: 例」

索引の物理属性の制限事項

索引の物理属性には、次の制限事項があります。

- この句は、一時表の索引に対して指定できません。
- 索引の変更中は、PCTUSEDパラメータを指定できません。
- PCTFREEパラメータは、`rebuild_clause`、`modify_index_default_attrs`句または`split_index_partition`句の一部としてのみ指定できます。

storage_clause

`storage_clause`を使用すると、非パーティション索引、索引パーティション、またはパーティション索引のすべてのパーティションの記憶域パラメータ、あるいはパーティション索引の記憶域パラメータのデフォルト値を変更できます。この句の詳細は、「storage_clause」を参照してください。

logging_clause

`logging_clause`を使用すると、索引のロギング属性を変更できます。REBUILD句も指定すると、この新しい設定は再構築操作に影響します。REBUILD句でロギングに異なる値を指定した場合、索引および再構築操作のロギング属性として指定された最後のロギング値が使用されます。

索引セグメントには、実表の属性と異なるロギング属性、および同じ実表の他の索引セグメントと異なるロギング属性を指定できます。

索引のログの制限事項

この句は、一時表の索引に対して指定できません。

関連項目:

- この句の詳細は、「logging_clause」を参照してください。
- パラレルDMLの詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

partial_index_clause

partial_index_clauseを使用すると、索引を全索引または部分索引に変更できます。INDEXING FULLを指定すると、索引を全索引に変更できます。INDEXING PARTIALを指定すると、索引を部分索引に変更できます。この句は、パーティション表の索引に対してのみ有効です。この句のセマンティクスの詳細は、CREATE INDEXの「partial_index_clause」を参照してください。

RECOVERABLE | UNRECOVERABLE

これらのキーワードは以前のリリースで非推奨になったもので、それぞれLOGGINGおよびNOLOGGINGに置き換えられています。RECOVERABLEおよびUNRECOVERABLEは、下位互換性のためにサポートされていますが、LOGGINGおよびNOLOGGINGキーワードを使用することをお薦めします。

RECOVERABLEは、パーティション表またはLOBの記憶特性の作成時には無効なキーワードです。UNRECOVERABLEは、パーティション表または索引構成表の作成時には無効なキーワードです。また、CREATE INDEXのAS副問合せ句を使用してのみ指定できます。

rebuild_clause

rebuild_clauseを使用すると、既存の索引、あるいはパーティションまたはサブパーティションのいずれかを再構築できます。索引にUNUSABLEのマークが付いている場合、正常に再構築するとUSABLEになります。ファンクション索引も使用可能にします。索引の基になるファンクションが存在しない場合、再構築文は正常に実行されません。

ノート:

索引構成表の 2 次索引を再構築する場合、Oracle Database は、索引が作成されたときの論理 ROWID に含まれる主キー列を保持します。したがって、COMPATIBLE 初期化パラメータが 10.0.0 未満に設定された状態で索引が作成された場合、再構築された索引には、索引キーと、索引キーには含まれない表の主キー列が含まれます。COMPATIBLE 初期化パラメータが 10.0.0 以上に設定された状態で索引が作成された場合、再構築された索引には、索引キーと、索引キーに含まれる主キー列を含む、表のすべての主キー列が含まれます。

索引の再構築の制限事項

索引の再構築には、次の制限事項があります。

- 一時表の索引は、再構築できません。
- INVALIDのマークが付いているビットマップ索引は再構築できません。制約を削除してからそれを再作成する必要があります。
- パーティション索引全体は、再構築できません。PARTITION句で説明するとおり、各パーティションまたはサブパーティションを再構築する必要があります。

- 同じ文でdeallocate_unused_clauseとrebuild_clauseを指定することはできません。
- 索引全体(ALTER INDEX)またはパーティション(ALTER INDEX ... MODIFY PARTITION)に対して、PCTFREEパラメータ値を変更できません。ALTER INDEX文の他のすべての形式では、PCTFREEを指定できます。
- ドメイン索引の場合
    - 指定できるのは、PARAMETERS句(索引または索引のパーティション用)またはparallel_clauseのみです。その他の再構築の句は無効です。
    - 索引にIN_PROGRESSのマークが付いていない場合にのみ、索引を再構築できます。
    - 索引にIN_PROGRESSまたはFAILEDのマーク、パーティションにIN_PROGRESSのマークが付いていない場合にのみ、索引パーティションを再構築できます。
- ローカル索引は再構築できませんが、ALTER INDEX ... REBUILD PARTITIONでローカル索引のパーティションを再構築できます。
- ハッシュ・パーティションまたはハッシュ・サブパーティションのローカル索引に指定できるパラメータは、TABLESPACEのみです。
- 遅延可能一意制約を適用するために使用されるオンライン索引は、再構築できません。

PARTITION句

PARTITION句を使用すると、索引の1つのパーティションを再構築できます。この句は、索引パーティションを別の表領域に移動したり、作成時の物理属性を変更するために使用できます。

ブロック・サイズが異なる表領域のパーティション化されたデータベース・エンティティの記憶域には、制限事項があります。これらの制限事項については、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

パーティションの再構築の制限事項

コンポジット・パーティション表のローカル索引に対してはこの句を指定できません。かわりに、REBUILD SUBPARTITION句を使用してください。

関連項目:

パーティションのメンテナンス操作の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』および「使用禁止の索引パーティションの再構築: 例」を参照してください。

SUBPARTITION句

SUBPARTITION句を使用すると、索引の1つのサブパーティションを再構築できます。この句を使用して、索引サブパーティションを他の表領域に移動することもできます。TABLESPACEを指定しないと、サブパーティションは同じ表領域に再構築されます。

ブロック・サイズが異なる表領域のパーティション化されたデータベース・エンティティの記憶域には、制限事項があります。これらの制限事項については、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

索引サブパーティションの変更の制限事項

パラメータTABLESPACE、ONLINEおよびparallel_clause以外は、サブパーティションに対して指定できません。

REVERSE | NOREVERSE

索引ブロックのバイトを逆順に格納するかどうかを示します。

- REVERSEを指定すると、索引の再構築時に、索引ブロックのバイトが逆順で格納され、ROWIDが除外されます。
- NOREVERSEを指定すると、索引の再構築時に、逆順にせずに索引ブロックのバイトが格納されます。NOREVERSEキーワードを指定せずにREVERSE索引を再構築すると、再構築された索引は、逆キーの索引になります。

逆索引の制限事項

逆索引には、次の制限事項があります。

- ビットマップ索引または索引構成表は逆順には格納できません。
- パーティションまたはサブパーティションに対して、REVERSEまたはNOREVERSEを指定できません。

関連項目:

索引ブロックの逆順格納：例

parallel_clause

`parallel_clause`を使用すると、索引の再構築をパラレル化して索引自体の並列度を変更できます。その索引に対する後続のすべての操作は、次にデータ定義言語(DDL)文の`parallel_clause`によって上書きされるまで、この句に指定された並列度で実行されます。次の例外があります。

- 索引を再構築する前に`ALTER SESSION DISABLE PARALLEL DDL`が指定されている場合は、索引がシリアルで再構築され、索引の並列度が1に変更されます。
- 索引を再構築する前に`ALTER SESSION FORCE PARALLEL DDL`が指定されている場合は、索引がパラレルで再構築され、索引の並列度が`ALTER SESSION`文で指定された値に変更されます。値がない場合は、`DEFAULT`に変更されます。

関連項目:

索引のパラレル再構築：例

TABLESPACE句

再構築された索引、索引パーティションまたは索引サブパーティションが格納される表領域を指定します。デフォルトは、再構築の前に索引またはパーティションが格納されていた表領域です。

index_compression

`index_compression`句を使用して、索引の索引圧縮を有効化または無効化します。`prefix_compression`句を指定して、索引の接頭辞圧縮を有効化または無効化します。`advanced_index_compression`句を指定して、索引の拡張索引圧縮を有効化または無効化します。

`index_compression`句は、`CREATE INDEX`および`ALTER INDEX`に対して同じセマンティクスを持ちます。これらの句の詳細は、`CREATE INDEX`のindex_compressionを参照してください。

ONLINE句

`ONLINE`を指定すると、表またはパーティションのDML操作を索引の再構築中に可能にするかどうかを指定できます。

オンライン索引の制限事項

オンライン索引には、次の制限事項があります。

- オンライン索引の作成中は、パラレルDMLはサポートされません。ONLINEを指定し、続いてパラレルDML文を発行すると、Oracle Databaseはエラーを戻します。
- ビットマップ結合索引またはクラスタ索引には、ONLINEを指定できません。
- 索引構成表の一意でない2次索引の場合、索引構成表内の索引キー列の数と論理ROWIDの主キー列の数の合計は、32以下にする必要があります。論理ROWIDは、索引キーに含まれる列を除外します。

logging_clause

ALTER INDEX ... REBUILD操作をログに記録するかどうかを指定します。

この句の詳細は、「logging_clause」を参照してください。

PARAMETERS句

この句は、トップレベルのALTER INDEX文およびrebuild_clauseのドメイン索引に対してのみ有効です。この句は、未解析のまま適切なODCI索引タイプ・ルーチンに渡されたパラメータ文字列を指定します。

パラメータ文字列の最大長は1,000文字です。

索引全体を変更または再構築する場合、文字列は索引レベルのパラメータを参照する必要があります。索引のパーティションを再構築する場合、文字列はパーティション・レベルのパラメータを参照する必要があります。

indexにUNUSABLEのマークが付いている場合、パラメータを変更するのみではUSABLEにはなりません。UNUSABLEのマークが付いた索引を使用可能にするには、その索引を再構築する必要があります。

Oracle Textがインストール済の場合、Oracle Text固有のパラメータを使用するOracle Textのドメイン索引を再構築することができます。パラメータの詳細は、『Oracle Textリファレンス』を参照してください。

PARAMETERS句の制限事項

indexにIN_PROGRESSおよびFAILEDのマークが付いておらず、索引パーティションにIN_PROGRESSのマークが付いておらず、変更するパーティションにFAILEDのマークが付いていない場合にのみ、索引パーティションを変更できます。

関連項目:

- ドメイン索引の索引タイプ・ルーチンの詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。
- ドメイン索引の詳細は、「CREATE INDEX」を参照してください。

XMLIndex_parameters_clause

この句は、XMLIndex索引に対してのみ有効です。この句は、XMLIndex実装を定義するパラメータ文字列を指定します。

パラメータ文字列の最大長は1,000文字です。

索引全体を変更または再構築する場合、文字列は索引レベルのパラメータを参照する必要があります。索引のパーティションを再構築する場合、文字列はパーティション・レベルのパラメータを参照する必要があります。

indexにUNUSABLEのマークが付いている場合、パラメータを変更するのみではUSABLEにはなりません。UNUSABLEのマークが付いた索引を使用可能にするには、その索引を再構築する必要があります。

関連項目:

XMLIndex_parameters_clauseの構文およびセマンティクスを含むXMLIndexの詳細は、『Oracle XML DB開発者ガイド』を参照してください

XMLIndex_parameters_clauseの制限事項

indexにIN_PROGRESSおよびFAILEDのマークが付いておらず、索引パーティションにIN_PROGRESSのマークが付いておらず、変更するパーティションにFAILEDのマークが付いていない場合にのみ、索引パーティションを変更できます。

{ DEFERRED | IMMEDIATE } INVALIDATION

この句を使用すると、索引の再構築中または索引にUNUSABLEのマークを付けているときにデータベースで依存カーソルがいつ無効化されるかを制御できます。

- DEFERRED INVALIDATIONを指定した場合、データベースでは可能な場合、依存カーソルの無効化を回避または遅延します。
- IMMEDIATE INVALIDATIONを指定した場合、データベースでは、Oracle Database 12cリリース1 (12.1)以前のリリースでの動作と同様に、依存カーソルを即時に無効にします。これはデフォルトです。

この句を省略した場合、いつカーソルが無効になるかは、CURSOR_INVALIDATION初期化パラメータの値によって決定されます。

関連項目:

- カーソルの無効化の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。
- CURSOR_INVALIDATION初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

COMPILE句

この句を使用すると、無効な索引を明示的に再コンパイルできます。ドメイン索引の場合、この句は、システム管理されたドメイン索引をサポートするように基礎となる索引タイプが変更され、その結果として既存のドメイン索引がINVALIDとマークされた場合に有効です。これにより、このALTER INDEX文によって、ドメイン索引はユーザー管理されたドメイン索引からシステム管理されたドメイン索引に移行されます。すべてのタイプの索引の場合、ALTER TABLE文によって索引がINVALIDとマークされている場合にこの句が役立ちます。この場合、このALTER INDEX文は、再作成しないで索引を再検証します。

関連項目:

システム管理されたドメイン索引の作成の詳細は、「CREATE INDEXTYPE」の「storage_table_clause」および『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。

ENABLE句

ENABLEは、使用禁止になったファンクション索引(ALTER INDEX ... DISABLE文によって使用禁止になったものか、その索引で使用されるユーザー定義ファンクションが削除または置換されたために使用禁止になったもののいずれか)にのみ適用されます。次の条件が該当する場合、この句によって、このような索引が使用可能になります。

- ファンクションが現在有効な場合。
- 現行のファンクションのシグネチャが、索引が作成された場合のファンクションのシグネチャと一致する場合。
- ファンクションに現在DETERMINISTICのマークが付いている場合

ファンクション索引の有効化の制限事項

ENABLE句には、次の制限事項があります。

- ENABLEと同じ文内では、ALTER INDEXの他の句の指定はできません。
- この句は、一時表の索引に対して指定できません。かわりに、その索引を削除し、再作成する必要があります。DBMS_METADATAパッケージを使用して、その索引の作成DDLを取得できます。

DISABLE句

DISABLEは、ファンクション索引のみに適用されます。この句を使用してファンクション索引を使用禁止にします。たとえば、ファンクションの本体を処理する場合に、これを行います。その後、ENABLEキーワードを使用して、索引を再構築、または別のALTER INDEX文を指定できます。

USABLE | UNUSABLE

索引、索引パーティションまたは索引サブパーティションにUNUSABLEのマークを付けるには、UNUSABLEを指定します。索引、索引パーティションまたは索引サブパーティションにUNUSABLEのマークを付けると、そのオブジェクトに割り当てられていた領域はすぐに解放されます。使用禁止の索引を使用可能にする場合、再構築するか、または削除して再作成する必要があります。1つのパーティションにUNUSABLEのマークが付いている場合も、同じ索引の他のパーティションは有効です。その索引を必要とする文が使用禁止のパーティションにアクセスしない場合、その文を実行できます。また、使用禁止のパーティションは、分割または名前を変更してから再構築できます。詳細は、CREATE INDEX ... USABLE | UNUSABLEを参照してください。

ONLINE

ONLINEを指定すると、表またはパーティションに対するDML操作を許可しながら、索引にUNUSABLEのマークを付けることを指示できます。この句を指定すると、データベースは索引セグメントを削除しないようになります。

索引へのUnusableのマーク付けの制限事項

索引へのUnusableのマーク付けには、次の制限事項が適用されます。

- 一時表の索引に対してUNUSABLEは指定できません。
- グローバル索引がパーティション・メンテナンス操作中にUNUSABLEにマーク付けされた場合、データベースは使用禁止索引セグメントを削除しません。

VISIBLE | INVISIBLE

この句を使用すると、オプティマイザで索引を参照可能にするかどうかを指定できます。この句の詳細は、「CREATE INDEX」の「VISIBLE | INVISIBLE」を参照してください。

RENAME句

この句を指定すると、索引の名前を変更できます。new_index_nameは単一の識別子で、スキーマ名は含まれません。

索引の名前変更の制限事項

ドメイン索引では、indexおよびindexのパーティションのいずれも、IN_PROGRESS状態またはFAILED状態にできません。

関連項目:

- 『データ・カートリッジ開発者ガイド』のドメイン索引の作成を参照してください。
- 『データ・カートリッジ開発者ガイド』の拡張可能索引付けインタフェースを参照してください。

- 索引の名前変更：例

COALESCE句

COALESCEを指定すると、ブロックを再利用するために、索引ブロックの内容を空きブロックにマージできます(可能な場合)。

CLEANUP

CLEANUPを指定すると、表のパーティション・メンテナンス操作で以前に削除または切り捨てられたレコードに対する孤立した索引エントリを削除できます。

孤立した索引エントリが索引に含まれているかどうかを判断するには、USER_、DBA_、ALL_INDEXESデータ・ディクショナリ・ビューのORPHANED_ENTRIES列を問い合せます。詳細は、『Oracle Databaseリファレンス』を参照してください。

ONLY

索引ブロックを結合せずに索引をクリーン・アップする場合は、ONLYを指定します。

parallel_clause

parallel_clauseを使用すると、結合操作をパラレル化するかどうかを指定できます。

この句の詳細は、「CREATE TABLE」の「parallel_clause」を参照してください。

索引ブロックの結合の制限事項

索引ブロックの結合には、次の制限事項があります。

- この句は、一時表の索引に対して指定できません。
- この句は、索引構成表の主キー索引に対して指定できません。かわりにALTER TABLEのCOALESCE句を使用します。

関連項目:

- 領域管理および索引の結合の詳細は、『Oracle Database管理者ガイド』を参照してください。
- 索引構成表の領域の結合の詳細は、「COALESCE句」を参照してください。
- 索引セグメントを縮小化する他の方法については、「shrink_clause」を参照してください。

MONITORING USAGE | NOMONITORING USAGE

この句を使用すると、索引の使用を監視するかどうかを決定できます。

- MONITORING USAGEを指定すると、索引の監視が開始されます。まず、索引の使用に関する既存の情報が削除され、ALTER INDEX ... NOMONITORING USAGE文が次に実行されるまで、索引の使用が監視されます。
- NOMONITORING USAGEを指定すると、索引の監視を終了できます。

このALTER INDEX ... NOMONITORING USAGE文が発行された後、索引が使用されたかどうかを調べるには、USER_OBJECT_USAGEデータ・ディクショナリ・ビューのUSED列を問い合せます。

関連項目:

USER_OBJECT_USAGEデータ・ディクショナリ・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

UPDATE BLOCK REFERENCES句

UPDATE BLOCK REFERENCES句は、索引構成表の通常のドメイン索引に対してのみ有効です。この句を指定すると、主キーによって識別されるブロックの適切なデータベース・アドレスとともに索引行の一部として格納され、失効したと推測されるすべてのデータ・ブロック・アドレスを更新することができます。

ドメイン索引では、AlterIndexUpdBlockRefsに設定されたalter_optionパラメータでODCIIndexAlterルーチンが実行されます。このルーチンはカートリッジ・コードを使用可能にし、失効したと推測される索引のデータ・ブロック・アドレスを更新します。

UPDATE BLOCK REFERENCESの制限事項

この句は、ALTER INDEXの他の句と組み合せることはできません。

alter_index_partitioning

ALTER INDEX文のパーティション化句は、パーティション索引に対してのみ有効です。

ブロック・サイズが異なる表領域のパーティション化されたデータベース・エンティティの記憶域には、制限事項があります。これらの制限事項については、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

索引パーティションの変更の制限事項

索引パーティションの変更には、次の制限事項があります。

これらの句は、一時表の索引に対して指定できません。

ベース索引に対するいくつかの操作を1つのALTER INDEX文にまとめることはできますが(RENAMEおよびREBUILDは除く)、パーティション操作を、他のパーティション操作またはベース索引に対する操作と組み合せることはできません。

modify_index_default_attrs

パーティション索引のデフォルト属性に新しい値を指定します。

パーティションのデフォルト属性の変更の制限事項

ハッシュ・パーティション・グローバル索引またはハッシュ・パーティション表の索引に対して指定できる属性は、TABLESPACEのみです。

TABLESPACE

索引の新規パーティション、または索引パーティションのサブパーティションに対して、デフォルトの表領域を指定します。

logging_clause

パーティション化された索引または索引パーティションのデフォルト・ロギング属性を指定します。

この句の詳細は、「logging_clause」を参照してください。

FOR PARTITION

FOR PARTITION句を使用すると、コンポジット・パーティション表にあるローカル索引のパーティションのサブパーティションに対して、デフォルト属性を指定できます。

FOR PARTITIONの制限事項

FOR PARTITIONは、リスト・パーティションには指定できません。

関連項目:

デフォルト属性の変更: 例

add_hash_index_partition

この句を使用すると、ハッシュ・パーティション・グローバル索引にパーティションを追加できます。Oracle Databaseは、ハッシュ・パーティションを追加し、ハッシュ・ファンクションによって索引の既存のハッシュ・パーティションから再ハッシュされた索引エントリをそのハッシュ・パーティションに移入します。パーティション名を省略すると、SYS_Pnの形式でパーティション名が割り当てられます。TABLESPACE句を省略すると、その索引に対して指定された表領域にパーティションが配置されます。索引の表領域が指定されていない場合、パーティションは、ユーザーのデフォルトの表領域(指定されている場合)またはシステムのデフォルトの表領域に配置されます。

modify_index_partition

`modify_index_partition`句を使用すると、索引パーティション`partition`またはそのサブパーティションの実物理属性、ロギング属性または記憶特性を変更できます。ハッシュ・パーティション・グローバル索引の場合、この句に指定できる副次句はUNUSABLEのみです。

COALESCE

この句を指定すると、ブロックを再利用するために、索引パーティション・ブロックの内容を空きブロックにマージできます(可能な場合)。

CLEANUP

CLEANUPを指定すると、表のパーティション・メンテナンス操作で以前に削除または切り捨てられたレコードに対する孤立した索引エントリを削除できます。

孤立した索引エントリが索引パーティションに含まれているかどうかを判断するには、USER_、DBA_、ALL_PART_INDEXESデータ・ディクショナリ・ビューのORPHANED_ENTRIES列を問い合せます。詳細は、『Oracle Databaseリファレンス』を参照してください。

UPDATE BLOCK REFERENCES

UPDATE BLOCK REFERENCES句は、索引構成表の通常の索引に対してのみ有効です。この句を使用すると、2次索引パーティションに格納されている、失効したと推測されるすべてのデータ・ブロック・アドレスを更新することができます。

UPDATE BLOCK REFERENCESの制限事項

この句には、次の制限事項があります。

- ハッシュ・パーティション表の索引に対して、`physical_attributes_clause`を指定することはできません。
- ALTER INDEXの他の句とともに、UPDATE BLOCK REFERENCESを指定することはできません。

ノート:

索引がコンポジット・パーティション表のローカル索引である場合、ここで指定した変更は、以前に索引のサブパーティションに対して指定した属性を上書きします。また、そのパーティションに対して今後作成されるサブパーティションの属性値のデフォルト値となります。サブパーティションの属性はそのままで、パーティションのデフォルトの属性を変更するには、ALTER TABLE ... MODIFY DEFAULT ATTRIBUTES FOR PARTITION を使用します。

関連項目:

索引へのUnusableのマーク付け：例

UNUSABLE句

索引パーティションに対するこの句の機能は、索引全体の場合と同じです。「USABLE | UNUSABLE」を参照してください。

index_compression

この句は、コンポジット・パーティション索引で有効です。この句を使用すると、パーティションおよびそのパーティションのすべてのサブパーティションの圧縮属性を変更できます。パーティション内の各索引サブパーティションの状態は自動的にUNUSABLEと設定されるため、これらのサブパーティションを再構築する必要があります。パーティションの`prefix_compression`句を指定する前に、接頭辞圧縮を索引に指定する必要があります。または、パーティションの`advanced_index_compression`句を指定する前に、拡張索引圧縮を索引に指定する必要があります。この句は、パーティション・レベルでのみ指定できます。個々のサブパーティションの圧縮属性を変更することはできません。

非コンポジット索引パーティションにもこの句を指定できます。ただし、非コンポジット・パーティションには、再構築と圧縮属性の設定を1つのステップで行う`rebuild_clause`を使用する方が効率的です。

rename_index_partition

`rename_index_partition`句を使用すると、索引パーティションまたはサブパーティションの名前を`new_name`に変更できます。

索引パーティションの名前変更の制限事項

索引パーティションの名前変更には、次の制限事項があります。

- リスト・パーティションのサブパーティションは名前を変更できません。
- ドメイン索引のパーティションの場合、`index`に`IN_PROGRESS`または`FAILED`のマークを付けることはできません。また、パーティションに`IN_PROGRESS`、名前を変更するパーティションに`FAILED`のマークを付けることはできません。

関連項目:

索引パーティションの名前変更：例

drop_index_partition

`drop_index_partition`句を使用すると、グローバル・パーティション索引からパーティションとその中のデータを削除できます。グローバル索引のパーティションを削除する場合、その索引の次のパーティションにUNUSABLEのマークが付けられます。グローバル索引の最上位のパーティションは削除できません。

関連項目:

索引パーティションの削除：例

split_index_partition

`split_index_partition`句を使用すると、レンジ・パーティション・グローバル索引のパーティションを2つのパーティションに分割し、新しいパーティションを索引に追加できます。この句は、ハッシュ・パーティション・グローバル索引に対しては無効です。かわりに、`add_hash_index_partition`句を使用してください。

UNUSABLEのマークが付いたパーティションを分割すると、2つのパーティションが生成されますが、その両方にUNUSABLEのマークが付けられます。このようなパーティションは、使用前に再構築する必要があります。

USABLEのマークが付いたパーティションを分割すると、索引データが移入された2つのパーティションが生成されます。新しいパーティションは、どちらもUSABLEのマークが付きます。

AT句

split_partition_1に新しい上限(境界は含まない)を指定します。value_listの値は、partition_name_oldの分割前のパーティション境界より小さく、その次の最小のパーティション(そのようなパーティションがある場合)のパーティション境界より大きい値である必要があります。

INTO句

分割の結果、生成される2つのパーティションの名前と物理属性を任意に指定します。

関連項目:

パーティションの分割: 例

coalesce_index_partition

この句は、ハッシュ・パーティション・グローバル索引に対してのみ有効です。索引パーティションの数が1つ減少します。結合するパーティションは、ハッシュ・ファンクションの要件に基づいて選択されます。この句を使用するのは、選択されたパーティションの索引エントリを残りのパーティションのいずれかに分散した後で、その選択されたパーティションを削除する場合です。

modify_index_subpartition

modify_index_subpartition句を使用すると、コンポジット・パーティション表にあるローカル索引のサブパーティションに対する領域のUNUSABLEのマーク付け、割当てまたは割当て解除が可能になります。このようなサブパーティションの他のすべての属性は、パーティション・レベルのデフォルトの属性から継承されます。

例

索引ブロックの逆順格納: 例

次の文は、索引ブロックのバイトが逆順に格納されるように、索引ord_customer_ix (「索引の作成: 例」で作成)を再構築します。

```
ALTER INDEX ord_customer_ix REBUILD REVERSE;
```

索引のパラレル再構築: 例

次の文は、パラレル実行プロセスを使用して、既存の索引のスキャンおよび新しい索引の構築を行い、既存の索引から索引を再構築します。

```
ALTER INDEX ord_customer_ix REBUILD PARALLEL;
```

索引の実属性の変更: 例

次の文は、同じ索引に将来追加されるデータ・ブロックが、5つの初期トランザクション・エントリを使用するように、索引oe.cust_lname_ixを変更します。

```
ALTER INDEX oe.cust_lname_ix
    INITRANS 5;
```

索引oe.cust_lname_ixがパーティション化されている場合、この文は将来追加される索引のパーティションのデフォルト属性も変更します。将来追加されるパーティションでは、5つの初期トランザクション・エントリと100KBの増分エクステントが使用されます。

パラレル問合せの有効化: 例

次の文は、索引upper_ix (「ファンクション索引の作成: 例」で作成)に対するスキャンがパラレル化されるように、索引のパラレル属性を設定します。

```
ALTER INDEX upper_ix PARALLEL;
```

索引の名前変更: 例

次の文は、索引名を変更します。

```
ALTER INDEX upper_ix RENAME TO upper_name_ix;
```

索引へのUnusableのマーク付け: 例

次の文は、cost_ix索引(「レンジ・パーティション・グローバル索引の作成: 例」で作成)を使用します。この索引のパーティションp1は、「索引パーティションの削除: 例」で削除されています。最初の文は、索引パーティションp2にUNUSABLEのマークを付けます。

```
ALTER INDEX cost_ix
   MODIFY PARTITION p2 UNUSABLE;
```

次の文は、索引cost_ix全体にUNUSABLEのマークを付けます。

```
ALTER INDEX cost_ix UNUSABLE;
```

使用禁止の索引パーティションの再構築: 例

次の文は、cost_ix索引のパーティションp2およびp3を再構築し、索引を再度使用可能にします。パーティションp3の再構築は記録されません。

```
ALTER INDEX cost_ix
   REBUILD PARTITION p2;
ALTER INDEX cost_ix
   REBUILD PARTITION p3 NOLOGGING;
```

MAXEXTENTSの変更: 例

次の文は、パーティションp3のエクステントの最大数を変更し、ロギング属性を変更します。

```
/* This example will fail if the tablespace in which partition p3
   resides is locally managed.
*/
ALTER INDEX cost_ix MODIFY PARTITION p3
   STORAGE(MAXEXTENTS 30) LOGGING;
```

索引パーティションの名前変更: 例

次の文は、cost_ix索引(「レンジ・パーティション・グローバル索引の作成: 例」で作成)の索引パーティションの名前を変更します。

```
ALTER INDEX cost_ix
   RENAME PARTITION p3 TO p3_Q3;
```

パーティションの分割: 例

次の文は、索引cost_ixのパーティションp2 (「レンジ・パーティション・グローバル索引の作成: 例」で作成)をp2aとp2bに分割します。

```
ALTER INDEX cost_ix
   SPLIT PARTITION p2 AT (1500)
```

```
   INTO ( PARTITION p2a TABLESPACE tbs_01 LOGGING,
          PARTITION p2b TABLESPACE tbs_02);
```

索引パーティションの削除: 例

次の文は、cost_ix索引から索引パーティションp1を削除します。

```
ALTER INDEX cost_ix
   DROP PARTITION p1;
```

デフォルト属性の変更: 例

次の文は、ローカル・パーティション索引prod_idx (「ハッシュ・パーティション表の索引の作成: 例」で作成)のデフォルトの属性を変更します。将来追加されるパーティションでは、5つの初期トランザクション・エントリが使用されます。

```
ALTER INDEX prod_idx
      MODIFY DEFAULT ATTRIBUTES INITRANS 5;
```

ALTER INDEXTYPE

目的

ALTER INDEXTYPE文を使用すると、索引タイプの演算子を追加または削除したり、実装タイプを変更したり、索引タイプのプロパティを変更することができます。

前提条件

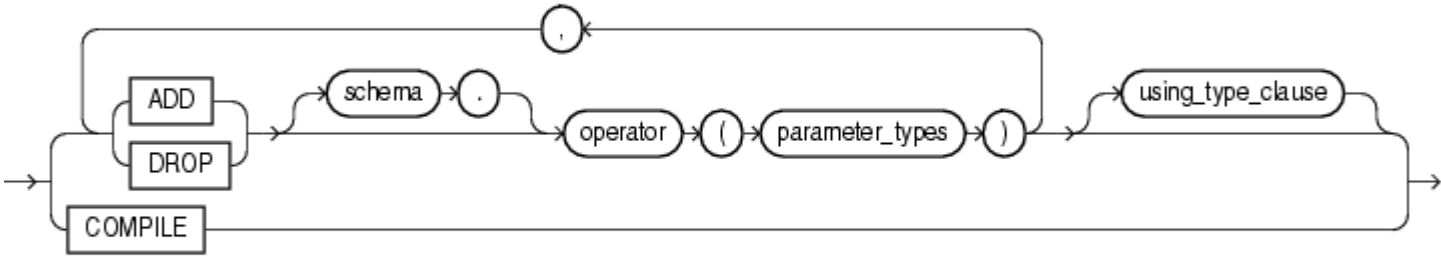
索引タイプが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY INDEXTYPEシステム権限が必要です。

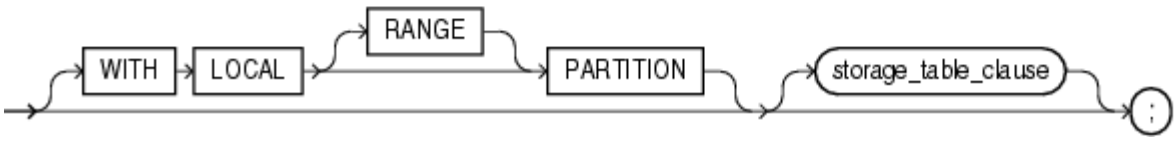
新しい演算子を追加する場合は、その演算子に対するEXECUTEオブジェクト権限が必要です。

実装タイプを変更する場合は、新しい実装タイプに対するEXECUTEオブジェクト権限が必要です。

構文

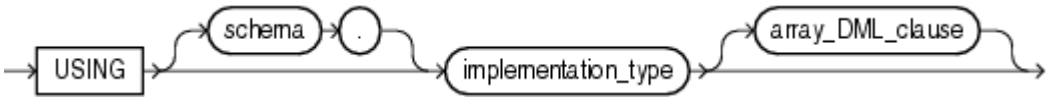
alter_indextype::=





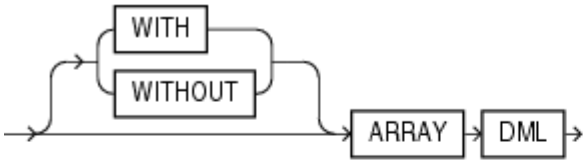
(using_type_clause::=、storage_table_clause)

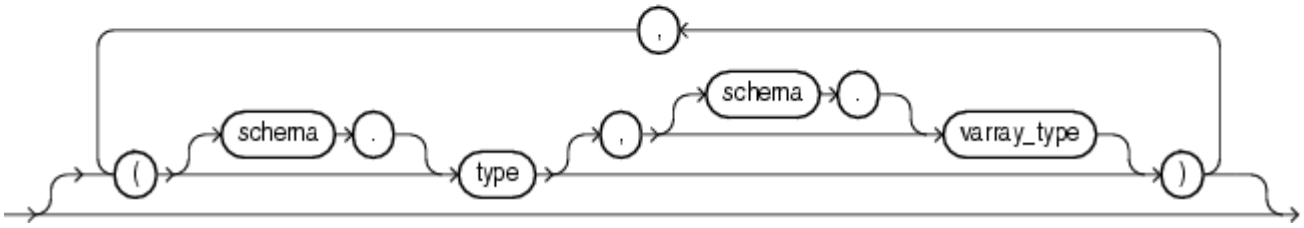
using_type_clause::=



(array_DML_clause)

array_DML_clause





storage_table_clause

セマンティクス

schema

索引タイプが存在するスキーマ名を指定します。schemaを指定しない場合、この索引タイプは自分のスキーマ内にあるとみなされます。

indextype

変更する索引タイプの名前を指定します。

ADD | DROP

ADDまたはDROP句を使用すると、演算子を追加または削除できます。

削除には特別な権限は必要ありません。

- schemaには、演算子を含むスキーマを指定します。schemaを指定しない場合、その演算子は自分のスキーマ内にあるとみなされます。
- operatorには、索引タイプによってサポートされる演算子の名前を指定します。

  この句に指定するすべての演算子は有効な演算子である必要があります。
- parameter_typeには、演算子へのパラメータ・タイプを指定します。

using_type_clause

USING句を使用すると、索引タイプを実装する新しいタイプを指定できます。

array_DML_clause

この句を使用すると、ODCIIndexInsertメソッドで配列インタフェースをサポートする索引タイプを変更できます。

typeおよびvarray_type

索引付けする列のデータ型がユーザー定義オブジェクト型である場合は、Oracleがtypeの列値の保持に使用するVARRAY varray_typeを識別するために、この句を指定する必要があります。索引タイプで型のリストがサポートされている場合、対応するVARRAY型のリストを指定できます。typeまたはvarray_typeでschemaを省略した場合、型が自分のスキーマ

内に定義されているとみなされます。

索引付けする列のデータ型が組込みシステム型である場合、その索引タイプに指定されたVARRAY型は、システムで定義されたODCI型よりも優先されます。

COMPILE

この句を使用すると、索引タイプを明示的に再コンパイルできます。通常、索引タイプは自動的に再コンパイルされるため、この句は一部のアップグレード操作の後でのみ指定する必要があります。

storage_table_clause

この句では、索引タイプを変更する場合に、索引タイプを作成するときと同じ動作になります。詳細は、「CREATE INDEXTYPE」の「storage_table_clause」を参照してください。

WITH LOCAL PARTITION

この句では、索引タイプを変更する場合に、索引タイプを作成するときと同じ動作になります。詳細は、「CREATE INDEXTYPE」の句「WITH LOCAL PARTITION」を参照してください。

例

索引タイプの変更: 例

次の例では、「索引タイプの作成: 例」で作成したposition_indextype索引タイプをコンパイルします。

```
ALTER INDEXTYPE position_indextype COMPILE;
```

ALTER INMEMORY JOIN GROUP

目的

ALTER INMEMORY JOIN GROUP文を使用すると、結合グループに対する表の列の追加または削除を行うことができます。

関連項目:

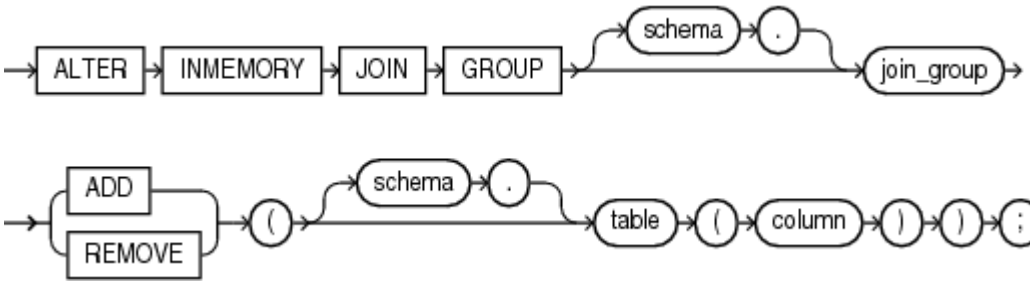
- 「CREATE INMEMORY JOIN GROUP」および「DROP INMEMORY JOIN GROUP」
- 結合グループの詳細は、『Oracle Database In-Memoryガイド』を参照してください。

前提条件

結合グループが自分のスキーマ内にない場合、または結合グループに対して追加または削除する列が自分のスキーマ内の表の列ではない場合は、ALTER ANY TABLEシステム権限が必要です。

構文

alter_inmemory_join_group::=



セマンティクス

schema

結合グループを含むスキーマを指定します。schemaを指定しない場合、結合グループは自分のスキーマ内にあるとみなされます。

join_group

変更する結合グループの名前を指定します。

DBA_JOINGROUPSまたはUSER_JOINGROUPSデータ・ディクショナリ・ビューを問い合せて、既存の結合グループを表示できます。これらのビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

ADD

ADDを指定して、結合グループに表の列を追加します。結合グループには、最大255列を含めることができます。

REMOVE

REMOVEを指定して、結合グループから表の列を削除します。結合グループには、2つ以上の列が含まれている必要があります。

schema

結合グループに追加する列または結合グループから削除する列を含む表のスキーマを指定します。schemaを指定しない場合、表は自分のスキーマ内にあるとみなされます。

table

結合グループに追加する列または結合グループから削除する列を含む表の名前を指定します。

column

結合グループに追加する列または結合グループから削除する列の名前を指定します。

例

次の例では、CREATE INMEMORY JOIN GROUPのドキュメントの「例」で作成したprod_id1結合グループに列を追加します。

```
ALTER INMEMORY JOIN GROUP prod_id1
  ADD(product_descriptions(product_id));
```

次の例では、prod_id1結合グループから列を削除します。

```
ALTER INMEMORY JOIN GROUP prod_id1
  REMOVE(product_descriptions(product_id));
```

ALTER JAVA

目的

ALTER JAVA文を使用すると、Javaクラス・スキーマ・オブジェクトの変換、またはJavaソース・スキーマ・オブジェクトのコンパイルを強制実行できます。(Java名に対するすべての外部参照を他のクラスと対応付ける前に、Javaクラスのメソッドをコールすることはできません。)

関連項目:

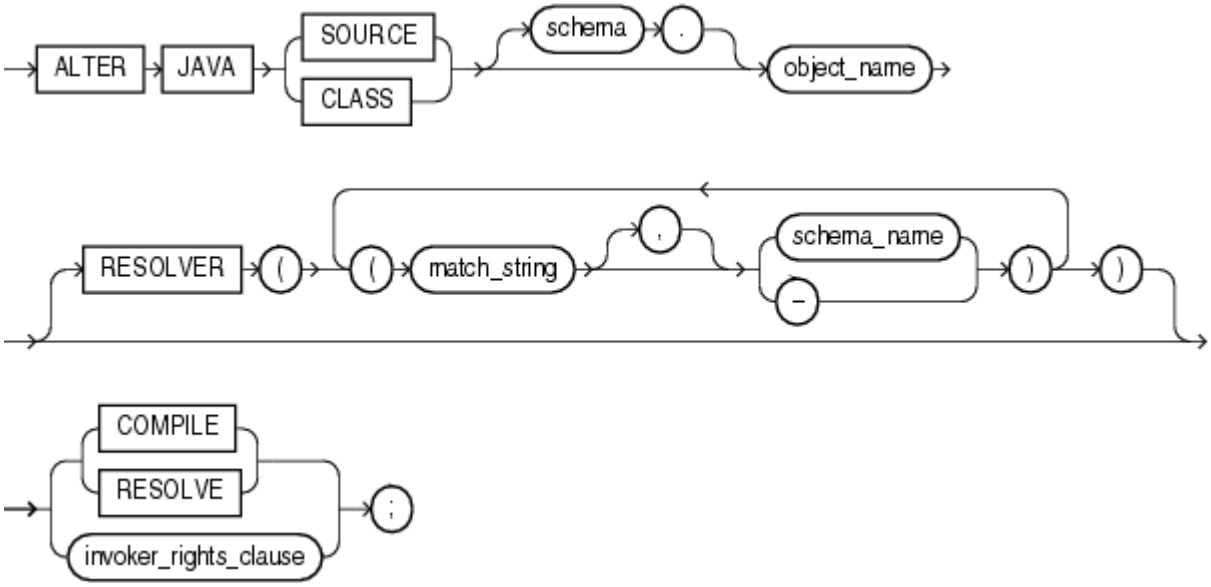
Javaクラスの変換およびJavaソースのコンパイルの詳細は、『Oracle Database Java開発者ガイド』を参照してください。

前提条件

Javaソースやクラスが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY PROCEDUREシステム権限が必要です。さらに、Javaクラスに対するEXECUTEオブジェクト権限も必要です。

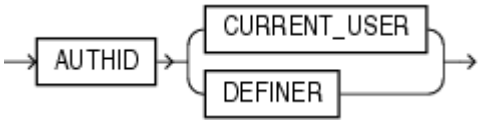
構文

alter_java::=



(invoker_rights_clause::=)

invoker_rights_clause::=



セマンティクス

JAVA SOURCE

ALTER JAVA SOURCEを使用すると、Javaソース・スキーマ・オブジェクトをコンパイルできます。

JAVA CLASS

ALTER JAVA CLASSを使用すると、Javaクラス・スキーマ・オブジェクトを変換できます。

object_name

以前作成したJavaクラスまたはソース・スキーマ・オブジェクトを指定します。小文字、または大文字と小文字を組み合せた名前を付けるには、二重引用符を使用してください。

RESOLVER

RESOLVER句を使用すると、Javaクラスまたはソースが作成されたときに指定したマッピング・ペアを使用して、完全に指定された参照用のJava名に対するスキーマの検索方法を指定できます。

関連項目:

「CREATE JAVA」および「Javaクラスの変換: 例」

RESOLVE | COMPILE

RESOLVEおよびCOMPILEは、同義のキーワードです。これらの句を使用すると、プライマリJavaクラス・スキーマ・オブジェクトの変換を指定できます。

- クラスに適用された場合、他のクラス・スキーマ・オブジェクトに対する参照名に変換されます。
- ソースに適用された場合、ソースがコンパイルされます。

invoker_rights_clause

invoker_rights_clauseを使用すると、クラスを定義したユーザーのスキーマ内で、そのユーザーの権限を使用してクラスのメソッドを実行するか、または、CURRENT_USERのスキーマ内で、そのユーザーの権限を使用してクラスのメソッドを実行するかを指定できます。

また、この句は、問合せ、DML操作、およびその型のメンバー・ファンクションおよびプロシージャ内の動的SQL文の外部名の変換方法も定義します。

AUTHID CURRENT_USER

CURRENT_USERを指定すると、クラスのメソッドがCURRENT_USER権限で実行されることを指定できます。この句はデフォルトで、実行者権限クラスを作成します。

また、この句は、問合せ、DML操作、および動的SQL文の外部名をCURRENT_USERのスキーマで変換することも指定します。他のすべての文における外部名は、メソッドを含むスキーマで変換します。

AUTHID DEFINER

DEFINERを指定すると、クラスのメソッドが、そのクラスを定義したユーザーの権限で実行されることを指定できます。

さらに、メソッドのあるスキーマ内で外部名を変換するかどうかを指定します。

関連項目:

CURRENT_USERの判断方法については、『Oracle Database PL/SQL言語リファレンス』を参照してください。

例

Javaクラスの変換: 例

次の文は、Javaクラスの変換を強制的に実行します。

```
ALTER JAVA CLASS "Agent"
   RESOLVER (("/usr/bin/bfile_dir/*" pm)(* public))
   RESOLVE;
```

11 SQL文: ALTER LIBRARYからALTER SESSION

この章では、次のSQL文について説明します。

- ALTER LIBRARY
- ALTER LOCKDOWN PROFILE
- ALTER MATERIALIZED VIEW
- ALTERMATERIALIZEDVIEWLOG
- ALTER MATERIALIZED ZONEMAP
- ALTER OPERATOR
- ALTER OUTLINE
- ALTER PACKAGE
- ALTER PLUGGABLE DATABASE
- ALTER PROCEDURE
- ALTER PROFILE
- ALTER RESOURCE COST
- ALTER ROLE
- ALTER ROLLBACK SEGMENT
- ALTER SEQUENCE
- ALTER SESSION

ALTER LIBRARY

目的

ALTER LIBRARY文を使用すると、ライブラリを明示的に再コンパイルできます。明示的に再コンパイルすることによって、実行時に暗黙的に再コンパイルする必要がなくなり、また、実行時のコンパイル・エラーとパフォーマンス上のオーバーヘッドもなくなります。

ノート:

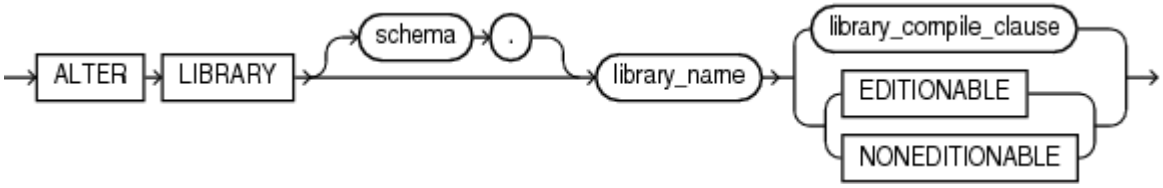
この文では、既存のライブラリの宣言または定義は変更されません。ライブラリを再宣言または再定義する場合は、CREATE LIBRARY に OR REPLACE 句を指定します。

前提条件

ライブラリがSYSスキーマ内にある場合、SYSDBAとして接続する必要があります。そうでない場合は、ライブラリが自分のスキーマ内にあるか、ALTER ANY LIBRARYシステム権限が必要です。

構文

alter_library::=



(library_compile_clause: この句の構文の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。)

セマンティクス

schema

ライブラリが含まれているスキーマを指定します。schemaを指定しない場合、プロシージャは自分のスキーマ内にあるとみなされます。

library_name

再コンパイルするライブラリの名前を指定します。

library_compile_clause

この句の構文とセマンティクスの詳細およびライブラリの作成とコンパイルの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

EDITIONABLE | NONEDITIONABLE

この句を使用すると、schemaのスキーマ・オブジェクト・タイプLIBRARYのエディションニングが後で有効化されたときに、そのライブラリをエディション・オブジェクトにするか非エディション・オブジェクトにするかを指定できます。デフォルトは、EDITIONABLEです。エディション・オブジェクトと非エディション・オブジェクトの変更の詳細は、『Oracle Database開発ガイド』を参照してください。

ALTER LOCKDOWN PROFILE

目的

ALTER LOCKDOWN PROFILE文を使用すると、PDBロックダウン・プロファイルを変更できます。マルチテナント環境でPDBロックダウン・プロファイルを使用して、プラガブル・データベース(PDB)のユーザー操作を制限できます。

CREATE LOCKDOWN PROFILE文を使用してロックダウン・プロファイルを作成した直後、そのプロファイルではすべてのユーザー操作が有効になっています。その後、ALTER LOCKDOWN PROFILE文を使用して、そのプロファイルで特定のユーザー操作を無効にできます。CDB、アプリケーション・コンテナまたはPDBにロックダウン・プロファイルが適用されている場合、ユーザーはそのプロファイルで無効になっている操作を実行できません。無効になっているユーザー操作のいずれかを再度有効にする場合は、ALTER LOCKDOWN PROFILE文を使用して有効にできます。

ALTER LOCKDOWN PROFILE文を使用すると、次の操作を無効または有効にできます。

- 特定のデータベース機能に関連付けられているユーザー操作(lockdown_features句を使用して)
- 特定のデータベース・オプションに関連付けられているユーザー操作(lockdown_options句を使用して)
- 特定のSQL文の発行(lockdown_statements句を使用して)

関連項目:

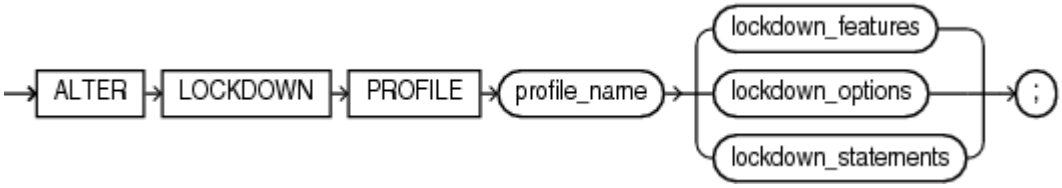
- 「CREATE LOCKDOWN PROFILE」および「DROP LOCKDOWN PROFILE」
- PDBロックダウン・プロファイルの詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

前提条件

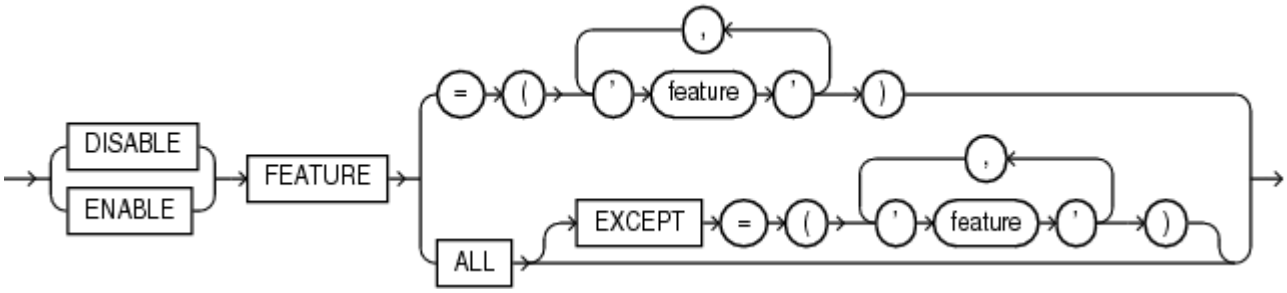
- ALTER LOCKDOWN PROFILE文は、CDBルートまたはアプリケーション・ルートから発行する必要があります。
- 文を発行するコンテナでALTER LOCKDOWN PROFILEシステム権限を持っている必要があります。

構文

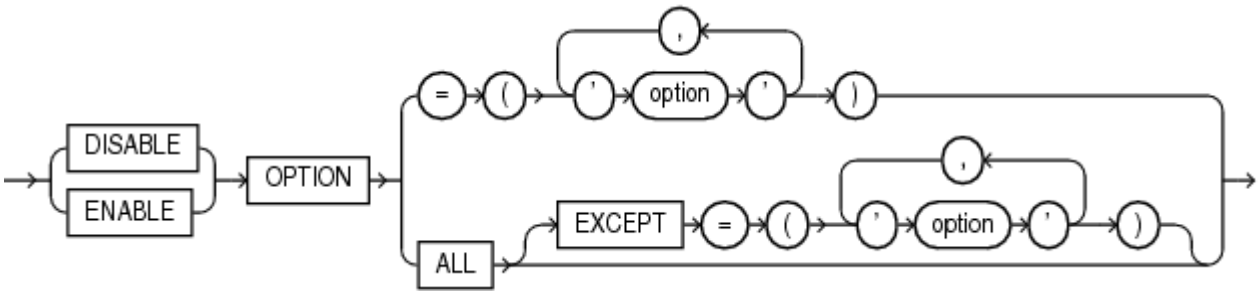
alter_lockdown_profile::=



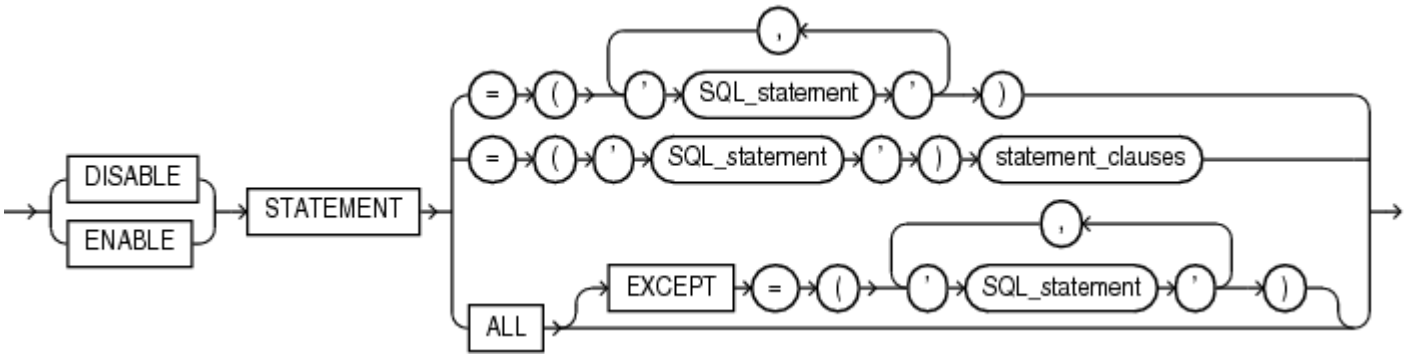
lockdown_features::=



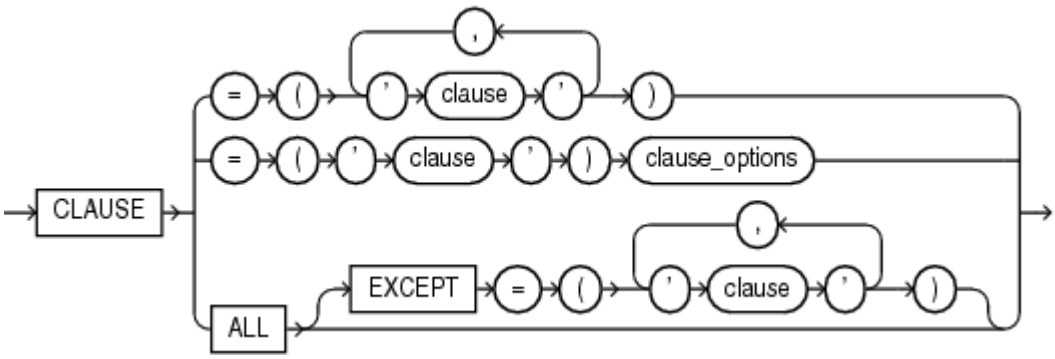
lockdown_options::=



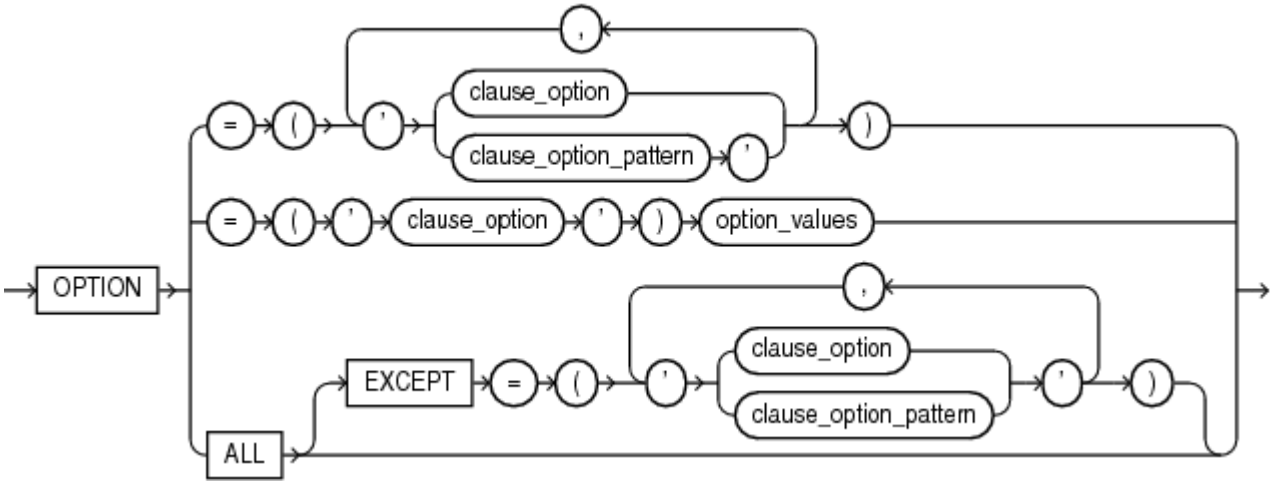
lockdown_statements::=



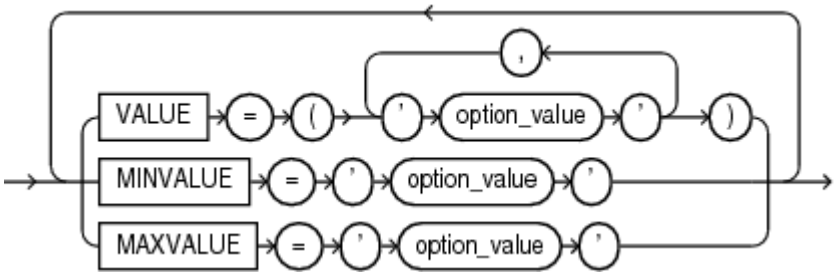
statement_clauses::=



clause_options::=



option_values::=



セマンティクス

profile_name

変更するPDBロックダウン・プロファイルの名前を指定します。

DBA_LOCKDOWN_PROFILESデータ・ディクショナリ・ビューを問い合せて、既存のPDBロックダウン・プロファイルの名前を検索できます。

lockdown_features

この句を使用すると、特定のデータベース機能に関連付けられているユーザー操作を有効または無効にできます。

- 指定した機能に対する制限を追加するには、DISABLEを指定します。ユーザーはプロファイルが適用されるPDBで、これらの操作の実行を制限されます。
- 指定した機能に対する制限を削除するには、ENABLEを指定します。ユーザーはプロファイルが適用されるPDBで、これらの操作を実行することを許可されます。
- どの機能について、操作を,無効または有効にするかを指定するには、featureを使用します。表11-1に、指定できる機能と、各機能に関連付けられている操作の説明を示します。表には、各機能の機能バンドルも示します。featureには、機能バンドル名を指定して、そのバンドル内のすべての機能のユーザー操作を無効または有効にすることも、個別の機能名を指定することもできます。機能バンドル名および機能名は、大文字と小文字を任意に組み合せて指定できます。
- ALLを使用して、表に示されているすべての機能名を指定します。
- ALL EXCEPTを使用して、指定した機能を除いて、表に示されているすべての機能名を指定します。

この句を指定しない場合、ENABLE ALLがデフォルトになります。

ノート:

- Oracle Text の FILE_DATASTORE 型は非推奨です。セキュリティを改善するために、FILE_DATASTORE 索引を DIRECTORY_DATASTORE 索引タイプに置き換えることをお薦めします。これにより、ディレクトリ・オブジェクトに基づくファイル・アクセスが可能になります。
- Oracle Text の URL_DATASTORE 型は非推奨です。URL_DATASTORE を NETWORK_DATASTORE で置き換えることをお薦めします。これは、ACL を使用して特定のサーバーへのアクセスを制御します。

表11-1 PDBロックダウン・プロファイルの機能

機能バンドル	機能	操作
AWR_ACCESS	AWR_ACCESS	PDB によって、自動ワークロード・リポジトリ(AWR)スナップショットが手動または自動で取得されること
COMMON_SCHEMA_ACCESS	COMMON_USER_LOCAL_SCHEMA_ACCESS	共通ユーザーが、実行者の権限コード・ユニットを起動するか、または PDB 内の任意のローカル・ユーザーが所有する BEQUEATH CURRENT_USER ビューにアクセスすること
COMMON_SCHEMA_ACCESS	LOCAL_USER_COMMON_SCHEMA_ACCESS	● ANY システム権限(たとえば、CREATE ANY TABLE)を持つローカル・ユーザーが、共通ユーザーのスキーマ内で、その権限が適用されるオブジェクトを作成したり、オブジェクトにアクセスすること。ノート: LOCAL_USER_COMMON_SCHEMA_ACCESS 機能を無効にしても、SYSDBA 権限または特定のオブジェクト権限を持つローカル・ユーザーが共通ユーザーのスキーマ内でオブジェクトを作成したり、オブジェクトにアクセスすることは禁止されません。そのため、そのような権限をローカル・ユーザーに付与しないことをお薦めします。 ● BECOME USER システム権限を持つローカル・ユーザーが共通ユーザーになること。 ● ローカル・ユーザーが ALTER USER 文を発行して共通ユーザーを変更すること。 ● ローカル・ユーザーがプロキシ接続に共通ユーザーを使用すること。
COMMON_SCHEMA_ACCESS	SECURITY_POLICIES	ローカル・ユーザーが、共通オブジェクトに対して、次のような特定のセキュリティ・ポリシーを作成すること。 ● データ・リダクション ● ファイングレイン監査(FGA)

機能バンドル	機能	操作
		● Real Application Security (RAS) ● Virtual Private Database(VPD)
CONNECTIONS	COMMON_USER_CONNECT	共通ユーザーが PDB に直接接続すること。この機能が無効になっている場合、PDB に接続するには、共通ユーザーはまず CDB ルートに接続してから、ALTER SESSION SET CONTAINER 文を使用して目的の PDB に切り替える必要があります。
CONNECTIONS	LOCAL_SYSOPER_RESTRICTED_MODE_CONNECT	SYSOPER 権限を持つローカル・ユーザーが、RESTRICTED モードでオープンしている PDB に接続すること
CTX_LOGGING	CTX_LOGGING	CTX_OUTPUT.START_LOG や CTX_OUTPUT.START_QUERY_LOG などの Oracle Text PL/SQL プロシージャでのロギングの使用
JAVA	JAVA	Java 全体。この機能が無効になっている場合、Java に依存するデータベースのすべてのオプションおよび機能が無効になります。
JAVA_RUNTIME	JAVA_RUNTIME	java.lang.RuntimePermission を必要とする、Java を介した操作
NETWORK_ACCESS	AQ_PROTOCOLS	HTTP、SMTP、OCI の通知を使用。
NETWORK_ACCESS	CTX_PROTOCOLS	● Oracle Text データストア型 DIRECTORY_DATASTORE および NETWORK_DATASTORE にアクセスする操作。 DIRECTORY_DATASTORE 型には DIRECTORY という属性があり、これは、索引付け対象のデータが属しているディレクトリ・オブジェクトです。この属性のデフォルト値は NULL です。

機能バンドル	機能	操作
		DIRECTORY_DATASTORE 型は、非推奨になる FILE_DATASTORE 型を置き換えます。 NETWORK_DATASTORE 型は、非推奨になる URL_DATASTORE 型を置き換えます。 NETWORK_DATASTORE 型は、アクセス制御リスト(ACL)に基づいて URL アクセスを提供する標準のデータベース・セキュリティ・モデルに準拠しており、HTTP および HTTPS プロトコルをサポートしています。 URL_DATASTORE 型は HTTPS をサポートしていませんでした。 ● イベント EVENT_INDEX_PRINT_TOKEN および EVENT_OPT_PRINT_TOKEN が、CTX ロギングの一部としてトークンを出力すること
NETWORK_ACCESS	DBMS_DEBUG_JDWP	DBMS_DEBUG_JDWP PL/SQL パッケージの使用
NETWORK_ACCESS	UTL_HTTP	UTL_HTTP PL/SQL パッケージの使用
NETWORK_ACCESS	UTL_INADDR	UTL_INADDR PL/SQL パッケージの使用
NETWORK_ACCESS	UTL_SMTP	UTL_SMTP PL/SQL パッケージの使用
NETWORK_ACCESS	UTL_TCP	UTL_TCP PL/SQL パッケージの使用
NETWORK_ACCESS	XDB_PROTOCOLS	XDB を介した HTTP、FTP およびその他のネットワーク・プロトコルの使用
OS_ACCESS	DROP_TABLESPACE_KEEP_DATAFILES	DROP TABLESPACE 文内で INCLUDING CONTENTS AND DATAFILES 句を指定しないで、PDB 内の表領域を削除すること

機能バンドル	機能	操作
OS_ACCESS	EXTERNAL_FILE_ACCESS	PDB に対して PATH_PREFIX が設定されていない場合に、PDB で外部ファイルまたはディレクトリ・オブジェクトを使用すること
OS_ACCESS	EXTERNAL_PROCEDURES	PDB で外部プロシージャ・エージェント extproc を使用すること
OS_ACCESS	FILE_TRANSFER	DBMS_FILE_TRANSFER パッケージの使用
OS_ACCESS	JAVA_OS_ACCESS	Java からの java.io.FilePermission の使用
OS_ACCESS	LOB_FILE_ACCESS	BFILE および CFILE データ型の使用
OS_ACCESS	TRACE_VIEW_ACCESS	次のトレース・ビューの使用: ● [G]V$DIAG_OPT_TRACE_RECORDS ● [G]V$DIAG_SQL_TRACE_RECORDS ● [G]V$DIAG_TRACE_FILE_CONTENTS ● V$DIAG_SESS_OPT_TRACE_RECORDS ● V$DIAG_SESS_SQL_TRACE_RECORDS
OS_ACCESS	UTL_FILE	UTL_FILE の使用。この機能が無効になっている場合、データベースで UTL_FILE.FOPEN 機能の使用がブロックされます。

lockdown_options

この句を使用すると、データベースの特定のオプションに関連付けられているユーザー操作を有効または無効にできます。

- 指定したオプションに対するユーザー操作を無効にするには、DISABLEを指定します。ユーザーはプロファイルが適用されるPDBで、これらの操作の実行を制限されます。
- 指定したオプションに対するユーザー操作を有効にするには、ENABLEを指定します。ユーザーはプロファイルが適用されるPDBで、これらの操作を実行することを許可されます。
- optionには、次のデータベース・オプションを、大文字と小文字を任意に組み合せて指定できます。

- DATABASE QUEUING – Oracle Database Advanced Queuingオプションに関連付けられているユーザー操作を表します。
- PARTITIONING – Oracle Partitioningオプションに関連付けられているユーザー操作を表します。

- ALLを使用して、前のリストに示されているすべてのオプションを指定します。
- ALL EXCEPTを使用して、指定したオプションを除いて、前のリストに示されているすべてのオプションを指定します。

この句を指定しない場合、ENABLE OPTION ALLがデフォルトになります。

lockdown_statements

この句を使用すると、特定のSQL文の発行を無効または有効にできます。

- 指定したSQL文の発行を無効にするには、DISABLEを指定します。ユーザーはプロファイルが適用されるPDBで、これらの文を発行することを制限されます。
- 指定したSQL文の発行を有効にするには、ENABLEを指定します。ユーザーはプロファイルが適用されるPDBで、これらの文を発行することを許可されます。
- SQL_statementには、次の文を、大文字と小文字を任意に組み合せて指定できます。
  - ADMINISTER KEY MANAGEMENT
  - ALTER DATABASE
  - ALTER PLUGGABLE DATABASE
  - ALTER SESSION
  - ALTER SYSTEM
  - ALTER TABLE
  - ALTER INDEX
  - ALTER TABLESPACE
  - ALTER PROFILE
  - CREATE TABLE
  - CREATE INDEX
  - CREATE TABLESPACE
  - CREATE PROFILE
  - CREATE DATABASE LINK
  - DROP TABLE
  - DROP INDEX
  - DROP TABLESPACE
  - DROP PROFILE
- ALLを使用して、前のリストに示されているすべての文を指定します。
- ALL EXCEPTを使用して、指定した文を除いて、前のリストに示されているすべての文を指定します。

この句を指定しない場合、ENABLE STATEMENT ALLがデフォルトになります。

statement_clauses

この句を使用すると、指定したSQL文の特定の句を無効または有効にできます。

- 無効または有効にする句を形成するSQLキーワードを指定するには、clauseを使用します。句は、大文字と小文字

を任意で組み合せて指定できます。

- ALLを使用して、SQL文のすべての句を指定します。
- ALL EXCEPTを使用して、指定した句を除く、SQL文のすべての句を指定します。

clauseには、SQL文の単一の句を明確に識別できるだけの十分なキーワードを指定する必要があります。次に、ALTER SYSTEM文のclauseを指定する方法の例を示します。

- archive_log_clause::=を指定するには、ARCHIVEを指定します。キーワードARCHIVEで始まるALTER SYSTEM句は他にないため、これで十分です。意味を明確にするためにARCHIVE LOGを指定することもできますが、LOGキーワードは不要です。
- いずれかのrolling_migration_clauses::=を指定する場合は、これらの句を同じような名前のrolling_patch_clauses::= START ROLLING PATCHおよびSTOP ROLLING PATCHと区別するために、START ROLLING MIGRATIONまたはSTOP ROLLING MIGRATIONを指定する必要があります。
- 単一のキーワードFLUSHを指定することはできません。複数のALTER SYSTEM句がこのキーワードで始まるためです。かわりに、FLUSH SHARED_POOLまたはFLUSH GLOBAL CONTEXTなど、各句を別々に指定する必要があります。

何の効果もないため、句の中でオプションのキーワードを指定する必要はありません。たとえば:

- archive_log_clause::=には、オプションのINSTANCEキーワードがあります。しかし、INSTANCEキーワードを含むARCHIVE LOG句のみを有効または無効にすることはできません。ARCHIVE LOG INSTANCEを指定することは、ARCHIVEまたはARCHIVE LOGを指定することと同じです。

何の効果もないため、句の中でパラメータ値を指定する必要はありません。たとえば:

- shutdown_dispatcher_clause::=ではdispatcher_nameを指定する必要があります。しかし、特定のディスパッチャ名を含むSHUTDOWN句を有効または無効にすることはできません。SHUTDOWN dispatcher1を指定することは、SHUTDOWNを指定することと同じです。

関連項目:

これらの文の句の詳細は、「ALTER DATABASE」、「ALTER PLUGGABLE DATABASE」、「ALTER SESSION」および「ALTER SYSTEM」を参照してください。

clause_options

この句は、lockdown_statementsおよびstatement_clausesに対して、次のいずれかを指定している場合にのみ有効です。

```
{ DISABLE | ENABLE } STATEMENT = ('ALTER SESSION') CLAUSE = ('SET')
{ DISABLE | ENABLE } STATEMENT = ('ALTER SYSTEM') CLAUSE = ('SET')
```

この句を使用すると、ALTER SESSION SETまたはALTER SYSTEM SET文の特定のオプションの設定または変更を無効または有効にできます。

- 無効または有効にするオプションを指定するには、clause_optionを使用します。
- 複数のオプションに一致するパターンを指定するには、clause_option_patternを使用します。パターン内でパーセント記号(%)を指定すると、オプション名に含まれるゼロ個以上の文字に一致させることができます。たとえば、'QUERY_REWRITE_%'を指定することは、QUERY_REWRITE_ENABLEDおよび

QUERY_REWRITE_INTEGRITYの両方のオプションを指定することと同等です。

- clause_optionおよびclause_option_patternは、大文字と小文字を任意に組み合せて指定できます。
- ALLを使用して、すべてのオプションを指定します。
- ALL EXCEPTを使用して、指定したオプションを除くすべてのオプションを指定します。

関連項目:

これらの文で指定できるオプションの詳細は、「ALTER SESSION」の「alter_session_set_clause」句および「ALTER SYSTEM」の「alter_system_set_clause」句を参照してください。

option_values

この句は、lockdown_statements、statement_clausesおよびclause_optionsに対して、次のいずれかを指定している場合にのみ有効です。

```
DISABLE STATEMENT = ('ALTER SESSION') CLAUSE = ('SET') OPTION = clause_option
DISABLE STATEMENT = ('ALTER SYSTEM') CLAUSE = ('SET') OPTION = clause_option
```

この句を使用すると、オプションの設定を無効にした場合の、そのオプションのデフォルト値を指定できます。数値を取得するオプションの場合は、この句を使用して、ユーザーがオプションに特定の値を設定しないように制限することもできます。

- VALUE句を使用すると、clause_optionのデフォルトのoption_valueを指定でき、これは、PDBをクローズして再オープンした後、プロファイルが適用されるPDBで有効になります。clause_optionが複数のデフォルト値を受け入れる場合は、複数のoption_valueをカンマ区切りリストで指定できます。この句を使用する目的は、オプションのデフォルト値を設定すると同時に、ユーザーによる値の設定または変更を制限することです。
- MINVALUE句を使用すると、ユーザーがclause_optionの値をoption_value未満の値に設定しないように制限できます。この句は、数値を取得するオプションに対してのみ指定できます。
- MAXVALUE句を使用すると、ユーザーがclause_optionの値をoption_valueよりも大きい値に設定しないように制限できます。この句は、数値を取得するオプションに対してのみ指定できます。
- MINVALUE句とMAXVALUE句の両方を同時に指定すると、ユーザーがclause_optionsの値をMINVALUE未満の値にも、MAXVALUEよりも大きい値にも設定しないように制限できます。
- MINVALUEおよびMAXVALUE設定は、ロックダウン・プロファイルがPDBに割り当てられるとすぐに有効になります。PDBをクローズして再オープンする必要はありません。

関連項目:

様々なオプションで許可される値の詳細は、『Oracle Databaseリファレンス』を参照してください。

例

次の文は、PDBロックダウン・プロファイルhr_profを作成します。

```
CREATE LOCKDOWN PROFILE hr_prof;
```

この項の残りの例では、hr_profを変更します。

PDBロックダウン・プロファイルの機能の無効化: 例

次の文は、機能バンドルNETWORK_ACCESSのすべての機能を無効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  DISABLE FEATURE = ('NETWORK_ACCESS');
```

次の文は、LOB_FILE_ACCESSおよびTRACE_VIEW ACCESS機能を無効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  DISABLE FEATURE = ('LOB_FILE_ACCESS', 'TRACE_VIEW_ACCESS');
```

次の文は、COMMON_USER_LOCAL_SCHEMA_ACCESSおよびLOCAL_USER_COMMON_SCHEMA_ACCESS機能を除くすべての機能を無効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  DISABLE FEATURE ALL EXCEPT = ('COMMON_USER_LOCAL_SCHEMA_ACCESS',
'LOCAL_USER_COMMON_SCHEMA_ACCESS');
```

次の文は、すべての機能を無効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  DISABLE FEATURE ALL;
```

PDBロックダウン・プロファイルの機能の有効化: 例

次の文は、UTL_HTTPおよびUTL_SMTP機能、および機能バンドルOS_ACCESSのすべての機能を有効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  ENABLE FEATURE = ('UTL_HTTP', 'UTL_SMTP', 'OS_ACCESS');
```

次の文は、AQ_PROTOCOLSおよびCTX_PROTOCOLS機能を除くすべての機能を有効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  ENABLE FEATURE ALL EXCEPT = ('AQ_PROTOCOLS', 'CTX_PROTOCOLS');
```

次の文は、すべての機能を有効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  ENABLE FEATURE ALL;
```

PDBロックダウン・プロファイルのオプションの無効化: 例

次の文は、Oracle Database Advanced Queuingオプションに関連付けられているユーザー操作を無効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  DISABLE OPTION = ('DATABASE QUEUING');
```

次の文は、Oracle Partitioningオプションに関連付けられているユーザー操作を無効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  DISABLE OPTION = ('PARTITIONING');
```

PDBロックダウン・プロファイルのオプションの有効化: 例

次の文は、Oracle Database Advanced Queuingオプションに関連付けられているユーザー操作を有効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  ENABLE OPTION = ('DATABASE QUEUING');
```

次の文は、Oracle Database Advanced QueuingオプションおよびOracle Partitioningオプションの両方に関連付けられているユーザー操作を有効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
```

```
  ENABLE OPTION ALL;
```

PDBロックダウン・プロファイルのSQL文の無効化: 例

次の文は、ALTER DATABASE文を無効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  DISABLE STATEMENT = ('ALTER DATABASE');
```

次の文は、ALTER SYSTEM SUSPENDおよびALTER SYSTEM RESUME文を無効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  DISABLE STATEMENT = ('ALTER SYSTEM')
          CLAUSE = ('SUSPEND', 'RESUME');
```

次の文は、DEFAULT TABLESPACEおよびDEFAULT TEMPORARY TABLESPACEを除く、ALTER PLUGGABLE DATABASE文のすべての句を無効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  DISABLE STATEMENT = ('ALTER PLUGGABLE DATABASE')
          CLAUSE ALL EXCEPT = ('DEFAULT TABLESPACE', 'DEFAULT TEMPORARY TABLESPACE');
```

次の文は、ALTER SESSION文を使用した、COMMIT_WAITまたはCURSOR_SHARINGの設定または変更を無効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  DISABLE STATEMENT = ('ALTER SESSION')
          CLAUSE = ('SET')
          OPTION = ('COMMIT_WAIT', 'CURSOR_SHARING');
```

次の文は、ALTER SYSTEM文を使用した、PDB_FILE_NAME_CONVERTの値の設定または変更を無効にします。また、PDB_FILE_NAME_CONVERTのデフォルト値を'cdb1_pdb0', 'cdb1_pdb1'に設定します。このデフォルト値は、次回PDBをクローズして再オープンしたときに有効になります。

```
ALTER LOCKDOWN PROFILE hr_prof
  DISABLE STATEMENT = ('ALTER SYSTEM')
          CLAUSE = ('SET')
          OPTION = ('PDB_FILE_NAME_CONVERT')
          VALUE = ('cdb1_pdb0', 'cdb1_pdb1');
```

次の文は、ALTER SYSTEM文を使用した、CPU_COUNTの値の8未満の値への設定または変更を無効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  DISABLE STATEMENT = ('ALTER SYSTEM')
          CLAUSE = ('SET')
          OPTION = ('CPU_COUNT')
          MINVALUE = '8';
```

次の文は、ALTER SYSTEM文を使用した、CPU_COUNTの値の2より大きい値への設定または変更を無効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  DISABLE STATEMENT = ('ALTER SYSTEM')
          CLAUSE = ('SET')
          OPTION = ('CPU_COUNT')
          MAXVALUE = '2';
```

次の文は、ALTER SYSTEM文を使用した、CPU_COUNTの値の2未満の値または6より大きい値への設定または変更を無効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  DISABLE STATEMENT = ('ALTER SYSTEM')
```

```
          CLAUSE = ('SET')
          OPTION = ('CPU_COUNT')
          MINVALUE = '2'
          MAXVALUE = '6';
```

PDBロックダウン・プロファイルのSQL文の有効化: 例

次の文は、ALTER DATABASEを除くすべての文を有効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  ENABLE STATEMENT ALL EXCEPT = ('ALTER DATABASE');
```

次の文は、ALTER DATABASE MOUNTおよびALTER DATABASE OPEN文を有効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  ENABLE STATEMENT = ('ALTER DATABASE')
           CLAUSE = ('MOUNT', 'OPEN');
```

次の文は、DEFAULT TABLESPACEおよびDEFAULT TEMPORARY TABLESPACEを除く、ALTER PLUGGABLE DATABASE文のすべての句を有効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  ENABLE STATEMENT = ('ALTER PLUGGABLE DATABASE')
          CLAUSE ALL EXCEPT = ('DEFAULT TABLESPACE', 'DEFAULT TEMPORARY TABLESPACE');
```

次の文は、ALTER SESSION文を使用した、COMMIT_WAITまたはCURSOR_SHARINGの設定または変更を有効にします。

```
ALTER LOCKDOWN PROFILE hr_prof
  ENABLE STATEMENT = ('ALTER SESSION')
          CLAUSE = ('SET')
          OPTION = ('COMMIT_WAIT', 'CURSOR_SHARING');
```

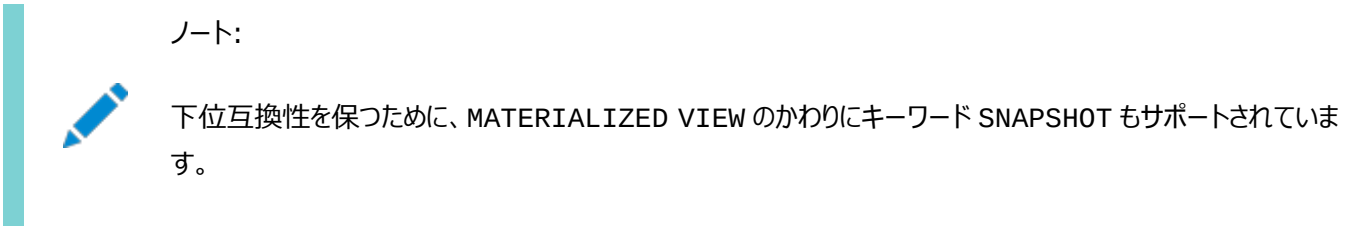
ALTER MATERIALIZED VIEW

目的

マテリアライズド・ビューは、問合せ結果を含むデータベース・オブジェクトです。問合せのFROM句には、表、ビューおよびその他のマテリアライズド・ビューを指定できます。これらをあわせて、マスター表(レプリケーション用語)またはディテール表(データ・ウェアハウス用語)といいます。このマニュアルでは、マスター表という用語を使用します。マスター表が格納されているデータベースをマスター・データベースといいます。

ALTER MATERIALIZED VIEW文を使用すると、既存のマテリアライズド・ビューを次の方法で変更できます。

- 記憶特性を変更します。
- リフレッシュ方法、モードまたは時間を変更します。
- 別のタイプのマテリアライズド・ビューになるように構造を変更します。
- クエリー・リライトを使用可能または使用禁止にします。



ノート:

下位互換性を保つために、MATERIALIZED VIEW のかわりにキーワード SNAPSHOT もサポートされています。

関連項目:

- マテリアライズド・ビューの作成の詳細は、「CREATE MATERIALIZED VIEW」を参照してください。
- レプリケーション環境でのマテリアライズド・ビューの詳細は、『Oracle Database管理者ガイド』を参照してください。
- データ・ウェアハウス環境でのマテリアライズド・ビューの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

前提条件

マテリアライズド・ビューが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY MATERIALIZED VIEWシステム権限が必要です。

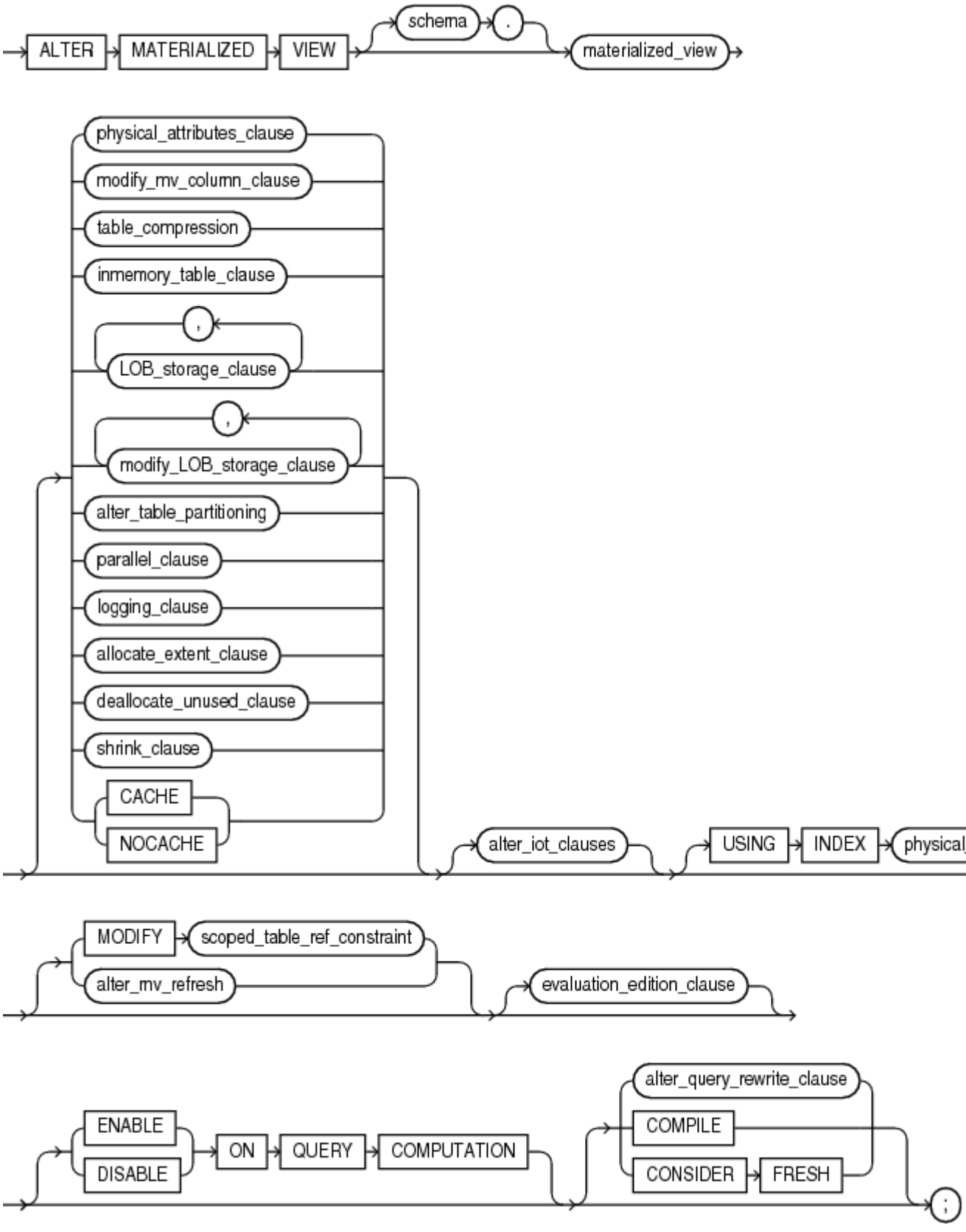
クエリー・リライトでマテリアライズド・ビューを使用可能にするには:

- マテリアライズド・ビュー内のすべてのマスター表が自分のスキーマ内にある場合、QUERY REWRITE権限が必要です。
- いずれかのマスター表が別のスキーマ内にある場合、GLOBAL QUERY REWRITE権限が必要です。
- マテリアライズド・ビューが別のユーザーのスキーマ内にある場合、ユーザーおよびそのスキーマ所有者の両方に、前述の適切なQUERY REWRITE権限が必要です。また、マテリアライズド・ビューの所有者は、マテリアライズド・ビュー所有者が所有しないすべてのマスター表へのSELECT権限を持っている必要があります。

evaluation_edition_clauseまたはunusable_editions_clauseでエディションを指定するには、そのエディションに対するUSE権限が必要になります。

構文

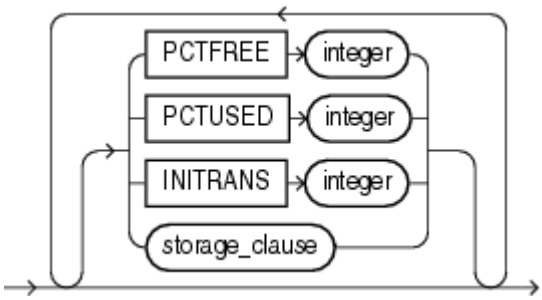
alter_materialized_view::=



(physical_attributes_clause::=、modify_mv_column_clause::=、table_compression::=、inmemory_table_clause::=、LOB_storage_clause::=、modify_LOB_storage_clause::=、alter_table_partitioning::= (ALTER TABLEの一部)、parallel_clause::=、logging_clause::=、allocate_extent_clause::=、deallocate_unused_clause::=、shrink_clause::=、alter_iot_clauses::=、scoped_table_ref_constraint::=、alter_mv_refresh::=、evaluation_edition_clause::=、

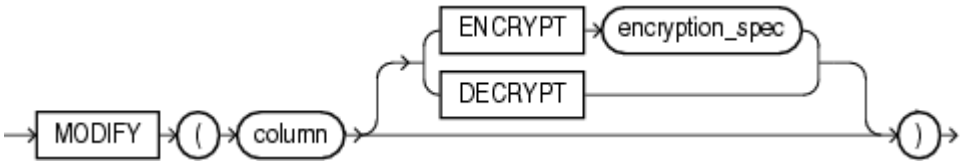
alter_query_rewrite_clause::=)

physical_attributes_clause::=

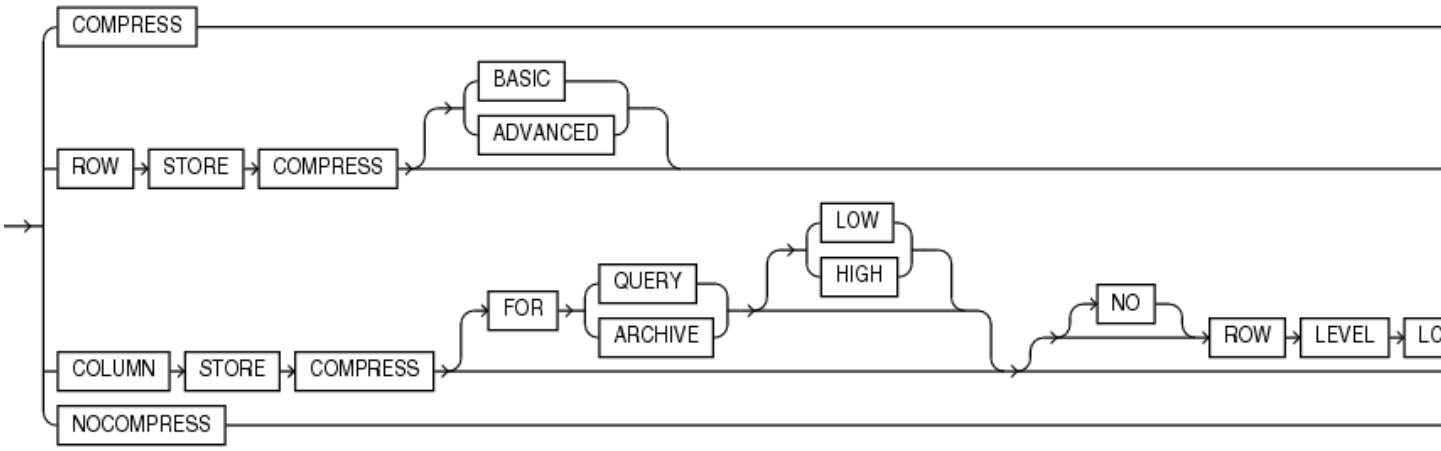


(storage_clause::=)

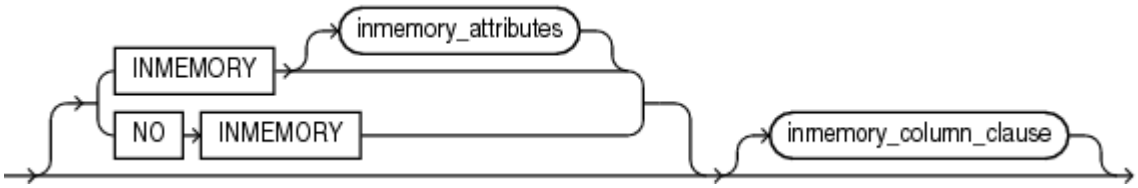
modify_mv_column_clause::=



table_compression::=



inmemory_table_clause::=



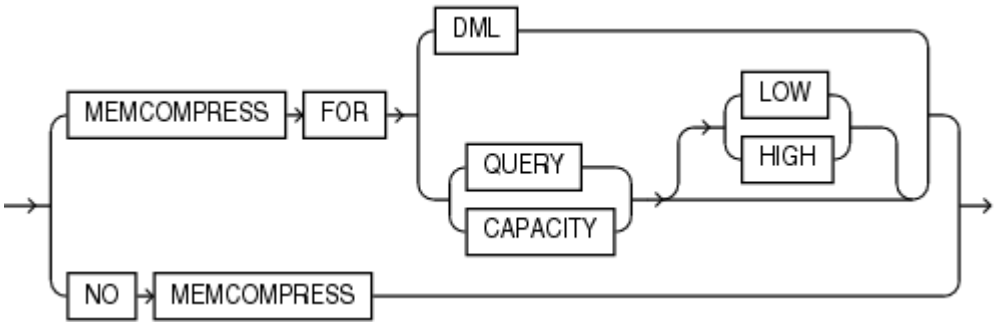
(inmemory_attributes::=、inmemory_column_clause::=)

inmemory_attributes::=

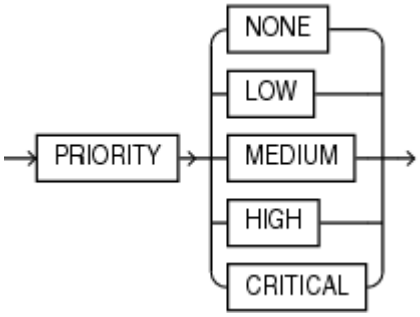
(inmemory_memcompress::=、inmemory_priority::=、inmemory_distribute::=、

inmemory_duplicate::=)

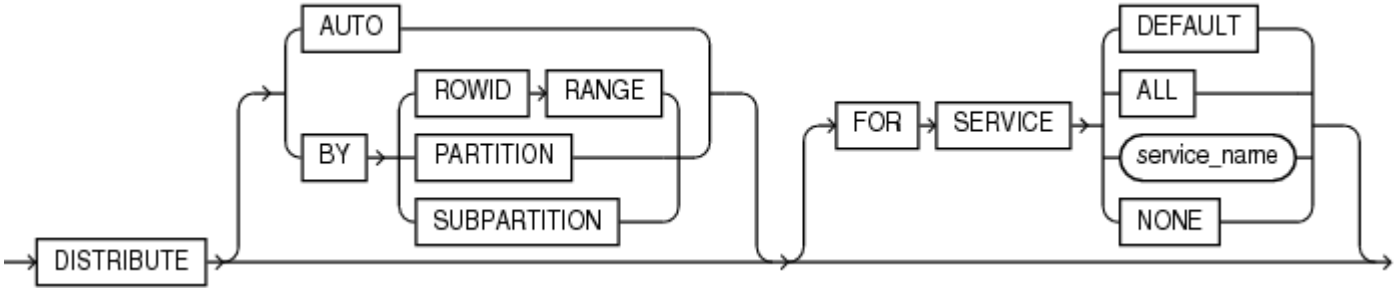
inmemory_memcompress::=



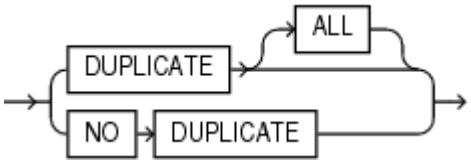
inmemory_priority::=



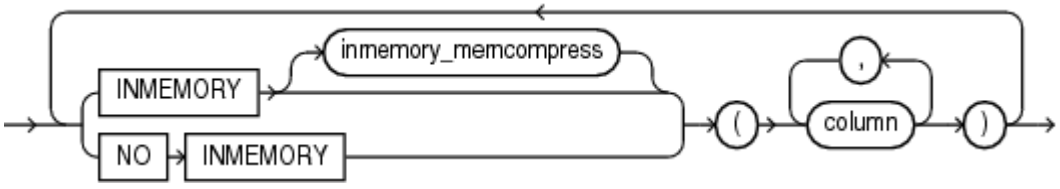
inmemory_distribute::=



inmemory_duplicate::=

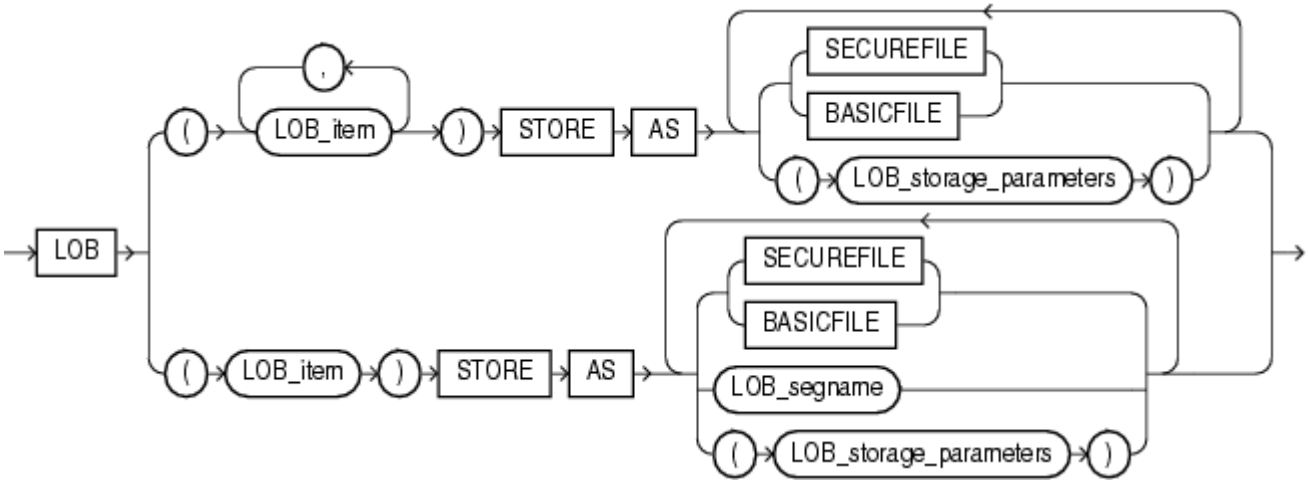


inmemory_column_clause::=



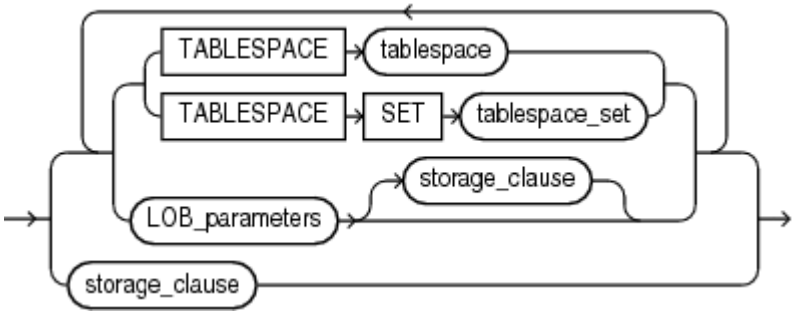
(inmemory_memcompress::=)

LOB_storage_clause::=



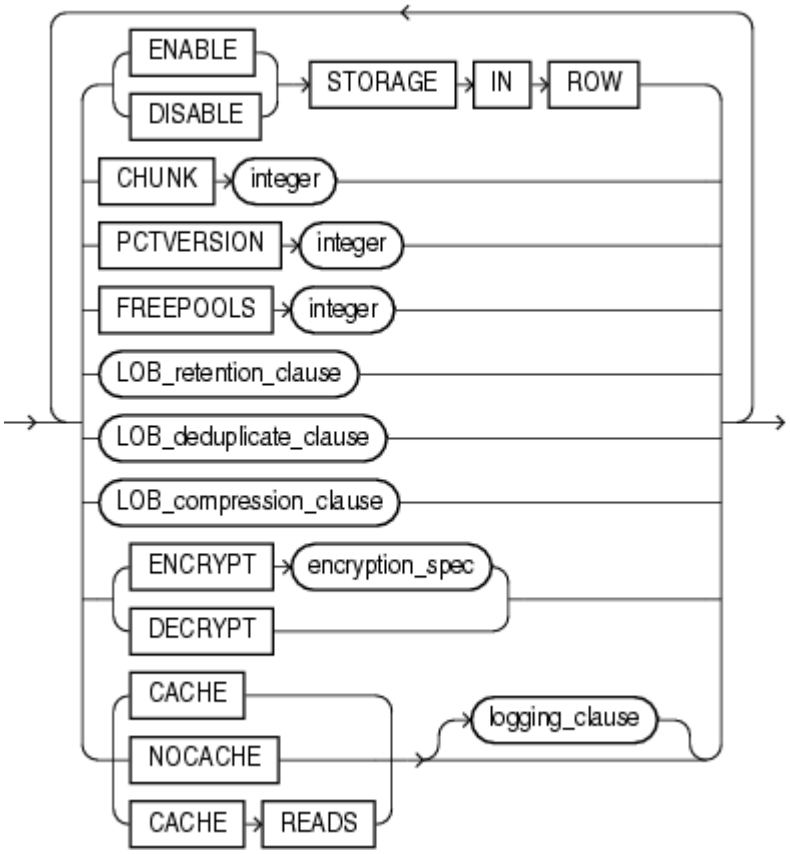
(LOB_storage_parameters::=)

LOB_storage_parameters::=



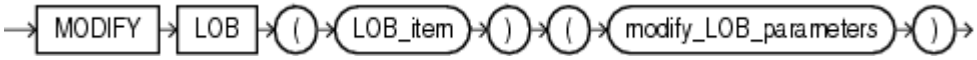
(TABLESPACE SET: ALTER MATERIALIZED VIEWではサポートされていません、LOB_parameters::=、storage_clause::=)

LOB_parameters::=



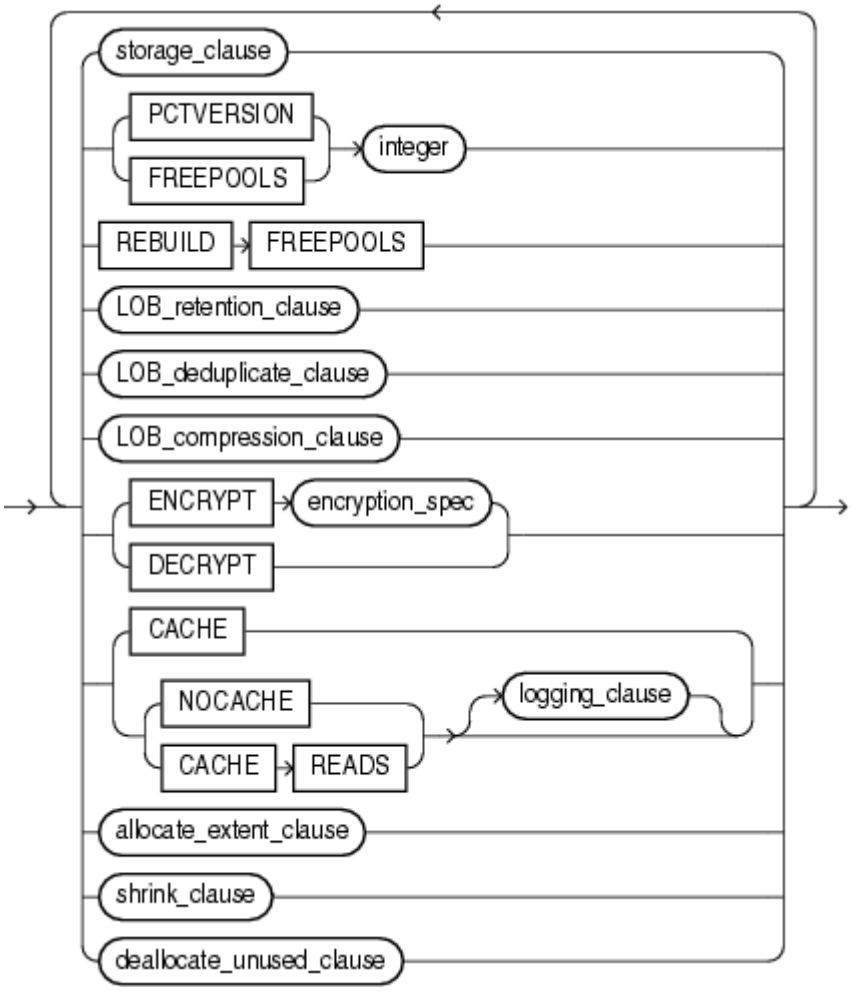
(storage_clause::=、logging_clause::=)

modify_LOB_storage_clause::=



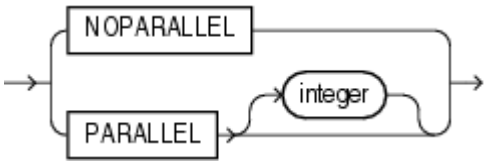
(modify_LOB_parameters::=)

modify_LOB_parameters::=

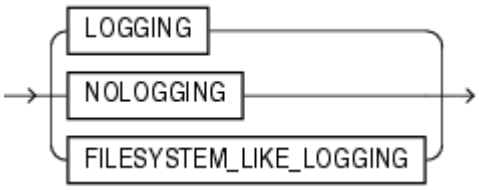


(storage_clause::=、LOB_retention_clause::=、LOB_compression_clause::=、logging_clause::=、allocate_extent_clause::=、shrink_clause::=、deallocate_unused_clause::=)

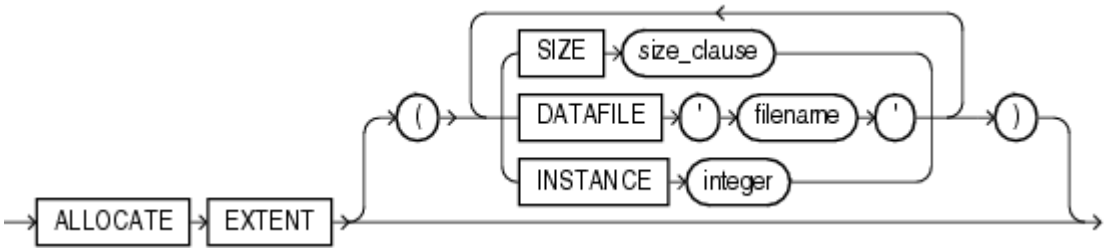
parallel_clause::=



logging_clause::=

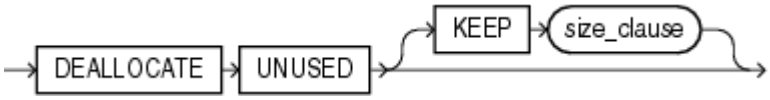


allocate_extent_clause::=



(size_clause::=)

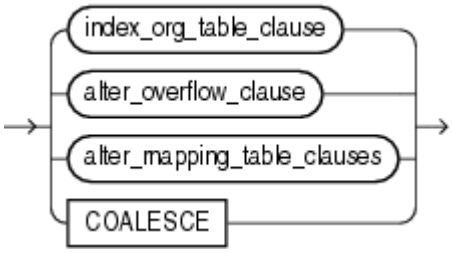
deallocate_unused_clause::=



(size_clause::=)

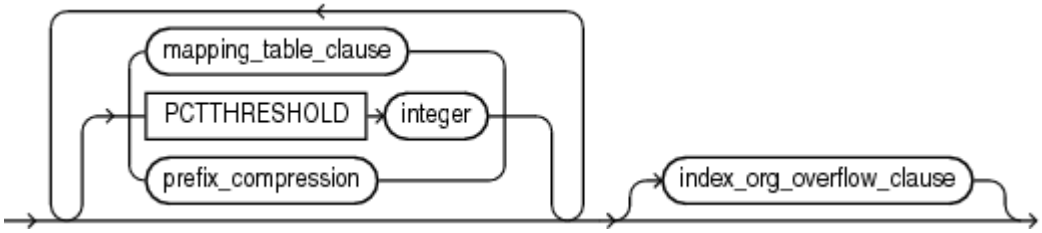
shrink_clause::=

alter_iot_clauses::=



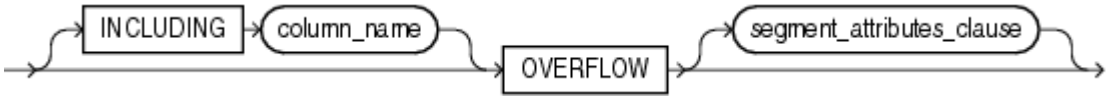
(index_org_table_clause::=、alter_overflow_clause::=、`alter_mapping_table_clauses`: マテリアライズド・ビューではサポートされていません)

index_org_table_clause::=



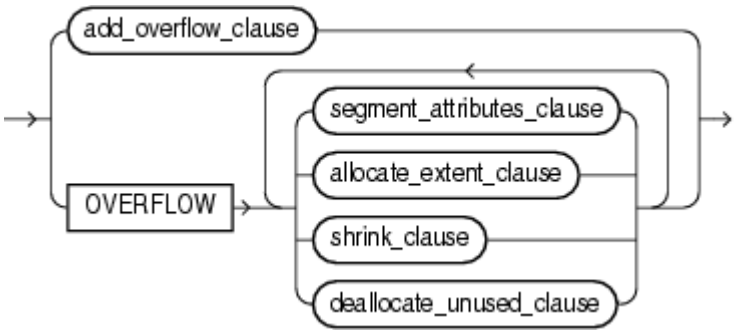
(`mapping_table_clause`: マテリアライズド・ビューではサポートされていません、`prefix_compression`: マテリアライズド・ビューの変更ではサポートされていません、index_org_overflow_clause::=)

index_org_overflow_clause::=



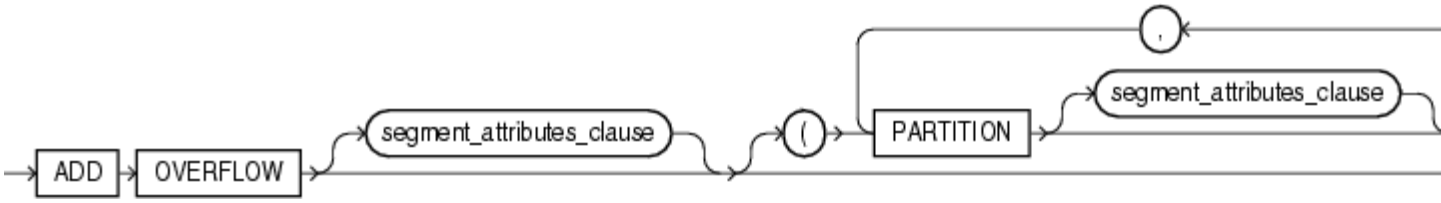
(segment_attributes_clause::= (ALTER TABLEの一部))

alter_overflow_clause::=



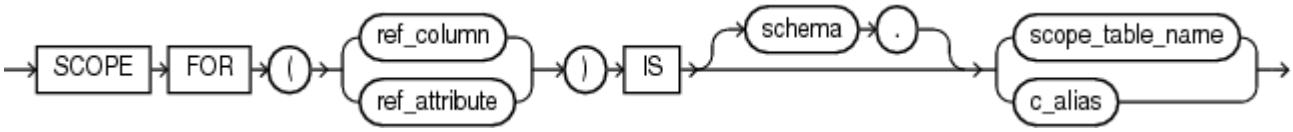
(allocate_extent_clause::=、shrink_clause::=、deallocate_unused_clause::=)

add_overflow_clause::=

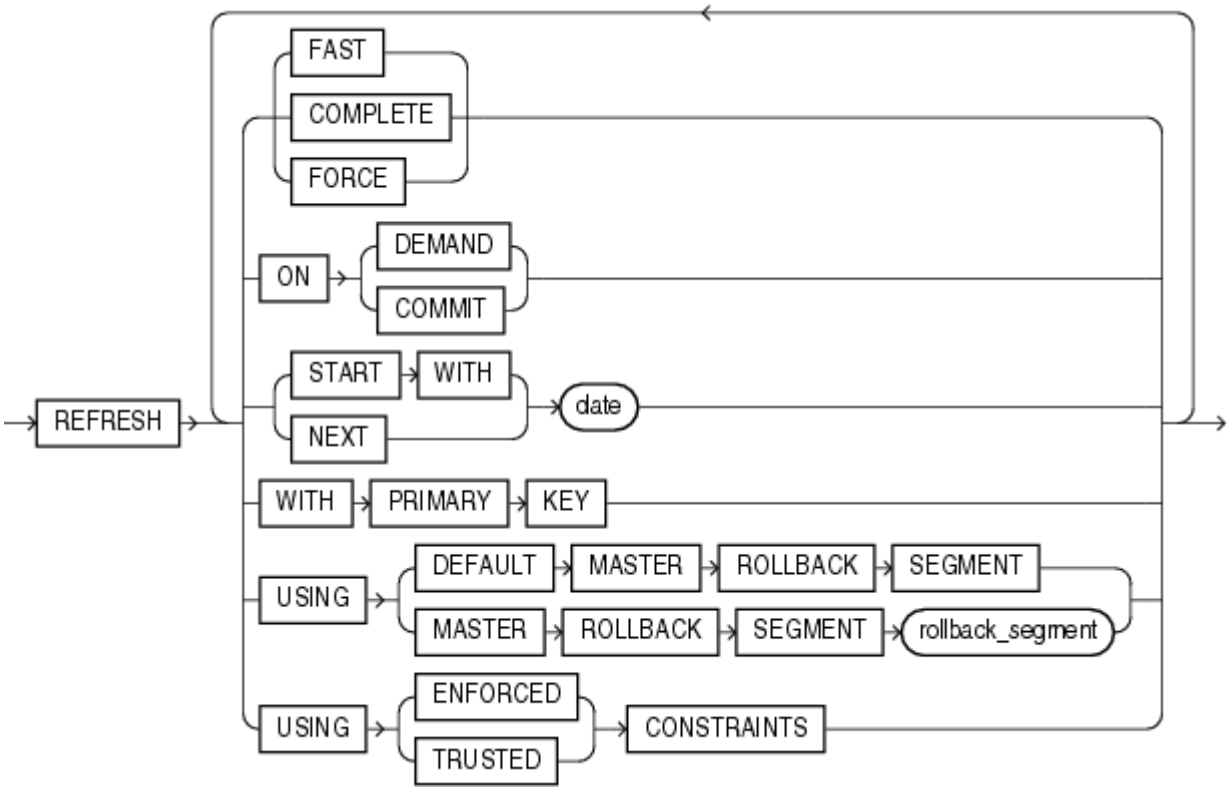


(segment_attributes_clause::= (ALTER TABLEの一部))

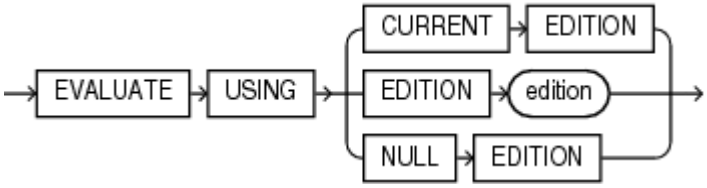
scoped_table_ref_constraint::=



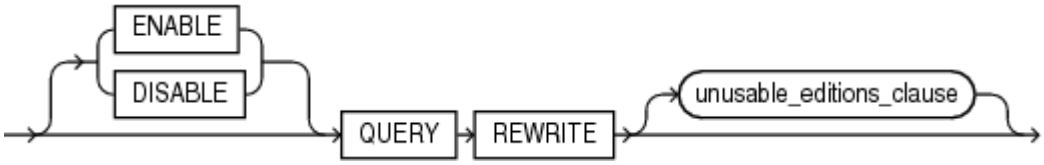
alter_mv_refresh::=



evaluation_edition_clause::=



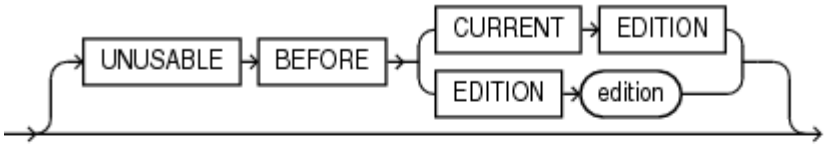
alter_query_rewrite_clause::=

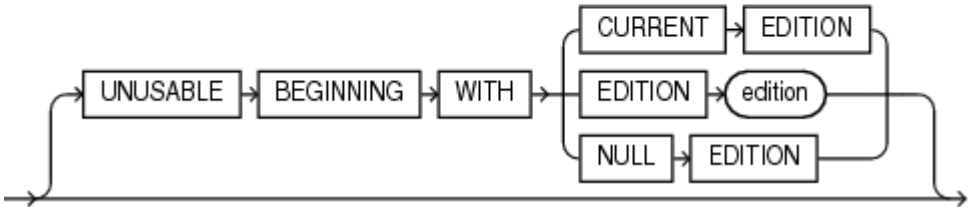


ノート:

`QUERY REWRITE` のみを指定することはできません。`ENABLE` または `DISABLE` のどちらかを指定するか、副次句の `unusable_editions_clause` を指定する必要があります。

unusable_editions_clause::=





セマンティクス

schema

マテリアライズド・ビューが含まれているスキーマを指定します。schemaを指定しない場合、このマテリアライズド・ビューは自分のスキーマ内にあるとみなされます。

materialized_view

変更するマテリアライズド・ビューの名前を指定します。

physical_attributes_clause

PCTFREE、PCTUSED、INITRANSパラメータの値(USING INDEX句で使用する場合は、INITRANSパラメータ値のみ)、およびマテリアライズド・ビューの記憶特性を指定します。PCTFREE、PCTUSEDおよびINITRANSパラメータの詳細は「ALTER TABLE」、記憶特性の詳細は「storage_clause」を参照してください。

modify_mv_column_clause

この句を使用すると、マテリアライズド・ビューのこの列を暗号化または復号化できます。この句の詳細は、「CREATE TABLE」の句「encryption_spec」を参照してください。

table_compression

table_compression句を使用すると、ディスクおよびメモリーの使用量を削減するために、データ・セグメントを圧縮するかどうかを指定できます。この句のセマンティクスの詳細は、「CREATE TABLE」の句「table_compression」を参照してください。

inmemory_table_clause

inmemory_table_clauseを使用すると、インメモリー列ストア(IM列ストア)のマテリアライズド・ビューまたはその列を有効または無効にするか、マテリアライズド・ビューまたはその列のインメモリー属性を変更できます。この句のセマンティクスは、ALTER TABLE文のものと同じです。この句のセマンティクスの詳細は、「ALTER TABLE」の「inmemory_table_clause」を参照してください。

LOB_storage_clause

LOB_storage_clauseを使用すると、新しいLOBの記憶特性を指定できます。マテリアライズド・ビューのLOB記憶域は、表の場合と同様に動作します。LOB記憶域パラメータの詳細は、「CREATE TABLE」の「LOB_storage_clause」 を参照してください。

modify_LOB_storage_clause

modify_LOB_storage_clauseを使用すると、LOB属性LOB_itemの物理属性またはLOBオブジェクト属性を変更できます。マテリアライズド・ビューのLOB記憶域の変更は、表の場合と同様に動作します。

関連項目:

変更可能なLOB記憶域パラメータの詳細は、「ALTER TABLE」の「modify_LOB_storage_clause」を参照してください。

alter_table_partitioning

マテリアライズド・ビューのパーティション化の句の構文および一般的な機能は、パーティション表と同じです。「ALTER TABLE」の「alter_table_partitioning」を参照してください。

マテリアライズド・ビュー・パーティションの変更の制限事項

いずれのpartitioning_clauses内でも、LOB_storage_clauseおよびmodify_LOB_storage_clauseは指定できません。

ノート:

マテリアライズド・ビューの内容をマスター表の内容と同期させて保持するには、表パーティションを削除または切り捨てた後、表に依存しているすべてのマテリアライズド・ビューを手動で完全リフレッシュすることをお薦めします。

MODIFY PARTITION UNUSABLE LOCAL INDEXES

この句を使用すると、partitionに関連付けられたすべてのローカル索引パーティションに、UNUSABLEのマークが付きます。

MODIFY PARTITION REBUILD UNUSABLE LOCAL INDEXES

この句を使用すると、partitionに関連付けられた、使用禁止のローカル索引パーティションを再構築できます。

parallel_clause

parallel_clauseを使用すると、マテリアライズド・ビューのデフォルトの並列度を変更できます。

この句の詳細は、「CREATE TABLE」の「parallel_clause」を参照してください。

logging_clause

この句を使用すると、マテリアライズド・ビューのロギング特性を指定または変更できます。この句の詳細は、「logging_clause」を参照してください。

allocate_extent_clause

allocate_extent_clauseを使用すると、マテリアライズド・ビューの新しいエクステントを明示的に割り当てることができます。この句の詳細は、「allocate_extent_clause」を参照してください。

deallocate_unused_clause

deallocate_unused_clause句を使用すると、マテリアライズド・ビューの末尾にある未使用領域の割当てを明示的に解除でき、解放された領域を他のセグメントに使用できるようになります。この句の詳細は、「deallocate_unused_clause」を参照してください。

shrink_clause

この句を使用すると、マテリアライズド・ビューのセグメントを縮小化できます。この句の詳細は、「CREATE TABLE」の「shrink_clause」を参照してください。

CACHE | NOCACHE

アクセス頻度の高いデータについて、CACHEは、全表スキャンの実行時にこの表に対して取り出された各ブロックを、バッファ・キャッシュのLRUリストの最高使用頻度側に入れることを指定します。この属性は、小規模な参照表で有効です。NOCACHEは、ブロックをLRUリストの最低使用頻度側に入れることを指定します。この句の詳細は、CREATE TABLEのドキュメントの「CACHE | NOCACHE | CACHE READS」を参照してください。

alter_iot_clauses

`alter_iot_clauses`を使用すると、索引構成マテリアライズド・ビューの特性を変更できます。`alter_iot_clauses`のコンポーネントのキーワードおよびパラメータのセマンティクスは、ALTER TABLEと同じですが、次の制限事項があります。

索引構成マテリアライズド・ビューの変更の制限事項

`index_org_table_clause`の`mapping_table_clause`および`prefix_compression`句は指定できません。

関連項目:

索引構成マテリアライズド・ビューの作成については、「CREATE MATERIALIZED VIEW」の「`index_org_table_clause`」を参照してください。

USING INDEX句

この句を使用すると、マテリアライズド・ビューのデータをメンテナンスするために使用される索引の`INITRANS`パラメータおよび`STORAGE`パラメータの値を変更できます。

USING INDEX句の制限事項

この句ではPCTUSEDおよびPCTFREEパラメータは指定できません。

MODIFY scoped_table_ref_constraint

`MODIFY scoped_table_ref_constraint`句を使用すると、新しい表または新しい列の別名にREF列または属性の有効範囲を再指定できます。

REF列の有効範囲の再指定の制限事項

各`ALTER MATERIALIZED VIEW`文で、1つのREF列または属性のみの有効範囲を再指定でき、この句がこの文で唯一の句である必要があります。

alter_mv_refresh

`alter_mv_refresh`句を使用すると、自動リフレッシュの方法、モードおよび日時のデフォルト値を変更できます。マテリアライズド・ビューのマスター表の内容が変更された場合、マテリアライズド・ビューのデータを更新し、現在マスター表にあるデータを正確に反映させる必要があります。この句によって、自動的にマテリアライズド・ビューをリフレッシュする日時をスケジューリングし、リフレッシュの方法およびモードを指定できます。

関連項目:

- この句では、デフォルトのリフレッシュ・オプションのみを設定します。リフレッシュを実際に行う手順は、『Oracle Database管理者ガイド』および『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- リフレッシュ統計を使用してマテリアライズド・ビューのリフレッシュ操作のパフォーマンスを監視する方法を学習するには、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

FAST句

FASTを指定すると、高速リフレッシュ方法を指定できます。これはマスター表に対して行った変更に従ってリフレッシュを行います。この変更は、マスター表に関連付けられたマテリアライズド・ビュー・ログ(従来型DML変更の場合)またはダイレクト・ローダー・ログ(ダイレクト・パス・インサート操作の場合)に格納されます。

従来型DMLの変更の場合も、ダイレクト・パス・インサート操作の場合も、他の条件によって、高速リフレッシュへのマテリアライズド・ビューの適応性が制限されることがあります。

ALTER MATERIALIZED VIEW文でリフレッシュ方法をFASTに変更した場合、これは検証されていません。マテリアライズド・ビューが高速リフレッシュに適応しない場合、このビューをリフレッシュしようとするとエラーが戻されます。

関連項目:

- レプリケーション環境における高速リフレッシュの制限事項は、『Oracle Database管理者ガイド』を参照してください。
- データ・ウェアハウス環境における高速リフレッシュの制限については、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- 「自動リフレッシュ: 例」

COMPLETE句

COMPLETEを指定すると、完全リフレッシュ(マテリアライズド・ビューを定義する問合せを実行することによって実装)が実行されます。完全リフレッシュを指定すると、高速リフレッシュが実行可能であっても、完全リフレッシュが実行されます。

関連項目:

「完全リフレッシュ: 例」

FORCE句

FORCEを指定すると、リフレッシュ時に、高速リフレッシュが可能な場合は高速リフレッシュを実行し、そうでない場合は完全リフレッシュを実行できます。

ON COMMIT句

ON COMMITを指定すると、マテリアライズド・ビューのマスター表に対するトランザクションをコミットするときに必ずリフレッシュが実行されます。

ON COMMITおよびON DEMANDの両方を指定することはできません。ON COMMITを指定した場合、START WITHまたはNEXTを指定できません。

ON COMMITの制限事項

この句は、マテリアライズド結合ビューおよび単一表マテリアライズド集計ビューでのみサポートされます。

ON DEMAND句

ON DEMANDを指定すると、マテリアライズド・ビューは、3つのDBMS_MVIEWリフレッシュ・プロシージャのいずれかのコールによる要求でリフレッシュされます。ON COMMITおよびON DEMANDのどちらも指定しなかった場合、ON DEMANDがデフォルトになります。

ON COMMITおよびON DEMANDの両方を指定することはできません。START WITHおよびNEXTは、ON DEMANDより優先されます。このため、START WITHまたはNEXTを指定したときは、ほとんどの場合、ON DEMANDを指定しても意味がありません。

関連項目:

- これらのプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- REFRESH ON DEMANDを指定することによって作成できるマテリアライズド・ビューのタイプについては、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

START WITH句

START WITH dateを指定すると、最初の自動リフレッシュ時間を表す日付を指定できます。

NEXT句

NEXTを指定すると、自動リフレッシュの間隔を計算するための日付式を指定できます。

START WITH値およびNEXT値は、将来の時刻に評価される値です。START WITH値を省略した場合、Oracle Databaseはマテリアライズド・ビューのNEXT式を評価することによって、最初の自動リフレッシュ時刻を判断します。START WITH値を指定し、NEXT値を指定しない場合、Oracle Databaseは1回のみマテリアライズド・ビューをリフレッシュします。START WITH値およびNEXT値のどちらも指定しない場合、またはalter_mv_refreshを指定しない場合、Oracle Databaseはマテリアライズド・ビューを自動リフレッシュしません。

WITH PRIMARY KEY句

WITH PRIMARY KEYを指定すると、ROWIDマテリアライズド・ビューを主キー・マテリアライズド・ビューに変更できます。主キー・マテリアライズド・ビューを使用すると、高速リフレッシュを継続できるマテリアライズド・ビューの機能に影響せずに、マテリアライズド・ビュー・マスター表を再編成できます。

この句を指定するには、マスター表に、使用可能な主キー制約が定義され、この制約に基づき、主キー情報を記録するマテリアライズド・ビュー・ログが定義されている必要があります。

関連項目:

- 主キー・マテリアライズド・ビューの詳細は、『Oracle Database管理者ガイド』を参照してください。
- 主キー・マテリアライズド・ビュー: 例

USING ROLLBACK SEGMENT句

自動UNDOモードではロールバック・セグメントではなくUNDO表領域が使用されるため、データベースが自動UNDOモードの場合、この句は無効です。自動UNDOモードを使用することをお薦めします。この句は、ロールバック・セグメントが使用される以前のバージョンのOracle Databaseが含まれるレプリケーション環境との下位互換性のためにサポートされています。

この句の詳細は、「CREATE MATERIALIZED VIEW」の「USING ROLLBACK SEGMENT句」を参照してください。

USING ... CONSTRAINTS句

この句のセマンティクスは、CREATE MATERIALIZED VIEWおよびALTER MATERIALIZED VIEW文で同じです。詳細は、「CREATE MATERIALIZED VIEW」の「USING ... CONSTRAINTS句」を参照してください。

evaluation_edition_clause

この句を使用すると、マテリアライズド・ビューの評価エディションを変更できます。この句のセマンティクスは、CREATE MATERIALIZED VIEWおよびALTER MATERIALIZED VIEW文で同じです。この句の詳細は、CREATE MATERIALIZED VIEWの「evaluation_edition_clause」を参照してください。

マテリアライズド・ビューの評価エディションの変更のノート

マテリアライズド・ビューの評価エディションを変更するときには、次のノートが適用されます。

- REFRESH ON COMMITモードのマテリアライズド・ビューを変更するきに、CONSIDER FRESHを指定していないと、Oracle Databaseはマテリアライズド・ビューの完全リフレッシュを実行します。
- REFRESH ON DEMANDモードのマテリアライズド・ビューを変更するきに、CONSIDER FRESHを指定していないと、Oracle Databaseはマテリアライズド・ビューの失効状態をSTALEに設定します。
- REFRESH ON COMMITモードおよびREFRESH ON DEMANDモードのマテリアライズド・ビューの場合、評価エディションを変更して、CONSIDER FRESHを指定すると、Oracle Databaseはマテリアライズド・ビューの失効状態の更新と、マテリアライズド・ビューの再構築を実行しなくなります。そのため、CONSIDER FRESHを指定することで、評価エディションが変更されていても、subqueryが生成する結果に違いがないことを示せます。マテリアライズド・ビューが失効していて、この文を発行する前に高速リフレッシュまたは完全リフレッシュが必要になる場合、この状態は変更されないため、マテリアライズド・ビューには不適切なデータが含まれる可能性があります。

{ ENABLE | DISABLE } ON QUERY COMPUTATION

この句を使用すると、マテリアライズド・ビューがリアルタイムのマテリアライズド・ビューまたは通常のマテリアライズド・ビューのいずれであるかを制御できます。

- 問合せ時計算を有効にして、通常のマテリアライズド・ビューをリアルタイムのマテリアライズド・ビューに変換する場合は、ENABLE ON QUERY COMPUTATIONを指定します。
- 問合せ時計算を無効にして、リアルタイムのマテリアライズド・ビューを通常のマテリアライズド・ビューに変換する場合は、DISABLE ON QUERY COMPUTATIONを指定します。

この句のセマンティクスは、CREATE MATERIALIZED VIEWおよびALTER MATERIALIZED VIEW文で同じです。この句の詳細は、CREATE MATERIALIZED VIEWの{ ENABLE | DISABLE } ON QUERY COMPUTATIONを参照してください。

alter_query_rewrite_clause

この句を使用すると、マテリアライズド・ビューをクエリー・リライトで使用できるかどうかを指定できます。

ENABLE句

ENABLEを指定すると、クエリー・リライトでマテリアライズド・ビューを使用可能にできます。マテリアライズド・ビューにunusable_editions_clause指定している場合や、以前に指定していた場合、使用禁止のエディションでクエリー・リライトを有効にすることはできません。

関連項目:

- クエリー・リライトの有効化：例
- リフレッシュ統計を使用してマテリアライズド・ビューのリフレッシュ操作のパフォーマンスを監視する方法を学習するには、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

マテリアライズド・ビューの有効化の制限事項

マテリアライズド・ビューの有効化には、次の制限事項があります。

- マテリアライズド・ビューが無効または使用禁止の場合、ENABLEモードでもクエリー・リライトに適応しません。
- マテリアライズド・ビューの全体または一部がビューから作成されている場合、クエリー・リライトを使用可能にできません。
- マテリアライズド・ビューのすべてのユーザー定義ファンクションがDETERMINISTICである場合のみ、クエリー・リライトを使用可能にできます。

  関連項目:

  CREATE FUNCTION

- 文内の式が反復可能な場合のみ、クエリー・リライトを使用可能にできます。たとえば、CURRENT_TIMEまたはUSERを含めることはできません。

関連項目:

クエリー・リライトの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

DISABLE句

DISABLEを指定すると、クエリー・リライトでマテリアライズド・ビューを使用禁止にできます。マテリアライズド・ビューが無効な場合、使用禁止であるかどうかにかかわらず、クエリー・リライトの使用には適応しません。ただし、使用禁止にされたマテリアライズド・ビューをリフレッシュすることはできます。

unusable_editions_clause

この句を使用すると、クエリー・リライトに使用できないマテリアライズド・ビューのエディションを指定できます。この句のセマンティクスは、CREATE MATERIALIZED VIEWおよびALTER MATERIALIZED VIEW文で同じです。この句の詳細は、CREATE MATERIALIZED VIEWの「unusable_editions_clause」を参照してください。

クエリー・リライトにマテリアライズド・ビューを使用し、使用禁止にされたエディションでコンパイルされたカーソルは無効になります。

COMPILE

COMPILEを指定すると、マテリアライズド・ビューを明示的に再検証できます。マテリアライズド・ビューが依存するオブジェクトを削除または変更した場合、マテリアライズド・ビューはアクセス可能のままですが、クエリー・リライトに対しては無効です。再度、明示的にマテリアライズド・ビューの妥当性チェックを行い、クエリー・リライトの使用に適応させるには、この句を使用します。

マテリアライズド・ビューの再妥当性チェックに失敗すると、リフレッシュできなくなるか、またはクエリー・リライトに使用できなくなります。

関連項目:

マテリアライズド・ビューのコンパイル: 例

CONSIDER FRESH

この句を使用すると、マスター表が変更された後のマテリアライズド・ビューの失効状態を管理することができます。CONSIDER FRESHは、マテリアライズド・ビューが最新であり、TRUSTEDまたはSTALE_TOLERATEDモードでのクエリー・リライトに適応するとみなされるように指定します。

注意:

CONSIDER FRESH 句は、CONSIDER FRESH 句を発行する前にマテリアライズド・ビュー・ログの行を適用したり、パーティション・チェンジ・トラッキングの変更をマテリアライズド・ビューに適用しないようにデータベースに指示します。つまり、保留中の変更は無視され削除されます。これらはマテリアライズド・ビューに適用されません。これにより、マテリアライズド・ビューに、実表とほぼ同じ量のデータが含まれる可能性があります。

Oracle Databaseは、マテリアライズド・ビューが最新であるかどうかを保証できないため、ENFORCEDモードでのクエリー・リライトはサポートしません。また、この句はマテリアライズド・ビューの失効状態をUNKNOWNに設定します。失効状態は、ALL_MVIEWS、DBA_MVIEWSおよびUSER_MVIEWSの各データ・ディクショナリ・ビューのSTALENESS列に表示されます。

いずれかのマスター表の内容が変更された場合、マテリアライズド・ビューは失効します。この句は、Oracle Databaseに対して、マテリアライズド・ビューが最新で、変更されていないものと仮定するように指示します。そのため、リフレッシュが保留されているこれらの表に対する実際の更新内容は、マテリアライズド・ビューから削除されます。

関連項目:

- クエリー・リライトの詳細、およびマスター表へのパーティション・メンテナンス操作の影響については、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- 「CONSIDER FRESH: 例」

例

自動リフレッシュ: 例

次の文は、マテリアライズド・ビューsales_by_month_by_state (「マテリアライズド集計ビューの作成: 例」で作成)のリフレッシュ方法のデフォルトをFASTに変更します。

```
ALTER MATERIALIZED VIEW sales_by_month_by_state
   REFRESH FAST;
```

これ以降のマテリアライズド・ビューの自動リフレッシュは、高速リフレッシュになります。これは、単純なマテリアライズド・ビューであり、そのマスター表には、マテリアライズド・ビューの作成前または最後のリフレッシュ前に作成されたマテリアライズド・ビュー・ログがあります。

REFRESH句にSTART WITHまたはNEXTの値が指定されていないため、マテリアライズド・ビューsales_by_month_by_stateを作成したとき、または最後に変更したときにREFRESH句で設定されたリフレッシュ間隔がそのまま使用されます。

次の文は、マテリアライズド・ビューsales_by_month_by_stateの新しい自動リフレッシュ間隔を設定します。

```
ALTER MATERIALIZED VIEW sales_by_month_by_state
   REFRESH NEXT SYSDATE+7;
```

REFRESH句にSTART WITHの値が指定されていないため、マテリアライズド・ビューsales_by_month_by_stateが作成されたとき、または最後に変更されたときに指定されたSTART WITHとNEXTの値によって設定された日時に次の自動リフレッシュが行われます。

次回の自動リフレッシュのときに、このマテリアライズド・ビューがリフレッシュされるとともに、NEXTの式SYSDATE+7が評価されて次回の自動リフレッシュの日時が決定し、それ以降もこのマテリアライズド・ビューのリフレッシュは自動的に週1回行われます。

REFRESH句ではリフレッシュ方法が明示的には指定されていないため、CREATE MATERIALIZED VIEW文または最後に実行されたALTER MATERIALIZED VIEW文のREFRESH句で指定されたリフレッシュ方法が引き続き使用されます。

CONSIDER FRESH: 例

次の文は、Oracle Databaseがマテリアライズド・ビューsales_by_month_by_stateを最新であるとみなすように指定します。この文を使用すると、sales_by_month_by_stateのマスター表に対してパーティション・メンテナンス操作を実行した後でも、sales_by_month_by_stateに対してTRUSTEDモードでのクエリー・リライトが可能です。

```
ALTER MATERIALIZED VIEW sales_by_month_by_state CONSIDER FRESH;
```

前の文の結果として、マテリアライズド・ビューの最後のリフレッシュ以降に実表に加えられたパーティション管理操作は、マテリアライズド・ビューには適用されません。たとえば、マテリアライズド・ビューの次のリフレッシュの前にCONSIDER FRESHが使用されると、実表内のパーティションの中のデータに対する追加、削除または変更はマテリアライズド・ビューに反映されません。詳細は、「CONSIDER FRESH」を参照してください。

関連項目:

このALTER MATERIALIZED VIEWの例を必要とするパーティション・メンテナンスの例は、「表パーティションの分割: 例」を参照してください。

完全リフレッシュ: 例

次の文は、マテリアライズド・ビューemp_data (「マテリアライズド・ビューの定期的リフレッシュ: 例」で作成)の新しいリフレッシュ方法、NEXTで示す新しいリフレッシュ時間および新しい自動リフレッシュ間隔を指定します。

```
ALTER MATERIALIZED VIEW emp_data
   REFRESH COMPLETE
   START WITH TRUNC(SYSDATE+1) + 9/24
   NEXT SYSDATE+7;
```

START WITH句の値によって、このマテリアライズド・ビューの次回の自動リフレッシュは翌日の午前9時に行うように設定されます。この時刻になると、マテリアライズド・ビューの完全リフレッシュが実行されるとともに、NEXTの式が評価されて、それ以降はこのマテリアライズド・ビューのリフレッシュが毎週実行されます。

クエリー・リライトの有効化: 例

次の文は、マテリアライズド・ビューemp_dataのクエリー・リライトを有効にし、暗黙的に再検証します。

```
ALTER MATERIALIZED VIEW emp_data
   ENABLE QUERY REWRITE;
```

主キー・マテリアライズド・ビュー: 例

次の文は、ROWIDマテリアライズド・ビューorder_data (「ROWIDマテリアライズド・ビューの作成: 例」で作成)を主キー・マテリアライズド・ビューに変更します。この例では、order_dataに主キーを持つマテリアライズド・ビュー・ログを定義していることが必要です。

```
ALTER MATERIALIZED VIEW order_data
   REFRESH WITH PRIMARY KEY;
```

マテリアライズド・ビューのコンパイル: 例

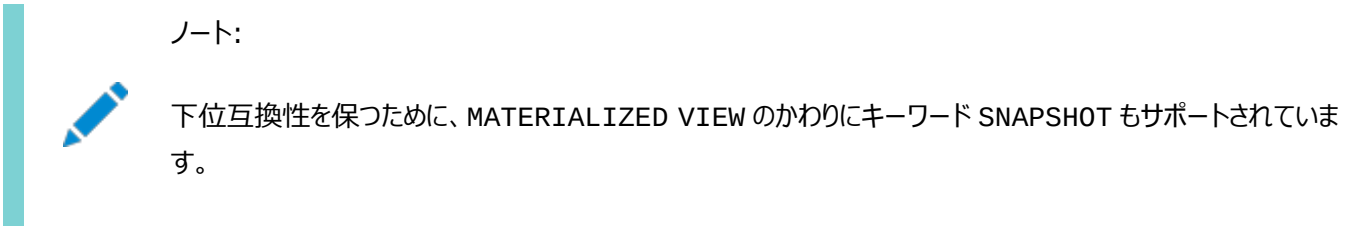
次の文は、マテリアライズド・ビューstore_mvを再検証します。

```
ALTER MATERIALIZED VIEW order_data COMPILE;
```

ALTERMATERIALIZEDVIEWLOG

目的

マテリアライズド・ビュー・ログとは、マテリアライズド・ビューのマスター表に関連付けられる表です。ALTER MATERIALIZED VIEW LOG文を使用すると、既存のマテリアライズド・ビュー・ログの記憶特性またはタイプを変更できます。



ノート:

下位互換性を保つために、MATERIALIZED VIEW のかわりにキーワード SNAPSHOT もサポートされています。

関連項目:

- マテリアライズド・ビュー・ログの作成については、「CREATE MATERIALIZED VIEW LOG」を参照してください。
- マテリアライズド・ビューのリフレッシュ方法など、マテリアライズド・ビューの詳細は、「ALTER MATERIALIZED VIEW」を参照してください。
- マテリアライズド・ビューの様々なタイプの詳細は、「CREATE MATERIALIZED VIEW」を参照してください。

前提条件

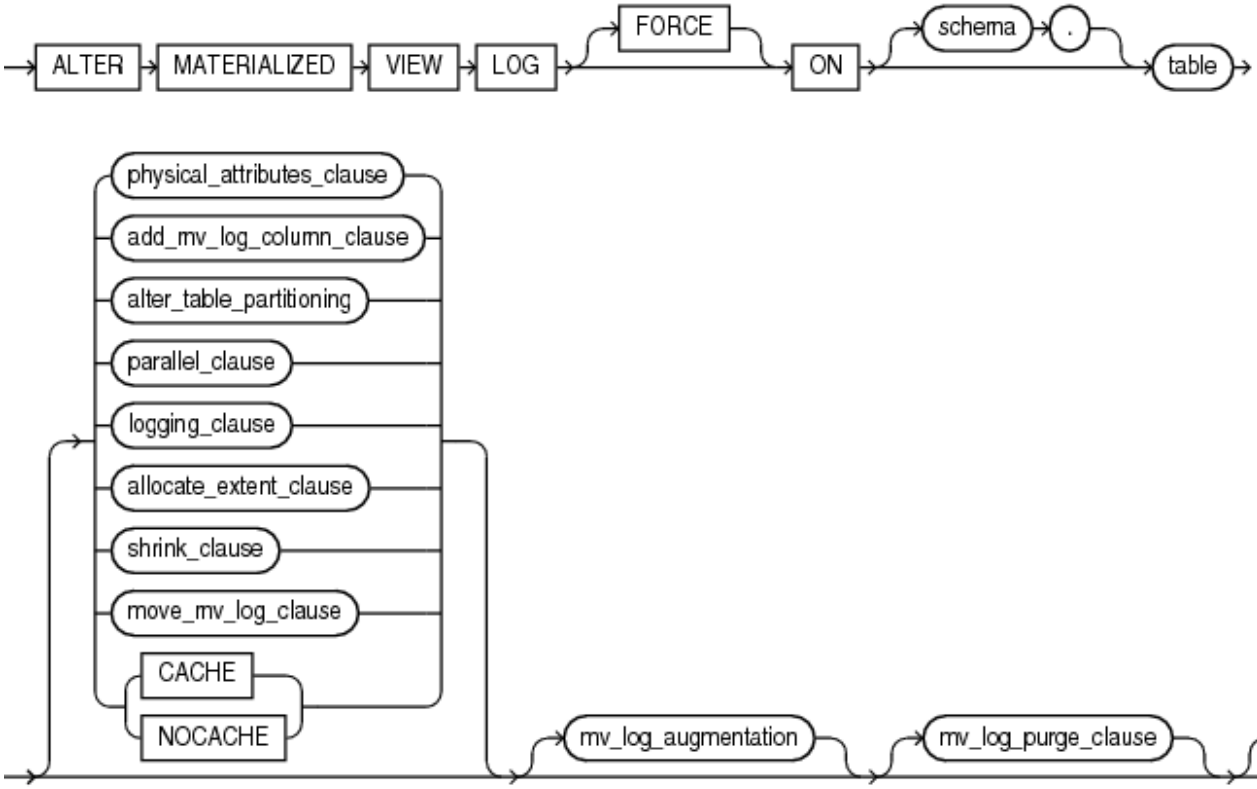
マスター表の所有者であるか、またはマスター表に対するREADまたはSELECT権限およびマテリアライズド・ビュー・ログに対するALTER権限が必要です。

関連項目:

ALTER MATERIALIZED VIEW LOGの前提条件の詳細は、『Oracle Database管理者ガイド』を参照してください。

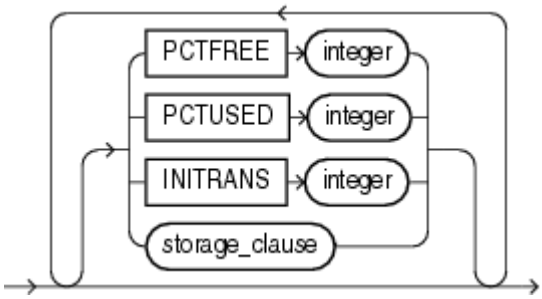
構文

alter_materialized_view_log::=



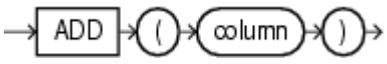
(physical_attributes_clause::=、add_mv_log_column_clause::=、alter_table_partitioning::= (ALTER TABLEの)、parallel_clause::=、logging_clause::=、allocate_extent_clause::=、shrink_clause::=、move_mv_log_clause::=、mv_log_augmentation::=、mv_log_purge_clause::=、for_refresh_clause::=)

physical_attributes_clause::=

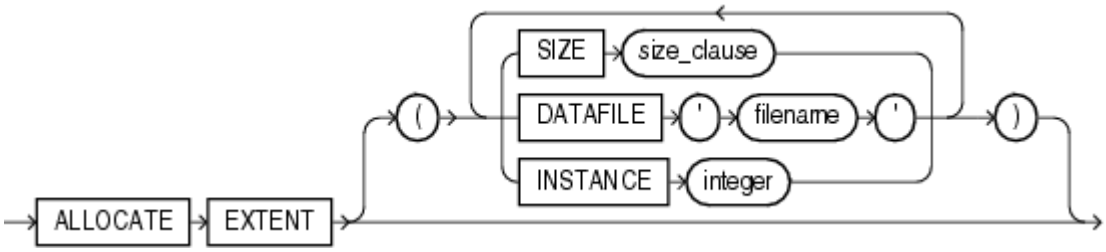


storage_clause::=

add_mv_log_column_clause::=

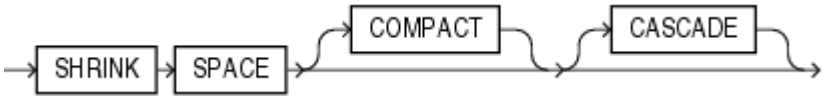


allocate_extent_clause::=

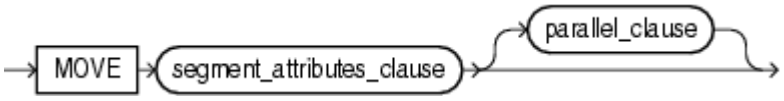


(size_clause::=)

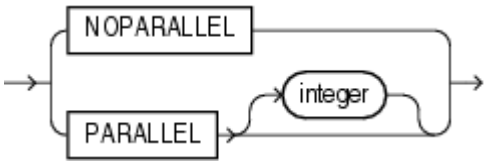
shrink_clause::=



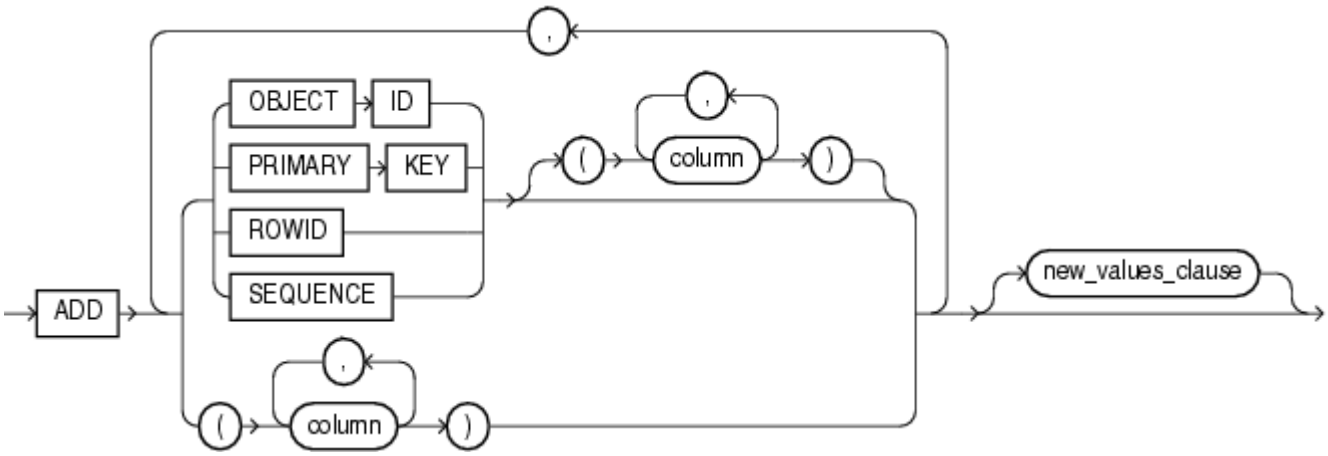
move_mv_log_clause::=



parallel_clause::=

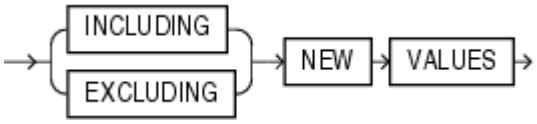


mv_log_augmentation::=

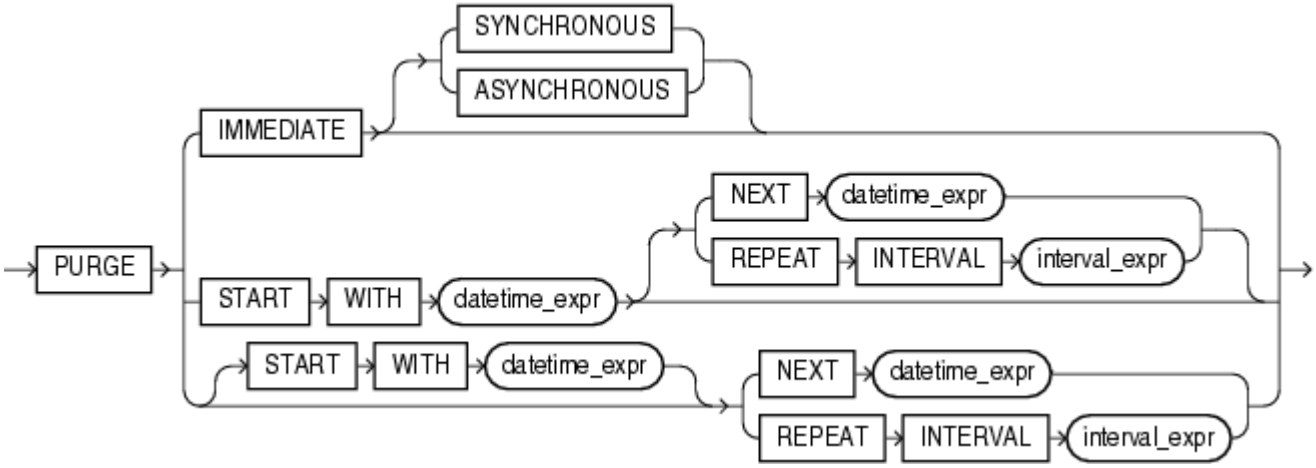


(new_values_clause::=

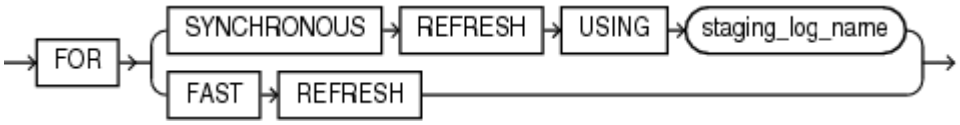
new_values_clause::=



mv_log_purge_clause::=



for_refresh_clause::=



セマンティクス

FORCE

FORCEを指定すると、ADD句で指定したいずれかの項目がすでにマテリアライズド・ビュー・ログに指定されている場合、エラーは戻されませんが、既存の要素は無視され、マテリアライズド・ビュー・ログに存在しないすべての項目が追加されます。同様に、INCLUDING NEW VALUESを指定すると、この属性がすでにマテリアライズド・ビュー・ログに指定されている場合、冗長は無視され、エラーも戻されません。

schema

マスター表が定義されているスキーマを指定します。schemaを指定しない場合、マテリアライズド・ビュー・ログは自分のスキーマ内にあるとみなされます。

table

変更するマテリアライズド・ビュー・ログに関連付けられたマスター表の名前を指定します。

physical_attributes_clause

physical_attributes_clauseを使用すると、PCTFREE、PCTUSEDおよびINITRANSの各パラメータの値、マテリアライズド・ビュー・ログ、パーティションおよびオーバーフロー・データ・セグメントの記憶特性、またはパーティション・マテリアライズド・ビュー・ログのデフォルト特性を変更できます。

マテリアライズド・ビュー・ログの物理属性の制限事項

マテリアライズド・ビュー・ログがローカル管理表領域内に存在する場合は、storage_clauseを使用してエクステント・パラメータを変更することはできません。このパラメータについては、「CREATE TABLE」を参照してください。

add_mv_log_column_clause

マテリアライズド・ビュー・ログのマスター表に列を追加した場合、列はマテリアライズド・ビュー・ログに自動的には追加されません。そのため、この句を使用してマテリアライズド・ビュー・ログに列を追加します。マスター表の対応する列が暗号化されている場合、新規に追加された列はOracle Databaseによって暗号化されます。

alter_table_partitioning

パーティション化の句の構文および一般的な機能は、ALTER TABLE文の場合と同じです。「ALTER TABLE」の「alter_table_partitioning」を参照してください。

マテリアライズド・ビュー・ログのパーティションの変更の制限事項

マテリアライズド・ビュー・ログのパーティションの変更には、次の制限事項があります。

- マテリアライズド・ビュー・ログのパーティションを変更する場合、LOB_storage_clauseまたはmodify_LOB_storage_clauseは使用できません。
- マテリアライズド・ビュー・ログのパーティションを削除、切捨てまたは交換しようとすると、Oracle Databaseはエラーを戻します。

parallel_clause

parallel_clauseを使用すると、マテリアライズド・ビュー・ログへのパラレル操作がサポートされているかどうかを指定できます。

この句の詳細は、「CREATE TABLE」の「parallel_clause」を参照してください。

logging_clause

マテリアライズド・ビュー・ログに対するロギング属性を指定します。この句の詳細は、「logging_clause」を参照してください。

allocate_extent_clause

allocate_extent_clauseを使用すると、マテリアライズド・ビュー・ログの新しいエクステントを明示的に割り当てることができます。この句の詳細は、「allocate_extent_clause」を参照してください。

shrink_clause

この句を使用すると、マテリアライズド・ビュー・ログのセグメントを縮小化できます。この句の詳細は、「CREATE TABLE」の「shrink_clause」を参照してください。

move_mv_log_clause

MOVE句を使用すると、マテリアライズド・ビュー・ログ表を他の表領域に移動して、マテリアライズド・ビュー・ログの他のセグメントまたは記憶域属性を変更したり、マテリアライズド・ビュー・ログのパラレル化を変更できます。

マテリアライズド・ビュー・ログの移動の制限事項

segment_attributesのstorage_clauseのENCRYPT句は、マテリアライズド・ビュー・ログでは無効です。

CACHE | NOCACHE句

アクセス頻度が高いデータについて、CACHEは、全表スキャンの実行時にこのログ用に取り出された各ブロックを、バッファ・キャッシュ内のLRUリストの最高使用頻度側に入れることを指定します。この属性は、小規模な参照表で有効です。NOCACHEは、ブロックをLRUリストの最低使用頻度側に入れることを指定します。この句の詳細は、CREATE TABLEのドキュメントの「CACHE | NOCACHE | CACHE READS」を参照してください。

mv_log_augmentation

ADD句を使用すると、マテリアライズド・ビュー・マスター表内の行が変更される際に、主キー値、ROWID値、オブジェクトID値または順序も記録するようにマテリアライズド・ビュー・ログを拡張できます。また、この句は、新しく列を記録するためにも使用できます。

これらの情報の記録を停止する場合は、マテリアライズド・ビュー・ログを削除してから、再作成する必要があります。マテリアライズド・ビュー・ログを削除した後再作成した場合、マスター表に依存するすべての既存マテリアライズド・ビューが、次回のリフレッシュ時に強制的に完全リフレッシュされます。

マテリアライズド・ビュー・ログの拡張の制限事項

各マテリアライズド・ビュー・ログに指定できるのは、PRIMARY KEY、ROWID、OBJECT ID、SEQUENCE、および列リストの各列を1つずつです。このALTER文にPRIMARY KEY、ROWID、OBJECT ID、SEQUENCEおよび列リストを指定できるのはそれぞれ1回のみです。また、FORCEオプションを指定しないかぎり、これらの値のいずれかが作成時に(暗黙的または明示的に)指定された場合、このALTER文にはそれらの値を指定できません。

OBJECT ID

OBJECT IDを指定すると、更新されるすべての行の適切なオブジェクト識別子をマテリアライズド・ビュー・ログに記録できます。

OBJECT ID句の制限事項

OBJECT IDはオブジェクト表のログに対してのみ指定でき、記憶表に対しては指定できません。

PRIMARY KEY

PRIMARY KEYを指定すると、更新されるすべての行の主キー値をマテリアライズド・ビュー・ログに記録できます。

ROWID

ROWIDを指定すると、更新されるすべての行のROWID値をマテリアライズド・ビュー・ログに記録できます。

SEQUENCE

SEQUENCEを指定すると、追加の順序情報を提供する順序値をマテリアライズド・ビュー・ログに記録できます。

column

更新されるすべての行に対して、マテリアライズド・ビュー・ログに記録する値を持つ新しい列を指定します。通常、フィルタ列(副問合せマテリアライズド・ビューが参照する主キー以外の列)および結合列(副問合せのWHERE句で結合を定義する主キー以外の列)を指定します。

関連項目:

- マテリアライズド・ビュー・ログに値を明示的および暗黙的に含める方法については、「CREATE MATERIALIZED VIEW」を参照してください。
- フィルタ列および結合列の詳細は、『Oracle Database管理者ガイド』を参照してください。
- ROWIDマテリアライズド・ビュー・ログ: 例

NEW VALUES句

NEW VALUES句を使用すると、更新DML操作で、古い値と新しい値の両方をマテリアライズド・ビュー・ログに保存するかどうかを指定できます。ALTER MATERIALIZED VIEW LOG文で追加した列のみでなく、ログのすべての列にこの句で設定した値を適用します。

INCLUDING

INCLUDINGを指定すると、新しい値と古い値の両方をログに保存できます。このログが単一表マテリアライズド集計ビューの表用で、マテリアライズド・ビューに高速リフレッシュを実行する場合、INCLUDINGを指定してください。

EXCLUDING

EXCLUDINGを指定すると、ログに新しい値が記録されなくなります。この句を使用すると、新しい値の記録によるオーバーヘッドを回避できます。

マテリアライズド・ビューのリフレッシュ・モードをFAST以外のモードに変更した場合を除き、高速リフレッシュが可能な単一表マテリアライズド集計ビューが定義されている場合は、EXCLUDING NEW VALUESを使用しないでください。

関連項目:

マテリアライズド・ビュー・ログEXCLUDING NEW VALUES: 例

mv_log_purge_clause

この句を使用すると、次のようにマテリアライズド・ビュー・ログのパージ属性を変更できます。

- パージを、IMMEDIATE SYNCHRONOUSからIMMEDIATE ASYNCHRONOUSに、またはIMMEDIATE ASYNCHRONOUSからIMMEDIATE SYNCHRONOUSに変更します。
- パージをIMMEDIATEからスケジュール実行に、またはスケジュール実行からIMMEDIATEに変更します。
- 新しい開始時間と、新しいnext時間およびintervalを指定します。

パージをスケジュール実行からIMMEDIATEに変更すると、そのマテリアライズド・ビュー・ログに関連付けられているスケジュール実行のパージ・ジョブは削除されます。パージをIMMEDIATEからスケジュール実行に変更すると、指定した属性を持つパージ・ジョブが作成されます。スケジュール実行のパージ属性を変更すると、スケジューラ・パージ・ジョブ内では指定した属性のみが変更されます。

スケジュール実行のパージ属性を変更する場合を除いて、ログ・パージを現在の状態に変更する場合(つまり変更しない状態にする場合)は、FORCEを指定する必要があります。

マテリアライズド・ビュー・ログに対してパージ時間または間隔がすでに設定されているかどうかを確認するには、*_MVIEW_LOGSデータ・ディクショナリ・ビューを問い合せます。この句のセマンティクスの詳細は、「CREATE MATERIALIZED VIEW LOG」の句「mv_log_purge_clause」を参照してください。

for_refresh_clause

この句を使用すると、マテリアライズド・ビュー・ログに使用するリフレッシュ方法を変更できます。

FOR SYNCHRONOUS REFRESH

この句を指定すると、高速リフレッシュまたは完全リフレッシュを同期リフレッシュに変更できます。ステージング・ログが作成されます。

高速リフレッシュを変更する場合は、この句を使用する前に、次の条件が満たされていることを確認してください。

- マテリアライズド・ビュー・ログ内のすべての変更を使用している。
- マスター表に関連付けられたREFRESH ON DEMANDモードのマテリアライズド・ビューがリフレッシュされている。
- マスター表に関連付けられたREFRESH ON COMMITモードのマテリアライズド・ビューがREFRESH ON DEMANDモードのマテリアライズド・ビューに変換されている。

この句の使用後には、マスター表に対する直接のDML操作は実行できなくなります。データの変更操作の準備と実行には、DBMS_SYNC_REFRESHパッケージに含まれるプロシージャを使用する必要があります。

FOR FAST REFRESH

この句を指定すると、同期リフレッシュまたは完全リフレッシュを高速リフレッシュに変更できます。マテリアライズド・ビュー・ログが作成されます。

同期リフレッシュを高速リフレッシュに変更する場合は、この句を使用する前に、ステージング・ログ内のすべての変更を使用していることを確認してください。

この句の使用後には、マスター表に対する直接のDML操作を実行できるようになります。

この句のセマンティクスの詳細は、CREATE MATERIALIZED VIEW LOGの句「for_refresh_clause」を参照してください。

例

ROWIDマテリアライズド・ビュー・ログ: 例

次の文は、既存の主キー・マテリアライズド・ビュー・ログを変更して、ROWID情報も記録されるようにします。

```
ALTER MATERIALIZED VIEW LOG ON order_items ADD ROWID;
```

マテリアライズド・ビュー・ログEXCLUDING NEW VALUES: 例

次の文は、フィルタ列を追加し、新規の値を除外することによって、hr.employeesのマテリアライズド・ビュー・ログを変更します。このログを使用するマテリアライズド集計ビューは、これ以降高速リフレッシュされません。ただし、高速リフレッシュが不要になる場合は、この処理によって新しい値の記録によるオーバーヘッドを回避できます。

```
ALTER MATERIALIZED VIEW LOG ON employees
   ADD (commission_pct)
   EXCLUDING NEW VALUES;
```

ALTER MATERIALIZED ZONEMAP

目的

ALTER MATERIALIZED ZONEMAP文を使用すると、既存のゾーン・マップを次の方法で変更できます。

- 属性の変更
- デフォルトのリフレッシュ方法とモードの変更
- プルーニングの使用の有効化または無効化
- コンパイル、再作成または使用不可にする

関連項目:

- ゾーン・マップの作成の詳細は、CREATE MATERIALIZED ZONEMAPを参照してください。
- ゾーン・マップの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

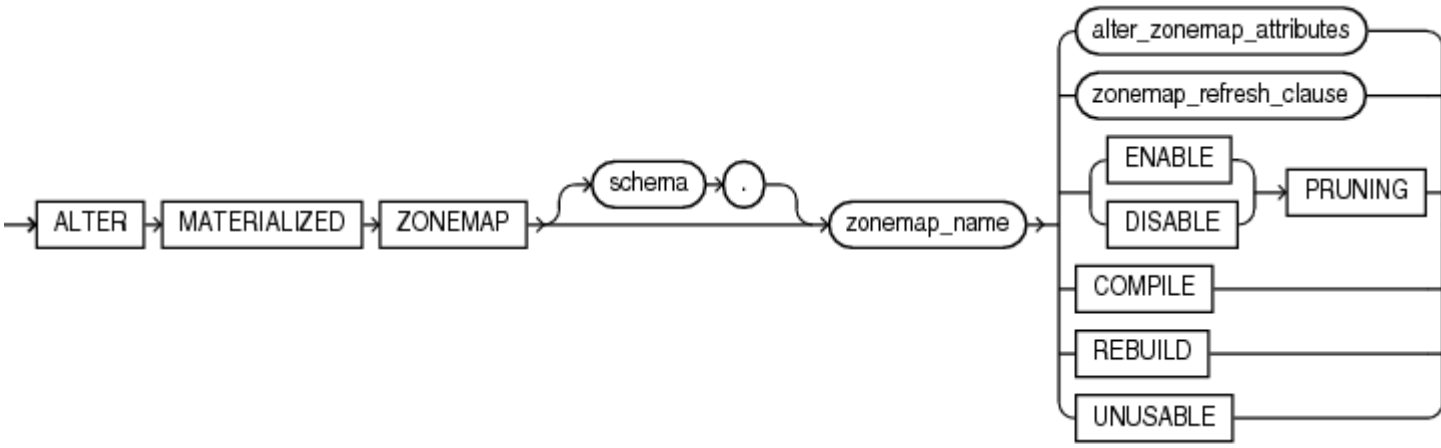
前提条件

ゾーン・マップが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY MATERIALIZED VIEWシステム権限が必要です。

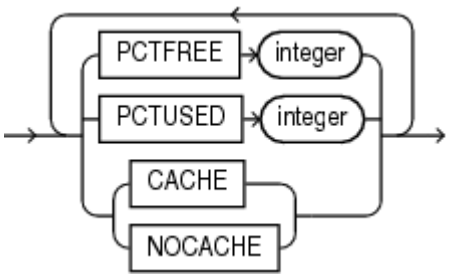
ゾーン・マップを含むスキーマを所有するユーザーは、各表のREADまたはSELECTオブジェクト権限あるいはREAD ANY TABLEまたはSELECT ANY TABLEシステム権限を介して、スキーマの外にあるゾーン・マップの実表のアクセスを必要とします。

構文

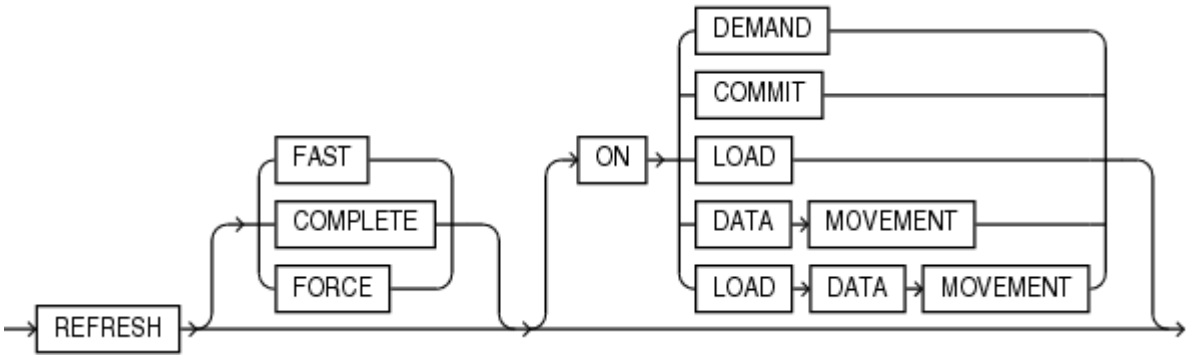
alter_materialized_zonemap::=



alter_zonemap_attributes::=



zonemap_refresh_clause::=



ノート:

zonemap_refresh_clause を指定する場合、REFRESH キーワードの後に少なくとも 1 つの句を指定する必要があります。

セマンティクス

schema

ゾーン・マップが含まれているスキーマを指定します。schemaを指定しない場合、このゾーン・マップは自分のスキーマ内にあるとみなされます。

zonemap_name

変更するゾーン・マップの名前を指定します。

alter_zonemap_attributes

この句を使用してゾーン・マップの次の属性を変更します: PCTFREE、PCTUSEDおよびCACHEまたはNOCACHE。これらの属性は、ALTER MATERIALIZED ZONEMAPおよびCREATE MATERIALIZED ZONEMAPに対して同じセマンティクスを持ちます。これらの属性の詳細は、CREATE MATERIALIZED ZONEMAPのドキュメントのPCTFREE, PCTUSEDおよびCACHE | NOCACHEを参照してください。

zonemap_refresh_clause

この句を使用して、ゾーン・マップのデフォルトのリフレッシュ方法およびモードを変更します。この句は、ALTER MATERIALIZED ZONEMAPおよびCREATE MATERIALIZED ZONEMAPに対して同じセマンティクスを持ちます。この句の詳細は、CREATE MATERIALIZED ZONEMAPの「zonemap_refresh_clause」を参照してください。

ENABLE | DISABLE PRUNING

この句を使用して、プルーニングのゾーン・マップの使用を有効化または無効化します。この句は、ALTER MATERIALIZED ZONEMAPおよびCREATE MATERIALIZED ZONEMAPに対して同じセマンティクスを持ちます。この句の詳細は、CREATE MATERIALIZED ZONEMAPのENABLE | DISABLE PRUNINGを参照してください。

COMPILE

この句を使用すると、ゾーン・マップを明示的にコンパイルできます。DDL操作が1つ以上の実表の構造を変更した後、この操作はゾーン・マップを検証します。Oracleデータベースが使用前にコンパイルが必要なゾーン・マップを自動的にコンパイルするため、通常この句を発行する必要はありません。ただし、ゾーン・マップを明示的にコンパイルする場合、この句を使用して実行で

きます。

ゾーン・マップのコンパイル結果は、実表がゾーン・マップに影響する方法で変更されるかどうかによって異なります。たとえば、列が実表に追加されると、変更がゾーン・マップに影響しないため、コンパイル後にゾーン・マップが有効になります。ただし、ゾーン・マップの定義する副問合せに含まれる列が実表から削除される場合、コンパイル後にゾーン・マップが無効になります。

ALL_, DBA_のCOMPILE_STATE列およびUSER_ZONEMAPSデータ・ディクショナリ・ビューを問い合せて、ゾーン・マップでコンパイルが必要かどうかを判断できます。列の値がNEEDS_COMPILEの場合、ゾーン・マップにコンパイルが必要です。

REBUILD

この句を使用すると、ゾーン・マップを明示的に再作成できます。この操作は、ゾーン・マップのデータをリフレッシュします。この句は、次の場合に有効です。

- この句を使用して、REFRESH ON DEMANDモードのゾーン・マップのデータをリフレッシュできます。詳細は、CREATE MATERIALIZED ZONEMAPのドキュメントのON DEMAND句を参照してください。
- ゾーン・マップのデフォルト・リフレッシュ・モードに関係なく、ゾーン・マップの実表のいずれかのEXCHANGE PARTITION操作の後にこの句を発行する必要があります。
- ゾーン・マップが使用不可とマークされている場合、この句を発行して使用不可とマークできます。ALL_, DBA_のUNUSABLE列およびUSER_ZONEMAPSデータ・ディクショナリ・ビューを問い合せて、ゾーン・マップが使用不可とマークされているかどうかを判断できます。

UNUSABLE

この句を指定して、ゾーン・マップを使用不可にします。後続の問合せはゾーン・マップを使用しないため、データベースはゾーン・マップを保持しません。ALTER MATERIALIZED ZONEMAP ... REBUILD文を発行して、再度ゾーン・マップを使用可能にすることができます。

例

ゾーン・マップ属性の変更: 例

次の文はゾーン・マップsales_zmapのPCTFREEおよびPCTUSED属性を変更し、キャッシュを使用しないようにゾーン・マップを変更します。

```
ALTER MATERIALIZED ZONEMAP sales_zmap
  PCTFREE 20 PCTUSED 50 NOCACHE;
```

デフォルトのリフレッシュ方法とゾーン・マップのモードの変更: 例

次の文は、ゾーン・マップsales_zmapのデフォルトのリフレッシュ方法をFASTに変更し、デフォルトのリフレッシュ・モードをON COMMITに変更します。

```
ALTER MATERIALIZED ZONEMAP sales_zmap
  REFRESH FAST ON COMMIT;
```

プルーニングのためのゾーン・マップの使用の無効化: 例

次の文は、ゾーン・マップsales_zmapをプルーニングに使用できないようにします。

```
ALTER MATERIALIZED ZONEMAP sales_zmap
  DISABLE PRUNING;
```

ゾーン・マップのコンパイル: 例

次の文は、ゾーン・マップsales_zmapをコンパイルします。

```
ALTER MATERIALIZED ZONEMAP sales_zmap
  COMPILE;
```

ゾーン・マップの再構築: 例

次の文は、ゾーン・マップsales_zmapを再構築します。

```
ALTER MATERIALIZED ZONEMAP sales_zmap
  REBUILD;
```

ゾーン・マップの使用禁止: 例

次の文は、ゾーン・マップsales_zmapを使用禁止にします。

```
ALTER MATERIALIZED ZONEMAP sales_zmap
  UNUSABLE;
```

ALTER OPERATOR

目的

ALTER OPERATOR文を使用すると、既存の演算子に対するバインドを追加または削除したり、既存の演算子をコンパイルすることができます。

関連項目:

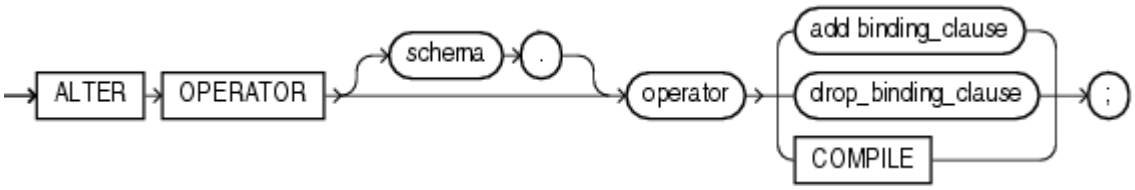
CREATE OPERATOR

前提条件

事前にCREATE OPERATOR文によって演算子が作成されている必要があります。演算子が自分のスキーマ内にあるか、またはALTER ANY OPERATORシステム権限が必要です。ALTER OPERATOR文で参照される演算子およびファンクションに対するEXECUTEオブジェクト権限が必要です。

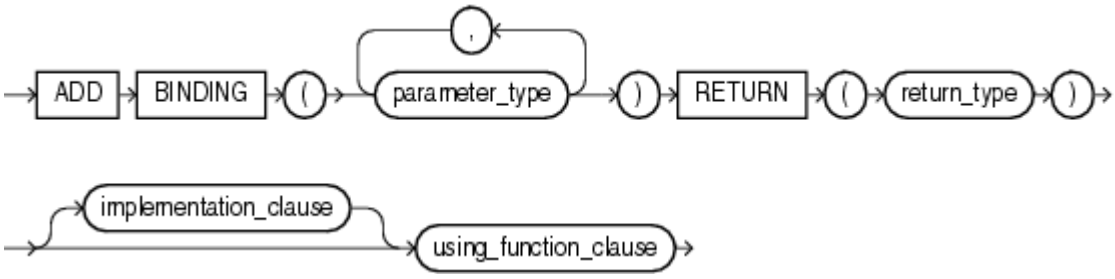
構文

alter_operator::=



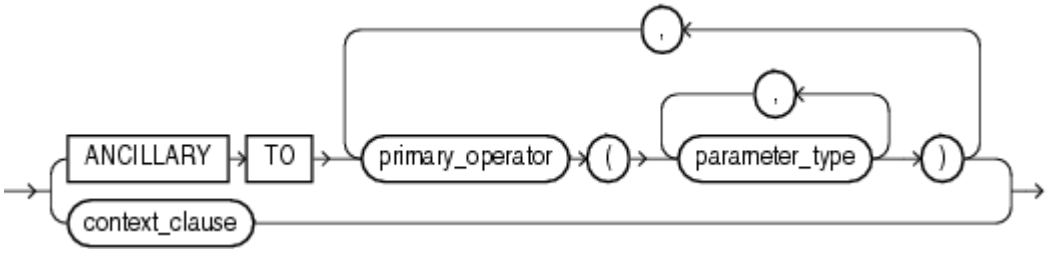
(add_binding_clause::=、drop_binding_clause::=)

add_binding_clause::=



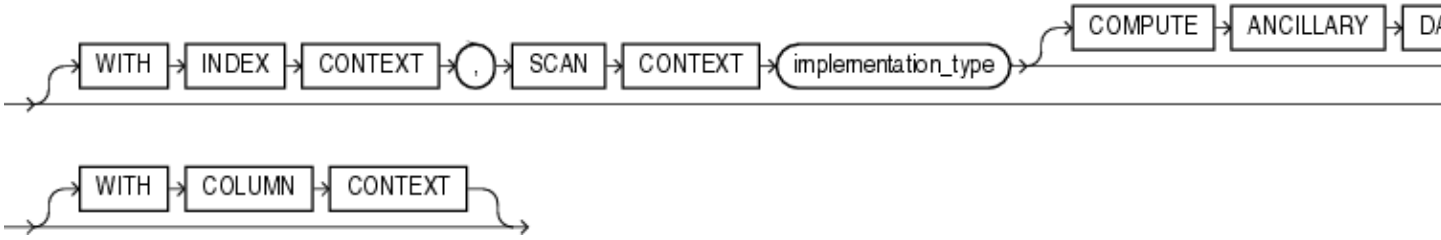
(implementation_clause::=、using_function_clause::=)

implementation_clause::=



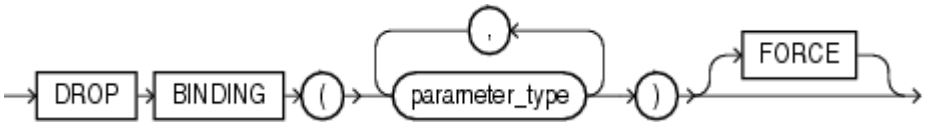
(context_clause::=)

context_clause::=



using_function_clause::=

drop_binding_clause::=



セマンティクス

schema

演算子が含まれているスキーマを指定します。この句を指定しない場合、その演算子は自分のスキーマにあるとみなされます。

operator

変更する演算子の名前を指定します。

add_binding_clause

この句を使用すると、演算子バインドを追加し、パラメータ・データ型および戻り型を指定できます。この演算子の既存のバインドと異なるシグネチャを使用する必要があります。

演算子のバインドが索引タイプに関連付けられており、演算子に別のバインドを追加する場合、Oracle Databaseは新しいバインドと索引タイプの関連付けを自動的に行いません。関連付けを行うには、明示的なALTER INDEXTYPE ... ADD OPERATOR文を発行する必要があります。

implementation_clause

この句のセマンティクスは、CREATE OPERATORおよびALTER OPERATOR文で同じです。詳細は、「CREATE OPERATOR」の「implementation_clause」を参照してください。

context_clause

この句のセマンティクスは、CREATE OPERATORおよびALTER OPERATOR文で同じです。詳細は、「CREATE OPERATOR」の「context_clause」を参照してください。

using_function_clause

この句のセマンティクスは、CREATE OPERATORおよびALTER OPERATOR文で同じです。詳細は、「CREATE OPERATOR」の「using_function_clause」を参照してください。

drop_binding_clause

この句を使用すると、演算子から削除するバインドのパラメータ・データ型のリストを指定できます。バインドに索引タイプや補助演算子バインドなどの依存オブジェクトが含まれる場合、FORCEを指定する必要があります。FORCEを指定すると、そのバインドに依存するすべてのオブジェクトにINVALIDのマークが付きます。依存オブジェクトは、DDL文、DML文または問合せで次に参照された際に再検証されます。

この句を使用して、この演算子に関連付けられた唯一のバインドを削除することはできません。この演算子に関連付けられた唯一のバインドを削除するには、DROP OPERATOR文を使用する必要があります。詳細は、「DROP OPERATOR」を参照してください。

COMPILE

COMPILEを指定すると、演算子を再コンパイルできます。

例

ユーザー定義演算子のコンパイル: 例

次の文は、演算子eq_op (「ユーザー定義演算子の作成: 例」で作成)をコンパイルします。

```
ALTER OPERATOR eq_op COMPILE;
```

ALTER OUTLINE

目的

> ノート:
>
> ストアド・アウトラインは非推奨になりました。ストアド・アウトラインは、下位互換性を保つために今でもサポートされています。ただし、かわりに SQL 計画管理を使用することをお薦めします。SQL 計画管理では、ストアド・アウトラインよりも非常に安定した SQL パフォーマンスを実現する SQL 計画ベースラインが作成されます。
>
> DBMS_SPM パッケージの MIGRATE_STORED_OUTLINE ファンクションまたは Enterprise Manager Cloud Control を使用して、既存のストアド・アウトラインを SQL 計画ベースラインに移行できます。移行が完了したら、ストアド・アウトラインに移行済のマークが付けられ、削除できるようになります。DBMS_SPM パッケージの DROP_MIGRATED_STORED_OUTLINE ファンクションを使用して、システム上のすべての移行済ストアド・アウトラインを削除できます。
>
> 参照: SQL 計画管理の詳細は、『Oracle Database SQL チューニング・ガイド』を参照してください。DBMS_SPM パッケージの詳細は、『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』を参照してください。

ALTER OUTLINE文を使用すると、ストアド・アウトラインの名前を変更したり、ストアド・アウトラインを異なるカテゴリに再度割り当てることができます。また、アウトラインのSQL文をコンパイルし、古いアウトライン・データを現行の条件で作成したアウトラインと置き換えて、ストアド・アウトラインを再生成できます。

関連項目:

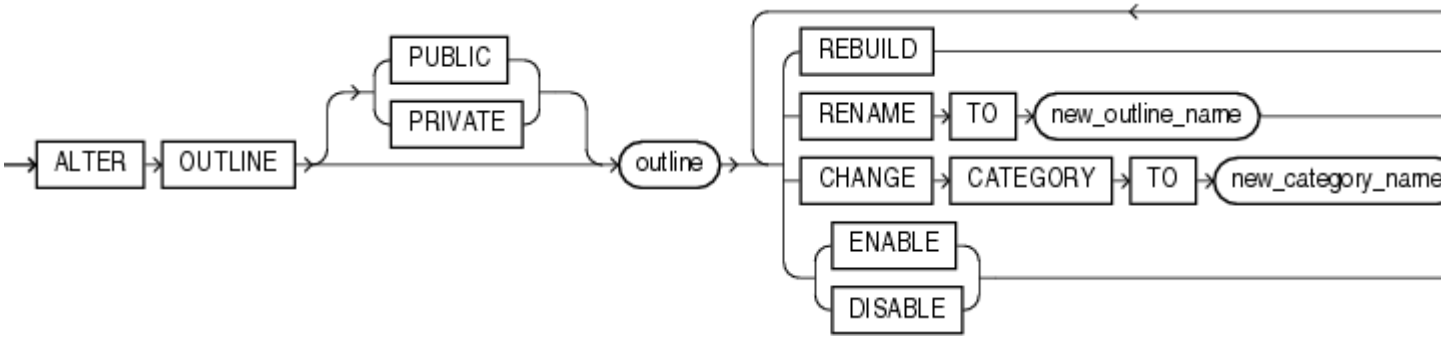
アウトラインの作成については、「CREATE OUTLINE」を参照してください。

前提条件

アウトラインを変更する場合は、ALTER ANY OUTLINEシステム権限が必要です。

構文

alter_outline::=



セマンティクス

PUBLIC | PRIVATE

PUBLICを指定すると、アウトラインのパブリック・バージョンを変更できます。これはデフォルトです。

PRIVATEを指定すると、現行のセッションに対してプライベートで、現行の解析スキーマにデータが格納されているアウトラインを変更できます。

outline

変更するアウトラインの名前を指定します。

REBUILD

REBUILDを指定すると、現行の条件で、outlineの実行計画が再生成されます。

関連項目:

アウトラインの再構築: 例

RENAME TO句

RENAME TO句を使用すると、outlineの値と置き換えるアウトライン名を指定できます。

CHANGE CATEGORY TO句

CHANGE CATEGORY TO句を使用すると、outlineの移動先となるカテゴリ名を指定できます。

ENABLE | DISABLE

この句を使用すると、このアウトラインを選択的に使用可能または使用禁止にできます。デフォルトでは、アウトラインは使用可能になっています。DISABLEキーワードを使用すると、他のアウトラインの使用に影響を与えずに、1つのアウトラインを使用禁止にできます。

例

アウトラインの再構築: 例

次の文は、アウトラインのテキストをコンパイルし、古いアウトライン・データを現行の条件で作成したアウトラインと置き換えて、salariesというストアド・アウトラインを再生成します。

```
ALTER OUTLINE salaries REBUILD;
```

ALTER PACKAGE

目的

パッケージはPL/SQLを使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

ALTER PACKAGE文を使用すると、パッケージ仕様部またはパッケージ本体(あるいはその両方)を明示的に再コンパイルできます。明示的に再コンパイルすることによって、実行時に暗黙的に再コンパイルする必要がなくなり、また、実行時のコンパイル・エラーとパフォーマンス上のオーバーヘッドもなくなります。

パッケージ中のすべてのオブジェクトは、1つの単位として格納されているため、ALTER PACKAGE文によって、すべてのパッケージ・オブジェクトがまとめて再コンパイルされます。ALTER PROCEDURE文またはALTER FUNCTION文を使用して、パッケージ中の一部のプロシージャまたはファンクションを再コンパイルすることはできません。

ノート:

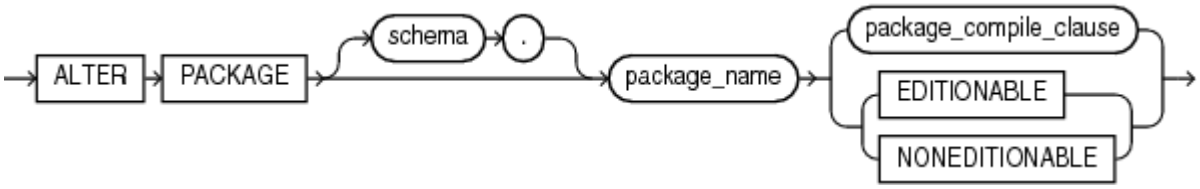
この文では、既存のパッケージの宣言または定義は変更されません。パッケージを再宣言または再定義する場合は、「CREATE PACKAGE」または「CREATE PACKAGE BODY」に OR REPLACE 句を指定します。

前提条件

パッケージを変更するには、パッケージが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY PROCEDUREシステム権限が必要です。

構文

alter_package::=



(package_compile_clause: この句の構文の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。)

セマンティクス

schema

パッケージが含まれているスキーマを指定します。schemaを指定しない場合、パッケージは自分のスキーマ内にあるとみなされます。

package_name

再コンパイルするパッケージの名前を指定します。

package_compile_clause

この句の構文とセマンティクスの詳細およびパッケージの作成とコンパイルの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

EDITIONABLE | NONEDITIONABLE

この句を使用すると、schemaのスキーマ・オブジェクト・タイプPACKAGEのエディションニングが後で有効化されたときに、そのパッケージをエディション・オブジェクトにするか非エディション・オブジェクトにするかを指定できます。デフォルトは、EDITIONABLEです。エディション・オブジェクトと非エディション・オブジェクトの変更の詳細は、『Oracle Database開発ガイド』を参照してください。

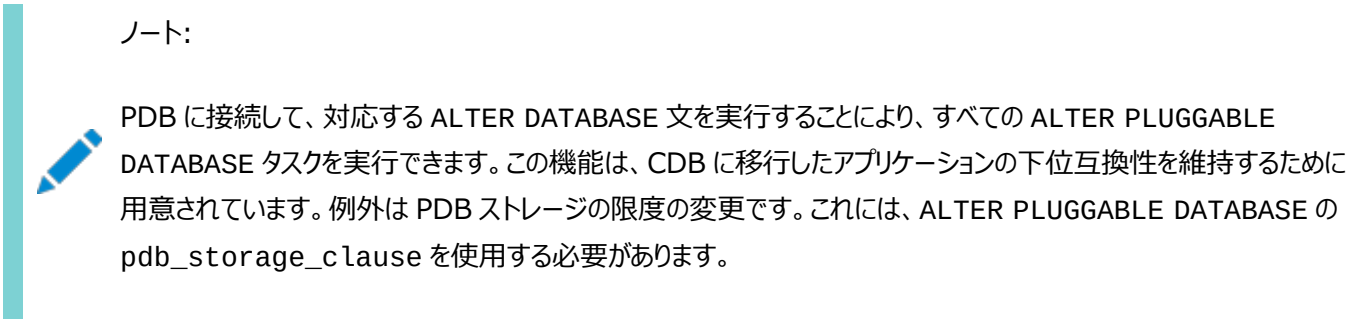
ALTER PLUGGABLE DATABASE

目的

ALTER PLUGGABLE DATABASE文を使用すると、プラガブル・データベース(PDB)を変更できます。PDBには、従来のPDB、アプリケーション・コンテナまたはアプリケーションPDBを指定できます。

この句で実行できるタスクは、次のとおりです。

- PDBをマルチテナント・コンテナ・データベース(CDB)から切断する(pdb_unplug_clauseを使用)。
- PDBの設定を変更する(pdb_settings_clausesを使用)。
- PDBデータファイルをオンラインまたはオフラインにする(pdb_datafile_clauseを使用)。
- PDBをバックアップおよびリカバリする(pdb_recovery_clausesを使用)。
- PDBの状態を変更する(pdb_change_state句を使用)。
- CDB内の複数のPDBの状態を変更する(pdb_change_state_from_root句を使用)。
- アプリケーション・コンテナでアプリケーションに対して操作を実行する(application_clausesを使用)。
- snapshot_clausesを使用してPDBのスナップショットを作成および管理する。



ノート:

PDB に接続して、対応する ALTER DATABASE 文を実行することにより、すべての ALTER PLUGGABLE DATABASE タスクを実行できます。この機能は、CDB に移行したアプリケーションの下位互換性を維持するために用意されています。例外は PDB ストレージの限度の変更です。これには、ALTER PLUGGABLE DATABASE の pdb_storage_clause を使用する必要があります。

関連項目:

PDBの作成の詳細は、「CREATE PLUGGABLE DATABASE」を参照してください。

前提条件

CDBに接続している必要があります。

pdb_unplug_clauseを指定するには、現在のコンテナがルートまたはアプリケーション・ルートである必要があります。また、AS SYSDBAまたはAS SYSOPERとして認証される必要があるとともに、SYSDBAまたはSYSOPER権限(共通に付与されている権限か、ルートと切断対象のPDBのそれぞれでローカルに付与されている権限のいずれか)が必要です。

pdb_settings_clausesを指定するには、現在のコンテナが設定の変更対象のPDBである必要があります。また、ALTER DATABASE権限(共通に付与されている権限か、そのPDBでローカルに付与されている権限のいずれか)が必要です。pdb_logging_clausesまたはRENAME GLOBAL_NAME句を指定するには、RESTRICTED SESSION権限(共通に付与されている権限か、名前を変更するPDBでローカルに付与されている権限のいずれか)も必要です。PDBはREAD WRITE RESTRICTEDモードである必要があります。

pdb_datafile_clauseを指定するには、現在のコンテナが、データファイルをオンラインまたはオフラインにするPDBである必要があり、ALTER DATABASE権限(共通に付与されている権限か、そのPDBでローカルに付与されている権限のいずれか)

が必要です。

pdb_recovery_clausesを指定するには、現在のコンテナがバックアップまたはリカバリの実行対象のPDBである必要があります。また、ALTER DATABASE権限(共通に付与されている権限か、そのPDBでローカルに付与されている権限のいずれか)が必要です。

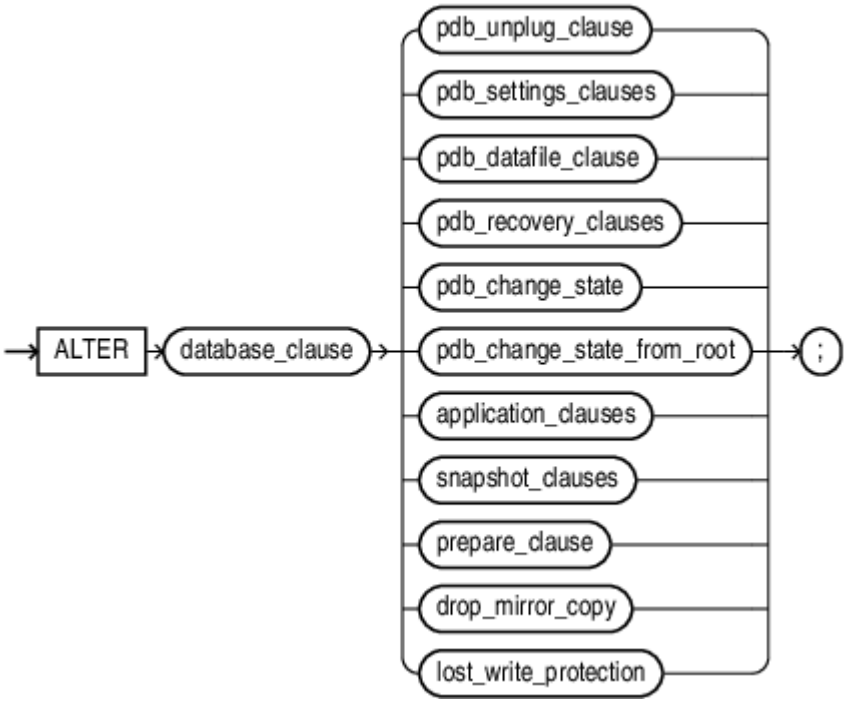
pdb_change_state句を指定するには、現在のコンテナが状態の変更対象のPDBである必要があります。また、AS SYSBACKUP、AS SYSDBA、AS SYSDGまたはAS SYSOPERとして認証される必要があります。

pdb_change_state_from_root句を指定するには、現在のコンテナがルートまたはアプリケーション・ルートである必要があります。また、AS SYSBACKUP、AS SYSDBA、AS SYSDGまたはAS SYSOPERとして認証される必要があるとともに、SYSBACKUP、SYSDBA、SYSDGまたはSYSOPER権限(共通に付与されている権限か、ルートまたはアプリケーション・ルートと状態の変更対象のPDBのそれぞれでローカルに付与されている権限のいずれか)が必要です。

application_clausesを指定するには、現在のコンテナがアプリケーション・コンテナである必要があります。また、AS SYSBACKUPまたはAS SYSDBAとして認証される必要があるとともに、SYSBACKUPまたはSYSDBA権限(共通に付与されている権限か、アプリケーション・ルートとアプリケーション操作を実行するアプリケーションPDBのそれぞれでローカルに付与されている権限のいずれか)が必要です。

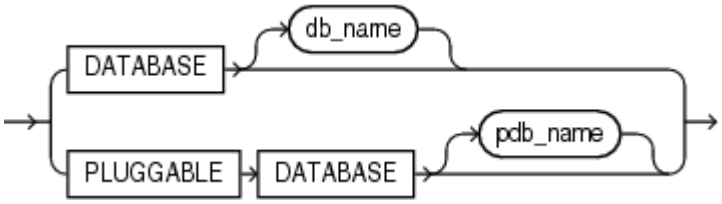
構文

alter_pluggable_database::=



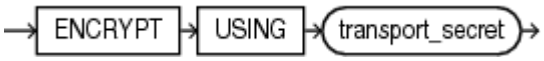
(pdb_unplug_clause::=、pdb_settings_clauses::=、pdb_datafile_clause::=、pdb_recovery_clauses、pdb_change_state::=、pdb_change_state_from_root::=、application_clauses::=)

database_clause::=

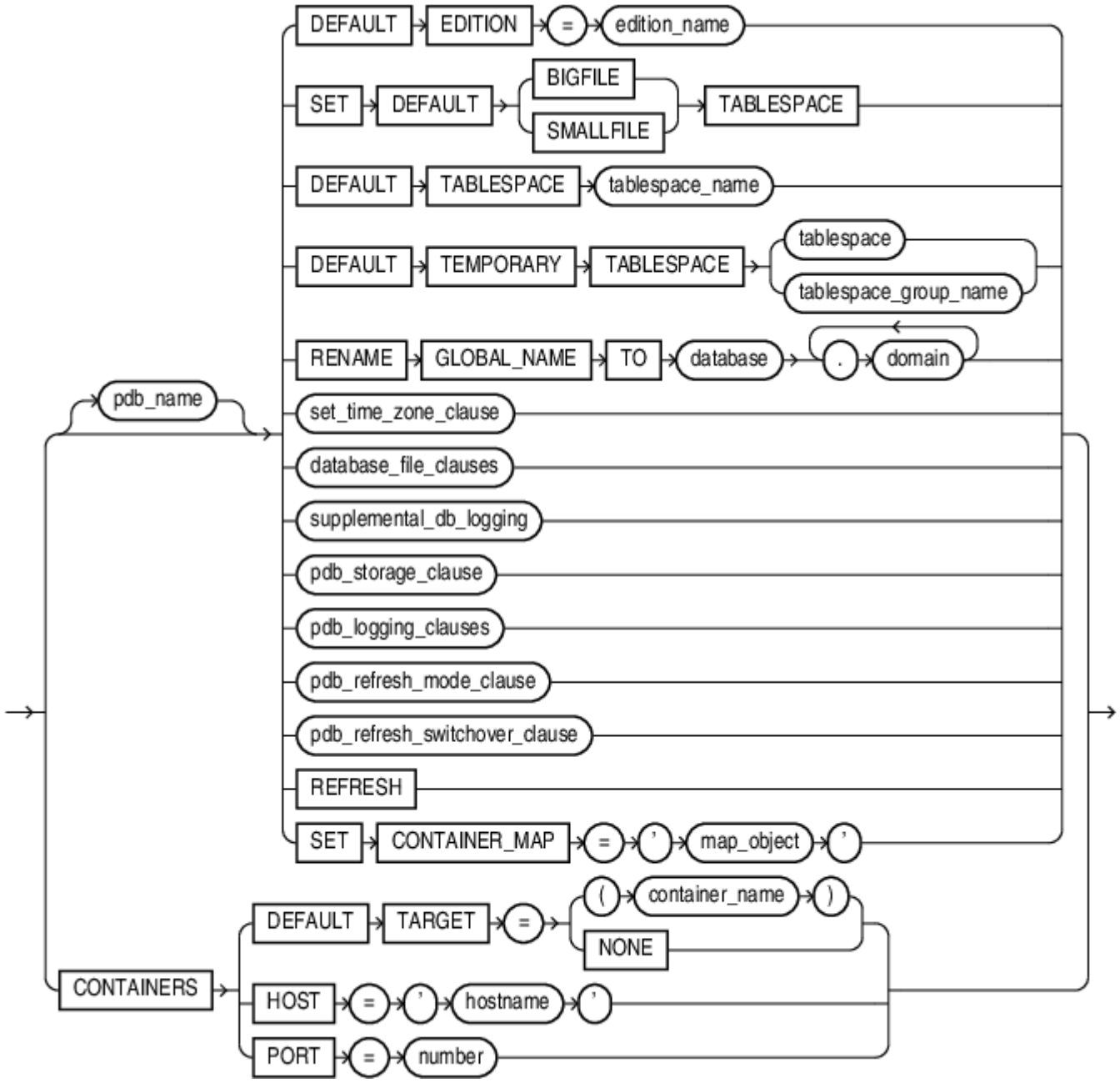


pdb_unplug_clause::=

pdb_unplug_encrypt::=

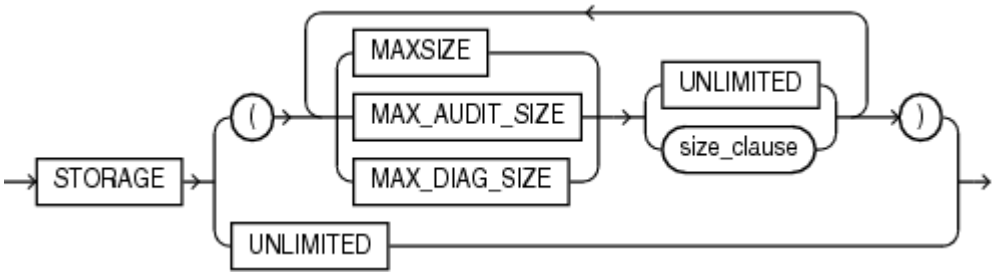


pdb_settings_clauses::=



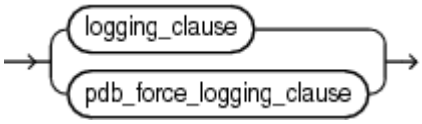
(set_time_zone_clause::=、database_file_clauses::=、supplemental_db_logging::=、pdb_storage_clause::=、pdb_logging_clauses::=、pdb_refresh_mode_clause::=)

pdb_storage_clause::=

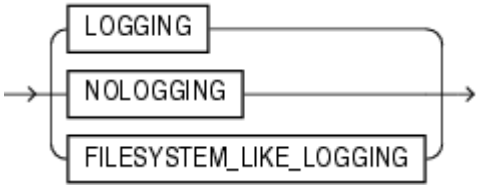


(size_clause::=)

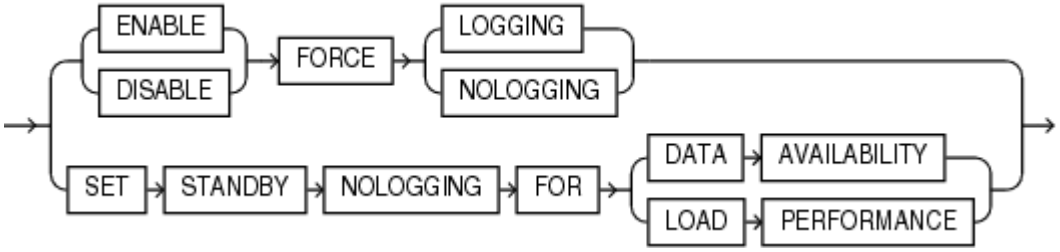
pdb_logging_clauses::=



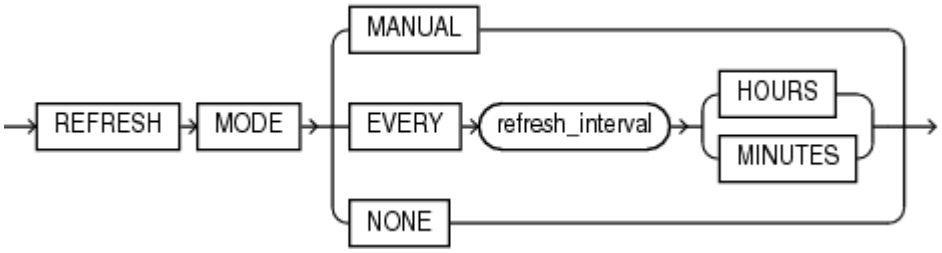
logging_clause::=



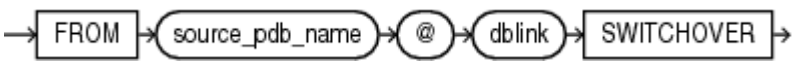
pdb_force_logging_clause::=



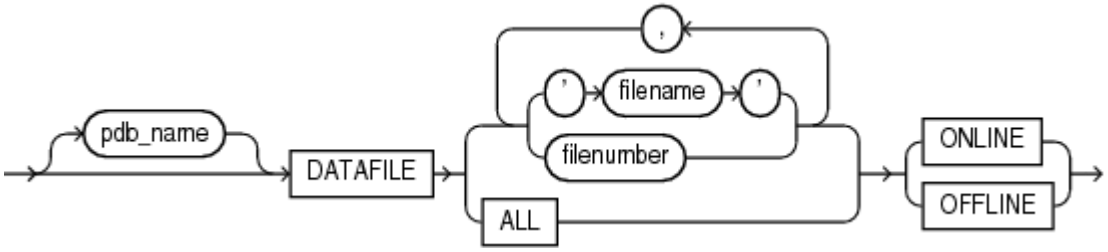
pdb_refresh_mode_clause::=



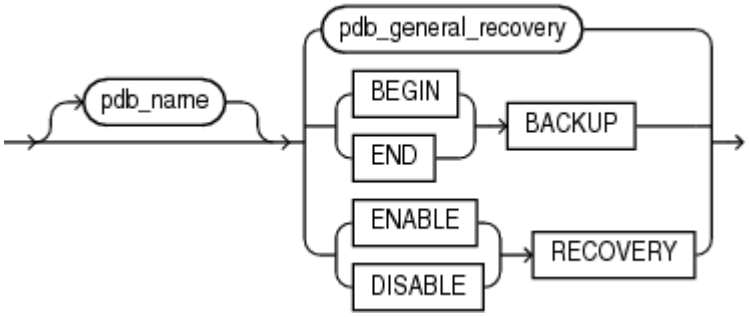
pdb_refresh_switchover_clause ::=



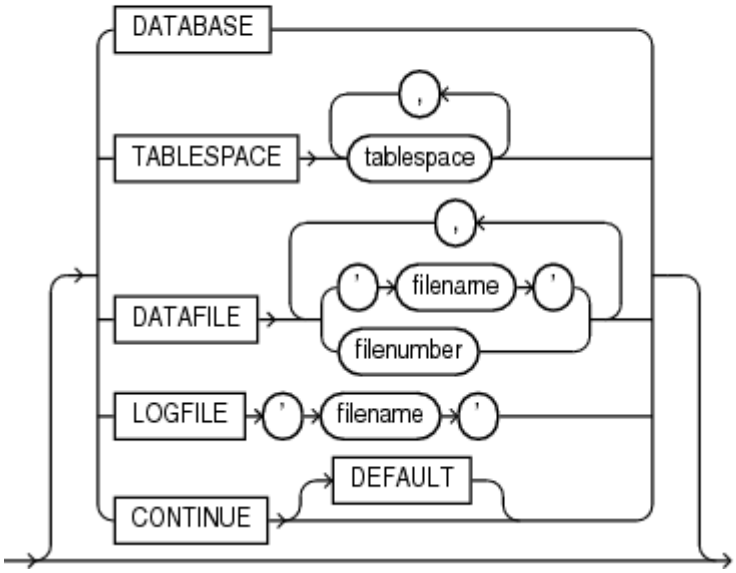
pdb_datafile_clause::=



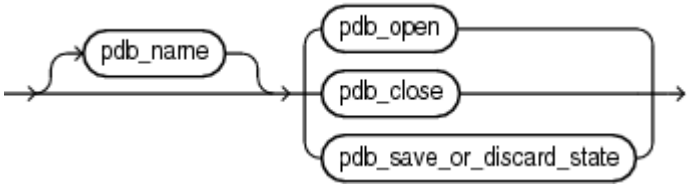
pdb_recovery_clauses



pdb_general_recovery::=

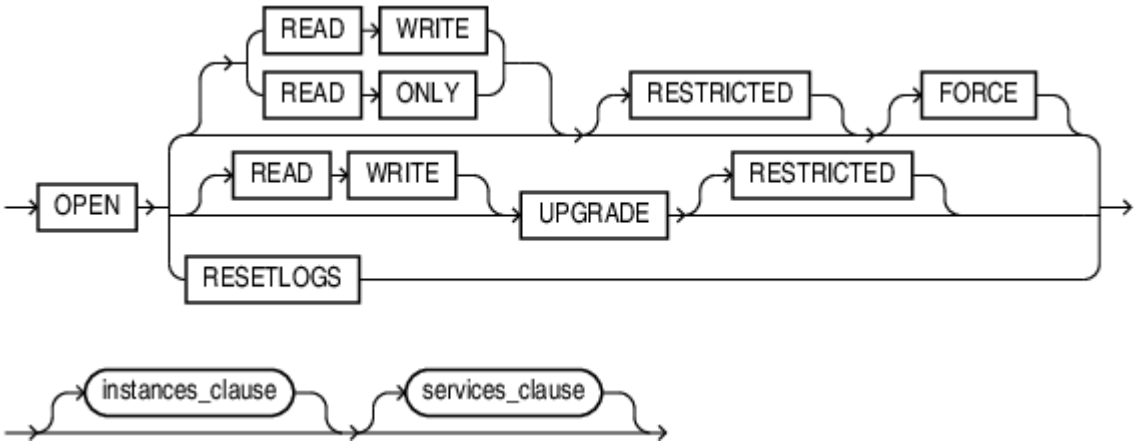


pdb_change_state::=

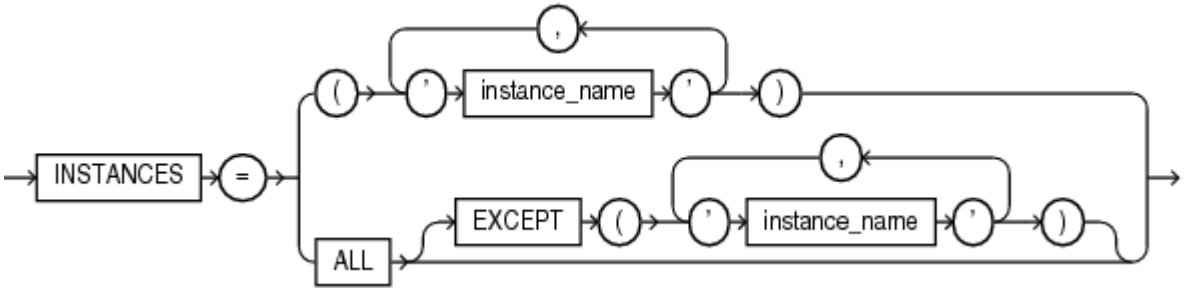


(pdb_open::=、pdb_close::=、pdb_save_or_discard_state::=)

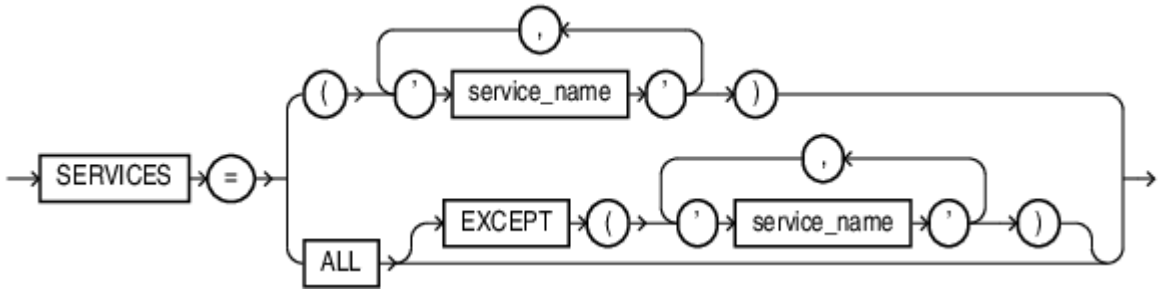
pdb_open::=



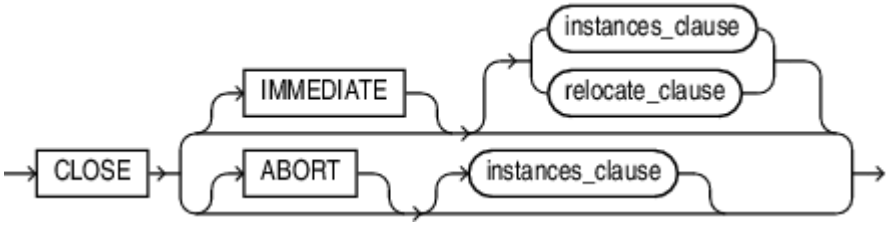
instances_clause::=



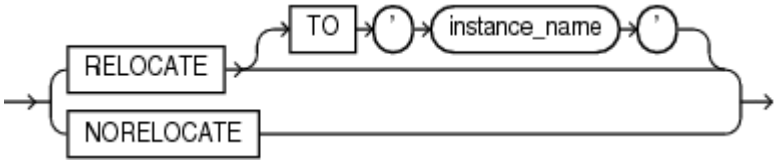
services_clause::=



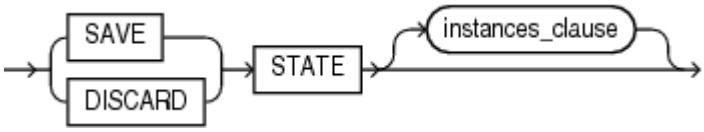
pdb_close::=



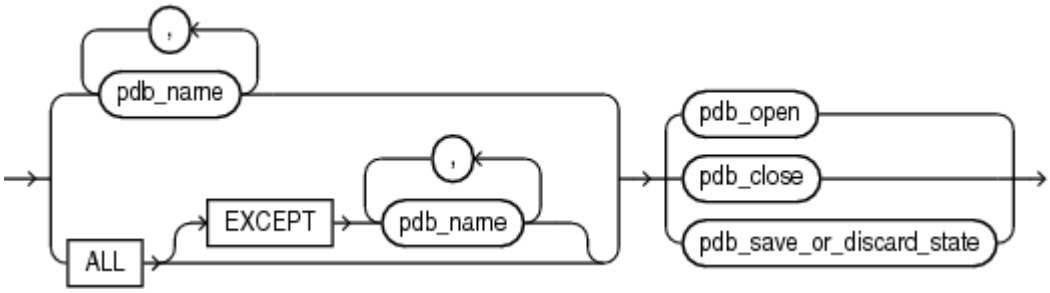
relocate_clause::=



pdb_save_or_discard_state::=

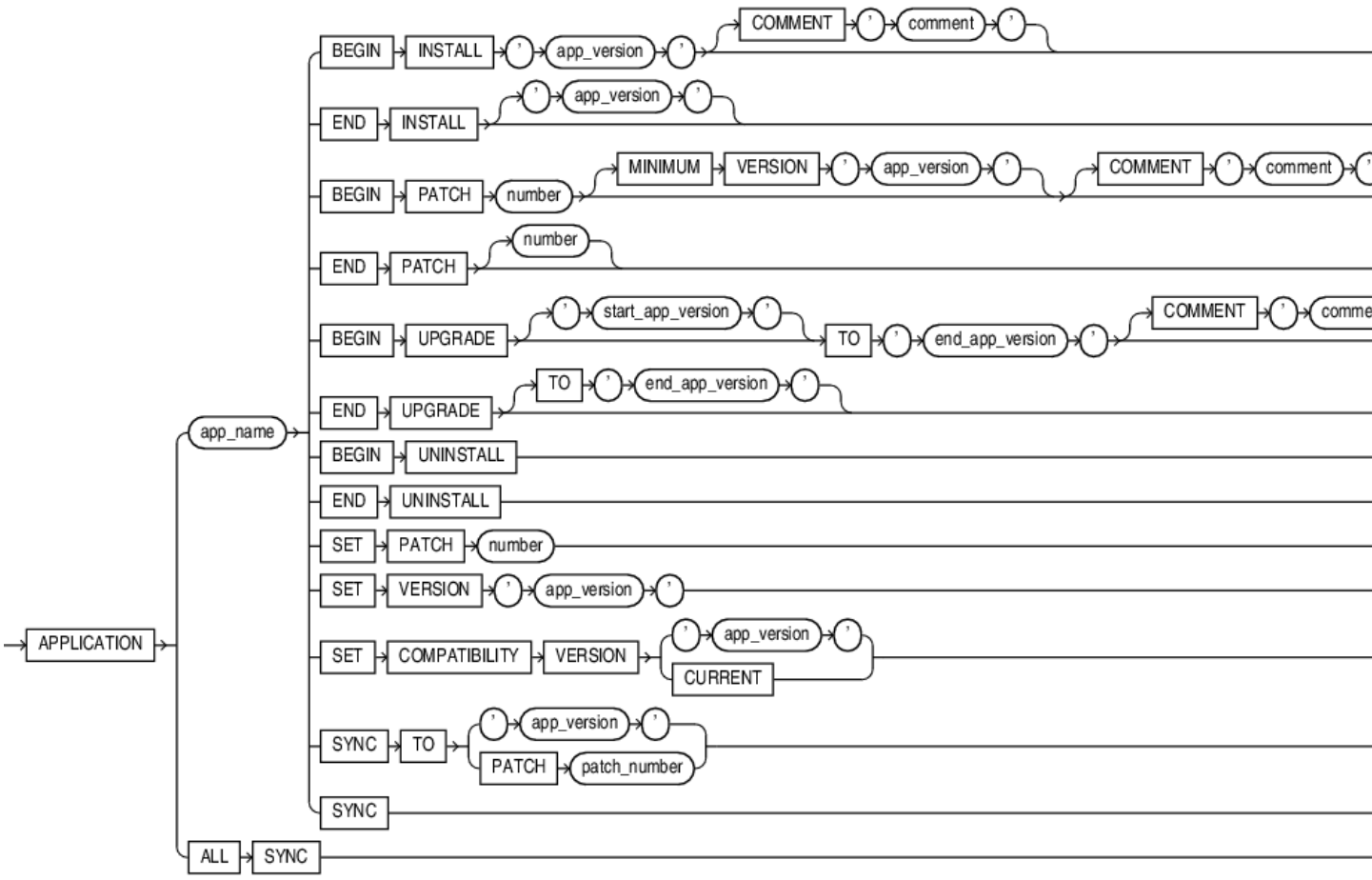


pdb_change_state_from_root::=

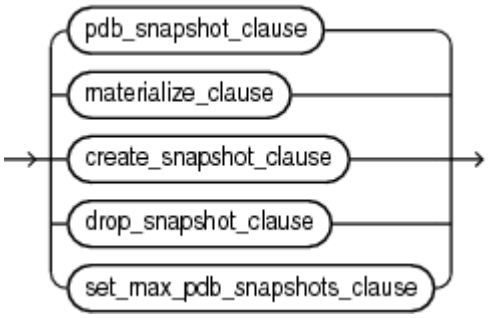


(pdb_open::=、pdb_close::=、pdb_save_or_discard_state::=)

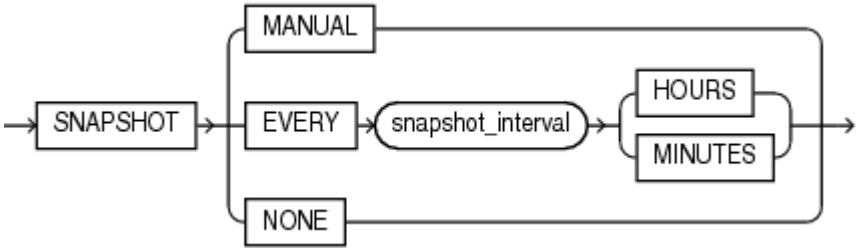
application_clauses::=



snapshot_clauses ::=

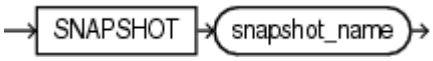


pdb_snapshot_clause ::=

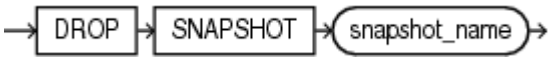


materialize_clause::=

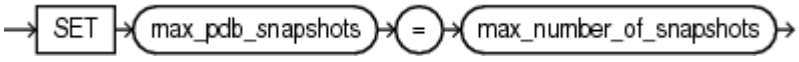
create_snapshot_clause::=



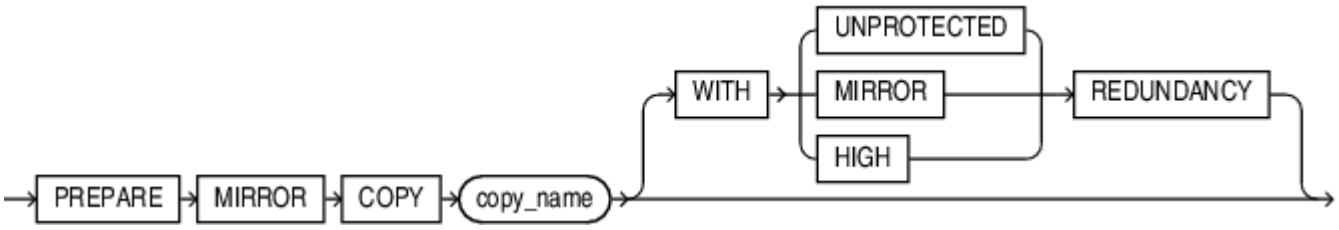
drop_snapshot_clause::=



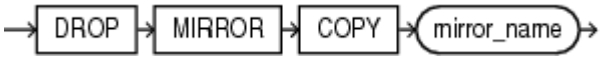
set_max_pdb_snapshots::=



prepare_clause::=



drop_mirror_copy::



lost_write_protection ::=

ALTER PLUGGABLE DATABASE文でのlost_write_protection句の使用方法は、ALTER DATABASE文と同じです。構文の詳細は、ALTER DATABASEを参照してください。

セマンティクス

database_clause

コンテナ・データベースのPLUGGABLE DATABASEオプションを指定します。

pdb_name

変更するデータベースの名前を指定します。db_nameを省略した場合は、初期化パラメータDB_NAMEの値によって特定されたデータベースが変更されます。なお、データベースの制御ファイルが初期化パラメータCONTROL_FILESに指定されている場合にのみ、そのデータベースを変更できます。データベース識別子は、Oracle Netのデータベース指定とは関係ありません。

pdb_unplug_clause

この句を使用すると、CDBからPDBを切断できます。PDBを切断すると、PDBに関する情報がオペレーティング・システム上のファイルに保存されます。その後、このファイルを使用して、PDBをCDBに接続できます。

pdb_nameには、切断するPDBの名前を指定します。PDBはクローズされている必要があります。つまり、オープン・モードは、MOUNTEDである必要があります。Oracle Real Application Clusters(Oracle RAC)環境では、すべてのOracle RACインスタンスでそのPDBがクローズされている必要があります。

filenameには、PDBの情報を保存するオペレーティング・システム・ファイルのフルパス名を指定します。指定するファイル名によって、保存される情報のタイプおよび保存方法が決まります。

- 拡張子.xmlで終わるファイル名を指定すると、PDBに関するメタデータを含むXMLファイルが作成されます。その後、XMLファイルとPDBのデータファイルを新しい場所にコピーして、PDBをCDBに接続する際にXMLファイル名を指定します。この場合、PDBのデータファイルを別個にコピーする必要があります。
- 拡張子.pdbで終わるファイル名を指定すると、.pdbアーカイブ・ファイルが作成されます。これは、PDBに関するメタデータを含むXMLファイルとPDBのデータファイルを含む、圧縮されたファイルです。その後、この単一のアーカイブ・ファイルを新しい場所にコピーして、PDBをCDBに接続する際にアーカイブ・ファイル名を指定します。これにより、PDBのデータファイルを別個にコピーする必要がなくなります。PDBの接続時に.pdbアーカイブ・ファイルを使用する場合、このファイルはPDBの接続時に抽出され、PDBのファイルは.pdbアーカイブ・ファイルと同じディレクトリに配置されます。

切断後、そのPDBは、オープン・モードMOUNTED、ステータスUNPLUGGEDとしてCDBに留まります。切断されたPDBに対して実行できる操作は、DROP PLUGGABLE DATABASEのみです。この操作により、PDBはCDBから削除されます。同じCDBまたは別のCDBにPDBを接続するには、先にそのPDBを削除する必要があります。

関連項目:

- PDBの切断の詳細は、『Oracle Database管理者ガイド』を参照してください。
- CDBへのPDBの接続の詳細は、「CREATE PLUGGABLE DATABASE」の「create_pdb_from_xml」句を参照してください。

pdb_unplug_encrypt

このコマンドを実行するにはSYSKM権限が必要です。

統合PDB

- ENCRYPT USING transport_secretはオプションです。
- TDEが使用されている場合は、この句を指定する必要があります。TDEが使用されていない場合、この文によってエラーORA-46680:master keys of the container database must be exportedがスローされます。
- TDEが使用されている場合、ウォレットがROOTでオープンしている必要があります。
- キーは、指定されたtransport secretを使用して暗号化され、.XMLファイルまたはアーカイブ・ファイルにエクスポートされます。

XMLメタデータ・ファイルへのPDBの切断: 例

```
ALTER PLUGGABLE DATABASE CDB1_PDB2 UNPLUG INTO '/tmp/cdb1_pdb2.xml' ENCRYPT USING
transport_secret
```

アーカイブ・ファイルへのPDBの切断: 例

```
ALTER PLUGGABLE DATABASE CDB1_PDB1_1 UNPLUG INTO '/tmp/CDB1_PDB1_1.pdb' ENCRYPT USING
transport_secret
```

分離モードのPDBの場合、ENCRYPT USING transport_secretを指定する必要はありません。PDBのウォレット・ファイルはXMLファイルからのプラガブル・データベースの作成時にコピーされるため、これは必須ではありません。アーカイブ・ファイルとしてPDBを切断する場合、PDBのウォレット・ファイルは、.pdb拡張子を持つ圧縮アーカイブに追加されます。

ewallet.p12ファイルがすでに宛先に存在している場合、バックアップが自動的に開始されます。バックアップ・ファイルの形式はewallet_PLGDB_2017090517455564.p12です。

pdb_settings_clauses

これらの句を使用すると、PDBの様々な設定を変更できます。

pdb_name

オプションでpdb_nameを使用して、設定を変更するPDBの名前を指定できます。

DEFAULT EDITION句

この句を使用すると、指定したエディションをPDBのデフォルト・エディションとして設定できます。この句のセマンティクスの詳細は、「ALTER DATABASE」の「DEFAULT EDITION句」を参照してください。

SET DEFAULT TABLESPACE句

この句を使用すると、以降にPDBで作成する表領域のデフォルト・タイプを指定または変更できます。この句のセマンティクスの詳細は、「ALTER DATABASE」の「SET DEFAULT TABLESPACE句」を参照してください。

DEFAULT TABLESPACE句

この句を使用すると、PDBのデフォルトの永続表領域を構築または変更できます。この句のセマンティクスの詳細は、「ALTER DATABASE」の「DEFAULT TABLESPACE句」を参照してください。

DEFAULT TEMPORARY TABLESPACE句

この句を使用すると、PDBのデフォルトの一時表領域を、新しい表領域または表領域グループに変更できます。この句のセマンティクスの詳細は、「ALTER DATABASE」の「DEFAULT [LOCAL] TEMPORARY TABLESPACE句」を参照してください。

RENAME GLOBAL_NAME TO句

この句を使用して、PDBのグローバル名を変更できます。新しいグローバル名は、CDB内で一意にする必要があります。Oracle Real Application Clusters(Oracle RAC)データベースの場合は、PDBを現在のインスタンスのみでREAD WRITE RESTRICTEDモードで開く必要があります。その他すべてのインスタンスでは、PDBをクローズする必要があります。この句のセマンティクスの詳細は、「ALTER DATABASE」の「RENAME GLOBAL_NAME句」を参照してください。

ノート:

PDB のグローバル名を変更するときには必ず、その PDB への接続に使用しているデータベース・サービスの PLUGGABLE DATABASE プロパティを変更してください。

set_time_zone_clauses

この句を使用すると、PDBのタイムゾーン設定を変更できます。この句のセマンティクスの詳細は、ALTER DATABASEのset_time_zone_clauseを参照してください。

database_file_clauses

この句を使用すると、PDBのデータファイルと一時ファイルを変更できます。この句のセマンティクスの詳細は、ALTER DATABASEのdatabase_file_clausesを参照してください。

supplemental_db_logging

これらの句を使用すると、PDBのログ・ストリームへのサプリメンタル・データを追加する(または追加を停止する)ようにOracle Databaseに指示できます。

- ADD SUPPLEMENTAL LOG句を指定すると、PDBのログ・ストリームにサプリメンタル・データを追加できます。この句を発行するには、ALTER DATABASE ... ADD SUPPLEMENTAL LOG ...文を使用してCDBルートのサプリメンタル・ロギングを有効にする必要があります。PDBに指定するサプリメンタル・ロギングのレベルは、CDBルートのログと一致する必要はありません。つまり、CDBルートのサプリメンタル・ロギングを有効にするときに指定された句に関係なく、PDBに句DATA、supplemental_id_key_clause、またはsupplemental_plsql_clauseのいずれかを指定できます。
- DROP SUPPLEMENTAL LOG句を指定すると、PDBのログ・ストリームへのサプリメンタル・データの追加を停止できます。

ALTER PLUGGABLE DATABASEのADD SUPPLEMENTAL LOG DATA SUBSET DATABASE REPLICATIONは、PDBで影響の少ない最小サプリメンタル・ロギングを有効化します。

- このDDLは、プラガブル・データベースでのみ実行できます。
- このDDLを実行できるのは、enable_goldengate_replicationパラメータがTRUEで、データベースが19.0以上と互換のときだけです。
- このコマンドを実行するには、CDB$ROOTで最小サプリメンタル・ロギングを有効にする必要があります。
- このDDLを実行すると、プラガブル・データベースに対して、最小サプリメンタル・ロギングの影響が少なくなります。SYS.PROP$が更新され、影響の少ない最小サプリメンタル・ロギングが、このプラガブル・データベースのPDBレベルで有効であることが示されます。

ALTER PLUGGABLE DATABASEのDROP SUPPLEMENTAL LOG DATA SUBSET DATABASE

REPLICATIONは、PDBで影響の少ない最小サプリメンタル・ロギングを有効化します。

- このDDLは、プラガブル・データベースでのみ実行できます。
- このDDLを実行できるのは、enable_goldengate_replicationパラメータがTRUEで、データベースが19.0以上と互換のときだけです。
- このコマンドを実行するには、CDB$ROOTで最小サプリメンタル・ロギングを有効にする必要があります。
- SYS.PROP$が更新され、サブセット・データベース・レプリケーションに対するサプリメンタル・ロギングが、このプラガブル・データベースのPDBレベルで無効になることが示されます。サブセット・データベース・レプリケーションに対するサプリメンタル・ロギングがCDB$ROOT (CDBレベル)でも無効の場合、このプラガブル・データベースでは影響の少ない最小サプリメンタル・ロギングが無効になります。

この句のセマンティクスの詳細は、ALTER DATABASEのsupplemental_db_loggingを参照してください。

pdb_storage_clause

この句を使用して、PDBのストレージ制限を変更できます。

この句のセマンティクスは、CREATE PLUGGABLE DATABASEのpdb_storage_clauseと同じですが、次が追加されます。

- MAXSIZE size_clauseを指定する場合は、size_clauseに、PDBの既存の表領域の合計サイズ以上の値を指定する必要があります。それ以外の場合はエラーが発生します。
- MAX_AUDIT_SIZE size_clauseを指定する場合、size_clauseに指定する値は、PDBで既存のあふれた統合監査OSファイル(.bin形式)で使用される記憶域の容量以上である必要があります。それ以外の場合はエラーが発生します。
- MAX_DIAG_SIZE size_clauseを指定する場合、size_clauseに指定する値は、PDBにより現在使用されている自動診断リポジトリ(ADR)での診断用の記憶域の容量以上である必要があります。そうでない場合はエラーが発生します。

pdb_logging_clauses

この句を使用すると、PDBのロギング特性を設定または変更できます。

logging_clause

この句を使用して、PDB内で以降に作成する表領域のデフォルトのロギング属性を変更します。この句のセマンティクスは、CREATE PLUGGABLE DATABASEのlogging_clauseと同じです。

pdb_force_logging_clause

この句を使用すると、PDBを4つのロギング・モードのいずれかに設定したり、そのモードから戻したりできます。

強制ロギング・モードでは、一時表領域および一時セグメントの変更を除くPDBのすべての変更の記録をデータベースに指示します。強制ロギングなしモードでは、PDBの変更を記録しないようデータベースに指示します。

STANDBY NOLOGGINGは、ロギングなしで実行できる操作をログに記録しないようにデータベースに対して指示します。データベースは、操作によって作成されたデータ・ブロックをData Guard構成内の資格を持つ各スタンバイ・データベースに送信します。これにより、通常、それらのスタンバイに無効なブロックがなくなります。

CDB全体の強制ロギング・モードは、他のどの設定より優先されます。PDBレベルの強制ロギング・モードおよび強制ロギングなしモードは、PDBの個々の表領域に指定するLOGGING、NOLOGGINGまたはFORCE LOGGING設定あるいはPDBの個々のデータベース・オブジェクトに指定するLOGGINGまたはNOLOGGING設定よりも優先され、これらの設定には影響されません。

- ENABLE FORCE LOGGINGを指定して、強制ロギング・モードのPDBを配置します。PDBが現在強制ロギングなしモードである場合、この句を指定するとエラーになります。最初にDISABLE FORCE NOLOGGINGを指定する必要があります。
- DISABLE FORCE LOGGINGを指定して、PDBの強制ロギング・モードを解除します。PDBが現在強制ロギング・モードでない場合、この句を指定するとエラーになります。
- ENABLE FORCE NOLOGGINGを指定して、強制ロギングなしモードのPDBを配置します。PDBが現在強制ロギング・モードである場合、この句を指定するとエラーになります。最初にDISABLE FORCE LOGGINGを指定する必要があります。ログに記録されない操作では、CDBにスタンバイ・ロギングなしモードが設定されている場合でも、従来の無効化REDOが使用されます。
- DISABLE FORCE NOLOGGINGを指定して、PDBの強制ロギングなしモードを解除します。PDBが現在強制ロギングなしモードでない場合、この句を指定するとエラーになります。
- PDBをロード・パフォーマンスのスタンバイ・ロギングなしモードにするには、SET STANDBY NOLOGGING FOR LOAD PERFORMANCEを指定します。このモードでは、ロード・プロセスの速度が低下しない場合には、ログに記録されないタスクの一部としてロードされたデータは、プライベート・ネットワーク接続を介して資格のあるスタンバイに送信されます。低下する場合には、データは送信されず、各スタンバイで無効化REDOが発生するとプライマリから自動的にフェッチされ、データ・ブロックが受信されるまで再試行されます。
- PDBをデータ可用性のスタンバイ・ロギングなしモードにするには、SET STANDBY NOLOGGING FOR DATA AVAILABILITYを指定します。このモードでは、ログに記録されないタスクの一部としてロードされたデータは、ネットワーク接続を介して、またはネットワーク接続が失敗する場合にはREDO内のブロック・イメージを介して、資格のあるスタンバイに送信されます。つまり、このモードでは、ネットワーク接続または関連するプロセスによってプライベート・ネットワーク接続を介したデータの送信が妨げられる場合には、ロードはログに記録する方法で行うように切り替わります。

スタンバイ・ロギングなしモードの場合、資格のあるスタンバイは、読取りのためにオープンされ、管理リカバリを実行しており、スタンバイREDOログでREDOを受信しているスタンバイです。

この句は、logging_clauseで指定されたPDBのデフォルトのLOGGINGまたはNOLOGGINGモードを変更しません。

pdb_refresh_mode_clause

この句を使用して、PDBのリフレッシュ・モードを変更します。この句は、リフレッシュ可能なPDB、つまり現在のリフレッシュ・モードがMANUALまたはEVERY refresh_interval MINUTESまたはHOURSのPDBに対してのみ指定できます。PDBは、手動リフレッシュから自動リフレッシュに、または自動リフレッシュから手動リフレッシュに切り替えることができます。この句を使用して、自動リフレッシュの間隔(分単位)を変更することもできます。PDBを手動リフレッシュまたは自動リフレッシュからリフレッシュなしに切り替えることはできますが、リフレッシュ可能でないPDBで手動リフレッシュまたは自動リフレッシュを有効にすることはできません。この句のセマンティックスの詳細は、CREATE PLUGGABLE DATABASEのドキュメントの「pdb_refresh_mode_clause」を参照してください。

REFRESH

この句を指定すると、リフレッシュ可能なPDB、つまり現在のリフレッシュ・モードがMANUALまたはEVERY number MINUTESのPDBの手動リフレッシュを実行できます。PDBはクローズされている必要があります。リフレッシュ可能なPDBの詳細は、「CREATE PLUGGABLE DATABASE」の「pdb_refresh_mode_clause」を参照してください。

pdb_refresh_switchover_clause

この句を使用して、リフレッシュ可能なクローンPDBとプライマリPDBの間でロールを逆にします。この句により、リフレッシュ可能なクローンPDBは、読み書き両用モードでオープンできるプライマリPDBになります。元のプライマリPDBはリフレッシュ可能なクロー

ンになります。

- このコマンドは、プライマリPDBから実行する必要があります。
- この文を発行するときに、REFRESH MODE NONEは指定できません。
- dblinkは、リフレッシュ可能なクローンPDBが現在置かれているCDBのルートを指す必要があります。
- この操作の後、現在のPDBはリフレッシュ可能なクローンになり、読取り専用モードでのみオープンできるようになります。
- リフレッシュ可能なクローンが別のCDB内に存在している場合、データベース・リンク・ユーザーはプライマリPDBに存在している必要があります。

SET CONTAINER_MAP

この句を使用すると、アプリケーション・コンテナのCONTAINER_MAPデータベース・プロパティを指定できます。現在のコンテナはアプリケーション・ルートである必要があります。map_objectの形式は[schema.]tableです。schemaには、tableを含むスキーマを指定します。schemaを指定しない場合、表は自分のスキーマ内にあるとみなされます。tableには、レンジ、リストまたはハッシュ・パーティション化された表を指定します。

CONTAINERS DEFAULT TARGET

この句を使用すると、アプリケーション・コンテナ内のDML文のデフォルト・コンテナを指定できます。アプリケーション・ルートに接続している必要があります。

- container_nameには、デフォルト・コンテナの名前を指定します。デフォルト・コンテナには、アプリケーション・ルートまたはアプリケーションPDBを含む、アプリケーション・コンテナ内の任意のコンテナを指定できます。指定できるデフォルト・コンテナは1つのみです。
- NONEを指定した場合は、デフォルト・コンテナはCDBルートです。これはデフォルトです。

WHERE句でコンテナを指定せずにDML文がアプリケーション・ルートで発行されている場合、DML文はアプリケーション・コンテナのデフォルト・コンテナに影響します。

CONTAINERS HOSTおよびPORT

プロキシPDBから参照するPDBを作成する場合は、HOST句およびPORT句を使用します。このタイプのPDBは、参照先PDBと呼ばれます。

次の文はPDB内で実行できます。

```
ALTER PLUGGABLE DATABASE CONTAINERS HOST='myhost.example.com';

ALTER PLUGGABLE DATABASE CONTAINERS PORT=1599;
```

次の文は、CDBルート、アプリケーション・ルートまたはPDB内で実行できます。

```
ALTER PLUGGABLE DATABASE <pdbname> CONTAINERS HOST='myhost.example.com';

ALTER PLUGGABLE DATABASE <pdbname> CONTAINERS PORT=1599;
```

pdbnameは、次の条件を満たす必要があります。

- この文がアプリケーション・ルートで実行される場合、pdbnameはアプリケーション・ルートの名前またはそのアプリケーションPDBのいずれかの名前と一致する必要があります。
- 文がCDBルートで実行される場合、pdbnameはCDB内のいずれかのPDBの名前と一致する必要があります。

- 文がPDBで実行される場合、pdbnameは現在のPDBの名前と一致する必要があります。

pdb_datafile_clause

この句を使用して、PDBに関連付けられたデータファイルをオンラインまたはオフラインにできます。この句を発行するときには、そのPDBがクローズされている必要があります。

- pdb_nameには、PDBの名前を指定します。現在のコンテナがそのPDBである場合は、pdb_nameは省略できます。
- DATAFILE句では、オンラインまたはオフラインにするデータファイルを指定できます。filenameまたはfilenumberを使用して、特定のデータファイルを名前または番号によって識別します。V$DATAFILE動的パフォーマンス・ビューのNAME列とFILE#列の問合せを実行することによって、データファイルの名前と番号を表示できます。そのPDBに関連付けられたすべてのデータファイルを指定するには、ALLを使用します。
- データファイルをオンラインにする場合はONLINEを指定し、データファイルをオフラインにする場合はOFFLINEを指定します。

pdb_recovery_clauses

pdb_recovery_clausesを使用して、PDBをバックアップおよびリカバリできます。

pdb_name

オプションでpdb_nameを使用して、バックアップまたはリカバリするPDBの名前を指定できます。

pdb_general_recovery

この句を指定すると、PDB、スタンバイ・データベース、または指定した表領域やファイルのメディア・リカバリを制御できます。pdb_general_recovery句は、ALTER DATABASEのgeneral_recovery句と同じセマンティクスを持ちます。詳細は、ALTER DATABASEのgeneral_recovery句を参照してください。

BACKUP句

この句を使用すると、PDB内のすべてのデータファイルをオンライン・バックアップ・モード(ホット・バックアップ・モードともいう)にしたり、このモードから戻すことができます。これらの句についてのセマンティクスは、ALTER PLUGGABLE DATABASEおよびALTER DATABASEと同じです。詳細は、「ALTER DATABASE」の「BACKUP句」を参照してください。

RECOVERY句

これらの句を使用して、リカバリのPDBを有効化または無効化します。PDBはクローズされている必要があります。つまり、オープン・モードは、MOUNTEDである必要があります。

- ENABLE RECOVERYを指定して、PDBに属するすべてのデータファイルをオンラインにし、リカバリのPDBを有効化します。
- DISABLE RECOVERYを指定して、PDBに属するすべてのデータファイルをオフラインにし、リカバリのPDBを無効化します。

関連項目:

RECOVERY句の詳細は、『Oracle Data Guard概要および管理』を参照してください。

pdb_change_state

この句を使用すると、PDBの状態またはオープン・モードを変更できます。表11-2は、PDBのオープン・モードを示しています。

- オープン・モードをREAD WRITE、READ ONLYまたはMIGRATEに変更するには、pdb_open句を指定します。
- オープン・モードをMOUNTEDに変更するには、pdb_close句を指定します。

表11-2 PDBのオープン・モード

オープン・モード	説明
READ WRITE	読取り/書込みオープン・モードの PDB では、問合せおよびユーザー・トランザクションを実行でき、ユーザーは REDO ログを生成できます。
READ ONLY	読取り専用オープン・モードの PDB では、問合せは実行できますが、ユーザー変更を実行することはできません。
MIGRATE	PDB が移行オープン・モードの場合は、PDB でデータベース・アップグレード・スクリプトを実行できます。
MOUNTED	PDB は、マウント・モードでは、マウント・モードの非 CDB と同じように動作します。どのオブジェクトの変更も行えず、またデータベース管理者のみがアクセス可能です。データファイルからの読取りも、データファイルへの書込みもできません。PDB に関する情報は、メモリー・キャッシュから削除されます。PDB のコールド・バックアップを実行できます。

V$PDBSビューのOPEN_MODE列を問い合せて、PDBのオープン・モードを表示できます。

関連項目:

PDBのオープン・モードの詳細は、『Oracle Database管理者ガイド』を参照してください。

pdb_name

オプションでpdb_nameを使用して、オープン・モードを変更するPDBの名前を指定できます。

pdb_open

この句を使用して、PDBのオープン・モードをREAD WRITE、READ ONLYまたはMIGRATEに変更できます。この句を指定するときにFORCEキーワードを指定しない場合は、そのPDBがMOUNTEDモードになっている必要があります。

READ WRITEまたはREAD ONLYを指定しない場合は、デフォルトでREAD WRITEが設定されます。例外は、そのPDBが、フィジカル・スタンバイ・データベースとして使用されるCDBに属している場合です。この場合、デフォルトはREAD ONLYになります。

READ WRITE

この句を指定すると、オープン・モードをREAD WRITEに変更できます。

READ ONLY

この句を指定すると、オープン・モードをREAD ONLYに変更できます。

[READ WRITE] UPGRADE

この句を指定すると、オープン・モードをMIGRATEに変更できます。READ WRITEキーワードはオプションであり、意味を明確にするためのものです。

RESTRICTED

オプションのRESTRICTEDキーワードを指定すると、PDBへのアクセスはPDB内でRESTRICTED SESSION権限を持つユーザーに限られます。

PDBがREAD WRITEまたはREAD ONLYモードにあるときに、RESTRICTEDキーワードとFORCEキーワードを指定してオープン・モードの変更を実行すると、そのPDBに接続しているセッションのうち、そのPDBにおけるRESTRICTED SESSION権限を持っていないすべてのセッションが終了され、ロールバックされます。

FORCE

このキーワードを指定すると、PDBのオープン・モードをREAD WRITEからREAD ONLYに、またはREAD ONLYからREAD WRITEに変更できます。FORCEキーワードを使用すると、オープン・モードの変更中にユーザーがPDBへの接続を維持できます。

READ WRITEからREAD ONLYへのPDBのオープン・モードの変更でFORCEを指定すると、オープン・モードの変更時にオープンになっているあらゆるREAD WRITEトランザクションが、DML操作をそれ以上実行できなくなり、またCOMMITもできなくなります。

FORCEの制限事項

PDBが現在MIGRATEモードにある場合は、FORCEキーワードは指定できません。また、現在オープン状態になっているPDBをMIGRATEモードに変更する際には、FORCEキーワードは指定できません。

RESETLOGS

この句を指定すると、新しいPDBインカネーションを作成し、PDBのポイント・イン・タイム・リカバリ後に、そのPDBをREAD WRITEモードでオープンできます。

関連項目:

CDBおよびPDBのPoint-in-Timeリカバリの実行の詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

instances_clause

Oracle Real Application Clusters環境でこの句を使用すると、指定されたOracle RACインスタンス内のPDBの状態を変更できます。この句を省略すると、PDBの状態は現在のインスタンス内のみで変更されます。

- instance_nameを使用して、1つ以上のインスタンス名のカンマ区切りのリストをかっこで囲んで指定します。これにより、それらのインスタンスのみPDBの状態が変更されます。
- Specify ALLを指定すると、すべてのインスタンス内のPDBの状態を変更できます。
- ALL EXCEPTを指定すると、指定されたインスタンスを除くすべてのインスタンス内のPDBの状態を変更できます。

PDBが1つ以上のインスタンス内ですでにオープンされている場合、別のインスタンス内でもオープンできますが、すでにオープンされているPDBと同じモードでオープンする必要があります。

pdb_close

この句を使用すると、PDBのオープン・モードをMOUNTEDに変更できます。この句を指定するときには、そのPDBがREAD WRITE、READ ONLYまたはMIGRATEモードにある必要があります。この句は、PDBでのSQL*Plus SHUTDOWNコマンドに相当します。

IMMEDIATE

オプションのIMMEDIATEキーワードを指定した場合、この句は、PDBでの即時モードのSQL*Plus SHUTDOWNコマンドに相当します。それ以外の場合、PDBは通常モードでシャットダウンされます。

関連項目:

SQL*Plus SHUTDOWNコマンドの詳細は、『SQL*Plusユーザーズ・ガイドおよびリファレンス』を参照してください。

ABORT

PDBを強制的に停止するには、ABORTを指定します。

instances_clause

Oracle Real Application Clusters環境でこの句を使用すると、指定されたOracle RACインスタンス内のPDBの状態を変更できます。PDBは、いくつかのインスタンスでクローズし、別のインスタンスでオープンしておくことができます。この句のセマンティクスの詳細は、instances_clauseを参照してください。

relocate_clause

Oracle Real Application Clusters環境では、この句を使用して、異なるOracle RACインスタンス上でPDBを再オープンするようにデータベースに指示できます。

- Oracle Databaseで選択された別のインスタンス上のPDBを再オープンするには、RELOCATEを指定します。
- 指定されたインスタンス内でPDBを再オープンするには、RELOCATE TO 'instance_name'を指定します。
- 現在のインスタンス内でPDBをクローズするには、NORELOCATEを指定します。これはデフォルトです。

pdb_save_or_discard_state

この句を使用して、CDBの再起動時にPDBのオープン・モードを保存または破棄するようデータベースに指示します。

- SAVEを指定する場合、CDBの再起動後のPDBのオープン・モードは、CDBの再起動前のオープン・モードと同じです。
- DISCARDを指定する場合、CDBの再起動後のPDBのオープン・モードはMOUNTEDになります。これはデフォルトです。

instances_clause

Oracle Real Application Clusters環境でこの句を使用して、指定されたOracle RACインスタンス内のPDBのオープン・モードを保存または破棄するようデータベースに指示します。この句を省略すると、データベースはSAVEまたはDISCARD設定を現在のインスタンスのPDBにのみ適用します。

- instance_nameを使用して、1つ以上のインスタンス名のカンマ区切りのリストをかっこで囲んで指定します。これにより、SAVEまたはDISCARD設定がそれらのインスタンスのPDBにのみ適用されます。
- ALLを指定して、SAVEまたはDISCARD設定をすべてのインスタンスのPDBに適用します。
- ALL EXCEPTを指定して、SAVEまたはDISCARD設定を指定されたインスタンスを除くすべてのインスタンスのPDBに

適用します。

pdb_change_state_from_root

この句を使用して、1つ以上のPDBの状態を変更できます。

- 状態を変更する1つ以上のPDBのpdb_nameを指定します。
- ALLを指定すると、CDB内のすべてのPDBの状態を変更できます。
- ALL EXCEPTを指定すると、pdb_nameを使用して指定したPDBを除く、CDB内のすべてのPDBの状態を変更できます。

PDBがすでに指定した状態にある場合は、PDBの状態は変わらず、エラーは返されません。状態を変更できないPDBがあると、そのPDBにエラーが発生します。

application_clauses

APPLICATION句を使用すると、アプリケーション・コンテナで次の操作を実行できます。

- アプリケーションのインストール、パッチ適用、アップグレードおよびアンインストール
- アプリケーション・バージョンおよびパッチ番号の登録
- アプリケーション・ルートとアプリケーションPDBの間でのアプリケーションの同期

関連項目:

アプリケーション・コンテナの管理の詳細は、『Oracle Database管理者ガイド』を参照してください。

アプリケーション名の指定

ほとんどのapplication_clausesでは、アプリケーション名を指定する必要があります。アプリケーション名の最大長は30バイトです。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。アプリケーション名は、アプリケーション・コンテナ内で一意である必要があります。

アプリケーション・バージョンの指定

いくつかのapplication_clausesでは、アプリケーション・バージョンを指定する必要があります。アプリケーション・バージョンは最大長が30バイトで、英数字、句読点、スペースを使用できます。アプリケーション・バージョンには大文字と小文字の区別があり、一重引用符で囲む必要があります。

コメントの指定

いくつかのapplication_clausesでは、アプリケーションのインストール、パッチ適用またはアップグレード操作に関連付けるコメントを指定できます。commentには、一重引用符で囲んだ文字列を入力します。

INSTALL句

INSTALL句は、アプリケーション・コンテナでアプリケーションをインストールする際に使用します。現在のコンテナはアプリケーションPDBではなくアプリケーション・ルートである必要があります。

- アプリケーションのインストールを開始する前に、BEGIN INSTALL句を指定します。
  - app_nameを使用して、アプリケーションに名前を割り当てます。
  - app_versionを使用して、アプリケーションにバージョンを割り当てます。

- オプションのCOMMENT句を指定すると、このインストールで作成されるアプリケーション・バージョンに関連付けるコメントを入力できます。

- アプリケーションのインストールが終了したら、END INSTALL句を指定します。
  - 対応するBEGIN INSTALL句で指定したものと同じapp_nameを指定する必要があります。
  - app_versionを指定する必要はありませんが、指定する場合は、対応するBEGIN INSTALL句に指定したものと同じバージョンを指定する必要があります。

PATCH句

PATCH句は、アプリケーション・コンテナでアプリケーションにパッチを適用する際に使用します。現在のコンテナはアプリケーションPDBではなくアプリケーション・ルートである必要があります。

- アプリケーションへのパッチ適用を開始する前に、BEGIN PATCH句を指定します。
  - app_nameに、パッチを適用するアプリケーション名を指定します。
  - numberに、パッチ番号を指定します。
  - オプションのMINIMUM VERSION句を使用すると、パッチを適用する前にアプリケーションがあるべき最小バージョンを指定できます。app_versionに、最小アプリケーション・バージョンを指定します。現在のアプリケーション・バージョンが最小アプリケーション・バージョンよりも低い場合、エラーが発生します。この句を省略した場合、最小バージョンは現在のアプリケーション・バージョンになります。
  - オプションのCOMMENT句を指定すると、パッチ適用に関連付けるコメントを入力できます。
- アプリケーションへのパッチ適用が終了したら、END PATCH句を指定します。
  - 対応するBEGIN PATCH句で指定したものと同じapp_nameを指定する必要があります。
  - numberを指定する必要はありませんが、指定する場合は、対応するBEGIN PATCH句に指定したものと同じ値を指定する必要があります。

UPGRADE句

UPGRADE句は、アプリケーション・コンテナでアプリケーションをアップグレードする際に使用します。現在のコンテナはアプリケーションPDBではなくアプリケーション・ルートである必要があります。

アプリケーション・ルートでTDEが使用されている場合は、アプリケーションをアップグレードする前に外部ストアを構成する必要があります。

- アプリケーションのアップグレードを開始する前に、BEGIN UPGRADE句を指定します。
  - app_nameに、アップグレードするアプリケーション名を指定します。
  - start_app_versionに、アプリケーションのアップグレード元バージョンを指定します。このバージョンが現在のアプリケーション・バージョンと一致しない場合、エラーが発生します。
  - end_app_versionに、アプリケーションのアップグレード先バージョンを指定します。
  - オプションのCOMMENT句を指定すると、アップグレードに関連付けるコメントを入力できます。
- アプリケーションのアップグレードが終了したら、END UPGRADE句を指定します。
  - 対応するBEGIN UPGRADE句で指定したものと同じapp_nameを指定する必要があります。
  - TO end_app_versionを指定する必要はありませんが、指定する場合は、対応するBEGIN UPGRADE

句に指定したものと同じバージョンを指定する必要があります。

UNINSTALL句

UNINSTALL句は、アプリケーション・コンテナからアプリケーションをアンインストールする際に使用します。現在のコンテナはアプリケーションPDBではなくアプリケーション・ルートである必要があります。

- アプリケーションのアンインストールを開始する前に、BEGIN UNINSTALL句を指定します。
  - app_nameに、アンインストールするアプリケーション名を指定します。
- アプリケーションのアンインストールが終了したら、END UNINSTALL句を指定します。
  - 対応するBEGIN UNINSTALL句で指定したものと同じapp_nameを指定する必要があります。

SET PATCH

SET PATCH句を使用すると、アプリケーション・コンテナにすでにインストールされているアプリケーションのパッチ番号を登録できます。この句を使用すると、PATCH句を使用してパッチ適用しなかったアプリケーションにパッチ番号を割り当てることができます。これは、PATCH句が使用できなかったOracle Databaseの旧リリースで、アプリケーションがPDBから移行された場合に便利です。現在のコンテナには、アプリケーション・ルートまたはアプリケーションPDBを指定できます。

- app_nameに、既存のアプリケーションの名前を指定します。
- numberを使用して、既存のアプリケーションにパッチ番号を割り当てます。

SET VERSION

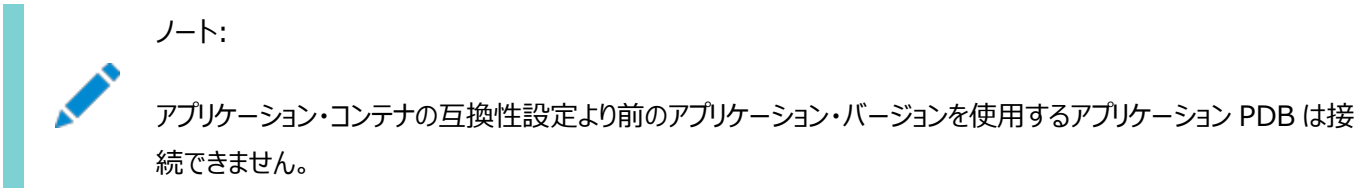
SET VERSION句を使用すると、アプリケーション・コンテナにすでにインストールされているアプリケーションのバージョンを登録できます。この句を使用すると、INSTALL句を使用してインストールしなかったアプリケーションに名前とバージョンを割り当てることができます。これは、INSTALL句が使用できなかったOracle Databaseの旧リリースで、アプリケーションがPDBから移行された場合に便利です。現在のコンテナには、アプリケーション・ルートまたはアプリケーションPDBを指定できます。

- app_nameを使用して、既存のアプリケーションに名前を割り当てます。
- app_versionを使用して、既存のアプリケーションにバージョンを割り当てます。

SET COMPATIBILITY VERSION

SET COMPATIBILITY VERSION句を使用すると、アプリケーションの互換性バージョンを設定できます。

アプリケーションの互換性バージョンは、アプリケーション・コンテナに属するアプリケーションPDBで使用可能なアプリケーション・バージョンのうち、最も古いバージョンです。現在のコンテナはアプリケーションPDBではなくアプリケーション・ルートである必要があります。



ノート:

アプリケーション・コンテナの互換性設定より前のアプリケーション・バージョンを使用するアプリケーション PDB は接続できません。

- app_nameを使用して、アプリケーションの名前を指定します。
- app_versionを使用して、アプリケーションの互換性バージョンを指定します。
- CURRENTを指定すると、互換性バージョンはアプリケーション・ルートにあるアプリケーションのバージョンに設定されます。

互換性バージョンは、互換性バージョンが設定され、アプリケーションPDBが作成されたときに適用されます。アプリケーションのアップグレードによって作成されたアプリケーション・ルート・クローンがある場合、互換性バージョンより古いバージョンに相当するすべてのアプリケーション・ルート・クローンが暗黙的に削除されます。

SYNC TO

アプリケーションを特定のバージョンまたはパッチ番号と同期できます。次の2つのバリエーションがあります。

1. `SYNC TO app_version`
2. `SYNC TO PATCH patch_number`

例

アプリケーション`salesapp`に対して次の操作を実行するとします。

1. バージョン1.0のインストール
2. パッチ101の適用
3. バージョン2.0へのアップグレード
4. パッチ102の適用
5. 3.0へのアップグレード

`ALTER PLUGGABLE DATABASE APPLICATION salesapp SYNC TO 2.0`では、バージョン2.0へのアップグレードまで(それを含む)のすべての文が再実行されます。

`ALTER PLUGGABLE DATABASE APPLICATION salesapp SYNC TO PATCH 102`では、パッチ102の適用まで(それを含む)のすべての文が再実行されます。

SYNC

SYNC句を使用すると、アプリケーションPDBのアプリケーションを、アプリケーション・ルートの同じアプリケーションのバージョンおよびパッチ・レベルと同期できます。これは、アプリケーション・ルートのアプリケーションに変更が加えられた後に役立ちます。現在のコンテナはアプリケーションPDBである必要があります。

- `app_name`に、アプリケーション・ルートに存在するアプリケーションの名前を指定します。アプリケーションはアプリケーションPDBに存在する場合もしない場合もあります。

ALL SYNC

`ALL SYNC`句を使用すると、アプリケーションPDBのすべてのアプリケーションを、アプリケーション・ルートのすべてのアプリケーションと同期できます。この句は、最近アプリケーションPDBをCDBに追加し、そのアプリケーションをアプリケーション・コンテナと同期する場合に便利です。現在のコンテナはアプリケーションPDBである必要があります。

snapshot_clauses

snapshot_clausesを使用すると、PDBの存続期間中にPDBのスナップショットを作成および管理できます。

pdb_snapshot_clause

PDBスナップショットを作成できるようにするには、この句を指定します。`CREATE PLUGGABLE DATABASE`文でこの句を指定することもできます。

- `NONE`はデフォルトであり、PDBのスナップショットを作成できないことを意味します。
- `MANUAL`は、PDBのスナップショットを手動でのみ作成できることを意味します。

- snapshot_intervalが指定されている場合、指定された間隔でPDBスナップショットが自動的に作成されます。また、ユーザーがPDBスナップショットを手動で作成できるようにもなります。
- snapshot_intervalは、分単位で指定する場合は3000未満である必要があります。
- snapshot_intervalは、時間単位で指定する場合は2000未満である必要があります。

materialize_clause

スナップショットPDBを完全なPDBクローンに変換するには、この句を使用します。この方法でこの句を使用すると、PDBのスナップショットを削除および消去できます。

- この句は、スナップショットとして作成されたPDBに対してのみ指定できます。
- PDBに属するすべてのデータファイル内のすべてのブロックがコピーされます。

create_snapshot_clause

この句を使用して、PDBに接続した後でPDBスナップショットを手動で作成します。

- PDBスナップショットが自動的に作成されるようにPDBが設定された場合でも、この分を発行できます。
- 指定された名前のPDBスナップショットがすでに存在する場合は、エラーが報告されます。
- 指定された名前のPDBスナップショットが作成されます。

drop_snapshot_clause

この句を使用して、PDBに接続した後でPDBスナップショットを手動で削除します。

- このスナップショットがいずれかのPDBによって使用されている場合は、エラーが報告されます。

set_max_pdb_snapshots

この句を使用して、特定のPDBのスナップショットの最大数を増加または減少させます。最初にPDBに接続する必要があります。

- この文を発行するときにPDBが読取り/書込みモードでオープンされていない場合は、エラーが発生します。
- 最大数を0に設定することにより、すべてのPDBスナップショットを削除できます。
- PDBごとに設定できるスナップショットの最大数は8です。

prepare_clause

- この句を使用して、データベースのミラー・コピーを準備します。作成されるファイル・グループを識別するためにmirror_nameを指定する必要があります。作成されたグループにはすべての準備完了ファイルが含まれています。
- REDUNDANCYオプションのEXTERNAL、NORMALまたはHIGHによって準備されるコピーの数を指定します。
- ミラーの冗長性を指定しない場合、ソース・データベースの冗長性が使用されます。

名前によるプラガブル・データベースの作成:例を準備します

プラガブル・データベースの名前(pdb_name)を指定した場合、pdb_nameは現在のPDBと一致するかどうかをチェックします。一致した場合は、それが実行されます。

```
ALTER PLUGGABLE DATABASE pdb_name PREPARE MIRROR COPY mirror_name WITH HIGH
REDUNDANCY
```

名前なしのプラガブル・データベースの作成:例を準備します

プラガブル・データベースの名前(pdb_name)を指定しない場合、文は現在のPDBで実行されます。

```
ALTER PLUGGABLE DATABASE PREPARE MIRROR COPY mirror_name WITH HIGH REDUNDANCY
```

drop_mirror_copy

この句を使用すると、プリコンパイルされた文によって作成されるデータおよびメタデータのミラー・コピーが破棄されます。準備操作に使用したものと同じミラー名を指定する必要があります。

この句を使用すると、すでにCREATE DATABASEまたはCREATE PLUGGABLE DATABASE文で分割されているデータベースを削除できません。

lost_write_protection

プラガブル・データベースの書込みの欠落を有効化:例

```
 ALTER PLUGGABLE DATABASE
   ENABLE LOST WRITE PROTECTION
```

プラガブル・データベースの書込みの欠落:例を無効にしてください

```
 ALTER PLUGGABLE DATABASE
   DISABLE LOST WRITE PROTECTION
```

データベースの消失書込み保護を無効化しても、消失書込み記憶域の割当て解除は行われないことに注意してください。消失書込み記憶域の割当て解除を行うには、DROP TABLESPACE文を使用する必要があります。

例

CDBからのPDBの切断: 例

次の文はPDB pdb1を切断し、PDBのメタデータをXMLファイル/oracle/data/pdb1.xmlに格納します。

```
ALTER PLUGGABLE DATABASE pdb1
  UNPLUG INTO '/oracle/data/pdb1.xml';
```

PDBの設定の変更: 例

次の文は、PDB pdb2内のすべての表領域で使用される記憶域の量に対する制限を500Mに変更します。

```
ALTER PLUGGABLE DATABASE pdb2
  STORAGE (MAXSIZE 500M);
```

PDBのデータファイルのオフライン化: 例

次の文は、PDB pdb3に関連付けられているデータファイルをオフラインにします。

```
ALTER PLUGGABLE DATABASE pdb3
  DATAFILE ALL OFFLINE;
```

PDBの状態の変更: 例

PDB pdb4がクローズされている、つまり、オープン・モードがMOUNTEDであることが前提となります。次の文は、READ ONLYオープン・モードでpdb4をオープンします。

```
ALTER PLUGGABLE DATABASE pdb4
  OPEN READ ONLY;
```

次の文は、FORCEキーワードを使用して、pdb4のオープン・モードをREAD ONLYからREAD WRITEに変更します。

```
ALTER PLUGGABLE DATABASE pdb4
  OPEN READ WRITE FORCE;
```

次の文は、PDB pdb4をクローズします。

```
ALTER PLUGGABLE DATABASE pdb4
  CLOSE;
```

次の文は、READ ONLYオープン・モードでPDB pdb4をオープンします。RESTRICTEDキーワードが指定されているため、PDBへのアクセスは、PDBの中でRESTRICTED SESSION権限を持っているユーザーのみに許可されます。

```
ALTER PLUGGABLE DATABASE pdb4
  OPEN READ ONLY RESTRICTED;
```

PDB pdb5がクローズされている、つまり、オープン・モードがMOUNTEDであることが前提となります。Oracle Real Application Clusters環境で、次の文は、インスタンスORCLDB_1およびORCLDB_2の中でPDB pdb5をREAD WRITEオープン・モードでオープンします。

```
ALTER PLUGGABLE DATABASE pdb5
  OPEN READ WRITE INSTANCES = ('ORCLDB_1', 'ORCLDB_2');
```

Oracle Real Application Clusters環境で、次の文は、現在のインスタンスの中のPDB pdb6をクローズして、インスタンスORCLDB_3の中でpdb6を再オープンするようにデータベースに指示します。

```
ALTER PLUGGABLE DATABASE pdb6
  CLOSE RELOCATE TO 'ORCLDB_3';
```

CDB内のすべてのPDBの状態の変更: 例

現在のコンテナがルートであることが前提となります。次の文はCDB内のすべてのPDBをREAD ONLYオープン・モードでオープンします。

```
ALTER PLUGGABLE DATABASE ALL
  OPEN READ ONLY;
```

ALTER PROCEDURE

目的

パッケージはPL/SQLを使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

ALTER PROCEDURE文を使用すると、スタンドアロンのストアド・プロシージャを明示的に再コンパイルできます。明示的に再コンパイルすることによって、実行時に暗黙的に再コンパイルする必要がなくなり、また、実行時のコンパイル・エラーとパフォーマンス上のオーバーヘッドもなくなります。

パッケージの一部であるプロシージャを再コンパイルする場合、ALTER PACKAGE文を使用して、そのパッケージ全体を再コンパイルします(「ALTER PACKAGE」を参照)。

ノート:

この文では、既存のプロシージャの宣言または定義は変更されません。プロシージャを再宣言または再定義する場合は、OR REPLACE 句を指定して CREATE PROCEDURE 文を使用します(「CREATE PROCEDURE」を参照)。

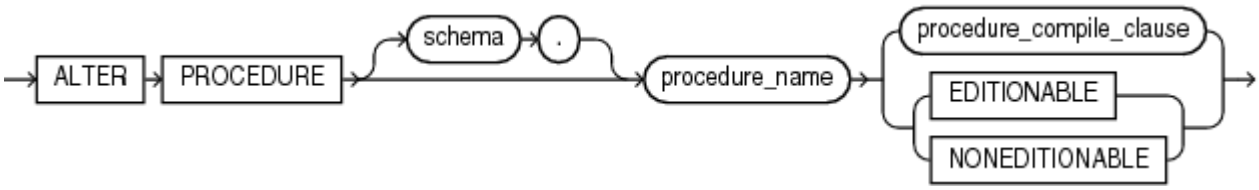
ALTER PROCEDURE文は、ALTER FUNCTION文と似ています。詳細は、「ALTER FUNCTION」を参照してください。

前提条件

プロシージャは、自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY PROCEDUREシステム権限が必要です。

構文

alter_procedure::=



(procedure_compile_clause: この句の構文の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。)

セマンティクス

schema

プロシージャが含まれているスキーマを指定します。schemaを指定しない場合、プロシージャは自分のスキーマ内にあるとみなされます。

procedure_name

再コンパイルするプロシージャの名前を指定します。

procedure_compile_clause

この句の構文とセマンティクスの詳細およびプロシージャの作成とコンパイルの詳細は、『Oracle Database PL/SQL言語リ

ファレンス』を参照してください。

EDITIONABLE | NONEDITIONABLE

この句を使用すると、schemaのスキーマ・オブジェクト・タイプPROCEDUREのエディショニングが後で有効化されたときに、そのプロシージャをエディション・オブジェクトにするか非エディション・オブジェクトにするかを指定できます。デフォルトは、EDITIONABLEです。エディション・オブジェクトと非エディション・オブジェクトの変更の詳細は、『Oracle Database開発ガイド』を参照してください。

ALTER PROFILE

目的

ALTER PROFILE文を使用すると、プロファイルのリソース制限またはパスワード管理パラメータを追加、変更または削除できます。

ALTER PROFILE文を使用すると、プロファイルに対して行った変更は、このプロファイルの現行のセッションのユーザーには影響しません。後続セッションのユーザーのみに影響します。

関連項目:

プロファイルの作成の詳細は、「CREATE PROFILE」を参照してください。

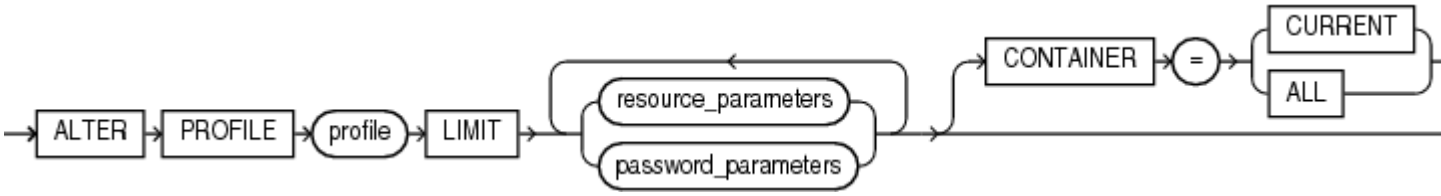
前提条件

ALTER PROFILEシステム権限が必要です。

CONTAINER句を指定する場合は、マルチテナント・コンテナ・データベース(CDB)に接続している必要があります。CONTAINER = ALLを指定する場合は、現在のコンテナがルートである必要があります。CONTAINER = CURRENTを指定するには、現在のコンテナがプラガブル・データベース(PDB)である必要があります。

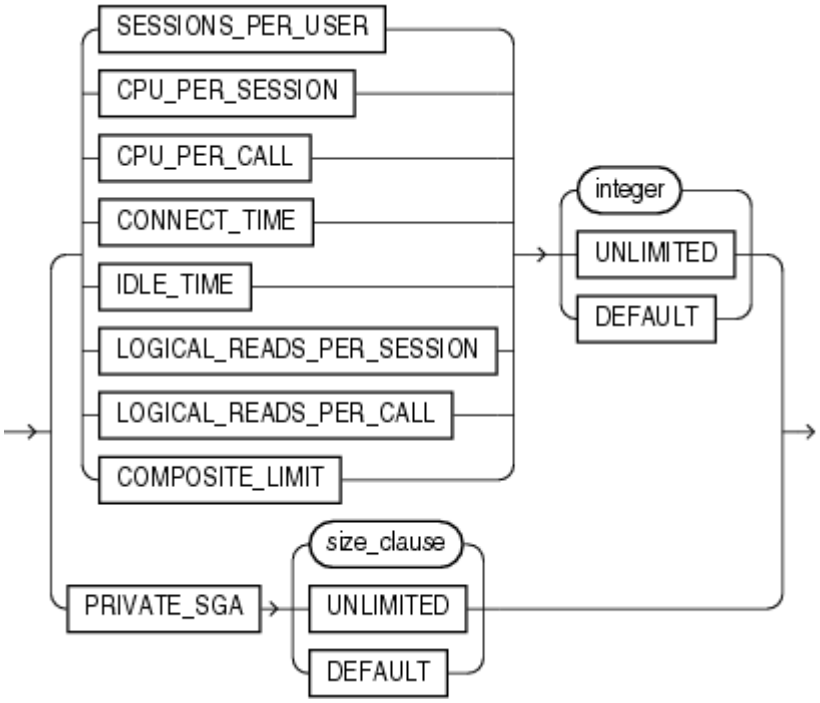
構文

alter_profile::=



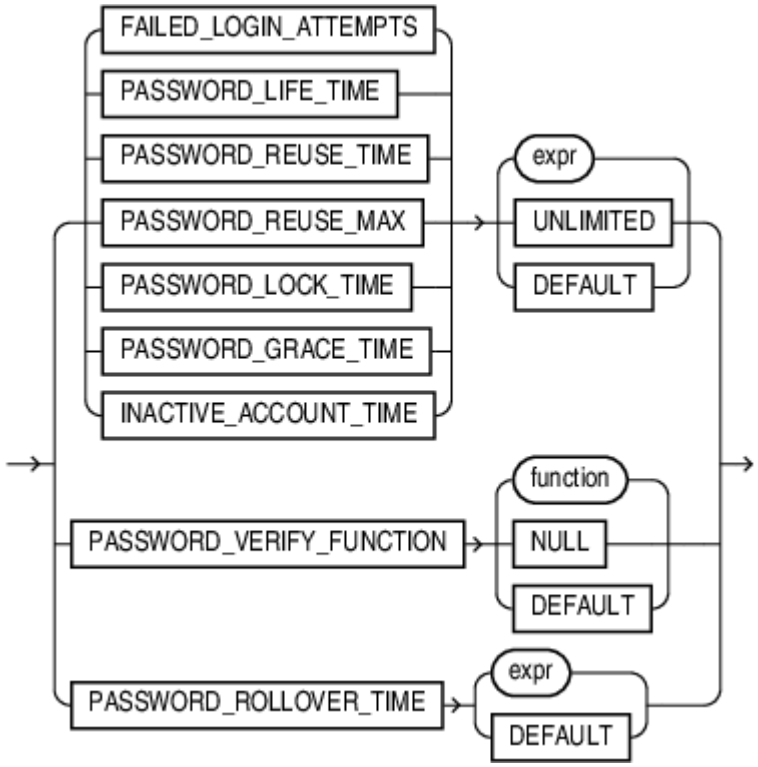
(resource_parameters::=、password_parameters::=)

resource_parameters::=



(size_clause::=)

password_parameters::=



セマンティクス

ALTER PROFILE文のキーワード、パラメータおよび句の意味は、すべてCREATE PROFILE文のキーワード、パラメータおよび句と同じです。

DEFAULTプロファイルから制限を削除することはできません。

詳細は、「CREATE PROFILE」および次の例を参照してください。

例

パスワードの使用不可化：例

次の文は、new_profileプロファイル(「プロファイルの作成：例」で作成)のパスワードを90日間再利用できないようにします。

```
ALTER PROFILE new_profile
   LIMIT PASSWORD_REUSE_TIME 90
   PASSWORD_REUSE_MAX UNLIMITED;
```

デフォルトのパスワード値の設定：例

次の文は、app_userプロファイル(「プロファイルのリソース制限の設定：例」で作成)のPASSWORD_REUSE_TIME値をDEFAULTプロファイルに定義された値にデフォルト設定します。

```
ALTER PROFILE app_user
   LIMIT PASSWORD_REUSE_TIME DEFAULT
   PASSWORD_REUSE_MAX UNLIMITED;
```

ログイン試行およびパスワードのロック回数の制限：例

次の文は、プロファイルapp_userを変更してFAILED_LOGIN_ATTEMPTSを5に設定し、PASSWORD_LOCK_TIMEを1に設定します。

```
ALTER PROFILE app_user LIMIT
   FAILED_LOGIN_ATTEMPTS 5
   PASSWORD_LOCK_TIME 1;
```

この文を使用すると、ログインに5回連続して失敗した場合に、app_userプロファイルに割り当てられるユーザー・アカウントが1日ロックされます。

パスワードの存続期間と猶予期間の変更：例

次の文は、プロファイルapp_user2のPASSWORD_LIFE_TIMEを90日に変更し、PASSWORD_GRACE_TIMEを5日に変更します。

```
ALTER PROFILE app_user2 LIMIT
   PASSWORD_LIFE_TIME 90
   PASSWORD_GRACE_TIME 5;
```

アカウントの非アクティブ期間の制限：例

次の文は、プロファイルapp_user2のINACTIVE_ACCOUNT_TIMEを連続した30日に変更します。

```
ALTER PROFILE app_user2 LIMIT
  INACTIVE_ACCOUNT_TIME 30;
```

アカウントがすでに一定の日数、非アクティブになっている場合、それらの日数が新しい30日間制限に考慮されます。

同時セッション数の制限：例

次の文は、app_userプロファイルの新しい同時セッション数の制限を5に定義します。

```
ALTER PROFILE app_user LIMIT SESSIONS_PER_USER 5;
```

現在、プロファイルapp_userにSESSIONS_PER_USERの制限が定義されていない場合は、このプロファイルに制限5が追加されます。プロファイルに制限が定義されている場合は、前述の文によってその制限が5に再定義されます。プロファイルapp_userが割り当てられているすべてのユーザーは、同時実行のセッションが5件に制限されます。

プロファイル制限の削除: 例

この文は、app_userプロファイルからIDLE_TIME制限を削除します。

```
ALTER PROFILE app_user LIMIT IDLE_TIME DEFAULT;
```

プロファイルapp_userが割り当てられているユーザーは、以降のセッションからプロファイルDEFAULTに定義されたIDLE_TIME制限に従います。

プロファイルのアイドル時間の制限: 例

次の文は、DEFAULTプロファイルのアイドル時間の制限を2分に定義します。

```
ALTER PROFILE default LIMIT IDLE_TIME  2;
```

IDLE_TIMEの制限は、次のユーザーに適用されます。

- プロファイルが明示的に割り当てられていないユーザー
- IDLE_TIMEの制限が定義されていないプロファイルが明示的に割り当てられているユーザー

次の文は、プロファイルapp_user2に無制限のアイドル時間を設定します。

```
ALTER PROFILE app_user2 LIMIT IDLE_TIME UNLIMITED;
```

プロファイルapp_user2が割り当てられているすべてのユーザーは、以降のセッションから無制限のアイドル時間が許可されます。

段階的なパスワード・ロールオーバーの有効化: 例

次の文は、プロファイルusr_profのパスワード・ロールオーバー時間を2日に設定します。

```
ALTER PROFILE usr_prof LIMIT PASSWORD_ROLLOVER_TIME 2 ;
```

ALTER RESOURCE COST

目的

ALTER RESOURCE COST文を使用すると、セッションで使用するリソース・コストの合計を算出するための式を指定または変更できます。

Oracle Databaseは、その他のリソースの使用も監視しますが、セッションに対するリソース・コストの合計は、この構文の4種類のリソースに基づいて算出されます。

この文を使用すると、4種類のリソースに重みを適用できます。Oracle Databaseは、プロファイルに指定されたこれらのリソースの値に重みを適用し、リソース・コストの合計を算出する計算式を設定します。CREATE PROFILE文のCOMPOSITE_LIMITパラメータを使用して、セッションに対するコストを制限できます。セッションのリソース・コストが制限を超えた場合、セッションは異常終了し、エラーが戻ります。各リソースに割り当てた重みを変更するためにALTER RESOURCE COST文を使用した場合、現行のセッション以降のすべてのセッションで、その新しい重みを基にリソース・コストが計算されます。

関連項目:

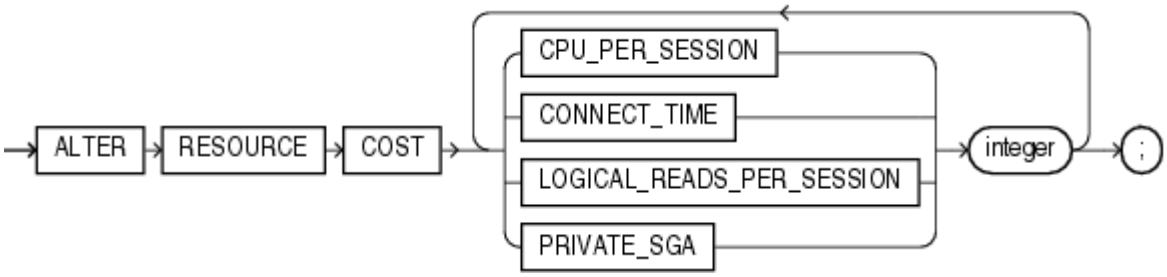
すべてのリソースおよびリソース制限の設定については、「CREATE PROFILE」を参照してください。

前提条件

ALTER RESOURCE COSTシステム権限が必要です。

構文

alter_resource_cost::=



セマンティクス

最初にセッションで使用された各リソースの量にそのリソースの重みを乗算し、次に、4種類のリソースの乗算結果を加算することによって、リソース・コストの合計が計算されます。どのセッションについても、このコストは、ユーザーのプロファイル内のCOMPOSITE_LIMITパラメータの値によって制限されます。乗算結果と総コストは、ともにサービス単位と呼ばれる単位で表されます。

CPU_PER_SESSION

このキーワードを使用すると、CPU_PER_SESSIONリソースに重みを適用できます。

CONNECT_TIME

このキーワードを使用すると、CONNECT_TIMEリソースに重みを適用できます。

LOGICAL_READS_PER_SESSION

この句を使用すると、LOGICAL_READS_PER_SESSIONリソースに重みを適用できます。論理読取りには、メモリーおよび

ディスクの両方から読み取られたブロックが含まれます。

PRIVATE_SGA

この句を使用すると、PRIVATE_SGAリソースに重みを適用できます。共有サーバー・アーキテクチャを使用して、セッション用としてSGA内でプライベート領域を割り当てている場合のみ、この制限が適用されます。

integer

各リソースの重みを指定します。各リソースに割り当てる重みによって、各リソースがリソース・コストの合計に影響する程度が決定されます。リソースに重みを割り当てない場合は、デフォルト値の0(ゼロ)が適用され、コストへの影響はありません。重みを割り当てた場合は、データベースの次のセッション以降のすべてのセッションで、その重みが適用されます。

例

リソース・コストの変更: 例

次の文は、リソースCPU_PER_SESSIONおよびCONNECT_TIMEに重みを割り当てます。

```
ALTER RESOURCE COST
   CPU_PER_SESSION 100
   CONNECT_TIME      1;
```

この重みによって、セッションごとに次のコスト計算式が設定されます。

```
cost = (100 * CPU_PER_SESSION) + (1 * CONNECT_TIME)
```

この例では、CPU_PER_SESSIONおよびCONNECT_TIMEの値は、DEFAULTプロファイルまたはセッションのユーザーのプロファイルにある値のいずれかです。

ここでは、リソースLOGICAL_READS_PER_SESSIONおよびPRIVATE_SGAに重みを割り当てていないため、これらのリソースは式に含まれません。

プロファイルでCOMPOSITE_LIMIT値として500を割り当てた場合、costが500を超えると、必ず、セッションはこの制限を超えます。たとえば、CPU時間0.04秒、経過時間101分を使用するセッションは、この制限を超えます。同様に、CPU時間が0.0301秒、経過時間が200分のセッションもこの制限を超えます。

一度割り当てた重みは、次のように、別のALTER RESOURCE文を発行することによって変更できます。

```
ALTER RESOURCE COST
   LOGICAL_READS_PER_SESSION 2
   CONNECT_TIME 0;
```

新しく割り当てた重みによって、新しいコスト計算式が設定されます。

```
cost = (100 * CPU_PER_SESSION) + (2 * LOGICAL_READ_PER_SECOND)
```

CPU_PER_SESSIONおよびLOGICAL_READS_PER_SECONDの値は、DEFAULTプロファイルまたはセッションのユーザーのプロファイルにある値のいずれかです。

このALTER RESOURCE COST文によって、式は次のように変更されます。

- CPU_PER_SESSIONリソースの重みは指定しません。このリソースにはすでに重みが割り当てられているため、式では先に指定した重みがそのまま使用されます。
- LOGICAL_READS_PER_SESSIONリソースに重みを割り当てたため、このリソースが式で使用されます。
- CONNECT_TIMEリソースに0(ゼロ)を割り当てたため、このリソースは式に含まれていません。

- PRIVATE_SGAリソースの重みは指定しません。このリソースには重みを割り当てていないため、式に含まれていません。

ALTER ROLE

目的

ALTER ROLE文を使用すると、ロールを使用可能にするために必要な認可を変更できます。

関連項目:

- ロールの作成の詳細は、「CREATE ROLE」を参照してください。
- セッションのロールを使用可能または使用禁止にする方法については、「SET ROLE」を参照してください。

前提条件

ロールにADMIN OPTIONが付与されている必要があります。付与されていない場合は、ALTER ANY ROLEシステム権限が必要です。

ロールをIDENTIFIED GLOBALLYに変更する前に、次の作業が必要です。

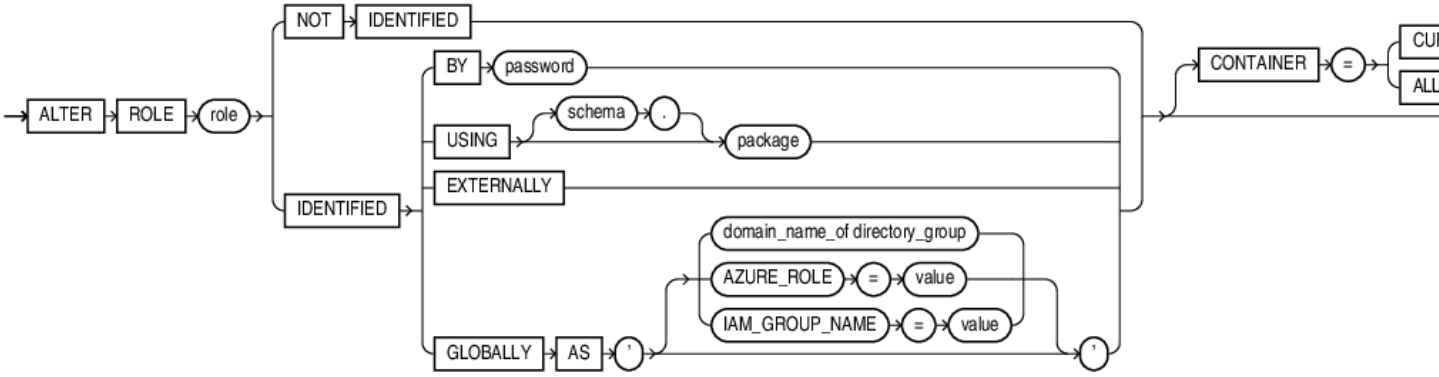
- ロールに対して外部的に識別されたロール権限をすべて取り消します。
- すべてのユーザー、ロールおよびPUBLICからロールの付与を取り消します。

この規則の唯一の例外として、現在ロールを変更しているユーザーからはそのロールを取り消さないでください。

CONTAINER句を指定する場合は、マルチテナント・コンテナ・データベース(CDB)に接続している必要があります。CONTAINER = ALLを指定する場合は、現在のコンテナがルートである必要があります。CONTAINER = CURRENTを指定するには、現在のコンテナがプラガブル・データベース(PDB)である必要があります。

構文

alter_role::=



セマンティクス

ALTER ROLE文のキーワード、パラメータおよび句の意味は、すべてCREATE ROLE文のキーワード、パラメータおよび句と同じです。

中央管理されているユーザーを使用している場合、GLOBALLYをASとともに指定して、ディレクトリ・グループをグローバル・ロールにマップします。ディレクトリ・グループはそのドメイン名によって識別されます。

ロールの変更の制限事項

NOT IDENTIFIEDロールが別のロールに付与されている場合、このロールはいずれのIDENTIFIEDタイプにも変更できませ

ん。

ロールの変更のノート:

- すでに使用可能なロールのユーザー・セッションには影響しません。
- パスワードによって識別されるロールをアプリケーション・ロールに変更する場合(USING package句を使用)、ロールに対応付けられたパスワード情報は失われます。次にロールが使用可能になるときから、新しい認証方式が使用されます。
- ALTER ANY ROLEシステム権限を持つユーザーが、IDENTIFIED GLOBALLYロールをIDENTIFIED BY password、IDENTIFIED EXTERNALLYまたはNOT IDENTIFIEDに変更すると、非グローバルなロールを作成した場合と同様に、そのユーザーに変更されたロールがADMIN OPTION付きで付与されます。

詳細は、「CREATE ROLE」および次の例を参照してください。

例

ロールの識別の変更: 例

次の文は、ロールwarehouse_user (「ロールの作成: 例」で作成)をNOT IDENTIFIEDに変更します。

```
ALTER ROLE warehouse_user NOT IDENTIFIED;
```

ロールのパスワードの変更: 例

次の文は、dw_managerロール(「ロールの作成: 例」で作成)のパスワードをdataに変更します。

```
ALTER ROLE dw_manager
   IDENTIFIED BY data;
```

パスワードの変更後、ロールdw_managerが付与されているユーザーは、新しいパスワードdataを使用してこのロールを使用可能にする必要があります。

アプリケーション・ロール: 例

次の例は、ロールdw_managerをhr.adminパッケージを使用してアプリケーション・ロールに変更します。

```
ALTER ROLE dw_manager IDENTIFIED USING hr.admin;
```

ALTER ROLLBACK SEGMENT

ノート:

ロールバック・セグメントを使用せずに、自動 UNDO 管理モードでデータベースを実行することを強くお薦めします。ロールバック・セグメントは、以前のバージョンの Oracle Database との互換性に必要な場合以外は使用しないでください。自動 UNDO 管理の詳細は、『Oracle Database 管理者ガイド』を参照してください。

目的

ALTER ROLLBACK SEGMENT文を使用すると、ロールバック・セグメントのオンライン/オフライン切替え、記憶特性の変更、またはロールバック・セグメントの最適サイズまたは指定サイズへの縮小を行うことができます。

ここでは、データベースがロールバックUNDOモードで実行されている(初期化パラメータUNDO_MANAGEMENTにMANUALを設定、またはすべて設定しない)ことを前提としています。データベースが自動UNDOモードで実行されている場合(初期化パラメータUNDO_MANAGEMENTにデフォルト値のAUTOを設定)、ユーザーが作成したロールバック・セグメントは意味を持ちません。

関連項目:

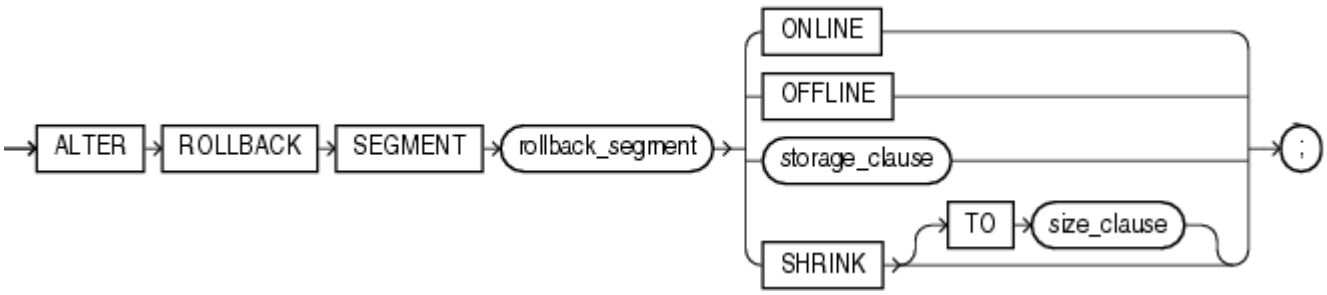
- ロールバック・セグメントの作成については、「CREATE ROLLBACK SEGMENT」を参照してください。
- UNDO_MANAGEMENTパラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

前提条件

ALTER ROLLBACK SEGMENTシステム権限が必要です。

構文

alter_rollback_segment::=



(storage_clause、size_clause::=)

セマンティクス

rollback_segment

既存のロールバック・セグメントの名前を指定します。

ONLINE

ONLINEを指定すると、ロールバック・セグメントをオンラインにできます。ロールバック・セグメントを作成した場合、最初はオフライン状態になり、トランザクションに使用できなくなります。この句を指定した場合、ロールバック・セグメントはオンラインになり、イン

スタンスは、トランザクションに対してそのロールバック・セグメントを使用できるようになります。また、初期化パラメータROLLBACK_SEGMENTSを使用すると、インスタンスの起動時にロールバック・セグメントをオンラインにできます。

関連項目:

ロールバック・セグメントのオンライン化：例

OFFLINE

OFFLINEを指定すると、ロールバック・セグメントをオフラインにできます。

- ロールバック・セグメント内に、アクティブ・トランザクションのロールバックに必要な情報が含まれていない場合は、すぐにオフラインになります。
- ロールバック・セグメントにアクティブ・トランザクションについての情報が含まれている場合、このロールバック・セグメントをその後のトランザクションに対して使用できないようにします。また、そのすべてのアクティブ・トランザクションがコミットまたはロールバックされた後、ロールバック・セグメントはオフラインになります。

オフラインになっているロールバック・セグメントは、どのインスタンスからもオンラインにできます。

ロールバック・セグメントがオンラインかオフラインかを確認するには、データ・ディクショナリ・ビューDBA_ROLLBACK_SEGSのSTATUS列を問い合せます。オンライン・ロールバック・セグメントの値はIN_USEです。オフライン・ロールバック・セグメントの値はAVAILABLEです。

ロールバック・セグメントのオフライン化の制限事項

SYSTEMロールバック・セグメントはオフラインにできません。

storage_clause

storage_clauseを使用すると、ロールバック・セグメントの記憶特性を変更できます。

ロールバック・セグメント記憶域の制限事項

INITIALパラメータの値は変更できません。ロールバック・セグメントがローカル管理表領域にある場合、変更可能な記憶域パラメータはOPTIMALのみです。ロールバック・セグメントがディクショナリ管理表領域にある場合、変更可能な記憶域パラメータは、NEXT、MINEXTENTS、MAXEXTENTSおよびOPTIMALのみです。

関連項目:

構文および追加情報については、「storage_clause」を参照してください。

SHRINK句

SHRINKを指定すると、ロールバック・セグメントを最適サイズまたは指定サイズに縮小できます。縮小されるかどうか、および縮小量は、ロールバック・セグメントの使用可能領域およびアクティブ・トランザクションのロールバック・セグメント内での領域保持状態の状況によって異なります。

TO size_clauseに値を指定しなかった場合、ロールバック・セグメントを作成したCREATE ROLLBACK SEGMENT文のstorage_clauseのOPTIMALで指定した値が、デフォルトのサイズになります。OPTIMAL値を指定しなかった場合、CREATE ROLLBACK SEGMENT文のstorage_clauseのMINEXTENTSで指定した値がデフォルトのサイズになります。

TO size_clauseに値を指定するかどうかにかかわらず、次のことがいえます。

- この文の実行時には、ロールバック・セグメントの縮小値が有効です。その後、サイズはCREATE ROLLBACK SEGMENT文のOPTIMAL値に戻ります。
- ロールバック・セグメントは、エクステント数2未満には縮小できません。

ロールバック・セグメントを縮小した後でロールバック・セグメントの実際のサイズを確認する場合は、DBA_SEGMENTSビューのBYTES列、BLOCKS列およびEXTENTS列を問い合せます。

ロールバック・セグメントの縮小の制限事項

Oracle Real Application Clusters環境では、自分のインスタンスに対してオンラインになっているロールバック・セグメントのみを縮小できます。

関連項目:

この句の詳細は、「size_clause」および「ロールバック・セグメントのサイズ変更: 例」を参照してください。

例

次の例では、ロールバック・セグメントrbs_one (「ロールバック・セグメントの作成: 例」で作成)を使用します。

ロールバック・セグメントのオンライン化: 例

この文は、ロールバック・セグメントrbs_oneをオンラインにします。

```
ALTER ROLLBACK SEGMENT rbs_one ONLINE;
```

ロールバック・セグメントのサイズ変更: 例

この文は、ロールバック・セグメントrbs_oneを縮小します。

```
ALTER ROLLBACK SEGMENT rbs_one
   SHRINK TO 100M;
```

ALTER SEQUENCE

目的

ALTER SEQUENCE文を使用すると、既存の順序の増分値、最小値および最大値、キャッシュされる数および動作を変更できます。この文は、順序番号に影響します。

関連項目:

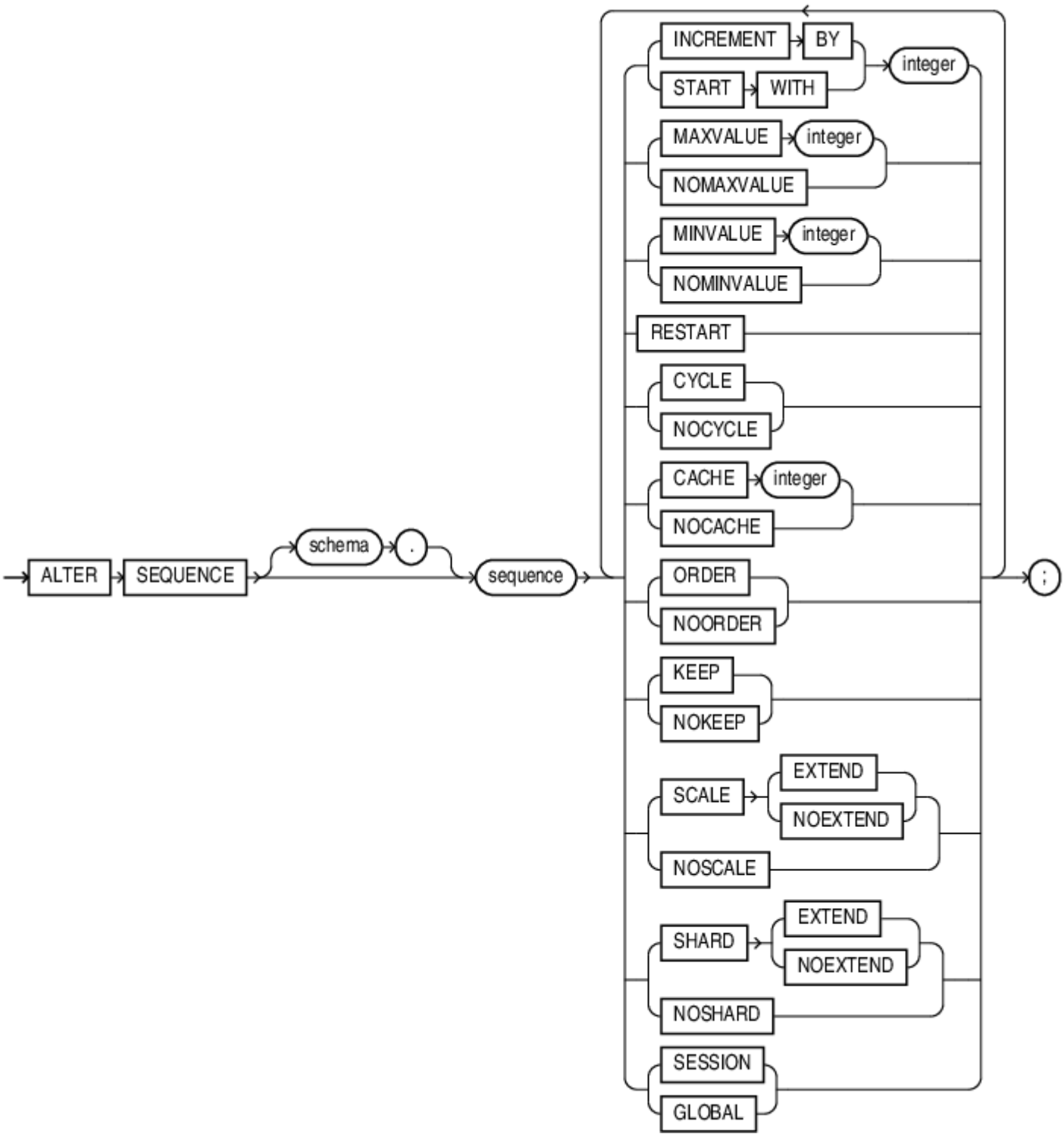
順序の詳細は、「CREATE SEQUENCE」を参照してください。

前提条件

順序が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、順序に対するALTERオブジェクト権限またはALTER ANY SEQUENCEシステム権限が必要です。

構文

alter_sequence::=



セマンティクス

この文のキーワードおよびパラメータの意味は、順序を作成する場合と同じです。

- NEXTVALを最初に呼び出す前に、INCREMENT BYの値を変更する場合、いくつかの順序番号がスキップされます。このため、元のSTART WITHの値を保持するには、順序を削除し、これを元のSTART WITHの値および新しいINCREMENT BYの値を使用して再作成する必要があります。
- 昇順の場合、RESTARTを指定してNEXTVALをMINVALUEにリセットします。降順の場合、RESTARTによってNEXTVALはMAXVALUEにリセットされます。
- 順序を別の番号で再開するには、RESTARTをSTART WITH句で指定して、順序を再開する値を設定します。

- KEEP句またはNOKEEP句を使用して、要求のランタイムとフェイルオーバーとの間で順序を変更すると、その要求に対するアプリケーション・コンティニュイティのための再実行中にNEXTVALの元の値は維持されません。
- いくつかの妥当性チェックが行われます。たとえば、MAXVALUEの値に現行の順序番号より小さい値は指定できません。

関連項目:

順序の作成については、「CREATE SEQUENCE」を参照してください。順序の削除および再作成については、「DROP SEQUENCE」を参照してください。

SCALE

順序の拡張性を有効にするには、SCALEを使用します。SCALEが指定されている場合、数値オフセットが順序の先頭に付加され、生成された値からすべての重複が削除されます。

EXTEND

EXTENDをSCALEとともに指定すると、生成される順序値は全長(x+y)になります。ここで、xは拡張可能なオフセット(デフォルト値は6)、yは順序内の数字の最大数(maxvalue/minvalue)です。

SCALEを使用する場合は、順序に対してORDERを同時に使用しないことをお薦めします。

NOEXTEND

NOEXTENDは、SCALE句のデフォルト設定です。NOEXTENDが設定されていると、生成される順序値の幅は最大でも順序内の数字の最大数(maxvalue/minvalue)です。この設定は、固定幅の列を移入するために順序が使用される、既存のアプリケーションとの統合に役立ちます。

SHARD

シャード間で一意の連番を生成するには、この句を使用します。

グローバル全シャード間で一意の順序を返す、グローバルで小さなシャード・オブジェクトとして、順序オブジェクトが作成されます。順序オブジェクトは、シャード・データベースと相対的な一意の順序値を返すカタログ・データベースでも作成されます。

シャード順序の動作は、EXTENDおよびNOEXTENDのキーワードで定義します。

EXTEND

SHARD句でEXTENDを指定する場合、生成される順序値は(x + y)の合計の長さになります。xはSHARDオフセットの長さで、サイズは4です。サイズ4というのは、最大数のシャードの幅に対応します。つまり、順序値の最初に1000が追加されます。yは順序maxvalue/minvalueでの最大桁数です。

NOEXTEND

SHARD句のデフォルト設定はNOEXTENDです。

NOEXTENDを指定すると、生成される順序値の幅は最大でも順序内の数字の最大数(maxvalue/minvalue)です。この設定は、固定幅の列を移入するために順序が使用される、既存のアプリケーションとの統合に役立ちます。

SHARD NOEXTENDを指定して順序でNEXTVALを呼び出す場合、生成される値が順序のmaxvalue/minvalueで表される桁数を超えると、ユーザー・エラーがスローされます。

SHARDとSCALEによる順序

SCALE句とSHARD句を一緒に指定すると、順序は複数のインスタンスおよびセッションのシャード・データベースでグローバルに一意の、拡張可能な値になります。

SCALE句とSHARD句でEXTENDを指定する場合、生成される順序値は(x+y+z)の合計の長さになります。xはSHARDオフセットの長さで、デフォルトのサイズは4です。yは拡張可能なオフセットの長さで、デフォルト値は6(5)、zは順序maxvalue/minvalueでの最大桁数です。

SHARD句とSCALE句でEXTENDまたはNOEXTENDを指定すると、SHARDとSCALEの両方に適用されます。EXTENDまたはNOEXTENDを別々に指定する必要はありません。SHARD句とSCALE句の両方にEXTENDまたはNOEXTENDオプションを別々に指定すると、その値が同じでも違っていても解析エラーが発生し、EXTEND句が重複または競合しているというメッセージが返されます。

SHARDを使用する場合は、順序に対してORDERを同時に使用しないことをお薦めします。

SHARDは、CACHEモードとNOCACHEモードで使用できます。

例

順序の変更: 例

次の文は、customers_seq順序(「順序の作成: 例」で作成)に新しい最大値を設定します。

```
ALTER SEQUENCE customers_seq
   MAXVALUE 1500;
```

次の文は、customers_seq順序にCYCLEおよびCACHEオプションを指定します。

```
ALTER SEQUENCE customers_seq
   CYCLE
   CACHE 5;
```

ALTER SESSION

目的

ALTER SESSION文を使用すると、データベースへの接続に影響するすべての条件またはパラメータを、設定または変更できます。この文は、データベースとの接続を切断するまで有効です。

前提条件

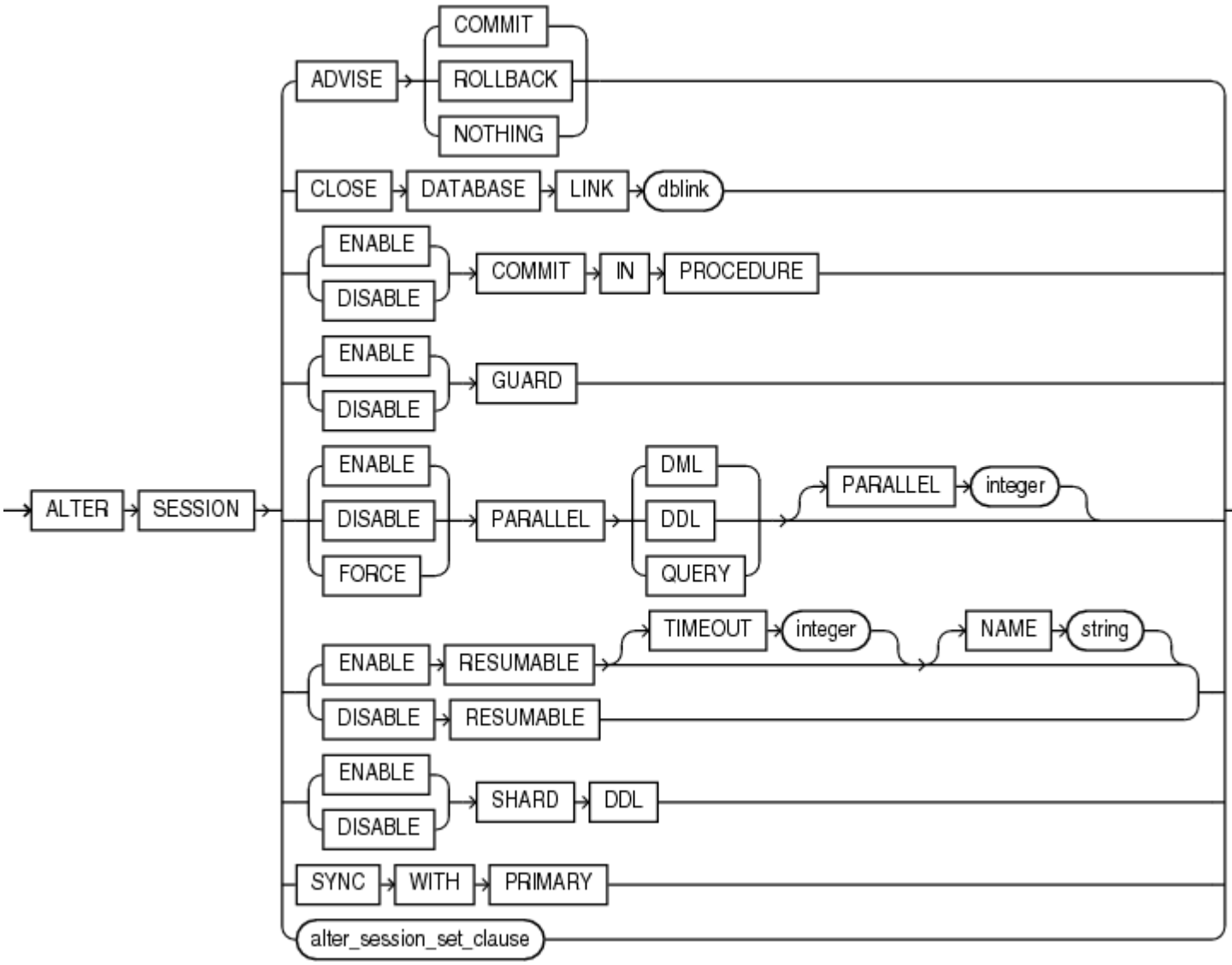
SQLトレース機能を使用可能または使用禁止にするには、ALTER SESSIONシステム権限が必要です。

再開可能な領域割当てを使用可能または使用禁止にするには、RESUMABLEシステム権限が必要です。

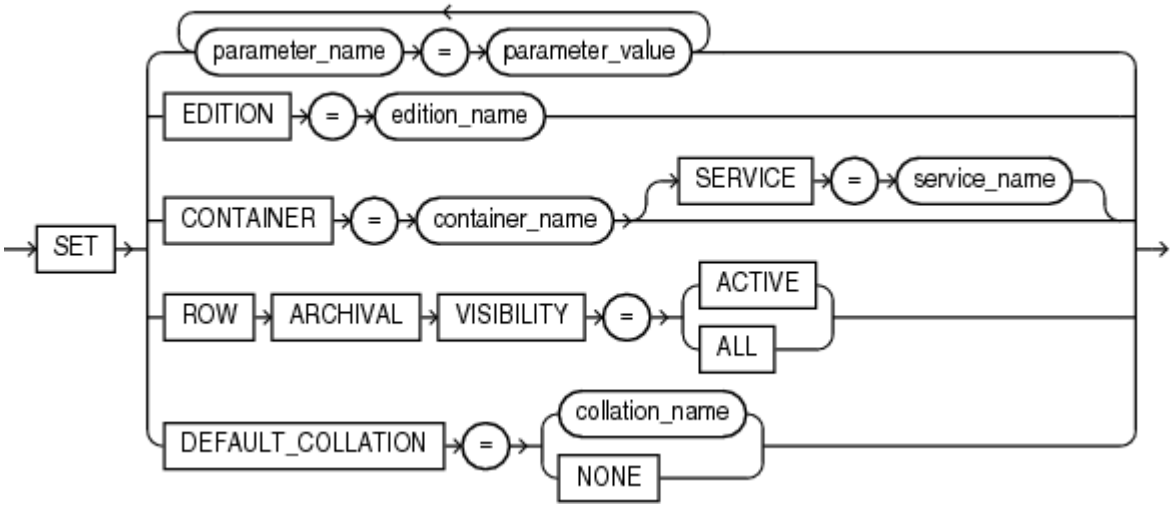
特に指定がないかぎり、これ以外の操作についての権限は必要ありません。

構文

alter_session::=



alter_session_set_clause::=



セマンティクス

ADVISE句

ADVISE句を指定すると、分散トランザクションを強制処理するためのアドバイスをリモート・データベースに送ることができます。リモート・データベース内のDBA_2PC_PENDINGビューのADVICE列に、アドバイスが表示されます(値'C'がCOMMIT、'R'がROLLBACK、' 'がNOTHINGを示します)。トランザクションの状態がインダウトになった場合、データベース管理者は、このアドバイスを使用してトランザクションをコミットするか、ロールバックするかを決定できます。

単一トランザクションにおいて、ADVISE句を指定したALTER SESSION文を複数発行し、リモート・データベースごとに異なるアドバイスを送ることができます。ADVISE句を指定した文はそれぞれ、ADVISE句を指定した別の文が発行されるまで、トランザクション内の後続する文で参照されるデータベースに対してアドバイスを送ります。

関連項目:

分散トランザクションの強制実行: 例

CLOSE DATABASE LINK句

CLOSE DATABASE LINKを指定すると、データベース・リンクdblinkをクローズできます。データベース・リンクを使用するSQL文を発行した場合、Oracle Databaseは、このデータベース・リンクを使用してリモート・データベース上にセッションを作成します。この接続は、セッションの終了またはデータベース・リンクの数が初期化パラメータOPEN_LINKSの値を超えるまでオープンされています。リンクをオープンしたままにしておくことによって発生するネットワークのオーバーヘッドを減らすには、セッションでデータベース・リンクを再度使用しない場合に、この句を使用してデータベース・リンクを明示的にクローズします。

関連項目:

「データベース・リンクをクローズする: 例」

ENABLE | DISABLE COMMIT IN PROCEDURE

プロシージャおよびストアド・ファンクションはPL/SQLで記述されるため、COMMIT文とROLLBACK文を発行できます。アプリケーション自体が直接発行していないCOMMIT文やROLLBACK文によって、アプリケーションが中断される場合、DISABLE COMMIT IN PROCEDUREを指定して、セッション中にコールされるプロシージャおよびストアド・ファンクションがこれらの文を発行しないように制御します。

その後、ENABLE COMMIT IN PROCEDUREを指定することによって、セッションでプロシージャおよびストアド・ファンクションがCOMMITおよびROLLBACK文を発行できるようになります。

一部のアプリケーションは、自動的にプロシージャおよびストアド・ファンクションでのCOMMIT文やROLLBACK文を禁止します。詳細は、ご使用のアプリケーションのドキュメントを参照してください。

ENABLE | DISABLE GUARD

ALTER DATABASEのsecurity_clauseを使用すると、SYSユーザー以外のユーザーは、プライマリ・データベースまたはスタンバイ・データベース上のデータまたはデータベース・オブジェクトを変更できなくなります。この句を使用すると、現行のセッションの設定を上書きできます。

関連項目:

GUARDの設定の詳細は、「security_clause」 を参照してください。

PARALLEL DML | DDL | QUERY

PARALLELパラメータを使用すると、そのセッションの後続のDML、DDLまたは問合せ文をパラレル実行するかどうかを指定できます。この句は、現行のセッション中に表自体を変更せずに、表の並列度を上書き可能にします。コミットされていないトランザクションは、DMLに対してこの句を実行する前に、コミットまたはロールバックされる必要があります。

関連項目:

パラレルDMLの有効化：例

ENABLE句

ENABLEを指定すると、セッション内の後続文をパラレルで実行できます。これは、DDL文および問合せ文のデフォルトです。

- DML: パラレル・ヒントまたはパラレル句が指定されている場合に、DML文をパラレル・モードで実行します。
- DDL: パラレル句が指定されている場合に、DDL文をパラレル・モードで実行します。
- QUERY: パラレル・ヒントまたはパラレル句が指定されている場合に、問合せをパラレル・モードで実行します。

ENABLE句の制限事項

オプションのPARALLEL integerをENABLEと組み合せて指定することはできません。

DISABLE句

DISABLEを指定すると、セッション内の後続文をシリアルで実行できます。これは、DML文のデフォルトです。

- DML: DML文をシリアルで実行します。
- DDL: DDL文をシリアルで実行します。
- QUERY: 問合せをシリアルで実行します。

DISABLE句の制限事項

オプションのPARALLEL integerをDISABLEと組み合せて指定することはできません。

FORCE句

FORCEを使用すると、セッションの後続文を強制的にパラレル実行できます。パラレル句もパラレル・ヒントも指定されていない

場合は、デフォルトの並列度が使用されます。この句は、セッションの後続文に指定されたすべてのparallel_clauseを上書きしますが、パラレル・ヒントによって上書きされます。

- DML: パラレルDML制限のどれにも違反していない場合、特定の並列度がこの句に指定されていないかぎり、セッションの後続のDML文は、デフォルトの並列度で実行されます。
- DDL: 特定の並列度がこの句に指定されていないかぎり、セッションの後続のDDL文は、デフォルトの並列度で実行されます。結果のデータベース・オブジェクトは、通常の並列度に対応します。

  FORCE DDLを指定した場合、そのセッションで作成されるすべての表は、自動的にデフォルトの並列度で作成されます。結果は、CREATE TABLE文で(デフォルトの並列度を使用して)parallel_clauseを指定した場合と同じです。
- QUERY: 特定の並列度がこの句に指定されていないかぎり、後続の問合せは、デフォルトの並列度で実行されます。

PARALLEL integer

並列度を明示的に指定する整数を指定します。

- FORCE DDLでは、並列度は後続のDDL文のパラレル句を上書きします。
- FORCE DMLおよびFORCE QUERYでは、並列度は、データ・ディクショナリの表に格納されている現行の並列度を上書きします。
- ヒントによって文に指定される並列度は、強制的に実行される並列度を上書きします。

次のDML操作は、この句に関係なくパラレル化されません。

- クラスタ化表に対する操作
- データベースまたはパッケージの状態を読み書きする埋込みファンクションを使用した操作
- 起動する可能性のあるトリガーを使用した表に対する操作
- オブジェクト型、LONGまたはLOBデータ型が含まれている表またはスキーマ・オブジェクトでの操作

RESUMABLE句

これらの句を使用すると、再開可能な領域割当てを使用可能および使用禁止にできます。この機能によって、領域不足のエラー条件が発生した場合に操作は停止され、エラー条件が修復されたときに中断したところから自動的に再開されます。

ノート:

再開可能な領域割当ては、ローカル管理表領域での操作で、完全にサポートされます。ディクショナリ管理表領域を使用する場合は、制限事項があります。制限事項の詳細は、『Oracle Database 管理者ガイド』を参照してください。

ENABLE RESUMABLE

この句を使用すると、セッションに対する再開可能な領域割当てを使用可能にできます。

TIMEOUT

TIMEOUTを使用すると、エラー条件が修復されるまで操作を停止する時間を秒単位で指定できます。エラー条件がTIMEOUTで指定した時間までに修復されない場合は、停止操作は異常終了します。

NAME

NAMEを使用すると、ユーザー定義のテキスト文字列を指定でき、再開可能モードのセッション中に発行される文の識別に有効です。USER_RESUMABLEデータ・ディクショナリ・ビューおよびDBA_RESUMABLEデータ・ディクショナリ・ビューに、テキスト文字列が挿入されます。NAMEを指定しない場合は、デフォルト文字列「`User username(userid), Session sessionid, Instance instanceid`」が挿入されます。

関連項目:

データ・ディクショナリ・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

DISABLE RESUMABLE

この句を使用すると、セッションに対する再開可能な領域割当てを使用禁止にできます。

SHARD DDL句

これらの句は、シャード・データベースに接続している場合にのみ有効です。これらを使用すると、セッションで発行されるDDLが、シャード・カタログ・データベースおよびすべてのシャードに対して発行されるか、シャード・カタログ・データベースに対してのみ発行されるかを制御できます。

- `ENABLE SHARD DDL`を指定すると、セッションで発行されるDDLは、シャード・カタログ・データベースおよびすべてのシャードに対して発行されます。このモードは、SDBユーザー(シャード・カタログ・データベースおよびすべてのシャードに存在するユーザー)のデフォルトです。
- `DISABLE SHARD DDL`を指定すると、セッションで発行されるDDLは、シャード・カタログ・データベースに対してのみ発行されます。このモードは、ローカル・ユーザー(シャード・カタログ・データベースのみに存在するユーザー)のデフォルトです。

関連項目:

Oracle Shardingの使用

SYNC WITH PRIMARY

この句を使用すると、フィジカル・スタンバイ・データベース上のREDO Applyをプライマリ・データベースに同期できます。この句を`ALTER SESSION`文とともに使用すると、文の発行時にスタンバイによって受信されたすべてのREDOデータがREDO Applyで適用されるまでブロックされます。スタンバイ・データベースのREDO転送状態が`SYNCHRONIZED`でない場合、またはREDO Applyがアクティブでない場合、この句はエラーを戻し、同期は実行されません。

関連項目:

このセッション・パラメータの詳細は、『Oracle Data Guard概要および管理』を参照してください。

alter_session_set_clause

`alter_session_set_clause`を使用すると、現行セッションの初期化パラメータ値またはエディションを設定できます。

初期化パラメータ

この句を使用して、次の2種類のパラメータを設定できます。

- ALTER SESSION文の有効範囲内で動的な初期化パラメータ(「初期化パラメータおよびALTER SESSION」を参照)
- セッション・パラメータ(「セッション・パラメータおよびALTER SESSION」を参照)

同じalter_session_set_clauseで複数のパラメータに対する値を設定できます。

EDITION

EDITION = editionを指定すると、指定したエディションをデータベース・セッションのエディションとして設定できます。editionに対するUSEオブジェクト権限が必要であり、editionは、あらかじめ作成されている必要があります。このエディションはUSABLEである必要があります。

この文が正常に実行された場合、エディション化可能なパッケージに対応するPL/SQLパッケージ状態はデータベースにより廃棄されますが、エディション化が可能でないパッケージに対応するパッケージ状態は保持されます。

また、SQL*Plus CONNECTコマンドのEDITIONパラメータを使用すると、起動時に現行セッションのエディションを設定できます。ただし、再帰的SQLまたはPL/SQLブロックでは、ALTER SESSION SET EDITION文を指定できません。

次の問合せを使用して、現行セッションで使用中のエディションを確認できます。

```
SELECT SYS_CONTEXT('USERENV', 'CURRENT_EDITION_NAME') FROM DUAL;
```

関連項目:

エディションの詳細は「CREATE EDITION」、エディションをUSABLEに指定する方法の詳細は『Oracle Database PL/SQL言語リファレンス』を参照してください。

CONTAINER

マルチテナント・コンテナ・データベース(CDB)でこの句を使用すると、container_nameで指定したコンテナに切り替えることができます。

この句を使用するには、SET CONTAINER権限(共通に付与されている権限か、container_nameでローカルに付与されている権限のいずれか)を持つ共通ユーザーである必要があります。

container_nameには、次のいずれかを指定します。

- ルートに切り替えるにはCDB$ROOT
- シードに切り替えるにはPDB$SEED
- プラガブル・データベース(PDB)に切り替えるにはPDB名。CDB内のPDBの名前を表示するには、DBA_PDBSビューを問い合せます。

現行のセッションの接続先のコンテナを確認するには、SQL*Plus SHOW CON_NAMEコマンドまたは次のSQL問合せを使用します。

```
SELECT SYS_CONTEXT('USERENV', 'CON_NAME') FROM DUAL;
```

SERVICE

デフォルトでは、コンテナに切り替えると、セッションはコンテナのデフォルト・サービスを使用します。コンテナの別のサービスを使用する場合は、SERVICE句を指定します。service_nameに、使用するサービスの名前を指定します。

関連項目:

特定のコンテナへの切替えの詳細は、『Oracle Database管理者ガイド』を参照してください。

ROW ARCHIVAL VISIBILITY

この句を使用すると、セッションに対する行アーカイブの可視性を構成できます。この句によりインデータベース・アーカイブを実装でき、表の行をアクティブまたはアーカイブ済として指定できます。その後、表内のアクティブな行のみを対象に問合せを実行できます。

- ACTIVEを指定すると、行アーカイブが有効な表の問合せ実行時にアクティブな行のみが考慮されます。これはデフォルトです。
- ALLを指定すると、行アーカイブが有効な表の問合せ実行時にすべての行が考慮されます。

この句は、行アーカイブが無効な表の問合せには影響しません。

関連項目:

- 新しい表の行アーカイブを有効にする方法を学習するには、「CREATE TABLE」の「ROW ARCHIVAL」句を参照してください。
- 既存の表の行アーカイブを有効または無効にする方法を学習するには、「ALTER TABLE」の「[NO] ROW ARCHIVAL」句を参照してください。
- インデータベース・アーカイブの詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

DEFAULT_COLLATION

この句を使用すると、セッションのデフォルト照合を設定できます。

- collation_nameを使用して、セッションのデフォルト照合を指定します。有効な名前付き照合または疑似照合の名前を指定できます。この照合は有効スキーマのデフォルト照合になります。この照合は、セッション中に任意のスキーマでそれ以降に作成される表、ビューおよびマテリアライズド・ビューに割り当てられます。セッションのデフォルト照合は、DBリンクを使用して現行のセッションに接続されているリモート・セッションには伝播されません。
- NONEを指定した場合は、セッションのデフォルト照合は設定されません。この場合は、特定のスキーマのデフォルト照合がそのスキーマの有効スキーマのデフォルト照合になります。このデフォルト照合は、セッション中にそのスキーマでそれ以降に作成される表、ビューおよびマテリアライズド・ビューに割り当てられます。

前述のいずれの場合も、有効スキーマのデフォルト照合を上書きし、特定の表、マテリアライズド・ビューまたはビューのCREATEまたはALTER文のDEFAULT COLLATION句を指定することで、その表、マテリアライズド・ビューまたはビューにデフォルト照合を割り当てることができます。

有効スキーマのデフォルト照合は、DDL文CREATE FUNCTION、CREATE PACKAGE、CREATE PROCEDURE、CREATE TRIGGERおよびCREATE TYPEにも影響します。これらの文の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

次の文を使用して、セッションのデフォルト照合を問い合せることができます。

```
SELECT SYS_CONTEXT('USERENV', 'SESSION_DEFAULT_COLLATION') FROM DUAL;
```

SET DEFAULT_COLLATION句を指定できるのは、COMPATIBLE初期化パラメータが12.2以上に設定されており、

MAX_STRING_SIZE初期化パラメータがEXTENDEDに設定されている場合のみです。

関連項目:

スキーマのデフォルト照合の詳細は、「CREATE USER」の「DEFAULT COLLATION句」を参照してください。

ノート:

セッションの有効スキーマのデフォルト照合を、セッション・パラメータ NLS_SORT と混同しないでください。有効スキーマのデフォルト照合は、表、ビューおよびマテリアライズド・ビューの作成時に、それらのデフォルトのデータ・バインドされた照合を決定するために DDL 文で使用されます。セッション・パラメータ NLS_SORT は、決定された照合が USING_NLS_COMP や USING_NLS_SORT などの疑似照合である SQL 操作を含む、問合せ、DML 文または PL/SQL コードの実行時に使用される名前付き照合を指定します。詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

初期化パラメータおよびALTER SESSION

初期化パラメータの中には、ALTER SESSIONの有効範囲内で動的であるものがあります。このようなパラメータをALTER SESSIONを使用して設定したときに、設定した値が有効であるのは現行セッションの終了までとなります。パラメータをALTER SESSION文を使用して変更できるかどうかを調べるには、V$PARAMETER動的パフォーマンス・ビューのISSES_MODIFIABLE列を問い合せます。

ノート:

初期化パラメータの値を変更する前に、『Oracle Database リファレンス』を参照してください。

ALTER SESSIONで設定可能な一部のパラメータは、初期化パラメータではありません。初期化パラメータ・ファイルではなく、ALTER SESSIONでのみ設定可能です。これらのセッション・パラメータの詳細は、「セッション・パラメータおよびALTER SESSION」を参照してください。

セッション・パラメータおよびALTER SESSION

次のパラメータは、セッション・パラメータであり、初期化パラメータではありません。

CONSTRAINT[S]

構文:

```
CONSTRAINT[S] = { IMMEDIATE | DEFERRED | DEFAULT }
```

CONSTRAINT[S]は、遅延可能制約によって指定された条件を、いつ適用するかを指定します。

- IMMEDIATEを設定すると、遅延可能な制約によって指定される条件は、各DML文の直後にチェックされます。これは、セッションの各トランザクションの開始時に、SET CONSTRAINTS ALL IMMEDIATE文を発行することと同じです。

- DEFERREDを設定すると、遅延可能な制約によって指定される条件は、トランザクションのコミット時にチェックされます。これは、セッションの各トランザクションの開始時に、SET CONSTRAINTS ALL DEFERRED文を発行することと同じです。
- DEFAULTを設定すると、すべての制約は各トランザクションの開始時に、DEFERREDまたはIMMEDIATEの初期状態にリストアされます。

CURRENT_SCHEMA

構文:

```
CURRENT_SCHEMA = schema
```

CURRENT_SCHEMAは、セッションの現行のスキーマを、指定したスキーマに変更します。セッション中のスキーマ・オブジェクトに対する後続の未修飾の参照は、この指定したスキーマ内でオブジェクトに変換されます。この設定は、現行のセッションの存続期間中、またはALTER SESSION SET CURRENT_SCHEMA文を再発行するまで保持されます。

この設定を利用すると、現行のユーザーのものではないスキーマにあるオブジェクトに対する操作を実行するときも、オブジェクトをスキーマ名で修飾することは不要になります。この設定を指定すると、現行スキーマは変更されますが、セッション・ユーザーや現行ユーザーが変更されることはなく、セッションに対する追加のシステム権限やオブジェクト権限がセッション・ユーザーに付与されることもありません。

ERROR_ON_OVERLAP_TIME

構文:

```
ERROR_ON_OVERLAP_TIME = {TRUE | FALSE}
```

ERROR_ON_OVERLAP_TIMEパラメータには、Oracle Databaseが不明瞭な境界日時値(日時が標準か夏時間かが明確でない場合)を処理する方法を指定します。

- TRUEを指定すると、不明瞭なオーバラップ・タイムスタンプに対してエラーが戻されます。
- FALSEを指定すると、不明瞭なオーバラップ・タイムスタンプは標準時刻のデフォルトになります。これはデフォルトです。

境界日時値の詳細は、「夏時間のサポート」を参照してください。

FLAGGER

構文:

```
FLAGGER = { ENTRY | OFF }
```

FLAGGERパラメータは、FIPSのフラグ付け(Federal Information Processing Standard 127-2で規定)を指定するものであり、このフラグを設定すると、SQL-92のエントリ・レベル(正式には、ANSI X3.135-1992。現在、この規格はSQL:2016に置き換えられています)の拡張であるSQL文が発行されたときにエラー・メッセージが生成されます。FLAGGERはセッション・パラメータであり、初期化パラメータではありません。

セッションでフラグ付けが設定されると、これに続くALTER SESSION SET FLAGGER文は成功しますが、ORA-00097のメッセージが生成されます。このため、セッションを切断しなくてもFIPSのフラグ付けを変更できます。OFFを設定した場合、フラグ付けの使用は停止されます。

関連項目:

現行のANSI SQL規格に対するOracleの準拠の詳細は、「Oracleと標準SQL」を参照してください。

INSTANCE

構文:

```
INSTANCE = integer
```

INSTANCEパラメータを設定すると、自分のインスタンスに接続している場合と同様に、別のインスタンスにもアクセスできます。INSTANCEはセッション・パラメータであり、初期化パラメータではありません。Oracle Real Application Clusters(Oracle RAC)環境では、このパラメータの設定に基づき、各Oracle RACインスタンスで、最適なDMLパフォーマンスを実現するようにディスク領域の静的または動的な所有権が保持されます。

ISOLATION_LEVEL

構文:

```
ISOLATION_LEVEL = {SERIALIZABLE | READ COMMITTED}
```

ISOLATION_LEVELパラメータは、データベースを変更するトランザクションがどのように処理されるかを指定します。ISOLATION_LEVELはセッション・パラメータであり、初期化パラメータではありません。

- SERIALIZABLEを設定すると、セッション内のトランザクションは、SQL規格に規定されているとおりシリアル化可能トランザクション分離モードを使用します。シリアル化可能トランザクションが行を更新するDML文を実行する場合、現在の更新対象の行がそのシリアル化可能トランザクションの開始時にコミットされていない別のトランザクションによって更新されていたときは、そのDML文は失敗します。シリアル化可能トランザクションは、同一トランザクション内で行った更新を確認できます。
- READ COMMITTEDを設定すると、セッション内のトランザクションは、Oracle Databaseトランザクションのデフォルトの動作を行います。別のトランザクションで行ロックを保持しておく必要があるDMLがトランザクションに指定されていると、DML文は行ロックが解除されるまで待ち状態になります。

ノート:

シリアル化可能トランザクションは、遅延セグメント作成または時間隔パーティションでは動作しません。セグメントが作成されていない空の表、またはセグメントが存在しない時間隔パーティション表のパーティションにデータを挿入しようとすると、エラーになります。

STANDBY_MAX_DATA_DELAY

構文:

```
STANDBY_MAX_DATA_DELAY =  { integer | NONE }
```

Active Data Guard環境では、このセッション・パラメータを設定すると、管理ユーザー以外のユーザーがリアルタイム問合せモードのフィジカル・スタンバイ・データベースに発行した問合せに関して、セッション固有の適用ラグの許容値を秒単位で指定できます。この機能によって、スタンバイ・データベースが許容できないほど失効したかどうかを検出できるため、プライマリ・データベースからフィジカル・スタンバイ・データベースに問合せを安全に移動できます。

STANDBY_MAX_DATA_DELAYがデフォルト値NONEに設定されている場合、フィジカル・スタンバイ・データベースに発行された問合せは、そのデータベースの適用ラグに関係なく実行されます。

STANDBY_MAX_DATA_DELAYを0(ゼロ)以外の値に設定すると、フィジカル・スタンバイ・データベースに発行された問合せは、適用ラグがSTANDBY_MAX_DATA_DELAY以下の場合のみ実行されます。それ以外の場合、ORA-3172エラーが戻され、適用ラグが大きすぎることがクライアントに警告されます。

STANDBY_MAX_DATA_DELAYを0に設定すると、フィジカル・スタンバイ・データベースに発行された問合せは、プライマリ・データベースに対して発行された場合と同じ結果を戻すことが保証されます。ただし、スタンバイ・データベースがプライマリ・データベースよりも遅れている場合、ORA-3172エラーが戻されます。

関連項目:

Active Data Guardおよびこのセッション・パラメータの使用方法の詳細は、『Oracle Data Guard概要および管理』を参照してください。

TIME_ZONE

構文:

```
TIME_ZONE =  '[+ | -] hh:mi'
             | LOCAL
             | DBTIMEZONE
             | 'time_zone_region'
```

TIME_ZONEパラメータには、現行のSQLセッションのデフォルトのローカル・タイムゾーン・オフセットまたは地域名を指定します。TIME_ZONEはセッション・パラメータであり、初期化パラメータではありません。現行のセッションのタイムゾーンを確認するには、組込み関数SESSIONTIMEZONEを問い合せます(「SESSIONTIMEZONE」を参照)。

- UTCの前または後の時間および分を示す書式マスク('[+|-]hh:mi')を指定します。hh:miの有効範囲は、-12:00から+14:00です。
- LOCALを指定すると、現行のSQLセッションのデフォルトのローカル・タイムゾーン・オフセットが、現行のSQLセッションを起動したときに構築された、元のデフォルトのローカル・タイムゾーン・オフセットに設定されます。
- DBTIMEZONEを指定すると、現行セッションのタイムゾーンはデータベースのタイムゾーンの値と一致するように設定されます。この設定を指定した場合は、DBTIMEZONEファンクションによってデータベースのタイムゾーンがUTCオフセットまたはタイムゾーン地域名として戻されるようになりますが、どちらで戻されるかはデータベースのタイムゾーンの設定によって決まります。
- 有効なtime_zone_regionを指定します。有効なタイムゾーン地域名を表示するには、V$TIMEZONE_NAMES動的パフォーマンス・ビューのTZNAME列を問い合せます。この設定を指定する場合、SESSIONTIMEZONEファンクションは地域の名前を戻します。

ノート:

夏時間機能には、タイムゾーン地域名が必要です。この名前は、大小 2 つのタイムゾーン・ファイルに格納されます。これらのファイルのうち、使用する環境および使用する Oracle Database のリリースに応じて、いずれか一方がデフォルトのファイルになります。タイムゾーン・ファイルおよびタイムゾーン名の詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

関連項目:

両方のファイル内のすべてのタイムゾーン地域名のリストは、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

ノート:

環境変数 ORA_SDTZ を使用して、クライアント・セッションのデフォルトのタイムゾーンを設定することもできます。この変数の詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

USE_PRIVATE_OUTLINES

構文:

```
USE_PRIVATE_OUTLINES = { TRUE | FALSE | category_name }
```

USE_PRIVATE_OUTLINESパラメータを使用すると、プライベート・アウトラインの使用を制御することができます。このパラメータが使用可能で、アウトライン化されたSQL文が発行された場合、オプティマイザは、USE_STORED_OUTLINESが使用可能なときに使用されるパブリック領域ではなく、そのセッションのプライベート領域からアウトラインを検索します。そのセッションのプライベート領域にアウトラインが存在しない場合、オプティマイザは、文のコンパイルにアウトラインを使用しません。
USE_PRIVATE_OUTLINESは、初期化パラメータではありません。

- TRUEに設定すると、要求をコンパイルするときに、オプティマイザはDEFAULTカテゴリのストアド・プライベート・アウトラインを使用します。
- FALSEに設定すると、オプティマイザはストアド・プライベート・アウトラインを使用しません。これはデフォルトです。USE_STORED_OUTLINESが使用可能な場合、オプティマイザはストアド・パブリック・アウトラインを使用します。
- category_nameに設定すると、要求をコンパイルするときに、オプティマイザはcategory_nameカテゴリのストアド・アウトラインを使用します。

USE_PRIVATE_OUTLINESの制限事項

USE_STORED_OUTLINESが有効な場合は、このパラメータを有効化できません。

USE_STORED_OUTLINES

ノート:

ストアド・アウトラインは非推奨になりました。ストアド・アウトラインは、下位互換性を保つために今でもサポートされています。ただし、かわりに SQL 計画管理を使用することをお薦めします。SQL 計画管理の詳細は、Oracle Database SQL チューニング・ガイドを参照してください。

構文:

```
USE_STORED_OUTLINES = { TRUE | FALSE | category_name }
```

USE_STORED_OUTLINESパラメータは、オプティマイザが実行計画を生成するためにストアド・パブリック・アウトラインを使用するかどうかを決定します。USE_STORED_OUTLINESは、初期化パラメータではありません。

- TRUEに設定すると、要求をコンパイルするときに、オプティマイザはDEFAULTカテゴリのストアド・アウトラインを使用します。
- FALSEに設定すると、オプティマイザはストアド・アウトラインを使用しません。これはデフォルトです。
- category_nameに設定すると、要求をコンパイルするときに、オプティマイザはcategory_nameカテゴリのストアド・アウトラインを使用します。

USED_STORED_OUTLINESの制限事項

USE_PRIVATE_OUTLINESが有効な場合は、このパラメータを有効化できません。

例

パラレルDMLの有効化: 例

次の文を発行して、現行のセッションでパラレルDMLモードを有効にします。

```
ALTER SESSION ENABLE PARALLEL DML;
```

分散トランザクションの強制実行: 例

次のトランザクションは、データベース・リンクremoteによって識別されるデータベース上のemployees表に従業員のレコードを挿入し、localによって識別されるデータベース上のemployees表から従業員のレコードを削除します。

```
ALTER SESSION
   ADVISE COMMIT;
INSERT INTO employees@remote
   VALUES (8002, 'Juan', 'Fernandez', 'juanf@example.com', NULL,
   TO_DATE('04-OCT-1992', 'DD-MON-YYYY'), 'SA_CLERK', 3000,
   NULL, 121, 20);
ALTER SESSION
   ADVISE ROLLBACK;
DELETE FROM employees@local
   WHERE employee_id = 8002;
COMMIT;
```

このトランザクションには、ADVISE句を指定したALTER SESSION文が2つあります。このトランザクションが状態不明(インダウト)になった場合、remoteには、最初に指定したALTER SESSION文によってアドバイス'COMMIT'が送信され、localには、2番目の文によってアドバイス'ROLLBACK'が送信されます。

「データベース・リンクをクローズする: 例」

次の文は、データベース・リンクを使用しているlocalデータベース上のjobs表を更新し、このトランザクションをコミットして、データベース・リンクを明示的にクローズします。

```
UPDATE jobs@local SET min_salary = 3000
   WHERE job_id = 'SH_CLERK';
COMMIT;
ALTER SESSION
   CLOSE DATABASE LINK local;
```

日付書式の動的変更: 例

次の文は、セッションのデフォルトの日付形式を動的に'YYYY MM DD-HH24:MI:SS'に変更します。

```
ALTER SESSION
   SET NLS_DATE_FORMAT = 'YYYY MM DD HH24:MI:SS';
```

変更後は、新しい日付書式が次のように適用されます。

```
SELECT TO_CHAR(SYSDATE) Today
   FROM DUAL;
TODAY
-------------------
2001 04 12 12:30:38
```

日付言語の動的変更: 例

次の文は、日付書式要素の言語をフランス語に変更します。

```
ALTER SESSION
   SET NLS_DATE_LANGUAGE = French;
SELECT TO_CHAR(SYSDATE, 'Day DD Month YYYY') Today
   FROM DUAL;
TODAY
---------------------------
Jeudi    12 Avril     2001
```

ISO通貨の変更: 例

次の文は、ISO通貨記号を、地域がアメリカの場合のISO通貨記号に動的に変更します。

```
ALTER SESSION
   SET NLS_ISO_CURRENCY = America;
SELECT TO_CHAR( SUM(salary), 'C999G999D99') Total
   FROM employees;
TOTAL
------------------
     USD694,900.00
```

小数点文字および桁区切りの変更: 例

次の文は、小数点文字をカンマ(,)に、桁区切りをピリオド(.)に動的に変更します。

```
ALTER SESSION SET NLS_NUMERIC_CHARACTERS = ',.' ;
```

これらの数値書式要素を使用した場合、新しい文字が戻ります。

```
ALTER SESSION SET NLS_CURRENCY = 'FF';
SELECT TO_CHAR( SUM(salary), 'L999G999D99') Total FROM employees;
TOTAL
---------------------
         FF694.900,00
```

NLS通貨の変更: 例

次の文は、各国通貨記号をDMに動的に変更します。

```
ALTER SESSION
   SET NLS_CURRENCY = 'DM';
SELECT TO_CHAR( SUM(salary), 'L999G999D99') Total
   FROM employees;
TOTAL
---------------------
         DM694.900,00
```

NLS言語の変更: 例

次の文は、エラー・メッセージが表示される言語をフランス語に動的に変更します。

```
ALTER SESSION
   SET NLS_LANGUAGE = FRENCH;
Session modifiee.
SELECT * FROM DMP;
ORA-00942: Table ou vue inexistante
```

言語ソート順の変更: 例

次の文は、言語ソート順をスペイン語に動的に変更します。

```
ALTER SESSION
   SET NLS_SORT = XSpanish;
```

これによって、文字の値はスペイン語のソート順序に基づいてソートされます。

クエリー・リライトの有効化：例

次の文は、明示的に無効にされていないすべてのマテリアライズド・ビューに対する現行のセッションのクエリー・リライトを有効にします。

```
ALTER SESSION
  SET QUERY_REWRITE_ENABLED = TRUE;
```

12 SQL文: ALTER SYNONYMからCOMMENT

この章では、次のSQL文について説明します。

- ALTER SYNONYM
- ALTER SYSTEM
- ALTER TABLE
- ALTER TABLESPACE
- ALTER TABLESPACE SET
- ALTER TRIGGER
- ALTER TYPE
- ALTER USER
- ALTER VIEW
- ANALYZE
- ASSOCIATE STATISTICS
- AUDIT (従来型監査)
- AUDIT (統合監査)
- CALL
- COMMENT

ALTER SYNONYM

目的

ALTER SYNONYM文を使用すると、既存のシノニムを変更できます。

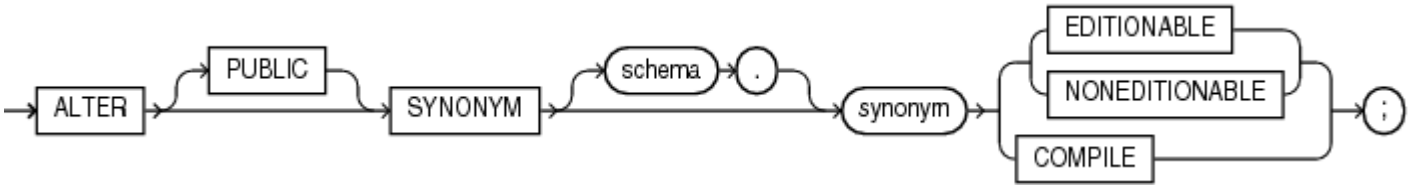
前提条件

他のユーザーのスキーマに含まれるシノニムを変更する場合は、CREATE ANY SYNONYMシステム権限とDROP ANY SYNONYMシステム権限が必要です。

PUBLICシノニムを変更する場合、CREATE PUBLIC SYNONYMシステム権限とDROP PUBLIC SYNONYMシステム権限が必要です。

構文

alter_synonym::=



セマンティクス

PUBLIC

synonymがパブリック・シノニムの場合は、PUBLICを指定します。この句は、パブリック・シノニムをプライベート・シノニムに変更する(または、その逆に変更する)場合には使用できません。

schema

シノニムが含まれているスキーマを指定します。schemaを指定しない場合、シノニムは自分のスキーマ内にあるとみなされます。

synonym

変更するシノニムの名前を指定します。

EDITIONABLE | NONEDITIONABLE

この句を使用すると、schemaのスキーマ・オブジェクト・タイプSYNONYMのエディションニングが後で有効化されたときに、そのシノニムをエディション・オブジェクトにするか非エディション・オブジェクトにするかを指定できます。デフォルトは、EDITIONABLEです。エディション・オブジェクトと非エディション・オブジェクトの変更の詳細は、『Oracle Database開発ガイド』を参照してください。

EDITIONABLE | NONEDITIONABLEの制限事項

これらの句は、パブリック・シノニムには指定できません。PUBLICスキーマのオブジェクト型SYNONYMについては、エディションニングが常に有効化されています。

COMPILE

この句を使用すると、synonymをコンパイルできます。シノニムによってシノニムのターゲット・オブジェクトへの依存関係が設定され、ターゲット・オブジェクトが変更または削除されるとシノニムも無効になります。無効のシノニムをコンパイルすると、そのシノニムは再度有効になります。

ノート:

シノニムが有効か無効かを判断するには、ALL_、DBA_、および USER_OBJECTS データ・ディクショナリ・ビューの STATUS 列を問い合せます。

例

次の例では、「CREATE SYNONYM」の「例」で作成したシノニムを変更します。

次の文では、シノニムofficesをコンパイルします。

```
ALTER SYNONYM offices COMPILE;
```

次の文では、パブリック・シノニムemp_tableをコンパイルします。

```
ALTER PUBLIC SYNONYM emp_table COMPILE;
```

次の文では、シノニムofficesを含むスキーマのスキーマ・オブジェクト・タイプSYNONYMのエディショニングが後から有効化されたときにも、シノニムofficesが非エディション・オブジェクトの状態を維持するようにします。

```
ALTER SYNONYM offices NONEDITIONABLE;
```

ALTER SYSTEM

目的

ALTER SYSTEM文を使用すると、Oracle Databaseインスタンスを動的に変更できます。この設定は、データベースがマウントされているかぎり有効です。

マルチテナント・コンテナ・データベース(CDB)でALTER SYSTEM文を使用する場合、CDB全体を変更する句および特定のプラガブル・データベース(PDB)を変更する句を指定できます。

関連項目:

CDBでのALTER SYSTEM文の使用方法の詳細は、『Oracle Database管理者ガイド』を参照してください。

前提条件

RELOCATE CLIENT句を指定するには、AS SYSASMとして認証されている必要があります。

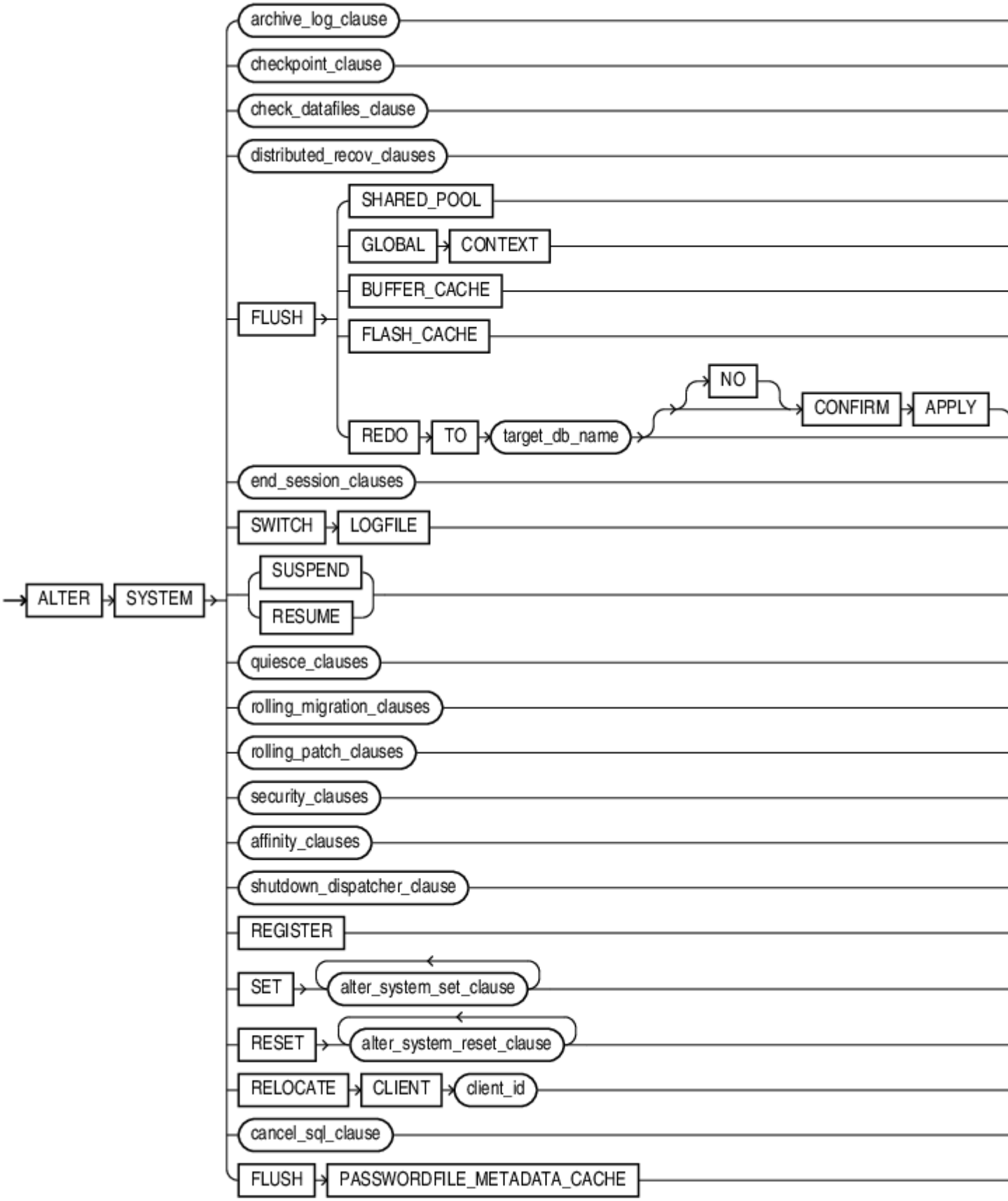
他をすべての句を指定するには、ALTER SYSTEMシステム権限が必要です。

CDBに接続しているときには、次の条件が適用されます。

- CDB全体を変更する場合は、現在のコンテナがルートである必要があります。また、共通に付与されているALTER SYSTEM権限が必要です。
- PDBを変更する場合は、現在のコンテナがそのPDBである必要があります。また、ALTER SYSTEM権限(共通に付与されている権限か、そのPDBでローカルに付与されている権限のいずれか)が必要です。

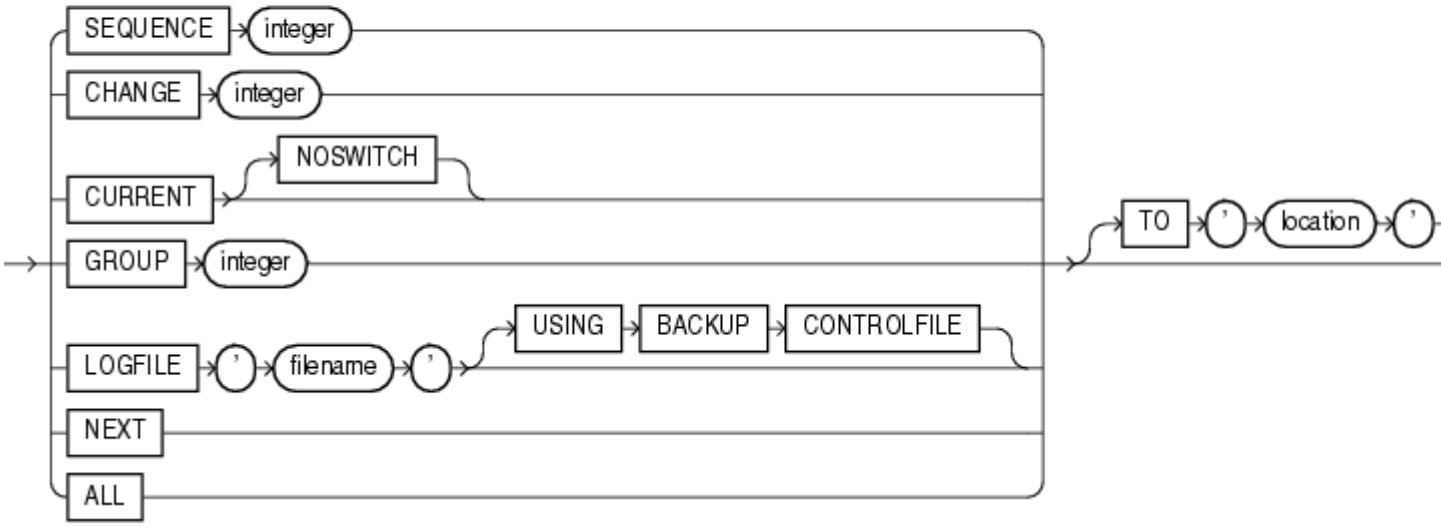
構文

alter_system::=

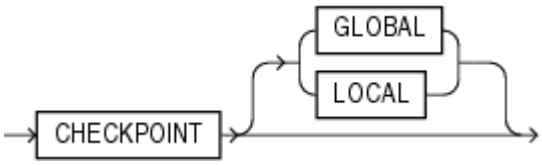


(archive_log_clause::=、checkpoint_clause::=、check_datafiles_clause::=、distributed_recov_clauses::=、end_session_clauses::=、quiesce_clauses::=、rolling_migration_clauses::=、rolling_patch_clauses::=、security_clauses::=、shutdown_dispatcher_clause::=、alter_system_set_clause::=、alter_system_reset_clause::=)

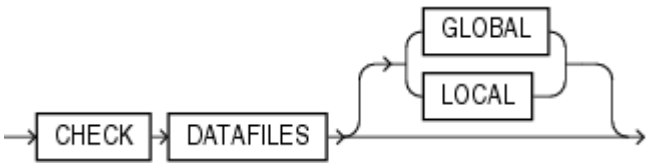
archive_log_clause::=



checkpoint_clause::=

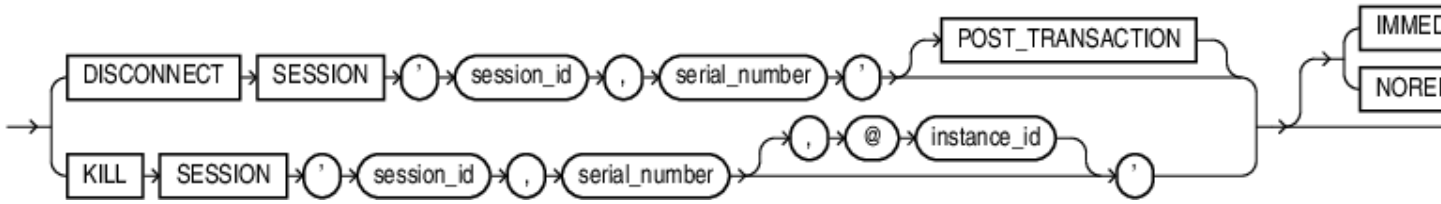


check_datafiles_clause::=

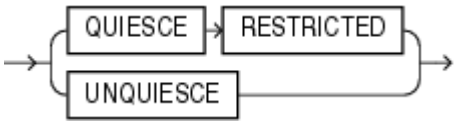


distributed_recov_clauses::=

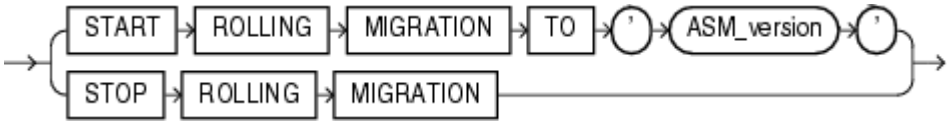
end_session_clauses::=



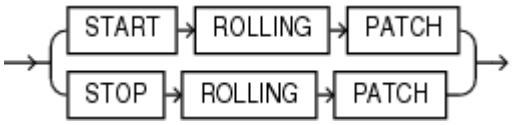
quiesce_clauses::=



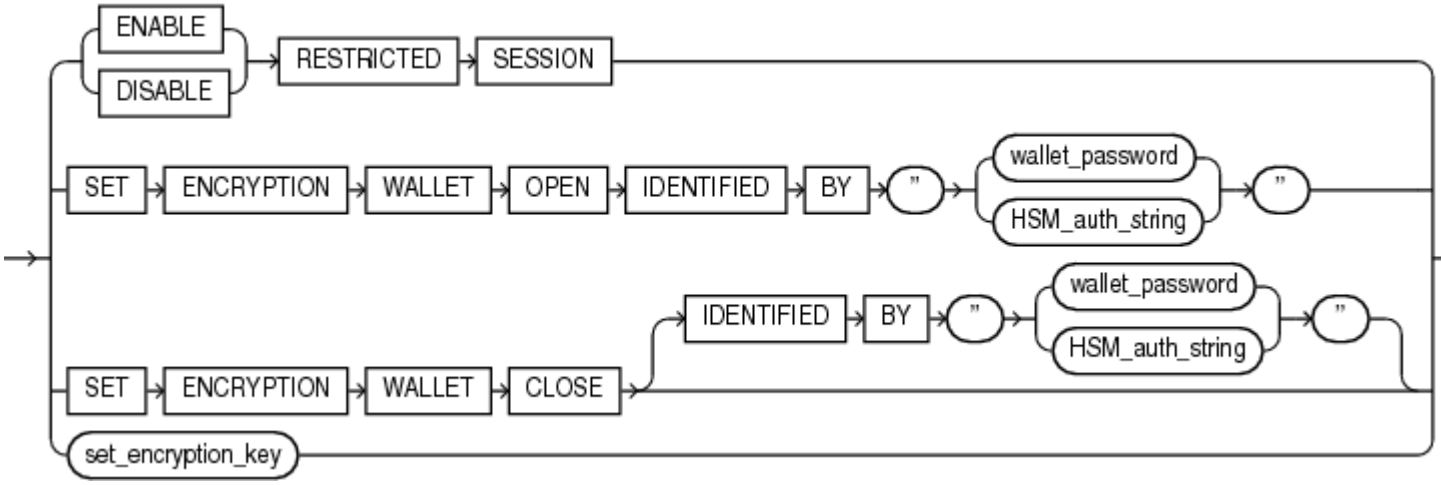
rolling_migration_clauses::=



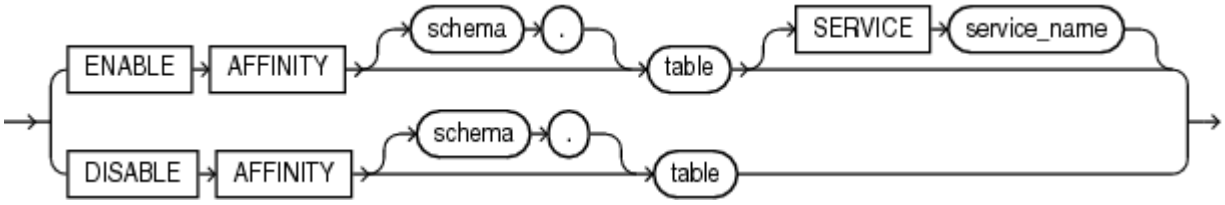
rolling_patch_clauses::=



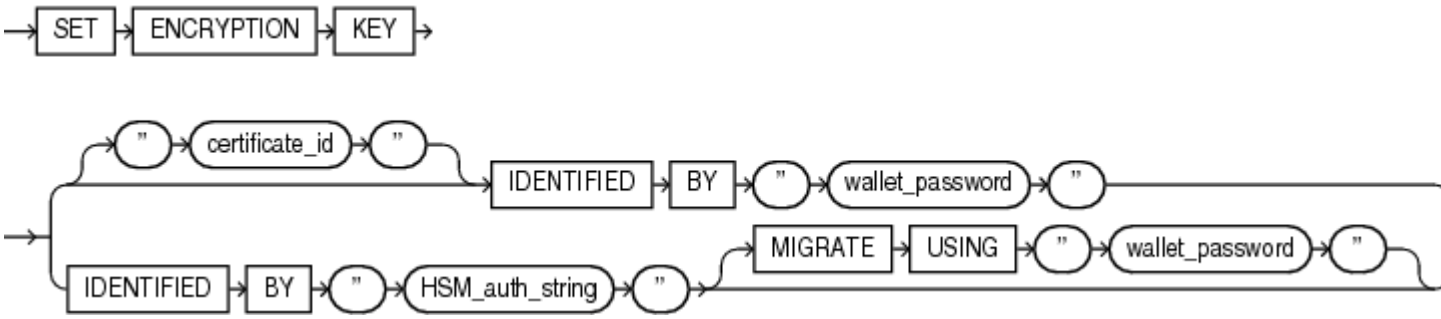
security_clauses::=



affinity_clauses::=

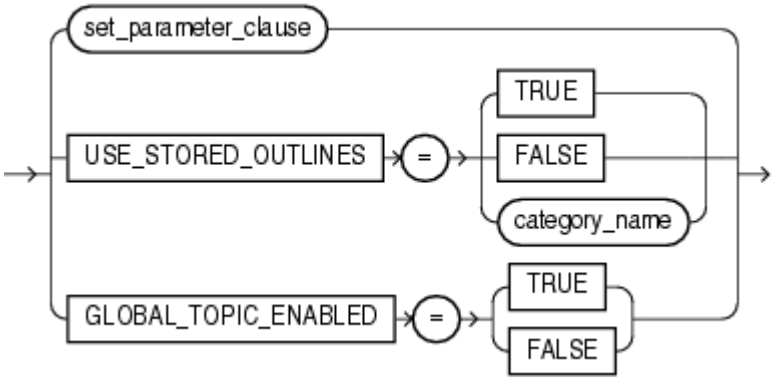


set_encryption_key::=

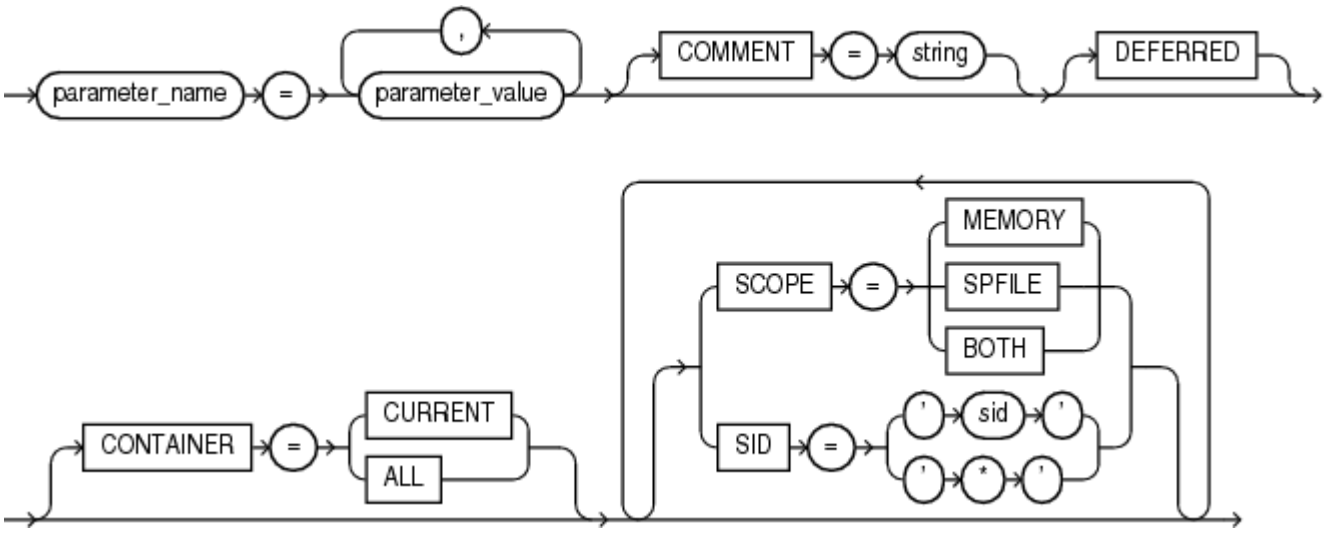


shutdown_dispatcher_clause::=

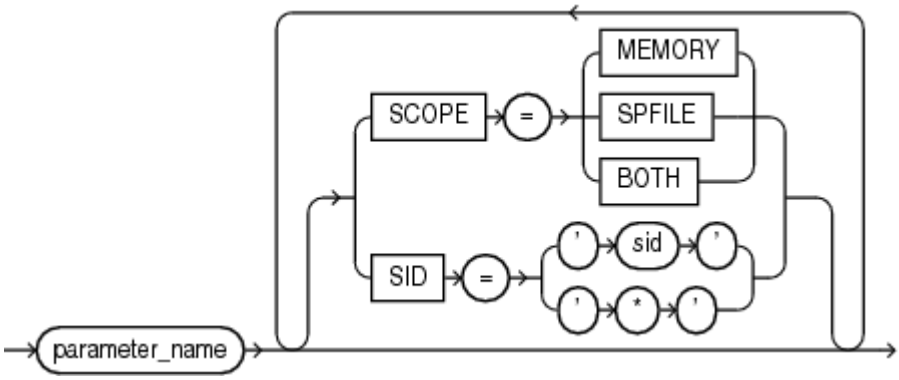
alter_system_set_clause::=



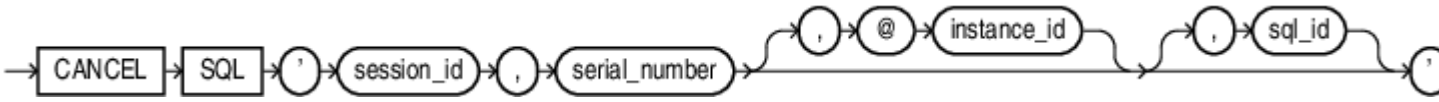
set_parameter_clause::=



alter_system_reset_clause::=



cancel_sql_clause::=



セマンティクス

archive_log_clause

archive_log_clauseを使用すると、REDOログ・ファイルを手動でアーカイブしたり、自動アーカイブを使用可能または使用禁止にすることができます。この句を使用する場合、インスタンスでデータベースをマウントする必要があります。特に指定がないかぎり、データベースはオープンまたはクローズできます。

INSTANCE句

この句が関係するのは、Oracle Real Application Clusters(Oracle RAC)を使用している場合のみです。REDOログ・ファイル・グループをアーカイブするインスタンスの名前を指定します。インスタンス名は最大80文字の文字列です。指定したインスタンスにマップするスレッドはOracle Databaseによって自動的に決定され、対応するREDOログ・ファイル・グループがアーカイブされます。指定したインスタンスにマップされているスレッドがない場合、エラーが戻されます。

SEQUENCE句

SEQUENCEを指定すると、指定したスレッド内のログ順序番号integerによって識別されるオンラインREDOログ・ファイル・グループを手動でアーカイブできます。THREADパラメータを指定しなかった場合、インスタンスに割り当てられているスレッドから、指定したグループがアーカイブされます。

CHANGE句

CHANGEを指定すると、指定されたスレッド内の、integerで指定されたシステム変更番号(SCN)を持つREDOログ・エントリが格納されているオンラインREDOログ・ファイル・グループを、手動でアーカイブできます。このSCNが現行のREDOログ・ファイル・グループ内にある場合は、ログ・スイッチが実行されます。THREADパラメータを省略した場合は、使用可能な状態にあるすべてのスレッドから、このSCNが含まれているグループがアーカイブされます。

インスタンスでデータベースをオープンしている場合にのみ、この句を使用できます。

CURRENT句

CURRENTを指定すると、ログ・スイッチを強制的に発生させ、指定したスレッドの現行のREDOログ・ファイル・グループを手動でアーカイブできます。THREADパラメータを指定しない場合、すべての使用可能なスレッドから、現行のログ以前のログも含むすべてのREDOログ・ファイル・グループがアーカイブされます。データベースがオープンしているときのみ、CURRENTを指定できます。

NOSWITCH

NOSWITCHを指定すると、ログ・スイッチの強制実行なしで現行のREDOログ・ファイル・グループを手動でアーカイブできます。この設定は、プライマリ・データベースが停止したときに、データ分岐が発生しないようにするために、主にスタンバイ・データベースで使用されます。データ分岐は、プライマリ・データベースに障害が発生した場合に、データが消失する可能性があることを意味します。

インスタンスでデータベースがマウントされているがオープンされていない場合にのみ、NOSWITCH句を使用できます。データベースがオープンしている場合は、この操作によってデータベースは自動的にクローズされます。再オープンする前にデータベースを手動で停止する必要があります。

GROUP句

GROUPを指定すると、GROUPの値がintegerで指定した値に等しいオンラインREDOログ・ファイル・グループを手動でアーカ

イブできます。REDOログ・ファイル・グループのGROUP値を判別するには、動的パフォーマンス・ビューV$LOGを問い合せます。THREADパラメータとGROUPパラメータの両方を指定する場合は、指定したREDOログ・ファイル・グループが指定したスレッド内に含まれている必要があります。

LOGFILE句

LOGFILEを指定すると、filenameで指定した名前のREDOログ・ファイル・メンバーが含まれるオンラインREDOログ・ファイル・グループを手動でアーカイブできます。THREADパラメータとLOGFILEパラメータの両方を指定する場合は、指定したREDOログ・ファイル・グループが指定したスレッド内に含まれている必要があります。

データベースがバックアップ制御ファイルでマウントされている場合は、USING BACKUP CONTROLFILEを指定し、現行のログ・ファイルを含むすべてのオンライン・ログ・ファイルのアーカイブを許可します。

LOGFILE句の制限事項

REDOログ・ファイル・グループは、入力された順序でアーカイブする必要があります。LOGFILEパラメータを使用してREDOログ・ファイル・グループのアーカイブを指定した場合、それ以前のREDOログ・ファイル・グループがアーカイブされていないとエラー・メッセージが戻ります。

NEXT句

NEXTを指定すると、一杯になってもアーカイブされていない次のオンラインREDOログ・ファイル・グループを、指定したスレッドから手動でアーカイブできます。THREADパラメータを指定しない場合、使用可能な任意のスレッド上の、アーカイブされていない最初のREDOログ・ファイル・グループがアーカイブされます。

ALL句

ALLを指定すると、一杯になってもアーカイブされていないすべてのオンラインREDOログ・ファイル・グループを、指定したスレッドから手動でアーカイブできます。THREADパラメータを指定しない場合、使用可能なすべてのスレッドから、一杯でアーカイブされていないすべてのREDOログ・ファイル・グループがアーカイブされます。

TO location 句

TO 'location'を指定すると、REDOログ・ファイル・グループがアーカイブされる位置を指定できます。このパラメータの値には、オペレーティング・システムの規則に従って、ファイルの位置を完全に指定する必要があります。このパラメータを指定しない場合、REDOログ・ファイル・グループは初期化パラメータLOG_ARCHIVE_DESTまたはLOG_ARCHIVE_DEST_nに指定された場所に格納されます。

checkpoint_clause

CHECKPOINTを指定すると、チェックポイントを明示的に強制処理して、コミット済のトランザクションによる変更をディスク上のデータファイルに書き込むことができます。インスタンスでデータベースがオープンしている場合にのみ、この句を指定できます。チェックポイントが完了するまで、ユーザーに制御は戻りません。

GLOBAL

Oracle Real Application Clusters (Oracle RAC)環境で、データベースをオープンしているすべてのインスタンスに対してチェックポイントを実行します。これはデフォルトです。

LOCAL

Oracle RAC環境で、文を発行するインスタンスのREDOログ・ファイル・グループのスレッドに対してのみチェックポイントを実行します。

関連項目:

チェックポイントの強制実行: 例

check_datafiles_clause

Oracle RAC環境などの分散データベース・システムで、データベース制御ファイルからインスタンスのSGAを更新し、すべてのオンライン・データファイルに情報を反映します。

- GLOBALを指定すると、データベースをオープンしているすべてのインスタンスに対して、この同期化を実行できます。これはデフォルトです。
- LOCALを指定すると、ローカル・インスタンスに対してのみこの同期化を実行できます。

インスタンスでデータベースをオープンしておく必要があります。

distributed_recov_clauses

DISTRIBUTED RECOVERY句を使用すると、分散リカバリを使用可能または使用禁止にできます。この句を使用する場合、インスタンスでデータベースをオープンする必要があります。

ENABLE

ENABLEを指定すると、分散リカバリを有効にできます。シングルプロセス環境では、分散リカバリを開始する場合にこの句を使用する必要があります。

トランザクションに関係があるリモート・ノードにアクセスできない場合、インダウト・トランザクションをリカバリするには、ENABLE DISTRIBUTED RECOVERY文を複数回発行する必要がある場合もあります。インダウト・トランザクションは、データ・ディクショナリ・ビューDBA_2PC_PENDINGに表示されます。

関連項目:

分散リカバリの有効化: 例

DISABLE

DISABLEを指定すると、分散リカバリを無効にできます。

FLUSH SHARED_POOL句

FLUSH SHARED_POOL句を指定すると、システム・グローバル領域(SGA)の共有プール上のデータが消去されます。共有プールは次のものを格納します。

- キャッシュされたデータ・ディクショナリ情報
- SQL文の共有SQL領域、共有PL/SQL領域、ストアド・プロシージャ、ファンクション、パッケージおよびトリガー

この文は、グローバル・アプリケーション・コンテキスト情報を消去せず、また現在実行中のアイテムで共有SQL領域およびPL/SQL領域を消去しません。インスタンスでデータベースがマウントされていてもディスマウントされていても、またはオープン状態でもクローズ状態でも、この句を使用できます。

関連項目:

共有プールのクリア: 例

FLUSH GLOBAL CONTEXT句

FLUSH GLOBAL CONTEXT句を指定すると、すべてのグローバル・アプリケーション・コンテキスト情報がシステム・グローバル領域(SGA)の共有プール上からフラッシュされます。インスタンスでデータベースがマウントされていてもディスマウントされていても、またはオープン状態でもクローズ状態でも、この句を使用できます。

FLUSH BUFFER_CACHE句

FLUSH BUFFER_CACHE句を指定すると、KEEP、RECYCLEおよびDEFAULTバッファ・プールを含め、システム・グローバル領域(SGA)のバッファ・キャッシュ上のすべてのデータを消去できます。

ノート:

この句は、テスト・データベース上のみで使用することを目的としています。この句を本番データベース上で使用しないでください(この文を実行すると、それ以降の問合せでは結果が何も戻されなくなるため)。

この句は、リライトされた問合せ、または同一の開始点からの一連の問合せのパフォーマンスを測定する必要がある場合に有効です。

FLUSH FLASH_CACHE句

FLUSH FLASH_CACHE句を使用すると、データベース・スマート・フラッシュ・キャッシュをフラッシュできます。この句は、リライトされた問合せまたは同一の開始点からの一連の問合せのパフォーマンスを測定する必要がある場合、またはキャッシュに破損がある可能性がある場合に有効です。

FLUSH REDO句

FLUSH REDO句を指定すると、プライマリ・データベースからスタンバイ・データベースにREDOデータをフラッシュできます。また、オプションで、フラッシュされたREDOデータがフィジカル・スタンバイ・データベースまたはロジカル・スタンバイ・データベースに適用されるのを待機することもできます。

この句により、プライマリ・データベースが生成したすべてのREDOデータをスタンバイ・データベースにフラッシュできれば、プライマリ・データベースがデータ非消失データ保護モードでない場合でも、データを失うことなくターゲット・スタンバイ・データベースにフェイルオーバーを実行できます。

FLUSH REDO句は、マウントされている(ただし、オープンされていない)プライマリ・データベースで発行する必要があります。

target_db_name

target_db_nameには、プライマリ・データベースからフラッシュされるREDOデータを受け取るスタンバイ・データベースのDB_UNIQUE_NAMEを指定します。

ターゲット・スタンバイ・データベースに対応するLOG_ARCHIVE_DEST_nデータベース初期化パラメータの値は、DB_UNIQUE_NAME属性を含む必要があります。また、その属性の値は、ターゲット・スタンバイ・データベースのDB_UNIQUE_NAMEに一致する必要があります。

NO CONFIRM APPLY

この句を指定すると、スタンバイ・データベースがフラッシュされたすべてのREDOデータを受け取るまで、ALTER SYSTEM文は完了しません。ターゲット・スタンバイ・データベースがスナップショット・スタンバイ・データベースである場合は、この句を指定する必要があります。

CONFIRM APPLY

この句を指定すると、ターゲット・スタンバイ・データベースがフラッシュされたすべてのREDOデータを受け取って適用するまで、

ALTER SYSTEM文は完了しません。これは、NO CONFIRM APPLYを指定しない場合のデフォルトの動作です。ターゲット・スタンバイ・データベースがスナップショット・スタンバイ・データベースである場合は、この句を指定できません。

関連項目:

FLUSH REDO句およびフェイルオーバーの詳細は、『Oracle Data Guard概要および管理』を参照してください。

end_session_clauses

end_session_clausesを使用すると、現行のセッションを終了することができます。

DISCONNECT SESSION句

DISCONNECT SESSION句を使用すると、専用サーバー・プロセス(共有サーバーによって接続が確立されていた場合は、仮想回路)を破棄することによって、現行のセッションが切断されます。この句を使用する場合、インスタンスでデータベースをオープンする必要があります。次の両方の値をV$SESSIONビューで確認して、このセッションを識別する必要があります。

- integer1には、SID列の値を指定します。
- integer2には、SERIAL#列の値を指定します。

システム・パラメータを適切に設定した場合、アプリケーション・フェイルオーバーが有効になります。

- POST_TRANSACTIONを設定すると、セッションが切断される前に、実行中のトランザクションを完了できます。セッションに実行中のトランザクションがない場合、この句は、後述のKILL SESSIONと同様の効果があります。
- IMMEDIATEを設定すると、実行中のトランザクションの完了を待たずにセッションを切断し、すぐにセッション全体の状態をリカバリできます。
    - POST_TRANSACTIONを指定した場合、セッションに実行中のトランザクションがあれば、IMMEDIATEキーワードは無視されます。
    - POST_TRANSACTIONを指定しない場合、またはPOST_TRANSACTIONを指定していてもセッションに実行中のトランザクションがない場合、この句は、後述のKILL SESSION IMMEDIATEと同様の効果があります。

関連項目:

セッションの切断: 例

KILL SESSION句

KILL SESSION句を指定すると、セッションの状態は終了済となり、実行中のトランザクションがロールバックされ、すべてのセッション・ロックが解放され、セッション・リソースの部分的リカバリが行われます。この句を使用する場合、インスタンスでデータベースをオープンする必要があります。integer3を指定しない場合は、自セッションおよび終了されるセッションは、同じインスタンスにある必要があります。次の値をV$SESSIONビューで確認して、このセッションを識別する必要があります。

- integer1には、SID列の値を指定します。
- integer2には、SERIAL#列の値を指定します。
- オプションのinteger3には、終了されるターゲット・セッションが存在するインスタンスのIDを指定します。GV$表を問い合せることによって、インスタンスIDを確認することができます。

完了する必要があるアクティビティ(リモート・データベースからの応答の待機やトランザクションのロールバックなど)がセッションで実行されている場合、Oracle Databaseはそのアクティビティが完了するのを待機し、セッションに終了のマークを付けてから、ユーザーに制御を戻します。待機が数分間続く場合、セッションに終了予定のマークが付けられ、セッションに終了予定のマークを付けたというメッセージとともにユーザーに制御が戻されます。アクティビティが完了すると、PMONバックグラウンド・プロセスは、セッションに終了済のマークを付けます。

セッションに実行中のトランザクションがあるかどうかにかかわらず、セッション・ユーザーがセッションに要求を発行してセッションが終了されたことを示すメッセージを受け取るまで、Oracle Databaseは、セッション全体の状態をリカバリしません。

関連項目:

セッションの終了: 例

IMMEDIATE

IMMEDIATEを指定すると、実行中のトランザクションをロールバックしてすべてのセッション・ロックを解放し、セッション全体の状態をリカバリしてから、すぐにユーザーに制御を戻すようにOracle Databaseに指示できます。

NOREPLAY

この句は、アプリケーション・コンティニュイティを使用している場合に有効です。アプリケーション・コンティニュイティが有効なサービスに接続しているとき(つまり、FAILOVER_TYPE = TRANSACTION)、セッションが失敗したり強制終了されると、セッションがリカバリされます。終了したセッションをリカバリする必要がない場合は、NOREPLAYを指定します。

SWITCH LOGFILE句

SWITCH LOGFILE句を指定すると、現行のREDOログ・ファイル・グループのファイルが一杯であるかどうかにかかわらず、新しいREDOログ・ファイル・グループへの書込みを明示的かつ強制的に開始できます。ログ・スイッチを強制的に発生させると、チェックポイントが実行されますが、チェックポイントが完了する前に、すぐに制御が戻されます。この句を使用する場合、インスタンスでデータベースをオープンする必要があります。

関連項目:

ログ・スイッチの強制実行: 例

SUSPEND | RESUME

SUSPEND句を指定すると、すべてのI/O(データファイル、制御ファイルおよびファイル・ヘッダー)および問合せを停止し、すべてのインスタンスで実行中のトランザクションを処理せずにデータベースのコピーが作成可能になります。

SUSPENDおよびRESUMEの制限事項

SUSPENDおよびRESUMEには、次の制限事項があります。

- ホット・バックアップ・モードでデータベースの表領域を確保するまで、この句は使用できません。
- ALTER SYSTEM SUSPEND文を発行したセッションは終了しないでください。システムが一時停止しているときに再接続しようとすると、SYSログイン中に実行される再帰的SQLが原因となって接続が失敗することがあります。
- システムが停止中に新しいインスタンスを起動する場合、この新しいインスタンスは停止しません。

RESUME句を指定すると、問合せおよびI/Oに対して、再度、データベースが使用可能になります。

quiesce_clauses

QUIESCE RESTRICTED句およびUNQUIESCE句を使用すると、データベースを静止状態にしたり、静止状態から戻すことができます。この状態では、データベース管理者は、トランザクション、問合せまたはPL/SQL操作が同時に存在する状態では安全に実行できない管理操作を実行することができます。

ノート:

QUIESCE RESTRICTED 句は、データベース・リソース・マネージャがインストールされている場合、およびデータベースをオープンしたインスタンスでデータベースが起動された後、リソース・マネージャが継続的にアクティブになっている場合のみ有効です。

複数のQUIESCE RESTRICTED文またはUNQUIESCE文が異なるセッションまたはインスタンスで同時に発行された場合、1つを除いた他のすべてのセッションまたはインスタンスにエラーが戻されます。

QUIESCE RESTRICTED

QUIESCE RESTRICTEDを指定すると、データベースを静止状態にできます。この句は、データベースがオープン中のすべてのインスタンスに次の影響を与えます。

- Oracle Databaseは、すべてのインスタンスのデータベース・リソース・マネージャに、アクティブでないすべてのセッション(SYSおよびSYSTEM以外)をアクティブにしないように指示します。SYSおよびSYSTEM以外のユーザーは、新しいトランザクション、問合せ、フェッチまたはPL/SQL操作を開始できません。
- Oracle Databaseは、すべてのインスタンスの既存のトランザクションのうち、起動したユーザーがSYSでもSYSTEMでもないものがすべて終了(コミットまたは異常終了)するまで待機します。また、Oracle Databaseは、すべてのインスタンスの実行中の問合せ、フェッチおよびPL/SQLプロシージャのうち、開始したユーザーがSYSでもSYSTEMでもなく、トランザクション内にもないものがすべて完了するまで待機します。連続する複数のOCIのフェッチによって問合せが実行される場合、Oracle Databaseはすべてのフェッチが終了するまで待機しません。現行のフェッチが終了するまで待機し、次のフェッチの実行をブロックします。さらに、Oracle Databaseは、エンキューなどの共有リソースを保持するセッション(SYSおよびSYSTEMのものを除く)すべてがそのリソースを解放するまで待機します。これらの操作がすべて完了すると、データベースが静止状態になり、QUIESCE RESTRICTED文の実行が終了します。
- 共有サーバー・モードでインスタンスを実行している場合、Oracle Databaseは、SYSまたはSYSTEM以外のユーザーがそのインスタンスにログインすることを阻止するようにデータベース・リソース・マネージャに指示します。非共有サーバー・モードでインスタンスを実行している場合、そのインスタンスへのユーザー・ログインに制限はありません。

静止状態中、すべてのインスタンスにおいてリソース・マネージャのプランは変更できません。

UNQUIESCE

UNQUIESCEを指定すると、データベースを静止状態から戻すことができます。これによって、SYSまたはSYSTEM以外のユーザーによって開始された、トランザクション、問合せ、フェッチおよびPL/SQLプロシージャを再開できます。UNQUIESCE文は、QUIESCE RESTRICTED文を発行したセッションと同じセッションで起動する必要はありません。

rolling_migration_clauses

クラスタ化されたOracle Automatic Storage Management (Oracle ASM)環境内でこれらの句を使用すると、Oracle ASMクラスタまたはストレージにOracle ASMを使用するデータベース・クラスタの全体的な可用性に影響することなく、ノードを一度に1つずつ別のOracle ASMバージョンに移行できます。

START ROLLING MIGRATION

ローリング・アップグレードを開始するときに、ASM_versionについて、次の文字列を指定する必要があります。

```
'<version_num>, <release_num>, <update_num>,<port_release_num>,<port_update_num>'
```

ASM_versionは、11.1.0.0.0以上である必要があります。一重引用符で囲む必要があります。次に、Oracle ASMは、最初に、指定されたリリースへの移行について現在のリリースに互換性があることを確認し、制限された機能モードになります。Oracle ASMは、次に、クラスタ内で均衡の再調整処理が進行中かどうかを確認します。そのような操作がある場合、文は失敗し、均衡の再調整操作の完了後に文を再発行する必要があります。

ローリング・アップグレード・モードは、クラスタ全体でインメモリーに永続的に保持される状態です。クラスタは、少なくとも1つのOracle ASMインスタンスがクラスタ内で実行中になるまで、この状態であり続けます。クラスタに参加する新しいインスタンスは、起動時にすぐに移行モードに切り替わります。クラスタ内のすべてのインスタンスが終了すると、この後にOracle ASMインスタンスを起動しても、この文を再発行してOracle ASMインスタンスのローリング・アップグレードを再開するまでは、ローリング・アップグレード・モードにはなりません。

STOP ROLLING MIGRATION

この句を使用すると、ローリング・アップグレードを停止して、クラスタを通常の操作に戻すことができます。クラスタ内のすべてのインスタンスが同じソフトウェア・バージョンに移行した後にのみ、この句を指定します。クラスタがローリング・アップグレード・モードではない場合、文は失敗します。

この句を指定すると、クラスタのすべてのメンバーが同じソフトウェア・バージョンであることの検証が行われ、Oracle ASMインスタンスのローリング・アップグレード・モードが解除され、Oracle ASMクラスタの機能が完全に元に戻ります。ディスクがオフラインであるために均衡の再調整操作が保留中の場合、操作は再起動されます。ただし、そのような再起動によってASM_POWER_LIMITパラメータに違反することにならない場合にかぎります。

関連項目:

ローリング・アップグレードの詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

rolling_patch_clauses

クラスタ化されたOracle Automatic Storage Management (Oracle ASM)環境内でこれらの句を使用すると、Oracle ASMクラスタまたはストレージにOracle ASMを使用するデータベース・クラスタの全体的な可用性に影響することなく、ノードを一度に最新のパッチ・レベルに更新できます。

START ROLLING PATCH

この句を使用すると、ローリング・パッチ操作を開始できます。Oracle ASMはクラスタ内のすべてのライブ・ノードが同じバージョンであることを検証してから、ローリング・パッチ・モードに移行します。このモードは、クラスタ全体でインメモリーに永続的に保持される状態です。クラスタは、すべてのライブ・ノードにパッチが適用されて最新のパッチ・レベルになるまで、この状態を保持します。

この操作中に停止していたノードにはパッチが適用されません。これは、ローリング・パッチ操作の成否には影響しません。ただし、これらのノードには開始される前にパッチを適用する必要があります。適用しないと、これらのノードがクラスタに参加できなくなります。

STOP ROLLING PATCH

この句を使用すると、ローリング・パッチ操作を停止して、クラスタを通常の操作に戻すことができます。この句は、クラスタ内のすべてのライブ・ノードに最新のパッチ・レベルまでパッチが適用されてから指定してください。クラスタがローリング・パッチ・モードではない場合は、文が失敗します。

この句を指定すると、Oracle ASMインスタンスのすべてのメンバーが同じパッチ・レベルであることが検証され、インスタンスのローリング・パッチ・モードが解除され、Oracle ASMクラスタのフル機能に戻ります。クラスタのいずれかのメンバーが同じパッチ・レベルでない場合、この操作は失敗し、クラスタは限定機能モードに移行します。

次に示す問合せは、ローリング・パッチの情報を表示します。これらの問合せを実行するには、Gridホーム内のOracle ASMインスタンスに接続している必要があります。また、Grid InfrastructureホームがOracle RAC環境用のOracle Clusterwareオプションを使用して構成されている必要があります。

- クラスタがローリング・パッチ・モードであるかどうかを確認するには、次の問合せを使用します。

```
SELECT SYS_CONTEXT('SYS_CLUSTER_PROPERTIES', 'CLUSTER_STATE') FROM DUAL;
```

- クラスタのパッチ・レベルを確認するには、次の問合せを使用します。

```
SELECT SYS_CONTEXT('SYS_CLUSTER_PROPERTIES', 'CURRENT_PATCHLVL') FROM DUAL;
```

- Oracle ASMインスタンスに適用されているパッチのリストを表示するには、V$PATCHES動的パフォーマンス・ビューを問い合せます。詳細は、『Oracle Databaseリファレンス』を参照してください。

関連項目:

ローリング・パッチの詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

security_clauses

`security_clauses`を使用すると、インスタンスへのアクセスを制御できます。また、インスタンスで暗号化されたデータへのアクセスを許可または禁止できます。

RESTRICTED SESSION

`RESTRICTED SESSION`句を指定すると、Oracle Databaseにログインできるユーザーを制限できます。インスタンスでデータベースがマウントされていてもディスマウントされていても、またはオープン状態でもクローズ状態でも、この句を使用できます。

- `ENABLE`を指定すると、`RESTRICTED SESSION`システム権限が付与されているユーザーのみがOracle Databaseにログインできます。既存のセッションは終了しません。

  この句は、現行のインスタンスのみに適用されます。そのため、Oracle RAC環境では、`RESTRICTED SESSION`システム権限を持たない認可済ユーザーも他のインスタンスでデータベースにアクセスできます。

- `DISABLE`を指定すると、`ENABLE RESTRICTED SESSION`句の効果が逆になり、`CREATE SESSION`システム権限が付与されているすべてのユーザーがOracle Databaseにログオンできるようになります。これはデフォルトです。

関連項目:

「セッションの制限: 例」

SET ENCRYPTION WALLET句

この句を使用すると、データの透過的暗号化(TDE)のマスター暗号化キーへのデータベース・アクセスを管理できます。TDEのマスター暗号化キーは、外部セキュリティ・モジュール(暗号化ウォレットまたはハードウェア・セキュリティ・モジュール(HSM))に格納されています。

この文は、キーワードALTERで始まりますが、`ALTER SYSTEM SET ENCRYPTION WALLET`文はDDL句ではありません。ただし、このような文はロールバックすることはできません。

この句はSETキーワードで始まりますが、初期化パラメータの値を設定するためにSETキーワードの使用を許可するalter_system_set_clauseと混同しないでください。ENCRYPTION WALLETは、初期化パラメータではありません。

OPEN

この句を指定すると、データベースは指定したパスワードを使用してウォレットを開き、TDEマスター・キーをデータベース・メモリーにロードするか(このキーはインスタンスが存続している間有効)、HSMとの接続を確立するようになります(暗号化された表/表領域キーをHSMに送信して、復号化されたものを受け取るため)。

- wallet_passwordを指定すると、暗号化ウォレットからマスター暗号化キーを取り出すことができます。ウォレットが使用できないか、すでに開いている場合は、データベースからエラーが戻ります。wallet_passwordは、二重引用符で囲む必要があります。
- HSM_auth_stringを指定すると、HSMがアクセス可能になります。HSM_auth_stringの形式は"user_id:password"で、それぞれの意味は次のとおりです。
    - user_idは、HSM管理インタフェースを使用してデータベース用に作成されたユーザーIDです。
    - passwordは、HSM管理インタフェースを使用してユーザーIDに対して作成されたパスワードです。

  HSM_auth_stringを囲む二重引用符は必須です

CLOSE

この句を使用すると、データベースの暗号化と復号化を無効にできます。暗号化ウォレットを閉じるには、wallet_passwordが必要です。HSMへのアクセスを禁止するには、HSM_auth_stringが必要です。HSM_auth_stringを指定する方法の詳細は、「OPEN」 を参照してください。

自動オープン・ウォレットのみが存在する場合、自動オープン・ウォレットを閉じるためにパスワードは必要ありません。自動オープン・ウォレットと暗号化ウォレットの両方が開いている場合、自動オープン・ウォレットを閉じるためにパスワードが必要です。この場合、CLOSEでパスワードを指定すると、自動オープン・ウォレットと暗号化ウォレットが閉じます。

関連項目:

Oracle Real Application Clusters(Oracle RAC)環境での暗号化ウォレットの設定については、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください。

set_encryption_key

この句を使用すると、TDEマスター暗号化キーが存在しない場合に新規作成できます。既存のマスター・キーがHSMまたはキーストアにある場合は、この句を指定すると既存の表および表領域の新しいキーが生成され、したがってすべての表および表領域のキーが古いマスター・キーで復号化されて新しいマスター・キーで再度暗号化されます。

ALTER SYSTEM SET ENCRYPTION KEY文はDDL文です。この文は、スキーマ内で保留中のトランザクションを自動的にコミットします。

この句はSETキーワードで始まりますが、初期化パラメータの値を設定するためにSETキーワードの使用を許可するalter_system_set_clauseと混同しないでください。ENCRYPTION KEYは、初期化パラメータではありません。

IDENTIFIED BY wallet_password

この句を指定すると、暗号化データにアクセスするためのTDEマスター暗号化キーが暗号化ウォレットからメモリーにロードされます。

- certificate_idが必要になるのは、PKI非対称型キー・ペアをマスター暗号化キーとして使用する場合です。証明書を表す整数を指定します。この値を見つけるには、V$WALLET動的パフォーマンス・ビューのCERT_ID列を問い合せます。デフォルトである対称型キーを使用する場合は、certificate_idを指定しないでください。
- wallet_passwordには、セキュリティ・モジュールへの接続に使用するパスワードを指定します。

指定したcertificate_idまたはwallet_passwordが無効の場合は、データベース・エラーが戻されます。certificate_idおよびwallet_passwordを囲む二重引用符は必須です。

IDENTIFIED BY wallet_passwordの制限事項

統合マスター暗号化キーを含むPKIベースのマスター・キーは、TDE列の暗号化およびOracleウォレットでのみ使用でき、HSMでは使用できません。

> ノート:
>
> PKI 暗号化と透過的データ暗号化を組み合せた使用は非推奨です。透過的データ暗号化を構成するには、ADMINISTER KEY MANAGEMENT 文を使用します。詳細は、Oracle Database Advanced Security ガイドを参照してください。

IDENTIFIED BY HSM_auth_string

この句は、HSM内部に格納されるマスター暗号化キーを作成します。マスター暗号化キーは、HSM内の表キーを暗号化または復号化するために使用されます。

HSM_auth_stringの形式は"user_id:password"で、それぞれの意味は次のとおりです。

- user_idは、HSM管理インタフェースを使用してデータベース用に作成されたユーザーIDです。
- passwordは、HSM管理インタフェースを使用してユーザーIDに対して作成されたパスワードです。

HSM_auth_stringを囲む二重引用符は必須です。

すでに透過的データ暗号化をOracleウォレットとともに使用している場合に、HSMに移行するには、MIGRATE USING wallet_password句を指定します。これによって、既存の表/表領域キーがいったん復号化され、新規作成されたHSMベースのマスター暗号化キーを使用して再び暗号化されます。ウォレットはHSMへの移行後も使用されることに注意してください(このウォレットには、エクスポート・ファイルやRMANバックアップに、あるいは一時またはUNDO表領域やREDOログ・ファイル内の暗号化データに使用されたマスター暗号化キーが格納されている可能性があるため)。移行後に、次のいずれかのステップを実行します。

- ウォレット・パスワードをHSM_auth_stringに変更します(変更するにはOracle Wallet Managerまたはorapkiコマンドライン・ツールを使用)。
- ローカルの自動オープン・ウォレットを暗号化ウォレットから作成し、その後で、暗号化ウォレットの名前を変更するか、sqlnet.oraのENCRYPTION_WALLET_LOCATIONで指定されたディレクトリの外に移動します。暗号化ウォレットを削除したり、ウォレット・パスワードを忘れないよう注意してください。

ENCRYPTION_WALLET_LOCATIONパラメータは非推奨になりつつあることに注意してください。今後は、静的な初期化パラメータWALLET_ROOTと、動的な初期化パラメータTDE_CONFIGURATIONを使用することをお薦めします。

関連項目:

- 透過的データ暗号化の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。
- この機能を利用して表の列を暗号化する方法の詳細は、「CREATE TABLE」の「encryption_spec」を参照してください。
- ウォレットおよび暗号化キーの確立: 例

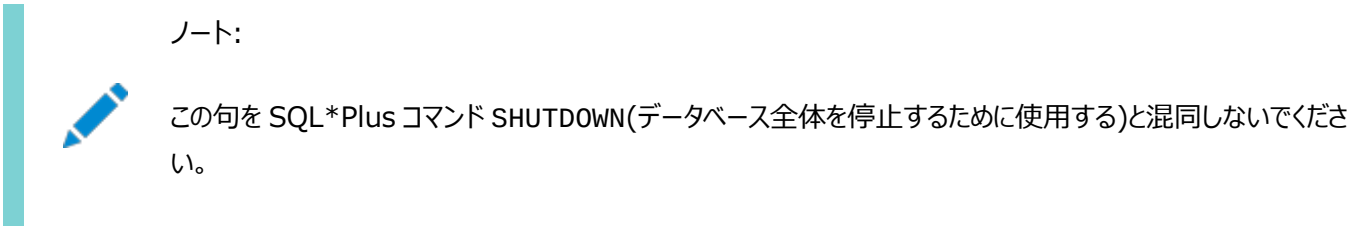
affinity_clauses

affinity_clausesを使用して、データ依存型ルーティングでRACデータベースに対するキャッシュ・アフィニティを提供できるようにします。アフィニティにより、データの個別のサブセットが各インスタンスに割り当てられるように、データが複数のRACインスタンスに論理的にパーティション化されます。シャーディング・キーを使用してデータがアクセスされると、要求は、データの対応するサブセットを保持しているインスタンスにルーティングされます。アフィニティには次の利点があります。

- シャード認識アプリケーションのシャード・アクセス、および非シャード・アプリケーションの透過性
- キャッシュの使用の効率化およびブロックのpingの減少

shutdown_dispatcher_clause

SHUTDOWN句が意味を持つのは、Oracle Databaseの共有サーバー・アーキテクチャを使用しているシステムのみです。dispatcher_nameで指定されたディスパッチャが停止します。



ノート:

この句を SQL*Plus コマンド SHUTDOWN(データベース全体を停止するために使用する)と混同しないでください。

dispatcher_nameは'Dxxx'の形式の文字列で、xxxはディスパッチャの番号です。ディスパッチャ名のリストを取得するには、V$DISPATCHER動的パフォーマンス・ビューのNAME列を問い合せます。

- IMMEDIATEを指定した場合、ディスパッチャは新しい接続の受入れをすぐに中止し、そのディスパッチャによる既存の接続はすべて終了されます。すべてのセッションがクリーン・アップされてから、ディスパッチャ・プロセスが停止します。
- IMMEDIATEを指定しない場合、ディスパッチャは新しい接続の受入れをすぐに中止しますが、すべてのユーザーが切断し、すべてのデータベース・リンクが終了されるのを待ちます。その後、ディスパッチャは停止されます。

REGISTER句

REGISTERを指定すると、PMONバックグラウンド・プロセスによってリスナーにインスタンスがすぐに登録されます。この句を指定しない場合、PMONが次に検出ルーチンを実行するまでインスタンスの登録は行われません。その結果、クライアントは、リスナー起動後最大60秒間サービスにアクセスできない可能性があります。

関連項目:

PMONバックグラウンド・プロセスおよびリスナーの詳細は、『Oracle Database概要』および『Oracle Database Net Services管理者ガイド』を参照してください。

alter_system_set_clause

この句を使用すると、パラメータ値を変更できます。set_parameter_clauseを使用すると、指定した初期化パラメータの値を変更できます。USE_STORED_OUTLINESおよびGLOBAL_TOPIC_ENABLED句を使用すると、これらのシステム・パラ

メータの値を変更できます。

set_parameter_clause

従来のプレーン・テキストのパラメータ・ファイル(PFILE)を使用してデータベースを起動したか、サーバー・パラメータ・ファイル(SPFILE)を使用してデータベースを起動したかに応じて、現行のインスタンスの多くの初期化パラメータ値を変更できます。『Oracle Databaseリファレンス』では、各パラメータの説明で、これらのパラメータが「変更可能」というカテゴリに分類されています。PFILEを使用した場合、変更はインスタンスの存続期間中のみ保持されます。一方、SPFILEを使用してデータベースを起動した場合、SPFILE自体のパラメータの値を変更できるため、後続のインスタンスで新しい値が使用されます。

『Oracle Databaseリファレンス』には、すべての初期化パラメータが詳細に記載されています。パラメータは、次の3つのカテゴリに分類されます。

- 基本パラメータ: データベース管理者は、すべての基本パラメータについて熟知し、これらのパラメータの設定を考慮する必要があります。
- 機能のカテゴリ: 『Oracle Databaseリファレンス』には、初期化パラメータが機能のカテゴリ別にも示されています。
- アルファベット順: 『Oracle Databaseリファレンス』の目次には、すべての初期化パラメータがアルファベット順に示されています。

初期化パラメータ値を変更する権限は、従来のプレーン・テキストの初期化パラメータ・ファイル(pfile)を使用してデータベースを起動したか、サーバー・パラメータ・ファイル(spfile)を使用してデータベースを起動したかによって異なります。特定のパラメータ値を変更する権限を持っているかどうかを確認するには、V$PARAMETER動的パフォーマンス・ビューのISSYS_MODIFIABLE列を問い合せます。

パラメータ値を設定するときに、次の設定も行えます。

COMMENT

COMMENT句を使用すると、コメント文字列をパラメータ値の変更に関連付けることができます。コメント文字列には、制御文字または改行を含めることはできません。SPFILEをあわせて指定すると、パラメータ・ファイルにコメントが表示され、そのパラメータに対する直前の変更がわかります。

DEFERRED

DEFERREDキーワードを指定すると、データベースに接続するその後のセッションに対するパラメータの値を設定または変更できます。現行のセッションでは変更前の値が残ります。

このパラメータのV$PARAMETERのISSYS_MODIFIABLE列の値がDEFERREDの場合は、DEFERREDを指定する必要があります。この列の値がIMMEDIATEの場合、この句のDEFERREDキーワードはオプションです。この列の値がFALSEの場合、このALTER SYSTEM文ではDEFERREDを指定できません。

関連項目:

V$PARAMETER動的パフォーマンス・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

CONTAINER

CDBでパラメータ値を設定するときには、CONTAINER句を指定できます。CDBでは初期化パラメータの継承モデルが使用されます。このモデルでは、PDBがルートから初期化パラメータの値を継承します。この場合、継承は、ルートの特定のパラメータ値が個別PDBに適用されることを意味します。

一部のパラメータについては、各PDBでルートの設定を上書きできます。これは、各PDBが初期化パラメータごとにtrueまたは

falseの継承プロパティを持つことを意味します。あるパラメータでPDBがルートの値を継承する場合、そのパラメータの継承プロパティはtrueです。PDBがルートの値を継承しない場合、そのパラメータの継承プロパティはfalseです。

一部のパラメータについては、継承プロパティがtrueである必要があります。その他のパラメータでは、継承プロパティを変更できます。この変更を行うには、現在のコンテナが操作対象のPDBであるときに、ALTER SYSTEM SET文を使用してパラメータを設定します。V$SYSTEM_PARAMETERビューで初期化パラメータのISPDB_MODIFIABLE値がTRUEになっている場合、そのパラメータの継承プロパティをfalseに設定できます。

- CONTAINER = ALLを指定すると、CDB内のすべてのコンテナ(ルートとすべてのPDBを含む)にそのパラメータ設定が適用されます。現在のコンテナがルートである必要があります。

  ALLを指定すると、すべてのPDB内のパラメータの継承プロパティはtrueに設定されます。

- CONTAINER = CURRENTを指定すると、現在のコンテナのみにパラメータ設定が適用されます。現在のコンテナがルートである場合は、ルートと、そのパラメータについてtrueの継承プロパティを持つすべてのPDBにそのパラメータ設定が適用されます。

この句を省略すると、CONTAINER = CURRENTがデフォルトになります。

関連項目:

CDBのパラメータの変更の詳細は、『Oracle Database管理者ガイド』を参照してください。

SCOPE

SCOPE句を使用すると、変更が有効になるタイミングを指定できます。この句の動作は、非CDB、CDBルートまたはPDBのいずれに接続されているかによって異なります。

非CDBまたはCDBルートに接続しているときにALTER SYSTEM文を発行した場合、有効範囲は、データベースの起動に使用したファイルが従来のプレーン・テキストのパラメータ・ファイル(pfile)か、サーバー・パラメータ・ファイル(spfile)かによって異なります。

- MEMORYを指定すると、変更がメモリーで行われ、すぐに有効になり、データベースが停止するまで持続されます。パラメータ・ファイル(pfile)を使用してデータベースを起動した場合、この有効範囲のみを指定できます。

  MEMORYは、すべてのインスタンスのメモリーを変更し、インスタンスで個別に設定された値を上書きすることに注意してください。

- SPFILEを指定すると、変更がサーバー・パラメータ・ファイルで行われます。新しい設定は、データベースが次に停止し、再起動されたときに有効になります。『Oracle Databaseリファレンス』に変更不可能と示されている静的パラメータ値を変更する場合は、SPFILEを指定する必要があります。

  SPFILEはメモリーを変更しないことに注意してください。つまり、インスタンスで個別に設定されたインスタンス・パラメータはグローバルよりも優先されます。

- BOTHを指定すると、変更がメモリーとサーバー・パラメータ・ファイルの両方で行われます。新しい設定はすぐに有効になり、データベースが停止し、再起動された後も持続します。

  BOTHは、インスタンスを再起動するまで、すべてのインスタンスのメモリーを変更し、インスタンスで個別に設定された値を上書きすることに注意してください。インスタンスを再起動すると、spfileが読み取られ、その後インスタンス・パラメータが優先されます。

データベースの起動でサーバー・パラメータ・ファイルを使用した場合は、BOTHがデフォルトです。データベースの起動でパラメー

タ・ファイルを使用した場合は、MEMORYがデフォルトで、これ以外の有効範囲は指定できません。

PDBに接続しているときにALTER SYSTEM文を発行した場合は、V$SYSTEM_PARAMETERビューのISPDB_MODIFIABLE列がTRUEである初期化パラメータのみを変更できます。初期化パラメータ値はPDBでのみ有効になります。初期化パラメータがPDB用に明示的に設定されていない場合、PDBではCDBルートのパラメータ値を継承します。

- MEMORYを指定すると、変更がメモリーで行われ、PDBですぐに有効になります。この設定によって、次のいずれかの場合、CDBルートで設定された値に戻ります。
    - ALTER SYSTEM SET文によって、SCOPEがBOTHまたはMEMORYに等しい場合はルートのパラメータの値が設定され、PDBがクローズして再オープンされます。SCOPEがSPFILEと等しい場合、PDBをクローズして再オープンしてもPDBのパラメータ値は変更されません。
    - PDBがクローズして、再オープンされます。
    - CDBが停止し、再オープンされます。
- SPFILEを指定すると、変更がPDBに対して行われ、永続的に格納されます。新しい設定はPDBのみに影響し、次のいずれかの場合に有効になります。
    - PDBがクローズして、再オープンされます。
    - CDBが停止し、再オープンされます。
- BOTHを指定すると、変更がメモリーで行われ、PDBに対して行われて、永続的に格納されます。新しい設定はPDBで即時に有効になり、PDBがクローズして再オープンされた後、またはCDBが停止して再オープンされた後も保持されます。新しい設定は、PDBにのみ影響を及ぼします。

PDBがCDBから切断されると、SCOPE=BOTHまたはSCOPE=SPFILEを使用してPDBに指定された初期化パラメータの値は、PDBのXMLメタデータ・ファイルに追加されます。これらの値は、PDBがCDBに接続したときにPDBでリストアされます。

ノート:

ALTER SYSTEM SET に渡されるパラメータ値は、メモリーまたは spfile で設定される前に、Oracle により内部的に調整されることがあります。たとえば、パラメータ値が素数である必要があるときに素数以外の値を入力すると、Oracle により、値は次の素数に調整されます。パラメータ値は、パラメータ・ビューV$PARAMETER、V$SYSTEM_PARAMETER および V$SPPARAMETER から問い合せることができます。

SID

SID句を使用すると、値を有効にするインスタンスのSIDを指定できます。

- このパラメータがまだ明示的に設定されていないすべてのインスタンスに対してパラメータ値をOracle Databaseで変更する場合は、SID = '*'を指定します。
- sidのインスタンスのみでパラメータ値を変更する場合、SID = 'sid'を指定します。この設定は、SID = '*'を指定する前後のALTER SYSTEM SET文より優先されます。

この句を指定しない場合は、次のようになります。

- pfile(従来のプレーン・テキストの初期化パラメータ・ファイル)を使用してインスタンスを起動する場合、現行のインスタンスのSIDを指定したとみなされます。
- spfile(サーバー・パラメータ・ファイル)を使用してインスタンスを起動する場合、SID = '*'を指定したとみなされま

す。

現行のインスタンス以外のインスタンスを指定すると、そのインスタンスに、メモリーのパラメータ値の変更を通知するメッセージが送信されます。

USE_STORED_OUTLINES句

> ノート:
>
> ストアド・アウトラインは非推奨になりました。ストアド・アウトラインは、下位互換性を保つために今でもサポートされています。ただし、かわりに SQL 計画管理を使用することをお薦めします。SQL 計画管理の詳細は、Oracle Database SQL チューニング・ガイドを参照してください。

USE_STORED_OUTLINESは、システム・パラメータであり、初期化パラメータではありません。pfileまたはspfile内に設定することはできませんが、ALTER SYSTEM文とともに設定できます。このパラメータは、オプティマイザが実行計画を生成するためにストアド・パブリック・アウトラインを使用するかどうかを指定します。

- TRUEに設定すると、要求をコンパイルするときに、オプティマイザはDEFAULTカテゴリのストアド・アウトラインを使用します。
- FALSEに設定すると、オプティマイザはストアド・アウトラインを使用しません。これはデフォルトです。
- category_nameに設定すると、要求をコンパイルするときに、オプティマイザはcategory_nameカテゴリのストアド・アウトラインを使用します。

GLOBAL_TOPIC_ENABLED

GLOBAL_TOPIC_ENABLEDは、システム・パラメータであり、初期化パラメータではありません。pfileまたはspfile内に設定することはできませんが、ALTER SYSTEM文とともに設定できます。GLOBAL_TOPIC_ENABLED = TRUEの場合、キュー表が作成、変更または削除されると、対応するLightweight Directory Access Protocol(LDAP)エントリも作成、変更または削除されます。

このパラメータは、Java Message Service(JMS)に対しても同様に機能します。LDAPを使用するようにデータベースが構成されており、GLOBAL_TOPIC_ENABLEDパラメータがTRUEに設定されている場合、すべてのJMSキューおよびトピックは作成時にLDAPサーバーに自動的に登録されます。管理者は、LDAPに登録されたキューおよびトピックの別名を作成することもできます。LDAPに登録されたキューおよびトピックは、JNDIを介してキューまたはトピックの名前または別名を使用してルックアップできます。

共有サーバーのパラメータ

インスタンスを起動すると、Oracle DatabaseはSHARED_SERVERSおよびDISPATCHERS初期化パラメータの値に基づく共有サーバー・アーキテクチャの共有サーバー・プロセスおよびディスパッチャ・プロセスを作成します。ALTER SYSTEM文でSHARED_SERVERSおよびDISPATCHERSパラメータを設定し、インスタンスの実行中に次の操作のいずれかを実行できます。

- 共有サーバー・プロセスの最小値を増やして、追加の共有サーバー・プロセスを作成します。
- 現行のコールが処理を終了した後、既存の共有サーバー・プロセスを終了します。
- 特定のプロトコルに対するディスパッチャ・プロセスをより多く作成します。プロトコル全体で、初期化パラメータMAX_DISPATCHERSによって指定される数まで作成できます。

- 現行のユーザー・プロセスがインスタンスから切断した後、特定のプロトコルに対する既存のディスパッチャ・プロセスを終了します。

関連項目:

- Oracle Real Application Clusters環境の各インスタンス用のパラメータ値設定の詳細は、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください
- ALTER SYSTEM文を使用した例である「ライセンス・パラメータの変更: 例」、「クエリー・リライトの有効化: 例」、「リソース制限の有効化: 例」、「共有サーバーのパラメータ」および「共有サーバー設定の変更: 例」を参照してください。

alter_system_reset_clause

この句を使用すると、初期化パラメータをリセットできます。

この句のセマンティクスは、初期化パラメータの値を変更するかわりに、初期化パラメータの設定を削除する点を除き、`set_parameter_clause`に似ています。リセットできるパラメータおよびSCOPEおよびSID句のセマンティクスについて学習するには、「set_parameter_clause」を参照してください。

RELOCATE CLIENT

この句は、Oracle Flex ASMを使用している場合にのみ有効です。この句の発行は、通常のデータベース・インスタンスからではなく、Oracle ASMインスタンスから行う必要があります。

この句を使用すると、特定のクライアントを最も負荷が低いOracle ASMインスタンスに再配置できます。この句を発行すると、クライアントとの接続が終了し、クライアントは最も負荷が低いインスタンスにフェイルオーバーします。クライアントが最も少なくロードされたインスタンスに現在接続している場合、クライアントへの接続は終了し、クライアントはその同じインスタンスにフェイルオーバーします。

`client_id`には、次の形式の文字列を一重引用符で囲って指定します。

```
instance_name:db_name
```

ここで、`instance_name`はクライアントの識別子、`db_name`はクライアントのデータベース名です。これらの値を確認するには、`V$ASM_CLIENT`動的パフォーマンス・ビューの`INSTANCE_NAME`列と`DB_NAME`列を問い合せます。

関連項目:

- Oracle Flex ASMの管理の詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。
- `V$ASM_CLIENT`動的パフォーマンス・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

cancel_sql_clause

この句を使用して、パラレル・サーバーなどのリソースを過度に消費しているSQL操作を終了します。アクティブなSQL文を取り消すセッションのセッションIDおよびセッション・シリアル番号を指定する必要があります。セッションがアイドル状態(アクティブにSQL文を実行していない)場合、次のSQL文が取り消されます。次のSQL文が取り消されないようにするには、取り消すSQL文を識別するために引数に`sql_id`を指定します。

- `session_id`は必須であり、セッション識別子を表します。

- serial_numberで必須であり、セッションのシリアル番号を表します。
- instance_idはオプションです。この引数を省略すると、現在のセッションのインスタンスIDが使用されます。
- sql_idはオプションです。この引数が指定されている場合、sql_idは、SQLを終了する前にセッション内でアクティブに実行されているSQL文と一致します。セッションでsql_id引数で指定されたSQL文以外のSQL文が実行されている場合は、エラーが発生します。

FLUSH PASSWORDFILE_METADATA_CACHE

パスワード・ファイルの名前または場所を変更する場合は、その変更が起こるデータベースに通知する必要があります。コマンドALTER SYSTEM FLUSH PASSWORDFILE_METADATA_CACHEは、SGAに格納されているパスワード・ファイルのメタデータ・キャッシュをフラッシュし、変更があったことをデータベースに通知します。

クラスタ環境で実行されている場合には、このコマンドでRACインスタンスすべてからもキャッシュがフラッシュされます。全インスタンス間で変更の伝播に時間がかかることもあります。フラッシュが完全に伝播されるまでは、一部のインスタンスで古いパスワード・ファイルが使い続けられる可能性があります。

例

REDOログの手動アーカイブ: 例

次の文は、SCN 9356083のREDOログ・エントリを含むREDOログ・ファイル・グループを手動でアーカイブします。

```
ALTER SYSTEM ARCHIVE LOG CHANGE 9356083;
```

次の文は、メンバー'diskl:log6.log'を含むREDOログ・ファイル・グループを、'diska:[arch$]'という場所にあるアーカイブREDOログ・ファイルに手動でアーカイブします。

```
ALTER SYSTEM ARCHIVE LOG
    LOGFILE 'diskl:log6.log'
    TO 'diska:[arch$]';
```

クエリー・リライトの有効化: 例

次の文は、クエリー・リライトが明示的に無効にされていないすべてのマテリアライズド・ビューに対するすべてのセッションで、クエリー・リライトを有効にします。

```
ALTER SYSTEM SET QUERY_REWRITE_ENABLED = TRUE;
```

セッションの制限: 例

たとえば、アプリケーションのメンテナンス中は、RESTRICTED SESSIONシステム権限が付与されているアプリケーション開発者のみがログインできるようにセッションを制限できます。このためには、次の文を発行します。

```
ALTER SYSTEM
   ENABLE RESTRICTED SESSION;
```

次に、ALTER SYSTEM文のKILL SESSION句を使用すると、既存のセッションをどれでも終了できます。

アプリケーションのメンテナンスが終了した後で、次の文を発行することによって、CREATE SESSIONシステム権限が付与されているユーザーもログインできるようになります。

```
ALTER SYSTEM
   DISABLE RESTRICTED SESSION;
```

ウォレットおよび暗号化キーの確立: 例

次の文は、サーバー・ウォレットからメモリーに情報をロードし、透過的データ暗号化マスター・キーを設定します。

```
ALTER SYSTEM SET ENCRYPTION WALLET OPEN IDENTIFIED BY "password";
ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY "password";
```

これらの文では、セキュリティ・モジュールが初期化済で、passwordを使用してウォレットが作成済であることを前提としています。

ウォレットのクローズ: 例

次の文は、パスワード・ベースのウォレット情報をメモリーから削除します。

```
ALTER SYSTEM SET ENCRYPTION WALLET CLOSE IDENTIFIED BY "password";
```

次の文を使用すると、パスワードベースのウォレット情報および自動ログイン情報(存在する場合)をメモリーから削除できます。

```
ALTER SYSTEM SET ENCRYPTION WALLET CLOSE;
```

共有プールのクリア: 例

たとえば、パフォーマンス分析を開始する前に、共有プールをクリアできます。共有プールをクリアする場合、次の文を発行します。

```
ALTER SYSTEM FLUSH SHARED_POOL;
```

チェックポイントの強制実行: 例

次の文は、チェックポイントを強制実行します。

```
ALTER SYSTEM CHECKPOINT;
```

リソース制限の有効化: 例

このALTER SYSTEM文は、リソース制限を動的に有効にします。

```
ALTER SYSTEM SET RESOURCE_LIMIT = TRUE;
```

共有サーバー設定の変更: 例

次の文は、共有サーバー・プロセスの最小数を25に変更します。

```
ALTER SYSTEM SET SHARED_SERVERS = 25;
```

共有サーバー・プロセスの数が25より少ない場合は追加作成されます。共有サーバー・プロセスが25より多く、25あれば負荷を管理できる場合は、現行のコールによる処理が終了した時点で、25を超える分の共有サーバー・プロセスは終了します。

次の文は、TCP/IPプロトコルのディスパッチャ・プロセス数を5に、ipcプロトコルのディスパッチャ・プロセス数を10に動的に変更します。

```
ALTER SYSTEM
   SET DISPATCHERS =
      '(INDEX=0)(PROTOCOL=TCP)(DISPATCHERS=5)',
      '(INDEX=1)(PROTOCOL=ipc)(DISPATCHERS=10)';
```

TCPのディスパッチャ・プロセスの数が5より少ない場合、ディスパッチャ・プロセスが新しく作成されます。5より多い場合は、接続されているユーザーが接続を切断した後に、5を超える分のディスパッチャ・プロセスは終了します。

ipcのディスパッチャ・プロセスの数が10より少ない場合、ディスパッチャ・プロセスが新しく作成されます。10より多い場合は、接続されているユーザーが接続を切断した後に、10を超える分のディスパッチャ・プロセスは終了します。

これ以外のプロトコル用として既存ディスパッチャがある場合、この文は、そのディスパッチャの数に影響しません。

ライセンス・パラメータの変更: 例

次の文は、インスタンスにおけるセッションの最大数を64に、セッションの警告しきい値を54に動的に変更します。

```
ALTER SYSTEM
   SET LICENSE_MAX_SESSIONS = 64
   LICENSE_SESSIONS_WARNING = 54;
```

セッション数が54に達した場合、後続の各セッションのALERTファイルに警告メッセージが書き込まれます。RESTRICTED SESSIONシステム権限を持つユーザーも、後続セッションを開始する場合に、警告メッセージを受け取ります。

セッション数が64に達した場合、セッション数が再び64を下回るまでは、RESTRICTED SESSIONシステム権限を持つユーザー以外は新しいセッションを開始できません。

次の文は、インスタンスのセッションの最大数を動的に使用禁止にします。この文の実行後は、インスタンスのセッション数に制限がなくなります。

```
ALTER SYSTEM SET LICENSE_MAX_SESSIONS = 0;
```

次の文は、データベースのユーザー数の制限を200に動的に変更します。この文の実行後は、データベース・ユーザー数が200を超えることはありません。

```
ALTER SYSTEM SET LICENSE_MAX_USERS = 200;
```

ログ・スイッチの強制実行: 例

強制的なログ・スイッチが必要になるのは、たとえば現行のREDOログ・ファイル・グループまたはそのメンバーの削除や名前の変更を行う場合です(Oracle Databaseが書き込んでいるファイルの削除や名前変更はできないため)。強制ログ・スイッチの影響を受けるのは、このインスタンスのREDOログ・スレッドのみです。次の文は、ログ・スイッチを強制します。

```
ALTER SYSTEM SWITCH LOGFILE;
```

分散リカバリの有効化: 例

次の文は、分散リカバリを有効にします。

```
ALTER SYSTEM ENABLE DISTRIBUTED RECOVERY;
```

デモンストレーションまたはテストの目的で、分散リカバリを使用禁止にする場合があります。次の文は、シングルプロセス・モードとマルチプロセス・モードの両方で分散リカバリを使用禁止にします。

```
ALTER SYSTEM DISABLE DISTRIBUTED RECOVERY;
```

デモンストレーションまたはテストが終了した場合、ALTER SYSTEM文にENABLE DISTRIBUTED RECOVERY句を指定して実行すると、分散リカバリを再び使用可能にできます。

セッションの終了: 例

たとえば、他のユーザーが必要とするリソースを保持しているユーザーのセッションを終了できます。そのユーザーは、セッションが終了したことを示すエラー・メッセージを受け取ります。そのユーザーがデータベースに対する呼出しを行うには、新しいセッションを開始する必要があります。V$SESSION動的パフォーマンス表に次のデータがあるとします(ユーザーSYSとoeの両方がセッションを開いています)。

```
SELECT sid, serial#, username
   FROM V$SESSION;
       SID    SERIAL# USERNAME
---------- ---------- ------------------------------
        29         85 SYS
        33          1
```

```
        35          8
        39         23 OE
        40          1
. . .
```

次の文は、V$SESSIONのSID値とSERIAL#値を使用して、ユーザーscottのセッションを終了します。

```
ALTER SYSTEM KILL SESSION '39, 23';
```

セッションの切断: 例

次の文は、V$SESSIONのSIDとSERIAL#の値を使用して、ユーザーscottのセッションを切断します。

```
ALTER SYSTEM DISCONNECT SESSION '13, 8' POST_TRANSACTION;
```

ALTER TABLE

目的

ALTER TABLE文を使用すると、非パーティション表、パーティション表、表パーティションおよび表サブパーティションの定義を変更できます。オブジェクト表またはオブジェクト列を含むリレーショナル表の場合は、ALTER TABLEを使用して型が変更された後に、表を参照する型の最新の定義に変換します。

ノート:

マテリアライズド・ビュー・ログ表の操作では、可能な場合は常に、ALTER TABLE 文ではなく ALTER MATERIALIZED VIEW LOG 文を使用することをお薦めします。

関連項目:

- 表の作成については、「CREATE TABLE」を参照してください。
- Oracle Textとともに使用するALTER TABLE文については、『Oracle Textリファレンス』を参照してください。

前提条件

表が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、その表に対するALTERオブジェクト権限またはALTER ANY TABLEシステム権限が必要です。

パーティション化操作のその他の前提条件

表の所有者ではない場合、drop_table_partitionまたはtruncate_table_partition句を使用するには、DROP ANY TABLE権限が必要です。

add_table_partition、modify_table_partition、move_table_partitionおよびsplit_table_partition句を使用する場合、領域を確保する表領域に領域割当て制限が必要です。

パーティション化操作が子の参照パーティション表にカスケードする場合、子の参照パーティション表では権限は必要ありません。

exchange_partition_subpart句を使用する際、交換される表データにID列が含まれており、自分が交換に関与する両方の表の所有者ではない場合は、ALTER ANY SEQUENCEシステム権限が必要です。

オブジェクト型を持つ非パーティション表はパーティション化できません。

制約およびトリガーのその他の前提条件

一意キー制約または主キー制約を有効にする場合は、表に索引を作成するための権限が必要です。Oracle Databaseでは、表を含むスキーマにおいて、一意キーまたは主キーの列に索引を作成するため、この権限が必要になります。

トリガーを使用可能または使用禁止にする場合、トリガーが自分のスキーマ内にある必要があります。自分のスキーマにない場合は、ALTER ANY TRIGGERシステム権限が必要です。

関連項目:

索引を作成する場合に必要な権限については、「CREATE INDEX」を参照してください。

オブジェクト型を使用する場合のその他の前提条件

表を変更するときに列定義でオブジェクト型を使用する場合、そのオブジェクトが、変更する表と同じスキーマに属している必要があります。または、EXECUTE ANY TYPEシステム権限またはそのオブジェクト型に対するEXECUTEオブジェクト権限が必要です。

フラッシュバック・データ・アーカイブ操作のその他の前提条件

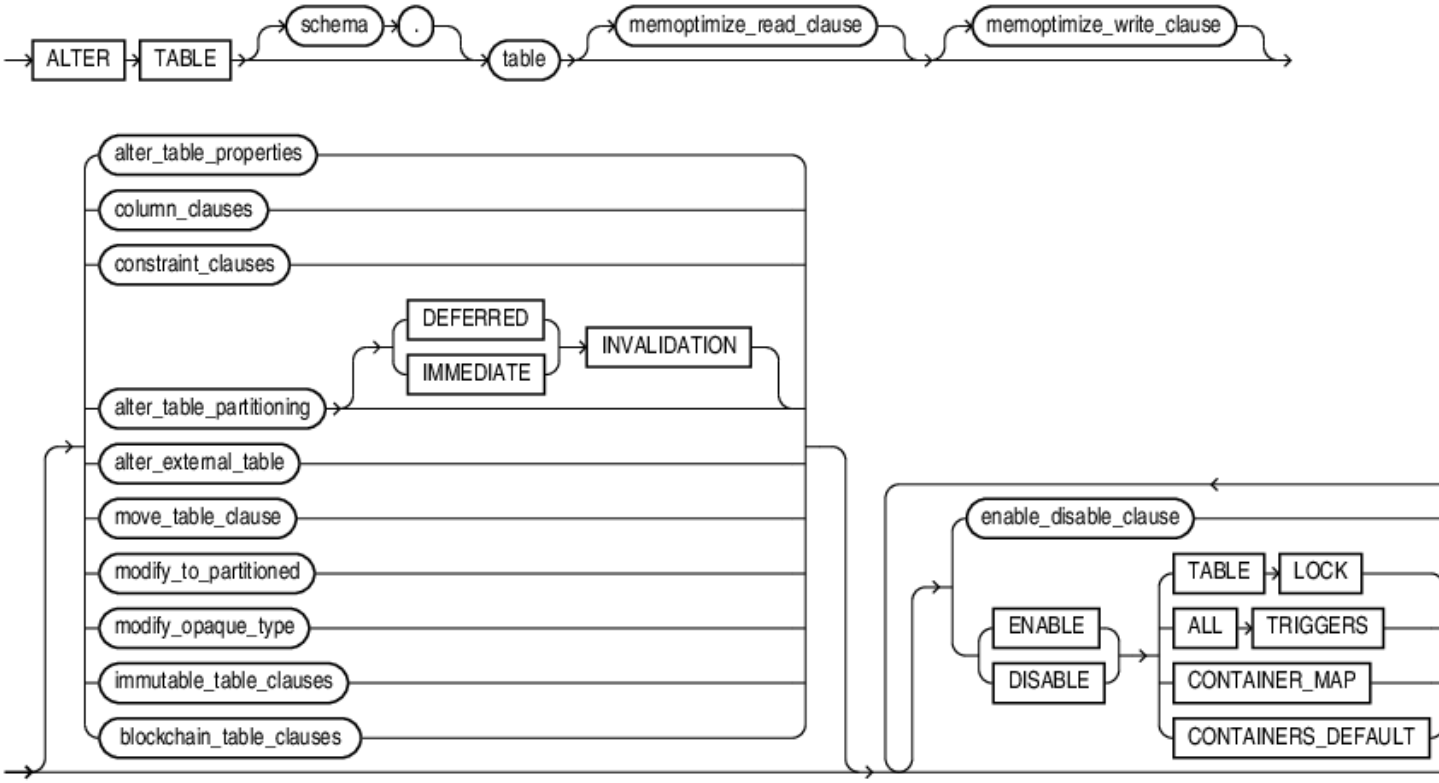
flashback_archive_clauseを使用して表の履歴追跡を有効化するには、履歴データが格納されるフラッシュバック・アーカイブに対するFLASHBACK ARCHIVEオブジェクト権限が必要です。flashback_archive_clauseを使用して表の履歴追跡を無効にするには、FLASHBACK ARCHIVE ADMINSTERシステム権限を持っているか、またはSYSDBAとしてログインしている必要があります。

エディション・オブジェクトの参照に対するその他の前提条件

evaluation_edition_clauseまたはunusable_editions_clauseでエディションを指定するには、そのエディションに対するUSE権限が必要になります。

構文

alter_table::=



ノート:

table の後に句を指定する必要があります。必須の句はありませんが、1 つ以上の句を指定する必要があります。

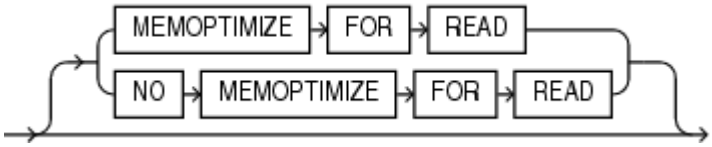
ALTER TABLE構文のグループは、次のとおりです。

- alter_table_properties::=

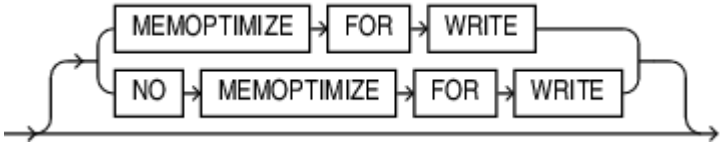
- column_clauses::=
- constraint_clauses::=
- alter_table_partitioning::=
- alter_external_table::=
- move_table_clause::=
- modify_to_partitioned::=
- modify_opaque_type::=
- enable_disable_clause::=

各句の後には、そのコンポーネントの副次句の参照先が記載されています。

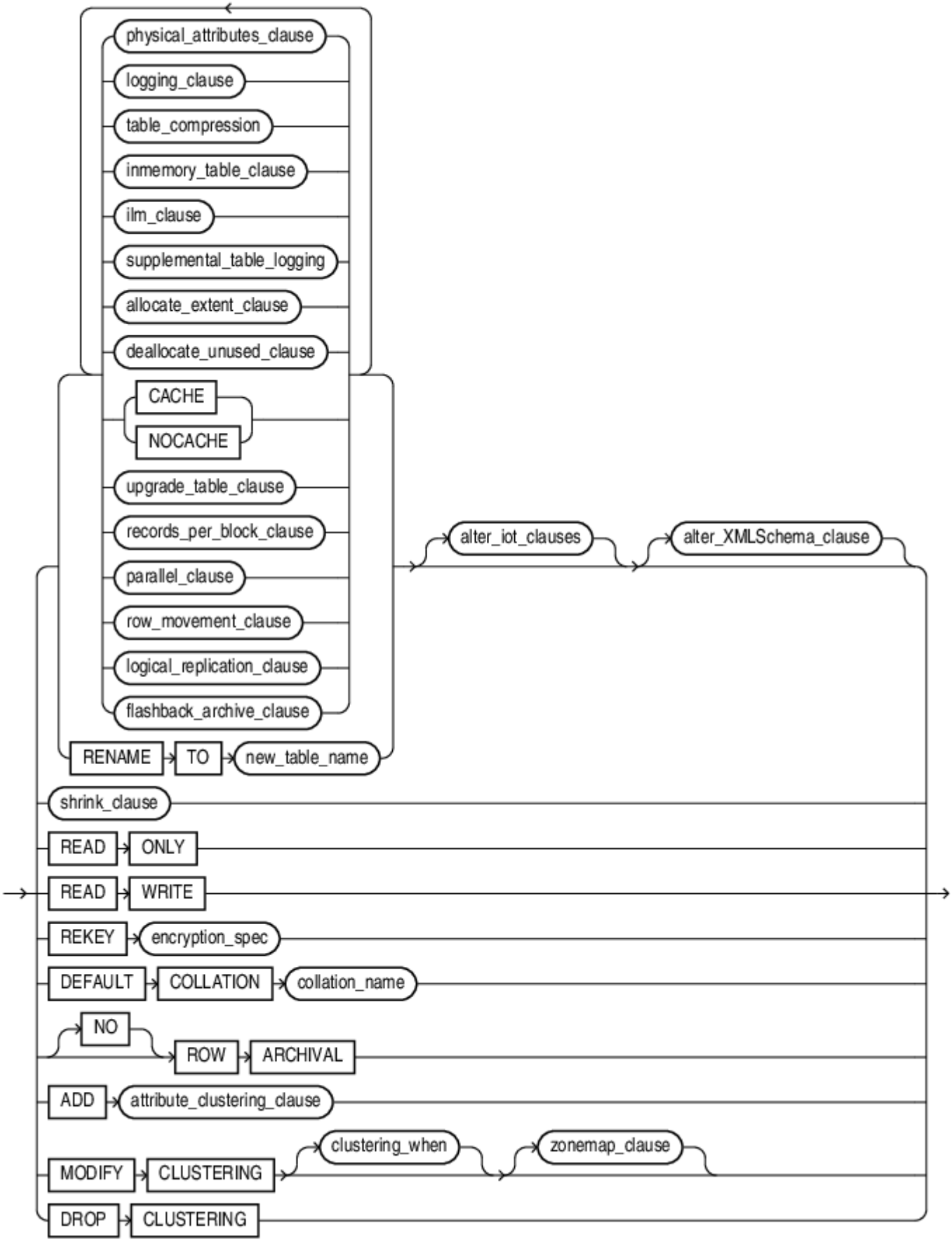
memoptimize_read_clause::=



memoptimize_write_clause



alter_table_properties::=

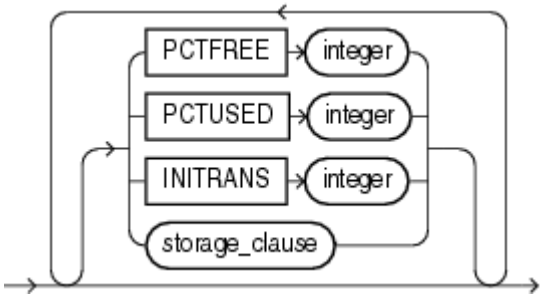

physical_attributes_clause
logging_clause
table_compression
inmemory_table_clause
ilm_clause
supplemental_table_logging
allocate_extent_clause
deallocate_unused_clause
CACHE
NOCACHE
upgrade_table_clause
records_per_block_clause
parallel_clause
row_movement_clause
logical_replication_clause
flashback_archive_clause
RENAME
TO
new_table_name
alter_iot_clauses
alter_XMLSchema_clause
shrink_clause
READ
ONLY
READ
WRITE
REKEY
encryption_spec
DEFAULT
COLLATION
collation_name
NO
ROW
ARCHIVAL
ADD
attribute_clustering_clause
MODIFY
CLUSTERING
clustering_when
zonemap_clause
DROP
CLUSTERING

ノート:

MODIFY CLUSTERING 句を指定する場合、clustering_when または zonemap_clause 句のいずれかを指定する必要があります。

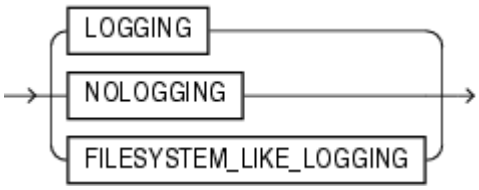
(physical_attributes_clause::=、logging_clause::=、table_compression::=、inmemory_table_clause::=、ilm_clause::=、supplemental_table_logging::=、allocate_extent_clause::=、deallocate_unused_clause::=、upgrade_table_clause::=、records_per_block_clause::=、parallel_clause::=、row_movement_clause::=、flashback_archive_clause::=、shrink_clause::=、attribute_clustering_clause::=、clustering_when::=、zonemap_clause::=、alter_iot_clauses::=、alter_XMLSchema_clause::=)

physical_attributes_clause::=

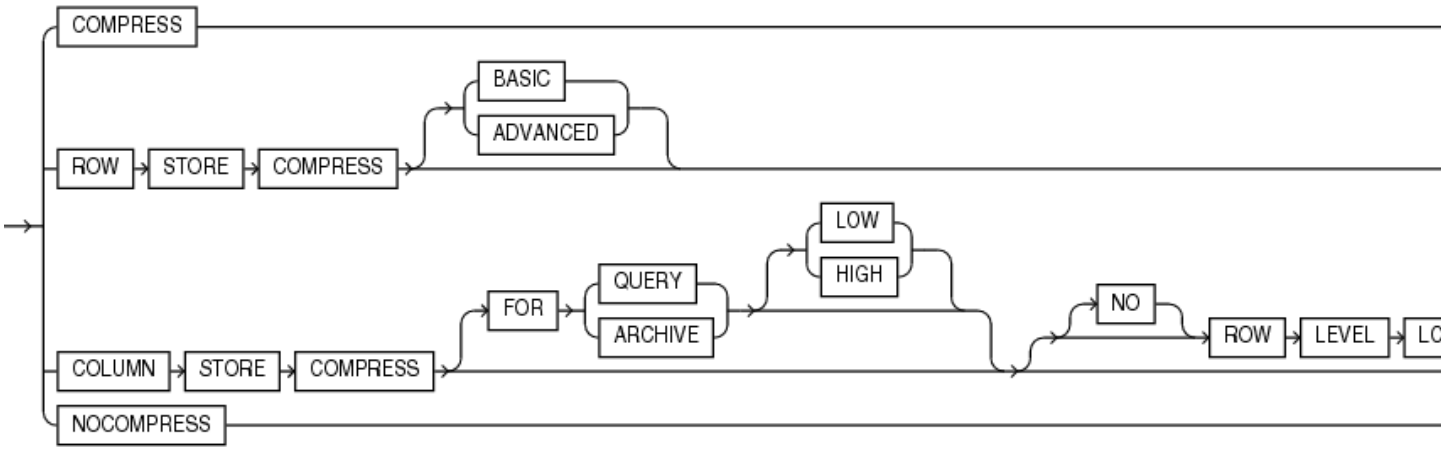


(storage_clause::=)

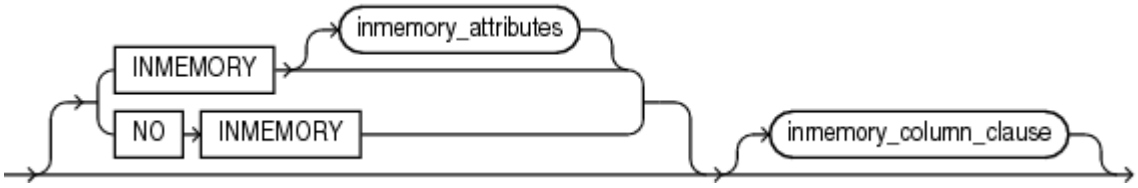
logging_clause::=



table_compression::=



inmemory_table_clause::=

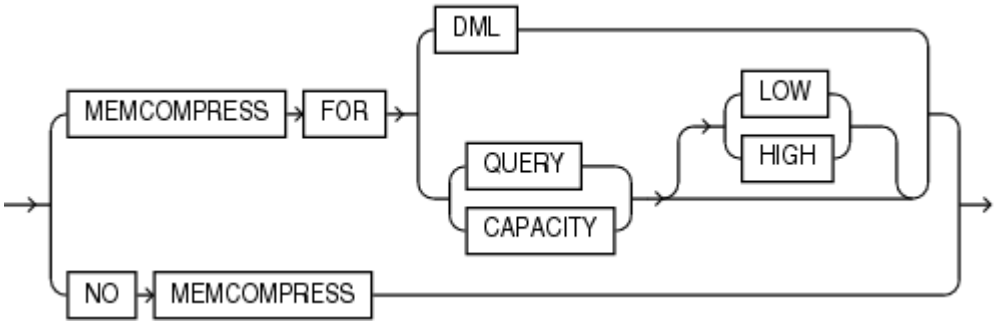


(inmemory_attributes::=、inmemory_column_clause::=)

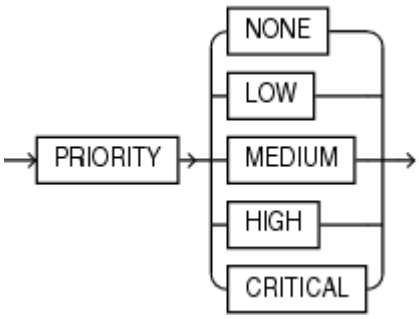
inmemory_attributes::=

(inmemory_memcompress::=、inmemory_priority::=、inmemory_distribute::=、inmemory_duplicate::=)

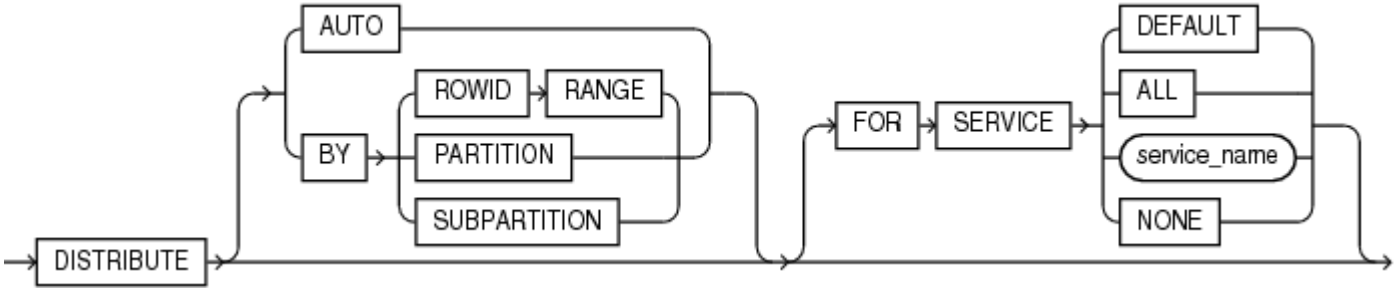
inmemory_memcompress::=



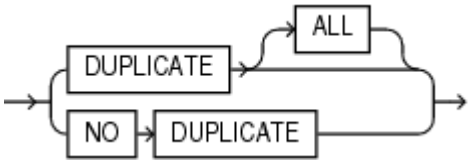
inmemory_priority::=



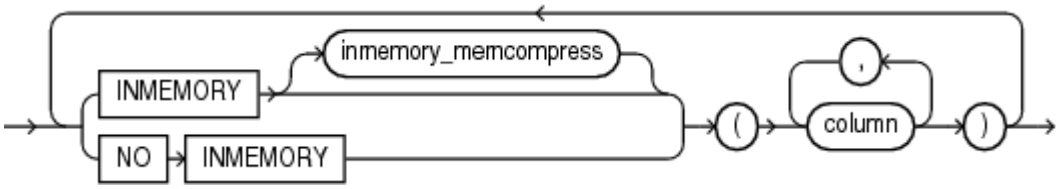
inmemory_distribute::=



inmemory_duplicate::=

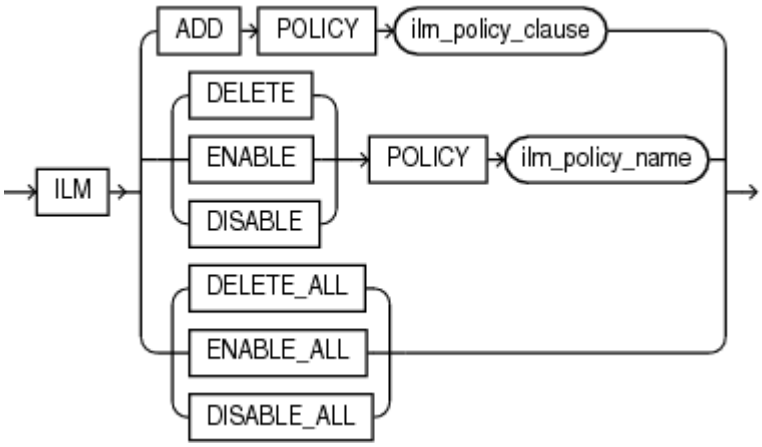


inmemory_column_clause::=

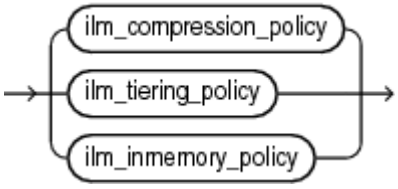


(inmemory_memcompress::=)

ilm_clause::=

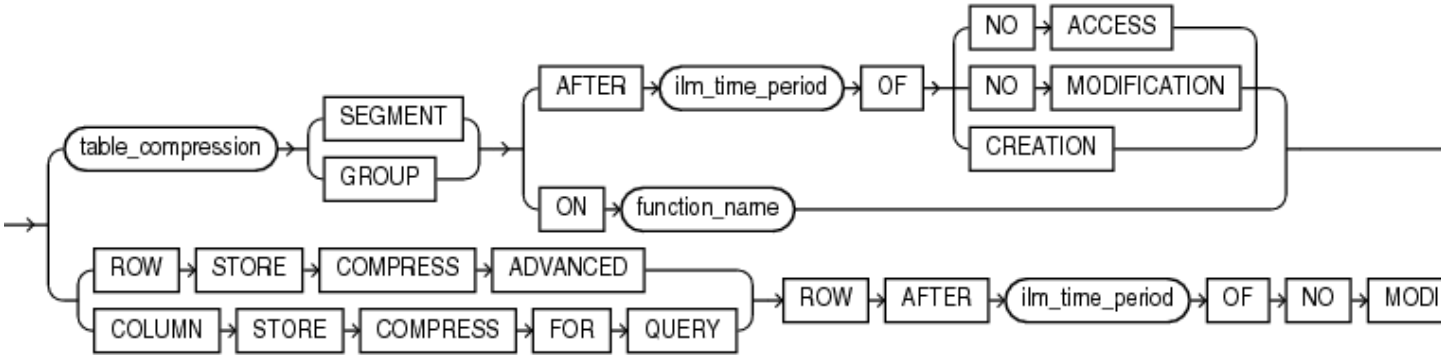


ilm_policy_clause::=



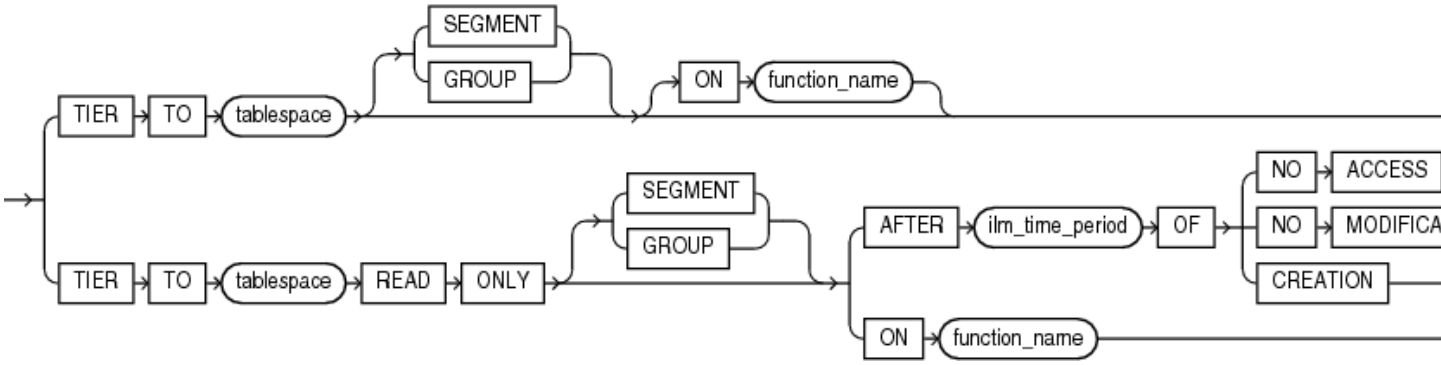
(ilm_compression_policy::=、ilm_tiering_policy::=、ilm_inmemory_policy::=)

ilm_compression_policy::=



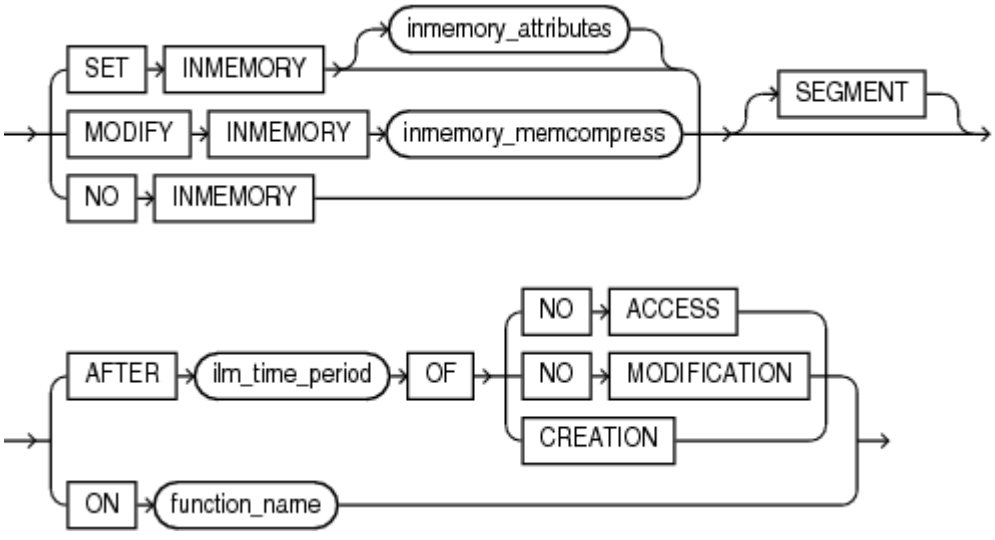
(table_compression::=、ilm_time_period::=)

ilm_tiering_policy::=

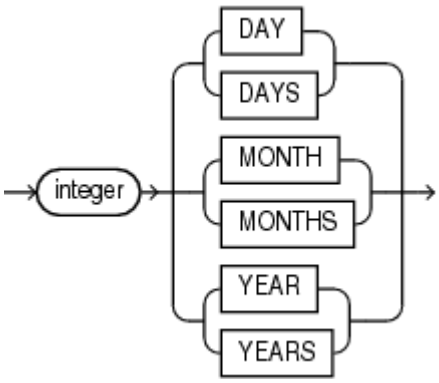


(ilm_time_period::=)

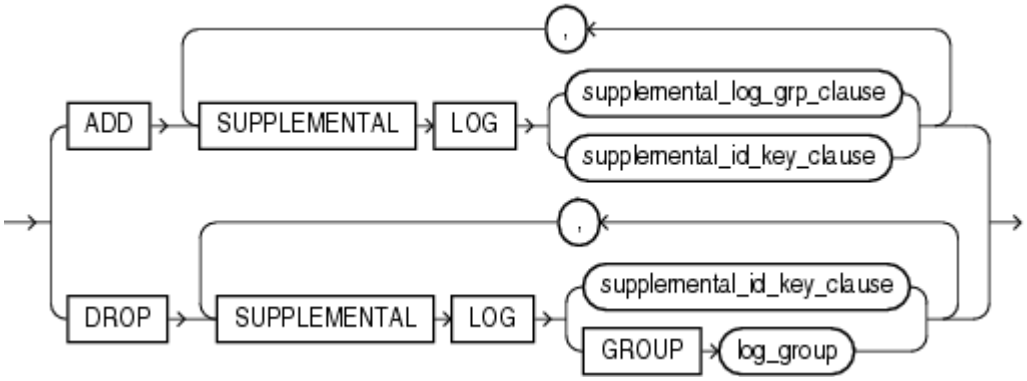
ilm_inmemory_policy::=



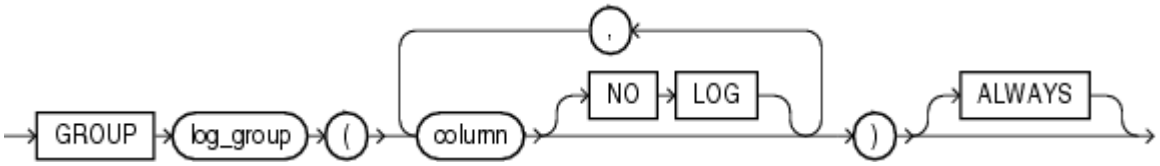
ilm_time_period::=



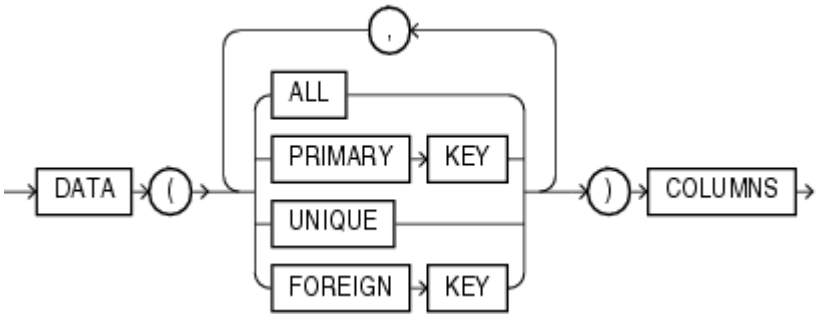
supplemental_table_logging::=



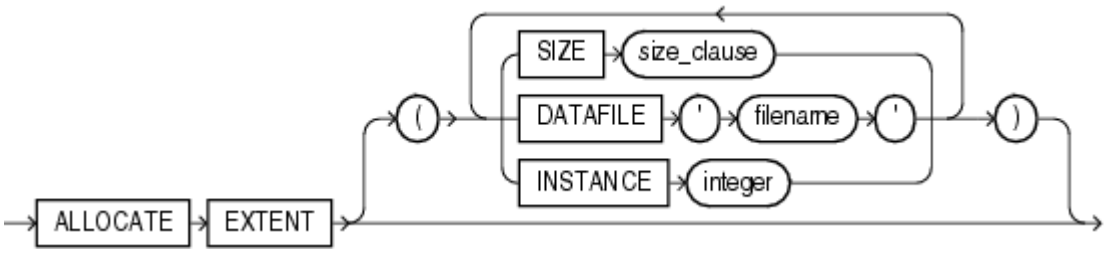
supplemental_log_grp_clause::=



supplemental_id_key_clause::=を参照

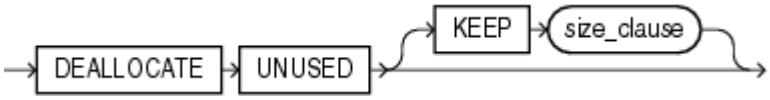


allocate_extent_clause::=



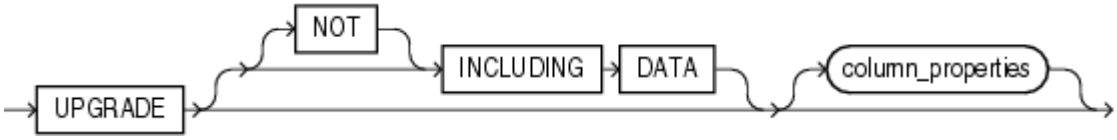
(size_clause::=)

deallocate_unused_clause::=



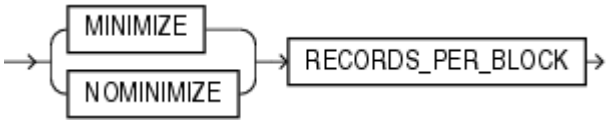
(size_clause::=)

upgrade_table_clause::=

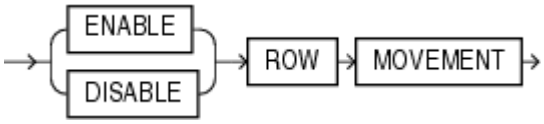


(column_properties::=)

records_per_block_clause::=

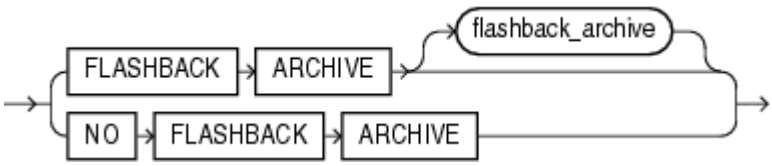


row_movement_clause::=

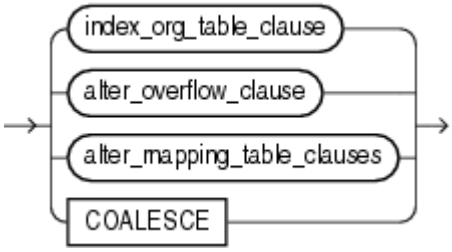


logical_replication_clause

flashback_archive_clause::=

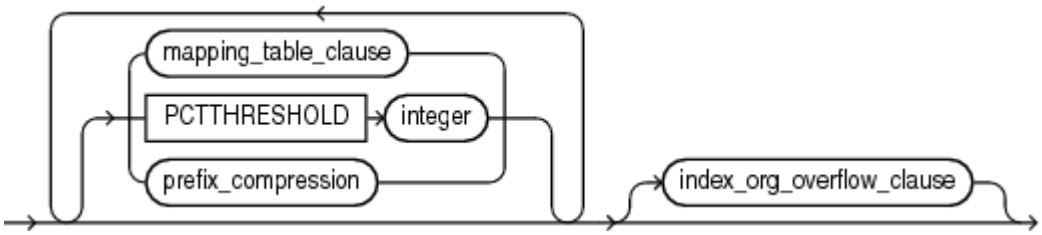


alter_iot_clauses::=



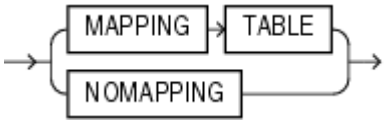
(alter_overflow_clause::=、alter_mapping_table_clauses::=)

index_org_table_clause::=

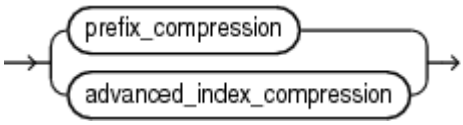


(mapping_table_clauses::=、prefix_compression::=、index_org_overflow_clause::=)

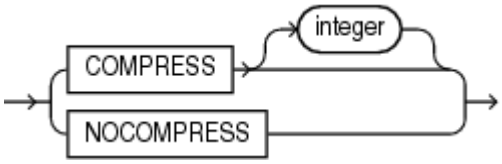
mapping_table_clauses::=



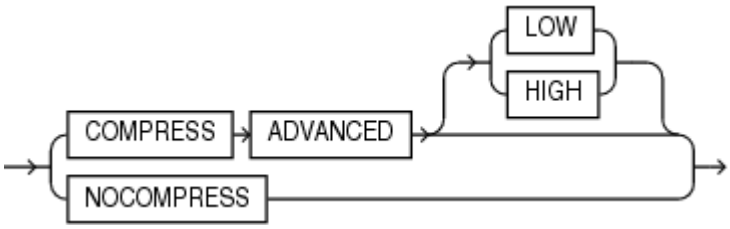
index_compression::=



prefix_compression::=



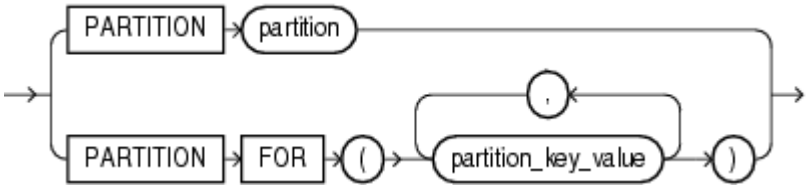
advanced_index_compression::=



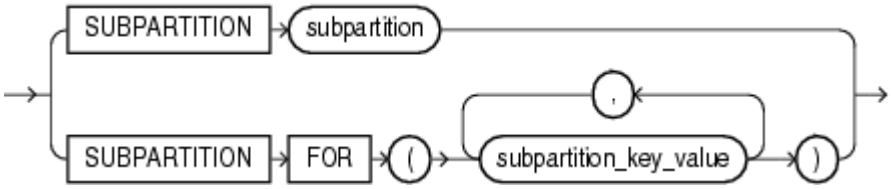
index_org_overflow_clause::=

(segment_attributes_clause::=)

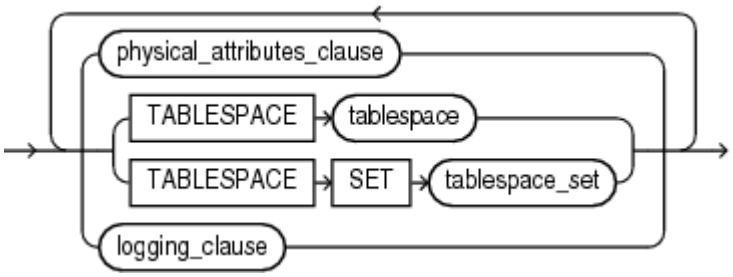
partition_extended_name::=



subpartition_extended_name::=

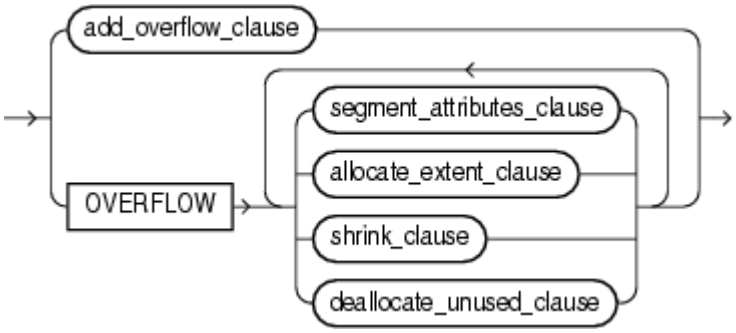


segment_attributes_clause::=



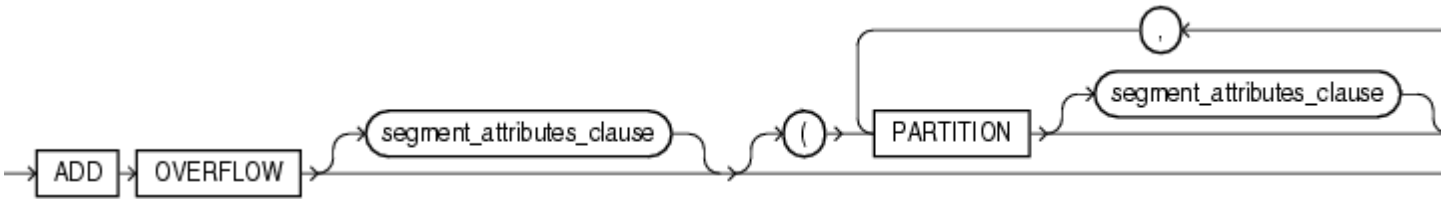
(physical_attributes_clause::=、TABLESPACE SET: ALTER TABLEではサポートされていません、logging_clause::=)

alter_overflow_clause::=



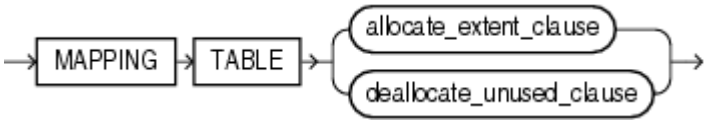
(segment_attributes_clause::=、allocate_extent_clause::=、shrink_clause::=、deallocate_unused_clause::=)

add_overflow_clause::=



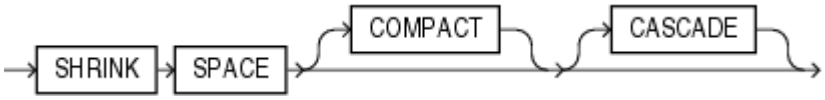
(segment_attributes_clause::=)

alter_mapping_table_clauses::=

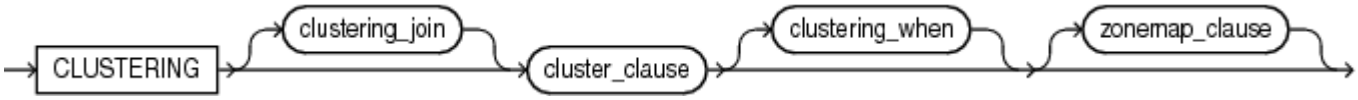


(allocate_extent_clause::=、deallocate_unused_clause::=)

shrink_clause::=

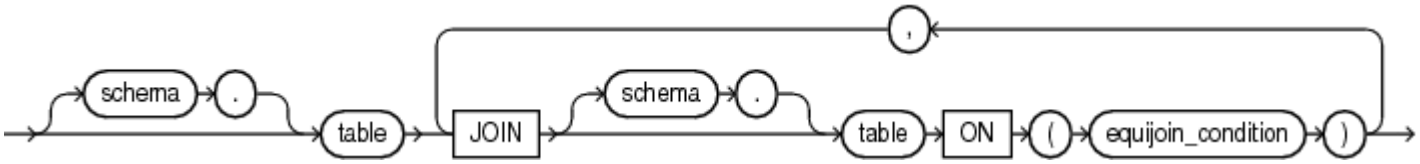


attribute_clustering_clause::=

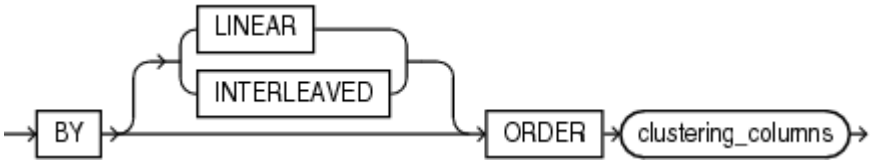


(clustering_join::=、cluster_clause::=、clustering_when::=、zonemap_clause::=)

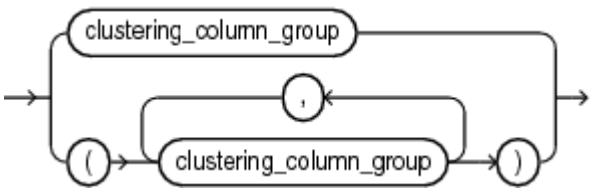
clustering_join::=



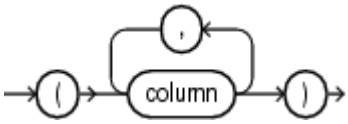
cluster_clause::=



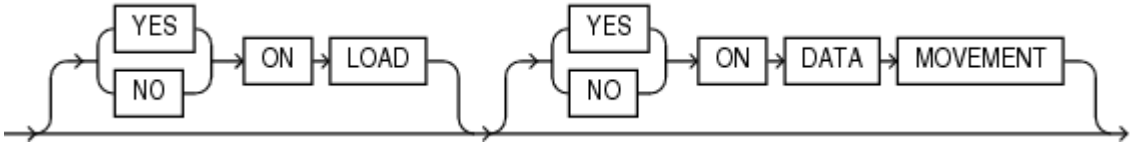
clustering_columns::=



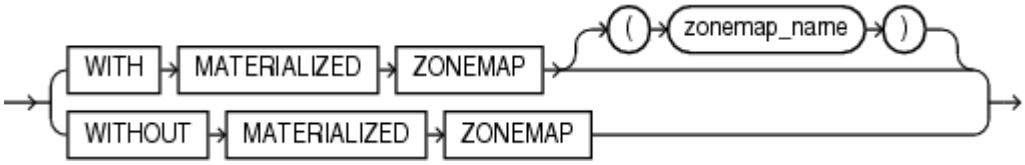
clustering_column_group::=



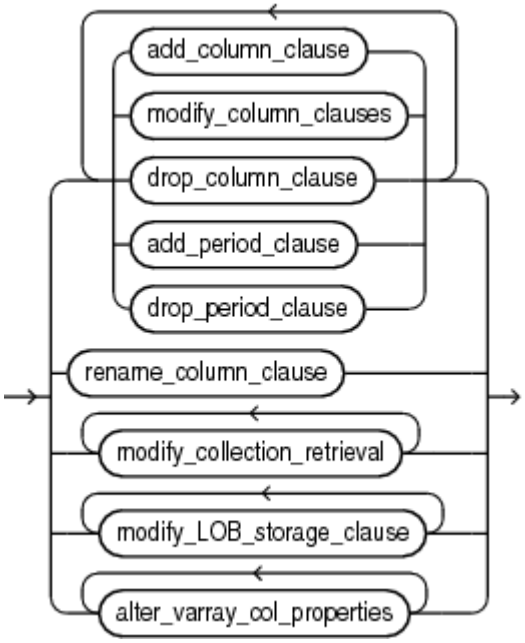
clustering_when::=



zonemap_clause::=

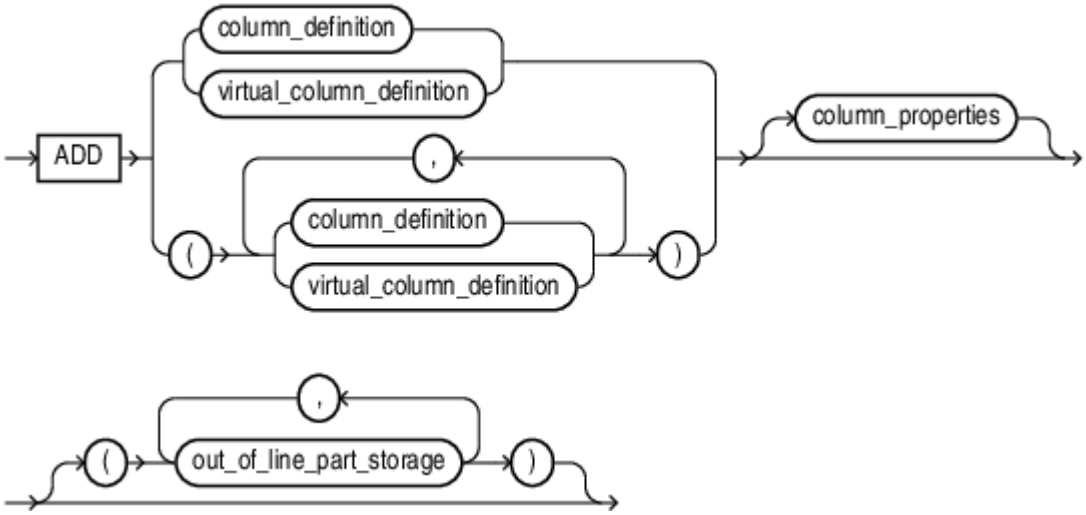


column_clauses::=



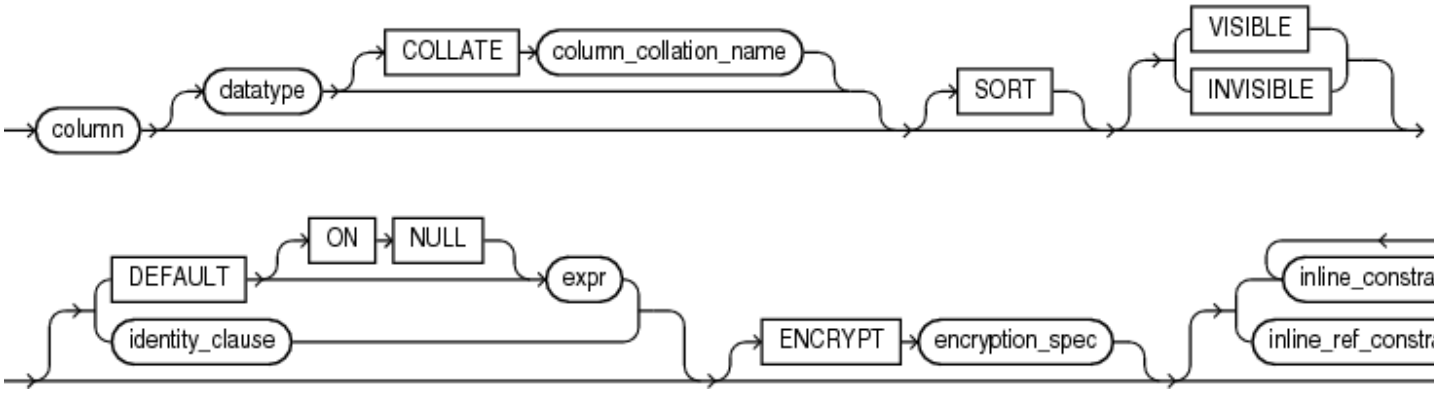
(add_column_clause::=、modify_column_clauses::=、drop_column_clause::=、
add_period_clause::=、drop_period_clause::=、rename_column_clause::=、
modify_collection_retrieval::=、modify_LOB_storage_clause::=、alter_varray_col_properties::=)

add_column_clause::=



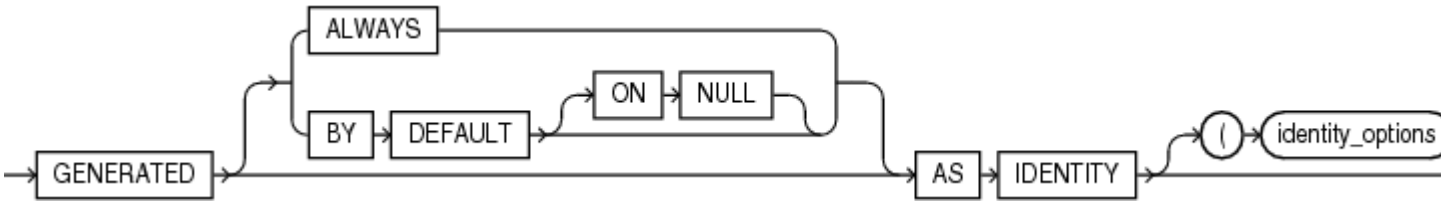
(column_definition::=、virtual_column_definition::=、column_properties::=、out_of_line_part_storage::=)

column_definition::=

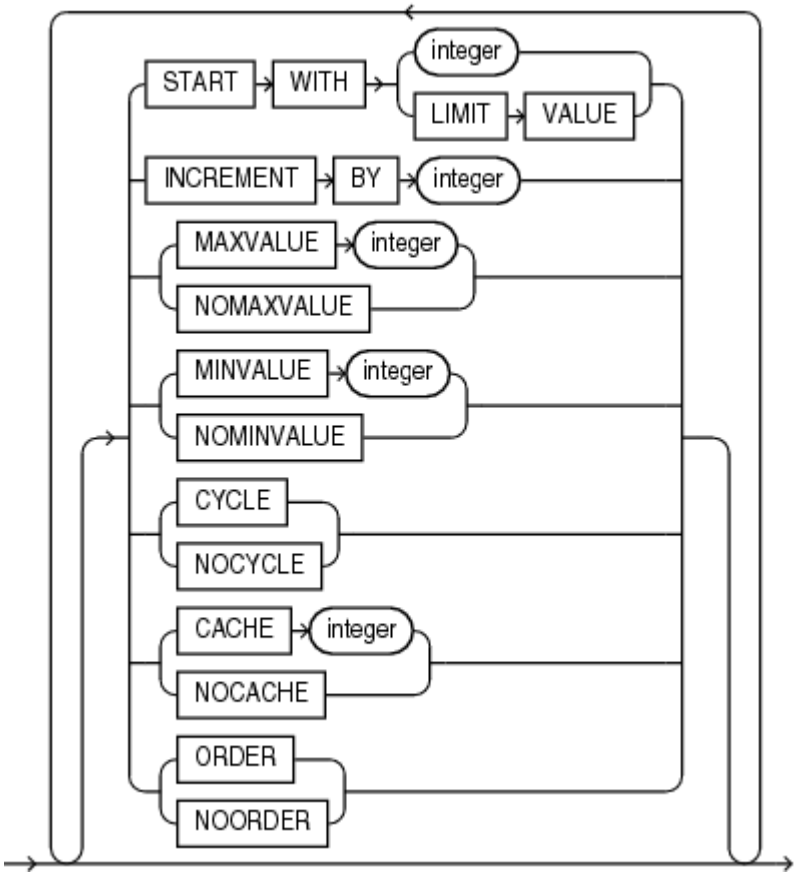


(identity_clause::=、encryption_spec::=、inline_constraintおよびinline_ref_constraint: constraint::=)

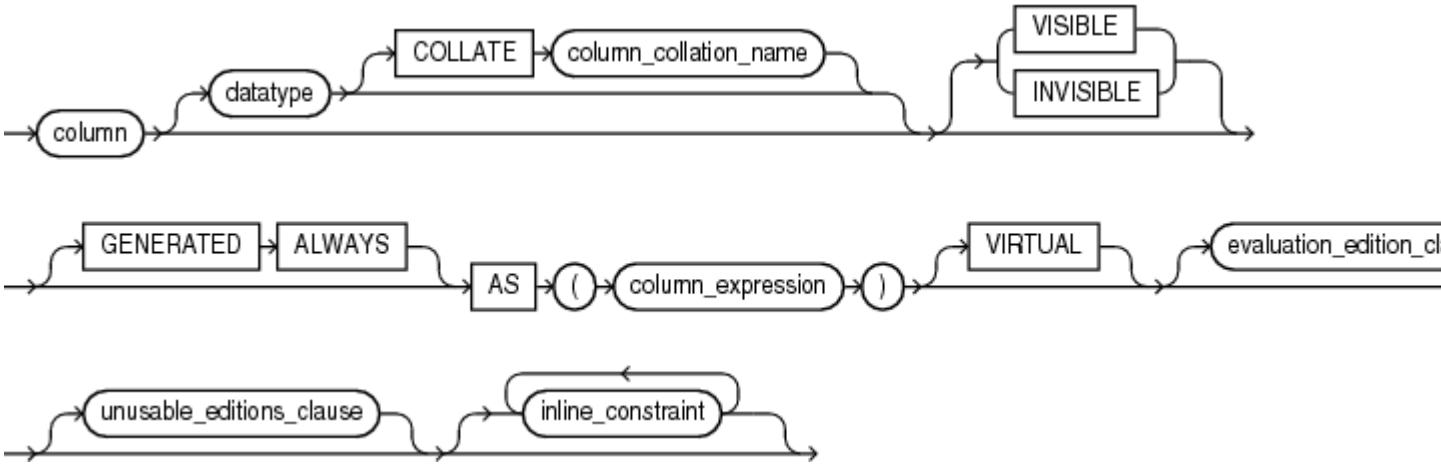
identity_clause::=



identity_options::=

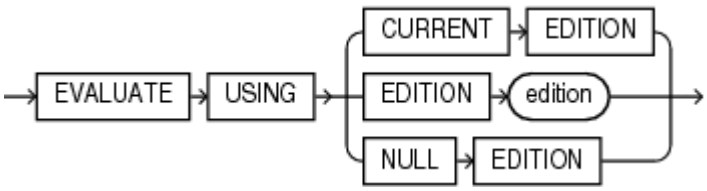


virtual_column_definition::=

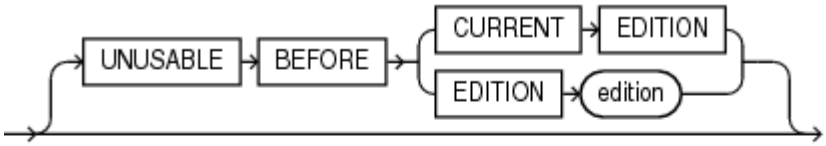


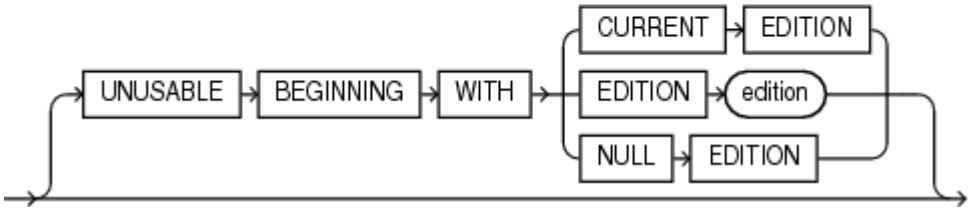
(evaluation_edition_clause::=、unusable_editions_clause::=、constraint::=)

evaluation_edition_clause::=

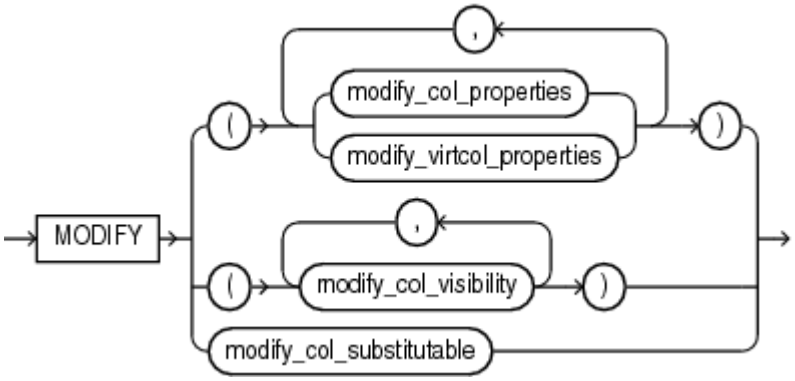


unusable_editions_clause::=



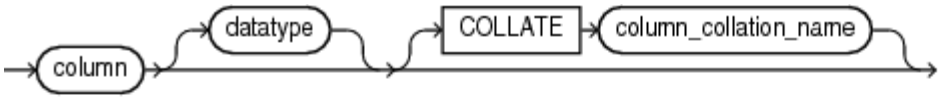


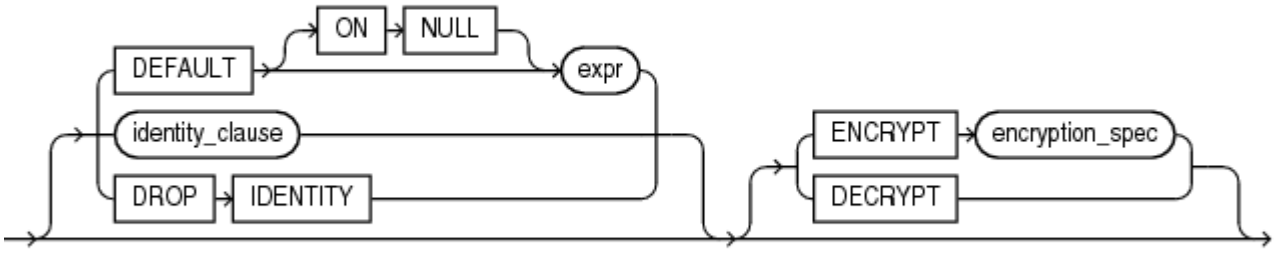
modify_column_clauses::=

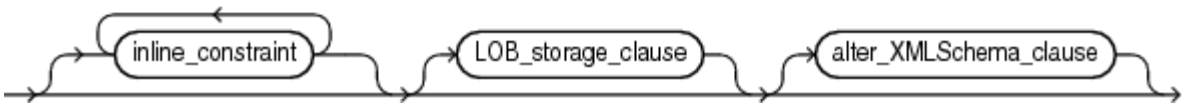


(modify_col_properties::=、modify_virtcol_properties::=、modify_col_visibility::=、modify_col_substitutable::=)

modify_col_properties::=

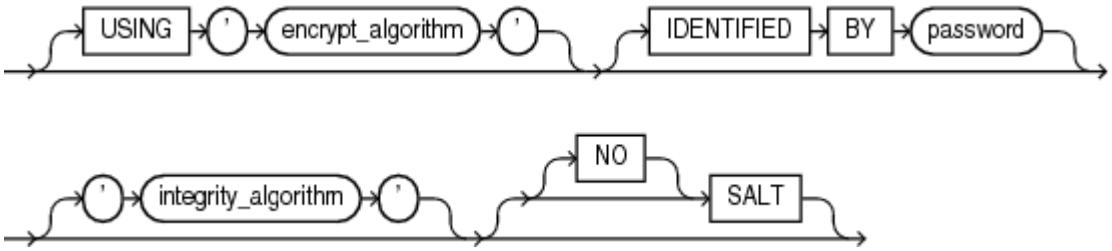




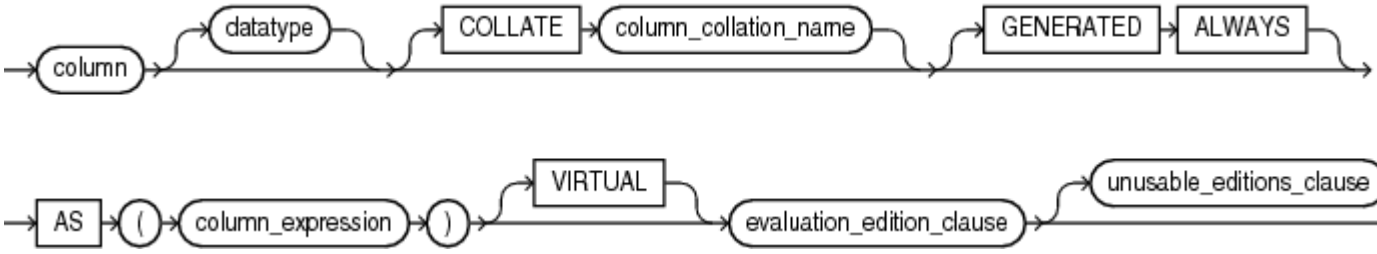


(identity_clause::=、encryption_spec::=、`inline_constraint`: constraint::=、LOB_storage_clause::=、alter_XMLSchema_clause::=)

encryption_spec::=

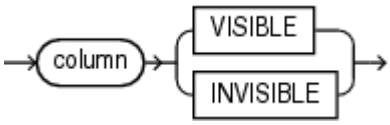


modify_virtcol_properties::=



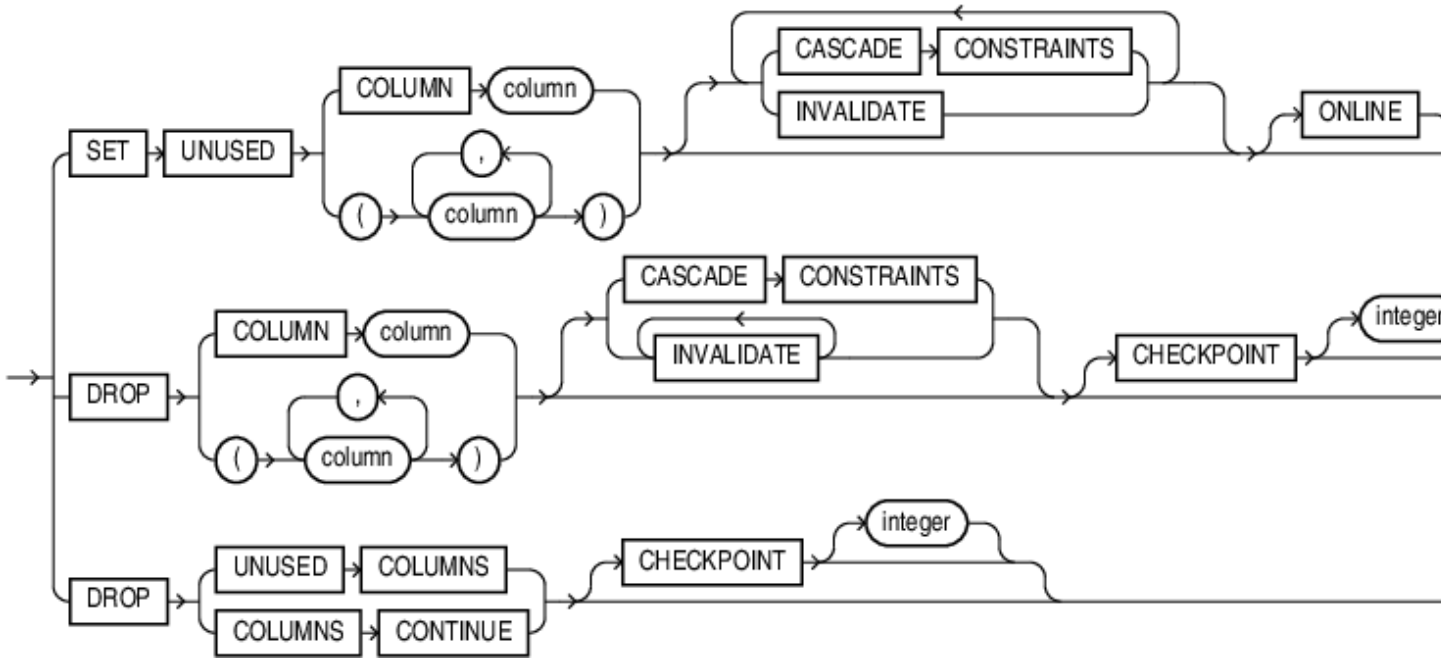
(evaluation_edition_clause::=、unusable_editions_clause::=)

modify_col_visibility::=

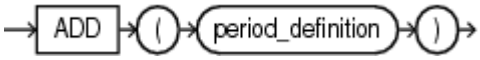


modify_col_substitutable::=

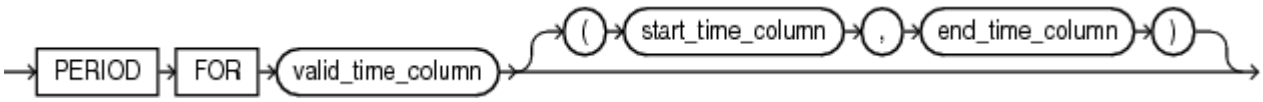
drop_column_clause::=



add_period_clause::=



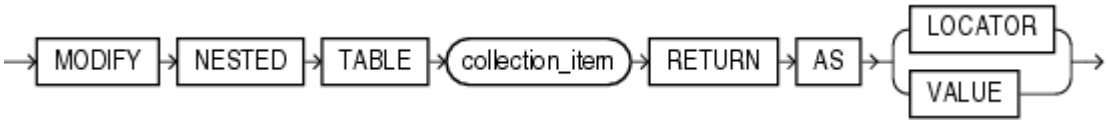
period_definition::=



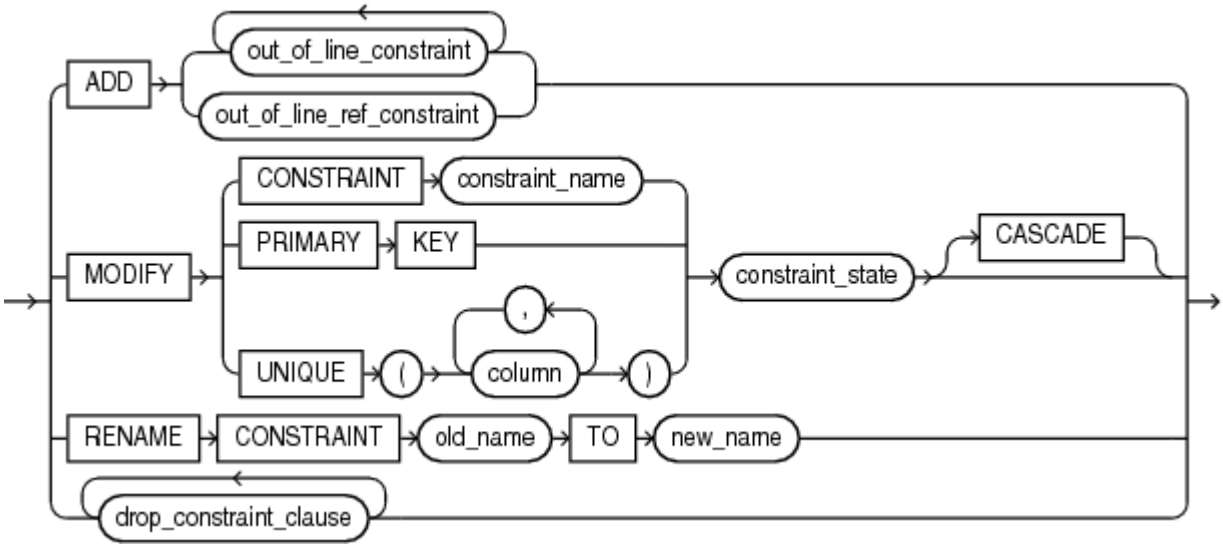
drop_period_clause::=

rename_column_clause::=

modify_collection_retrieval::=

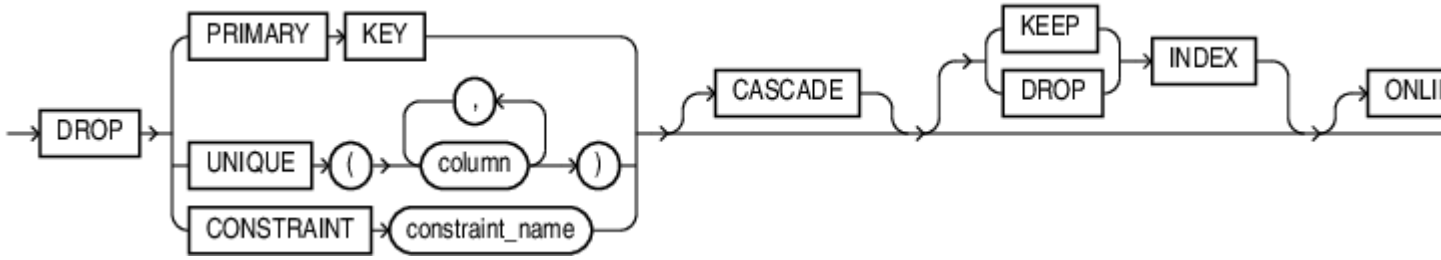


constraint_clauses::=

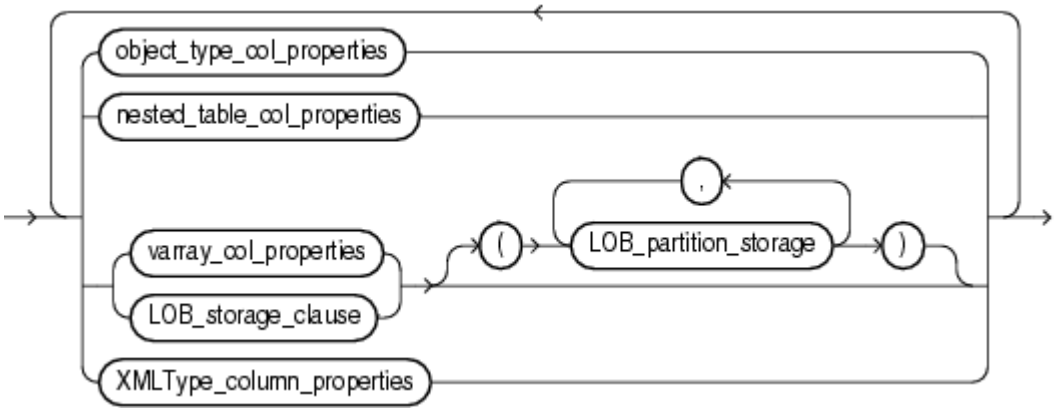


(out_of_line_constraint::=、out_of_line_ref_constraint::=、constraint_state::=)

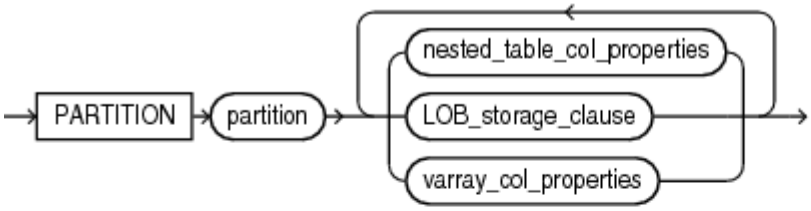
drop_constraint_clause::=

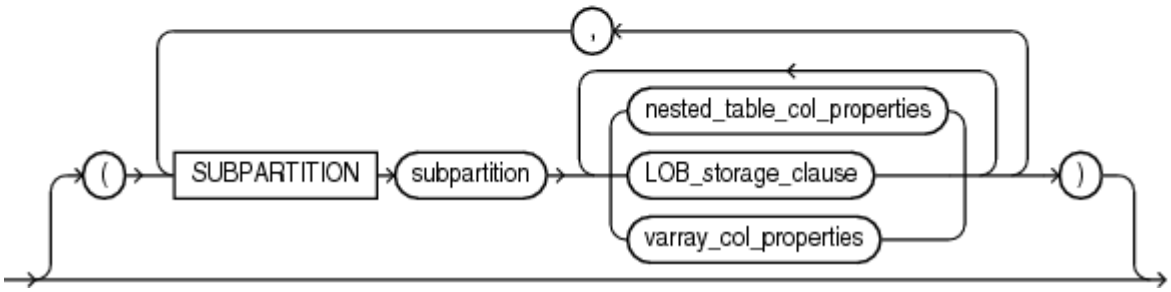


column_properties::=

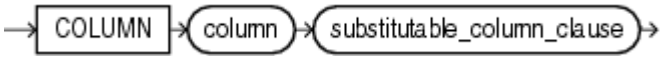


out_of_line_part_storage::=

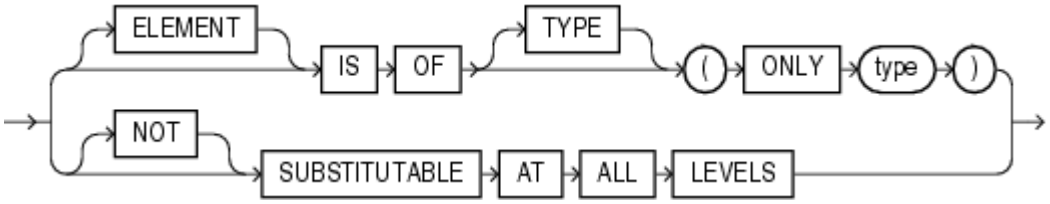




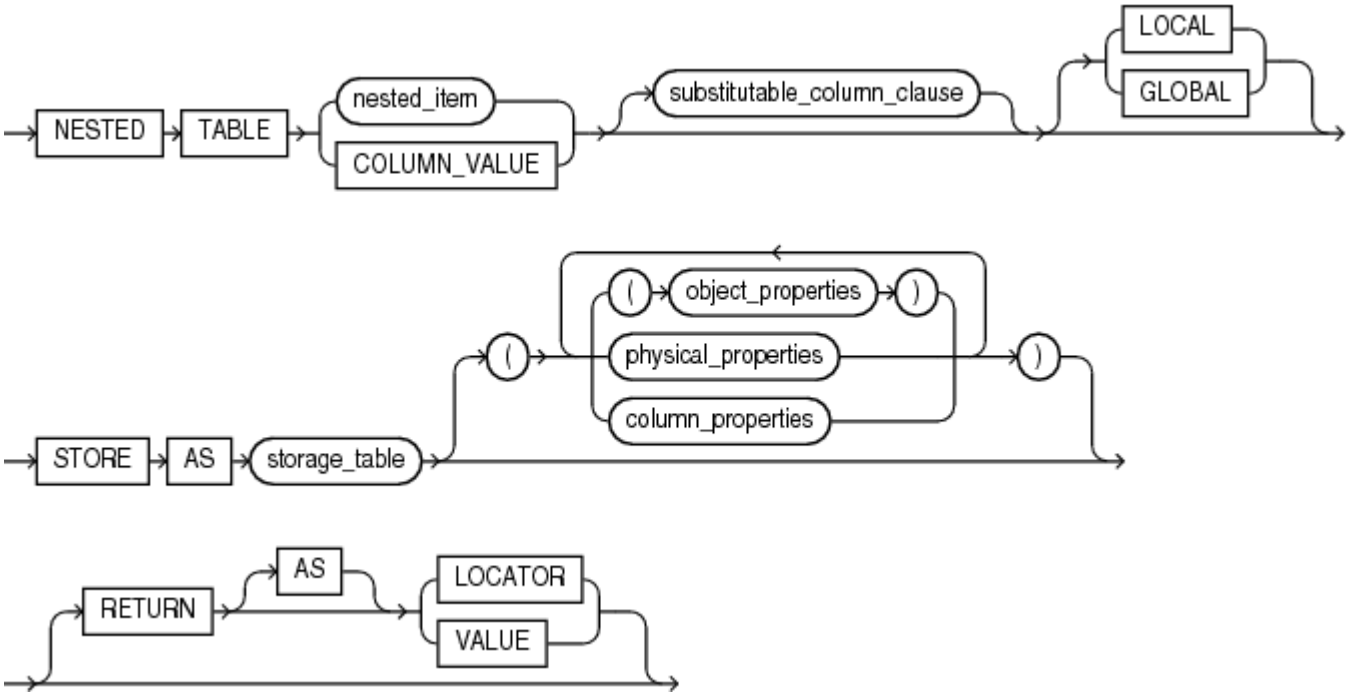
object_type_col_properties::=



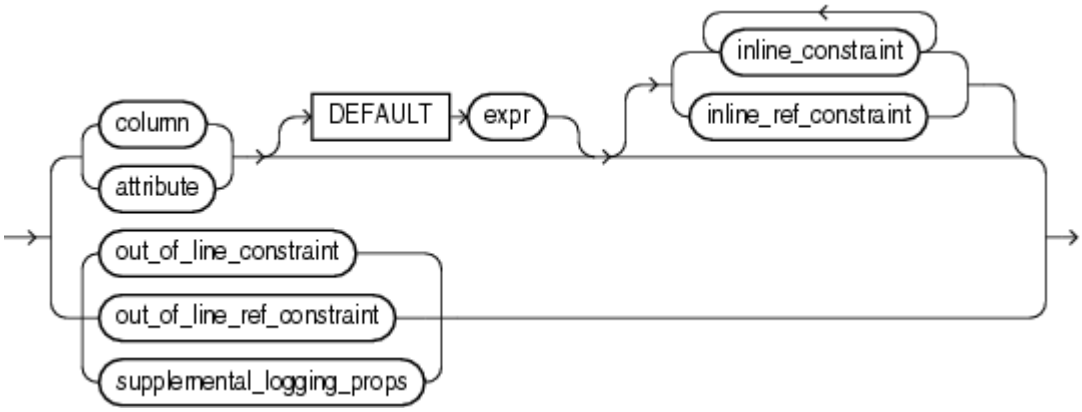
substitutable_column_clause::=



nested_table_col_properties::=

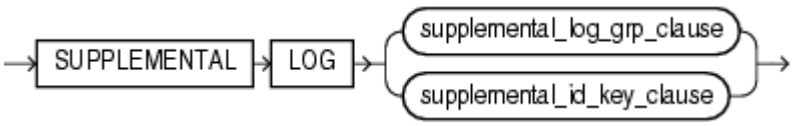


object_properties::=



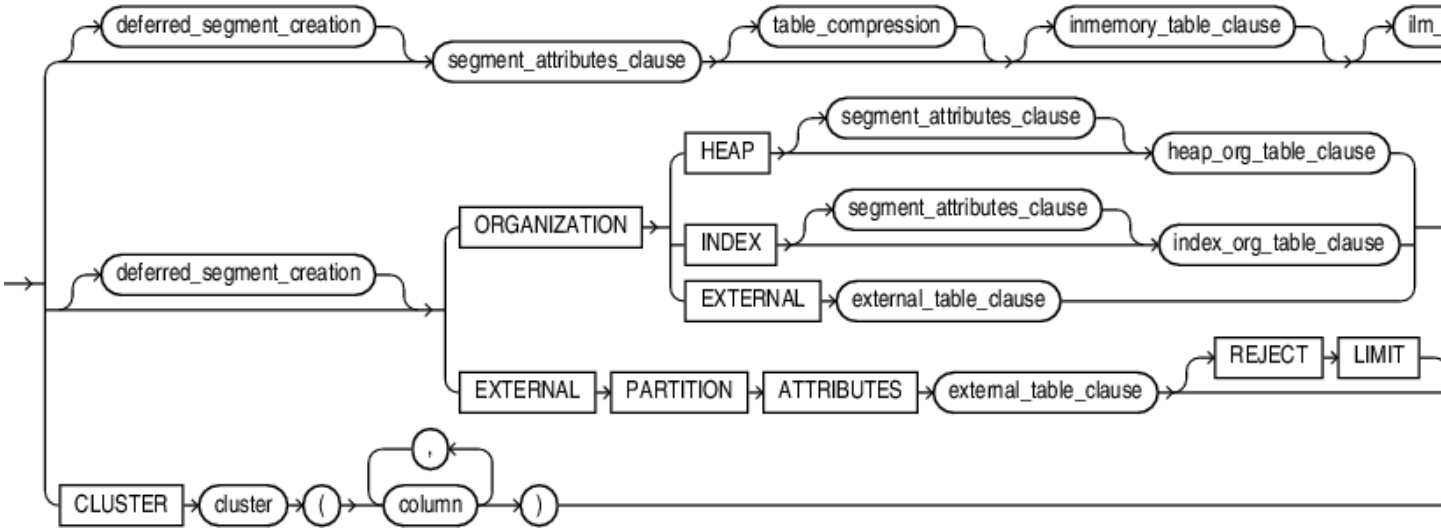
(constraint::=の`inline_constraint`、`inline_ref_constraint`、`out_of_line_constraint`、`out_of_line_ref_constraint`を参照)

supplemental_logging_props::=



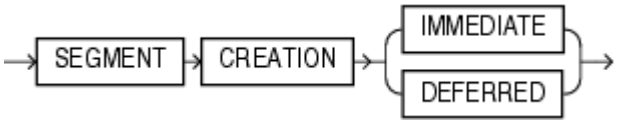
(supplemental_log_grp_clause::=、supplemental_id_key_clause::=)

physical_properties::=

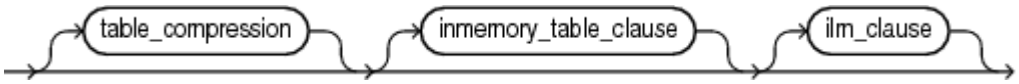


(deferred_segment_creation::=、segment_attributes_clause::=、table_compression::=、inmemory_table_clause::= (CREATE TABLE構文の一部)、ilm_clause::=、heap_org_table_clause::=、index_org_table_clause::=、external_table_clause::= (CREATE TABLE構文の一部))

deferred_segment_creation::=

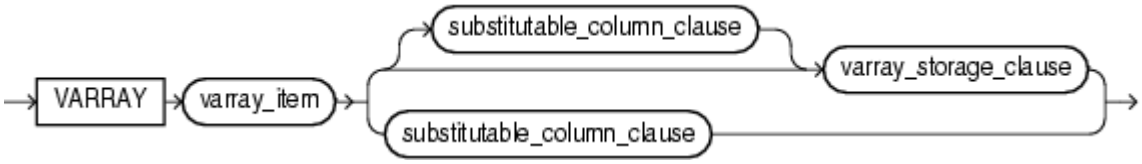


heap_org_table_clause::=



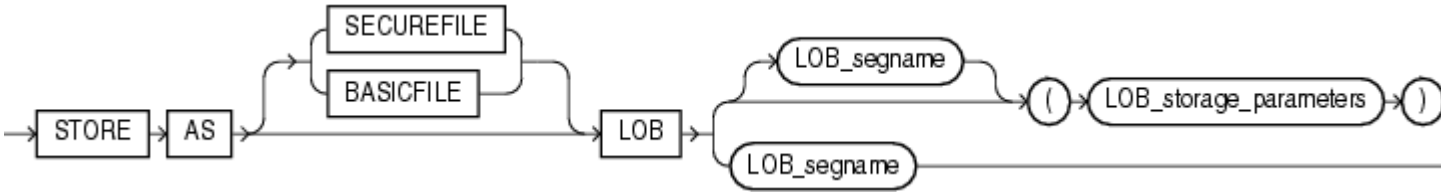
(table_compression::=、inmemory_table_clause::= (CREATE TABLE構文の一部)、ilm_clause::=)

varray_col_properties::=



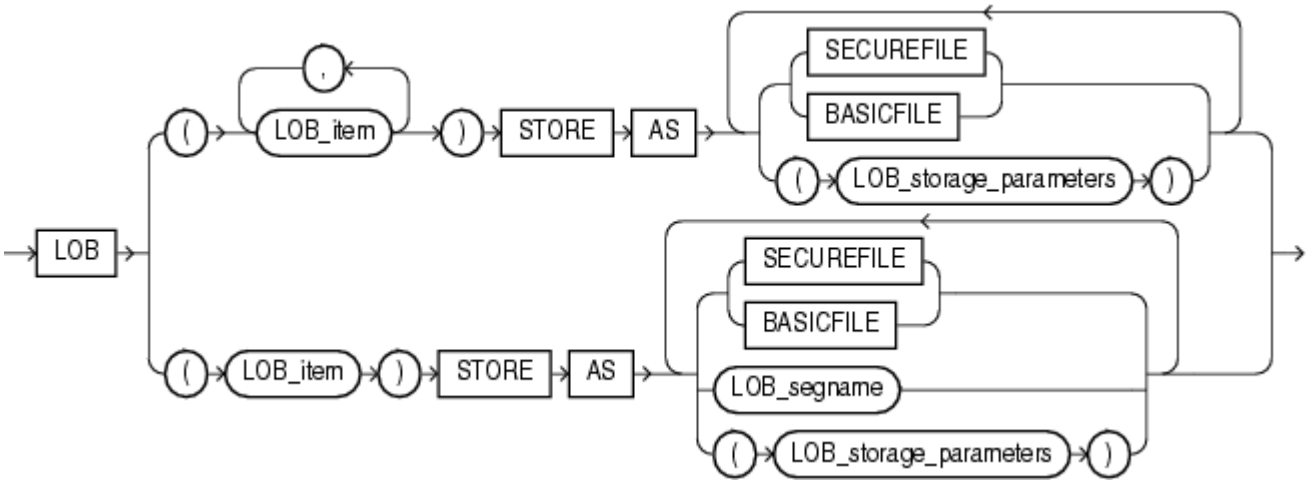
(substitutable_column_clause::=、varray_storage_clause::=)

varray_storage_clause::=



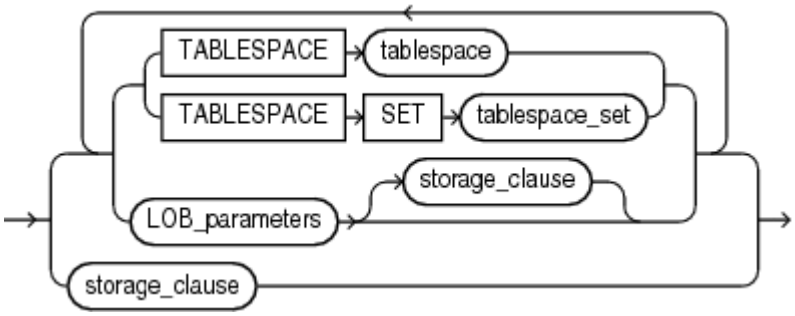
(LOB_parameters::=)

LOB_storage_clause::=



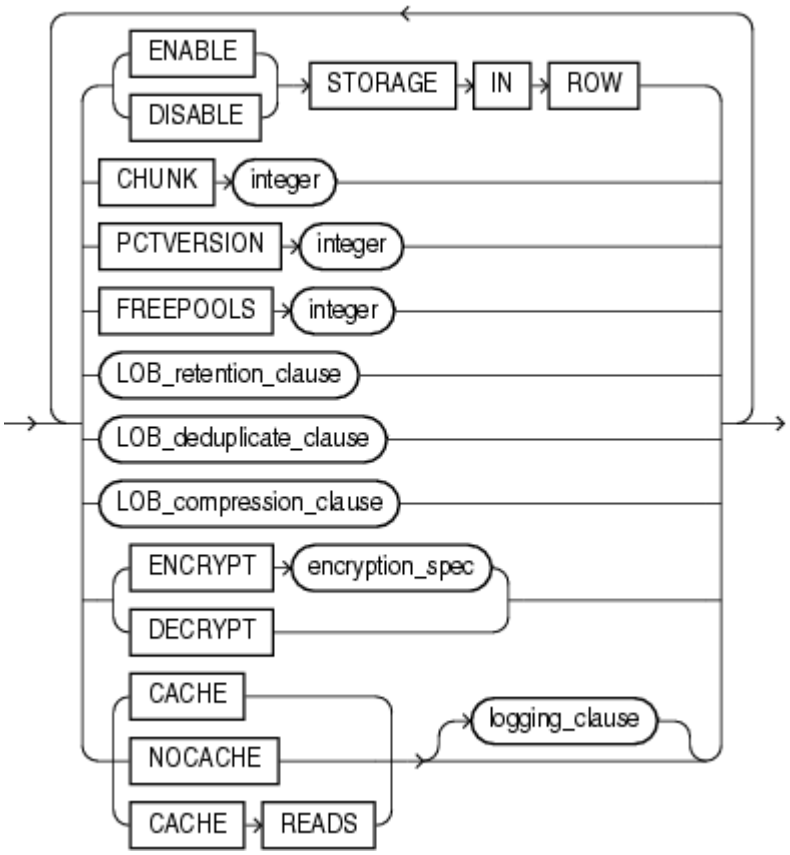
(LOB_storage_parameters::=)

LOB_storage_parameters::=



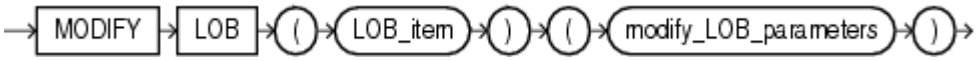
(TABLESPACE SET: ALTER TABLEではサポートされていません、LOB_parameters::=、storage_clause::=)

LOB_parameters::=

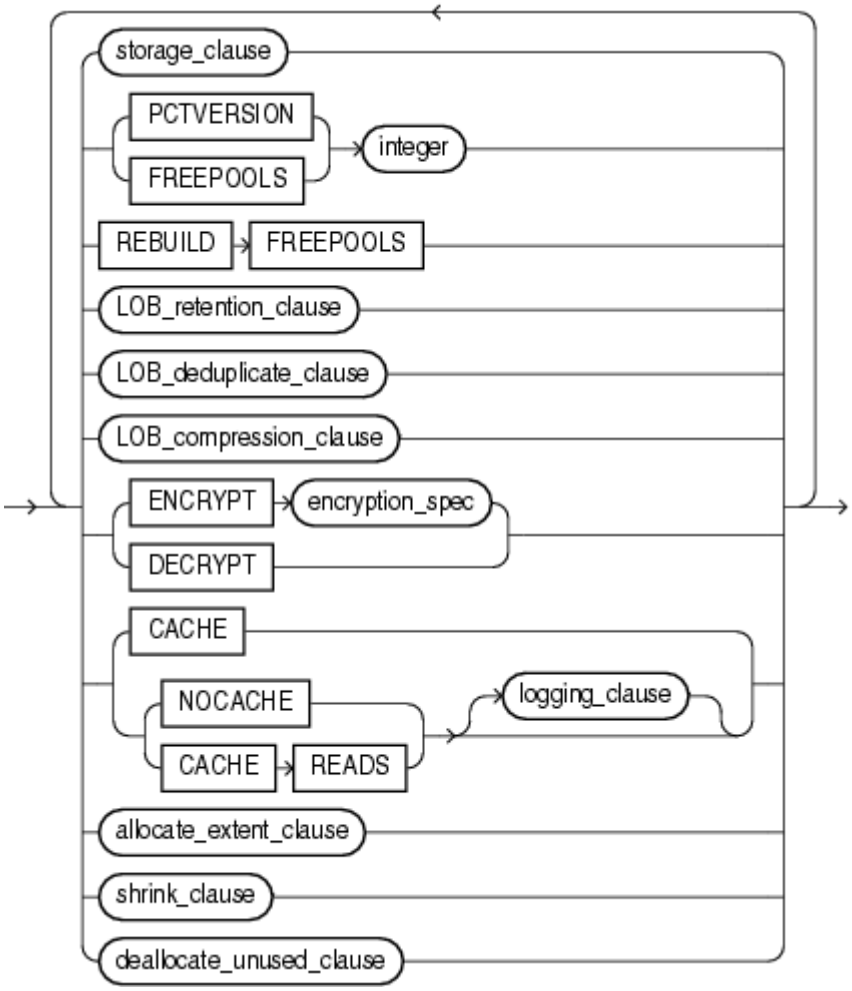


(LOB_retention_clause::=、LOB_deduplicate_clause::=、LOB_compression_clause::=、encryption_spec::=、logging_clause::=)

modify_LOB_storage_clause::=

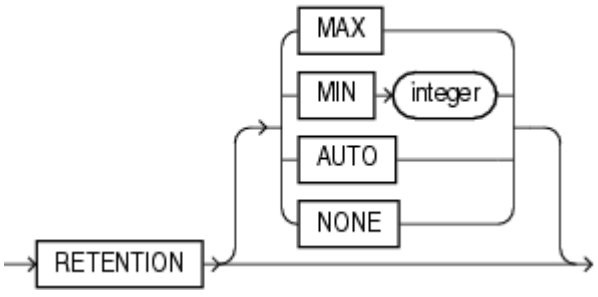


modify_LOB_parameters::=

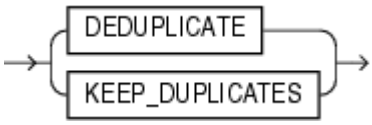


(storage_clause::=、LOB_retention_clause::=、LOB_compression_clause::=、encryption_spec::=、logging_clause::=、allocate_extent_clause::=、shrink_clause::=、deallocate_unused_clause::=)

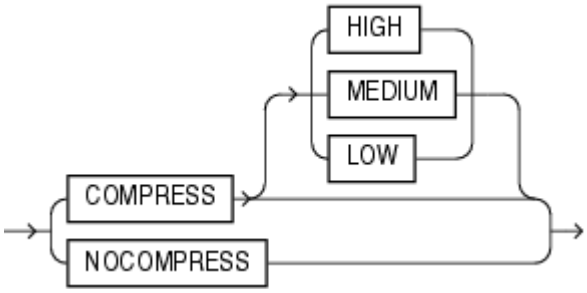
LOB_retention_clause::=



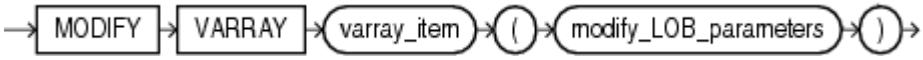
LOB_deduplicate_clause::=



LOB_compression_clause::=

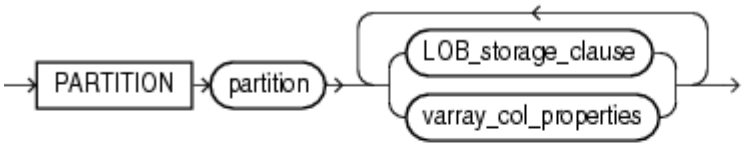


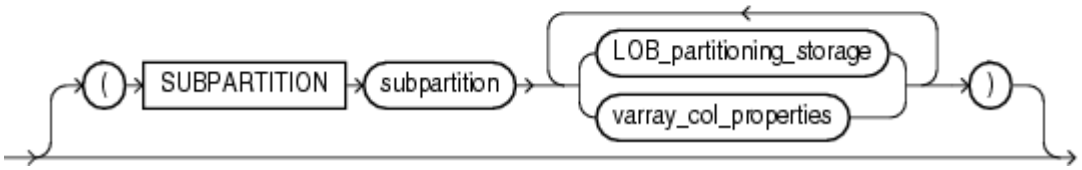
alter_varray_col_properties::=



(modify_LOB_parameters::=)

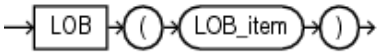
LOB_partition_storage::=

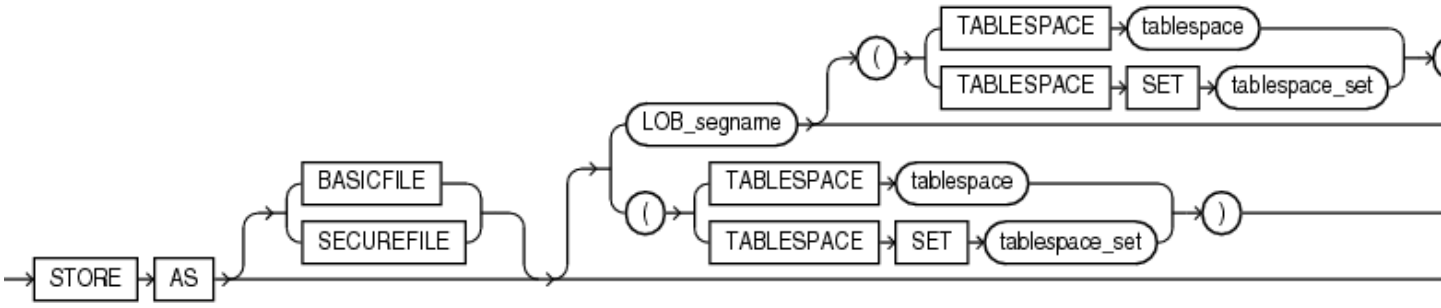




(LOB_storage_clause::=、varray_col_properties::=、LOB_partitioning_storage::=)

LOB_partitioning_storage::=

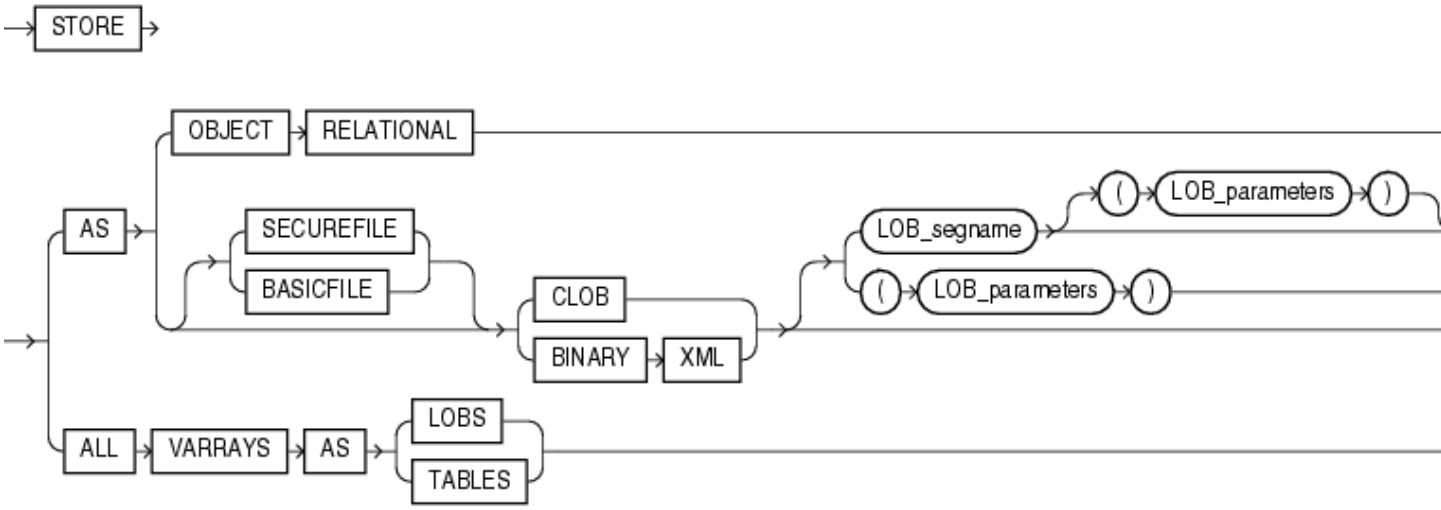




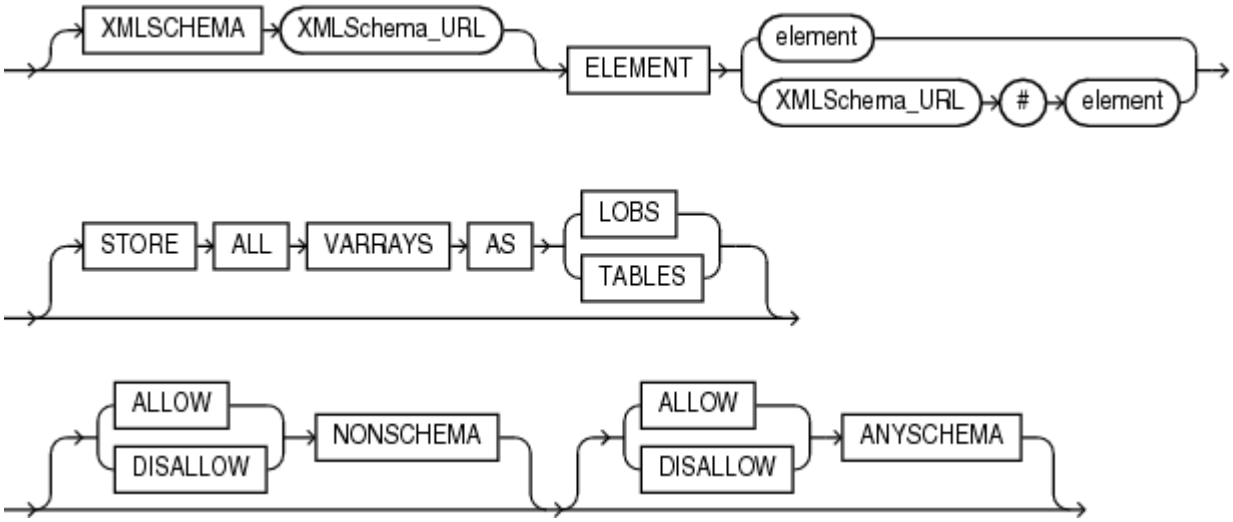
(TABLESPACE SET: ALTER TABLEではサポートされていません)

XMLType_column_properties::=

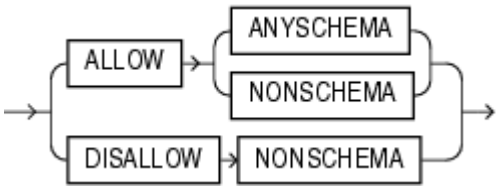
XMLType_storage::=



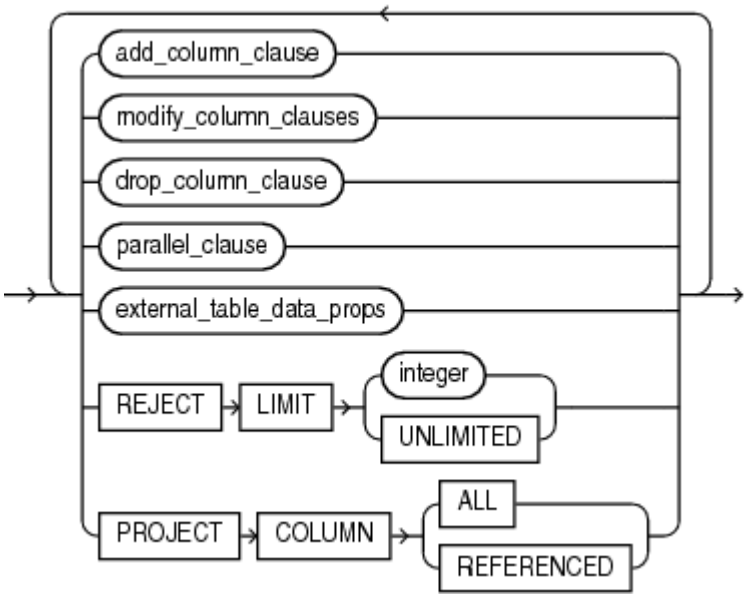
XMLSchema_spec::=



alter_XMLSchema_clause::=

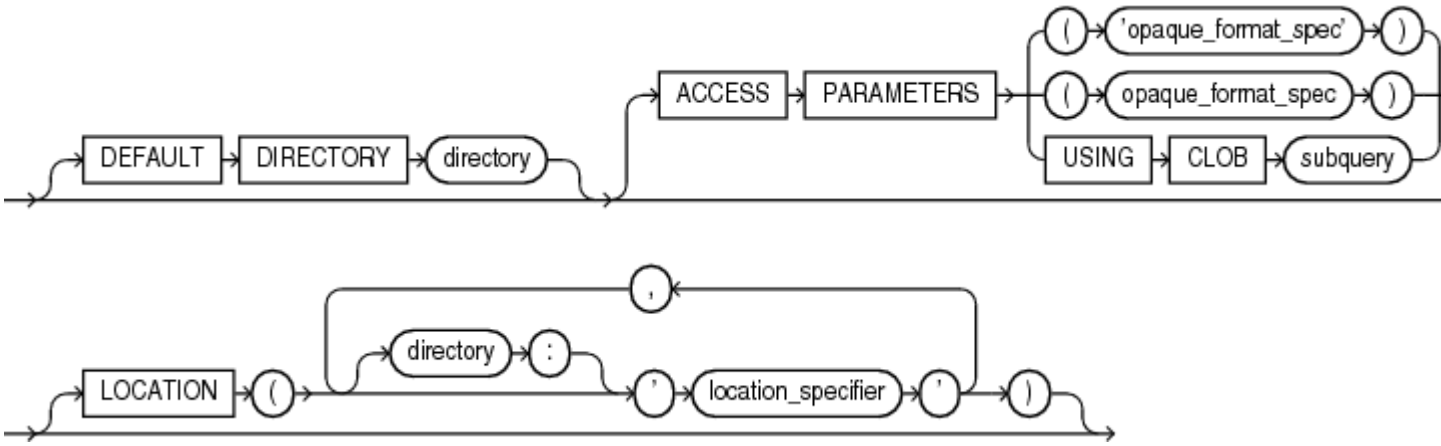


alter_external_table::=

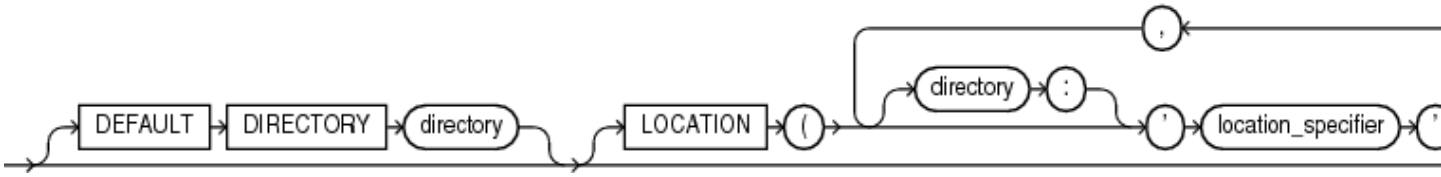


(add_column_clause::=、modify_column_clauses::=、drop_column_clause::=、parallel_clause::=、external_table_data_props::=)

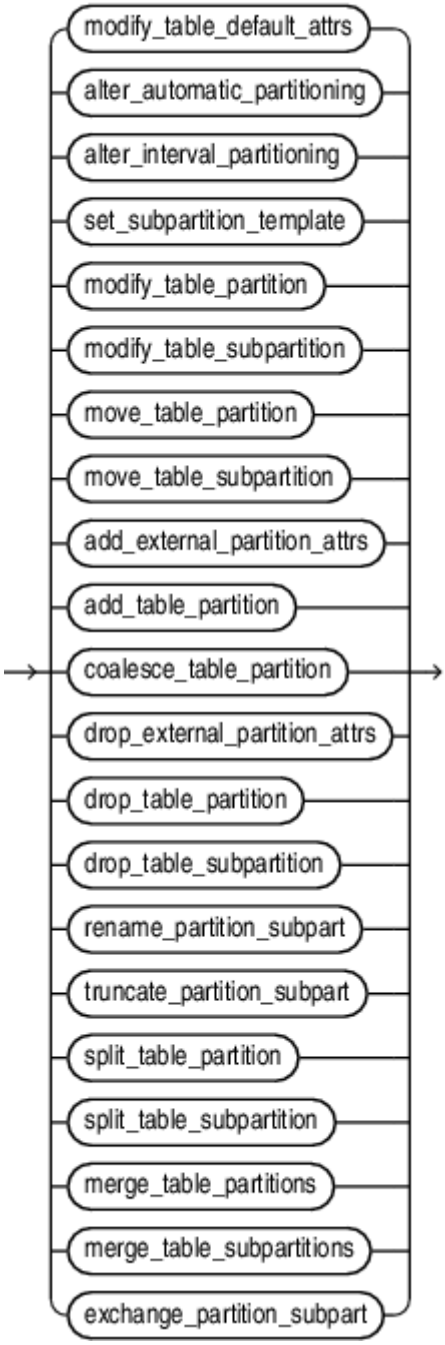
external_table_data_props::=



external_part_subpart_data_props::=

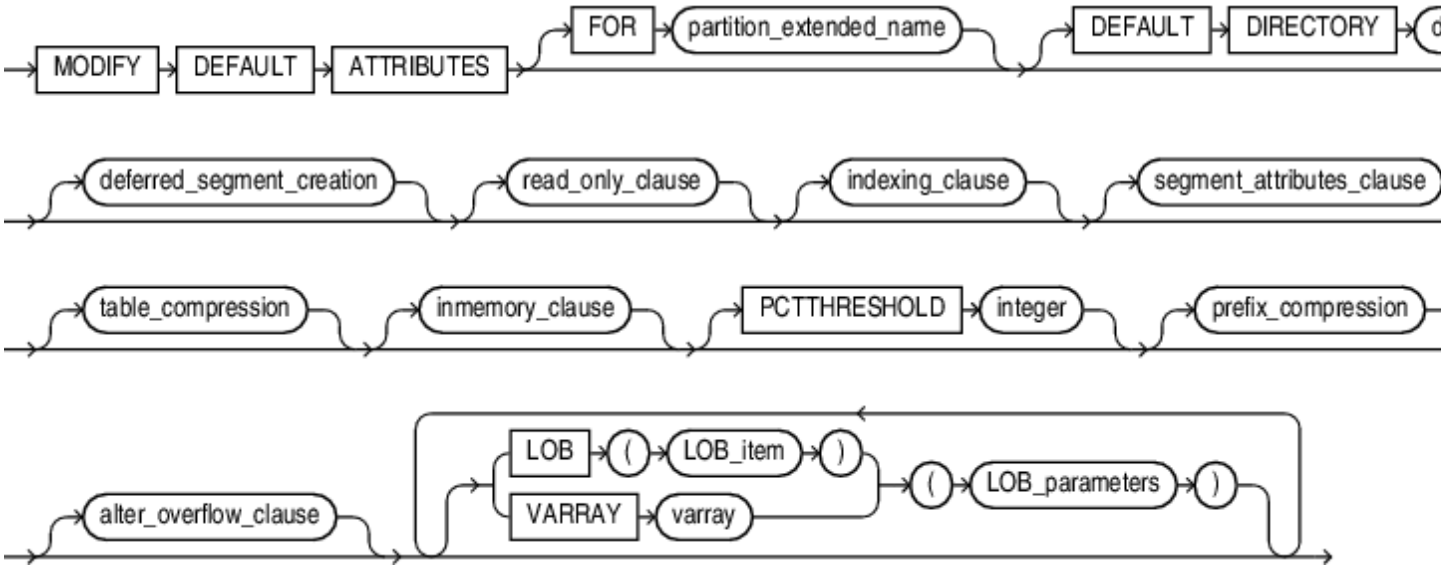


alter_table_partitioning::=



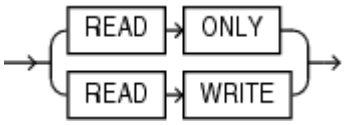
(modify_table_default_attrs::=、alter_automatic_partitioning::=、alter_interval_partitioning::=、set_subpartition_template::=、modify_table_partition::=、modify_table_subpartition::=、move_table_partition::=、move_table_subpartition::=、add_table_partition::=、coalesce_table_partition::=、drop_table_partition::=、drop_table_subpartition::=、rename_partition_subpart::=、truncate_partition_subpart::=、split_table_partition::=、split_table_subpartition::=、merge_table_partitions::=、merge_table_subpartitions::=、exchange_partition_subpart::=)

modify_table_default_attrs::=

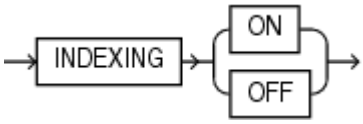


(partition_extended_name::=、deferred_segment_creation::=、read_only_clause::=、indexing_clause::=、segment_attributes_clause::=、table_compression::=、inmemory_clause::=、prefix_compression::=、alter_overflow_clause::=、LOB_parameters::=)

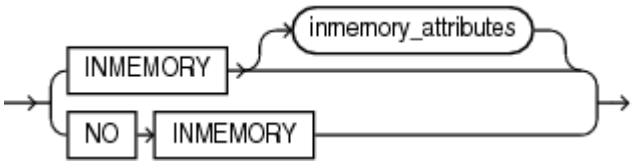
read_only_clause::=



indexing_clause::=

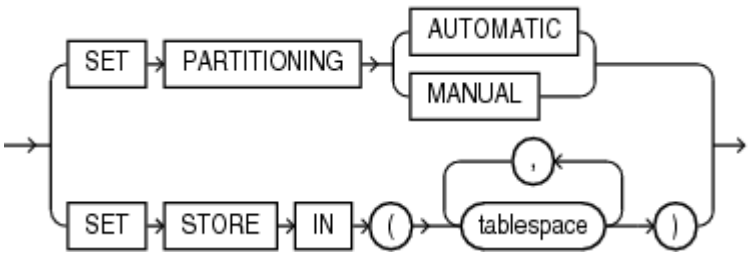


inmemory_clause::=

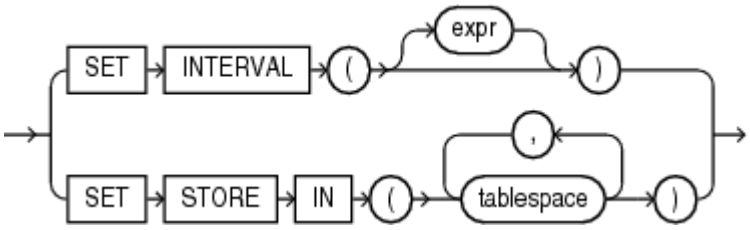


(inmemory_attributes::=)

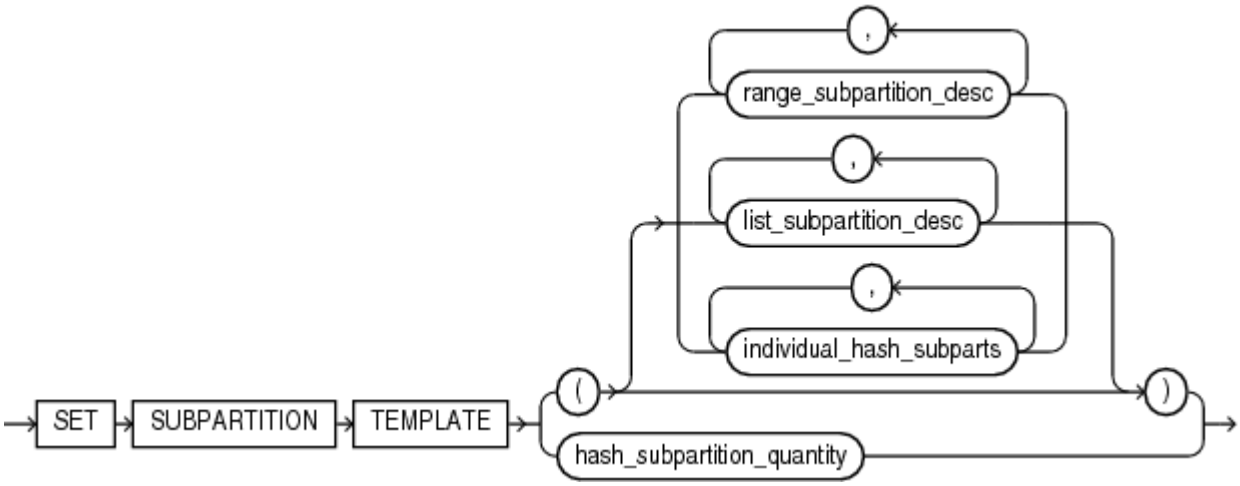
alter_automatic_partitioning::=



alter_interval_partitioning::=

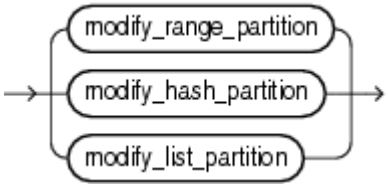


set_subpartition_template::=



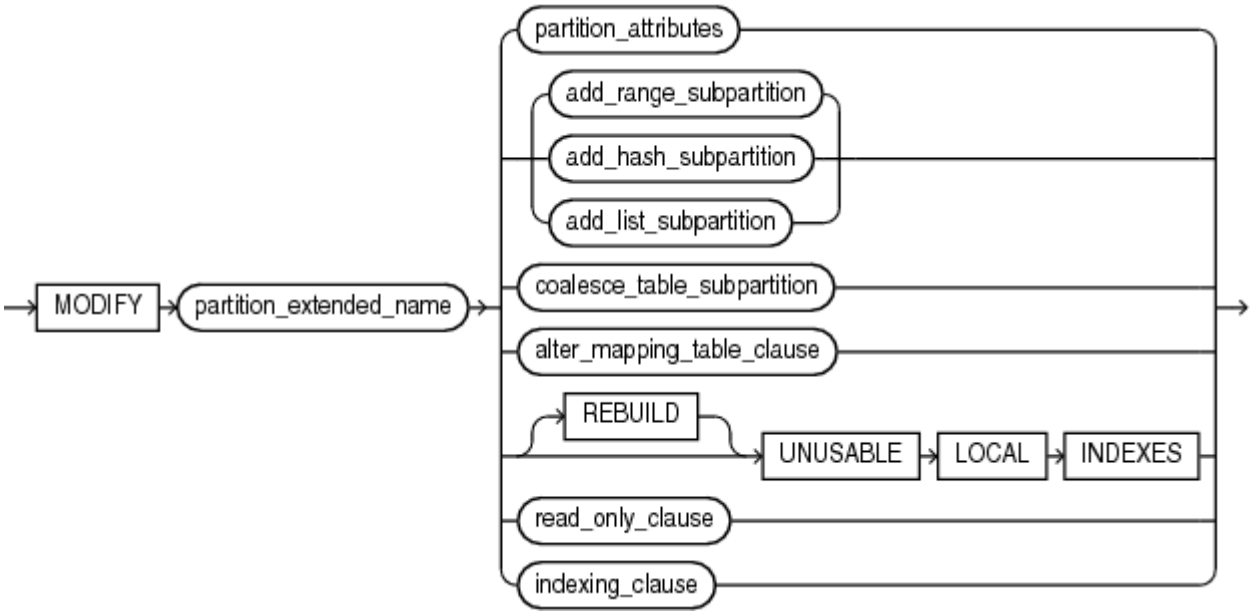
(range_subpartition_desc::=、list_subpartition_desc::=、individual_hash_subparts::=)

modify_table_partition::=



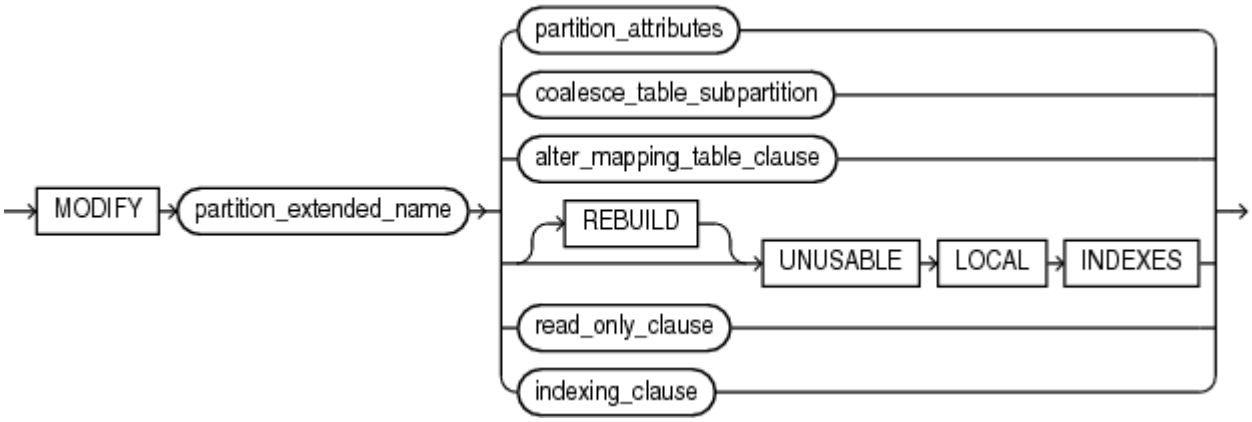
(modify_range_partition::=、modify_hash_partition::=、modify_list_partition::=)

modify_range_partition::=



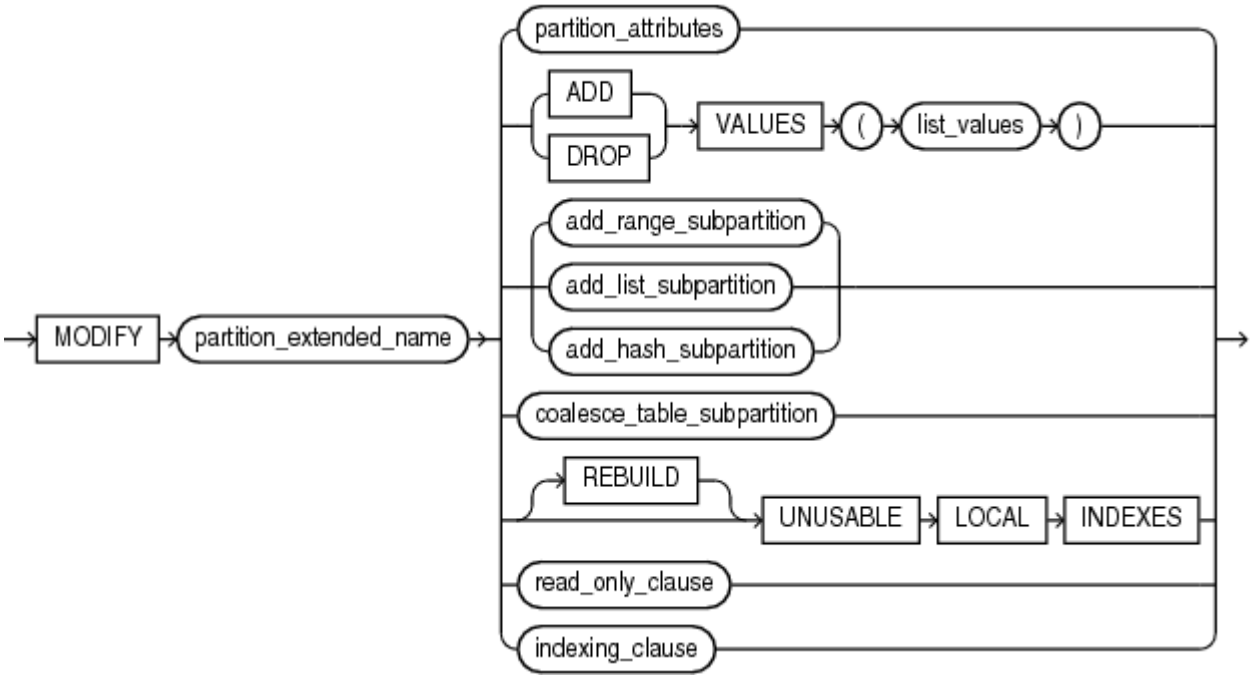
(partition_extended_name::=、partition_attributes::=、add_range_subpartition::=、
add_hash_subpartition::=、add_list_subpartition::=、coalesce_table_subpartition::=、
alter_mapping_table_clauses::=、read_only_clause::=、indexing_clause::=)

modify_hash_partition::=



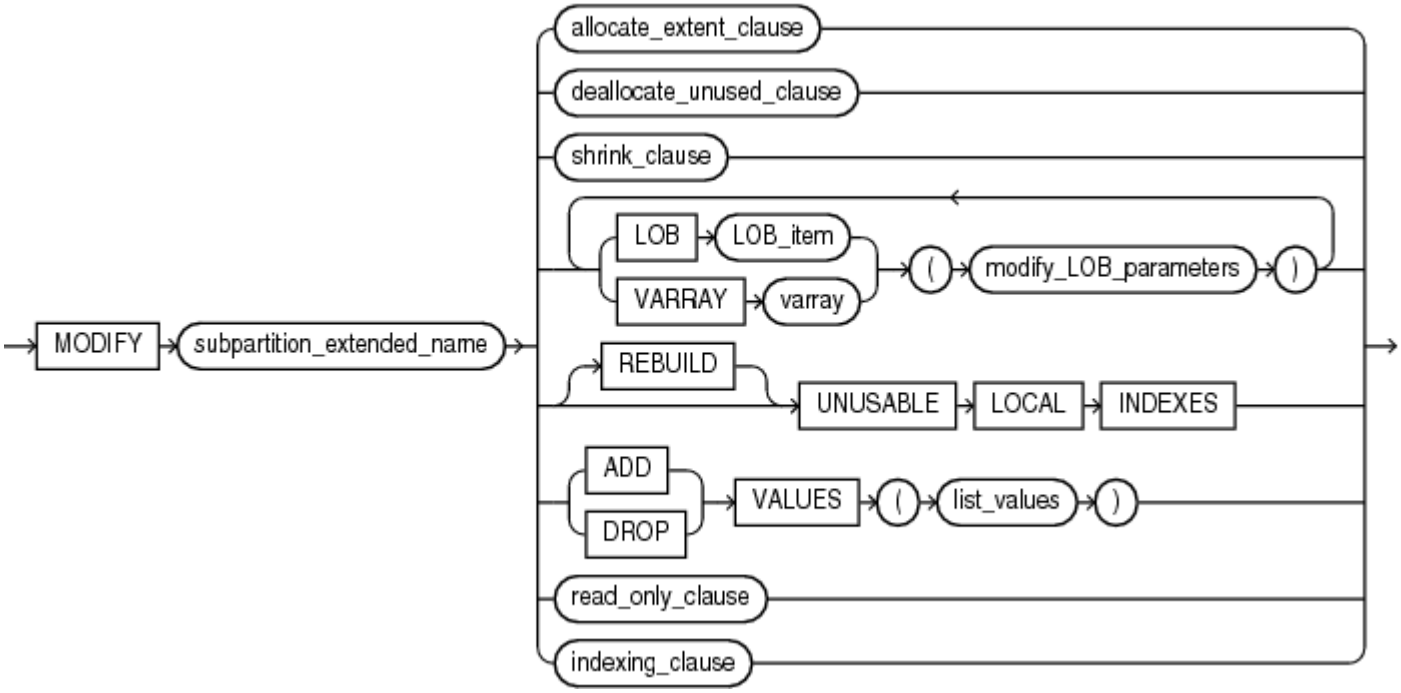
(partition_extended_name::=、coalesce_table_subpartition::=、partition_attributes::=、
alter_mapping_table_clauses::=、read_only_clause::=、indexing_clause::=)

modify_list_partition::=



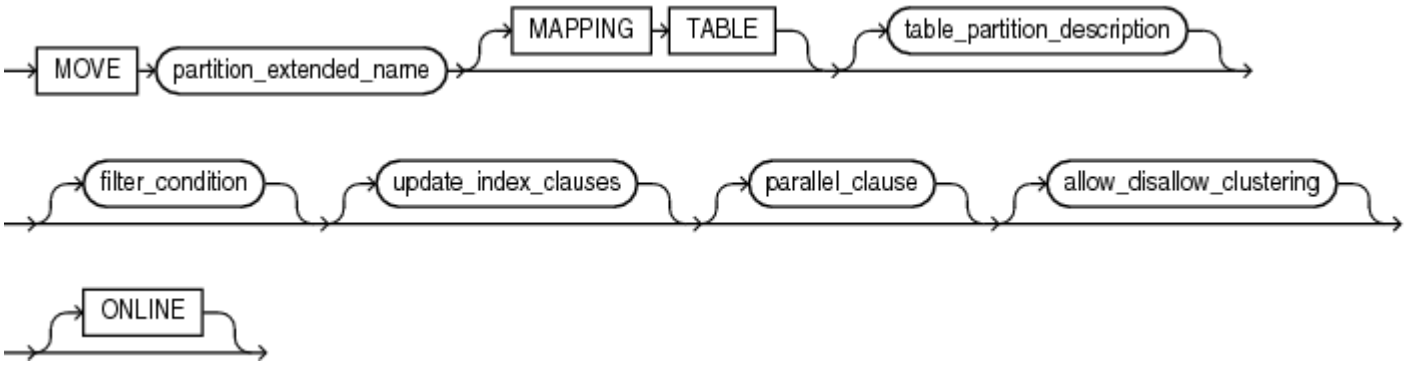
(partition_extended_name::=、partition_attributes::=、list_values::=、add_range_subpartition::=、
add_list_subpartition::=、add_hash_subpartition::=、coalesce_table_subpartition::=、
read_only_clause::=、indexing_clause::=)

modify_table_subpartition::=



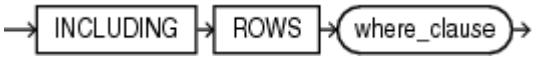
(subpartition_extended_name::=、allocate_extent_clause::=、deallocate_unused_clause::=、shrink_clause::=、modify_LOB_parameters::=、list_values::=、read_only_clause::=、indexing_clause::=)

move_table_partition::=

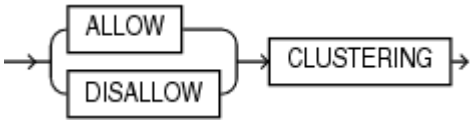


(partition_extended_name::=、table_partition_description::=、filter_condition::=、update_index_clauses::=、parallel_clause::=、allow_disallow_clustering::=)

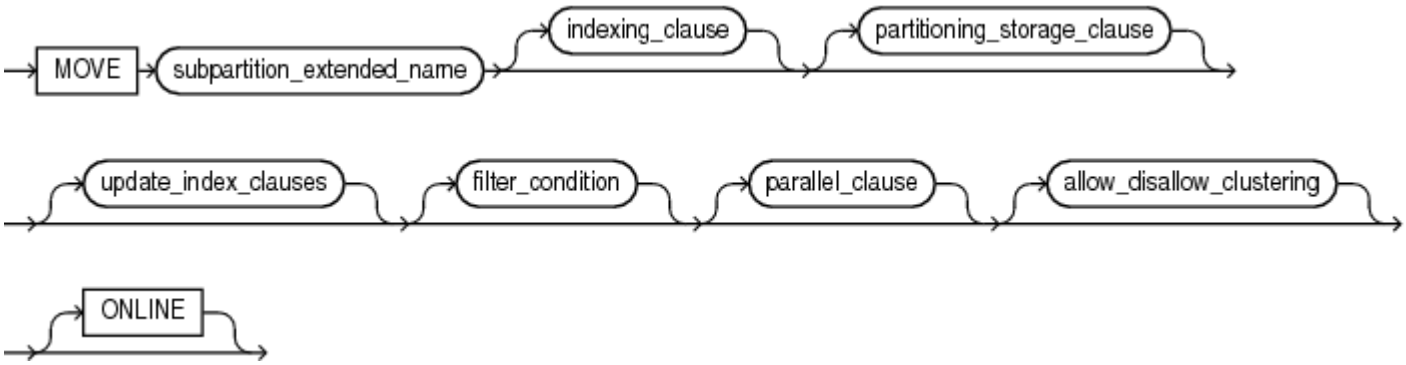
filter_condition::=



allow_disallow_clustering::=

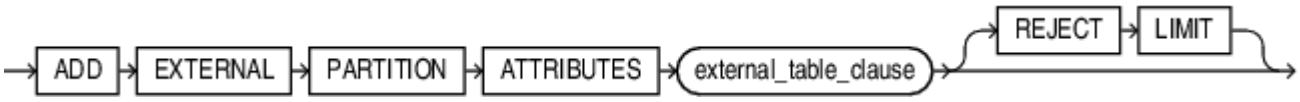


move_table_subpartition::=

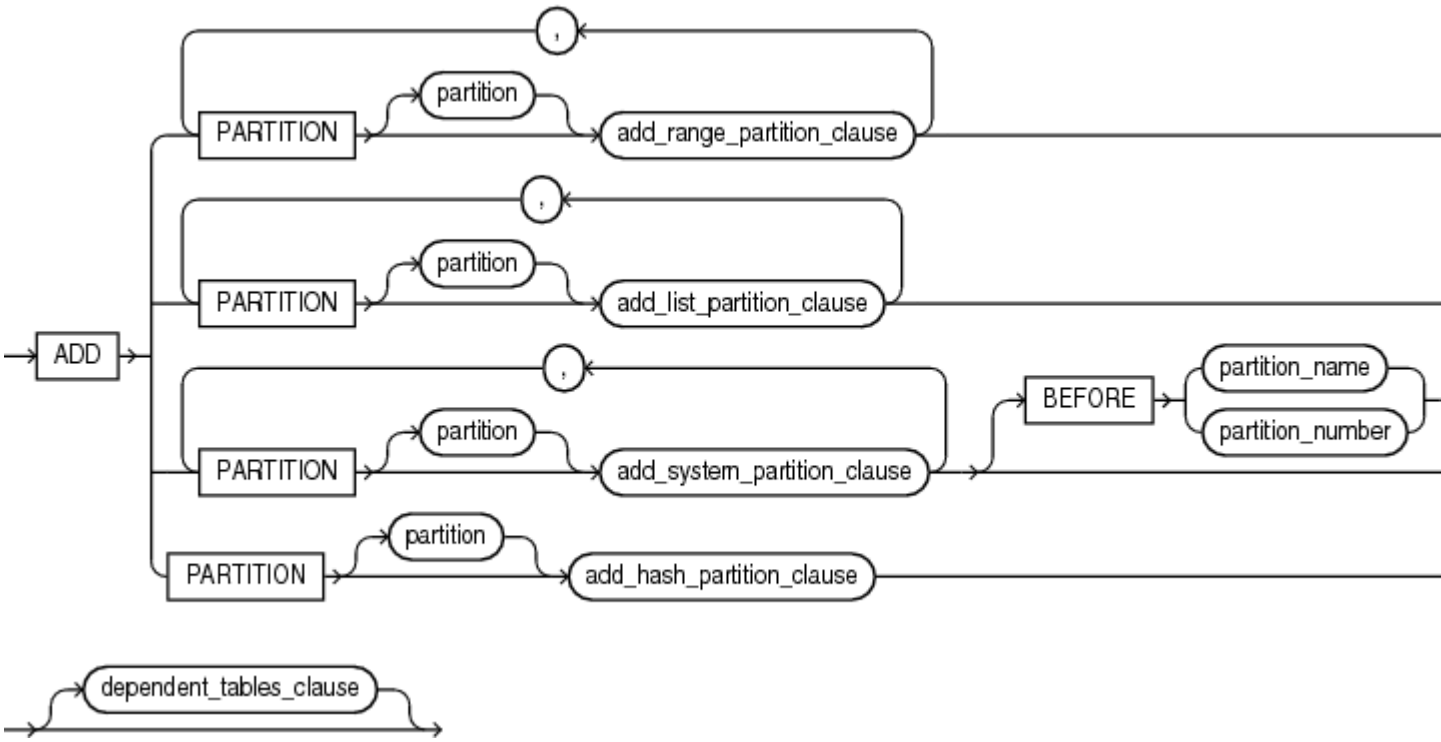


(subpartition_extended_name::=、indexing_clause::=、partitioning_storage_clause::=、update_index_clauses::=、filter_condition::=、parallel_clause::=、allow_disallow_clustering::=)

add_external_partition_attrs

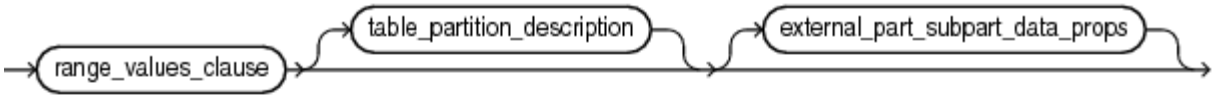


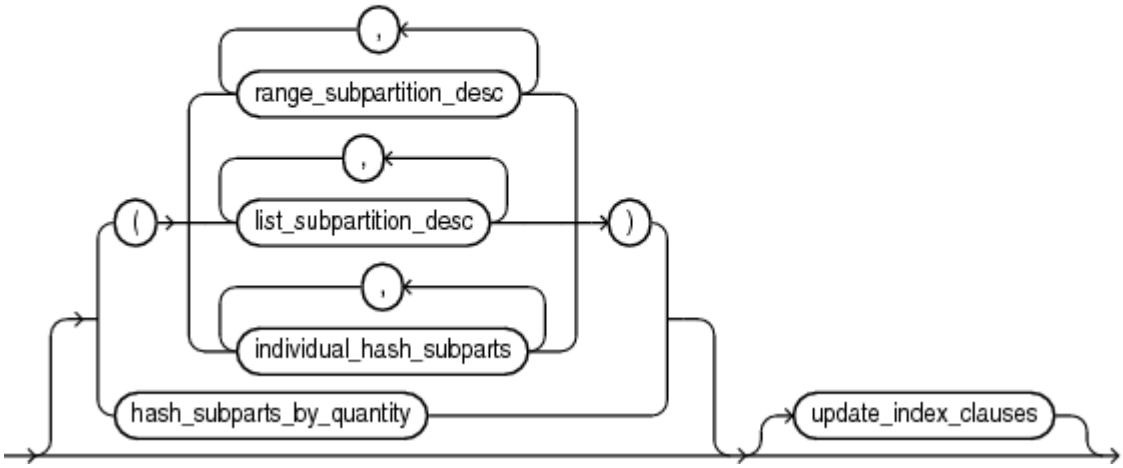
add_table_partition::=



(add_range_partition_clause::=、add_list_partition_clause::=、add_system_partition_clause::=、add_hash_partition_clause::=、dependent_tables_clause:=)

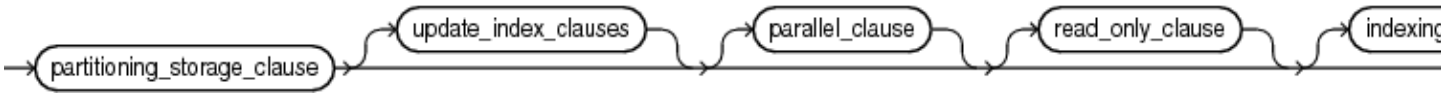
add_range_partition_clause::=





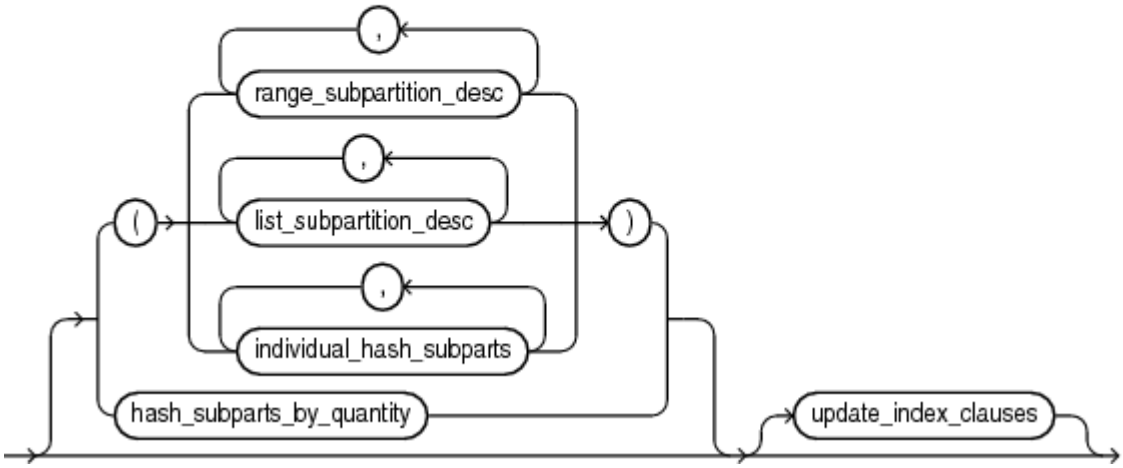
(range_values_clause::=、table_partition_description::=、external_part_subpart_data_props::=、range_subpartition_desc::=、list_subpartition_desc::=、individual_hash_subparts::=、hash_subparts_by_quantity::=、update_index_clauses::=)

add_hash_partition_clause::=



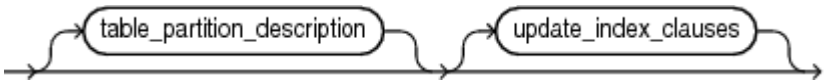
(partitioning_storage_clause::=、update_index_clauses::=、parallel_clause::=、read_only_clause::=、indexing_clause::=)

add_list_partition_clause::=



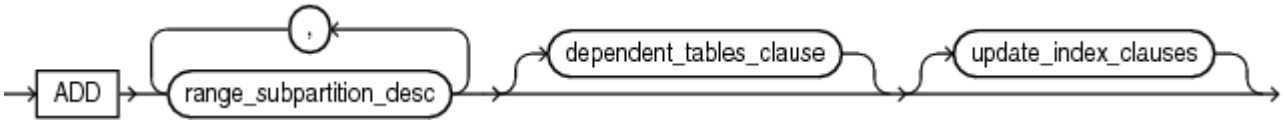
(list_values_clause::=、table_partition_description::=、external_part_subpart_data_props::=、range_subpartition_desc::=、list_subpartition_desc::=、individual_hash_subparts::=、hash_subparts_by_quantity::=、update_index_clauses::=)

add_system_partition_clause::=



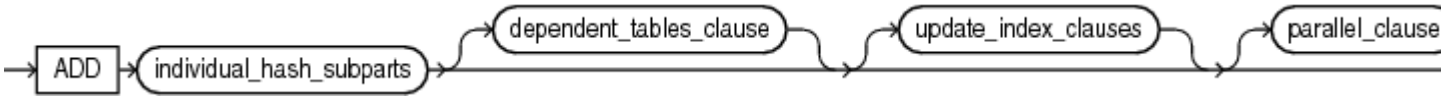
(table_partition_description::=、update_index_clauses::=)

add_range_subpartition::=



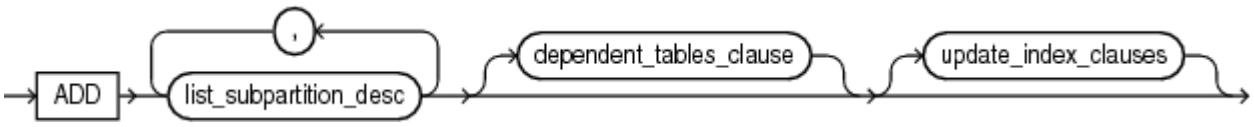
(range_subpartition_desc::=、dependent_tables_clause:=、update_index_clauses::=)

add_hash_subpartition::=



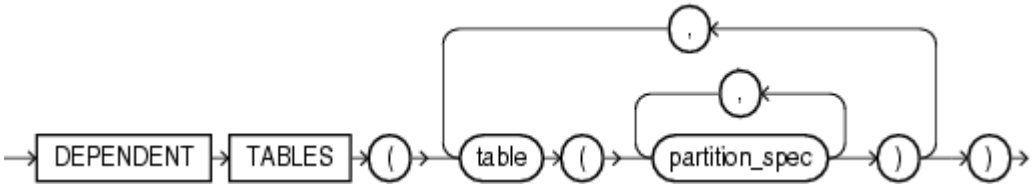
(individual_hash_subparts::=、dependent_tables_clause:=、update_index_clauses::=、parallel_clause::=)

add_list_subpartition::=



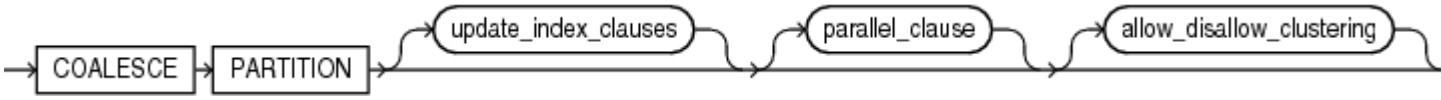
(list_subpartition_desc::=、dependent_tables_clause:=、update_index_clauses::=)

dependent_tables_clause:=



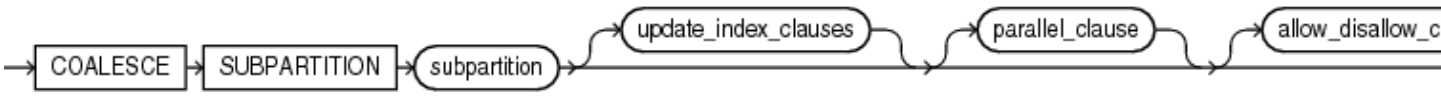
(partition_spec::=)

coalesce_table_partition::=



(update_index_clauses::=、parallel_clause::=、allow_disallow_clustering::=)

coalesce_table_subpartition::=



(update_index_clauses::=、parallel_clause::=、allow_disallow_clustering::=)

drop_external_partition_attrs::=

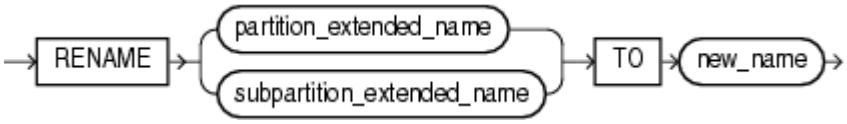
drop_table_partition::=

(partition_extended_names::=、update_index_clauses::=、parallel_clause::=)

drop_table_subpartition::=

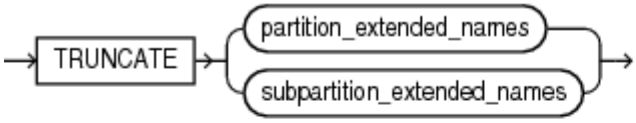
(subpartition_extended_names::=、update_index_clauses::=、parallel_clause::=)

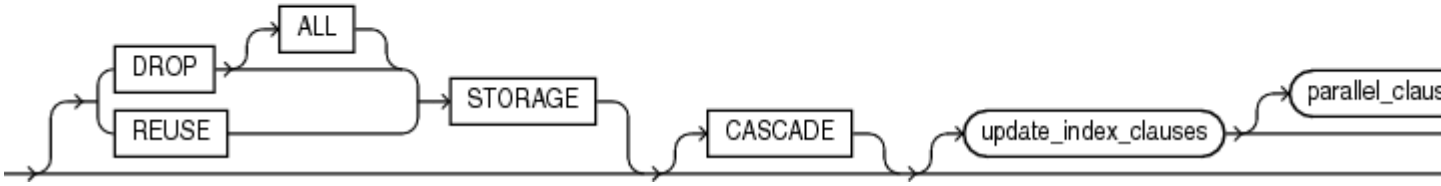
rename_partition_subpart::=



(partition_extended_name::=、subpartition_extended_name::=)

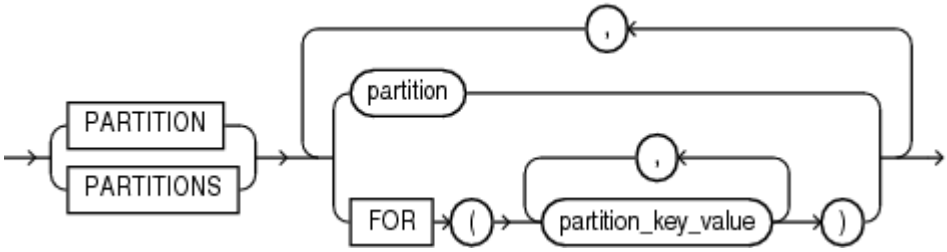
truncate_partition_subpart::=



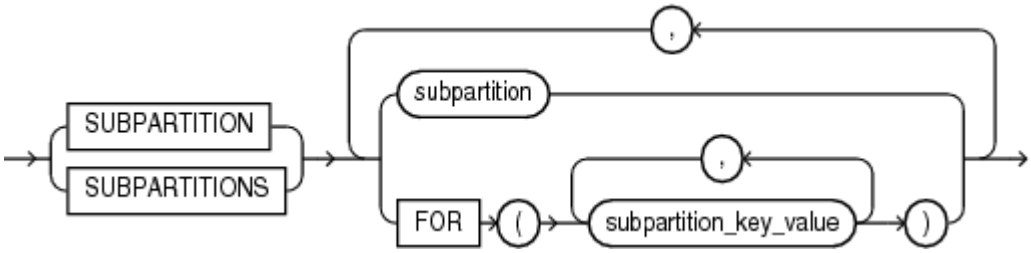


(partition_extended_names::=、subpartition_extended_names::=、update_index_clauses::=、parallel_clause::=)

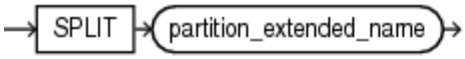
partition_extended_names::=

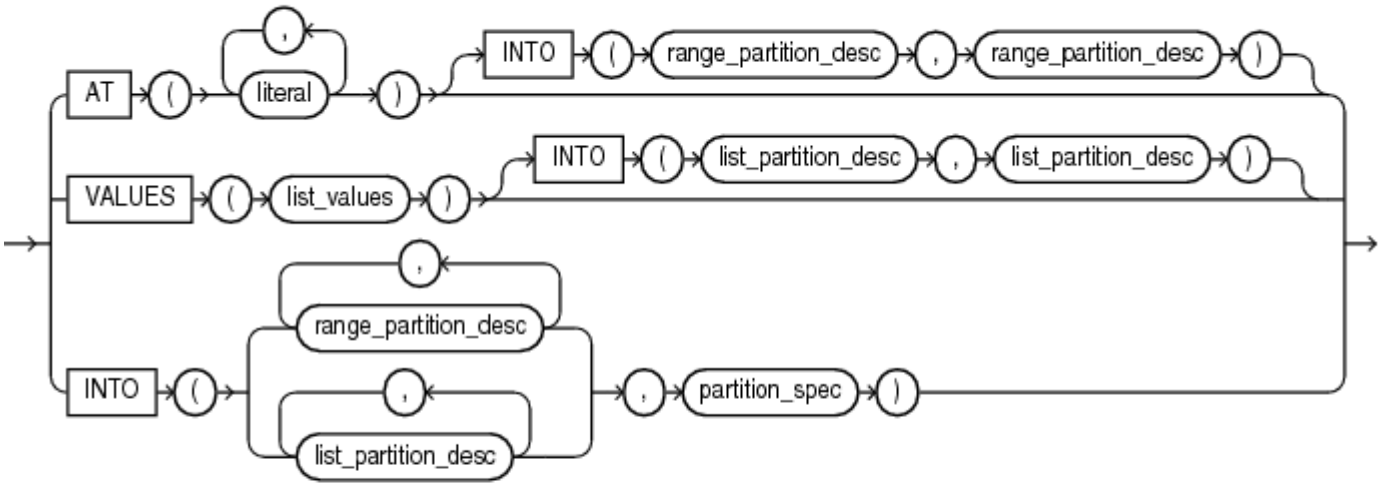


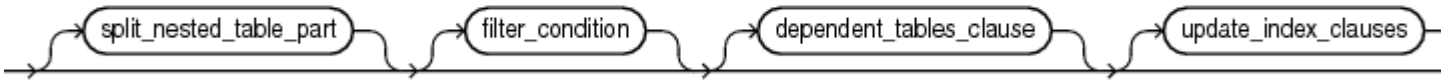
subpartition_extended_names::=

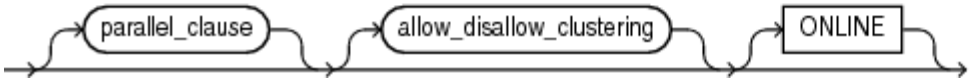


split_table_partition::=



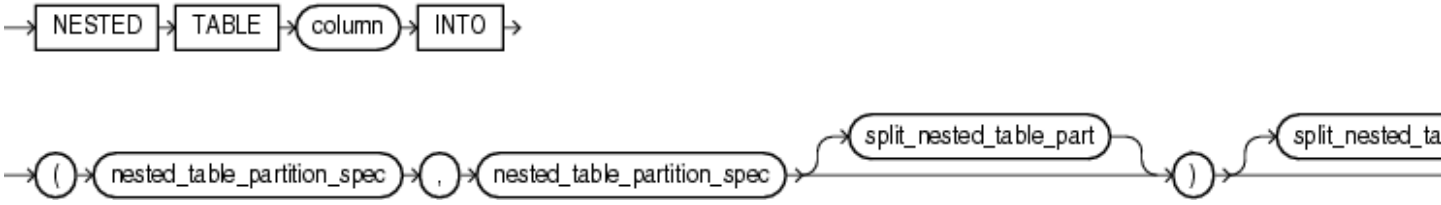






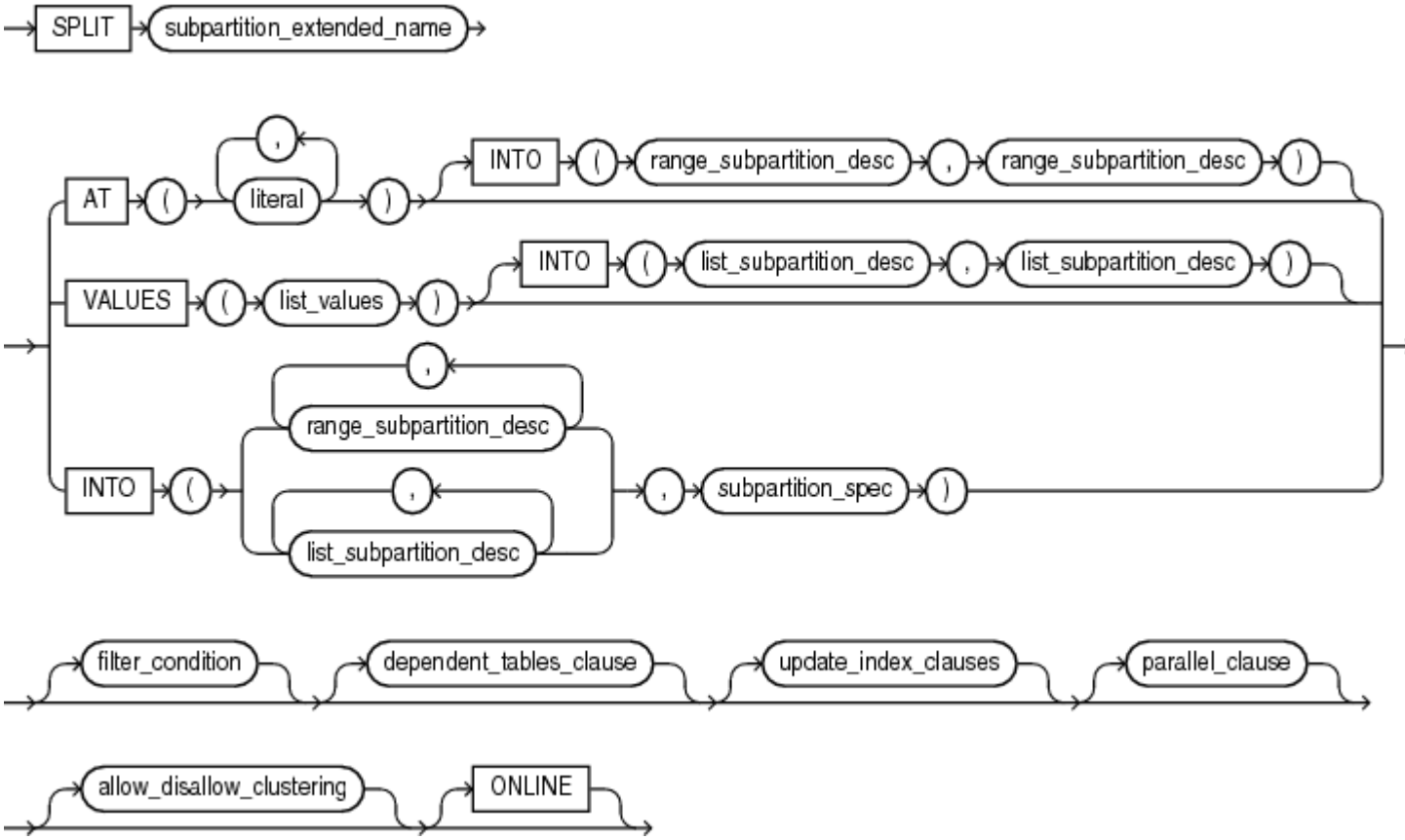
(partition_extended_name::=、range_partition_desc::=、list_values::=、list_partition_desc::=、partition_spec::=、split_nested_table_part::=、filter_condition::=、dependent_tables_clause:=、update_index_clauses::=、parallel_clause::=、allow_disallow_clustering::=)

split_nested_table_part::=



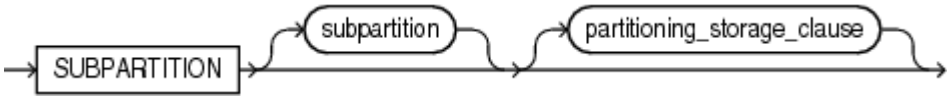
nested_table_partition_spec::=

split_table_subpartition::=

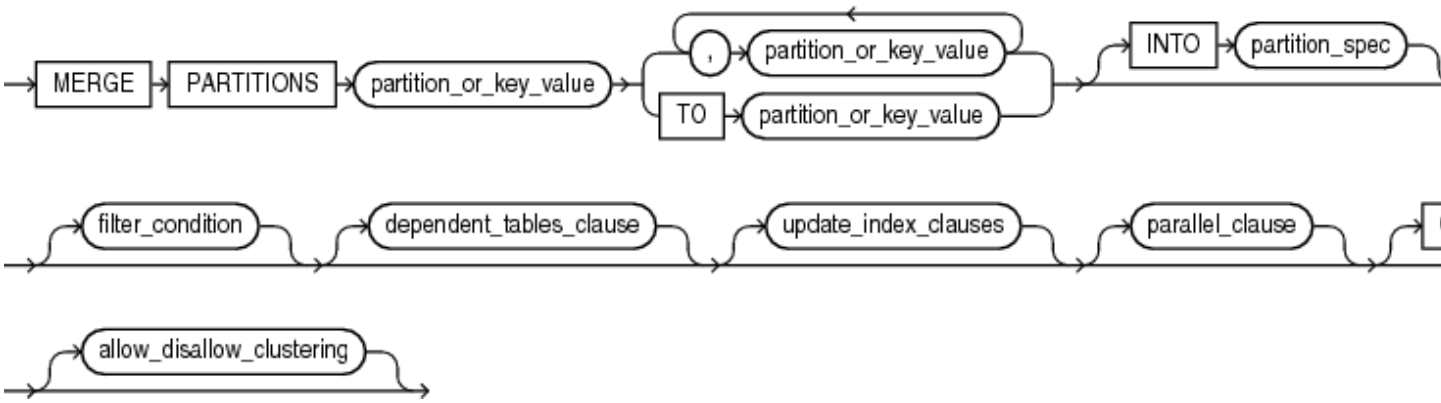


(subpartition_extended_name::=、range_subpartition_desc::=、list_values::=、list_subpartition_desc::=、subpartition_spec::=、filter_condition::=、dependent_tables_clause:=、update_index_clauses::=、parallel_clause::=、allow_disallow_clustering::=)

subpartition_spec::=

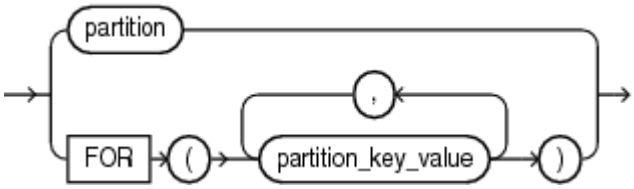


merge_table_partitions::=

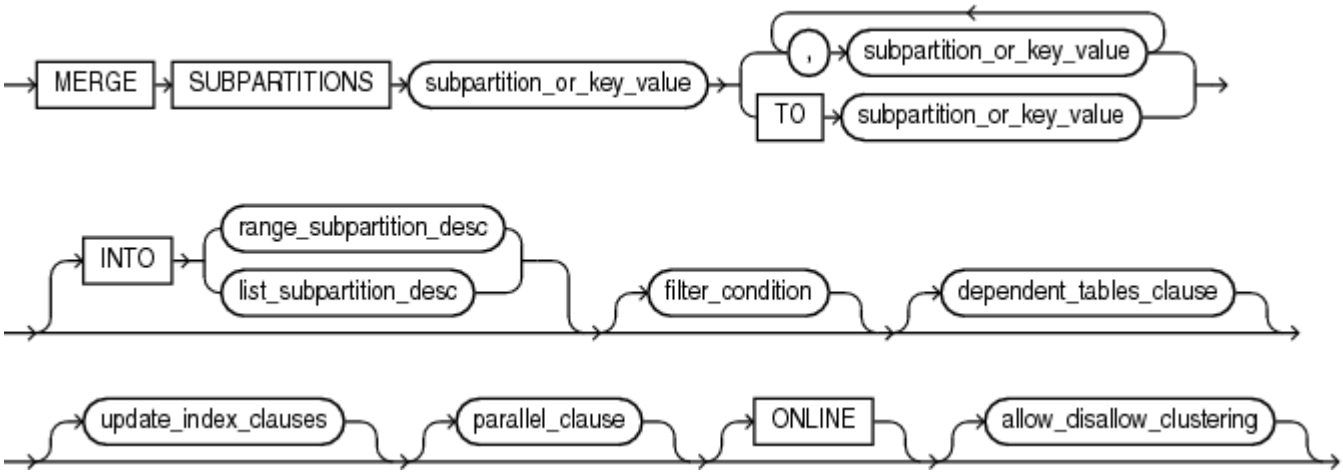


(partition_or_key_value::=、partition_spec::=、filter_condition::=、dependent_tables_clause:=、update_index_clauses::=、parallel_clause::=、allow_disallow_clustering::=)

partition_or_key_value::=

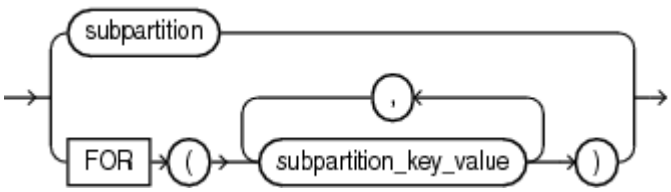


merge_table_subpartitions::=

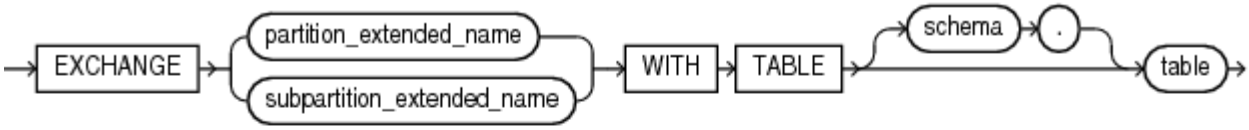


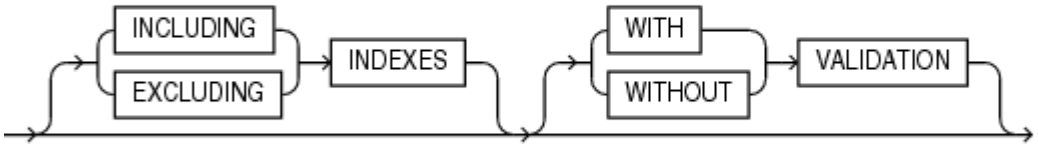
(subpartition_or_key_value::=、range_subpartition_desc::=、list_subpartition_desc::=、filter_condition::=、dependent_tables_clause:=、update_index_clauses::=、parallel_clause::=、allow_disallow_clustering::=)

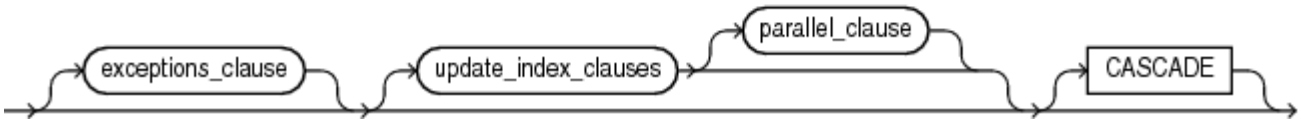
subpartition_or_key_value::=



exchange_partition_subpart::=

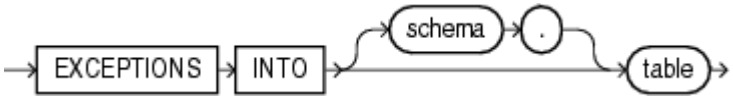




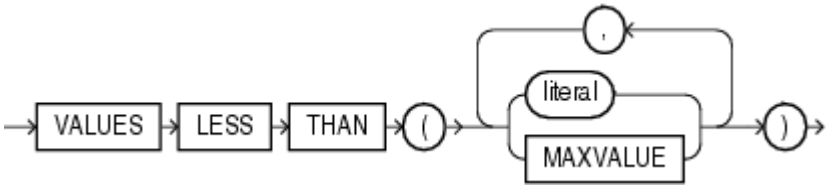


(partition_extended_name::=、subpartition_extended_name::=、exceptions_clause::=、update_index_clauses::=、parallel_clause::=)

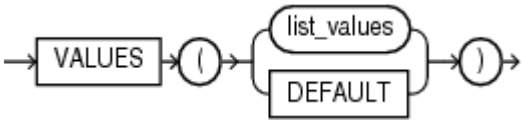
exceptions_clause::=



range_values_clause::=

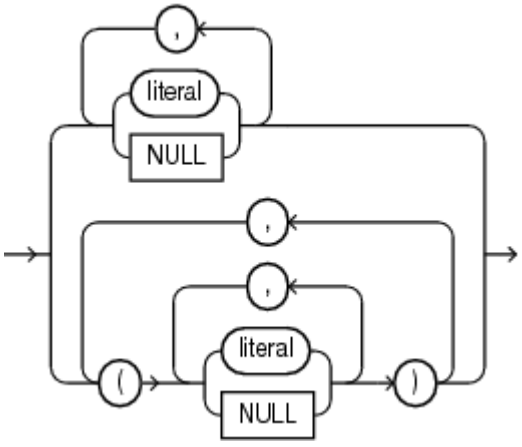


list_values_clause::=

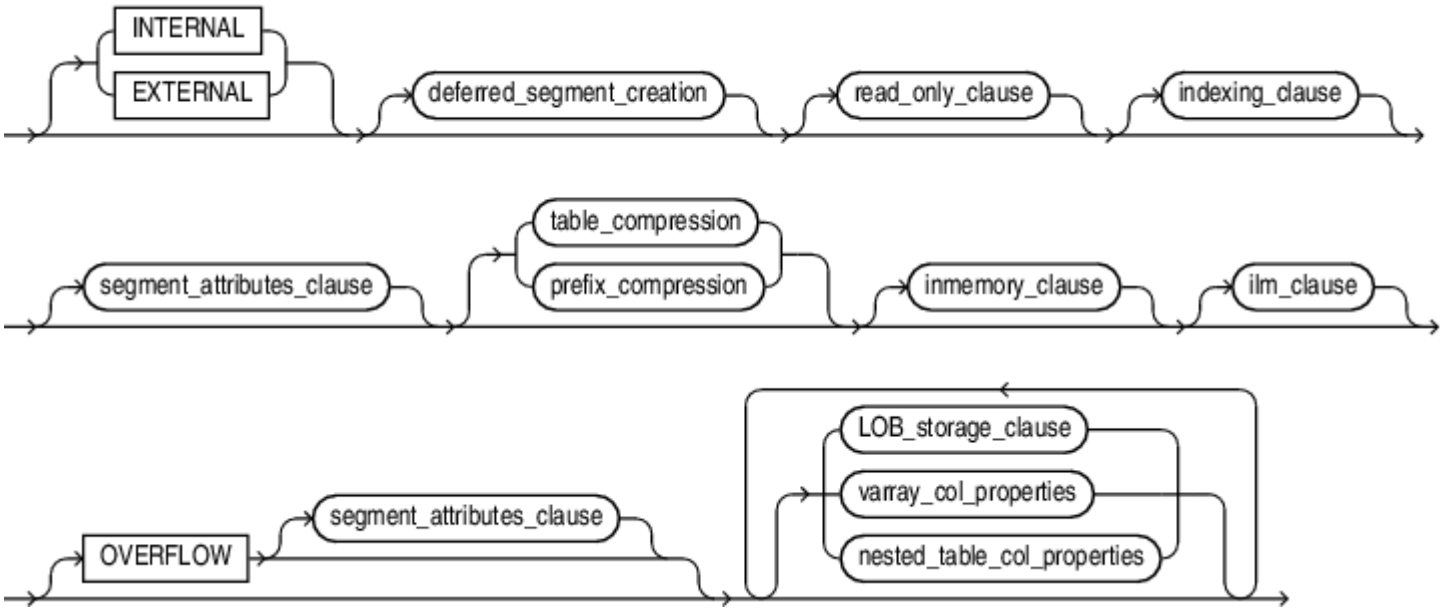


(list_values::=)

list_values::=

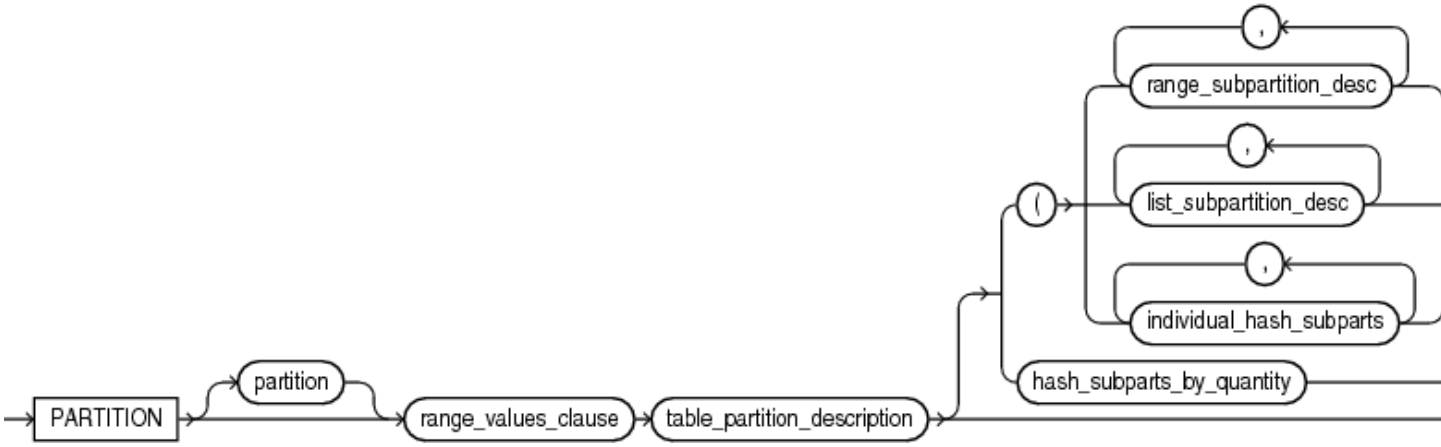


table_partition_description::=



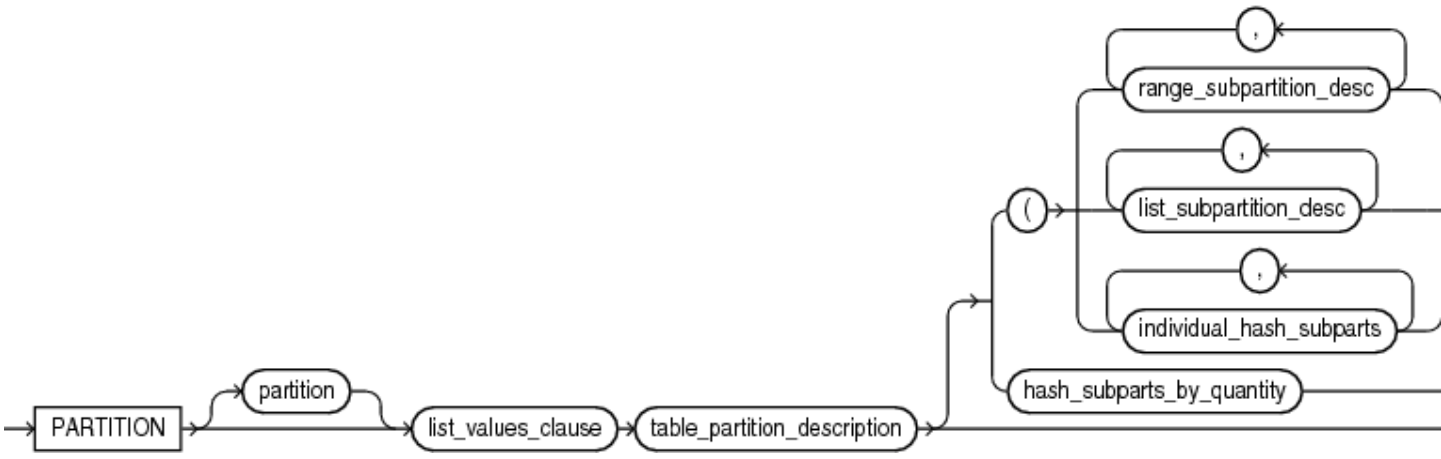
(deferred_segment_creation::=、read_only_clause::=、indexing_clause::=、segment_attributes_clause::=、table_compression::=、prefix_compression::=、inmemory_clause::=、LOB_storage_clause::=、varray_col_properties::=)

range_partition_desc::=



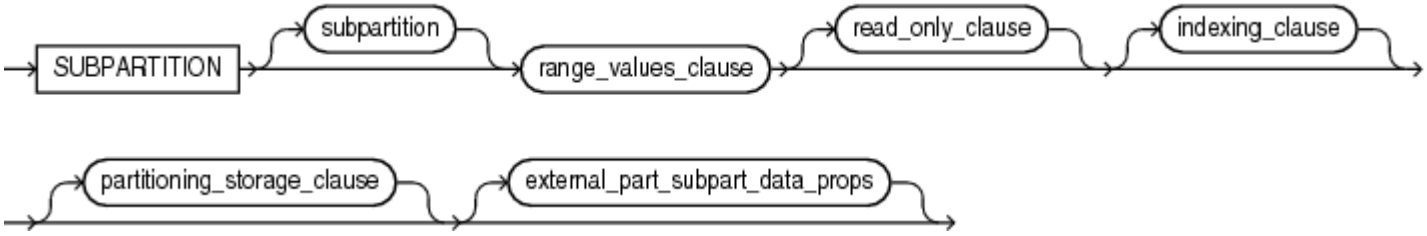
(range_values_clause::=、table_partition_description::=、range_subpartition_desc::=、list_subpartition_desc::=)

list_partition_desc::=



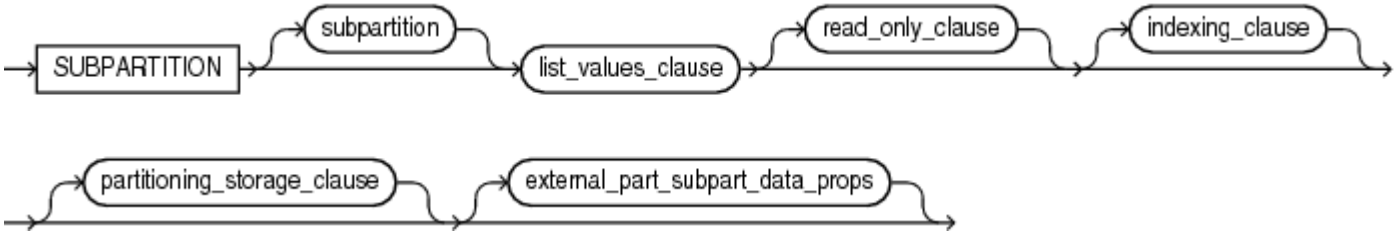
(list_values_clause::=、table_partition_description::=、range_subpartition_desc::=、list_subpartition_desc::=)

range_subpartition_desc::=



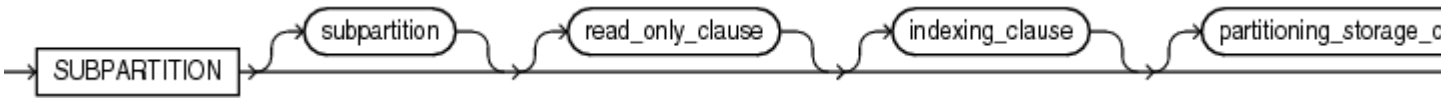
(range_values_clause::=、read_only_clause::=、indexing_clause::=、partitioning_storage_clause::=、external_part_subpart_data_props::=)

list_subpartition_desc::=



(list_values_clause::=、read_only_clause::=、indexing_clause::=、partitioning_storage_clause::=、external_part_subpart_data_props::=)

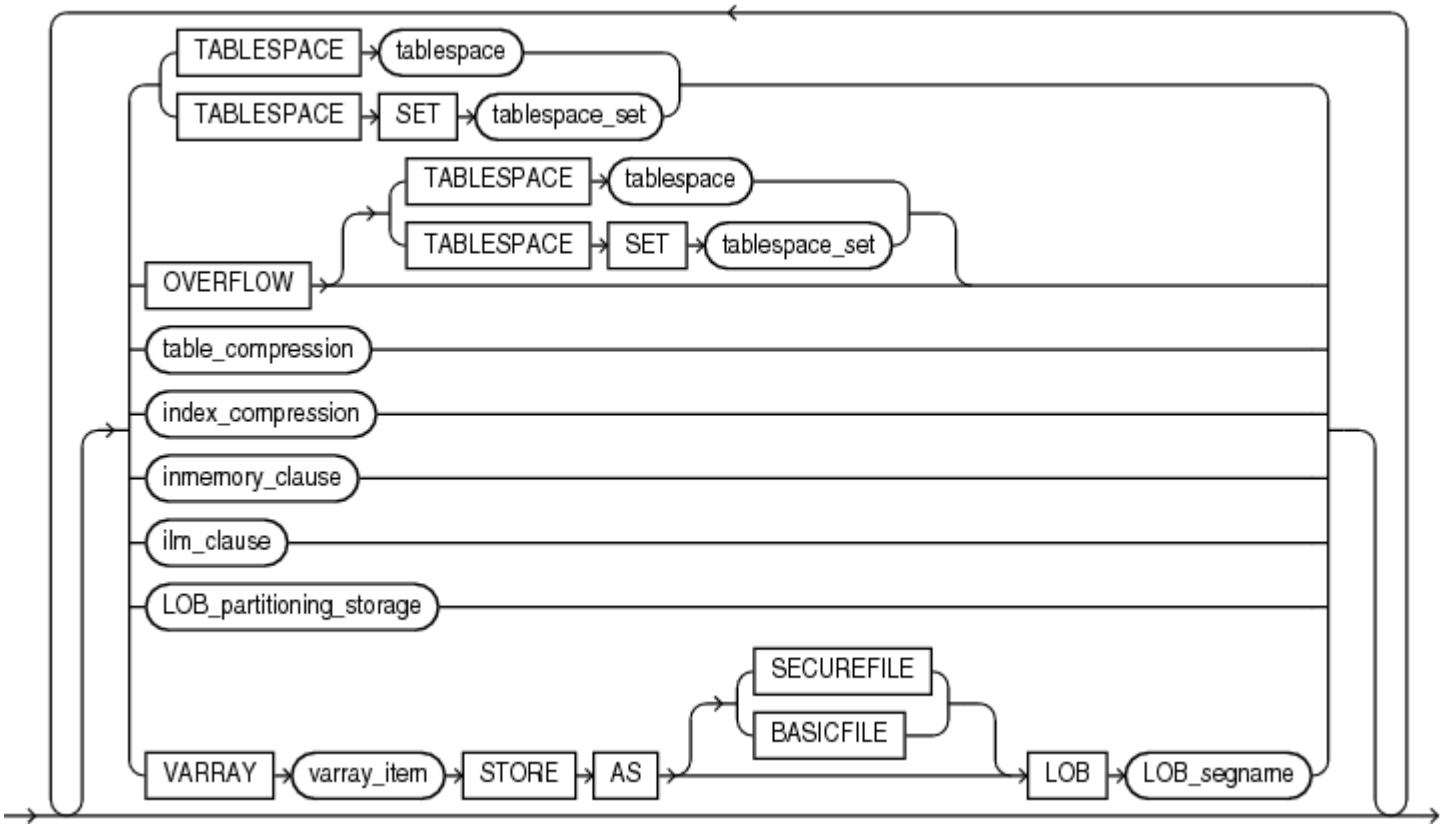
individual_hash_subparts::=



(read_only_clause::=、indexing_clause::=、partitioning_storage_clause::=)

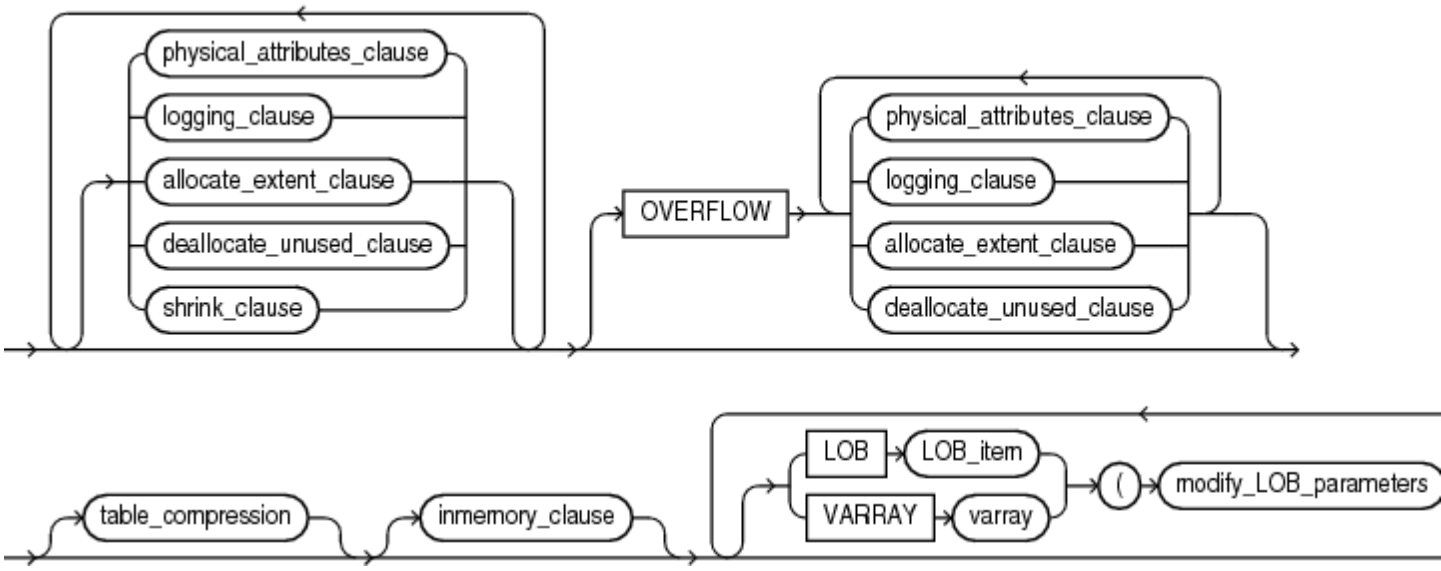
hash_subparts_by_quantity::=

partitioning_storage_clause::=



(TABLESPACE SET: ALTER TABLEではサポートされていません、table_compression::=、
index_compression::=、inmemory_clause::=、LOB_partitioning_storage::=)

partition_attributes::=



(physical_attributes_clause::=、logging_clause::=、allocate_extent_clause::=、

deallocate_unused_clause::=、shrink_clause::=、table_compression::=、inmemory_clause::=、modify_LOB_parameters::=)

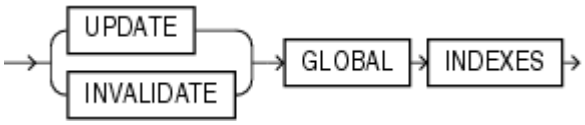
partition_spec::=

(table_partition_description::=)

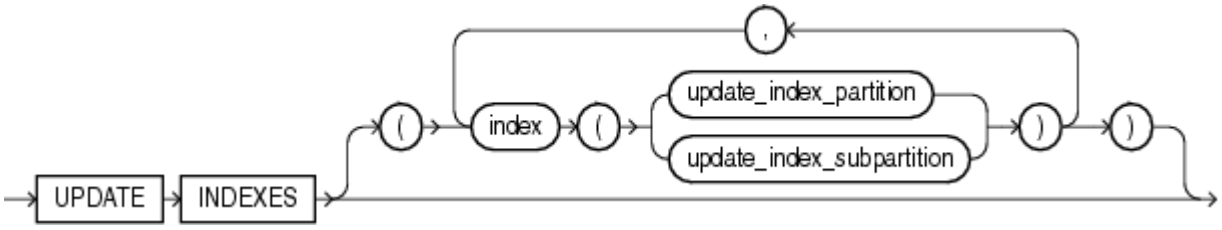
update_index_clauses::=

(update_global_index_clause::=、update_all_indexes_clause::=)

update_global_index_clause::=

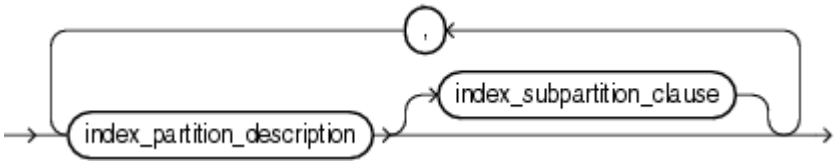


update_all_indexes_clause::=



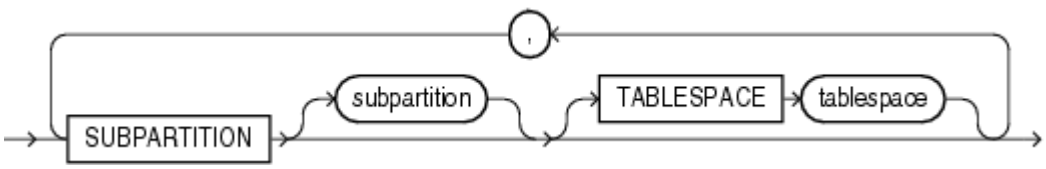
(update_index_partition::=、update_index_subpartition::=)

update_index_partition::=

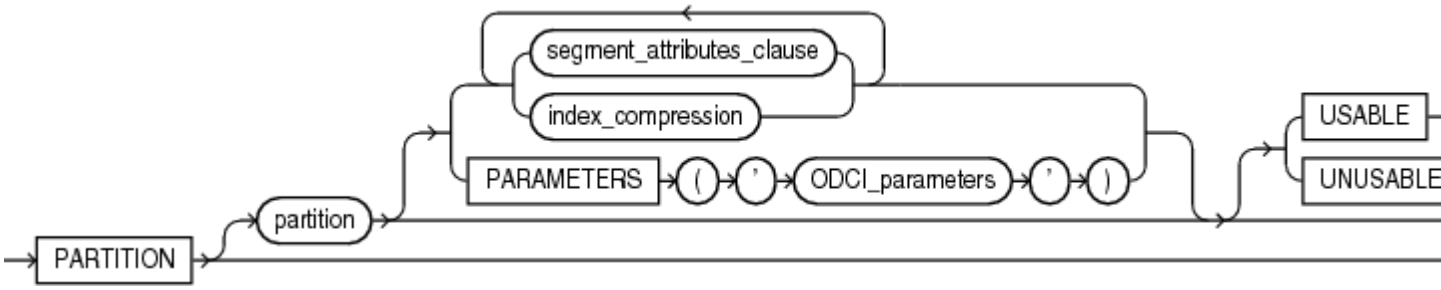


(index_partition_description::=、index_subpartition_clause::=)

update_index_subpartition::=

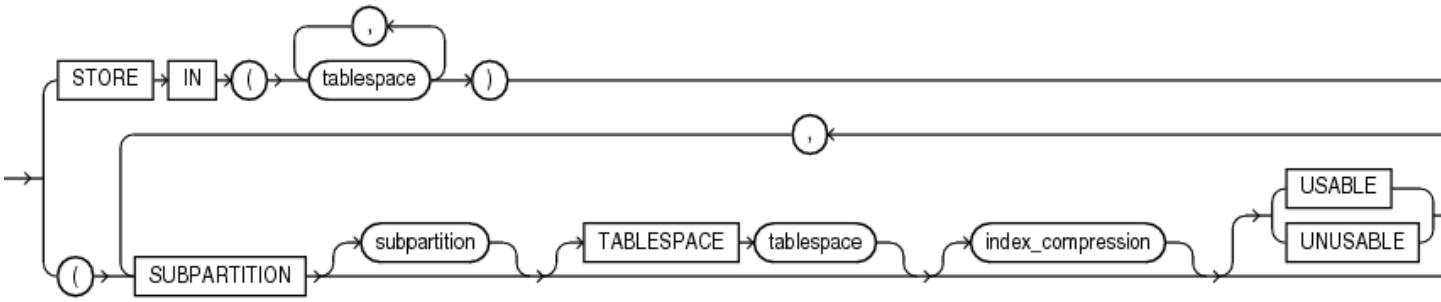


index_partition_description::=



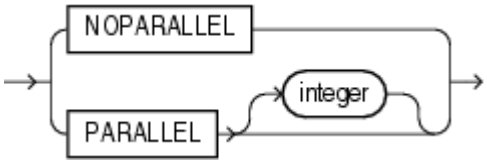
(segment_attributes_clause::=、index_compression::=)

index_subpartition_clause::=

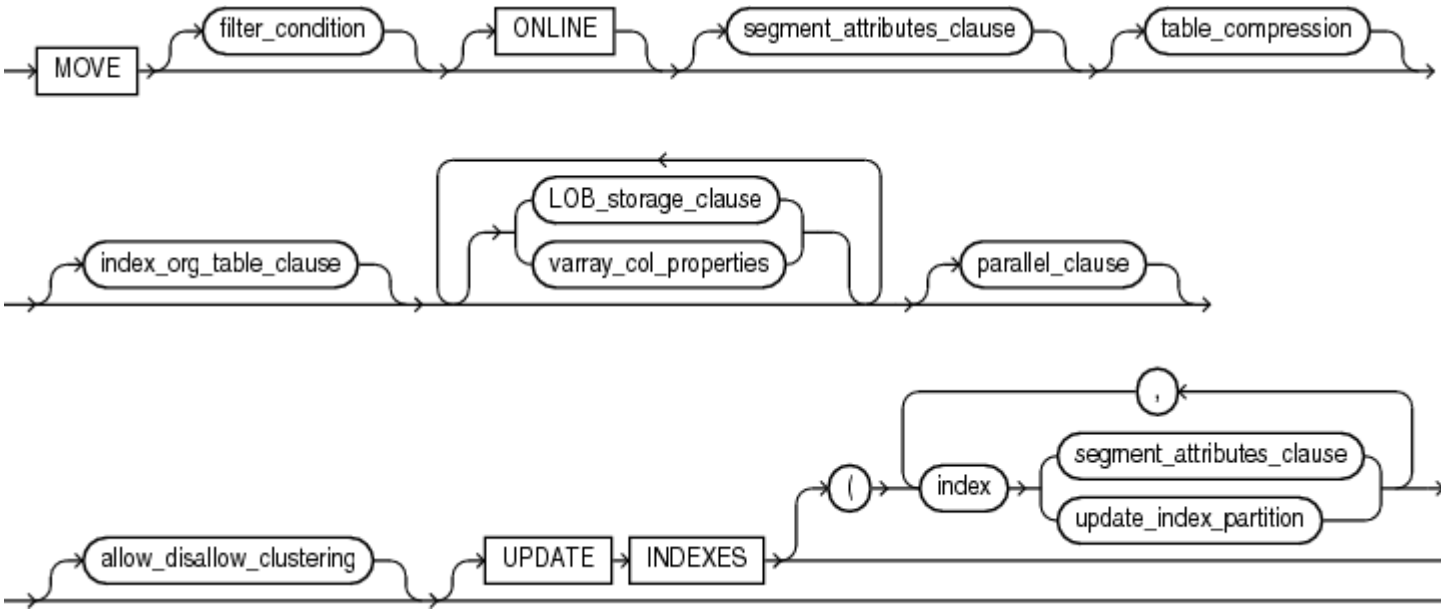


(index_compression::=)

parallel_clause::=



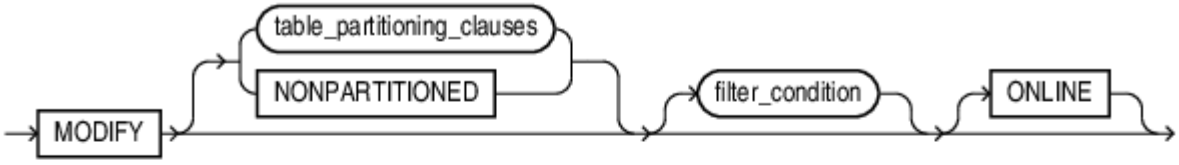
move_table_clause::=

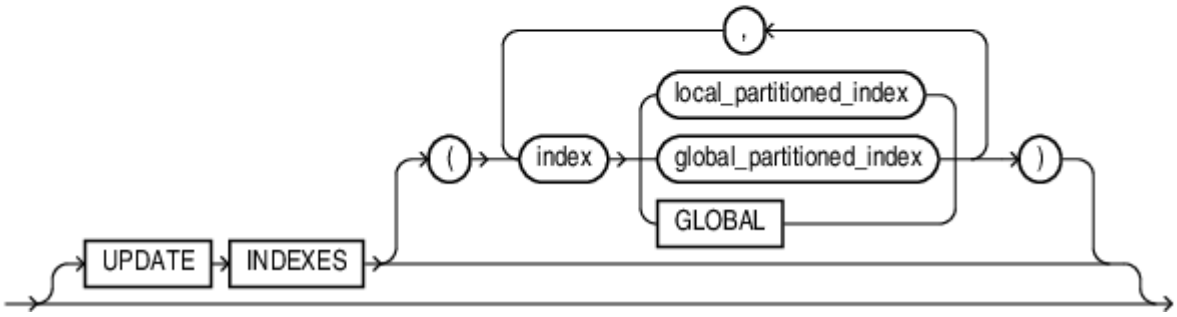


(filter_condition::=、segment_attributes_clause::=、table_compression::=、
index_org_table_clause::=、LOB_storage_clause::=、varray_col_properties::=、parallel_clause::=、

allow_disallow_clustering::=、update_index_partition::=)

modify_to_partitioned::=

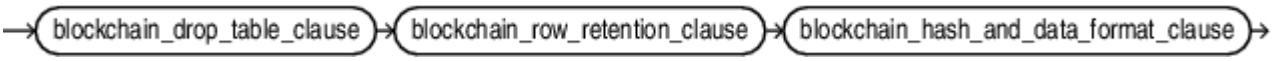




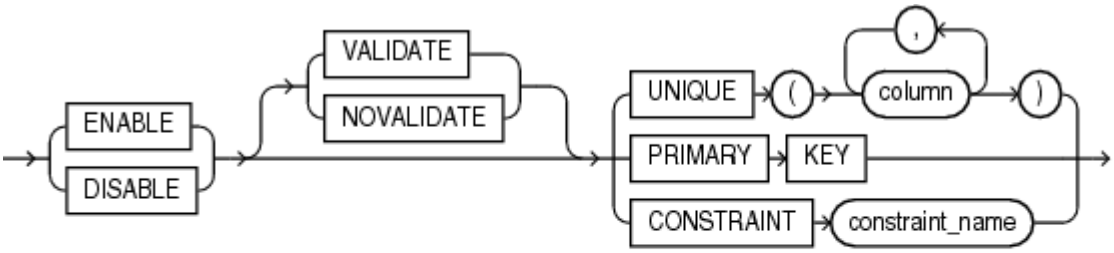
modify_opaque_type::=

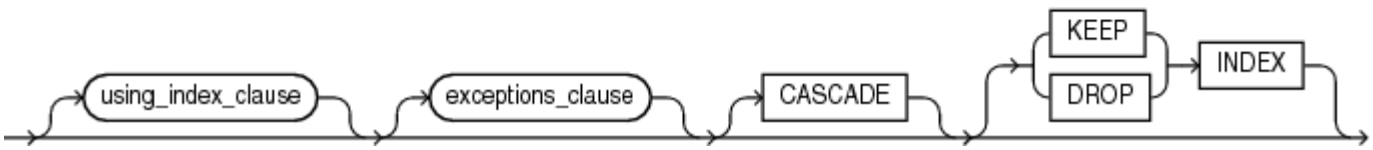
immutable_table_clauses::=

blockchain_table_clauses::=



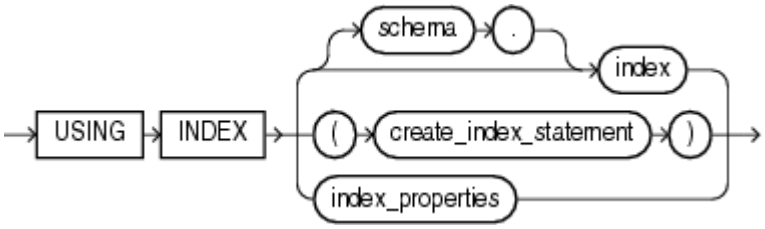
enable_disable_clause::=





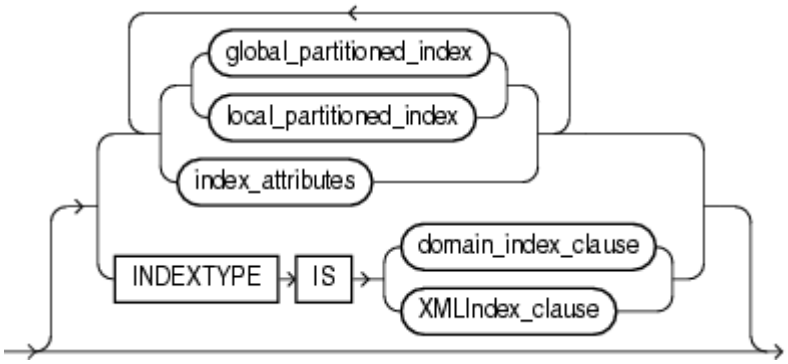
(using_index_clause::=、exceptions_clause::=)

using_index_clause::=



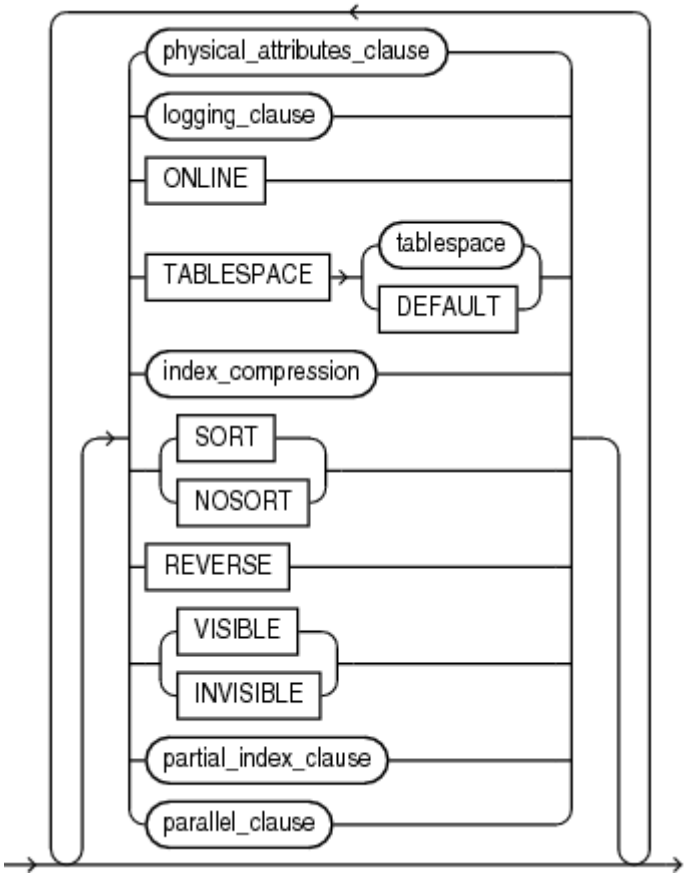
(create_index::=、index_properties::=)

index_properties::=



(global_partitioned_index::=、local_partitioned_index::= (CREATE INDEXの一部)、index_attributes::=、domain_index_clause: using_index_clauseではサポートされません)

index_attributes::=



(physical_attributes_clause::=、logging_clause::=、index_compression::=、partial_index_clauseおよびparallel_clause: (using_index_clauseではサポートされません))

セマンティクス

ALTER TABLE文の多くの句の機能は、CREATE TABLE文での機能と同じです。これらの句の詳細は、「CREATE TABLE」を参照してください。

> ノート:
>
> ALTER TABLE 文を実行すると、変更した表にアクセスするプロシージャおよびストアド・ファンクションが無効になる場合があります。無効になる条件および具体的な状況については、『Oracle Database 開発ガイド』を参照してください。

schema

表が含まれているスキーマを指定します。schemaを指定しない場合、表は自分のスキーマ内にあるとみなされます。

table

変更する表の名前を指定します。

> ノート:
>
> 変更対象の表が、1 つ以上のマテリアライズド・ビューのマスター表である場合は、そのマテリアライズド・ビューの状態が INVALID に設定されます。無効なマテリアライズド・ビューは、クエリー・リライトには使用できず、リフレッシュもできません。マテリアライズド・ビューを再び有効にする方法は、「ALTER MATERIALIZED VIEW」を参照してください。

関連項目:

マテリアライズド・ビューに関する一般的な情報については、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

一時表の変更の制限事項

一時表の列を変更または削除したり、一時表の名前を変更できます。ただし、一時表に対して次のことはできません。

- ネストした表型の列の追加(他の型の列は追加できます)。
- 追加または変更された列の参照整合性(外部キー)制約の指定。
- 追加または変更されたLOB列に対する次のLOB_storage_clauseの句の指定: TABLESPACE、storage_clause、logging_clause、allocate_extent_clauseまたはdeallocate_unused_clause。
- physical_attributes_clause、nested_table_col_properties、parallel_clause、allocate_extent_clause、deallocate_unused_clauseまたは索引構成表(IOT)句の指定。

- パーティション表と一時表の間でのパーティションの交換。
- logging_clauseの指定。
- MOVEの指定。
- INVISIBLEの列の追加、または既存の列をINVISIBLEにするための変更。

外部表の変更の制限事項

外部表の列を追加、削除または変更できます。ただし、外部表に対して次のことはできません。

- LONG、LOBまたはオブジェクト型の列の追加、または外部表の列のデータ型をこれらのデータ型のいずれかに変換
- 外部表の記憶域パラメータの変更。
- logging_clauseの指定。
- MOVEの指定。
- INVISIBLEの列の追加、または既存の列をINVISIBLEにするための変更。

memoptimize_read_clause

この句を使用して、頻度の高いデータ問合せ操作のパフォーマンスを向上させます。MEMOPTIMIZE_POOL_SIZE初期化パラメータは、memoptimizeプールのサイズを制御します。この機能は、SGAから追加のメモリーを使用することに注意してください。

- この句は、表の最上位の属性として指定する必要があり、パーティション・レベルまたはサブパーティション・レベルで指定することはできません。
- 表からデータを読み取る前に、MEMOPTIMIZE FOR READに対して表を明示的に有効にする必要があります。
- 表が不要になった場合は、NO MEMOPTIMIZE FOR READに対して表を明示的に無効にする必要があります。

memoptimize_write_clause

高速収集を有効にするには、この句を使用します。高速収集は、ディスクに書き込む前に挿入を格納するバッファ・プールを利用して、モノのインターネット(IoT)アプリケーションから1行データを挿入する頻繁な処理を最適化します。

alter_table_properties

alter_table_clausesを使用すると、データベースの表を変更できます。

physical_attributes_clause

physical_attributes_clauseを使用すると、PCTFREE、PCTUSEDおよびINITRANSパラメータの値と記憶特性を変更できます。これらのパラメータおよび特性の詳細は、「physical_attributes_clause」および「storage_clause」を参照してください。

表の物理属性の変更の制限事項

物理属性の変更には、次の制限事項があります。

- PCTUSEDパラメータは、索引構成表の索引セグメントに対して指定できません。
- ローカル管理表領域にある表の記憶域属性を変更しようとすると、Oracle Databaseエラーが発生します。ただし、パーティション表のセグメントのいくつかがローカル管理表領域にあり、それ以外のセグメントがディクショナリ管理表領域にある場合は、ディクショナリ管理表領域のセグメントの記憶域属性は変更されますが、ローカル管理表領域のセグメントの記憶域属性は変更されず、エラーも発生しません。

- 自動セグメント領域管理のセグメントの場合は、PCTUSED設定の変更は無視されます。PCTFREE設定を変更した場合、その変更をセグメントに割当て済のブロックに実装するには、DBMS_REPAIR.SEGMENT_FIX_STATUSプロシージャを実行する必要があります。

表の物理属性の変更の注意事項

この句で指定する値は、次のように表に影響します。

- 非パーティション表の場合、作成時に表に指定した値は新しく指定した値によって上書きされます。
- レンジ・パーティション表、リスト・パーティション表またはハッシュ・パーティション表の場合、新しく指定した値がその表のデフォルト値およびすべての既存パーティションに対する実際の値となり、そのパーティションにすでに設定されていた値は上書きされます。既存のパーティション値を上書きせずにデフォルトの表属性を変更する場合は、modify_table_default_attrs句を使用してください。
- コンポジット・パーティション表の場合、新しく指定した値がその表とその表のすべてのパーティションのデフォルト値、およびその表のすべてのサブパーティションに対する実際の値となり、そのサブパーティションにすでに設定されていた値は上書きされます。既存のサブパーティションの値を上書きせずにデフォルトのパーティション属性を変更する場合は、FOR PARTITION句とともにmodify_table_default_attrs句を使用してください。

logging_clause

logging_clauseを使用すると、表のロギング属性を変更できます。また、logging_clauseでは、後続のALTER TABLE ... MOVEおよびALTER TABLE ... SPLIT操作のログを記録するかどうかを指定します。

modify_table_default_attrs句とともにこの句を使用した場合、パーティション表のロギング属性が影響を受けます。

関連項目:

- この句の詳細は、「logging_clause」を参照してください。
- logging_clauseおよびパラレルDMLの詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

table_compression

table_compression句は、ヒープ構成表に対してのみ有効です。この句を使用すると、ディスクおよびメモリーの使用量を削減するために、データ・セグメントを圧縮するかどうかを指定できます。この句のセマンティクスの詳細および表の圧縮を使用したオブジェクトの作成方法の詳細は、「CREATE TABLE」の「table_compression」を参照してください。

ノート:

初めて、圧縮データが追加されるように表を変更する場合は、表のすべてのビットマップ索引およびビットマップ索引パーティションに UNUSABLE のマークを付ける必要があります。

inmemory_table_clause

この句を使用して、インメモリー列ストア(IM列ストア)の表または表の列を有効化または無効化するか、表または表の列のインメモリーの属性を変更します。

- INMEMORYを指定して、IM列ストアの表を有効化するか、IM列ストアにすでに有効化されている表のinmemory_attributesを変更します。
- NO INMEMORYを指定して、IM列ストアの表を無効化します。
- inmemory_column_clauseを指定して、IM列ストアの表の列を有効化または無効化するか、表の列のinmemory_memcompress設定を変更します。IM列ストアに対して表またはパーティションが無効になっているときにこの句を指定した場合、列の設定は、その後IM列ストアに対して表またはパーティションが有効化されたときに有効になります。IM列ストアに対して表またはパーティションが有効になっているか無効になっているかにかかわらず、列に対してNO INMEMORYを指定した場合は、列に対して以前に指定されたinmemory_memcompress設定が失われます。この句のセマンティクスの詳細は、「CREATE TABLE」の「inmemory_column_clause」を参照してください。

このinmemory_table_clauseのセマンティクスはCREATE TABLEのinmemory_table_clauseと同じですが、次の追加点があります。

- inmemory_memcompress句を指定して、IM列ストアにすでに有効化されている表のデータ圧縮方法を変更する場合、以前に特定のデータ圧縮方法を割り当てた列にデータ圧縮方法が保持されます。この句の詳細は、CREATE TABLEのinmemory_memcompress句を参照してください。
- inmemory_distribute句を指定する際に1つの副句を省略した場合、設定は変更されないままです。つまり、AUTOまたはBY句のみを指定した場合、表のFOR SERVICE設定は変更されないままで、FOR SERVICE句のみを指定した場合、表のAUTOまたはBY設定は変更されないままです。両方の副句を省略してDISTRIBUTEキーワードのみを指定した場合、表にはDISTRIBUTE AUTO設定が割り当てられ、FOR SERVICE設定は変更されないままです。この句の詳細は、CREATE TABLEのinmemory_distribute句を参照してください。
- NO INMEMORYを指定してIM列ストアのパーティション表または非パーティション表を無効化すると、列レベルのインメモリー設定が失われます。その後、IM列ストアの表を有効化すると、表を有効化するときに特に指定しないかぎり、すべての列で表のインメモリー設定が使用されます。
- NO INMEMORYを指定してIM列ストアのパーティションを無効化した場合は、表のすべてのパーティションが無効化されていても、列レベルのインメモリー設定が保持されます。その後、IM列ストアの表またはパーティションを有効化すると、表またはパーティションを有効化するときに特に指定しないかぎり、列レベルのインメモリー設定が有効になります。
- 表が現在IM列ストアに移入されており、PRIORITY以外の表のinmemory_attributeを変更した場合、データベースではIM列ストアからその表を削除します。再移入動作はPRIORITY設定によって異なります。

inmemory_clause

この句を使用して、IM列ストアの表のパーティションを有効化または無効化するか、表のパーティションのインメモリーのパラメータを変更します。この句のセマンティクスは、CREATE TABLEおよびALTER TABLEで同じです。この句のセマンティクスの詳細は、CREATE TABLEのドキュメントのinmemory_clauseを参照してください。

ORACLE_HIVE、ORACLE_HDFS、およびORACLE_BIGDATAのドライバを使用する非パーティション表にIMEMORYを指定できます。

制限

ディスク上のセグメントが64KB以下の場合、IM列ストアに移入されません。したがって、IM列ストアに対して有効になっている小規模データベース・オブジェクトは、移入されないことがあります。

ilm_clause

この句を使用すると、表の自動データ最適化ポリシーの追加、削除、有効化または無効化を実行できます。

ADD POLICY

この句を指定すると、表にポリシーを追加できます。

ポリシーを指定するには、`ilm_policy_clause`を使用します。この句のセマンティクスの詳細は、「ilm_policy_clause」を参照してください

ポリシーにはPnという形式の名前が割り当てられます。nは整数値です。

{ DELETE | ENABLE | DISABLE } POLICY

これらの句を指定して、表のポリシーの削除、表のポリシーの有効化または表のポリシーの無効化をそれぞれ実行します。

`ilm_policy_name`には、ポリシーの名前を指定します。ポリシー名は、`DBA_ILMPOLICIES`ビューの`POLICY_NAME`列に問い合せて確認できます。

{ DELETE_ALL, ENABLE_ALL, DISABLE_ALL }

これらの句を指定して、表のすべてのポリシーの削除、表のすべてのポリシーの有効化または表のすべてのポリシーの無効化をそれぞれ実行します。

関連項目:

自動データ最適化ポリシーの管理の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

ilm_policy_clause

この句を使用すると、自動データ最適化ポリシーを指定できます。`ilm_compression_policy`句を使用して圧縮ポリシーを指定し、`ilm_tiering_policy`句を使用して記憶域階層化ポリシーを指定し、`ilm_inmemory_policy`句を使用してインメモリー列ストア・ポリシーを指定できます。

ilm_compression_policy

この句を使用すると、圧縮ポリシーを指定できます。このタイプのポリシーは、指定された条件が満たされた場合、データを圧縮するようデータベースに指示します。`SEGMENT`、`GROUP`または`ROW`句を使用して、セグメントレベル、グループレベルまたは行レベルの圧縮ポリシーを指定します。

table_compression

`table_compression`句を使用して、圧縮タイプを指定します。この句は、セグメントレベルおよびグループレベルの圧縮ポリシーに適用されます。

現在の圧縮タイプより高い圧縮タイプを指定する必要があります。圧縮タイプを、最低のものから最高のものの順に示します。

- `NOCOMPRESS`
- `ROW STORE COMPRESS BASIC`
- `ROW STORE COMPRESS ADVANCED`
- `COLUMN STORE COMPRESS FOR QUERY LOW`
- `COLUMN STORE COMPRESS FOR QUERY HIGH`
- `COLUMN STORE COMPRESS FOR ARCHIVE LOW`
- `COLUMN STORE COMPRESS FOR ARCHIVE HIGH`

この句のセマンティクスの詳細は、「table_compression」を参照してください。

SEGMENT

SEGMENTを指定して、セグメントレベルの圧縮ポリシーを作成します。このタイプのポリシーは、AFTER句で指定した条件が満たされた場合またはON句で指定したPL/SQLファンクションがTRUEを返す場合に表セグメントを圧縮するようデータベースに指示します。

ALTER TABLEを使用してセグメントを変更できないことに注意してください。

GROUP

GROUPを指定して、グループレベルの圧縮ポリシーを作成します。このタイプのポリシーは、AFTER句で指定した条件が満たされた場合またはON句で指定したPL/SQLファンクションがTRUEを返す場合に、表とその従属オブジェクト(索引やセキュア・ファイルLOBなど)を圧縮するようデータベースに指示します。

ROW

ROWを指定して、行レベルの圧縮ポリシーを作成します。このタイプのポリシーは、指定された期間にすべての行が変更されていないデータベース・ブロックを圧縮するようデータベースに指示します。行レベルのポリシーを作成する場合は、ROW STORE COMPRESS ADVANCEDまたはCOLUMN STORE COMPRESS FOR QUERY圧縮を指定し、AFTER ilm_time_period OF NO MODIFICATIONを指定する必要があります。ROW STORE COMPRESS ADVANCEDおよびCOLUMN STORE COMPRESS FOR QUERY句の完全なセマンティックスについては、「table_compression」を参照してください。

AFTER

この句を使用すると、ポリシーを有効にするために満たす必要のある条件を定義できます。条件は、ilm_time_period句で指定する時間と、次のいずれかの条件タイプで構成されます。

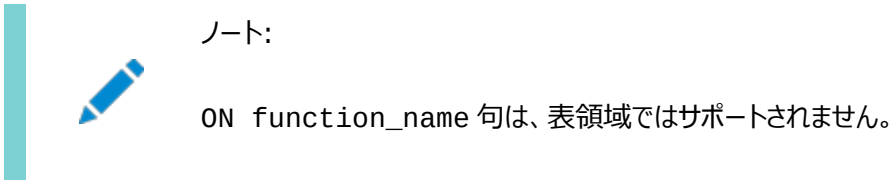
- OF NO ACCESS: 指定した時間にわたってtableがアクセスされないと、ポリシーが有効になります。
- OF NO MODIFICATION: 指定した時間にわたってtableが変更されないと、ポリシーが有効になります。
- OF CREATION: tableが作成されてから指定した時間が経過すると、ポリシーが有効になります。

ilm_time_period

その期間の経過後に条件が満たされている必要がある期間の長さを、日数、月数または年数で指定します。integerに、正の整数を指定します。DAYキーワードとDAYSキーワードは、区別なしに使用できますが、意味を明確にするために用意されています。これはMONTHキーワードとMONTHSキーワード、およびYEARキーワードとYEARSキーワードの場合も同様です。

ON

この句を使用すると、ブール値を戻すPL/SQLファンクションを指定できます。function_nameに、ファンクションの名前を指定します。このファンクションがTRUEを返すと、ポリシーが有効になります。



ノート:

ON function_name 句は、表領域ではサポートされません。

ilm_tiering_policy

この句を使用すると、記憶域階層化ポリシーを指定できます。このタイプのポリシーは、特定の条件が満たされたか、またはデータ使用量が指定された制限に達したいずれかの場合に、データを指定された表領域に移行するようデータベースに指示します。

SEGMENTまたはGROUP句を使用して、セグメントレベルまたはグループレベルのポリシーを指定します。データは、読取り/書込み表領域または読取り専用表領域に移行できます。

TIER TO tablespace

この句を使用すると、データを読取り/書込みtablespaceに移行できます。

- ON function句を指定した場合、functionがTRUEを返すときにデータが移行されます。この句のセマンティックスの詳細は、ON句を参照してください。
- ON function句を省略すると、表領域割当てのデータ使用量がTBS_PERCENT_USEDで定義された割合に達した場合、データが移行されます。表領域割当ての空き領域量がTBS_PERCENT_FREEで定義されたパーセントに達するだけの十分な量のデータが可能なかぎり移行されます。DBMS_ILM_ADMINパッケージの定数であるTBS_PERCENT_USEDおよびTBS_PERCENT_FREEの詳細は、Oracle Database PL/SQLパッケージおよびタイプ・リファレンスを参照してください。

TIER TO tablespace READ ONLY

この句を使用すると、データを読取り専用表領域に移行できます。データを表領域に移行する際、データベースは一時的に表領域を読取り/書込みモードにしてデータを移行し、その後表領域を読取り専用モードに戻します。

- AFTER句を指定した場合、指定した条件が満たされたときにデータが移行されます。この句のセマンティックスの詳細は、AFTER句を参照してください
- ON function句を指定した場合、functionがTRUEを返すときにデータが移行されます。この句のセマンティックスの詳細は、ON句を参照してください。

SEGMENT | GROUP

SEGMENTを指定すると、セグメントレベルの記憶域階層化ポリシーを作成できます。このタイプのポリシーは、表セグメントをtablespaceに移行するようデータベースに指示します。GROUPを指定して、グループレベルの記憶域階層化ポリシーを作成します。このタイプのポリシーは、表とその従属オブジェクト(索引やセキュア・ファイルLOBなど)をtablespaceに移行するようデータベースに指示します。デフォルトはSEGMENTです。

ノート:

ON function_name 句は、表領域ではサポートされません。

ilm_inmemory_policy

この句を使用すると、インメモリー列ストア(IM列ストア)ポリシーを指定できます。このタイプのポリシーは、指定した条件が満たされた場合に、IM列ストアの表を有効化または無効化するか、IM列ストアの表の圧縮方法を変更するようにデータベースに指示します。

SET INMEMORY

この句を使用すると、指定された条件が満たされた場合に、IM列ストアの表を有効化できます。オプションでinmemory_attributes句を使用して、表データをIM列ストアに格納する方法を指定できます。この句のセマンティックスの詳細は、「inmemory_attributes」を参照してください。

MODIFY INMEMORY

この句を使用すると、指定された条件が満たされた場合に、IM列ストアに格納されている表データの圧縮方法を変更できます。IM列ストアに対して表が有効化されている必要があります。

現在の圧縮方法より高い圧縮方法を指定する必要があります。圧縮方法を、最低のものから最高のものの順に示します。

- NO INMEMORY
- MEMCOMPRESS FOR DML
- MEMCOMPRESS FOR QUERY LOW
- MEMCOMPRESS FOR QUERY HIGH
- MEMCOMPRESS FOR CAPACITY LOW
- MEMCOMPRESS FOR CAPACITY HIGH

この句のセマンティクスの詳細は、「inmemory_memcompress」を参照してください。

NO INMEMORY

この句を使用すると、指定された条件が満たされた場合に、IM列ストアの表を無効化できます。

SEGMENT

SEGMENTキーワードはオプションで、セマンティクスを明確化するために使用されます。IM列ストア・ポリシーは、常にセグメントレベルのポリシーです。

AFTER | ON

AFTERおよびON句を使用すると、IM列ストア・ポリシーが有効になるために満たされる必要がある条件を指定できます。

- AFTER句を指定すると、指定された条件が満たされたときにポリシーが有効になります。この句のセマンティックスの詳細は、AFTER句を参照してください
- ONファンクション句を指定した場合は、ファンクションからTRUEが戻されたときにポリシーが有効になります。この句のセマンティックスの詳細は、ON句を参照してください。

ノート:

ON function_name 句は、表領域ではサポートされません。

関連項目:

IM列ストアでの自動データ最適化ポリシーの使用方法の詳細は、『Oracle Database In-Memoryガイド』を参照してください。

supplemental_table_logging

supplemental_table_logging句を使用すると、REDOログ・グループ、またはサプリメンタル・ログが記録されるREDOログ・グループの1つ以上の列を追加または削除できます。

- ADD句でsupplemental_log_grp_clauseを使用すると、指定したサプリメンタル・ログ・グループを作成できます。supplemental_id_key_clauseを使用すると、システム生成のログ・グループを作成できます。
- DROP句でGROUP log_group構文を使用すると、指定したサプリメンタル・ログ・グループを削除できます。supplemental_id_key_clauseを使用すると、システム生成のログ・グループを削除できます。

supplemental_log_grp_clauseおよびsupplemental_id_key_clauseは、CREATE TABLE文および

ALTER TABLE文で同じセマンティクスを持ちます。これらの句の詳細は、「CREATE TABLE」の「supplemental_log_grp_clause」および「supplemental_id_key_clause」を参照してください。

関連項目:

サプリメンタルREDOログ・グループの詳細は、『Oracle Data Guard概要および管理』を参照してください。

allocate_extent_clause

allocate_extent_clauseを使用すると、表、パーティション、サブパーティション、オーバーフロー・データ・セグメント、LOBデータ・セグメントまたはLOB索引に新しいエクステントを明示的に割り当てることができます。

表エクステントの割当ての制限事項

一時表、レンジ・パーティション表またはコンポジット・パーティション表にエクステントを割り当てることはできません。

関連項目:

この句の詳細は、「allocate_extent_clause」および「エクステントの割当て: 例」を参照してください。

deallocate_unused_clause

deallocate_unused_clause deallocate_unused_clauseを使用すると、表、パーティション、サブパーティション、オーバーフロー・データ・セグメント、LOBデータ・セグメントまたはLOB索引の最後にある未使用領域の割当てを明示的に解除できます。解放された領域は、表領域の他のセグメントから利用できます。

関連項目:

この句の詳細は、「deallocate_unused_clause」および「未使用領域の割当て解除: 例」を参照してください。

CACHE | NOCACHE

CACHE句およびNOCACHE句のセマンティクスは、CREATE TABLE文およびALTER TABLE文で同じです。これらの句の詳細は、「CREATE TABLE」の「CACHE | NOCACHE | CACHE READS」を参照してください。ALTER TABLE文でこれらの句をいずれも省略した場合、既存の値は変更されません。

RESULT_CACHE

RESULT_CACHE句は、CREATE TABLE文およびALTER TABLE文と同じセマンティクスを持ちます。この句の詳細は、「CREATE TABLE」の「RESULT_CACHE句」を参照してください。ALTER TABLE文でこの句を指定しない場合、既存の設定は変更されません。

upgrade_table_clause

upgrade_table_clauseが意味を持つのは、オブジェクト表や、オブジェクト列を持つリレーショナル表に対して使用する場合です。ターゲットとなる表のメタデータが、参照される型それぞれの最新バージョンに準拠するように変換されます。表がすでに有効なものである場合は、表のメタデータは変更されません。

オブジェクト表および列のアップグレードの制限事項

この句内で、column_propertiesの句としてobject_type_col_propertiesを指定することはできません。

INCLUDING DATA

INCLUDING DATAを指定すると、表のデータを型の最新バージョンの形式に変換できます。column_propertiesおよびLOB_partition_storageを使用して表をアップグレードする間に、新しいすべての列の記憶域を定義できます。これはデフォルトです。

ALTER TYPE文のdependent_handling_clauseにCASCADE INCLUDING TABLE DATAを指定すると、型の更新時に表のデータを変換できます。この句の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。表が古いバージョンの型に基づくデータを含むかどうかを確認するには、USER_TAB_COLUMNSデータ・ディクショナリ・ビューのDATA_UPGRADED列を参照します。

NOT INCLUDING DATA

NOT INCLUDING DATAを指定すると、列データは変更されません。

NOT INCLUDING DATAの制限事項

表にOracle8リリース8.0.xのイメージ・フォーマットの列が含まれる場合、NOT INCLUDING DATAは指定できません。表がこのような列を含むかどうかを確認するには、USER_TAB_COLUMNSデータ・ディクショナリ・ビューのV80_FMT_IMAGE列を参照します。

関連項目:

- データ・ディクショナリ・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。
- 表が依存する型を変更する際に、依存表のデータを変換する場合の詳細は、「ALTER TYPE」を参照してください。

records_per_block_clause

records_per_block_clauseを使用すると、1ブロックに格納できるレコード数を制限するかどうかを指定できます。この句によって、この後、表に作成されるビットマップ索引はできるだけ縮小されます。

ブロックのレコードの制限事項

record_per_block_clauseには、次の制限事項があります。

- 表にすでにビットマップ索引が定義されている場合は、MINIMIZEまたはNOMINIMIZEのいずれも指定できません。まず、ビットマップ索引を削除する必要があります。
- この句は、索引構成表またはネストした表に対して指定できません。

MINIMIZE

MINIMIZEを指定すると、表の各ブロックの最大レコード数を計算し、ブロックに含まれるレコード数がその数を超えないように挿入操作を制限できます。

MINIMIZEを指定する前に、表にデータのサンプル・セットを定義しておくことをお薦めします。表の圧縮を使用している場合(「table_compression」を参照)、圧縮データのサンプル・セットは、すでに表に存在している必要があります。

MINIMIZEの制限事項

空の表にMINIMIZEを指定することはできません。

NOMINIMIZE

NOMINIMIZEを指定すると、MINIMIZE機能が無効になります。これはデフォルトです。

row_movement_clause

親表でも行の移動が無効化されている場合を除き、参照パーティション表で行の移動を無効にすることはできません。それ以外では、この句のセマンティクスは、CREATE TABLE文およびALTER TABLE文で同じです。これらの句の詳細は、「CREATE TABLE」の「row_movement_clause」を参照してください。

flashback_archive_clause

この句を指定するには、指定されたフラッシュバック・アーカイブに対するFLASHBACK ARCHIVEオブジェクト権限が必要です。この句を使用すると、表の履歴追跡を有効または無効にできます。

- 表の追跡を有効にするには、FLASHBACK ARCHIVEを指定します。flashback_archiveは、この表に特定のフラッシュバック・アーカイブを指定する場合に指定できます。指定するフラッシュバック・アーカイブは、すでに存在している必要があります。

  アーカイブ名を指定しない場合、データベースは、システムに指定されたデフォルトのフラッシュバック・アーカイブを使用します。システムにデフォルトのフラッシュバック・アーカイブが指定されていない場合は、flashback_archiveを指定する必要があります。

  FLASHBACK ARCHIVEを指定して、この表のフラッシュバック・アーカイブを変更することはできません。かわりに、まずALTER TABLE文にNO FLASHBACK ARCHIVE句を付けて実行し、次に、ALTER TABLE文にFLASHBACK ARCHIVE句を付けて実行します。

- 表の追跡を無効にするには、NO FLASHBACK ARCHIVEを指定します。

関連項目:

追跡が有効な表の作成の詳細は、CREATE TABLEのflashback_archive_clauseを参照してください。デフォルトのフラッシュバック・アーカイブの作成の詳細は、CREATE FLASHBACK ARCHIVEを参照してください。

RENAME TO

RENAME句を使用すると、tableの名前をnew_table_nameに変更できます。

この句を使用した場合、依存するすべてのマテリアライズド・ビューは無効になります。マテリアライズド・ビューの詳細は、「CREATE MATERIALIZED VIEW」および『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

表にドメイン索引が定義されている場合、ODCIIndexAlter()メソッドがRENAMEオプション付きで起動されます。この操作によって、索引タイプ・メタデータと実表の間の対応が確立されます。

表名の変更の制限事項

シャード表または重複表の名前は変更できません。

shrink_clause

shrink_clauseを使用すると、表、索引構成表またはそのオーバーフロー・セグメント、索引、パーティション、サブパーティション、LOBセグメント、マテリアライズド・ビューまたはマテリアライズド・ビュー・ログの領域を手動で縮小できます。この句は、自動セグメント管理を使用した表領域のセグメントに対してのみ有効です。デフォルトでは、セグメントが縮小されて最高水位標が調整され、再生領域がすぐに解放されます。

セグメントの縮小では、行移動が必要です。したがって、この句を指定する前に、縮小するオブジェクトの行移動を使用可能にする必要があります。また、アプリケーションでROWIDに基づいたトリガーを使用している場合、この句を発行する前にこのトリガーを使用禁止にする必要があります。

ノート:

shrink_clause を指定する前に、索引構成表に対して行の移動を有効にしないでください。索引構成表の ROWID は、その主キーであり、変更できません。このため、そのような表に対して行の移動は関係なく、有効でもありません。

制限事項:

shrink_clauseは、IOTパーティション表でサポートされていません。

COMPACT

COMPACTを指定すると、セグメント領域のデフラグのみが行われ、後で領域を解放できるように表の行が縮小されます。最高水位標の再調整および領域の解放は、すぐに行われません。操作を完了するには、後で別のALTER TABLE ... SHRINK SPACE文を発行する必要があります。この句は、1回の長いステップのかわりに2回の短いステップで縮小操作を実行する場合に便利です。

索引や索引構成表の場合は、ALTER [INDEX | TABLE] ... SHRINK SPACE COMPACTを指定することはALTER [INDEX | TABLE ... COALESCEを指定することと同じです。shrink_clauseはカスケード実行できます(次のCASCADE句を参照)。また結合操作の場合よりもセグメントを高密度に圧縮できるため、パフォーマンスが向上します。ただし、未使用領域を解放しない場合は、適切なCOALESCE句を使用します。

CASCADE

CASCADEを指定すると、tableのすべての依存オブジェクト(索引構成表の2次索引を含む)に対して同じ操作を実行できます。

shrink_clauseの制限事項

shrink_clauseには、次の制限事項があります。

- 同じALTER TABLE文で、この句と別の句を組み合せて使用することはできません。

  この句は、クラスタ、クラスタ化表またはLONG列を含むすべてのオブジェクトには指定できません。
- セグメントの縮小は、ファンクション索引、ドメイン索引またはビットマップ結合索引を含む表ではサポートされません。
- CASCADEを指定しても、この句では索引構成表のマッピング表を縮小できません。
- この句は、高度な行圧縮が有効な(ROW STORE COMPRESS ADVANCED)表に対して指定できます。その他の種類の表圧縮が有効な表に対しては、この句を指定できません。
- ON COMMITマテリアライズド・ビューのマスター表は縮小できません。ROWIDマテリアライズド・ビューは、縮小操作の実行後に再構築する必要があります。

READ ONLY | READ WRITE

READ ONLYを指定すると、表を読取り専用モードに設定できます。表がREAD ONLYモードの場合は、表に影響するDML文またはSELECT ... FOR UPDATE文は発行できません。表データを変更しないDDL文は発行できます。表がREAD ONLYモードの場合は、表に関連する索引の操作は可能です。読取り専用表で許可される操作および許可されない操作の完全なリストは、『Oracle Database管理者ガイド』を参照してください。

READ WRITEを指定すると、読取り専用表を読取り/書込みモードに戻すことができます。

REKEY encryption_spec

REKEY句を使用すると、新しい暗号化キーの生成または異なるアルゴリズム間の切替えができます。この操作は、表内のLOB列を含むすべての暗号化された列が再度暗号化された後にのみ戻されます。

DEFAULT COLLATION

この句を使用すると、表のデフォルト照合を変更できます。collation_nameには、有効な名前付き照合または疑似照合を指定します。

表の新しいデフォルト照合は、ALTER TABLE ADD文を使用して後から表に追加されたか、またはALTER TABLE MODIFY文を使用して非文字データ型から変更された文字データ型の列に割り当てられます。表の既存列の照合は変更されません。この句のセマンティクスの詳細は、「CREATE TABLE」の「DEFAULT COLLATION」句を参照してください。

[NO] ROW ARCHIVAL

この句を指定すると、tableの行アーカイブを有効化または無効化できます。

- tableの行アーカイブを有効にするには、ROW ARCHIVALを指定します。非表示列ORA_ARCHIVE_STATEが表に作成されます。テーブルにデータがすでに移入されている場合、表内の既存の各行でORA_ARCHIVE_STATEの値が0に設定されます。その後、UPDATE文を使用して、アーカイブする行のORA_ARCHIVE_STATEの値を1に設定できます。
- tableの行アーカイブを無効にするには、NO ROW ARCHIVALを指定します。非表示列ORA_ARCHIVE_STATEが表から削除されます。

[NO] ROW ARCHIVALの制限事項

この句には、次の制限事項が適用されます。

- ROW ARCHIVAL句は、ORA_ARCHIVE_STATEという名前の列がすでに含まれている表に対しては指定できません。
- NO ROW ARCHIVAL句は、SYSが所有する表に対しては指定できません。

関連項目:

- この句のセマンティクスの詳細は、「CREATE TABLE」の「ROW ARCHIVAL」句を参照してください。
- インデータベース・アーカイブの詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

attribute_clustering_clause

ADD attribute_clustering_clauseを使用して、属性クラスタリングの表を有効化します。attribute_clustering_clauseは、ALTER TABLEおよびCREATE TABLEと同じセマンティクスを持ちます。CREATE TABLEのドキュメントのattribute_clustering_clauseを参照してください。

MODIFY CLUSTERING

この句を使用して、ダイレクト・パス・インサート操作またはデータ移動操作中の表の属性クラスタリングを許可または禁止します。表の属性クラスタリングを有効化する必要があります。clustering_when句およびzonemap_clauseは、ALTER TABLEおよびCREATE TABLEと同じセマンティクスを持ちます。CREATE TABLEのドキュメントのclustering_when句およびzonemap_clauseを参照してください。

DROP CLUSTERING

この句を使用して、属性クラスタリングの表を無効化します。

CREATE TABLEまたはALTER TABLEのWITH MATERIALIZED ZONEMAP句を使用して表のゾーン・マップが作成された場合、ゾーン・マップが削除されます。CREATE MATERIALIZED ZONEMAP文を使用して表のゾーン・マップが作成された場合、ゾーン・マップが削除されません。

alter_iot_clauses

index_org_table_clause

この句を使用すると、既存の索引構成表の特性の一部を変更できます。索引構成表は主キーでデータをソートするため、主キーベースのアクセスおよび操作に最適です。「CREATE TABLE」のコンテキストで、「index_org_table_clause」を参照してください。

関連項目:

索引構成表の変更: 例

prefix_compression

prefix_compression句を使用して、表の接頭辞圧縮を有効化します。COMPRESSを指定すると、ブロックを再利用するために、索引構成表の主キー索引ブロックを空きブロックに結合できます(可能な場合)。この句はparallel_clauseとあわせて指定できます。NOCOMPRESSを指定して、表の接頭辞圧縮を無効化します。

PCTTHRESHOLD integer

「CREATE TABLE」の「PCTTHRESHOLD integer」を参照してください。

INCLUDING column_name

「CREATE TABLE」の「INCLUDING column_name」を参照してください。

overflow_attributes

overflow_attributesを使用すると、索引構成表に対して、変更するオーバーフロー・データ・セグメントの物理記憶域属性およびロギング属性を指定できます。この句に指定するパラメータの値は、オーバーフロー・データ・セグメントにのみ適用されます。

関連項目:

CREATE TABLE

add_overflow_clause

add_overflow_clauseを使用すると、指定した索引構成表にオーバーフロー・データ・セグメントを追加できます。また、この句を使用すると、エクステントを明示的に割り当てたり、既存のオーバーフロー・セグメントから未使用領域の割当てを解除することができます。

STORE IN tablespace句を使用すると、オーバーフロー・セグメント全体に対する表領域の記憶域を指定できます。PARTITION句を使用すると、パーティションによるセグメントに対する表領域の記憶域を指定できます。

パーティション化された索引構成表の場合、次の点に注意してください。

- PARTITIONを指定しない場合、それぞれのパーティションに自動的にオーバーフロー・セグメントが割り当てられます。

これらのセグメントの物理属性は表のレベルから継承されます。

- 1つ以上のパーティションに別々の物理属性を指定する場合は、その属性を表のすべてのパーティションに対して指定する必要があります。パーティションの名前を指定する必要はありませんが、パーティションが作成された順番で属性を指定する必要があります。

パーティションの順番を確認するには、USER_IND_PARTITIONSビューのPARTITION_NAMEおよびPARTITION_POSITION列を問い合せます。

TABLESPACEを指定していないパーティションがある場合、表に対して指定された表領域が使用されます。表レベルでTABLESPACEを指定していない場合は、そのパーティションの主キー索引セグメントの表領域が使用されます。

オーバーフロー属性の制限事項

segment_attributes_clause内では、次の制限事項があります。

- physical_attributes_clauseのOPTIMALパラメータを指定できません。
- この句を使用して、オーバーフロー・セグメントの表領域の記憶域を指定できません。非パーティション表の場合、ALTER TABLE ... MOVE ... OVERFLOWを使用して、セグメントを異なる表領域に移動します。パーティション表の場合、ALTER TABLE ... MODIFY DEFAULT ATTRIBUTES ... OVERFLOWを使用して、オーバーフロー・セグメントのデフォルト表領域を変更します。

tableがローカル管理表領域にある場合、表領域のセグメント属性の中にはデータベースによって自動的に管理されるものがあるため、制限事項が追加されます。

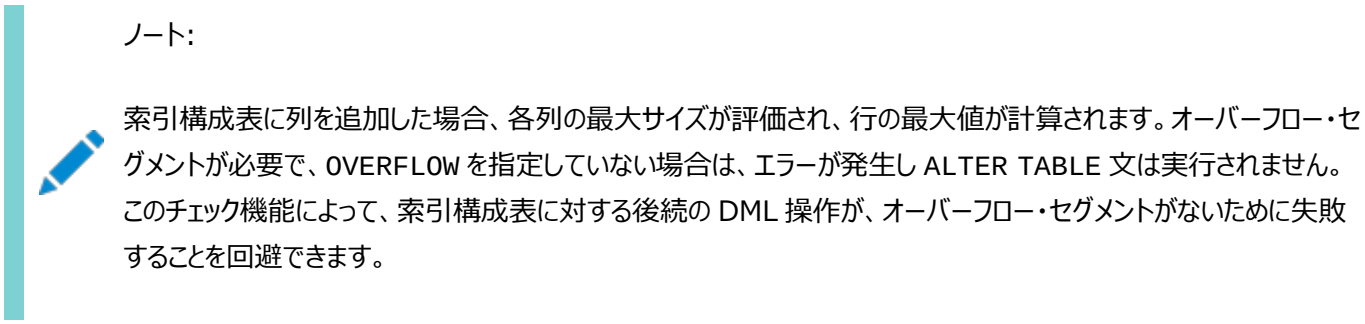
関連項目:

add_overflow_clause句の詳細は、「allocate_extent_clause」および「deallocate_unused_clause」を参照してください。

alter_overflow_clause

alter_overflow_clauseを使用すると、既存の索引構成表のオーバーフロー・セグメントの定義を変更できます。

add_overflow_clauseの制限事項は、alter_overflow_clauseにも適用されます。



ノート:

索引構成表に列を追加した場合、各列の最大サイズが評価され、行の最大値が計算されます。オーバーフロー・セグメントが必要で、OVERFLOW を指定していない場合は、エラーが発生し ALTER TABLE 文は実行されません。このチェック機能によって、索引構成表に対する後続の DML 操作が、オーバーフロー・セグメントがないために失敗することを回避できます。

alter_mapping_table_clauses

alter_mapping_table_clausesは、tableが索引構成されており、マッピング表を持つ場合にのみ有効です。

allocate_extent_clause

allocate_extent_clauseを使用すると、新しいエクステントを索引構成表のマッピング表の終わりに割り当てることがで

きます。この句の詳細は、「allocate_extent_clause」を参照してください。

deallocate_unused_clause

deallocate_unused_clauseを指定すると、索引構成表のマッピング表の終わりにある未使用領域の割当てを解除できます。この句の詳細は、「deallocate_unused_clause」を参照してください。

マッピング表またはそのパーティションに関する他のすべての属性は、自動的に管理されます。

COALESCE句

COALESCEを指定すると、ブロックを再利用するために、索引構成表の管理に使用される索引の索引ブロックの内容を空きブロックにマージできます(可能な場合)。この句とshrink_clauseの関係については、「shrink_clause」を参照してください。

alter_XMLSchema_clause

この句は、BINARY XML記憶域のXMLType表を変更する場合にのみ、alter_table_propertiesの一部として有効です。ALLOWおよびDISALLOW句の詳細は、「CREATE TABLE」の「XMLSchema_spec」を参照してください。

column_clauses

これらの句を指定すると、列を追加、削除または変更できます。

add_column_clause

add_column_clauseを使用すると、表に列を追加できます。

関連項目:

この句のキーワードとパラメータの詳細は、「CREATE TABLE」および「表の列の追加: 例」を参照してください。

column_definition

既存の表に列を追加するときのcolumn_definitionの要素の動作は、この項で特に述べられていないかぎり、新しい表の作成時に列を追加するときの動作と同じです。詳細は、「CREATE TABLE」の「column_definition」句を参照してください。

column_definitionの制限事項

SORTパラメータは、新しい表の作成時にのみ有効です。ALTER TABLE ... ADD文のcolumn_definitionでは、SORTを指定できません。

列を追加するときに、DEFAULT句を指定していないと、新しい列の各行の初期値はNULLになります。

パーティション化された索引構成表の各パーティションには、オーバーフロー・データ・セグメントを追加できます。

非パーティションおよびパーティション表にLOB列を追加できます。表、およびパーティションまたはサブパーティションのレベルでLOB記憶域を指定できます。

SELECT *構文を使用して、表からすべての列を選択するように指定した問合せを使用してビューを作成した場合、tableに列を追加しても、新しい列がビューに自動的に追加されることはありません。ビューに新しい列を追加する場合、CREATE VIEW文にOR REPLACE句を指定してビューを再作成してください。詳細は、「CREATE VIEW」を参照してください。

列の追加の制限事項

列の追加には、次の制限事項があります。

- LOB列またはINVISIBLEの列は、クラスタ表には追加できません。

- LOB列をハッシュ・パーティション表に追加する場合、新しいパーティションに対して指定できる属性は、TABLESPACEのみです。
- tableに行がある場合、DEFAULT句を指定しないかぎり、NOT NULL制約のある列を追加できません。
- 索引構成表にこの句を指定した場合、同じ文では他の句を指定できません。
- 重複表には列を追加できません。

DEFAULT

DEFAULT句を使用すると、新しい列にデフォルト値を指定したり、既存の列に新しいデフォルト値を指定することができます。後続のINSERT文で列に値を指定しない場合、この値が自動的に割り当てられます。

この式のデータ型は、列に指定したデータ型と一致している必要があります。列には、この式を保持できる大きさが必要です。

DEFAULT式には、リテラル引数、列の参照またはネストしたファンクションの起動を戻さない、任意のSQLファンクションを含めることができます。

このDEFAULT式には、順序疑似列のCURRVALとNEXTVALを含めることができます。ただし、順序が存在していて、その順序にアクセスするために必要な権限を所持している必要があります。それ以降に、DEFAULT式を使用して挿入を実行するユーザーには、表に対するINSERT権限と、順序に対するSELECT権限が必要になります。その後で順序が削除されると、それ以降のDEFAULT式を使用する挿入文でエラーが発生します。新しい列を表に追加すると、既存の各行に割り当てられるNEXTVALの順序は、非決定的になります。順序の所有者を指定して完全修飾(たとえば、SCOTT.SEQ1)していない場合、Oracle Databaseは、ALTER TABLE文を発行したユーザーを順序のデフォルトの所有者にします。たとえば、ユーザーMARYがSCOTT.TABLEに列を追加して、SEQ2のように完全修飾していない順序を参照すると、その列は順序MARY.SEQ2を使用するようになります。順序に対するシノニムは完全に名前解決され、完全修飾された順序としてデータ・ディクショナリに格納されます。これは、パブリック・シノニムとプライベート・シノニムに当てはまります。たとえば、ユーザーBETHが、パブリック・シノニムまたはプライベート・シノニムのSYN1を参照する列を追加したときに、そのシノニムがPETER.SEQ7を参照していると、その列はデフォルトとしてPETER.SEQ7を格納するようになります。

列にDEFAULT句を指定すると、デフォルト値はメタデータとして格納されますが、列自体にはデータは移入されません。ただし、デフォルト値が結果セットに戻されるように、新しい列を指定する後続の問合せは再書込みされます。この最適化された動作には、次の制限事項があります。

- 表にLOB列を持つことはできません。対象となる表は、索引構成化したり、一時表またはクラスタ化された表にすることはできません。また、キュー表、オブジェクト表またはマテリアライズド・ビューのコンテナ表にすることもできません。
- 表が仮想プライベート・データベース(VPD)ポリシーを持つ場合、ALTER TABLE ... ADD文を発行するユーザーにEXEMPT ACCESS POLICYシステム権限がないかぎり、最適化された動作は有効ではありません。
- 追加する列は暗号化できず、オブジェクト列、ネストした表の列またはLOB列にすることはできません。
- このDEFAULT式には、順序疑似列のCURRVALまたはNEXTVALを含めることができません。

前述の制限事項のために最適な動作が実行できない場合、Oracle Databaseは、新しく作成した列の各行をデフォルト値で更新します。この場合、データベースは表に定義されたUPDATEトリガーを起動しません。

デフォルト列値の制限事項

デフォルト列値には、次の制限事項があります。

- DEFAULT式に、PL/SQLファンクション、他の列、疑似列LEVEL、PRIORおよびROWNUMへの参照または完全に指定されていない日付定数は指定できません。

- 式には、スカラー副問合せ式を除くすべての書式を使用できます。

ON NULL

ON NULL句を指定すると、Oracle Databaseは、それ以降のINSERT文でNULLに評価される値を割り当てようとするときに、DEFAULTの列値を割り当てるようになります。

ON NULLを指定すると、NOT NULL制約と、NOT DEFERRABLE制約状態が暗黙的に指定されます。NOT NULLおよびNOT DEFERRABLEと競合する表内制約を指定すると、エラーが発生します。

関連項目:

デフォルト列値の指定: 例

identity_clause

identity_clauseはID列を追加する場合、ID列を作成する場合と同じセマンティクスを持ちます。詳細は、CREATE TABLEの「identity_clause」を参照してください。

新しいID列を追加すると、既存のすべての行は順序ジェネレータを使用して更新されます。既存の各行に割り当てられる値の順序は、非決定的になります。

identity_options

identity_options句を使用すると、順序ジェネレータを構成できます。identity_options句のパラメータは、CREATE SEQUENCE文と同じです。これらのパラメータと特性の詳細は、「CREATE SEQUENCE」を参照してください。identity_optionsに固有のSTART WITH LIMIT VALUEは例外であり、ALTER TABLE MODIFYでのみ使用できます。詳細は、「identity_options」を参照してください。

inline_constraint

inline_constraintを使用すると、新しい列に制約を追加できます。

inline_ref_constraint

この句を使用すると、新しいREF型の列を定義できます。制約の型の構文や制限などの詳細は、「constraint」を参照してください。

virtual_column_definition

virtual_column_definitionは列を追加する場合、列を作成する場合と同じセマンティクスを持ちます。

関連項目:

詳細は、「CREATE TABLE」の「virtual_column_definition」および「仮想表の列の追加: 例」を参照してください。

仮想列の追加の制限事項

仮想列のSQL式にOracle Data Redactionポリシーが定義されている列が関与するときには、仮想列は追加できません。

column_properties

column_propertiesの句を使用すると、オブジェクト型、ネストした表、VARRAYまたはLOB列の記憶特性を指定できます。

object_type_col_properties

この句は、新しいオブジェクト型の列または属性の追加時にのみ有効です。modify_column_clausesを使用すると、既存のオブジェクト型列のプロパティを変更できます。この句のセマンティクスは、特に指定がないかぎり、CREATE TABLEと同じです。

object_type_col_properties句を使用すると、新しいオブジェクト列、オブジェクト属性、コレクション列およびコレクション属性の要素に対する記憶特性を指定できます。

この句の詳細は、「CREATE TABLE」の「object_type_col_properties」を参照してください。

nested_table_col_properties

nested_table_col_properties句を使用すると、ネストした表に対して別の記憶特性を指定し、そのネストした表を索引構成表として定義できます。ネストした表の型を持つ列または列属性付きで表を作成する場合は、この句を挿入する必要があります。(この句の中で、親オブジェクト表に対する場合と同じ働きをする句は、ここでは繰り返されません。これらの句の詳細は、「CREATE TABLE」の句「nested_table_col_properties」を参照してください。

- nested_itemには、型がネストした表である列(または、ネストした表のオブジェクト型の最上位の属性)の名前を指定します。

  ネストした表がマルチレベル・コレクションで、内部のネストした表には名前が割り当てられていない場合、nested_item名のかわりにCOLUMN_VALUEを指定します。

- storage_tableには、nested_itemの行を含む表の名前を指定します。記憶表は、親表と同じスキーマ、および親表と同じ表領域内に作成されます。

ネストした表の列のプロパティの制限事項

ネストした表の列のプロパティには、次の制限事項があります。

- parallel_clauseは指定できません。
- physical_properties句の一部としてCLUSTERを指定できません。

関連項目:

「ネストした表: 例」

varray_col_properties

varray_col_properties句を使用すると、VARRAY型のデータが格納されているLOBに対して、別の記憶特性を指定できます。この句を指定する場合、表内に格納できるほど小さい値でも、VARRAYは必ずLOBに格納されます。varray_itemがマルチレベル・コレクションの場合、varray_item内にネストされたすべてのコレクション項目は、常にvarray_itemと同じLOBに格納されます。

VARRAY列のプロパティの制限事項

VARRAY列のLOB_parametersの一部としてTABLESPACEを指定することはできません。VARRAYに対するLOB表領域のデフォルトは、表を含む表領域になります。

out_of_line_part_storage

この句を使用すると、新しく追加した列に対し、パーティション表内のパーティションまたはサブパーティションごとに記憶域属性を指定できます。この句で指定しないパーティションまたはサブパーティションについては、新しい列の記憶域属性は、表レベルのnested_table_col_propertiesで指定したものと同じになります。

LOB_storage_clause

LOB_storage_clauseを使用すると、新しく追加したLOB列、LOBパーティション、LOBサブパーティションまたはLONG列からLOB列への変換時のLOB記憶特性を指定できます。この句では、既存のLOBを変更できません。かわりに、modify_LOB_storage_clauseを使用する必要があります。

この項で特に指定がない場合は、LOB_storage_clauseおよびmodify_LOB_storage_clause内のすべてのLOBパラメータは、ALTER TABLE文で、CREATE TABLE文と同じセマンティクスを持ちます。この句の詳細は、「CREATE TABLE」の「LOB_storage_clause」を参照してください。

LOBパラメータの制限事項

ハッシュ・パーティションまたはハッシュ・サブパーティションに指定できるLOB_parametersのパラメータは、TABLESPACEのみです。

CACHE READS句

新しいLOB列の追加時には、作成時にLOB列を定義する場合と同様に、CACHE READSでロギング属性を指定できます。この句の詳細は、CREATE TABLEの句「CACHE READS」を参照してください。

ENABLE|DISABLE STORAGE IN ROW

STORAGE IN ROWは、一度設定すると変更できません。したがって、この句はmodify_col_properties句の一部には指定できません。ただし、新しい列を追加するとき(add_column_clause)、または表を移動するとき(move_table_clause)に、この設定を変更できます。この句の詳細は、「CREATE TABLE」の句「ENABLE STORAGE IN ROW」を参照してください。

CHUNK integer

modify_col_properties句を使用して、CHUNKの値を設定後に変更することはできません。作成後の列に異なるCHUNK値が必要な場合は、ALTER TABLE … MOVEを使用します。詳細は、「CREATE TABLE」の句「CHUNK integer」を参照してください。

RETENTION

データベースが自動UNDOモードで稼働している場合、BasicFiles LOBに対してPCTVERSIONではなくRETENTIONを指定することで、このLOBの古いバージョンを保持できます。この句によって、PCTVERSIONのこれまでの設定が上書きされます。このパラメータの詳細は、「CREATE TABLE」の句「LOB_retention_clause」を参照してください。

FREEPOOLS integer

データベースが自動UNDOモードで稼働している場合、BasicFiles LOBに対してこの句を使用するとLOBの空きリスト・グループ数を指定できます。この句によって、FREELIST GROUPSのこれまでの設定が上書きされます。このパラメータの詳細は、「CREATE TABLE」の句「FREEPOOLS integer」を参照してください。SecureFiles LOBの場合は、データベースはこのパラメータを無視します。

LOB_partition_storage

1つのALTER TABLE文で指定できるLOB_partition_storage句のリストは1つのみであり、すべてのLOB_storage_clausesおよびvarray_col_properties句はLOB_partition_storage句のリストの前に指定する必要があります。制限事項を含むこの句の詳細は、「CREATE TABLE」の句「LOB_partition_storage」を参照してください。

XMLType_column_properties

この句の詳細は、「CREATE TABLE」の句「XMLType_column_properties」を参照してください。

関連項目:

- LOB_segnameおよびLOB_parameters句の詳細は、「LOB_storage_clause」を参照してください
- オブジェクト・リレーショナル表におけるXMLType列の例は、「XMLType列の例」を参照してください。XMLスキーマの作成例は、「SQL文でのXMLの使用方法」を参照してください。
- XMLType列と表およびXMLスキーマの作成の詳細は、『Oracle XML DB開発者ガイド』を参照してください。

modify_column_clauses

modify_column_clausesを使用すると、既存の列のプロパティ、既存の列の可視性、または既存のオブジェクト型列の代替性を変更できます。

関連項目:

表の列の変更: 例

modify_col_properties

この句を使用すると、列のプロパティを変更できます。この句で省略した列定義のオプション部分(データ型、デフォルト値、制約)は、変更されません。

datatype

列のすべての行がNULLの場合、列のデータ型を変更できます。ただし、マテリアライズド・ビューのコンテナ表にある列のデータ型を変更した場合、それに対応するマテリアライズド・ビューが無効になります。

参照整合性制約の外部キーの一部として、文で列が指定されている場合のみ、データ型を省略できます。参照整合性制約の参照キーに対応する列のデータ型が、その列に自動的に割り当てられます。

すべての行にNULL値が存在するどうかにかかわらず、文字型またはRAW型の列のサイズ、または数値型の列の精度は、いつでも大きくすることができます。変更対象のデータに変更の必要がない場合は、列のデータ型のサイズを削減できます。データベースは、既存のデータをスキャンして、新しい長さ制限を超過するデータが存在する場合はエラーを返します。

VARCHAR2、NVARCHAR2、またはRAWの列のサイズが4,000バイトを超えるように拡大すると、Oracle Databaseはインプレースで長さの拡張を実行し、表内記憶域を外部LOB記憶域に移行しなくなります。これにより、大きい表の連続移行が可能になり、特に移行後に拡張データ型を活用できます。ただし、列の表内記憶域は、CREATE TABLE ... AS SELECT、エクスポート、インポート、オンライン再定義などの表の再編成操作時には保存されません。拡張データ型列の新しい表外記憶域に移行する場合は、前述の方法のいずれかを使用して表を再作成する必要があります。列の表内記憶域は、表またはパーティションの移動操作時(ALTER TABLE MOVE [[SUB]PARTITION]など)およびパーティションのメンテナンス操作時(ALTER TABLE SPLIT [SUB]PARTITION、ALTER TABLE MERGE [SUB]PARTITIONS、ALTER TABLE COALESCE [SUB]PARTITIONSなど)に保存されます。


ノート:

VARCHAR2、NVARCHAR2、または RAW の列のサイズは、次の理由から、4,000 バイトを超える過度の拡大は避けるようにしてください。

- 行の変更が発生する場合があります。
- 表内に格納されているデータは、列が選択されているかどうかにかかわらず、全体を読み込むことが必要になります。そのため、表内に格納される拡張データ型の列は、パフォーマンスに悪影響を与えることがあります。

変更対象のデータに変更の必要がない場合は、列のデータ型のサイズを削減できます。データベースは、既存のデータをスキャンして、新しい長さ制限を超過するデータが存在する場合はエラーを返します。

DATE列をTIMESTAMPまたはTIMESTAMP WITH LOCAL TIME ZONE列に、TIMESTAMPまたはTIMESTAMP WITH LOCAL TIME ZONE列をDATE列に変更できます。以下のルールが適用されます。

- TIMESTAMPまたはTIMESTAMP WITH LOCAL TIME ZONE列をDATE列に変更する場合、秒に0以外の小数部が含まれる各列の値が最も近い秒に丸めて更新されます。そのような値の更新時に60以上の分フィールドがある場合(夏時間の規則が切り換えられた場合に境界で発生)、分フィールドから60を引いてフィールドが更新されます。
- TIMESTAMP WITH LOCAL TIME ZONE列をDATE列に変更しても、列の値はデータベースのタイムゾーンで示したローカル時間を示します。ただし、データベースのタイムゾーンは値とは関連付けされなくなります。SQL*Plusで問い合せた場合、値がセッションのタイムゾーンに自動で調整されることはなくなります。列の値を処理するアプリケーションが特定のタイムゾーンで値を解析することになります。

表が空の場合、日時列または期間列の先行フィールドまたは秒の小数部を増やすことも減らすこともできます。表が空でない場合、日時列または期間列の先行フィールドまたは秒の小数部を増やすことのみできます。

TO_LOBファンクションを使用すると、LONG列をCLOB列またはNCLOB列に、LONG RAW列をBLOB列に変更できます。ただし、PL/SQLパッケージ内からはTO_LOBファンクションを使用できません。かわりに、TO_CLOB (文字)ファンクションまたはTO_BLOB (RAW)ファンクションを使用してください。

- 変更されたLOB列は、元のLONG列で定義されたすべての制約およびトリガーを継承します。いずれかの制約を変更する場合、後続のALTER TABLE文で変更する必要があります。
- ドメイン索引がLONG列で定義されている場合、列をLOBに変更する前に削除する必要があります。
- 変更後、表のすべての列にある他のすべての索引を再構築する必要があります。

TO_CLOB (文字)ファンクションを使用すると、NCLOB列をCLOB列に変換できます。

関連項目:

- LONGからLOBへの移行の詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。
- 索引の削除および再構築の詳細は、「ALTER INDEX」を参照してください。

CHARおよびVARCHAR2列の場合、CHAR(元々バイトで指定されていた列に対するキャラクタ・セマンティクス)またはBYTE(元々文字で指定されていた列に対するバイト・セマンティクス)を指定すると、長さセマンティクスを変更できます。既存の列の長さセマンティクスを確認するには、ALL_TAB_COLUMNS、USER_TAB_COLUMNSまたはDBA_TAB_COLUMNSデータ・ディクショナリ・ビューのCHAR_USED列を問い合せます。

関連項目:

- バイト・セマンティクスおよびキャラクタ・セマンティクスの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。
- データ・ディクショナリ・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

データ型の作成時または変更時に、ユーザー定義のデータ型を永続不可として指定できます。永続不可型のインスタンスは、ディスク上で保持することはできません。永続不可型として宣言されるユーザー定義のデータ型の詳細は、「CREATE TYPE」を参照してください。

COLLATE

この句を使用すると、列のデータ・バインドされた照合を設定または変更できます。`column_collation_name`には、有効な名前付き照合または疑似照合を指定します。データ・バインドされた照合の詳細は、「CREATE TABLE」の「DEFAULT COLLATION」句を参照してください。

列照合の変更の制限事項

列照合の変更には、次の制限事項があります。

- 列が索引キーに属する場合、その照合は次の場合にのみ変更できます。
  - 照合BINARY、USING_NLS_COMP、USING_NLS_SORTおよびUSING_NLS_SORT_CSの間
  - 照合BINARY_CIおよびUSING_NLS_SORT_CIの間
  - 照合BINARY_AIおよびUSING_NLS_SORT_AIの間
- 列が、レンジ・パーティション・キーまたはリスト・パーティション・キーに属しているか、ビットマップ結合索引により参照されているか、索引構成表の主キーに属しているか、Oracle Text索引を含むドメイン索引のキーに属している場合、その照合はBINARY、USING_NLS_COMP、USING_NLS_SORTおよびUSING_NLS_SORT_CSの各照合間でのみ変更できます。
- 列が属性クラスタリング・キーに属している場合、その照合はBINARYおよびUSING_NLS_COMP照合間でのみ変更できます。

関連項目:

詳細な大/小文字の区別に関する列の照合の変更: 例

identity_clause

`identity_clause`を使用すると、ID列のプロパティを変更できます。この句は、ID列ではない列に対しては指定できません。ALWAYSまたはBY DEFAULTを省略すると、現在の生成タイプが保持されます。ALWAYSおよびBY DEFAULTの詳細は、CREATE TABLEの「identity_clause」を参照してください。

identity_options

`identity_options`句を使用すると、順序ジェネレータを構成できます。`identity_options`句のパラメータは、CREATE SEQUENCE文と同じです。これらのパラメータと特性の詳細は、「CREATE SEQUENCE」を参照してください。例外は次のとおりです。

- `identity_options`に固有のSTART WITH LIMIT VALUEは、ALTER TABLE MODIFYでのみ使用できま

す。START WITH LIMIT VALUEを指定すると、Oracle Databaseは表をロックして、表内で最大のID列の値(増加する順序の場合)、または最小のID列の値(減少する順序の場合)を検出して、その値を順序ジェネレータの最高水位標として割り当てます。順序ジェネレータが返す次の値は、増加する順序の場合は最高水位標 + INCREMENT BY integerになり、減少する順序の場合は最高水位標 - INCREMENT BY integerになります。

- START WITHの値を変更すると、この句内のその他すべてのパラメータにデフォルトの値が使用されるようになります。ただし、別の値が指定されている場合を除きます。

DROP IDENTITY

この句を使用すると、列からIDプロパティを削除できます。このプロパティには、シーケンス・ジェネレータや、NOT NULL制約、NOT DEFERRABLE制約などが含まれます。既存の行のID列の値には影響しません。

ENCRYPT encryption_spec | DECRYPT

この句は、暗号化された列の復号化や暗号化されていない列の暗号化を行うとき、および暗号化された列の整合性アルゴリズムやSALTオプションを変更するときに使用します。

encryption_specを指定して既存の列を暗号化する場合は、同じ表内で暗号化されている他の列の暗号化仕様と一致させることが必要です。encryption_specの詳細や制限事項は、「CREATE TABLE」の句「encryption_spec」を参照してください。

マテリアライズド・ビュー・ログが表に対して定義されている場合は、この句内で暗号化または復号化される列が、マテリアライズド・ビュー・ログ内でも暗号化または復号化されます。

ENCRYPT encryption_spec | DECRYPTの制限事項

この句には、次の制限事項があります。

- 新規または既存の列がLOB列の場合は、列をSecureFiles LOBとして格納する必要があり、SALTオプションは指定できません。
- 暗号化または復号化する列に対して、UPDATE文のファイングレイン監査ポリシーが有効化されていてはなりません。ただし、ファイングレイン監査ポリシーをいったん無効にしてその列を暗号化または復号化し、その後でファイングレイン監査ポリシーを有効にすることは可能です。

関連項目:

「データの暗号化: 例」

inline_constraint

この句を使用すると、変更対象の列に制約を追加できます。既存の列に設定されている既存の制約の状態を変更する場合は、constraint_clausesを使用します。

LOB_storage_clause

LOB_storage_clauseは、LONG列をLOB列に変換する場合のみ、modify_col_properties内で使用できます。この場合のみ、LOB_storage_clauseを使用して列に対するLOB記憶域を指定できます。ただし、指定できるのは、1つの列(LOB_item)のみです。LOB_storage_clauseで省略したすべての属性には、デフォルトのLOB記憶域属性が適用されます。

alter_XMLSchema_clause

この句は、BINARY XML記憶域のXMLType表のmodify_col_properties内でのみ有効です。ALLOWおよびDISALLOW句の詳細は、「CREATE TABLE」の「XMLSchema_spec」を参照してください。

列プロパティの変更の制限事項

列プロパティの変更には、次の制限事項があります。

- LOB列のデータ型は変更できません。
- 列にドメイン索引が定義されている場合は、表の列を変更できません。最初にドメイン索引を削除してから列を変更する必要があります。
- 表または索引のパーティション化キーまたはサブパーティション化キーの一部である列の長さまたはデータ型は、変更できません。
- CHAR型の列をVARCHAR2(またはVARCHAR)型に変更またはVARCHAR2(またはVARCHAR)型の列をCHAR型に変更できるのは、BLANK_TRIMMING初期化パラメータがTRUEに設定され、列のサイズが同じまたは増加する場合のみです。BLANK_TRIMMING初期化パラメータがTRUEに設定されている場合、列のサイズを切捨て後のデータの最大値以上の値まで減らすこともできます。
- 表がクラスタの一部である場合は、LONGまたはLONG RAW列をLOBに変更できません。LONGまたはLONG RAW列をLOBに変更する場合は、同じALTER TABLE文では、DEFAULT句およびLOB_storage_clause以外は指定できません。
- LONGまたはLONG RAW列をLOBに変更する場合のみ、LOB_storage_clauseをmodify_col_propertiesの一部として指定できます。
- 索引構成表に対してROWIDデータ型の列は指定できませんが、UROWID型の列は指定できます。
- 列のデータ型をREFに変更できません。
- 重複表の列のプロパティは変更できません。

関連項目:

マテリアライズド・ビューの再検証の詳細は、「ALTER MATERIALIZED VIEW」を参照してください。

modify_virtcol_properties

この句を使用すると、仮想列を次の方法で変更できます。

- COLLATE句を指定すると、仮想列のデータ・バインドされた照合を設定または変更できます。column_collation_nameには、有効な名前付き照合または疑似照合を指定します。データ・バインドされた照合の詳細は、「CREATE TABLE」の「DEFAULT COLLATION」句を参照してください。
- 仮想列がエディショニングされたPL/SQLファンクションを参照している場合、その仮想列の評価エディションまたは使用禁止エディションを変更できます。仮想列を変更する場合のevaluation_edition_clauseおよびunusable_editions_clauseのセマンティクスは、仮想列を作成する場合と同じです。詳細は、「CREATE TABLE」の「evaluation_edition_clause」および「unusable_editions_clause」を参照してください。

仮想列の変更の制限事項

仮想列の変更には、次の制限事項が適用されます。

- COLLATE句の指定による仮想列のデータ・バインドされた照合の設定または変更には、「列照合の変更の制限事

項」に示されている制限事項が適用されます。

- 索引が仮想列で定義されている場合に、その評価エディションまたは使用禁止エディションを変更すると、データベースによってその仮想列のすべての索引が無効化されます。仮想列の他のプロパティを変更しようとすると、エラーが発生します。

modify_col_visibility

この句を使用すると、columnの可視性を変更できます。詳細は、「CREATE TABLE」の「VISIBLE | INVISIBLE」を参照してください。

列の可視性の変更の制限事項

SYSが所有している表で、VISIBLE列をINVISIBLEに変更することはできません。

modify_col_substitutable

この句を使用すると、既存のオブジェクト型列の代替性を設定または変更できます。

FORCEキーワードを指定すると、型ID情報またはサブタイプ属性に関するデータが含まれる非表示列が削除されます。オブジェクト型の列または属性がFINALでない場合、FORCEを指定する必要があります。

列の代替性の変更の制限事項

列の代替性の変更には、次の制限事項があります。

- ALTER TABLE文には、この句を一度のみ指定できます。
- オブジェクト表自体の代替性が設定されている場合、オブジェクト表の列の代替性は変更できません。
- この句は、列がIS OF TYPE構文を使用して作成または追加されたものである場合は指定できません(この構文を使用すると、オブジェクト列や属性で使用できるサブタイプの範囲が、特定のサブタイプに限定されます)。IS OF TYPE構文の詳細は、CREATE TABLEの「substitutable_column_clause」を参照してください。
- 列の属性にネストしたオブジェクト型(FINAL以外)が含まれる場合、FORCEを指定しても、VARRAY列をNOT SUBSTITUTABLEに変更できません。

drop_column_clause

drop_column_clauseを使用すると、不要になった列を削除したり、将来、システム・リソースへの要求が少なくなったときに削除するように列にマークを付けることによって、データベースの領域を解放できます。

- ネストした表の列を削除すると、その記憶表も削除されます。
- LOB列を削除すると、LOBデータおよび対応するLOB索引セグメントも削除されます。
- BFILE列を削除すると、その列に格納されたロケータのみ削除され、ロケータによって参照されるファイルは削除されません。
- INCLUDING列として定義した列を削除(または未使用とマーク)すると、この列の直前に格納された列が新しいINCLUDING列になります。

SET UNUSED句

SET UNUSEDを使用すると、1つ以上の列が未使用としてマーク付けされます。内部ヒープ構成表の場合は、この句を指定しても、対象の列が表の各行から実際に削除されることはありません。その列に使用されているディスク領域がリストアされることはありません。したがって、応答時間はDROP句を実行したときよりも短縮されます。

外部表内の列に対してこの句を指定すると、句は透過的にALTER TABLE ... DROP COLUMN文に変換されます。その理由は、外部表に対する操作はメタデータのみの操作であるため、2つのコマンドのパフォーマンスに違いがないためです。

UNUSEDのマークが付いた列を持つすべての表は、データ・ディクショナリ・ビューUSER_UNUSED_COL_TABS、DBA_UNUSED_COL_TABSおよびALL_UNUSED_COL_TABSで参照できます。

関連項目:

データ・ディクショナリ・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

未使用列のデータは表の行に残っていますが、この列は削除されたものとして扱われます。UNUSEDのマークが付けられた列にはアクセスできなくなります。SELECT *問合せでも、未使用列からデータを取り出すことはできません。また、UNUSEDのマークが付けられた列の名前および型は、DESCRIBEコマンドでは表示されず、未使用列と同じ名前の新しい列を表に追加できます。

ノート:

これらの列を実際に削除するまでは、表当たり 1000 列の制限に対して、これらの列もカウント対象になります。ただし、すべての DDL 文と同様に、この句の結果をロールバックすることはできません。SET UNUSED 列を取り出すために、対応する SET USED を発行することはできません。1000 列の制限の詳細は、「CREATE TABLE」を参照してください。

また、LONG 列に UNUSED のマークを付けた場合、この未使用の LONG 列を実際に削除しないかぎり、その表には別の LONG 列を追加できません。

ONLINE

ONLINEを指定すると、表に対するDML操作を許可しながら、列にUNUSEDのマークを付けることを指定できます。

列へのUNUSEDのマーク付けの制限事項

SET UNUSED句には、次の制限事項が適用されます。

- ONLINE句は、DEFERRABLE制約で列をUNUSEDとしてマークする場合は指定できません。
- SYSが所有する表の列にUNUSEDのマークを付けることはできません。

DROP句

DROPを指定すると、表のそれぞれの行から、対象となる列に関連付けられた列記述子およびデータを削除できます。特定の列を明示的に削除した場合、対象の表でUNUSEDのマークが付いている列もすべて同時に削除されます。

列データを削除した場合、次のものが削除されます。

- 対象の列に定義されているすべての索引。
- 対象の列を参照しているすべての制約。
- 対象の列に統計タイプが関連付けられている場合、FORCEオプションによって、関連付けは解除され、その統計タイプを使用して収集したすべての統計情報は削除されます。

ノート:

対象の列が対象でない列の親キーである場合または CHECK 制約が対象である列と対象でない列の両方を参照している場合は、Oracle Database はエラーを戻し、CASCADE CONSTRAINTS 句を指定しないかぎり、列を削除しません。この句を指定した場合、対象である列を参照しているすべての制約が削除されます。

関連項目:

統計タイプの関連付けの解除方法の詳細は、「DISASSOCIATE STATISTICS」を参照してください。

DROP UNUSED COLUMNS句

DROP UNUSED COLUMNSを指定すると、未使用とマークされているすべての列を表から削除できます。表の未使用の列からディスク領域を回収する場合に、この文を使用します。表に未使用の列がない場合でも、エラーは戻されません。

column

未使用として設定または削除する1つ以上の列を指定します。列を1つのみ指定する場合にかぎり、COLUMNキーワードを使用します。列リストを指定する場合、リストには重複する列を指定できません。

CASCADE CONSTRAINTS

CASCADE CONSTRAINTSを指定すると、削除する列に定義されている主キーおよび一意キーを参照する外部キー制約をすべて削除したり、削除する列に定義されているすべての複数列制約を削除することができます。他の表の列、または対象である表の他の列が参照している制約がある場合は、CASCADE CONSTRAINTSを指定する必要があります。CASCADE CONSTRAINTSを指定しない場合、その文は異常終了し、エラーが戻されます。

INVALIDATE

INVALIDATEキーワードはオプションです。ビュー、トリガー、ストアド・プログラム・ユニットなどのすべての依存オブジェクトが自動的に無効になります。オブジェクトの無効化は再帰的プロセスです。したがって、すべての直接的な依存オブジェクトおよび間接的な依存オブジェクトが無効になります。ただし、データベースでは、リモート依存性をローカル依存性と別に管理しているため、ローカル依存性のみが無効になります。

この文によって無効となったオブジェクトは、次に参照された際に自動的に再検証されます。オブジェクトを参照する前に、オブジェクトに存在するエラーは、すべて修正しておく必要があります。

関連項目:

依存性の詳細は、『Oracle Database概要』を参照してください。

CHECKPOINT

CHECKPOINTを指定すると、DROP COLUMN操作でinteger行が処理された後にチェックポイントが適用されます。integerはオプションであり、1以上である必要があります。integerが表の行数より大きい場合は、すべての行が処理された後にチェックポイントが適用されます。integerを指定しない場合は、デフォルトの512に設定されます。チェックポイントを適用すると、DROP COLUMN操作中に蓄積されるUNDOログの量が削減されるため、UNDO領域の不足を回避できます。ただし、この文が中断されたときにすでにチェックポイントが適用されていた場合は、表は使用禁止の状態のままになります。使用禁止状態の表に対して実行可能な操作は、DROP TABLE、TRUNCATE TABLEおよびALTER TABLE DROP ... COLUMNS CONTINUE(後述)のみです。

この句は列データを削除しないため、SET UNUSEDと同時に使用できません。

DROP COLUMNS CONTINUE句

DROP COLUMNS CONTINUEを指定すると、中断されたところから列削除操作を続行できます。表が無効な状態にあるときにこの文を発行すると、エラーになります。

列の削除の制限事項

列の削除には、次の制限事項があります。

- この句の各部分は、文の中で1回のみ指定でき、他のALTER TABLE句と同時に使用することはできません。たとえば、次のような文は許可されません。

```
ALTER TABLE t1 DROP COLUMN f1 DROP (f2);
ALTER TABLE t1 DROP COLUMN f1 SET UNUSED (f2);
ALTER TABLE t1 DROP (f1) ADD (f2 NUMBER);
ALTER TABLE t1 SET UNUSED (f3)
   ADD (CONSTRAINT ck1 CHECK (f2 > 0));
```

- オブジェクト型の列は、エンティティとしてのみ削除できます。オブジェクト型の列から属性を削除するには、ALTER TYPE ... DROP ATTRIBUTE文をCASCADE INCLUDING TABLE DATA句とともに使用します。属性の削除は、すべての依存オブジェクトに影響することに注意してください。詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。
- 索引構成表からは、主キー列でない場合にかぎり、列を削除できます。索引構成表の主キー制約は削除できないため、CASCADE CONSTRAINTSを指定しても主キー列は削除できません。
- 削除した列または未使用の列を含む表をエクスポートできます。ただし、エクスポート・ファイルに指定されたすべての列が表に存在する(これらの列のいずれも削除または未使用のマークを付けられていない)場合のみ、その表をインポートできます。そうでない場合は、エラーが戻ります。
- COMPRESS BASICを使用する表の列を未使用として設定できますが、その列を削除することはできません。ただし、drop_column_clauseのすべての句は、ROW STORE COMPRESS ADVANCEDを使用する表に対して有効です。詳細は、「table_compression」のセマンティクスを参照してください。
- ドメイン索引が構築されている列は削除できません。
- REF列からはSCOPE表制約およびWITH ROWID制約を削除できません。
- 次のものは、この句を使用して削除できません。
    - 疑似列、クラスタ列またはパーティション列。パーティションが作成されたすべての表領域がオンラインで読取り/書込みモードである場合、パーティション表から非パーティション列を削除できます。
    - ネストした表の列、オブジェクト表の列、重複表の列またはSYSが所有する表の列。

関連項目:

列の削除: 例

add_period_clause

add_period_clauseを使用すると、有効な時間ディメンションをtableに追加できます。

ALTER TABLEのperiod_definition句は、CREATE TABLEと同じセマンティクスを持ちます。ただし、次の例外と追加

事項があります。

- valid_time_columnがtableにすでに存在していてはいけません。
- start_time_columnとend_time_columnを指定する場合、これらの列がtableにすでに存在する必要があります。または、これらの各列に対してadd_column_clauseを指定する必要があります。
- start_time_columnとend_time_columnを指定した場合で、これらの列がtableにすでに存在し、データが移入されているときは、この両方の列の値がNULLでないすべての行で、start_time_columnの値がend_time_columnの値より前である必要があります。

関連項目:

この句のセマンティクスの詳細は、CREATE TABLE period_definitionを参照してください。

drop_period_clause

drop_period_clauseを使用すると、有効な時間ディメンションをtableから削除できます。

valid_time_columnには、削除する有効な時間ディメンションの名前を指定します。

この句には次のような効果があります。

- valid_time_columnがtableから削除されます。
- CREATE TABLE ... period_definitionまたはALTER TABLE ... add_period_clauseを使用して有効な時間ディメンションを作成したときに開始時間列と終了時間列が自動的に作成された場合は、これらの列が削除されます。それ以外の場合、これらの列はtableに残り、通常の表の列に戻ります。

関連項目:

valid_time_column、開始時間列、終了時間列の詳細は、CREATE TABLE period_definitionを参照してください。

rename_column_clause

rename_column_clauseを使用すると、tableの列名を変更できます。新しい列には、table内の他の列と同じ名前を指定しないでください。

列名を変更すると、依存オブジェクトは次のように処理されます。

- 名前が変更された列に依存するファンクション索引およびCHECK制約は、引き続き有効です。
- 依存するビュー、トリガー、ファンクション、プロシージャおよびパッケージは無効になります。これらのオブジェクトが次回アクセスされたときに、Oracle Databaseによって再有効化が試行されますが、再有効化に失敗した場合は、管理者がそのオブジェクトを変更して新しい列名を指定する必要があります。
- 名前を変更する列にドメイン索引を定義している場合、ODCIIndexAlterメソッドがRENAMEオプション付きで起動されます。この操作によって、索引タイプ・メタデータと実表の間の対応が確立されます。

列名の変更の制限事項

列名の変更には、次の制限事項があります。

- 同じ文の中で、この句を他のcolumn_clausesと同時に使用することはできません。

- 結合索引の定義に使用される列名は変更できません。列名を変更する場合、索引を削除し、列名を変更してから、索引を再作成する必要があります。
- 重複表の列の名前は変更できません。

関連項目:

列名の変更：例

modify_collection_retrieval

`modify_collection_retrieval`句を使用すると、データベースからコレクション型の項目が取り出されたときの戻り値を変更できます。

collection_item

型がネストされた表またはVARRAYである、列修飾属性の名前を指定します。

RETURN AS

問合せの結果として何を戻り値とするかを指定します。

- LOCATORは、ネストした表に対して一意のロケータを戻すことを指定します。
- VALUEは、ネストした表のコピーをそのまま戻すことを指定します。

関連項目:

「コレクションの取出し：例」

modify_LOB_storage_clause

`modify_LOB_storage_clause`を使用すると、`LOB_item`の物理属性を変更できます。各`modify_LOB_storage_clause`に対して、`LOB_item`を1つのみ指定できます。

以降の項では、modify_LOB_parametersに固有のパラメータのセマンティクスについて説明します。この項で特に指定がない場合は、残りのLOBパラメータのセマンティクスは、表の作成時と表の変更時で同じです。詳細は、この項の最後にある制限と、「CREATE TABLE」の句「LOB_storage_parameters」を参照してください。

ノート:

- LOB 記憶域に変更を加えるには、ALTER TABLE 文を使用する方法と、DBMS_REDEFINITION パッケージを使用したオンライン再定義による方法があります。LOB 暗号化、圧縮または重複除外を作成時に有効にしなかった場合は、作成後にオンライン再定義を使用してこれらを有効にすることをお薦めします(この 3 つのパラメータの変更に対しては、この方法の方がディスク領域の効率の点で優れているため)。DBMS_REDEFINITION の詳細は、『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』を参照してください。
- LOB の記憶域の種類を変換することはできません。かわりに、オンライン再定義またはパーティション交換を使用して、SecureFile または BasicFile を移行する必要があります。

PCTVERSION integer

この句の詳細は、「CREATE TABLE」の句「PCTVERSION integer」を参照してください。

LOB_retention_clause

データベースが自動UNDOモードで稼働している場合、旧バージョンのLOBを保持するには、PCTVERSIONではなくRETENTIONを指定します。この句によって、PCTVERSIONのこれまでの設定が上書きされます。

FREEPOOLS integer

データベースが自動UNDOモードで稼働している場合、BasicFiles LOBに対してこの句を使用するとLOBの空きリスト・グループ数を指定できます。この句によって、FREELIST GROUPSのこれまでの設定が上書きされます。このパラメータの詳細は、「CREATE TABLE」の句「FREEPOOLS integer」を参照してください。SecureFiles LOBの場合は、データベースはこのパラメータを無視します。

REBUILD FREEPOOLS

この句はBasicFiles LOBにのみ適用され、SecureFiles LOBには適用されません。REBUILD FREEPOOLS句を指定すると、LOB列から古いバージョンのデータがすべて削除されます。この句が役に立つのは、1つのLOBセグメント内に保持されている古いバージョンの領域をすべて削除する場合です(その領域が解放されて、すぐに新しいLOBデータに使用できるようになります)。

LOB_deduplicate_clause

この句は、SecureFiles LOBに対してのみ有効です。KEEP_DUPLICATESは、LOB重複除外を無効にします。DEDUPLICATEは、LOB重複除外を有効にします。セグメント内のすべてのLOBが読み取られ、一致するLOBがある場合は重複が除外されてから戻されます。

LOB_compression_clause

この句は、SecureFiles LOBに対してのみ有効です。COMPRESSは、セグメント内のすべてのLOBを圧縮して戻します。NOCOMPRESSは、セグメント内のすべてのLOBを圧縮解除して戻します。

ENCRYPT | DECRYPT

LOB暗号化は、一般に列の暗号化と同じセマンティクスを持ちます。詳細は、「ENCRYPT encryption_spec | DECRYPT」を参照してください。

CACHE, NOCACHE, CACHE READS

CACHEまたはNOCACHEからCACHE READSへ、またはCACHE READSからCACHEまたはNOCACHEへLOB列を変更するときに、ロギング属性を変更できます。LOGGINGまたはNOLOGGINGを指定しない場合、LOB列の現行ロギング属性がデフォルトになります。CACHE、NOCACHEまたはCACHE READSを指定しない場合、Oracle Databaseは、LOB属性の既存の値を保持します。

LOB記憶域の変更の制限事項

LOB記憶域の変更には、次の制限事項があります。

- LOB記憶域属性を変更する場合、storage_clauseのINITIALパラメータの値は変更できません。
- 同じ文でallocate_extent_clauseとdeallocate_unused_clauseの両方を指定することはできません。
- PCTVERSIONパラメータとRETENTIONパラメータの両方を指定することはできません。

- shrink_clauseは、SecureFiles LOBに対して指定できません。

関連項目:

LOBパラメータの設定の詳細は、「CREATE TABLE」の「LOB_storage_clause」および「LOB列：例」を参照してください。

alter_varray_col_properties

alter_varray_col_properties句を使用すると、VARRAYが格納されている既存のLOBの記憶特性を変更できます。

VARRAY列のプロパティの変更の制限事項

LOB_parametersのTABLESPACE句をこの句の一部として指定することはできません。VARRAYに対するLOB表領域のデフォルトは、表を含む表領域になります。

REKEY encryption_spec

REKEY句を指定すると、データベースの新しい暗号化キーが生成されます。表内の暗号化されている列はすべて、新しいキーを使用して再度暗号化され、このときに、encryption_specのUSING句が指定されている場合は新しい暗号化アルゴリズムが使用されます。この句は、このALTER TABLE文の中の他の句と組み合せることはできません。

関連項目:

列の透過的な暗号化の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

constraint_clauses

constraint_clausesを使用すると、表外宣言を使用して新しい制約を追加、既存の制約の状態を変更、および制約を削除できます。表外制約およびconstraint_stateのすべてのキーワードとパラメータの詳細は、「constraint」を参照してください。

制約の追加

ADD句を使用すると、表外制約または表外REF制約を表に追加できます。

制約の追加の制限事項

制約の追加には、次の制限事項があります。

- 重複表には制約を追加できません。
- シャード表には外部キー制約を追加できません。

関連項目:

「CHECK制約の無効化：例」、「オブジェクト識別子の指定：例」および「REF列：例」を参照してください。

制約の変更

MODIFY CONSTRAINT句を使用すると、既存の制約の状態を変更できます。

CASCADEキーワードは、外部キー制約が定義されている一意制約または主キー制約を使用禁止にする場合にのみ有効で

す。この場合、CASCADEを指定することによって、一意制約または主キー制約とその依存するすべての外部キー制約を使用禁止にする必要があります。

制約の変更の制限事項

制約の変更には、次の制限事項があります。

- NOT DEFERRABLE制約は、INITIALLY DEFERREDには変更できません。
- 索引構成表にこの句を指定した場合、同じ文では他の句を指定できません。
- 参照パーティション表の外部キー列のNOT NULL制約は変更できません。また、参照パーティション表の参照制約のパーティション化状態は変更できません。
- 重複表の制約を変更することはできません。

関連項目:

制約状態の変更: 例

制約名の変更

RENAME CONSTRAINT句を使用すると、tableの既存の制約名を変更できます。新しい制約名は、同一スキーマ内にあるオブジェクトの既存の制約と同じ名前にはできません。制約に依存するすべてのオブジェクトは、引き続き有効です。

関連項目:

「制約名の変更: 例」

drop_constraint_clause

drop_constraint_clauseを使用すると、データベースの整合性制約を削除できます。制約の適用を中止し、データ・ディクショナリから制約が削除されます。各drop_constraint_clauseには、制約を1つのみ指定できますが、1つの文の中では、複数のdrop_constraint_clauseを指定できます。

PRIMARY KEY

PRIMARY KEYを指定すると、tableの主キー制約を削除できます。

UNIQUE

UNIQUEを指定すると、指定した列の一意制約を削除できます。

ビットマップ結合索引が定義されている列から主キー制約または一意制約を削除すると、索引は無効になります。ビットマップ結合索引の詳細は、「CREATE INDEX」を参照してください。

CONSTRAINT

CONSTRAINT constraint_nameを指定すると、主キー制約または一意制約以外の整合性制約を削除できます。

CASCADE

CASCADEを指定すると、削除する整合性制約に依存するその他の整合性制約もすべて削除できます。

KEEP INDEX | DROP INDEX

KEEP INDEXまたはDROP INDEXを指定すると、PRIMARY KEYまたはUNIQUE制約の適用に使用する索引を残すか削除

するかを指定できます。

ONLINE

ONLINEを指定すると、制約を削除して表に対するDML操作が許可されるようになります。

制約の削除の制限事項

制約の削除には、次の制限事項があります。

- 参照整合性制約の一部の主キー制約または一意キー制約は、外部キーを削除しないと削除できません。参照されたキーと外部キーをともに削除する場合は、CASCADE句を使用してください。CASCADEを省略すると、外部キーによって参照される主キー制約および一意制約は削除されません。
- 主キーをオブジェクト識別子(OID)として使用している表では、主キー制約は(CASCADE句を使用しても)削除できません。
- REF列の参照整合性制約を削除した場合、REF列の有効範囲には参照先の表が含まれたままになります。
- REF列の有効範囲は削除できません。
- 参照パーティション表の外部キー列のNOT NULL制約は削除できません。また、参照パーティション表のパーティション化参照制約は削除できません。
- 列に対するNOT NULL制約は、その列のデフォルトの列値がON NULL句を使用して定義されていると削除できません。
- ONLINE句は、DEFERRABLE制約を削除しているときには指定できません。

関連項目:

「制約の削除: 例」

alter_external_table

alter_external_table句を使用すると、外部表の特性を変更できます。この句は、外部データには影響しません。parallel_clause、enable_disable_clause、external_table_data_propsおよびREJECT LIMIT句の構文およびセマンティクスは、CREATE TABLEの場合と同じです。「CREATE TABLE」の「external_table_clause」 を参照してください。

PROJECT COLUMN句

この句を使用すると、後続の問合せで、アクセス・ドライバが外部表に含まれる行を検証する方法を指定できます。デフォルトはPROJECT COLUMN ALLで、この場合、アクセス・ドライバは、選択されている列にかかわらず、すべての列の値を処理し、列のエントリが有効な行のみを検証します。いずれかの列の値でデータ型変換エラーなどのエラーが発生すると、その列がSELECT構文のリスト内で参照されていない場合でも、行が拒否されます。PROJECT COLUMN REFERENCEDを指定すると、アクセス・ドライバはSELECT構文のリスト内の列のみを処理します。

ALL設定の場合、一貫した結果セットが保証されます。REFERENCED設定の場合、後続の問合せで参照される列の数に応じて、戻される行の数が異なる可能性がありますが、ALL設定よりも高速です。後続の問合せで外部表のすべての列が選択されている場合、2つの設定の動作は同じになります。

外部表の変更の制限事項

外部表の変更には、次の制限事項があります。

- この句以外の句を使用して外部表を変更できません。
- LONG、VARRAYまたはオブジェクト型の列は外部表に追加できません。また、外部表の列のデータ型を、これらの型に変更できません。
- 外部表の記憶域パラメータは変更できません。

alter_table_partitioning

ここで説明する句は、パーティション表にのみ適用されます。1つのALTER TABLE文の中では、パーティション操作を他のパーティション操作または実表での操作と組み合せて使用することはできません。

表のパーティション化の変更のノート

表のパーティション化の変更時には、次のノートが適用されます。

- 1つ以上のマテリアライズド・ビューのマスター表で、パーティションを削除、交換、切捨て、移動、変更または分割した場合、その表に関する大量の既存ロード情報が削除されます。したがって、前述の操作のいずれかを行う前に、必ず依存するマテリアライズド・ビューをすべてリフレッシュしてください。
- tableにビットマップ結合索引が定義されている場合、tableのパーティションの変更操作を実行すると、索引にUNUSABLEのマークが付けられます。
- alter_table_partitioningの句のうち、参照パーティション表に対して指定できるのはmodify_table_default_attrs、move_table_[sub]partition、truncate_partition_subpartおよびexchange_partition_subpartのみです。これらの操作が参照パーティション表の子表に対してカスケードすることはありません。これ以外のパーティション・メンテナンス操作は、参照パーティション表に対しては有効ではありませんが、参照パーティション表の親表に対して指定することは可能であり、その操作は子である参照パーティション表にカスケードします。
- パーティションまたはサブパーティションを追加する際、表ごとに指定できるパーティションとサブパーティションの合計は1024K-1です。
- 表のパーティションまたはサブパーティションを追加するときにパーティション名を省略すると、「一般的なパーティション化のノート」で説明されているルールに従い、データベースによって名前が生成されます。
- 表のパーティションまたはサブパーティションに対して、移動、追加(ハッシュのみ)、結合、削除、分割、マージ、名前の変更、または切捨てを行った場合、その表を参照するプロシージャ、ファンクション、パッケージ、パッケージ本体、ビュー、型本体およびトリガーは、引き続き有効です。他の依存オブジェクトはすべて無効になります。
- LOB列を使用した表に新規セグメントを作成するパーティションのメンテナンス操作の遅延セグメント作成は、サポートされていません。セグメントは含まれる(サブ)パーティションに常に作成されます。
- シャード表では、表のパーティションおよびサブパーティションの変更に指定できる句は、UNUSABLE LOCAL INDEXESおよびREBUILD UNUSABLE LOCAL INDEXESのみです。システム・シャード表の個々のパーティションおよびサブパーティションに対して他の変更を実行することはできません。
- ユーザー定義のシャード表では、パーティションおよびサブパーティションに対する次の操作がサポートされています。
    - パーティションの追加、サブパーティションの追加
    - パーティションの削除、サブパーティションの削除
    - パーティションの分割
    - リスト・パーティションで値を追加または削除するためのパーティションの変更

- シャード表でサポートされるパーティション・メンテナンス操作は、パーティションおよびサブパーティションの切捨てのみです。シャード表でその他のパーティション・メンテナンス操作を実行することはできません。

関連するCONTEXTドメイン索引が含まれる表におけるパーティション操作の詳細は、『Oracle Textリファレンス』を参照してください。

ブロック・サイズが異なる表領域のパーティション化されたデータベース・エンティティの記憶域には、制限事項があります。これらの制限事項については、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

modify_table_default_attrs

modify_table_default_attrs句を使用すると、tableの属性に対する新しいデフォルト値を指定できます。文に指定した属性のみが影響を受けます。その後に作成するパーティションおよびLOBパーティションは、パーティションまたはLOBパーティションの作成時に明示的に上書きしないかぎり、この値を継承します。既存のパーティションおよびLOBパーティションは、この句の影響を受けません。

文の中で指定した属性のみが影響を受けます。指定されたデフォルト値は、個々のパーティションまたはLOBパーティションのレベルで指定された属性で上書きされます。

- FOR partition_extended_nameは、コンポジット・パーティション表にのみ適用されます。この句は、partition_extended_nameで指定されたパーティションの属性に新しいデフォルト値を指定します。その後に作成するパーティションのサブパーティションおよびLOBパーティションは、サブパーティションまたはLOBパーティションの作成時に明示的に上書きしないかぎり、この値を継承します。既存のサブパーティションは、この句の影響を受けません。

  デフォルトのディレクトリを変更している場合は、DEFAULT DIRECTORY ディレクトリを使用してその場所を保存できます。

- PCTTHRESHOLD、prefix_compressionおよびalter_overflow_clauseは、パーティション化された索引構成表にのみ有効です。

- 表レベルで接頭辞圧縮がすでに指定されている場合にかぎり、prefix_compression句を指定できます。また、COMPRESSキーワードの後にint型を指定することはできません。接頭辞の長さは、表の作成時にのみ指定できます。

- 索引構成表の索引セグメントに対しては、segment_attributes句でPCTUSEDパラメータを指定できません。

- read_only_clauseを使用すると、表のデフォルトの読取り専用または読取り/書込みモードを変更できます。新しいデフォルト・モードは、それ以降に表に追加されたパーティションまたはサブパーティションに割り当てられます。ただし、この動作は、新しいパーティションまたはサブパーティションにモードを指定することで上書きされます。表のデフォルトの読取り専用または読取り/書込みモードを変更するときには、表の既存のパーティションおよびサブパーティションのモードを変更しないでください。この句のセマンティクスの詳細は、「CREATE TABLE」の「read_only_clause」を参照してください。

- indexing_clauseを使用すると、表に対するデフォルトの索引付けプロパティを変更できます。新しいデフォルトの索引付けプロパティは、それ以降に表に追加されたパーティションまたはサブパーティションに割り当てられます。ただし、この動作は、新しいパーティションまたはサブパーティションに索引付けプロパティを指定することで上書きされます。表のデフォルトの索引付けプロパティを変更しても、その表内に既存のパーティションまたはサブパーティションの索引付けプロパティが変更されることはありません。この句のセマンティクスの詳細は、CREATE TABLEの「indexing_clause」を参照してください。

alter_automatic_partitioning

この句を使用すると、自動リスト・パーティション表を次のように管理できます。

- SET PARTITIONING AUTOMATIC句を使用すると、通常のリスト・パーティション表を自動リスト・パーティション表に変換できます。
- SET PARTITIONING MANUAL句を使用すると、自動リスト・パーティション表を通常のリスト・パーティション表に変換できます。
- SET STORE IN句は、自動リスト・パーティション表に対してのみ指定できます。これを指定すると、それ以降に自動的に作成されたリスト・パーティションのデータをデータベースが格納する1つ以上の表領域を指定できます。この句は、以前にSET STORE IN句を発行することで表に対して設定された表領域を上書きします。

既存の表が自動リスト・パーティション表かどうかを判別するには、USER_、DBA_、ALL_PART_TABLESデータ・ディクショナリ・ビューのAUTOLIST列を問い合せます。

alter_automatic_partitioningの制限事項

DEFAULTパーティションを含む通常のリスト・パーティション表を自動リスト・パーティション表に変換することはできません。

関連項目:

自動リスト・パーティション表の詳細は、「CREATE TABLE」の「AUTOMATIC」句を参照してください。

alter_interval_partitioning

この句は次の場合に使用します。

- 既存のレンジ・パーティション表を時間隔パーティションに変換する場合。データベースにより、最後のレンジ・パーティションの最大値を超えるデータの必要に応じて、指定した数値範囲または日時間隔のパーティションが自動的に作成されます。その表に子の参照パーティション表が存在する場合、その子表は子の時間隔参照パーティション表に変換されます。
- 既存の時間隔パーティション表の時間隔を変更する場合。データベースは、先に既存の時間隔パーティションをレンジ・パーティションに変換し、定義されたレンジ・パーティションの上限の値を決定します。次に、データベースにより、上限を超えたデータの必要に応じて、指定した数値範囲または日時間隔のパーティションが自動的に作成されます。
- 既存の時間隔パーティション表の表領域の記憶域を変更する場合。その表に子の時間隔参照パーティション表が存在する場合、新しい表領域の記憶域は、専用の表レベルのデフォルト表領域を持たない子表に継承されます。
- 時間隔パーティション表をレンジ・パーティション表に戻す場合。SET INTERVAL ( )を使用して、時間隔パーティションを無効にします。データベースは、作成されるレンジ・パーティションの上限として、作成済の時間隔パーティションの上限を使用し、既存の時間隔パーティションをレンジ・パーティションに変換します。その表に子の時間隔参照パーティション表が存在する場合、その子表は通常の子の参照パーティション表に変換されます。

exprには、有効な数値または期間式を指定します。

関連項目:

時間隔パーティションの詳細は、「CREATE TABLE」の「INTERVAL句」および『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

set_subpartition_template

set_subpartition_template句は、個々の表パーティションのデフォルトのレンジ、リストまたはハッシュ・サブパーティショ

ン定義を作成するときや、既存の定義を置き換えるときに使用します。この句は、コンポジット・パーティション表に対してのみ有効です。既存のサブパーティション・テンプレートが置き換えられますが、テンプレートがまだ作成されていない場合は新規作成されます。既存のサブパーティションや、既存のローカルおよびグローバルの索引に影響が及ぶことはありません。ただし、以降のパーティション化操作(追加やマージなどの操作)には、新しいテンプレートが使用されます。

既存のサブパーティション・テンプレートを削除するには、ALTER TABLE table SET SUBPARTITION TEMPLATE ()を指定します。

set_subpartition_template句は、CREATE TABLEのsubpartition_template句と同じセマンティクスを持ちます。詳細は、CREATE TABLEのsubpartition_template句を参照してください。

modify_table_partition

modify_table_partition句を使用すると、レンジ・パーティション、ハッシュ・パーティション、リスト・パーティションまたはシステム・パーティションの実際の物理属性を変更できます。そのパーティションの1つ以上のLOB項目の記憶域属性を任意に変更できます。物理属性(制限事項については後述)、ロギングおよび記憶域パラメータに対して、新しい値を指定できます。

すべてのタイプのパーティションについて、パーティションを変更した結果、使用不可能になったローカル索引の処理方法も指定できます。「UNUSABLE LOCAL INDEXES句」を参照してください。

パーティション化された索引構成表の場合、パーティションの変更時にマッピング表も同時に更新できます。「alter_mapping_table_clauses」を参照してください。

read_only_clause

read_only_clauseを使用して、表パーティションを読取り専用または読取り/書込みモードにすることができます。この句のセマンティクスの詳細は、「CREATE TABLE」の「read_only_clause」を参照してください。

indexing_clause

indexing_clauseを使用すると、表パーティションの索引付けプロパティを変更できます。この索引付けプロパティにより、パーティションが表の部分索引に含まれるかどうかが決まります。indexing_clauseは、modify_range_partition句、modify_hash_partition句、およびmodify_list_partition句に指定できます。

INDEXING ONを指定すると、表パーティションの索引付けプロパティをONに変更できます。この操作は、表の全索引には作用しません。表の部分索引に対して、次のように作用します。

- ローカルの部分索引: 表パーティションが、索引に含まれます。それに対応する索引パーティションは、再構築されてからUSABLEのマークが付けられます。
- グローバルのパーティション索引: 表パーティションは、索引に含まれます。表パーティションの索引エントリは、定期の索引メンテナンスの一環として索引に追加されます。

INDEXING OFFを指定すると、表パーティションの索引付けプロパティをOFFに変更できます。この操作は、表の全索引には作用しません。表の部分索引に対して、次のように作用します。

- ローカルの部分索引: 表パーティションが、索引から除外されます。それに対応する索引パーティションは、再構築されてからUNUSABLEのマークが付けられます。
- グローバルの部分索引: 表パーティションが、索引から除外されます。表パーティションの索引エントリは、索引から削除されます。これは、メタデータ専用の操作であるため、索引エントリは索引内に物理的に格納され続けます。このような孤立した索引エントリは、ALTER INDEX文またはmodify_index_partition句で、COALESCE CLEANUPを指定すると削除できます。

オブジェクト型の列の制限事項

オブジェクト型を持つ表はパーティション化できません。パーティション状態へのALTER TABLE変更は、オブジェクト型の列がない非パーティション・ヒープ表でのみサポートされます。

indexing_clauseの制限事項

この句は、単純なパーティション表のパーティションに対してのみ指定できます。コンポジット・パーティション表の場合は、表サブパーティション・レベルで`indexing_clause`を指定できます。詳細は、「modify_table_subpartition」を参照してください。

表パーティションの変更のノート

レンジ、リストおよびハッシュ表パーティションの操作には、次のノートが適用されます。

- すべてのタイプの表パーティションについて、`partition_attributes`句で`shrink_clause`を使用すると、各パーティションのセグメントを縮小できます。この句の詳細は、「shrink_clause」を参照してください。
- システム・パーティションを変更する構文およびセマンティクスは、ハッシュ・パーティションを変更する場合と同じです。「modify_hash_partition」を参照してください。
- `table`がコンポジット・パーティション化されている場合:
  - `allocate_extent_clause`を指定すると、`partition`のそれぞれのサブパーティションにエクステントが割り当てられます。
  - `deallocate_unused_clause`を指定すると、`partition`のそれぞれのサブパーティションから未使用の記憶域の割当てが解除されます。
  - この句で変更された他の属性は、`partition`のサブパーティションでも変更され、既存の値は上書きされます。既存のサブパーティションの属性が変更されないようにするには、modify_table_default_attrsのFOR PARTITION句を使用します。
- パーティション化されているネストした表を含む表パーティションの`partition_attributes`を変更した場合、その変更内容は、変更した表パーティションに対応するネストした表パーティションには適用されません。ただし、ALTER TABLE文を使用すると、ネストした表パーティションの記憶表を直接変更できます。
- 特に指定がないかぎり、`partition_attributes`の残りの句の動作は、パーティション表の作成時と同じです。詳細は、「CREATE TABLE」の「table_partitioning_clauses」を参照してください。

関連項目:

表のパーティションの変更: 例

modify_range_partition

この句を使用すると、レンジ・パーティションの特性を変更できます。

add_range_subpartition

この句は、レンジ-レンジ・コンポジット・パーティションに対してのみ有効です。これにより、1つ以上のレンジ・サブパーティションを`partition`に追加できます。

Oracle Database 12cリリース2 (12.2)以降、この句を使用して、コンポジット・パーティション外部表にサブパーティションを追加できるようになりました。この場合、`range_subpartition_desc`句のオプションの`external_part_subpart_data_props`句を指定できます。この句のセマンティクスの詳細は、

「external_part_subpart_data_props」を参照してください。

レンジ・サブパーティションの追加の制限事項

tableが索引構成表の場合は、一度に追加できるレンジ・サブパーティションは1つのみになります。

add_hash_subpartition

この句は、レンジ-ハッシュ・コンポジット・パーティションに対してのみ有効です。add_hash_subpartition句を使用すると、ハッシュ・サブパーティションをpartitionに追加できます。Oracle Databaseは、ハッシュ・ファンクションによってpartitionの他のサブパーティションから再ハッシュされた行を、新しいサブパーティションに移入します。ロード・バランシングを最適化する場合、サブパーティションの合計数は2の累乗にする必要があります。

partitioning_storage_clauseでサブパーティションに対して指定できる句は、TABLESPACE句のみです。TABLESPACEを指定しなかった場合、新しいサブパーティションはpartitionのデフォルトの表領域に格納されます。

選択したパーティションに対応するローカル索引パーティションが追加されます。

追加したパーティションに対応するローカル索引パーティションにUNUSABLEのマークが付けられます。ヒープ構成表のすべての索引が無効になります。update_index_clausesを使用し、操作中にこれらの索引を更新できます。

add_list_subpartition

この句は、レンジ-リストおよびリスト-リスト・コンポジット・パーティションに対してのみ有効です。これにより、1つ以上のリスト・サブパーティションをpartitionに追加できます。ただし、DEFAULTのサブパーティションが作成されていない場合に限られます。

- この操作にはlist_values_clauseが必要です。また、list_values_clauseには、partitionのその他のサブパーティションには存在していない値を指定する必要があります。ただし、他のパーティションのサブパーティションで使用されている値は指定できます。
- partitioning_storage_clauseでサブパーティションに対して指定できる句は、TABLESPACE句および表の圧縮のみです。
- Oracle Database 12cリリース2 (12.2)以降、この句を使用して、コンポジット・パーティション外部表にサブパーティションを追加できるようになりました。この場合、list_subpartition_desc句のオプションのexternal_part_subpart_data_props句を指定できます。この句のセマンティクスの詳細は、「external_part_subpart_data_props」を参照してください。

追加されたサブパーティションごとに、Oracle Databaseは、表のすべてのローカル索引パーティションに、同じ値リストを持つサブパーティションも追加します。表の既存のローカル索引パーティションおよびグローバル索引パーティションは影響を受けません。

リスト・サブパーティションの追加の制限事項

リスト・サブパーティションの追加には、次の制限事項が適用されます。

- この句は、このパーティションにDEFAULTのサブパーティションが作成されている場合は指定できません。その場合、split_list_subpartition句を使用して、DEFAULTパーティションを分割する必要があります。
- tableが索引構成表の場合は、一度に追加できるリスト・サブパーティションは1つのみになります。

coalesce_table_subpartition

COALESCE SUBPARTITIONは、ハッシュ・サブパーティションにのみ適用されます。COALESCE SUBPARTITIONを使用すると、最後のハッシュ・サブパーティションが選択され、その内容が1つ以上の残りのサブパーティション(ハッシュ・ファンクションが決定)に分散された後、選択されたサブパーティションが削除されます。

- 選択したパーティションに対応するローカル索引パーティションが削除されます。
- 1つ以上の吸収パーティションに対応するローカル索引パーティションにUNUSABLEのマークが付けられます。ヒープ構成表のすべてのグローバル索引が無効になります。update_index_clausesを使用し、操作中にこれらの索引を更新できます。

modify_hash_partition

ハッシュ・パーティションに変更を加えるときは、partition_attributes句の中で指定できるのはallocate_extent_clauseおよびdeallocate_unused_clauseのみです。パーティションのそれ以外の属性はすべて、表レベルのデフォルトから継承されますが、TABLESPACEは作成時のままとなります。

modify_list_partition

リスト・パーティションを変更するときに使用できる句のセマンティクスは、レンジ・パーティションを変更するときと同じです。リスト・パーティションを変更する場合、次の句も使用できます。

ADD | DROP VALUES句

これらの句は、コンポジット・パーティションを変更する場合のみ有効です。これらの句によって、表のローカル索引とグローバル索引が影響を受けることはありません。

- ADD VALUES句を使用すると、partitionのpartition_key_valueリストが拡張され、追加した値が含まれます。追加するパーティションの値は、「CREATE TABLE」の句「list_partitions」に示すすべてのルールおよび制限事項に準拠する必要があります。
- DROP VALUES句を使用すると、1つ以上のpartition_key_valueが削除され、partitionのpartition_key_valueリストが縮小されます。この句を指定すると、Oracle Databaseはこの値の行が存在しないことを検証します。そのような行が存在する場合は、エラーが戻されます。

ノート:

表にローカル同一キー索引が定義されている場合、DEFAULT リスト・パーティションのある表では ADD VALUES 操作および DROP VALUES 操作が改善されます。

リスト値の追加および削除の制限事項

リスト値の追加および削除には、次の制限事項があります。

- デフォルトのリスト・パーティションに対して値を追加または値を削除することはできません。
- tableにDEFAULTパーティションが定義されている場合、デフォルト・パーティション以外に値を追加しようとすると、その値がDEFAULTパーティションに存在しているかどうかが確認されます。デフォルト・パーティションにその値が存在する場合、エラーが戻されます。

modify_table_subpartition

この句は、コンポジット・パーティション表にのみ適用されます。その副次句によって、個別のレンジ、リストまたはハッシュ・サブパーティションの特性を変更できます。

shrink_clauseを使用すると、サブパーティションの各セグメントを縮小できます。この句の詳細は、「shrink_clause」を参照してください。

また、パーティションを変更した結果、使用不可能になったローカル索引の処理方法も指定できます。「UNUSABLE LOCAL

INDEXES句」を参照してください。

read_only_clauseを使用して、表サブパーティションを読取り専用または読取り/書込みモードにすることができます。この句のセマンティクスの詳細は、「CREATE TABLE」の「read_only_clause」を参照してください。

indexing_clauseを使用すると、表サブパーティションの索引付けプロパティを変更できます。索引付けプロパティにより、表の部分索引にサブパーティションを含めるかどうかが決まります。表サブパーティションの索引付けプロパティの変更による索引サブパーティションへの影響は、表パーティションの索引付けプロパティの変更による索引パーティションへの影響と同じです。詳細は、modify_table_partitionの「indexing_clause」を参照してください。

ハッシュ・サブパーティションの変更の制限事項

subpartitionに指定できるmodify_LOB_parametersは、allocate_extent_clauseおよびdeallocate_unused_clauseのみです。

ADD | DROP VALUES句

これらの句は、リスト・サブパーティションを変更する場合のみ有効です。これらの句によって、表のローカル索引とグローバル索引が影響を受けることはありません。

- ADD VALUES句を使用すると、subpartitionのsubpartition_key_valueリストが拡張され、追加した値が含まれます。追加するパーティションの値は、「CREATE TABLE」の句「list_partitions」に示すすべてのルールおよび制限事項に準拠する必要があります。
- DROP VALUES句を使用すると、1つ以上のsubpartition_key_valueが削除され、subpartitionのsubpartition_key_valueリストが縮小されます。この句を指定すると、Oracle Databaseはこの値の行が存在しないことを検証します。そのような行が存在する場合は、エラーが戻されます。

また、パーティションを変更した結果、使用不可能になったローカル索引の処理方法も指定できます。「UNUSABLE LOCAL INDEXES句」を参照してください。

リスト・サブパーティションの変更の制限事項

subpartitionに指定できるmodify_LOB_parametersは、allocate_extent_clauseおよびdeallocate_unused_clauseのみです。

move_table_partition

move_table_partition句を使用すると、partitionを別のセグメントへ移動できます。パーティション・データの別の表領域への移動、断片化を削減するためのデータの再クラスタ化、および作成時の物理属性の変更ができます。

表にLOB列が含まれている場合、LOB_storage_clauseを使用して、このパーティションに関連付けられたLOBデータおよびLOB索引セグメントを移動できます。この場合、指定したLOBのみが影響を受けます。特定のLOB列にLOB_storage_clauseを指定しなかった場合、その列のLOBデータおよびLOB索引セグメントは移動されません。

ネストした表の列が表に含まれる場合、table_partition_descriptionのnested_table_col_properties句を使用すると、このパーティションに関連付けられているネストした表のセグメントを移動できます。この場合、指定しているネストした表のみが影響を受けます。特定のネストした表の列に対しtable_partition_descriptionのnested_table_col_properties句を指定しなかった場合、そのセグメントは移動されません。

指定したパーティションに対応するローカル索引パーティションが、Oracle Databaseによって移動されます。移動したパーティションが空でない場合、UNUSABLEのマークが付けられます。ヒープ構成表のグローバル索引が無効になります。update_index_clausesを使用し、操作中にこれらの索引を更新できます。

LOBデータ・セグメントを移動する場合、古いデータ・セグメントおよび対応する索引セグメントが削除され、新しい表領域を指

定しない場合でも、新しいセグメントが作成されます。

移動操作では、parallel_clause(指定されている場合)からパラレル属性が取得されます。parallel_clauseが指定されていない場合は、表のデフォルトのパラレル属性があれば、これが使用されます。いずれも指定されていない場合は、シリアルに移動が行われます。

MOVE PARTITIONでparallel_clauseを指定した場合、tableのデフォルトのパラレル属性は変更されません。

ノート:

索引構成表の場合、主キーのアドレスおよびその値を使用して、論理 ROWID が構成されます。論理 ROWID は、表の 2 次索引に格納されます。索引構成表のパーティションを移動した場合、ROWID のアドレス部分が変更され、パフォーマンスの障害になる場合があります。最適なパフォーマンスを維持するには、移動したパーティションの 2 次索引を再構築し、ROWID を更新してください。

関連項目:

表パーティションの移動: 例

MAPPING TABLE

MAPPING TABLE句は、マッピング表が定義されている索引構成表のみに有効です。マッピング表は、索引構成表のパーティションとともに移動されます。マッピング表のパーティションは、移動した索引構成表のパーティションの物理属性を継承します。これは、マッピング表のパーティションの属性を変更する唯一の方法です。この句を省略した場合、マッピング表のパーティションでは、元の属性が保持されます。

対応するすべてのビットマップ索引パーティションにはUNUSABLEのマークが付けられます。

この句の詳細は、「CREATE TABLE」の「mapping_table_clauses」を参照してください。

ONLINE

ONLINEを指定すると、表パーティションの移動中に、表パーティションに対するDML操作が許可されるようになります。

ONLINE句の制限事項

表パーティションの移動時には、ONLINE句に次の制限事項が適用されます。

- ONLINE句は、SYSが所有する表に対しては指定できません。
- ONLINE句は、索引構成表には指定できません。
- ONLINE句は、オブジェクト・タイプを格納するヒープ構成表や、ビットマップ結合索引またはドメイン索引が定義されているヒープ構成表には指定できません。
- パラレルDMLおよびダイレクト・パスINSERT操作は、表に対する排他的ロックを必要とします。したがって、これらの操作は、競合するロックにより、実行中のオンライン・パーティションMOVEと同時にはサポートされません。

表パーティションの移動の制限事項

表パーティションの移動には、次の制限事項があります。

- partitionがハッシュ・パーティションである場合、この句にTABLESPACEの属性以外は指定できません。

- この句は、サブパーティションを含むパーティションに対して指定できません。ただし、move_table_subpartition句を使用してサブパーティションを移動できます。

move_table_subpartition

move_table_subpartition句を使用して、subpartition_extended_nameで識別されるサブパーティションを別のセグメントに移動します。TABLESPACEを指定しない場合、サブパーティションは同じ表領域に残ります。

サブパーティションが空でない場合、移動したサブパーティションに対応するすべてのローカル索引サブパーティションにUNUSABLEのマークが付けられます。update_index_clausesを使用し、操作中にヒープ構成表のすべての索引を更新できます。

表にLOB列が含まれている場合、LOB_storage_clauseを使用して、このサブパーティションに関連付けられたLOBデータおよびLOB索引セグメントを移動できます。この場合、指定したLOBのみが影響を受けます。特定のLOB列にLOB_storage_clauseを指定しなかった場合、その列のLOBデータおよびLOB索引セグメントは移動されません。

LOBデータ・セグメントを移動する場合、古いデータ・セグメントおよび対応する索引セグメントが削除され、新しい表領域を指定しない場合でも、新しいセグメントが作成されます。

ONLINE

ONLINEを指定すると、表サブパーティションの移動中に、表サブパーティションに対するDML操作が許可されるようになります。

ONLINE句の制限事項

表サブパーティションを移動するためのONLINE句には、表パーティションを移動するためのONLINE句と同じ制限事項があります。「ONLINE句の制限事項」を参照してください。

表サブパーティションの移動の制限事項

指定できるpartitioning_storage_clauseの句は、TABLESPACE句およびtable_compressionのみです。

add_external_partition_attrs

パーティション表に外部パラメータを追加するには、この句を使用します。

add_table_partition

add_table_partition句を使用すると、1つ以上のレンジ・パーティション、リスト・パーティション、またはシステム・パーティションをtableに追加できます。または、1つのハッシュ・パーティションをtableに追加できます。

追加されたパーティションごとに、Oracle Databaseは、tableに定義されているローカル索引に、実表のパーティションと同じ名前で新しいパーティションを追加します。索引に同じ名前のパーティションがすでに存在する場合、SYS_Pnという形式でパーティションの名前が生成されます。

tableが索引構成されている場合、追加されたパーティションごとに、Oracle Databaseは、表に定義されたすべてのマッピング表およびオーバーフロー領域にパーティションを追加します。

tableが参照パーティション表の親表の場合は、dependent_tables_clauseを使用して、この文に指定するパーティション・メンテナンス操作を参照パーティション表のすべての子表に伝播できます。

tableのデフォルトの索引付けプロパティは、新しい表パーティションに継承されます。これは、table_partition_description句にindexing_clauseを使用して、リスト・パーティション、レンジ・パーティション、またはシステム・パーティションの索引付けプロパティを設定するか、add_hash_partition_clauseにindexing_clauseを使用して、ハッシュ・パーティションの索引付けプロパティを設定することで上書きできます。

コンポジット・パーティション表に追加されたパーティションごとに、Oracle Databaseは、同じサブパーティション記述を持つ新しい索引パーティションを、tableに定義されたすべてのローカル索引に追加します。tableのグローバル索引が影響を受けることはありません。新しい表パーティションに索引付けプロパティを指定すると、新しいサブパーティションは、そのパーティションの索引付けプロパティを継承します。それ以外の場合、新しいサブパーティションは、表のデフォルトの索引付けプロパティを継承します。これは、range_subpartition_desc句、individual_hash_subparts句、およびlist_subpartition_desc句でindexing_clauseを使用して、サブパーティションの索引付けプロパティを設定することで上書きできます。

BEFORE句

オプションのBEFORE句は、システム・パーティションをtableに追加するときにのみ指定できます。この句を使用すると、新しいパーティションを追加する場所を、既存のパーティションとの関係で指定できます。システム・パーティションは分割できません。そのため、この句は、既存の1つのパーティションの内容を複数の新しいパーティションに分割する場合に有効です。この句を省略した場合、新しいパーティションは既存のパーティションの後に追加されます。

表パーティションの追加の制限事項

tableが索引構成表の場合や、ローカル・ドメイン索引がtableに定義されている場合は、一度に追加できるパーティションは1つのみになります。

関連項目:

「LOBおよびネストした表の記憶域を持つ表パーティションの追加: 例」および「表への複数のパーティションの追加: 例」を参照してください。

add_range_partition_clause

add_range_partition_clauseを使用すると、新しいレンジ・パーティションをレンジ・パーティション表またはコンポジット・レンジ・パーティション表の一番上(最後の既存のパーティションの後)に追加できます。

ドメイン索引がtableで定義されている場合、IN_PROGRESSまたはFAILEDのマークが付いていると無効になります。

レンジ・パーティションの追加の制限事項

レンジ・パーティションの追加には、次の制限事項があります。

- 既存の上位パーティションにある各パーティション化キーのパーティションの上限がMAXVALUEの場合、表にパーティションを追加できません。そのかわり、split_table_partition句を使用して、表の始めまたは中間にパーティションを追加します。
- prefix_compressionおよびOVERFLOW句は、パーティション化された索引構成表に対してのみ有効です。表レベルで接頭辞圧縮が使用可能な場合にかぎり、prefix_compressionを指定できます。パーティション表にすでにオーバーフロー・セグメントが存在する場合にかぎり、OVERFLOWを指定できます。
- PCTUSEDパラメータは、索引構成表の索引セグメントに対して指定できません。

range_values_clause

新しいパーティションの上限を指定します。value_listは、パーティション・キー列に対応するリテラル値を順序どおりにカンマで区切ったリストです。value_listの値は、表内にある既存の最上位パーティションのパーティション境界より大きくする必要があります。

table_partition_description

この句を使用して、新しいパーティションに作成時の物理属性を指定できます。表にLOB列が含まれている場合、1つ以上のLOB項目にパーティション・レベルの属性を指定することもできます。

external_part_subpart_data_props

Oracle Database 12cリリース2 (12.2)以降、パーティション外部表およびコンポジット・パーティション外部表がサポートされるようになりました。これらの表にパーティションを追加する場合、オプションでこの句を使用して、パーティションにDEFAULT DIRECTORYおよびLOCATIONを指定できます。これらの句のセマンティクスの詳細は、「CREATE TABLE」の「DEFAULT DIRECTORY」および「LOCATION」を参照してください。

サブパーティションの説明

これらの句は、コンポジット・パーティション表に対してのみ有効です。新しいパーティションのサブパーティションを指定する場合は、range_subpartition_desc、list_subpartition_desc、individual_hash_subpartsまたはhash_subparts_by_quantity句を適切に使用します。この句によって、表レベルでsubpartition_templateに定義された任意のサブパーティションの設定は上書きされます。

add_hash_partition_clause

add_hash_partition_clauseを使用すると、ハッシュ・パーティション表の一番上に新しいハッシュ・パーティションを追加できます。Oracle Databaseは、ハッシュ・ファンクションによってtableの他のパーティションから再ハッシュされた行を、新しいパーティションに移入します。ロード・バランシングを最適化する場合、パーティションの合計数は2の累乗にする必要があります。

パーティションの名前を指定でき、パーティションが格納される表領域を指定することもできます。名前を指定しない場合は、SYS_Pnという形式のパーティション名が割り当てられます。TABLESPACEを指定しない場合は、新しいパーティションは表のデフォルトの表領域に格納されます。それ以外の属性は常に、表レベルのデフォルトから継承されます。

この操作によってパーティション間でデータが再ハッシュされると、対応するすべてのローカル索引パーティションにUNUSABLEのマークが付けられます。update_index_clausesを使用し、操作中にヒープ構成表のすべての索引を更新できます。

parallel_clauseを使用すると、新しいパーティションの作成をパラレル化するかどうかを指定できます。

read_only_clauseを使用して、表パーティションを読取り専用または読取り/書込みモードにすることができます。この句のセマンティクスの詳細は、「CREATE TABLE」の「read_only_clause」を参照してください。

indexing_clauseを使用すると、パーティションに索引付けプロパティを指定できます。この句を省略すると、そのパーティションはtableのデフォルトの索引付けプロパティを継承します。

関連項目:

ハッシュ・パーティション化の詳細は、「CREATE TABLE」および『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

add_list_partition_clause

add_list_partition_clauseを使用すると、パーティションの新しい一連の値を使用して、tableに新しいパーティションを追加できます。新しいパーティションに作成時の物理属性を指定できます。表にLOB列が含まれている場合、1つ以上のLOB項目にパーティション・レベルの属性を指定することもできます。

リスト・パーティションの追加の制限事項

表にDEFAULTパーティションをすでに定義している場合は、リスト・パーティションを追加できません。その場合、split_table_partition句を使用して、DEFAULTパーティションを分割する必要があります。

関連項目:

- リスト・パーティションの詳細および制限事項については、「CREATE TABLE」の「list_partitions」を参照してください。
- デフォルト・リスト・パーティションの使用: 例

add_system_partition_clause

この句を使用すると、パーティションをシステム・パーティション表に追加できます。表に定義されているすべてのローカル索引に、対応する索引パーティションが追加されます。

table_partition_descriptionによって、新しいパーティションのパーティション・レベルの属性を指定できます。指定しない属性の値は、表レベルの値から継承されます。

システム・パーティションの追加の制限事項

システム・パーティションの追加時には、OVERFLOW句を指定できません。

関連項目:

システム・パーティションの詳細は、「CREATE TABLE」の句「system_partitioning」を参照してください。

coalesce_table_partition

COALESCEは、ハッシュ・パーティションにのみ適用されます。coalesce_table_partition句を使用すると、最後のハッシュ・パーティションが選択され、その内容がハッシュ・ファンクションによって決定される1つ以上の残りのパーティションに分散された後、選択されたパーティションが削除されます。

選択したパーティションに対応するローカル索引パーティションが削除されます。1つ以上の吸収パーティションに対応するローカル索引パーティションにUNUSABLEのマークが付けられます。ヒープ構成表のすべての索引が無効になります。update_index_clausesを使用すると、操作中にすべての索引を更新できます。

表パーティションの結合の制限事項

update_all_indexes_clauseを使用してグローバル索引を更新する場合、副次句ではなくUPDATE INDEXESキーワードのみを指定できます。

drop_external_partition_attrs

パーティション表で外部パラメータを削除するには、この句を使用します。

drop_table_partition

drop_table_partition句は、パーティション表から、パーティションと、そのパーティションに含まれるデータを削除します。データを表に残したままパーティションを削除する場合は、そのパーティションを隣接するパーティションにマージする必要があります。

Oracle Database 12cリリース2 (12.2)以降、この句を使用して、パーティション表またはコンポジット・パーティション外部表からパーティションを削除できるようになりました。

関連項目:

merge_table_partitions

partition_extended_names句を使用して、1つ以上のパーティションを削除するように指定します。複数のパーティションを指定する場合は、構文図の上位ブランチに示すようにすべてのパーティションを名前で指定するか、構文図の下位ブランチに示すようにFOR句を使用してすべてのパーティションを指定する必要があります。両方のタイプの構文を1回の削除操作で使用することはできません。

- tableにLOB列が存在する場合は、削除される表パーティションに対応するLOBデータ、LOB索引パーティション、およびサブパーティションも削除されます。
- tableにパーティション化されているネストした表の列が含まれる場合は、削除される表パーティションに対応するネストした表パーティションも削除されます。
- tableが索引構成されており、定義されたマッピング表を持つ場合、対応するマッピング表のパーティションも同様に削除されます。
- ローカル索引パーティションおよび削除されるパーティションに対応するサブパーティションは、UNUSABLEのマークが付いている場合でも削除されます。

update_index_clausesを使用し、操作中にtableの索引を更新できます。グローバル索引の更新は、メタデータ専用のため、削除操作で削除されるレコードの索引エントリは索引に物理的に格納されたままになります。このような孤立した索引エントリは、ALTER INDEX文またはmodify_index_partition句で、COALESCE CLEANUPを指定すると削除できます。

update_index_clausesとともにparallel_clauseを指定すると、削除操作ではなく、索引の更新がパラレル化されます。

レンジ・パーティションを削除し、その後、削除したパーティションに属していた行を挿入した場合、1つ上位のパーティションに行が格納されます。ただし、そのパーティションが最上位のパーティションである場合、削除したパーティションが表していた値の範囲が表に対して無効になるため、挿入は失敗します。

表パーティションの削除の制限事項

表パーティションの削除には、次の制限事項があります。

- ハッシュ・パーティション表のパーティションは削除できません。かわりに、coalesce_table_partition句を使用してください。
- 表のすべてのパーティションを削除できません。かわりに表を削除します。
- update_all_indexes_clauseを使用してグローバル索引を更新する場合、副次句ではなくUPDATE INDEXESキーワードのみを指定できます。
- tableが索引構成表の場合や、ローカル・ドメイン索引がtableに定義されている場合は、一度に削除できるパーティションは1つのみになります。
- 重複表のパーティションは削除できません。

  パーティションを削除しても、ごみ箱の設定にかかわらずそのパーティションはOracle Databaseのごみ箱に置かれません。削除されたパーティションはただちにシステムから削除されます。

関連項目:

表パーティションの削除: 例

drop_table_subpartition

この句を使用すると、レンジ、リスト、またはハッシュ・コンポジット・パーティション表からレンジまたはリスト・サブパーティションを削除できます。削除対象のサブパーティション内の行はすべて削除されます。

Oracle Database 12cリリース2 (12.2)以降、この句を使用して、コンポジット・パーティション外部表からサブパーティションを削除できるようになりました。

subpartition_extended_names句を使用して、1つ以上のサブパーティションを削除するように指定します。複数のサブパーティションを指定する場合は、構文図の上位ブランチに示すようにすべてのサブパーティションを名前で指定するか、構文図の下位ブランチに示すようにFOR句を使用してすべてのサブパーティションを指定する必要があります。両方のタイプの構文を1回の削除操作で使用することはできません。

ローカル索引の対応するサブパーティションは、自動的に削除されます。その他の索引サブパーティションには影響がありません。update_global_index_clauseまたはupdate_all_indexes_clauseを指定しないかぎり、グローバル索引にはUNUSABLEのマークが付けられます。グローバル索引の更新は、メタデータ専用のため、削除操作で削除されるレコードの索引エントリは索引に物理的に格納されたままになります。このような孤立した索引エントリは、ALTER INDEX文またはmodify_index_partition句で、COALESCE CLEANUPを指定すると削除できます。

表サブパーティションの削除の制限事項

表サブパーティションの削除には、次の制限事項があります。

- ハッシュ・サブパーティションは削除できません。かわりに、MODIFY PARTITION ... COALESCE SUBPARTITION構文を使用してください。
- パーティションのすべてのサブパーティションを削除できません。かわりに、drop_table_partition句を使用してください。
- グローバル索引を更新する場合、update_all_indexes_clauseのオプションの副次句を指定することはできません。
- tableが索引構成表の場合は、一度に削除できるサブパーティションは1つのみになります。
- 複数のサブパーティションを削除する場合、すべてのサブパーティションは同じパーティションにある必要があります。
- 重複表のサブパーティションは削除できません。

rename_partition_subpart

rename_partition_subpart句を使用すると、表パーティションまたは表サブパーティションの名前をnew_nameに変更できます。パーティションおよびサブパーティションのどちらの場合も、new_nameは同じ表に存在するすべてのパーティションおよびサブパーティションと異なる値である必要があります。

tableが索引構成されている場合、対応する主キー索引パーティションに、既存のオーバーフロー・パーティションおよびマッピング表パーティションと同じ名前が割り当てられます。

Oracle Database 12cリリース2 (12.2)以降、この句を使用して、パーティションまたはコンポジット・パーティション外部表のパーティションまたはサブパーティションの名前を変更できるようになりました。

関連項目:

表のパーティション名の変更: 例

truncate_partition_subpart

TRUNCATE partition_extended_namesを指定すると、partition_extended_namesで指定したパーティションからすべての行を削除することができ、表がコンポジット・パーティション化されている場合は、それらのパーティションのサブパーティションからすべての行が削除されます。個別のサブパーティションからすべての行を削除するには、TRUNCATE subpartition_extended_namesを指定します。tableが索引構成されている場合、対応するすべてのマッピング表のパーティションおよびオーバーフロー領域のパーティションが切り捨てられます。

複数のパーティションを指定する場合は、partition_extended_names構文図の上位ブランチに示すようにすべてのパーティションを名前で指定するか、構文図の下位ブランチに示すようにFOR句を使用してすべてのパーティションを指定する必要があります。両方のタイプの構文を1回の切捨て操作で使用することはできません。subpartition_extended_names句で複数のサブパーティションを指定する場合も同じルールが適用されます。

指定された各パーティションまたはサブパーティションについて、次の事項が適用されます。

- 切り捨てるパーティションまたはサブパーティションにデータが含まれている場合は、まず、その表の参照整合性制約を使用禁止にする必要があります。また、別の方法として、行を削除してからパーティションを切り捨てる方法もあります。
- tableにLOB列が存在する場合、このパーティションのLOBデータおよびLOB索引セグメントも切り捨てられます。tableがコンポジット・パーティション化されている場合、このパーティションのサブパーティションのLOBデータおよびLOB索引セグメントは切り捨てられます。
- tableにパーティション化されているネストした表が含まれる場合、親パーティションに対応するネストした表パーティションが空でないかぎり、親パーティションを切り捨てることはできません。
- tableでドメイン索引が定義されている場合、索引にIN_PROGRESSまたはFAILEDのマークが付いていると無効になります。また、切り捨てられる表パーティションに対応する索引パーティションにIN_PROGRESSのマークが付いていると無効になります。

切り捨てられるそれぞれのパーティションまたはサブパーティションでは、対応するローカル索引パーティションおよびサブパーティションも切り捨てられます。これらの索引パーティションまたはサブパーティションにUNUSABLEのマークが付いている場合、これらは切り捨てられ、UNUSABLEのマークはVALIDにリセットされます。

update_index_clausesを使用し、操作中にtableの索引を更新できます。グローバル索引の更新は、メタデータ専用のため、切捨て操作で削除されるレコードの索引エントリは索引に物理的に格納されたままになります。このような孤立した索引エントリは、ALTER INDEX文またはmodify_index_partition句で、COALESCE CLEANUPを指定すると削除できます。

update_index_clausesとともにparallel_clauseを指定すると、切捨て操作ではなく、索引の更新がパラレル化されます。

DROP STORAGE

DROP STORAGEを指定すると、MINEXTENTSパラメータで割り当てられた領域を除き、削除された行からすべての領域の割当てを解除できます。この領域は、後で表領域内の他のオブジェクトで使用できます。

DROP ALL STORAGE

DROP ALL STORAGEを指定すると、MINEXTENTSパラメータで割り当てられた領域も含め、削除された行からすべての領域の割当てを解除できます。パーティションまたはサブパーティションのすべてのセグメント、およびそれらの依存オブジェクトのすべてのセグメントも、割当てが解除されます。

DROP ALL STORAGEの制限事項

この句には、「遅延セグメント作成の制限事項」に示されているものと同じ制限事項があります。

REUSE STORAGE

REUSE STORAGEを指定すると、削除した行が占有していた領域をパーティションまたはサブパーティションに割り当てることができます。この領域は、そのパーティションまたはサブパーティションに対する、それ以降の挿入および更新のためにのみ使用できます。

CASCADE

CASCADEを指定すると、tableに含まれるすべての子の参照パーティション表で、対応するパーティションまたはサブパーティションを切捨てできます。

表のパーティションおよびサブパーティションの切捨ての制限事項

表のパーティションおよびサブパーティションの切捨てには、次の制限事項があります。

- update_all_indexes_clauseを使用してグローバル索引を更新する場合、副次句ではなくUPDATE INDEXESキーワードのみを指定できます。
- tableが索引構成表の場合や、ローカル・ドメイン索引がtableに定義されている場合は、一度に切捨てできる表パーティションまたは表サブパーティションは1つのみになります。
- 重複表のパーティションまたはサブパーティションを切り捨てることはできません。

関連項目:

表パーティションの切捨て: 例

split_table_partition

split_table_partition句を使用すると、partition_extended_nameによって指定されるパーティションから新しいセグメント、物理属性および初期エクステントをそれぞれ含む、複数の新しいパーティションが作成されます。現行パーティションに対応付けられたセグメントは、廃棄されます。

新しいパーティションは、指定されていないすべての物理属性を現行パーティションから継承します。

ノート:

一定の条件が満たされると、SPLIT PARTITION および SPLIT SUBPARTITION 操作を最適化して処理速度を上げることができます。これらの操作の最適化の詳細は、『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。

- DEFAULTのリスト・パーティションを分割すると、最終的な結果のパーティションは、DEFAULTの値を保持するようになります。それ以外の結果のパーティションは、指定した分割値を保持するようになります。
- tableが索引構成表である場合は、対応するマッピング表のパーティションが分割されて、親の索引構成表のパーティションと同じ表領域に配置されます。対応するオーバーフロー領域も分割されるので、OVERFLOW句を使用して新しいオーバーフロー領域のセグメント属性を指定できます。
- tableにLOB列がある場合、LOB_storage_clauseを使用して、分割の結果生成されたLOBデータ・セグメントに対して個々のLOB記憶域属性を指定できます。現行パーティションのLOBデータおよびLOB索引セグメントが削除された後、新しい表領域を指定しなくても、各パーティションの各LOB列に新しいセグメントが作成されます。

- tableにネストした表の列が含まれる場合、split_nested_table_part句を使用して、分割の結果生成されたネストした表セグメントに対して、記憶表名およびセグメント属性を指定できます。現行パーティションのネストした表セグメントが削除された後、各パーティションのネストした表の列ごとに新しいセグメントが作成されます。この句により、親表には、複数レベルのネストした表の列のみでなく、複数のネストした表の列を含めることが可能になります。

対応するローカル索引パーティションにUNUSABLEのマークが付いている場合でも、それらは分割されます。UNUSABLEのマークが付けられ、ユーザーは、分割パーティションに対応するローカル索引パーティションを再構築する必要があります。新しい索引パーティションの属性は、分割されたパーティションから継承されます。新しい索引パーティションは、分割された索引パーティションのデフォルト表領域に格納されます。索引パーティションにデフォルト表領域が定義されていない場合、基礎となる新しい表のパーティションの表領域が使用されます。

AT句

AT句は、レンジ・パーティションにのみ適用されます。この句を使用すると、1つのレンジ・パーティションを2つのレンジ・パーティションに分割できます。新しい2つのパーティションのうちの最初の方に、新しい上限(境界を含まない)を指定します。値リストは、現行パーティションの元のパーティション境界より小さく、その次に小さいパーティション(そのようなパーティションがある場合)のパーティション境界より大きい値にする必要があります。

VALUES句

VALUES句は、リスト・パーティションにのみ適用されます。この句を使用すると、1つのリスト・パーティションを2つのリスト・パーティションに分割できます。1つのキー列で表がパーティション化されている場合は、list_values構文の上位ブランチを使用して、その列の値リストを指定します。NULLを指定できるのは、表内の別のパーティションに対してまだNULLが指定されていない場合です。複数のキー列で表がパーティション化されている場合は、list_values構文の下位ブランチを使用して、値リストを指定します。各値リストはカッコで囲まれ、キー列の値のリストを表します。新しいパーティションのうち、最初のものは指定のlist_valuesを使用して作成され、2つ目のパーティションは現行パーティションの残りのパーティション値を使用して作成されます。このため、値リストには現行パーティションのすべてのパーティション値を含めることはできません。また、現行パーティションに存在しないパーティション値も含めることはできません。

INTO句

INTO句を使用すると、分割の結果生成される新しいパーティションを記述できます。

- AT ... INTO句を使用すると、1つのレンジ・パーティションを2つのレンジ・パーティションに分割することによって生成されるパーティションについて記述できます。range_partition_descには、分割の結果としての2つのパーティションについて、オプションの名前と物理属性の指定を省略している場合でも、キーワードPARTITIONが必要になります。新しいパーティション名を指定しない場合、SYS_Pnという形式の名前が割り当てられます。指定しないすべての属性は、現行パーティションから継承されます。
- VALUES ... INTO句を使用すると、1つのリスト・パーティションを2つのリスト・パーティションに分割することによって生成されるパーティションについて記述できます。list_partition_descには、分割の結果としての2つのパーティションについて、オプションの名前と物理属性の指定を省略している場合でも、キーワードPARTITIONが必要になります。新しいパーティション名を指定しない場合、SYS_Pnという形式の名前が割り当てられます。指定しないすべての属性は、現行パーティションから継承されます。
- INTO句を使用すると、1つのレンジ・パーティションを2つ以上のレンジ・パーティションに分割できます。または、1つのリスト・パーティションを2つ以上のリスト・パーティションに分割できます。新しいパーティション名を指定しない場合、SYS_Pnという形式の名前が割り当てられます。指定しないすべての属性は、現行パーティションから継承されます。
    - 各レンジ・パーティションは、それらのパーティション境界を昇順で指定する必要があります。最初に指定するレンジ・パーティションのパーティション境界は、その次に低いパーティションが表内に存在する場合、そのパーティ

ション境界よりも大きくする必要があります。最後のレンジ・パーティションのパーティション境界は指定できません。これは、現在のパーティションのパーティション境界を継承します。

- リスト・パーティションの場合は、新しいパーティションに指定されたすべてのパーティション値が、現在のパーティション内に存在する必要があります。最後のパーティションには、パーティション値を指定できません。Oracle Databaseは、残りのパーティション値を使用して、現在のパーティションから最後のパーティションを作成します。

レンジ-ハッシュ・コンポジット・パーティション表の場合、新しいパーティションにサブパーティションを指定するとき、サブパーティションに対してTABLESPACEおよび表の圧縮のみを指定できます。他のすべての属性は、現行パーティションから継承されます。新しいパーティションにサブパーティション化を指定しない場合は、表領域も現行パーティションから継承されます。

レンジ-リストおよびリスト-リスト・コンポジット・パーティション表の場合、新しいパーティションにサブパーティションを指定できません。分割パーティションのリスト・サブパーティションでは、サブパーティションの数および値リストは現行パーティションから継承されます。

新しく作成したサブパーティションに対して名前を指定しなかったすべてのコンポジット・パーティション表では、次のように名前が親パーティションから継承されます。

- 親パーティション内のサブパーティションが`partition_name`アンダースコア(_)`subpartition_name`という形式の名前(たとえば、`P1_SUBP1`)を持つ場合、新しいパーティション名に基づいて、サブパーティションの名前が生成されます(たとえば、`P1A_SUB1`や`P1B_SUB1`)。
- 親パーティション内のサブパーティションがその他の形式の名前を持つ場合、SYS_SUBPnという形式のサブパーティション名が生成されます。

索引にUNUSABLEのマークが付いている場合でも、`table`で定義されている各ローカル索引の対応するパーティションが分割されます。

`table`が参照パーティション表の親表の場合は、`dependent_tables_clause`を使用して、この文に指定するパーティション・メンテナンス操作を参照パーティション表のすべての子表に伝播できます。

ヒープ構成表のすべての索引が無効になります。update_index_clausesを使用し、操作中にこれらの索引を更新できます。

`parallel_clause`を使用すると、表のデフォルトのパラレル属性を変更せずに、分割操作をパラレル化できます。

ONLINE

`ONLINE`を指定すると、表パーティションの分割中に、表に対するDML操作が許可されるようになります。

ONLINE句の制限事項

表パーティションの分割時には、`ONLINE`句に次の制限事項が適用されます。

- `ONLINE`句は、SYSが所有する表に対しては指定できません。
- `ONLINE`句は、索引構成表には指定できません。
- `ONLINE`句は、表でドメイン索引が定義されている場合は指定できません。
- `ONLINE`句は、オブジェクト型を格納するヒープ構成表や、ビットマップ結合索引が定義されているヒープ構成表には指定できません。
- パラレルDMLおよびダイレクト・パス`INSERT`操作は、表に対する排他的ロックを必要とします。したがって、ロックが競合するため、これらの操作は実行中のオンライン・パーティション分割と同時にはサポートされません。

表パーティションの分割の制限事項

表パーティションの分割には、次の制限事項があります。

- この句は、ハッシュ・パーティションには指定できません。
- parallel_clauseは、索引構成表には指定できません。
- tableが索引構成表の場合や、ローカル・ドメイン索引がtableに定義されている場合は、パーティションは2つの新しいパーティションにのみ分割可能です。

関連項目:

表パーティションの分割: 例

split_table_subpartition

この句を使用すると、1つのサブパーティションを、重複しない値リストを持つ複数の新しいサブパーティションに分割できます。

ノート:

一定の条件が満たされると、SPLIT PARTITION および SPLIT SUBPARTITION 操作を最適化して処理速度を上げることができます。これらの操作の最適化の詳細は、『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。

AT句

AT句は、レンジ・サブパーティションに対してのみ有効です。新しい2つのサブパーティションのうちの1つ目に、新しく上限(この値は含まない)を指定します。値リストは、subpartition_extended_nameで指定されるサブパーティションの元のサブパーティション境界より小さく、その次に小さいサブパーティション(そのようなパーティションがある場合)のパーティション境界より大きい値にする必要があります。

VALUES句

VALUES句は、リスト・サブパーティションに対してのみ有効です。1つのキー列で表がサブパーティション化されている場合は、list_values構文の上位ブランチを使用して、その列の値リストを指定します。NULLを指定できるのは、まだ同じパーティション内の別のサブパーティションに対してNULLが指定されていない場合です。複数のキー列で表がサブパーティション化されている場合は、list_values構文の下位ブランチを使用して、値リストを指定します。各値リストはカッコで囲まれ、キー列の値のリストを表します。新しいサブパーティションのうち、最初のものは指定のサブパーティション値リストを使用して作成され、2つ目のパーティションは、現行サブパーティションの残りのパーティション値を使用して作成されます。このため、値リストには現行のサブパーティションのすべてのパーティション値を含めることはできません。また、現行のサブパーティションに存在しないパーティション値も含めることはできません。

INTO句

INTO句を使用すると、分割の結果生成される新しいサブパーティションを記述できます。

- AT ... INTO句を使用すると、1つのレンジ・パーティションを2つのレンジ・パーティションに分割することによって生成される2つのサブパーティションについて記述できます。2つの新しいサブパーティションのオプションの名前と属性を指定しない場合でも、range_subpartition_descには、キーワードSUBPARTITIONが必要になります。新しいサブパーティションの名前を省略すると、Oracle Databaseは、SYS_SUBPnの形式の名前を割り当てます。指定されていな

い属性があると、その属性は現在のサブパーティションから継承されます。

- VALUES ... INTO句を使用すると、1つのリスト・パーティションを2つのリスト・パーティションに分割することによって生成される2つのサブパーティションについて記述できます。2つの新しいサブパーティションについてのオプションの名前と属性を指定しない場合でも、list_subpartition_descには、キーワードSUBPARTITIONが必要になります。新しいサブパーティションの名前を省略すると、Oracle Databaseは、SYS_SUBPnの形式の名前を割り当てます。指定されていない属性があると、その属性は現在のサブパーティションから継承されます。
- INTO句を使用すると、1つのレンジ・サブパーティションを2つ以上のレンジ・サブパーティションに分割できます。または、1つのリスト・サブパーティションを2つ以上のリスト・サブパーティションに分割できます。新しいサブパーティション名を指定しない場合、SYS_SUBPnという形式の名前が割り当てられます。指定しないすべての属性は、現行のサブパーティションから継承されます。
    - 各レンジ・サブパーティションは、それらのサブパーティション境界を昇順で指定する必要があります。最初に指定するレンジ・サブパーティションのサブパーティション境界は、その次に低いサブパーティションが存在する場合、そのサブパーティション境界よりも大きくする必要があります。最後のレンジ・サブパーティションのサブパーティション境界は指定できません。これは、現在のサブパーティションのパーティション境界を継承します。
    - リスト・サブパーティションの場合は、新しいサブパーティションに指定されたすべてのサブパーティション値が、現在のサブパーティション内に存在する必要があります。最後のサブパーティションには、サブパーティション値を指定できません。Oracle Databaseは、残りのパーティション値を使用して、現在のサブパーティションから最後のサブパーティションを作成します。

対応するローカル索引サブパーティションにUNUSABLEのマークが付いている場合でも、それらは分割されます。新しい索引サブパーティションに名前があらかじめ指定されていないかぎり、新しい表のサブパーティションの名前が継承されます。この場合、SYS_SUBPnという形式の新しい索引サブパーティション名が割り当てられます。新しい索引サブパーティションの物理属性は、親サブパーティションから継承されます。親サブパーティションにデフォルトのTABLESPACE属性が定義されていない場合、対応する新しい表のサブパーティションの表領域が継承されます。

ヒープ構成表の索引が無効になります。update_index_clausesを使用し、これらの索引を更新できます。

ONLINE

ONLINEを指定すると、表サブパーティションの分割中に、表に対するDML操作が許可されるようになります。

ONLINE句の制限事項

表サブパーティションを分割するためのONLINE句には、表パーティションを分割するためのONLINE句と同じ制限事項があります。「ONLINE句の制限事項」を参照してください。

表サブパーティションの分割の制限事項

表サブパーティションの分割には、次の制限事項があります。

- この句は、ハッシュ・サブパーティションに対して指定できません。
- サブパーティションの定義で指定できるpartitioning_storage_clauseの句は、TABLESPACEおよび表の圧縮のみです。
- parallel_clauseは、索引構成表には指定できません。
- tableが索引構成表の場合は、そのサブパーティションは2つの新しいサブパーティションにのみ分割可能です。

merge_table_partitions

merge_table_partitions句を使用すると、tableの2つ以上のレンジ・パーティション、リスト・パーティション、またはシステム・パーティションの内容を1つの新しいパーティションにマージして、元のパーティションを削除できます。この句はハッシュ・パーティションでは無効です。かわりに、coalesce_table_partition句を使用してください。

マージする2つ以上のレンジ・パーティション、リスト・パーティション、またはシステム・パーティションのカンマ区切りリストを指定します。TO句を使用すると、マージする2つ以上の隣接しているレンジ・パーティションを指定できます。

各パーティションに、partitionを使用してパーティション名を指定するか、またはFOR句を使用して名前なしでパーティションを指定します。FOR句の詳細は、「パーティション表と索引の参照」を参照してください。

- マージするパーティションは、隣接している必要があり、そのパーティションがレンジ・パーティションの場合は、パーティション境界を昇順で指定する必要があります。リスト・パーティションおよびシステム・パーティションをマージする場合は、隣接している必要はありません。
- レンジ・パーティションをマージすると、新しいパーティションは、元のパーティションのうち上位のパーティションのパーティション境界を継承します。
- リスト・パーティションをマージする場合、結果のパーティションの値リストは、マージされるパーティションの値リストの集合をあわせたものです。DEFAULTのリスト・パーティションを別のリスト・パーティションとマージすると、結果のパーティションはDEFAULTパーティションになり、DEFAULT値が含まれます。
- コンポジット・レンジ・パーティションまたはコンポジット・リスト・パーティション、レンジ-リストまたはリスト-リスト・コンポジット・パーティションをマージする場合、サブパーティションの定義は指定できません。サブパーティション化情報は、サブパーティション・テンプレートから取得できます。サブパーティションのテンプレートが指定されていない場合は、1つのMAXVALUEサブパーティションがレンジ・サブパーティションから、または1つのDEFAULTサブパーティションがリスト・サブパーティションから作成されます。

新しいパーティションに対して明示的に指定しなかった属性は、表レベルのデフォルトから継承されます。ただし、新しいパーティションに対してパーティション名を再利用すると、表レベルのデフォルト値ではなく、名前が再利用されたパーティションの値が新しいパーティションに継承されます。

選択したパーティションに対応するローカル索引パーティションは削除され、マージされたパーティションに対応するローカル索引パーティションにUNUSABLEのマークが付けられます。ヒープ構成表のすべてのグローバル索引にも、UNUSABLEのマークが付けられます。update_index_clausesを使用し、操作中にこれらすべての索引を更新できます。

tableが参照パーティション表の親表の場合は、dependent_tables_clauseを使用して、この文に指定するパーティション・メンテナンス操作を参照パーティション表のすべての子表に伝播できます。

ONLINE

パーティションのマージ操作中に表パーティションに対するDML操作を許可するには、ONLINEを指定します。

表パーティションのマージの制限事項

tableが索引構成表の場合や、ローカル・ドメイン索引がtableに定義されている場合は、一度にマージできるパーティションは2つのみになります。

関連項目:

「2つの表パーティションのマージ: 例」、「隣接する4つのレンジ・パーティションのマージ例」および「デフォルト・リスト・パーティションの使用: 例」を参照してください。

merge_table_subpartitions

merge_table_subpartitions句を使用すると、tableの2つ以上のレンジ・サブパーティションまたはリスト・サブパーティションの内容を1つの新しいサブパーティションにマージして、元のサブパーティションを削除できます。この句はハッシュ・サブパーティションでは無効です。かわりに、coalesce_hash_subpartition句を使用してください。

マージする2つ以上のレンジ・サブパーティション、またはリスト・サブパーティションのカンマ区切りリストを指定します。TO句を使用すると、マージする2つ以上の隣接しているレンジ・サブパーティションを指定できます。

各サブパーティションに、subpartitionを使用してサブパーティション名を指定するか、またはFOR句を使用して名前なしでサブパーティションを指定します。FOR句の詳細は、「パーティション表と索引の参照」を参照してください。

マージされるサブパーティションは、同じパーティションに属している必要があります。レンジ・サブパーティションである場合は、隣接している必要があります。リスト・サブパーティションである場合は、隣接している必要はありません。マージの結果生成されるサブパーティション内のデータは、マージされたサブパーティションのデータが結合されたものです。

INTO句を指定する場合、range_subpartition_descまたはlist_subpartition_descで、それぞれrange_values_clauseまたはlist_values_clauseを指定することはできません。また、partitioning_storage_clauseに指定できる句は、TABLESPACEおよびtable_compressionのみです。

新しいサブパーティションに対して明示的に指定しなかった属性は、パーティション・レベルの値から継承されます。ただし、新しいサブパーティションに対してサブパーティション名を再利用すると、パーティション・レベルのデフォルト値ではなく、名前が再利用されたサブパーティションの値が新しいサブパーティションに継承されます。

対応するローカル索引のサブパーティションがマージされ、結果として生成される索引サブパーティションにはUNUSABLEのマークが付けられます。また、ヒープ構成表のパーティション化されたグローバル索引とパーティション化されていないグローバル索引の両方に対して、UNUSABLEのマークが付けられます。update_index_clausesを使用すると、操作中にすべての索引を更新できます。

ONLINE

サブパーティションのマージ操作中に表サブパーティションに対するDML操作を許可するには、ONLINEを指定します。

表サブパーティションのマージの制限事項

tableが索引構成表の場合は、一度にマージできるサブパーティションは2つのみになります。

exchange_partition_subpart

EXCHANGE PARTITION句またはEXCHANGE SUBPARTITION句を使用すると、次のデータおよび索引セグメントを交換できます。

- 1つの非パーティション表と、次のいずれかを交換できます。
  - 1つのレンジ・パーティション、リスト・パーティションまたはハッシュ・パーティション
  - 1つのレンジ・サブパーティション、リスト・サブパーティションまたはハッシュ・サブパーティション
- 1つのレンジ・パーティション表とレンジ-レンジまたはリスト-レンジ・コンポジット・パーティション表パーティションのレンジ・サブパーティション
- 1つのハッシュ・パーティション表とレンジ-ハッシュまたはリスト-ハッシュのコンポジット・パーティション表パーティションのハッシュ・サブパーティション
- 1つのリスト・パーティション表とレンジ-リストまたはハッシュ-リストのコンポジット・パーティション表パーティションのリスト・サブパーティション

交換対象の表、パーティションおよびサブパーティションの構造は、パーティション・キーを含め常に同じである必要があります。リス

ト・パーティションとリスト・サブパーティションの場合、対応する値リストも一致している必要があります。

この句をトランスポータブル表領域とともに使用すると、高速データ・ロードが容易になります。

関連項目:

トランスポータブル表領域の詳細は、『Oracle Database管理者ガイド』を参照してください。

tableにLOB列がある場合、各LOB列のLOBデータ、およびLOB索引パーティション・セグメントまたはLOB索引サブパーティション・セグメントは、tableの対応するLOBデータおよびLOB索引セグメントと交換されます。

tableにネストした表の列が含まれる場合、そのような各列のネストした表パーティション・セグメントは、非パーティション表の対応するネストした表セグメントと交換されます。

tableにID列が格納されている場合、パーティションまたはサブパーティションが交換されることになります。その逆も同様になります。順序ジェネレータは、増加または減少になります。順序ジェネレータは交換されないため、tableとパーティションまたはサブパーティションは、同じ順序ジェネレータを引き続き使用することになります。両方の順序ジェネレータの最高水位標は、新しいID列の値が既存の値と競合しないようにが調整されます。

2つのオブジェクトのすべてのセグメント属性(表領域およびロギングを含む)も交換されます。

パーティション表に交換される表の既存の統計情報は、交換されます。ただし、パーティション化された表のグローバル統計は変更されません。DBMS_STATS.GATHER_TABLE_STATSプロシージャを使用すると、グローバル統計を再作成できます。GRANULARITY属性をAPPROX_GLOBAL AND PARTITIONと同じに設定して、処理速度を上げ、既存のパーティション統計に基づいて新しいグローバル統計情報を集計できます。このパッケージ・プロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

交換されるオブジェクトのすべてのグローバル索引は、無効になります。update_global_index_clauseまたはupdate_all_indexes_clauseを使用し、パーティションが交換された表のグローバル索引を更新できます。update_all_indexes_clauseには、副次句ではなくUPDATE INDEXESキーワードのみを指定できます。交換される表のグローバル索引は、無効のままになります。update_global_index_clauseおよびupdate_all_indexes_clauseは、交換操作中にローカル索引を更新しません。INCLUDING | EXCLUDING INDEXES句を使用して、ローカル索引メンテナンスを指定できます。これらの句とともにparallel_clauseを指定すると、交換操作ではなく、索引の更新がパラレル化されます。

関連項目:

パーティションおよびサブパーティションの交換のノート

WITH TABLE

パーティションまたはサブパーティションを交換するtableを指定します。schemaを指定しない場合、そのtableは自分のスキーマ内にあるとみなされます。

INCLUDING | EXCLUDING INDEXES

INCLUDING INDEXESを指定すると、ローカル索引パーティションまたはサブパーティションを対応する表索引(非パーティション表の場合)またはローカル索引(ハッシュ・パーティション表の場合)と交換できます。EXCLUDING INDEXESを指定すると、交換された表のすべての標準索引、索引パーティションおよびパーティションに対応するすべての索引パーティションまたはサブパーティションにUNUSABLEのマークを付けることができます。この句を指定しない場合、EXCLUDING INDEXESがデフォルトになります。

WITH | WITHOUT VALIDATION

WITH VALIDATIONを指定すると、交換された表にあるいずれかの行が交換されたパーティションまたはサブパーティションにマップされない場合にエラーが戻されます。WITHOUT VALIDATIONを指定すると、交換された表にある行が正しくマップされたかどうかがチェックされません。この句を指定しない場合、WITH VALIDATIONがデフォルトになります。

exceptions_clause

この句の詳細は、「制約の例外の処理」を参照してください。パーティション交換のコンテキストでは、この句は、パーティション表がUNIQUE制約を使用して定義されている場合にのみ有効です。また、制約はDISABLE VALIDATEの状態である必要があります。この句は、サブパーティションではなくパーティション交換に対してのみ有効です。

CASCADE

CASCADEを指定すると、tableに含まれるすべての子の参照パーティション表の対応するパーティションまたはサブパーティションを交換できます。参照パーティション表の階層は、ソースとターゲットで一致している必要があります。

CASCADEの制限事項

CASCADE句には、次の制限事項が適用されます。

- 参照パーティション表階層の親キーが、複数のパーティション化の制約で参照されている場合、CASCADEは指定できません。
- tableの参照パーティション子表でドメイン索引またはXMLIndex索引が定義されている場合、CASCADEは指定できません。

関連項目:

- SQLスクリプトの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』のDBMS_IOTパッケージを参照してください。
- 移行行および連鎖行の削除については、『Oracle Database管理者ガイド』を参照してください。
- 制約の確認の詳細は、「constraint」および「索引構成表の例外表の作成: 例」を参照してください。

パーティションおよびサブパーティションの交換のノート

パーティションおよびサブパーティションの交換時には、次のノートが適用されます。

- 交換される両方の表は同じ主キーを含む必要があり、参照表が空でないかぎり、どちらの表も有効な外部キーを参照できません。
- パーティション化された索引構成表の交換時には、次の点に注意します。
    - ソースおよびターゲットの表およびパーティションは、その主キーが同じ列に同じ順序で設定されている必要があります。
    - 接頭辞圧縮が使用可能な場合は、ソースおよびターゲットの両方で使用可能で、接頭辞の長さは同じである必要があります。
    - ソースおよびターゲットの両方は、索引構成されている必要があります。
    - ソースおよびターゲットの両方に、オーバーフロー・セグメントが必要です。または、ソースおよびターゲットの両方がオーバーフロー・セグメントを持ってはいけません。また、ソースおよびターゲットの両方ともマッピング表を含むか、または両方とも含まない必要があります。

- ソースおよびターゲットの両方は、LOB列に対して記憶域属性が同一である必要があります。

関連項目:

表パーティションの交換: 例

dependent_tables_clause

この句は、参照パーティション表の親表を変更する場合にのみ有効です。この句では、親表の子の参照パーティション表の操作によって作成されるパーティションの属性を指定できます。

- 親表がコンポジット・パーティションではない場合、1つ以上の子表を指定し、子表ごとに、親表内に作成された各パーティションの1つの`partition_spec`を指定します。
- 親表がコンポジットの場合、1つ以上の子表を指定し、子表ごとに、親表内に作成された各サブパーティションの1つの`partition_spec`を指定します。

関連項目:

参照パーティション表の作成の詳細は、「CREATE TABLE」の句「reference_partitioning」を参照してください。参照によるパーティション化の概要は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

UNUSABLE LOCAL INDEXES句

この2つの句を使用すると、パーティションとサブパーティションのどちらを変更するかに応じて、`partition`に対応するローカル索引パーティションおよび索引サブパーティションの属性を変更できます。

- `UNUSABLE LOCAL INDEXES`を指定すると、`partition`に関連付けられたローカル索引パーティションまたは索引サブパーティションに、`UNUSABLE`のマークが付けられます。
- `REBUILD UNUSABLE LOCAL INDEXES`を指定すると、`partition`に関連付けられたUNUSABLEのローカル索引パーティションまたは索引サブパーティションが再構築されます。

UNUSABLE LOCAL INDEXESの制限事項

この句には、次の制限事項があります。

- この句を`modify_table_partition`の他の句と同時に指定することはできません。
- サブパーティションが含まれるパーティションに対しては、`modify_table_partition`でこの句を指定できません。ただし、`modify_table_subpartition`句では指定できます。

update_index_clauses

`update_index_clauses`を使用すると、表のパーティション化操作の一部として`table`の索引を更新できます。表パーティションでDDL文を実行する場合、`table`に索引が定義されていると、DDL文を実行中のパーティションだけでなく索引全体が無効になります。この句を使用すると、変更する索引パーティションをDDL操作中に更新できるため、DDL文の後で索引を再構築する必要がなくなります。

パーティション化された索引構成表では、`update_index_clauses`は不要であり、無効です。索引構成表は主キー・ベースであるため、値を変更せずにデータを移動する操作では、グローバル索引はUSABLEのままです。

update_global_index_clause

この句を指定すると、tableのグローバル索引のみを更新できます。tableのすべてのローカル索引には、UNUSABLEのマークが付けられます。

UPDATE GLOBAL INDEXES

UPDATE GLOBAL INDEXESを指定すると、tableで定義したグローバル索引を更新できます。

グローバル索引の更新の制限事項

グローバル索引がLOB列のグローバル・ドメイン索引として定義されている場合、ドメイン索引は、更新されるのではなくUNUSABLEのマークが付けられます。

INVALIDATE GLOBAL INDEXES

INVALIDATE GLOBAL INDEXESを指定すると、tableで定義したグローバル索引を無効にできます。

どちらも指定しない場合、グローバル索引は無効になります。

グローバル索引の無効化の制限事項

この句は、グローバル索引でのみサポートされます。索引構成表に対してはサポートされません。また、この句によって更新されるのは、USABLEおよびVALIDの索引のみです。UNUSABLEの索引は使用禁止のままになり、INVALIDのグローバル索引は無視されます。

update_all_indexes_clause

この句を使用すると、tableのすべての索引を更新できます。

update_index_partition

この句は、表パーティションに対する操作でのみ有効で、ローカル索引のみに影響します。

- index_partition_descriptionを使用すると、各ローカル索引のそれぞれのパーティションの物理属性、表領域の記憶域およびロギングを指定できます。PARTITIONキーワードのみを指定すると、索引のパーティションは次のように更新されます。
    - 単一の表パーティションに対する操作(MOVE PARTITION、SPLIT PARTITIONなど)の場合、対応する索引パーティションは処理された索引表パーティションの属性を継承します。索引パーティションの名前は生成されないため、この操作によって作成された新しい索引パーティションは、対応する新しい表パーティションから名前を継承します。
    - MERGE PARTITION操作の場合、この操作によって作成されたローカル索引パーティションは、作成された表パーティションの名前とローカル索引の属性を継承します。

  ドメイン索引の場合、PARAMETERS句を使用すると、未解析のまま適切なODCI索引タイプ・ルーチンに渡すパラメータ文字列を指定できます。PARAMETERS句は、ドメイン索引に対してのみ有効であり、ドメイン索引について指定できるindex_partition_descriptionの唯一の部分です。

  関連項目:

  ドメイン索引の詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。

- コンポジット・パーティション索引の場合、index_subpartition_clauseを使用すると、各サブパーティションに対して表領域の記憶域を指定できます。update_index_partition句のこのコンポーネントの詳細は、「CREATE INDEX」の「index_subpartition_clause」を参照してください。

USABLEキーワードとUNUSABLEキーワードの詳細は、ALTER INDEX ... USABLE | UNUSABLEを参照してください。

update_index_subpartition

この句は、コンポジット・パーティション表のサブパーティションに対する操作でのみ有効で、コンポジット・パーティション表のローカル索引のみに影響します。1つ以上のサブパーティションに対して、格納する表領域を指定できます。

すべての索引の更新の制限事項

update_all_indexes_clauseには、次の制限事項が適用されます。

- この句は、索引構成表に対して指定できません。
- exchange_partition_subpart句を使用してパーティションまたはサブパーティションを交換する場合、update_all_indexes_clauseは、グローバル索引にのみ適用可能です。このため、update_index_partitionまたはupdate_index_subpartition句を指定できません。ただし、INCLUDING | EXCLUDING INDEXES句を使用して、交換操作中にローカル索引メンテナンスを指定できます。

関連項目:

「グローバル索引の更新: 例」および「パーティション索引の更新: 例」を参照してください。

parallel_clause

parallel_clauseを使用すると、表の問合せおよびDMLに対するデフォルトの並列度を変更できます。

この句の詳細は、「CREATE TABLE」の「parallel_clause」を参照してください。

表のパラレル化の変更の制限事項

パラレル化の変更には、次の制限事項があります。

- tableにLOB型またはユーザー定義オブジェクト型の列が含まれている場合、このtableでのINSERT、UPDATEおよびDELETEは、通知なしに逐次実行されます。ただし、後続の問合せはパラレルで実行されます。
- parallel_clauseをmove_table_clauseと組み合せて指定する場合、このパラレル化は移動のみに適用され、後続の表でのDML操作および問合せには適用されません。

関連項目:

パラレル処理の指定: 例

filter_condition

この句を使用すると、表のパーティションまたはサブパーティションの移動、分割またはマージ、表の移動または非パーティション表のパーティション表への変換の各ALTER TABLE操作の実行中に保持する行を指定できます。データベースでは、where_clauseで指定された条件を満たす行のみが保持されます。この句のセマンティクスの詳細は、「SELECT」の「where_clause」を参照してください。

フィルタ条件の制限事項

filter_condition句には、次の制限事項が適用されます。

- フィルタ条件は、ヒープ構成表でのみサポートされます。

- フィルタ条件が参照できるのは、変更される表の列のみです。フィルタ条件に、結合や副問合せなど、他のデータベース・オブジェクトを参照する操作を含めることはできません。
- フィルタ条件は、有効化された外部キーによって参照される主キーまたは一意キーがある表ではサポートされません。

オンライン操作でのフィルタ条件の使用の制限事項およびノート

オンラインALTER TABLE操作のフィルタ条件の指定時には、次の制限事項およびノートが適用されます。

- サプリメンタル・ロギングが有効になっている場合は、filter_condition句およびONLINE句の両方を指定することはできません。
- filter_condition句およびONLINE句の両方を指定すると、ALTER TABLE操作の実行中に表に対するDML操作が許可されます。フィルタ条件による、同時DML操作に対する直接の影響はありません。ただし、フィルタ操作とDML操作は意図せずに次のような競合を引き起こす可能性があるため、この組合せは慎重に使用してください。
  - 非パーティション表への挿入は成功します。パーティション表への挿入は、この操作がパーティション化キー基準に違反しない場合は成功します。
  - 削除操作は、ALTER TABLE操作の間、フィルタ条件により保持対象となる行に対してのみ適用されます。
  - 更新操作は、ALTER TABLE操作の間、フィルタ条件により保持対象となる行に対してのみ適用されます。これらの更新操作は、更新操作によって、フィルタ条件により保持対象となる行が対象外になるかどうかにかかわらず、成功します。
  - ALTER TABLE操作の開始時にフィルタ条件による保持対象ではなかった行は、更新操作によってその行が保持対象になるかどうかにかかわらず、保持されません。

allow_disallow_clustering

この句は、属性クラスタリングを使用する表に有効です。move_table_clauseで指定された表の移動操作とcoalesce_table_[sub]partition、merge_table_[sub]partitions、move_table_[sub]partitionおよびsplit_table_[sub]partition句で指定された表のパーティションおよびサブパーティション・メンテナンス操作中に発生するデータ移動の属性クラスタリングを許可または禁止できます。

- ALLOW CLUSTERINGを指定して、データ移動の属性クラスタリングを許可します。この句は、表を作成または変更したDDLのNO ON DATA MOVEMENT設定をオーバーライドします。
- DISALLOW CLUSTERINGを指定して、データ移動の属性クラスタリングを禁止します。この句は、表を作成または変更したDDLのYES ON DATA MOVEMENT設定をオーバーライドします。

属性クラスタリングを使用しない表に指定する場合、allow_disallow_clustering句は影響しません。

関連項目:

NO ON DATA MOVEMENTおよびYES ON DATA MOVEMENT句の詳細は、CREATE TABLEのclustering_when句を参照してください。

{ DEFERRED | IMMEDIATE } INVALIDATION

この句を使用すると、データベースで表パーティションのメンテナンス操作の実行中に、依存カーソルが無効化されるタイミングを制御できます。

- DEFERRED INVALIDATIONを指定した場合、データベースでは可能な場合、依存カーソルの無効化を回避また

は遅延します。

- IMMEDIATE INVALIDATIONを指定した場合、データベースでは、Oracle Database 12cリリース1 (12.1)以前のリリースでの動作と同様に、依存カーソルを即時に無効にします。これはデフォルトです。

この句を省略した場合、いつカーソルが無効になるかは、CURSOR_INVALIDATION初期化パラメータの値によって決定されます。

この句を指定できるのは、表パーティションのメンテナンス操作を実行する場合のみで、その他のALTER TABLE操作に対してはサポートされません。

関連項目:

- カーソルの無効化の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。
- CURSOR_INVALIDATION初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

move_table_clause

move_table_clauseを使用すると、非パーティション表のデータまたはパーティション表のパーティションのデータを新しいセグメントに再配置できます。オプションとして、別の表領域への配置および記憶域属性の変更を行うこともできます。

LOB_storage_clause句およびvarray_col_properties句を使用して、表またはパーティションに関連付けられたLOBデータ・セグメントを移動することもできます。この句で指定していないLOB項目は移動できません。

表を別の表領域に移動する場合、COMPATIBLEパラメータが10.0以上に設定されていると、ネストした表の列に対する記憶表は、その表が作成された表領域に残ります。COMPATIBLEが10.0未満に設定されている場合、表と記憶表は新しい表領域に自動的に移動します。

ONLINE句

ONLINEを指定すると、表の移動中に、表に対してDML操作を実行できます。

表のオンライン化の制限事項

表のオンライン化には、次の制限事項があります。

- 同じ文でこの句と他の句は結合できません。
- この句は、パーティション化された索引構成表に対して指定できません。
- 空間、XML、テキスト索引などの表でドメイン索引が定義されている場合、この句を指定することはできません。
- パラレルDMLおよびダイレクト・パスINSERT操作は、表に対する排他的ロックを必要とします。したがって、ロックが競合するため、これらの操作は実行中のオンライン表MOVEと同時にサポートされません。
- LOB、VARRAY、Oracleが提供する型またはユーザー定義オブジェクト型の列が含まれている索引構成表には、この句は指定できません。

index_org_table_clause

索引構成表の場合は、move_table_clauseのindex_org_table_clauseを使用するとオーバーフロー・セグメント属性も指定できます。move_table_clauseを指定すると、索引構成表の主キー索引が再構築されます。オーバーフロー・データ・セグメントは、OVERFLOWキーワードが明示的に指定されていないかぎり、再構築されません。ただし、次の場合は例外です。

- ALTER TABLE文の一部としてPCTTHRESHOLDの値またはINCLUDING列を変更する場合は、オーバーフロー・データ・セグメントが再構築されます。
- 索引構成表内の表外列(LOB列、VARRAY列、ネストした表の列)のいずれかを明示的に移動する場合は、オーバーフロー・データ・セグメントも再構築されます。

LOB列の索引およびデータ・セグメントは、LOB列をALTER TABLE文の一部として明示的に指定しないかぎり、再構築されません。

mapping_table_clause

MAPPING TABLEを指定すると、マッピング表がまだ存在しない場合、Oracle Databaseによりマッピング表が作成されます。マッピング表がすでに存在する場合、マッピング表は索引構成表とともに移動され、すべてのビットマップ索引にはUNUSABLEのマークが付けられます。新しいマッピング表は、親表と同じ表領域に作成されます。

NOMAPPINGを指定すると、既存のマッピング表が削除されます。

この句の詳細は、「CREATE TABLE」の「mapping_table_clauses」を参照してください。

マッピング表の制限事項

tableでビットマップ索引が定義されている場合は、NOMAPPINGを指定できません。

prefix_compression

prefix_compression句を使用すると、索引構成表の接頭辞圧縮を有効または無効にできます。

- COMPRESSを指定すると、接頭辞圧縮が使用可能になり、その結果、索引構成表における主キー列の値が重複しなくなります。integerを使用して、接頭辞の長さ(圧縮する接頭辞列数)を指定します。

  接頭辞の長さの有効範囲は、1から(主キー列数-1)までです。デフォルトでは(主キー列数-1)になります。
- NOCOMPRESSを指定すると、索引構成表での接頭辞圧縮が使用禁止になります。これはデフォルトです。

TABLESPACE tablespace

再構築した索引構成表を格納する表領域を指定します。

LOB_storage_clause

この句を使用すると、別の表領域にLOBセグメントを移動できます。表にLONG列が含まれている場合は、この句を使用してLOBセグメントを移動することはできません。かわりに、LONG列をLOBに変換するか、または表をエクスポートし、LOB列に必要な表領域の記憶域を指定する表を再作成して表データを再インポートする必要があります。

UPDATE INDEXES

この句は、ヒープ構成表のオンラインまたはオフライン移動の実行時にのみ有効です。これを使用して、表のすべてのグローバル索引を更新できます。

オプションで、索引または索引パーティションの表領域を次のように変更できます。

- segment_attributes_clauseを指定すると、非パーティション・グローバル索引の表領域を変更できます。この句内で指定できるのはTABLESPACE句のみです。
- update_index_partition句を指定すると、パーティション・グローバル索引のパーティションの表領域を変更できます。この句内で指定できるのは、segment_attributes_clauseのTABLESPACE句のみです。

表の移動の制限事項

表の移動には、次の制限事項があります。

- MOVEを指定する場合は、ALTER TABLE文の最初の句にする必要があります。この句以外では、physical_attributes_clause、parallel_clauseおよびLOB_storage_clauseのみが指定できます。
- LONGまたはLONG RAW列を含む表は、移動できません。
- パーティション表(ヒープ表または索引構成表)全体の移動はできません。個々のパーティションまたはサブパーティションを移動してください。

ノート:

move_table_clause で指定するすべての LOB 列については、次のことに注意してください。

- 新しい表領域が指定されていない場合でも、古い LOB データ・セグメントとこれに対応する索引セグメントは削除され、新しいセグメントが作成されます。
- table の LOB 索引がその LOB データと異なる表領域にある場合、移動後の LOB 索引は、LOB データと同じ表領域にまとめて格納されます。

関連項目:

「move_table_partition」および「move_table_subpartition」を参照してください。

modify_to_partitioned

この句を使用すると、非パーティション表またはパーティション表を、索引を含めてオンラインまたはオフラインでパーティション化できます。

非パーティション表またはパーティション表を、次の特性を持つ任意のタイプのパーティション表またはコンポジット・パーティション表に変更できます。

- 元の表内のすべてのデータが保持されます。
- 新しく作成されたパーティションのデータまたは変更された表のサブパーティションは、table_partitioning_clausesで別途指定しないかぎり、元の表と同じ表領域に格納されます。
- ローカル索引パーティションまたはサブパーティションとLOBパーティションまたはサブパーティションは、table_partitioning_clausesで別途指定しないかぎり表パーティションまたはサブパーティションと同じ場所に配置されます。
- 元の表で定義されているすべてのトリガー、制約およびVPDポリシーが保持されます。
- 元の非パーティション表で表圧縮が定義されている場合、パーティション表でも同じタイプの表圧縮が使用されます。
- パーティション表を変更する場合、新しく作成するパーティションまたはサブパーティションの圧縮設定は、すべてのパーティションまたはサブパーティションが同じ圧縮方法を共有していないかぎり、変更前のパーティション表のデフォルトの圧縮設定から導出されます。

modify_to_partitioned句で指定された、文字データ型のレンジ、リストまたはハッシュ・パーティション化キー列またはサ

ブパーティション化キー列のそれぞれに、BINARY、USING_NLS_COMP、USING_NLS_SORTまたはUSING_NLS_SORT_CSのいずれかの宣言された照合が必要です。

table_partitioning_clauses

この句を使用すると、表のパーティション化属性を指定できます。

modify_to_partitioned句で指定された、文字データ型のレンジ、リストまたはハッシュ・パーティション化キー列またはサブパーティション化キー列のそれぞれに、BINARY、USING_NLS_COMP、USING_NLS_SORTまたはUSING_NLS_SORT_CSのいずれかの宣言された照合が必要です。

この句のセマンティクスは、CREATE TABLE文と同じです。この句のセマンティクスの詳細は、「CREATE TABLE」の「table_partitioning_clauses」を参照してください。

NONPARTITIONED

パーティション表を非パーティション状態に戻すには、NONPARTITIONEDを指定します。

ONLINE

ONLINEを指定すると、パーティション表の変更中に、表に対するDML操作が許可されるようになります。

UPDATE INDEXES

この句を使用すると、表の既存の索引が、グローバル・パーティション索引またはローカル・パーティション索引にどのように変換されるかを指定できます。

- indexには、表の既存の索引の名前を指定します。
- local_partitioned_index句を指定すると、indexをローカル・パーティション索引に変換できます。この句のセマンティクスは、CREATE INDEX文と同じです。この句のセマンティクスの詳細は、「CREATE INDEX」の句「local_partitioned_index」を参照してください。
- global_partitioned_index句を指定すると、indexをグローバル・パーティション索引に変換できます。この句のセマンティクスは、CREATE INDEX文と同じです。この句のセマンティクスの詳細は、「CREATE INDEX」の句「global_partitioned_index」を参照してください。
- GLOBALキーワードを指定すると、同一キーのパーティション・グローバル索引および非パーティション・グローバル索引がグローバル形状を保持できるようになります。この句は、それらの索引がローカル・パーティション索引に変換されないようにします。非同一グローバル索引には効果がありません。

UPDATE INDEXESキーワードのみを指定した場合、またはUPDATE INDEXES句をすべて省略した場合は、既存の索引は次のように変換されます。

- 非同一キー索引は元の形状を保持します。通常の索引は非パーティション・グローバル索引に変換され、非パーティション・グローバル索引は変化せず、パーティション・グローバル索引は変化せずにそのパーティション形状を保持します。
- 同一キー索引はローカル・パーティション索引に変換されます。同一キー索引の索引定義にはパーティション化キーが含まれていますが、索引定義にはパーティション化キーのみが含まれているわけではありません。
- ビットマップ索引は、同一キー索引であるかどうかにかかわらず、ローカル・パーティション索引に変換されます。

パーティション表からパーティション表への変換のデフォルトの索引規則

パーティション表からパーティション表へのデフォルトの索引変換について設定されている規則は、非パーティション表からパーティション表への変換と同じですが、パーティション表の既存のローカル索引に対する追加の処理があります。

- 索引がすでにローカルである場合には、パーティション化ディメンションの両側で索引列が同一キーであれば、索引はローカル索引のままになります。
- パーティション化列がキー列のサブセット(つまり、同一キー)である場合は、グローバル索引がローカルに変換されます。グローバル索引が同一キー索引でない場合は、グローバル索引の形状が保持されます。

非パーティション表のパーティション表への変更の制限事項

modify_to_partitioned句には、次の制限事項が適用されます。

- この句は、索引構成表に対しては指定できません。
- この句は、表でドメイン索引が定義されている場合は指定できません。
- 非パーティション表を参照パーティション表の子表に変更するときには、ONLINEを指定できません。この操作は、オフライン・モードでのみサポートされます。

関連項目:

非パーティション表のパーティション表への変換の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

modify_opaque_type

modify_opaque_type句を使用すると、指定した抽象データ型またはXMLTypeをアンパック記憶域を使用してANYDATA列に格納するようにデータベースに指示できます。

この句には、任意の抽象データ型を指定できます。ただし、従来型の記憶域を使用することではANYDATA列に格納できない、次のデータ型を指定できることが主な利点になります。

- XMLType
- XMLType型、CLOB型、BLOB,型、またはNCLOB型の1つ以上の属性を含む抽象データ型。

アンパック記憶域を使用すると、データ型は、ANYDATA列に関連付けられたシステム生成の非表示列に格納されます。これらのデータ型は、従来型記憶域を使用してANYDATA列に格納されたデータ型と同じように挿入および問合せできます。

anydata_column

ANYDATA型の列の名前を指定します。type_nameが、XMLType型、CLOB型、BLOB型、またはNCLOB型の属性を含まない抽象データ型の場合、anydata_columnは空にする必要があります。

type_name

1つ以上の抽象データ型またはXMLTypeの名前を指定します。この抽象データ型には、XMLType型、CLOB型、BLOB型、またはNCLOB型の属性を含めることができます。この型は、EDITIONABLEにすることができます。これらのデータ型を、その後でanydata_columnに挿入すると、そのデータ型はアンパック記憶域を使用するようになります。この句を以前に同一のanydata_columnに指定していた場合、以前に指定したデータ型と新しく指定したデータ型に、アンパック記憶域が引き続き使用されます。

関連項目:

ANYDATA型の詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』および「ANYDATA列のアンパック記憶域: 例」

immutable_table_clauses

NO DROP句またはNO DELETE句を使用して、不変表の定義を変更できます。

NO DROP句を使用して、不変表または不変表内の行の保存期間を変更します。保存期間を短くすることはできません。

例: 不変表の保存期間の変更

次の文では、不変表imm_tabの定義を変更して、最新の行が50日経過するまでは削除できないように指定します。

```
ALTER TABLE imm_tab NO DROP UNTIL 50 DAYS IDLE;
```

例: 不変表の行の保存期間の変更

次の文では、不変表imm_tabの定義を変更し、作成してから120日経過するまで行を削除できないように指定します。

```
ALTER TABLE imm_tab NO DELETE UNTIL 120 DAYS AFTER
    INSERT;
```

blockchain_table_clauses

ALTER TABLE文でキーワードBLOCKCHAINを使用して作成された表と、1つ以上のblockchain_table_clausesを使用して作成された表を変更できます。

句のセマンティクスについては、CREATE TABLEのblockchain_table_clausesを参照してください。

制限事項

ALTER TABLE文ではblockchain_table_clausesのblockchain_hash_and_data_format_clauseを使用できません。

次の句を除いてブロックチェーン表でALTER TABLEのすべての句を使用できます。

- RENAME table
- ADD COLUMN
- DROP COLUMN
- RENAME COLUMN
- DROP (SUB)PARTITION
- TRUNCATE(SUB)PARTITION
- EXCHANGE(SUB)PARTITION
- MODIFY TYPE

enable_disable_clause

enable_disable_clauseを使用すると、Oracle Databaseが整合性制約を適用するかどうか、およびその方法を指定できます。DROPおよびKEEP句は、一意制約または主キー制約を使用禁止にする場合のみに有効です。

関連項目:

- この文に関するノートおよび制限事項を含むこの句の詳細は、「CREATE TABLE」の「enable_disable_clause」を参照してください。

TABLE LOCK

DDL操作中にロックされた表にのみDDL操作を実行できます。このような表ロックは、DML操作中は必要ありません。

ノート:

一時表に表ロックを適用することはできません。

- ENABLE TABLE LOCKを指定すると、表ロックが有効になり、表に対するDDL操作が実行可能になります。現在実行中のすべてのトランザクションは、表ロックが有効になる前にコミットまたはロールバックする必要があります。

ノート:

Oracle Database は、表をロックする前に、データベース内のアクティブな DML トランザクションが完了するまで待機します。その結果、遅延が生じる可能性があることに注意してください。

- DISABLE TABLE LOCKを指定すると、表ロックが無効になり、表に対するDDL操作が実行できなくなります。

ノート:

ターゲット表の表ロックが使用禁止の場合、パラレル DML 操作は実行されません。

ALL TRIGGERS

ALL TRIGGERS句を使用すると、表に関連するすべてのトリガーを有効または無効にできます。

- ENABLE ALL TRIGGERSを指定すると、表に関連するすべてのトリガーが使用可能になります。トリガー条件が満たされた場合に、トリガーが起動されます。

  1つのトリガーを使用可能にする場合は、ALTER TRIGGERのenable_clauseを使用してください。

  関連項目:

  「CREATE TRIGGER」、「ALTER TRIGGER」および「トリガーの有効化: 例」を参照してください。

- DISABLE ALL TRIGGERSを指定すると、表に関連するすべてのトリガーが使用禁止になります。トリガー条件が満たされた場合でも、使用禁止のトリガーは起動されません。

CONTAINER_MAP

CONTAINER_MAP句を使用すると、コンテナ・マップを使用した表の問合せを有効または無効にできます。

- ENABLE CONTAINER_MAPを指定して、コンテナ・マップを使用した表の問合せを有効化します。
- DISABLE CONTAINER_MAPを指定して、コンテナ・マップを使用した表の問合せを無効化します。

CONTAINERS_DEFAULT

CONTAINERS_DEFAULT句を使用して、CONTAINERS句に対して表を有効化または無効化します。

- ENABLE CONTAINERS_DEFAULTを指定して、CONTAINERS句に対して表を有効化します。
- DISABLE CONTAINERS_DEFAULTを指定して、CONTAINERS句に対して表を無効化します。

例

表への制約の追加: 例

次の文は、データを操作するために新しい表を作成して、新しく作成された表に情報を表示します。

```
CREATE TABLE JOBS_Temp AS SELECT * FROM HR.JOBS;
SELECT * FROM JOBS_Temp WHERE MIN_SALARY < 3000;
JOB_ID     JOB_TITLE                          MIN_SALARY MAX_SALARY
---------- ---------------------------------- ---------- ----------
PU_CLERK   Purchasing Clerk                         2500       5500
ST_CLERK   Stock Clerk                              2008       5000
SH_CLERK   Shipping Clerk                           2500       5500
```

次の文は、列値をより高い値に更新します。

```
UPDATE JOBS_Temp SET MIN_SALARY = 2300 WHERE MIN_SALARY < 2010;
```

次の文は、制約を追加します

```
ALTER TABLE JOBS_Temp ADD CONSTRAINT chk_sal_min CHECK (MIN_SALARY >=2010);
```

次の文は、表の情報を表示します。

```
SELECT * FROM JOBS_Temp WHERE MIN_SALARY < 3000;
JOB_ID     JOB_TITLE                          MIN_SALARY MAX_SALARY
---------- ---------------------------------- ---------- ----------
PU_CLERK   Purchasing Clerk                         2500       5500
ST_CLERK   Stock Clerk                              2300       5000
SH_CLERK   Shipping Clerk                           2500       5500
```

次の文は、制約を表示します。

```
SELECT CONSTRAINT_NAME FROM USER_CONSTRAINTS WHERE TABLE_NAME='JOBS_TEMP';
CONSTRAINT_NAME
--------------------------------------------------------------------------------
SYS_C008830
CHK_SAL_MIN
```

コレクションの取出し: 例

次の文は、サンプル表sh.print_mediaのネストした表の列ad_textdocs_ntabを変更し、問合せ時にロケータのかわりに実値を戻します。

```
ALTER TABLE print_media MODIFY NESTED TABLE ad_textdocs_ntab
   RETURN AS VALUE;
```

パラレル処理の指定: 例

次の文は、サンプル表oe.customersに対する問合せのパラレル処理を指定します。

```
ALTER TABLE customers
   PARALLEL;
```

制約状態の変更: 例

次の文は、employees表のemp_manager_fkという名前の整合性制約をENABLE VALIDATE状態にします。

```
ALTER TABLE employees
   ENABLE VALIDATE CONSTRAINT emp_manager_fk
   EXCEPTIONS INTO exceptions;
```

Oracle Databaseが制約を使用可能にするためには、employees表の各行がこの制約を満たしている必要があります。制約に違反する行があれば、制約は使用禁止のままになります。すべての例外は、exceptions表に記述されます。次の文

で、employees表の例外を検出することもできます。

```
SELECT e.*
   FROM employees e, exceptions ex
   WHERE e.rowid = ex.row_id
      AND ex.table_name = 'EMPLOYEES'
      AND ex.constraint = 'EMP_MANAGER_FK';
```

次の文は、employees表の2つの制約をENABLE NOVALIDATE状態にします。

```
ALTER TABLE employees
   ENABLE NOVALIDATE PRIMARY KEY
   ENABLE NOVALIDATE CONSTRAINT emp_last_name_nn;
```

この文には、次の2つのENABLE句が含まれています。

- 1番目のENABLE句は、表の主キー制約をENABLE NOVALIDATE状態にします。
- 2番目のENABLE句は、emp_last_name_nnという制約をENABLE NOVALIDATE状態にします。

この例では、表のそれぞれの行が2つの制約を満たす場合にかぎり、その制約が使用可能になります。どちらかの制約に違反する行があった場合、エラーが戻され、どちらの制約も使用禁止のままになります。

departments表のlocation_id列の外部キー制約について考えます。ここでは、locations表の主キーを参照しています。次の文は、locations表の主キーを使用禁止にします。

```
ALTER TABLE locations
   MODIFY PRIMARY KEY DISABLE CASCADE;
```

locations表の一意キーは、departments表の外部キーによって参照されるため、この主キーを使用禁止にする場合は、CASCADE句を指定します。この句によって、外部キーも使用禁止になります。

索引構成表の例外表の作成: 例

次の例では、索引構成表hr.countriesの行のうち、主キー制約に違反する行を格納するexcept_table表を作成します。

```
EXECUTE DBMS_IOT.BUILD_EXCEPTIONS_TABLE ('hr', 'countries', 'except_table');

ALTER TABLE countries
   ENABLE PRIMARY KEY
   EXCEPTIONS INTO except_table;
```

例外表を指定する場合は、この表に行を挿入する権限が必要です。検出された例外を調べる場合、例外表を問い合せる権限が必要です。

関連項目:

表に行を挿入するために必要な権限の詳細は、「INSERT」および「SELECT」を参照してください。

CHECK制約の無効化: 例

次の文は、employees表でCHECK制約を定義して無効にします。

```
ALTER TABLE employees ADD CONSTRAINT check_comp
   CHECK (salary + (commission_pct*salary) <= 5000)
   DISABLE;
```

check_comp制約は、給与総額が$5000を超える従業員がいないことを保証します。ただし、この制約が使用禁止になって

いるため、従業員の給与をこの制限以上に増やすことができます。

トリガーの有効化: 例

次の文は、employees表に関連付けられているすべてのトリガーを有効にします。

```
ALTER TABLE employees
   ENABLE ALL TRIGGERS;
```

未使用領域の割当て解除: 例

次の文は、employees表で再利用できるように最高水位標がMINEXTENTSを超えるすべての未使用領域を解放します。

```
ALTER TABLE employees
    DEALLOCATE UNUSED;
```

詳細な大/小文字の区別に関する列の照合の変更: 例

この例では、大/小文字を区別しないように列を変更する方法を示しています。まず、次のようにstudents表を作成し、データを移入します。

```
CREATE TABLE students (last_name VARCHAR2(20), id NUMBER);
INSERT INTO students VALUES('Dodd', 364);
INSERT INTO students VALUES('de Niro', 132);
INSERT INTO students VALUES('Vogel', 837);
INSERT INTO students VALUES('van der Kamp', 549);
INSERT INTO students VALUES('van Der Meer', 624);
```

次の文は、列last_nameをアルファベット順に戻します。結果では大/小文字が区別されることに注意してください。小文字は大文字より後に示されています。

```
SELECT last_name, id
  FROM students
  ORDER BY last_name;
LAST_NAME                      ID
-------------------- ----------
Dodd                          364
Vogel                         837
de Niro                       132
van Der Meer                  624
van der Kamp                  549
```

次の文は、列last_nameのデータ・バインドされた照合を、大/小文字が区別されない照合BINARY_CIに変更します。

```
ALTER TABLE students
  MODIFY (last_name COLLATE BINARY_CI);
```

次の文も、列last_nameをアルファベット順に戻します。今回は、結果で大/小文字が区別されないことに注意してください。

```
SELECT last_name, id
  FROM students
  ORDER BY last_name;
LAST_NAME                      ID
-------------------- ----------
de Niro                       132
Dodd                          364
van der Kamp                  549
van Der Meer                  624
Vogel                         837
```

列名の変更: 例

次の例では、サンプル表oe.customersの列名をcredit_limitからcredit_amountに変更します。

```
ALTER TABLE customers
   RENAME COLUMN credit_limit TO credit_amount;
```

列の削除: 例

次の文は、CASCADE CONSTRAINTSが指定されているdrop_column_clauseです。表t1が次のように作成されているとします。

```
CREATE TABLE t1 (
   pk NUMBER PRIMARY KEY,
   fk NUMBER,
   c1 NUMBER,
   c2 NUMBER,
   CONSTRAINT ri FOREIGN KEY (fk) REFERENCES t1,
   CONSTRAINT ck1 CHECK (pk > 0 and c1 > 0),
   CONSTRAINT ck2 CHECK (c2 > 0)
);
```

次の文に対してエラーが戻されます。

```
/* The next two statements return errors:
ALTER TABLE t1 DROP (pk);  -- pk is a parent key
ALTER TABLE t1 DROP (c1);  -- c1 is referenced by multicolumn
                           -- constraint ck1
```

次の文を発行すると、列pk、主キー制約、外部キー制約riおよびCHECK制約ck1が削除されます。

```
ALTER TABLE t1 DROP (pk) CASCADE CONSTRAINTS;
```

削除される列に対して定義されている制約で参照されている列も、すべて削除される場合は、CASCADE CONSTRAINTSは必要ありません。たとえば、列pkを他の表から参照する参照制約が他に存在しない場合は、次の文を指定するときにCASCADE CONSTRAINTS句がなくてもエラーにはなりません。

```
ALTER TABLE t1 DROP (pk, fk, c1);
```

未使用の列の削除: 例

次の文は、データを操作するために新しい表を作成して、新しく作成された表に情報を表示します。

```
CREATE TABLE JOBS_Temp AS SELECT * FROM HR.JOBS;
SELECT * FROM JOBS_Temp WHERE MAX_SALARY > 20000;
JOB_ID     JOB_TITLE                          MIN_SALARY MAX_SALARY
---------- ---------------------------------- ---------- ----------
AD_PRES    President                               20080      40000
AD_VP      Administration Vice President           15000      30000
SA_MAN     Sales Manager                           10000      20080
```

次の文は、2つの新しい列を追加します。

```
ALTER TABLE JOBS_Temp ADD (DUMMY1 NUMBER(2), DUMMY2 NUMBER(2));
```

次の文は、新しく追加された列に値を挿入します。

```
INSERT INTO JOBS_Temp(JOB_ID, JOB_TITLE, DUMMY1, DUMMY2) VALUES ('D','DUMMY',10,20);
INSERT INTO JOBS_Temp(JOB_ID, JOB_TITLE, DUMMY1, DUMMY2) VALUES ('D','DUMMY',10,20)
```

次の文は、新しく追加された列を未使用に設定します。

```
ALTER TABLE JOBS_TEMP SET UNUSED (DUMMY1, DUMMY2);
```

次の文は、未使用の列数を表示します。

```
SELECT * FROM USER_UNUSED_COL_TABS WHERE TABLE_NAME='JOBS_TEMP';
TABLE_NAM      COUNT
--------- ----------
JOBS_TEMP          2
```

次の文は、未使用の列を削除します。

```
ALTER TABLE JOBS_TEMP DROP UNUSED COLUMNS;
```

次の文は、表の情報を表示します。

```
SELECT * FROM JOBS_TEMP;
JOB_ID     JOB_TITLE                           MIN_SALARY MAX_SALARY
---------- ---------------------------------- ---------- ----------
AD_PRES    President                               20080      40000
AD_VP      Administration Vice President           15000      30000
AD_ASST    Administration Assistant                 3000       6000
FI_MGR     Finance Manager                          8200      16000
FI_ACCOUNT Accountant                               4200       9000
AC_MGR     Accounting Manager                       8200      16000
AC_ACCOUNT Public Accountant                        4200       9000
SA_MAN     Sales Manager                           10000      20080
SA_REP     Sales Representative                     6000      12008
PU_MAN     Purchasing Manager                       8000      15000
PU_CLERK   Purchasing Clerk                         2500       5500
ST_MAN     Stock Manager                            5500       8500
ST_CLERK   Stock Clerk                              2008       5000
SH_CLERK   Shipping Clerk                           2500       5500
IT_PROG    Programmer                               4000      10000
MK_MAN     Marketing Manager                        9000      15000
MK_REP     Marketing Representative                 4000       9000
HR_REP     Human Resources Representative           4000       9000
PR_REP     Public Relations Representative          4500      10500
D          DUMMY
D          DUMMY
```

索引構成表の変更: 例

次の文は、hr.countriesに基づく索引構成表countries_demoの索引セグメントのINITRANSパラメータを変更します。

```
ALTER TABLE countries_demo INITRANS 4;
```

次の文は、オーバーフロー・データ・セグメントを索引構成表countriesに追加します。

```
ALTER TABLE countries_demo ADD OVERFLOW;
```

次の文は、索引構成表countriesのオーバーフロー・データ・セグメントのINITRANSパラメータを変更します。

```
ALTER TABLE countries_demo OVERFLOW INITRANS 4;
```

表パーティションの分割: 例

次の文は、サンプル表sh.salesの古いパーティションsales_q4_2000を分割して2つの新しいパーティションを作成し、1つにはsales_q4_2000bという名前を付け、もう1つには旧パーティションの名前を再利用します。

```
ALTER TABLE sales SPLIT PARTITION SALES_Q4_2000
   AT (TO_DATE('15-NOV-2000','DD-MON-YYYY'))
   INTO (PARTITION SALES_Q4_2000, PARTITION SALES_Q4_2000b);
```

次の文では、元のパーティションsales_q1_2002を、新しい3つのパーティションsales_jan_2002、sales_feb_2002、およびsales_mar_2002に分割します。

```
ALTER TABLE sales SPLIT PARTITION SALES_Q1_2002 INTO (
 PARTITION SALES_JAN_2002 VALUES LESS THAN (TO_DATE('01-FEB-2002','DD-MON-YYYY')),
 PARTITION SALES_FEB_2002 VALUES LESS THAN (TO_DATE('01-MAR-2002','DD-MON-YYYY')),
 PARTITION SALES_MAR_2002);
```

次の文は、pm.print_media表のパーティション・バージョンを作成します。print_media表のLONG列はLOBに変換されています。表は、「Oracle Managed Filesの作成：例」で作成された表領域に格納されます。ad_textdocs_ntab列およびad_header列の基礎となるオブジェクト型は、pmサンプル・スキーマを作成するスクリプトで作成されます。

```
CREATE TABLE print_media_part (
    product_id NUMBER(6),
    ad_id               NUMBER(6),
    ad_composite        BLOB,
    ad_sourcetext       CLOB,
    ad_finaltext        CLOB,
    ad_fltextn          NCLOB,
    ad_textdocs_ntab    TEXTDOC_TAB,
    ad_photo            BLOB,
    ad_graphic          BFILE,
    ad_header           ADHEADER_TYP)
  NESTED TABLE ad_textdocs_ntab STORE AS textdoc_nt
  PARTITION BY RANGE (product_id)
    (PARTITION p1 VALUES LESS THAN (100),
     PARTITION p2 VALUES LESS THAN (200));
```

次の文は、表のパーティションp2をパーティションp2aとp2bに分割します。

```
ALTER TABLE print_media_part
   SPLIT PARTITION p2 AT (150) INTO
   (PARTITION p2a TABLESPACE omf_ts1
      LOB (ad_photo, ad_composite) STORE AS (TABLESPACE omf_ts2),
   PARTITION p2b
      LOB (ad_photo, ad_composite) STORE AS (TABLESPACE omf_ts2))
   NESTED TABLE ad_textdocs_ntab INTO (PARTITION nt_p2a, PARTITION nt_p2b);
```

p2aとp2bのパーティションでは、列ad_photoとad_compositeに対するLOBセグメントが表領域omf_ts2内に作成されます。パーティションp2aのその他の列に対するLOBセグメントは、表領域omf_ts1に保存されます。パーティションp2bのその他の列に対するLOBセグメントは、このALTER文の実行前の表領域で保持されます。ただし、LOBデータおよびLOB索引セグメントが新しい表領域に移動されない場合でも、これらの新しいセグメントが作成されます。

また、ネストした表の列ad_textdocs_ntabに対する新しいセグメントも作成されます。これらの新しいセグメントの記憶表は、nt_p2aおよびnt_p2bです。

2つの表パーティションのマージ：例

次の文は、「表パーティションの分割：例」で作成されたパーティションをマージして1つのパーティションに戻します。

```
ALTER TABLE sales
   MERGE PARTITIONS sales_q4_2000, sales_q4_2000b
   INTO PARTITION sales_q4_2000;
```

次の文は、「表のパーティションの分割：例」の例と逆の操作を実行します。

```
ALTER TABLE print_media_part
   MERGE PARTITIONS p2a, p2b INTO PARTITION p2ab TABLESPACE example
   NESTED TABLE ad_textdocs_ntab STORE AS nt_p2ab;
```

隣接する4つのレンジ・パーティションのマージ例

次の文では、隣接する4つのレンジ・パーティションsales_q1_2000、sales_q2_2000、sales_q3_2000、およびsales_q4_2000を、1つのパーティションsales_all_2000にマージします。

```
ALTER TABLE sales
  MERGE PARTITIONS sales_q1_2000 TO sales_q4_2000
  INTO PARTITION sales_all_2000;
```

LOBおよびネストした表の記憶域を持つ表パーティションの追加: 例

次の文は、パーティションp3をprint_media_part表に追加し(前述の例を参照)、その表のBLOB、CLOBおよびネストした表の列の記憶特性を指定します。

```
ALTER TABLE print_media_part ADD PARTITION p3 VALUES LESS THAN (400)
  LOB(ad_photo, ad_composite) STORE AS (TABLESPACE omf_ts1)
  LOB(ad_sourcetext, ad_finaltext) STORE AS (TABLESPACE omf_ts2)
  NESTED TABLE ad_textdocs_ntab STORE AS nt_p3;
```

パーティションp3の列ad_photoおよびad_compositeに対するLOBデータとLOB索引セグメントは、表領域omf_ts1に格納されます。LOB列の他の属性は、まず表レベルのデフォルトから継承され、次に表領域のデフォルトから継承されます。

列ad_source_textおよびad_finaltextのLOBデータ・セグメントは、omf_ts2表領域に格納され、他のすべての属性は、まず表レベルのデフォルト値から継承され、次に表領域のデフォルト値から継承されます。

実表のパーティションp3に対応する、ネストした表の記憶列ad_textdocs_ntabの記憶表のパーティションの名前はnt_p3になり、他のすべての属性は、まず表レベルのデフォルト値から継承され、次に表領域のデフォルト値から継承されます。

表への複数パーティションの追加: 例

次の文は、「表パーティションの分割: 例」で作成された表print_media_partに3つのパーティションを追加します。

```
ALTER TABLE print_media_part ADD
  PARTITION p3 values less than (300),
  PARTITION p4 values less than (400),
  PARTITION p5 values less than (500);
```

デフォルト・リスト・パーティションの使用: 例

次の文は、「リスト・パーティション化の例」で作成されたリスト・パーティション表を使用します。最初の文は、既存のデフォルト・パーティションを新規のsouthパーティションとデフォルト・パーティションに分割します。

```
ALTER TABLE list_customers SPLIT PARTITION rest
   VALUES ('MEXICO', 'COLOMBIA')
   INTO (PARTITION south, PARTITION rest);
```

次の文は、結果のデフォルト・パーティションをasiaパーティションとマージします。

```
ALTER TABLE list_customers
   MERGE PARTITIONS asia, rest INTO PARTITION rest;
```

次の文は、デフォルト・パーティションを分割して、asiaパーティションを再作成します。

```
ALTER TABLE list_customers SPLIT PARTITION rest
   VALUES ('CHINA', 'THAILAND')
   INTO (PARTITION asia, PARTITION rest);
```

表パーティションの削除: 例

次の文は、パーティションp3 (「LOBおよびネストした表の記憶域を持つ表パーティションの追加: 例」で作成)を削除します。

```
ALTER TABLE print_media_part DROP PARTITION p3;
```

表パーティションの交換: 例

この例では、「リスト・パーティション化の例」で作成したlist_customers表のパーティションと同じ構造を持つ

exchange_table表を作成します。次に、表list_customersのパーティションrestを表exchange_tableに置き換えますが、ローカル索引パーティションと対応するexchange_tableの索引との交換は行わず、exchange_table内のデータがパーティションrestの範囲内にあるかどうかの検証も行いません。

```
CREATE TABLE exchange_table (
   customer_id     NUMBER(6),
   cust_first_name VARCHAR2(20),
   cust_last_name  VARCHAR2(20),
   cust_address    CUST_ADDRESS_TYP,
   nls_territory   VARCHAR2(30),
   cust_email      VARCHAR2(40));
ALTER TABLE list_customers
   EXCHANGE PARTITION rest WITH TABLE exchange_table
   WITHOUT VALIDATION;
```

表パーティションの変更: 例

次の文は、list_customers表のパーティションasiaに対応するすべてのローカル索引パーティションに、UNUSABLEのマークを付けます。

```
ALTER TABLE list_customers MODIFY PARTITION asia
   UNUSABLE LOCAL INDEXES;
```

次の文は、UNUSABLEのマークが付けられたすべてのローカル索引パーティションを再構築します。

```
ALTER TABLE list_customers MODIFY PARTITION asia
   REBUILD UNUSABLE LOCAL INDEXES;
```

表パーティションの移動: 例

次の文は、パーティションp2b (「表パーティションの分割: 例」で作成)を表領域omf_ts1に移動します。

```
ALTER TABLE print_media_part
   MOVE PARTITION p2b TABLESPACE omf_ts1;
```

表パーティション名の変更: 例

次の文は、sh.sales表のパーティション名を変更します。

```
ALTER TABLE sales RENAME PARTITION sales_q4_2003 TO sales_currentq;
```

表パーティションの切捨て: 例

次の文は、「LOB列のあるパーティション表の例」で作成したprint_media_demo表を使用します。p1パーティション内のデータがすべて削除され、解放された領域の割当てが解除されます。

```
ALTER TABLE print_media_demo
   TRUNCATE PARTITION p1 DROP STORAGE;
```

グローバル索引の更新: 例

次の文は、サンプル表sh.salesのパーティションsales_q1_2000を分割し、定義されているグローバル索引を更新します。

```
ALTER TABLE sales SPLIT PARTITION sales_q1_2000
   AT (TO_DATE('16-FEB-2000','DD-MON-YYYY'))
   INTO (PARTITION q1a_2000, PARTITION q1b_2000)
   UPDATE GLOBAL INDEXES;
```

パーティション索引の更新: 例

次の文は、サンプル表sh.costsのパーティションcosts_Q4_2003を分割し、定義されているローカル索引を更新します。使用される表領域は、「基本的な表領域の作成: 例」で作成されたものです。

```
CREATE INDEX cost_ix ON costs(channel_id) LOCAL;
ALTER TABLE costs
  SPLIT PARTITION costs_q4_2003 at
    (TO_DATE('01-Nov-2003','dd-mon-yyyy'))
    INTO (PARTITION c_p1, PARTITION c_p2)
  UPDATE INDEXES (cost_ix (PARTITION c_p1 tablespace tbs_02,
                           PARTITION c_p2 tablespace tbs_03));
```

オブジェクト識別子の指定: 例

次の文は、オブジェクト型、主キーに基づくオブジェクト識別子に対応するオブジェクト表、およびユーザー定義REF列を持つ表を作成します。

```
CREATE TYPE emp_t AS OBJECT (empno NUMBER, address CHAR(30));
CREATE TABLE emp OF emp_t (
   empno PRIMARY KEY)
   OBJECT IDENTIFIER IS PRIMARY KEY;
CREATE TABLE dept (dno NUMBER, mgr_ref REF emp_t SCOPE is emp);
```

次の文は、emp表を参照する制約およびユーザー定義REF列を追加します。

```
ALTER TABLE dept ADD CONSTRAINT mgr_cons FOREIGN KEY (mgr_ref)
   REFERENCES emp;
ALTER TABLE dept ADD sr_mgr REF emp_t REFERENCES emp;
```

表の列の追加: 例

次の文は、NUMBERデータ型の列duty_pct、サイズが3のVARCHAR2データ型の列visa_neededおよびCHECK整合性制約をcountries表に追加します。

```
ALTER TABLE countries
   ADD (duty_pct     NUMBER(2,2)  CHECK (duty_pct < 10.5),
        visa_needed  VARCHAR2(3));
```

仮想表の列の追加: 例

次の文は、hr.employees表のコピーにincomeという列(給与と歩合の合計)を追加します。給与と歩合はどちらもNUMBER型の列であるため、文の中でデータ型が指定されていなくても、仮想列はNUMBER型の列として作成されます。

```
CREATE TABLE emp2 AS SELECT * FROM employees;
ALTER TABLE emp2 ADD (income AS (salary + (salary*commission_pct)));
```

表の列の変更: 例

次の文は、duty_pct列のサイズを増大させます。

```
ALTER TABLE countries
   MODIFY (duty_pct NUMBER(3,2));
```

MODIFY句には列の定義が1つのため、定義を囲むカッコは任意指定です。

次の文は、employees表のPCTFREEパラメータとPCTUSEDパラメータの値を、それぞれ30と60に変更します。

```
ALTER TABLE employees
   PCTFREE 30
   PCTUSED 60;
```

表の記憶域属性の変更

次の文は、既存のJOBS表を使用して、JOBS_TEMPという名前の表を作成します。

```
CREATE TABLE JOBS_TEMP AS SELECT * FROM HR.JOBS;
```

次の文は、USER_TABLES表で記憶域パラメータを問い合せます。

```
SELECT initial_extent,
       next_extent,
       min_extents,
       max_extents,
       pct_increase,
       blocks,
       sample_size
FROM   user_tables
WHERE  table_name = 'JOBS_TEMP';
INITIAL_EXTENT NEXT_EXTENT MIN_EXTENTS MAX_EXTENTS PCT_INCREASE     BLOCKS
SAMPLE_SIZE
-------------- ----------- ----------- ----------- ------------ ---------- ----------
-
         65536     1048576           1  2147483645                       1
19
```

次の文は、新しい記憶域パラメータを使用してJOBS_TEMP表を変更します。

```
ALTER TABLE JOBS_TEMP MOVE
      STORAGE ( INITIAL 20K
                NEXT 40K
                MINEXTENTS 2
                MAXEXTENTS 20
                PCTINCREASE 0 )
      TABLESPACE USERS;
```

次の文は、USER_TABLES表で新しい記憶域パラメータを問い合せます。

```
SELECT initial_extent,
       next_extent,
       min_extents,
       max_extents,
       pct_increase,
       blocks,
       sample_size
FROM   user_tables
WHERE  table_name = 'JOBS_TEMP';
INITIAL_EXTENT NEXT_EXTENT MIN_EXTENTS MAX_EXTENTS PCT_INCREASE     BLOCKS
SAMPLE_SIZE
-------------- ----------- ----------- ----------- ------------ ---------- ----------
-
         65536       40960           1  2147483645                       1
19
```

表の列の追加、変更、名前変更および削除: 例

次の文は、データを操作するために新しい表を作成して、新しく作成された表に情報を表示します。

```
CREATE TABLE JOBS_Temp AS SELECT * FROM HR.JOBS;
SELECT * FROM JOBS_Temp WHERE MAX_SALARY > 30000;
JOB_ID     JOB_TITLE                           MIN_SALARY MAX_SALARY
---------- ----------------------------------- ---------- ----------
AD_PRES    President                                20080      40000
```

次の文は、既存の列定義を変更します。

```
ALTER TABLE JOBS_Temp MODIFY(JOB_TITLE VARCHAR2(100));
```

次の文は、表に2つの新しい列を追加します。

```
ALTER TABLE JOBS_Temp ADD (BONUS NUMBER (7,2), COMM NUMBER (5,2), DUMMY NUMBER(2));
```

次の文は、新しく追加された列を表示します。

```
SELECT JOB_ID, BONUS, COMM, DUMMY FROM JOBS_Temp WHERE MAX_SALARY > 20000;
JOB_ID          BONUS       COMM      DUMMY
---------- ---------- ---------- ----------
AD_PRES
AD_VP
SA_MAN
```

次の文は、既存の列の名前を変更し、変更された列を表示します。

```
ALTER TABLE JOBS_Temp RENAME COLUMN COMM TO COMMISSION;
SELECT JOB_ID, COMMISSION FROM JOBS_Temp WHERE MAX_SALARY > 20000;
JOB_ID     COMMISSION
---------- ----------
AD_PRES
AD_VP
SA_MAN
```

次の文は、表から単一の列を削除します。

```
ALTER TABLE JOBS_Temp DROP COLUMN DUMMY;
```

次の文は、表から複数の列を削除します。

```
ALTER TABLE JOBS_Temp DROP (BONUS, COMMISSION);
```

データの暗号化: 例

次の文は、暗号化アルゴリズムAES256を使用して、hr.employees表のsalary列を暗号化します。前述の「セマンティクス」で説明したように、まず、透過的データ暗号化を有効にする必要があります。

```
ALTER TABLE employees
   MODIFY (salary ENCRYPT USING 'AES256' 'NOMAC');
```

次の文は、デフォルトの暗号化アルゴリズムAES192を使用して、暗号化された新しい列online_acct_pwをoe.customers表に追加します。NO SALTを指定すると、Bツリー索引をこの列に対して作成できるようになります。

```
ALTER TABLE customers
   ADD (online_acct_pw VARCHAR2(8) ENCRYPT 'NOMAC' NO SALT);
```

次の例は、customer.online_acct_pw列を復号化します。

```
ALTER TABLE customers
   MODIFY (online_acct_pw DECRYPT);
```

エクステントの割当て: 例

次の文は、employees表に5KBのエクステントを割り当て、そのエクステントをインスタンス4が使用できるようにします。

```
ALTER TABLE employees
  ALLOCATE EXTENT (SIZE 5K INSTANCE 4);
```

この文には、DATAFILEパラメータが指定されていないため、エクステントは表が入っている表領域に属するデータファイルの1つに割り当てられます。

デフォルト列値の指定: 例

次の文は、product_information表のmin_price列のデフォルト値を10に変更します。

```
ALTER TABLE product_information
  MODIFY (min_price DEFAULT 10);
```

続いてmin_price列に値を指定せずに、product_information表に新しい行を追加する場合、min_price列の値は自動的に10(ゼロ)になります。

```
INSERT INTO product_information (product_id, product_name,
   list_price)
   VALUES (300, 'left-handed mouse', 40.50);
SELECT product_id, product_name, list_price, min_price
    FROM product_information
    WHERE product_id = 300;
PRODUCT_ID PRODUCT_NAME         LIST_PRICE  MIN_PRICE
---------- -------------------- ---------- ----------
       300 left-handed mouse          40.5         10
```

以前に指定したデフォルト値を中止して、新しく追加する行にその値が自動的に挿入されないようにする場合、次の文に示すように、デフォルト値をNULLに置き換えます。

```
ALTER TABLE product_information
   MODIFY (min_price DEFAULT NULL);
```

MODIFY句で指定する必要があるのは、列名と、定義のうち変更される部分のみです(列の定義をすべて指定する必要はありません)。この文は、既存の行の既存の値には影響しません。

次の例では、DEFAULT ON NULLで定義した列を表に追加します。DEFAULTの列値には、順序疑似列NEXTVALが含まれます。

次に示すように、順序s1と表t1を作成します。

```
CREATE SEQUENCE s1 START WITH 1;
CREATE TABLE t1 (name VARCHAR2(10));
INSERT INTO t1 VALUES('Kevin');
INSERT INTO t1 VALUES('Julia');
INSERT INTO t1 VALUES('Ryan');
```

列idを追加します。この列のデフォルトは、s1.NEXTVALに設定されます。idのデフォルトの列値は、表内に既存の各行に割り当てられます。s1.NEXTVALが各行に割り当てられる順序は、非決定的になります。

```
ALTER TABLE t1 ADD (id NUMBER DEFAULT ON NULL s1.NEXTVAL NOT NULL);
SELECT id, name FROM t1 ORDER BY id;
        ID NAME
---------- ----------
         1 Kevin
         2 Julia
         3 Ryan
```

それ以降に新しい行を表に追加して、id列にNULL値を指定すると、DEFAULT ON NULL式のs1.NEXTVALが挿入されます。

```
INSERT INTO t1(id, name) VALUES(NULL, 'Sean');
SELECT id, name FROM t1 ORDER BY id;
        ID NAME
---------- ----------
         1 Kevin
         2 Julia
         3 Ryan
         4 Sean
```

XMLType表への制約の追加: 例

次の例では、xwarehouses表(「XMLTypeの例」で作成)に主キー制約を追加します。

```
ALTER TABLE xwarehouses
   ADD (PRIMARY KEY(XMLDATA."WarehouseID"));
```

この疑似列の詳細は、「XMLDATA疑似列」を参照してください。

制約名の変更: 例

次の文は、サンプル表oe.customersの制約名をcust_fname_nnからcust_firstname_nnに変更します。

```
ALTER TABLE customers RENAME CONSTRAINT cust_fname_nn
   TO cust_firstname_nn;
```

制約の削除: 例

次の文は、departments表の主キーを削除します。

```
ALTER TABLE departments
    DROP PRIMARY KEY CASCADE;
```

PRIMARY KEY制約の名前がpk_deptであることがわかっている場合は、次のように指定しても削除できます。

```
ALTER TABLE departments
    DROP CONSTRAINT pk_dept CASCADE;
```

CASCADE句によって、主キーを参照するすべての外部キーが削除されます。

次の文は、employees表のemail列の一意キーを削除します。

```
ALTER TABLE employees
    DROP UNIQUE (email);
```

この文のDROP句ではCASCADE句を省略します。CASCADEオプションを省略することによって、一意キーを参照する外部キーがある場合、その一意キーは削除されません。

LOB列: 例

次の文は、CLOB列であるresumeをemployee表に追加し、この新しい列のLOB記憶特性を指定します。

```
ALTER TABLE employees ADD (resume CLOB)
  LOB (resume) STORE AS resume_seg (TABLESPACE example);
```

次の文は、キャッシュを使用できるようにLOB列のresumeを変更します。

```
ALTER TABLE employees MODIFY LOB (resume) (CACHE);
```

次の文は、SecureFiles CLOB column resume to the employee表に追加し、この新しい列のLOB記憶特性を指定します。SecureFiles LOBを格納する表領域に対しては、自動セグメント領域管理が行われている必要があります。したがって、この例のLOBデータは、auto_seg_ts表領域(「表領域に対するセグメント領域管理の指定: 例」で作成)に格納されます。

```
ALTER TABLE employees ADD (resume CLOB)
LOB (resume) STORE AS SECUREFILE resume_seg (TABLESPACE auto_seg_ts);
```

次の文は、LOB列のresumeをキャッシュを使用しないように変更します。

```
ALTER TABLE employees MODIFY LOB (resume) (NOCACHE);
```

ネストした表: 例

次の文は、ネストした表の列skillsをemployee表に追加します。

```
ALTER TABLE employees ADD (skills skill_table_type)
    NESTED TABLE skills STORE AS nested_skill_table;
```

また、ネストした表の記憶特性も変更できます。変更する場合、nested_table_col_propertiesに指定した記憶表の名前を使用してください。記憶表では、問合せまたはDML文を実行することはできません。記憶表は、ネストした表の列の記憶特性を変更するためにのみ使用します。

次の文は、ネストした表の列clientと記憶表client_tabを使用して、表vet_serviceを作成します。ネストした表client_tabを変更して制約を指定します。

```
CREATE TYPE pet_t AS OBJECT
   (pet_id NUMBER, pet_name VARCHAR2(10), pet_dob DATE);
/
CREATE TYPE pet AS TABLE OF pet_t;
/
CREATE TABLE vet_service (vet_name VARCHAR2(30),
                          client   pet)
  NESTED TABLE client STORE AS client_tab;
ALTER TABLE client_tab ADD UNIQUE (pet_id);
```

次の文は、REF値のネストした表用の記憶表を変更して、REFの範囲が限定されることを指定します。

```
CREATE TYPE emp_t AS OBJECT (eno number, ename char(31));
CREATE TYPE emps_t AS TABLE OF REF emp_t;
CREATE TABLE emptab OF emp_t;
CREATE TABLE dept (dno NUMBER, employees emps_t)
   NESTED TABLE employees STORE AS deptemps;
ALTER TABLE deptemps ADD (SCOPE FOR (COLUMN_VALUE) IS emptab);
```

同様に、次の文は、REFをROWIDとともに格納することを指定します。

```
ALTER TABLE deptemps ADD (REF(column_value) WITH ROWID);
```

これらのALTER TABLE文を正確に実行するためには、記憶表deptempsが空である必要があります。また、ネストした表は、スカラー値(REF値)の表として定義されるため、Oracle Databaseは、暗黙的に列名COLUMN_VALUEを記憶表に設定します。

関連項目:

- ネストした表の記憶域の詳細は、「CREATE TABLE」を参照してください。
- ネストした表の詳細は、『Oracle Databaseオブジェクト・リレーショナル開発者ガイド』を参照してください。

REF列: 例

次の文は、オブジェクト型dept_tを作成し、表staffを作成します。

```
CREATE TYPE dept_t AS OBJECT
   (deptno NUMBER, dname VARCHAR2(20));
/
CREATE TABLE staff
   (name   VARCHAR2(100),
    salary NUMBER,
    dept   REF dept_t);
```

オブジェクト表officesを次のように作成します。

```
CREATE TABLE offices OF dept_t;
```

dept列は、任意の表に格納されたdept_tのオブジェクトに参照を格納できます。次のようにdept列に有効範囲制約を追加することによって、departments表に格納されたオブジェクトのみが参照されるように制限できます。

```
ALTER TABLE staff
    ADD (SCOPE FOR (dept) IS offices);
```

前述のALTER TABLE文は、staff表が空である場合のみ正常に実行されます。

次の文は、staffのdept列にREF値を格納する際、ROWIDも同時に格納します。

```
ALTER TABLE staff
   ADD (REF(dept) WITH ROWID);
```

ANYDATA列のアンパック記憶域: 例

この例では、ANYDATAの列を含む表を作成し、アンパック記憶域を使用してOPAQUEデータ型をANYDATAの列に格納してから、そのデータ型を問い合せます。この例では、データベースにユーザーhrとして接続していると仮定しています。

表t1を作成します。この表は、NUMBERの列nと、ANYDATAの列xを格納します。

```
CREATE TABLE t1 (n NUMBER, x ANYDATA);
```

CLOB属性を含む、オブジェクト・タイプclob_typを作成します。

```
CREATE OR REPLACE TYPE clob_typ AS OBJECT (c clob);
/
```

表t1のANYDATAの列xに含まれるOPAQUEデータ型のXMLTypeとclob_typのアンパック記憶域を有効にします。

```
ALTER TABLE t1 MODIFY OPAQUE TYPE x STORE (XMLType, clob_typ) UNPACKED;
```

XMLTypeオブジェクトとclob_typオブジェクトを、表t1に挿入します。これらの型は、アンパック記憶域を使用するようになります。

```
INSERT INTO t1
  VALUES(1, anydata.convertobject(XMLType('<Test>This is test XML</Test>')));
INSERT INTO t1
  VALUES(2, anydata.convertobject(clob_typ(TO_CLOB('This is a test CLOB'))));
```

表t1を問い合せて、ANYDATAの列xに格納されたタイプの名前を確認します。

```
SELECT t1.*, anydata.getTypeName(t1.x) typename FROM t1;
    N X()                  TYPENAME
----- -------------------- --------------------
    1 ANYDATA()            SYS.XMLTYPE
    2 ANYDATA()            HR.CLOB_TYP
```

XMLTypeデータ型とclob_typデータ型に格納された値を問い合せることができる、ファンクションを作成します。

```
CREATE FUNCTION get_xmltype (ad IN ANYDATA) RETURN VARCHAR2 AS
      rtn_val PLS_INTEGER;
      my_xmltype XMLType;
      string_val VARCHAR2(30);
   BEGIN
      rtn_val := ad.getObject(my_xmltype);
      string_val := my_xmltype.getstringval();
      return (string_val);
   END;
```

```
/
CREATE FUNCTION get_clob_typ (ad IN ANYDATA) RETURN VARCHAR2 AS
      rtn_val PLS_INTEGER;
      my_clob_typ clob_typ;
      string_val VARCHAR2(30);
   BEGIN
      rtn_val := ad.getObject(my_clob_typ);
      string_val := (my_clob_typ.c);
      return (string_val);
   END;
/
```

表t1を問い合せて、ANYDATAの列xの各データ型に格納された値を確認します。

```
SELECT t1.*, anydata.getTypeName(t1.x) typename,
  CASE
    WHEN anydata.gettypename(t1.x) = 'SYS.XMLTYPE' THEN get_xmltype(t1.x)
    WHEN anydata.gettypename(t1.x) = 'HR.CLOB_TYP' THEN get_clob_typ(t1.x)
  END string_value
FROM t1;
    N X()                  TYPENAME             STRING_VALUE
----- -------------------- -------------------- ------------------------------
    1 ANYDATA()            SYS.XMLTYPE          <Test>This is test XML</Test>
    2 ANYDATA()            HR.CLOB_TYP          This is a test CLOB
```

その他の例

ALTER TABLE文を使用して整合性制約を定義する例は、「constraint」を参照してください。

表の記憶域パラメータの変更例は、「storage_clause」を参照してください。

ALTER TABLESPACE

目的

ALTER TABLESPACE文を使用すると、既存の表領域、1つ以上のデータファイルまたは一時ファイルを変更できます。

この句では、ディクショナリ管理表領域をローカル管理表領域に変換することはできません。変換するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』に記載されているDBMS_SPACE_ADMINパッケージを使用してください。

関連項目:

表領域作成の詳細は、『Oracle Database管理者ガイド』および「CREATE TABLESPACE」を参照してください。

前提条件

SYSAUX表領域を変更する場合は、SYSDBAシステム権限が必要です。

ALTER TABLESPACEシステム権限を持っている場合、すべてのALTER TABLESPACE操作を実行できます。MANAGE TABLESPACEシステム権限を持っている場合は、次の操作のみを実行できます。

- 表領域をオンラインまたはオフラインにする。
- バックアップを開始または終了する。
- 表領域を読取り専用または読み書き両用にする。
- 表領域の状態をPERMANENTまたはTEMPORARYに変更する。
- 表領域のデフォルト・ロギング・モードをLOGGINGまたはNOLOGGINGに設定する。
- 表領域で強制ロギング・モードを有効または無効にする。
- 表領域または表領域データ・ファイルの名前を変更する。
- UNDO表領域に対するRETENTION GUARANTEEまたはRETENTION NOGUARANTEEを指定する。
- 表領域のデータ・ファイルのサイズを変更する。
- 表領域内のデータ・ファイルに対する自動拡張を使用可能または使用禁止にする。
- 一時表領域または一時ファイルによって使用される領域の大きさを縮小する。

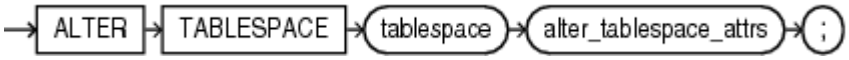
表領域を読取り専用にする場合、次の条件が満たされている必要があります。

- 表領域は必ずオンラインにする。
- 表領域にアクティブなロールバック・セグメントがない。SYSTEM表領域にはSYSTEMロールバック・セグメントがあるため、読取り専用にはできません。また、読取り専用表領域のロールバック・セグメントにはアクセスできないため、ロールバック・セグメントを削除してから、表領域を読取り専用にすることをお薦めします。
- 表領域がオープン・バックアップに使用されていない。バックアップの終わりに表領域内のすべてのデータファイルのヘッダー・ファイルが更新されるためです。

これらの条件を満たすために、制限モードでこの機能を実行すると有効です。制限モードでは、RESTRICTED SESSIONシステム権限を持つユーザーのみがログインできます。

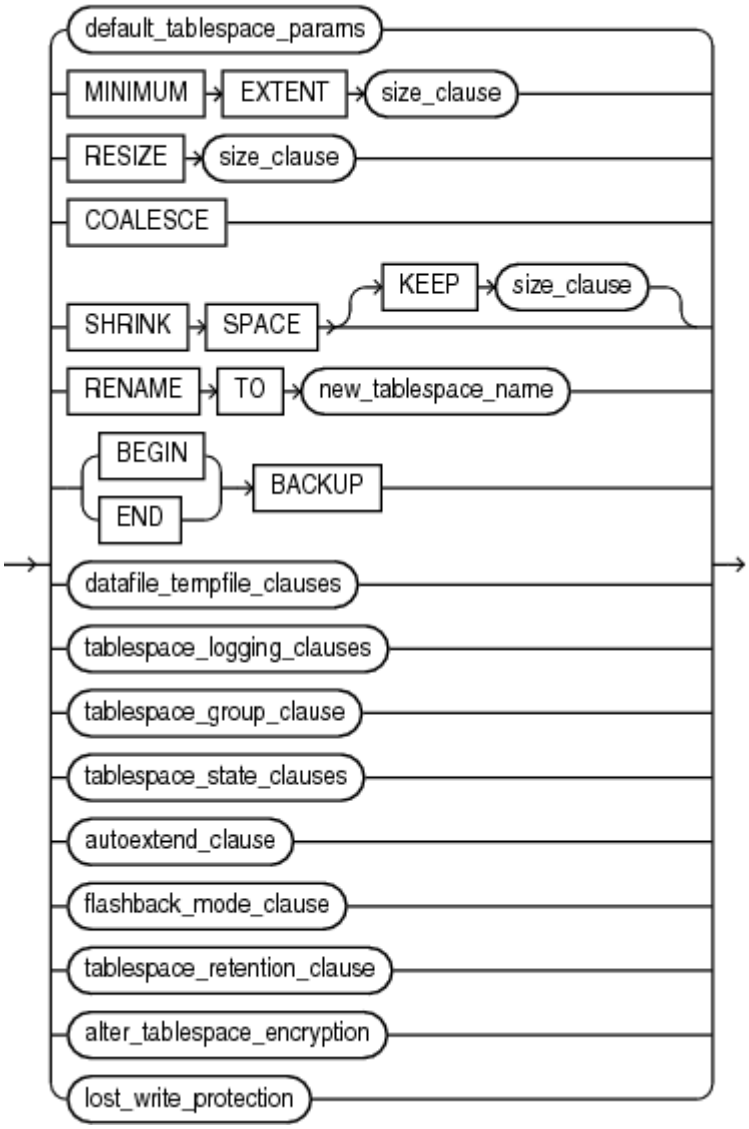
構文

alter_tablespace::=



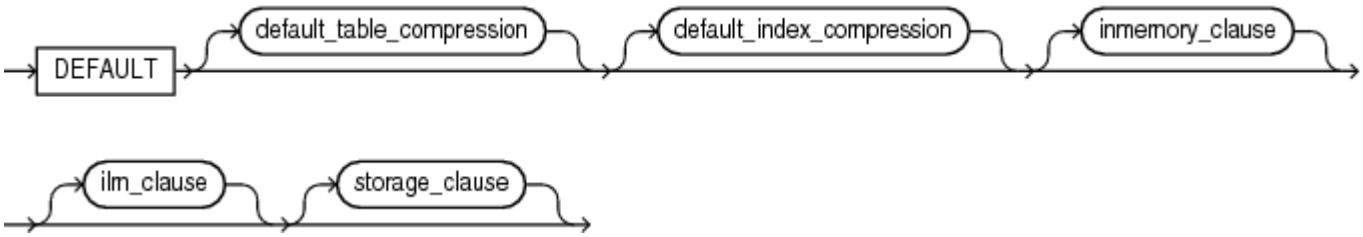
(alter_tablespace_attrs::=)

alter_tablespace_attrs::=



(default_tablespace_params::=、size_clause::=、datafile_tempfile_clauses::=、tablespace_logging_clauses::=、tablespace_group_clause::=、tablespace_state_clauses::=、autoextend_clause::=、flashback_mode_clause::=、tablespace_retention_clause::=、alter_tablespace_encryption::=、lost_write_protection::=)

default_tablespace_params::=

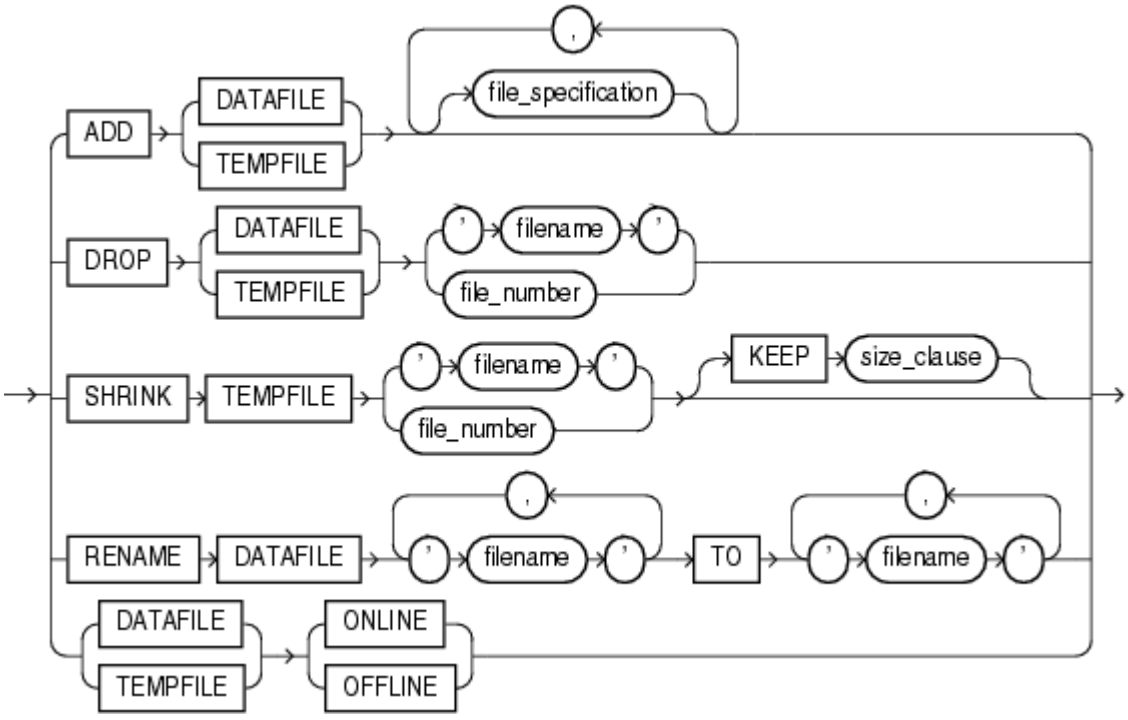


(default_table_compression::= (CREATE TABLESPACEの一部)、default_index_compression::= (CREATE TABLESPACEの一部)、inmemory_clause::= (CREATE TABLESPACEの一部)、ilm_clause::= (ALTER TABLEの一部)、storage_clause::=)

ノート:

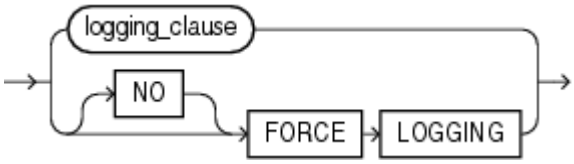
DEFAULT 句を指定する場合、default_table_compression、default_index_compression、inmemory_clause、ilm_clause または storage_clause の各句のうち、1 つ以上を指定する必要があります。

datafile_tempfile_clauses::=



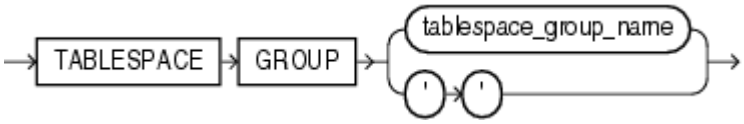
(file_specification::=).

tablespace_logging_clauses::=

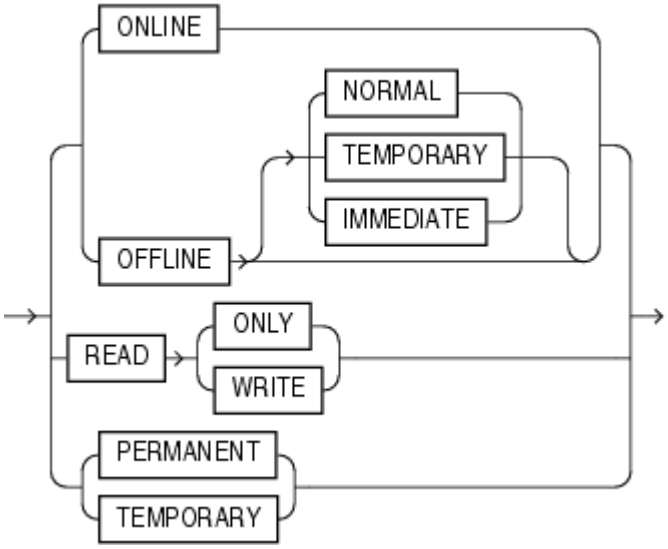


(logging_clause::=)

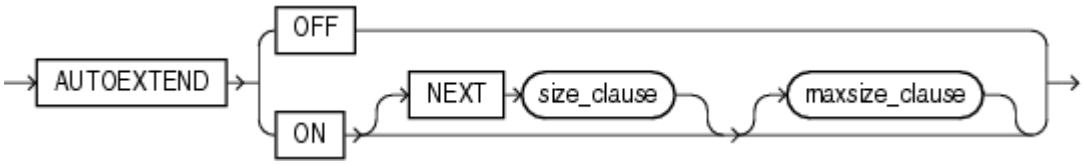
tablespace_group_clause::=



tablespace_state_clauses::=

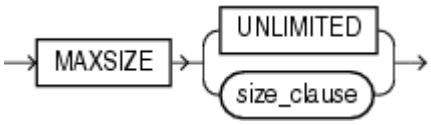


autoextend_clause::=



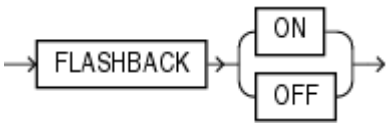
(size_clause::=)

maxsize_clause::=

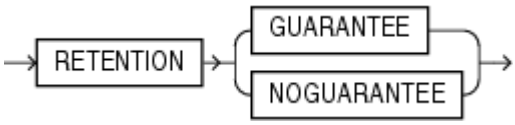


(size_clause::=)

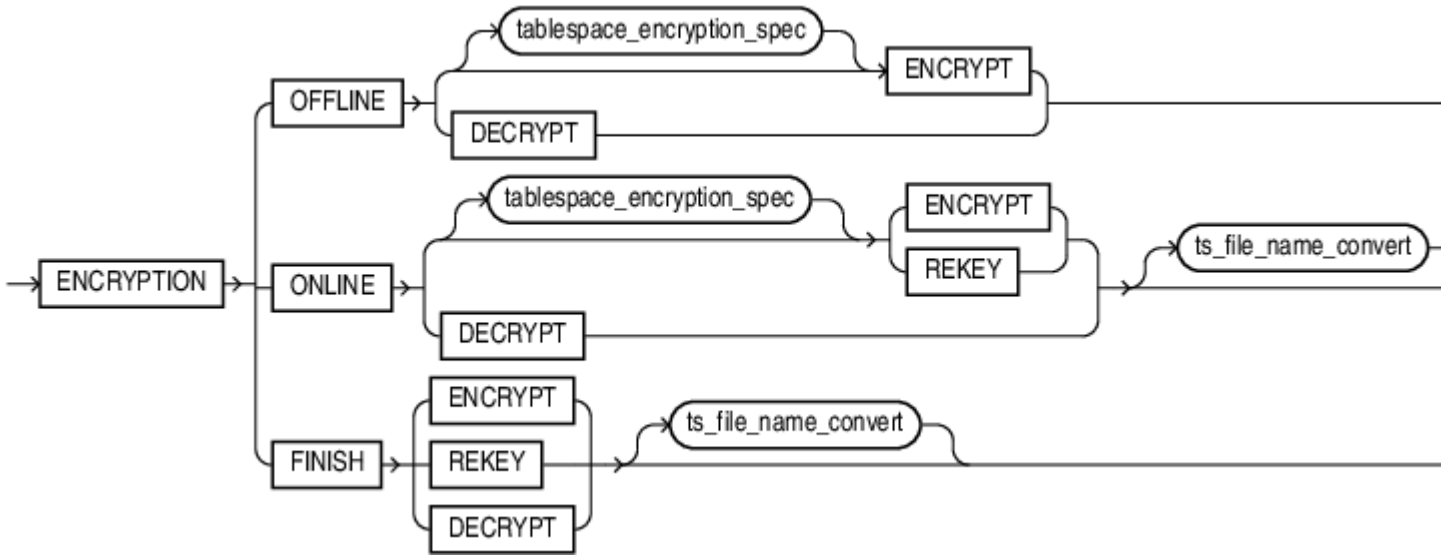
flashback_mode_clause::=



tablespace_retention_clause::=

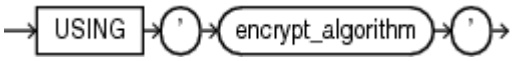


alter_tablespace_encryption::=

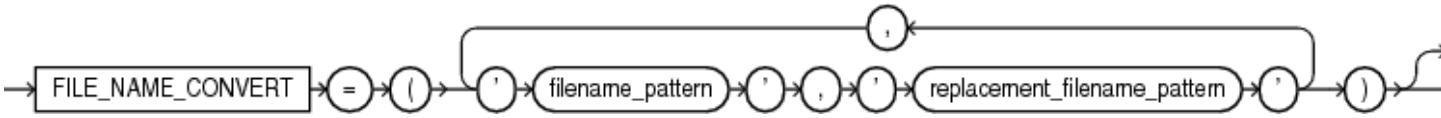


(tablespace_encryption_spec::=、ts_file_name_convert::=)

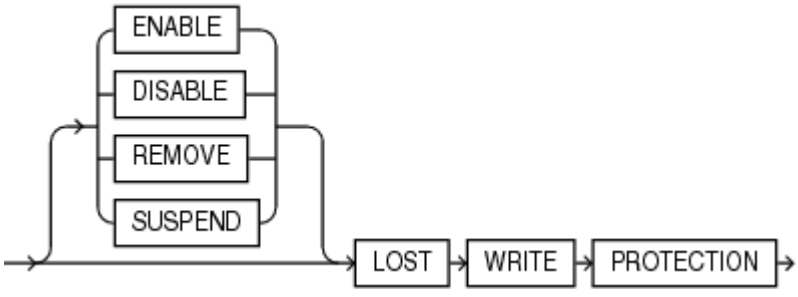
tablespace_encryption_spec::=



ts_file_name_convert::=



lost_write_protection::=



セマンティクス

tablespace

変更する表領域の名前を指定します。

表領域の変更の制限事項

表領域の変更には、次の制限事項があります。

- tablespaceがUNDO表領域の場合、この文ではADD DATAFILE、RENAME DATAFILE、RENAME TO(表領域の名前の変更)、DATAFILE ... ONLINE、DATAFILE ... OFFLINE、BEGIN BACKUPおよびEND BACKUPのみが指定可能です。
- SYSTEM表領域を、読取り専用または一時表領域にしたり、オフラインにすることはできません。
- ローカル管理の一時表領域に対してこの文で指定できるのは、ADD句のみです。

関連項目:

自動UNDO管理およびUNDO表領域の詳細は、『Oracle Database管理者ガイド』を参照してください。

alter_tablespace_attrs

alter_tablespace_attrs句を使用すると、表領域の属性を変更できます。

default_tablespace_params

この句を使用すると、表領域の新しいデフォルト・パラメータを指定できます。新しいデフォルト・パラメータは、後で表領域に作成されるオブジェクトに適用されます。

default_table_compression、default_index_compression、inmemory_clause、ilm_clauseおよびstorage_clause句のセマンティクスは、CREATE TABLESPACEおよびALTER TABLESPACEと同じです。これらの句の詳細は、CREATE TABLESPACEのドキュメントの「default_tablespace_params」句を参照してください。

MINIMUM EXTENT

この句は、永続的なディクショナリ管理表領域に対してのみ有効です。MINIMUM EXTENT句を指定すると、表領域内のすべての使用済エクステントまたは未使用エクステントの大きさが、size_clauseで指定したサイズ以上であること、およびその倍数であることが保証され、表領域における空き領域の断片化を制御できます。

MINIMUM EXTENTの制限事項

この句は、ローカル管理の表領域またはディクショナリ管理の一時表領域に対して指定できません。

関連項目:

この句の詳細は、「size_clause」 を参照してください。MINIMUM EXTENTを使用した領域の断片化の制御の詳細は、『Oracle Database管理者ガイド』を参照してください

RESIZE句

この句は、消失書込み保護トラッキング・データを格納するシャドウ表領域などのbigfile表領域に対してのみ有効です。1つのデータファイルのサイズを指定のサイズまで拡張または縮小できます。K、M、GまたはTを使用して、それぞれKB、MB、GBまたはTB単位で指定することもできます。

smallfile表領域に新しく追加されたデータファイルまたは一時ファイルのサイズを変更するには、ALTER DATABASE ... autoextend_clauseを使用します(「database_file_clauses」を参照)。

関連項目:

bigfile表領域については、「BIGFILE | SMALLFILE」を参照してください。

COALESCE

この句を使用すると、表領域内の各データファイルで、連続する未使用エクステントをすべて結合し、連続するより大きいエクステントを作成します。

SHRINK SPACE句

この句が有効であるのは、一時表領域に対してのみです。表領域に使用される領域の大きさを縮小できます。任意指定のKEEP句では、size_clauseで表領域の縮小後サイズの下限を定義します。これは、自動拡張可能な表領域のMAXSIZEとは逆の機能です。KEEP句を省略した場合は、表領域は他の記憶域属性が満たされることを条件として可能なかぎり縮小されます。

RENAME句

この句を指定すると、tablespaceの名前を変更できます。この句は、tablespaceおよびそのすべてのデータファイルがオンラインで、COMPATIBLEパラメータが10.0.0以上に設定されている場合にのみ有効です。名前の変更は、永続表領域および一時表領域の両方に対して実行できます。

tablespaceが読取り専用の場合、データファイルのヘッダーは更新されず、新しい名前は反映されません。アラート・ログに、データファイルのヘッダーが更新されなかったことが記録されます。

ノート:

バックアップからリストアしたデータファイルを使用して制御ファイルを再作成する場合、データファイルのヘッダーに古い表領域の名前が反映されていると、再作成された制御ファイルにも古い表領域の名前が反映されます。ただし、データベースが完全にリカバリされた後は、制御ファイルに新しい名前が反映されます。

tablespaceが、Oracle Real Application Clusters(Oracle RAC)環境のインスタンスに対するUNDO表領域として指定されており、データベースの起動にサーバー・パラメータ・ファイルが使用されている場合、そのサーバー・パラメータ・ファイル(SPFILE)で、インスタンスに対するUNDO_TABLESPACEパラメータの値は、新しい表領域の名前を反映するように変更されます。単一インスタンス・データベースで、spfileのかわりにパラメータ・ファイル(pfile)が使用されている場合、データベース管理者にpfile内の値を手動で変更することを推奨するメッセージがアラート・ログに書き込まれます。

ノート:

RENAME 句は、実行中のインスタンスの UNDO_TABLESPACE パラメータの値を変更しません。これは UNDO 表領域の機能に影響しませんが、次の文を発行してインスタンスの実行中に UNDO_TABLESPACE の値を新しい表領域名に手動で変更することをお薦めします。

```
ALTER SYSTEM SET UNDO_TABLESPACE = new_tablespace_name SCOPE = MEMORY;
```

この文は一度だけ発行する必要があります。UNDO_TABLESPACE パラメータが pfile または spfile の新しい表領域名に設定されている場合、次にインスタンスを再起動するときにパラメータが正しく設定されます。

表領域名の変更の制限事項

SYSTEMおよびSYSAUXの各表領域の名前は変更できません。

BACKUP句

この句を使用すると、表領域のすべてのデータファイルをオンライン(ホット)・バックアップ・モードにしたり、このモードから戻すことができます。

関連項目:

- メディア・リカバリなしでデータベースを再起動する場合の詳細は、『Oracle Database管理者ガイド』を参照してください。
- データベース内のすべてのデータファイルに対するオンライン・バックアップ・モードの設定または解除の詳細は、「ALTER DATABASE」の「BACKUP句」を参照してください。
- 個々のデータファイルをオンライン・バックアップ・モードから解除する場合の詳細は、「ALTER DATABASE」の「alter_datafile_clause」を参照してください。

BEGIN BACKUP

BEGIN BACKUPを指定すると、表領域を構成するデータファイルのオープン・バックアップを実行することを示すことができます。この句を指定することによって、ユーザーがこの表領域にアクセスできなくなることはありません。オープン・バックアップを開始する前に、この句を指定してください。

表領域のバックアップの開始の制限事項

表領域のバックアップの開始には、次の制限事項があります。

- この句は、読取り専用の表領域またはローカル管理の一時表領域に対して指定できません。
- バックアップ中は、表領域のオフラインへの正常な切替え、インスタンスの停止または表領域の別のバックアップ処理の開始は実行できません。

関連項目:

表領域のバックアップ: 例

END BACKUP

END BACKUPを指定すると、表領域のオンライン・バックアップが完了したことを示すことができます。オンライン・バックアップの完了後、できるだけ早くこの句を指定してください。インスタンスに障害またはSHUTDOWN ABORTが発生した場合、次のインスタンス起動時にメディア・リカバリ(必要に応じて、アーカイブREDOログも)が必要であるとみなされます。

表領域のバックアップの終了の制限事項

読取り専用表領域ではこの句を使用できません。

datafile_tempfile_clauses

datafile_tempfile_clausesを使用すると、データファイルまたは一時ファイルを追加および変更できます。

ADD句

ADDを指定すると、file_specificationによって指定されたデータファイルまたは一時ファイルを表領域に追加できます。オペレーティング・システムのファイル・システム内の標準データファイルと一時ファイル、またはOracle Automatic Storage Managementディスク・グループのファイルを指定するには、file_specificationのdatafile_tempfile_spec

書式(「file_specification」を参照)を使用します。

この句は、ローカル管理の一時表領域に対して、どんな場合でも指定できる唯一の句です。

`file_specification`を指定しないと、AUTOEXTENDが有効になった100MBのOracle管理ファイルが作成されます。

データファイルまたは一時ファイルを、オンラインのローカル管理表領域、またはオンラインまたはオフラインのディクショナリ管理表領域に追加できます。なお、そのデータファイルが別のデータベースで使用中でないことを確認してください。

データファイルおよび一時ファイルの追加の制限事項

この句は、データファイルまたは一時ファイルを1つのみ含む表領域である、bigfile (単一ファイル)表領域には指定できません。

ノート:

オペレーティング・システムによっては、一時ファイルのブロックが実際にアクセスされるまで、一時ファイル用の領域が割り当てられない場合があります。領域の割当ての遅延のため、一時ファイルの作成およびサイズ変更が速くなります。ただし、後で一時ファイルが使用されるときに、十分なディスク領域を使用可能にする必要があります。問題を回避するには、一時ファイルの作成またはサイズ変更の前に、ディスクの空き領域が、新しく作成する一時ファイルまたはサイズ変更後の一時ファイルのサイズよりも大きいことを確認してください。ディスク領域に余裕を持たせておくと、関連のない操作による、予期されるディスク使用量の増加にも対応できます。その後で、作成またはサイズ変更操作を実行してください。

関連項目:

「file_specification」、「データファイルおよび一時ファイルの追加と削除: 例」および「Oracle Managed Filesのデータファイルの追加: 例」

DROP句

`DROP`を指定すると、`filename`や`file_number`によって指定された空のデータファイルまたは一時ファイルを表領域から削除できます。この句は、データファイルまたは一時ファイルをデータ・ディクショナリから削除し、オペレーティング・システムから削除します。この句を指定するときには、データベースがオープンしている必要があります。

`ALTER TABLESPACE ... DROP TEMPFILE`文は、`ALTER DATABASE TEMPFILE ... DROP INCLUDING DATAFILES`を指定することと同じです。

ファイルの削除の制限事項

データファイルまたは一時ファイルを削除するには、データファイルまたは一時ファイルが次の状態である必要があります。

- 空であること。
- 表領域内で最初に作成されたデータ・ファイルではないこと。この場合、かわりに表領域が削除されます。
- ローカル管理されるディクショナリから移行された読取り専用表領域でないこと。他のすべての読取り専用表領域からのデータファイルの削除はサポートされています。
- オフラインでないこと。

関連項目:

- 一時ファイルの削除の詳細は、「ALTER DATABASE」の「alter_tempfile_clause」を参照してください。
- データファイル番号の詳細と、データファイルの管理ガイドラインの詳細は、『Oracle Database管理者ガイド』を参照してください。
- データファイルおよび一時ファイルの追加と削除：例

SHRINK TEMPFILE句

この句が有効であるのは、一時表領域を変更する場合のみです。指定した一時ファイルに使用される領域の大きさを縮小できます。任意指定のKEEP句では、size_clauseで一時ファイルの縮小後サイズの下限を定義します。これは、自動拡張可能な表領域のMAXSIZEとは逆の機能です。KEEP句を省略した場合は、一時ファイルは他の記憶域属性が満たされることを条件として可能なかぎり縮小されます。

RENAME DATAFILE句

RENAME DATAFILEを指定すると、表領域の1つ以上のデータファイルの名前を変更できます。データベースをオープンしておくこと、および名前の変更前に表領域をオフラインにすることが必要です。それぞれのfilenameには、ご使用のオペレーティング・システムのファイル名の表記規則に従って、データファイル名を完全に指定してください。

この句では、表領域を古いファイルではなく新しいファイルに対応付けます。オペレーティング・システムのファイル名は実際には変更されません。このため、オペレーティング・システム上でこのファイル名を変更する必要があります。

関連項目:

表領域の移動および名前の変更：例

ONLINE | OFFLINE句

これらの句を使用すると、表領域のすべてのデータファイルまたは一時ファイルを、オフラインまたはオンラインにできます。これらの句は、表領域のONLINEまたはOFFLINE状態には影響しません。

データベースはマウントされている必要があります。tablespaceがSYSTEM、UNDO表領域、またはデフォルトの一時表領域の場合、データベースをオープンしないでおく必要があります。

tablespace_logging_clauses

この句を使用すると、表領域のロギング特性を設定または変更できます。

logging_clause

LOGGINGを指定すると、表領域内のすべての表、索引およびパーティションのロギング属性を指定できます。表レベル、索引レベルおよびパーティション・レベルでのロギング指定によって、表領域レベルのロギング属性を上書きできます。

既存の表領域のロギング属性をALTER TABLESPACE文によって変更した場合、この文の実行後に作成されたすべての表、索引およびパーティションに、新しいデフォルトのロギング属性(これは後で上書きもできます)が適用されます。既存のオブジェクトのロギング属性は変更されません。

FORCE LOGGINGモードの表領域がある場合、この文でNOLOGGINGを指定すると、表領域のデフォルト・ロギング・モードをNOLOGGINGに設定できます。ただし、この設定によって表領域のFORCE LOGGINGモードは解除されません。

[NO] FORCE LOGGING

この句を使用すると、表領域で強制ロギング・モードを有効または無効にできます。データベースをオープンし、READ WRITEモードにしておく必要があります。この設定により、表領域のデフォルトLOGGINGモードまたはNOLOGGINGモードは変更されま

せん。

強制ロギング・モードの制限事項

FORCE LOGGINGは、UNDO表領域または一時表領域には指定できません。

関連項目:

FORCE LOGGINGモードをいつ使用するかの詳細は、『Oracle Database管理者ガイド』および「表領域のロギング属性の変更: 例」を参照してください。

tablespace_group_clause

この句は、ローカル管理の一時表領域に対してのみ有効です。この句を使用すると、tablespace_group_name表領域グループに対してtablespaceを追加または削除できます。

- グループ名を指定すると、tablespaceがその表領域グループのメンバーであることを示すことができます。tablespace_group_nameが存在しない場合、表領域を変更して表領域グループのメンバーにすると、その表領域グループが暗黙的に作成されます。
- 空の文字列(' ')を指定すると、tablespace_group_name表領域グループからtablespaceを削除できます。

表領域グループの制限事項

表領域グループは、永続表領域またはディクショナリ管理の一時表領域に対しては指定できません。

関連項目:

表領域グループの詳細は、『Oracle Database管理者ガイド』および「表領域グループの割当て: 例」を参照してください。

tablespace_state_clauses

この句を使用すると、表領域の状態を設定または変更できます。

ONLINE | OFFLINE

ONLINEを指定すると、表領域をオンラインにできます。OFFLINEを指定すると、表領域をオフラインにし、そのセグメントへの後続のアクセスを禁止できます。表領域をオフラインにすると、そのすべてのデータファイルもオフラインになります。

ノート:

表領域を長期間オフラインにする前に、デフォルト表領域または一時表領域としてその表領域が割り当てられているユーザーに対して、表領域の割当てを変更することを検討します。表領域をオフラインにしている間は、これらのユーザーは、その表領域内でオブジェクトに対して領域を割り当てたり、領域をソートすることはできません。ユーザーへの表領域の割当ての詳細は、「ALTER USER」を参照してください。

表領域のオフライン化の制限事項

一時表領域はオフライン化できません。

OFFLINE NORMAL

NORMALを指定すると、システム・グローバル領域(SGA)以外にある表領域のすべてのデータファイルにあるすべてのブロックをフ

ラッシュできます。データファイルをオンラインに戻す前に、表領域のメディア・リカバリを行う必要はありません。これはデフォルトです。

OFFLINE TEMPORARY

TEMPORARYを指定すると、Oracle Databaseは表領域内のすべてのオンライン・データファイルに対してチェックポイントを実行しますが、すべてのファイルに対して書込みを実行できるかどうかは保証しません。この文の発行時にオフラインであるファイルでは、表領域をオンラインに戻す前に、メディア・リカバリが必要な場合があります。

OFFLINE IMMEDIATE

IMMEDIATEを指定した場合は、表領域のファイルが使用可能であることは保証されず、チェックポイントも実行されません。表領域をオンラインに戻す前に、メディア・リカバリを行う必要があります。

ノート:

ALTER TABLESPACE ... OFFLINE に対する FOR RECOVER 設定は、非推奨になっています。この構文は、下位互換性を保つためにのみサポートされています。ただし、表領域のリカバリにはトランスポータブル表領域機能を使用することをお薦めします。

関連項目:

メディア・リカバリを実行するトランスポータブル表領域の使用の詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

READ ONLY | READ WRITE

READ ONLYを指定すると、表領域を読取り専用遷移モードに設定できます。この状態では、既存のトランザクションは完了(コミットまたはロールバック)できますが、表領域内のブロックを変更した既存のトランザクションをロールバックすること以外は、その表領域に対してさらにDML操作を行うことはできません。SYSAUX、SYSTEMまたは一時表領域は、READ ONLYに設定できません。

表領域が読取り専用の場合は、そのファイルを読取り専用メディアにコピーできます。その場合、SQL文のALTER DATABASE ... RENAMEを使用して、新しいファイル位置を示すように制御ファイル内のデータファイルの名前を変更する必要があります。

関連項目:

- 読取り専用の表領域の詳細は、『Oracle Database概要』を参照してください。
- ALTER DATABASE

READ WRITEを指定すると、読取り専用に指定されている表領域に対して書込み操作を実行できるようになります。

PERMANENT | TEMPORARY

PERMANENTを指定すると、一時表領域を永続表領域に変換できます。永続表領域とは、永続的なデータベース・オブジェクトを格納できる場所です。表領域を作成するときのデフォルトです。

TEMPORARYを指定すると、永続表領域を一時表領域に変換できます。一時表領域とは、永続的なデータベース・オブジェク

トを格納できない表領域です。一時表領域の中のオブジェクトはセッション中のみ保持されます。

一時表領域の制限事項

一時表領域には、次の制限事項があります。

- SYSAUX表領域には、TEMPORARYを指定できません。
- tablespaceを標準的なブロック・サイズで作成しなかった場合、永続表領域を一時表領域に変換できません。
- FORCE LOGGINGモードでは、表領域に対してTEMPORARYを指定できません。

autoextend_clause

この句は、bigfile(単一ファイル)表領域に対してのみ有効です。この句を使用すると、表領域内の単一のデータファイルに対して自動拡張を使用可能または使用禁止にできます。smallfile表領域に新しく追加されたデータファイルまたは一時ファイルの自動拡張を使用可能または使用禁止にするには、ALTER DATABASE文でautoextend_clauseを使用します(「database_file_clauses」を参照)。

関連項目:

- bigfile(単一ファイル)表領域については、『Oracle Database管理者ガイド』を参照してください。
- autoextend_clauseの詳細は、「file_specification」を参照してください。

flashback_mode_clause

この句を使用すると、後続のFLASHBACK DATABASE操作でこの表領域を使用するかどうかを指定できます。

- FLASHBACKモードをオンにするには、データベースはマウント済で、クローズ状態である必要があります。
- FLASHBACKモードをオフにするには、データベースはマウント済(READ WRITEのオープン状態またはクローズ状態)である必要があります。

この句は一時表領域では無効です。

この句の詳細は、「CREATE TABLESPACE」を参照してください。

関連項目:

データベースのフラッシュバックの詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

tablespace_retention_clause

この句のセマンティクスは、CREATE TABLESPACE文およびALTER TABLESPACE文で同じです。「CREATE TABLESPACE」の「tablespace_retention_clause」を参照してください。

alter_tablespace_encryption

これらの句を使用すると、表領域を暗号化、復号化またはキー更新できます。

OFFLINE

この句を使用すると、オフライン変換により表領域を暗号化または復号化できます。表領域がオフラインになっているか、データベースがマウントされているが、オープンしていない必要があります。オフライン変換方法では、補助ディスク領域またはファイルは

使用されません。既存のデータファイルが直接操作されます。このため、オフライン変換を実行する前に、表領域の全体バックアップを実行してください。

- 表領域を暗号化するには、ENCRYPTを指定します。AES128、AES192またはAES256のアルゴリズムを使用して表領域を暗号化できます。表領域は暗号化されていない必要があります。
- 表領域を復号化するには、DECRYPTを指定します。表領域は、オフライン変換(OFFLINE ENCRYPT)により事前に暗号化されている必要があります。

オフライン変換操作が中断された場合は、オフライン変換コマンドを再発行して操作を完了できます。

ONLINE

この句を使用すると、オンライン変換により表領域を暗号化、復号化またはキー更新できます。表領域は必ずオンラインにする。オンライン変換方法では、表領域内のデータファイルごとに新しいデータファイルが作成されます。このため、この句を使用する前に、ディスクの空き領域の量が、表領域で現在使用されているディスク領域の量以上であることを確認してください。

- 表領域を暗号化するには、ENCRYPTを指定します。表領域は暗号化されていない必要があります。
- 暗号化されている表領域を異なる暗号化アルゴリズムを使用して暗号化するには、REKEYを指定します。表領域は、作成時に暗号化されているか、オンライン変換(ONLINE ENCRYPT)により暗号化されている必要があります。
- 表領域を復号化するには、DECRYPTを指定します。表領域は、作成時に暗号化されているか、オンライン変換(ONLINE ENCRYPT)により暗号化されている必要があります。

オンライン変換操作が中断された場合は、FINISH句を発行して操作を完了できます。「FINISH」句を参照してください。

tablespace_encryption_spec

この句を使用すると、表領域を暗号化またはキー更新する際に使用する暗号化アルゴリズムを指定できます。この句を省略した場合、データファイルはAES128アルゴリズムを使用して暗号化されます。この句のセマンティクスの詳細は、CREATE TABLESPACEのドキュメントの「tablespace_encryption_spec」を参照してください。

ts_file_name_convert

この句を使用すると、オンライン変換中に作成される新しいデータファイルの名前がデータベースにより生成される方法を指定できます。

FILE_NAME_CONVERTを省略すると、Oracleは補助ファイルの名前を内部的に選択し、後でその名前を元の名前に戻します。

- filename_patternには、既存のデータファイル名にある文字列を指定します。
- replacement_filename_patternには、置換文字列を指定します。新しいデータファイルに名前が付けられるとき、filename_patternがreplacement_filename_patternに置換されます。
- 表領域変換が終了した後に元のファイルを保持する場合は、KEEPを指定します。この句を省略した場合、変換が終了したときに元のファイルが削除されます。

FINISH

この句を使用すると、以前に中断されたオンライン変換操作を完了できます。ENCRYPT、DECRYPT、REKEYおよびts_file_name_convert句のセマンティクスは、ONLINE句と同じです。詳細は、「ONLINE」句を参照してください。

alter_tablespace_encryption句の制限事項

一時表領域に対しては、オフライン変換およびオンライン変換は実行できません。

lost_write_protection

個々の表領域の消失書込み保護を有効化する前に、まずALTER DATABASEを使用してデータベースでシャドウ消失書込み保護を有効化する必要があります。その後、CREATE TABLESPACEコマンドを使用して、そのデータベースに少なくとも1つのシャドウ表領域を作成する必要があります。

これらのステップの後に、ALTER TABLESPACEを使用して、シャドウ表領域で消失書込み保護を有効化、削除および一時停止できます。

例: 表領域に対する消失書込み保護の有効化

次のコマンドでは、tbsu1表領域の消失書込み保護を有効にします。

```
ALTER TABLESPACE tbsu1 ENABLE LOST WRITE PROTECTION
```

例: シャドウ表領域の消失書込み保護の削除

次のコマンドでは、tbsu1表領域の消失書込み保護を削除します。

```
ALTER TABLESPACE tbsu1 REMOVE LOST WRITE PROTECTION
```

例: シャドウ表領域の消失書込み保護の一時停止

次のコマンドでは、tbsu1表領域の消失書込み保護を一時停止します。

```
ALTER TABLESPACE tbsu1 SUSPEND LOST WRITE PROTECTION
```

関連項目:

シャドウ表領域を使用した消失書込み保護の管理

例

表領域のバックアップ: 例

次の文は、バックアップが間もなく開始されることをデータベースに通知します。

```
ALTER TABLESPACE tbs_01
    BEGIN BACKUP;
```

次の文は、バックアップが終了したことをデータベースに通知します。

```
ALTER TABLESPACE tbs_01
   END BACKUP;
```

表領域の移動および名前の変更: 例

次の例は、tbs_02表領域(「表領域の自動拡張の有効化: 例」で作成)に関連付けられたデータファイルを、diskb:tbs_f5.dbfからdiska:tbs_f5.dbfに移動して、名前を変更します。

- OFFLINE句を指定したALTER TABLESPACE文を使用して、この表領域をオフラインにします。

  ```
  ALTER TABLESPACE tbs_02 OFFLINE NORMAL;
  ```

- オペレーティング・システムのコマンドを使用して、このファイルをdiskb:tbs_f5.dbfからdiska:tbs_f5.dbfにコピーします。
- RENAME DATAFILE句を指定したALTER TABLESPACE文を使用して、このデータファイルの名前を変更します。

```
ALTER TABLESPACE tbs_02
  RENAME DATAFILE 'diskb:tbs_f5.dbf'
  TO              'diska:tbs_f5.dbf';
```

- ONLINE句を指定したALTER TABLESPACE文を使用して、この表領域をオンラインに戻します。

```
ALTER TABLESPACE tbs_02 ONLINE;
```

データファイルおよび一時ファイルの追加と削除：例

次の文は、表領域にデータファイルを追加します。さらに多くの領域が必要な場合、10KBの新しいエクステントが最大100KBまで追加されます。

```
ALTER TABLESPACE tbs_03
    ADD DATAFILE 'tbs_f04.dbf'
    SIZE 100K
    AUTOEXTEND ON
    NEXT 10K
    MAXSIZE 100K;
```

次の文は、空のデータファイルを削除します。

```
ALTER TABLESPACE tbs_03
    DROP DATAFILE 'tbs_f04.dbf';
```

次の文は、「一時表領域の作成：例」で作成された一時表領域に一時ファイルを追加し、その後、削除します。

```
ALTER TABLESPACE temp_demo ADD TEMPFILE 'temp05.dbf' SIZE 5 AUTOEXTEND ON;
ALTER TABLESPACE temp_demo DROP TEMPFILE 'temp05.dbf';
```

一時表領域の領域の管理：例

次の文は、「一時表領域の作成：例」で作成した一時表領域の領域を、SHRINK SPACE句を使用して管理します。KEEP句が省略されているので、表領域の大きさは、表領域の他の記憶域属性を満たしていることを条件として可能なかぎり縮小されます。

```
ALTER TABLESPACE temp_demo SHRINK SPACE;
```

Oracle Managed Filesのデータファイルの追加：例

次の例では、Oracle Managed Filesのデータファイルを`omf_ts1`表領域に追加します(この表領域の作成の詳細は、「Oracle Managed Filesの作成：例」を参照)。新しいデータファイルは100MBで、最大サイズが制限なしで自動拡張されます。

```
ALTER TABLESPACE omf_ts1 ADD DATAFILE;
```

表領域のロギング属性の変更：例

次の例では、表領域のデフォルトのロギング属性を`NOLOGGING`に変更します。

```
ALTER TABLESPACE tbs_03 NOLOGGING;
```

表領域のロギング属性を変更した場合でも、その表領域内の既存のスキーマ・オブジェクトのロギング属性には影響しません。表レベル、索引レベルおよびパーティション・レベルでのロギング指定によって、表領域レベルのロギング属性を上書きできます。

UNDOデータの保持の変更：例

次の文は、`undots1`表領域のUNDOデータの保持を、通常のUNDOデータ動作に変更します。

```
ALTER TABLESPACE undots1
  RETENTION NOGUARANTEE;
```

次の文は、undots1表領域のUNDOデータの保持を、期限が切れていないUNDOデータを保持する動作に変更します。

```
ALTER TABLESPACE undots1
  RETENTION GUARANTEE;
```

ALTER TABLESPACE SET

ノート:

この SQL 文は、Oracle Sharding を使用している場合にのみ有効です。Oracle Sharding の詳細は、『Oracle Database 管理者ガイド』を参照してください。

目的

ALTER TABLESPACE SET文を使用して、既存の表領域セットの属性を変更します。属性の変更は、表領域セットのすべての表領域に適用されます。

関連項目:

「CREATE TABLESPACE SET」および「DROP TABLESPACE SET」

前提条件

シャード・カタログ・データベースにSDBユーザーとして接続している必要があります。

ALTER TABLESPACEシステム権限を持っている場合、すべてのALTER TABLESPACE SET操作を実行できます。MANAGE TABLESPACEシステム権限を持っている場合は、次の操作のみを実行できます。

- 表領域セットのすべての表領域をオンラインまたはオフラインにする。
- バックアップを開始または終了する。
- 表領域セットのすべての表領域を読取り専用または読取り/書込みにする。
- 表領域セットのすべての表領域のデフォルト・ロギング・モードをLOGGINGまたはNOLOGGINGに設定する。
- 表領域セットのすべての表領域で、強制ロギング・モードを有効または無効にする。
- 表領域セットのすべてのデータファイルのサイズを変更する。
- 表領域セットのすべてのデータファイルに対する自動拡張を有効または無効にする。

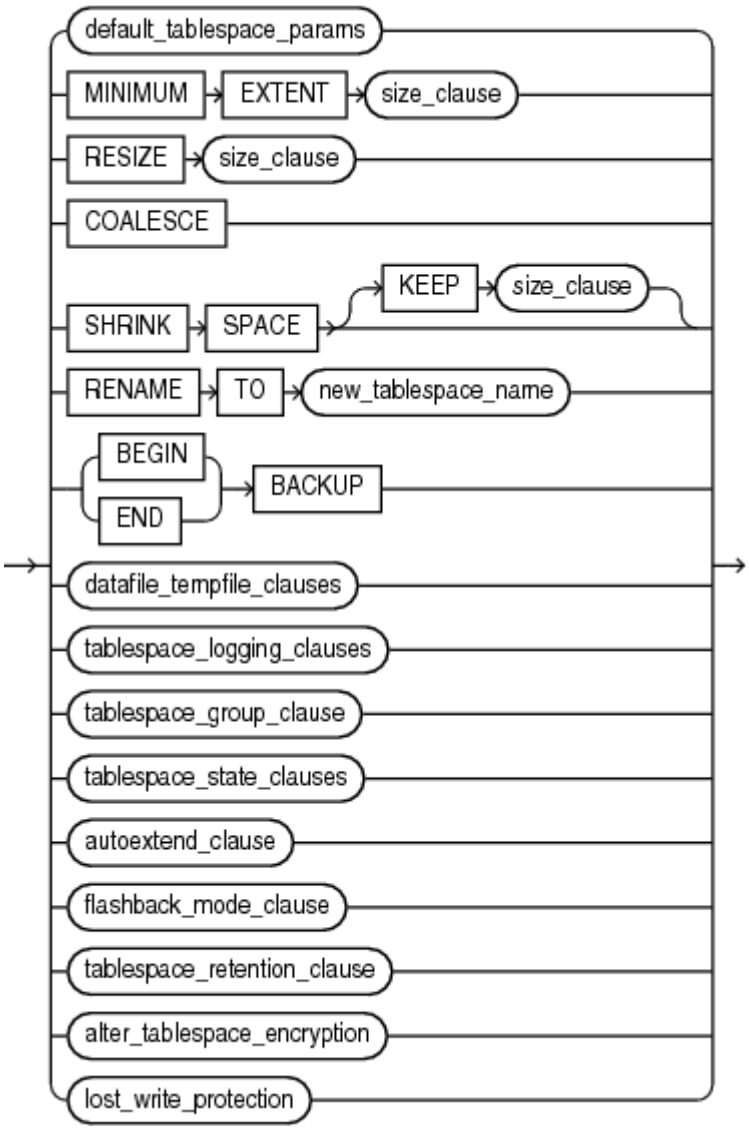
表領域セットを読取り専用にする場合、次の条件が満たされている必要があります。

- 表領域セットの表領域がオンラインである。
- 表領域セットにアクティブなロールバック・セグメントがない。また、読取り専用表領域セットのロールバック・セグメントにはアクセスできないため、ロールバック・セグメントを削除してから、表領域セットを読取り専用にすることをお薦めします。
- 表領域セットがオープン・バックアップに使用されていない。これは、バックアップの終わりに表領域セットのすべてのデータファイルのヘッダー・ファイルが更新されるためです。

構文

alter_tablespace_set::=

alter_tablespace_attrs::=



(ALTER TABLESPACEのdefault_tablespace_params::=、size_clause::=、datafile_tempfile_clauses::=、tablespace_logging_clauses::=、tablespace_state_clauses::=、autoextend_clause::=、alter_tablespace_encryption::=の各句を参照)

セマンティクス

tablespace_set

変更する表領域セットの名前を指定します。

alter_tablespace_attrs

この句を使用すると、表領域セットのすべての表領域の属性を変更できます。

`alter_tablespace_attrs`の副次句のセマンティクスはALTER TABLESPACE文と同じですが、次の例外があります。

- 表領域セットには次の副次句は指定できません。
  - `MINIMUM EXTENT size_clause`
  - `SHRINK SPACE [ KEEP size_clause ]`
  - `tablespace_group_clause`
  - `flashback_mode_clause`
  - `tablespace_retention_clause`

- datafile_tempfile_clausesでは、表領域セットに対して次の副次句のみがサポートされます。
  - RENAME DATAFILE
  - DATAFILE { ONLINE | OFFLINE }
- tablespace_state_clausesでは、表領域セットに対してPERMANENTおよびTEMPORARY副次句はサポートされません。

関連項目:

この句のセマンティクスの詳細は、ALTER TABLESPACEのドキュメントの「alter_tablespace_attrs」を参照してください。

例

表領域セットの変更: 例

次の文は、表領域セットts1のすべての表領域を強制ロギング・モードにします。

```
ALTER TABLESPACE SET ts1
  FORCE LOGGING;
```

ALTER TRIGGER

目的

トリガーはPL/SQLを使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

ALTER TRIGGER文を使用すると、データベース・トリガーを使用可能化、使用禁止化またはコンパイルできます。

ノート:

この文では、既存のトリガーの宣言または定義は変更されません。トリガーを再宣言または再定義する場合は、OR REPLACE キーワードを指定した CREATE TRIGGER 文を使用します。

関連項目:

- トリガーの作成については、「CREATE TRIGGER」を参照してください。
- トリガーの削除については、「DROP TRIGGER」を参照してください。
- トリガーの概要は、『Oracle Database概要』を参照してください。

前提条件

トリガーが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY TRIGGERシステム権限が必要です。

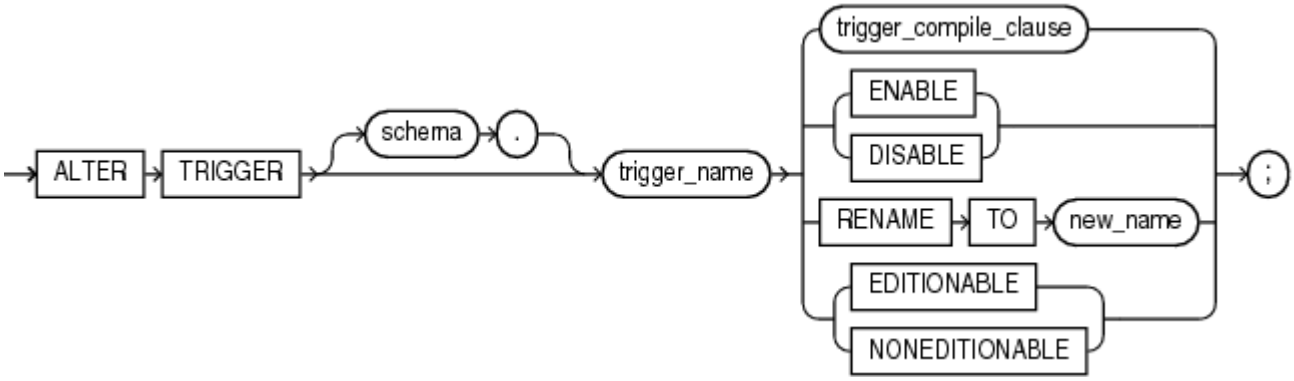
DATABASE上のトリガーを変更する場合は、ADMINISTER DATABASE TRIGGER権限が必要です。

関連項目:

DATABASEトリガーに基づいたトリガーの詳細は、「CREATE TRIGGER」を参照してください。

構文

alter_trigger::=



(trigger_compile_clause: この句の構文の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。)

セマンティクス

schema

トリガーが含まれているスキーマを指定します。schemaを指定しない場合、トリガーは自分のスキーマ内にあるとみなされます。

trigger_name

変更するトリガーの名前を指定します。

trigger_compile_clause

この句の構文とセマンティクスの詳細およびトリガーの作成とコンパイルの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

ENABLE | DISABLE

ENABLEを指定すると、トリガーを使用可能にできます。また、ALTER TABLEのENABLE ALL TRIGGERS句を使用することによって、表に対応付けられたすべてのトリガーを使用可能にできます。「ALTER TABLE」を参照してください。

DISABLEを指定すると、トリガーを使用禁止にできます。また、ALTER TABLEのDISABLE ALL TRIGGERS句を使用することによって、表に対応付けられたすべてのトリガーを使用禁止にできます。

RENAME句

RENAME TO new_nameを指定すると、トリガーの名前を変更できます。トリガーの名前は変更され、名前が変更される前と同じ状態になります。

トリガーの名前を変更すると、USER_SOURCE、ALL_SOURCEおよびDBA_SOURCEデータ・ディクショナリ・ビューに記憶されているトリガーのソースが再構築されます。その結果、トリガー・ソースが変更されていなくても、これらのビューのTEXT列のコメントおよび書式設定が変更される場合があります。

EDITIONABLE | NONEDITIONABLE

この句を使用すると、schemaのスキーマ・オブジェクト・タイプTRIGGERのエディションニングが後で有効化されたときに、そのトリガーをエディション・オブジェクトにするか非エディション・オブジェクトにするかを指定できます。デフォルトは、EDITIONABLEです。エディション・オブジェクトと非エディション・オブジェクトの変更の詳細は、『Oracle Database開発ガイド』を参照してください。

NONEDITIONABLEの制限事項

crossedition トリガーに対してNONEDITIONABLEを指定できません。

ALTER TYPE

目的

オブジェクト型はPL/SQLを使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

ALTER TYPE文を使用すると、メンバー属性またはメソッドを追加または削除できます。オブジェクト型の既存のプロパティ(FINALまたはINSTANTIABLE)、または型のスカラー属性も変更できます。

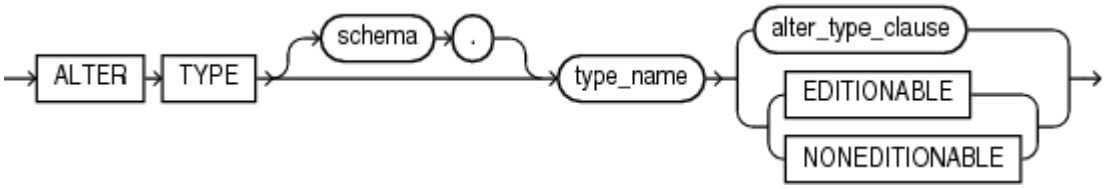
この文を使用すると、新しいオブジェクト・メンバーのサブプログラム仕様を追加することによって、型の仕様部または本体を再コンパイルしたり、オブジェクト型の仕様を変更することができます。

前提条件

オブジェクト型が自分のスキーマ内にあり、CREATE TYPEかCREATE ANY TYPE権限を持っている必要があります。または、ALTER ANY TYPEシステム権限が必要です。

構文

alter_type::=



(alter_type_clause: この句の構文の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。)

セマンティクス

schema

型が含まれているスキーマを指定します。schemaを指定しない場合、この型は現行のスキーマ内にあるとみなされます。

type_name

オブジェクト型、ネストした表型またはVARRAY型の名前を指定します。

type_nameの制限事項

エディション化されたオブジェクト型を進化させることはできません。次のいずれかに該当する場合は、ORA-22348が発生してALTER TYPE文は失敗します。

- typeがエディション化されたオブジェクト型であり、ALTER TYPE文にtype_compile_clauseがない。ALTER TYPE文は、エディション化されたオブジェクト型の再コンパイルには使用できますが、他の目的には使用できません。
- typeに依存するエディション化されたオブジェクト型が存在し、ALTER TYPE文にCASCADE句がある。

type_compile_clauseおよびCASCADE句の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

alter_type_clause

この句の構文とセマンティクスの詳細およびオブジェクト型の作成とコンパイルの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

EDITIONABLE | NONEDITIONABLE

この句を使用すると、schemaのスキーマ・オブジェクト・タイプTYPEのエディションニングが後で有効化されたときに、その型をエディション・オブジェクトにするか非エディション・オブジェクトにするかを指定できます。デフォルトは、EDITIONABLEです。エディション・オブジェクトと非エディション・オブジェクトの変更の詳細は、『Oracle Database開発ガイド』を参照してください。

ALTER USER

目的

ALTER USERを使用すると、次の操作を実行できます。

- データベース・ユーザーの認証またはデータベース・リソースの特性を変更します。
- プロキシ・サーバーが認証なしでクライアントとして接続することを許可します。
- Oracle Automatic Storage Management (Oracle ASM)クラスタで、ユーザーのパスワードを現在のノードのOracle ASMインスタンスに対してローカルなパスワード・ファイルで変更します。

関連項目:

ユーザーの認証方式の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

前提条件

通常、ALTER USERシステム権限が必要です。ただし、現行ユーザーはこの権限がない場合でも自分のパスワードは変更できます。

SYSパスワードを変更するには、パスワード・ファイルが存在する必要があり、SYSパスワードを変更できるようにするには、alter user権限を付与されたアカウントにSYSDBA管理ロールが必要です。

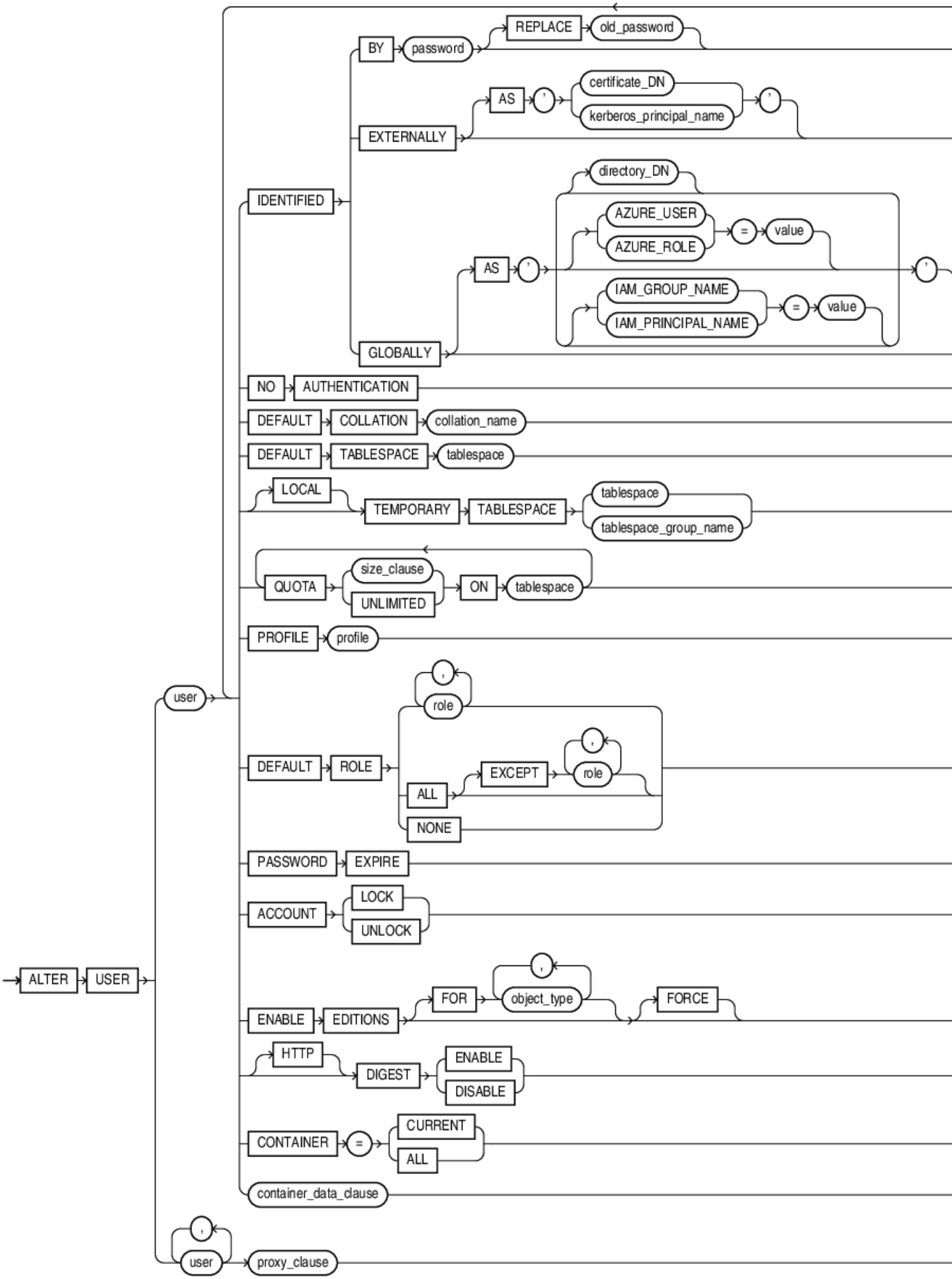
Oracle ASMインスタンス・パスワード・ファイルで自分以外のユーザーのパスワードを変更するには、AS SYSASMとして認証されている必要があります。

CONTAINER句を指定する場合は、マルチテナント・コンテナ・データベース(CDB)に接続している必要があります。現在のコンテナがルートである場合は、CONTAINER = ALLまたはCONTAINER = CURRENTを指定できます。現在のコンテナがプラガブル・データベース(PDB)である場合は、CONTAINER = CURRENTのみ指定可能です。

container_data_clauseを使用してCONTAINER_DATA属性を設定および変更する場合は、CDBに接続している必要があります。また、現在のコンテナがルートである必要があります。

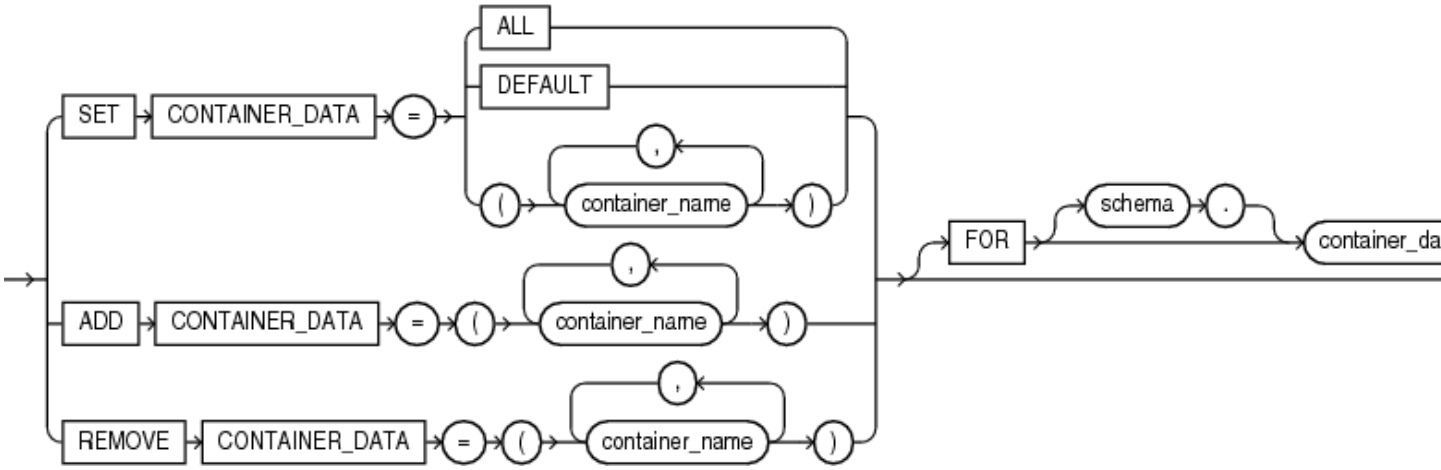
構文

alter_user::=

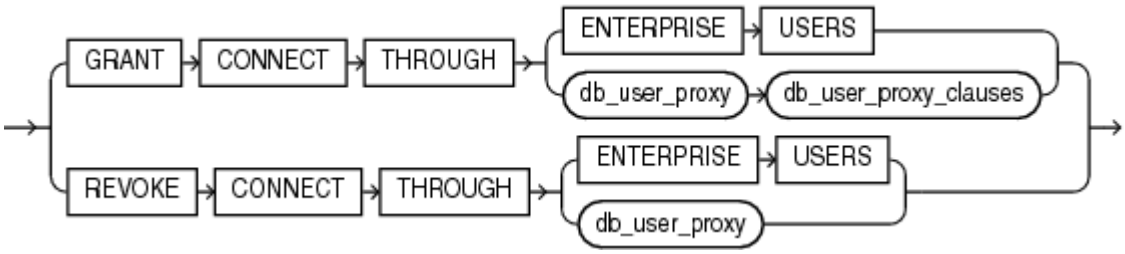

ALTER
USER
user
,
user
IDENTIFIED
BY
password
REPLACE
old_password
EXTERNALLY
AS
'
certificate_DN
kerberos_principal_name
'
GLOBALLY
AS
'
directory_DN
AZURE_USER
AZURE_ROLE
=
value
IAM_GROUP_NAME
IAM_PRINCIPAL_NAME
=
value
'
NO
AUTHENTICATION
DEFAULT
COLLATION
collation_name
DEFAULT
TABLESPACE
tablespace
LOCAL
TEMPORARY
TABLESPACE
tablespace
tablespace_group_name
QUOTA
size_clause
UNLIMITED
ON
tablespace
PROFILE
profile
DEFAULT
ROLE
,
role
ALL
EXCEPT
,
role
NONE
PASSWORD
EXPIRE
ACCOUNT
LOCK
UNLOCK
ENABLE
EDITIONS
FOR
,
object_type
FORCE
HTTP
DIGEST
ENABLE
DISABLE
CONTAINER
=
CURRENT
ALL
container_data_clause
,
proxy_clause

(size_clause::=)

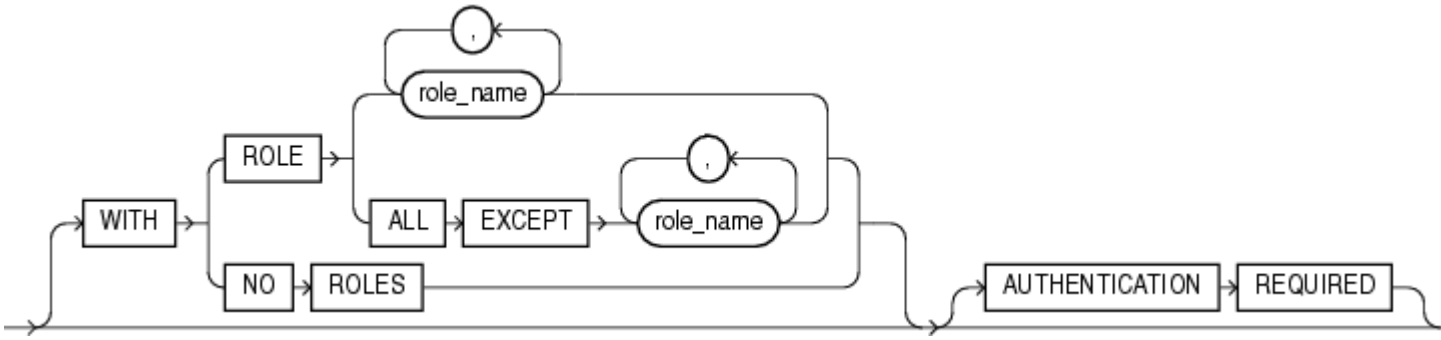
container_data_clause::=



proxy_clause::=



db_user_proxy_clauses::=



セマンティクス

この項で説明するキーワード、パラメータおよび句は、ALTER USER独自か、または、CREATE USERでの場合とセマンティクスが異なります。その他のキーワード、パラメータおよび句には、CREATE USER文と同じ意味があります。

ノート:

ユーザー名およびパスワードは、ご使用のプラットフォームに応じて、ASCII または EBCDIC 文字のみでエンコードすることをお薦めします。

関連項目:

キーワードおよびパラメータについては、「CREATE USER」を参照してください。ユーザーにデータベース・リソースへの制限を割り当てる方法については、「CREATE PROFILE」を参照してください。

IDENTIFIED句

BY password

BY passwordを指定すると、ユーザーの新しいパスワードを指定できます。パスワードは大/小文字が区別されます。この後に、ユーザーをデータベースに接続するために使用されるCONNECT文字列は、このALTER USER文で使用されているものと同じ文字(大文字、小文字または混在)を使用してパスワードを指定する必要があります。パスワードには、データベース文字セットから、シングルバイト文字、マルチバイト文字または両方を含めることができます。

ノート:

異なるタイムスタンプで特定のパスワードを再設定する必要があります。1 秒以内に 1 つのパスワードを複数回再設定した場合(たとえば、スクリプトを使用して一連のパスワードの設定を繰り返した場合)、データベースはパスワードが再利用できないというエラー・メッセージを戻すことがあります。このため、パスワードの再設定には、スクリプトを使用しないことをお薦めします。

自分のパスワードを設定する場合、またはALTER USERシステム権限を持っていて、他のユーザーのパスワードを変更する場合は、REPLACE句を省略できます。ただし、ALTER USERシステム権限を持たないかぎり、複雑なパスワードの検証機能が使用可能な場合は、UTLPWDMG.SQLスクリプトを実行するか、またはユーザーに割り当てられたプロファイルのPASSWORD_VERIFY_FUNCTIONパラメータに検証機能を指定して、常にREPLACE句を指定する必要があります。

Oracle ASMクラスタで、この句を使用して、ユーザーのパスワードを、現行のノードのOracle ASMインスタンスに対してローカルなパスワード・ファイルで変更できます。REPLACE old_password句を使用せずにIDENTIFIED BY passwordを指定するには、AS SYSASMとして認証されている必要があります。AS SYSASMとして認証されていない場合は、REPLACE old_passwordを指定して自分のパスワードのみを変更できます。

以前のパスワードをREPLACE句に指定した場合でも、自分以外の既存のパスワードを変更している場合は、以前のパスワードはチェックされません。

段階的なデータベース・パスワード・ロールオーバー期間を開始するためのパスワードの変更

前提条件

CREATE PROFILEまたはALTER PROFILEを使用してPASSWORD_ROLLOVER_TIMEユーザー・プロファイル・パラメータにゼロ以外の値を設定し、段階的なデータベース・パスワード・ロールオーバー期間を有効にします。

PASSWORD_ROLLOVER_TIMEを設定して、ユーザーのプロファイルで段階的なパスワード・ロールオーバー期間を指定した後、ALTER USER文を使用してユーザーのパスワードを変更でき、これにより、パスワード・ロールオーバー期間が期限切れになるまで、クライアントは古いパスワードと新しいパスワードの両方を使用してログインできるようになります。

パスワード・ロールオーバー期間中に、(PASSWORD_ROLLOVER_TIMEが終了する前に)新しいパスワードをすべてのクライアントに伝播する必要があります。新しいパスワードを早期に(パスワード・ロールオーバー期間の終了前に)すべてのクライアントに正常に伝播した場合は、EXPIRE PASSWORD ROLLOVER PERIOD句を使用してパスワード・ロールオーバーを終了できます(新しいパスワードのみを使用できるように、パスワードの変更をファイナライズできます)。

段階的なデータベース・パスワード・ロールオーバー期間中のパスワードの変更

パスワード・ロールオーバー期間中に(ロールオーバー期間が期限切れになる前に)、REPLACE句の有無にかかわらず、ALTER

USERを使用してパスワードを変更できます。

たとえば、ユーザーu1が元のパスワードp1を持ち、p2がロールオーバー・プロセスを開始した新しいパスワードであるとします。ここで、p2のかわりにp3に切り替えます。次のいずれかの文を使用して、パスワードをp3に変更できます。

```
ALTER USER u1 IDENTIFIED BY p3;

ALTER USER u1 IDENTIFIED BY p3 REPLACE p1;

ALTER USER u1 IDENTIFIED BY p3 REPLACE p2;
```

パスワードをp3に変更した後、ユーザーはp1またはp3を使用してログインできます。p2を使用してログインすると、エラー`Invalid credential or not authorized; logon denied`が戻され、ログイン試行の失敗として記録されます。

ロールオーバーの開始時間は、パスワード変更のタイムスタンプ、つまり、ユーザーのパスワードが変更された時間に設定されたままとなります。ロールオーバーの開始時間とパスワードの変更時間は、パスワード・ロールオーバー期間中にさらにパスワードが変更されても影響を受けません。古いパスワードは、最大で`PASSWORD_ROLLOVER_TIME`の日数の間使用できます。

関連項目:

パスワード作成のガイドラインに関しては『Oracle Databaseセキュリティ・ガイド』を参照してください

GLOBALLY

この句の詳細は、「CREATE USER」を参照してください。

ユーザー・アクセスの検証方法は、`IDENTIFIED GLOBALLY`から`IDENTIFIED BY password`または`IDENTIFIED EXTERNALLY`のいずれかに変更できます。ユーザーのアクセス検証方法を他のいずれかの検証方法から`IDENTIFIED GLOBALLY`に変更できるのは、ユーザーに明示的に付与されたすべての外部ロールが取り消された場合のみです。

EXTERNALLY

この句の詳細は、「CREATE USER」を参照してください。

関連項目:

グローバルまたは外部で識別されるユーザーの詳細は、『Oracle Databaseエンタープライズ・ユーザー・セキュリティ管理者ガイド』を参照してください。また、「ユーザー識別の変更: 例」および「ユーザー認証の変更: 例」も参照してください。

NO AUTHENTICATION句

この句を使用して、認証を使用する既存のユーザー・アカウントを認証を使用しないスキーマ・アカウントに変更して、アカウントにログインできないようにします。

DEFAULT COLLATION句

この句を使用すると、ユーザーが所有するスキーマのデフォルトの照合を変更できます。新しいデフォルトの照合は、それ以降にスキーマで作成される表、ビューおよびマテリアライズド・ビューに割り当てられます。これは、既存の表ビューおよびマテリアライズド・ビューのデフォルトの照合には影響を及ぼしません。この句のセマンティクスの詳細は、「CREATE USER」の「DEFAULT COLLATION句」を参照してください。

DEFAULT TABLESPACE句

この句を使用すると、ユーザーの永続セグメントの表領域の割当てまたは再割当てを行うことができます。この句は、データベース用に指定されているデフォルトの表領域を上書きします。

デフォルト表領域の制限事項

ローカル管理の一時表領域(UNDO表領域を含む)またはディクショナリ管理の一時表領域は、ユーザーのデフォルトの表領域として指定できません。

[LOCAL] TEMPORARY TABLESPACE句

この句を使用すると、ユーザーの一時セグメントの一時表領域または表領域グループの割当てまたは再割当てを行うことができます。

- `tablespace`に、ユーザーの一時セグメント表領域を指定します。共有一時表領域を指定するには、TEMPORARY TABLESPACEを指定します。ローカル一時表領域を指定するには、LOCAL TEMPORARY TABLESPACEを指定します。CDBに接続しているときに、CDB$DEFAULTを指定すると、CDB全体のデフォルト一時表領域を使用できます。
- `tablespace_group_name`を指定すると、ユーザーは、`tablespace_group_name`で指定された表領域グループ内の任意の表領域に一時セグメントを保存できるようになります。ローカル一時表領域を表領域グループに含めることはできません。

ユーザーの一時表領域の制限事項

ユーザーの一時表領域として割り当てる表領域、または再度割り当てる表領域は、標準的なブロック・サイズの一時表領域である必要があります。

関連項目:

表領域グループの割当て: 例

DEFAULT ROLE句

ログオン時に、ユーザーに対してデフォルトで有効になるロールを指定します。この句では、GRANT文を使用してユーザーに直接付与されているロール、またはCREATE ROLE権限を持つユーザーが作成したロールのみ指定できます。DEFAULT ROLE句を使用して次のロールを指定することはできません。

- ユーザーに付与されていないロール
- 他のロールを介して付与されているロール
- 外部サービス(オペレーティング・システムなど)またはOracle Internet Directoryによって管理されるロール
- パスワード認証されるロールや保護アプリケーション・ロールなど、SET ROLE文によって使用可能になるロール

関連項目:

CREATE ROLE

CDBの共通ユーザーへのデフォルト・ロールの割当て

現在のコンテナや、CDB内のすべてのコンテナの共通ユーザーに割り当てられたデフォルト・ロールを変更できます。

すべてのコンテナの共通ユーザーにデフォルト・ロールを割り当てているときには、`role`は、共通ユーザーに共通に付与された共

通ロールにする必要があります。

現在のコンテナの共通ユーザーにデフォルト・ロールを割り当てているときには、roleは、次のいずれかのロールにする必要があります。

- 現在のコンテナの共通ユーザーに付与されたローカル・ロール
- 共通ユーザーに付与された共通ロール(共通に付与されたものか、現在のコンテナでローカルに付与されたもののいずれか)

EXPIRE PASSWORD ROLLOVER PERIOD句

EXPIRE PASSWORD ROLLOVER PERIODを使用して、パスワード・ロールオーバー期間を手動で終了できます。

ENABLE EDITIONS

この句は元に戻すことができません。ENABLE EDITIONSを指定すると、ユーザーは、エディションを使用しているスキーマ内で、エディション化可能なオブジェクトの複数のバージョンを作成できるようになります。エディションが有効でないスキーマ内のエディション化可能なオブジェクトは、エディション化できません。

FOR句を使用すると、ユーザーが作成できるエディション化可能オブジェクトについて、1つ以上のオブジェクト・タイプを指定できます。object_typeの有効な値のリストを表示するには、V$EDITIONABLE_TYPES動的パフォーマンス・ビューを問い合せます。FOR句を省略すると、ユーザーは、すべてのエディション化可能オブジェクト・タイプのエディション化可能オブジェクトを作成できるようになります。

FOR句を省略した場合、スキーマでエディション化可能になる型は、VIEW、SYNONYM、PROCEDURE、FUNCTION、PACKAGE、PACKAGE BODY、TRIGGER、TYPE、TYPE BODYおよびLIBRARYです。

デフォルトで有効になっていない他のオブジェクト・タイプのエディションを有効にするには、FOR句でオブジェクト・タイプを明示的に指定する必要があります。

例: デフォルトで有効化されていないエディションの有効化

```
ALTER USER username ENABLE EDITIONS FOR SQL TRANSLATION PROFILE;
```

関連項目:

- ENABLE EDITIONS句のセマンティクスの詳細は、CREATE USERの対応する項を参照してください。
- ユーザーに対するエディションの有効化
- V$EDITIONABLE_TYPES動的パフォーマンス・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

エディションを有効にするスキーマ内にエディション化が可能でないオブジェクトが含まれており、そのオブジェクトがスキーマ内のエディション化可能なオブジェクトに依存している場合は、FORCEを指定して、このスキーマのエディションを有効にする必要があります。この場合、エディションを有効にするスキーマ内にある、エディション化可能なタイプのオブジェクトに依存するオブジェクトのうち、エディション化が可能でないオブジェクトは、すべて無効になります。

[HTTP] DIGEST句

この句を使用すると、ユーザーのHTTP Digestアクセス認証を有効または無効にできます。

- HTTP Digestアクセス認証を有効にする場合は、ENABLEを指定します。この句を指定した後、ユーザーのパスワードを変更する必要があります。これにより、データベースで新しいパスワードのHTTP Digest検証が生成されます。この

ようにすることによってのみ、HTTP Digestアクセス認証が有効になります。この句を発行した後でユーザーのパスワードが確実に変更されるようにする1つの方法は、HTTP DIGEST ENABLE句と同じ文内で、次のようにPASSWORD EXPIRE句を指定することです。

```
ALTER USER user PASSWORD EXPIRE HTTP DIGEST ENABLE;
```

これにより、ユーザーが次回データベースにログインしようとすると、新しいパスワードを要求されます。その後、ユーザーのHTTP Digestアクセス認証が有効になります。

- ユーザーのHTTP Digestアクセス認証を無効にする場合は、DISABLEを指定します。この句を有効にするために、ユーザーのパスワードを変更する必要はありません。DISABLE句を指定すると、ディクショナリ表からHTTPダイジェストが削除されます。

  ```
  ALTER USER user PASSWORD EXPIRE HTTP DIGEST DISABLE;
  ```

この句の詳細は、CREATE USERのドキュメントの「[HTTP] DIGEST句」を参照してください。

CONTAINER句

現在のコンテナがPDBである場合、CONTAINER = CURRENTを指定すると、現在のコンテナ内で、ローカル・ユーザーの属性、または共通ユーザーのコンテナ固有の属性(デフォルト表領域など)を変更できます。現在のコンテナがルートである場合は、CONTAINER = ALLを指定することによって、CDB全体の共通ユーザーの属性を変更できます。現在のコンテナがPDBであるときにこの句を省略した場合、デフォルトはCONTAINER = CURRENTです。現在のコンテナがルートであるときにこの句を省略した場合、デフォルトはCONTAINER = ALLです。

CDBの共通ユーザーの変更の制限事項

共通ユーザーの特定の属性は、CDB内の一部のコンテナに限定するのではなく、すべてのコンテナに対して変更する必要があります。そのため、次のいずれかの句を使用して共通ユーザーを変更にするときには、ルートに接続し、CONTAINER=ALLを指定することによって、すべてのコンテナを変更するようにしてください。

- IDENTIFIED句
- PASSWORD句
- [HTTP] DIGEST句

container_data_clause

container_data_clauseを使用すると、共通ユーザーのCONTAINER_DATA属性を設定および変更できます。FOR句を使用すると、デフォルトのCONTAINER_DATA属性を設定または変更するのか、オブジェクト固有のCONTAINER_DATA属性を設定または変更するのかを指定できます。これらの属性によって決定されるコンテナ・セット(このセットからのルートの除外は不可)のデータが、現在のセッションがルートであるときに、CONTAINER_DATAオブジェクトを介して、指定した共通ユーザーに表示されます。

container_data_clauseを指定するには、現在のセッションがルートである必要があります。また、CONTAINER = CURRENTを指定する必要があります。

SET CONTAINER_DATA

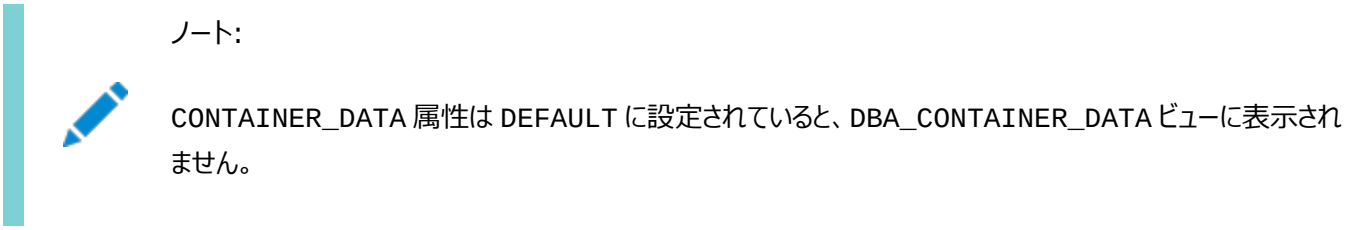
この句を使用すると、デフォルトのCONTAINER_DATA属性、または共通ユーザーのオブジェクト固有のCONTAINER_DATA属性を設定できます。この句を使用すると、CONTAINER_DATA属性に既存の値がある場合は、その値が置換されます。

container_nameを使用すると、ユーザーがアクセスできるようになる1つ以上のコンテナを指定できます。

ALLを使用すると、ユーザーは、CDBに含まれる現在または将来のすべてのコンテナにアクセスできるようになります。

DEFAULTを使用すると、次に示すデフォルトの動作を指定できます。

- デフォルトのCONTAINER_DATA属性の場合、ユーザーは、現在のコンテナ(つまり、ルート)と、CDB全体にアクセスできるようになります。
- オブジェクト固有のCONTAINER_DATA属性の場合、データベースはユーザーのデフォルトのCONTAINER_DATA属性を使用します。



ノート:

CONTAINER_DATA 属性は DEFAULT に設定されていると、DBA_CONTAINER_DATA ビューに表示されません。

ADD CONTAINER_DATA

この句を使用すると、デフォルトのCONTAINER_DATA属性、または共通ユーザーのオブジェクト固有のCONTAINER_DATA属性にコンテナを追加できます。container_nameを使用すると、追加するコンテナを1つ以上指定できます。

この句は、デフォルトのCONTAINER_DATA属性にALLを設定しているときには指定できません。デフォルトのCONTAINER_DATA属性にDEFAULTを設定しているときに、この句を使用すると、コンテナのセットにCDB$ROOTが自動的に追加されます(このセットにCDB$ROOTがすでに含まれていない場合)。

この句は、オブジェクト固有のCONTAINER_DATA属性をALLまたはDEFAULTに設定されているときには使用できません。

REMOVE CONTAINER_DATA

この句を使用すると、デフォルトのCONTAINER_DATA属性、または共通ユーザーのオブジェクト固有のCONTAINER_DATA属性からコンテナを削除できます。container_nameを使用すると、削除するコンテナを1つ以上指定できます。

この句は、デフォルトのCONTAINER_DATA属性またはオブジェクト固有のCONTAINER_DATA属性をALLまたはDEFAULTに設定しているときには使用できません。

FOR container_data_object

FOR句を指定すると、共通ユーザーのcontainer_data_objectのオブジェクト固有のCONTAINER_DATA属性を設定および変更できます。container_data_objectは、CONTAINER_DATA表またはビューであることが必要です。schemaを省略すると、そのcontainer_data_objectは自分のスキーマ内にあるとみなされます。

FOR句を省略すると、共通ユーザーのデフォルトのCONTAINER_DATA属性を設定および変更できるようになります。

関連項目:

PDBオブジェクトに関する情報が共通ユーザーに表示されるようにする方法の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

proxy_clause

proxy_clauseを使用すると、エンタープライズ・ユーザー(データベースの外側のユーザー)またはデータベース・プロキシ(別のデータベース・ユーザー)が、変更対象のデータベース・ユーザーとして、どのように接続できるようにするかを制御できます。

GRANT CONNECT THROUGH

GRANT CONNECT THROUGHを指定すると、接続を許可できます。

REVOKE CONNECT THROUGH

REVOKE CONNECT THROUGHを指定すると、接続を禁止できます。

ENTERPRISE USER

この句を使用すると、userを、エンタープライズ・ユーザーがプロキシ使用できるように公開できます。Oracle Internet Directoryを担当する管理者は、userの代理として作業するエンタープライズ・ユーザーに権限を適切に付与する必要があります。

db_user_proxy

この句を使用すると、userを、データベース・ユーザーdb_user_proxy (プロキシ)がプロキシ使用できるように公開できます。

- プロキシは、userに直接付与されたすべての権限を持ちます。
- db_user_proxy_clausesのWITH句を指定してプロキシをuserの一部のロールまたはロールなしに制限した場合を除き、プロキシはuserに関連付けられているすべてのロールを持ちます。プロキシに関連付けられている各ロールについて、デフォルトでログイン時にuserに対してロールが有効化される場合、プロキシに対してもデフォルトでログイン時にそのロールが有効化されます。

db_user_proxy_clauses

プロキシ・セッションで、パスワード保護されたロールを有効にできます。セキュア・アプリケーション・ロールでも、パスワード保護されたロールでも、セッションでロールを有効にする安全な方法があります。安全でないチャネルでパスワードを管理、転送しなければならない場合には、または複数の担当者がパスワードを知る必要がある場合には、パスワード保護されたロールではなく、セキュア・パスワード・ロールを使用することをお薦めします。プロキシ・セッションの場合、パスワード保護されたロールは、自動化処理を利用してロールを設定する状況に適しています。

プロキシ・ユーザーはパスワード保護されたロールにアクセスできます。WITH句を指定して、プロキシをuserに関連付けられている一部のロールまたはロールなしに制限し、AUTHENTICATION REQUIRED句を使用して、認証が必要かどうかを指定します。

WITH ROLE

WITH ROLE role_nameを使用すると、プロキシは指定したユーザーとして接続でき、role_nameで指定されたロールのみをアクティブにできますこの句には、userに関連付けられているロールのみを含めることができます。プロキシ・ユーザーをセキュアなロールとして使用する必要がある場合、パスワード保護されたロールと、セキュア・アプリケーション・ロールは、WITH ROLE句で使われている必要があります。 こうしたセキュアなロールを、WITH ROLE ALL句に追加します(WITH ROLEを指定しない場合はデフォルト)。 WITH ROLEにセキュアなロールを指定しない場合は、正しいパスワードを使っても有効にできません。

WITH ROLE ALL EXCEPT

WITH ROLE ALL EXCEPT role_nameを使用すると、プロキシは指定したユーザーとして接続でき、role_nameで指定されたロール以外の、このユーザーに対応付けられたすべてのロールをアクティブにできます。この句には、userに関連付けられているロールのみを含めることができます。

WITH NO ROLES

WITH NO ROLESを使用すると、プロキシは指定したユーザーとして接続できますが、パスワード保護されたロールなどのセキュア・ロールとセキュア・アプリケーション・ロールであっても、接続後にそのユーザーのロールを1つでもアクティブにすることは禁止されます。

AUTHENTICATION REQUIRED

AUTHENTICATION REQUIRED句を指定しない場合、Oracle Databaseは、ユーザーの認証にプロキシを想定しません。この句は、ユーザーが指定されたプロキシを介して認証される場合に、ユーザーの認証資格証明が提示される必要があることを示しています。資格証明は、パスワードです。

AUTHENTICATED USING

以前のリリースの構文に出現したAUTHENTICATED USING句は非推奨になり、必要でなくなりました。AUTHENTICATED USING PASSWORD句を指定した場合、AUTHENTICATION REQUIRED句に変換されます。AUTHENTICATED USING CERTIFICATE句またはAUTHENTICATED USING DISTINGUISHED NAME句を指定することは、AUTHENTICATION REQUIRED句を省略することと同じです。

関連項目:

- データベース・セキュリティの概要および中間層システムおよびプロキシ認証の詳細は、『Oracleセキュリティ概要』を参照してください。
- プロキシおよびデータベースの使用方法の詳細は、『Oracle Databaseセキュリティ・ガイド』および「プロキシ・ユーザー: 例」を参照してください。

例

ユーザー識別の変更: 例

次の文は、ユーザーsidney (「データベース・ユーザーの作成: 例」で作成)のパスワードをsecond_2nd_pwdに、デフォルト表領域を表領域exampleに変更します。

```
ALTER USER sidney
    IDENTIFIED BY second_2nd_pwd
    DEFAULT TABLESPACE example;
```

次の文は、new_profileプロファイル(「プロファイルの作成: 例」で作成)をサンプル・ユーザーshに割り当てます。

```
ALTER USER sh
    PROFILE new_profile;
```

後続のセッションでは、shはnew_profileプロファイルの制限に従います。

次の文は、shに直接付与されているすべてのロール(dw_managerロールを除く)をデフォルト・ロールに設定します。

```
ALTER USER sh
    DEFAULT ROLE ALL EXCEPT dw_manager;
```

shの次のセッションの開始時には、dw_manager以外でshに直接付与されているすべてのロールが使用可能になります。

ユーザー認証の変更: 例

次の文は、ユーザーapp_user1(「データベース・ユーザーの作成: 例」で作成)の認証メカニズムを変更します。

```
ALTER USER app_user1 IDENTIFIED GLOBALLY AS 'CN=tom,O=oracle,C=US';
```

次の文は、ユーザーsidneyのパスワードを期限切れにします。

```
ALTER USER sidney PASSWORD EXPIRE;
```

PASSWORD EXPIREを使用してデータベース・ユーザーのパスワードを期限切れにした場合、そのユーザー(またはDBA)は、期限切れの後でデータベースにログインする際、パスワードを変更する必要があります。ただし、SQL*Plusなどのツール製品を使用した場合、期限切れの後の最初のログイン時に、パスワードを変更できます。

表領域グループの割当て: 例

次の文は、tbs_grp_01 (「表領域グループへの一時表領域の追加: 例」で作成)を表領域グループとしてユーザーshに割り当てます。

```
ALTER USER sh
  TEMPORARY TABLESPACE tbs_grp_01;
```

プロキシ・ユーザー: 例

次の文は、ユーザーapp_user1を変更します。例では、app_user1がプロキシ・ユーザーshを使用して接続できます。また、プロキシshを使用して接続したときに、app_user1がwarehouse_userロール(「ロールの作成: 例」で作成)を有効にできます。

```
ALTER USER app_user1
   GRANT CONNECT THROUGH sh
   WITH ROLE warehouse_user;
```

基本的な構文を示すため、サンプル・データベースSales Historyのユーザー(sh)をプロキシとして使用します。通常、プロキシ・ユーザーは、アプリケーション・サーバーまたは中間層のエンティティに存在します。アプリケーション・サーバーを経由して、アプリケーション・ユーザーとデータベースの間のインタフェースを作成する場合の詳細は、『Oracle Call Interfaceプログラマーズ・ガイド』を参照してください。

関連項目:

- app_userユーザーの作成方法の詳細は、「外部データベース・ユーザーの作成: 例」を参照してください。
- dw_userロールの作成方法の詳細は、「ロールの作成: 例」を参照してください。

次の文は、ユーザーapp_user1がプロキシ・ユーザーshを使用して接続する権限を取り消します。

```
ALTER USER app_user1 REVOKE CONNECT THROUGH sh;
```

次の仮想例は、プロキシ認証の他の方法を示します。

```
ALTER USER sully GRANT CONNECT THROUGH OAS1
   AUTHENTICATED USING PASSWORD;
```

次の例では、ユーザーapp_user1をエンタープライズ・ユーザーがプロキシ使用できるように公開します。エンタープライズ・ユーザーは、Oracle Internet Directory管理者が適切な権限を付与するまでは、app_user1の代理として作業することができません。

```
ALTER USER app_user1
   GRANT CONNECT THROUGH ENTERPRISE USERS;
```

ALTER VIEW

目的

ALTER VIEWを使用すると、無効なビューを明示的に再コンパイルしたり、ビューの制約を変更することができます。明示的に再コンパイルすると、実行前にコンパイル・エラーを検査できます。再コンパイルは、ビューの実表を変更した後で、その変更がビューまたはそのビューに依存するオブジェクトに影響していないかどうかを確認するときに便利です。

ALTER VIEWを使用して、制約のビューを定義、変更または削除することもできます。

この文を使用して既存のビュー定義を変更することはできません。また、ビューの実表に対するDDL変更によってビューが無効になる場合、この文を使用して無効なビューをコンパイルすることはできません。このような場合は、CREATE VIEWにOR REPLACEキーワードを指定して使用し、ビューを再定義する必要があります。

ALTER VIEW文を発行した場合、指定したビューは有効か無効かにかかわらず再コンパイルされます。また、そのビューに依存するすべてのローカル・オブジェクトが無効になります。

1つ以上のマテリアライズド・ビューが参照しているビューを変更した場合、これらのマテリアライズド・ビューは無効になります。無効なマテリアライズド・ビューは、クエリー・リライトには使用できず、リフレッシュもできません。

関連項目:

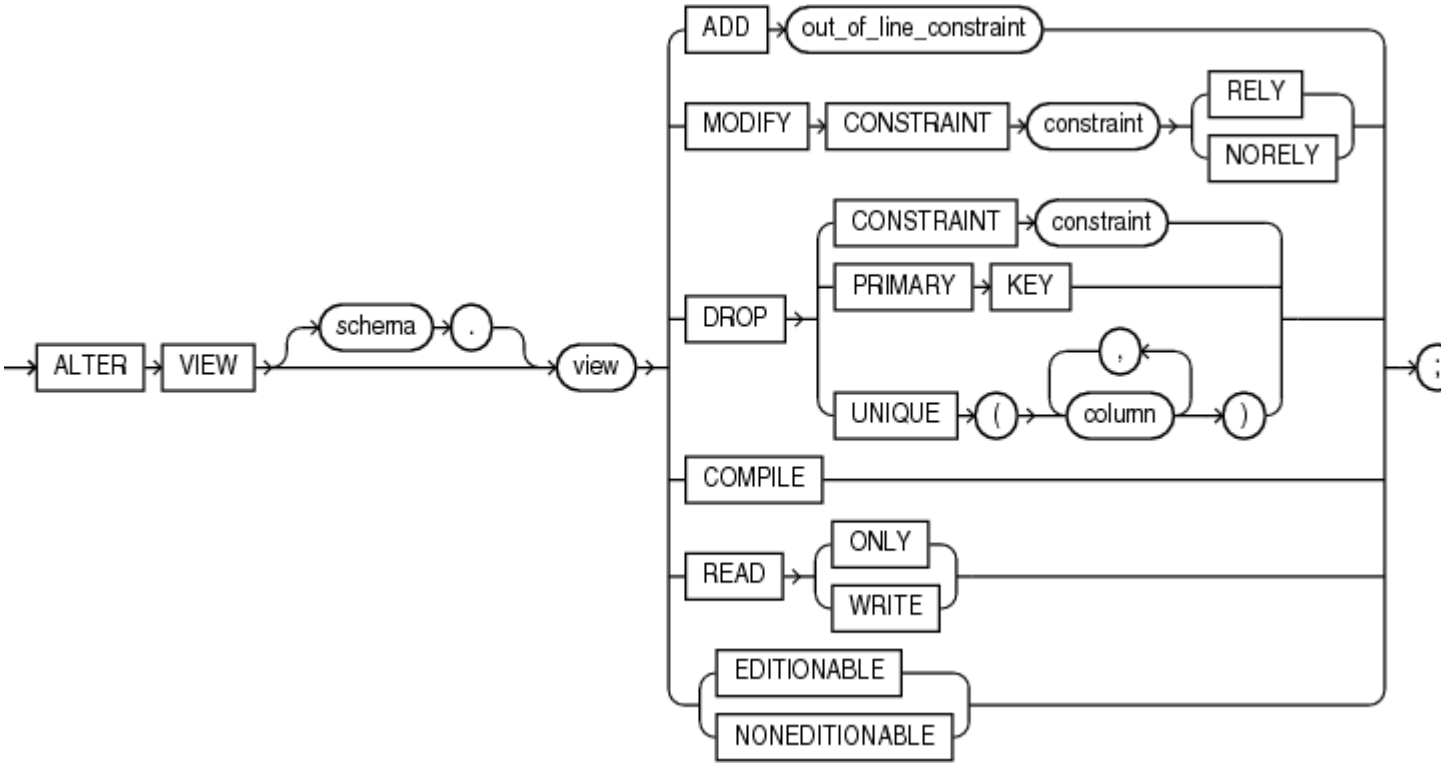
- ビューの再定義の詳細は、「CREATE VIEW」を参照してください。無効なマテリアライズド・ビューの再検証の詳細は、「ALTER MATERIALIZED VIEW」を参照してください。
- データ・ウェアハウスの概要は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- スキーマ・オブジェクト間の依存性については、『Oracle Database概要』を参照してください。

前提条件

ビューが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY TABLEシステム権限が必要です。

構文

alter_view::=



(out_of_line_constraint::= (constraint::=構文の一部))

セマンティクス

schema

ビューが含まれているスキーマを指定します。schemaを指定しない場合、ビューは自分のスキーマ内にあるとみなされます。

view

再コンパイルするビューの名前を指定します。

MODIFY CONSTRAINT句

MODIFY CONSTRAINT句を使用すると、既存のビュー制約のRELYまたはNORELY設定を変更できます。ビュー制約の概要は、「ビュー制約のノート」を参照してください。

制約の変更の制限事項

一意制約または主キー制約が参照整合性制約の一部である場合、外部キーを削除するか、viewの設定に合うように変更しないで、設定を変更することはできません。

ADD句

ADD句を使用すると、viewに制約を追加できます。ビュー制約および制限事項については、「constraint」を参照してください。

DROP句

DROPを使用すると、既存のビューの制約を削除できます。

制約の削除の制限事項

一意制約または主キー制約がビューの参照整合性制約の一部である場合、それらの制約は削除できません。

COMPILE

COMPILEキーワードを指定すると、ビューを再コンパイルできます。

{ READ ONLY | READ WRITE }

これらの句は、エディショニング・ビューに対してのみ有効です。

- READ ONLYを指定すると、エディショニング・ビューを更新禁止にできます。
- READ WRITEを指定すると、読取り専用エディショニング・ビューを読取り/書込みステータスに戻すことができます。

これらの句を指定すると、依存オブジェクトは無効になりませんが、カーソルが無効になることがあります。

EDITIONABLE | NONEDITIONABLE

この句を使用すると、schemaのスキーマ・オブジェクト・タイプVIEWのエディショニングが後で有効化されたときに、そのビューをエディション・オブジェクトにするか非エディション・オブジェクトにするかを指定できます。デフォルトは、EDITIONABLEです。エディション・オブジェクトと非エディション・オブジェクトの変更の詳細は、『Oracle Database開発ガイド』を参照してください。

関連項目:

エディショニング・ビューの詳細は、「CREATE VIEW」を参照してください。

例

ビューの変更: 例

次の文は、ビューcustomer_ro (「読取り専用ビューの作成: 例」で作成)を再コンパイルします。

```
ALTER VIEW customer_ro
    COMPILE;
```

customer_roの再コンパイル時にエラーが発生しなければ、customer_roは有効になります。再コンパイル・エラーが発生した場合はエラーが戻り、customer_roは無効のままとなります。

依存するオブジェクトもすべて無効になります。依存オブジェクトとは、customer_roを参照する、プロシージャ、ファンクション、パッケージ本体、ビューなどです。その後、最初に明示的に再コンパイルせずにこれらのオブジェクトのいずれかを参照すると、データベースによって実行時にそのオブジェクトが暗黙的に再コンパイルされます。

ANALYZE

目的

ANALYZE文を使用すると、統計情報を収集して、次のような操作を実行できます。

- 索引または索引パーティション、表または表パーティション、索引構成表、クラスタまたはスカラー・オブジェクト属性の統計情報を収集または削除します。
- 索引または索引パーティション、表または表パーティション、索引構成表、クラスタまたはオブジェクト参照(REF)の構造を検証します。
- 表またはクラスタの移行行と連鎖行を識別します。

ノート:

オプティマイザ統計の収集での ANALYZE の使用はサポートされなくなりました。

オプティマイザ統計の収集が必要な場合は、DBMS_STATS パッケージを使用してパラレルで統計を収集し、パーティション化されたオブジェクトのグローバル統計を収集できます。これにより、他の方法で統計収集を微調整できます。DBMS_STATS パッケージの詳細は、『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』を参照してください。

ANALYZE 文は、次の場合にのみ使用してください。

- VALIDATE 句、LIST CHAINED ROWS 句の使用
- 空きリスト・ブロックの情報を収集する場合

前提条件

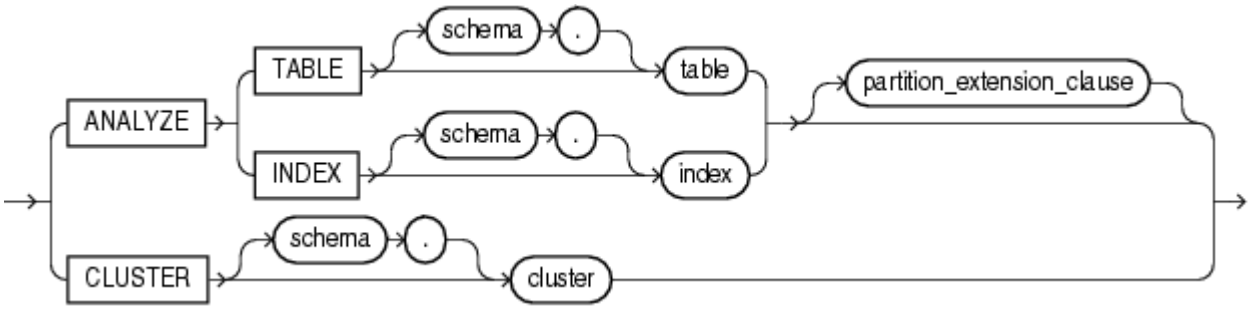
分析するスキーマ・オブジェクトがローカルである必要があります。自分のスキーマ内にない場合は、ANALYZE ANYシステム権限が必要です。

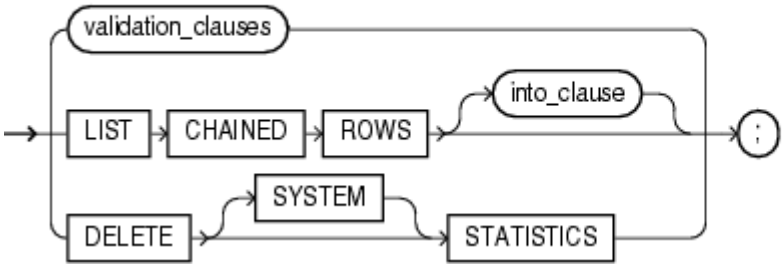
表またはクラスタの連鎖行をリスト表へ入れる場合、このリスト表が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、そのリスト表のINSERT権限またはINSERT ANY TABLEシステム権限が必要です。

パーティション表の妥当性チェックを行う場合は、分析したROWIDを入れる表に対するINSERTオブジェクト権限またはINSERT ANY TABLEシステム権限が必要です。

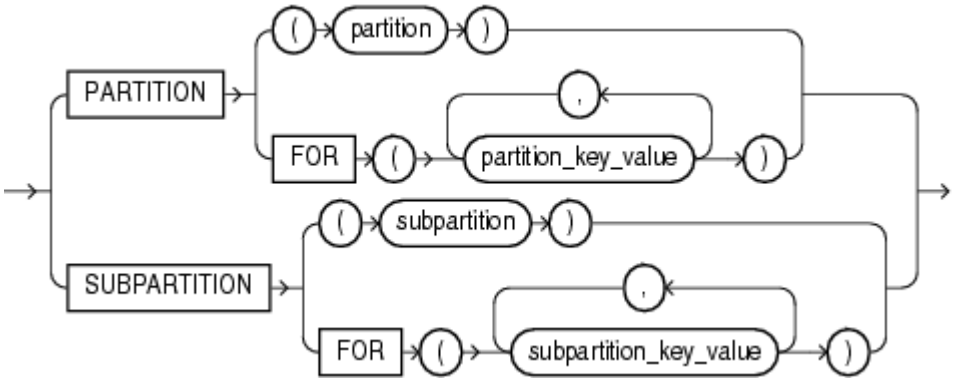
構文

analyze::=

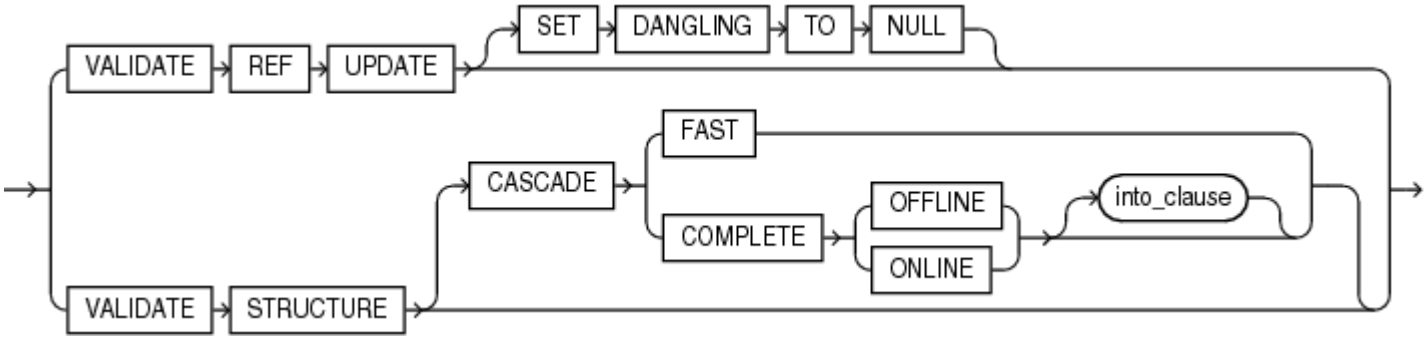




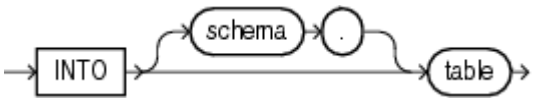
partition_extension_clause::=



validation_clauses::=



into_clause::=



セマンティクス

schema

表、索引またはクラスタが含まれているスキーマを指定します。schemaを指定しない場合、表、索引またはクラスタは自分のス

キーマ内にあるとみなされます。

TABLE table

分析する表を指定します。表を分析すると、すべてのファンクション索引に発生する式について統計情報が収集されます。したがって、表を分析する前に、必ずファンクション索引を作成してください。ファンクション索引の詳細は、「CREATE INDEX」を参照してください。

表を分析すると、LOADINGまたはFAILEDのマークが付いたドメイン索引はすべてスキップされます。

索引構成表の場合、マッピング表が分析され、そのPCT_ACCESSS_DIRECT統計情報も計算されます。これらの統計情報は、マッピング表のローカルROWIDの一部として格納されたと推測されるデータ・ブロック・アドレスの精度を評価します。

表については、次の統計情報が収集されます。アスタリスクが付いた統計情報は、常に厳密に計算されます。表の統計情報(ドメイン索引の状態を含む)は、データ・ディクショナリ・ビューUSER_TABLES、ALL_TABLESおよびDBA_TABLESのカッコで示す列に表示されます。

- 行数(NUM_ROWS)
- 最高水位標を下回るデータ・ブロックの数(現在データを含むか含まないかにかかわらず、データを格納するようにフォーマットされているデータ・ブロックの数)(BLOCKS)*
- 未使用の表に対して割り当てられているデータ・ブロックの数(EMPTY_BLOCKS)*
- 各データ・ブロックにおける使用可能な空き領域サイズの平均値(バイト単位)(AVG_SPACE)
- 連鎖行の数(CHAIN_COUNT)
- 行のオーバーヘッドを含む、バイト単位での行の平均の長さ(AVG_ROW_LEN)

表の分析の制限事項

表の分析には、次の制限事項があります。

- ANALYZEを使用して、データ・ディクショナリ表の統計情報を収集しないでください。
- ANALYZEを使用して、外部表の統計情報を収集しないでください。かわりに、DBMS_STATSパッケージを使用する必要があります。
- ANALYZEを使用して、一時表のデフォルト統計情報を収集しないでください。ただし、すでに一時表の1つ以上の列とユーザー定義統計タイプを対応付けている場合、ANALYZEを使用して一時表のユーザー定義統計情報を収集できます
- REF列型、VARRAY、ネストした表、LOB列型(LOB列型は分析されずにスキップされる)、LONG列型、オブジェクト型などの列型の統計情報は計算または推定できません。ただし、このような列に統計タイプが対応付けられている場合は、ユーザー定義統計情報が収集されます。

関連項目:

- ASSOCIATE STATISTICS
- データ・ディクショナリ・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

partition_extension_clause

partition_extension_clause

統計情報を収集するパーティションまたはサブパーティション、あるいはパーティション値またはサブパーティション値を指定します。クラスタの分析時にこの句は使用できません。

tableがコンポジット・パーティションのときにPARTITIONを指定した場合、指定したパーティション内ですべてのサブパーティションが分析されます。

INDEX index

分析する索引を指定します。

索引については、次の統計情報が収集されます。アスタリスクが付いた統計情報は、常に厳密に計算されます。従来索引について統計情報を計算または推定する場合、統計情報は、データ・ディクショナリ・ビューUSER_INDEXES、ALL_INDEXESおよびDBA_INDEXESのカッコで示す列に表示されます。

- ルート・ブロックからリーフ・ブロックまでの索引の深さ(BLEVEL)*
- リーフ・ブロックの数(LEAF_BLOCKS)
- 個別索引値の数(DISTINCT_KEYS)
- 索引の値ごとのリーフ・ブロックの平均数(AVG_LEAF_BLOCKS_PER_KEY)
- (表に対する索引の)索引の値ごとのデータ・ブロックの平均数(AVG_DATA_BLOCKS_PER_KEY)
- クラスタ係数(索引付きの値についての行が、どれだけ効率的に順序付けられているか)(CLUSTERING_FACTOR)

ドメイン索引の場合、索引に関連付けられた統計タイプで指定したユーザー定義統計収集ファンクションが、この文によって起動されます(「ASSOCIATE STATISTICS」を参照)。ドメイン索引に関連付けられた統計タイプがない場合、その索引タイプに関連付けられた統計タイプが使用されます。索引またはその索引タイプの統計タイプがない場合、ユーザー定義統計情報は収集されません。ユーザー定義索引統計情報は、データ・ディクショナリ・ビューUSER_USTATS、ALL_USTATSおよびDBA_USTATSのSTATISTICS列に表示されます。

ノート:

- 多数の行が削除されている索引を分析する場合、統計情報操作として ESTIMATE を要求しても、COMPUTE 統計情報操作が実行され、全表スキャンが行われることがあります。このような操作は、非常に時間がかかる場合があります。
- ANALYZE 文で索引を分析すると、完了までかなりの時間がかかる場合があります。この場合、SQL 問合せを使用して索引を検証できます。問合せにより表と索引間に不整合があると判定された場合、索引を詳細に分析するために ANALYZE 文を使用できます。詳細は、『Oracle Database 管理者ガイド』を参照してください。

索引の分析の制限事項

IN_PROGRESSまたはFAILEDのマークが付けられたドメイン索引は分析できません。

関連項目:

8. ドメイン索引の詳細は、「CREATE INDEX」を参照してください。
9. データ・ディクショナリ・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

10. 索引の分析：例

CLUSTER cluster

分析するクラスタを指定します。クラスタの統計情報を収集すると、すべてのクラスタ表、およびクラスタ索引を含むすべての索引の統計情報も自動的に収集されます。

索引クラスタとハッシュ・クラスタには、単一クラスタ・キー(AVG_BLOCKS_PER_KEY)が使用するデータ・ブロックの平均数が収集されます。これらの統計情報は、データ・ディクショナリ・ビューALL_CLUSTERS、USER_CLUSTERSおよびDBA_CLUSTERSに表示されます。

関連項目:

データ・ディクショナリ・ビューの詳細は、『Oracle Databaseリファレンス』および「クラスタの分析：例」を参照してください。

validation_clauses

validation_clausesを使用すると、REF値および分析したオブジェクトの構造を検証できます。

関連項目:

表、索引、クラスタおよびマテリアライズド・ビューの検証の詳細は、『Oracle Database管理者ガイド』を参照してください。

VALIDATE REF UPDATE句

VALIDATE REF UPDATEを指定すると、指定した表内のREF値の妥当性チェックを行って、個々のREFのROWID部分を検査し、真のROWIDと比較して、必要に応じて修正できます。この句を使用できるのは、表を分析する場合のみです。

表の所有者が参照先オブジェクトに対するREADまたはSELECTオブジェクト権限を持っていない場合、このオブジェクトは無効とみなされ、NULLに設定されます。その結果、オブジェクトに対して適切な権限を持つユーザーによって発行された問合せであっても、問合せでこれらのREF値を使用することはできません。

SET DANGLING TO NULL

SET DANGLING TO NULLを指定すると、指定した表内のREF値が(範囲が限定されるかどうかにかかわらず)無効なオブジェクトまたは存在しないオブジェクトを指している場合に、REF値がNULLに設定されます。

VALIDATE STRUCTURE

VALIDATE STRUCTUREを指定すると、分析対象オブジェクトの構造を検証できます。この句で収集された統計情報はOracle Databaseオプティマイザでは使用されません。

関連項目:

表の検証：例

- 表に対して、それぞれのデータ・ブロックと行の整合性が検証されます。索引構成表の場合、表の主キー索引に対する圧縮統計情報(最適なプレフィックス圧縮件数)も生成されます。
- クラスタに対して、自動的にクラスタ表の構造が検証されます。
- パーティション表の場合、行が適切なパーティションに属するかどうかが検証されます。行が正しく照合されなかった場合は、ROWIDがINVALID_ROWS表に挿入されます。

- 一時表に対して、現行のセッション中に表の構造および索引が検証されます。
- 索引に対して、索引のそれぞれのデータ・ブロックの整合性が検証され、ブロックの破損がチェックされます。この句は、表のそれぞれの行が索引エントリを持っていること、またはそれぞれの索引エントリが表の行を指していることを確認するわけではありません。これらを確認する場合は、CASCADE句を使用して表の構造を検証します。

  通常のすべての索引用の圧縮統計情報(最適なプレフィックス圧縮件数)も計算されます。

  索引の統計情報は、データ・ディクショナリ・ビューINDEX_STATSおよびINDEX_HISTOGRAMに格納されます。

  関連項目:

  これらのビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

オブジェクトの構造に障害がある場合は、エラー・メッセージが戻ります。この場合、オブジェクトを削除して作成しなおす必要があります。

CASCADE

CASCADEを指定すると、表またはクラスタに関連付けられた索引の構造を検証できます。表を検証するときにこの句を指定すると、その表に定義された索引も検証されます。クラスタを検証するときにこの句を指定した場合、クラスタ表のすべての索引(クラスタ索引を含む)が検証されます。

デフォルトでは、CASCADEを指定すると完全な検証(COMPLETE)が実行されますが、この処理はリソースを大量に消費する可能性があります。FASTを指定すると、破損の有無の検査が行われますが、破損の詳細はレポートされません。FASTを指定したときに破損が検出された場合は、CASCADEオプションをFAST句なしで使用すると、破損の場所を特定してその詳細を知ることができます。

この句を使用して使用可能な(以前は使用禁止であった)ファンクション索引を検証すると、検証エラーになる場合があります。この場合は、索引を再構築する必要があります。

ONLINE | OFFLINE

ONLINEを指定すると、Oracle DatabaseがオブジェクトのDML操作中に検証を実行できるようになります。データベースは、並行して操作が行える程度に、実行する検証の量を減らします。

ノート:

OFFLINE 指定でオブジェクトの構造を検証する場合と同様に、ONLINE 指定でオブジェクトの構造を検証する場合、Oracle Database は統計情報を収集しません。

OFFLINEを指定すると、実行する検証の量が最大になります。この設定は、検証中にINSERT、UPDATEおよびDELETE文がオブジェクトに平行してアクセスすることを防ぎますが、問合せは許可されます。これはデフォルトです。

ONLINEの制限事項

クラスタを分析するときには、ONLINEを指定できません。

INTO

VALIDATE STRUCTUREのINTO句は、パーティション表のみに有効です。正しく照合されなかった行を持つパーティションのROWIDを格納するリスト表を指定します。schemaを指定しない場合、リストは自分のスキーマ内にあるとみなされます。この

句自体を指定しない場合、表の名前はINVALID_ROWSになります。この表を作成するために使用するSQLスクリプトはUTLVALID.SQLです。

LIST CHAINED ROWS

LIST CHAINED ROWSを指定すると、分析した表またはクラスタの移行行および連鎖行を識別できます。索引の分析時にこの句は使用できません。

INTO句には、移行行および連鎖行をリストする表を指定します。schemaを指定しない場合、連鎖行表は自分のスキーマ内にあるとみなされます。この句自体を指定しない場合、表の名前はCHAINED_ROWSになります。連鎖行表は、ローカル・データベース内にある必要があります。

次のいずれかのスクリプトを使用して、CHAINED_ROWS表を作成できます。

6. UTLCHAIN.SQL: 物理ROWIDを使用します。そのため、行は、索引構成表からではなく従来表から収集されます。(次の注意を参照。)
7. UTLCHN1.SQL: ユニバーサルROWIDを使用します。そのため、行は、従来表および索引構成表の両方から収集されます。

独自の連鎖行表を作成する場合、この2つのスクリプトのいずれかで規定されるフォーマットに従う必要があります。

分析対象の索引構成表が、ユニバーサルROWIDではなく主キーに基づくものである場合は、表の主キーを格納するために、索引構成表ごとに別の連鎖行表を作成する必要があります。SQLスクリプトDBMSIOTC.SQLおよびPRVTIOTC.PLBを使用してBUILD_CHAIN_ROWS_TABLEプロシージャを定義してから、このプロシージャを実行して、索引構成表ごとにIOT_CHAINED_ROWS表を作成します。

関連項目:

- パッケージ化されたSQLスクリプトの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』のDBMS_IOTパッケージを参照してください。
- 連鎖行のリスト: 例

DELETE STATISTICS

DELETE STATISTICSを指定すると、現在データ・ディクショナリに格納されている、分析したオブジェクトの統計情報を削除できます。Oracle Databaseに統計情報を使用させないようにする場合、この文を使用します。

表を指定してこの句を使用した場合、指定した表に定義されたすべての索引の統計情報も自動的に削除されます。クラスタを指定してこの句を使用した場合、指定したクラスタのすべての表、およびこれらの表のすべての索引(クラスタ索引を含む)の統計情報も自動的に削除されます。

ユーザー定義統計情報ではなく、システム統計情報のみを削除する場合は、SYSTEMを指定します。SYSTEMを指定しないと、オブジェクトのユーザー定義列または索引統計情報が収集された場合、データベースは、統計情報を収集するために使用された情報タイプに指定されている統計削除ファンクションを起動して、ユーザー定義統計情報も削除します。

関連項目:

「統計情報の削除: 例」

例

統計情報の削除: 例

次の文は、サンプル表oe.ordersとデータ・ディクショナリにあるそのすべての索引の統計情報を削除します。

```
ANALYZE TABLE orders DELETE STATISTICS;
```

索引の分析: 例

次の文は、サンプル索引oe.inv_product_ixの構造を検証します。

```
ANALYZE INDEX inv_product_ix VALIDATE STRUCTURE;
```

表の検証: 例

次の文は、サンプル表hr.employeesおよびそのすべての索引を分析します。

```
ANALYZE TABLE employees VALIDATE STRUCTURE CASCADE;
```

表の場合は、VALIDATE REF UPDATE句を指定すると、指定した表内のREF値が検証され、REF値それぞれのROWID部分が検査された後で、真のROWIDと比較されます。その結果、ROWIDが誤っていることが判明した場合は、ROWID部分が正しくなるようにREFが更新されます。

次の文は、サンプル表oe.customersのREF値を検証します。

```
ANALYZE TABLE customers VALIDATE REF UPDATE;
```

次の文は、DML操作を同時に実行することを許可して、サンプル表oe.customersの構造を検証します。

```
ANALYZE TABLE customers VALIDATE STRUCTURE ONLINE;
```

クラスタの分析: 例

次の文は、personnelクラスタ(「クラスタの作成: 例」で作成)、そのすべての表、クラスタ索引を含むすべての索引を分析します。

```
ANALYZE CLUSTER personnel
    VALIDATE STRUCTURE CASCADE;
```

連鎖行のリスト: 例

次の文は、表ordersのすべての連鎖行に関する情報を収集します。

```
ANALYZE TABLE orders
   LIST CHAINED ROWS INTO chained_rows;
```

前述の文では、情報は表chained_rowsに格納されます。次の問合せでその行を検証できます(表に連鎖行が含まれない場合は、行は戻されません)。

```
SELECT owner_name, table_name, head_rowid, analyze_timestamp
    FROM chained_rows
    ORDER BY owner_name, table_name, head_rowid, analyze_timestamp;
OWNER_NAME  TABLE_NAME  HEAD_ROWID         ANALYZE_TIMESTAMP
----------  ----------  ------------------ -----------------
OE          ORDERS      AAAAZzAABAAABrXAAA 25-SEP-2000
```

ASSOCIATE STATISTICS

目的

ASSOCIATE STATISTICS文を使用すると、統計収集、選択性またはコストに関するファンクションが含まれた統計タイプ(またはデフォルトの統計)を、1つ以上の列、スタンドアロン・ファンクション、パッケージ、型、ドメイン索引または索引タイプに関連付けることができます。

現在のすべての統計タイプの関連付けのリストについては、USER_ASSOCIATIONSデータ・ディクショナリ・ビューを問い合せます。統計情報に関連付けるオブジェクトを分析する場合、USER_USTATSビューでその関連性を問い合せることができます。

関連項目:

ANALYZEが関連性で使用する優先順位については、「ANALYZE」を参照してください。

前提条件

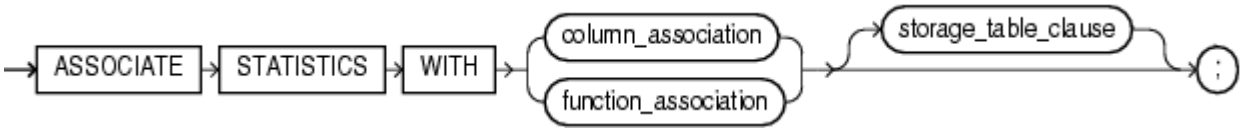
この文を発行する場合は、ベース・オブジェクト(表、ファンクション、パッケージ、型、ドメイン索引または索引タイプ)を変更する適切な権限が必要です。さらに、デフォルト統計情報のみを関連付けていないかぎり、統計タイプに対する実行権限が必要です。統計タイプは、すでに定義されている必要があります。

関連項目:

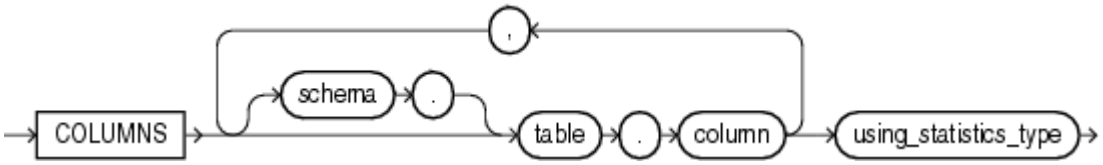
型の定義の詳細は、「CREATE TYPE」を参照してください。

構文

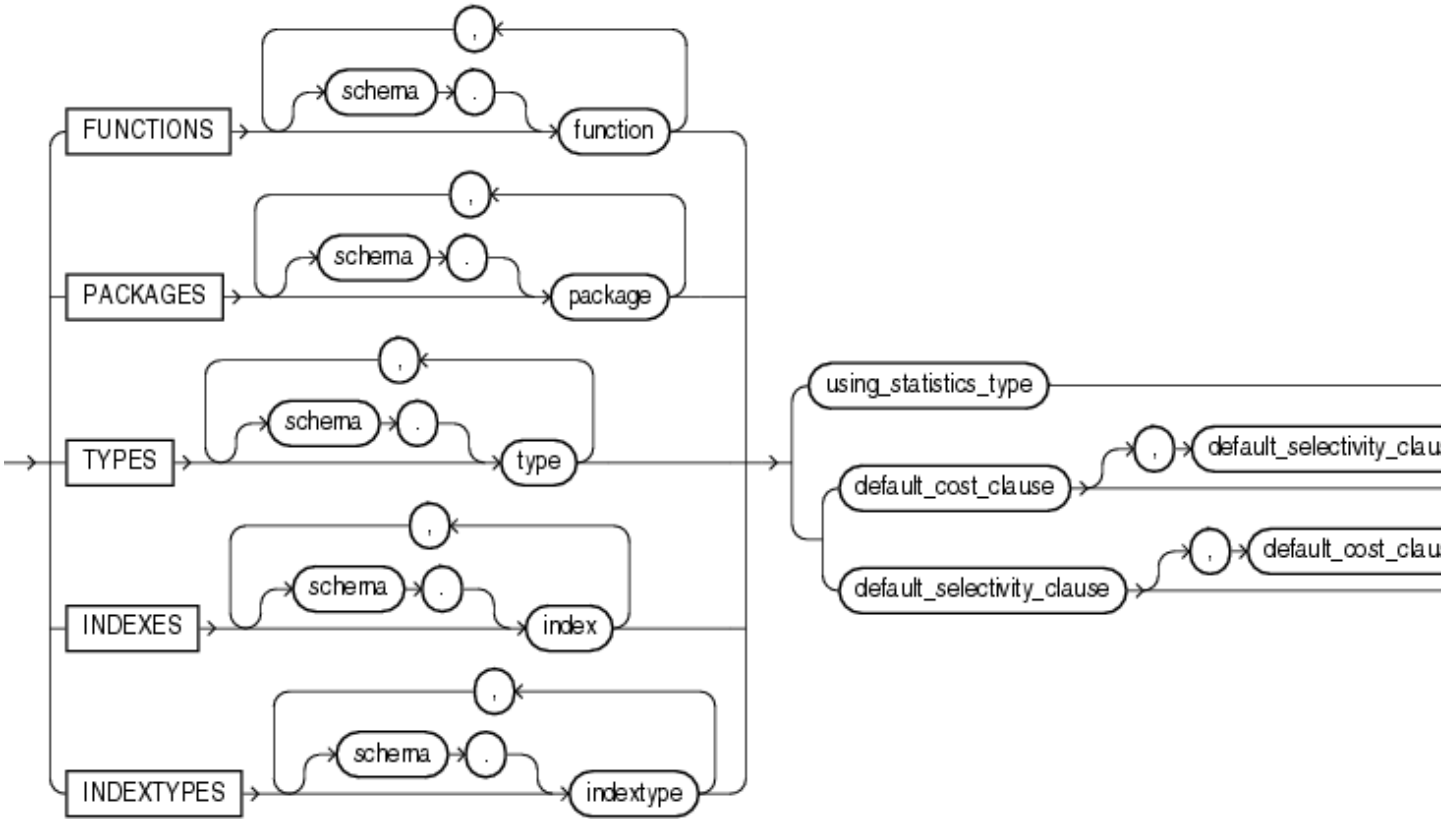
associate_statistics::=



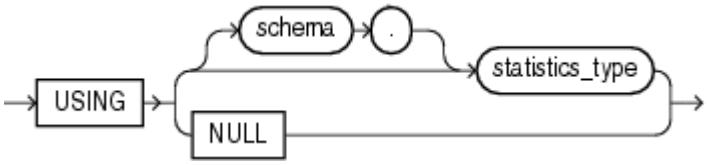
column_association::=



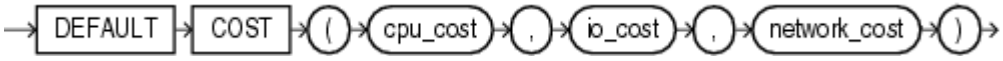
function_association::=



using_statistics_type::=



default_cost_clause::=



default_selectivity_clause::=

storage_table_clause::=

セマンティクス

column_association

1つ以上の表の列を指定します。schemaを指定しない場合、表は自分のスキーマ内にあるとみなされます。

function_association

1つ以上のスタンドアロン・ファンクション、パッケージ、ユーザー定義データ型、ドメイン索引または索引タイプを指定します。schemaを指定しない場合、オブジェクトは自分のスキーマ内にあるとみなされます。

- FUNCTIONSは、スタンドアロン・ファンクションのみを参照し、メソッド型または組込みファンクションは参照しません。
- TYPESはユーザー定義型のみを参照し、組込みSQLデータ型は参照しません。

function_associationの制限事項

関連付けをすでに定義したオブジェクトを指定することはできません。まず、このオブジェクトと統計情報の関連性を取り消す必要があります。

関連項目:

「DISASSOCIATE STATISTICS」の「統計情報の関連付け: 例」を参照してください。

using_statistics_type

列、ファンクション、パッケージ、型、ドメイン索引または索引タイプと関連付ける統計タイプ(または型のシノニム)を指定します。statistics_typeは作成済である必要があります。

NULLキーワードが有効であるのは、統計情報を列または索引に関連付ける場合のみです。統計タイプをオブジェクト型に関連付けると、そのオブジェクト型の列は統計タイプを継承します。同様に、統計タイプを索引タイプに関連付けると、その索引タイプの索引インスタンスは統計タイプを継承します。この継承を無効にするには、その列または索引に別の統計タイプを関連付けます。その列や索引に統計タイプが何も関連付けられていない状態にするには、using_statistics_type句の中でNULLを指定します。

統計タイプの指定の制限事項

ファンクション、パッケージ、タイプまたは索引タイプに、NULLを指定することはできません。

関連項目:

統計収集ファンクションの作成の詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。

default_cost_clause

スタンドアロン・ファンクション、パッケージ、型、ドメイン索引または索引タイプのデフォルトのコストを指定します。この句を指定する場合、CPUコスト、I/Oコスト、ネットワーク・コストの順でそれぞれに対して1つの数を指定する必要があります。それぞれのコストは、ファンクションまたはメソッドを一度実行した場合、またはドメイン索引に一度アクセスした場合の値です。指定できる値は0(ゼロ)以上の整数です。

default_selectivity_clause

スタンドアロン・ファンクション、型、パッケージまたはユーザー定義演算子が指定された述語に対するデフォルトの選択性をパーセントで指定します。default_selectivity_clauseは、0から100の数値である必要があります。範囲外のものは無視されます。

default_selectivity_clauseの制限事項

ドメイン索引または索引タイプに、DEFAULT SELECTIVITYを指定することはできません。

関連項目:

デフォルト・コストの指定: 例

storage_table_clause

この句は、INDEXTYPEの統計に対してのみ有効です。

- WITH SYSTEM MANAGED STORAGE TABLESを指定すると、統計データの格納がシステムで管理されます。statistics_typeに指定するタイプによって、システムで保持される表に統計関連の情報が格納されます。また、指定する索引タイプはすでに登録済であるか、またはWITH SYSTEM MANAGED STORAGE TABLES句をサポートするように変更されている必要があります。
- WITH USER MANAGED STORAGE TABLESを指定すると、ユーザー定義の統計情報を格納する表は、ユーザーによって管理されます。これはデフォルトの動作です。

例

統計情報の関連付け: 例

この文は、スタンドアロン・パッケージemp_mgmtに関連付けを作成します。このパッケージを作成する例については、『Oracle Database PL/SQL言語リファレンス』を参照してください。

```
ASSOCIATE STATISTICS WITH PACKAGES emp_mgmt DEFAULT SELECTIVITY 10;
```

デフォルト・コストの指定: 例

次の文は、ドメイン索引salary_index (「拡張索引作成機能の使用方法」で作成)を使用して任意の述語を実装した場合、常にCPUコストは100、I/Oコストは5およびネットワーク・コストは0になるように指定します。

```
ASSOCIATE STATISTICS WITH INDEXES salary_index DEFAULT COST (100,5,0);
```

オプティマイザは、コスト・ファンクションをコールするかわりに、これらのデフォルト・コストを使用します。

AUDIT (従来型監査)

この項では、従来型監査のAUDIT文について説明します。 これは、Oracle Database 12cより前のリリースで使用されていた監査機能と同じものです。

Oracle Database 12cから、高度な監査機能を完備した統合監査が導入されました。下位互換性を確保するために、従来型監査も引き続きサポートされています。ただし、既存の監査設定から新しい統合監査ポリシー構文への移行を計画することをお薦めします。新しい監査要件については、新しい統合監査を使用してください。従来型監査は、将来のメジャー・リリースでサポート対象外になる可能性があります。

関連項目:

統合監査用のAUDIT文の詳細は、「AUDIT (統合監査)」を参照してください。

目的

AUDIT文を使用すると、次の操作を実行できます。

- 後続のユーザー・セッションでのSQL文の発行の監査。特定のSQL文、または特定のシステム権限によって許可されたすべてのSQL文の発行を監査できます。SQL文操作の監査は、後続セッションにのみ適用され、現行のセッションには適用されません。
- 特定のスキーマ・オブジェクトに対する操作の監査。スキーマ・オブジェクト操作の監査は、後続のセッションと同様に、現行のセッションにも適用されます。

関連項目:

- 値に基づいた監査方針を作成および管理するDBMS_FGAパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- NOAUDIT (従来型監査)

前提条件

SQL文の発行を監査するには、AUDIT SYSTEMシステム権限が必要です。ただし、IN SESSION CURRENT句を使用する場合はAUDIT SYSTEMシステム権限は必要ありません。

監査結果を収集するには、初期化パラメータAUDIT_TRAILをデフォルトの設定であるNONE以外の値に設定して、監査を有効にする必要があります。監査オプションは、監査が使用可能であるかどうかにかかわらず指定できます。ただし、監査を有効にしないと、監査レコードは作成されません。

スキーマ・オブジェクト操作を監査するためには、監査対象のオブジェクトが自分のスキーマにある必要があります。自分のスキーマにない場合は、AUDIT ANYシステム権限が必要です。また、監査の対象とするオブジェクトがディレクトリ・オブジェクトの場合は、それが自分で作成したものであっても、AUDIT ANYシステム権限が必要です。

マルチテナント・コンテナ・データベース(CDB)に接続している場合、この項で説明されている権限(現在のコンテナでローカルに付与されているか共通に付与されているいずれかの権限)が必要です。

CONTAINER句を指定する場合は、マルチテナント・コンテナ・データベース(CDB)に接続している必要があります。
CONTAINER = CURRENTを指定するには、現在のコンテナがプラガブル・データベース(PDB)である必要があります。
CONTAINER = ALLを指定する場合は、現在のコンテナがルートである必要があります。

CDBでのAUDIT文の使用についてのノート

CDBでAUDIT文を発行すると、データベースは次のように監査を実行します。

- 現在のコンテナがPDBの場合にAUDIT文を発行すると、データベースはそのPDBで監査を実行します。auditing_by_clauseを指定する場合、userは、PDBのローカル・ユーザーまたは共通ユーザーにする必要があります。audit_schema_object_clauseを指定する場合、オブジェクトはPDBのローカル・オブジェクトにする必要があります。
- 現在のコンテナがルートの場合にAUDIT文を発行すると、データベースはCDB全体つまりルートおよびすべてのPDBで監査を実行します。auditing_by_clauseを指定する場合、userは、共通ユーザーにする必要があります。auditing_by_clauseを省略すると、すべての共通ユーザーが監査されます。audit_schema_object_clauseを指定する場合、オブジェクトはルートまたは共通オブジェクトのローカル・オブジェクトにする必要があります。

ノート:

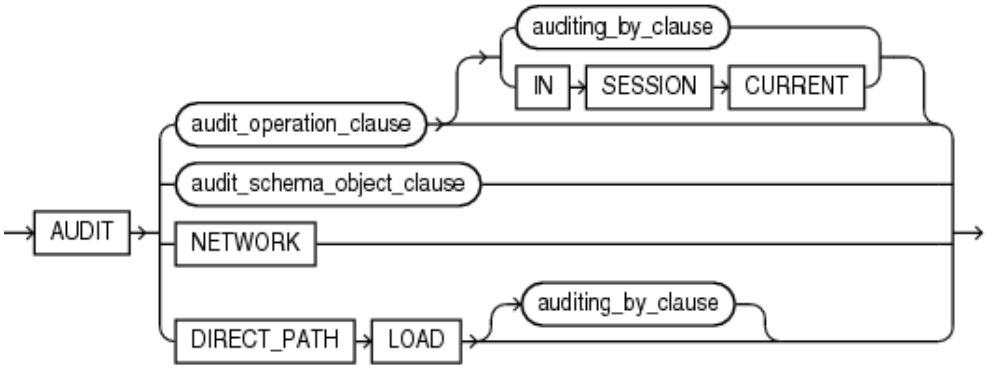
AUDIT ANY システム権限によって、権限受領者は、SYS スキーマを除く任意のスキーマ内の任意のオブジェクトを監査できます。

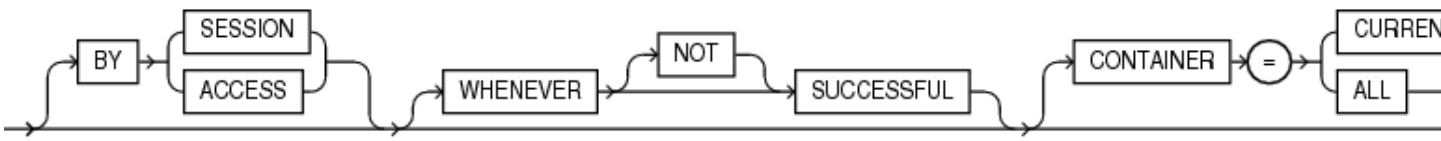
関連項目:

AUDIT_TRAILパラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

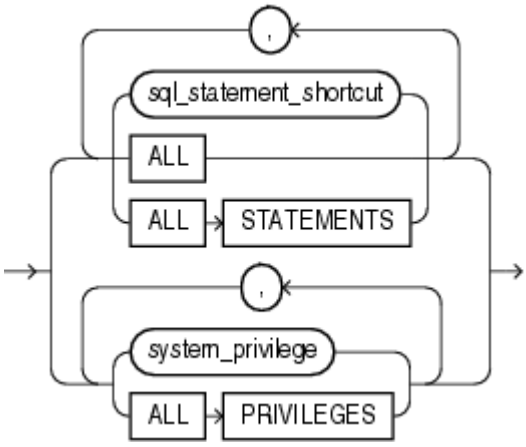
構文

audit::=

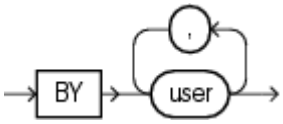




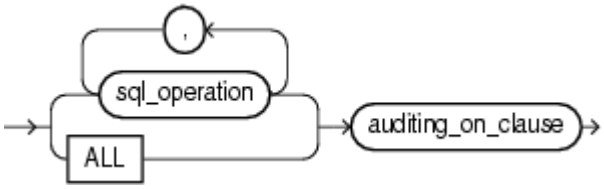
audit_operation_clause::=



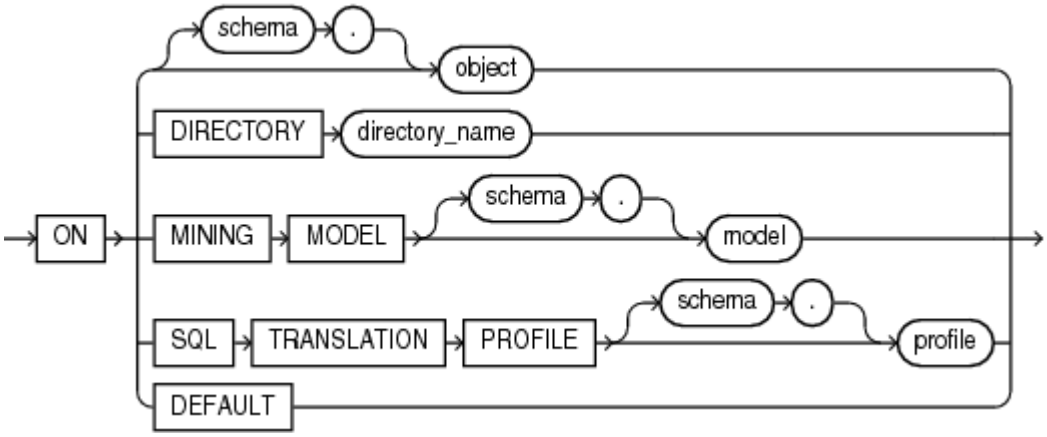
auditing_by_clause::=



audit_schema_object_clause::=



auditing_on_clause::=



セマンティクス

audit_operation_clause

`audit_operation_clause`を使用すると、操作の影響を受けるスキーマ・オブジェクトに関係なく、指定された操作を監査できます。

sql_statement_shortcut

ショートカットを指定すると、特定のSQL文の使用を監査できます。表12-1および表12-2に、ショートカットとその監査対象となるSQL文を示します。

ノート:

SQL 文ショートカットとシステム権限を混同しないでください。たとえば:

- AUDIT USER 文は、SQL の CREATE USER、ALTER USER および DROP USER 文のすべてを監査する USER ショートカットを指定します。この場合の監査には、ユーザーが ALTER USER 文で自分のパスワードを変更する操作が含まれています。

- AUDIT ALTER USER 文では、ALTER USER システム権限を指定して、そのシステム権限を使用するすべての操作を監査できるようにします。この場合の監査には、ユーザーが自分のパスワードを変更する操作は含まれません。これは、その操作には、ALTER USER システム権限は必要ないためです。

監査されるたびに、次の情報を持つ監査レコードが生成されます。

- 操作を行ったユーザー
- 操作の種類
- 操作に関連するオブジェクト
- 操作の日付と時刻

監査レコードは、監査証跡に書き込まれます。監査証跡とは、監査レコードが入っているデータベースの表です。データ・ディクショナリ・ビューを問い合せて監査証跡を調べることによって、データベース・アクティビティを再検討できます。

関連項目:

- 監査証跡データ・ディクショナリ・ビューのリストの詳細は、『Oracle Databaseセキュリティ・ガイド11gリリース2(11.2)』を参照してください。Oracle Database 11g Release 2 (11.2)のドキュメントを見つける方法は、『Oracle Databaseアップグレード・ガイド』を参照してください。
- データ・ディクショナリ・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。
- 「ロールに関連するSQL文の監査: 例」を参照してください。

system_privilege

システム権限を指定すると、そのシステム権限で許可されているSQL文およびその他の操作を監査できます。

ノート:

ANY キーワードを含むシステム権限の使用の監査は、ANY キーワードを含まない同じ権限の使用の監査より限定的です。たとえば:

- AUDIT CREATE PROCEDURE は、CREATE PROCEDURE または CREATE ANY PROCEDURE 権限を使用して発行された文を監査します。

- AUDIT CREATE ANY PROCEDURE は、CREATE ANY PROCEDURE 権限を使用して発行された文のみを監査します。

多くの個々のシステム権限を指定するのではなく、ロールCONNECT、RESOURCEおよびDBAを指定できます。これは、そのロールに付与されているすべてのシステム権限を監査することと同じです。

Oracle Databaseには、システム権限と文オプションをまとめて指定するための、次の3つのショートカットが用意されています。

ALL

ALLを指定すると、表12-1のすべての文オプションを監査できます。ただし、表12-2の追加文オプションは監査しません。

ALL STATEMENTS

ALL STATEMENTSを指定すると、実行されるすべてのトップレベルSQL文を監査できます。トップレベルSQL文は、ユーザーによって直接発行される文です。PL/SQLプロシージャ内またはファンクション内から実行されるSQL文は、トップレベルの文とはみなされません。そのため、この句では、PL/SQLプロシージャ内またはファンクション内から実行される文は監査されません。ただし、PL/SQLプロシージャまたはファンクション自体の実行は監査されます。この句は、システム全体またはユーザー固有の他の監査構成に関係なく、特定の環境内の文をすべて監査する場合に有効です。

ALL PRIVILEGES

ALL PRIVILEGESを指定すると、システム権限を監査できます。

ノート:

ロールまたはショートカットでなく、監査に個々のシステム権限および文オプションを指定することをお薦めします。ロールおよびショートカットに含まれるシステム権限および文オプションは、リリースごとに異なり、Oracle Database の今後のバージョンでサポートされない場合があります。

関連項目:

- すべてのシステム権限とそれによって許可される操作のリストは、表18-1を参照してください。
- CONNECT、RESOURCEおよびDBAロールの詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。
- 「問合せおよび更新を行うSQL文の監査: 例」、「削除の監査: 例」および「ディレクトリに関連する文の監査: 例」を参照してください。

auditing_by_clause

auditing_by_clauseを指定すると、特定のユーザーによって発行されたSQL文のみを監査するように制限できます。この句を指定しない場合、すべてのユーザー文が監査されます。

IN SESSION CURRENT

この句を使用すると、監査を現行のセッションに制限できます。監査はセッションが終了するまで続行し、NOAUDIT文を使用して停止することはできません。

audit_schema_object_clause

audit_schema_object_clauseを使用すると、特定のスキーマ・オブジェクトの操作を監査できます。

audit_schema_object_clauseの制限事項

CDBに接続する場合、audit_schema_object_clauseを指定できますが、CONTAINER句は指定できません。この制限事項は、CONTAINER句で使用できる値がデフォルト値のみのため、機能を制限することはありません。詳細は、「CONTAINER句」を参照してください。

sql_operation

監査するSQL操作を指定します。表12-3に、監査可能なオブジェクトのタイプおよびオブジェクトごとの監査可能なSQL文を示します。たとえば、ALTER操作を指定して表の監査を選択した場合、その表に対して発行されるALTER TABLE文がすべて監査されます。また、SELECT操作を指定して順序の監査を選択した場合、その順序の値を使用するすべての文が監査されます。

ALL

ALLをショートカットに指定することは、オブジェクト・タイプに適用できるSQL操作をすべて指定することと同じです。

auditing_on_clause

auditing_on_clauseを使用すると、監査する特定のスキーマ・オブジェクトを指定できます。

関連項目:

「表の問合せの監査: 例」、「表の挿入および更新の監査: 例」および「順序の操作の監査: 例」を参照してください。

schema

監査の対象として選択したオブジェクトが含まれているスキーマを指定します。schemaを指定しない場合、オブジェクトは自分のスキーマ内にあるとみなされます。

object

監査するオブジェクトの名前を指定します。オブジェクトは、表、ビュー、順序、ストアド・プロシージャ、ストアド・ファンクション、ストアド・パッケージ、マテリアライズド・ビュー、マイニング・モデルまたはライブラリのいずれかである必要があります。

表、ビュー、順序、プロシージャ、ストアド・ファンクション、パッケージ、マテリアライズド・ビューまたはユーザー定義型については、それぞれシノニムも指定できます。

ON DEFAULT

ON DEFAULTを指定すると、指定したオブジェクト・オプションが、それ以降に作成するオブジェクトのデフォルト・オブジェクト・オプションになります。デフォルトの監査オプションが確立されると、その後作成されるオブジェクトに対して、これらのオプションが自動的に適用され、監査が行われます。ビューに対するデフォルト監査オプションは、常に、そのビューの実表に対する監査オプションの論理和となります。ALL_DEF_AUDIT_OPTSデータ・ディクショナリ・ビューを問い合せることによって、現在のデフォルト監査オプションを表示できます。

デフォルト監査オプションを変更した場合でも、以前作成したオブジェクトの監査オプションはそのまま残ります。AUDIT文のON句にオブジェクトを指定した場合のみ、既存のオブジェクトの監査オプションを変更できます。

関連項目:

デフォルト監査オプションの設定: 例

ON DIRECTORY

ON DIRECTORY句を使用すると、監査するディレクトリ名を指定できます。

ON MINING MODEL

ON MINING MODEL句を使用すると、監査するマイニング・モデル名を指定できます。

ON SQL TRANSLATION PROFILE

ON SQL TRANSLATION PROFILE句を使用すると、監査対象にするSQL翻訳プロファイルを指定できます。

NETWORK

この句を使用すると、ネットワーク・レイヤーの内部的な障害を検出できます。

関連項目:

ネットワーク監査の詳細は、『Oracle Databaseセキュリティ・ガイド11gリリース2(11.2)』を参照してください。Oracle Database 11g Release 2 (11.2)のドキュメントを見つける方法は、『Oracle Databaseアップグレード・ガイド』を参照してください。

DIRECT_PATH LOAD

この句を使用すると、SQL*Loaderのダイレクト・パス・ロードを監査できます。

BY SESSION

以前のリリースでは、BY SESSIONを指定すると、データベースによって、同一セッションの同一スキーマ・オブジェクトで実行された同じ種類のすべてのSQL文または操作に対して1つのレコードが書き込まれました。このリリースのOracle Databaseからは、BY SESSIONおよびBY ACCESSを指定すると、監査された文および操作ごとに1つの監査レコードが書き込まれます。BY ACCESSと比較すると、BY SESSIONでは引き続き異なる値が監査証跡に移入されます。すべてのAUDIT文にBY ACCESS句を指定し、より詳細な監査レコードを書き込むことをお薦めします。どちらの句も指定しない場合は、BY ACCESSがデフォルトになります。

ノート:

この句は、データ定義言語(DDL)文以外の SQL 文を監査するスキーマ・オブジェクト監査オプション、文オプションおよびシステム権限にのみ適用されます。DDL 文を監査するすべての BY ACCESS SQL 文およびシステム権限は、データベースによって常に監査されます。

BY ACCESS

BY ACCESSを指定すると、監査された各文および操作について、1つのレコードを書き込むことができます。

ノート:

データ定義言語(DDL)文を監査する SQL 文ショートカットまたはシステム権限を指定した場合は、常にアクセスごとに監査されます。それ以外のすべての場合、データベースは BY SESSION または BY ACCESS の指定に従います。

DDL文以外のSQL文を監査する文オプションとシステム権限には、BY SESSIONまたはBY ACCESSのいずれかを指定できま

す。BY ACCESSがデフォルトです。

WHENEVER [NOT] SUCCESSFUL

WHENEVER SUCCESSFULを指定すると、正常に実行されたSQL文および操作のみを監査できます。

WHENEVER NOT SUCCESSFULを指定すると、失敗またはエラーが発生したSQL文および操作のみを監査できます。

この句を指定しない場合、処理結果にかかわらず監査が行われます。

CONTAINER句

CONTAINER句は、CDBに接続している場合にのみ適用されます。この句を使用すると、AUDIT文の有効範囲を指定できます。ただし、デフォルト値が使用できる唯一の値であるため、CONTAINER句を指定する必要はありません。

- 現在のコンテナがPDBであるときにAUDIT文を発行する場合は、オプションで、デフォルトのCONTAINER = CURRENTを指定できます。
- 現在のコンテナがルートであるときにAUDIT文を発行する場合は、オプションで、デフォルトのCONTAINER = ALLを指定できます。

監査オプションの表

表12-1 監査のSQL文ショートカット

SQL文ショートカット	監査済のSQL文と操作
ALTER SYSTEM	ALTER SYSTEM
CLUSTER	CREATE CLUSTER ALTER CLUSTER DROP CLUSTER TRUNCATE CLUSTER
CONTEXT	CREATE CONTEXT DROP CONTEXT
DATABASE LINK	CREATE DATABASE LINK ALTER DATABASE LINK DROP DATABASE LINK
DIMENSION	CREATE DIMENSION ALTER DIMENSION DROP DIMENSION
DIRECTORY	CREATE DIRECTORY DROP DIRECTORY

SQL文ショートカット	監査済のSQL文と操作
INDEX	CREATE INDEX ALTER INDEX ANALYZE INDEX DROP INDEX
MATERIALIZED VIEW	CREATE MATERIALIZED VIEW ALTER MATERIALIZED VIEW DROP MATERIALIZED VIEW
NOT EXISTS	指定したオブジェクトが存在しない場合に失敗するすべての SQL 文。
OUTLINE	CREATE OUTLINE ALTER OUTLINE DROP OUTLINE
PLUGGABLE DATABASE	CREATE PLUGGABLE DATABASE ALTER PLUGGABLE DATABASE DROP PLUGGABLE DATABASE
PROCEDURE(表の後の「ノート」を参照)	CREATE FUNCTION CREATE LIBRARY CREATE PACKAGE CREATE PACKAGE BODY CREATE PROCEDURE DROP FUNCTION DROP LIBRARY DROP PACKAGE DROP PROCEDURE
PROFILE	CREATE PROFILE ALTER PROFILE DROP PROFILE

SQL文ショートカット	監査済のSQL文と操作
PUBLIC DATABASE LINK	CREATE PUBLIC DATABASE LINK ALTER PUBLIC DATABASE LINK DROP PUBLIC DATABASE LINK
PUBLIC SYNONYM	CREATE PUBLIC SYNONYM DROP PUBLIC SYNONYM
ROLE	CREATE ROLE ALTER ROLE DROP ROLE SET ROLE
ROLLBACK SEGMENT	CREATE ROLLBACK SEGMENT ALTER ROLLBACK SEGMENT DROP ROLLBACK SEGMENT
SEQUENCE	CREATE SEQUENCE DROP SEQUENCE
SESSION	ログオン
SYNONYM	CREATE SYNONYM DROP SYNONYM
SYSTEM AUDIT	AUDIT sql_statements NOAUDIT sql_statements
SYSTEM GRANT	GRANT system_privileges_and_roles REVOKE system_privileges_and_roles
TABLE	CREATE TABLE DROP TABLE TRUNCATE TABLE
TABLESPACE	CREATE TABLESPACE

SQL文ショートカット	監査済のSQL文と操作
	ALTER TABLESPACE DROP TABLESPACE
TRIGGER	CREATE TRIGGER ALTER TRIGGER ENABLE および DISABLE 句付き DROP TRIGGER ALTER TABLE ENABLE ALL TRIGGERS 句付き および DISABLE ALL TRIGGERS 句付き
TYPE	CREATE TYPE CREATE TYPE BODY ALTER TYPE DROP TYPE DROP TYPE BODY
USER	CREATE USER ALTER USER DROP USER ノート: ● AUDIT USER は、これら 3 つの SQL 文を監査します。AUDIT ALTER USER を使用すると、ALTER USER システム権限を必要とする文を監査できます。 ● AUDIT ALTER USER 文は、自分のパスワードを変更するユーザーを監査しません。これは、このアクティビティには ALTER USER システム権限は必要ではないためです。
VIEW	CREATE VIEW DROP VIEW

ノート:

Java スキーマ・オブジェクト(ソース、クラスおよびリソース)は、SQL 文の監査ではプロシージャと同じであるとみなされます。

表12-2 監査のその他のSQL文ショートカット

SQL文ショートカット	監査済のSQL文と操作
ALTER SEQUENCE	ALTER SEQUENCE
ALTER TABLE	ALTER TABLE
COMMENT TABLE	COMMENT ON TABLE table, view, materialized view COMMENT ON COLUMN table.column, view.column, materialized view.column
DELETE TABLE	DELETE FROM table, view
EXECUTE DIRECTORY	ディレクトリでのプログラムの実行
EXECUTE PROCEDURE	CALL プロシージャやファンクションの実行、またはパッケージ内の変数、ライブラリまたはカーソルへのアクセス。
GRANT DIRECTORY	GRANT privilege ON directory REVOKE privilege ON directory
GRANT PROCEDURE	GRANT privilege ON procedure, function, package REVOKE privilege ON procedure, function, package
GRANT SEQUENCE	GRANT privilege ON sequence REVOKE privilege ON sequence
GRANT TABLE	GRANT privilege ON table, view, materialized view REVOKE privilege ON table, view, materialized view

SQL文ショートカット	監査済のSQL文と操作
GRANT TYPE	GRANT privilege ON TYPE REVOKE privilege ON TYPE
INSERT TABLE	INSERT INTO table, view
LOCK TABLE	LOCK TABLE table, view
READ DIRECTORY	ディレクトリでの読取り操作
SELECT SEQUENCE	sequence.CURRVAL または sequence.NEXTVAL を含む文
SELECT TABLE	SELECT FROM table, view, materialized view
UPDATE TABLE	UPDATE table, view
WRITE DIRECTORY	ディレクトリでの書込み操作

表12-3 スキーマ・オブジェクト監査オプション

オブジェクト	SQL操作
表	● ALTER ● AUDIT ● COMMENT ● DELETE ● FLASHBACK (ノート 1) ● GRANT ● INDEX ● INSERT ● LOCK ● RENAME ● SELECT ● UPDATE
ビュー	● AUDIT ● COMMENT ● DELETE ● FLASHBACK (ノート 1) ● GRANT ● INSERT ● LOCK ● RENAME ● SELECT

オブジェクト	SQL操作
	● UPDATE
順序	● ALTER ● AUDIT ● GRANT ● SELECT
プロシージャ、ファンクション、パッケージ(ノート 2)	● AUDIT ● EXECUTE (ノート 3 および 4) ● GRANT
マテリアライズド・ビュー(ノート 5)	● ALTER ● AUDIT ● COMMENT ● DELETE ● INDEX ● INSERT ● LOCK ● SELECT ● UPDATE
マイニング・モデル	● AUDIT ● COMMENT ● GRANT ● RENAME ● SELECT
ディレクトリ	● AUDIT ● GRANT ● READ
ライブラリ	● EXECUTE ● GRANT
オブジェクト型	● ALTER ● AUDIT ● GRANT

ノート1: FLASHBACK監査オブジェクト・オプションは、フラッシュバック問合せに対してのみ適用されます。

ノート2: Javaスキーマ・オブジェクト(ソース、クラスおよびリソース)は、監査オプションではプロシージャ、ファンクションおよびパッケージと同じであるとみなされます。

ノート3: PL/SQLストアド・プロシージャまたはストアド・ファンクションに対するEXECUTE操作を監査する場合は、監査目的での操作の成否を判断する際に、プロシージャまたはファンクションが見つかるか、およびその実行を認証できるかどうかのみが考慮されます。したがって、WHENEVER NOT SUCCESSFUL句を指定すると、無効なオブジェクト・エラー、存在しないオブジェクト・エラー、および認証の失敗が監査されます。プロシージャまたはファンクションの実行時に検出されたエラーは監査されません。WHENEVER SUCCESSFUL句を指定すると、実行時にエラーが検出されたかどうかに関係なく、無効なオブジェクト・エラー、存

在しないオブジェクト・エラー、および認証の失敗が監査されます。

ノート4: PL/SQLストアド・プロシージャまたはストアド・ファンクション内の再帰的なSQL操作の失敗を監査するには、SQL操作の監査を構成します。

ノート5: INSERT、UPDATEおよびDELETE操作の監査は、更新可能なマテリアライズド・ビューに対してのみ可能です。

例

ロールに関連するSQL文の監査: 例

次の文は、ロールの作成、変更、削除または設定を行う各SQL文が正常に終了したかどうかにかかわらず、それらの文について監査を行います。

```
AUDIT ROLE;
```

次の文は、ロールの作成、変更、削除または設定を行う、正常に終了したSQL文ごとに監査を行います。

```
AUDIT ROLE
    WHENEVER SUCCESSFUL;
```

次の文は、Oracle Databaseエラーが発生したCREATE ROLE文、ALTER ROLE文、DROP ROLE文またはSET ROLE文について監査を行います。

```
AUDIT ROLE
    WHENEVER NOT SUCCESSFUL;
```

問合せおよび更新を行うSQL文の監査: 例

次の文は、任意の表について問合せまたは更新を行う任意の文について監査を行います。

```
AUDIT SELECT TABLE, UPDATE TABLE;
```

次の文は、ユーザーhrおよびoeが発行する、表やビューの問合せまたは更新を実行する文について監査を行います。

```
AUDIT SELECT TABLE, UPDATE TABLE
    BY hr, oe;
```

削除の監査: 例

次の文は、DELETE ANY TABLEシステム権限を使用して発行された文について監査を行います。

```
AUDIT DELETE ANY TABLE;
```

ディレクトリに関連する文の監査: 例

次の文は、CREATE ANY DIRECTORYシステム権限を使用して発行された文について監査を行います。

```
AUDIT CREATE ANY DIRECTORY;
```

CREATE ANY DIRECTORYシステム権限を使用しないCREATE DIRECTORY(およびDROP DIRECTORY)文を監査する場合は、次の文を発行します。

```
AUDIT DIRECTORY;
```

次の文は、bfile_dirディレクトリからファイルを読み込む各文について監査を行います。

```
AUDIT READ ON DIRECTORY bfile_dir;
```

次の文は、任意のディレクトリからファイルを読み込む各文について監査を行います。

```
AUDIT READ DIRECTORY;
```

表の問合せの監査: 例

次の文は、スキーマhr内のemployees表を問い合せる各SQL文について監査を行います。

```
AUDIT SELECT
    ON hr.employees;
```

次の文は、スキーマhr内のemployees表を問い合せて正常に終了した各文について監査を行います。

```
AUDIT SELECT
    ON hr.employees
    WHENEVER SUCCESSFUL;
```

次の文は、スキーマhr内のemployees表を問い合せてOracle Databaseエラーが発生した各文について監査を行います。

```
AUDIT SELECT
    ON hr.employees
    WHENEVER NOT SUCCESSFUL;
```

表の挿入および更新の監査: 例

次の文は、スキーマoe内のcustomers表内の行を挿入または更新する各文について監査を行います。

```
AUDIT INSERT, UPDATE
    ON oe.customers;
```

順序の操作の監査: 例

次の文は、スキーマhr内のemployees_seq順序に対する操作を行う各文について監査を行います。

```
AUDIT ALL
    ON hr.employees_seq;
```

この文では、順序に対して操作を行う次の文について監査を行うため、ALLショートカットを使用しています。

- ALTER SEQUENCE
- AUDIT
- GRANT
- 疑似列CURRVALまたはNEXTVALを使用して、順序の値にアクセスするすべての文

デフォルト監査オプションの設定: 例

次の文は、将来作成されるオブジェクトについてデフォルト監査オプションを指定します。

```
AUDIT ALTER, GRANT, INSERT, UPDATE, DELETE
    ON DEFAULT;
```

以降に作成されるオブジェクトは、指定したオプションによって自動的に監査用に構成されます。

- 表を作成した場合、その表に対して発行されるALTER文、GRANT文、INSERT文、UPDATE文またはDELETE文の監査オプションが自動的に構成されます。
- ビューを作成した場合、そのビューに対するGRANT文、INSERT文、UPDATE文またはDELETE文の監査オプションが自動的に構成されます。
- 順序を作成した場合、その順序に対するALTER文またはGRANT文の監査オプションが自動的に構成されます。

- プロシージャ、パッケージまたはファンクションを作成した場合、それらに対するALTER文またはGRANT文の監査オプションが自動的に構成されます。

AUDIT (統合監査)

この項では、統合監査のAUDIT文について説明します。この種類の監査は、Oracle Database 12cで新たに導入されたもので、完全かつ高度な監査機能を提供します。統合監査の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

目的

AUDIT文は、次の目的に使用します。

- すべてのユーザーまたは特定のユーザーに対する統合監査ポリシーを有効にする。
- 監査対象イベントが失敗または成功(または、その両方)した場合に、監査レコードを作成するかどうかを指定する。
- アプリケーション・コンテキスト属性を指定する(属性値が監査レコードに記録されます)。

この文で実行される操作は、現行のセッションではなく、それ以降のユーザー・セッションで有効になります。

関連項目:

- NOAUDIT (統合監査)
- CREATE AUDIT POLICY (統合監査)
- ALTER AUDIT POLICY (統合監査)
- DROP AUDIT POLICY (統合監査)

前提条件

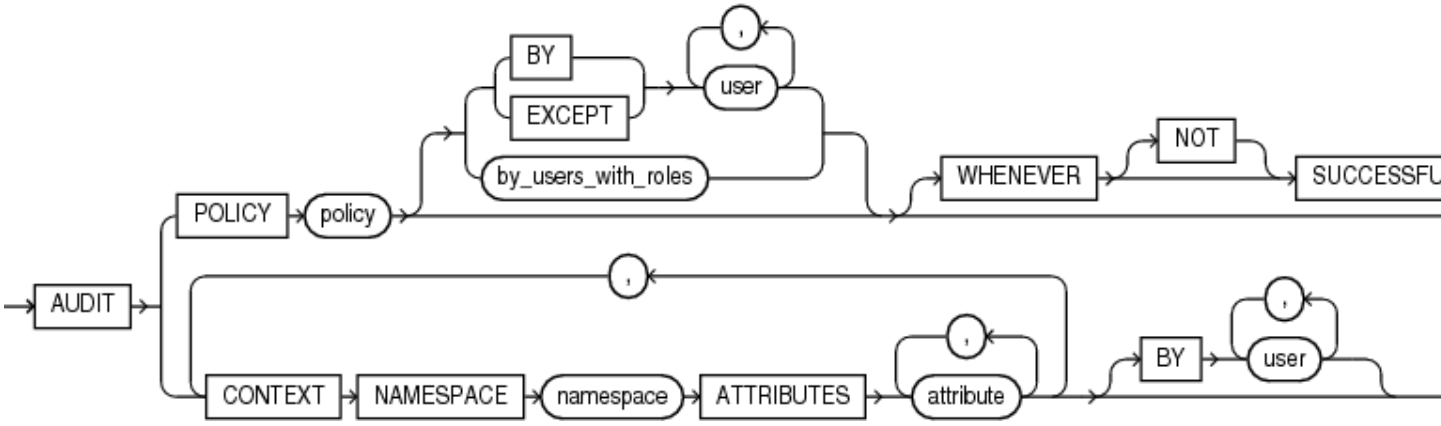
AUDIT SYSTEMシステム権限、またはAUDIT_ADMINロールが必要になります。

マルチテナント・コンテナ・データベース(CDB)に接続している場合、共通の統合監査ポリシーを有効にするには、現在のコンテナがルートである必要があります。また、共通に付与されているAUDIT SYSTEM権限またはAUDIT_ADMIN共通ロールが必要です。ローカルの統合監査ポリシーを有効にするには、現在のコンテナが、その監査ポリシーが作成されたコンテナである必要があります。また、共通に付与されているAUDIT SYSTEM権限またはAUDIT_ADMIN共通ロールを保有しているか、そのコンテナでローカルに付与されているAUDIT SYSTEM権限またはAUDIT_ADMINローカル・ロールを保有している必要があります。

CDBに接続しているときにAUDIT CONTEXT ...文を指定するには、共通に付与されているAUDIT SYSTEM権限またはAUDIT_ADMIN共通ロールを保有しているか、現在のセッションのコンテナでローカルに付与されているAUDIT SYSTEM権限またはAUDIT_ADMINローカル・ロールを保有している必要があります。

構文

unified_audit::=



by_users_with_roles::=

セマンティクス

policy

有効にする統合監査ポリシーの名前を指定します。このポリシーは、CREATE AUDIT POLICY文を使用して作成されている必要があります。

すべての統合監査ポリシーの説明を検索するには、AUDIT_UNIFIED_POLICIESビューを問い合わせます。すべての有効な統合監査ポリシーの説明を検索するには、AUDIT_UNIFIED_ENABLED_POLICIESビューを問い合わせます。

統合監査ポリシーを有効にすると、その有効にしたポリシーで指定されているシステム権限、アクションまたはロールの監査オプションに合致するすべてのSQL文および操作が監査されます。つまり、統合監査レコードがUNIFIED_AUDIT_TRAILビューに作成されます。1つのSQL文または操作が複数の有効なポリシーに合致する場合、統合監査レコードは1つのみ作成され、合致するすべての監査ポリシーの名前がUNIFIED_AUDIT_TRAILビューのUNIFIED_AUDIT_POLICIES列にカンマ区切りリストで表示されます。

関連項目:

- CREATE AUDIT POLICY (統合監査)
- AUDIT_UNIFIED_POLICIESビュー、AUDIT_UNIFIED_ENABLED_POLICIESビューおよびUNIFIED_AUDIT_TRAILビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

BY | EXCEPT

BY句を指定すると、指定したユーザーに対してのみpolicyを有効にできます。

EXCEPT句を指定すると、指定したユーザーを除くすべてのユーザーに対してpolicyを有効にできます。

BY、EXCEPTおよびby_users_with_roles句を指定しない場合、policyはすべてのユーザーを対象として有効化されます。

policyが共通の統合監査ポリシーの場合、userは共通ユーザーにする必要があります。policyがローカルの統合監査ポリシーの場合、userは共通ユーザーまたは接続先のコンテナのローカル・ユーザーにする必要があります。

BY句およびEXCEPT句のノート

BY句およびEXCEPT句には、次のノートが適用されます。

- 同じ統合監査ポリシーに対して複数のAUDIT ... BY ... 文を指定した場合、そのポリシーは各文で指定したユーザーの和集合に対して有効になります。
- 同じ統合監査ポリシーに対して複数のAUDIT ... EXCEPT ... 文を指定した場合、最後に指定した文のみが作用します。つまり、最後のAUDIT ... EXCEPT ... 文で指定したユーザーを除くすべてのユーザーに対してポリシーが有効になります。
- BY句を使用して有効化されたポリシーを、EXCEPT句を使用して有効にする場合、まずNOAUDIT ... BY ... 文を使用して、そのポリシーが現在有効になっているすべてのユーザーに対してポリシーを無効にしてから、AUDIT ... EXCEPT ... 文でポリシーを有効にする必要があります。
- EXCEPT句を使用して有効化されてたポリシーを、BY句を使用して有効にする場合、まずNOAUDIT文を使用してその監査ポリシーを無効にする必要があります。EXCEPT句はNOAUDIT文では指定できません。次に、AUDIT ... BY ... 文でポリシーを有効にします。

BY句およびEXCEPT句の制限事項

同じ統合監査ポリシーに、AUDIT ... BY ...文およびAUDIT ... EXCEPT ...文を指定することはできません。指定すると、エラーが発生します。

by_users_with_roles

この句を指定すると、指定したロールを直接的または間接的に付与されたユーザーに対してpolicyを有効化できます。その後、いずれかのロールを追加のユーザー、またはユーザーに直接的または間接的に付与されたロールに付与すると、ポリシーはそのユーザーに自動的に適用されます。その後、いずれかのロールをユーザーまたはロール(ロールまたはユーザーに直接的または間接的に付与されたロール)から取り消した場合、ポリシーはそのユーザーに適用されなくなります。

CDBに接続されているときに、policyが共通の統合監査ポリシーである場合、roleは共通ロールである必要があります。policyがローカルの統合監査ポリシーの場合、roleは共通ロールまたは接続先のコンテナ内のローカル・ロールにする必要があります。

ロールに対するローカル監査ポリシーの有効化

ローカル監査ポリシーは、ローカル・ロールおよび共通ロールに対して有効にできます。ローカル監査ポリシーが共通ロールに対して有効になっている場合、共通ロールがユーザーにコンテナ内でローカルまたは共通に付与されると、監査レコードが生成されます。

ロールに対する共通監査ポリシーの有効化

共通監査ポリシーは、共通ロールに対してのみ有効化できます。共通監査ポリシーが共通ロールに対して有効になっている場合、共通ロールがユーザーにルート・コンテナ内で共通またはローカルに付与されると、監査レコードが生成されます。

WHENEVER [NOT] SUCCESSFUL

WHENEVER SUCCESSFULを指定すると、正常に実行されたSQL文および操作のみを監査できます。

WHENEVER NOT SUCCESSFULを指定すると、失敗またはエラーが発生したSQL文および操作のみを監査できます。

この句を指定しない場合、処理結果にかかわらず監査が行われます。

CONTEXT句

CONTEXT句を指定すると、監査レコードにコンテキスト属性の値を含めることができます。

- namespaceには、コンテキスト・ネームスペースを指定します。
- attributeには、監査レコードに含める値を持つ1つ以上のコンテキスト属性を指定します。
- オプションのBY user句を使用すると、指定したユーザーが実行したイベントについての監査レコードにのみ、コンテキスト属性の値が含まれるようになります。BY句を省略すると、コンテキスト属性の値がすべての監査レコードに含まれるようになります。

現在のコンテナがCDBのルートであるときにCONTEXT句を指定すると、ルートで実行されたイベントについての監査レコードにのみ、コンテキスト属性の値が含まれるようになります。オプションのBY句を指定する場合、userは共通ユーザーにする必要があります。

現在のコンテナがプラガブル・データベース(PDB)であるときにCONTEXT句を指定すると、PDBで実行されたイベントについての監査レコードにのみ、コンテキスト属性の値が含まれるようになります。オプションのBY句を指定する場合、userは、共通ユーザーまたはそのPDBのローカル・ユーザーにする必要があります。

監査証跡に取得されるように構成されている、アプリケーション・コンテキストの属性を調べるには、AUDIT_UNIFIED_CONTEXTSビューを問い合わます。

関連項目:

AUDIT_UNIFIED_CONTEXTSビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

例

次の例では、CREATE AUDIT POLICYの「例」で作成した統合監査ポリシーを有効にします。

すべてのユーザーを対象とした統合監査ポリシーの有効化: 例

次の文は、統合監査ポリシーtable_polをすべてのユーザーに対して有効にします。

```
AUDIT POLICY table_pol;
```

次の文は、table_polがすべてのユーザーに対して有効になっていることを確認します。

```
SELECT policy_name, enabled_option, entity_name
  FROM audit_unified_enabled_policies
  WHERE policy_name = 'TABLE_POL';

POLICY_NAME  ENABLED_OPTION  ENTITY_NAME
-----------  -----------  ---------
TABLE_POL    BY           ALL USERS
```

特定のユーザーを対象とした統合監査ポリシーの有効化: 例

次の文は、統合監査ポリシーdml_polをユーザーhrおよびshに対してのみ有効にします。

```
AUDIT POLICY dml_pol BY hr, sh;
```

次の文は、dml_polがユーザーhrとshに対してのみ有効になっていることを確認します。

```
SELECT policy_name, enabled_option, entity_name
  FROM audit_unified_enabled_policies
  WHERE policy_name = 'DML_POL'
  ORDER BY user_name;

POLICY_NAME  ENABLED_OPTION  ENTITY_NAME
```

```
-----------  -----------  ---------
DML_POL      BY           HR
DML_POL      BY           SH
```

次の文は、統合監査ポリシーread_dir_polをhr以外のすべてのユーザーに対して有効にします。

```
AUDIT POLICY read_dir_pol EXCEPT hr;
```

次の文は、read_dir_polがhr以外のすべてのユーザーに対して有効になっていることを確認します。

```
SELECT policy_name, enabled_option, entity_name
  FROM audit_unified_enabled_policies
  WHERE policy_name = 'READ_DIR_POL';

POLICY_NAME   ENABLED_OPTION  ENTITY_NAME
------------  -----------  ---------
READ_DIR_POL  EXCEPT       HR
```

次の文は、統合監査ポリシーsecurity_polをユーザーhrに対して有効にして、失敗したSQL文および操作のみを監査します。

```
AUDIT POLICY security_pol BY hr WHENEVER NOT SUCCESSFUL;
```

次の文は、統合監査ポリシーsecurity_polがユーザーhrに対してのみ有効になっており、失敗したSQL文および操作のみが監査対象になることを確認します。

```
SELECT policy_name, enabled_option, entity_name, success, failure
  FROM audit_unified_enabled_policies
  WHERE policy_name = 'SECURITY_POL';

POLICY_NAME   ENABLED_OPTION           ENTITY_NAME   SUCCESS  FAILURE
------------  -------------------  ----------  -------  -------
SECURITY_POL  BY                   HR          NO       YES
```

監査レコードへのコンテキスト属性値の追加: 例

次の文では、ネームスペースUSERENVの属性CURRENT_USERとDB_NAMEの値を、ユーザーhrについてのすべての監査レコードに含めるようにデータベースに指示します。

```
AUDIT CONTEXT NAMESPACE userenv
  ATTRIBUTES current_user, db_name
  BY hr;
```

CALL

目的

CALL文を使用すると、SQL内からルーチン(スタンドアロン・プロシージャ、スタンドアロン・ファンクション、あるいは型またはパッケージ内で定義されたプロシージャまたはファンクション)を実行できます。

ノート:

「ファンクション式」に指定されているユーザー定義ファンクション式の制限事項は、CALL 文にも適用されます。

関連項目:

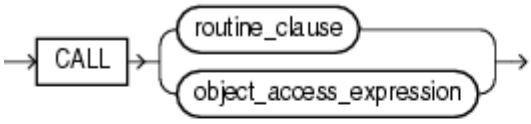
これらのルーチンの作成の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

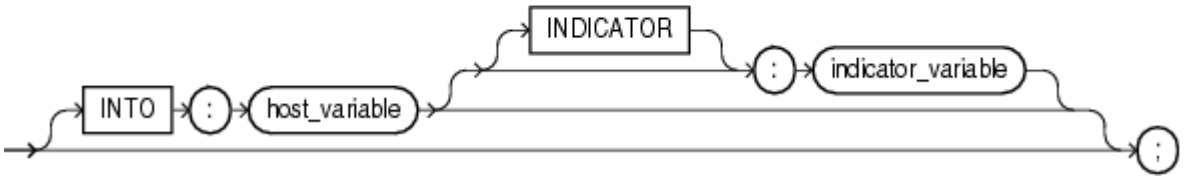
前提条件

スタンドアロンのルーチン上、またはルーチンが定義されている型やパッケージ上で、EXECUTE権限を持っている必要があります。

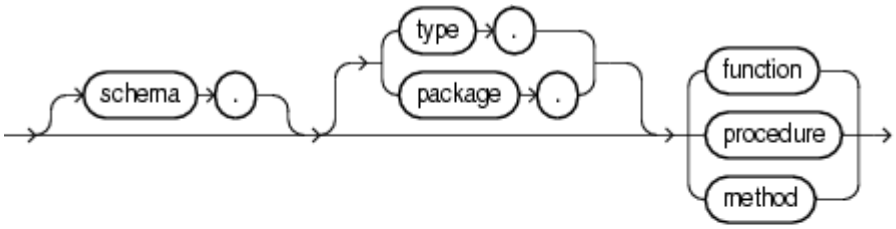
構文

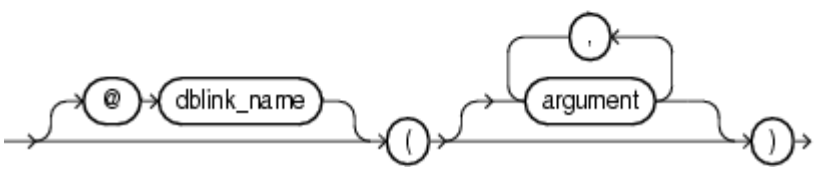
call::=



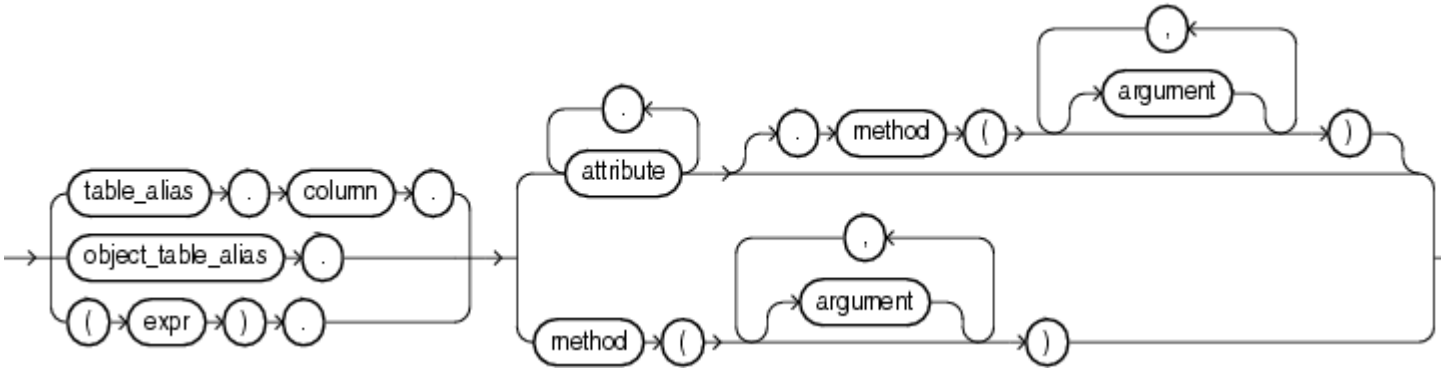


routine_clause::=





object_access_expression::=



セマンティクス

ルーチンは2つの方法で実行できます。routine_clauseを使用してルーチン自体を名前でコールするか、またはobject_access_expressionを使用して式の型の中でルーチンをコールします。

routine_clause

コールするファンクション名またはプロシージャ名、またはファンクションまたはプロシージャに変換されるシノニムを指定します。

型のメンバー・ファンクションまたはメンバー・プロシージャをコールする場合、最初の引数(SELF)がNULLのIN OUTの場合は、エラーが戻されます。SELFがNULLのINの場合には、NULLが戻されます。どちらの場合も、ファンクションまたはプロシージャはコールされません。

ファンクションの制限事項

ルーチンがファンクションの場合、INTO句が必要です。

schema

スタンドアロン・ルーチン(または、ルーチンが含まれている型またはパッケージ)が存在するスキーマを指定します。schemaを指定しない場合、ルーチンは自分のスキーマ内にあるとみなされます。

typeまたはpackage

ルーチンが定義されている型またはパッケージを指定します。

@dblink

分散データベース・システムで、スタンドアロン・ルーチンが含まれているデータベース、またはルーチンが含まれているパッケージまたはファンクションの名前を指定します。dblinkを指定しない場合、ローカル・データベースを指定したとみなされます。

関連項目:

直接ルーチンをコールする例は、「プロシージャのコール: 例」を参照してください。

object_access_expression

型コンストラクタ、バインド変数などのオブジェクト型の式を使用している場合は、この形式の式を使用して、型内で定義されたルーチンをコールできます。このコンテキストでは、object_access_expressionはメソッドのコールに制限されます。

関連項目:

この形式の式の構文およびセマンティクスは、「オブジェクト・アクセス式」を参照してください。オブジェクト型の式を使用してルー

チンをコールする例は、「オブジェクト型の式を使用したプロシージャのコール: 例」を参照してください。

argument

ルーチンに引数が必要な場合に、ルーチンの1つ以上の引数を指定します。argumentを、位置表記法、名前表記法および混合表記法で表記できます。たとえば、次の表記はすべて正しい表記です。

```
CALL my_procedure(arg1 => 3, arg2 => 4)

CALL my_procedure(3, 4)
CALL my_procedure(3, arg2 => 4)
```

ルーチンに対する引数の適用の制限事項

argumentには、次の制限事項があります。

- CALL文によって渡されるパラメータのデータ型は、SQLのデータ型であることが必要です。BOOLEANなど、PL/SQL専用のデータ型は使用できません。
- 引数には、疑似列、オブジェクト参照ファンクションVALUEおよびREFは指定できません。
- ルーチンのIN OUT引数またはOUT引数であるすべての引数は、ホスト変数の式に対応している必要があります。
- すべての戻り引数を含む引数の数は、1000に制限されています。
- 4KBを超える文字データ型またはRAWデータ型(CHAR、VARCHAR2、NCHAR、NVARCHAR2、RAW、LONG RAW)の引数はバインドできません。

INTO :host_variable

INTO句は関数のコールにのみ適用されます。ファンクションの戻り値を格納するホスト変数を指定します。

:indicator_variable

ホスト変数の値または状態を指定します。

関連項目:

ホスト変数および標識変数の詳細は、『Pro*C/C++プログラマーズ・ガイド』を参照してください。

例

プロシージャのコール: 例

次の文は、remove_deptプロシージャを使用して、Entertainment部門(「順序値の挿入: 例」で作成)を削除します。このプロシージャを作成する例については、『Oracle Database PL/SQL言語リファレンス』を参照してください。

```
CALL emp_mgmt.remove_dept(162);
```

オブジェクト型の式を使用したプロシージャのコール: 例

次の例では、CALL文のオブジェクト型の式を使用したプロシージャのコール方法を示します。この例では、サンプルの注文入力スキーマOEでwarehouse_typオブジェクト型を使用しています。

```
ALTER TYPE warehouse_typ
      ADD MEMBER FUNCTION ret_name
      RETURN VARCHAR2
      CASCADE;
CREATE OR REPLACE TYPE BODY warehouse_typ
      AS MEMBER FUNCTION ret_name
```

```
      RETURN VARCHAR2
      IS
         BEGIN
            RETURN self.warehouse_name;
         END;
      END;
/
VARIABLE x VARCHAR2(25);
CALL warehouse_typ(456, 'Warehouse 456', 2236).ret_name()
   INTO :x;
PRINT x;
X
--------------------------------
Warehouse 456
```

次の例では、外部ファンクションを使用したプロシージャのコール方法を示します。

```
CREATE OR REPLACE FUNCTION ret_warehouse_typ(x warehouse_typ)
  RETURN warehouse_typ
  IS
    BEGIN
      RETURN x;
    END;
/
CALL ret_warehouse_typ(warehouse_typ(234, 'Warehouse 234',
   2235)).ret_name()
   INTO :x;
PRINT x;
X
--------------------------------
Warehouse 234
```

COMMENT

目的

COMMENT文を使用すると、表または表の列、統合監査ポリシー、エディション、索引タイプ、マテリアライズド・ビュー、マイニング・モデル、演算子またはビューに関するコメントを、データ・ディクショナリに追加できます。

データベースからコメントを削除する場合、空の文字列' 'を設定します。

関連項目:

- SQL文およびスキーマ・オブジェクトへのコメントの関連付けの詳細は、「コメント」を参照してください。
- コメントを表示するデータ・ディクショナリ・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

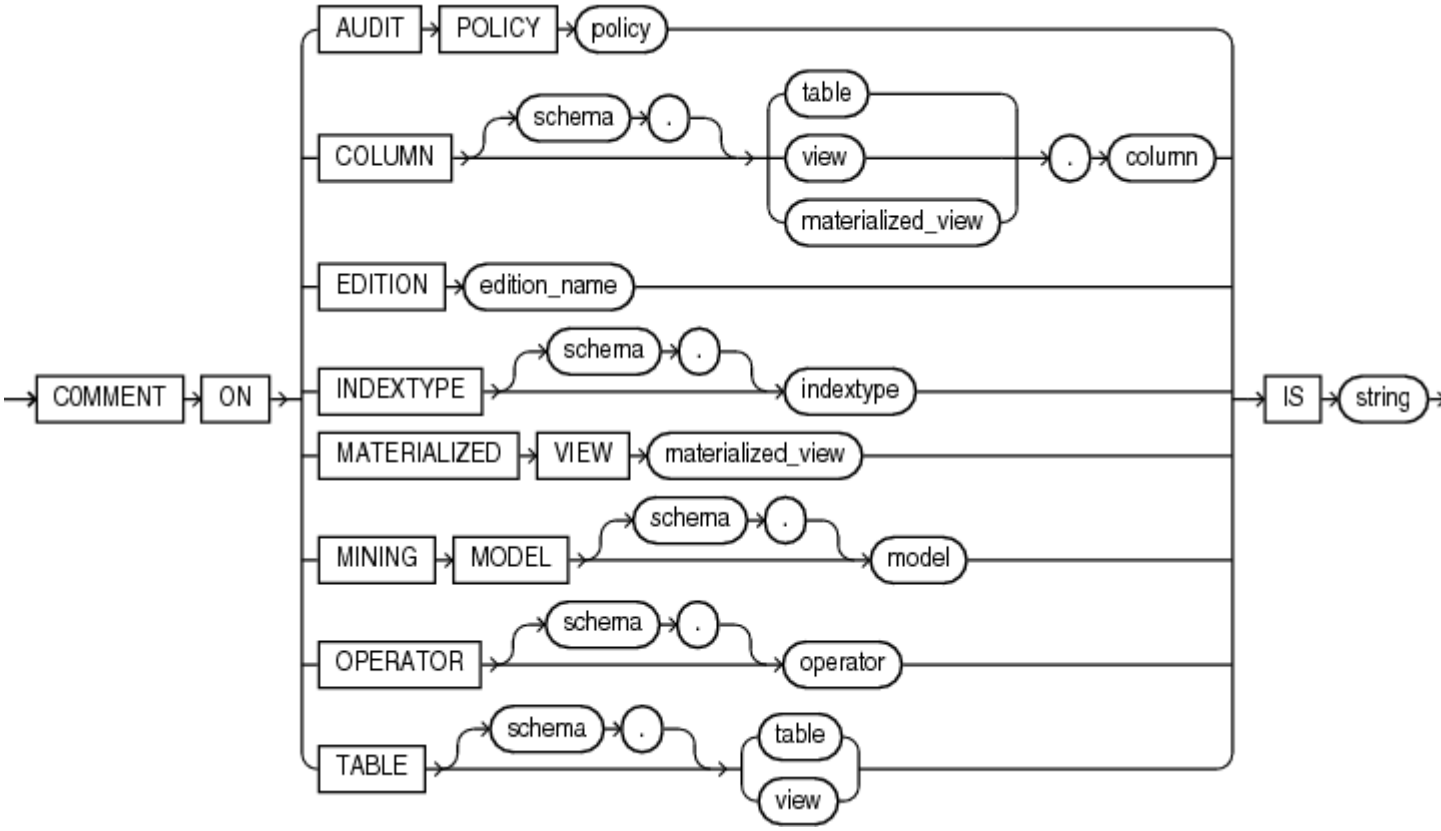
前提条件

コメントを追加するオブジェクトが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、次の条件を満たす必要があります。

- 表、ビューまたはマテリアライズド・ビューにコメントを追加する場合は、COMMENT ANY TABLEシステム権限を持っている。
- 統合監査ポリシーにコメントを追加する場合は、AUDIT SYSTEMシステム権限またはAUDIT_ADMINロールが必要です。
- エディションにコメントを追加する場合は、直接またはロールを介して付与されたCREATE ANY EDITIONシステム権限を持っている。
- 索引タイプにコメントを追加する場合は、CREATE ANY INDEXTYPEシステム権限を持っている。
- マイニング・モデルにコメントを追加する場合は、COMMENT ANY MINING MODELシステム権限が必要です。
- 演算子にコメントを追加する場合は、CREATE ANY OPERATORシステム権限を持っている。

構文

comment::=



セマンティクス

AUDIT POLICY句

コメントする統合監査ポリシーの名前を指定します。

AUDIT_UNIFIED_POLICY_COMMENTSデータ・ディクショナリ・ビューを問い合せると、特定の統合監査ポリシーに対するコメントを表示できます。

COLUMN句

コメントする表、ビューまたはマテリアライズド・ビューの列の名前を指定します。schemaを指定しない場合、この表、ビューおよびマテリアライズド・ビューは自分のスキーマ内にあるとみなされます。

データ・ディクショナリ・ビューUSER_TAB_COMMENTS、DBA_TAB_COMMENTS、ALL_TAB_COMMENTS、USER_COL_COMMENTS、DBA_COL_COMMENTSまたはALL_COL_COMMENTSを問い合せることによって、特定の表または列に関するコメントを表示できます。

EDITION句

コメントする既存のエディションの名前を指定します。

データ・ディクショナリ・ビューALL_EDITION_COMMENTSを問い合せると、現在のユーザーがアクセスできるエディションに関連付けられたコメントを表示できます。DBA_EDITION_COMMENTSを問い合せると、データベース内のすべてのエディションに関連付けられたコメントを表示できます。

TABLE句

コメントする表またはマテリアライズド・ビューの名前とスキーマを指定します。schemaを指定しない場合、この表およびマテリアライズド・ビューは自分のスキーマ内にあるとみなされます。

ノート:

以前のリリースでは、この句を使用してマテリアライズド・ビューにコメントを作成できました。今回のリリースでは、マテリアライズド・ビューに、COMMENT ON MATERIALIZED VIEW 句を使用する必要があります。

INDEXTYPE句

コメントする索引タイプの名前を指定します。schemaを指定しない場合、この索引タイプは自分のスキーマ内にあるとみなされます。

データ・ディクショナリ・ビューUSER_INDEXTYPE_COMMENTS、DBA_INDEXTYPE_COMMENTSまたはALL_INDEXTYPE_COMMENTSを問い合せることによって、特定の索引タイプに関するコメントを表示できます。

MATERIALIZED VIEW句

コメントするマテリアライズド・ビューの名前を指定します。schemaを指定しない場合、このマテリアライズド・ビューは自分のスキーマ内にあるとみなされます。

データ・ディクショナリ・ビューUSER_MVIEW_COMMENTS、DBA_MVIEW_COMMENTSまたはALL_MVIEW_COMMENTSを問い合せることによって、特定のマテリアライズド・ビューに関するコメントを表示できます。

MINING MODEL

コメントするマイニング・モデルの名前を指定します。

USER_MINING_MODELS、DBA_MINING_MODELSまたはALL_MINING_MODELSデータ・ディクショナリ・ビューのCOMMENTS列を問い合せると、特定のマイニング・モデルに対するコメントを表示できます。

OPERATOR句

コメントする演算子の名前を指定します。schemaを指定しない場合、その演算子は自分のスキーマ内にあるとみなされます。

データ・ディクショナリ・ビューUSER_OPERATOR_COMMENTS、DBA_OPERATOR_COMMENTSまたはALL_OPERATOR_COMMENTSを問い合せることによって、特定の演算子に関するコメントを表示できます。

IS 'string'

コメントのテキストを指定します。'string'の構文の詳細は、「テキスト・リテラル」を参照してください。

例

コメントの作成: 例

次の文は、employees表のjob_id列にコメントを挿入します。

```
COMMENT ON COLUMN employees.job_id
   IS 'abbreviated job title';
```

次の文は、データベースからこのコメントを削除します。

```
COMMENT ON COLUMN employees.job_id IS '';
```

13 SQL文: CREATE COMMITからCREATE JAVA

この章では、次のSQL文について説明します。

- COMMIT
- CREATE ANALYTIC VIEW
- CREATE ATTRIBUTE DIMENSION
- CREATE AUDIT POLICY (統合監査)
- CREATE CLUSTER
- CREATE CONTEXT
- CREATE CONTROLFILE
- CREATE DATABASE
- CREATE DATABASE LINK
- CREATE DIMENSION
- CREATE DIRECTORY
- CREATE DISKGROUP
- CREATE EDITION
- CREATE FLASHBACK ARCHIVE
- CREATE FUNCTION
- CREATE HIERARCHY
- CREATE INDEX
- CREATE INDEXTYPE
- CREATE INMEMORY JOIN GROUP
- CREATE JAVA

COMMIT

目的

COMMIT文を使用すると、現行のトランザクションを終了し、トランザクションで実行したすべての変更を確定できます。トランザクションとは、Oracle Databaseが1つの単位として扱う一連のSQL文です。また、この文によって、トランザクション内のセーブポイントがすべて消去され、トランザクション・ロックが解除されます。

トランザクションのコミット前:

- 変更された表を問い合せることで、トランザクション中に加えた変更内容を確認する。ただし、他のユーザーは変更内容を参照できません。トランザクションをコミットすると、コミット後に実行される他のユーザーの文に変更が表示されるようになります。
- トランザクション中に行った変更を、ROLLBACK文でロールバックする(元に戻す)。「ROLLBACK」を参照してください。

次のような状況では、Oracle Databaseによって暗黙的なCOMMITが発行されます。

- 構文が有効なデータ定義言語(DDL)文の前(文がエラーになる場合も同様)
- エラーが発生することなく完了するデータ定義言語(DDL)文の後

この文を使用して次の操作を実行することもできます。

- インダウト分散トランザクションを手動でコミットします。
- SET TRANSACTION文で開始した読取り専用トランザクションを終了します。

Oracle Databaseとの接続を切断する前に、最新のトランザクションを含むアプリケーション・プログラムのすべてのトランザクションを、COMMIT文またはROLLBACK文を使用して明示的に終了することをお薦めします。トランザクションを明示的にコミットしなかった場合にプログラムが異常終了すると、コミットされていない最後のトランザクションは、自動的にロールバックされます。

Oracleユーティリティおよびオラクルのツール製品が正常に終了すると、現行のトランザクションがコミットされます。Oracleプリコンパイラ・プログラムが正常に終了した場合は、トランザクションはコミットされず、現行のトランザクションがOracle Databaseによってロールバックされます。

関連項目:

- トランザクションの詳細は、『Oracle Database概要』を参照してください。
- トランザクションの特性の指定の詳細は、「SET TRANSACTION」を参照してください。

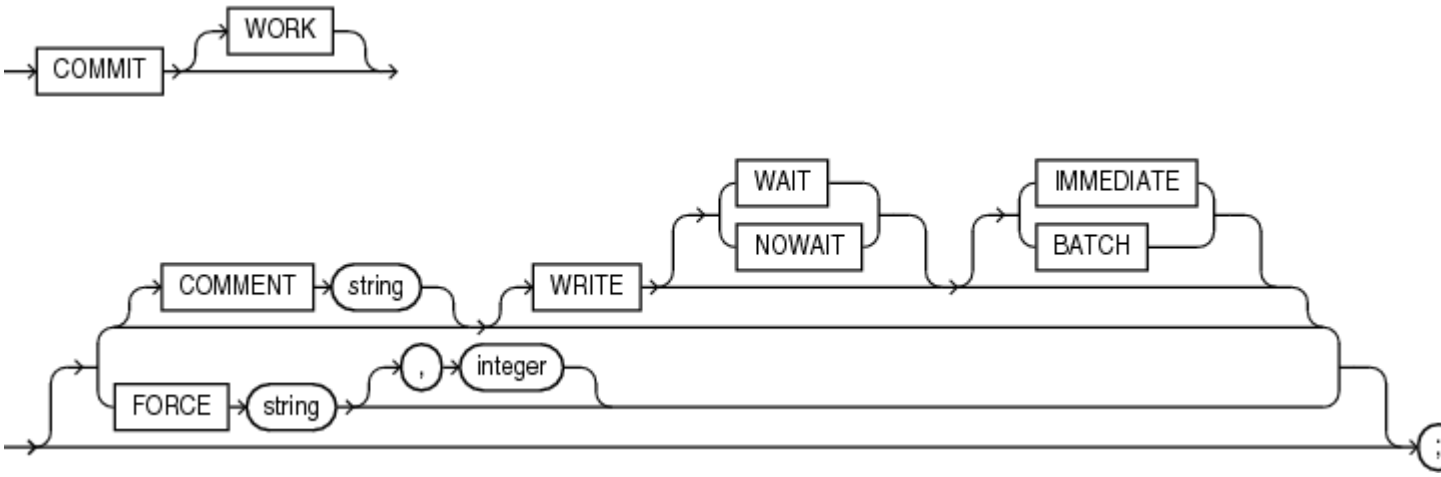
前提条件

現行のトランザクションをコミットするために、必要な権限は特にありません。

自分がコミットしたインダウト分散トランザクションを手動でコミットする場合は、FORCE TRANSACTIONシステム権限が必要です。別のユーザーがコミットしたインダウト分散トランザクションを手動でコミットする場合は、FORCE ANY TRANSACTIONシステム権限が必要です。

構文

commit::=



セマンティクス

COMMIT

COMMITキーワードに続く句は、すべてオプションです。COMMITのみを指定した場合、デフォルトはCOMMIT WORK WRITE WAIT IMMEDIATEです。

WORK

標準SQLに準拠するために、WORKキーワードがサポートされています。COMMIT文とCOMMIT WORK文は同じです。

COMMENT句

この句は、下位互換性を保つためにのみサポートされています。コミット・コメントのかわりに名前付きトランザクションを使用することをお薦めします。

関連項目:

名前付きトランザクションの詳細は、「SET TRANSACTION」および『Oracle Database概要』を参照してください。

現行のトランザクションに関するコメントを指定します。'text'は引用符で囲まれた最大255バイトのリテラルで、分散トランザクションの状態が不明(インダウト)になった場合に、そのトランザクションIDとともに、データ・ディクショナリ・ビューDBA_2PC_PENDINGに格納されます。このコメントは、分散トランザクションの障害を診断するときに役立ちます。

関連項目:

SQL文へのコメントの追加の詳細は、「COMMENT」を参照してください。

WRITE句

この句を使用すると、コミット操作で生成されるREDO情報をREDOログに書き込む優先度を指定できます。この句によって、待機時間を減らしREDOログへのI/Oを待機しないようにすることで、パフォーマンスを向上させることができます。この句は、レスポンス時間に対する要件が厳しい次のような環境下でのレスポンス時間を改善するために使用します。

- 更新トランザクションの量が多く、REDOログを頻繁にディスクに書き込む必要がある。
- アプリケーションが、非同期でコミットされるトランザクションの消失を許容できる。
- REDOログの書込みの発生を待つ待機時間が、全体のレスポンス時間に大きく影響する。

WAIT | NOWAITおよびIMMEDIATE | BATCH句を任意の順序で指定できます。

ノート:

この句を省略したときのコミット操作は、COMMIT_LOGGING および COMMIT_WAIT 初期化パラメータで制御されます(パラメータが設定されている場合)。

WAIT | NOWAIT

この句を使用すると、制御をいつユーザーに戻すかを指定できます。

- WAITパラメータを指定すると、対応するREDOがオンラインREDOログで永続的になった後にのみコミットが戻ります。BATCHモードでもIMMEDIATEモードでも、このCOMMIT文から正常にクライアントに戻ったときは、トランザクションは永続メディアにコミットされています。ログへの正常な書込みの後で障害が発生した場合、成功のメッセージがクライアントに戻らない場合があります。この場合には、トランザクションがコミットされたかどうかはクライアントにはわかりません。
- NOWAITパラメータを指定すると、REDOログへの書込みが完了したかどうかに関係なく、コミットはクライアントに戻ります。この動作はトランザクションのスループットを向上させます。WAITパラメータを指定すると、コミット・メッセージを受け取った場合にデータの損失がないことがわかります。

ノート:

NOWAIT を指定すると、コミット・メッセージを受け取った後で、REDO ログ・レコードが書き込まれる前に障害が発生した場合、その変更が永続的であることをトランザクションに誤って示す場合があります。

この句を指定しない場合、トランザクションはWAITの動作でコミットされます。

IMMEDIATE | BATCH

この句を使用すると、REDOをいつログに書き込むかを指定できます。

- IMMEDIATEパラメータを指定すると、ログ・ライター・プロセス(LGWR)によって、トランザクションのREDO情報がログに書き込まれます。この操作オプションはディスクI/Oを強制するため、トランザクションのスループットは低下します。
- BATCHパラメータを指定すると、同時に実行されている他のトランザクションとともに、REDOがREDOログにバッファされます。十分なREDO情報が収集されると、REDOログのディスク書込みが開始されます。この動作はグループ・コミットと呼ばれます。複数のトランザクションのREDOが一度のI/O操作でログに書き込まれるためです。

この句を指定しない場合、トランザクションはIMMEDIATEの動作でコミットされます。

関連項目:

非同期のコミットの詳細は、『Oracle Database概要』を参照してください。

FORCE句

分散データベース・システムでは、FORCE string [, integer]句によって、手動でインダウト分散トランザクションをコミットできます。このトランザクションは、ローカル・トランザクションIDまたはグローバル・トランザクションIDを含む'string'で識別されます。このトランザクションのIDを確認する場合は、データ・ディクショナリ・ビューDBA_2PC_PENDINGを問い合せます。また、integerを指定することによって、このトランザクションにシステム変更番号(SCN)を具体的に割り当てることもできます。

integerを指定しない場合、このトランザクションは現行のSCNを使用してコミットされます。

ノート:

FORCE 句を指定して COMMIT 文を発行した場合、指定したトランザクションのみがコミットされます。この文は、現行のトランザクションには影響しません。

関連項目:

前述の項目の詳細は、『Oracle Database管理者ガイド』を参照してください。

例

挿入のコミット: 例

この文は、hr.regions表に行を挿入して、この変更をコミットします。

```
INSERT INTO regions VALUES (5, 'Antarctica');
COMMIT WORK;
```

同じ挿入操作をコミットし、データベースに対して、ディスクI/Oを開始せずに変更をREDOログにバッファするよう指示するには、次のCOMMIT文を使用します。

```
COMMIT WRITE BATCH;
```

COMMITについてのコメント: 例

次の文は、現行のトランザクションをコミットして、そのトランザクションにコメントを関連付けます。

```
COMMIT
    COMMENT 'In-doubt transaction Code 36, Call (415) 555-2637';
```

ネットワーク障害またはマシン障害によって分散トランザクションを適切にコミットできない場合、トランザクションIDとともにデータ・ディクショナリにコメントが格納されます。そのコメントには、障害が発生したアプリケーション部分が示されており、トランザクションがコミットされたデータベースの管理者に連絡する情報が提供されています。

インダウト・トランザクションの強制: 例

次の文は、仮想のインダウト分散トランザクションを手動でコミットします。V$CORRUPT_XID_LISTデータ・ディクショナリ・ビューを問い合せて、破損トランザクションのトランザクションIDを検索します。V$CORRUPT_XID_LISTを表示し、この文を発行するには、DBA権限が必要です。

```
COMMIT FORCE '22.57.53';
```

CREATE ANALYTIC VIEW

目的

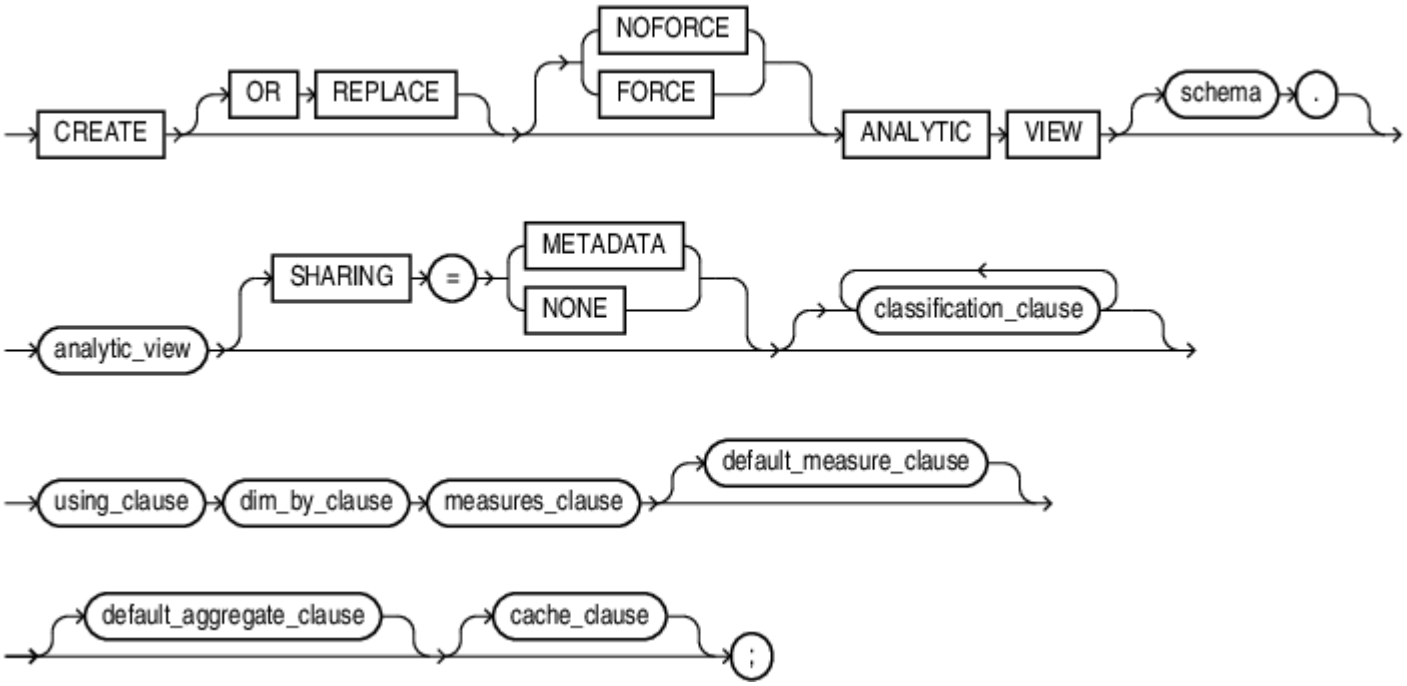
CREATE ANALYTIC VIEW文を使用すると、分析ビューを作成できます。分析ビューでは、ファクト・データのソースを指定して、計算またはデータに対して実行する他の分析操作を説明するメジャーを定義します。分析ビューでは、分析ビューの行を定義する属性ディメンションおよび属性階層も指定します。

前提条件

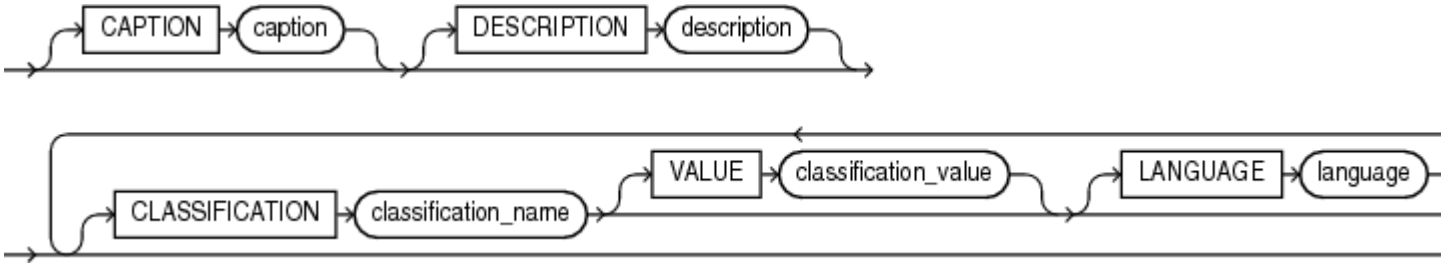
自分のスキーマ内に分析ビューを作成する場合は、CREATE ANALYTIC VIEWシステム権限が必要です。他のユーザーのスキーマ内に分析ビューを作成する場合は、CREATE ANY ANALYTIC VIEWシステム権限が必要です。

構文

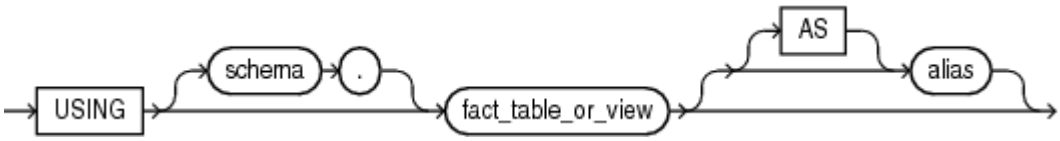
create_analytic_view::=



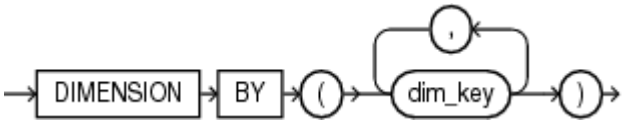
classification_clause::=



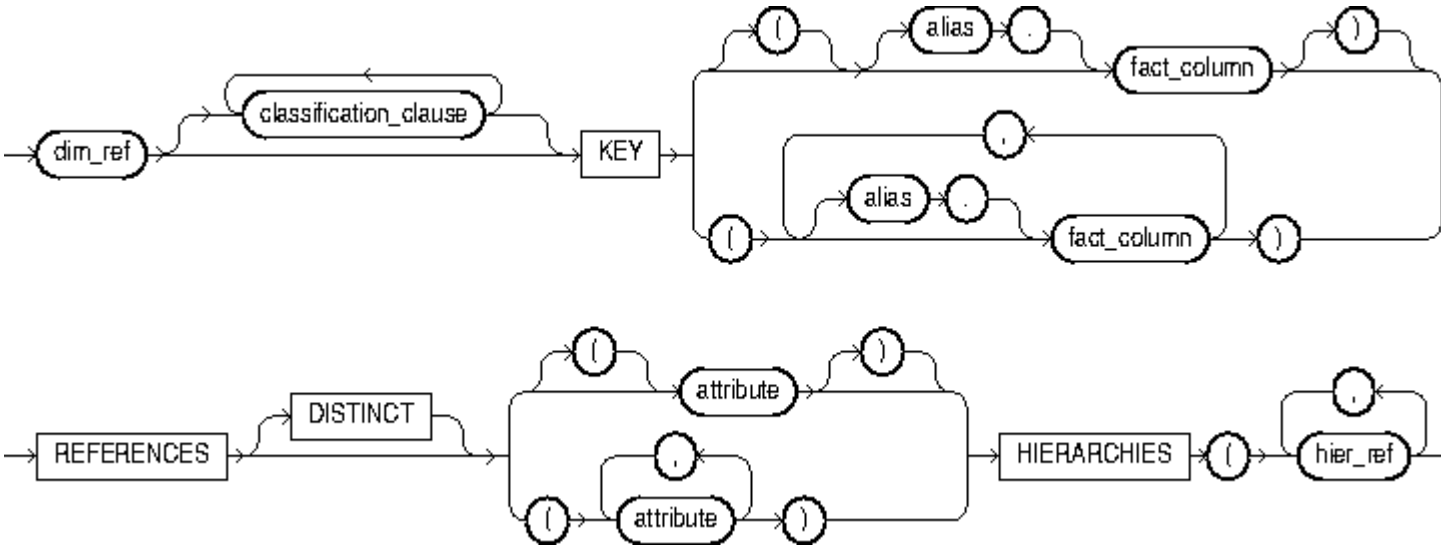
using_clause::=



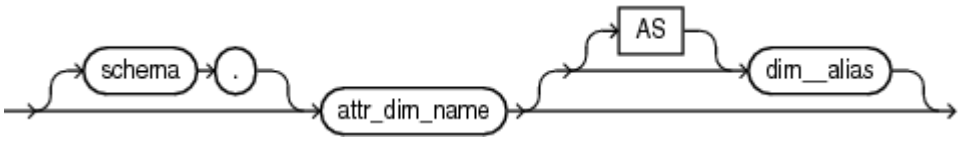
dim_by_clause::=



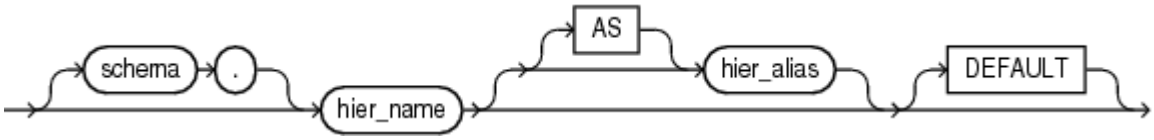
dim_key::=



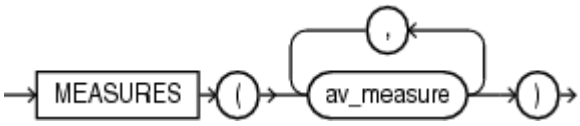
dim_ref::=



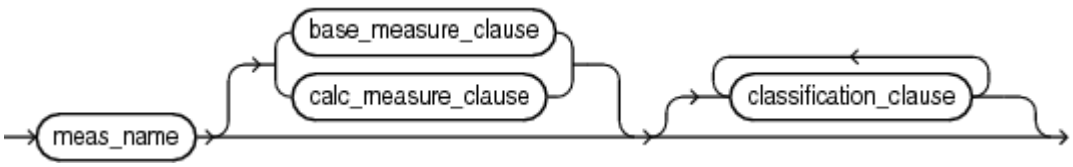
hier_ref::=



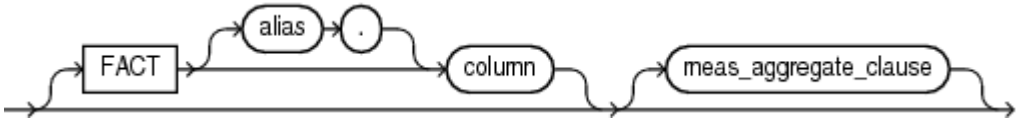
measures_clause::=



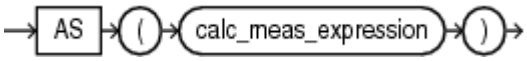
av_measure::=



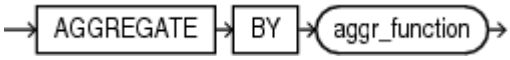
base_measure_clause::=



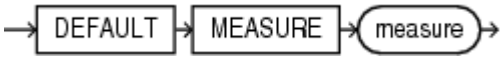
calc_measure_clause::=



meas_aggregate_clause::=

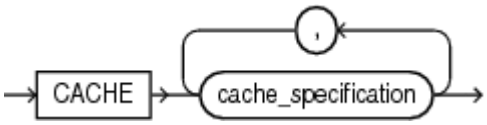


default_measure_clause::=

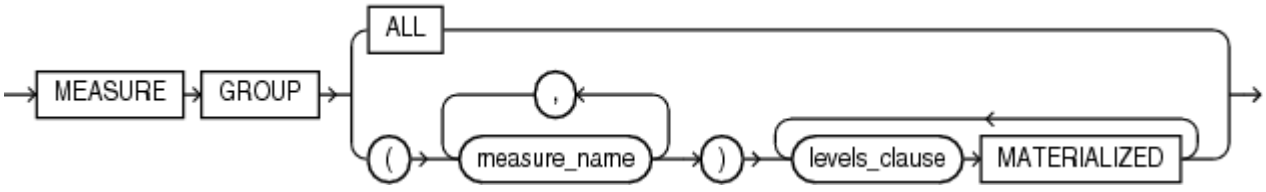


default_aggregate_clause::=

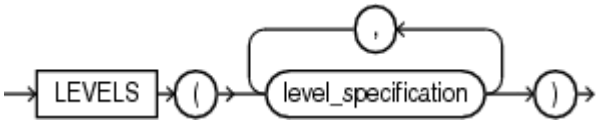
cache_clause::=



cache_specification::=



levels_clause::=



level_specification::=

セマンティクス

OR REPLACE

分析ビューの既存の定義を異なる定義で置換するには、OR REPLACEを指定します。

FORCEおよびNOFORCE

正常にコンパイルされない場合でも強制的に分析ビューを作成するには、FORCEを指定します。NOFORCEを指定した場合、分析ビューを正常にコンパイルする必要があり、そうでない場合はエラーが発生します。デフォルトは、NOFORCEです。

schema

分析ビューを作成するスキーマを指定します。スキーマを指定しない場合、自分のスキーマに分析ビューが作成されます。

analytic_view

分析ビューの名前を指定します。

SHARING

SHARING句は、アプリケーションのメンテナンスの際に、アプリケーション・ルートにオブジェクトを作成する場合に使用します。このタイプのオブジェクトはアプリケーション共通オブジェクトと呼ばれ、アプリケーション・ルートに属するアプリケーションPDBと共有できます。

次の共有属性のいずれかを使用して、オブジェクトを共有する方法を指定できます。

- METADATA - メタデータ・リンクはメタデータを共有しますが、そのデータは各コンテナに固有です。このタイプのオブジェクトは、メタデータリンク・アプリケーション共通オブジェクトと呼ばれます。
- NONE - オブジェクトは共有されず、アプリケーション・ルートでのみアクセスできます。

classification_clause

classification句を使用して、CAPTIONまたはDESCRIPTION分類の値を指定し、ユーザー定義分類を指定します。分類では、分析ビューとそのコンポーネントの情報を提供するためにアプリケーションで使用される、説明的なメタデータを提供します。

同じオブジェクトに、任意の数の分類を指定できます。分類の最大長は4000バイトです。

CAPTIONとDESCRIPTION分類では、DDLショートカットCAPTION 'caption'およびDESCRIPTION 'description'または完全な分類構文を使用できます。

言語別に分類値を変更できます。CAPTIONまたはDESCRIPTION分類に言語を指定するには、完全な構文を使用する必要があります。言語を指定しない場合、分類の言語の値はNULLです。言語の値は、NULLまたは有効なNLS_LANGUAGE値である必要があります。

using_clause

ファクト表またはビューを指定します。外部表およびリモート表を使用できます。他のスキーマ内に表またはビューを指定できます。表またはビューに別名を指定できます。

dim_by_clause

分析ビューの属性ディメンションを指定します。

dim_key

分析ビューで関連付けられている属性ディメンション、ファクト表の列、属性ディメンションの列および階層を指定します。

KEYキーワードを使用して、ファクト表に1つ以上の列を指定します。

REFERENCESキーワードを使用して、分析ビューをディメンション化する属性ディメンションの属性を指定します。各属性はレベ

ル・キーである必要があります。DISTINCTキーワードでは、非正規化ファクト表(属性ディメンションとファクト・データが同じ表に含まれている)の使用がサポートされています。ファクト・テーブルを使用して属性ディメンションが定義されている場合は、REFERENCES DISTINCTを使用します。

HIERARCHIESキーワードを使用して、属性ディメンションを使用する分析ビューの階層を指定します。

dim_ref

属性ディメンションを指定します。属性ディメンションの別名を指定できます。これは、同じディメンションを複数回使用する場合や、異なるスキーマからの同じ名前の複数のディメンションを使用する場合に必要になります。

hier_ref

階層を指定します。階層の別名を指定できます。リストのいずれかの階層をデフォルトとして指定できます。デフォルトを指定しない場合、リストの最初の階層がデフォルトになります。

measures_clause

分析ビューのメジャーを指定します。

av_measure

この分析ビュー内のメジャー全体について、単一のファクト列または式のいずれかを使用してメジャーを定義します。メジャーは、ベース・メジャーまたは計算済メジャーのいずれかになります。

base_measure_clause

オプションでファクト列またはmeas_aggregate_clauseあるいはその両方を指定して、ベース・メジャーを定義します。ファクト列を指定しない場合、分析ビューではメジャーと同じ名前のファクト表の列が使用されます。同じ名前の列が存在しない場合、エラーが発生します。

calc_measure_clause

分析ビュー式を指定して、計算済メジャーを定義します。式は分析ビューの他のメジャーを参照することもありますが、ファクト列を参照しないこともあります。計算済メジャーは集計されたベース・メジャーに対して計算されるため、aggregate句はありません。

分析ビュー式の構文と説明については、「分析ビュー式」を参照してください。

default_measure_clause

分析ビューのデフォルト・メジャーとして使用するメジャーを指定します。メジャーを指定しない場合、最初に定義されたメジャーがデフォルトになります。

meas_aggregate_clausè

ベース・メジャーのデフォルト集計演算子を指定します。集計演算子を指定しない場合は、default_aggregate_clauseによって指定された演算子が使用されます。

default_aggregate_clause

分析ビュー内のすべてのベース・メジャーについてデフォルト集計を指定します。デフォルト集計を指定しない場合、デフォルト値はSUMです。

cache_clause

適切なマテリアライズド・ビューが使用可能な場合に問合せの応答時間を改善するには、cache句を指定します。1つ以上のキャッシュ仕様を指定できます。

cache_specification

cache句に、1つ以上のメジャー・グループを指定します。すべてのメジャー・グループを含めるには、ALLを指定します。各メジャー・グループに、1つ以上のメジャーおよび1つ以上のレベルのグループ化を含めることができます。レベルのグループ化には、1つ以上のレベルを指定できます。

level_specification

キャッシュ仕様のメジャー・グループのレベルのグループ化のために、1つ以上のレベルを指定します。階層ごとに1つのレベルのみを指定します。階層レベルには、集計値を含むマテリアライズド・ビューが存在する必要があります。

例

次に、SALES_FACT表の説明を示します。

```
desc SALES_FACT
Name               Null? Type
------------------ ----- -------------
MONTH_ID                 VARCHAR2(10)
CATEGORY_ID              NUMBER(6)
STATE_PROVINCE_ID        VARCHAR2(120)
UNITS                    NUMBER(6)
SALES                    NUMBER(12,2)
```

次の例では、SALES_FACT表を使用してSALES_AV分析ビューを作成します。

```
CREATE OR REPLACE ANALYTIC VIEW sales_av
USING sales_fact
DIMENSION BY
  (time_attr_dim                          -- An attribute dimension of time data
    KEY month_id REFERENCES month_id
    HIERARCHIES (
      time_hier DEFAULT),
   product_attr_dim                       -- An attribute dimension of product data
    KEY category_id REFERENCES category_id
    HIERARCHIES (
      product_hier DEFAULT),
   geography_attr_dim                     -- An attribute dimension of store data
    KEY state_province_id
    REFERENCES state_province_id HIERARCHIES (
      geography_hier DEFAULT)
   )
MEASURES
 (sales FACT sales,                       -- A base measure
  units FACT units,                       -- A base measure
  sales_prior_period AS                   -- Calculated measures
    (LAG(sales) OVER (HIERARCHY time_hier OFFSET 1)),
  sales_year_ago AS
    (LAG(sales) OVER (HIERARCHY time_hier OFFSET 1
     ACROSS ANCESTOR AT LEVEL year)),
  chg_sales_year_ago AS
    (LAG_DIFF(sales) OVER (HIERARCHY time_hier OFFSET 1
     ACROSS ANCESTOR AT LEVEL year)),
  pct_chg_sales_year_ago AS
    (LAG_DIFF_PERCENT(sales) OVER (HIERARCHY time_hier OFFSET 1
     ACROSS ANCESTOR AT LEVEL year)),
  sales_qtr_ago AS
    (LAG(sales) OVER (HIERARCHY time_hier OFFSET 1
     ACROSS ANCESTOR AT LEVEL quarter)),
  chg_sales_qtr_ago AS
    (LAG_DIFF(sales) OVER (HIERARCHY time_hier OFFSET 1
     ACROSS ANCESTOR AT LEVEL quarter)),
  pct_chg_sales_qtr_ago AS
    (LAG_DIFF_PERCENT(sales) OVER (HIERARCHY time_hier OFFSET 1
```

```
      ACROSS ANCESTOR AT LEVEL quarter))
  )
DEFAULT MEASURE SALES;
```

CREATE ATTRIBUTE DIMENSION

目的

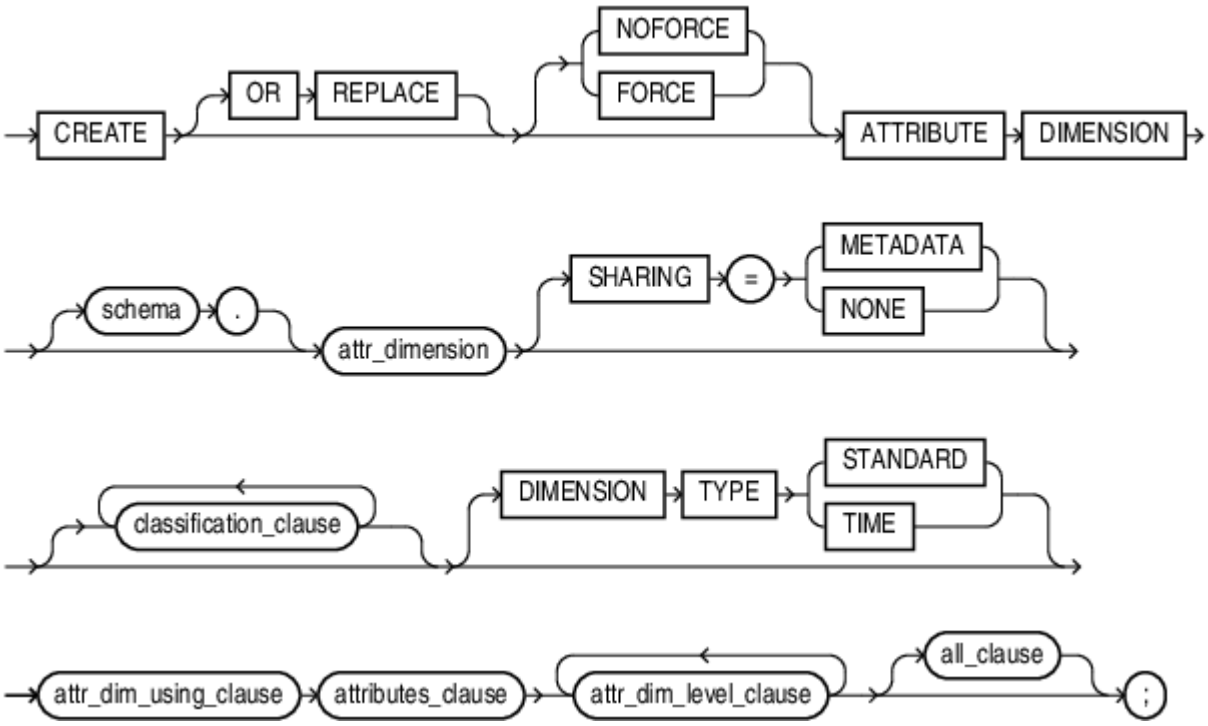
CREATE ATTRIBUTE DIMENSION文を使用して、属性ディメンションを作成します。属性ディメンションは、1つ以上の分析ビュー階層のディメンション・メンバーを指定します。使用しているデータ・ソースおよび含まれるメンバーを指定します。メンバーのレベルを指定して、レベル間の属性リレーションシップを決定します。

前提条件

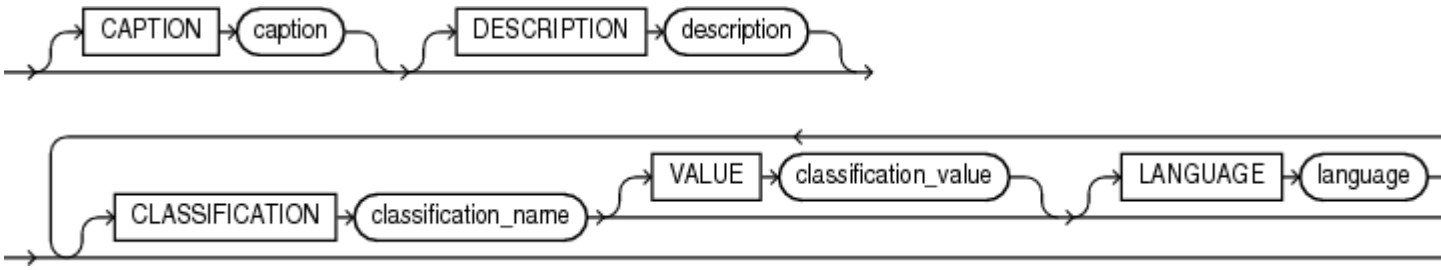
自分のスキーマ内に属性ディメンションを作成する場合は、CREATE ATTRIBUTE DIMENSIONシステム権限が必要です。他のユーザーのスキーマ内に属性ディメンションを作成するには、CREATE ANY ATTRIBUTE DIMENSIONシステム権限が必要です。

構文

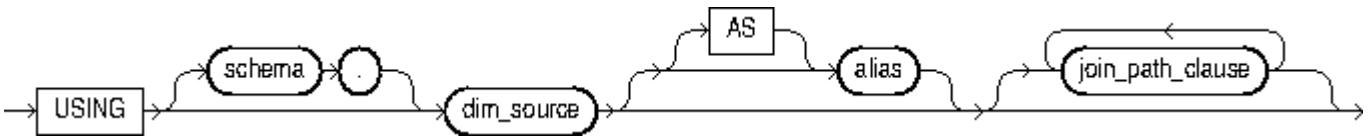
create_attribute_dimension::=



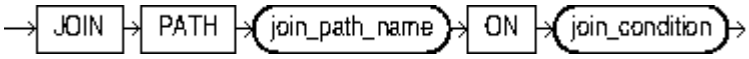
classification_clause::=



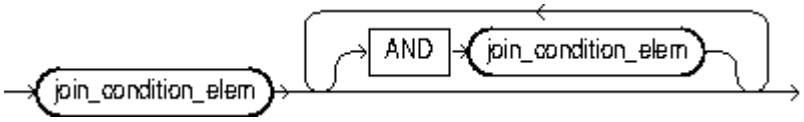
attr_dim_using_clause::=



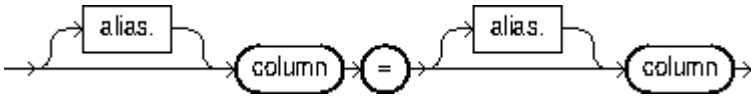
join_path_clause::=



join_condition::=

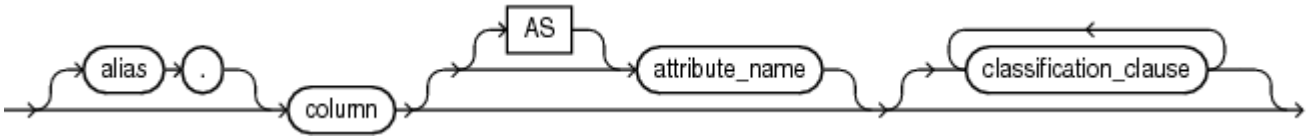


join_condition_elem ::=

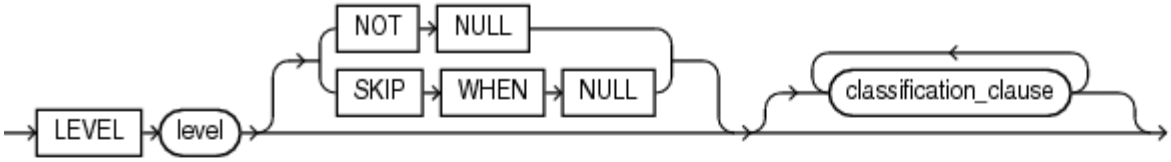


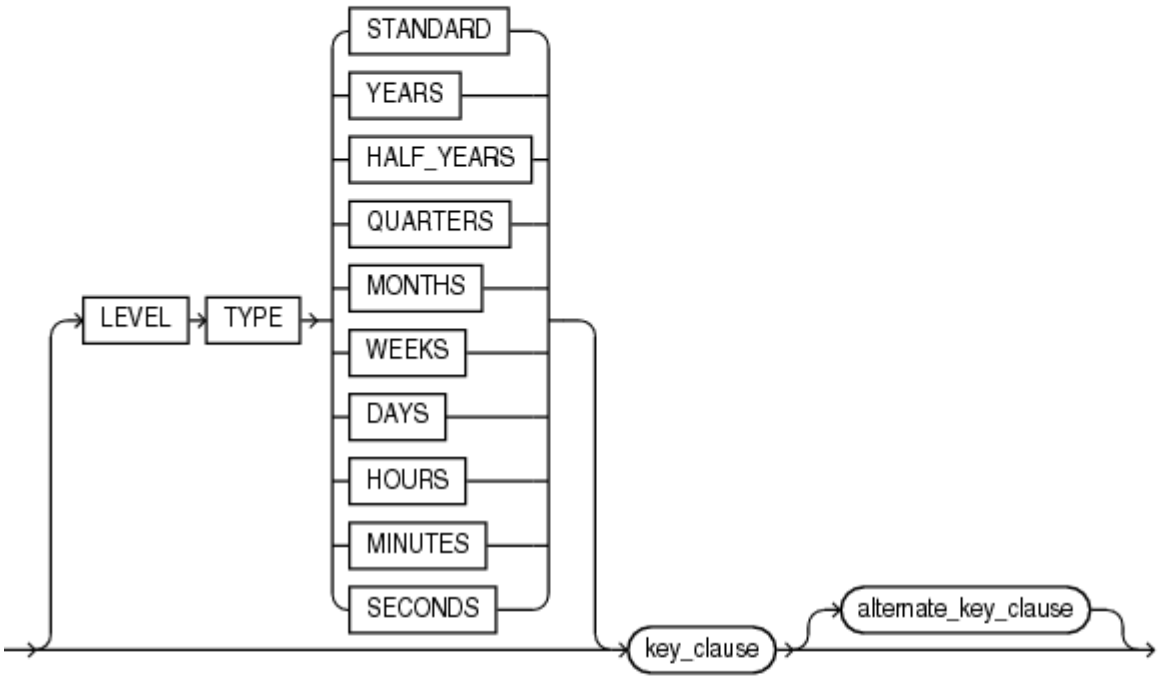
attributes_clause::=

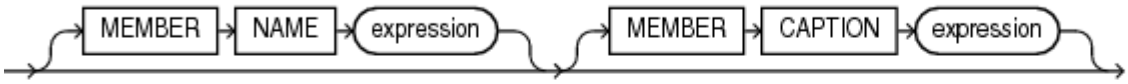
attr_dim_attributes_clause::=

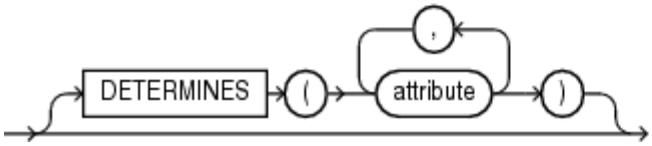


attr_dim_level_clause::=

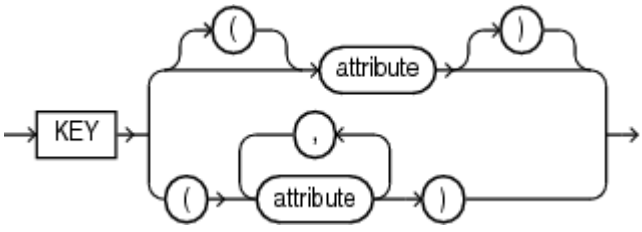




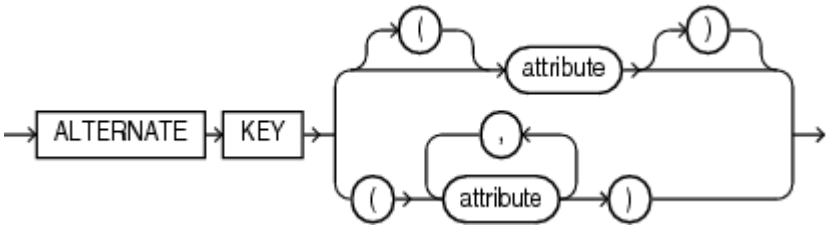




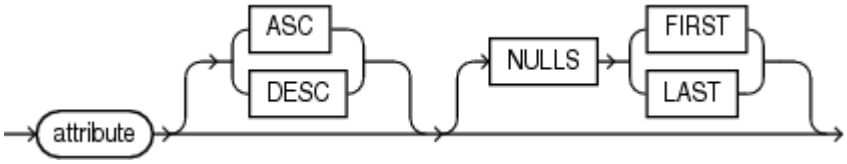
key_clause::=



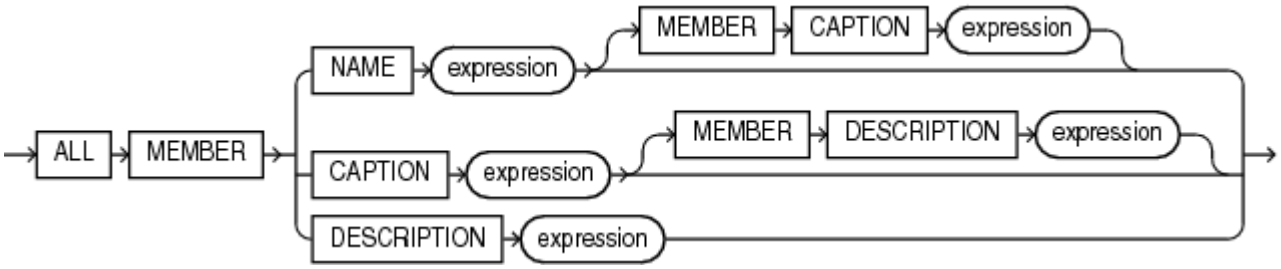
alternate_key_clause::=



dim_order_clause::=



all_clause::=



セマンティクス

OR REPLACE

属性ディメンションの既存の定義を異なる定義で置換するには、OR REPLACEを指定します。

FORCEおよびNOFORCE

正常にコンパイルされない場合でも強制的に属性ディメンションを作成するには、FORCEを指定します。NOFORCEを指定した場合、属性ディメンションを正常にコンパイルする必要があり、そうでない場合はエラーが発生します。デフォルトは、NOFORCEです。

schema

属性ディメンションを作成するスキーマを指定します。スキーマを指定しない場合、自分のスキーマに属性ディメンションが作成されます。

attr_dimension

属性ディメンションの名前を指定します。

SHARING

SHARING句は、アプリケーションのメンテナンスの際に、アプリケーション・ルートにオブジェクトを作成する場合に使用します。このタイプのオブジェクトはアプリケーション共通オブジェクトと呼ばれ、アプリケーション・ルートに属するアプリケーションPDBと共有できます。

次の共有属性のいずれかを使用して、オブジェクトを共有する方法を指定できます。

- METADATA - メタデータ・リンクはメタデータを共有しますが、そのデータは各コンテナに固有です。このタイプのオブジェクトは、メタデータリンク・アプリケーション共通オブジェクトと呼ばれます。

- NONE - オブジェクトは共有されず、アプリケーション・ルートでのみアクセスできます。

classification_clause

classification句を使用して、CAPTIONまたはDESCRIPTION分類の値を指定し、ユーザー定義分類を指定します。分類では、分析ビューとそのコンポーネントの情報を提供するためにアプリケーションで使用される、説明的なメタデータを提供します。

同じオブジェクトに、任意の数の分類を指定できます。分類の最大長は4000バイトです。

CAPTIONとDESCRIPTION分類では、DDLショートカットCAPTION 'caption'およびDESCRIPTION 'description'または完全な分類構文を使用できます。

言語別に分類値を変更できます。CAPTIONまたはDESCRIPTION分類に言語を指定するには、完全な構文を使用する必要があります。言語を指定しない場合、分類の言語の値はNULLです。言語の値は、NULLまたは有効なNLS_LANGUAGE値である必要があります。

DIMENSION TYPE

属性ディメンションのタイプとして、STANDARDまたはTIMEのいずれかを選択できます。STANDARDタイプの属性ディメンションのタイプ・レベルは、STANDARDとなります。TIMEタイプの属性ディメンションの各レベルは、いずれかの時間タイプとなります。デフォルトのDIMENSION TYPEはSTANDARDです。

attr_dim_using_clause

USINGキーワードを使用して、表またはビューを指定します。ASキーワードを使用して、表またはビューの別名を指定できます。必要に応じて、結合パスを指定できます。結合パスを使用して、属性ディメンションにスノーフレーク・スキーマ編成の表を使用する場合に結合を指定します。

join_path_clause

結合パス句では、異なる表の列間の結合条件を指定します。join_path_name引数で指定される結合パスの名前は、USING句に含まれる結合パスごとに一意である必要があります。

join_condition

結合条件は1つ以上の結合条件要素で構成され、追加の各結合条件要素はAND演算子を使用して追加されます。

join_condition_element

結合条件要素では、左側の列参照は、右側の列参照とは異なる表から取得される必要があります。

attributes_clause

1つ以上のattr_dim_attribute_clause句を指定します。

attr_dim_attribute_clause

attr_dim_using_clauseソースから列を指定します。ASキーワードを使用して別名を指定した場合を除き、属性の名前は列名です。各属性の分類を指定できます。

attr_dim_level_clause

属性ディメンションのレベルを指定します。レベルは、レベルのメンバーを指定するキー属性およびオプションの代替キー属性を指定します。

キー属性にNULL値がない場合、デフォルトであるNOT NULLを指定できます。NULL値が1つ以上ある場合は、SKIP WHEN NULLを指定します。

LEVEL TYPE

STANDARDタイプの属性ディメンションのタイプ・レベルは、STANDARDとなります。STANDARDタイプの属性ディメンションにLEVEL TYPEを指定する必要はありません。

TIMEタイプの属性ディメンションでは、レベル・タイプを指定する必要があります。レベルのタイプには、時間タイプのいずれかを選択できます。レベル・メンバーの値が時間タイプではない場合でも、時間タイプを指定する必要があります。たとえば、SEASONレベルには、季節名の値を指定できます。レベルを定義する場合は、QUARTERSなど、いずれかの時間レベル・タイプを指定する必要があります。アプリケーションでは、その目的に応じて、レベル・タイプの指定を使用できます。

DETERMINES

DETERMINESキーワードを使用すると、このレベルによって決定される属性ディメンションの他の属性を指定できます。属性に、別の属性の各値について1つの値のみが含まれる場合、最初の属性の値によってもう一方の属性の値が決まります。たとえば、QUARTER_ID属性には、MONTH_ID属性の各値について1つの値のみが含まれるため、MONTHSレベルのDETERMINES句にQUARTER_ID属性を含めることができます。

key_clause

レベルのキーとして1つ以上の属性を指定します。

alternate_key_clause

レベルの代替キーとして1つ以上の属性を指定します。

dim_order_clause

レベルのメンバーの順序を指定します。

all_clause

オプションで、MEMBER NAME、MEMBER CAPTIONおよびMEMBER DESCRIPTIONの値を暗黙的なすべてのレベルに指定します。デフォルトでは、MEMBER NAME値はALLです。

例

次の例では、TIME_DIM表について説明します。

```
desc TIME_DIM
Name               Null?   Type
------------------ -----   -------------
MONTH_ID                   VARCHAR2(10)
CATEGORY_ID                NUMBER(6)
STATE_PROVINCE_ID          VARCHAR2(120)
UNITS                      NUMBER(6)
SALES                      NUMBER(12,2)
YEAR_ID           NOT NULL VARCHAR2(30)
YEAR_NAME         NOT NULL VARCHAR2(40)
YEAR_END_DATE              DATE
QUARTER_ID        NOT NULL VARCHAR2(30)
QUARTER_NAME      NOT NULL VARCHAR2(40)
QUARTER_END_DATE           DATE
QUARTER_OF_YEAR            NUMBER
MONTH_ID          NOT NULL VARCHAR2(30)
MONTH_NAME        NOT NULL VARCHAR2(40)
MONTH_END_DATE             DATE
MONTH_OF_YEAR              NUMBER
MONTH_LONG_NAME            VARCHAR2(30)
SEASON                     VARCHAR2(10)
SEASON_ORDER               NUMBER(38)
MONTH_OF_QUARTER           NUMBER(38)
```

次の例では、TIME_DIM表からの列を使用して、TIMEタイプの属性ディメンションを作成します。

```
CREATE OR REPLACE ATTRIBUTE DIMENSION time_attr_dim
DIMENSION TYPE TIME
USING time_dim
ATTRIBUTES
 (year_id
   CLASSIFICATION caption VALUE 'YEAR_ID'
   CLASSIFICATION description VALUE 'YEAR ID',
  year_name
    CLASSIFICATION caption VALUE 'YEAR_NAME'
    CLASSIFICATION description VALUE 'Year',
  year_end_date
    CLASSIFICATION caption VALUE 'YEAR_END_DATE'
    CLASSIFICATION description VALUE 'Year End Date',
  quarter_id
    CLASSIFICATION caption VALUE 'QUARTER_ID'
    CLASSIFICATION description VALUE 'QUARTER ID',
  quarter_name
    CLASSIFICATION caption VALUE 'QUARTER_NAME'
    CLASSIFICATION description VALUE 'Quarter',
  quarter_end_date
    CLASSIFICATION caption VALUE 'QUARTER_END_DATE'
    CLASSIFICATION description VALUE 'Quarter End Date',
  quarter_of_year
    CLASSIFICATION caption VALUE 'QUARTER_OF_YEAR'
    CLASSIFICATION description VALUE 'Quarter of Year',
  month_id
    CLASSIFICATION caption VALUE 'MONTH_ID'
    CLASSIFICATION description VALUE 'MONTH ID',
  month_name
    CLASSIFICATION caption VALUE 'MONTH_NAME'
    CLASSIFICATION description VALUE 'Month',
  month_long_name
    CLASSIFICATION caption VALUE 'MONTH_LONG_NAME'
    CLASSIFICATION description VALUE 'Month Long Name',
  month_end_date
    CLASSIFICATION caption VALUE 'MONTH_END_DATE'
    CLASSIFICATION description VALUE 'Month End Date',
  month_of_quarter
    CLASSIFICATION caption VALUE 'MONTH_OF_QUARTER'
    CLASSIFICATION description VALUE 'Month of Quarter',
  month_of_year
    CLASSIFICATION caption VALUE 'MONTH_OF_YEAR'
    CLASSIFICATION description VALUE 'Month of Year',
  season
    CLASSIFICATION caption VALUE 'SEASON'
    CLASSIFICATION description VALUE 'Season',
  season_order
    CLASSIFICATION caption VALUE 'SEASON_ORDER'
    CLASSIFICATION description VALUE 'Season Order')
LEVEL month
  LEVEL TYPE MONTHS
  CLASSIFICATION caption VALUE 'MONTH'
  CLASSIFICATION description VALUE 'Month'
  KEY month_id
  MEMBER NAME month_name
  MEMBER CAPTION month_name
  MEMBER DESCRIPTION month_long_name
  ORDER BY month_end_date
  DETERMINES (month_end_date,
    quarter_id,
    season,
    season_order,
    month_of_year,
    month_of_quarter)
LEVEL quarter
```

```
  LEVEL TYPE QUARTERS
  CLASSIFICATION caption VALUE 'QUARTER'
  CLASSIFICATION description VALUE 'Quarter'
  KEY quarter_id
  MEMBER NAME quarter_name
  MEMBER CAPTION quarter_name
  MEMBER DESCRIPTION quarter_name
  ORDER BY quarter_end_date
  DETERMINES (quarter_end_date,
    quarter_of_year,
    year_id)
LEVEL year
  LEVEL TYPE YEARS
  CLASSIFICATION caption VALUE 'YEAR'
  CLASSIFICATION description VALUE 'Year'
  KEY year_id
  MEMBER NAME year_name
  MEMBER CAPTION year_name
  MEMBER DESCRIPTION year_name
  ORDER BY year_end_date
  DETERMINES (year_end_date)
LEVEL season
  LEVEL TYPE QUARTERS
  CLASSIFICATION caption VALUE 'SEASON'
  CLASSIFICATION description VALUE 'Season'
  KEY season
  MEMBER NAME season
  MEMBER CAPTION season
  MEMBER DESCRIPTION season
LEVEL month_of_quarter
  LEVEL TYPE MONTHS
  CLASSIFICATION caption VALUE 'MONTH_OF_QUARTER'
  CLASSIFICATION description VALUE 'Month of Quarter'
  KEY month_of_quarter;
```

次の例では、PRODUCT_DIM表について説明します。

```
desc PRODUCT_DIM
Name            Null?    Type
--------------- -------- -------------
DEPARTMENT_ID   NOT NULL NUMBER
DEPARTMENT_NAME NOT NULL VARCHAR2(100)
CATEGORY_ID     NOT NULL NUMBER
CATEGORY_NAME   NOT NULL VARCHAR2(100)
```

次の例では、PRODUCT_DIM表からの列を使用して、STANDARDタイプの属性ディメンションを作成します。

```
CREATE OR REPLACE ATTRIBUTE DIMENSION product_attr_dim
USING product_dim
ATTRIBUTES
 (department_id,
  department_name,
  category_id,
  category_name)
LEVEL DEPARTMENT
  KEY department_id
  ALTERNATE KEY department_name
  MEMBER NAME department_name
  MEMBER CAPTION department_name
  ORDER BY department_name
LEVEL CATEGORY
  KEY category_id
  ALTERNATE KEY category_name
  MEMBER NAME category_name
  MEMBER CAPTION category_name
  ORDER BY category_name
```

```
  DETERMINES(department_id)
ALL MEMBER NAME 'ALL PRODUCTS';
```

次の例では、GEOGRAPHY_DIM表について説明します。

```
desc GEOGRAPHY_DIM
Name                Null?    Type
---------------     -------- -------------
DEPARTMENT_ID       NOT NULL NUMBER
DEPARTMENT_NAME     NOT NULL VARCHAR2(100)
CATEGORY_ID         NOT NULL NUMBER
CATEGORY_NAME       NOT NULL VARCHAR2(100)
REGION_ID           NOT NULL VARCHAR2(120)
REGION_NAME         NOT NULL VARCHAR2(100)
COUNTRY_ID          NOT NULL VARCHAR2(2)
COUNTRY_NAME        NOT NULL VARCHAR2(120)
STATE_PROVINCE_ID   NOT NULL VARCHAR2(120)
STATE_PROVINCE_NAME NOT NULL VARCHAR2(400)
```

次の例では、GEOGRAPHY_DIM表からの列を使用して、STANDARDタイプの属性ディメンションを作成します。

```
CREATE OR REPLACE ATTRIBUTE DIMENSION geography_attr_dim
USING geography_dim
ATTRIBUTES
 (region_id,
  region_name,
  country_id,
  country_name,
  state_province_id,
  state_province_name)
LEVEL REGION
  KEY region_id
  ALTERNATE KEY region_name
  MEMBER NAME region_name
  MEMBER CAPTION region_name
  ORDER BY region_name
LEVEL COUNTRY
  KEY country_id
  ALTERNATE KEY country_name
  MEMBER NAME country_name
  MEMBER CAPTION country_name
  ORDER BY country_name
  DETERMINES(region_id)
LEVEL STATE_PROVINCE
  KEY state_province_id
  ALTERNATE KEY state_province_name
  MEMBER NAME state_province_name
  MEMBER CAPTION state_province_name
  ORDER BY state_province_name
  DETERMINES(country_id)
ALL MEMBER NAME 'ALL CUSTOMERS';
```

CREATE AUDIT POLICY (統合監査)

この項では、統合監査のCREATE AUDIT POLICY文について説明します。この種類の監査は、Oracle Database 12cで新たに導入されたもので、完全かつ高度な監査機能を提供します。統合監査の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

目的

CREATE AUDIT POLICY文を使用すると、統合監査ポリシーを作成できます。

関連項目:

- ALTER AUDIT POLICY (統合監査)
- DROP AUDIT POLICY (統合監査)
- AUDIT (統合監査)
- NOAUDIT (統合監査)

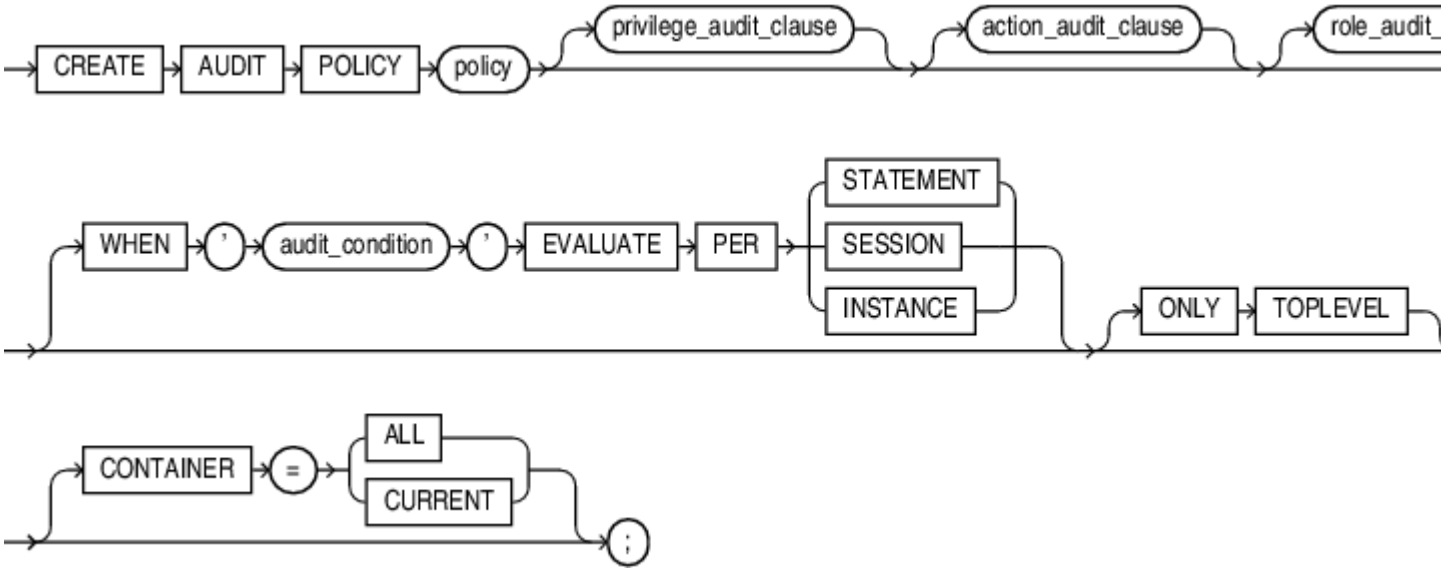
前提条件

AUDIT SYSTEMシステム権限、またはAUDIT_ADMINロールが必要になります。

CONTAINER句を指定する場合は、マルチテナント・コンテナ・データベース(CDB)に接続している必要があります。共通の統合監査ポリシーを作成する場合は、共通に付与されているAUDIT SYSTEM権限またはAUDIT_ADMIN共通ロールが必要です。ローカルの統合監査ポリシーを作成する場合は、共通に付与されているAUDIT SYSTEM権限またはAUDIT_ADMIN共通ロールを保有しているか、接続先のコンテナでローカルに付与されているAUDIT SYSTEM権限またはAUDIT_ADMINローカル・ロールを保有している必要があります。

構文

create_audit_policy::=

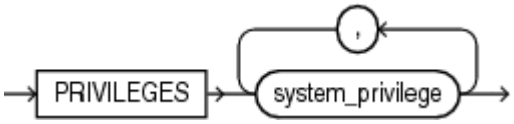


ノート:

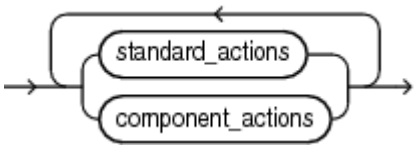
少なくとも `privilege_audit_clause` 句、`action_audit_clause` 句、または `role_audit_clause` 句のいずれか 1 つを指定する必要があります。

(privilege_audit_clause::=、action_audit_clause::=、role_audit_clause::=)

privilege_audit_clause::=



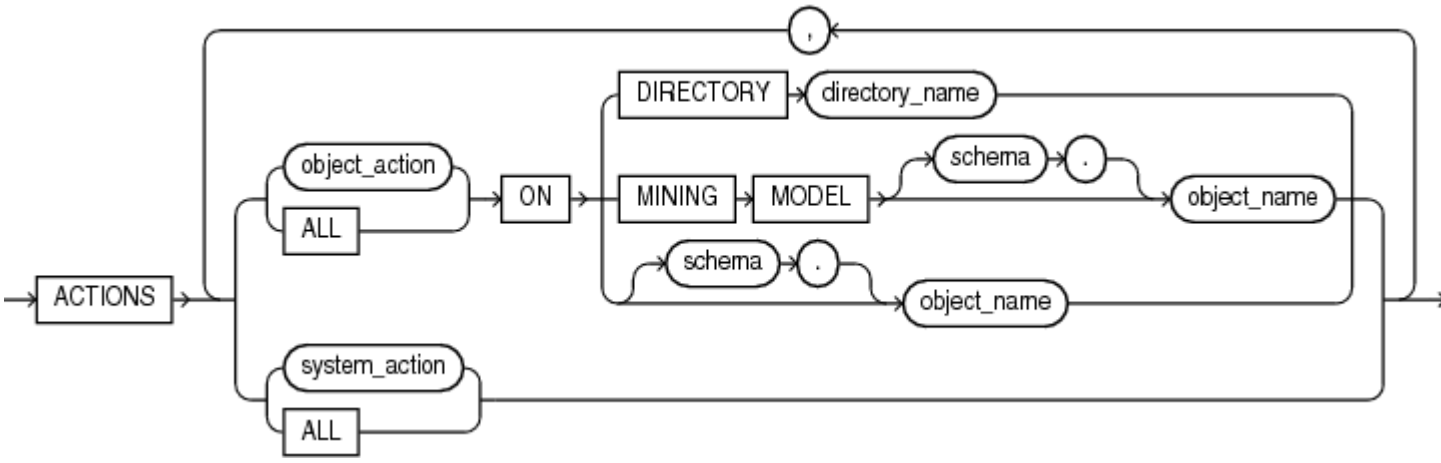
action_audit_clause::=



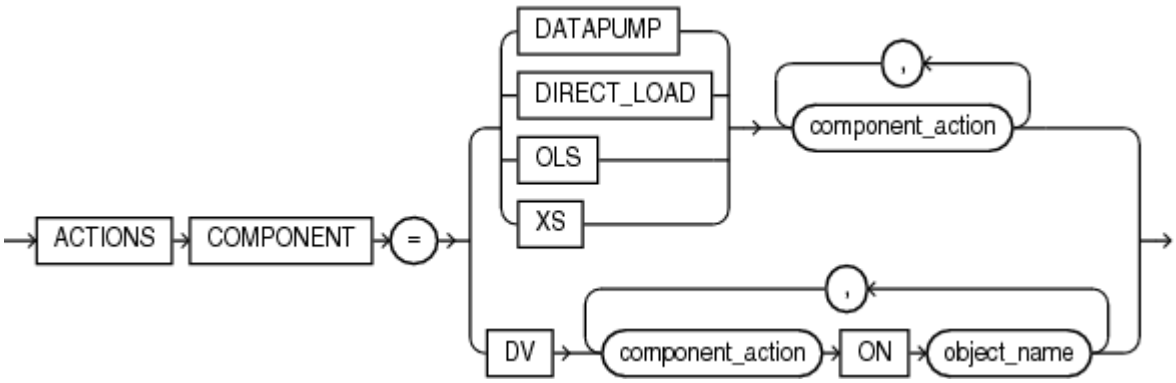
ノート:

`standard_actions` 句と `component_actions` 句は単独で指定することも、どちらかの順序で指定することもできますが、それぞれの句を指定できるのは 1 回のみです。

standard_actions::=



component_actions::=



role_audit_clause::=

セマンティクス

policy

作成する統合監査ポリシーの名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

すべての監査ポリシーの名前を確認するには、AUDIT_UNIFIED_POLICIESビューを問い合せます。

関連項目:

AUDIT_UNIFIED_POLICIESビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

privilege_audit_clause

この句を使用すると、1つ以上のシステム権限を監査できます。system_privilegeには、有効なシステム権限を指定します。すべての有効なシステム権限を確認するには、SYSTEM_PRIVILEGE_MAPビューのNAME列を問い合せます。

システム権限を正常に利用するSQL文のみが監査されます。文でシステム権限が使用されていない場合、privilege_audit_clauseでは文の監査は行われません。

システム権限の監査の制限事項

システム権限INHERIT ANY PRIVILEGES、SYSASM、SYSBACKUP、SYSDBA、SYSDG、SYSKM、SYSRACおよびSYSOPERは監査できません。

action_audit_clause

この句を使用すると、1つ以上のアクションを監査対象に指定できます。standard_actions句を使用すると、標準RDBMSオブジェクトに対するアクションおよびデータベースに対する標準RDBMSシステム・アクションを監査できます。component_actions句を使用すると、コンポーネントに対するアクションを監査できます。

standard_actions

この句を使用すると、標準RDBMSオブジェクトに対するアクションおよびデータベースに対する標準RDBMSシステム・アクションを監査できます。

object_action ON

この句を使用すると、指定したオブジェクトに対するアクションを監査できます。object_actionには、監査するアクションを指定します。表13-1に、各オブジェクト型について監査可能なアクションを示します。

ALL ON

この句を使用すると、指定したオブジェクトに対するすべてのアクションを監査できます。ON句で指定したオブジェクトのタイプについて、表13-1にリストされたすべてのアクションが監査されます。

ON句

ON句を使用すると、監査するオブジェクトを指定できます。ディレクトリとデータ・マイニング・モデルは、個別のネームスペースに含まれるため個別に識別されます。ディレクトリに対するアクションを監査する場合は、ON DIRECTORY directory_nameを指定します。データ・マイニング・モデルに対するアクションを監査する場合は、ON MINING MODEL object_nameを指定します。表13-1にリストされたその他のオブジェクトのタイプに対するアクションを監査するには、ON object_nameを指定します。object_nameをschemaで修飾しなかった場合、そのオブジェクトは自分のスキーマ内にあるとみなされます。

表13-1 統合監査の対象オブジェクトとアクション

オブジェクトのタイプ	**アクション**
ディレクトリ	AUDIT、GRANT、READ
ファンクション	AUDIT、EXECUTE (ノート 1 と 2)、GRANT
Java スキーマ・オブジェクト(ソース、クラス、リソース)	AUDIT、EXECUTE、GRANT
ライブラリ	EXECUTE、GRANT
マテリアライズド・ビュー	ALTER、AUDIT、COMMENT、DELETE、INDEX、INSERT、LOCK、SELECT、UPDATE
マイニング・モデル	AUDIT、COMMENT、GRANT、RENAME、SELECT
オブジェクト型	ALTER、AUDIT、GRANT
パッケージ	AUDIT、EXECUTE、GRANT
プロシージャ	AUDIT、EXECUTE (ノート 1 と 2)、GRANT
順序	ALTER、AUDIT、GRANT、SELECT
表	ALTER、AUDIT、COMMENT、DELETE、FLASHBACK、GRANT、INDEX、INSERT、LOCK、RENAME、SELECT、UPDATE

オブジェクトのタイプ	アクション
ビュー	AUDIT、DELETE、FLASHBACK、GRANT、INSERT、LOCK、RENAME、SELECT、UPDATE

ノート1: PL/SQLストアド・プロシージャまたはストアド・ファンクションに対するEXECUTE操作を監査する場合は、監査目的での操作の成否を判断する際に、プロシージャまたはファンクションを見つけてその実行を認証できるかどうかのみが監査対象となります。したがって、WHENEVER NOT SUCCESSFUL句を指定すると、無効なオブジェクト・エラー、存在しないオブジェクト・エラー、および認証の失敗が監査されます。プロシージャまたはファンクションの実行時に検出されたエラーは監査されません。WHENEVER SUCCESSFUL句を指定すると、実行時にエラーが検出されたかどうかに関係なく、無効なオブジェクト・エラー、存在しないオブジェクト・エラー、および認証の失敗が監査されます。

ノート2: PL/SQLストアド・プロシージャまたはストアド・ファンクション内の再帰的なSQL操作の失敗を監査するには、SQL操作の監査を構成します。

ノート3: データベース内のPL/SQLストアド・プロシージャ、ファンクションまたはパッケージに対するEXECUTEの監査は、プロシージャ、ファンクションまたはパッケージのインスタンス化フェーズで行われます。

system_action

この句を使用すると、データベースに対するシステム・アクションを監査できます。system_actionに有効な値を確認するには、COMPONENTが'Standard'であるAUDITABLE_SYSTEM_ACTIONSビューのNAME列を問い合せます。

例: 統合監査でのCHANGE PASSWORDの監査

パスワードの変更を監査する監査ポリシーを構成することで、CHANGE PASSWORDシステム・アクションを監査できます。監査ポリシーを構成した後、その監査ポリシーを有効にする必要があります。

次の例では、アクションCHANGE PASSWORDを監査する監査ポリシーmypolicyを作成します:

```
CREATE AUDIT POLICY mypolicy ACTIONS CHANGE PASSWORD;
----------------------
Audit policy created.
```

次の文は、監査ポリシーmypolicyを有効にします:

```
AUDIT POLICY mypolicy;
```

監査ポリシーmypolicyは、パスワードの変更の成功および失敗の両方についてCHANGE PASSWORDアクションを監査します。

パスワードhr_pwdを持つユーザーhr_usrは、PDB hr_pdbに接続し、次のようにパスワードを変更できます。

```
CONNECT hr_usr/hr_pwd@hr_pdb;
PASSWORD
Changing password for hr_usr
Old password:
New password:
Retype new password:
Password changed.
```

前述のSQL*Plusの例では、ユーザーhr_usrによって実行されるコマンドPASSWORDによって、監査レコードを生成するCHANGE PASSWORDアクションが開始されます。

次のようにUNIFIED_AUDIT_TRAILを問い合せると、レコードを表示できます:

```
SELECT ACTION_NAME, UNIFIED_AUDIT_POLICIES, OBJECT_NAME FROM UNIFIED_AUDIT_TRAIL;
ACTION_NAME
--------------------------------------------------------------------
UNIFIED_AUDIT_POLICIES
--------------------------------------------------------------------
OBJECT_NAME
--------------------------------------------------------------------
CHANGE PASSWORD
MYPOLICY
HR_USR
```

監査ポリシーmypolicyは、ALTER USER文を介してパスワードの変更を取得しないことに注意してください。

ALL

この句を使用すると、データベースに対するシステムのすべてのアクションを監査できます。

component_actions

この句を使用すると、各コンポーネント(Oracle Data Pump、Oracle SQL*Loader Direct Path Load、Oracle Label Security、Oracle Database Real Application Security、Oracle Database Vault)に対するアクションを監査できます。

DATAPUMP

この句を使用すると、Oracle Data Pumpに対するアクションを監査できます。component_actionには、監査するアクションを指定します。Oracle Data Pumpの有効なアクションを確認するには、COMPONENTがDatapumpであるAUDITABLE_SYSTEM_ACTIONSビューのNAME列を問い合せます。たとえば:

```
SELECT name FROM auditable_system_actions WHERE component = 'Datapump';
```

Oracle Data Pumpの監査方法の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

DIRECT_LOAD

この句を使用すると、Oracle SQL*Loader Direct Path Loadに対するアクションを監査できます。component_actionには、監査するアクションを指定します。Oracle SQL*Loader Direct Path Loadの有効なアクションを確認するには、COMPONENTがDirect path APIであるAUDITABLE_SYSTEM_ACTIONSビューのNAME列を問い合せます。たとえば:

```
SELECT name FROM auditable_system_actions WHERE component = 'Direct path API';
```

Oracle SQL*Loader Direct Path Loadの監査方法の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

OLS

この句を使用すると、Oracle Label Securityに対するアクションを監査できます。component_actionには、監査するアクションを指定します。Oracle Label Securityの有効なアクションを確認するには、COMPONENTがLabel SecurityであるAUDITABLE_SYSTEM_ACTIONSビューのNAME列を問い合せます。たとえば:

```
SELECT name FROM auditable_system_actions WHERE component = 'Label Security';
```

Oracle Label Securityの監査方法の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

XS

この句を使用すると、Oracle Database Real Application Securityに対するアクションを監視できます。component_actionには、監査するアクションを指定します。Oracle Database Real Application Securityの有効なアクションを確認するには、COMPONENTがXSであるAUDITABLE_SYSTEM_ACTIONSビューのNAME列を問い合せます。たとえば:

```
SELECT name FROM auditable_system_actions WHERE component = 'XS';
```

Oracle Database Real Application Securityの監査方法の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

DV

この句を使用すると、Oracle Database Vaultに対するアクションを監査できます。component_actionには、監査するアクションを指定します。Oracle Database Vaultの有効なアクションを確認するには、COMPONENTがDatabase VaultであるAUDITABLE_SYSTEM_ACTIONSビューのNAME列を問い合せます。たとえば:

```
SELECT name FROM auditable_system_actions WHERE component = 'Database Vault';
```

object_nameには、監査するDatabase Vaultオブジェクトの名前を指定します。

Oracle Database Vaultの監査方法の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

role_audit_clause

この句を使用すると、監査するロールを1つ以上指定できます。ロールを監査すると、そのロールに直接付与されたすべてのシステム権限が監査されます。このシステム権限が必要になるSQL文が監査されます。roleには、ユーザー定義(ローカルまたは外部)のロール、または事前定義されたロールを指定します。事前定義されているロールのリストは、『Oracle Databaseセキュリティ・ガイド』を参照してください。

WHEN句

この句を使用すると、統合監査ポリシーが適用される時間を制御できます。

audit_condition

統合監査ポリシーが適用されているかどうかを判断する条件を指定します。audit_conditionの評価がTRUEの場合は、ポリシーが適用されています。FALSEの場合は、ポリシーが適用されていません。

audit_conditionの最大長は4,000文字です。これには、式だけでなく、次のファンクションと条件を含めることができます。

- 数値ファンクション: BITAND、CEIL、FLOOR、POWER
- 文字値を返す文字ファンクション: CONCAT、LOWER、UPPER
- 数値を返す文字ファンクション: INSTR、LENGTH
- 環境および識別子ファンクション: SYS_CONTEXT、UID
- 比較条件: =、!=、<>、<、>、<=、>=
- 論理条件: AND、OR
- NULL条件: IS [NOT] NULL
- [NOT] BETWEEN条件
- [NOT] IN条件

audit_conditionは、一重引用符で囲む必要があります。audit_conditionに一重引用符が含まれているときには、そのかわりに2つの一重引用符を指定します。たとえば、次の条件を指定するとします。

```
SYS_CONTEXT('USERENV', 'CLIENT_IDENTIFIER') = 'myclient'
```

'audit_condition'は、次のように指定します。

```
'SYS_CONTEXT(''USERENV'', ''CLIENT_IDENTIFIER'') = ''myclient'''
```

EVALUATE PER句は、各コンテナのインスタンスごとに監査条件を評価します。たとえば、あるコンテナ内で条件が評価された場合、その条件は、インスタンスが同一の場合でも他のコンテナ内で再度評価されます。

EVALUATE PER STATEMENT

この句を指定すると、監査可能な文ごとにaudit_conditionを評価できます。audit_conditionの評価がTRUEの場合、統合監査ポリシーが文に適用されます。FALSEの場合、統合監査ポリシーは文に適用されません。

EVALUATE PER SESSION

この句を指定すると、セッション中にaudit_conditionを1回評価します。audit_conditionは、セッション中に実行される最初の監査可能な文に対して評価されます。audit_conditionの評価がTRUEの場合、統合監査ポリシーは、その後のセッション中に該当するすべての文に適用されます。FALSEの場合、統合監査ポリシーは、その後のセッション中に該当するすべての文に適用されません。

EVALUATE PER INSTANCE

この句を指定すると、インスタンスの存続期間中にaudit_conditionを1回評価します。audit_conditionは、インスタンスの存続期間中に実行される最初の監査可能な文に対して評価されます。audit_conditionの評価がTRUEの場合、統合監査ポリシーは、インスタンスのその後の存続期間中に該当するすべての文に適用されます。FALSEの場合、統合監査ポリシーは、インスタンスのその後の存続期間中に該当するすべての文に適用されません。

ONLY TOPLEVEL

ユーザーが直接発行したSQL文を監査する場合は、ONLY TOPLEVELを指定します。

PL/SQLプロシージャ内から実行されるSQL文は、トップレベルの文とはみなされません。トップレベルの文は、SYSも含めてすべてのユーザーが監査できます。

詳細は、Oracle Databaseセキュリティ・ガイドを参照してください。

CONTAINER句

CONTAINER句を使用すると、統合監査ポリシーの有効範囲を指定できます。

- CONTAINER = ALLを指定すると、共通の統合監査ポリシーを作成できます。このタイプのポリシーは、CDB内のすべてのプラガブル・データベース(PDB)に使用できます。現在のコンテナがルートである必要があります。ACTIONS object_action ONまたはACTIONS ALL ON句を指定する場合は、共通オブジェクトまたはアプリケーション共通オブジェクトを指定する必要があります。
- CONTAINER = CURRENTを指定すると、接続先のコンテナにローカルの統合監査ポリシーを作成できます。現在のコンテナは、ルートまたはPDBにできます。

この句を省略すると、CONTAINER = CURRENTがデフォルトになります。

ノート:

統合監査ポリシーは、作成後に有効範囲を変更できません。

例

システム権限の監査: 例

次の文は、統合監査ポリシーtable_polを作成します。このポリシーは、システム権限CREATE ANY TABLEおよびDROP ANY TABLEを監査します。

```
CREATE AUDIT POLICY table_pol
  PRIVILEGES CREATE ANY TABLE, DROP ANY TABLE;
```

次の文は、table_polがAUDIT_UNIFIED_POLICIESビューにあることを確認します。

```
SELECT *
  FROM audit_unified_policies
  WHERE policy_name = 'TABLE_POL';
```

オブジェクトに対するアクションの監査: 例

次の文では、統合監査ポリシーdml_polを作成します。このポリシーは、表hr.employeesに対するDELETE、INSERT、UPDATEアクション、および表hr.departmentsに対するすべての監査可能なアクションを監査します。

```
CREATE AUDIT POLICY dml_pol
  ACTIONS DELETE on hr.employees,
          INSERT on hr.employees,
          UPDATE on hr.employees,
          ALL on hr.departments;
```

次の文は、統合監査ポリシーread_dir_polを作成します。このポリシーは、ディレクトリbfile_dir (「ディレクトリの作成: 例」で作成したディレクトリ)に対するREADアクションを監査します。

```
CREATE AUDIT POLICY read_dir_pol
  ACTIONS READ ON DIRECTORY bfile_dir;
```

システム・アクションの監査: 例

次の問合せは、データベースに対する監査可能な標準RDBMSシステム・アクションを表示します。

```
SELECT name FROM auditable_system_actions
  WHERE component = 'Standard'
  ORDER BY name;
NAME
----
ADMINISTER KEY MANAGEMENT
ALL
ALTER ASSEMBLY
ALTER AUDIT POLICY
ALTER CLUSTER
...
```

次の文は、統合監査ポリシーsecurity_polを作成します。このポリシーは、システム・アクションADMINISTER KEY MANAGEMENTを監査します。

```
CREATE AUDIT POLICY security_pol
  ACTIONS ADMINISTER KEY MANAGEMENT;
```

次の文は、統合監査ポリシーdir_polを作成します。このポリシーは、任意のディレクトリに対するすべての読取り、書込みおよび実行操作を監査します。

```
CREATE AUDIT POLICY dir_pol
  ACTIONS READ DIRECTORY, WRITE DIRECTORY, EXECUTE DIRECTORY;
```

次の文は、統合監査ポリシーall_actions_polを作成します。このポリシーは、データベースに対するすべての標準RDBMSシステム・アクションを監査します。

```
CREATE AUDIT POLICY all_actions_pol
  ACTIONS ALL;
```

コンポーネント・アクションの監査: 例

次の問合せは、Oracle Data Pumpに対する監査可能なアクションを表示します。

```
SELECT name FROM auditable_system_actions
  WHERE component = 'Datapump';
NAME
----
EXPORT
IMPORT
ALL
```

次の文は、統合監査ポリシーdp_actions_polを作成します。このポリシーは、Oracle Data Pumpに対するIMPORTアクションを監査します。

```
CREATE AUDIT POLICY dp_actions_pol
  ACTIONS COMPONENT = datapump IMPORT;
```

ロールの監査: 例

次の文は、統合監査ポリシーjava_polを作成します。このポリシーは、事前定義されたロールjava_adminとjava_deployを監査します。

```
CREATE AUDIT POLICY java_pol
  ROLES java_admin, java_deploy;
```

システム権限、アクションおよびロールの監査: 例

次の文は、統合監査ポリシーhr_admin_polを作成します。このポリシーは、複数のシステム権限、アクションおよびロールを監査します。

```
CREATE AUDIT POLICY hr_admin_pol
  PRIVILEGES CREATE ANY TABLE, DROP ANY TABLE
  ACTIONS DELETE on hr.employees,
          INSERT on hr.employees,
          UPDATE on hr.employees,
          ALL on hr.departments,
          LOCK TABLE
  ROLES audit_admin, audit_viewer;
```

統合監査ポリシーをいつ適用するかの制御: 例

次の文は、統合監査ポリシーorder_updates_polを作成します。このポリシーは、表oe.ordersに対するUPDATEアクションを監査します。このポリシーは、外部ユーザーが監査可能な文を発行したときのみ適用されます。監査条件は、セッションごとに1回検査されます。

```
CREATE AUDIT POLICY order_updates_pol
  ACTIONS UPDATE ON oe.orders
  WHEN 'SYS_CONTEXT(''USERENV'', ''IDENTIFICATION_TYPE'') = ''EXTERNAL'''
```

```
  EVALUATE PER SESSION;
```

次の文は、統合監査ポリシーemp_updates_polを作成します。このポリシーは、表hr.employeesに対するDELETE、INSERT、UPDATEアクションを監査します。このポリシーは、UIDが100、105、または107ではないユーザーが監査可能な文を発行したときにのみ適用されます。監査条件は、監査可能な文ごとに検査されます。

```
CREATE AUDIT POLICY emp_updates_pol
  ACTIONS DELETE on hr.employees,
          INSERT on hr.employees,
          UPDATE on hr.employees
  WHEN 'UID NOT IN (100, 105, 107)'
  EVALUATE PER STATEMENT;
```

ローカルの統合監査ポリシーの作成: 例

次の文は、統合監査ポリシーlocal_table_polを作成します。このポリシーは、現在のコンテナ内のシステム権限CREATE ANY TABLEおよびDROP ANY TABLEを監査します。

```
CREATE AUDIT POLICY local_table_pol
  PRIVILEGES CREATE ANY TABLE, DROP ANY TABLE
  CONTAINER = CURRENT;
```

共通の統合監査ポリシーの作成: 例

次の文は、共通の統合監査ポリシーcommon_role1_polを作成します。このポリシーは、CDB全体にわたって共通ロールc##role1 (「CREATE ROLE」の「例」で作成した共通ロール)を監査します。

```
CREATE AUDIT POLICY common_role1_pol
  ROLES c##role1
  CONTAINER = ALL;
```

CREATE CLUSTER

目的

CREATE CLUSTER文を使用すると、クラスタを作成できます。クラスタとは、1つ以上の表のデータが含まれているスキーマ・オブジェクトのことです。

- 索引クラスタには複数の表が含まれている必要があり、クラスタ内のすべての表には1つ以上の共通の列があります。Oracle Databaseでは、同一のクラスタ・キーを共有するすべての表からすべての行がまとめて格納されます。
- ハッシュ・クラスタ(1つ以上の表を含むことが可能)には、同一のハッシュ・キー値を持つ行がまとめて格納されます。

既存のクラスタの情報を取得するには、データ・ディクショナリ・ビューUSER_CLUSTERS、ALL_CLUSTERSおよびDBA_CLUSTERSを問い合せます。

関連項目:

- クラスタの概要は、『Oracle Database概要』を参照してください。
- クラスタを使用するタイミングについては、『Oracle Database SQLチューニング・ガイド』を参照してください。
- データ・ディクショナリ・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

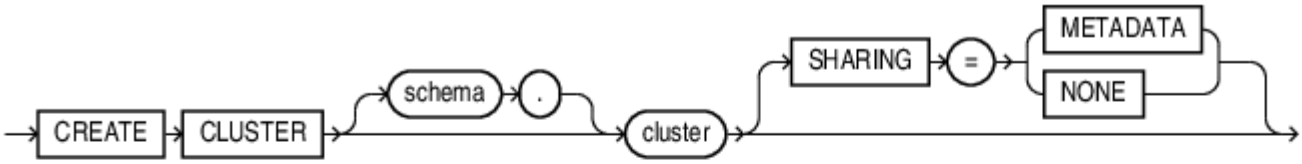
前提条件

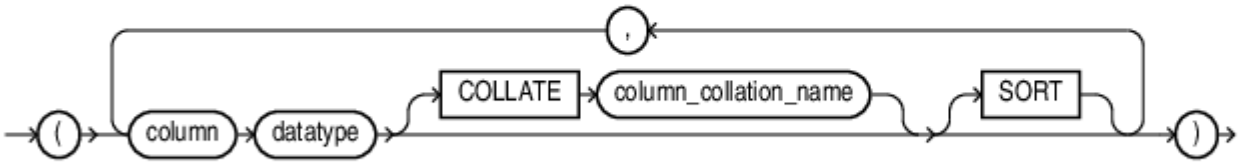
自分のスキーマにクラスタを作成する場合は、CREATE CLUSTERシステム権限が必要です。他のユーザーのスキーマ内にクラスタを作成する場合は、CREATE ANY CLUSTERシステム権限が必要です。また、クラスタを設定するスキーマの所有者は、クラスタが定義されている表領域に対する領域の割当て制限、またはUNLIMITED TABLESPACEシステム権限のいずれかが必要です。

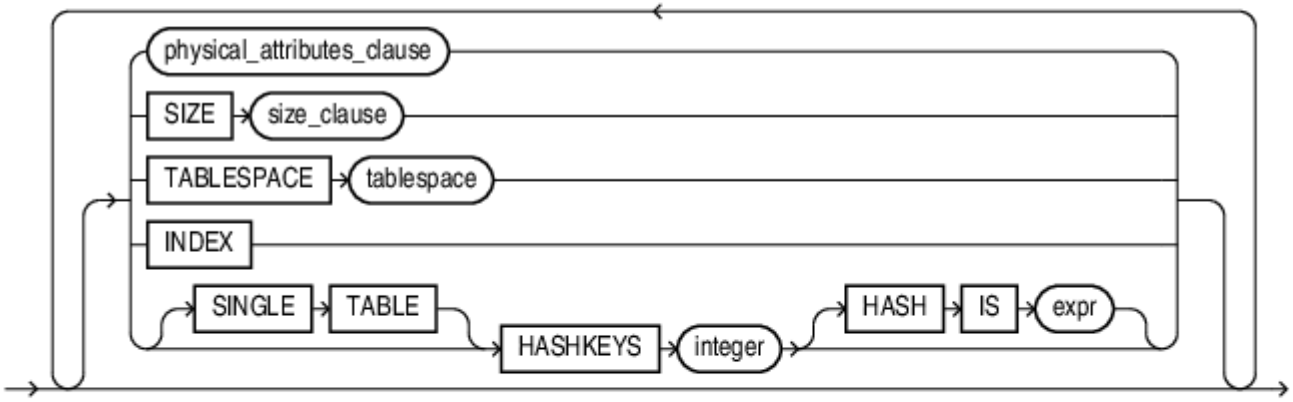
クラスタを最初に作成する場合は、Oracle Databaseはクラスタに対する索引を自動的に作成しません。このため、CREATE INDEX文でクラスタ索引を作成するまでは、索引クラスタのクラスタ表に対して、データ操作言語(DML)文を発行できません。

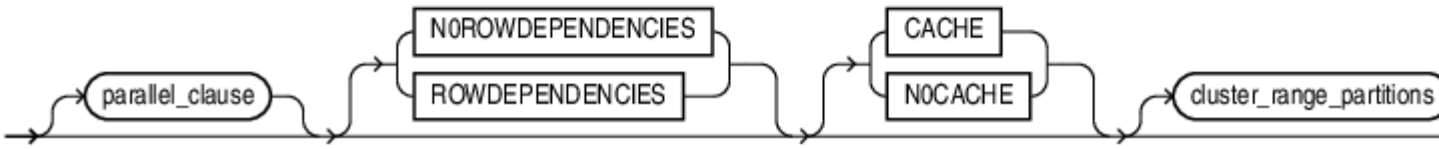
構文

create_cluster::=



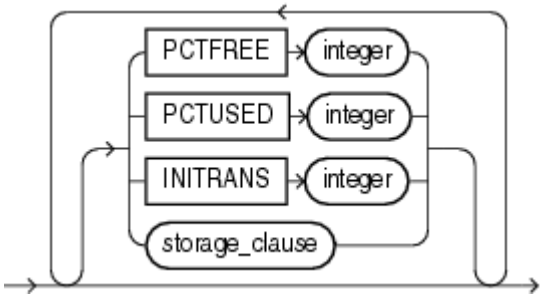






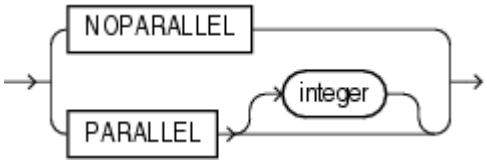
(physical_attributes_clause::=、size_clause::=、cluster_range_partitions::=)

physical_attributes_clause::=

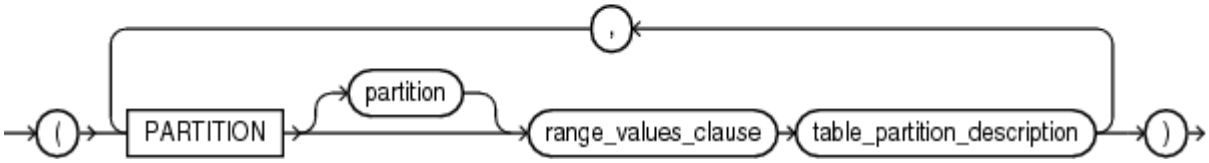


(storage_clause::=)

parallel_clause::=



cluster_range_partitions::=



(range_values_clause::=、table_partition_description::=)

セマンティクス

schema

作成するクラスタが定義されるスキーマを指定します。schemaを指定しない場合、現行のスキーマにクラスタが作成されます。

cluster

作成するクラスタの名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

クラスタを作成した後、そのクラスタに表を追加します。クラスタには、最大32個の表を指定できます。オブジェクト表、およびLOB列またはOracleが提供するAny*型の列を含む表はクラスタの一部にはできません。クラスタを作成し、そのクラスタに表を追加しても、クラスタを意識する必要はありません。クラスタ化されていない表と同様に、SQL文を使用してクラスタ化表にアクセスできます。

関連項目:

クラスタへの表の追加の詳細は、「CREATE TABLE」、「クラスタの作成: 例」および「クラスタへの表の追加: 例」を参照してください。

SHARING

SHARING句は、アプリケーションのメンテナンスの際に、アプリケーション・ルートにクラスタを作成する場合に使用します。このタイプのオブジェクトはアプリケーション共通オブジェクトと呼ばれ、アプリケーション・ルートに属するアプリケーションPDBと共有できます。

次の共有属性のいずれかを使用して、オブジェクトを共有する方法を指定できます。

- METADATA - メタデータ・リンクはメタデータを共有しますが、そのデータは各コンテナに固有です。このタイプのオブジェクトは、メタデータリンク・アプリケーション共通オブジェクトと呼ばれます。
- NONE - オブジェクトは共有されず、アプリケーション・ルートでのみアクセスできます。

column

クラスタ・キーに1つ以上の列名を指定します。最大16個のクラスタ・キー列を指定できます。これらの列は、データ型およびサイズについて、各クラスタ化表の列と対応している必要があります。名前は対応している必要はありません。

クラスタ・キー列の定義の一部として整合性制約は指定できません。そのかわり、クラスタに属している表に整合性制約を対応付けることができます。

関連項目:

「クラスタ・キー: 例」

datatype

各クラスタ・キー列のデータ型を指定します。

クラスタ・データ型の制限事項

クラスタ・データ型には、次の制限事項があります。

- データ型がLONG、LONG RAW、REF、ネストした表、VARRAY、BLOB、CLOB、BFILE、Oracleが提供する型Any*またはユーザー定義オブジェクト型であるクラスタ・キー列は指定できません。
- ROWID型の列を指定することはできますが、それらの列の値が有効な行IDであることは保証されません。

関連項目:

データ型の詳細は、「データ型」を参照してください。

COLLATE

この句を使用すると、クラスタ・キーの文字データ型列にデータ・バインドされた照合を指定できます。

column_collation_nameには、次のように照合を指定します。

- 索引クラスタまたはソートされたハッシュ・クラスタを作成する場合、BINARY、USING_NLS_COMP、USING_NLS_SORTまたはUSING_NLS_SORT_CSのいずれかの照合を指定できます。
- ソートされていないハッシュ・クラスタを作成する場合、任意の有効な名前付き照合または疑似照合を指定できます。

この句を指定しない場合、クラスタ・キー内の列は、クラスタを含むスキーマの、有効なスキーマのデフォルトの照合を継承します。有効なスキーマのデフォルトの照合の詳細は、「ALTER SESSION」のDEFAULT_COLLATION句を参照してください。

クラスタ・キー列の照合は、クラスタ内で作成された表の対応する列の照合と一致する必要があります。

COLLATE句を指定できるのは、COMPATIBLE初期化パラメータが12.2以上に設定され、かつMAX_STRING_SIZE初期化パラメータがEXTENDEDに設定されている場合のみです。

クラスタ・キー列の照合を変更するには、クラスタを再作成する必要があります。

SORT

SORTキーワードは、ハッシュ・クラスタを作成する場合にのみ有効です。表の行はSORTのないクラスタ・キー列のバケットにハッシュされ、この句を使用して、列の各バケット内でソートされます。これによって、クラスタ化データでの後続の問合せ時に、応答時間が短縮される場合があります。

SORT句のないすべての列は、CREATE CLUSTER文のSORT句を持つすべての列の前に指定する必要があります。

ソートされたハッシュ・クラスタの制限事項

ソートされたハッシュ・クラスタでは、行依存性はサポートされません。

関連項目:

- ハッシュ・クラスタの作成の詳細は、「HASHKEYS句」を参照してください。

- 詳細は、ハッシュ・クラスタの管理を参照してください。

physical_attributes_clause

physical_attributes_clauseを使用すると、クラスタの記憶特性を指定できます。クラスタ内の各表もこれらの記憶特性を使用します。これらのパラメータの値を指定しない場合、次のデフォルトが使用されます。

- PCTFREE: 10
- PCTUSED: 40
- INITRANS: 2、またはクラスタを含む表領域のデフォルト値のいずれか大きい方

関連項目:

これらの句の詳細は、「physical_attributes_clause」および「storage_clause」を参照してください。

SIZE

同一クラスタ・キー値または同一ハッシュ値を持つすべての行を格納するために確保する領域を、バイト単位で指定します。次に、この領域によって、データ・ブロックごとに格納されるクラスタやハッシュ値の最大値が決まります。SIZEの値がデータ・ブロック・サイズの約数でない場合、Oracle Databaseは、次に大きい約数を使用します。SIZEがデータ・ブロック・サイズより大きい場合、データベースは、クラスタまたはハッシュ値ごとに、1つ以上のデータ・ブロックを確保し、オペレーティング・システムのブロック・サイズを採用します。

また、データベースは、クラスタ・キー値を持つ行に対して確保する必要がある領域を決定する際に、クラスタ・キーの長さを考慮します。クラスタ・キーが大きければ、それに必要なサイズも大きくなります。実際のサイズを参照するには、USER_CLUSTERSデータ・ディクショナリ・ビューのKEY_SIZE列を問い合せます。(ハッシュ値は実際にはクラスタ内に格納されていないため、この値はハッシュ・クラスタには適用されません。)

このパラメータを指定しない場合、クラスタ・キー値またはハッシュ値ごとにデータ・ブロックが1つ確保されます。

TABLESPACE

クラスタを作成する表領域を指定します。

INDEX句

INDEXを指定すると、索引クラスタを作成できます。索引クラスタには、同一のクラスタ・キー値が指定されている行がまとめて格納されます。それぞれのクラスタ・キー値は、そのキーを持つ表および行の数に関係なく、各データ・ブロックに1回のみ格納されます。INDEXもHASHKEYSも指定しない場合は、デフォルトで索引クラスタが作成されます。

索引クラスタの作成後、クラスタ内の表に対してデータ操作言語(DML)文を発行する前に、そのクラスタ・キーに索引を作成する必要があります。この索引をクラスタ索引と呼びます。

ハッシュ・クラスタに対してクラスタ索引は作成できないため、ハッシュ・クラスタ・キーで索引を作成する必要はありません。

関連項目:

クラスタ索引の作成方法の詳細は「CREATE INDEX」、索引クラスタの概要は『Oracle Database概要』を参照してください。

HASHKEYS句

HASHKEYSを指定すると、ハッシュ・クラスタを作成し、ハッシュ・クラスタのハッシュ値の数を指定できます。ハッシュ・クラスタには、同一のハッシュ・キー値を持つ行がまとめて格納されます。それぞれの行のハッシュ値は、そのクラスタのハッシュ・ファンクションが戻す値です。

Oracle Databaseは、ハッシュ値の実際の数を決めるため、HASHKEYS値を一番近い次の素数に切り上げます。このパラメータの最小値は2です。INDEX句とHASHKEYSパラメータの両方を指定しないとき、デフォルトで索引クラスタが作成されます。

ハッシュ・クラスタの作成時に、データベースは、SIZEパラメータおよびHASHKEYSパラメータの値に基づいて、クラスタに領域を割り当てます。

関連項目:

クラスタに領域を割り当てる方法の詳細は、『Oracle Database概要』および「ハッシュ・クラスタ: 例」を参照してください。

SINGLE TABLE

SINGLE TABLEを指定すると、表を1つのみ持つタイプのハッシュ・クラスタを作成できます。この句によって、表の行へのアクセスがより高速になります。

単一表クラスタの制限事項

クラスタ内で一度に存在できる表は1つのみです。ただし、表を削除して、同一のクラスタに別の表を作成することはできます。

関連項目:

単一表ハッシュ・クラスタ: 例

HASH IS expr

ハッシュ・クラスタのハッシュ・ファンクションとして使用する式を指定します。式には、次の制限事項があります。

- 正の値に評価される必要があります。
- 式全体の値が位取り0(ゼロ)の数になる場合は、任意のデータ型の列が参照される1つ以上の列を持つ必要があります。たとえば、number_column * LENGTH(varchar2_column)です。
- ユーザー定義のPL/SQLファンクションを参照できません。
- 疑似列LEVELまたはROWNUMを参照できません。
- ユーザー関連ファンクション(USERENV、UIDおよびUSER)または日時ファンクション(CURRENT_DATE、CURRENT_TIMESTAMP、DBTIMEZONE、EXTRACT(日時)、FROM_TZ、LOCALTIMESTAMP、NUMTODSINTERVAL、NUMTOYMINTERVAL、SESSIONTIMEZONE、SYSDATE、SYSTIMESTAMP、TO_DSINTERVAL、TO_TIMESTAMP、TO_DATE、TO_TIMESTAMP_TZ、TO_YMINTERVALおよびTZ_OFFSET)を参照できません。
- 定数に評価されることはありません。
- スカラー副問合せ式には指定できません。
- (クラスタ名ではなく)スキーマ名またはオブジェクト名で修飾された列を持つことができません。

HASH IS句を指定しない場合、Oracle Databaseはハッシュ・クラスタに対して内部ハッシュ・ファンクションを使用します。

既存のハッシュ・ファンクションの詳細は、データ・ディクショナリ表USER_CLUSTER_HASH_EXPRESSIONS、ALL_CLUSTER_HASH_EXPRESSIONSおよびDBA_CLUSTER_HASH_EXPRESSIONSに問い合せます。

ハッシュ列のクラスタ・キーは、任意のデータ型で構成される1つ以上の列を持つことができます。複合クラスタ・キー、または整数以外の列で構成されるクラスタ・キーを持つハッシュ・クラスタに対しては、内部ハッシュ・ファンクションを使用する必要があります。

関連項目:

データ・ディクショナリ・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

parallel_clause

parallel_clauseを使用すると、クラスタの作成をパラレル化できます。

この句の詳細は、「CREATE TABLE」の「parallel_clause」を参照してください。

NOROWDEPENDENCIES | ROWDEPENDENCIES

この句のクラスタに対する動作は、表に対する動作と同じです。詳細は、「CREATE TABLE」の「NOROWDEPENDENCIES | ROWDEPENDENCIES」を参照してください。

CACHE | NOCACHE

CACHE

全表スキャンの実行時に、このクラスタに対して取り出されたブロックをバッファ・キャッシュの最低使用頻度リスト(LRU)の最高使用頻度側に配置する場合は、CACHEを指定します。この句は、小規模な参照表で有効です。

NOCACHE

NOCACHEを指定すると、全表スキャンの実行時に、このクラスタに対して取り出されたブロックを、バッファ・キャッシュ内のLRUリストの最低使用頻度側に入れることができます。これはデフォルトの動作です。

NOCACHEは、storage_clauseにKEEPを指定したクラスタには影響しません。

cluster_range_partitions

cluster_range_partitions句を指定すると、レンジ・パーティション・ハッシュ・クラスタを作成できます。この句を指定する場合は、HASHKEYS句も指定する必要があります。

cluster_range_partitions句を使用すると、列リストの値の範囲でクラスタをパーティション化できます。レンジ・パーティション・ハッシュ・クラスタに表を追加すると、この表は、そのクラスタと同じパーティション数で、同じ列の同じパーティション境界に自動的にパーティション化されます。Oracle Databaseは、表パーティションにシステム生成の名前を割り当てます。

文字データ型が指定された各パーティション化キー列には、宣言された照合BINARY、USING_NLS_COMP、USING_NLS_SORTまたはUSING_NLS_SORT_CSのいずれかが必要です。

INTERVAL句を指定できない点を除いて、cluster_range_partitions句は、CREATE TABLEのrange_partitions句と同じセマンティクスを持ちます。詳細は、CREATE TABLEのドキュメントのrange_partitionsを参照してください。

関連項目:

レンジ・パーティション・ハッシュ・クラスタ: 例

例

クラスタの作成: 例

次の文は、クラスタ・キー列department、クラスタ・サイズ512バイトおよび記憶域パラメータ値を指定したクラスタpersonnelを作成します。

```
CREATE CLUSTER personnel
   (department NUMBER(4))
SIZE 512
STORAGE (initial 100K next 50K);
```

クラスタ・キー: 例

次の文は、personnelのクラスタ・キーに対するクラスタ索引を作成します。

```
CREATE INDEX idx_personnel ON CLUSTER personnel;
```

クラスタ索引の作成後、索引に表を追加し、その表に対してDML操作を行うことができます。

クラスタへの表の追加: 例

次の文は、サンプル表hr.employeesから部門表を作成し、前述の例で作成したpersonnelクラスタに追加します。

```
CREATE TABLE dept_10
   CLUSTER personnel (department_id)
   AS SELECT * FROM employees WHERE department_id = 10;
CREATE TABLE dept_20
   CLUSTER personnel (department_id)
   AS SELECT * FROM employees WHERE department_id = 20;
```

ハッシュ・クラスタ: 例

次の文は、クラスタ・キー列cust_language、最大ハッシュ・キー値10 (各サイズ512バイト)および記憶域パラメータ値を指定したハッシュ・クラスタlanguageを作成します。

```
CREATE CLUSTER language (cust_language VARCHAR2(3))
   SIZE 512 HASHKEYS 10
   STORAGE (INITIAL 100k next 50k);
```

この文では、HASH IS句を指定していないため、Oracle Databaseは、そのクラスタに対して内部ハッシュ・ファンクションを採用します。

次の文は、postal_codeとcountry_id列で構成されるクラスタ・キーを持つaddressという名前のハッシュ・クラスタを作成し、これらの列を含むSQL式をハッシュ・ファンクションに使用します。

```
CREATE CLUSTER address
   (postal_code NUMBER, country_id CHAR(2))
   HASHKEYS 20
   HASH IS MOD(postal_code + country_id, 101);
```

単一表ハッシュ・クラスタの例

次の文は、クラスタ・キーcustomer_idおよび最大ハッシュ・キー値100 (各サイズ512バイト)を指定した単一表ハッシュ・クラスタcust_ordersを作成します。

```
CREATE CLUSTER cust_orders (customer_id NUMBER(6))
   SIZE 512 SINGLE TABLE HASHKEYS 100;
```

レンジ・パーティション・ハッシュ・クラスタ: 例

次の文は、販売額を基準にして5つの範囲にパーティション化された、salesというレンジ・パーティション・ハッシュ・クラスタを作成します。クラスタ・キーは、amount_sold列とprod_id列で構成されます。このクラスタは、ハッシュ・ファンクション(amount_sold * 10 + prod_id)を使用して、最大100,000個のハッシュ・キー値を保持します。このキー値には、それぞれ300バイトが割り当てられます。

```
CREATE CLUSTER sales (amount_sold NUMBER, prod_id NUMBER)
  HASHKEYS 100000
  HASH IS (amount_sold * 10 + prod_id)
  SIZE 300
  TABLESPACE example
  PARTITION BY RANGE (amount_sold)
    (PARTITION p1 VALUES LESS THAN (2001),
     PARTITION p2 VALUES LESS THAN (4001),
     PARTITION p3 VALUES LESS THAN (6001),
     PARTITION p4 VALUES LESS THAN (8001),
     PARTITION p5 VALUES LESS THAN (MAXVALUE));
```

クラスタ表の作成: 例

次の文は、デフォルトのキー・サイズ(600)を使用して、emp_deptという名前のクラスタを作成します。

```
CREATE CLUSTER emp_dept (deptno NUMBER(3))
   SIZE 600
   TABLESPACE USERS
   STORAGE (INITIAL 200K
      NEXT 300K
      MINEXTENTS 2
      PCTINCREASE 33);
```

次の文は、emp_deptクラスタの下にdeptという名前のクラスタ表を作成します。

```
CREATE TABLE dept (
   deptno NUMBER(3) PRIMARY KEY)
   CLUSTER emp_dept (deptno);
```

次の文は、emp_deptクラスタの下にemplという名前の別のクラスタ表を作成します。

```
CREATE TABLE empl (
   emplno NUMBER(5) PRIMARY KEY,
   emplname VARCHAR2(15) NOT NULL,
   deptno NUMBER(3) REFERENCES dept)
   CLUSTER emp_dept (deptno);
```

次の文は、emp_deptクラスタに対する索引を作成します。

```
CREATE INDEX emp_dept_index
   ON CLUSTER emp_dept
   TABLESPACE USERS
   STORAGE (INITIAL 50K
      NEXT 50K
      MINEXTENTS 2
      MAXEXTENTS 10
      PCTINCREASE 33);
```

次の文は、USER_CLUSTERSを問い合せてクラスタ・メタデータを表示します。

```
SELECT CLUSTER_NAME, TABLESPACE_NAME, CLUSTER_TYPE, PCT_INCREASE, MIN_EXTENTS,
MAX_EXTENTS FROM USER_CLUSTERS;
CLUSTER_NAME    TABLESPACE CLUST PCT_INCREASE MIN_EXTENTS MAX_EXTENTS
--------------- ---------- ----- ------------ ----------- -----------
EMP_DEPT        USERS      INDEX                       1  2147483645
```

次の文は、USER_CLU_COLUMNSを問い合せてクラスタ・メタデータを表示します。

```
SELECT * FROM USER_CLU_COLUMNS;
CLUSTER_NAME    CLU_COLUMN_NAME      TABLE_NAME TAB_COLUMN_NAME
--------------- -------------------- ---------- --------------------
EMP_DEPT        DEPTNO               DEPT       DEPTNO
EMP_DEPT        DEPTNO               EMPL       DEPTNO
```

次の文は、USER_INDEXESを問い合せて、emp_deptクラスタの索引属性を表示します。

```
SELECT INDEX_NAME, INDEX_TYPE, PCT_INCREASE, MIN_EXTENTS, MAX_EXTENTS FROM
USER_INDEXES WHERE TABLE_NAME='EMP_DEPT';
INDEX_NAME      INDEX_TYPE      PCT_INCREASE MIN_EXTENTS MAX_EXTENTS
--------------- --------------- ------------ ----------- -----------
EMP_DEPT_INDEX  CLUSTER                                1  2147483645
```

CREATE CONTEXT

目的

CREATE CONTEXT文を使用すると、次の操作を実行できます。

- コンテキスト(アプリケーションを検証および保護するアプリケーション定義の一連の属性)のネームスペースを作成します。
- ネームスペースを、コンテキストを設定する外部作成パッケージと関連付けます。

専用パッケージのDBMS_SESSION.SET_CONTEXTプロシージャを使用して、コンテキスト属性を設定または再設定できます。

関連項目:

- コンテキストについては、『Oracle Databaseセキュリティ・ガイド』を参照してください。
- DBMS_SESSION.SET_CONTEXTプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

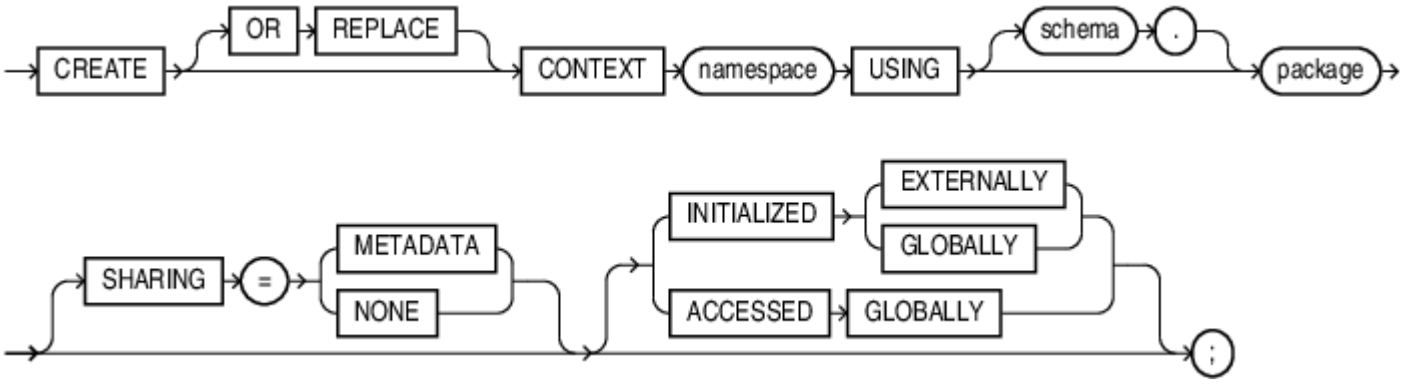
前提条件

コンテキスト・ネームスペースを作成する場合、CREATE ANY CONTEXTシステム権限が必要です。

CREATE CONTEXTコマンドには、パッケージ名のシノニムを使用できないことに注意してください。

構文

create_context::=



セマンティクス

OR REPLACE

OR REPLACEを指定すると、異なるパッケージを使用して既存のコンテキスト・ネームスペースを再定義できます。

namespace

作成または変更するコンテキスト・ネームスペース名を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。コンテキスト・ネームスペースは、常にスキーマSYSに格納されます。

関連項目:

コンテキスト・ネームスペースのネーミングに関するガイドラインは、「データベース・オブジェクトのネーミング規則」を参照してください。

schema

packageを所有するスキーマを指定します。schemaを指定しない場合、Oracle Databaseは現行のスキーマを使用します。

package

ユーザー・セッションのネームスペースに基づくコンテキスト属性を設定または再設定するPL/SQLパッケージを指定します。

柔軟に設計できるように、Oracle Databaseは、コンテキスト作成時のスキーマの存在またはパッケージの妥当性を検証しません。

SHARING

SHARING句は、アプリケーションのメンテナンスの際に、アプリケーション・ルートにオブジェクトを作成する場合に使用します。このタイプのオブジェクトはアプリケーション共通オブジェクトと呼ばれ、アプリケーション・ルートに属するアプリケーションPDBと共有できます。

次の共有属性のいずれかを使用して、オブジェクトを共有する方法を指定できます。

- METADATA - メタデータ・リンクはメタデータを共有しますが、そのデータは各コンテナに固有です。このタイプのオブジェクトは、メタデータリンク・アプリケーション共通オブジェクトと呼ばれます。
- NONE - オブジェクトは共有されず、アプリケーション・ルートでのみアクセスできます。

INITIALIZED句

INITIALIZED句を使用すると、コンテキスト・ネームスペースを初期化するOracle Database以外のエンティティを指定できます。

EXTERNALLY

EXTERNALLYを使用すると、セッション確立時にOCIインタフェースを使用してネームスペースを初期化できます。

関連項目:

OCIを使用したセッション確立の詳細は、『Oracle Call Interfaceプログラマーズ・ガイド』を参照してください。

GLOBALLY

GLOBALLYを指定すると、グローバル・ユーザーがデータベースへ接続するときに、LDAPディレクトリによってネームスペースを初期化できます。

セッション確立後、設定したPL/SQLパッケージのみがネームスペースのすべての属性に書込みを行うコマンドを発行することができます。

関連項目:

グローバルに初期化されるコンテキストの確立の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

ACCESSED GLOBALLY

この句を使用すると、namespaceに設定したすべてのアプリケーション・コンテキストによるインスタンス全体へのアクセスを許可できます。この設定によって、複数のセッションがアプリケーション属性を共有できます。

例

アプリケーション・コンテキストの作成: 例

この例では、PL/SQLパッケージemp_mgmtを使用して、人事管理アプリケーションを検証および保護します。このパッケージを作成する例については、『Oracle Database PL/SQL言語リファレンス』を参照してください。次の文は、コンテキスト・ネームスペースhr_contextを作成し、emp_mgmtパッケージに関連付けます。

```
CREATE CONTEXT hr_context USING emp_mgmt;
```

SYS_CONTEXTファンクションを使用して、このコンテキストに基づいて、データ・アクセスを制御できます。たとえば、emp_mgmtパッケージで、特定の部門識別子として、属性department_idが定義されているとします。次のコマンドを実行し、department_idの値に基づいてアクセスを制限するビューを作成して、employees実表を保護できます。

```
CREATE VIEW hr_org_secure_view AS
   SELECT * FROM employees
   WHERE department_id = SYS_CONTEXT('hr_context', 'department_id');
```

関連項目:

アプリケーション・コンテキストを使用したユーザー情報の取得の詳細は、「SYS_CONTEXT」および『Oracle Databaseセキュリティ・ガイド』を参照してください。

CREATE CONTROLFILE

ノート:

この文を使用する前に、データベース内のすべてのファイルの全体バックアップを実行することをお薦めします。詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

目的

CREATE CONTROLFILE文は、ごく限られた状況でのみ使用します。データベースによって使用されているすべての制御ファイルが失われ、かつバックアップ制御ファイルが存在しない場合に、この文を使用して制御ファイルを再作成します。この文を使用して、REDOログ・ファイル・グループ、REDOログ・ファイル・メンバー、アーカイブREDOログ・ファイル、データファイル、またはデータベースを同時にマウントおよびオープンするインスタンスの最大数を変更することもできます。

データベースの名前を変更するには、CREATE CONTROLFILE文ではなく、DBNEWIDユーティリティを使用することをお薦めします。データベース名の変更後にOPEN RESETLOGS操作が必要ないため、DBNEWIDの方が適しています。

関連項目:

- DBNEWIDユーティリティの詳細は、『Oracle Databaseユーティリティ』を参照してください。
- 既存のデータベース制御ファイルに基づくスクリプトの作成の詳細は、「ALTER DATABASE」の「BACKUP CONTROLFILE句」を参照してください。

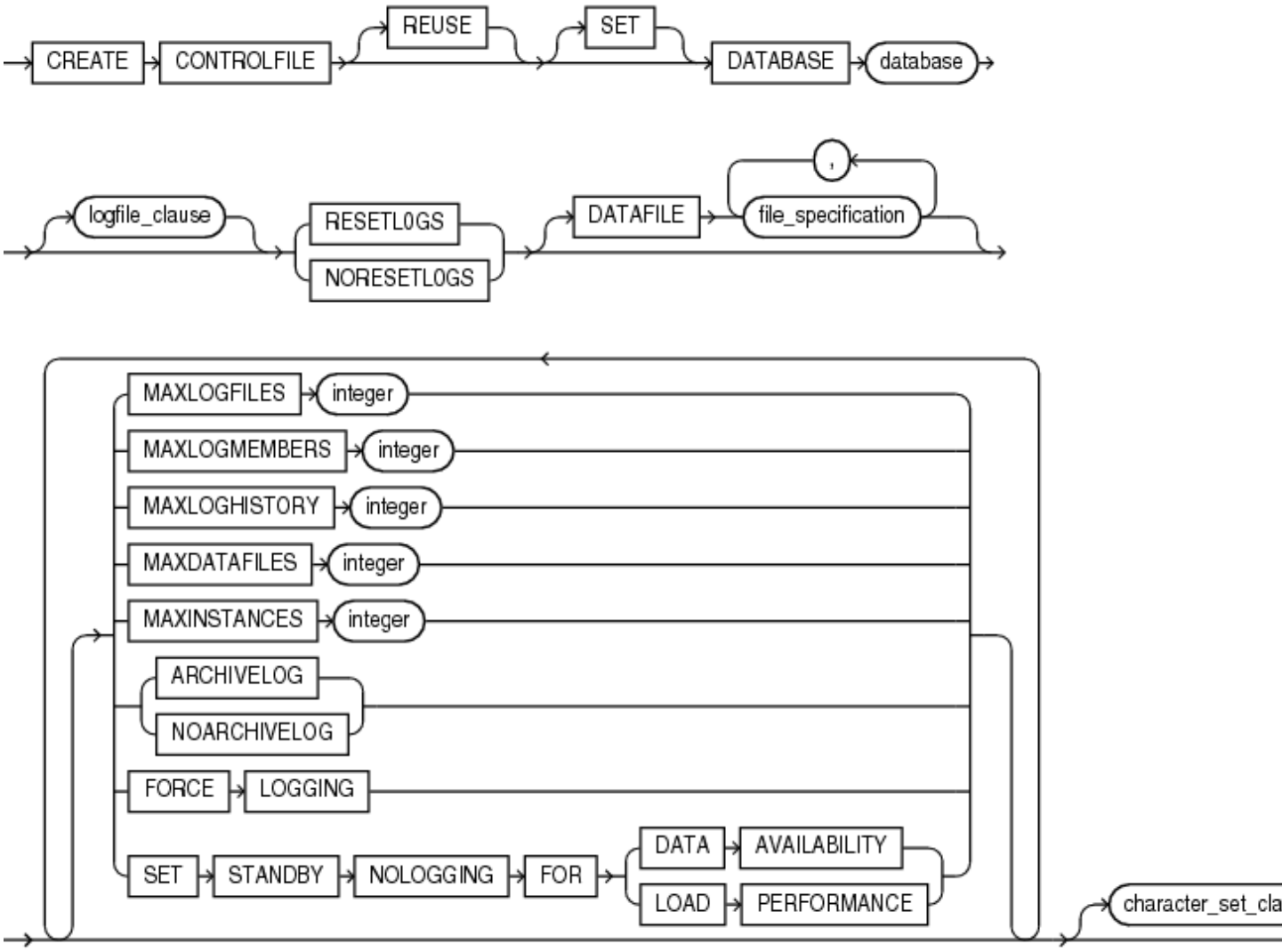
前提条件

制御ファイルを作成する場合は、SYSDBAシステム権限またはSYSBACKUPシステム権限が必要です。

データベースをマウントしているインスタンスがあってはいけません。制御ファイルが正常に作成された後、CLUSTER_DATABASEパラメータで指定したモードでデータベースがマウントされます。続いて、DBAは、データベースをオープンする前にメディア・リカバリを行う必要があります。データベースをOracle Real Application Clusters(Oracle RAC)と併用している場合、次のインスタンスが起動する前に、データベースを停止してSHAREDモード(CLUSTER_DATABASE初期化パラメータの値にTRUEを設定)で再マウントする必要があります。

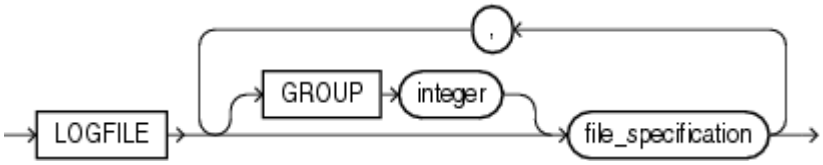
構文

create_controlfile::=



(storage_clause::=)

logfile_clause::=



(file_specification::=)

character_set_clause::=

セマンティクス

CREATE CONTROLFILE文を発行すると、この文に指定した情報に基づいて、新しい制御ファイルが作成されます。制御ファイルは、CONTROL_FILES初期化パラメータで指定した場所に格納されます。このパラメータに値を指定していない場合、Oracle Managed Filesの制御ファイルは、次のいずれか(優先度の高い順に示します)のとおり、制御ファイルのデフォルトの格納先に作成されます。

- DB_CREATE_ONLINE_LOG_DEST_n初期化パラメータで指定されている1つ以上の制御ファイル。最初のディレクトリに作成されたファイルが主制御ファイルになります。DB_CREATE_ONLINE_LOG_DEST_nを指定している場合、DB_CREATE_FILE_DESTまたはDB_RECOVERY_FILE_DEST(高速リカバリ領域)には制御ファイルが作成されません。
- DB_CREATE_ONLINE_LOG_DEST_nに値が指定されていない場合で、DB_CREATE_FILE_DESTおよびDB_RECOVERY_FILE_DESTの両方に値が設定されている場合は、それぞれの場所に1つの制御ファイルが作成されます。DB_CREATE_FILE_DESTに指定された場所が主制御ファイルの場所になります。
- DB_CREATE_FILE_DESTに対してのみ値が指定されている場合は、その場所に1つの制御ファイルが作成されます。
- DB_RECOVERY_FILE_DESTに対してのみ値が指定されている場合は、その場所に1つの制御ファイルが作成されます。

これらのパラメータのいずれにも値を設定していない場合、データベースが稼働しているオペレーティング・システムのデフォルトの場所に制御ファイルが作成されます。この制御ファイルはOracle Managed Fileではありません。

句を指定しない場合、Oracle Databaseは、以前の制御ファイルに対する値ではなく、デフォルト値を使用します。制御ファイルが正常に作成された後、Oracle Databaseは、初期化パラメータCLUSTER_DATABASEで指定したモードでデータベースをマウントします。パラメータが設定されていないときのデフォルト値はFALSEで、その場合はEXCLUSIVEモードでデータベースがマウントされます。その後、インスタンスを停止し、データベースのすべてのファイルの全体バックアップを取ることをお薦めします。

関連項目:

Oracle Databaseバックアップおよびリカバリ・アドバンスト・ユーザーズ・ガイド

REUSE

REUSEを指定すると、初期化パラメータCONTROL_FILESによって特定される既存の制御ファイルを再利用可能にできます。このとき、それらの制御ファイルに現在格納されている情報は上書きされます。この句を指定しないと、既存の制御ファイルがある場合に、エラーが戻されます。

DATABASE句

データベース名を指定します。このパラメータの値は、事前にCREATE DATABASE文またはCREATE CONTROLFILE文で設定した既存のデータベース名である必要があります。

SET DATABASE句

SET DATABASEを使用すると、データベース名を変更できます。データベース名には最大8バイトの名前を指定できます。

この句を指定する場合は、RESETLOGSも指定する必要があります。データベースの名前を変更し、既存のログ・ファイルを保持する場合は、このCREATE CONTROLFILE文を発行した後、ALTER DATABASE RECOVER USING BACKUP CONTROLFILE文を使用して、全データベースのリカバリを完了する必要があります。

logfile_clause

logfile_clauseを使用すると、データベースのREDOログ・ファイルを指定できます。すべてのREDOログ・ファイル・グループのすべてのメンバーを指定する必要があります。

オペレーティング・システムのファイル・システム内の標準REDOログ・ファイル、またはOracle ASMディスク・グループのREDOログ・ファイルをリスト表示するには、file_specificationのredo_log_file_spec書式(「file_specification」を参

照)を使用します。ASM_filenameの書式を使用する場合、redo_log_file_specのautoextend_clauseは指定できません。

この句にRESETLOGSを指定する場合、ASM_filenameのいずれかのファイル作成形式を使用する必要があります。NORESETLOGSを指定する場合、ASM_filenameのいずれかの参照書式を指定する必要があります。

関連項目:

構文の様々な書式は、「ASM_filename」を参照してください。Oracle ASMの使用方法の概要は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

GROUP integer

ログファイル・グループ番号を指定します。GROUP値を指定した場合、データベースが前回オープンされたときのGROUP値を基にして、この値が検証されます。

この句を指定しない場合、システムのデフォルト値を使用してログ・ファイルが作成されます。また、DB_CREATE_ONLINE_LOG_DEST_n初期化パラメータまたはDB_CREATE_FILE_DEST初期化パラメータのいずれかを設定した場合、およびRESETLOGSを指定した場合は、DB_CREATE_ONLINE_LOG_DEST_nパラメータで指定したログ・ファイルのデフォルトの格納先に2つのログが作成されます。このパラメータを設定していない場合は、DB_CREATE_FILE_DESTパラメータで指定した格納先に作成されます。

関連項目:

この句の詳細は、「file_specification」を参照してください。

RESETLOGS

RESETLOGSを指定すると、LOGFILE句にリストされたファイルの内容が無視されるように設定できます。LOGFILE句に指定したファイルは、存在していなくてもかまいません。SET DATABASE句を指定した場合、この句を指定する必要があります。

LOGFILE句の各redo_log_file_specで、SIZEパラメータを指定する必要があります。データベースでは、スレッド1にすべてのオンラインREDOログ・ファイル・グループを割り当てることによって、このスレッドを任意のインスタンスで共通に使用できるようにします。この句を使用した後は、RESETLOGS句を指定したALTER DATABASE文を使用してデータベースをオープンする必要があります。

NORESETLOGS

NORESETLOGSを指定すると、LOGFILE句に指定したすべてのファイルが、前回データベースをオープンしたときの状態で使用されるようになります。これらのファイルは存在している必要があります。また、バックアップからのリストアではなく、現行のオンラインREDOログ・ファイルである必要があります。前回割り当てたスレッドに、REDOログ・ファイル・グループが再度割り当てられ、そのスレッドが前回と同じく再度使用可能になります。

SET DATABASE句を指定してデータベースの名前を変更した場合は、NORESETLOGSを指定することはできません。詳細は、「SET DATABASE句」を参照してください。

DATAFILE句

データベースのデータファイルを指定します。すべてのデータファイルを指定する必要があります。これらのファイルは既存のファイルである必要がありますが、メディア・リカバリを必要とするリストア・バックアップでもかまいません。

DATAFILE句には、読取り専用の表領域のデータファイルを含めないでください。これらのタイプのファイルは、後でデータベース

に追加できます。また、この句には、一時データファイル(一時ファイル)も含めないでください。

オペレーティング・システムのファイル・システム内の標準データファイルと一時ファイル、またはOracle ASMディスク・グループのファイルを指定するには、file_specificationのdatafile_tempfile_spec書式(「file_specification」を参照)を使用します。ASM_filenameの書式を使用する場合、ASM_filenameのいずれかの参照書式を使用する必要があります。構文の様々な書式は、「ASM_filename」を参照してください。

関連項目:

Oracle ASMの使用方法の概要は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

DATAFILEの制限事項

このDATAFILE句では、file_specificationのautoextend_clauseを指定することはできません。

MAXLOGFILES句

データベースに対して作成可能なオンラインREDOログ・ファイル・グループの最大数を指定します。Oracle Databaseは、この値を基にして、制御ファイル内でREDOログ・ファイル名に割り当てる領域の量を決定します。デフォルト値や最大値は、使用するオペレーティング・システムによって異なります。指定する値は、すべてのREDOログ・ファイル・グループのGROUPの最大値以上である必要があります。

MAXLOGMEMBERS句

REDOログ・ファイル・グループのメンバー(同一コピー)の最大数を指定します。Oracle Databaseは、この値を基にして、制御ファイル内でREDOログ・ファイル名に割り当てる領域の量を決定します。最小値は1です。最大値およびデフォルト値は、使用するオペレーティング・システムによって異なります。

MAXLOGHISTORY句

このパラメータは、Oracle DatabaseをARCHIVELOGモードで使用している場合にのみ有効です。データベースの自動メディア・リカバリに必要なアーカイブREDOログ・ファイル・グループの現在見積もられている最大数を指定します。この値を基にして、制御ファイル内でアーカイブREDOログ・ファイル名に割り当てられる領域が決定されます。

最小値は0です。デフォルト値はMAXINSTANCES値の倍数で、使用するオペレーティング・システムによって異なります。最大値は、制御ファイルの最大サイズの制限のみを受けます。必要に応じてデータベースによって制御ファイルの該当セクションに引き続き領域が追加されるため、元の構成で十分でなくなった場合でも制御ファイルを再作成する必要はありません。その結果、このパラメータの実際の値は、指定した値を超える場合があります。

MAXDATAFILES句

CREATE DATABASEまたはCREATE CONTROLFILE実行時の、制御ファイルのデータファイル・セクションの初期サイズを指定します。値がMAXDATAFILESより大きく、DB_FILES以下のファイルを追加した場合、データファイル・セクションにさらに多くのファイルを格納できるように、Oracleの制御ファイルが自動的に拡張されます。

インスタンスでアクセスできるデータファイルの数は、初期化パラメータDB_FILESの制限を受けます。

MAXINSTANCES句

データベースを同時にマウントおよびオープンできるインスタンスの最大数を指定します。この値は、初期化パラメータINSTANCESの値より優先されます。最小値は1です。最大値およびデフォルト値は、使用するオペレーティング・システムによって異なります。

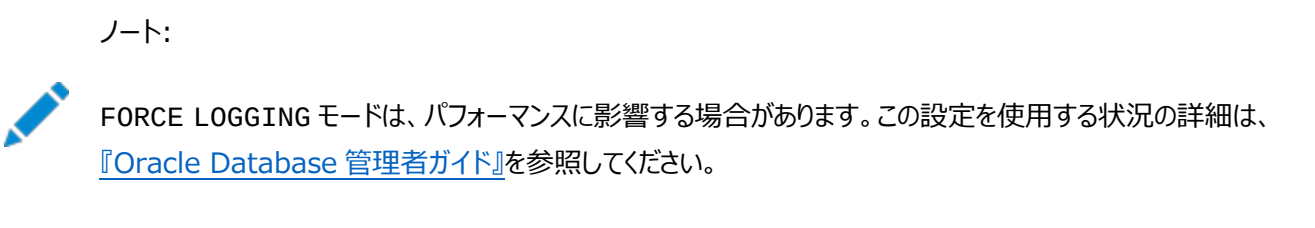
ARCHIVELOG | NOARCHIVELOG

ARCHIVELOGを指定すると、REDOログ・ファイルを再利用する前に、ファイルの内容をアーカイブできます。この句を指定した場合、インスタンスまたはシステム障害リカバリのみでなく、メディア・リカバリも実行できるようになります。

ARCHIVELOG句およびNOARCHIVELOG句を指定しない場合、デフォルトでNOARCHIVELOGモードが選択されます。制御ファイルの作成後に、ALTER DATABASE文を使用して、ARCHIVELOGモードとNOARCHIVELOGモードを切り替えることができます。

FORCE LOGGING

この句を使用すると、制御ファイルの作成後にデータベースをFORCE LOGGINGモードにできます。データベースがこのモードで実行されている場合、一時表領域および一時セグメントへの変更以外のすべてのデータベース内の変更が記録されます。この設定は、各表領域で指定するNOLOGGINGまたはFORCE LOGGING設定、および各データベース・オブジェクトで指定するNOLOGGING設定より優先され、これらの設定には影響されません。この句を指定しない場合、制御ファイルの作成後にデータベースはFORCE LOGGINGモードになりません。



ノート:

FORCE LOGGING モードは、パフォーマンスに影響する場合があります。この設定を使用する状況の詳細は、『Oracle Database 管理者ガイド』を参照してください。

SET STANDBY NOLOGGING FOR DATA AVAILABILITY | LOAD PERFORMANCE

SET STANDBY NOLOGGING

SET STANDBY NOLOGGINGはスタンバイでのロギングを無効にします。これは2種類のモードで指定できます。

- SET STANDBY NOLOGGINGはFOR DATA AVAILABILITYでは、スタンバイ・データベースへの完全なデータ・レプリケーションが保証されます。プライマリ・データベースとスタンバイ・データベースは、ロード中に同期されます。ネットワークの輻輳が発生すると、プライマリ・データベースはそのロードを抑制します。
- SET STANDBY NOLOGGINGはFOR LOAD PERFORMANCEは、プライマリ・データベースのロード速度を維持し、後でスタンバイと同期させます。

SET STANDBY NOLOGGINGの制限事項

SET STANDBY NOLOGGING句はFORCE LOGGINGと同時に使用することはできません。

character_set_clause

文字セットを指定した場合、制御ファイルの文字セット情報が再構成されます。データベースのメディア・リカバリが後で必要となる場合、データベースがオープンする前にこの情報が使用可能になり、リカバリ時に表領域名が正しく解析されます。この句は、デフォルト以外の文字セットを使用している場合にのみ必要です。デフォルトはオペレーティング・システムによって異なります。現行のデータベース文字セットは、起動時に$ORACLE_HOME/logのアラート・ログに出力されます。

表領域のリカバリにRecovery Managerを使用して制御ファイルを再作成する場合、およびデータ・ディクショナリに格納されている文字セットとは異なる文字セットを指定した場合、表領域のリカバリは失敗します。ただし、データベースのオープン時、制御ファイルの文字セットは、データ・ディクショナリの正しい文字セットに更新されます。

この句ではデータベース文字セットを変更できません。

関連項目:

表領域のリカバリの詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

例

制御ファイルの作成: 例

この文は、制御ファイルを再作成します。この文のデータベースdemoは、WE8DEC文字セットで作成されています。この例のpathには、ご使用のシステムでの適切なOracle Databaseディレクトリへのパスを挿入してください。

```
STARTUP NOMOUNT
CREATE CONTROLFILE REUSE DATABASE "demo" NORESETLOGS NOARCHIVELOG
    MAXLOGFILES 32
    MAXLOGMEMBERS 2
    MAXDATAFILES 32
    MAXINSTANCES 1
    MAXLOGHISTORY 449
LOGFILE
  GROUP 1 '/path/oracle/dbs/t_log1.f'  SIZE 500K,
  GROUP 2 '/path/oracle/dbs/t_log2.f'  SIZE 500K
# STANDBY LOGFILE
DATAFILE
  '/path/oracle/dbs/t_db1.f',
  '/path/oracle/dbs/dbu19i.dbf',
  '/path/oracle/dbs/tbs_11.f',
  '/path/oracle/dbs/smundo.dbf',
  '/path/oracle/dbs/demo.dbf'
CHARACTER SET WE8DEC
;
```

CREATE DATABASE

ノート:

この文を実行すると、データベースの初期設定が行われ、指定ファイル内の現行のデータは消去されます。そのことを十分理解したうえで、この文を使用してください。

ノート:

このリリースの Oracle Database と以降のリリースで、デフォルトのデータベース・ユーザー・アカウントのセキュリティを確保するために、セキュリティ機能が拡張されています。このリリースのセキュリティ・チェックリストは、『Oracle Database セキュリティ・ガイド』にあります。このチェックリストを参照し、その内容に従ってデータベースを構成してください。

目的

CREATE DATABASE文を使用すると、汎用的なデータベースを作成できます。

この文を実行すると、データベースの初期使用に備えて、指定した既存のデータファイル上のデータがすべて消去されます。したがって、既存のデータベースに対してこの文を実行した場合、データファイル上のすべてのデータが失われます。

この文を指定した場合、データベースの作成後、データベースは排他モードまたはパラレル・モード(初期化パラメータCLUSTER_DATABASEの値に依存する)でマウントおよびオープンされ、通常の用途に使用可能になります。その後、データベースの表領域を作成できます。

関連項目:

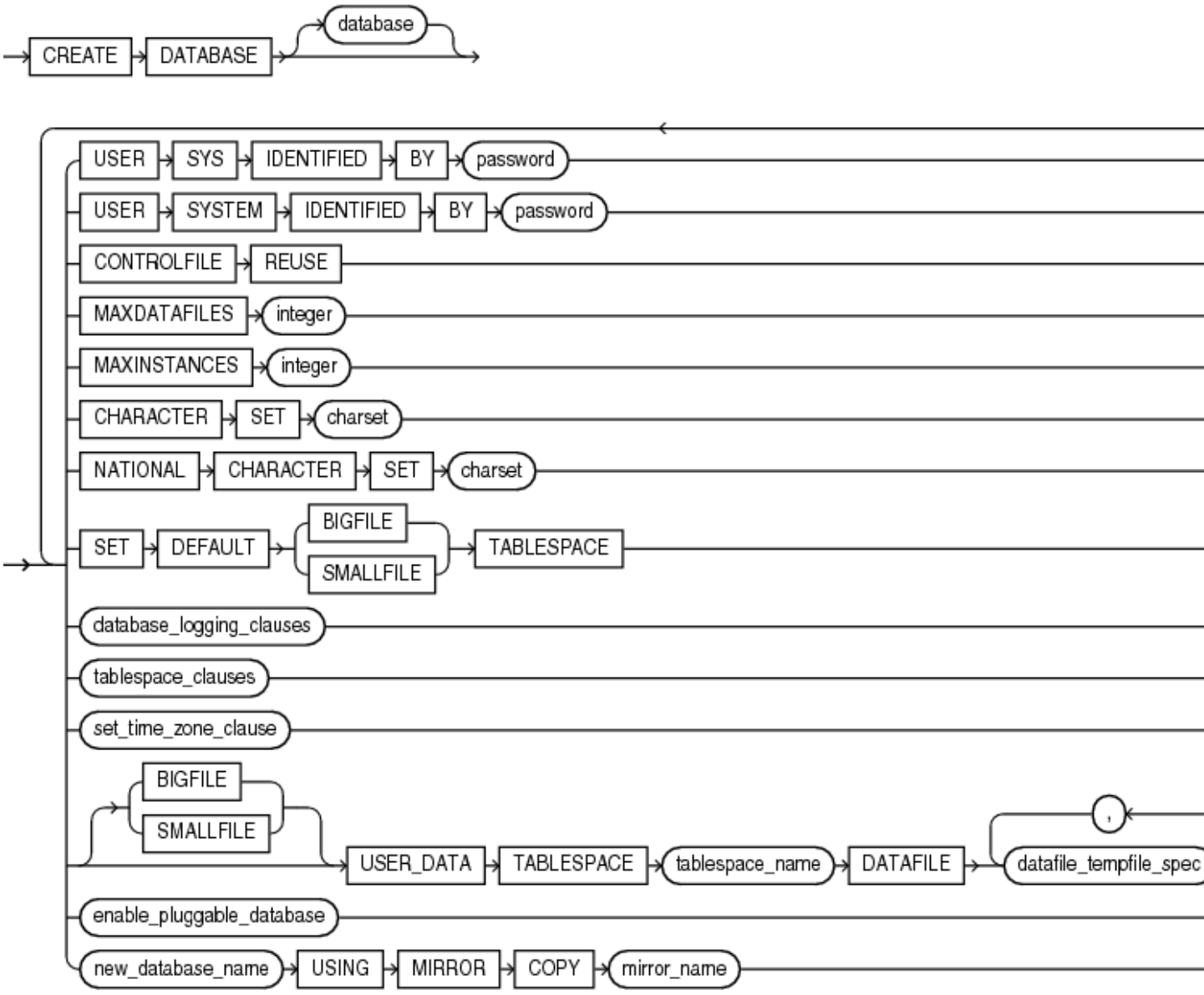
- データベースの変更の詳細は、「ALTER DATABASE」を参照してください。
- Oracle Java Virtual Machineの作成の詳細は、『Oracle Database Java開発者ガイド』を参照してください。
- 表領域の作成については、「CREATE TABLESPACE」を参照してください。

前提条件

データベースを作成する場合は、SYSDBAシステム権限が必要です。作成するデータベースの名前を持つ初期化パラメータ・ファイルが使用可能である必要があります。また、STARTUP NOMOUNTモードになっている必要があります。

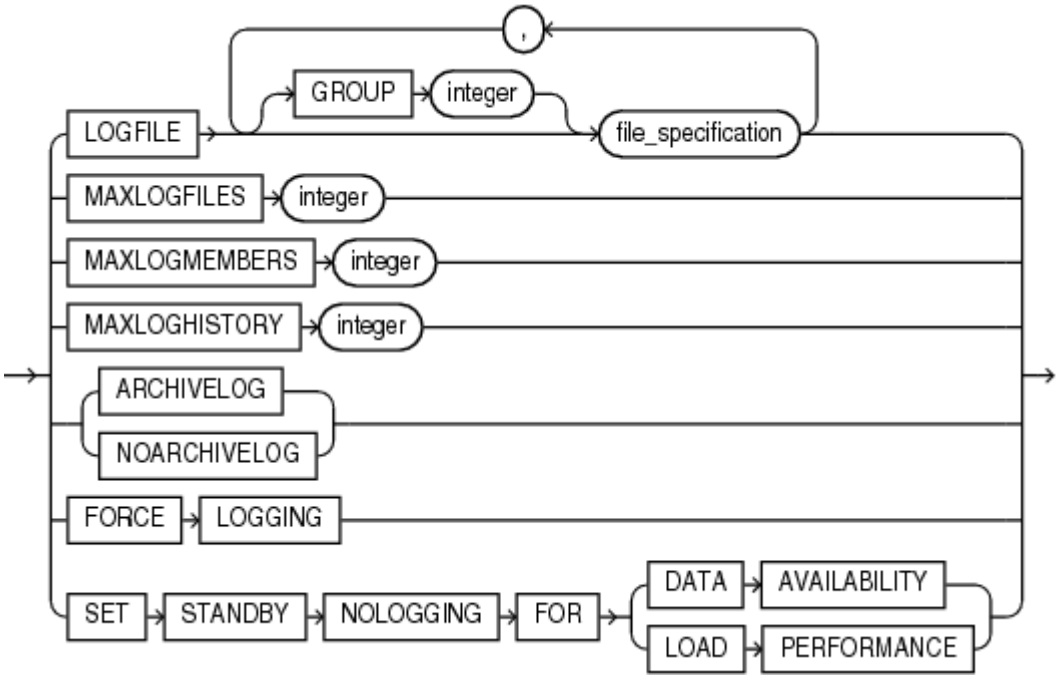
構文

create_database::=



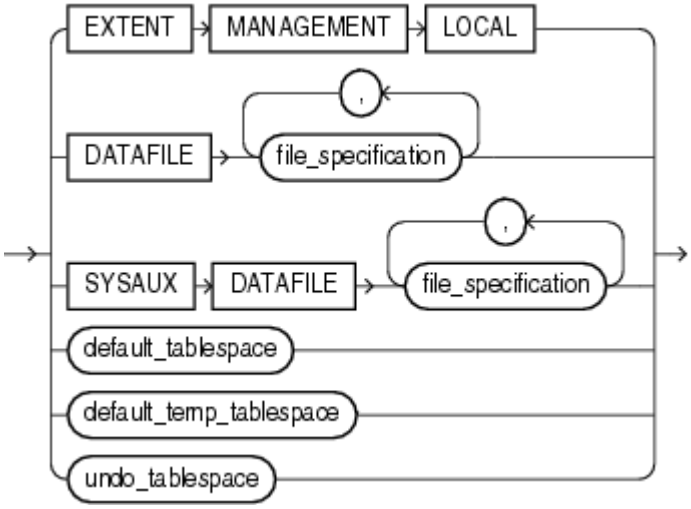
(database_logging_clauses::=、tablespace_clauses::=、set_time_zone_clause::=、datafile_tempfile_spec::=、enable_pluggable_database::=)

database_logging_clauses::=



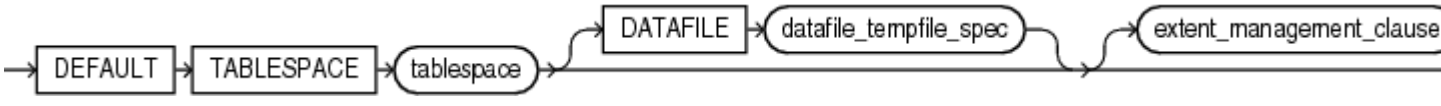
(file_specification::=)

tablespace_clauses::=

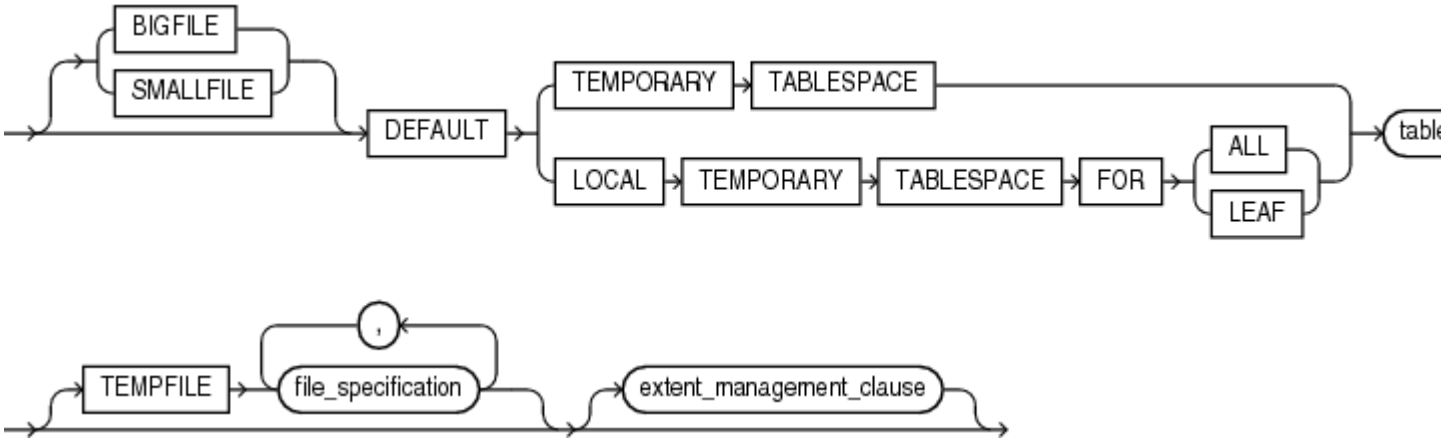


(file_specification::=、default_tablespace::=、default_temp_tablespace::=、undo_tablespace::=、undo_tablespace::=)

default_tablespace::=

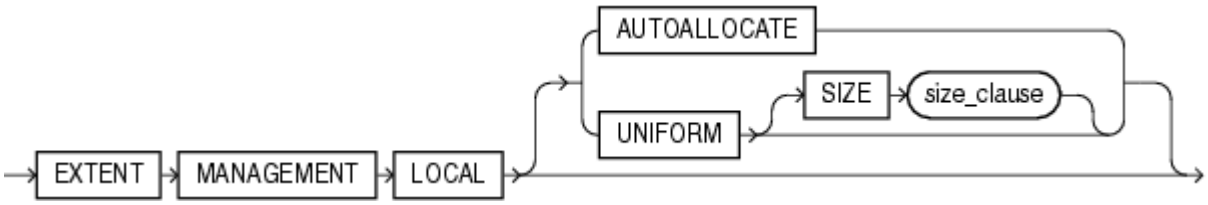


default_temp_tablespace::=



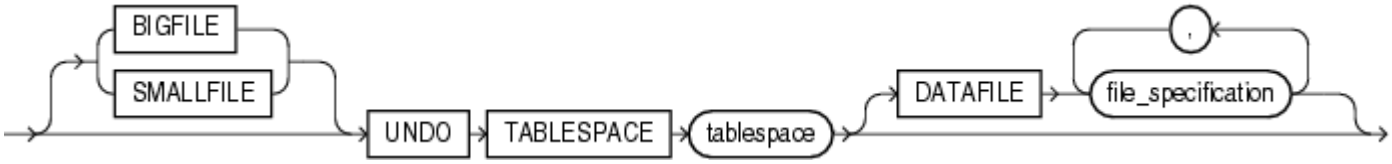
(file_specification::=)

extent_management_clause::=



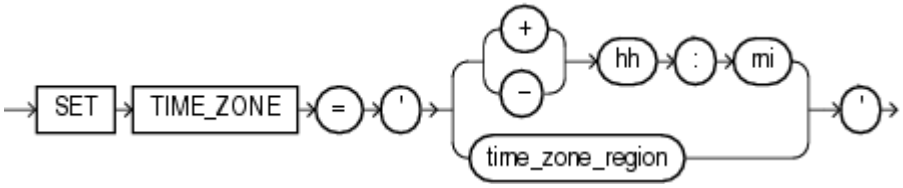
(size_clause::=)

undo_tablespace::=

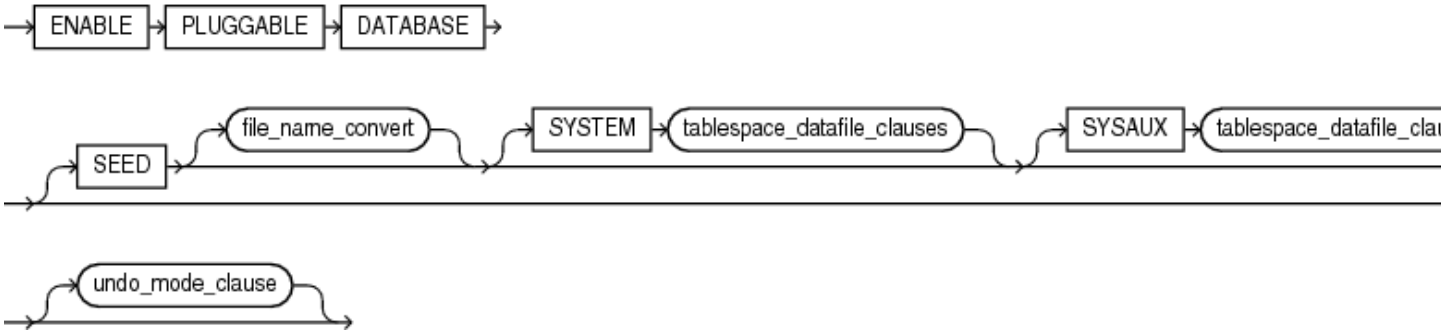


(file_specification::=)

set_time_zone_clause::=

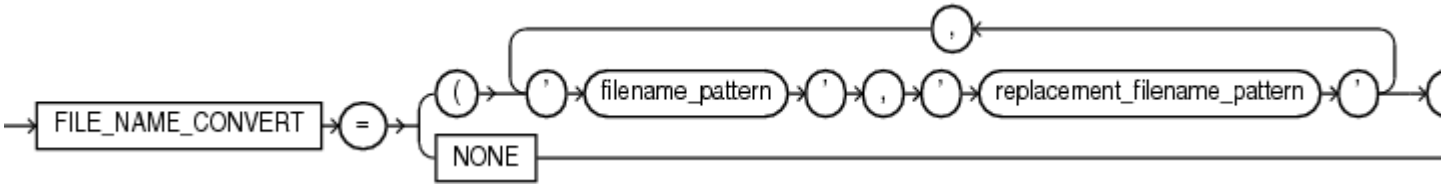


enable_pluggable_database::=

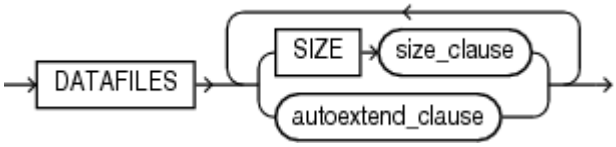


(tablespace_datafile_clauses::=、undo_mode_clause::=)

file_name_convert::=



tablespace_datafile_clauses::=



(size_clause::=、autoextend_clause::=)

undo_mode_clause::=

セマンティクス

database

作成するデータベースの名前を指定します。名前は、DB_NAME初期化パラメータの値と一致する必要があります。名前の長さは最大8バイトです。データベース名には、ASCII文字のみを指定できます。データベース名で有効な文字は、英数字、アンダースコア(_)、シャープ記号(#)およびドル記号($)です。この他の文字は無効です。データベース名はアルファベットで開始する必要があります。Oracle Databaseは、この名前を制御ファイルに書き込みます。後で、データベース名を明示的に指定するALTER DATABASE文を発行すると、制御ファイル内の名前に基づいて、そのデータベース名が検証されます。

データベース名は、大文字と小文字が区別されず、大文字のASCII文字で保存されます。データベース名を引用識別子として指定した場合、引用符は特に警告もなく無視されます。

ノート:

データベース名には、ヨーロッパやアジアの文字セットの中の特殊文字は使用できません。たとえば、ウムラウト付きの文字は使用できません。

CREATE DATABASE文でデータベース名を指定しない場合、DB_NAME初期化パラメータで指定した名前が採用されます。DB_NAME初期化パラメータは、データベースの初期化パラメータ・ファイルに設定する必要があります。そのパラメータの値とは異なる名前を指定した場合、データベースはエラーを戻します。データベース名のその他の規則は、「データベース・オブジェクトのネーミング規則」を参照してください。

USER SYS ...、USER SYSTEM ...

これらの句を使用すると、SYSおよびSYSTEMユーザーのパスワードを設定できます。これらの句は今回のリリースでは必須では

ありません。ただし、いずれか一方の句を指定した場合は、もう一方の句も指定する必要があります。

これらの句を指定しない場合は、Oracle Databaseにより、ユーザーSYSのデフォルト・パスワードchange_on_installが作成されます。このパスワードは、後でALTER USER文を使用して変更できます。

関連項目:

ALTER USER

CONTROLFILE REUSE句

CONTROLFILE REUSEを指定すると、初期化パラメータCONTROL_FILESによって特定される既存の制御ファイルを再利用可能にできます。このとき、それらの制御ファイルに現在格納されている情報は上書きされます。通常、この句は、初めてデータベースを作成する際ではなく、データベースを再作成する際に使用します。データベースを初めて作成する場合、制御ファイルはデフォルトの格納先に作成されます。この場所は、いくつかの初期化パラメータの値によって決まります。「CREATE CONTROLFILE」の「セマンティクス」を参照してください。

制御ファイルを既存のファイルより大きくするためのパラメータ値もあわせて指定する場合、この句は使用できません。MAXLOGFILES、MAXLOGMEMBERS、MAXLOGHISTORY、MAXDATAFILESおよびMAXINSTANCESがこのようなパラメータです。

この句を指定しないと、CONTROL_FILESで指定した制御ファイルのいずれかがすでに存在する場合、エラーが戻ります。

MAXDATAFILES句

CREATE DATABASEまたはCREATE CONTROLFILE実行時の、制御ファイルのデータファイル・セクションの初期サイズを指定します。値がMAXDATAFILESより大きく、DB_FILES以下のファイルを追加した場合、データファイル・セクションにさらに多くのファイルを格納できるように、Oracle Databaseの制御ファイルが自動的に拡張されます。

インスタンスでアクセスできるデータファイルの数は、初期化パラメータDB_FILESの制限を受けます。

MAXINSTANCES句

データベースを同時にマウントおよびオープンできるインスタンスの最大数を指定します。この値は、初期化パラメータINSTANCESの値より優先されます。最小値は1です。最大値は1055です。デフォルトは、使用するオペレーティング・システムによって異なります。

CHARACTER SET句

データベースにデータを格納するときの文字セットを指定します。サポートされている文字セットおよびこのパラメータのデフォルト値は、使用するオペレーティング・システムによって異なります。

CHARACTER SETの制限事項

AL16UTF16文字セットをデータベース文字セットとして指定することはできません。

関連項目:

文字セットの選択の詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

NATIONAL CHARACTER SET句

データ型がNCHAR、NCLOBまたはNVARCHAR2として定義された列にデータを格納する際に使用する各国語文字セットを指定します。有効な値は、AL16UTF16およびUTF8です。デフォルトはAL16UTF16です。

関連項目:

Unicodeデータ型のサポートについては、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

SET DEFAULT TABLESPACE句

この句を使用すると、以降に作成される表領域、およびSYSTEMとSYSAUX表領域のデフォルトの型を指定できます。以降に作成される表領域のデフォルトの型をbigfileまたはsmallfileに設定するには、BIGFILEまたはSMALLFILEを指定します。

- bigfile表領域に格納されるのは、1つのデータファイルまたは一時ファイルのみであり、このファイルには最大約40億(2^{32})ブロックを格納できます。データファイルまたは一時ファイル1つ当たりの最大サイズは、32Kブロックの表領域の場合は128TB、8Kブロックの表領域の場合は32TBです。
- smallfile表領域は、Oracleの従来の表領域であり、1022のデータファイルまたは一時ファイルを含めることができます。それぞれのファイルは、最大で約400万(2^{22})のブロックを格納できます。

この句を指定しない場合、デフォルトでsmallfile表領域が作成されます。

関連項目:

- bigfile表領域の詳細は、『Oracle Database管理者ガイド』を参照してください。
- この構文の使用例は、「表領域のデフォルト・タイプの設定: 例」を参照してください。

database_logging_clauses

database_logging_clausesを使用すると、データベースのREDOログ・ファイルの処理方法を指定できます。

LOGFILE句

REDOログ・ファイルとして使用する1つ以上のファイルを指定します。オペレーティング・システムのファイル・システム内の標準REDOログ・ファイル、またはOracle ASMディスク・グループのREDOログ・ファイルを作成するには、file_specificationのredo_log_file_spec書式を使用します。ASM_filenameの書式を使用する場合、redo_log_file_specのautoextend_clauseは指定できません。

redo_log_file_spec句には、1つ以上のREDOログ・ファイルのメンバー(コピー)を含むREDOログ・ファイル・グループを指定します。CREATE DATABASE文に指定したすべてのREDOログ・ファイルは、REDOログのスレッド番号1に追加されます。

関連項目:

この句の詳細は、「file_specification」を参照してください。

LOGFILE句を指定しない場合、Oracle Managed Filesのログ・ファイル・メンバーは、(優先度の高い順に示している)次のいずれかのとおり、デフォルトの格納先に作成されます。

- DB_CREATE_ONLINE_LOG_DEST_nを設定している場合、指定した各ディレクトリに、MAXLOGMEMBERS初期化パラメータの値までログ・ファイル・メンバーが作成されます。
- DB_CREATE_ONLINE_LOG_DEST_nパラメータを設定しておらず、DB_CREATE_FILE_DESTおよびDB_RECOVERY_FILE_DEST初期化パラメータの両方を設定している場合、それぞれの場所に1つのOracle Managed Filesのログ・ファイル・メンバーが作成されます。DB_CREATE_FILE_DESTアーカイブ先のログ・ファイル

が最初のメンバーです。

- DB_CREATE_FILE_DEST初期化パラメータのみを指定している場合、その場所にログ・ファイル・メンバーが作成されます。
- DB_RECOVERY_FILE_DEST初期化パラメータのみを指定している場合、その場所にログ・ファイル・メンバーが作成されます。

いずれの場合でも、パラメータ設定では、オペレーティング・システムのファイル名または作成書式Oracle ASMのファイル名を正確に指定する必要があります。

これらのパラメータのいずれにも値を設定していない場合、データベースが稼働しているオペレーティング・システムのデフォルトの場所にログ・ファイルが作成されます。このログ・ファイルはOracle Managed Fileではありません。

GROUP integer

REDOログ・ファイル・グループを識別する番号を指定します。integerの値は1からMAXLOGFILESパラメータの値の範囲です。データベースには、2つ以上のREDOログ・ファイル・グループが必要です。同一のGROUP値を持つREDOログ・ファイル・グループは複数指定できません。このパラメータを指定しない場合、値が自動的に生成されます。REDOログ・ファイル・グループのGROUP値は、動的パフォーマンス・ビューV$LOGで確認できます。

MAXLOGFILES句

データベースに対して作成可能なREDOログ・ファイル・グループの最大数を指定します。Oracle Databaseは、この値を基にして、制御ファイル内でREDOログ・ファイル名に割り当てる領域の量を決定します。デフォルト値、最小値および最大値は、使用するオペレーティング・システムによって異なります。

MAXLOGMEMBERS句

REDOログ・ファイル・グループのメンバー(コピー)の最大数を指定します。Oracle Databaseは、この値を基にして、制御ファイル内でREDOログ・ファイル名に割り当てる領域の量を決定します。最小値は1です。最大値およびデフォルト値は、使用するオペレーティング・システムによって異なります。

MAXLOGHISTORY句

このパラメータが有用なのは、Oracle Real Application Clusters(Oracle RAC)でOracle DatabaseをARCHIVELOGモードで使用している場合のみです。Oracle RACの自動メディア・リカバリに使用するアーカイブREDOログ・ファイルの最大数を指定します。この値を基にして、制御ファイル内でアーカイブREDOログ・ファイル名に割り当てられる領域が決定されます。最小値は0です。デフォルト値はMAXINSTANCES値の倍数で、使用するオペレーティング・システムによって異なります。最大値は、制御ファイルの最大サイズの制限のみを受けます。

ARCHIVELOG

ARCHIVELOGを指定すると、REDOログ・ファイル・グループを再利用する前に、グループの内容をアーカイブできます。この句を指定すると、メディア・リカバリを実行できるようになります。

NOARCHIVELOG

NOARCHIVELOGを指定すると、REDOログ・ファイル・グループを再利用する前に、グループの内容がアーカイブされません。この句を指定した場合、メディア・リカバリは実行できません。

デフォルトはNOARCHIVELOGモードです。データベースの作成後に、ALTER DATABASE文を使用して、ARCHIVELOGモードとNOARCHIVELOGモードを切り替えることができます。

FORCE LOGGING

この句を使用すると、データベースをFORCE LOGGINGモードにできます。一時表領域および一時セグメントへの変更以外のすべてのデータベース内の変更が記録されます。この設定は、各表領域で指定するNOLOGGINGまたはFORCE LOGGING設定、および各データベース・オブジェクトで指定するNOLOGGING設定より優先され、これらの設定には影響されません。

FORCE LOGGINGモードは、データベースのインスタンスで永続的です。データベースを停止し、再起動しても、データベースはFORCE LOGGINGモードのままです。ただし、制御ファイルを再作成した場合は、CREATE CONTROLFILE文でFORCE LOGGINGを指定しないかぎり、データベースはFORCE LOGGINGモードではなくなります。

ノート:

FORCE LOGGING モードは、パフォーマンスに影響する場合があります。この設定を使用する状況の詳細は、『Oracle Database 管理者ガイド』を参照してください。

関連項目:

CREATE CONTROLFILE

SET STANDBY NOLOGGING FOR DATA AVAILABILITY | LOAD PERFORMANCE

SET STANDBY NOLOGGINGはスタンバイでのロギングを無効にします。これは2種類のモードで指定できます。

- SET STANDBY NOLOGGINGはFOR DATA AVAILABILITYでは、スタンバイ・データベースへの完全なデータ・レプリケーションが保証されます。プライマリ・データベースとスタンバイ・データベースは、ロード中に同期されます。ネットワークの輻輳が発生すると、プライマリ・データベースはそのロードを抑制します。
- SET STANDBY NOLOGGINGはFOR LOAD PERFORMANCEは、プライマリ・データベースのロード速度を維持し、後でスタンバイと同期させます。

SET STANDBY NOLOGGINGの制限事項

SET STANDBY NOLOGGING句はFORCE LOGGINGと同時に使用することはできません。

tablespace_clauses

tablespace_clausesを使用すると、SYSTEMおよびSYSAUX表領域を構成したり、デフォルトの一時表領域とUNDO表領域を指定することができます。

extent_management_clause

この句を使用すると、ローカル管理SYSTEM表領域を作成できます。この句を指定しない場合、SYSTEM表領域はディクショナリ管理となります。

ノート:

ローカル管理 SYSTEM 表領域を作成すると、この表領域をディクショナリ管理に変更することはできません。このデータベース内に別のディクショナリ管理表領域を作成することもできません。

この句を指定した場合、ローカル管理のSYSTEM表領域には一時セグメントを格納できないため、データベースにデフォルトの一時表領域が必要になります。

- EXTENT MANAGEMENT LOCALを指定して、DATAFILE句を指定しない場合は、default_temp_tablespace句を省略できます。Oracle Databaseは、データファイル・サイズが10MBで、自動拡張を使用禁止にした状態で、TEMPという一時表領域を作成します。
- EXTENT MANAGEMENT LOCALおよびDATAFILE句の両方を指定する場合は、default_temp_tablespace句も指定し、その一時表領域の一時ファイルを明示的に指定する必要があります。

インスタンスを自動UNDOモードでオープンしている場合も、データベースUNDO表領域には同様の要件があります。

- EXTENT MANAGEMENT LOCALを指定して、DATAFILE句を指定しない場合は、undo_tablespace句を省略できます。Oracle Databaseは、SYS_UNDOTBSというUNDO表領域を作成します。
- EXTENT MANAGEMENT LOCALおよびDATAFILEの両方の句を指定する場合は、undo_tablespaceを指定し、その表領域のデータファイルを明示的に指定する必要があります。

関連項目:

ローカル管理表領域およびディクショナリ管理表領域の詳細は、『Oracle Database管理者ガイド』を参照してください。

DATAFILE句

データファイルとして使用する1つ以上のファイルを指定します。ファイルは、すべてSYSTEM表領域の一部となります。オペレーティング・システムのファイル・システム内の標準データファイルと一時ファイル、またはOracle ASMディスク・グループのファイルを作成するには、file_specificationのfile_tempfile_spec書式を使用します。

ノート:

undo_tablespace 句の DATAFILE 句と同様に、この句はオプションです。したがって、あいまいさを回避するために、この句で SYSTEM 表領域のデータファイルを指定する場合は、オプションの DATAFILE 句を含まない undo_tablespace 句の直後にこの句を指定しないでください。指定した場合、Oracle Database は DATAFILE 句を undo_tablespace 句の一部と認識します。

SYSTEM表領域のデータファイルを指定する構文は、Oracle ASMを使用してファイルを格納する場合も、ファイル・システムに格納する場合も、表領域の作成時にCREATE TABLESPACE文を使用してデータファイルを指定する構文と同じです。

関連項目:

データファイルの指定の詳細は、「CREATE TABLESPACE」を参照してください。

自動UNDOモードでデータベースを実行し、SYSTEM表領域のデータファイル名を指定した場合、すべての表領域に対してデータファイルを生成するものとみなされます。Oracle Managed Filesを使用する場合(DB_CREATE_FILE_DEST初期化パラメータに値を設定している場合)、データファイルは自動的に生成されます。ただし、Oracle Managed Filesを使用しないでこの句を指定する場合は、undo_tablespace句およびdefault_temp_tablespace句も指定する必要があります。

この句を指定しない場合、次のようになります。

- DB_CREATE_FILE_DEST初期化パラメータを設定している場合、このパラメータで指定したデフォルトのファイル格納先に、サイズが100MBでシステムが生成する名前を持つ、Oracle Managed Filesのデータファイルが作成されます。
- DB_CREATE_FILE_DEST初期化パラメータを設定していない場合、1つのデータファイルが作成されます。そのファイル名およびサイズは、使用するオペレーティング・システムによって異なります。

関連項目:

構文の詳細は、「file_specification」を参照してください。

SYSAUX句

Oracle Databaseは、すべてのデータベースに、SYSTEMとSYSAUXの両方の表領域を作成します。Oracle Managed Filesを使用しておらず、SYSAUX表領域の1つ以上のデータファイルを指定する場合、この句を使用します。

DATAFILE句を使用してSYSTEM表領域の1つ以上のデータファイルを指定した場合は、この句を指定する必要があります。Oracle Managed Filesを使用している場合、この句を指定しないと、SYSAUXデータファイルがOracle Managed Filesに設定されたデフォルトの場所に作成されます。

Oracle Managed Filesを使用可能にした場合にSYSAUX句を省略すると、SYSAUX表領域が、オンライン表領域、永続表領域およびローカル管理表領域として作成されます。この表領域は、100MBの1つのデータファイルを持ち、ロギングおよび自動セグメント領域管理が有効になっています。

SYSAUX表領域のデータファイルを指定する構文は、Oracle ASMを使用してファイルを格納する場合も、ファイル・システムに格納する場合も、表領域の作成時にCREATE TABLESPACE文を使用してデータファイルを指定する構文と同じです。

関連項目:

- データベースのアップグレード時のSYSAUX表領域の作成方法、および表領域のデータファイルの指定方法は、「CREATE TABLESPACE」を参照してください。
- SYSAUX表領域作成の詳細は、『Oracle Database管理者ガイド』を参照してください。

default_tablespace

この句を指定すると、データベースのデフォルトの永続表領域を作成できます。Oracle Databaseは、smallfile表領域を作成し、その後、この表領域に対して、別の永続表領域を指定していない、SYSTEM以外のユーザーを割り当てます。この句を指定しない場合、SYSTEM以外のユーザーに対するデフォルトの永続表領域はSYSTEM表領域です。

DATAFILE句およびextent_management_clauseは、CREATE TABLESPACE文で同じセマンティクスを持ちます。これらの句の詳細は、「DATAFILE | TEMPFILE句」および「extent_management_clause」を参照してください。

default_temp_tablespace

この句を使用すると、デフォルトの共有一時表領域またはデフォルトのローカル一時表領域を作成できます。これらの一時表領域には、別の一時表領域を指定していないユーザーが割り当てられます。

- データベースにデフォルトの共有一時表領域を作成するには、DEFAULT TEMPORARY TABLESPACEを指定します。共有一時表領域はOracle Databaseの以前のリリースでも使用可能で、「一時表領域」と呼ばれていました。このガイドの他の部分では、特に記載がないかぎり、一時表領域という用語は共有一時表領域を指します。この句を指定しないと、ローカル管理のSYSTEM表領域の作成プロセスでデフォルトの共有一時表領域が自動的に作成され

ない場合、SYSTEM表領域がデフォルトの共有一時表領域になります。

- Oracle Database 12cリリース2 (12.2)から、DEFAULT LOCAL TEMPORARY TABLESPACEを指定して、デフォルトのローカル一時表領域を作成できるようになりました。ローカル一時表領域は、Oracle Real Application ClustersおよびOracle Flexクラスタに役立ちます。これらには、各データベース・インスタンスの共有されない個別の一時ファイルが格納され、I/Oパフォーマンスが向上します。ローカル一時表領域は、BIGFILE表領域である必要があります。
  - HUBとLEAFのすべてのノードに、個別の非共有一時ファイルを作成するようデータベースに指示するには、FOR ALLを指定します。
  - LEAFノードにのみ個別の非共有一時ファイルを作成するようデータベースに指示するには、FOR LEAFを指定します。この句を指定すると、HUBノードでデフォルトの共有一時表領域が使用されます。HUBとLEAFの両方のノードにわたるSQL操作の場合、HUBノードではデフォルトの共有一時表領域が使用され、LEAFノードではデフォルトのローカル一時表領域が使用されます。

  ローカル一時表領域を作成しない場合、HUBとLEAFの各ノードで、デフォルトの共有一時表領域が使用されます。

デフォルトの一時表領域がbigfileかsmallfileかを指定するには、BIGFILEまたはSMALLFILEを指定します。これらの句のセマンティクスは、SET DEFAULT TABLESPACE句のセマンティクスと同じです。

DB_CREATE_FILE_DEST初期化パラメータを設定してOracle Managed Filesを使用可能にした場合、この句のTEMPFILE句の部分はオプションです。このパラメータの値を指定していない場合は、TEMPFILE句を指定する必要があります。BIGFILEを指定した場合、この句で指定できるのは1つの一時ファイルのみです。

デフォルトの一時表領域の一時ファイルを指定する構文は、Oracle ASMを使用してファイルを格納する場合も、ファイル・システムに格納する場合も、一時表領域の作成時にCREATE TABLESPACE文を使用して一時ファイルを指定する構文と同じです。

extent_management_clause句のセマンティクスは、CREATE DATABASEおよびCREATE TABLESPACE文で同じです。この句の詳細は、「CREATE TABLESPACE」の「extent_management_clause」を参照してください。

関連項目:

一時ファイルの指定の詳細は、「CREATE TABLESPACE」を参照してください。

ノート:

オペレーティング・システムによっては、一時ファイルのブロックが実際にアクセスされるまで、一時ファイル用の領域が割り当てられない場合があります。領域の割当ての遅延のため、一時ファイルの作成およびサイズ変更が速くなります。ただし、後で一時ファイルが使用されるときに、十分なディスク領域を使用可能にする必要があります。問題を回避するには、一時ファイルの作成またはサイズ変更の前に、ディスクの空き領域が、新しく作成する一時ファイルまたはサイズ変更後の一時ファイルのサイズよりも大きいことを確認してください。ディスク領域に余裕を持たせておくと、関連のない操作による、予期されるディスク使用量の増加にも対応できます。その後で、作成またはサイズ変更操作を実行してください。

デフォルトの一時表領域の制限事項

デフォルトの一時表領域には、次の制限事項があります。

- この句では、SYSTEM表領域を指定できません。
- デフォルトの一時表領域は、標準的なブロック・サイズである必要があります。

undo_tablespace

インスタンスを自動UNDOモードでオープンした場合(UNDO_MANAGEMENT初期化パラメータをデフォルトのAUTOに設定した場合)、undo_tablespaceを指定して、UNDOデータで使用する表領域を作成できます。自動UNDOモードを使用することをお薦めします。ただし、UNDO領域管理をロールバック・セグメントによって処理する場合、この句は指定しないでください。UNDO_TABLESPACE初期化パラメータの値を設定した場合も、この句を省略できます。パラメータが設定されており、この句を指定した場合、tablespaceはそのパラメータ値と同じである必要があります。

- UNDO表領域をbigfile表領域にする場合は、BIGFILEを指定します。bigfile表領域には、1つのデータファイルのみが含まれます。このファイルの最大サイズは8EB(8,000,000TB)です。
- UNDO表領域をsmallfile表領域にする場合は、SMALLFILEを指定します。smallfile表領域は、Oracle Databaseの従来の表領域であり、1022のデータファイルまたは一時ファイルを含めることができます。それぞれのファイルは、最大で約400万(2^{22})のブロックを格納できます。
- DB_CREATE_FILE_DEST初期化パラメータを設定してOracle Managed Filesを使用可能にした場合、この句のDATAFILE句の部分はオプションです。このパラメータの値を指定していない場合は、DATAFILE句を指定する必要があります。BIGFILEを指定した場合、この句で指定できるのは1つのデータファイルのみです。

UNDO表領域のデータファイルを指定する構文は、Oracle ASMを使用してファイルを格納する場合も、ファイル・システムに格納する場合も、表領域の作成時にCREATE TABLESPACE文を使用してデータファイルを指定する構文と同じです。

関連項目:

データファイルの指定の詳細は、「CREATE TABLESPACE」を参照してください。

この句を指定すると、tablespaceという名前のUNDO表領域、およびUNDO表領域の一部として指定したデータファイルが作成され、インスタンスのUNDO表領域としてこの表領域が割り当てられます。Oracle Databaseは、このUNDO表領域を使用してUNDOデータを管理します。この句のDATAFILE句は、「DATAFILE句」で説明されているように動作します。

初期化パラメータ・ファイル内のUNDO_TABLESPACE初期化パラメータの値を指定している場合は、データベースをマウントする前に、この句で同じ名前を指定する必要があります。この名前が異なると、データベースのオープン時にエラーが戻されます。

この句を指定しないと、SYS_UNDOTBSという名前のデフォルトのsmallfileのUNDO表領域を持つデフォルトのデータベースが作成され、インスタンスのUNDO表領域としてこのデフォルトの表領域が割り当てられます。このUNDO表領域は、CREATE DATABASE文で使用するデフォルトのファイルから、初期エクステントが10MBのディスク領域を割り当てます。システムが生成したデータファイルは、「DATAFILE句」で説明されているように処理されます。Oracle DatabaseがUNDO表領域を作成できない場合、CREATE DATABASE操作全体が失敗します。

関連項目:

- 自動UNDO管理およびUNDO表領域の詳細は、『Oracle Database管理者ガイド』を参照してください。
- データベースの作成後にUNDO表領域を作成する方法は、「CREATE TABLESPACE」を参照してください。

set_time_zone_clause

SET TIME_ZONE句を使用すると、データベースのタイムゾーンを設定できます。次の2つ方法でタイムゾーンを設定します。

- UTC(協定世界時、以前のグリニッジ標準時)からの時差を指定。hh:miの有効範囲は、-12:00から+14:00です。
- タイムゾーン地域を指定。有効なタイムゾーン地域名を表示するには、V$TIMEZONE_NAMES動的パフォーマンス・ビューのTZNAME列を問い合せます。

ノート:

データベースのタイムゾーンを UTC(0:00)に設定することをお薦めします。これによって、タイムゾーンの変換が不要になるため、特にデータベース間のパフォーマンスを向上できます。

関連項目:

動的パフォーマンス・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

すべてのTIMESTAMP WITH LOCAL TIME ZONEデータは、ディスクに格納されるときにデータベースのタイムゾーンに正規化されます。SET TIME_ZONE句を指定しない場合、サーバーのオペレーティング・システムのタイムゾーンが使用されます。オペレーティング・システムのタイムゾーンがOracle Databaseで有効でない場合、データベースのタイムゾーンは、デフォルトでUTCになります。

USER_DATA TABLESPACE句

この句では、ユーザー・データおよびデータベース・オプション(Oracle XML DBなど)の格納に使用する表領域を作成できます。

- マルチテナント・コンテナ・データベース(CDB)の作成時にこの句を指定すると、シードの一部として表領域が作成されます。シードを使用してその後に作成されるプラガブル・データベース(PDB)には、この表領域とそのデータファイルが含まれます。この句で指定された表領域およびデータファイルは、ルートでは使用されません。
- 非CDBの作成時にこの句を指定すると、非CDBの一部として表領域が作成されます。

表領域がbigfileかsmallfileかを指定するには、BIGFILEまたはSMALLFILEを指定します。これらの句を指定しないと、SET DEFAULT TABLESPACE句で指定したタイプの表領域が作成されます。SET DEFAULT TABLESPACE句を使用しない場合、smallfile表領域が作成されます。これらの句のセマンティクスは、SET DEFAULT TABLESPACE句のセマンティクスと同じです。

datafile_tempfile_spec句を使用して、その表領域の1つ以上のデータファイルを指定できます。この句のセマンティクスの詳細は、「datafile_tempfile_spec」を参照してください。

enable_pluggable_database

この句を指定すると、CDBを作成できます。CREATE DATABASEを発行する前に、ENABLE_PLUGGABLE_DATABASE初期化パラメータをTRUEに設定する必要があります。CREATE DATABASE文によって、ルートおよびシード・コンテナを格納するCDBが作成されます。その後、CREATE PLUGGABLE DATABASE文を使用して、このCDB内にPDBを作成できます。enable_pluggable_database句を省略すると、非CDBが作成されます。これにはコンテナは何も格納できません。

関連項目:

- CDBの作成ステップの詳細は、『Oracle Database管理者ガイド』を参照してください。
- CREATE PLUGGABLE DATABASE

- 「CDBの作成: 例」を参照してください。

file_name_convert

file_name_convert句を使用すると、シードの関連ファイル(データファイルやウォレット・ファイルなど)の名前を、ルートの関連ファイルの名前を使用して生成する方法を指定できます。

- filename_patternには、ルートの関連ファイル名にある文字列を指定します。
- replacement_filename_patternには、置換文字列を指定します。

シードの関連ファイルの名前の生成時にfilename_patternがreplacement_filename_patternに置換されます。

Oracle Managed Filesで管理されているファイルまたはディレクトリと一致するファイル名のパターンは指定できません。

FILE_NAME_CONVERT = NONEを指定できます。これは、この句を省略した場合と同じになります。この句を省略すると、データベースはまずOracle Managed Filesを使用してシード・ファイル名を生成しようとします。Oracle Managed Filesを使用している場合は、PDB_FILE_NAME_CONVERT初期化パラメータを使用してファイル名が生成されます。このパラメータが設定されていないと、エラーが発生します。

tablespace_datafile_clauses

これらの句を使用して、シードPDB内のSYSTEM表領域とSYSAUX表領域を構成するすべてのデータファイルの属性を指定できます。SIZE size_clauseを指定しないと、その表領域のデータファイル・サイズは、対応するルートのデータファイル・サイズの事前定義された割合に設定されます。autoextend_clauseを指定しないと、それらの値はルートから継承されます。

これらの句のセマンティクスの詳細は、「size_clause」および「autoextend_clause」を参照してください。

undo_mode_clause

この句を使用すると、CDBにローカルUNDOモードまたは共有UNDOモードを指定できます。

- LOCAL UNDO ONを使用して、CDBにローカルUNDOモードを指定します。このモードでは、CDB内のすべてのコンテナでローカルUNDOが使用されます。
- LOCAL UNDO OFFを使用して、CDBに共有UNDOモードを指定します。このモードでは、アクティブなUNDO表領域は単一インスタンスCDBに対して1つ存在し、Oracle RAC CDBの場合、アクティブなUNDO表領域はインスタンスごとに1つ存在します。

この句を指定しない場合、LOCAL UNDO OFFがデフォルトになります。

USING MIRROR COPY

mirror_nameで指定されるミラー・コピーの準備ファイルを使用して、new_database_nameという名前のデータベースを作成するには、この句を使用します。

例

データベースの作成: 例

次の文は、データベースを作成して各引数をすべて指定します。

```
CREATE DATABASE sample
   CONTROLFILE REUSE
   LOGFILE
      GROUP 1 ('diskx:log1.log', 'disky:log1.log') SIZE 50K,
      GROUP 2 ('diskx:log2.log', 'disky:log2.log') SIZE 50K
   MAXLOGFILES 5
   MAXLOGHISTORY 100
```

```
   MAXDATAFILES 10
   MAXINSTANCES 2
   ARCHIVELOG
   CHARACTER SET AL32UTF8
   NATIONAL CHARACTER SET AL16UTF16
   DATAFILE
      'disk1:df1.dbf' AUTOEXTEND ON,
      'disk2:df2.dbf' AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED
   DEFAULT TEMPORARY TABLESPACE temp_ts
   UNDO TABLESPACE undo_ts
   SET TIME_ZONE = '+02:00';
```

この例では、初期化パラメータ・ファイルのDB_CREATE_FILE_DESTパラメータに値を設定して、Oracle Managed Filesを使用可能にしている状態を想定しています。したがって、DEFAULT TEMPORARY TABLESPACEおよびUNDO TABLESPACE句にファイル指定は必要ありません。

CDBの作成: 例

次の文は、CDB newcdbを作成します。ENABLE PLUGGABLE DATABASE句は、CDBを作成することを示します。このCDBには、ルート(CDB$ROOT)とシード(PDB$SEED)が格納されます。FILE_NAME_CONVERT句は、ルートに関連付けられたファイルの名前に含まれる/u01/app/oracle/oradata/newcdbを/u01/app/oracle/oradata/pdbseedに置換することによって、シードのファイルの名前を生成することを指定します。

```
CREATE DATABASE newcdb
  USER SYS IDENTIFIED BY sys_password
  USER SYSTEM IDENTIFIED BY system_password
  LOGFILE GROUP 1 ('/u01/logs/my/redo01a.log','/u02/logs/my/redo01b.log')
             SIZE 100M BLOCKSIZE 512,
          GROUP 2 ('/u01/logs/my/redo02a.log','/u02/logs/my/redo02b.log')
             SIZE 100M BLOCKSIZE 512,
          GROUP 3 ('/u01/logs/my/redo03a.log','/u02/logs/my/redo03b.log')
             SIZE 100M BLOCKSIZE 512
  MAXLOGHISTORY 1
  MAXLOGFILES 16
  MAXLOGMEMBERS 3
  MAXDATAFILES 1024
  CHARACTER SET AL32UTF8
  NATIONAL CHARACTER SET AL16UTF16
  EXTENT MANAGEMENT LOCAL
  DATAFILE '/u01/app/oracle/oradata/newcdb/system01.dbf'
    SIZE 700M REUSE AUTOEXTEND ON NEXT 10240K MAXSIZE UNLIMITED
  SYSAUX DATAFILE '/u01/app/oracle/oradata/newcdb/sysaux01.dbf'
    SIZE 550M REUSE AUTOEXTEND ON NEXT 10240K MAXSIZE UNLIMITED
  DEFAULT TABLESPACE deftbs
    DATAFILE '/u01/app/oracle/oradata/newcdb/deftbs01.dbf'
    SIZE 500M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED
  DEFAULT TEMPORARY TABLESPACE tempts1
    TEMPFILE '/u01/app/oracle/oradata/newcdb/temp01.dbf'
    SIZE 20M REUSE AUTOEXTEND ON NEXT 640K MAXSIZE UNLIMITED
  UNDO TABLESPACE undotbs1
    DATAFILE '/u01/app/oracle/oradata/newcdb/undotbs01.dbf'
    SIZE 200M REUSE AUTOEXTEND ON NEXT 5120K MAXSIZE UNLIMITED
  ENABLE PLUGGABLE DATABASE
    SEED
    FILE_NAME_CONVERT = ('/u01/app/oracle/oradata/newcdb/',
                         '/u01/app/oracle/oradata/pdbseed/')
    SYSTEM DATAFILES SIZE 125M AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED
    SYSAUX DATAFILES SIZE 100M
  USER_DATA TABLESPACE usertbs
    DATAFILE '/u01/app/oracle/oradata/pdbseed/usertbs01.dbf'
    SIZE 200M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;
```

CREATE DATABASE LINK

目的

CREATE DATABASE LINK文を使用すると、データベース・リンクを作成できます。データベース・リンクとは、他のデータベース上のオブジェクトにアクセスできる、データベース上のスキーマ・オブジェクトです。他のデータベースは、Oracle Databaseシステムである必要はありません。ただし、Oracle以外のシステムにアクセスする場合は、Oracle異機種間サービスを使用する必要があります。

データベース・リンクを作成した後で、表名、ビュー名またはPL/SQLオブジェクト名に@dblinkを追加し、そのリンクをSQL文で利用して、他のデータベース上の表、ビューおよびPL/SQLオブジェクトを参照できます。SELECT文を使用して、他のデータベース上の表またはビューを問い合せることができます。INSERT文、UPDATE文、DELETE文またはLOCK TABLE文を使用してもリモート表およびビューにアクセスできます。

関連項目:

- PL/SQLファンクション、プロシージャ、パッケージおよびデータ型を使用してリモート表またはビューにアクセスする方法の詳細は、『Oracle Database開発ガイド』を参照してください。
- 分散データベース・システムについては、『Oracle Database管理者ガイド』を参照してください。
- ALL_DB_LINKS、DBA_DB_LINKSおよびUSER_DB_LINKSデータ・ディクショナリ・ビューの既存のデータベース・リンクの詳細、およびV$DBLINK動的パフォーマンス・ビューを使用して既存のリンクのパフォーマンスを監視する方法については、『Oracle Databaseリファレンス』を参照してください。
- 接続または認証ユーザーのパスワードが変更された場合に、データベース・リンクを変更する方法については、「ALTER DATABASE LINK」を参照してください。
- 既存のデータベース・リンクを削除する方法については、「DROP DATABASE LINK」を参照してください。
- DML操作でリンクを使用する方法については、「INSERT」、「UPDATE」、「DELETE」および「LOCK TABLE」を参照してください。

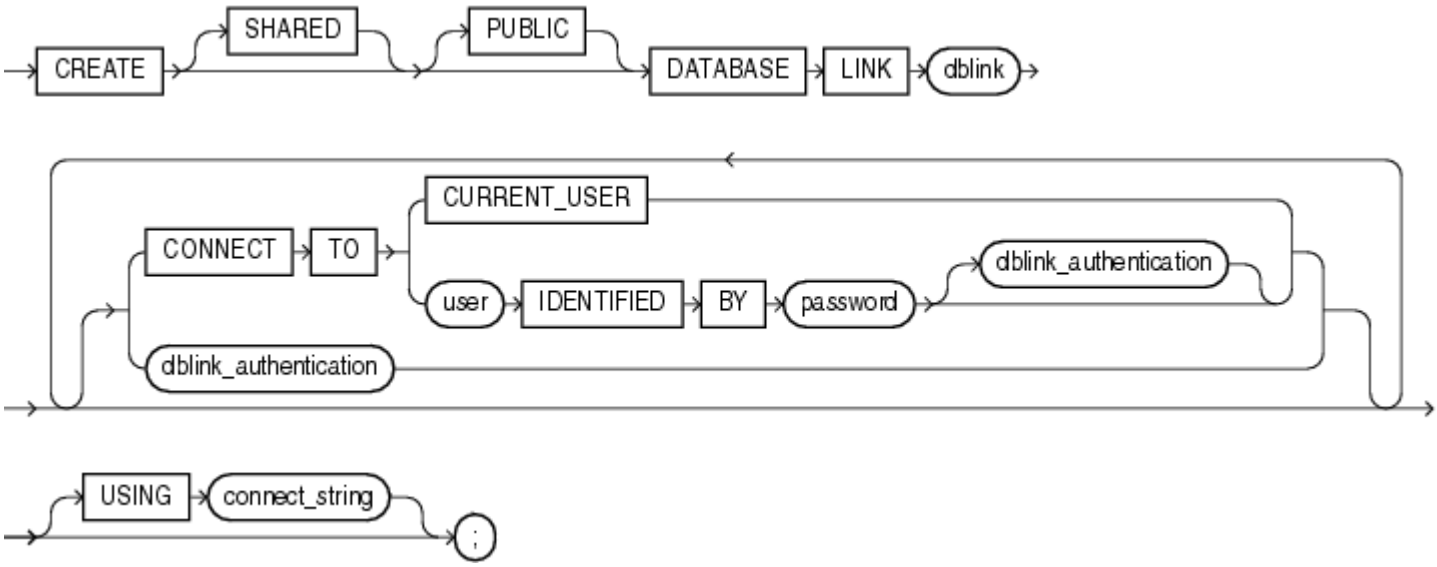
前提条件

プライベート・データベース・リンクを作成する場合、CREATE DATABASE LINKシステム権限が必要です。パブリック・データベース・リンクを作成する場合、CREATE PUBLIC DATABASE LINKシステム権限が必要です。また、リモートのOracle Databaseに対するCREATE SESSION権限が必要です。

なお、ローカルとリモートの両方のOracle Databaseに、Oracle Netをインストールしておく必要があります。

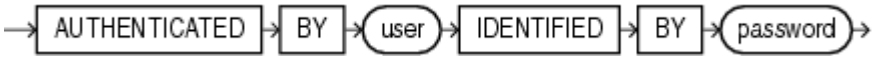
構文

create_database_link::=



(dblink::=)

dblink_authentication::=



セマンティクス

SHARED

SHAREDを指定すると、ソース・データベースからターゲット・データベースへの1つのネットワーク接続を使用する複数のセッションで共有可能な1つのデータベース・リンクを作成できます。共有サーバー構成では、共有データベース・リンクによって、リモート・データベースへの接続数が多くなりすぎるのを防ぐことができます。通常、共有リンクはパブリック・データベース・リンクでもあります。ただし、多くのクライアントが同じローカル・スキーマにアクセスして同じプライベート・データベース・リンクを使用する場合は、共有プライベート・データベース・リンクが役立つ場合もあります。

共有データベース・リンクでは、ソース・データベースの複数のセッションでターゲット・データベースへの同じ接続が共有されます。ターゲット・データベースでセッションが確立されると、ソース・データベースの別のセッションで接続を使用できるように、そのセッションは接続から関連付けを解除されます。認可されていないセッションがデータベース・リンクを使用して接続できないように、SHAREDを指定するときは、データベース・リンクの使用を認可されたユーザーに対して`dblink_authentication`句も指定する必要があります。

関連項目:

共有データベース・リンクの詳細は、『Oracle Database管理者ガイド』を参照してください。

PUBLIC

PUBLICを指定すると、すべてのユーザーが参照可能なパブリック・データベース・リンクを作成できます。この句を指定しない場合、データベース・リンクはプライベートとなり、作成したユーザー専用になります。

リモート・データベース上のアクセス可能なデータは、リモート・データベースへの接続時にデータベース・リンクで使用される識別によって異なります。

- `CONNECT TO user IDENTIFIED BY password`を指定した場合、データベース・リンクは、指定されたユーザー

とパスワードを使用して接続します。

- CONNECT TO CURRENT_USERを指定した場合、データベース・リンクは、そのリンクが使用される有効範囲に基づいて有効なユーザーを使用して接続します。
- いずれの句も指定しない場合、データベース・リンクは、ローカル接続されたユーザーとしてリモート・データベースに接続します。

関連項目:

パブリック・データベース・リンクの定義: 例

dblink

データベース・リンクの完全な名前または名前の一部を指定します。データベース名のみを指定した場合、ローカル・データベースのデータベース・ドメインが暗黙的に追加されます。

dblinkには、ASCII文字のみを使用してください。マルチバイト文字はサポートされていません。データベース・リンク名は、大文字と小文字が区別されず、大文字のASCII文字で保存されます。データベース名を引用識別子として指定した場合、引用符は特に警告もなく無視されます。

GLOBAL_NAMES初期化パラメータの値がTRUEの場合、データベース・リンクは、接続先のデータベースと同じ名前を持つ必要があります。GLOBAL_NAMESの値がFALSEで、データベースのグローバル名を変更した場合、グローバル名を指定できます。

Oracle RAC構成の1つのセッションまたは1つのインスタンスでオープンできるデータベース・リンクの最大数は、OPEN_LINKSおよびOPEN_LINKS_PER_INSTANCE初期化パラメータの値で指定します。

データベース・リンクの作成の制限事項

他のユーザーのスキーマ内にデータベース・リンクを作成したり、スキーマ名でdblinkを修飾することはできません。データベース・リンク名にはピリオドを指定できるため、ralph.linktosalesのような名前を付けた場合、スキーマralphのlinktosalesという名前のデータベース・リンクと解析されるのではなく、名前全体が自分のスキーマにあるデータベース・リンク名と解析されます。

関連項目:

17. データベース・リンクのネーミングのガイドラインは、「リモート・データベース内のオブジェクトの参照」を参照してください。
18. GLOBAL_NAMES、OPEN_LINKSおよびOPEN_LINKS_PER_INSTANCE初期化パラメータについては、『Oracle Databaseリファレンス』を参照してください。
19. データベースのグローバル名の変更の詳細は、「RENAME GLOBAL_NAME句」(ALTER DATABASE句)を参照してください。

CONNECT TO句

CONNECT TO句を使用すると、リモート・データベースへの接続に使用するユーザーおよび資格証明がある場合は、それらを指定できます。

CURRENT_USER句

CURRENT_USERを指定すると、現行のユーザーのデータベース・リンクを作成できます。現行ユーザーは、リモート・データベースに対し有効なアカウントを持つグローバル・ユーザーである必要があります。

データベース・リンクがストアド・オブジェクト内からではなく直接使用される場合、現行のユーザーは接続ユーザーと同じです。

データベース・リンクを開始するストアド・オブジェクト(プロシージャ、ビュー、トリガーなど)を実行する場合、CURRENT_USERは、ストアド・オブジェクトを所有するユーザーの名前であり、オブジェクトをコールしたユーザーの名前ではありません。たとえば、データベース・リンクが(scottによって作成された)プロシージャscott.p内にあり、ユーザーjaneがプロシージャscott.pをコールした場合、現行のユーザーはscottになります。

ただし、ストアド・オブジェクトが実行者権限ファンクション、プロシージャまたはパッケージである場合、実行者認可IDはリモート・ユーザーとしての接続に使用されます。たとえば、権限を持つデータベース・リンクがプロシージャscott.p(scottによって作成された実行者権限プロシージャ)内にあり、ユーザーJaneがプロシージャscott.pをコールした場合、CURRENT_USERはjaneであり、プロシージャは、Janeの権限で実行されます。

関連項目:

- 実行者権限ファンクションの詳細は、「CREATE FUNCTION」を参照してください。
- 「CURRENT_USERデータベース・リンクの定義: 例」を参照してください。

user IDENTIFIED BY password

固定ユーザー・データベース・リンクを使用して、リモート・データベースに接続するためのユーザー名およびパスワードを指定できます。この句を指定しない場合、データベース・リンクでは、データベースに接続している各ユーザーのユーザー名およびパスワードが使用されます。これを接続ユーザー・データベース・リンクといいます。

関連項目:

固定ユーザー・データベース・リンクの定義: 例

dblink_authentication

共有データベース・リンクを作成する場合(SHARED句を指定した場合)にのみ、この句を指定できます。ターゲット・インスタンスのユーザー名およびパスワードを指定します。この句は、リモート・サーバーに対してユーザーを認証するもので、セキュリティ上必要です。指定するユーザー名およびパスワードは、リモート・インスタンスで有効なユーザー名およびパスワードである必要があります。ユーザー名およびパスワードは、認証用としてのみ使用されます。このユーザーを対象とした認証以外の操作はありません。

USING 'connect string'

リモート・データベースのサービス名を指定します。データベース名のみを指定した場合、接続文字列にデータベース・ドメインが暗黙的に追加され、完全なサービス名が作成されます。したがって、リモート・データベースのデータベース・ドメインが現行のデータベース・ドメインと異なる場合は、完全なサービス名を指定する必要があります。

関連項目:

リモート・データベースの指定の詳細は、『Oracle Database管理者ガイド』を参照してください。

例

次の例では、localという名前のデータベースとremoteという名前の2つのデータベースを使用することを想定しています。この

例では、Oracle Databaseドメインを使用します。ご使用のドメインとは異なります。

パブリック・データベース・リンクの定義: 例

次の文は、remoteという名前の共有パブリック・データベース・リンクを定義します。このデータベース・リンクは、サービス名remoteで指定されたデータベースを参照します。

```
CREATE PUBLIC DATABASE LINK remote
   USING 'remote';
```

このデータベース・リンクによって、localデータベース上のユーザーhrがリモート・データベース上の表を更新できます(hrに適切な権限があることを想定しています)。

```
UPDATE employees@remote
   SET salary=salary*1.1
   WHERE last_name = 'Baer';
```

固定ユーザー・データベース・リンクの定義: 例

次の文で、remoteデータベース上のユーザーhrが、localデータベース上のhrスキーマに、localという名前の固定ユーザー・データベース・リンクを定義します。

```
CREATE DATABASE LINK local
   CONNECT TO hr IDENTIFIED BY password
   USING 'local';
```

このデータベース・リンクが作成されると、hrは、次のようにlocalデータベース上のスキーマhrの表を問い合せることができます。

```
SELECT * FROM employees@local;
```

ユーザーhrは、次のDML文を使用して、localデータベース上のデータを変更することもできます。

```
INSERT INTO employees@local
   (employee_id, last_name, email, hire_date, job_id)
   VALUES (999, 'Claus', 'sclaus@example.com', SYSDATE, 'SH_CLERK');
UPDATE jobs@local SET min_salary = 3000
   WHERE job_id = 'SH_CLERK';
DELETE FROM employees@local
   WHERE employee_id = 999;
```

この固定データベース・リンクを使用すると、remoteデータベース上のユーザーhrは、同じデータベース上の他のユーザーが所有する表にもアクセスできます。この文は、ユーザーhrがoe.customers表に対するREADまたはSELECT権限を持っていることを想定しています。この文では、localデータベースのユーザーhrに接続した後で、次のようにoe.customers表への問合せが行われます。

```
SELECT * FROM oe.customers@local;
```

CURRENT_USERデータベース・リンクの定義: 例

次の文は、リンク名としてサービス名全体を使用し、現行のユーザーのデータベース・リンクをremoteデータベースに定義します。

```
CREATE DATABASE LINK remote.us.example.com
   CONNECT TO CURRENT_USER
   USING 'remote';
```

この文を発行するユーザーは、LDAPディレクトリ・サービスに登録されたグローバル・ユーザーである必要があります。

特定の表がremoteデータベース上にあることを示さないように、シノニムを作成できます。次の文によって、これ以降にemp_tableを参照すると、リモート・データベース上のhrが所有するemployees表にアクセスします。

```
CREATE SYNONYM emp_table
   FOR oe.employees@remote.us.example.com;
```

CREATE DIMENSION

目的

CREATE DIMENSION文を使用すると、ディメンションを作成できます。ディメンションは、列セットのペア間の親子関係を定義するもので、この列セットに含まれるすべての列は、同じ表の列である必要があります。ただし、1つの列集合(レベル)の列は、別の集合の列とは異なる表から得ることができます。オプティマイザは、マテリアライズド・ビューとの関係を使用してクエリー・リライトを行います。SQLアクセス・アドバイザは、この関係に基づいて、特定のマテリアライズド・ビューの作成を推奨します。

> ノート:
>
> Oracle Database は、ディメンションの作成中に宣言する関係の妥当性チェックを自動的には行いません。hierarchy_clause および CREATE DIMENSION の dimension_join_clause で指定する関係の妥当性チェックを行うには、DBMS_OLAP.VALIDATE_DIMENSION プロシージャを実行する必要があります。

関連項目:

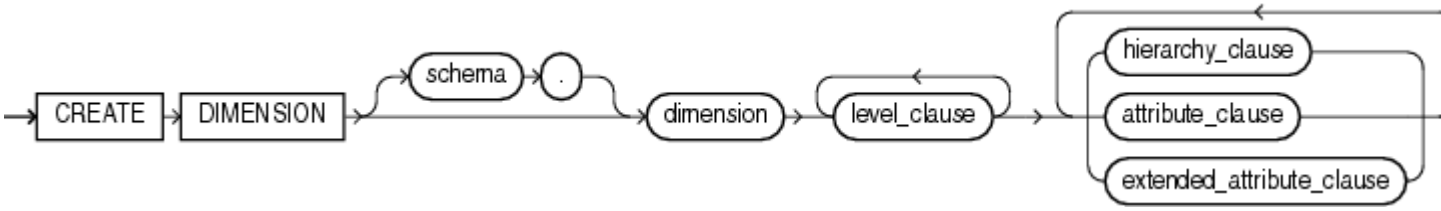
- マテリアライズド・ビューの詳細は、「CREATE MATERIALIZED VIEW」を参照してください。
- クエリー・リライト、オプティマイザおよびSQLアクセス・アドバイザの詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

前提条件

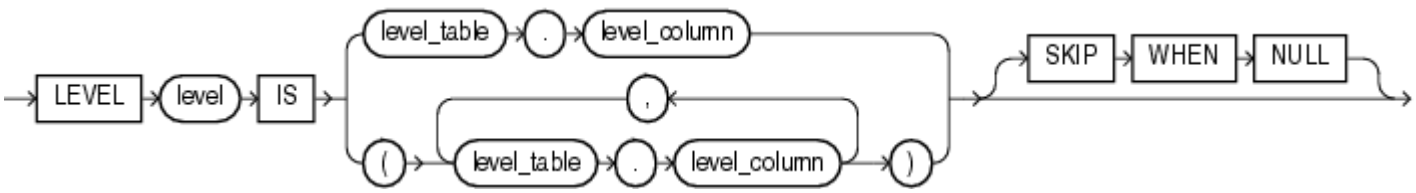
自分のスキーマ内にディメンションを作成する場合は、CREATE DIMENSIONシステム権限が必要です。他のユーザーのスキーマ内にディメンションを作成する場合は、CREATE ANY DIMENSIONシステム権限が必要です。どちらの場合も、ディメンションで参照されるオブジェクトに対して、READまたはSELECTオブジェクト権限が必要です。

構文

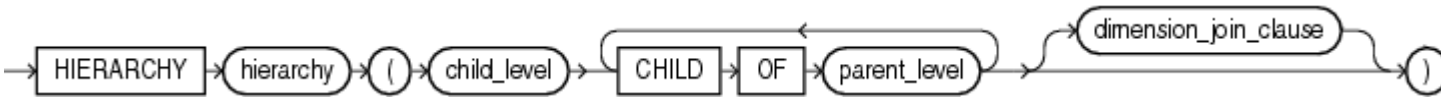
create_dimension::=



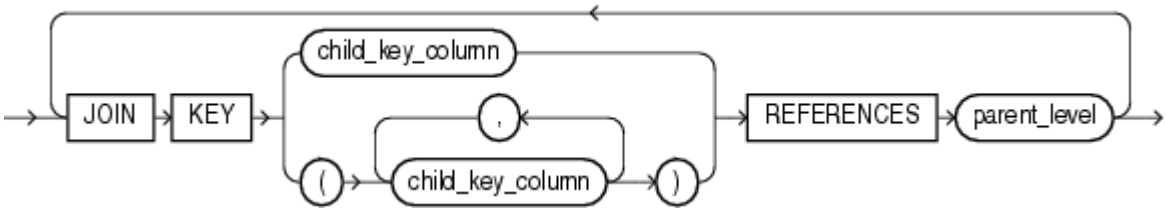
level_clause::=



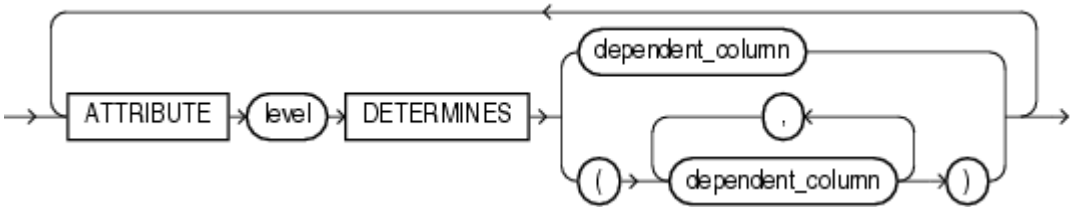
hierarchy_clause::=



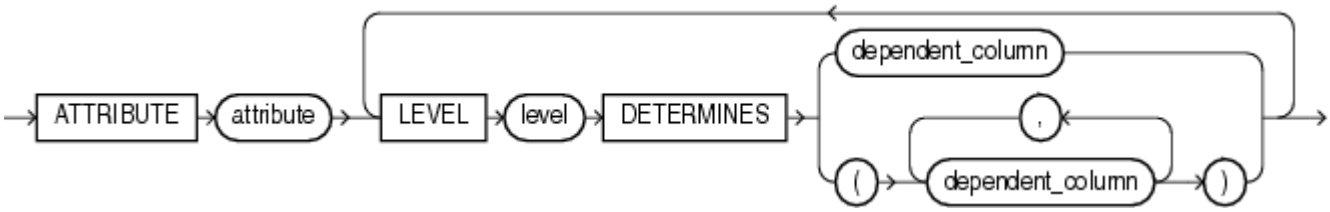
dimension_join_clause::=



attribute_clause::=



extended_attribute_clause::=



セマンティクス

schema

ディメンションを作成するスキーマを指定します。schemaを指定しない場合、自分のスキーマにそのディメンションが作成されます。

dimension

ディメンション名を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

level_clause

level_clauseでは、ディメンションのレベルを指定します。レベルは、ディメンション階層および属性を定義します。

level

レベルの名前を指定します。

level_table . level_column

レベル内の列を指定します。最大32列を指定できます。この句で指定する表は、すでに存在している必要があります。

SKIP WHEN NULL

この句を指定すると、指定したレベルがNULLの場合、そのレベルはスキップされます。この句を使用すると、指定したレベルをスキップする代替パスによって、親子関係の階層のつながりを維持できます。「hierarchy_clause」を参照してください。

ディメンション・レベル列の制限事項

ディメンション・レベル列には、次の制限事項があります。

- レベルの列は、すべて同じ表から得られたものである必要があります。
- 異なるレベルの列が異なる表から得られる場合、dimension_join_clauseを指定する必要があります。
- 指定する列の集合は、このレベルに一意である必要があります。
- 指定する列は、他のディメンションでは指定できません。
- レベルがSKIP WHEN NULLで指定されていないかぎり、各level_columnは、NULL以外である必要があります。NULL以外の列にNOT NULL制約を指定する必要はありません。SKIP WHEN NULLを指定する列に、NOT NULL制約を指定することはできません。

hierarchy_clause

hierarchy_clauseでは、ディメンションのレベルの線形階層を定義します。各階層が、ディメンションのレベル間で親子関係の連鎖を形成します。ディメンションの階層は、互いに依存していません。階層は、共通の列を持つことができます。

ディメンションの各レベルは、句の中で最高1回指定され、level_clauseで名前を付けておく必要があります。

hierarchy

階層名を指定します。この名前は、ディメンションで一意である必要があります。

child_level

親レベルとのn:1関係を持つレベルの名前を指定します。child_levelのlevel_columnsはNULL以外である必要があります。各child_level値は、次の名前付きparent_levelの値を一意に定義します。

子level_tableが親level_tableと異なる場合、dimension_join_clauseでそれらの結合関係を指定する必要があります。

parent_level

レベル名を指定します。

dimension_join_clause

dimension_join_clauseを使用すると、複数の表に列が含まれるディメンションに内部等価結合関係を指定できます。この句は、階層で指定されたすべての列が同じ表にあるとはかぎらない場合にのみ指定する必要があり、このときのみ指定できます。

child_key_column

親レベルの列と結合互換性のある1つ以上の列を指定します。

スキーマおよび各child_columnの表を指定しない場合、hierarchy_clauseのCHILD OF関係からスキーマおよび表が判断されます。child_key_columnのスキーマおよび列を指定する場合は、hierarchy_clauseのparent_levelの子のスキーマおよび列の表と一致している必要があります。

parent_level

レベル名を指定します。

結合ディメンションの制限事項

結合ディメンションには、次の制限事項があります。

- 同じ階層の既存のレベルの組に対して、1つのdimension_join_clauseのみを指定できます。
- child_key_columnsはNULL以外であり、親キーが一意でNULL以外である必要があります。条件を適用するために制約を定義する必要はありません。ただし、条件を満たさない場合、問合せが不適切な結果を戻すことがあります。
- 各子キーは、parent_level表のキーと結合する必要があります。
- 自己結合は使用できません。child_key_columnsを、parent_levelと同じ表に置くことはできません。
- 子キー列は、すべて同じ表から得られたものである必要があります。
- 子キー列数は、parent_levelの列数と一致し、列は結合可能である必要があります。
- 親レベルが複数の列で構成されている場合のみ、子キー列を指定します。

attribute_clause

attribute_clauseを使用すると、階層レベルによって一意に定義されている列を指定できます。levelの列は、dependent_columnsと同じ表からすべて得る必要があります。dependent_columnsは、level_clauseで指定されている必要はありません。

たとえば、階層レベルが市、都道府県名、および国の場合、市は市長、都道府県名は知事、国は首相を決定します。

extended_attribute_clause

この句を使用すると、1つ以上のレベルと列の関係に属性名を指定できます。この句で作成する属性の種類は、attribute_clauseを使用して作成される属性の種類と同じです。唯一の違いは、属性にレベル名とは異なる名前を割り当てることができることです。

例

ディメンションの作成: 例

この文は、サンプル・スキーマshにcustomers_dimディメンションを作成するために使用されました。

```
CREATE DIMENSION customers_dim
   LEVEL customer   IS (customers.cust_id)
   LEVEL city       IS (customers.cust_city)
   LEVEL state      IS (customers.cust_state_province)
   LEVEL country    IS (countries.country_id)
   LEVEL subregion  IS (countries.country_subregion)
   LEVEL region     IS (countries.country_region)
   HIERARCHY geog_rollup (
      customer      CHILD OF
      city          CHILD OF
      state         CHILD OF
      country       CHILD OF
      subregion     CHILD OF
      region
   JOIN KEY (customers.country_id) REFERENCES country
   )
   ATTRIBUTE customer DETERMINES
   (cust_first_name, cust_last_name, cust_gender,
    cust_marital_status, cust_year_of_birth,
    cust_income_level, cust_credit_limit)
   ATTRIBUTE country DETERMINES (countries.country_name)
;
```

拡張属性を含むディメンションの作成：例

または、次の例に示すように、attribute_clauseのかわりにextended_attribute_clauseを使用することもできます。

```
CREATE DIMENSION customers_dim
   LEVEL customer   IS (customers.cust_id)
   LEVEL city       IS (customers.cust_city)
   LEVEL state      IS (customers.cust_state_province)
   LEVEL country    IS (countries.country_id)
   LEVEL subregion  IS (countries.country_subregion)
   LEVEL region     IS (countries.country_region)
   HIERARCHY geog_rollup (
      customer      CHILD OF
      city          CHILD OF
      state         CHILD OF
      country       CHILD OF
      subregion     CHILD OF
      region
   JOIN KEY (customers.country_id) REFERENCES country
   )
   ATTRIBUTE customer_info LEVEL customer DETERMINES
   (cust_first_name, cust_last_name, cust_gender,
    cust_marital_status, cust_year_of_birth,
    cust_income_level, cust_credit_limit)
   ATTRIBUTE country DETERMINES (countries.country_name);
```

NULL列値を含むディメンションの作成：例

次の例では、レベル列のいずれかがNULLで、階層のつながりを保持する必要がある場合のディメンションの作成方法を示します。この例では、簡単に説明するためにcust_marital_status列を使用しています。この列はNOT NULL列ではありません。この制約がある場合は、SKIP WHEN NULL句を使用する前にこの制約を使用禁止にする必要があります。

```
CREATE DIMENSION customers_dim
   LEVEL customer IS (customers.cust_id)
   LEVEL status IS (customers.cust_marital_status) SKIP WHEN NULL
   LEVEL city IS (customers.cust_city)
   LEVEL state IS (customers.cust_state_province)
   LEVEL country IS (countries.country_id)
   LEVEL subregion IS (countries.country_subregion) SKIP WHEN NULL
   LEVEL region IS (countries.country_region)
   HIERARCHY geog_rollup (
      customer CHILD OF
      city CHILD OF
      state CHILD OF
      country CHILD OF
      subregion CHILD OF
      region
   JOIN KEY (customers.country_id) REFERENCES country
   )
   ATTRIBUTE customer DETERMINES
   (cust_first_name, cust_last_name, cust_gender,
    cust_marital_status, cust_year_of_birth,
    cust_income_level, cust_credit_limit)
   ATTRIBUTE country DETERMINES (countries.country_name)
;
```

CREATE DIRECTORY

目的

CREATE DIRECTORY文を使用すると、ディレクトリ・オブジェクトを作成できます。ディレクトリ・オブジェクトは、外部バイナリ・ファイルLOB(BFILE)および外部表データが存在するサーバー・ファイル・システム上のディレクトリの別名を示します。PL/SQLコードおよびOCIコールでBFILEを参照する際、管理の汎用性のために、オペレーティング・システムのパス名をハード・エンコードせずにディレクトリ名を使用できます。

すべてのディレクトリは1つのネームスペースに作成され、個別のスキーマでは所有されません。ディレクトリに対するオブジェクト権限を特定ユーザーに付与することによって、そのディレクトリ構造内に格納されているBFILEへのアクセスを制限できます。

関連項目:

6. BFILEオブジェクトの詳細は、「ラージ・オブジェクト(LOB)・データ型」を参照してください。
7. オブジェクト権限の付与の詳細は、「GRANT」を参照してください。
8. 「CREATE TABLE」の「external_table_clause::=」を参照してください。

前提条件

ディレクトリを作成するには、CREATE ANY DIRECTORYシステム権限が必要です。

ディレクトリを作成すると、そのディレクトリに対するREAD、WRITEおよびEXECUTEオブジェクト権限が自動的に付与され、他のユーザーおよびロールにこれらの権限を付与できます。DBAも、これらの権限を他のユーザーおよびロールに付与できます。

ディレクトリに対するWRITE権限は、外部表との接続において便利です。これによって、権限受領者は、外部表のエージェントがディレクトリに書き込めるのがログ・ファイルなのか不良ファイルなのかを判断できます。

ファイルの記憶域用に、それに応じたオペレーティング・システムのディレクトリ、Oracle Automatic Storage Management (Oracle ASM)ディスク・グループ、またはOracle ASMディスク・グループ内のディレクトリを作成する必要もあります。システム管理者およびデータベース管理者は、このオペレーティング・システムのディレクトリに、Oracle Databaseプロセスに対する読取り権限および書込み権限が正しく設定されていることを確認する必要があります。

ディレクトリに対して付与される権限は、オペレーティング・システムのディレクトリ用に定義されたアクセス権限とは無関係に作成されるため、これらの権限は完全に対応しない場合があります。たとえば、サンプル・ユーザーhrに、ディレクトリ・オブジェクトに対するREAD権限が付与されていても、それに対応するオペレーティング・システムのディレクトリにOracle Databaseプロセスに対するREAD権限が付与されていない場合は、エラーが発生します。

制限事項

BFILEオブジェクトを開く際に、ディレクトリ・オブジェクト・パスまたはファイル名ではシンボリック・リンクは使用できません。ディレクトリ・パス全体および ファイル名がチェックされ、シンボリック・リンクが見つかると、次のエラーが返されます。

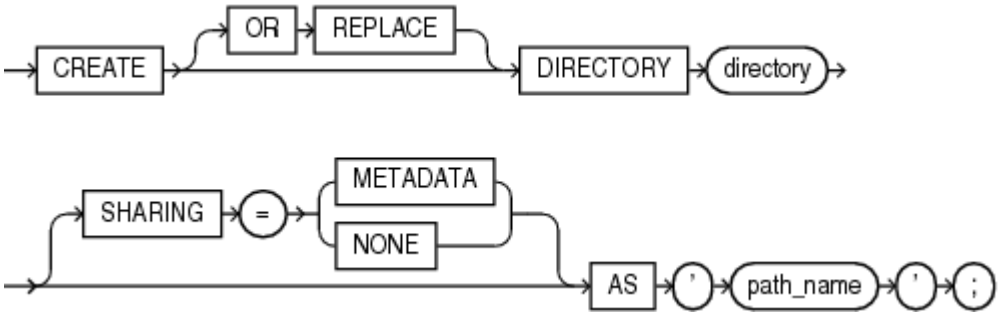
```
ORA-22288: file or LOB operation FILEOPEN failed soft link in path
```

回避策

開こうとしているデータベース・ディレクトリ・オブジェクトまたはファイル名にシンボリック・リンクが含まれている場合、実際のパスおよびファイル名を指定するように変更します。

構文

create_directory::=



セマンティクス

OR REPLACE

OR REPLACEを指定すると、既存のディレクトリ・データベース・オブジェクトを再作成できます。この句を指定した場合、既存のディレクトリに付与されているデータベース・オブジェクト権限を削除、再作成および再付与しなくても、そのディレクトリの定義を変更できます。

再定義したディレクトリに対する権限が付与されていたユーザーは、権限が再付与されなくてもそのディレクトリにアクセスできます。

関連項目:

データベースからのディレクトリの削除については、「DROP DIRECTORY」を参照してください。

SHARING

この句は、アプリケーション・ルートにディレクトリを作成する場合にのみ適用されます。このタイプのディレクトリはアプリケーション共通オブジェクトと呼ばれ、アプリケーション・ルートに属するアプリケーションPDBと共有できます。ディレクトリの共有方法を決定するには、次の共有属性のいずれかを指定します。

- METADATA - メタデータ・リンクはディレクトリのメタデータを共有しますが、データは各コンテナに固有です。このタイプのディレクトリは、メタデータリンク・アプリケーション共通オブジェクトと呼ばれます。
- NONE - ディレクトリは共有されません。

この句を指定しない場合、DEFAULT_SHARING初期化パラメータの値を使用して、ディレクトリの共有属性が決定されます。DEFAULT_SHARING初期化パラメータに値が含まれていない場合、デフォルトはMETADATAです。

ディレクトリの共有属性を作成後に変更することはできません。

関連項目:

- DEFAULT_SHARING初期化パラメータの詳細は、Oracle Databaseリファレンスを参照してください
- アプリケーション共通オブジェクトの作成の詳細は、『Oracle Database管理者ガイド』を参照してください。

directory

作成するディレクトリ・オブジェクトの名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

Oracle Databaseでは、指定するディレクトリが実際に存在するかどうかが検証されません。このため、オペレーティング・システムに存在するディレクトリを指定してください。また、オペレーティング・システムでパス名の大/小文字が区別される場合は、正しい形式でディレクトリ名を指定する必要があります。パス名の最後にスラッシュを付ける必要はありません。

ディレクトリ名で親ディレクトリを参照しないでください。たとえば、次の構文は有効です。

```
CREATE DIRECTORY mydir AS '/scratch/data/file_data';
```

ただし、次の構文は無効です。

```
CREATE DIRECTORY mydir AS '/scratch/../file_data';
```

path_name

ファイルがあるサーバーのオペレーティング・システム・ディレクトリのフル・パス名を指定します。一重引用符が必要です。その結果、パス名の大/小文字が区別されます。

例

ディレクトリの作成: 例

次の文は、サーバー上のディレクトリを指定するディレクトリ・データベース・オブジェクトを作成します。

```
CREATE DIRECTORY admin AS '/disk1/oracle/admin';
```

次の文は、オペレーティング・システムのディレクトリusr/bin/bfile_dirに格納されているBFILEにアクセスできるように、ディレクトリのデータベース・オブジェクトbfile_dirを再定義します。

```
CREATE OR REPLACE DIRECTORY bfile_dir AS '/usr/bin/bfile_dir';
```

CREATE DISKGROUP

ノート:

この SQL 文は、Oracle ASM を使用しており、Oracle ASM インスタンスを起動している場合にのみ有効です。この文の発行は、通常のデータベース・インスタンスからではなく、Oracle ASM インスタンスから行う必要があります。Oracle ASM インスタンスの起動の詳細は、『Oracle Automatic Storage Management 管理者ガイド』を参照してください。

目的

CREATE DISKGROUP句を使用すると、ディスクのグループに名前を付け、そのグループをOracle Databaseが管理するように指定できます。Oracle Databaseは、ディスク・グループを論理単位として管理し、I/Oの均衡を保つために各ファイルを均等に分散します。また、Oracle Databaseでは、ディスク・グループで使用可能なすべてのディスクにデータベース・ファイルが自動的に分散され、記憶域構成が変わると常に、記憶域が自動的にリバランスされます。

この文を使用すると、ディスク・グループが作成され、1つ以上のディスクがディスク・グループに割り当てられ、ディスク・グループの初めてのマウントが実行されます。CREATE DISKGROUPは、ローカル・ノードのディスク・グループのみマウントします。後続のインスタンスで、Oracle ASMによって自動的にディスク・グループをマウントさせる場合は、初期化パラメータ・ファイルのASM_DISKGROUPS初期化パラメータの値に、そのディスク・グループ名を追加する必要があります。SPFILEを使用している場合、そのディスク・グループは自動的に初期化パラメータに追加されます。

関連項目:

- ディスク・グループの変更については、「ALTER DISKGROUP」を参照してください。
- Oracle ASMおよびディスク・グループを使用してデータベース管理を簡略化する方法については、『Oracle Automatic Storage Management管理者ガイド』を参照してください。
- 初期化パラメータ・ファイルへのディスク・グループ名の追加の詳細は、「ASM_DISKGROUPS」を参照してください。
- Oracle ASM操作を監視する方法については、「V$ASM_OPERATION」を参照してください。
- ディスク・グループの削除については、「DROP DISKGROUP」を参照してください。

前提条件

この文を発行するには、SYSASMシステム権限が必要です。

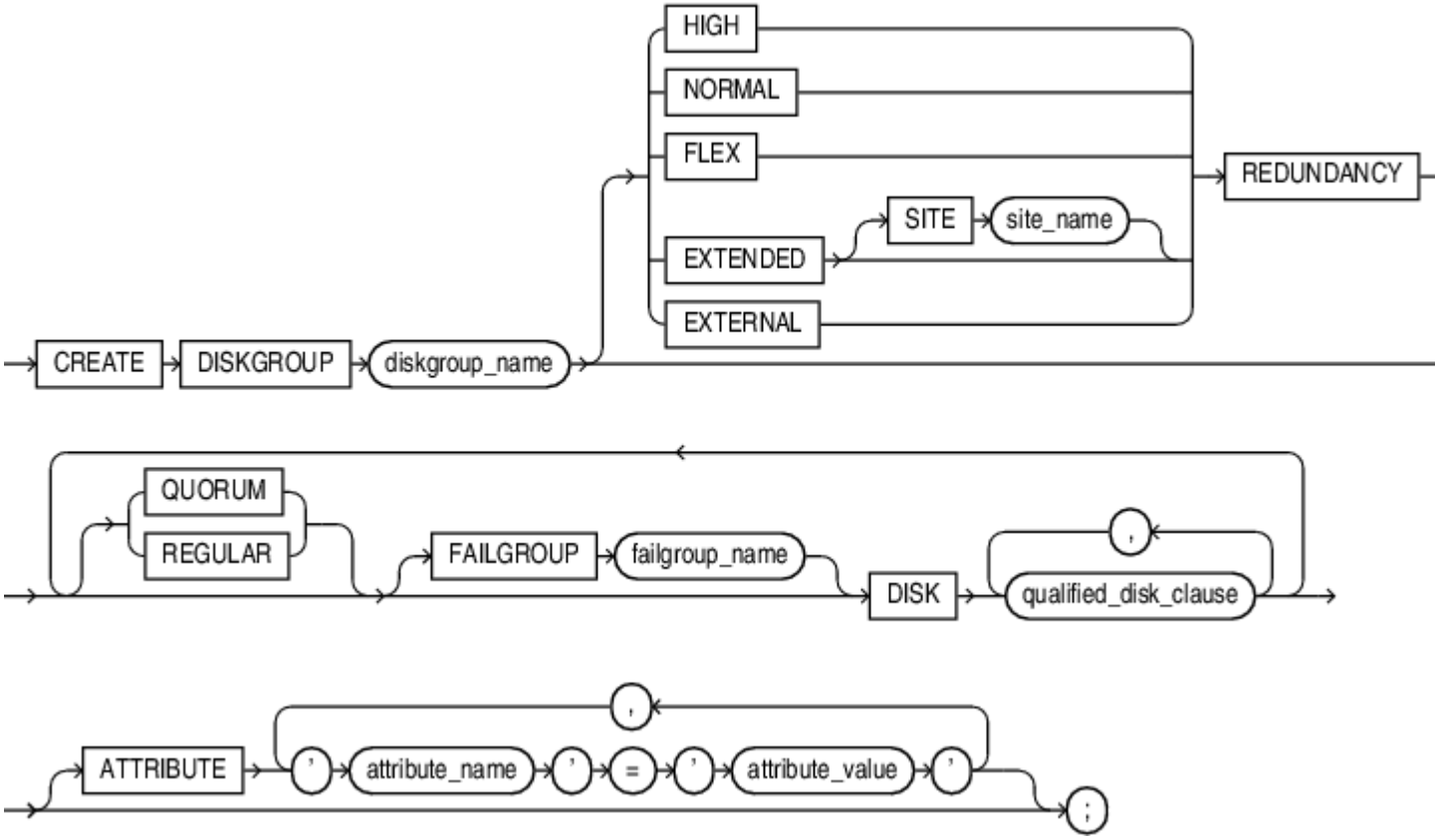
この文を発行する前に、オペレーティング・システムのフォーマット・ユーティリティを使用して、ディスクをフォーマットしておく必要があります。また、Oracle Databaseユーザーが読取り/書込み権限を持ち、ASM_DISKSTRINGを使用してディスクが検出可能であることを確認します。

ファイル・システムではなく、Oracle ASMディスク・グループにデータベース・ファイルを格納している場合、Oracle ASMインスタンスを構成および起動してディスク・グループを管理しないと、データベース・インスタンスがディスク・グループのファイルにアクセスできません。

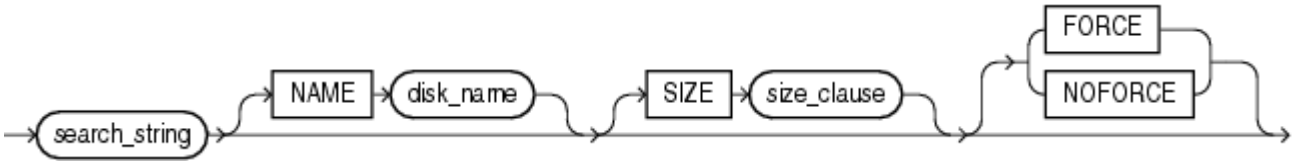
各データベース・インスタンスは、データベースと同じノード上にある1つのOracle ASMインスタンスと通信します。同じノード上にある複数のデータベース・インスタンスが、1つのOracle ASMインスタンスと通信できます。

構文

create_diskgroup::=



qualified_disk_clause::=



(size_clause::=)

diskgroup_name

ディスク・グループ名を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。ただし、ディスク・グループはスキーマ・オブジェクトではありません。

ノート:

ディスク・グループ名には引用識別子を使用しないことをお薦めします。SQL*Plus で CREATE DISKGROUP 文を発行する場合はこのような引用識別子は許可されていますが、ディスク・グループを管理する他のツールでは有効でない場合があります。

REDUNDANCY句

REDUNDANCY句を使用すると、ディスク・グループの冗長レベルを指定できます。

- NORMAL REDUNDANCYの場合、2つ以上の障害グループが存在する必要があります(次の「FAILGROUP句」を参照)。Oracle ASMでは、ディスク・グループ・テンプレートで指定された属性に従って、ディスク・グループのすべてのファイルの冗長性が提供されます。NORMAL REDUNDANCYディスク・グループでは、1つのグループが消失してもリカバリできます。ディスク・グループ・テンプレートの詳細は、「ALTER DISKGROUP」の「diskgroup_template_clauses」を参照してください。

  REDUNDANCY句を指定しない場合、NORMAL REDUNDANCYがデフォルトになります。したがって、この句を指定しない場合、2つ以上の障害グループを作成する必要があります。それを作成しないと、作成操作は正常に実行されません。

- HIGH REDUNDANCYの場合、少なくとも3つの障害グループが存在する必要があります。Oracle ASMでは、エクステントごとにミラー化された2つのコピーが存在する3方向ミラー化でミラーリングが行われます。HIGH REDUNDANCYディスク・グループでは、2つの障害グループが消失してもリカバリできます。

- FLEX REDUNDANCYディスク・グループ・タイプを指定すると、ディスク・グループを作成した後、データベースによって独自の冗長性が指定されることが許可されます。ファイルを作成した後で、その冗長性を変更することもできます。このタイプのディスク・グループでは、Oracle ASMファイル・グループおよび割当て制限グループがサポートされます。フレックス・ディスク・グループには、3つ以上の障害グループが存在する必要があります。フレックス・ディスク・グループに含まれる障害グループの数が5個より少ない場合、1つが消失してもリカバリできます。それ以外の場合は、2つの障害グループが消失してもリカバリできます。フレックス・ディスク・グループを作成するには、COMPATIBLE.ASMおよびCOMPATIBLE.RDBMSディスク・グループ属性が12.2以上に設定されている必要があります。

- EXTENDED REDUNDANCYは、拡張クラスタ環境で可用性が高いことに加えて、フレックス・ディスク・グループのすべての機能を持つディスク・グループです。クラスタには、物理的に食べられた複数のサイトにまたがるノートがあります。詳細は、「Oracle ASM拡張ディスク・グループについて」を参照してください。

  SITEキーワードを使用すると、拡張ディスク・グループ内のファイルおよびファイル・グループの冗長性を、ディスク・グループごとではなく、サイトごとに指定できます。

- EXTERNAL REDUNDANCYは、Oracle ASMでディスク・グループの冗長性が提供されないことを示します。ディスク・グループを構成する各ディスクで冗長性を確保する(たとえばストレージ・アレイを使用する)か、ディスク障害が発生した場合にはディスク・グループのデータ損失を想定しておく(テスト環境など)必要があります。EXTERNAL REDUNDANCYを指定する場合、FAILGROUP句は指定できません。

作成したディスク・グループの冗長性レベルを後で変更することはできませんが、例外として、標準冗長性および高冗長性のディスク・グループをフレックス・ディスク・グループに変換できます。詳細は、「ALTER DISKGROUP」の「convert_redundancy_clause」を参照してください。

QUORUM | REGULAR

これらのキーワードを使用すると、障害グループまたはディスク指定を修飾できます。

10. REGULARディスク、またはクォーラム以外の障害グループ内のディスクには、任意のファイルを含めることができます。

11. QUORUMディスクまたはクォーラム障害グループ内のディスクには、データベース・ファイル、Oracle Cluster Registry(OCR)または動的ボリュームを含めることはできません。ただし、QUORUMディスクにCluster Synchronization Services(CSS)の投票ファイルは含めることができます。Oracle ASMは可能な場合は常に、投票ファイル用に、クォーラム・ディスクまたはクォーラム障害グループ内のディスクを使用します。

    クォーラム障害グループは、ユーザー・データの格納に関する冗長性要件の決定時に考慮されません。

いずれのキーワードも指定しない場合、デフォルトでREGULARが使用されます。

障害グループを明示的に指定する場合は、キーワードFAILGROUPの前にQUORUMまたはREGULARを指定します。暗黙的に作成した障害グループとともにディスク・グループを作成する場合は、キーワードDISKの前にこれらのキーワードを指定します。

関連項目:

クォーラム・ディスクと標準ディスクおよび障害グループの詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

FAILGROUP句

この句を使用すると、1つ以上の障害グループの名前を指定できます。NORMALまたはHIGH REDUNDANCYを指定している場合にこの句を省略すると、Oracle Databaseは、自動的にディスク・グループの各ディスクを障害グループに追加します。障害グループの暗黙的な名前は、オペレーティング・システムに依存しないディスク名と同じです(「NAME句」を参照)。

EXTERNAL REDUNDANCYディスク・グループを作成している場合、この句は指定できません。

qualified_disk_clause

DISK qualified_disk_clauseを指定すると、ディスク・グループにディスクを追加できます。

search_string

ディスク・グループに追加するディスクごとに、オペレーティング・システム依存の検索文字列を指定します。Oracle ASMでは、この文字列を使用してディスクが検索されます。search_stringは、ASM_DISKSTRING初期化パラメータの文字列を使用した検索で戻されるディスクのサブセットを指している必要があります。search_stringが、Oracle Databaseユーザーが読取り/書込み権限を持っているディスクを指していない場合、Oracle ASMはエラーを戻します。また、異なるディスク・グループに割り当てられている1つ以上のディスクを指している場合、FORCEを指定していなければ、エラーが戻されます。

Oracle ASMでは、有効な追加候補のディスクごとにディスク・ヘッダーがフォーマットされ、このディスクが新規のディスク・グループのメンバーであることが示されます。

関連項目:

検索文字列の指定の詳細は、「ASM_DISKSTRING」初期化パラメータを参照してください。

NAME句

NAME句は、search_stringが1つのディスクを指す場合にのみ有効です。この句を使用すると、オペレーティング・システムに依存しないディスクの名前を指定できます。名前の長さは最大30文字で、英数字のみを使用できます。最初の文字は、英字にする必要があります。ASMLIBでディスクにラベルを割り当てている場合にこの句を省略すると、ディスク名としてそのラベルが使用されます。ASMLIBでラベルを割り当てていない場合にこの句を省略すると、Oracle ASMは、「diskgroupname_####」(####はディスク番号)という書式のデフォルト名を作成します。この名前を使用して、後続のOracle ASM操作でディスクを参照できます。

SIZE句

この句を使用すると、ディスクのサイズをバイト単位で指定できます。ディスク容量を超えるサイズを指定すると、Oracle ASMはエラーを戻します。ディスク容量よりも小さいサイズを指定した場合、Oracle ASMで使用されるディスク領域が制限されます。サイズ値は、1つのディスク・グループ内のすべてのディスクで同一にする必要があります。この句を指定しない場合、Oracle ASMはプログラム的にディスクのサイズを決定します。

FORCE

FORCEを指定すると、Oracle ASMで、別のディスク・グループのメンバーであるディスクをディスク・グループに追加できます。

ノート:

この方法で FORCE を使用すると、既存のディスク・グループが破棄される可能性があります。

この句を有効にするには、ディスクはディスク・グループのメンバーである必要があり、ディスクがマウントされたディスク・グループの一部であってはなりません。

NOFORCE

NOFORCEを指定すると、Oracle ASMで、ディスクが別のディスク・グループのメンバーである場合にエラーを戻すことができます。デフォルトはNOFORCEです。

ATTRIBUTE句

この句を使用すると、ディスク・グループの属性値を設定できます。V$ASM_ATTRIBUTEビューを問い合せることによって、現在の属性値を確認できます。表13-2に、この句で設定できる属性を示します。属性値はすべて文字列です。

表13-2 ディスク・グループの属性

属性	有効な値	説明
ACCESS_CONTROL.ENABLED	true または false	ディスク・グループで Oracle ASM のファイル・アクセス制御を有効にするかどうかを指定します。true に設定した場合、Oracle ASM ファイルへのアクセスがアクセス制御の対象になります。false に設定した場合、すべてのユーザーがディスク・グループ内のすべてのファイルにアクセスできます。他のすべての操作は、この属性とは関係なく動作します。デフォルト値は false です。 compatible.rdbms 属性と compatible.asm 属性が両方とも 11.2 以上に設定されている場合、ALTER DISKGROUP ... SET ATTRIBUTE 文でこの属性を設定できます。この属性は、ディスク・グループの作成時には設定できません。 ファイル・アクセス制御を既存のディスク・グループに対して設定した場合、作成済のファイルはすべてのユーザーからアクセス可能なままになります。これを避けるには、ALTER DISKGROUP SET PERMISSION 文を実行して権限を制限します。 ノート: この属性は、Oracle ASM のファイル・アクセス制御を管理するために ACCESS_CONTROL.UMASK とともに使用します。ACCESS_CONTROL.ENABLED ディスク属性を設定したら、ACCESS_CONTROL.UMASK 属性で権限を設定する必要があります。

属性	有効な値	説明
ACCESS_CONTROL.UMASK	3 桁の数(各桁は 0(ゼロ)、2 または 6)。	Oracle ASM ファイルの作成時に、そのファイルを所有するユーザー(最初の桁)、同じユーザー・グループ内のユーザー(2 桁目)、およびそのユーザー・グループに属さない他のユーザー(3 桁目)に対してマスクする権限を指定します。この属性は、ディスク・グループ内のすべてのファイルに適用されます。0(ゼロ)に設定すると、何もマスクされません。2 に設定すると、書込み権限がマスクされます。6 に設定すると、読取り権限と書込み権限の両方がマスクされます。デフォルト値は 066 です。 compatible.rdbms 属性と compatible.asm 属性が両方とも 11.2 以上に設定されている場合、ALTER DISKGROUP ... SET ATTRIBUTE 文でこの属性を設定できます。この属性は、ディスク・グループの作成時には設定できません。 ファイル・アクセス制御を既存のディスク・グループに対して設定した場合、作成済のファイルはすべてのユーザーからアクセス可能なままになります。これを避けるには、ALTER DISKGROUP SET PERMISSION 文を実行して権限を制限します。 ノート: この属性は、Oracle ASM のファイル・アクセス制御を管理するために ACCESS_CONTROL.ENABLED とともに使用します。ACCESS_CONTROL.UMASK を設定する前に、ACCESS_CONTROL.ENABLED を true に設定する必要があります。
AU_SIZE	サイズ(バイト単位)。有効な値は、1M から 64M の 2 の累乗です。例: 4M、4194304。	割当て単位サイズを指定します。この属性は、ディスク・グループの作成時にのみ設定でき、ALTER DISKGROUP 文で変更することはできません。
COMPATIBLE.ADVM	有効な Oracle Database バージョン番号脚注 1	ディスク・グループに Oracle ADVM ボリュームを含めることができるかどうかを決定します。11.2 以上の値を設定する必要があります。この属性を設定する前に、COMPATIBLE.ASM の値を 11.2 以上に設定しておく必要があります。また、Oracle ADVM ボリューム・ドライバをロードしておく必要があります。 デフォルトでは、COMPATIBLE.ADVM 属性の値は、設定するまでは空です。

属性	有効な値	説明
COMPATIBLE.ASM	有効な Oracle Database バージョン番号脚注 1	Oracle ASM インスタンスでディスク・グループを使用するために必要な最小ソフトウェア・バージョンを指定します。この設定は、ディスク上の Oracle ASM メタデータのデータ構造の形式にも影響します。 Oracle Database 11g の Oracle ASM では、CREATE DISKGROUP SQL 文、ASMCMD mkdg コマンド、および Oracle Enterprise Manager の「ディスク・グループの作成」ページを使用する場合の COMPATIBLE.ASM 属性のデフォルト設定は 10.1 です。ASMCA を使用してディスク・グループを作成する場合、デフォルト設定は 11.2 です。
COMPATIBLE.RDBMS	有効な Oracle Database バージョン番号脚注 1	すべてのデータベース・インスタンスでディスク・グループを使用するために必要な COMPATIBLE データベース初期化パラメータの最小設定を指定します。 COMPATIBLE.RDBMS 属性を設定する前に、そのディスク・グループにアクセスするすべてのデータベースで、COMPATIBLE 初期化パラメータの値が COMPATIBLE.RDBMS の新しい設定値以上に設定されていることを確認します。たとえば、データベースの COMPATIBLE 初期化パラメータが 11.1 または 11.2 に設定されている場合、COMPATIBLE.RDBMS は、10.1 から 11.1 の間(これらの値を含む)の任意の値に設定できます。 Oracle Database 11g の Oracle ASM では、SQL の CREATE DISKGROUP 文、ASMCMD mkdg コマンド、ASMCA の「ディスク・グループの作成」ページ、および Oracle Enterprise Manager の「ディスク・グループの作成」ページを使用する場合の COMPATIBLE.RDBMS 属性のデフォルト設定は 10.1 です。
CONTENT.CHECK	true または false	ディスク・グループのリバランスのためにデータ・コピー操作を実行する際のコンテンツ・チェックを有効化(true)または無効化(false)します。この属性は、ディスク・グループの作成時には設定できません。 デフォルト値は COMPATIBLE.ASM 属性に依存し、次のルールに従います。 ● COMPATIBLE.ASM > = 19.0.0.0.0 の場合、CONTENT.CHECK のデフォルトは true です。 ● COMPATIBLE.ASM < 19.0.0.0.0 の場合、CONTENT.CHECK のデフォルトは false です。

属性	有効な値	説明
		ノート: このルールは、新しいディスク・グループの作成に対してのみ有効です。既存のディスク・グループの COMPATIBLE.ASM 属性が 19.0.0.0.0 以上に更新された場合、CONTENT.CHECK 属性は現在の値のままです。
DISK_REPAIR_TIME	0 から 136 年	ディスクは、オフラインに切り替えられると、デフォルトの時間が経過した後に Oracle ASM によって削除されます。compatible.rdbms と compatible.asm の両方の属性が 11.1 以上に設定されている場合、ディスクを修復してオンラインに戻すことができるように、ALTER DISKGROUP ... SET ATTRIBUTE 文で disk_repair_time 属性を設定して、そのデフォルトの時間を変更できます。この属性は、ディスク・グループの作成時には設定できません。 時間は分単位(M)または時間単位(H)で指定できます。指定した経過時間は、ディスク・グループがマウントされているときにのみ加算されます。単位を省略した場合、デフォルトは H になります。この属性を省略し、compatible.rdbms と compatible.asm の両方が 11.1 以上に設定されている場合、デフォルトは 12H になります。それ以外の場合、ディスクは即時に削除されます。この属性は、ALTER DISKGROUP ... OFFLINE DISK 文および DROP AFTER 句によって上書きできます。 ノート: disk_repair_time の現行の値を使用してディスクがオフラインに切り替えられ、その後この属性の値が変更された場合、変更された値が Oracle ASM によってディスク・オフライン・ロジックで使用されます。 参照: 詳細は、「ALTER DISKGROUP」の「disk_offline_clause」および『Oracle Automatic Storage Management 管理者ガイド』を参照してください。
FAILGROUP_REPAIR_TIME	<number>m (分数)または <number>h (時間数)	ディスク・グループ内の障害グループのデフォルトの修復時間を指定します。Oracle ASM で障害グループ全体が失敗したと判別される場合、障害グループの修復時間が使用されます。デフォルト値は、24 時間(24h)です。 ALTER DISKGROUP OFFLINE DISK 文の DROP AFTER 句などでディスクに指定された修復時間がある場合は、そのディスク修復時間が障害グループの修復時間に優先します。

属性	有効な値	説明
		この属性は、ディスク・グループの変更時にのみ設定でき、標準冗長性および高冗長性のディスク・グループにのみ適用できます。
LOGICAL_SECTOR_SIZE	512、4096 または 4K	ディスク・グループの論理セクター・サイズを設定します。この値は、ディスク・グループが受け入れることのできる最小許容 I/O を指定します。デフォルト値は、ディスク・グループに参加するディスクから推定されます。 このディスク・グループ属性をディスク・グループの作成中に設定するか、またはディスク・グループの作成後に変更するには、COMPATIBLE.ASM ディスク・グループ属性を 12.2 以上に設定する必要があります。
PHYS_META_REPLICATED	true または false	ディスク・グループのレプリケーション・ステータスを追跡します。ディスク・グループの Oracle ASM 互換性が 12.0 以降に昇格されていると、各ディスクの物理メタデータ(ディスク・ヘッダー、表のブロックの空き領域、表のブロックの割当てなど)がレプリケートされます。このレプリケーションは、オンライン非同期で実行されます。ディスク・グループに含まれる各ディスクの物理メタデータがレプリケートされると、Oracle ASM は PHYS_META_REPLICATED を true に設定します。 このディスク・グループ属性は、Oracle ASM ディスク・グループ互換性(COMPATIBLE.ASM)が 12.0 以上に設定されているディスク・グループでのみ定義されています。この属性は読取り専用であり、情報提供のみを目的としています。この値は、設定や変更ができません。
PREFERRED_READ.ENABLED	true または false	物理的に離れた複数サイトにわたるノードを含む Oracle 拡張クラスタでは、ディスク・グループに対して優先読取り機能を有効化するかどうかは、PREFERRED_READ.ENABLED ディスク・グループ属性によって制御します。優先読取り機能が有効化されている場合、この機能により、インスタンスがそれ自体と同じサイトにあるディスクを特定して読み取ることができるため、パフォーマンスが向上することがあります。拡張クラスタの場合、デフォルト値は true です。拡張されていないクラスタ(1 つの物理サイトのみ)の場合、優先読取りは無効(false)です。優先読取りステータスは、拡張、標準、高およびフレックス冗長性ディスク・グループに適用されます。 このディスク・グループ属性は、Oracle ASM ディスク・グループ互換性(COMPATIBLE.ASM)が 12.2 以上に設定されているディスク・グループでのみ定義されています。

属性	有効な値	説明
SECTOR_SIZE	512、4096 または 4K	ディスク・グループの物理セクター・サイズを設定します。ディスク・グループ内のすべてのディスクは、この物理セクター・サイズである必要があります。デフォルト値は、ディスク・グループに参加するディスクから取得されます。 このディスク・グループ属性をディスク・グループの作成中に設定するには、COMPATIBLE.ASM および COMPATIBLE.RDBMS ディスク・グループ属性を 11.2 以上に設定する必要があります。ディスク・グループの作成後にこのディスク・グループ属性を変更するには、COMPATIBLE.ASM ディスク・グループ属性を 12.2 以上に設定する必要があります。
THIN_PROVISIONED	true または false	ディスク・グループのリバランスの完了後に未使用の記憶域領域を破棄する機能を有効化(true)または無効化(false)します。デフォルト値は false です。
CONTENT_HARDCHECK	true または false	CONTENT_HARDCHECK は、ディスク・グループのリバランスのためにデータ・コピー操作を実行する際の Hardware Assisted Resilient Data (HARD)チェックを有効化または無効化します。この属性は、ディスク・グループの変更時にのみ設定できます。

脚注 1

有効なOracle Databaseリリース番号の最初の2桁以上を指定してください。有効なバージョン番号の指定の詳細は、『Oracle Database管理者ガイド』を参照してください。たとえば、compatibilityを'11.2'または'12.1'として指定できます。

関連項目:

これらの属性設定の管理の詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

例

次の例では、ASM_DISKSTRINGパラメータは/devices/disks/c*のスーパーセットであり、/devices/disks/c*がOracle ASMディスクとして使用される1つ以上のデバイスを指し、Oracle Databaseユーザーはディスクに対する読取り/書込み権限を持っていることを想定しています。

関連項目:

Oracle ASMおよびディスク・グループを使用してデータベース管理を簡略化する方法については、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

ディスク・グループの作成: 例

次の文は、Oracle ASMディスク・グループdgroup_01を作成します。このディスク・グループには、Oracle ASMによって冗長性が提供されておらず、search_stringと一致するすべてのディスクが含まれています。

```
CREATE DISKGROUP dgroup_01
  EXTERNAL REDUNDANCY
  DISK '/devices/disks/c*';
```

CREATE EDITION

目的

この文は、既存のエディションの子として新規エディションを作成します。エディションによって、データベース内に同じエディション化可能なオブジェクトを2バージョン以上保持できるようになります。エディションを作成すると、このエディションは、その親エディションのエディション化可能なオブジェクトのすべてを即座に継承します。次のオブジェクト型は、エディション化可能です。

- シノニム
- ビュー
- ファンクション
- プロシージャ
- パッケージ(仕様部および本体)
- タイプ(仕様部および本体)
- ライブラリ
- トリガー

エディション化可能なオブジェクトは、エディションが有効化されたスキーマ内にある、前述のいずれかのエディション化可能なオブジェクト型です。データベース内にこのような複数バージョンのオブジェクトを含めることができるため、オンラインのアプリケーション・アップグレードが大幅に簡略化されます。

ノート:

前述のリストにないすべてのデータベース・オブジェクト型は、エディション化可能ではありません。エディション化可能でないオブジェクト型に対する変更内容は、データベース内のすべてのエディションからすぐに参照可能になります。

新規作成またはアップグレードされたOracle Databaseにはそれぞれ、ORA$BASEという名前のデフォルトのエディションがあり、それは、CREATE EDITION文により作成される最初のエディションの親として機能します。その後に、ALTER DATABASE DEFAULT EDITION文を使用して、ユーザー定義のエディションをデータベースのデフォルト・エディションとして指定できます。

関連項目:

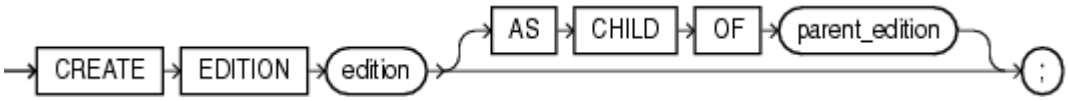
- エディション化可能なオブジェクト・タイプおよびエディションの詳細は、『Oracle Database開発ガイド』を参照してください。
- エディションをデータベースのデフォルト・エディションとして指定する方法の詳細は、「ALTER DATABASE」の「DEFAULT EDITION句」を参照してください。

前提条件

エディションを作成する場合は、直接またはロールを介して付与されたCREATE ANY EDITIONシステム権限が必要です。エディションを他のエディションの子として作成する場合は、親エディションに対するUSEオブジェクト権限が必要です。

構文

create_edition::=



セマンティクス

edition

作成するエディションの名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

データベースに対して作成済のエディションを表示する場合は、DBA_OBJECTSまたはALL_OBJECTSデータ・ディクショナリ・ビューのEDITION_NAME列を問い合せます。

エディションを作成すると、作成したエディションに対するWITH GRANT OPTION付きのUSEオブジェクト権限がシステムにより自動的に付与されます。

ノート:

エディションに、ORA、ORACLE、SYS、DBA および DBMS の接頭辞を含む名前を付けないことをお薦めします。これらの接頭辞は内部使用のために確保されているためです。

AS CHILD OF句

この句を使用した場合、新規エディションはparent_editionの子として作成されます。この句を指定しない場合、新規エディションはリーフ・エディションの子として作成されます。新規エディションは、作成時に、その親エディションからエディショニングされたすべてのオブジェクトを継承します。

エディションの制限事項

エディションには、子エディションを1つのみ含めることができます。parent_editionにすでに子エディションを持つエディションを指定した場合、エラーが戻されます。

例

次の簡単な例は、エディションの作成および使用のための構文を示すためのものです。エディションの現実的な使用例については、『Oracle Database開発ガイド』を参照してください。

次の文で、ユーザーHRには、エディションを作成および使用するために必要な権限が付与されます。

```
GRANT CREATE ANY EDITION, DROP ANY EDITION to HR;
Grant succeeded.
ALTER USER hr ENABLE EDITIONS;
User altered.
```

HRは、テストの目的で新規のエディションTEST_EDを作成します。

```
CREATE EDITION test_ed;
```

次に、HRは、最初に現行のエディションがデフォルト・エディションであることを確認し、テストの目的で、デフォルト・エディションORA$BASE内にエディショニング・ビューed_viewを作成します。

```
SELECT SYS_CONTEXT('userenv', 'current_edition_name') FROM DUAL;
SYS_CONTEXT('USERENV','CURRENT_EDITION_NAME')
--------------------------------------------------------------------------------
ORA$BASE
```

```
1 row selected.
CREATE EDITIONING VIEW e_view AS
  SELECT last_name, first_name, email FROM employees;
View created.
DESCRIBE e_view
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 LAST_NAME                                 NOT NULL VARCHAR2(25)
 FIRST_NAME                                         VARCHAR2(20)
 EMAIL                                     NOT NULL VARCHAR2(25)
```

その後、HRがTEST_EDエディションを使用して異なる書式でビューを再作成すると、ビューはTEST_EDエディション内で実体化されます。

```
ALTER SESSION SET EDITION = TEST_ED;
Session altered.
CREATE OR REPLACE EDITIONING VIEW e_view AS
  SELECT last_name, first_name, email, salary FROM employees;
View created.
```

TEST_EDエディション内のビューには、次の追加の列が含まれます。

```
DESCRIBE e_view
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 LAST_NAME                                 NOT NULL VARCHAR2(25)
 FIRST_NAME                                         VARCHAR2(20)
 EMAIL                                     NOT NULL VARCHAR2(25)
 SALARY                                             NUMBER(8,2)
```

ORA$BASEエディション内のビューは、テスト環境から孤立したままです。

```
ALTER SESSION SET EDITION = ora$base;
Session altered.
DESCRIBE e_view;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 LAST_NAME                                 NOT NULL VARCHAR2(25)
 FIRST_NAME                                         VARCHAR2(20)
 EMAIL                                     NOT NULL VARCHAR2(25)
```

テスト環境からビューを削除しても、ビューはORA$BASEエディション内に残ります。

```
ALTER SESSION SET EDITION = TEST_ED;
Session altered.
DROP VIEW e_view;
View dropped.
ALTER SESSION SET EDITION = ORA$BASE;
Session altered.
DESCRIBE e_view;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 LAST_NAME                                 NOT NULL VARCHAR2(25)
 FIRST_NAME                                         VARCHAR2(20)
 EMAIL                                     NOT NULL VARCHAR2(25)
```

TEST_EDエディションを必要とするアップグレードのテストが完了したら、エディションを削除できます。

```
DROP EDITION TEST_ED;
```

CREATE FLASHBACK ARCHIVE

目的

CREATE FLASHBACK ARCHIVE文は、フラッシュバック・アーカイブを作成する場合に使用します。フラッシュバック・アーカイブによって、指定したデータベース・オブジェクトへのトランザクションでのデータ変更を自動的に追跡およびアーカイブできます。フラッシュバック・アーカイブは複数の表領域で構成され、追跡対象の表に対するすべてのトランザクションの履歴データが格納されます。データは内部履歴表に格納されます。

フラッシュバック・データ・アーカイブには、RETENTIONパラメータで指定した期間の履歴データが格納されます。履歴データは、フラッシュバック問合せのAS OF句を使用して問い合せることができます。指定した保存期間を超えた古いアーカイブ済履歴データは、自動的に消去されます。

フラッシュバック・データ・アーカイブには、フラッシュバック・アーカイブ用に使用される表に対するデータ定義言語(DDL)の変更があっても、履歴データが保持されます。フラッシュバック・データ・アーカイブは、多くの一般的なDDL文をサポートします。これには、表の定義を変更したりデータ移動を伴ういくつかのDDL文が含まれます。サポートされていないDDL文を実行すると、ORA-55610エラーが戻されます。

関連項目:

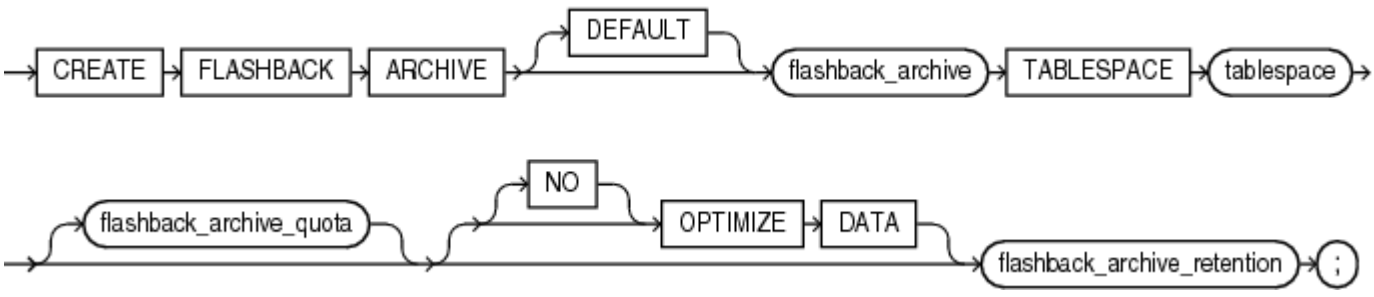
- フラッシュバック・タイム・トラベルの使用についての一般情報は、Oracle Database開発ガイドを参照してください。
- 表を追跡対象の表として指定する方法の詳細は、「CREATE TABLE」の「flashback_archive_clause」を参照してください。
- フラッシュバック・アーカイブの割当て制限属性と保存属性の変更およびフラッシュバック・アーカイブにおける表領域の記憶域の追加と変更の詳細は、ALTER FLASHBACK ARCHIVEを参照してください。

前提条件

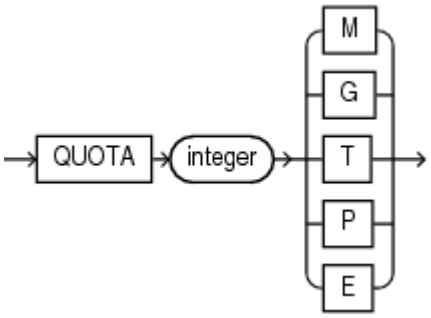
フラッシュバック・アーカイブを作成するには、FLASHBACK ARCHIVE ADMINISTERシステム権限が必要です。また、フラッシュバック・アーカイブを作成するにはCREATE TABLESPACEシステム権限が必要であり、履歴情報が格納される表領域に十分な割当て制限も必要です。フラッシュバック・アーカイブをシステムのデフォルトのフラッシュバック・アーカイブとして指定するには、SYSDBAとしてログインしている必要があります。

構文

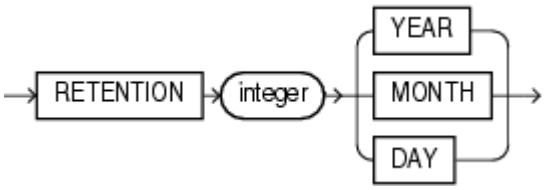
create_flashback_archive::=



flashback_archive_quota::=



flashback_archive_retention::=



セマンティクス

DEFAULT

DEFAULTを指定するには、SYSDBAとしてログインしている必要があります。この句は、このフラッシュバック・アーカイブをデータベースのデフォルトのフラッシュバック・アーカイブとして指定する場合に使用します。CREATE TABLE文またはALTER TABLE文に、フラッシュバック・アーカイブ名を指定しないでflashback_archive_clauseが指定されている場合、データベースはデフォルトのフラッシュバック・アーカイブを使用して、その表のデータを格納します。

デフォルトのフラッシュバック・アーカイブがすでに存在する場合は、この句を指定できません。ただし、ALTER FLASHBACK ARCHIVE ... SET DEFAULT句を使用して、既存のデフォルトのフラッシュバック・アーカイブを置き換えることはできます。

関連項目:

詳細は、「CREATE TABLE」の「flashback_archive_clause」を参照してください。

flashback_archive

フラッシュバック・アーカイブの名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

TABLESPACE句

このフラッシュバック・アーカイブのアーカイブ・データが格納される表領域を指定します。この句では表領域を1つのみ指定できます。ただし、ALTER FLASHBACK ARCHIVE文を使用して、後で表領域をフラッシュバック・アーカイブに追加できます。

flashback_archive_quota

アーカイブ・データ用に確保する初期表領域の領域を指定します。フラッシュバック・アーカイブ内のアーカイブ用の領域が一杯になると、このフラッシュバック・アーカイブを使用する追跡対象の表でのDML操作は失敗します。フラッシュバック・アーカイブの内容が指定された割当て制限の90%になると、領域不足のアラートがデータベースから発行され、古いデータを消去するか、割当て制限を追加する時間が確保されます。この句を省略すると、指定された表領域でフラッシュバック・アーカイブの割当て制限が無制限になります。

[NO] OPTIMIZE DATA

OPTIMIZE DATAは、フラッシュバック・アーカイブの履歴表の最適化を有効にする場合に指定します。これにより、高度な行圧縮、高度なLOB圧縮、高度なLOB重複除外、セグメント・レベルの圧縮階層化および行レベルの圧縮階層化のいずれかの機能を使用して、履歴表のデータ記憶域を最適化するようデータベースに指示します。この句を指定するには、拡張圧縮オプションのライセンスが必要です。

履歴表のデータ記憶域を最適化しないようにするには、NO OPTIMIZE DATAを指定します。これはデフォルトです。

flashback_archive_retention

アーカイブされたデータがフラッシュバック・アーカイブに保持される時間の長さを月、日または年単位で指定します。指定した時間でフラッシュバック・アーカイブがいっぱいになると、データベースはflashback_archive_quotaで説明されているように応答します。

例

次の文では、2つのフラッシュバック・アーカイブがテストのために作成されます。最初のフラッシュバック・データ・アーカイブは、データベースのデフォルトとして指定されます。どちらも、領域割当て制限は1MBであり、アーカイブの保存は1日です。

```
CREATE FLASHBACK ARCHIVE DEFAULT test_archive1
   TABLESPACE example
   QUOTA 1 M
   RETENTION 1 DAY;
CREATE FLASHBACK ARCHIVE test_archive2
   TABLESPACE example
   QUOTA 1 M
   RETENTION 1 DAY;
```

次の文では、デフォルトのフラッシュバック・アーカイブを変更して、保存期間を1か月に延長します。

```
ALTER FLASHBACK ARCHIVE test_archive1
   MODIFY RETENTION 1 MONTH;
```

次の文では、oe.customers表の追跡を指定します。フラッシュバック・アーカイブが指定されていないため、データはデフォルトのフラッシュバック・アーカイブtest_archive1にアーカイブされます。

```
ALTER TABLE oe.customers
   FLASHBACK ARCHIVE;
```

次の文では、oe.orders表の追跡を指定します。この場合、データは指定したフラッシュバック・アーカイブtest_archive2にアーカイブされます。

```
ALTER TABLE oe.orders
   FLASHBACK ARCHIVE test_archive2;
```

次の文では、test_archive2フラッシュバック・アーカイブを削除します。

```
DROP FLASHBACK ARCHIVE test_archive2;
```

CREATE FUNCTION

目的

ファンクションはPL/SQLを使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

CREATE FUNCTION文を使用すると、スタンドアロン・ストアド・ファンクションまたはコール仕様を作成できます。

- ストアド・ファンクション(ユーザー・ファンクションまたはユーザー定義ファンクション)は、名前でコールできるPL/SQL文の集合です。ストアド・ファンクションは、プロシージャとよく似ていますが、ファンクションは、コールした環境に値を戻す点で異なります。ストアド・ファンクションは、SQL式の一部として使用できます。
- コール仕様は、PL/SQLからコールできるように、Javaメソッドまたは第三世代言語(3GL)ルーチンを宣言します。CALL SQL文を使用して、そのようなメソッドまたはルーチンをコールすることもできます。コール仕様は、コールされたときに起動するJavaメソッドまたは共有ライブラリの名前付きファンクションをOracle Databaseに指示します。また、引数および戻り値に対して実行する型変換もデータベースに指示します。

ノート:

CREATE PACKAGE 文を使用して、パッケージの一部としてファンクションを作成することもできます。

関連項目:

- プロシージャおよびファンクションの概要は、「CREATE PROCEDURE」を参照してください。パッケージの作成については、「CREATE PACKAGE」を参照してください。ファンクションの変更および削除については、「ALTER FUNCTION」および「DROP FUNCTION」を参照してください。
- 共有ライブラリについては、「CREATE LIBRARY」を参照してください。
- 外部ファンクションの登録の詳細は、『Oracle Database開発ガイド』を参照してください。

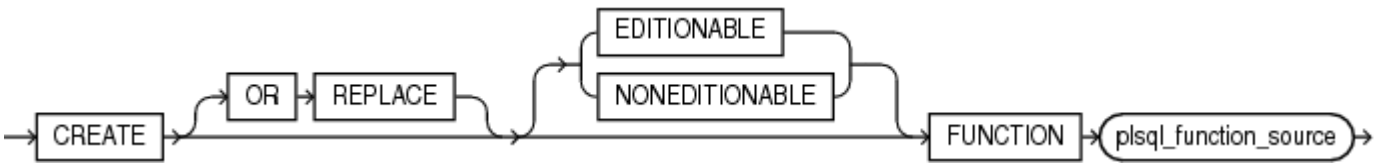
前提条件

自分のスキーマ内にファンクションを作成または再作成する場合は、CREATE PROCEDUREシステム権限が必要です。他のユーザーのスキーマ内にファンクションを作成または再作成する場合は、CREATE ANY PROCEDUREシステム権限が必要です。

構文

ファンクションはPL/SQLを使用して定義されます。このため、このマニュアルの構文図ではSQLキーワードのみを示します。PL/SQLの構文、セマンティクスおよび例については、『Oracle Database PL/SQL言語リファレンス』を参照してください。

create_function::=



(plsql_function_source: 『Oracle Database PL/SQL言語リファレンス』を参照。)

セマンティクス

OR REPLACE

OR REPLACEを指定すると、既存のファンクションを再作成できます。この句を指定した場合、既存のファンクションに付与されているオブジェクト権限を削除、再作成および再付与しなくても、そのファンクションの定義を変更できます。ファンクションを再定義した場合、そのファンクションは再コンパイルされます。

再定義したファンクションに対して権限が付与されていたユーザーは、権限を再付与されなくても、そのファンクションにアクセスできます。

ファンクション索引がファンクションに依存している場合、索引にDISABLEDのマークが付きます。

関連項目:

SQLを使用したファンクションの再コンパイルについては、「ALTER FUNCTION」を参照してください。

[EDITIONABLE | NONEDITIONABLE]

この句を使用すると、schemaのスキーマ・オブジェクト・タイプFUNCTIONのエディショニングが有効化されたときに、そのファンクションをエディション・オブジェクトにするか非エディション・オブジェクトにするかを指定できます。デフォルトは、EDITIONABLEです。エディション・オブジェクトと非エディション・オブジェクトの詳細は、『Oracle Database開発ガイド』を参照してください。

plsql_function_source

plsql_function_sourceの構文、セマンティクスおよび例については、『Oracle Database PL/SQL言語リファレンス』を参照してください。

CREATE HIERARCHY

目的

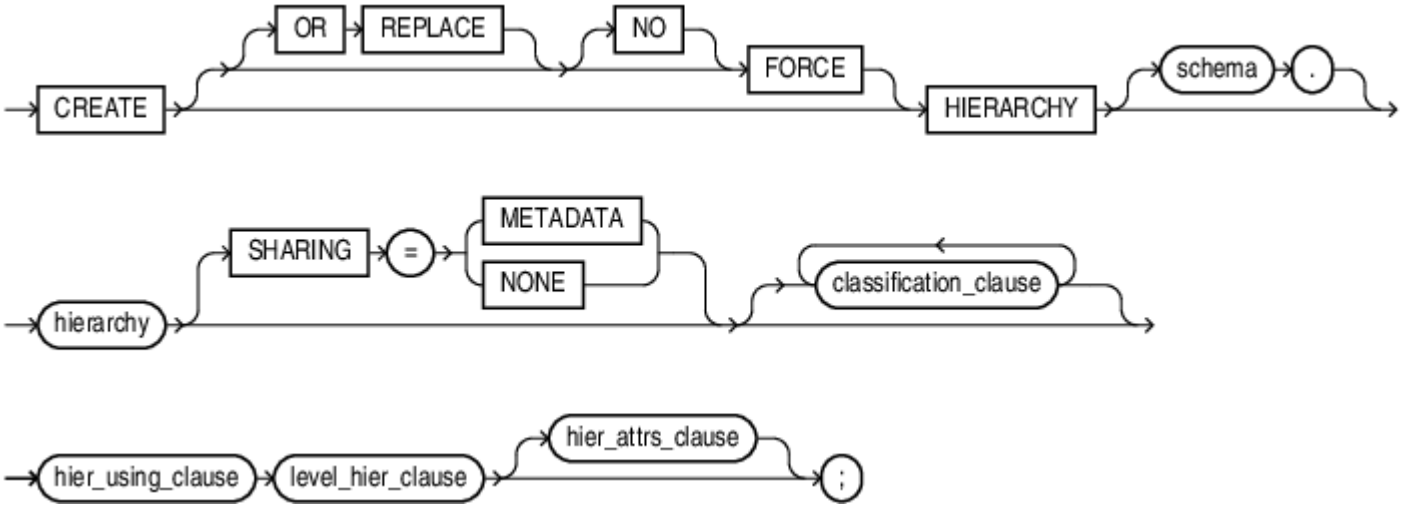
CREATE HIERARCHY文を使用して、階層を作成します。階層では、属性ディメンションのレベル間の階層関係を指定します。

前提条件

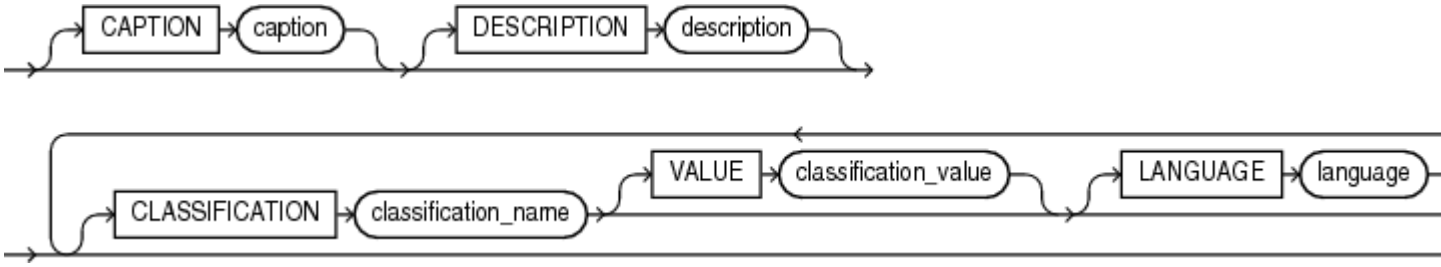
自分のスキーマ内に階層を作成する場合は、CREATE HIERARCHYシステム権限が必要です。別のユーザーのスキーマに階層を作成する場合は、CREATE ANY HIERARCHYシステム権限が必要です。

構文

create_hierarchy::=

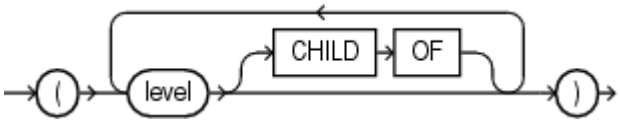


classification_clause::=

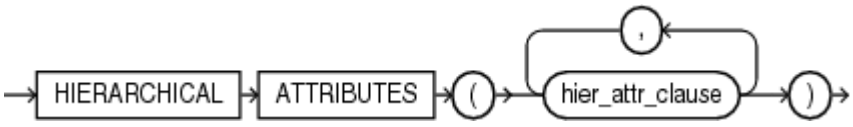


hier_using_clause::=

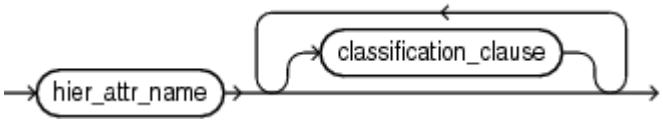
level_hier_clause::=



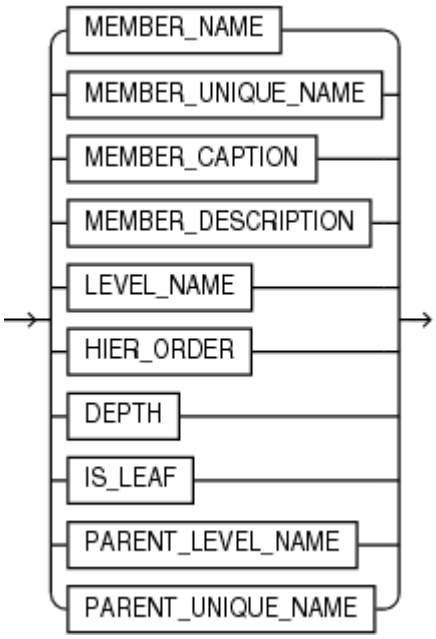
hier_attrs_clause::=



hier_attr_clause::=



hier_attr_name::=



セマンティクス

OR REPLACE

階層の既存の定義を異なる定義で置換するには、OR REPLACEを指定します。

FORCEおよびNOFORCE

正常にコンパイルされない場合でも強制的に階層を作成するには、FORCEを指定します。NOFORCEを指定した場合、階層を正常にコンパイルする必要があり、そうでない場合はエラーが発生します。デフォルトは、NOFORCEです。

schema

階層を作成するスキーマを指定します。スキーマを指定しない場合、自分のスキーマに階層が作成されます。

階層

階層の名前を指定します。

SHARING

SHARING句は、アプリケーションのメンテナンスの際に、アプリケーション・ルートにオブジェクトを作成する場合に使用します。このタイプのオブジェクトはアプリケーション共通オブジェクトと呼ばれ、アプリケーション・ルートに属するアプリケーションPDBと共有できます。

次の共有属性のいずれかを使用して、オブジェクトを共有する方法を指定できます。

- METADATA - メタデータ・リンクはメタデータを共有しますが、そのデータは各コンテナに固有です。このタイプのオブジェクトは、メタデータリンク・アプリケーション共通オブジェクトと呼ばれます。
- NONE - オブジェクトは共有されず、アプリケーション・ルートでのみアクセスできます。

classification_clause

classification句を使用して、CAPTIONまたはDESCRIPTION分類の値を指定し、ユーザー定義分類を指定します。分類では、分析ビューとそのコンポーネントの情報を提供するためにアプリケーションで使用される、説明的なメタデータを提供します。

同じオブジェクトに、任意の数の分類を指定できます。分類の最大長は4000バイトです。

CAPTIONとDESCRIPTION分類では、DDLショートカットCAPTION 'caption'およびDESCRIPTION 'description'または完全な分類構文を使用できます。

言語別に分類値を変更できます。CAPTIONまたはDESCRIPTION分類に言語を指定するには、完全な構文を使用する必要があります。言語を指定しない場合、分類の言語の値はNULLです。言語の値は、NULLまたは有効なNLS_LANGUAGE値である必要があります。

hier_using_clause

階層のメンバーを持つ属性ディメンションを指定します。

level_hier_clause

階層レベルの編成を指定します。

hier_attrs_clause

階層属性の説明的なメタデータを含む分類を指定します。特定のhier_attr_nameのhier_attr_clauseは、リストに1回のみ出現できます。

すべての階層には常に、すべての階層属性が含まれますが、この句で指定した場合を除き、階層属性には関連付けられている説明的なメタデータはありません。

hier_attr_clause

階層属性を指定して、1つ以上の分類を指定します。

hier_attr_name

階層属性を指定します。

例

次の例では、TIME_HIER階層を作成します。

```
CREATE OR REPLACE HIERARCHY time_hier  -- Hierarchy name
USING time_attr_dim               -- Refers to TIME_ATTR_DIM attribute dimension
 (month CHILD OF                  -- Months in the attribute dimension
 quarter CHILD OF
 year);
```

次の例では、PRODUCT_HIER階層を作成します。

```
CREATE OR REPLACE HIERARCHY product_hier
USING product_attr_dim
 (category
  CHILD OF department);
```

次の例では、GEOGRAPHY_HIER階層を作成します。

```
CREATE OR REPLACE HIERARCHY geography_hier
USING geography_attr_dim
 (state_province
  CHILD OF country
  CHILD OF region);
```

CREATE INDEX

目的

CREATE INDEX文を使用すると、次のものに索引を作成できます。

- 表の1つ以上の列、パーティション表、索引構成表またはクラスタ
- 表またはクラスタの1つ以上のスカラー型オブジェクト属性
- ネストした表の列の索引を作成するためのネストした表の記憶表

索引は、表またはクラスタの索引列に表示されるそれぞれの値のエントリが含まれるスキーマ・オブジェクトであり、行に対する直接かつ高速のアクセスを提供します。単一の索引エントリの最大サイズは、データベースのブロック・サイズによって異なります。

Oracle Databaseは次のタイプの索引をサポートしています。

- 通常の索引。(デフォルトではBツリー索引が作成されます。)
- ビットマップ索引。キー値に関連付けられたROWIDをビットマップとして格納します。
- パーティション索引。表の索引付き列に表示される各値のエントリを含むパーティションで構成されます。
- ファンクション索引。式をベースとしています。式によって戻される値を評価する問合せを構成できます。その式には、組込みファンクションまたはユーザー定義ファンクションを含めることができます。
- ドメイン索引。アプリケーション固有のindextype索引タイプのインスタンスです。

ノート:

- 索引については、『Oracle Database 概要』を参照してください。
- 索引サイズに関連する制限の詳細は、『Oracle Database リファレンス』を参照してください。
- 「ALTER INDEX」および「DROP INDEX」を参照してください。

前提条件

自分のスキーマ内に索引を作成する場合は、次のいずれかの条件が満たされている必要があります。

- 索引を作成する表またはクラスタが自分のスキーマ内に定義されている。
- 索引を作成する表に対するINDEXオブジェクト権限を持っている。
- CREATE ANY INDEXシステム権限を持っている。

他のユーザーのスキーマ内に索引を作成する場合は、CREATE ANY INDEXシステム権限が必要です。また、索引が定義されているスキーマの所有者には、UNLIMITED TABLESPACEシステム権限、あるいはその索引または索引パーティションを格納するための表領域の割当て制限のいずれかが必要です。

ファンクション索引を作成する場合は、従来索引の作成の前提条件の他に、索引がユーザー定義ファンクションに基づいている場合は、ファンクションにDETERMINISTICのマークを付ける必要があります。ファンクション索引は、索引の所有者の資格証明を使用して実行されるため、索引の所有者には、ファンクションに対するEXECUTEオブジェクト権限が必要です。

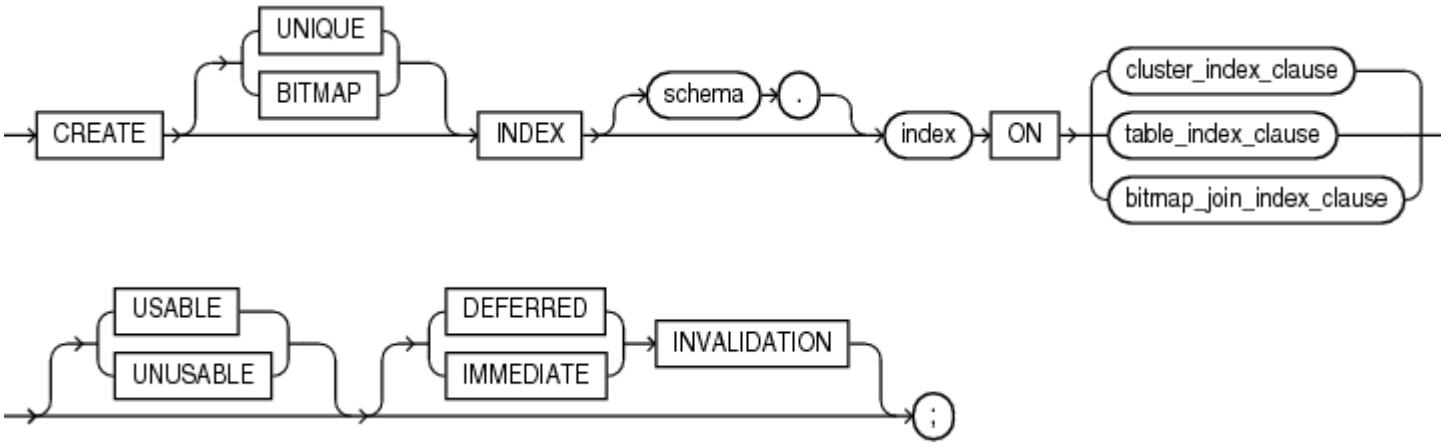
自分のスキーマにドメイン索引を作成する場合、従来索引の作成の前提条件の他に、索引タイプに対するEXECUTEオブジェクト権限が必要です。他のユーザーのスキーマにドメイン索引を作成する場合、索引の所有者にも索引タイプおよびその基礎となる実装タイプに対するEXECUTEオブジェクト権限が必要です。ドメイン索引を作成する前に、索引タイプを定義する必要があります。

関連項目:

CREATE INDEXTYPE

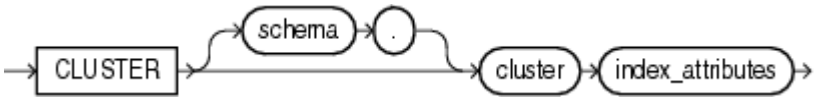
構文

create_index::=



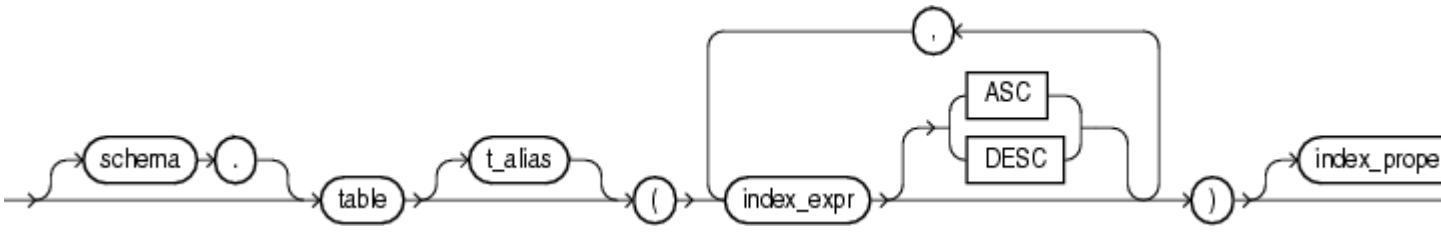
(cluster_index_clause::=、table_index_clause::=、bitmap_join_index_clause::=)

cluster_index_clause::=



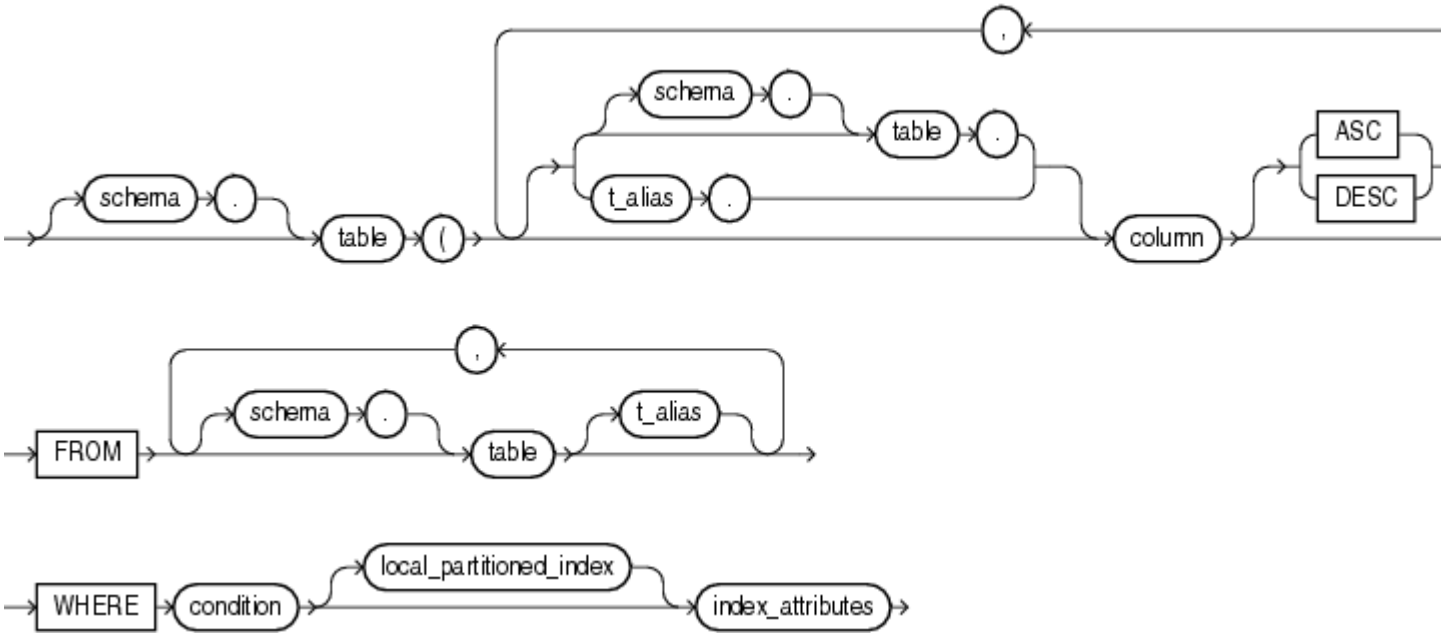
(index_attributes::=)

table_index_clause::=



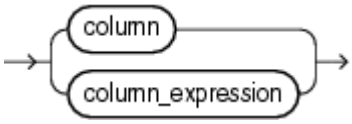
(index_properties::=)

bitmap_join_index_clause::=

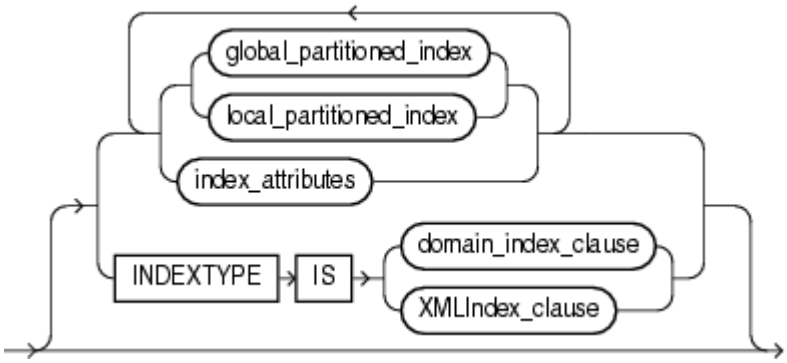


(local_partitioned_index::=、index_attributes::=)

index_expr::=

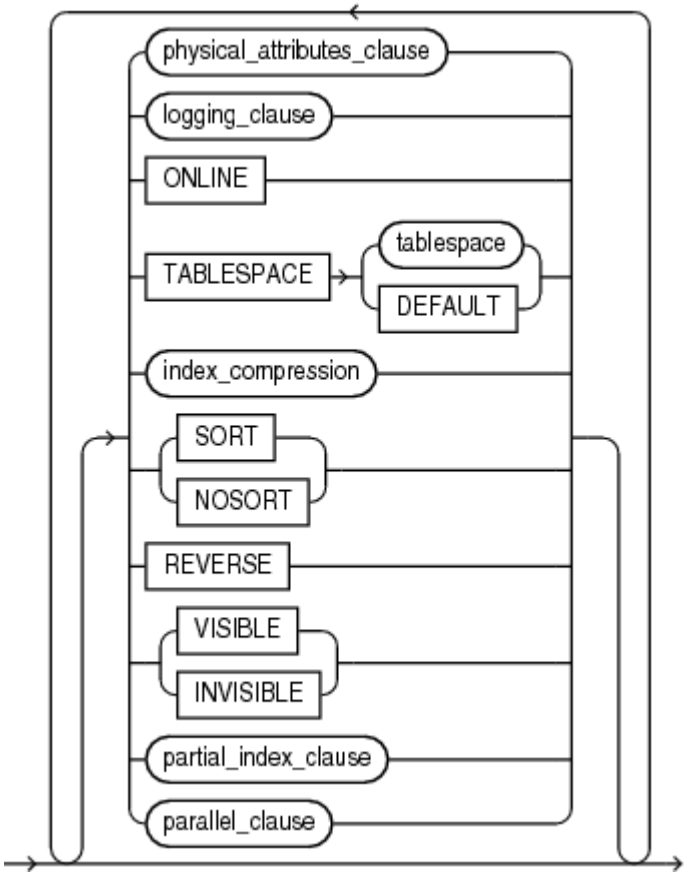


index_properties::=



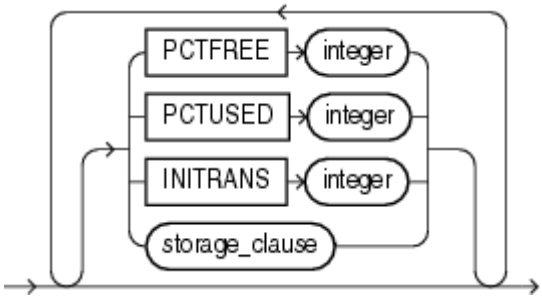
(global_partitioned_index::=、local_partitioned_index::=、index_attributes::=、domain_index_clause::=、XMLIndex_clause::=)

index_attributes::=



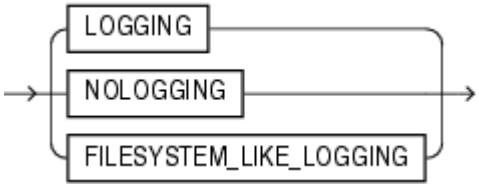
(physical_attributes_clause::=、logging_clause::=、index_compression::=、partial_index_clause::=、parallel_clause::=)

physical_attributes_clause::=

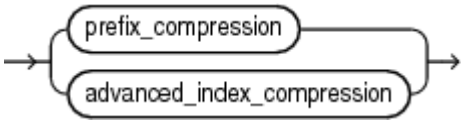


(storage_clause::=)

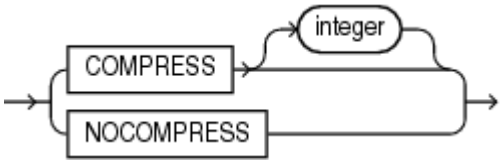
logging_clause::=



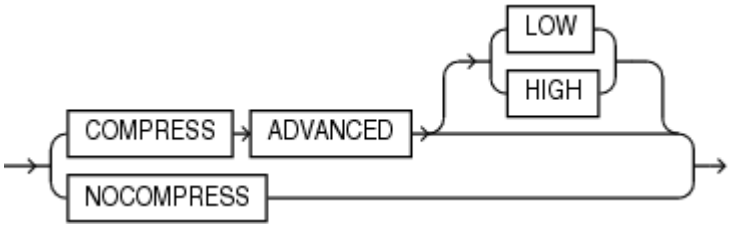
index_compression::=



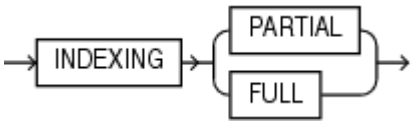
prefix_compression::=



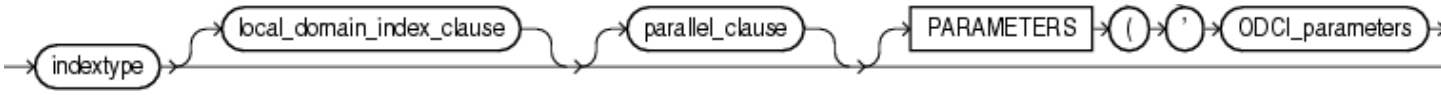
advanced_index_compression::=



partial_index_clause::=

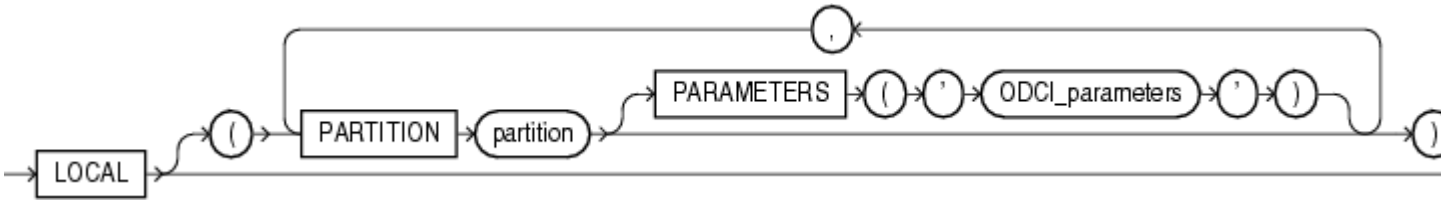


domain_index_clause::=

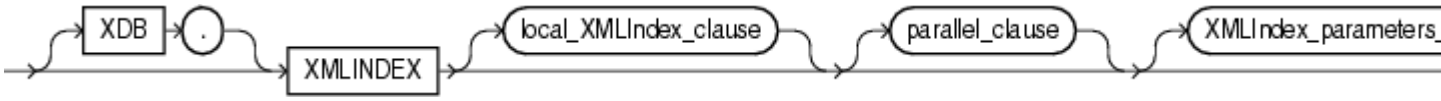


(parallel_clause::=)

local_domain_index_clause::=

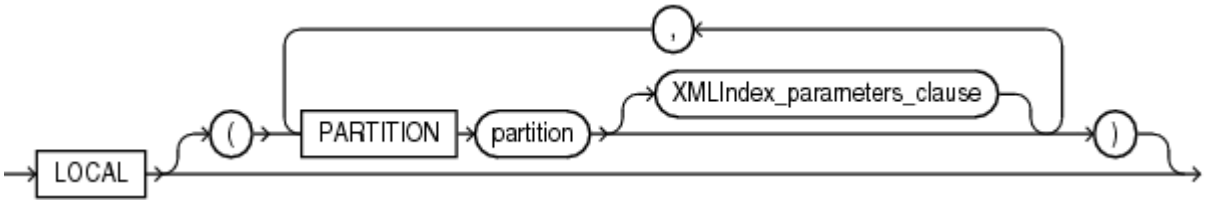


XMLIndex_clause::=



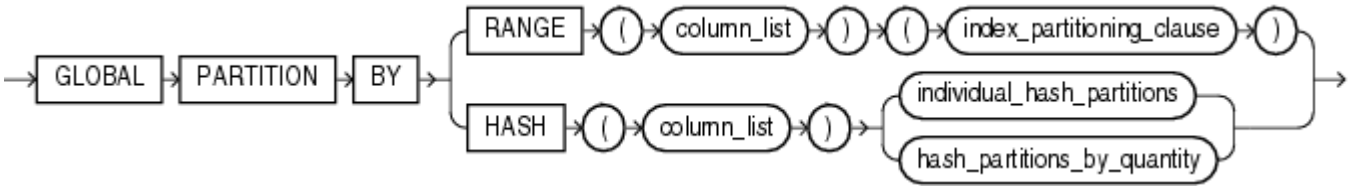
(XMLIndex_parameters_clauseについては、『Oracle XML DB開発者ガイド』を参照してください。)

local_XMLIndex_clause::=



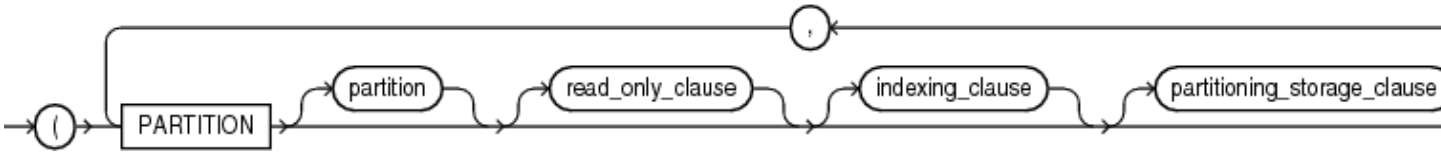
(XMLIndex_parameters_clauseについては、『Oracle XML DB開発者ガイド』を参照してください。)

global_partitioned_index::=



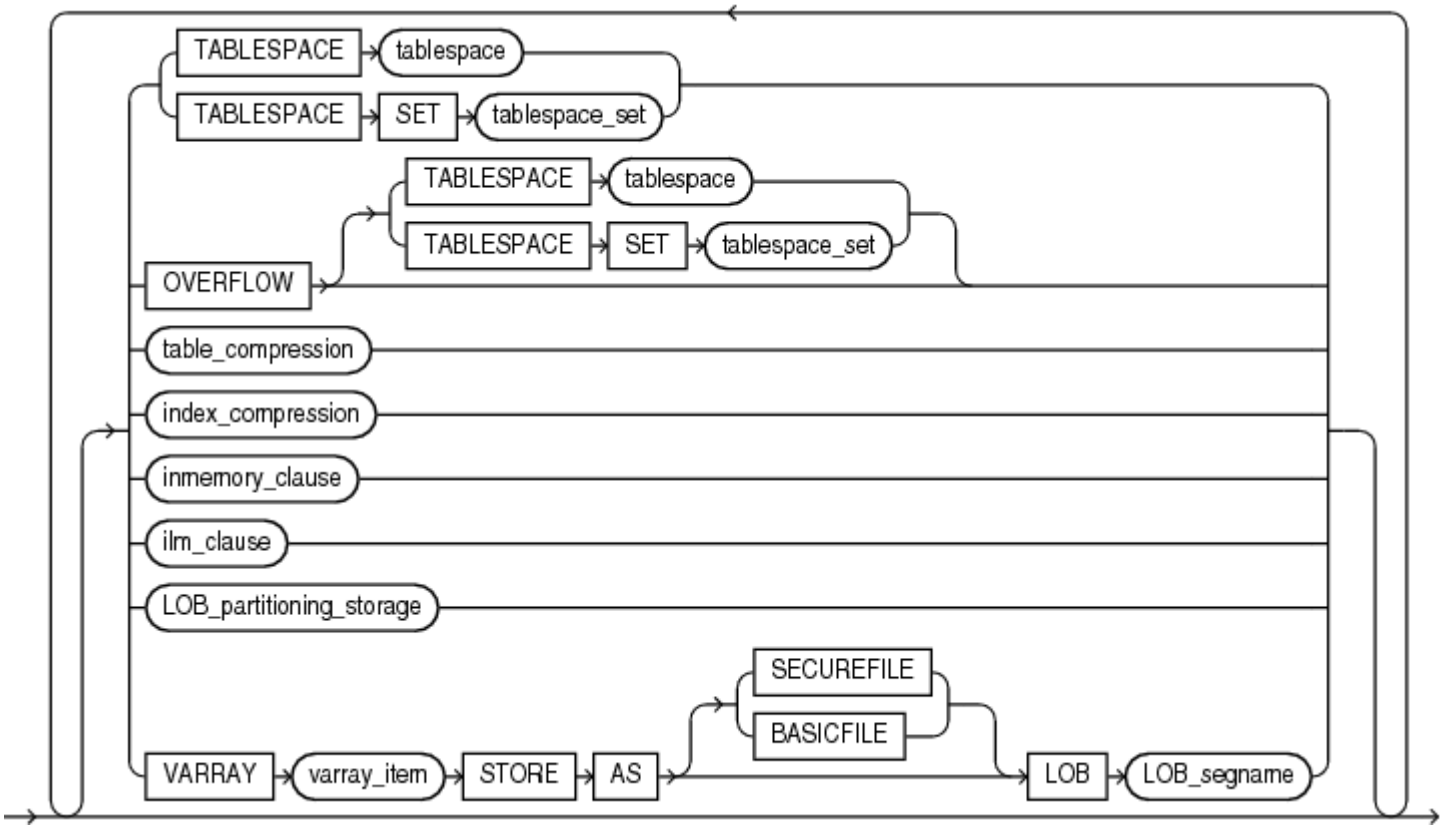
(index_partitioning_clause::=、individual_hash_partitions::=、hash_partitions_by_quantity::=)

individual_hash_partitions::=



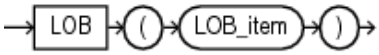
(read_only_clauseおよびindexing_clauseは、table_index_clause、partitioning_storage_clause::=ではサポートされていません。)

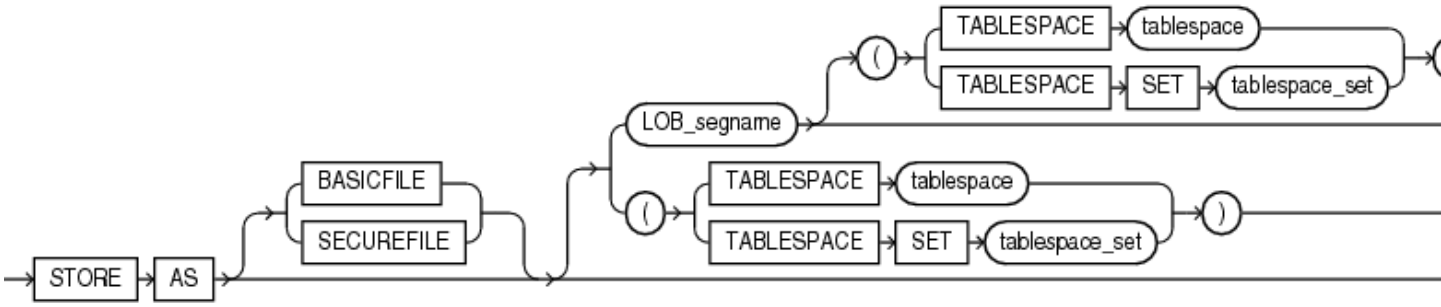
partitioning_storage_clause::=



(TABLESPACE SET、table_compression、inmemory_clauseおよびilm_clauseは、CREATE INDEX、index_compression::=、LOB_partitioning_storage::=ではサポートされていません。)

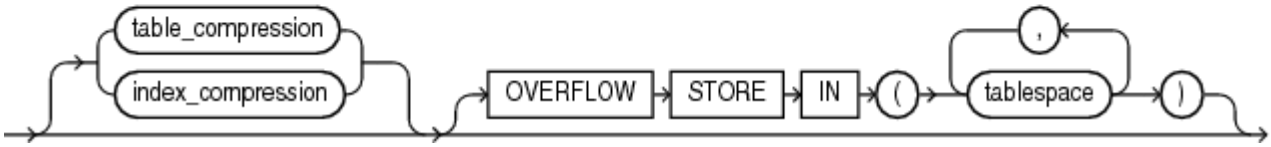
LOB_partitioning_storage::=



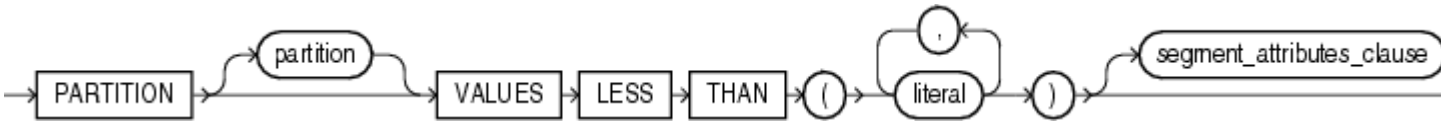


(TABLESPACE SET: CREATE INDEXではサポートされていません)

hash_partitions_by_quantity::=を参照

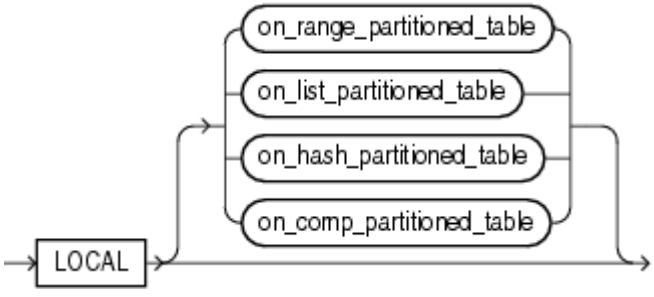


index_partitioning_clause::=



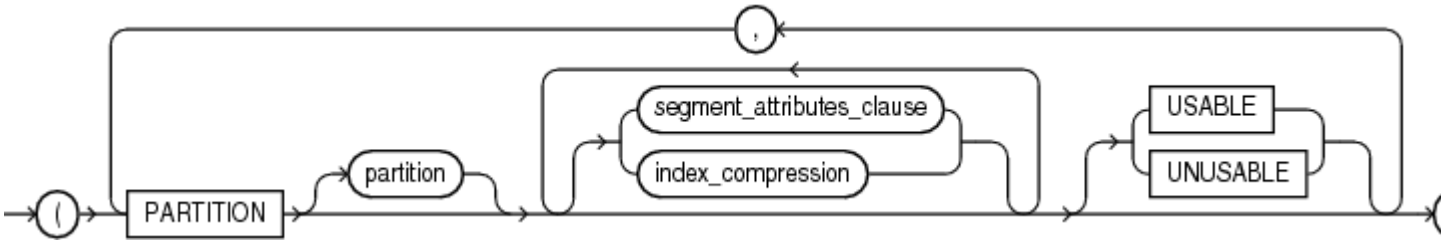
(segment_attributes_clause::=)

local_partitioned_index::=



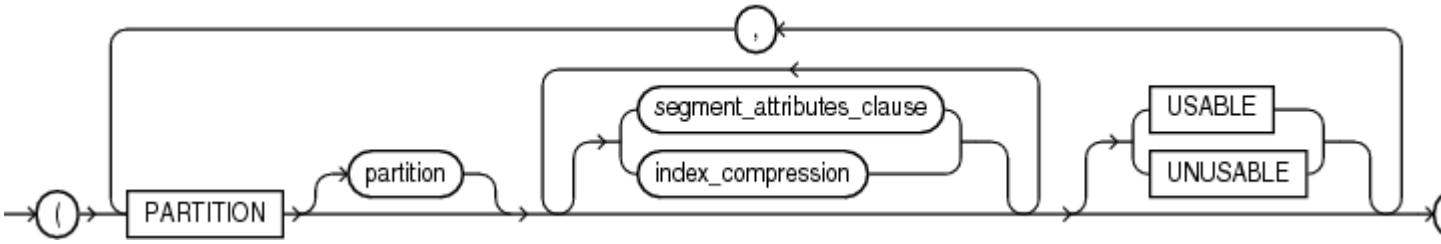
(on_range_partitioned_table::=、on_list_partitioned_table::=、on_hash_partitioned_table::=、on_comp_partitioned_table::=)

on_range_partitioned_table::=



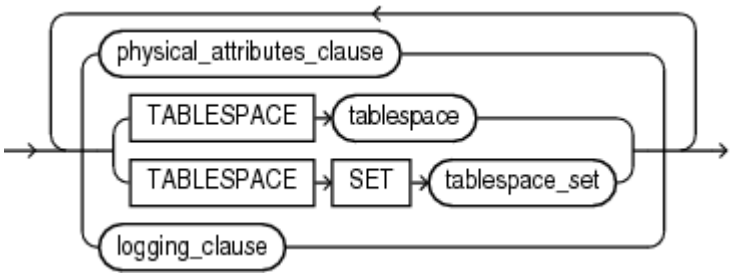
(segment_attributes_clause::=)

on_list_partitioned_table::=



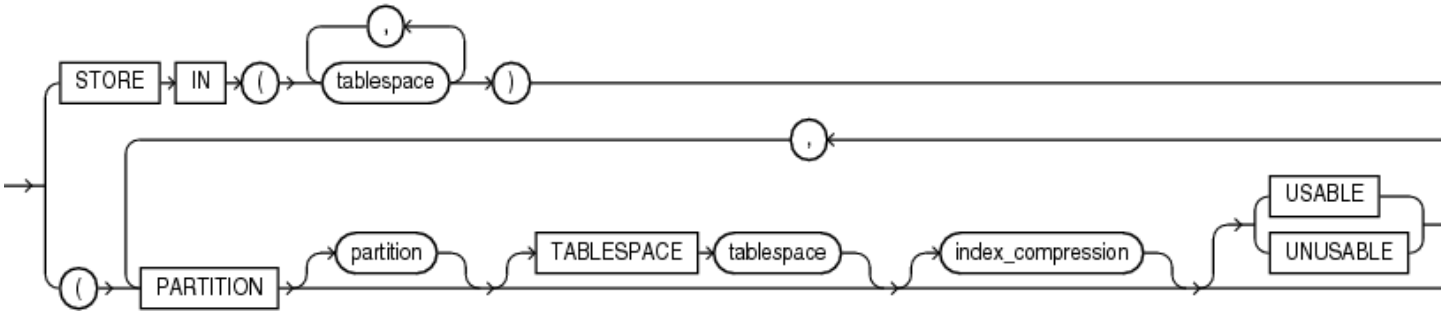
(segment_attributes_clause::=)

segment_attributes_clause::=

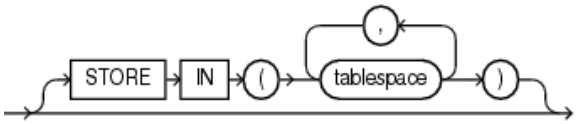


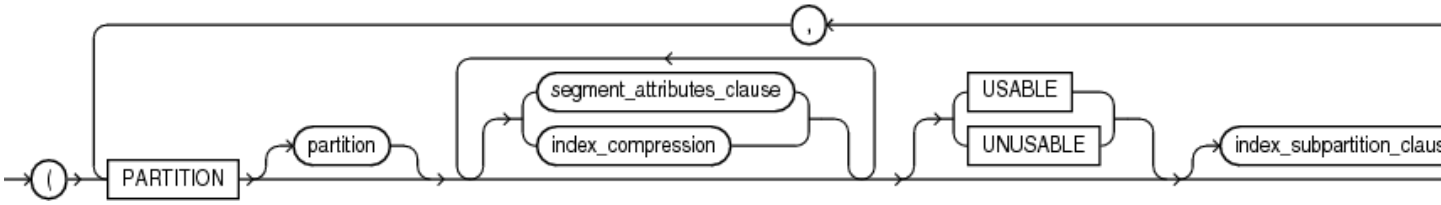
(physical_attributes_clause::=、TABLESPACE SET: CREATE INDEXではサポートされていません、logging_clause::=)

on_hash_partitioned_table::=



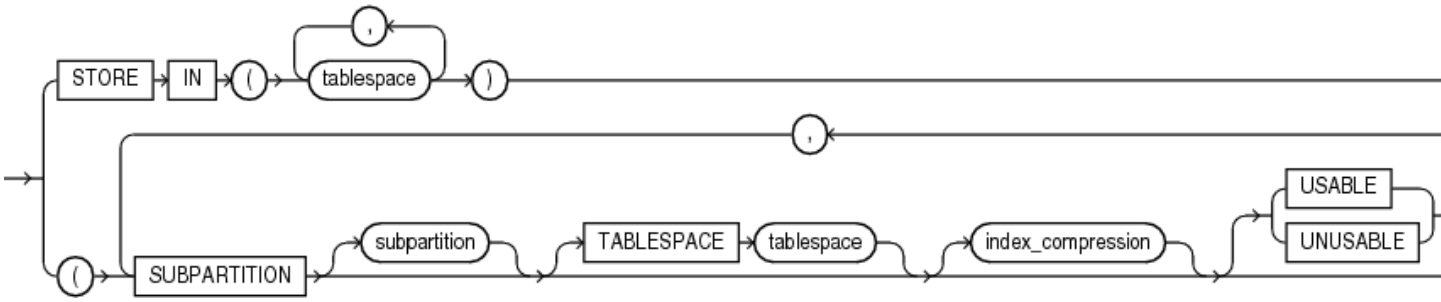
on_comp_partitioned_table::=



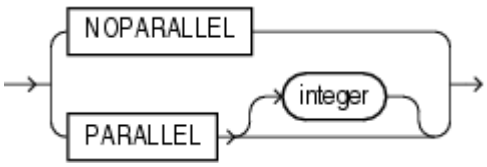


(segment_attributes_clause::=、index_compression::=、index_subpartition_clause::=)

index_subpartition_clause::=



parallel_clause::=



セマンティクス

UNIQUE

UNIQUEを指定すると、索引のベースとなっている列の値が一意である必要があることを指定できます。

一意索引の制限事項

一意索引には、次の制限事項があります。

- UNIQUEおよびBITMAPは同時に指定できません。
- ドメイン索引にはUNIQUEを指定できません。

関連項目:

一意制約を満たす条件の詳細は、「一意制約」を参照してください。

BITMAP

BITMAPを指定すると、indexが各行を分割した索引付けではなく、各個別キーのビットマップで作成されることを指定できます。ビットマップ索引では、キー値にビットマップとして関連付けられたROWIDが格納されます。ビットマップ内の各ビットは、使用可能なROWIDに対応しています。ビットが設定されていれば、それに対応するROWIDを持つ行に、キー値が設定されている

ことになります。ビットマップの内部表現は、データ・ウェアハウスなど、低レベルの同時実行トランザクションが実行されるアプリケーションに最適です。

ノート:

Oracle では、すべてのキー列が NULL の表の行には索引を付けません(ただし、ビットマップ索引の場合を除きます)。したがって、表のすべての行に索引を作成する場合、索引のキー列に NOT NULL 制約を指定するか、またはビットマップ索引を作成する必要があります。

ビットマップ索引の制限事項

ビットマップ索引には、次の制限事項があります。

- グローバル・パーティション索引の作成にはBITMAPを指定できません。
- 索引構成表がビットマップ化した2次索引に対応するマッピング表を持たない場合、索引構成表にビットマップ化した2次索引を作成できません。
- UNIQUEおよびBITMAPは同時に指定できません。
- ドメイン索引にはBITMAPを指定できません。
- ビットマップ索引には最大30列を指定できます。

関連項目:

- ビットマップ索引の使用の詳細は、『Oracle Database概要』および『Oracle Database SQLチューニング・ガイド』を参照してください。
- マッピング表の詳細は、「CREATE TABLE」を参照してください。
- 「ビットマップ索引の例」

schema

索引を作成するスキーマを指定します。schemaを指定しない場合、自分のスキーマ内に索引が作成されます。

index

作成する索引の名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

関連項目:

「索引の作成: 例」および「XMLType表の索引の作成: 例」

cluster_index_clause

cluster_index_clauseを使用すると、クラスタ索引を作成するクラスタを指定できます。clusterをschemaで修飾しない場合、そのクラスタは自分のスキーマ内にあるとみなされます。ハッシュ・クラスタにはクラスタ索引を作成できません。

関連項目:

「CREATE CLUSTER」および「クラスタ索引の作成：例」を参照してください。

table_index_clause

索引を定義する表を指定します。tableをschemaで修飾しない場合、その表は自分のスキーマにあるとみなされます。

ネストした表の記憶表に索引を作成することによって、ネストした表の列に索引を作成します。記憶表のNESTED_TABLE_ID疑似列を組み込んだUNIQUE索引を作成することは、ネストした表の値を持つ行がそれぞれ確実に異なるようにする有効な手段です。

関連項目:

ネストした表の索引：例

セッションがバインドされていない場合にのみ、一時表でDDL操作(ALTER TABLE、DROP TABLE、CREATE INDEXなど)を実行できます。セッションを一時表にバインドするには、一時表でINSERT操作を実行します。セッションを一時表からアンバインドするには、TRUNCATE文を発行するか、セッションを終了します。また、トランザクション固有の一時表からアンバインドするには、COMMITまたはROLLBACK文を発行します。

table_index_clauseの制限事項

この句には、次の制限事項があります。

- indexがローカル・パーティション索引の場合は、tableをパーティション化する必要があります。
- tableが索引構成表の場合、この文は2次索引を作成します。この索引には、索引構成表の索引キーおよび論理ROWIDが含まれます。論理ROWIDは、索引キーにも含まれる列を除外します。この2次索引には、REVERSEを指定できません。索引キーおよび論理ROWIDの結合サイズは、ブロック・サイズ未満にする必要があります。
- tableが一時表の場合、indexもtableと同様の有効範囲(セッションまたはトランザクション)を持つ一時的なものとなります。一時表の索引には、次の制限があります。
  - index_propertiesで指定できるのは、index_attributesの部分のみです。
  - index_attributesでは、physical_attributes_clause、parallel_clause、logging_clauseまたはTABLESPACEを指定できません。
  - 一時表には、ドメイン索引またはパーティション索引は作成できません。
- 外部表に索引を作成することはできません。

関連項目:

一時表の詳細は、「CREATE TABLE」および『Oracle Database概要』を参照してください。

t_alias

索引を作成する表に対して相関名(別名)を指定します。

ノート:

index_expr がオブジェクト型属性またはオブジェクト型メソッドを参照する場合、この別名が必要になります。「型メソッドのファンクション索引の作成：例」および「置換可能な列の索引の作成：例」を参照してください。

index_expr

index_exprでは、索引のベースとなる列または列式を指定します。

次の条件を満たしている場合には、同じ列のセット、同じ列式のセットまたはその両方に、複数の索引を作成できます。

- 索引の種類が異なる場合、索引に異なるパーティション化を使用する場合、または索引に異なる一意のプロパティがある場合。
- 常にVISIBLEである索引が1つのみである場合。

関連項目:

複数の索引の作成の詳細は、『Oracle Database管理者ガイド』を参照してください。

column

表内の1つ以上の列の名前を指定します。ビットマップ索引には最大30列を指定できます。他の索引には最大32列を指定できます。これらの列は索引キーを定義します。

一意索引がローカル非同一キー索引(「local_partitioned_index」を参照)の場合、索引キーはパーティション・キーを含んでいる必要があります。

関連項目:

同一キー索引および非同一キー索引の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

スカラー・オブジェクト属性列またはネストした表の記憶表のシステム定義のNESTED_TABLE_ID列には索引を作成できます。オブジェクト属性列を指定する場合、列名を表名で修飾する必要があります。ネストした表の列属性を指定する場合は、この属性は、一番外側の表の名前、ネストした表が定義されている列の名前、およびそのネストした表の列属性となるすべての中間属性の名前で修飾する必要があります。

BINARY以外の宣言または導出された名前付き照合を使用して、あるいは宣言または導出された疑似照合USING_NLS_SORT_CIやUSING_NLS_SORT_AIを使用して列または式に索引を作成する場合、ファンクションNLSSORTに対してファンクション索引が作成されます。詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

拡張データ型の列に対する索引の作成

columnが拡張データ型列の場合は、索引を作成しようとすると、「最大キー長を超過しました」というエラーが通知されることがあります。索引の最大キー長は、データベース・ブロック・サイズと、ブロック内に格納された追加の索引メタデータによって異なります。たとえば、Oracle標準の8Kのブロック・サイズを使用するデータベースでは、最大キー長は約6400バイトです。

この状況を回避するには、索引付けする値の長さを次のいずれかの方法で短くする必要があります。

- ファンクション索引を作成し、その索引定義に使用される式の一部である拡張データ型の列に格納される値を短くします。

- 仮想列を作成して、その仮想列定義に使用される式の一部である拡張データ型の列に格納される値を短くし、その仮想列で通常の索引を構築します。仮想列の使用時には、統計の収集や制約およびトリガーの使用など、通常の索引用の機能を活用することもできます。

どちらの方法でも、SUBSTRまたはSTANDARD_HASHファンクションを使用して拡張データ型の列の値を短くし、索引を構築できます。これらの方法には、次のメリットとデメリットがあります。

- SUBSTRファンクションを使用すると、索引キーに許容される長さのcolumnのサブストリングまたは接頭辞が返されます。このタイプの索引は、元の列に対する等価性、INリストおよび範囲述語用に使用できます。この場合、述語の一部としてSUBSTR列を指定する必要はありません。詳細は、「SUBSTR」を参照してください。
- STANDARD_HASHファンクションを使用した場合、サブストリングを基にした索引よりも縮小された索引が作成される可能性があり、その結果、索引への不要なアクセスが減少することがあります。このタイプの索引は、元の列に対する等価性およびINリスト述語用に使用できます。この場合、述語の一部としてSTANDARD_HASH列を指定する必要はありません。詳細は、「STANDARD_HASH」を参照してください。

次の例は、拡張データ型の列にファンクション索引を作成する方法を示しています。

```
CREATE INDEX index ON table (SUBSTR(column, 1, n));
```

nには、columnのそれぞれの値を区別するために十分な接頭辞の長さを指定します。

次の例は、拡張データ型の列の仮想列を作成し、その後この仮想列に索引を作成する方法を示しています。

```
ALTER TABLE table ADD (new_hash_column AS (STANDARD_HASH(column)));
CREATE INDEX index ON table (new_hash_column);
```

関連項目:

拡張データ型の詳細は、「拡張データ型」を参照してください。

索引列の制限事項

索引列には、次の制限事項が適用されます。

- Oracle Databaseでサポートされている、SCOPE句で定義したREF型の列または属性の索引を除き、ユーザー定義型、LONG、LONG RAW、LOBまたはREF型の列または属性に索引を作成できません。
- 暗号化された列には、標準の(Bツリー)索引しか作成できません。これらの索引は、等価検索にのみ使用できます。

column_expression

tableの列、定数、SQLファンクションおよびユーザー定義ファンクションから作成された式を指定します。column_expressionを指定した場合、ファンクション索引が作成されます。

関連項目:

「列式」、「ファンクション索引のノート」、「ファンクション索引の制限事項」および「ファンクション索引の例」を参照してください。

ファンクションの名前解決は、索引作成者のスキーマに基づきます。column_expressionで使用されるユーザー定義ファンクションは、CREATE INDEX操作中に完全に名前解決されます。

ファンクション索引の作成後、DBMS_STATSパッケージを使用して、索引と実表の両方に関する統計情報を収集します。これらの統計によって、いつ索引を使用するかをOracle Databaseで正確に決定できます。

ファンクションの一意索引は、列または列の組合せに対して条件付きの一意制約を定義する場合に便利なことがあります。例は、「条件付き一意性を定義するためのファンクション索引の使用方法：例」を参照してください。

関連項目:

DBMS_STATSパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

ファンクション索引のノート

ファンクション索引には、次のノートが適用されます。

- ファンクション索引を使用する表を後で問い合せる場合、問合せでNULLが除外されないかぎり索引は使用されません。ただし、WHERE句に指定した列の順序が、ファンクション索引を定義したcolumn_expressionでの順序と異なる場合でも、問合せにファンクション索引が使用されます。

  関連項目:

  「ファンクション索引の例」

- 索引のベースになるファンクションが無効であるか、または削除された場合は、索引にDISABLEDのマークが付けられます。オプティマイザが索引の使用を選択した場合、DISABLED索引の問合せは失敗します。索引にUNUSABLEのマークも付けて、パラメータSKIP_UNUSABLE_INDEXESをtrueに設定しないかぎり、DISABLED索引のDML操作は失敗します。このパラメータの詳細は、「ALTER SESSION」を参照してください。
- ファンクション、パッケージまたは型のパブリック・シノニムがcolumn_expressionで使用され、後で同じ名前の実際のオブジェクトが表の所有者のスキーマに作成された場合、ファンクション索引は使用禁止になります。その後、ALTER INDEX ... ENABLEまたはALTER INDEX ... REBUILDを使用してファンクション索引を使用可能にした場合、column_expressionで使用されているファンクション、パッケージまたは型は、引き続き、パブリック・シノニムが最初に指していたファンクション、パッケージまたは型に変換されます。新しいファンクション、パッケージまたは型への変換は行われません。
- ファンクション索引の定義によって文字データに内部変換が生成される場合に、NLSパラメータの設定を変更するときは注意が必要です。ファンクション索引は、NLSパラメータの現行のデータベース設定を使用します。セッション・レベルでパラメータを再設定した場合、ファンクション索引を使用して問合せを行うと、無効な結果が戻る場合があります。2つの照合パラメータ(NLS_SORTおよびNLS_COMP)は例外です。これらがセッション・レベルで再設定された場合でも、Oracle Databaseは正常に変換を処理します。
- 変換が明示的に要求された場合でも、Oracle Databaseはすべての場合にデータを変換できるわけではありません。たとえば、TO_NUMBERファンクションを使用して文字列'105 lbs'をVARCHAR2からNUMBERに変換しようとすると、エラーが発生して失敗します。したがって、column_expressionにTO_NUMBERやTO_DATEなどのデータ変換ファンクションが含まれており、後続のINSERTまたはUPDATE文に変換ファンクションで変換できないデータが含まれている場合、INSERTまたはUPDATE文は索引が原因で失敗します。
- column_expressionに日時書式モデルが含まれている場合、列を定義するファンクション索引式では、指定の要素とは異なる書式要素が使用されることがあります。たとえば、次のように、yyyy日時書式要素を使用してファンクション索引を定義します。

```
CREATE INDEX cust_eff_ix ON customers
  (NVL(cust_eff_to, TO_DATE('9000-01-01 00:00:00', 'yyyy-mm-dd hh24:mi:ss')));
```

ALL_IND_EXPRESSIONSビューを問い合せると、列を定義するファンクション索引式でsyyyy日時書式要素が使用されていることがわかります。

```
SELECT column_expression
  FROM all_ind_expressions
  WHERE index_name='CUST_EFF_IX';
COLUMN_EXPRESSION
--------------------------------------------------------------------------------
NVL("CUST_EFF_TO",TO_DATE(' 9000-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss'))
```

ファンクション索引の制限事項

ファンクション索引には、次の制限事項があります。

- column_expressionで参照されるファンクションが戻す値には、Bツリー索引の索引列と同じ制限があります。「索引列の制限事項」を参照してください。
- column_expressionで参照されるユーザー定義ファンクションは、DETERMINISTICとして宣言されている必要があります。
- グローバル・パーティション・ファンクション索引では、column_expressionがパーティション・キーであってはいけません。
- column_expressionには、「列式」で説明されているすべての形式の式を指定できます。
- パラメータがない場合も、すべてのファンクションをカッコで指定する必要があります。カッコで指定していない場合は、列名として解析されます。
- column_expressionで指定するファンクションは、リピータブル値を戻す必要があります。たとえば、SYSDATEやUSERファンクションまたはROWNUM疑似列は指定できません。

関連項目:

「CREATE FUNCTION」および『Oracle Database PL/SQL言語リファレンス』を参照してください。

ASC | DESC

ASCまたはDESCを使用すると、索引を昇順で作成するか降順で作成するかを指定できます。文字データの索引は、データベース文字セットの文字値の昇順または降順で作成されます。

Oracle Databaseは、降順索引をファンクション索引として扱います。他のファンクション索引のように、最初に索引および索引が定義されている表を分析するまで、降順索引は使用しません。「column_expression」句を参照してください。

昇順の一意索引は、複数のNULL値を含むことができます。ただし、降順の一意索引の場合は、複数のNULL値は重複した値として扱われるため、許可されていません。

昇順索引と降順索引の制限事項

ドメイン索引には、これらのいずれの句も指定できません。逆索引にはDESCを指定できません。indexをビットマップ化したり、COMPATIBLE初期化パラメータに8.1.0未満の値を設定すると、DESCは無視されます。

index_attributes

オプションの索引属性を指定します。

physical_attributes_clause

physical_attributes_clauseを使用すると、索引に物理特性および記憶特性の値を設定できます。

この句を指定しない場合、PCTFREEが10、INITRANSが2に設定されます。

索引の物理属性の制限事項

索引には、PCTUSEDパラメータを指定できません。

関連項目:

これらの句の詳細は、「physical_attributes_clause」および「storage_clause」を参照してください。

TABLESPACE

tablespaceには、索引、索引パーティションまたは索引サブパーティションを格納する表領域の名前を指定します。この句を指定しない場合、その索引を定義しているスキーマの所有者のデフォルトの表領域内に索引が作成されます。

ローカル索引の場合、tablespaceのかわりにキーワードDEFAULTを指定できます。ローカル索引に追加される新規パーティションまたはサブパーティションは、基礎となる表の対応するパーティションまたはサブパーティションと同じ表領域内に作成されます。

index_compression

index_compression句を使用すると、索引の索引圧縮を有効化または無効化できます。prefix_compressionのCOMPRESS句を指定して索引の接頭辞圧縮を有効化するか、advanced_index_compressionのCOMPRESS ADVANCED句を指定して索引の拡張索引圧縮を有効化するか、prefix_compressionまたはadvanced_index_compressionのNOCOMPRESS句を指定して索引の圧縮を無効化します。デフォルトはNOCOMPRESSです。

パーティション索引の圧縮を使用する場合は、索引レベルで圧縮を有効にして索引を作成する必要があります。そのようなパーティション索引の個々のパーティションの圧縮設定は、後で有効および無効にすることができます。個々のパーティションを再作成するときに圧縮を有効および無効にできます。索引の再作成時にのみ、既存の非パーティション索引を変更して圧縮を有効化または無効化できます。

prefix_compression

COMPRESSを指定すると、キー列値の重複を排除する接頭辞圧縮(キー圧縮とも呼ばれます)を有効化できます。integerを使用して、接頭辞の長さ(圧縮する接頭辞列数)を指定します。一意でない索引または2列以上の一意索引に、接頭辞圧縮を指定できます。

- 一意の索引の場合、接頭辞の長さの有効範囲は、1からキー列の数から1を引いた数までです。デフォルトの接頭辞の長さは、(キー列の数-1)です。
- 一意でない索引の場合、接頭辞の長さの有効範囲は、1からキー列の数までです。デフォルトの接頭辞の長さはキー列数です。

advanced_index_compression

この句を指定すると拡張索引圧縮を有効化できます。拡張索引圧縮では、索引の効率的なアクセスを維持しながら圧縮率を大幅に向上します。このため、拡張索引圧縮は、接頭辞圧縮の適切な候補ではない索引を含むすべてのサポートされている索引で正常に動作します。

- COMPRESS ADVANCED LOW - このレベルでは、索引をHIGHレベル未満で圧縮しますが、索引へのアクセスは高速化されます。この句は、一意でない索引または2列以上の一意索引に指定できます。COMPRESS ADVANCED

LOWを有効化する前に、データベースの互換性レベルが12.1.0以上である必要があります。

- COMPRESS ADVANCED HIGH - このレベルでは、索引をLOWレベル超で圧縮しますが、索引へのアクセスは低速になります。この句は、一意でない索引または1列以上の一意索引に指定できます。COMPRESS ADVANCED HIGHを有効化する前に、データベースの互換性レベルが12.2.0以上である必要があります。

LOWおよびHIGHキーワードを省略した場合、HIGHがデフォルトになります。

索引圧縮の制限事項

索引圧縮には、次の制限事項が適用されます。

- ビットマップ索引に接頭辞圧縮または拡張索引圧縮は指定できません。
- 索引構成表に拡張索引圧縮を指定できません。

関連項目:

- 接頭辞圧縮および拡張索引圧縮の詳細は、『Oracle Database管理者ガイド』を参照してください。
- 「索引の圧縮: 例」を参照してください。

partial_index_clause

この句は、パーティション表に索引を作成する場合にのみ指定できます。INDEXING FULLを指定すると、全索引を作成できます。INDEXING PARTIALを指定すると、部分索引を作成できます。デフォルトは、INDEXING FULLです。

全索引には、索引付けのプロパティとは関係なく、基になる表のすべてのパーティションが含まれます。部分索引には、基になる表で索引付けのプロパティがONのパーティションのみが含まれます。

部分索引がローカル・パーティション索引の場合、索引付けプロパティがONの表パーティションに対応する索引パーティションには、USABLEのマークが付けられます。索引付けプロパティがOFFの表パーティションに対応する索引パーティションには、UNUSABLEのマークが付きます。

基になる表がコンポジット・パーティション表の場合は、索引パーティションと表パーティションについての前述の条件が、索引サブパーティションと表サブパーティションに適用されます。

部分索引の制限事項

部分索引には、次の制限事項があります。

- 部分索引の基になる表は、非パーティション表にすることはできません。
- 一意索引は、部分索引することができません。これは、CREATE UNIQUE INDEX文で作成された索引と、1つ以上の列に一意制約を指定して暗黙的に作成された索引に適用されます。

関連項目:

索引付けプロパティの詳細は、CREATE TABLEの「indexing_clause」を参照してください。

SORT | NOSORT

デフォルトでは、索引は作成時に昇順でソートされます。NOSORTを指定すると、索引の作成時に、データベース内ですでに昇順で格納されている行のソートを行わないようにできます。索引列の行または列が昇順に格納されていない場合、データベースはエラーを戻します。ソート時間および領域を削減するため、列を表へ初期ロードした直後にこの句を使用します。これらのキー

ワードのいずれも指定しない場合、デフォルトでSORTが使用されます。

NOSORTの制限事項

このパラメータには、次の制限事項があります。

- この句は、REVERSEと同時には指定できません。
- この句を使用して、クラスタ索引、パーティション索引またはビットマップ索引を作成することはできません。
- 索引構成表の2次索引には、この句を指定できません。

REVERSE

REVERSEを指定すると、ROWID以外の索引ブロックのバイトを逆順に格納できます。

逆索引の制限事項

逆索引には、次の制限事項があります。

- この句は、NOSORTと同時には指定できません。
- ビットマップ索引または索引構成表の索引は逆順にできません。

VISIBLE | INVISIBLE

この句を使用すると、オプティマイザで索引を参照可能にするかどうかを指定できます。参照不可の索引はDML操作によってメンテナンスされますが、パラメータOPTIMIZER_USE_INVISIBLE_INDEXESをセッションまたはシステム・レベルで明示的にTRUEに設定しないと、問合せ時にオプティマイザによって使用されません。

既存の索引がオプティマイザによって参照可能か参照不可かを確認するには、USER_、DBA_、ALL_INDEXESデータ・ディクショナリ・ビューのVISIBILITY列を問い合せます。

関連項目:

この機能の詳細は、『Oracle Database管理者ガイド』を参照してください。

logging_clause

索引作成を、REDOログ・ファイル内に記録する(LOGGING)か記録しない(NOLOGGING)かを指定します。この設定によって、索引に対する後続のダイレクト・ローダー(SQL*Loader)およびダイレクト・パス・インサート操作を記録するか記録しないかも決定されます。デフォルトはLOGGINGです。

indexが非パーティション索引の場合、この句は索引のロギング属性を指定します。

indexがパーティション索引の場合、この句は次の値を決定します。

- CREATE文で指定されたすべてのパーティションのデフォルト値(PARTITION記述句でlogging_clauseを指定している場合を除く)
- 索引パーティションに関連付けられたセグメントに対するデフォルト値
- 後続のALTER TABLE ... ADD PARTITION操作中に暗黙的に追加されたローカル索引パーティションまたはサブパーティションに対するデフォルト値

索引のロギング属性は、その実表の属性に依存しません。

この句を指定しない場合、ロギング属性は表が存在する表領域の属性になります。

関連項目:

- この句の詳細は、「logging_clause」を参照してください。
- ロギングおよびパラレルDMLの詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。
- 「NOLOGGINGモードでの索引の作成: 例」を参照してください。

ONLINE

ONLINEを指定すると、索引作成中の表でのDML操作を許可できます。

オンライン索引の作成の制限事項

オンライン索引の作成には、次の制限事項があります。

- オンライン索引の作成中は、パラレルDMLはサポートされません。ONLINEを指定し、パラレルDML文を発行すると、Oracle Databaseはエラーを戻します。
- COMPATIBLEを10以上に設定すると、ビットマップ索引またはクラスタ索引にONLINEを指定できます。
- UROWID列の従来索引には、ONLINEを指定できません。
- 索引構成表の一意でない2次索引の場合、索引構成表内の索引キー列の数と論理ROWIDの主キー列の数の合計は、32以下にする必要があります。論理ROWIDは、索引キーに含まれる列を除外します。

関連項目:

オンライン索引の作成および再作成については、『Oracle Database概要』を参照してください。

parallel_clause

parallel_clauseを指定すると、索引の作成をパラレル化できます。

この句の詳細は、「CREATE TABLE」の「parallel_clause」を参照してください。

Index Partitioning句

global_partitioned_index句およびlocal_partitioned_index句を使用すると、indexをパーティション化できます。

ブロック・サイズが異なる表領域のパーティション化されたデータベース・エンティティの記憶域には、制限事項があります。これらの制限事項については、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

関連項目:

「パーティション索引の例」

global_partitioned_index

global_partitioned_indexを使用すると、索引のパーティション化がユーザー定義であり、基礎となる表と同一レベルでパーティション化されないことを指定できます。デフォルトでは、非パーティション索引はグローバル索引です。

グローバル索引には、レンジ・パーティション化またはハッシュ・パーティション化を実行できます。どちらの場合でも、パーティション・

キー列に最大32列を指定できます。列リストをパーティション化する場合、索引の列リストの左の接頭辞を指定する必要があります。索引が列a、bおよびcに定義されている場合は、列に(a, b, c)、(a, b)または(a, c)は指定できますが、(b, c)、(c)または(b, a)は指定できません。パーティション名を指定する場合は、スキーマ・オブジェクトのネーミング規則および「データベース・オブジェクトのネーミング規則」にある該当部分の記述に従って指定する必要があります。パーティション名を省略すると、SYS_Pnの形式で名前が生成されます。

GLOBAL PARTITION BY RANGE

この句を使用すると、レンジ・パーティション・グローバル索引を作成できます。列リストに指定した表の列の値の範囲に基づいて、グローバル索引がパーティション化されます。

関連項目:

レンジ・パーティション・グローバル索引の作成: 例

GLOBAL PARTITION BY HASH

この句を使用すると、ハッシュ・パーティション・グローバル索引を作成できます。パーティション・キー列の値にハッシュ・ファンクションを使用して、行がパーティションに割り当てられます。

関連項目:

ハッシュ・パーティションの2つの方法の詳細は、「CREATE TABLE」句の「hash_partitions」および「ハッシュ・パーティション・グローバル索引の作成: 例」を参照してください。

グローバル・パーティション索引の制限事項

グローバル・パーティション索引には、次の制限事項があります。

- パーティション・キー列リストには、ROWID類似列またはROWID型の列は指定できません。
- ハッシュ・パーティションに指定できるプロパティは、表領域の記憶域のみです。そのため、individual_hash_partitionsのpartitioning_storage_clauseにLOBまたはVARRAYの記憶域句を指定できません。
- hash_partitions_by_quantityのOVERFLOW句は、索引構成表のパーティションに対してのみ指定できます。
- partitioning_storage_clauseでは、table_compressionまたはinmemory_clauseは指定できませんが、index_compressionは指定できます。

ノート:

異なる文字セットを使用してデータベースを使用しているか、使用する予定がある場合は、キャラクタ列を分割する際に注意してください。文字のソート順序は、すべての文字セットで同一ではありません。

関連項目:

文字セット・サポートの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

index_partitioning_clause

この句を使用すると、個々の索引パーティションを記述できます。この句が繰り返される数によってパーティションの数が決まります。partitionを指定しない場合、名前はSYS_Pnの形式で生成されます。

VALUES LESS THAN(value_list)には、グローバル索引の現在のパーティションの境界は含まない上限を指定します。値のリストは、global_partitioned_index句の列リストに対応するリテラル値を含む、カンマで区切られた順序リストです。最後のパーティションの値としては、必ずMAXVALUEを指定します。

ノート:

索引が DATE 列でパーティション化されている場合、および日付書式で年の最初の 2 桁の数字が指定されていない場合、年の 4 文字書式マスクで TO_DATE ファンクションを使用する必要があります。日付書式は、NLS_TERRITORY によって暗黙的に決定され、NLS_DATE_FORMAT によって明示的に決定されます。これらの初期化パラメータの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

関連項目:

「レンジ・パーティション化の例」

local_partitioned_index

local_partitioned_index句を使用すると、tableと同じパーティション数および同じパーティション境界を使用し、同じ列で索引をパーティション化できます。コンポジット・パーティション表では、この句を使用すると、tableと同じパーティション数および同じパーティション境界を使用し、同じ列で索引をパーティション化できます。基礎となる表が再パーティション化された場合、ローカル索引のパーティションは自動的に保持されます。

キーワードLOCALのみを指定して、副次句を指定しない場合、Oracle Databaseは対応する表パーティションとして、同じ表領域に各索引パーティションを作成し、対応する表パーティションと同じ名前を割り当てます。表がコンポジット・パーティション表である場合、Oracle Databaseは対応する表サブパーティションとして、同じ表領域に各索引パーティションを作成して、対応する表サブパーティションと同じ名前を割り当てます。

パーティション名を指定する場合は、スキーマ・オブジェクトのネーミング規則および「データベース・オブジェクトのネーミング規則」にある該当部分の記述に従って指定する必要があります。パーティション名を指定しない場合、対応する表のパーティションと一貫した名前が生成されます。その名前が既存の索引パーティション名と競合する場合は、SYS_Pnの形式が使用されます。

on_range_partitioned_table

この句を使用すると、レンジ・パーティション表の索引パーティションに名前および属性を指定できます。この句を指定する場合、PARTITION句は、表パーティションと同一の数と順序である必要があります。

索引に接頭辞圧縮が指定されていない場合、索引パーティションに接頭辞圧縮を指定することはできません。

USABLE句とUNUSABLE句の詳細は、「USABLE | UNUSABLE」を参照してください。

on_list_partitioned_table

on_list_partitioned_table句は、on_range_partitioned_tableと同一です。

on_hash_partitioned_table

この句を使用すると、ハッシュ・パーティション表の索引パーティションに名前および表領域の記憶域を指定できます。

PARTITION句を指定する場合、これらの句の数は表パーティションの数と同一である必要があります。オプションで、1つ以上の個々のパーティションに表領域の記憶域を指定できます。この句またはSTORE IN句に表領域の記憶域を指定しない場合、各索引パーティションが、対応する表パーティションと同じ表領域に格納されます。

STORE IN句を使用して、すべての索引ハッシュ・パーティションを分散させる1つ以上の表領域を指定できます。表領域の数は、索引パーティションの数と等しくなる必要はありません。索引パーティションの数が表領域の数より多い場合は、表領域名が繰り返し使用されます。

USABLE句とUNUSABLE句の詳細は、「USABLE | UNUSABLE」を参照してください。

on_comp_partitioned_table

この句を使用すると、コンポジット・パーティション表の索引パーティションに名前および属性を指定できます。

STORE IN句は、レンジ-ハッシュまたはリスト-ハッシュ・コンポジット・パーティション表に対してのみ有効です。すべてのパーティションのすべての索引ハッシュ・サブパーティションを分散させる1つ以上のデフォルト表領域を指定できます。index_subpartition_clauseの第2 STORE IN句で個々のパーティションのサブパーティションに対して異なるデフォルト表領域の記憶域を指定すると、この記憶域を上書きできます。

レンジ-レンジ、レンジ-リストおよびリスト-リスト・コンポジット・パーティション表に対しては、PARTITION句に指定したレンジまたはリスト・サブパーティションのデフォルトの属性を指定できます。index_subpartition_clauseのSUBPARTITION句の個々のパーティションのレンジまたはリスト・サブパーティションに異なる属性を指定すると、この記憶域を上書きできます。

索引に接頭辞圧縮が指定されていない場合、索引パーティションに接頭辞圧縮を指定することはできません。

USABLE句とUNUSABLE句の詳細は、「USABLE | UNUSABLE」を参照してください。

index_subpartition_clause

この句を使用すると、コンポジット・パーティション表の索引サブパーティションに名前および表領域の記憶域を指定できます。

STORE IN句は、レンジ-ハッシュおよびリスト-ハッシュ・コンポジット・パーティション表のハッシュ・サブパーティションに対してのみ有効です。すべての索引ハッシュ・サブパーティションを分散させる1つ以上の表領域を指定できます。SUBPARTITION句は、すべてのサブパーティション・タイプに対して有効です。

SUBPARTITION句を指定する場合、これらの句の数は表サブパーティションの数と同一である必要があります。サブパーティション名を指定する場合は、スキーマ・オブジェクトのネーミング規則および「データベース・オブジェクトのネーミング規則」にある該当部分の記述に従って指定する必要があります。subpartitionを指定しない場合、対応する表のサブパーティションと一貫した名前が生成されます。その名前が既存の索引サブパーティション名と競合する場合は、SYS_SUBPnの形式が使用されます。

表領域の数は、索引サブパーティションの数と等しくなる必要はありません。索引サブパーティションの数が表領域の数より多い場合は、表領域名が繰り返し使用されます。

on_comp_partitioned_table句またはindex_subpartition_clauseにサブパーティションの表領域の記憶域を指定しない場合、indexに指定された表領域が使用されます。indexに表領域の記憶域を指定しない場合、サブパーティションが、対応する表サブパーティションと同じ表領域に格納されます。

USABLE句とUNUSABLE句の詳細は、CREATE INDEX ... USABLE | UNUSABLEを参照してください。

domain_index_clause

domain_index_clauseを使用して、indexが、アプリケーション固有のindextype索引タイプのインスタンスであるドメイン索引であることを示します。

ドメイン索引を作成する前に、複数の操作を行う必要があります。まず、索引タイプの実装タイプを作成します。また、ファンクション実装を作成し、そのファンクションを使用する演算子も作成します。次に、演算子と実装タイプを関連付ける索引タイプを作成します。最後にこの句を使用してドメイン索引を作成します。「詳細な例」に、これらのすべての操作を含んだ単純なドメイン索引の作成例を記載しています。

index_expr

index_expr (table_index_clause内)に、索引が定義されている表の列またはオブジェクトの属性を指定します。基礎となる索引タイプが異なり、その索引タイプがユーザー定義操作の分割セットをサポートする場合のみ、1つの列に複数のドメイン索引を定義できます。

ドメイン索引の制限事項

ドメイン索引には、次の制限事項があります。

- index_expr (table_index_clause内)には1つの列のみ指定でき、データ型がREF、VARRAY、ネストした表、LONGまたはLONG RAWの列は指定できません。
- ビットマップ索引または一意ドメイン索引は作成できません。
- 一時表には、ドメイン索引は作成できません。
- ローカル・ドメイン索引は、レンジ・パーティション表、リスト・パーティション表、ハッシュ・パーティション表および時間間隔パーティション表にのみ作成できますが、例外として、ローカル・ドメイン索引を自動リスト・パーティション表に作成することはできません。
- ドメイン索引は、照合BINARY、USING_NLS_COMP、USING_NLS_SORTまたはUSING_NLS_SORT_CSを使用して宣言された表の列に対してのみ作成できます。詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

indextype

indextypeには、索引タイプの名前を指定します。名前は、作成済の有効なスキーマ・オブジェクトである必要があります。

Oracle Textをインストールしている場合、様々な組込み索引タイプを使用して、Oracle Textドメイン索引を作成できます。Oracle TextおよびOracle Textが使用する索引の詳細は、『Oracle Textリファレンス』を参照してください。

関連項目:

CREATE INDEXTYPE

local_domain_index_clause

この句を使用すると、索引がパーティション表のローカル索引であることを指定できます。

- PARTITIONS句を使用すると、索引パーティションの名前を指定できます。指定するパーティション数は、実表内のパーティション数と一致する必要があります。この句を省略すると、SYS_Pnという形式のシステム生成の名前でパーティションが作成されます。
- PARAMETERS句を使用すると、個々のパーティション固有のパラメータ文字列を指定できます。この句を省略した場合、索引に関連付けられたパラメータ文字列はパーティションにも関連付けられます。

parallel_clause

parallel_clauseを使用すると、ドメイン索引の作成をパラレル化できます。非パーティション・ドメイン索引の場合、

Oracle Databaseは明示的またはデフォルトの並列度をODCIIndexCreateカートリッジ・ルーチンに渡し、ODCIIndexCreateカートリッジ・ルーチンが索引の並列性を確立します。ローカル・ドメイン索引の場合、この句によって索引パーティションはパラレルに作成されます。

関連項目:

Oracle Data Cartridge Interface(ODCI)ルーチンの詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。

PARAMETERS

PARAMETERS句で、解析されずに適切なODCI索引タイプ・ルーチンに渡されるパラメータ文字列を指定します。パラメータ文字列の最大長は1,000文字です。

構文の最上位でこの句を指定した場合、パラメータは索引パーティションのデフォルトのパラメータになります。local_domain_index_clauseの一部としてこの句を指定すると、個々のパーティションのパラメータでデフォルトのパラメータを上書きできます。

ドメイン索引が作成されると、適切なODCIルーチンがコールされます。ルーチンが正常に戻らない場合、ドメイン索引はFAILEDのマークが付けられます。失敗したドメイン索引でサポートされる操作は、DROP INDEXおよびREBUILD INDEX(非ローカル索引用)のみです。

関連項目:

Oracle Data Cartridge Interface(ODCI)ルーチンの詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。

XMLIndex_clause

XMLIndex_clauseを使用すると、一般にはXMLデータが含まれる列に、XMLIndex索引を定義できます。XMLIndex索引は、特にXMLデータのドメイン用に設計されたドメイン索引の一種です。

XMLIndex_parameters_clause

この句を使用すると、パス表に関する情報およびXMLIndexのコンポーネントに対応する2次索引に関する情報を指定できます。また、この句を使用すると、索引の構造化コンポーネントの詳細を指定できます。パラメータ文字列の最大長は1,000文字です。

構文の最上位でこの句を指定した場合、パラメータは索引のパラメータおよび索引パーティションのデフォルトのパラメータになります。local_xmlindex_clause句の一部としてこの句を指定すると、個々のパーティションのパラメータでデフォルトのパラメータを上書きできます。

関連項目:

XMLIndex_parameters_clauseの構文およびセマンティクスの詳細、およびXMLIndexの使用の詳細は、『Oracle XML DB開発者ガイド』を参照してください

bitmap_join_index_clause

bitmap_join_index_clauseを使用すると、ビットマップ結合索引を定義できます。ビットマップ結合索引は、単一の表

に定義します。ディメンション表の列で構成される索引キーには、そのキーに対応するファクト表のROWIDが格納されます。データ・ウェアハウス環境では、一般的に、索引を定義する表をファクト表といい、ファクト表と結合した表をディメンション表といいます。ただし、結合索引の作成にはスター・スキーマは必須ではありません。

ON

ON句には、まずファクト表を指定し、次に索引を定義するディメンション表の列をカッコ内に指定します。

FROM

FROM句には、結合した表を指定します。

WHERE

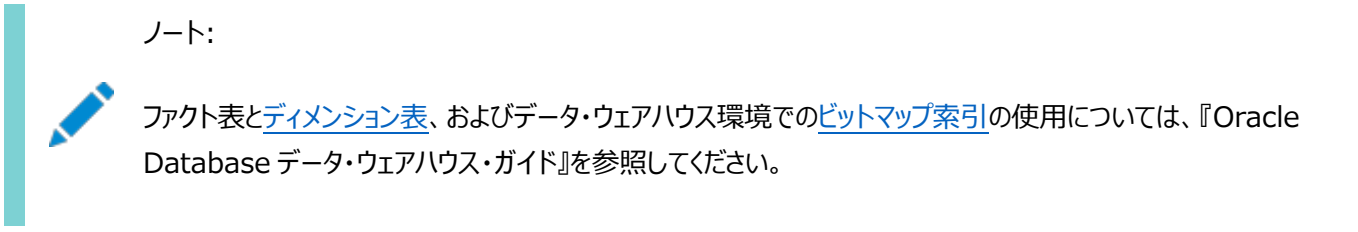
WHERE句には、結合条件を指定します。

基礎となるファクト表がパーティション化されている場合、local_partitioned_index句(「local_partitioned_index」を参照)のいずれかを指定する必要があります。

ビットマップ結合索引の制限事項

一般的なビットマップ索引の制限事項(「BITMAP」を参照)に加え、ビットマップ結合索引には次の制限事項が適用されます。

- 一時表にはビットマップ結合索引は作成できません。
- FROM句で表を2回指定できません。
- ファンクション結合索引は作成できません。
- ディメンション表の列は、主キー列であるか、または一意制約を含む必要があります。
- ディメンション表が複合主キーを含む場合、主キーの各列は結合の一部である必要があります。
- ファクト表がパーティション化されていない場合は、local_partitioned_index句を指定できません。
- ビットマップ結合索引定義では、照合BINARY、USING_NLS_COMP、USING_NLS_SORTまたはUSING_NLS_SORT_CSを含む列のみを参照できます。これらのいずれかの照合について索引キーが照合され、結合条件がBINARY照合を使用して評価されます。詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。



ノート:

ファクト表とディメンション表、およびデータ・ウェアハウス環境でのビットマップ索引の使用については、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

USABLE | UNUSABLE

USABLEキーワードとUNUSABLEキーワードは、次の項目に指定できます。

- CREATE INDEX文内の索引
- on_range_partitioned_table句、on_list_partitioned_table句、on_hash_partitioned_table句、およびon_comp_partitioned_table句内の索引パーティション

- index_subpartition_clause内の索引サブパーティション

非パーティション索引の場合、UNUSABLEを指定すると、使用禁止状態で索引を作成できます。使用禁止の索引を使用可能にする場合、再構築するか、または削除して再作成する必要があります。USABLEを指定すると、使用可能状態で索引を作成できます。USABLEは、デフォルトです。

パーティション索引の場合は、次のようにUSABLEまたはUNUSABLEを指定します。

- この索引にUNUSABLEを指定すると、すべての索引パーティションにUNUSABLEのマークが付きます。
- この索引にUSABLEを指定すると、すべての索引パーティションにUSABLEのマークが付きます。
- この索引にUSABLEまたはUNUSABLEを指定していないと、すべての索引パーティションにUSABLEのマークが付きます。ただし、ローカルの部分索引を除きます。LOCAL句とINDEXING PARTIAL句を指定するときに、USABLEまたはUNUSABLEを省略すると、各索引パーティションには、それに対応する表パーティションの索引付けプロパティがONの場合にUSABLEのマークが付けられ、対応する表パーティションの索引付けプロパティがOFFの場合にUNUSABLEのマークが付けられます。

前述の条件は、特定の索引パーティションにUSABLEまたはUNUSABLEを指定することで上書きできます。

基になる表がコンポジット・パーティション表の場合は、索引パーティションと表パーティションについての前述の条件が、索引サブパーティションと表サブパーティションに適用されます。

パーティション索引の作成後には、特定の索引パーティションまたはサブパーティションをUSABLEにするように再構築できます。これは、一部の索引パーティションまたはサブパーティションのみで索引を維持する場合に有効です。たとえば、新しいパーティションでは索引アクセスを有効にするものの、古いパーティションでは有効にしない場合などに有効です。

索引または索引の一部のパーティションまたはサブパーティションにUNUSABLEのマークが付けられている場合は、その使用禁止のオブジェクトにはセグメントは割り当てられません。使用禁止の索引または索引パーティションは、データベース内で領域を使用しません。

索引、または索引の一部のパーティションやサブパーティションにUNUSABLEのマークが付けられている場合は、2つの条件を満たすときにのみ、索引はオプティマイザによってアクセス・パスとみなされます。1つは、オプティマイザがアクセス対象のパーティションをコンパイル時に認識していること、もう1つは、アクセス対象のすべてのパーティションにUSABLEのマークが付けられていることです。そのため、問合せにバインド変数を含めることはできません。

USABLE | UNUSABLEの制限事項

索引にUSABLEまたはUNUSABLEのマークを付ける場合は、次の制限事項が適用されます。

- この句は、一時表の索引に対して指定できません。
- 次の状況では、使用禁止の索引または索引パーティションにもセグメントが割り当てられます。
    - 索引(または索引パーティション)がSYS、SYSTEM、PUBLIC、OUTLNまたはXDBによって所有されている場合
    - 索引(または索引パーティション)がディクショナリ管理表領域に格納されている場合
    - パーティションのメンテナンス操作が行われているため、パーティション表のグローバル・パーティション索引または非パーティション索引が使用禁止になっている場合

{ DEFERRED | IMMEDIATE } INVALIDATION

この句を使用すると、索引の作成中に依存カーソルがいつ無効化されるかを制御できます。この場合のセマンティクスは、ALTER INDEX文のものと同じです。この句のセマンティクスの詳細は、ALTER INDEXのドキュメントの「{ DEFERRED |

IMMEDIATE } INVALIDATION」を参照してください。

例

一般的な索引の例

索引の作成: 例

次の文は、サンプル表oe.ordersのcustomer_id列にサンプル索引ord_customer_ixを作成します。

```
CREATE INDEX ord_customer_ix
   ON orders (customer_id);
```

索引の圧縮: 例

COMPRESS句を使用してord_customer_ix_demo索引を作成する場合は、次の文を発行できます。

```
CREATE INDEX ord_customer_ix_demo
   ON orders (customer_id, sales_rep_id)
   COMPRESS 1;
```

索引は、customer_id列値の繰返し項目を圧縮します。

NOLOGGINGモードでの索引の作成: 例

サンプル表ordersが高速パラレル・ロードで作成された場合(すべての行がソート済である場合)、次の文を発行して、迅速に索引を作成できます。

```
/* Unless you first sort the table oe.orders, this example fails
   because you cannot specify NOSORT unless the base table is
   already sorted.
*/
CREATE INDEX ord_customer_ix_demo
   ON orders (order_mode)
   NOSORT
   NOLOGGING;
```

クラスタ索引の作成: 例

personnelクラスタ(「クラスタの作成: 例」で作成)に索引を作成するには、次の文を発行します。

```
CREATE INDEX idx_personnel ON CLUSTER personnel;
```

クラスタ・キーのすべての列にクラスタ索引が自動的に作成されるため、索引列は指定しません。クラスタ索引の場合は、すべての行に索引が付きます。

「XMLType表の索引の作成: 例」

次の例では、xwarehouses表(「XMLType表の例」で作成)の領域要素に索引を作成します。

```
CREATE INDEX area_index ON xwarehouses e
   (EXTRACTVALUE(VALUE(e),'/Warehouse/Area'));
```

この索引によって、たとえば、次の文にあるような倉庫の面積(平方フィート)に基づいた表から選択する問合せのパフォーマンスが大幅に向上します。

```
SELECT e.getClobVal() AS warehouse
   FROM xwarehouses e
   WHERE EXISTSNODE(VALUE(e),'/Warehouse[Area>50000]') = 1;
```

関連項目:

EXISTSNODE and VALUE

ファンクション索引の例

次の例では、ファンクション索引を作成および使用する方法を示します。

ファンクション索引の作成: 例

次の文は、last_name列の大文字評価に基づいてemployees表にファンクション索引を作成します。

```
CREATE INDEX upper_ix ON employees (UPPER(last_name));
```

ファンクション索引の作成に必要な権限およびパラメータ設定は、「前提条件」を参照してください。

全表スキャンを実行するのではなく、索引が使用される可能性を高めるには、ファンクションが戻す値を後続の問合せでNULL以外にします。たとえば、次の文では、オプティマイザの動作を阻止する条件が他に存在しないかぎり、索引が使用されます。

```
SELECT first_name, last_name
   FROM employees WHERE UPPER(last_name) IS NOT NULL
   ORDER BY UPPER(last_name);
```

WHERE句を指定しないと、全表スキャンが実行される場合があります。

索引の作成および後続の問合せを示す次の文では、問合せで列の順序が逆であっても、Oracle Databaseはincome_ixを使用します。

```
CREATE INDEX income_ix
   ON employees(salary + (salary*commission_pct));
SELECT first_name||' '||last_name "Name"
   FROM employees
   WHERE (salary*commission_pct) + salary > 15000
   ORDER BY employee_id;
```

LOB列のファンクション索引の作成: 例

次の文は、text_lengthファンクションを使用して、サンプルのpmスキーマ内のLOB列にファンクション索引を作成します。このファンクションを作成する例については、『Oracle Database PL/SQL言語リファレンス』を参照してください。例では、CLOB列が1000字未満のサンプル表print_mediaから行を検索します。

```
CREATE INDEX src_idx ON print_media(text_length(ad_sourcetext));
SELECT product_id FROM print_media
   WHERE text_length(ad_sourcetext) < 1000
   ORDER BY product_id;
PRODUCT_ID
----------
      2056
      2268
      3060
      3106
```

型メソッドのファンクション索引の作成: 例

この例では、2つの数値属性lengthおよびwidthを含むオブジェクト型rectangleが必要です。area()メソッドは、四角形の面積を計算します。

```
CREATE TYPE rectangle AS OBJECT
( length    NUMBER,
  width     NUMBER,
  MEMBER FUNCTION area RETURN NUMBER DETERMINISTIC
);
```

```
CREATE OR REPLACE TYPE BODY rectangle AS
  MEMBER FUNCTION area RETURN NUMBER IS
  BEGIN
   RETURN (length*width);
  END;
END;
```

rectangle型の表rect_tabを作成する場合、次のようにarea()メソッドにファンクション索引を作成できます。

```
CREATE TABLE rect_tab OF rectangle;
CREATE INDEX area_idx ON rect_tab x (x.area());
```

この索引を効率的に使用して、次の形式の問合せを評価できます。

```
SELECT * FROM rect_tab x WHERE x.area() > 100;
```

条件付き一意性を定義するためのファンクション索引の使用方法: 例

次の文は、一意のファンクション索引をoe.orders表に作成して、顧客がプロモーションID 2 (blowout sale)を複数回利用できないようにします。

```
CREATE UNIQUE INDEX promo_ix ON orders
   (CASE WHEN promotion_id =2 THEN customer_id ELSE NULL END,
    CASE WHEN promotion_id = 2 THEN promotion_id ELSE NULL END);
INSERT INTO orders (order_id, order_date, customer_id, order_total, promotion_id)
   VALUES (2459, systimestamp, 106, 251, 2);
1 row created.
INSERT INTO orders (order_id, order_date, customer_id, order_total, promotion_id)
   VALUES (2460, systimestamp+1, 106, 110, 2);
insert into orders (order_id, order_date, customer_id, order_total, promotion_id)
*
ERROR at line 1:
ORA-00001: unique constraint (OE.PROMO_IX) violated
```

promotion_idが2以外の行を索引から削除するようにします。Oracle Databaseでは、すべてのキーがNULLの行は索引内に格納されません。したがって、この例では、promotion_idが2である場合以外は、customer_idとpromotion_idの両方がNULLにマップされます。その結果、customer_id値が同じ2つの行についてpromotion_idが2の場合にのみ、索引の制約に違反することになります。

パーティション索引の例

レンジ・パーティション・グローバル索引の作成: 例

次の文は、コストの範囲を3つのグループに分割した3つのパーティションを含むサンプル表sh.salesにグローバル同一キー索引cost_ixを作成します。

```
CREATE INDEX cost_ix ON sales (amount_sold)
   GLOBAL PARTITION BY RANGE (amount_sold)
      (PARTITION p1 VALUES LESS THAN (1000),
       PARTITION p2 VALUES LESS THAN (2500),
       PARTITION p3 VALUES LESS THAN (MAXVALUE));
```

ハッシュ・パーティション・グローバル索引の作成: 例

次の文は、4つのパーティションを持つサンプル表sh.customersに、ハッシュ・パーティション・グローバル索引cust_last_name_ixを作成します。

```
CREATE INDEX cust_last_name_ix ON customers (cust_last_name)
  GLOBAL PARTITION BY HASH (cust_last_name)
  PARTITIONS 4;
```

ハッシュ・パーティション表の索引の作成: 例

次の文は、hash_productsパーティション表(「ハッシュ・パーティション化の例」で作成)のcategory_id列にローカル索引を作成します。LOCALの直後に記述されたSTORE IN句は、hash_productsがハッシュ・パーティション化されていることを示します。Oracle Databaseは、tbs1表領域とtbs2表領域にハッシュ・パーティションを分散させます。

```
CREATE INDEX prod_idx ON hash_products(category_id) LOCAL
   STORE IN (tbs_01, tbs_02);
```

索引を作成するには、指定する表領域に対して割当て制限を持つ必要があります。表領域tbs_01およびtbs_02を作成する例は、「CREATE TABLESPACE」を参照してください。

コンポジット・パーティション表の索引の作成: 例

次の文は、composite_sales表(「コンポジット・パーティション表の例」で作成)にローカル索引を作成します。STORAGE句では、索引のデフォルトの記憶域属性を指定します。ただし、別のTABLESPACE記憶域が指定されているため、このデフォルトは、パーティションq3_2000およびq4_2000の5つのサブパーティションに上書きされます。

索引を作成するには、指定する表領域に対して割当て制限を持つ必要があります。表領域tbs_02およびtbs_03を作成する例は、「CREATE TABLESPACE」を参照してください。

```
CREATE INDEX sales_ix ON composite_sales(time_id, prod_id)
   STORAGE (INITIAL 1M)
   LOCAL
   (PARTITION q1_1998,
    PARTITION q2_1998,
    PARTITION q3_1998,
    PARTITION q4_1998,
    PARTITION q1_1999,
    PARTITION q2_1999,
    PARTITION q3_1999,
    PARTITION q4_1999,
    PARTITION q1_2000,
    PARTITION q2_2000
      (SUBPARTITION pq2001, SUBPARTITION pq2002,
       SUBPARTITION pq2003, SUBPARTITION pq2004,
       SUBPARTITION pq2005, SUBPARTITION pq2006,
       SUBPARTITION pq2007, SUBPARTITION pq2008),
    PARTITION q3_2000
      (SUBPARTITION c1 TABLESPACE tbs_02,
       SUBPARTITION c2 TABLESPACE tbs_02,
       SUBPARTITION c3 TABLESPACE tbs_02,
       SUBPARTITION c4 TABLESPACE tbs_02,
       SUBPARTITION c5 TABLESPACE tbs_02),
    PARTITION q4_2000
      (SUBPARTITION pq4001 TABLESPACE tbs_03,
       SUBPARTITION pq4002 TABLESPACE tbs_03,
       SUBPARTITION pq4003 TABLESPACE tbs_03,
       SUBPARTITION pq4004 TABLESPACE tbs_03)
);
```

ビットマップ索引の例

次の文は、表oe.hash_products (「ハッシュ・パーティション化の例」で作成)にビットマップ索引を作成します。

```
CREATE BITMAP INDEX product_bm_ix
   ON hash_products(list_price)
   LOCAL(PARTITION ix_p1 TABLESPACE tbs_01,
         PARTITION ix_p2,
         PARTITION ix_p3 TABLESPACE tbs_02,
         PARTITION ix_p4 TABLESPACE tbs_03)
   TABLESPACE tbs_04;
```

hash_productsはパーティション表であるため、ビットマップ結合索引はローカル・パーティションである必要があります。この例では、指定する表領域に対して割当て制限を持つ必要があります。表領域tbs_01、tbs_02、tbs_03およびtbs_04を作成する例は、「CREATE TABLESPACE」を参照してください。

次の一連の文は、結合とディメンション表を使用してファクト表上にビットマップ結合索引を作成する方法を示します。

```
CREATE TABLE hash_products
    ( product_id          NUMBER(6)
    , product_name        VARCHAR2(50)
    , product_description VARCHAR2(2000)
    , category_id         NUMBER(2)
    , weight_class        NUMBER(1)
    , warranty_period     INTERVAL YEAR TO MONTH
    , supplier_id         NUMBER(6)
    , product_status      VARCHAR2(20)
    , list_price          NUMBER(8,2)
    , min_price           NUMBER(8,2)
    , catalog_url         VARCHAR2(50)
    , CONSTRAINT          pk_product_id PRIMARY KEY (product_id)
    , CONSTRAINT          product_status_lov_demo
                          CHECK (product_status in ('orderable'
                                                    ,'planned'
                                                    ,'under development'
                                                    ,'obsolete')
 ) )
 PARTITION BY HASH (product_id)
 PARTITIONS 5
 STORE IN (example);

CREATE TABLE sales_quota
    ( product_id          NUMBER(6)
    , customer_name       VARCHAR2(50)
    , order_qty           NUMBER(6)
  ,CONSTRAINT u_product_id UNIQUE(product_id)
 );

CREATE BITMAP INDEX product_bm_ix
   ON hash_products(list_price)
   FROM hash_products h, sales_quota s
   WHERE h.product_id = s.product_id
   LOCAL(PARTITION ix_p1 TABLESPACE example,
         PARTITION ix_p2,
         PARTITION ix_p3 TABLESPACE example,
         PARTITION ix_p4,
         PARTITION ix_p5 TABLESPACE example)
   TABLESPACE example;
```

ネストした表の索引: 例

サンプル表pm.print_mediaには、記憶表textdocs_nestedtabに格納されたネストした表の列ad_textdocs_ntabが含まれます。次の文は、記憶表textdocs_nestedtabに一意索引を作成します。

```
CREATE UNIQUE INDEX nested_tab_ix
      ON textdocs_nestedtab(NESTED_TABLE_ID, document_typ);
```

疑似列NESTED_TABLE_IDを組み込むことによって、ネストした表の列ad_textdocs_ntab内に固有の行が確保されます。

置換可能な列の索引の作成例:

置換可能な列を宣言した型の属性に、索引を作成できます。また、適切なTREATファンクションの使用によって、サブタイプの属性を参照できます。次の例では、表books (「置換可能な表および列のサンプル」で作成)を使用します。この文は、

books表の、すべてのemployee_t型のauthorのsalary属性に索引を作成します。

```
CREATE INDEX salary_i
   ON books (TREAT(author AS employee_t).salary);
```

TREATファンクションの引数のターゲットとなる型は、参照する属性を追加した型である必要があります。例では、TREATのターゲットは、employee_tで、salary属性を追加した型です。

この条件を満たさない場合、TREATファンクションがファンクションの定義式として解析され、ファンクション索引が作成されます。たとえば、次の文は、パートタイム従業員のsalary属性にファンクション索引を作成し、型の階層に含まれる他のすべての型のインスタンスにNULLを割り当てます。

```
CREATE INDEX salary_func_i ON persons p
   (TREAT(VALUE(p) AS part_time_emp_t).salary);
```

SYS_TYPEIDファンクションの使用によって、置換可能な列を基礎とする型判別式の列に索引を作成できます。

ノート:

Oracle Database は型判別式の列を使用し、IS OF type 条件を含む問合せを評価します。型 ID 列のカーディナリティが通常低い場合、ビットマップ索引を作成することをお薦めします。

次の文は、books表のauthor列の型IDにビットマップ索引を作成します。

```
CREATE BITMAP INDEX typeid_i ON books (SYS_TYPEID(author));
```

関連項目:

- books表を基礎とする型の階層の作成については、『Oracle Database PL/SQL言語リファレンス』を参照してください。
- ファンクション「TREAT」、「SYS_TYPEID」および照合「IS OF type条件」を参照してください。

CREATE INDEXTYPE

目的

CREATE INDEXTYPE文を使用すると、(アプリケーション固有の)ドメイン索引を管理するルーチンを指定するオブジェクトである索引タイプを作成できます。索引タイプは、表、ビューおよび他のスキーマ・オブジェクトと同じネームスペースにあります。この文は、索引タイプ名を実装タイプに結合し、順番に索引タイプを実装するユーザー定義索引ファンクションおよびプロシージャを指定し、参照します。

関連項目:

索引タイプの実装の詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。

前提条件

自分のスキーマに索引タイプを作成する場合は、CREATE INDEXTYPEシステム権限が必要です。他のユーザーのスキーマ内に索引タイプを作成する場合は、CREATE ANY INDEXTYPEシステム権限が必要です。どちらの場合も、実装タイプおよびサポートしている演算子に対するEXECUTEオブジェクト権限が必要です。

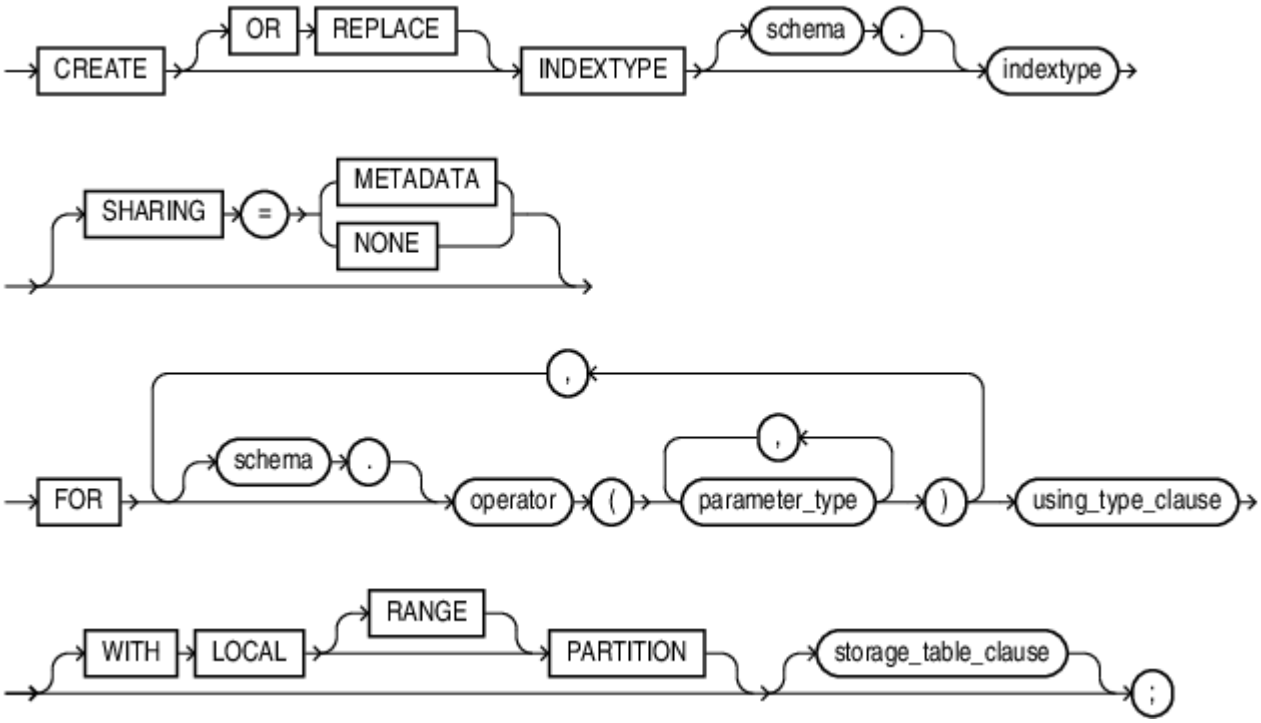
索引タイプは、1つ以上の演算子をサポートしているため、索引タイプを作成する前に、サポートする演算子を設計し、これらの演算子に機能的な実装を提供する必要があります。

関連項目:

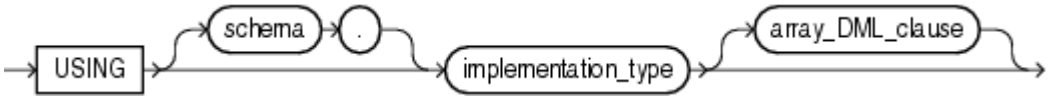
CREATE OPERATOR

構文

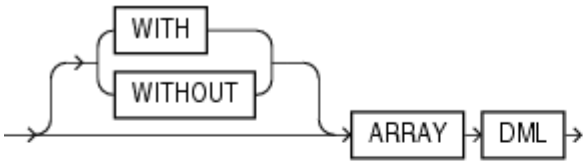
create_indextype::=

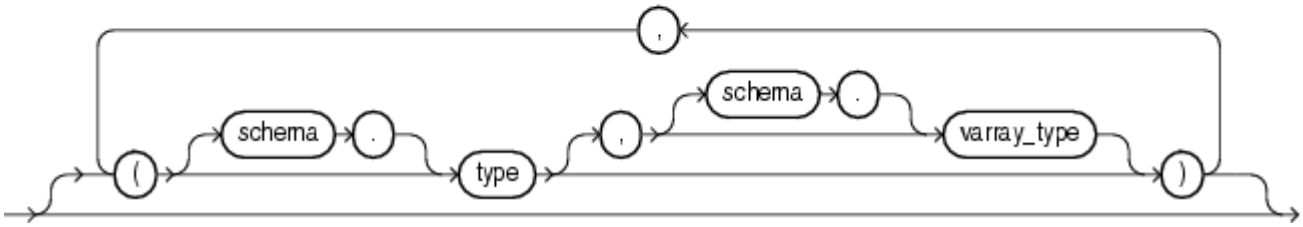


using_type_clause::=



array_DML_clause::=





storage_table_clause::=

セマンティクス

schema

索引タイプが存在するスキーマ名を指定します。schemaを指定しない場合、自分のスキーマ内に索引タイプが作成されます。

indextype

作成する索引タイプの名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

SHARING

SHARING句は、アプリケーションのメンテナンスの際に、アプリケーション・ルートにオブジェクトを作成する場合に使用します。このタイプのオブジェクトはアプリケーション共通オブジェクトと呼ばれ、アプリケーション・ルートに属するアプリケーションPDBと共有できます。

次の共有属性のいずれかを使用して、オブジェクトを共有する方法を指定できます。

- METADATA - メタデータ・リンクはメタデータを共有しますが、そのデータは各コンテナに固有です。このタイプのオブジェクトは、メタデータリンク・アプリケーション共通オブジェクトと呼ばれます。
- NONE - オブジェクトは共有されず、アプリケーション・ルートでのみアクセスできます。

FOR句

FOR句を使用すると、索引タイプでサポートされる演算子のリストを指定できます。

- schemaには、演算子を含むスキーマを指定します。schemaを指定しない場合、その演算子は自分のスキーマ内に

あるとみなされます。

- operatorには、索引タイプによってサポートされる演算子の名前を指定します。

  この句に指定するすべての演算子は有効な演算子である必要があります。

- parameter_typeには、演算子へのパラメータ・タイプを指定します。

using_type_clause

USING句を使用すると、新しい索引タイプを実装するタイプを指定できます。

implementation_typeには、適切なOracle Data Cartridge Interface(ODCI)を実装するタイプ名を指定します。

- ODCIでルーチンを実装する有効なタイプを指定する必要があります。
- 実装タイプは、索引タイプと同じスキーマに存在する必要があります。

関連項目:

このインタフェースの詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。

WITH LOCAL PARTITION

この句を使用すると、レンジ・パーティション表、リスト・パーティション表、ハッシュ・パーティション表、および時間隔パーティション表に対するローカル・ドメイン索引の作成に、索引タイプが使用できることを指定できます。この句は、複数の方法でstorage_table_clauseと組み合せて使用します(storage_table_clauseを参照)。

- WITH LOCAL PARTITION WITH SYSTEM MANAGED STORAGE TABLESを指定する方法をお薦めします。この組合せでは、システム管理記憶域表を使用しますが、これは推奨の記憶域管理であり、レンジパーティション表、リスト・パーティション表、ハッシュ・パーティション表、および時間隔パーティション表にローカル・ドメイン索引を作成できます。RANGEキーワードは、WITH LOCAL PARTITION WITH SYSTEM MANAGED STORAGE TABLESを指定すると不要となるため、この場合はオプションとなり無視されます。
- WITH LOCAL RANGE PARTITION(RANGEキーワードを含む)を指定してstorage_table句を省略できます。レンジ・パーティション表上のローカル・ドメイン索引は、ユーザー管理記憶表で、下位互換性のためにサポートされています。この組合せは効率性の低いユーザー管理記憶表を使用するため、お薦めしません。

この句を完全に省略すると、後でこの索引タイプを使用してレンジ・パーティション表、リスト・パーティション表、ハッシュ・パーティション表、時間隔パーティション表にローカル・ドメイン索引を作成することができなくなります。

storage_table_clause

この句を使用すると、この索引タイプに基づいて作成された索引の記憶表およびパーティション・メンテナンス操作の管理方法を指定できます。

- WITH SYSTEM MANAGED STORAGE TABLESを指定すると、統計データの格納がシステムで管理されます。statistics_typeに指定するタイプによって、システムで保持される表に統計関連の情報が格納されます。また、指定する索引タイプはすでに登録済であるか、またはWITH SYSTEM MANAGED STORAGE TABLES句をサポートするように変更されている必要があります。
- WITH USER MANAGED STORAGE TABLESを指定すると、ユーザー定義の統計情報を格納する表は、ユーザーによって管理されます。これはデフォルトの動作です。

関連項目:

ドメイン索引の記憶表の詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。

array_DML_clause

この句を使用すると、索引タイプで`ODCIIndexInsert`メソッドの配列インタフェースをサポートできるようになります。

type および`varray_type`

索引付けする列のデータ型がユーザー定義オブジェクト型である場合は、Oracleが`type`の列値の保持に使用するVARRAY `varray_type`を識別するために、この句を指定する必要があります。索引タイプで型のリストがサポートされている場合、対応するVARRAY型のリストを指定できます。`type`または`varray_type`で`schema`を省略した場合、型が自分のスキーマ内に定義されているとみなされます。

索引付けする列のデータ型が組込みシステム型である場合、その索引タイプに指定されたVARRAY型は、システムで定義されたODCI型よりも優先されます。

関連項目:

ODCI配列インタフェースの詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。

例

索引タイプの作成: 例

次の文は、`position_indextype`という名前の索引タイプを作成し、その索引タイプでサポートされている`position_between`演算子、および索引インタフェースを実装する`position_im`タイプを指定します。この索引タイプを使用する拡張索引作成機能の使用例は、「拡張索引作成機能の使用方法」を参照してください。

```
CREATE INDEXTYPE position_indextype
   FOR position_between(NUMBER, NUMBER, NUMBER)
   USING position_im;
```

CREATE INMEMORY JOIN GROUP

目的

CREATE INMEMORY JOIN GROUP文を使用すると、同じ表または異なる表から頻繁に結合される列を指定するオブジェクトである結合グループを作成できます。通常、このような列には、類似する範囲内の互換性のあるデータ型の値が含まれています。結合グループを作成する際、列固有のメタデータがグローバル・ディクショナリに格納されるため、列の結合問合せが最適化されます。最適化を実現するためには、表の列をインメモリー列ストア(IM列ストア)に移入する必要があります。

表に結合グループを作成すると、現在メモリー内にあるこれらの表の内容は無効になります。これ以降に再移入すると、表のインメモリー圧縮単位(IMCU)がグローバル・ディクショナリを使用して再エンコードされます。このため、最初に結合グループを作成し、その後、表に移入することをお薦めします。

関連項目:

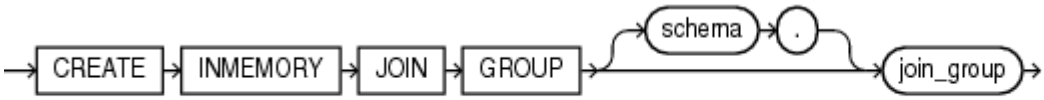
- ALTER INMEMORY JOIN GROUPおよびDROP INMEMORY JOIN GROUP
- 結合グループの詳細は、『Oracle Database In-Memoryガイド』を参照してください。

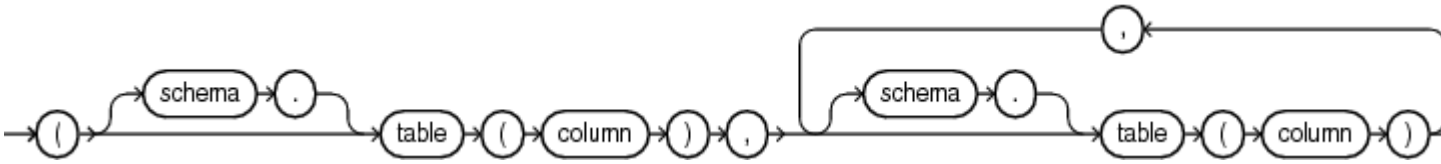
前提条件

別のユーザーのスキーマに結合グループを作成したり、結合グループに別のユーザーのスキーマ内の表の列を含めるには、CREATE ANY TABLEシステム権限が必要です。

構文

create_inmemory_join_group::=





セマンティクス

schema

結合グループを含むスキーマを指定します。schemaを指定しない場合、自分のスキーマ内に結合グループが作成されます。

join_group

作成する結合グループの名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

schema

結合グループに含まれる列を含む表のスキーマを指定します。schemaを指定しない場合、表は自分のスキーマ内に存在すると想定されます。

table

結合グループに含める列を含む表の名前を指定します。

column

結合グループに含める列の名前を指定します。結合グループには、同じ表または異なる表の列を含めることができます。

結合グループの制限事項

結合グループには、次の制限事項が適用されます。

- 結合グループには、1つ以上の列が含まれている必要があります。
- 結合グループには、最大で255列を含めることができます。
- 表の列は、最大で1つの結合グループのメンバーにすることができます。
- Oracle Active Data Guardでは、結合グループはサポートされていません。

例

次の文は、prod_id1という名前の結合グループをoeスキーマ内に作成します。この結合グループに関連する表は、どちらもoeスキーマ内に格納されます。

```
CREATE INMEMORY JOIN GROUP prod_id1
  (inventories(product_id), order_items(product_id));
```

次の文は、prod_id2という名前の結合グループをoeスキーマ内に作成します。表inventoriesはoeスキーマ内に格納され、表online_mediaはpm内に格納されます。

```
CREATE INMEMORY JOIN GROUP prod_id2
  (inventories(product_id), pm.online_media(product_id));
```

CREATE JAVA

目的

CREATE JAVAを使用すると、Javaソース、クラスまたはリソースを含むスキーマ・オブジェクトを作成できます。

関連項目:

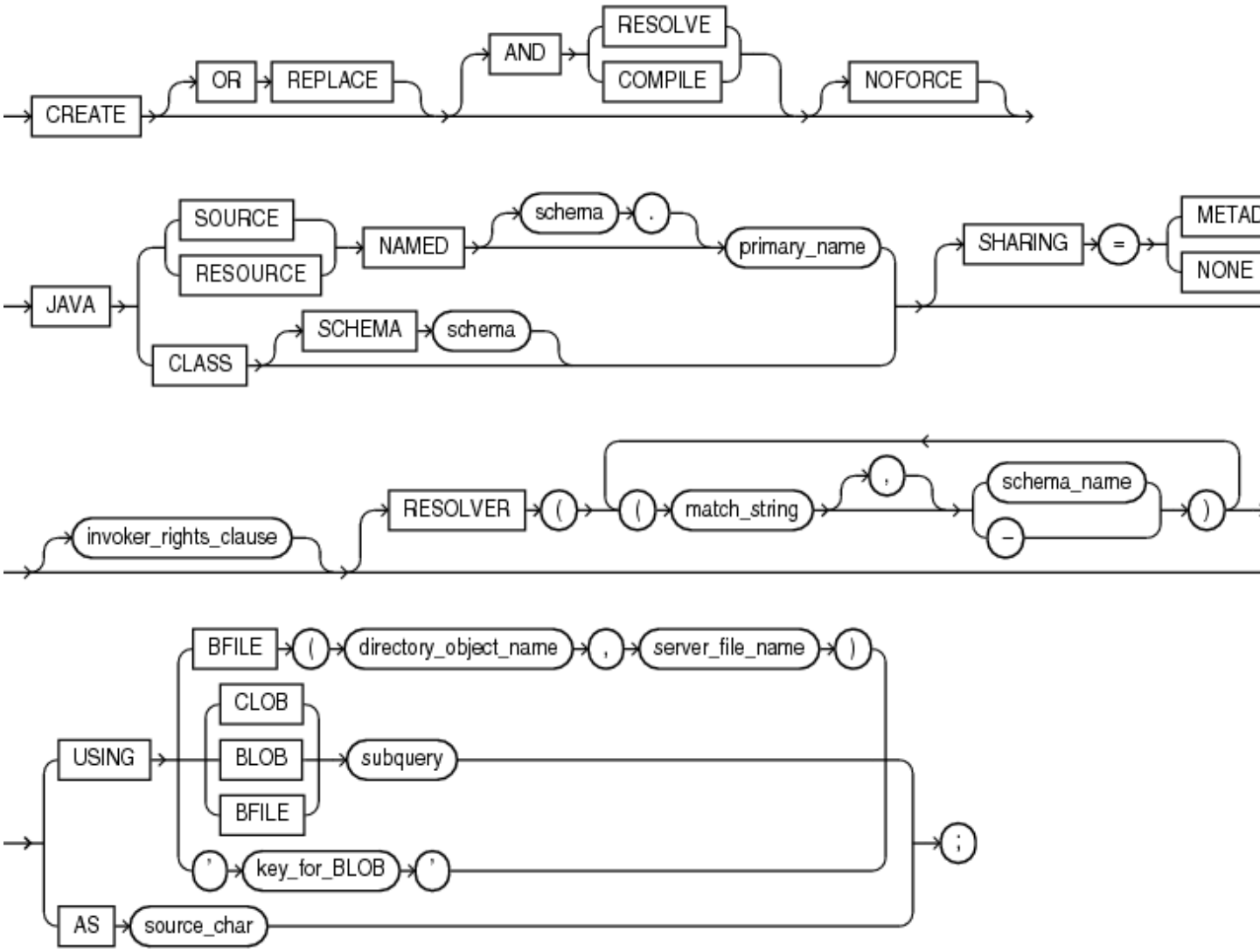
- Javaの概要およびJavaストアド・プロシージャの詳細は、『Oracle Database Java開発者ガイド』を参照してください。
- JDBCについては、『Oracle Database JDBC開発者ガイド』を参照してください。

前提条件

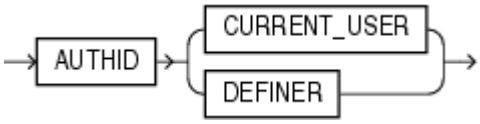
自分のスキーマにJavaソース、クラスまたはリソースを含むスキーマ・オブジェクトを作成または再作成する場合は、CREATE PROCEDUREシステム権限が必要です。他のユーザーのスキーマ内にスキーマ・オブジェクトを作成または再作成する場合は、CREATE ANY PROCEDUREシステム権限が必要です。

構文

create_java::=



invoker_rights_clause::=



セマンティクス

OR REPLACE

OR REPLACEを指定すると、既存のJavaクラス、ソースまたはリソースを含むスキーマ・オブジェクトを再作成できます。この句を指定した場合、付与されているオブジェクト権限を削除、再作成および再付与しなくても、既存のオブジェクトの定義を変更できます。

Javaスキーマ・オブジェクトを再定義し、RESOLVEまたはCOMPILEを指定した場合、そのオブジェクトが再コンパイルまたは変換されます。正常に変換またはコンパイルされたかどうかにかかわらず、Javaスキーマ・オブジェクトを参照するクラスは有効になります。

再定義したファンクションに対して権限が付与されていたユーザーは、権限を再付与されなくても、そのファンクションにアクセスできます。

関連項目:

その他の情報は、「ALTER JAVA」を参照してください。

RESOLVE | COMPILE

RESOLVEおよびCOMPILEは、同義のキーワードです。この文が正常に実行された場合に作成されるJavaスキーマ・オブジェクトを変換することを指定します。

- クラスに適用された場合、他のクラス・スキーマ・オブジェクトに対する参照名に変換されます。
- ソースに適用された場合、ソースがコンパイルされます。

RESOLVEおよびCOMPILEの制限事項

Javaリソースには、これらのキーワードを指定できません。

NOFORCE

RESOLVEまたはCOMPILEを指定しても、正常に変換またはコンパイルできない場合は、NOFORCEを指定するとCREATEコマンドの結果をロールバックできます。このオプションを指定しないと、正常に変換またはコンパイルできない場合でも何も処理が行われず、作成されたスキーマ・オブジェクトはそのままです。

JAVA SOURCE句

JAVA SOURCEを指定すると、Javaソース・ファイルをロードできます。

JAVA CLASS句

JAVA CLASSを指定すると、Javaクラス・ファイルをロードできます。

JAVA RESOURCE句

JAVA RESOURCEを指定すると、Javaリソース・ファイルをロードできます。

NAMED句

NAMED句は、Javaソースまたはリソースの場合に指定します。primary_nameは、二重引用符で囲む必要があります。長さの上限は、データベース文字セットで4000バイトです。

- Javaソースの場合、この句にはソース・コードが保持されているスキーマ・オブジェクト名を指定します。正常なCREATE JAVA SOURCE文は、ソースによって定義されたJavaクラスをそれぞれ保持するために、追加スキーマ・オブジェクトも作成します。
- Javaリソースの場合、この句にはJavaリソースを保持するスキーマ・オブジェクト名を指定します。

primary_nameの小文字、または大文字と小文字の組合せを保持するには、二重引用符を使用します。

schemaを指定しない場合、自分のスキーマにそのオブジェクトが作成されます。

NAMED Javaクラスの制限事項

NAMED句には、次の制限事項があります。

- Javaクラスには、NAMEDを指定できません。
- primary_nameは、データベース・リンクを含むことはできません。

SCHEMA句

SCHEMA句は、Javaクラスにのみ適用されます。このオプションは、Javaファイルを含むオブジェクトが存在するスキーマを指定します。この句を指定しない場合、自分のスキーマにそのオブジェクトが作成されます。

SHARING

この句は、アプリケーション・ルートにJavaスキーマ・オブジェクトを作成する場合にのみ適用されます。このタイプのオブジェクトはアプリケーション共通オブジェクトと呼ばれ、アプリケーション・ルートに属するアプリケーションPDBと共有できます。Javaスキーマ・オブジェクトの共有方法を決定するには、次の共有属性のいずれかを指定します。

- METADATA - メタデータ・リンクはJavaスキーマ・オブジェクトのメタデータを共有しますが、データは各コンテナに固有です。このタイプのJavaスキーマ・オブジェクトは、メタデータリンク・アプリケーション共通オブジェクトと呼ばれます。
- NONE - Javaスキーマ・オブジェクトは共有されません。

この句を指定しない場合、DEFAULT_SHARING初期化パラメータの値を使用して、Javaスキーマ・オブジェクトの共有属性が決定されます。DEFAULT_SHARING初期化パラメータに値が含まれていない場合、デフォルトはMETADATAです。

Javaスキーマ・オブジェクトの共有属性を作成後に変更することはできません。

関連項目:

- DEFAULT_SHARING初期化パラメータの詳細は、Oracle Databaseリファレンスを参照してください
- アプリケーション共通オブジェクトの作成の詳細は、『Oracle Database管理者ガイド』を参照してください。

invoker_rights_clause

invoker_rights_clauseを使用すると、クラスを所有するユーザーのスキーマ内で、そのユーザーの権限を使用してクラスのメソッドが実行されるか、または、CURRENT_USERのスキーマ内で、そのユーザーの権限を使用してクラスのメソッドを実行するかを指定できます。

また、この句は、問合せ、DML操作、およびその型のメンバー・ファンクションおよびプロシージャ内の動的SQL文の外部名の変換方法も定義します。

AUTHID CURRENT_USER

CURRENT_USERを使用すると、クラスのメソッドがCURRENT_USER権限で実行されることを指定できます。この句はデフォルトで、実行者権限クラスを作成します。

また、この句は、問合せ、DML操作、および動的SQL文の外部名をCURRENT_USERのスキーマで変換することも指定します。他のすべての文における外部名は、メソッドを含むスキーマで変換します。

AUTHID DEFINER

DEFINERを使用すると、クラスが存在するスキーマの所有者権限でクラスのメソッドを実行すること、およびクラスが存在するスキーマで外部名を変換することを指定できます。この句によって定義者権限クラスが作成されます。

関連項目:

- Oracle Database Java開発者ガイド
- CURRENT_USERの判断方法については、『Oracle Database PL/SQL言語リファレンス』を参照してください。

RESOLVER句

RESOLVER句を使用すると、Javaスキーマ・オブジェクトに対する完全修飾Java名のマッピングを指定できます。ここでは次のとおりです。

- match_stringは、完全修飾Java名、そのようなJava名と一致するワイルド・カードまたは任意の名前と一致するワイルド・カードを指定します。
- schema_nameは、対応するJavaスキーマ・オブジェクトを検索するスキーマを指定します。
- schema_nameの代替としてのダッシュ(-)は、match_stringが有効なJava名と一致した場合、名前が変換されないことを示します。名前の変換は成功しますが、実行時にクラスがその名前を使用することはできません。

このマッピングは、後の変換で(暗黙的に、またはALTER JAVA ... RESOLVE文で明示的に)使用されるコマンドで作成されるスキーマ・オブジェクトの定義とともに格納されます。

USING句

USINGは、Javaクラスまたはリソースに対する文字データ(CLOBまたはBFILE)またはバイナリ・データ(BLOBまたはBFILE)の順序を決定します。文字の順序を使用して、1つのファイルがJavaクラスまたはリソースに、または1つのソース・ファイルおよび1つ以上の導出クラスがJavaソースに定義されます。

BFILE句

順序を含む、オペレーティング・システム(directory_object_name)およびサーバー・ファイル(server_file_name)であらかじめ作成されているファイルのディレクトリおよびファイル名を指定します。BFILEは、通常、CREATE JAVA SOURCEによって文字順序として、CREATE JAVA CLASSまたはCREATE JAVA RESOURCEによってバイナリ順序として解析されます。

CLOB | BLOB | BFILE 副問合せ

指定した型(CLOB、BLOBまたはBFILE)の行と列を選択する副問合せを指定します。列の値は文字列を構成します。

ノート:

以前のリリースでは、USING 句は暗黙的にキーワード SELECT を提供しました。今回のリリースでは提供されません。ただし、キーワード SELECT なしの副問合せは、(下位互換性のために)サポートされています。

key_for_BLOB

key_for_BLOB句は、次の暗黙的な問合せを提供します。

```
SELECT LOB FROM CREATE$JAVA$LOB$TABLE
   WHERE NAME = 'key_for_BLOB';
```

key_for_BLOB句の制限事項

このケースを使用するには、表CREATE$JAVA$LOB$TABLEが現行のスキーマ内に存在し、かつ、BLOB型の列LOBおよびVARCHAR2型の列NAMEがこの表に含まれている必要があります。

AS source_char

Javaソースの文字列を指定します。

例

Javaクラス・オブジェクトの作成: 例

次の文は、Javaバイナリ・ファイルに含まれる名前を使用して、Javaクラスを含むスキーマ・オブジェクトを作成します。

```
CREATE JAVA CLASS USING BFILE (java_dir, 'Agent.class')
/
```

この例では、JavaクラスAgent.classを含むオペレーティング・システム・ディレクトリを指すディレクトリ・オブジェクトjava_dirがすでに存在していると想定しています。この例では、クラス名がJavaクラス・スキーマ・オブジェクトの名前を決定します。

Javaソース・オブジェクトの作成: 例

次の文は、Javaソース・スキーマ・オブジェクトを作成します。

```
CREATE JAVA SOURCE NAMED "Welcome" AS
   public class Welcome {
      public static String welcome() {
         return "Welcome World";   } }
/
```

Javaリソース・オブジェクトの作成: 例

次の文は、apptextという名前のJavaリソース・スキーマ・オブジェクトをbfileから作成します。

```
CREATE JAVA RESOURCE NAMED "appText"
   USING BFILE (java_dir, 'textBundle.dat')
/
```

14 SQL文: CREATE LIBRARYからCREATE SCHEMA

この章では、次のSQL文について説明します。

- CREATE LIBRARY
- CREATE LOCKDOWN PROFILE
- CREATE MATERIALIZED VIEW
- CREATEMATERIALIZEDVIEWLOG
- CREATE MATERIALIZED ZONEMAP
- CREATE OPERATOR
- CREATE OUTLINE
- CREATE PACKAGE
- CREATE PACKAGE BODY
- CREATE PFILE
- CREATE PLUGGABLE DATABASE
- CREATE PROCEDURE
- CREATE PROFILE
- CREATE RESTORE POINT
- CREATE ROLE
- CREATE ROLLBACK SEGMENT
- CREATE SCHEMA

CREATE LIBRARY

目的

CREATE LIBRARY文を使用すると、オペレーティング・システム共有ライブラリに関連するスキーマ・オブジェクトを作成できます。このスキーマ・オブジェクトの名前は、CREATE FUNCTIONまたはCREATE PROCEDURE文のcall_specで使用できます。また、パッケージまたは型におけるファンクションまたはプロシージャを宣言する際にも使用できます。これによって、SQLおよびPL/SQLは、第三世代言語(3GL)ファンクションおよびプロシージャに対してコールできます。

関連項目:

ファンクションおよびプロシージャの詳細は、「CREATE FUNCTION」および『Oracle Database PL/SQL言語リファレンス』を参照してください。

前提条件

CREATE LIBRARY文は、共有ライブラリおよび動的リンクをサポートするプラットフォーム上でのみ有効です。

自分のスキーマ内にライブラリを作成する場合は、CREATE LIBRARYシステム権限が必要です。他のユーザーのスキーマ内にライブラリを作成する場合は、CREATE ANY LIBRARYシステム権限が必要です。

CREATE FUNCTION文のcall_specでライブラリを使用したり、パッケージまたは型のファンクションを宣言する場合には、そのライブラリに対するEXECUTEオブジェクト権限とCREATE FUNCTIONシステム権限が必要です。CREATE FUNCTION文のcall_specの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

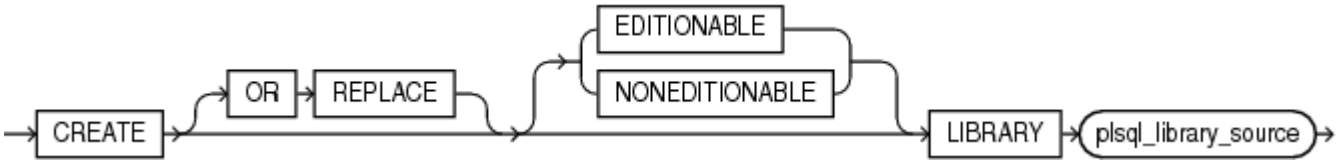
CREATE PROCEDURE文のcall_specでライブラリを使用したり、パッケージまたは型のプロシージャを宣言する場合には、そのライブラリに対するEXECUTEオブジェクト権限とCREATE PROCEDUREシステム権限が必要です。CREATE PROCEDURE文のcall_specの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

call_specで定義したプロシージャまたはファンクション(パッケージまたは型で定義したプロシージャまたはファンクションを含む)を実行するには、そのプロシージャまたはファンクションに対するEXECUTEオブジェクト権限が必要です(ただし、ライブラリに対するEXECUTEオブジェクト権限は不要です)。

構文

ライブラリはPL/SQLを使用して定義されます。このため、このマニュアルの構文図ではSQLキーワードのみを示します。PL/SQLの構文、セマンティクスおよび例については、『Oracle Database PL/SQL言語リファレンス』を参照してください。

create_library::=



(plsql_library_source: 『Oracle Database PL/SQL言語リファレンス』を参照。)

セマンティクス

OR REPLACE

OR REPLACEを指定すると、既存のライブラリを再作成できます。この句を指定した場合、既存のライブラリに付与されているオブジェクト権限を削除、再作成および再付与しなくても、ライブラリの定義を変更できます。

再定義したライブラリに対して権限を付与されていたユーザーは、権限を再付与されなくてもライブラリにアクセスできます。

[EDITIONABLE | NONEDITIONABLE]

この句を使用すると、schemaのスキーマ・オブジェクト・タイプLIBRARYのエディションニングが有効化されたときに、そのライブラリをエディション・オブジェクトにするか非エディション・オブジェクトにするかを指定できます。デフォルトは、EDITIONABLEです。エディション・オブジェクトと非エディション・オブジェクトの詳細は、『Oracle Database開発ガイド』を参照してください。

plsql_library_source

plsql_library_sourceの構文およびセマンティクスについては、『Oracle Database PL/SQL言語リファレンス』を参照してください。

CREATE LOCKDOWN PROFILE

目的

CREATE LOCKDOWN PROFILE文を使用すると、PDBロックダウン・プロファイルを作成できます。マルチテナント・コンテナ・データベース(CDB)でPDBロックダウン・プロファイルを使用して、PDBのユーザー操作を制限できます。

PDBロックダウン・プロファイルの作成後、ALTER LOCKDOWN PROFILE文でプロファイルに制限事項を追加できます。特定のデータベースの機能、オプションおよびSQL文に関連付けられるユーザー操作を制限できます。

ロックダウン・プロファイルがPDBに割り当てられている場合、該当のPDBで、ユーザーがプロファイルに対して無効化されている操作を実行することはできません。ロックダウン・プロファイルを割り当てるには、PDB_LOCKDOWN初期化パラメータの値にその名前を設定します。ロックダウン・プロファイルは、次のように、個別のPDBまたはCDBやアプリケーション・コンテナ内のすべてのPDBに割り当てることができます。

- CDBルートへの接続中にPDB_LOCKDOWNを設定すると、ロックダウン・プロファイルは、CDB内のすべてのPDBに適用されます。CDBルートには適用されません。
- アプリケーション・ルートへの接続中にPDB_LOCKDOWNを設定すると、ロックダウン・プロファイルは、アプリケーション・ルートおよびアプリケーション・コンテナ内のすべてのPDBに適用されます。
- 特定のPDBへの接続中にPDB_LOCKDOWNを設定すると、ロックダウン・プロファイルはそのPDBに適用され、CDBまたはアプリケーション・コンテナ用のロックダウン・プロファイル(存在する場合)を上書きします。

関連項目:

- 「ALTER LOCKDOWN PROFILE」および「DROP LOCKDOWN PROFILE」を参照してください。
- PDBロックダウン・プロファイルの詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

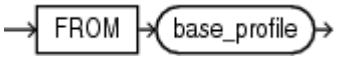
前提条件

- CREATE LOCKDOWN PROFILE文は、CDBまたはアプリケーション・ルートから発行する必要があります。
- この文が発行されるコンテナのCREATE LOCKDOWN PROFILEシステム権限を持っている必要があります。
- PDBロックダウン・プロファイルの名前は、この文が発行されるコンテナ内で一意である必要があります。

構文

create_lockdown_profile::=

static_base_profile ::=



dynamic_base_profile ::=

セマンティクス

profile_name

指定した名前で新しいPDBロックダウン・プロファイルを作成できます。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。ロックダウン・プロファイルは、静的または動的なベース・プロファイルから取得できます。

static_base_profile

ベース・プロファイルを使用して新しいロックダウン・プロファイルを作成するには、このオプションを使用します。プロファイルの作成時に有効なベース・プロファイルのルールは、新しいロックダウン・プロファイルにコピーされます。ロックダウン・プロファイルの作成後にベース・プロファイルに対して行われた変更は、ロックダウン・プロファイルには適用されません。

dynamic_base_profile

ベース・プロファイルに対する変更に伴って変更される新しいロックダウン・プロファイルを作成するには、このオプションを使用します。新しいロックダウン・プロファイルは、ベース・プロファイルのDISABLEルールと、それ以降にベース・プロファイルに対して行われる変更を継承します。ベース・プロファイルのルールがロックダウン・プロファイルに明示的に追加されるルールと競合する場合は、ベース・プロファイルのルールが優先されます。たとえば、ベース・プロファイルのOPTION_VALUE句は、動的なベース・プロファイルのOPTION_VALUE句よりも優先されます。

例

次の文は、動的ベース・プロファイルPRIVATE_DBAASを使用してPDBロックダウン・プロファイルhr_profを作成します。

```
CREATE LOCKDOWN PROFILE hr_prof INCLUDING PRIVATE_DBAAS;
```

CREATE MATERIALIZED VIEW

目的

CREATE MATERIALIZED VIEW文を使用すると、マテリアライズド・ビューを作成できます。マテリアライズド・ビューは、問合せ結果を含むデータベース・オブジェクトです。問合せのFROM句には、表、ビューおよびその他のマテリアライズド・ビューを指定できます。これらのオブジェクトをあわせて、マスター表(レプリケーション用語)またはディテール表(データ・ウェアハウス用語)といいます。このマニュアルでは、「マスター表」を使用します。マスター表が格納されているデータベースをマスター・データベースといいます。

ノート:

下位互換性を保つために、MATERIALIZED VIEW のかわりにキーワード SNAPSHOT もサポートされています。

レプリケーションでは、マテリアライズド・ビューを使用すると、ローカル・ノード上にあるリモート・データの読取り専用コピーをメンテナンスできます。マテリアライズド・ビューのデータを、表またはビューと同じように選択することができます。レプリケーション環境では、通常作成されるマテリアライズド・ビューは、主キー、ROWID、オブジェクトおよび副問合せのマテリアライズド・ビューです。

関連項目:

レプリケーションのサポートに使用するマテリアライズド・ビューのタイプの詳細は、『Oracle Database管理者ガイド』を参照してください。

データ・ウェアハウスでは、通常作成されるマテリアライズド・ビューは、マテリアライズド集計ビュー、単一表マテリアライズド集計ビューおよびマテリアライズド結合ビューです。3つのマテリアライズド・ビューはすべてクエリー・リライトで使用できます。クエリー・リライトとは、マスター表に関して記述されたユーザー要求を、1つ以上のマテリアライズド・ビューを含む意味的に同等の要求に変換するための最適化手法です。

関連項目:

- ALTER MATERIALIZED VIEW
- データ・ウェアハウスのサポートに使用するマテリアライズド・ビューの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

前提条件

マテリアライズド・ビューの作成に必要な権限は、ロールを介してではなく、直接付与する必要があります。

自分のスキーマ内にマテリアライズド・ビューを作成する場合は、次の条件に従う必要があります。

- CREATE MATERIALIZED VIEWシステム権限と、CREATE TABLEまたはCREATE ANY TABLEのいずれかのシステム権限が必要です。
- 所有していないマテリアライズド・ビューのマスター表にアクセスする場合は、各表に対するREADまたはSELECTオブジェクト権限またはREAD ANY TABLEまたはSELECT ANY TABLEシステム権限が必要です。

他のユーザーのスキーマ内にマテリアライズド・ビューを作成する場合、次の条件に従う必要があります。

- CREATE ANY MATERIALIZED VIEWシステム権限が必要です。
- マテリアライズド・ビューの所有者には、CREATE TABLEシステム権限が必要です。スキーマ所有者が所有していないマテリアライズド・ビューの任意のマスター表(リモート・データベースに存在するマスター表など)、およびそのマスター表に定義された任意のマテリアライズド・ビュー・ログにアクセスするには、各表に対するREADまたはSELECTオブジェクト権限、あるいはREAD ANY TABLEまたはSELECT ANY TABLEシステム権限が必要です。

REFRESH ON COMMITモードのマテリアライズド・ビューを作成する場合(REFRESH ON COMMIT句を使用する場合)は、前述の権限の他に、所有していないマスター表に対するON COMMIT REFRESHオブジェクト権限、またはON COMMIT REFRESHシステム権限が必要です。

前述の権限の他にも、クエリー・リライトが使用可能なマテリアライズド・ビューを作成する場合は、次の条件に従う必要があります。

- スキーマ所有者がマスター表を所有していない場合は、そのスキーマ所有者にはGLOBAL QUERY REWRITE権限、または自分のスキーマ以外の各表に対するQUERY REWRITEオブジェクト権限が必要です。
- 事前作成コンテナにマテリアライズド・ビューを定義する(ON PREBUILT TABLE句を使用)場合は、コンテナ表に対するWITH GRANT OPTION付きのREADまたはSELECT権限が必要です。

マテリアライズド・ビューを含むスキーマのユーザーには、マテリアライズド・ビューのマスター表および索引を格納するターゲット表領域への十分な割当て制限またはUNLIMITED TABLESPACEシステム権限が必要です。

マテリアライズド・ビューを作成すると、そのマテリアライズド・ビューのスキーマ内に、1つの内部表および1つ以上の索引が作成されます。また、1つのビューが作成されることもあります。これらのオブジェクトは、マテリアライズド・ビューのデータをメンテナンスするために使用されます。ユーザーには、これらのオブジェクトを作成するための権限が必要です。

コミットSCNベース・マテリアライズド・ビュー・ログを使用するマスター表には、次のようなローカル・マテリアライズド・ビューを作成できます(ON COMMITおよびON DEMANDを含む)。

- マテリアライズド集計ビュー(単一表に対するマテリアライズド集計ビューを含む)
- マテリアライズド結合ビュー
- 主キーおよびROWIDに基づく単一表マテリアライズド・ビュー
- UNION ALLマテリアライズド・ビュー(各UNION ALLブランチはここに示したマテリアライズド・ビューのいずれかのタイプ)

コミットSCNベース・マテリアライズド・ビュー・ログを使用するマスター表には、リモート・マテリアライズド・ビューを作成できません。

異なるマテリアライズド・ビュー・ログを使用するマスター表(つまり、タイムスタンプ・ベース・マテリアライズド・ビュー・ログを使用するマスター表およびコミットSCNベース・マテリアライズド・ビュー・ログを使用するマスター表)にはマテリアライズド・ビューを作成できず、ORA-32414が戻されます。

evaluation_edition_clauseまたはunusable_editions_clauseでエディションを指定するには、そのエディションに対するUSE権限が必要になります。

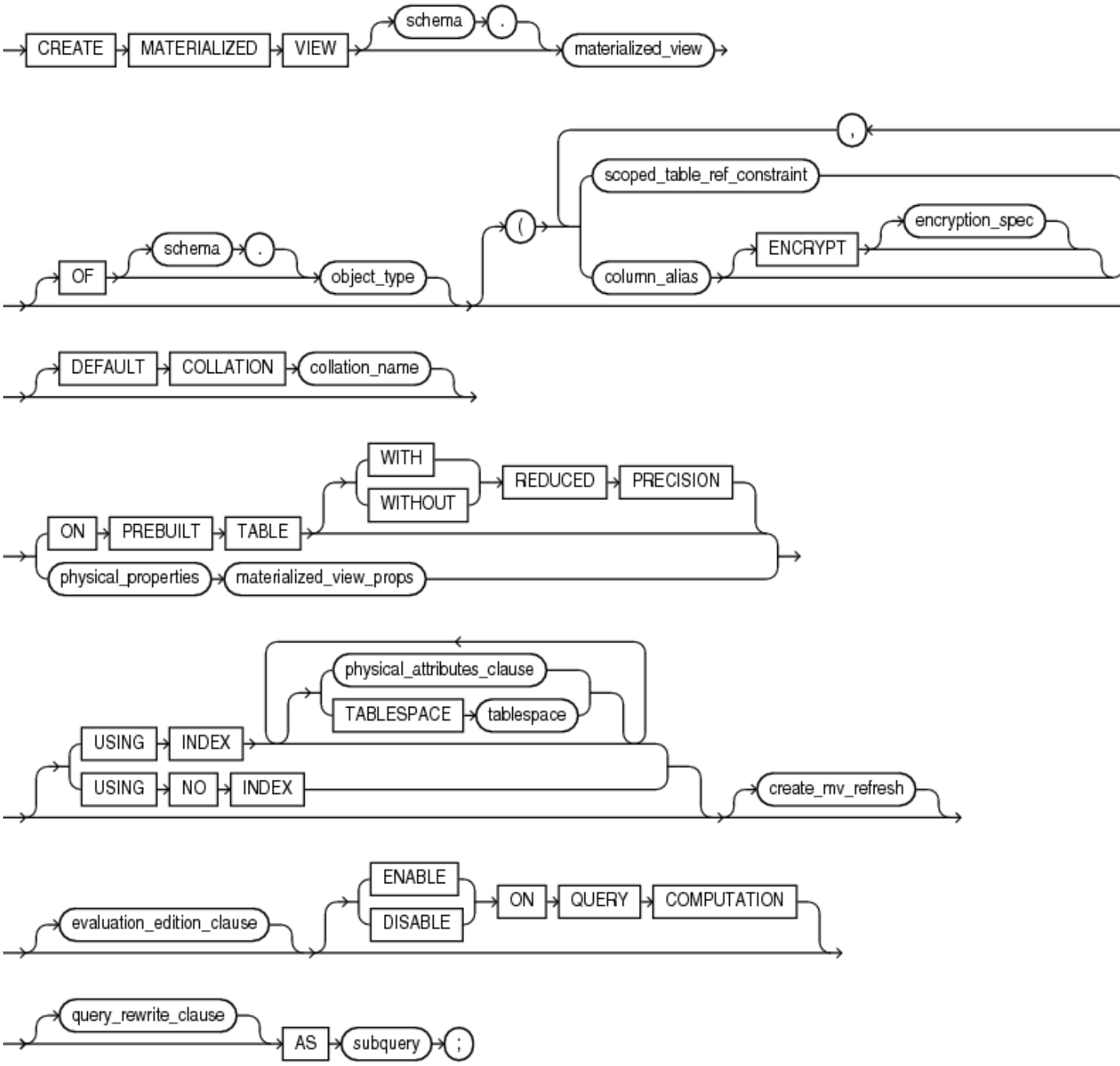
関連項目:

- これらの権限については、「CREATE TABLE」、「CREATE VIEW」および「CREATE INDEX」を参照してください。
- レプリケーションのためのマテリアライズド・ビューの作成に適用される前提条件の詳細は、『Oracle Database管理者ガイド』を参照してください。

- データ・ウェアハウスのためのマテリアライズド・ビューの作成についての前提条件は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

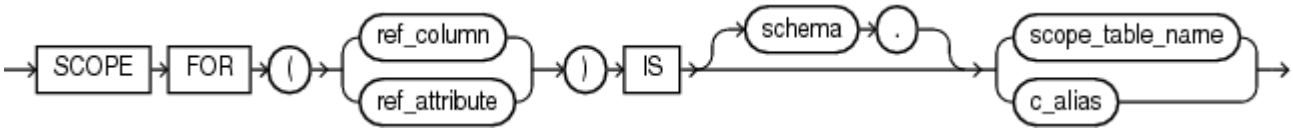
構文

create_materialized_view::=

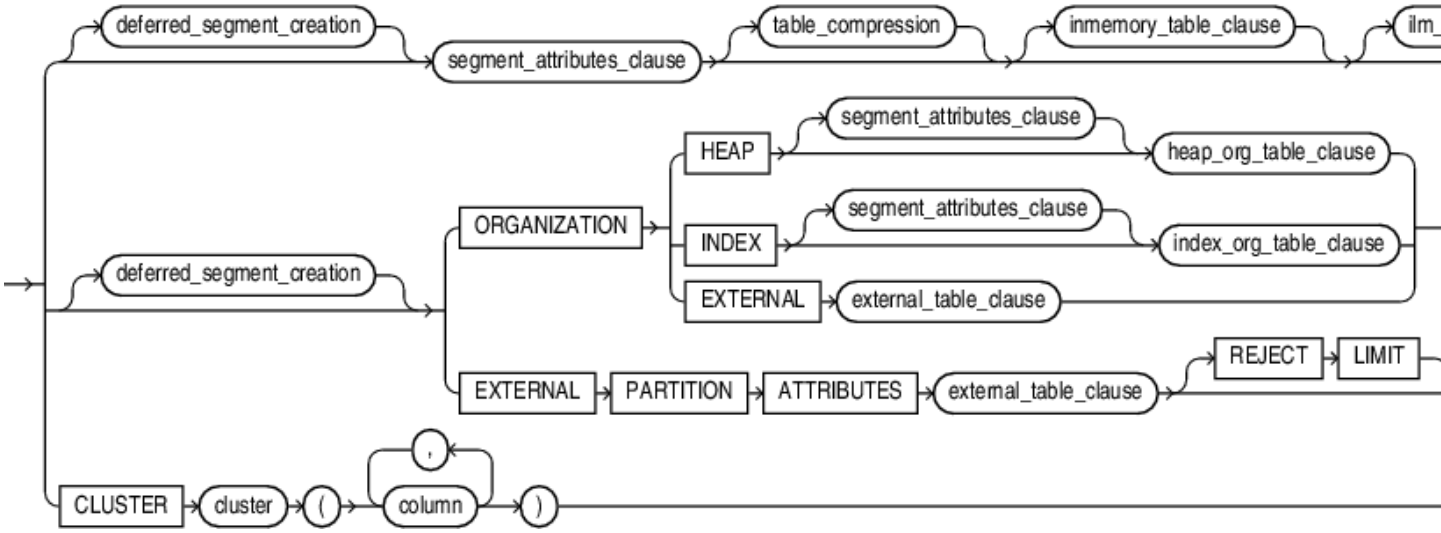


(scoped_table_ref_constraint::=、physical_properties::=、materialized_view_props::=、physical_attributes_clause::=、create_mv_refresh::=、evaluation_edition_clause::=、query_rewrite_clause::=、subquery::=)

scoped_table_ref_constraint::=

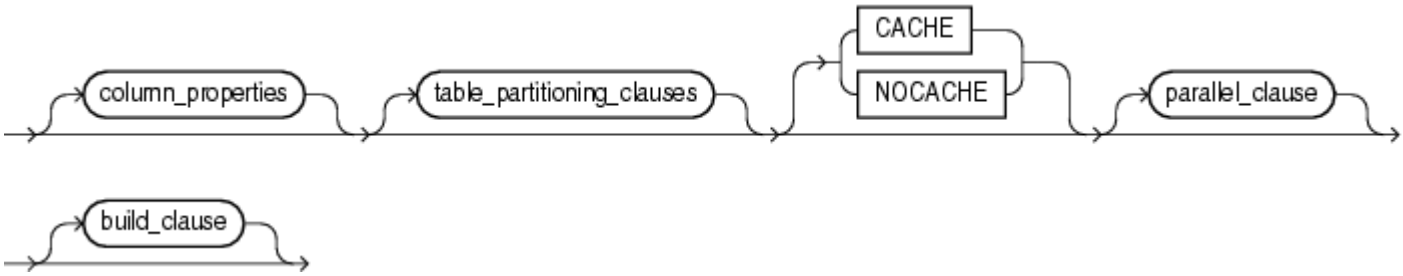


physical_properties::=



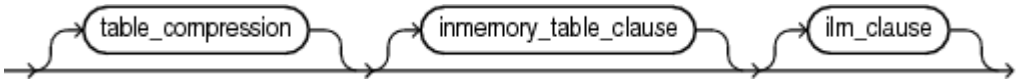
(deferred_segment_creation::=、segment_attributes_clause::=、table_compression::=、inmemory_table_clause::=、heap_org_table_clause::=、index_org_table_clause::=)

materialized_view_props::=

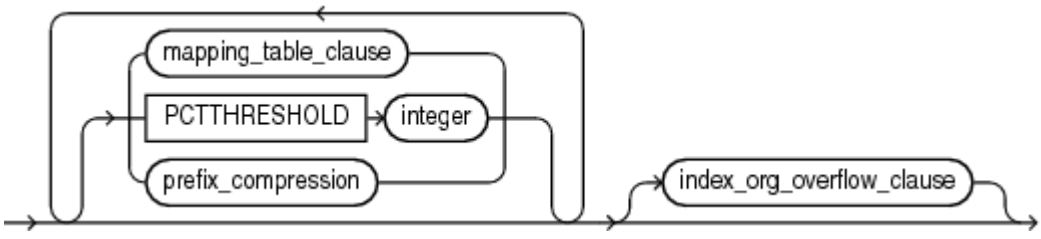


(column_properties::=、table_partitioning_clauses::= (CREATE TABLE構文の一部)、parallel_clause::=、build_clause::=)

heap_org_table_clause::=

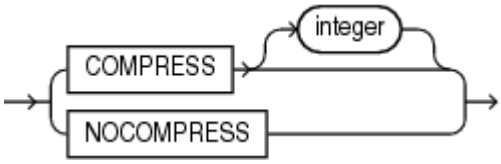


index_org_table_clause::=



(mapping_table_clause: マテリアライズド・ビューではサポートされていません、prefix_compression::=、index_org_overflow_clause::=)

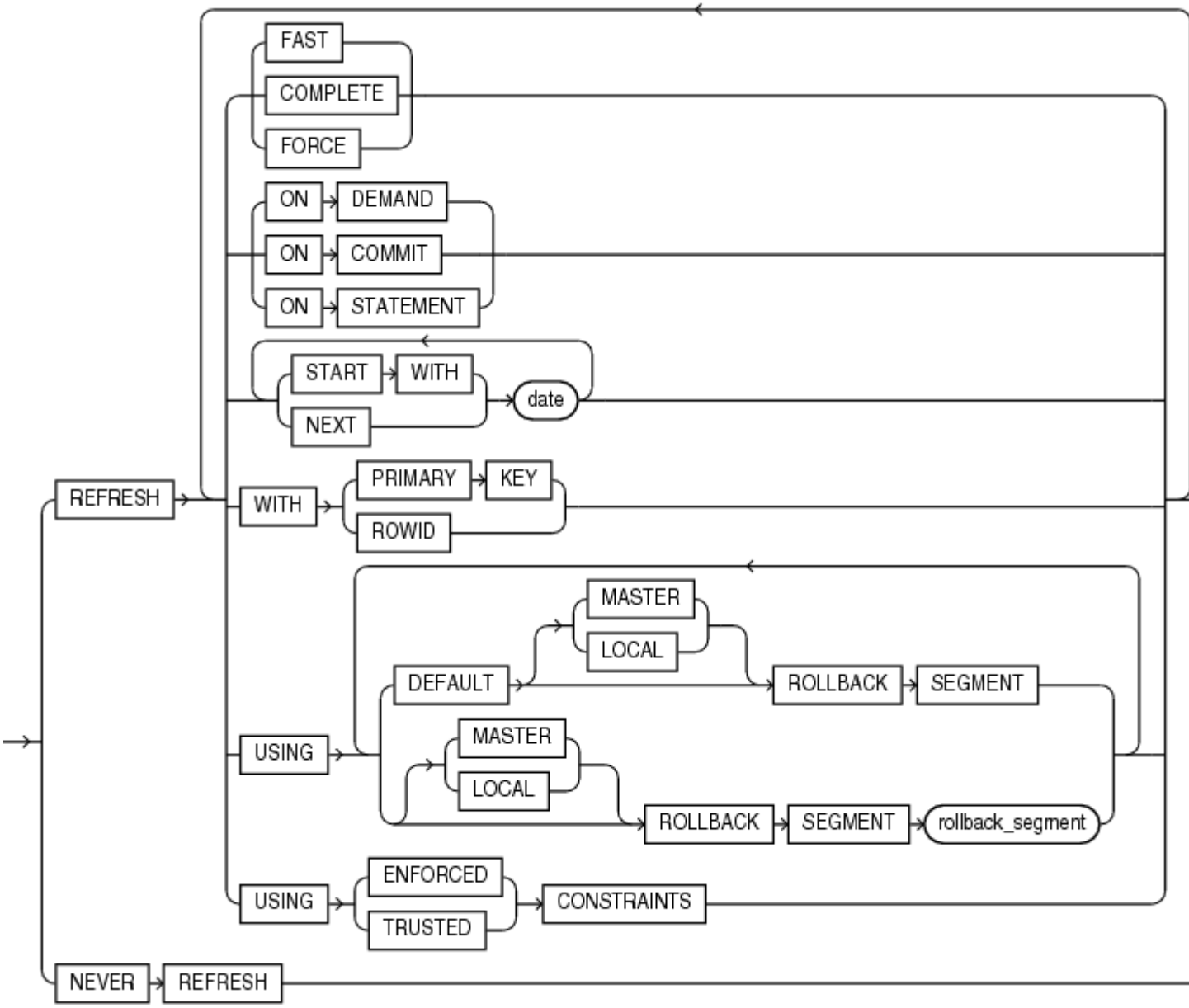
prefix_compression::=



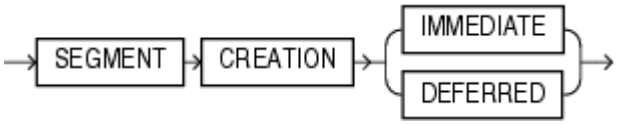
index_org_overflow_clause::=

(segment_attributes_clause::=)

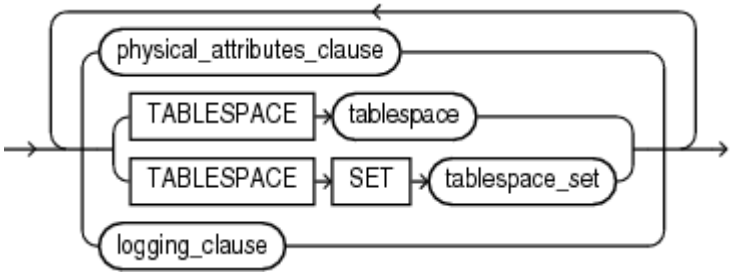
create_mv_refresh::=



deferred_segment_creation::=

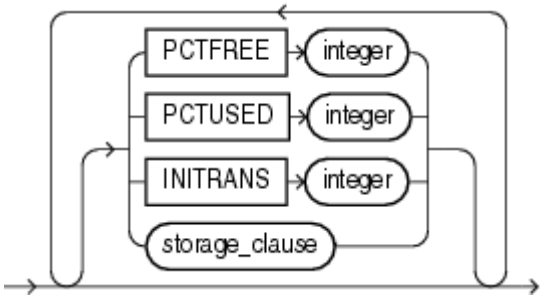


segment_attributes_clause::=



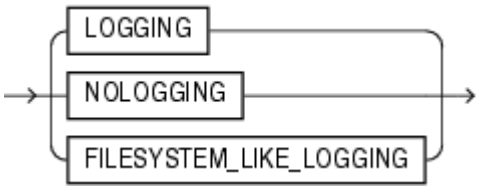
(physical_attributes_clause::=、TABLESPACE SET: CREATE MATERIALIZED VIEWではサポートされていません、logging_clause::=)

physical_attributes_clause::=

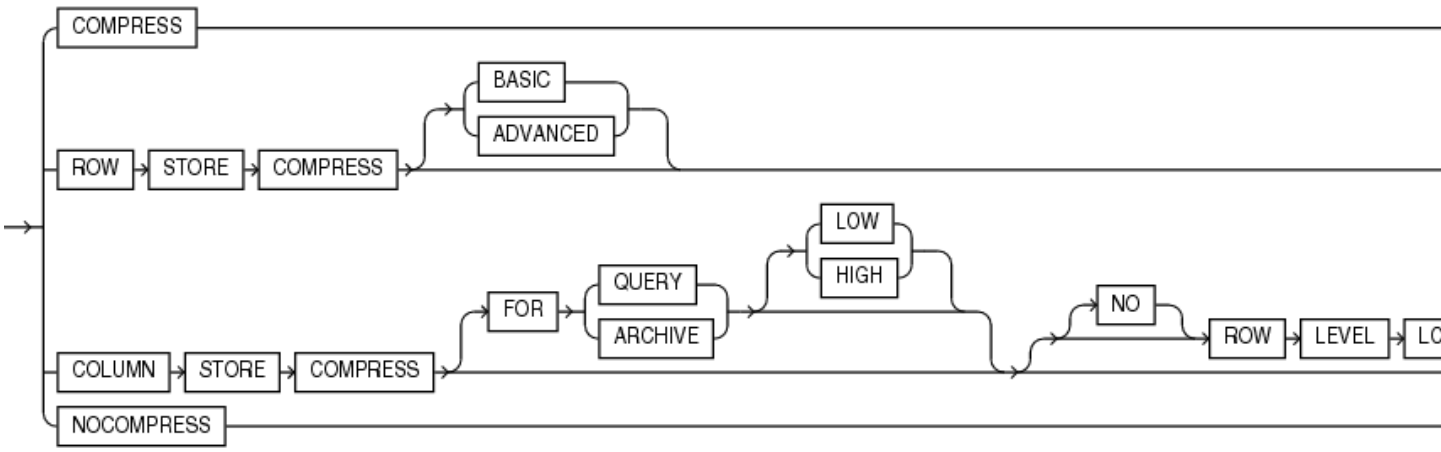


(logging_clause::=)

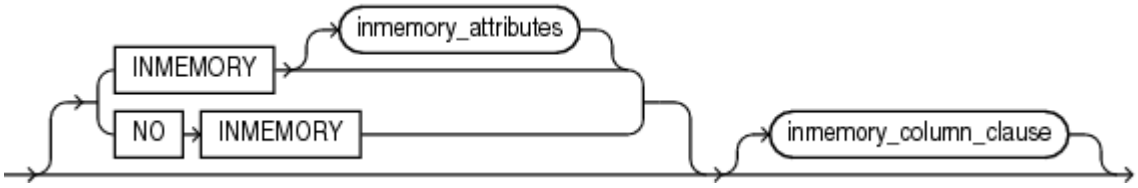
logging_clause::=



table_compression::=



inmemory_table_clause::=

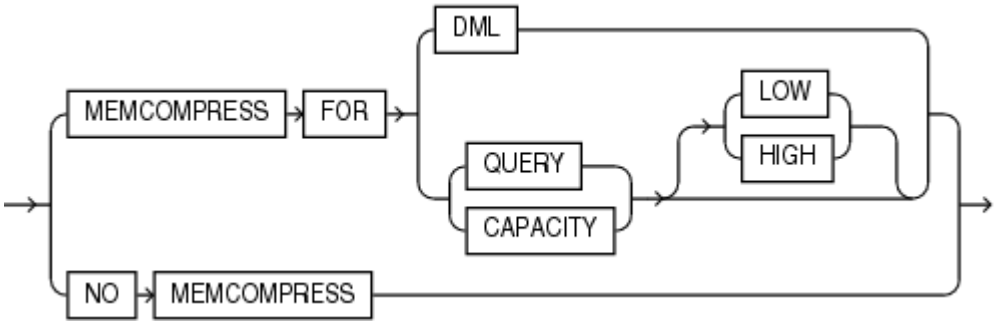


(inmemory_attributes::=、inmemory_column_clause::=)

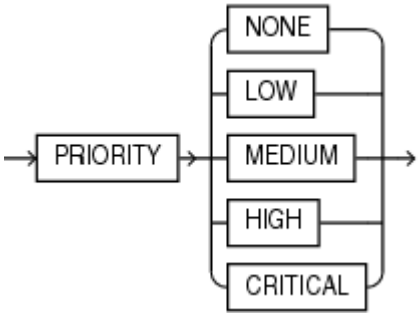
inmemory_attributes::=

(inmemory_memcompress::=、inmemory_priority::=、inmemory_distribute::=、inmemory_duplicate::=)

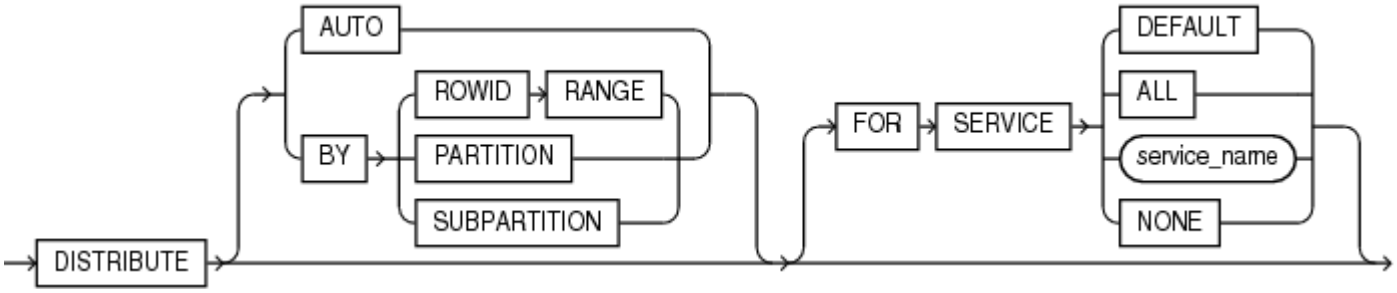
inmemory_memcompress::=



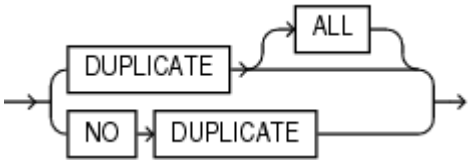
inmemory_priority::=



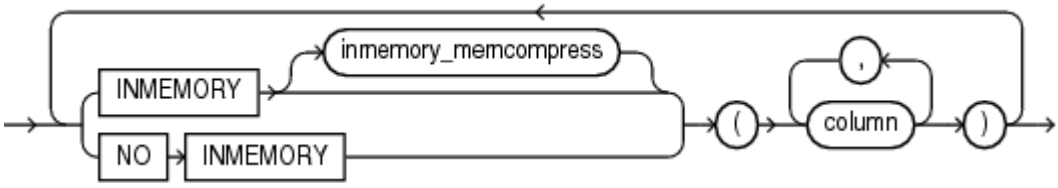
inmemory_distribute::=



inmemory_duplicate::=

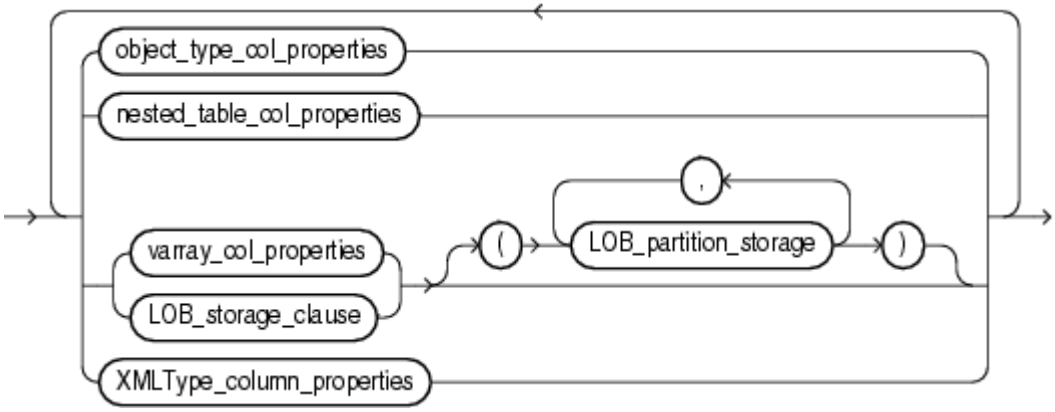


inmemory_column_clause::=



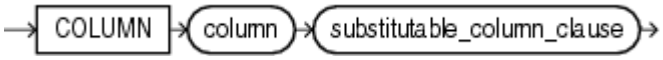
(inmemory_memcompress::=)

column_properties::=



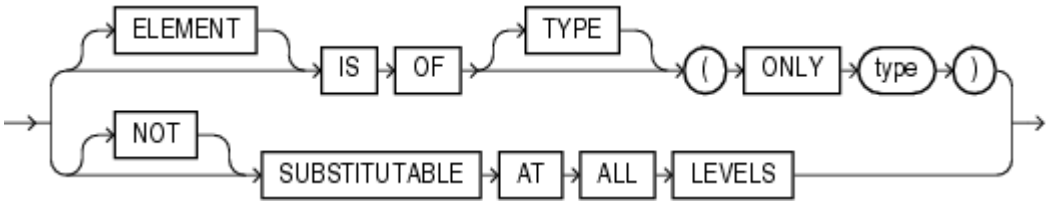
(object_type_col_properties::=、nested_table_col_properties::=、varray_col_properties::=、LOB_partition_storage::=、LOB_storage_clause::=。XMLType_column_propertiesは、マテリアライズド・ビューではサポートされていません)

object_type_col_properties::=

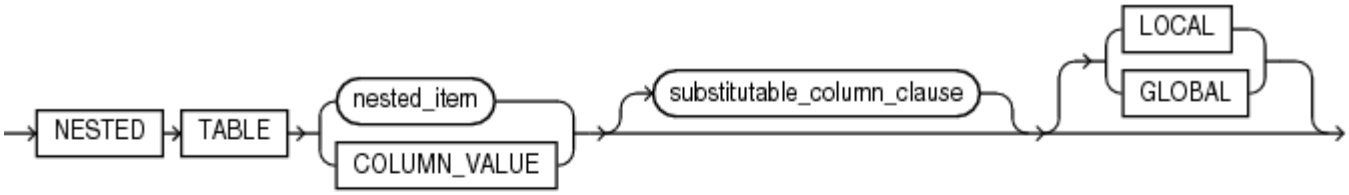


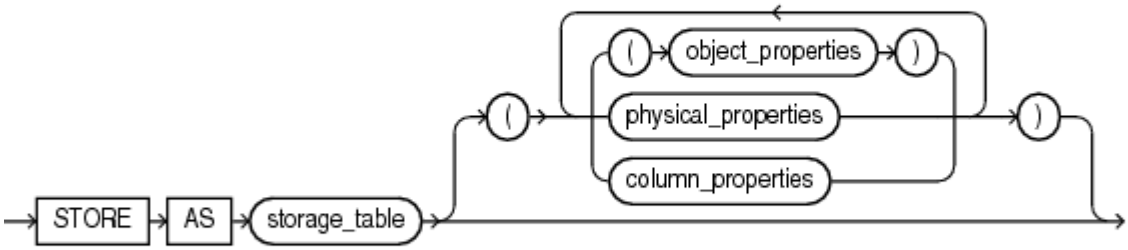
(substitutable_column_clause::=)

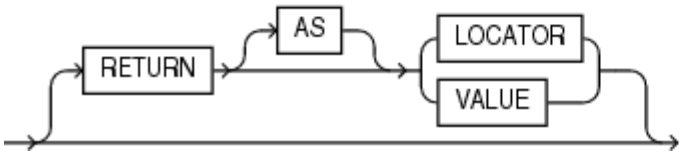
substitutable_column_clause::=



nested_table_col_properties::=

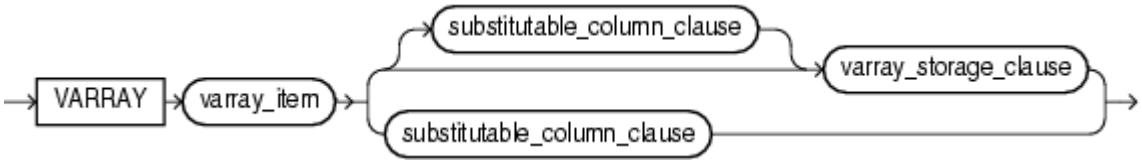






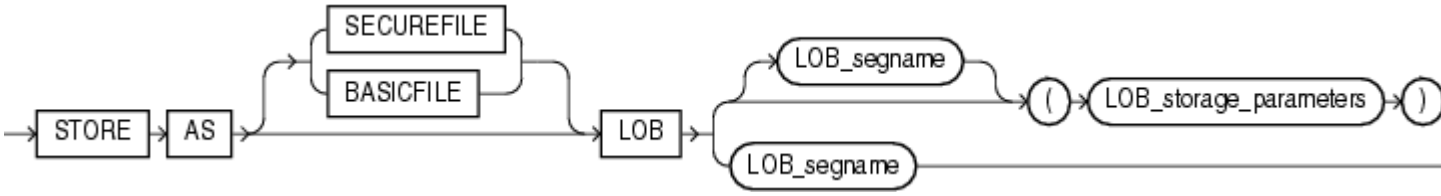
(substitutable_column_clause::=、object_properties::=、physical_properties::= (CREATE TABLE構文の一部)、column_properties::=)

varray_col_properties::=



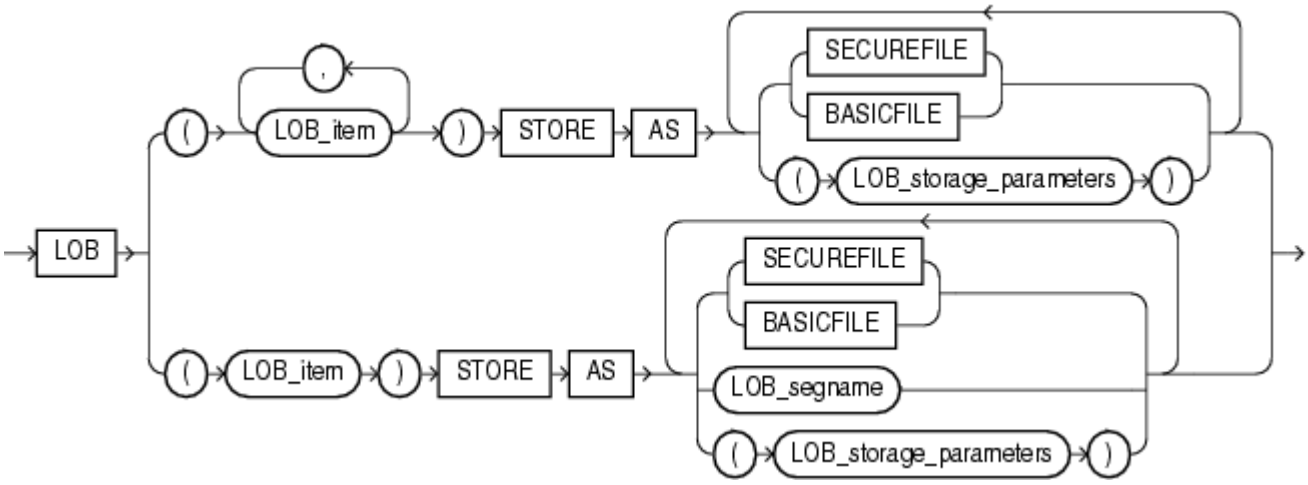
(substitutable_column_clause::=、varray_storage_clause::=)

varray_storage_clause::=



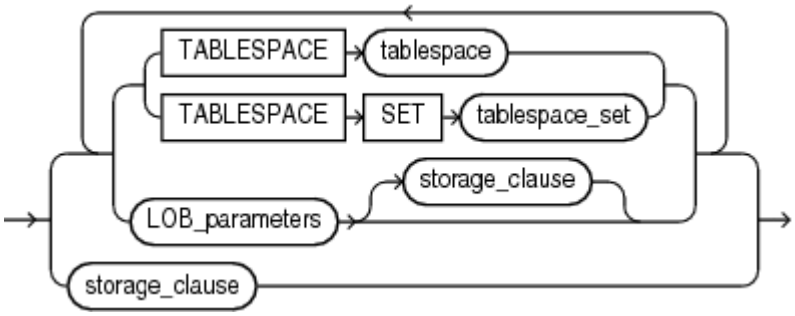
(LOB_parameters::=)

LOB_storage_clause::=



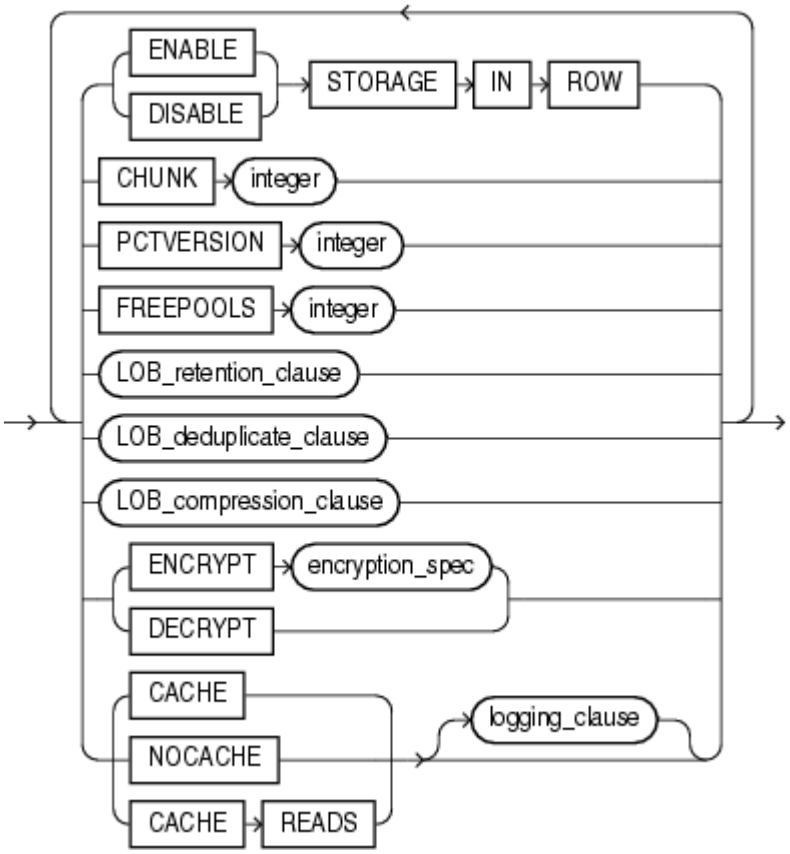
(LOB_storage_parameters::=)

LOB_storage_parameters::=



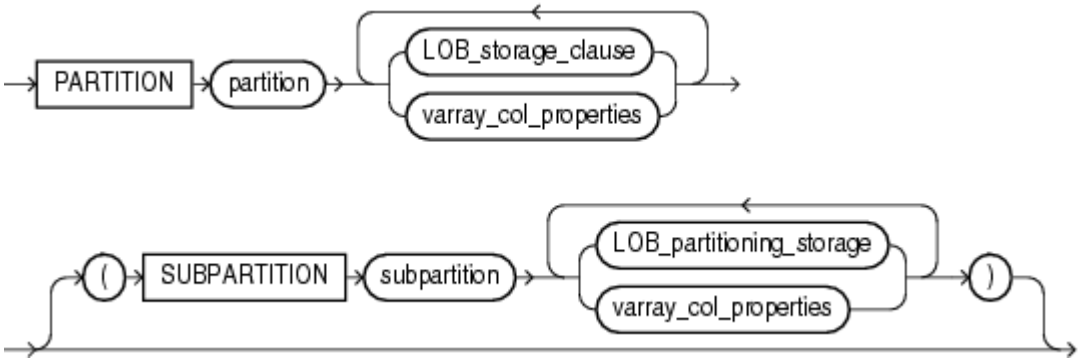
(TABLESPACE SET: CREATE MATERIALIZED VIEWではサポートされていません、LOB_parameters::=、storage_clause::=)

LOB_parameters::=



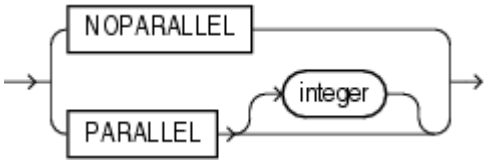
(storage_clause::=、logging_clause::=)

LOB_partition_storage::=

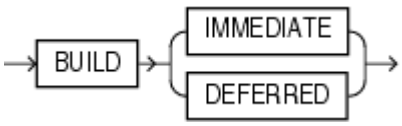


(LOB_storage_clause::=、varray_col_properties::=)

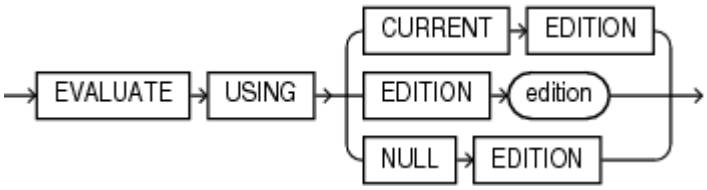
parallel_clause::=



build_clause::=

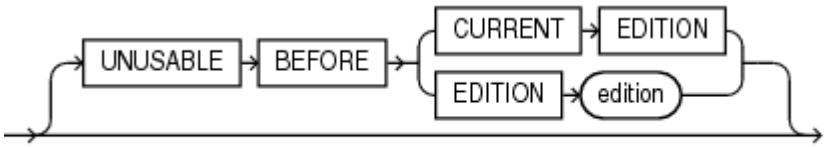


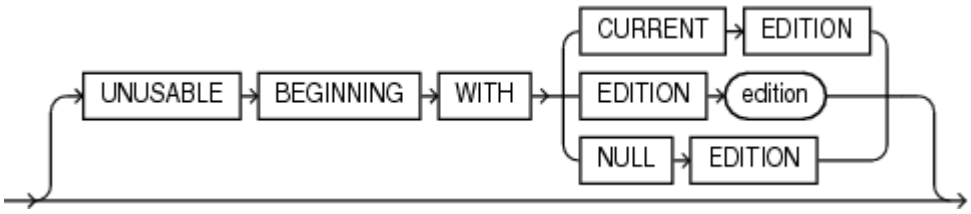
evaluation_edition_clause::=



query_rewrite_clause::=

unusable_editions_clause::=





セマンティクス

schema

マテリアライズド・ビューを含めるスキーマを指定します。schemaを指定しない場合、自分のスキーマにそのマテリアライズド・ビューが作成されます。

materialized_view

作成するマテリアライズド・ビューの名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。Oracle Databaseは、マテリアライズド・ビュー名に接頭辞または接尾辞を追加して、マテリアライズド・ビューをメンテナンスするための表および索引の名前を生成します。

column_alias

マテリアライズド・ビューの各列に対して別名を指定できます。列の別名のリストによって、競合する列名が明示的に解決されます。したがって、マテリアライズド・ビューのSELECT句で別名を指定する必要がなくなります。この句で列の別名を指定する場合は、SELECT句内で参照される各データソースの別名を指定する必要があります。

ENCRYPT句

この句を使用すると、マテリアライズド・ビューの列を暗号化できます。列の暗号化の詳細は、「CREATE TABLE」の句「encryption_spec」を参照してください。

OF object_type

OF object_type句を指定すると、object_type型のオブジェクト・マテリアライズド・ビューを明示的に作成できます。

関連項目:

OF type_name句の詳細は、「CREATE TABLE」の「object_table」を参照してください。

scoped_table_ref_constraint

SCOPE FOR句を使用すると、参照の有効範囲を1つのオブジェクト表に制限できます。scope_table_nameを持つ表名または列の別名を参照できます。REF列または属性の値はscope_table_nameまたはc_alias内のオブジェクトを指し、その場所にREF列と同じ型のオブジェクト・インスタンスが格納されます。別名を指定する場合は、その別名が、マテリアライズド・ビューを定義する問合せのSELECT構文のリストの列と1対1で対応する必要があります。

関連項目:

詳細は、「REF列のSCOPE制約」を参照してください。

DEFAULT COLLATION

この句を使用して、マテリアライズド・ビューのデフォルトの照合を指定します。デフォルトの照合は、マテリアライズド・ビューを定義する問合せに含まれるすべての文字リテラルについて、導出された照合として使用されます。デフォルトの照合は、マテリアライズド・ビューの列では使用されません。マテリアライズド・ビューの列の照合は、マテリアライズド・ビューの定義副問合せから導出されます。CREATE MATERIALIZED VIEW文は、エラーが発生して失敗するか、その文字列のいずれかが、導出された照合がない定義副問合せの式に基づいている場合、マテリアライズド・ビューは無効な状態で作成されます。

collation_nameには、有効な名前付き照合または疑似照合を指定します。

この句を省略した場合、マテリアライズド・ビューのデフォルトの照合は、マテリアライズド・ビューを含むスキーマの有効スキーマのデフォルトの照合に設定されます。有効なスキーマのデフォルトの照合の詳細は、「ALTER SESSION」のDEFAULT_COLLATION句を参照してください。

DEFAULT COLLATION句を指定できるのは、COMPATIBLE初期化パラメータが12.2以上に設定され、かつMAX_STRING_SIZE初期化パラメータがEXTENDEDに設定されている場合のみです。

マテリアライズド・ビューのデフォルトの照合を変更するには、マテリアライズド・ビューを再作成する必要があります。

マテリアライズド・ビューのデフォルトの照合の制限事項

次の制限事項は、マテリアライズド・ビューのデフォルトの照合を指定する場合に適用されます。

- マテリアライズド・ビューを定義する問合せにWITH plsql_declarations句が含まれている場合、マテリアライズド・ビューのデフォルトの照合はUSING_NLS_COMPである必要があります。
- マテリアライズド・ビューを事前作成表に対して作成する場合、表の列の宣言された照合は、定義する問合せから導出された、マテリアライズド・ビューの列の対応する照合と同じである必要があります。

ON PREBUILT TABLE句

ON PREBUILT TABLE句を指定すると、既存の表を再初期化したマテリアライズド・ビューとして登録できます。この句は、データ・ウェアハウス環境において、大きいマテリアライズド・ビューを登録する場合に有効です。その表は、結果マテリアライズド・ビューと同じ名前で、同じスキーマにある必要があります。

マテリアライズド・ビューが削除されると、その既存の表は、1つの表としての元の形に戻ります。

ノート:

この句は、表オブジェクトが副問合せの具体化を反映することを前提としています。マテリアライズド・ビューがそのマスター表のデータを正しく反映することを保証するために、この前提が満たされていることを確認することをお薦めします。

ON PREBUILT TABLE句は、次のような使用例で役立ちます。

- 問合せの結果を示す表があります。この表の作成コストは高く、作成に長時間かかりました。この問合せに対し、マテリアライズド・ビューを作成します。この場合、ON PREBUILT TABLE句を使用することで、問合せを実行してマテリアライズド・ビューのコンテナにデータを移入するコストを回避できます。
- マテリアライズド・ビューを一時的に破棄しますが、そのマテリアライズド・ビューのコンテナ表は保持します。このために、DROP MATERIALIZED VIEW ... PRESERVE TABLE文を使用します。その後、マテリアライズド・ビューを再度作成することにします。このとき、マテリアライズド・ビューのマスター表に変更がないことが判明しています。この場合、ON PREBUILT TABLE句を使用してこのマテリアライズド・ビューを作成できます。これにより、マテリアライズド・ビューのコンテナ表を作成してデータを移入するコストと時間を節約できます。

ON PREBUILT TABLEを指定すると、I_SNAP$索引が生成されません。この索引は、高速リフレッシュのパフォーマンスを改善します。この索引の利点を活用するには、索引を手動で作成できます。詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

WITH REDUCED PRECISION

WITH REDUCED PRECISIONを指定すると、表またはマテリアライズド・ビューの列の精度が、subqueryによって戻される精度と正確に一致しない場合に、精度の低下が許可されます。

WITHOUT REDUCED PRECISION

WITHOUT REDUCED PRECISIONを指定すると、表またはマテリアライズド・ビューの列の精度が、subqueryによって戻される精度と正確に一致する必要があります。一致しない場合、作成操作は失敗します。これはデフォルトです。

事前作成表の使用の制限事項

事前作成表には、次の制限事項があります。

- subqueryの各列の別名は、事前作成表の列に対応し、対応する列のデータ型が一致している必要があります。
- この句を指定する場合、subqueryで参照されない列にデフォルト値も指定しないかぎり、その列にNOT NULL制約は指定できません。
- ROWIDマテリアライズド・ビューの作成時にON PREBUILT TABLE句を指定することはできません。

関連項目:

事前作成したマテリアライズド・ビューの作成: 例

physical_properties_clause

physical_properties_clauseの構成要素は、マテリアライズド・ビューと表に対して同一のセマンティクスを持ちます。ただし、次の項で説明する例外および追加事項があります。

physical_properties_clauseの制限事項

マテリアライズド・ビューにはORGANIZATION EXTERNALを指定できません。

deferred_segment_creation

この句を使用すると、このマテリアライズド・ビューのセグメントを作成するタイミングを指定できます。詳細は、「CREATE TABLE」の句「deferred_segment_creation」を参照してください。

segment_attributes_clause

segment_attributes_clauseを使用すると、PCTFREE、PCTUSED、INITRANSパラメータの値、およびマテリアライズド・ビューの記憶特性の設定、表領域の割当て、ロギングが実行されるかどうかを指定できます。USING INDEX句では、PCTFREEまたはPCTUSEDは指定できません。

TABLESPACE句

マテリアライズド・ビューを作成する表領域を指定します。この句を指定しないと、マテリアライズド・ビューを含むスキーマのデフォルト表領域内にマテリアライズド・ビューが作成されます。

関連項目:

デフォルト値を含むこれらの句の詳細は、「physical_attributes_clause」および「storage_clause」を参照してください。

logging_clause

LOGGINGまたはNOLOGGINGを指定すると、マテリアライズド・ビューのロギング特性を設定できます。ロギング特性は、マテリアライズド・ビューの作成およびDBMS_REFRESHパッケージによって開始される非アトミック・リフレッシュに影響します。デフォルトは、マテリアライズド・ビューが存在する表領域のロギング特性です。

関連項目:

この句の詳細は、「logging_clause」を参照してください。アトミック・リフレッシュおよび非アトミック・リフレッシュの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

table_compression

table_compression句を使用すると、ディスクおよびメモリーの使用量を削減するために、データ・セグメントを圧縮するかどうかを指定できます。この句のセマンティクスは、CREATE MATERIALIZED VIEWおよびCREATE TABLE文で同じです。この句のセマンティクスの詳細は、CREATE TABLEのドキュメントのtable_compression句を参照してください。

inmemory_table_clause

inmemory_table_clauseを使用して、インメモリー列ストア(IM列ストア)のマテリアライズド・ビューを有効化または無効化します。この句のセマンティクスは、CREATE TABLEドキュメントのinmemory_table_clauseと同じです。

inmemory_column_clause

inmemory_column_clauseを使用して、IM列ストアの特定のマテリアライズド・ビュー列を無効化し、特定の列のデータ圧縮方法を指定します。この句のセマンティクスは、CREATE TABLEのドキュメントのinmemory_column_clauseと同じですが、次が追加されます。inmemory_column_clauseを指定する場合、マテリアライズド・ビューの各列にcolumn_aliasを指定する必要もあります。

index_org_table_clause

ORGANIZATION INDEXを指定すると、索引構成マテリアライズド・ビューを作成できます。このマテリアライズド・ビューでは、

データ行は、マテリアライズド・ビューの主キーに定義した索引に格納されます。次のようなマテリアライズド・ビューの索引構成を指定できます。

- 読取り専用および更新可能なオブジェクト・マテリアライズド・ビュー。マスター表に主キーが含まれている必要があります。
- 読取り専用および更新可能な主キー・マテリアライズド・ビュー。
- 読取り専用のROWIDマテリアライズド・ビュー。

index_org_table_clauseのキーワードおよびパラメータは、CREATE TABLEと同じです。また、次の制限事項があります。

関連項目:

「CREATE TABLE」の「index_org_table_clause」を参照してください。

索引構成マテリアライズド・ビューの制限事項

索引構成マテリアライズド・ビューには、次の制限事項があります。

- CREATE MATERIALIZED VIEWのCACHE、NOCACHE、CLUSTERまたはON PREBUILT TABLE句は指定できません。
- index_org_table_clauseには次の制限事項があります。
  - mapping_table_clauseを指定できません。
  - 複合主キーに基づくマテリアライズド・ビューのみにCOMPRESSを指定できます。単一主キーと複合主キーのいずれかに基づくマテリアライズド・ビューにNOCOMPRESSを指定できます。

CLUSTER句

CLUSTER句を指定すると、指定したクラスタの一部としてマテリアライズド・ビューを作成できます。クラスタ化マテリアライズド・ビューは、クラスタの領域割当てを使用します。したがって、CLUSTER句で物理属性またはTABLESPACE句を指定しないでください。

クラスタ化マテリアライズド・ビューの制限事項

CLUSTERを指定すると、materialized_view_propsにtable_partitioning_clausesを指定できません。

materialized_view_props

これらのプロパティ句を使用すると、既存の表に基づかないマテリアライズド・ビューを定義できます。既存の表に基づくマテリアライズド・ビューを作成するには、ON PREBUILT TABLE句を使用します。

column_properties

column_properties句を使用すると、LOB、ネストした表、VARRAYまたはXMLTypeの列の記憶特性を指定できます。object_type_col_propertiesは、マテリアライズド・ビューには関連しません。

関連項目:

この句のパラメータの指定の詳細は、「CREATE TABLE」を参照してください。

table_partitioning_clauses

table_partitioning_clausesを使用すると、マテリアライズド・ビューを、指定した範囲の値またはハッシュ・ファンクションでパーティション化できます。マテリアライズド・ビューのパーティション化は、表のパーティション化と同じです。

関連項目:

「CREATE TABLE」の「table_partitioning_clauses」 を参照してください。

CACHE | NOCACHE

アクセス頻度の高いデータについて、CACHEは、全表スキャンの実行時にこの表に対して取り出された各ブロックを、バッファ・キャッシュの最低使用頻度(LRU)リストの最高使用頻度側に入れることを指定します。この属性は、小規模な参照表で有効です。NOCACHEは、ブロックをLRUリストの最低使用頻度側に入れることを指定します。

ノート:

NOCACHE は、storage_clause に KEEP を指定したマテリアライズド・ビューには、影響しません。

関連項目:

CACHEまたはNOCACHEの指定の詳細は、「CREATE TABLE」を参照してください。

parallel_clause

parallel_clauseを使用すると、マテリアライズド・ビューへのパラレル操作をサポートするかどうかを指定できます。作成後にマテリアライズド・ビューに対する問合せおよびDMLのデフォルトの並列度を設定します。

この句の詳細は、「CREATE TABLE」の「parallel_clause」を参照してください。

build_clause

build_clauseを指定すると、マテリアライズド・ビューをいつ移入するかを指定できます。

IMMEDIATE

IMMEDIATEを指定すると、マテリアライズド・ビューにすぐに移入できます。これはデフォルトです。

DEFERRED

DEFERREDを指定すると、次のREFRESH操作でマテリアライズド・ビューに移入できます。最初の(遅延)リフレッシュは、常に、完全リフレッシュである必要があります。それ以前のマテリアライズド・ビューの値はUNUSABLEであるため、クエリー・リライトには使用できません。

USING INDEX句

USING INDEX句を使用すると、マテリアライズド・ビューのデータをメンテナンスするために使用されるデフォルトの索引のINITRANSパラメータおよびSTORAGEパラメータの値を変更できます。USING INDEXが設定されていない場合、この索引にはデフォルト値が使用されます。デフォルトの索引は、マテリアライズド・ビューの増分リフレッシュ(FAST)を高速に処理するために使用されます。

USING INDEX句の制限事項

この句には、PCTUSEDパラメータを指定できません。

USING NO INDEX句

USING NO INDEX句を指定すると、デフォルトの索引の作成を抑制できます。CREATE INDEX文を使用することによって、代替する索引を明示的に作成できます。USING NO INDEXを指定し、高速リフレッシュ(REFRESH FAST)でマテリアライズド・ビューを作成する場合は、このような索引を作成する必要があります。

create_mv_refresh

create_mv_refresh句を使用すると、マテリアライズド・ビューのデフォルトのリフレッシュ方法、リフレッシュ・モードおよびリフレッシュ時刻を指定できます。マテリアライズド・ビューのマスター表が変更された場合、マテリアライズド・ビューのデータを更新し、その時点でマスター表にあるデータを正確に反映させる必要があります。この句によって、自動的にマテリアライズド・ビューをリフレッシュする日時をスケジューリングし、リフレッシュの方法およびモードを指定できます。

同期リフレッシュの制限事項

同期リフレッシュの方法を使用する場合は、ON DEMANDおよびUSING TRUSTED CONSTRAINTSを指定する必要があります。

ノート:

この句では、デフォルトのリフレッシュ・オプションのみを設定します。リフレッシュを実際に行う手順は、『Oracle Database 管理者ガイド』および『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

関連項目:

- 「マテリアライズド・ビューの定期的リフレッシュ: 例」および「マテリアライズド・ビューの自動リフレッシュ時刻: 例」を参照してください。
- リフレッシュ方法の詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- リフレッシュ統計を使用してマテリアライズド・ビューのリフレッシュ操作のパフォーマンスを監視する方法を学習するには、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

FAST句

FASTを指定すると、高速リフレッシュ方法が使用されます。この方法では、マスター表に対して行われた変更に従ってリフレッシュを実行します。従来型DML変更の場合、変更は、マスター表に関連付けられたマテリアライズド・ビュー・ログに格納されます。ダイレクト・パス・インサート操作の変更は、ダイレクト・ローダー・ログに格納されます。

REFRESH FASTを指定すると、マテリアライズド・ビューのマスター表のマテリアライズド・ビュー・ログが存在していない場合に、CREATE文は正常に実行されません。ダイレクト・パス・インサートが行われると、ダイレクト・ローダー・ログが自動的に作成されます。手動での操作は必要ありません。

従来型DMLの変更の場合も、ダイレクト・パス・インサート操作の場合も、他の条件によって、高速リフレッシュへのマテリアライズド・ビューの適応性が制限されることがあります。

FASTリフレッシュの制限事項

FASTリフレッシュには、次の制限事項があります。

- 作成時にFASTリフレッシュを指定した場合、作成するマテリアライズド・ビューは高速リフレッシュに適応することが検証されています。ALTER MATERIALIZED VIEW文でリフレッシュ方法をFASTに変更した場合、これは検証されていま

せん。マテリアライズド・ビューが高速リフレッシュに適応しない場合、このビューをリフレッシュしようとするとエラーが戻されます。

- 定義する問合せに分析ファンクションまたはXMLTableファンクションが含まれている場合、マテリアライズド・ビューは高速リフレッシュに適応しません。
- 定義する問合せで、XMLIndex索引が定義されている表を参照する場合、マテリアライズド・ビューは高速リフレッシュに適格となりません。
- 暗号化されている列がある場合は、マテリアライズド・ビューを高速リフレッシュできません。

関連項目:

- レプリケーション環境における高速リフレッシュの制限事項は、『Oracle Database管理者ガイド』を参照してください。
- データ・ウェアハウス環境における高速リフレッシュの制限については、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- 高速リフレッシュに関する問題の診断については、DBMS_MVIEWパッケージのEXPLAIN_MVIEWプロシージャを参照し、クエリー・リライトの問題の修正については、DBMS_MVIEWパッケージのTUNE_MVIEWプロシージャを参照してください。
- 「分析ファンクション」
- 高速リフレッシュ可能なマテリアライズド・ビューの作成: 例

COMPLETE句

COMPLETEを指定すると、完全リフレッシュ方法が使用されます。この方法は、マテリアライズド・ビューを定義する問合せを実行することによって実装されます。完全リフレッシュを要求すると、高速リフレッシュが実行可能であっても、完全リフレッシュが実行されます。

FORCE句

FORCEを指定すると、リフレッシュ時に、高速リフレッシュが可能な場合は高速リフレッシュを実行し、可能でない場合は完全リフレッシュを実行するように指定できます。リフレッシュ方法(FAST、COMPLETEまたはFORCE)を指定しないと、デフォルトでFORCEが指定されます。

ON COMMIT句

ON COMMITを指定すると、マテリアライズド・ビューのマスター表に対するトランザクションをコミットするときに、必ずリフレッシュが実行されるように指定できます。この句を指定すると、リフレッシュ操作がコミット処理の一部として行われるため、コミットの完了に時間がかかります。

指定できるのは、ON COMMIT、ON DEMANDおよびON STATEMENTのいずれか1つの句のみです。ON COMMITを指定した場合、START WITHまたはNEXTを指定できません。

ON COMMITのリフレッシュの制限事項:

ON COMMIT句には、次の制限事項があります。

- この句は、オブジェクト型またはOracleが提供する型を含むマテリアライズド・ビューについてはサポートしていません。
- この句はリモート表を使用するマテリアライズド・ビューではサポートされていません。

- この句を指定すると、それ以降は、このマテリアライズド・ビューのすべてのマスター表で分散トランザクションが実行できなくなります。たとえば、リモート表から選択してマスターに挿入することはできません。ON DEMAND句の場合は、それ以降のマスター表での分散トランザクションに、このような制限はありません。

ON DEMAND句

ON DEMANDを指定すると、3つのDBMS_MVIEWリフレッシュ・プロシージャのうちのいずれかを使用してユーザーが手動でリフレッシュしないかぎりデータベースによってマテリアライズド・ビューがリフレッシュされないように指定できます。

指定できるのは、ON COMMIT、ON DEMANDおよびON STATEMENTのいずれか1つの句のみです。これら3つの句をすべて省略すると、ON DEMANDがデフォルトになります。このデフォルト設定は、同じCREATE MATERIALIZED VIEW文または後続のALTER MATERIALIZED VIEW文のいずれかにSTART WITH句またはNEXT句を指定することで上書きできます。

START WITHおよびNEXTは、ON DEMANDより優先されます。このため、START WITHまたはNEXTを指定したときは、ほとんどの場合、ON DEMANDを指定しても意味がありません。

関連項目:

- これらのプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- REFRESH ON DEMANDを指定することによって作成できるマテリアライズド・ビューのタイプについては、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

ON STATEMENT句

ON STATEMENTを指定すると、マテリアライズド・ビューの実表のいずれかでDML操作が実行されるたびに自動リフレッシュが発生します。

指定できるのは、ON COMMIT、ON DEMANDおよびON STATEMENTのいずれか1つの句のみです。ON STATEMENTは、マテリアライズド・ビューを作成する場合にのみ指定できます。ON STATEMENTリフレッシュを使用するマテリアライズド・ビューは後から変更できません。

ON STATEMENTのリフレッシュの制限事項

ON STATEMENT句には、次の制限事項があります。

- この句は、高速リフレッシュが可能なマテリアライズド・ビューでのみ使用できます。ON STATEMENT句は、REFRESH FAST句とともに指定する必要があります。
- マテリアライズド・ビューを定義する問合せで参照する実表は、スター・スキーマまたはスノーフレーク・スキーマ・モデルを使用する図形結合で接続する必要があります。問合せには、1つのみの一元化されたファクト表および1つ以上のディメンション表を含み、関連する結合表のすべてのペアは主キーと外部キーの制約を使用する必要があります。
    - スノーフレーク・モデルの深さに制限はありません。
    - RELYモードにはこの制約があります。ただし、RELY制約を使用するマテリアライズド・ビューの作成中に、USING TRUSTED CONSTRAINT句を含める必要があります。
- マテリアライズド・ビューを定義する問合せには、ファクト表のROWID列を含める必要があります。
- マテリアライズド・ビューを定義する問合せには、非表示列、ANSI結合構文、複合問合せ、実表としてのインライン・ビュー、複合主キー、LONG列およびLOB列のいずれも含めることはできません。

- ON STATEMENTリフレッシュ・モードを使用する既存のマテリアライズド・ビューの定義は変更できません。
- 既存のマテリアライズド・ビューは変更できないため、ON STATEMENTリフレッシュを有効にできません。
- 次の操作により、ON STATEMENTリフレッシュが指定されたマテリアライズド・ビューは使用禁止となります。
  - マテリアライズド・ビューのベースとなっている1つ以上のディメンション表に対するUPDATE操作
  - すべての実表に対するパーティション・メンテナンス操作およびTRUNCATE操作

    ただし、ON STATEMENTリフレッシュ・モードが指定されたマテリアライズド・ビューはパーティション化できます。
- ON COMMIT句に適用されるすべての制限はON STATEMENTに適用されます。

START WITH句

最初の自動リフレッシュ時刻を表す日時式を指定します。

NEXT句

自動リフレッシュの間隔を計算するための日時式を指定します。

START WITH値およびNEXT値は、将来の時刻に評価される値です。START WITH値を省略した場合、Oracle DatabaseはマテリアライズドビューのNEXT式を評価することによって、最初の自動リフレッシュ時刻を判断します。START WITH値を指定し、NEXT値を指定しない場合、Oracle Databaseは1回のみマテリアライズド・ビューをリフレッシュします。START WITH値およびNEXT値のどちらも指定しない場合、またはcreate_mv_refreshを指定しない場合は、マテリアライズド・ビューは自動リフレッシュされません。

WITH PRIMARY KEY句

WITH PRIMARY KEYを指定すると、主キー・マテリアライズド・ビューを作成できます。これはデフォルトであり、WITH ROWIDの項で説明する場合を除き、すべての場合に使用する必要があります。主キー・マテリアライズド・ビューを使用すると、高速リフレッシュに対するマテリアライズド・ビューの適応性に影響せずに、マテリアライズド・ビューのマスター表を再編成できます。マスター表には、使用可能な主キー制約が定義されている必要があり、マテリアライズド・ビューを定義する問合せでは、すべての主キー列が直接指定される必要があります。定義する問合せでは、UPPERなどのファンクションへの引数として主キー列を指定できません。

主キー・マテリアライズド・ビューの制限事項

この句は、オブジェクト・マテリアライズド・ビューには指定できません。WITH OBJECT IDを指定してマテリアライズドされたオブジェクトは暗黙的にリフレッシュされます。

関連項目:

主キー・マテリアライズド・ビューの詳細は、『Oracle Database管理者ガイド』および「主キー・マテリアライズド・ビューの作成: 例」を参照してください。

WITH ROWID句

WITH ROWIDを指定すると、ROWIDマテリアライズド・ビューを作成できます。マテリアライズド・ビューがマスター表の主キー列をすべて含まない場合に、ROWIDマテリアライズド・ビューは有効です。ROWIDマテリアライズド・ビューは、単一表を基にしている必要があり、次のいずれも含むことができません。

- distinctまたは集計ファンクション
- GROUP BY句または CONNECT BY句

- 副問合せ
- 結合
- 集合演算

WITH ROWID句は、定義する問合せに複数のマスター表がある場合は効果がありません。

完全リフレッシュが行われるまでは、マスター表の再編成後に、ROWIDマテリアライズド・ビューは高速リフレッシュされません。

ROWIDマテリアライズド・ビューの制限事項

この句は、オブジェクト・マテリアライズド・ビューには指定できません。WITH OBJECT IDを指定してマテリアライズドされたオブジェクトは暗黙的にリフレッシュされます。

関連項目:

「マテリアライズド集計ビューの作成: 例」および「ROWIDマテリアライズド・ビューの作成: 例」を参照してください。

USING ROLLBACK SEGMENT句

自動UNDOモードではロールバック・セグメントではなくUNDO表領域が使用されるため、データベースが自動UNDOモードの場合、この句は無効です。自動UNDOモードを使用することをお薦めします。この句は、ロールバック・セグメントが使用される以前のバージョンのOracle Databaseが含まれるレプリケーション環境との下位互換性のためにサポートされています。

rollback_segmentに、マテリアライズド・ビューのリフレッシュ中に使用するリモート・ロールバック・セグメントを指定します。

DEFAULT

DEFAULTを使用すると、使用するロールバック・セグメントを自動的に選択できます。DEFAULTを指定した場合、rollback_segmentは指定できません。DEFAULTは、マテリアライズド・ビューを(作成ではなく)変更する場合に有効です。

関連項目:

ALTER MATERIALIZED VIEW

MASTER

MASTERを使用すると、個々のマテリアライズド・ビュー用のリモート・マスター・サイトで使用されるリモート・ロールバック・セグメントを指定できます。

LOCAL

LOCALを使用すると、マテリアライズド・ビューが含まれているローカル・リフレッシュ・グループで使用されるリモート・ロールバック・セグメントを指定できます。これはデフォルトです。

関連項目:

DBMS_REFRESHパッケージを使用するローカル・マテリアライズド・ビューのロールバック・セグメントの指定の詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

rollback_segmentを指定しない場合、使用するロールバック・セグメントが自動的に選択されます。各マテリアライズド・ビューに対して1つのマスター・ロールバック・セグメントが格納され、マテリアライズド・ビューの作成およびリフレッシュ時に検証され

ます。複合マテリアライズド・ビューの場合、マスター・ロールバック・セグメントの指定は無視されます。

USING ... CONSTRAINTS句

USING ... CONSTRAINTS句を使用すると、リフレッシュ操作中にOracle Databaseでより多くのリライト・オプションを選択でき、リフレッシュをより効果的に実行できます。リフレッシュ操作中に、適用される制約のみに依存するのではなく、適用されていない制約(RELY状態のディメンションの関係や制約など)を使用できます。

USING TRUSTED CONSTRAINTS句では、NULL以外の仮想プライベート・データベース(VPD)ポリシーを持つ表の最上位にマテリアライズド・ビューを作成できます。この場合、マテリアライズド・ビューが適切に動作することを確認する必要があります。マテリアライズド・ビューの結果は、VPDポリシーによってフィルタ処理された行と列に基づいて計算されます。したがって、正しい結果を得るには、マテリアライズド・ビュー定義をVPDポリシーで調整する必要があります。USING TRUSTED CONSTRAINTS句を指定しないと、マスター表のVPDポリシーにより、マテリアライズド・ビューを作成できなくなります。

ノート:

USING TRUSTED CONSTRAINTS 句を指定すると、データベース管理者が信頼できると宣言したが、データベースで検証されていないディメンションおよび制約の情報を、Oracle Database で使用できます。ディメンションおよび制約の情報が有効であると、パフォーマンスを向上できる場合があります。ただし、この情報が無効であると、正常な状態が戻されても、リフレッシュ・プロシージャによってマテリアライズド・ビューが破損する場合があります。

この句を指定しない場合、USING ENFORCED CONSTRAINTSがデフォルトになります。

NEVER REFRESH句

NEVER REFRESHを指定すると、Oracle Databaseのリフレッシュ・メカニズムまたはパッケージ・プロシージャを使用したマテリアライズド・ビューのリフレッシュを回避できます。このようなプロシージャから発行された、マテリアライズド・ビューに対するREFRESH文は、無視されます。この句を指定すると、マテリアライズド・ビューでDML操作を実行できます。この句を無効にするには、ALTER MATERIALIZED VIEW ... REFRESH文を発行する必要があります。

evaluation_edition_clause

subqueryでエディショニングされたオブジェクトを参照する場合は、この句を指定する必要があります。この句を使用すると、エディショニングされたオブジェクトの名前解決時に検索されたエディション(評価エディション)を指定できます。

- CURRENT EDITIONを指定すると、このDDL文が実行されるエディションを検索できます。
- EDITION editionを指定すると、editionを検索できます。
- NULL EDITIONを指定することは、evaluation_edition_clauseを省略することと同じです。

evaluation_edition_clauseを省略すると、エディショニングされたオブジェクトは名前解決時に認識されなくなるためエラーが発生します。評価エディションを削除すると、マテリアライズド・ビューが無効になります。

関連項目:

マテリアライズド・ビューの評価エディションを指定する方法の詳細は、『Oracle Database開発ガイド』を参照してください。

{ ENABLE | DISABLE } ON QUERY COMPUTATION

この句を使用すると、リアルタイムのマテリアライズド・ビューまたは通常のビューを作成できます。リアルタイムのマテリアライズド・ビューでは、データ変更によりマテリアライズド・ビューがその実表と同期していない場合も、ユーザーの問合せに対して最新の

データを提供します。マテリアライズド・ビューを変更するかわりに、オプティマイザはマテリアライズド・ビューの既存の行とログ・ファイル(マテリアライズド・ビュー・ログまたはダイレクト・ローダー・ログ)に記録された変更を結合する問合せを記述します。これは、問合せ時計算と呼ばれます。

- ENABLE ON QUERY COMPUTATIONを指定し、問合せ時計算を有効にすると、リアルタイムのマテリアライズド・ビューを作成できます。これにより、SELECT文でFRESH_MVヒントを指定して、マテリアライズド・ビューから直接最新データを問い合せることができます。マテリアライズド・ビューがクエリー・リライトに対しても有効化されている場合、クエリー・リライト中に自動的に問合せ時計算が行われます。
- DISABLE ON QUERY COMPUTATIONを指定し、問合せ時計算を無効にすると、通常のマテリアライズド・ビューを作成できます。これはデフォルトです。

リアルタイムのマテリアライズド・ビューの制限事項

リアルタイムのマテリアライズド・ビューには、次の制限事項があります。

- リアルタイムのマテリアライズド・ビューは、実表に対して作成された1つ以上のマテリアライズド・ビュー・ログが使用禁止になっているか、存在しない場合は、使用できません。
- リアルタイムのマテリアライズド・ビューは、ホーム外のリフレッシュ、ログベースのリフレッシュまたはパーティション・チェンジ・トラッキング(PCT)リフレッシュを使用してリフレッシュできる必要があります。
- REFRESH ON COMMITモードのマテリアライズド・ビューをリアルタイムのマテリアライズド・ビューにすることはできません。
- リアルタイムのマテリアライズド・ビューが、1つ以上のベース・マテリアライズド・ビューの上位に定義されているネステッド・マテリアライズド・ビューである場合、すべてのベース・マテリアライズド・ビューが最新である場合にのみ、クエリー・リライトが発生します。1つ以上のベース・マテリアライズド・ビューが失効している場合、クエリー・リライトは、このリアルタイムのマテリアライズド・ビューを使用して実行されません。
- リアルタイムのマテリアライズド・ビューに直接アクセスする問合せのカーソルは共有されません。

関連項目:

- FRESH_MVヒント
- リアルタイムのマテリアライズド・ビューの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

query_rewrite_clause

query_rewrite_clauseを使用すると、マテリアライズド・ビューがクエリー・リライトに適格であるかどうかを指定できます。

ENABLE句

ENABLEを指定すると、クエリー・リライトでマテリアライズド・ビューを使用可能にできます。unusable_editions_clauseを同時に指定すると、使用禁止エディションでマテリアライズド・ビューがクエリー・リライトに使用されなくなります。

クエリー・リライトの有効化の制限事項

クエリー・リライトの有効化には、次の制限事項があります。

- マテリアライズド・ビューのすべてのユーザー定義ファンクションがDETERMINISTICである場合のみ、クエリー・リライトを使用可能にできます。
- 文内の式が反復可能な場合のみ、クエリー・リライトを使用可能にできます。たとえば、CURRENT_TIME、USER、順

序値(たとえば、CURRVALまたはNEXTVAL疑似列)、またはSAMPLE句(マテリアライズド・ビューの変更内容と異なる行をサンプルとして抽出します)を含めることはできません。

ノート:

- クエリー・リライトでのマテリアライズド・ビューの使用は、デフォルトでは無効です。この句を指定して、マテリアライズド・ビューをクエリー・リライトに適用できるようにする必要があります。
- マテリアライズド・ビューを作成した後は、DBMS_STATS パッケージを使用して、その統計情報を収集する必要があります。クエリー・リライトを最適化するために、このパッケージで生成した統計情報が必要です。

関連項目:

- クエリー・リライトの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- リフレッシュ統計を使用してマテリアライズド・ビューのリフレッシュ操作のパフォーマンスを監視する方法を学習するには、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- DBMS_STATSパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- クエリー・リライトに関する問題の診断については、DBMS_MVIEWパッケージのEXPLAIN_MVIEWプロシージャを参照し、クエリー・リライトの問題の修正については、DBMS_MVIEWパッケージのTUNE_MVIEWプロシージャを参照してください
- CREATE FUNCTION

DISABLE句

DISABLEを指定すると、マテリアライズド・ビューはクエリー・リライトに適格でなくなります。使用禁止にしたマテリアライズド・ビューはリフレッシュ可能です。

unusable_editions_clause

この句を使用すると、1つ以上のエディションでマテリアライズド・ビューがクエリー・リライトの対象でないことを指定できます。この句は、ENABLE句またはDISABLE句を指定しているかどうかにかかわらず指定できます。DISABLE句を指定すると、ALTER MATERIALIZED VIEW ... ENABLE QUERY REWRITE文を使用して、マテリアライズド・ビューが後にクエリー・リライトに使用できるようにされたときに、この句の効果が現れるようになります。

UNUSABLE BEFORE句

この句を使用すると、先行エディションでマテリアライズド・ビューがクエリー・リライトに適格でないことを指定できます。

- CURRENT EDITIONを指定すると、現在のエディションの先行エディションでマテリアライズド・ビューがクエリー・リライトに適格でなくなります。
- EDITION editionを指定すると、指定したeditionの先行エディションでマテリアライズド・ビューが適格でなくなります。

UNUSABLE BEGINNING WITH句

この句を使用すると、特定のエディションとそれ以降のエディションでマテリアライズド・ビューがクエリー・リライトに適格でなくなること

を指定できます。

- CURRENT EDITIONを指定すると、現在のエディションとそれ以降のエディションで、マテリアライズド・ビューがクエリー・リライトに適格ではなくなります。
- EDITION editionを指定すると、指定したエディションとそれ以降のエディションで、マテリアライズド・ビューがクエリー・リライトに適格ではなくなります。
- NULL EDITIONを指定することは、UNUSABLE BEGINNING WITH句を省略することと同じです。

マテリアライズド・ビューは、クエリー・リライトに適格ではない各エディションに対する依存関係があります。これに該当するエディションが後に削除されると、依存関係が削除されます。ただし、そのマテリアライズド・ビューが無効にされることはありません。

AS副問合せ

マテリアライズド・ビューを定義する問合せを指定します。マテリアライズド・ビューの作成時に、この副問合せが実行され、実行結果がマテリアライズド・ビューに格納されます。この副問合せは、有効なSQL副問合せである必要があります。ただし、すべての副問合せに高速リフレッシュを適用できるわけではなく、すべての副問合せがクエリー・リライトに使用できるわけでもありません。

マテリアライズド・ビューを定義する問合せのノート

マテリアライズド・ビューには、次のノートがあります。

- BUILD DEFERREDを指定する場合、定義する問合せはすぐには実行されません。
- マテリアライズド・ビューを定義する問合せのFROM句に指定する各表およびビューを、それを含むスキーマで修飾することをお薦めします。
- 仮想プライベート・データベース(VPD)ポリシーを持つマスター表から定義問合せが選択するマテリアライズド・ビューを作成するには、REFRESH USING TRUSTED CONSTRAINTS句を指定する必要があります。

マテリアライズド・ビューを定義する問合せの制限事項

マテリアライズド・ビュー問合せには、次の制限事項があります。

- マテリアライズド・ビューを定義する問合せでは、ユーザーSYSが所有する表、ビューまたはマテリアライズド・ビューから選択できますが、このようなマテリアライズド・ビューに対してQUERY REWRITEは使用できません。
- マテリアライズド・ビューを定義する問合せは、V$ビューまたはGV$ビューから選択できません。
- マテリアライズド・ビューを定義するとき、定義する問合せのSELECT構文のリストに副問合せを使用することはできません。WHERE句など、定義問合せのその他の場所に副問合せを記述することは可能です。
- マテリアライズド・ビューを定義する問合せでは、flashback_query_clauseのAS OF句を使用することはできません。
- GROUP BY句を含むマテリアライズド結合ビューおよびマテリアライズド集計ビューは、索引構成表からは選択できません。
- マテリアライズド・ビューにLONGまたはLONG RAWのデータ型の列を含めることはできません。
- マテリアライズド・ビューに仮想列を含めることはできません。
- 一時表に対しては、マテリアライズド・ビュー・ログを作成できません。そのため、定義する問合せが一時表を参照する場合、マテリアライズド・ビューは高速リフレッシュ(FAST)に適応せず、この文でQUERY REWRITEを指定することもできません。
- 定義する問合せのFROM句が他のマテリアライズド・ビューを参照する場合、この文で作成するマテリアライズド・ビューの

リフレッシュの前に、定義する問合せで参照されているマテリアライズド・ビューを常にリフレッシュしておく必要があります。

- 定義する問合せ内に結合式を持つマテリアライズド・ビューは、XMLデータ型の列を持つことができません。XMLデータ型には、XMLTypeおよびURIデータ型の列が含まれます。

クエリー・リライトが可能なマテリアライズド・ビューを作成する場合、次の制限があります。

- 定義する問合せには、ROWNUM、USER、SYSDATE、リモート表、順序、または(データベースやパッケージ状態の書込みまたは読取りを行う)PL/SQLファンクションに対する直接またはビューを介した参照を含めることはできません。
- マテリアライズド・ビューおよびマテリアライズド・ビューのマスター表はリモートにできません。

マテリアライズド・ビュー・ログを使用した高速リフレッシュ、またはステージング・ログを使用した同期リフレッシュにマテリアライズド・ビューが適格になるようにする場合は、いくつかの追加の制限事項が適用されます。

関連項目:

- 高速リフレッシュおよび同期リフレッシュに関連する制限事項については、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- レプリケーションに関連する制限の詳細は、『Oracle Database管理者ガイド』を参照してください。
- 「マテリアライズド結合ビューの作成: 例」、「副問合せマテリアライズド・ビューの作成: 例」および「ネステッド・マテリアライズド・ビューの作成: 例」を参照してください。

例

次の例では、「CREATE MATERIALIZED VIEW LOG」の「例」で作成したマテリアライズド・ログが必要です。

単純なマテリアライズド・ビューの作成: 例

次の文は、hrスキーマのemployeesおよび表に基づいた非常に単純なマテリアライズド・ビューを作成します。

```
CREATE MATERIALIZED VIEW mv1 AS SELECT * FROM hr.employees;
```

デフォルトでは、REFRESH ON DEMANDのみが指定されている主キー・マテリアライズド・ビューが作成されます。マテリアライズド・ビュー・ログがemployeesに存在する場合は、高速リフレッシュできるようにmv1を変更できます。このようなログが存在しない場合は、mv1の完全リフレッシュのみが可能です。Oracle Databaseでは、mv1に対してデフォルトの記憶域プロパティが使用されます。この操作に必要な権限は、CREATE MATERIALIZED VIEWシステム権限と、hr.employeesに対するREADまたはSELECTオブジェクト権限のみです。

副問合せマテリアライズド・ビューの作成: 例

次の文は、remoteデータベースのshスキーマのcustomersおよびcountries表に基づいた副問合せマテリアライズド・ビューを作成します。

```
CREATE MATERIALIZED VIEW foreign_customers
   AS SELECT * FROM sh.customers@remote cu
   WHERE EXISTS
     (SELECT * FROM sh.countries@remote co
      WHERE co.country_id = cu.country_id);
```

マテリアライズド集計ビューの作成: 例

次の文は、サンプル表sh.salesにマテリアライズド集計ビューを作成して移入し、デフォルトのリフレッシュ方法、モードおよび時刻を指定します。これは、「高速リフレッシュ用のマテリアライズド・ビュー・ログの作成: 例」で作成したマテリアライズド・ビュー・ロ

グと、ここに示す2つの追加ログを使用します。

```
CREATE MATERIALIZED VIEW LOG ON times
   WITH ROWID, SEQUENCE (time_id, calendar_year)
   INCLUDING NEW VALUES;
CREATE MATERIALIZED VIEW LOG ON products
   WITH ROWID, SEQUENCE (prod_id)
   INCLUDING NEW VALUES;
CREATE MATERIALIZED VIEW sales_mv
   BUILD IMMEDIATE
   REFRESH FAST ON COMMIT
   AS SELECT t.calendar_year, p.prod_id,
      SUM(s.amount_sold) AS sum_sales
      FROM times t, products p, sales s
      WHERE t.time_id = s.time_id AND p.prod_id = s.prod_id
      GROUP BY t.calendar_year, p.prod_id;
```

マテリアライズド結合ビューの作成: 例

次の文は、サンプル・スキーマshの表を使用して、マテリアライズド集計ビューsales_by_month_by_stateを作成して移入します。マテリアライズド・ビューは、この文が正しく実行されるとすぐに移入されます。デフォルトで、次のマテリアライズド・ビューを定義する問合せを再実行することによって、後続のリフレッシュが行われます。

```
CREATE MATERIALIZED VIEW sales_by_month_by_state
     TABLESPACE example
     PARALLEL 4
     BUILD IMMEDIATE
     REFRESH COMPLETE
     ENABLE QUERY REWRITE
     AS SELECT t.calendar_month_desc, c.cust_state_province,
        SUM(s.amount_sold) AS sum_sales
        FROM times t, sales s, customers c
        WHERE s.time_id = t.time_id AND s.cust_id = c.cust_id
        GROUP BY t.calendar_month_desc, c.cust_state_province;
```

事前作成したマテリアライズド・ビューの作成: 例

次の文は、既存の集計表sales_sum_tableに対するマテリアライズド集計ビューを作成します。

```
CREATE TABLE sales_sum_table
   (month VARCHAR2(8), state VARCHAR2(40), sales NUMBER(10,2));
CREATE MATERIALIZED VIEW sales_sum_table
   ON PREBUILT TABLE WITH REDUCED PRECISION
   ENABLE QUERY REWRITE
   AS SELECT t.calendar_month_desc AS month,
             c.cust_state_province AS state,
             SUM(s.amount_sold) AS sales
      FROM times t, customers c, sales s
      WHERE s.time_id = t.time_id AND s.cust_id = c.cust_id
      GROUP BY t.calendar_month_desc, c.cust_state_province;
```

前述の例では、マテリアライズド・ビューは事前作成表と同じ名前を持ち、かつ事前作成表と同じデータ型で同じ数の列を持ちます。WITH REDUCED PRECISION句によって、マテリアライズド・ビュー列の精度と副問合せによって戻された値の精度の違いは許可されます。

主キー・マテリアライズド・ビューの作成: 例

次の文は、サンプル表oe.product_informationに主キー・マテリアライズド・ビューcatalogを作成します。

```
CREATE MATERIALIZED VIEW catalog
   REFRESH FAST START WITH SYSDATE NEXT SYSDATE + 1/4096
   WITH PRIMARY KEY
   AS SELECT * FROM product_information;
```

ROWIDマテリアライズド・ビューの作成: 例

次の文は、サンプル表oe.ordersにROWIDマテリアライズド・ビューを作成します。

```
CREATE MATERIALIZED VIEW order_data REFRESH WITH ROWID
   AS SELECT * FROM orders;
```

マテリアライズド・ビューの定期的リフレッシュ: 例

次の文は、主キー・マテリアライズド・ビューemp_dataを作成し、サンプル表hr.employeesのデータを移入します。

```
CREATE MATERIALIZED VIEW LOG ON employees
   WITH PRIMARY KEY
   INCLUDING NEW VALUES;
CREATE MATERIALIZED VIEW emp_data
   PCTFREE 5 PCTUSED 60
   TABLESPACE example
   STORAGE (INITIAL 50K)
   REFRESH FAST NEXT sysdate + 7
   AS SELECT * FROM employees;
```

前述の文にはSTART WITHパラメータが指定されていないため、Oracle Databaseでは、現行のSYSDATEを使用してNEXT値が評価され、最初の自動リフレッシュ時刻が判断されます。従業員表のマテリアライズド・ビュー・ログが作成されたため、マテリアライズド・ビュー作成の7日後から、7日ごとにマテリアライズド・ビューの高速リフレッシュが実行されます。

マテリアライズド・ビューが高速リフレッシュを行うための条件と一致するため、高速リフレッシュが行われます。また、前述の文では、マテリアライズド・ビューをメンテナンスするためにデータベースで使用される表の記憶特性も設定しています。

マテリアライズド・ビューの自動リフレッシュ時刻: 例

次の文は、remoteおよびlocalデータベースの従業員表を問い合せる複合マテリアライズド・ビューall_customersを作成します。

```
CREATE MATERIALIZED VIEW all_customers
   PCTFREE 5 PCTUSED 60
   TABLESPACE example
   STORAGE (INITIAL 50K)
   USING INDEX STORAGE (INITIAL 25K)
   REFRESH START WITH ROUND(SYSDATE + 1) + 11/24
   NEXT NEXT_DAY(TRUNC(SYSDATE), 'MONDAY') + 15/24
   AS SELECT * FROM sh.customers@remote
         UNION
      SELECT * FROM sh.customers@local;
```

このマテリアライズド・ビューは、翌日の午前11:00に自動的にリフレッシュされ、その後は、毎週月曜日の午後3:00にリフレッシュされます。デフォルトのリフレッシュ方法は、FORCEです。定義する問合せにはUNION演算子が含まれますが、この演算子は高速リフレッシュでサポートされていないため、自動的に完全リフレッシュが実行されます。

前述の文では、マテリアライズド・ビューおよびこのマテリアライズド・ビューをメンテナンスするために使用される索引の記憶特性を次のように設定しています。

- 最初のSTORAGE句で、マテリアライズド・ビューの1番目と2番目のエクステントのサイズをそれぞれ50KBに設定します。
- 次のUSING INDEX句付きのSTORAGE句では、索引の1番目と2番目のエクステントのサイズをそれぞれ25KBに変更します。

高速リフレッシュ可能なマテリアライズド・ビューの作成: 例

次の文は、product_informationおよびinventories表からWHERE条件で戻される行を制限するUNION集合演

算子を使用して、サンプル・スキーマoeのorder_items表から列を選択する高速リフレッシュが可能なマテリアライズド・ビューを作成します。order_itemsおよびproduct_informationのマテリアライズド・ビュー・ログは、CREATE MATERIALIZED VIEW LOGの「例」の項で作成されたものです。また、次の例では、oe.inventoriesのマテリアライズド・ビュー・ログが必要です。

```
CREATE MATERIALIZED VIEW LOG ON inventories
   WITH (quantity_on_hand);
CREATE MATERIALIZED VIEW warranty_orders REFRESH FAST AS
  SELECT order_id, line_item_id, product_id FROM order_items o
    WHERE EXISTS
    (SELECT * FROM inventories i WHERE o.product_id = i.product_id
      AND i.quantity_on_hand IS NOT NULL)
  UNION
    SELECT order_id, line_item_id, product_id FROM order_items
    WHERE quantity > 5;
```

マテリアライズド・ビューwarranty_ordersには、order_items(product_idを結合列とする)およびinventories(quantity_on_handをフィルタ列とする)で定義されたマテリアライズド・ビュー・ログが必要です。「マテリアライズド・ビュー・ログに対するフィルタ列の指定: 例」および「マテリアライズド・ビュー・ログに対する結合列の指定: 例」を参照してください。

ネステッド・マテリアライズド・ビューの作成: 例

次の例は、前述の例のマテリアライズド・ビューをマスター表として使用し、サンプル・スキーマoeの特定の販売員で構成されるマテリアライズド・ビューを作成します。

```
CREATE MATERIALIZED VIEW my_warranty_orders
   AS SELECT w.order_id, w.line_item_id, o.order_date
   FROM warranty_orders w, orders o
   WHERE o.order_id = o.order_id
   AND o.sales_rep_id = 165;
```

CREATEMATERIALIZEDVIEWLOG

目的

CREATE MATERIALIZED VIEW LOG文を使用すると、マテリアライズド・ビューのマスター表に関連する表マテリアライズド・ビュー・ログを作成できます。

ノート:

下位互換性を保つために、MATERIALIZED VIEW のかわりにキーワード SNAPSHOT もサポートされています。

マテリアライズド・ビュー・ログは、高速リフレッシュと同期リフレッシュという、2種類のマテリアライズド・ビュー・リフレッシュに使用されます。

高速リフレッシュには、従来型のマテリアライズド・ビュー・ログが使用されます。高速リフレッシュ(増分リフレッシュとも呼ばれます)の実行時、マスター表のデータにDML変更が加えられていると、Oracle Databaseは、その変更を記述した行をマテリアライズド・ビュー・ログに格納し、そのマテリアライズド・ビュー・ログを使用して、マスター表に基づくマテリアライズド・ビューをリフレッシュします。

同期リフレッシュには、ステージング・ログという特別なマテリアライズド・ビュー・ログが使用されます。同期リフレッシュの実行時、DMLの変更は、最初にステージング・ログに記述されてから、マスター表とマテリアライズド・ビューに同時に適用されます。これにより、マスター表のデータとマテリアライズド・ビューのデータが、リフレッシュ・プロセス全体で同期されていることを保証できます。このリフレッシュ方法は、データ・ウェアハウス環境で役立ちます。

マテリアライズド・ビュー・ログを使用しない場合、Oracle Databaseは、マテリアライズド・ビュー問合せを再実行してマテリアライズド・ビューをリフレッシュする必要があります。このプロセスが完全リフレッシュです。通常、完全リフレッシュを完了するまでの時間は、高速リフレッシュや同期リフレッシュよりも長くなります。

マテリアライズド・ビュー・ログは、マスター表と同一のスキーマ内のマスター・データベースにあります。マスター表に定義できるマテリアライズド・ビュー・ログは1つのみです。

マテリアライズド結合ビューを高速リフレッシュまたは同期リフレッシュする場合は、そのマテリアライズド・ビューが参照するマスター表ごとにマテリアライズド・ビュー・ログを作成する必要があります。

高速リフレッシュでは、タイムスタンプ・ベース・マテリアライズド・ビュー・ログと、コミットSCNベース・マテリアライズド・ビュー・ログという、2つのタイプのマテリアライズド・ビュー・ログをサポートします。タイムスタンプ・ベース・マテリアライズド・ビュー・ログは、タイムスタンプに基づきます。このログを使用する場合、マテリアライズド・ビューのリフレッシュを準備するためにいくつかの設定操作が必要です。コミットSCNベース・マテリアライズド・ビュー・ログは、タイムスタンプではなく、コミットSCNデータに基づきます。この場合、設定操作が必要なくなり、マテリアライズド・ビューのリフレッシュ速度を向上させることができます。COMMIT SCN句を指定すると、コミットSCNベース・マテリアライズド・ビュー・ログが作成されます。それ以外の場合は、タイムスタンプ・ベース・マテリアライズド・ビュー・ログが作成されます。COMMIT SCNを利用できるのは、新しく作成するマテリアライズド・ビュー・ログのみであることに注意してください。既存のマテリアライズド・ビュー・ログは、削除して再度作成しないかぎり、COMMIT SCNを追加するように変更できません。詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

同期リフレッシュは、タイムスタンプ・ベース・ステージング・ログのみをサポートします。

関連項目:

- マテリアライズド・ビューの概要は、「CREATE MATERIALIZED VIEW」、「ALTER MATERIALIZED VIEW」、『Oracle Database概要』、『Oracle Databaseデータ・ウェアハウス・ガイド』および『Oracle Database管理者ガイド』を参照してください。
- マテリアライズド・ビュー・ログの変更については、「ALTER MATERIALIZED VIEW LOG」を参照してください。
- マテリアライズド・ビュー・ログの削除については、「DROP MATERIALIZED VIEW LOG」を参照してください。
- ダイレクト・ローダー・ログの使用については、『Oracle Databaseユーティリティ』を参照してください。

前提条件

マテリアライズド・ビュー・ログを作成するために必要な権限は、マテリアライズド・ビュー・ログに関連付けられた基礎となるオブジェクトの作成に必要な権限と直接関連しています。

- マスター表を所有しているユーザーにCREATE TABLE権限がある場合、関連するマテリアライズド・ビュー・ログを作成できます。
- 他のユーザーのスキーマ内に表のマテリアライズド・ビュー・ログを作成する場合は、CREATE ANY TABLEシステム権限、COMMENT ANY TABLEシステム権限、およびマスター表に対するREADまたはSELECTオブジェクト権限またはREAD ANY TABLEまたはSELECT ANY TABLEシステム権限が必要です。

どちらの場合も、マテリアライズド・ビュー・ログの所有者には、マテリアライズド・ビュー・ログを格納するための表領域への十分な割当て制限またはUNLIMITED TABLESPACEシステム権限が必要です。

関連項目:

マテリアライズド・ビュー・ログを作成する場合の前提条件については、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

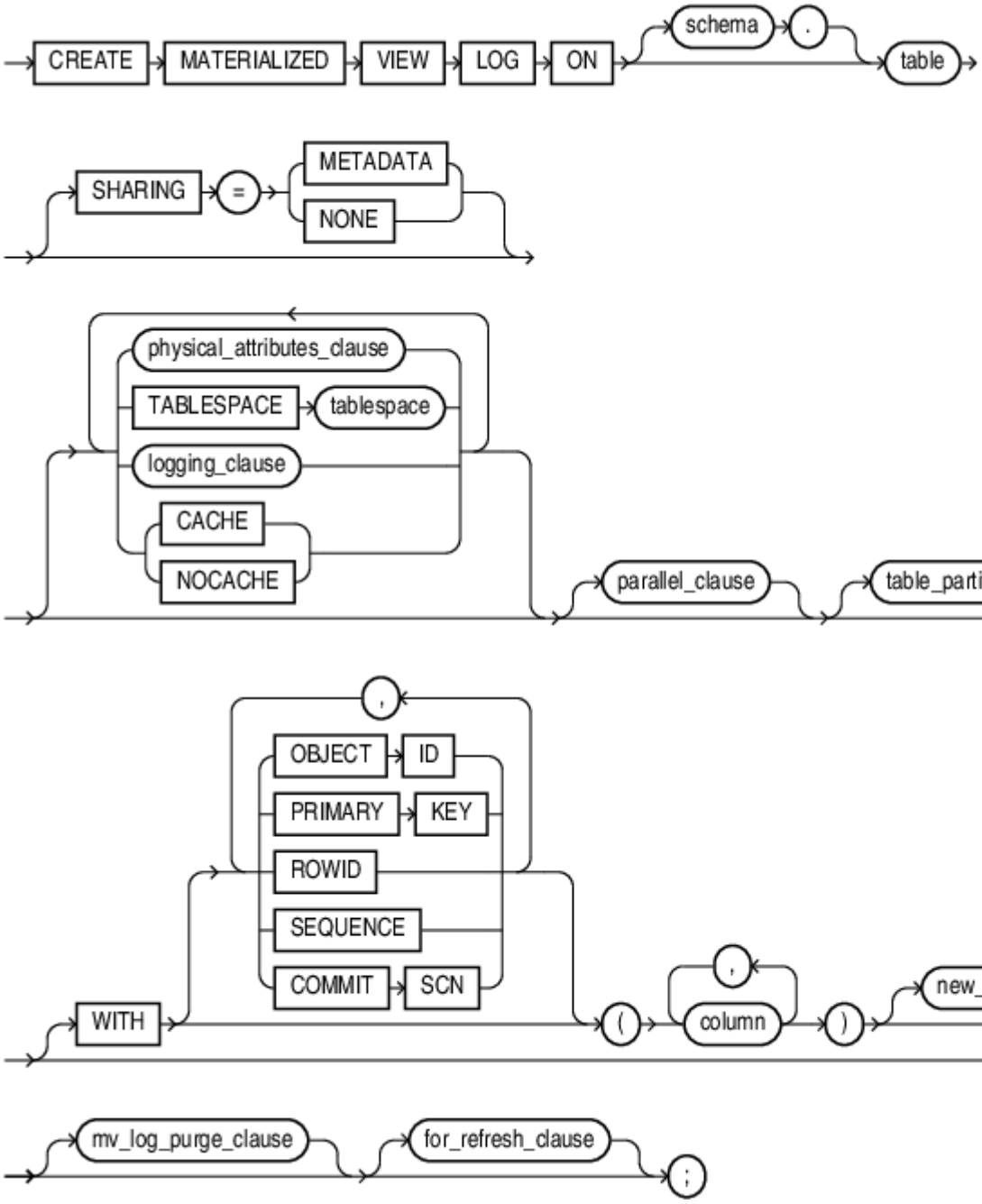
制限事項

CREATE MATERIALIZED VIEW LOG文では、マスター表の次の列はサポートされていません。

- 非表示列
- ID列
- BFILE列
- 時制有効性列

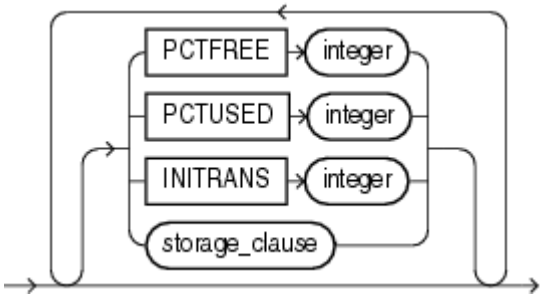
構文

create_materialized_vw_log::=



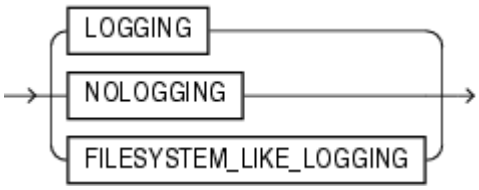
(physical_attributes_clause::=、logging_clause::=、parallel_clause::=、table_partitioning_clauses::= (CREATE TABLE内)、new_values_clause::=、mv_log_purge_clause::=、for_refresh_clause::=)

physical_attributes_clause::=

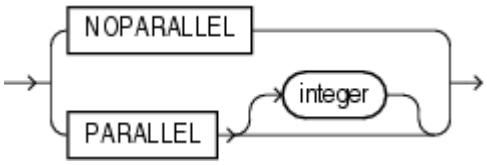


(storage_clause::=)

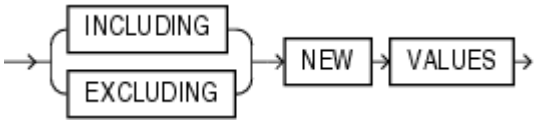
logging_clause::=



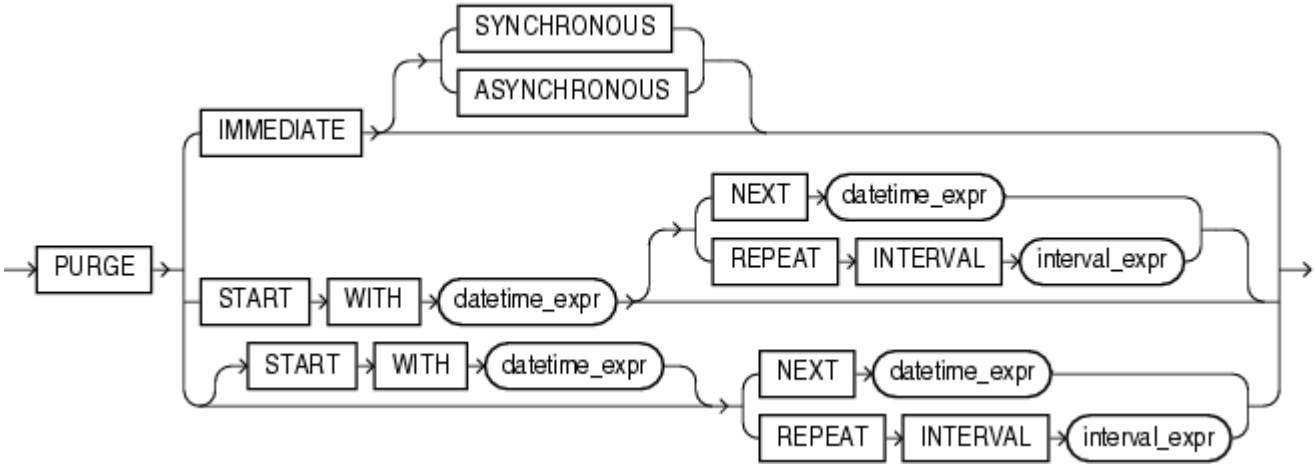
parallel_clause::=



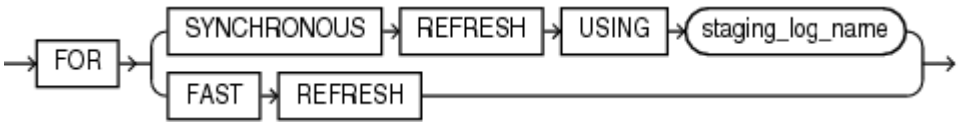
new_values_clause::=



mv_log_purge_clause::=



for_refresh_clause::=



セマンティクス

schema

マテリアライズド・ビュー・ログのマスター表が含まれているスキーマを指定します。schemaを省略した場合、マスター表は自分のスキーマ内に含まれているとみなされます。マテリアライズド・ビュー・ログは、マスター表のスキーマ内に作成されます。なお、ユーザーSYSのスキーマ内の表に対しては、マテリアライズド・ビュー・ログを作成できません。

table

マテリアライズド・ビュー・ログを作成するマスター表の名前を指定します。マスター表内で暗号化されるマテリアライズド・ビュー・ログの列は、同じ暗号化アルゴリズムを使用して暗号化されます。

マテリアライズド・ビュー・ログのマスター表の制限事項

マテリアライズド・ビュー・ログのマスター表には、次の制限が適用されます。

- 一時表またはビューに対しては、マテリアライズド・ビュー・ログを作成できません。
- 仮想列を含むマスター表に対しては、マテリアライズド・ビュー・ログを作成できません。

関連項目:

高速リフレッシュ用のマテリアライズド・ビュー・ログの作成: 例

SHARING

SHARING句は、アプリケーションのメンテナンスの際に、アプリケーション・ルートにオブジェクトを作成する場合に使用します。このタイプのオブジェクトはアプリケーション共通オブジェクトと呼ばれ、アプリケーション・ルートに属するアプリケーションPDBと共有できます。

次の共有属性のいずれかを使用して、オブジェクトを共有する方法を指定できます。

- `METADATA` - メタデータ・リンクはメタデータを共有しますが、そのデータは各コンテナに固有です。このタイプのオブジェクトは、メタデータリンク・アプリケーション共通オブジェクトと呼ばれます。
- `NONE` - オブジェクトは共有されず、アプリケーション・ルートでのみアクセスできます。

physical_attributes_clause

`physical_attributes_clause`を使用すると、マテリアライズド・ビュー・ログにおける物理特性および記憶特性を定義できます。

関連項目:

これらの句の詳細(デフォルト値など)は、「physical_attributes_clause」および「storage_clause」を参照してください。

TABLESPACE句

マテリアライズド・ビュー・ログを作成する表領域を指定します。この句を省略した場合、マテリアライズド・ビュー・ログのスキーマのデフォルト表領域内にマテリアライズド・ビュー・ログが作成されます。

logging_clause

`LOGGING`または`NOLOGGING`を指定すると、マテリアライズド・ビュー・ログのロギング特性を設定できます。デフォルトは、マテリアライズド・ビュー・ログが存在する表領域のロギング特性です。

関連項目:

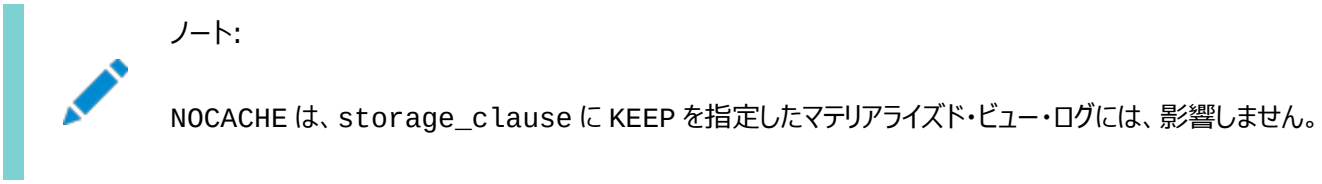
この句の詳細は、「logging_clause」を参照してください。

CACHE | NOCACHE

アクセス頻度の高いデータについて、`CACHE`は、全表スキャンの実行時にこのログ用に取り出された各ブロックを、バッファ・キャッ

シュの最低使用頻度(LRU)リストの最高使用頻度側に入れることを指定します。この属性は、小規模な参照表で有効です。

NOCACHEは、ブロックをLRUリストの最低使用頻度側に入れることを指定します。デフォルトはNOCACHEです。



ノート:

NOCACHE は、storage_clause に KEEP を指定したマテリアライズド・ビュー・ログには、影響しません。

関連項目:

CACHEまたはNOCACHEの指定の詳細は、「CREATE TABLE」を参照してください。

parallel_clause

parallel_clauseを使用すると、パラレル操作でマテリアライズド・ビュー・ログをサポートするかどうかを指定できます。

この句の詳細は、「CREATE TABLE」の「parallel_clause」を参照してください。

table_partitioning_clauses

table_partitioning_clausesを使用すると、マテリアライズド・ビュー・ログが、指定された範囲の値またはハッシュ・ファンクションでパーティション化されることを指定できます。マテリアライズド・ビュー・ログのパーティション化は、表のパーティション化と同じです。

関連項目:

「CREATE TABLE」の「table_partitioning_clauses」 を参照してください。

WITH句

WITH句を使用すると、マスター内の行の更新時に、マテリアライズド・ビュー・ログに主キー、ROWID、オブジェクトIDまたはこれらの行識別子の組合せを記録するかどうかを指定できます。また、マテリアライズド・ビュー・ログに順序を追加し、レコードに対する追加の順序情報を提供することもできます。

この句を指定すると、マテリアライズド・ビュー・ログがフィルタ列(副問合せマテリアライズド・ビューが参照する主キー以外の列)または結合列(副問合せのWHERE句で結合を定義する主キー以外の列)として参照する追加の列を記録するかどうかを指定できます。

この句を指定しない場合、またはPRIMARY KEY、ROWIDまたはOBJECT IDなしで句を指定する場合は、デフォルトで主キー値が格納されます。ただし、作成時にOBJECT IDまたはROWIDのみを指定した場合は、主キー値は暗黙的に格納されません。明示的に、またはデフォルトで作成された主キー・ログによって、主キー制約の追加の検証が実行されます。

OBJECT ID

OBJECT IDを指定すると、変更されるすべての行に対するシステム生成またはユーザー定義のオブジェクト識別子をマテリアライズド・ビュー・ログに記録する必要があることを指定できます。

OBJECT IDの制限事項

OBJECT IDは、オブジェクト表のログの作成時のみに指定でき、記憶表用には指定できません。

PRIMARY KEY

PRIMARY KEYを指定すると、変更されるすべての行の主キーをマテリアライズド・ビュー・ログに記録する必要があることを指定できます。

ROWID

ROWIDを指定すると、変更されるすべての行のROWIDをマテリアライズド・ビュー・ログに記録する必要があることを指定できます。

SEQUENCE

SEQUENCEを指定すると、追加の順序情報を提供する順序値をマテリアライズド・ビュー・ログに記録できます。更新操作後の高速リフレッシュには、順序番号が必要です。

関連項目:

マテリアライズド・ビュー・ログの順序番号の使用およびこの句の使用例については、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

COMMIT SCN

COMMIT SCN句を使用しない場合、マテリアライズド・ビュー・ログはタイムスタンプ・ベースになり、マテリアライズド・ビューのリフレッシュを準備するためにいくつかの設定操作が必要になります。COMMIT SCNを指定すると、タイムスタンプのかわりにコミットSCNデータが使用されます。この設定により、設定操作が必要なくなり、マテリアライズド・ビューのリフレッシュ速度を向上できます。

コミットSCNベース・マテリアライズド・ビュー・ログを使用するマスター表には、次のようなローカル・マテリアライズド・ビューを作成できます(ON COMMITおよびON DEMANDを含む)。

- マテリアライズド集計ビュー(単一表に対するマテリアライズド集計ビューを含む)
- マテリアライズド結合ビュー
- 主キーおよびROWIDに基づく単一表マテリアライズド・ビュー
- UNION ALLマテリアライズド・ビュー(各UNION ALLブランチはここに示したマテリアライズド・ビューのいずれかのタイプ)

コミットSCNベース・マテリアライズド・ビュー・ログを使用するマスター表には、リモート・マテリアライズド・ビューを作成できません。

COMMIT SCNの制限事項

COMMIT SCNには、次の制限事項が適用されます。

- 1つ以上のLOB列を持つ表に対するCOMMIT SCNの使用はサポートされておらず、ORA-32421が戻されます。
- 異なるマテリアライズド・ビュー・ログを使用するマスター表(つまり、タイムスタンプ・ベース・マテリアライズド・ビュー・ログを使用するマスター表およびコミットSCNベース・マテリアライズド・ビュー・ログを使用するマスター表)にはマテリアライズド・ビューを作成できず、ORA-32414が戻されます。
- COMMIT SCNを指定すると、FOR SYNCHRONOUS REFRESHが指定できなくなります。

column

変更されるすべての行に対して、マテリアライズド・ビュー・ログに値を記録する列を指定します。通常、フィルタ列および結合列を指定します。

WITH句の制限事項

この句には、次の制限事項があります。

- 各マテリアライズド・ビュー・ログに指定できるのは、PRIMARY KEY、ROWID、OBJECT ID、SEQUENCE、および列リストを1つずつです。
- 主キー列は、マテリアライズド・ビュー・ログに暗黙的に記録されます。そのため、columnが主キー列の1つを含む場合は、次の結合はいずれも指定できません。

```
WITH ... PRIMARY KEY ... (column)
WITH ... (column) ... PRIMARY KEY
WITH (column)
```

関連項目:

- マテリアライズド・ビュー・ログに値を明示的および暗黙的に含める方法については、「CREATE MATERIALIZED VIEW」を参照してください。
- フィルタ列および結合列の詳細は、『Oracle Database管理者ガイド』を参照してください。
- 「マテリアライズド・ビュー・ログへのフィルタ列の指定: 例」および「マテリアライズド・ビュー・ログへの結合列の指定: 例」を参照してください。

NEW VALUES句

NEW VALUES句を使用すると、更新DML操作で、古い値と新しい値の両方をマテリアライズド・ビュー・ログに保存するかどうかを指定できます。

関連項目:

マテリアライズド・ビュー・ログへの新しい値の追加: 例

INCLUDING

INCLUDINGを指定すると、新しい値と古い値の両方をログに保存できます。このログが単一表マテリアライズド集計ビューの表用で、マテリアライズド・ビューに高速リフレッシュを実行する場合、INCLUDINGを指定してください。

EXCLUDING

EXCLUDINGを指定すると、ログに新しい値が記録されなくなります。これはデフォルトです。この句を使用すると、新しい値の記録によるオーバーヘッドを回避できます。マスター表に高速リフレッシュが可能な単一表マテリアライズド集計ビューが定義されている場合は、この句を使用しないでください。

mv_log_purge_clause

この句を使用すると、マテリアライズド・ビュー・ログのパージ時間を指定できます。

- IMMEDIATE SYNCHRONOUS: マテリアライズド・ビュー・ログは、リフレッシュの直後に消去されます。これはデフォルトです。
- IMMEDIATE ASYNCHRONOUS: マテリアライズド・ビュー・ログは、リフレッシュ操作後に、別のOracleスケジューラ・ジョブで消去されます。
- START WITH、NEXTおよびREPEAT INTERVALは、CREATEまたはALTER MATERIALIZED VIEW LOG文で開始される、マテリアライズド・ビューのリフレッシュに依存しないスケジュール実行のパージを設定します。これは、

CREATEまたはALTER MATERIALIZED VIEW文の、スケジュール実行のリフレッシュ構文と似ています。

- START WITH日時式は、パージをいつ開始するかを指定します。
- NEXT日時式は、パージの次の実行時間を計算します。

REPEAT INTERVALを指定すると、次の実行時間は、SYSDATE + interval_exprとなります。

CREATE MATERIALIZED VIEW LOG文でスケジュール実行のパージを指定すると、ログ・パージを実行するOracleスケジューラ・ジョブが作成されます。このジョブは、DBMS_SNAPSHOT.PURGE_LOGプロシージャをコールして、マテリアライズド・ビュー・ログを消去します。この処理によって、マテリアライズド・ビューを何度もリフレッシュする場合のパージ・コストを軽減できます。

mv_log_purge_clauseの制限事項

この句は、一時表のマテリアライズド・ビュー・ログでは無効です。

関連項目:

マテリアライズド・ビュー・ログのパージの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

for_refresh_clause

この句を使用すると、マテリアライズド・ビュー・ログを使用するリフレッシュ方法を指定できます。指定できるリフレッシュ方法は、マスター表ごとに1つのみです。

FOR SYNCHRONOUS REFRESH

この句を指定すると、同期リフレッシュに使用可能なステージング・ログを作成できます。staging_log_nameを使用すると、作成するステージング・ログの名前を指定できます。このステージング・ログは、マスター表が存在するスキーマ内に作成されます。

ステージング・ログを作成すると、マスター表に対する直接のDML操作は実行できなくなります。データの変更操作の準備と実行には、DBMS_SYNC_REFRESHパッケージに含まれるプロシージャを使用する必要があります。

同期リフレッシュの制限事項

同期リフレッシュには、次の制限事項が適用されます。

- FOR SYNCHRONOUS REFRESHを指定すると、COMMIT SCNは指定できなくなります。
- 同期リフレッシュの対象にするには、マスター表が次の条件を満たしている必要があります。
  - マスター表がファクト表の場合は、その表がパーティション化されている必要があります。
  - マスター表は、キーを保持している必要があります。マスター表がディメンション表の場合は、その表に対して定義された主キーを保持している必要があります。マスター表がファクト表の場合は、そのファクト表に結合されたディメンション表の外部キーになる列のセットがキーになるとみなされます。
  - マスター表は、その表に対して定義されたNULL以外の仮想プライベート・データベース(VPD)ポリシーやトリガーを保持できません。

  Oracle Databaseでは、上記のすべての条件が満たされていなくても、マスター表に対するステージング・ログを作成できます。ただし、そのマスター表は同期リフレッシュの対象にはなりません。
- マスター表に対する既存のマテリアライズド・ビューは、REFRESH ON DEMANDモードのマテリアライズド・ビューであ

ることが必要です。既存のマテリアライズド・ビューがREFRESH ON COMMITモードのマテリアライズド・ビューの場合は、ALTER MATERIALIZED VIEWのalter_mv_refresh句でREFRESH ON DEMANDモードのマテリアライズド・ビューに変換してから、ステージング・ログを作成できます。

関連項目:

- 同期リフレッシュを使用するステップの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- DBMS_SYNC_REFRESHパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

FOR FAST REFRESH

この句を指定すると、高速リフレッシュに使用可能なマテリアライズド・ビュー・ログを作成できます。このマテリアライズド・ビュー・ログは、マスター表が存在するスキーマ内に作成されます。これはデフォルトです。

例

高速リフレッシュ用のマテリアライズド・ビュー・ログの作成: 例

次の文は、物理特性および記憶特性を指定するoe.customers表にマテリアライズド・ビュー・ログを作成します。

```
CREATE MATERIALIZED VIEW LOG ON customers
   PCTFREE 5
   TABLESPACE example
   STORAGE (INITIAL 10K);
```

customersのマテリアライズド・ビュー・ログは、主キー・マテリアライズド・ビューの高速リフレッシュのみサポートします。

次の文は、ROWID句を指定してマテリアライズド・ビュー・ログの別のバージョンを作成します。これによって、高速リフレッシュが使用可能になるマテリアライズド・ビューのタイプが追加されます。

```
CREATE MATERIALIZED VIEW LOG ON customers WITH PRIMARY KEY, ROWID;
```

このcustomersのマテリアライズド・ビュー・ログによって、ROWIDマテリアライズド・ビューおよびマテリアライズド結合ビューに対する高速リフレッシュが実行可能になります。マテリアライズド集計ビューの高速リフレッシュを指定するには、次の例に示すとおり、SEQUENCEおよびINCLUDING NEW VALUES句も指定する必要があります。

マテリアライズド・ビュー・ログへのパージの繰返し間隔の指定: 例

次の文は、oe.orders表にマテリアライズド・ビュー・ログを作成します。このログの内容は、マテリアライズド・ビュー・ログの作成日の5日後から5日間に1回、消去されます。

```
CREATE MATERIALIZED VIEW LOG ON orders
  PCTFREE 5
  TABLESPACE example
  STORAGE (INITIAL 10K)
  PURGE REPEAT INTERVAL '5' DAY;
```

マテリアライズド・ビュー・ログへのフィルタ列の指定: 例

次の文は、sh.sales表にマテリアライズド・ビュー・ログを作成し、作成されたマテリアライズド・ビュー・ログは「マテリアライズド集計ビューの作成: 例」で使用されます。ここでは、このマテリアライズド・ビューで参照される表のすべての列をフィルタ列として指定します。

```
CREATE MATERIALIZED VIEW LOG ON sales
   WITH ROWID, SEQUENCE(amount_sold, time_id, prod_id)
   INCLUDING NEW VALUES;
```

マテリアライズド・ビュー・ログへの結合列の指定: 例

次の文は、サンプルのoeスキーマのorder_items表にマテリアライズド・ビュー・ログを作成します。ログには、主キーおよび「高速リフレッシュ可能なマテリアライズド・ビューの作成: 例」で結合列として使用されたproduct_idが記録されます。

```
CREATE MATERIALIZED VIEW LOG ON order_items WITH (product_id);
```

マテリアライズド・ビュー・ログへの新しい値の追加: 例

次の例では、INCLUDING NEW VALUESを指定するoe.product_information表にマテリアライズド・ビュー・ログを作成します。

```
CREATE MATERIALIZED VIEW LOG ON product_information
   WITH ROWID, SEQUENCE (list_price, min_price, category_id), PRIMARY KEY
   INCLUDING NEW VALUES;
```

次の文は、マテリアライズド集計ビューを作成し、product_informationログを使用します。

```
CREATE MATERIALIZED VIEW products_mv
   REFRESH FAST ON COMMIT
   AS SELECT SUM(list_price - min_price), category_id
         FROM product_information
         GROUP BY category_id;
```

マスター表に定義されたログに古い値と新しい値の両方が含まれるため、このマテリアライズド・ビューでは、高速リフレッシュを実行できます。

同期リフレッシュのステージング・ログの作成: 例

次の文は、sh.salesファクト表にステージング・ログを作成します。このステージング・ログには、mystage_logという名前が付けられ、shスキーマに格納されます。これは、同期リフレッシュに使用できます。

```
CREATE MATERIALIZED VIEW LOG ON sales
   PCTFREE 5
   TABLESPACE example
   STORAGE (INITIAL 10K)
   FOR SYNCHRONOUS REFRESH USING mystage_log;
```

CREATE MATERIALIZED ZONEMAP

目的

CREATE MATERIALIZED ZONEMAP文を使用して、ゾーン・マップを作成します。

ゾーン・マップは、ゾーンの情報を格納する特別なタイプのマテリアライズド・ビューです。ゾーンは、1つ以上の表の列の値を格納するディスク上の一連の連続したデータ・ブロックです。通常、複数のゾーンが表の列のすべての値を格納するために必要です。ゾーン・マップは、各ゾーンに格納された最小および最大の表の列値を追跡します。

ゾーン・マップによって、表スキャンのI/OおよびCPUコストを削減できます。SQL文にゾーン・マップの列の述語が含まれる場合、データベースは述語の値と各ゾーンに格納されている最小および最大の表の列値を比較して、SQL実行中に読み込まれるゾーンを判断します。

Oracle Databaseでは次の種類のゾーン・マップがサポートされます。

- 基本的なゾーン・マップは単一の表に定義され、表の指定された列のゾーン情報を保持します。

  create_zonemap_on_table句を指定するか、定義する副問合せのFROM句で単一の表を指定するcreate_zonemap_as_subquery句を指定して、基本的なゾーン・マップを作成できます。

- 結合ゾーン・マップは2つ以上の結合した表に定義され、結合した表の指定された列のゾーン情報を保持します。

  create_zonemap_as_subquery句を指定して、結合ゾーン・マップを作成できます。定義する副問合せのFROM句では、1つ以上の他の表と左側外部結合されている表を指定する必要があります。

ゾーン・マップは、データ・ウェアハウス環境のスター・スキーマで一般的に使用されます。ただし、ゾーン・マップの作成にはスター・スキーマは必須ではありません。どちらの場合も、このマニュアルでは、スター・スキーマという用語を使用してゾーン・マップの表を参照します。結合ゾーン・マップでは、結合の外部表は、ファクト表と呼ばれ、この表を結合する表をディメンション表と呼びます。これらの表をあわせて、ゾーン・マップ実表と呼びます。基本的なゾーン・マップでは、ゾーン・マップが定義される単一の表は、ゾーン・マップのファクト表および実表と呼ばれます。

ゾーン・マップの実表をパーティション表またはコンポジット・パーティション表にすることができます。この場合、ゾーン・マップは、各パーティション(およびサブパーティション)と各ゾーンの最小および最大の列値を保持します。

属性クラスタリングの使用に関係なくゾーン・マップを作成できます。

- 属性クラスタリングで使用するゾーン・マップを作成するには、次の方法のいずれかを使用します。
  - CREATE MATERIALIZED ZONEMAP文を使用し、ゾーン・マップの属性クラスタ列を含めます。詳細は、CREATE TABLEのattribute_clustering_clauseおよびALTER TABLEのattribute_clustering_clause句を参照してください。
  - 属性クラスタ表の作成時または変更時にWITH MATERIALIZED ZONEMAP句を指定します。詳細は、CREATE TABLEのzonemap_clauseおよびALTER TABLEのMODIFY CLUSTERING句を参照してください。
- 属性クラスタリングを使用しないゾーン・マップを作成するには、CREATE MATERIALIZED ZONEMAP文を使用し、ゾーン・マップにクラスタリングされている属性がない列を含めます。

関連項目:

ゾーン・マップの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

前提条件

自分のスキーマへのゾーン・マップの作成

- CREATE MATERIALIZED VIEWシステム権限と、CREATE TABLEまたはCREATE ANY TABLEのいずれかのシステム権限が必要です。
- 所有していないゾーン・マップの実表にアクセスする場合は、各表に対するREADまたはSELECTオブジェクト権限またはREAD ANY TABLEまたはSELECT ANY TABLEシステム権限が必要です。

別のユーザーのスキーマへのゾーン・マップの作成

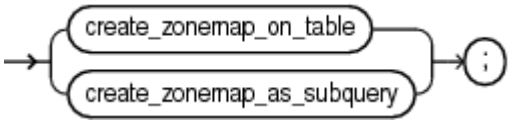
- CREATE ANY MATERIALIZED VIEWシステム権限が必要です。
- ゾーン・マップの所有者には、CREATE TABLEシステム権限が必要です。スキーマ所有者が所有していないゾーン・マップの実表にアクセスする場合は、各表に対するREADまたはSELECTオブジェクト権限またはREAD ANY TABLEまたはSELECT ANY TABLEシステム権限が所有者に必要です。

REFRESH ON COMMITモードのゾーン・マップを作成する場合(REFRESH ON COMMIT句を使用する場合)は、前述の権限の他に、所有していない実表に対するON COMMIT REFRESHオブジェクト権限、またはON COMMIT REFRESHシステム権限が必要です。マテリアライズド・ビューと異なり、実表のマテリアライズド・ビュー・ログがなくても、REFRESH ON COMMITモードのゾーン・マップを作成できます。

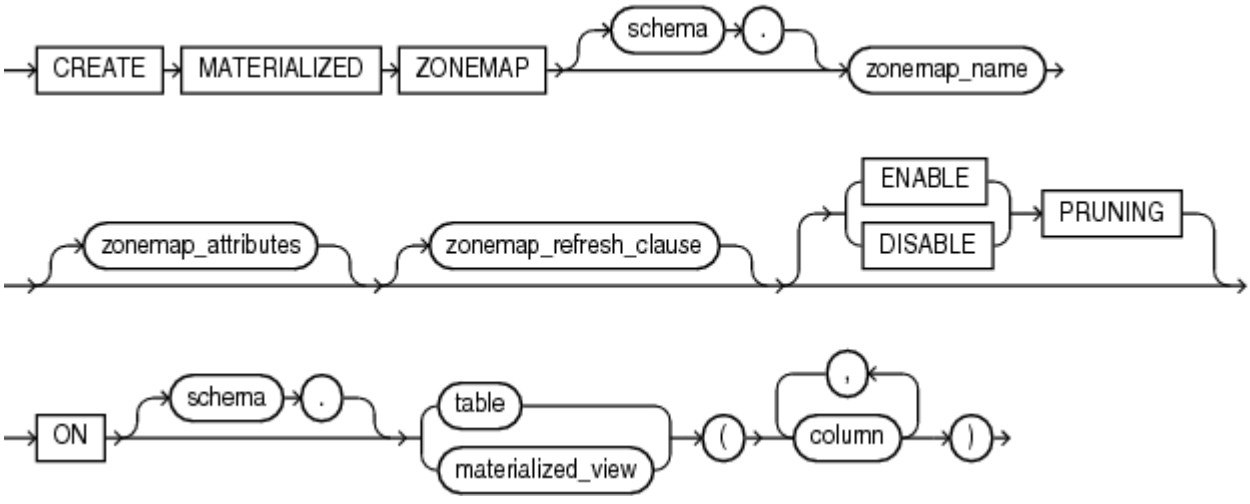
ゾーン・マップを作成する場合、ゾーン・マップのすべてのスキーマに1つの内部表および少なくとも1つの索引が作成されます。これらのオブジェクトは、ゾーン・マップのデータをメンテナンスするために使用されます。これらのオブジェクトの作成に必要な権限を持つか、これらのオブジェクトを格納するターゲット表領域への十分な割当て制限があるか、UNLIMITED TABLESPACEシステム権限を持つ必要があります。

構文

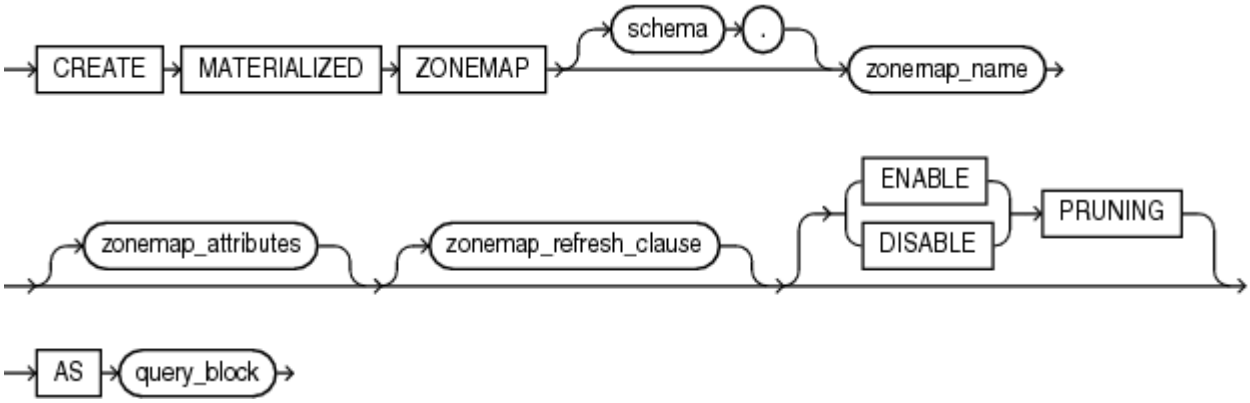
create_materialized_zonemap::=



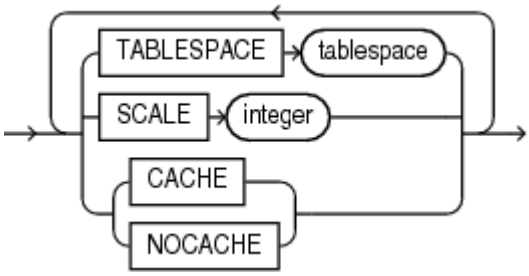
create_zonemap_on_table::=



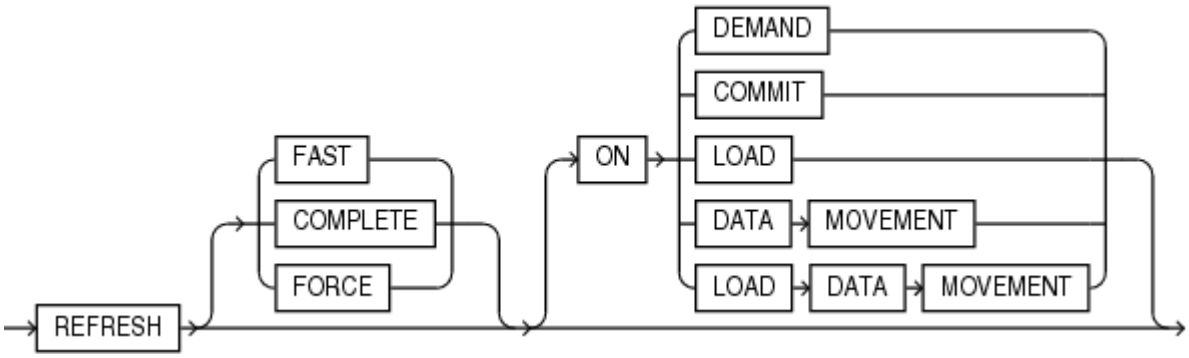
create_zonemap_as_subquery::=



zonemap_attributes::=



zonemap_refresh_clause::=



ノート:

`zonemap_refresh_clause` を指定する場合、REFRESH キーワードの後に少なくとも 1 つの句を指定する必要があります。

セマンティクス

create_zonemap_on_table

この句を使用して、基本的なゾーン・マップを作成します。

ON句

ON句には、まずゾーン・マップのファクト表を指定し、次にゾーン・マップに含めるファクト表の1つ以上の列をカッコ内に指定します。

指定されたファクト表の列ごとに、ゾーン・マップの2つの列が作成されます。これらの2つの列には、各ゾーンのファクト表の列の最小および最大の値が含まれます。最初に指定したファクト表columnに対してMIN_1_columnおよびMAX_1_column、2番目に指定したファクト表columnに対してMIN_2_columnおよびMAX_2_columnといった形式でゾーン・マップ列の名前が生成されます。

schemaを指定しない場合、自分のスキーマ内にファクト表があるとみなされます。ファクト表には表またはマテリアライズド・ビューを指定できます。

create_zonemap_as_subquery

この句を使用して、基本的なゾーン・マップまたは結合ゾーン・マップを作成します。基本的なゾーン・マップを作成するには、定義する副問合せのFROM句の単一の実表を指定します。結合ゾーン・マップを作成するには、定義する副問合せのFROM句の1つ以上の他の表に左側外部結合される表を指定します。

column_alias

ゾーン・マップに含める表の列ごとに列の別名を指定できます。列の別名のリストによって、競合する列名が明示的に解決されます。したがって、定義する副問合せのSELECTリストで別名を指定する必要がなくなります。この句で列の別名を指定する場合は、定義する副問合せのSELECTリストで各列の別名を指定する必要があります。指定する最初の列の別名は、SELECTリストの最初の列のSYS_OP_ZONE_IDファンクション式に対応するZONE_ID$である必要があります。

AS query_block

ゾーン・マップの定義副問合せを指定します。副問合せは、単一のquery_blockで構成する必要があります。query_blockのSELECT、FROM、WHEREおよびGROUP BY句のみ指定できます。それらの句は次の要件を満たす必要があります。

- SELECTリストの最初の列は、SYS_OP_ZONE_IDファンクション式である必要があります。詳細は、SYS_OP_ZONE_IDを参照してください。
- SELECTリストの残りの列は、ゾーン・マップに含める列の最小および最大の値を戻すファンクション式である必要があります。列ごとに、次の形式のファンクション式の組を指定します。

  ```
  MIN([table.]column), MAX([table.]column)
  ```

  tableには、列を含む表の名前または表の別名を指定します。表にはファクト表またはディメンション表を指定できます。columnには、列の名前または列の別名を指定します。
- FROM句にはファクト表のみ、またはファクト表とそれぞれがファクト表に左側外部結合されている1つ以上のディメンション表を指定できます。FROM句にLEFT [OUTER] JOIN構文を指定したり、外部結合演算子(+)をWHERE句の結合条件のディメンション表の列に適用できます。表の別名をFROM句の任意の表にオプションで指定できます。ファクト表およびディメンション表に表またはマテリアライズド・ビューを指定できます。
- WHERE句には、外部結合演算子(+)を使用した左側外部結合条件のみ指定できます。
- SELECTリストの最初の列に指定したSYS_OP_ZONE_IDファンクション式を使用して、GROUP BY句を指定する必要があります。

schema

ゾーン・マップを含むスキーマを指定します。schemaを指定しない場合、自分のスキーマにそのゾーン・マップが作成されます。

zonemap_name

変更するゾーン・マップの名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満

たしている必要があります。

zonemap_attributes

この句を使用してゾーン・マップの次の属性を指定します: TABLESPACE、SCALE、PCTFREE、PCTUSEDおよびCACHEまたはNOCACHE。

TABLESPACE

ゾーン・マップを作成するtablespaceを指定します。この句を省略すると、ゾーン・マップを含むスキーマのデフォルト表領域にゾーン・マップが作成されます。

SCALE

この句を使用すると、ゾーンを形成する連続したディスク・ブロックの数を判断するゾーン・マップ・スケールを指定できます。このスケールは、2の累乗を表す整数値です。たとえば、10のスケールの場合、2の10乗(1024)まで連続したディスク・ブロックがゾーンを形成することを意味します。integerの場合、4から16の値(それらの値を含む)を指定します。推奨値は10です。この値はデフォルトです。

PCTFREE

ゾーン・マップの各データ・ブロック内で、ゾーン・マップの行を将来更新するために確保しておく領域の割合を表すintegerを指定します。整数値の範囲は0から99の値(それらの値を含む)です。デフォルト値は10です。PCTFREEパラメータの詳細は、physical_attributes_clauseを参照してください。

PCTUSED

使用済領域のうち、ゾーン・マップのデータ・ブロックごとに確保される最小限の割合を表すintegerを指定します。整数値の範囲は0から99の値(それらの値を含む)です。デフォルト値は40です。PCTUSEDパラメータの詳細は、physical_attributes_clauseを参照してください。

CACHE | NOCACHE

頻繁にアクセスするデータの場合、CACHEを指定すると、全表スキャンの実行時、このゾーン・マップに対して取り出されたは、バッファ・キャッシュで最低使用頻度(LRU)リストの最高使用頻度側に配置されます。

NOCACHEは、ブロックをLRUリストの最低使用頻度側に入れることを指定します。デフォルトはNOCACHEです。

zonemap_refresh_clause

この句を使用して、ゾーン・マップのデフォルトのリフレッシュ方法およびモードを指定します。リフレッシュ方法(FAST、COMPLETEまたはFORCE)を指定しないと、デフォルトの方法でFORCEが指定されます。リフレッシュ・モード(ON句)を指定しない場合、ON LOAD DATA MOVEMENTがデフォルト・モードです。

FAST

FASTを指定すると、高速リフレッシュ方法が使用されます。この方法では、実表に対して発生した変更に従ってリフレッシュを実行します。ゾーン・マップがマテリアライズド・ビューのタイプとして内部的に実装される場合、実表のマテリアライズド・ビュー・ログはゾーン・マップの高速リフレッシュを実行するために必要ではありません。

COMPLETE

COMPLETEを指定すると、完全リフレッシュ方法が使用されます。この方法は、ゾーン・マップを定義する問合せを実行することによって実装されます。完全リフレッシュを要求すると、高速リフレッシュが実行可能であっても、完全リフレッシュが実行されます。

FORCE

FORCEを指定すると、リフレッシュ時に、高速リフレッシュが可能な場合は高速リフレッシュを実行し、可能でない場合は完全リフレッシュを実行するように指定できます。これはデフォルトです。

ON DEMAND

ON DEMANDを指定すると、ALTER MATERIALIZED ZONEMAP ... REBUILD文を手動で発行しない場合にデータベースでゾーン・マップがリフレッシュされません。この句を指定する場合、ゾーン・マップがREFRESH ON DEMANDモードのゾーン・マップと呼ばれます。ゾーン・マップの再作成の詳細は、ALTER MATERIALIZED ZONEMAPのドキュメントのREBUILDを参照してください。

ON COMMIT

ON COMMITを指定すると、ゾーン・マップの実表に対するトランザクションをコミットするときに、必ずリフレッシュが発生します。この句を指定する場合、ゾーン・マップがREFRESH ON COMMITモードのゾーン・マップと呼ばれます。この句を指定すると、リフレッシュ操作がコミット処理の一部として行われるため、コミットの完了に時間がかかります。

ON LOAD

ON LOADを指定すると、INSERTまたはMERGE操作のいずれかの結果として実行されるダイレクト・パス・インサート(シリアルまたはパラレル)の最後にリフレッシュが発生します。

ON DATA MOVEMENT

ON DATA MOVEMENTを指定すると、次のデータ移動操作の最後にリフレッシュが発生します。

- DBMS_REDEFINITIONパッケージを使用したデータの再定義
- ALTER TABLEの次の句で指定される表パーティションのメンテナンス操作: coalesce_table、merge_table_partitions、move_table_partitionおよびsplit_table_partition

ON LOAD DATA MOVEMENT

ON LOAD DATA MOVEMENTを指定すると、ダイレクト・パス・インサートまたはデータ移動操作の最後にリフレッシュが発生します。これはデフォルトです。

ENABLE | DISABLE PRUNING

この句を使用すると、プルーニングのゾーン・マップの使用を制御できます。

- ENABLE PRUNINGを指定して、プルーニングのゾーン・マップの使用を有効化します。これはデフォルトです。
- DISABLE PRUNINGを指定して、プルーニングのゾーン・マップの使用を無効化します。オプティマイザはプルーニングのゾーン・マップを使用しませんが、データベースは継続してゾーン・マップを保持します。

設定がENABLE PRUNINGである場合、オプティマイザは、次の条件のいずれかを含むSQL操作中にプルーニングのゾーン・マップの使用を検討します。

- 比較条件: =、<=、<、>=、>

  片側が列名で反対側がリテラルまたはバインド変数の簡単な比較条件である必要があります。たとえば:

  ```
  WHERE country_name = 'United States of America'
  WHERE country_name = :country1
  WHERE 10000 >= salary
  ```

- IN条件

IN条件は、左側に列名、右側にリテラルまたはバインド変数の式リストを指定する必要があります。たとえば:

```
WHERE country_name IN ('Germany', 'India', 'United Kingdom')
WHERE country_name IN (:country1, :country2, :country3)
WHERE prod_id IN (20, 48, 132, 143)
```

- LIKE条件

  LIKE条件は、左側に列名、右側にテキスト・リテラルを指定する必要があります。テキスト・リテラルはLIKE条件のパターンで、少なくとも1つのパターン一致文字を含む必要があります。有効なパターン一致文字は、1つの文字と完全に一致するアンダースコア(_)およびゼロ以上の文字と一致するパーセント記号(%)です。パターンの最初の文字にパターン一致文字を指定できません。たとえば:

```
WHERE prod_name LIKE 'DVD%'
WHERE prod_name LIKE 'Model%Cordless%Battery'
WHERE prod_name LIKE 'CD%Pack of _'
```

関連項目:

条件の詳細は、「条件」を参照してください。

ゾーン・マップの制限事項

ゾーン・マップには、次の制限事項があります。

- 最大で1つのゾーン・マップにファクト表を指定できます。複数のゾーン・マップにディメンション表を指定できます。1つのゾーン・マップにファクト表を指定し、他のゾーン・マップにディメンション表を指定できます。
- ゾーン・マップの実表に外部表、索引構成表、リモート表、一時表またはビューを指定できません。
- ゾーン・マップの実表は、ユーザーSYSのスキーマ内に指定できません。
- ゾーン・マップはパーティション化できます。
- BFILE、BLOB、CLOB、LONG、LONG RAWまたはNCLOB以外のスカラー・データ型の列のゾーン・マップを定義できます。
- ゾーン・マップの定義する副問合せで指定されるすべての結合は、左側のファクト表の左側外部等価結合である必要があります。
- ゾーン・マップの定義する副問合せのFROM句がマテリアライズド・ビューを参照する場合、ゾーン・マップのリフレッシュの前に、マテリアライズド・ビューをリフレッシュしておく必要があります。
- ゾーン・マップで直接DML操作を実行できません。
- ゾーン・マップの各列には、宣言された照合BINARYまたはUSING_NLS_COMPのいずれかが必要です。

例

次の文は、sales_zmapと呼ばれる基本的なゾーン・マップを作成します。このゾーン・マップは、表salesの列cust_idおよびprod_idを追跡します。

```
CREATE MATERIALIZED ZONEMAP sales_zmap
  ON sales(cust_id, prod_id);
```

次の文は、前の例で作成されたゾーン・マップと似ているsales_zmapと呼ばれる基本的なゾーン・マップを作成します。ただし、この文は定義する副問合せを使用して、ゾーン・マップを作成します。

```
CREATE MATERIALIZED ZONEMAP sales_zmap
  AS SELECT SYS_OP_ZONE_ID(rowid),
            MIN(cust_id), MAX(cust_id),
            MIN(prod_id), MAX(prod_id)
     FROM sales
     GROUP BY SYS_OP_ZONE_ID(rowid);
```

次の文は、sales_zmapと呼ばれる結合ゾーン・マップを作成します。ゾーン・マップのファクト表はsalesで、ゾーン・マップに1つのディメンション表customersがあります。このゾーン・マップは、ディメンション表の2つの列cust_state_provinceおよびcust_cityを追跡します。

```
CREATE MATERIALIZED ZONEMAP sales_zmap
  AS SELECT SYS_OP_ZONE_ID(s.rowid),
            MIN(cust_state_province), MAX(cust_state_province),
            MIN(cust_city), MAX(cust_city)
     FROM sales s
          LEFT OUTER JOIN customers c ON s.cust_id = c.cust_id
     GROUP BY SYS_OP_ZONE_ID(s.rowid);
```

次の文は、sales_zmapと呼ばれる結合ゾーン・マップを作成します。ゾーン・マップのファクト表はsalesで、ゾーン・マップに2つのディメンション表productsとcustomersがあります。このゾーン・マップは、products表のprod_categoryとprod_subcategory、customers表のcountry_id、cust_state_provinceおよびcust_cityのディメンション表の5個の列を追跡します。

```
CREATE MATERIALIZED ZONEMAP sales_zmap
  AS SELECT SYS_OP_ZONE_ID(s.rowid),
            MIN(prod_category), MAX(prod_category),
            MIN(prod_subcategory), MAX(prod_subcategory),
            MIN(country_id), MAX(country_id),
            MIN(cust_state_province), MAX(cust_state_province),
            MIN(cust_city), MAX(cust_city)
    FROM sales s
       LEFT OUTER JOIN products p ON s.prod_id = p.prod_id
       LEFT OUTER JOIN customers c ON s.cust_id = c.cust_id
    GROUP BY sys_op_zone_id(s.rowid);
```

次の文は、前の例で作成されたゾーン・マップと同じ結合ゾーン・マップを作成します。違いは、前の例がFROM句にLEFT OUTER JOIN構文を使用し、次の例がWHERE句に外部結合演算子(+)を使用しているだけです。

```
CREATE MATERIALIZED ZONEMAP sales_zmap
  AS SELECT SYS_OP_ZONE_ID(s.rowid),
            MIN(prod_category), MAX(prod_category),
            MIN(prod_subcategory), MAX(prod_subcategory),
            MIN(country_id), MAX(country_id),
            MIN(cust_state_province), MAX(cust_state_province),
            MIN(cust_city), MAX(cust_city)
     FROM sales s, products p, customers c
     WHERE s.prod_id = p.prod_id(+) AND
           s.cust_id = c.cust_id(+)
     GROUP BY sys_op_zone_id(s.rowid);
```

CREATE OPERATOR

目的

CREATE OPERATOR文を使用すると、新しい演算子を作成し、バインディングを定義できます。

演算子は、索引タイプ、SQL問合せおよびDML文で参照できます。それに対して演算子は、ファンクション、パッケージ、型および他のユーザー定義オブジェクトを参照します。

関連項目:

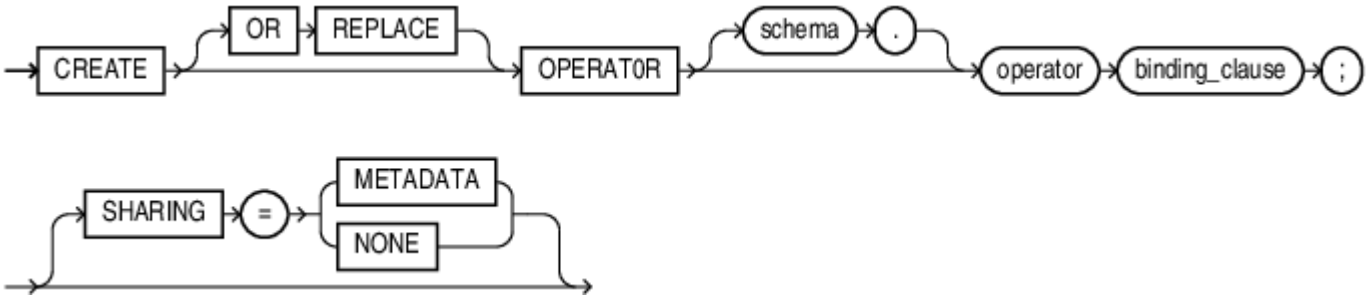
これらの依存性および演算子の概要は、『Oracle Databaseデータ・カートリッジ開発者ガイド』および『Oracle Database概要』を参照してください。

前提条件

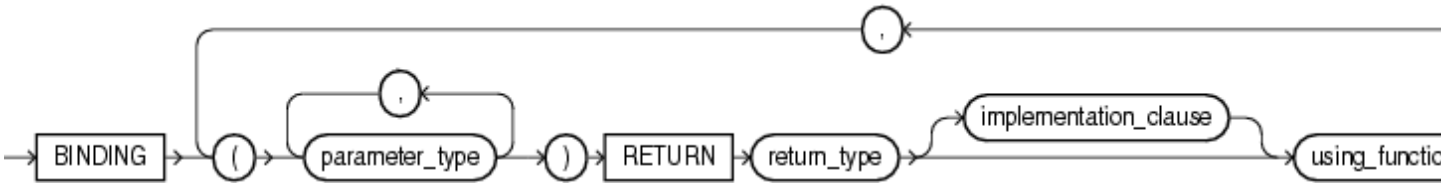
自分のスキーマ内に演算子を作成する場合は、CREATE OPERATORシステム権限が必要です。他のユーザーのスキーマ内に演算子を作成する場合は、CREATE ANY OPERATORシステム権限が必要です。どちらの場合も、参照するファンクションおよび演算子に対するEXECUTEオブジェクト権限が必要です。

構文

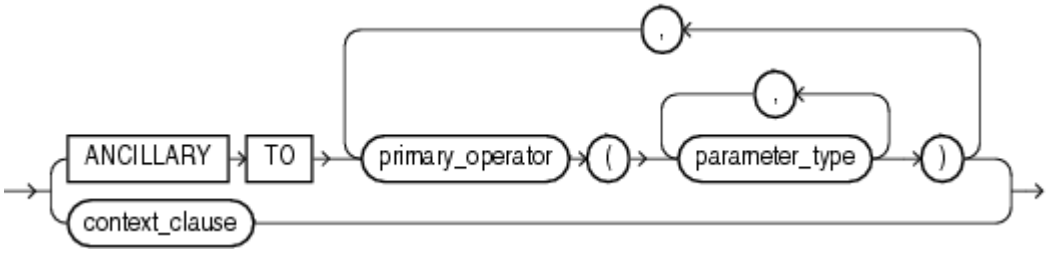
create_operator::=



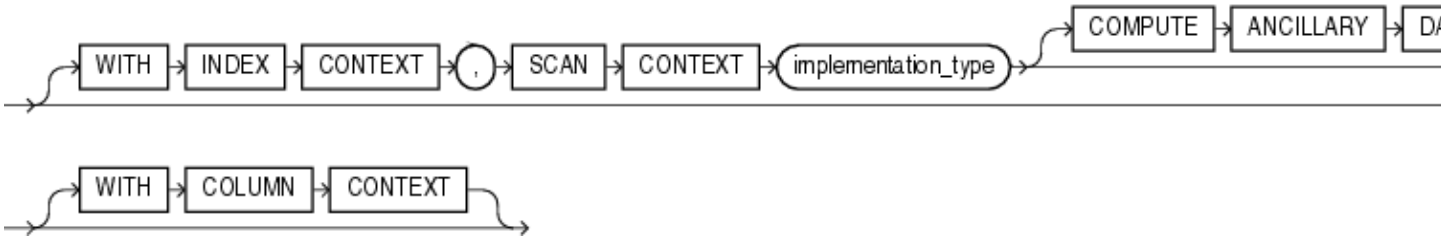
binding_clause::=



implementation_clause::=



context_clause::=



using_function_clause::=

セマンティクス

OR REPLACE

OR REPLACEを指定すると、演算子スキーマ・オブジェクトの定義を置換できます。

演算子の置換の制限事項

演算子に対して、その演算子をサポートする索引タイプなどの依存オブジェクトがない場合のみ、定義を置換できます。

schema

演算子が含まれているスキーマを指定します。schemaを省略した場合、自分のスキーマ内に演算子が作成されます。

operator

作成する演算子の名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

binding_clause

binding_clauseを使用すると、演算子をファンクションにバインドするために、1つ以上のパラメータ・データ型(parameter_type)を指定できます。各バインドのシグネチャ(対応するファンクションに対する引数のデータ型の順序)は、オーバーロードの規則に従って一意である必要があります。

parameter_type自体をオブジェクト型にできます。この場合、任意にスキーマで修飾することもできます。

演算子のバインドの制限事項

REF、LONGまたはLONG RAWのparameter_typeは指定できません。

関連項目:

オーバーロードの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

RETURN句

バインドに戻りデータ型を指定します。

return_type自体をオブジェクト型にできます。この場合、任意にスキーマで修飾することもできます。

戻りデータ型のバインドの制限事項

REF、LONGまたはLONG RAWのreturn_typeは指定できません。

SHARING

SHARING句は、アプリケーションのメンテナンスの際に、アプリケーション・ルートにオブジェクトを作成する場合に使用します。このタイプのオブジェクトはアプリケーション共通オブジェクトと呼ばれ、アプリケーション・ルートに属するアプリケーションPDBと共有できます。

次の共有属性のいずれかを使用して、オブジェクトを共有する方法を指定できます。

- METADATA - メタデータ・リンクはメタデータを共有しますが、そのデータは各コンテナに固有です。このタイプのオブジェクトは、メタデータリンク・アプリケーション共通オブジェクトと呼ばれます。
- NONE - オブジェクトは共有されず、アプリケーション・ルートでのみアクセスできます。

implementation_clause

この句を使用すると、バインドの実装を指定できます。

ANCILLARY TO句

ANCILLARY TO句を使用すると、演算子バインドが、指定した主演算子バインド(primary_operator)を補助することを指定できます。この句を指定する場合は、1つのみの数値パラメータで前のバインドを指定しないでください。

context_clause

context_clauseを使用すると、主演算子バインドを補助しないバインドのファンクション実装を指定できます。

WITH INDEX CONTEXT、SCAN CONTEXT

この句を使用すると、演算子のファンクション評価に、実装タイプで指定した索引およびスキャン・コンテキストが使用されるように指定できます。

COMPUTE ANCILLARY DATA

COMPUTE ANCILLARY DATAを指定すると、演算子バインディングによって補助データが計算されます。

WITH COLUMN CONTEXT

WITH COLUMN CONTEXTを指定すると、列情報が演算子のファンクション実装に渡されます。

この句を指定する場合は、実装するファンクションのシグネチャに、余分なODCIFuncCallInfo構造を1つ含める必要があります。

関連項目:

ODCIFuncCallInfoルーチンを使用する手順については、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。

using_function_clause

using_function_clauseを使用すると、バインディングを実装するファンクションを指定できます。function_nameには、スタンドアロン・ファンクション、パッケージ・ファンクション、型メソッドまたはこれらのシノニムを指定できます。

ファンクションが後で削除された場合は、演算子などのすべての依存オブジェクトに、INVALIDのマークが付けられます。ただし、その後ALTER OPERATOR ... DROP BINDING文を発行してバインディングを削除した場合は、後続の問合せおよびDMLで依存オブジェクトが再検証されます。

例

ユーザー定義演算子の作成: 例

この例は、非常に単純な等価性のファンクション実装を作成した後、このファンクションを使用する演算子を作成します。より詳細な例については、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。

```
CREATE FUNCTION eq_f(a VARCHAR2, b VARCHAR2) RETURN NUMBER AS
BEGIN
   IF a = b THEN RETURN 1;
   ELSE RETURN 0;
   END IF;
END;
/
CREATE OPERATOR eq_op
   BINDING (VARCHAR2, VARCHAR2)
   RETURN NUMBER
   USING eq_f;
```

CREATE OUTLINE

目的

ノート:

ストアド・アウトラインは非推奨になりました。ストアド・アウトラインは、下位互換性を保つために今でもサポートされています。ただし、かわりに SQL 計画管理を使用することをお薦めします。SQL 計画管理では、ストアド・アウトラインよりも非常に安定した SQL パフォーマンスを実現する SQL 計画ベースラインが作成されます。

DBMS_SPM パッケージの MIGRATE_STORED_OUTLINE ファンクションまたは Enterprise Manager Cloud Control を使用して、既存のストアド・アウトラインを SQL 計画ベースラインに移行できます。移行が完了したら、ストアド・アウトラインに移行済のマークが付けられ、削除できるようになります。DBMS_SPM パッケージの DROP_MIGRATED_STORED_OUTLINE ファンクションを使用して、システム上のすべての移行済ストアド・アウトラインを削除できます。

参照: SQL 計画管理の詳細は、『Oracle Database SQL チューニング・ガイド』を参照してください。
DBMS_SPM パッケージの詳細は、『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』を参照してください。

CREATE OUTLINE文を使用すると、ストアド・アウトラインを作成できます。ストアド・アウトラインは、実行計画を生成するためにオプティマイザが使用する属性集合です。最適化に影響する要因に変更があるかどうかにかかわらず、特定のSQL文が発行された場合に、実行計画の生成に影響するアウトラインの集合を使用するように、オプティマイザに指示します。これらの要因における変更を考慮するように、アウトラインを変更することもできます。

ノート:

影響を与える SQL 文には、アウトライン作成時に指定した文と一致する文字列を指定する必要があります。

関連項目:

- 実行計画の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。
- アウトラインの変更については、「ALTER OUTLINE」を参照してください。
- USE_STORED_OUTLINESパラメータおよびUSE_PRIVATE_OUTLINESパラメータの詳細は、「ALTER SESSION」および「ALTER SYSTEM」を参照してください。

前提条件

パブリック・アウトラインまたはプライベート・アウトラインの作成には、CREATE ANY OUTLINEシステム権限が必要です。

ソース・アウトラインからクローン・アウトラインを作成するには、SELECT_CATALOG_ROLEロールも必要です。

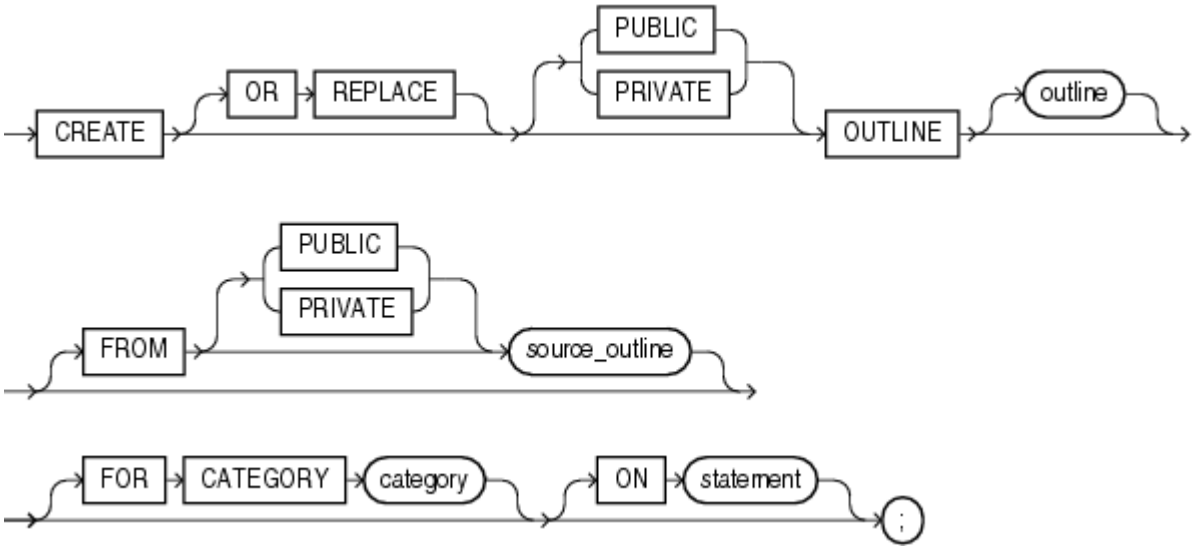
個々のセッションまたはシステムに対して、ストアド・アウトラインを動的に使用可能または使用禁止にできます。

- USE_STORED_OUTLINESパラメータを使用可能にすると、パブリック・アウトラインが使用可能になります。

- USE_PRIVATE_OUTLINESパラメータを使用可能にすると、プライベート・ストアド・アウトラインが使用可能になります。

構文

create_outline::=



ノート:

outline の後に指定する必須の句はありません。ただし、outline の後には 1 つ以上の句を指定する必要があり、FROM 句または ON 句のどちらかを指定する必要があります。

セマンティクス

OR REPLACE

OR REPLACEを指定すると、既存のアウトラインを同じ名前の新しいアウトラインと置き換えることができます。

PUBLIC | PRIVATE

PUBLICを指定すると、PUBLICが使用するアウトラインを作成できます。これはデフォルトです。

PRIVATEを指定すると、現行のセッションのみがプライベートに使用するアウトラインを作成できます。このアウトラインのデータは、現行のスキーマに格納されます。

outline

ストアド・アウトラインに割り当てる一意の名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。outlineを指定しない場合、データベースがアウトライン名を生成します。

関連項目:

アウトラインの作成: 例

FROM source_outline句

FROM句を使用すると、既存のアウトラインのコピーによって新しいアウトラインを作成できます。デフォルトでは、パブリック領域の

source_categoryが検索されます。PRIVATEを指定すると、現行のスキーマのアウトラインが検索されます。

アウトラインのコピーの制限事項

FROM句を指定する場合、ON句は指定できません。

関連項目:

「プライベート・クローン・アウトラインの作成: 例」および「プライベート・アウトラインのパブリック領域への公開: 例」を参照してください。

FOR CATEGORY句

ストアド・アウトラインをグループ化するために使用する任意の名前を指定します。たとえば、週末に使用するアウトラインの1つのカテゴリおよび四半期末に使用する別のカテゴリを指定できます。categoryを指定しない場合、アウトラインはDEFAULTカテゴリに格納されます。

ON句

文をコンパイルする際にアウトラインが作成されるSQL文を指定します。この句は、FROM句を使用して既存のアウトラインのコピーを作成する場合のみに指定するオプションです。

指定できる文は、SELECT、DELETE、UPDATE、INSERT ... SELECTまたはCREATE TABLE ... AS SELECTのいずれかです。

ON句の制限事項:

この句には、次の制限事項があります。

- ON句を指定する場合、FROM句は指定できません。
- マルチテーブルINSERT文でアウトラインを作成できません。
- ON句内のSQL文に、リモート・オブジェクトのDML操作を含めることはできません。

ノート:

後続の文で、同じ SQL 文に対して追加のアウトラインを指定できますが、同じ文の各アウトラインは CATEGORY 句で異なるカテゴリを指定する必要があります。

例

アウトラインの作成: 例

次の文は、ON文をコンパイルすることによってストアド・アウトラインを作成します。アウトラインはsalariesという名前で、specialカテゴリに格納されます。

```
CREATE OUTLINE salaries FOR CATEGORY special
   ON SELECT last_name, salary FROM employees;
```

USE_STORED_OUTLINESパラメータにspecialが設定されている場合、同じSELECT文が後でコンパイルされると、アウトラインsalariesを作成する場合と同様に実行計画が生成されます。

プライベート・クローン・アウトラインの作成: 例

次の文は、前述の例で作成したパブリック・カテゴリsalariesに基づいて、ストアド・プライベート・アウトラインmy_salariesを作成します。

```
CREATE OR REPLACE PRIVATE OUTLINE my_salaries
   FROM salaries;
```

プライベート・アウトラインのパブリック領域への公開: 例

次の文は、プライベート編集の後、プライベート・アウトラインをパブリック領域にコピー(公開)します。

```
CREATE OR REPLACE OUTLINE public_salaries
   FROM PRIVATE my_salaries;
```

CREATE PACKAGE

目的

パッケージはPL/SQLを使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

CREATE PACKAGE文を使用すると、ストアド・パッケージの仕様部を作成できます。パッケージとは、関連するプロシージャ、ファンクション、およびデータベース上にまとめて格納されるその他のプログラム・オブジェクトの集合のことです。パッケージ仕様部でこれらのオブジェクトを宣言します。後で指定するパッケージ本体では、これらのオブジェクトを定義します。

関連項目:

- パッケージ実装の指定については、「CREATE PACKAGE BODY」を参照してください。
- スタンドアロン・ファンクションおよびプロシージャの作成については、「CREATE FUNCTION」および「CREATE PROCEDURE」を参照してください。
- パッケージの変更および削除については、「ALTER PACKAGE」および「DROP PACKAGE」を参照してください。
- パッケージの説明と使用方法の詳細は、『Oracle Database開発ガイド』と『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

前提条件

自分のスキーマ内にパッケージを作成または再作成する場合は、CREATE PROCEDUREシステム権限が必要です。他のユーザーのスキーマ内にパッケージを作成または再作成する場合は、CREATE ANY PROCEDUREシステム権限が必要です。

Oracle Databaseのプリコンパイラ・プログラム内にCREATE PACKAGE文を埋め込む場合、キーワードEND-EXECに続けて、各言語の埋込みSQL文の終了記号を記述して文を終了する必要があります。

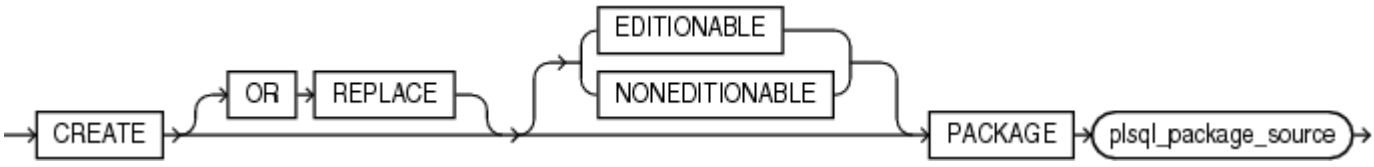
関連項目:

詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

構文

パッケージはPL/SQLを使用して定義されます。このため、このマニュアルの構文図ではSQLキーワードのみを示します。PL/SQLの構文、セマンティクスおよび例については、『Oracle Database PL/SQL言語リファレンス』を参照してください。

create_package::=



(plsql_package_source: 『Oracle Database PL/SQL言語リファレンス』を参照。)

セマンティクス

OR REPLACE

OR REPLACEを指定すると、既存のパッケージ仕様部を再作成できます。この句を指定した場合、パッケージに対して付与されていたオブジェクト権限を削除、再作成および再付与しなくても、既存のパッケージの仕様部を変更できます。パッケージ仕様部を変更した場合、その仕様部は自動的に再コンパイルされます。

再定義したパッケージに対して権限が付与されていたユーザーは、権限が再付与されなくてもそのパッケージにアクセスできます。

ファンクション索引がパッケージに依存している場合、索引にDISABLEDのマークが付きます。

関連項目:

パッケージ仕様部の再コンパイルについては、「ALTER PACKAGE」を参照してください。

[EDITIONABLE | NONEDITIONABLE]

この句を使用すると、schemaのスキーマ・オブジェクト・タイプPACKAGEのエディションニングが有効化されたときに、そのパッケージをエディション・オブジェクトにするか非エディション・オブジェクトにするかを指定できます。デフォルトは、EDITIONABLEです。エディション・オブジェクトと非エディション・オブジェクトの詳細は、『Oracle Database開発ガイド』を参照してください。

plsql_package_source

plsql_package_sourceの構文、セマンティクスおよび例については、『Oracle Database PL/SQL言語リファレンス』を参照してください。

CREATE PACKAGE BODY

目的

パッケージ本体はPL/SQLを使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

CREATE PACKAGE BODY文を使用すると、ストアド・パッケージの本体を作成できます。パッケージとは、関連するプロシージャ、ストアド・ファンクション、およびデータベース上にまとめて格納されるその他のプログラム・オブジェクトの集合のことです。パッケージ本体でこれらのオブジェクトを定義します。前述のCREATE PACKAGE文で定義したパッケージ仕様部で、これらのオブジェクトを宣言します。

一連のプロシージャやファンクションをスタンドアロンのスキーマ・オブジェクトとして作成するかわりの方法としてパッケージを使用する方法があります。

関連項目:

- スタンドアロン・ファンクションおよびプロシージャの作成については、「CREATE FUNCTION」および「CREATE PROCEDURE」を参照してください。
- 作成方法を含むパッケージの詳細は、「CREATE PACKAGE」を参照してください。
- パッケージの変更については、「ALTER PACKAGE」を参照してください。
- データベースからのパッケージの削除については、「DROP PACKAGE」を参照してください。

前提条件

自分のスキーマ内にパッケージを作成または再作成する場合は、CREATE PROCEDUREシステム権限が必要です。他のユーザーのスキーマ内にパッケージを作成または再作成する場合は、CREATE ANY PROCEDUREシステム権限が必要です。いずれの場合も、パッケージ本体をパッケージと同じスキーマに作成する必要があります。

Oracle Databaseのプリコンパイラ・プログラム内にCREATE PACKAGE BODY文を埋め込む場合、キーワードEND-EXECに続けて、各言語の埋込みSQL文の終了記号を記述して文を終了する必要があります。

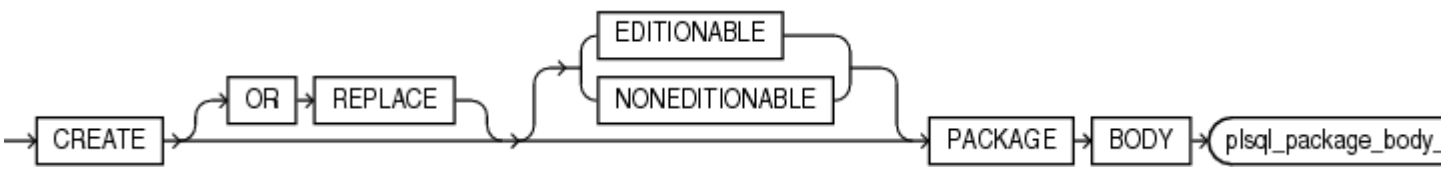
関連項目:

Oracle Database PL/SQL言語リファレンス

構文

パッケージ本体はPL/SQLを使用して定義されます。このため、このマニュアルの構文図ではSQLキーワードのみを示します。PL/SQLの構文、セマンティクスおよび例については、『Oracle Database PL/SQL言語リファレンス』を参照してください。

create_package_body::=



(plsql_package_body_source: 『Oracle Database PL/SQL言語リファレンス』を参照。)

セマンティクス

OR REPLACE

OR REPLACEを指定すると、既存のパッケージ本体を再作成できます。この句を指定した場合、パッケージに対して付与されていたオブジェクト権限を削除、再作成および再付与しなくても、既存のパッケージの本体を変更できます。パッケージ本体を変更した場合、その本体は自動的に再コンパイルされます。

再定義したパッケージに対して権限が付与されていたユーザーは、権限が再付与されなくてもそのパッケージにアクセスできます。

関連項目:

パッケージ本体の再コンパイルについては、「ALTER PACKAGE」を参照してください。

[EDITIONABLE | NONEDITIONABLE]

この句を省略すると、パッケージ本体は、EDITIONABLEまたはNONEDITIONABLEをパッケージ仕様部から継承します。この句を指定する場合は、パッケージ仕様部と一致させる必要があります。

plsql_package_body_source

plsql_package_body_sourceの構文およびセマンティクスについては、『Oracle Database PL/SQL言語リファレンス』を参照してください。

CREATE PFILE

目的

CREATE PFILE文を使用すると、バイナリのサーバー・パラメータ・ファイルまたは現行のインメモリーのパラメータ設定をテキストの初期化パラメータ・ファイルにエクスポートできます。テキストのパラメータ・ファイルの作成は、データベースで使用している現行のパラメータ設定リストを取得する便利な方法です。また、このファイルは、テキスト・エディタで簡単に編集でき、CREATE SPFILE文を使用してサーバー・パラメータ・ファイルに変換して戻すこともできます。

この文が正常に実行されると、サーバーにテキストのパラメータ・ファイルが作成されます。Oracle Real Application Clusters環境では、すべてのインスタンスのすべてのパラメータ設定が含まれます。サーバー・パラメータ・ファイルのパラメータ設定と同じ行のコメントも含まれます。

CDBにテキスト・パラメータ・ファイルを作成する場合のノート

マルチテナント・コンテナ・データベース(CDB)にテキスト・パラメータ・ファイルを作成する場合、現在のコンテナがルートまたはPDBである可能性があります。

- 現在のコンテナがルートである場合、データベースはルートのパラメータ設定を含むテキスト・ファイルを作成します。
- 現在のコンテナがPDBである場合、データベースはPDBのパラメータ設定を含むテキスト・ファイルを作成します。この場合、pfile_nameを指定する必要があります。

関連項目:

- サーバー・パラメータ・ファイルの詳細は、「CREATE SPFILE」を参照してください。
- テキストの初期化パラメータ・ファイルおよびバイナリのサーバー・パラメータ・ファイルの詳細は、『Oracle Database管理者ガイド』を参照してください。
- Oracle Real Application Clusters環境でのサーバー・パラメータ・ファイルの使用については、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください。

前提条件

この文を実行するには、次のいずれかのシステム権限が必要です。

7. SYSDBA
8. SYSDG
9. SYSOPER
10. SYSBACKUP
11. SYSASM
12. SYSRAC

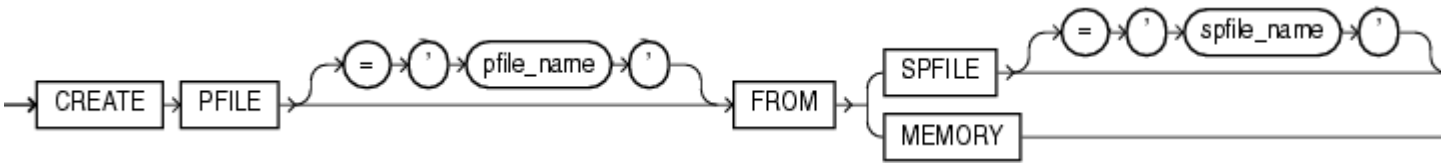
この文は、インスタンスの起動前と起動後のいずれかで実行できます。

制限事項

SYSDG、SYSOPERまたはSYSRACユーザーとしてOSファイルを上書きすることはできません。

構文

create_pfile::=



セマンティクス

pfile_name

作成するテキストのパラメータ・ファイル名を指定します。pfile_nameを指定しないと、プラットフォーム固有のデフォルトの初期化パラメータ・ファイル名が使用されます。pfile_nameには、パス接頭辞を含めることができます。パス接頭辞を指定しない場合は、データベースによってデフォルトの格納場所(プラットフォームによって異なる)のパス接頭辞が追加されます。

spfile_name

テキストのファイルを作成する元となるバイナリのサーバー・パラメータ・ファイル名を指定します。

- spfile_nameを指定する場合、そのファイルがサーバーに存在する必要があります。ファイルがオペレーティング・システムのサーバー・パラメータ・ファイルのデフォルト・ディレクトリに存在しない場合、フルパス名を指定する必要があります。
- spfile_nameを指定しない場合は、現在インスタンスに関連付けられているspfileが使用されます。通常、これは起動時に使用されたspfileです。インスタンスにspfileが関連付けられていない場合は、プラットフォーム固有のデフォルトのサーバー・パラメータ・ファイル名が検索されます。このファイルが存在しない場合は、エラーが戻されます。

関連項目:

デフォルトのパラメータ・ファイル名については、オペレーティング・システム固有のドキュメントを参照してください。

MEMORY

MEMORYを指定すると、現行のシステム全体のパラメータ設定を使用してpfileを作成できます。Oracle RAC環境では、作成されたファイルには各インスタンスからのパラメータ設定が含まれます。

例

パラメータ・ファイルの作成: 例

次の例は、バイナリのサーバー・パラメータ・ファイルs_params.oraからテキストのパラメータ・ファイルmy_init.oraを作成します。

```
CREATE PFILE = 'my_init.ora' FROM SPFILE = 's_params.ora';
```

ノート:

通常、オペレーティング・システムのパラメータ・ファイルには、フルパスのファイル名を指定する必要があります。パスについては、ご使用のオペレーティング・システムの Oracle マニュアルを参照してください。

CREATE PLUGGABLE DATABASE

目的

CREATE PLUGGABLE DATABASE文を使用すると、プラガブル・データベース(PDB)を作成できます。

この句で実行できるタスクは、次のとおりです。

- シードをテンプレートとして使用したPDBの作成

  create_pdb_from_seed句を使用すると、マルチテナント・コンテナ・データベース(CDB)内のシードをテンプレートとして使用して、PDBを作成できます。シードに関連付けられた各ファイルが新しい場所にコピーされ、その後これらのファイルが新しいPDBに関連付けられます。

- 既存のPDBまたはCDB以外のクローニングによるPDBの作成

  create_pdb_clone句を使用すると、既存のPDBまたはCDB以外をコピーし、その後このコピーをCDBに接続することによって、PDBを作成できます。既存のPDBまたはCDB以外に関連付けられたファイルは新しい場所にコピーされ、このコピーされたファイルが新しいPDBに関連付けられます。

- CDBへの切断されたPDBまたは非CDBの接続によるPDBの作成

  create_pdb_from_xml句を使用すると、XMLメタデータ・ファイルを使用して、切断されているPDBまたは非CDBをCDBに接続できます。

- 別のPDBを参照することによるプロキシPDBの作成。プロキシPDBは、参照先PDBへのフル機能アクセスを提供します。

  create_pdb_clone句を使用して、AS PROXY FROMを指定すると、プロキシPDBを作成できます。

- アプリケーション・コンテナ、アプリケーション・シードまたはアプリケーションPDBの作成

  create_pdb_from_seed、create_pdb_cloneまたはcreate_pdb_from_xml clauseを使用します。アプリケーション・コンテナを作成するには、AS APPLICATION CONTAINER句を指定する必要があります。アプリケーション・シードを作成するには、AS SEED句を指定する必要があります。

ノート:

新しい PDB が CDB で確立されるときに、新しい PDB によって提供されるサービスの名前が既存のサービス名と競合することがあります。競合が発生する可能性があるネームスペースは、その CDB へのアクセスを可能にするリスナーのネームスペースです。このネームスペース内で、非 CDB のデフォルト・サービス名、CDB のデフォルト・サービス名、PDB のデフォルト・サービス名およびユーザー定義サービス名の間で競合が発生することがあります。たとえば、同じコンピュータ・システム上に存在する複数の CDB が同じリスナーを使用しており、新たに確立される PDB に含まれるサービス名がこれらの CDB 内の別の PDB に含まれるサービス名と同じである場合、競合が発生します。

PDB を作成すると、競合する可能性のあるサービス名に新しい名前を指定できます。「service_name_convert」句を参照してください。PDB を作成した後でサービス名の競合が見つかった場合は、既存のサービス名との競合を引き起こす PDB に対する操作は試行しないでください。競合する名前が PDB のデフォルト・サービス名である場合は、その PDB の名前を変更する必要があります。競合する名前が PDB 内のユー

ザー作成サービス名である場合は、そのサービスを削除し、そのサービスのかわりに、競合しない名前を使用して、同じプロパティを持つ同じ用途のサービスを別に作成します。

関連項目:

- マルチテナント・アーキテクチャおよび概念の詳細は、Oracle Multitenant管理者ガイドを参照してください。
- PDBの変更および削除の詳細は、「ALTER PLUGGABLE DATABASE」および「DROP PLUGGABLE DATABASE」を参照してください。

前提条件

CDBに接続している必要があります。CDBをオープンし、READ WRITEモードにしておく必要があります。

PDBまたはアプリケーション・コンテナを作成する場合は、現在のコンテナがルートである必要があります。また、共通に付与されているCREATE PLUGGABLE DATABASEシステム権限が必要です。

アプリケーション・シードまたはアプリケーションPDBを作成する場合は、現在のコンテナがアプリケーション・ルートである必要があり、アプリケーション・コンテナをオープンしてREAD WRITEモードにしておく必要があります。また、CREATE PLUGGABLE DATABASEシステム権限(共通に付与されている権限か、そのアプリケーション・コンテナでローカルに付与されている権限のいずれか)が必要です。

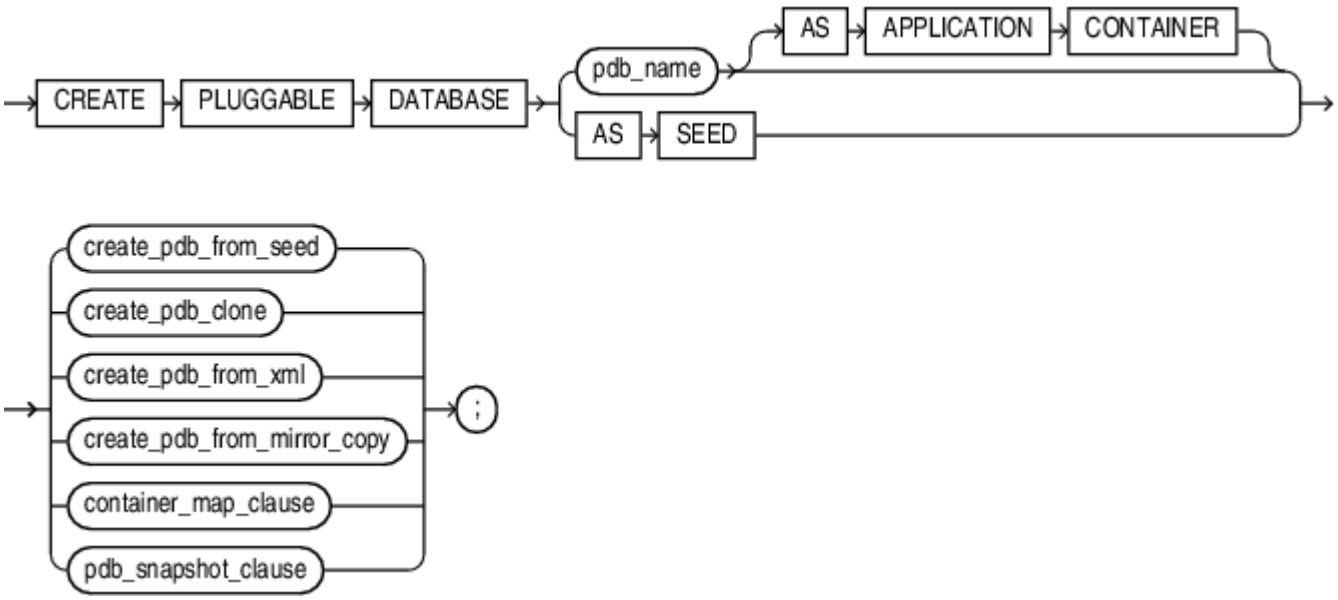
create_pdb_clone句を指定するには:

- src_pdb_nameが同じCDB内のPDBを指す場合は、新規PDBを作成するCDBのルートおよびクローンの作成元にするPDBにおけるCREATE PLUGGABLE DATABASEシステム権限が必要になります。
- src_pdb_nameがリモート・データベース内のPDBまたはCDB以外を指す場合は、新規PDBを作成するCDBのルートにおけるCREATE PLUGGABLE DATABASEシステム権限が必要になります。また、src_pdb_nameが指すPDBまたはCDB以外におけるCREATE PLUGGABLE DATABASEシステム権限を、リモート・ユーザーが保有している必要があります。

PDB作成の前提条件の詳細は、Oracle Multitenant管理者ガイドを参照してください。

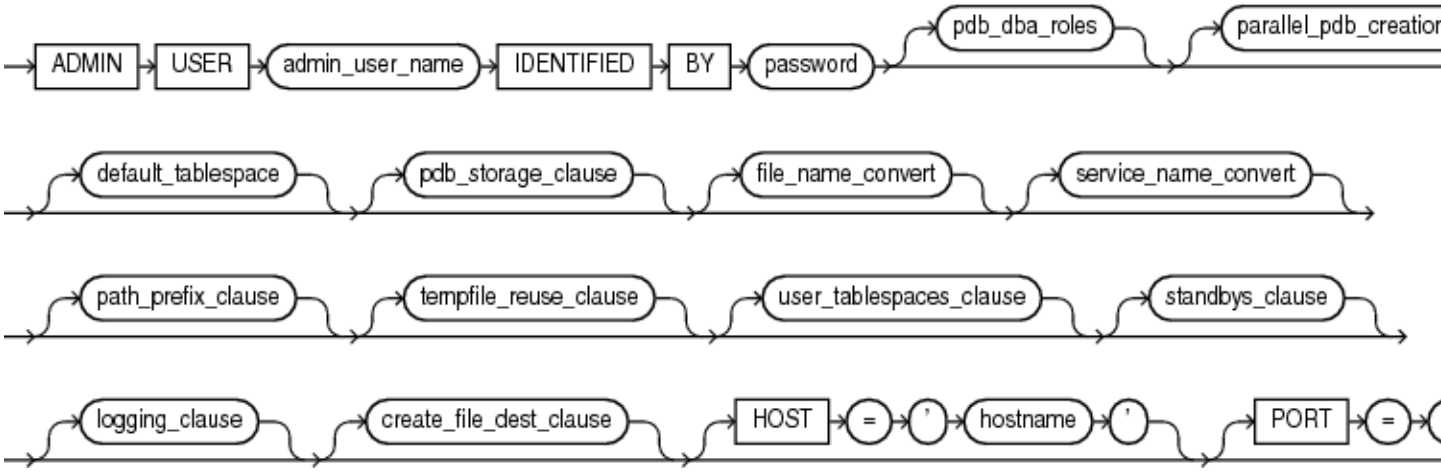
構文

create_pluggable_database::=



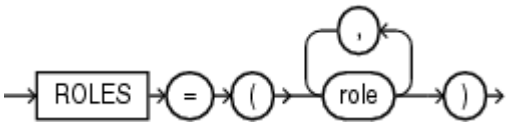
(create_pdb_from_seed::=、create_pdb_clone::=、create_pdb_from_xml::=)

create_pdb_from_seed::=

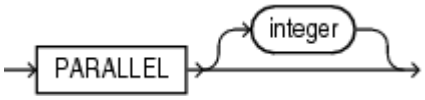


(pdb_dba_roles::=、parallel_pdb_creation_clause::=、default_tablespace::=、file_name_convert::=、service_name_convert::=、pdb_storage_clause::=、path_prefix_clause::=、tempfile_reuse_clause::=、user_tablespaces_clause::=、standbys_clause::=、logging_clause::=、create_file_dest_clause::=)

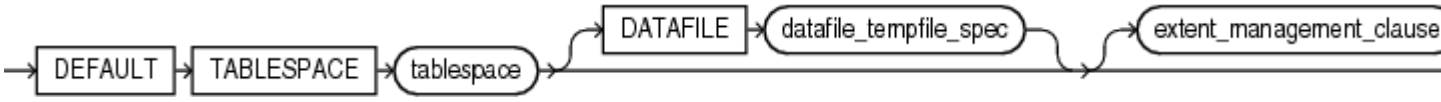
pdb_dba_roles::=



parallel_pdb_creation_clause::=

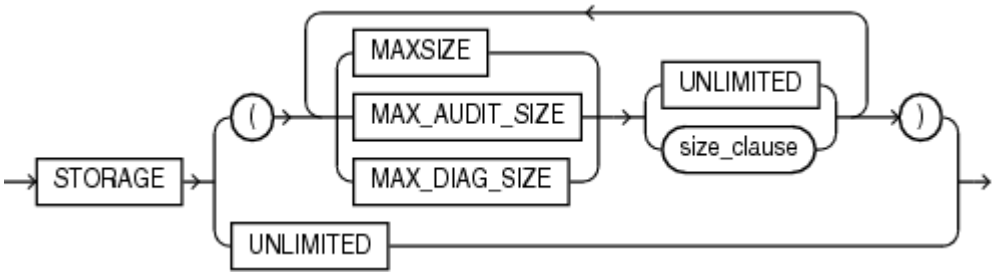


default_tablespace::=



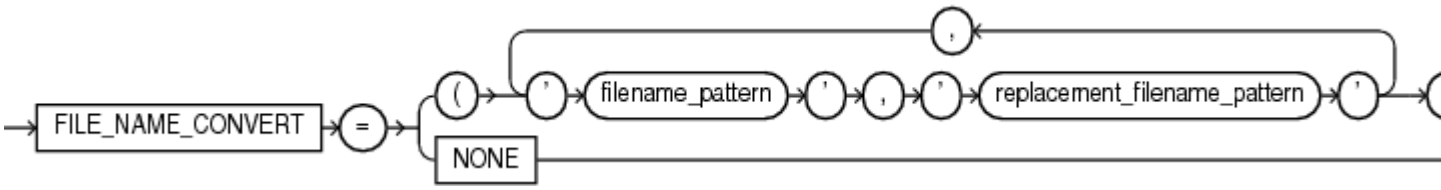
(datafile_tempfile_spec::=、extent_management_clause::=)

pdb_storage_clause::=

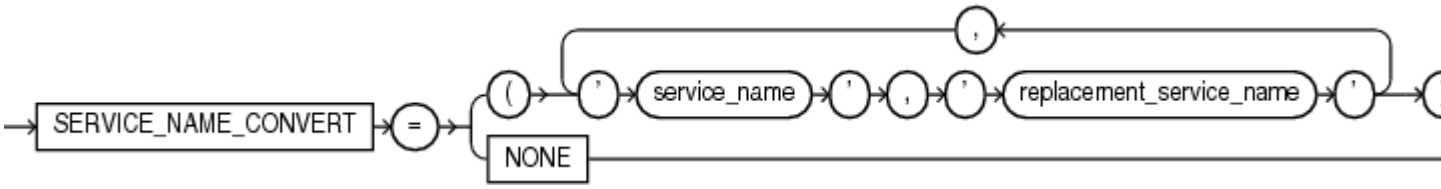


(size_clause::=)

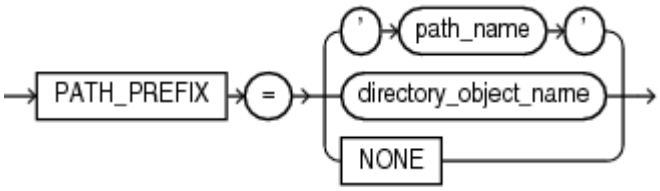
file_name_convert::=



service_name_convert::=

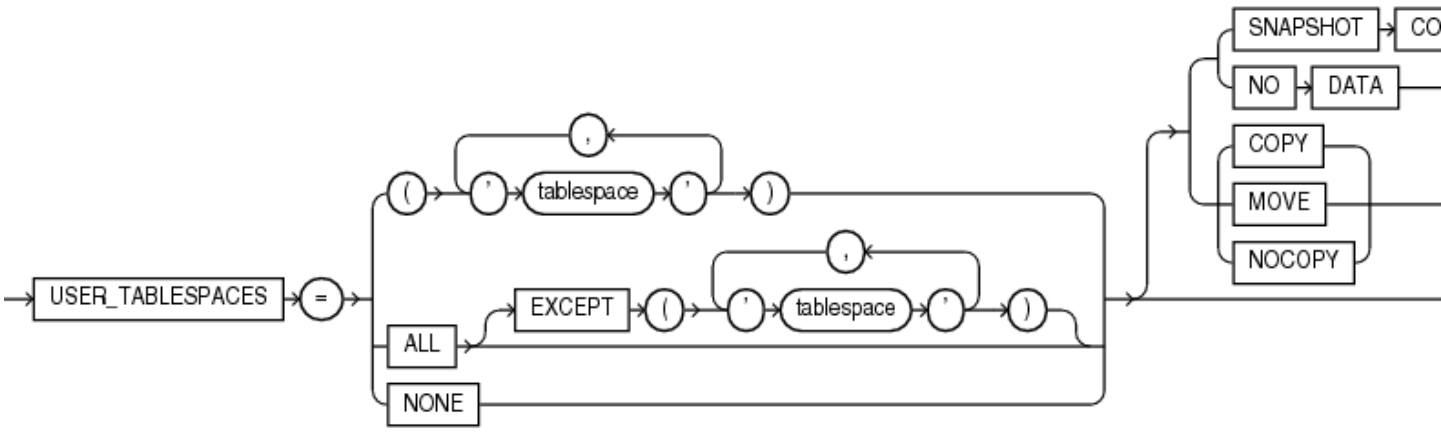


path_prefix_clause::=

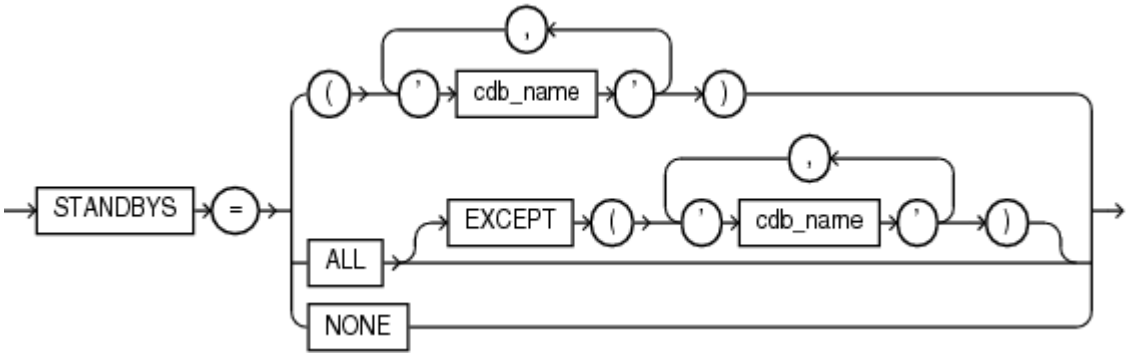


tempfile_reuse_clause::=

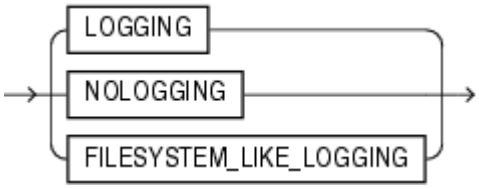
user_tablespaces_clause::=



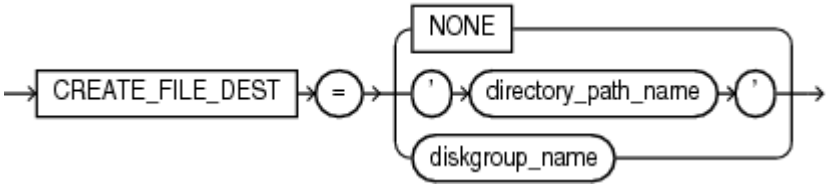
standbys_clause::=



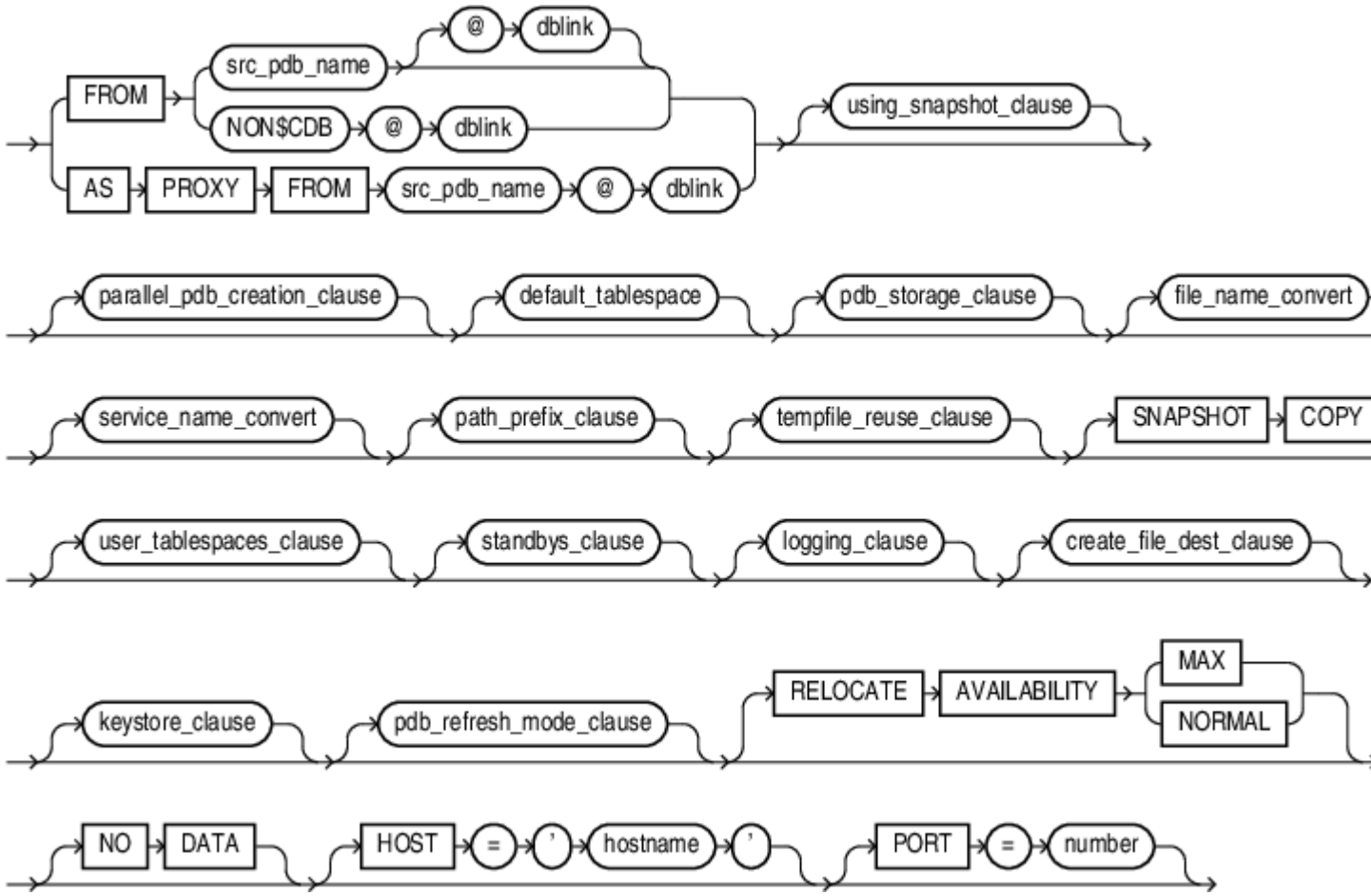
logging_clause::=



create_file_dest_clause::=

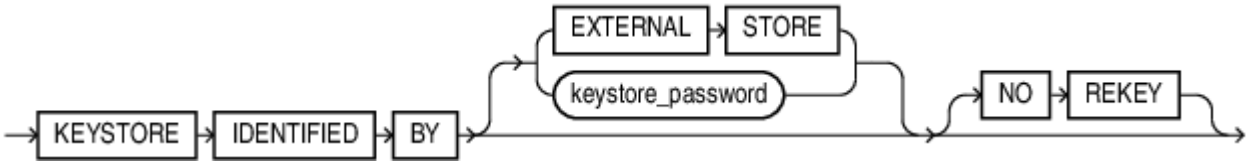


create_pdb_clone::=

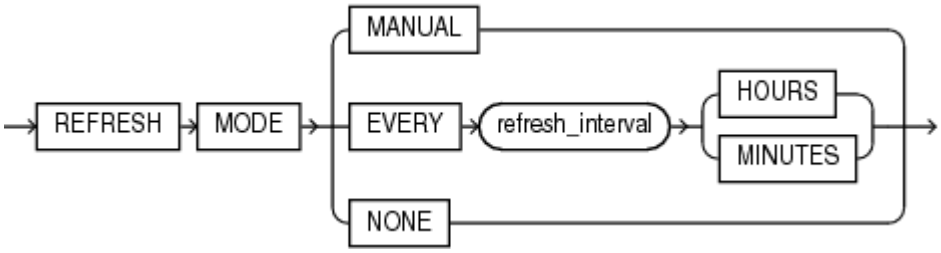


(parallel_pdb_creation_clause::=、default_tablespace::=、pdb_storage_clause::=、
file_name_convert::=、service_name_convert::=、path_prefix_clause::=、
tempfile_reuse_clause::=、user_tablespaces_clause::=、standbys_clause::=、logging_clause::=、
create_file_dest_clause::=、keystore_clause::=、pdb_refresh_mode_clause::=)

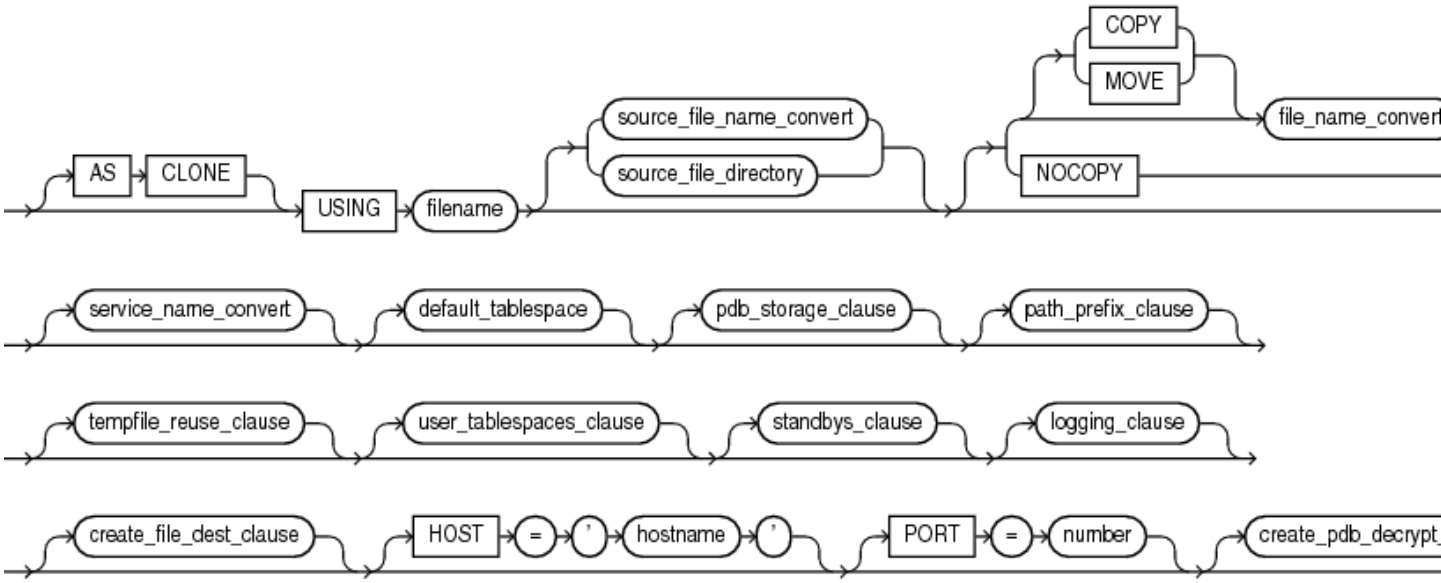
keystore_clause::=



pdb_refresh_mode_clause::=

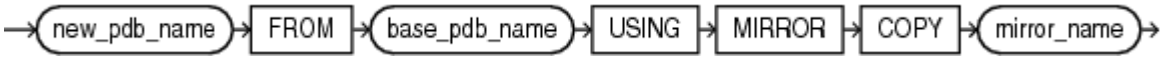


create_pdb_from_xml::=

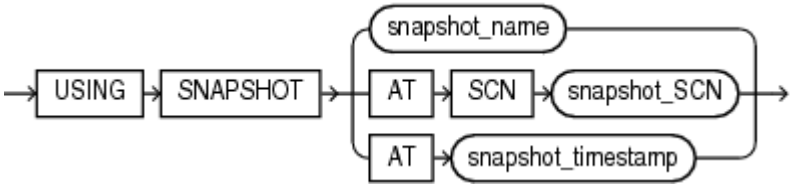


(source_file_name_convert::=、source_file_directory::=、file_name_convert::=、
service_name_convert::=、default_tablespace::=、pdb_storage_clause::=、path_prefix_clause::=、
tempfile_reuse_clause::=、user_tablespaces_clause::=、standbys_clause::=、logging_clause::=、
create_file_dest_clause::=)

create_pdb_from_mirror_copy::=

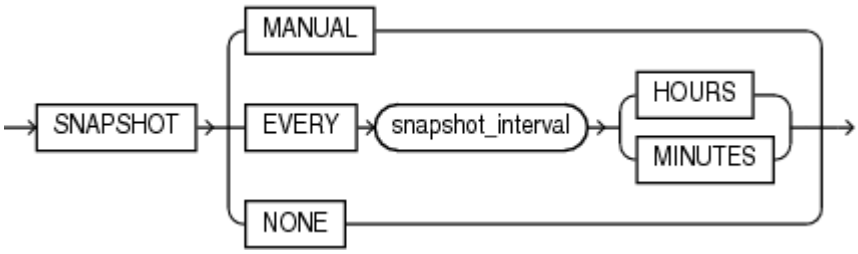


using_snapshot_clause ::=

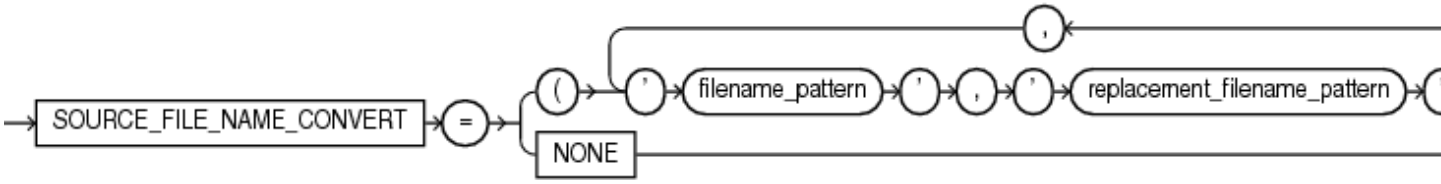


container_map_clause ::=

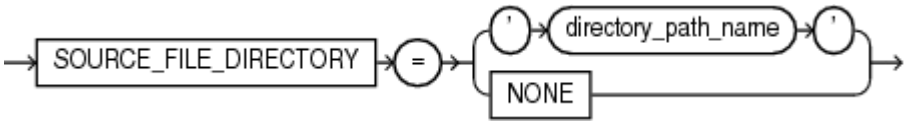
pdb_snapshot_clause ::=



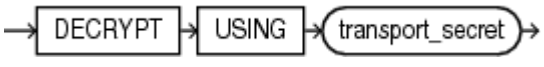
source_file_name_convert::=



source_file_directory::=



create_pdb_decrypt_from_xml::=



セマンティクス

pdb_name

作成するPDBの名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。PDB名の最初の文字はアルファベット文字にする必要があります。残りの文字には英数字またはアンダースコア(_)を使用できます。

PDB名はCDB内で一意にする必要があります。また、そのインスタンスが特定のリスナーを介してアクセスされるすべてのCDBのスコープ内で一意にする必要があります。

AS APPLICATION CONTAINER

この句を指定すると、アプリケーション・コンテナを作成できます。

関連項目:

アプリケーション・コンテナの作成ステップの詳細は、『Oracle Database管理者ガイド』を参照してください。

using_snapshot_clause

名前、SCNまたはタイムスタンプで識別できる既存のPDBスナップショットからPDBを作成する場合は、この句を指定します。

SNAPSHOT COPYを追加で指定する場合は、新しいPDBは指定するPDBスナップショットの有無に依存します。これは、ユーザーが新しいPDBを削除またはパージできるかどうかに影響します。

AS SEED

この句を指定すると、アプリケーション・シードを作成できます。データベースによってシードに`application_container_name$SEED`形式の名前が付けられます。

アプリケーション・コンテナは最大1つのアプリケーション・シードを持つことができます。アプリケーション・シードはオプションですが、存在する場合は、アプリケーション・コンテナの要件を満たすアプリケーションPDBを迅速に作成するために使用できます。アプリケーション・シードにより、そこから作成されるアプリケーションPDBを瞬時にプロビジョニングできます。

関連項目:

アプリケーション・シードの作成ステップの詳細は、『Oracle Database管理者ガイド』を参照してください。

create_pdb_from_seed

この句を使用すると、CDB内のシードをテンプレートとして使用して、PDBを作成できます。

関連項目:

シードを使用したPDBの作成ステップの詳細は、『Oracle Database管理者ガイド』を参照してください。

ADMIN USER

この句を使用して、PDBに対する管理タスクを実行するために必要な権限を持つ管理ユーザーを作成できます。`admin_user_name`に、作成するユーザーの名前を指定します。`IDENTIFIED BY`句を使用すると、`admin_user_name`のパスワードを指定できます。PDBにローカル・ユーザーが作成され、このユーザーに`PDB_DBA`ローカル・ロールが付与されます。

pdb_dba_roles

この句を使用すると、`PDB_DBA`ロールに1つ以上のロールを付与できます。この句を使用して、PDBの管理ユーザーが必要とする権限を付与してください。`role`には、事前定義されたロールを指定します。事前定義されているロールのリストは、『Oracle Databaseセキュリティ・ガイド』を参照してください。

また、`GRANT`文を使用して、PDBの作成後に`PDB_DBA`ロールにロールを付与することもできます。`PDB_DBA`ロールに適切な権限を付与するまでは、SYSユーザーおよびSYSTEMユーザーがPDB上で管理タスクを実行できます。

parallel_pdb_creation_clause

この句はCDBに、パラレル実行サーバーを使用して新しいPDBのデータファイルを新しい場所にコピーするよう指示します。これ

により、PDBの作成が高速化されることがあります。

PARALLEL

PARALLELを指定すると、CDBは使用するパラレル実行サーバーの数を自動的に選択します。COMPATIBLE初期化パラメータが12.2以上に設定されている場合、これがデフォルトです。

PARALLEL integer

integerを使用して、使用するパラレル実行サーバーの数を指定します。現在のデータベースのロードおよび使用可能なパラレル実行サーバーの数によっては、CDBはこの設定を無視できます。0または1を指定した場合、CDBはPDBの作成をパラレル化しません。これにより、PDBの作成時間が長くなることがあります。

default_tablespace

この句を指定すると、smallfile表領域が作成され、これがPDBのデフォルトの永続表領域として設定されます。デフォルトの表領域は、Oracle Databaseによって、別の永続表領域が指定されていないSYSTEM以外のユーザーに割り当てられます。default_tablespace句のセマンティクスは、CREATE DATABASE文と同じです。詳細は、「CREATE DATABASE」の「default_tablespace」を参照してください。

pdb_storage_clause

この句を使用して、PDBの記憶域制限値を指定します。

- MAXSIZEを使用すると、PDB内のすべての表領域で使用可能にする記憶域の容量を、size_clauseで指定した値に制限できます。この制限には、PDBに属する表領域のデータファイルと一時ファイルのサイズも含まれます。MAXSIZE UNLIMITEDを指定すると、容量は無制限になります。
- MAX_AUDIT_SIZEを使用して、PDBからあふれた統合監査OS (.bin形式)ファイルで使用できる記憶域の量を、size_clauseで指定された値に制限します。MAX_AUDIT_SIZE UNLIMITEDを指定すると、容量は無制限になります。
- MAX_DIAG_SIZEを使用して、PDBで使用できる自動診断リポジトリ(ADR)の診断(トレース・ファイルおよびインシデント・ダンプ)用の記憶域の量を、size_clauseで指定された値に制限します。MAX_DIAG_SIZE UNLIMITEDを指定すると、容量は無制限になります。

この句を省略するかSTORAGE UNLIMITEDを指定すると、PDBの記憶域には制限がなくなります。これは、STORAGE (MAXSIZE UNLIMITED MAX_AUDIT_SIZE UNLIMITED MAX_DIAG_SIZE UNLIMITED)を指定することと同じになります。

file_name_convert

この句を使用して、PDBのファイル(データファイルやウォレット・ファイルなど)の名前をデータベースでどのように作成するかを決定できます。

- filename_patternには、シードに関連付けられたファイル名(シードを使用してPDBを作成する場合)、ソースPDBに関連付けられたファイル名(PDBをクローニングする場合)またはXMLファイル内にリストされたファイル名(PDBをCDBに接続する場合)に含まれる文字列を指定します。
- replacement_filename_patternには、置換文字列を指定します。

新規PDBに関連付けられたファイル名の生成時に、filename_patternがreplacement_filename_patternで置換されます。

Oracle Managed Filesで管理されているファイルまたはディレクトリと一致するファイル名のパターンは指定できません。

FILE_NAME_CONVERT = NONEを指定できます。これは、この句を省略した場合と同じになります。この句を省略すると、データベースはまずOracle管理ファイルを使用してファイル名を生成しようとします。Oracle Managed Filesを使用している場合は、PDB_FILE_NAME_CONVERT初期化パラメータを使用してファイル名が生成されます。このパラメータが設定されていないと、エラーが発生します。

service_name_convert

この句を使用して、ソースPDBのサービス名に基づいて新しいPDBのユーザー定義サービス名を変更します。新しいPDBのサービス名がCDB内の既存のサービス名と競合する場合は、プラグイン違反が発生することがあります。この句を使用すると、これらの違反を回避できます。

- service_nameには、PDBシード(アプリケーション・シードを使用してアプリケーション・コンテナにPDBを作成する場合)またはソースPDB (PDBをクローニングする場合またはPDBをCDBに接続する場合)内のサービス名を指定します。
- replacement_service_nameには、このサービスの置換名を指定します。

Oracle Databaseでは、作成されるPDBのサービスに置換サービス名を使用します。

SERVICE_NAME_CONVERT = NONEを指定できます。これは、この句を省略した場合と同じになります。

service_name_convertの制限事項

service_name_convert句には、次の制限事項があります。

- PDBのデフォルトのサービス名は変更できません。デフォルト・サービスの名前はPDBと同じになります。
- CDBシードにはユーザー定義サービスがないため、create_pdb_from_seed句を使用して、CDBシードからPDBを作成する場合は、この句を指定できません。ただし、create_pdb_from_seed句を使用して、アプリケーション・シードからアプリケーションPDBを作成する場合は、この句を指定できます。

path_prefix_clause

この句を使用して、PDBに関連付けられたディレクトリ・オブジェクトのファイル・パスを、指定したディレクトリまたはそのサブディレクトリに確実に制限できます。また、この句により、PDBに関連付けられているファイル(CREATE PFILE文で作成されたファイル)は指定されたディレクトリ(PDBのOracle XMLリポジトリ)およびOracleウォレットのエクスポート・ディレクトリに制限されます。PDBの作成後は、この句を変更することはできません。この句は、Oracle Managed Filesによって作成されたファイルには影響しません。

8. path_nameには、オペレーティング・システムのディレクトリの絶対パス名を指定します。一重引用符が必要です。その結果、パス名の大/小文字が区別されます。Oracle Databaseでは、PDBに関連付けられているすべてのファイル・パスの接頭辞としてpath_nameが使用されます。

   相対パスが付加されたときに結果として生成されるパス名が正しい書式で形成されるようにpath_nameを指定してください。たとえば、UNIXシステムでは、次に示すようにpath_nameの最後にフォワード・スラッシュ(/)を使用するようにしてください。

   ```
   PATH_PREFIX = '/disk1/oracle/dba/salespdb/'
   ```

9. directory_object_nameには、CDBルート(CDB$ROOT)に存在するディレクトリ・オブジェクトの名前を指定します。ディレクトリ・オブジェクトは、PATH_PREFIXに使用する絶対パスを指します。
10. PATH_PREFIX = NONEを指定すると、PDBに関連付けられてたディレクトリ・オブジェクトの相対パスが絶対パスとして処理され、特定のディレクトリに制限されなくなります。

path_prefix_clauseの省略は、PATH_PREFIX = NONEの指定と同じです。

path_prefix_clauseがPDBに指定されると、PATH_PREFIX文字列がPDBのすべてのローカル・ディレクトリ・オブジェクトに接頭辞として常に追加されるため、既存のディレクトリ・オブジェクトが適切に機能しない場合があります。
path_prefix_clauseは、ユーザーが作成したディレクトリ・オブジェクトにのみ適用されます。これは、Oracleが提供するディレクトリ・オブジェクトには適用されません。

tempfile_reuse_clause

PDBを作成すると、一時ファイルが新しいPDBと関連付けられます。PDBの作成方法によっては、一時ファイルはすでに存在し、使用されている場合もあります。

TEMPFILE REUSEを指定すると、新しいPDBに関連付けられている一時ファイルがすでに存在する場合は、それをフォーマットして再利用できます。この句を指定しており、一時ファイルが存在しない場合は、データベースで一時ファイルが作成されます。

TEMPFILE REUSEを指定しない場合、新しいPDBに関連付ける一時ファイルがすでに存在すると、データベースからエラーが戻され、PDBは作成されません。

user_tablespaces_clause

この句を使用すると、新しいPDBで使用可能な表領域を指定できます。SYSTEM、SYSAUXおよびTEMP表領域はすべてのPDBで使用可能であり、この句に指定することはできません。

この句を使用すると、複数のスキーマのデータを異なるPDBに分けることができます。たとえば、非CDBをPDBに移動する場合、その非CDBに多数のスキーマがあって、それぞれ異なるアプリケーションをサポートしているとき、非CDBで各スキーマが別個の表領域を使用していたとすると、この句を使用して各スキーマに属するデータを別個のPDBに分けることができます。

- tablespaceを指定して、新しいPDBで表領域を使用可能にします。表領域を複数指定する場合は、カンマ区切りのリストにします。
- ALLを指定して、新しいPDBですべての表領域を使用可能にします。これはデフォルトです。
- ALL EXCEPTを指定して、指定された表領域を除くすべての表領域を新しいPDBで使用可能にします。
- NONEを指定して、SYSTEM、SYSAUXおよびTEMP表領域のみ新しいPDBで使用可能にします。

CDBの互換性レベルが12.2以上の場合、この句によって除外される表領域は新しいPDBにオフラインで作成され、これらにデータファイルは関連付けられません。CDBの互換性レベルが12.2未満の場合、この句で除外される表領域は新しいPDBでオフラインとなり、このような表領域に属するデータファイルはすべて無名でオフラインとなります。

{ SNAPSHOT COPY | NO DATA }

これらの句は、create_pdb_clone句でPDBをクローニングする場合にのみ適用されます。非CDBをクローニングする場合には適用されません。デフォルトでは、PDBのクローニングに指定された設定に応じて、新しいPDBで使用可能な各表領域が作成されます。これらの句により、これらの設定を次のようにオーバーライドできます。

- SNAPSHOT COPY - 記憶域スナップショットを使用して表領域をクローニングします。
- NO DATA - 表領域のデータ・モデル定義をクローニングしますが、表領域のデータはクローニングしません。

{ COPY | MOVE | NOCOPY }

これらの句は、create_pdb_from_xml句でPDBを接続する場合に適用されます。デフォルトでは、PDBの接続に指定された設定に応じて、新しいPDBで使用可能な各表領域が作成されます。これらの句により、これらの設定を次のようにオーバーライドできます。

- COPY - 表領域のファイルを新しい場所にコピーします。
- MOVE - 表領域のファイルを新しい場所に移動します。
- NOCOPY - 表領域のファイルを新しい場所にコピーまたは移動しません。

standbys_clause

この句を使用して、新しいPDBを1つ以上のスタンバイCDBに含めるかどうかを指定します。PDBをスタンバイCDBに含める場合、スタンバイ・リカバリ中に、スタンバイCDBはPDBのデータファイルを探します。データファイルが見つからない場合、スタンバイ・リカバリが停止するため、リカバリが再開する前にデータファイルを正しい場所にコピーする必要があります。

- cdb_nameを指定して、新しいPDBを指定したスタンバイCDBに含めます。カンマ区切りのリストで、複数のスタンバイCDB名を指定できます。
- ALLを指定して、すべてのスタンバイCDBの新しいPDBを含めます。これはデフォルトです。
- ALL EXCEPTを指定して、新しいPDBを指定したスタンバイCDB以外のすべてのスタンバイCDBに含めます。
- NONEを指定して、すべてのスタンバイCDBの新しいPDBを除外します。PDBがすべてのスタンバイCDBから除外される場合、PDBのデータファイルは名前がなく、すべてのスタンバイCDBでオフラインとマークされます。PDBのデータファイルがスタンバイに見つからない場合、スタンバイ・リカバリは停止しません。PDBの作成後に新しいスタンバイCDBをインスタンス化する場合、新しいスタンバイCDBのリカバリ用のPDBを明示的に無効化する必要があります。

データファイルを正しい場所にコピーし、PDBをオンラインにし、リカバリ用に有効とマークすることによって、スタンバイCDBから除外した後にスタンバイCDBのPDBを有効化できます。

logging_clause

この句を使用して、PDB内で作成する表領域のデフォルトのロギング属性を指定します。ロギング属性は、特定のDML操作がREDOログ・ファイルで記録されるかどうか(LOGGINGまたはNOLOGGING)を制御します。デフォルトはLOGGINGです。

表領域を作成する場合、CREATE TABLESPACE文のlogging_clauseを指定して、デフォルトのロギング属性をオーバーライドできます。

この句の詳細は、「logging_clause」を参照してください。

create_file_dest_clause

デフォルトでは、新しく作成されたPDBは、ルートからOracle Managed Files設定を継承します。ルートがOracle Managed Filesを使用する場合、PDBもOracle Managed Filesを使用します。PDBは、Oracle Managed Filesのベース・ファイル・システム・ディレクトリをルートと共有し、PDBのGUIDで名前が付けられた固有のサブディレクトリがあります。ルートがOracle Managed Filesを使用しない場合、PDBもOracle Managed Filesを使用しません。

この句を使用すると、デフォルトの動作をオーバーライドできます。PDBのOracle Managed Filesを有効化または無効化できます。PDBのファイルの別のベース・ファイル・システム・ディレクトリまたはOracle ASMディスク・グループを指定します。

- NONEを指定して、PDBのOracle Managed Filesを無効化します。
- directory_path_nameまたはdiskgroup_nameを指定して、PDBのOracle Managed Filesを有効化します。

  directory_path_nameを指定して、PDBのファイルのベース・ファイル・システム・ディレクトリを指定します。オペレーティング・システム・ディレクトリの完全パス名を指定します。ディレクトリが存在する必要があり、Oracleプロセスにはディレクトリに対する適切な権限が必要です。一重引用符が必要です。その結果、パス名の大/小文字が区別されます。

diskgroup_nameを指定して、PDBのファイルのデフォルトのOracle ASMディスク・グループを指定します。

NONE以外の値を指定する場合、データベースはPDBのSCOPE=SPFILEを使用したDB_CREATE_FILE_DEST初期化パラメータを暗黙的に設定します。

HOSTおよびPORT

これらの句は、プロキシPDBから参照するPDBを作成する場合にのみ役立ちます。このタイプのPDBは、参照先PDBと呼ばれます。

参照先PDBを作成する場合:

- リスナーの名前がPDBのホスト名と異なる場合は、HOST句を指定する必要があります。hostnameには、リスナーの完全修飾ドメイン名を指定します。hostnameは一重引用符で囲みます。たとえば、'myhost.example.com'です。

  Oracle Real Application Clusters (Oracle RAC)環境では、PDBのいずれかのホストをhostnameに指定できます。

- リスナーのポート番号が1521でない場合は、PORT句を指定する必要があります。numberには、リスナーのポート番号を指定します。

プロキシPDBは、データベース・リンクを使用して、その参照先PDBとの通信を確立します。通信が確立した後、プロキシPDBでは、データベース・リンクを使用しないで参照先PDBと直接通信します。プロキシPDBが正常に機能するためには、参照先PDBのリスナーのホスト名およびポート番号が正しい必要があります。

関連項目:

プロキシPDBの作成の詳細は、「create_pdb_clone」の「AS PROXY FROM」句を参照してください。

create_pdb_clone

この句を使用すると、ソースとターゲットPDBのクローニングによって、新しいPDBを作成できます。ソースは、ローカルCDBのPDB、リモートCDBのPDB、非CDBのいずれでもかまいません。ターゲットPDBはソースのクローンです。

ソースがローカルCDBのPDBである場合、ソースPDBを接続または切断できます。ソースがリモートCDBのPDBである場合、ソースPDBを接続する必要があります。

ソースがCDB以外またはリモートCDBのPDBである場合、ソースおよびターゲットPDBを含むCDBは次の要件を満たす必要があります。

- エンディアン形式が同じであること。
- 互換性のある文字セットおよび各国語文字セットが設定されていること。これは次のことを意味します。
  - ソース文字セット内の各文字がローカルCDBの文字セットで使用可能であること。
  - ソースの文字セット内の各文字のコード・ポイント値がローカルCDBの文字セットのコード・ポイント値と同じであること。
- これらには、同じセットのデータベース・オプションがインストールされている必要があります。

ソース非CDBまたはPDBのデフォルト一時表領域を使用していたPDBのユーザーは、新しいPDBのデフォルト一時表領域を使用します。非CDBまたはPDB内のデフォルト以外の一時表領域を使用していたユーザーは、新しいPDB内の同じローカル一時表領域を引き続き使用します。

同じコマンドを使用して、統合されたPDBまたは分離されたPDBをクローニングできます。唯一異なる点は、指定する必要のあるキーストア・パスワードが複数のキーストアのパスワードであることです。

PDBのホット・クローニング: 例

```
CREATE PLUGGABLE DATABASE CDB1_PDB2_CLONE FROM CDB1_PDB2
KEYSTORE IDENTIFIED BY keystore_password
```

統合されたPDBの場合:

- keystore_passwordは、ROOTキーストアのパスワードです。
- ウォレットがROOTでオープンしている必要があります。

分離されたPDBの場合:

- keystore_passwordは、PDB CDB1_PDB2_CLONEの新しいキーストアのパスワードです。
- ウォレットがCDB1_PDB2_CLONEでオープンしている必要があります。

PDBのクローニング: 例

統合されたPDB

```
CREATE PLUGGABLE DATABASE CDB1_PDB1_C AS CLONE USING '/tmp/cdb1_pdb3.pdb'
KEYSTORE IDENTIFED BY keystore_password DECRYPT USING transport_secret
```

- TDEが使用されている場合はウォレットがROOTでオープンしている必要があります。
- .pdbファイルにTDEキーが含まれている場合、KEYSTORE IDENTIFED BYとtransport_secretを指定する必要があります。
- keystore_passwordは、ROOTキーストアのパスワードです。

分離されたPDB

```
CREATE PLUGGABLE DATABASE CDB1_PDB2_C AS CLONE USING '/tmp/cdb1_pdb2.pdb'
```

- KEYSTORE IDENTIFED BYまたはtransport_secretを指定する必要はありません。指定されている場合は無視されます。
- ウォレットはROOT内で開いている必要はありません。

関連項目:

既存のPDBのクローニングによるPDBの作成ステップの詳細は、『Oracle Database管理者ガイド』を参照してください。

FROM

この句を使用して、ソースPDBまたはCDB以外を指定します。ソースに関連付けられた各ファイルが新しい場所にコピーされ、その後これらのファイルが新しいPDBに関連付けられます。

ソースPDB または非CDBはクローズできません。次のようにオープンできます。

- ソースPDB (ソースCDB)またはソース非CDBを含むCDBがARCHIVELOGモードおよびローカルUNDOモードの場合、ソースPDBまたはソース非CDBはREAD WRITEモードでオープンでき、クローニング操作中に完全に機能します。このことは、ホットPDBクローニングと呼ばれます。
- ソースCDBまたはソース非CDBがARCHIVELOGモードでない場合、ソースPDBまたは非CDBはREAD ONLYでオー

プンする必要があります。

ソースPDBまたは非CDBを次のように指定します。

- ソースがローカルCDBのPDBである場合、src_pdb_nameを使用して、ソースPDBの名前を指定します。src_pdb_nameにはPDB$SEEDは指定できません。かわりに、create_pdb_from_seed句を使用することによって、シードをテンプレートとして使用してPDBを作成します。
- ソースがリモートCDBのPDBである場合、src_pdb_nameを使用してソースPDBの名前を指定し、dblinkを使用してリモートCDBに接続するために使用するデータベース・リンクの名前を指定します。
- ソースがCDB以外の場合、NON$CDB@dblinkを指定します。dblinkは、CDB以外に接続するために使用するデータベース・リンクの名前です。

AS PROXY FROM

この句を使用すると、参照PDBと呼ばれる別のPDBを参照してプロキシPDBを作成できます。参照先PDBは、プロキシPDBと同じCDBまたは別のCDBに存在できます。ローカル・プロキシPDBは参照PDBと同じCDBに含まれており、リモート・プロキシPDBは参照PDBとは異なるCDBに含まれています。

src_pdb_name@dblinkに、参照PDBを指定します。

関連項目:

プロキシPDBの作成ステップの詳細は、『Oracle Database管理者ガイド』を参照してください。

default_tablespace

この句を使用すると、PDBのデフォルトの永続表領域を指定できます。デフォルトの表領域は、Oracle Databaseによって、別の永続表領域が指定されていないSYSTEM以外のユーザーに割り当てられます。ソースPDBまたは非CDBにtablespaceがすでに存在している必要があります。この表領域はすでに存在するため、create_pdb_clone句でPDBを作成する場合は、DATAFILE句またはextent_management_clauseを指定できません。

pdb_storage_clause

この句を使用して、新規PDBの記憶域制限値を指定します。この句のセマンティクスの詳細は、「pdb_storage_clause」を参照してください。

file_name_convert

この句を使用して、新しいPDBのファイルの名前をデータベースでどのように作成するかを決定できます。この句のセマンティクスの詳細は、「file_name_convert」を参照してください。

service_name_convert

この句を使用して、データベースで新しいPDBのサービス名を変更する方法を決定します。この句のセマンティクスの詳細は、「service_name_convert::=」を参照してください。

path_prefix_clause

この句を使用して、PDBに関連付けられたすべてのディレクトリ・オブジェクト・パスを、指定したディレクトリまたはそのサブディレクトリに確実に制限できます。この句のセマンティクスの詳細は、「path_prefix_clause」を参照してください。

tempfile_reuse_clause

TEMPFILE REUSEを指定すると、新しいPDBに関連付けられている一時ファイルがすでに存在する場合は、それをフォーマッ

トして再利用できます。このセマンティクスの詳細は、「tempfile_reuse_clause」を参照してください。

SNAPSHOT COPY

PDBをクローニングする場合のみ、SNAPSHOT COPYを指定できます。CDB以外をクローニングする場合、この句はサポートされていません。ソースPDBは、ローカルCDBとリモートCDBのどちらに存在してもかまいません。SNAPSHOT COPY句は、記憶域スナップショットを使用してソースPDBをクローニングするようデータベースに指示します。これにより、ソース・データファイルの完全なコピーを作成する必要がなくなるため、クローンを作成する時間が削減されます。

SNAPSHOT COPY句を使用してソースPDBのクローンを作成し、CLONEDB初期化パラメータをFALSEに設定する場合、ソースPDBのファイルの基になるファイル・システムが記憶域スナップショットをサポートしている必要があります。そのようなファイル・システムとして、Oracle Automatic Storage Management Cluster File System (Oracle ACFS)およびDirect NFS Client記憶域などがあります。

SNAPSHOT COPY句を使用してソースPDBのクローンを作成し、CLONEDB初期化パラメータをTRUEに設定する場合、ソースPDBのファイルの基になるファイル・システムはローカル・ファイル・システム、ネットワーク・ファイル・システム(NFS)またはDirect NFSを有効化したクラスタ・ファイル・システムの可能性があります。ただし、クローンが存在するかぎり、ソースPDBは読取り専用オープン・モードのままである必要があります。

Direct NFSクライアントにより、Oracle Databaseは、オペレーティング・システム・カーネルのNFSクライアントを使用するかわりに、ネットワーク接続記憶域(NAS)デバイスに直接アクセスできます。PDBファイルがDirect NFSクライアント記憶域に格納されている場合は、次の追加の要件が満たされている必要があります。

- ソースPDBファイルはNFSボリューム上に存在する必要があります。
- 記憶域の資格証明は、透過的データ暗号化キーストアに格納されている必要があります。
- 記憶域ユーザーは、ソースPDBファイルをホストするボリューム上でスナップショットを作成および破棄するのに必要な権限を持っている必要があります。
- 資格証明は、ADMINISTER KEY MANAGEMENT ADD SECRET SQL文を使用してキーストアに格納されている必要があります。

SNAPSHOT COPY句を使用してソースPDBのクローンを作成する場合は、クローンが存在するかぎり、次の制限がソースPDBに適用されます。

- 切断できません。
- 削除できません。

SNAPSHOT COPY句を使用して作成したPDBのクローンは、切断できません。削除のみ可能です。SNAPSHOT COPY句を使用して作成したクローンを切断しようとすると、エラーが発生します。

Oracle Real Application Clusters (Oracle RAC)環境でSNAPSHOT COPY句を使用して作成されたPDBの場合、PDBのファイルにアクセスする必要がある各ノードは、マウントされている必要があります。LinuxまたはUNIXプラットフォームで実行されているOracle RACデータベースには、基礎となるNFSボリュームがマウントされている必要があります。Oracle RACデータベースがWindowsプラットフォームで実行され、かつ、共有記憶域にDirect NFSを使用している場合は、作成したボリュームexportおよびmountのエントリを含むすべてのノード上のoranfstabファイルを更新する必要があります。

記憶域クローンには、新しいPDBのGUIDを使用して名前およびタグが付けられます。DBA_PDB_HISTORYビューのCLONETAG列を問い合せて、記憶域クローンのクローン・タグを表示できます。

keystore_clause

ソース・データベースに暗号化データまたはキーストア・セットがある場合は、この句を指定します。

別のPDBをクローニングしてPDBを作成する場合に、ソース・データベースに暗号化されたデータまたは設定されたTDEマスター暗号化キーがある場合、keystore_passwordにターゲット・キーストアのキーストア・パスワードを指定する必要があります。

ソース・データベースに暗号化データがあるかどうかを確認するには、DBA_ENCRYPTED_COLUMNSデータ・ディクショナリ・ビューまたはV$ENCRYPTED_TABLESPACES動的パフォーマンス・ビューを問い合せます。

keystore_passwordのかわりにEXTERNAL STORE句を使用して、統合されたキーストアを使用しているPDBをクローニングできます。このオプションを使用する前に、最初にTDE SEPSウォレットを構成する必要があります。

分離されたキーストアを使用しているPDBには、EXTERNAL STORE句を使用できません。

pdb_refresh_mode_clause

REFRESH MODE句は、PDBをクローニングする場合にのみ適用されます。ソースPDBはリモートCDBに含まれている必要があります。つまり、FROM src_pdb_name@dblink句を使用してソースPDBを指定する必要があります。

この句を使用すると、PDBのリフレッシュ・モードを指定できます。この句を使用すると、リフレッシュ可能PDBを作成できます。ソースPDBの変更は、リフレッシュ可能PDBに手動または自動で伝播できます。この操作はリフレッシュと呼ばれます。次のリフレッシュ・モードを指定できます。

- MANUAL - このモードでは、ALTER PLUGGABLE DATABASE REFRESH文を発行してリフレッシュ可能PDBをいつでも手動でリフレッシュできます。
- EVERY refresh_interval MINUTESまたはHOURS – このモードでは、データベースに対し、選択した時間単位(分または時間)でのrefresh_intervalごとに、リフレッシュ可能なPDBをリフレッシュするように指示します。MINUTESを選択した場合、refresh_intervalが3000未満である必要があります。HOURSを選択した場合、refresh_intervalが2000未満である必要があります。このモードでは、ALTER PLUGGABLE DATABASE REFRESH文を発行していつでもPDBを手動でリフレッシュすることもできます。
- NONE - このモードを指定した場合、クローンPDBはリフレッシュ可能PDBではありません。データベースでPDBを自動的にリフレッシュすることはできず、PDBを手動でリフレッシュすることもできません。このモードを指定した場合、後からPDBをリフレッシュ可能PDBに変更できません。これはデフォルトです。

リフレッシュ可能PDBは、READ ONLYモードでのみオープンできます。リフレッシュが発生するには、リフレッシュ可能PDBをクローズする必要があります。クローズされていない場合、手動リフレッシュを実行しようとするとエラーが発生します。クローズされていない場合、データベースで自動リフレッシュが試行されると、リフレッシュは次回のスケジュール・リフレッシュまで延期されます。

関連項目:

- PDBの手動リフレッシュの詳細は、「ALTER PLUGGABLE DATABASE」の「REFRESH」を参照してください。
- PDBのリフレッシュ・モードの変更の詳細は、「ALTER PLUGGABLE DATABASE」の「pdb_refresh_mode_clause」を参照してください。
- リフレッシュ可能PDBの詳細は、『Oracle Database管理者ガイド』を参照してください。

RELOCATE

この句を使用すると、CDB間でPDBを再配置できます。データベースはソースPDBをターゲットPDBにクローニングした後で、ソースPDBを削除します。また、データベースはPDBに関連付けられているファイルを新しい場所に移動させます。この操作によって、最小の停止時間で最も迅速にPDBを再配置できます。PDBの停止時間はほとんど、PDBのファイルを古い場所から新しい場所にコピーするのに要する時間です。再配置操作中、ソースPDBはREAD WRITEモードでオープンでき、すべての機

能を利用できます。

AVAILABILITYキーワードを使用して可用性レベルを指定できます。デフォルトの可用性はNORMALです。
AVAILABILITY MAXを指定すると、ソースとターゲットの間の永続接続でワークロードを円滑に移行するための追加の操作が実行されます。

create_pdb_clone句では、FROM src_pdb_name@dblink構文を使用して、ソースPDBの場所を指定する必要があります。src_pdb_nameには、ソースPDBの名前を指定します。dblinkには、ソースPDBの場所を示すデータベース・リンクを指定します。データベース・リンクは、PDBが再配置されるCDBに作成する必要があります。これは、リモートCDBのルートまたはリモートPDBのいずれかに接続できます。

関連項目:

PDBを再配置するステップの詳細は、『Oracle Database管理者ガイド』を参照してください。

NO DATA

PDBをクローニングする場合のみ、NO DATA句が適用されます。この句は、PDBのデータではなくソースPDBのデータ・モデル定義のクローニングを指定します。ソースPDBのディクショナリ・データはクローニングされますが、ソースPDBのユーザーが作成した表および索引データはすべて破棄されます。

NO DATA句の制限事項

NO DATA句には、次の制限事項があります。

- NO DATA句を使用してPDBをクローニングする場合は、ソースPDBを読取り専用モードでオープンする必要があります。
- CDB以外をクローニングする場合、NO DATAを指定できません。
- ソースPDBにクラスタ表、クラスタ化表、アドバンスト・キューイング(AQ)表、索引構成表または抽象データ型列が含まれる場合、NO DATAを指定できません。

HOSTおよびPORT

これらの句は、プロキシPDBから参照するPDBを作成する場合にのみ役立ちます。このタイプのPDBは、参照先PDBと呼ばれます。これらの句のセマンティクスの詳細は、「HOSTおよびPORT」を参照してください。

create_pdb_from_xml

この句を使用すると、切断されたPDBまたは非CDB(ソース・データベース)をCDB(ターゲットCDB)に接続することによって、PDBを作成できます。ソース・データベースが切断されたPDBである場合は、ターゲットCDBから切断されたものでも、別のCDBから切断されたものでもかまいません。

ソース・データベースとターゲットCDBが次の要件を満たしている必要があります。

- エンディアン形式が同じであること。
- 互換性のある文字セットおよび各国語文字セットが設定されていること。これは次のことを意味します。
  - ソース・データベースの文字セット内の各文字がターゲットCDBの文字セットで使用可能であること。
  - ソース・データベースの文字セット内の各文字のコード・ポイント値がターゲットCDBの文字セットのコード・ポイント値と同じであること。
- これらには、同じセットのデータベース・オプションがインストールされている必要があります。

関連項目:

- CDBへの切断されたPDBの接続によるPDBの作成ステップおよび非CDBを使用したPDBの作成ステップの詳細は、『Oracle Database管理者ガイド』を参照してください。
- DBMS_PDBパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

AS CLONE

この句は、同じデータファイル・セットを使用して作成されたPDBがターゲットCDBにすでに含まれている場合にのみ指定します。ソース・ファイルは切断されたPDBとして維持され、再使用可能です。また、AS CLONE句を指定すると、DBIDやGUIDなどの新しい識別子が新しいPDB用に確実に生成されるようになります。

USING

この句を使用すると、接続しているソース・データベースに関する情報を含むファイルを指定できます。filenameには、ファイルのフルパス名を指定します。次のいずれかの方法でこのファイルを取得できます。

- ソース・データベースが切断されたPDBである場合、ファイルはALTER PLUGGABLE DATABASEのpdb_unplug_clauseによって次のように作成されています。
  - ファイル名が拡張子.xmlで終わる場合は、PDBに関するメタデータを含むXMLファイルです。この場合、XMLメタデータ・ファイルとPDBのデータファイルはCDBにアクセスできる場所にある必要があります。
  - ファイル名が拡張子.pdbで終わる場合は、PDBアーカイブ・ファイルです。これは、PDBに関するメタデータを含むXMLファイルとPDBのデータファイルを含む、圧縮されたファイルです。PDBアーカイブ・ファイルは、CDBにアクセスできる場所に存在する必要があります。.pdbアーカイブ・ファイルを使用する場合は、PDBの接続時にこのファイルが抽出され、PDBのファイルは.pdbアーカイブ・ファイルと同じディレクトリに配置されます。したがって、source_file_directory句は必要ありません。
- ソース・データベースが非CDBの場合、DBMS_PDBパッケージを使用してXMLメタデータ・ファイルを作成する必要があり、XMLメタデータ・ファイルとソース非CDBのデータファイルがCDBにアクセスできる場所にある必要があります。

関連項目:

- ALTER PLUGGABLE DATABASEのpdb_unplug_clause
- DBMS_PDBパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

source_file_name_convert

XMLファイルの内容がソース・ファイルの場所を正確に記述していない場合にのみ、この句を指定してください。ソース・データベースを接続するために使用する必要があるファイルが、XMLファイルに指定されている場所に今ではもう存在しない場合は、この句を使用して、指定されているファイル名を実際のファイル名にマップします。

- filename_patternには、XMLファイルで指定されたファイルの場所の文字列を指定します。
- replacement_filename_patternには、PDBを作成するために使用する必要のあるファイルが含まれる実際の場所の文字列を指定します。

ソース・データベース・ファイルの検索時に、filename_patternがreplacement_filename_patternに置換されます。

Oracle Managed Filesで管理されているファイルまたはディレクトリと一致するファイル名のパターンは指定できません。

PDBの作成に使用する必要のあるファイルが、XMLファイルで指定された場所に存在するときには、この句を省略してもかまいません。また、SOURCE_FILE_NAME_CONVERT=NONEを指定することもできます。

source_file_directory

XMLファイルの内容がソース・ファイルの場所を正確に記述しておらず、かつソース・ファイルがすべて単一ディレクトリに存在する場合にのみ、この句を指定してください。この句は、大量のデータ・ファイルがあり、source_file_name_convert句を使用した各ファイルの置換ファイル名パターンの指定が現実的でない場合に便利です。

- directory_path_nameには、ソース・ファイルを含むディレクトリの絶対パスを指定します。切断されたPDBのXMLファイルに基づいて適切なファイルを検索するためにディレクトリがスキャンされます。

この句は、Oracle Managed Filesを使用する構成とOracle Managed Filesを使用しない構成に対して指定できます。

PDBの作成に使用する必要のあるファイルが、XMLファイルで指定された場所に存在するときには、この句を省略してもかまいません。また、SOURCE_FILE_DIRECTORY=NONEを指定することもできます。

COPY

XMLファイルにリストされたファイルを新しい場所にコピーして、新規PDB用に使用する場合は、COPYを指定します。これはデフォルトです。オプションのfile_name_convert句を使用して、新しいファイル名でパターンの置換を使用することができます。この句のセマンティクスの詳細は、「file_name_convert」を参照してください。

MOVE

XMLファイルにリストされたファイルを新しい場所にコピーではなく移動して、新規PDB用に使用する場合は、MOVEを指定します。オプションのfile_name_convert句を使用して、新しいファイル名でパターンの置換を使用することができます。この句のセマンティクスの詳細は、「file_name_convert」を参照してください。

記憶域の場所が異なるマウントである場合、または記憶域の場所がOSレベルまたは記憶域レベルで移動をサポートしていない場合、MOVE句によって最初にファイルがコピーされ、元のファイルが削除されます。

NOCOPY

PDBのファイルを現在の場所に残す場合は、NOCOPYを指定します。PDBを接続するために、ファイルをコピーしたり移動したりする必要がない場合は、この句を使用してください。

service_name_convert

この句を使用して、データベースで新しいPDBのサービス名を変更する方法を決定します。この句のセマンティクスの詳細は、「service_name_convert::=」を参照してください。

default_tablespace

この句を使用すると、PDBのデフォルトの永続表領域を指定できます。デフォルトの表領域は、Oracle Databaseによって、別の永続表領域が指定されていないSYSTEM以外のユーザーに割り当てられます。tablespaceがすでにソース・データベースに存在している必要があります。この表領域はすでに存在するため、create_pdb_from_xml句でPDBを作成する場合は、DATAFILE句またはextent_management_clauseを指定できません。

pdb_storage_clause

この句を使用して、新規PDBの記憶域制限値を指定します。この句のセマンティクスの詳細は、「pdb_storage_clause」を参照してください。

path_prefix_clause

この句を使用して、PDBに関連付けられたすべてのディレクトリ・オブジェクト・パスを、指定したディレクトリまたはそのサブディレクトリに確実に制限できます。この句のセマンティクスの詳細は、「path_prefix_clause」を参照してください。

tempfile_reuse_clause

TEMPFILE REUSEを指定すると、新しいPDBに関連付けられている一時ファイルがすでに存在する場合は、それをフォーマットして再利用できます。このセマンティクスの詳細は、「tempfile_reuse_clause」を参照してください。

HOSTおよびPORT

これらの句は、プロキシPDBから参照するPDBを作成する場合にのみ役立ちます。このタイプのPDBは、参照先PDBと呼ばれます。これらの句のセマンティクスの詳細は、「HOSTおよびPORT」を参照してください。

create_pdb_from_mirror_copy

ミラー・コピー mirror_nameの準備ファイルを使用してプラガブル・データベースnew_pdb_nameを作成するには、この句を指定します。prepare_clauseによって作成された準備ファイルを使用して、ソース・データベースから新しいPDBが分割されます。

- この句はルート・コンテナから実行する必要があります。
- その他のオプション・パラメータの意味がこの句によって変更されることはありません。
- 準備済ミラー・コピーからは1つのデータベースのみを分割できます。追加の分割を作成する場合は、新しいミラー・コピーを準備する必要があります。

using_snapshot_clause

名前、SCNまたはタイムスタンプで識別できる既存のPDBスナップショットを使用してPDBを作成する場合は、この句を指定します。

SNAPSHOT COPYを指定してPDBを作成する場合、新しいPDBはPDBスナップショットの有無に応じて異なります。これは、ユーザーがPDBを削除またはパージできるかどうかに影響します。

container_map_clause

新しいPDBに対する変更が発生するたびに動的に変更を更新するには、この句をCDBルートとアプリケーション・ルートのいずれかまたは両方に指定します。

コンテナ・マップについて次の点に注意する必要があります。

- container_map_clauseはオプションです。
- add_partition_clauseは、新しいPDBのルート(CDBルートまたはアプリケーション・ルート(あるいはその両方))で定義されているコンテナ・マップに新しいパーティションを追加します。
- split_partition_clauseは、新しいPDBのルート(CDBルートまたはアプリケーション・ルート(あるいはその両方))で定義されているコンテナ・マップの既存のパーティションを分割します。
- add_partition_clauseとsplit_partition_clauseがない場合は、新しいPDBのルートで定義されているコンテナ・マップは更新されません。
- PDBの再配置では、ソースPDBのルート(CDBルートまたはアプリケーション・ルート(あるいはその両方))で定義されて

いるコンテナ・マップが自動的に更新され、ソースPDBの「削除」が反映されます。

- ハッシュ・パーティション化を使用して定義されたコンテナ・マップの動的なメンテナンスはサポートされていません

レンジ・パーティション・コンテナ・マップへの新規パーティションの追加: 例

```
  CREATE PLUGGABLE DATABASE cdb1_pdb3
    ADMIN USER IDENTIFIED BY manager
    FILE_NAME_CONVERT=('cdb1_pdb0, cdb1_pdb3')
    CONTAINER_MAP UPDATE (ADD PARTITION cdb1_pdb3 VALUES LESS THAN (100));
    ALTER PLUGGABLE DATABASE cdb1_pdb3 OPEN
```

レンジ・パーティション・コンテナ・マップの既存のパーティションを分割して新規パーティションを作成: 例

```
  CREATE PLUGGABLE DATABASE cdb1_pdb4
    ADMIN USER IDENTIFIED BY manager
    FILE_NAME_CONVERT=('cdb1_pdb0, cdb1_pdb4')
    CONTAINER_MAP UPDATE (SPLIT PARTITION cdb1_pdb3
                            AT (50)
                            INTO
                            (PARTITION cdb1_pdb3, PARTITION cdb1_pdb3)
    ALTER PLUGGABLE DATABASE cdb1_pdb4 OPEN
```

レンジ・パーティション・コンテナ・マップでの更新の確認: 例

```
  SELECT partition_name, high_value
    FROM dba_tab_partitions
    WHERE table_name='MAP' AND table_owner='SYS'
```

pdb_snapshot_clause

PDBスナップショットを作成できるようにする場合は、この句を指定します。

- デフォルトはNONEです。これは、PDBのスナップショットを作成できないことを意味します。
- MANUALは、PDBスナップショットが手動でのみ作成可能であることを意味します。
- スナップショット間隔が指定されている場合、PDBスナップショットは指定した間隔で自動的に作成されます。また、ユーザーがPDBスナップショットを手動で作成できるようにもなります。
- 分単位で表される場合、snapshot_intervalは3000未満である必要があります。
- 時間単位で表される場合、snapshot_intervalは2000未満である必要があります。

create_pdb_decrypt_from_xml

このコマンドを実行するにはSYSKM権限が必要です。

統合モードのPDBの場合、次の制限事項が適用されます。

- TDEで保護されているデータベースを使用している場合は、この句を指定する必要があります。それ以外の場合はオプションです。
- 分離されたPDBの場合はこの句を指定する必要はありません。
- ウォレットがROOTでオープンしている必要があります。
- ウォレット・ファイルは、すべてのケース(NOCOPY、COPYおよびMOVE)でコピーされます。

XMLメタデータ・ファイルからPDBの接続: 例

```
CREATE PLUGGABLE DATABASE CDB1_PDB2 USING '/tmp/cdb1_pdb2.xml' NOCOPY
KEYSTORE IDENTIFIED BY keystore_password DECRYPT USING transport_secret
```

アーカイブ・ファイルからのPDBの接続: 例

```
CREATE PLUGGABLE DATABASE CDB1_PDB1_1_C USING '/tmp/cdb1_pdb3.pdb' DECRYPT USING
transport_secret
```

分離モードのPDBの場合、DECRYPT USING transport_secretを指定する必要はありません。切断されたPDBをXMLファイルから作成するときにウォレット・ファイルがコピーされるため、これは必須ではありません。.pdb拡張子が付いたアーカイブ・ファイルからPDBを作成する場合、PDBのウォレット・ファイルがzip圧縮アーカイブに含まれます。

ewallet.p12ファイルがすでに宛先に存在している場合、バックアップが自動的に開始されます。バックアップ・ファイルの形式はewallet_PLGDB_2017090517455564.p12です。

例

シードを使用したPDBの作成: 例

次の文は、CDBのシードをテンプレートとして使用することで、PDB salespdbを作成します。管理ユーザーsalesadmが作成され、このユーザーにdbaロールが付与されます。永続表領域が割り当てられていないSYSTEM以外のユーザーには、デフォルト表領域salesが割り当てられます。新しいPDBのファイル名は、シードのファイル名の/disk1/oracle/dbs/pdbseed/を/disk1/oracle/dbs/salespdb/に置換することによって構成されます。salesに属するすべての表領域は2Gを超えてはいけません。salespdbに関連付けられているすべてのディレクトリ・オブジェクト・パスの場所が、/disk1/oracle/dbs/salespdb/ディレクトリに制限されます。

```
CREATE PLUGGABLE DATABASE salespdb
  ADMIN USER salesadm IDENTIFIED BY password
  ROLES = (dba)
  DEFAULT TABLESPACE sales
    DATAFILE '/disk1/oracle/dbs/salespdb/sales01.dbf' SIZE 250M AUTOEXTEND ON
  FILE_NAME_CONVERT = ('/disk1/oracle/dbs/pdbseed/',
                       '/disk1/oracle/dbs/salespdb/')
  STORAGE (MAXSIZE 2G)
  PATH_PREFIX = '/disk1/oracle/dbs/salespdb/';
```

既存のPDBからのPDBのクローニング: 例

次の文は、PDB salespdbをクローニングすることによって、PDB newpdbを作成します。PDB newpdbとsalespdbは同じCDB内に存在します。記憶域の制限値を明示的に指定していないため、newpdbの容量には制限がなくなります。ファイルは/disk1/oracle/dbs/salespdb/から/disk1/oracle/dbs/newpdb/にコピーされます。newpdbに関連付けられているすべてのディレクトリ・オブジェクト・パスの場所が、/disk1/oracle/dbs/newpdb/ディレクトリに制限されます。

```
CREATE PLUGGABLE DATABASE newpdb FROM salespdb
  FILE_NAME_CONVERT = ('/disk1/oracle/dbs/salespdb/', '/disk1/oracle/dbs/newpdb/')
  PATH_PREFIX = '/disk1/oracle/dbs/newpdb';
```

CDBへのPDBの接続: 例

次の文は、以前に切断されたPDB salespdbをCDBに接続します。salespdbについて記述するメタデータの詳細は、XMLファイル/disk1/usr/salespdb.xmlに格納されています。XMLファイルでは、ファイルの現在の場所が正確に記述されていません。したがって、SOURCE_FILE_NAME_CONVERT句は、ファイルが/disk1/oracle/dbs/salespdb/ではなく/disk2/oracle/dbs/salespdb/にあることを示すために使用されます。NOCOPY句は、ファイルがすでに適切な場所に存在することを示します。salesに属するすべての表領域は2Gを超えてはいけません。ターゲットの場所に、XMLファイルで指定されている一時ファイルと同じ名前のファイルがあります。したがって、TEMPFILE REUSE句が必要です。

```
CREATE PLUGGABLE DATABASE salespdb
  USING '/disk1/usr/salespdb.xml'
```

```
  SOURCE_FILE_NAME_CONVERT =
    ('/disk1/oracle/dbs/salespdb/', '/disk2/oracle/dbs/salespdb/')
  NOCOPY
  STORAGE (MAXSIZE 2G)
  TEMPFILE REUSE;
```

CREATE PROCEDURE

目的

プロシージャはPL/SQLを使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

CREATE PROCEDURE文を使用すると、スタンドアロンのストアド・プロシージャまたはコール仕様を作成できます。

プロシージャとは、名前によってコールできるPL/SQL文の集合です。コール仕様は、SQLおよびPL/SQLからコールできるように、Javaメソッドまたは第三世代言語(3GL)ルーチンを宣言します。コール仕様は、コールされたときに起動するJavaメソッドをOracle Databaseに指示します。また、引数および戻り値に対して実行する型変換もデータベースに指示します。

ストアド・プロシージャには、開発、整合性、セキュリティ、パフォーマンスおよびメモリー割当ての面でいくつかのメリットがあります。

関連項目:

- ストアド・プロシージャのコール方法など、ストアド・プロシージャの詳細および外部プロシージャの登録については、『Oracle Database開発ガイド』を参照してください。
- プロシージャと類似点が多いファンクションの詳細は、「CREATE FUNCTION」を参照してください。
- パッケージの作成の詳細は、「CREATE PACKAGE」を参照してください。CREATE PROCEDURE文では、スタンドアロンのスキーマ・オブジェクトとしてプロシージャが作成されます。また、プロシージャをパッケージの一部としても作成できます。
- スタンドアロン・プロシージャの変更および削除については、「ALTER PROCEDURE」および「DROP PROCEDURE」を参照してください。
- 共有ライブラリについては、「CREATE LIBRARY」を参照してください。

前提条件

自分のスキーマ内にプロシージャを作成または再作成する場合は、CREATE PROCEDUREシステム権限が必要です。他のユーザーのスキーマ内にプロシージャを作成または再作成する場合は、CREATE ANY PROCEDUREシステム権限が必要です。

コール仕様を起動する場合、その他の権限が必要になることがあります。たとえば、Cコール仕様には、Cライブラリに対するEXECUTEオブジェクト権限が必要です。

Oracleプリコンパイラ・プログラム内にCREATE PROCEDURE文を埋め込む場合、キーワードEND-EXECに続けて、各言語の埋込みSQL文の終了記号を記述して文を終了する必要があります。

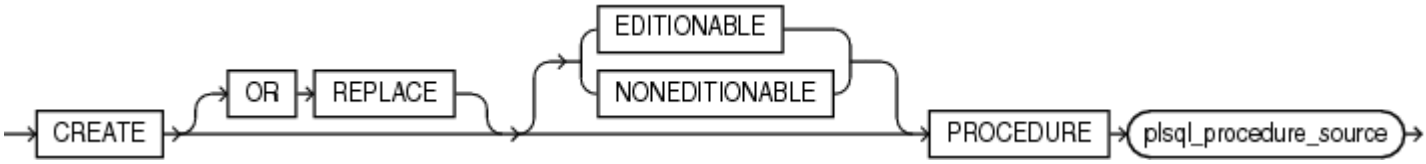
関連項目:

詳細は、『Oracle Database PL/SQL言語リファレンス』または『Oracle Database Java開発者ガイド』を参照してください。

構文

プロシージャはPL/SQLを使用して定義されます。このため、このマニュアルの構文図ではSQLキーワードのみを示します。PL/SQLの構文、セマンティクスおよび例については、『Oracle Database PL/SQL言語リファレンス』を参照してください。

create_procedure::=



(plsql_procedure_source: 『Oracle Database PL/SQL言語リファレンス』を参照。)

セマンティクス

OR REPLACE

OR REPLACEを指定すると、既存のプロシージャを再作成できます。この句を指定した場合、プロシージャに付与されているオブジェクト権限を削除、再作成および再付与しなくても、既存のプロシージャの定義を変更できます。プロシージャを再定義した場合、そのプロシージャは自動的に再コンパイルされます。

再定義したプロシージャに対して権限が付与されていたユーザーは、権限が再付与されなくてもそのプロシージャにアクセスできます。

ファンクション索引がプロシージャに依存している場合、索引にDISABLEDのマークが付きます。

関連項目:

プロシージャの再コンパイルについては、「ALTER PROCEDURE」を参照してください。

[EDITIONABLE | NONEDITIONABLE]

この句を使用すると、schemaのスキーマ・オブジェクト・タイプPROCEDUREのエディションニングが有効化されたときに、そのプロシージャをエディション・オブジェクトにするか非エディション・オブジェクトにするかを指定できます。デフォルトは、EDITIONABLEです。エディション・オブジェクトと非エディション・オブジェクトの詳細は、『Oracle Database開発ガイド』を参照してください。

plsql_procedure_source

plsql_procedure_sourceの構文およびセマンティクスについては、『Oracle Database PL/SQL言語リファレンス』を参照してください。

CREATE PROFILE

ノート:

リソース制限を設定する場合、この SQL 文ではなく、データベース・リソース・マネージャを使用することをお薦めします。データベース・リソース・マネージャを使用すると、リソース使用の管理および監査を柔軟に行うことができます。データベース・リソース・マネージャの詳細は、『Oracle Database 管理者ガイド』を参照してください。

目的

CREATE PROFILE文を使用すると、プロファイルを作成できます。プロファイルとは、データベース・リソースの制限の集合です。プロファイルをユーザーに割り当てた場合、そのユーザーはそれらの制限を超えることはできません。

次の方法でユーザーに対するリソース制限を指定します。

- ALTER SYSTEM文または初期化パラメータRESOURCE_LIMITで動的にリソース制限を使用可能にします。このパラメータは、パスワード・リソースには適用されません。パスワード・リソースは、常に使用可能です。
- CREATE PROFILE文を使用して、制限を定義するプロファイルを作成します。
- CREATE USERまたはALTER USER文を使用して、プロファイルを割り当てます。

マルチテナント環境では、ルートおよびPDBの共通ユーザーに異なるプロファイルを割り当てることができます。共通ユーザーがPDBにログインすると、設定がセッションに適用されるプロファイルは、設定がパスワード関連であるかリソース関連であるかに応じて異なります。

- パスワード関連のプロファイル設定は、ルートの共通ユーザーに割り当てられたプロファイルからフェッチされます。 たとえば、共通プロファイルc##prof (FAILED_LOGIN_ATTEMPTSを1に設定)をルートの共通ユーザーc##adminに割り当てるとします。PDBでは、そのユーザーにローカル・プロファイルlocal_prof(FAILED_LOGIN_ATTEMPTSが6に設定されている)が割り当てられます。 共通ユーザーc##adminが、loc_profが割り当てられているPDBにログインしようとする際、許容されるログイン試行失敗回数は1回のみになります。
- ルートの共通ユーザーに割り当てられたプロファイルのリソース関連の設定を参照せずに、PDBのユーザーに割り当てられたプロファイルに指定されたリソース関連のプロファイル設定が使用されます。たとえば、PDBのユーザーc##adminに割り当てられているプロファイルlocal_profでSESSIONS_PER_USERが2に設定されている場合、ルートのこのユーザーに割り当てられているプロファイルのこの設定の値に関係なく、PDB loc_profにログインするときに、c##adminは2回の同時セッションのみが許可されます。

関連項目:

パスワード管理およびパスワード保護の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

前提条件

プロファイルを作成する場合、CREATE PROFILEシステム権限が必要です。

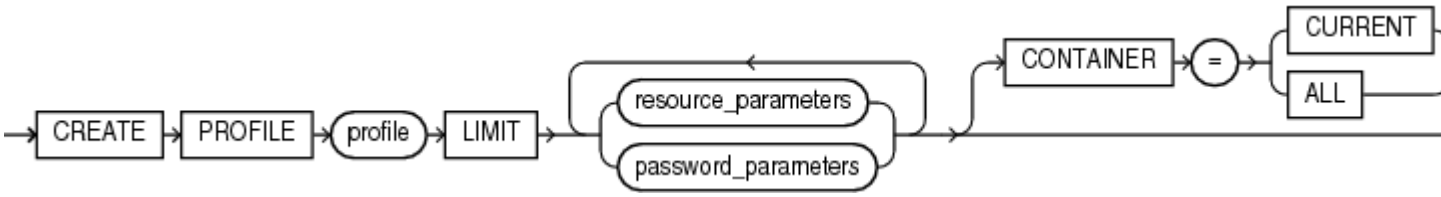
CONTAINER句を指定する場合は、マルチテナント・コンテナ・データベース(CDB)に接続している必要があります。CONTAINER = ALLを指定する場合は、現在のコンテナがルートである必要があります。CONTAINER = CURRENTを指定するには、現在のコンテナがプラガブル・データベース(PDB)である必要があります。

関連項目:

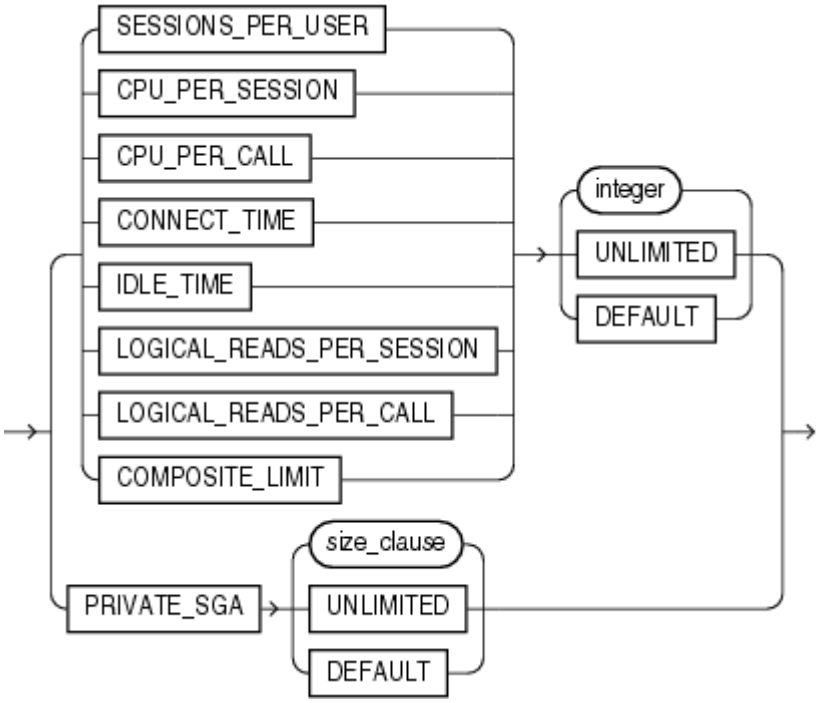
- 動的にリソース制限を使用可能にする場合の詳細は、「ALTER SYSTEM」を参照してください。
- RESOURCE_LIMITパラメータについては、『Oracle Databaseリファレンス』を参照してください。
- プロファイルについては、「CREATE USER」および「ALTER USER」を参照してください。

構文

create_profile::=

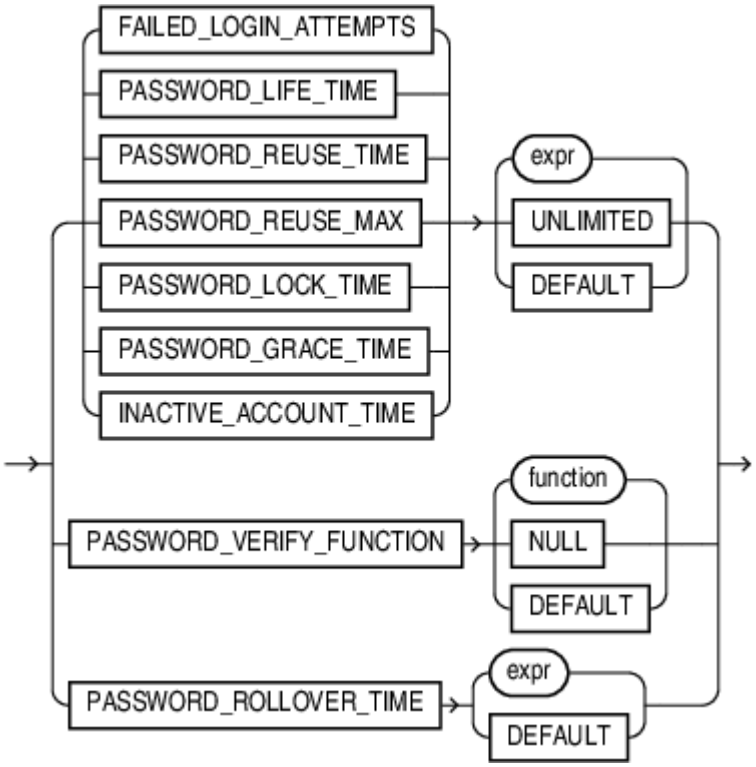


resource_parameters::=



(size_clause::=

password_parameters::=



セマンティクス

profile

作成するプロファイルの名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。プロファイルを使用した場合、ユーザーが使用可能なデータベース・リソースを1つのコールまたは1つのセッションごとに制限できます。

CDB以外では、プロファイル名の先頭をC##またはc##にできません。

CDBでは、プロファイル名の要件は次のとおりです。

- 共通プロファイルの名前は、COMMON_USER_PREFIX初期化パラメータで指定された接頭辞と大/小文字を区別しないで一致する文字から始める必要があります。デフォルトでは、接頭辞はC##です。
- ローカル・プロファイルの名前は、COMMON_USER_PREFIX初期化パラメータで指定された接頭辞と大/小文字を区別しないで一致する文字から始めないでください。COMMON_USER_PREFIXの値に関係なく、ローカル・プロファイルの名前の先頭にC##またはc##を使用できません。

ノート:

COMMON_USER_PREFIX の値が空の文字列である場合、共通またはローカル・プロファイル名に要件はありませんが、次の 1 つの例外があります。ローカル・プロファイルの名前の先頭に C##または c##を使用できません。PDB を別の CDB に接続する場合または共通ユーザーの作成時に閉じられた PDB を開く場合、ローカルおよび共通プロファイルの名前が競合する可能性があるため、空の文字列を使用しないことをお薦めします。

Oracle Databaseでは、次の方法でリソース制限を適用します。

- CONNECT_TIMEまたはIDLE_TIMEで指定したセッション・リソース制限を超えた場合、現行のトランザクションが自動的にロールバックされ、セッションが終了します。ユーザー・プロセスで次にコールを実行すると、エラーが戻ります。

- 他のセッション・リソースに対する制限を超える処理を実行しようとした場合、その処理が自動的に中断され、現行の文がロールバックされ、すぐにエラーが戻されます。この後、ユーザーは、現行のトランザクションをコミットまたはロールバックできます。そして、セッションを終了する必要があります。
- 1つのコールに対する制限を超える処理を実行しようとした場合、その処理が自動的に中断され、現行の文がロールバックされ、エラーが戻されます。この場合、現行のトランザクションは有効です。

ノート:

- 日を単位として、時間を制限するすべてのパラメータに対して、分数で日数を指定できます。たとえば、1時間は 1/24、1 分は 1/1440 になります。
- リソース制限を使用可能にしているかどうかにかかわらず、ユーザーに対してリソース制限を指定できます。ただし、Oracle Database では、実際にその制限が使用可能になってからリソース制限が適用されます。

関連項目:

プロファイルの作成: 例

UNLIMITED

リソース・パラメータでUNLIMITEDを指定すると、このプロファイルを割り当てられたユーザーは無制限にリソースを使用できます。パスワード・パラメータでUNLIMITEDを指定した場合は、パラメータに制限が設定されていないことを示します。

DEFAULT

DEFAULTを指定すると、このプロファイルでリソースの制限を省略できます。このプロファイルを割り当てられたユーザーは、DEFAULTプロファイルで指定した対象リソースに対する制限を受けます。DEFAULTプロファイルは、最初に無制限のリソースを定義します。ALTER PROFILE文でこの制限を変更できます。

明示的にプロファイルが割り当てられていないユーザーは、DEFAULTプロファイルに定義されている制限を受けます。ユーザーに明示的に割り当てられているプロファイルでリソースに対する制限が省略されている場合、または制限に対してDEFAULTが指定されている場合、ユーザーはDEFAULTプロファイルで定義されているリソースに関する制限を受けます。

resource_parameters

SESSIONS_PER_USER

ユーザーを制限する同時セッションの数を指定します。

CPU_PER_SESSION

1セッション当たりのCPU時間制限を指定します。この値は100分の1秒単位で指定します。

CPU_PER_CALL

1コール(解析、実行またはフェッチ)当たりのCPU時間制限を指定します。この値は100分の1秒単位で指定します。

CONNECT_TIME

1セッション当たりの合計経過時間制限を指定します。この値は分単位で指定します。

IDLE_TIME

セッション中の連続的な非活動時間の許容される長さを指定します。この値は分単位で指定します。実行時間の長い問合せおよびその他の操作には、この制限は適用されません。

アイドル・タイムアウトのX分を設定すると、セッションが終了するのにさらに数分かかることに注意してください。

クライアント・アプリケーション側では、アイドル・クライアントが次に新しいコマンドを発行しようとしたときにのみ、エラー・メッセージが表示されます。

LOGICAL_READS_PER_SESSION

メモリーおよびディスクから読み込まれるブロックなど、1セッション中に読み込まれるデータ・ブロックの許容数を指定します。

LOGICAL_READS_PER_CALL

SQL文(解析、実行またはフェッチ)を処理する1コールで読み込まれるデータ・ブロックの許容数を指定します。

PRIVATE_SGA

1セッションでシステム・グローバル領域(SGA)の共有プール内に割り当てることができるプライベート領域の大きさを指定します。この句の詳細は、「size_clause」を参照してください。

ノート:

この制限は、共有サーバー・アーキテクチャを使用している場合のみに適用されます。SGA 内のセッション用のプライベート領域には、プライベート SQL および PL/SQL 領域が含まれますが、共有 SQL および PL/SQL 領域は含まれません。

COMPOSITE_LIMIT

1セッション当たりのリソースの総コストを指定します。この値はサービス単位で指定します。サービス単位の合計は、CPU_PER_SESSION、CONNECT_TIME、LOGICAL_READS_PER_SESSIONおよびPRIVATE_SGAの重み付き合計として計算されます。

関連項目:

- セッション・リソースの重み付けを指定する方法については、「ALTER RESOURCE COST」を参照してください。
- プロファイルのリソース制限の設定: 例

password_parameters

次の句を使用すると、パスワード・パラメータを設定できます。時間の長さを設定するパラメータ(FAILED_LOGIN_ATTEMPTSおよびPASSWORD_REUSE_MAXを除くすべてのパスワード・パラメータ)は、日数で解析されます。これらのパラメータには、テストのために、分(n/1440)または秒(n/86400)を指定できます。また、この目的のために小数値(約1時間を示す.0833など)も使用できます。最小値は1秒です。最大値は24855日です。FAILED_LOGIN_ATTEMPTSおよびPASSWORD_REUSE_MAXには、整数を指定する必要があります。

FAILED_LOGIN_ATTEMPTS

ユーザー・アカウントがロックされる前に、そのアカウントへのログインに連続して失敗できる回数を指定します。この句を指定しない場合、10回がデフォルトになります。

PASSWORD_LIFE_TIME

同じパスワードを認証に使用できる日数を指定します。PASSWORD_GRACE_TIMEの値とともに設定した場合、猶予期間内にパスワードを変更しないと、そのパスワードは使用できなくなり、それ以降の接続は拒否されます。この句を指定しない場合、180日がデフォルトになります。

関連項目:

PASSWORD_LIFE_TIMEの値を短くする場合は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

PASSWORD_REUSE_TIMEおよびPASSWORD_REUSE_MAX

これらの2つのパラメータは互いに組み合せて設定する必要があります。PASSWORD_REUSE_TIMEは、パスワードを再利用できない日数を指定します。PASSWORD_REUSE_MAXは、現行のパスワードを再利用する前に必要な、パスワードの変更回数を指定します。これらのパラメータを有効にするには、両方のパラメータに値を指定する必要があります。

- 両方のパラメータに値を指定すると、PASSWORD_REUSE_TIMEに指定した日数の間は、PASSWORD_REUSE_MAXに指定した回数までパスワードが変更されなりかぎり、パスワードを再利用できません。

  たとえば、PASSWORD_REUSE_TIMEに30を指定し、PASSWORD_REUSE_MAXに10を指定した場合、パスワードが10回変更されていると、30日後にパスワードを再利用できます。
- これらのパラメータのいずれかに値を指定し、他方にUNLIMITEDを指定すると、パスワードは再利用できなくなります。
- いずれかのパラメータにDEFAULTを指定すると、DEFAULTプロファイルに定義された値が使用されます。DEFAULTプロファイルでは、すべてのパラメータはデフォルトでUNLIMITEDに設定されます。DEFAULTプロファイルのデフォルト設定のUNLIMITEDを変更していない場合、パラメータの値はUNLIMITEDとして扱われます。
- 両方のパラメータをUNLIMITEDに設定すると、これらのパラメータは無視されます。これは、両方のパラメータを指定しない場合のデフォルトです。

PASSWORD_LOCK_TIME

ログインが指定された回数連続して失敗した場合、アカウントがロックされる日数を指定します。この句を指定しない場合、1日がデフォルトになります。

PASSWORD_GRACE_TIME

警告が発行され、ログインが許可される猶予期間の日数を指定します。この句を指定しない場合、7日がデフォルトになります。

INACTIVE_ACCOUNT_TIME

ユーザー・アカウントがロックされるまでの連続非ログイン日数の許容値を指定します。最小値は15日です。最大値は24855です。この句を指定しない場合、UNLIMITEDがデフォルトになります。

PASSWORD_VERIFY_FUNCTION

PASSWORD_VERIFY_FUNCTIONを指定することにより、PL/SQLパスワードの複雑性の検証スクリプトをCREATE PROFILEの引数として渡すことができます。Oracle Databaseにはデフォルトのスクリプトが用意されていますが、独自のファンクションを作成することも、サード・パーティのソフトウェアを使用することもできます。

- functionには、複雑なパスワード検証ルーチンの名前を指定します。ファンクションがSYSスキーマ内にある必要があり、ファンクションにEXECUTE権限が必要です。

- NULLを指定すると、パスワードの検証が行われないことを示します。

パスワード・パラメータにexprを指定する場合、スカラー副問合せ式を除くすべての形式の式を指定できます。

パスワード・パラメータの制限事項

外部ユーザーまたはグローバル・ユーザーにプロファイルを割り当てる場合、パスワードのパラメータは、そのユーザーに対して有効ではありません。

関連項目:

プロファイルのパスワード制限の設定: 例

PASSWORD_ROLLOVER_TIME

段階的なデータベース・パスワード・ロールオーバーを有効にするには、PASSWORD_ROLLOVER_TIMEユーザー・プロファイル・パラメータにゼロ以外の制限を構成する必要があります。このパラメータは、CREATE PROFILEまたはALTER PROFILEを使用して構成できます。

exprを使用して、PASSWORD_ROLLOVER_TIMEの値を日数で指定します。時間は1日の分数として指定する必要があります。たとえば、制限を4時間に設定する場合、exprは4/24になります。

PASSWORD_ROLLOVER_TIME制限値の粒度は1秒です。たとえば、(1/24) + (3/1440) + (5/86400))のようにexprを指定することで、1時間と3分5秒の制限を設定できます。

PASSWORD_ROLLOVER_TIMEのデフォルト設定は0で、段階的なパスワード・ロールオーバーが無効であることを意味します。

例

この例では、段階的なパスワード・ロールオーバー期間を1日に設定します。

```
CREATE PROFILE usr_prof LIMIT PASSWORD_ROLLOVER_TIME 1
```

PASSWORD_ROLLOVER_TIMEの制限:

- パスワード・ロールオーバー期間を無効にする場合は、PASSWORD_ROLLOVER_TIMEに値0を指定します。
- PASSWORD_ROLLOVER_TIMEに正の値を指定すると、プロファイルのメンバーであるすべてのユーザーに対してパスワード・ロールオーバー機能が有効になります。
- PASSWORD_ROLLOVER_TIMEに指定できる最小値は1時間です。これには、1/24と入力します。パスワード・ロールオーバー時間を6時間に設定する場合は、PASSWORD_ROLLOVER_TIMEの値として6/24を入力します。
- PASSWORD_ROLLOVER_TIMEの値は、60日、またはプロファイルのPASSWORD_GRACE_TIME制限の現在の値、あるいはプロファイルのPASSWORD_LIFE_TIME制限の現在の値のいずれか低い方を超えることはできません。

パスワード・ロールオーバー期間中のユーザー・アカウントを検索するには、DBA_USERSデータ・ディクショナリ・ビューのACCOUNT_STATUS列を問い合せます。ステータスはIN ROLLOVERになります。

パスワード・ロールオーバー期間は、ユーザーがパスワードを変更した時点から始まります。

関連項目:

認証の構成

CONTAINER句

CONTAINER句は、CDBに接続しているときに適用されます。ただし、デフォルト値が使用できる唯一の値であるため、CONTAINER句を指定する必要はありません。

- 共通プロファイルを作成するには、ルートに接続する必要があります。ルートに接続する場合にデフォルトのCONTAINER = ALLをオプションで指定できます。
- ローカル・プロファイルを作成するには、PDBに接続する必要があります。PDBに接続する場合にデフォルトのCONTAINER = CURRENTをオプションで指定できます。

例

プロファイルの作成: 例

次の文は、プロファイルnew_profileを作成します。

```
CREATE PROFILE new_profile
  LIMIT PASSWORD_REUSE_MAX 10
        PASSWORD_REUSE_TIME 30;
```

プロファイルのリソース制限の設定: 例

次の文は、プロファイルapp_userを作成します。

```
CREATE PROFILE app_user LIMIT
   SESSIONS_PER_USER          UNLIMITED
   CPU_PER_SESSION            UNLIMITED
   CPU_PER_CALL               3000
   CONNECT_TIME               45
   LOGICAL_READS_PER_SESSION  DEFAULT
   LOGICAL_READS_PER_CALL     1000
   PRIVATE_SGA                15K
   COMPOSITE_LIMIT            5000000;
```

ユーザーにapp_userプロファイルを割り当てた場合、そのユーザーは、後続のセッションで次の制限を受けます。

- ユーザーは、無制限に同時セッションを使用できます。
- 1つのセッションにおいて、ユーザーはCPU時間を無制限に使用できます。
- ユーザーが作成した1つのコールで消費できるCPU時間は30秒以下です。
- 1つのセッションで継続できる時間は45分以下です。
- 1つのセッションでメモリーおよびディスクから読み込むデータ・ブロック数は、DEFAULTプロファイルで指定した制限を受けます。
- ユーザーが実行した1つのコールで、メモリーまたはディスクから読み込むことができるデータ・ブロック数の合計は1000以下です。
- 1つのセッションで割り当てることができるSGA内のメモリーは、15KB以下です。
- 1つのセッションのリソースの総コストは、サービス単位で500万を超えることはできません。リソースの総コストの計算式は、ALTER RESOURCE COST文で指定します。
- app_userプロファイルでは、IDLE_TIMEおよびパスワードに対する制限が指定されていないため、ユーザーはDEFAULTプロファイルに指定されているこれらのリソースの制限を受けます。

プロファイルのパスワード制限の設定: 例

次の文は、パスワード制限値が設定されたapp_user2プロファイルを作成します。

```
CREATE PROFILE app_user2 LIMIT
   FAILED_LOGIN_ATTEMPTS 5
   PASSWORD_LIFE_TIME 60
   PASSWORD_REUSE_TIME 60
   PASSWORD_REUSE_MAX 5
   PASSWORD_VERIFY_FUNCTION ora12c_verify_function
   PASSWORD_LOCK_TIME 1/24
   PASSWORD_GRACE_TIME 10
   INACTIVE_ACCOUNT_TIME 30;
```

この例では、Oracle Databaseのデフォルトのパスワード検証ファンクションora12c_verify_functionを使用します。この検証ファンクションの使用方法または独自の検証ファンクションの設計方法については、『Oracle Databaseセキュリティ・ガイド』を参照してください。

CREATE RESTORE POINT

目的

CREATE RESTORE POINT文を使用すると、リストア・ポイント(タイムスタンプまたはデータベースのSCNに関連付けられた名前)を作成できます。リストア・ポイントを使用すると、リストア・ポイントで指定された時点に表またはデータベースをフラッシュバックできるため、SCNまたはタイムスタンプかを判断する必要がありません。リストア・ポイントは、バックアップやデータベースの複製など、様々なRMAN操作でも有効です。RMANを使用して、アーカイブ・バックアップの実装プロセスにリストア・ポイントを作成できます。

関連項目:

- リストア・ポイントおよび保証付きリストア・ポイントの作成と使用、データベースの複製およびアーカイブ・バックアップの詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- リストア・ポイントの使用方法および削除方法については、「FLASHBACK DATABASE」、「FLASHBACK TABLE」および「DROP RESTORE POINT」を参照してください。

前提条件

通常のリストア・ポイントを作成するには、SELECT ANY DICTIONARY、FLASHBACK ANY TABLE、SYSDBA、SYSBACKUP、SYSDGシステム権限が必要です。

保証付きリストア・ポイントを作成するには、次の条件のうち1つを満たしている必要があります。

- AS SYSDBA、AS SYSBACKUPまたはAS SYSDGを接続する必要があります。
- SYSDBA権限が付与されており、マルチテナント・データベースを使用している必要があります。
- ユーザーSYSとして実行しており、マルチテナント・データベースを使用している必要があります。

リストア・ポイントを表示または使用するには、SELECT ANY DICTIONARYまたはFLASHBACK ANY TABLE、SYSDBA、SYSBACKUP、またはSYSDGのいずれかのシステム権限か、SELECT_CATALOG_ROLEロールが必要です。

リストア・ポイントは、プライマリ・データベースまたはスタンバイ・データベースに作成できます。データベースは、オープンされている状態でも、マウントされているがオープンされていない状態でもかまいません。データベースがマウントされている場合、それがフィジカル・スタンバイ・データベースでなければ、整合性を保持して停止されてからマウントされている必要があります。

保証付きリストア・ポイントを作成する場合は、その前に、高速リカバリ領域を作成しておく必要があります。保証付きリストア・ポイントを作成する前に、フラッシュバック・データベースを使用可能にする必要はありません。保証付きリストア・ポイントを作成する場合、データベースはARCHIVELOGモードになっていることが必要です。

通常のリストア・ポイントを作成する前にフラッシュバック・データベースを有効にする必要はありません。通常のリストア・ポイントにはFLASHBACK DATABASE以外のアプリケーションがあるためです。ただし、通常のリストア・ポイントに対してFLASHBACK DATABASEを実行する場合は、通常のリストア・ポイントを作成する前にフラッシュバック・データベースを有効にしておく必要があります。

マルチテナント・コンテナ・データベース(CDB)に接続しているときに、次のようにリストア・ポイントを作成、使用または表示できます。

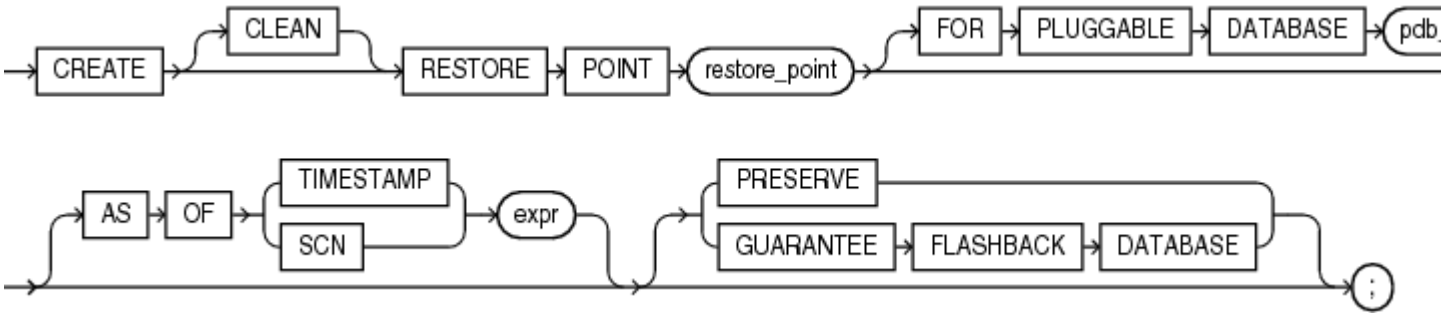
- 通常のCDBリストア・ポイントを作成するには、現在のコンテナがルートである必要があります。また、SELECT ANY DICTIONARYまたはFLASHBACK ANY TABLEシステム権限(共通に付与されている権限か、ルートでローカルに付

与されている権限のいずれか)または共通に付与されているSYSDBA、SYSBACKUPまたはSYSDGシステム権限が必要です。

- 保証されたCDBリストア・ポイントを作成する場合は、現在のコンテナがルートである必要があります。また、共通に付与されているSYSDBA、SYSBACKUPまたはSYSDGシステム権限が必要です。
- CDBリストア・ポイントを表示するには、現在のコンテナがルートである必要があり、SELECT ANY DICTIONARYまたはFLASHBACK ANY TABLEシステム権限またはSELECT_CATALOG_ROLEロール(共通に付与されているか権限/ロールか、ルートでローカルに付与されている権限/ロールのいずれか)、または共通に付与されているSYSDBA、SYSBACKUPまたはSYSDGシステム権限が必要であるか、あるいは現在のコンテナがPDBである必要があり、SELECT ANY DICTIONARY、FLASHBACK ANY TABLE、SYSDBA、SYSBACKUPまたはSYSDGシステム権限(共通に付与されている権限か、そのPDBでローカルに付与されている権限のいずれか)が必要です。
- CDBリストア・ポイントを使用するには、SELECT ANY DICTIONARYまたはFLASHBACK ANY TABLEシステム権限またはSELECT_CATALOG_ROLEロール(共通に付与されている権限/ロールか、ルートでローカルに付与されている権限/ロールのいずれか)または共通に付与されているSYSDBA、SYSBACKUPまたはSYSDGシステム権限が必要です。
- 通常のPDBリストア・ポイントを作成するには、現在のコンテナがルートである必要があり、SELECT ANY DICTIONARYまたはFLASHBACK ANY TABLEシステム権限(共通に付与されているか権限か、ルートでローカルに付与されている権限のいずれか)、または共通に付与されているSYSDBA、SYSBACKUPまたはSYSDGシステム権限が必要であるか、または現在のコンテナがリストア・ポイントを作成するPDBである必要があり、SELECT ANY DICTIONARY、FLASHBACK ANY TABLE、SYSDBA、SYSBACKUPまたはSYSDGシステム権限(共通に付与されている権限か、そのPDBでローカルに付与されている権限)が必要です。
- 保証されたPDBリストア・ポイントを作成するには、現在のコンテナがルートである必要があり、共通に付与されているSYSDBA、SYSBACKUPまたはSYSDGシステム権限が必要であるか、または現在のコンテナがリストア・ポイントを作成するPDBである必要があり、SYSDBA、SYSBACKUPまたはSYSDGシステム権限(共通に付与されている権限か、そのPDBでローカルに付与されている権限)が必要です。
- PDBリストア・ポイントを表示するには、現在のコンテナがルートである必要があり、SELECT ANY DICTIONARYまたはFLASHBACK ANY TABLEシステム権限またはSELECT_CATALOG_ROLEロール(共通に付与されているか権限/ロールか、ルートでローカルに付与されている権限/ロールのいずれか)、または共通に付与されているSYSDBA、SYSBACKUPまたはSYSDGシステム権限が必要であるか、または現在のコンテナがリストア・ポイントのPDBである必要があり、SELECT ANY DICTIONARY、FLASHBACK ANY TABLE、SYSDBA、SYSBACKUPまたはSYSDGシステム権限(共通に付与されている権限か、そのPDBでローカルに付与されている権限のいずれか)が必要です。
- PDBリストア・ポイントを使用するには、現在のコンテナがリストア・ポイントのPDBである必要があります。また、SELECT ANY DICTIONARY、FLASHBACK ANY TABLE、SYSDBA、SYSBACKUPまたはSYSDGシステム権限(共通に付与されている権限か、そのPDBでローカルに付与されている権限)が必要です。

構文

create_restore_point::=



セマンティクス

CLEAN

CLEANは、PDBリストア・ポイントを作成する際にのみ指定できます。PDBは共有UNDOを使用する必要があり、未処理のトランザクションがない状態でクローズする必要があります。共有UNDOを使用してPDBをフラッシュバックしてPDBリストア・ポイントをクリーンにする場合、バックアップをリストアしたりクローン・インスタンスを作成する必要はありません。このため、共有UNDOを使用してPDBをSCNまたは他のタイプのリストア・ポイントにフラッシュバックするよりも高速です。

restore_point

リストア・ポイントの名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

マルチテナント環境では、CDBとPDBはリストア・ポイント用の独自のネームスペースを持ちます。このため、CDBおよび各PDBは同じ名前のリストア・ポイントを持つことがあります。PDB内でまたはPDB操作でリストア・ポイント名を指定すると、最初に名前が当該PDBのPDBリストア・ポイントとして解釈されます。指定された名前のPDBリストア・ポイントが見つからない場合、CDBリストア・ポイントとして解釈されます。

データベースには、2048個以上の通常のリストア・ポイントを保持できます。マルチテナント環境では、CDBはCDB全体で少なくとも2048の通常のリストア・ポイント(PDBリストア・ポイントを含む)を保持できます。通常のリストア・ポイントは、少なくとも初期化パラメータCONTROL_FILE_RECORD_KEEP_TIMEで指定された日数はデータベース内に保持されます。そのパラメータのデフォルト値は7日です。保証付きのリストア・ポイントは、ユーザーが明示的に削除するまでデータベース内に保持されます。

PRESERVEもGUARANTEE FLASHBACK DATABASEも指定しない場合、結果のリストア・ポイントによって、DB_FLASHBACK_RETENTION_TARGET初期化パラメータで設定された期間内のリストア・ポイントにデータベースをフラッシュバックできます。そのようなリストア・ポイントは、データベースによって自動的に管理されます。リストア・ポイントの数が、前述のrestore_pointで定められた最大数に達すると、最も古いリストア・ポイントが自動的に削除されます。一定の状況では長期バックアップのリストアに使用するために、リストア・ポイントはRMANリカバリ・カタログに保持されます。DROP RESTORE POINT文を使用して、リストア・ポイントを明示的に削除することもできます。

FOR PLUGGABLE DATABASE

この句を使用すると、ルートに接続されている場合にPDBリストア・ポイントを作成できます。pdb_nameには、PDBの名前を指定します。

リストア・ポイントを作成するPDBに接続されている場合、この句を指定する必要はありません。ただし、この句を指定する場合は、接続しているPDBの名前を指定する必要があります。

AS OF句

この句を使用すると、過去の指定した日時またはSCNでリストア・ポイントを作成できます。TIMESTAMPを指定する場合、

exprは過去の日時となる有効な日時式である必要があります。SCNを指定する場合、exprは過去のデータベースの有効なSCNである必要があります。どちらの場合にも、exprはデータベースの現行のインカネーションの日時またはSCNを参照する必要があります。

PRESERVE

PRESERVEを指定すると、リストア・ポイントを明示的に削除する必要があることを指定できます。このようなリストア・ポイントは、フラッシュバック・データベースを保持するのに有効です。

GUARANTEE FLASHBACK DATABASE

保証付きリストア・ポイントの場合、初期化パラメータDB_FLASHBACK_RETENTION_TARGETの設定とは無関係に、データベースをリストア・ポイントに確定的にフラッシュバックできます。フラッシュバック機能が保証されるかどうかは、高速リカバリ領域に十分な使用可能スペースがあるかどうかによって決まります。

保証付きリストア・ポイントで保証されるのは、データベースを保証付きリストア・ポイントにフラッシュバックするのに十分なフラッシュバック・ログが、データベースに保持されるということのみです。すべての表を同じリストア・ポイントにフラッシュバックするのに十分なUNDOがデータベースに保持されるかどうかは保証されません。

保証付きリストア・ポイントは、常に保存されます。DROP RESTORE POINT文を使用してユーザーが明示的に削除する必要があります。このリストア・ポイントは、期限切れになることはありません。保証付きリストア・ポイントは、高速リカバリ領域内の領域を大量に使用する場合があります。このため、保証付きリストア・ポイントは十分に検討したうえで作成することをお薦めします。

例

リストア・ポイントの作成および使用方法: 例

次の例は、通常のリストア・ポイントを作成し、表を更新した後、変更した表をリストア・ポイントにフラッシュバックします。この例は、ユーザーhrがそれぞれの文を使用する適切なシステム権限を持っていることを前提としています。

```
CREATE RESTORE POINT good_data;
SELECT salary FROM employees WHERE employee_id = 108;
    SALARY
----------
     12000
UPDATE employees SET salary = salary*10
   WHERE employee_id = 108;
SELECT salary FROM employees
   WHERE employee_id = 108;
    SALARY
----------
    120000
COMMIT;
FLASHBACK TABLE employees TO RESTORE POINT good_data;
SELECT salary FROM employees
   WHERE employee_id = 108;
    SALARY
----------
     12000
```

CREATE ROLE

目的

CREATE ROLE文を使用すると、ロールを作成できます。ロールは、ユーザーまたは他のロールに付与できる権限の集合です。ロールを使用して、データベース権限を管理できます。ロールに権限を追加し、そのロールをユーザーに付与することができます。その後、ユーザーはロールを有効にして、そのロールによって付与された権限を行使することができます。

ロールには、そのロールに付与されたすべての権限、およびそのロールに付与された他のロールのすべての権限が含まれています。新しく作成されたロールには、ロールや権限は付与されていません。GRANT文を使用して、ロールに様々な権限を追加します。

NOT IDENTIFIED、IDENTIFIED EXTERNALLYまたはBY passwordロールを作成した場合、そのロールはADMIN OPTION付きで付与されます。ただし、ロールIDENTIFIED GLOBALLYを作成した場合、ロールは付与されません。グローバル・ロールは、ユーザーまたはロールに対して直接付与できません。グローバル・ロールを付与する場合は、エンタープライズ・ロールを介して行う必要があります。

関連項目:

- ロールの付与については、「GRANT」を参照してください。
- ロールを使用可能にする場合は、「ALTER USER」を参照してください。
- データベースのロールの変更または削除については、「ALTER ROLE」および「DROP ROLE」を参照してください。
- 現行のセッションに対するロールの使用可能化および使用禁止化については、「SET ROLE」を参照してください。
- ロールの概要は、『Oracle Databaseセキュリティ・ガイド』を参照してください。
- エンタープライズ・ロールの詳細は、『Oracle Databaseエンタープライズ・ユーザー・セキュリティ管理者ガイド』を参照してください。

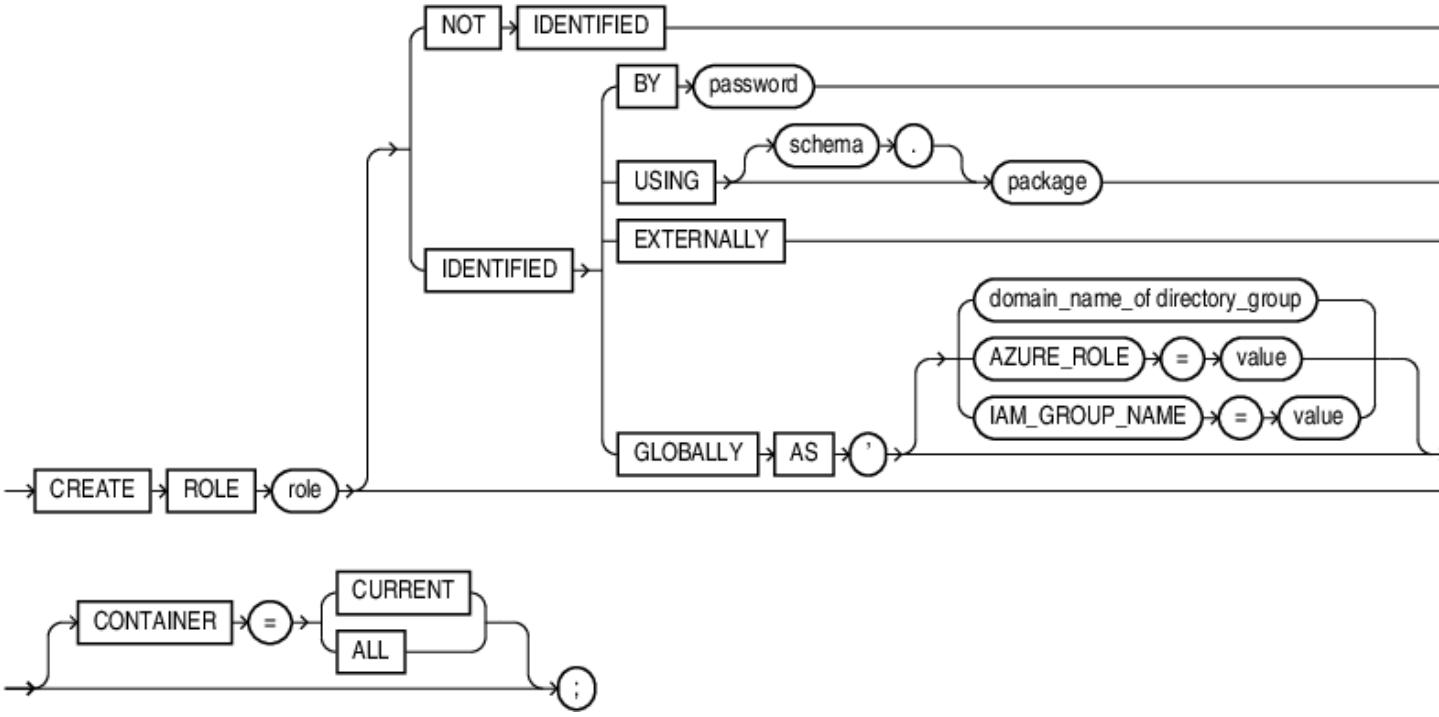
前提条件

CREATE ROLEシステム権限が必要です。

CONTAINER句を指定する場合は、マルチテナント・コンテナ・データベース(CDB)に接続している必要があります。CONTAINER = ALLを指定する場合は、現在のコンテナがルートである必要があります。CONTAINER = CURRENTを指定するには、現在のコンテナがプラガブル・データベース(PDB)である必要があります。

構文

create_role::=



セマンティクス

role

作成するロールの名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。データベース文字セットにマルチバイト文字がサポートされている場合でも、ロールにはシングルバイト文字を1つ以上使用することをお薦めします。ロールの名前の最大長は128バイトです。各ユーザーに対し、一度に使用可能にできるユーザー定義のロールの最大数は148です。

CDB以外では、ロール名の先頭をC##またはc##にできません。

CDBでは、ロール名の要件は次のとおりです。

- 共通ロールの名前は、COMMON_USER_PREFIX初期化パラメータで指定された接頭辞と大/小文字を区別しないで一致する文字から始める必要があります。デフォルトでは、接頭辞はC##です。
- ローカル・ロールの名前は、COMMON_USER_PREFIX初期化パラメータで指定された接頭辞と大/小文字を区別しないで一致する文字から始めないでください。COMMON_USER_PREFIXの値に関係なく、ローカル・ロールの名前の先頭にC##またはc##を使用できません。

ノート:

COMMON_USER_PREFIX の値が空の文字列である場合、共通またはローカル・ロール名に要件はありませんが、次の 1 つの例外があります。ローカル・ロールの名前の先頭に C##または c##を使用できません。PDB を別の CDB に接続する場合または共通ユーザーの作成時に閉じられた PDB を開く場合、ローカルおよび共通ロールの名前が競合する可能性があるため、空の文字列を使用しないことをお薦めします。

配布メディアで提供されているSQLスクリプトには、いくつかのロールが定義されています。

関連項目:

事前定義済のロールのリストについては、「GRANT」を、ユーザーに対するロールの使用可能化および使用禁止化については、「SET ROLE」を参照してください。

NOT IDENTIFIED句

NOT IDENTIFIEDを指定すると、指定するロールがデータベースによって認可され、パスワードを入力しなくてもこのロールを使用可能にできます。

IDENTIFIED句

IDENTIFIED句を使用すると、SET ROLE文によってロールを使用可能にする前に、指定したメソッドによってユーザーが認可される必要があります。

BY password

BY password句を使用して、パスワードを持つローカル・ロールを作成できます。つまり、ロールを有効にするときに、データベースにパスワードを指定する必要があるということです。

パスワードには、NULL文字(CHR(0))と二重引用符を除いて、データベース文字セットの任意の文字を含めることができます。パスワードは構文上は識別子であり、データベース・オブジェクトのネーミング規則で指定されているとおり、二重引用符で囲む必要があります。データベースと、ロールを有効化する必要があるクライアントは、パスワードを構成するすべての文字をサポートするように構成されている必要があります。

プロキシ・セッションで、パスワード保護されたロールを有効にできます。セキュア・アプリケーション・ロールでも、パスワード保護されたロールでも、セッションでロールを有効にする安全な方法があります。　安全でないチャネルでパスワードを管理、転送しなければならない場合には、または複数の担当者がパスワードを知る必要がある場合には、パスワード保護されたロールではなく、セキュア・パスワード・ロールを使用することをお薦めします。プロキシ・セッションの場合、パスワード保護されたロールは、自動化処理を利用してロールを設定する状況に適しています。

USING package

USING package句を使用すると、保護アプリケーション・ロールを作成できます。保護アプリケーション・ロールとは、認可済パッケージを使用するアプリケーションによってのみ有効にできるロールです。schemaを指定しない場合、パッケージは自分のスキーマ内にあるとみなされます。

関連項目:

保護アプリケーション・ロールの作成については、『Oracle Databaseセキュリティ・ガイド』を参照してください。

EXTERNALLY

EXTERNALLYを指定すると、外部ロールを作成できます。外部ユーザーは、ロールを使用可能にする前に、オペレーティング・システムやサード・パーティ・サービスなどの外部サービスによって認可されている必要があります。

オペレーティング・システムによっては、ユーザーがオペレーティング・システムに対してパスワードを指定しないと、ロールが使用可能にできない場合もあります。

GLOBALLY

GLOBALLYを指定すると、グローバル・ロールを作成できます　グローバル・ユーザーは、ログイン時にロールの使用が可能になる前に、エンタープライズ・ディレクトリ・サービスによってロールの使用を認可されている必要があります。

中央管理されているユーザーを使用している場合、GLOBALLYをASとともに指定して、ディレクトリ・グループをグローバル・ロールにマップします。ディレクトリ・グループはそのドメイン名によって識別されます。

例: グローバル・ユーザーへのディレクトリ・ユーザーのマッピング

```
  CREATE USER scott_global IDENTIFIED GLOBALLY AS ‘cn=scott
taylor,ou=sales,dc=abccorp,dc=com’;
```

これは実質的に、abccorp.comドメインのsales組織単位のscott taylorという名前のディレクトリ・ユーザーを、データベース・グローバル・ユーザーscott_globalに割り当てます。

Oracle Databaseグローバル・ロールをAzureアプリケーション・ロールにマップして、Azureのユーザーおよびアプリケーションに、ログイン・スキーマを介して付与された権限およびロールを超える権限およびロールを付与できます。

例: アプリケーション・ロールへのOracle Databaseグローバル・ロールのマッピング

この例では、新しいデータベース・グローバル・ロール`widget_sales_role`を作成し、既存のAzure ADアプリケーション・ロール`WidgetManagerGroup`にマップします。

```
CREATE ROLE widget_sales_role IDENTIFIED GLOBALLY AS 'AZURE_ROLE=WidgetManagerGroup';
```

関連項目:

Oracle Autonomous DatabasesのMicrosoft Azure Active Directoryユーザーの認証および認可

CONTAINER句

`CONTAINER`句は、CDBに接続しているときに適用されます。ただし、デフォルト値が使用できる唯一の値であるため、`CONTAINER`句を指定する必要はありません。

- 共通ロールを作成するには、ルートに接続する必要があります。ルートに接続する場合にデフォルトの`CONTAINER = ALL`をオプションで指定できます。
- ローカル・ロールを作成するには、PDBに接続する必要があります。PDBに接続する場合にデフォルトの`CONTAINER = CURRENT`をオプションで指定できます。

例

ロールの作成: 例

次の文は、ロール`dw_manager`を作成します。

```
CREATE ROLE dw_manager;
```

この後に`dw_manager`ロールを付与されるユーザーは、このロールに付与されているすべての権限を継承します。

次の例のように、パスワードの指定によってロールにセキュリティのレイヤーを追加できます。

```
CREATE ROLE dw_manager
   IDENTIFIED BY warehouse;
```

この後、`dw_manager`ロールを付与されたユーザーは、パスワード`warehouse`を指定して、`SET ROLE`文でロールを使用可能にする必要があります。

次の文は、グローバル・ロール`warehouse_user`を作成します。

```
CREATE ROLE warehouse_user IDENTIFIED GLOBALLY;
```

次の文は、同じロールを外部ロールとして作成します。

```
CREATE ROLE warehouse_user IDENTIFIED EXTERNALLY;
```

次の文は、現在のPDBにローカル・ロールrole1を作成します。この文を発行するときには、現在のコンテナがPDBである必要があります。

```
CREATE ROLE role1 CONTAINER = CURRENT;
```

次の文では、共通ロールc##role1を作成します。この文を発行するときには、現在のコンテナがルートである必要があります。

```
CREATE ROLE c##role1 CONTAINER = ALL;
```

CREATE ROLLBACK SEGMENT

ノート:

ロールバック・セグメントを使用せずに、自動 UNDO 管理モードでデータベースを実行することを強くお薦めします。ロールバック・セグメントは、以前のバージョンの Oracle Database との互換性に必要な場合以外は使用しないでください。自動 UNDO 管理の詳細は、『Oracle Database 管理者ガイド』を参照してください。

目的

CREATE ROLLBACK SEGMENT文を使用すると、ロールバック・セグメントを作成できます。ロールバック・セグメントとは、トランザクションによる変更を元に戻す(取り消す)ために必要なデータを格納する際にOracle Databaseが使用するオブジェクトです。

ここでは、データベースがロールバックUNDOモードで実行されている(初期化パラメータUNDO_MANAGEMENTにMANUALを設定、またはすべて設定しない)ことを前提としています。データベースを自動UNDOモード(UNDO_MANAGEMENT初期化パラメータを、デフォルト設定であるAUTOに設定します)で実行している場合は、ロールバック・セグメントは使用できません。ただし、ロールバック・セグメントでのエラーの生成は抑制されます。

また、データベースにローカル管理SYSTEM表領域がある場合、ディクショナリ管理表領域にロールバック・セグメントは作成できません。かわりに、自動UNDO管理機能を使用するか、またはローカル管理表領域を作成して、ロールバック・セグメントを保持する必要があります。

ノート:

1 つの表領域に複数のロールバック・セグメントを作成できます。一般に、複数のロールバック・セグメントがあると、パフォーマンスが向上します。

- 表領域にロールバック・セグメントを追加する場合、表領域は必ずオンラインである必要があります。
- ロールバック・セグメントを作成した場合、最初はオフライン状態になります。そのロールバック・セグメントを Oracle Database インスタンスでトランザクション可能にする場合、ALTER ROLLBACK SEGMENT 文を使用してオンラインの状態にしてください。データベース起動時に自動的にオンライン状態にする場合、初期化パラメータ ROLLBACK_SEGMENT の値にそのセグメントの名前を追加してください。

SYSTEM以外の表領域のオブジェクトを使用するには:

- UNDO用にロールバック・セグメントを使用するには、1つ以上のロールバック・セグメント(SYSTEMロールバック・セグメント以外)がオンラインである必要があります。
- 自動UNDOモードでデータベースを実行するには、1つ以上のUNDO表領域がオンラインである必要があります。

関連項目:

- ロールバック・セグメントの変更については、「ALTER ROLLBACK SEGMENT」を参照してください。

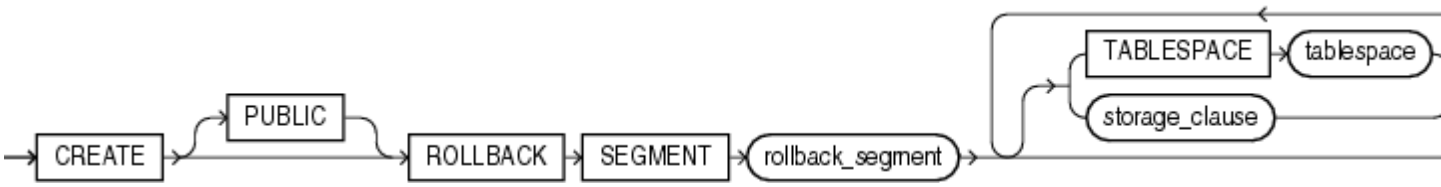
- ロールバック・セグメントの削除については、「DROP ROLLBACK SEGMENT」を参照してください。
- UNDO_MANAGEMENTパラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。
- 自動UNDOモードの詳細は、『Oracle Database管理者ガイド』を参照してください。

前提条件

ロールバック・セグメントを作成するには、CREATE ROLLBACK SEGMENTシステム権限が必要です。

構文

create_rollback_segment::=



(storage_clause)

セマンティクス

PUBLIC

PUBLICを指定すると、ロールバック・セグメントがパブリックとなり、すべてのインスタンスに対して使用可能にできます。この句を省略した場合、ロールバック・セグメントはプライベートになり、インスタンスの初期化パラメータROLLBACK_SEGMENTSで指定したインスタンスに対してのみ使用可能になります。

rollback_segment

作成するロールバック・セグメントの名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

TABLESPACE

TABLESPACE句を使用すると、ロールバック・セグメントが作成される表領域を指定できます。この句を省略した場合、ロールバック・セグメントはSYSTEM表領域に作成されます。

ノート:

Oracle Database は、頻繁にロールバック・セグメントにアクセスする必要があります。そのため、明示的または(この句を省略して)暗黙的に、ロールバック・セグメントを SYSTEM 表領域に作成しないことをお薦めします。さらに、ロールバック・セグメントが含まれている表領域の競合を回避するために、この表領域には表、索引などの他のオブジェクトを含めないでください。また、エクステントの割当ておよび割当て解除は、最小限に抑える必要があります。

そのためには、自動割当てを使用禁止にしたローカル管理表領域にロールバック・セグメントを作成します。このような表領域は、EXTENT MANAGEMENT LOCAL 句に UNIFORM を指定して作成します。AUTOALLOCATE 設定はサポートされません。

関連項目:

CREATE TABLESPACE

storage_clause

storage_clauseを使用すると、ロールバック・セグメントの記憶特性を指定できます。

- storage_clauseのOPTIMALパラメータは、ロールバック・セグメントにのみ適用されるので、特に重要です。
- storage_clauseのPCTINCREASEパラメータは、CREATE ROLLBACK SEGMENTでは指定できません。

関連項目:

storage_clause

例

ロールバック・セグメントの作成: 例

次の文は、適切に構成された表領域にデフォルトの記憶域値を含むロールバック・セグメントを作成します。

```
CREATE TABLESPACE rbs_ts
   DATAFILE 'rbs01.dbf' SIZE 10M
   EXTENT MANAGEMENT LOCAL UNIFORM SIZE 100K;
/* This example and the next will fail if your database is in
   automatic undo mode.
*/
CREATE ROLLBACK SEGMENT rbs_one
   TABLESPACE rbs_ts;
```

前述の文は、次の文と同じ結果になります。

```
CREATE ROLLBACK SEGMENT rbs_one
   TABLESPACE rbs_ts
   STORAGE
   ( INITIAL 10K );
```

CREATE SCHEMA

目的

CREATE SCHEMA文を使用すると、複数表およびビューを作成し、自分のスキーマ内に1つのトランザクションで複数の権限を付与できます。

CREATE SCHEMA文を実行すると、挿入されている個々の文が実行されます。すべての文が正常に実行された場合、そのトランザクションがコミットされます。文の結果が1つでもエラーになった場合は、すべての文がロールバックされます。

ノート:

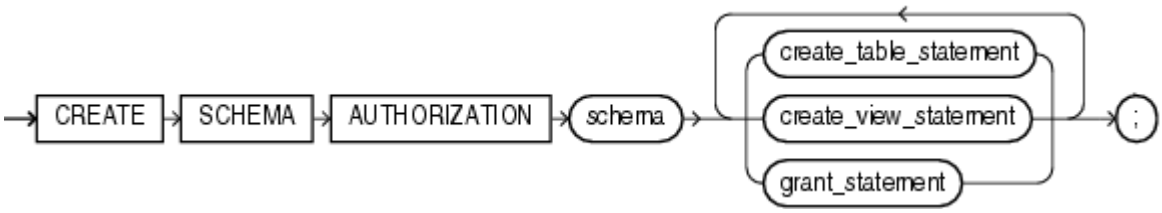
この文では、実際にスキーマが作成されるわけではありません。ユーザーを作成すると、自動的にスキーマが作成されます(「CREATE USER」を参照)。この文を実行すると、複数のトランザクションで複数の SQL 文を発行しなくても、スキーマに表およびビューが移入され、これらのオブジェクトに対する権限が付与されます。

前提条件

CREATE SCHEMA文には、CREATE TABLE文、CREATE VIEW文およびGRANT文を含めることができます。CREATE SCHEMA文を発行する場合は、挿入した文を発行するための権限が必要です。

構文

create_schema::=



セマンティクス

schema

スキーマの名前を指定します。スキーマ名は、Oracle Databaseユーザー名と一致している必要があります。

create_table_statement

このCREATE SCHEMA文の一部として発行するCREATE TABLE文を指定します。この文の終わりには、セミコロン(またはその他の終了文字)を付けないでください。

関連項目:

CREATE TABLE

create_view_statement

このCREATE SCHEMA文の一部として発行するCREATE VIEW文を指定します。この文の終わりには、セミコロン(またはその他の終了文字)を付けないでください。

関連項目:

CREATE VIEW

grant_statement

このCREATE SCHEMA文の一部として発行するGRANT文を指定します。この文の終わりには、セミコロン(またはその他の終了文字)を付けないでください。この句を使用すると、所有するオブジェクトに対するオブジェクト権限を、他のユーザーに付与できます。また、WITH ADMIN OPTION付きでシステム権限を付与されている場合、それらの権限を他のユーザーに付与できます。

関連項目:

GRANT

CREATE SCHEMA文は、Oracle Databaseでサポートされている完全な構文ではなく、標準SQLで定義されている構文のみをサポートします。

CREATE TABLE、CREATE VIEWおよびGRANTの各文を指定する順序は重要ではありません。CREATE SCHEMA文の中の文では、既存のオブジェクトまたは同じCREATE SCHEMA文の他の文で作成したオブジェクトを参照できます。

スキーマに対する権限付与の制限事項

parallel_clauseの構文は、CREATE SCHEMAのCREATE TABLE文で使用できますが、オブジェクトを作成するときにパラレル化は使用されません。

関連項目:

「CREATE TABLE」の「parallel_clause」を参照してください。

例

スキーマの作成: 例

次の文は、サンプルの注文入力ユーザーoe用のoeという名前のスキーマ、表new_product、ビューnew_product_viewを作成し、サンプルの人事部門のユーザーhrにnew_product_viewに対するSELECTオブジェクト権限を付与します。

```
CREATE SCHEMA AUTHORIZATION oe
   CREATE TABLE new_product
      (color VARCHAR2(10)  PRIMARY KEY, quantity NUMBER)
   CREATE VIEW new_product_view
      AS SELECT color, quantity FROM new_product WHERE color = 'RED'
   GRANT select ON new_product_view TO hr;
```

15 SQL文: CREATE SEQUENCEからDROP CLUSTER

この章では、次のSQL文について説明します。

- CREATE SEQUENCE
- CREATE SPFILE
- CREATE SYNONYM
- CREATE TABLE
- CREATE TABLESPACE
- CREATE TABLESPACE SET
- CREATE TRIGGER
- CREATE TYPE
- CREATE TYPE BODY
- CREATE USER
- CREATE VIEW
- DELETE
- DISASSOCIATE STATISTICS
- DROP ANALYTIC VIEW
- DROP ATTRIBUTE DIMENSION
- DROP AUDIT POLICY (統合監査)
- DROP CLUSTER

CREATE SEQUENCE

目的

CREATE SEQUENCE文を使用すると、順序を作成できます。順序とは、複数のユーザーが一意の整数を生成するときに使用できるデータベース・オブジェクトです。順序を使用すると、主キー値を自動的に生成できます。

順序番号が生成されると、順序はトランザクションのコミットやロールバックとは無関係に増加していきます。2人のユーザーが、同時に同一の順序を増加させると、ユーザーがそれぞれ順序番号を生成しているため、取得する順序番号間に違いが発生することもあります。他のユーザーが生成した順序番号は取得できません。あるユーザーが順序値を生成すると、他のユーザーがその順序を増加させたかどうかに関係なく、順序を生成したユーザーは引き続きその値にアクセスできます。

順序番号は表とは関係なく生成されるため、1つ以上の表に対して同一の順序を使用できます。順序番号が生成および使用されるトランザクションが最終的にロールバックされた場合、個々の順序番号が連続していないように見える場合があります。また、他のユーザーが同一順序を使用していることを個々のユーザーが認識しない場合もあります。

順序が作成されると、SQL文の中でCURRVAL疑似列を使用してその値にアクセスできます(この場合、その順序の現在の値が戻ります)。また、NEXTVAL疑似列を使用してもアクセスできます(この場合は、順序が増加され、新しい値が戻ります)。

関連項目:

- CURRVALおよびNEXTVALの使用方法の詳細は、「疑似列」を参照してください。
- 順序の使用方法の詳細は、「順序値の使用方法」を参照してください。
- 順序の変更または削除の詳細は、「ALTER SEQUENCE」または「DROP SEQUENCE」を参照してください。

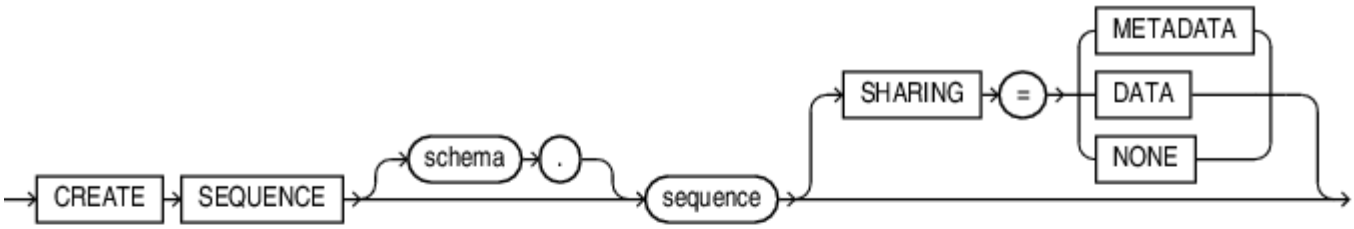
前提条件

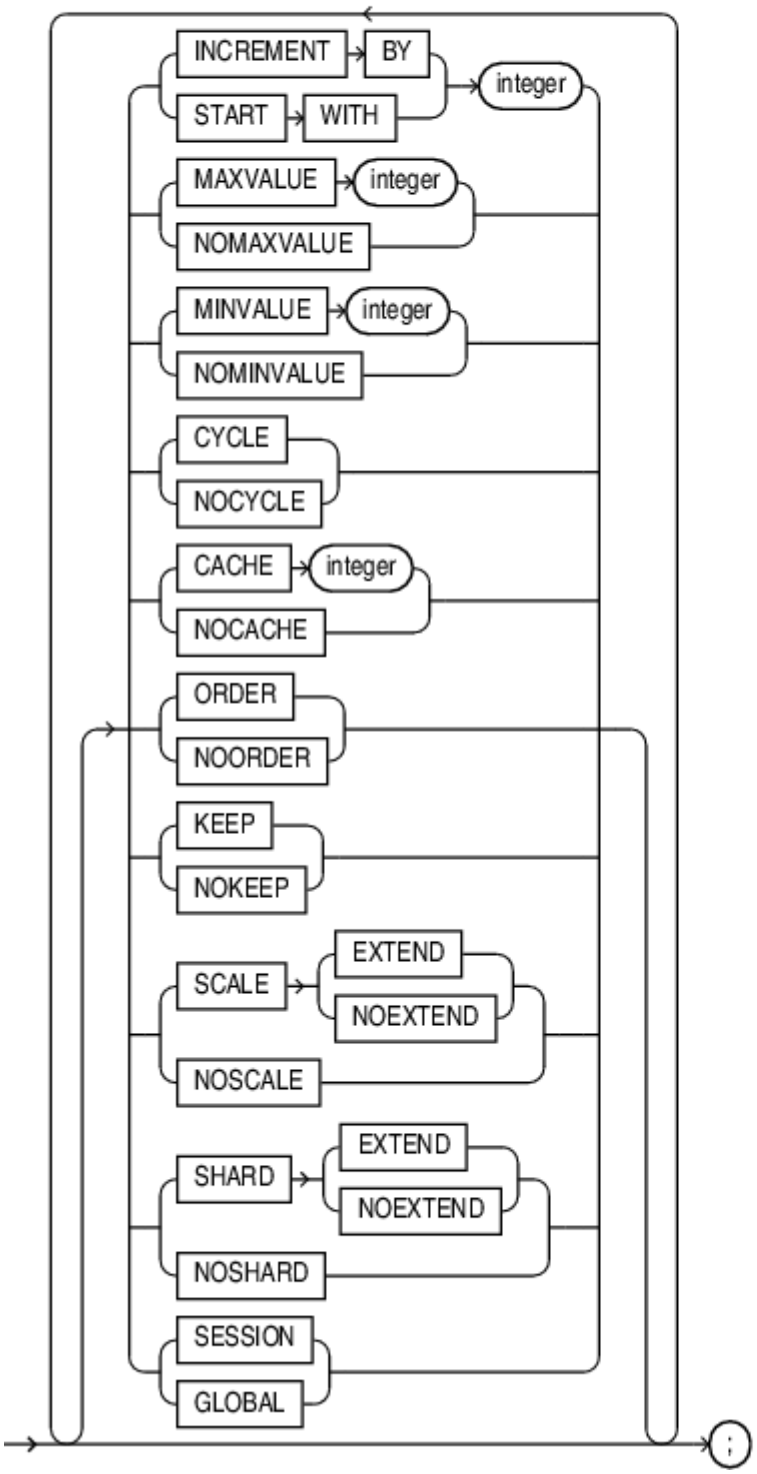
自分のスキーマ内に順序を作成する場合は、CREATE SEQUENCEシステム権限が必要です。

他のユーザーのスキーマ内に順序を作成する場合は、CREATE ANY SEQUENCEシステム権限が必要です。

構文

create_sequence::=





セマンティクス

schema

順序を含めるスキーマを指定します。schemaを省略した場合、自分のスキーマ内に順序が作成されます。

sequence

作成する順序の名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

INCREMENT BYからGLOBALまでの句のうちどれも指定しない場合は、1から始まる昇順の順序が作成され、上限なしで1ずつ増加していきます。INCREMENT BY -1のみを指定した場合は、-1から始まる降順の順序が作成され、下限なしで1ずつ減少していきます。

- 昇順で、無制限に増加する順序を作成する場合は、MAXVALUEパラメータを省略するか、NOMAXVALUEを指定します。降順の場合は、MINVALUEパラメータを省略するか、またはNOMINVALUEを指定します。
- 昇順で、事前に定義した制限で停止する順序を作成する場合は、MAXVALUEパラメータに値を指定します。降順の場合は、MINVALUEパラメータの値を指定します。NOCYCLEも指定します。順序が制限に達したときに順序番号をさらに生成しようとした場合、エラーが発生します。
- 事前に定義した制限に達した後で初期値に戻る順序を作成する場合は、MAXVALUEパラメータとMINVALUEパラメータの両方に値を指定します。CYCLEも指定します。

SHARING

この句は、アプリケーション・ルートに順序を作成する場合にのみ適用されます。このタイプの順序はアプリケーション共通オブジェクトと呼ばれ、アプリケーション・ルートに属するアプリケーションPDBと共有できます。順序の共有方法を決定するには、次の共有属性のいずれかを指定します。

- METADATA - メタデータ・リンクは順序のメタデータを共有しますが、そのデータは各コンテナに固有です。このタイプの順序は、メタデータリンク・アプリケーション共通オブジェクトと呼ばれます。
- DATA - データ・リンクは順序を共有し、そのデータはアプリケーション・コンテナ内のすべてのコンテナに対して同じです。そのデータはアプリケーション・ルートにのみ格納されます。このタイプの順序は、データリンク・アプリケーション共通オブジェクトと呼ばれます。
- NONE - 順序は共有されません。

この句を省略した場合、データベースはDEFAULT_SHARING初期化パラメータの値を使用して、順序の共有属性を決定します。DEFAULT_SHARING初期化パラメータに値が含まれていない場合、デフォルトはMETADATAです。

順序を作成した後、その共有属性を変更することはできません。

関連項目:

- DEFAULT_SHARING初期化パラメータの詳細は、Oracle Databaseリファレンスを参照してください
- アプリケーション共通オブジェクトの作成の詳細は、『Oracle Database管理者ガイド』を参照してください。

INCREMENT BY

順序の番号間の増分間隔を指定します。この値は、0(ゼロ)以外の正の整数または負の整数になります。この値には、昇順の場合は28桁以内、降順の場合は27桁以内の値を指定できます。この値の絶対値は、MAXVALUEとMINVALUEの差未満である必要があります。この値が負の場合、順序は降順になります。この値が正の場合、順序は昇順になります。この句を省略した場合、デフォルトで増分間隔は1に設定されます。

START WITH

生成する順序番号の初期値を指定します。この句を指定した場合、順序の最小値より大きい値を初期値として昇順を開始することも、最大値よりも小さい値を初期値として降順を開始することもできます。昇順の場合、デフォルト値は順序の最小値になります。降順の場合、デフォルト値は順序の最大値になります。正の値の場合は28桁以内、負の値の場合は27桁以内の値を指定できます。

ノート:

この値は、必ずしも、順序の最大値または最小値にそれぞれ達した後に、昇順または降順で循環する順序が戻るときの値ではありません。

MAXVALUE

順序が生成できる最大値を指定します。正の値の場合は28桁以内、負の値の場合は27桁以内の値を指定できます。MAXVALUE値は、START WITH以上で、かつMINVALUEを超える値である必要があります。

NOMAXVALUE

NOMAXVALUEを指定すると、昇順の場合は10^{28}-1、降順の場合は-1が最大値になります。これはデフォルトです。

MINVALUE

順序の最小値を指定します。正の値の場合は28桁以内、負の値の場合は27桁以内の値を指定できます。MINVALUE値は、START WITH以下で、かつMAXVALUE未満である必要があります。

NOMINVALUE

NOMINVALUEを指定すると、昇順の場合は1、降順の場合は-(10^{27} -1)が最小値になります。これはデフォルトです。

CYCLE

CYCLEを指定すると、順序が最大値または最小値に達した後、引き続き値が生成されます。つまり、昇順の場合は、最大値に達すると最小値が生成されます。降順の場合は、最小値に達すると最大値が生成されます。

NOCYCLE

NOCYCLEを指定すると、順序が最大値または最小値に達した後、それ以上値が生成されません。これはデフォルトです。

CACHE

より高速にアクセスできるように、データベースがメモリーに事前に割り当て、保持しておく順序の値の数を指定します。28桁以内の整数値を指定できます。このパラメータの最小値は2です。循環する順序の場合、この値は、そのサイクル内で生成される値の数未満である必要があります。指定したサイクル内で生成される順序番号の数を超える値はキャッシュできません。したがって、CACHEに指定できる値の最大値は、次の式で求められる値未満である必要があります。

```
CEIL ( (MAXVALUE - MINVALUE) / ABS (INCREMENT) )
```

システム障害が発生すると、キャッシュされた順序の値のうち、コミットされたDML文で使用されていなかったものはすべて失われます。したがって、失われる可能性がある値の数は、CACHEパラメータの値と等しくなります。

ノート:

Oracle Real Application Clusters 環境で順序を使用する場合は、パフォーマンスを向上するために、CACHE の設定を使用することをお薦めします。

NOCACHE

NOCACHEを指定すると、順序の値が事前に割り当てられません。CACHEおよびNOCACHEの両方を省略した場合、デフォルトで20の順序番号がキャッシュされます。

ORDER

ORDERを指定すると、要求どおりの順序で順序番号を生成することを保証できます。順序番号をタイムスタンプとして使用する場合に、この句は有効です。通常、主キー生成用の順序については、順序どおりに生成するかどうかの保証は重要ではありません。

NOORDER

要求どおりの順序で順序番号を生成することを保証しない場合は、NOORDERを指定します。これはデフォルトです。

KEEP

アプリケーション・コンティニュイティのための再実行中に、NEXTVALが元の値を保持するようにするには、KEEPを指定します。この動作は、アプリケーションを実行しているユーザーが、順序を含むスキーマの所有者である場合にのみ行われます。この句は、リカバリ可能なエラー後の再実行でバインド変数の一貫性を保持するために役立ちます。アプリケーション・コンティニュイティの詳細は、『Oracle Database開発ガイド』を参照してください。

NOKEEP

アプリケーション・コンティニュイティのための再実行中に、NEXTVALが元の値を保持しないようにするには、NOKEEPを指定します。これはデフォルトです。

ノート:

KEEP 句と NOKEEP 句は、順序を含むスキーマの所有者にのみ適用されます。アプリケーション・コンティニュイティのための再実行中に、その他のユーザーの元の値を NEXTVAL で維持するかどうかは、順序に対する KEEP SEQUENCE オブジェクト権限を付与または取消しすることで制御できます。KEEP SEQUENCE オブジェクト権限の詳細は、「表 18-2」を参照してください。

SCALE

順序の拡張性を有効にするには、SCALEを使用します。SCALEが指定されている場合、数値オフセットが順序の先頭に付加され、生成された値からすべての重複が削除されます。

EXTEND

EXTENDをSCALEとともに指定すると、生成される順序値は全長(x+y)になります。ここで、xは拡張可能なオフセット(デフォルト値は6)、yは順序内の数字の最大数(maxvalue/minvalue)です。

SCALEを使用する場合は、順序に対してORDERを同時に使用しないことをお薦めします。

NOEXTEND

NOEXTENDは、SCALE句のデフォルト設定です。NOEXTENDが設定されていると、生成される順序値の幅は最大でも順序内

の数字の最大数(maxvalue/minvalue)です。この設定は、固定幅の列を移入するために順序が使用される、既存のアプリケーションとの統合に役立ちます。

NOSCALE

順序の拡張性を無効にするには、NOSCALEを使用します。

SHARD

SHARD句のセマンティクスの詳細は、ALTER SEQUENCE文のSHARD句を参照してください。

SESSION

SESSIONを指定すると、セッションの順序を作成できます。この順序は、セッションから認識できるグローバル一時表に使用することを具体的に指定する特種な順序です。既存の通常の順序(対比のために「グローバルな」順序と呼びます)とは異なり、セッションの順序は、複数のセッション全体ではなく、1つのセッションに限定された順序番号の一意の範囲を返します。また、セッションの順序は永続されない点も異なります。セッションが失われると、そのセッション中にアクセスされるセッションの順序も失われます。

セッションの順序は、読取り/書込みデータベースで作成する必要がありますが、読取り/書込みデータベースまたは読取り専用の任意のデータベース(一時的に読取り専用でオープンされた通常のデータベース、またはスタンバイ・データベース)からアクセスできます。

CACHE句、NOCACHE句、ORDER句、またはNOORDER句は、SESSION句と同時に指定すると無視されます。

関連項目:

セッション順序の詳細は、『Oracle Data Guard概要および管理』を参照してください。

GLOBAL

GLOBALを指定すると、グローバルな順序(つまり、通常の順序)を作成できます。これはデフォルトです。

例

順序の作成: 例

次の文は、サンプル・スキーマoeに順序customers_seqを作成します。この順序は、customers表に行が追加されるときに、顧客ID番号として使用できます。

```
CREATE SEQUENCE customers_seq
 START WITH     1000
 INCREMENT BY   1
 NOCACHE
 NOCYCLE;
```

最初にcustomers_seq.nextvalを参照したときは、1000が戻されます。次に参照したときは、1001が戻されます。同様に、この後の各参照で、前回参照された値より1大きい値が戻されます。

CREATE SPFILE

目的

CREATE SPFILE文を使用すると、従来のプレーンテキストの初期化パラメータ・ファイルまたは現行のシステム全体の設定から、サーバー・パラメータ・ファイルを作成できます。サーバー・パラメータ・ファイルは、サーバーのみに存在し、データベースを起動するためにクライアントからコールされるバイナリ・ファイルです。

サーバー・パラメータ・ファイルを指定すると、個々のパラメータを永続的に変更できます。サーバー・パラメータ・ファイルを使用する場合、新しいパラメータ値を永続化するALTER SYSTEM SET parameter文を指定できます。新しい値は、現行のインスタンスのみでなく、その後で起動するすべてのインスタンスにおいて適用されます。従来のプレーンテキストのパラメータ・ファイルでは、パラメータ値を永続的に変更できません。

サーバー・パラメータ・ファイルは、サーバー上に存在するため、Oracle Databaseによる自動データベースのチューニングおよびRecovery Manager(RMAN)によるバックアップが可能です。

データベースの起動時にサーバー・パラメータ・ファイルを使用するには、CREATE SPFILE文を使用してサーバー・パラメータ・ファイルを作成する必要があります。

Oracle Real Application Clusters環境のすべてのインスタンスは、同じサーバー・パラメータ・ファイルを使用する必要があります。ただし、別途許可されている場合は、個々のインスタンスで、1つのファイル内の同じパラメータの設定値をそれぞれ変えて使用できます。インスタンス固有のパラメータ定義は、SID.parameter = valueで指定します。SIDにはインスタンス識別子を指定します。

サーバー・パラメータ・ファイルを使用したデータベースの起動方法は、作成したサーバー・パラメータ・ファイルをデフォルトで作成したか、または非デフォルトで作成したかによって異なります。サーバー・パラメータ・ファイルの使用方法の例は、「サーバー・パラメータ・ファイルの作成: 例」を参照してください。

CDBにサーバー・パラメータ・ファイルを作成する場合のノート

マルチテナント・コンテナ・データベース(CDB)にサーバー・パラメータ・ファイルを作成する場合、現在のコンテナがルートまたはPDBである可能性があります。

- 現在のコンテナがルートである場合、rootの初期化パラメータに設定した値が、その他のすべてのコンテナのデフォルト値として使用されます。
- 現在のコンテナがPDBの場合、データベースはPDBの初期化パラメータ値をファイルにではなく内部的に格納します。このため、spfile_nameを指定できません。PDBの初期化パラメータに設定する値は永続的で、ルートのそれらのパラメータに設定される値をオーバーライドします。

その後ALTER SYSTEM文を使用して、rootまたはPDBの初期化パラメータを変更することができます。

関連項目:

- バイナリのサーバー・パラメータ・ファイルから正規のテキストのパラメータ・ファイルを作成する場合の詳細は、「CREATE PFILE」を参照してください。
- 従来のプレーンテキストの初期化パラメータ・ファイルおよびサーバー・パラメータ・ファイルの詳細は、『Oracle Database管理者ガイド』を参照してください。
- Oracle Real Application Clusters環境でのサーバー・パラメータ・ファイルの使用については、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください。

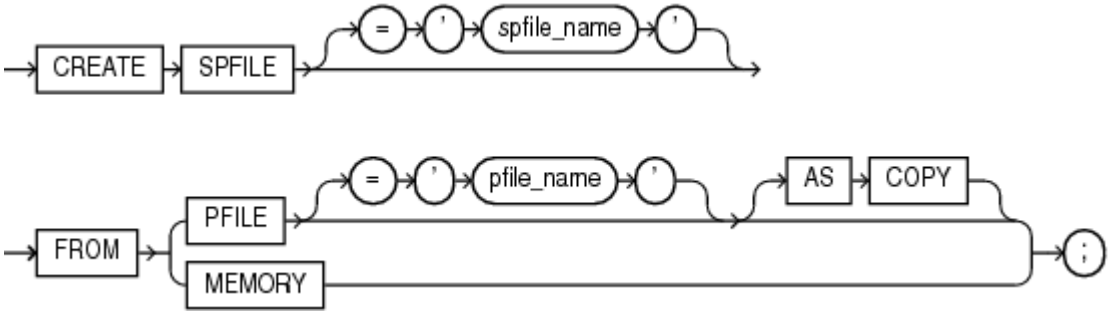
前提条件

この文を実行するには、SYSBACKUP、SYSDBA、SYSDGまたはSYSOPERシステム権限が必要です。この文は、インスタンスの起動前と起動後のいずれかで実行できます。ただし、spfile_nameを使用して、インスタンスがすでに起動済の場合は、この文で同じspfile_nameを指定できません。

CDBにサーバー・パラメータ・ファイルを作成する場合は、現在のコンテナがルートである必要があります。また、共通に付与されているSYSBACKUP、SYSDBA、SYSDGまたはSYSOPERシステム権限が必要です。

構文

create_spfile::=



セマンティクス

spfile_name

この句を指定すると、作成するサーバー・パラメータ・ファイルの名前を指定できます。

spfile_nameを指定した場合、Oracle Databaseではデフォルト以外のサーバー・パラメータ・ファイルが作成されます。

- spfile_nameでは、従来のファイル名、Oracle ACFSファイル・システム内のファイルまたはOracle Storage Management (Oracle ASM)ファイル名を指定できます。
- 従来のファイル名またはOracle ACFSファイル・システム内のファイルを指定した場合、spfile_nameにはパス接頭辞を含めることができます。パス接頭辞を指定しない場合は、データベースによってデフォルトの格納場所(プラットフォームによって異なる)のパス接頭辞が追加されます。
- Oracle ASMファイル名構文を指定する場合、データベースはOracle ASMディスク・グループにspfileを作成します。
- デフォルト以外のサーバー・パラメータ・ファイルを使用する場合は、データベースの起動時にSTARTUPコマンドでサーバー・パラメータ・ファイルを指定する必要があります。このルールの例外は次のとおりです。
    - データベースがOracle Clusterwareでリソースとして定義されており、コマンドの発行元のインスタンスが実行中で、spfile_nameとFROM PFILE句を指定し、AS COPY句を省略した場合、この文はデータベース・リソース内のSPFILEを自動的に更新します。この場合、サーバー・パラメータ・ファイルを名前で参照せずにデータベースを起動できます。コマンドの発行元のインスタンスが実行中でない場合は、データベース・リソース内のSPFILEを、srvctl modify database -d dbname -spfile spfile_pathを使用して手動で更新する必要があります。

spfile_nameを省略した場合、プラットフォーム固有のデフォルトのサーバー・パラメータ・ファイル名が使用されます。このようなファイルがサーバーにすでに存在する場合、そのファイルは上書きされます。デフォルトのサーバー・パラメータ・ファイルを使用する場合、ファイル名を参照せずにデータベースを起動できます。

spfile_nameの制限事項

PDBへの接続中にサーバー・パラメータ・ファイルを作成する場合は、spfile_nameを指定できません。

関連項目:

- デフォルトおよびデフォルト以外のサーバー・パラメータ・ファイルを使用したデータベースの起動の詳細は、「サーバー・パラメータ・ファイルの作成: 例」を参照してください。
- 従来のファイル名およびOracle ASMファイル名の構文については「file_specification」を、Oracle ASMファイルの特性の変更の詳細は「ALTER DISKGROUP」を参照してください。
- デフォルトのパラメータ・ファイル名については、オペレーティング・システム固有のドキュメントを参照してください。

pfile_name

サーバー・パラメータ・ファイルを作成する元になる従来のプレーンテキストの初期化パラメータ・ファイル名を指定します。従来のパラメータ・ファイルは、サーバー上にある必要があります。

- pfile_nameを指定して、従来のパラメータ・ファイルがオペレーティング・システムのパラメータ・ファイルのデフォルト・ディレクトリに存在しない場合は、フルパス名を指定する必要があります。
- pfile_nameを指定しない場合、オペレーティング・システムのパラメータ・ファイルのデフォルト・ディレクトリからデフォルトのパラメータ・ファイル名が検索され、使用されます。そのディレクトリにファイルが存在しない場合、エラーが戻されます。

ノート:

Oracle Real Application Clusters 環境では、この文でパラメータ・ファイル名を指定してサーバー・パラメータ・ファイルを作成する前に、まず、すべてのインスタンスのパラメータ・ファイルを 1 つのファイルに結合する必要があります。このステップの実行の詳細は、『Oracle Real Application Clusters 管理およびデプロイメント・ガイド』を参照してください。

AS COPY

この句は、データベースがOracle Clusterwareでリソースとして定義されている場合にのみ適用されます。デフォルトでは、spfile_nameとFROM PFILE句の両方を指定した場合、CREATE SPFILE文はデータベース・リソース内のSPFILEを自動的に更新します。AS COPYを指定して、データベース・リソース内のSPFILEが更新されるのを回避できます。

MEMORY

MEMORYを指定すると、現行のシステム全体のパラメータ設定を使用してspfileを作成できます。Oracle RAC環境では、作成されたファイルには各インスタンスからのパラメータ設定が含まれます。

例

サーバー・パラメータ・ファイルの作成: 例

次の例は、従来のプレーンテキストのパラメータ・ファイルt_init1.oraからデフォルトのサーバー・パラメータ・ファイルを作成します。

```
CREATE SPFILE
   FROM PFILE = '$ORACLE_HOME/work/t_init1.ora';
```

ノート:

通常、オペレーティング・システムのパラメータ・ファイルには、フルパスのファイル名を指定する必要があります。

デフォルトのサーバー・パラメータ・ファイルを作成する場合、その後のデータベースの起動は、次のようにPFILEパラメータなしでSQL*PlusのコマンドSTARTUPを実行し、サーバー・パラメータ・ファイルを使用します。

```
STARTUP
```

次の例は、従来のプレーンテキストのパラメータ・ファイルt_init1.oraからデフォルト以外のサーバー・パラメータ・ファイルs_params.oraを作成します。

```
CREATE SPFILE = 's_params.ora'
   FROM PFILE = '$ORACLE_HOME/work/t_init1.ora';
```

デフォルト以外のサーバー・パラメータ・ファイルを作成する場合、次の単一行を含む以前のパラメータ・ファイルを最初に作成し、その後でデータベースを起動します。

```
spfile = 's_params.ora'
```

このパラメータ・ファイル名は、ご使用のオペレーティング・システムのネーミング規則に従う必要があります。その後、STARTUPコマンドで単一行のパラメータ・ファイルを使用します。次の例では、単一行のパラメータ・ファイルの名前がnew_param.oraであるとした場合のデータベースの起動方法を示します。

```
STARTUP PFILE=new_param.ora
```

CREATE SYNONYM

目的

CREATE SYNONYM文を使用すると、シノニムを作成できます。シノニムとは、表、ビュー、順序、演算子、プロシージャ、ストアド・ファンクション、パッケージ、マテリアライズド・ビュー、Javaクラス・スキーマ・オブジェクト、ユーザー定義オブジェクト型または別のシノニムに付ける別名です。シノニムによってシノニムのターゲット・オブジェクトへの依存関係が設定され、ターゲット・オブジェクトが変更または削除されるとシノニムも無効になります。

シノニムによって、データの独立性および位置の透過性を実現できます。シノニムを使用した場合、どのユーザーが表やビューを所有しているか、どのデータベースに表やビューが格納されているかに関係なく、アプリケーションは変更なしで機能します。ただし、シノニムはデータベース・オブジェクトに対する権限にかわるものではありません。シノニムを使用するユーザーに対して、適切な権限をあらかじめ付与しておく必要があります。

シノニムを参照できるDML文は、SELECT、INSERT、UPDATE、DELETE、FLASHBACK TABLE、EXPLAIN PLAN、LOCK TABLE、MERGEおよびCALLです。

シノニムを参照できるDDL文は、AUDIT、NOAUDIT、GRANT、REVOKEおよびCOMMENTです。

関連項目:

シノニムの概要については、『Oracle Database概要』を参照してください。

前提条件

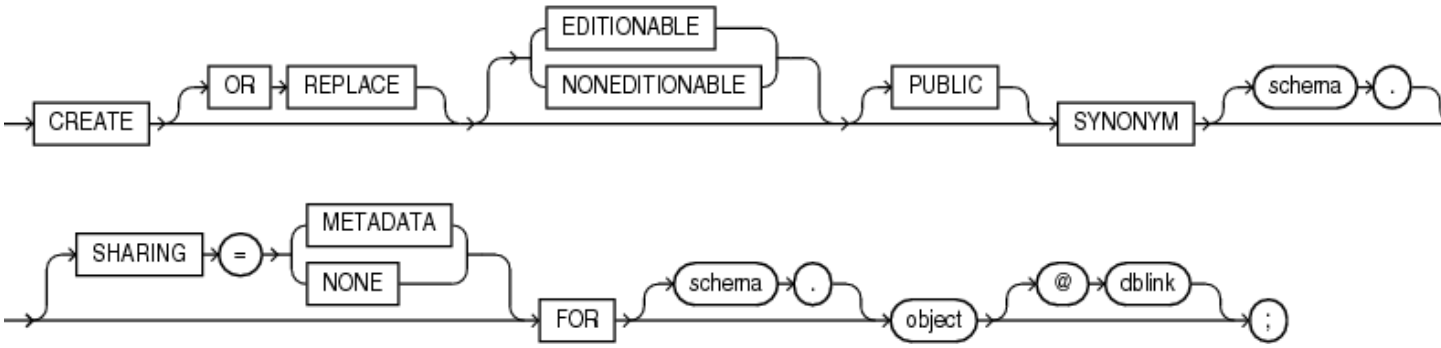
自分のスキーマ内にプライベート・シノニムを作成する場合は、CREATE SYNONYMシステム権限が必要です。

他のユーザーのスキーマ内にプライベート・シノニムを作成する場合は、CREATE ANY SYNONYMシステム権限が必要です。

PUBLICシノニムを作成する場合は、CREATE PUBLIC SYNONYMシステム権限が必要です。

構文

create_synonym::=



セマンティクス

OR REPLACE

OR REPLACEを指定すると、既存のシノニムを再作成できます。この句を使用すると、既存のシノニムの定義をはじめに削除しなくても、その定義を変更できます。

シノニムの置換の制限事項

OR REPLACE句は、依存表または依存する有効なユーザー定義オブジェクト型を持つ型シノニムに対して使用できません。

[EDITIONABLE | NONEDITIONABLE]

この句を使用すると、schemaのスキーマ・オブジェクト・タイプSYNONYMのエディショニングが有効化されたときに、そのシノニムをエディション・オブジェクトにするか非エディション・オブジェクトにするかを指定できます。プライベート・シノニムの場合、デフォルトはEDITIONABLEです。パブリック・シノニムの場合、デフォルトはNONEDITIONABLEです。エディション・オブジェクトと非エディション・オブジェクトの詳細は、『Oracle Database開発ガイド』を参照してください。

PUBLIC

PUBLICを指定すると、パブリック・シノニムを作成できます。パブリック・シノニムには、すべてのユーザーがアクセスできます。ただし、シノニムを使用するには、基礎となるオブジェクトに対する適切な権限が必要です。

オブジェクトの先頭にスキーマ名が指定されておらず、オブジェクトの後にデータベース・リンクが指定されていない場合のみ、オブジェクトへの参照を変換するときに、パブリック・シノニムが使用されます。

この句を指定しない場合、シノニムはプライベートです。プライベート・シノニム名は、スキーマ内で一意である必要があります。プライベート・シノニムに所有者以外のユーザーがアクセスできるのは、基礎となるデータベース・オブジェクトに対する適切な権限がユーザーにあり、シノニム名とともにスキーマを指定する場合のみです。

パブリック・シノニムのノート

パブリック・シノニムには、次のノートがあります。

- パブリック・シノニムを作成した後、依存表または依存する有効なユーザー定義オブジェクト型が存在する場合、依存オブジェクトと同じスキーマ内には、そのシノニムと同じ名前で別のデータベース・オブジェクトは作成できません。
- 既存のスキーマと同じ名前のパブリック・シノニムを作成しないでください。同じ名前のパブリック・シノニムを作成すると、その名前が使用されるすべてのPL/SQLユニットが無効になります。

schema

シノニムを含めるスキーマを指定します。schemaを省略した場合、自分のスキーマ内にシノニムが作成されます。PUBLICを指定した場合、スキーマは指定できません。

synonym

作成するシノニムの名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

ノート:

シノニム名の最大長には、次のルールがあります。

- COMPATIBLE 初期化パラメータが 12.2 以上の値に設定されている場合、シノニム名の最大長は 128 バイトです。データベースでは、129 から 4000 バイトの長さのシノニムを作成および削除できます。ただし、これらのより長いシノニム名が Java 名を表す場合を除いて、他の SQL コマンドでは機能しません。
- COMPATIBLE 初期化パラメータが 12.2 より小さい値に設定されている場合、シノニム名の最大長は 30 バイトです。データベースでは、31 から 128 バイトの長さのシノニムを作成および削除できます。ただし、これらのより長いシノニム名が Java 名を表す場合を除いて、他の SQL コマンドでは機能しません。

より長いシノニム名は、データ・ディクショナリに格納するために不確定で短い文字列に変換されます。

関連項目:

「CREATE SYNONYM: 例」および「Oracle Databaseによるシノニムの変換: 例」を参照してください。

SHARING

この句は、アプリケーション・ルートにシノニムを作成する場合にのみ適用されます。このタイプのシノニムはアプリケーション共通オブジェクトと呼ばれ、アプリケーション・ルートに属するアプリケーションPDBと共有できます。シノニムの共有方法を決定するには、次の共有属性のいずれかを指定します。

- `METADATA` - メタデータ・リンクはシノニムのメタデータを共有しますが、そのデータは各コンテナに固有です。このタイプのシノニムは、メタデータリンク・アプリケーション共通オブジェクトと呼ばれます。
- `NONE` - シノニムは共有されません。

この句を省略すると、データベースでは、`DEFAULT_SHARING`初期化パラメータの値を使用して、シノニムの共有属性を決定します。`DEFAULT_SHARING`初期化パラメータに値が含まれていない場合、デフォルトは`METADATA`です。

シノニムの共有属性は、作成後には変更できません。

関連項目:

- `DEFAULT_SHARING`初期化パラメータの詳細は、Oracle Databaseリファレンスを参照してください
- アプリケーション共通オブジェクトの作成の詳細は、『Oracle Database管理者ガイド』を参照してください。

FOR句

シノニムを作成するオブジェクトを指定します。シノニムを作成するスキーマ・オブジェクトには、次のものがあります。

- 表またはオブジェクト表
- ビューまたはオブジェクト・ビュー
- 順序
- ストアド・プロシージャ、ファンクションまたはパッケージ
- マテリアライズド・ビュー
- Javaクラス・スキーマ・オブジェクト
- ユーザー定義オブジェクト型
- シノニム

スキーマ・オブジェクトは、現在存在している必要はなく、スキーマ・オブジェクトへのアクセス権限も必要ありません。

FOR句の制限事項

スキーマ・オブジェクトは、パッケージに含めることはできません。

schema

オブジェクトが含まれているスキーマを指定します。オブジェクトにschemaを指定しなかった場合、そのスキーマ・オブジェクトは自

分のスキーマ内にあるとみなされます。

リモート・データベース上のプロシージャやファンクションに対するシノニムを作成する場合、このCREATE文でschemaを指定する必要があります。または、オブジェクトが存在するデータベースにローカル・パブリック・シノニムを作成することもできます。ただし、その後は、プロシージャやファンクションの後続のコールすべてにデータベース・リンクを組み込む必要があります。

dblink

データベース・リンクを完全に指定するか、データベース・リンクの一部を指定すると、スキーマ・オブジェクトが格納されているリモート・データベース上のオブジェクトのシノニムを作成できます。dblinkを指定して、schemaを省略した場合、シノニムは、データベース・リンクで指定されたスキーマ内のオブジェクトを参照します。リモート・データベースのオブジェクトが含まれているスキーマを指定することをお薦めします。

dblinkを省略した場合、オブジェクトがローカル・データベース上にあるものとみなされます。

データベース・リンクの制限事項

dblinkは、Javaクラス・シノニムに対して指定できません。

関連項目:

- データベース・リンクの参照方法の詳細は、「リモート・データベース内のオブジェクトの参照」を参照してください。
- データベース・リンクの作成方法の詳細は、「CREATE DATABASE LINK」を参照してください。

例

CREATE SYNONYM: 例

スキーマhr内の表locationsに対してシノニムofficesを定義するには、次の文を発行します。

```
CREATE SYNONYM offices
   FOR hr.locations;
```

remoteデータベース上のスキーマhr内のemployees表に対してPUBLICシノニムを作成するには、次の文を発行します。

```
CREATE PUBLIC SYNONYM emp_table
   FOR hr.employees@remote.us.example.com;
```

別のスキーマ内に基礎となるオブジェクトが含まれている場合は、基礎となるオブジェクトと同じ名前をシノニムに指定することもできます。

Oracle Databaseによるシノニムの変換: 例

Oracle Databaseは、オブジェクトの参照を、PUBLICシノニム・レベルで変換する前に、スキーマ・レベルで変換しようとします。たとえば、スキーマoeとshの両方にcustomersという名前の表が存在するとします。次の例では、ユーザーSYSTEMが、oe.customersに対してcustomersという名前のPUBLICシノニムを作成します。

```
CREATE PUBLIC SYNONYM customers FOR oe.customers;
```

ユーザーshが次の文を発行すると、sh.customersから行数が戻されます。

```
SELECT COUNT(*) FROM customers;
```

oe.customersから行数を取得するには、ユーザーshは、customersの前にスキーマ名を指定する必要があります。(ユーザーshは、oe.customersに対するSELECT権限も持っている必要があります。)

```
SELECT COUNT(*) FROM oe.customers;
```

ユーザーhrのスキーマにcustomersという名前のオブジェクトは存在しないが、hrがoe.customersに対するSELECT権限を持つ場合、hrは、パブリック・シノニムcustomersを使用して、oeのスキーマ内のcustomers表にアクセスできます。

```
SELECT COUNT(*) FROM customers;
```

CREATE TABLE

目的

CREATE TABLE文を使用すると、次の型の表を作成できます。

- リレーショナル表。ユーザー・データを格納する基本構造です。
- オブジェクト表。列の定義にオブジェクト型を使用する表です。特定の型のオブジェクト・インスタンスを格納するように明示的に定義されます。

オブジェクト型を作成しておき、リレーショナル表の作成時に列の中でそのオブジェクト型を使用することもできます。

副問合せを指定しない場合、データを含まない表が作成されます。INSERT文を使用した場合、表に行を追加できます。表を作成した後、ALTER TABLE文でADD句を指定すると、追加する列、パーティションおよび整合性制約を定義できます。ALTER TABLE文でMODIFY句を指定すると、既存の列またはパーティションの定義を変更できます。

関連項目:

- オブジェクトの作成の詳細は、『Oracle Database管理者ガイド』および「CREATE TYPE」を参照してください。
- 表の変更および削除の詳細は、「ALTER TABLE」および「DROP TABLE」を参照してください。

前提条件

自分のスキーマ内にリレーショナル表を作成する場合は、CREATE TABLEシステム権限が必要です。他のユーザーのスキーマ内に表を作成する場合は、CREATE ANY TABLEシステム権限が必要です。また、表が含まれるスキーマの所有者は、表を格納するため表領域への割当て制限またはUNLIMITED TABLESPACEシステム権限が必要です。

これらの表権限に加え、オブジェクト表またはオブジェクト型の列が存在するリレーショナル表を作成する場合は、表の所有者に、表が参照するすべての型にアクセスするためのEXECUTEオブジェクト権限が付与されているかまたはEXECUTE ANY TYPEシステム権限が付与されている必要があります。これらの権限は、ロールを介して取得するのではなく、明示的に付与される必要があります。

さらに、表の所有者が表へのアクセス権限を他のユーザーに付与する場合、所有者には、参照する型に対するWITH GRANT OPTION付きのEXECUTEオブジェクト権限またはWITH ADMIN OPTION付きのEXECUTE ANY TYPEシステム権限が必要です。これらの権限を持っていない場合、表の所有者は、表へのアクセス権限を他のユーザーに付与できません。

一意キー制約または主キー制約を有効にする場合は、表に索引を作成するための権限が必要です。Oracle Databaseでは、表を含むスキーマにおいて、一意キーまたは主キーの列に索引を作成するため、この権限が必要になります。

evaluation_edition_clauseまたはunusable_editions_clauseでエディションを指定するには、そのエディションに対するUSE権限が必要になります。

zonemap_clauseを指定するには、ゾーン・マップを作成する権限が必要です。「CREATE MATERIALIZED ZONEMAP」の項目の「前提条件」の項を参照してください。

外部表を作成する場合は、適切なオペレーティング・システム・ディレクトリに対する、オペレーティング・システムの読取り権限および書込み権限が必要です。外部データが存在するオペレーティング・システム・ディレクトリに対応するデータベース・ディレクトリ・オブジェクトに対するREADオブジェクト権限が必要です。また、opaque_format_specでログ・ファイルまたは不良ファイルを指定する場合、またはAS subquery句を指定してデータベース表から外部表にデータをアンロードする場合、ファイルが格納されるデータベース・ディレクトリに対するWRITEオブジェクト権限が必要です。

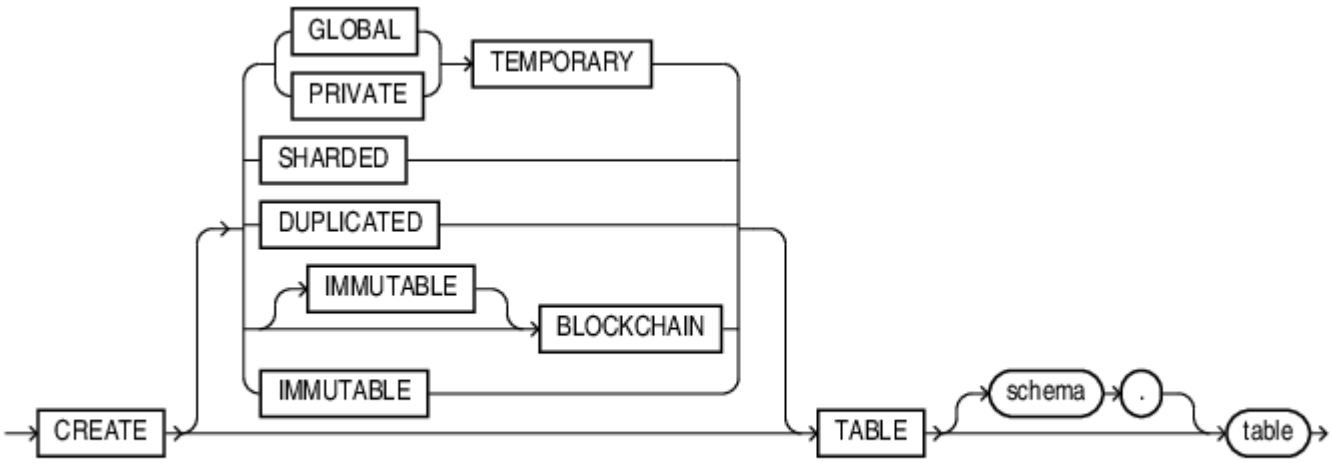
自分のスキーマ以外のデータベース・スキーマ内にXMLType表を作成する場合は、CREATE ANY TABLE権限のみでなく、CREATE ANY INDEX権限も必要です。これは、表の作成時に列OBJECT_IDに一意索引が作成されるためです。列OBJECT_IDには、システム生成のオブジェクト識別子が格納されます。

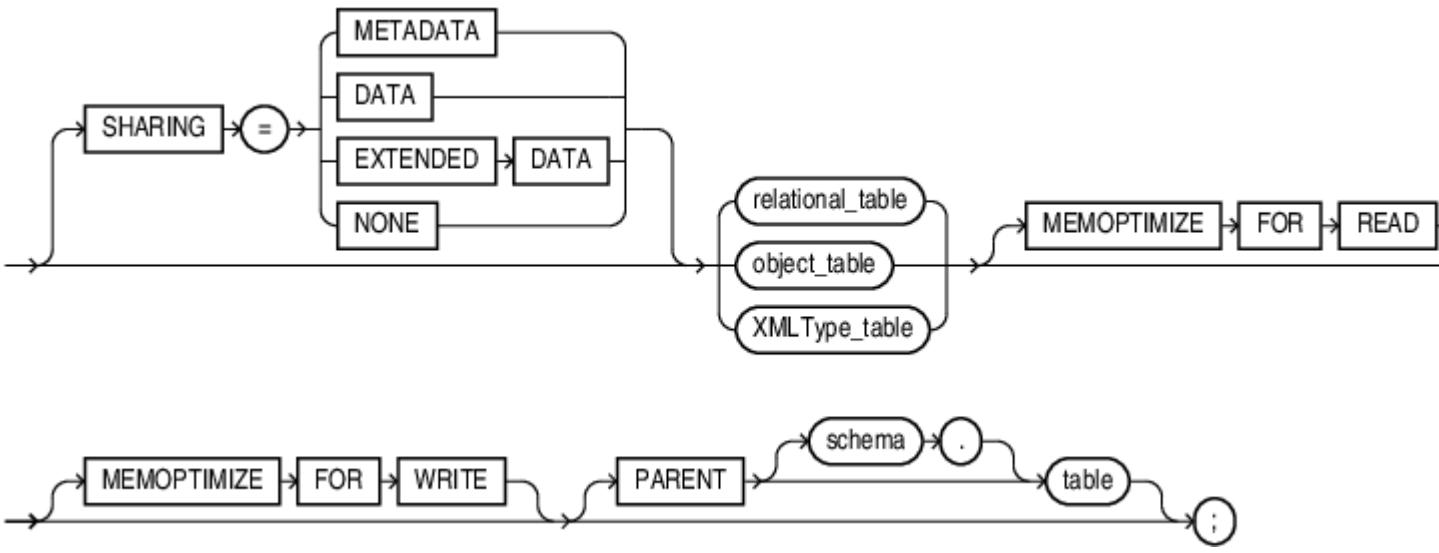
関連項目:

- CREATE INDEX
- 型を使用する表の作成に必要な権限については、『Oracle Database管理者ガイド』を参照してください。

構文

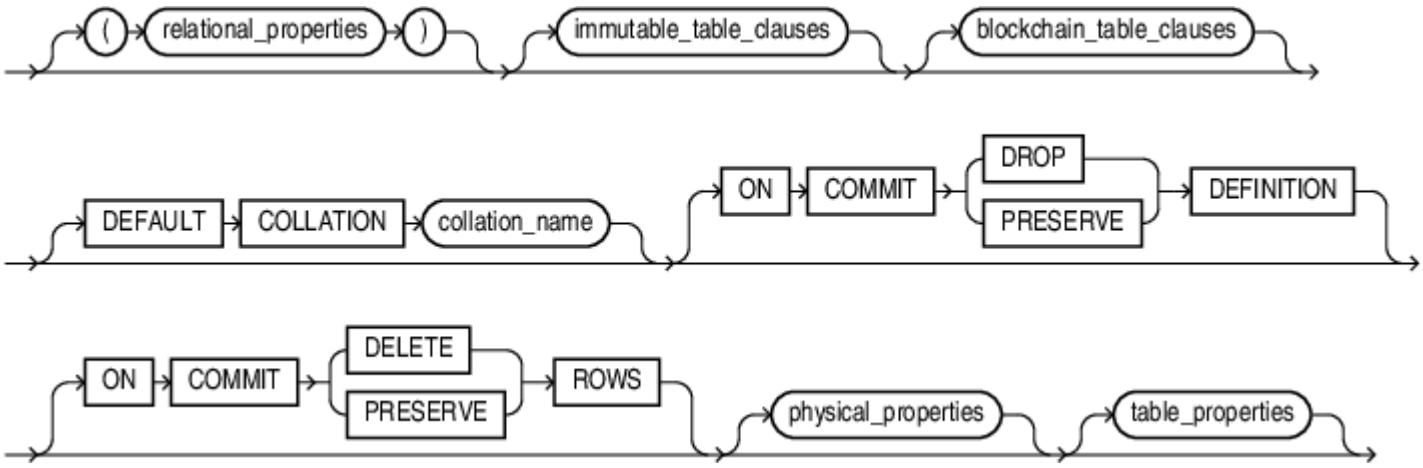
create_table::=





(relational_table::=、object_table::=、XMLType_table::=)

relational_table::=

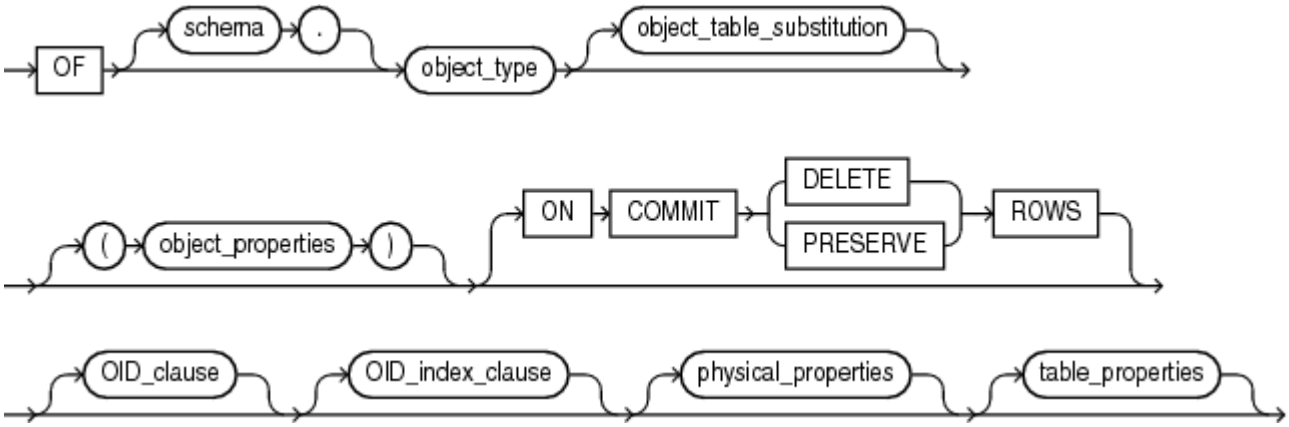


ノート:

表名の後の各句は、任意のリレーショナル表に対して任意で指定します。ただし、すべての表に対して少なくとも、`relational_properties` 句を使用して列名およびデータ型を指定するか、`table_properties` 句を使用して `AS subquery` 句を指定する必要があります。

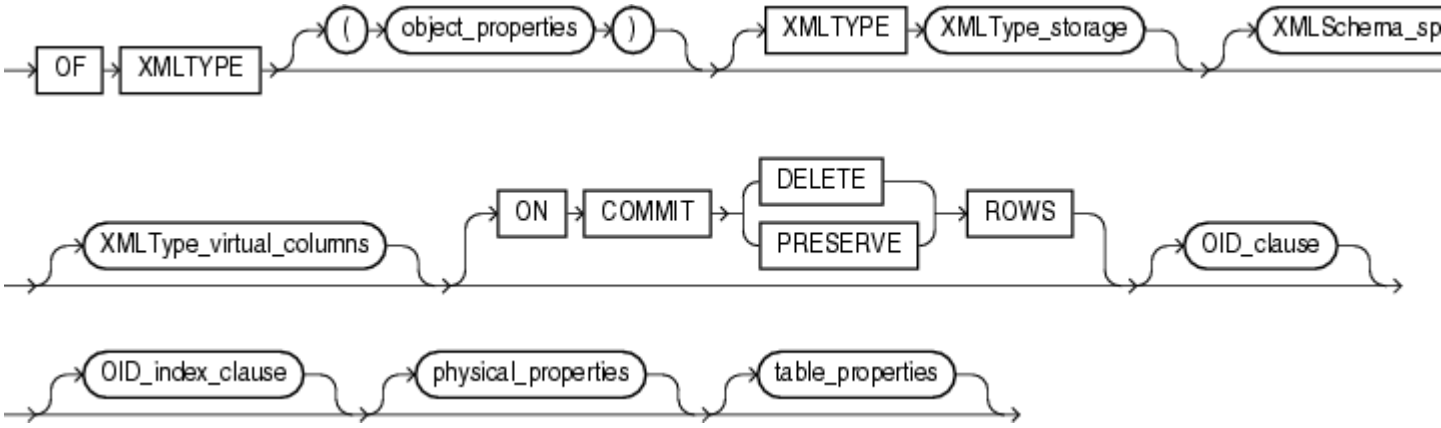
(relational_properties::=, , blockchain_table_clauses ::=,physical_properties::=, table_properties::=)

object_table::=



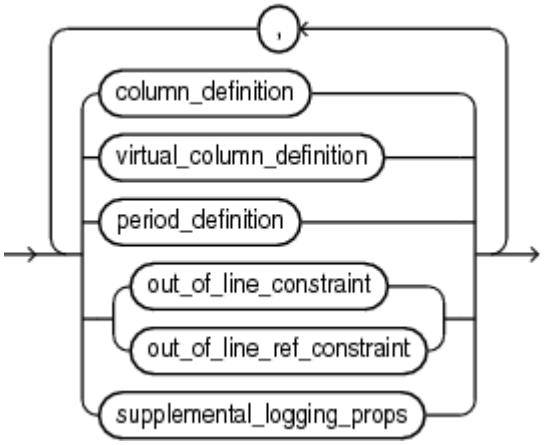
(object_table_substitution::=、object_properties::=、oid_clause::=、oid_index_clause::=、physical_properties::=、table_properties::=)

XMLType_table::=



(XMLType_storage::=、XMLSchema_spec::=、XMLType_virtual_columns::=、oid_clause::=、oid_index_clause::=、physical_properties::=、table_properties::=)

relational_properties::=

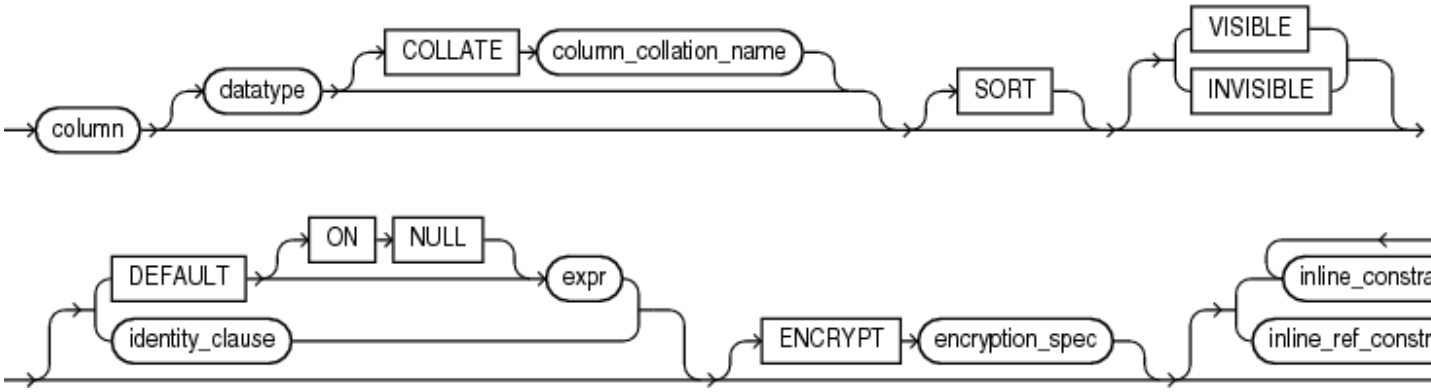


ノート:

これらの句はどのような順序で指定してもかまいませんが、次の例外があります。`period_definition` を指定する場合、その前に 1 つ以上の `column_definition` または `virtual_column_definition` を指定する必要があります。`period_definition` は、1 回のみ指定できます。

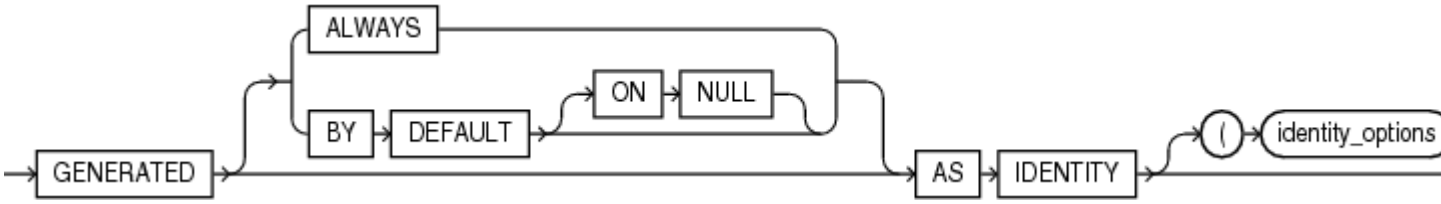
(column_definition::=、virtual_column_definition::=、period_definition::=、constraint::=、supplemental_logging_props::=)

column_definition::=

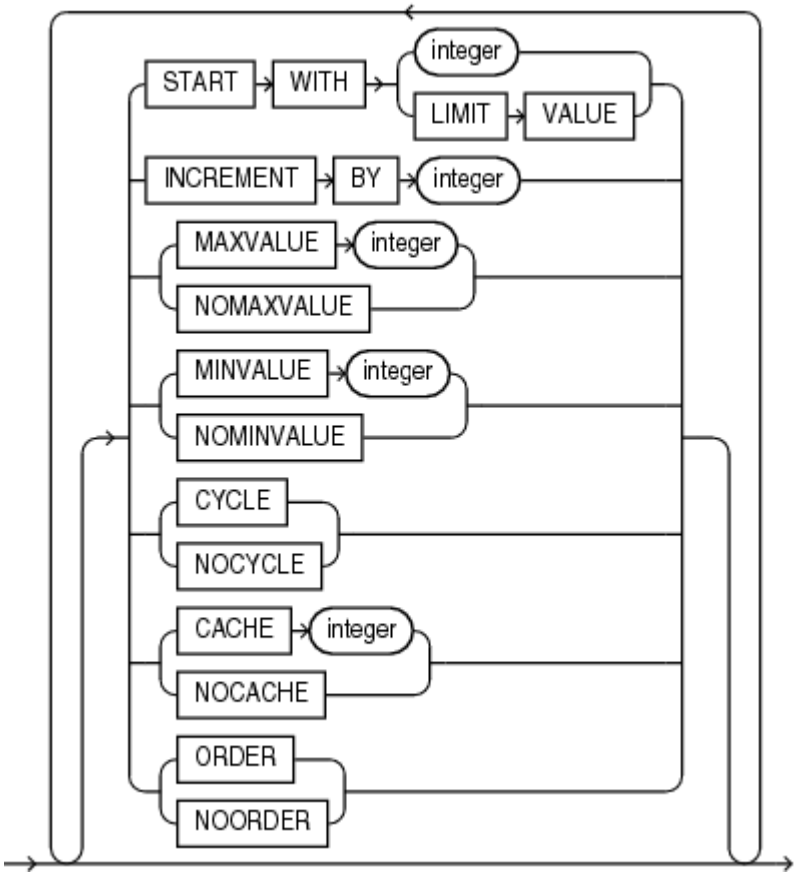


(identity_clause::=、encryption_spec::=、constraint::=)

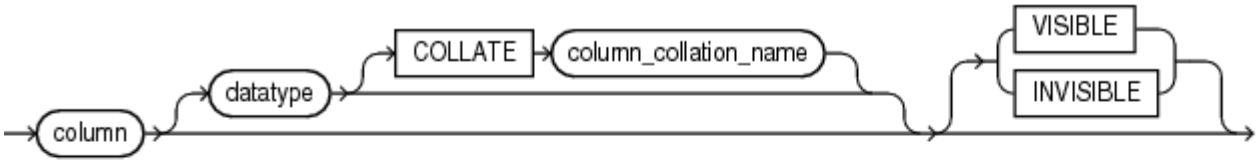
identity_clause::=

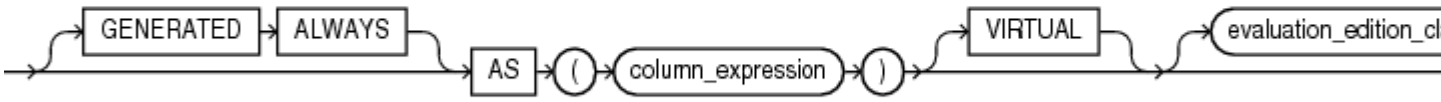


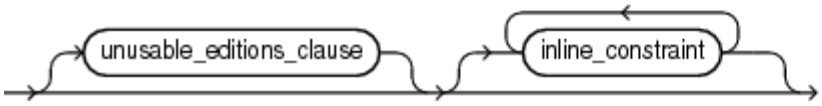
identity_options::=



virtual_column_definition::=

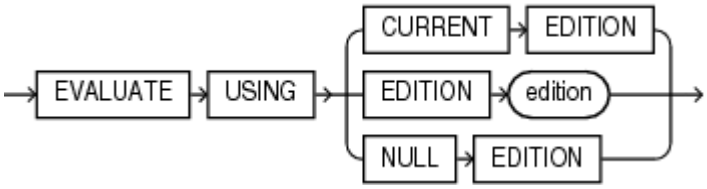




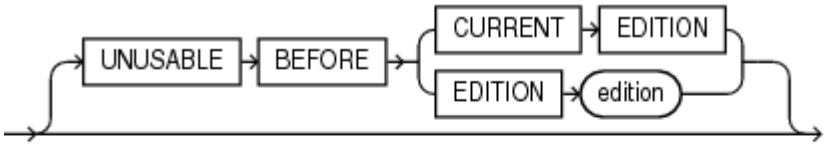


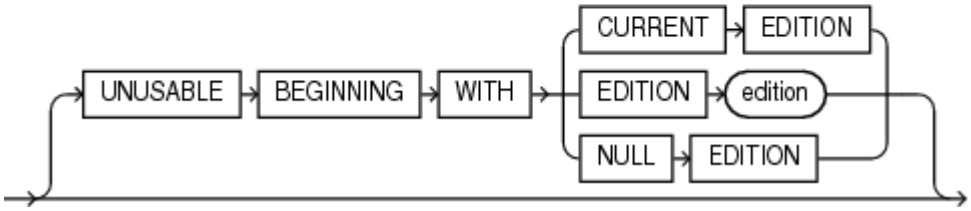
(evaluation_edition_clause::=、unusable_editions_clause::=、constraint::=)

evaluation_edition_clause::=

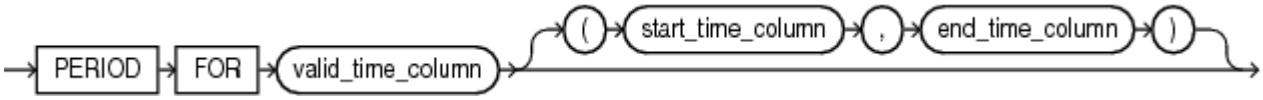


unusable_editions_clause::=

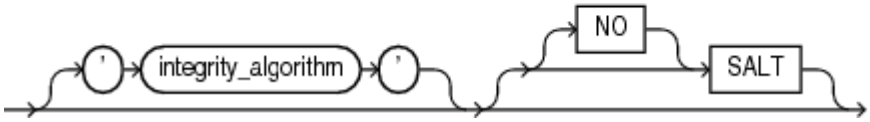




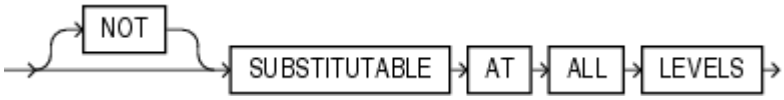
period_definition::=



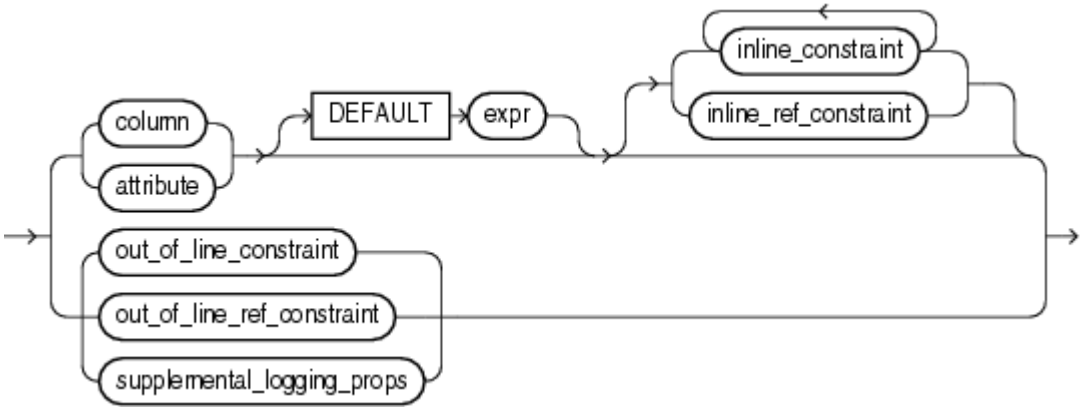
encryption_spec::=



object_table_substitution::=

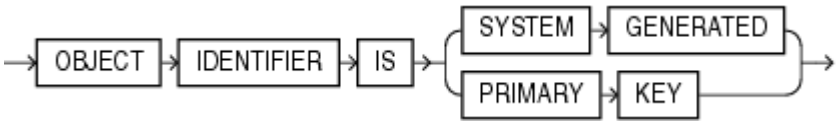


object_properties::=

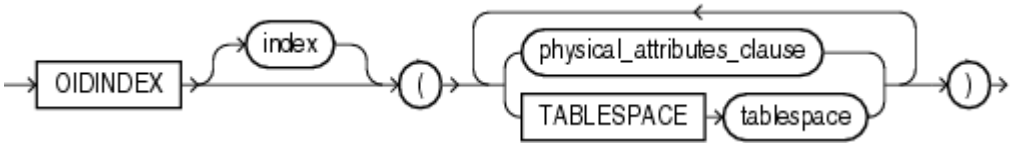


(constraint::=、supplemental_logging_props::=)

oid_clause::=

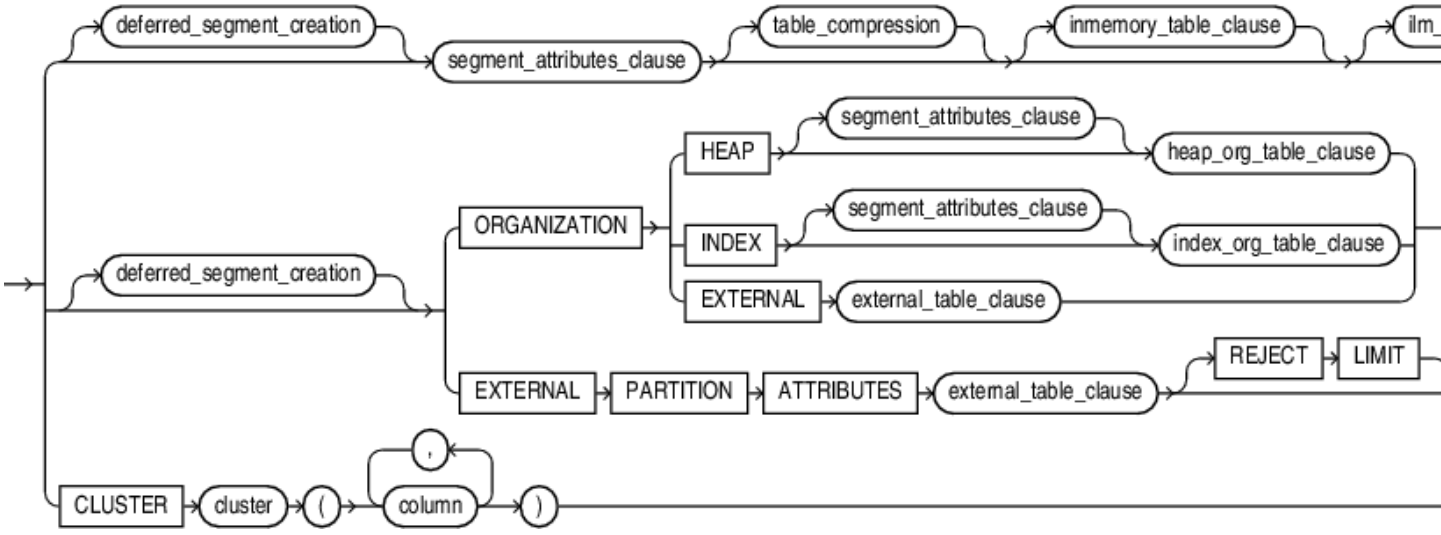


oid_index_clause::=



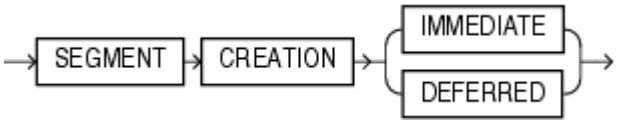
(physical_attributes_clause::=)

physical_properties::=

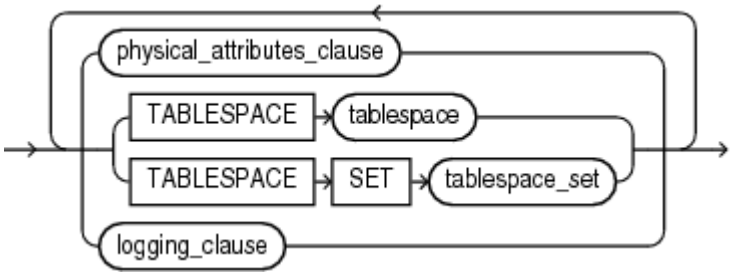


(deferred_segment_creation::=、segment_attributes_clause::=、table_compression::=、inmemory_table_clause::=、ilm_clause::=、heap_org_table_clause::=、index_org_table_clause::=、external_table_clause::=)

deferred_segment_creation::=

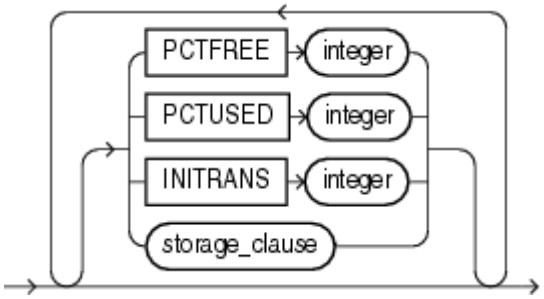


segment_attributes_clause::=



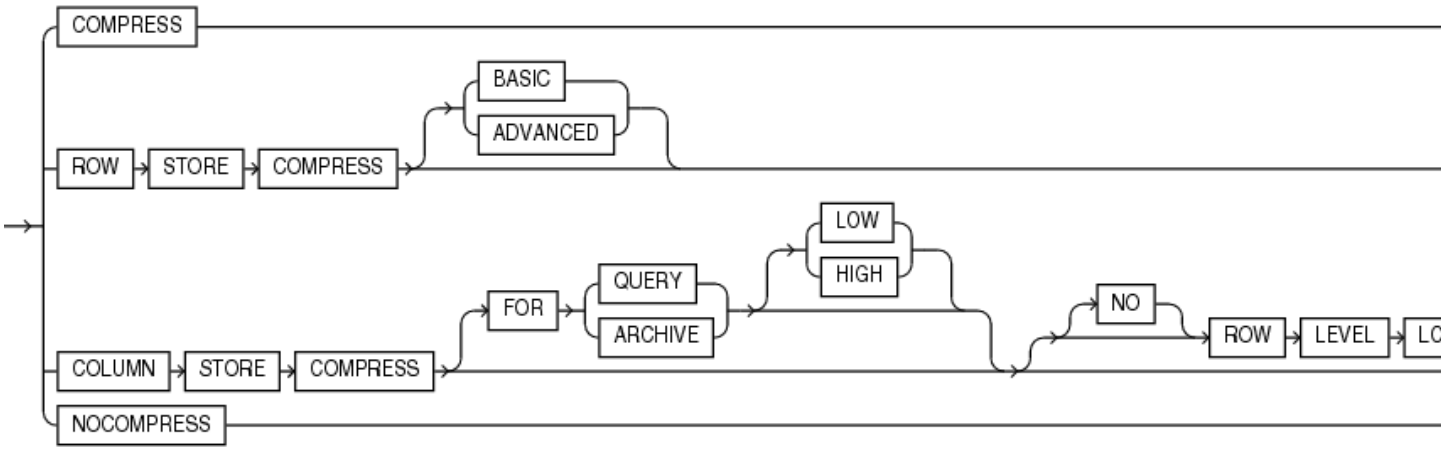
(physical_attributes_clause::=、logging_clause::=)

physical_attributes_clause::=

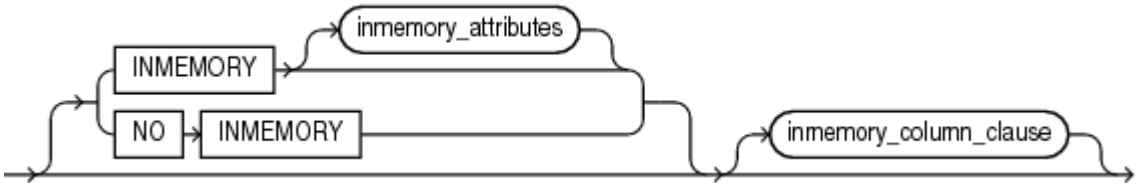


(storage_clause::=)

table_compression::=



inmemory_table_clause::=

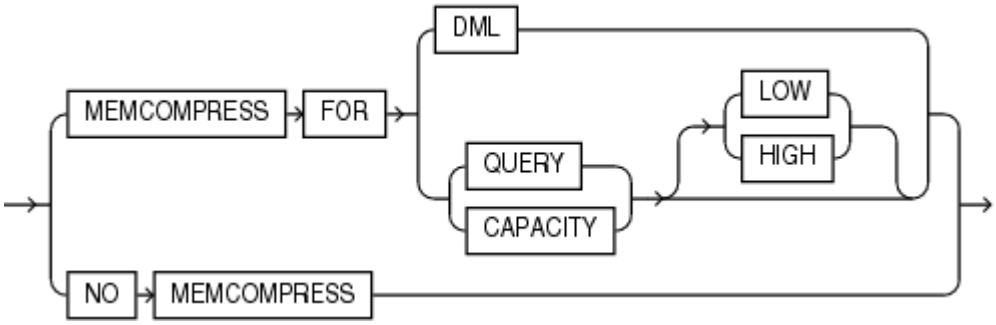


(inmemory_attributes::=、inmemory_column_clause::=)

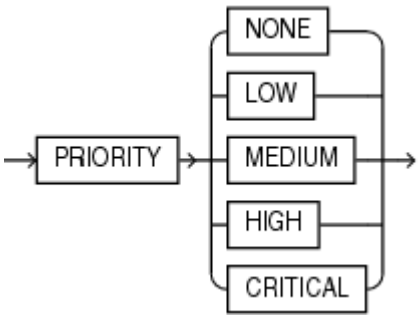
inmemory_attributes::=

(inmemory_memcompress::=、inmemory_priority::=、inmemory_distribute::=、inmemory_duplicate::=)

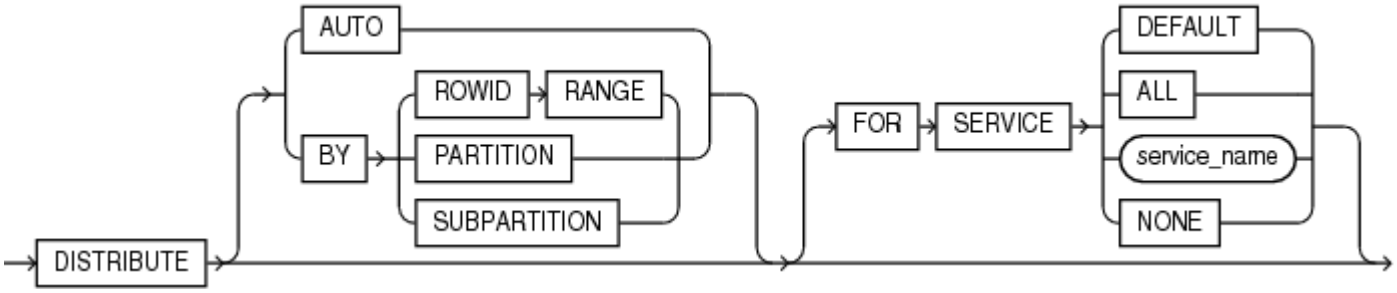
inmemory_memcompress::=



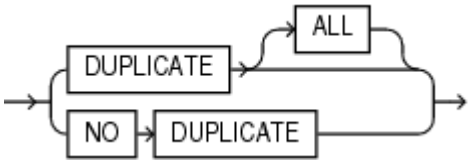
inmemory_priority::=



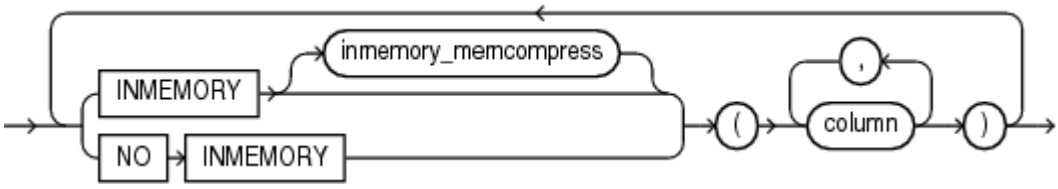
inmemory_distribute::=



inmemory_duplicate::=

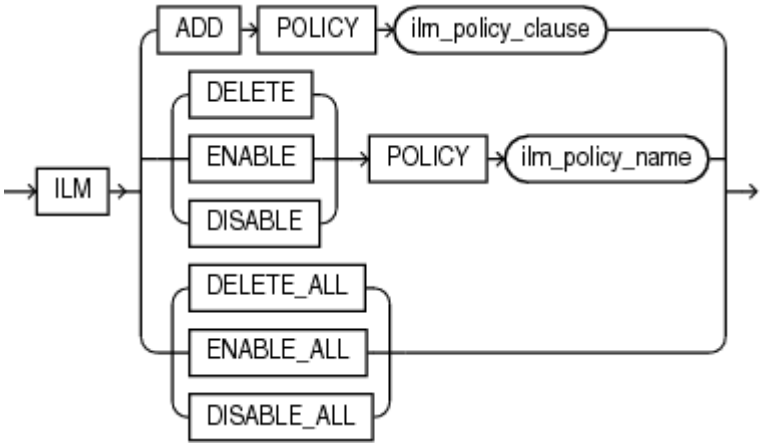


inmemory_column_clause::=

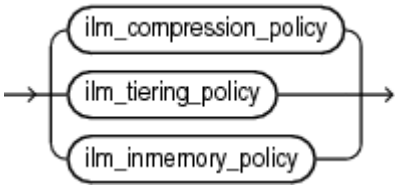


(inmemory_memcompress::=)

ilm_clause::=

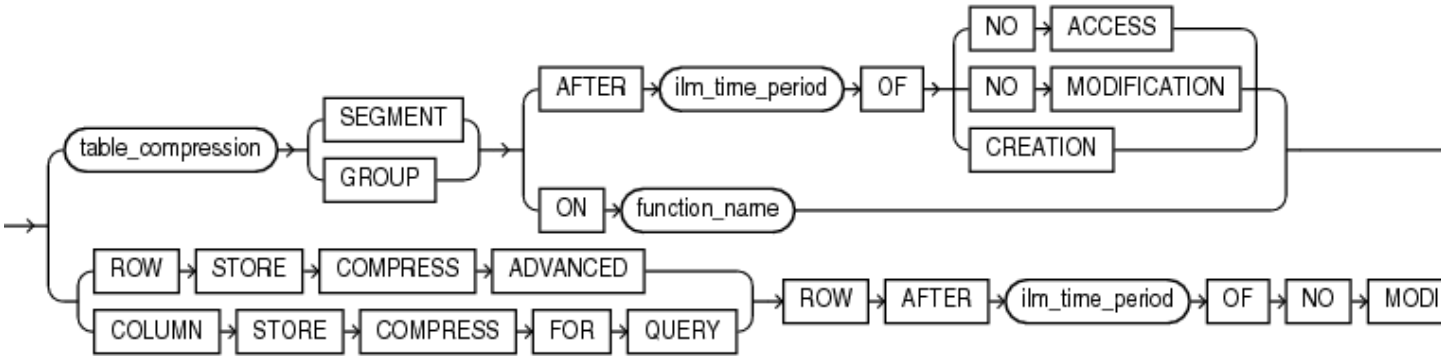


ilm_policy_clause::=



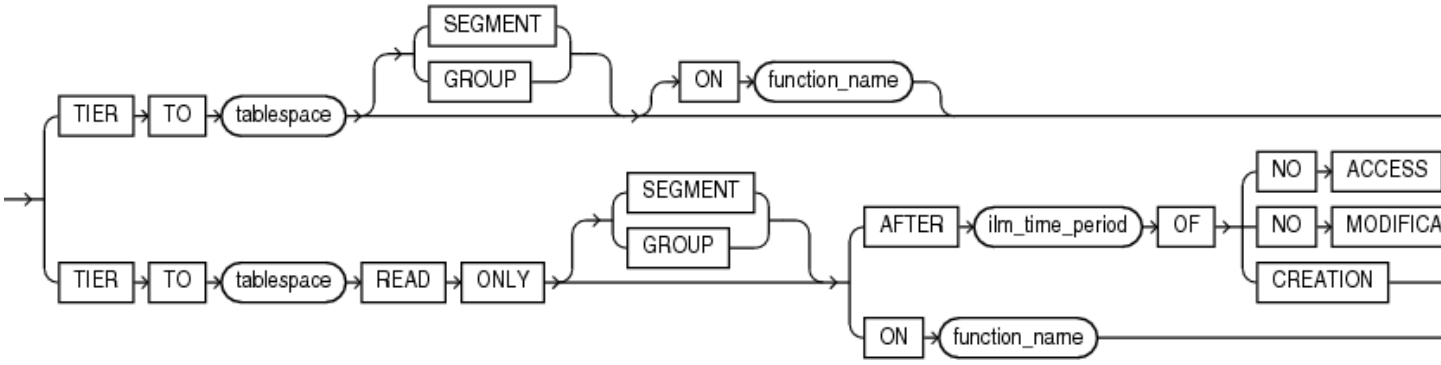
(ilm_compression_policy::=、ilm_tiering_policy::=、ilm_inmemory_policy::=)

ilm_compression_policy::=



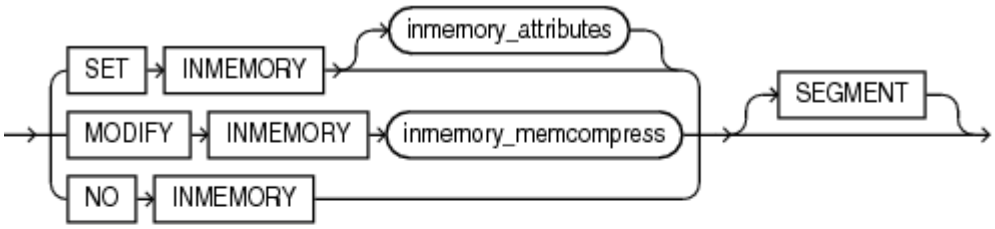
(table_compression::=、ilm_time_period::=)

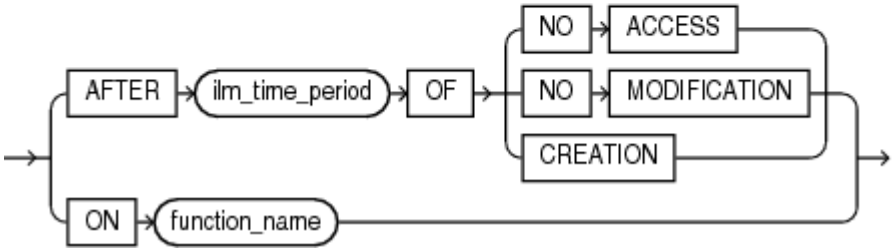
ilm_tiering_policy::=



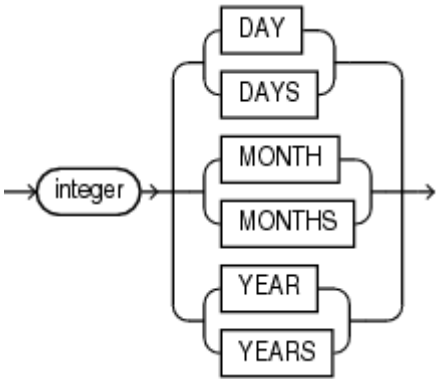
(ilm_time_period::=)

ilm_inmemory_policy::=

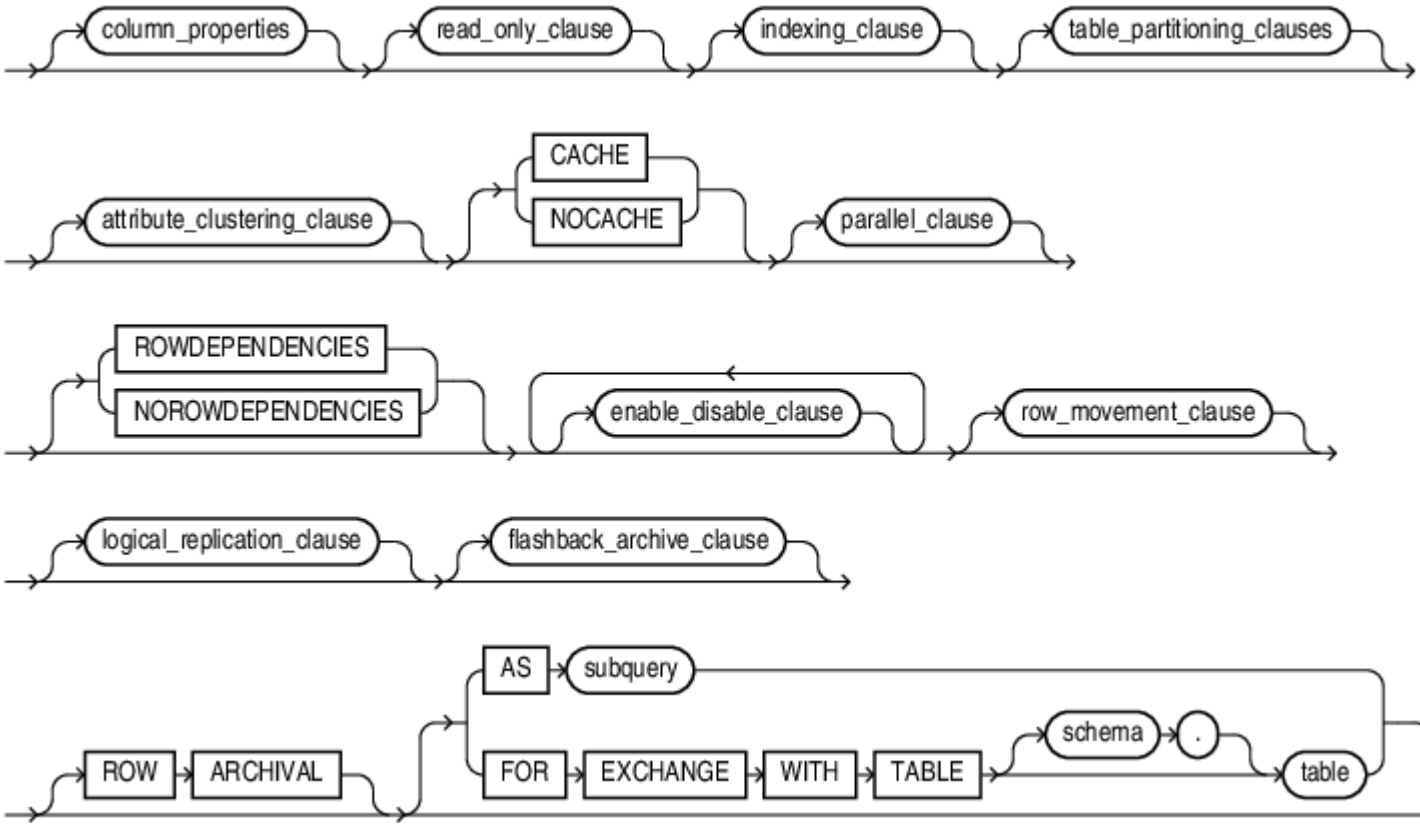




ilm_time_period::=

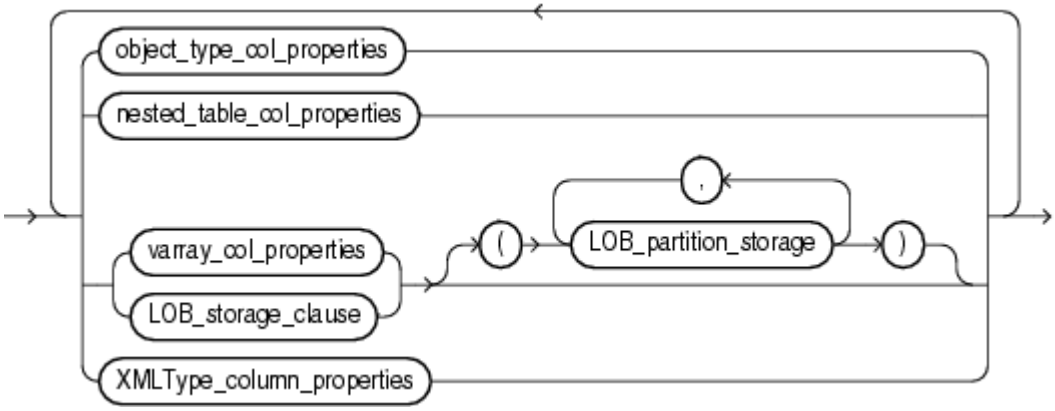


table_properties::=



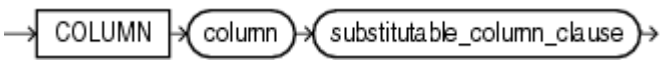
(column_properties::=、read_only_clause::=、indexing_clause::=、table_partitioning_clauses::=、attribute_clustering_clause::=、parallel_clause::=、enable_disable_clause::=、row_movement_clause::=、flashback_archive_clause::=、subquery::=)

column_properties::=

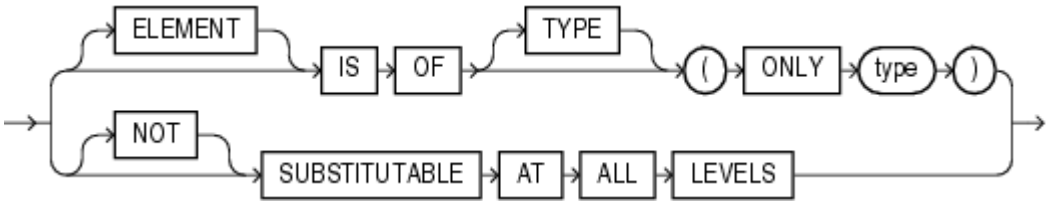


(object_type_col_properties::=、nested_table_col_properties::=、varray_col_properties::=、LOB_storage_clause::=、LOB_partition_storage::=、XMLType_column_properties::=)

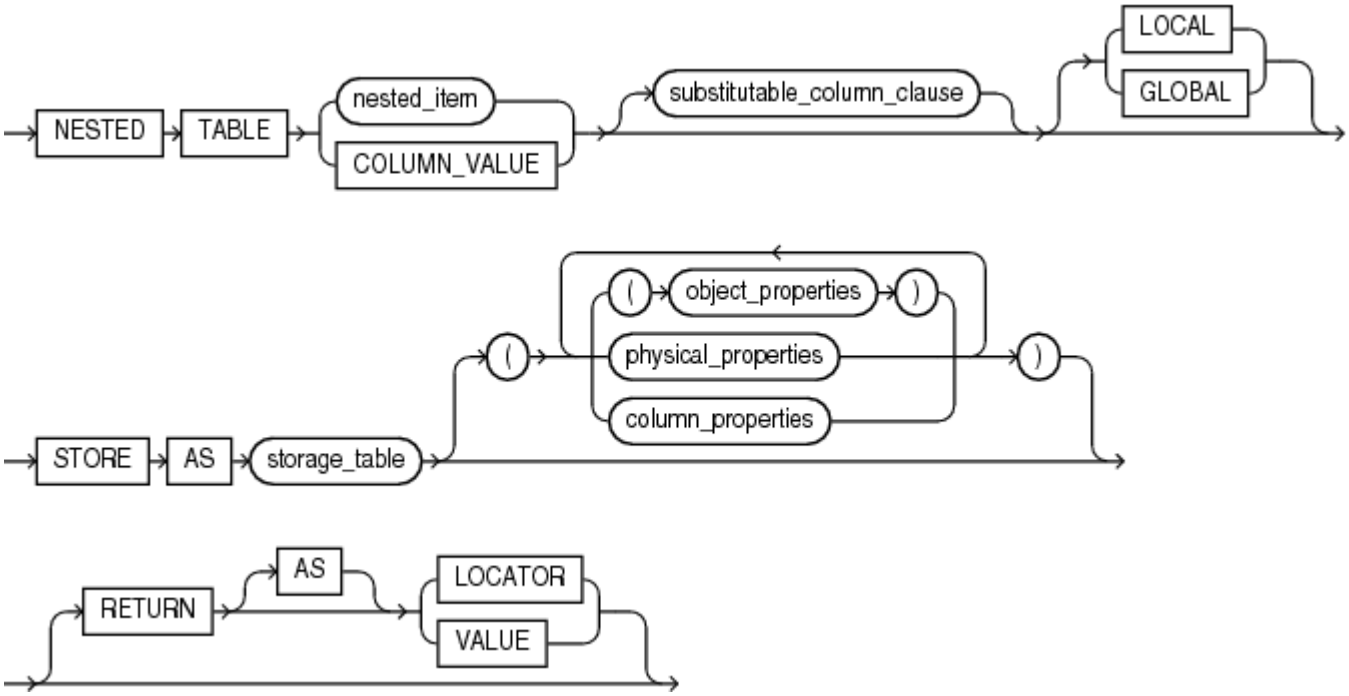
object_type_col_properties::=



substitutable_column_clause::=

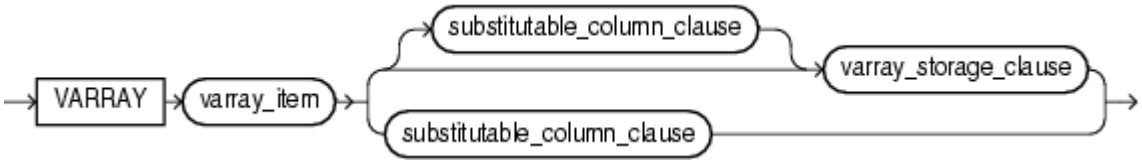


nested_table_col_properties::=



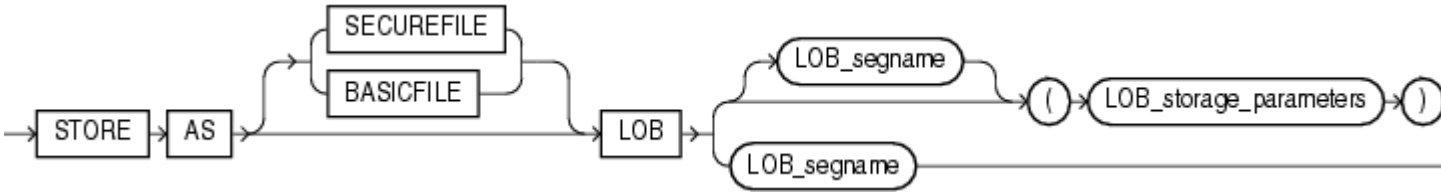
(substitutable_column_clause::=、object_properties::=、physical_properties::=、column_properties::=)

varray_col_properties::=



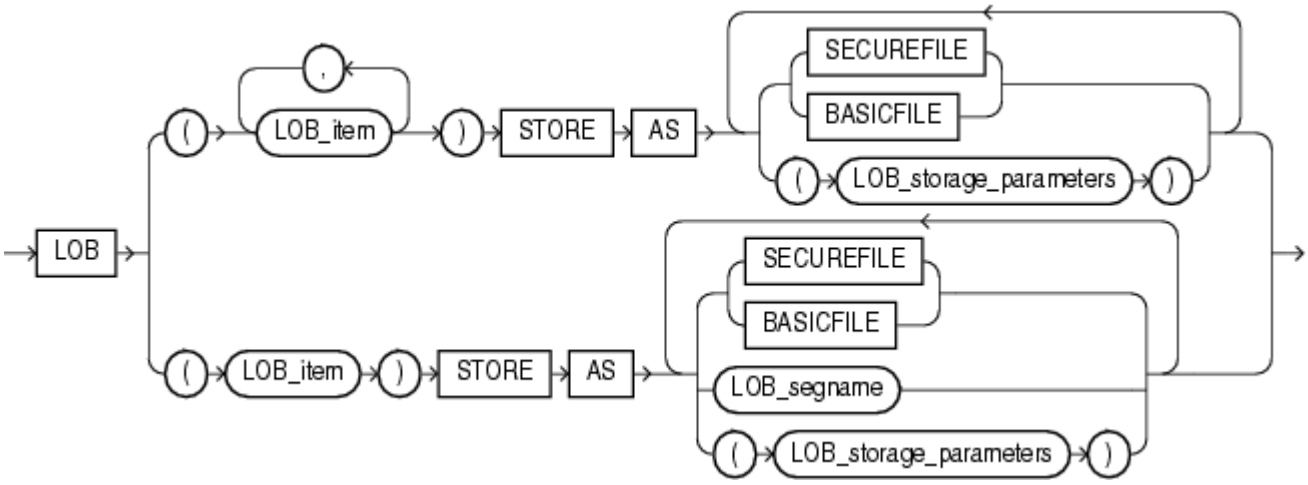
(substitutable_column_clause::=、varray_storage_clause::=)

varray_storage_clause::=



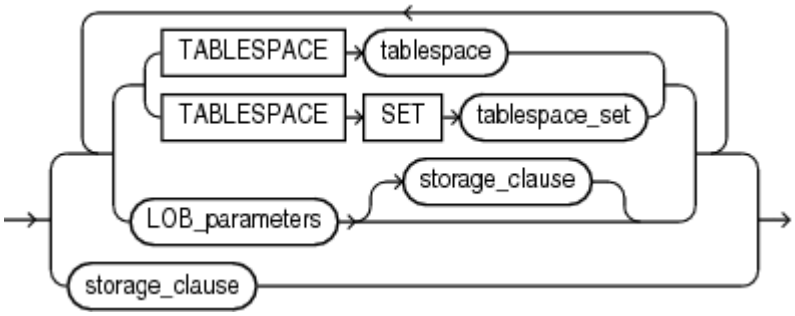
(LOB_parameters::=)

LOB_storage_clause::=



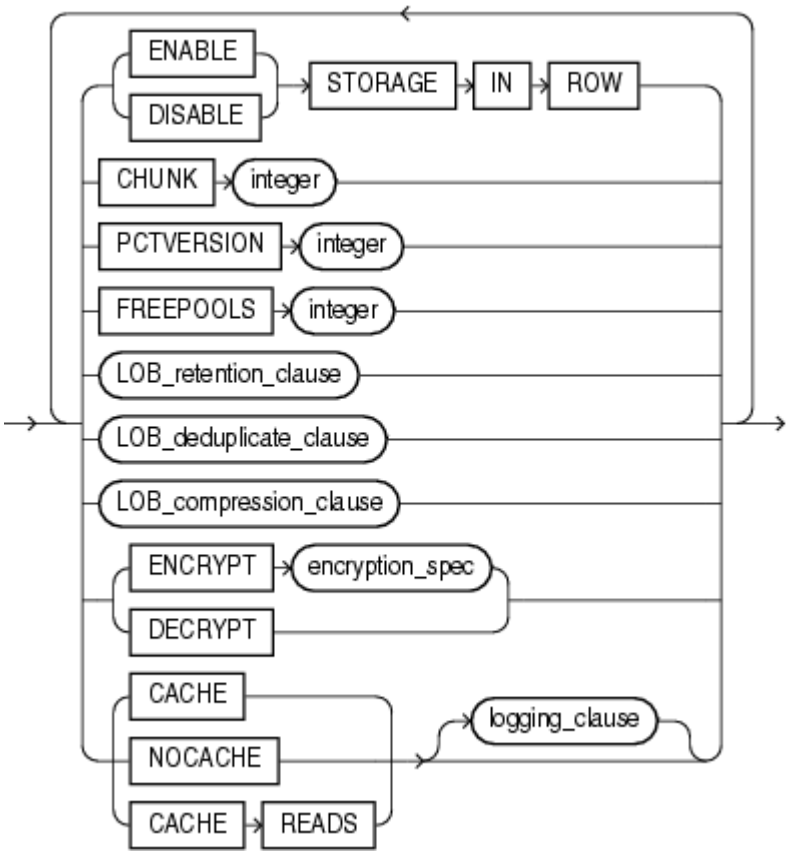
(LOB_storage_parameters::=)

LOB_storage_parameters::=



(LOB_parameters::=、storage_clause::=)

LOB_parameters::=

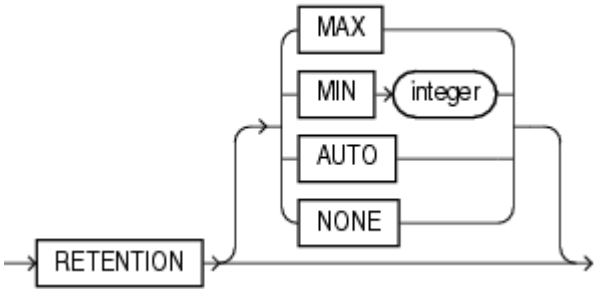


(LOB_deduplicate_clause::=、LOB_compression_clause::=、encryption_spec::=、logging_clause::=)

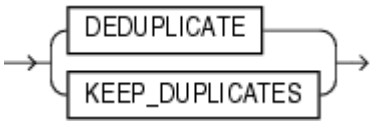
ノート:

LOB 記憶域に SecureFiles を使用する場合、いくつかの LOB パラメータは不要になります。詳細は、「LOB_storage_parameters」を参照してください。

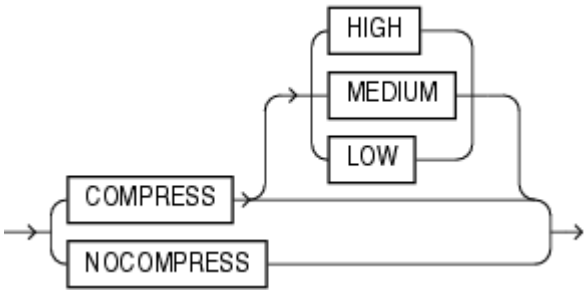
LOB_retention_clause::=



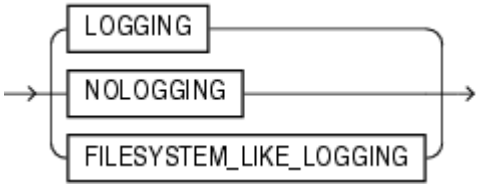
LOB_deduplicate_clause::=



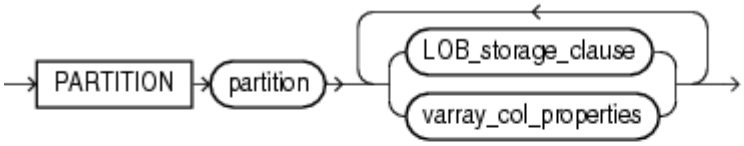
LOB_compression_clause::=

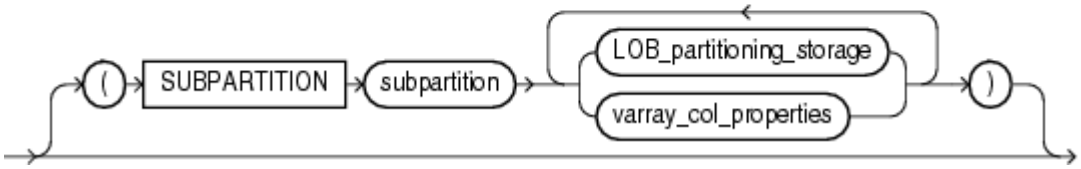


logging_clause::=



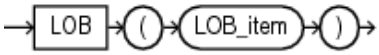
LOB_partition_storage::=

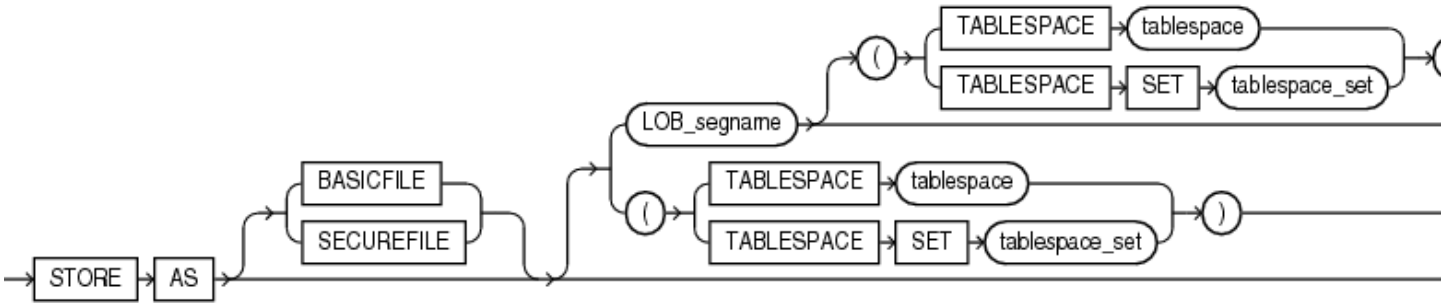




(LOB_storage_clause::=、varray_col_properties::=、LOB_partitioning_storage::=)

LOB_partitioning_storage::=

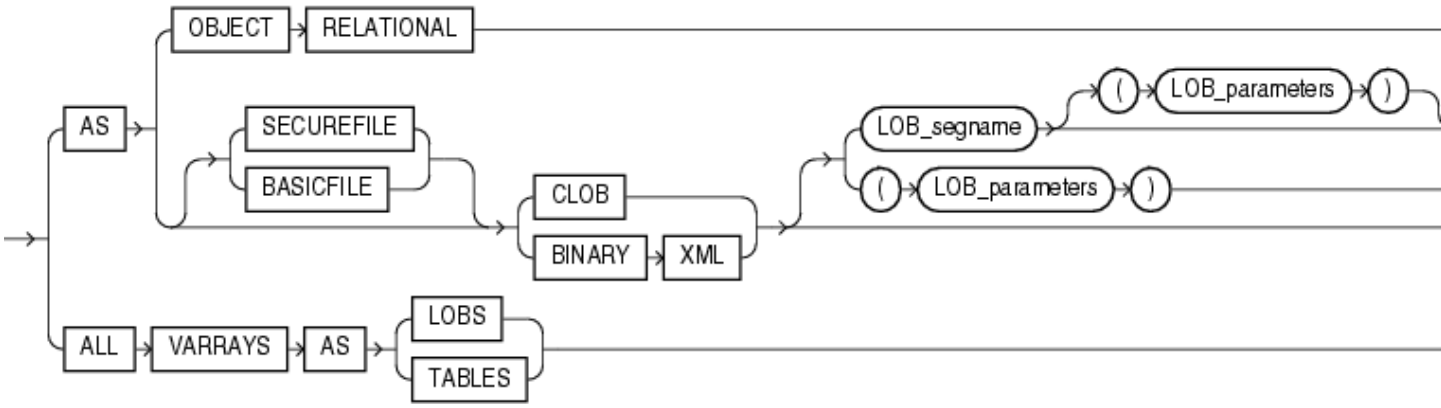




XMLType_column_properties::=

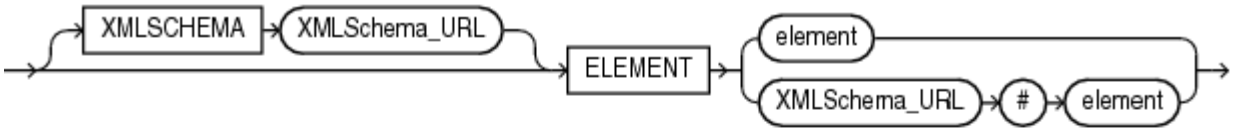
(XMLType_storage::=、XMLSchema_spec::=)

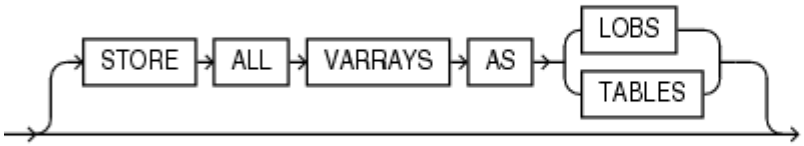
XMLType_storage::=

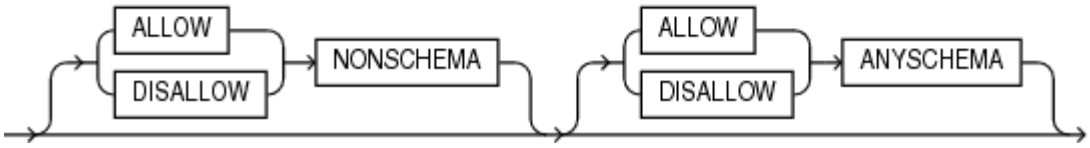


(LOB_parameters::=)

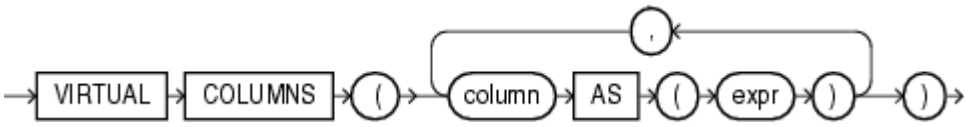
XMLSchema_spec::=



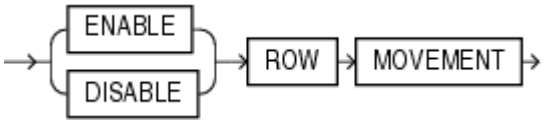




XMLType_virtual_columns::=

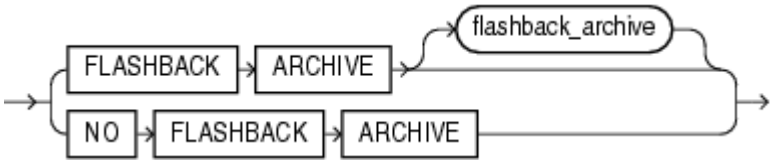


row_movement_clause::=

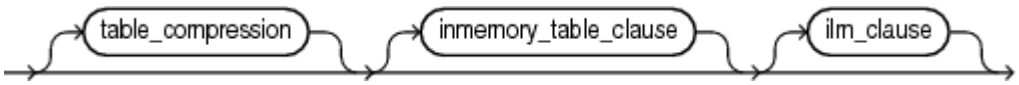


logical_replication_clause::=

flashback_archive_clause::=

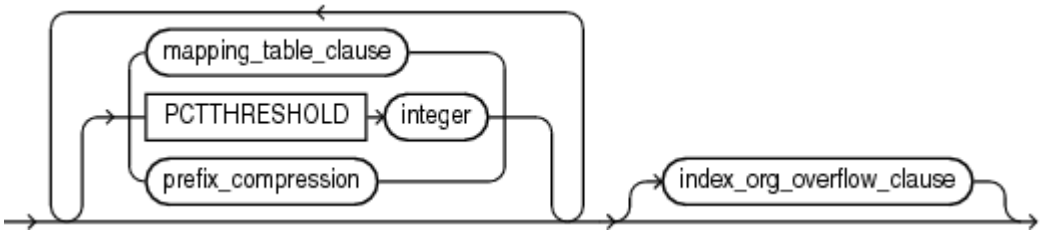


heap_org_table_clause::=



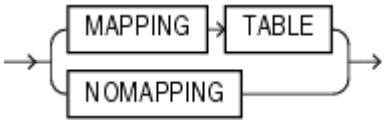
(table_compression::=、inmemory_table_clause::=、ilm_clause::=)

index_org_table_clause::=

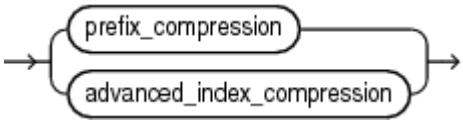


(mapping_table_clauses::=、prefix_compression::=、index_org_overflow_clause::=)

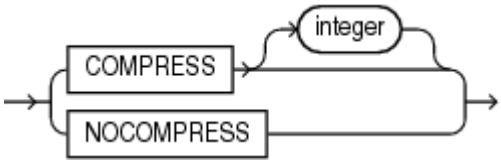
mapping_table_clauses::=



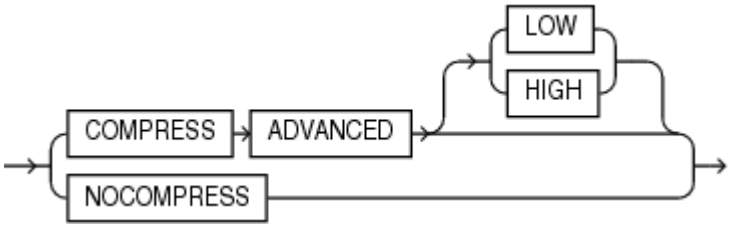
index_compression::=



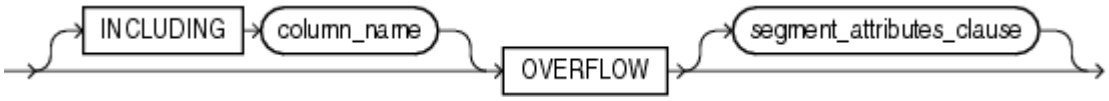
prefix_compression::=



advanced_index_compression::=

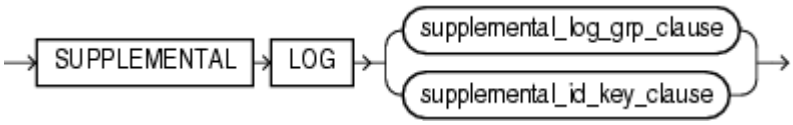


index_org_overflow_clause::=

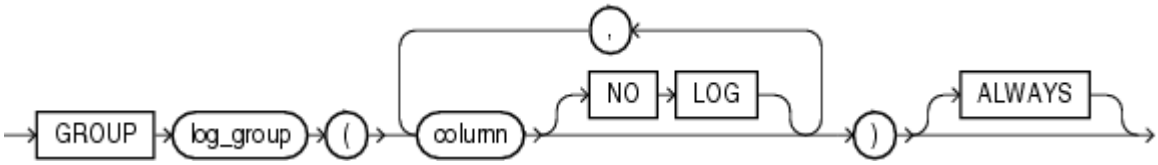


(segment_attributes_clause::=)

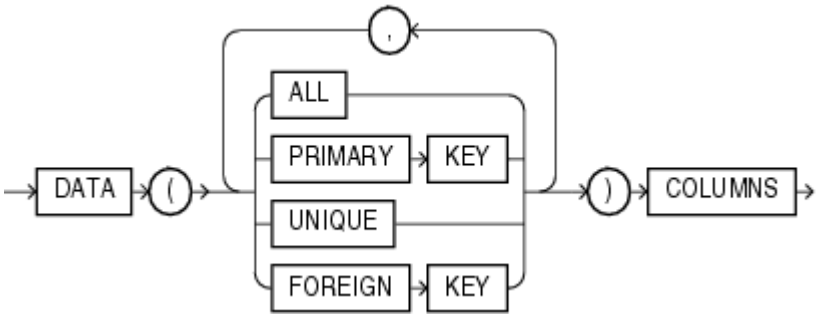
supplemental_logging_props::=



supplemental_log_grp_clause::=



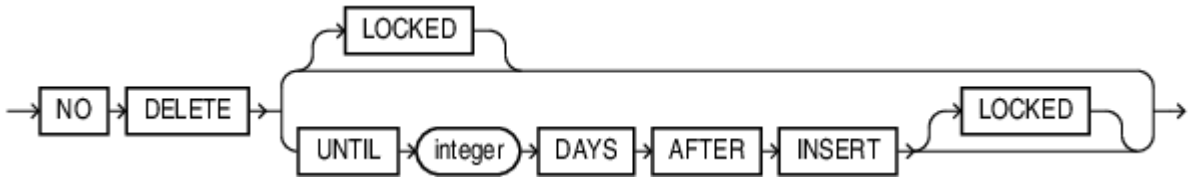
supplemental_id_key_clause::=を参照



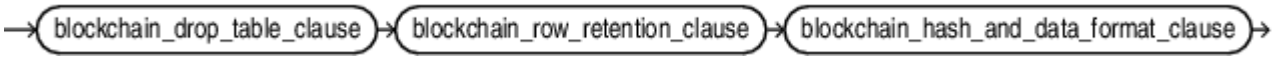
immutable_table_clauses::=

immutable_table_no_drop_clause::=

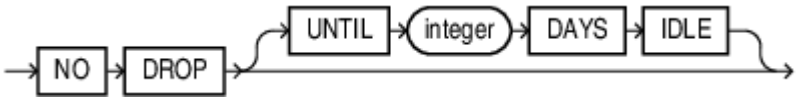
immutable_table_no_delete_clause::=



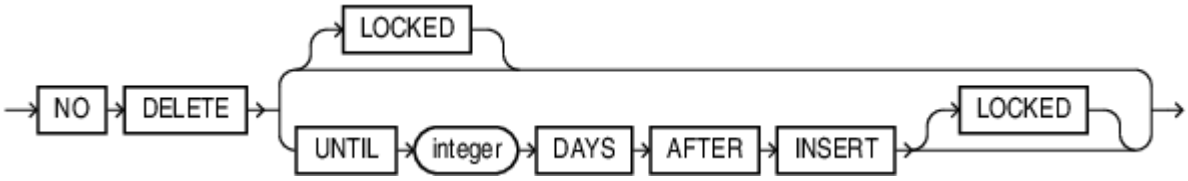
blockchain_table_clauses::=



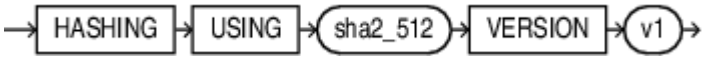
blockchain_drop_table_clause::=



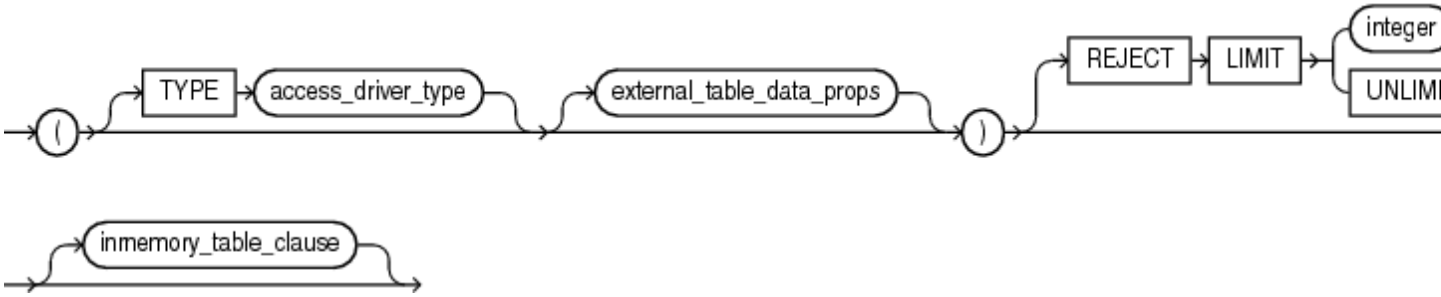
blockchain_row_retention_clause::=



blockchain_hash_and_data_format_clause::=

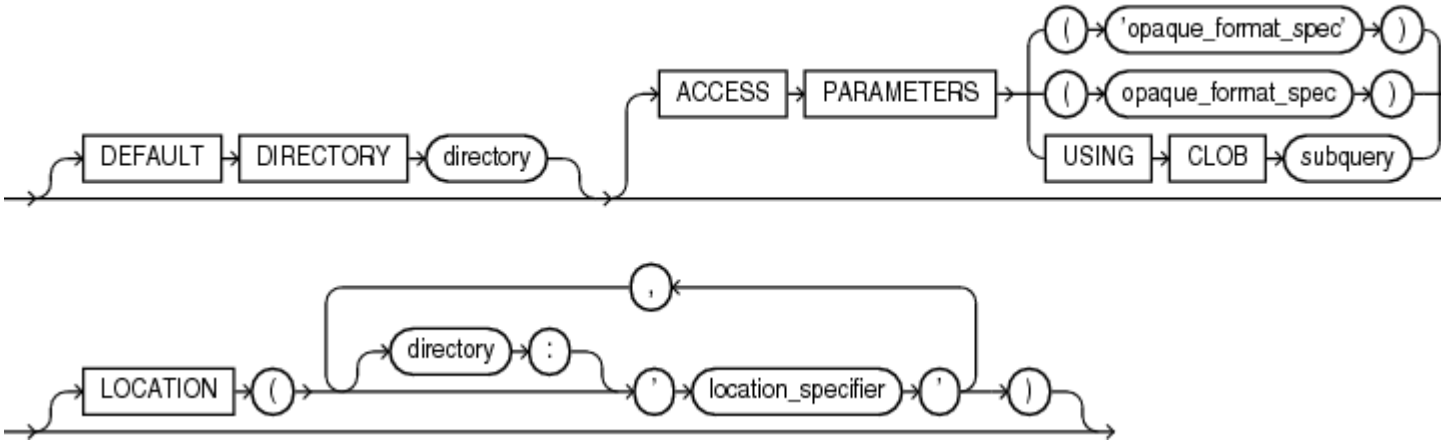


external_table_clause::=



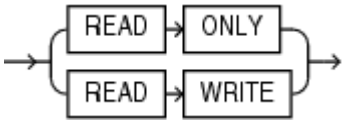
(external_table_data_props::=)

external_table_data_props::=

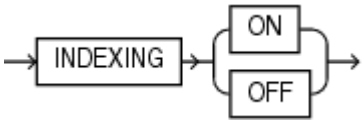


(opaque_format_spec: この句は、ORACLE_LOADER、ORACLE_DATAPUMP、ORACLE_HDFSおよびORACLE_HIVEアクセス・ドライバのすべてのアクセス・パラメータを指定します。これらのパラメータの詳細は、『Oracle Databaseユーティリティ』を参照してください。)

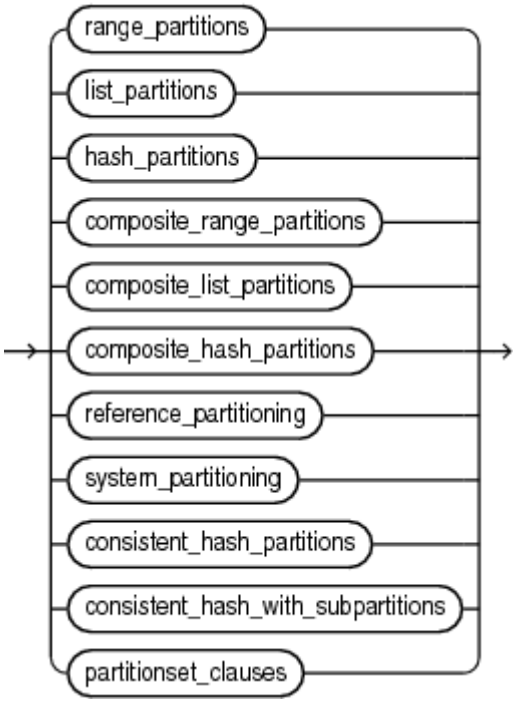
read_only_clause::=



indexing_clause::=

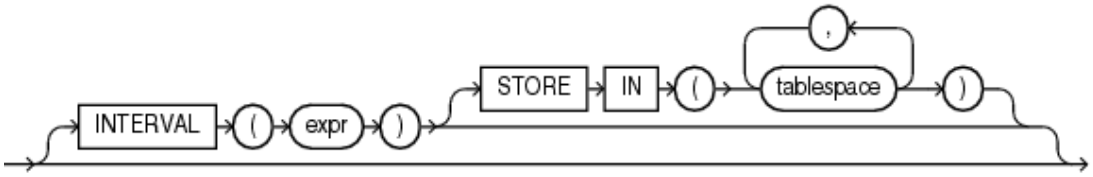


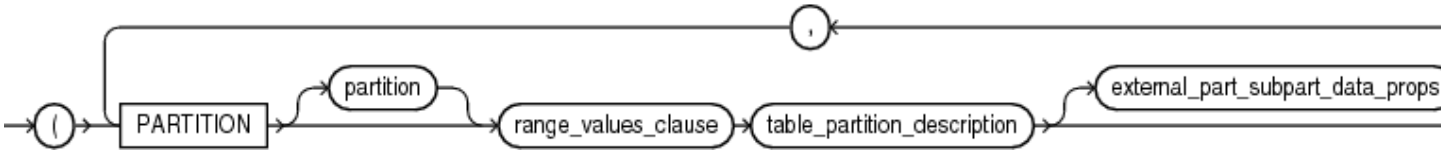
table_partitioning_clauses::=



(range_partitions::=、list_partitions::=、hash_partitions::=、composite_range_partitions::=、composite_list_partitions::=、composite_hash_partitions::=、reference_partitioning::=、system_partitioning::=、consistent_hash_partitions::=、consistent_hash_with_subpartitions::=、partitionset_clauses::=)

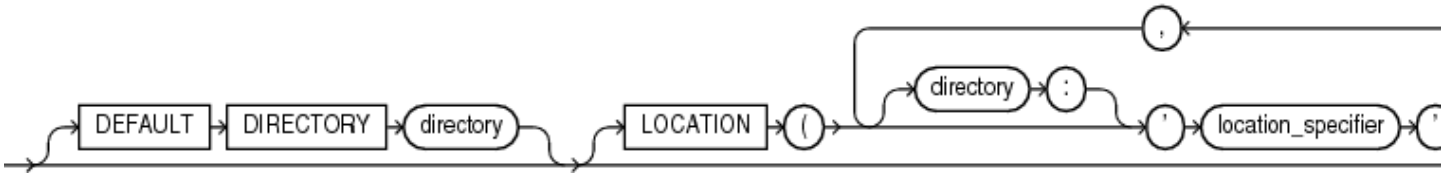
range_partitions::=



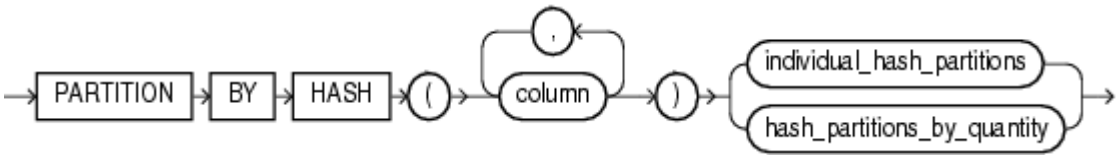


(range_values_clause::=、table_partition_description::=、external_part_subpart_data_props::=)

external_part_subpart_data_props::=

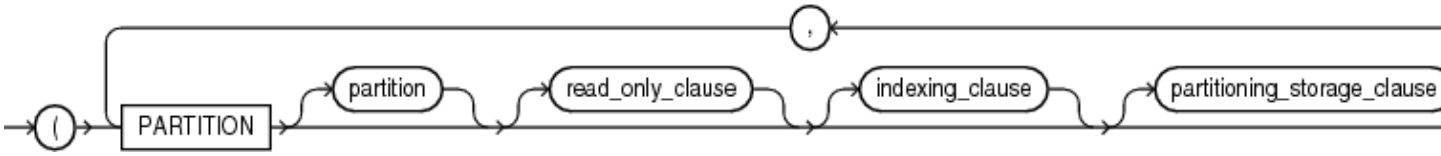


hash_partitions::=



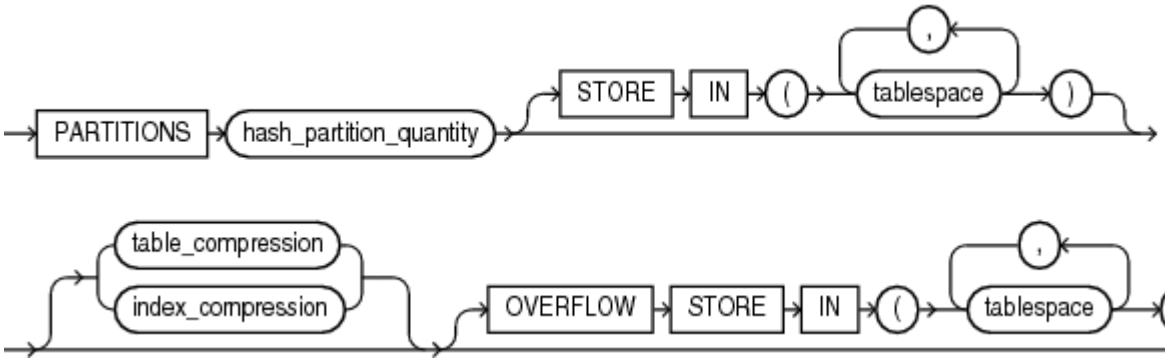
(individual_hash_partitions::=、hash_partitions_by_quantity::=)

individual_hash_partitions::=



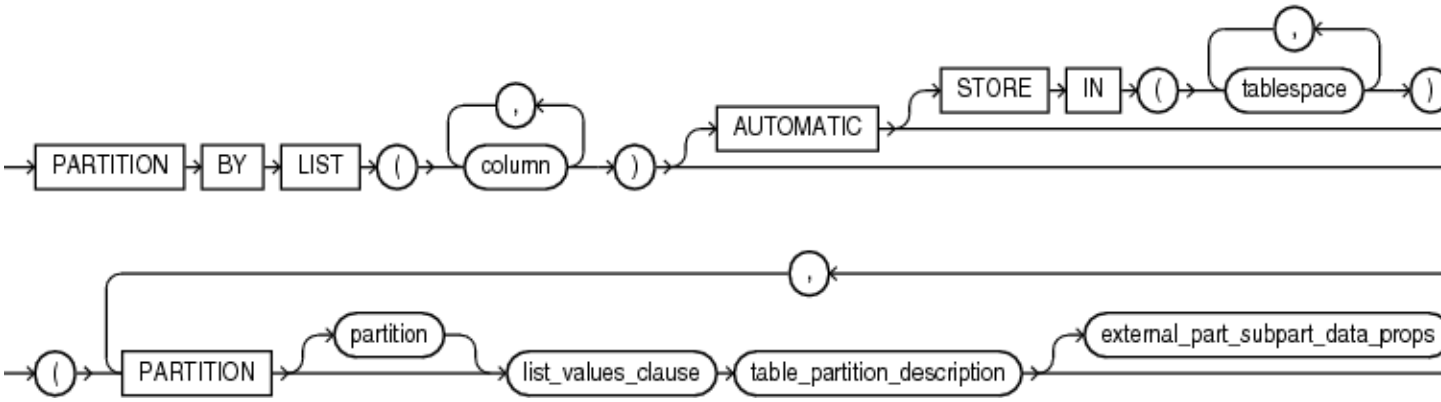
(read_only_clause::=、indexing_clause::=、partitioning_storage_clause::=)

hash_partitions_by_quantity::=を参照



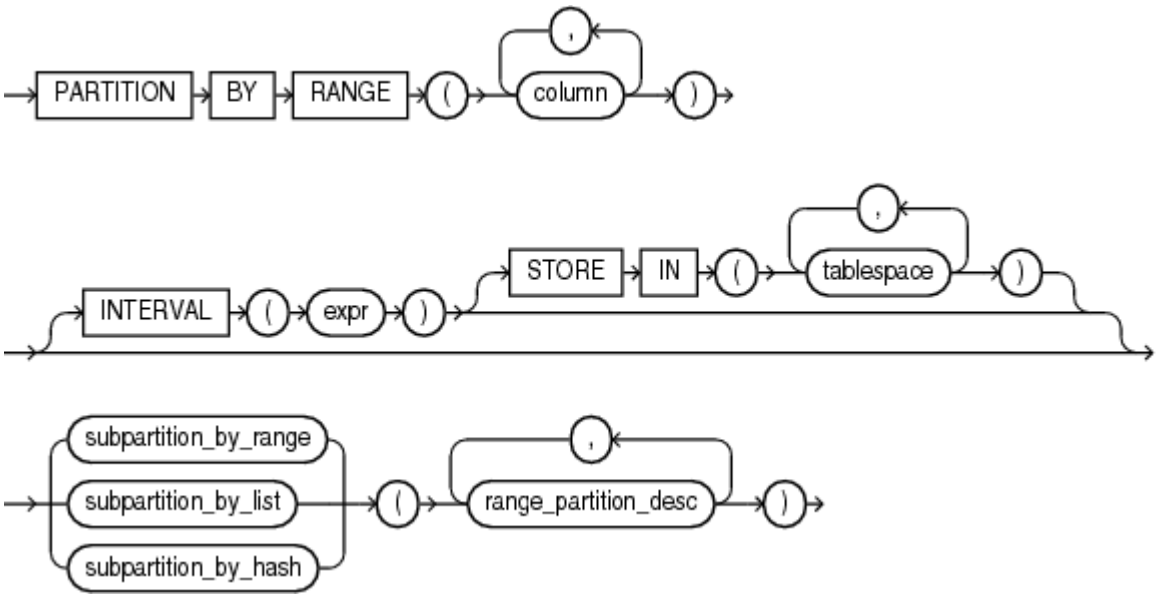
(table_compression::=、index_compression::=)

list_partitions::=



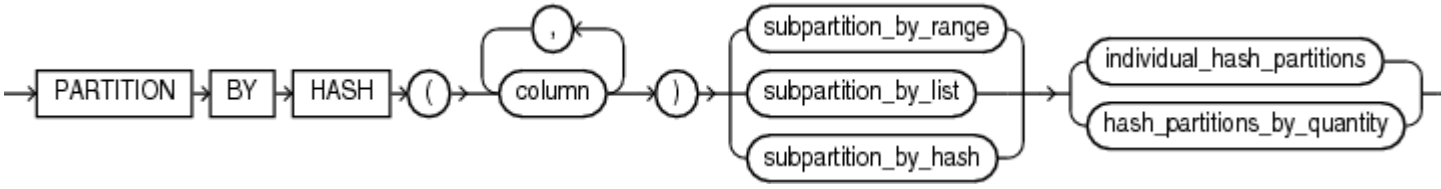
(list_values_clause::=、table_partition_description::=、external_part_subpart_data_props::=)

composite_range_partitions::=



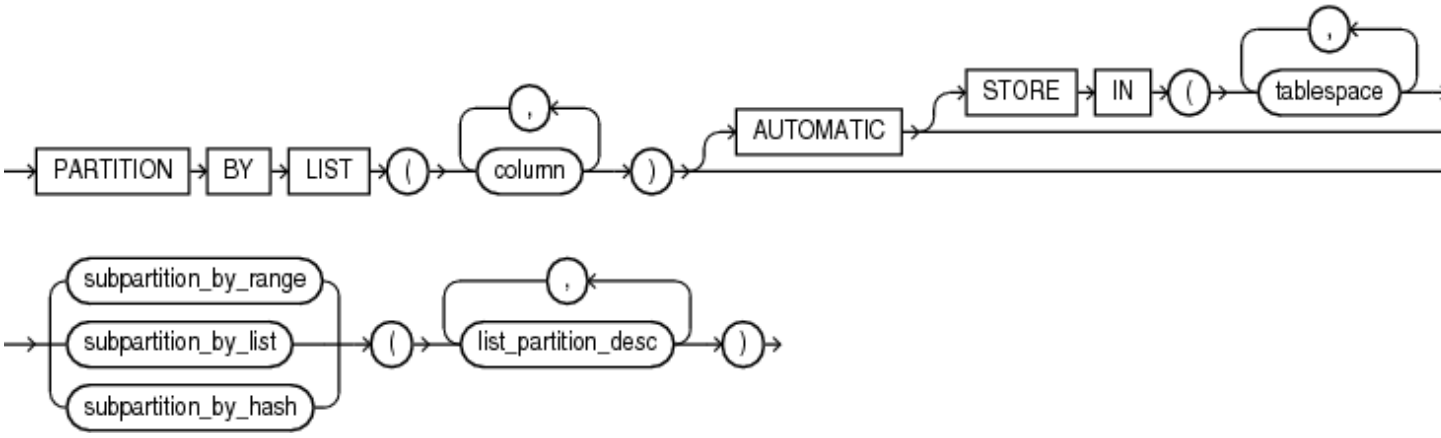
(subpartition_by_range::=、subpartition_by_list::=、subpartition_by_hash::=、range_partition_desc::=)

composite_hash_partitions::=



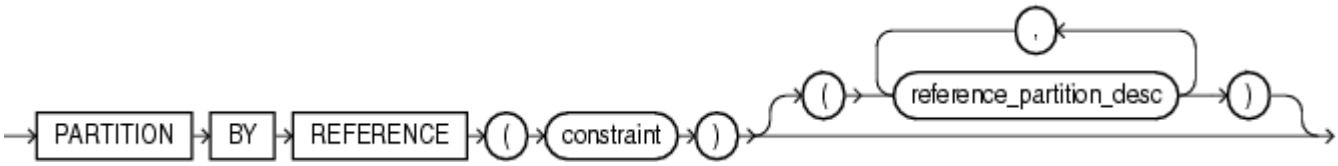
(subpartition_by_range::=、subpartition_by_list::=、subpartition_by_hash::=、individual_hash_partitions::=、hash_partitions_by_quantity::=)

composite_list_partitions::=



(subpartition_by_range::=、subpartition_by_list::=、subpartition_by_hash::=、list_partition_desc::=)

reference_partitioning::=

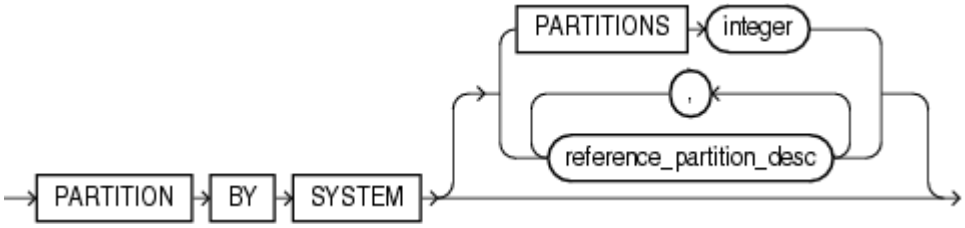


(constraint::=、reference_partition_desc::=)

reference_partition_desc::=

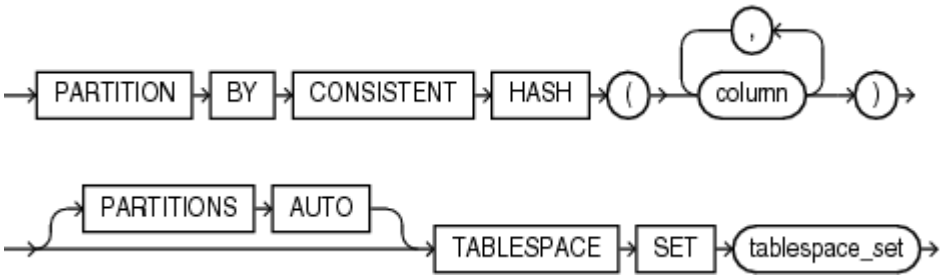
(table_partition_description::=)

system_partitioning::=

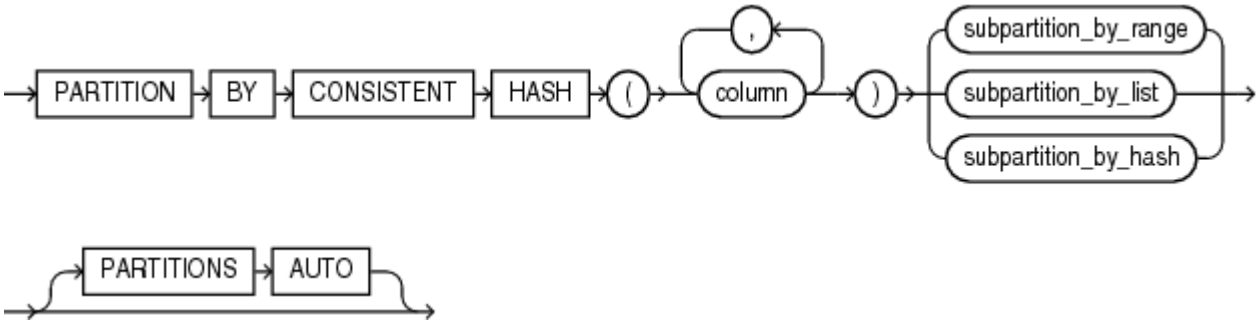


(reference_partition_desc::=)

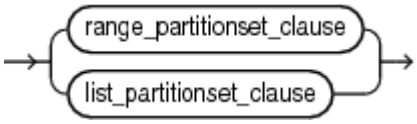
consistent_hash_partitions::=



consistent_hash_with_subpartitions::=

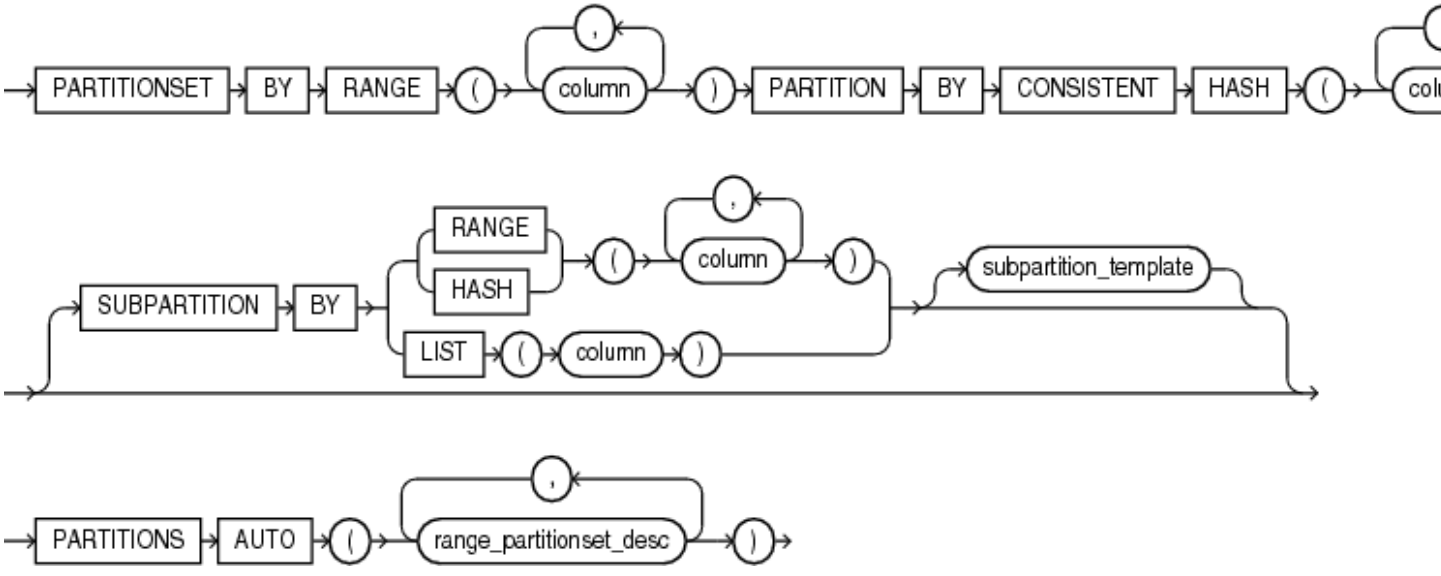


partitionset_clauses::=

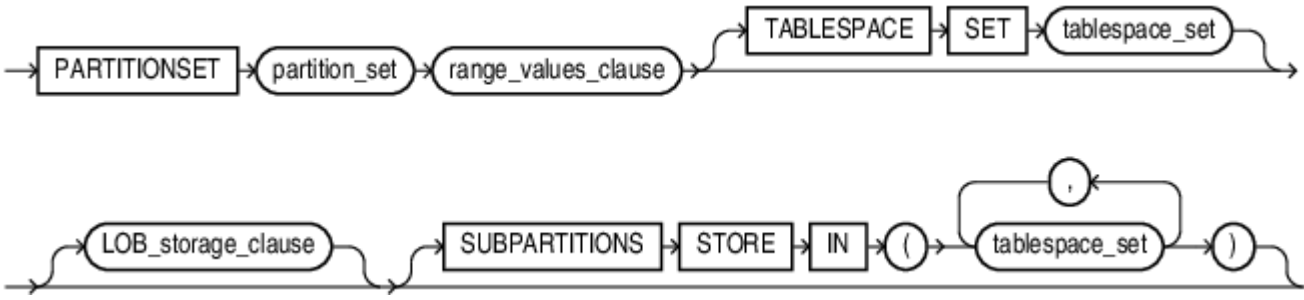


(range_partitionset_clause::=、list_partitionset_clause::=)

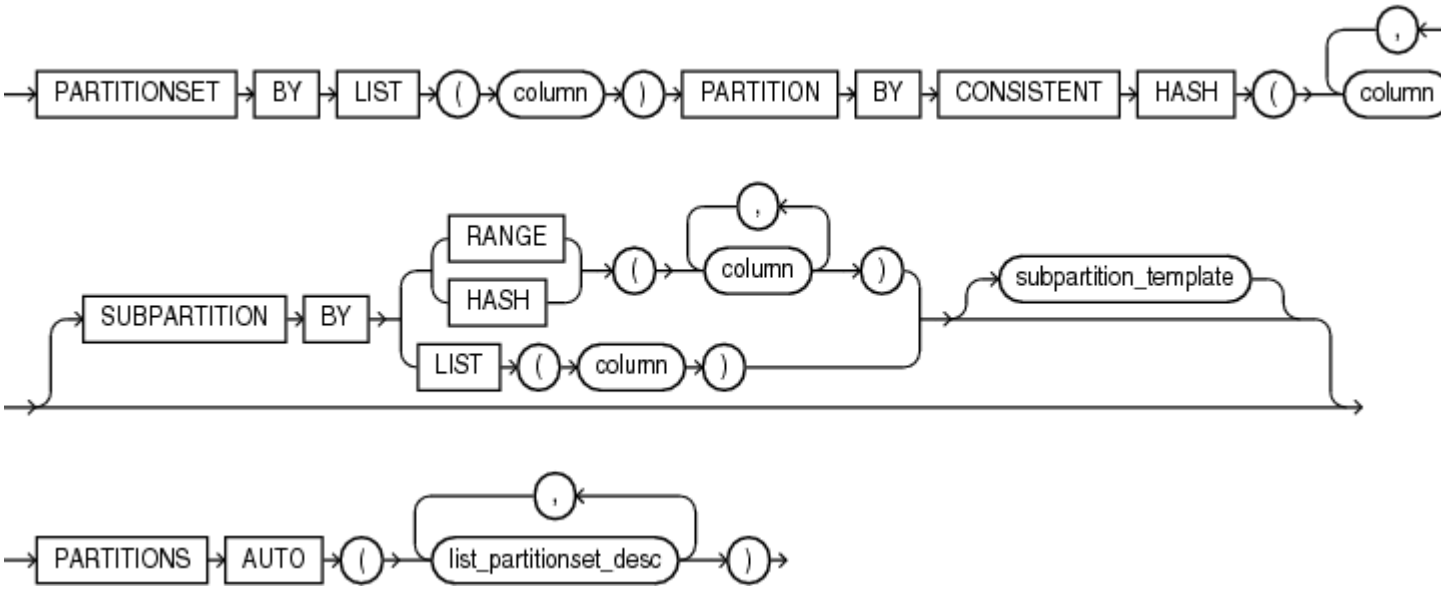
range_partitionset_clause::=



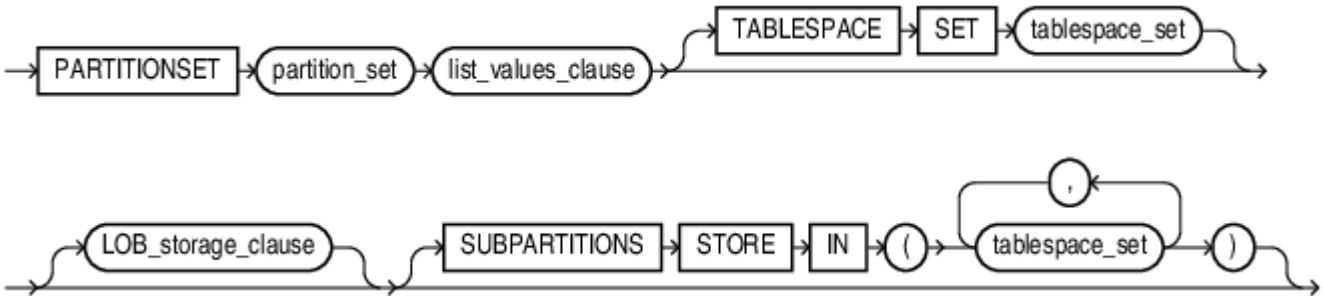
range_partitionset_desc::=



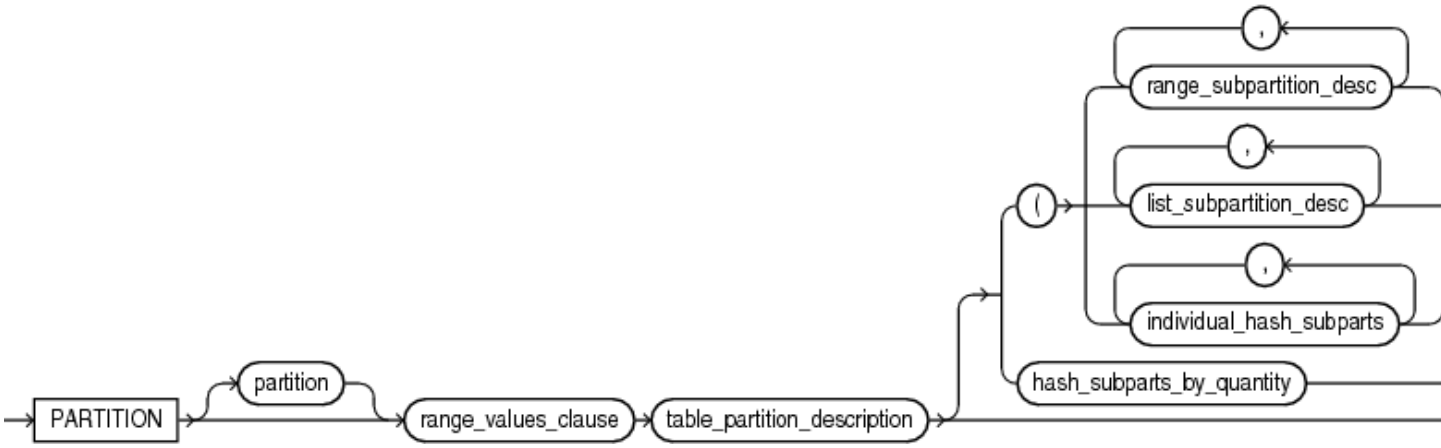
list_partitionset_clause::=



list_partitionset_desc::=

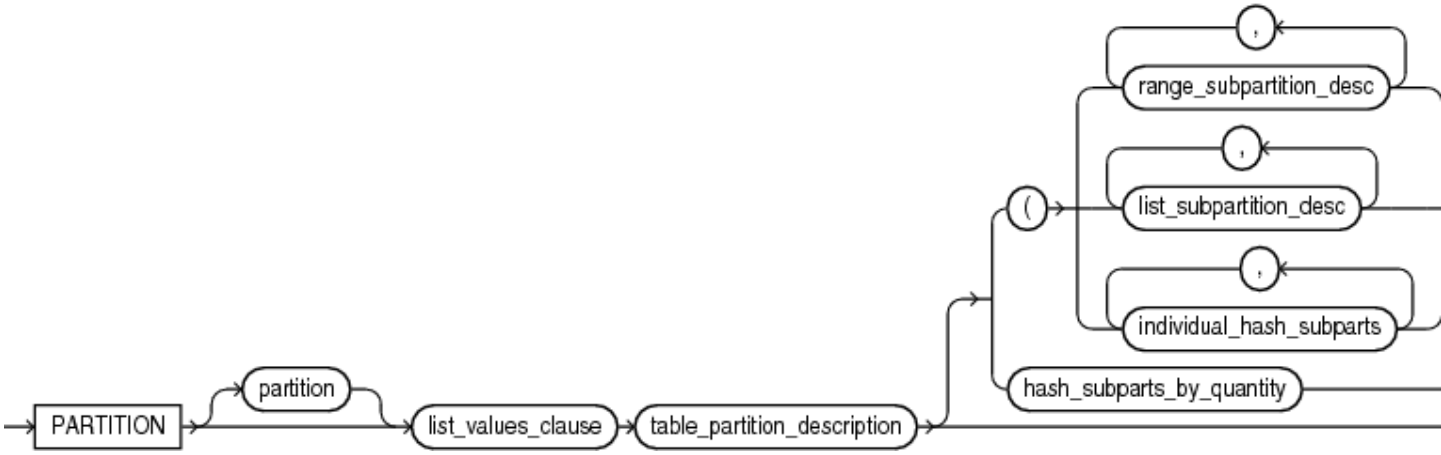


range_partition_desc::=



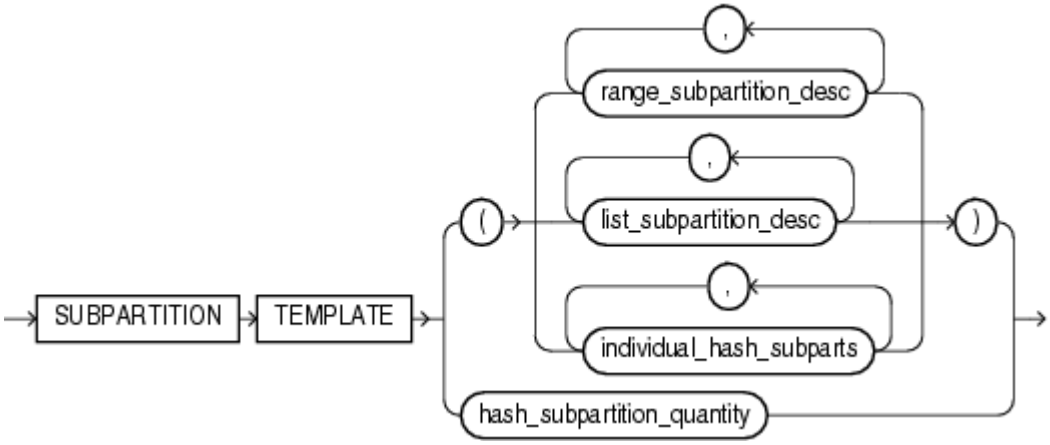
(range_values_clause::=、table_partition_description::=、range_subpartition_desc::=、list_subpartition_desc::=、individual_hash_subparts::=、hash_subparts_by_quantity::=)

list_partition_desc::=



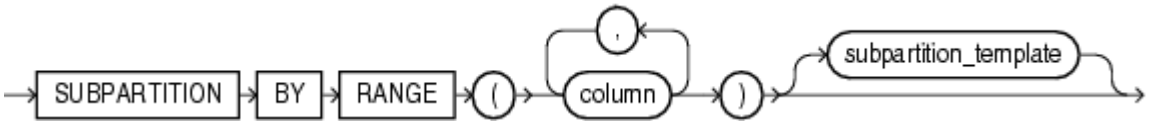
(list_values_clause::=、table_partition_description::=、range_subpartition_desc::=、list_subpartition_desc::=、individual_hash_subparts::=、hash_subparts_by_quantity::=)

subpartition_template::=



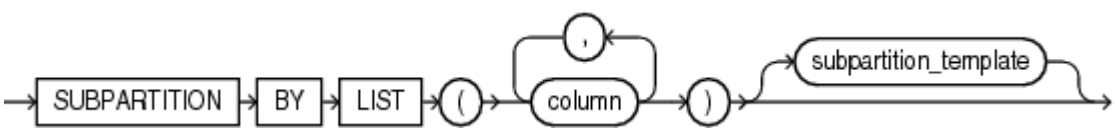
(range_subpartition_desc::=、list_subpartition_desc::=、individual_hash_subparts::=)

subpartition_by_range::=



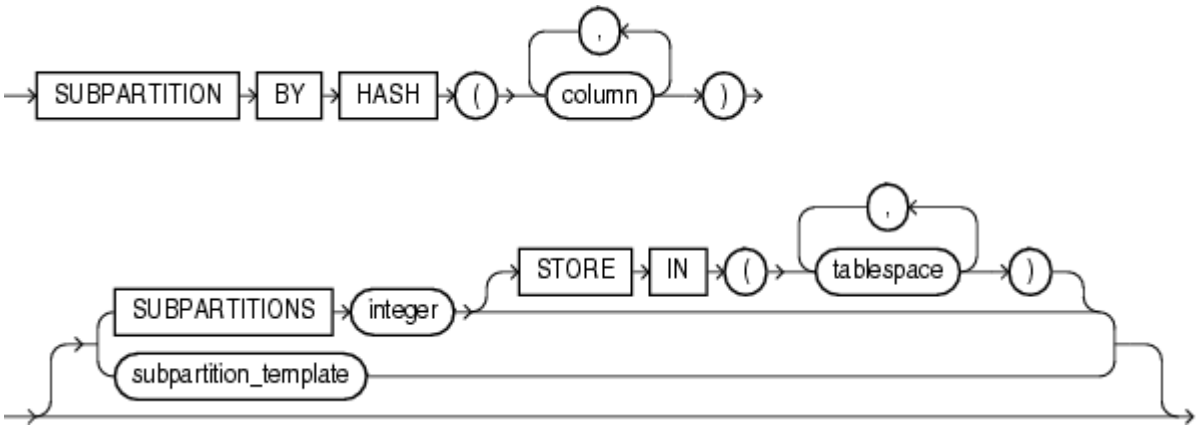
(subpartition_template::=)

subpartition_by_list::=



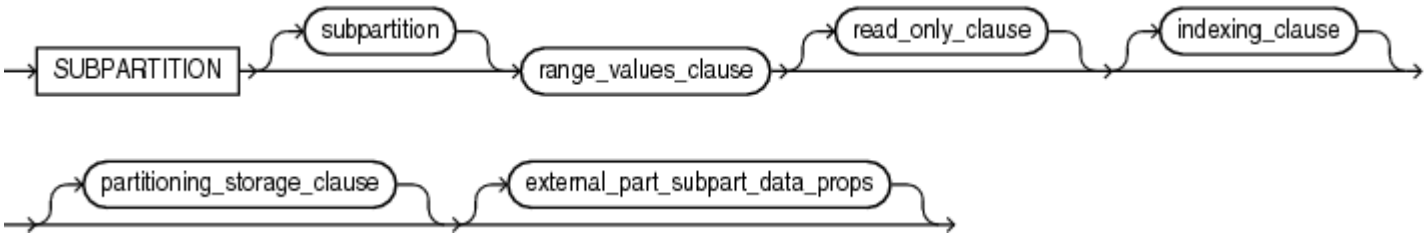
(subpartition_template::=)

subpartition_by_hash::=



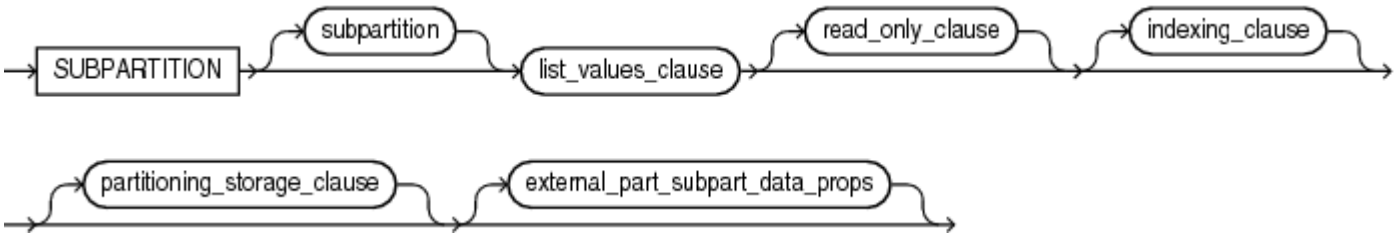
(subpartition_template::=)

range_subpartition_desc::=



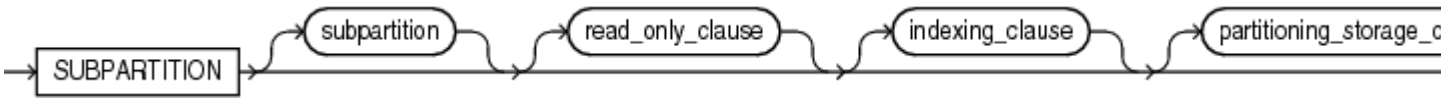
(range_values_clause::=、read_only_clause::=、indexing_clause::=、
partitioning_storage_clause::=、external_part_subpart_data_props::=)

list_subpartition_desc::=



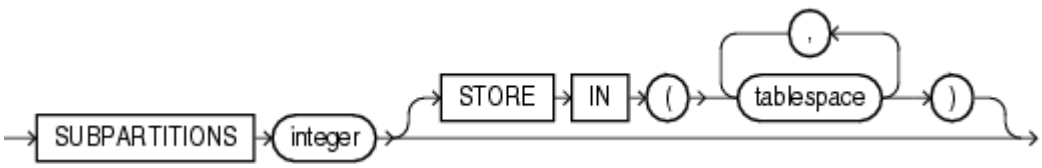
(list_values_clause::=、read_only_clause::=、indexing_clause::=、partitioning_storage_clause::=、
external_part_subpart_data_props::=)

individual_hash_subparts::=

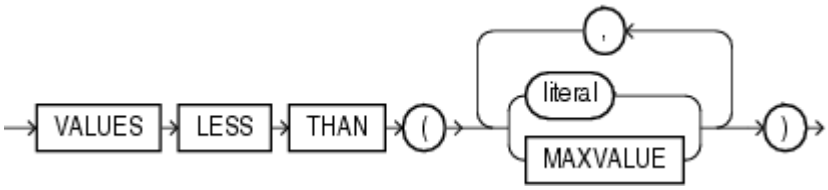


(read_only_clause::=、indexing_clause::=、partitioning_storage_clause::=)

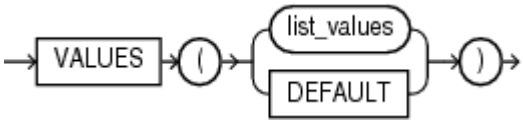
hash_subparts_by_quantity::=



range_values_clause::=

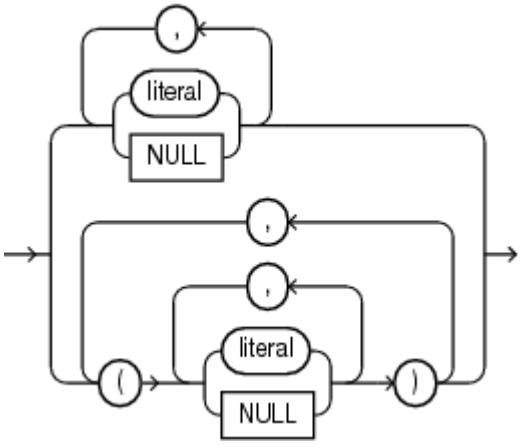


list_values_clause::=

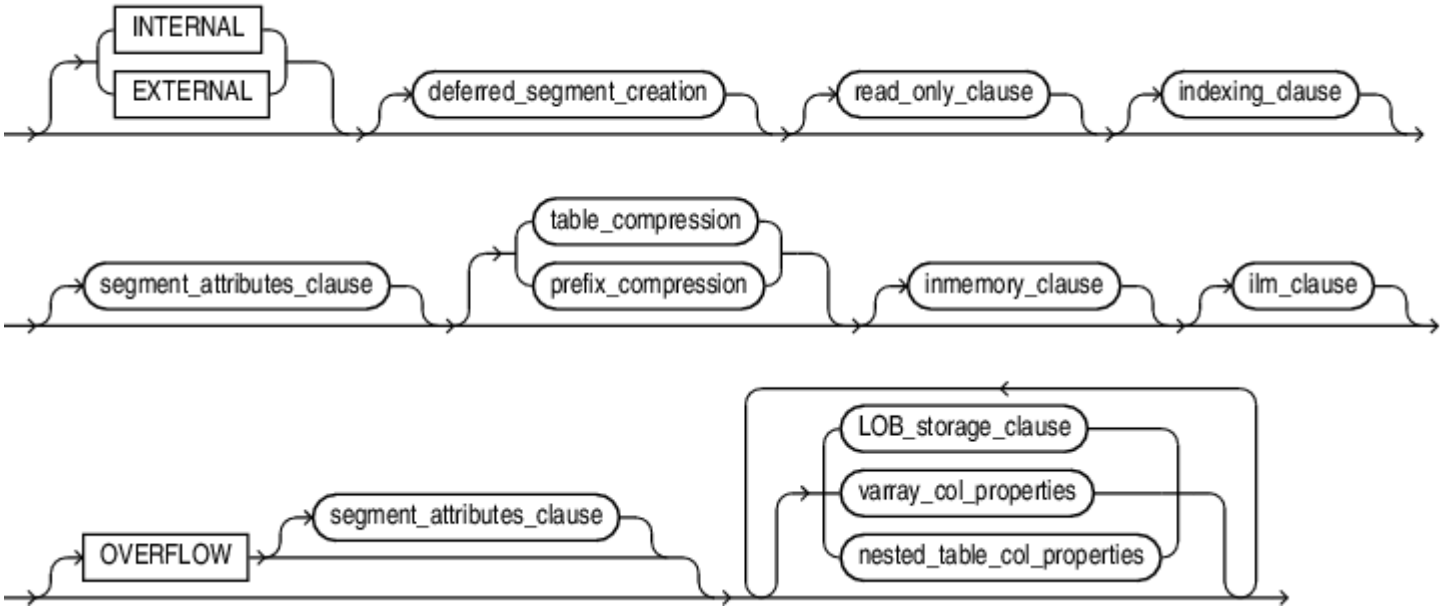


(list_values::=)

list_values::=



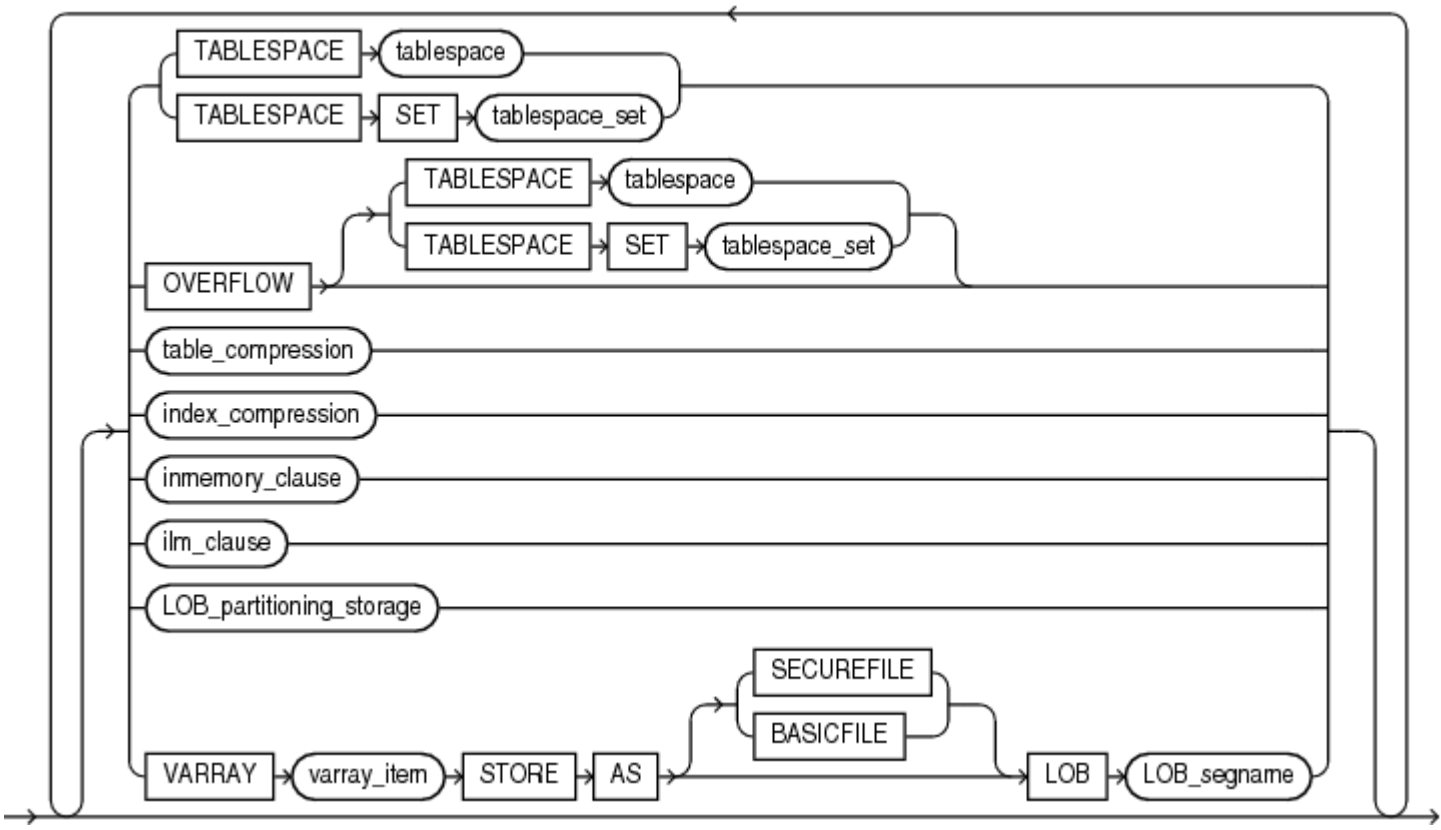
table_partition_description::=



(deferred_segment_creation::=、read_only_clause::=、indexing_clause::=、
segment_attributes_clause::=、table_compression::=、prefix_compression::=、

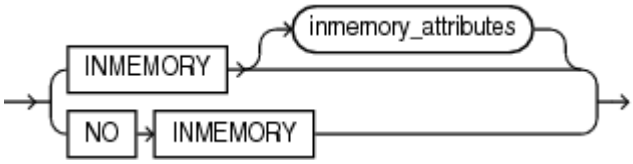
inmemory_clause::=、segment_attributes_clause::=、LOB_storage_clause::=、
varray_col_properties::=、nested_table_col_properties::=)

partitioning_storage_clause::=



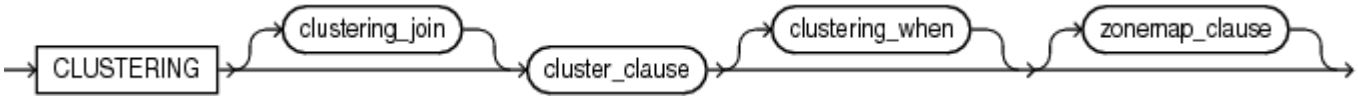
(table_compression::=、index_compression::=、inmemory_clause::=、
LOB_partitioning_storage::=)

inmemory_clause::=



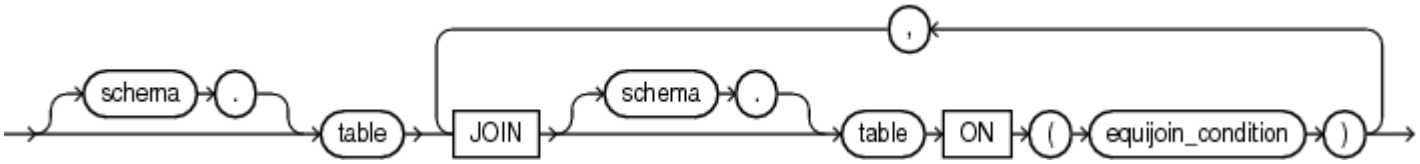
(inmemory_memcompress::=、inmemory_attributes::=)

attribute_clustering_clause::=

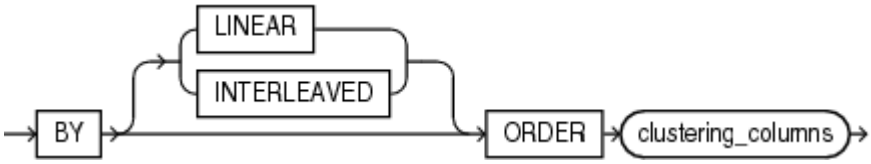


(clustering_join::=、cluster_clause::=、clustering_when::=、zonemap_clause::=)

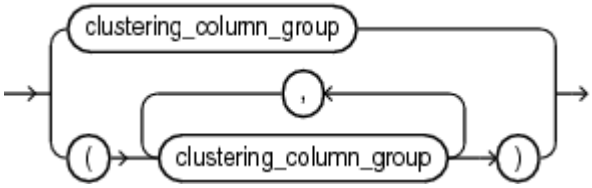
clustering_join::=



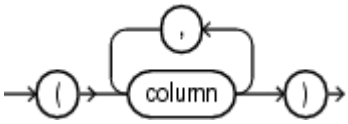
cluster_clause::=



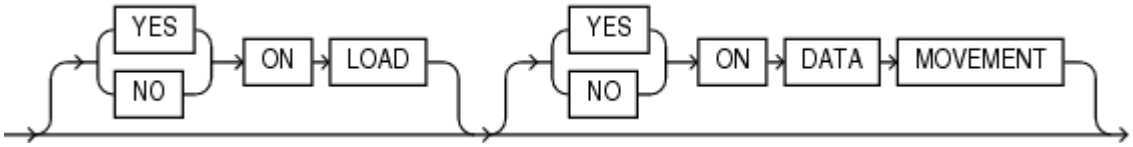
clustering_columns::=



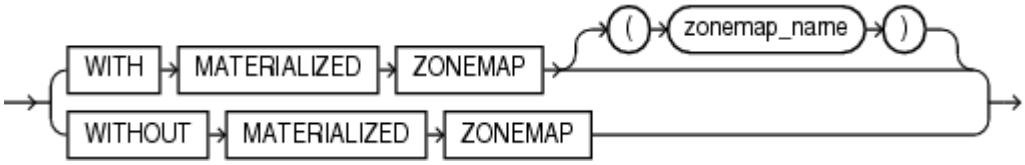
clustering_column_group::=



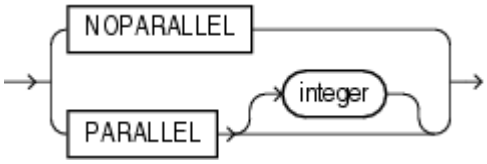
clustering_when::=



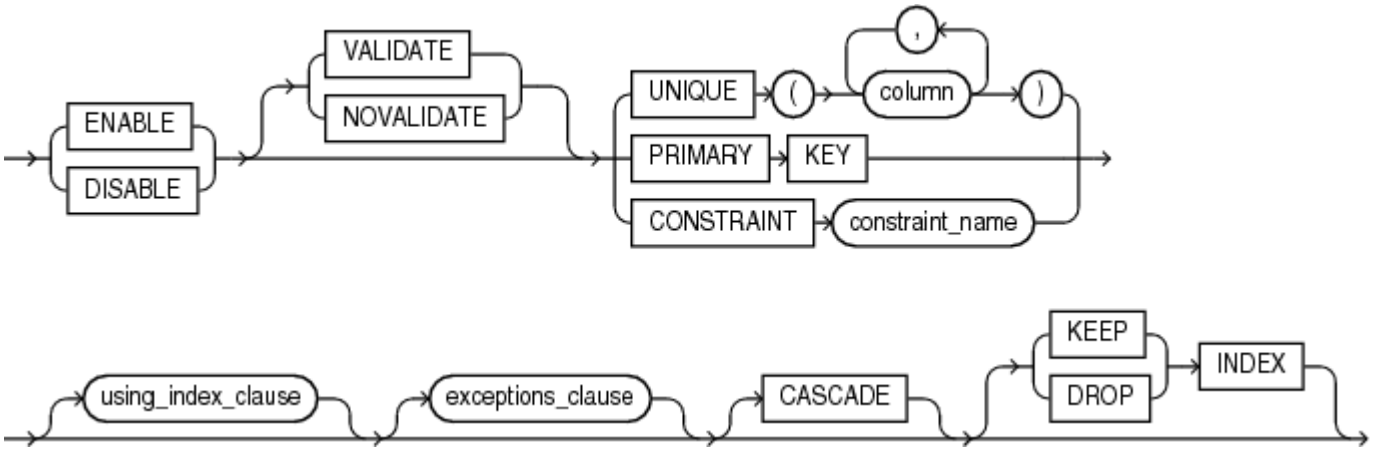
zonemap_clause::=



parallel_clause::=

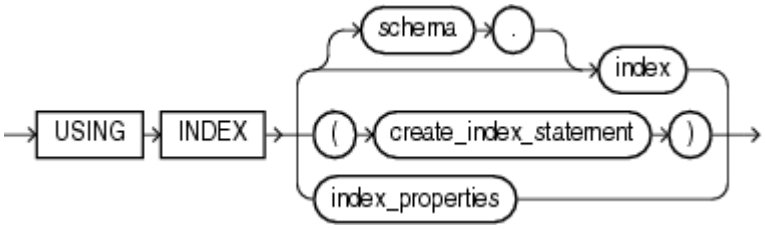


enable_disable_clause::=



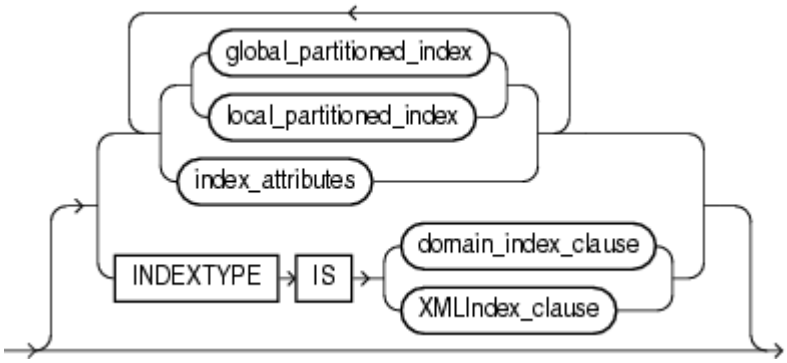
(using_index_clause::=。exceptions_clauseはCREATE TABLE文ではサポートされていません)

using_index_clause::=



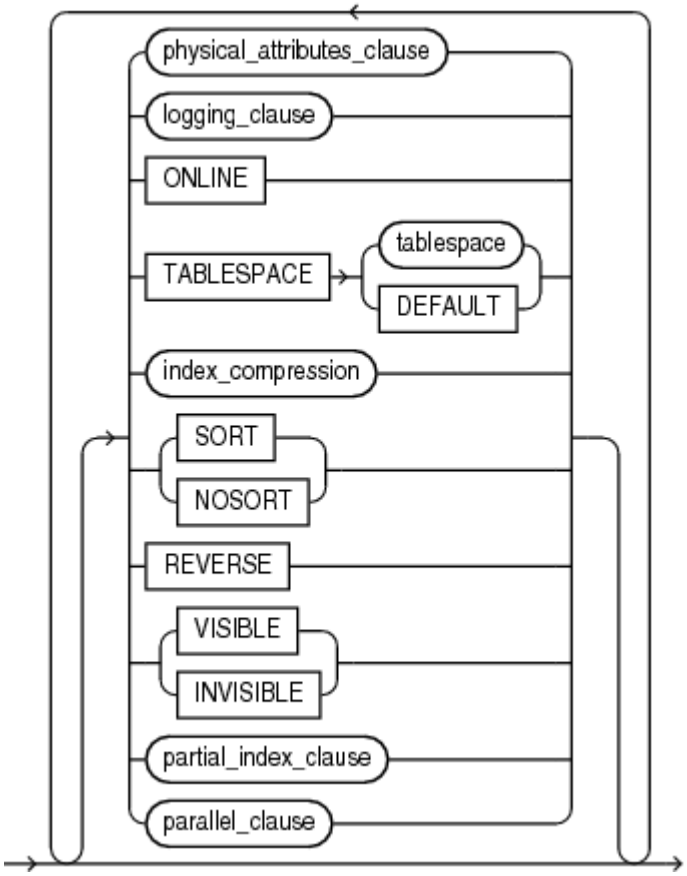
(create_index::=、index_properties::=)

index_properties::=



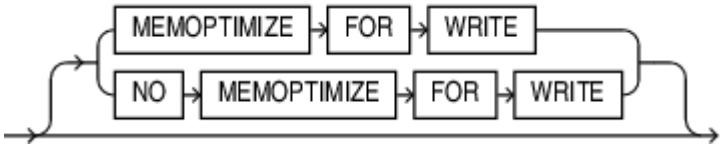
(global_partitioned_index::=、local_partitioned_index::= (CREATE INDEXの一部)、index_attributes::=。domain_index_clauseおよびXMLIndex_clauseはusing_index_clauseではサポートされていません)

index_attributes::=



(physical_attributes_clause::=、logging_clause::=、index_compression::=、partial_index_clauseおよびparallel_clause: (using_index_clauseではサポートされません))

memoptimize_write_clause



セマンティクス

GLOBAL TEMPORARY

適切な権限を持つすべてのセッションからその定義が参照可能な一時表を作成するには、GLOBAL TEMPORARYを指定します。一時表のデータは、データを表に挿入するセッションでのみ参照できます。

初めて一時表を作成した時点では、その表のメタデータはデータ・ディクショナリに格納されますが、表データの領域は割り当てられません。表セグメントの領域は、その表に初めてDML操作を実行したときに割り当てられます。一時表の定義は、標準的な表の定義と同じように維持されますが、表に含まれる表セグメントとデータは、セッション固有またはトランザクション固有のデータのいずれかになります。表のセグメントとデータをセッション固有にするか、トランザクション固有にするかは、ON COMMIT句で指定します。

セッションがバインドされていない場合にのみ、一時表でDDL操作(ALTER TABLE、DROP TABLE、CREATE INDEXなど)を実行できます。セッションを一時表にバインドするには、その表でINSERT操作を実行します。セッションを一時表からアンバインドするには、TRUNCATE文を発行するか、セッションを終了します。また、トランザクション固有の一時表からアンバインドするには、COMMIT文またはROLLBACK文を発行します。

PRIVATE TEMPORARY

プライベート一時表を作成するにはPRIVATE TEMPORARYを指定します。

プライベート一時表は、その表を作成したセッション内でのみ定義とデータが参照可能である点で、一時表とは異なります。プライベート一時表の有効範囲(トランザクションまたはセッション)を定義するには、ON COMMIT句を使用します。ON COMMIT句にキーワードDROP DEFINITIONを使用すると、作成されるトランザクション固有の表では、そのデータおよび定義がトランザクションのコミット時に削除されます。これはデフォルトの動作です。ON COMMIT句にキーワードPRESERVE DEFINITIONを使用すると、作成されるセッション固有の表では、トランザクションのコミット時にその定義が維持されます。この句の使用方法の詳細は、ON COMMITを参照してください。

プライベート一時表でサポートされている3つのDDL文は、CREATE、DROPおよびTRUNCATEです。

制限事項

プライベート一時表を作成するには、SYS以外のユーザーである必要があります。

関連項目:

一時表の詳細は、『Oracle Database概要』および「表の作成: 一時表の例」を参照してください。

一時表の制限事項

一時表には、次の制限事項があります。

- 一時表は、パーティション化、索引構成化またはクラスタ化できません。
- 一時表には、外部キー制約を指定できません。
- 一時表は、ネストした表の列を含むことはできません。
- LOB_storage_clauseのTABLESPACE、storage_clauseまたはlogging_clauseは指定できません。
- 一時表にパラレルUPDATE、DELETEおよびMERGEはサポートされていません。
- 一時表に対して指定できるsegment_attributes_clauseは、TABLESPACEのみで、単一の一時表領域を指定できます。
- 一時表では、分散トランザクションはサポートされません。
- 一時表には、INVISIBLEの列を格納できません。

プライベート一時表の制限事項

一時表の一般的な制限事項に加えて、プライベート一時表には次の制限事項があります。

- プライベート一時表の名前には、init.oraのパラメータPRIVATE_TEMP_TABLE_PREFIXで定義されている任意の接頭辞を常に付ける必要があります。デフォルトはORA$PTT_です。
- プライベート一時表では索引、マテリアライズド・ビューおよびゾーンマップを作成できません。
- デフォルト値を使用して列を定義することはできません。
- 永続オブジェクト(例: ビューまたはトリガー)ではプライベート一時表を参照できません。
- データベース・リンクを介してプライベート一時表を参照することはできません。

SHARDED

SHARDEDを指定して、シャード表を作成します。

この句は、Oracle Shardingを使用している場合にのみ有効で、これは、独立したデータベース間でデータが水平にパーティション化されるデータ階層アーキテクチャです。このような構成の各データベースはシャードと呼ばれます。すべてのシャードが一体的に単一の論理データベースを形成し、これはシャード・データベース(SDB)と呼ばれます。水平パーティション化では、同じ列を持つが行の異なるサブセットを持つ表が各シャードに含まれるように、表をシャード間に分割します。このように分割された表はシャード表と呼ばれます。

シャード表を作成する場合は、表を作成する表領域セットを指定する必要があります。シャード表のデフォルトの表領域セットはありません。詳細は、「CREATE TABLESPACE SET」を参照してください。

Oracle Shardingは、Oracle Partitioning機能に基づいています。そのため、シャード表は、パーティション表またはコンポジット・パーティション表である必要があります。シャード表を作成する場合は、`table_partitioning_clauses`のいずれかを指定する必要があります。これらの句のセマンティクスの詳細は、「table_partitioning_clauses」を参照してください。

シャード表の制限事項

シャード表には、次の制限事項があります。

- システム管理シャーディングでは、`CREATE SHARDED TABLE`文に`PARTITION BY REFERENCE`または`PARENT`句が含まれておらず、すでにルート表が存在するときに、`ORA-02530`をスローせずに複数のルート表(および表ファミリ)を作成できます。
- システム・シャーディングを使用する表ファミリは複数作成できますが、ユーザー定義および複合シャーディングでは1つの表ファミリのみがサポートされます。
- シャード表は、一時表または索引構成表にすることができません。
- シャード表には、ネストした表の列またはアイデンティティ列を含めることはできません。
- ユーザー定義のシャード表には、表領域セットではなくシャード表領域が必要であるため、TABLESPACE句を使用してシャード・システムまたはコンポジット・シャード表の表領域を指定することはできません。
- ユーザー定義のシャーディング環境では表領域セットは作成できません。
- シャード表にはシャード表領域が必要です。通常の表領域はサポートされていません。
- シャード表の複数のパーティションに対して同一の表領域は指定できません。このルールはサブパーティションにも適用されます。シャード表の異なるパーティションに属するサブパーティションに対して同一の表領域を指定することはできません。
- 非参照パーティション・シャード表のパーティションごとに表領域を指定する必要があります。
- ユーザー定義のシャーディングでは、パーティション・メソッドを範囲またはリストにする必要があります。自動リスト・パーティションと時間隔パーティションはサポートされていません。
- リスト・パーティション・メソッドでは1つのパーティション列のみを含めることができます。
- リスト・パーティション表ではデフォルト・パーティションはサポートされていません。
- リスト・パーティション表ではNULLのパーティションはサポートされていません。
- シャード表に定義される主キー制約には、シャーディング列を含める必要があります。重複表の列を参照するシャード表の列の外部キー制約はサポートされていません。
- シャード表では、システム・パーティション化および時間隔-レンジ・パーティション化はサポートされていません。

- シャード表の仮想列は、PARTITION BY句またはPARTITIONSET BY句に指定できません。

関連項目:

- Oracle Shardingの使用
- 『Oracle Database管理者ガイド』

DUPLICATED

この句は、Oracle Shardingを使用している場合にのみ有効です。DUPLICATEDを指定して重複表を作成します。これは、すべてのシャードに複製されます。これは非パーティション表またはパーティション表にすることができます。

重複表は表ファミリに関連付けられていません。

重複表の制限事項

重複表には、次の制限事項があります。

- 重複表にLONG列を含めることはできません。
- 重複表の主キー以外の列の最大数は999です。
- 重複表のXMLType列は、自動セグメント領域管理(ASSM)表領域でのみ使用できます。
- 重複表は、一時表にすることができません。
- 重複表は、参照パーティション表またはシステム・パーティション表にすることができません。
- 重複表には、NOLOGGINGまたはPARALLELを指定できません。
- 重複表は、インメモリー列ストアに対して有効にできません。

IMMUTABLE

IMMUTABLEキーワードを指定して、内部関係者による不正変更からデータを保護する読取り専用表を作成します。

CREATE TABLEにキーワードIMMUTABLE BLOCKCHAINを使用して、同様に不変であるブロックチェーン表を作成できます。

CREATE IMMUTABLE TABLE文を使用して不変表を作成するときに、必須のimmutable_table_clausesを指定する必要があります。

不変表では、VPDポリシー、分散トランザクションおよびXAトランザクションがサポートされます。

前提条件

- COMPATIBLE初期化パラメータは19.11.0.0以上に設定されている必要があります。
- 自分のスキーマ内に不変表を作成するには、CREATE TABLEシステム権限が必要です。他のユーザーのスキーマ内に不変表を作成するには、CREATE ANY TABLEシステム権限が必要です。
- NO DROPおよびNO DELETE句は必須です。

関連項目:

表の管理

BLOCKCHAIN

ブロックチェーン表を作成するには、BLOCKCHAINキーワードを指定します。

CREATE BLOCKCHAIN TABLE文を使用してブロックチェーン表を作成するときに、必須のblockchain_table_clausesを指定する必要があります。

ブロックチェーン表を作成すると、SYSが所有するディクショナリ表blockchain_table$にエントリが作成されます。

RESTRICTIONS

次のCREATE TABLE句は、ブロックチェーン表の作成では許可されていません。

- ORGANIZATION INDEX
- ORGANIZATION EXTERNAL
- NESTED TABLE

関連項目:

ブロックチェーン表の管理

schema

表を含めるスキーマを指定します。schemaを省略した場合、自分のスキーマ内に表が作成されます。

table

作成する表またはオブジェクト表の名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

関連項目:

シャード表の詳細は、『Oracle Database管理者ガイド』を参照してください。

SHARING

この句は、アプリケーション・ルートに表を作成する場合にのみ適用されます。このタイプの表はアプリケーション共通オブジェクトと呼ばれ、そのデータはアプリケーション・ルートに属するアプリケーションPDBと共有できます。表データの共有方法を決定するには、次の共有属性のいずれかを指定します。

- METADATA - メタデータ・リンクは表のメタデータを共有しますが、そのデータは各コンテナに固有です。このタイプの表は、メタデータリンク・アプリケーション共通オブジェクトと呼ばれます。
- DATA - データ・リンクは表を共有し、そのデータはアプリケーション・コンテナ内のすべてのコンテナに対して同じです。そのデータはアプリケーション・ルートにのみ格納されます。このタイプの表は、データリンク・アプリケーション共通オブジェクトと呼ばれます。
- EXTENDED DATA - 拡張データ・リンクは表を共有し、アプリケーション・ルートのそのデータは、アプリケーション・コンテナ内のすべてのコンテナに対して同じです。ただし、アプリケーション・コンテナの各アプリケーションPDBには、アプリケーションPDBに一意のデータを格納できます。このタイプの表の場合、データはアプリケーション・ルートに格納され、オプションで各アプリケーションPDBに格納されます。このタイプの表は、拡張データリンク・アプリケーション共通オブジェクトと呼ばれます。
- NONE - 表は共有されません。

この句を省略すると、データベースでは、DEFAULT_SHARING初期化パラメータの値を使用して、表の共有属性を決定します。DEFAULT_SHARING初期化パラメータに値が含まれていない場合、デフォルトはMETADATAです。

リレーショナル表を作成する場合は、METADATA、DATA、EXTENDED DATAまたはNONEを指定できます。

オブジェクト表またはXMLTYPE表を作成する場合は、METADATAまたはNONEのみ指定できます。

表の共有属性は、作成後には変更できません。

関連項目:

- DEFAULT_SHARING初期化パラメータの詳細は、Oracle Databaseリファレンスを参照してください
- アプリケーション共通オブジェクトの作成の詳細は、『Oracle Database管理者ガイド』を参照してください。

relational_table

この句を使用すると、リレーショナル表を作成できます。

relational_properties

リレーショナル表のコンポーネントを指定します。

column_definition

column_definitionでは、列の特性を定義できます。

AS subqueryを使用したcolumn_definitionの指定

AS subquery句を指定し、subqueryによって戻される各列に列名があるか、指定した列別名を持つ式である場合、column_definition句を省略できます。この場合、tableの列名はsubqueryによって返される列の名前と同じになります。索引構成表の作成は例外です。この場合、主キー列を指定する必要があるため、column_definition句を指定する必要があります。表型に関係なく、column_definition句とAS subquery句を指定する場合は、column_definition句からdatatypeを省略する必要があります。

column

表の列の名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

AS subqueryを指定する場合、索引構成表を作成しないかぎり、columnおよびdatatypeを省略できます。索引構成表の作成時にAS subqueryを指定する場合は、columnを指定し、datatypeを省略する必要があります。

表の列の絶対最大数は1000です。オブジェクト表、またはオブジェクトの列、ネストした表、VARRAYまたはREF型のリレーショナル表を作成する場合、制限の1000列までをカウントする有効な非表示列を作成して、ユーザー定義型の列をリレーショナル列にマップします。ユーザー定義型の属性を格納するリレーショナル列は、属性の照合プロパティを継承します。Oracle Database 12cリリース2 (12.2)では、ユーザー定義型は疑似照合プロパティUSING_NLS_COMPを使用して作成され、対応するリレーショナル列はこのプロパティを継承します。

datatype

列のデータ型を指定します。

一般に、datatypeを指定する必要があります。ただし、次の例外があります。

- AS subquery句を指定する場合は、datatypeを省略する必要があります。

- 参照整合性制約の外部キーの一部として、文で列が指定されている場合も、datatypeを省略できます。参照整合性制約の参照キーに対応する列のデータ型が、その列に自動的に割り当てられます。

表の列のデータ型の制限事項

- LONG列を持つ表は作成しないでください。かわりに、LOB列(CLOB、NCLOBまたはBLOB)を使用してください。LONG列は、下位互換性のためにサポートされています。
- ROWID型の列を指定することはできますが、それらの列の値が有効な行IDであることは保証されません。

関連項目:

LONG列およびOracleが提供するデータ型の詳細は、「データ型」を参照してください。

データ型の作成時または変更時に、ユーザー定義のデータ型を永続不可として指定できます。永続不可型のインスタンスは、ディスク上で保持することはできません。永続不可型として宣言されるユーザー定義のデータ型の詳細は、「CREATE TYPE」を参照してください。

COLLATE

COLLATE句を使用すると、列のデータ・バインドされた照合を指定できます。

column_collation_nameには、有効な名前付き照合または疑似照合を指定します。データ型CLOBまたはNCLOBの列の場合、column_collation_nameの許容値のみが疑似照合USING_NLS_COMPです。

この句を省略すると、列が次に割り当てられます。

- 疑似照合USING_NLS_COMP (列にデータ型CLOBまたはNCLOBがある場合)
- 対応する親キー列の照合(列が外部キーに属する場合)
- 列の作成時に示される表のデフォルトの照合

表のデフォルトの照合の詳細は、「DEFAULT COLLATION」句を参照してください。

COLLATE句を指定できるのは、COMPATIBLE初期化パラメータが12.2以上に設定され、かつMAX_STRING_SIZE初期化パラメータがEXTENDEDに設定されている場合のみです。

SORT

SORTキーワードは、この表をハッシュ・クラスタの一部として作成する場合、およびクラスタ列でもある列にのみ有効です。

表の行はSORTのないクラスタ・キー列のバケットにハッシュされ、この句を使用して、列の各バケット内でソートされます。これによって、クラスタ化データでの後続の操作時に、応答時間が短縮される場合があります。

関連項目:

- クラスタ表の作成の詳細は、「CLUSTER句」を参照してください。
- ハッシュ・クラスタの管理

VISIBLE | INVISIBLE

この句を使用すると、columnをVISIBLEにするか、INVISIBLEにするかを指定できます。デフォルトは、VISIBLEです。

INVISIBLEの列は、ユーザー指定の非表示列です。INVISIBLEの列を表示する場合や、この列に値を割り当てる場合

は、この列の名前を明示的に指定する必要があります。たとえば:

- SELECT *構文では、INVISIBLEの列が表示されません。ただし、SELECT文の選択リストにINVISIBLEの列を含めると、その列は表示されるようになります。
- INVISIBLEの列の値は、INSERT文のVALUES句で、暗黙的に指定することはできません。INVISIBLEの列は、列リストで指定する必要があります。
- Oracle Call Interface (OCI)の説明で、INVISIBLEの列を明示的に指定する必要があります。
- SQL*Plusを構成することで、INVISIBLEの列情報をDESCRIBEコマンドで表示されるようにできます。詳細は、『SQL*Plusユーザーズ・ガイドおよびリファレンス』を参照してください。

VISIBLE列およびINVISIBLE列のノート

VISIBLE列およびINVISIBLE列には、次のノートがあります。

- INVISIBLEの列は、CREATE TABLEの一部として指定すると、パーティション化キーとして使用できます。
- INVISIBLEの列は、column_expression内に指定できます。
- 仮想列は、INVISIBLEの列にすることができます。
- PL/SQL %ROWTYPE属性では、INVISIBLE列は表示されません。
- データ・ディクショナリ・ビューのALL_、DBA_、およびUSER_TAB_COLUMNSのCOLUMN_ID列により、SELECT *問合せで返される表、ビューまたはマテリアライズド・ビューの列の順序が決まります。COLUMN_IDの値は、INVISIBLEの列についてはNULLになります。非表示の列を表示されるようにすると、この列には、最高値の次に利用可能なCOLUMN_ID値が割り当てられます。表示可能な列を非表示の列にすると、その列のCOLUMN_ID値はNULLに設定され、その列より大きい値のCOLUMN_IDを持つ列のCOLUMN_IDは、それぞれ1ずつ減らされます。

VISIBLE列およびINVISIBLE列の制限事項

VISIBLE列およびINVISIBLE列には、次の制限事項があります。

- INVISIBLEの列は、外部表、クラスタ表、または一時表ではサポートされません。
- システム生成の非表示列を参照可能にすることはできません。

> ノート:
>
> ある列がシステム生成の非表示列であるかどうかを判断するには、データ・ディクショナリ・ビューの ALL_、DBA_、および USER_TAB_COLS の HIDDEN_COLUMN 列と USER_GENERATED 列を問い合せます。詳細は、『Oracle Database リファレンス』を参照してください。

DEFAULT

DEFAULT句を指定すると、後続のINSERT文が列の値を省略した場合に列に割り当てられる値を指定できます。この式のデータ型は、列に指定したデータ型と一致している必要があります。列には、この式を保持できる大きさが必要です。

DEFAULT式には、リテラル引数、列の参照またはネストしたファンクションの起動を戻さない、任意のSQLファンクションを含めることができます。

このDEFAULT式には、順序疑似列のCURRVALとNEXTVALを含めることができます。ただし、順序が存在していて、その順序

にアクセスするために必要な権限を所持している必要があります。それ以降に、DEFAULT式を使用して挿入を実行するユーザーには、表に対するINSERT権限と、順序に対するSELECT権限が必要になります。順序が後に削除されると、それ以降のDEFAULT式が使用されるINSERT文はエラーになります。順序の所有者を指定して完全修飾(たとえば、SCOTT.SEQ1)していない場合、Oracle Databaseは、CREATE TABLE文を発行したユーザーを順序のデフォルトの所有者にします。たとえば、ユーザーMARYがSCOTT.TABLEを作成して、SEQ2のように完全修飾していない順序を参照すると、その列は順序MARY.SEQ2を使用するようになります。順序に対するシノニムは完全に名前解決され、完全修飾された順序としてデータ・ディクショナリに格納されます。これは、パブリック・シノニムとプライベート・シノニムに当てはまります。たとえば、ユーザーBETHが、パブリック・シノニムまたはプライベート・シノニムのSYN1を参照する列を追加したときに、そのシノニムがPETER.SEQ7を参照していると、その列はデフォルトとしてPETER.SEQ7を格納するようになります。

デフォルト列値の制限事項

デフォルト列値には、次の制限事項があります。

- DEFAULT式に、PL/SQLファンクション、他の列、疑似列LEVEL、PRIORおよびROWNUMへの参照または完全に指定されていない日付定数は指定できません。
- 式には、スカラー副問合せ式を除くすべての書式を使用できます。

関連項目:

exprの構文は、「SQL式」を参照してください。

ON NULL

ON NULL句を指定すると、Oracle Databaseは、それ以降のINSERT文でNULLに評価される値を割り当てようとするときに、DEFAULTの列値を割り当てるようになります。

ON NULLを指定すると、NOT NULL制約と、NOT DEFERRABLE制約状態が暗黙的に指定されます。NOT NULLおよびNOT DEFERRABLEと競合する表内制約を指定すると、エラーが発生します。

ON NULL句の制限事項

この句は、オブジェクト型列またはREF列には指定できません。

関連項目:

DEFAULT ON NULL列値を含む表の作成: 例

identity_clause

この句を使用すると、ID列を作成できます。このID列には、それ以降のINSERT文ごとに、順序ジェネレータからの増加または減少する整数値が割り当てられます。identity_options句を使用すると、順序ジェネレータを構成できます。

ALWAYS

ALWAYSを指定すると、Oracle Databaseは、この列に値を割り当てるために、常に順序ジェネレータを使用するようになります。INSERTまたはUPDATEを使用して、列に明示的に値を割り当てようとすると、エラーが返されます。これはデフォルトです。

BY DEFAULT

BY DEFAULTを指定すると、Oracle Databaseはデフォルトで順序ジェネレータを使用して列に値を割り当てますが、指定した値を列に明示的に割り当てることもできるようになります。ON NULLを指定すると、それ以降のINSERT文でNULLに評価さ

れる値を割り当てようとしたときに、Oracle Databaseは順序ジェネレータを使用して列に値を割り当てます。

identity_options

identity_options句を使用すると、順序ジェネレータを構成できます。identity_options句のパラメータは、CREATE SEQUENCE文と同じです。これらのパラメータと特性の詳細は、「CREATE SEQUENCE」を参照してください。identity_optionsに固有のSTART WITH LIMIT VALUEは例外であり、ALTER TABLE MODIFYでのみ使用できます。詳細は、「identity_options」を参照してください。

ノート:

ID 列を作成するときには、パフォーマンスを向上するために、デフォルトの 20 よりも大きな値を使用して CACHE 句を指定してください。

ID列の制限事項

ID列には、次の制限事項があります。

- ID列は、表ごとに1列のみ指定できます。
- identity_clauseを指定する場合は、column_definition句のdatatypeに、数値データ型を指定する必要があります。ユーザー定義のデータ型は指定できません。
- identity_clauseを指定すると、column_definitionにDEFAULT句を指定できなくなります。
- identity_clauseを指定すると、NOT NULL制約と、NOT DEFERRABLE制約状態が暗黙的に指定されます。NOT NULLおよびNOT DEFERRABLEと競合する表内制約を指定すると、エラーが発生します。
- ID列が暗号化されていると、その暗号化アルゴリズムが推測される可能性があります。ID列には、強い暗号化アルゴリズムを使用してください。
- CREATE TABLE AS SELECTを使用すると、列に対するIDのプロパティが継承されなくなります。

関連項目:

ID列を含む表の作成例:

encryption_spec

ENCRYPT句を指定すると、透過的データ暗号化(TDE)機能を利用して、定義する列を暗号化できます。暗号化できる列の型は、CHAR、NCHAR、VARCHAR2、NVARCHAR2、NUMBER、DATE、LOBおよびRAWです。列を暗号化するユーザーなど、認可されたユーザーには、データは暗号化された形で表示されません。

ノート:

列を暗号化するには、適切な権限を持つシステム管理者が、セキュリティ・モジュールを初期化し、キーストアをオープンし、暗号化キーを設定しておく必要があります。列の暗号化の一般的な情報については、『Oracle Database Advanced Security ガイド』を参照してください。関連する ALTER SYSTEM 文については、「security_clauses」を参照してください。

USING 'encrypt_algorithm'

この句では、使用するアルゴリズムの名前を指定できます。有効なアルゴリズムは、AES256、AES192、AES128および3DES168です。COMPATIBLE初期化パラメータが12.2以上に設定されている場合は、アルゴリズムARIA128、ARIA192、ARIA256、GOST256およびSEED128も有効です。この句を省略すると、AES192が使用されます。同じ表内の複数の列を暗号化するときに、ある1つの列に対してUSING句を指定した場合は、暗号化する他のすべての列についても同じアルゴリズムを指定する必要があります。

IDENTIFIED BY password

この句を指定すると、指定したパスワードから列のキーが導出されます。

'integrity_algorithm'

この句では、使用する整合性アルゴリズムを指定できます。有効な整合性アルゴリズムは、SHA-1およびNOMACです。

- SHA-1を指定すると、TDEによってSecure Hash Algorithm(SHA-1)が使用され、整合性チェック用として、暗号化された各値に20バイトのMessage Authentication Code(MAC)が追加されます。これはデフォルトです。
- NOMACを指定すると、TDEによってMACは追加されず、整合性チェックも実行されません。この場合、暗号化された値ごとに20バイトのディスク領域が節約されます。NOMACを使用してディスク領域を節約し、パフォーマンスを向上させる方法の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

表のすべての暗号化列で同じ整合性アルゴリズムが使用される必要があります。表の列ですでにSHA-1アルゴリズムを使用している場合は、NOMACパラメータによって同じ表の別の列を暗号化することはできません。表のすべての暗号化列で使用されている整合性アルゴリズムを変更する方法を学習するには、ALTER TABLEのREKEY encryption_spec句に関する説明を参照してください。

SALT | NO SALT

SALTを指定すると、列のクリア・テキストを暗号化する前に、「salt」と呼ばれるランダムな文字列をクリア・テキストに追加するようデータベースに指示します。これはデフォルトです。

NO SALTを指定すると、列のクリア・テキストを暗号化する前に、データベースはクリア・テキストにsaltを追加しません。

暗号化する列に対してSALTまたはNO SALTを指定する場合、次の点を考慮してください。

- 列を索引キーとして使用する場合は、NO SALTを指定する必要があります。このような場面で使用されるSALTの詳細は、『Oracle Database Advanced Securityガイド』を参照してください。
- 表に対して表圧縮を指定した場合、SALTで暗号化される列内のデータは圧縮されません。

LOB暗号化に対しては、SALTまたはNO SALTを指定できません。

encryption_specの制限事項

列暗号化には、次の制限事項があります。

- 透過的データ暗号化は、従来のインポート/エクスポート・ユーティリティまたはトランスポータブル表領域ベースのエクスポートによってサポートされていません。かわりに、暗号化された列には、データ・ポンプ・インポート/エクスポート・ユーティリティを使用してください。
- 外部表の列を暗号化する場合、その表のアクセス・タイプとしてORACLE_DATAPUMPが使用されている必要があります。
- SYSが所有する表の列は暗号化できません。

- 外部キーの列は、暗号化できません。

関連項目:

透過的データ暗号化の詳細は、『Oracle Database Advanced Securityガイド』を参照してください。

virtual_column_definition

`virtual_column_definition`句によって、仮想列を作成できます。仮想列はディスクには格納されません。仮想列の値は、一連の式またはファンクションを計算することによって必要に応じて導出されます。仮想列は、問合せ、DMLおよびDDL文で使用できます。索引付けが可能であり、統計を収集できます。したがって、他の列と同様に処理できます。例外と制限は、「仮想列のノート:」および「仮想列の制限事項:」で説明します。

column

`column`には、仮想列の名前を指定します。

datatype

オプションで、仮想列のデータ型を指定できます。`datatype`を省略すると、列のデータ型は基礎となる式のデータ型に基づいて決定されます。すべてのOracleスカラー・データ型および`XMLType`がサポートされています。

COLLATE

`COLLATE`句を使用すると、仮想列のデータ・バインドされた照合を指定できます。`column_collation_name`には、有効な名前付き照合または疑似照合を指定します。この句を省略すると、対応する親キーの列から照合が継承される場合に列が外部キーに属していない場合は、列の作成時に示される表のデフォルトの照合に列が割り当てられます。表のデフォルトの照合の詳細は、「DEFAULT COLLATION」句を参照してください。

`COLLATE`句を指定できるのは、`COMPATIBLE`初期化パラメータが`12.2`以上に設定され、かつ`MAX_STRING_SIZE`初期化パラメータが`EXTENDED`に設定されている場合のみです。

VISIBLE | INVISIBLE

この句を使用すると、仮想列を`VISIBLE`にするか、`INVISIBLE`にするかを指定できます。デフォルトは、`VISIBLE`です。詳細は、「VISIBLE | INVISIBLE」を参照してください。

GENERATED ALWAYS

オプション・キーワード`GENERATED ALWAYS`は、セマンティクスを明確にするためのものです。列はディスクには格納されませんが、必要に応じて評価されることが示されます。

column_expression

`AS column_expression`句によって、列の内容が決まります。`column_expression`の詳細は、「列式」を参照してください。

VIRTUAL

オプション・キーワード`VIRTUAL`は、セマンティクスを明確にするためのものです。

evaluation_edition_clause

`column_expression`がエディショニングされたPL/SQLファンクションを参照する場合は、この句を指定する必要があります。この句を使用すると、エディショニングされたPL/SQLファンクションの名前解決時に検索されたエディション(評価エディション)を指定できます。

- CURRENT EDITIONを指定すると、このDDL文が実行されるエディションを検索できます。
- EDITION editionを指定すると、editionを検索できます。
- NULL EDITIONを指定することは、evaluation_edition_clauseを省略することと同じです。

evaluation_edition_clauseを省略すると、エディショニングされたオブジェクトは名前解決時に認識されなくなるためエラーが発生します。評価エディションが削除されると、それ以降の仮想列への問合せでエラーが発生します。

データベースは、仮想列が参照するファンクションの依存関係を維持しません。そのため、仮想列が非エディション・ファンクションを参照しているときに、そのファンクションをエディション・ファンクションにすると、次の操作でエラーが発生することがあります。

- その仮想列への問合せ
- その仮想列を含む行の更新
- その仮想列にアクセスするトリガーの起動

関連項目:

仮想列に評価エディションを指定する方法の詳細は、『Oracle Database開発ガイド』を参照してください。

unusable_editions_clause

この句を使用すると、1つ以上のエディションで仮想列の式を問合せの評価に使用できないことを指定できます。残りの一連のエディションでは、オプティマイザが安全に仮想列の式を使用して問合せを評価できます。

たとえば、仮想列にファンクション索引を定義したとします。オプティマイザはそのファンクション索引を使用して、仮想列の式がWHERE句に含まれる問合せを評価できます。問合せが、仮想列を使用できる一連のエディションに含まれるエディションでコンパイルされた場合、オプティマイザは、索引を使用して問合せを評価することを検討します。問合せが、仮想列を使用できる一連のエディションに含まれないエディションでコンパイルされた場合、オプティマイザは、索引を使用することを検討しません。

関連項目:

ファンクション索引を使用した最適化の詳細は、『Oracle Database概要』を参照してください。

UNUSABLE BEFORE句

この句を使用すると、先行のエディションで仮想列の式を問合せの評価に使用できないことを指定できます。

- CURRENT EDITIONを指定すると、このDDL文が実行されるエディションより前のエディションで仮想列の式が使用禁止になります。
- EDITION editionを指定すると、指定したeditionより前のエディションで仮想列の式が使用禁止になります。

UNUSABLE BEGINNING WITH句

この句を使用すると、特定のエディション以降のエディションで仮想列の式を問合せの評価に使用できないことを指定できます。

- CURRENT EDITIONを指定すると、このDDL文が実行されるエディションと、それ以降のエディションで仮想列の式が使用禁止になります。
- EDITION editionを指定すると、指定したeditionとそれ以降のエディションで仮想列の式が使用禁止になります。
- NULL EDITIONを指定することは、UNUSABLE BEGINNING WITH句を省略することと同じです。

この句に指定したエディションがその後削除されても、仮想列への影響はありません。

仮想列のノート:

- column_expressionで列レベルのセキュリティが実装された列を参照する場合、仮想列は基本列のセキュリティ・ルールを継承しません。この場合は、仮想列に対して列レベルのセキュリティ・ポリシーを複製するか、またはデータを暗黙的にマスクするファンクションを適用して、仮想列のデータを保護する必要があります。たとえば、一般的にクレジット・カード番号は列レベルのセキュリティ・ポリシーで保護しますが、コール・センターの従業員に対しては確認目的でクレジット・カード番号の下4桁を参照できるようにします。このような場合、クレジット・カード番号の下4桁のサブストリングを取るように仮想列を定義できます。
- 仮想列に定義された表の索引は、表のファンクション索引と同じです。
- 仮想列を直接更新することはできません。したがって、UPDATE文のSET句に仮想列を指定することはできません。ただし、UPDATE文のWHERE句には仮想列を指定できます。同様に、DELETE文のWHERE句に仮想列を指定して、仮想列の導出値に基づいて表から行を削除できます。
- 仮想列を含む表をFROM句に指定する問合せは、結果キャッシュに適応します。結果キャッシュの詳細は、「RESULT_CACHEヒント」を参照してください。
- 作成時にファンクションにDETERMINISTICが明示的に指定されている場合、column_expressionは、PL/SQLファンクションを参照できます。ただし、後でファンクションが置き換えられた場合、仮想列に依存する定義は無効にされません。そのような場合、表にデータが含まれていると、仮想列が制約、索引またはマテリアライズド・ビューの定義あるいは結果キャッシュで使用された場合に、仮想列を参照する問合せで不適切な結果が戻される場合があります。そのため、仮想列の決定的なPL/SQLファンクションを置き換えるために、次の手順を実行します。
  - 仮想列の制約を無効にして再度有効にします。
  - 仮想列の索引を再作成します。
  - 仮想列にアクセスするマテリアライズド・ビューを完全にリフレッシュします。
  - キャッシュされた問合せが仮想列にアクセスした場合、結果キャッシュをフラッシュします。
  - 表の統計情報を再収集します。
- 仮想列は、INVISIBLEの列にすることができます。column_expressionには、INVISIBLEの列を含めることができます。

仮想列の制限事項:

- 仮想列は、リレーショナル・ヒープ表にのみ作成できます。仮想列は、索引構成表、外部表、オブジェクト表、クラスタ化表または一時表ではサポートされません。
- AS句のcolumn_expressionには、次の制限事項があります。
  - 別の仮想列を名前で参照できません。
  - column_expressionで参照される列は、同じ表で定義されている必要があります。
  - 決定的なユーザー定義ファンクションを参照できますが、その場合、仮想列をパーティション化キー列として使用できません。
  - column_expressionの出力は、スカラー値である必要があります。

関連項目:

column_expressionの詳細および制限事項は、「列式」を参照してください。

- 仮想列は、Oracleが提供するデータ型、ユーザー定義型、またはLOBまたはLONG RAWにすることはできません。
- パーティション列として使用する仮想列を定義する式には、PL/SQLファンクションへのコールを指定できません。

関連項目:

仮想列を持つ表の作成例は、「仮想表の列の追加: 例」および『Oracle Database管理者ガイド』を参照してください。

period_definition

period_definition句を使用すると、tableの有効期間ディメンションを作成できます。

この句によって、tableの時制有効性のサポートが実装されます。この句を指定する場合、table内の1つの列の開始時間列に開始日とタイムスタンプが含まれ、table内の別の列の終了時間列に終了日とタイムスタンプが含まれます。これら2つの列によって、tableの有効期間ディメンション、つまり各行が有効と見なされる期間が定義されます。Oracleフラッシュバック問合せを使用すると、指定した時間の前の特定の時点で、または特定の期間中に有効と見なされるかどうかに基づいて、tableから行を取り出すことができます。

表の作成時には、最大で1つの有効期間ディメンションを指定できます。その後、ALTER TABLEのadd_period_clauseを使用して、他の有効期間ディメンションを表に追加できます。

valid_time_column

有効期間ディメンションの名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。Oracle Databaseによって、この名前を持つNUMBERデータ型のINVISIBLE仮想列がtable内に作成されます。

start_time_columnおよびend_time_column

オプションで、次のようにこれらの句を指定できます。

- start_time_columnは、開始日またはタイムスタンプを含む開始時間列の名前を指定するために使用します。
- end_time_columnは、終了日またはタイムスタンプを含む終了時間列の名前を指定するために使用します。

start_time_columnとend_time_columnに指定する名前は、「データベース・オブジェクトのネーミング規則」に記載されている要件を満たしている必要があります。

これらの句を指定する場合、start_time_columnとend_time_columnをCREATE TABLEのcolumn_definition句内で定義する必要があります。各列は日時データ型(DATE、TIMESTAMP, TIMESTAMP WITH TIME ZONEまたはTIMESTAMP WITH LOCAL TIME ZONE)にする必要があり、VISIBLEまたはINVISIBLEにすることができます。

これらの句を指定しない場合、valid_time_column_STARTという開始時間列およびvalid_time_column_ENDという終了時間列が作成されます。これらの列のデータ型はTIMESTAMP WITH TIME ZONEであり、すべてINVISIBLEです。

開始時間列と終了時間列の値は、他の列と同じように挿入および更新できます。考慮事項は次のとおりです。

- 開始時間列の値がNULLの場合、終了時間列の値(その値を含まない)よりも早い時間を示すすべての時間値の行が有効と見なされます。

- 終了時間列の値がNULLの場合、開始時間列の値(その値を含む)よりも遅い時間を示すすべての時間値の行が有効と見なされます。
- どちらの列の値もNULLではない場合、開始時間列の値を終了時間列の値よりも早くする必要があります。開始時間列の値(その値を含む)よりも早い時間および終了時間列の値(その値を含まない)よりも遅い時間を示すすべての時間値の行が有効と見なされます。
- どちらの列の値もNULLの場合、すべての時間値の行が有効と見なされます。

有効期間ディメンション列の制限事項

有効期間ディメンション列には、次の制限事項があります。

- `valid_time_column`は、内部使用のみを目的としています。DDLまたはDML操作は実行できませんが、1つ例外があります。ALTER TABLEの`drop_period_clause`を使用すると、列を削除できます。
- 開始時間列と終了時間列を削除する唯一の方法は、ALTER TABLEの`drop_period_clause`を使用することです。
- 開始時間列と終了時間列がOracle Databaseによって自動的に作成された場合、それらは`INVISIBLE`になり、後から`VISIBLE`には変更できません。

関連項目:

- 時制有効性の詳細は、『Oracle Database開発ガイド』を参照してください。
- Oracleフラッシュバック問合せの詳細は、「SELECT」の「flashback_query_clause」を参照してください。
- 有効期間ディメンションを追加および削除する方法の詳細は、「ALTER TABLE」の「add_period_clause」と「drop_period_clause」を参照してください

制約句

制約句を使用すると、表の列に対する制約を作成できます。DEFERRABLE以外のPRIMARY KEY制約を索引構成表に指定してください。これらの制約の構文、詳細および使用例は、「constraint」を参照してください。

inline_ref_constraintおよびout_of_line_ref_constraint

これらの句を使用すると、REF型の列を指定できます。これらの句の唯一の違いは、表レベルで`out_of_line_ref_constraint`を指定することです。このため、定義するREF型の列または属性を識別する必要があります。`inline_ref_constraint`は、REF型の列または属性の定義の一部として指定してください。

関連項目:

「REF制約の例」

inline_constraint

`inline_constraint`を使用すると、整合性制約を列定義の一部として定義できます。

オブジェクト型の列のスカラー属性に、UNIQUE制約、PRIMARY KEY制約およびREFERENCES制約を作成できます。また、オブジェクト型の列のNOT NULL制約、オブジェクト型の列またはオブジェクト型の列の属性を参照するCHECK制約も作成できます。

out_of_line_constraint

out_of_line_constraint構文を使用すると、整合性制約を表定義の一部として定義できます。

supplemental_logging_props

supplemental_logging_props句を指定すると、追加のデータがログ・ストリームに入れられ、ログに基づくツール製品をサポートできます。

supplemental_log_grp_clause

この句を使用すると、名前付きのログ・グループを作成できます。

- NO LOG句を使用すると、REDOログから1つ以上の列を省略できます。この句を指定しない場合、これらの列は名前付きのログ・グループのREDOに含まれます。名前付きのログ・グループに、1つ以上の固定長列をNO LOGを使用せずに指定する必要があります。
- ALWAYSを指定すると、更新時にログ・グループのすべての列がREDOに含まれます。関連付けられた行が変更されるとログ・グループのすべての列に対してサプリメンタル・ロギングが行われるため、これは無条件ログ・グループといいます(「常時ログ・グループ」ともいいます)。ALWAYSを指定しない場合、ログ・グループの任意の列が変更された場合のみ、ログ・グループのすべての列に対してサプリメンタル・ロギングが行われます。これは、条件付きログ・グループといいます。

サプリメンタル・ロギングが指定されているかどうかを確認するには、適切なUSER_、ALL_またはDBA_LOG_GROUP_COLUMNSデータ・ディクショナリ・ビューを問い合せます。

supplemental_id_key_clause

この句を使用すると、主キー列、一意キー列および外部キー列のすべて、またはこれらの列の組合せを補足的に記録する必要があることを指定できます。Oracle Databaseは、無条件ログ・グループまたは条件付きログ・グループのいずれかを生成します。無条件ログ・グループでは、関連付けられた行が変更されると、ログ・グループのすべての列に対してサプリメンタル・ロギングが行われます。条件付きログ・グループでは、ログ・グループの任意の列が変更された場合のみ、ログ・グループのすべての列に対してサプリメンタル・ロギングが行われます。

- ALL COLUMNSを指定すると、この行の最大サイズが固定長のすべての列がREDOログに含まれます。このようなREDOログは、システム生成無条件ログ・グループといいます。
- PRIMARY KEY COLUMNSを指定すると、主キーを持つすべての表において、更新が実行されるたびに、主キーのすべての列がREDOログに置かれます。Oracle Databaseは、次のとおりサプリメンタル・ロギングを行う列を評価します。
  - まず、主キー制約が指定されている列が選択されます(制約が検証済か、または制約にRELYのマークが付いていて、DISABLEDおよびINITIALLY DEFERREDのマークが付いていない場合)。
  - 主キー列が存在しない場合、1つ以上のNOT NULL列を持つ最小のUNIQUE索引が検索され、この索引の列が使用されます。
  - このような索引が存在しない場合、表のすべてのスカラー列に対してサプリメンタル・ロギングが行われます。
- UNIQUE COLUMNSを指定すると、一意キーまたはビットマップ索引を持つすべての表において、一意キー列またはビットマップ索引列が変更された場合、一意キーまたはビットマップ索引に属するその他のすべての列もREDOログに置かれます。このようなログ・グループは、システム生成条件付きログ・グループといいます。
- FOREIGN KEY COLUMNSを指定すると、外部キーを持つすべての表において、外部キー列が変更された場合、外部キーに属するその他のすべての列もREDOログに置かれます。このようなREDOログは、システム生成条件付きログ・グループといいます。

この句を複数回指定すると、指定するたびに個別のログ・グループが作成されます。サプリメンタル・ロギング・データが指定されているかどうかを確認するには、適切なUSER_、ALL_またはDBA_LOG_GROUPSデータ・ディクショナリ・ビューを問い合せます。

immutable_table_clauses

この句は、不変表の作成時に指定する必要があります。

例: 不変表の作成

次の例は、ユーザー・スキーマにtrade_ledgerという名前の不変表を作成します。この不変表は、非アクティブな状態で40日経過した後にのみ削除できます。行は、挿入後100日経過するまで削除できません。

```
CREATE IMMUTABLE TABLE trade_ledger (tr_id NUMBER, user_name VARCHAR2(40), tr_value
NUMBER)
       NO DROP UNTIL 40 DAYS IDLE
       NO DELETE UNTIL 100 DAYS AFTER INSERT;
```

blockchain_table_clauses

ブロックチェーン表を作成するときは、blockchain_table_clausesを指定する必要があります。

- blockchain_drop_table_clause
- blockchain_row_retention_clause
- blockchain_hash_and_data_format_clause

blockchain_drop_table_clause

```
NO DROP [ UNTIL integer DAYS IDLE ]
```

integerを使用して、ブロックチェーン表がアイドル状態である(つまり、行が挿入されない)必要がある日数を指定します。最小のアイドル状態保持期間は0日ですが、推奨されるアイドル状態保持期間は16日です。

この句は、次の2つの方法で指定できます。

- NO DROPは、ブロックチェーン表を削除できないことを意味します。
- NO DROP UNTIL integer DAYS IDLEは、最新行の経過日数がinteger未満である場合、ブロックチェーン表は削除できないことを意味します。

blockchain_row_retention_clause

```
NO DELETE [ LOCKED ]
  | NO DELETE UNTIL integer DAYS AFTER INSERT [LOCKED]
```

- integerは、挿入された行を削除するまでのアイドル状態保持期間を指定します。最小のアイドル状態保持期間は0日ですが、推奨されるアイドル状態保持期間は16日です。
- LOCKEDを指定すると、ALTER TABLEを使用して保存期間を変更できません。
- UNTIL number DAYS AFTER INSERT句にLOCKEDを指定しない場合、ALTER TABLEを使用して保存期間を変更できますが、変更できるのは前の保存期間よりも大きい値のみです。
- NO DELETE LOCKEDを指定すると、この表から行を削除できません。しかし、表がblockchain_drop_table_clauseに指定された日数を超えて非アクティブな場合は、表全体を削除できます。

blockchain_hash_and_data_format_clause

```
HASHING USING sha2_512 VERSION v1
```

ブロックチェーン表を作成するときは、最後にblockchain_drop_table_clauseとblockchain_row_retention_clauseの後にこの句を指定する必要があります。

ALTER TABLE文を使用してブロックチェーン表を変更する場合、この句は指定できません。

DEFAULT COLLATION

この句を使用すると、表のデフォルトの照合を指定できます。デフォルトの照合は、文字データ型の表の列に割り当てられ、この文で作成されるか、ALTER TABLE文で表に後から追加されます。collation_nameには、有効な名前付き照合または疑似照合を指定します。

この句を省略した場合、表のデフォルトの照合は、表を含むスキーマの有効スキーマのデフォルトの照合に設定されます。有効なスキーマのデフォルトの照合の詳細は、「ALTER SESSION」のDEFAULT_COLLATION句を参照してください。

表のデフォルトの照合を上書きし、データ・バインドされた照合を特定の列に割り当てるには、CREATE TABLEまたはALTER TABLEのcolumn_definition句またはvirtual_column_definition句に、またはALTER TABLEのmodify_col_properties句またはmodify_virtcol_properties句にCOLLATE句を指定します。

DEFAULT COLLATION句を指定できるのは、COMPATIBLE初期化パラメータが12.2以上に設定され、かつMAX_STRING_SIZE初期化パラメータがEXTENDEDに設定されている場合のみです。

CLOB列およびNCLOB列の照合の制限事項

列にCLOBまたはNCLOBのデータ型がある場合は、その指定された照合をUSING_NLS_COMPにする必要があります。CLOB列およびNCLOB列の照合は常にUSING_NLS_COMPであり、表のデフォルトの照合の影響は受けません。

関連項目:

デフォルトの照合およびデータ・バインドされた照合の詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

ON COMMIT

ON COMMIT句は、グローバル一時表を作成する場合にのみ有効です。この句を使用すると、一時表のデータがトランザクションまたはセッションの存続期間中保持されるかどうかを指定できます。

DELETE ROWS

DELETE ROWSは、トランザクション固有の一時表に対して指定します。これはデフォルトです。各コミット後に表が切り捨てられます(すべての行が削除されます)。

PRESERVE ROWS

PRESERVE ROWSは、セッション固有の一時表に対して指定します。セッション終了時に表が切り捨てられます(すべての行が削除されます)。

プライベート一時表の有効範囲を定義するときにもON COMMIT句を使用しますが、トランザクション固有の表定義する場合はキーワードDROP DROP DEFINITIONを使用し、セッション固有の表を定義する場合はPRESERVE DEFINITIONを使用します。

DROP DEFINITION

トランザクションのコミット時にその内容と定義が削除されるプライベート一時表を作成するには、DROP DEFINITIONを指定します。このプライベート一時表の有効範囲はトランザクションに制限されています。これはデフォルトです。

PRESERVE DEFINITION

トランザクションのコミット時にその定義が維持されるプライベート一時表を作成するには、PRESERVE DEFINITIONを指定します。このプライベート一時表の有効範囲は、セッションに拡大されます。

physical_properties

物理プロパティは、エクステントとセグメントの処理、および表の記憶特性に関係します。

INTERNAL | EXTERNAL

内部パーティションを指定するにはINTERNALキーワードを使用します。これはデフォルトです。外部パーティションを指定するにはEXTERNALキーワードを使用します。

deferred_segment_creation

この句を使用すると、この表のセグメントを作成するタイミングを指定できます。

- SEGMENT CREATION DEFERRED: 表のセグメント、表のLOB列のセグメント、表作成の一部として暗黙的に作成される索引、およびその後で明示的に表に作成される索引の作成が遅延されます。これらは、最初のデータ行が表に挿入された後に作成されます。このとき、表のセグメント、LOB列と索引、および明示的に作成される索引がすべて実体化され、このCREATE TABLE文(明示的に作成される索引の場合は、CREATE INDEX文)に指定されたすべての記憶域プロパティが継承されます。これらのセグメントは、最初の挿入操作がコミットされていないか、ロールバックされたかに関係なく作成されます。これはデフォルト値です。

  注意:

  遅延セグメント作成を使用する表を多く作成する場合、データベースに十分な領域を割り当て、最初の行の挿入時にすべての新しいセグメントに対して十分な領域が確保されるようにしてください。

- SEGMENT CREATION IMMEDIATE: 表のセグメントは、このCREATE TABLE文の一部として作成されます。

即時セグメント作成は、たとえば、アプリケーションがDBA_、USER_およびALL_SEGMENTSデータ・ディクショナリ・ビューに表示されるオブジェクトに依存している場合に有効です。このようなオブジェクトは、セグメントが作成されるまでビューに表示されないためです。この句によって、DEFERRED_SEGMENT_CREATION初期化パラメータの設定が上書きされます。

既存の表、そのLOB列または索引にセグメントが作成されているかどうかを確認するには、USER_TABLES、USER_INDEXESまたはUSER_LOBSのSEGMENT_CREATED列を問い合せます。

セグメントのない表のノート

セグメントが実体化されていない表には、次のルールが適用されます。

- この表をCREATE TABLE ... AS副問合せを使用して作成する場合は、ソース表に行がなければ、新しい表のセグメントの作成は遅延されます。ソース表に行がある場合は、新しい表のセグメントは遅延なしで作成されます。
- セグメントが実体化されないうちにALTER TABLE ... ALLOCATE EXTENTを指定すると、セグメントが実体化され、エクステントが割り当てられます。ただし、表の索引に対するDDL文のALLOCATE EXTENT句からはエラーが戻されます。
- 表、そのLOB列または索引に対するDDL文の場合は、DEALLOCATE UNUSEDを指定しても無視されます。
- セグメントが作成されていない表または表パーティションの索引に対するONLINE操作は、特に警告もなく無効になり、OFFLINE操作として実行されます。

- 1つ以上のLOB列がある表で次のいずれかのDDL文を実行すると、結果のパーティションまたはサブパーティションが実体化されます。
  - ALTER TABLE SPLIT [SUB]PARTITION
  - ALTER TABLE MERGE [SUB]PARTITIONS
  - ALTER TABLE ADD [SUB]PARTITION (ハッシュ・パーティションのみ)
  - ALTER TABLE COALESCE [SUB]PARTITION (ハッシュ・パーティションのみ)

遅延セグメント作成の制限事項

この句には、次の制限事項があります。

- クラスタ化表、グローバル一時表、セッション固有の一時表、内部表、外部表、およびSYS、SYSTEM、PUBLIC、OUTLNまたはXDBが所有する表に対して、セグメント作成を遅延させることはできません。
- 遅延セグメント作成は、ディクショナリ管理表領域ではサポートされていません。
- 遅延セグメント作成は、SYSTEM表領域ではサポートされていません。
- シリアル化可能トランザクションは、遅延セグメント作成では動作しません。セグメントが作成されていない空の表にデータを挿入しようとすると、エラーになります。

関連項目:

セグメント割当ての概要は、『Oracle Database概要』を参照してください。DEFERRED_SEGMENT_CREATION初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

segment_attributes_clause

segment_attributes_clauseを指定すると、表の物理属性および表領域の記憶域を指定できます。

physical_attributes_clause

physical_attributes_clauseを使用すると、PCTFREE、PCTUSED、INITRANSパラメータの値、および表の記憶特性を指定できます。

- 非パーティション表の場合、指定した各パラメータおよび記憶特性は、表に関連付けられたセグメントの実際の物理属性となります。
- パーティション表の場合、パーティション作成文のPARTITION句で明示的に値を上書きしないかぎり、指定したパラメータおよび記憶特性の値は、CREATE文(および後続のALTER TABLE ... ADD PARTITION文)で指定するすべてのパーティションに関連付けられたセグメントのデフォルト物理属性になります。

この句を省略すると、PCTFREEは10、PCTUSEDは40、INITRANSは1に設定されます。

関連項目:

これらの句の詳細は、「physical_attributes_clause」および「storage_clause」を参照してください。

表の作成: 記憶域の例

TABLESPACE

Oracle Databaseが、表、オブジェクト表OIDINDEX、パーティション、LOBデータ・セグメント、LOB索引セグメントまたは索

引構成表のオーバーフロー・データ・セグメントを作成する表領域を指定します。TABLESPACEを省略した場合、その表を含むスキーマの所有者のデフォルトの表領域内に作成されます。

1つ以上のLOB列を持つヒープ構成表の場合、LOB記憶域に対するTABLESPACEを省略すると、表を作成する表領域にLOBデータおよび索引セグメントが作成されます。

1つ以上のLOB列を持つ索引構成表の場合、TABLESPACEを省略すると、索引構成表の主キー索引セグメントが作成された表領域に、LOBデータおよび索引セグメントが作成されます。

非パーティション表の場合、TABLESPACEに指定する値は、表に関連付けられたセグメントの実際の物理属性となります。パーティション表の場合、TABLESPACEに指定する値は、PARTITION記述でTABLESPACEを指定しないかぎり、このCREATE文(および後続のALTER TABLE ... ADD PARTITION文)で指定されたすべてのパーティションに関連付けられたセグメントのデフォルト物理属性となります。

関連項目:

表領域の詳細は、「CREATE TABLESPACE」を参照してください。

TABLESPACE SET

この句は、CREATE TABLEのSHARDEDキーワードを指定してシャード表を作成する場合にのみ有効です。この句を使用して、Oracle Databaseが表を作成する表領域セットを指定します。

CREATE SHARDED TABLE文を使用するとき、1つの表ファミリに関連付けられる表領域セットは1つだけです。複数の表ファミリを持つ表領域セットを使用しようとすると、エラーがスローされます。

logging_clause

表、および制約のために必要な索引、パーティションまたはLOBの記憶特性の作成をREDOログ・ファイルに記録する(LOGGING)かしないか(NOLOGGING)を指定します。表のロギング属性は、その索引の属性に依存しません。

表、パーティションまたはLOBの記憶域に対して、後で実行されるダイレクト・ローダー(SQL*Loader)操作およびダイレクト・パスINSERT操作のログをとる(LOGGING)かとらない(NOLOGGING)かも指定します。

この句の詳細は、「logging_clause」を参照してください。

table_compression

table_compression句は、ヒープ構成表に対してのみ有効です。この句を使用すると、ディスク使用量を削減するためにデータ・セグメントを圧縮するかどうかを指定できます。COMPRESS句を使用すると、表を圧縮できます。NOCOMPRESS句を指定すると、表の圧縮が無効になります。デフォルトはNOCOMPRESSです。

COMPRESS

キーワードCOMPRESSのみを指定することは、ROW STORE COMPRESS BASICを指定することと同じであり、基本表圧縮が使用可能になります。

ROW STORE COMPRESS BASIC

ROW STORE COMPRESSまたはROW STORE COMPRESS BASICを指定して表の圧縮を使用可能にすると、基本表圧縮が使用可能になります。Oracle Databaseでは、表の圧縮が効果的である場合に、ダイレクト・パス・インサート操作中にデータの圧縮を開始します。元のインポート・ユーティリティ(imp)はダイレクト・パス・インサートをサポートしないため、圧縮フォーマットでデータをインポートすることはできません。

基本表圧縮が指定された表では、physical_attributes_clauseでPCTFREEの値を明示的に設定しないかぎり、PCTFREE値0を使用して圧縮が最大化されます。

以前のリリースでは、基本表圧縮はCOMPRESS BASICを使用することで有効になっていました。この構文は、下位互換性のために引き続きサポートされています。

関連項目:

ダイレクト・パスINSERT操作の制限などの詳細は、「従来型INSERTおよびダイレクト・パスINSERT」を参照してください。

ROW STORE COMPRESS ADVANCED

ROW STORE COMPRESS ADVANCEDを指定して表の圧縮を有効にすると、高度な行圧縮が有効になります。Oracle Databaseは表でのすべてのDML操作中にデータの圧縮を開始します。OLTP環境にはこの圧縮形式をお薦めします。

ROW STORE COMPRESS ADVANCEDまたはNOCOMPRESSが指定された表では、PCTFREEのデフォルトを明示的に上書きしないかぎり、デフォルト値10を使用して、圧縮を最大限にしながらデータへの今後のDML変更も考慮します。

以前のリリースでは、OLTP表圧縮と呼ばれる高度な行圧縮は、COMPRESS FOR OLTPを使用することで有効になっていました。この構文は、下位互換性のために引き続きサポートされています。

COLUMN STORE COMPRESS FOR { QUERY | ARCHIVE }

COLUMN STORE COMPRESS FOR QUERYまたはCOLUMN STORE COMPRESS FOR ARCHIVEを指定すると、ハイブリッド列圧縮が有効になります。ハイブリッド列圧縮を使用すると、ダイレクト・パス・インサート、従来型の挿入および配列挿入中にデータを圧縮できます。ロード処理中にデータが列指向の書式に変換されてから、圧縮されます。指定したレベルに適した圧縮アルゴリズムがOracle Databaseで使用されます。一般的に、レベルが高くなるほど、圧縮率は大きくなります。ハイブリッド列圧縮を使用すると高い圧縮率が得られますが、CPUコストも高くなります。そのため、この圧縮形式は更新頻度が高くないデータに使用することをお薦めします。

COLUMN STORE COMPRESS FOR QUERYは、データ・ウェアハウス環境で有効です。有効な値はLOWおよびHIGHで、HIGHの方が高い圧縮率が得られます。デフォルトはHIGHです。

COLUMN STORE COMPRESS FOR ARCHIVEを使用すると、COLUMN STORE COMPRESS FOR QUERYより高い圧縮率が得られます。長期間格納するデータを圧縮する場合に有効です。有効な値はLOWおよびHIGHで、HIGHを指定すると実行可能な最も高い圧縮率が得られます。デフォルトはLOWです。

COLUMN STORE COMPRESSを指定することは、COLUMN STORE COMPRESS FOR QUERY HIGHを指定することと同じです。

COLUMN STORE COMPRESS FOR QUERYまたはCOLUMN STORE COMPRESS FOR ARCHIVEが指定された表では、physical_attributes_clauseでPCTFREEの値を明示的に設定しないかぎり、PCTFREE値に0を使用して圧縮を最大限にします。これらの表では、PCTFREEは、ダイレクト・パスINSERTを使用してロードしたブロックには影響しません。PCTFREEは、従来型INSERTを使用してロードしたブロック、およびダイレクト・パスINSERTを使用して最初にロードしたブロックに対してDML操作を行った結果作成されるブロックに適用されます。

[NO] ROW LEVEL LOCKING

ROW LEVEL LOCKINGを指定する場合、Oracle DatabaseはDML操作中に行レベル・ロックを使用します。これにより、ハイブリッド列圧縮したデータにアクセスする場合にこれらの操作のパフォーマンスが向上します。NO ROW LEVEL LOCKINGを使用する場合、行レベル・ロックは使用されません。デフォルトはNO ROW LEVEL LOCKINGです。

以前のリリースでは、ハイブリッド列圧縮は、COMPRESS FOR QUERYおよびCOMPRESS FOR ARCHIVEを使用することで有

効になっていました。この構文は、下位互換性のために引き続きサポートされています。

関連項目:

一部のOracleストレージ・システムに備わっているハイブリッド列圧縮機能の詳細は、『Oracle Database概要』を参照してください。

表の圧縮のノート

表の圧縮は、ヒープ構成表の次の部分に対して指定できます。

- 表全体の場合は、`relational_table`または`object_table`の`physical_properties`句で指定します。
- レンジ・パーティションの場合は、`range_partitions`句の`table_partition_description`で指定します。
- コンポジット・レンジ・パーティションの場合は、`range_partition_desc`句の`table_partition_description`で指定します。
- コンポジット・リスト・パーティションの場合は、`list_partition_desc`句の`table_partition_description`で指定します。
- リスト・パーティションの場合は、`list_partitions`句の`table_partition_description`で指定します。
- システム・パーティションまたは参照パーティションの場合は、`reference_partition_desc`句の`table_partition_description`で指定します。
- ネストした表の記憶表の場合は、`nested_table_col_properties`句で指定します。

関連項目:

`DBMS_COMPRESSION`パッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。これは、アプリケーションに適した圧縮レベルを選択するのに役立つパッケージです。例を含む表の圧縮の詳細は、『Oracle Database管理者ガイド』を参照してください。

表の圧縮の制限事項

表の圧縮には、次の制限事項があります。

- BasicFiles LOBのデータ・セグメントは圧縮されません。SecureFiles LOBの圧縮の詳細は、「LOB_compression_clause」を参照してください。
- `COMPRESS BASIC`を使用する表の列は削除できませんが、未使用に設定することはできます。`ROW STORE COMPRESS ADVANCED`、`COLUMN STORE COMPRESS FOR QUERY`および`COLUMN STORE COMPRESS FOR ARCHIVE`を使用する表では、`ALTER TABLE ... drop_column_clause`のすべての操作が有効です。
- 索引構成表、オーバーフロー・セグメント、オーバーフロー・セグメントのパーティションまたは索引構成表のマッピング表セグメントには、どのタイプの表の圧縮も指定できません。
- 外部表またはクラスタの一部である表には、どのタイプの表の圧縮も指定できません。
- `LONG`列または`LONG RAW`列を持つ表、SYSスキーマによって所有された`SYSTEM`表領域内の表、あるいは`ROWDEPENDENCIES`が使用可能な表に対しては、どのタイプの表の圧縮も指定できません。

- フラッシュバック・アーカイブに対応した表に対しては、ハイブリッド列圧縮は指定できません。
- オブジェクト表、XMLType表、抽象データ型を持つ列、表として格納されるコレクションまたはOPAQUE型(オブジェクトとして格納されるXMLType列を含む)のオブジェクト・リレーショナル機能に対しては、ハイブリッド列圧縮は指定できません。
- ハイブリッド列圧縮で圧縮された表の行を更新すると、行のROWIDが変更される可能性があります。
- ハイブリッド列圧縮で圧縮された表では、単一の行に対する更新によって複数の行がロックされる可能性があります。そのため、書込みトランザクションの同時実効性が影響を受ける場合があります。
- ハイブリッド列圧縮で圧縮された表に外部キー制約が定義されている場合、INSERTにAPPENDヒントを指定してデータを挿入すると、挿入するデータは通常のハイブリッド列圧縮より低いレベルで圧縮されます。ハイブリッド列圧縮でデータを圧縮するには、外部キー制約を使用禁止にし、INSERTにAPPENDヒントを指定してデータを挿入してから、外部キー制約を使用可能にします。

inmemory_table_clause

この句を使用して、インメモリー列ストア(IM列ストア)の表を有効化または無効化します。IM列ストアは任意に入力できる静的なSGAプールです。ここには、迅速にスキャンできるように最適化された特殊な列形式で、表のコピーおよびパーティションが格納されています。IM列ストアでは、バッファ・キャッシュは置換しませんが、両方のメモリー領域において同じデータを異なる形式で格納するための補足としての役割を果たします。

- INMEMORYを指定して、IM列ストアの表を有効化します。

  オプションでinmemory_attributes句を使用して、表データをIM列ストアに格納する方法を指定できます。この句を使用すると、データ圧縮方法およびデータ移入優先度を指定できます。Oracle RAC環境では、Oracle RACインスタンス間のデータの分散および複製方法も指定できます。詳細は、inmemory_attributes句を参照してください。
- NO INMEMORYを指定して、IM列ストアの表を無効化します。
- inmemory_column_clauseを指定して、IM列ストアの特定の表の列を有効化または無効化し、特定の列のデータ圧縮方法を指定します。詳細は、inmemory_column_clauseを参照してください。

この句を省略すると、作成される表領域のデフォルトのIM列ストア設定が表に割り当てられます。表領域のデフォルトのIM列ストア設定の指定の詳細は、CREATE TABLESPACEのinmemory_clauseを参照してください。

Oracle Active Data Guard環境では、プライマリ・データベースの表にこの句を指定した場合、表はOracle Active Data GuardインスタンスのIM列ストアに対して有効化または無効化されます。

ノート:

INMEMORY_CLAUSE_DEFAULT の初期化パラメータを使用すると、新しい表およびマテリアライズド・ビューに対してデフォルトの IM 列ストア句を指定できます。INMEMORY_CLAUSE_DEFAULT 初期化パラメータの詳細は、Oracle Database Reference を参照してください。

インメモリー列ストアの制限事項

インメモリー列ストアには、次の制限事項があります。

- INMEMORY句は、索引構成表には指定できません。

- SYSスキーマによって所有され、SYSTEMまたはSYSAUX表領域にある表のINMEMORY句は指定できません。
- Oracle Database 18c以降は、外部表に対してINMEMORY句を指定できます。DDLが適切に解析できるようにするには、QUERY_REWRITE_INTEGRITY初期化パラメータをstale_toleratedに設定する必要があります。INMEMORYが指定されている場合、ALTERによってポリシーをstale_tolerated以外に変更することはできません。
- IM列ストアは、LONGまたはLONG RAW列、表外列(LOB列、VARRAY列、ネストした表の列)または拡張データ型列をサポートしません。IM列ストアの表を有効化し、これらの型の列が含まれる場合、列はIM列ストアに移入されません。
- IM列ストアに対して仮想列を含む表を有効化した場合、列がIM列ストアに移入されるのは、INMEMORY_VIRTUAL_COLUMNS初期化パラメータの値がENABLEDで、仮想列のSQL式がIM列ストアに対して有効化されている列のみを参照する場合のみです。

関連項目:

IM列ストアの概要は、『Oracle Database In-Memoryガイド』を参照してください。

inmemory_attributes

inmemory_memcompress、inmemory_priority、inmemory_distributeおよびinmemory_duplicate句を使用して、表データをIM列ストアに格納する方法を指定します。

inmemory_memcompress

この句を使用して、IM列ストアに格納される表データの圧縮方法を指定します。このデータは、インメモリー・データと呼ばれます。インメモリー・データを圧縮しないようデータベースに指示するには、NO MEMCOMPRESSを指定します。インメモリー・データを圧縮するようデータベースに指示するには、MEMCOMPRESS FORの後に次の方法のいずれかを指定します。

- DML - この方法はDML操作に対して最適化され、ほとんどデータ圧縮は実行されません。
- QUERY - QUERYの指定は、QUERY LOWの指定と同じです。
- QUERY LOW - この方法は最小限にインメモリー・データを圧縮する(DMLを除く)ため、最適な問合せパフォーマンスが得られます。これはデフォルトです。
- QUERY HIGH - この方法はQUERY LOWよりインメモリー・データを圧縮しますが、CAPACITY LOWほどではありません。
- CAPACITY - CAPACITYの指定は、CAPACITY LOWの指定と同じです。
- CAPACITY LOW - この方法はQUERY HIGHよりインメモリー・データを圧縮し、CAPACITY HIGHほどではありませんが、優れた問合せパフォーマンスが得られます。
- CAPACITY HIGH - この方法は最大限にインメモリー・データを圧縮し、適切な問合せパフォーマンスを得られます。

memcompressレベルはDDLを介して指定できますが、移入時には無視されます。すべてのインメモリー圧縮ユニット(IMCU)は、QUERY LOWとして透過的に移入されます。

inmemory_priority

PRIORITY句を使用して、IM列ストアの表データのデータ移入優先度を指定します。この句が制御するのは、移入の優先度であり、移入の速度ではありません。

- オンデマンドの移入にはNONEを指定します。この場合、表が全表スキャンによってアクセスされるときに、データベースはIM列ストアに表データを移入します。表がアクセスされない場合、またはROWIDによる索引スキャンまたはフェッチによってのみアクセスされる場合は、移入は行われません。これはデフォルトです。
- 優先度ベースの移入には、優先度レベルLOW、MEDIUM、HIGHまたはCRITICALのいずれかを指定します。この場合、データベースは、内部的に管理された優先度キューを使用してIM列ストアに表データを自動的に移入します。全体スキャンは移入に必要な条件ではありません。データベースは、指定された優先度レベルに基づいて表データの移入をキューに入れます。たとえば、INMEMORY PRIORITY CRITICALが設定された表は、INMEMORY PRIORITY HIGHが設定された表より優先され、同様に、この表はINMEMORY PRIORITY LOWが設定された表より優先されます。IM列ストアに十分な領域がない場合、データベースは領域が使用可能になるまで追加の表データを移入しません。

inmemory_distribute

Oracle Real Application Clusters (Oracle RAC)またはOracle Active Data Guardを使用している場合のみ、DISTRIBUTE句を適用できます。これを使用すると、IM列ストアの表データをOracle RACインスタンス間で分散する方法を指定し、データの移入に適したデータベース・インスタンスを指定できます。

AUTOおよびBY

AUTO句およびBY句を使用して、IM列ストアの表データをOracle RACインスタンス間で分散する方法を指定します。次のオプションを指定できます。

- AUTO - Oracle DatabaseがOracle RACインスタンス間のデータの分散方法を制御します。アクセス・パターンに応じて、大きい表がOracle RACインスタンス間で分散されます。小さい表がインスタンス間で分散される場合があります。これはデフォルトです。
- BY ROWID RANGE - 特定の範囲のROWIDのデータが異なるOracle RACインスタンスに分散されます。
- BY PARTITION - パーティションのデータが異なるOracle RACインスタンスに分散されます。
- BY SUBPARTITION - サブパーティションのデータが異なるOracle RACインスタンスに分散されます。

AUTOおよびBYは、Active Data Guardのプライマリ・インスタンスとスタンバイ・インスタンスの間ではなく、単一のOracle RACデータベースのインスタンス間でオブジェクトのインメモリー圧縮単位(IMCU)を分散する場合にのみ使用できます。

FOR SERVICE

FOR SERVICE句を使用して、オブジェクトの移入に適したOracle RACまたはOracle Active Data Guardのインスタンスを指定します。次のオプションを指定できます。

- DEFAULT - このオブジェクトは、PARALLEL_INSTANCE_GROUP初期化パラメータで指定したすべてのインスタンスに対する移入の対象となります。このパラメータが設定されていない場合は、すべてのインスタンスでオブジェクトが移入されます。これはデフォルトです。
- ALL - PARALLEL_INSTANCE_GROUP初期化パラメータの値にかかわらず、このオブジェクトはすべてのインスタンスに対する移入の対象になります。
- service_name - オブジェクトは指定したサービスに属するインスタンスに対してのみ、およびサービスがアクティブでインスタンス上でブロックされていない場合にのみ移入の対象となります。
- NONE - このオブジェクトは、すべてのインスタンスに対する移入の対象になりません。このオプションを使用すると、表の他のインメモリー属性を維持しつつ、IM列ストアへの移入を無効にできます。これらの属性が有効になるのは、FOR SERVICE DEFAULT、FOR SERVICE ALL、またはALTER TABLE文のinmemory_distribute句のFOR

SERVICE service_nameを指定して、表のIM列ストアへの移入を後から有効にする場合です。

Oracle RACでは、FOR SERVICE句は、Oracle RACデータベース内のインスタンスを指定します。Active Data Guardでは、プライマリ・データベースおよびスタンバイ・データベースは単一インスタンスまたはOracle RAC構成を使用できます。Active Data Guardでは、FOR SERVICE句でプライマリ・データベースのインスタンス、スタンバイ・データベースのインスタンスまたはプライマリ・インスタンスとスタンバイ・インスタンスの組合せを指定します。

inmemory_duplicate

エンジニアド・システムでOracle Real Application Clusters (Oracle RAC)を使用している場合のみ、DUPLICATE句を適用できます。IM列ストアの表データをOracle RACインスタンス間で複製する方法を制御します。次のオプションを指定できます。

- DUPLICATE - データが1つのOracle RACインスタンスで複製され、合計2つのOracle RACインスタンスにデータが存在するようになります。
- DUPLICATE ALL - データがすべてのOracle RACインスタンス間で複製されます。DUPLICATE ALLを指定する場合、inmemory_distribute句の指定方法に関係なく、データベースはDISTRIBUTE AUTO設定を使用します。
- NO DUPLICATE - データがOracle RACインスタンス間で複製されません。これはデフォルトです。

inmemory_column_clause

この句を使用して、IM列ストアの特定の表の列を有効化または無効化し、特定の列のデータ圧縮方法を指定します。NO INMEMORY表の作成時にこの句を指定した場合、IM列ストアで表またはパーティションが後で有効になると、列設定が有効になります。

- INMEMORYを指定して、IM列ストアの指定された表の列を有効化します。

  オプションでinmemory_memcompress句を使用して、特定の列のデータ圧縮方法を指定できます。「inmemory_memcompress」を参照してください。inmemory_memcompress句を省略すると、表の列には、表のデータ圧縮方法が使用されます。特定の表の列のPRIORITY、DISTRIBUTEまたはDUPLICATE設定を指定できません。これらの設定は、表とすべての表の列で同じです。
- NO INMEMORYを指定して、IM列ストアの指定された表の列を無効化します。

inmemory_column_clauseを省略すると、すべての表の列には、表のIM列ストア設定が使用されます。

inmemory_column_clauseの制限事項

- この句は、LONGまたはLONG RAW列、表外列(LOB列、VARRAY列、ネストした表の列)あるいは拡張データ型列に対して指定できません。
- IM列ストアに対して仮想列を選択的に有効化するには、INMEMORY_VIRTUAL_COLUMNS初期化パラメータの値がENABLEDまたはMANUALで、仮想列のSQL式がIM列ストアに対して有効化されている列のみを参照する必要があります。

inmemory_clause

この句を使用して、IM列ストアの表のパーティションを有効化または無効化します。この句を指定するには、IM列ストアに対して表を有効化する必要があります。この句を省略すると、表のパーティションには、表のIM列ストア設定が使用されます。

inmemory_attributes句は、表のパーティションと表に同じセマンティクスを持ちます。詳細は、inmemory_attributes句を参照してください。

ORACLE_HIVE、ORACLE_HDFSおよびORACLE_BIGDATAのドライバ・タイプを使用する非パーティション表にINMEMORYを指定できます。

ilm_clause

この句を使用すると、自動データ最適化ポリシーをtableに追加できます。

この句のセマンティクスは、CREATE TABLEおよびALTER TABLEで同じですが、追加で次の制限事項があります。CREATE TABLEに指定できるのは、ADD POLICY句のみです。この句のセマンティクスの詳細は、「ilm_clause」を参照してください。

関連項目:

自動データ最適化ポリシーの管理の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

自動データ最適化の制限事項

自動データ最適化には、次の制限事項があります。

- オブジェクト・タイプ、索引構成表、クラスタ化表またはマテリアライズド・ビューを含む表では自動データ最適化はサポートされません。
- 時制有効性をサポートする表またはインデータベース・アーカイブの行アーカイブが有効な表では、行レベルのポリシーはサポートされません。

ilm_policy_clause

この句を使用すると、自動データ最適化ポリシーの説明を記述できます。

この句のセマンティクスは、CREATE TABLEおよびALTER TABLEで同じです。この句のセマンティクスの詳細は、「ilm_policy_clause」を参照してください。

RECOVERABLE | UNRECOVERABLE

これらのキーワードは以前のリリースで非推奨になったもので、それぞれLOGGINGおよびNOLOGGINGに置き換えられています。RECOVERABLEおよびUNRECOVERABLEは、下位互換性のためにサポートされていますが、LOGGINGおよびNOLOGGINGキーワードを使用することをお薦めします。

[UN]RECOVERABLEの制限事項

この句には、次の制限事項があります。

- パーティション表またはLOB記憶特性にRECOVERABLEを指定できません。
- パーティション表または索引構成表にUNRECOVERABLEを指定できません。
- AS subqueryでのみUNRECOVERABLEを指定できます。

ORGANIZATION

ORGANIZATION句を指定すると、表のデータ行が格納される順序を指定できます。

HEAP

HEAPを使用すると、tableのデータ行の格納順序を特定しないことを指定できます。これはデフォルトです。

INDEX

INDEXを使用すると、tableを索引構成表として作成することを指定できます。索引構成表では、表の主キーが定義された索引内にデータ行が格納されます。

EXTERNAL

EXTERNALを使用すると、表がデータベースの外部にある読取り専用表であることを指定できます。

関連項目:

「外部表の例」

index_org_table_clause

index_org_table_clauseを使用すると、索引構成表を作成できます。表の行(主キー列の値と非キー列の値の両方)は、主キーに基づいて作成された索引に格納されます。このため、索引構成表は主キーベースのアクセスおよび操作に最適です。索引構成表は、次のいずれかの表のかわりです。

- CREATE INDEX文を使用して主キー・ベースで索引付けされるクラスタ化されていない表。
- 索引クラスタに格納されるクラスタ表。索引クラスタは、表に対する主キーをクラスタ・キーにマップするCREATE CLUSTER文を使用して作成されます。

主キーは行を一意に識別するため、索引構成表には主キーを指定してください。主キーにはDEFERRABLEを指定できません。索引構成表の行に直接アクセスする場合は、ROWIDのかわりに主キーを使用してください。

索引構成表がパーティション化され、LOB列を含む場合、最初にindex_org_table_clause、次にLOB_storage_clause、その後に適切なtable_partitioning_clausesを指定する必要があります。

索引構成表を作成する場合は、CREATE TABLE ... AS SELECT文の副問合せで、TO_LOBファンクションを使用してLONG列をLOB列に変換することはできません。LONG列を含まない索引構成表を作成し、INSERT ...AS SELECT文でTO_LOBファンクションを使用してください。

索引構成表のROWID疑似列は、物理ROWIDではなく、論理ROWIDを戻します。データ型ROWIDとして作成した列には、IOTの論理ROWIDを格納できません。データ型ROWIDの列に格納できるデータは、ヒープ構成表のROWIDのみです。IOTの論理ROWIDを格納する場合は、かわりに型UROWIDの列を作成します。データ型UROWIDの列には、物理ROWIDと論理ROWIDの両方を格納できます。

関連項目:

「索引構成表の例」

索引構成表の制限事項

索引構成表には、次の制限事項があります。

- 索引構成表には、仮想列は定義できません。
- 索引構成表には、composite_range_partitions、composite_list_partitionsまたはcomposite_hash_partitions句は指定できません。
- 索引構成表がネストした表またはVARRAYである場合は、table_partitioning_clausesを指定できません。
- 索引構成表の主キーに属する文字データ型の列の照合は、BINARY、USING_NLS_COMP、USING_NLS_SORT

またはUSING_NLS_SORT_CSである必要があります。

PCTTHRESHOLD integer

索引ブロック内で、索引構成表の行を格納するために確保されている領域の割合を指定します。PCTTHRESHOLDは、主キーを保持するために十分な大きさである必要があります。指定したしきい値を超える列から始まる行の後続列はすべて、オーバーフロー・セグメントに格納されます。PCTTHRESHOLDは1から50の値を取る必要があります。PCTTHRESHOLDを指定しない場合のデフォルト値は50です。

PCTTHRESHOLDの制限事項

PCTTHRESHOLDは、索引構成表の個別パーティションに対して指定できません。

mapping_table_clauses

MAPPING TABLEを指定すると、ローカルから物理ROWIDへのマッピングを作成してヒープ構成表に格納するようデータベースに指示できます。このマッピングは、索引構成表のビットマップ索引の作成に必要です。索引構成表がパーティション化されている場合、マッピング表もパーティション化され、マッピング表のパーティションの名前および物理属性は実表のパーティションと同じになります。

マッピング表またはマッピング表のパーティションは、親である索引構成表またはパーティションと同じ表領域に作成されます。マッピング表またはそのパーティションの記憶特性に対して、問合せ、DML操作または変更は実行できません。

prefix_compression

prefix_compression句を使用すると、索引構成表の接頭辞圧縮を有効または無効にできます。

- COMPRESSを指定して、索引構成表に対してキー圧縮とも呼ばれる接頭辞圧縮を有効化します。これによって、索引構成表の主キー列の値が重複しなくなります。integerを使用して、接頭辞の長さ(圧縮する接頭辞列数)を指定します。

  接頭辞の長さの有効範囲は、1から(主キー列数-1)までです。デフォルトでは(主キー列数-1)になります。

- NOCOMPRESSを指定すると、索引構成表での接頭辞圧縮が使用禁止になります。これはデフォルトです。

索引構成表の接頭辞圧縮の制限事項

パーティション・レベルでは、COMPRESSを指定できますが、integerで接頭辞の長さを指定できません。

index_org_overflow_clause

index_org_overflow_clauseを指定すると、指定されたしきい値を超える索引構成表のデータ行を、この句で指定したデータ・セグメントに格納するようデータベースに指示できます。

- 索引構成表を作成した場合、各列の最大サイズが評価され、行の最大値が計算されます。オーバーフロー・セグメントが必要で、OVERFLOWを指定していない場合は、エラーが発生しCREATE TABLE文は実行されません。このチェック機能によって、索引構成表に対する後続のDML操作が、オーバーフロー・セグメントがないために失敗することを回避できます。
- OVERFLOWキーワードの後の句に指定するすべての物理属性および記憶特性は、表のオーバーフロー・セグメントにのみ適用されます。索引構成表自体の物理属性と記憶特性、すべてのパーティションに対するデフォルト値、および各パーティションに対する値は、このキーワードの前に指定する必要があります。
- 索引構成表に1つ以上のLOB列が含まれる場合は、LOBが表内に格納できるほど小さい場合でも、OVERFLOWを指定しないと、表外に格納されます。

- 表がパーティション化されている場合、オーバーフロー・データ・セグメントが主キー索引セグメントと同一レベルでパーティション化されます。

INCLUDING column_name

索引構成表の行を索引部分とオーバーフロー部分に分割する列を指定します。主キー列は常に索引に格納されます。column_nameは、最後の主キー列でも主キー以外の列でもかまいません。column_nameに続くすべての主キー以外の列は、オーバーフロー・データ・セグメントに格納されます。

column_name句で行を分割しようとした場合に、行の索引部分のサイズがPCTTHRESHOLDの指定値またはデフォルト値を超えると、PCTTHRESHOLDの値に基づいて行が切り離されます。

INCLUDING句の制限事項

この句は、索引構成表の個々のパーティションに対して指定できません。

EXTERNAL PARTITION ATTRIBUTES

ハイブリッド・パーティション表で表レベルの外部パラメータを指定するには、 EXTERNAL PARTITION ATTRIBUTES句を使用します。

external_table_clause

external_table_clauseを使用して外部表を作成します。これによって、データベース内からデータベース外に格納されているデータをデータベースにロードすることなく処理できます。

外部表を定義すると、データ・ディクショナリ内にメタデータのみが作成されます。これは、データベース外のデータを指し、このようなデータへのシームレスな読取り専用アクセスを提供します。

外部表の場合、データベースにデータが存在しないため、表の作成時に通常は使用可能な句の小規模のサブセットを定義します。

オペレーティング・ファイル・システムやビッグ・データ・ソースに存在する外部データ、およびHDFSやHiveなどの形式をサポートする以外に、Oracleでは、オブジェクトに存在する外部データをDBMS_CLOUDパッケージを介してサポートします。

オブジェクト・ストア内のデータを操作するには、DBMS_CLOUDパッケージを使用するか、外部表を手動で定義します。Oracle自律型データベースと完全に互換性のあるDBMS_CLOUDを追加機能に使用することをお薦めします。

関連項目:

- DBMS_CLOUD
- 外部表の管理

外部表の場合、データベースにデータが存在しないため、表の作成時に通常は使用可能な句の小規模のサブセットを定義します。

- relational_properties句内では、column、datatype、ENCRYPT、inline_constraintおよびout_of_line_constraintのみ指定できます。外部表にデータをロードするためにORACLE_DATAPUMPアクセス・ドライバおよびAS subquery句を指定する場合にのみ、ENCRYPT句を指定できます。inline_constraint句およびout_of_line_constraint句内では、CHECKを除くすべての副次句を指定できます。
- physical_properties_clause内では、表の構成(ORGANIZATION EXTERNAL

external_table_clause)のみを指定できます。

- table_properties句内では、parallel_clauseを指定できます。parallel_clauseを使用すると、外部データに対する後続の問合せおよび外部表を移入する後続の操作をパラレル化できます。

  Oracle Database 12cリリース2 (12.2)以降は、パーティション化された外部表を作成できます。このために、table_properties内では、次のtable_partitioning_clausesの副次句を指定できます。

  - range_partitions - レンジ・パーティションまたは時間隔パーティションの外部表を作成するには、この句を指定します。
  - list_partitions - リスト・パーティションの外部表を作成するには、この句を指定します。この句内では、AUTOMATIC句を指定できません。自動リスト・パーティション表を外部表にすることはできません。
  - composite_range_partitions - レンジ-レンジ、レンジ-リスト、時間隔-レンジまたは時間隔-リスト・コンポジット・パーティション外部表を作成するには、この句を指定します。
  - composite_list_partitions - リスト-レンジまたはリスト-リスト・コンポジット・パーティション外部表を作成するには、この句を指定します。この句内では、AUTOMATIC句を指定できません。自動コンポジット・パーティション表を外部表にすることはできません。

- 外部表は、作成時にAS subquery句を使用することによって移入できます。

同じCREATE TABLE文で他の句を指定することはできません。

関連項目:

- 外部表の例
- 列の投影のデフォルトのプロパティを変更することによる影響の詳細は、「ALTER TABLE」の「PROJECT COLUMN句」を参照してください。
- 外部表の使用方法の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』、『Oracle Database管理者ガイド』および『Oracle Databaseユーティリティ』を参照してください。

外部表の制限事項

外部表には、次の制限事項があります。

- 外部表を一時表にすることはできません。
- 外部表に指定できる制約のタイプは、NOT NULL制約、一意制約、主キー制約および外部キー制約のみです。一意制約、主キー制約または外部キー制約を指定する場合は、RELY DISABLEも指定する必要があります。これらの制約は宣言的であり、適用されません。より多くのオプティマイザ変換を考慮できるため、問合せパフォーマンスが向上し、リソース使用量が削減される可能性があります。オプティマイザがこれらのRELY DISABLE制約を利用できるようにするには、QUERY_REWRITE_INTEGRITY初期化パラメータをtrustedまたはstale_toleratedに設定する必要があります。
- 外部表に索引を作成することはできません。
- 外部表には、INVISIBLEの列を格納できません。
- 外部表にオブジェクト型、VARRAYまたはLONG列を含めることはできません。ただし、内部データベース表のVARRAYまたはLONGデータを外部表のLOB列に移入することはできます。

- インメモリー列ストアに移入できる外部表で許可されているアクセス・タイプは、ORACLE_LOADERとORACLE_DATAPUMPのみです。

TYPE

TYPE access_driver_typeを指定すると、外部表のアクセス・ドライバを指定できます。アクセス・ドライバは、データベースに対する外部データを解析するAPIです。Oracle Databaseでは次のアクセス・ドライバが提供されています: ORACLE_LOADER、ORACLE_DATAPUMP、ORACLE_HDFSおよびORACLE_HIVE。TYPEを指定しない場合、デフォルトのアクセス・ドライバORACLE_LOADERが使用されます。AS subquery句を指定して1つのOracle Databaseからデータをアンロードし、同じ、または異なるOracle Databaseに再ロードする場合、ORACLE_DATAPUMPアクセス・ドライバを指定する必要があります。

関連項目:

ORACLE_LOADER、ORACLE_DATAPUMP、ORACLE_HDFSおよびORACLE_HIVEアクセス・ドライバの詳細は、『Oracle Databaseユーティリティ』を参照してください。

DEFAULT DIRECTORY

DEFAULT DIRECTORYを指定すると、外部データソースが存在するファイル・システムのディレクトリに対応するデフォルト・ディレクトリ・オブジェクトを1つ指定できます。デフォルト・ディレクトリは、アクセス・ドライバから使用でき、エラー・ログなどの補助ファイルを格納できます。

ACCESS PARAMETERS

オプションのACCESS PARAMETERS句を指定すると、この外部表用の特定のアクセス・ドライバのパラメータに値を割り当てることができます。

- opaque_format_specは、ORACLE_LOADER、ORACLE_DATAPUMP、ORACLE_HDFSおよびORACLE_HIVEアクセス・ドライバのすべてのアクセス・パラメータを指定します。ORACLE_LOADER、ORACLE_DATAPUMP、ORACLE_HDFSおよびORACLE_HIVEアクセス・パラメータの詳細は、『Oracle Databaseユーティリティ』を参照してください。

  opaque_format_specで指定するフィールド名は、表定義の列と一致している必要があります。表定義の列と一致していないopaque_format_specのフィールドは無視されます。

- USING CLOB subqueryを指定すると、副問合せを使用して、パラメータおよびその値を導出できます。副問合せには、集合演算子またはORDER BY句を含めません。CLOBデータ型の1つの項目を含む単一行を戻します。

opaque_format_specでパラメータを指定するか、副問合せを使用してそれらを導出するかに関係なく、この句は解析されません。この情報は、外部データに照らしてアクセス・ドライバによって解析されます。

インライン外部表および外部変更問合せ文では、opaque_format_specを一重引用符で囲んで使用する必要があります。DDL文では、opaque_format_specは単一引用符で囲まずに使用する必要があります。

LOCATION

LOCATION句を使用すると、1つ以上の外部データソースを指定できます。通常、location_specifierはファイルですが、ファイル以外も指定できます。Oracle Databaseでは、この句は解析されません。この情報は、外部データに照らしてアクセス・ドライバによって解析されます。

LOCATION句は、次のように指定する必要があります。

- パーティション化されていない外部表を作成する場合は、external_table_data_props句の表レベルでLOCATION句を指定する必要があります。
- パーティション化された外部表を作成する場合は、external_part_subpart_data_props句のパーティション・レベルでLOCATION句を指定する必要があります。
- コンポジット・パーティション外部表を作成する場合は、external_part_subpart_data_props句のサブパーティション・レベルでLOCATION句を指定する必要があります。

REJECT LIMIT

REJECT LIMIT句を指定すると、Oracle Databaseエラーが戻され、問合せが異常終了するまでに、外部データの問合せで許容される変換エラーの数を指定できます。デフォルトの値は0。

CLUSTER句

CLUSTER句は、表がclusterの一部であることを示します。この句で指定する各列は、クラスタの各列に対応する表の列となります。一般に、表のクラスタ列は、主キーまたは主キーの一部を構成する1つ以上の列です。詳細は、「CREATE CLUSTER」を参照してください。

クラスタ・キー内の列ごとに表から1つの列を指定します。列は、名前ではなく位置で一致させます。

クラスタ表はクラスタの領域割当てを使用します。このため、PCTFREE、PCTUSEDまたはINITRANSパラメータ、TABLESPACE句またはstorage_clauseをCLUSTER句とともに使用しないでください。

クラスタ表の制限事項

クラスタ表には、次の制限事項があります。

- オブジェクト表、およびLOB列またはOracleが提供するAny*型の列を含む表はクラスタの一部にはできません。
- クラスタの一部である表にparallel_clause、CACHEまたはNOCACHEは指定できません。
- クラスタが同じROWDEPENDENCIESまたはNOROWDEPENDENCIES設定で作成されていないかぎり、CLUSTERをROWDEPENDENCIESまたはNOROWDEPENDENCIESとともに指定することはできません。
- クラスタ表には、INVISIBLEの列を格納できません。

table_properties

table_propertiesを使用すると、表の特性をさらに詳しく定義できます。

column_properties

column_properties句を使用すると、列の記憶域属性を指定できます。

object_type_col_properties

object_type_col_propertiesを使用すると、オブジェクト列、属性、あるいは列または属性の集合要素の記憶特性を指定できます。

column

columnには、オブジェクト列または属性を指定します。

substitutable_column_clause

substitutable_column_clauseを使用すると、同じ階層のオブジェクト列または属性が互いに置換可能かどうかを指定できます。列が特定の型であるか、サブタイプのインスタンスを含むものであるか、またはその両方を指定できます。

- ELEMENTを指定すると、コレクション列または属性の要素型が宣言した型のサブタイプに制約されます。
- IS OF [TYPE] (ONLY type)句を指定すると、オブジェクト列の型が宣言した型のサブタイプに制約されます。
- NOT SUBSTITUTABLE AT ALL LEVELSを指定すると、オブジェクト列がサブタイプに対応するインスタンスを持つことはできないことを指定できます。また、置換は、埋込みオブジェクト属性、埋込みのネストした表およびVARRAYの要素には使用できません。デフォルトは、SUBSTITUTABLE AT ALL LEVELSです。

substitutable_column_clauseの制限事項

この句には、次の制限事項があります。

- この句は、オブジェクト列の属性には指定できません。ただし、オブジェクト表自体の代替性が設定されていない場合、リレーショナル表におけるオブジェクト型の列およびオブジェクト表のオブジェクト列に対してこの句を指定できます。
- コレクション型の列の場合、この句で指定できる部分は[NOT] SUBSTITUTABLE AT ALL LEVELSのみです。

LOB_storage_clause

LOB_storage_clauseを使用すると、LOBデータ・セグメントの記憶域属性を指定できます。STORE ASキーワードの後に、1つ以上の句を指定する必要があります。複数の句を指定する場合は、構文図で上から下に表示されている順に指定する必要があります。

非パーティション表の場合、この句は、表のLOBデータ・セグメントの記憶域属性を指定します。

パーティション表の場合、この句は指定した位置に応じて実装されます。

- 表レベルで指定されたパーティション表の場合(パーティション句とともにphysical_properties句で指定した場合)、この句は、各パーティションまたはサブパーティションに関連付けられたLOBデータ・セグメントに対するデフォルト記憶域属性を指定します。この記憶域属性は、パーティションまたはサブパーティション・レベルでLOB_storage_clauseによって上書きされないかぎり、すべてのパーティションまたはサブパーティションに適用されます。
- パーティション表の各パーティションの場合(table_partition_descriptionの一部として指定した場合)、この句は、そのパーティションのデータ・セグメントの記憶域属性、またはこのパーティションのサブパーティションのデフォルト記憶域属性を指定します。パーティション・レベルのLOB_storage_clauseは、表レベルのLOB_storage_clauseを上書きします。
- パーティション表の各サブパーティションの場合(subpartition_by_hashまたはsubpartition_by_listの一部として指定した場合)、この句は、サブパーティションのデータ・セグメントの記憶域属性を指定します。サブパーティション・レベルのLOB_storage_clauseは、パーティション・レベルおよび表レベルのLOB_storage_clausesを上書きします。

LOB_storage_clauseの制限事項:

サブパーティションでLOB_storage_clauseを指定する場合、使用できるのはTABLESPACE句のみです。

関連項目:

- サイズがGBになるLOBを作成する場合のガイドラインを含むLOBの詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。
- 表の作成: LOB列の例

LOB_item

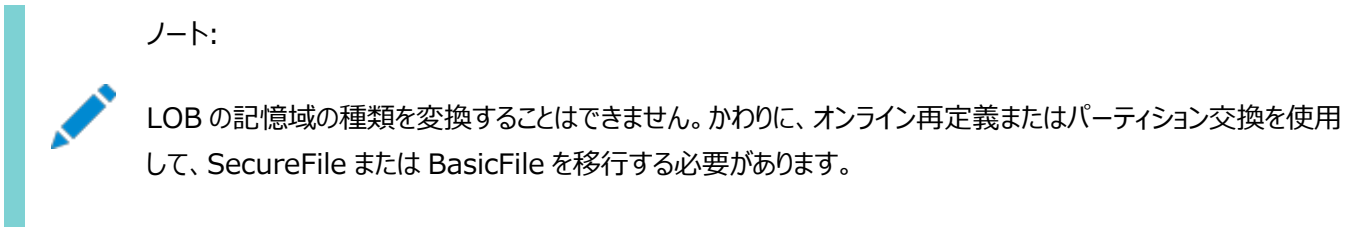
表の表領域および記憶特性とは異なる表領域および記憶特性を明示的に定義する場合に、そのLOB列名またはLOBオブジェクト属性を指定します。作成する各LOB_itemに、システム管理された索引が自動的に作成されます。

SECUREFILE | BASICFILE

この句を使用して、LOB記憶域のタイプに、高パフォーマンスのLOB (SecureFiles)または従来のLOB (BasicFiles)を指定します。

関連項目:

SecureFiles LOBの詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。



ノート:

LOB の記憶域の種類を変換することはできません。かわりに、オンライン再定義またはパーティション交換を使用して、SecureFile または BasicFile を移行する必要があります。

LOB_segname

LOBデータ・セグメントの名前を指定します。LOB_itemが複数指定されている場合は、LOB_segnameを使用できません。

LOB_storage_parameters

LOB_storage_parameters句を使用すると、様々なLOB記憶域の要素を指定できます。

TABLESPACE句

この句を使用して、LOBデータが格納される表領域を指定します。

TABLESPACE SET句

この句は、CREATE TABLEのSHARDEDキーワードを指定してシャード表を作成する場合にのみ有効です。この句を使用して、LOBデータが格納される表領域セットを指定します。

storage_clause

storage_clauseを使用すると、LOBセグメント記憶域の様々な側面を指定できます。LOB記憶域に関して特に重要なのは、storage_clauseのMAXSIZE句です。これは、LOB_parametersのLOB_retention_clauseと組み合せて使用できます。詳細は、「storage_clause」を参照してください。

LOB_parameters

LOB記憶域にSecureFileを使用する場合、いくつかのLOB_parametersは不要になります。PCTVERSIONおよびFREEPOOLSパラメータは、BasicFiles LOB記憶域を使用する場合にのみ有効かつ有用です。

ENABLE STORAGE IN ROW

行の記憶域を有効にした場合、LOB値の長さが、約4000バイトからシステム制御情報分を引いた長さより小さければ、LOB値が行(インライン)に格納されます。これはデフォルトです。

行の記憶域の有効化の制限事項

index_org_table_clauseでOVERFLOWセグメントを指定しないかぎり、索引構成表に対して、このパラメータを指定できません。

DISABLE STORAGE IN ROW

行の記憶域を無効にした場合、LOB値の長さに関係なく、LOB値は表外(行の外側)に格納されます。

LOB値が格納されている場所にかかわらず、LOBロケータは、常に表内に格納されます。STORAGE IN ROWの値は、一度設定すると、表を移動しないかぎり、変更できません。詳細は、「ALTER TABLE」の「move_table_clause」を参照してください。

CHUNK integer

LOBの操作用に割り当てるバイト数を指定します。integerにデータベースのブロック・サイズの倍数を指定しなかった場合、自動的に次に大きい倍数(バイト単位)に切り上げられます。たとえば、データベースのブロック・サイズが2048バイトのときにintegerに2050を指定すると、4096バイト(2ブロック)が割り当てられます。最大値は32768(32KB)で、これがOracle Databaseのブロック・サイズとして使用できる最も大きな値です。デフォルトのCHUNKサイズは、Oracleでの1データベース・ブロックです。

CHUNKの値は、NEXTの値(デフォルト値またはstorage_clauseで指定された値)以下である必要があります。CHUNKの値がNEXTの値を超えると、エラーが戻ります。CHUNKの値は、一度設定すると変更できません。

PCTVERSION integer

LOBの記憶域全体のうち、旧バージョンのLOBの保持に使用される割合(パーセント)の最大値を指定します。データベースが手動UNDOモードで実行されている場合、デフォルト値は10です。この場合、LOB記憶域全体の10%が消費されるまで、旧バージョンのLOBデータは上書きされません。

データベースが手動UNDOモードと自動UNDOモードのどちらで稼働されていても、PCTVERSIONパラメータを指定できます。PCTVERSIONは、手動UNDOモードのデフォルト値です。RETENTIONは、自動UNDOモードのデフォルト値です。PCTVERSIONとRETENTIONの両方は指定できません。

この句は、SECUREFILEを指定した場合は無効です。SECUREFILEとPCTVERSIONの両方を指定した場合、PCTVERSIONパラメータは特に警告もなく無視されます。

LOB_retention_clause

この句を使用すると、LOBセグメントを保持する用途に、フラッシュバック、読取り一貫性、その両方、またはどちらでもないを指定できます。

データベースが自動UNDOモードで稼働している場合にのみ、RETENTIONパラメータを使用できます。データベースに保持されるコミット済のUNDOデータの量は、UNDO_RETENTION初期化パラメータの値を使用して決定されます。自動UNDOモードでは、PCTVERSIONを指定しないかぎり、RETENTIONがデフォルト値となります。PCTVERSIONとRETENTIONの両方は指定できません。

SecureFileを使用している場合にのみ、RETENTIONの後にオプションの設定を指定できます。LOB_storage_clauseのSECUREFILEパラメータは、データベースがSecureFileを使用して記憶域を動的に管理することを示します。データベースのUNDOモードなどの要因が考慮されます。

- MAX: LOBセグメントがMAXSIZEに達するまでUNDOを保持するように指定します。MAXを指定する場合は、storage_clauseでMAXSIZE句も指定する必要があります。
- MIN: データベースがフラッシュバック・モードで、特定のLOBセグメントのUNDO保存期間をn秒に制限する場合に指定します。

- AUTO: 読取り一貫性に必要十分なUNDOを保持します。
- NONE: 読取り一貫性またはフラッシュバックのどちらにもUNDOが必要ない場合に指定します。

RETENTIONパラメータを指定しないか、オプションの設定なしでRETENTIONを指定すると、RETENTIONがDEFAULTに設定されます。この機能はAUTOと同等です。

関連項目:

- SecureFileを使用して簡略化されたLOB記憶域の詳細は、「CREATE TABLE」の「LOB_storage_parameters」句を参照してください。
- SecureFileの使用の詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。
- データベースをフラッシュバック・モードにする方法については、「ALTER DATABASE」の「flashback_mode_clause」 を参照してください。
- UNDO表領域の作成: 例

FREEPOOLS integer

LOBセグメントに対する空きリストのグループ数を指定します。通常、integerは、Oracle Real Application Clusters環境のインスタンス数です。単一インスタンス・データベースの場合、この値は1になります。

データベースが自動UNDOモードで稼働している場合にのみ、このパラメータを指定できます。このモードでは、storage_clauseでFREELIST GROUPSパラメータを指定しないかぎり、FREEPOOLSがデフォルト値になります。FREEPOOLSとFREELIST GROUPSのどちらも指定しない場合、データベースが自動UNDO管理モードで稼働している場合はFREEPOOLS 1のデフォルト値が使用され、手動UNDO管理モードで稼働している場合はFREELIST GROUPS 1のデフォルト値が使用されます。

この句は、SECUREFILEを指定した場合は無効です。SECUREFILEとFREEPOOLSの両方を指定した場合、FREEPOOLSパラメータは特に警告もなく無視されます。

FREEPOOLSの制限事項

storage_clauseのFREEPOOLSおよびFREELIST GROUPSパラメータは指定できません。

LOB_deduplicate_clause

この句は、SecureFiles LOBに対してのみ有効です。LOB_deduplicate_clauseを使用すると、重複するLOBデータを除外する、LOBの重複の除外を有効または無効にできます。

DEDUPLICATEキーワードは、LOBの重複コピーを除外するようデータベースに対して指定します。セキュアなハッシュ索引を使用して重複を検出すると、同じ内容を持つLOBは単一のコピーに結合され、消費される記憶域が削減されて記憶域の管理が簡素化されます。

この句を指定しない場合、デフォルトでは、LOBの重複除外は無効です。

この句は、LOBセグメント全体に対して、LOBの重複除外を実装します。個々のLOBに対して重複除外を有効または無効にするには、DBMS_LOB.SETOPTIONSプロシージャを使用します。

関連項目:

LOBの重複除外の詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。DBMS_LOBパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

LOB_compression_clause

この句は、BasicFiles LOBではなく、SecureFiles LOBに対してのみ有効です。LOB_compression_clauseを使用すると、サーバー側のLOB圧縮を有効または無効にすることをデータベースに指定できます。サーバー側の圧縮されたLOBセグメントで、ランダムな読取り/書込みアクセスが可能です。LOB圧縮は、表の圧縮または索引の圧縮からは独立しています。この句を指定しない場合、NOCOMPRESSがデフォルトになります。

HIGH、MEDIUMまたはLOWを指定して、圧縮の程度を変更できます。圧縮の程度をHIGHにすると、待機時間はMEDIUMよりも長くなりますが、圧縮率は高くなります。LOWを指定すると、HIGHまたはMEDIUMのいずれの場合よりも圧縮率がわずかに低くなるものの、解凍および圧縮の処理速度が大幅に向上します。このオプションのパラメータを指定しない場合、MEDIUMがデフォルトになります。

この句は、LOBセグメント全体のサーバー側のLOB圧縮を実装します。個々のLOBの圧縮を有効または無効にするには、DBMS_LOB.SETOPTIONSプロシージャを使用します。

関連項目:

サーバー側のLOB記憶域の詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。提供されているパッケージUTL_COMPRESSを使用したクライアント側のLOB圧縮およびDBMS_LOBパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

ENCRYPT | DECRYPT

これらの句は、LOB記憶域にSecureFileを使用しているLOBに対してのみ有効です。ENCRYPTを指定すると、列内のすべてのLOBを暗号化できます。DECRYPTを指定すると、LOBをクリアテキストで保持できます。この句を指定しない場合、DECRYPTがデフォルトになります。

この句の概要は、「encryption_spec」を参照してください。LOB列に適用するとencryption_specは個々のLOB列固有になるため、他のLOB列や他の非LOB列とは、暗号化アルゴリズムが異なる場合があります。column_definitionの一部としてencryption_specを使用すると、LOB列全体を暗号化できます。table_partition_descriptionでLOB_storage_clauseの一部としてencryption_specを使用すると、LOBパーティションを暗号化できます。

LOBのencryption_specの制限事項

LOB暗号化に対しては、encryption_specのSALT句またはNO SALT句を指定できません。

関連項目:

LOB暗号化の詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。DBMS_LOBパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

CACHE | NOCACHE | CACHE READS

これらの句の詳細は、「CACHE | NOCACHE | CACHE READS」を参照してください。

LOB_partition_storage

LOB_partition_storage句を使用すると、各パーティションに、別のLOB_storage_clauseまたはvarray_col_propertiesを指定できます。パーティションは、その位置の順に指定してください。パーティションの順番を確

認するには、USER_IND_PARTITIONSビューのPARTITION_NAMEおよびPARTITION_POSITION列を問い合せます。

特定のパーティションに、LOB_storage_clauseまたはvarray_col_properties句を指定しなかった場合、表レベルでLOB項目に指定された記憶特性が設定されます。表のレベルでもLOB項目に記憶特性を指定しなかった場合、LOBデータ・パーティションは、対応する表パーティションと同じ表領域に格納されます。

LOB_partition_storageの制限事項

LOB_partition_storageには、次の制限事項があります。

- LOB_storage_clauseのLOB_parameterでは、encryption_specを指定できません。パーティションおよびサブパーティションに対する暗号化アルゴリズムの指定は無効であるためです。
- ハッシュ・パーティションおよびすべてのタイプのサブパーティションには、TABLESPACE句のみ指定できます。

varray_col_properties

varray_col_propertiesを使用すると、VARRAY型のデータが格納されるLOBに対して、別々の記憶特性を指定できます。varray_itemがマルチレベル・コレクションの場合、varray_item内にネストされたすべてのコレクション項目は、常にvarray_itemと同じLOBに格納されます。

- 非パーティション表の場合(パーティション句なしでphysical_properties句に指定した場合)、この句は、VARRAYのLOBデータ・セグメントの記憶域属性を指定します。
- 表レベルで指定されたパーティション表の場合(パーティション句とともにphysical_properties句で指定した場合)、この句は、各パーティション(またはサブパーティション)に対応付けられたVARRAYのLOBデータ・セグメントに対するデフォルト記憶域属性を指定します。
- パーティション表の各パーティションの場合(table_partition_descriptionの一部として指定した場合)、この句は、そのパーティションのVARRAYのLOBデータ・セグメントの記憶域属性、またはこのパーティションのサブパーティションにあるVARRAYのLOBデータ・セグメントのデフォルト記憶域属性を指定します。パーティション・レベルのvarray_col_propertiesは、表レベルのvarray_col_propertiesを上書きします。
- パーティション表の各サブパーティションの場合(subpartition_by_hashまたはsubpartition_by_listの一部として指定した場合)、この句は、このサブパーティションのVARRAYデータ・セグメントの記憶域属性を指定します。サブパーティション・レベルのvarray_col_propertiesは、パーティション・レベルおよび表レベルのvarray_col_propertiesを上書きします。

STORE AS [SECUREFILE | BASICFILE] LOB句

STORE AS LOBを指定した場合に実行される処理は、次のとおりです。

- VARRAYの最大サイズが約4000バイト未満で、行の記憶域を使用禁止にしていない場合、VARRAYは表内LOBに格納されます。
- VARRAYの最大サイズが約4000バイトを超える場合または行の記憶域を使用禁止にしている場合、VARRAYは表外LOBとして格納されます。

STORE AS LOBを指定しなかった場合、記憶域は、VARRAY列の実際のサイズではなく、VARRAYの最大サイズに基づいて決定されます。VARRAYの最大サイズは、要素数×要素サイズ+システム制御情報分の容量です。この句を指定しない場合、次のようになります。

- VARRAYの最大サイズが約4000バイト未満の場合、VARRAYはLOBとしてではなく表内データとして格納されます。

- 最大サイズが約4000バイトを超える場合、VARRAYは常にLOBとして格納されます。
    - 実際のサイズが約4000バイト未満の場合、VARRAYは表内LOBとして格納されます。
    - 実際のサイズが約4000バイトを超える場合、VARRAYは表外LOBとして格納されます。これはその他のLOB列でも同様です。

substitutable_column_clause

`substitutable_column_clause`の動作は、「object_type_col_properties」で説明されている動作と同じです。

関連項目:

置換可能な表および列のサンプル

VARRAY列のプロパティの制限事項

この句は、時間隔パーティション表には指定できません。

nested_table_col_properties

`nested_table_col_properties`を使用すると、ネストした表に対して別々の記憶特性を指定し、そのネストした表を索引構成表として定義できるようになります。この句で特に明示的に指定しないかぎり、記憶表は次のとおり作成されます。

- 非パーティション表の場合、記憶表は親表と同じスキーマおよび同じ表領域内に作成されます。
- パーティション表の場合、記憶表はスキーマのデフォルトの表領域内に作成されます。デフォルトでは、ネストした表はパーティション実表でパーティション化されます。
- どちらの場合も、記憶表ではデフォルトの記憶特性が使用され、この表の作成の基になった列のネストした表の値が格納されます。

ネストした表の型を持つ列または列属性付きで表を作成する場合は、この句を挿入する必要があります。`nested_table_col_properties`句内で、親表に対する場合と同じ働きをする句については、ここでは説明しません。

nested_item

型がネストした表である列、またはその表のオブジェクト型の最上位の属性の名前を指定します。

COLUMN_VALUE

ネストした表がマルチレベル・コレクションの場合、内部のネストした表またはVARRAYには名前が割り当てられていない場合があります。この場合、`nested_item`名のかわりに`COLUMN_VALUE`を指定します。

関連項目:

`nested_item`および`COLUMN_VALUE`の使用例は、「表の作成: マルチレベル・コレクションの例」を参照してください。

LOCAL | GLOBAL

ネストした表を実表でパーティション化するには、`LOCAL`を指定します。これはデフォルトです。ネストしたパーティション表のローカル・パーティション索引は自動的に作成されます。

ネストした表が、パーティション実表のネストした非パーティション表であることを示すには、`GLOBAL`を指定します。

storage_table

nested_itemの行を含む表の名前を指定します。

storage_tableに対して問合せやDML文を直接実行することはできませんが、その記憶特性は、ALTER TABLE文で名前を指定することによって変更できます。

関連項目:

ネストした表の列に対する記憶特性の変更方法については、「ALTER TABLE」を参照してください。

RETURN [AS]

問合せの結果として何を戻り値とするかを指定します。

- VALUEは、ネストした表自体のコピーを戻します。
- LOCATORは、ネストした表のコピーに対するコレクション・ロケータを戻します。

  ロケータの有効範囲は1つのセッションであり、複数のセッションにわたって使用できません。LOBロケータとは異なり、コレクション・ロケータはコレクション・インスタンスの変更に使用できません。

segment_attributes_clauseまたはLOB_storage_clauseを指定しない場合、ネストした表はヒープ構成され、デフォルトの記憶特性で作成されます。

ネストした表の列のプロパティの制限事項

ネストした表の列のプロパティには、次の制限事項があります。

- この句は、一時表には指定できません。
- この句は、時間隔パーティション表には指定できません。
- oid_clauseは指定できません。
- 作成時、object_propertiesを使用して、out_of_line_ref_constraint、inline_ref_constraintまたはネストした表の属性に対する外部キー制約を指定することはできません。

関連項目:

- ネストした表の列に対する記憶特性の変更方法については、「ALTER TABLE」を参照してください。
- 「ネストした表の例」および「表の作成: マルチレベル・コレクションの例」を参照してください。

XMLType_column_properties

XMLType_column_propertiesを指定すると、XMLTYPE列に対する記憶域属性を指定できます。

XMLType_storage

XMLTypeデータは、バイナリXML、CLOBまたはオブジェクト・リレーショナル列に格納できます。

- BINARY XMLを指定すると、縮小されたバイナリXML書式でXMLデータを格納できます。

  指定したLOBパラメータは、バイナリXMLエンコード値を格納するために作成された、基礎となるBLOB列に適用されます。

  以前のリリースでは、バイナリXMLデータはデフォルトでBasicFiles LOBに格納されます。Oracle Database 11gリリース2 (11.2.0.2)以降では、COMPATIBLE初期化パラメータが11.2以上の場合にBASICFILEまたは

SECUREFILEを指定しないと、バイナリXMLデータは可能なかぎりSecureFiles LOBに格納されます。SecureFiles LOB記憶域を使用できない場合、バイナリXMLデータはBasicFiles LOBに格納されます。これは、次のいずれかの場合に発生します。

- XMLType表の表領域で、自動セグメント領域管理が使用されていない。
- init.oraファイルの設定により、SecureFiles LOB記憶域が使用できない。例については、『Oracle Databaseリファレンス』のDB_SECUREFILEパラメータを参照してください。

- CLOBを指定すると、CLOB列にXMLTypeデータを格納できます。CLOB列にデータを格納すると、元の内容が保持されるため、検索時間が短縮されます。

  LOB記憶域を定義する場合、LOBパラメータとXMLSchema_spec句のいずれかを指定できますが、両方は指定できません。XMLSchema_spec句を指定すると、特定のスキーマ・ベースのXMLインスタンスに表や列を制限できます。

  この句でBASICFILEまたはSECUREFILEを指定しないと、CLOB列はBasicFiles LOBに格納されます。

  ノート:

  XMLType データを CLOB 列に格納しないことをお薦めします。XMLType の CLOB 記憶域は非推奨になっています。かわりに XMLType のバイナリ XML 記憶域を使用してください。

- OBJECT RELATIONALを指定すると、オブジェクト・リレーショナル列にXMLTypeデータを格納できます。データ・オブジェクトをリレーショナルに格納すると、リレーショナル列に索引を定義できるため、問合せのパフォーマンスが向上します。

  オブジェクト・リレーショナル形式での格納を指定する場合、XMLSchema_spec句も指定する必要があります。

ALL VARRAYS AS句を使用すると、XMLType列にすべてのVARRAYを格納できます。

以前のリリースでは、XMLTypeデータはデフォルトでBasicFiles LOBのCLOB列に格納されます。Oracle Database 11gリリース2 (11.2.0.2)以降では、COMPATIBLE初期化パラメータが11.2以上の場合にXMLType_storage句を指定しないと、XMLTypeデータはSecureFiles LOBのバイナリXML列に格納されます。SecureFiles LOB記憶域を使用できない場合は、BasicFiles LOBのバイナリXML列に格納されます。

関連項目:

SecureFiles LOBの詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。

XMLSchema_spec

この句のセマンティクスの詳細は、「XMLSchema_spec」を参照してください。

関連項目:

- LOB_segnameおよびLOB_parameters句の詳細は、「LOB_storage_clause」を参照してください
- オブジェクト・リレーショナル表のXMLType列の例は、「XMLType列の例」を参照し、XMLスキーマの作成例は、

「SQL文でのXMLの使用方法」を参照してください。

- XMLType列と表およびXMLスキーマの作成の詳細は、『Oracle XML DB開発者ガイド』を参照してください。
- DBMS_XMLSCHEMAパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

XMLType_virtual_columns

この句は、バイナリXML記憶域のXMLType表に対してのみ有効です。この記憶域は、XMLType_storage句で指定します。VIRTUAL COLUMNS句を指定して、仮想列を定義します。仮想列は、ファンクション索引または制約の定義で使用できます。そのような仮想列には表の作成時には制約を定義できませんが、その後、ALTER TABLE文を使用して制約を列に追加できます。

関連項目:

XML環境でこの句を使用する方法の例は、『Oracle XML DB開発者ガイド』を参照してください。

read_only_clause

この句を使用すると、表、パーティションまたはサブパーティションを読取り専用または読取り/書込みのどちらのモードで作成するかを指定できます。

- 読取り専用モードを指定するには、READ ONLYを使用します。オブジェクトが読取り専用モードの場合は、オブジェクトまたはオブジェクトのSELECT ... FOR UPDATE ...文に影響するDML文は発行できません。表データを変更しないDDL文は発行できます。読取り専用オブジェクトで許可される操作および許可されない操作の完全なリストは、『Oracle Database管理者ガイド』を参照してください。
- 読取り/書込みモードを指定するには、READ WRITEを使用します。これはデフォルトです。

パーティション表に対してこの句を指定する場合は、表のデフォルトの読取り専用モードまたは読取り/書込みモードを指定します。パーティション・レベルでモードを指定してこの動作を上書きする場合を除いて、このモードは、作成時に表のすべてのパーティションと、後で表に追加されるすべてのパーティションに割り当てられます。

コンポジット・パーティション表に対してこの句を指定する場合は、表のすべてのパーティションにデフォルトの読取り専用モードまたは読取り/書込みモードを指定します。この動作を上書きするには、特定のパーティションに対してこの句を指定します。サブパーティション・レベルでモードを指定してこの動作を上書きする場合を除いて、パーティションのデフォルトのモードは、作成時にパーティションのすべてのサブパーティションと、後でパーティションに追加されるすべてのサブパーティションに割り当てられます。

indexing_clause

indexing_clauseは、パーティション表に対してのみ有効です。この句を使用すると、表、表パーティション、または表サブパーティションに対して索引付けプロパティを設定できます。

- INDEXING ONを指定すると、索引付けプロパティをONに設定できます。これはデフォルトです。
- INDEXING OFFを指定すると、索引付けプロパティをOFFに設定できます。

この索引付けプロパティにより、表のパーティションとサブパーティションが、表の部分索引に含まれるかどうかが決まります。

- 単純なパーティション表の場合、索引付けプロパティがONのパーティションは、その表の部分索引に含まれます。索引付けプロパティがOFFのパーティションは除外されます。
- コンポジット・パーティション表の場合、索引付けプロパティがONのサブパーティションは、その表の部分索引に含まれま

す。索引付けプロパティがOFFのサブパーティションは除外されます。

indexing_clauseは、表レベル、パーティション・レベル、またはサブパーティション・レベルで指定できます。indexing_clauseを表レベルでtable_properties句内に指定すると、その表のデフォルトの索引付けプロパティを設定することになります。データベースが自動的に作成する時間隔パーティションは、常に表のデフォルトの索引付けプロパティを継承します。その他のタイプのパーティションおよびサブパーティションは、次のようにデフォルトの索引付けプロパティを継承します。

- 単純なパーティション表の場合、パーティションはその表のデフォルトの索引付けプロパティを継承します。この動作は、次のように、パーティションごとにindexing_clauseを指定することで上書きできます。
  - レンジ・パーティションの場合は、range_partitions句のtable_partition_descriptionで指定します。
  - ハッシュ・パーティションの場合は、hash_partitions句のindividual_hash_partitions句で指定します。
  - リスト・パーティションの場合は、list_partitions句のtable_partition_descriptionで指定します。
  - 参照パーティションの場合は、reference_partitioning句のreference_partition_desc句のtable_partition_descriptionで指定します。
  - システム・パーティションの場合は、system_partitioning句のreference_partition_desc句のtable_partition_descriptionで指定します。
- コンポジット・パーティション表の場合、サブパーティションはその表のデフォルトの索引付けプロパティを継承します。この動作は、パーティションまたはサブパーティションごとにindexing_clauseを指定することで上書きできます。

  次のように、パーティションにindexing_clauseを指定すると、サブパーティションはそのパーティションの索引付けプロパティを継承します。
  - コンポジット・レンジ・パーティションの場合は、composite_range_partitions句のtable_partition_descriptionで指定します。
  - コンポジット・リスト・パーティションの場合は、composite_list_partitions句のtable_partition_descriptionで指定します。
  - コンポジット・ハッシュ・パーティションの場合は、composite_hash_partitions句のindividual_hash_partitions句で指定します。

  サブパーティションの索引付けプロパティは、次のように、そのサブパーティションにindexing_clauseを指定することで設定できます。
  - レンジ・サブパーティションの場合は、composite_range_partitions句のrange_subpartition_desc句で指定します。
  - リスト・サブパーティションの場合は、composite_list_partitions句のlist_subpartition_desc句で指定します。
  - ハッシュ・サブパーティションの場合は、composite_hash_partitions句のindividual_hash_subparts句で指定します。

関連項目:

表、表パーティション、または表サブパーティションの索引付けプロパティを確認する方法の詳細は、『Oracle Databaseリファレンス』を参照してください。

- 表のデフォルトの索引付けプロパティを確認するには、*_PART_TABLESビューのDEF_INDEXING列を問い合せます。
- 表パーティションの索引付けプロパティを確認するには、*_TAB_PARTITIONSビューのINDEXING列を問い合せます。
- 表サブパーティションの索引付けプロパティを確認するには、*_TAB_SUBPARTITIONSビューのINDEXING列を問い合せます。

indexing_clauseの制限事項

indexing_clauseには、次の制限事項があります。

- indexing_clauseは、非パーティション表には指定できません。
- indexing_clauseは、索引構成表には指定できません。

関連項目:

部分索引の詳細は、CREATE INDEXの「partial_index_clause」を参照してください。

table_partitioning_clauses

table_partitioning_clausesを使用すると、パーティション表を作成できます。

一般的なパーティション化のノート

すべてのタイプのパーティション化には、次のノートがあります。

- 指定できるパーティションとサブパーティションの合計は1024K-1です。
- パーティションが1つのみのパーティション表も作成できます。パーティションが1つの表は、非パーティション表とは異なります。たとえば、非パーティション表にはパーティションを追加できません。
- すべての表とLOBパーティションおよびすべての表とLOBサブパーティションに名前を指定できますが、必須ではありません。名前を指定する場合は、スキーマ・オブジェクトのネーミング規則および「データベース・オブジェクトのネーミング規則」にある該当部分の記述に従って指定する必要があります。名前を省略すると、次のように名前が生成されます。
    - パーティション名を省略すると、SYS_Pnの形式で名前が生成されます。LOBデータおよびLOB索引パーティションに対するシステム生成名は、それぞれSYS_LOB_PnおよびSYS_IL_Pnの形式をとります。
    - subpartition_templateでサブパーティション名を指定すると、そのテンプレートで作成される各サブパーティションに対して、パーティション名とテンプレートのサブパーティション名を連結して名前が生成されます。LOBサブパーティションの場合、生成されるLOBサブパーティション名は、パーティション名とテンプレートのLOBセグメント名の連結です。COMPATIBLE初期化パラメータが12.2以上に設定されている場合、連結の最大長は128バイトです。それ以外の場合、最大長は30バイトです。連結が最大長を超える場合、エラーが戻されて文は失敗します。
    - 個々のサブパーティションを指定するときにサブパーティション名を指定せず、subpartition_templateを指定していない場合、SYS_SUBPnという形式で名前が生成されます。LOBデータおよび索引サブパーティションに対する、対応するシステム生成名は、それぞれSYS_LOB_SUBPnおよびSYS_IL_SUBPnです。

- 表領域の記憶域は、CREATE TABLE文で、表セグメントとLOBセグメントの両方に対して様々なレベルで指定できます。表領域数は、パーティション数またはサブパーティション数と同じである必要はありません。パーティション数またはサブパーティション数が表領域数より多い場合は、表領域名が繰り返し使用されます。

  データベースでは、表領域の記憶域を、次の順序で(優先度の高いものから順に)評価します。

  - 個々の表サブパーティションまたはLOBサブパーティション・レベルで指定された表領域の記憶域が、最も優先度が高くなります。次に、subpartition_templateでパーティションまたはLOBに対して指定された記憶域です。
  - 個々の表パーティションまたはLOBパーティション・レベルで指定された表領域の記憶域。ここで指定された記憶域パラメータは、subpartition_templateよりも優先します。
  - 表に対して指定された表領域の記憶域
  - ユーザーに対して指定されたデフォルトの表領域の記憶域

- デフォルトでは、ネストした表はパーティション実表でパーティション化されます。

一般的なパーティション化の制限事項

すべてのパーティション化には、次の制限事項があります。

- クラスタの一部である表は、パーティション化できません。
- ネストした表をパーティション化したり、索引構成表として定義されているVARRAYをパーティション化することはできません。
- LONGまたはLONG RAW列を含む表は、パーティション化できません。

ハイブリッド・パーティション表の制限

ハイブリッド・パーティション表には、次の制限事項があります。

- 特に明記しないかぎり、外部表に適用される制限は、ハイブリッド・パーティション表にも適用されます
- REFERENCEおよびSYSTEMパーティション化メソッドはサポートされていません
- 単一レベルのLISTおよびRANGEパーティション化のみがサポートされます。
- 一意索引やグローバル一意索引は存在しません。部分索引のみが許可され、一意索引は部分索引にできません。
- HIVEについては、単一レベルのリスト・パーティション化のみがサポートされます。
- 属性クラスタリング(CLUSTERING句)は使用できません。
- DML操作は、ハイブリッド・パーティション表の内部パーティションに対してのみ実行できます(外部パーティションは読取り専用パーティションとして処理されます)
- 表レベルで定義されたインメモリーは、ハイブリッド・パーティション表の内部パーティションにのみ影響します。
- 列のデフォルト値は許可されていません。
- 非表示の列は使用できません。
- CELLMEMORY句は使用できません。
- SPLIT、MERGEおよびMOVEメンテナンス操作は、外部パーティションに対しては使用できません。

ブロック・サイズが異なる表領域のパーティション化されたデータベース・エンティティの記憶域には、制限事項があります。これらの

制限事項については、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

関連項目:

「パーティション化の例」

range_partitions

range_partitions句を使用すると、列リストの値の範囲で表をパーティション化できます。索引構成表の場合、列リストは表の主キー列のサブセットである必要があります。

レンジ・パーティション化の制限事項

レンジ・パーティション化には、「一般的なパーティション化の制限事項」に示されている制限事項があります。次の追加の制限が適用されます。

- 指定できるパーティション化キー列は16以下です。
- パーティション化キー列は、CHAR型、NCHAR型、VARCHAR2型、NVARCHAR2型、VARCHAR型、NUMBER型、FLOAT型、DATE型、TIMESTAMP型、TIMESTAMP WITH LOCAL TIMEZONE型またはRAW型である必要があります。
- XMLType表またはXMLType列を含む表に属しているか、あるいはシャーディング・キー列として使用される、文字列データ型の各レンジ・パーティション化キー列には、宣言された照合(BINARY、USING_NLS_COMP、USING_NLS_SORT、USING_NLS_SORT_CSのいずれか)が必要です。これらすべての照合では、パーティション・バウンドは、照合BINARYを使用してチェックされます。
- VALUES句にNULLを指定することはできません。

column

行がどのパーティションに属するかを判断するために使用される、列の順序リストを指定します。これらの列は、パーティション化キーです。仮想列とINVISIBLEの列は、パーティション化キー列として指定できます。

INTERVAL句

この句を使用すると、表の時間隔パーティションを設定できます。時間隔パーティションは、数値範囲または日時期間に基づくパーティションです。表に挿入されたデータがすべてのレンジ・パーティションを超える場合に、指定されたレンジまたは期間のパーティションを自動的に作成することをデータベースに指定することによって、レンジ・パーティション化を拡張します。自動的に作成された各パーティションには、SYS_Pnの形式で名前が生成されます。自動的に生成されたパーティション名は一意で、ネームスペース規則に違反しないことが保証されます。

- exprには、有効な数値または期間式を指定します。
- オプションのSTORE IN句を使用して、時間隔パーティション・データが格納される1つ以上の表領域を指定できます。
- また、range_partitionsのPARTITION句を使用して、1つ以上のレンジ・パーティションを指定する必要があります。レンジ・パーティション・キーの値によって、遷移ポイントと呼ばれるレンジ・パーティションの上限が決まります。その遷移ポイントを超えるデータに対して時間隔パーティションが作成されます。

時間隔パーティション化の制限事項

INTERVAL句には、「一般的なパーティション化の制限事項」および「レンジ・パーティション化の制限事項」に示されている制限事項があります。次の追加の制限が適用されます。

- パーティション化キー列は1つのみ指定でき、NUMBER型、DATE型、FLOAT型またはTIMESTAMP型である必要があります。
- この句は索引構成表ではサポートされていません。
- この句は、VARRAY列を含む表ではサポートされていません。
- 同一レベル・パーティション化されているネストした表を含む時間隔パーティション表は、作成できません。ネストした表またはXMLオブジェクト・リレーショナル・データ型を使用して時間隔パーティション表を作成すると、ネストした表は非パーティション表として作成されます。
- この句は、XMLデータがバイナリXMLとして格納される場合にのみ、XMLType列を含む表に対してサポートされます。
- 時間隔パーティションは、サブパーティション・レベルではサポートされていません。
- シリアル化可能トランザクションは、時間隔パーティションでは動作しません。セグメントが存在しない時間隔パーティション表のパーティションにデータを挿入しようとすると、エラーになります。
- VALUES句の場合:
    - MAXVALUEは指定できません(無限)。指定すると、必要に応じてパーティションを自動追加するという目的に反するためです。
    - パーティション化キー列にはNULL値を指定できません。

関連項目:

時間隔パーティションの詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

PARTITION partition

パーティション名を指定する際、partitionの名前は、スキーマ・オブジェクトのネーミング規則および「データベース・オブジェクトのネーミング規則」にある該当部分の記述に従う必要があります。partitionを省略すると、「一般的なパーティション化のノート」で説明されているように名前が生成されます。

range_values_clause

現行パーティションの上限(境界は含まない)を指定します。値リストは、range_partitions句の列リストに対応するリテラル値を含む順序リストです。値リスト内のリテラルのかわりに、キーワードMAXVALUEを使用できます。MAXVALUEには、常に他の値(NULLを含む)より高位にソートされる最大値を指定します。

パーティション境界の上限にMAXVALUE以外の値を指定した場合、表に暗黙の整合性制約が課せられます。

ノート:

表が DATE 列でパーティション化されている場合および日付書式で年の最初の 2 桁の数字が指定されていない場合、年の「YYYY」4 文字書式マスクで TO_DATE ファンクションを使用する必要があります。この句では、「RRRR」書式マスクはサポートしていません。日付書式は、NLS_TERRITORY によって暗黙的に決定され、NLS_DATE_FORMAT によって明示的に決定されます。これらの初期化パラメータの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

関連項目:

パーティション・バウンドの詳細は、『Oracle Database概要』を参照してください。また、「レンジ・パーティション化の例」を参照してください。

table_partition_description

`table_partition_description`を使用すると、表の物理特性および記憶特性を定義できます。

`deferred_segment_creation`、`segment_attributes_clause`、`table_compression`、`inmemory_clause`および`ilm_clause`句の機能は、表全体のphysical_propertiesと同じです。

`indexing_clause`を使用すると、レンジ、リスト、システムまたは参照の表パーティションに索引付けプロパティを設定できます。詳細は、「indexing_clause」を参照してください。

`prefix_compression`句および`OVERFLOW`句の機能は、index_org_table_clauseと同じです。

LOB_storage_clause

`LOB_storage_clause`を使用すると、このパーティションまたはこのパーティションの任意のレンジまたはリスト・サブパーティション内にある1つ以上のLOB項目に対してLOB記憶特性を指定できます。LOB項目に`LOB_storage_clause`を指定しない場合、「一般的なパーティション化のノート」で説明されているように、各LOBデータ・パーティションに対する名前が生成されます。

varray_col_properties

`varray_col_properties`を使用すると、このパーティション、またはこのパーティションの任意のレンジまたはリスト・サブパーティション内にある1つ以上のVARRAY項目に対して記憶特性を指定できます。

nested_table_col_properties

`nested_table_col_properties`を使用すると、このパーティション、またはこのパーティションの任意のレンジまたはリスト・サブパーティション内にある1つ以上のネストした表の記憶域表項目に記憶特性を指定できます。この句に指定した記憶特性は、表レベルで指定したすべての記憶域属性を上書きします。

partitioning_storage_clause

`partitioning_storage_clause`を使用すると、ハッシュ・パーティションおよびレンジ、ハッシュおよびリスト・サブパーティションの記憶特性を指定できます。

partitioning_storage_clauseの制限事項

この句には、次の制限事項があります。

- TABLESPACE SET句は、CREATE TABLEのSHARDEDキーワードを指定してシャード表を作成する場合にのみ有効です。この句を使用して、表パーティション・データが格納される表領域セットを指定します。
- `OVERFLOW`句は、索引構成パーティション表にのみ関連し、`individual_hash_partitions`句でのみ有効です。レンジ・パーティションやハッシュ・パーティションまたは任意のタイプのサブパーティションには有効ではありません。
- `index_compression`句の`advanced_index_compression`句は指定できません。
- `indexing_clause`の`prefix_compression`句は索引構成表のパーティションにのみ指定できます。`COMPRESS`または`NOCOMPRESS`を指定できますが、`integer`で接頭辞の長さを指定できません。

list_partitions

`list_partitions`句を使用すると、`column`リスト内の列ごとにリテラル値のリストで表をパーティション化できます。リスト・

パーティション化は、個々の行が固有のパーティションにマップする方法に関する制御に便利です。

リスト・パーティション化の制限事項

リスト・パーティション化には、「一般的なパーティション化の制限事項」に示されている制限事項があります。次の追加の制限が適用されます。

- 指定できるパーティション化キー列は16以下です。
- 索引構成表をパーティション化する場合は、複数のパーティション化キー列を指定できません。
- パーティション化キー列は、CHAR型、NCHAR型、VARCHAR2型、NVARCHAR2型、VARCHAR型、NUMBER型、FLOAT型、DATE型、TIMESTAMP型、TIMESTAMP WITH LOCAL TIMEZONE型またはRAW型である必要があります。
- XMLType表またはXMLType列を含む表に属しているか、あるいはシャーディング・キー列として使用される、文字列データ型の各リスト・パーティション化キー列には、宣言されたBINARY照合(BINARY 、USING_NLS_COMP、USING_NLS_SORT、USING_NLS_SORT_CSのいずれか)が必要です。これらすべての照合では、パーティションの照合には照合BINARYが使用されます。

AUTOMATIC

AUTOMATICを指定して、自動リスト・パーティション表を作成します。このタイプの表により、データベースはオンデマンドで追加のパーティションを作成できます。

自動リスト・パーティション表を作成する場合は、通常のリスト・パーティション表を作成するときと同じようにパーティションおよびパーティション化キーの値を指定します。ただし、DEFAULTパーティションは指定しません。データが表にロードされると、ロードされたパーティション化キー値が既存のどのパーティションにも対応していない場合は、自動的に新しいパーティションが作成されます。単一のパーティション化キー値でリスト・パーティション化が定義されている場合は、新しいパーティション化キー値ごとに新しいパーティションが作成されます。複数のパーティション化キー列でリスト・パーティション化が定義されている場合は、新しいパーティション化キー値のセットおよび一意のパーティション化キー値のセットごとに新しいパーティションが作成されます。自動的に作成された各パーティションには、SYS_Pnの形式で名前が生成されます。自動的に生成されたパーティション名は一意で、ネームスペース規則に違反しないことが保証されます。

AUTOMATICキーワードは、リスト・パーティション表、およびリスト-レンジ、リスト-リスト、リスト-ハッシュおよびリスト-時間隔コンポジット・パーティション表に指定できます。コンポジット・パーティション表の場合、自動的に作成された各リスト・パーティションには、1つのサブパーティションが含まれます(表にサブパーティション・テンプレートが定義されていない場合)。

自動リスト・パーティション表にローカル・パーティション索引が定義されている場合、対応する表パーティションが作成されると、ローカル索引パーティションが作成されます。

自動リスト・パーティション化の制限事項

自動リスト・パーティション化には、「リスト・パーティション化の制限事項」に示されている制限事項があります。次の追加の制限が適用されます。

- 自動リスト・パーティション表の作成時には、少なくとも1つのパーティションが必要です。新規および未知のパーティション化キー値に対して新しいパーティションが自動的に作成されるため、自動リスト・パーティション表にDEFAULTパーティションを含めることはできません。
- 自動リスト・パーティション化は、索引構成表または外部表ではサポートされていません。
- 自動リストパーティション化は、VARRAY列を含む表ではサポートされていません。
- ローカル・ドメイン索引は、自動リスト・パーティション表には作成できません。グローバル・ドメイン索引は、自動リスト・

パーティション表に作成できます。

- 自動リスト・パーティション表は、参照パーティション化の子表または親表にすることはできません。
- 自動リスト・パーティション化は、サブパーティション・レベルではサポートされていません。

STORE IN

オプションのSTORE IN句を使用すると、自動的に作成されたリスト・パーティションのデータが格納される1つ以上の表領域を指定できます。

ノート:

自動リスト・パーティション表は、通常のリスト・パーティション表に変更でき、その逆の変更も可能です。自動的に作成されたリスト・パーティションのデータが格納される表領域を変更することもできます。詳細は、「ALTER TABLE」の「alter_automatic_partitioning」句を参照してください。

list_values_clause

各パーティションのlist_values_clauseでは、1つ以上の値を割り当てる必要があります。1つのキー列で表がパーティション化されている場合は、list_values構文の上位ブランチを使用して、その列の値リストを指定します。この場合、複数のパーティションに値(NULLを含む)を割り当てることはできません。複数のキー列で表がパーティション化されている場合は、list_values構文の下位ブランチを使用して、値リストを指定します。各値リストはカッコで囲まれ、キー列の値のリストを表します。この場合、個々のキー列値を複数のパーティションに割り当てることができます。ただし、複数のパーティションに完全な値リストを割り当てることはできません。リスト・パーティションは、順序付けされていません。

VALUES句のパーティション値にリテラルNULLを指定した場合、後続の問合せで、そのパーティション内のデータにアクセスするには、WHERE句で、比較条件ではなくIS NULL条件を使用する必要があります。

DEFAULTキーワードを指定すると、行の挿入先となるパーティションが作成されます。この行は、別のパーティションにはマップされません。このため、DEFAULTを指定できるのは1つのパーティションのみです。そのパーティションに対して、その他の値を指定することはできません。また、デフォルト・パーティションは、パーティションの中で最後に定義する必要があります。DEFAULTは、レンジ・パーティションでMAXVALUEを使用する場合と同様に使用します。

各パーティションの値のリストを構成する文字列は、最大4KBです。すべてのパーティションの値の総数を、64K-1以下に指定します。

リスト・パーティションのパーティション化キー列は、最大サイズが32,767バイトの拡張データ型列になることがあります。この場合、パーティションに指定する値のリストは、4Kバイトの制限を超過することがあります。この制限を回避するには、次のいずれかの方法を使用します。

- 4Kバイトの制限を超過する値に、DEFAULTパーティションを使用します。
- パーティション・キー列でハッシュ・ファンクション(STANDARD_HASHなど)を使用して、4Kバイトよりも短い一意の識別子を作成します。詳細は、「STANDARD_HASH」を参照してください。

list_values_clauseの制限事項

DEFAULTパーティションは、自動リスト・パーティション表に指定できません。

関連項目:

拡張データ型の詳細は、「拡張データ型」を参照してください。

table_partition_description

table_partition_descriptionの副次句の動作は、「table_partition_description」のレンジ・パーティションで説明した動作と同じです。

hash_partitions

hash_partitions句を使用すると、表がハッシュ方式でパーティション化されるように指定できます。列の値にパーティション化キーとして指定されたハッシュ・ファンクションを使用して、行がパーティションに割り当てられます。個々のハッシュ・パーティションを指定するか、または作成されるハッシュ・パーティションの数を指定できます。

ハッシュ・パーティション化の制限事項

ハッシュ・パーティション化には、「一般的なパーティション化の制限事項」に示されている制限事項があります。次の追加の制限が適用されます。

- 指定できるパーティション化キー列は16以下です。
- パーティション化キー列は、CHAR型、NCHAR型、VARCHAR2型、NVARCHAR2型、VARCHAR型、NUMBER型、FLOAT型、DATE型、TIMESTAMP型、TIMESTAMP WITH LOCAL TIMEZONE型またはRAW型である必要があります。
- XMLType表またはXMLType列を含む表に属しているか、あるいはシャーディング・キー列として使用される、文字データ型の各ハッシュ・パーティション化キー列には、宣言された照合(BINARY、USING_NLS_COMP、USING_NLS_SORT、USING_NLS_SORT_CSのいずれか)が必要です。

column

行がどのパーティションに属するかを判断するために使用される、列の順序リストを指定します(パーティション化キー)。

individual_hash_partitions

この句を使用して、個々のパーティションを名前で指定します。

indexing_clauseを使用すると、ハッシュ・パーティションに索引付けプロパティを設定できます。詳細は、「indexing_clause」を参照してください。

個々のハッシュ・パーティションを指定する場合の制限事項

partitioning_storage_clauseで指定できる句は、TABLESPACE句および表の圧縮のみです。

> ノート:
>
> 異なる文字セットを使用してデータベースを使用しているか、使用する予定がある場合は、文字列を分割する際に注意してください。文字のソート順序は、すべての文字セットで同一ではありません。文字セット・サポートの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

hash_partitions_by_quantity

個々のパーティションを定義するかわりに、ハッシュ・パーティションの数を指定します。この場合、SYS_Pnの形式でパーティション

名が割り当てられます。STORE IN句を使用すると、ハッシュ・パーティション・データが格納される1つ以上の表領域を指定できます。表領域の数とパーティションの数が同じである必要はありません。パーティション数が表領域数より多い場合は、表領域名が繰り返し使用されます。

ハッシュ・パーティション化の両方の方法でロード・バランシングを最適化するには、2の累乗のパーティション数を指定します。個々のハッシュ・パーティションを指定する際は、partitioning_storage_clauseにTABLESPACEと表の圧縮の両方を指定できます。ハッシュ・パーティションを数で指定する場合は、TABLESPACEのみを指定できます。ハッシュ・パーティションは、その他のすべての属性を表レベルのデフォルトから継承します。

table_compression句の機能は、表の「table_properties」について説明されている機能と、ほぼ同じです。

prefix_compression句およびOVERFLOW句の機能は、index_org_table_clauseと同じです。

表レベルで指定された表領域の記憶域は、パーティション・レベルで指定された表領域の記憶域で上書きされ、パーティション・レベルで指定された表領域の記憶域は、サブパーティション・レベルで指定された表領域の記憶域で上書きされます。

individual_hash_partitions句に含まれるpartitioning_storage_clauseのTABLESPACE句は、作成される個々のパーティションのみについて、表領域の記憶域を決定します。hash_partitions_by_quantity句では、STORE IN句によって、表の作成時のパーティションの位置と、後から追加されるパーティションのデフォルトの格納場所が決定されます。

ハッシュ・パーティションを数量で指定する場合の制限事項

index_compression句のadvanced_index_compression句は指定できません。

関連項目:

ハッシュ・パーティション化については、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

composite_range_partitions

composite_range_partitions句を使用すると、まず、表をレンジ・パーティション化し、次にそれらのパーティションをレンジ・サブパーティション化、ハッシュ・サブパーティション化またはリスト・サブパーティション化できます。

コンポジット・レンジ・パーティション化でのINTERVAL句のセマンティクスは、レンジ・パーティション化の場合と同じです。詳細は、「INTERVAL句」を参照してください。

subpartition_by_range、subpartition_by_hashまたはsubpartition_by_listでは、各コンポジット・レンジ・パーティションのサブパーティション化のタイプを指定できます。これらの句では、サブパーティション・テンプレートを指定できます。サブパーティション・テンプレートによって、この文の一部として作成されるサブパーティションまたは後で作成されるサブパーティションのデフォルトのサブパーティション特性が設定されます。

表のサブパーティション化のタイプ、およびオプションでサブパーティション・テンプレートを設定した後、1つ以上のレンジ・パーティションを定義する必要があります。

- 非コンポジット・レンジ・パーティションと同じ要件を持つrange_values_clauseを指定する必要があります。
- table_partition_descriptionを使用すると、各パーティションの物理特性および記憶特性を定義できます。
- range_partition_descで、range_subpartition_desc、list_subpartition_desc、individual_hash_subpartsまたはhash_subparts_by_quantityを使用して、パーティションの個々のサブパーティションの特性を指定します。これらの句で指定する値は、これらのサブパーティションについて、subpartition_templateで指定した値にかわるものです。

- ハッシュ・サブパーティション、リスト・サブパーティションまたはLOBサブパーティションに指定できる特性は、TABLESPACEおよびtable_compressionのみです。

コンポジット・レンジ・パーティション化の制限事項

サブパーティション化のタイプにかかわらず、コンポジット・レンジ・パーティションには次の制限事項があります。

- サブパーティション・レベルで指定できる物理属性は、TABLESPACEおよび表の圧縮のみです。
- コンポジット・パーティション化は、索引構成表に対して指定できません。そのため、コンポジット・パーティション表に対して、table_partition_descriptionのOVERFLOW句は無効です。
- XMLType列を含む表に対してはコンポジット・パーティション化を指定できません。
- XMLType表またはXMLType列を含む表に属している、文字データ型の各レンジ・パーティション・キー列、リスト・パーティション・キー列またはハッシュ・パーティション・キー列には、宣言された照合(BINARY、USING_NLS_COMP、USING_NLS_SORT、USING_NLS_SORT_CSのいずれか)が必要です。

関連項目:

コンポジット・レンジ・パーティション化の例は、「コンポジット・パーティション表の例」を参照し、コンポジット・リスト・パーティション化の例は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

composite_list_partitions

composite_list_partitions句を使用すると、まず、表をリスト・パーティション化し、次にそれらのパーティションをレンジ・サブパーティション化、ハッシュ・サブパーティション化またはリスト・サブパーティション化できます。

subpartition_by_range、subpartition_by_hashまたはsubpartition_by_listでは、各コンポジット・リスト・パーティションのサブパーティション化のタイプを指定できます。これらの句では、サブパーティション・テンプレートを指定できます。サブパーティション・テンプレートによって、この文の一部として作成されるサブパーティションおよび後で作成されるサブパーティションのデフォルトのサブパーティション特性が設定されます。

各コンポジット・パーティションのサブパーティション化のタイプを設定し、オプションでサブパーティション・テンプレートを定義した後、1つ以上のリスト・パーティションを定義する必要があります。

- list_partition_descで、非コンポジット・リスト・パーティションと同じ要件を持つlist_values_clauseを指定する必要があります。
- table_partition_descriptionを使用すると、各パーティションの物理特性および記憶特性を定義できます。
- list_partition_descで、range_subpartition_desc、list_subpartition_desc、individual_hash_subpartsまたはhash_subparts_by_quantityを使用して、パーティションの個々のサブパーティションの特性を指定します。これらの句で指定する値は、これらのサブパーティションについて、subpartition_templateで指定した値にかわるものです。

AUTOMATICを指定して、自動リスト-レンジ、リスト-リスト、リスト-ハッシュまたはリスト-時間隔コンポジット・パーティション表を作成します。このタイプの表により、データベースはオンデマンドで追加のパーティションを作成できます。オプションのSTORE IN句を使用すると、自動的に作成されたパーティションのデータが格納される1つ以上の表領域を指定できます。AUTOMATIC句およびSTORE IN句のセマンティクスは、非コンポジット・リスト・パーティションと同じです。これらの句のセマンティクスの詳細は、「list_partitions」の項目の「AUTOMATIC」および「STORE IN」を参照してください。自動コンポジット・パーティション表には、「コンポジット・リスト・パーティション化の制限事項」および「自動リスト・パーティション化の制限事項」に示されている制限事項があります。

コンポジット・リスト・パーティション化の制限事項

コンポジット・リスト・パーティション化には、「コンポジット・レンジ・パーティション化の制限事項」に示されているものと同じ制限事項があります。

composite_hash_partitions

composite_hash_partitions句を使用すると、まず、ハッシュ方式で表をパーティション化し、次にそれらのパーティションをレンジ・サブパーティション化、ハッシュ・サブパーティション化またはリスト・サブパーティション化できます。

subpartition_by_range、subpartition_by_hashまたはsubpartition_by_listでは、各コンポジット・レンジ・パーティションのサブパーティション化のタイプを指定できます。これらの句では、サブパーティション・テンプレートを指定できます。サブパーティション・テンプレートによって、この文の一部として作成されるサブパーティションまたは後で作成されるサブパーティションのデフォルトのサブパーティション特性が設定されます。

表のサブパーティション化のタイプを設定した後、individual_hash_partitionsまたはhash_partitions_by_quantityを指定する必要があります。

コンポジット・ハッシュ・パーティション化の制限事項

コンポジット・ハッシュ・パーティション化には、「コンポジット・レンジ・パーティション化の制限事項」に示されているものと同じ制限事項があります。

subpartition_template

subpartition_templateは、レンジ・サブパーティション化、リスト・サブパーティション化およびハッシュ・サブパーティション化のオプション要素です。このテンプレートを使用することで、表の各パーティションにデフォルトのサブパーティションを定義できます。明示的にサブパーティションを定義していないパーティションには、このデフォルト・サブパーティション特性が作成されます。この句は、対称型パーティションの作成時に有効です。パーティション・レベルでサブパーティションを明示的に定義すると(range_subpartition_desc、list_subpartition_desc、individual_hash_subpartsまたはhash_subparts_by_quantity句で指定)、この句を上書きできます。

サブパーティションとテンプレートを定義する場合、レンジ・サブパーティション、リスト・サブパーティションまたはハッシュ・サブパーティションを明示的に定義するか、ハッシュ・サブパーティション数量を定義できます。

- サブパーティションを明示的に定義するには、range_subpartition_desc、list_subpartition_descまたはindividual_hash_subpartsを使用します。各パーティションに名前を指定する必要があります。partitioning_storage_clauseのLOB_partitioning_clauseを指定する場合、LOB_segnameを指定する必要があります。
- ハッシュ・サブパーティション数量を指定するには、hash_subpartition_quantityに正の整数を指定します。その数のサブパーティションが各パーティションに作成され、SYS_SUBPnという形式のサブパーティション名が割り当てられます。

ノート:

サブパーティションのテンプレートに対して表領域の記憶域を指定しても、table のパーティションに対して明示的に指定した表領域の記憶域は上書きされません。サブパーティションに対して表領域の記憶域を指定するには、次のいずれかの操作を実行します。

- パーティション・レベルでの表領域の記憶域を省略し、サブパーティションのテンプレートに対して表領域の記憶域を指定します。
- 固有の表領域の記憶域を持つサブパーティションを個別に定義します。

サブパーティション・テンプレートの制限事項

サブパーティション・テンプレートには、次の制限事項があります。

- 1つのLOBサブパーティションに対してTABLESPACEを指定した場合、次にそのLOB列のすべてのLOBサブパーティションに対してTABLESPACEを指定する必要があります。複数のLOBサブパーティションに対して同じ表領域を指定できます。
- `subpartition_template`内で、または個々のサブパーティションを定義するときに、`partitioning_storage_clause`を使用してリスト・サブパーティションに対して異なるLOB記憶域を指定した場合、LOBとVARRAY列の両方に`LOB_segname`を指定する必要があります。

subpartition_by_range

`subpartition_by_range`句を使用すると、表の各パーティションがレンジ・サブパーティション化されるように指定できます。列リストのサブパーティション化はパーティション化キーには関連しませんが、同じ制限事項が適用されます(「column」を参照)。

`subpartition_template`を使用して、デフォルトのサブパーティション特性値を指定できます。「subpartition_template」を参照してください。データベースは、特性を明示的に指定していないこのパーティションの任意のサブパーティションに対して、これらの値を使用します。

`range_partition_desc`または`list_partition_desc`の`range_subpartition_desc`を使用して、各パーティションのレンジ・サブパーティションを個別に定義することもできます。`subpartition_template`と`range_subpartition_desc`の両方を指定しない場合、単一のMAXVALUEサブパーティションが作成されます。

subpartition_by_list

`subpartition_by_list`句を使用すると、`column`リストからリテラル値のリストの表に各パーティションをサブパーティション化する必要があることを指定できます。最大16のリスト・サブパーティション化キー列を指定できます。

`subpartition_template`を使用して、デフォルトのサブパーティション特性値を指定できます。「subpartition_template」を参照してください。データベースは、特性を明示的に指定していないこのパーティションの任意のサブパーティションに対して、これらの値を使用します。

`range_partition_desc`または`list_partition_desc`の`list_subpartition_desc`を使用して、各パーティションのリスト・サブパーティションを個別に定義することもできます。`subpartition_template`と`list_subpartition_desc`の両方を指定しない場合、単一のDEFAULTサブパーティションが作成されます。

リスト・サブパーティション化の制限事項

リスト・サブパーティション化には、「コンポジット・レンジ・パーティション化の制限事項」に示されているものと同じ制限事項があります。

subpartition_by_hash

`subpartition_by_hash`句を使用すると、表の各パーティションがハッシュ・サブパーティション化されるように指定できます。列リストのサブパーティション化はパーティション化キーには関連しませんが、同じ制限事項が適用されます(「column」を参

照)。

subpartition_template句またはSUBPARTITIONS integer句を使用すると、サブパーティションを定義できます。「subpartition_template」を参照してください。どちらの場合も、ロード・バランシングを最適化するには、2の累乗のパーティション数を指定する必要があります。

SUBPARTITIONS integerを指定する場合、表の各パーティションにおけるデフォルトのサブパーティション数を設定します。また、サブパーティションが格納される1つ以上の表領域を指定することもできます。デフォルト値は1です。この句とsubpartition_templateの両方を指定しない場合、1つのハッシュ・サブパーティションを持つパーティションが作成されます。

コンポジット・パーティションのノート

コンポジット・パーティションには、次のノートがあります。

- すべてのサブパーティションについて、range_subpartition_desc、list_subpartition_desc、individual_hash_subpartsまたはhash_subparts_by_quantityを使用して、個々のサブパーティション名およびオプションでその他の特性を指定できます。
- または、ハッシュ・サブパーティションおよびリスト・サブパーティションの場合:
    - サブパーティションの数を指定できます。また、サブパーティションが格納される1つ以上の表領域を指定することもできます。この場合、SYS_SUBPnという形式でサブパーティション名が割り当てられます。
    - サブパーティションの定義を省略すると、サブパーティションのテンプレートが作成されている場合は、そのテンプレートに基づいてサブパーティションが作成されます。サブパーティションのテンプレートが作成されていない場合、1つのハッシュ・サブパーティションまたは1つのデフォルト・リスト・サブパーティションが作成されます。
- すべてのタイプのサブパーティションについて、サブパーティションの定義全体を指定しない場合、次のようにサブパーティション名が割り当てられます。
    - サブパーティションのテンプレートを指定し、またパーティション名を指定した場合、「partition_nameアンダースコア(_)subpartition_name」(たとえば、P1_SUB1)という形式でサブパーティション名が生成されます。
    - サブパーティションのテンプレートを指定していない場合、またはサブパーティションのテンプレートは指定したがパーティション名を指定しなかった場合、SYS_SUBPnという形式でサブパーティション名が生成されます。

reference_partitioning

この句を使用すると、参照によって表をパーティション化できます。参照によるパーティション化は、作成される表(子表)を既存のパーティション表(親表)への参照制約によって同一レベルでパーティション化する方法です。参照によって表をパーティション化すると、その後に親表で実行されるパーティションのメンテナンス操作は子表に自動的にカスケードします。そのため、パーティションのメンテナンス操作は、参照パーティション表で直接実行できません。

親表が時間隔パーティション表の場合は、親表の時間隔パーティションに対応する子の参照パーティション表内のパーティションが、子表への挿入時に作成されます。子表内に時間隔パーティションが作成されるときには、そのパーティションに関連する親表のパーティションからパーティション名が継承されます。子表に表レベルのデフォルト表領域がある場合は、その表領域が新しい時間隔パーティションの表領域として使用されます。それ以外の場合は、表領域が親表のパーティションから継承されます。時間隔パーティション表の参照方法の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

constraint

パーティション化参照制約は、次の条件を満たしている必要があります。

- 作成される表で定義される参照整合性制約を指定する必要があります。これは、親表の主キーまたは一意制約を参照する必要があります。制約は、ENABLE VALIDATE NOT DEFERRABLE状態である必要があります。これは、表の作成時に参照整合性制約を指定するときのデフォルトです。
- 制約で参照されるすべての外部キー列は、NOT NULLである必要があります。
- 制約を指定するときに、references_clauseのON DELETE SET NULL句は指定できません。
- 制約で参照される親表は、既存のパーティション表である必要があります。これは、どの方法でパーティション化されていてもかまいません。
- 外部キーおよび親キーにはPL/SQLファンクションまたはLOB列を参照する仮想列を含めることはできません。

reference_partition_desc

このオプションの句を使用すると、パーティション名を指定し、パーティションの物理特性および記憶特性を定義できます。table_partition_descriptionの副次句の動作は、「table_partition_description」のレンジ・パーティションで説明した動作と同じです。

子の参照パーティション表を作成する場合、その親表が時間隔パーティション表のときに、この句を指定すると、子表の時間隔パーティション以外のパーティションには、パーティション記述子が使用されます。パーティション記述子は、時間隔パーティションには指定できません。

参照パーティション化の制限事項

参照パーティション化には、「一般的なパーティション化の制限事項」に示されている制限事項があります。次の追加の制限が適用されます。

- 参照パーティション化の制限事項は、親表のパーティション計画から導出されます。
- 親表と子表のどちらも自動リスト・パーティション表にすることはできません。
- この句は、索引構成表、外部表またはドメイン索引記憶表に対して指定できません。
- 親表を参照でパーティション化することはできますが、constraintを自己参照型にすることはできません。作成される表を自己参照に基づいてパーティション化することはできません。
- ROW MOVEMENTが親表に対して有効である場合、子表に対しても有効である必要があります。

関連項目:

参照によるパーティション化の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。また、「参照パーティション化の例」を参照してください。

system_partitioning

この句を使用すると、システム・パーティションを作成できます。システム・パーティション化には、パーティション化キー列は必要ありません。またシステム・パーティションには、レンジ境界やリスト境界またはハッシュ・アルゴリズムはありません。システム・パーティションは、パーティション化された実表を持つネストした表やドメイン索引記憶表などの依存表を同一レベルでパーティション化する方法です。

- PARTITION BY SYSTEMのみを指定すると、SYS_Pnという形式のシステム生成の名前で1つのパーティションが作成されます。

- PARTITION BY SYSTEM PARTITIONS integerを指定すると、integerで指定した数(1から1024K-1の範囲)のパーティションが作成されます。
- パーティションの定義は参照パーティションと同じ構文であるため、reference_partition_descを共有します。reference_partition_desc構文で、その他のパーティション属性を指定できます。ただし、table_partition_descriptionで、OVERFLOW句は指定できません。

システム・パーティション化の制限事項

システム・パーティション化には、次の制限事項があります。

- 索引構成表またはクラスタの一部である表は、システム・パーティション化できません。
- コンポジット・パーティション化は、システム・パーティション化ではサポートされていません。
- システム・パーティションは分割できません。
- CREATE TABLE ...AS SELECT文でシステム・パーティション化は指定できません。
- INSERT INTO ... AS subquery文を使用してシステム・パーティション表にデータを挿入するには、拡張パーティション構文を使用して、副問合せによって戻された値が挿入されるパーティションを指定する必要があります。

関連項目:

システム・パーティション化の使用方法の詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。また、「パーティション表と索引の参照」を参照してください。

consistent_hash_partitions

この句は、シャード表に対してのみ有効です。この句を使用すると、コンシステント・ハッシュ・パーティションを作成できます。

文字データ型の各シャーディング・キー列には、宣言された照合(BINARY、USING_NLS_COMP、USING_NLS_SORT、USING_NLS_SORT_CSのいずれか)が必要です。

consistent_hash_with_subpartitions

この句は、シャード表に対してのみ有効です。この句を使用すると、サブパーティションがあるコンシステント・ハッシュを作成できます。

文字データ型の各シャーディング・キー列には、宣言された照合(BINARY、USING_NLS_COMP、USING_NLS_SORT、USING_NLS_SORT_CSのいずれか)が必要です。

range_partitionset_clause

この句を使用すると、レンジ・パーティション・セットを作成できます。

SUBPARTITION BY句のsubpartition_template句内では、サブパーティションの表領域を指定できません。つまり、レンジ、リストおよび個々のハッシュ・サブパーティションには、partitioning_storage_clauseのTABLESPACE句を指定できず、hash_subpartitions_by_quantity句には、STORE IN (tablespace)句を指定できません。

PARTITIONS AUTO句の、range_partitionset_desc句のsubpartition_template句内では、サブパーティションの表領域を指定できます。

文字データ型の各スーパー・シャーディング・キー列またはシャーディング・キー列には、宣言された照合(BINARY、USING_NLS_COMP、USING_NLS_SORT、USING_NLS_SORT_CSのいずれか)が必要です。

list_partitionset_clause

この句を使用すると、リスト・パーティション・セットを作成できます。

SUBPARTITION BY句のsubpartition_template句内では、サブパーティションの表領域を指定できません。つまり、レンジ、リストおよび個々のハッシュ・サブパーティションには、partitioning_storage_clauseのTABLESPACE句を指定できず、hash_subpartitions_by_quantity句には、STORE IN (tablespace)句を指定できません。

PARTITIONS AUTO句の、list_partitionset_desc句のsubpartition_template句内では、サブパーティションの表領域を指定できます。

文字データ型の各スーパー・シャーディング・キー列またはシャーディング・キー列には、宣言された照合(BINARY、USING_NLS_COMP、USING_NLS_SORT、USING_NLS_SORT_CSのいずれか)が必要です。

attribute_clustering_clause

この句を使用して、属性クラスタリングの表を有効化します。属性クラスタリングは、指定された列の内容に基づいて物理的に近いデータをクラスタリングできます。

属性クラスタリングは、tableの列または他の表の結合された値にのみ基づきます。後者は結合属性クラスタリングと呼ばれます。

関連項目:

属性クラスタリングの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

clustering_join

この句を使用して、属性クラスタリングを指定します。JOIN句を使用して、属性クラスタリングの基になる他の表の結合された値を指定します。最大4個のJOIN句を指定できます。

cluster_clause

この句を使用して、表に使用する順序のタイプ(線形順序またはインターリーブ順序)を指定します。LINEARまたはINTERLEAVEDキーワードを指定しない場合、デフォルトはLINEARです。

BY LINEAR ORDER

この句を使用して、線形順序を指定します。このタイプの順序は、指定された列の順序に従ってデータを格納します。この句を指定する場合、最大で10個の列を含むことができる1つのクラスタリング列グループのみ指定できます。

BY INTERLEAVED ORDER

この句を使用して、インターリーブ順序を指定します。このタイプの順序は、複数列のクラスタリングを許可するz順序と似ている特殊な多次元クラスタリング技術を使用します。この句を指定する場合、すべてのグループで最大40個の列を使用した最大で4個のクラスタリング列グループを指定できます。

clustering_columns

この句を使用して、1つ以上のクラスタリング列グループを指定します。

clustering_column_group

この句を使用して、クラスタリング列グループに含まれる1つ以上の列を指定します。

属性クラスタリング列の制限事項

クラスタリング列グループの各文字列には、宣言された照合BINARYまたはUSING_NLS_COMPのいずれかが必要です。

clustering_when

これらの句を使用して、ダイレクト・パス・インサート操作およびデータ移動操作中の属性クラスタリングを許可または禁止します。

ON LOAD

INSERTまたはMERGE操作のいずれかの結果として実行されるダイレクト・パス・インサート(シリアルまたはパラレル)の実行中に属性クラスタリングをYES ON LOADを指定して許可するか、NO ON LOADを指定して禁止します。

デフォルトはYES ON LOADです。

ON DATA MOVEMENT

次の操作中に発生するデータ移動の属性クラスタリングをYES ON DATA MOVEMENTを指定して許可するか、NO ON DATA MOVEMENTを指定して禁止します。

- DBMS_REDEFINITIONパッケージを使用したデータの再定義
- ALTER TABLEの次の句で指定される表パーティションのメンテナンス操作: coalesce_table、merge_table_partitions、move_table_partitionおよびsplit_table_partition

デフォルトはYES ON DATA MOVEMENTです。

zonemap_clause

この句を使用して、表のゾーン・マップを作成します。ゾーン・マップは、clustering_columns句で指定された列を追跡します。

- WITH MATERIALIZED ZONEMAPを指定して、ゾーン・マップを作成します。zonemap_nameでは、作成されるゾーン・マップの名前を指定します。zonemap_nameを省略する場合、ゾーン・マップの名前はZMAP$_tableです。
- WITHOUT MATERIALIZED ZONEMAPを指定すると、ゾーン・マップを作成しません。これはデフォルトです。

後で表を削除するか、DROP CLUSTERINGまたはMODIFY CLUSTERING ... WITHOUT MATERIALIZED ZONEMAPに対してALTER TABLE文を使用すると、ゾーン・マップが削除されます。

関連項目:

ゾーン・マップの詳細は、CREATE MATERIALIZED ZONEMAPを参照してください。

属性クラスタリングの制限事項

属性クラスタリングには、次の制限事項があります。

- 属性クラスタリングは、一時表または外部表に対してサポートされていません。
- 作成される表はヒープ構成表である必要があります。ただし、clustering_join句で指定された表は、ヒープ構成表または索引構成表である可能性があります。
- クラスタリング列は、スカラー・データ型である必要があり、暗号化できません。
- BY LINEAR ORDERを指定する場合、最大で10個の列を含むことができる1つのクラスタリング列グループのみ指定できます。

- BY INTERLEAVED ORDERを指定する場合、すべてのグループで最大40個の列を使用した最大で4個のクラスタリング列グループを指定できます。
- 結合属性クラスタリングの場合
    - ディメンション表の数が4を超えることはできません。
    - 属性クラスタリング列を提供する表への結合は、一意のキーまたは行の重複を回避する主キー列で行う必要があります。
- 属性クラスタリングは、MERGE文または複数表挿入操作を使用して挿入された行を並べ替えません。

CACHE | NOCACHE | CACHE READS

これらの句を使用すると、バッファ・キャッシュ内でのブロックの格納方法を指定できます。LOB記憶域の場合、CACHE、NOCACHEまたはCACHE READSを指定できます。他のタイプの記憶域の場合、CACHEまたはNOCACHEのみ指定できます。

これらの句を指定しない場合、次のようになります。

3. CREATE TABLE文では、NOCACHEがデフォルトです。
4. ALTER TABLE文では、既存の値は変更されません。

この項で説明するCACHEおよびNOCACHEの動作は、直接読取りが使用される場合、またはパラレル問合せを使用して表スキャンが実行される場合には適用されません。

CACHE

アクセス頻度が高いデータの場合、この句を使用して、全表スキャンの実行時に、この表に対して取得されたブロックがバッファ・キャッシュ内の最低使用頻度(LRU)リストの最高使用頻度側に配置されるように指定します。この属性は、小規模な参照表で有効です。

関連項目:

データベースによる最低使用頻度(LRU)リストの保持の詳細は、『Oracle Database概要』を参照してください。

LOB_storage_clauseのパラメータとしてCACHEを使用すると、より高速なアクセスのために、LOBデータ値がバッファ・キャッシュに配置されます。このパラメータは、logging_clauseと組み合せて評価されます。この句を省略すると、BasicFiles LOBとSecureFiles LOBの両方のデフォルト値は、NOCACHE LOGGINGになります。

CACHEの制限事項

CACHEは、索引構成表に対して指定できません。ただし、索引構成表は暗黙的にCACHE動作を提供します。

NOCACHE

アクセス頻度が低いデータの場合、この句を使用して、全表スキャンの実行時に、この表に対して取得されたブロックがバッファ・キャッシュ内のLRUリストの最低使用頻度側に配置されるように指定します。NOCACHEは、LOB記憶域のデフォルトです。

LOB_storage_clauseのパラメータとしてNOCACHEを使用すると、LOB値はバッファ・キャッシュに入りません。NOCACHEは、LOB記憶域のデフォルトです。

NOCACHEの制限事項

NOCACHEは、索引構成表に対して指定できません。

CACHE READS

CACHE READSはLOB記憶域にのみ適用されます。LOB値が書込み操作中ではなく読取り操作中にバッファ・キャッシュに入れられるように指定します。

logging_clause

この句を使用すると、データ・ブロックの記憶域を記録するかどうかを指定できます。

関連項目:

`logging_clause`を`LOB_parameters`の一部として指定した場合については、「logging_clause」を参照してください。

RESULT_CACHE句

この句を使用すると、この表が指名されている文または問合せブロックの結果を、結果キャッシュに格納する対象とするかどうかを指定できます。結果キャッシュでは2つのモードを使用できます。

6. DEFAULT: 結果キャッシュは表レベルでは決定されません。RESULT_CACHE_MODE初期化パラメータがFORCEに設定されているか、またはMANUALに設定されていて問合せにRESULT_CACHEヒントを指定する場合は、問合せが結果キャッシュの対象となります。これは、この句を指定しない場合のデフォルトです。
7. FORCE: 問合せで指定したすべての表がこの設定になっている場合は、問合せにNO_RESULT_CACHEヒントが指定されていないかぎり、その問合せは常にキャッシュの対象となります。問合せで指定した1つ以上の表がDEFAULTに設定されている場合、この問合せにおける有効な表の注釈はDEFAULTであるとみなされ、前述したセマンティクスに従います。

DBA_、ALL_およびUSER_TABLESデータ・ディクショナリ・ビューのRESULT_CACHE列を問い合せると、表の結果キャッシュのモードを確認できます。

RESULT_CACHEおよびNO_RESULT_CACHEのSQLヒントは、これらの結果キャッシュの表の注釈および RESULT_CACHE_MODE初期化パラメータより優先されます。RESULT_CACHE_MODEをFORCEに設定すると、その設定がこの表の注釈句より優先されます。

ノート:

RESULT_CACHE_MODE を FORCE に設定することはお薦めしません。なぜなら、データベースとクライアントがすべての問合せをキャッシュしようとするため、パフォーマンスおよびラッチのオーバーヘッドが非常に大きくなる可能性があるためです。

関連項目:

- 結果キャッシュの概要は、『Oracle Call Interfaceプログラマーズ・ガイド』および『Oracle Database概要』を参照してください。
- この句の使用方法の詳細は、『Oracle Databaseパフォーマンス・チューニング・ガイド』を参照してください。
- RESULT_CACHE_MODE初期化パラメータと*_TABLESデータ・ディクショナリ・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。
- ヒントの詳細は、「RESULT_CACHEヒント」および「NO_RESULT_CACHEヒント」を参照してください。

parallel_clause

parallel_clauseを使用すると、表をパラレル作成した後、問合せおよびDMLのINSERT、UPDATE、DELETEおよびMERGEに対するデフォルトの並列度を設定できます。

ノート:

parallel_clause 構文は、以前のリリースの構文にかわるものです。以前のリリースの構文は下位互換用にサポートされていますが、説明されている動作とわずかに異なることがあります。

NOPARALLEL

NOPARALLELを指定すると、シリアル実行が行われます。これはデフォルトです。

PARALLEL

PARALLELを指定すると、並列度を選択できます。並列度は、すべての関係するインスタンスで使用可能なCPUの数に、初期化パラメータPARALLEL_THREADS_PER_CPUの値を掛けたものです。

PARALLEL integer

integerには、パラレル操作で使用するパラレル・スレッド数である並列度を指定します。各パラレル・スレッドは、1、2個のパラレル実行サーバーを使用します。通常、最適な並列度が計算されるため、integerに値を指定する必要はありません。

関連項目:

この句の詳細は、「parallel_clause」を参照してください。

NOROWDEPENDENCIES | ROWDEPENDENCIES

この句では、表が行レベル依存の追跡を使用するかどうかを指定できます。この機能を使用すると、表の各行は、行を変更した最後のトランザクションのコミット時刻以降を示すシステム変更番号(SCN)を持つことになります。表の作成後は、この設定を変更できません。

ROWDEPENDENCIES

ROWDEPENDENCIESを指定すると、行レベル依存の追跡を有効にできます。この設定は、主にレプリケーション環境でパラレル伝播を許可する場合に便利です。各行につきサイズが6バイト増えます。

ROWDEPENDENCIES句の制限事項

Oracleでは、行レベル依存の追跡を使用する表の表圧縮はサポートされません。ROWDEPENDENCIES句とtable_compression句の両方を指定した場合、table_compression句は無視されます。ROWDEPENDENCIES属性を削除するには、DBMS_REDEFINITIONパッケージを使用して表を再定義するか、表を再作成する必要があります。

NOROWDEPENDENCIES

NOROWDEPENDENCIESを指定すると、tableで行レベル依存の追跡機能が使用されなくなります。これはデフォルトです。

enable_disable_clause

enable_disable_clauseを使用すると、制約が適用されるかどうかを指定できます。デフォルトでは、制約はENABLE VALIDATE状態で作成されます。

制約の有効化および無効化の制限事項

制約を有効および無効にするには、次の制限事項があります。

- 整合性制約を使用可能または使用禁止にする場合、この文または前の文に制約を定義する必要があります。
- 参照された一意キー制約または主キー制約が使用可能でない場合は、外部キー制約を使用可能にできません。
- `using_index_clause`の`index_properties`句で、`INDEXTYPE IS ...`句は制約の定義において有効ではありません。

関連項目:

制約の詳細は、「constraint」を参照してください。また、「表の作成: ENABLE/DISABLEの例」も参照してください。

ENABLE句

この句を使用すると、表のデータに制約を適用できます。この句の詳細は、制約に関する項目の「ENABLE句」を参照してください。

DISABLE句

この句を使用すると、整合性制約を無効にできます。この句の詳細は、制約に関する項目の「DISABLE句」を参照してください。

UNIQUE

`UNIQUE`句を使用すると、指定された列または列の組合せに対して定義された一意制約を有効または無効にできます。

PRIMARY KEY

`PRIMARY KEY`句を使用すると、表に対して定義された主キー制約を有効または無効にできます。

CONSTRAINT

`CONSTRAINT`句を使用すると、`constraint_name`に指定する整合性制約を有効または無効にできます。

KEEP | DROP INDEX

この句を使用すると、一意キー制約または主キー制約を適用するためにOracle Databaseで使用する索引を保持または削除できます。

索引の保持と削除の制限事項

一意キー制約または主キー制約が無効になっている場合にのみ、この句を指定できます。

using_index_clause

`using_index_clause`を使用すると、一意キー制約または主キー制約を適用するためにOracle Databaseで使用する索引を指定するか、制約の適用に使用される索引を作成するようデータベースに指示できます。

関連項目:

- index_attributes、global_partitioned_indexおよびlocal_partitioned_index句の詳細、および索引に関連する`NOSORT`と`logging_clause`の詳細は、「CREATE INDEX」を参照してください。
- `using_index_clause`、`PRIMARY KEY`制約および`UNIQUE`制約の詳細は、「constraint」を参照してくださ

い。

- 索引を使用した制約の適用例は、「明示的な索引の制御例」を参照してください。

CASCADE

CASCADEを指定すると、指定した整合性制約に依存する整合性制約を無効にできます。参照整合性制約を構成する主キーまたは一意キーを使用禁止にする場合、この句を指定します。

CASCADEの制限事項

DISABLEを指定した場合のみ、CASCADEを指定できます。

row_movement_clause

row_movement_clauseを使用すると、表の行が移動されるかどうかを指定できます。表の圧縮時や、パーティション・データの更新操作時などに、行を移動できます。

> ノート:
>
> データ・アクセスで静的な ROWID が必要な場合は、行移動を有効にしないでください。通常の表(ヒープ構成表)の場合は、行を移動すると、その行の ROWID が変更されます。索引構成表での行の移動の場合は、論理 ROWID の物理推測コンポーネントは不正確になりますが、論理 ROWID は有効のままです。

- ENABLEを指定すると、行の移動を許可できます。このとき、ROWIDは変更されます。
- DISABLEを指定すると、行の移動を禁止できます。このとき、ROWIDは変更されません。

この句を省略すると、行移動は使用禁止になります。

行の移動の制限事項

この句は、非パーティション索引構成表に対して指定できません。

flashback_archive_clause

この句を指定するには、指定されたフラッシュバック・アーカイブに対するFLASHBACK ARCHIVEオブジェクト権限が必要です。この句を使用すると、表の履歴追跡を有効または無効にできます。

- 表の追跡を有効にするには、FLASHBACK ARCHIVEを指定します。flashback_archiveは、この表に特定のフラッシュバック・アーカイブを指定する場合に指定できます。指定するフラッシュバック・アーカイブは、すでに存在している必要があります。

  flashback_archiveを省略すると、システムに対して指定されているデフォルトのフラッシュバック・アーカイブが使用されます。システムにデフォルトのフラッシュバック・アーカイブが指定されていない場合は、flashback_archiveを指定する必要があります。

- 表の追跡を無効にするには、NO FLASHBACK ARCHIVEを指定します。これはデフォルトです。

flashback_archive_clauseの制限事項

フラッシュバック・データ・アーカイブには、次の制限事項があります。

- この句は、ネストした表、クラスタ化表、一時表、リモート表または外部表に対しては指定できません。
- ハイブリッド列圧縮によって圧縮された表に対しては、この句を指定できません。

- この句を指定する表には、LONG列またはネストした表の列を含めることはできません。
- この句を指定した後でエクスポート・ユーティリティ、インポート・ユーティリティまたはトランスポータブル表領域機能を使用して表を別のデータベースにコピーすると、コピー先の表では追跡は有効にならず、元の表のアーカイブ・データは使用できません。

関連項目:

- フラッシュバック・タイム・トラベルの使用についての一般情報は、Oracle Database開発ガイドを参照してください。
- フラッシュバック・アーカイブの割当て制限属性と保存属性の変更およびフラッシュバック・アーカイブにおける表領域の記憶域の追加と変更の詳細は、ALTER FLASHBACK ARCHIVEを参照してください。

ROW ARCHIVAL

この句を指定すると、tableで行のアーカイブを有効にできます。この句によりインデータベース・アーカイブを実装でき、表の行をアクティブまたはアーカイブ済として指定できます。その後、表内のアクティブな行のみを対象に問合せを実行できます。

この句を指定すると、非表示列ORA_ARCHIVE_STATEが表に作成されます。この列のデータ型は、VARCHAR2です。この列に値0または1を指定すると、行がアクティブであるか(0)アーカイブ対象であるか(1)を示すことができます。表へのデータの挿入時、ORA_ARCHIVE_STATEに値を指定しなかった場合、その値は0に設定されます。

- セッションがROW ARCHIVE VISIBILITY = ACTIVEの場合、データベースは表に対する問合せの実行時に、アクティブな行のみを考慮します。
- セッションがROW ARCHIVE VISIBILITY = ALLの場合、データベースは表に対する問合せの実行時に、すべての行を考慮します。

関連項目:

- セッションにおける行のアーカイブの可視性を構成する方法を学習するには、「ALTER SESSION」の「セマンティクス」の句を参照してください。
- 既存の表の行アーカイブを有効または無効にする方法を学習するには、「ALTER TABLE」の「[NO] ROW ARCHIVAL」句を参照してください。
- インデータベース・アーカイブの詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

FOR EXCHANGE WITH TABLE

この句を使用すると、既存のパーティション表の構造に一致する表を作成できます。これら2つの表は、パーティションとサブパーティションの交換に適格となります。tableには、既存のパーティション表を指定します。schemaには、既存のパーティション表を含むスキーマを指定します。schemaを指定しない場合、表は自分のスキーマ内にあるとみなされます。

この操作により、パーティション表のメタデータ・クローン(データなし)が作成されます。このクローンには、元の表と同じ列順序および列プロパティが含まれます。この操作中にクローンにコピーされる列プロパティには、使用禁止列、非表示列、仮想式列、ファンクション索引式列、および他の内部設定と属性が含まれます。既存のパーティション表の索引は、クローン表には作成されません。

後でALTER TABLEのexchange_partition_subpart句を使用して、2つの表の間でパーティションまたはサブパーティションを交換できます。詳細は、「ALTER TABLE」の項目の「exchange_partition_subpart」を参照してください。

FOR EXCHANGE WITH TABLEの制限事項

文字データ型の各スーパー・シャーディング・キー列またはシャーディング・キー列には、宣言された照合(BINARY、USING_NLS_COMP、USING_NLS_SORT、USING_NLS_SORT_CSのいずれか)が必要です。

FOR EXCHANGE WITH TABLE句には、次の制限事項があります。

- この句を指定する場合、relational_properties句は指定できません。
- この句を指定する場合、table_properties句内で指定できるのは、table_partitioning_clauseのみです。
- table_partitioning_clause内では、文字データ型が指定された各キー列には、宣言された照合(BINARY、USING_NLS_COMP、USING_NLS_SORT、USING_NLS_SORT_CSのいずれか)が必要です。
- コンポジット・パーティション表のパーティションのクローンを作成する場合は、交換するパーティションのサブパーティション化に厳密に一致する適切なtable_partitioning_clauseを明示的に指定する必要があります。
- パーティション化された索引構成表のクローンは作成できません。
- パーティション表の統計設定はクローニングされません。たとえば、増分統計が有効になっているパーティション表との交換を実行する場合は、クローン表の表シノプシスの作成を手動で有効にする必要があります。パーティション表の増分統計のメンテナンスの詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。
- ターゲット表でVPDポリシーが有効になっている場合、この句は使用できません。

AS subquery

副問合せを指定して表の内容を定義します。表の作成時に、副問合せの結果として戻された行を表の中に挿入します。

オブジェクト表の場合、subqueryには表の型に対応する1つの式、または表の型の最上位属性の数のどちらかを設定できます。詳細は、「SELECT」を参照してください。

subqueryが、既存のマテリアライズド・ビューと部分的または完全に同じビューを戻す場合、subqueryに指定された1つ以上の表のかわりにマテリアライズド・ビューがクエリー・リライトに使用されることがあります。

関連項目:

マテリアライズド・ビューおよびクエリー・リライトの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

データ型およびデータ長は、副問合せから導出されます。整合性制約や、その他の列および表の属性には次の規則が適用されます。

- 副問合せで、列を含む式ではなく列を選択した場合、選択した表の列にNOT NULL制約が明示的に作成されていると、新しい表の対応する列にもNOT DEFERRABLEおよびVALIDATE状態の同じ制約が自動的に定義されます。制約に違反する行がある場合、表は作成されずエラーが戻されます。
- 選択した表の列に暗黙的に作成されたNOT NULL制約(たとえば、主キー)は、新しい表には引き継がれません。
- 主キー、一意キー、外部キー、CHECK制約、パーティション化条件、索引および列のデフォルト値は、新しい表に引き継がれません。
- 選択した表がパーティション化されている場合、新しい表を同じ方法でパーティション化するか、別の方法でパーティション化するか、またはパーティション化しないかを選択できます。パーティション化は、新しい表に引き継がれません。AS subquery句の前に、CREATE TABLE文の一部として、必要なパーティション化を指定します。

- 選択した表で透過的データ暗号化を使用して暗号化された列は、新しい表の作成時に暗号化として列を定義しないかぎり、新しい表では暗号化されません。

ノート:

機密列は、データを移入する前に暗号化することをお薦めします。これにより、機密データのクリア・テキスト・コピーの作成を回避できます。

subqueryによって返される各列に列名がある場合、または指定した列別名を持つ式である場合、表定義から列を完全に省略できます。この場合、tableの列名はsubqueryの列の名前と同じになります。索引構成表の作成は例外です。この場合、主キー列を指定する必要があるため、表定義で列を指定する必要があります。

TO_LOBファンクションと組み合せてsubqueryを使用すると、別の表のLONG列の値を、作成する表の列のLOB値に変換できます。

関連項目:

- LONGデータをLOBにコピーする理由およびタイミングについては、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。
- TO_LOBファンクションの使用方法は、「変換ファンクション」を参照してください。
- order_by_clauseの詳細は、「SELECT」を参照してください。
- AS subquery句を使用するときの統計収集の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

parallel_clause

この文でparallel_clauseを指定した場合、INITIAL記憶域パラメータに対して指定する値は無視され、かわりにNEXTパラメータの値が使用されます。

関連項目:

これらのパラメータの詳細は、「storage_clause」を参照してください。

ORDER BY

ORDER BY句を使用すると、副問合せによって戻される行の順序付けを行うことができます。

この句をCREATE TABLEで指定した場合、この句が表全体にわたるデータを順序付けるとはかぎりません。たとえば、パーティション間での順序付けは行われません。同じキーの索引をORDER BYキー列として作成する場合に、この句を指定します。Oracle Databaseは、ORDER BYキーのデータをクラスタ化し、索引キーに対応させます。

表を定義する問合せの制限事項

表問合せには、次の制限事項があります。

- 表中の列数は、副問合せ中の式の数と同じである必要があります。
- 列定義では、列名、デフォルト値および整合性制約のみを指定できます。データ型は指定できません。

- 表が参照パーティション表で、なおかつ制約が表のパーティション化参照制約である場合を除いては、AS subqueryを含むCREATE TABLE文には、外部キー制約を定義できません。それ以外のすべての場合は、制約を指定せずに表を作成し、後でALTER TABLE文を使用してその制約を追加する必要があります。

object_table

OF句を使用すると、明示的にobject_type型のオブジェクト表を作成できます。オブジェクト表の各列は、object_type型の最上位の属性に対応します。各行には、オブジェクト・インスタンスが入り、また各インスタンスには、行の挿入時に一意のシステム生成オブジェクト識別子が割り当てられます。schemaを省略した場合、自分のスキーマ内にオブジェクト表が作成されます。

オブジェクト表、XMLType表、オブジェクト・ビューおよびXMLTypeビューには、列名は付けられません。そのため、システム生成疑似列OBJECT_IDが定義されます。問合せでこの列名を使用し、WITH OBJECT IDENTIFIER句を指定して、オブジェクト・ビューを作成できます。

関連項目:

オブジェクト列と表の例

object_table_substitution

object_table_substitution句を使用すると、サブタイプに対応する行オブジェクトの、このオブジェクト表への挿入を許可するかどうかを指定できます。

NOT SUBSTITUTABLE AT ALL LEVELS

NOT SUBSTITUTABLE AT ALL LEVELSを指定すると、作成するオブジェクト表は置換できなくなります。また、置換は、すべての埋込みオブジェクト属性および埋込みのネストした表と配列の要素には使用禁止です。デフォルトは、SUBSTITUTABLE AT ALL LEVELSです。

関連項目:

- オブジェクト型の作成の詳細は、「CREATE TYPE」を参照してください。
- REF型の使用方法の詳細は、「ユーザー定義型」、「ユーザー定義ファンクション」、「SQL式」、「CREATE TYPE」および『Oracle Databaseオブジェクト・リレーショナル開発者ガイド』を参照してください。

object_properties

オブジェクト表のプロパティは、基本的にリレーショナル表と同じです。ただし、列を指定するかわりに、オブジェクトの属性を指定します。

attributeには、オブジェクト内の項目の修飾した列名を指定します。

oid_clause

oid_clauseを使用すると、オブジェクト表のオブジェクト識別子がシステム生成されるか、表の主キーを基に作成されるかを指定できます。デフォルトはSYSTEM GENERATEDです。

oid_clauseの制限事項

この句には、次の制限事項があります。

- 主キー制約を表に指定していないと、OBJECT IDENTIFIER IS PRIMARY KEYは指定できません。

- この句は、ネストした表には指定できません。

ノート:

主キー・オブジェクト識別子はローカルで一意ですが、グローバルで一意であるとはかぎりません。グローバルで一意の識別子が必要な場合は、主キーがグローバルで一意であることを確認してください。

oid_index_clause

この句は、oid_clauseをSYSTEM GENERATEDとして指定している場合のみに適用されます。非表示のオブジェクト識別子列に索引を指定します。また、任意に記憶特性を指定します。

indexには、非表示のシステム生成オブジェクト識別子列の索引の名前を指定します。indexを省略すると、名前が生成されます。

physical_propertiesおよびtable_properties

これらの句のセマンティクスについては、リレーショナル表の対応する項を参照してください。「physical_properties」および「table_properties」を参照してください。

XMLType_table

XMLType_table構文を使用すると、XMLTypeデータ型の表を作成できます。XMLType表の作成に使用されるほとんどの句のセマンティクスは、オブジェクト表のセマンティクスと同じです。この項では、XMLType表固有の句について説明します。

オブジェクト表、XMLType表、オブジェクト・ビューおよびXMLTypeビューには、列名は付けられません。そのため、システム生成疑似列OBJECT_IDが定義されます。問合せでこの列名を使用し、WITH OBJECT IDENTIFIER句を指定して、オブジェクト・ビューを作成できます。

XMLSchema_spec

この句を使用すると、登録済XMLスキーマ(XMLSCHEMA句で登録するかELEMENT句の一部として登録)のURLおよびXML要素名を指定できます。

XMLスキーマのURLは省略可能ですが、要素は必ず指定します。XMLスキーマのURLを指定する場合は、DBMS_XMLSCHEMAパッケージを使用してXMLスキーマをあらかじめ登録しておく必要があります。

オプションのSTORE ALL VARRAYS AS句を使用すると、XMLType表または列に含まれるすべてのVARRAYの格納方法を指定できます。

- STORE ALL VARRAYS AS LOBSを指定すると、すべてのVARRAYがLOBとして格納されます。
- STORE ALL VARRAYS AS TABLESを指定すると、すべてのVARRAYが表として格納されます。

オプションのALLOW | DISALLOW句は、BINARY XML記憶域を指定した場合にのみ有効です。

- ALLOW NONSCHEMAを指定すると、スキーマ・ベース以外のドキュメントをXMLType列に格納できます。
- DISALLOW NONSCHEMAを指定すると、スキーマ・ベース以外のドキュメントはXMLType列に格納できません。これはデフォルトです。
- ALLOW ANYSCHEMAを指定すると、スキーマ・ベースの任意のドキュメントをXMLType列に格納できます。
- DISALLOW ANYSCHEMAを指定すると、スキーマ・ベースの任意のドキュメントはXMLType列に格納できません。これはデフォルトです。

関連項目:

- DBMS_XMLSCHEMAパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- XMLデータの作成方法および使用方法の詳細は、『Oracle XML DB開発者ガイド』を参照してください。
- 「XMLType表の例」

MEMOPTIMIZE FOR READ

高速参照を有効にするには、この句を使用します。高速参照を使用すると、頻度の高いデータ問合せ操作でパフォーマンスが向上します。MEMOPTIMIZE_POOL_SIZE初期化パラメータは、memoptimizeプールのサイズを制御します。この機能は、SGAから追加のメモリーを使用することに注意してください。

- この句は、表の最上位の属性として指定する必要があり、パーティション・レベルまたはサブパーティション・レベルで指定することはできません。
- 表からデータを読み取る前に、MEMOPTIMIZE FOR READに対して表を明示的に有効にする必要があります。

MEMOPTIMIZE FOR WRITE

高速収集を有効にするには、この句を使用します。高速収集は、モノのインターネット(IoT)アプリケーションから1行データを挿入する頻繁な操作でメモリーを最適化します。

- MEMOPTIMZE FOR WRITEは最上位の属性であり、パーティションまたはサブパーティション・レベルでは使用できません。
- 表のデータをIGAに書き込むには、その表であらかじめMEMOPTIMIZE FOR WRITEが有効になっている必要があります。

制限事項

ブロックチェーンおよび不変表は、MEMOPTIMZE FOR WRITEをサポートしていません。

BFILEデータ型の列は、MEMOPTIMZE FOR WRITEをサポートしていません。

PARENT

この句を使用して、シャード表ファミリに子表を作成できます。

シャード表ファミリは、同様にシャーディングされた一連の表です。表ファミリにあるすべてのテーブルに対応するパーティションは、同じシャードに格納されます。これにより、表ファミリのデータを問い合せるときに、マルチシャード結合の数を最小限に抑えることができます。

シャード表ファミリの作成方法は2種類あります。参照パーティション化を使用する方法をお薦めします。ただし、参照パーティション化に必要な主キー制約と外部キー制約を作成できない場合や、作成が望ましくない場合は、PARENT句を使用してシャード表ファミリを作成できます。

シャード表ファミリを作成するための規則は、使用する方法によって異なります。PARENT句を使用してシャード表ファミリを作成する場合は、次の規則が適用されます。

- シャード表ファミリには2レベルの表(親表と1つ以上の子表)のみを含めることができます。
- ファミリ内のすべての表は、同じパーティション化スキームを使用して明示的にパーティション化する必要があります。各表は、別々のサブパーティション化スキームを使用することも、一切使用しないこともできます。

- 最初に親表を作成する必要があります。また、この表はシャード表にする必要があります。
- CREATE SHARDED TABLE ... PARENT ...文を使用して、各子表を作成できます。tableには、親表の名前を指定します。schemaには、親表を含むスキーマを指定します。schemaを省略すると、親表は自分のスキーマ内にあるとみなされます。

システム・シャーディングを使用すれば複数のシャード表ファミリを作成できますが、コンポジットまたはユーザー定義のシャーディングを使用する場合は最大1つのみを作成できます。

関連項目:

Oracle Shardingの使用

例

表の作成: 一般的な例

次の文は、人事情報のサンプル・スキーマ(hr)が所有するemployees表を定義します。テスト・データベースでこの例を使用できるように、表および制約には不確定な名前が指定されています。

```
CREATE TABLE employees_demo
    ( employee_id    NUMBER(6)
    , first_name     VARCHAR2(20)
    , last_name      VARCHAR2(25)
          CONSTRAINT emp_last_name_nn_demo NOT NULL
    , email          VARCHAR2(25)
          CONSTRAINT emp_email_nn_demo     NOT NULL
    , phone_number   VARCHAR2(20)
    , hire_date      DATE  DEFAULT SYSDATE
          CONSTRAINT emp_hire_date_nn_demo  NOT NULL
    , job_id         VARCHAR2(10)
       CONSTRAINT     emp_job_nn_demo  NOT NULL
    , salary         NUMBER(8,2)
       CONSTRAINT     emp_salary_nn_demo  NOT NULL
    , commission_pct NUMBER(2,2)
    , manager_id     NUMBER(6)
    , department_id  NUMBER(4)
    , dn             VARCHAR2(300)
    , CONSTRAINT     emp_salary_min_demo
                     CHECK (salary > 0)
    , CONSTRAINT     emp_email_uk_demo
                     UNIQUE (email)
    ) ;
```

この表は12列で構成されます。employee_id列は、NUMBERデータ型です。hire_date列は、データ型がDATE、デフォルト値がSYSDATEです。last_name列は、VARCHAR2型で、NOT NULL制約があります。他の列については省略します。

表の作成: 記憶域の例

記憶域の容量が小さいexample表領域の中に同じemployees_demo表を定義するには、次の文を発行します。

```
CREATE TABLE employees_demo
    ( employee_id    NUMBER(6)
    , first_name     VARCHAR2(20)
    , last_name      VARCHAR2(25)
          CONSTRAINT emp_last_name_nn_demo NOT NULL
    , email          VARCHAR2(25)
          CONSTRAINT emp_email_nn_demo     NOT NULL
    , phone_number   VARCHAR2(20)
    , hire_date      DATE  DEFAULT SYSDATE
          CONSTRAINT emp_hire_date_nn_demo  NOT NULL
```

```
    , job_id          VARCHAR2(10)
       CONSTRAINT     emp_job_nn_demo  NOT NULL
    , salary          NUMBER(8,2)
       CONSTRAINT     emp_salary_nn_demo  NOT NULL
    , commission_pct NUMBER(2,2)
    , manager_id      NUMBER(6)
    , department_id  NUMBER(4)
    , dn              VARCHAR2(300)
    , CONSTRAINT      emp_salary_min_demo
                      CHECK (salary > 0)
    , CONSTRAINT      emp_email_uk_demo
                      UNIQUE (email)
    )
   TABLESPACE example
   STORAGE (INITIAL 8M);
```

DEFAULT ON NULL列値を含む表の作成: 例

次の文では、従業員データを格納するために使用できる、表myempを作成します。department_id列は、値が50のDEFAULT ON NULL列値で定義されています。そのため、それ以降のINSERT文でNULL値をdepartment_idに割り当てようとすると、そのかわりに値50が割り当てられるようになります。

```
CREATE TABLE myemp (employee_id number, last_name varchar2(25),
                    department_id NUMBER DEFAULT ON NULL 50 NOT NULL);
```

employees表内のemployee_id 178は、department_idにNULL値を保持しています。

```
SELECT employee_id, last_name, department_id
  FROM employees
  WHERE department_id IS NULL;
EMPLOYEE_ID LAST_NAME                 DEPARTMENT_ID
----------- ------------------------- -------------
        178 Grant
```

employees表のemployee_id列、last_name列、およびdepartment_id列のデータを、myemp表に移入します。

```
INSERT INTO myemp (employee_id, last_name, department_id)
  (SELECT employee_id, last_name, department_id from employees);
```

myemp表のemployee_id 178は、department_idに50の値を保持しています。

```
SELECT employee_id, last_name, department_id
  FROM myemp
  WHERE employee_id = 178;

EMPLOYEE_ID LAST_NAME                 DEPARTMENT_ID
----------- ------------------------- -------------
        178 Grant                              50
```

ID列を含む表の作成例:

次の文では、ID列idを含む表t1を作成します。この順序ジェネレータは、増加する整数値を常にidに割り当てます。この整数値は、1から始まります。

```
CREATE TABLE t1 (id NUMBER GENERATED AS IDENTITY);
```

次の文では、ID列idを含む表t2を作成します。この順序ジェネレータは、デフォルトで、100から始まり10ずつ増加する整数値をidに割り当てるようになります。

```
CREATE TABLE t2 (id NUMBER GENERATED BY DEFAULT AS IDENTITY (START WITH 100 INCREMENT
BY 10));
```

表の作成: 一時表の例

次の文は、販売員が使用するサンプル・データベースの一時表today_salesを作成します。各販売員のセッションは、自身のその日の販売データを表に格納します。一時的なデータは、セッションの終わりに削除されます。

```
CREATE GLOBAL TEMPORARY TABLE today_sales
   ON COMMIT PRESERVE ROWS
   AS SELECT * FROM orders WHERE order_date = SYSDATE;
```

遅延セグメント作成を使用する表の作成: 例

次の文は、遅延セグメント作成を使用する表を作成します。この表のデータを保持するためのセグメントは、表にデータが挿入されるまで作成されません。

```
CREATE TABLE later (col1 NUMBER, col2 VARCHAR2(20))    SEGMENT CREATION DEFERRED;
```

置換可能な表および列のサンプル

次の文は、置換可能な表の作成に使用できる型の階層を作成します。employee_t型は、person_t型からnameおよびssn属性を継承し、department_idおよびsalary属性を追加しています。part_time_emp_t型は、employee_tおよび(employee_tを介して)person_tからすべての属性を継承し、num_hrs属性を追加しています。part_time_emp_tはデフォルトで最終型であるため、そのサブタイプを作成できません。

```
CREATE TYPE person_t AS OBJECT (name VARCHAR2(100), ssn NUMBER)
   NOT FINAL;
/
CREATE TYPE employee_t UNDER person_t
   (department_id NUMBER, salary NUMBER) NOT FINAL;
/
CREATE TYPE part_time_emp_t UNDER employee_t (num_hrs NUMBER);
/
```

次の文は、person_t型から置換可能な表を作成します。

```
CREATE TABLE persons OF person_t;
```

次の文は、person_t型の置換可能な列で表を作成します。

```
CREATE TABLE books (title VARCHAR2(100), author person_t);
```

personsまたはbooksに挿入するときに、person_tまたはそのサブタイプの属性に対する値を指定できます。挿入文の例は、「置換可能な表および列への挿入:例」を参照してください。

組込みファンクションおよび条件を使用して、このような表からデータを抽出することができます。例は、ファンクション「TREAT」および「SYS_TYPEID」、および「IS OF type条件」の条件を参照してください。

表の作成: パラレル化の例

次の文は、最適な数のパラレル実行サーバーを使用する表を作成して、employeesをスキャンし、dept_80に移入します。

```
CREATE TABLE dept_80
   PARALLEL
   AS SELECT * FROM employees
   WHERE department_id = 80;
```

パラレル化を使用することによって、表を作成するためにパラレル実行サーバーが使用されるため、表作成が高速化されます。表が作成された後、表へのアクセスに表作成と同じ並列度が使用されるため、表の問合せも高速化します。

次の文は、同じ表をシリアルで作成します。後続のDMLおよび表の問合せもシリアルで実行されます。

```
CREATE TABLE dept_80
   AS SELECT * FROM employees
   WHERE department_id = 80;
```

表の作成: ENABLE/DISABLEの例

次の文は、サンプル表departmentsの作成方法を示しています。この例では、NOT NULL制約が定義され、ENABLE VALIDATE状態に置かれます。テスト・データベースでこの例を複製できるように、表に仮定の名前が指定されています。

```
CREATE TABLE departments_demo
    ( department_id    NUMBER(4)
    , department_name  VARCHAR2(30)
          CONSTRAINT  dept_name_nn  NOT NULL
    , manager_id       NUMBER(6)
    , location_id      NUMBER(4)
    , dn               VARCHAR2(300)
    ) ;
```

次の文は、同じdepartments_demo表を作成しますが、使用禁止の主キー制約も定義します。

```
CREATE TABLE departments_demo
    ( department_id    NUMBER(4)   PRIMARY KEY DISABLE
    , department_name  VARCHAR2(30)
          CONSTRAINT  dept_name_nn  NOT NULL
    , manager_id       NUMBER(6)
    , location_id      NUMBER(4)
    , dn               VARCHAR2(300)
    ) ;
```

ネストした表の例

次の文は、ネストした表の列ad_textdocs_ntabを使用したサンプル表pm.print_mediaの作成方法を示しています。

```
CREATE TABLE print_media
    ( product_id        NUMBER(6)
    , ad_id             NUMBER(6)
    , ad_composite      BLOB
    , ad_sourcetext     CLOB
    , ad_finaltext      CLOB
    , ad_fltextn        NCLOB
    , ad_textdocs_ntab  textdoc_tab
    , ad_photo          BLOB
    , ad_graphic        BFILE
    , ad_header         adheader_typ
    ) NESTED TABLE ad_textdocs_ntab STORE AS textdocs_nestedtab;
```

表の作成: マルチレベル・コレクションの例

次の例は、アカウント・マネージャが2つのレベルのネストした表を使用して顧客の表を作成する方法を示しています。

```
CREATE TYPE phone AS OBJECT (telephone NUMBER);
/
CREATE TYPE phone_list AS TABLE OF phone;
/
CREATE TYPE my_customers AS OBJECT (
   cust_name VARCHAR2(25),
   phones phone_list);
/
CREATE TYPE customer_list AS TABLE OF my_customers;
/
CREATE TABLE business_contacts (
   company_name VARCHAR2(25),
   company_reps customer_list)
   NESTED TABLE company_reps STORE AS outer_ntab
   (NESTED TABLE phones STORE AS inner_ntab);
```

前述の例を次のように使用した場合、内部のネストした表に列または属性名がない場合にCOLUMN_VALUEキーワードが使用されます。

```
CREATE TYPE phone AS TABLE OF NUMBER;
/
CREATE TYPE phone_list AS TABLE OF phone;
/
CREATE TABLE my_customers (
   name VARCHAR2(25),
   phone_numbers phone_list)
   NESTED TABLE phone_numbers STORE AS outer_ntab
   (NESTED TABLE COLUMN_VALUE STORE AS inner_ntab);
```

表の作成: LOB列の例

次の文は、いくつかのLOB記憶特性が追加されたpm.print_media表を作成した文の例です。

```
CREATE TABLE print_media_new
    ( product_id        NUMBER(6)
    , ad_id             NUMBER(6)
    , ad_composite      BLOB
    , ad_sourcetext     CLOB
    , ad_finaltext      CLOB
    , ad_fltextn        NCLOB
    , ad_textdocs_ntab  textdoc_tab
    , ad_photo          BLOB
    , ad_graphic        BFILE
    , ad_header         adheader_typ
    ) NESTED TABLE ad_textdocs_ntab STORE AS textdocs_nestedtab_new
    LOB (ad_sourcetext, ad_finaltext) STORE AS
      (TABLESPACE example
       STORAGE (INITIAL 6144)
       CHUNK 4000
       NOCACHE LOGGING);
```

この例では、CHUNKの値を4096(2048のブロック・サイズの近似倍数)まで切り上げます。

索引構成表の例

次の文は、索引構成されているサンプル表hr.countriesの例です。

```
CREATE TABLE countries_demo
    ( country_id      CHAR(2)
      CONSTRAINT country_id_nn_demo NOT NULL
    , country_name    VARCHAR2(40)
    , currency_name   VARCHAR2(25)
    , currency_symbol VARCHAR2(3)
    , region          VARCHAR2(15)
    , CONSTRAINT    country_c_id_pk_demo
                    PRIMARY KEY (country_id ) )
    ORGANIZATION INDEX
    INCLUDING   country_name
    PCTTHRESHOLD 2
    STORAGE
     ( INITIAL  4K )
   OVERFLOW
    STORAGE
      ( INITIAL  4K );
```

外部表の例

次の文は、サンプル表hr.departmentsのサブセットを示す外部表を作成します。TYPE句は、表のアクセス・ドライバ・タイプをORACLE_LOADERに指定します。ACCESS PARAMETERS( )句は、ORACLE_LOADERアクセス・ドライバのパラメータ値

を指定します。これらのパラメータはイタリック体で表され、opaque_format_specを形成します。

opaque_format_specの構文はアクセス・ドライバ・タイプに依存するため、このドキュメントの対象外です。

ORACLE_LOADERアクセス・ドライバおよびopaque_format_spec構文の詳細は、『Oracle Databaseユーティリティ』を参照してください。

```
CREATE TABLE dept_external (
   deptno     NUMBER(6),
   dname      VARCHAR2(20),
   loc        VARCHAR2(25)
)
ORGANIZATION EXTERNAL
(TYPE oracle_loader
 DEFAULT DIRECTORY admin
 ACCESS PARAMETERS
 (
  RECORDS DELIMITED BY newline
  BADFILE 'ulcase1.bad'
  DISCARDFILE 'ulcase1.dis'
  LOGFILE 'ulcase1.log'
  SKIP 20
  FIELDS TERMINATED BY ","  OPTIONALLY ENCLOSED BY '"'
  (
   deptno     INTEGER EXTERNAL(6),
   dname      CHAR(20),
   loc        CHAR(25)
  )
 )
 LOCATION ('ulcase1.ctl')
)
REJECT LIMIT UNLIMITED;
```

関連項目:

adminディレクトリの作成方法は、「ディレクトリの作成: 例」を参照してください。

XMLTypeの例

この項では、XMLType表またはXMLType列の作成例について説明します。これらの例の詳細は、「SQL文でのXMLの使用方法」を参照してください。

XMLType表の例

次の例は、暗黙的な1つのバイナリXML列を持つ非常に単純なXMLType表を作成します。

```
CREATE TABLE xwarehouses OF XMLTYPE;
```

次の例では、XMLSchemaベースの表を作成します。XMLスキーマは事前に作成しておく必要があります(詳細は、「SQL文でのXMLの使用方法」を参照してください)。

```
CREATE TABLE xwarehouses OF XMLTYPE
   XMLSCHEMA "http://www.example.com/xwarehouses.xsd"
   ELEMENT "Warehouse";
```

XMLスキーマベースの表に制約を定義したり、索引を作成できるため、後続の問合せのパフォーマンスが大幅に向上します。

XMLType表にオブジェクト・リレーショナル・ビューを作成することも、オブジェクト・リレーショナル表にXMLTypeビューを作成することもできます。

関連項目:

- 制約の追加例は、「SQL文でのXMLの使用方法」を参照してください。
- 索引の作成例は、「XMLType表の索引の作成例:」を参照してください。
- XMLTypeビューの作成例は、「XMLTypeビューの作成: 例」を参照してください。

XMLType列の例

次の例は、CLOBとして格納されるXMLType列を持つ表を作成します。この表ではXMLスキーマが必要ないため、コンテンツ構造は事前に定義しません。

```
CREATE TABLE xwarehouses (
   warehouse_id        NUMBER,
   warehouse_spec      XMLTYPE)
   XMLTYPE warehouse_spec STORE AS CLOB
   (TABLESPACE example
    STORAGE (INITIAL 6144)
    CHUNK 4000
    NOCACHE LOGGING);
```

次の例は、同様の表を作成しますが、オブジェクト・リレーショナルXMLType列にXMLTypeデータを格納します。この列の構造は、指定したスキーマによって判断されます。

```
CREATE TABLE xwarehouses (
   warehouse_id    NUMBER,
   warehouse_spec  XMLTYPE)
   XMLTYPE warehouse_spec STORE AS OBJECT RELATIONAL
      XMLSCHEMA "http://www.example.com/xwarehouses.xsd"
      ELEMENT "Warehouse";
```

次の例では、SecureFiles CLOBとして格納されたXMLType列を持つ別の同様の表を作成します。この表ではXMLスキーマが必要ないため、コンテンツ構造は事前に定義しません。SecureFiles LOBには自動セグメント領域管理が行われる表領域が必要であるため、この例では「表領域に対するセグメント領域管理の指定: 例」で作成した表領域を使用します。

```
CREATE TABLE xwarehouses (
  warehouse_id   NUMBER,
  warehouse_spec XMLTYPE)
  XMLTYPE        warehouse_spec STORE AS SECUREFILE CLOB
  (TABLESPACE auto_seg_ts
  STORAGE (INITIAL 6144)
  CACHE);
```

パーティション化の例

レンジ・パーティション化の例

サンプル・スキーマshのsales表は、レンジでパーティション化されています。次の例では、簡略化したsales表を作成します。この例では、制約および記憶域要素は省略されています。

```
CREATE TABLE range_sales
    ( prod_id        NUMBER(6)
    , cust_id        NUMBER
    , time_id        DATE
    , channel_id     CHAR(1)
    , promo_id       NUMBER(6)
    , quantity_sold  NUMBER(3)
    , amount_sold          NUMBER(10,2)
    )
PARTITION BY RANGE (time_id)
  (PARTITION SALES_Q1_1998 VALUES LESS THAN (TO_DATE('01-APR-1998','DD-MON-YYYY')),
   PARTITION SALES_Q2_1998 VALUES LESS THAN (TO_DATE('01-JUL-1998','DD-MON-YYYY')),
```

```
   PARTITION SALES_Q3_1998 VALUES LESS THAN (TO_DATE('01-OCT-1998','DD-MON-YYYY')),
   PARTITION SALES_Q4_1998 VALUES LESS THAN (TO_DATE('01-JAN-1999','DD-MON-YYYY')),
   PARTITION SALES_Q1_1999 VALUES LESS THAN (TO_DATE('01-APR-1999','DD-MON-YYYY')),
   PARTITION SALES_Q2_1999 VALUES LESS THAN (TO_DATE('01-JUL-1999','DD-MON-YYYY')),
   PARTITION SALES_Q3_1999 VALUES LESS THAN (TO_DATE('01-OCT-1999','DD-MON-YYYY')),
   PARTITION SALES_Q4_1999 VALUES LESS THAN (TO_DATE('01-JAN-2000','DD-MON-YYYY')),
   PARTITION SALES_Q1_2000 VALUES LESS THAN (TO_DATE('01-APR-2000','DD-MON-YYYY')),
   PARTITION SALES_Q2_2000 VALUES LESS THAN (TO_DATE('01-JUL-2000','DD-MON-YYYY')),
   PARTITION SALES_Q3_2000 VALUES LESS THAN (TO_DATE('01-OCT-2000','DD-MON-YYYY')),
   PARTITION SALES_Q4_2000 VALUES LESS THAN (MAXVALUE))
;
```

パーティション表のメンテナンス操作については、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

Live SQLのレンジ・パーティション化の例

次の文は、レンジによってパーティション化された表を作成します。

```
CREATE TABLE empl_h
  (
     employee_id  NUMBER(6) PRIMARY KEY,
     first_name   VARCHAR2(20),
     last_name    VARCHAR2(25),
     email        VARCHAR2(25),
     phone_number VARCHAR2(20),
     hire_date    DATE DEFAULT SYSDATE,
     job_id       VARCHAR2(10),
     salary       NUMBER(8, 2),
     part_name    VARCHAR2(25)
  ) PARTITION BY RANGE (hire_date) (
PARTITION hire_q1 VALUES less than(to_date('01-APR-2014', 'DD-MON-YYYY')),
PARTITION hire_q2 VALUES less than(to_date('01-JUL-2014', 'DD-MON-YYYY')),
PARTITION hire_q3 VALUES less than(to_date('01-OCT-2014', 'DD-MON-YYYY')),
PARTITION hire_q4 VALUES less than(to_date('01-JAN-2015', 'DD-MON-YYYY'))
);
```

次の文は行をパーティションに挿入します。

```
INSERT INTO empl_h (employee_id, first_name, last_name, email, phone_number,
hire_date, job_id, salary, Part_name)
VALUES (1, 'Jane', 'Doe', 'example.com', '415.555.0100', '10-Feb-2014', '1001',
5001,'HIRE_Q1');
INSERT INTO empl_h (employee_id, first_name, last_name, email, phone_number,
hire_date, job_id, salary, Part_name)
VALUES (2, 'John', 'Doe', 'example.net', '415.555.0101', '10-Apr-2014', '1002',
7001,'HIRE_Q2');
INSERT INTO empl_h (employee_id, first_name, last_name, email, phone_number,
hire_date, job_id, salary, Part_name)
VALUES (3, 'Isabelle', 'Owl', 'example.org', '415.555.0102', '10-Sep-2014', '1003',
10001,'HIRE_Q3');
INSERT INTO empl_h (employee_id, first_name, last_name, email, phone_number,
hire_date, job_id, salary, Part_name)
VALUES (4, 'Smith', 'Jones', 'example.in', '415.555.0103', '10-Dec-2014', '1004',
12001,'HIRE_Q4');
```

次の文は、データ・ディクショナリ表を使用してパーティション名を表示します。

```
SELECT PARTITION_NAME FROM USER_TAB_PARTITIONS WHERE TABLE_NAME = 'EMPL_H';
PARTITION_NAME
----------------
HIRE_Q1
HIRE_Q2
HIRE_Q3
HIRE_Q4
```

```
SELECT TABLE_NAME, PARTITIONING_TYPE, STATUS FROM USER_PART_TABLES WHERE TABLE_NAME =
'EMPL_H';
TABLE_NAME      PARTITIONING_TYPE       STATUS
----------      -----------------       ------
EMPL_H          RANGE                   VALID
```

次の文は、データ・ディクショナリ表user_tab_partitionsから特定の列を選択することにより、partsという名前の表を作成します。

```
CREATE TABLE parts (p_name) AS SELECT PARTITION_NAME FROM USER_TAB_PARTITIONS WHERE
TABLE_NAME = 'EMPL_H';
```

次の文は表データを表示します。

```
select * from parts;
P_NAME
-----------
HIRE_Q1
HIRE_Q2
HIRE_Q3
HIRE_Q4
```

次の文は2つの表の列を比較し、比較に基づいて情報を表示します。

```
select E.HIRE_DATE,E.JOB_ID,P.p_name from empl_h E, parts P where E.Part_name =
P.p_name;
HIRE_DATE JOB_ID     P_NAME
--------- ---------- ------------
10-FEB-14 1001       HIRE_Q1
10-APR-14 1002       HIRE_Q2
10-SEP-14 1003       HIRE_Q3
10-DEC-14 1004       HIRE_Q4
```

時間隔パーティションの例

次の例は、credit_limit列の期間によってパーティション化されるoe.customers表を作成します。遷移ポイントを設定するために1つのレンジ・パーティションが作成されます。表内の元のデータはすべてレンジ・パーティションの境界内にあります。次に、レンジ・パーティションを超えるデータが追加され、新しい時間隔パーティションが作成されます。

```
CREATE TABLE customers_demo (
  customer_id number(6),
  cust_first_name varchar2(20),
  cust_last_name varchar2(20),
  credit_limit number(9,2))
PARTITION BY RANGE (credit_limit)
INTERVAL (1000)
(PARTITION p1 VALUES LESS THAN (5001));

INSERT INTO customers_demo
  (customer_id, cust_first_name, cust_last_name, credit_limit)
  (select customer_id, cust_first_name, cust_last_name, credit_limit
  from customers);
```

USER_TAB_PARTITIONSデータ・ディクショナリ・ビューを問い合せてから、時間隔パーティションが作成されます。

```
SELECT partition_name, high_value FROM user_tab_partitions  WHERE table_name =
'CUSTOMERS_DEMO';
PARTITION_NAME                 HIGH_VALUE
------------------------------ ---------------
P1                             5001
```

レンジ・パーティションの上限を超えるデータを表に挿入します。

```
INSERT INTO customers_demo
  VALUES (699, 'Fred', 'Flintstone', 5500);
```

挿入後にUSER_TAB_PARTITIONSビューを再度問い合せ、挿入されたデータを格納するために作成された時間隔パーティションのシステム生成名を確認します。(システム生成名はセッションごとに異なります。)

```
SELECT partition_name, high_value FROM user_tab_partitions
  WHERE table_name = 'CUSTOMERS_DEMO'
  ORDER BY partition_name;
PARTITION_NAME                 HIGH_VALUE
------------------------------ ---------------
P1                             5001
SYS_P44                        6001
```

リスト・パーティション化の例

次の文は、サンプル表oe.customersをリスト・パーティション表として作成する方法を示しています。この例では、サンプル表の一部の列およびすべての制約は省略されています。

```
CREATE TABLE list_customers
   ( customer_id             NUMBER(6)
   , cust_first_name         VARCHAR2(20)
   , cust_last_name          VARCHAR2(20)
   , cust_address            CUST_ADDRESS_TYP
   , nls_territory           VARCHAR2(30)
   , cust_email              VARCHAR2(40))
   PARTITION BY LIST (nls_territory) (
   PARTITION asia VALUES ('CHINA', 'THAILAND'),
   PARTITION europe VALUES ('GERMANY', 'ITALY', 'SWITZERLAND'),
   PARTITION west VALUES ('AMERICA'),
   PARTITION east VALUES ('INDIA'),
   PARTITION rest VALUES (DEFAULT));
```

LOB列のあるパーティション表の例

次の文は、2つのパーティションp1とp2、およびいくつかのLOB列を持つパーティション表print_media_demoを作成します。この文では、サンプル表pm.print_mediaを使用しています。

```
CREATE TABLE print_media_demo
   ( product_id NUMBER(6)
   , ad_id NUMBER(6)
   , ad_composite BLOB
   , ad_sourcetext CLOB
   , ad_finaltext CLOB
   , ad_fltextn NCLOB
   , ad_textdocs_ntab textdoc_tab
   , ad_photo BLOB
   , ad_graphic BFILE
   , ad_header adheader_typ
   ) NESTED TABLE ad_textdocs_ntab STORE AS textdocs_nestedtab_demo
      LOB (ad_composite, ad_photo, ad_finaltext)
      STORE AS(STORAGE (INITIAL 20M))
   PARTITION BY RANGE (product_id)
      (PARTITION p1 VALUES LESS THAN (3000) TABLESPACE tbs_01
         LOB (ad_composite, ad_photo)
         STORE AS (TABLESPACE tbs_02 STORAGE (INITIAL 10M))
         NESTED TABLE ad_textdocs_ntab STORE AS nt_p1 (TABLESPACE example),
       PARTITION P2 VALUES LESS THAN (MAXVALUE)
         LOB (ad_composite, ad_finaltext)
         STORE AS SECUREFILE (TABLESPACE auto_seg_ts)
         NESTED TABLE ad_textdocs_ntab STORE AS nt_p2
       )
   TABLESPACE tbs_03;
```

パーティションp1は、表領域tbs_01にあります。ad_compositeおよびad_photoに対するLOBデータのパーティションは、表領域tbs_02にあります。残りのLOB列に対するLOBデータのパーティションは、表領域tbs_01にあります。LOB列ad_compositeおよびad_photoに、記憶域属性INITIALを指定します。他の属性は、デフォルトの表レベルの指定から継承されます。表レベルで指定されていないデフォルトのLOB記憶域属性は、ad_composite列およびad_photo列については表領域tbs_02から継承されます。残りのLOB列については、表領域tbs_01から継承されます。LOB索引パーティションは、対応するLOBデータ・パーティションと同じ表領域に存在します。他の記憶域属性は、LOBデータ・パーティションの対応する属性値および索引パーティションがある表領域のデフォルト属性に基づきます。ad_textdocs_ntabのネストされた表パーティションは、nt_p1として表領域exampleに格納されます。

パーティションp2は、デフォルトの表領域tbs_03内にあります。ad_compositeおよびad_finaltextに対するLOBデータは、SecureFiles LOBとして表領域auto_seg_ts内に存在します。残りのLOB列に対するLOBデータは、表領域tbs_03にあります。ad_composite列およびad_finaltext列に対するLOB索引は、表領域auto_seg_ts内にあります。残りのLOB列に対するLOB索引は、表領域tbs_03にあります。ad_textdocs_ntabのネストされた表パーティションは、nt_p2としてデフォルト表領域tbs_03に格納されます。

ハッシュ・パーティション化の例

サンプル表oe.product_informationは、パーティション化されていません。次の例は、パフォーマンス上の理由で、このような大規模な表をハッシュでパーティション化します。この例では、表領域は仮想の名前です。

```
CREATE TABLE hash_products
    ( product_id          NUMBER(6)   PRIMARY KEY
    , product_name        VARCHAR2(50)
    , product_description VARCHAR2(2000)
    , category_id         NUMBER(2)
    , weight_class        NUMBER(1)
    , warranty_period     INTERVAL YEAR TO MONTH
    , supplier_id         NUMBER(6)
    , product_status      VARCHAR2(20)
    , list_price          NUMBER(8,2)
    , min_price           NUMBER(8,2)
    , catalog_url         VARCHAR2(50)
    , CONSTRAINT          product_status_lov_demo
                          CHECK (product_status in ('orderable'
                                                    ,'planned'
                                                    ,'under development'
                                                    ,'obsolete')
 ) )
 PARTITION BY HASH (product_id)
 PARTITIONS 4
 STORE IN (tbs_01, tbs_02, tbs_03, tbs_04);
```

参照パーティション化の例

次の文は、前述の例で作成されたhash_productsパーティション表を使用しています。hash_productsの製品IDに基づくハッシュ・パーティション化への参照によってパーティション化されたoe.order_items表を作成します。結果の子表は、5つのパーティションで作成されます。子表part_order_itemsの各行について、外部キー値(product_id)が評価され、参照されるキーが属する親表hash_productsのパーティション番号が確認されます。part_order_itemsの行は、対応するパーティションに配置されます。

```
CREATE TABLE part_order_items (
    order_id        NUMBER(12) PRIMARY KEY,
    line_item_id    NUMBER(3),
    product_id      NUMBER(6) NOT NULL,
    unit_price      NUMBER(8,2),
    quantity        NUMBER(8),
    CONSTRAINT product_id_fk
    FOREIGN KEY (product_id) REFERENCES hash_products(product_id))
```

```
 PARTITION BY REFERENCE (product_id_fk);
```

コンポジット・パーティション表の例

「レンジ・パーティション化の例」で作成された表のデータを販売時刻で分割します。時刻と販売チャネルに従って最近のデータにアクセスする場合、コンポジット・パーティション化は、より適切です。次の例では、同じrange_sales表のコピーを作成しますが、レンジ-ハッシュ・コンポジット・パーティション化を指定します。最新のデータがあるパーティションは、システム生成とユーザー定義の両方のサブパーティション名でサブパーティション化されます。この例では、制約および記憶域属性は省略されています。

```
CREATE TABLE composite_sales
    ( prod_id        NUMBER(6)
    , cust_id        NUMBER
    , time_id        DATE
    , channel_id     CHAR(1)
    , promo_id       NUMBER(6)
    , quantity_sold  NUMBER(3)
    , amount_sold         NUMBER(10,2)
    )
PARTITION BY RANGE (time_id)
SUBPARTITION BY HASH (channel_id)
  (PARTITION SALES_Q1_1998 VALUES LESS THAN (TO_DATE('01-APR-1998','DD-MON-YYYY')),
   PARTITION SALES_Q2_1998 VALUES LESS THAN (TO_DATE('01-JUL-1998','DD-MON-YYYY')),
   PARTITION SALES_Q3_1998 VALUES LESS THAN (TO_DATE('01-OCT-1998','DD-MON-YYYY')),
   PARTITION SALES_Q4_1998 VALUES LESS THAN (TO_DATE('01-JAN-1999','DD-MON-YYYY')),
   PARTITION SALES_Q1_1999 VALUES LESS THAN (TO_DATE('01-APR-1999','DD-MON-YYYY')),
   PARTITION SALES_Q2_1999 VALUES LESS THAN (TO_DATE('01-JUL-1999','DD-MON-YYYY')),
   PARTITION SALES_Q3_1999 VALUES LESS THAN (TO_DATE('01-OCT-1999','DD-MON-YYYY')),
   PARTITION SALES_Q4_1999 VALUES LESS THAN (TO_DATE('01-JAN-2000','DD-MON-YYYY')),
   PARTITION SALES_Q1_2000 VALUES LESS THAN (TO_DATE('01-APR-2000','DD-MON-YYYY')),
   PARTITION SALES_Q2_2000 VALUES LESS THAN (TO_DATE('01-JUL-2000','DD-MON-YYYY'))
      SUBPARTITIONS 8,
   PARTITION SALES_Q3_2000 VALUES LESS THAN (TO_DATE('01-OCT-2000','DD-MON-YYYY'))
     (SUBPARTITION ch_c,
      SUBPARTITION ch_i,
      SUBPARTITION ch_p,
      SUBPARTITION ch_s,
      SUBPARTITION ch_t),
   PARTITION SALES_Q4_2000 VALUES LESS THAN (MAXVALUE)
      SUBPARTITIONS 4)
;
```

次の例は、サンプル表oe.customersに基づいて、顧客のパーティション表を作成します。この例では、表はcredit_limit列でパーティション化され、nls_territory列でリスト・サブパーティション化されます。個々のサブパーティションを定義してテンプレートを上書きしないかぎり、サブパーティション・テンプレートによって、後から追加されたパーティションのサブパーティション化が決定されます。このコンポジット・パーティション化によって、指定した地域内の掛貸限度範囲に基づいて、表を問い合せることができます。

```
CREATE TABLE customers_part (
   customer_id        NUMBER(6),
   cust_first_name    VARCHAR2(20),
   cust_last_name     VARCHAR2(20),
   nls_territory      VARCHAR2(30),
   credit_limit       NUMBER(9,2))
   PARTITION BY RANGE (credit_limit)
   SUBPARTITION BY LIST (nls_territory)
      SUBPARTITION TEMPLATE
         (SUBPARTITION east  VALUES
            ('CHINA', 'JAPAN', 'INDIA', 'THAILAND'),
          SUBPARTITION west VALUES
             ('AMERICA', 'GERMANY', 'ITALY', 'SWITZERLAND'),
          SUBPARTITION other VALUES (DEFAULT))
      (PARTITION p1 VALUES LESS THAN (1000),
```

```
       PARTITION p2 VALUES LESS THAN (2500),
       PARTITION p3 VALUES LESS THAN (MAXVALUE));
```

オブジェクト列と表の例

オブジェクト表の作成: 例

オブジェクト型department_typについて考えます。

```
CREATE TYPE department_typ AS OBJECT
   ( d_name   VARCHAR2(100),
     d_address VARCHAR2(200) );
/
```

オブジェクト表departments_obj_tは、department_typ型の部門オブジェクトを保持します。

```
CREATE TABLE departments_obj_t OF department_typ;
```

次の文は、ユーザー定義オブジェクト型salesrep_typを持つオブジェクト表salesrepsを作成します。

```
CREATE OR REPLACE TYPE salesrep_typ AS OBJECT
  ( repId NUMBER,
    repName VARCHAR2(64));
CREATE TABLE salesreps OF salesrep_typ;
```

ユーザー定義のオブジェクト識別子を含む表の作成: 例

次の例は、オブジェクト型およびオブジェクト識別子が主キー・ベースの対応するオブジェクト表を作成します。

```
CREATE TYPE employees_typ AS OBJECT
   (e_no NUMBER, e_address CHAR(30));
/
CREATE TABLE employees_obj_t OF employees_typ (e_no PRIMARY KEY)
   OBJECT IDENTIFIER IS PRIMARY KEY;
```

その後、inline_ref_constraint構文またはout_of_line_ref_constraint構文のいずれかを使用して、employees_obj_tオブジェクト表を参照できます。

```
CREATE TABLE departments_t
   (d_no    NUMBER,
    mgr_ref REF employees_typ SCOPE IS employees_obj_t);
CREATE TABLE departments_t (
    d_no NUMBER,
    mgr_ref REF employees_typ
       CONSTRAINT mgr_in_emp REFERENCES employees_obj_t);
```

型列に対する制約の指定: 例

次の例は、オブジェクト型列の属性に制約を定義する方法を示しています。

```
CREATE TYPE address_t AS OBJECT
  ( hno    NUMBER,
    street VARCHAR2(40),
    city   VARCHAR2(20),
    zip    VARCHAR2(5),
    phone  VARCHAR2(10) );
/
CREATE TYPE person AS OBJECT
  ( name        VARCHAR2(40),
    dateofbirth DATE,
    homeaddress address_t,
    manager     REF person );
/
```

```
CREATE TABLE persons OF person
  ( homeaddress NOT NULL,
      UNIQUE (homeaddress.phone),
      CHECK (homeaddress.zip IS NOT NULL),
      CHECK (homeaddress.city <> 'San Francisco') );
```

CREATE TABLESPACE

目的

CREATE TABLESPACE文を使用すると、表領域を作成できます。表領域とは、スキーマ・オブジェクトを格納する、データベース内に割り当てられた領域です。

- 永続表領域には、永続スキーマ・オブジェクトが含まれます。永続表領域のオブジェクトは、データファイルに格納されます。
- UNDO表領域は、データベースを自動UNDO管理モードで実行している場合に、Oracle DatabaseがUNDOデータを管理するために使用する、一種の永続表領域です。元に戻す場合は、ロールバック・セグメントではなく、自動UNDO管理モードを使用することをお薦めします。
- 一時表領域には、セッション期間のみのスキーマ・オブジェクトが含まれます。一時表領域のオブジェクトは、一時ファイルに格納されます。

表領域は、最初は読み書き両用として作成されます。その後、ALTER TABLESPACE文でその表領域をオフラインまたはオンラインに設定したり、表領域にデータファイルや一時ファイルを追加したり、読取り専用に設定することができます。

DROP TABLESPACE文を使用して、データベースから表領域を削除することもできます。

関連項目:

- 表領域については、『Oracle Database概要』を参照してください。
- 表領域の変更および削除の詳細は、「ALTER TABLESPACE」および「DROP TABLESPACE」を参照してください。

前提条件

CREATE TABLESPACEシステム権限が必要です。SYSAUX表領域を作成する場合は、SYSDBAシステム権限が必要です。

表領域を作成する場合、表領域を格納するデータベースを作成する必要があります。また、そのデータベースをオープンしておく必要もあります。

関連項目:

CREATE DATABASE

SYSTEM以外の表領域のオブジェクトを使用するには:

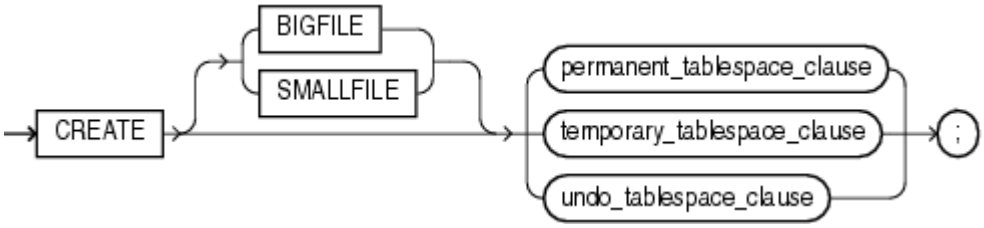
- 自動UNDO管理モードでデータベースを実行するには、1つ以上のUNDO表領域がオンラインである必要があります。
- 手動UNDO管理モードのデータベースを実行するには、SYSTEMロールバック・セグメント以外の1つ以上のロールバック・セグメントがオンラインである必要があります。

ノート:

データベースを自動 UNDO 管理モードで実行することをお薦めします。詳細は、『Oracle Database 管理者ガイド』を参照してください。

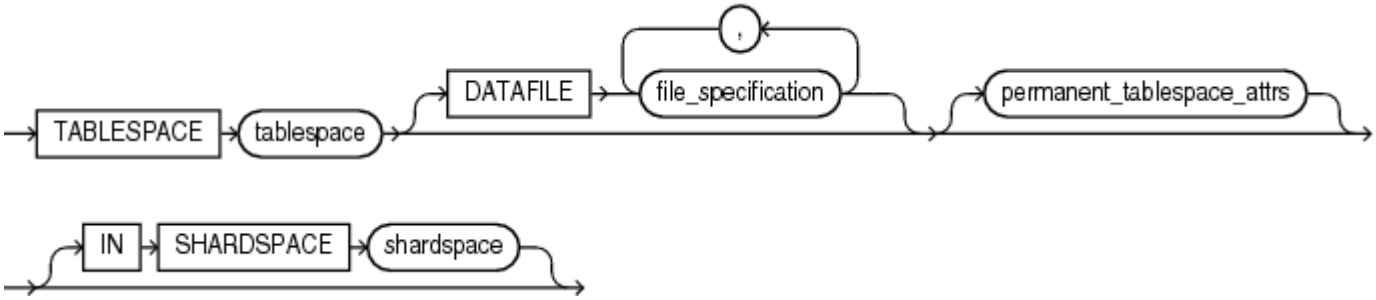
構文

create_tablespace::=



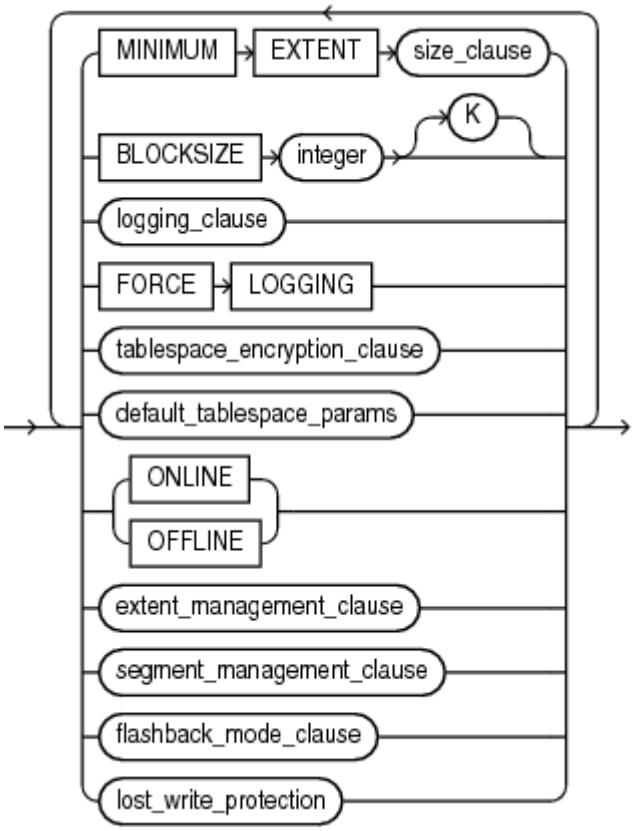
(permanent_tablespace_clause::=、temporary_tablespace_clause::=、undo_tablespace_clause::=)

permanent_tablespace_clause::=



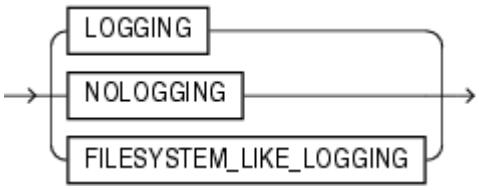
(file_specification::=、permanent_tablespace_attrs::=)

permanent_tablespace_attrs::=

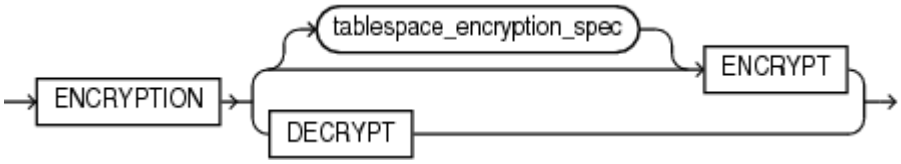


(size_clause::=、logging_clause::=、tablespace_encryption_clause::=、default_tablespace_params::=、extent_management_clause::=、segment_management_clause::=、flashback_mode_clause::=)

logging_clause::=

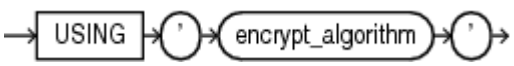


tablespace_encryption_clause::=

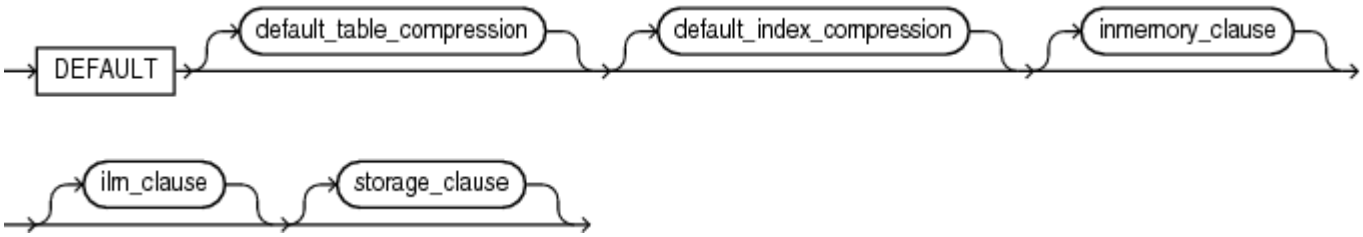


(tablespace_encryption_spec::=)

tablespace_encryption_spec::=



default_tablespace_params::=

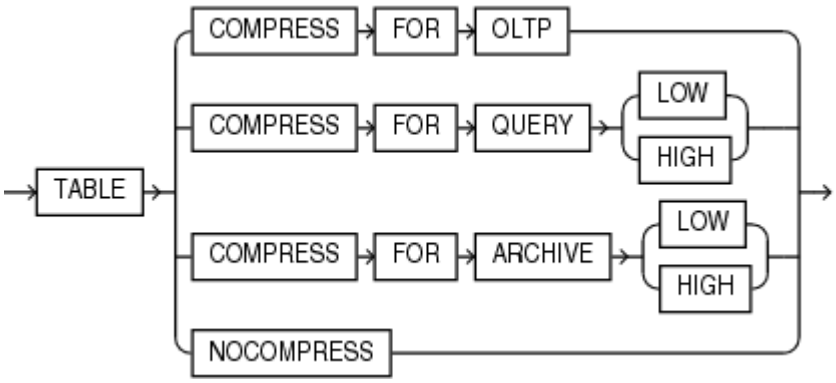


(default_table_compression::=、default_index_compression::=、inmemory_clause::=、ilm_clause::= (CREATE TABLE構文の一部)、storage_clause::=)

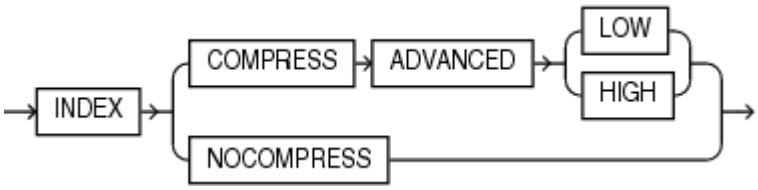
ノート:

DEFAULT 句を指定する場合、default_table_compression、default_index_compression、inmemory_clause、ilm_clause または storage_clause の各句のうち、1 つ以上を指定する必要があります。

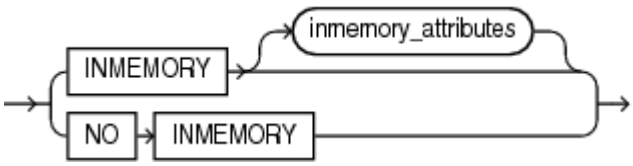
default_table_compression::=



default_index_compression::=



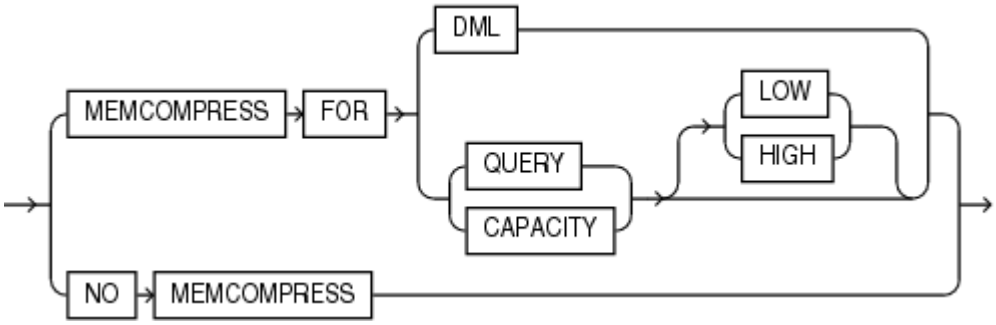
inmemory_clause::=



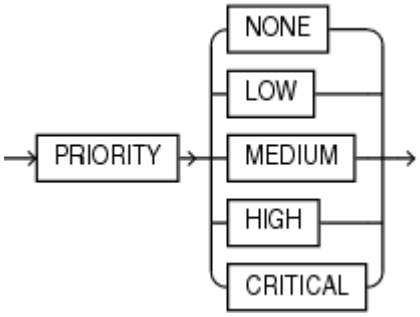
inmemory_attributes::=

(inmemory_memcompress::=, inmemory_priority::=, inmemory_distribute_tablespace::=, inmemory_duplicate::=)

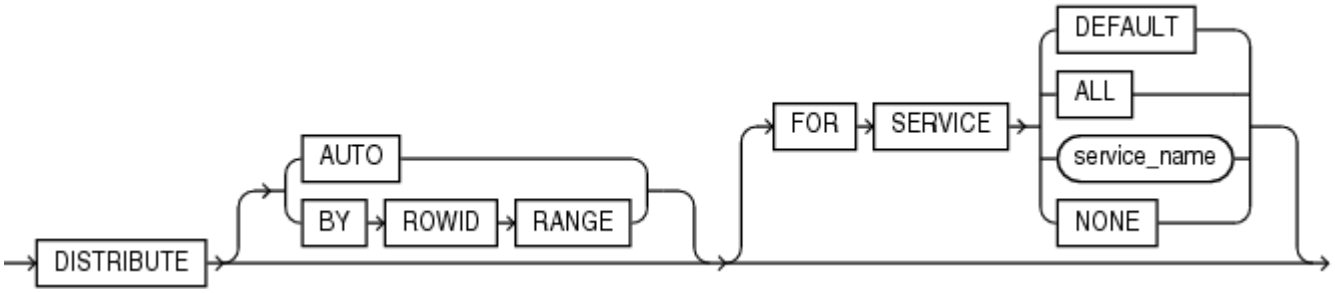
inmemory_memcompress::=



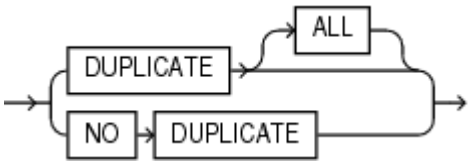
inmemory_priority::=



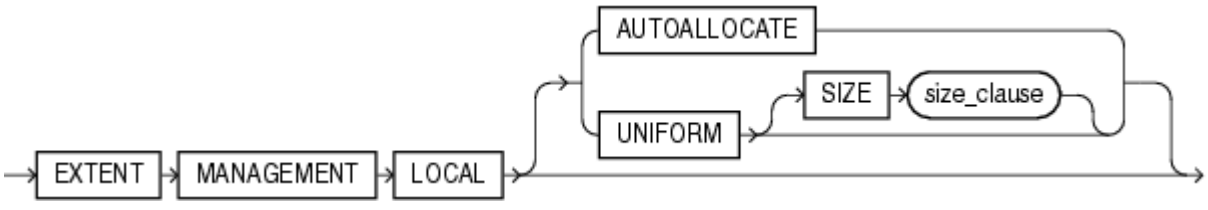
inmemory_distribute_tablespace::=



inmemory_duplicate::=



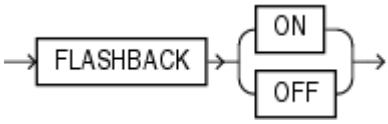
extent_management_clause::=



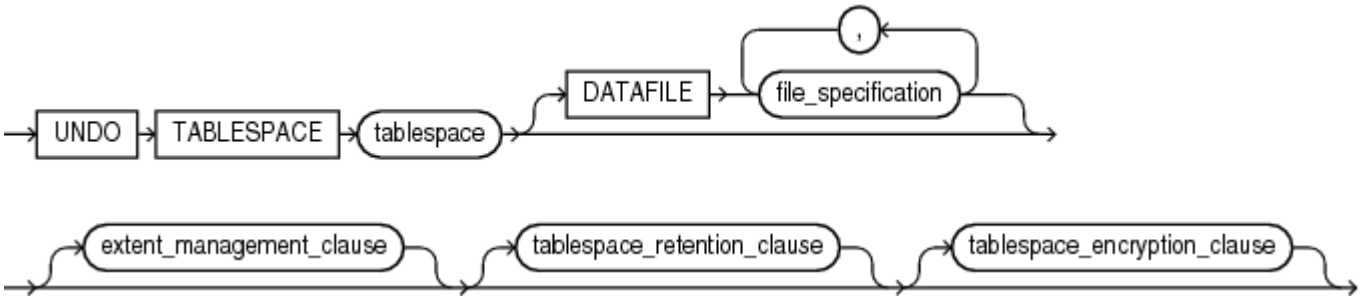
(size_clause::=)

segment_management_clause::=

flashback_mode_clause::=

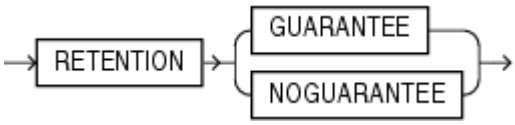


undo_tablespace_clause::=

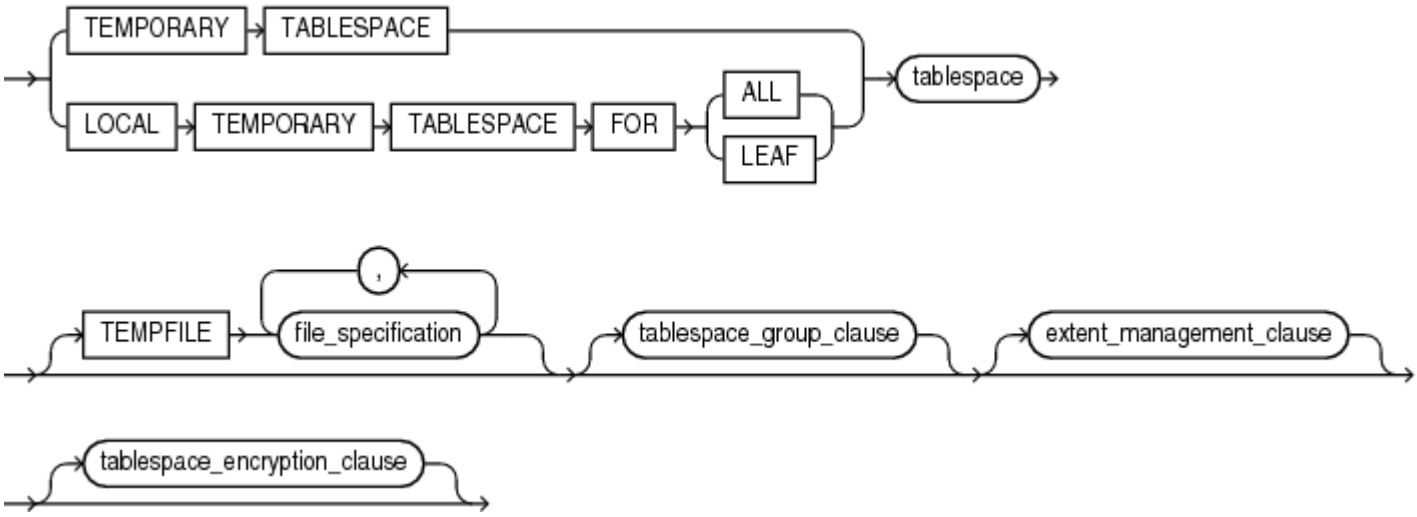


(file_specification::=、extent_management_clause::=、tablespace_retention_clause::=)

tablespace_retention_clause::=

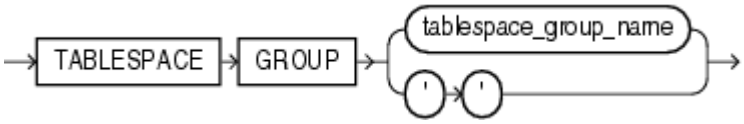


temporary_tablespace_clause::=

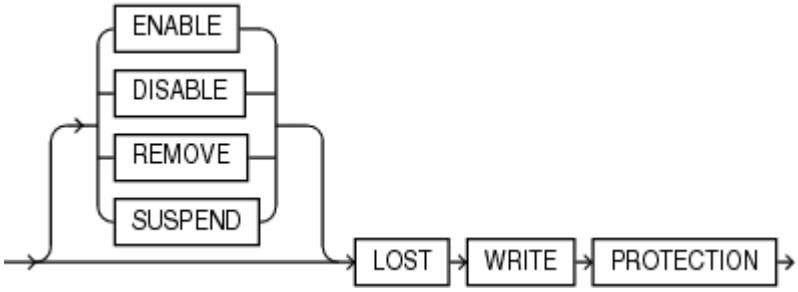


(file_specification::=、tablespace_group_clause::=、extent_management_clause::=、tablespace_encryption_clause::=)

tablespace_group_clause::=



lost_write_protection ::=



セマンティクス

BIGFILE | SMALLFILE

この句を使用すると、表領域がbigfileかsmallfileかを指定できます。この句は、データベースのデフォルトの表領域タイプの設定を上書きします。

- bigfile表領域に格納されるのは、1つのデータファイルまたは一時ファイルのみであり、このファイルには最大約40億(2^{32})ブロックを格納できます。データファイルまたは一時ファイル1つ当たりの最小サイズは、32Kブロックの表領域の場合は12MB、8Kブロックの表領域の場合は7MBです。データファイルまたは一時ファイル1つ当たりの最大サイズは、32Kブロックの表領域の場合は128TB、8Kブロックの表領域の場合は32TBです。
- smallfile表領域は、Oracleの従来の表領域であり、1022のデータファイルまたは一時ファイルを含めることができます。それぞれのファイルは、最大で約400万(2^{22})のブロックを格納できます。

この句を省略した場合、データベースに設定された永続表領域または一時表領域の、現在のデフォルト表領域タイプが使用されます。永続表領域に`BIGFILE`を指定すると、デフォルトで、自動セグメント領域管理のローカル管理表領域が作成されます。

bigfile表領域の制限事項

`DATAFILE`句に1つのみのデータファイルを指定するか、`TEMPFILE`句に1つのみの一時ファイルを指定できます。

関連項目:

- bigfile表領域の使用の詳細は、『Oracle Database管理者ガイド』を参照してください。
- bigfile表領域の作成: 例

permanent_tablespace_clause

次の句を使用すると、永続表領域を作成できます。(これらの句の一部は、一時表領域またはUNDO表領域の作成にも使用されます。)

tablespace

作成する表領域の名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たして

いる必要があります。

SYSAUX表領域のノート

SYSAUXは、必須の補助システム表領域です。Oracle Database 11gより以前のリリースからアップグレードする場合は、CREATE TABLESPACE文を使用してSYSAUX表領域を作成する必要があります。この句を指定するには、SYSDBAシステム権限が必要となり、また、データベースをUPGRADEモードでオープンしておく必要があります。

SYSAUX表領域には、EXTENT MANAGEMENT LOCALおよびSEGMENT SPACE MANAGEMENT AUTOを指定する必要があります。DATAFILE句は、Oracle Managed Filesが使用可能になっている場合のみのオプションです。DATAFILE句の動作は、「DATAFILE | TEMPFILE句」を参照してください。

SYSAUX表領域には十分な領域を割り当ててください。この表領域の作成のガイドラインは、『Oracle Databaseアップグレード・ガイド』を参照してください。

SYSAUX表領域の制限事項

SYSAUX表領域には、OFFLINEまたはTEMPORARYを指定できません。

DATAFILE | TEMPFILE句

永続表領域を構成するデータファイルを指定するか、または一時表領域を構成する一時ファイルを指定します。オペレーティング・システムのファイル・システム内の標準データファイルと一時ファイル、またはOracle Automatic Storage Management (Oracle ASM)ディスク・グループのファイルを作成するには、file_specificationのdatafile_tempfile_spec書式を使用します。

DB_CREATE_FILE_DEST初期化パラメータに値を設定してOracle Managed Filesを使用可能にしていないかぎり、DATAFILEまたはTEMPFILE句を指定する必要があります。Oracle ASMディスク・グループ・ファイルの場合、このパラメータをOracle ASMファイル名の複数ファイル作成形式に設定する必要があります。このパラメータが設定されている場合、パラメータで指定したデフォルトのファイルの出力先に、100MBのファイルがシステム名で作成されます。このファイルではAUTOEXTENDが有効になっているため、最大サイズの制限はありません。

ノート:

メディア・リカバリは一時ファイルを認識しません。

関連項目:

- Oracle ASMの使用方法の詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。
- AUTOEXTENDパラメータやOracle ASMファイル名の複数ファイル作成形式などの詳細は、「file_specification」 を参照してください。

データファイルおよび一時ファイルの指定のノート:

- datafile_tempfile_specでディスク・グループ名のみを指定すると、Oracle ASMディスク・グループ内に表領域を作成できます。この場合、Oracle ASMにより指定したディスク・グループにデータファイルが作成され、システム生成のファイル名が付けられます。データファイルは無制限に自動拡張可能です。デフォルト・サイズは100MBです。autoextend_clauseを使用すると、デフォルト・サイズを上書きできます。

- 既存のファイルを参照するASM_filenameの参照形式の1つを使用する場合、REUSEも指定する必要があります。

ノート:

オペレーティング・システムによっては、一時ファイルのブロックが実際にアクセスされるまで、一時ファイル用の領域が割り当てられない場合があります。領域の割当ての遅延のため、一時ファイルの作成およびサイズ変更が速くなります。ただし、後で一時ファイルが使用されるときに、十分なディスク領域を使用可能にする必要があります。問題を回避するには、一時ファイルの作成またはサイズ変更の前に、ディスクの空き領域が、新しく作成する一時ファイルまたはサイズ変更後の一時ファイルのサイズよりも大きいことを確認してください。ディスク領域に余裕を持たせておくと、関連のない操作による、予期されるディスク使用量の増加にも対応できます。その後で、作成またはサイズ変更操作を実行してください。

関連項目:

- AUTOEXTENDパラメータなどの詳細は、「file_specification」 を参照してください。
- 「表領域の自動拡張の有効化: 例」および「Oracle Managed Filesの作成: 例」を参照してください。

permanent_tablespace_attrs

permanent_tablespace_attrs句を使用して、表領域の属性を設定します。

MINIMUM EXTENT句

この句は、ディクショナリ管理表領域にのみ有効です。表領域で使用されるエクステントの最小サイズを指定します。この句を指定すると、表領域内のすべての使用済エクステントまたは未使用エクステントのサイズが、size_clauseで指定された値以上であること、およびその倍数であることが保証され、表領域における空き領域の断片化を制御できます。

関連項目:

この句の詳細は、「size_clause」を参照してください。MINIMUM EXTENTを使用した断片化の制御については、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください

BLOCKSIZE句

表領域の非標準ブロック・サイズを指定するには、BLOCKSIZE句を使用します。この句を指定するには、DB_CACHE_SIZEと少なくとも1つのDB_nK_CACHE_SIZEパラメータを設定する必要があります。また、この句で指定する整数は、DB_nK_CACHE_SIZEパラメータの1つの設定と対応している必要があります。

BLOCKSIZEの制限事項

一時表領域の場合、またはこの表領域を一時表領域として任意のユーザーに割り当てる場合は、標準以外のブロック・サイズを指定できません。

ノート:

パフォーマンスが低下する可能性があるため、セクター・サイズが 4K のディスクにはブロック・サイズが 2K の表領域を格納しないことをお薦めします。

関連項目:

DB_nK_CACHE_SIZEパラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。マルチ・ブロック・サイズの詳細は、『Oracle Database概要』を参照してください。

logging_clause

表領域内のすべての表、索引、マテリアライズド・ビュー、マテリアライズド・ビュー・ログおよびパーティションのデフォルトのロギング属性を指定します。この句は、一時表領域またはUNDO表領域では無効です。

この句を指定しない場合、LOGGINGがデフォルトになります。例外は、PDBの表領域の作成です。この場合、この句を省略すると、表領域はPDBのロギング属性を使用します。詳細は、CREATE PLUGGABLE DATABASEのlogging_clauseを参照してください。

表レベル、索引レベル、マテリアライズド・ビュー・レベル、マテリアライズド・ビュー・ログ・レベル、パーティション・レベルでのロギングを指定することで、表領域レベルのロギング属性を上書きできます。

関連項目:

この句の詳細は、「logging_clause」を参照してください。

FORCE LOGGING

この句を使用すると、表領域はFORCE LOGGINGモードになります。一時セグメントへの変更を除き、表領域内のすべてのオブジェクトに対するすべての変更が記録され、個々のオブジェクトのNOLOGGING設定が上書きされます。データベースをオープンし、READ WRITEモードにしておく必要があります。

この設定によって、NOLOGGING属性が削除されることはありません。FORCE LOGGINGとNOLOGGINGの両方を指定できます。この場合、後から表領域に作成されるオブジェクトのデフォルト・ロギング・モードはNOLOGGINGになりますが、表領域またはデータベースがFORCE LOGGINGモードの場合、このデフォルト値は無視されます。この表領域でFORCE LOGGINGモードを無効にすると、デフォルト値であるNOLOGGINGが再度適用されます。

ノート:

FORCE LOGGING モードは、パフォーマンスに影響する場合があります。この設定を使用する状況の詳細は、『Oracle Database 管理者ガイド』を参照してください。

強制ロギングの制限事項

FORCE LOGGINGは、UNDO表領域および一時表領域に指定できません。

tablespace_encryption_clause

この句を使用して、暗号化された表領域を作成するか暗号化されていない表領域を作成するかを指定します。暗号化された表領域を作成すると、表領域のすべてのデータファイルに透過的データ暗号化(TDE)が適用されます。

ENCRYPT | DECRYPT

ENCRYPTを指定して、暗号化された表領域を作成します。DECRYPTを指定して、暗号化されていない表領域を作成します。

この句を省略した場合、ENCRYPT_NEW_TABLESPACES初期化パラメータの値によって、作成時に表領域を暗号化するかどうかが決定されます。ENCRYPT_NEW_TABLESPACES初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

この句を発行する前に、TDEマスター・キーをデータベース・メモリーにロードするか、HSMへの接続を確立しておく必要があります。詳細は、「ADMINISTER KEY MANAGEMENT」の「open_keystore」句または「ALTER SYSTEM」の「SET ENCRYPTION WALLET句」を参照してください。

tablespace_encryption_spec

USING 'encrypt_algorithm'を指定すると、使用する暗号化アルゴリズムの名前を指定できます。有効なアルゴリズムは、AES256、AES192、AES128および3DES168です。COMPATIBLE初期化パラメータが12.2以上に設定されている場合は、ARIA128、ARIA192、ARIA256、GOST256およびSEED128の各アルゴリズムも有効です。この句を省略すると、AES128が使用されます。

関連項目:

暗号化された表領域の作成: 例

default_tablespace_params

DEFAULT句を使用すると、表領域のデフォルト・パラメータを指定できます。

default_table_compression

この句を使用して、表領域に作成されるすべての表のデータのデフォルト圧縮を指定します。この句は、一時表領域では無効です。この句の副次句のセマンティクスは、CREATE TABLE文のtable_compression句と同じですが、次の例外があります。ここでのCOMPRESS FOR OLTP句は、CREATE TABLEのROW STORE COMPRESS ADVANCED句と同等です。これらの副次句のセマンティクスの詳細は、「CREATE TABLE」の「table_compression」句を参照してください。

default_index_compression

この句を使用して、表領域に作成されるすべての索引のデータのデフォルト圧縮を指定します。この句は、一時表領域では無効です。この句の副次句のセマンティクスは、CREATE INDEX文のadvanced_index_compression句と同じです。これらの副次句のセマンティクスの詳細は、「CREATE INDEX」の「advanced_index_compression」句を参照してください。

inmemory_clause

inmemory_clauseを使用して、表領域で作成されたすべての表およびマテリアライズド・ビューのデフォルトのインメモリー列ストア(IM列ストア)設定を指定します。この句は、一時表領域では無効です。

- INMEMORYを指定して、IM列ストアのすべての表およびマテリアライズド・ビューを有効化します。

  オプションでinmemory_attributes句を使用して、表またはマテリアライズド・ビューのデータをIM列ストアに格納する方法を指定できます。inmemory_attributes句は、CREATE TABLEおよびCREATE TABLESPACEと同じセマンティクスを持ちます。この句のセマンティクスの詳細は、CREATE TABLEのinmemory_attributes句を参照してください。

- NO INMEMORYを指定して、IM列ストアのすべての表およびマテリアライズド・ビューを無効化します。これはデフォルト

です。

ilm_clause

ilm_clauseを使用して、表領域で作成されたすべての表のデフォルトの自動データ最適化設定を指定します。この句は、一時表領域では無効です。この句のセマンティクスの詳細は、CREATE TABLEのilm_clauseを参照してください。

storage_clause

storage_clauseを使用して、表領域で作成されるすべてのオブジェクトの記憶域パラメータを指定します。この句は、一時表領域またはローカル管理の表領域に対しては無効です。ディクショナリ管理表領域の場合、この句ではENCRYPT、INITIAL、NEXT、MINEXTENTS、MAXEXTENTS、MAXSIZEおよびPCTINCREASE記憶域パラメータを指定できます。詳細は、「storage_clause」を参照してください。

ノート:

下位互換性のために storage_clause の ENCRYPT 句がサポートされています。ただし、Oracle Database 12c リリース 2 (12.2)以降は、かわりに tablespace_encryption_clause で ENCRYPT を指定できます。詳細は、tablespace_encryption_clause を参照してください。

関連項目:

基本的な表領域の作成: 例

ONLINE | OFFLINE句

この句を使用すると、表領域がオンラインまたはオフラインのいずれであるかを決定できます。この句は、一時表領域では無効です。

ONLINE

ONLINEを指定すると、表領域に対するアクセス権限を付与されているユーザーに対して、作成直後の表領域を使用可能にできます。これはデフォルトです。

OFFLINE

OFFLINEを指定すると、作成直後の表領域を使用禁止にできます。

データ・ディクショナリ・ビューDBA_TABLESPACESは、各表領域がオンラインまたはオフラインのいずれであるかを示します。

extent_management_clause

extent_management_clauseを使用すると、表領域のエクステントの管理方法を指定できます。

ノート:

この句でエクステントの管理を指定した後は、表領域を移行しないかぎり、エクステントの管理を変更できません。

- AUTOALLOCATEを指定すると、表領域がシステム管理されます。ユーザーはエクステント・サイズを指定できません。AUTOALLOCATEは、一時表領域に対して指定できません。

- UNIFORMを指定すると、表領域をSIZEバイトの均一のエクステントで管理できます。SIZEのデフォルト値は1MBです。一時表領域のすべてのエクステントはサイズが均一であるため、このキーワードは一時表領域ではオプションです。ただし、SIZEを指定する場合は、UNIFORMを指定する必要があります。UNDO表領域にUNIFORMを指定することはできません。

AUTOALLOCATEまたはUNIFORMを指定しない場合のデフォルトは、一時表領域ではUNIFORM、他のすべてのタイプの表領域ではAUTOALLOCATEです。

extent_management_clauseを指定しない場合、MINIMUM EXTENT句およびDEFAULT storage_clauseが解析され、エクステント管理が判断されます。

ノート:

DICTIONARY キーワードは、非推奨になる予定です。これは、下位互換性を保つためにのみサポートされています。ただし、ローカル管理表領域を作成することをお薦めします。ローカル管理表領域は、ディクショナリ管理表領域より非常に効率的に管理されます。新しいディクショナリ管理表領域の作成は、サポートされなくなる予定です。

関連項目:

ローカル管理表領域については、『Oracle Database概要』を参照してください。

エクステント管理の制限事項

エクステント管理には、次の制限事項があります。

- 永続的なローカル管理表領域には、永続オブジェクトのみを格納できます。ローカル管理表領域に一時オブジェクトを格納する必要がある場合(たとえば、ユーザーの一時表領域に割り当てる場合)は、temporary_tablespace_clauseを使用します。
- この句を指定する場合、DEFAULT storage_clause、MINIMUM EXTENTまたはtemporary_tablespace_clauseは指定できません。

関連項目:

表領域の移行によってエクステントの管理を変更する方法の詳細は、『Oracle Database管理者ガイド』および「ローカル管理表領域の作成: 例」を参照してください。

segment_management_clause

segment_management_clauseは、永続的なローカル管理表領域に対してのみ有効です。Oracle Databaseが、空きリストまたはビットマップのどちらを使用して、表領域のセグメントにある使用済領域および空き領域を追跡するかを指定できます。この句は、一時表領域では無効です。

AUTO

AUTOを指定すると、データベースでビットマップを使用して表領域のセグメントの空き領域を管理できます。AUTOを指定すると、この表領域のオブジェクトに対して後で指定する記憶域のPCTUSED、FREELISTおよびFREELIST GROUPSの値は無視されます。この設定を自動セグメント領域管理といい、これがデフォルトです。

MANUAL

MANUALを指定すると、データベースで空きリストを使用して表領域のセグメントの空き領域を管理できます。この設定は使用しないようにして、自動セグメント領域の表領域を作成することを強くお薦めします。

既存の表領域のセグメント管理を確認するには、DBA_TABLESPACESまたはUSER_TABLESPACESデータ・ディクショナリ・ビューのSEGMENT_SPACE_MANAGEMENT列を問い合せます。

ノート:

AUTO セグメント管理を指定する場合、次のことに注意します。

- エクステント管理を LOCAL UNIFORM に設定する場合は、各エクステントに 5 以上のデータベース・ブロックがあることを確認する必要があります。
- エクステント管理を LOCAL AUTOALLOCATE に設定する場合、およびデータベースのブロック・サイズが 16KB 以上の場合は、最小で 5 ブロック(64KB に切り上げられる)のエクステントが作成され、セグメント領域管理が管理されます。

自動セグメント領域管理の制限事項

この句には、次の制限事項があります。

- この句は、永続的なローカル管理表領域に対してのみ指定できます。
- この句は、SYSTEM表領域に対して指定できません。

関連項目:

- 自動セグメント領域管理およびその使用については、『Oracle Automatic Storage Management管理者ガイド』を参照してください。
- データ・ディクショナリ・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。
- 表領域に対するセグメント領域管理の指定: 例

flashback_mode_clause

この句をALTER DATABASE FLASHBACK句と組み合せて使用すると、FLASHBACK DATABASE操作に表領域を使用できるかどうかを指定できます。この句は、データベースがFLASHBACKモードでオープンされているときに、この表領域のフラッシュバック・ログ・データを保持しない場合に便利です。

この句は、一時表領域またはUNDO表領域では無効です。

FLASHBACK ON

FLASHBACK ONを指定すると、表領域でFLASHBACKモードを有効にできます。この表領域のフラッシュバック・ログ・データが保存され、FLASHBACK DATABASE操作でこの表領域を使用できるようになります。flashback_mode_clauseを指定しない場合、デフォルトでFLASHBACK ONが指定されます。

FLASHBACK OFF

FLASHBACK OFFを指定すると、表領域でFLASHBACKモードを無効にできます。この表領域のフラッシュバック・ログ・データは保存されません。FLASHBACK DATABASE操作の実行前に、この表領域のデータファイルをオフラインにするか、または削除

する必要があります。または、表領域全体をオフラインにすることもできます。どちらの場合も、既存のフラッシュバック・ログは削除されません。

ノート:

表領域の FLASHBACK モードは、個々の表の FLASHBACK モードに依存しません。

関連項目:

- Oracle Flashback Databaseの詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- データベース全体をFLASHBACKモードに設定し、データベースを以前のバージョンに戻す方法の詳細は、「ALTER DATABASE」および「FLASHBACK DATABASE」を参照してください。
- 「FLASHBACK TABLE」および「flashback_query_clause」を参照してください。

lost_write_protection

消失書込みレコードの記憶域を作成するには、`lost_write_protection`句を指定します。データファイルとデータベースに対して消失書込み保護を有効にするには、事前にこの記憶域またはシャドウ表領域を作成する必要があります。

他のすべての表領域と同様に、必要な数のシャドウ表領域を作成し、それらに名前を付けることができます。

例: データベースでのシャドウ表領域の作成

この例では、消失書込み保護のためのシャドウ表領域`sh_lwp1`を作成します。

```
CREATE BIGFILE TABLESPACE sh_lwp1 DATAFILE sh_lwp1.df SIZE 10M BLOCKSIZE 8K
  LOST WRITE PROTECTION;
```

表領域の消失書込み保護を有効化するには、`lost_write_protection`句を`ALTER TABLESPACE`で指定します。

データファイルの消失書込み保護を有効化するには、`lost_write_protection`句を`ALTER DATABASE`で指定します。

undo_tablespace_clause

`UNDO`を指定すると、UNDO表領域を作成できます。自動UNDO管理モードでデータベースを実行する場合、Oracle Databaseは、ロールバック・セグメントのかわりにUNDO表領域を使用してUNDO領域を管理します。この句は、自動UNDO管理モードで作成しなかったデータベースを自動UNDO管理モードで実行している場合に便利です。

自動UNDO管理モードでデータベースを起動すると、UNDO表領域が割り当てられます。UNDO表領域がインスタンスに割り当てられない場合、`SYSTEM`ロールバック・セグメントが使用されます。UNDO表領域の作成によってこれを回避することができ、他のUNDO表領域がその時点で割り当てられていない場合、インスタンスに暗黙的に割り当てられます。

`DATAFILE`句の詳細は、「DATAFILE | TEMPFILE句」を参照してください。

extent_management_clause

UNDO表領域を作成するときには`extent_management_clause`を指定する必要はありません。これは、UNDO表領域が、`AUTOALLOCATE`エクステント管理を使用するローカル管理表領域である必要があるためです。この句を指定する場合は、`EXTENT MANAGEMENT LOCAL`または`EXTENT MANAGEMENT LOCAL AUTOALLOCATE`を指定する必要がありま

す。どちらも、この句を省略した場合と同じ結果になります。この句のセマンティクスの詳細は、「extent_management_clause」を参照してください。

tablespace_retention_clause

この句は、UNDO表領域に対してのみ有効です。

- RETENTION GUARANTEEを指定すると、tablespaceのすべてのUNDOセグメントの期限切れ前のUNDOデータが、これらのセグメントのUNDO領域を必要とする実行中の操作が失敗する場合でも保持されます。この設定は、Oracleフラッシュバック問合せまたはOracleフラッシュバック・トランザクション問合せを発行し、データの問題を診断および修正する必要がある場合に便利です。
- RETENTION NOGUARANTEEを指定すると、UNDO動作を通常の動作に戻すことができます。実行中のトランザクションで必要な場合には、期限切れ前のUNDOデータによって使用されているUNDOセグメントの領域を使用できます。これはデフォルトです。

tablespace_encryption_clause

この句のUNDO表領域のセマンティクスは、永続表領域と同じです。詳細は、永続表領域のドキュメントで「tablespace_encryption_clause」を参照してください。

UNDO表領域の制限事項

UNDO表領域には、次の制限事項があります。

- この表領域にはデータベース・オブジェクトを作成できません。システム管理のUNDOデータ用に確保されています。
- ローカルのAUTOALLOCATEエクステント管理を指定するためにUNDO表領域に対して指定できる句は、DATAFILE句、tablespace_retention_clause、tablespace_encryption_clauseおよびextent_management_clauseのみです。extent_management_clauseを使用した、ローカルのUNIFORMエクステント管理またはディクショナリ・エクステント管理の指定はいずれもできません。すべてのUNDO表領域は、永続的、読取り/書込み可能およびロギング・モードで作成されます。MINIMUM EXTENTおよびDEFAULT STORAGEに対する値は、システムで生成されます。

関連項目:

- 自動UNDO管理およびUNDO表領域の詳細は、『Oracle Database管理者ガイド』を参照してください。UNDO_MANAGEMENTパラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。
- データベースの作成時にUNDO表領域を作成する方法は、「CREATE DATABASE」を参照してください。また、「ALTER TABLESPACE」および「DROP TABLESPACE」を参照してください。
- UNDO表領域の作成：例

temporary_tablespace_clause

この句を使用すると、一時表領域を作成できます。一時表領域は、セッションの存続期間中にのみ保持される一時データを格納できるデータベース内の領域割当てです。プロセスまたはインスタンスに障害が発生した場合、この一時データをリカバリすることはできません。

一時データとは、一時表などのユーザー生成スキーマ・オブジェクト、またはハッシュ結合およびソート操作で使用される一時領域などのシステム生成データです。一時表領域、またはこの表領域が含まれる表領域グループを特定のユーザーに割り当てると、このユーザーによって開始されるトランザクションでのソート操作にこの表領域が使用されます。

次の2つのタイプの一時表領域を作成できます。

- 共有一時表領域は、TEMPORARY TABLESPACE句を指定すると作成できます。共有一時表領域は共有ディスク上に一時ファイルを格納するため、一時領域はすべてのデータベース・インスタンスにとってアクセス可能です。共有一時表領域はOracle Databaseの以前のリリースでも使用可能で、「一時表領域」と呼ばれていました。このガイドの他の部分では、特に記載がないかぎり、一時表領域という用語は共有一時表領域を指します。
- ローカル一時表領域は、Oracle Database 12cリリース2 (12.2)以降、LOCAL TEMPORARY TABLESPACE句を指定すると作成できます。ローカル一時表領域は、Oracle Clusterware環境で役立ちます。これらには、各データベース・インスタンスの共有されない個別の一時ファイルが格納され、I/Oパフォーマンスが向上します。ローカル一時表領域は、BIGFILE表領域である必要があります。
    - HUBとLEAFのすべてのノードに、個別の非共有一時ファイルを作成するようデータベースに指示するには、FOR ALLを指定します。
    - LEAFノードにのみ個別の非共有一時ファイルを作成するようデータベースに指示するには、FOR LEAFを指定します。

TEMPFILE

TEMPFILE句の詳細は、「DATAFILE | TEMPFILE句」を参照してください。

tablespace_group_clause

この句は、一時表領域に対してのみ有効です。この句を使用すると、tablespaceが表領域グループに含まれるかどうかを決定できます。表領域グループを使用すると、複数の一時表領域を1人のユーザーに割り当て、一時表領域のアクセス性を向上できます。

- グループ名を指定すると、tablespaceがその表領域グループのメンバーであることを示すことができます。グループ名にtablespaceまたは他の既存の表領域と同じ名前を指定することはできません。表領域グループがすでに存在する場合、そのグループに新しい表領域が追加されます。表領域グループが存在しない場合、グループが作成され、そのグループに新しい表領域が追加されます。
- 空の文字列(' ')を指定すると、tablespaceがいずれの表領域グループにも属さないことを示すことができます。

表領域グループの制限事項

表領域グループでは、共有一時表領域のみサポートされます。ローカル一時表領域は、表領域グループに追加できません。

extent_management_clause

extent_management_clauseの詳細は、「extent_management_clause」を参照してください。

tablespace_encryption_clause

この句の一時表領域のセマンティクスは、永続表領域と同じです。 詳細は、永続表領域のドキュメントで「tablespace_encryption_clause」を参照してください。

関連項目:

- 表領域グループへの表領域の追加の詳細は、「ALTER TABLESPACE」および「表領域グループへの一時表領域の追加: 例」を参照してください。
- ユーザーに対する一時表領域の割当てについては、「CREATE USER」を参照してください。

- 表領域グループの詳細は、『Oracle Database管理者ガイド』を参照してください。

一時表領域の制限事項

一時表領域に格納されているデータは、セッションの存続期間中にのみ保持されます。そのため、CREATE TABLESPACE句のサブセットのみが一時表領域に対して有効です。一時表領域に対して指定できる句は、TEMPFILE句、tablespace_group_clause、extent_management_clauseおよびtablespace_encryption_clauseのみです。

例

これらの例では、8Kブロックを使用していると想定します。

bigfile表領域の作成: 例

次の例は、20MBのデータファイルbigtbs_f1.dbfを持つbigfile表領域bigtbs_01を作成します。

```
CREATE BIGFILE TABLESPACE bigtbs_01
  DATAFILE 'bigtbs_f1.dbf'
  SIZE 20M AUTOEXTEND ON;
```

UNDO表領域の作成: 例

次の例は、10MBのUNDO表領域undots1を作成します。

```
CREATE UNDO TABLESPACE undots1
   DATAFILE 'undotbs_1a.dbf'
   SIZE 10M AUTOEXTEND ON
   RETENTION GUARANTEE;
```

一時表領域の作成: 例

次の文は、サンプル・データベースのデータベース・ユーザーに対するデフォルトの一時表領域として機能する一時表領域がどのように作成されたかを示しています。

```
CREATE TEMPORARY TABLESPACE temp_demo
   TEMPFILE 'temp01.dbf' SIZE 5M AUTOEXTEND ON;
```

デフォルトのデータベース・ブロック・サイズを2KBとした場合、マップの各ビットは1つのエクステントを表し、各ビットは2,500ブロックをマップします。

次の例は、データファイルを作成するデフォルトの位置を設定し、デフォルトの位置にOracle Managed Filesの一時ファイルを持つ表領域を作成します。一時ファイルは100MBで、最大サイズが制限なしで自動拡張されます。これらは、Oracle Managed Filesのデフォルト値です。

```
ALTER SYSTEM SET DB_CREATE_FILE_DEST = '$ORACLE_HOME/rdbms/dbs';
CREATE TEMPORARY TABLESPACE tbs_05;
```

表領域グループへの一時表領域の追加: 例

次の文は、tbs_temp_02一時表領域をtbs_grp_01表領域グループのメンバーとして作成します。この表領域グループが存在しない場合、文の実行中に作成されます。

```
CREATE TEMPORARY TABLESPACE tbs_temp_02
  TEMPFILE 'temp02.dbf' SIZE 5M AUTOEXTEND ON
  TABLESPACE GROUP tbs_grp_01;
```

基本的な表領域の作成: 例

次の文は、1つのデータファイルを持つtbs_01という表領域を作成します。

```
CREATE TABLESPACE tbs_01
   DATAFILE 'tbs_f2.dbf' SIZE 40M
   ONLINE;
```

次の文は、1つのデータファイルを持つ表領域tbs_03を作成し、すべてのエクステントを500KBの倍数として割り当てます。

```
CREATE TABLESPACE tbs_03
   DATAFILE 'tbs_f03.dbf' SIZE 20M
   LOGGING;
```

表領域の自動拡張の有効化: 例

次の文は、1つのデータファイルを持つtbs_02という表領域を作成します。さらに領域が必要な場合、500KBのエクステントが最大サイズ100MBまで追加されます。

```
CREATE TABLESPACE tbs_02
   DATAFILE 'diskb:tbs_f5.dbf' SIZE 500K REUSE
   AUTOEXTEND ON NEXT 500K MAXSIZE 100M;
```

ローカル管理表領域の作成: 例

次の文では、データベース・ブロック・サイズが2Kであると仮定します。

```
CREATE TABLESPACE tbs_04 DATAFILE 'file_1.dbf' SIZE 10M
   EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K;
```

この文で、すべてのエクステントが128KBで、ビットマップの各ビットが64ブロックを示す、ローカル管理表領域を作成します。

次の文は、均一のエクステントを持つローカル管理表領域を作成して、その表領域に格納された表の例を示します。

```
CREATE TABLESPACE lmt1 DATAFILE 'lmt_file2.dbf' SIZE 100m REUSE
  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M;
CREATE TABLE lmt_table1 (col1 NUMBER, col2 VARCHAR2(20))
  TABLESPACE lmt1 STORAGE (INITIAL 2m);
```

表の初期セグメント・サイズは2MBです。

次の例は、均一のエクステントを持たないローカル管理表領域を作成します。

```
CREATE TABLESPACE lmt2 DATAFILE 'lmt_file3.dbf' SIZE 100m REUSE
  EXTENT MANAGEMENT LOCAL;
CREATE TABLE lmt_table2 (col1 NUMBER, col2 VARCHAR2(20))
  TABLESPACE lmt2 STORAGE (INITIAL 2m MAXSIZE 100m);
```

表の初期セグメント・サイズは2MBです。Oracle Databaseによって、初期セグメント・サイズを満たすように各エクステントのサイズおよび割り当てられるエクステントの合計数が決定されます。セグメントの最大サイズは、100MBに制限されています。

暗号化された表領域の作成: 例

次の例の最初の文は、ウォレットを開くことによって、データベースに対して暗号化を可能にします。2番目の文は、暗号化された表領域を作成します。

```
ALTER SYSTEM SET ENCRYPTION WALLET OPEN IDENTIFIED BY "wallet_password";
CREATE TABLESPACE encrypt_ts
  DATAFILE '$ORACLE_HOME/dbs/encrypt_df.dbf' SIZE 1M
  ENCRYPTION USING 'AES256' ENCRYPT;
```

表領域に対するセグメント領域管理の指定: 例

次の例は、自動セグメント領域管理が指定された表領域を作成します。

```
CREATE TABLESPACE auto_seg_ts DATAFILE 'file_2.dbf' SIZE 1M
   EXTENT MANAGEMENT LOCAL
```

```
   SEGMENT SPACE MANAGEMENT AUTO;
```

Oracle Managed Filesの作成: 例

次の例は、データファイルを作成するデフォルトの位置を設定し、デフォルトの位置にデータファイルを持つ表領域を作成します。データファイルは100MBで自動拡張可能であり、最大サイズの制限がありません。

```
ALTER SYSTEM SET DB_CREATE_FILE_DEST = '$ORACLE_HOME/rdbms/dbs';
CREATE TABLESPACE omf_ts1;
```

次の例は、自動拡張されない100MBのOracle Managed Filesのデータファイルを持つ表領域を作成します。

```
CREATE TABLESPACE omf_ts2 DATAFILE AUTOEXTEND OFF;
```

CREATE TABLESPACE SET

ノート:

この SQL 文は、Oracle Sharding を使用している場合にのみ有効です。Oracle Sharding の詳細は、『Oracle Database 管理者ガイド』を参照してください。

目的

CREATE TABLESPACE SET文を使用して、表領域セットを作成します。表領域セットは、1つ以上のシャード表および索引の論理的な記憶域単位として、シャード・データベースで使用できます。

表領域セットは、シャード領域内のシャード間で分散される複数の表領域で構成されます。表領域は、データベースによって表領域セットとして自動的に作成されます。表領域の数は自動的に決定され、対応するシャード領域のチャンクの数と等しくなります。

表領域セットのすべての表領域は、永続bigfile表領域です。表領域セットには、SYSTEM表領域、UNDO表領域および一時表領域は含まれません。データベースは、各表領域に1つのデータファイルを自動的に作成します。表領域セットのすべての表領域は、同じ属性を共有します。表領域セットのすべての表領域の属性を変更するには、ALTER TABLESPACE SET文を使用します。

関連項目:

「ALTER TABLESPACE SET」および「DROP TABLESPACE SET」を参照してください。

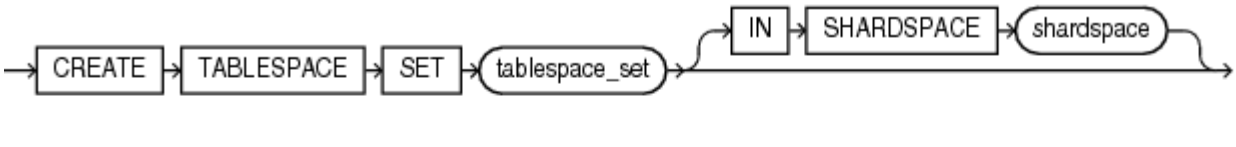
前提条件

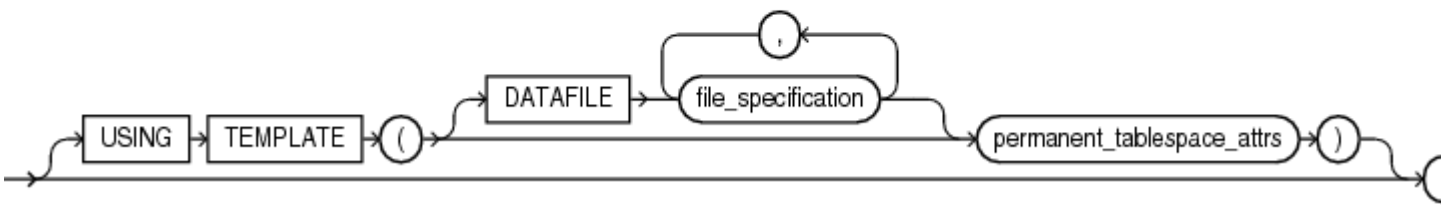
シャード・カタログ・データベースにSDBユーザーとして接続している必要があります。

CREATE TABLESPACEシステム権限が必要です。

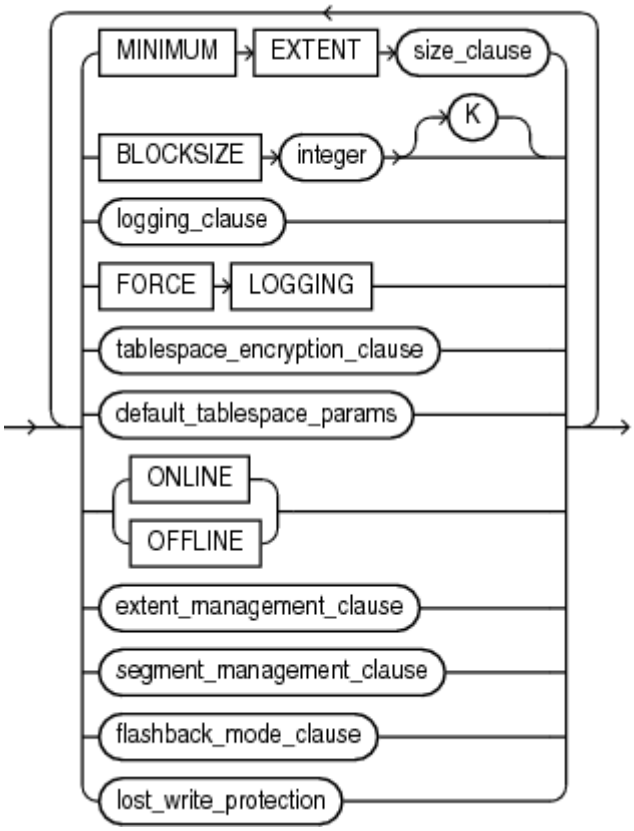
構文

create_tablespace_set::=





permanent_tablespace_attrs::=



(file_specification::=。CREATE TABLESPACEの句(logging_clause::=、tablespace_encryption_clause::=、default_tablespace_params::=、extent_management_clause::=、segment_management_clause::=、flashback_mode_clause::=)を参照)

セマンティクス

tablespace_set

作成する表領域セットの名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

IN SHARDSPACE

複合シャーディングを使用している場合は、この句を指定します。shardspace_nameには、表領域セットを作成するシャード領域の名前を指定します。

システム管理シャーディングを使用している場合は、この句を省略します。この場合、表領域セットは、シャード・データベースのデフォルトのシャード領域に作成されます。

USING TEMPLATE

USING TEMPLATE句を使用すると、表領域セット内の表領域の属性を指定できます。

DATAFILEおよびpermanent_tablespace_attrs句のセマンティクスは、CREATE TABLESPACE文と同じですが、次の例外があります。

- DATAFILE file_specification句に指定できるのは、SIZE句およびautoextend_clauseのみです。
- MINIMUM EXTENT size_clauseは指定できません。
- segment_management_clauseに指定できるのは、SEGMENT SPACE MANAGEMENT AUTOのみです。

MANUAL設定はサポートされません。

関連項目:

これらの句のセマンティクスの詳細は、CREATE TABLESPACEのドキュメントの「file_specification」および「permanent_tablespace_attrs」を参照してください。

例

表領域セットの作成: 例

次の文は、表領域セットts1を作成します。

```
CREATE TABLESPACE SET ts1
  IN SHARDSPACE sgr1
  USING TEMPLATE
  ( DATAFILE SIZE 100m
    EXTENT MANAGEMENT LOCAL
    SEGMENT SPACE MANAGEMENT AUTO
  );
```

CREATE TRIGGER

目的

トリガーはPL/SQLを使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

CREATE TRIGGER文を使用すると、データベース・トリガーを作成できます。データベース・トリガーとは、次のとおりです。

- 表、スキーマまたはデータベースに対応したストアドPL/SQLブロック
- 無名PL/SQLブロック、あるいはPL/SQLまたはJAVAで実装されているプロシージャへのコール

Oracle Databaseでは、指定した条件が発生するとトリガーは自動的に実行されます。

関連項目:

「ALTER TRIGGER」および「DROP TRIGGER」を参照してください。

前提条件

自分のスキーマ内の表または自分のスキーマ(SCHEMA)に対するトリガーを自分のスキーマ内に作成する場合は、CREATE TRIGGERシステム権限が必要です。

任意のスキーマ内の表または別のユーザーのスキーマ(schema.SCHEMA)に対するトリガーを任意のスキーマ内に作成する場合は、CREATE ANY TRIGGERシステム権限が必要です。

前述の権限に加えて、DATABASEに対するトリガーを作成する場合は、ADMINISTER DATABASE TRIGGERシステム権限が必要です。

プラガブル・データベース(PDB)に対するトリガーを作成する場合は、現在のコンテナがそのPDBである必要があります。また、ADMINISTER DATABASE TRIGGERシステム権限が必要です。PDBの詳細は、『Oracle Database管理者ガイド』を参照してください。

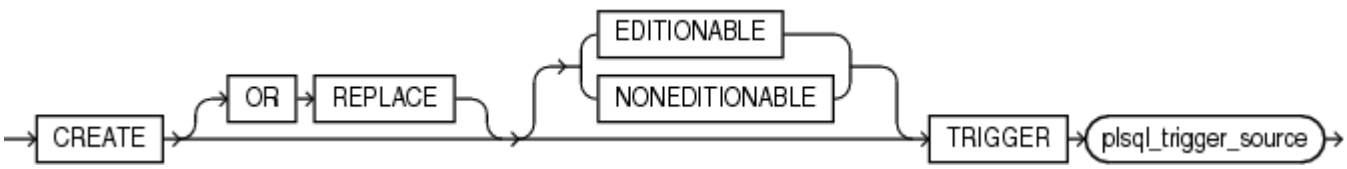
前述の権限に加えて、crosseditionトリガーを作成する場合は、エディションが有効になっている必要があります。ユーザーに対するエディションの有効化の詳細は、『Oracle Database開発ガイド』を参照してください。

トリガーがSQL文を発行、またはプロシージャやファンクションをコールする場合、そのトリガーの所有者には、これらの操作を行うための権限が必要です。これらの権限は、ロールを介して付与するのではなく、所有者に直接付与する必要があります。

構文

トリガーはPL/SQLを使用して定義されます。このため、このマニュアルの構文図ではSQLキーワードのみを示します。PL/SQLの構文、セマンティクスおよび例については、『Oracle Database PL/SQL言語リファレンス』を参照してください。

create_trigger::=



(plsql_trigger_source: 『Oracle Database PL/SQL言語リファレンス』を参照。)

セマンティクス

OR REPLACE

OR REPLACEを指定すると、既存のトリガーを再作成できます。この句を指定すると、既存のトリガーの定義を削除しなくても変更できます。

[EDITIONABLE | NONEDITIONABLE]

この句を使用すると、schemaのスキーマ・オブジェクト・タイプTRIGGERのエディションニングが有効化されたときに、そのトリガーをエディション・オブジェクトにするか非エディション・オブジェクトにするかを指定できます。デフォルトは、EDITIONABLEです。エディション・オブジェクトと非エディション・オブジェクトの詳細は、『Oracle Database開発ガイド』を参照してください。

NONEDITIONABLEの制限事項

crossedition トリガーに対してNONEDITIONABLEを指定できません。

plsql_trigger_source

plsql_trigger_sourceの構文およびセマンティクスについては、『Oracle Database PL/SQL言語リファレンス』を参照してください。

CREATE TYPE

目的

オブジェクト型はPL/SQLを使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

CREATE TYPE文を使用すると、オブジェクト型、SQLJオブジェクト型、名前付きの可変配列(VARRAY)、ネストした表型または不完全なオブジェクト型の仕様部を作成できます。CREATE TYPE文およびCREATE TYPE BODY文を使用してオブジェクト型を作成します。CREATE TYPE文では、オブジェクト型の名前、オブジェクトの属性、メソッドおよびその他のプロパティを指定します。CREATE TYPE BODY文には、その型を実装するメソッドに対するコードが含まれます。

ノート:

- 型仕様で属性のみ宣言し、メソッドは宣言しないオブジェクト型を作成する場合は、型本体を指定する必要はありません。
- SQLJ オブジェクト型を作成する場合は、型本体を指定できません。型の実装は Java クラスとして指定されます。

不完全型とは、フォワード型定義によって作成される型です。このオブジェクト型には名前はありますが、属性およびメソッドがないため、不完全といわれます。他の型からの参照が可能なため、互いに参照する型の定義に使用できます。ただし、不完全オブジェクト型を使用して表やオブジェクト列またはネストした表型の列を作成する場合は、型を完全に指定しておく必要があります。

関連項目:

- 型のメンバー・メソッドの作成については、「CREATE TYPE BODY」を参照してください。
- オブジェクト、不完全型、VARRAYおよびネストした表の詳細は、『Oracle Databaseオブジェクト・リレーショナル開発者ガイド』を参照してください。

前提条件

自分のスキーマ内に型を作成する場合は、CREATE TYPEシステム権限が必要です。他のユーザーのスキーマ内に型を作成する場合は、CREATE ANY TYPEシステム権限が必要です。これらの権限は、明示的に取得することもロールを介して取得することもできます。

サブタイプを作成するには、UNDER ANY TYPEシステム権限またはスーパータイプに対するUNDERオブジェクト権限が必要です。

型の所有者には、型定義内で参照する他のすべての型にアクセスするためのEXECUTEオブジェクト権限が明示的に付与されている必要があります。または、EXECUTE ANY TYPEシステム権限が付与されている必要があります。所有者は、これらの権限をロールを介して取得することはできません。

型の所有者が、型にアクセスする権限を他のユーザーに付与する場合、所有者には、参照型に対するGRANT OPTION付きのEXECUTEオブジェクト権限、またはADMIN OPTION付きのEXECUTE ANY TYPEシステム権限が必要です。これらの権限がない場合、型の所有者は、型にアクセスする権限を他のユーザーに付与できません。

永続不可のデータ型として宣言されるユーザー定義のデータ型

データ型の作成時に、ユーザー定義のデータ型を永続不可として指定できます。永続不可型のインスタンスは、ディスク上で保持することはできません。永続可能なデータ型は次のとおりです。

- ANSI対応データ型(NUMERIC、DECIMAL、REALなど)。
- Oracleの組込みデータ型(NUMBER、VARCHAR2、TIMESTAMPなど)。
- Oracleが提供するデータ型(ANYDATA、XML Type、ORDImageなど)。

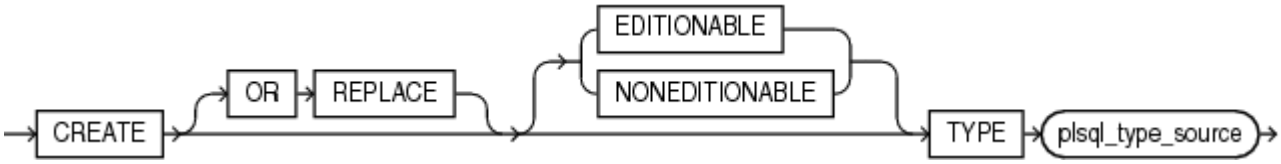
SQLユーザー定義データ型のルール

- 永続可能な型には、永続不可の型の属性または要素を指定できません。
- 永続不可の型には、永続可能な型と永続不可の型の両方の属性または要素を指定できます。
- サブタイプは、そのスーパータイプから永続プロパティを継承する必要があります。
- REF型は永続可能であり、永続可能な型のオブジェクトへの参照のみを保持できます。
- 永続不可の型のインスタンスをディスク上で保持することはできません。永続不可として宣言された型を使用して表を作成する場合、CREATE TABLE文は失敗します。同様に次の操作は失敗します。
    - 永続不可の型の列を含むリレーショナル表を作成または変更する。
    - 永続不可の型の列を含むオブジェクト表を作成する。
    - ディスク上で保持されているANYDATAインスタンスに、永続不可の型のインスタンスを格納する。

構文

型はPL/SQLを使用して定義されます。このため、このマニュアルの構文図ではSQLキーワードのみを示します。PL/SQLの構文、セマンティクスおよび例については、『Oracle Database PL/SQL言語リファレンス』を参照してください。

create_type::=



(plsql_type_sourceについては、『Oracle Database PL/SQL言語リファレンス』を参照してください。)

セマンティクス

OR REPLACE

OR REPLACEを指定すると、既存の型を再作成できます。この句を指定した場合、既存の型の定義をはじめに削除しなくても、その定義を変更できます。

再作成するオブジェクト型に対する権限があらかじめ付与されている場合は、再作成後にあらためて権限を付与されなくてもそのオブジェクト型を使用および参照できます。

ファンクション索引が型に依存している場合、索引にDISABLEDのマークが付きます。

[EDITIONABLE | NONEDITIONABLE]

この句を使用すると、schemaのスキーマ・オブジェクト・タイプTYPEのエディションニングが有効化されたときに、その型をエディショ

ン・オブジェクトにするか非エディション・オブジェクトにするかを指定できます。デフォルトはEDITIONABLEです。エディション・オブジェクトと非エディション・オブジェクトの詳細は、『Oracle Database開発ガイド』を参照してください。

plsql_type_source

plsql_type_sourceの構文およびセマンティクスについては、『Oracle Database PL/SQL言語リファレンス』を参照してください。

CREATE TYPE BODY

目的

型本体はPL/SQLを使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

CREATE TYPE BODYを使用すると、オブジェクト型仕様部で定義されたメンバー・メソッドを定義または実装することができます。CREATE TYPE文およびCREATE TYPE BODY文を使用してオブジェクト型を作成します。CREATE TYPE文では、オブジェクト型の名前、オブジェクトの属性、メソッドおよびその他のプロパティを指定します。CREATE TYPE BODY文には、その型を実装するメソッドに対するコードが含まれます。

call_specを定義していないオブジェクト型仕様部に指定された各メソッドには、オブジェクト型本体の対応するメソッド本体を指定する必要があります。

ノート:

SQLJ オブジェクト型を作成する場合は、Java クラスとして指定されます。

関連項目:

- 型仕様部の作成の詳細は、「CREATE TYPE」を参照してください。
- 型仕様部の変更の詳細は、「ALTER TYPE」を参照してください。

前提条件

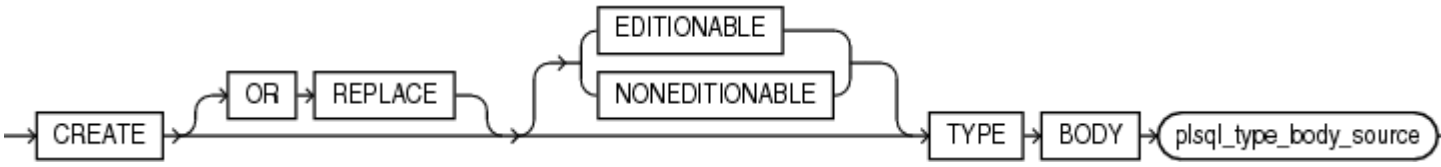
オブジェクト型用のCREATE TYPE仕様部で行われるすべてのメンバー宣言には、それに対応する構造がCREATE TYPE文またはCREATE TYPE BODY文内に存在する必要があります。

自分のスキーマ内で型本体を作成または再作成する場合は、CREATE TYPEシステム権限またはCREATE ANY TYPEシステム権限が必要です。他のユーザーのスキーマ内でオブジェクト型を作成する場合は、CREATE ANY TYPEシステム権限が必要です。他のユーザーのスキーマ内でオブジェクト型を置換する場合は、DROP ANY TYPEシステム権限が必要です。

構文

型本体はPL/SQLを使用して定義されます。このため、このマニュアルの構文図ではSQLキーワードのみを示します。PL/SQLの構文、セマンティクスおよび例については、『Oracle Database PL/SQL言語リファレンス』を参照してください。

create_type_body::=



(plsql_type_body_sourceについては、『Oracle Database PL/SQL言語リファレンス』を参照してください。)

セマンティクス

OR REPLACE

OR REPLACEを指定すると、既存の型本体を再作成できます。この句を指定した場合、既存の型本体の定義をはじめに削除しなくても、その定義を変更できます。

再作成されたオブジェクト型本体に対する権限を付与されているユーザーは、権限を再付与されなくても、そのオブジェクト型本体を使用および参照できます。

この句を使用した場合、ALTER TYPE ... REPLACE文によって追加された仕様部に、新規メンバー・サブプログラム定義を追加できます。

[EDITIONABLE | NONEDITIONABLE]

この句を省略すると、型本体は、EDITIONABLEまたはNONEDITIONABLEを型仕様部から継承します。この句を指定する場合は、型仕様部と一致させる必要があります。

plsql_type_body_source

plsql_type_body_sourceの構文およびセマンティクスについては、『Oracle Database PL/SQL言語リファレンス』を参照してください。

CREATE USER

目的

CREATE USER文を使用すると、データベース・ユーザー(データベースにログイン可能なアカウント)を作成および構成でき、Oracle Databaseがそのユーザーによるアクセスを許可する方法が確立されます。

この文をOracle Automatic Storage Management (Oracle ASM)クラスタで発行すると、現行のノードのOracle ASMインスタンスに対してローカルなパスワード・ファイルに、ユーザーおよびパスワードの組合せを追加できます。各ノードのOracle ASMインスタンスでは、この文を使用して個々のパスワード・ファイルを更新できます。このパスワード・ファイル自体は、ORAPWDユーティリティで作成されている必要があります。

プロキシ・アプリケーションまたはアプリケーション・サーバーによって、データベースとユーザーを接続できます。構文および説明は、「ALTER USER」を参照してください。

前提条件

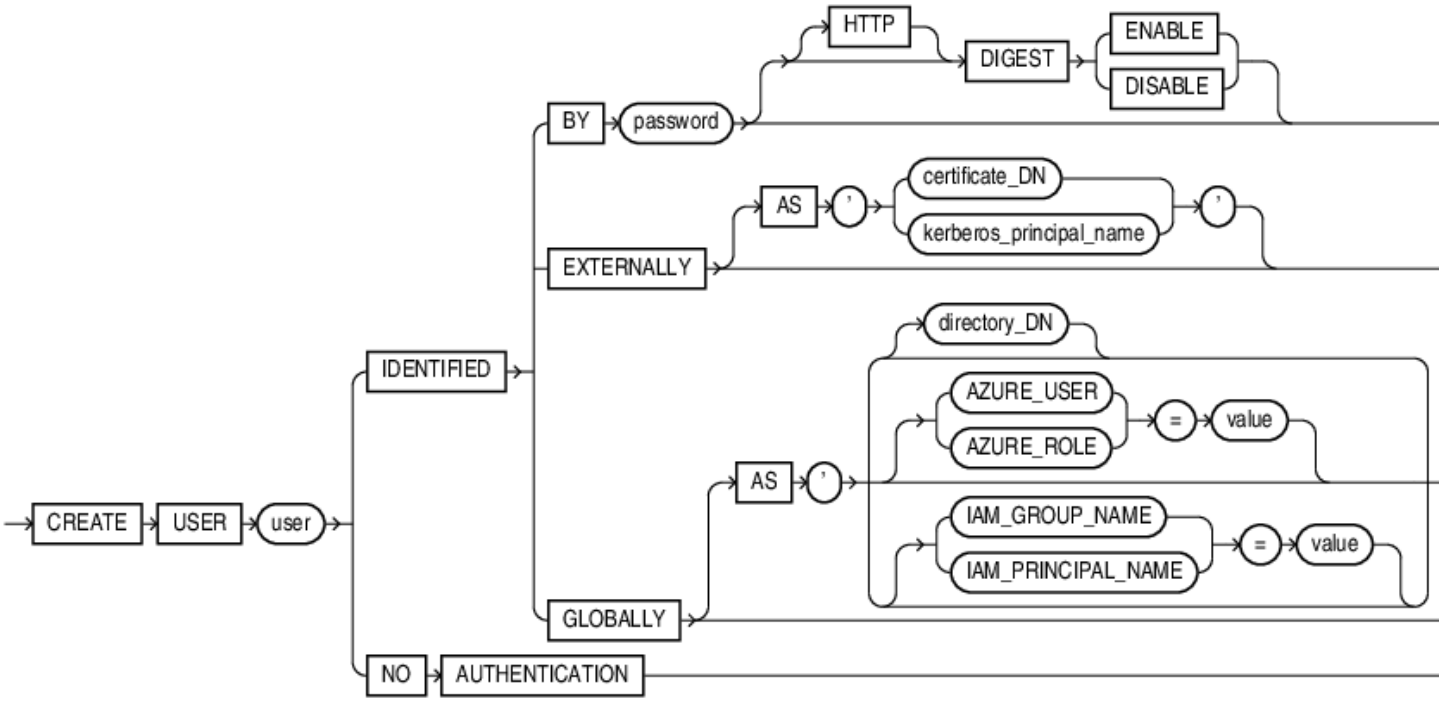
CREATE USERシステム権限が必要です。CREATE USER文を使用して作成したユーザーの権限ドメインは空(権限を付与されていない状態)になります。Oracle Databaseにログオンするユーザーには、CREATE SESSIONシステム権限が必要です。そのため、ユーザーを作成した場合、必ずCREATE SESSIONシステム権限をそのユーザーに付与してください。詳細は、「GRANT」を参照してください。

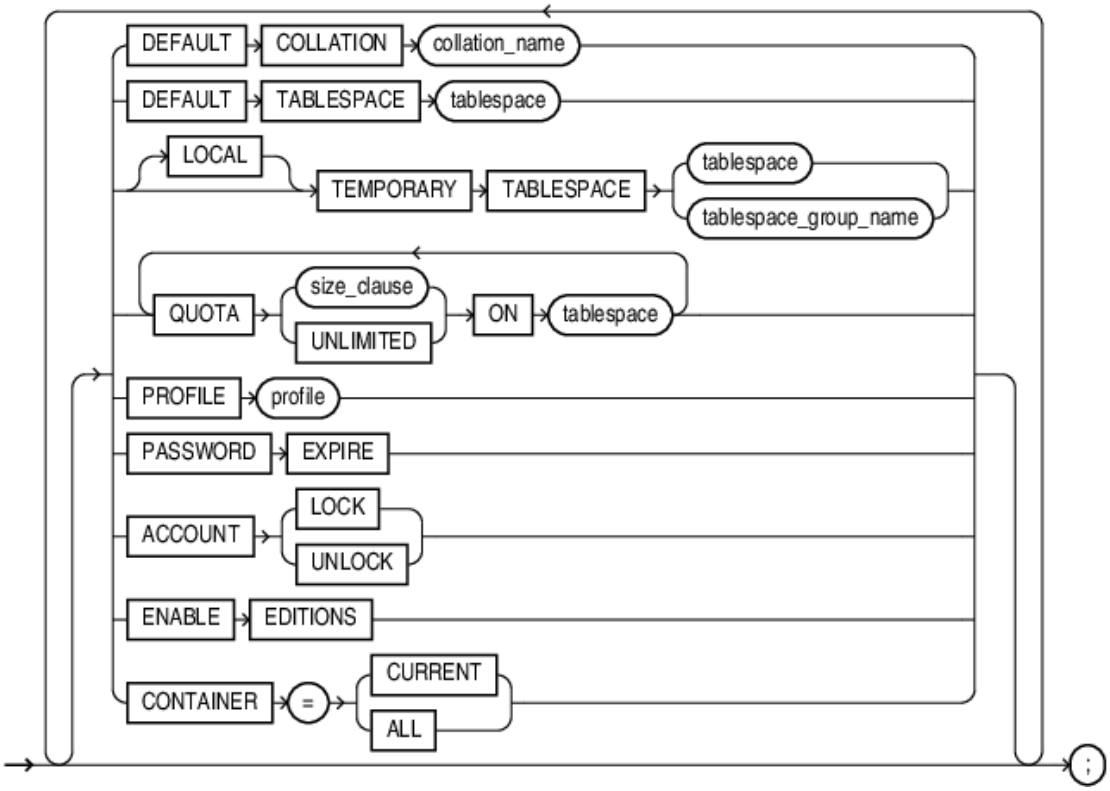
AS SYSASMとして認証されたユーザーのみが、このコマンドを発行して、Oracle ASMインスタンスのパスワード・ファイルを変更できます。

CONTAINER句を指定する場合は、マルチテナント・コンテナ・データベース(CDB)に接続している必要があります。CONTAINER = ALLを指定する場合は、現在のコンテナがルートである必要があります。CONTAINER = CURRENTを指定するには、現在のコンテナがプラガブル・データベース(PDB)である必要があります。

構文

create_user::=





(size_clause::=)

セマンティクス

user

作成するユーザーの名前を指定します。この名前には、使用しているデータベース文字セットの文字のみを指定できます。また、「データベース・オブジェクトのネーミング規則」の項で説明した規則に従う必要があります。データベース文字セットにマルチバイト文字が含まれている場合でも、ユーザーにはシングルバイト文字を1つ以上使用することをお薦めします。

CDB以外では、ユーザー名の先頭をC##またはc##にできません。

CDBでは、ユーザー名の要件は次のとおりです。

- 共通ユーザーの名前は、COMMON_USER_PREFIX初期化パラメータで指定された接頭辞と大/小文字を区別しないで一致する文字から始める必要があります。デフォルトでは、接頭辞はC##です。
- ローカル・ユーザーの名前は、COMMON_USER_PREFIX初期化パラメータで指定された接頭辞と大/小文字を区別しないで一致する文字から始めないでください。COMMON_USER_PREFIXの値に関係なく、ローカル・ユーザーの名前の先頭にC##またはc##を使用できません。

ノート:

COMMON_USER_PREFIX の値が空の文字列である場合、共通またはローカル・ユーザー名に要件はありませんが、次の 1 つの例外があります。ローカル・ユーザーの名前の先頭に C##または c##を使用できません。PDB を別の CDB に接続する場合または共通ユーザーの作成時に閉じられた PDB を開く場合、ローカルおよび共通ユーザーの名前が競合する可能性があるため、空の文字列を使用しないことをお薦めします。

ノート:

ユーザー名およびパスワードは、ご使用のプラットフォームに応じて、ASCII または EBCDIC 文字のみでエンコードすることをお薦めします。

関連項目:

データベース・ユーザーの作成: 例

IDENTIFIED句

IDENTIFIED句を使用すると、Oracle Databaseによるユーザーの認証方法を指定できます。

BY password

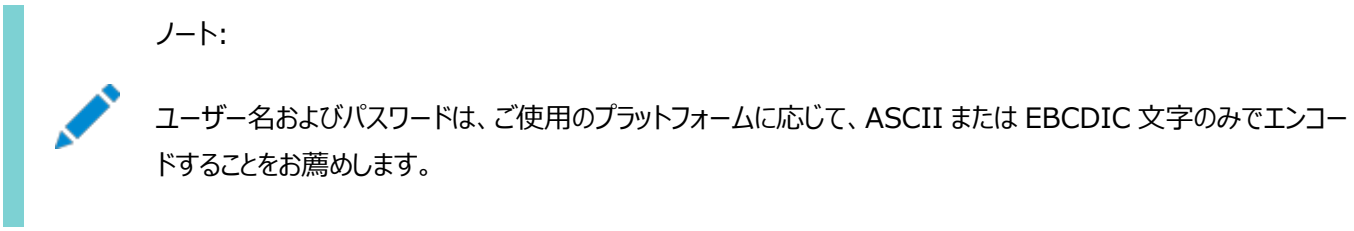
BY password句を使用すると、ローカル・ユーザーを作成し、データベースへのログイン時に、パスワードの指定が必要であることを指定できます。パスワードは大/小文字が区別されます。この後に、ユーザーをデータベースに接続するために使用されるCONNECT文字列では、このCREATE USER文または後述のALTER USER文で使用されているものと同じ文字(大文字、小文字または混在)を使用してパスワードを指定する必要があります。パスワードには、二重引用符(")およびリターン文字を除く、データベース文字セットのシングルバイト文字、マルチバイト文字、特殊文字またはこれらの組合せを含めることができます。パスワードがアルファベット以外の文字で始まるか、英数字、アンダースコア(_)、ドル記号($)およびポンド記号(#)以外の文字を含む場合は、その文字を二重引用符で囲む必要があります。そうでない場合は、任意でパスワードを二重引用符で囲むこともできます。

関連項目:

パスワードでの大/小文字の区別、パスワードの複雑さ、その他パスワードに関するガイドラインについては、『Oracle Databaseセキュリティ・ガイド』を参照してください。

Oracle Databaseの、3つの複雑なパスワード検証ルーチンの1つを使用していないかぎり、パスワードは、「データベース・オブ

ジェクトのネーミング規則」の項で説明した規則に従う必要があります。これらのルーチンでは、通常のネーミング規則より複雑な文字の組合せが要求されます。これらのルーチンは、UTLPWDMG.SQLスクリプトを使用して実装します。詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。



ノート:

ユーザー名およびパスワードは、ご使用のプラットフォームに応じて、ASCII または EBCDIC 文字のみでエンコードすることをお薦めします。

関連項目:

パスワード管理およびパスワード保護の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

[HTTP] DIGEST句

この句を使用すると、ユーザーのHTTP Digestアクセス認証をENABLEまたはDISABLEにできます。デフォルトはDISABLEです。

HTTPキーワードはオプションで、セマンティクスを明確にするために使用されます。

[HTTP] DIGEST句の制限事項

この句は、外部ユーザーまたはグローバル・ユーザーには指定できません。

EXTERNALLY句

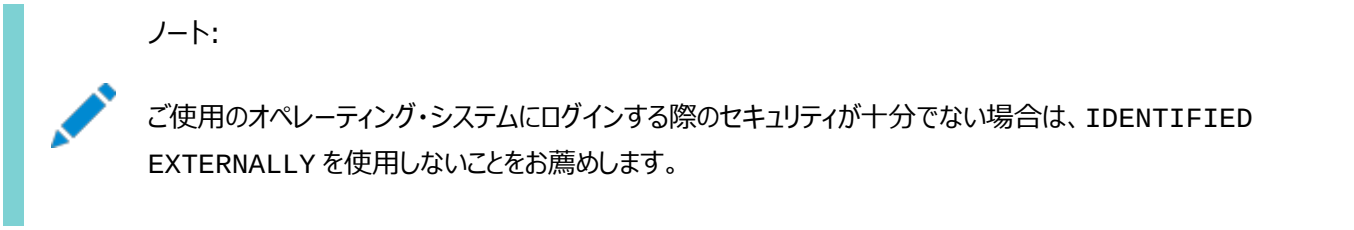
EXTERNALLYを指定すると、外部ユーザーを作成できます。このユーザーは、外部サービス(オペレーティング・システムやサード・パーティのサービスなど)で認証する必要があります。その場合、オペレーティング・システムまたはサード・パーティ・サービスによる認証を使用して、特定の外部ユーザーが特定のデータベース・ユーザーへのアクセス権を所有することが可能になります。

AS 'certificate_DN'

この句はSSL認証の外部ユーザーに必須で、このユーザーに対してのみ使用されます。certificate_DNは、ユーザーのウォレット内のユーザーのPKI証明書にある識別名です。certificate_DNの最大長は1024文字です。

AS 'kerberos_principal_name'

この句はKerberos認証の外部ユーザーに必須で、このユーザーに対してのみ使用されます。kerberos_principal_nameの最大長は1024文字です。



ノート:

ご使用のオペレーティング・システムにログインする際のセキュリティが十分でない場合は、IDENTIFIED EXTERNALLY を使用しないことをお薦めします。

外部ユーザーの作成の制限事項

Oracle ASMでは、外部ユーザーの作成はサポートされません。

関連項目:

- 外部で識別されるユーザーの詳細は、『Oracle Databaseエンタープライズ・ユーザー・セキュリティ管理者ガイド』を参照してください。
- 外部データベース・ユーザーの作成: 例

GLOBALLY句

GLOBALLY句を使用すると、グローバル・ユーザーを作成することができます。このユーザーは、エンタープライズ・ディレクトリ・サービス(Oracle Internet Directory)によって認可される必要があります。

directory_DN文字列は、次のいずれかの形式になります。

- このユーザーを識別するエンタープライズ・ディレクトリ・サービスのX.509の名前。文字列は、CN=username,other_attributesという形式である必要があります。other_attributesは、ディレクトリ内のユーザーの識別名(DN)以外の部分です。このフォームはLDAP Data Interchange Format(LDIF)を使用しており、プライベート・グローバル・スキーマを作成します。
- NULL文字列(' ')。エンタープライズ・ディレクトリ・サービスによって、認証されたグローバル・ユーザーがこのデータベース・スキーマにマップされ、適切なロールが割り当てられます。この形式を使用すると、GLOBALLYキーワードのみを指定する場合と同様に、共有グローバル・スキーマが作成されます。

directory_DNの最大長は1024文字です。

特定のユーザーとして接続し、ALTER USER文を使用してユーザーのロールをアクティブにするために、アプリケーション・サーバーの機能を制御できます。

GLOBALLY AS AZURE_USERを使用して、Oracle DatabaseスキーマをMicrosoft Azure ADユーザーに排他的にマップできます。CREATE USERまたはALTER USERシステム権限が付与されたユーザーとして、Oracle Autonomous Databaseインスタンスにログインする必要があります。

例: Microsoft Azure ADユーザーへのOracle Databaseスキーマのマッピング

この例では、peter_fitchという名前の新しいデータベース・スキーマ・ユーザーを作成し、このユーザーをpeter.fitch@example.comという名前の既存のAzure ADユーザーにマップします。

```
CREATE USER peter_fitch IDENTIFIED GLOBALLY AS 'AZURE_USER=peter.fitch@example.com';
```

例: アプリケーション・ロールへの共有Oracle Databaseスキーマのマッピング

この例では、dba_azureという名前の新しいデータベース・グローバル・ユーザー・アカウント(スキーマ)を作成し、AZURE_DBAという名前の既存のAzure ADアプリケーション・ロールにマップします。

```
CREATE USER dba_azure IDENTIFIED GLOBALLY AS 'AZURE_ROLE=AZURE_DBA';
```

グローバル・ユーザーの作成の制限事項

Oracle ASMでは、グローバル・ユーザーの作成はサポートされません。

関連項目:

- グローバル・ユーザーの詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。
- Oracle Autonomous DatabasesのMicrosoft Azure Active Directoryユーザーの認証および認可
- ALTER USER

- グローバル・データベース・ユーザーの作成: 例

NO AUTHENTICATION句

パスワードがないためログインできないスキーマを作成するには、NO AUTHENTICATION句を使用します。これはスキーマ専用アカウントを対象としており、デフォルト・パスワードを削除しパスワードのローテーションの必要性を解消することで、メンテナンス作業が軽減されます。

DEFAULT COLLATION句

この句を使用すると、ユーザーが所有するスキーマのデフォルトの照合を指定できます。デフォルトの照合は、スキーマに後で作成される表、ビューおよびマテリアライズド・ビューに割り当てられます。

collation_nameには、有効な名前付き照合または疑似照合を指定します。

この句を省略すると、ユーザーが所有するスキーマのデフォルトの照合がUSING_NLS_COMP疑似照合に設定されます。

この句を上書きして、特定の表、マテリアライズド・ビューまたはビューに別のデフォルトの照合を割り当てるには、表、マテリアライズド・ビューまたはビューにCREATE文またはALTER文のDEFAULT COLLATION句を指定します。また、データベース・セッション中にすべてのスキーマのデフォルトの照合を上書きするには、このセッションに対するデフォルトの照合を設定します。詳細は、「ALTER SESSION」の「DEFAULT_COLLATION」句を参照してください。

DEFAULT COLLATION句を指定できるのは、COMPATIBLE初期化パラメータが12.2以上に設定され、かつMAX_STRING_SIZE初期化パラメータがEXTENDEDに設定されている場合のみです。

DEFAULT TABLESPACE句

ユーザーのスキーマ内に作成されるオブジェクトを格納するデフォルトの表領域を指定します。この句を省略した場合、ユーザーのオブジェクトはデータベースのデフォルトの表領域に格納されます。データベースのデフォルトの表領域が指定されていない場合、ユーザーのオブジェクトはSYSTEM表領域に格納されます。

デフォルト表領域の制限事項

ローカル管理の一時表領域(UNDO表領域を含む)またはディクショナリ管理の一時表領域は、ユーザーのデフォルトの表領域として指定できません。

関連項目:

- 表領域の概要およびUNDO表領域の詳細は、「CREATE TABLESPACE」を参照してください。
- デフォルトの表領域をユーザーに割り当てる方法の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

[LOCAL] TEMPORARY TABLESPACE句

ユーザーの一時セグメントが確保される表領域または表領域グループを指定します。この句を省略した場合、ユーザーの一時セグメントはデータベースのデフォルトの一時表領域に格納されます。データベースのデフォルトの一時表領域が指定されていない場合は、SYSTEM表領域に格納されます。

- tablespaceに、ユーザーの一時セグメント表領域を指定します。共有一時表領域を指定するには、TEMPORARY TABLESPACEを指定します。ローカル一時表領域を指定するには、LOCAL TEMPORARY TABLESPACEを指定します。CDBに接続しているときに、CDB$DEFAULTを指定すると、CDB全体のデフォルト一時表領域を使用できます。

- `tablespace_group_name`を指定すると、ユーザーは、`tablespace_group_name`で指定された表領域グループ内の任意の表領域に一時セグメントを保存できるようになります。ローカル一時表領域を表領域グループに含めることはできません。

一時表領域の制限事項

この句には、次の制限事項があります。

- 表領域は一時表領域で、標準ブロック・サイズである必要があります。
- 表領域は、UNDO表領域または自動セグメント領域管理の表領域にできません。

関連項目:

- 表領域グループの詳細は、『Oracle Database管理者ガイド』を参照してください。一時表領域をユーザーに割り当てる方法の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。
- UNDO表領域およびセグメント管理の詳細は、「CREATE TABLESPACE」を参照してください。
- 表領域グループの割当て: 例

QUOTA句

`QUOTA`句を使用すると、ユーザーが表領域内に割り当てることができる最大サイズを指定できます。

`CREATE USER`文では、複数の表領域に対して複数の`QUOTA`句を指定できます。

`UNLIMITED`を使用すると、ユーザーは、表領域の領域を無制限に割り当てることができます。

指定できる領域の最大量は2テラバイト(TB)です。より多くの領域が必要な場合は、`UNLIMITED`を指定します。

QUOTA句の制限事項

この句は、一時表領域には指定できません。

関連項目:

この句の詳細は、「size_clause」を参照してください。表領域の割当て制限の指定については、『Oracle Databaseセキュリティ・ガイド』を参照してください。

PROFILE句

ユーザーに割り当てるプロファイルを指定します。このプロファイルによって、ユーザーが使用できるデータベース・リソース容量が制限されます。この句を省略した場合、`DEFAULT`プロファイルがユーザーに割り当てられます。

ノート:

データベース・リソース制限を設定する場合、この SQL プロファイルではなく、データベース・リソース・マネージャを使用することをお薦めします。データベース・リソース・マネージャを使用すると、リソース使用の管理および監査を柔軟に行うことができます。データベース・リソース・マネージャの詳細は、『Oracle Database 管理者ガイド』を参照してください。

関連項目:

「GRANT」および「CREATE PROFILE」を参照してください。

PASSWORD EXPIRE句

PASSWORD EXPIREを指定すると、ユーザーのパスワードを期限切れにできます。この設定によって、ユーザーがデータベースにログインする前に、ユーザーまたはDBAはパスワードの変更が必要となります。

ACCOUNT句

ACCOUNT LOCKを指定すると、ユーザー・アカウントをロックし、アクセスを禁止できます。ACCOUNT UNLOCKを指定すると、ユーザー・アカウントのロックを解除し、アカウントへのアクセスを許可できます。デフォルトはACCOUNT UNLOCKです。

ENABLE EDITIONS

この句は元に戻すことができません。ENABLE EDITIONSを指定すると、ユーザーは、エディションを使用しているスキーマ内で、エディション化可能なオブジェクトの複数のバージョンを作成できるようになります。エディションが有効でないスキーマ内のエディション化可能なオブジェクトは、エディション化できません。

ALTER USERでエディションを有効にする前に、次の点に注意してください。

- エディションの有効化は、ライブ操作ではありません。
- データベースがリリース11.2からリリース12.1にアップグレードされると、アップグレード前のデータベースでエディションに対して有効であったユーザーは、アップグレード後のデータベースでもエディションに対して有効になり、デフォルトのスキーマ・オブジェクト・タイプはこれらのスキーマでエディション対応になります。デフォルトのスキーマ・オブジェクト・タイプは、静的データ・ディクショナリ・ビューDBA_EDITIONED_TYPESによって表示されます。アップグレード前のデータベースでエディションに対して無効だったユーザーは、アップグレード後のデータベースでもエディションに対して無効であり、スキーマ・オブジェクト・タイプはこれらのスキーマではエディション対応になりません。
- エディションが有効になっているユーザー確認するには、静的データ・ディクショナリ・ビューDBA_USERSまたはUSER_USERSのEDITIONS_ENABLED列を参照してください。

エディションの有効化の制限事項

FOR句は、ENABLE EDITIONSとともに使用すると無視されます。これはALTER USER文ではなく、CREATE USER文にのみ適用されます。

シード・データベースのサンプル・スキーマを除いて、Oracle提供のすべてのスキーマではエディションを有効にできません。

関連項目:

- ユーザーに対するエディションの有効化
- V$EDITIONABLE_TYPES動的パフォーマンス・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

CONTAINER句

CONTAINER句は、CDBに接続しているときに適用されます。ただし、デフォルト値が使用できる唯一の値であるため、CONTAINER句を指定する必要はありません。

- 共通ユーザーを作成するには、ルートに接続する必要があります。ルートに接続する場合にデフォルトのCONTAINER = ALLをオプションで指定できます。

- ローカル・ユーザーを作成するには、PDBに接続する必要があります。PDBに接続する場合にデフォルトの CONTAINER = CURRENTをオプションで指定できます。

共通ユーザーの作成時には、次の句を使用して指定したデフォルトの表領域、一時表領域またはプロファイルがいずれも、CDBに属するすべてのコンテナ内に存在している必要があります。

- DEFAULT TABLESPACE
- TEMPORARY TABLESPACE
- QUOTA
- PROFILE

これらのオブジェクトが存在していないコンテナがあると、CREATE USER文は失敗します。

例

次のすべての例では、example表領域を使用します。この表領域はシード・データベースに存在し、サンプル・スキーマにアクセス可能です。

データベース・ユーザーの作成: 例

PASSWORD EXPIREを指定して新規ユーザーを作成する場合、データベースにログインする前に、そのユーザーのパスワードを変更する必要があります。次の文は、sidneyユーザーを作成します。

```
CREATE USER sidney
    IDENTIFIED BY out_standing1
    DEFAULT TABLESPACE example
    QUOTA 10M ON example
    TEMPORARY TABLESPACE temp
    QUOTA 5M ON system
    PROFILE app_user
    PASSWORD EXPIRE;
```

前述のユーザーsidneyには次の特性があります。

- パスワードout_standing1
- 10MBの割当て制限のあるデフォルト表領域example
- 一時表領域temp
- 5MBの割当て制限のある表領域SYSTEMへのアクセス
- プロファイルapp_user (「プロファイルの作成: 例」で作成)によって定義されているデータベース・リソースの制限
- sidneyがデータベースにログインする前に変更する必要がある期限切れのパスワード

外部データベース・ユーザーの作成: 例

次の例は、データベースにアクセスする前に外部ソースによって識別される必要のある、外部ユーザーを作成します。

```
CREATE USER app_user1
   IDENTIFIED EXTERNALLY
   DEFAULT TABLESPACE example
   QUOTA 5M ON example
   PROFILE app_user;
```

ユーザーapp_user1には次の追加の特性があります。

- デフォルトの表領域example
- デフォルトの一時表領域example

- 表領域exampleに5MBの領域、およびデータベースの一時表領域に無制限の割当て
- プロファイルapp_userによって定義されているデータベース・リソースの制限

オペレーティング・システム・アカウントによってのみアクセス可能な別のユーザーを作成する場合、ユーザー名に初期化パラメータOS_AUTHENT_PREFIX値の接頭辞を付けます。たとえば、この値が「ops$」の場合、次の文を使用して、外部で識別されるユーザーexternal_userを作成できます。

```
CREATE USER ops$external_user
   IDENTIFIED EXTERNALLY
   DEFAULT TABLESPACE example
   QUOTA 5M ON example
   PROFILE app_user;
```

グローバル・データベース・ユーザーの作成: 例

次の例は、グローバル・ユーザーを作成します。グローバル・ユーザーを作成する場合、エンタープライズ・ディレクトリ・サーバーでユーザーを識別するX.509の名前を指定することができます。

```
CREATE USER global_user
   IDENTIFIED GLOBALLY AS 'CN=analyst, OU=division1, O=oracle, C=US'
   DEFAULT TABLESPACE example
   QUOTA 5M ON example;
```

CDB内の共通ユーザーの作成

次の例は、c##comm_userという共通ユーザーをCDBに作成します。このCREATE USER文を実行する前に、CDBに属するすべてのコンテナに表領域exampleとtemp_tbsが存在していることを確認してください。

```
CREATE USER c##comm_user
   IDENTIFIED BY comm_pwd
   DEFAULT TABLESPACE example
   QUOTA 20M ON example
   TEMPORARY TABLESPACE temp_tbs;
```

ユーザーcomm_userには次の追加の特性があります。

- パスワードcomm_pwd
- 20MBの割当て制限のあるデフォルト表領域example
- 一時表領域temp_tbs

CREATE VIEW

目的

CREATE VIEW文を使用すると、ビューを定義できます。ビューとは、1つ以上の表またはビューをベースとした論理表です。ビュー自体にはデータは含まれていません。ビューのベースとなる表は、実表と呼ばれます。

LOB、オブジェクト型、REFデータ型、ネストした表またはVARRAY型をサポートするオブジェクト・ビューまたはリレーショナル・ビューを、既存のビュー・メカニズムで作成することもできます。オブジェクト・ビューとは、ユーザー定義型のビューのことで、ビューの各行に、それぞれが一意のオブジェクト識別子を持つオブジェクトが含まれます。

XMLTypeビューも作成できます。このビューは、オブジェクト・ビューと似ていますが、XMLTypeのXMLスキーマベースの表のデータを表示します。

関連項目:

- 様々な種類のビューおよびその使用方法については、『Oracle Database概要』、『Oracle Database開発ガイド』および『Oracle Database管理者ガイド』を参照してください。
- XMLTypeビューの詳細は、『Oracle XML DB開発者ガイド』を参照してください。
- ビューの変更およびデータベースからのビューの削除については、「ALTER VIEW」および「DROP VIEW」を参照してください。

前提条件

自分のスキーマ内にビューを作成する場合は、CREATE VIEWシステム権限が必要です。他のユーザーのスキーマ内にビューを作成する場合は、CREATE ANY VIEWシステム権限が必要です。

サブビューを作成する場合は、UNDER ANY VIEWシステム権限またはスーパービューに対するUNDERオブジェクト権限が必要です。

ビューが含まれているスキーマの所有者は、そのビューの基礎となっているすべての表またはビューに対する行の選択(READまたはSELECT権限)、挿入、更新または削除の権限が必要です。また、所有者には、これらの権限がロールを介してではなく、直接付与されている必要があります。

オブジェクト・ビューの作成時にオブジェクト型の基本コンストラクタ・メソッドを使用する場合、次のいずれかの条件を満たしている必要があります。

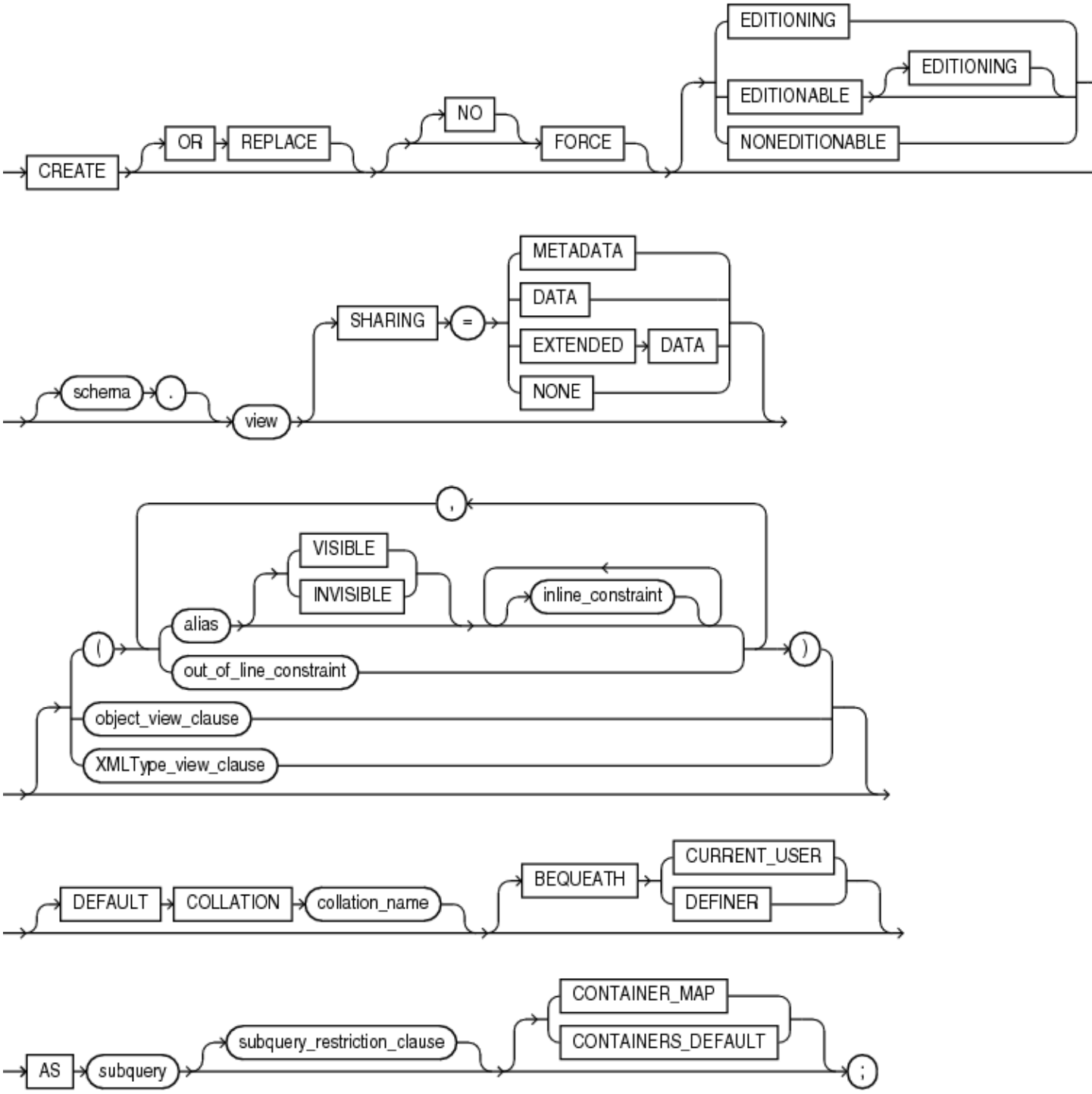
- オブジェクト型が作成対象のビューと同じスキーマに属している。
- EXECUTE ANY TYPEシステム権限を持っている。
- そのオブジェクト型に対するEXECUTEオブジェクト権限を持っている。

関連項目:

作成するビューの基礎となる表またはビューに対して、ビューの所有者に必要な権限の詳細は、「SELECT」、「INSERT」、「UPDATE」および「DELETE」を参照してください。

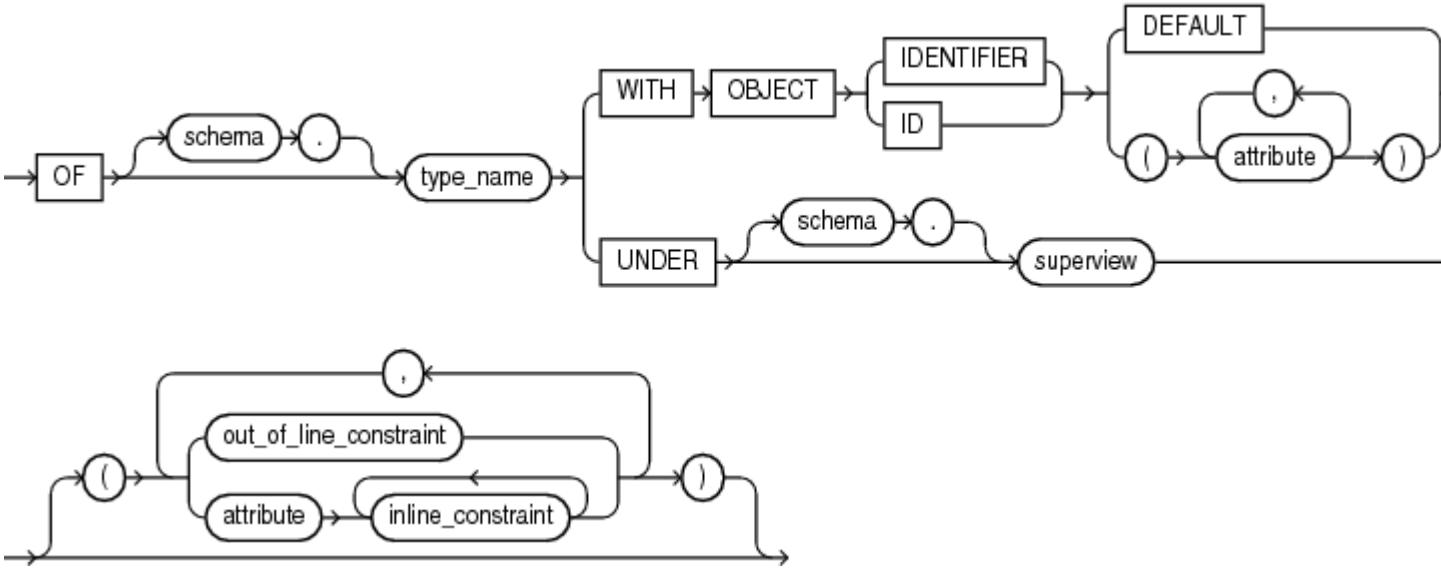
構文

create_view::=



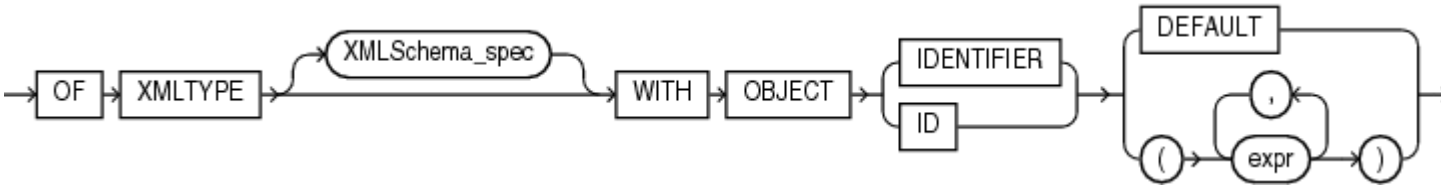
(inline_constraint::=およびout_of_line_constraint::=、object_view_clause::=、XMLType_view_clause::=、subquery::= (SELECTの一部)、subquery_restriction_clause::=)

object_view_clause::=

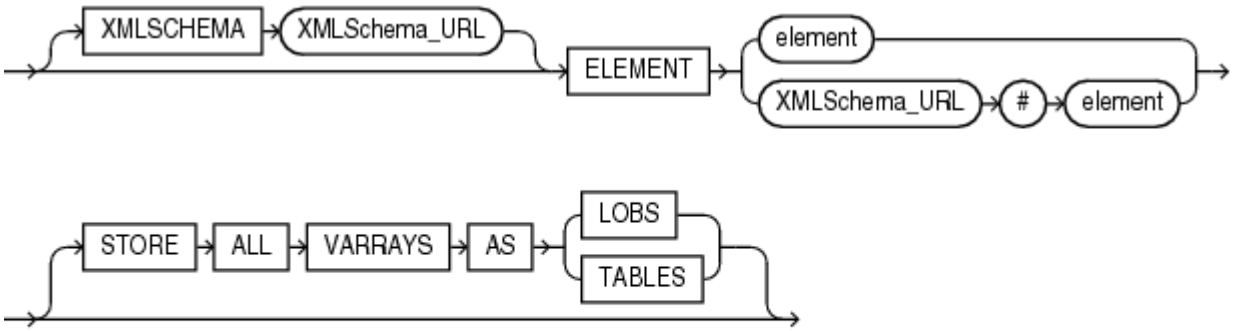


(inline_constraint::=および out_of_line_constraint::=)

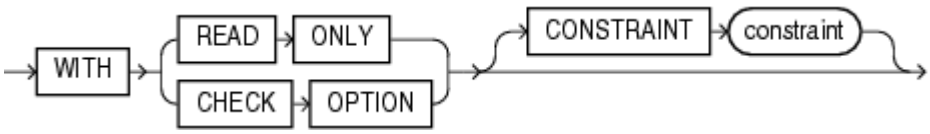
XMLType_view_clause::=



XMLSchema_spec::=



subquery_restriction_clause::=



セマンティクス

OR REPLACE

OR REPLACEを指定すると、既存のビューを再作成できます。この句を使用した場合、以前に付与されたオブジェクト権限を削除、再作成、再付与しなくても、既存のビュー定義を変更できます。

従来型のビューを再作成した場合、そのビューで定義されたINSTEAD OFトリガーは削除されます。エディショニング・ビューを再作成した場合、エディショニング・ビューで定義されたDMLトリガーは保持されます。ただし、エディショニング・ビューの変更によってコンパイルできない場合(たとえば、トリガー定義で参照されるエディショニング・ビュー列を削除した場合など)は、これらのトリガーを永続的に無効にできます。

すべてのマテリアライズド・ビューがviewに依存している場合、そのマテリアライズド・ビューにはUNUSABLEというマークが付けられ、それらが再利用可能になるようにリストアする場合は、完全なリフレッシュが必要になります。無効なマテリアライズド・ビューはクエリー・リライトで使用されることはなく、また、再コンパイルされるまでリフレッシュされません。

従来型のビューからエディショニング・ビューに、またはエディショニング・ビューから従来型のビューに置き換えることはできません。エディショニング・ビューの詳細は、『Oracle Database開発ガイド』を参照してください。

関連項目:

- 無効なマテリアライズド・ビューのリフレッシュについては、「ALTER MATERIALIZED VIEW」を参照してください。
- マテリアライズド・ビューの概要は、『Oracle Database概要』を参照してください。
- INSTEAD OF句の詳細は、「CREATE TRIGGER」を参照してください。

FORCE

FORCEを指定すると、ビューの実表または参照するオブジェクト型が存在しているか、またはそのビューを含むスキーマの所有者が、それらの表やオブジェクト型に対する権限を持っているかにかかわらず、ビューを作成できます。SELECT、INSERT、UPDATEまたはDELETE文をビューに対して発行する場合、これらの条件を満たしている必要があります。

ビュー定義が制約を含む場合、実表または参照するオブジェクト型が存在しないと、CREATE VIEW ... FORCEは正常に実行されません。また、ビュー定義で存在しない制約が指定される場合も、CREATE VIEW ... FORCEは正常に実行されません。

NO FORCE

NOFORCEを指定すると、実表が存在し、ビューを含むスキーマの所有者がその実表に対する権限を持っている場合のみ、ビューを作成できます。これはデフォルトです。

EDITIONING

この句を使用すると、エディショニング・ビューを作成できます。エディショニング・ビューは単一表ビューであり、実表からすべての行を選択し、実表の列のサブセットを表示します。エディショニング・ビューを使用すると、アップグレードなどの管理操作時に、アプリケーションに影響を与えることなく実表に対するDDL変更を実行できます。実表に対する既存のエディショニング・ビューの関係について、情報を取得する場合は、USER_、ALL_およびDBA_EDITIONING_VIEWデータ・ディクショナリ・ビューを問い合せます。

エディショニング・ビューの所有者は、エディション化が可能である必要があります。詳細は、「ENABLE EDITIONS」を参照してください。

エディショニング・ビューのノート

エディショニング・ビューと従来型のビューには、次のような重要な違いがあります。

- エディショニング・ビューは、表の列のサブセットの別名を選択して表示するためにのみ使用されます。そのため、エディショ

ニング・ビューを作成する構文には、従来型のビューを作成する構文よりも多くの制限事項があります。制限事項(内容については後述)に違反すると、FORCEを指定しても、ビューを作成できません。

- エディションニング・ビューには、DMLトリガーを作成できます。この場合、データベースは、エディションニング・ビューをこのトリガーに対する基本オブジェクトとみなします。このようなトリガーは、DML操作がエディションニング・ビューそのものを対象とした場合に起動します。DML操作が実表を対象とした場合には起動しません。
- エディションニング・ビューには、INSTEAD OFトリガーを作成できません。

エディションニング・ビューの制限事項

エディションニング・ビューには、次の制限事項があります。

- どのエディション内でも、単一表に対して作成できるエディションニング・ビューは1つのみです。
- object_view_clause句、XMLType_view_clause句、またはBEQUEATH句は指定できません。
- エディションニング・ビューには、WITH CHECK OPTION制約を定義できません。
- 定義する副問合せのSELECT構文のリストには、実表の列に対する単純な参照のみを指定できます。また、実表の各列を1回にかぎりSELECT構文のリストに指定できます。ワイルド・カード記号であるアスタリスク*およびt_alias.*は、実表のすべての列を指定する場合に使用できます。
- ビューを定義する副問合せのFROM句で参照できるのは、既存の1つのデータベース表のみです。結合は使用できません。実表は作成対象のビューと同じスキーマに属している必要があります。表の識別にシノニムは使用できません。ただし、表の別名を指定することはできます。
- 副問合せを定義する次の句は、エディションニング・ビューには無効です。subquery_factoring_clause、DISTINCTまたはUNIQUE、where_clause、hierarchical_query_clause、group_by_clause、HAVING条件、model_clause、または集合演算子(UNION、INTERSECTまたはMINUS)。

関連項目:

- エディションニング・ビューの詳細は、『Oracle Database開発ガイド』を参照してください。
- エディションの詳細およびエディションニング・ビューの例は、「CREATE EDITION」を参照してください。

EDITIONABLE | NONEDITIONABLE

この句を使用すると、schemaのスキーマ・オブジェクト・タイプVIEWのエディションニングが有効化されたときに、そのビューをエディション・オブジェクトにするか非エディション・オブジェクトにするかを指定できます。デフォルトは、EDITIONABLEです。エディション・オブジェクトと非エディション・オブジェクトの詳細は、『Oracle Database開発ガイド』を参照してください。

schema

ビューを含めるスキーマを指定します。schemaを指定しない場合、自分のスキーマにそのビューが作成されます。

view

ビューまたはオブジェクト・ビューの名前を指定します。名前は、「データベース・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

ビューの制限事項

あるビューにINSTEAD OFトリガーが含まれている場合、そのビューに対して作成されるビューが更新可能であっても、そのビューにはINSTEAD OFトリガーが必要です。

関連項目:

ビューの作成: 例

SHARING

この句は、アプリケーション・ルートにビューを作成する場合にのみ適用されます。このタイプのビューはアプリケーション共通オブジェクトと呼ばれ、そのデータは、アプリケーション・ルートに属するアプリケーションPDBと共有できます。ビュー・データの共有方法を決定するには、次の共有属性のいずれかを指定します。

- METADATA - メタデータ・リンクはビューのメタデータを共有しますが、そのデータは各コンテナに固有です。このタイプのビューは、メタデータリンク・アプリケーション共通オブジェクトと呼ばれます。
- DATA - データ・リンクはビューを共有し、そのデータはアプリケーション・コンテナ内のすべてのコンテナに対して同じです。そのデータはアプリケーション・ルートにのみ格納されます。このタイプのビューは、データリンク・アプリケーション共通オブジェクトと呼ばれます。
- EXTENDED DATA - 拡張データ・リンクはビューを共有し、アプリケーション・ルートのそのデータはアプリケーション・コンテナ内のすべてのコンテナに対して同じです。ただし、アプリケーション・コンテナの各アプリケーションPDBには、アプリケーションPDBに一意のデータを格納できます。このタイプのビューの場合、データはアプリケーション・ルートに格納され、オプションで各アプリケーションPDBに格納されます。このタイプのビューは、拡張データリンク・アプリケーション共通オブジェクトと呼ばれます。
- NONE - ビューは共有されません。

この句を省略すると、データベースはDEFAULT_SHARING初期化パラメータの値を使用して、ビューの共有属性を決定します。DEFAULT_SHARING初期化パラメータに値が含まれていない場合、デフォルトはMETADATAです。

従来型のビューを作成する場合は、METADATA、DATA、EXTENDED DATAまたはNONEを指定できます。

オブジェクト・ビューまたはXMLTYPEビューを作成する場合は、METADATAまたはNONEのみを指定できます。

ビューの共有属性は、作成後は変更できません。

関連項目:

- DEFAULT_SHARING初期化パラメータの詳細は、Oracle Databaseリファレンスを参照してください
- アプリケーション共通オブジェクトの作成の詳細は、『Oracle Database管理者ガイド』を参照してください。

alias

ビューを定義する問合せによって選択された式の名前を指定します。別名の数は、ビューによって選択された式の数と一致している必要があります。別名は、Oracle Databaseのスキーマ・オブジェクトのネーミング規則に従う必要があります。別名は、ビュー内で一意である必要があります。

別名を省略した場合、別名は問合せの列または列の別名から導出されます。このため、問合せに列の名前のみでなく式が含まれている場合は、別名を使用する必要があります。また、ビュー定義に制約が含まれている場合も、別名を指定する必要があります。

ビューの別名の制限事項

オブジェクト・ビューを作成するとき、別名を指定することはできません。

関連項目:

スキーマ・オブジェクトの構文およびSQL文の構成要素

VISIBLE | INVISIBLE

この句を使用すると、ビュー列をVISIBLEにするか、INVISIBLEにするかを指定できます。デフォルトでは、実表の可視性にかかわらず、ビュー列はVISIBLEになります。ただし、INVISIBLEを指定した場合を除きます。これは、従来型のビューとエディションニング・ビューに当てはまります。これらの句の詳細は、CREATE TABLEのドキュメントの「VISIBLE | INVISIBLE」を参照してください。

inline_constraintおよびout_of_line_constraint

ビューおよびオブジェクト・ビューには制約を指定できます。out_of_line_constraint句を使用すると、ビュー・レベルで制約を定義できます。適切な別名の後でinline_constraint句を使用すると、列定義または属性定義の一部として制約を定義できます。

Oracle Databaseでは、ビュー制約を適用していません。制限事項を含むビュー制約の詳細は、「ビュー制約」を参照してください。

関連項目:

制約付きのビューの作成: 例

object_view_clause

object_view_clauseを使用すると、オブジェクト型にビューを定義できます。

関連項目:

オブジェクト・ビューの作成: 例

OF type_name句

この句を使用すると、type_name型のオブジェクト・ビューを明示的に作成できます。オブジェクト・ビューの列は、type_name型の最上位属性に対応しています。各行にはオブジェクト・インスタンスが含まれ、また、各インスタンスはWITH OBJECT IDENTIFIER句で指定したオブジェクト識別子に関連付けられます。schemaを指定しない場合、自分のスキーマ内にオブジェクト・ビューが作成されます。

オブジェクト表、XMLType表、オブジェクト・ビューおよびXMLTypeビューには、列名は付けられません。そのため、システム生成疑似列OBJECT_IDが定義されます。問合せでこの列名を使用し、WITH OBJECT IDENTIFIER句を指定して、オブジェクト・ビューを作成できます。

WITH OBJECT IDENTIFIER句

WITH OBJECT IDENTIFIER句を使用すると、最上位(ルート)のオブジェクト・ビューを指定できます。オブジェクト・ビュー内の各行を識別するためのキーとして使用されるオブジェクト型の属性を指定します。ほとんどの場合、各属性は実表の主キー列と対応します。属性リストが一意で、ビューの1行ずつを識別することを確認する必要があります。WITH OBJECT IDENTIFIERおよびWITH OBJECT ID句は、区別なしに使用できますが、意味を明確にするために用意されています。

オブジェクト・ビューの制限事項

オブジェクト・ビューには、次の制限事項があります。

- オブジェクト・ビュー内の複数のインスタンスに変換される主キーREFを参照解除または確保しようとした場合、データベースはエラーを戻します。
- サブビューはスーパービューのオブジェクト識別子を継承するため、サブビューを作成する場合は、この句を指定できません。

オブジェクト・ビューがオブジェクト表またはオブジェクト・ビュー上で定義されている場合は、この句を省略するか、または`DEFAULT`を指定します。

DEFAULT

`DEFAULT`を指定すると、基礎となるオブジェクト表またはオブジェクト・ビュー固有のオブジェクト識別子を使用して、各行を一意に識別できます。

attribute

`attribute`では、オブジェクト・ビューに対して作成されるオブジェクト識別子の基になるオブジェクト型の属性を指定します。

UNDER句

`UNDER`句を使用すると、オブジェクト・スーパービューに基づくサブビューを指定できます。

サブビューの制限事項

サブビューには、次の制限事項があります。

- サブビューは、スーパービューと同じスキーマに作成する必要があります。
- オブジェクト型`type_name`は、`superview`直属のサブタイプである必要があります。
- 同一のスーパービューには、特定の型のサブビューを1つのみ作成できます。

関連項目:

- オブジェクトの作成の詳細は、「CREATE TYPE」を参照してください。
- データ・ディクショナリ・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

XMLType_view_clause

この句を使用すると、`XMLType`ビューを作成できます。このビューには、`XMLType`のXMLスキーマベースの表のデータが表示されます。`XMLSchema_spec`は、XMLデータを対応するオブジェクト・リレーショナル・データにマップするために使用するXMLスキーマを示します。XMLスキーマは、`XMLType`ビューを作成する前に作成しておく必要があります。

`WITH OBJECT IDENTIFIER`および`WITH OBJECT ID`句は、区別なしに使用できますが、意味を明確にするために用意されています。

オブジェクト表、`XMLType`表、オブジェクト・ビューおよび`XMLType`ビューには、列名は付けられません。そのため、システム生成疑似列`OBJECT_ID`が定義されます。問合せでこの列名を使用し、`WITH OBJECT IDENTIFIER`句を指定して、オブジェクト・ビューを作成できます。

関連項目:

- XMLTypeビューおよびXMLSchemasの詳細は、『Oracle XML DB開発者ガイド』を参照してください。

- 「XMLTypeビューの作成：例」および「XMLType表のビューの作成：例」を参照してください。

DEFAULT COLLATION

この句を使用して、ビューのデフォルトの照合を指定します。デフォルトの照合は、ビューを定義する問合せに含まれるすべての文字リテラルについて、導出された照合として使用されます。デフォルトの照合は、ビューの列では使用されません。ビューの列の照合は、ビューの定義副問合せから導出されます。CREATE VIEW文は、その文字列のいずれかが、導出された照合がない定義副問合せの式に基づいている場合は、エラーが発生して失敗します。

collation_nameには、有効な名前付き照合または疑似照合を指定します。

この句を省略した場合、ビューのデフォルトの照合は、ビューを含むスキーマの有効スキーマのデフォルトの照合に設定されます。有効なスキーマのデフォルトの照合の詳細は、「ALTER SESSION」のDEFAULT_COLLATION句を参照してください。

DEFAULT COLLATION句を指定できるのは、COMPATIBLE初期化パラメータが12.2以上に設定され、かつMAX_STRING_SIZE初期化パラメータがEXTENDEDに設定されている場合のみです。

ビューのデフォルトの照合を変更するには、ビューを再作成する必要があります。

ビューのデフォルトの照合の制限事項

ビューを定義する問合せにWITH plsql_declarations句が含まれている場合、ビューのデフォルトの照合はUSING_NLS_COMPである必要があります。

BEQUEATH

BEQUEATH句を使用すると、ビューで参照されるファンクションをビューの実行者権限で実行するか、ビューの定義者権限で実行するかを指定できます。

CURRENT_USER

BEQUEATH CURRENT_USERを指定すると、ビューで参照されるファンクションは、次の条件のいずれかが満たされているときに、ビューの実行者権限で実行されます。

- ビューの所有者が、実行ユーザーに対するINHERIT PRIVILEGESオブジェクト権限を所持している。
- ビューの所有者が、INHERIT ANY PRIVILEGESシステム権限を所持している。

ビューの問合せがIDまたは権限に影響を受けるSQLファンクションを実行する場合や、実行者権限のPL/SQLまたはJavaファンクションを実行する場合、その操作の実行範囲内での現在のスキーマ、現在のユーザー、およびその時点で有効にされるロールは、ビューの所有者から継承されるのではなく、問合せを実行したユーザーの環境から継承されます。

この句が、ビュー自体を実行者権限オブジェクトに変換することはありません。ビュー内での名前解決は、引き続きビューの所有者のスキーマを使用して処理され、ビューの権限チェックは、ビューの所有者の権限で実行されます。

DEFINER

BEQUEATH DEFINERを指定すると、ビューで参照されるファンクションは、ビューの定義者権限で実行されます。ビューへの問合せがIDまたは権限に影響を受けるSQLファンクションを実行する場合や、実行者権限のPL/SQLまたはJavaファンクションを実行する場合、その操作の実行範囲内での現在のスキーマ、現在のユーザー、およびその時点で有効にされるロールは、ビューの所有者から継承されます。

ビュー内での名前解決はビューの所有者のスキーマを使用して処理され、ビューの権限チェックはビューの所有者の権限で実行されます。

これはデフォルトです。

BEQUEATH句の制限事項

この句はEDITIONING句とともに指定できません。

関連項目:

ビューでの実行者権限および定義者権限を制御する方法の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

AS subquery

ビューの基になっている表(実表)の列と行を識別する副問合せを指定します。副問合せのSELECT構文のリストには1000以下の式を指定できます。

リモート表およびリモート・ビューを参照するビューを作成する場合、CREATE DATABASE LINK文のCONNECT TO句を使用して、指定するデータベース・リンクを作成しておく必要があります。また、ビュー副問合せのスキーマ名で修飾する必要があります。

ビューを定義する問合せでflashback_query_clauseを使用してビューを作成した場合、AS OF式は作成時には解析されず、ユーザーがビューを問い合せるたびに解析されます。

関連項目:

Oracleフラッシュ・バック問合せの詳細は、「結合ビューの作成: 例」および『Oracle Database開発ガイド』を参照してください。

ビューを定義する問合せの制限事項

ビュー問合せには、次の制限事項があります。

- 副問合せでは、CURRVALおよびNEXTVAL疑似列を選択できません。
- 副問合せでROWID、ROWNUMまたはLEVELの各疑似列を選択する場合、そのビュー副問合せには列の別名が必要です。
- 副問合せでアスタリスク(*)を使用して表のすべての列を選択し、後でその表に新しい列を追加する場合、CREATE OR REPLACE VIEW文を発行してビューを再作成するまで、追加した列はそのビューに含まれません。
- オブジェクト・ビューの場合、副問合せのSELECT構文のリスト内の要素数は、そのオブジェクト型の最上位属性数と同じである必要があります。それぞれの選択要素のデータ型は、対応する最上位属性と同じである必要があります。
- SAMPLE句は指定できません。

これらの制限は、マテリアライズド・ビューにも適用されます。

更新可能なビューのノート

次のノートは、更新可能なビューに適用されます。

更新可能なビューとは、実表の行を挿入、更新または削除できるビューです。更新可能なビューを作成するか、またはビューにINSTEAD OFトリガーを作成して更新可能にすることができます。

更新可能なビューの列を更新する方法を学習するには、USER_UPDATABLE_COLUMNSデータ・ディクショナリ・ビューを問い合せます。このビューで表示される情報は、更新可能なビューのみに意味があります。ビューを更新可能にするには、次の条件

が満たされている必要があります。

- ビューの各列は、単一表の列にマップする必要があります。たとえば、ビューの列がTABLE句の出力にマップする場合(コレクション・ネスト解除)、ビューは更新可能ではありません。
- ビューには、次の構造体が含まれていない必要があります。
  - 集合演算子
  - DISTINCT演算子
  - 集計ファンクションまたは分析ファンクション
  - GROUP BY、ORDER BY、MODEL、CONNECT BYまたはSTART WITH句
  - SELECT構文のリストにあるコレクション式
  - SELECT構文のリストにある副問合せ
  - WITH READ ONLYが指定された副問合せ
  - 結合(一部の例外を除く。詳細は、『Oracle Database管理者ガイド』を参照してください。)
- 更新可能なビューが疑似列または式を含む場合は、これらの疑似列または式を参照するUPDATE文を使用して、実表の行を更新できません。
- 結合ビューを更新可能にする場合、次のすべての条件が満たされている必要があります。
  - DML文は、結合の基になる1つの表のみに影響する。
  - INSERT文で、WITH CHECK OPTIONを指定してビューを作成しておらず、値が挿入されたすべての列がキー保存表の列である。実表の各主キーまたは一意キーの値に対するキー保存表は、結合ビューで一意である。
  - UPDATE文で、WITH CHECK OPTIONを指定してビューを作成しておらず、更新されるすべての列はキー保存表から抽出されている。
  - DELETE文の場合、結合によって複数のキー保存表が作成されると、ビューがWITH CHECK OPTIONを指定して作成されたかどうかにかかわらず、FROM句に指定された最初の表から削除されます。

関連項目:

- 更新可能なビューの詳細は、『Oracle Database管理者ガイド』を参照してください。
- 更新可能な結合ビューおよびキー保存表の例は、「更新可能なビューの作成: 例」および「結合ビューの作成: 例」を参照し、更新できないビューのINSTEAD OFトリガーの例は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

subquery_restriction_clause

subquery_restriction_clauseを使用すると、次のいずれかの方法で、ビューを定義する問合せを制限できます。

WITH READ ONLY

WITH READ ONLYを指定すると、表またはビューを更新禁止にできます。

WITH CHECK OPTION

WITH CHECK OPTIONを指定すると、副問合せに含まれない行を生成する表またはビューの変更を禁止できます。この句をDML文の副問合せ内で使用する場合、FROM句内の副問合せには指定できますが、WHERE句内の副問合せには指定できません。

CONSTRAINT constraint

READ ONLYまたはCHECK OPTION制約の名前を指定します。この識別子を省略した場合、その制約にSYS_Cnという形式の名前が自動的に割り当てられます。この場合のnは、その制約名をデータベース内で一意の名前にする整数です。

ノート:

表に WITH CHECK OPTION を指定すると、挿入および更新処理の結果、表を定義する副問合せによって選択可能な表が戻されることが保証されます。ビューでは、次の場合、WITH CHECK OPTION を指定してもこれらの保証はされません。

- 対象のビュー、または対象のビューの基礎となるビューを定義する問合せ内に、副問合せが含まれている場合。
- INSERT、UPDATE または DELETE 操作が INSTEAD OF トリガーを使用して実行された場合。

subquery_restriction_clauseの制限事項

ORDER BY句を指定する場合、この句は指定できません。

関連項目:

読取り専用ビューの作成: 例

CONTAINER_MAP

CONTAINER_MAP句を指定して、コンテナ・マップを使用したビューの問合せを有効化します。

CONTAINERS_DEFAULT

CONTAINERS_DEFAULT句を指定して、CONTAINERS句に対してビューを有効化します。

例

ビューの作成: 例

次の文は、emp_viewというサンプル表employeesのビューを作成します。このビューには、部門20の従業員とその年間給与が表示されます。

```
CREATE VIEW emp_view AS
   SELECT last_name, salary*12 annual_salary
   FROM employees
   WHERE department_id = 20;
```

この場合、副問合せで式salary*12に列の別名(annual_salary)を使用しているため、この式に基づく列の名前をビュー宣言で定義する必要はありません。

エディションニング・ビューの作成: 例

次の文は、orders表のエディションニング・ビューを作成します。

```
CREATE EDITIONING VIEW ed_orders_view (o_id, o_date, o_status)
  AS SELECT order_id, order_date, order_status FROM orders
  WITH READ ONLY;
```

このビューを使用すると、アップグレードなどの管理操作時に、アプリケーションに影響を与えることなくorders表に対するDDL変更を実行できます。DML操作がこのビューそのものを対象とした場合にDMLトリガーを起動するよう、このビューにDMLトリガーを作成できます。ただし、DML操作がorders表を対象とした場合には起動しません。

制約付きのビューの作成: 例

次の文は、サンプル表hr.employeesの制限付きのビューを作成し、ビューの列emailに一意制約を定義し、ビューの列emp_idにビューに対する主キー制約を定義します。

```
CREATE VIEW emp_sal (emp_id, last_name,
      email UNIQUE RELY DISABLE NOVALIDATE,
   CONSTRAINT id_pk PRIMARY KEY (emp_id) RELY DISABLE NOVALIDATE)
   AS SELECT employee_id, last_name, email FROM employees;
```

更新可能なビューの作成: 例

次の文は、employees表内の事務員全員の更新可能なビューclerkを作成します。従業員のID、姓、部門番号および職種はビューで参照でき、事務員の行のこれらの列のみを更新できます。

```
CREATE VIEW clerk AS
   SELECT employee_id, last_name, department_id, job_id
   FROM employees
   WHERE job_id = 'PU_CLERK'
      or job_id = 'SH_CLERK'
      or job_id = 'ST_CLERK';
```

このビューでは、購買係のjob_idを購買マネージャ(PU_MAN)に変更できます。

```
UPDATE clerk SET job_id = 'PU_MAN' WHERE employee_id = 118;
```

次の例は、同じビューをWITH CHECK OPTION付きで作成します。新しい従業員が事務員ではない場合は、clerkに新しい行を挿入できません。従業員のjob_idを他の事務タイプに更新できますが、事務員以外のjob_idの場合はビューがemployeesにアクセスできないため、正常に更新できません。

```
CREATE VIEW clerk AS
   SELECT employee_id, last_name, department_id, job_id
   FROM employees
   WHERE job_id = 'PU_CLERK'
      or job_id = 'SH_CLERK'
      or job_id = 'ST_CLERK'
   WITH CHECK OPTION;
```

結合ビューの作成: 例

結合ビューは、結合を含むビューの副問合せを持つビューです。副問合せの結合において、一意索引を持つ列が1列以上ある場合、結合ビューで1つの実表を変更できます。結合ビューの中の列が更新可能であるかどうかは、USER_UPDATABLE_COLUMNSを問い合せることでわかります。たとえば:

```
CREATE VIEW locations_view AS
   SELECT d.department_id, d.department_name, l.location_id, l.city
   FROM departments d, locations l
   WHERE d.location_id = l.location_id;
SELECT column_name, updatable
   FROM user_updatable_columns
   WHERE table_name = 'LOCATIONS_VIEW'
   ORDER BY column_name, updatable;
COLUMN_NAME                    UPD
------------------------------ ---
DEPARTMENT_ID                  YES
DEPARTMENT_NAME                YES
LOCATION_ID                    NO
```

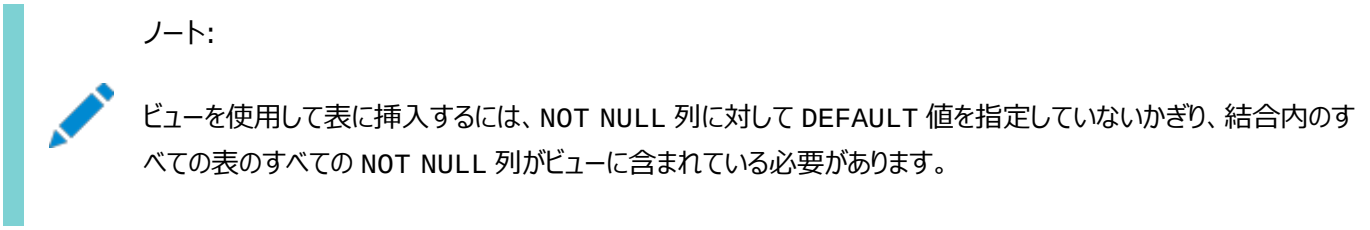
```
CITY                           NO
```

前述の例では、locations表のlocation_id列に対する主キー索引は、locations_viewビュー内で一意ではありません。そのため、locationsはキー保存表ではなく、その実表の列は更新できません。

```
INSERT INTO locations_view VALUES
   (999, 'Entertainment', 87, 'Roma');
INSERT INTO locations_view VALUES
*
ERROR at line 1:
ORA-01776: cannot modify more than one base table through a join view
```

departments表にマップするビュー内のすべての列に更新可能のマークが付けられていて、departmentsの主キーがビューに保持されているため、departments実表に対して、行の挿入、更新または削除を実行できます。

```
INSERT INTO locations_view (department_id, department_name)
   VALUES (999, 'Entertainment');
1 row created.
```



ノート:

ビューを使用して表に挿入するには、NOT NULL 列に対して DEFAULT 値を指定していないかぎり、結合内のすべての表のすべての NOT NULL 列がビューに含まれている必要があります。

関連項目:

結合ビューの更新の詳細は、『Oracle Database管理者ガイド』を参照してください。

読取り専用ビューの作成: 例

次の文は、oe.customers表のcustomer_roという読取り専用ビューを作成します。顧客の姓、言語、掛貸限度額のみがこのビューで参照できます。

```
CREATE VIEW customer_ro (name, language, credit)
      AS SELECT cust_last_name, nls_language, credit_limit
      FROM customers
      WITH READ ONLY;
```

オブジェクト・ビューの作成: 例

次の例は、ocスキーマ内でのinventory_typ型の作成方法、およびその型を基にしたoc_inventoriesビューの作成方法を示しています。

```
CREATE TYPE inventory_typ
 OID '82A4AF6A4CD4656DE034080020E0EE3D'
 AS OBJECT
    ( product_id          NUMBER(6)
    , warehouse           warehouse_typ
    , quantity_on_hand    NUMBER(8)
    ) ;
/
CREATE OR REPLACE VIEW oc_inventories OF inventory_typ
 WITH OBJECT OID (product_id)
 AS SELECT i.product_id,
           warehouse_typ(w.warehouse_id, w.warehouse_name, w.location_id),
           i.quantity_on_hand
    FROM inventories i, warehouses w
```

```
    WHERE i.warehouse_id=w.warehouse_id;
```

XMLType表のビューの作成: 例

次の例は、XMLType表xwarehouses (「例」で作成)の通常のビューを作成します。

```
CREATE VIEW warehouse_view AS
   SELECT VALUE(p) AS warehouse_xml
   FROM xwarehouses p;
```

このビューからは、次のように選択します。

```
SELECT e.warehouse_xml.getclobval()
   FROM warehouse_view e
   WHERE EXISTSNODE(warehouse_xml, '//Docks') =1;
```

XMLTypeビューの作成: 例

オブジェクト・リレーショナル表に、XMLTypeビューを作成する場合もあります。次の例は、サンプル表oe.warehouses内のXMLType列warehouse_specと同様のオブジェクト・リレーショナル表を作成し、その表のXMLTypeビューを作成します。

```
CREATE TABLE warehouse_table
(
  WarehouseID       NUMBER,
  Area              NUMBER,
  Docks             NUMBER,
  DockType          VARCHAR2(100),
  WaterAccess       VARCHAR2(10),
  RailAccess        VARCHAR2(10),
  Parking           VARCHAR2(20),
  VClearance        NUMBER
);
INSERT INTO warehouse_table
   VALUES(5, 103000,3,'Side Load','false','true','Lot',15);
CREATE VIEW warehouse_view OF XMLTYPE
 XMLSCHEMA "http://www.example.com/xwarehouses.xsd"
    ELEMENT "Warehouse"
    WITH OBJECT ID
    (extract(OBJECT_VALUE, '/Warehouse/Area/text()').getnumberval())
 AS SELECT XMLELEMENT("Warehouse",
            XMLFOREST(WarehouseID as "Building",
                      area as "Area",
                      docks as "Docks",
                      docktype as "DockType",
                      wateraccess as "WaterAccess",
                      railaccess as "RailAccess",
                      parking as "Parking",
                      VClearance as "VClearance"))
  FROM warehouse_table;
```

このビューは、次のように問い合せます。

```
SELECT VALUE(e) FROM warehouse_view e;
```

DELETE

目的

DELETE文を使用すると、次の表から行を削除できます。

- 非パーティション表またはパーティション表
- ビューの非パーティション実表またはパーティション実表
- 書込み可能なマテリアライズド・ビューの非パーティション・コンテナ表またはパーティション・コンテナ表
- 更新可能なマテリアライズド・ビューの非パーティション・マスター表またはパーティション・マスター表

前提条件

表から行を削除する場合、その表が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、その表に対するDELETEオブジェクト権限が必要です。

更新可能なマテリアライズド・ビューから行を削除する場合、そのマテリアライズド・ビューが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、そのマテリアライズド・ビューに対するDELETEオブジェクト権限が必要です。

ビューの実表から行を削除する場合、そのビューが含まれるスキーマの所有者には、その実表に対するDELETEオブジェクト権限が必要です。また、他のスキーマ内にビューが存在している場合は、そのビューに対するDELETEオブジェクト権限が必要です。

DELETE ANY TABLEシステム権限を持っている場合、任意の表、表パーティションまたは任意のビューの実表から行を削除できます。

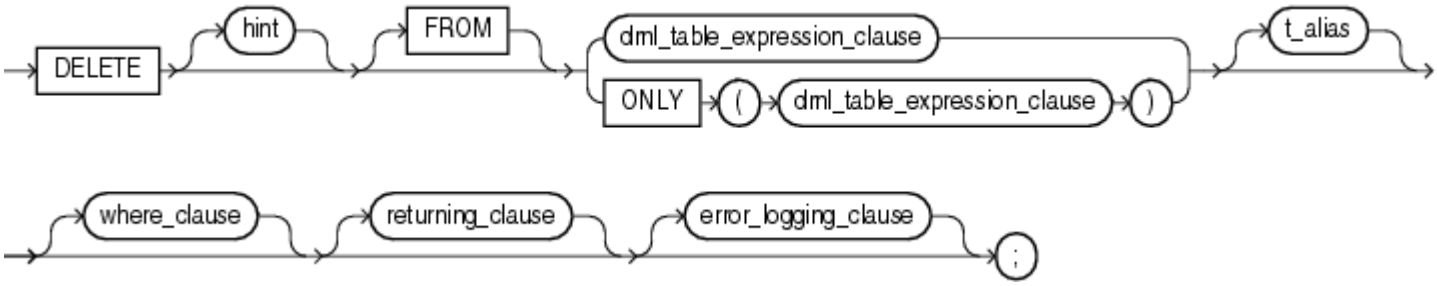
リモート・データベースのオブジェクトの行を削除するには、オブジェクトのREADまたはSELECTオブジェクト権限も必要です。

SQL92_SECURITY初期化パラメータがTRUEに設定され、DELETE操作がwhere_clauseの列などの表の列を参照する場合、行を削除するオブジェクトのSELECTオブジェクト権限も必要です。

表のファンクション索引が無効な状態にあるとき、表から行を削除できません。まず、ファンクション索引を検証する必要があります。

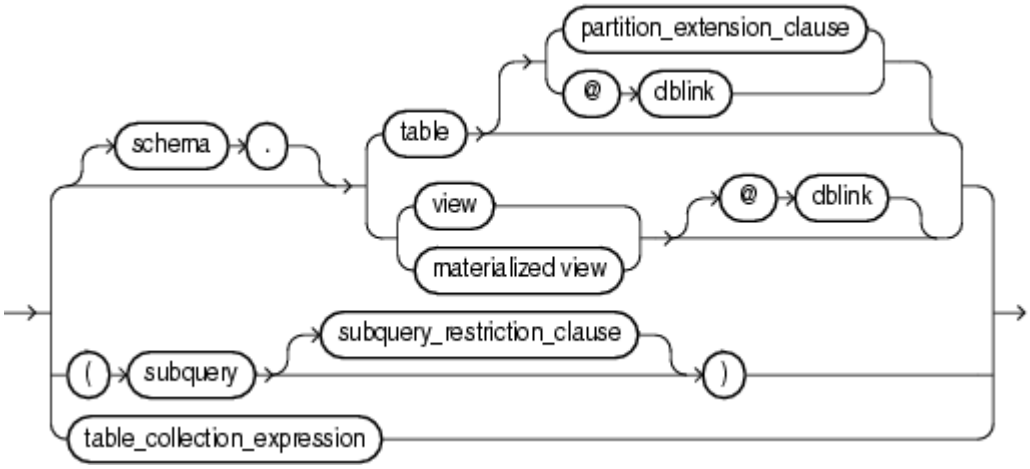
構文

delete::=



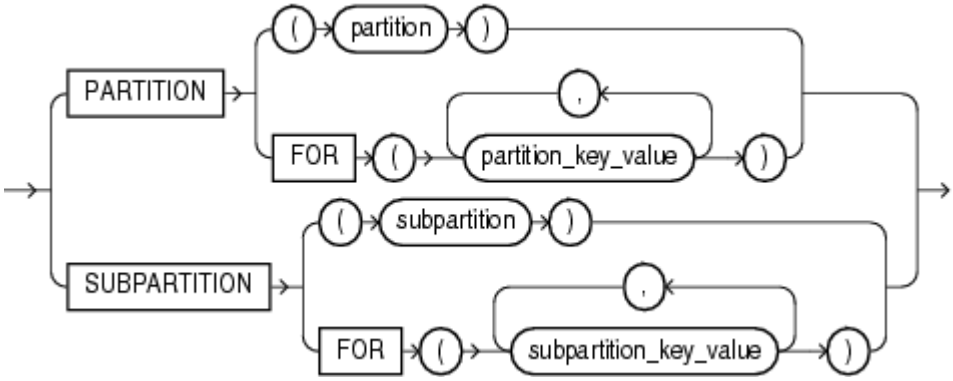
(DML_table_expression_clause::=、where_clause::=、returning_clause::=、error_logging_clause::=)

DML_table_expression_clause::=を参照

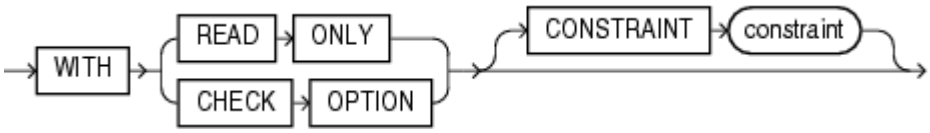


(partition_extension_clause::=、subquery::=、subquery_restriction_clause::=、table_collection_expression::=)

partition_extension_clause::=



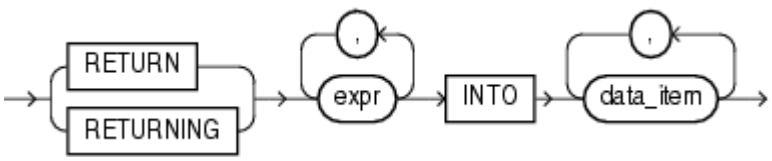
subquery_restriction_clause::=



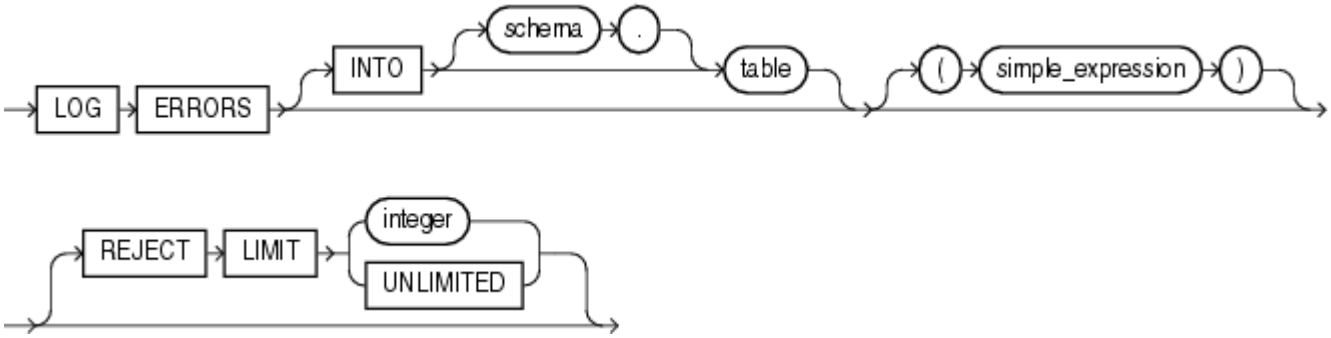
table_collection_expression::=

where_clause::=

returning_clause::=



error_logging_clause::=



セマンティクス

hint

文の実行計画を選択する場合に、オプティマイザに指示を与えるためのコメントを指定します。

関連項目:

ヒントの構文および説明は、「ヒント」を参照してください。

from_clause

FROM句を使用すると、行を削除するデータベース・オブジェクトを指定できます。

ONLY構文は、ビューのみに関連します。FROM句のビューがビューの階層に属し、そのいずれのサブビューからも行を削除しない場合は、ONLY句を使用します。

DML_table_expression_clause

この句を使用すると、データを削除するオブジェクトを指定できます。

schema

表またはビューが含まれているスキーマを指定します。schemaを指定しない場合、表またはビューは自分のスキーマにあるとみなされます。

table | view | materialized view | subquery

行を削除する表、ビュー、マテリアライズド・ビュー、列または副問合せの結果の列の名前を指定します。

更新可能なビューから行を削除すると、実表から行が削除されます。

読取り専用マテリアライズド・ビューからは行を削除できません。書込み可能なマテリアライズド・ビューから行を削除する場合、基礎となるコンテナ表から行が削除されます。ただし、その削除は次のリフレッシュ操作で上書きされます。マテリアライズド・ビュー・グループ内の更新可能なマテリアライズド・ビューから行を削除する場合、マスター表の対応する行も削除されます。

table、viewの実表またはmaterialized_viewのマスター表に、1列以上のドメイン索引列が含まれる場合は、この文によって適切な索引タイプの削除ルーチンが実行されます。

関連項目:

これらのルーチンの詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。

表に対してDELETE文を実行すると、その表に定義されているDELETEトリガーが起動されます。

行の削除によって解放される表や索引のすべての領域は、表および索引によって引き続き保持されます。

partition_extension_clause

オブジェクト内にある削除対象のパーティションまたはサブパーティションの名前またはパーティション・キー値を指定します。

パーティション・オブジェクトから値を削除する場合、そのパーティション名を指定する必要はありません。ただし、パーティション名を指定した方が、複雑なwhere_clauseより効率が上がることがあります。

関連項目:

「パーティション表と索引の参照」および「パーティションからの行の削除: 例」を参照してください。

dblink

オブジェクトが格納されているリモート・データベースへのデータベース・リンクの完全名または部分名を指定します。Oracle Databaseの分散機能を使用している場合にのみ、リモート・オブジェクトから行を削除できます。

ノート:

Oracle Database 12c リリース 2 (12.2)以降、DELETE 文は、バインド変数としてリモート LOB ロケータを受け入れます。詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』の「分散LOB」の章を参照してください。

関連項目:

データベース・リンクの参照方法に関する「リモート・データベース内のオブジェクトの参照」、および「リモート・データベースからの行の削除: 例」を参照してください。

dblinkを省略した場合、オブジェクトがローカル・データベース上にあるとみなされます。

subquery_restriction_clause

subquery_restriction_clauseを使用すると、次のいずれかの方法で副問合せを制限できます。

WITH READ ONLY

WITH READ ONLYを指定すると、表またはビューを更新禁止にできます。

WITH CHECK OPTION

WITH CHECK OPTIONを指定すると、副問合せに含まれない行を生成する表またはビューの変更を禁止できます。この句をDML文の副問合せ内で使用する場合、FROM句内の副問合せには指定できますが、WHERE句内の副問合せには指定できません。

CONSTRAINT constraint

CHECK OPTION制約の名前を指定します。この識別子を省略した場合は、Oracleによって自動的にSYS_Cnという形式の制約名が割り当てられます(nはデータベース内で制約名を一意にするための整数)。

関連項目:

WITH CHECK OPTION句の使用方法: 例

table_collection_expression

table_collection_expressionを使用すると、問合せおよびDML操作で、collection_expression値を表として扱うことができます。collection_expressionには、副問合せ、列、ファンクションまたはコレクション・コンストラクタのいずれかを指定できます。その形式にかかわらず、集合値(ネストした表型またはVARRAY型の値)を戻す必要があります。このようなコレクションの要素抽出プロセスをコレクション・ネスト解除といいます。

TABLEコレクション式を親表と結合する場合は、オプションのプラス(+)には大きな意味があります。+を指定すると、その2つの外部結合が作成され、コレクション式がNULLの場合でも、外部表の行が問合せで戻されるようになります。

ノート:

以前のリリースの Oracle では、collection_expression が副問合せの場合、table_collection_expression を THE subquery と表現していました。現在、このような表現方法は非推奨になっています。

相関副問合せでtable_collection_expressionを使用すると、他の表に存在する値で行を削除できます。

関連項目:

「表のコレクション: 例」

collection_expression

行を削除するオブジェクトからネストした表の列を選択する副問合せを指定します。

dml_table_expression_clause句の制限事項

この句には、次の制限事項があります。

- viewまたはmaterialized_viewのtable(実表またはマスター表)にIN_PROGRESSまたはFAILEDのマークが付いたドメイン索引がある場合、この文は実行できません。
- 関係する索引パーティションがUNUSABLEとマークされている場合は、パーティションに挿入できません。
- DML_table_expression_clauseの副問合せでは、ORDER BY句を指定できません。
- ビューを定義する問合せに次のいずれかの構造体が含まれている場合は、INSTEAD OFトリガーを使用する場合を除いて、ビューから行を削除できません。
    - 集合演算子
    - DISTINCT演算子
    - 集計ファンクションまたは分析ファンクション
    - GROUP BY、ORDER BY、MODEL、CONNECT BYまたはSTART WITH句
    - SELECT構文のリストにあるコレクション式
    - SELECT構文のリストにある副問合せ
    - WITH READ ONLYが指定された副問合せ
    - 結合(一部の例外を除く。詳細は、『Oracle Database管理者ガイド』を参照してください。)

UNUSABLEのマークが付いている索引、索引パーティションまたは索引サブパーティションを指定する場合、SKIP_UNUSABLE_INDEXES初期化パラメータがtrueに設定されていないかぎり、DELETE文は正常に実行されません。

関連項目:

ALTER SESSION

t_alias

文中で参照する表、ビュー、マテリアライズド・ビュー、副問合せまたはコレクション値の相関名を指定します。DML_table_expression_clauseがいずれかのオブジェクト型属性またはオブジェクト型メソッドを参照する場合、この別名が必要です。通常、別名は相関問合せを持つDELETE文で使用されます。

where_clause

where_clauseを使用すると、条件を満たす行のみを削除できます。この条件は、行を削除するオブジェクトを参照したり、副問合せを含むことができます。Oracle Databaseの分散機能を使用している場合にのみ、リモート・オブジェクトから行を削除できます。conditionの構文は、「条件」を参照してください。

この句がリモート・オブジェクトを参照するsubqueryを含む場合、参照がローカル・データベース上でオブジェクトにループバックしないかぎり、DELETEはパラレルで実行されます。ただし、DML_table_expression_clauseのsubqueryがリモート・オブジェクトを参照する場合は、DELETE操作はシリアルで実行されます。詳細は、「CREATE TABLE」の「parallel_clause」を参照してください。

dblinkを省略した場合、表またはビューがローカル・データベース上にあるとみなされます。

where_clauseを省略した場合、オブジェクトのすべての行が削除されます。

returning_clause

この句を使用すると、削除された列から値を戻すことができるため、DELETE文の後でSELECT文を実行する必要がなくなります。

この句を使用すると、DML文に影響される行を取り出すことができます。この句は、表、マテリアライズド・ビュー、および単一の実表を持つビューに指定できます。

returning_clauseを指定したDML文を単一行に実行すると、影響された行、ROWID、および処理された行へのREFを使用している列式が取り出され、ホスト変数またはPL/SQL変数に格納されます。

returning_clauseを指定したDML文を複数行に実行すると、式の値、ROWIDおよび処理された行に関連するREFがバインド配列に格納されます。

expr

exprリストの各項目は、適切な構文で表す必要があります。

INTO

INTO句を指定すると、変更された行の値を、data_itemリストに指定する変数に格納できます。

data_item

data_itemはそれぞれ、取り出したexpr値を格納するためのホスト変数またはPL/SQL変数です。

RETURNINGリストの各式については、INTOリストに、対応する型に互換性があるPL/SQL変数またはホスト変数を指定する必要があります。

RETURNING句の制限事項

RETURNING句には、次の制限事項があります。

- exprに次の制限事項があります。
  - UPDATE文およびDELETE文の場合、各exprは、単純式または単一セットの集計ファンクション式である必要があります。1つのreturning_clause内に単純式と単一セットの集計ファンクション式を混在させることはできません。INSERT文の場合、各exprは単純式である必要があります。INSERT文のRETURNING句では、集計ファンクションはサポートされていません。
  - 単一セットの集計ファンクション式をDISTINCTキーワードに含めることはできません。
- exprのリストに主キー列または他のNOT NULL列が含まれている場合は、表にBEFORE UPDATEトリガーが定義されていると、UPDATE文の実行に失敗します。
- マルチテーブル・インサートではreturning_clauseを指定できません。
- パラレルDMLまたはリモート・オブジェクトにはこの句を使用できません。
- LONG型を取り出すことはできません。
- INSTEAD OFトリガーが定義されたビューに対して指定することはできません。

関連項目:

- BULK COLLECT句を使用してコレクション変数に複数の値を戻す場合は、『Oracle Database PL/SQL言語リファレンス』を参照してください。
- RETURNING句の使用例

error_logging_clause

error_logging_clauseのDELETE文での動作は、INSERT文の場合と同じです。詳細は、「INSERT」文の「error_logging_clause」を参照してください。

関連項目:

エラー・ロギングによる表への挿入: 例

例

行の削除: 例

次の文は、language_id列の値がARの、サンプル表oe.product_descriptionsからすべての行を削除します。

```
DELETE FROM product_descriptions
   WHERE language_id = 'AR';
```

次の文は、サンプル表hr.employeesから歩合率が10%未満の購買係を削除します。

```
DELETE FROM employees
   WHERE job_id = 'SA_REP'
   AND commission_pct < .2;
```

次の文は前述の例と同じ結果を表しますが、副問合せを使用します。

```
DELETE FROM (SELECT * FROM employees)
```

```
   WHERE job_id = 'SA_REP'
   AND commission_pct < .2;
```

リモート・データベースからの行の削除: 例

次の文は、データベース・リンクremoteからアクセス可能なデータベース上のユーザーhrが所有するlocations表から、指定された行を削除します。

```
DELETE FROM hr.locations@remote
   WHERE location_id > 3000;
```

ネストした表の行の削除: 例

ネストした表の行の削除例は、「表のコレクション: 例」を参照してください。

パーティションからの行の削除: 例

次の例は、sh.sales表のパーティションsales_q1_1998から行を削除します。

```
DELETE FROM sales PARTITION (sales_q1_1998)
   WHERE amount_sold > 1000;
```

RETURNING句の使用: 例

次の例は、削除された行からsalary列を戻し、その結果をバインド配列:bnd1に格納します。バインド配列は事前に宣言しておく必要があります。

```
DELETE FROM employees
   WHERE job_id = 'SA_REP'
   AND hire_date + TO_YMINTERVAL('01-00') < SYSDATE
   RETURNING salary INTO :bnd1;
```

表からのデータの削除: 例

次の文は、product_price_historyという名前の表を作成し、この表にデータを挿入します。

```
CREATE TABLE product_price_history (
  product_id          INTEGER NOT NULL,
  price               INTEGER NOT NULL,
  currency_code       VARCHAR2(3 CHAR) NOT NULL,
  effective_from_date DATE NOT NULL,
  effective_to_date   DATE,
  CONSTRAINT product_price_history_pk
    PRIMARY KEY (product_id, currency_code, effective_from_date)
) PARTITION BY RANGE (effective_from_date) (
    PARTITION p0 VALUES less than (DATE'2015-01-02'),
    PARTITION p1 VALUES less than (DATE'2015-01-03'),
    PARTITION p2 VALUES less than (DATE'2015-01-04')
);
INSERT INTO product_price_history
  WITH prices AS (
    SELECT 1, 100, 'USD', DATE'2015-01-01', DATE'2015-01-02'
    FROM   dual UNION ALL
    SELECT 1, 60, 'GBP', DATE'2015-01-01', DATE'2015-01-02'
    FROM   dual UNION ALL
    SELECT 1, 110, 'EUR', DATE'2015-01-01', DATE'2015-01-02'
    FROM   dual UNION ALL
    SELECT 1, 101, 'USD', DATE'2015-01-02', DATE'2015-01-03'
    FROM   dual UNION ALL
    SELECT 1, 62, 'GBP', DATE'2015-01-02', DATE'2015-01-03'
    FROM   dual UNION ALL
    SELECT 1, 109, 'EUR', DATE'2015-01-02', DATE'2015-01-03'
    FROM   dual UNION ALL
    SELECT 1, 105, 'USD', DATE'2015-01-03', NULL
```

```
    FROM   dual UNION ALL
    SELECT 1, 61, 'GBP', DATE'2015-01-03', NULL
    FROM   dual UNION ALL
    SELECT 1, 107, 'EUR', DATE'2015-01-03', NULL
    FROM   dual UNION ALL
    SELECT 2, 30, 'USD', DATE'2015-01-01', DATE'2015-01-03'
    FROM   dual UNION ALL
    SELECT 2, 33, 'USD', DATE'2015-01-03', NULL
    FROM   dual UNION ALL
    SELECT 3, 100, 'GBP', DATE'2015-01-03', NULL
    FROM   dual
  )
SELECT *
FROM   prices;
```

次の文は、表product_price_historyからproduct_idが3の行を削除します。

```
DELETE FROM product_price_history WHERE product_id = 3;
```

次のプロシージャは、product_idが2で、effective_to_dateがNULLの行をproduct_price_historyから削除します。

```
DECLARE
  currency product_price_history.currency_code%TYPE;
BEGIN
  DELETE product_price_history
  WHERE  product_id = 2
  AND    effective_to_date IS NULL
  returning currency_code INTO currency;

  dbms_output.Put_line(currency);
END;
USD
```

次の文は、currency_codeがEURの行を表product_price_historyから削除します。

```
DELETE (SELECT * FROM product_price_history) WHERE  currency_code = 'EUR';
```

次の文は、副問合せを使用してproduct_price_historyから行を削除します。

```
DELETE product_price_history pp
WHERE  (product_id, currency_code, effective_from_date)
   IN (SELECT product_id, currency_code, Max(effective_from_date)
       FROM   product_price_history
       GROUP BY product_id, currency_code);
```

次の文は、パーティションを使用してproduct_price_historyから行を削除します。

```
DELETE product_price_history partition (p1);
```

次の文は、表の情報を表示します。

```
SELECT * FROM product_price_history;
PRODUCT_ID      PRICE CUR EFFECTIVE EFFECTIVE
---------- ---------- --- --------- ---------
         1        100 USD 01-JAN-15 02-JAN-15
         1         60 GBP 01-JAN-15 02-JAN-15
```

次の文は、product_price_historyからすべての行を削除します。

```
DELETE product_price_history;
```

Live SQL:

「表からのデータの削除」で、Oracle Live SQL に関連する例を参照および実行します。

DISASSOCIATE STATISTICS

目的

DISASSOCIATE STATISTICS文を使用すると、デフォルトの統計情報または統計タイプと、列、スタンドアロン・ファンクション、パッケージ、型、ドメイン索引または索引タイプとの関連付けを解除できます。

関連項目:

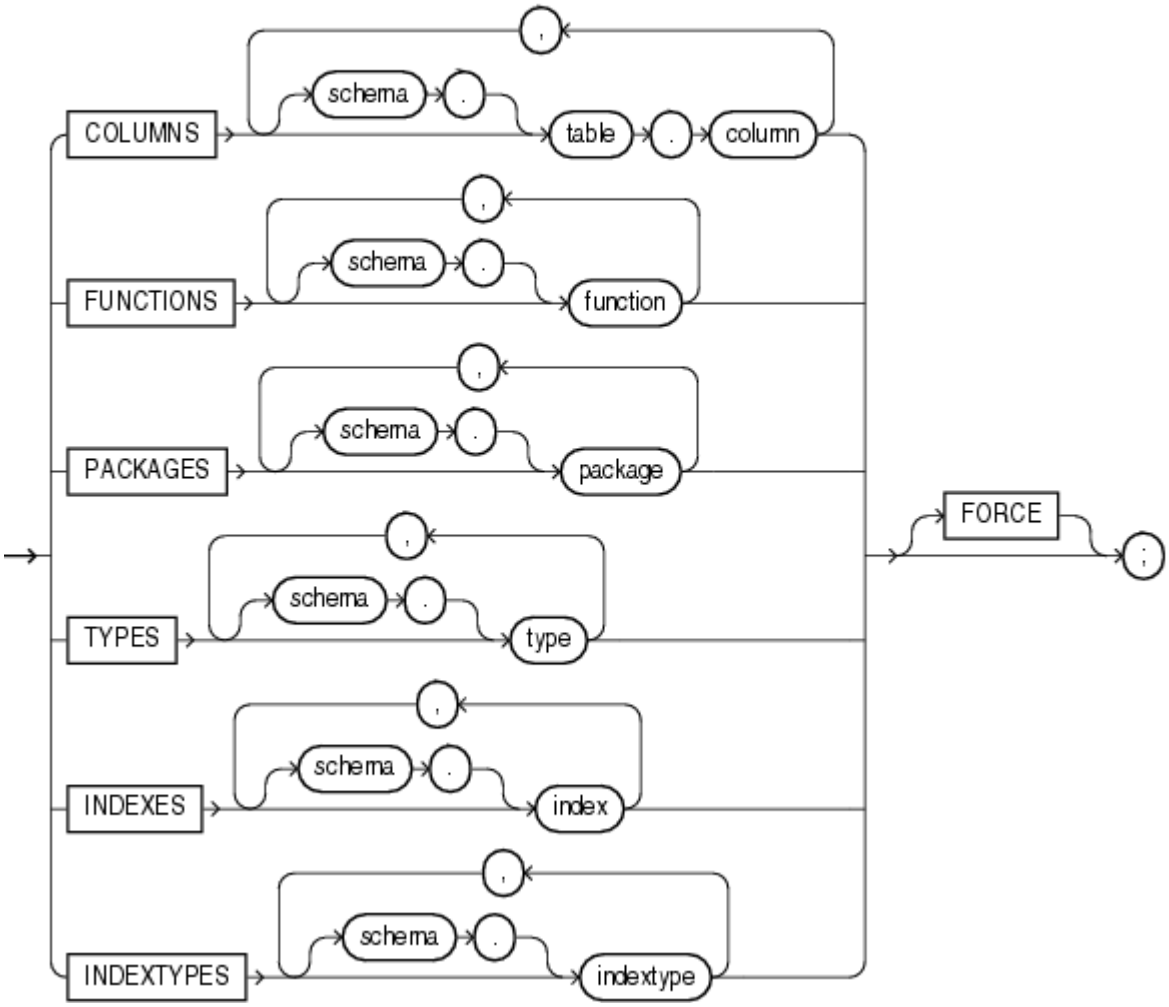
統計タイプの関連付けの詳細は、「ASSOCIATE STATISTICS」を参照してください。

前提条件

この文を発行する場合は、基礎となる表、ファンクション、パッケージ、型、ドメイン索引または索引タイプを変更する適切な権限が必要です。

構文

disassociate_statistics::=



セマンティクス

FROM COLUMNS | FUNCTIONS | PACKAGES | TYPES | INDEXES | INDEXTYPES

統計との関連付けを解除する1つ以上の列、スタンドアロン・ファンクション、パッケージ、型、ドメイン索引または索引タイプを指定します。

schemaを指定しない場合、オブジェクトは自分のスキーマ内にあるとみなされます。

オブジェクトのユーザー定義の統計情報を収集する場合、FORCEを指定しないかぎり文は実行されません。

FORCE

FORCEを指定すると、統計タイプを使用するオブジェクトの統計情報の有無にかかわらず、関連付けが解除されます。統計情報が存在する場合、関連付けが解除される前にその統計情報は削除されます。

ノート:

統計タイプが関連付けられているオブジェクトを削除する場合、FORCE オプションによって統計タイプの関連付けが自動的に解除され、統計タイプを使用して収集したすべての統計情報が削除されます。

例

統計情報の関連付け解除: 例

この文は、統計情報とemp_mgmtパッケージの関連付けを解除します。hrスキーマにこのパッケージを作成する例については、『Oracle Database PL/SQL言語リファレンス』を参照してください。

```
DISASSOCIATE STATISTICS FROM PACKAGES hr.emp_mgmt;
```

DROP ANALYTIC VIEW

目的

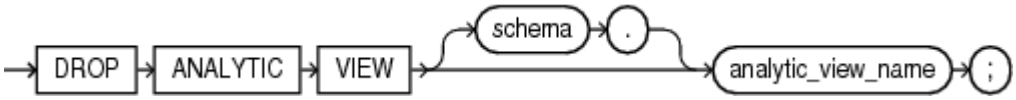
DROP ANALYTIC VIEW文を使用すると、分析ビューを削除できます。ANALYTIC VIEWオブジェクトは、分析ビューのコンポーネントです。

前提条件

自分のスキーマ内の分析ビューを削除する場合は、DROP ANALYTIC VIEWシステム権限が必要です。他のユーザーのスキーマ内の分析ビューを削除する場合は、DROP ANY ANALYTIC VIEWシステム権限が必要です。

構文

drop_analytic_view::=



セマンティクス

schema

分析ビューが存在するスキーマを指定します。スキーマを指定しない場合、自分のスキーマ内で分析ビューが検索されます。

analytic_view_name

削除する分析ビューの名前を指定します。

例

次の文は、指定した分析ビュー・オブジェクトを削除します。

```
DROP ANALYTIC VIEW sales_av;
```

DROP ATTRIBUTE DIMENSION

目的

DROP ATTRIBUTE DIMENSION文を使用すると、属性ディメンションを削除できます。ATTRIBUTE DIMENSIONオブジェクトは、分析ビューのコンポーネントです。

前提条件

自分のスキーマ内の属性ディメンションを削除する場合は、DROP ATTRIBUTE DIMENSIONシステム権限が必要です。他のユーザーのスキーマ内の属性ディメンションを削除する場合は、DROP ANY ATTRIBUTE DIMENSIONシステム権限が必要です。

構文

drop_attribute_dimension::=

セマンティクス

schema

属性ディメンションが存在するスキーマを指定します。スキーマを指定しない場合、自分のスキーマ内で属性ディメンションが検索されます。

attr_dimension_name

削除する属性ディメンションの名前を指定します。

例

次の文は、指定した属性ディメンション・オブジェクトを削除します。

```
DROP ATTRIBUTE DIMENSION product_attr_dim;
```

DROP AUDIT POLICY (統合監査)

この項では、統合監査用のDROP AUDIT POLICY文について説明します。この種類の監査は、Oracle Database 12cで新たに導入されたもので、完全かつ高度な監査機能を提供します。統合監査の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

目的

DROP AUDIT POLICY文を使用すると、データベースから統合監査ポリシーを削除できます。

関連項目:

- CREATE AUDIT POLICY (統合監査)
- ALTER AUDIT POLICY (統合監査)
- AUDIT (統合監査)
- NOAUDIT (統合監査)

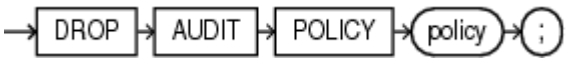
前提条件

AUDIT SYSTEMシステム権限、またはAUDIT_ADMINロールが必要になります。

共通の統合監査ポリシーを削除する場合は、現在のコンテナがルートである必要があります。また、共通に付与されているAUDIT SYSTEM権限またはAUDIT_ADMIN共通ロールが必要です。ローカルの統合監査ポリシーを削除する場合は、現在のコンテナが、その監査ポリシーが作成されたコンテナである必要があります。また、共通に付与されているAUDIT SYSTEM権限またはAUDIT_ADMIN共通ロールを保有しているか、そのコンテナでローカルに付与されているAUDIT SYSTEM権限またはAUDIT_ADMINローカル・ロールを保有している必要があります。

構文

drop_audit_policy::=



セマンティクス

policy

削除する統合監査ポリシーの名前を指定します。このポリシーは、CREATE AUDIT POLICY文を使用して作成されている必要があります。

AUDIT_UNIFIED_POLICIESビューを問い合せて、すべての統合監査ポリシーの名前を検索したり、AUDIT_UNIFIED_ENABLED_POLICIESビューを問い合せて、有効になっているすべての統合監査ポリシーの名前を検索できます。

統合監査ポリシーの削除の制限事項

有効になっている統合監査ポリシーは削除できません。まずNOAUDIT文を使用してポリシーを無効化する必要があります。

関連項目:

- CREATE AUDIT POLICY (統合監査)
- AUDIT_UNIFIED_POLICIESおよびAUDIT_UNIFIED_ENABLED_POLICIESビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

例

統合監査ポリシーの削除：例

次の文では、統合監査ポリシーtable_polを削除します。

```
DROP AUDIT POLICY table_pol;
```

DROP CLUSTER

目的

DROP CLUSTER文を使用すると、データベースからクラスタを削除できます。

ノート:

クラスタを削除すると、ごみ箱内のそのクラスタの一部だった表はごみ箱から消去され、FLASHBACK TABLE 操作でもリカバリできなくなります。

個々の表は非クラスタ化できません。そのかわり、次のステップを実行します。

- 既存の表と同じ構成および内容で、新しい表をCLUSTER句なしで作成します。
- 既存の表を削除します。
- RENAME文を使用して、新しい表に既存の表の名前を指定します。
- 新しいクラスタ化表に対する権限を付与します。古いクラスタ化表に対する権限は使用できません。

関連項目:

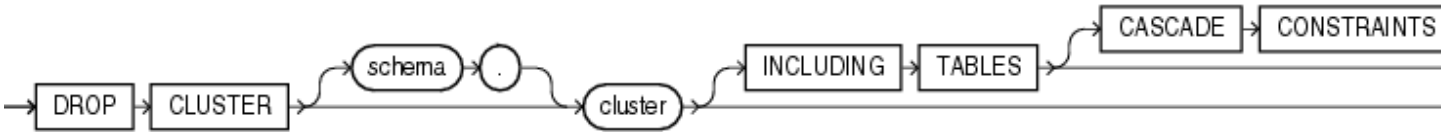
これらのステップの詳細は、「CREATE TABLE」、「DROP TABLE」、「RENAME」および「GRANT」を参照してください。

前提条件

削除するクラスタが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY CLUSTERシステム権限が必要です。

構文

drop_cluster::=



セマンティクス

schema

クラスタが含まれているスキーマを指定します。schemaを指定しない場合、クラスタは自分のスキーマ内にあるとみなされます。

cluster

削除するクラスタの名前を指定します。クラスタを削除するとクラスタの索引も削除され、その索引のデータ・ブロックを含むクラスタ領域が表領域に戻されます。

INCLUDING TABLES

INCLUDING TABLESを指定すると、クラスタに属するすべての表を削除できます。

CASCADE CONSTRAINTS

CASCADE CONSTRAINTSを指定すると、クラスタに含まれる表の主キーまたは一意キーを参照するクラスタ外の表から、すべての参照整合性制約を削除できます。このような参照整合性制約があるときにこの句の指定を省略した場合、エラーが戻され、クラスタは削除されません。

例

クラスタの削除: 例

次の例では、CREATE CLUSTERの「例」で作成したクラスタを削除します。

次の文は、languageクラスタを削除します。

```
DROP CLUSTER language;
```

次の文は、dept_10、dept_20およびこれらの表の主キーまたは一意キーを参照する参照整合性制約とともにpersonnelクラスタを削除します。

```
DROP CLUSTER personnel
   INCLUDING TABLES
   CASCADE CONSTRAINTS;
```

16 SQL文: DROP CONTEXTからDROP JAVA

この章では、次のSQL文について説明します。

- DROP CONTEXT
- DROP DATABASE
- DROP DATABASE LINK
- DROP DIMENSION
- DROP DIRECTORY
- DROP DISKGROUP
- DROP EDITION
- DROP FLASHBACK ARCHIVE
- DROP FUNCTION
- DROP HIERARCHY
- DROP INDEX
- DROP INDEXTYPE
- DROP INMEMORY JOIN GROUP
- DROP JAVA

DROP CONTEXT

目的

DROP CONTEXT文を使用すると、データベースからコンテキスト・ネームスペースを削除できます。

コンテキスト・ネームスペースを削除した場合でも、ユーザー・セッションに設定されたネームスペースのコンテキストは無効になりません。ただし、ユーザーがそのコンテキストを設定しようとした場合、そのコンテキストは無効になります。

関連項目:

コンテキストの詳細は、「CREATE CONTEXT」および『Oracle Databaseセキュリティ・ガイド』を参照してください。

前提条件

DROP ANY CONTEXTシステム権限が必要です。

構文

drop_context::=

セマンティクス

namespace

削除するコンテキスト・ネームスペースの名前を指定します。組込みネームスペースUSERENVは削除できません。

関連項目:

USERENVネームスペースの詳細は、「SYS_CONTEXT」を参照してください。

例

アプリケーション・コンテキストの削除: 例

次の文は、CREATE CONTEXTで作成したコンテキストを削除します。

```
DROP CONTEXT hr_context;
```

DROP DATABASE

目的

ノート:

DROP DATABASE 文はロールバックできません。

DROP DATABASE文を使用すると、データベースを削除できます。この文は、テスト・データベースを削除したり、新しいホストへの移行が完了した後に古いデータベースを削除する場合に便利です。

関連項目:

データベースの削除の詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

前提条件

この文を発行するには、SYSDBAシステム権限が必要です。データベースは、排他モードおよび制限モードでマウントされている必要があります。また、データベースはクローズ状態である必要があります。

構文

drop_database::=

セマンティクス

この文を発行すると、データベースを削除し、すべての制御ファイル、およびその制御ファイルに記述されているデータファイルを削除できます。サーバー・パラメータ・ファイル(SPFILE)を使用している場合は、それも削除されます。

アーカイブ・ログおよびバックアップは削除されません。これらを削除するには、Recovery Manager(RMAN)を使用します。データベースがRAWディスク上に存在する場合、この文では実際のRAWディスクの特殊ファイルは削除されません。

DROP DATABASE LINK

目的

DROP DATABASE LINK文を使用すると、データベースからデータベース・リンクを削除できます。

関連項目:

データベース・リンクの作成方法については、「CREATE DATABASE LINK」を参照してください。

前提条件

プライベート・データベース・リンクは自分のスキーマにある必要があります。パブリック・データベース・リンクを削除する場合は、DROP PUBLIC DATABASE LINKシステム権限が必要です。

構文

drop_database_link::=

セマンティクス

PUBLIC

PUBLICを指定すると、PUBLICデータベース・リンクを削除できます。

dblink

削除するデータベース・リンクの名前を指定します。

データベース・リンクの削除の制限事項

他のユーザーのスキーマ内のデータベース・リンクは削除できません。また、データベース・リンク名ではピリオドが許可されるため、スキーマ名でdblinkを修飾することはできません。そのため、ralph.linktosalesのような名前を付けた場合、スキーマralphの中のlinktosalesという名前のデータベース・リンクであるとはみなされず、名前全体が自分のスキーマにあるデータベース・リンク名であると解析されます。

例

データベース・リンクの削除: 例

次の文は、パブリック・データベース・リンクremote (「パブリック・データベース・リンクの定義: 例」で作成)を削除します。

```
DROP PUBLIC DATABASE LINK remote;
```

DROP DIMENSION

目的

DROP DIMENSION文を使用すると、指定したディメンションを削除できます。

この文によって、ディメンションに指定された関連を使用するマテリアライズド・ビューが無効になるわけではありません。ただし、クエリー・リライトによってリライトされた要求が無効になり、そのようなビューに対する後続の操作の処理速度が遅くなることがあります。

関連項目:

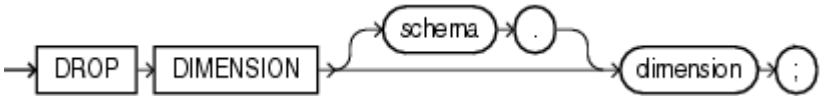
- ディメンションの作成および変更については、「CREATE DIMENSION」および「ALTER DIMENSION」を参照してください。
- ディメンションの概要は、『Oracle Database概要』を参照してください。

前提条件

この文を使用する場合、ディメンションが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY DIMENSIONシステム権限が必要です。

構文

drop_dimension::=



セマンティクス

schema

ディメンションが格納されているスキーマの名前を指定します。schemaを指定しない場合、ディメンションは自分のスキーマ内にあるとみなされます。

dimension

削除するディメンションの名前を指定します。そのディメンションはすでに存在している必要があります。

例

ディメンションの削除: 例

この例では、sh.customers_dimディメンションを削除します。

```
DROP DIMENSION customers_dim;
```

関連項目:

ディメンションの作成および変更の例については、「ディメンションの作成: 例」および「ディメンションの変更: 例」を参照してください。

DROP DIRECTORY

目的

DROP DIRECTORY文を使用すると、データベースからディレクトリ・オブジェクトを削除できます。

関連項目:

ディレクトリの作成については、「CREATE DIRECTORY」を参照してください。

前提条件

ディレクトリを削除する場合は、DROP ANY DIRECTORYシステム権限が必要です。

ノート:

PL/SQL または OCI プログラムによる関連ファイル・システム内のファイルへのアクセス中は、DROP によるディレクトリの削除はできません。

構文

drop_directory::=

セマンティクス

directory_name

削除するディレクトリ・データベース・オブジェクトの名前を指定します。

Oracle Databaseはディレクトリ・オブジェクトを削除しますが、サーバーのファイル・システム上の関連するオペレーティング・システム・ディレクトリは削除しません。

例

ディレクトリの削除: 例

次の文は、ディレクトリ・オブジェクトbfile_dirを削除します。

```
DROP DIRECTORY bfile_dir;
```

関連項目:

ディレクトリの作成: 例

DROP DISKGROUP

ノート:

この SQL 文は、Oracle ASM を使用しており、Oracle ASM インスタンスを起動している場合にのみ有効です。この文の発行は、通常のデータベース・インスタンスからではなく、Oracle ASM インスタンスから行う必要があります。Oracle ASM インスタンスの起動の詳細は、『Oracle Automatic Storage Management 管理者ガイド』を参照してください。

目的

DROP DISKGROUP文を使用すると、Oracle ASMディスク・グループおよびそのディスク・グループのすべてのファイルを削除できます。Oracle ASMでは、最初にディスク・グループのファイルが開かれていないことが確認されます。次に、ディスク・グループおよびそのメンバー・ディスクがすべて削除され、ディスク・ヘッダーがクリアされます。

関連項目:

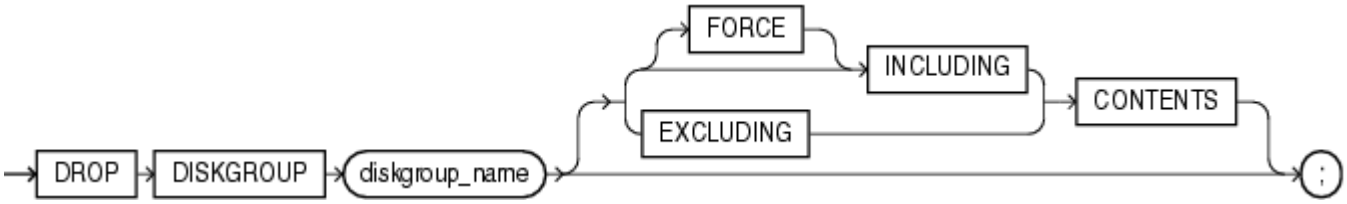
- ディスク・グループの作成および変更については、「CREATE DISKGROUP」および「ALTER DISKGROUP」を参照してください。
- Oracle ASMおよびディスク・グループを使用してデータベース管理を簡略化する方法については、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

前提条件

SYSASMシステム権限が必要です。また、この文を発行するOracle ASMインスタンスが起動されている必要があります。削除するディスク・グループはマウントされている必要があります。

構文

drop_diskgroup::=



セマンティクス

diskgroup_name

削除するディスク・グループの名前を指定します。

INCLUDING CONTENTS

INCLUDING CONTENTSを指定すると、Oracle ASMによってディスク・グループのすべてのファイルが削除されます。ディスク・グループにファイルが含まれている場合、この句を指定する必要があります。

FORCE

この句では、Oracle ASMインスタンスではマウントできないディスク・グループに属するディスク上のヘッダーがクリアされます。ディスク・グループは、データベースのインスタンスからはマウントできません。

Oracle ASMインスタンスは、ディスク・グループが、同じストレージ・サブシステムを使用する他のOracle ASMで使用されているかどうかをまず確認します。ディスク・グループが使用されており、このディスク・グループが同じクラスタまたは同じノードにあるときは、文の実行に失敗します。ディスク・グループが異なるクラスタにある場合は、さらに、そのディスク・グループが別のクラスタ内のインスタンスによってマウントされているかどうかがチェックされます。ディスク・グループがマウントされている場合、この文の実行は失敗します。ただし、後者のチェックは、同じクラスタのディスク・グループのチェックほど確実なものではありません。そのため、この句の使用には注意が必要です。

EXCLUDING CONTENTS

`EXCLUDING CONTENTS`を指定すると、Oracle ASMによって空のディスク・グループのみが削除されます。これはデフォルトです。ディスク・グループが空ではない場合、エラーが戻されます。

例

ディスク・グループの削除: 例

次の文は、Oracle ASMディスク・グループ`dgroup_01` (「ディスク・グループの作成: 例」で作成)およびそのディスク・グループ内のすべてのファイルを削除します。

```
DROP DISKGROUP dgroup_01 INCLUDING CONTENTS;
```

DROP EDITION

目的

DROP EDITION文を使用すると、エディション、およびそれに含まれる実際のエディション化可能なすべてのオブジェクトを削除できます。実際のエディション化可能なオブジェクトとは、エディション内で作成または変更されたエディション化可能なオブジェクトのことです。

関連項目:

エディション化可能なオブジェクト型のリストについては、「CREATE EDITION」を参照してください。

前提条件

直接またはロールを介して付与されたDROP ANY EDITIONシステム権限が必要です。

構文

drop_edition::=

セマンティクス

エディション化が可能でないオブジェクト、またはエディション化が可能でも現行のエディションで実体化されていないオブジェクトは、削除されません。

指定したエディションに実際のエディション化可能なオブジェクトが含まれている場合は、CASCADEを指定する必要があります。

この文には、次の条件および制限事項があります。

- 指定したエディションに、親エディションと子エディションの両方は存在できません。
- デフォルト・エディションを削除しようとすると、DROP EDITIONは失敗します。
- ルート・エディションを削除しようとしたときに、そのエディションから以前に削除されたオブジェクトがごみ箱に少なくとも1つ存在する場合、DROP EDITIONは失敗します。このような状況では、DROP EDITION CASCADEも失敗します。この場合、PURGE DBA_RECYCLEBIN文を使用してごみ箱内のすべてのオブジェクトを消去してから、エディションを削除してください。詳細は、「PURGE」を参照してください。

  同様に、リーフ・エディションを削除しようとしたときに、そのエディションから以前に削除されたオブジェクトがごみ箱に少なくとも1つ存在する場合、DROP EDITIONは失敗します。ただし、このような状況でも、DROP EDITION CASCADEは成功します。

  エディション化されたオブジェクトがごみ箱に入れられるのは、そのオブジェクトがトリガーである場合のみです。

関連項目:

- Oracle Database開発ガイド
- Oracle Database PL/SQLパッケージおよびタイプ・リファレンス

例

この文の使用例については、「CREATE EDITION」を参照してください。

DROP FLASHBACK ARCHIVE

目的

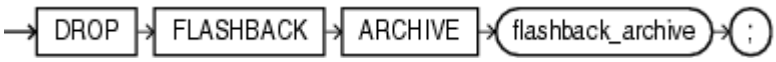
DROP FLASHBACK ARCHIVE句は、システムからフラッシュバック・アーカイブを削除する場合に使用します。この文では、フラッシュバック・アーカイブとこの中にあるすべての履歴データが削除されますが、フラッシュバック・アーカイブによって使用されている表領域は削除されません。

前提条件

フラッシュバック・アーカイブを削除するには、FLASHBACK ARCHIVE ADMINISTERシステム権限が必要です。

構文

drop_flashback_archive::=



セマンティクス

flashback_archive

削除するフラッシュバック・アーカイブの名前を指定します。

関連項目:

フラッシュバック・アーカイブの作成の詳細およびフラッシュバック・アーカイブの簡単な使用例は、CREATE FLASHBACK ARCHIVEを参照してください。

DROP FUNCTION

目的

ファンクションはPL/SQLを使用して定義されます。ファンクションの作成、変更および削除の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

DROP FUNCTION文を使用すると、データベースからスタンドアロン・ストアド・ファンクションを削除できます。

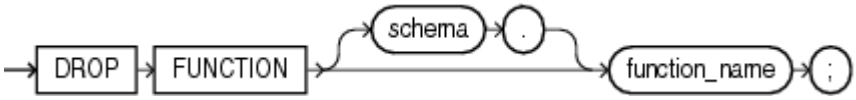
> ノート:
>
> この文を使用してパッケージの一部のファンクションを削除しないでください。かわりに、DROP PACKAGE 文を使用してパッケージ全体を削除するか、または OR REPLACE 句を指定した CREATE PACKAGE 文を使用して、そのファンクションを含めずにパッケージを再定義してください。

前提条件

削除するファンクションが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY PROCEDURE システム権限が必要です。

構文

drop_function::=



セマンティクス

schema

ファンクションが含まれているスキーマを指定します。schemaを指定しない場合、ファンクションは自分のスキーマ内にあるとみなされます。

function_name

削除するファンクションの名前を指定します。

Oracle Databaseは、削除したファンクションに依存するローカル・オブジェクト、または削除したファンクションをコールするローカル・オブジェクトを無効にします。後でこれらのオブジェクトのいずれかを参照した場合、データベースは、そのオブジェクトを再コンパイルしようとします。削除したファンクションを再作成しないかぎり、エラーが戻されます。

任意の統計タイプがファンクションに関連付けられている場合、FORCE句によって統計タイプの関連付けが解除され、統計タイプを使用して収集したユーザー定義のすべての統計情報が削除されます。

関連項目:

- リモート・オブジェクトを含むスキーマ・オブジェクト間の依存性をOracle Databaseが管理する方法の詳細は、『Oracle Database概要』を参照してください。
- 統計タイプの関連付けの詳細は、「ASSOCIATE STATISTICS」および「DISASSOCIATE STATISTICS」を参

照してください。

例

ファンクションの削除：例

次の文は、サンプル・スキーマoe内のファンクションSecondMaxを削除し、SecondMaxに依存するすべてのオブジェクトを無効にします。

```
DROP FUNCTION oe.SecondMax;
```

関連項目:

SecondMaxファンクションを作成する例については、『Oracle Database PL/SQL言語リファレンス』を参照してください。

DROP HIERARCHY

目的

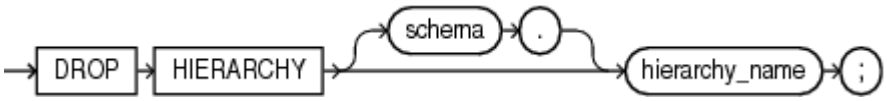
DROP HIERARCHY文を使用すると、階層を削除できます。HIERARCHYオブジェクトは、分析ビューのコンポーネントです。

前提条件

自分のスキーマ内の階層を削除する場合は、DROP HIERARCHYシステム権限が必要です。別のユーザーのスキーマ内にある階層を削除するには、DROP ANY HIERARCHYシステム権限が必要です。

構文

drop_hierarchy::=



セマンティクス

schema

階層が存在するスキーマを指定します。スキーマを指定しない場合、自分のスキーマ内で階層が検索されます。

hierarchy_name

削除する階層の名前を指定します。

例

次の文は、指定した階層オブジェクトを削除します。

```
DROP HIERARCHY product_hier;
```

DROP INDEX

目的

DROP INDEX文を使用すると、データベースから索引またはドメイン索引を削除できます。

グローバル・パーティション索引、レンジ・パーティション索引またはハッシュ・パーティション索引を削除すると、すべての索引パーティションも削除されます。コンポジット・パーティション索引を削除した場合、すべての索引パーティションおよびサブパーティションも削除されます。

また、ドメイン索引を削除すると、次のようになります。

- 適切なルーチンが起動されます。
- 任意の統計情報がドメイン索引に関連付けられている場合、FORCE句によって統計タイプの関連付けが解除され、その統計タイプを使用して収集したユーザー定義の統計情報が削除されます。

関連項目:

- ルーチンの詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。
- 索引の作成および変更の詳細は、「CREATE INDEX」および「ALTER INDEX」を参照してください。
- ドメイン索引の詳細は、「CREATE INDEX」の「domain_index_clause」を参照してください。
- 統計タイプの関連付けの詳細は、「ASSOCIATE STATISTICS」および「DISASSOCIATE STATISTICS」を参照してください。

前提条件

削除する索引が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY INDEXシステム権限が必要です。

構文

drop_index::=

セマンティクス

schema

索引が含まれているスキーマを指定します。schemaを指定しない場合、索引は自分のスキーマ内にあるとみなされます。

index

削除する索引の名前を指定します。索引を削除すると、その索引に割り当てられていたすべてのデータ・ブロックが、その索引が含まれていた表領域に戻されます。

ONLINE

ONLINEを指定すると、索引の削除中に、表またはパーティションに対するDML操作が許可されるようになります。

FORCE

FORCEは、ドメイン索引のみに適用されます。この句を指定すると、索引タイプ・ルーチンの起動がエラーを戻した場合、または索引にIN PROGRESSのマークが付けられている場合でも、ドメイン索引を削除できます。FORCEがないと、その索引タイプ・ルーチンの起動がエラーを戻す場合、または索引にIN PROGRESSマークが付けられている場合にドメイン索引を削除できません。

ノート:

FORCE オプションを指定してドメイン索引を削除すると、索引は、索引タイプ・ルーチンで発生しているエラーに関係なく削除されます。索引タイプ・ルーチンによって発生したエラーは報告されません。

FORCE オプションは、索引または索引パーティションが IN PROGRESS としてマークされている場合、または DROP INDEX がすでに失敗した場合にのみ使用します。

{ DEFERRED | IMMEDIATE } INVALIDATION

この句を使用すると、索引の削除中にデータベースが依存カーソルを無効にする時間を制御できます。セマンティクスはALTER INDEX文と同じですが、次が追加されます。DEFERRED INVALIDATIONで索引を削除すると、Oracleデータベースは、計画内の削除済索引を参照するDML文または問合せをすぐに無効にします。

この句のセマンティクスの詳細は、ALTER INDEXに関する項目の「{ DEFERRED | IMMEDIATE } INVALIDATION」を参照してください。

索引の削除の制限事項

索引の削除には、次の制限事項が適用されます。

- 索引またはその索引の索引パーティションにIN_PROGRESSのマークが付いているときには、ドメイン索引を削除できません。
- ドメイン索引、クラスタ索引、またはキュー表の索引を削除するときには、ONLINE句は指定できません。

例

索引の削除: 例

この文は、索引ord_customer_ix_demo (「索引の圧縮: 例」で作成)を削除します。

```
DROP INDEX ord_customer_ix_demo;
```

DROP INDEXTYPE

目的

DROP INDEXTYPE文を使用すると、索引タイプを削除できます。同時に、統計タイプとの関連付けも解除されます。

関連項目:

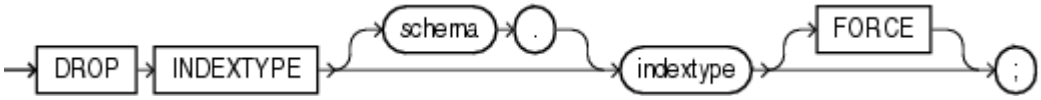
索引タイプの詳細は、「CREATE INDEXTYPE」を参照してください。

前提条件

削除する索引タイプが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY INDEXTYPEシステム権限が必要です。

構文

drop_indextype::=



セマンティクス

schema

索引タイプが含まれているスキーマを指定します。schemaを指定しない場合、この索引タイプは自分のスキーマ内にあるとみなされます。

indextype

削除する索引タイプの名前を指定します。

任意の統計タイプが索引タイプに関連付けられている場合、統計タイプと索引タイプの関連付けが解除され、統計タイプを使用して収集したすべての統計情報が削除されます。

関連項目:

統計情報の関連付けの詳細は、「ASSOCIATE STATISTICS」および「DISASSOCIATE STATISTICS」を参照してください。

FORCE

FORCEを指定すると、1つ以上のドメイン索引から現在参照されている状態でも、その索引タイプを削除できます。これらのドメイン索引には、INVALIDマークが付けられます。FORCEを指定しない場合、任意のドメイン索引で参照されている索引タイプを削除できません。

例

索引タイプの削除: 例

次の文は、索引タイプposition_indextype (「拡張索引作成機能の使用方法」で作成)を削除し、この索引タイプで定義されているすべてのドメイン索引にINVALIDのマークを付けます。

```
DROP INDEXTYPE position_indextype FORCE;
```

DROP INMEMORY JOIN GROUP

目的

DROP INMEMORY JOIN GROUP文を使用すると、データベースから結合グループを削除できます。

関連項目:

- 「CREATE INMEMORY JOIN GROUP」および「ALTER INMEMORY JOIN GROUP」を参照してください。
- 結合グループの詳細は、『Oracle Database In-Memoryガイド』を参照してください。

前提条件

結合グループが他のユーザーのスキーマ内にある場合は、DROP ANY TABLEシステム権限が必要です。

構文

drop_inmemory_join_group::=

セマンティクス

schema

結合グループを含むスキーマを指定します。schemaを指定しない場合、結合グループは自分のスキーマ内にあるとみなされます。

join_group

削除する結合グループの名前を指定します。

DBA_JOINGROUPSまたはUSER_JOINGROUPSデータ・ディクショナリ・ビューを問い合せて、既存の結合グループを表示できます。これらのビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

例

次の文は、結合グループprod_id1を削除します。

```
DROP INMEMORY JOIN GROUP prod_id1;
```

DROP JAVA

目的

DROP JAVA文を使用すると、Javaソース、クラスまたはリソース・スキーマ・オブジェクトを削除できます。

関連項目:

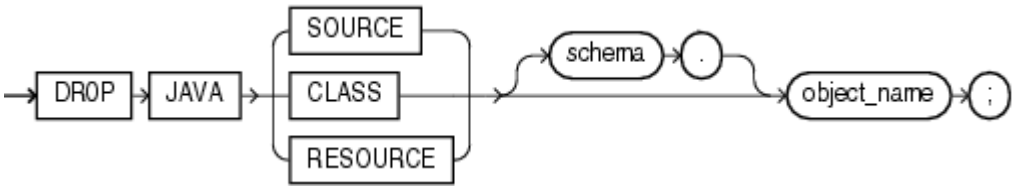
- Javaオブジェクトの作成については、「CREATE JAVA」を参照してください。
- Javaソース、クラスおよびリソースの変換の詳細は、『Oracle Database Java開発者ガイド』を参照してください。

前提条件

削除するJavaソース、クラスまたはリソースが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY PROCEDUREシステム権限が必要です。また、このコマンドを使用する場合は、Javaクラスに対するEXECUTEオブジェクト権限も必要です。

構文

drop_java::=



セマンティクス

JAVA SOURCE

SOURCEを指定すると、Javaソース・スキーマ・オブジェクトおよびそのオブジェクトから導出されたすべてのJavaクラス・スキーマ・オブジェクトを削除できます。

JAVA CLASS

CLASSを指定すると、Javaクラス・スキーマ・オブジェクトを削除できます。

JAVA RESOURCE

RESOURCEを指定すると、Javaリソース・スキーマ・オブジェクトを削除できます。

object_name

既存のJavaクラス、ソースまたはリソース・スキーマ・オブジェクトの名前を指定します。object_nameを二重引用符で囲むと、小文字の名前、または大文字と小文字を組み合せた名前を指定できます。

例

Javaクラス・オブジェクトの削除: 例

次の文は、JavaクラスAgent (「Javaクラス・オブジェクトの作成: 例」で作成)を削除します。

```
DROP JAVA CLASS "Agent";
```

17 SQL文: DROP LIBRARYからDROP SYNONYM

この章では、次のSQL文について説明します。

- DROP LIBRARY
- DROP LOCKDOWN PROFILE
- DROP MATERIALIZED VIEW
- DROPMATERIALIZEDVIEWLOG
- DROP MATERIALIZED ZONEMAP
- DROP OPERATOR
- DROP OUTLINE
- DROP PACKAGE
- DROP PLUGGABLE DATABASE
- DROP PROCEDURE
- DROP PROFILE
- DROP RESTORE POINT
- DROP ROLE
- DROP ROLLBACK SEGMENT
- DROP SEQUENCE
- DROP SYNONYM

DROP LIBRARY

目的

DROP LIBRARY文を使用すると、データベースから外部プロシージャ・ライブラリを削除できます。

関連項目:

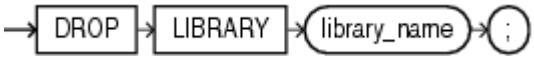
ライブラリの作成については、「CREATE LIBRARY」を参照してください。

前提条件

DROP ANY LIBRARYシステム権限が必要です。

構文

drop_library::=



セマンティクス

library_name

削除する外部プロシージャ・ライブラリの名前を指定します。

例

ライブラリの削除: 例

次の文は、ext_libライブラリを削除します。

```
DROP LIBRARY ext_lib;
```

DROP LOCKDOWN PROFILE

目的

DROP LOCKDOWN PROFILE文を使用すると、データベースからPDBロックダウン・プロファイルを削除できます。PDBに割り当てられているプロファイルが削除された場合、PDBにはそのプロファイルが割り当てられたままになりますが、削除されたプロファイルによる制約は適用されません。

CDB、アプリケーション・ルートまたはPDBのPDB_LOCKDOWN初期化パラメータに、削除されるロックダウン・プロファイルの値が指定されている場合、削除されるプロファイルによる制限は、削除の時点で無効になります。ただし、PDB_LOCKDOWN初期化パラメータの値は、このパラメータが明示的に設定解除されるまで維持されます。

関連項目:

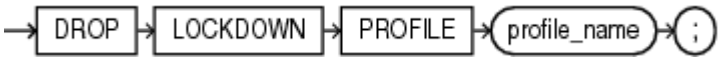
- CREATE LOCKDOWN PROFILEおよびALTER LOCKDOWN PROFILE
- PDBロックダウン・プロファイルの詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

前提条件

- この文は、CDBルートまたはアプリケーション・ルートから発行する必要があります。
- 文を発行する予定のコンテナでのDROP LOCKDOWN PROFILEシステム権限を持っている必要があります。

構文

drop_lockdown_profile::=



セマンティクス

profile_name

削除するPDBロックダウン・プロファイルの名前を指定します。

DBA_LOCKDOWN_PROFILESデータ・ディクショナリ・ビューを問い合せて、既存のPDBロックダウン・プロファイルの名前を検索できます。

関連項目:

DBA_LOCKDOWN_PROFILESデータ・ディクショナリ・ビューとPDB_LOCKDOWN初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

例

次の文は、PDBロックダウン・プロファイルhr_profを削除します。

```
DROP LOCKDOWN PROFILE hr_prof;
```

DROP MATERIALIZED VIEW

目的

DROP MATERIALIZED VIEW文を使用すると、データベースから既存のマテリアライズド・ビューを削除できます。

削除したマテリアライズド・ビューは、ごみ箱内には移動しません。このため、削除したマテリアライズド・ビューを消去またはリカバリすることはできません。

ノート:

下位互換性を保つために、MATERIALIZED VIEW のかわりにキーワード SNAPSHOT もサポートされています。

関連項目:

- 様々なタイプのマテリアライズド・ビューの詳細は、「CREATE MATERIALIZED VIEW」を参照してください。
- マテリアライズド・ビューの変更については、「ALTER MATERIALIZED VIEW」を参照してください。
- レプリケーション環境でのマテリアライズド・ビューの詳細は、『Oracle Database管理者ガイド』を参照してください。
- データ・ウェアハウス環境でのマテリアライズド・ビューの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

前提条件

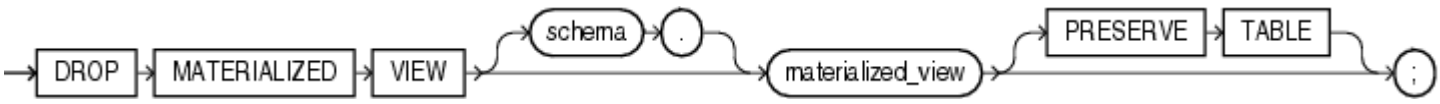
削除するマテリアライズド・ビューが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY MATERIALIZED VIEWシステム権限が必要です。また、データベースがマテリアライズド・ビューのデータを管理するために使用する内部表、ビュー、索引を削除する権限も必要です。

関連項目:

マテリアライズド・ビューを管理するために使用するオブジェクトの削除に必要な権限については、「DROP TABLE」、「DROP VIEW」および「DROP INDEX」を参照してください。

構文

drop_materialized_view::=



セマンティクス

schema

マテリアライズド・ビューが含まれているスキーマを指定します。schemaを指定しない場合、このマテリアライズド・ビューは自分のスキーマ内にあるとみなされます。

materialized_view

削除する既存のマテリアライズド・ビューの名前を指定します。

- マスター表から、最後にリフレッシュされた単純マテリアライズド・ビューを削除した場合、マスター表のマテリアライズド・ビュー・ログのうち、削除されたマテリアライズド・ビューのリフレッシュに必要な行のみが自動的に削除されます。
- 事前作成表で作成されたマテリアライズド・ビューを削除した場合、そのマテリアライズド・ビューが削除され、事前作成表は1つの表としての元の形に戻ります。
- マスター表を削除した場合、その表に基づくマテリアライズド・ビューは自動的には削除されません。ただし、削除したマスター表に基づくマテリアライズド・ビューをリフレッシュしようとした場合、エラー・メッセージが戻されます。
- マテリアライズド・ビューを削除した場合、マテリアライズド・ビューを使用するためにリライトされたコンパイル済の要求が無効になり、自動的に再コンパイルされます。そのマテリアライズド・ビューが表に事前作成されていた場合、その表は削除されませんが、マテリアライズド・ビューのリフレッシュ・メカニズムで管理できなくなります。

PRESERVE TABLE句

この句を使用すると、マテリアライズド・ビュー・オブジェクトを削除した後も、そのマテリアライズド・ビューのコンテナ表およびその内容を保持できます。結果の表の名前は、削除したマテリアライズド・ビューと同じになります。

マテリアライズド・ビューに関連付けられているすべてのメタデータは削除されます。ただし、そのマテリアライズド・ビューの作成時にコンテナ表で自動的に作成された索引は保存されますが、1つの例外があります。ROWIDマテリアライズド・ビューの作成時に作成された索引は削除されます。また、そのマテリアライズド・ビューにネストした表の列が含まれる場合、その列の記憶表がそのメタデータとともに保存されます。

PRESERVE TABLE句の制限事項

この句は、(マテリアライズド・ビューがスナップショットと呼ばれていた) Oracle9iより前のリリースからインポートしたマテリアライズド・ビューには無効です。

例

マテリアライズド・ビューの削除: 例

次の文は、サンプル・スキーマhr内のマテリアライズド・ビューemp_dataを削除します。

```
DROP MATERIALIZED VIEW emp_data;
```

次の文は、sales_by_month_by_stateマテリアライズド・ビューおよびそのマテリアライズド・ビューの基礎となる表を削除します(基礎となる表がON PREBUILT TABLE句が指定されたCREATE MATERIALIZED VIEW文に登録されていない場合)。

```
DROP MATERIALIZED VIEW sales_by_month_by_state;
```

DROPMATERIALIZEDVIEWLOG

目的

DROP MATERIALIZED VIEW LOG文を使用すると、データベースからマテリアライズド・ビュー・ログを削除できます。

> ノート:
>
> 下位互換性を保つために、MATERIALIZED VIEW のかわりにキーワード SNAPSHOT もサポートされています。

関連項目:

- マテリアライズド・ビューの詳細は、「CREATE MATERIALIZED VIEW」および「ALTER MATERIALIZED VIEW」を参照してください。
- マテリアライズド・ビュー・ログの詳細は、「CREATE MATERIALIZED VIEW LOG」を参照してください。
- レプリケーション環境でのマテリアライズド・ビューの詳細は、『Oracle Database管理者ガイド』を参照してください。
- データ・ウェアハウス環境でのマテリアライズド・ビューの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

前提条件

マテリアライズド・ビュー・ログを削除する場合は、表を削除する権限が必要です。

関連項目:

DROP TABLE

構文

drop_materialized_view_log::=

セマンティクス

schema

マテリアライズド・ビュー・ログおよびそのマスター表が含まれているスキーマを指定します。schemaを指定しない場合、マテリアライズド・ビュー・ログおよびマスター表は自分のスキーマ内にあるとみなされます。

table

削除するマテリアライズド・ビュー・ログに関連付けられたマスター表の名前を指定します。

FOR FAST REFRESHのために作成されたマテリアライズド・ビュー・ログを削除した場合、マテリアライズド・ビュー・ログのマスター表に基づく一部のマテリアライズド・ビューが高速リフレッシュできなくなります。高速リフレッシュできなくなるマテリアライズド・ビュー

には、ROWIDマテリアライズド・ビュー、主キー・マテリアライズド・ビューおよび副問合せマテリアライズド・ビューが含まれます。

関連項目:

これらの種類のマテリアライズド・ビューについては、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

FOR SYNCHRONOUS REFRESH (ステージング・ログ)のために作成されたマテリアライズド・ビュー・ログを削除した場合、ステージング・ログのマスター表に基づくマテリアライズド・ビューが同期リフレッシュできなくなります。

例

マテリアライズド・ビュー・ログの削除: 例

次の文は、oe.customersマスター表のマテリアライズド・ビュー・ログを削除します。

```
DROP MATERIALIZED VIEW LOG ON customers;
```

DROP MATERIALIZED ZONEMAP

目的

DROP MATERIALIZED ZONEMAP文を使用すると、データベースから既存のゾーン・マップを削除します。

前提条件

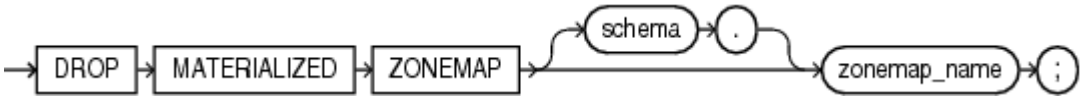
ゾーン・マップが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY MATERIALIZED VIEWシステム権限が必要です。また、データベースがゾーン・マップのデータを管理するために使用する内部表および索引を削除する権限も必要です。

関連項目:

ゾーン・マップを管理するために使用するオブジェクトの削除に必要な権限については、「DROP TABLE」および「DROP INDEX」を参照してください。

構文

drop_materialized_zonemap::=



セマンティクス

schema

ゾーン・マップが含まれているスキーマを指定します。schemaを指定しない場合、このゾーン・マップは自分のスキーマ内にあるとみなされます。

zonemap_name

削除する既存のゾーン・マップの名前を指定します。

例

ゾーン・マップの削除: 例

次の文は、ゾーン・マップsales_zmapを削除します。

```
DROP MATERIALIZED ZONEMAP sales_zmap;
```

DROP OPERATOR

目的

DROP OPERATOR文を使用すると、ユーザー定義演算子を削除できます。

関連項目:

- 演算子の作成および変更の詳細は、「CREATE OPERATOR」および「ALTER OPERATOR」を参照してください。
- 演算子の概要については、「ユーザー定義演算子」および『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。
- ユーザー定義索引タイプの演算子の削除については、「ALTER INDEXTYPE」を参照してください。

前提条件

削除する演算子が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY OPERATORシステム権限が必要です。

構文

drop_operator::=

セマンティクス

schema

演算子が含まれているスキーマを指定します。schemaを指定しない場合、その演算子は自分のスキーマ内にあるとみなされます。

operator

削除する演算子の名前を指定します。

FORCE

FORCEを指定すると、演算子が現在1つ以上のスキーマ・オブジェクト(索引タイプ、パッケージ、ファンクション、プロシージャなど)によって参照されている場合でも、その演算子を削除できます。演算子に依存するこのようなオブジェクトには、INVALIDのマークが付けられます。FORCEを使用しないと、任意のスキーマ・オブジェクトに参照されている演算子を削除できません。

例

ユーザー定義演算子の削除: 例

次の文は、演算子eq_opを削除します。

```
DROP OPERATOR eq_op;
```

FORCE句が指定されていないため、この演算子のバインディングのいずれかが索引タイプによって参照されている場合、この操作は実行されません。

DROP OUTLINE

目的

ノート:

- ストアド・アウトラインは非推奨になりました。ストアド・アウトラインは、下位互換性を保つために今でもサポートされています。ただし、かわりに SQL 計画管理を使用することをお薦めします。SQL 計画管理では、ストアド・アウトラインよりも非常に安定した SQL パフォーマンスを実現する SQL 計画ベースラインが作成されます。

- DBMS_SPM パッケージの MIGRATE_STORED_OUTLINE ファンクションまたは Enterprise Manager Cloud Control を使用して、既存のストアド・アウトラインを SQL 計画ベースラインに移行できます。移行が完了したら、ストアド・アウトラインに移行済のマークが付けられ、削除できるようになります。DBMS_SPM パッケージの DROP_MIGRATED_STORED_OUTLINE ファンクションを使用して、システム上のすべての移行済ストアド・アウトラインを削除できます。

- 関連項目: SQL 計画管理の詳細は、『Oracle Database SQL チューニング・ガイド』を参照してください。DBMS_SPM パッケージの詳細は、『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』を参照してください。

DROP OUTLINE文を使用すると、ストアド・アウトラインを削除できます。

関連項目:

アウトラインの作成については、「CREATE OUTLINE」を参照してください。

前提条件

アウトラインを削除する場合は、DROP ANY OUTLINEシステム権限が必要です。

構文

drop_outline::=

セマンティクス

outline

削除するアウトラインの名前を指定します。

そのアウトラインが削除された後、ストアド・アウトラインが作成されたSQL文がコンパイルされると、オプティマイザはアウトラインの影響なしに新しい実行計画を生成します。

例

アウトラインの削除: 例

次の文は、salariesというストアド・アウトラインを削除します。

```
DROP OUTLINE salaries;
```

DROP PACKAGE

目的

パッケージはPL/SQLを使用して定義されます。パッケージの作成、変更および削除の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

DROP PACKAGE文を使用すると、データベースからストアド・パッケージを削除できます。この文は、パッケージの本体および仕様部を削除します。

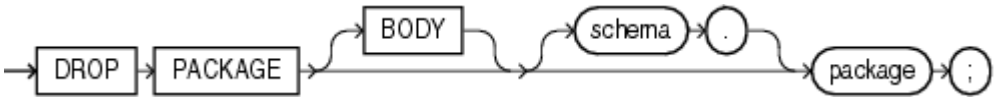
> ノート:
>
> この文を使用してパッケージから単一のオブジェクトを削除しないでください。かわりに、CREATE PACKAGE 文および CREATE PACKAGE BODY 文に OR REPLACE 句を指定して、そのオブジェクトを含めずにパッケージを再作成してください。

前提条件

削除するパッケージが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY PROCEDUREシステム権限が必要です。

構文

drop_package::=



セマンティクス

BODY

BODYを指定すると、パッケージ本体のみを削除できます。この句を省略した場合、パッケージ本体と仕様部の両方が削除されます。

パッケージ本体のみを削除して仕様部は削除しない場合、依存するオブジェクトは無効になりません。ただし、パッケージ本体を再作成しないかぎり、パッケージ仕様部で宣言したプロシージャやストアド・ファンクションはコールできません。

schema

パッケージが含まれているスキーマを指定します。schemaを指定しない場合、パッケージは自分のスキーマ内にあるとみなされます。

package

削除するパッケージの名前を指定します。

そのパッケージ仕様部に依存するすべてのローカル・オブジェクトが無効になります。後でこれらのオブジェクトのいずれかを参照した場合、データベースは、そのオブジェクトを再コンパイルしようとします。削除したパッケージを再作成しないかぎり、エラーが戻されます。

任意の統計タイプがパッケージに関連付けられている場合、FORCE句によって統計タイプの関連付けが解除され、統計タイプ

を使用して収集されたユーザー定義のすべての統計情報が削除されます。

関連項目:

「ASSOCIATE STATISTICS」および「DISASSOCIATE STATISTICS」を参照してください。

例

パッケージの削除: 例

次の文は、emp_mgmtパッケージの仕様部および本体を削除し、その仕様部に依存するすべてのオブジェクトを無効にします。このパッケージを作成する例については、『Oracle Database PL/SQL言語リファレンス』を参照してください。

```
DROP PACKAGE emp_mgmt;
```

DROP PLUGGABLE DATABASE

目的

DROP PLUGGABLE DATABASE文を使用すると、プラガブル・データベース(PDB)を削除できます。PDBには、従来のPDB、アプリケーション・コンテナ、アプリケーション・シードまたはアプリケーションPDBを指定できます。

PDBを削除すると、削除したPDBおよびそのデータファイルの参照をすべて削除するように、マルチテナント・コンテナ・データベース(CDB)の制御ファイルが変更されます。削除したPDBに関連付けられたアーカイブ・ログとバックアップは、削除されません。これらを削除するにはOracle Recovery Manager (RMAN)を使用することができ、これらを残してPDBのPoint-in-Timeリカバリを続けて実行することもできます。

注意:

DROP PLUGGABLE DATABASE 文はロールバックできません。

前提条件

CDBに接続している必要があります。

従来のPDBまたはアプリケーション・コンテナを削除するには、現在のコンテナがルートである必要があります。また、AS SYSDBAまたはAS SYSOPERとして認証される必要があるとともに、SYSDBAまたはSYSOPER権限(共通に付与されている権限か、ルートおよび削除対象の従来のPDBまたはアプリケーション・コンテナのそれぞれでローカルに付与されている権限のいずれか)が必要です。アプリケーション・コンテナは空である必要があります(アプリケーション・シードやアプリケーションPDBが含まれていない)。

アプリケーション・シードを削除するには、現在のコンテナがルートまたはアプリケーション・ルートである必要があります。また、AS SYSDBAまたはAS SYSOPERとして認証される必要があるとともに、SYSDBAまたはSYSOPER権限(共通に付与されている権限か、ルートまたはアプリケーション・ルートでローカルに付与されている権限のいずれか)が必要です。

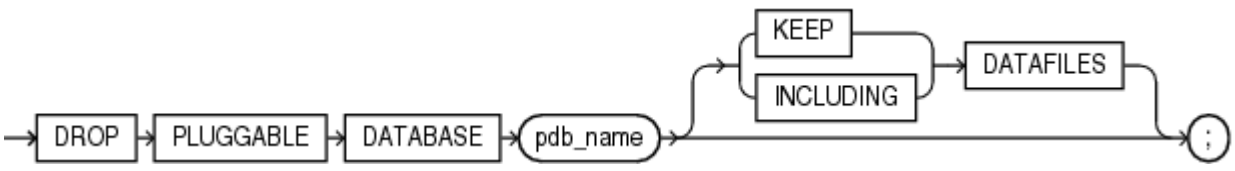
アプリケーションPDBを削除するには、現在のコンテナがルートまたはアプリケーション・ルートである必要があります。また、AS SYSDBAまたはAS SYSOPERとして認証される必要があるとともに、SYSDBAまたはSYSOPER権限(共通に付与されている権限か、ルートまたはアプリケーション・ルートおよび削除対象のアプリケーションPDBのそれぞれでローカルに付与されている権限のいずれか)が必要です。

KEEP DATAFILES(デフォルト)を指定するには、削除する対象のPDBを切断する必要があります。

INCLUDING DATAFILESを指定するには、削除する対象のPDBはマウント・モードである、または切断されている必要があります。

構文

drop_pluggable_database::=



セマンティクス

pdb_name

削除するPDBの名前を指定します。シード(PDB$SEED)は削除できません。しかし、アプリケーション・シードは削除できます。

KEEP DATAFILES

KEEP DATAFILESを指定すると、PDBの削除後に、そのPDBに関連付けられたデータファイルを保持できます。PDBの一時ファイルは、不要になるため削除されます。これはデフォルトです。

2つのCDBが同じストレージ・デバイスを共有しており、一方のCDBからPDBを削除し、もう一方のCDBに接続するシナリオでは、データファイルを保持しておくと便利なことがあります。

INCLUDING DATAFILES

INCLUDING DATAFILESを指定すると、削除するPDBに関連付けられたデータファイルを削除できます。PDBの一時ファイルも削除されます。

SNAPSHOT COPY PDBの削除の制限事項

PDBがSNAPSHOT COPY句を使用して作成された場合、PDBを削除する際にINCLUDING DATAFILESを指定する必要があります。

例

PDBの削除: 例

次の文は、PDB pdb1とそれに対応付けられているデータファイルを削除します。

```
DROP PLUGGABLE DATABASE pdb1
  INCLUDING DATAFILES;
```

DROP PROCEDURE

目的

プロシージャはPL/SQLを使用して定義されます。プロシージャの作成、変更および削除の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

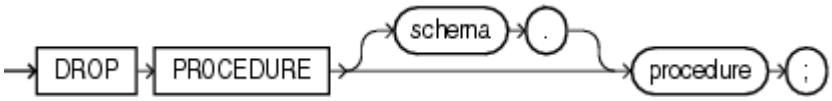
DROP PROCEDURE文を使用すると、データベースからスタンドアロンのストアド・プロシージャを削除できます。この文を使用してパッケージの一部であるプロシージャを削除しないでください。かわりに、DROP PACKAGE文を使用してパッケージ全体を削除するか、OR REPLACE句を指定したCREATE PACKAGE文を使用して、そのプロシージャを含めずにパッケージを再定義してください。

前提条件

削除するプロシージャが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY PROCEDUREシステム権限が必要です。

構文

drop_procedure::=



セマンティクス

schema

プロシージャが含まれているスキーマを指定します。schemaを指定しない場合、プロシージャは自分のスキーマ内にあるとみなされます。

プロシージャ

削除するプロシージャの名前を指定します。

プロシージャを削除すると、そのプロシージャに依存するすべてのローカル・オブジェクトが無効になります。後でこれらのオブジェクトのいずれかを参照した場合、データベースは、そのオブジェクトを再コンパイルしようとします。削除したプロシージャを再作成しないかぎり、エラー・メッセージが戻されます。

例

プロシージャの削除: 例

次の文は、ユーザーhrが所有するプロシージャremove_empを削除し、remove_empに依存するすべてのオブジェクトを無効にします。

```
DROP PROCEDURE hr.remove_emp;
```

DROP PROFILE

目的

DROP PROFILE文を使用すると、データベースからプロファイルを削除できます。DEFAULTプロファイル以外のプロファイルを削除できます。

関連項目:

プロファイルの作成および変更の詳細は、「CREATE PROFILE」および「ALTER PROFILE」を参照してください。

前提条件

DROP PROFILEシステム権限が必要です。

構文

drop_profile::=

セマンティクス

profile

削除するプロファイルの名前を指定します。

CASCADE

CASCADEを指定すると、プロファイルの割当て先のユーザーからプロファイルの割当てを解除できます。このようなユーザーには、自動的にDEFAULTプロファイルが割り当てられます。現在、複数のユーザーに割り当てられているプロファイルを削除する場合は、必ずこの句を指定します。

例

プロファイルの削除: 例

次の文は、プロファイルapp_user (「プロファイルの作成: 例」で作成)を削除します。プロファイルapp_userは削除され、現在app_userプロファイルが割り当てられているユーザーにはDEFAULTプロファイルが割り当てられます。

```
DROP PROFILE app_user CASCADE;
```

DROP RESTORE POINT

目的

DROP RESTORE POINT文を使用すると、通常のリストア・ポイントまたは保証付きリストア・ポイントをデータベースから削除できます。

- 通常のリストア・ポイントは、ユーザーが削除する必要はありません。必要に応じて古いものから順に自動的に削除されます(「restore_point」の説明を参照)。ただし、同じ名前を再利用する必要がある場合は、通常のリストア・ポイントをユーザーが削除することもできます。
- 保証付きリストア・ポイントは、自動的に削除されません。したがって、保証付きリストア・ポイントをデータベースから削除するには、明示的にこの文を使用する必要があります。

関連項目:

リストア・ポイントの作成方法および使用方法については、「CREATE RESTORE POINT」、「FLASHBACK DATABASE」および「FLASHBACK TABLE」を参照してください。

前提条件

通常のリストア・ポイントを削除するには、SELECT ANY DICTIONARY、FLASHBACK ANY TABLE、SYSBACKUP、またはSYSDGシステム権限が必要です。

保証付きリストア・ポイントを削除するには、次の条件のうち1つを満たしている必要があります。

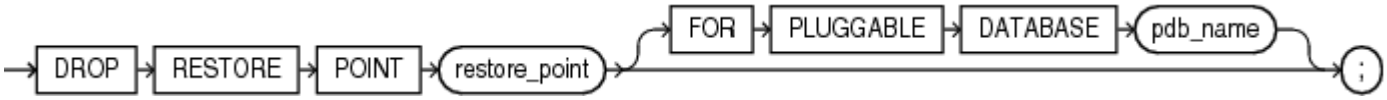
- AS SYSDBA、AS SYSBACKUPまたはAS SYSDGを接続する必要があります。
- SYSDBA権限が付与されており、マルチテナント・データベースを使用している必要があります。
- ユーザーSYSとして実行しており、マルチテナント・データベースを使用している必要があります。

マルチテナント・コンテナ・データベース(CDB)に接続しているときに、次のようにリストア・ポイントを削除できます。

- 通常のCDBリストア・ポイントを削除するには、現在のコンテナがルートである必要があります。また、SELECT ANY DICTIONARYまたはFLASHBACK ANY TABLEシステム権限(共通に付与されている権限か、ルートでローカルに付与されている権限のいずれか)または共通に付与されているSYSDBA、SYSBACKUPまたはSYSDGシステム権限が必要です。
- 保証されたCDBリストア・ポイントを削除する場合は、現在のコンテナがルートである必要があります。また、共通に付与されているSYSDBA、SYSBACKUPまたはSYSDGシステム権限が必要です。
- 通常のPDBリストア・ポイントを削除するには、現在のコンテナがルートである必要があり、共通に付与されているSELECT ANY DICTIONARY、FLASHBACK ANY TABLE、SYSDBA、SYSBACKUPまたはSYSDGシステム権限があるか、または現在のコンテナがリストア・ポイントを作成するPDBである必要があり、SELECT ANY DICTIONARY、FLASHBACK ANY TABLE、SYSDBA、SYSBACKUPまたはSYSDGシステム権限(共通に付与されている権限か、そのPDBでローカルに付与されている権限)が必要です。
- 保証されたPDBリストア・ポイントを削除するには、現在のコンテナがルートである必要があり、共通に付与されているSYSDBA、SYSBACKUPまたはSYSDGシステム権限が必要であるか、または現在のコンテナがリストア・ポイントを作成するPDBである必要があり、SYSDBA、SYSBACKUPまたはSYSDGシステム権限(共通に付与されている権限か、そのPDBでローカルに付与されている権限)が必要です。

構文

drop_restore_point::=



セマンティクス

restore_point

削除するリストア・ポイントの名前を指定します。

FOR PLUGGABLE DATABASE

この句を使用すると、ルートに接続されている場合にPDBリストア・ポイントを削除できます。pdb_nameには、削除するリストア・ポイントが含まれているPDBの名前を指定します。

リストア・ポイントを削除するPDBに接続されている場合、この句を指定する必要はありません。ただし、この句を指定する場合は、接続しているPDBの名前を指定する必要があります。

例

リストア・ポイントの削除: 例

次の例は、good_dataリストア・ポイント(「リストア・ポイントの作成および使用方法: 例」で作成)を削除します。

```
DROP RESTORE POINT good_data;
```

DROP ROLE

目的

DROP ROLE文を使用すると、データベースからロールを削除できます。ロールを削除した場合、そのロールが付与されていたすべてのユーザーおよびロールからそのロールが取り消され、データベースからも削除されます。すでに使用可能なロールのユーザー・セッションには影響しません。ただし、削除後は新しいユーザー・セッションでロールを使用可能にできません。

関連項目:

- ロールの作成、およびロールを使用可能にするために必要な認可の変更については、「CREATE ROLE」および「ALTER ROLE」を参照してください。
- 現行のセッションに対するロールの使用禁止化については、「SET ROLE」を参照してください。

前提条件

ADMIN OPTION付きのロールが付与されているか、またはDROP ANY ROLEシステム権限を持っている必要があります。

構文

drop_role::=

セマンティクス

role

削除するロールの名前を指定します。

例

ロールの削除: 例

次の文は、ロールdw_manager (「ロールの作成: 例」で作成)を削除します。

```
DROP ROLE dw_manager;
```

DROP ROLLBACK SEGMENT

目的

DROP ROLLBACK SEGMENTを使用すると、データベースからロールバック・セグメントを削除できます。ロールバック・セグメントを削除した場合、ロールバック・セグメントに割り当てられていたすべての領域が表領域に戻されます。

ノート:

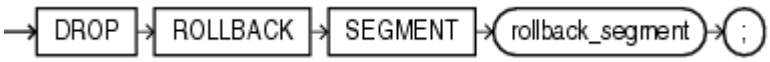
データベースが自動 UNDO モードで実行されている場合、ロールバック・セグメントに有効な操作はこの文のみです。このモードの場合、ロールバック・セグメントを作成または変更できません。

前提条件

DROP ROLLBACK SEGMENTシステム権限が必要です。また、ロールバック・セグメントをオフラインにしておく必要があります。

構文

drop_rollback_segment::=



セマンティクス

rollback_segment

削除するロールバック・セグメントの名前を指定します。

ロールバック・セグメントの削除の制限事項

この文には、次の制限事項があります。

- ロールバック・セグメントは、オフラインになっている場合にのみ削除できます。ロールバック・セグメントがオフラインであるかどうかを判断するには、DBA_ROLLBACK_SEGSデータ・ディクショナリ・ビューを問い合せます。オフラインのロールバック・セグメントは、STATUS列の値がAVAILABLEになっています。また、ALTER ROLLBACK SEGMENT文にOFFLINE句を指定して、ロールバック・セグメントをオフラインにすることもできます。
- SYSTEMロールバック・セグメントは削除できません。

例

ロールバック・セグメントの削除: 例

次の構文は、ロールバック・セグメント(「ロールバック・セグメントの作成: 例」で作成)を削除します。

```
DROP ROLLBACK SEGMENT rbs_one;
```

DROP SEQUENCE

目的

DROP SEQUENCE文を使用すると、データベースから順序を削除できます。

この文を使用すると、順序の削除および再作成を行って、順序を再開することもできます。たとえば、現在、値が150の順序を初期値27で再開する場合、順序を削除して同じ名前で再作成し、START WITH値を27にします。

関連項目:

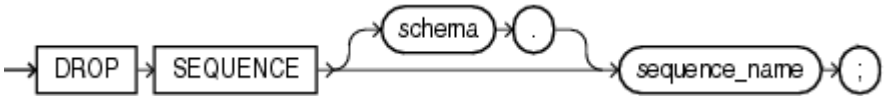
順序の作成および変更の詳細は、「CREATE SEQUENCE」および「ALTER SEQUENCE」を参照してください。

前提条件

削除する順序が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY SEQUENCEシステム権限が必要です。

構文

drop_sequence::=



セマンティクス

schema

順序が含まれているスキーマを指定します。schemaを指定しない場合、順序は自分のスキーマ内にあるとみなされます。

sequence_name

削除する順序の名前を指定します。

例

順序の削除: 例

次の文は、ユーザーoeが所有する順序customers_seq (「順序の作成: 例」で作成)を削除します。この文を発行する場合は、ユーザーoeとして接続するか、またはDROP ANY SEQUENCEシステム権限が必要です。

```
DROP SEQUENCE oe.customers_seq;
```

DROP SYNONYM

目的

DROP SYNONYM文を使用すると、データベースからシノニムを削除できます。また、シノニムの削除および再作成を行って、シノニムの定義を変更することもできます。

関連項目:

シノニムの詳細は、「CREATE SYNONYM」を参照してください。

前提条件

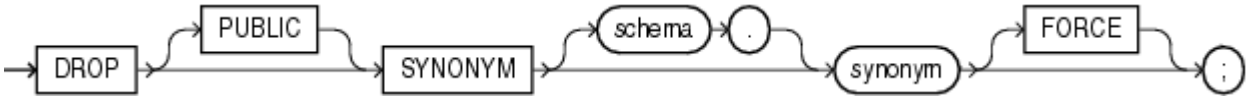
プライベート・シノニムを削除する場合は、シノニムが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY SYNONYMシステム権限が必要です。

PUBLICシノニムを削除する場合は、DROP PUBLIC SYNONYMシステム権限が必要です。

構文

drop_synonym::=

drop_synonym::=



セマンティクス

PUBLIC

PUBLICを指定すると、パブリック・シノニムを削除できます。PUBLICを指定した場合、schemaは指定できません。

schema

シノニムが含まれているスキーマを指定します。schemaを指定しない場合、シノニムは自分のスキーマ内にあるとみなされます。

synonym

削除するシノニムの名前を指定します。

マテリアライズド・ビューを定義する問合せに実際の表名ではなくシノニムが指定されていた場合に、このマテリアライズド・ビューのマスター表のシノニムを削除すると、このマテリアライズド・ビューには使用禁止のマークが付けられます。

依存表またはユーザー定義型を持つオブジェクト型のシノニムを削除するには、FORCEも指定する必要があります。

FORCE

FORCEを指定すると、依存表またはユーザー定義型を持つシノニムでも強制的に削除できます。

ノート:

FORCE を使用して、依存性のあるオブジェクト型のシノニムを削除することはお薦めしません。この操作によって、他のユーザー定義型が無効になるか、またはこのシノニムに依存する表列に UNUSED のマークが付く場合があります。型の依存性の詳細は、『Oracle Database オブジェクト・リレーショナル開発者ガイド』を参照してください。

例

シノニムの削除: 例

次の文は、パブリック・シノニムcustomers (「Oracle Databaseによるシノニムの変換: 例」で作成)を削除します。

```
DROP PUBLIC SYNONYM customers;
```

18 SQL文: DROP TABLEからLOCK TABLE

この章では、次のSQL文について説明します。

- DROP TABLE
- DROP TABLESPACE
- DROP TABLESPACE SET
- DROP TRIGGER
- DROP TYPE
- DROP TYPE BODY
- DROP USER
- DROP VIEW
- EXPLAIN PLAN
- FLASHBACK DATABASE
- FLASHBACK TABLE
- GRANT
- INSERT
- LOCK TABLE

DROP TABLE

目的

DROP TABLE文を使用すると、表またはオブジェクト表をごみ箱に移動したり、表およびそれに含まれるすべてのデータをデータベースから完全に削除することができます。

ノート:

PURGE 句を指定しないかぎり、DROP TABLE を実行しても、他のオブジェクトが使用できるように領域が解放されて表領域に戻されることはなく、その領域はユーザーの領域割当てとして計算されたままになります。

外部表の場合、この文はデータベースにある表のメタデータのみを削除します。データベースの外に常駐する実際のデータには影響を与えません。

クラスタの一部である表を削除すると、その表はごみ箱に移動します。ただし、その後にそのクラスタを削除すると、その表はごみ箱から消去され、FLASHBACK TABLE操作でもリカバリできなくなります。

表を削除すると、依存オブジェクトが無効になり、表のオブジェクト権限が取り消されます。表を再作成する際、表のオブジェクト権限を再度付与し、表の索引、整合性制約およびトリガーを再作成し、記憶域パラメータを再指定する必要があります。切捨ての場合は、このような影響はありません。そのため、TRUNCATE文による行の削除は、表を削除して再作成するよりも効率的です。

関連項目:

- 表の作成および変更の詳細は、「CREATE TABLE」および「ALTER TABLE」を参照してください。
- 表からデータを削除する方法については、「TRUNCATE TABLE」および「DELETE」を参照してください。
- 削除した表をごみ箱から取り出す方法については、「FLASHBACK TABLE」を参照してください。

前提条件

削除する表が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY TABLEシステム権限が必要です。

セッションがバインドされていない場合にのみ、一時表でDDL操作(ALTER TABLE、DROP TABLE、CREATE INDEXなど)を実行できます。セッションを一時表にバインドするには、一時表でINSERT操作を実行します。セッションを一時表からアンバインドするには、TRUNCATE文を発行するか、セッションを終了します。また、トランザクション固有の一時表からアンバインドするには、COMMITまたはROLLBACK文を発行します。

プライベート一時表の削除

既存のDROP TABLEコマンドを使用して、プライベート一時表を削除できます。プライベート一時表を削除しても、既存のトランザクションはコミットされません。これは、トランザクション固有とセッション固有の両方のプライベート一時表に適用されます。削除されたプライベート一時表はRECYCLEBINに移動されないことに注意してください。

不変表の削除

不変表を削除するには、DROP TABLE文を使用します。不変表を削除する際は、PURGEオプションを含めることをお薦めします。不変表を削除すると、データ・ディクショナリから定義が削除され、そのすべての行が削除され、表に定義されている索引およ

びトリガーも削除されます。

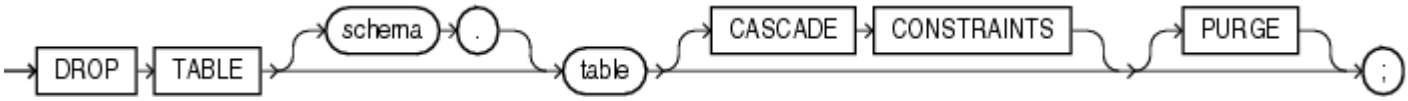
不変表が自分のスキーマに含まれているか、DROP ANY TABLEシステム権限を持っている必要があります。

不変表は、その表の保存期間で定義されている期間が経過するまで変更されていなかった場合にのみ削除できます。

空の不変表は、保存期間に関係なく削除できます。

構文

drop_table::=



セマンティクス

schema

表が含まれているスキーマを指定します。schemaを指定しない場合、表は自分のスキーマ内にあるとみなされます。

table

削除する表の名前を指定します。Oracle Databaseによって次の操作が自動的に実行されます。

- 表のすべての行が削除されます。
- 表のすべての索引およびドメイン索引のみでなく、その表に定義されているすべてのトリガーが、作成者やスキーマの所有者とは関係なく削除されます。tableがパーティション化されている場合、対応するローカル索引パーティションも同様に削除されます。
- tableのネストした表およびLOBのすべての記憶表が削除されます。
- レンジ・パーティション表、ハッシュ・パーティション表またはリスト・パーティション表を削除すると、すべての表パーティションも削除されます。コンポジット・パーティション表を削除した場合、すべてのパーティションおよびサブパーティションが同様に削除されます。
- PURGEキーワードでパーティション表を削除する場合、その文は一連のサブトランザクションとして実行され、それぞれのサブトランザクションで、パーティションまたはサブパーティションのサブセットおよびそのメタデータが削除されます。削除操作がこのように複数のサブトランザクションに分割されていることで、特に大きなパーティション表を削除する場合に、内部システム・リソース(ライブラリ・キャッシュなど)の消費処理が最適化されます。最初のサブトランザクションがコミットされるとすぐにUNUSABLEのマークが表に付けられます。いずれかのサブトランザクションが失敗した場合、その表に対して可能な操作は、もう一度DROP TABLE ... PURGE文を実行することのみです。この場合、前回のDROP TABLE文が失敗したところから、処理が継続されます。ただし、この処理は、前回の操作で発生したエラーが修正されていることを前提としています。

  このような削除操作でUNUSABLEのマークが付けられた表のリストは、該当する*_TABLES、*_PART_TABLES、*_ALL_TABLESまたは*_OBJECT_TABLESデータ・ディクショナリ・ビューのstatus列を問い合せて表示できます。

関連項目:

パーティション表の削除の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

- 索引構成表の場合、索引構成表に定義したマッピング表も削除されます。
- ドメイン索引の場合、適切な削除ルーチンが起動されます。これらのルーチンの詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。
- 任意の統計タイプが表に関連付けられている場合、FORCE句によって統計タイプの関連付けが解除され、統計タイプを使用して収集されたユーザー定義のすべての統計情報が削除されます。

  関連項目:

  統計タイプの関連付けの詳細は、「ASSOCIATE STATISTICS」および「DISASSOCIATE STATISTICS」を参照してください。

- 表がクラスタの一部でない場合、表とその索引に割り当てられていたすべてのデータ・ブロックが、表とその索引が格納されていた表領域に戻されます。

  DROP CLUSTER文にINCLUDING TABLES句を指定した場合、クラスタおよびそのすべての表を削除できます。これによって、表を1つずつ削除する必要がなくなります。「DROP CLUSTER」を参照してください。

- 表がビュー、マテリアライズド・ビューのコンテナ表またはマスター表の実表である場合、あるいは表がストアド・プロシージャ、ファンクションまたはパッケージで参照されている場合、依存するオブジェクトは無効になりますが、削除はされません。表を再作成するか、これらのオブジェクトを一度削除してその表に依存しない形で再作成しないかぎり、これらのオブジェクトは使用できません。

  表を再作成する場合、ストアド・プロシージャ、ファンクションまたはパッケージで参照する列、およびマテリアライズド・ビューの定義で使用されていた副問合せで選択されるすべての列が、その表に含まれている必要があります。ビュー、ストアド・プロシージャ、ファンクション、パッケージに対するオブジェクト権限が付与されていたユーザーに、これらの権限を再付与する必要はありません。

  表がマテリアライズド・ビューのマスター表の場合、マテリアライズド・ビューの問合せはできます。ただし、そのマテリアライズド・ビューを定義する問合せによって選択されるすべての列を含む表を再作成しないかぎり、リフレッシュはできません。

  表がマテリアライズド・ビュー・ログを含む場合、このログおよびその表に関連付けられているダイレクト・パス・インサートのその他のリフレッシュ情報は削除されます。

表の削除の制限事項:

- ネストした表の記憶表は直接削除できません。かわりに、ALTER TABLE ... DROP COLUMN句を使用して、ネストした表の列を削除する必要があります。
- 参照パーティション表の親表は削除できません。まず、参照パーティション表のすべての子表を削除する必要があります。
- 履歴追跡にフラッシュバック・データ・アーカイブを使用する表は削除できません。まず、表でのフラッシュバック・アーカイブの使用を無効にする必要があります。

CASCADE CONSTRAINTS

CASCADE CONSTRAINTSを指定すると、削除される表の主キーまたは一意キーを参照するすべての参照整合性制約を削除できます。このような参照整合性制約があるときにこの句を省略した場合、エラーが戻され、表は削除されません。

PURGE

PURGEを指定すると、1つのステップで、表を削除してその表に関連付けられた領域を解放できます。PURGEを指定すると、表

およびその依存オブジェクトはごみ箱に移動しません。

ノート:

PURGE 句を指定した DROP TABLE 文はロールバックできません。また、PURGE 句で削除した表はリカバリできません。

この句を使用することは、表を削除してからその表をごみ箱から消去することと同じです。この句を使用すると、それらの操作を1つのステップで実行できます。また、機密情報をごみ箱に表示しないようにできるため、セキュリティを強化できます。

関連項目:

ごみ箱の詳細、およびごみ箱内のオブジェクトのネーミング規則の詳細は、『Oracle Database管理者ガイド』を参照してください。

例

表の削除: 例

次の文は、oe.list_customers表(「リスト・パーティション化の例」で作成)を削除します。

```
DROP TABLE list_customers PURGE;
```

DROP TABLESPACE

目的

DROP TABLESPACE文を使用すると、データベースから表領域を削除できます。

削除した表領域は、ごみ箱内には移動しません。このため、削除した表領域を消去またはリカバリすることはできません。

関連項目:

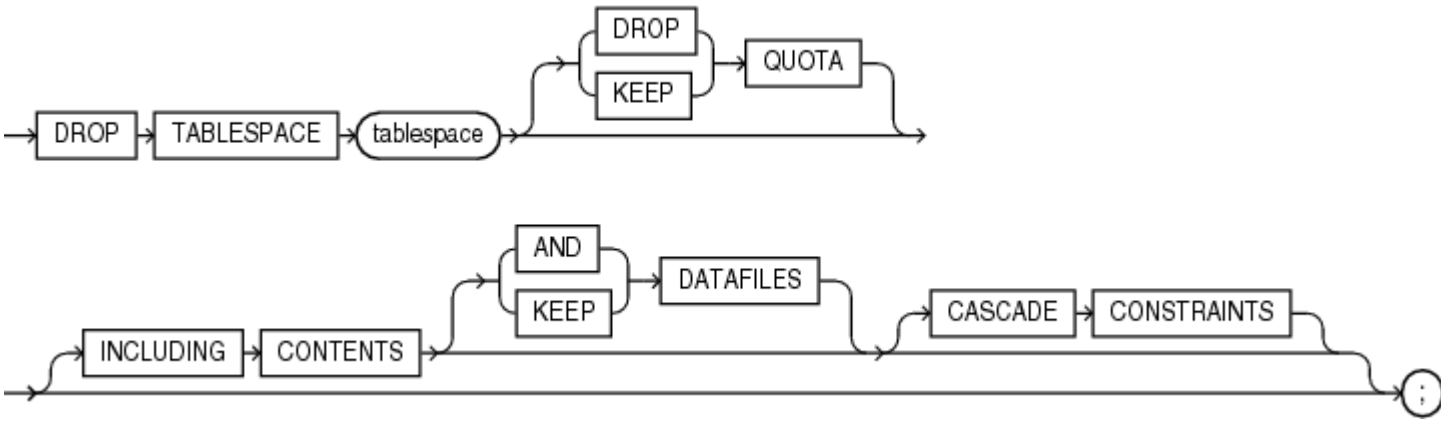
表領域の作成および変更の詳細は、「CREATE TABLESPACE」および「ALTER TABLESPACE」を参照してください。

前提条件

DROP TABLESPACEシステム権限が必要です。アクティブ・トランザクションを保持するロールバック・セグメントを含む場合は、表領域を削除できません。

構文

drop_tablespace::=



セマンティクス

tablespace

削除する表領域(消失書込み保護更新が格納されているシャドウ表領域を含む)の名前を指定します。

表領域の状態がオンラインまたはオフラインのどちらであっても、その表領域を削除できます。実行中のトランザクション内のSQL文で、表領域内のいずれかのオブジェクトにアクセスすることがないように、表領域はオフラインにしてから削除することをお薦めします。

SYSTEM表領域は削除できません。SYSAUX表領域は、SYSDBAシステム権限を持ち、UPGRADEモードでデータベースを起動した場合にのみ削除できます。

削除する表領域がデフォルト表領域または一時表領域として割り当てられていたユーザーにアラートを出す必要がある場合があります。表領域が削除された後では、このようなユーザーはオブジェクトに領域を割り当てたり、表領域内で領域をソートすることはできません。ALTER USER文を使用すると、ユーザーに新しいデフォルト表領域および一時表領域を割り当てることができます。

以前に表領域から削除し、ごみ箱に移動したオブジェクトがごみ箱から消去されます。表領域に関連するすべてのメタデータ、および表領域に含まれるすべてのデータファイルと一時ファイルが、データ・ディクショナリから削除されます。また、表領域にある

Oracle Managed Filesのデータファイルおよび一時ファイルが、オペレーティング・システムから自動的に削除されます。その他のデータファイルおよび一時ファイルは、INCLUDING CONTENTS AND DATAFILESを指定しないかぎり、オペレーティング・システムから削除されません。

この文を使用して表領域グループを削除することはできません。ただし、tablespaceが表領域グループ内で唯一の表領域である場合、その表領域グループもデータ・ディクショナリから削除されます。

表領域の削除の制限事項

表領域の削除には、次の制限事項があります。

- ドメイン索引またはドメイン索引によって作成されたオブジェクトを格納している表領域は削除できません。
- いずれかのインスタンスによって使用されている場合、またはコミットされていないトランザクションのロールバックに必要なUNDOデータを含む場合は、UNDO表領域を削除できません。
- データベースのデフォルト表領域に指定されている表領域は削除できません。この表領域を削除するには、まず他の表領域をデフォルト表領域として再割当てする必要があります。
- データベースのデフォルトの一時表領域グループに属する一時表領域は削除できません。この表領域を削除するには、まずその表領域をデータベースのデフォルトの一時表領域グループから削除する必要があります。
- 既存のセッションで使用されているセグメントが含まれる場合、一時表領域を削除できません。この場合は、エラーは発生しません。データベースは、既存のセッションで使用されているセグメントがなくなるまで待機し、表領域を削除します。
- ある表領域を削除すると他の表領域の主キー制約または一意制約が無効になる場合、INCLUDING CONTENTSおよびCASCADE CONSTRAINTS句を指定してもその表領域を削除できません。たとえば、削除する表領域に主キー列索引が含まれており、主キー列自体が別の表領域に存在する場合、その別の表領域内の主キー制約を手動で無効にしないかぎり、表領域を削除できません。

関連項目:

ドメイン索引の詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』および『Oracle Database概要』を参照してください。

{ DROP | KEEP } QUOTA

DROP QUOTAを指定すると、表領域のすべてのユーザー割当てを削除できます。KEEP QUOTAを指定すると、表領域のすべてのユーザー割当てを保持できます。デフォルトはKEEP QUOTAです。

DBA_TS_QUOTASデータ・ディクショナリ・ビューを問い合せると、表領域のすべてのユーザー割当てを表示できます。

INCLUDING CONTENTS

表領域(消失書込み保護更新が格納されている表領域を含む)の内容をすべて削除するには、INCLUDING CONTENTSを指定します。データベース・オブジェクトを格納している表領域を削除する場合は、必ずこの句を指定します。表領域が空でない場合にこの句を省略した場合、エラーが戻され、表領域は削除されません。

1つの表のパーティションまたはサブパーティションが表領域に(すべてではなく)一部含まれていると、INCLUDING CONTENTSを指定してもDROP TABLESPACEコマンドが正常に実行されません。パーティション表のすべてのパーティションまたはサブパーティションがtablespaceに存在する場合、DROP TABLESPACE ... INCLUDING CONTENTSは、tablespaceを削除し、関連付けられた索引セグメント、LOBデータ・セグメントと索引セグメント、ネストした表のデータ・セグメント、および他の表領

域にある表の索引セグメントも削除します。

パーティション化された索引構成表の場合、すべての主キー索引セグメントがこの表領域に存在していれば、他の表領域にあるオーバーフロー・セグメントや関連するマッピング表もすべて削除されます。主キー索引セグメントのいくつかが存在しない場合、文は正常に実行されません。その場合、その表領域を削除する前に、ALTER TABLE ... MOVE PARTITIONを使用して、それらの主キー索引セグメントをこの表領域に移動し、この表領域にオーバーフロー・データ・セグメントが存在しないパーティションを削除します。また、パーティション化された索引構成表も削除します。

表領域がマテリアライズド・ビューのマスター表を含む場合、マテリアライズド・ビューは無効になります。

表領域がマテリアライズド・ビュー・ログを含む場合、このログおよびその表に関連付けられているダイレクト・パス・インサートのその他のリフレッシュ情報は削除されます。

AND DATAFILES

INCLUDING CONTENTSを指定するときにAND DATAFILES句を指定すると、関連するオペレーティング・システム・ファイルも削除できます。Oracle Databaseによって、アラート・ログに、削除された各オペレーティング・システム・ファイルに関するメッセージが書き込まれます。この句は、Oracle Managed Filesに対しては不要です。Oracle Managed Filesは、AND DATAFILESを指定しなくてもシステムから削除されます。

KEEP DATAFILES

INCLUDING CONTENTSを指定するときにKEEP DATAFILES句も指定すると、関連するオペレーティング・システム・ファイル(Oracle Managed Filesも含む)を処理せずにそのままにしておくことができます。この句を指定する必要があるのは、Oracle Managed Filesを使用しているときに、関連するオペレーティング・システム・ファイルをINCLUDING CONTENTS句で削除しない場合です。

CASCADE CONSTRAINTS

CASCADE CONSTRAINTSを指定すると、tablespaceに含まれる表の主キーまたは一意キーを参照する、tablespaceの外の表からすべての参照整合性制約を削除できます。このような参照整合性制約があるときにこの句を省略した場合、エラーが戻され、表領域は削除されません。

例

表領域の削除: 例

次の文は、tbs_01表領域を削除し、tbs_01に含まれる主キーおよび一意キーを参照するすべての参照整合性制約を削除します。

```
DROP TABLESPACE tbs_01
    INCLUDING CONTENTS
        CASCADE CONSTRAINTS;
```

シャドウ表領域の削除: 例

次の文は、シャドウ表領域内の追跡対象データを別のシャドウ表領域に移動させます。これは、十分な空き領域のあるPDBにシャドウ表領域がある場合にのみ機能します。

```
DROP TABLESPACE <shadow_tablespace_name>
```

次の文は、シャドウ表領域とその内容をすべて削除します。追跡対象データはすべて失われます。

シャドウ表領域とその内容の削除: 例

```
DROP TABLESPACE <shadow_tablespace_name>
    INCLUDING CONTENTS
```

オペレーティング・システム・ファイルの削除: 例

次の例は、tbs_02表領域およびそれに関連するすべてのオペレーティング・システムのデータファイルを削除します。

```
DROP TABLESPACE tbs_02
   INCLUDING CONTENTS AND DATAFILES;
```

DROP TABLESPACE SET

ノート:

この SQL 文は、Oracle Sharding を使用している場合にのみ有効です。Oracle Sharding の詳細は、『Oracle Database 管理者ガイド』を参照してください。

目的

DROP TABLESPACE SET文を使用すると、シャード・グループから表領域セットを削除できます。

削除した表領域セットは、ごみ箱内には移動しません。このため、削除した表領域セットを消去またはリカバリすることはできません。

関連項目:

「CREATE TABLESPACE SET」および「ALTER TABLESPACE SET」を参照してください。

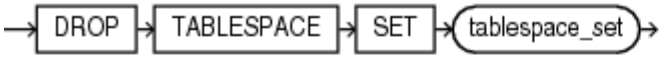
前提条件

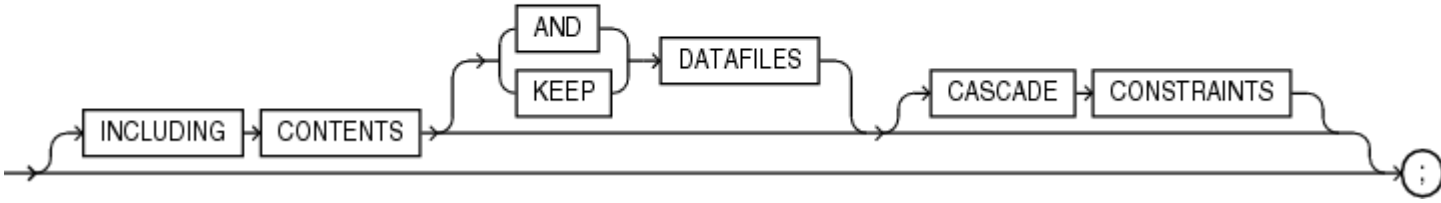
シャード・カタログ・データベースにSDBユーザーとして接続している必要があります。

DROP TABLESPACEシステム権限が必要です。アクティブ・トランザクションを保持するロールバック・セグメントを表領域に含む表領域セットは削除できません。

構文

drop_tablespace_set::=





セマンティクス

tablespace_set

削除する表領域セットの名前を指定します。

INCLUDING CONTENTS

この句では、削除操作中に表領域セット内の表領域に関連付けられているオブジェクトとデータファイルをデータベースで管理する方法を指定できます。INCLUDING CONTENTS句のセマンティクスは、DROP TABLESPACE文と同じです。この句のセマンティクスの詳細は、「INCLUDING CONTENTS」を参照してください。

例

表領域セットの削除：例

次の文は、表領域セット`ts1`を削除します。

```
DROP TABLESPACE SET ts1;
```

DROP TRIGGER

目的

トリガーはPL/SQLを使用して定義されます。トリガーの作成、変更および削除の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

DROP TRIGGER文を使用すると、データベースからデータベース・トリガーを削除できます。

関連項目:

「CREATE TRIGGER」および「ALTER TRIGGER」を参照してください。

前提条件

削除するトリガーが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY TRIGGERシステム権限が必要です。他のユーザーのスキーマ内のDATABASEにあるトリガーを削除する場合は、ADMINISTER DATABASE TRIGGERシステム権限も必要です。

構文

drop_trigger::=

セマンティクス

schema

トリガーが含まれているスキーマを指定します。schemaを指定しない場合、トリガーは自分のスキーマ内にあるとみなされます。

trigger

削除するトリガーの名前を指定します。データベースからトリガーが削除され、再度、起動されることはありません。

例

トリガーの削除: 例

次の文は、スキーマhr内のsalary_checkトリガーを削除します。

```
DROP TRIGGER hr.salary_check;
```

DROP TYPE

目的

オブジェクト型はPL/SQLを使用して定義されます。オブジェクト型の作成、変更および削除の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

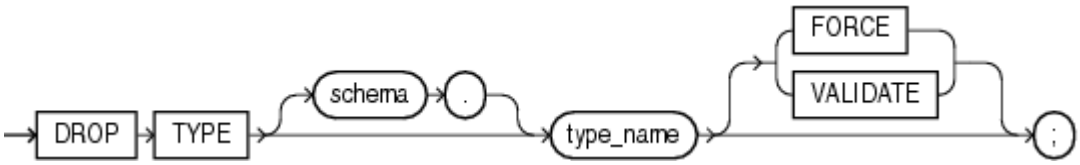
DROP TYPE文を使用すると、オブジェクト型、VARRAY型またはネストした表型の仕様部および本体を削除できます。

前提条件

削除するオブジェクト型、VARRAY型またはネストした表型が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY TYPEシステム権限が必要です。

構文

drop_type::=



セマンティクス

schema

型が含まれているスキーマを指定します。schemaを指定しない場合、型は自分のスキーマ内にあるとみなされます。

type_name

削除するオブジェクト型、VARRAY型またはネストした表型の名前を指定します。型依存性または表依存性のない型のみ削除できます。

type_nameがスーパータイプの場合、FORCEも指定しないと、この文は正常に実行されません。FORCEを指定すると、そのスーパータイプに依存するすべてのサブタイプが無効になります。

type_nameが統計タイプの場合、FORCEも指定しないと、この文は正常に実行されません。FORCEを指定した場合、最初にtype_nameに関連付けられたすべてのオブジェクトの関連付けが解除され、type_nameが削除されます。

関連項目:

統計タイプの詳細は、「ASSOCIATE STATISTICS」および「DISASSOCIATE STATISTICS」を参照してください。

type_nameが統計タイプに関連付けられたオブジェクト型の場合、最初にこの統計タイプからtype_nameの関連付けが解除され、その後でtype_nameが削除されます。ただし、統計タイプを使用して統計情報が収集された場合、この統計タイプからtype_nameの関連付けを解除することはできません。このため、この文は正常に実行されません。

type_nameが索引タイプの実装タイプの場合、索引タイプにINVALIDのマークが付けられます。

type_nameにパブリック・シノニムが定義されている場合、このシノニムも削除されます。

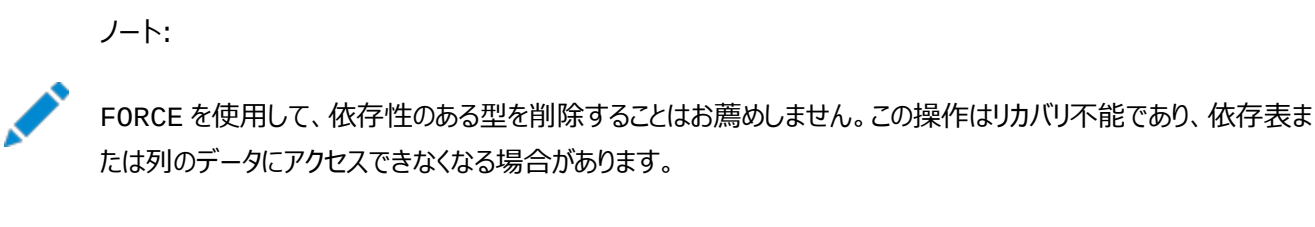
FORCEオプションを指定していない場合に削除できるのは、依存性のないスタンドアロンのスキーマ・オブジェクトとして定義されているオブジェクト型またはネストした表型またはVARRAY型のみです。これはデフォルトの動作です。

関連項目:

CREATE INDEXTYPE

FORCE

FORCEを指定すると、依存するデータベース・オブジェクトを持つ型でも強制的に削除できます。削除する型に依存するすべての列にUNUSEDのマークが付けられ、それらの列にアクセスできなくなります。



ノート:

FORCE を使用して、依存性のある型を削除することはお薦めしません。この操作はリカバリ不能であり、依存表または列のデータにアクセスできなくなる場合があります。

VALIDATE

型を削除するときにVALIDATEを指定すると、格納されているこの型のインスタンスがスーパータイプのいずれかの置換可能な列の範囲であることが検証されます。このようなインスタンスがない場合、削除操作が完了します。

この句はサブタイプのみに意味があります。明示的な型または表依存性のないサブタイプを安全に削除するために、このオプションの使用をお薦めします。

例

オブジェクト型の削除: 例

次の文は、オブジェクト型person_tを削除します。このオブジェクト型を作成する例については、『Oracle Database PL/SQL言語リファレンス』を参照してください。person_tに依存するすべての列は、UNUSEDのマークが付けられ、アクセスできなくなります。

```
DROP TYPE person_t FORCE;
```

DROP TYPE BODY

目的

オブジェクト型はPL/SQLを使用して定義されます。オブジェクト型の作成、変更および削除の詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

DROP TYPE BODY文を使用すると、オブジェクト型、VARRAY型またはネストした表型の本体を削除できます。型本体を削除しても、オブジェクト型の仕様部は残ります。また、削除した型本体は再作成できます。本体を再作成する前でも、そのオブジェクト型は使用できます。ただし、メンバー・ファンクションはコールできません。

前提条件

オブジェクト型の本体が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY TYPEシステム権限が必要です。

構文

drop_type_body::=

セマンティクス

schema

オブジェクト型が含まれているスキーマを指定します。schemaを指定しない場合、オブジェクト型は自分のスキーマ内にあるとみなされます。

type_name

削除するオブジェクト型の本体の名前を指定します。

型本体の削除の制限事項

型依存性または表依存性のない型本体のみ削除できます。

例

オブジェクト型本体の削除: 例

次の文は、オブジェクト型本体data_typ1を削除します。このオブジェクト型を作成する例については、『Oracle Database PL/SQL言語リファレンス』を参照してください。

```
DROP TYPE BODY data_typ1;
```

DROP USER

目的

DROP USER文を使用すると、データベース・ユーザーを削除できます。また、オプションでユーザーのオブジェクトを削除することもできます。

Oracle Automatic Storage Management (Oracle ASM)クラスタでは、AS SYSASMと認証されたユーザーは、この句を使用して、現行のノードのOracle ASMインスタンスに対してローカルなパスワード・ファイルでユーザーを削除できます。

ユーザーを削除すると、そのユーザーのすべてのスキーマ・オブジェクトもごみ箱から消去されます。

ノート:

SYS ユーザーまたは SYSTEM ユーザーを削除しないでください。これらのユーザーを削除すると、データベースが破損します。

関連項目:

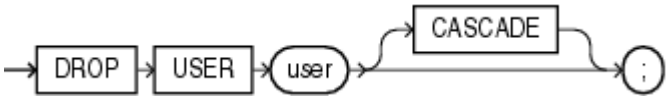
ユーザーの作成および変更の詳細は、「CREATE USER」および「ALTER USER」を参照してください。

前提条件

DROP USERシステム権限が必要です。Oracle ASMクラスタでは、AS SYSASMと認証される必要があります。

構文

drop_user::=



セマンティクス

user

削除するユーザーを指定します。CASCADEを指定しない場合、またはユーザーのオブジェクトを最初に明示的に削除しない場合、所有するスキーマにオブジェクトが含まれているユーザーは削除されません。

ユーザーの削除の制限事項

履歴追跡にフラッシュバック・データ・アーカイブを使用する表がスキーマに含まれているユーザーは削除できません。まず、表でのフラッシュバック・データ・アーカイブの使用を無効にする必要があります。

CASCADE

CASCADEを指定すると、ユーザーを削除する前に、そのユーザーのスキーマ内にあるすべてのオブジェクトを削除できます。所有するスキーマにオブジェクトが含まれているユーザーを削除する場合は、必ずこの句を指定します。

- ユーザーのスキーマに表がある場合、表が削除され、その表の主キーまたは一意キーを参照している他のユーザーのスキーマ内にある表の参照整合性制約も自動的に削除されます。

- この句で表が削除される場合、その表の列で作成されたすべてのドメイン索引も削除され、適切な削除ルーチンが起動されます。

  関連項目:

  これらのルーチンの詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。

- 他のスキーマにある次のオブジェクトは削除されず、無効にされます。
  - 削除されたユーザーのスキーマ内のビューまたはシノニム
  - 削除されたユーザーのスキーマ内のオブジェクトを問い合せるストアド・プロシージャ、ファンクションまたはパッケージ
- 削除されたユーザーのスキーマ内にある表に基づく他のスキーマ内のマテリアライズド・ビューは削除されません。ただし、実表は存在しないため、他のスキーマ内のマテリアライズド・ビューはリフレッシュできなくなります。
- ユーザーのスキーマにあるすべてのトリガーは削除されます。
- ユーザーが作成したロールは削除されません。

ノート:

Oracle Database では、FORCE を使用するとユーザーが所有するすべての型が削除されます。詳細は、「DROP TYPE」の「FORCE」キーワードを参照してください。

例

データベース・ユーザーの削除: 例

ユーザーSidneyのスキーマ内にオブジェクトがない場合は、次の文を発行してsidneyを削除できます。

```
DROP USER sidney;
```

sidneyのスキーマ内にオブジェクトがある場合は、次の文のようにCASCADE句を指定して、sidneyとそのオブジェクトを削除する必要があります。

```
DROP USER sidney CASCADE;
```

DROP VIEW

目的

DROP VIEW文を使用すると、データベースからビューまたはオブジェクト・ビューを削除できます。ビューを削除して再作成すると、ビューの定義を変更できます。

関連項目:

ビューの作成および変更の詳細は、「CREATE VIEW」および「ALTER VIEW」を参照してください。

前提条件

削除するビューが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY VIEWシステム権限が必要です。

構文

drop_view::=

セマンティクス

schema

ビューが含まれているスキーマを指定します。schemaを指定しない場合、ビューは自分のスキーマ内にあるとみなされます。

view

削除するビューの名前を指定します。

Oracle Databaseでは、ビューに依存するビュー、マテリアライズド・ビューおよびシノニムは削除されませんが、INVALIDのマークが付けられます。このようなビューおよびシノニムは削除または再定義するか、無効なビューやシノニムをもう一度有効にするビューを別に定義します。

指定するビューにサブビューが定義されている場合、サブビューも同様に無効になります。ビューがサブビューを持つかどうかを確認するには、USER_VIEWS、ALL_VIEWSまたはDBA_VIEWSデータ・ディクショナリ・ビューのSUPERVIEW_NAME列を問い合せます。

関連項目:

- 「CREATE TABLE」および「CREATE SYNONYM」を参照してください。
- 無効なマテリアライズド・ビューの再検証の詳細は、「ALTER MATERIALIZED VIEW」を参照してください。

CASCADE CONSTRAINTS

CASCADE CONSTRAINTSを指定すると、削除するビューの主キーまたは一意キーを参照するすべての参照整合性制約を削除できます。このような制約が存在する状態でこの句を省略すると、DROP文は正常に実行されません。

例

ビューの削除：例

次の文は、emp_viewビュー(「ビューの作成：例」で作成)を削除します。

```
DROP VIEW emp_view;
```

EXPLAIN PLAN

目的

EXPLAIN PLAN文を使用すると、指定したSQL文を実行するためにOracle Databaseが使用する実行計画を決定できます。この文によって、実行計画の各ステップを記述している行が、指定した表に挿入されます。SQLトレース機能の一部としてEXPLAIN PLAN文を発行することもできます。

この文によって、文を実行するコストも決まります。表にドメイン索引が定義されている場合、ユーザー定義のCPUおよびI/Oコストが挿入されます。

配布メディアのSQLスクリプトの中に、サンプル出力表PLAN_TABLEの定義があります。使用する出力表は、列の名前およびデータ型がこのサンプル表と同じである必要があります。このスクリプトの一般的な名前は、UTLXPLAN.SQLです。正確な名前および位置は、使用するオペレーティング・システムによって異なります。

キャッシュされたカーソルに関する情報は、次の動的パフォーマンス・ビューから得られます。

- SQLカーソルが使用する作業領域の詳細は、V$SQL_WORKAREAを問い合せます。
- キャッシュされたカーソルの実行計画の詳細は、V$SQL_PLANを問い合せます。
- キャッシュされたカーソルの各ステップでの実行統計または実行計画の操作(生成された行数、読み取られたブロック数など)は、V$SQL_PLAN_STATISTICSを問い合せます。
- 前述の3つのビューの処理結果を1つにまとめた情報を取得する場合は、V$SQL_PLAN_STATISTICS_ALLを問い合せます。
- この文の実行が監視されている場合、キャッシュされたカーソルの実行計画の各ステップまたは各操作での実行統計は、V$SQL_PLAN_MONITORに表示されます。MONITORヒントを使用すると、監視を強制的に適用できます。

関連項目:

- EXPLAIN PLANの出力、SQLトレース機能の使用方法、および実行計画の生成と解析方法については、『Oracle Database SQLチューニング・ガイド』を参照してください。
- 動的パフォーマンス・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

前提条件

EXPLAIN PLAN文を実行する場合、実行計画を格納する既存の出力表に行を挿入するための権限が必要です。

出力する実行計画の適用対象であるSQL文を実行するための権限も必要です。このSQL文でビューにアクセスする場合は、このビューの基礎になっているすべての表およびビューへのアクセス権限が必要です。このビューが別のビューに基づき、さらにこの別のビューが、ある表に基づいている場合、別のビューとそのビューの基礎になっている表へのアクセス権限が必要です。

EXPLAIN PLANで作成された実行計画を検証する場合は、出力表へ問い合せる権限が必要です。

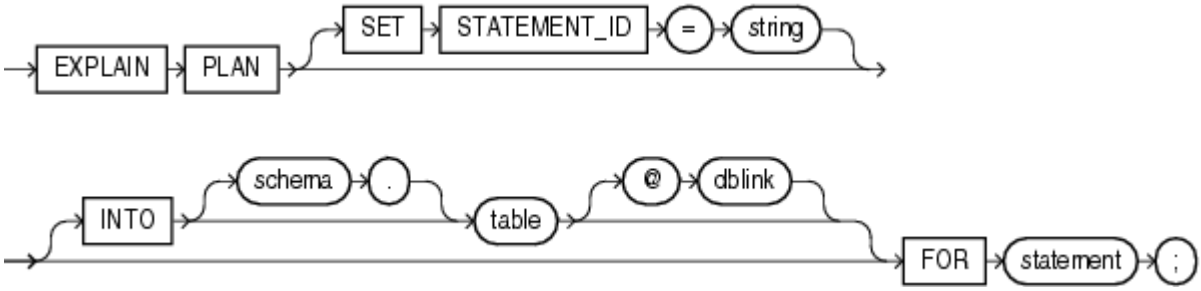
EXPLAIN PLAN文はデータ操作言語(DML)文であり、データ定義言語(DDL)文ではありません。そのため、EXPLAIN PLAN文で加えられた変更内容は暗黙的にコミットされません。出力表のEXPLAIN PLAN文で生成された行を保存する場合は、この文を指定したトランザクションをコミットする必要があります。

関連項目:

計画表の移入および問合せに必要な権限の詳細は、「INSERT」および「SELECT」を参照してください。

構文

explain_plan::=



セマンティクス

SET STATEMENT_ID句

実行計画が出力される表の中で、この実行計画に該当する行にあるSTATEMENT_ID列の値を指定します。この値によって、実行計画の行を出力表の中の他の行と区別できます。ユーザーの出力表に多数の実行計画の行が含まれている場合は、必ず、STATEMENT_IDの値を指定します。この句を省略した場合、デフォルトでSTATEMENT_ID値がNULLに設定されます。

INTO table句

出力表の名前を指定し、オプションとしてそのスキーマおよびデータベースの名前も指定します。この表は、EXPLAIN PLAN文を使用する前に作成しておく必要があります。

schemaを指定しない場合、表は自分のスキーマ内にあるとみなされます。

dblinkには、出力表が格納されているリモートのOracle Databaseに対するデータベース・リンクの完全な名前または名前の一部を指定します。Oracle Databaseの分散機能を使用している場合にのみ、リモート出力表を指定できます。dblinkを省略した場合、表がローカル・データベース上にあるとみなされます。データベース・リンクの参照方法の詳細は、「リモート・データベース内のオブジェクトの参照」を参照してください。

INTO句を省略した場合、出力表は、ローカル・データベース上の自分のスキーマ内にあるPLAN_TABLEであるとみなされます。

FOR statement句

実行計画生成の対象となるSELECT、INSERT、UPDATE、DELETE、MERGE、CREATE TABLE、CREATE INDEXまたはALTER INDEX ... REBUILD文を指定します。

EXPLAIN PLANのノート

EXPLAIN PLANには、次のノートがあります。

- statementにparallel_clauseを指定した場合、結果として生成される実行計画はパラレルで実行されます。ただし、EXPLAIN PLANコマンドによって計画表に実際に文が挿入されるため、ユーザーが発行したパラレルDML文は、トランザクションの最初のDML文ではなくなります。これは、トランザクション1つにつきパラレルDML文は1つというOracle Databaseの制限に違反するため、この文はシリアルで実行されます。文をパラレルで実行するには、EXPLAIN PLAN文をコミットまたはロールバックしてから、パラレルDML文を発行する必要があります。
- 一時表上で操作するための実行計画を判断する場合、EXPLAIN PLANは、同一セッションから実行する必要があり

ます。これは、一時表内のデータがセッション固有であるためです。

例

EXPLAIN PLANの例

次の文は、UPDATE文の実行計画およびコストを決定し、実行計画を記述した行をSTATEMENT_ID値'Raise in Tokyo'とともに、指定したplan_table表に挿入します。

```
EXPLAIN PLAN
    SET STATEMENT_ID = 'Raise in Tokyo'
    INTO plan_table
    FOR UPDATE employees
        SET salary = salary * 1.10
        WHERE department_id =
           (SELECT department_id FROM departments
               WHERE location_id = 1700);
```

次のSELECT文は、plan_table表への問合せを実行し、実行計画およびコストを戻します。

```
SELECT id, LPAD(' ',2*(LEVEL-1))||operation operation, options,
       object_name, object_alias, position
    FROM plan_table
    START WITH id = 0 AND statement_id = 'Raise in Tokyo'
    CONNECT BY PRIOR id = parent_id AND statement_id = 'Raise in Tokyo'
    ORDER BY id;
```

問合せによって、次の実行計画が戻されます。

```
 ID OPERATION            OPTIONS              OBJECT_NAME          OBJECT_ALIAS
POSITION
--- -------------------- -------------------- -------------------- ------------------
-- --------
  0 UPDATE STATEMENT
4
  1   UPDATE                                   EMPLOYEES
1
  2     INDEX            RANGE SCAN           EMP_DEPARTMENT_IX    EMPLOYEES@UPD$1
1
  3       TABLE ACCESS   BY INDEX ROWID       DEPARTMENTS          DEPARTMENTS@SEL$1
1
  4         INDEX        RANGE SCAN           DEPT_LOCATION_IX     DEPARTMENTS@SEL$1
1
```

POSITION列の1行目の値は、文のコストが4であることを示しています。

EXPLAIN PLAN: パーティション化された例

サンプル表sh.salesは、time_id列でパーティション化されます。パーティションsales_q3_2000には、2000年10月1日より小さい時刻の値があり、time_id列にローカル索引sales_time_bixがあります。

次の問合せについて考えてみます。

```
EXPLAIN PLAN FOR
  SELECT * FROM sales
     WHERE time_id BETWEEN :h AND '01-OCT-2000';
```

ここで、:hはすでに宣言されているバインド変数を指します。EXPLAIN PLAN文は、PLAN_TABLEを出力表とする次の問合せを実行します。次の問合せによって、パーティション情報などの基本的な実行計画が得られます。

```
SELECT operation, options, partition_start, partition_stop,
       partition_id
  FROM plan_table;
```

FLASHBACK DATABASE

目的

FLASHBACK DATABASE文を使用すると、データベースを過去の時点またはシステム変更番号(SCN)まで戻すことができます。この文を使用すると、データベースの不完全リカバリと同じ操作をより高速に実行できます。

FLASHBACK DATABASE操作の後、フラッシュバックされたデータベースに書込みアクセスするためには、ALTER DATABASE OPEN RESETLOGS文によってそのデータベースを再オープンする必要があります。

関連項目:

FLASHBACK DATABASEの詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

前提条件

SYSDBA、SYSBACKUP、またはSYSDGシステム権限が必要です。

マルチテナント・コンテナ・データベース(CDB)に接続しているときには、次の条件が適用されます。

- CDBをフラッシュバックするには、ルートに接続する必要があります。また、共通に付与されているSYSDBA、SYSBACKUPまたはSYSDGシステム権限が必要です。
- PDBをフラッシュバックするには、ルートに接続されている必要があり、共通に付与されているSYSDBA、SYSBACKUPまたはSYSDGシステム権限が必要であるか、またはフラッシュバックするPDBに接続されている必要があり、SYSDBA、SYSBACKUPまたはSYSDGシステム権限(共通に付与されている権限か、そのPDBでローカルに付与されている権限)が必要です。

高速リカバリ領域をデータベースに設定しておく必要があります。また、データベースを保証付きリストア・ポイントにフラッシュバックするのでなければ、ALTER DATABASE FLASHBACK ON文を使用して、データベースをFLASHBACKモードにしておく必要があります。データベースは、マウントされているがオープンされていない状態であることが必要です。

また、次のことに注意してください。

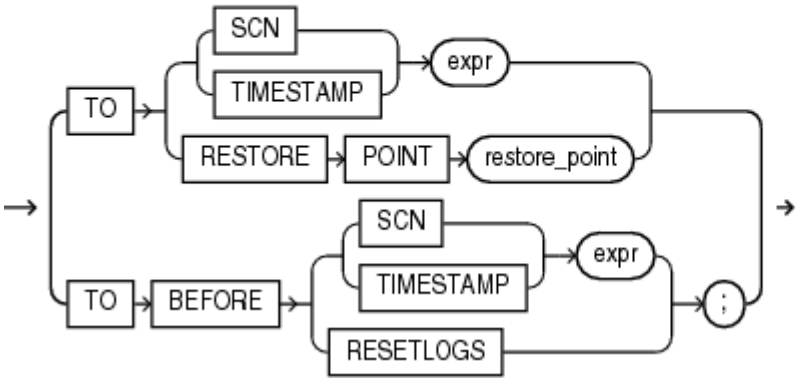
- データベースは、ARCHIVELOGモードで実行されている必要があります。
- データベースは、現行の制御ファイルとともにマウントされている必要がありますが、オープンされている必要はありません。制御ファイルは、バックアップおよび再作成することはできません。データベース制御ファイルがバックアップからリストアされるか、または再作成されると、既存のすべてのフラッシュバック・ログ情報は廃棄されます。
- データベースには、SQL文ALTER TABLESPACE ... FLASHBACK OFFを使用して、フラッシュバック機能が使用禁止にされたオンライン表領域は含めないでください。

関連項目:

- データベースをFLASHBACKモードにする方法については、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』および「ALTER DATABASE」の「flashback_mode_clause」を参照してください。
- リストア・ポイントおよび保証付きリストア・ポイントの詳細は、「CREATE RESTORE POINT」を参照してください。

構文

flashback_database::=



セマンティクス

FLASHBACK DATABASE文を発行すると、最初に、必要なすべてのアーカイブREDOログおよびオンラインREDOログが使用可能であるかどうかが確認されます。これらのログが使用可能な場合、データベース内で現在オンラインになっているすべてのデータファイルが、この文に指定したSCNまたは時刻まで戻されます。

- データベースに保持されるフラッシュバック・データの量は、DB_FLASHBACK_RETENTION_TARGET初期化パラメータおよび高速リカバリ領域のサイズによって制御されます。V$FLASHBACK_DATABASE_LOGビューを問い合せることによって、どの時点までデータベースをフラッシュバックできるかを判断できます。
- フラッシュバックを実行するために十分なデータがデータベースに残っていない場合、標準のリカバリ手順によってデータベースを過去のある時点の状態にリカバリできます。
- 一連のデータファイルに十分なデータが残っていない場合、エラーが戻されます。その場合、それらのデータファイルをオフラインにし、この文を再実行してデータベースの残りのデータファイルをリカバリできます。その後、標準のリカバリ手順を使用して、オフラインにしたデータファイルをリカバリできます。

関連項目:

データファイルのリカバリの詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

STANDBY

STANDBYを指定すると、スタンバイ・データベースを以前のSCNまたは時点まで戻すことができます。スタンバイ・データベースではない場合、エラーが戻されます。この句を省略すると、databaseには、プライマリ・データベースまたはスタンバイ・データベースのどちらでも指定できます。

関連項目:

スタンバイ・データベースでFLASHBACK DATABASEを使用して複数の遅延を実現する方法については、『Oracle Data Guard概要および管理』を参照してください。

PLUGGABLE

PLUGGABLEを指定してPDBをフラッシュバックします。現在のコンテナがルートまたはフラッシュバックするPDBのどちらであっても、この句を指定する必要があります。

PDBのフラッシュバックの制限事項

- プロキシPDBをフラッシュバックすることはできません。
- CDBが共有UNDOモードの場合は、PDBリストア・ポイントをクリーン・アップする場合にのみPDBをフラッシュバックできます。詳細は、CREATE RESTORE POINTのCLEAN句を参照してください。

database

非CDBまたはCDBをフラッシュバックする場合、オプションでフラッシュバックするデータベースの名前を指定できます。databaseを省略した場合は、初期化パラメータDB_NAMEの値によって特定されたデータベースがフラッシュバックされます。

PDBをフラッシュバックする際に現在のコンテナがルートである場合、databaseを使用してフラッシュバックするPDBの名前を指定します。PDBをフラッシュバックする際に現在のコンテナがそのPDBである場合、オプションでdatabaseを使用してPDB名を指定できます。

TO SCN句

システム変更番号(SCN)を指定します。

- TO SCNでは、指定したSCNの状態にデータベースが戻されます。
- TO BEFORE SCNでは、指定したSCNの直前のSCNの状態にデータベースが戻されます。

現行のSCNを判断するには、V$DATABASEビューのCURRENT_SCN列を問い合せます。これによって、高リスクのバッチ・ジョブを実行する前などに、SCNをスプール・ファイルに保存することもできます。

TO TIMESTAMP句

有効な日時式を指定します。

- TO TIMESTAMPでは、指定したタイムスタンプ時点の状態にデータベースが戻されます。
- TO BEFORE TIMESTAMPでは、指定したタイムスタンプの1秒前の状態にデータベースが戻されます。

タイムスタンプには、基準の値(SYSDATEなど)からのオフセットか、またはシステム・タイムスタンプの絶対値を指定できます。

TO RESTORE POINT句

この句を使用すると、指定したリストア・ポイントにデータベースをフラッシュバックできます。フラッシュバック・データベースが使用可能になっていない場合、このFLASHBACK DATABASE文では、この句のみを指定できます。データベースのモードがFLASHBACKでない場合、前述の「前提条件」で示したように、これは、この文で指定可能な唯一の句です。

RESETLOGS

TO BEFORE RESETLOGSを指定すると、データベースを最後のリセットログ操作(ALTER DATABASE OPEN RESETLOGS)の直前の状態にフラッシュバックできます。

関連項目:

この句の詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

例

データベースに高速リカバリ領域が設定してあり、メディア・リカバリが使用可能になっていると想定します。次の文は、データベースのFLASHBACKモードを有効にして、データベースをオープンします。

```
STARTUP MOUNT
```

```
ALTER DATABASE FLASHBACK ON;
ALTER DATABASE OPEN;
```

次の文は、データベースを1日以上オープンしていた場合、データベースを1日フラッシュバックします。

```
SHUTDOWN DATABASE
STARTUP MOUNT
FLASHBACK DATABASE TO TIMESTAMP SYSDATE-1;
```

FLASHBACK TABLE

目的

FLASHBACK TABLE文を使用すると、人為的エラーまたはアプリケーション・エラーが発生した場合に、表を以前の状態にリストアできます。表をフラッシュバックできる過去の時点は、システム内のUNDOデータの量によって異なります。また、Oracle Databaseでは、表の構造を変更するDDL操作が行われた場合は、表を以前の状態にリストアできません。

ノート:

UNDO_MANAGEMENT 初期化パラメータをデフォルトの AUTO のままにして、データベースを自動 UNDO モードで実行することをお薦めします。また、UNDO_RETENTION 初期化パラメータを、必要となる最も古いデータを含むために十分長い期間に設定してください。詳細は、UNDO_MANAGEMENT 初期化パラメータおよびUNDO_RETENTION 初期化パラメータに関するドキュメントを参照してください。

FLASHBACK TABLE文はロールバックできません。ただし、もう1つFLASHBACK TABLE文を発行し、現在の時間の直前の時間を指定することはできます。このため、FLASHBACK TABLE句を発行する前に、現在のSCNを記録しておくことをお薦めします。

関連項目:

- データベース全体を以前の状態に戻す方法の詳細は、「FLASHBACK DATABASE」を参照してください。
- 表から過去のデータを取り出す方法の詳細は、「SELECT」の「flashback_query_clause」を参照してください。
- FLASHBACK TABLE文を使用する方法の詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

前提条件

表を以前のSCNまたはタイムスタンプまでフラッシュバックするには、その表に対するFLASHBACKオブジェクト権限か、FLASHBACK ANY TABLEシステム権限が必要です。また、表に対するREADまたはSELECTオブジェクト権限と表に対するINSERT、DELETEおよびALTERオブジェクト権限が必要です。

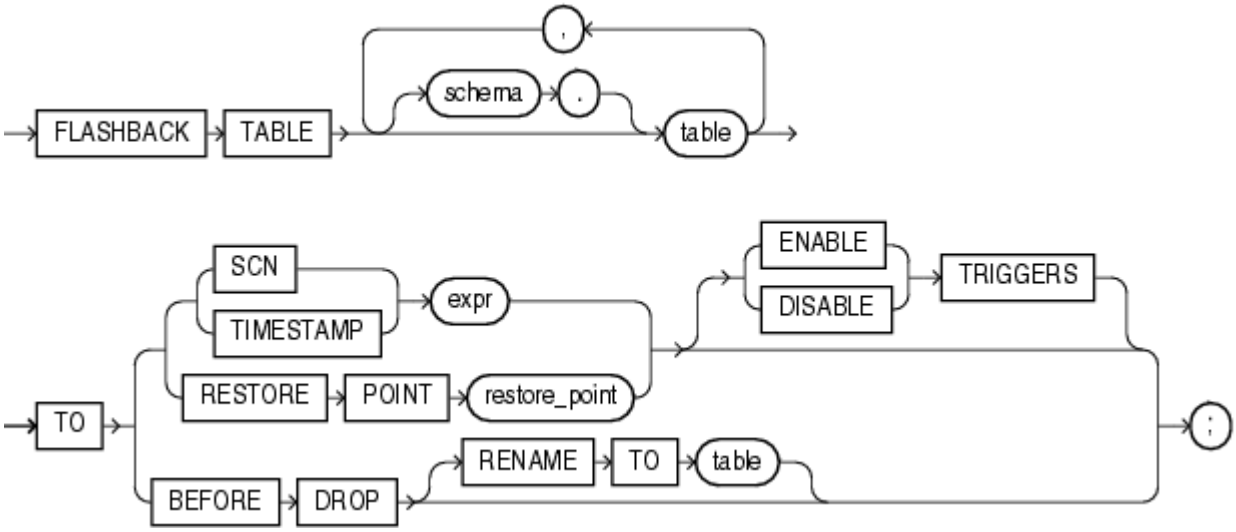
TO BEFORE DROPを使用せずに表をフラッシュバックする場合、フラッシュバック・リスト内のすべての表で、行の移動を有効にする必要があります。この操作はフラッシュバック削除といい、UNDOデータではなく、ごみ箱に削除されたデータを使用します。行の移動の有効化の詳細は、「row_movement_clause」を参照してください。

表をリストア・ポイントにフラッシュバックするには、SELECT ANY DICTIONARYまたはFLASHBACK ANY TABLEのいずれかのシステム権限か、SELECT_CATALOG_ROLEロールが必要です。

DROP TABLE操作の前まで表をフラッシュバックするために必要な権限は、その表の削除に必要な権限のみです。

構文

flashback_table::=



セマンティクス

Oracle Flashback Tableの操作中は、フラッシュバック・リストに指定されたすべての表が、排他DMLロックによってロックされます。これらのロックによって、表を以前の状態に戻す操作の間に、それらの表に他の操作が行われなくなります。

表のフラッシュバック操作は、フラッシュバック・リストに指定された表の数に関係なく、1回のトランザクションで実行されます。すべての表が以前の状態に戻されるか、何も戻されないかのいずれかです。いずれかの表のフラッシュバック操作が正常に実行されなかった場合、この文全体が正常に実行されません。

表のフラッシュバック操作が完了すると、表内のデータは、指定した過去の状態の表と同一になります。ただし、FLASHBACK TABLE TO SCNまたはTIMESTAMPでは行IDが保持されず、FLASHBACK TABLE TO BEFORE DROPでは参照制約がリカバリされません。

表に関連付けられた統計情報は、以前の形式には戻されません。表の現行の索引は戻され、フラッシュバック時点での表の状態が反映されます。現行の索引がフラッシュバック時点で存在していなかった場合、その索引は、フラッシュバック時点の表の状態を反映するように更新されます。ただし、フラッシュバック時点から現時点の間に削除された索引は、リストアされません。

schema

表が含まれているスキーマを指定します。schemaを指定しない場合、表は自分のスキーマ内にあるとみなされます。

table

以前の状態に戻すデータを含む1つ以上の表の名前を指定します。

表のフラッシュバックの制限事項

この文には、次の制限事項があります。

- 表のフラッシュバック操作は、クラスタ内の表、マテリアライズド・ビュー、アドバンスト・キューイング(AQ)表、静的データ・ディクショナリ表、システム表、リモート表、オブジェクト表、ネストした表、表の個々のパーティションまたはサブパーティションには無効です。
- 表のアップグレード、移動、切捨て、表への制約の追加、クラスタへの表の追加、列の変更または削除、列の暗号化キーの変更、パーティションまたはサブパーティションの追加、削除、マージ、分割、結合、切捨て(レンジ・パーティションの追加を除く)のDDL操作を実行すると表の構造が変更されるため、これらの操作後にTO SCNまたはTO TIMESTAMP句を使用して表をその操作以前の時点にフラッシュバックすることはできません。

TO SCN句

表を戻す時点に対応するシステム変更番号(SCN)を指定します。exprは、有効なSCNに評価される数値である必要があります。

TO TIMESTAMP句

表を戻す時点に対応するタイムスタンプ値を指定します。exprは、過去の有効なタイムスタンプに評価される必要があります。表は、指定したタイムスタンプの約3秒以内の時点にフラッシュバックされます。

TO RESTORE POINT句

表をフラッシュバックするリストア・ポイントを指定します。リストア・ポイントは作成済である必要があります。

関連項目:

リストア・ポイントの作成の詳細は、「CREATE RESTORE POINT」を参照してください。

ENABLE | DISABLE TRIGGERS

デフォルトでは、表のフラッシュバック操作中は、tableに定義したすべての有効なトリガーが無効にされ、表のフラッシュバック操作の完了後に再度有効にされます。このデフォルト動作を上書きして、フラッシュバック処理中もトリガーを有効にする必要がある場合、ENABLE TRIGGERSを指定します。

この句は、tableに定義され、すでに有効にされているデータベース・トリガーのみに影響します。現在無効になっているトリガーを選択して有効にするには、ALTER TABLE ... enable_disable_clauseを使用してから、FLASHBACK TABLE文にENABLE TRIGGERS句を指定して発行します。

TO BEFORE DROP句

この句を使用すると、削除された表およびすべての依存するオブジェクトをごみ箱から取り出すことができます。表は、SYSTEM表領域以外のローカル管理表領域内に置いておく必要があります。

関連項目:

- ごみ箱の詳細、およびごみ箱内のオブジェクトのネーミング規則の詳細は、『Oracle Database管理者ガイド』を参照してください。
- ごみ箱からオブジェクトを完全に削除する方法の詳細は、「PURGE」を参照してください。

ユーザーが指定した元の表名か、オブジェクトの削除時にそのオブジェクトに割り当てられたシステム生成名を指定できます。

- ごみ箱内のオブジェクトのシステム生成名は一意です。そのため、システム生成名を指定すると、その名前を持つ特定のオブジェクトが取り出されます。

  ごみ箱の内容を参照するには、USER_RECYCLEBINデータ・ディクショナリ・ビューを問い合せます。かわりにRECYCLEBINシノニムを使用することもできます。次の2つの文は、同じ行を戻します。

  ```
  SELECT * FROM RECYCLEBIN;
  SELECT * FROM USER_RECYCLEBIN;
  ```

- ユーザー指定の名前を指定した場合、その名前を持つオブジェクトがごみ箱内に複数存在していると、ごみ箱に移動した日時が一番近いオブジェクトが取り出されます。その表のより古い状態で取り出す場合、次のいずれかの操作を実行します。

- 取り出す表の、ごみ箱内でのシステム生成名を指定します。
- 必要な表が取り出されるまでFLASHBACK TABLE ... TO BEFORE DROP文を繰り返し発行します。

Oracle Databaseは、元の表名を保持します。元の表が削除された後、同じスキーマ内に削除された表と同じ名前を持つ新しい表が作成されていた場合、RENAME TO句も指定しないかぎりエラーが戻されます。

RENAME TO句

この句を使用すると、ごみ箱から取り出される表に新しい名前を指定できます。

削除された表のフラッシュバックのノート

削除された表のフラッシュバックには、次のノートがあります。

- ごみ箱から取り出された表に定義されたすべての索引(ビットマップ結合索引およびドメイン索引以外)が取り出されます。(ビットマップ結合索引およびドメイン索引はDROP TABLE操作中にごみ箱に移動しないため、取り出すことができません。)
- 表に定義されたすべてのトリガーおよび制約も取り出されます。ただし、他の表を参照する参照整合制約は取り出されません。

  取り出された索引、トリガーおよび制約には、ごみ箱での名前が付けられています。そのため、FLASHBACK TABLE ... TO BEFORE DROP文を発行する前に、USER_RECYCLEBINビューを問い合せて、取り出されたトリガーおよび制約の名前を使用しやすい名前に変更することをお薦めします。
- 表を削除すると、その表に定義されたすべてのマテリアライズド・ビュー・ログも削除されますが、これらはごみ箱には移動しません。そのため、マテリアライズド・ビュー・ログを表とともにフラッシュバックすることはできません。
- 表を削除すると、その表の索引も削除され、表とともにごみ箱に移動します。その後領域が不足すると、領域を再利用するために、まずごみ箱内の索引が消去されます。この場合、表をフラッシュバックしても、その表に定義されていた索引の一部が戻されない場合があります。
- ユーザーによって、または領域の再利用操作の結果として消去された表は、フラッシュバックできません。

例

以前の状態への表のリストア: 例

次の例では、新しい表employees_testを行の移動を有効にして作成し、新しい表内の値を更新し、FLASHBACK TABLE文を発行します。

hrサンプル・スキーマのemployees表から、employees_test表を、行の移動を有効にして作成します。

```
CREATE TABLE employees_test
  AS SELECT * FROM employees;
```

指標として、2500未満の給与をリストします。

```
SELECT salary
  FROM employees_test
  WHERE salary < 2500;
    SALARY
----------
      2400
      2200
      2100
      2400
      2200
```

ノート:

FLASHBACK TABLE 文によって使用されるマッピング表に SCN が伝播されるまで時間がかかるため、5 分以上待ってから次の文を発行してください。この例で、既存の表を使用した場合は、この待機時間は必要ありません。

表の行の移動を可能にします。

```
ALTER TABLE employees_test
   ENABLE ROW MOVEMENT;
```

給与が2500未満の従業員の給与を10%上げます。

```
UPDATE employees_test
  SET salary = salary * 1.1
  WHERE salary < 2500;
5 rows updated.
COMMIT;
```

2つ目の指標として、10%の昇給後にも2500未満である給与をリストします。

```
SELECT salary
  FROM employees_test
  WHERE salary < 2500;
    SALARY
----------
      2420
      2310
      2420
```

employees_test表を、現在のシステム時間より前の状態にリストアします。この例では、一連の例を迅速にテストできるように、1分間(実際にはこのような短い期間は設定しない)を使用しています。通常の環境では、より長い期間が経過します。

```
FLASHBACK TABLE employees_test
  TO TIMESTAMP (SYSTIMESTAMP - INTERVAL '1' minute);
```

2500未満の給与をリストします。前述のFLASHBACK TABLE文を発行したため、このリストは1つ目の指標リストと一致しています。

```
SELECT salary
  FROM employees_test
  WHERE salary < 2500;
    SALARY
----------
      2400
      2200
      2100
      2400
      2200
```

削除された表の取得: 例

次の文は、誤ってpm.print_media表を削除したときにそれを取得します。

```
FLASHBACK TABLE print_media TO BEFORE DROP;
```

pmスキーマ内に別のprint_media表が作成されていた場合、RENAME TO句を使用して、取り出された表の名前を変更します。

```
FLASHBACK TABLE print_media TO BEFORE DROP RENAME TO print_media_old;
```

複数回削除された従業員表を最も古い状態で取り出す必要がある場合、USER_RECYLEBIN表を問い合せてシステム生成名を判断し、その名前をFLASHBACK TABLE文で使用します。(ご使用のデータベースでのシステム生成名は、ここに示すものとは異なります。)

```
SELECT object_name, droptime FROM user_recyclebin
   WHERE original_name = 'PRINT_MEDIA';
OBJECT_NAME                    DROPTIME
------------------------------ -------------------
RB$$45703$TABLE$0              2003-06-03:15:26:39
RB$$45704$TABLE$0              2003-06-12:12:27:27
RB$$45705$TABLE$0              2003-07-08:09:28:01
```

GRANT

目的

GRANT文を使用すると、次の権限またはロールを付与できます。

- ユーザーおよびロールに対するシステム権限。表18-1に、システム権限のリストを、操作対象のデータベース・オブジェクト別に示します。

  ANYシステム権限(SELECT ANY TABLE,など)は、SYSオブジェクトやその他のディクショナリ・オブジェクトでは機能しません。

- ユーザー、ロール、およびプログラム・ユニットに対するロール。付与されたロールは、ユーザー定義(ローカルまたは外部)または事前定義済のいずれかです。事前定義されているロールのリストは、『Oracle Databaseセキュリティ・ガイド』を参照してください。

- ユーザーおよびロールに対する、特定のオブジェクトに対するオブジェクト権限。表18-2に、オブジェクト権限のリストを、操作対象のデータベース・オブジェクト別に示します。

ノート:

グローバル・ロール(IDENTIFIED GLOBALLY で作成)は、エンタープライズ・ロールを介して付与します。GRANT 文を使用して付与することはできません。

データベース・ユーザーの認可のノート

データベース以外の手段とGRANT文を使用して、データベース・ユーザーを認可できます。

- 多くのOracle Database権限は、提供されるPL/SQLパッケージおよびJavaパッケージを介して付与されます。これらの権限の詳細は、該当するパッケージのドキュメントを参照してください。
- オペレーティング・システムによっては、Oracle Databaseユーザーに対して、初期化パラメータOS_ROLESでロールを付与できます。オペレーティング・システム機能を使用してユーザーにロールを付与する場合、GRANT文でユーザーにロールを付与することはできなくなります。ただし、GRANT文を使用して、ユーザーにシステム権限を付与したり、その他のロールにシステム権限およびロールを付与することはできます。

Oracle Automatic Storage Managementのノート

AS SYSASMとして認証されたユーザーは、この文を使用して、現在のノードのOracle ASMパスワード・ファイルで、システム権限SYSASM、SYSOPERおよびSYSDBAをユーザーに付与できます。

エディション化可能なオブジェクトのノート:

エディション化可能なオブジェクトにオブジェクト権限を付与するGRANT操作を実行すると、オブジェクトが現行のエディションで実体化されます。エディションおよびエディション化可能なオブジェクトの詳細は、『Oracle Database開発ガイド』を参照してください。

関連項目:

- ローカル、グローバルおよび外部権限の定義については、「CREATE USER」および「CREATE ROLE」を参照してください。

- その他の認可方法および権限の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。
- 権限付与の取消しについては、「REVOKE」を参照してください。

前提条件

システム権限を付与にするには、次のいずれかの条件が満たされている必要があります。

- GRANT ANY PRIVILEGEシステム権限が付与されている必要があります。この場合、ロールがユーザーのセッションで有効になっていないかぎり、システム権限をロールに付与しても、ロールが付与されているユーザーに権限は付与されません。
- ADMIN OPTION付きのシステム権限が付与されている必要があります。この場合、ロールがユーザーのセッションで有効になっているかどうかに関係なく、システム権限をロールに付与すると、ロールが付与されているユーザーに権限が付与されます。

ユーザーに対するロールまたは別のロールを付与する場合は、ADMIN OPTION付きのロールが直接付与されているか、GRANT ANY ROLEシステム権限が付与されているか、または付与するロールが自分で作成したロールである必要があります。

自分のスキーマ内のプログラム・ユニットにロールを付与する場合は、ADMIN OPTIONまたはDELEGATE OPTION付きのロールが直接付与されているか、GRANT ANY ROLEシステム権限が付与されているか、または付与するロールが自分で作成したロールである必要があります。

別のユーザーのスキーマ内のプログラム・ユニットに対するロールを付与するには、ユーザーSYSである必要があり、ロールがスキーマ所有者によって作成されているか、直接スキーマ所有者に付与されている必要があります。

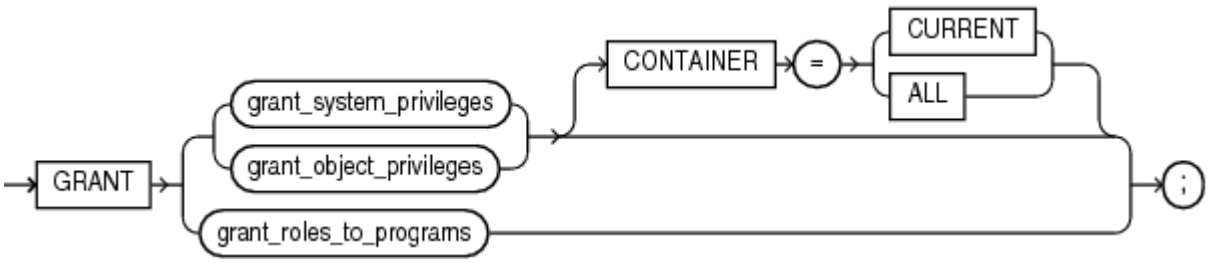
on_object_clauseのON USER句を指定してobject privilege on a user権限をユーザーに付与する場合は、その権限が付与されたユーザーであるか、そのユーザーに対するオブジェクト権限がWITH GRANT OPTION付きで付与されているか、GRANT ANY OBJECT PRIVILEGEシステム権限が付与されている必要があります。GRANT ANY OBJECT PRIVILEGEを所持しているという理由だけで、あるユーザーに対するオブジェクト権限を付与できる場合は、*_TAB_PRIVSビューのGRANTOR列には、GRANT文を発行したユーザーではなく、権限が付与されたユーザーが表示されます。

その他のオブジェクトのタイプすべてにオブジェクト権限を付与する場合は、オブジェクトの所有者であるか、オブジェクトの所有者からWITH GRANT OPTION付きのオブジェクト権限が付与されている必要があります。または、GRANT ANY OBJECT PRIVILEGEシステム権限が付与されている必要があります。GRANT ANY OBJECT PRIVILEGEを持っている場合は、オブジェクトの所有者が付与可能なオブジェクト権限と同じオブジェクト権限のみを付与できます。その場合、*_TAB_PRIVSビューのGRANTOR列には、GRANT文を発行したユーザーではなく、オブジェクトの所有者が表示されます。

CONTAINER句を指定する場合は、マルチテナント・コンテナ・データベース(CDB)に接続している必要があります。CONTAINER = ALLを指定する場合は、現在のコンテナがルートである必要があります。

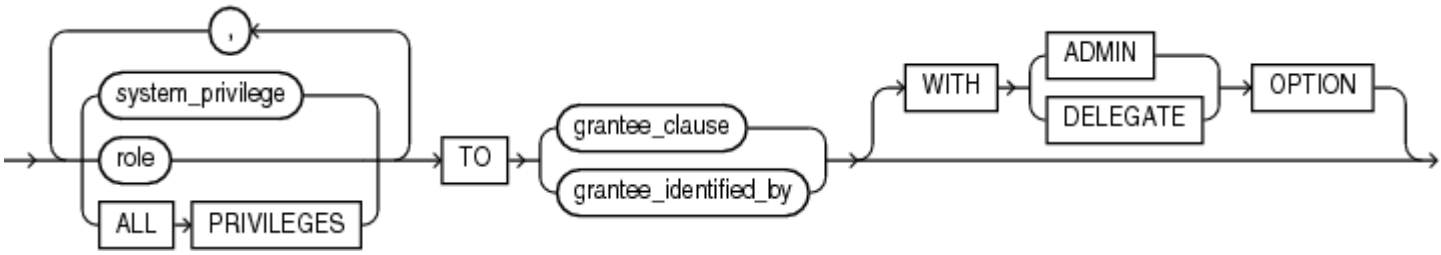
構文

grant::=



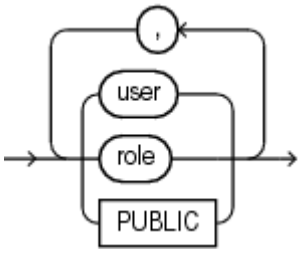
(grant_system_privileges::=、grant_object_privileges::=、grant_roles_to_programs::=)

grant_system_privileges::=



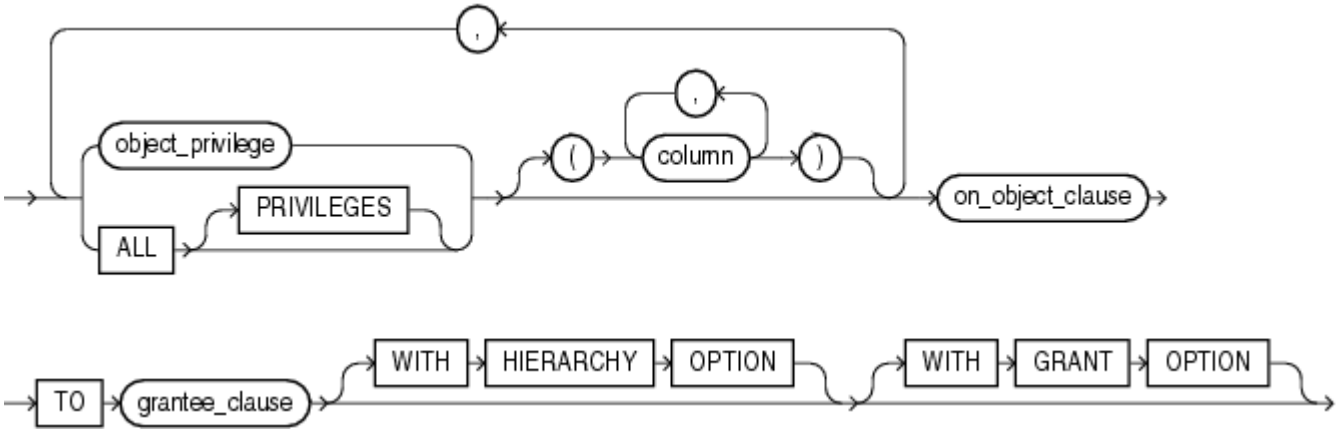
(grantee_clause::=、grantee_identified_by::=)

grantee_clause::=



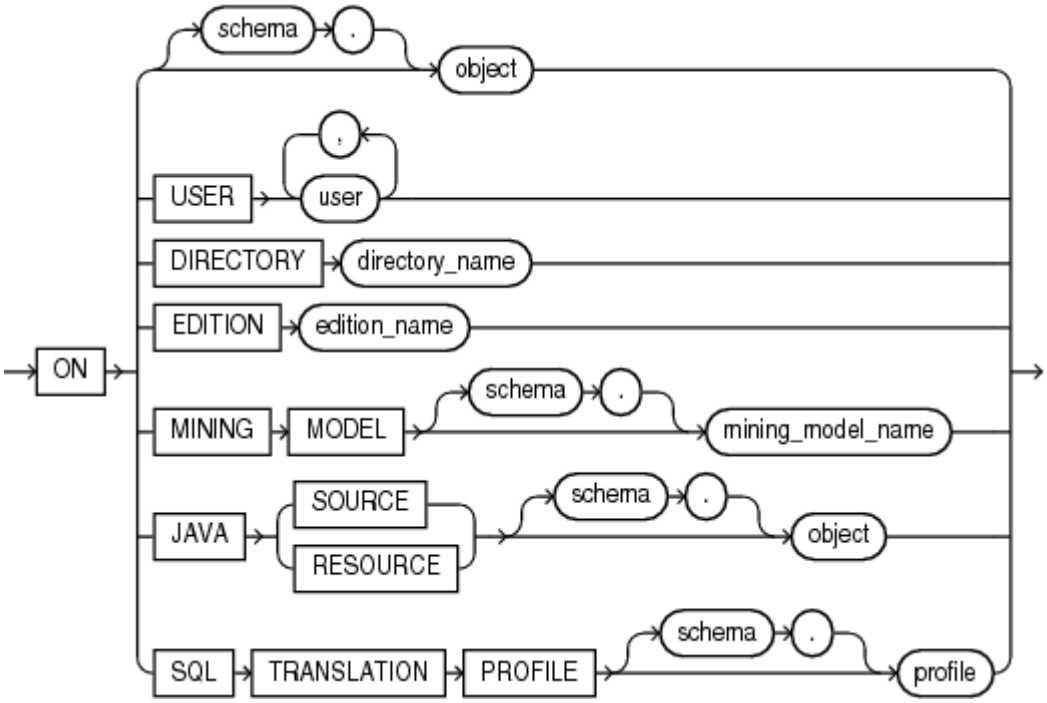
grantee_identified_by::=

grant_object_privileges::=

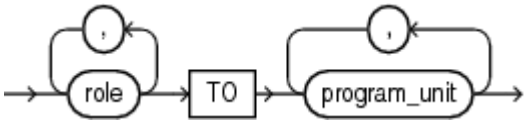


(on_object_clause::=、grantee_clause::=)

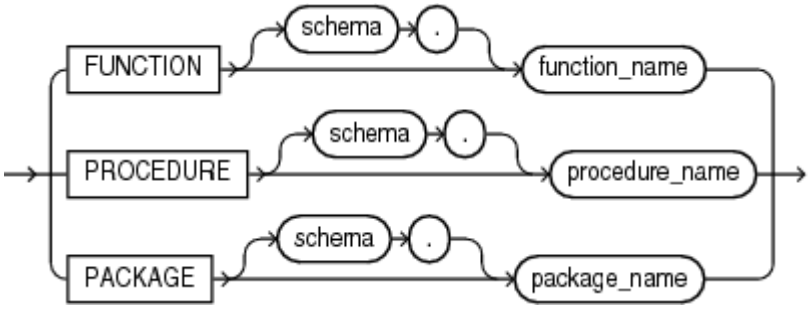
on_object_clause::=



grant_roles_to_programs::=



program_unit::=



セマンティクス

grant_system_privileges

システム権限を付与するには、次の句を使用します。

system_privilege

付与するシステム権限を指定します。表18-1に、システム権限のリストを、操作対象のデータベース・オブジェクト別に示します。

- ユーザーに権限を付与する場合、ユーザーの権限ドメインに権限が登録されます。このユーザーはその権限をすぐに使用できます。ANY権限は、信頼できるユーザーにのみ付与することをお薦めします。
- ロールに権限を付与する場合、ロールの権限ドメインに権限が登録されます。ロールが付与されているまたは使用可能となっているユーザーは、その権限をすぐに使用できます。なお、ロールが付与されている他のユーザーも、そのロールを使用可能にしてその権限を使用できます。

関連項目:

「ユーザーに対するシステム権限の付与: 例」および「ロールに対するシステム権限の付与: 例」を参照してください。

- PUBLICに権限を付与する場合、各ユーザーの権限ドメインに権限が登録されます。すべてのユーザーは、その権限によって許可される操作をすぐに実行できます。システム権限をPUBLICに付与しないことをお薦めします。

role

付与するロールを指定します。Oracle Databaseの事前定義ロールまたはユーザー定義ロールを付与できます。

- ユーザーにロールを付与する場合、そのユーザーが、付与されたロールを使用できます。ユーザーはそのロールをすぐに使用可能にし、ロールの権限ドメインに登録されている権限を使用できます。

  保護アプリケーション・ロールは、ユーザーに直接付与する必要はありません。ユーザーが適切なセキュリティ・ポリシーを渡すことを前提として、関連付けられたPL/SQLパッケージでこの処理を行うことができます。詳細は、「USING package」 の「CREATE ROLEセマンティクス」および『Oracle Databaseセキュリティ・ガイド』を参照してください。

- 他のロールにロールを付与する場合、権限受領者のロールの権限ドメインに、権限付与者のロールの権限ドメインが登録されます。権限受領者のロールを付与されているユーザーは、それを使用可能にした場合、付与されたロールの権限ドメイン内の権限を使用できます。

- PUBLICにロールを付与する場合、すべてのユーザーがロールを使用できます。すべてのユーザーはそのロールをすぐに使用可能にし、ロールの権限ドメインに登録されている権限を使用できます。

ALL PRIVILEGES

ALL PRIVILEGESを指定すると、SELECT ANY DICTIONARY、ALTER DATABASE LINKおよびALTER PUBLIC DATABASE LINK権限を除く、表18-1に示すすべてのシステム権限を付与できます。

ただし、ALL PRIVILEGESの付与および取消しはADMINISTER KEY MANAGEMENTには適用されません。ALL PRIVILEGESを付与しても、ADMINISTER KEY MANAGEMENTは付与されません。同様に、ALL PRIVILEGESを取り消しても、ADMINISTER KEY MANAGEMENTは取り消されません。

関連項目:

- Oracleの事前定義されているロールの詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。
- ロールへのロールの付与: 例
- ユーザー定義ロールの作成については、「CREATE ROLE」を参照してください。

grantee_clause

grantee_clauseを使用すると、システム権限、ロールまたはオブジェクト権限が付与されるユーザーまたはロールを指定できます。

PUBLIC

PUBLICを指定すると、すべてのユーザーに権限を付与できます。システム権限をPUBLICに付与しないことをお薦めします。

権限受領者の制限事項

grantee_clauseにユーザー、ロールまたはPUBLICを指定できるのは1回のみです。

grantee_identified_by

grantee_identified_by句を使用すると、ユーザーにシステム権限とロールを付与するときにユーザーにパスワードを割り当てることができます。ユーザーとパスワードを同じ数だけ指定する必要があります。最初のパスワードは最初のユーザーに、2番目のパスワードは2番目のユーザーに、というように割り当てられます。指定したユーザーが存在する場合、データベースはそのユーザーのパスワードをリセットします。指定したユーザーが存在しない場合、データベースはパスワードとともにユーザーを作成します。

関連項目:

「CREATE USER」(ユーザー名とパスワードの制限事項について)および「システム権限の付与時のユーザー・パスワードの割当て: 例」を参照してください。

WITH ADMIN OPTION

WITH ADMIN OPTIONを指定すると、権限受領者は次のことができます。

- ロールがGLOBALロールでない場合、権限またはロールを他のユーザーまたはロールに付与できます。
- 権限またはロールを他のユーザーまたは他のロールから取り消すことができます。
- 権限またはロールへのアクセスに必要な認可を変更するため、その権限またはロールを変更できます。
- 権限またはロールを削除します。
- 権限を付与したスキーマ内のプログラム・ユニットに、ロールを付与します。
- 権限を付与したスキーマ内のプログラム・ユニットからロールを取り消します。

たとえば、WITH ADMIN OPTIONを指定せずにユーザーにロールまたはシステム権限を付与し、後でWITH ADMIN OPTIONを指定してその権限およびロールを付与した場合、そのユーザーはその権限またはロールに関してADMIN OPTIONを持つことになります。

システム権限またはロールのADMIN OPTIONをユーザーから取り消す場合、そのユーザーの権限またはロールを完全に取り消し、その次にADMIN OPTIONを指定せずに権限またはロールをユーザーに付与します。

関連項目:

ADMIN OPTION付きロールの付与: 例

WITH DELEGATE OPTION

ロールをユーザーに付与する場合のみ、この句を指定できます。

WITH DELEGATE OPTIONを指定すると、権限受領者は次のことができます。

- 権限を付与したスキーマ内のプログラム・ユニットに、ロールを付与します。
- 権限を付与したスキーマ内のプログラム・ユニットからロールを取り消します。

たとえば、WITH DELEGATE OPTIONを指定せずにユーザーにロールを付与し、後でWITH DELEGATE OPTIONを指定してそのロールを付与した場合、そのユーザーはそのロールに関してDELEGATE OPTIONを持つことになります。

ロールのDELEGATE OPTIONをユーザーから取り消す場合、そのユーザーのロールを完全に取り消し、その次にDELEGATE OPTIONを指定せずにロールをユーザーに付与する必要があります。

関連項目:

- DELEGATE OPTION付きロールの付与: 例
- プログラム・ユニットへのロールの付与の詳細は、grant_roles_to_programs句を参照してください。

システム権限とロールの付与の制限事項

権限およびロールには、次の制限事項があります。

- 付与する権限およびロールのリストに、権限またはロールを指定できるのは1回のみです。
- ロールに対してそのロールと同じロールは付与できません。
- ロールIDENTIFIED GLOBALLYは、どのようなユーザーまたはロールに対しても付与できません。
- ロールIDENTIFIED EXTERNALLYは、グローバル・ユーザーまたはグローバル・ロールに対して付与できません。
- ロールは交互に付与できません。たとえば、ロールbankerをロールtellerに付与した場合、逆にtellerをbankerに付与することはできません。
- IDENTIFIED BYロール、IDENTIFIED USINGロールおよびIDENTIFIED EXTERNALLYロールの別のロールへの付与はできません。

grant_object_privileges

オブジェクト権限を付与するには、次の句を使用します。

object_privilege

付与するオブジェクト権限を指定します。表18-2に、オブジェクト権限のリストを、付与されるオブジェクトのタイプ別に示します。ユーザーまたはロールのいずれかのオブジェクト権限をエディション化可能なオブジェクトに付与すると、オブジェクトは付与したエディションで実体化されます。エディション化可能なオブジェクト型およびエディションの詳細は、「CREATE EDITION」を参照してください。

ノート:

ビューに対する SELECT を他のユーザーに付与するには、ビューの基礎となるすべてのオブジェクトを所有しているか、またはこれらすべての基礎となるオブジェクトに対する SELECT オブジェクト権限を WITH GRANT OPTION で付与されている必要があります。これは、これらの基礎となるオブジェクトに対する SELECT 権限を権限受領者がすでに持っている場合にも当てはまります。

ビューに対する READ を他のユーザーに付与するには、ビューの基礎となるすべてのオブジェクトを所有しているか、またはこれらすべての基礎となるオブジェクトに対する READ または SELECT オブジェクト権限を WITH GRANT OPTION で付与されている必要があります。これは、これらの基礎となるオブジェクトに対する READ または SELECT 権限を権限受領者がすでに持っている場合にも当てはまります。

オブジェクト権限の制限事項

付与する権限のリストに、権限を指定できるのは1回のみです。

ALL [PRIVILEGES]

ALLを指定すると、GRANT OPTION付きで付与されているオブジェクト権限に対するすべての権限を付与できます。オブジェクトが定義されているスキーマを所有しているユーザーは、自動的にGRANT OPTION付きのオブジェクトに対するすべての権限を持っています。キーワードPRIVILEGESはセマンティクスを明確にするためのものであり、指定は任意です。

column

権限を付与する表またはビューの列を指定します。INSERT、REFERENCESまたはUPDATEの各権限を付与する場合にのみ、列を指定できます。列を指定しない場合、権限受領者には表またはビューのすべての列に対して指定した権限が付与されます。

既存の列オブジェクトに対して付与された権限については、データ・ディクショナリ・ビューUSER_COL_PRIVS、ALL_COL_PRIVSまたはDBA_COL_PRIVSを問い合せます。

関連項目:

データ・ディクショナリ・ビューの詳細は、『Oracle Databaseリファレンス』および「別々の列に対する複数のオブジェクト権限の付与: 例」を参照してください。

on_object_clause

on_object_clauseを指定すると、権限が付与されているオブジェクトを識別できます。ユーザー、ディレクトリ・オブジェクト、エディション、データ・マイニング・モデル、Javaソースおよびリソース・スキーマ・オブジェクト、およびSQL翻訳プロファイルは、異なるネームスペースに格納されるため、別々に識別されます。

関連項目:

ロールへのオブジェクト権限の付与: 例

object

権限を付与するスキーマ・オブジェクトを指定します。objectをschemaで修飾しなかった場合、そのオブジェクトは自分のスキーマ内にあるとみなされます。オブジェクトには次のタイプがあります。

- 表、ビューまたはマテリアライズド・ビュー
- 順序
- プロシージャ、ファンクションまたはパッケージ
- ユーザー定義型
- これらの項目のシノニム
- ディレクトリ、ライブラリ、演算子または索引タイプ
- Javaソース、クラスまたはリソース

パーティション表の単一パーティションに直接権限を付与することはできません。

関連項目:

「表に対するオブジェクト権限のユーザーへの付与: 例」、「ビューに対するオブジェクト権限の付与: 例」および「別のスキーマの順序へのオブジェクト権限の付与: 例」を参照してください。

ON USER

権限を付与するデータベース・ユーザーを1人以上指定します。

ユーザーへの権限の付与の制限事項

PUBLICユーザーには権限を付与できません。

関連項目:

ユーザーに対するオブジェクト権限の付与: 例

ON DIRECTORY

権限を付与するディレクトリ・オブジェクトの名前を指定します。directory_nameはスキーマ名で修飾できません。

関連項目:

「CREATE DIRECTORY」および「ディレクトリに対するオブジェクト権限の付与: 例」を参照してください。

ON EDITION

USEオブジェクト権限を付与するエディションの名前を指定します。edition_nameはスキーマ名で修飾できません。

ON MINING MODEL

権限を付与するマイニング・モデルの名前を指定します。mining_model_nameをschemaで修飾しなかった場合、そのマイニング・モデルは自分のスキーマ内にあるとみなされます。

ON JAVA SOURCE | RESOURCE

権限を付与するJavaソースまたはリソース・スキーマ・オブジェクトの名前を指定します。objectをschemaで修飾しなかった場合、そのオブジェクトは自分のスキーマ内にあるとみなされます。

関連項目:

CREATE JAVA

ON SQL TRANSLATION PROFILE

権限を付与するSQL翻訳プロファイルの名前を指定します。profileをschemaで修飾しなかった場合、そのプロファイルは自分のスキーマ内にあるとみなされます。

WITH HIERARCHY OPTION

WITH HIERARCHY OPTIONを指定すると、この文の後に作成されるサブオブジェクトも含め、objectのすべてのサブオブジェクト(ビューを基に作成したサブビューなど)に対する指定したオブジェクト権限を付与できます。

この句は、READまたはSELECTオブジェクト権限とあわせて指定する場合のみ意味があります。

WITH GRANT OPTION

WITH GRANT OPTIONを指定すると、権限受領者による、他のユーザーまたはロールに対するオブジェクト権限の付与を許可できます。

たとえば、WITH GRANT OPTIONを指定せずにユーザーにオブジェクト権限を付与し、後でWITH GRANT OPTIONを指定し

てその権限を付与した場合、そのユーザーはその権限に関してGRANT OPTIONを持つことになります。

オブジェクト権限のGRANT OPTIONをユーザーから取り消す場合、そのユーザーの権限を完全に取り消し、その次にGRANT OPTIONを指定せずに権限をユーザーに付与する必要があります。

WITH GRANT OPTIONの付与の制限事項

ロールではなく、ユーザーまたはPUBLICに付与する場合にのみ、WITH GRANT OPTIONを指定できます。

grant_roles_to_programs

この句を使用して、ロールをプログラム・ユニットに付与します。そのようなロールは、コード・ベースのアクセス制御(CBAC)ロールと呼ばれます。

role

付与するロールを指定します。Oracle Databaseの事前定義ロールまたはユーザー定義ロールを付与できます。ロールはプログラム・ユニットのスキーマ所有者によって作成されているか、直接付与する必要があります。

program_unit

ロールを付与するプログラム・ユニットを指定します。PL/SQLファンクション、プロシージャ、またはパッケージを指定できます。schemaを指定しない場合、ファンクション、プロシージャ、またはパッケージは独自のスキーマにあると見なされます。

関連項目:

プログラム・ユニットへのコードに基づくアクセス制御ロールの割当ての詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

CONTAINER句

現在のコンテナがプラガブル・データベース(PDB)である場合:

- CONTAINER = CURRENTを指定すると、システム権限、オブジェクト権限またはロールをユーザーまたはロールにローカルに付与できます。現在のPDBでのみ、その権限またはロールがユーザーまたはロールに付与されます。

現在のコンテナがルートである場合:

- CONTAINER = CURRENTを指定すると、システム権限、オブジェクト権限またはロールを共通ユーザーまたは共通ロールにローカルに付与できます。ルートでのみ、ユーザーまたはロールにその権限が付与されます。
- CONTAINER = ALLを指定すると、システム権限、共通オブジェクトに対するオブジェクト権限またはロールを共通ユーザーまたは共通ロールに共通に付与できます。

この句を省略すると、CONTAINER = CURRENTがデフォルトになります。

ノート:

権限またはロールを付与するときに CONTAINER 句を指定する場合は、現在のコンテナが同じコンテナである必要があります。また、その権限またはロールを取り消すときには、同じ CONTAINER 句を指定する必要があります。詳細は、REVOKE 文の「CONTAINER 句」を参照してください。

表18-1 システム権限(操作対象のデータベース・オブジェクト別)

システム権限名	許可されている操作
アドバイザ・フレームワーク権限(すべてのアドバイザ・フレームワーク権限は、DBA ロールに含まれています。)	—
ADVISOR	PL/SQL パッケージ(DBMS_ADVISOR や DBMS_SQLTUNE など)を介したアドバイザ・フレームワークへのアクセス。
ADMINISTER SQL TUNING SET	DBMS_SQLTUNE パッケージを介した、権限受領者が所有する SQL チューニング・セットの作成、削除、選択(読取り)およびロード(書込み)。
ADMINISTER ANY SQL TUNING SET	DBMS_SQLTUNE パッケージを介した、任意のユーザーが所有する SQL チューニング・セットの作成、削除、選択(読取り)およびロード(書込み)。
CREATE ANY SQL PROFILE	Enterprise Manager または DBMS_SQLTUNE パッケージを介してアクセスする、SQL チューニング・アドバイザが推奨した SQL プロファイルの受入れ ノート: この権限は現在非推奨になっています。かわりに、ADMINISTER SQL MANAGEMENT OBJECT を使用してください。
ALTER ANY SQL PROFILE	既存の SQL プロファイルの属性の変更。 ノート: この権限は現在非推奨になっています。かわりに、ADMINISTER SQL MANAGEMENT OBJECT を使用してください。
DROP ANY SQL PROFILE	既存の SQL プロファイルの削除。 ノート: この権限は現在非推奨になっています。かわりに、ADMINISTER SQL MANAGEMENT OBJECT を使用してください。
ADMINISTER SQL MANAGEMENT OBJECT	DBMS_SQLTUNE パッケージを介して任意のユーザーが所有している SQL プロファイルの作成、変更および削除。
分析ビュー	—
CREATE ANALYTIC VIEW	権限を付与したスキーマ内での分析ビューの作成。

システム権限名	許可されている操作
CREATE ANY ANALYTIC VIEW	SYS、AUDSYS.を除く任意のスキーマ内での分析ビューの作成。
ALTER ANY ANALYTIC VIEW	SYS、AUDSYS を除く任意のスキーマ内での分析ビューの名前の変更。
DROP ANY ANALYTIC VIEW	SYS、AUDSYS を除く任意のスキーマ内での分析ビューの削除。
属性ディメンション	—
CREATE ATTRIBUTE DIMENSION	権限を付与したスキーマ内での属性ディメンションの作成。
CREATE ANY ATTRIBUTE DIMENSION	SYS、AUDSYS を除く任意のスキーマ内での属性ディメンションの作成。
ALTER ANY ATTRIBUTE DIMENSION	SYS、AUDSYS を除く任意のスキーマ内での属性ディメンションの名前の変更。
DROP ANY ATTRIBUTE DIMENSION	SYS、AUDSYS を除く任意のスキーマ内での属性ディメンションの削除。
クラスタ:	—
CREATE CLUSTER	権限を付与したスキーマ内でのクラスタの作成。
CREATE ANY CLUSTER	SYS、AUDSYS を除く任意のスキーマ内でのクラスタの作成。CREATE ANY TABLE と同様に動作します。
ALTER ANY CLUSTER	SYS、AUDSYS を除く任意のスキーマ内でのクラスタの変更。
DROP ANY CLUSTER	SYS、AUDSYS を除く任意のスキーマ内でのクラスタの削除。
コンテキスト:	—
CREATE ANY CONTEXT	任意のコンテキスト・ネームスペースの作成。
DROP ANY CONTEXT	任意のコンテキスト・ネームスペースの削除。
データ・リダクション:	—

システム権限名	許可されている操作
EXEMPT REDACTION POLICY	既存の Oracle Data Redaction ポリシーのバイパス、および Data Redaction が定義されている表またはビューの実際のデータの表示。
データベース:	—
ALTER DATABASE	データベースの変更。
ALTER SYSTEM	ALTER SYSTEM 文の発行
AUDIT SYSTEM	AUDIT 文の発行。
データベース・リンク:	—
CREATE DATABASE LINK	権限を付与したスキーマ内でのプライベート・データベース・リンクの作成。
CREATEPUBLICDATABASELINK	パブリック・データベース・リンクの作成。
ALTER DATABASE LINK	接続または認証ユーザーのパスワードが変更された場合の固定ユーザー・データベース・リンクの変更。
ALTER PUBLIC DATABASE LINK	接続または認証ユーザーのパスワードが変更された場合のパブリック固定ユーザー・データベース・リンクの変更。
DROPPUBLIC DATABASELINK	パブリック・データベース・リンクの削除。
デバッグ:	—
DEBUG CONNECT SESSION	デバッガへの現行のセッションの接続。
DEBUG ANY PROCEDURE	任意のデータベース・オブジェクトのすべての PL/SQL コードおよび Java コードのデバッグ。アプリケーションによって実行されたすべての SQL 文に関する情報の表示。 ノート: この権限を付与することは、データベース内の適用可能なすべてのオブジェクトについて DEBUG オブジェクト権限を付与することと同じです。

システム権限名	許可されている操作
DICTIONARIES:	—
ANALYZE ANY DICTIONARY	データ・ディクショナリ・オブジェクトの分析。
ディメンション:	—
CREATE DIMENSION	権限を付与したスキーマ内でのディメンションの作成。
CREATE ANY DIMENSION	SYS、AUDSYS を除く任意のスキーマ内でのディメンションの作成。
ALTER ANY DIMENSION	SYS、AUDSYS を除く任意のスキーマ内でのディメンションの変更。
DROP ANY DIMENSION	SYS、AUDSYS を除く任意のスキーマ内でのディメンションの削除。
ディレクトリ:	—
CREATE ANY DIRECTORY	ディレクトリ・データベース・オブジェクトの作成。
DROP ANY DIRECTORY	ディレクトリ・データベース・オブジェクトの削除。
エディション:	—
CREATE ANY EDITION	エディションの作成。
DROP ANY EDITION	エディションの削除。
フラッシュバック・データ・アーカイブ:	—
FLASHBACK ARCHIVE ADMINISTER	フラッシュバック・データ・アーカイブの作成、変更または削除。
階層	—
CREATE HIERARCHY	権限を付与したスキーマ内での階層の作成。
CREATE ANY HIERARCHY	SYS、AUDSYS を除く任意のスキーマ内での階層の作成。
ALTER ANY HIERARCHY	SYS、AUDSYS を除く任意のスキーマ内での階層の名前変更。

システム権限名	許可されている操作
DROP ANY HIERARCHY	SYS、AUDSYS を除く任意のスキーマ内での階層の削除。
索引:	—
CREATE ANY INDEX	SYS、AUDSYS を除く任意のスキーマ内でのドメイン索引の作成、または SYS、AUDSYS を除く任意のスキーマ内での任意の表のインデックスの作成。
ALTER ANY INDEX	SYS、AUDSYS を除く任意のスキーマ内での索引の変更。
DROP ANY INDEX	SYS、AUDSYS を除く任意のスキーマ内での索引の削除。
索引タイプ:	—
CREATE INDEXTYPE	権限を付与したスキーマ内での索引タイプの作成。
CREATE ANY INDEXTYPE	SYS を除く任意のスキーマ内での索引タイプの作成、および SYS を除く任意のスキーマ内の索引タイプに対するコメントの作成。
ALTER ANY INDEXTYPE	SYS、AUDSYS を除く任意のスキーマ内での索引タイプの変更。
DROP ANY INDEXTYPE	SYS、AUDSYS を除く任意のスキーマ内での索引タイプの削除。
EXECUTE ANY INDEXTYPE	SYS、AUDSYS を除く任意のスキーマ内での索引タイプの参照。
ジョブ・スケジューラ・オブジェクト:	次の権限は、DBMS_SCHEDULER パッケージのプロシージャを実行する場合に必要です。この権限は、データベース・オブジェクトではない軽量ジョブには適用されません。軽量ジョブの詳細は、『Oracle Database 管理者ガイド』を参照してください。
CREATE JOB	権限を付与したスキーマ内でのジョブ、チェーン、スケジュール、プログラム、資格証明、リソース・オブジェクトまたは非互換性リソース・オブジェクトの作成、変更または削除。
CREATE ANY JOB	SYS、AUDSYS を除く任意のスキーマ内でのジョブ、チェーン、スケジュール、プログラム、資格証明、リソース・オブジェクトまたは非互換性リソース・オブジェクトの作成、変更または削除。

システム権限名	許可されている操作
	ノート: この強力な権限を使用すると、権限受領者は任意のユーザーとしてコードを実行できます。この権限の付与には注意が必要です。
CREATE EXTERNAL JOB	権限を付与したスキーマ内での、オペレーティング・システム上で実行可能なスケジューラ・ジョブの作成。
EXECUTE ANY CLASS	権限を付与したスキーマ内でのジョブのジョブ・クラスの指定。
EXECUTE ANY PROGRAM	権限を付与したスキーマ内でのジョブの任意のプログラムの使用。
MANAGE SCHEDULER	任意のジョブ・クラス、ウィンドウまたはウィンドウ・グループの作成、変更または削除。
USE ANY JOB RESOURCE	権限を付与したスキーマ内での、任意のスケジュール・リソース・オブジェクトの任意のプログラムまたはジョブとの関連付け。
キー管理フレームワーク:	—
ADMINISTER KEY MANAGEMENT	キーおよびキーストアの管理。
ライブラリ:	注意: CREATE LIBARARY、CREATE ANY LIBRARY、ALTER ANY LIBRARY および EXECUTE ANY LIBRARY は、非常に強力な権限であり、信頼できるユーザーにのみ付与する必要があります。これらの権限を付与する前に、『Oracle Database セキュリティ・ガイド』を参照してください。
CREATE LIBRARY	権限を付与したスキーマ内での外部プロシージャまたはファンクション・ライブラリの作成。
CREATE ANY LIBRARY	SYS、AUDSYS を除く任意のスキーマ内での外部プロシージャまたはファンクション・ライブラリの作成。
ALTER ANY LIBRARY	SYS、AUDSYS を除く任意のスキーマ内での外部プロシージャまたはファンクション・ライブラリの変更。
DROP ANY LIBRARY	SYS、AUDSYS を除く任意のスキーマ内での外部プロシージャまたはファンクション・ライブラリの削除。

システム権限名	許可されている操作
EXECUTE ANY LIBRARY	SYS、AUDSYS を除く任意のスキーマ内での外部プロシージャまたはファンクション・ライブラリの使用。
LogMiner:	—
LOGMINING	CDB または PDB の DBMS_LOGMNR パッケージのプロシージャを実行します。V$LOGMNR_CONTENTS ビューの内容の問合せ
マテリアライズド・ビュー:	—
CREATE MATERIALIZED VIEW	権限を付与したスキーマ内でのマテリアライズド・ビューの作成。
CREATE ANY MATERIALIZED VIEW	SYS、AUDSYS を除く任意のスキーマ内でのマテリアライズド・ビューの作成。
ALTER ANY MATERIALIZED VIEW	SYS、AUDSYS を除く任意のスキーマ内でのマテリアライズド・ビューの変更。
DROP ANY MATERIALIZED VIEW	SYS、AUDSYS を除く任意のスキーマ内でのマテリアライズド・ビューの削除。
QUERY REWRITE	この権限は現在非推奨になっています。ユーザー自身のスキーマ内の表またはビューを参照するマテリアライズド・ビューのリライトを可能にする場合、権限は不要です。
GLOBAL QUERY REWRITE	マテリアライズド・ビューが SYS を除く任意のスキーマ内の表またはビューを参照している場合のそのマテリアライズド・ビューを使用したリライトの有効化。
ON COMMIT REFRESH	データベースの任意の表に対する REFRESH ON COMMIT モードのマテリアライズド・ビューの作成。 データベースの任意の表に対するマテリアライズド・ビューのモードを REFRESH ON DEMAND モードから REFRESH ON COMMIT モードに変更。
FLASHBACK ANY TABLE	SYS を除く任意のスキーマ内の任意の表、ビューまたはマテリアライズド・ビューでの SQL フラッシュバック問合せの発行。この権限は、DBMS_FLASHBACK プロシージャの実行には不要です。

システム権限名	許可されている操作
マイニング・モデル:	—
CREATE MINING MODEL	権限を付与したスキーマでの DBMS_DATA_MINING.CREATE_MODEL プロシージャによるマイニング・モデルの作成
CREATE ANY MINING MODEL	SYS、AUDSYS を除く任意のスキーマでの、DBMS_DATA_MINING.CREATE_MODEL プロシージャを使用したマイニング・モデルの作成。
ALTER ANY MINING MODEL	SYS、AUDSYS を除く任意のスキーマでの、該当する DBMS_DATA_MINING プロシージャを使用した、マイニング・モデル名またはモデルの関連コスト・マトリックスの変更。
DROP ANY MINING MODEL	SYS、AUDSYS を除く任意のスキーマでの、DBMS_DATA_MINING.DROP_MODEL プロシージャを使用したマイニング・モデルの削除。
SELECT ANY MINING MODEL	SYS、AUDSYS を除く任意のスキーマでのマイニング・モデルのスコアリングまたは表示。スコアリングの実行には、SQL ファンクションの PREDICTION ファミリ、または DBMS_DATA_MINING.APPLY プロシージャを使用します。モデルの表示は、DBMS_DATA_MINING.GET_MODEL_DETAILS_*プロシージャで実行されます。
COMMENT ANY MINING MODEL	SYS、AUDSYS を除く任意のスキーマでの、SQL COMMENT 文を使用したマイニング・モデルに対するコメントの作成。
OLAP キューブ:	Oracle Database を OLAP オプションで使用している場合は、次の権限が有効です。
CREATE CUBE	権限を付与したスキーマでの OLAP キューブの作成。
CREATE ANY CUBE	SYS、AUDSYS を除く任意のスキーマ内での OLAP キューブの作成。
ALTER ANY CUBE	SYS、AUDSYS を除く任意のスキーマ内での OLAP キューブの変更。
DROP ANY CUBE	SYS、AUDSYS を除く任意のスキーマ内での OLAP キューブの削除。

システム権限名	許可されている操作
SELECT ANY CUBE	SYS、AUDSYS を除く任意のスキーマでの OLAP キューブの問合せまたは表示。
UPDATE ANY CUBE	SYS、AUDSYS を除く任意のスキーマ内での OLAP キューブの更新。
OLAP キューブ・メジャー・フォルダ:	Oracle Database を OLAP オプションで使用している場合は、次の権限が有効です。
CREATE MEASURE FOLDER	権限を付与したスキーマでの OLAP メジャー・フォルダの作成。
CREATE ANY MEASURE FOLDER	SYS、AUDSYS を除く任意のスキーマ内での OLAP メジャー・フォルダの作成。
DELETE ANY MEASURE FOLDER	SYS、AUDSYS を除く任意のスキーマ内での、OLAP メジャー・フォルダからのメジャーの削除。
DROP ANY MEASURE FOLDER	SYS、AUDSYS を除く任意のスキーマ内での OLAP メジャー・フォルダの削除。
INSERT ANY MEASURE FOLDER	SYS、AUDSYS を除く任意のスキーマ内での、OLAP メジャー・フォルダへのメジャーの挿入。
OLAP キューブ・ディメンション:	Oracle Database を OLAP オプションで使用している場合は、次の権限が有効です。
CREATE CUBE DIMENSION	権限を付与したスキーマでの OLAP キューブ・ディメンションの作成。
CREATE ANY CUBE DIMENSION	SYS、AUDSYS を除く任意のスキーマ内での OLAP キューブ・ディメンションの作成。
ALTER ANY CUBE DIMENSION	SYS、AUDSYS を除く任意のスキーマ内での OLAP キューブ・ディメンションの変更。
DELETE ANY CUBE DIMENSION	SYS、AUDSYS を除く任意のスキーマ内での OLAP キューブ・ディメンションからの削除。
DROP ANY CUBE DIMENSION	SYS、AUDSYS を除く任意のスキーマ内での OLAP キューブ・ディメンションの削除。

システム権限名	許可されている操作
INSERT ANY CUBE DIMENSION	SYS、AUDSYS を除く任意のスキーマ内での OLAP キューブ・ディメンションへの挿入。
SELECT ANY CUBE DIMENSION	SYS、AUDSYS を除く任意のスキーマ内での OLAP キューブ・ディメンションの表示または問合せ。
UPDATE ANY CUBE DIMENSION	SYS、AUDSYS を除く任意のスキーマ内での OLAP キューブ・ディメンションの更新。
OLAP キューブの作成プロセス:	—
CREATE CUBE BUILD PROCESS	権限を付与したスキーマでの OLAP キューブの作成プロセスの作成。
CREATE ANY CUBE BUILD PROCESS	SYS、AUDSYS を除く任意のスキーマ内での、OLAP キューブの作成プロセスの作成。
DROP ANY CUBE BUILD PROCESS	SYS、AUDSYS を除く任意のスキーマ内での、OLAP キューブの作成プロセスの削除。
UPDATE ANY CUBE BUILD PROCESS	SYS、AUDSYS を除く任意のスキーマ内での、OLAP キューブの作成プロセスの更新。
演算子:	—
CREATE OPERATOR	権限を付与したスキーマ内での演算子およびバインディングの作成。
CREATE ANY OPERATOR	任意のスキーマ内での演算子とそのバインディングの作成、および任意のスキーマ内の演算子に対するコメントの作成。
ALTER ANY OPERATOR	任意のスキーマ内での演算子の変更。
DROP ANY OPERATOR	任意のスキーマ内での演算子の削除。
EXECUTE ANY OPERATOR	任意のスキーマ内での演算子の参照。
アウトライン:	—

システム権限名	許可されている操作
CREATE ANY OUTLINE	アウトラインを使用する任意のスキーマ内で使用するパブリック・アウトラインの作成。
ALTER ANY OUTLINE	アウトラインの変更。
DROP ANY OUTLINE	アウトラインの削除。
PDB ロックダウン・プロファイル:	—
CREATE LOCKDOWN PROFILE	PDB ロックダウン・プロファイルの作成。
ALTER LOCKDOWN PROFILE	PDB ロックダウン・プロファイルの変更。
DROP LOCKDOWN PROFILE	PDB ロックダウン・プロファイルの削除。
計画管理:	—
ADMINISTER SQL MANAGEMENT OBJECT	様々な SQL 文に対して保持されている計画の履歴および SQL 計画ベースラインの制御操作の実行。
プラガブル・データベース:	—
CREATE PLUGGABLE DATABASE	PDB の作成。 CDB から以前に切断された PDB の接続。 PDB のクローニング。
SET CONTAINER	この権限が付与されていたコンテナへの共通ユーザーによる切替えの許可。この権限は、共通ユーザーまたは共通ロールにのみ付与できます。
プロシージャ:	—
CREATE PROCEDURE	権限を付与したスキーマ内でのストアド・プロシージャ、ストアド・ファンクションまたはストアド・パッケージの作成。
CREATE ANY PROCEDURE	SYS、AUDSYS を除く任意のスキーマ内での、ストアド・プロシージャ、ストアド・ファンクションまたはストアド・パッケージの作成。

システム権限名	許可されている操作
ALTER ANY PROCEDURE	SYS、AUDSYS を除く任意のスキーマ内での、ストアド・プロシージャ、ストアド・ファンクションまたはストアド・パッケージの変更。
DROP ANY PROCEDURE	SYS、AUDSYS を除く任意のスキーマ内での、ストアド・プロシージャ、ストアド・ファンクションまたはストアド・パッケージの削除。
EXECUTE ANY PROCEDURE	プロシージャまたはファンクションの実行(スタンドアロンまたはパッケージ)。 SYS、AUDSYS を除く任意のスキーマ内でのパブリック・パッケージ変数の参照。
INHERIT ANY REMOTE PRIVILEGES	現在のユーザー・データベース・リンクを含む定義者権限プロシージャまたはファンクションの実行。
プロファイル:	—
CREATE PROFILE	プロファイルの作成。
ALTER PROFILE	プロファイルの変更。
DROP PROFILE	プロファイルの削除。
ロール:	—
CREATE ROLE	ロールの作成。
ALTER ANY ROLE	データベース内の任意のロールの変更。
DROP ANY ROLE	ロールの削除。
GRANT ANY ROLE	データベース内の任意のロールの付与。
ロールバック・セグメント:	—
CREATE ROLLBACK SEGMENT	ロールバック・セグメントの作成。
ALTER ROLLBACK SEGMENT	ロールバック・セグメントの変更。

システム権限名	許可されている操作
DROP ROLLBACK SEGMENT	ロールバック・セグメントの削除。
順序:	—
CREATE SEQUENCE	権限を付与したスキーマ内での順序の作成。
CREATE ANY SEQUENCE	SYS、AUDSYS を除く任意のスキーマ内での順序の作成。
ALTER ANY SEQUENCE	SYS、AUDSYS を除く任意のスキーマ内での順序の変更。
DROP ANY SEQUENCE	SYS、AUDSYS を除く任意のスキーマ内での順序の削除。
SELECT ANY SEQUENCE	SYS、AUDSYS を除く任意のスキーマ内での順序の参照。
セッション:	—
CREATE SESSION	データベースへの接続。
ALTER RESOURCE COST	セッション・リソースに対するコストの設定。
ALTER SESSION	SQL トレース機能の有効と無効の切替え。
RESTRICTED SESSION	SQL*Plus の STARTUP RESTRICT 文を使用した、インスタンス起動後のログイン
スナップショット:	MATERIALIZED VIEWS を参照してください
SQL TRANSLATION PROFILES:	—
CREATE SQL TRANSLATION PROFILE	権限を付与したスキーマ内での SQL 翻訳プロファイルの作成。
CREATE ANY SQL TRANSLATION PROFILE	SYS、AUDSYS を除く任意のスキーマ内での SQL 翻訳プロファイルの作成。
ALTER ANY SQL TRANSLATION PROFILE	SYS、AUDSYS を除く任意のスキーマ内での、SQL 翻訳プロファイルのトランスレータ、カスタム SQL 文の翻訳、またはカスタム・エラーの翻訳の変更。

システム権限名	許可されている操作
USE ANY SQL TRANSLATION PROFILE	SYS、AUDSYS を除く任意のスキーマ内での SQL 翻訳プロファイルの使用。
DROP ANY SQL TRANSLATION PROFILE	SYS、AUDSYS を除く任意のスキーマ内での SQL 翻訳プロファイルの削除。
TRANSLATE ANY SQL	権限受領者の SQL 翻訳プロファイル使用した任意のユーザー用の SQL の変換。
シノニム:	注意: CREATE PUBLIC SYNONYM および DROP PUBLIC SYNONYM は、非常に強力な権限であり、信頼できるユーザーにのみ付与する必要があります。これらの権限を付与する前に、『Oracle Database セキュリティ・ガイド』を参照してください。
CREATE SYNONYM	権限を付与したスキーマ内でのシノニムの作成。
CREATE ANY SYNONYM	SYS、AUDSYS を除く任意のスキーマ内でのプライベート・シノニムの作成。
CREATE PUBLIC SYNONYM	パブリック・シノニムの作成。
DROP ANY SYNONYM	SYS、AUDSYS を除く任意のスキーマ内でのプライベート・シノニムの削除
DROP PUBLIC SYNONYM	パブリック・シノニムの削除。
表:	ノート: 外部表の場合、有効な権限は、CREATE ANY TABLE、ALTER ANY TABLE、DROP ANY TABLE、READ ANY TABLE および SELECT ANY TABLE です。
CREATE TABLE	権限を付与したスキーマ内での表の作成。
CREATE ANY TABLE	SYS、AUDSYS を除く任意のスキーマ内での表の作成。なお、表が設定されるスキーマの所有者は、表領域内にその表を定義するための割当て制限が必要です。
ALTER ANY TABLE	SYS、AUDSYS を除く任意のスキーマ内での表またはビューの変更。

システム権限名	許可されている操作
BACKUP ANY TABLE	SYS、AUDSYS を除く他のユーザーのスキーマからの、エクスポート・ユーティリティを使用したオブジェクトの増分エクスポート。
DELETE ANY TABLE	SYS、AUDSYS を除く任意のスキーマ内での、表、表パーティションまたはビューからの行の削除。
DROP ANY TABLE	SYS、AUDSYS を除く任意のスキーマ内での、表または表パーティションの削除または切捨て。
INSERT ANY TABLE	SYS、AUDSYS を除く任意のスキーマ内での表およびビューへの行の挿入。
LOCK ANY TABLE	SYS、AUDSYS を除く任意のスキーマ内での表およびビューのロック。
READ ANY TABLE	SYS、AUDSYS を除く任意のスキーマ内での、表、ビューまたはマテリアライズド・ビューの問合せ。
SELECT ANY TABLE	SYS、AUDSYS を除く任意のスキーマ内での、表、ビューまたはマテリアライズド・ビューの問合せ。SELECT ... FOR UPDATE を使用して、行ロックを取得します。
FLASHBACK ANY TABLE	SYS、AUDSYS を除く任意のスキーマ内での、任意の表、ビューまたはマテリアライズド・ビューでの SQL フラッシュバック問合せの発行。この権限は、DBMS_FLASHBACK プロシージャの実行には不要です。
UPDATE ANY TABLE	SYS、AUDSYS を除く任意のスキーマ内での表およびビューの行の更新。
REDEFINE ANY TABLE	USER モードまたは FULL モードで権限を付与せずに、オンライン再定義を実行します。
表領域:	—
CREATE TABLESPACE	表領域の作成。
ALTER TABLESPACE	表領域の変更。
DROP TABLESPACE	表領域の削除。

システム権限名	許可されている操作
MANAGE TABLESPACE	表領域のオンラインとオフラインの切替え、および表領域のバックアップの開始と終了の制御。
UNLIMITED TABLESPACE	任意の表領域の無制限な使用。この権限は、設定されている任意の割当て制限を上書きします。ユーザーからこの権限を取り消した場合、ユーザーのスキーマ・オブジェクトはそのまま残りますが、表領域の割当て制限が許可されないかぎり、それ以上表領域を割り当てることはできません。このシステム権限をロールに付与することはできません。
トリガー:	—
CREATE TRIGGER	権限を付与したスキーマ内でのデータベース・トリガーの作成。
CREATE ANY TRIGGER	SYS、AUDSYS を除く任意のスキーマ内でのデータベース・トリガーの作成。
ALTER ANY TRIGGER	SYS、AUDSYS を除く任意のスキーマ内でのデータベース・トリガーの有効化、無効化またはコンパイル。
DROP ANY TRIGGER	SYS、AUDSYS を除く任意のスキーマ内でのデータベース・トリガーの削除。
ADMINISTER DATABASE TRIGGER	データベースに対するトリガーの作成。CREATE TRIGGER または CREATE ANY TRIGGER システム権限も必要です。
型:	—
CREATE TYPE	権限を付与したスキーマ内でのオブジェクト型およびオブジェクト型本体の作成。
CREATE ANY TYPE	SYS、AUDSYS を除く任意のスキーマ内でのオブジェクト型およびオブジェクト型本体の作成。
ALTER ANY TYPE	SYS、AUDSYS を除く任意のスキーマ内でのオブジェクト型の変更。
DROP ANY TYPE	SYS、AUDSYS を除く任意のスキーマ内でのオブジェクト型およびオブジェクト型本体の削除。

システム権限名	許可されている操作
EXECUTE ANY TYPE	特定のユーザーに付与した場合の、SYS、 AUDSYS を除く任意のスキーマ内でのオブジェクト型およびコレクション型の使用と参照、および SYS、 AUDSYS を除く任意のスキーマ内でのオブジェクト型メソッドの起動。EXECUTE ANY TYPE をロールに付与した場合、使用可能なロールを保持したユーザーは、任意のスキーマ内のオブジェクト型メソッドを起動できません。
UNDER ANY TYPE	非最終オブジェクト型のサブタイプの作成。
ユーザー:	—
CREATE USER	ユーザーの作成。この権限によって、次の操作を実行できます。 ● 任意の表領域に対する割当て制限の設定。 ● デフォルトの表領域および一時表領域の設定。 ● CREATE USER 文の一部としてのプロファイルの設定
ALTER USER	SYS 以外のユーザーの変更。この権限によって、次の操作を実行できます。 ● 他のユーザーのパスワードまたは認証方法の変更。 ● 任意の表領域に対する割当て制限の設定。 ● デフォルトの表領域および一時表領域の設定。 ● プロファイルおよびデフォルト・ロールの設定。
DROP USER	ユーザーの削除。
ビュー:	—
CREATE VIEW	権限を付与したスキーマ内でのビューの作成。
CREATE ANY VIEW	SYS、 AUDSYS を除く任意のスキーマ内でのビューの作成。
DROP ANY VIEW	SYS、 AUDSYS を除く任意のスキーマ内でのビューの削除。

システム権限名	許可されている操作
UNDER ANY VIEW	オブジェクト・ビューのサブビューの作成。
FLASHBACK ANY TABLE	SYS、AUDSYS を除く任意のスキーマ内での、任意の表、ビューまたはマテリアライズド・ビューでの SQL フラッシュバック問合せの発行。この権限は、DBMS_FLASHBACK プロシージャの実行には不要です。
MERGE ANY VIEW	MERGE ANY VIEW 権限が付与されている場合、そのユーザーが発行するすべての問合せにおいて、オプティマイザはビューのマージを使用して問合せのパフォーマンスを向上することができます。このとき、ビューのマージがビュー作成者のセキュリティ意図に違反しないかどうかは確認されません。OPTIMIZER_SECURE_VIEW_MERGING パラメータの詳細は、『Oracle Database リファレンス』を参照してください。ビューのマージの詳細は、『Oracle Database SQL チューニング・ガイド』を参照してください。
その他:	—
ANALYZE ANY	SYS を除く任意のスキーマ内の表、クラスタまたは索引の分析。
AUDIT ANY	SYS、AUDSYS を除く任意のスキーマ内での、AUDIT schema_objects 文を使用したオブジェクトの監査。
BECOME USER	データ・ポンプ・インポート・ユーティリティ(impdp)および元のインポート・ユーティリティ(imp)のユーザーは、第三者が直接実行できない操作(オブジェクト権限を付与するようなオブジェクトのロード)を実行する場合には、別のユーザー・アイデンティティを引き受けることができます。 Streams 管理者は、取得ユーザーおよび適用ユーザーの作成または変更を 1 つの Streams 環境で行うことができます。この権限は、デフォルトでは DBA ロールに含まれています。Database Vault では、この権限は DBA ロールから除外されます。したがって、Streams でこの権限が必要とされるのは、Database Vault がインストールされている環境でのみです。
CHANGE NOTIFICATION	問合せの登録の作成と、登録された問合せに関連付けられたオブジェクトに対する DML または DDL 変更があったときのデータベース変更通知の受信。データベース変更通知の詳細は、『Oracle Database 開発ガイド』を参照してください。

システム権限名	許可されている操作
COMMENT ANY TABLE	SYS、 AUDSYS を除く任意のスキーマ内での表、ビューまたは列についてのコメントの記述。
EXEMPT ACCESS POLICY	ファイングレイン・アクセス・コントロールの回避。 注意: 強力なシステム権限で、権限受領者がアプリケーション駆動のセキュリティ・ポリシーを回避できます。データベース管理者がこの権限を付与する場合は、注意が必要です。
FORCE ANY TRANSACTION	ローカル・データベース内の、インダウト分散トランザクションのコミットまたはロールバック。 分散トランザクション・エラーの意図的な発生。
FORCE TRANSACTION	ローカル・データベース内の、インダウト分散トランザクションのコミットまたはロールバック。
GRANT ANY OBJECT PRIVILEGE	オブジェクト所有者が付与を許されている任意のオブジェクト権限の付与。 GRANT ANY OBJECT PRIVILEGE 権限を持つオブジェクト所有者または他のユーザーによって付与されたオブジェクト権限の取消し
GRANT ANY PRIVILEGE	任意のシステム権限の付与。
INHERIT ANY PRIVILEGES	実行者の権限を持つ権限受領者が所有する実行者権限プロシージャの実行。
KEEP DATE TIME	権限受領者がアプリケーションを実行している場合、SYSDATE ファンクションおよび SYSTIMESTAMP ファンクションは、アプリケーション・コンティニュイティのための再生中に元の値を戻します。この権限は、リカバリ可能なエラー後にバインド変数の一貫性を保持するために役立ちます。 ノート: この権限が要求の実行時またはフェイルオーバーの間に付与または取消しされると、その要求に対するアプリケーション・コンティニュイティのための再実行中に元の値が戻されなくなります。
KEEP SYSGUID	権限受領者がアプリケーションを実行している場合、SYS_GUID ファンクションは、アプリケーション・コンティニュイティのための再実行中に元の値

システム権限名	許可されている操作
	を戻します。この権限は、リカバリ可能なエラー後にバインド変数の一貫性を保持するために役立ちます。 ノート: この権限が要求の実行時またはフェイルオーバーの間に付与または取消しされると、その要求に対するアプリケーション・コンティニュイティのための再実行中に元の値が戻されなくなります。
PURGE DBA_RECYCLEBIN	システム全体のごみ箱からのすべてのオブジェクトの削除。
RESUMABLE	再開可能な領域割当ての使用可能化。
SELECT ANY DICTIONARY	SYS 内の任意のデータ・ディクショナリ・オブジェクトの問合せ。ただし、DEFAULT_PWD$、ENC$、LINK$、USER$、USER_HISTORY$、および XS$VERIFIERS オブジェクトを除きます。 初期化パラメータ O7_DICTIONARY_ACCESSIBILITY のデフォルトの FALSE 設定を選択的に上書きします。
SELECT ANY TRANSACTION	FLASHBACK_TRANSACTION_QUERY ビューの内容の問合せ 注意: 強力なシステム権限で、権限受領者が、過去のデータも含めてデータベース内のすべてのデータを参照できます。この権限は、Oracle フラッシュバック・トランザクション問合せ機能を使用する必要があるユーザーのみに付与してください。
SYSBACKUP	次のバックアップ操作およびリカバリ操作の実行 STARTUP および SHUTDOWN。 CREATE CONTROLFILE。 CREATE PFILE および CREATE SPFILE FLASHBACK DATABASE。 リストア・ポイント(保証付きリストア・ポイントを含む)の作成、使用、表示、および削除。 DBMS_DATAPUMP、DBMS_PIPE、DBMS_TDB、および DBMS_TTS パッケージ内のプロシージャの実行 X$の表、V$のビュー、および GV$のビューに対する SELECT

システム権限名	許可されている操作
	ALTER DATABASE、ALTER SESSION、ALTER SYSTEM、ALTER TABLESPACE、CREATE ANY CLUSTER、CREATE ANY DIRECTORY、CREATE ANY TABLE、CREATE SESSION、DROP DATABASE、DROP TABLESPACE、RESUMABLE、SELECT ANY DICTIONARY、SELECT ANY TRANSACTION、UNLIMITED TABLESPACE 権限および SELECT_CATALOG_ROLE ロールを含みます。
SYSDBA	STARTUP および SHUTDOWN。 ALTER DATABASE(オープン、マウント、バックアップまたは文字セットの変更) CREATE DATABASE。 DROP DATABASE。 ARCHIVELOG および RECOVERY。 CREATE SPFILE。 RESTRICTED SESSION 権限を含みます。
SYSDG	次の Oracle Data Guard 操作の実行 STARTUP および SHUTDOWN。 FLASHBACK DATABASE。 リストア・ポイント(保証付きリストア・ポイントを含む)の作成、使用、表示、および削除。 X$の表、V$のビュー、および GV$のビューに対する SELECT ALTER DATABASE、ALTER SESSION、ALTER SYSTEM、CREATE SESSION、および SELECT ANY DICTIONARY 権限を含みます。
SYSKM	次の暗号化キー管理操作の実行 データベースへの接続(データベースがオープンされていない場合でも)。

システム権限名	許可されている操作
	データベースがオープンされている場合は、V$CLIENT_SECRETS、V$ENCRYPTED_TABLESPACES、V$ENCRYPTION_KEYS、V$ENCRYPTION_WALLET および V$WALLET ビューに対する SELECT。 ADMINISTER KEY MANAGEMENT および CREATE SESSION 権限を含みます。
SYSOPER	STARTUP 操作および SHUTDOWN 操作 ALTER DATABASE(オープン、マウントまたはバックアップ) ARCHIVELOG および RECOVERY。 CREATE SPFILE。 RESTRICTED SESSION 権限を含みます。

表18-2 オブジェクト権限(操作対象のデータベース・オブジェクト別)

オブジェクト権限名	許可されている操作
分析ビュー権限	次の分析ビュー権限は、分析ビューの操作を許可します。
ALTER	分析ビューの名前の変更。
READ	SELECT 文でのオブジェクトの問合せ。
SELECT	SELECT 文でのオブジェクトの問合せ。
属性ディメンション権限	次の属性ディメンション権限は、属性ディメンションの操作を許可します。
ALTER	属性ディメンションの名前の変更。
ディレクトリ権限	次のディレクトリ権限では、ディレクトリ・オブジェクトをポインタとして使用することにより、オペレーティング・システムのディレクトリに格納されている各ファイルにデータベースから安全にアクセスできるようになります。このディレクトリ・オブジェクトには、ファイルが格納されているオペレーティング・システムのディレクトリへのフルパス名が定義されています。これらのファイルは実際にはデータベース外に格納されているため、Oracle Database サーバーの各プロセスは、ファイル・システム・サーバーに対して適切なファイル・アクセス権も持っている必要があります。オペレーティング・システムに対するオブジェ

オブジェクト権限名	許可されている操作
	クト権限ではなく、ディレクトリ・データベース・オブジェクトに対するオブジェクト権限を個々のデータベース・ユーザーに付与することによって、データベースでファイル運用時のセキュリティが確保されます。
READ	ディレクトリ内のファイルの読取り。
WRITE	ディレクトリ内へのファイルの書込み。この権限は、外部表に接続する場合のみに役立ちます。これによって、権限受領者は、外部表のエージェントがディレクトリに書き込めるのがログ・ファイルなのか不良ファイルなのかを判断できます。 制限事項: この権限を持っていても、BFILE への書込みを行うことはできません。
EXECUTE	ディレクトリ内に存在するプリプロセッサ・プログラムの実行。プリプロセッサ・プログラムは、ORACLE_LOADER アクセス・ドライバによって外部表からデータ・レコードをロードするときに、サポートされている形式にデータを変換します。詳細は、『Oracle Database ユーティリティ』を参照してください。この権限は、外部表データへの READ アクセスを暗黙的に許可しません。
エディション権限	次のエディション権限は、エディションの使用を許可します。
USE	エディションの使用。
フラッシュバック・データ・アーカイブ権限	次のフラッシュバック・データ・アーカイブ権限では、フラッシュバック・データ・アーカイブに対する操作を許可します。
FLASHBACK ARCHIVE	表の履歴追跡を使用可能または使用禁止にします。
階層権限	次の階層権限は、階層の操作を許可します。
ALTER	階層の名前の変更。
READ	SELECT 文でのオブジェクトの問合せ。
SELECT	SELECT 文でのオブジェクトの問合せ。
索引タイプ権限	次の索引タイプ権限は、索引タイプの操作を許可します。
EXECUTE	索引タイプの参照。

オブジェクト権限名	許可されている操作
ライブラリ権限	次のライブラリ権限は、ライブラリの操作を許可します。
EXECUTE	特定のオブジェクトの使用と参照、およびそのメソッドの起動。 注意: この権限は非常に強力であるため、信頼できるユーザーにのみ付与してください。この権限を付与する前に、『Oracle Database セキュリティ・ガイド』を参照してください。
マテリアライズド・ビュー権限	次のマテリアライズド・ビュー権限は、マテリアライズド・ビューについての操作を許可します。DELETE、INSERT および UPDATE 権限は、更新可能なマテリアライズド・ビューにのみ付与できます。
ON COMMIT REFRESH	指定した表に対する REFRESH ON COMMIT モードのマテリアライズド・ビューの作成。
QUERY REWRITE	指定した表で使用するクエリー・リライトに対するマテリアライズド・ビューの作成。
READ	マテリアライズド・ビューを問い合せます。
SELECT	マテリアライズド・ビューを問い合せます。SELECT ... FOR UPDATE または LOCK TABLE 文を使用して、行ロックを取得します。
マイニング・モデル権限	次のマイニング・モデル権限では、マイニング・モデルに対する操作を許可します。これらの権限は、ユーザー自身のスキーマ内にあるモデルに対しては必要ありません。
ALTER	該当する DBMS_DATA_MINING プロシージャを使用して、マイニング・モデル名または関連のコスト・マトリックスを変更します。
SELECT	マイニング・モデルのスコアリングまたは表示。スコアリングの実行には、SQL ファンクションの PREDICTION ファミリ、または DBMS_DATA_MINING.APPLY プロシージャを使用します。モデルの表示は、DBMS_DATA_MINING.GET_MODEL_DETAILS_*プロシージャで実行されます。
オブジェクト型権限	次のオブジェクト型権限は、データベース・オブジェクト型の操作を許可します。
DEBUG	オブジェクト型に定義されたすべてのパブリック変数、非パブリック変数、メソッドおよび型へのデバッガを介したアクセス。

オブジェクト権限名	許可されている操作
	型本体内の行または指示境界へのブレークポイントの設定、またはこれらの場所のいずれかでの停止。
EXECUTE	特定のオブジェクトの使用と参照、およびそのメソッドの起動。 オブジェクト型に定義されたパブリック変数、型およびメソッドへのデバッガを介したアクセス。
UNDER	型のサブタイプの作成。この型の直属のスーパータイプに WITH GRANT OPTION 付きの UNDER ANY TYPE 権限を持つ場合のみ、このオブジェクト権限を付与できます。
OLAP 権限	Oracle Database を OLAP オプションで使用している場合は、次のオブジェクト権限が有効になります。
INSERT	メンバーを OLAP キューブ・ディメンションに挿入するか、またはメジャーをメジャー・フォルダに挿入します。
ALTER	OLAP キューブ・ディメンションまたはキューブの定義を変更します。
DELETE	メンバーを OLAP キューブ・ディメンションから削除するか、またはメジャーをメジャー・フォルダから削除します。
SELECT	OLAP キューブまたはキューブ・ディメンションを表示または問い合せます。
UPDATE	OLAP キューブのメジャー値またはキューブ・ディメンションの属性値を更新します。
演算子権限	次の演算子権限は、ユーザー定義演算子の操作を許可します。
EXECUTE	演算子の参照。
プロシージャ、ファンクション、パッケージ権限	次のプロシージャ、ファンクションおよびパッケージ権限は、プロシージャ、ファンクションおよびパッケージの操作を許可します。これらの権限は、Java ソース、クラスおよびリソースにも適用されます。Oracle Database では、これらはオブジェクト権限の付与のために生成されたプロシージャとして扱われます。
DEBUG	オブジェクトに定義されたすべてのパブリック変数、非パブリック変数、メソッドおよび型へのデバッガを介したアクセス。

オブジェクト権限名	許可されている操作
	プロシージャ、ファンクションまたはパッケージ内の行または指示境界へのブレークポイントの設定、またはこれらの場所のいずれかでの停止。この権限は、メソッドまたはパッケージの仕様部および本体の宣言にアクセスする権限を付与します。
EXECUTE	プロシージャかファンクションの直接実行、パッケージの仕様部に宣言された任意のプログラム・オブジェクトへのアクセス、または現在無効であるかコンパイルされていないファンクションかプロシージャへのコール中の暗黙的なオブジェクトのコンパイル。この権限を持っていても、ALTER PROCEDURE または ALTER FUNCTION を使用して明示的なコンパイルを実行することはできません。明示的なコンパイルを実行するには、適切な ALTER SYSTEM 権限が必要です。 プロシージャ、ファンクションまたはパッケージに定義されたパブリック変数、型およびメソッドへのデバッガを介したアクセス。この権限は、メソッドまたはパッケージの仕様部のみの宣言にアクセスする権限を付与します。 ジョブ・スケジューラ・オブジェクトは、DBMS_SCHEDULER パッケージを使用して作成します。作成したオブジェクトに、ジョブ・スケジューラのクラスおよびプログラムに対する EXECUTE オブジェクト権限を付与できます。ジョブ・スケジューラのジョブ、プログラムおよびスケジュールに対しても ALTER 権限を付与できます。 ノート: プロシージャ、ファンクションまたはパッケージを間接的に実行する場合、ユーザーはこの権限を持つ必要はありません。
スケジューラ権限	ジョブ・スケジューラ・オブジェクトは、DBMS_SCHEDULER パッケージを使用して作成します。これらのオブジェクトを作成した後で、次の権限を付与できます。
EXECUTE	ジョブ・クラス、プログラム、チェーンおよび資格証明に対する操作。
ALTER	ジョブ、プログラム、チェーン、資格証明およびスケジュールに対する変更。
USE	指定したスケジューラ・リソース・オブジェクトの、プログラムおよびジョブとの関連付け。
順序権限	次の順序権限は、順序の操作を許可します。
ALTER	ALTER SEQUENCE 文での順序定義の変更。
KEEP SEQUENCE	権限受領者がアプリケーションを実行している場合、順序疑似列 NEXTVAL は、アプリケーション・コンティニュイティのための再実行中に元の値を保持します。この権限は、リカバリ可能なエラーの後で再実行するときに、バインド変数の一貫性を確保するために役立ちます。

オブジェクト権限名	許可されている操作
	この権限が要求の実行時およびフェイルオーバーの間に付与または取消しされると、その要求に対するアプリケーション・コンティニュイティのための再実行中に、NEXTVAL の元の値は保持されなくなります。 ノート: この権限は、GRANT ALL PRIVILEGES ON sequence 文では付与されません。この権限を明示的に付与する必要があります。 ノート: この権限は DBA ロールの一部です。
SELECT	CURRVAL 疑似列および NEXTVAL 疑似列を使用した順序の値の検査および増分。
SQL 翻訳プロファイル権限	次の SQL 翻訳プロファイルの権限は、SQL 翻訳プロファイルに対する操作を許可します。
ALTER	SQL 翻訳プロファイルのトランスレータ、カスタム SQL 文の翻訳、またはカスタム・エラーの翻訳の変更。
USE	SQL 翻訳プロファイルの使用。
シノニム権限	シノニム権限は、対象となるオブジェクトに対して付与される権限と同じです。シノニムに権限を付与することは、基本オブジェクトに権限を付与することと同じです。同様に、基本オブジェクトに対して権限を付与することは、オブジェクトのすべてのシノニムに権限を付与することと同じです。あるユーザーにシノニムの権限を付与した場合、そのユーザーは、シノニム名または基本オブジェクト名を SQL 文に指定して、その権限を使用できます。
表権限	次の表権限は、表の操作を許可します。READ 権限を除く次のいずれかのオブジェクト権限を持っている場合は、LOCK TABLE 文を使用して任意のロック・モードで表をロックできます。 ノート: 外部表に有効なオブジェクト権限は、ALTER、READ および SELECT のみです。
ALTER	ALTER TABLE 文での表定義の変更。
DEBUG	デバッガを介した次のものへのアクセス。 ● 表に定義されているトリガーの本体の PL/SQL コード

オブジェクト権限名	許可されている操作
	● 表を直接参照する SQL 文に関する情報
DELETE	DELETE 文での表の行の削除。 ノート: 表がリモート・データベースにある場合は、DELETE 権限とともに表に対する SELECT 権限を付与する必要があります。
INDEX	CREATE INDEX 文での表の索引の作成。
INSERT	INSERT 文での表への新しい行の追加。 ノート: 表がリモート・データベースにある場合は、INSERT 権限とともに表に対する SELECT 権限を付与する必要があります。
READ	SELECT 文での表の問合せ。SELECT ... FOR UPDATE を許可しません。
REFERENCES	表参照制約の作成。この権限はロールには付与できません。
SELECT	SELECT ... FOR UPDATE を含む SELECT 文を使用して、表を問い合せます。
UPDATE	UPDATE 文での表のデータの変更。 ノート: 表がリモート・データベースにある場合は、UPDATE 権限とともに表に対する SELECT 権限を付与する必要があります。
ユーザー権限	次の権限はユーザーの操作を許可します。
INHERIT PRIVILEGES	実行者がこの権限を付与されたユーザーの場合、実行者の権限を持つ権限受領者が所有する実行者権限プロシージャまたはファンクションの実行。
INHERIT REMOTE PRIVILEGES	この権限が付与されたユーザーが、権限受領者が所有する定義者権限プロシージャまたはファンクション(現在のユーザー・データベース・リンクを含む)を実行することを許可します。
TRANSLATE SQL	権限受領者の SQL 翻訳プロファイルを使用して、この権限が付与されたユーザー用に SQL を変換。

オブジェクト権限名	許可されている操作
ビュー権限	次のビュー権限は、ビューの操作を許可します。READ 権限を除く次のいずれかのオブジェクト権限を持っている場合は、LOCK TABLE 文を使用して任意のロック・モードでビューをロックできます。 ビューの権限を付与する場合、そのビューのすべての実表に関して GRANT OPTION 付きの権限が必要です。
DEBUG	デバッガを介した次のものへのアクセス。 ● ビューに定義されているトリガーの本体の PL/SQL コード ● ビューを直接参照する SQL 文に関する情報
DELETE	DELETE 文でのビューの行の削除。
INSERT	INSERT 文でのビューへの新しい行の追加。
MERGE VIEW	このオブジェクト権限の動作は、権限が ON 句で指定されたビューに制限される点を除き、MERGE ANY VIEW システム権限の動作と同じです。指定されたビューに対して権限受領者が発行するすべての問合せに対して、オプティマイザはビューのマージを使用して問合せのパフォーマンスを向上させることができます。このとき、ビューのマージがビュー作成者のセキュリティ意図に違反しないかどうかは確認されません。
READ	SELECT 文でのビューの問合せ。SELECT ... FOR UPDATE を許可しません。
REFERENCES	ビューへの外部キー制約の定義。
SELECT	SELECT ... FOR UPDATE を含む SELECT 文を使用して、ビューを問い合せます。 関連項目: ビューに対するこのオブジェクト権限の付与の詳細は、「object_privilege」を参照してください。
UNDER	ビューのサブビューの作成。このビューの直属のスーパービューに WITH GRANT OPTION 付きの UNDER ANY VIEW 権限を持つ場合のみ、このオブジェクト権限を付与できます。
UPDATE	UPDATE 文でのビューのデータの変更。

例

ユーザーへのシステム権限の付与: 例

次の文は、サンプル・ユーザーhrにCREATE SESSIONシステム権限を付与し、hrがOracle Databaseにログインできるようにします。

```
GRANT CREATE SESSION
   TO hr;
```

システム権限の付与時のユーザー・パスワードの割当て: 例

ユーザーhrが存在し、ユーザーnewuserが存在しないとします。次の文は、ユーザーhrのパスワードをpassword1にリセットし、ユーザーnewuserをpassword2とともに作成し、両ユーザーにCREATE SESSIONシステム権限を付与します。

```
GRANT CREATE SESSION
  TO hr, newuser IDENTIFIED BY password1, password2;
```

ロールへのシステム権限の付与: 例

次の文は、データ・ウェアハウス管理者ロール(「ロールの作成: 例」で作成)に、適切なシステム権限を付与します。

```
GRANT
     CREATE ANY MATERIALIZED VIEW
   , ALTER ANY MATERIALIZED VIEW
   , DROP ANY MATERIALIZED VIEW
   , QUERY REWRITE
   , GLOBAL QUERY REWRITE
   TO dw_manager
   WITH ADMIN OPTION;
```

dw_managerの権限ドメインには、マテリアライズド・ビューに関連するシステム権限が含まれます。

ADMIN OPTION付きロールの付与: 例

次の文は、ADMIN OPTION付きのdw_managerロールをサンプル・ユーザーshに付与します。

```
GRANT dw_manager
   TO sh
   WITH ADMIN OPTION;
```

dw_managerロールによって、shは次の操作を実行できます。

- ロールを使用可能にして、CREATE MATERIALIZED VIEWシステム権限を含むそのロールの権限ドメインに登録されている権限の使用
- 他のユーザーに対するそのロールの付与および取消し
- ロールの削除
- shスキーマ内のプログラム・ユニットのdw_managerロールを付与および取り消します。

DELEGATE OPTION付きロールの付与: 例

次の文は、DELEGATE OPTION付きのdw_managerロールをサンプル・ユーザーshに付与します。

```
GRANT dw_manager
   TO sh
   WITH DELEGATE OPTION;
```

ユーザーshは、shスキーマ内のプログラム・ユニットのdw_managerロールを付与および取り消すことができます。

ロールへのオブジェクト権限の付与: 例

次の例では、データ・ウェアハウス・ユーザー・ロール(「ロールの作成: 例」で作成)に、SELECTオブジェクト権限を付与します。

```
GRANT SELECT ON sh.sales TO warehouse_user;
```

ロールへのロールの付与: 例

次の文は、warehouse_userロールをdw_managerロールに付与します。どちらのロールも、「ロールの作成: 例」で作成されたものです。

```
GRANT warehouse_user TO dw_manager;
```

dw_managerロールには、warehouse_userロールのドメインにあるすべての権限が含まれます。

ユーザーに対するオブジェクト権限の付与: 例

次の文は、ユーザーshに対するINHERIT PRIVILEGESオブジェクト権限をユーザーhrに付与します。

```
GRANT INHERIT PRIVILEGES ON USER sh TO hr;
```

ディレクトリに対するオブジェクト権限の付与: 例

次の文は、ユーザーhrにディレクトリbfile_dirに対するREAD権限をGRANT OPTION付きで付与します。

```
GRANT READ ON DIRECTORY bfile_dir TO hr
   WITH GRANT OPTION;
```

表に対するオブジェクト権限のユーザーへの付与: 例

次の文は、GRANT OPTIONを使用して表oe.bonuses (「表へのマージ: 例」で作成)に対するすべての権限をユーザーhrに付与します。

```
GRANT ALL ON bonuses TO hr
   WITH GRANT OPTION;
```

この結果、hrユーザーは次の操作を実行できます。

- bonuses表に対するすべての権限の使用
- 他のユーザーまたはロールに対する、bonuses表についての権限の付与

ビューに対するオブジェクト権限の付与: 例

次の文は、ビューemp_view (「ビューの作成: 例」で作成)に対するSELECTとUPDATEの権限をすべてのユーザーに付与します。

```
GRANT SELECT, UPDATE
   ON emp_view TO PUBLIC;
```

この結果、すべてのユーザーが、従業員の詳細についてのビューを問合せおよび更新できるようになります。

別のスキーマの順序へのオブジェクト権限の付与: 例

次の文は、スキーマoe内のcustomers_seq順序に対するSELECT権限をユーザーhrに付与します。

```
GRANT SELECT
   ON oe.customers_seq TO hr;
```

ユーザーhrは、次の文を指定して、順序の次の値を作成できるようになります。

```
SELECT oe.customers_seq.NEXTVAL
   FROM DUAL;
```

別々の列に対する複数のオブジェクト権限の付与: 例

次の文は、ユーザーoeに、スキーマhrのemployees表のemployee_id列に対するREFERENCES権限と、employee_id列、salary列およびcommission_pct列に対するUPDATE権限を付与します。

```
GRANT REFERENCES (employee_id),
      UPDATE (employee_id, salary, commission_pct)
   ON hr.employees
   TO oe;
```

この結果、ユーザーoeは、employee_id列、salary列およびcommission_pct列の値を更新できるようになります。ユーザーoeは、employee_id列を参照する参照整合性制約を定義することもできます。ただし、GRANT文にはこれらの列のみが指定されているため、ユーザーoeはemployees表の他の列を操作できません。

たとえば、oeは制約付きの表を作成できます。

```
CREATE TABLE dependent
   (dependno   NUMBER,
    dependname VARCHAR2(10),
    employee   NUMBER
   CONSTRAINT in_emp REFERENCES hr.employees(employee_id) );
```

スキーマhr内のemployees表の従業員に対応するdependent表の依存性が、制約in_empによって保証されます。

INSERT

目的

INSERT文を使用すると、表、ビューの実表、パーティション表のパーティション、コンポジット・パーティション表のサブパーティション、オブジェクト表またはオブジェクト・ビューの実表に、行を追加できます。

前提条件

表に行を挿入する場合は、その表が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、その表に対するINSERTオブジェクト権限が必要です。

ビューの実表に行を挿入する場合、ビューが定義されているスキーマの所有者には、その実表に対するINSERTオブジェクト権限が必要です。また、他のユーザーのスキーマ内のビューに行を挿入する場合は、そのビューに対するINSERTオブジェクト権限が必要です。

INSERT ANY TABLEシステム権限があれば、任意の表または任意のビューの実表に行を挿入できます。

表がリモート・データベースにある場合は、行を挿入する表に対するREADまたはSELECTオブジェクト権限も必要です。

従来型INSERTおよびダイレクト・パスINSERT

表、パーティションまたはビューにデータを挿入するために使用するINSERT文には、従来型INSERTおよびダイレクト・パスINSERTの2種類があります。従来型INSERT文を発行すると、表の空き領域を再利用して挿入され、参照整合性制約が維持されます。ダイレクト・パス・インサートの場合、表の既存データの後に、挿入したデータが追加されます。データは、バッファ・キャッシュを回避してデータファイルに直接書き込まれます。既存データの空き領域は再利用されません。この代替手法を使用すると、挿入操作中のパフォーマンスが向上します。また、これは、Oracleのダイレクト・パス・ローダー・ユーティリティであるSQL*Loaderの機能に似ています。パラレル・モードで作成された表に挿入する場合は、ダイレクト・パス・インサートがデフォルトです。

データベースでのREDOデータおよびUNDOデータの生成方法は、従来型INSERTとダイレクト・パス・インサートのいずれを使用しているかに一部関係しています。

- 従来型INSERTでは、表とアーカイブ・ログのロギング設定に関係なく、データとメタデータの両方に対する変更に対して常に最大のREDOおよびUNDOが生成され、データベースのロギング設定が強制的に使用されます。
- ダイレクト・パス・インサートでは、メタデータの変更に対してREDOとUNDOの両方が生成されます。これらは操作のリカバリに必要であるためです。データの変更に大してはUNDOとREDOが次のように生成されます。
    - ダイレクト・パス・インサートでは、データの変更に対するUNDOの生成が常に回避されます。
    - データベースがARCHIVELOGまたはFORCE LOGGINGモードでない場合は、表のロギング設定に関係なく、データの変更に対するREDOは生成されません。
    - データベースがARCHIVELOGモード(ただしFORCE LOGGINGモードではない)の場合、ダイレクト・パスINSERTでは、LOGGING表のデータのREDOは生成されますが、NOLOGGING表のデータのREDOは生成されません。
    - データベースがARCHIVELOGモードかつFORCE LOGGINGモードの場合、ダイレクト・パスSQLでは、LOGGING表とNOLOGGING表の両方のデータのREDOが生成されます。

ダイレクト・パス・インサートには、次の制限があります。これらの制限に違反する場合、他にエラーがないかぎりメッセージが戻されず、従来型INSERTがシリアルで実行されます。

- DML文の有無にかかわらず、1つのトランザクションで複数のダイレクト・パス・インサート文を使用できます。ただし、あるDML文が特定の表、パーティションまたは索引を変更した後は、トランザクションの他のDML文は表、パーティションまたは索引にアクセスできません。
- 同じ表、パーティションまたは索引にアクセスする問合せは、ダイレクト・パス・インサート文の前であれば許可されますが、後は許可されません。
- シリアルまたはパラレル文が、同じトランザクションからダイレクト・パス・インサートによって変更された表にアクセスしようとする場合、エラーが戻され、文が拒否されます。
- 対象となる表は、クラスタにはできません。
- 対象となる表は、オブジェクト型列を含むことはできません。
- 索引構成表(IOT)がマッピング表を含んでいる場合、またはマテリアライズド・ビューによって参照されている場合、その索引構成表に対してダイレクト・パス・インサートを使用することはできません。
- 索引構成表(IOT)の1つのパーティション、1つのパーティションのみを含むパーティション化されたIOT、またはパーティション化されていないIOTへのダイレクト・パスINSERTは、IOTがパラレル・モードで作成されている場合またはAPPENDヒントやAPPEND_VALUESヒントを指定した場合でも、シリアルで実行されます。ただし、パーティション化されたIOTへのダイレクト・パス・インサート操作は、拡張パーティション名が使用されず、IOTに複数のパーティションが含まれているかぎり、パラレル・モードで実行されます。
- ターゲット表には、トリガーおよび参照整合性制約を定義できない。
- 対象となる表は、レプリケートできません。
- ダイレクト・パス・インサート文を含むトランザクションは、分散できません。

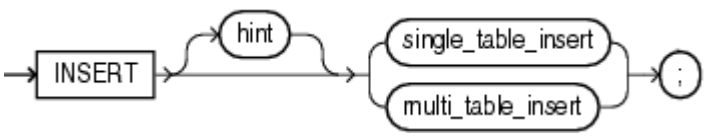
ダイレクト・パス・インサートによって挿入したデータを、挿入した直後に問合せまたは変更することはできません。問合せまたは変更しようとすると、ORA-12838エラーが発生します。新しく挿入したデータの読取りまたは変更を試みる前に、COMMIT文を発行する必要があります。

関連項目:

- ダイレクト・パス・インサートの詳細は、『Oracle Database管理者ガイド』を参照してください。
- SQL*Loaderの詳細は、『Oracle Databaseユーティリティ』を参照してください。
- ダイレクト・パスINSERTを使用して空の表への挿入を行うときの統計収集の詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください。

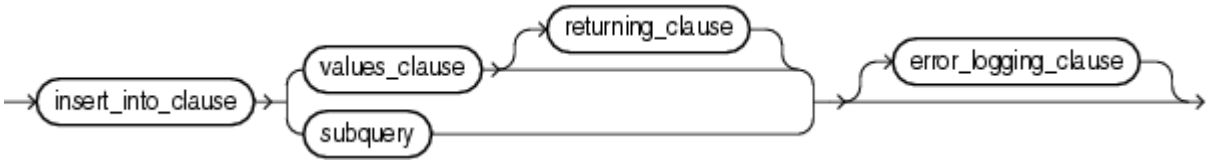
構文

insert::=



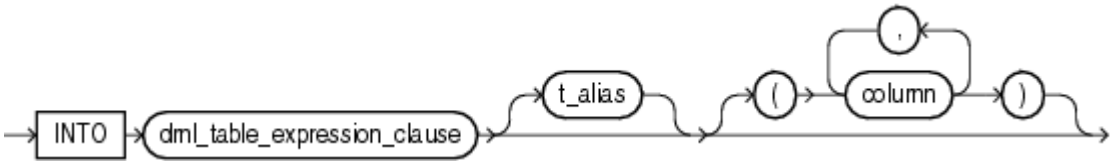
(single_table_insert::=、multi_table_insert::=)

single_table_insert::=



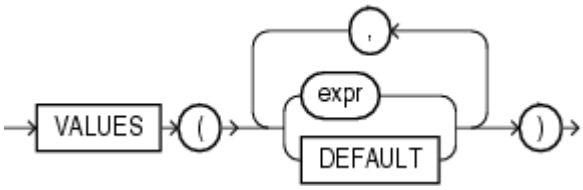
(insert_into_clause::=、values_clause::=、returning_clause::=、subquery::=、error_logging_clause::=)

insert_into_clause::=

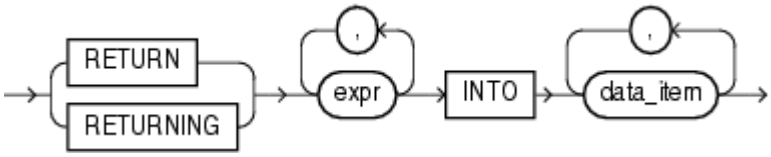


(DML_table_expression_clause::=)

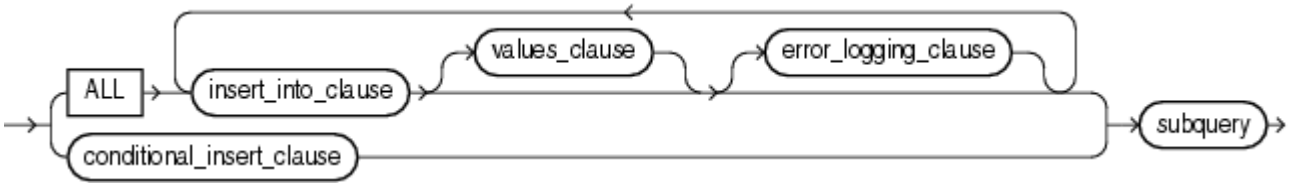
values_clause::=



returning_clause::=

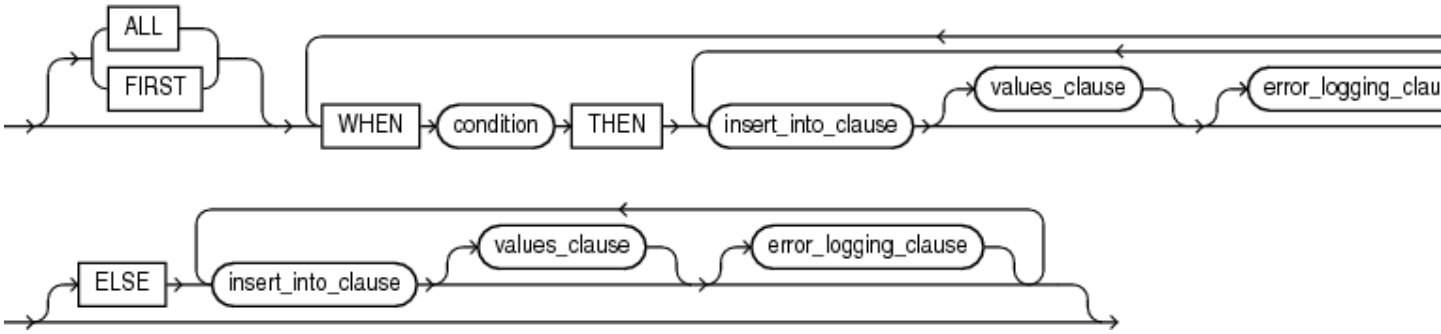


multi_table_insert::=



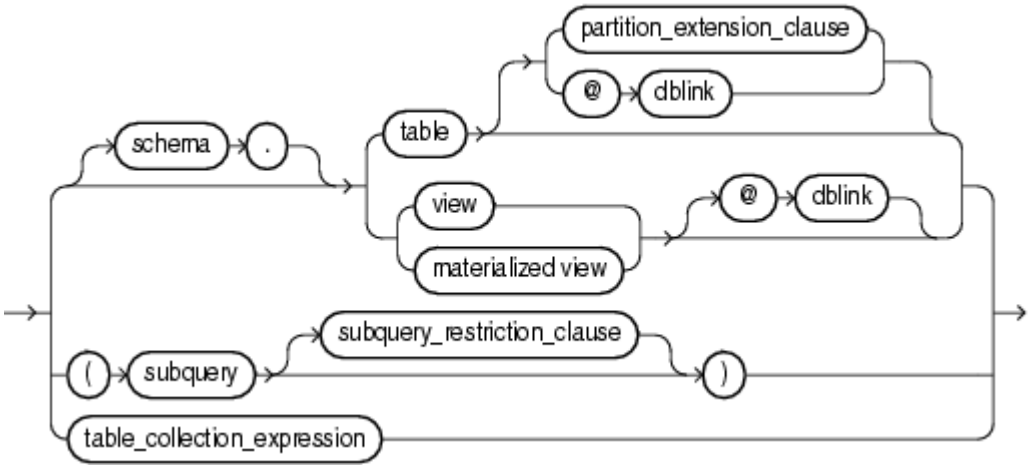
(insert_into_clause::=、values_clause::=、conditional_insert_clause::=、subquery::=、error_logging_clause::=)

conditional_insert_clause::=



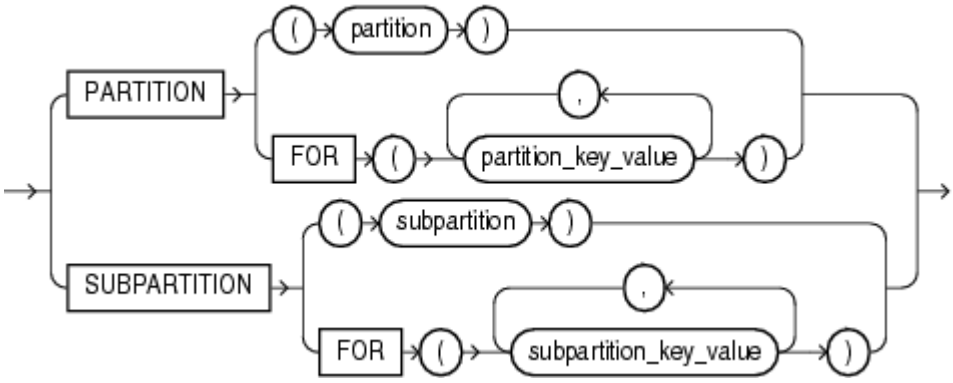
(insert_into_clause::=、values_clause::=)

DML_table_expression_clause::=を参照

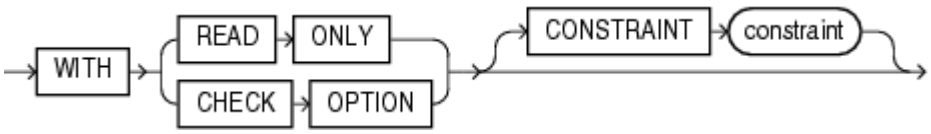


(partition_extension_clause::=、subquery::=(SELECTの一部)、subquery_restriction_clause::=、table_collection_expression::=)

partition_extension_clause::=

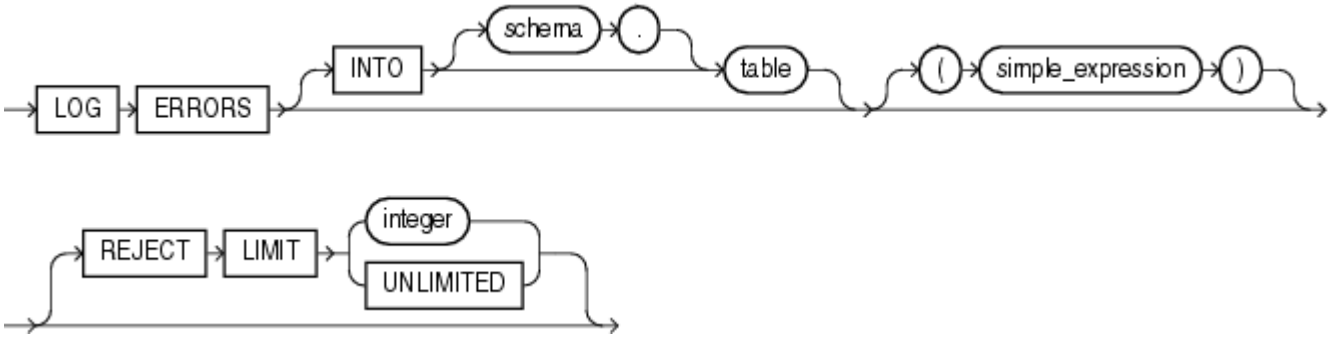


subquery_restriction_clause::=



table_collection_expression::=

error_logging_clause::=



セマンティクス

hint

文の実行計画を選択する場合に、オプティマイザに指示を与えるためのコメントを指定します。

マルチテーブル・インサートの場合、対象となる表に対してPARALLELヒントを指定すると、表がPARALLELの指定付きで作成または変更されていなくても、マルチテーブル・インサート文全体がパラレル化されます。PARALLELヒントを指定しない場合、対象となるすべての表がPARALLELの指定付きで作成または変更されていないかぎり、挿入操作はパラレル化されません。

関連項目:

- ヒントの構文および説明は、「ヒント」を参照してください。
- 「マルチテーブル・インサートの制限事項」

single_table_insert

シングルテーブル・インサートの場合、明示的に値を指定する、または副問合せで値を検索することによって、表、ビューまたはマテリアライズド・ビューの1行に値を挿入します。

subqueryでflashback_query_clauseを使用すると、過去のデータをtableに挿入できます。この句の詳細は、「SELECT」の「flashback_query_clause」 を参照してください。

シングルテーブル・インサートの制限事項

副問合せで値を検索する場合、副問合せのSELECT構文のリストにはINSERT文の列リストと同じ数の列が含まれている必要があります。列リストを指定しない場合は、副問合せで表の各列の値を指定する必要があります。

関連項目:

表への値の挿入：例

insert_into_clause

INSERT INTO句を使用すると、データを挿入する対象となる表またはオブジェクトを指定できます。

DML_table_expression_clause

INTO DML_table_expression_clauseを使用すると、データを挿入するオブジェクトを指定できます。

schema

表、ビューまたはマテリアライズド・ビューが含まれているスキーマを指定します。schemaを指定しない場合、オブジェクトは自分のスキーマ内にあるとみなされます。

table | view | materialized_view | subquery

行を挿入する表またはオブジェクト表の名前、ビューまたはオブジェクト・ビューの名前、マテリアライズド・ビューの名前、あるいは副問合せから戻された列の名前を指定します。ビューまたはオブジェクト・ビューを指定した場合、そのビューの実表に行が挿入されます。

読取り専用マテリアライズド・ビューには行を挿入できません。書込み可能なマテリアライズド・ビューに行を挿入した場合、基礎となるコンテナ表にその行が挿入されます。ただし、その挿入操作は次のリフレッシュ操作によって上書きされます。マテリアライズド・ビュー・グループ内の更新可能なマテリアライズド・ビューに行を挿入した場合、対応する行もマスター表に挿入されます。

挿入される値がオブジェクト表に対するREFの場合、およびそのオブジェクト表に主キー・オブジェクト識別子がある場合、REFを挿入する列は、オブジェクト表に対する参照整合性制約またはSCOPE制約を持つREF列である必要があります。

tableまたはviewの実表に、1列以上のドメイン索引がある場合は、この文が適切な索引タイプの挿入ルーチンを実行します。

表に対してINSERT文を発行した場合、その表に対して定義されているINSERTトリガーが起動します。

関連項目:

これらのルーチンの詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。

DML_table_expression_clauseの制限事項

この句には、次の制限事項があります。

- tableまたはviewの実表に、IN_PROGRESSまたはFAILEDとマークされたドメイン索引がある場合は、この文は実行できません。
- 関係する索引パーティションがUNUSABLEとマークされている場合は、パーティションに挿入できません。
- DML_table_expression_clauseのsubqueryのORDER BY句に関して、順序付けは、挿入された行または表の各エクステント内のみに保証されています。既存の行に関連する新しい行の順序付けは保証されていません。
- WITH CHECK OPTIONを指定してビューを作成した場合、ビューを定義する問合せを満たす行のみビューに挿入されます。
- 単一の実表を使用してビューを作成した場合、行をビューに挿入し、returning_clauseを使用してその行の値を取り出せます。
- ビューを定義する問合せに次のいずれかの構造体が含まれる場合は、INSTEAD OFトリガーを使用する場合を除いて、そのビューに行を挿入できません。
    - 集合演算子
    - DISTINCT演算子
    - 集計ファンクションまたは分析ファンクション
    - GROUP BY、ORDER BY、MODEL、CONNECT BYまたはSTART WITH句

- SELECT構文のリストにあるコレクション式
- SELECT構文のリストにある副問合せ
- WITH READ ONLYが指定された副問合せ
- 結合(一部の例外を除く。詳細は、『Oracle Database管理者ガイド』を参照してください。)
-

- UNUSABLEのマークが付いている索引、索引パーティションまたは索引サブパーティションを指定する場合、SKIP_UNUSABLE_INDEXESセッション・パラメータがTRUEに設定されていないかぎり、INSERT文は正常に実行されません。SKIP_UNUSABLE_INDEXESセッション・パラメータの詳細は、「ALTER SESSION」を参照してください。

partition_extension_clause

挿入対象のtableまたはviewの実表内にあるパーティションまたはサブパーティションの名前またはパーティション・キー値を指定します。

挿入する行が特定のパーティションまたはサブパーティションにマップされない場合、エラーが戻されます。

ターゲットのパーティションおよびサブパーティションの制限事項

この句は、オブジェクト表またはオブジェクト・ビューでは無効です。

関連項目:

パーティション表と索引の参照

dblink

表またはビューがあるリモート・データベースへのデータベース・リンクの完全または部分的な名前を指定します。Oracle Databaseの分散機能を使用している場合にのみ、リモート表またはリモート・ビューに行を挿入できます。

dblinkを指定しない場合、その表またはビューはローカル・データベース内にあるとみなされます。

ノート:

Oracle Database 12c リリース 2 (12.2)以降では、INSERT 文で、リモートの LOB ロケータをバインド変数として指定できます。詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』の「分散 LOB」の章を参照してください。

関連項目:

- データベース・リンクの参照方法の詳細は、「スキーマ・オブジェクトの構文およびSQL文の構成要素」および「リモート・データベース内のオブジェクトの参照」を参照してください。
- リモート・データベースへの挿入: 例

subquery_restriction_clause

subquery_restriction_clauseを使用すると、次のいずれかの方法で副問合せを制限できます。

WITH READ ONLY

WITH READ ONLYを指定すると、表またはビューを更新禁止にできます。

WITH CHECK OPTION

WITH CHECK OPTIONを指定すると、副問合せに含まれない行を生成する表またはビューの変更を禁止できます。この句をDML文の副問合せ内で使用する場合、FROM句内の副問合せには指定できますが、WHERE句内の副問合せには指定できません。

CONSTRAINT constraint

CHECK OPTION制約の名前を指定します。この識別子を省略した場合は、Oracleによって自動的にSYS_Cnという形式の制約名が割り当てられます(nはデータベース内で制約名を一意にするための整数)。

関連項目:

WITH CHECK OPTION句の使用方法: 例

table_collection_expression

table_collection_expressionを使用すると、問合せおよびDML操作で、collection_expression値を表として扱うことができます。collection_expressionには、副問合せ、列、ファンクションまたはコレクション・コンストラクタのいずれかを指定できます。その形式にかかわらず、集合値(ネストした表型またはVARRAY型の値)を戻す必要があります。このようなコレクションの要素抽出プロセスをコレクション・ネスト解除といいます。

TABLEコレクション式を親表と結合する場合は、オプションのプラス(+)には大きな意味があります。+を指定すると、その2つの外部結合が作成され、コレクション式がNULLの場合でも、外部表の行が問合せで戻されるようになります。

ノート:

以前のリリースの Oracle では、collection_expression が副問合せの場合、table_collection_expression を THE subquery と表現していました。現在、このような表現方法は非推奨になっています。

関連項目:

表のコレクション: 例

t_alias

相関名を指定します。これは、文中で参照する表、ビュー、マテリアライズド・ビューまたは副問合せの別名です。

表の別名の制限事項

マルチテーブル・インサート中はt_aliasを指定できません。

column

表、ビューまたはマテリアライズド・ビューの列を指定します。挿入された行では、このリストにある各列にvalues_clauseまたは副問合せの値が代入されます。INVISIBLE列に値を割り当てる場合は、列をこのリストに含める必要があります。

このリストに1つ以上の列を指定しない場合、挿入された行の、指定しなかった列の値には、表の作成時または最終更新時に指定した列のデフォルト値が使用されます。指定しなかった列のいずれかにNOT NULL制約があり、デフォルト値がない場合、

制約違反のエラーが発生してINSERT文がロールバックされます。列のデフォルト値の詳細は、「CREATE TABLE」を参照してください。

列リストを指定しない場合は、values_clauseまたは問合せに、表の列をすべて指定する必要があります。

values_clause

シングルテーブル・インサート操作の場合、表またはビューに挿入する値の行を指定します。なお、値は、列リスト内の各列についてvalues_clauseに指定する必要があります。列リストを指定しない場合、values_clauseまたは副問合せで、表の各列の値を指定する必要があります。

マルチテーブル・インサート操作の場合、values_clauseの各式は、副問合せのSELECT構文のリストによって戻される列を参照する必要があります。values_clauseを省略すると、副問合せのSELECT構文のリストによって挿入する値が決定されるため、insert_into_clauseに対応する列リストと同じ列数を含んでいる必要があります。
insert_into_clauseで列リストを指定しない場合、対象となる表のすべての列に対する値を計算した行を指定する必要があります。

どちらの挿入操作の場合も、insert_into_clauseで列リストを指定すると、リストの各列に値の句または副問合せからの対応する値が割り当てられます。values_clauseの任意の値に対してDEFAULTを指定できます。表またはビューの対応する列に対してデフォルト値を指定すると、その値が挿入されます。対応する列に対してデフォルト値を指定しない場合、NULLが挿入されます。有効な式の構文の詳細は、「SQL式」および「SELECT」を参照してください。

挿入値の制限事項

値には、次の制限事項があります。

- BFILEロケータをNULLに、またはディレクトリ名およびファイル名に初期化するまで、BFILE値を挿入できません。

  関連項目:

  - BFILE値の初期化の詳細およびBFILEへの挿入例については、「BFILENAME」 を参照してください。
  - BFILEロケータの初期化の詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。
- リスト・パーティション表に挿入する場合、1つのパーティションのpartition_key_valueリストに存在していないパーティション化キー列に値を挿入できません。
- ビューに挿入する場合は、DEFAULTを指定できません。
- RAW列に文字列リテラルを挿入する場合、後続の問合せ中にRAW列にある索引は使用されずに、全表スキャンが行われます。

  関連項目:

  - XMLType表への値の挿入の詳細は、「SQL文でのXMLの使用方法」を参照してください。
  - 「置換可能な表および列への挿入：例」、「TO_LOBファンクションを使用した挿入：例」、「表への順序値の挿入：例」および「バインド変数を使用した挿入：例」を参照してください。

returning_clause

この句を使用すると、DML文に影響される行を取り出すことができます。この句は、表、マテリアライズド・ビュー、および単一の

実表を持つビューに指定できます。

returning_clauseを指定したDML文を単一行に実行すると、影響された行、ROWID、および処理された行へのREFを使用している列式が取り出され、ホスト変数またはPL/SQL変数に格納されます。

returning_clauseを指定したDML文を複数行に実行すると、式の値、ROWIDおよび処理された行に関連するREFがバインド配列に格納されます。

expr

exprリストの各項目は、適切な構文で表す必要があります。

INTO

INTO句を指定すると、変更された行の値を、data_itemリストに指定する変数に格納できます。

data_item

data_itemはそれぞれ、取り出したexpr値を格納するためのホスト変数またはPL/SQL変数です。

RETURNINGリストの各式については、INTOリストに、対応する型に互換性があるPL/SQL変数またはホスト変数を指定する必要があります。

制限事項

RETURNING句には、次の制限事項があります。

- exprに次の制限事項があります。
  - UPDATE文およびDELETE文の場合、各exprは、単純式または単一セットの集計ファンクション式である必要があります。1つのreturning_clause内に単純式と単一セットの集計ファンクション式を混在させることはできません。INSERT文の場合、各exprは単純式である必要があります。INSERT文のRETURNING句では、集計ファンクションはサポートされていません。
  - 単一セットの集計ファンクション式をDISTINCTキーワードに含めることはできません。
- exprのリストに主キー列または他のNOT NULL列が含まれている場合は、表にBEFORE UPDATEトリガーが定義されていると、UPDATE文の実行に失敗します。
- マルチテーブル・インサートではreturning_clauseを指定できません。
- パラレルDMLまたはリモート・オブジェクトにはこの句を使用できません。
- LONG型を取り出すことはできません。
- INSTEAD OFトリガーが定義されたビューに対して指定することはできません。

関連項目:

BULK COLLECT句を使用してコレクション変数に複数の値を戻す場合は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

multi_table_insert

マルチテーブル・インサートでは、副問合せの評価によって戻された行から導出され、計算された行が1つ以上の表に挿入されます。

副問合せのSELECT構文のリストでは、表の別名は定義されていません。そのため、SELECT構文のリストに依存する句で

は、表の別名は参照できません。たとえば、式内のオブジェクト列を参照しようとしても、表の別名は参照できません。表の別名とともに式を使用する場合は、列の別名を使用して式をSELECT構文のリストに挿入してから、マルチテーブル・インサートのVALUES句またはWHEN条件で列の別名を参照する必要があります。

ALL into_clause

ALLの後に複数のinsert_into_clausesを指定すると、無条件のマルチテーブル・インサートを実行できます。副問合せによって戻される各行に対して、各insert_into_clauseが1回実行されます。

conditional_insert_clause

conditional_insert_clauseを指定すると、条件付きのマルチテーブル・インサートを実行できます。各insert_into_clauseは、それに対応するWHEN条件によってフィルタ処理されます。この条件によって、insert_into_clauseが実行されるかどうかが決まります。WHEN条件の各式は、副問合せのSELECT構文のリストによって戻される列を参照する必要があります。1つのマルチテーブル・インサートには、最大127個のWHEN句を指定できます。

ALL

ALL(デフォルト値)を指定すると、他のWHEN句の評価結果にかかわらず、各WHEN句が評価されます。条件が真と評価された各WHEN句に対して、対応するINTO句リストが実行されます。

FIRST

FIRSTを指定すると、文で指定されている順序で各WHEN句が評価されます。真と評価された最初のWHEN句に対して、対応するINTO句が実行され、指定された行に対する後続のWHEN句はスキップされます。

ELSE clause

指定された行に対して、WHEN句が真と評価されない場合、次のようになります。

- ELSE句を指定する場合、ELSE句に関連付けられたINTO句リストが実行されます。
- ELSE句を指定しない場合、その行に対して何も実行されません。

関連項目:

「マルチテーブル・インサート: 例」

マルチテーブル・インサートの制限事項

マルチテーブル・インサートには、次の制限事項があります。

- 表のみにマルチテーブル・インサートが実行でき、ビューまたはマテリアライズド・ビューには実行できません。
- リモート表には、マルチテーブル・インサートを実行できません。
- マルチテーブル・インサートの実行時、表のコレクション式は指定できません。
- マルチテーブル・インサートは、対象となる表が索引構成されている場合、または対象となる表にビットマップ索引が定義されている場合は、パラレル化されません。
- プラン・スタビリティは、マルチテーブル・インサート文にはサポートされません。
- マルチテーブル・インサート文のどの部分にも順序を指定することはできません。マルチテーブル・インサートは、単一のSQL文とみなされます。したがって、NEXTVALを初めて参照するときに、その次の番号が生成された後、この番号と同じ番号が、この文の後続のすべての参照で戻されます。

subquery

表に挿入される行を戻す副問合せを指定します。副問合せによって、INSERT文の対象となる表を含む任意の表、ビューおよびマテリアライズド・ビューを参照できます。副問合せで選択された行が1行もない場合、表に行は挿入されません。

subqueryをTO_LOBファンクションと組み合せて、LONG列にある値を、同じ表の異なる列または別の表にあるLOB値に変換できます。

- ビュー内でLONG値を別の列のLOB値に移行する場合、実表内で移行を実行してからビューにLOB列を追加する必要があります。
- リモート表のLONG値をローカル表のLOB値に移行する場合、TO_LOBファンクションを使用してリモート表で移行を実行してから、INSERT ... subquery操作を実行して、リモート表からローカル表にLOB値をコピーする必要があります。

副問合せを使用した挿入のノート

副問合せを使用した挿入時には、次の点に注意します。

- subqueryが、既存のマテリアライズド・ビューと部分的または完全に同じビューを戻す場合、subqueryに指定された1つ以上の表のかわりにマテリアライズド・ビューがクエリー・リライトに使用されることがあります。

  関連項目:

  マテリアライズド・ビューおよびクエリー・リライトの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

- subqueryがリモート・オブジェクトを参照する場合、参照がローカル・データベース上でオブジェクトにループバックしないかぎり、INSERTはパラレルで実行されます。ただし、DML_table_expression_clauseのsubqueryがリモート・オブジェクトを参照する場合は、INSERT操作はシリアルで実行されます。詳細は、「parallel_clause」を参照してください。
- subqueryにORDER BY句が含まれる場合、属性クラスタリング表のプロパティを使用して指定された行順序はオーバーライドされます。

関連項目:

- 副問合せを持つ値の挿入: 例
- BFILEへの挿入例については、「BFILENAME」を参照してください。
- BFILEの初期化の詳細は、『Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイド』を参照してください。
- 有効な式の構文の詳細は、「SQL式」および「SELECT」を参照してください。

error_logging_clause

error_logging_clauseを使用すると、DMLエラーと、その影響を受ける行のログ列値を取得して、エラー・ロギング表に保存できます。

INTO table

エラー・ロギング表の名前を指定します。この句を省略すると、DBMS_ERRLOGパッケージで生成されたデフォルトの名前が割り

当てられます。エラー・ログ表のデフォルトの名前は、DML操作の対象となっている表の名前の最初の25文字を、ERR$_の後に加えたものです。

simple_expression

文のタグとして使用する値を指定します。エラー・ロギング表内のエラーは、この文で識別することになります。この式には、テキスト・リテラル、数値リテラル、一般のSQL式(バインド変数など)のいずれかを指定できます。テキスト・リテラルに変換する場合は、TO_CHAR(SYSDATE)のようなファンクション式も使用できます。

REJECT LIMIT

この句を使用すると、記録するエラーの数の上限値を整数で指定できます。エラーの数がこの値を超えると、文が終了し、その文によって変更された内容がロールバックされます。この拒否の制限のデフォルトは0(ゼロ)です。パラレルDML操作の場合、拒否の制限はパラレル・サーバーごとに適用されます。

DMLエラー・ロギングの制限事項:

- 次の状態では、文の実行が失敗しロールバックは実行されますが、エラー・ロギング機能は起動されません。
    - 遅延制約の違反
    - 一意制約違反または一意索引違反を発生する、ダイレクト・パスのINSERTまたはMERGE操作
    - 一意制約違反または一意索引違反を発生する、更新操作のUPDATEまたはMERGE
- エラー・ロギング表では、LONG、LOBまたはオブジェクト型の列のエラーは追跡できません。ただし、DML操作を行う表に、これらの型の列を含めることができます。
    - 対応するエラー・ロギング表を、未サポートの型の列を含めるために作成または修正する場合、その列の名前が、ターゲットのDML表の未サポートの列に対応していると、DML文が解析時にエラーになります。
    - エラー・ロギング表に未サポートの列型が含まれない場合、エラー数が拒否の制限に達するまで、すべてのDMLエラーが記録されます。エラーが発生した行では、列の値とエラー・ロギング表の対応する列が、制御情報とともにログに記録されます。

関連項目:

- DBMS_ERRLOGパッケージのcreate_error_logプロシージャの使用方法は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。DMLエラー・ロギングに関する一般情報は、『Oracle Database管理者ガイド』を参照してください。
- エラー・ロギングによる表への挿入: 例

例

表への値の挿入: 例

次の文は、サンプル表departmentsに行を挿入します。

```
INSERT INTO departments
   VALUES (280, 'Recreation', 121, 1700);
```

manager_id列のデフォルト値が121としてdepartments表が作成された場合、次の文を発行できます。

```
INSERT INTO departments
   VALUES (280, 'Recreation', DEFAULT, 1700);
```

次の文は、employees表に6つの列で構成される行を挿入します。NULLまたは科学表記の数値を設定されている列がそれぞれ1つ含まれています。

```
INSERT INTO employees (employee_id, last_name, email,
      hire_date, job_id, salary, commission_pct)
   VALUES (207, 'Gregory', 'pgregory@example.com',
      sysdate, 'PU_CLERK', 1.2E3, NULL);
```

次の文は、前述の例と同じ結果を表しますが、DML_table_expression_clauseにある副問合せを使用します。

```
INSERT INTO
   (SELECT employee_id, last_name, email, hire_date, job_id,
      salary, commission_pct FROM employees)
   VALUES (207, 'Gregory', 'pgregory@example.com',
      sysdate, 'PU_CLERK', 1.2E3, NULL);
```

副問合せを持つ値の挿入: 例

次の文は、歩合給が給与の25%を超える従業員をbonuses表(「表へのマージ: 例」で作成)にコピーします。

```
INSERT INTO bonuses
   SELECT employee_id, salary*1.1
   FROM employees
   WHERE commission_pct > 0.25;
```

エラー・ロギングによる表への挿入: 例

次の文は、サンプル・スキーマhrにraises表を作成し、DBMS_ERRLOGパッケージを使用してエラー・ロギング表を作成し、raises表にemployees表からのデータを移入します。raisesのチェック制約に違反する挿入操作があると、その行をerrlogで参照できます。発生したエラーが10を超えると、その文は異常終了し、挿入された内容はロールバックされます。

```
CREATE TABLE raises (emp_id NUMBER, sal NUMBER
   CONSTRAINT check_sal CHECK(sal > 8000));
EXECUTE DBMS_ERRLOG.CREATE_ERROR_LOG('raises', 'errlog');

INSERT INTO raises
   SELECT employee_id, salary*1.1 FROM employees
   WHERE commission_pct > .2
   LOG ERRORS INTO errlog ('my_bad') REJECT LIMIT 10;
SELECT ORA_ERR_MESG$, ORA_ERR_TAG$, emp_id, sal FROM errlog;
ORA_ERR_MESG$               ORA_ERR_TAG$         EMP_ID SAL
--------------------------- -------------------- ------ -------
ORA-02290: check constraint my_bad               161    7700
 (HR.SYS_C004266) violated
```

リモート・データベースへの挿入: 例

次の文は、データベース・リンクremoteがアクセスできるデータベース上の、ユーザーhrが所有するemployees表に行を挿入します。

```
INSERT INTO employees@remote
   VALUES (8002, 'Juan', 'Fernandez', 'juanf@example.com', NULL,
   TO_DATE('04-OCT-1992', 'DD-MON-YYYY'), 'SH_CLERK', 3000,
   NULL, 121, 20);
```

表への順序値の挿入: 例

次の文は、departments_seq順序の次の値を持つ行を、departments表に挿入します。

```
INSERT INTO departments
   VALUES  (departments_seq.nextval, 'Entertainment', 162, 1400);
```

バインド変数を使用した挿入: 例

次の例では、出力バインド変数:bnd1および:bnd2に挿入された行の値を戻します。バインド変数は最初に宣言しておく必要があります。

```
INSERT INTO employees
      (employee_id, last_name, email, hire_date, job_id, salary)
   VALUES
   (employees_seq.nextval, 'Doe', 'john.doe@example.com',
       SYSDATE, 'SH_CLERK', 2400)
   RETURNING salary*12, job_id INTO :bnd1, :bnd2;
```

置換可能な表および列への挿入: 例

次の例は、persons表(「置換可能な表および列のサンプル」で作成)に挿入します。最初の文は、ルート型person_tを使用します。2番目の挿入は、person_tのサブタイプemployee_tを使用し、3番目の挿入はemployee_tのサブタイプpart_time_emp_tを使用します。

```
INSERT INTO persons VALUES (person_t('Bob', 1234));
INSERT INTO persons VALUES (employee_t('Joe', 32456, 12, 100000));
INSERT INTO persons VALUES (
   part_time_emp_t('Tim', 5678, 13, 1000, 20));
```

次の例は、books表(「置換可能な表および列のサンプル」で作成)に挿入します。属性値の指定が、置換可能な表の例と同一であることに注意してください。

```
INSERT INTO books VALUES (
   'An Autobiography', person_t('Bob', 1234));
INSERT INTO books VALUES (
   'Business Rules', employee_t('Joe', 3456, 12, 10000));
INSERT INTO books VALUES (
   'Mixing School and Work',
   part_time_emp_t('Tim', 5678, 13, 1000, 20));
```

組込みファンクションおよび条件を使用して、置換可能な表および列からデータを抽出することができます。例は、「TREAT」、「SYS_TYPEID」および「IS OF type条件」を参照してください。

TO_LOBファンクションを使用した挿入: 例

次の例では、LONGデータを、次のlong_tab表のLOB列にコピーします。

```
CREATE TABLE long_tab (pic_id NUMBER, long_pics LONG RAW);
```

まず、LOBを持つ表を作成します。

```
CREATE TABLE lob_tab (pic_id NUMBER, lob_pics BLOB);
```

次に、INSERT ... SELECTを使用して、LONG列のすべての行にあるデータを、新しく作成したLOB列にコピーします。

```
INSERT INTO lob_tab
   SELECT pic_id, TO_LOB(long_pics) FROM long_tab;
```

移行が問題なく終了したことを確認したら、long_pics表を削除できます。別の方法として、表が他の列を含む場合、次のように入力して表からLONG列を削除できます。

```
ALTER TABLE long_tab DROP COLUMN long_pics;
```

マルチテーブル・インサート: 例

次の例では、マルチテーブル・インサート構文を使用して、サンプル表sh.salesに異なる構造を持つ入力表からデータを挿入

しています。

ノート:

例を簡潔に示すために表の列が無視されているため、sales 表の NOT NULL 制約は使用禁止になっています。

入力表は次のように構成されています。

```
SELECT * FROM sales_input_table;

PRODUCT_ID CUSTOMER_ID WEEKLY_ST  SALES_SUN  SALES_MON  SALES_TUE  SALES_WED
SALES_THU  SALES_FRI  SALES_SAT
---------- ----------- --------- ---------- ---------- ---------- ----------
- ---------- ----------
       111         222 01-OCT-00        100        200        300        400
500        600         700
       222         333 08-OCT-00        200        300        400        500
600        700         800
       333         444 15-OCT-00        300        400        500        600
700        800         900
```

マルチテーブル・インサートの文を次に示します。

```
INSERT ALL
      INTO sales (prod_id, cust_id, time_id, amount)
      VALUES (product_id, customer_id, weekly_start_date, sales_sun)
      INTO sales (prod_id, cust_id, time_id, amount)
      VALUES (product_id, customer_id, weekly_start_date+1, sales_mon)
      INTO sales (prod_id, cust_id, time_id, amount)
      VALUES (product_id, customer_id, weekly_start_date+2, sales_tue)
      INTO sales (prod_id, cust_id, time_id, amount)
      VALUES (product_id, customer_id, weekly_start_date+3, sales_wed)
      INTO sales (prod_id, cust_id, time_id, amount)
      VALUES (product_id, customer_id, weekly_start_date+4, sales_thu)
      INTO sales (prod_id, cust_id, time_id, amount)
      VALUES (product_id, customer_id, weekly_start_date+5, sales_fri)
      INTO sales (prod_id, cust_id, time_id, amount)
      VALUES (product_id, customer_id, weekly_start_date+6, sales_sat)
   SELECT product_id, customer_id, weekly_start_date, sales_sun,
      sales_mon, sales_tue, sales_wed, sales_thu, sales_fri, sales_sat
      FROM sales_input_table;
```

sales表には次の行のみが存在するものとします。内容は次のとおりです。

```
SELECT * FROM sales
   ORDER BY prod_id, cust_id, time_id;
   PROD_ID    CUST_ID TIME_ID   C   PROMO_ID QUANTITY_SOLD     AMOUNT       COST
---------- ---------- --------- - ---------- ------------- ---------- ----------
       111        222 01-OCT-00                                   100
       111        222 02-OCT-00                                   200
       111        222 03-OCT-00                                   300
       111        222 04-OCT-00                                   400
       111        222 05-OCT-00                                   500
       111        222 06-OCT-00                                   600
       111        222 07-OCT-00                                   700
       222        333 08-OCT-00                                   200
       222        333 09-OCT-00                                   300
       222        333 10-OCT-00                                   400
       222        333 11-OCT-00                                   500
       222        333 12-OCT-00                                   600
       222        333 13-OCT-00                                   700
```

```
       222        333 14-OCT-00                                  800
       333        444 15-OCT-00                                  300
       333        444 16-OCT-00                                  400
       333        444 17-OCT-00                                  500
       333        444 18-OCT-00                                  600
       333        444 19-OCT-00                                  700
       333        444 20-OCT-00                                  800
       333        444 21-OCT-00                                  900
```

次の例は、複数の表に挿入します。販売員に様々なサイズの注文に関する情報を提供するとします。小口、中口、大口および特別注文についての表を作成し、これらの表にサンプル表oe.ordersのデータを移入します。

```
CREATE TABLE small_orders
   (order_id       NUMBER(12)   NOT NULL,
    customer_id    NUMBER(6)    NOT NULL,
    order_total    NUMBER(8,2),
    sales_rep_id   NUMBER(6)
   );
CREATE TABLE medium_orders AS SELECT * FROM small_orders;
CREATE TABLE large_orders AS SELECT * FROM small_orders;
CREATE TABLE special_orders
   (order_id       NUMBER(12)    NOT NULL,
    customer_id    NUMBER(6)     NOT NULL,
    order_total    NUMBER(8,2),
    sales_rep_id   NUMBER(6),
    credit_limit   NUMBER(9,2),
    cust_email     VARCHAR2(40)
   );
```

最初のマルチテーブル・インサートでは、小口、中口および大口注文の表に対してのみ移入を行います。

```
INSERT ALL
   WHEN order_total <= 100000 THEN
      INTO small_orders
   WHEN order_total > 1000000 AND order_total <= 200000 THEN
      INTO medium_orders
   WHEN order_total > 200000 THEN
      INTO large_orders
   SELECT order_id, order_total, sales_rep_id, customer_id
      FROM orders;
```

large_orders表への挿入のかわりに、ELSE句を使用しても、同じ結果が得られます。

```
INSERT ALL
   WHEN order_total <= 100000 THEN
      INTO small_orders
   WHEN order_total > 100000 AND order_total <= 200000 THEN
      INTO medium_orders
   ELSE
      INTO large_orders
   SELECT order_id, order_total, sales_rep_id, customer_id
      FROM orders;
```

次の例は、前述の例と同様、小口、中口および大口注文の表に挿入し、件数が290,000を超える注文をspecial_orders表に挿入します。この表は、文を単純にするために列の別名を使用する方法も示しています。

```
INSERT ALL
   WHEN ottl <= 100000 THEN
      INTO small_orders
         VALUES(oid, ottl, sid, cid)
   WHEN ottl > 100000 and ottl <= 200000 THEN
      INTO medium_orders
         VALUES(oid, ottl, sid, cid)
   WHEN ottl > 200000 THEN
```

```
      into large_orders
         VALUES(oid, ottl, sid, cid)
   WHEN ottl > 290000 THEN
      INTO special_orders
   SELECT o.order_id oid, o.customer_id cid, o.order_total ottl,
      o.sales_rep_id sid, c.credit_limit cl, c.cust_email cem
      FROM orders o, customers c
      WHERE o.customer_id = c.customer_id;
```

最後の例は、FIRST句を使用して、件数が290,000を超える注文をspecial_orders表に挿入し、これらの注文をlarge_orders表から削除します。

```
INSERT FIRST
   WHEN ottl <= 100000 THEN
      INTO small_orders
         VALUES(oid, ottl, sid, cid)
   WHEN ottl > 100000 and ottl <= 200000 THEN
      INTO medium_orders
         VALUES(oid, ottl, sid, cid)
   WHEN ottl > 290000 THEN
      INTO special_orders
   WHEN ottl > 200000 THEN
      INTO large_orders
         VALUES(oid, ottl, sid, cid)
   SELECT o.order_id oid, o.customer_id cid, o.order_total ottl,
      o.sales_rep_id sid, c.credit_limit cl, c.cust_email cem
      FROM orders o, customers c
      WHERE o.customer_id = c.customer_id;
```

単一文を使用した複数行の挿入: 例

次の文は、people、patientsおよびstaffという名前の3つの表を作成します。

```
CREATE TABLE people (
  person_id   INTEGER NOT NULL PRIMARY KEY,
  given_name  VARCHAR2(100) NOT NULL,
  family_name VARCHAR2(100) NOT NULL,
  title       VARCHAR2(20),
  birth_date  DATE
);
CREATE TABLE patients (
  patient_id          INTEGER NOT NULL PRIMARY KEY REFERENCES people (person_id),
  last_admission_date DATE
);
CREATE TABLE staff (
  staff_id   INTEGER NOT NULL PRIMARY KEY REFERENCES people (person_id),
  hired_date DATE
);
```

次の文は、people表に行を挿入します。

```
INSERT INTO people
VALUES (1, 'Dave', 'Badger', 'Mr', date'1960-05-01');
```

次の文は、birth_date列に指定された値がないため、エラーを戻します。

```
INSERT INTO people
VALUES (2, 'Simon', 'Fox', 'Mr');
```

次の文は、people表に行を挿入します。

```
INSERT INTO people (person_id, given_name, family_name, title)
VALUES (2, 'Simon', 'Fox', 'Mr');
```

次の文は、people表に1つの行を挿入し、DUAL表から静的値を選択してtitle列の値を移入します。

```
INSERT INTO people (person_id, given_name, family_name, title)
VALUES (3, 'Dave', 'Frog', (SELECT 'Mr' FROM dual));
```

次の文は、SELECT文を使用してpeople表に複数の行を挿入します。

```
INSERT INTO people (person_id, given_name, family_name, title)
  WITH names AS (
    SELECT 4, 'Ruth',     'Fox',      'Mrs'    FROM dual UNION ALL
    SELECT 5, 'Isabelle', 'Squirrel', 'Miss'   FROM dual UNION ALL
    SELECT 6, 'Justin',   'Frog',     'Master' FROM dual UNION ALL
    SELECT 7, 'Lisa',     'Owl',      'Dr'     FROM dual
  )
  SELECT * FROM names;
```

次の文は、これまでのDML操作をすべてロールバックします。

```
ROLLBACK;
```

次の文は、WHERE条件を指定したSELECT文を使用して、複数の行をpeople表に挿入します。

```
INSERT INTO people (person_id, given_name, family_name, title)
  WITH names AS (
    SELECT 4, 'Ruth',     'Fox' family_name,      'Mrs'    FROM dual UNION ALL
    SELECT 5, 'Isabelle', 'Squirrel' family_name, 'Miss'   FROM dual UNION ALL
    SELECT 6, 'Justin',   'Frog' family_name,     'Master' FROM dual UNION ALL
    SELECT 7, 'Lisa',     'Owl' family_name,      'Dr'     FROM dual
  )
  SELECT * FROM names
  WHERE  family_name LIKE 'F%';
```

次の文は、これまでのDML操作をすべてロールバックします。

```
ROLLBACK;
```

次の文は、INSERT ALL文を使用して、people表、patients表およびstaff表に複数の行を挿入します。

```
INSERT ALL
  /* Every one is a person */
  INTO people (person_id, given_name, family_name, title)
    VALUES (id, given_name, family_name, title)
  INTO patients (patient_id, last_admission_date)
    VALUES (id, admission_date)
  INTO staff (staff_id, hired_date)
    VALUES (id, hired_date)
  WITH names AS (
    SELECT 4 id, 'Ruth' given_name, 'Fox' family_name, 'Mrs' title,
           NULL hired_date, DATE'2009-12-31' admission_date
    FROM   dual UNION ALL
    SELECT 5 id, 'Isabelle' given_name, 'Squirrel' family_name, 'Miss' title ,
           NULL hired_date, DATE'2014-01-01' admission_date
    FROM   dual UNION ALL
    SELECT 6 id, 'Justin' given_name, 'Frog' family_name, 'Master' title,
           NULL hired_date, DATE'2015-04-22' admission_date
    FROM   dual UNION ALL
    SELECT 7 id, 'Lisa' given_name, 'Owl' family_name, 'Dr' title,
           DATE'2015-01-01' hired_date, NULL admission_date
    FROM   dual
  )
  SELECT * FROM names;
```

次の文は、これまでのDML操作をすべてロールバックします。

```
ROLLBACK;
```

次の文は、様々な条件を指定したINSERT ALL文を使用して、people表、patients表およびstaff表に複数の行を挿入します。

```
INSERT ALL
  /* Everyone is a person, so insert all rows into people */
  WHEN 1=1 THEN
    INTO people (person_id, given_name, family_name, title)
    VALUES (id, given_name, family_name, title)
  /* Only people with an admission date are patients */
  WHEN admission_date IS NOT NULL THEN
    INTO patients (patient_id, last_admission_date)
    VALUES (id, admission_date)
  /* Only people with a hired date are staff */
  WHEN hired_date IS NOT NULL THEN
    INTO staff (staff_id, hired_date)
    VALUES (id, hired_date)
  WITH names AS (
    SELECT 4 id, 'Ruth' given_name, 'Fox' family_name, 'Mrs' title,
           NULL hired_date, DATE'2009-12-31' admission_date
    FROM   dual UNION ALL
    SELECT 5 id, 'Isabelle' given_name, 'Squirrel' family_name, 'Miss' title ,
           NULL hired_date, DATE'2014-01-01' admission_date
    FROM   dual UNION ALL
    SELECT 6 id, 'Justin' given_name, 'Frog' family_name, 'Master' title,
           NULL hired_date, DATE'2015-04-22' admission_date
    FROM   dual UNION ALL
    SELECT 7 id, 'Lisa' given_name, 'Owl' family_name, 'Dr' title,
           DATE'2015-01-01' hired_date, NULL admission_date
    FROM   dual
  )
  SELECT * FROM names;
```

Live SQL:

「単一文を使用した複数行の挿入」で、Oracle Live SQL に関連する例を参照および実行します。

LOCK TABLE

目的

LOCK TABLE文を使用すると、1つ以上の表、表パーティションまたは表サブパーティションを特定のモードでロックできます。操作中の表またはビューに対する他のユーザーによるアクセスを許可または制限するため、自動ロックを手動で無効にします。

一部の形式のロックは、同じ表に同時に設定できます。それ以外のロックは、表ごとに1つしか設定できません。

ロックされた表は、トランザクションをコミットするか、全体をロールバックするか、または表をロックする前のセーブポイントにロールバックするまでロックされています。

ロックした場合でも他のユーザーが表を問い合せることができます。問合せによって表がロックされることはありません。読取りプログラムは書込みプログラムをブロックすることはなく、書込みプログラムが読取りプログラムをブロックすることもありません。

関連項目:

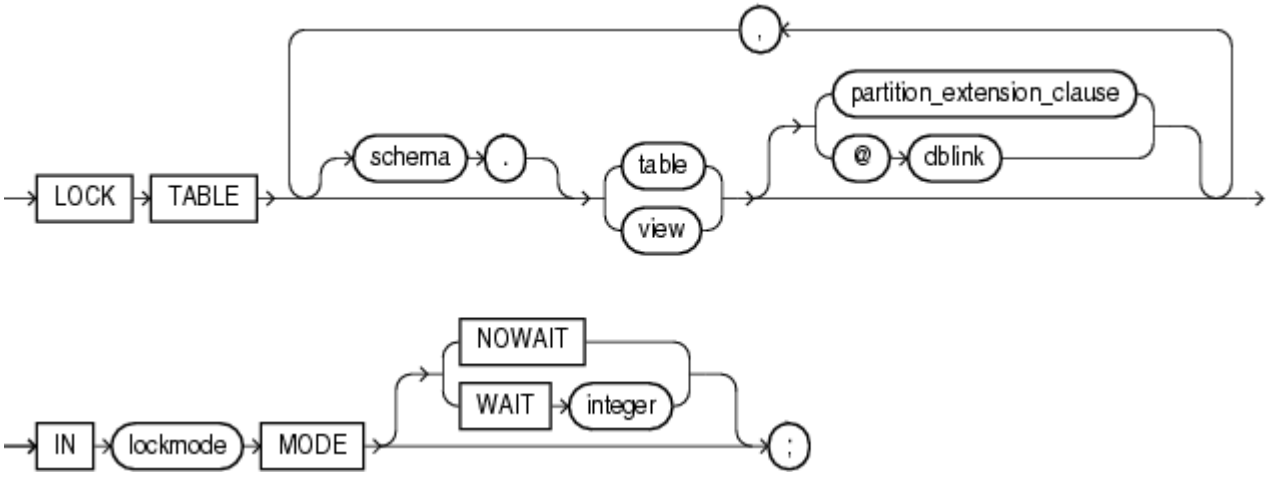
- ロック・モードの相互作用については、『Oracle Database概要』を参照してください。
- COMMIT
- ROLLBACK
- SAVEPOINT

前提条件

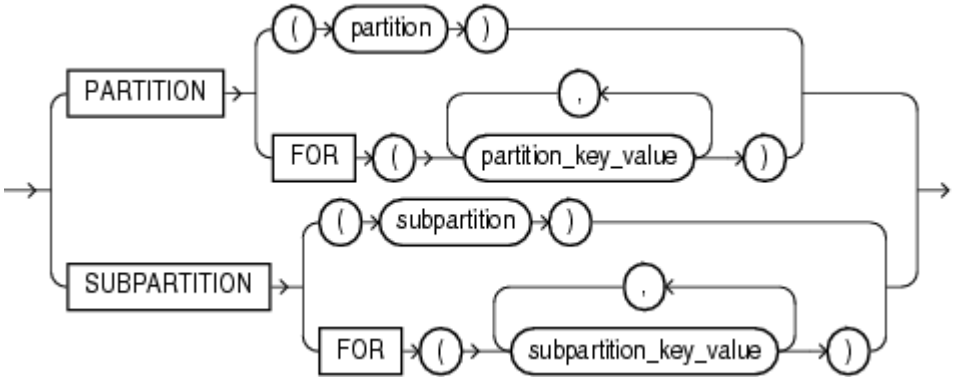
表またはビューが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、LOCK ANY TABLEシステム権限、または表やビューに対するオブジェクト権限(READオブジェクト権限を除く)が必要です。

構文

lock_table::=



partition_extension_clause::=



セマンティクス

schema

表またはビューが含まれているスキーマを指定します。schemaを指定しない場合、表またはビューは自分のスキーマにあるとみなされます。

table | view

ロックする表またはビューの名前を指定します。

viewを指定した場合、ビューの実表がロックされます。

partition_extension_clauseを指定した場合、Oracle Databaseでは最初にその表が暗黙的にロックされます。表ロックは、パーティションまたはサブパーティションに指定するロックと同じです。ただし、次の2つの例外があります。

- SHAREロックをサブパーティションに指定した場合、表が暗黙的にROW SHAREロックされます。
- EXCLUSIVEロックをサブパーティションに指定した場合、表が暗黙的にROW EXCLUSIVEロックされます。

PARTITIONを指定し、tableがコンポジット・パーティション化されている場合、パーティションのすべてのサブパーティションがロックされます。

表のロックの制限事項

表のロックには、次の制限事項が適用されます。

- viewは、階層の一部の場合、階層のルートである必要があります。
- ロックは、自動リスト・パーティション表の既存のパーティションに対してのみ適用できます。つまり、次の文を指定する場合、パーティション・キー値は、後でオンデマンドで作成できるパーティションに対応するパーティションではなく、表内にすでに存在するパーティションに対応している必要があります。

  ```
  LOCK TABLE ... PARTITION FOR (partition_key_value) ...
  ```

dblink

表またはビューが格納されている、Oracle Databaseのリモート・データベースに対するデータベース・リンクを指定します。Oracle分散機能を使用している場合のみ、リモート・データベースで表およびビューをロックできます。LOCK TABLE文を使用してロックする表は、すべて同じデータベース上にある必要があります。

dblinkを指定しない場合、その表またはビューはローカル・データベース内にあるとみなされます。

関連項目:

データベース・リンクの指定方法の詳細は、「リモート・データベース内のオブジェクトの参照」を参照してください。

lockmode句

次のいずれかのモードを指定します。

ROW SHARE

ROW SHAREを指定すると、ロックされた表への同時アクセスは可能になりますが、排他アクセスのために表全体をロックできなくなります。ROW SHAREは、SHARE UPDATEと同じ意味で、以前のリリースのOracle Databaseとの互換性を保つために用意されています。

ROW EXCLUSIVE

ROW EXCLUSIVEは、ROW SHAREと同じですが、SHAREモードでロックはできません。ROW EXCLUSIVEロックは、更新、挿入、削除の実行時に自動的に適用されます。

SHARE UPDATE

「ROW SHARE」を参照してください。

SHARE

SHAREを指定すると、同時問合せは実行可能ですが、ロックされた表は更新できなくなります。

SHARE ROW EXCLUSIVE

SHARE ROW EXCLUSIVEは、表全体を見る場合に使用します。これを使用すると他のユーザーがその表内の行を見ることはできますが、SHAREモードで表のロックまたは行の更新を行うことはできません。

EXCLUSIVE

EXCLUSIVEを指定すると、ロックされた表上での問合せは実行可能ですが、他のアクティビティは実行できなくなります。

NOWAIT

NOWAITを指定すると、指定した表、パーティションまたは表のサブパーティションが他のユーザーによってすでにロックされている場合に、制御をすぐに戻すことができます。この場合、表、パーティションまたはサブパーティションが他のユーザーによってロックされていることを示すエラー・メッセージが戻ります。

WAIT

WAIT句を使用すると、LOCK TABLE文では、DMLロックを取得するまでに指定した時間(秒数)待機するように指定できます。integerの値に制限はありません。

NOWAITもWAITも指定しない場合には、表が利用可能になり、ロックされ、制御が戻されるまで、データベースは無限に待機します。データベースでDML文と同時にDDL文が実行されている場合、タイムアウトまたはデッドロックが発生することがあります。このようなタイムアウトまたはデッドロックが検出されると、エラーが戻されます。

関連項目:

表のロックの詳細は、『Oracle Database管理者ガイド』を参照してください。

例

表のロック: 例

次の文は、employees表を排他モードでロックします。他のユーザーがすでに表をロックしている場合でも、待ち状態にはなり

ません。

```
LOCK TABLE employees
   IN EXCLUSIVE MODE
   NOWAIT;
```

次の文は、データベース・リンクremoteを介してアクセスできるリモート表employeesをロックします。

```
LOCK TABLE employees@remote
   IN SHARE MODE;
```

19 SQL文: MERGEからUPDATE

この章では、次のSQL文について説明します。

- MERGE
- NOAUDIT (従来型監査)
- NOAUDIT (統合監査)
- PURGE
- RENAME
- REVOKE
- ROLLBACK
- SAVEPOINT
- SELECT
- SET CONSTRAINT[S]
- SET ROLE
- SET TRANSACTION
- TRUNCATE CLUSTER
- TRUNCATE TABLE
- UPDATE

MERGE

目的

MERGE文を使用すると、1つ以上のソースから行を選択し、表またはビューに対して更新および挿入できます。対象となる表またはビューに対して更新と挿入のどちらを実行するかを決定する条件を指定できます。

この文は、複数の操作を組み合せるときに便利です。DML文INSERT、UPDATEおよびDELETEを複数指定する必要がなくなります。

MERGEは、決定的な文です。対象となる表の同じ行を、同一のMERGE文で何度も更新することはできません。

ノート:

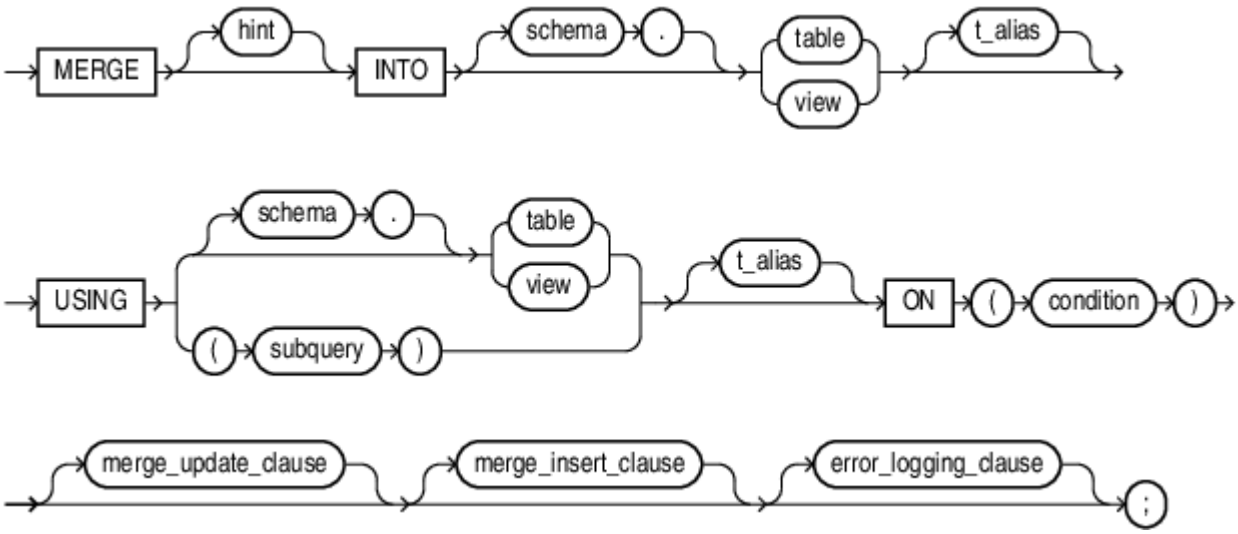
Oracle Database の以前のリリースでは、MERGE INTO 文を含むアプリケーションで Oracle Virtual Private Database ポリシーを作成すると、Virtual Private Database ポリシーが存在するため、MERGE INTO 文は ORA-28132「MERGE INTO 構文ではセキュリティ・ポリシーをサポートしていません」エラーにより回避されていました。Oracle Database 11g リリース 2(11.2.0.2)以降では、MERGE INTO 操作を含むアプリケーションでポリシーを作成できます。そのためには、DBMS_RLS.ADD_POLICY statement_types パラメータに INSERT、UPDATE および DELETE 文を含めるか、statement_types パラメータを完全に省略します。特定のタイプの SQL 文にポリシーを適用する方法の詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。

前提条件

対象となる表に対するINSERTオブジェクト権限とUPDATEオブジェクト権限、およびソース・オブジェクトに対するSELECTオブジェクト権限が必要です。merge_update_clauseのDELETE句を指定するには、対象となる表またはビューに対するDELETEオブジェクト権限も必要です。

構文

merge::=

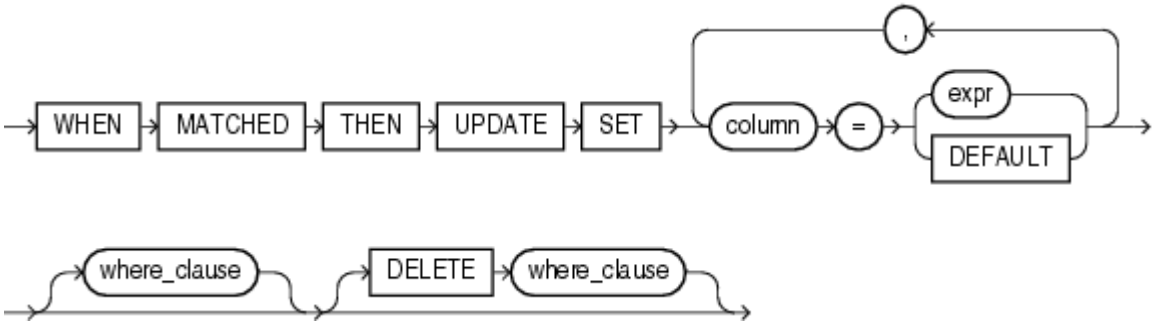


ノート:

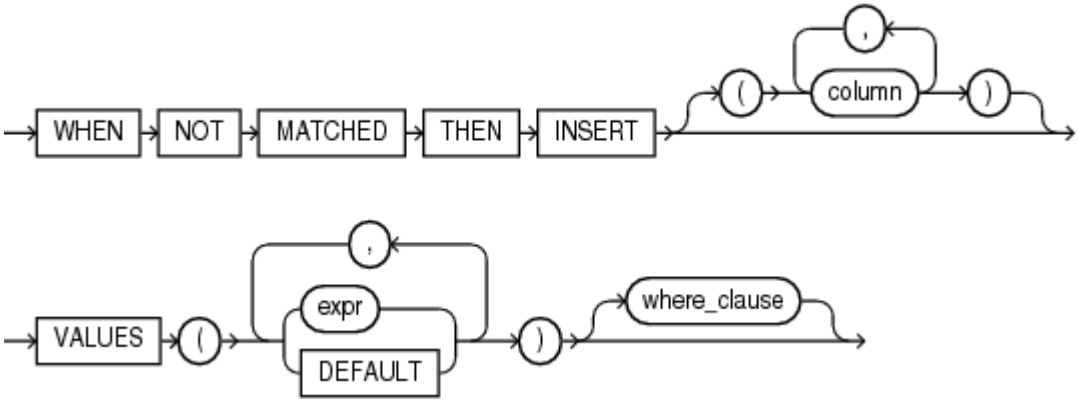
少なくとも merge_update_clause または merge_insert_clause 句のどちらかを指定する必要があります。

(merge_update_clause::=、merge_insert_clause::=、error_logging_clause::=)

merge_update_clause::=

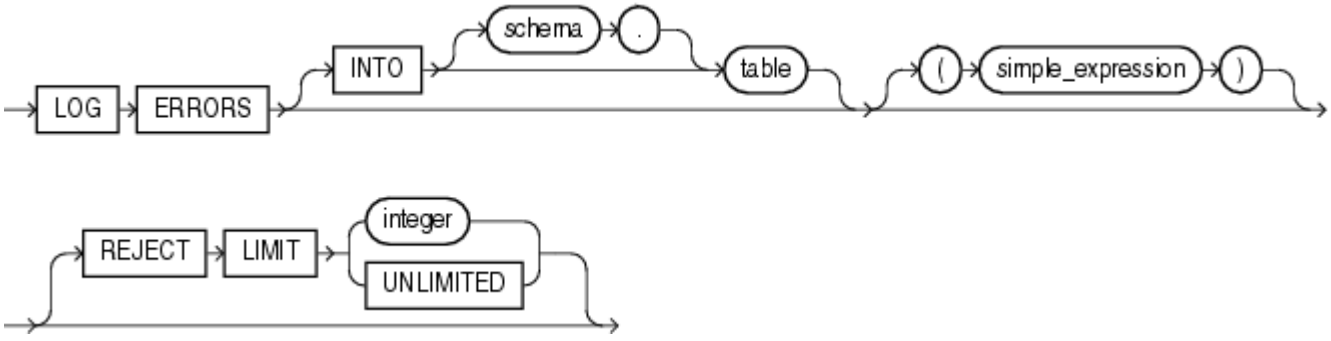


merge_insert_clause::=



where_clause::=

error_logging_clause::=



セマンティクス

INTO句

INTO句を使用すると、更新または挿入の対象となる表またはビューを指定できます。データをビューにマージする場合、そのビューは更新可能であることが必要です。詳細は、「更新可能なビューのノート」を参照してください。

ターゲット・ビューの制限事項

INSTEAD OFトリガーが定義されたターゲット・ビューは指定できません。

USING句

USING句を使用して、更新または挿入の元となるソースを指定します。

ON句

ON句を使用すると、MERGEの更新操作または挿入操作の条件を指定できます。対象となる表の中で検索条件が真となる各行は、ソースの対応するデータに基づいて行が更新されます。どの行も条件が真とならない場合、ソースの対応する行に基づいて対象となる表に行が挿入されます。

merge_update_clause

merge_update_clauseを指定すると、対象となる表またはビューの新しい列値を指定できます。ON句の条件が真となる場合、更新が実行されます。更新が実行されると、対象となる表に定義されているすべての更新トリガーがアクティブになります。

where_clauseを指定すると、指定した条件が真の場合のみに更新操作が実行されるようにできます。条件には、データソースまたは対象となる表を参照できます。条件が真ではない場合、行を表に挿入する際に更新操作がスキップされます。

DELETE where_clauseを指定すると、表の移入中または更新中にその表内のデータをクリーンアップできます。この句によって処理される行は、マージ操作によって更新される対象の表内の行のみです。DELETE WHERE条件は、更新後の値を評価し、UPDATE SET ... WHERE条件によって評価された元の値は評価しません。更新先の表の行がDELETE条件を満たし、ON句によって定義された結合に含まれていない場合、その行は削除されます。更新先の表に定義されている削除トリガーが起動し、各行が削除されます。

この句は、単独で、またはmerge_insert_clauseとともに指定できます。merge_insert_clauseとともに指定する場合は、どちらを先に指定してもかまいません。

merge_update_clauseの制限事項

この句には、次の制限事項があります。

- 参照する列は、ON condition句で更新できません。
- ビューを更新する場合は、DEFAULTを指定できません。

merge_insert_clause

merge_insert_clauseを指定すると、ON句の条件が偽となる場合に対象となる表の列に挿入する値を指定できます。挿入が実行されると、対象となる表に定義されているすべての挿入トリガーがアクティブになります。INSERTキーワードの後に列リストを指定しない場合、対象となる表内の列数は、VALUES句内の値の数と一致している必要があります。

すべてのソース行を表に挿入するには、ON句の条件に定数フィルタ条件を使用します。定数フィルタ条件の一例はON(0=1)です。Oracle Databaseはこのような条件を認識すると、すべてのソース行を無条件に表に挿入します。この方法は、merge_update_clauseを省略することとは異なります。merge_update_clauseを省略しても、結合は実行されます。定数フィルタ条件を設定すると、結合は実行されません。

where_clauseを指定すると、指定した条件が真の場合のみに更新操作が実行されるようにできます。条件には、データソース列のみを参照できます。条件が真ではないすべての行に対する挿入操作はスキップされます。

merge_insert_clauseは、単独で、またはmerge_update_clauseとともに指定できます。merge_insert_clauseとともに指定する場合は、どちらを先に指定してもかまいません。

merge_insert_clauseの制限事項

ビューに挿入する場合は、DEFAULTを指定できません。

error_logging_clause

error_logging_clauseのMERGE文での動作は、INSERT文の場合と同じです。詳細は、「INSERT」文の「error_logging_clause」を参照してください。

関連項目:

エラー・ロギングによる表への挿入：例

例

表へのマージ：例

次の例では、サンプル・スキーマoeのデフォルト・ボーナスが100であるbonuses表を使用します。次に、oe.orders表のsales_rep_id列に基づいて、販売実績があったすべての従業員をbonuses表に挿入します。最終的に、人事部門マネージャが、給与が$8000以下の従業員にボーナスを支給することを決定します。販売実績がなかった従業員には、給与の1%がボーナスとして支給されます。販売実績があった従業員には、給与の1%がボーナスに加算されて支給されます。MERGE文は、これらの変更を1行で実装します。

```
CREATE TABLE bonuses (employee_id NUMBER, bonus NUMBER DEFAULT 100);
INSERT INTO bonuses(employee_id)
   (SELECT e.employee_id FROM hr.employees e, oe.orders o
   WHERE e.employee_id = o.sales_rep_id
   GROUP BY e.employee_id);
SELECT * FROM bonuses ORDER BY employee_id;
EMPLOYEE_ID      BONUS
----------- ----------
        153        100
        154        100
        155        100
        156        100
        158        100
        159        100
        160        100
        161        100
        163        100
MERGE INTO bonuses D
   USING (SELECT employee_id, salary, department_id FROM hr.employees
   WHERE department_id = 80) S
   ON (D.employee_id = S.employee_id)
   WHEN MATCHED THEN UPDATE SET D.bonus = D.bonus + S.salary*.01
     DELETE WHERE (S.salary > 8000)
   WHEN NOT MATCHED THEN INSERT (D.employee_id, D.bonus)
     VALUES (S.employee_id, S.salary*.01)
     WHERE (S.salary <= 8000);
SELECT * FROM bonuses ORDER BY employee_id;
EMPLOYEE_ID      BONUS
----------- ----------
        153        180
        154        175
        155        170
        159        180
        160        175
        161        170
```

```
        164         72
        165         68
        166         64
        167         62
        171         74
        172         73
        173         61
        179         62
```

条件付き挿入と更新: 例

次の例では、MERGE文を使用して、条件に従って表データを挿入し更新します。

次の文は、people_sourceとpeople_targetという名前の2つの表を作成し、名前を移入します。

```
CREATE TABLE people_source (
  person_id  INTEGER NOT NULL PRIMARY KEY,
  first_name VARCHAR2(20) NOT NULL,
  last_name  VARCHAR2(20) NOT NULL,
  title      VARCHAR2(10) NOT NULL
);
CREATE TABLE people_target (
  person_id  INTEGER NOT NULL PRIMARY KEY,
  first_name VARCHAR2(20) NOT NULL,
  last_name  VARCHAR2(20) NOT NULL,
  title      VARCHAR2(10) NOT NULL
);
INSERT INTO people_target VALUES (1, 'John', 'Smith', 'Mr');
INSERT INTO people_target VALUES (2, 'alice', 'jones', 'Mrs');
INSERT INTO people_source VALUES (2, 'Alice', 'Jones', 'Mrs.');
INSERT INTO people_source VALUES (3, 'Jane', 'Doe', 'Miss');
INSERT INTO people_source VALUES (4, 'Dave', 'Brown', 'Mr');
COMMIT;
```

次の文は、person_id列を使用してpeople_targetとpeople_sourceの内容を比較します。people_source表に一致するものがあると、people_target表の値が更新されます。

```
MERGE INTO people_target pt
USING people_source ps
ON    (pt.person_id = ps.person_id)
WHEN MATCHED THEN UPDATE
  SET pt.first_name = ps.first_name,
      pt.last_name = ps.last_name,
      pt.title = ps.title;
```

次の文は、people_target表の内容を表示し、ロールバックを実行します。

```
SELECT * FROM people_target;
PERSON_ID FIRST_NAME            LAST_NAME            TITLE
---------- -------------------- -------------------- ----------
         1 John                 Smith                Mr
         2 Alice                Jones                Mrs.
ROLLBACK;
```

次の文は、person_id列を使用してpeople_target表とpeople_source表の内容を比較します。people_source表に一致するものがある場合にのみ、people_target表の値が更新されます。

```
MERGE INTO people_target pt
USING people_source ps
ON    (pt.person_id = ps.person_id)
WHEN NOT MATCHED THEN INSERT
  (pt.person_id, pt.first_name, pt.last_name, pt.title)
  VALUES (ps.person_id, ps.first_name, ps.last_name, ps.title);
```

次の文は、people_target表の内容を表示し、ロールバックを実行します。

```
SELECT * FROM people_target;
PERSON_ID FIRST_NAME            LAST_NAME            TITLE
--------- -------------------- -------------------- ----------
         1 John                 Smith                Mr
         2 alice                jones                Mrs
         3 Jane                 Doe                  Miss
         4 Dave                 Brown                Mr
ROLLBACK;
```

次の文は、person_id列を使用してpeople_target表とpeople_source表の内容を比較し、people_target表で条件に従ってデータを挿入し更新します。people_source表の一致する行ごとに、people_source表の値を使用してpeople_target表の値が更新されます。people_source表の一致しない行は、people_target表に追加されます。

```
MERGE INTO people_target pt
USING people_source ps
ON    (pt.person_id = ps.person_id)
WHEN MATCHED THEN UPDATE
  SET pt.first_name = ps.first_name,
      pt.last_name = ps.last_name,
      pt.title = ps.title
WHEN NOT MATCHED THEN INSERT
  (pt.person_id, pt.first_name, pt.last_name, pt.title)
  VALUES (ps.person_id, ps.first_name, ps.last_name, ps.title);
```

次の文は、people_target表の内容を表示し、ロールバックを実行します。

```
SELECT * FROM people_target;
PERSON_ID FIRST_NAME     LAST_NAME           TITLE
--------- -------------   ------------------ ----------
         1 John                 Smith                Mr
         2 Alice                Jones                Mrs.
         3 Jane                 Doe                    Miss
         4 Dave                 Brown                Mr
ROLLBACK;
```

次の文は、person_id列を使用してpeople_target表とpeople_source表を比較します。person_idが一致すると、people_source表の値を使用して、people_target表で対応する行が更新されます。DELETE句は、titleが'Mrs.'であるpeople_targetのすべての値を削除します。person_idが一致しない場合は、people_source表の行がpeople_target表に追加されます。WHERE句を指定しているので、titleが'Mr'である値のみがpeople_target表に追加されます。

```
MERGE INTO people_target pt
USING people_source ps
ON    (pt.person_id = ps.person_id)
WHEN MATCHED THEN UPDATE
  SET pt.first_name = ps.first_name,
      pt.last_name = ps.last_name,
      pt.title = ps.title
  DELETE where pt.title  = 'Mrs.'
WHEN NOT MATCHED THEN INSERT
  (pt.person_id, pt.first_name, pt.last_name, pt.title)
  VALUES (ps.person_id, ps.first_name, ps.last_name, ps.title)
  WHERE ps.title = 'Mr';
```

次の文は、people_target表の内容を表示し、ロールバックを実行します。

```
SELECT * FROM people_target;
PERSON_ID FIRST_NAME            LAST_NAME            TITLE
--------- -------------------- -------------------- ----------
```

```
         1 John                 Smith                Mr
         4 Dave                 Brown                Mr
ROLLBACK;
```

アプリケーションからの入力の処理

通常、アプリケーションでは、新しい行をINSERTするか、既存の行をUPDATEするかを決定するために、最初に行の存在をチェックする必要があります。MERGE文では、USING文内のバインド変数をソースとして使用できるため、このようなチェックが不要になります。

次の文は、バインド変数を使用して新しい行をpeople_target表に挿入する方法を示しています。

```
var person_id  NUMBER;
var first_name VARCHAR2(20);
var last_name  VARCHAR2(20);
var title      VARCHAR2(10);
exec :person_id := 3;
exec :first_name := 'Gerald';
exec :last_name := 'Walker';
exec :title := 'Mr';
MERGE INTO people_target pt
   USING (SELECT :person_id  AS person_id,
                 :first_name AS first_name,
                 :last_name  AS last_name,
                 :title      AS title FROM DUAL) ps
   ON (pt.person_id = ps.person_id)
WHEN MATCHED THEN UPDATE
SET pt.first_name = ps.first_name,
    pt.last_name = ps.last_name,
    pt.title = ps.title
WHEN NOT MATCHED THEN INSERT
    (pt.person_id, pt.first_name, pt.last_name, pt.title)
    VALUES (ps.person_id, ps.first_name, ps.last_name, ps.title);
```

次の文は、people_target表の内容を表示し、ロールバックを実行します。

```
SELECT * FROM people_target;
 PERSON_ID FIRST_NAME           LAST_NAME            TITLE
---------- -------------------- -------------------- ----------
         1   John                Smith                Mr
         2   alice               jones                Mrs
         3   Gerald              Walker               Mr
ROLLBACK;
```

次の文は、people_target内の既存の行を更新するためのバインド変数の使用を示しています。MERGE文は、新しい行の挿入に使用した文と同じであることに注意してください。

```
var person_id  NUMBER;
var first_name VARCHAR2(20);
var last_name  VARCHAR2(20);
var title      VARCHAR2(10);
exec :person_id := 2;
exec :first_name := 'Alice';
exec :last_name := 'Jones';
exec :title := 'Mrs';
MERGE INTO people_target pt
   USING (SELECT :person_id  AS person_id,
                 :first_name AS first_name,
                 :last_name  AS last_name,
                 :title      AS title FROM DUAL) ps
   ON (pt.person_id = ps.person_id)
WHEN MATCHED THEN UPDATE
SET pt.first_name = ps.first_name,
```

```
    pt.last_name = ps.last_name,
    pt.title = ps.title
WHEN NOT MATCHED THEN INSERT
    (pt.person_id, pt.first_name, pt.last_name, pt.title)
    VALUES (ps.person_id, ps.first_name, ps.last_name, ps.title);
```

次の文は、people_target表の内容を表示し、ロールバックを実行します。

```
SELECT * FROM people_target;
PERSON_ID FIRST_NAME            LAST_NAME            TITLE
--------- -------------------- -------------------- ----------
         1   John                 Smith              Mr
         2   Alice                Jones              Mrs
ROLLBACK;
```

NOAUDIT (従来型監査)

この項では、従来型監査用のNOAUDIT文について説明します。これは、Oracle Database 12cより前のリリースで使用されているものと同じ監査機能です。

Oracle Database 12cから、高度な監査機能を完備した統合監査が導入されました。下位互換性を確保するために、従来型監査も引き続きサポートされています。ただし、既存の監査設定から新しい統合監査ポリシー構文への移行を計画することをお薦めします。新しい監査要件については、新しい統合監査を使用してください。従来型監査は、将来のメジャー・リリースでサポート対象外になる可能性があります。

関連項目:

統合監査用のNOAUDIT文については、「NOAUDIT (統合監査)」を参照してください。

目的

NOAUDIT文を使用すると、AUDIT文によって有効になった監査操作を停止できます。

NOAUDIT文は先に発行したAUDIT文と同じ構文である必要があります。また、NOAUDIT文は、その特定のAUDIT文のみを無効にします。たとえば、1番目のAUDIT文Aは特定のユーザーに対する監査を有効にするものとします。2番目の文Bが、すべてのユーザーに対して監査を有効にします。すべてのユーザーに対して監査を無効にするNOAUDIT文Cは、文Bを無効にします。ただし、文Aは無効にされず、文Aが指定したユーザーの監査は継続されます。

関連項目:

AUDIT (従来型監査)

前提条件

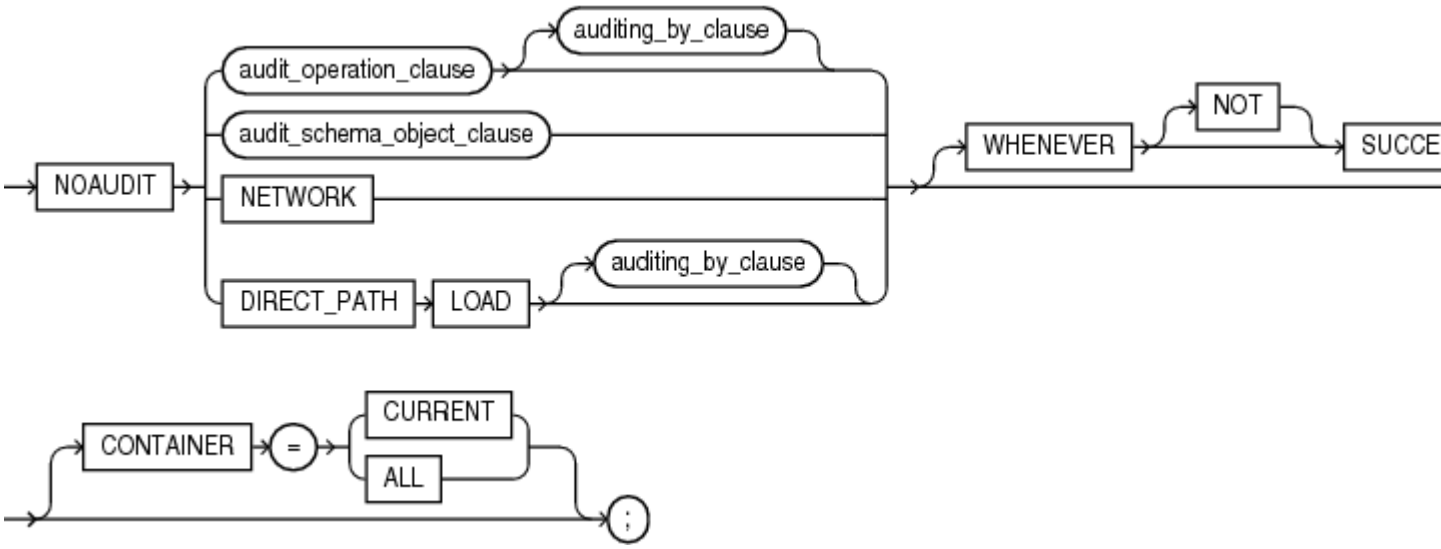
SQL文の監査を停止するには、AUDIT SYSTEMシステム権限が必要です。

スキーマ・オブジェクトの監査を停止するには、監査を停止するオブジェクトの所有者である必要があります。そうでない場合は、AUDIT ANYシステム権限が必要です。また、監査の対象として選択するオブジェクトがディレクトリの場合、自分が作成したディレクトリであっても、AUDIT ANYシステム権限が必要です。

CONTAINER句を指定する場合は、マルチテナント・コンテナ・データベース(CDB)に接続している必要があります。CONTAINER = ALLを指定する場合は、現在のコンテナがルートである必要があります。また、共通に付与されているAUDIT SYSTEM権限(SQL文の発行の監査を停止する場合)または共通に付与されているAUDIT ANY権限(スキーマ・オブジェクトに対する操作の監査を停止する場合)が必要です。CONTAINER = CURRENTを指定する場合は、現在のコンテナがプラガブル・データベース(PDB)である必要があります。また、ローカルに付与されているAUDIT SYSTEM権限(SQL文の発行の監査を停止する場合)またはローカルに付与されているAUDIT ANY権限(スキーマ・オブジェクトに対する操作の監査を停止する場合)が必要です。

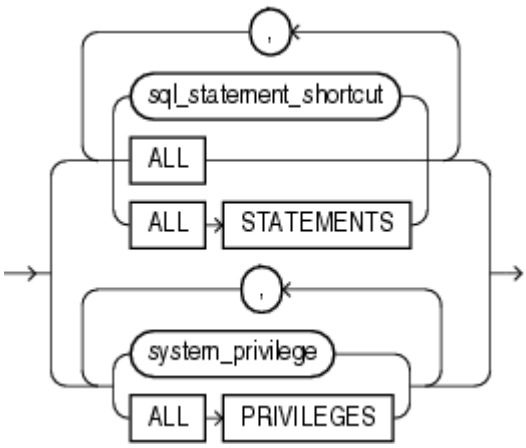
構文

noaudit::=

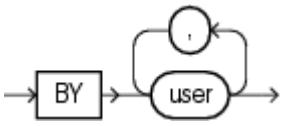


(audit_operation_clause::=、auditing_by_clause::=、audit_schema_object_clause::=)

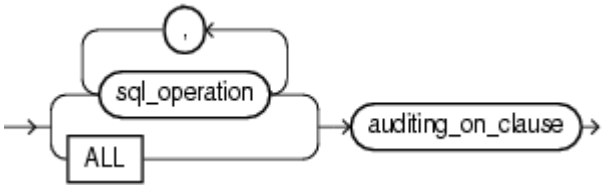
audit_operation_clause::=



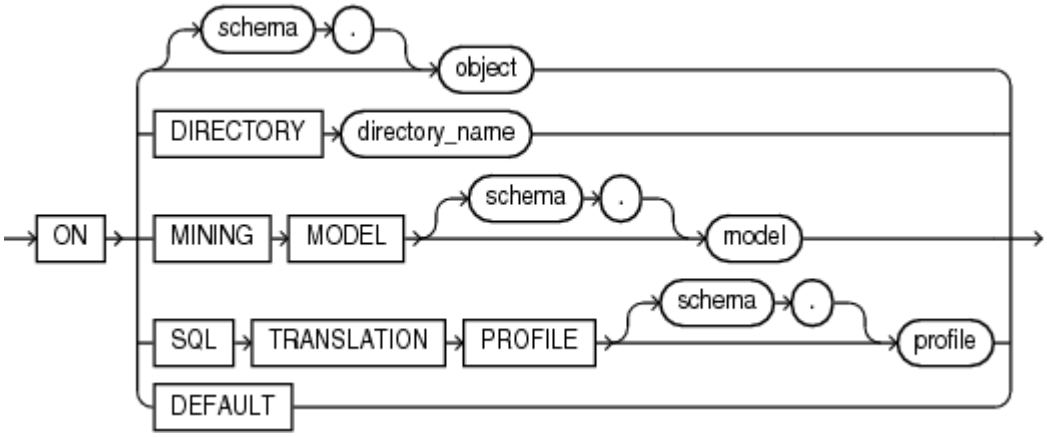
auditing_by_clause::=



audit_schema_object_clause::=



auditing_on_clause::=



セマンティクス

audit_operation_clause

`audit_operation_clause`を使用すると、特定のSQL文の監査を停止できます。

statement_option

`sql_statement_shortcut`には、監査を停止するSQL文のショートカットを指定します。SQL文のショートカットおよびショートカットによって監査されるSQL文の詳細は、表12-1および表12-2を参照してください。

ALL

`ALL`を指定すると、`AUDIT ALL ...`文の発行によって現在監査されているすべての文オプションの監査を停止できます。この句を使用して、発行済の`AUDIT ALL STATEMENTS ...`文を元に戻すことはできません。

ALL STATEMENTS

`ALL STATEMENTS`を指定すると、発行済の`AUDIT ALL STATEMENTS ...`文を元に戻すことができます。この句を使用して、発行済の`AUDIT ALL ...`文を元に戻すことはできません。

system_privilege

`system_privilege`では、監査を停止するシステム権限を指定します。システム権限および各システム権限によって許可される文については、表18-1を参照してください。

ALL PRIVILEGES

`ALL PRIVILEGES`を指定すると、現在監査されているすべてのシステム権限の監査を停止できます。

auditing_by_clause

指定したユーザーのそれ以降のセッションで発行されるSQL文の監査のみを停止する場合は、`auditing_by_clause`を使用します。この句を指定しない場合、すべてのユーザー文の監査が停止されます。ただし、「`WHENEVER SUCCESSFUL`」で説明する状況は除きます。

audit_schema_object_clause

`audit_schema_object_clause`を使用すると、特定のデータベース・オブジェクトの監査を停止できます。

sql_operation

`sql_operation`の場合、`ON`句で指定したオブジェクトへの監査を停止する操作の種類を指定します。これらのオプションのリストは、表12-3を参照してください。

ALL

ALLをショートカットに指定することは、オブジェクト・タイプに適用できるSQL操作をすべて指定することと同じです。

auditing_on_clause

auditing_on_clauseを使用すると、監査を停止する特定のスキーマ・オブジェクトを指定できます。

- オブジェクトには、表、ビュー、順序、ストアド・プロシージャ、ファンクション、パッケージ、マテリアライズド・ビューまたはライブラリのオブジェクト名を指定します。objectをschemaで修飾しなかった場合、そのオブジェクトは自分のスキーマ内にあるとみなされます。特定のスキーマ・オブジェクトの監査については、 「AUDIT (従来型監査)」を参照してください。
- DIRECTORYでは、監査を停止するディレクトリ名を指定できます。
- SQL TRANSLATION PROFILE句では、監査を停止するSQL翻訳プロファイルを指定できます。
- DEFAULTを指定して、オブジェクトを作成した後に、特定のオブジェクト・オプションをデフォルト・オブジェクト・オプションとして削除します。

NETWORK

この句を使用すると、データベース・リンクの使用とログインの監査を停止できます。

DIRECT_PATH LOAD

この句を使用すると、SQL*Loaderのダイレクト・パス・ロードの監査を停止できます。

WHENEVER [NOT] SUCCESSFUL

WHENEVER SUCCESSFULを指定すると、正常に実行されたスキーマ・オブジェクトに対するSQL文および操作の監査のみを停止できます。

WHENEVER NOT SUCCESSFULを指定すると、Oracle DatabaseエラーとなったSQL文および操作の監査のみが停止されます。

この句を指定しない場合、正常に実行されたかどうかにかかわらず、すべての文および操作の監査が停止されます。

CONTAINER句

CONTAINER句を使用すると、NOAUDITコマンドの有効範囲を指定できます。

- CONTAINER = CURRENTを指定すると、接続先のPDBで監査を停止できます。auditing_by_clauseを指定する場合、userは、共通ユーザーまたは現在のPDBのローカル・ユーザーにする必要があります。auditing_on_clauseを指定する場合、オブジェクトは現在のPDBのローカル・オブジェクトにする必要があります。
- CONTAINER = ALLを指定すると、CDB全体で監査を停止できます。auditing_by_clauseを指定する場合、userは、共通ユーザーにする必要があります。auditing_by_clauseを指定しないと、各PDBのすべての共通ユーザーとすべてのローカル・ユーザーに対する監査が停止されます。auditing_on_clauseを指定する場合、オブジェクトは共通オブジェクトにする必要があります。

この句を省略すると、CONTAINER = CURRENTがデフォルトになります。

例

ロールに関連するSQL文の監査の停止: 例

ロールを作成または削除するすべてのSQL文の監視を選択していた場合は、次の文を発行して、このような文の監査を停止できます。

```
NOAUDIT ROLE;
```

特定のユーザーが所有するオブジェクトに対する更新または問合せの監査の停止: 例

ユーザーhrとoeによって発行された、表に対する問合せまたは更新を行う任意の文の監査を選択していた場合は、次の文を発行して、hrによる問合せの監視を停止できます。

```
NOAUDIT SELECT TABLE BY hr;
```

この結果、ユーザーhrの問合せの監査のみが停止されます。oeの問合せと更新、およびhrの更新の監査は継続されます。

特定のオブジェクト権限によって許可された文の監査の停止: 例

DELETE ANY TABLEシステム権限によって許可されたすべての文の監査を停止するには、次の文を発行します。

```
NOAUDIT DELETE ANY TABLE;
```

特定のオブジェクトに対する問合せの監査の停止: 例

スキーマhr内のemployees表に問い合せるすべてのSQL文の監査を選択していた場合は、次の文を発行して、この問合せの監査を停止できます。

```
NOAUDIT SELECT
   ON hr.employees;
```

正常に実行される問合せの監査の停止: 例

次の文を発行して、正常に実行される問合せの監査を停止できます。

```
NOAUDIT SELECT
   ON hr.employees
   WHENEVER SUCCESSFUL;
```

この文は、正常に終了した問合せの監査のみ停止します。Oracle Databaseエラーが発生した問合せの監査は継続されます。

NOAUDIT (統合監査)

この項では、統合監査用のNOAUDIT文について説明します。この種類の監査は、Oracle Database 12cで新たに導入されたもので、完全かつ高度な監査機能を提供します。統合監査の詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

目的

NOAUDIT文を使用すると、次の操作を実行できます。

- すべてのユーザーまたは特定のユーザーに対して、統合監査ポリシーを無効化します。
- コンテキスト属性の値を監査レコードから除外します。

この文で実行される操作は、現行のセッションではなく、それ以降のユーザー・セッションで有効になります。

関連項目:

- AUDIT (統合監査)
- CREATE AUDIT POLICY (統合監査)
- ALTER AUDIT POLICY (統合監査)
- DROP AUDIT POLICY (統合監査)

前提条件

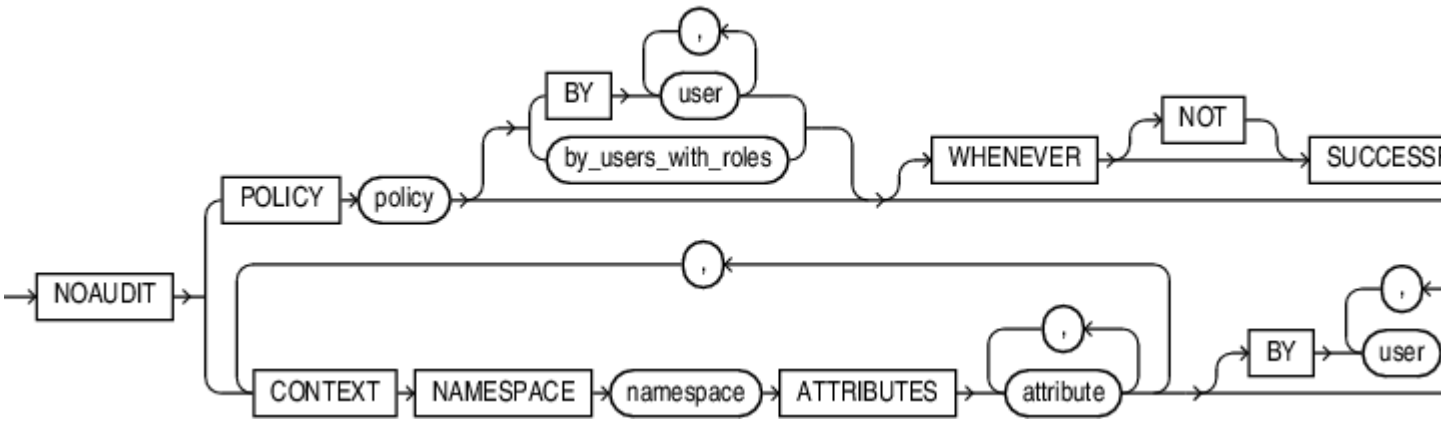
AUDIT SYSTEMシステム権限、またはAUDIT_ADMINロールが必要になります。

マルチテナント・コンテナ・データベース(CDB)に接続している場合、共通の統合監査ポリシーを無効化するには、現在のコンテナがルートである必要があります。また、共通に付与されているAUDIT SYSTEM権限またはAUDIT_ADMIN共通ロールが必要です。ローカルの統合監査ポリシーを無効化するには、現在のコンテナが、その監査ポリシーが作成されたコンテナである必要があります。また、共通に付与されているAUDIT SYSTEM権限またはAUDIT_ADMIN共通ロールを保有しているか、そのコンテナでローカルに付与されているAUDIT SYSTEM権限またはAUDIT_ADMINローカル・ロールを保有している必要があります。

CDBに接続しているときにNOAUDIT CONTEXT ...文を指定するには、共通に付与されているAUDIT SYSTEM権限またはAUDIT_ADMIN共通ロールを保有しているか、現在のセッションのコンテナでローカルに付与されているAUDIT SYSTEM権限またはAUDIT_ADMINローカル・ロールを保有している必要があります。

構文

unified_noaudit::=



by_users_with_roles::=

セマンティクス

policy

無効化する統合監査ポリシーの名前を指定します。

すべての統合監査ポリシーの説明を検索するには、AUDIT_UNIFIED_POLICIESビューを問い合わせます。すべての有効な統合監査ポリシーの説明を検索するには、AUDIT_UNIFIED_ENABLED_POLICIESビューを問い合わせます。

関連項目:

AUDIT_UNIFIED_POLICIESおよびAUDIT_UNIFIED_ENABLED_POLICIESビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

CONTEXT句

CONTEXT句を指定すると、監査レコードのコンテキスト属性の値を除外できます。

- namespaceには、コンテキスト・ネームスペースを指定します。
- attributeには、監査レコードから除外する値を持つ1つ以上のコンテキスト属性を指定します。

現在のコンテナがCDBのルートであるときにCONTEXT句を指定すると、ルートで実行されたイベントについての監査レコードにのみ、コンテキスト属性の値が含まれるようになります。オプションのBY句を指定する場合、userは共通ユーザーにする必要があります。

現在のコンテナがプラガブル・データベース(PDB)であるときにCONTEXT句を指定すると、PDBで実行されたイベントについての監査レコードにのみ、コンテキスト属性の値が含まれるようになります。オプションのBY句を指定する場合、userは、共通ユーザーまたはそのPDBのローカル・ユーザーにする必要があります。

監査証跡に取得されるように構成されている、アプリケーション・コンテキストの属性を調べるには、AUDIT_UNIFIED_CONTEXTSビューを問い合わます。

関連項目:

AUDIT_UNIFIED_CONTEXTSビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

BY

BY句は、NOAUDIT POLICYおよびNOAUDIT CONTEXT文に対して指定できます。

NOAUDIT POLICY ... BY

BY句の動作は、policyがすべてのユーザーと特定のユーザーのどちらに対して有効になっているかによって異なります。

- policyがすべてのユーザーに対して有効になっている場合、BY句を省略することによって、policyをすべてのユーザーに対して無効化できます。BY句を指定した場合、NOAUDIT POLICY文による影響はありません。
- policyが(AUDIT POLICY ... BY ...文を使用することによって)1人以上のユーザーに対して有効になっている場

合、次の操作を実行できます。

- BY句の後にpolicyを無効化するユーザーを指定することによって、1人以上のユーザーに対してpolicyを無効化できます
- BY句の後にpolicyが有効になっているすべてのユーザーを指定することによって、policyを完全に無効化できます

BY句を指定しなかった場合、NOAUDIT POLICY文による影響はありません。

- policyが(AUDIT POLICY ... EXCEPT ...文を使用することによって)特定のユーザーを除くすべてのユーザーに対して有効になっている場合、BY句を省略することによって、policyをすべてのユーザーに対して無効化できます。BY句を指定した場合、NOAUDIT POLICY文による影響はありません。

policyが共通の統合監査ポリシーの場合、userは共通ユーザーにする必要があります。policyがローカルの統合監査ポリシーの場合、userは共通ユーザーまたは接続先のコンテナのローカル・ユーザーにする必要があります。

NOAUDIT CONTEXT ... BY

BY句の動作は、attributeがすべてのユーザーと特定のユーザーのどちらの監査レコードに含まれるように構成されているかによって異なります。

- attributeがすべてのユーザーの監査レコードに含まれるように構成されている場合、BY句を省略することによって、attributeをすべてのユーザーの監査レコードから除外できます。BY句を指定した場合、NOAUDIT CONTEXT文による影響はありません。
- attributeが特定のユーザーの監査レコードに含まれるように構成されている場合、BY句の後にattributeを除外するユーザーを指定することによって、1人以上のユーザーのattributeを除外できます。BY句を指定しなかった場合、NOAUDIT CONTEXT文による影響はありません。

by_users_with_roles

この句を指定すると、指定したロールが直接付与されているユーザーに対してのみ、policyを無効にできます。後で追加ユーザーにいずれかのロールを付与した場合、ポリシーはそのユーザーに対して自動的に無効になります。

CDBに接続されているときに、policyが共通の統合監査ポリシーである場合、roleは共通ロールである必要があります。policyがローカルの統合監査ポリシーの場合、ロールは共通ロールまたは接続先のコンテナのローカル・ロールにする必要があります。

例

次の例では、CREATE AUDIT POLICYの「例」で作成し、AUDITの「例」で有効にした統合監査ポリシーを無効にします。

すべてのユーザーの統合監査ポリシーの無効化: 例

統合監査ポリシーtable_polがすべてのユーザーに対して有効になっているとします。次の文では、すべてのユーザーに対してtable_polを無効化します。

```
NOAUDIT POLICY table_pol;
```

次の文では、行が戻されません。これは、すべてのユーザーに対してtable_polが無効になっていることを示しています。

```
SELECT *
  FROM audit_unified_enabled_policies
  WHERE policy_name = 'TABLE_POL';
```

特定のユーザーの統合監査ポリシーの無効化: 例

次の問合せで表示されるように、統合監査ポリシーdml_polがユーザーhrとshに対して有効になっているとします。

```
SELECT policy_name, enabled_option, entity_name
  FROM audit_unified_enabled_policies
  WHERE policy_name = 'DML_POL'
  ORDER BY entity_name;

POLICY_NAME  ENABLED_OPTION  ENTITY_NAME
-----------  -----------  ---------
DML_POL      BY           HR
DML_POL      BY           SH
```

次の文では、ユーザーhrに対してdml_polを無効化します。

```
NOAUDIT POLICY dml_pol BY hr;
```

次の文では、dml_polがユーザーshに対してのみ有効になっていることを確認します。

```
SELECT policy_name, enabled_option, entity_name
  FROM audit_unified_enabled_policies
  WHERE policy_name = 'DML_POL';

POLICY_NAME  ENABLED_OPTION  ENTITY_NAME
-----------  -----------  ---------
DML_POL      BY           SH
```

次の文では、ユーザーshに対してdml_polを無効化します。

```
NOAUDIT POLICY dml_pol BY sh;
```

次の文では、行が戻されません。これは、すべてのユーザーに対してdml_polが無効になっていることを示しています。

```
SELECT *
  FROM audit_unified_enabled_policies
  WHERE policy_name = 'DML_POL';
```

監査レコードのコンテキスト属性値の除外: 例

次の文では、ネームスペースUSERENVの属性CURRENT_USERとDB_NAMEの値を、ユーザーhrについてのすべての監査レコードから除外するようにデータベースに指示します。

```
NOAUDIT CONTEXT NAMESPACE userenv
  ATTRIBUTES current_user, db_name
  BY hr;
```

PURGE

目的

PURGE文を使用すると、次の操作を実行できます。

- ごみ箱から表または索引を削除し、そのオブジェクトに関連付けられているすべての領域を解放します。
- 削除された表領域または表領域セットの一部または全部をごみ箱から削除します。
- ごみ箱全体を削除します。

ノート:

PURGE 文はロールバックできません。また、この文によって消去されたオブジェクトはリカバリできません。

ごみ箱の内容を参照するには、USER_RECYCLEBINデータ・ディクショナリ・ビューを問い合せます。かわりにRECYCLEBINシノニムを使用することもできます。次の2つの文は、同じ行を戻します。

```
SELECT * FROM RECYCLEBIN;
SELECT * FROM USER_RECYCLEBIN;
```

関連項目:

- ごみ箱の詳細、およびごみ箱内のオブジェクトのネーミング規則の詳細は、『Oracle Database管理者ガイド』を参照してください。
- 削除した表をごみ箱から取り出す方法については、「FLASHBACK TABLE」を参照してください。
- 削除した表をごみ箱に移動するかを制御するRECYCLEBIN初期化パラメータの使用の詳細は、『Oracle Databaseリファレンス』を参照してください。

前提条件

表を消去するには、その表が自分のスキーマ内にあるか、自分にDROP ANY TABLEシステム権限が付与されている、あるいはSYSDBAシステム権限が付与されている必要があります。

索引を消去するには、その索引が自分のスキーマ内にあるか、自分にDROP ANY INDEXシステム権限が付与されている、あるいはSYSDBAシステム権限が付与されている必要があります。

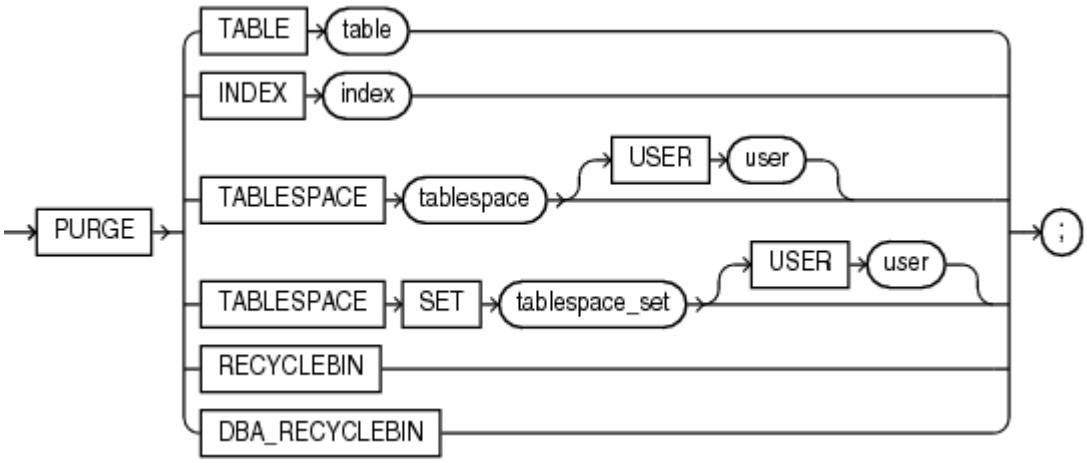
表領域または表領域セットを消去するには、自分にDROP TABLESPACEシステム権限が付与されているか、SYSDBAシステム権限が付与されている必要があります。

表領域セットを消去するには、SDBユーザーとしてシャード・カタログ・データベースに接続されている必要もあります。

PURGE DBA_RECYCLEBIN操作を実行するには、SYSDBAまたはPURGE DBA_RECYCLEBINシステム権限が必要です。

構文

purge::=



セマンティクス

TABLEまたはINDEX

ごみ箱内の消去する表または索引の名前を指定します。ユーザーが指定した元の名前か、オブジェクトの削除時にそのオブジェクトに割り当てられたシステム生成名を指定できます。

- ユーザー指定の名前を指定した場合、その名前を持つオブジェクトがごみ箱内に複数存在していると、ごみ箱内に最も長く存在しているオブジェクトが消去されます。
- ごみ箱内のオブジェクトのシステム生成名は一意です。そのため、システム生成名を指定すると、その名前を持つ特定のオブジェクトが消去されます。

表が消去されると、その表のすべての表パーティション、LOBとLOBパーティション、索引、およびその他の依存オブジェクトも消去されます。

TABLESPACEまたはTABLESPACE SET

この句を使用すると、ごみ箱から、指定した表領域または表領域セット内に存在するすべてのオブジェクトを消去できます。

USER user

この句を使用すると、指定したユーザーが、表領域または表領域セット内の領域を再利用できます。この操作は、特定のユーザーの、特定の表領域または表領域セットに対するディスク領域が割当て制限間近である場合に特に役立ちます。

RECYCLEBIN

この句を使用すると、現行のユーザーのごみ箱を空にできます。そのユーザーのごみ箱からすべてのオブジェクトが消去され、そのオブジェクトに関連付けられていたすべての領域が解放されます。

DBA_RECYCLEBIN

この句は、SYSDBAまたはPURGE DBA_RECYCLEBINシステム権限を持っている場合にのみ有効です。これを使用すると、システム全体のごみ箱からすべてのオブジェクトを削除できます。これは、各ユーザーのごみ箱を空にすることと同じです。この操作は、以前のリリースへの移行などの場合に役立ちます。

例

ごみ箱からのファイルの削除: 例

次の文は、ごみ箱から表testを削除します。複数のバージョンのtest表がごみ箱内に存在する場合、Oracle Databaseでは、ごみ箱内に最も長く存在しているものが削除されます。

```
PURGE TABLE test;
```

ごみ箱から削除する表のシステム生成名を判断するには、ごみ箱に対するSELECT文を発行します。そのオブジェクト名を使用して、次の文に類似した文を発行して表を削除できます。(システム生成名は、次の例に示すものとは異なります。)

```
PURGE TABLE RB$$33750$TABLE$0;
```

ごみ箱の内容の削除: 例

ごみ箱の内容全体を削除するには、次の文を発行します。

```
PURGE RECYCLEBIN;
```

RENAME

目的

ノート:

RENAME 文はロールバックできません。

RENAME文を使用すると、表、ビュー、順序またはプライベート・シノニムの名前を変更できます。

- 古いオブジェクトの整合性制約、索引および権限付与は、新しいオブジェクトに自動的に移行されます。
- 名前を変更した表を参照するビュー、シノニム、ストアド・プロシージャ、ストアド・ファンクションなど、名前を変更したオブジェクトに依存するオブジェクトはすべて無効になります。

関連項目:

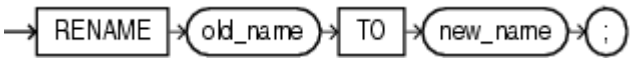
「CREATE SYNONYM」および「DROP SYNONYM」を参照してください。

前提条件

オブジェクトが自分のスキーマ内にある必要があります。

構文

rename::=



セマンティクス

old_name

既存の表、ビュー、順序またはプライベート・シノニムの名前を指定します。

new_name

既存のオブジェクトに割り当てる新しい名前を指定します。新しい名前は、同じネームスペース内の他のスキーマ・オブジェクトに使用されている名前以外にする必要があります。また、スキーマ・オブジェクトのネーミング規則に従って指定する必要があります。

オブジェクトの名前変更の制限事項

オブジェクトの名前変更には、次の制限事項があります。

パブリック・シノニムの名前は変更できません。そのかわり、該当するパブリック・シノニムを削除し、新しい名前で別のパブリック・シノニムを作成してください。

依存表または依存する有効なユーザー定義オブジェクト型を持つ型のシノニムの名前は変更できません。

関連項目:

データベース・オブジェクトのネーミング規則

例

データベース・オブジェクトの名前変更: 例

次の例では、サンプル表hr.departmentsのコピーを使用します。次の文は、表の名前をdepartments_newからemp_departmentsに変更します。

```
RENAME departments_new TO emp_departments;
```

この文では列名を直接変更できません。ただし、ALTER TABLE ... rename_column_clauseを使用すると、列の名前を変更できます。

関連項目:

rename_column_clause

列名を変更するもう1つの方法は、AS副問合せを指定したCREATE TABLE文とともに、RENAME文を使用する方法です。この方法は、単に列の名前を変更するのではなく、表の構造を変更する場合に有効です。次の文は、サンプル表hr.job_historyを再作成し、列の名前をdepartment_idからdept_idに変更します。

```
CREATE TABLE temporary
   (employee_id, start_date, end_date, job_id, dept_id)
AS SELECT
     employee_id, start_date, end_date, job_id, department_id
FROM job_history;
DROP TABLE job_history;
RENAME temporary TO job_history;
```

前述の例の場合、job_history表に定義されている整合性制約は失われます。これらの整合性制約は、ALTER TABLE文を使用して、新しいjob_history表に再定義する必要があります。

REVOKE

目的

REVOKE文を使用すると、次の操作を実行できます。

- ユーザーおよびロールからのシステム権限の取消し
- ユーザー、ロール、およびプログラム・ユニットからのロールの取消し。
- ユーザーまたはロールからの特定のオブジェクトに対するオブジェクト権限の取消し

Oracle Automatic Storage Managementのノート

AS SYSASMとして認証されたユーザーは、この文を使用して、現行のノードのOracle ASMパスワード・ファイル内のユーザーからシステム権限SYSASM、SYSOPERおよびSYSDBAを取り消すことができます。

エディション化可能なオブジェクトのノート:

エディション化可能なオブジェクトからオブジェクト権限を取り消すREVOKE操作を実行すると、オブジェクトが現行のエディションで実体化されます。エディションおよびエディション化可能なオブジェクトの詳細は、『Oracle Database開発ガイド』を参照してください。

関連項目:

- システム権限およびロールの付与については、「GRANT」を参照してください。
- それぞれのオブジェクト型に対するオブジェクト権限の詳細は、表18-2を参照してください。

前提条件

システム権限を取り消すには、Admin Option付きの権限が必要です。なお、GRANT ANY PRIVILEGEシステム権限を持っている場合は、権限を自由に取り消すことができます。

ユーザーのロールまたは別のロールを取り消すには、ADMIN OPTIONを使用してロールを直接付与するか、または付与するロールが自分で作成したロールである必要があります。なお、GRANT ANY ROLEシステム権限を持っている場合は、ロールを自由に取り消すことができます。

プログラム・ユニットのロールを取り消すには、ユーザーSYSまたはプログラム・ユニットのスキーマ所有者である必要があります。

オブジェクト権限を取り消すには、次のいずれかの条件が満たされている必要があります。

- 以前にそのユーザーまたはロールにそのオブジェクト権限を付与した。
- GRANT ANY OBJECT PRIVILEGEシステム権限を保有している。この場合は、オブジェクト所有者によって付与されたか、この所有者にかわってGRANT ANY OBJECT PRIVILEGEを持つユーザーによって付与されたあらゆるオブジェクト権限を取り消すことができます。ただし、With Grant Optionによって付与されたオブジェクト権限を取り消すことはできません。

関連項目:

GRANT ANY OBJECT PRIVILEGEを使用する操作の取消し: 例

REVOKE文によって取り消すことができる権限およびロールは、GRANT文によって直接付与されているものにかぎられます。この

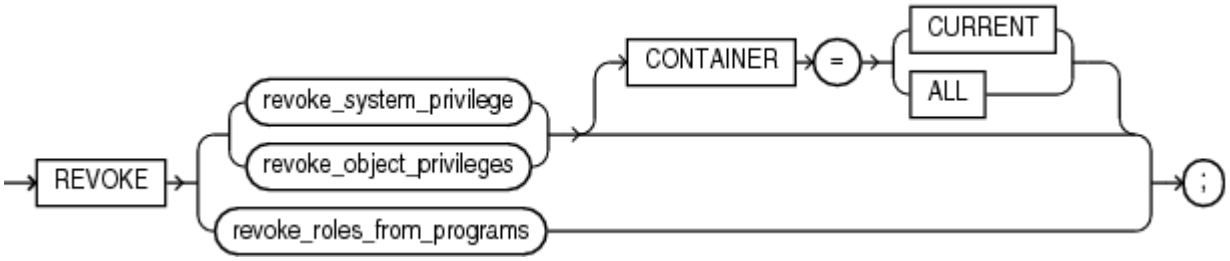
句では、次の権限を取り消すことはできません。

- 取消し側に付与されていない権限またはロール
- オペレーティング・システムを介して付与されているロールまたはオブジェクト権限
- ロールを介して取消し側に付与されている権限またはロール

CONTAINER句を指定する場合は、マルチテナント・コンテナ・データベース(CDB)に接続している必要があります。
CONTAINER = ALLを指定する場合は、現在のコンテナがルートである必要があります。

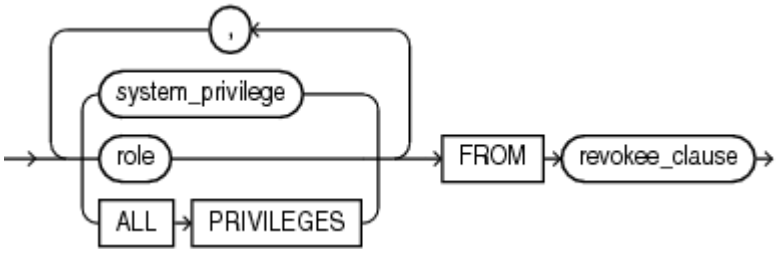
構文

revoke::=



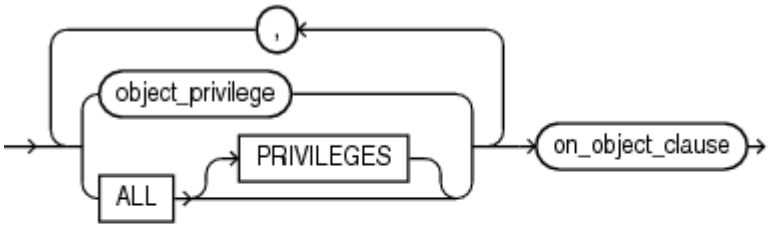
(revoke_system_privileges::=、revoke_object_privileges::=、revoke_roles_from_programs::=)

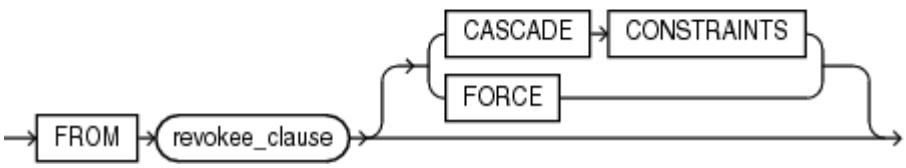
revoke_system_privileges::=



(revokee_clause::=)

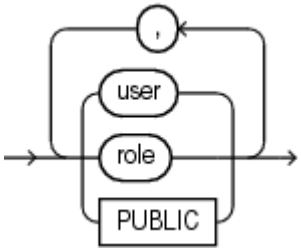
revoke_object_privileges::=



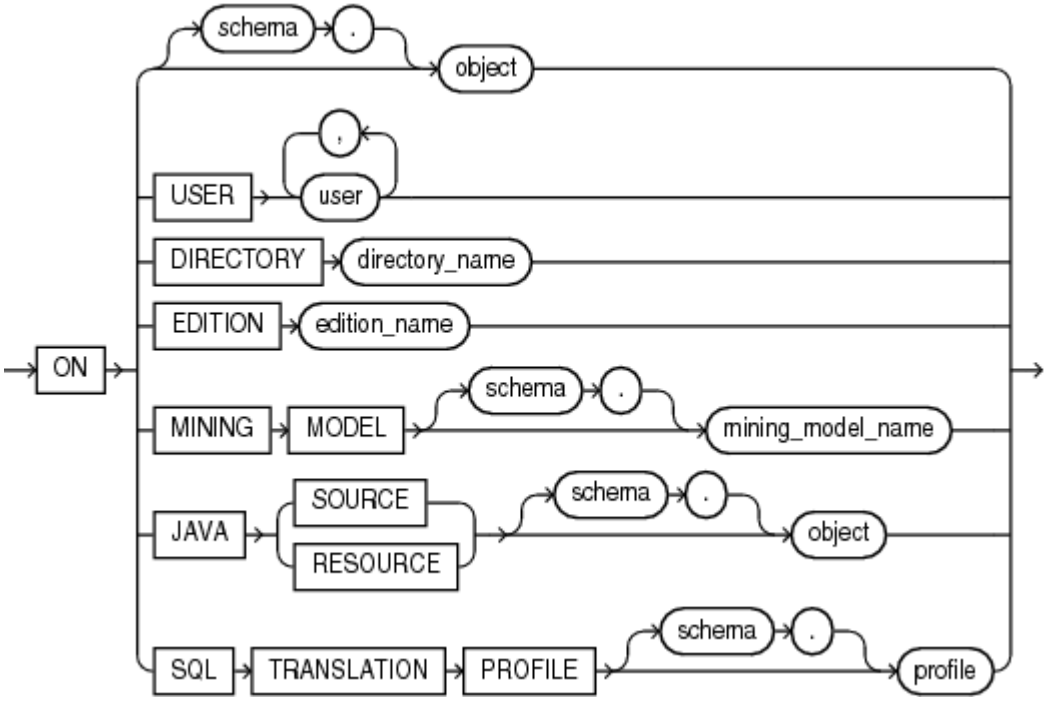


(on_object_clause::=、revokee_clause::=)

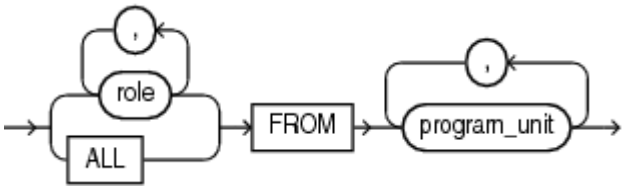
revokee_clause::=



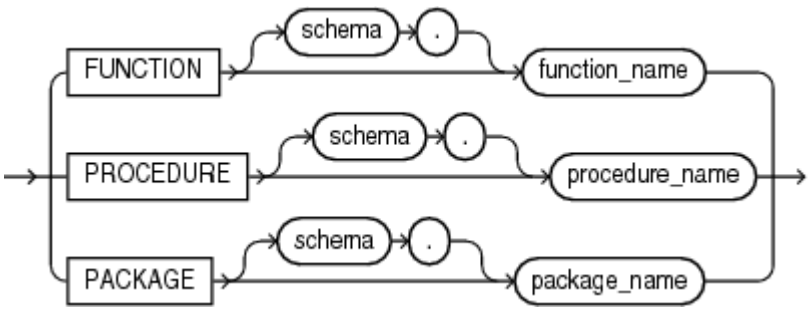
on_object_clause::=



revoke_roles_from_programs::=



program_unit::=



セマンティクス

revoke_system_privileges

システム権限を取り消すには、次の句を使用します。

system_privilege

取り消すシステム権限を指定します。システム権限のリストは、表18-1を参照してください。

ユーザーのシステム権限を取り消す場合、ユーザーの権限ドメインからその権限が取り消されます。この取消しはすぐに有効になるため、このユーザーはその権限を使用できなくなります。

ロールのシステム権限を取り消す場合、ロールの権限ドメインからその権限が取り消されます。この取消しはすぐに有効になるため、そのロールが使用可能となっている場合でも、この権限は使用できません。また、そのロールが付与されている他のユーザーは、ロールを使用可能にしても、その権限を使用できなくなります。

関連項目:

「ユーザーからのシステム権限の取消し: 例」および「ロールからのシステム権限の取消し: 例」を参照してください。

PUBLICのシステム権限を取り消す場合、PUBLICを介して権限を付与されている各ユーザーの権限ドメインからその権限が取り消されます。この取消しはすぐに有効になるため、ユーザーは権限を使用できなくなります。ただし、直接またはロールを介して権限が付与されているユーザーからは、権限を取り消すことはできません。

Oracle Databaseには、すべてのシステム権限を一度に指定できるショートカットがあります。ALL PRIVILEGESを指定すると、表18-1に示すすべてのシステム権限を取り消すことができます。

システム権限の取消しの制限事項

取り消す権限のリストにシステム権限を指定できるのは1回のみです。

role

取り消すロールを指定します。

ユーザーからロールを取り消すと、ユーザーによるロールの使用が禁止されます。そのロールが使用可能となっている場合に、ロールの権限ドメインの権限が使用可能な場合はその権限を使用できます。ただし、ユーザーが後からロールを使用可能にできません。

他のロールからロールを取り消すと、取消し側ロールの権限ドメインから、取り消されたロールの権限ドメインが削除されます。取り消されたロールの権限を付与され、そのロールの権限が使用可能になっているユーザーは、そのロールの権限が使用可能な間は、取り消されたロールの権限ドメインの権限を引き続き使用できます。ただし、取り消された権限を付与されていても、ロールの取消し操作の後でそれを使用可能にしたユーザーは、取り消されたロールの権限ドメインの権限を使用できません。

関連項目:

「ユーザーからのロールの取消し: 例」および「ロールからのロールの取消し: 例」を参照してください。

PUBLICのロールを取り消すと、PUBLICを介してロールが付与されているすべてのユーザーに対して、そのロールが使用禁止にされます。そのロールを使用可能としているユーザーは、権限ドメインの権限が使用可能であるかぎり、権限ドメインでその権限を引き続き使用できます。ただし、ユーザーが後からロールを使用可能にすることはできません。直接またはロールを介して権限が付与されているユーザーからは、ロールを取り消すことはできません。

システム・ロールの取消しの制限事項

取り消すロールのリストにシステム・ロールを指定できるのは1回のみです。事前定義されているロールの詳細は、『Oracle

Databaseセキュリティ・ガイド』を参照してください。

revokee_clause

revokee_clauseを使用すると、システム権限、ロールまたはオブジェクト権限を取り消すユーザーまたはロールを指定できます。

PUBLIC

PUBLICを指定すると、すべてのユーザーから権限またはロールを取り消すことができます。

revoke_object_privileges

オブジェクト権限を取り消すには、次の句を使用します。

object_privilege

取り消すオブジェクト権限を指定します。表18-2に、適用するオブジェクトのタイプによってカテゴリ分けされたオブジェクト権限を示します。

ノート:

操作は権限によって許可されます。権限を取り消すと、取り消されたユーザーは、その操作を実行できなくなります。ただし、複数のユーザーが、同じ権限を同一ユーザー、ロールまたは PUBLIC に対して付与している場合があります。権限受領者の権限ドメインの権限を取り消す場合、すべての権限付与者が権限を取り消す必要があります。権限を取り消さない権限付与者が 1 人でもいれば、権限受領者は引き続きその権限を使用できます。

ユーザーのオブジェクト権限を取り消すと、ユーザーの権限ドメインからその権限が取り消されます。この取消しはすぐに有効になるため、このユーザーはその権限を使用できなくなります。

- ユーザーがその権限を他のユーザーまたはロールに付与している場合、他のユーザーまたはロールの権限も取り消されます。
- 権限を使用するSQL文を記述したプロシージャ、ファンクションまたはパッケージが定義されているスキーマを持つユーザーのオブジェクト権限を取り消すと、それらのプロシージャ、ファンクションまたはパッケージは実行できなくなります。
- そのユーザーのスキーマにオブジェクトのビューが含まれる場合、ビューは無効になります。
- 参照整合性制約の定義権限を使用したユーザーのREFERENCESオブジェクト権限を取り消す場合、CASCADE CONSTRAINTS句も指定する必要があります。

ロールのオブジェクト権限を取り消すと、ロールの権限ドメインからその権限が取り消されます。この取消しはすぐに有効になるため、そのロールが使用可能となっている場合でも、この権限は使用できません。ロールが付与されている他のユーザーは、ロールを使用可能にした場合でも、権限を使用できなくなります。

PUBLICのオブジェクト権限を取り消すと、PUBLICを介して権限を付与されている各ユーザーの権限ドメインからその権限が取り消されます。この取消しはすぐに有効になるため、それらのすべてのユーザーは、その権限を使用できなくなります。ただし、直接またはロールを介して権限が付与されているユーザーからは、権限を取り消すことはできません。

ALL [PRIVILEGES]

ALLを指定すると、ユーザーまたはロールに付与されているすべてのオブジェクト権限を取り消すことができます。(キーワードPRIVILEGESはセマンティクスを明確にするためのものであり、指定は任意です。)

オブジェクトに権限が付与されていない場合、処理は行われず、エラーも戻りません。

オブジェクト権限の取消しの制限事項

取り消す権限のリストに権限を指定できるのは1回のみです。FROM句にユーザー、ロールまたはPUBLICを指定できるのは1回のみです。

関連項目:

「ユーザーからのオブジェクト権限の取消し: 例」、「PUBLICからのすべてのオブジェクト権限の取消し: 例」および「ユーザーからのすべてのオブジェクト権限の取消し: 例」を参照してください。

CASCADE CONSTRAINTS

この句は、REFERENCES権限またはALL [PRIVILEGES]を取り消すときにのみ適用されます。取消し側でREFERENCES権限(ALL [PRIVILEGES]を付与して明示的または暗黙的に付与された権限)を使用して定義した参照整合性制約を削除します。

関連項目:

CASCADE CONSTRAINTSによるオブジェクト権限の取消し: 例

FORCE

FORCEを指定すると、表または型に依存するユーザー定義型オブジェクトで、EXECUTEオブジェクト権限を取り消すことができます。表に依存するユーザー定義型オブジェクトでは、FORCEを使用してEXECUTEオブジェクト権限を取り消します。

FORCEを指定した場合、すべての権限が取り消され、すべての依存オブジェクトにINVALIDのマークが付けられ、依存表のデータにはアクセスできなくなります。また、すべての依存ファンクション索引にUNUSABLEのマークが付けられます。必要な型の権限を再付与した場合、表に対して再度妥当性チェックが行われます。

関連項目:

型依存性およびユーザー定義オブジェクト権限の詳細は、『Oracle Database概要』を参照してください。

on_object_clause

on_object_clauseを指定すると、権限を取り消すオブジェクトを識別できます。

object

オブジェクト権限を取り消すオブジェクトを指定します。取り消すことができるオブジェクトは次のとおりです。

- 表、ビュー、順序、プロシージャ、ストアド・ファンクション、パッケージまたはマテリアライズド・ビュー
- 表、ビュー、順序、プロシージャ、ストアド・ファンクション、パッケージ、マテリアライズド・ビューまたはユーザー定義型に対するシノニム
- ライブラリ、索引タイプまたはユーザー定義演算子

オブジェクトをschemaで修飾しなかった場合、そのオブジェクトは自分のスキーマ内にあるとみなされます。

関連項目:

ユーザーからの順序に対するオブジェクト権限の取消し: 例

システム権限のGrant Optionの有無にかかわらず、READまたはSELECTオブジェクト権限をマテリアライズド・ビューまたは表を含むマテリアライズド・ビューから取り消すと、そのマテリアライズド・ビューは無効になります。

システム権限のGRANT OPTIONの有無にかかわらず、マテリアライズド・ビューのマスター表のいずれかにおけるREADまたはSELECTオブジェクト権限を取り消すと、そのマテリアライズド・ビューとそれに含まれる表の両方またはマテリアライズド・ビューが無効になります。

ON USER

権限を取り消すデータベース・ユーザーを指定します。

関連項目:

ユーザーからのユーザーに対するオブジェクト権限の取消し: 例

ON DIRECTORY

権限を取り消すディレクトリ・オブジェクトの名前を指定します。directory_nameはスキーマ名で修飾できません。

関連項目:

「CREATE DIRECTORY」および「ユーザーからのディレクトリに対するオブジェクト権限の取消し: 例」を参照してください。

ON EDITION

USEオブジェクト権限を取り消すエディションの名前を指定します。edition_nameはスキーマ名で修飾できません。

ON MINING MODEL

権限を取り消すマイニング・モデルの名前を指定します。mining_model_nameをschemaで修飾しなかった場合、そのマイニング・モデルは自分のスキーマ内にあるとみなされます。

ON JAVA SOURCE | RESOURCE

権限を取り消すJavaソースまたはリソース・スキーマ・オブジェクトの名前を指定します。objectをschemaで修飾しなかった場合、そのオブジェクトは自分のスキーマ内にあるとみなされます。

ON SQL TRANSLATION PROFILE

権限を取り消すSQL翻訳プロファイルの名前を指定します。profileをschemaで修飾しなかった場合、そのプロファイルは自分のスキーマ内にあるとみなされます。

revoke_roles_from_programs

この句を使用して、コードに基づくアクセス制御ロール(CBAC)をプログラム・ユニットから取り消すことができます。

role

取り消すロールを指定します。

ALL

ALLを指定すると、プログラム・ユニットに付与されたすべてのロールを取り消すことができます。

program_unit

ロールを取り消すプログラム・ユニットを指定します。PL/SQLファンクション、プロシージャ、またはパッケージを指定できます。schemaを指定しない場合、ファンクション、プロシージャ、またはパッケージは独自のスキーマにあると見なされます。

関連項目:

CBACロールのプログラム・ユニットからの取消しの詳細は、『Oracle Databaseセキュリティ・ガイド』を参照してください。

CONTAINER句

現在のコンテナがプラガブル・データベース(PDB)である場合:

- CONTAINER = CURRENTを指定すると、ローカル・ユーザー、共通ユーザー、ローカル・ロールまたは共通ロールにローカルに付与されているシステム権限、オブジェクト権限またはロールを取り消すことができます。現在のPDBでのみ、その権限またはロールがそのユーザーまたはロールから取り消されます。この句は、CONTAINER = ALLで付与された権限を取り消しません。

現在のコンテナがルートである場合:

- CONTAINER = CURRENTを指定すると、共通ユーザーまたは共通ロールにローカルに付与されているシステム権限、オブジェクト権限またはロールを取り消すことができます。ルートのユーザーまたはロールからのみ、権限またはロールが取り消されます。この句は、CONTAINER = ALLで付与された権限を取り消しません。
- CONTAINER = ALLを指定すると、共通ユーザーまたは共通ロールに共通に付与されているシステム権限、共通オブジェクトに対するオブジェクト権限またはロールを取り消すことができます。CDB全体で、その権限またはロールがそのユーザーまたはロールから取り消されます。この句では、CONTAINER = ALLで付与された権限のみを、指定した共通ユーザーまたは共通ロールから取り消すことができます。CONTAINER = CURRENTで付与された権限は、この句では取り消されません。ただし、取消し対象の共通に付与されている権限に依存する、ローカルに付与されている権限も取り消されます。

この句を省略すると、CONTAINER = CURRENTがデフォルトになります。

例

ユーザーからのシステム権限の取消し: 例

次の文は、ユーザーhrおよびoeからDROP ANY TABLEシステム権限を取り消します。

```
REVOKE DROP ANY TABLE
    FROM hr, oe;
```

この結果、hrおよびoeは他のユーザーのスキーマに定義されている表を削除できなくなります。

ユーザーからのロールの取消し: 例

次の文は、ユーザーshからロールdw_managerを取り消します。

```
REVOKE dw_manager
    FROM sh;
```

この結果、shユーザーはdw_managerロールを使用可能にできなくなります。

ロールからのシステム権限の取消し: 例

次の文は、dw_managerロールからCREATE TABLESPACEシステム権限を取り消します。

```
REVOKE CREATE TABLESPACE
   FROM dw_manager;
```

ロールdw_managerを使用可能にしても、ユーザーは表領域を作成できません。

ロールからのロールの取消し: 例

次の文は、ロールdw_managerからロールdw_userを取り消します。

```
REVOKE dw_user
  FROM dw_manager;
```

この結果、dw_userロールの権限はdw_managerに付与されなくなります。

ユーザーからのオブジェクト権限の取消し: 例

次の文を使用すると、表ordersに対するDELETE、INSERT、READ、SELECTおよびUPDATE権限をユーザーhrに付与できます。

```
GRANT ALL
   ON orders TO hr;
```

次の文は、ユーザーhrから表ordersに対するDELETE権限を取り消します。

```
REVOKE DELETE
   ON orders FROM hr;
```

ユーザーからのすべてのオブジェクト権限の取消し: 例

次の文は、hrに付与したordersに対する残りの権限を取り消します。

```
REVOKE ALL
   ON orders FROM hr;
```

PUBLICからのオブジェクト権限の取消し: 例

ロールPUBLICに権限を付与すると、ビューemp_details_viewに対するSELECTおよびUPDATE権限をすべてのユーザーに付与できます。

```
GRANT SELECT, UPDATE
    ON emp_details_view TO public;
```

次の文は、すべてのユーザーからemp_details_viewに対するUPDATE権限を取り消します。

```
REVOKE UPDATE
    ON emp_details_view FROM public;
```

ユーザーは、emp_details_viewビューへの問合せはできますが、更新はできなくなります。ただし、emp_details_viewに対するUPDATE権限も任意のユーザーに直接またはロールを介して付与している場合は、これらのユーザーはその権限を保持します。

ユーザーからのユーザーに対するオブジェクト権限の取消し: 例

次の文を使用すると、ユーザーshに対するINHERIT PRIVILEGES権限をユーザーhrに付与できます。

```
GRANT INHERIT PRIVILEGES ON USER sh TO hr;
```

ユーザーshに対するINHERIT PRIVILEGES権限をユーザーhrから取り消すには、次の文を発行します。

```
REVOKE INHERIT PRIVILEGES ON USER sh FROM hr;
```

ユーザーからの順序に対するオブジェクト権限の取消し: 例

次の文を使用すると、スキーマhrのdepartments_seq順序に対するSELECT権限をユーザーoeに付与できます。

```
GRANT SELECT
    ON hr.departments_seq TO oe;
```

次の文は、oeからdepartments_seqに対するSELECT権限を取り消します。

```
REVOKE SELECT
    ON hr.departments_seq FROM oe;
```

ただし、ユーザーhrがdepartmentsに対するSELECT権限をshに付与している場合、shは、hrからの権限付与によってdepartmentsを使用できます。

CASCADE CONSTRAINTSによるオブジェクト権限の取消し: 例

次の文を使用すると、スキーマhrのemployees表に対する権限REFERENCESおよびUPDATEをoeに付与できます。

```
GRANT REFERENCES, UPDATE
    ON hr.employees TO oe;
```

ユーザーoeは、REFERENCES権限を使用して、hrスキーマ内のemployees表を参照するdependent表の制約を定義できます。

```
CREATE TABLE dependent
(dependno   NUMBER,
 dependname VARCHAR2(10),
 employee   NUMBER
    CONSTRAINT in_emp REFERENCES hr.employees(employee_id) );
```

CASCADE CONSTRAINTS句を指定した次の文を発行することによって、oeのhr.employeesに対するREFERENCES権限を取り消すことができます。

```
REVOKE REFERENCES
    ON hr.employees
    FROM oe
    CASCADE CONSTRAINTS;
```

oeのhr.employeesに対するREFERENCES権限を取り消した場合、oeは制約を定義する権限が必要になるため、in_emp制約が自動的に削除されます。

ただし、oeが他のユーザーからhr.employeesに対するREFERENCES権限を付与されている場合は、その制約は削除されません。他のユーザーから権限付与されたため、oeは制約に対して必要な権限を保持しています。

ユーザーからのディレクトリに対するオブジェクト権限の取消し: 例

次の文を発行することによって、hrからディレクトリbfile_dirに対するREADオブジェクト権限を取り消すことができます。

```
REVOKE READ ON DIRECTORY bfile_dir FROM hr;
```

GRANT ANY OBJECT PRIVILEGEを使用する操作の取消し: 例

データベース管理者が、GRANT ANY OBJECT PRIVILEGEをユーザーshに付与したとします。ユーザーhrが、employees表に対するUPDATE権限をユーザーoeに付与します。

```
CONNECT hr
GRANT UPDATE ON employees TO oe WITH GRANT OPTION;
```

これによって、オブジェクト権限を別のユーザーに付与する権限がユーザーoeに付与されます。

```
CONNECT oe
GRANT UPDATE ON hr.employees TO pm;
```

oeにはhrによって権限が与えられたため、GRANT ANY OBJECT PRIVILEGEが付与されているユーザーshは、ユーザーhrにかわってユーザーoeのUPDATE権限を取り消すことができます。

```
CONNECT sh
REVOKE UPDATE ON hr.employees FROM oe;
```

pmに権限を付与したのがオブジェクトの所有者(hr)、ユーザーshまたはGRANT ANY OBJECT PRIVILEGEを持つ他のユーザーではなく、ユーザーoeであったため、ユーザーshは、ユーザーpmのUPDATE権限を明示的に取り消すことはできません。ただし、前述の文は、連鎖的な取消しを行い、取り消された権限に依存するすべての権限を取り消します。そのため、pmのオブジェクト権限も暗黙的に取り消されます。

ROLLBACK

目的

ROLLBACK文を使用すると、現行のトランザクションで実行された処理を取り消すことができます。また、インダウト分散トランザクションで実行された処理を手動で取り消すこともできます。

> ノート:
>
> アプリケーション・プログラムでは、COMMIT または ROLLBACK 文を使用してトランザクションを明示的に終了することをお薦めします。トランザクションを明示的にコミットせずにプログラムが異常終了した場合、コミットされていない最後のトランザクションがロールバックされます。

関連項目:

- トランザクションの詳細は、『Oracle Database概要』を参照してください。
- 分散トランザクションの詳細は、『Oracle Database Heterogeneous Connectivityユーザーズ・ガイド』を参照してください。
- 現行トランザクションの特性の設定については、「SET TRANSACTION」を参照してください。
- COMMIT and SAVEPOINT

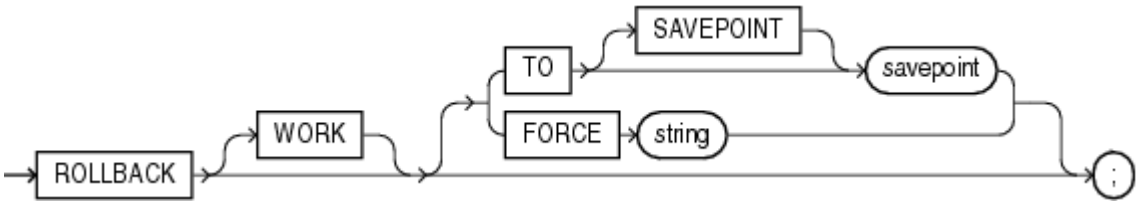
前提条件

現行のトランザクションをロールバックする場合、権限は不要です。

コミットしたインダウト分散トランザクションを手動でロールバックする場合は、FORCE TRANSACTIONシステム権限が必要です。他のユーザーがコミットしたインダウト分散トランザクションを手動でロールバックする場合は、FORCE ANY TRANSACTION システム権限が必要です。

構文

rollback::=



セマンティクス

WORK

WORKキーワードの指定は任意です。このキーワードは、SQL規格との互換性のために提供されています。

TO SAVEPOINT句

現行のトランザクションをロールバックするセーブポイントを指定します。この句を指定しない場合、ROLLBACK文によってトランザクション全体がロールバックされます。

TO SAVEPOINT句を指定しないでROLLBACKコマンドを実行すると、次の処理が行われます。

- トランザクションの終了
- 現行のトランザクションに対するすべての変更の取消し
- トランザクション内のすべてのセーブポイントが消去されます。
- トランザクションのロックの解除

関連項目:

SAVEPOINT

TO SAVEPOINT句を指定してROLLBACKコマンドを実行すると、次の処理が行われます。

- トランザクションのセーブポイント後の部分のみがロールバックされます。トランザクションは終了しません。
- 指定したセーブポイントより後に作成されたすべてのセーブポイントの消去。指定したセーブポイントはそのまま残るため、そのセーブポイントまで繰り返しロールバックできます。指定したセーブポイントより前に作成されたセーブポイントも残ります。
- 指定したセーブポイント以降に取得されたすべての表と行のロックの解除。セーブポイント後にロックされた行へのアクセスを要求した他のトランザクションは、コミットまたはロールバックされるまで待つ必要があります。行を要求していない他のトランザクションは、すぐに行の要求およびアクセスができます。

インダウト・トランザクションの制限事項

インダウトのトランザクションを手動でセーブポイントまでロールバックすることはできません。

FORCE句

FORCEを指定すると、インダウト分散トランザクションを手動でロールバックできます。このトランザクションは、ローカル・トランザクションIDまたはグローバル・トランザクションIDを含むstringで識別されます。このトランザクションのIDを確認する場合は、データ・ディクショナリ・ビューDBA_2PC_PENDINGを問い合せます。

FORCE句を指定したROLLBACK文では、指定したトランザクションのみがロールバックされます。この文は、現行のトランザクションには影響しません。

関連項目:

分散トランザクションおよびインダウト・トランザクションのロールバックの詳細は、『Oracle Database管理者ガイド』を参照してください。

例

トランザクションのロールバック: 例

次の文は、現行のトランザクション全体をロールバックします。

```
ROLLBACK;
```

次の文は、現行のトランザクションをセーブポイントbanda_salにロールバックします。

```
ROLLBACK TO SAVEPOINT banda_sal;
```

前述の例の詳細は、「セーブポイントの作成：例」を参照してください。

次の文は、インダウト分散トランザクションを手動でロールバックします。

```
ROLLBACK WORK
    FORCE '25.32.87';
```

SAVEPOINT

目的

SAVEPOINT文を使用すると、後でロールバックできるシステム変更番号(SCN)の名前を作成できます。

関連項目:

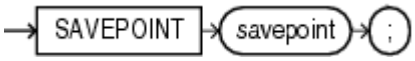
- セーブポイントの詳細は、『Oracle Database概要』を参照してください。
- トランザクションのロールバックの詳細は、「ROLLBACK」を参照してください。
- 現行トランザクションの特性の設定については、「SET TRANSACTION」を参照してください。

前提条件

なし。

構文

savepoint::=



セマンティクス

savepoint

作成するセーブポイントの名前を指定します。

同一トランザクション内のセーブポイント名は、区別する必要があります。同じ識別子のセーブポイントを作成した場合、最初のセーブポイントは消去されます。セーブポイントを作成した後は、処理の継続、作業のコミット、トランザクション全体のロールバックまたはセーブポイントまでのロールバックを実行できます。

例

セーブポイントの作成: 例

次の文は、サンプル表hr.employeesのBandaとGreeneの給与を更新するために、部門の給与合計が314,000ドルを超えていないことを確認してから、Greeneの給与を再入力します。

```
UPDATE employees
    SET salary = 7000
    WHERE last_name = 'Banda';
SAVEPOINT banda_sal;
UPDATE employees
    SET salary = 12000
    WHERE last_name = 'Greene';
SAVEPOINT greene_sal;
SELECT SUM(salary) FROM employees;
ROLLBACK TO SAVEPOINT banda_sal;

UPDATE employees
    SET salary = 11000
    WHERE last_name = 'Greene';

COMMIT;
```

SELECT

目的

1つ以上の表、オブジェクト表、ビュー、オブジェクト・ビュー、マテリアライズド・ビュー、分析ビューまたは階層からデータを取得するには、SELECT文または副問合せを使用します。

SELECT文の結果(またはその一部)が既存のマテリアライズド・ビューと同じ場合、そのマテリアライズド・ビューをSELECT文で指定した1つ以上の表のかわりに使用できます。この置き換えを、クエリー・リライトといいます。これが行われるのは、コスト最適化が有効であり、かつQUERY_REWRITE_ENABLEDパラメータがTRUEに設定されている場合のみです。クエリー・リライトが行われたかどうかを特定するには、EXPLAIN PLAN文を使用します。

関連項目:

- 問合せと副問合せの概要は、「SQL問合せおよび副問合せ」を参照してください。
- マテリアライズド・ビュー、クエリー・リライト、分析ビューおよび階層の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- JSONデータを問い合せる場合は、「JSONデータの問合せ」を参照してください。
- XMLデータを問い合せる場合は、「Oracle XML DBに格納されたXMLコンテンツの問合せ」を参照してください。
- EXPLAIN PLAN

前提条件

表、マテリアライズド・ビュー、分析ビューまたは階層からデータを選択する場合、オブジェクトは自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、その表、マテリアライズド・ビュー、分析ビューまたは階層に対するREADまたはSELECT権限が必要です。

ビューの実表から行を選択する場合、次の条件を2つとも満たしている必要があります。

- オブジェクトが自分のスキーマ内に設定されている必要がある。設定されていない場合は、オブジェクトに対するREADまたはSELECT権限が必要である。
- オブジェクトを含むスキーマの所有者が、実表に対するREADまたはSELECT権限を持っている必要がある。

READ ANY TABLEシステム権限またはSELECT ANY TABLEシステム権限が付与されている場合、任意の表、マテリアライズド・ビュー、分析ビューまたは階層、あるいはマテリアライズド・ビュー、分析ビューまたは階層の実表からデータを選択できます。

FOR UPDATE句を指定するには、前述の前提条件が次の例外とともに適用されます。READおよびREAD ANY TABLE権限が示されている場合、FOR UPDATE句を指定できません。

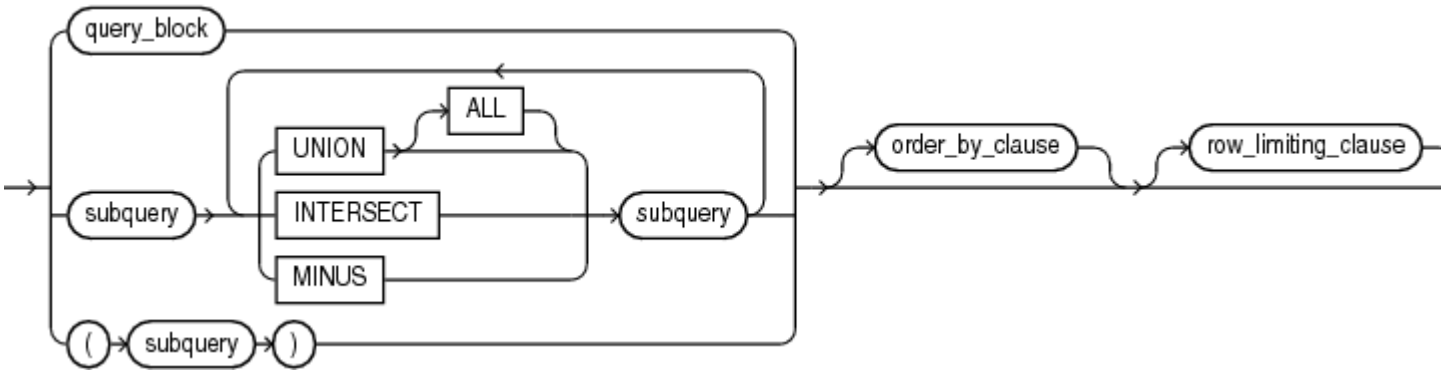
flashback_query_clauseを使用してOracleフラッシュバック問合せを発行する場合は、SELECT構文のリスト内のオブジェクトに対するREADまたはSELECT権限が必要です。さらに、SELECT構文のリスト内のオブジェクトに対するFLASHBACKオブジェクト権限またはFLASHBACK ANY TABLEシステム権限のいずれかが必要です。

構文

select::=

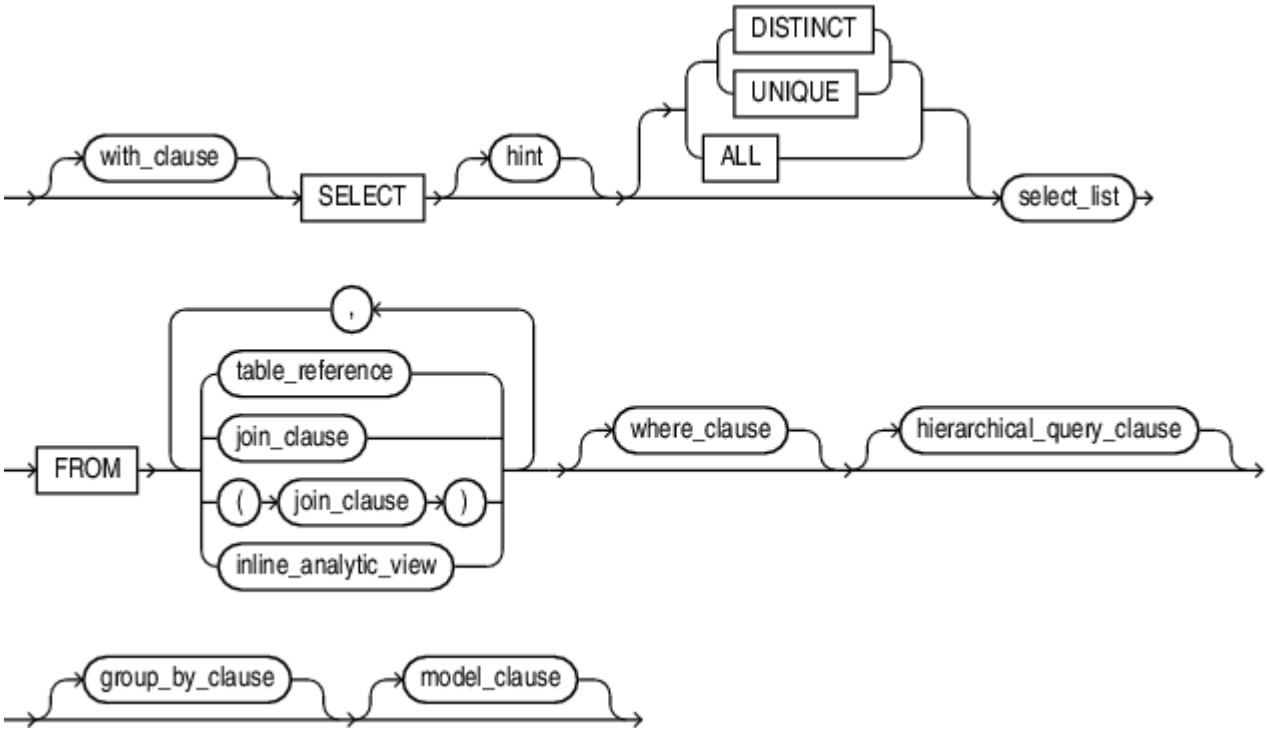
(subquery::=、for_update_clause::=)

subquery::=



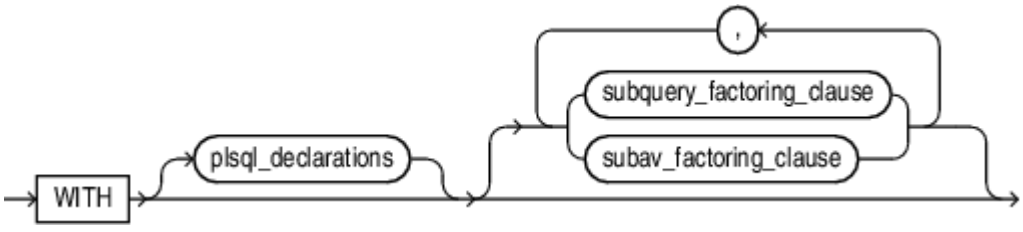
(query_block::=、order_by_clause::=、row_limiting_clause::=)

query_block::=



(with_clause::=, select_list::=, table_reference::=, join_clause::=, inline_analytic_view, where_clause::=, hierarchical_query_clause::=, group_by_clause::=, model_clause::=)

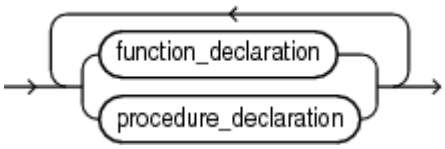
with_clause::=



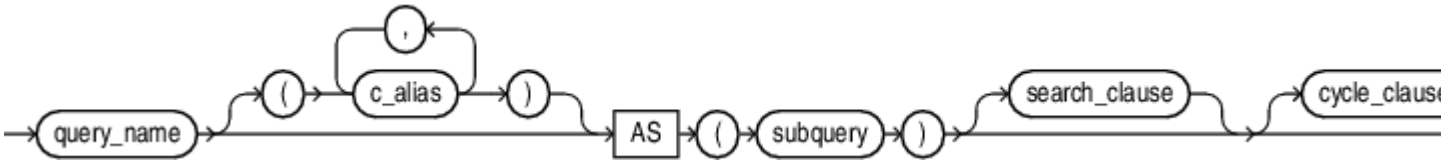
ノート:

`WITH` キーワードは、単独では指定できません。最低限、`plsql_declarations`、`subquery_factoring_clause`、`subav_factoring_clause` のいずれかの句を指定する必要があります。

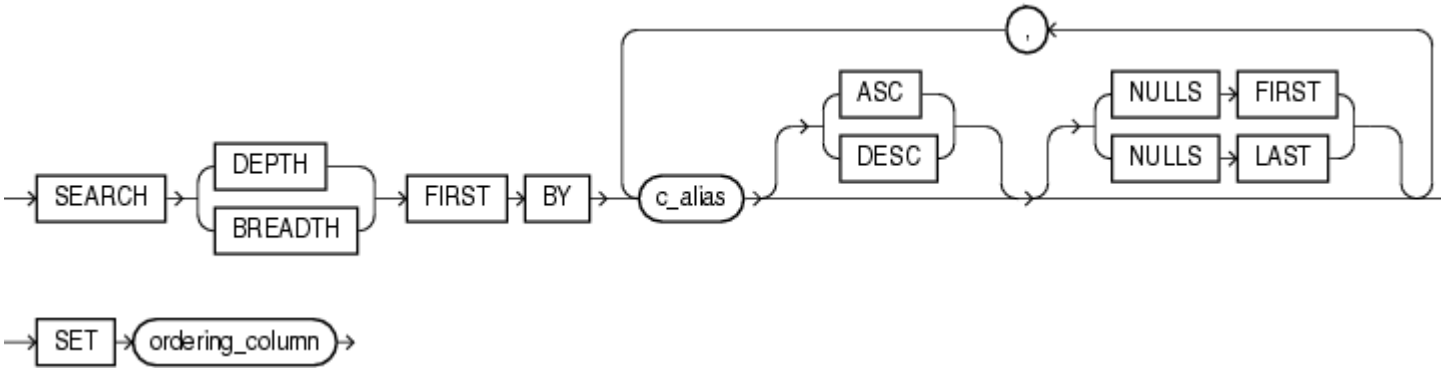
plsql_declarations::=



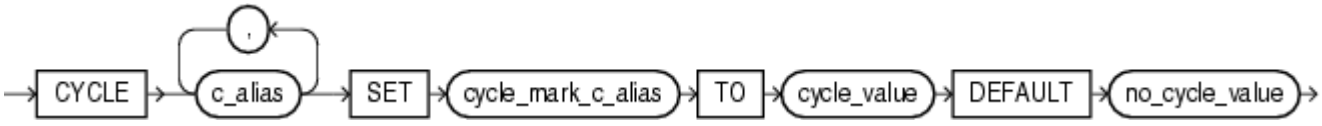
subquery_factoring_clause::=



search_clause::=

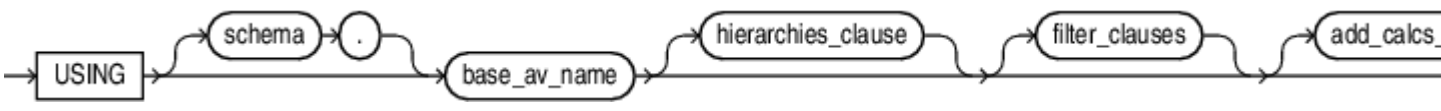


cycle_clause::=

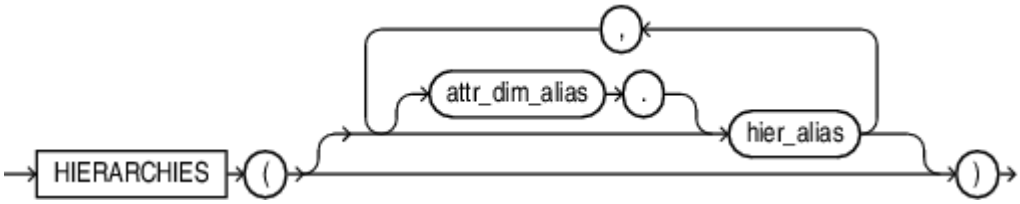


subav_factoring_clause::=

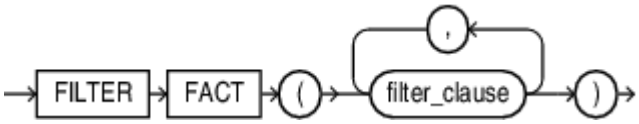
subav_clause::=



hierarchies_clause::=

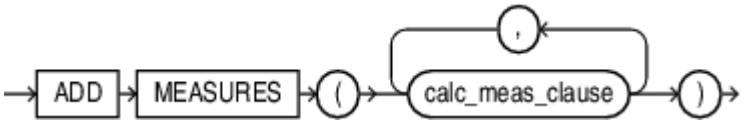


filter_clauses::=

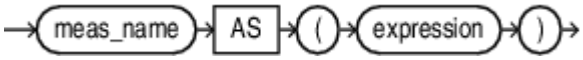


filter_clause::=

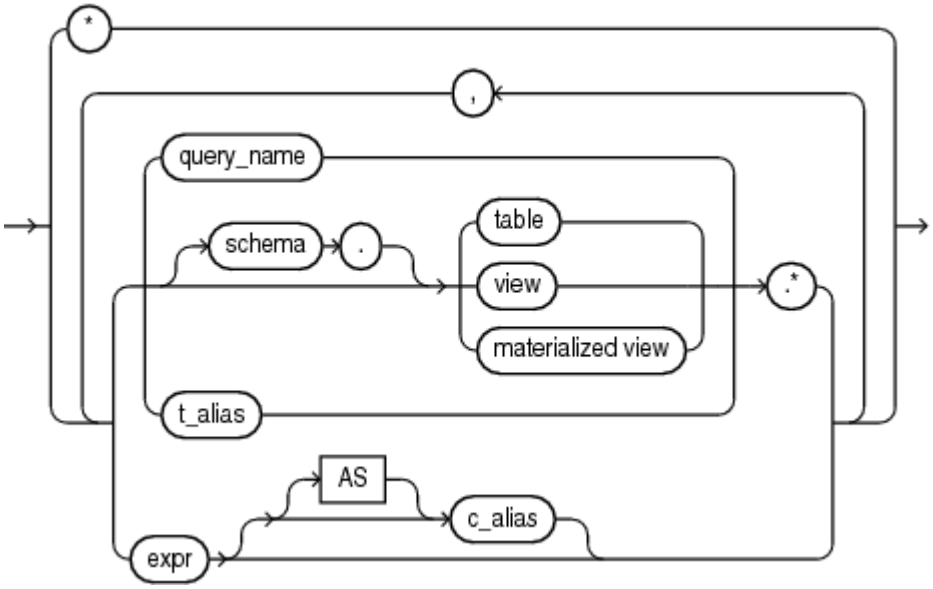
add_calcs_clause::=



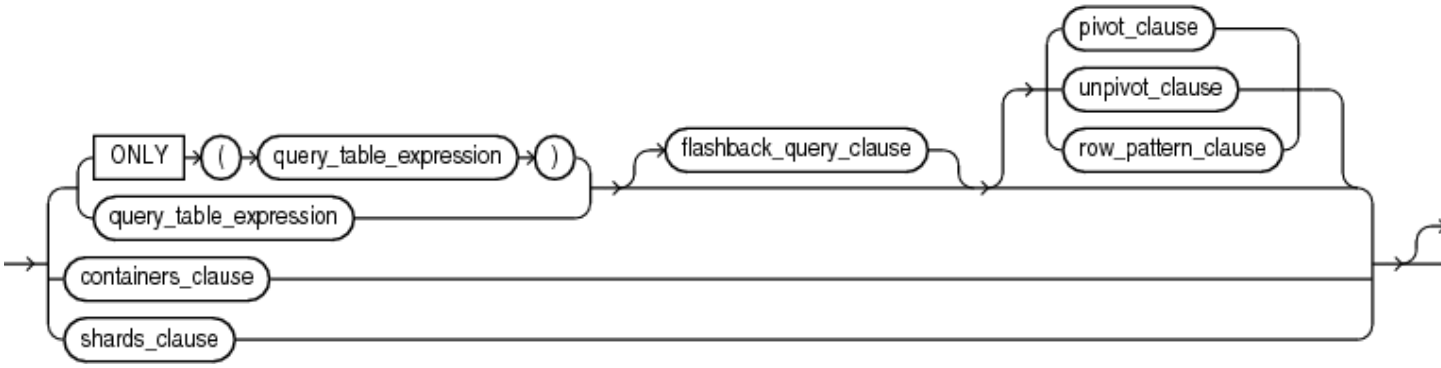
calc_meas_clause::=



select_list::=

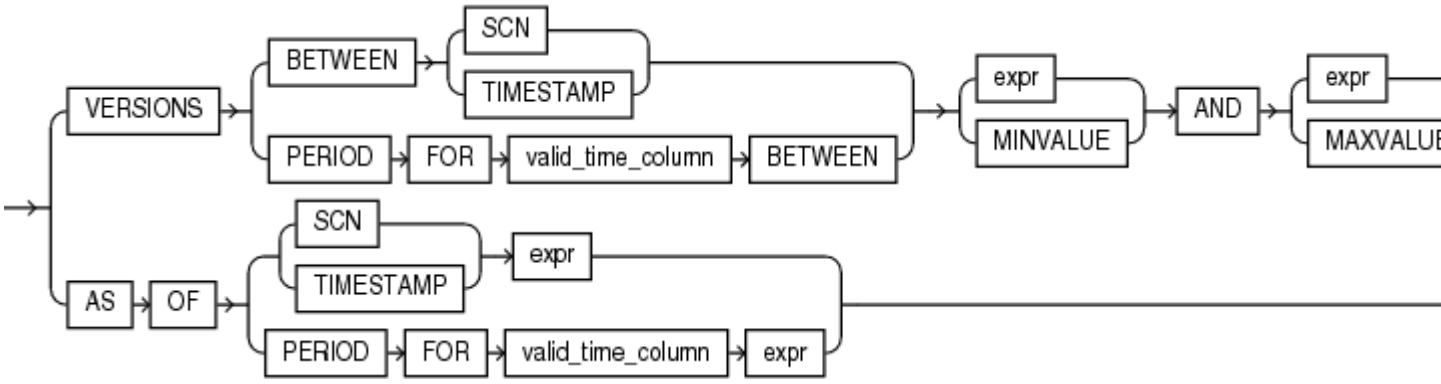


table_reference::=

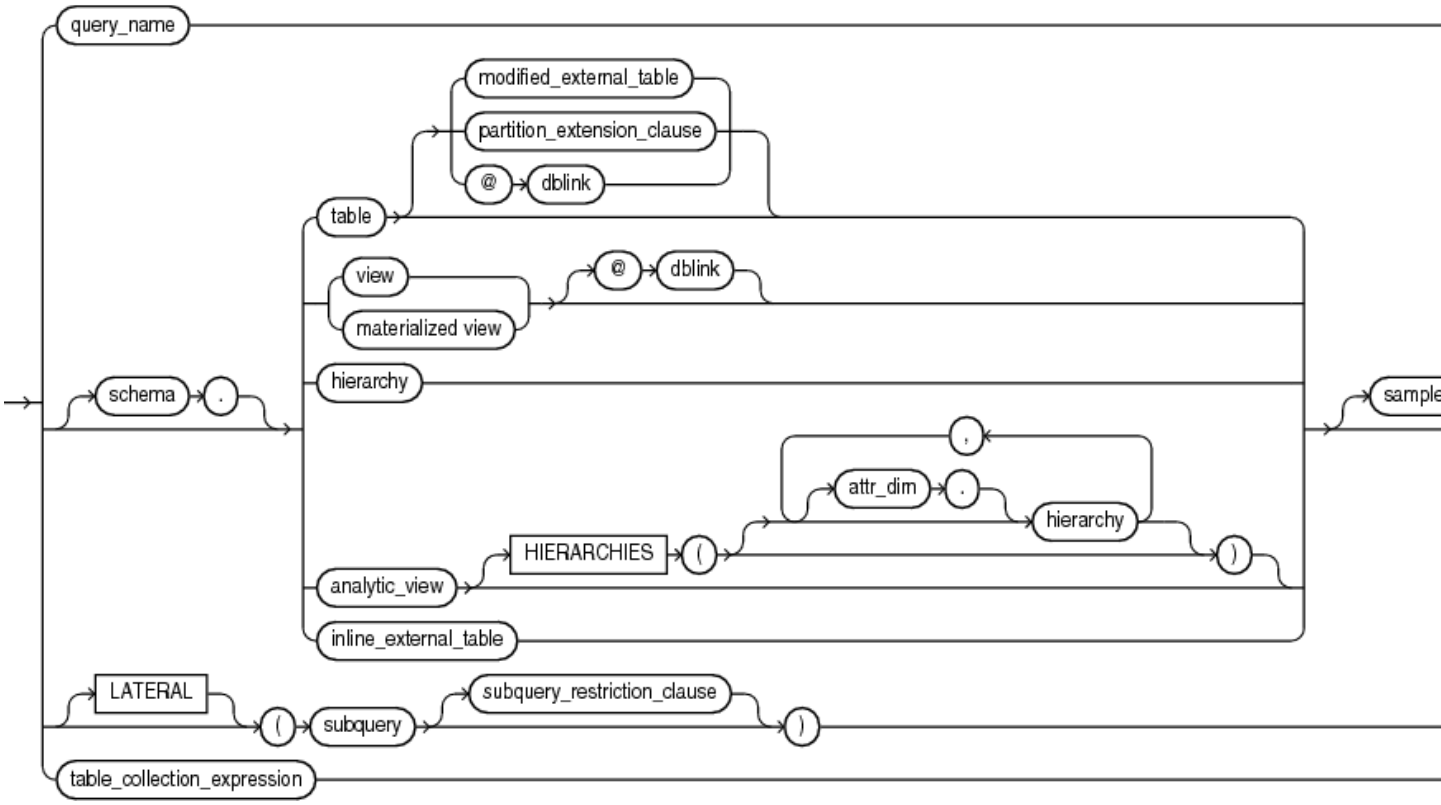


(query_table_expression::=、flashback_query_clause::=、pivot_clause::=、unpivot_clause::=、row_pattern_clause::=、containers_clause::=)

flashback_query_clause::=

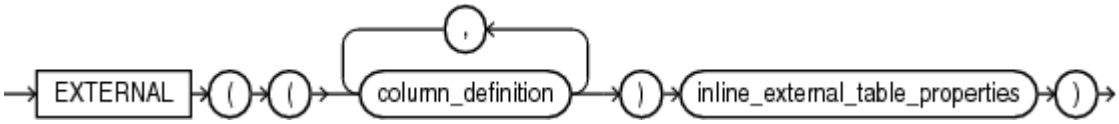


query_table_expression::=

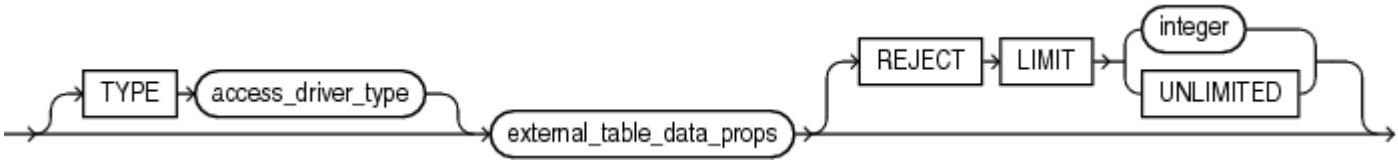


(analytic_view, hierarchy, subquery_restriction_clause::=, table_collection_expression::=)

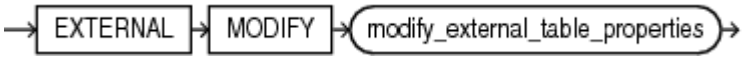
inline_external_table::=



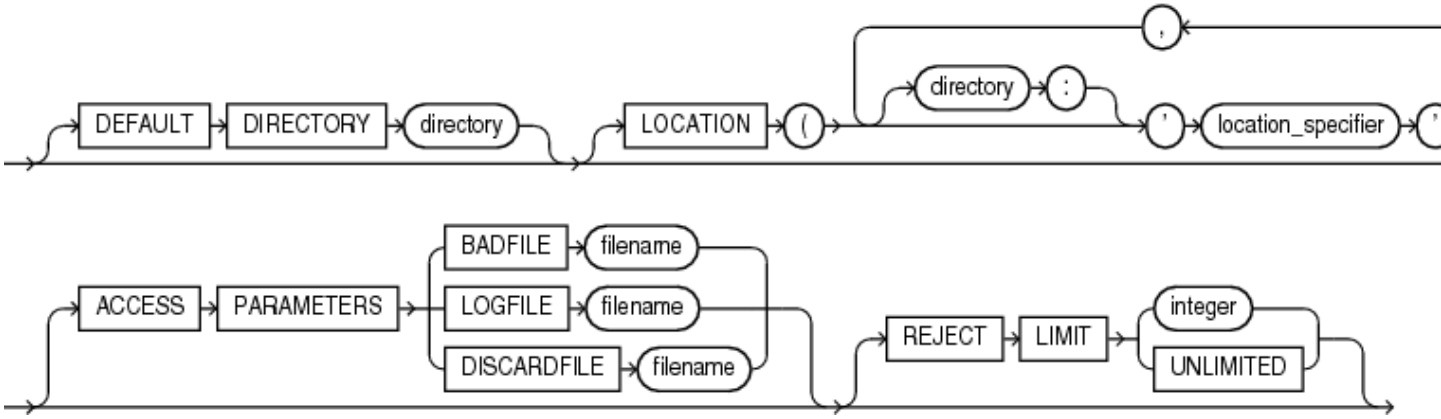
inline_external_table_properties::=



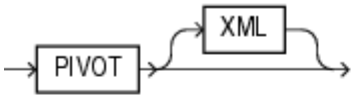
modified_external_table::=

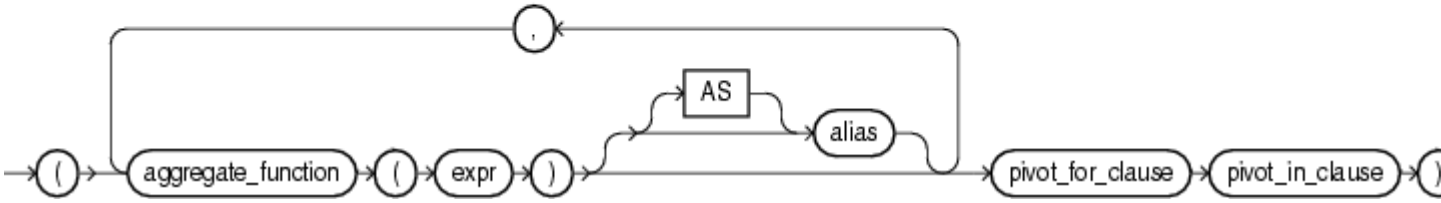


modify_external_table_properties::=

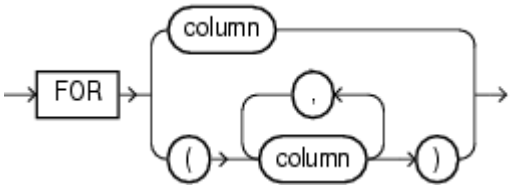


pivot_clause::=

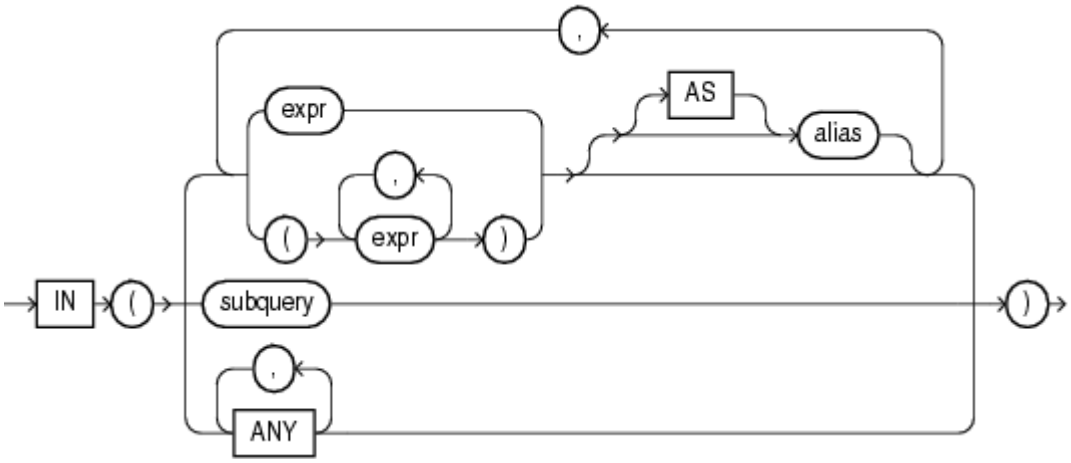




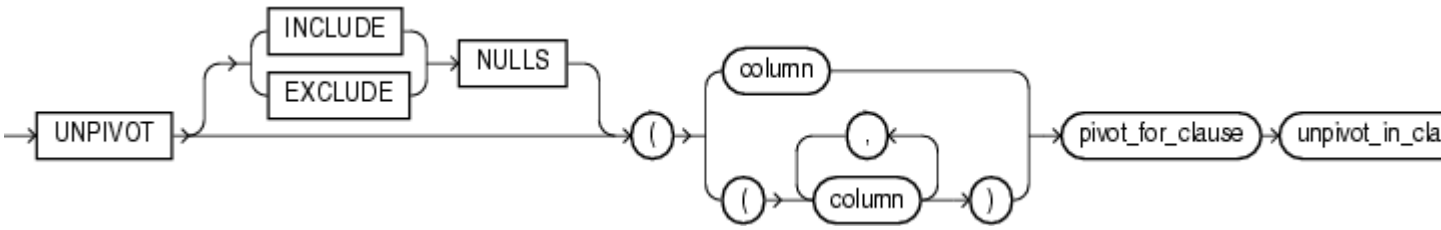
pivot_for_clause::=



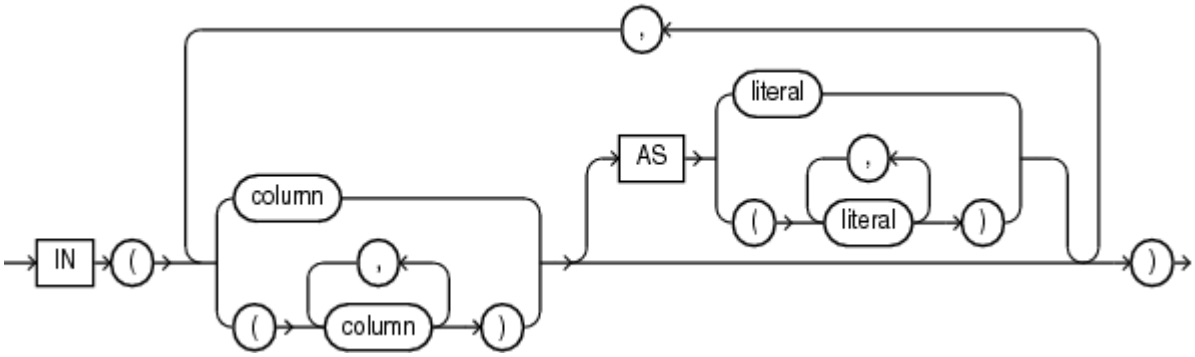
pivot_in_clause::=



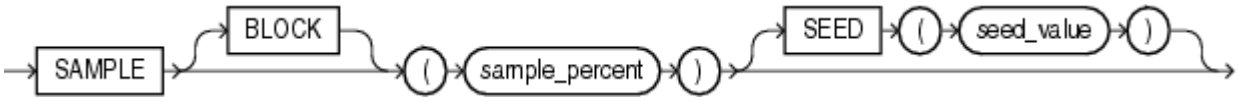
unpivot_clause::=



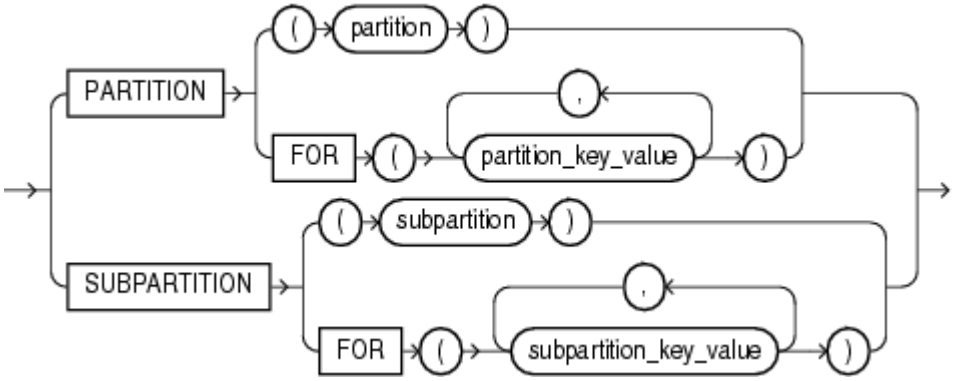
unpivot_in_clause::=



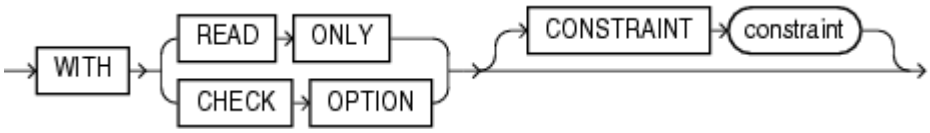
sample_clause::=



partition_extension_clause::=



subquery_restriction_clause::=

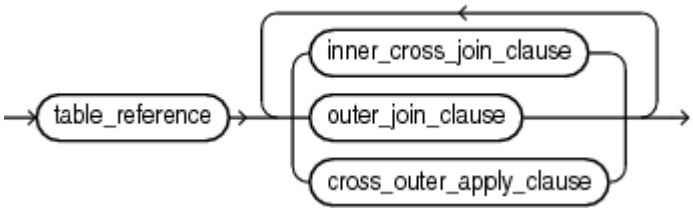


table_collection_expression::=

containers_clause::=

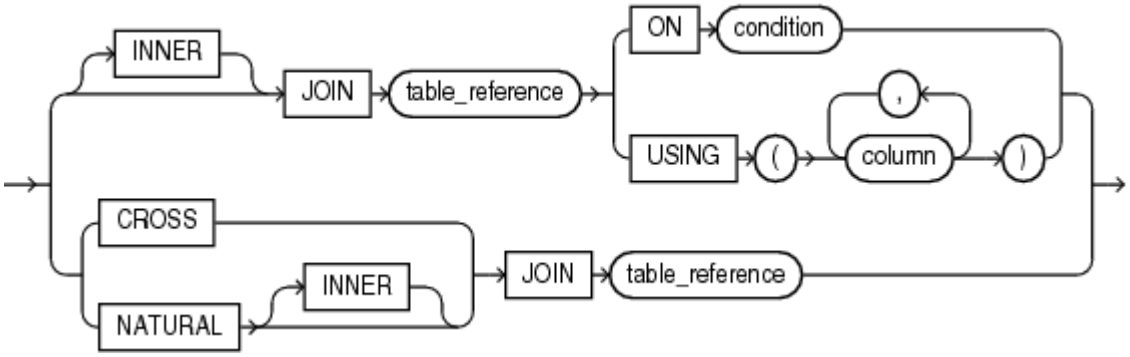
shards_clause::=

join_clause::=



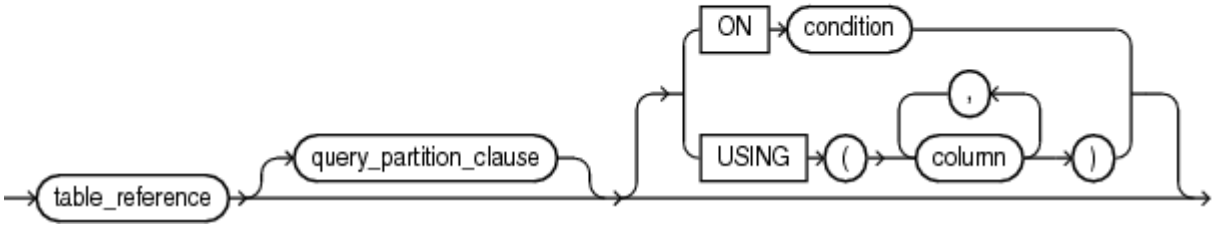
(inner_cross_join_clause::=、outer_join_clause::=、cross_outer_apply_clause::=)

inner_cross_join_clause::=



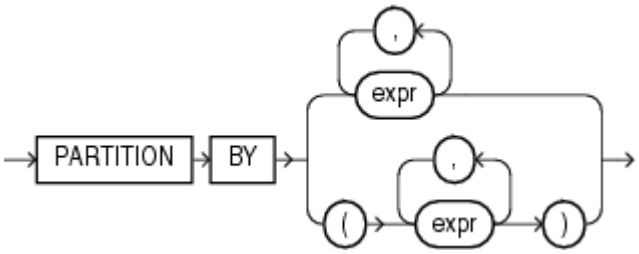
(table_reference::=)

outer_join_clause::=

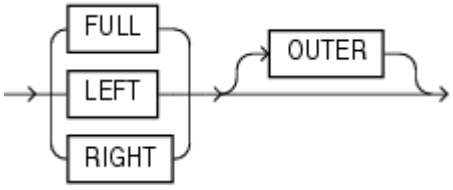


(query_partition_clause::=、outer_join_type::=、table_reference::=)

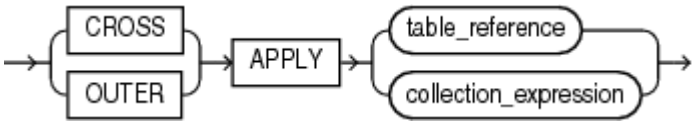
query_partition_clause::=



outer_join_type::=

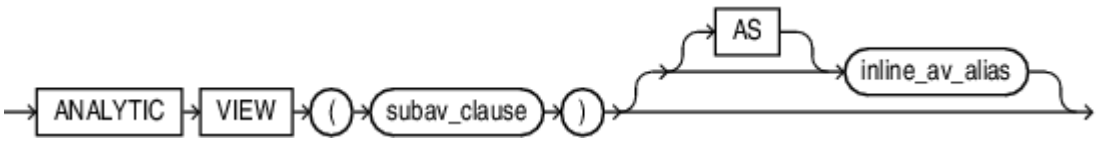


cross_outer_apply_clause::=



(table_reference::=、query_partition_clause::=)

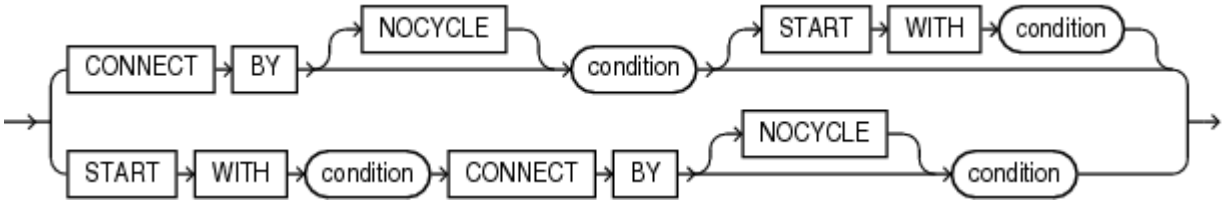
inline_analytic_view



(subav_clause::=)

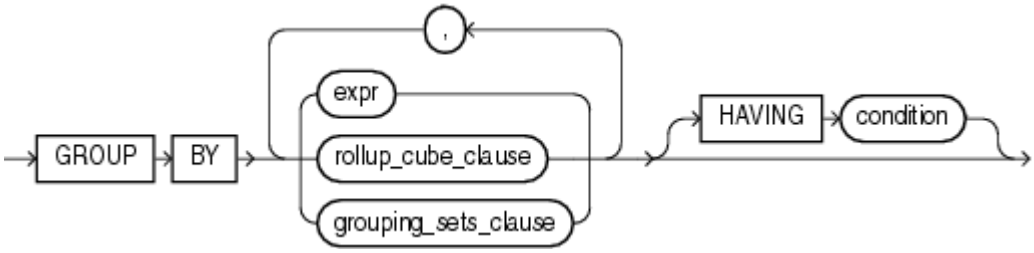
where_clause::=

hierarchical_query_clause::=



(conditionには、「条件」で説明されているいずれかの条件を指定できます。)

group_by_clause::=

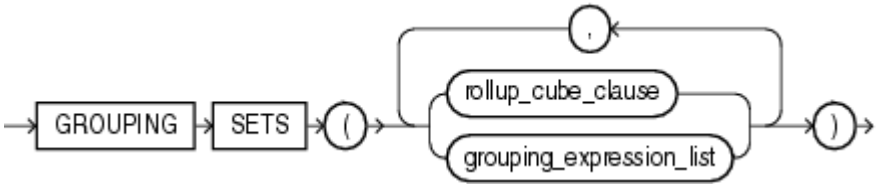


(rollup_cube_clause::=、grouping_sets_clause::=)

rollup_cube_clause::=

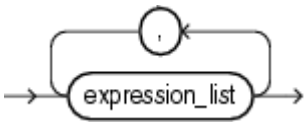
(grouping_expression_list::=)

grouping_sets_clause::=

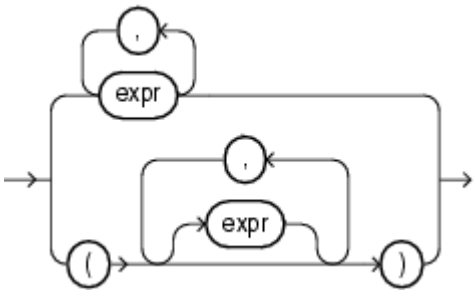


(rollup_cube_clause::=、grouping_expression_list::=)

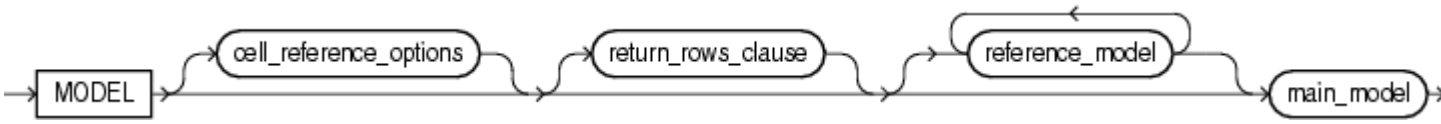
grouping_expression_list::=



expression_list::=

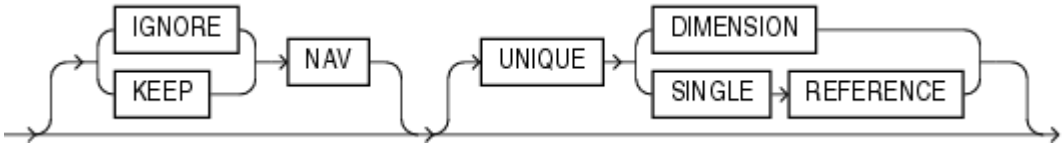


model_clause::=

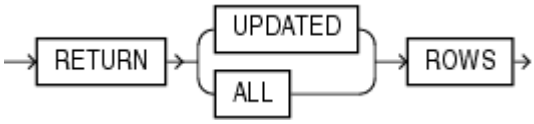


(cell_reference_options::=、return_rows_clause::=、reference_model::=、main_model::=)

cell_reference_options::=



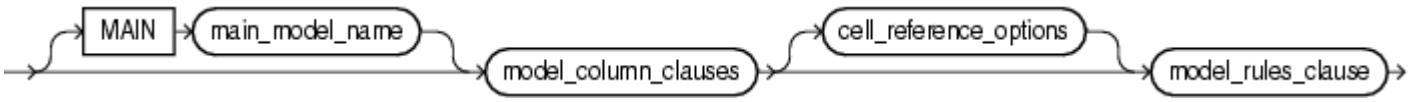
return_rows_clause::=



reference_model::=

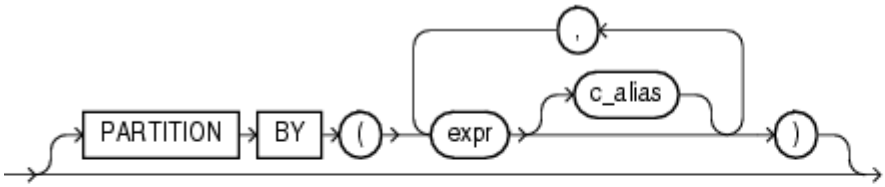
(model_column_clauses::=、cell_reference_options::=)

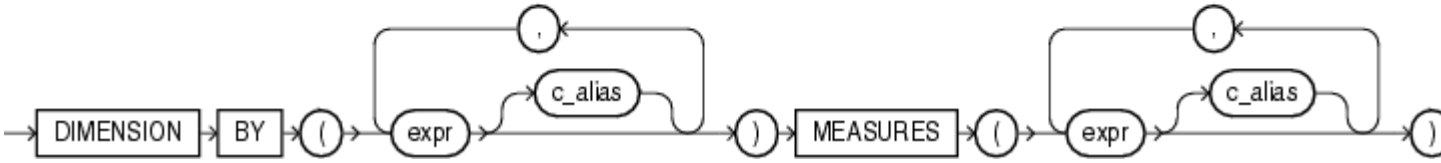
main_model::=



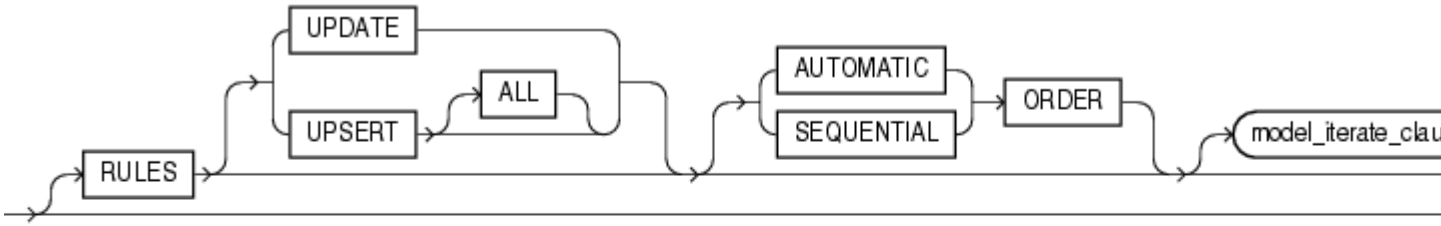
(model_column_clauses::=、cell_reference_options::=、model_rules_clause::=)

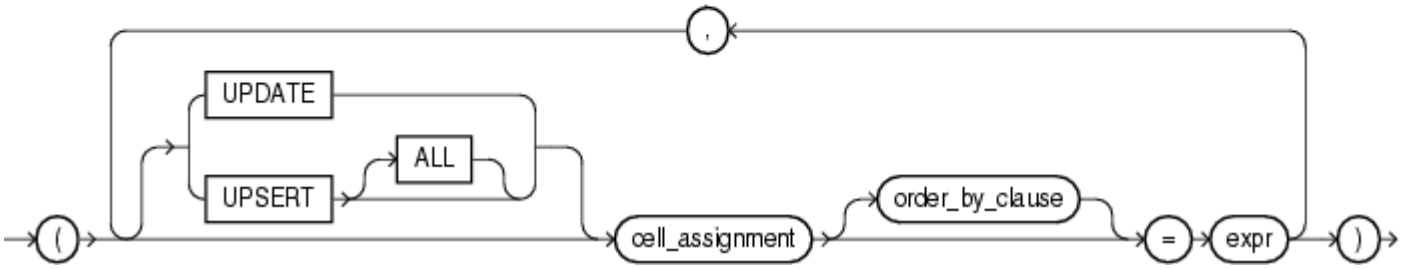
model_column_clauses::=





model_rules_clause::=

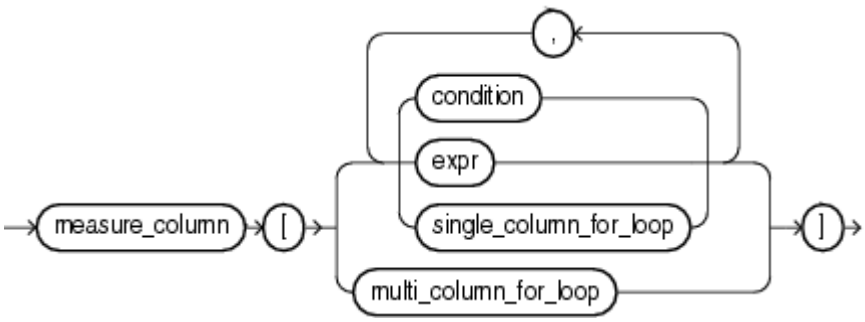




(model_iterate_clause::=、cell_assignment::=、order_by_clause::=)

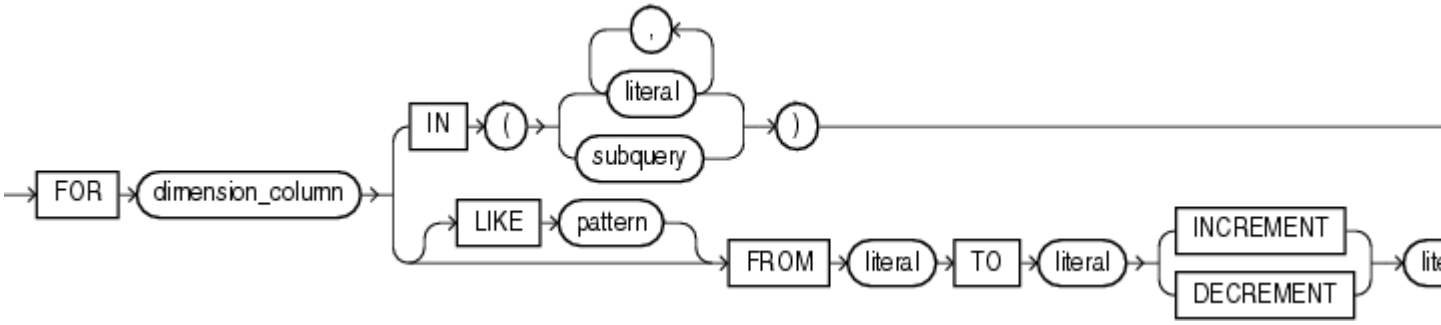
model_iterate_clause::=

cell_assignment::=

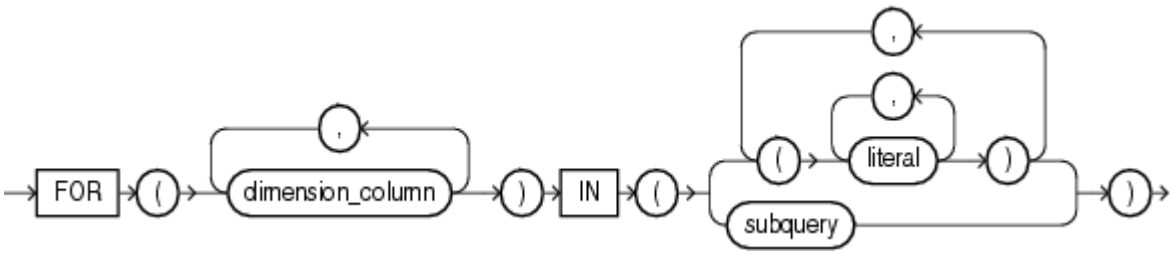


(single_column_for_loop::=、multi_column_for_loop::=)

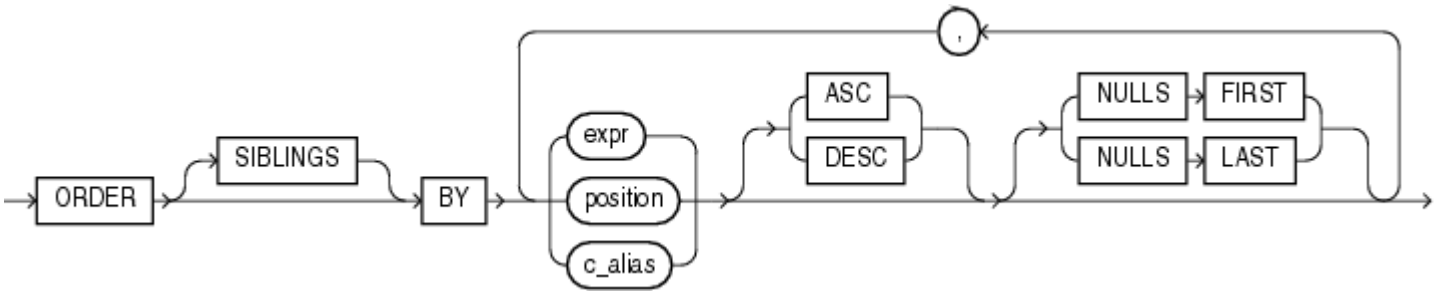
single_column_for_loop::=



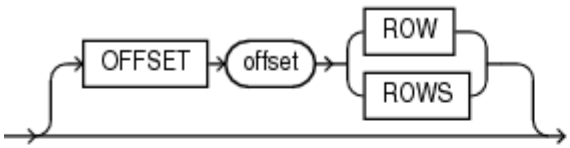
multi_column_for_loop::=

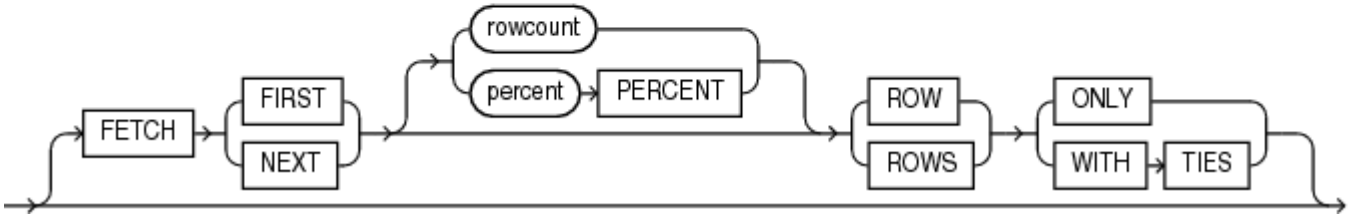


order_by_clause::=

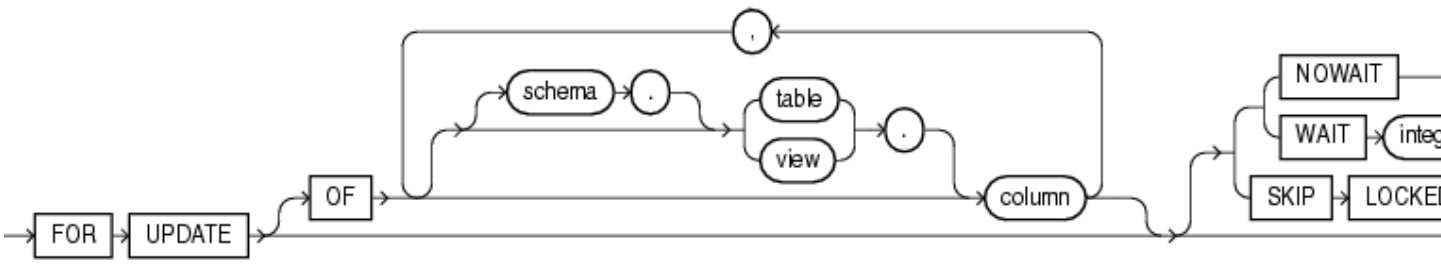


row_limiting_clause::=

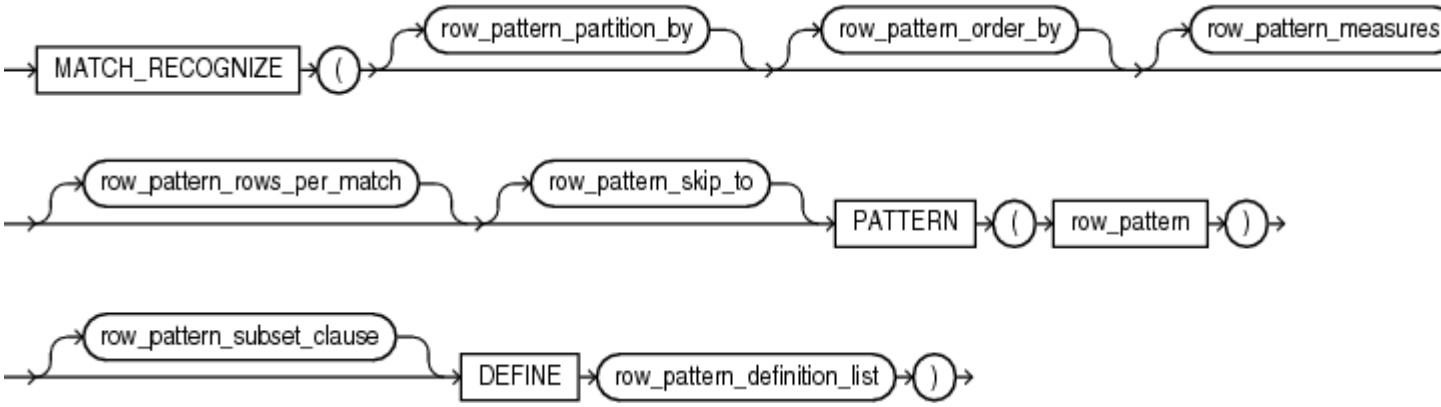




for_update_clause::=

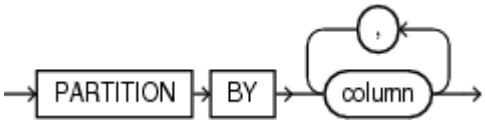


row_pattern_clause::=

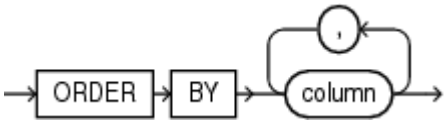


(row_pattern_partition_by::=、row_pattern_order_by::=、row_pattern_measures::=、row_pattern_rows_per_match::=、row_pattern_skip_to::=、row_pattern::=、row_pattern_subset_clause::=、row_pattern_definition_list::=)

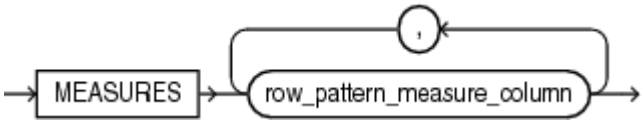
row_pattern_partition_by::=



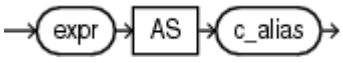
row_pattern_order_by::=



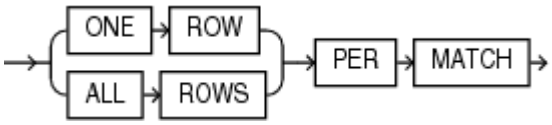
row_pattern_measures::=



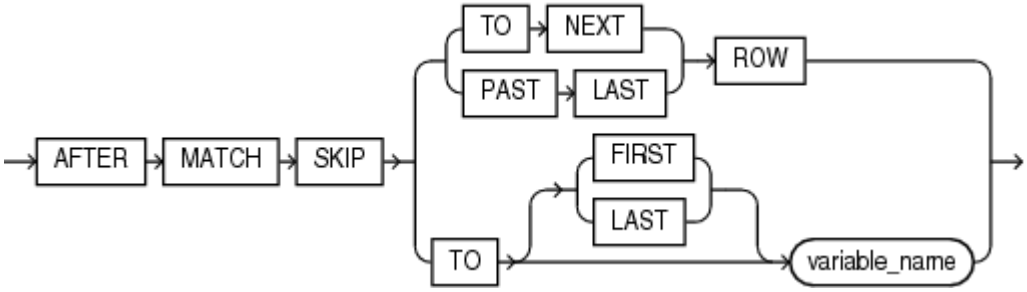
row_pattern_measure_column::=



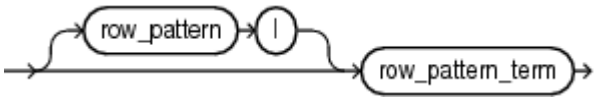
row_pattern_rows_per_match::=



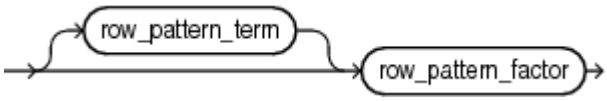
row_pattern_skip_to::=



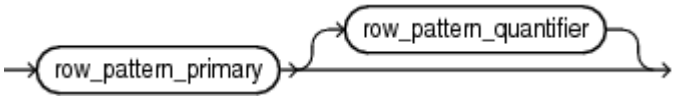
row_pattern::=



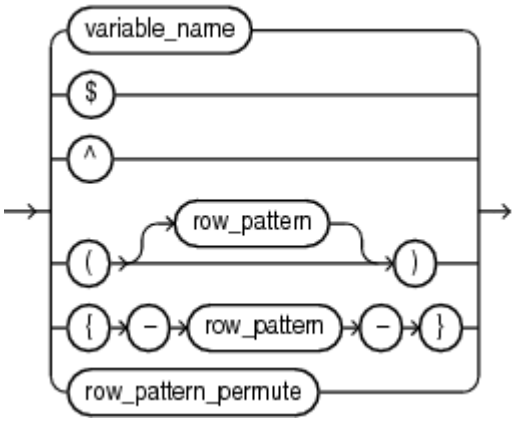
row_pattern_term::=



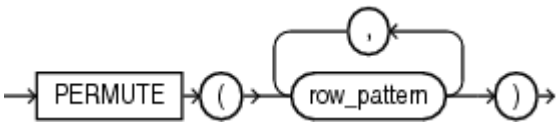
row_pattern_factor::=



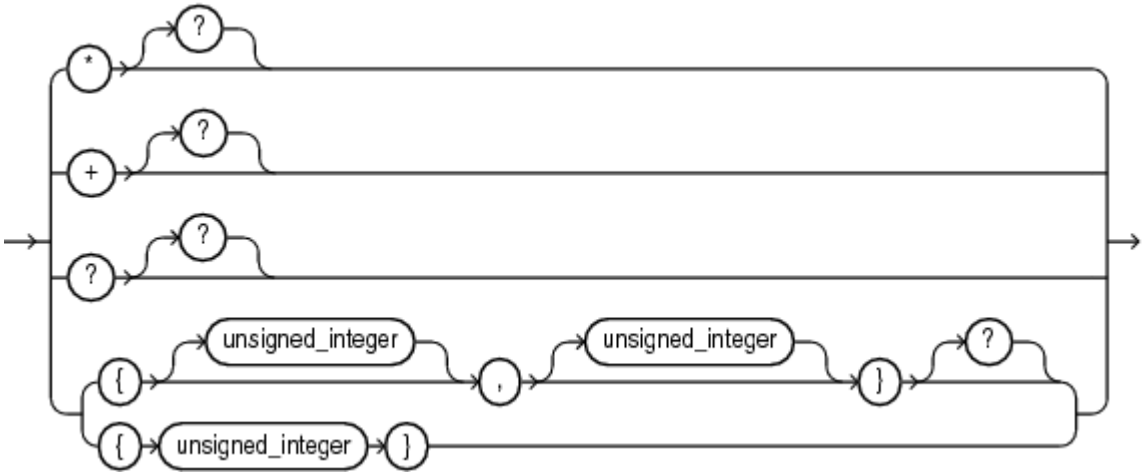
row_pattern_primary::=



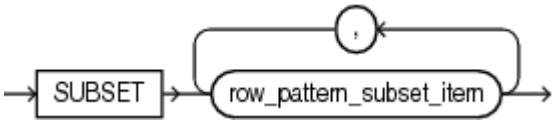
row_pattern_permute::=



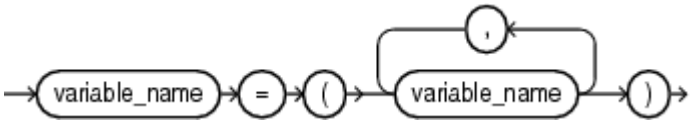
row_pattern_quantifier::=



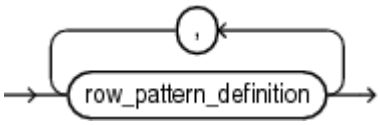
row_pattern_subset_clause::=



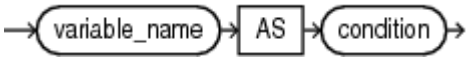
row_pattern_subset_item::=



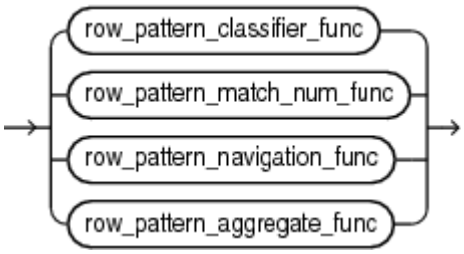
row_pattern_definition_list::=



row_pattern_definition::=



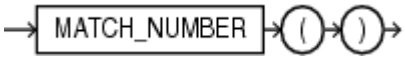
row_pattern_rec_func::=



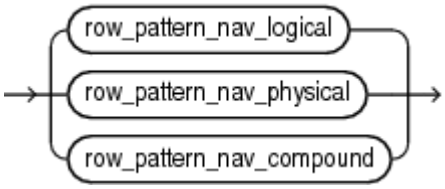
(row_pattern_classifier_func::=、row_pattern_match_num_func::=、
row_pattern_navigation_func::=、row_pattern_aggregate_func::=)

row_pattern_classifier_func::=

row_pattern_match_num_func::=

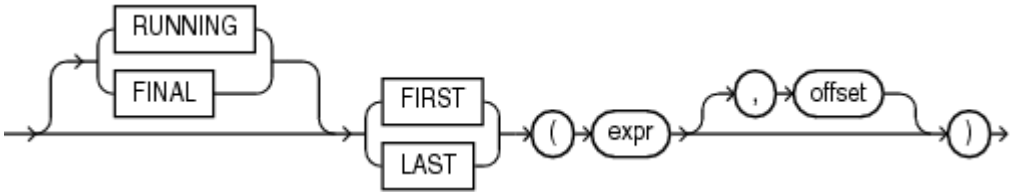


row_pattern_navigation_func::=



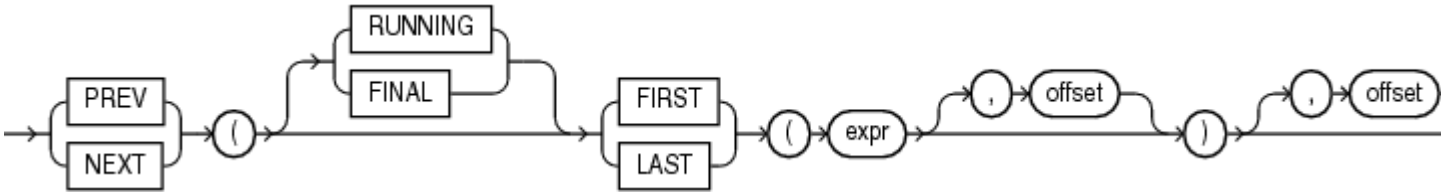
(row_pattern_nav_logical::=、row_pattern_nav_physical::=、row_pattern_nav_compound::=)

row_pattern_nav_logical::=

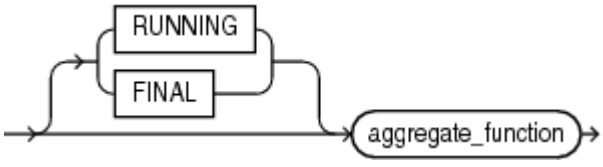


row_pattern_nav_physical::=

row_pattern_nav_compound::=



row_pattern_aggregate_func::=



セマンティクス

with_clause

`with_clause`を使用すると、次の項目を定義できます。

- PL/SQLプロシージャおよびファンクション(`plsql_declarations`句を使用)
- 副問合せブロック(`subquery_factoring_clause`または`subav_factoring_clause`のいずれかまたは両

方を使用)

plsql_declarations

plsql_declarations句を使用すると、PL/SQLファンクションとプロシージャの宣言と定義ができます。その後、この句で指定したPL/SQLファンクションは、問合せ内で参照できます。この問合せに副問合せがある場合は、その副問合せでも参照できます。これらのファンクションの名前が名前解決されるときには、スキーマ・レベルのストアド・ファンクションよりも優先されます。

この句で指定した問合せがトップレベルのSELECT文ではない場合、その問合せを含むトップレベルのSQL文には、次のルールが適用されます。

- トップレベルの文がSELECT文の場合は、WITH plsql_declarations句またはWITH_PLSQLヒントが含まれている必要があります。
- トップレベルの文がDELETE文、MERGE文、INSERT文、またはUPDATE文の場合は、WITH_PLSQLヒントが含まれている必要があります。

WITH_PLSQLヒントを使用して実行できる操作は、文へのWITH plsql_declarations句の指定のみです。これは、オプティマイザ・ヒントではありません。

関連項目:

- function_declarationとprocedure_declarationの構文と制限事項については、『Oracle Database PL/SQL言語リファレンス』を参照してください。
- 「WITH句でのPL/SQLファンクションの使用例:」を参照してください。

subquery_factoring_clause

subquery_factoring_clauseを使用すると、副問合せブロックに名前(query_name)を割り当てることができます。query_nameを指定することによって、問合せに複数存在する副問合せブロックを参照することができます。query_nameをインライン・ビューまたは一時表として扱うことによって、問合せが最適化されます。query_nameには、データベース・スキーマ・オブジェクトと同じネーミング規則および制限が適用されます。データベース・オブジェクト名の詳細は、「データベース・オブジェクトのネーミング規則」を参照してください。

query_nameの後に続く列別名およびAS句内にある複数の副問合せを区切る集合演算子は有効ですが、再帰的副問合せのファクタリングを必要とします。search_clauseおよびcycle_clauseは、再帰的副問合せのファクタリングに対してのみ有効ですが、必須ではありません。「再帰的副問合せのファクタリング」を参照してください。

最上位のSELECT文およびほとんどの副問合せでこの句を指定できます。問合せの名前は、主問合せおよび後続のすべての副問合せから参照できます。再帰的副問合せのファクタリングの場合、問合せの名前は、自身の問合せ名を定義する副問合せからも参照できます。

再帰的副問合せのファクタリング

自身の問合せ名を定義する副問合せからsubquery_factoring_clauseが自身のquery_nameを参照する場合、そのsubquery_factoring_clauseは再帰的であるといいます。再帰的subquery_factoring_clauseには、2つの問合せブロック(1つ目のアンカー・メンバーおよび2つ目の再帰的メンバー)が含まれている必要があります。アンカー・メンバーは、再帰的メンバーの前に指定する必要があり、このメンバーはquery_nameを参照できません。アンカー・メンバーは、集合演算子UNION ALL、UNION、INTERSECTまたはMINUSによって結合された1つ以上の問合せブロックで構成できます。再帰的メンバーは、アンカー・メンバーの後に指定し、query_nameの参照は1回のみにする必要があります。UNION ALL集合演算子を使用して、再帰的メンバーをアンカー・メンバーに結合する必要があります。

WITH query_nameの後に続く列別名の数、およびアンカーのSELECT構文のリストと再帰的問合せブロックの数は同じである必要があります。

再帰的メンバーには、次の要素を含めることができません。

- DISTINCTキーワードまたはGROUP BY句
- model_clause
- 集計ファンクション。ただし、SELECT構文のリスト内には分析ファンクションを含めることができます。
- query_nameを参照する副問合せ。
- query_nameを右側の表として参照する外部結合。

Oracle Databaseの前のリリースでは、再帰的WITH句の再帰的メンバーは、問合せ全体の並列度に関係なくシリアルで実行されます(トップレベルのSELECT文とも呼ばれます)。Oracle Database 12cリリース2 (12.2)以降、オプティマイザによってトップレベルのSELECT文をパラレルで実行できることが検出された場合、再帰的メンバーはパラレルで実行されます。

search_clause

行の順序付けを指定するには、SEARCH句を使用します。

- 子の行が戻される前に兄弟の行を戻す必要がある場合は、BREADTH FIRST BYを指定します。
- 兄弟の行が戻される前に子の行を戻す必要がある場合は、DEPTH FIRST BYを指定します。
- 兄弟の行は、BYキーワードの後にリストされる列によって順序付けされます。
- SEARCHキーワードの後に続くc_aliasリストには、query_nameに対する列別名のリストからの列名を含める必要があります。
- ordering_columnは、問合せ名の列リストに自動的に追加されます。query_nameから選択する問合せは、ordering_columnにORDER BYを含めて、SEARCH句によって指定された順序で行を戻すことができます。

cycle_clause

CYCLE句は、再帰型での繰返しをマーク付けするために使用します。

- CYCLEキーワードの後に続くc_aliasリストには、query_nameに対する列別名のリストからの列名を含める必要があります。Oracle Databaseは、これらの列を使用して繰返しを検出します。
- cycle_valueおよびno_cycle_valueは、長さが1の文字列です。
- 繰返しが検出されると、繰返しを引き起こしている行に対してcycle_mark_c_aliasによって指定されている繰返しマーク列に、cycle_valueとして指定されている値が設定されます。その後、この行への再帰は停止します。つまり、問題の行については、その子の行の検索は行われませんが、繰返しが発生していない行については、検索が継続されます。
- 繰返しが検出されない場合、no_cycle_valueとして指定されているデフォルト値が繰返しマーク列に設定されます。
- この繰返しマーク列は、query_nameの列リストに自動的に追加されます。
- 祖先の行のいずれかの繰返し列に同じ値がある行は、繰返しを形成するとみなされます。

CYCLE句を指定しない場合に、繰返しが検出されると、再帰的WITH句はエラーを戻します。この場合、再帰的メンバーのWHERE句で参照されるquery_nameの列別名リストのすべての列について、祖先の行のいずれかに同じ値がある行は繰返しを形成します。

副問合せのファクタリングの制限事項

この句には、次の制限事項があります。

- subquery_factoring_clauseは、1つのSQL文内に1つのみ指定できます。subquery_factoring_clauseで定義したquery_nameは、そのsubquery_factoring_clause内の後続の任意の名前付き問合せブロックで使用できます。
- 集合演算子を指定した複合問合せの場合、その問合せを構成する各問合せではquery_nameを使用できませんが、各問合せのFROM句ではquery_nameを使用できます。
- query_nameに対する列別名のリストには重複した名前を指定できません。
- ordering_columnの名前は、cycle_mark_c_aliasとは異なる名前を使用する必要があります。
- ordering_columnおよび繰返しマーク列には、query_nameに対する列別名のリストに存在する名前を使用することはできません。

関連項目:

- インライン・ビューの詳細は、『Oracle Database概要』を参照してください。
- 「副問合せのファクタリング: 例」
- 「再帰的副問合せのファクタリング: 例」を参照してください。

subav_factoring_clause

subav_factoring_clauseを使用して、一時分析ビュー(集計前にファクト・データをフィルタするか、分析ビューの問合せに対して計算済メジャーを追加する)を定義できます。subav_name引数により、一時分析ビューに名前が割り当てられます。その後、問合せ内の複数の位置で一時分析ビューを参照するために、subav_nameを指定できます。subav_nameには、データベース・スキーマ・オブジェクトと同じネーミング規則および制限事項が適用されます。データベース・オブジェクト名の詳細は、「データベース・オブジェクトのネーミング規則」を参照してください。

最上位のSELECT文およびほとんどの副問合せでこの句を指定できます。問合せの名前は、主問合せおよび後続のすべての副問合せから参照できます。

subav_clause引数は、一時分析ビューを定義します。

subav_clause

USINGキーワードを使用して、分析ビュー(WITH句で事前に定義された一時分析ビュー、または永続分析ビューのいずれか)の名前を指定します。永続分析ビューは、CREATE ANALYTIC VIEW文で定義されます。永続分析ビューの場合、現行ユーザーはこのビューに対するSELECTアクセス権限を持っている必要があります。

関連項目:

分析ビュー: 例

hierarchies_clause

hierarchies_clauseは、ベース分析ビューの階層を指定します。この階層に基づいて一時分析ビューの結果がディメンション化されます。HIERARCHIESキーワードを使用して、ベース分析ビューの1つ以上の階層の別名を指定します。

HIERARCHIES句を指定しない場合、ベース分析ビューのデフォルトの階層が使用されます。

filter_clauses

特定のhier_aliasを最大で1つのfilter_clauseに指定できます。

filter_clause

filter句は指定された述語条件をファクト表に適用します。これにより、メジャー値の集計前に表から戻される行が減少します。述語には、任意のSQL行ファンクションまたは操作を指定できます。述語では、指定された階層の任意の属性を参照したり、MEASURESキーワードを指定した場合は分析ビューのメジャーを参照したりできます。

たとえば次の句は、メジャー値の集計を、時間階層の各年度の第1四半期および第2四半期の値に限定します。

```
FILTER FACT (time_hier TO quarter_of_year IN (1,2))
```

その後、一時分析ビューから2000年度および2001年度のsalesを選択すると、第1四半期と第2四半期の集計値のみが戻されます。

filter句でメジャーの述語を指定する例を次に示します。

```
FILTER FACT (MEASURES TO sales BETWEEN 100 AND 200)
```

attr_dim_alias

ベース分析ビューの属性ディメンションの別名。USER_ANALYTIC_VIEW_DIMENSIONSビューには、分析ビューの属性ディメンションの別名が含まれています。

hier_alias

ベース分析ビューの階層の別名。USER_ANALYTIC_VIEW_HIERSビューには、分析ビューの階層の別名が含まれています。

add_calcs_clause

ADD MEASURESキーワードを使用して、一時分析ビューに計算済メジャーを追加できます。

calc_meas_clause

計算済メジャーの名前と、計算済メジャーの値を指定する分析ビュー式を指定します。分析ビュー式には、「分析ビュー式」で説明する任意の有効なcalc_meas_expressionを指定できます。たとえば、次の例ではshare_salesという計算済メジャーを追加します。

```
ADD MEASURES (share_sales AS (SHARE_OF(sales HIERARCHY time_hier PARENT)))
```

hint

文の実行計画を選択する場合に、オプティマイザに指示を与えるためのコメントを指定します。

関連項目:

ヒントの構文および説明は、「ヒント」を参照してください。

DISTINCT | UNIQUE

DISTINCTまたはUNIQUEを指定すると、選択された重複行の1行のみを戻すことができます。これらの2つのキーワードは同義です。重複行とは、SELECT構文のリスト中のそれぞれの式で一致する値を持つ行のことです。

DISTINCT問合せおよびUNIQUE問合せの制限事項

これらのタイプの問合せには、次の制限事項があります。

- DISTINCTまたはUNIQUEを指定する場合、SELECT構文のリスト中の式すべての総バイト数は、データ・ブロックのサイズからオーバーヘッド分を引いたサイズに制限されます。このサイズは、初期化パラメータDB_BLOCK_SIZEによって指定されます。
- select_listにLOB列が含まれている場合、DISTINCTは指定できません。

ALL

ALLを指定すると、重複行を含め、選択されたすべての行を戻すことができます。デフォルトはALLです。

select_list

select_listでは、データベースから取り出す列を指定できます。

*(全列ワイルド・カード)

全列ワイルド・カード(アスタリスク)を指定すると、疑似列とINVISIBLE列を除いて、FROM句に指定されているすべての表、ビューまたはマテリアライズド・ビューのすべての列を選択できます。列は、表、ビューまたはマテリアライズド・ビューの*_TAB_COLUMNSデータ・ディクショナリ・ビューのCOLUMN_IDによって指定されている順序で戻されます。

ビューやマテリアライズド・ビューではなく表から選択する場合、ALTER TABLE SET UNUSED文によってUNUSEDのマークが付けられた列は選択されません。

関連項目:

「ALTER TABLE」、「単純な問合せの例」および「DUAL表からの選択: 例」を参照してください。

query_name.*

query_nameの後にピリオドおよびアスタリスクを指定すると、指定した副問合せブロックのすべての列を選択できます。query_nameには、subquery_factoring_clauseですでに指定されている副問合せブロック名を指定します。select_listでquery_nameを指定するには、subquery_factoring_clauseを指定する必要があります。select_listでquery_nameを指定するには、query_table_expression(FROM句)でもquery_nameを指定する必要があります。

table.* | view.* | materialized view.*

オブジェクト名の後にピリオドおよびアスタリスクを指定すると、指定した表、ビューまたはマテリアライズド・ビューのすべての列を選択できます。オブジェクトの作成時に指定された順序で列の集合が戻されます。2つ以上の表、ビューまたはマテリアライズド・ビューの行を選択する問合せを結合といいます。

他のユーザーのスキーマの表、ビューまたはマテリアライズド・ビューから選択する場合には、スキーマ修飾子を使用します。schemaを指定しない場合、この表、ビューおよびマテリアライズド・ビューは自分のスキーマ内にあるとみなされます。

関連項目:

「結合」

t_alias .*

相関名(別名)の後にピリオドとアスタリスクを指定すると、同じ副問合せのFROM句にその相関名が指定されているオブジェクトからのすべての列を選択できます。オブジェクトは、表、ビュー、マテリアライズド・ビューまたは副問合せのいずれかです。オブジェクトの作成時に指定された順序で列の集合が戻されます。2つ以上のオブジェクトの行を選択する問合せを結合といいます。

expr

選択する情報を表す式を指定します。リスト中の列が含まれている表、ビューまたはマテリアライズド・ビューがFROM句でschema名で指定されている場合のみ、その列名をschema名で指定できます。オブジェクト型のメンバー・メソッドを指定するときは、メソッドが引数を取らない場合でも、カッコを使用するメソッド名に従う必要があります。

式には、PL/SQLファンクションの値を返すスカラー値、行ごとに1つの値を返す副問合せおよびSQLマクロも含めることができます。

c_alias

列式に別名を指定します。この別名は、結果セットの列のヘッダーで使用されます。ASキーワードはオプションです。別名によって、問合せ中にSELECT構文のリストの項目名を効果的に変更できます。問合せにおいて、別名はorder_by_clauseで使用できますが、他の句では使用できません。

関連項目:

- 複数のマテリアライズド・ビューの問合せで、UNION ALL演算子とともにexpr AS c_alias構文を使用する場合の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください
- exprの構文は、「SQL式」を参照してください。

選択リストの制限事項

選択リストには、次の制限事項があります。

- この文にgroup_by_clauseも指定している場合、このSELECT構文のリストには次の式のみ指定できます。
    - 定数
    - 集計ファンクション、USERファンクション、UIDファンクションおよびSYSDATEファンクション
    - group_by_clauseに指定されているものと同じ式。group_by_clauseが副問合せの中にある場合、その副問合せのSELECT構文のリストにあるすべての列が副問合せのGROUP BY列と対応する必要があります。SELECT構文のリストおよびトップレベル問合せまたは副問合せのGROUP BY列が対応しない場合、その文ではORA-00979が発生します。
    - グループ内のすべての行が同じ値に評価される前述の式を伴っている式
- 結合内のキー保存表が1つのみの場合、結合ビューからROWIDを選択することができます。表のROWIDがビューのROWIDになります。

    関連項目:

    キー保存表の詳細は、『Oracle Database管理者ガイド』を参照してください。
- 複数の表に同じ名前の列があり、FROM句で結合を指定した場合、表の名前または表の別名でその列名を修飾する必要があります。

FROM句

FROM句を指定すると、どのオブジェクトからデータを選択するかを指定できます。

多相表ファンクション(PTF)は、他の既存の表ファンクションと同様に、FROM句の問合せブロックで呼び出すことができます。PTFは、オペランドに複数の型を使用できる表ファンクションです。

Oracle Databaseリリース19c, バージョン19.7以降、表値マクロを作成してFROM句内で使用できます。この句では、PL/SQLファンクションをコールできます。SQL表マクロは、通常FROM句で使用される式で、多相(パラメータ化)ビューのように機能します。これらのマクロ・ファンクションをPL/SQLで定義し、SQLからコールしてマクロとして機能させる必要があります。

関連項目:

- PL/SQLの最適化とチューニング
- SQL_MACRO句

ONLY

ONLY句は、ビューのみに適用されます。FROM句のビューが階層に属し、サブビューの行を含めない場合は、ONLY句を使用します。

query_table_expression

query_table_expression句を使用すると、副問合せブロック、表、ビュー、マテリアライズド・ビュー、分析ビュー、階層、パーティションまたはサブパーティションを識別したり、オブジェクトを識別する副問合せを指定できます。副問合せブロックを指定するには、副問合せブロック名(subquery_factoring_clauseのquery_nameまたはsubav_factoring_clauseのsubav_name)を指定する必要があります。

この式の分析ビューには、with_clauseで定義されている一時分析ビュー、あるいは永続分析ビューを指定できます。

関連項目:

「副問合せの使用方法: 例」

LATERAL

LATERALを指定すると、LATERALインライン・ビューとしてsubqueryを指定できます。LATERALインライン・ビュー内では、そのLATERALインライン・ビューの左側に現れる表を、問合せのFROM句で指定できます。この左相関はsubquery(SELECT句、FROM句、WHERE句など)内のどこにでも、任意のネスト・レベルで指定できます。

LATERALの制限事項

LATERALインライン・ビューには、次の制限事項があります。

- LATERALを指定すると、pivot_clause句、またはunpivot_clause句が指定できなくなります。また、table_reference句にパターンを指定できなくなります。
- LATERALインライン・ビューにquery_partition_clauseが含まれていて、このビューが結合句の右側になる場合、その結合句には左側の表への左相関を含めることができません。ただし、表への左相関は、左側の表でない場合はFROM句の左側に含めることができます。
- LATERALインライン・ビューには、右外部結合または完全外部結合の最初の表への左相関を含めることができません。

関連項目:

LATERALインライン・ビューの使用方法: 例

inline_external_table

問合せで外部表をインライン化するには、この句を指定します。問合せでインライン化する外部表の表列およびプロパティを指定する必要があります。

inline_external_table_properties

この句は、REJECT LIMITオプションおよびaccess_driver_typeオプションを使用してexternal_table_data_propsを拡張します。この句を使用して外部表のプロパティを指定します。

オペレーティング・ファイル・システムやビッグ・データ・ソースに存在する外部データ、およびHDFSやHiveなどの形式をサポートする以外に、Oracleでは、オブジェクトに存在する外部データをサポートします。

modified_external_table

この句を使用して、問合せ内からCREATE TABLE文またはALTER TABLE文で指定されている外部表プロパティの一部をオーバーライドできます。

外部表のパラメータは実行時にオーバーライドできます。

制限事項

- 問合せでキーワードEXTERNAL MODIFYを指定する必要があります。このキーワードを指定しない場合、Missing or invalid optionエラーが表示されます。
- 問合せで外部表を参照する必要があります。このようにしないと、エラーが表示されます。
- 問合せでは少なくとも1つのプロパティを指定する必要があります。DEFAULT DIRECTORY、LOCATION、ACCESS PARAMETERS、REJECT LIMITのいずれかです。
- 複数の外部表プロパティを指定する場合は、順にリストする必要があります。DEFAULT DIRECTORYを最初に指定し、続いてACCESS PARAMETERS、LOCATION、REJECT LIMITの順に指定する必要があります。このようにしないと、エラーが発生します。
- DEFAULT DIRECTORY句では、適切なデフォルト・ディレクトリを1つのみ指定する必要があります。このようにしないと、Missing DEFAULT keywordエラーが発生します。
- LOCATION句のファイル名は、引用符で囲む必要があります。このようにしないと、Missing keywordエラーが発生します。問合せでLOCATION句を許可するかどうかはアクセス・ドライバによって決定されることに注意してください。特定のアクセス・ドライバでこの句が許可されていない場合は、エラーが発生します。
- ORACLE_LOADERアクセス・ドライバおよびORACLE_DATAPUMPアクセス・ドライバの場合、LOCATION句の外部ファイルの場所は、directory: locationの形式で指定する必要があります（ディレクトリとロケーションをコロンで区切ります)。この句で複数の値を指定する場合は、カンマで区切る必要があります。このようにしないと、Missing keywordエラーが発生します。
- LOCATIONはCREATE TABLEではオプションになり、外部表を作成または問い合せる場合には指定する必要がある点に注意してください。このようにしないと、アクセス・ドライバでエラーが発生します。
- CTASからORACLE DATAPUMPを使用して外部データを移入する場合は、外部ファイルの場所を指定する必要があります。これが該当するのは、CREATE TABLEでLOCATION句が必須である場合のみです。

- アクセス・パラメータをオーバーライドするときには、ACCESS PARAMETERS句に、適切なアクセス・パラメータのリストをカッコで囲んで指定する必要があります。

  modified_external_table句でのアクセス・パラメータの構文と許容値は、各アクセス・ドライバの外部表DDLの場合と同じであることに注意してください。構文と許容値の詳細は、『Oracle Databaseユーティリティ』を参照してください。

- REJECT LIMITを指定する場合、これはUNLIMITEDまたは範囲内の有効な値である必要があります。それ以外の場合、Reject limit out of rangeエラーが発生します。

modify_external_table_properties

この句を使用して、実行時に変更する外部表プロパティを指定できます。変更可能なパラメータは、DEFAULT DIRECTORY、LOCATION、ACCESS PARAMETERS (BADFILE, LOGFILE, DISCARDFILE)およびREJECT LIMITです。

例: 問合せによる外部表パラメータのオーバーライド

```
SELECT * FROM
sales_external EXTERNAL MODIFY (LOCATION 'sales_9.csv’ REJECT LIMIT UNLIMITED);
```

flashback_query_clause

flashback_query_clauseを使用すると、データに関連付けられた時間ディメンションに基づいて、表、ビューまたはマテリアライズド・ビューからデータを取得できます。

この句によってSQL駆動のフラッシュバックが実装されるため、次の項目を指定できます。

- 句VERSIONS BETWEEN { SCN | TIMESTAMP }またはVERSIONS AS OF { SCN | TIMESTAMP }を使用して、SELECT構文のリストに含まれるオブジェクトごとに、異なるシステム変更番号またはタイムスタンプを指定できます。また、セッション・レベルのフラッシュバックをDBMS_FLASHBACKパッケージを使用して実装できます。
- 句VERSIONS PERIOD FORまたはAS OF PERIOD FORを使用して、SELECT構文のリストに含まれるオブジェクトごとに有効期間を指定できます。また、セッション・レベルの有効期間フラッシュバックをDBMS_FLASHBACK_ARCHIVEパッケージを使用して実装できます。

フラッシュバック問合せを使用すると、行に対して行った変更の履歴を取り出すことができます。VERSIONS_XID疑似列を使用して、変更を行ったトランザクションの対応する識別子を取り出すことができます。また、Oracle Flashback Transaction Queryを発行して、特定の行バージョンを生成したトランザクションの情報を取り出すこともできます。これを行うには、特定のトランザクションIDをFLASHBACK_TRANSACTION_QUERYデータ・ディクショナリ・ビューで問い合せます。

VERSIONS BETWEEN { SCN | TIMESTAMP }

VERSIONS BETWEENを指定すると、問合せによって戻された行の複数のバージョンを取り出すことができます。2つのSCNまたは2つのタイムスタンプ値の間に存在する、行のすべてのコミット済バージョンが戻されます。最初に指定されたSCNまたはタイムスタンプは、2番目に指定されたSCNまたはタイムスタンプよりも前でなければなりません。戻された行には、削除後に再度挿入された行のバージョンが含まれます。

- VERSIONS BETWEEN SCN ...を指定すると、2つのSCNの間に存在する行のバージョンを取り出すことができます。どちらの式も、評価結果は数値でなければならず、評価結果がNULLであってはなりません。MINVALUEおよびMAXVALUEは、それぞれ使用可能な一番古いデータおよび最新のデータのSCNに解決されます。
- VERSIONS BETWEEN TIMESTAMP ...を指定すると、2つのタイムスタンプの間に存在する行のバージョンを取り出すことができます。どちらの式も、評価結果はタイムスタンプ値でなければならず、評価結果がNULLであってはなりませ

ん。MINVALUEおよびMAXVALUEは、それぞれ使用可能な一番古いデータおよび最新のデータのタイムスタンプに解決されます。

AS OF { SCN | TIMESTAMP }

AS OFを指定すると、特定のシステム変更番号(SCN)またはタイムスタンプでの問合せによって戻された行の単一のバージョンを取り出すことができます。SCNを指定する場合、exprは数値に評価される必要があります。TIMESTAMPを指定する場合、exprはタイムスタンプ値に評価される必要があります。いずれの場合も、exprの評価結果がNULLであってはなりません。指定されたシステム変更番号または時刻に存在した行が戻されます。

Oracle Databaseでは、バージョン問合せ疑似列のグループを使用して、様々な行のバージョンに関する追加情報を取り出すことができます。詳細は、「バージョン問合せ疑似列」を参照してください。

両方の句を同時に使用する場合、AS OF句によって、SCNまたはデータベースが問合せを発行した時点が判断されます。VERSIONS句によって、AS OFで指定した時点を基準とした行のバージョンが判断されます。トランザクションが、BETWEENの最初の値より前に開始したり、AS OFで指定した時点より後に終了した場合は、行のバージョンとしてNULLが戻されます。

VERSIONS PERIOD FOR

VERSIONS PERIOD FORを指定すると、特定の期間中に有効とみなされるかどうかに基づいて、tableから行を取り出すことができます。この句を使用するには、tableで時制有効性がサポートされている必要があります。

- valid_time_columnには、tableの有効期間ディメンション列の名前を指定します。
- 行が有効と見なされる期間を指定するには、BETWEEN句を使用します。どちらの式も、評価結果はタイムスタンプ値でなければならず、評価結果がNULLであってはなりません。MINVALUEは、tableの開始時間列に含まれる最も早い日付またはタイムスタンプに解決されます。MAXVALUEは、tableの終了時間列に含まれる最も遅い日付またはタイムスタンプに解決されます。

AS OF PERIOD FOR

AS OF PERIOD FORを指定すると、特定の時点で有効とみなされるかどうかに基づいて、tableから行を取り出すことができます。この句を使用するには、tableで時制有効性がサポートされている必要があります。

- valid_time_columnには、tableの有効期間ディメンション列の名前を指定します。
- 行が有効と見なされる時点を指定するには、exprを使用します。この式の評価結果はタイムスタンプ値でなければならず、評価結果がNULLであってはなりません。

関連項目:

- 時制有効性の詳細は、『Oracle Database開発ガイド』を参照してください。
- 時制有効性をサポートするために表を構成する方法を学習するには、およびvalid_time_column、開始時間列および終了時間列に関する情報は、CREATE TABLE period_definitionを参照してください。

フラッシュバック問合せのノート

フラッシュバック問合せの実行時、他のタイプの問合せとは異なり、問合せ最適化が使用されない場合があります。ここで問合せ最適化を使用すると、パフォーマンスが低下する可能性があります。これは特に、階層問合せに複数のフラッシュバック問合せを指定した場合に発生します。

フラッシュバック問合せの制限事項

これらの問合せには、次の制限事項があります。

- 列式や副問合せをAS OF句の式で指定することはできません。
- AS OF句を指定する場合、for_update_clause句は指定できません。
- AS OF句は、マテリアライズド・ビューを定義する問合せの中では使用できません。
- 一時表、外部表またはクラスタの一部である表に対するフラッシュバック問合せでは、VERSIONS句を使用できません。
- ビューに対するフラッシュバック問合せではVERSIONS句を使用できません。ただし、ビューを定義する問合せには、VERSIONS構文を使用できます。
- すでにquery_nameがquery_table_expressionの中で指定されている場合は、flashback_query_clauseは指定できません。

関連項目:

- Oracleフラッシュ・バック問合せの詳細は、『Oracle Database開発ガイド』を参照してください。
- 「フラッシュバック問合せの使用方法: 例」を参照してください。
- DBMS_FLASHBACKパッケージを使用するセッション・レベルのフラッシュバックの詳細は、『Oracle Database開発ガイド』および『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- トランザクション履歴の詳細は、『Oracle Database管理者ガイド』、および『Oracle Databaseリファレンス』のFLASHBACK_TRANSACTION_QUERYに関する説明を参照してください。

partition_extension_clause

PARTITIONまたはSUBPARTITIONには、データを取得するtable内のパーティションやサブパーティションの名前またはキー値を指定します。

レンジ・パーティション・データおよびリスト・パーティション・データでは、この句のかわりに、データの取出しをtableの1つ以上のパーティションに制限する条件をWHERE句に指定できます。Oracle Databaseは、この条件を解析して、そのパーティションからのデータのみをフェッチします。そのようなWHERE条件を、ハッシュ・パーティション・データに対して形成することは不可能です。

関連項目:

「パーティション表と索引の参照」および「パーティションからの選択: 例」を参照してください。

dblink

表、ビューまたはマテリアライズド・ビューが存在するリモート・データベースのデータベース・リンクの完全名または部分名を指定します。このデータベースは、Oracle Databaseである必要はありません。

関連項目:

- データベース・リンクの参照方法の詳細は、「リモート・データベース内のオブジェクトの参照」を参照してください。
- 分散問合せの詳細は、「分散問合せ」を参照してください。また、「分散問合せの使用方法: 例」を参照してください。

dblinkを指定しない場合、その表、ビューまたはマテリアライズド・ビューは、ローカル・データベース内にあるものとみなされます。

データベース・リンクの制限事項

データベース・リンクには、次の制限事項があります。

- リモート表のユーザー定義型またはオブジェクトREFを問い合せることはできません。
- リモート表のANYTYPE型、ANYDATA型またはANYDATASET型の列を問い合せることはできません。

table | view | materialized_view | analytic_view | hierarchy

データの選択元となる表、ビュー、マテリアライズド・ビュー、分析ビューまたは階層の名前を指定します。

analytic_view

CREATE ANALYTIC VIEW文で定義された永続分析ビュー、またはWITH句で定義された一時分析ビュー。

関連項目:

分析ビュー: 例

hierarchy

CREATE HIERARCHY文で定義された階層。

sample_clause

sample_clauseを指定すると、表全体からではなく、表のランダムなサンプル・データから選択が行われます。

関連項目:

サンプルの選択: 例

BLOCK

BLOCKを指定すると、ランダムな行サンプリングのかわりに、ランダムなブロック・サンプリングを実行できます。

ブロック・サンプリングは、全表スキャン中または高速全索引スキャン中にのみ使用可能です。より効率的な実行パスが存在する場合、ブロック・サンプリングは実行されません。特定の表または索引に対するブロック・サンプリングを確実に実行する場合は、FULLまたはINDEX_FFSのヒントを使用します。

Oracle Database 12cリリース2 (12.2)以降は、外部表にブロック・サンプリングを指定できます。以前のリリースでは、外部表にブロック・サンプリングを指定しても効果はなく、行サンプリングが実行されていました。

sample_percent

sample_percentには、全体の行またはブロック数のうち、サンプルに入れる割合(%)を指定します。0.000001以上100未満の範囲の値を指定します。この割合は、各行(ブロック・サンプリングの場合は行の各クラスタ)が、サンプルの一部として選択される可能性を示します。sample_percentに指定した割合の行がtableから正確に取り出されるわけではありません。

警告:

統計的に適切でない想定値でこの機能を使用した場合、不正確な、または望ましくない結果になります。

SEED seed_value

この句を指定すると、実行ごとに同じサンプルを戻すことを試行するようにデータベースに指示できます。seed_valueには、0(ゼロ)から4294967295の整数を指定します。この句を省略した場合、戻されるサンプルは実行ごとに異なります。

sample_clauseの制限事項

SAMPLE句には、次の制限事項が適用されます。

- SAMPLE句は、DML文の副問合せの中では指定できません。
- SAMPLE句を問合せで指定できるのは、問合せの対象が実表、マテリアライズド・ビューのコンテナ表、またはキー保存であるビューである場合です。この句は、キー保存ではないビューに対しては指定できません。

subquery_restriction_clause

subquery_restriction_clauseを使用すると、次のいずれかの方法で副問合せを制限できます。

WITH READ ONLY

WITH READ ONLYを指定すると、表またはビューを更新禁止にできます。

WITH CHECK OPTION

WITH CHECK OPTIONを指定すると、副問合せに含まれない行を生成する表またはビューの変更を禁止できます。この句をDML文の副問合せ内で使用する場合、FROM句内の副問合せには指定できますが、WHERE句内の副問合せには指定できません。

CONSTRAINT constraint

CHECK OPTION制約の名前を指定します。この識別子を省略した場合は、Oracleによって自動的にSYS_Cnという形式の制約名が割り当てられます(nはデータベース内で制約名を一意にするための整数)。

関連項目:

WITH CHECK OPTION句の使用方法: 例

table_collection_expression

table_collection_expressionを使用すると、問合せおよびDML操作で、collection_expression値を表として扱うことができます。collection_expressionには、副問合せ、列、ファンクションまたはコレクション・コンストラクタのいずれかを指定できます。その形式にかかわらず、集合値(ネストした表型またはVARRAY型の値)を戻す必要があります。このようなコレクションの要素抽出プロセスをコレクション・ネスト解除といいます。

TABLEコレクション式を親表と結合する場合は、オプションのプラス(+)には大きな意味があります。+を指定すると、その2つの外部結合が作成され、コレクション式がNULLの場合でも、外部表の行が問合せで戻されるようになります。

ノート:

以前のリリースの Oracle では、collection_expression が副問合せの場合、table_collection_expression を THE subquery と表現していました。現在、このような表現方法は非推奨になっています。

collection_expressionは、FROM句で左側に定義された表の列を参照できます。これを左相関といいます。左相関はtable_collection_expressionのみで行われます。その他の副問合せは、その副問合せ以外で定義された列を参照することはできません。

オプションの(+)を使用すると、コレクションがNULLまたは空である場合、すべてのフィールドにNULLが設定された行をtable_collection_expressionが戻すように指定できます。この(+)はcollection_expressionが左相関を使用する場合にのみ有効です。結果は、外部結合の結果と似ています。

UPDATEまたはDELETE操作で副問合せのWHERE句に(+)構文を使用する場合は、副問合せのFROM句に2つの表を指定する必要があります。副問合せに結合が存在しないかぎり、外部結合構文は無視されます。

関連項目:

- 「外部結合」
- 「表のコレクション: 例」および「コレクション・ネスト解除: 例」を参照してください。

t_alias

相関名(表、ビュー、マテリアライズド・ビューまたは問合せを評価するための副問合せの別名)を指定します。SELECT構文のリストがオブジェクト型属性またはオブジェクト型メソッドを参照する場合、この別名が必要になります。相関名は、相関問合せ内で最も頻繁に使用されます。表、ビューまたはマテリアライズド・ビューを参照する問合せでは、この別名を参照する必要があります。

関連項目:

相関副問合せの使用方法: 例

pivot_clause

pivot_clauseを使用すると、行を列に変換し、変換処理中にデータを集計するクロス集計問合せを記述できます。ピボット演算の出力には、最初のデータセットよりも多くの列と少ない行が含まれています。pivot_clauseでは、次のステップが実行されます。

- pivot_clauseの先頭で指定されている集計ファンクションが計算されます。集計ファンクションは、複数の値を戻すようにGROUP BY句を指定する必要がありますが、pivot_clauseには、明示的なGROUP BY句が含まれていません。かわりに、暗黙的なGROUP BYが実行されます。暗黙的なグループ化は、pivot_clauseで参照されていないすべての列、およびpivot_in_clauseで指定されている値セットに基づいています。複数の集計ファンクションを指定する場合、集計ファンクションの1つを除くすべてに別名を指定する必要があります。
- 列のグループ化およびステップ1で計算された集計値は、次のクロス集計出力を生成するように構成されています。
  - 最初に、pivot_clauseで参照されていないすべての暗黙的なグループ化列が出力されます。
  - pivot_in_clauseの値に対応する新しい列。各集計値がクロス集計の適切な新しい列に移動します。XMLキーワードを指定した場合は、結果は新しい列1つだけとなり、データは1つのXML文字列として表現されます。データベースは、新しい列ごとに名前を生成します。集計ファンクションの別名を指定しない場合、

データベースは、集計値を移動する新しい列ごとの名前としてピボット列値を使用します。集計ファンクションの別名を指定する場合、データベースは、ピボット列名、アンダースコア(_)、集計ファンクション別名を連結して集計値を移動する新しい列ごとの名前を生成します。生成された列名が列名の最大長を超える場合、ORA-00918エラーが戻されます。この問題を回避するには、ピボット列見出しまたは集計ファンクションあるいはその両方に短い別名を指定します。

pivot_clauseの副次句のセマンティクスは、次のとおりです。

XML

オプションのXMLキーワードは、問合せのXML出力を生成します。XMLキーワードを指定すると、pivot_in_clauseには、副問合せまたはワイルド・カード・キーワードANYを含めることができます。副問合せおよびANYワイルド・カードは、pivot_in_clause値が事前にわかっていない場合に有効です。XML出力では、ピボット列の値が実行時に評価されます。pivot_in_clauseで式を使用して明示的なピボット値を指定する場合は、XMLを指定することができません。

XML出力が生成される際、集計ファンクションが各ピボット値に適用され、データベースによって、値とメジャーのすべてのペアのXML文字列を含むXMLTypeの列が戻されます。

expr

ピボット列の定数値への評価を行う式を指定します。オプションで、各ピボット列値の別名を指定できます。別名がない場合は、列ヘッダーが引用識別子となります。

subquery

subqueryは、XMLキーワードとともにのみ使用されます。subqueryを指定すると、subqueryによって検出されたすべての値がピボットに使用されます。出力は、XML以外のピボット問合せによって戻されるクロス集計書式とは異なります。pivot_in_clauseで指定されている複数の列のかわりに、subqueryでは、XML文字列の列が1つ生成されます。各行のXML文字列は、その行の暗黙的なGROUP BY値に対応する集計データを保持します。入力データに対応する行がない場合でも、各出力行のXML文字列には、subqueryによって検出されたすべてのピボット値が含まれています。

subqueryは、ピボット問合せの実行時に、一意の値リストを戻します。subqueryが一意の値を戻さない場合、Oracle Databaseによってランタイム・エラーが生成されます。問合せが一意の値を戻すかどうかがわからない場合は、subqueryにDISTINCTキーワードを使用します。

ANY

ANYキーワードは、XMLキーワードとともにのみ使用されます。ANYキーワードは、ワイルド・カードとして機能し、subqueryと同様に動作します。出力は、XML以外のピボット問合せによって戻されるクロス集計書式とは異なります。pivot_in_clauseで指定されている複数の列のかわりに、ANYキーワードでは、XML文字列の列が1つ生成されます。各行のXML文字列は、その行の暗黙的なGROUP BY値に対応する集計データを保持します。ただし、subqueryを指定した場合と比較すると、ANYワイルド・カードでは、各出力行について、行に対応する入力データで検出されたピボット値のみを含むXML文字列が生成されます。

関連項目:

PIVOTおよびUNPIVOTの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。また、「PIVOTおよびUNPIVOTの使用方法: 例」を参照してください

unpivot_clause

unpivot_clauseは、列を行に変換します。

- INCLUDE | EXCLUDE NULLS句を使用すると、NULL値の行を含めるか除外するかを選択できます。INCLUDE NULLSを指定すると、NULL値の行もアンピボット操作の対象となり、EXCLUDE NULLSを指定するとNULL値の行は戻り値のセットから除外されます。この句を省略した場合は、アンピボット操作からNULLが除外されます。
- columnには、sales_quantityなどのメジャー値を保持する各出力列の名前を指定します。
- pivot_for_clauseには、四半期または製品などの記述子値を保持する各出力列の名前を指定します。
- unpivot_in_clauseには、名前がpivot_for_clauseの出力列の値となる入力データ列を指定します。これらの入力データ列には、Q1、Q2、Q3、Q4など、カテゴリ値を指定する名前が含まれています。任意指定のAS句を使用すると、入力データ列名を、出力列内の指定したliteral値にマッピングできます。

アンピボット操作は、複数の値列を単一の列に変更します。このため、値列のすべてのデータ型は、数値、文字などの同じデータ型グループに属している必要があります。

- すべての値列がCHARの場合、アンピボットされる列はCHARになります。値列がVARCHAR2の場合、アンピボットされる列はVARCHAR2になります。
- すべての値列がNUMBERの場合、アンピボットされる列はNUMBERになります。値列がBINARY_DOUBLEの場合、アンピボットされる列はBINARY_DOUBLEになります。BINARY_DOUBLEの値列はないが、いずれかの値列がBINARY_FLOATの場合、アンピボットされる列はBINARY_FLOATになります。

containers_clause

CONTAINERS句は、マルチテナント・コンテナ・データベース(CDB)で有益です。この句を使用すると、CDBのすべてのコンテナ間で指定された表またはビューのデータを問い合せることができます。

- CDBのデータを問い合せるには、CDBルートに接続された共通ユーザーであり、表またはビューがルートおよびすべてのPDBに存在している必要があります。問合せは、CDBルートおよびすべてのオープン状態になっているPDBの表またはビューのすべての行を戻します。
- アプリケーション・コンテナのデータを問い合せるには、アプリケーション・ルートに接続された共通ユーザーであり、表またはビューがアプリケーション・ルートおよびアプリケーション・コンテナ内のすべてのPDBに存在している必要があります。この問合せは、アプリケーション・ルートおよびアプリケーション・コンテナ内のオープン状態になっているすべてのPDBに存在する表またはビューのすべての行を戻します。

表またはビューは、自分のスキーマ内に存在している必要があります。schemaを指定する必要はありませんが、指定した場合は自分のスキーマを指定する必要があります。

問合せは、ルートおよびすべてのオープン状態になっているPDB (RESTRICTEDモードでオープン状態になっているPDBは除きます)の表またはビューのすべての行を戻します。問合せ対象の表またはビューにCON_ID列が含まれていない場合、問合せはCON_ID列を問合せ結果に追加し、与えられた行が表すデータがあるコンテナを識別します。

関連項目:

- CONTAINERSヒント
- CONTAINERS句の詳細は、『Oracle Database管理者ガイド』を参照してください。

shards_clause

シャード間のV$、DBA/USER/ALLビュー、ディクショナリ表などのOracle提供オブジェクトを問い合せるには、shards_clauseを使用します。shards_clauseを使用した問合せは、シャード・カタログ・データベースに対してのみ実行

できます。

この機能を使用すると、中央のシャード・カタログからすべてのシャードにわたって問合せを実行できるようになり、集中管理が容易になります。

join_clause

適切なjoin_clause構文を使用すると、データが選択され、結合の一部となる表を識別できます。inner_cross_join_clauseを使用すると、内部結合またはクロス結合を指定できます。outer_join_clauseを使用すると、外部結合を指定できます。cross_outer_apply_clauseを使用すると、ANSI CROSS JOINまたは左相関のサポートを利用したANSI LEFT OUTER JOINを指定できます。

結合する行ソースが3つ以上ある場合は、カッコを使用してデフォルトの優先順位を無効にすることができます。たとえば、次のような構文があるとします。

```
SELECT ... FROM a JOIN (b JOIN c) ...
```

この場合、bとcが結合され、次にその結果とaが結合されます。

関連項目:

結合の詳細は、「結合」を参照してください。また、「結合問合せの使用方法: 例」、「自己結合の使用方法: 例」および「外部結合の使用方法: 例」を参照してください。

inner_cross_join_clause

内部結合は、結合条件を満たす行のみを戻します。

INNER

INNERを指定すると、内部結合を明示的に指定できます。

JOIN

JOINキーワードを使用すると、結合の実行を明示的に示すことができます。この構文を使用すると、WHERE句の結合で使用されている、カンマで区切られた表の式を、FROM句の結合構文に置き換えることができます。

ON条件

ON句を使用して結合条件を指定します。これにより、WHERE句の検索条件またはフィルタ条件とは別個に結合条件を指定できます。

USING (列)

両方の表で同じ名前の列同士を等価結合する場合、USING column句に使用する列を指定します。両方の表で同じ名前の列同士を結合する場合のみ、この句を使用できます。この句の中では、列名を表の名前および別名で修飾しないでください。

CROSS

CROSSキーワードは、クロス結合を実行することを示します。クロス結合とは、2つの関係(リレーション)のクロス積を生成するものであり、実質的にはカンマ区切りのOracle Database表記法と同じです。

NATURAL

NATURALキーワードは、自然結合を実行することを示します。この句のセマンティクスの詳細は、「NATURAL」を参照してくだ

さい。

outer_join_clause

外部結合は、結合条件を満たすすべての行と、結合条件を満たす他方の表の行を除いた、一方の表のすべての行を戻します。指定可能な外部結合は、結合の両側にtable_reference構文を使用した従来の外部結合と、いずれかの側にquery_partition_clauseを使用したパーティション化された外部結合の2種類です。パーティション化された外部結合は、内部表の各パーティションと外部表の間で結合が行われるという点を除いて、従来の外部結合と同じです。この形式の結合では、対象のディメンションに沿って、選択的に疎データをより密にできます。このプロセスはデータの稠密化といいます。

query_partition_clause

query_partition_clauseを使用すると、パーティション化された外部結合を定義できます。このような結合は、問合せによって戻されたパーティションに外部結合を適用し、従来の外部結合構文を拡張します。PARTITION BY句で指定した各式に対する行のパーティションが作成されます。問合せの各パーティションの行は、PARTITION BY式に対して同じ値を持ちます。

query_partition_clauseは、外部結合のいずれかの側で使用できます。パーティション化された外部結合の結果は、パーティション化された結果セットの各パーティションと結合の反対側の表との外部結合のUNIONになります。この形式の結果は、疎データの欠損の補完に役立つため、分析計算が簡単になります。

この句を省略した場合、表の式全体(table_referenceに指定したすべてのもの)が単一のパーティションとして扱われるため、従来型の外部結合となります。

分析ファンクションでquery_partition_clauseを使用するには、構文の上位ブランチ(カッコなし)を使用します。この句をモデルの問合せ(model_column_clauses内)またはパーティション化された外部結合(outer_join_clause内)で使用するには、構文の下位ブランチ(カッコ付き)を使用します。

パーティション化された外部結合の制限事項

パーティション化された外部結合には、次の制限事項があります。

- query_partition_clauseは、結合の右側または左側に指定できますが、両方に指定することはできません。
- パーティション化された完全外部結合(FULL)は指定できません。
- ON句を使用して外部結合にquery_partition_clauseを指定した場合、ON条件内には副問合せを指定できません。

関連項目:

パーティション化された外部結合の使用方法: 例

NATURAL

NATURALキーワードは、自然結合を実行することを示します。自然結合は、2つの表の間で同じ名前のすべての列に基づきます。2つの表から関連する列の値が等しい行が選択されます。同じ名前の2つの列のデータ型の間に互換性がない場合は、エラーが発生します。自然結合で使用する列を指定する場合は、表の名前または別名で列名を修飾しないでください。

自然結合またはクロス結合の表の組合せが不明瞭な場合があります。たとえば、次のような結合構文があるとします。

```
   a NATURAL LEFT JOIN b LEFT JOIN c ON b.c1 = c.c1
```

この例は、次のどちらにも解釈できます。

```
   a NATURAL LEFT JOIN (b LEFT JOIN c ON b.c1 = c.c1)
   (a NATURAL LEFT JOIN b) LEFT JOIN c ON b.c1 = c.c1
```

このような不明瞭さをなくすため、カッコを使用して結合する表の組合せを明確にしてください。このようなカッコがないと、左から右へ表が組み合せられ、左の結合が使用されます。

自然結合の制限事項

LOB列、ANYTYPE列、ANYDATA列、ANYDATASET列またはコレクション列は、自然結合の一部として指定できません。

outer_join_type

outer_join_typeは、実行する外部結合の種類を示します。

- RIGHTを指定すると、右側外部結合が実行されます。
- LEFTを指定すると、左側外部結合が実行されます。
- FULLを指定すると、完全な外部結合または両側外部結合が実行されます。内部結合に加え、内部結合の結果に戻されない両方の表からの行は、保持され、NULLで拡張されます。
- RIGHT、LEFTまたはFULLの後にオプションのOUTERキーワードを指定し、外部結合の実行を明示的に示すことができます。

ON条件

ON句を使用して結合条件を指定します。これにより、WHERE句の検索条件またはフィルタ条件とは別個に結合条件を指定できます。

ON条件句の制限事項

NATURAL外部結合を使用してこの句を指定することはできません。

USING列

USING句による外部結合の場合、問合せによって単一列が戻されます。この単一列は、結合内の一致する2つの列が結合したものです。結合ファンクションは次のとおりです。

```
COALESCE (a, b) = a if a NOT NULL, else b.
```

そのため、次のようになります。

- 左側外部結合では、FROM句内の左側の表から共通するすべての列値が戻されます。
- 右側外部結合では、FROM句内の右側の表から共通するすべての列値が戻されます。
- 完全な外部結合では、結合された両方の表から共通するすべての列値が戻されます。

USING column 句の制限事項

USING column句には、次の制限事項があります。

- この句の中では、列名を表の名前および別名で修飾しないでください。
- LOB列またはコレクション列は、USING column句で指定できません。
- NATURAL外部結合を使用してこの句を指定することはできません。

関連項目:

- 外部結合に関するその他の規則および制限事項は、「外部結合」を参照してください。
- パーティション化された外部結合およびデータの稠密化の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- 「外部結合の使用方法: 例」を参照してください。

cross_outer_apply_clause

この句を使用すると、ANSI CROSS JOINまたは左相関のサポートを利用したANSI LEFT OUTER JOINを実行できます。APPLYキーワードの右側には、table_referenceまたはcollection_expressionを指定できます。table_referenceには、表、インライン・ビューまたはTABLEコレクション式を指定できます。collection_expressionには、副問合せ、列、ファンクションまたはコレクション・コンストラクタのいずれかを指定できます。その形式にかかわらず、集合値(ネストした表型またはVARRAY型の値)を戻す必要があります。table_referenceまたはcollection_expressionは、FROM句でAPPLYキーワードの左側に定義された表の列を参照できます。これを左相関といいます。

11. CROSS APPLYを指定すると、ANSI CROSS JOINを実行できます。結合の左側にある表の行のうち、table_referenceまたはcollection_expressionから結果セットを生成する行のみが返されます。
12. OUTER APPLYを指定すると、ANSI LEFT OUTER JOINを実行できます。結合の左側にある表の行がすべて返されます。table_referenceまたはcollection_expressionから結果を生成しない行は、対応する列内でNULL値を保持します。

cross_outer_apply_clauseの制限事項

table_referenceに、LATERALインライン・ビューを指定することはできません。

関連項目:

CROSS APPLYおよびOUTER APPLY結合の使用例

inline_analytic_view

インライン分析ビューは、FROM句で指定される一時分析ビューです。インライン分析ビューを作成するには、ANALYTIC VIEWキーワードを使用し、分析ビューを定義するsubav_clauseを指定します。必要に応じて、inline_av_alias (インライン分析ビューの別名)を指定できます。inline_av_aliasのルールは、インライン・ビュー別名のルールと同じです。

関連項目:

分析ビュー: 例

where_clause

WHERE条件を指定すると、選択する行を1つ以上の条件を満たす行のみに制限できます。conditionには、有効なSQL条件を指定します。

この句を省略した場合、FROM句に指定されている表、ビューまたはマテリアライズド・ビューのすべての行が戻されます。

ノート:

この句がパーティション表またはパーティション索引の DATE 列を参照している場合、データベースは、次の条件でのみパーティション・プルーニングを実行します。

- TO_DATE ファンクションで 4 桁書式マスクを使用して年を完全に指定した表または索引パーティションを作成した場合
- TO_DATE ファンクションで 2 または 4 桁書式マスクを使用して問合せの where_clause に日付を指定した場合。

関連項目:

8. conditionの構文の詳細は、「条件」を参照してください。
9. 「パーティションからの選択: 例」を参照してください。

hierarchical_query_clause

hierarchical_query_clauseを使用すると、階層順序で行を選択できます。

階層問合せを含むSELECT文では、SELECT構文のリスト内のLEVEL疑似列を使用できます。LEVELは、ルート・ノードには1を、ルート・ノードの子であるノードには2を、孫であるノードには3を戻します(以下同様)。階層問合せによって戻されるレベルの数値は、使用可能なユーザー・メモリーによって制限されます。

Oracleは次のように階層問合せを処理します。

- 最初に、結合(指定されている場合)が、FROM句で指定されているか、またはWHERE句述語で指定されているかが評価されます。
- CONNECT BY条件が評価されます。
- 残りのWHERE句述語が評価されます。

この句を指定する場合、ORDER BYおよびGROUP BYを指定すると、CONNECT BY結果の階層順序が破棄されるため、これらの句のどちらも指定しないでください。同じ親の兄弟である行を順序付ける場合は、ORDER SIBLINGS BY句を使用します。

関連項目:

階層問合せの詳細は、「階層問合せ」を参照してください。また、「LEVEL疑似列の使用方法: 例」を参照してください。

START WITH句

階層問合せのルートとして使用される行を識別する場合の条件を指定します。conditionには、「条件」で説明されているいずれかの条件を指定できます。Oracle Databaseでは、この条件を満たすすべての行がルートとして使用されます。この句を省略した場合、表内のすべての行がルート行として使用されます。

CONNECT BY句

階層の親/子の行の関連を識別する条件を指定します。conditionには、「条件」で説明されているいずれかの条件を指定できます。ただし、親である行を参照するためのPRIOR演算子を使用する必要があります。

関連項目:

- LEVELの詳細は、「疑似列」を参照してください。
- 階層問合せの概要は、「階層問合せ」を参照してください。
- 「階層問合せの例」

group_by_clause

GROUP BY句を指定すると、選択した行を各行のexprの値に基づいてグループ化し、各グループのサマリー情報を1行戻すことができます。この句にCUBEまたはROLLUP拡張要素を指定した場合、標準グループ化の他に超集合グループ化が生成されます。

GROUP BY句の式には、SELECT構文のリストに指定されている列であるかどうかにかかわらず、FROM句の表、ビューおよびマテリアライズド・ビューの列を指定できます。

GROUP BY句は行をグループ化しますが、結果セットの順序は保証しません。グループの並べ替えを行うには、ORDER BY句を使用します。

関連項目:

- データを集計するSQLグループ化構文の詳細および例については、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- この句の例については、「GROUP_ID」、「GROUPING」および「GROUPING_ID」を参照してください。
- 「GROUP BY句の使用方法: 例」を参照してください。
- GROUP BY文字値を言語的に比較する方法の影響の詳細は、「言語照合の制限事項」を参照してください。
- GROUP BY句の式の照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

ROLLUP

simple_grouping_clauseのROLLUP操作を使用すると、選択した行をGROUP BYで指定した式n、n-1、n-2、... 0の最初の値に基づいてグループ化し、各グループのサマリー情報を1行戻すことができます。ROLLUP操作をSUMファンクションとともに使用すると、小計値を出力できます。ROLLUPをSUMとともに使用すると、最も詳細なレベルの小計から総計までが生成されます。COUNTなどの集計ファンクションは、他の種類の超集合の出力に使用できます。

たとえば、simple_grouping_clauseのROLLUP句に式を3つ指定した場合(n=3)、操作の結果はn+1=3+1=4グループになります。

最初のn式の値でグループ化した行を標準行、その他を超集合行といいます。

関連項目:

マテリアライズド・ビューでROLLUPを使用する場合の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

CUBE

CUBE操作をsimple_grouping_clauseの中で指定すると、選択された行は、指定された式のすべての可能な組合せ

の値に基づいてグループ化されます。グループごとに1つのサマリー情報行が戻されます。CUBE操作を使用すると、クロス集計値を出力できます。

たとえば、simple_grouping_clauseのCUBE句に式を3つ指定した場合(n=3)、操作の結果は$2^n = 2^3 = 8$グループになります。n式の値でグループ化した行を標準行、その他を超集合行といいます。

関連項目:

- マテリアライズド・ビューでCUBEを使用する場合の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- 「GROUP BY CUBE句の使用方法: 例」を参照してください。

GROUPING SETS

GROUPING SETSは、データを複数にグループ化するGROUP BY句をさらに拡張したものです。これによって、不要な集計が排除され、効率的に集計できるようになります。必要なグループを指定すると、データベースがCUBEまたはROLLUPによって生成された集計のすべてを実行する必要がなくなります。GROUPING SETS句で指定したすべてのグループ化が計算され、UNION ALL操作で個々のグループ化の結果が組み合されます。UNION ALLは、結果セットが重複行を含むことを許可します。

GROUP BY句では、様々な方法で式を組み合せることができます。

- 複合列を指定するには、カッコで列をグループ化します。データベースは、ROLLUP操作またはCUBE操作の計算でこれらを1つの単位として処理します。
- グルーピング・セットの連結を指定するには、複数のグルーピング・セット、ROLLUP操作およびCUBE操作をカンマで区切って指定すると、データベースによってこれらが結合されて1つのGROUP BY句になります。結果は、各グルーピング・セットからのグループ化のクロス積です。

関連項目:

GROUPING SETS句の使用方法: 例

HAVING句

HAVING句を使用すると、指定したconditionがTRUEであるグループの行のみを戻すように制限できます。この句を省略した場合、すべてのグループのサマリー行が戻されます。

where_clauseおよびhierarchical_query_clauseの後に、GROUP BYおよびHAVINGを指定します。GROUP BYとHAVINGの両方を指定する場合は、どちらの順序でも指定できます。

関連項目:

HAVING条件の使用方法: 例

GROUP BY句の制限事項

この句には、次の制限事項があります。

- LOB列、ネストした表またはVARRAYをexprの一部として指定できません。
- 式には、スカラー副問合せ式を除くすべての形式が可能です。

- group_by_clauseがオブジェクト型列を参照する場合、問合せはパラレル化されません。

model_clause

model_clauseを使用すると、選択した行を多次元配列として表示して、その配列内のセルにランダムにアクセスできます。model_clauseを使用すると、一連のセル割当て(ルールと呼ばれます)を指定しておき、このルールによって個々のセルやセル範囲に対する計算を実行できます。このルールの操作の対象は問合せの結果であり、データベース表が更新されることはありません。

問合せでmodel_clauseを使用する場合、SELECT句およびORDER BY句は、model_column_clausesで定義された列のみを参照する必要があります。

関連項目:

- exprの構文の詳細は、「SQL式」を、conditionの構文の詳細は、「条件」を参照してください。
- 詳細および例は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- 「MODEL句: 例」を参照してください。

main_model

main_model句を使用すると、選択した行を多次元配列内で表示する方法および配列内の各セルに適用するルールを定義できます。

model_column_clauses

model_column_clausesを使用すると、問合せの列を、パーティション列、ディメンション列およびメジャー列の3つのグループに定義して分類できます。exprには、列、定数、ホスト変数、単一行ファンクション、集計ファンクションまたはこれらを含む任意の式を指定できます。exprが列の場合、列の別名(c_alias)はオプションです。exprが列でない場合、列の別名は必須です。列の別名を指定する場合、別名を使用してmodel_rules_clause、SELECTリストおよび問合せORDER BY句の列を参照する必要があります。

PARTITION BY

PARTITION BY句を使用すると、選択した行を列の値に基づいてパーティションに分割するために使用する列を指定できます。

DIMENSION BY

DIMENSION BY句を使用すると、パーティション内で行を識別する列を指定できます。ディメンション列およびパーティション列の値は、行のメジャー列に対する配列の索引として使用されます。

MEASURES

MEASURES句を使用すると、計算が実行可能な列を識別できます。個々の行のメジャー列は、パーティション列およびディメンション列の値を指定することによる参照および更新が可能なセルと同様に扱われます。

cell_reference_options

cell_reference_options句を使用すると、ルールでNULLまたは値なしを処理する方法および列の一意性を制約する方法を指定できます。

IGNORE NAV

IGNORE NAVを指定すると、指定したデータ型のNULLまたは値なしに対して、次の値が戻されます。

- 数値データ型: 0(ゼロ)
- 01-JAN-2000 日時データ型
- 文字データ型: 空の文字列
- その他のデータ型: NULL

KEEP NAV

KEEP NAVを指定すると、NULLまたは値なしのセル値に対してNULLが戻されます。KEEP NAVはデフォルトです。

UNIQUE SINGLE REFERENCE

UNIQUE SINGLE REFERENCEを指定すると、問合せの結果セット全体ではなく、ルールの右側の単一セルの参照のみが一意性をチェックされます。

UNIQUE DIMENSION

UNIQUE DIMENSIONを指定すると、PARTITION BYおよびDIMENSION BYで指定した列が、問合せに対する一意キーであるかどうかが確認されます。UNIQUE DIMENSIONはデフォルトです。

model_rules_clause

model_rules_clauseを使用すると、更新するセルおよびこれらのセルを更新するルールを指定できます。オプションで、ルールを適用および処理する方法も指定できます。

各ルールは割当てを表し、左側と右側にわかれています。ルールの左側は、ルールの右側によって更新されるセルを識別します。ルールの右側は、ルールの左側で指定されたセルに割り当てられる値を評価します。

UPSERT ALL

UPSERT ALLを使用すると、ルールの左側に位置参照と記号参照の両方があるルールに対してUPSERT動作が可能になります。UPSERT ALLルールが評価されると、次のステップが実行され、アップサートするセル参照のリストが作成されます。

- セル参照のすべてのシンボリック述語を満たす既存のセルを検索します。
- 記号参照があるディメンションのみを使用して、これらのセルの異なるディメンション値の組合せを検索します。
- 位置参照によって指定されたディメンション値を持つ、これらの値の組合せのクロス積が実行されます。

UPSERT ALLのセマンティクスの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

UPSERT

UPSERTを指定すると、ルールの左側で参照されるセルが多次元配列内に存在している場合、セルにルールが適用され、多次元配列内に存在しないセルに対しては新しい行が挿入されます。UPSERT動作は、ルールの左側で位置参照が使用され、単一セルが参照されている場合にのみ適用されます。UPSERTはデフォルトです。位置参照および単一セル参照の詳細は、「cell_assignment」を参照してください。

UPDATEおよびUPSERTは、個々のルールに同様に指定できます。特定のルールにUPDATEまたはUPSERTのいずれかを指定した場合、その指定はRULES句に指定したその他のオプションより優先されます。

ノート:

UPSERT ALL、UPSERT または UPDATE ルールに適切な条件が含まれていない場合は、別のタイプのルールに暗黙的に変換される場合があります。

- UPSERT ルールに存在述語が含まれている場合、そのルールは UPDATE ルールとして処理されます。
- UPSERT ALL ルールには、その左側に少なくとも 1 つの存在述語と 1 つの修飾述語が必要です。存在述語がない場合は、UPSERT ルールとして処理されます。修飾述語がない場合は、UPDATE ルールとして処理されます。

UPDATE

UPDATEを指定するとルールの左側で参照されるセルが多次元配列内に存在している場合、そのセルにルールが適用されます。セルが存在しない場合、割当ては無視されます。

AUTOMATIC ORDER

AUTOMATIC ORDERを指定すると、依存順序に基づいてルールが評価されます。この場合、セルには値が1回のみ割り当てられます。

SEQUENTIAL ORDER

SEQUENTIAL ORDERを指定すると、表示されている順序でルールが評価されます。この場合、セルには値が複数回割り当てられます。SEQUENTIAL ORDERはデフォルトです。

ITERATE ... [UNTIL]

ITERATE ... [UNTIL]を使用すると、ルールを繰り返す回数を指定でき、さらに早期終了条件を指定することもできます。UNTILを囲むカッコの使用は任意です。

ITERATE ... [UNTIL]を指定した場合、ルールは表示されている順序で評価されます。model_rules_clauseでAUTOMATIC ORDERおよびITERATE ... [UNTIL]の両方が指定されている場合、エラーが戻されます。

cell_assignment

cell_assignment句は、ルールの左側に使用し、更新する1つ以上のセルを指定します。単一セルを参照するcell_assignmentは、単一セル参照といいます。複数のセルが参照される場合は、複数セル参照といいます。

model_clauseで定義したすべてのディメンション列は、cell_assignment句で修飾する必要があります。ディメンションは、記号参照または位置参照を使用して修飾できます。

記号参照は、dimension_column=constantなどのブール条件を使用して、単一のディメンション列を修飾します。位置参照では、DIMENSION BY句でディメンション列の位置が示されます。記号参照と位置参照の唯一の相違点は、NULLの処理です。

a[x=null,y=2000]のような単一セルの記号参照を使用すると、x=nullがFALSEと評価されるため、該当するセルは存在しません。ただし、a[null,2000]のような単一セルの位置参照を使用すると、null = nullがTRUEと評価されるため、xがNULL、yが2000のセルが該当します。単一セルの位置参照を使用すると、ディメンション列がNULLのセルを参照、更新および挿入できます。

ディメンション列の値を表す条件または式を指定するときは、記号参照と位置参照のどちらも使用できます。conditionに集計ファンクションやCVファンクションを含めることはできず、conditionが参照するのは単一のディメンション列でなければなりません。exprに副問合せを含めることはできません。モデル式の詳細は、「モデル式」を参照してください。

single_column_for_loop

single_column_for_loop句を使用すると、更新するセルの範囲を単一のディメンション列内で指定できます。

IN句を使用すると、ディメンション列の値を値のリストまたは副問合せとして指定できます。subqueryを使用するときは、次の制限事項があります。

- 相関問合せは使用できません。
- 10,000を超える行を戻すことはできません。
- WITH句で定義された問合せを使用できません。

FROM句を使用すると、ディメンション列の値の範囲を指定できます(範囲内の増分は不連続でもかまいません)。FROM句を使用できるのは、列のデータ型が、加算および減算をサポートするものである場合のみです。INCREMENTおよびDECREMENTの値は、正の値である必要があります。

オプションで、FROM句内でLIKE句を指定することができます。LIKE句のpatternは、単一のパターン一致文字%を含む文字列です。この文字は、実行時にFROM句の現在の増分値または減分値で置き換えられます。

FORループで使用されるディメンション以外のすべてのディメンションが単一セル参照に関係する場合は、式で新しい行を挿入できます。FORループによって生成されたディメンション値の組合せの数は、MODEL句の行制限(10,000)の計算に含まれます。

multi_column_for_loop

multi_column_for_loop句を使用すると、更新するセルの範囲を複数のディメンション列にまたがって指定できます。IN句を使用すると、ディメンション列の値を複数の値のリストまたは副問合せとして指定できます。subqueryを使用するときは、次の制限事項があります。

- 相関問合せは使用できません。
- 10,000を超える行を戻すことはできません。
- WITH句で定義された問合せを使用できません。

FORループで使用されるディメンション以外のすべてのディメンションが単一セル参照に関係する場合は、式で新しい行を挿入できます。FORループによって生成されたディメンション値の組合せの数は、MODEL句の行制限(10,000)の計算に含まれます。

関連項目:

MODEL句でFORループを使用する方法の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

order_by_clause

ORDER BY句を使用すると、ルールの左側のセルを評価する順序を指定できます。exprは、ディメンションまたはメジャー列に変換される必要があります。ORDER BY句を指定しない場合、DIMENSION BY句で指定した列の順序がデフォルトで使用されます。詳細は、「order_by_clause」を参照してください。

order_by_clauseの制限事項

モデル・ルールでのORDER BY句の使用には、次の制限事項があります。

- model_clauseのorder_by_clauseに、SIBLINGS、positionまたはc_aliasを指定することはできません。
- モデル・ルールの左側にこの句を指定し、右側にFORループも指定することはできません。

expr

ルールの右側で指定されているセルの値を表す式を指定します。exprに副問合せを含めることはできません。モデル式の詳細は、「モデル式」を参照してください。

return_rows_clause

return_rows_clauseを使用すると、選択されたすべての行を戻すか、モデル・ルールによって更新された行のみを戻すかどうかを指定できます。ALLはデフォルトです。

reference_model

reference_modelは、model_clause内から複数の配列にアクセスする必要がある場合に使用します。この句は、問合せの結果に基づいて、読取り専用の多次元配列を定義します。

reference_model句の副次句は、main_model句と同じセマンティクスを持ちます。model_column_clausesおよびcell_reference_optionsを参照してください。

reference_model句の制限事項

この句には、次の制限事項があります。

- PARTITION BY列を参照モデルに指定することはできません。
- 参照モデルの副問合せから外部副問合せの列を参照することはできません。

集合演算子: UNION、UNION ALL、INTERSECTおよびMINUS

集合演算子は、2つのSELECT文によって戻された行を1つの結果に結合します。それぞれのコンポーネント問合せで選択される列の数とデータ型は同じである必要がありますが、列の長さは異なってもかまいません。結果セット内の列の名前は、集合演算子の前にあるSELECT構文のリスト内の式の名前です。

集合演算子で3つ以上の問合せを結合する場合、隣接する問合せが左から右へ評価されます。副問合せを囲むカッコは任意指定です。この評価順序を変更する場合、カッコを使用します。

これらの演算子の詳細および使用方法の制限事項は、「UNION [ALL]、INTERSECTおよびMINUS演算子」を参照してください。

order_by_clause

ORDER BY句を使用すると、文によって戻された行を順序付けることができます。order_by_clauseを指定しない場合、同じ問合せで取り出される行の順序が異なることがあります。

SIBLINGS

SIBLINGSキーワードは、hierarchical_query_clause(CONNECT BY)を指定する場合のみに有効です。ORDER SIBLINGS BYは階層問合せ句で指定した任意の順序を保持し、兄弟関係にある階層にorder_by_clauseを適用します。

expr

exprを使用すると、exprの値を基準にして行を順序付けることができます。式は、SELECT構文のリストの列、あるいはFROM句の表、ビューまたはマテリアライズド・ビューの列に基づきます。

position

positionを使用すると、SELECT構文のリストの指定した位置にある式の値に基づいて行を順序付けることができます。positionには整数を指定する必要があります。

order_by_clauseには複数の式を指定できます。この場合、まず、最初の式の値に基づいて行がソートされます。次に、

最初の式と同じ値を持つ行が2番目の式の値に基づいてソートされる、というように処理が行われます。NULL値は昇順では最後に、降順では先頭にソートされます。問合せ結果の順序付けの詳細は、「問合せ結果のソート」を参照してください。

ASC | DESC

昇順か降順かを指定します。デフォルトはASCです。

NULLS FIRST | NULLS LAST

NULL値を含む戻された行が順序の最初にくるか、最後にくるかを指定します。

NULLS LASTは昇順のデフォルトで、NULLS FIRSTは降順のデフォルトです。

ORDER BY句の制限事項

ORDER BY句には、次の制限事項が適用されます。

- この文中でDISTINCT演算子を指定した場合、SELECT構文のリストに指定された列でないかぎり、この句は列を参照することはできません。
- order_by_clauseには最大255個の式を指定できます。
- LOB列、LONG列、LONG RAW列、ネストした表またはVARRAYを使用して順位付けすることはできません。
- 同じ文中でgroup_by_clauseを指定する場合、このorder_by_clauseは次の式に制限されます。
    - 定数
    - 集計ファンクション
    - 分析ファンクション
    - USERファンクション、UIDファンクションおよびSYSDATEファンクション
    - group_by_clauseに指定されているものと同じ式
    - グループ内のすべての行が同じ値に評価されるこれらの式を導出する式

関連項目:

- 「ORDER BY句の使用方法: 例」を参照してください。
- ORDER BY文字値を言語的に比較する方法の影響の詳細は、「言語照合の制限事項」を参照してください。
- ORDER BY句の式の照合決定ルールは、『Oracle Databaseグローバリゼーション・サポート・ガイド』の付録Cを参照してください。

row_limiting_clause

row_limiting_clauseを使用すると、問合せから返される行数を制限できます。オフセットおよび戻される行の数または割合を指定できます。この句は、上位N番のレポートを実装するために使用できます。結果の一貫性を保つには、order_by_clauseを指定して、ソート順序を決定的にします。

OFFSET

この句を使用して、行制限が開始する前にスキップする行数を指定します。offsetは、数値または数値に評価される式にする必要があります。負の数値を指定すると、offsetは、0(ゼロ)とみなされます。NULLを指定した場合や、問合せから返される行数以上の数値を指定すると、返される行数は0(ゼロ)行になります。offsetに小数部が含まれているときには、小数部

分が切り捨てられます。この句を省略すると、offsetは0(ゼロ)になり、行制限の開始が最初の行になります。

ROW | ROWS

これらのキーワードは、区別なしに使用できますが、セマンティクスを明確にするために用意されています。

FETCH

この句を使用すると、返される行数または行の割合を指定できます。この句を指定しないと、offset + 1行目から始まるすべての行が返されます。

FIRST | NEXT

これらのキーワードは、区別なしに使用できますが、セマンティクスを明確にするために用意されています。

rowcount | percent PERCENT

rowcountを使用すると、戻す行数を指定できます。rowcountは、数値または数値に評価される式である必要があります。負の数値を指定すると、rowcountは、0(ゼロ)とみなされます。rowcountが、offset + 1行目から始まる使用可能な行数よりも大きいときには、すべての使用可能な行が返されます。rowcountに小数部が含まれているときには、小数部分が切り捨てられます。rowcountがNULLの場合、返される行数は0(ゼロ)行になります。

percent PERCENTを使用すると、選択された行の合計行数から、返される行数の割合を指定できます。percentは、数値または数値に評価される式にする必要があります。負の数値を指定すると、percentは、0(ゼロ)とみなされます。percentがNULLの場合、返される行数は0(ゼロ)行になります。

rowcountまたはpercent PERCENTを指定しないと、返される行数は1行になります。

ROW | ROWS

これらのキーワードは、区別なしに使用できますが、セマンティクスを明確にするために用意されています。

ONLY | WITH TIES

ONLYを指定すると、指定したとおりの行数または行数の割合が戻されます。

WITH TIESを指定すると、最後にフェッチされた行と同じソート・キーを持つ追加の行が返されます。WITH TIESを指定するときには、order_by_clauseを指定する必要があります。order_by_clauseを指定しないと、追加の行は返されなくなります。

row_limiting_clauseの制限事項

この句には、次の制限事項があります。

- この句は、for_update_clauseと同時には指定できません。
- この句を指定すると、順序疑似列のCURRVALまたはNEXTVALを、選択リストに含めることができなくなります。
- マテリアライズド・ビューは、それを定義する問合せにrow_limiting_clauseが含まれているときには、増分リフレッシュの対象にはなりません。
- 選択リストに同一の名前の列が含まれる場合にrow_limiting_clauseを指定すると、ORA-00918エラーが発生します。このエラーは、同一の名前の列が同じ表または別の表内のいずれにあっても発生します。この問題は、一意の列の別名を同一の名前の列に指定することで解決できます。

関連項目:

「行制限：例」

for_update_clause

FOR UPDATE句を使用すると、選択した行がロックされ、トランザクションが終了するまでは他のユーザーがその行をロックまたは更新することはできなくなります。この句を指定できるのは、最上位のSELECT文の中のみであり、副問合せでは指定できません。

> ノート:
>
> LOB 値を更新する場合、その LOB を含む行をロックしておく必要があります。行をロックする方法の 1 つに、埋込み SELECT ... FOR UPDATE 文があります。この場合、プログラム言語の 1 つまたは DBMS_LOB パッケージを使用します。LOB の書込み前に行う行のロックの詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』を参照してください。

親表の行がロックされても、ネストした表の行はロックされません。ネストした表の行をロックする場合、ネストした表を明示的にロックする必要があります。

FOR UPDATE句の制限事項

この句には、次の制限事項があります。

- この句をDISTINCT演算子またはCURSOR式、集合演算子、group_by_clause、または集計ファンクションの構造体とともに指定することはできません。
- この句がロックした表は、同じ文で参照されたLONG列および順序と同じデータベース内にある必要があります。

関連項目:

FOR UPDATE句の使用方法：例

ビューでのFOR UPDATE句の使用方法

一般的に、この句はビューではサポートされません。ただし、ビューでSELECT ... FOR UPDATE問合せを実行してもエラーが発生しない場合もあります。これは、問合せオプティマイザによってビューが問合せブロックに内部的にマージされ、内部で変換された問合せに対してSELECT ... FOR UPDATEが行われる場合です。この項では、ビューに対してFOR UPDATE句を使用できる場合とできない場合の例を示します。

- マージされたビューでのFOR UPDATE句の使用

  次の条件が両方とも満たされている場合、マージされたビューに対してFOR UPDATE句を使用するとエラーが発生する可能性があります。

  - ビューの基礎となる列が式である
  - FOR UPDATE句が列リストに適用される

  次の文は、ビューの基礎となる列が式ではないため、正常に実行されます。

```
SELECT employee_id FROM (SELECT * FROM employees)
   FOR UPDATE OF employee_id;
```

  次の文は、ビューの基礎となる列は式ですが、FOR UPDATE句が列リストに適用されないため、正常に実行されます。

```
SELECT employee_id FROM (SELECT employee_id+1 AS employee_id FROM employees)
   FOR UPDATE;
```

次の文は、ビューの基礎となる列が式であり、FOR UPDATE句が列リストに適用されるため、失敗します。

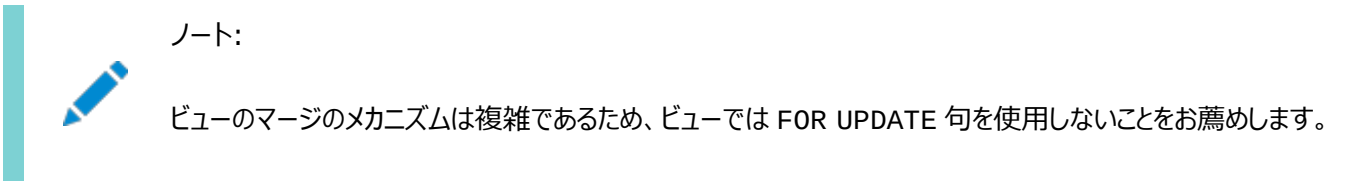
```
SELECT employee_id FROM (SELECT employee_id+1 AS employee_id FROM employees)
   FOR UPDATE OF employee_id;
                 *
Error at line 2:
ORA-01733: virtual column not allowed here
```

- マージされていないビューでのFOR UPDATE句の使用

  ビューではFOR UPDATE句がサポートされていないため、ビューのマージを回避する句(NO_MERGEヒントなど)、ビューのマージを禁止するパラメータ、またはビューのマージを回避する問合せ構造を使用すると、ORA-02014エラーが発生します。

  次の例では、GROUP BY文によってビューのマージが回避されるため、エラーが発生します。

```
SELECT avgsal
   FROM (SELECT AVG(salary) AS avgsal FROM employees GROUP BY job_id)
   FOR UPDATE;
FROM (SELECT AVG(salary) AS avgsal FROM employees GROUP BY job_id)
     *
ERROR at line 2:
ORA-02014: cannot select FOR UPDATE from view with DISTINCT, GROUP BY, etc.
```



ノート:

ビューのマージのメカニズムは複雑であるため、ビューでは FOR UPDATE 句を使用しないことをお薦めします。

OF ... column

OF ... column句を使用すると、結合内の特定の表またはビューで選択された行のみをロックできます。OF句の列は、どの表またはビューの行をロックするかを識別する場合にのみ使用します。指定する列は重要ではありません。ただし、列の別名ではなく、実際の列名を指定する必要があります。この句を省略した場合、問合せ内のすべての表の選択された行がロックされます。

NOWAIT | WAIT

NOWAITおよびWAIT句を使用すると、他のユーザーによってロックされている行をSELECT文がロックしようとする場合に処理する方法をデータベースに指示できます。

- NOWAITを指定すると、ロックされている場合に制御がすぐに戻ります。
- WAITを指定すると、行が使用可能になるまでinteger秒待機した後で制御が戻されます。

WAITおよびNOWAITのどちらも指定しない場合、行が使用可能になるまで待機した後でSELECT文の結果が戻されます。

SKIP LOCKED

SKIP LOCKEDは、競合するトランザクションを処理するもう1つの方法であり、対象の行のうち一部をロックするというものです。SKIP LOCKEDを指定すると、WHERE句で指定した行のロックが試行され、他のトランザクションによってすでにロックされている行はスキップされます。この機能は、マルチコンシューマ・キュー環境で使用する目的で設計されています。キュー・コンシューマは、他のコンシューマによってロックされた行はスキップして未ロックの行を取得できるので、他のコンシューマの操作が終了するまで待つ必要はなくなります。詳細は、Oracle Databaseアドバンスド・キューイング・ユーザーズ・ガイドを参照してください。

WAIT句およびSKIP LOCKED句のノート

WAITまたはSKIP LOCKEDを指定したときに排他モードで表がロックされていると、表のロックが解除されるまではSELECT文の結果が戻りません。WAITでは、指定されている待機時間にかかわらず、SELECT FOR UPDATE句がブロックされます。

row_pattern_clause

MATCH_RECOGNIZE句を使用すると、パターン一致を実行できます。この句は、table(行パターン入力表といいます)に含まれる一連の行のパターンを認識するために使用します。MATCH_RECOGNIZE句を使用する問合せの結果を行パターン出力表といいます。

MATCH_RECOGNIZEにより、次のタスクが実行可能になります。

- PARTITION BY句およびORDER BY句が含まれるデータを、論理的にパーティション化し、順序付けます。
- MEASURES句で、SQL問合せの他のパートで使用可能な式であるメジャーを定義します。
- PATTERN句を使用して、シークする行のパターンを定義します。これらのパターンでは、正規表現構文が使用されます。これは、強力かつ表現力の豊かな機能であり、ユーザーが定義するパターン変数に適用されます。
- 行をDEFINE句にある行パターン変数にマップするために必要な論理条件を指定します。

関連項目:

- パターン一致の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- 「行パターン一致: 例」を参照してください。

row_pattern_partition_by

PARTITION BYを指定すると、行パターン入力表に含まれる行を、行パターン・パーティションという論理グループに分割できます。columnを使用して、1つ以上のパーティション化列を指定します。各パーティションは、パーティション化列の値と同じ値が設定された、行パターン入力表に含まれる行のセットで構成されています。

この句を指定した場合、一致はパーティション内から検出され、他のパーティションからは検出されません。この句を指定しなかった場合、行入力表のすべての行から1つの行パターン・パーティションが構成されます。

row_pattern_order_by

ORDER BYを指定すると、各行パターン・パーティション内の行を順序付けできます。columnを使用して、順序付けする1つ以上の列を指定します。複数の列を指定した場合、Oracle Databaseはまず、最初のcolumnにおける行の値に基づいて、それらの行をソートします。次に、最初の列と同じ値を持つ行が2番目のcolumnの値に基づいてソートされる、というように処理が行われます。Oracle Databaseは、昇順でソートされた他のすべての値の後にNULLを配置します。

この句を指定しなかった場合、row_pattern_clauseの結果は非決定的になるため、問合せを実行するたびに異なる結果が返される場合があります。

row_pattern_measures

MEASURES句を使用すると、1つ以上の行パターンのメジャー列を定義できます。行パターン出力表に含まれるこれらの列には、データ分析に役立つ値が含まれています。

row_pattern_measure_column句を使用して行パターンのメジャー列を定義する場合、そのパターン・メジャー式を指定します。列内の値は、一致が見つかるたびにパターン・メジャー式を評価することによって計算されます。

row_pattern_measure_column

この句を使用すると、行パターンのメジャー列を定義できます。

- exprにはパターン・メジャー式を指定します。「式」で説明されているように、パターン・メジャー式には次の要素のみを含めることができます。
  - 定数: テキスト・リテラルおよび数値リテラル
  - 行パターン入力表に含まれる任意の列の参照
  - CLASSIFIER関数。行のマップ先であるプライマリ行パターン変数の名前を返します。詳細は、「row_pattern_classifier_func」を参照してください。
  - MATCH_NUMBER関数。行パターン・パーティション内の行パターン一致の連番を返します。詳細は、「row_pattern_match_num_func」を参照してください。
  - 行パターンのナビゲーション関数: PREV、NEXT、FIRSTおよびLAST。詳細は、「row_pattern_navigation_func」を参照してください。
  - 行パターン集計関数: AVG、COUNT、MAX、MINまたはSUM。詳細は、「row_pattern_aggregate_func」を参照してください。
- c_aliasには、パターン・メジャー式の別名を指定します。Oracle Databaseは、行パターン出力表の列ヘッダーでこの別名を使用します。ASキーワードはオプションです。別名は、SELECT ... ORDER BY句など、問合せのその他の部分で使用できます。

row_pattern_rows_per_match

この句を使用すると、行パターン出力表に、各一致のサマリー・データまたは詳細データを含めるかどうかを指定できます。

- ONE ROW PER MATCHを指定すると、一致ごとに1つのサマリー行が生成されます。これはデフォルトです。
- ALL ROWS PER MATCHを指定すると、一致に複数の行が含まれる場合、その一致に含まれる行ごとに1つの出力行が生成されます。

row_pattern_skip_to

この句を使用すると、空以外の一致が見つかった後の行パターン一致を再開する場所を指定できます。

- AFTER MATCH SKIP TO NEXT ROWを指定すると、現在の一致の先頭行の次の行からパターン一致が再開されます。
- AFTER MATCH SKIP PAST LAST ROWを指定すると、現在の一致の最終行の次の行からパターン一致が再開されます。これはデフォルトです。
- AFTER MATCH SKIP TO FIRST variable_nameを指定すると、パターン変数variable_nameにマップされた先頭行からパターン一致が再開されます。variable_nameはDEFINE句で定義する必要があります。
- AFTER MATCH SKIP TO LAST variable_nameを指定すると、パターン変数variable_nameにマップされた最終行からパターン一致が再開されます。variable_nameはDEFINE句で定義する必要があります。
- AFTER MATCH SKIP TO variable_nameの動作は、AFTER MATCH SKIP TO LAST variable_nameと同じです。

関連項目:

AFTER MATCH SKIP句の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

PATTERN

PATTERN句を使用すると、一致する必要のあるパターン変数、そのパターン変数の一致順序、および各パターン変数に一致する必要のある行数を定義できます。

行パターン一致は、1つの行パターン・パーティション内で連続する行のセットで構成されます。一致の各行はそれぞれ1つのパターン変数にマップされます。行とパターン変数とのマッピングは、row_pattern句に指定された正規表現に準拠している必要があり、DEFINE句のすべての条件を満たす必要があります。

ノート:

正規表現の概念と詳細についての説明は、このマニュアルでは対象外です。正規表現について十分に理解していない場合は、その他の資料を参照して理解しておくことをお薦めします。

PATTERNS句の正規表現で指定した要素の優先順位は、降順で次のとおりです。

- 行パターン要素(row_pattern_primary句で指定されます)
- 行パターン数量子(row_pattern_quantifier句で指定されます)
- 連結(row_pattern_term句で指定されます)
- 代替(row_pattern句で指定されます)

関連項目:

PATTERN句の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

row_pattern

この句を使用すると、行パターンを指定できます。行パターンは、次のいずれかの形式の正規表現です。

- 単一の行パターン用語

  たとえば: PATTERN(A)

- 行パターン、縦線および行パターン用語

  たとえば: PATTERN(A|B)

- 再帰的に構成された行パターン、縦線および行パターン用語

  たとえば: PATTERN(A|B|C)

この句内の縦線は、代替を表します。代替では、複数の使用可能な正規表現のリスト内の1つの正規表現に一致させます。代替項目は、指定された順序で優先されます。たとえば、PATTERN(A|B|C)を指定した場合、Oracle Databaseはまず Aとの一致を試みます。Aが一致しない場合は、Bとの一致を試みます。Bが一致しない場合は、Cとの一致を試みます。

row_pattern_term

この句を使用すると、行パターン用語を指定できます。行パターン用語の形式は、次のいずれかです。

- 単一の行パターン・ファクタ

  たとえば: PATTERN(A)

- 行パターン用語の後に行パターン・ファクタが続く形式。

たとえば: PATTERN(A B)

- 再帰的に構成された行パターン用語の後に行パターン・ファクタが続く形式

  たとえば: PATTERN(A B C)

2番目と3番目の例で使用されている構文は、連結を表します。連結は、一致の対象であるパターン内の2つ以上の項目と、それらの一致順序をリストするために使用します。たとえば、PATTERN(A B C)を指定した場合、Oracle Databaseはまず Aとの一致を試みた後、一致した行を使用してBとの一致を試み、さらに一致した行を使用してCとの一致を試みます。A、BおよびCに一致した行のみが、行パターン一致に含まれます。

row_pattern_factor

この句を使用すると、行パターン・ファクタを指定できます。行パターン・ファクタは、row_pattern_primary句を使用して指定された行パターン要素と、row_pattern_quantifier句を使用して指定されたオプションの行パターン数量子で構成されます。

row_pattern_primary

この句を使用すると、行パターン要素を指定できます。表19-1に、有効な行パターン要素とその説明を示します。

表19-1 行パターン要素

行パターン要素	説明
variable_name	row_pattern_definition 句で定義されたプライマリ・パターン変数名を指定します。row_pattern_subset_item 句で定義された共用体パターン変数は指定できません。
$	$は、パーティションの最後の行の後の位置と一致します。この要素はアンカーです。アンカーは、行ではなく位置に関して機能します。
^	^は、パーティションの最初の行の前の位置と一致します。この要素はアンカーです。アンカーは、行ではなく位置に関して機能します。
([row_pattern])	row_pattern は、一致の対象となる行パターンを指定するために使用します。空のパターン( )は、空の行セットと一致します。
{- row_pattern -}	除外構文。row_pattern は、ALL ROWS PER MATCH の出力から除外されるパターンの部分を指定するために使用します。
row_pattern_permute	row_pattern_permute は、行パターン要素の順列であるパターンを指定するために使用します。この句のセマンティクスの詳細は、「row_pattern_permute」を参照してください。

row_pattern_permute

PERMUTE句を使用すると、指定された行パターン要素の順列であるパターンを表記できます。たとえば、PATTERN

(PERMUTE (A, B, C))は、次のように、3つの行パターン要素A、BおよびCのすべての順列の代替と同じです。

```
PATTERN (A B C | A C B | B A C | B C A | C A B | C B A)
```

行パターン要素は辞書編集上で展開され、並べ替える各要素は他の要素からカンマで区切る必要があります。

関連項目:

順列の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

row_pattern_quantifier

この句を使用すると、行パターン数量子を指定できます。行パターン数量子は、一致で受け入れられる反復数を定義する後置演算子です。

行パターン数量子を強欲な数量子といい、適用されている正規表現で最大限のインスタンスの一致を試みます。例外は、接尾辞として疑問符(?)を持つ行パターン数量子であり、これらを最短一致数量子といいます。これらは、適用されている正規表現で最小限のインスタンスの一致を試みます。

表19-2は、有効な行パターン数量子と、それらが一致で受け入れる反復数を示しています。この表のnおよびmは、符号なし整数を表します。

表19-2 行パターン数量子

行パターン数量子	一致で受け入れられる反復数
*	0 以上の反復(強欲)
*?	0 以上の反復(最短一致)
+	1 以上の反復(強欲)
+?	1 以上の反復(最短一致)
?	0 または 1 個の反復(強欲)
??	0 または 1 個の反復(最短一致)
{n,}	n 個以上の反復(n >= 0) (強欲)
{n,}?	n 個以上の反復(n >= 0) (最短一致)
{n,m}	n から m 個(これを含む)の間の反復(0 <= n <= m, 0 < m) (強欲)
{n,m}?	n から m 個(これを含む)の間の反復(0 <= n <= m, 0 < m) (最短一致)

行パターン数量子	一致で受け入れられる反復数
{,m}	0 から m 個(これを含む)の間の反復(m > 0) (強欲)
{,m}?	0 から m 個(これを含む)の間の反復(m > 0) (最短一致)
{n}?	n 個の反復(n > 0)

関連項目:

行パターン数量子の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

row_pattern_subset_clause

SUBSET句を使用すると、1つ以上の共用体行パターン変数を指定できます。row_pattern_subset_item句を使用して、各共用体行パターン変数を宣言します。

次の句で、共用体行パターン変数を指定できます。

- MEASURES句: 行パターンのメジャー列の式内。つまり、row_pattern_measure_column句の式expr内です。
- DEFINE句: プライマリ・パターン変数を定義する条件内。つまり、row_pattern_definition句のcondition内です。

row_pattern_subset_item

この句を使用すると、それ自身の変数名を使って参照可能な複数のパターン変数のグループ化を作成できます。このグループ化を参照する変数名を、共用体行パターン変数といいます。

- 等号の左側にあるvariable_nameには、共用体行パターン変数の名前を指定します。
- 等号の右側には、カッコで括られた個別のプライマリ行パターン変数のカンマ区切りリストを指定します。このリストに共用体行パターン変数を含めることはできません。

関連項目:

共用体行パターン変数の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

DEFINE

DEFINE句を使用すると、1つ以上の行パターン定義を指定できます。行パターン定義は、行を特定のパターン変数にマップするために満たしている必要のある条件を指定します。

DEFINE句は実行中のセマンティクスのみをサポートします。

関連項目:

- DEFINE句の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- 実行中のセマンティクスおよび最終セマンティクスの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

row_pattern_definition_list

この句を使用すると、1つ以上の行パターン定義を指定できます。

row_pattern_definition

この句を使用すると、行パターン定義を指定できます。行パターン定義には、行を特定のパターン変数にマップするために満たしている必要のある条件が含まれます。

- variable_nameには、パターン変数の名前を指定します。
- conditionには、「Conditions」の説明に従って条件を指定しますが、次の追加事項があります。conditionには、row_pattern_navigation_func::=およびrow_pattern_aggregate_func::=によって記述された任意のファンクションを含めることができます。

row_pattern_rec_func

この句は、行パターン認識関数を指定できる、次の句で構成されます。

- row_pattern_classifier_func: この句を使用すると、CLASSIFIER関数を指定できます。この関数は、行のマップ先であるパターン変数の名前を値として持つ文字列を返します。
- row_pattern_match_num_func: この句を使用すると、MATCH_NUMBER関数を指定できます。この関数は、行パターン・パーティション内の一致の連番を表すスケール0(ゼロ)の数値を返します。
- row_pattern_navigation_func: この句を使用すると、行パターンのナビゲーション操作を実行する関数を指定できます。
- row_pattern_aggregate_func: この句を使用すると、行パターンのメジャー列の式、またはプライマリ・パターン変数を定義する条件で、集計関数を指定できます。

次の句で、行パターン認識関数を指定できます。

- MEASURES句: 行パターンのメジャー列の式内。つまり、row_pattern_measure_column句の式expr内です。
- DEFINE句: プライマリ・パターン変数を定義する条件内。つまり、row_pattern_definition句のcondition内です。

行パターン認識関数の動作は、この関数がMEASURES句とDEFINE句のどちらに指定されたかによって異なる場合があります。これらの詳細は、各句のセマンティクス内で説明されています。

row_pattern_classifier_func

CLASSIFIER関数は、行のマップ先である変数の名前を値として持つ文字列を返します。

- MEASURES句の場合:
    - ONE ROW PER MATCHが指定された場合、問合せはMEASURES句の処理時に一致の最後の行を使用するため、CLASSIFIER関数は一致の最後の行のマップ先であるパターン変数の名前を返します。
    - ALL ROWS PER MATCHが指定された場合、CLASSIFIER関数は、見つかった一致の行ごとに、行のマップ先であるパターン変数の名前を返します。

一致が空である場合、つまり一致に行が含まれない場合、CLASSIFER関数はNULLを返します。

- DEFINE句では、CLASSIFIER関数は現在行のマップ先であるプライマリ・パターン変数の名前を返します。

row_pattern_match_num_func

MATCH_NUMBER関数は、行パターン・パーティション内の一致の連番を表すスケール0(ゼロ)の数値を返します。

1つの行パターン・パーティション内の一致には、見つかった順に1から順次番号が付けられます。複数の行が一致した場合は、それらすべてに同じ一致番号が割り当てられます。行パターン・パーティション間では順序付けが継承されないため、一致の番号付は行パターン・パーティションごとに1から繰り返し開始されます。

- MEASURES句内の場合: MATCH_NUMBERを使用して、行パターン内の一致の連番を取得できます。
- DEFINE句内の場合: MATCH_NUMBERを使用して、一致番号に依存する条件を定義できます。

row_pattern_navigation_func

この句を使用すると、次の行パターンのナビゲーション操作を実行できます。

- row_pattern_nav_logical句のFIRSTおよびLAST関数を使用して、パターン変数にマップされる行のグループ内をナビゲートします。
- row_pattern_nav_physical句のPREVおよびNEXT関数を使用して、行パターン・パーティションに含まれるすべての行間をナビゲートします。
- row_pattern_nav_compound句を使用して、FIRSTまたはLAST関数をPREVまたはNEXT関数内にネストします。

row_pattern_nav_logical

この句を使用すると、FIRSTおよびLAST関数でオプションの論理オフセットを使用することによって、パターン変数にマップされた行のグループ内をナビゲートできます。

- FIRST関数は、式exprに指定されているパターン変数にマップされた行のグループの先頭行で評価されたときに、exprの値を返します。パターン変数に行がマップされていない場合、FIRST関数はNULLを返します。
- LAST関数は、式exprに指定されているパターン変数にマップされた行のグループの最終行で評価されたときに、exprの値を返します。パターン変数に行がマップされていない場合、LAST関数はNULLを返します。
- exprは、評価の対象となる式を指定するために使用します。1つ以上の行パターンの列参照が含まれている必要があります。複数の行パターンの列参照が含まれている場合、それらはすべて、同じパターン変数を参照している必要があります。
- オプションのoffsetは、パターン変数にマップされた行のセット内で論理オフセットを指定するために使用します。FIRST関数とともに指定した場合、先頭行から昇順で数えた行数がオフセットになります。LAST関数とともに指定した場合、最終行から降順で数えた行数がオフセットになります。デフォルトのオフセットは0です。

  offsetには、負以外の整数を指定します。列や副問合せではなく、ランタイム定数(リテラル、バインド変数またはそれらを含む式)である必要があります。

  パターン変数にマップされた行の数から1を引いた値以上の値をoffsetに指定した場合、この関数はNULLを返します。

FIRSTおよびLAST関数には、次のように実行中のセマンティクスまたは最終セマンティクスを指定できます。

- MEASURES句は、実行中のセマンティクスと最終セマンティクスをサポートします。実行中のセマンティクスは、

RUNNINGによって指定できます。最終セマンティクスは、FINALによって指定できます。デフォルトはRUNNINGです。

- DEFINE句は、実行中のセマンティクスのみをサポートします。そのため、RUNNINGを指定したか省略したかにかかわらず、実行中のセマンティクスが使用されます。FINALは指定できません。

関連項目:

- FIRSTおよびLAST関数の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- 実行中のセマンティクスおよび最終セマンティクスの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

row_pattern_nav_physical

この句を使用すると、PREVおよびNEXT関数でオプションの物理オフセットを使用することによって、行パターン・パーティション内のすべての行間をナビゲートできます。

- PREV関数は、パーティション内の前の行で評価されたときに、式exprの値を返します。パーティション内に前の行がない場合、PREV関数はNULLを返します。
- NEXT関数は、パーティション内の次の行で評価されたときに、式exprの値を返します。パーティション内に次の行がない場合、NEXT関数はNULLを返します。
- exprは、評価の対象となる式を指定するために使用します。1つ以上の行パターンの列参照が含まれている必要があります。複数の行パターンの列参照が含まれている場合、それらはすべて、同じパターン変数を参照している必要があります。
- オプションのoffsetは、パーティション内の物理オフセットを指定するために使用します。PREV関数とともに指定した場合、現在の行の前にある行数を表します。NEXT関数とともに指定した場合、現在の行の後にある行数を表します。デフォルトは1です。オフセットを0に指定した場合、現在の行が評価されます。

  offsetには、負以外の整数を指定します。列や副問合せではなく、ランタイム定数(リテラル、バインド変数またはそれらを含む式)である必要があります。

PREVおよびNEXT関数は、常に実行中のセマンティクスを使用します。したがって、この句にはRUNNINGまたはFINALキーワードを指定できません。

関連項目:

- PREVおよびNEXT関数の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- 実行中のセマンティクスおよび最終セマンティクスの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

row_pattern_nav_compound

この句を使用すると、row_pattern_nav_logical句をrow_pattern_nav_physical句内にネストできます。つまり、FIRSTまたはLAST関数をPREVまたはNEXT関数内にネストできます。まずrow_pattern_nav_logicalが評価された後、その結果がrow_pattern_nav_physical句に渡されます。

これらの句のセマンティクスの詳細は、「row_pattern_nav_logical」および「row_pattern_nav_physical」を参照してください。

関連項目:

FIRSTおよびLAST関数のPREVおよびNEXT関数内へのネストの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

row_pattern_aggregate_func

この句を使用すると、行パターンのメジャー列の式、またはプライマリ・パターン変数を定義する条件で、集計関数を使用できます。

aggregate_functionには、関数AVG、COUNT、MAX、MIN、SUMのいずれかを指定します。DISTINCTキーワードはサポートされていません。

集計関数には、次のように実行中のセマンティクスまたは最終セマンティクスを指定できます。

- MEASURES句は、実行中のセマンティクスと最終セマンティクスをサポートします。実行中のセマンティクスは、RUNNINGによって指定できます。最終セマンティクスは、FINALによって指定できます。デフォルトはRUNNINGです。
- DEFINE句は、実行中のセマンティクスのみをサポートします。そのため、RUNNINGを指定したか省略したかにかかわらず、実行中のセマンティクスが使用されます。FINALは指定できません。

関連項目:

- 集計関数の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。
- 実行中のセマンティクスおよび最終セマンティクスの詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

例

SQLマクロ - 表値のマクロ: 例

マクロ・ファンクションbudgetは、特定のジョブに対する各部門の予算額を計算します。指定された役職を持つ各部門の従業員数を返します。

```
create or replace function budget(job varchar2) return varchar2 SQL_MACRO is
begin
  return q'{
     select deptno, sum(sal) budget
     from emp
     where job = budget.job
     group by deptno
  }';
end;
/

SELECT * FROM budget ('MANAGER');
   DEPTNO     BUDGET
-----------  --------
     20        2975
     30        2850
     10        2450
```

「WITH句でのPL/SQLファンクションの使用例:」

次の例では、PL/SQLファンクションget_domainをWITH句で宣言および定義します。get_domainファンクションは、URL

文字列に含まれるドメイン名を返します。このファンクションでは、URL文字列はドメイン名の直前に"www"という接頭辞があり、ドメイン名は左右にあるドットで区切られていると仮定しています。SELECT文ではget_domainを使用して、oeスキーマのorders表から固有カタログのドメイン名を検出します。

```
WITH
 FUNCTION get_domain(url VARCHAR2) RETURN VARCHAR2 IS
   pos BINARY_INTEGER;
   len BINARY_INTEGER;
 BEGIN
   pos := INSTR(url, 'www.');
   len := INSTR(SUBSTR(url, pos + 4), '.') - 1;
   RETURN SUBSTR(url, pos + 4, len);
 END;
SELECT DISTINCT get_domain(catalog_url)
  FROM product_information;
/
```

副問合せのファクタリング: 例

次の文は、結合を含む初期問合せブロックに対する問合せの名前dept_costsおよびavg_costを作成し、主問合せの本体でその問合せの名前を使用します。

```
WITH
   dept_costs AS (
      SELECT department_name, SUM(salary) dept_total
         FROM employees e, departments d
         WHERE e.department_id = d.department_id
      GROUP BY department_name),
   avg_cost AS (
      SELECT SUM(dept_total)/COUNT(*) avg
      FROM dept_costs)
SELECT * FROM dept_costs
   WHERE dept_total >
      (SELECT avg FROM avg_cost)
      ORDER BY department_name;
DEPARTMENT_NAME                DEPT_TOTAL
------------------------------ ----------
Sales                              304500
Shipping                           156400
```

再帰的副問合せのファクタリング: 例

次の文は、従業員101の直接的または間接的な部下である従業員とそのレポート・レベルを表示します。

```
WITH
  reports_to_101 (eid, emp_last, mgr_id, reportLevel) AS
  (
     SELECT employee_id, last_name, manager_id, 0 reportLevel
     FROM employees
     WHERE employee_id = 101
   UNION ALL
     SELECT e.employee_id, e.last_name, e.manager_id, reportLevel+1
     FROM reports_to_101 r, employees e
     WHERE r.eid = e.manager_id
  )
SELECT eid, emp_last, mgr_id, reportLevel
FROM reports_to_101
ORDER BY reportLevel, eid;
       EID EMP_LAST                     MGR_ID REPORTLEVEL
---------- ------------------------ ---------- -----------
       101 Kochhar                         100           0
       108 Greenberg                       101           1
       200 Whalen                          101           1
       203 Mavris                          101           1
```

```
       204 Baer                            101           1
       205 Higgins                         101           1
       109 Faviet                          108           2
       110 Chen                            108           2
       111 Sciarra                         108           2
       112 Urman                           108           2
       113 Popp                            108           2
       206 Gietz                           205           2
```

次の文は、従業員101に直接的または間接的に報告する従業員、レポート・レベル、および管理チェーンを表示します。

```
WITH
  reports_to_101 (eid, emp_last, mgr_id, reportLevel, mgr_list) AS
  (
     SELECT employee_id, last_name, manager_id, 0 reportLevel,
            CAST(manager_id AS VARCHAR2(2000))
     FROM employees
     WHERE employee_id = 101
  UNION ALL
     SELECT e.employee_id, e.last_name, e.manager_id, reportLevel+1,
            CAST(mgr_list || ',' || manager_id AS VARCHAR2(2000))
     FROM reports_to_101 r, employees e
     WHERE r.eid = e.manager_id
  )
SELECT eid, emp_last, mgr_id, reportLevel, mgr_list
FROM reports_to_101
ORDER BY reportLevel, eid;
        EID EMP_LAST                     MGR_ID REPORTLEVEL MGR_LIST
 ---------- ------------------------- ---------- ----------- --------
       101 Kochhar                           100           0  100
       108 Greenberg                         101           1  100,101
       200 Whalen                            101           1  100,101
       203 Mavris                            101           1  100,101
       204 Baer                              101           1  100,101
       205 Higgins                           101           1  100,101
       109 Faviet                            108           2  100,101,108
       110 Chen                              108           2  100,101,108
       111 Sciarra                           108           2  100,101,108
       112 Urman                             108           2  100,101,108
       113 Popp                              108           2  100,101,108
       206 Gietz                             205           2  100,101,205
```

次の文は、従業員101の直接的または間接的な部下である従業員とそのレポート・レベルを表示します。これは、レポート・レベル1で停止します。

```
WITH
  reports_to_101 (eid, emp_last, mgr_id, reportLevel) AS
  (
    SELECT employee_id, last_name, manager_id, 0 reportLevel
    FROM employees
    WHERE employee_id = 101
  UNION ALL
    SELECT e.employee_id, e.last_name, e.manager_id, reportLevel+1
    FROM reports_to_101 r, employees e
    WHERE r.eid = e.manager_id
  )
SELECT eid, emp_last, mgr_id, reportLevel
FROM reports_to_101
WHERE reportLevel <= 1
ORDER BY reportLevel, eid;
       EID EMP_LAST                    MGR_ID REPORTLEVEL
---------- ------------------------- ---------- -----------
       101 Kochhar                           100           0
       108 Greenberg                         101           1
       200 Whalen                            101           1
       203 Mavris                            101           1
```

```
       204 Baer                            101           1
       205 Higgins                         101           1
```

次の文は、管理レベルごとにインデントを挿入して、組織全体を表示します。

```
WITH
  org_chart (eid, emp_last, mgr_id, reportLevel, salary, job_id) AS
  (
    SELECT employee_id, last_name, manager_id, 0 reportLevel, salary, job_id
    FROM employees
    WHERE manager_id is null
  UNION ALL
    SELECT e.employee_id, e.last_name, e.manager_id,
           r.reportLevel+1 reportLevel, e.salary, e.job_id
    FROM org_chart r, employees e
    WHERE r.eid = e.manager_id
  )
  SEARCH DEPTH FIRST BY emp_last SET order1
SELECT lpad(' ',2*reportLevel)||emp_last emp_name, eid, mgr_id, salary, job_id
FROM org_chart
ORDER BY order1;
EMP_NAME                   EID     MGR_ID     SALARY JOB_ID
-------------------- ---------- ---------- ---------- ----------
King                       100                 24000 AD_PRES
  Cambrault                148        100      11000 SA_MAN
    Bates                  172        148       7300 SA_REP
    Bloom                  169        148      10000 SA_REP
    Fox                    170        148       9600 SA_REP
    Kumar                  173        148       6100 SA_REP
    Ozer                   168        148      11500 SA_REP
    Smith                  171        148       7400 SA_REP
  De Haan                  102        100      17000 AD_VP
    Hunold                 103        102       9000 IT_PROG
      Austin               105        103       4800 IT_PROG
      Ernst                104        103       6000 IT_PROG
      Lorentz              107        103       4200 IT_PROG
      Pataballa            106        103       4800 IT_PROG
  Errazuriz                147        100      12000 SA_MAN
    Ande                   166        147       6400 SA_REP
. . .
```

次の文は組織全体を表示し、管理レベルごとにインデントを挿入します(各レベルの順序はhire_dateによって決定します)。is_cycleの値がYに設定されるのは、その従業員のhire_dateと同じ値を持つマネージャが管理チェーン内の上位にいる場合です。

```
WITH
  dup_hiredate (eid, emp_last, mgr_id, reportLevel, hire_date, job_id) AS
  (
    SELECT employee_id, last_name, manager_id, 0 reportLevel, hire_date, job_id
    FROM employees
    WHERE manager_id is null
  UNION ALL
    SELECT e.employee_id, e.last_name, e.manager_id,
           r.reportLevel+1 reportLevel, e.hire_date, e.job_id
    FROM dup_hiredate r, employees e
    WHERE r.eid = e.manager_id
  )
  SEARCH DEPTH FIRST BY hire_date SET order1
  CYCLE hire_date SET is_cycle TO 'Y' DEFAULT 'N'
SELECT lpad(' ',2*reportLevel)||emp_last emp_name, eid, mgr_id,
       hire_date, job_id, is_cycle
FROM dup_hiredate
ORDER BY order1;
EMP_NAME                   EID     MGR_ID HIRE_DATE JOB_ID     IS_CYCLE
-------------------- ---------- ---------- --------- ---------- --------
```

```
King                      100            17-JUN-03 AD_PRES           N
  De Haan                 102        100 13-JAN-01 AD_VP             N
    Hunold                103        102 03-JAN-06 IT_PROG           N
      Austin              105        103 25-JUN-05 IT_PROG           N
. . .
  Kochhar                 101        100 21-SEP-05 AD_VP             N
    Mavris                203        101 07-JUN-02 HR_REP            N
    Baer                  204        101 07-JUN-02 PR_REP            N
    Higgins               205        101 07-JUN-02 AC_MGR            N
      Gietz               206        205 07-JUN-02 AC_ACCOUNT        Y
    Greenberg             108        101 17-AUG-02 FI_MGR            N
      Faviet              109        108 16-AUG-02 FI_ACCOUNT        N
      Chen                110        108 28-SEP-05 FI_ACCOUNT        N
. . .
```

次の文は、各マネージャの配下の従業員の人数をカウントします。

```
WITH
  emp_count (eid, emp_last, mgr_id, mgrLevel, salary, cnt_employees) AS
  (
    SELECT employee_id, last_name, manager_id, 0 mgrLevel, salary, 0 cnt_employees
    FROM employees
  UNION ALL
    SELECT e.employee_id, e.last_name, e.manager_id,
           r.mgrLevel+1 mgrLevel, e.salary, 1 cnt_employees
    FROM emp_count r, employees e
    WHERE e.employee_id = r.mgr_id
  )
  SEARCH DEPTH FIRST BY emp_last SET order1
SELECT emp_last, eid, mgr_id, salary, sum(cnt_employees), max(mgrLevel) mgrLevel
FROM emp_count
GROUP BY emp_last, eid, mgr_id, salary
HAVING max(mgrLevel) > 0
ORDER BY mgr_id NULLS FIRST, emp_last;
EMP_LAST                   EID     MGR_ID     SALARY SUM(CNT_EMPLOYEES)   MGRLEVEL
------------------ ---------- ---------- ---------- ------------------ ----------
King                       100                 24000                106          3
Cambrault                  148        100      11000                  7          2
De Haan                    102        100      17000                  5          2
Errazuriz                  147        100      12000                  6          1
Fripp                      121        100       8200                  8          1
Hartstein                  201        100      13000                  1          1
Kaufling                   122        100       7900                  8          1
. . .
```

分析ビュー: 例

次の文は、永続分析ビューsales_avを使用します。この問合せでは、time_hierのmember_name階層属性(同名の階層の別名)、およびtime_hier階層で使用される時間属性ディメンションによりディメンション化される分析ビューのsalesメジャーおよびunitsメジャーの値を選択します。選択結果がフィルタ処理され、階層のYEARレベルの結果に絞り込まれます。結果が階層順に戻されます。

```
SELECT time_hier.member_name as TIME,
 sales,
 units
FROM
 sales_av HIERARCHIES(time_hier)
WHERE time_hier.level_name = 'YEAR'
ORDER BY time_hier.hier_order;
```

問合せの結果は次のとおりです。

```
TIME    SALES           UNITS
------  -------------  ---------
```

```
CY2011  6755115980.73  24462444
CY2012  6901682398.95  24400619
CY2013  7240938717.57  24407259
CY2014  7579746352.89  24402666
CY2015  7941102885.15  24475206
```

一時分析ビューの例

次の文は、WITH句で一時分析ビューmy_avを定義します。この一時分析ビューは、永続分析ビューsales_avに基づいています。lag_sales計算済メジャーは、問合せ時に使用されるLAG計算です。

```
WITH
  my_av ANALYTIC VIEW AS (
    USING sales_av HIERARCHIES (time_hier)
    ADD MEASURES (
      lag_sales AS (LAG(sales) OVER (HIERARCHY time_hier OFFSET 1))
    )
  )
SELECT time_hier.member_name time, sales, lag_sales
FROM my_av HIERARCHIES (time_hier)
WHERE time_hier.level_name = 'YEAR'
ORDER BY time_hier.hier_order;
```

問合せの結果は次のとおりです。

```
TIME         SALES   LAG_SALES
------  ----------  ----------
CY2011  6755115981      (null)
CY2012  6901682399  6755115981
CY2013  7240938718  6901682399
CY2014  7579746353  7240938718
CY2015  7941102885  7579746353
```

次の文は、filter句を使用する一時分析ビューを定義します。

```
WITH
  my_av ANALYTIC VIEW AS (
    USING sales_av HIERARCHIES (time_hier)
    FILTER FACT (
      time_hier TO quarter_of_year IN (1, 2)
        AND year_name IN ('CY2011', 'CY2012')
    )
  )
SELECT time_hier.member_name time, sales
  FROM my_av HIERARCHIES (time_hier)
  WHERE time_hier.level_name IN ('YEAR', 'QUARTER')
  ORDER BY time_hier.hier_order;
```

問合せの結果は次のとおりです。

```
TIME           SALES
--------  ----------
CY2011    3340459835
Q1CY2011  1625299627
Q2CY2011  1715160208
CY2012    3397271965
Q1CY2012  1644857783
Q2CY2012  1752414182
```

インライン分析ビューの例

次の文は、FROM句でインライン分析ビューを定義します。この一時分析ビューは、永続分析ビューsales_avに基づいています。lag_sales計算済メジャーは、問合せ時に使用されるLAG計算です。

```
SELECT time_hier.member_name time, sales, lag_sales
FROM
  ANALYTIC VIEW (
    USING sales_av HIERARCHIES (time_hier)
    ADD MEASURES (
      lag_sales AS (LAG(sales) OVER (HIERARCHY time_hier OFFSET 1))
    )
  )
WHERE time_hier.level_name = 'YEAR'
ORDER BY time_hier.hier_order;
```

問合せの結果は次のとおりです。

```
TIME         SALES   LAG_SALES
------  ----------  ----------
CY2011  6755115981      (null)
CY2012  6901682399  6755115981
CY2013  7240938718  6901682399
CY2014  7579746353  7240938718
CY2015  7941102885  7579746353
```

Simple Query Examples

次の文は、部門番号30の従業員表employeesの行を選択します。

```
SELECT *
   FROM employees
   WHERE department_id = 30
   ORDER BY last_name;
```

次の文は、部門番号30の購買係を除くすべての従業員の名前、職種、給与および部門番号を選択します。

```
SELECT last_name, job_id, salary, department_id
   FROM employees
   WHERE NOT (job_id = 'PU_CLERK' AND department_id = 30)
   ORDER BY last_name;
```

次の文は、FROM句の副問合せから、すべての従業員数と給与合計がすべての部門に対して占める割合を部門ごとに戻します。

```
SELECT a.department_id "Department",
   a.num_emp/b.total_count "%_Employees",
   a.sal_sum/b.total_sal "%_Salary"
FROM
(SELECT department_id, COUNT(*) num_emp, SUM(salary) sal_sum
   FROM employees
   GROUP BY department_id) a,
(SELECT COUNT(*) total_count, SUM(salary) total_sal
   FROM employees) b
ORDER BY a.department_id;
```

パーティションからの選択: 例

FROM句にキーワードPARTITIONを指定することによって、パーティション表の1つのパーティションから行を選択できます。次のSQL文は、サンプル表sh.salesのsales_q2_2000パーティションへ別名を割り当て、行を取り出します。

```
SELECT * FROM sales PARTITION (sales_q2_2000) s
   WHERE s.amount_sold > 1500
   ORDER BY cust_id, time_id, channel_id;
```

次の文は、oe.orders表から、指定した日付より早い注文が含まれる行を選択します。

```
SELECT * FROM orders
   WHERE order_date < TO_DATE('2006-06-15', 'YYYY-MM-DD');
```

サンプルの選択: 例

次の問合せは、oe.orders表内の注文数を推定します。

```
SELECT COUNT(*) * 10 FROM orders SAMPLE (10);
COUNT(*)*10
-----------
         70
```

問合せでは推定値が戻されるため、実際の戻り値は問合せを行うたびに異なる場合があります。

```
SELECT COUNT(*) * 10 FROM orders SAMPLE (10);
COUNT(*)*10
-----------
         80
```

次の問合せでは、前述の問合せにシード値を追加します。同じシード値では、常に同じ推定値が戻されます。

```
SELECT COUNT(*) * 10 FROM orders SAMPLE(10) SEED (1);
COUNT(*)*10
-----------
        130
SELECT COUNT(*) * 10 FROM orders SAMPLE(10) SEED(4);
COUNT(*)*10
-----------
        120
SELECT COUNT(*) * 10 FROM orders SAMPLE(10) SEED (1);
COUNT(*)*10
-----------
        130
```

フラッシュバック問合せの使用方法: 例

次の文は、サンプル表hr.employeesの現在の値を表示し、値を変更します。この例の目的はデモであるため、使用している時間隔が非常に短くなっています。実際の環境での時間隔は、これより長いのが一般的です。

```
SELECT salary FROM employees
   WHERE last_name = 'Chung';

    SALARY
----------
      3800
UPDATE employees SET salary = 4000
   WHERE last_name = 'Chung';
1 row updated.
SELECT salary FROM employees
   WHERE last_name = 'Chung';
    SALARY
----------
      4000
```

更新前の値を確認するには、次のフラッシュバック問合せを使用します。

```
SELECT salary FROM employees
   AS OF TIMESTAMP (SYSTIMESTAMP - INTERVAL '1' MINUTE)
   WHERE last_name = 'Chung';

    SALARY
----------
      3800
```

過去の特定の期間における値を確認するには、次のバージョン・フラッシュバック問合せを使用します。

```
SELECT salary FROM employees
```

```
  VERSIONS BETWEEN TIMESTAMP
    SYSTIMESTAMP - INTERVAL '10' MINUTE AND
    SYSTIMESTAMP - INTERVAL '1' MINUTE
  WHERE last_name = 'Chung';
```

元の値に戻すには、フラッシュバック問合せを別のUPDATE文の副問合せとして使用します。

```
UPDATE employees SET salary =
   (SELECT salary FROM employees
   AS OF TIMESTAMP (SYSTIMESTAMP - INTERVAL '2' MINUTE)
   WHERE last_name = 'Chung')
   WHERE last_name = 'Chung';
1 row updated.
SELECT salary FROM employees
   WHERE last_name = 'Chung';

    SALARY
----------
      3800
```

GROUP BY句の使用方法: 例

次の文は、employees表内の部門ごとに最高給与と最低給与を戻します。

```
SELECT department_id, MIN(salary), MAX (salary)
     FROM employees
     GROUP BY department_id
   ORDER BY department_id;
```

次の文は、各部門の事務員について最高給与と最低給与を戻します。

```
SELECT department_id, MIN(salary), MAX (salary)
     FROM employees
     WHERE job_id = 'PU_CLERK'
     GROUP BY department_id
   ORDER BY department_id;
```

GROUP BY CUBE句の使用方法: 例

部門および職種のすべての組合せについて、従業員数と平均年収を戻すには、サンプル表hr.employeesおよびhr.departmentsに次の問合せを発行します。

```
SELECT DECODE(GROUPING(department_name), 1, 'All Departments',
      department_name) AS department_name,
   DECODE(GROUPING(job_id), 1, 'All Jobs', job_id) AS job_id,
   COUNT(*) "Total Empl", AVG(salary) * 12 "Average Sal"
   FROM employees e, departments d
   WHERE d.department_id = e.department_id
   GROUP BY CUBE (department_name, job_id)
   ORDER BY department_name, job_id;
DEPARTMENT_NAME                JOB_ID     Total Empl Average Sal
------------------------------ ---------- ---------- -----------
Accounting                     AC_ACCOUNT          1       99600
Accounting                     AC_MGR              1      144000
Accounting                     All Jobs            2      121800
Administration                 AD_ASST             1       52800
. . .
Shipping                       ST_CLERK           20       33420
Shipping                       ST_MAN              5       87360
```

GROUPING SETS句の使用方法: 例

次の例では、正確に指定された3つのグループについて集計された売上合計を検出します。

- (channel_desc, calendar_month_desc, country_id)
- (channel_desc, country_id)
- (calendar_month_desc, country_id)

GROUPING SETS構文を指定しない場合、SQLはより複雑になり、問合せの効果は低くなります。たとえば、3つの個別の問合せを実行し、UNION演算を行うか、またはCUBE(channel_desc, calendar_month_desc, country_id)操作を指定した問合せを実行し、生成される8つのグループから5つを除去します。

```
SELECT channel_desc, calendar_month_desc, co.country_id,
      TO_CHAR(sum(amount_sold) , '9,999,999,999') SALES$
   FROM sales, customers, times, channels, countries co
   WHERE sales.time_id=times.time_id
      AND sales.cust_id=customers.cust_id
      AND sales.channel_id= channels.channel_id
      AND customers.country_id = co.country_id
      AND channels.channel_desc IN ('Direct Sales', 'Internet')
      AND times.calendar_month_desc IN ('2000-09', '2000-10')
      AND co.country_iso_code IN ('UK', 'US')
  GROUP BY GROUPING SETS(
      (channel_desc, calendar_month_desc, co.country_id),
      (channel_desc, co.country_id),
      (calendar_month_desc, co.country_id) );
CHANNEL_DESC         CALENDAR COUNTRY_ID     SALES$
-------------------- -------- ----------     ----------
Internet             2000-09       52790        124,224
Direct Sales         2000-09       52790        638,201
Internet             2000-10       52790        137,054
Direct Sales         2000-10       52790        682,297
                     2000-09       52790        762,425
                     2000-10       52790        819,351
Internet                           52790        261,278
Direct Sales                       52790      1,320,497
```

関連項目:

これらのファンクションの詳細は、「GROUP_ID」、「GROUPING」および「GROUPING_ID」を参照してください。

階層問合せの例

CONNECT BY句を指定した次の問合せは、親である行のemployee_id値が子である行のmanager_id値と等しいという階層関係を定義します。

```
SELECT last_name, employee_id, manager_id FROM employees
   CONNECT BY employee_id = manager_id
   ORDER BY last_name;
```

次のCONNECT BY句では、PRIOR演算子がemployee_id値のみに適用されます。この条件を評価するために、データベースは親である行に対してはemployee_idの値を評価し、子である行に対してはmanager_id、salaryおよびcommission_pctのそれぞれの値を評価します。

```
SELECT last_name, employee_id, manager_id FROM employees
   CONNECT BY PRIOR employee_id = manager_id
   AND salary > commission_pct
   ORDER BY last_name;
```

子である行を限定する場合、manager_idの値と親である行のemployee_idの値が等しく、salaryの値がcommission_pctの値より大きい必要があります。

HAVING条件の使用方法: 例

次の文は、従業員の最低給与が$5,000未満の部門についての最高給与と最低給与を戻します。

```
SELECT department_id, MIN(salary), MAX (salary)
   FROM employees
   GROUP BY department_id
   HAVING MIN(salary) < 5000
   ORDER BY department_id;
DEPARTMENT_ID MIN(SALARY) MAX(SALARY)
------------- ----------- -----------
           10        4400        4400
           30        2500       11000
           50        2100        8200
           60        4200        9000
```

次の例は、HAVING句で相関副問合せを使用して、マネージャのいない部門および部門のないマネージャを結果セットから除外しています。

```
SELECT department_id, manager_id
   FROM employees
   GROUP BY department_id, manager_id HAVING (department_id, manager_id) IN
   (SELECT department_id, manager_id FROM employees x
      WHERE x.department_id = employees.department_id)
   ORDER BY department_id;
```

ORDER BY句の使用方法: 例

次の文は、employees表からすべての購買係のレコードを選択し、その給与によって降順にソートします。

```
SELECT *
   FROM employees
   WHERE job_id = 'PU_CLERK'
   ORDER BY salary DESC;
```

次の文は、employees表から情報を選択し、最初に部門番号で昇順にソートした後、給与で降順にソートします。

```
SELECT last_name, department_id, salary
   FROM employees
   ORDER BY department_id ASC, salary DESC, last_name;
```

次の文は、前述のSELECT文と同じ情報を選択し、ORDER BY位置表記法を使用します。ここでは、まずdepartment_idで昇順に、次にsalaryで降順に、最後にlast_nameでアルファベット順に順序付けします。

```
SELECT last_name, department_id, salary
   FROM employees
   ORDER BY 2 ASC, 3 DESC, 1;
```

MODEL句: 例

次に作成するビューはサンプルのshスキーマに基づいており、この後に示す例で使用します。

```
CREATE OR REPLACE VIEW sales_view_ref AS
  SELECT country_name country,
         prod_name prod,
         calendar_year year,
         SUM(amount_sold) sale,
         COUNT(amount_sold) cnt
    FROM sales,times,customers,countries,products
    WHERE sales.time_id = times.time_id
      AND sales.prod_id = products.prod_id
      AND sales.cust_id = customers.cust_id
      AND customers.country_id = countries.country_id
      AND ( customers.country_id = 52779
            OR customers.country_id = 52776 )
      AND ( prod_name = 'Standard Mouse'
```

```
            OR prod_name = 'Mouse Pad' )
    GROUP BY country_name,prod_name,calendar_year;
SELECT country, prod, year, sale
  FROM sales_view_ref
  ORDER BY country, prod, year;
COUNTRY       PROD                                      YEAR        SALE
----------    ----------------------------------      --------   ---------
France        Mouse Pad                                   1998     2509.42
France        Mouse Pad                                   1999     3678.69
France        Mouse Pad                                   2000     3000.72
France        Mouse Pad                                   2001     3269.09
France        Standard Mouse                              1998     2390.83
France        Standard Mouse                              1999     2280.45
France        Standard Mouse                              2000     1274.31
France        Standard Mouse                              2001     2164.54
Germany       Mouse Pad                                   1998     5827.87
Germany       Mouse Pad                                   1999     8346.44
Germany       Mouse Pad                                   2000     7375.46
Germany       Mouse Pad                                   2001     9535.08
Germany       Standard Mouse                              1998     7116.11
Germany       Standard Mouse                              1999     6263.14
Germany       Standard Mouse                              2000     2637.31
Germany       Standard Mouse                              2001     6456.13

16 rows selected.
```

次の例では、国、製品、年および売上が格納されている列が存在するsales_view_refから多次元配列を作成します。さらに、次のことも行われます。

- 2001年のマウス・パッドの売上を含む行が存在する場合、1999年および2000年のマウス・パッドの売上を2001年のマウス・パッドの売上に割り当てます。
- 2002年のスタンダード・マウスの売上を含む行が存在しない場合、2001年のスタンダード・マウスの売上を2002年のスタンダード・マウスの売上に割り当てます。

```
SELECT country,prod,year,s
  FROM sales_view_ref
  MODEL
    PARTITION BY (country)
    DIMENSION BY (prod, year)
    MEASURES (sale s)
    IGNORE NAV
    UNIQUE DIMENSION
    RULES UPSERT SEQUENTIAL ORDER
    (
      s[prod='Mouse Pad', year=2001] =
        s['Mouse Pad', 1999] + s['Mouse Pad', 2000],
      s['Standard Mouse', 2002] = s['Standard Mouse', 2001]
    )
  ORDER BY country, prod, year;

COUNTRY       PROD                                      YEAR        SALE
----------    ----------------------------------      --------   ---------
France        Mouse Pad                                   1998     2509.42
France        Mouse Pad                                   1999     3678.69
France        Mouse Pad                                   2000     3000.72
France        Mouse Pad                                   2001     6679.41
France        Standard Mouse                              1998     2390.83
France        Standard Mouse                              1999     2280.45
France        Standard Mouse                              2000     1274.31
France        Standard Mouse                              2001     2164.54
France        Standard Mouse                              2002     2164.54
Germany       Mouse Pad                                   1998     5827.87
Germany       Mouse Pad                                   1999     8346.44
Germany       Mouse Pad                                   2000     7375.46
```

```
Germany       Mouse Pad                                          2001     15721.9
Germany       Standard Mouse                                     1998     7116.11
Germany       Standard Mouse                                     1999     6263.14
Germany       Standard Mouse                                     2000     2637.31
Germany       Standard Mouse                                     2001     6456.13
Germany       Standard Mouse                                     2002     6456.13
18 rows selected.
```

最初のルールでは、ルールの左側で記号参照が使用されているため、UPDATE動作が使用されます。ルールの左側で表される行が存在するため、メジャー列が更新されます。行が存在しない場合、何も実行されません。

2番目のルールでは、ルールの左側で位置参照が使用され、単一セルが参照されているため、UPSERT動作が使用されます。行が存在しないため、新しい行が追加され、関連するメジャー列が更新されます。行が存在する場合、メジャー列は更新されません。

関連項目:

詳細および例は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

次の例は、同じsales_view_refビューおよび分析ファンクションSUMを使用して、国および年ごとの累計(csum)を計算しています。

```
SELECT country, year, sale, csum
   FROM
   (SELECT country, year, SUM(sale) sale
    FROM sales_view_ref
    GROUP BY country, year
   )
   MODEL DIMENSION BY (country, year)
         MEASURES (sale, 0 csum)
         RULES (csum[any, any]=
                  SUM(sale) OVER (PARTITION BY country
                                  ORDER BY year
                                  ROWS UNBOUNDED PRECEDING)
                )
   ORDER BY country, year;
COUNTRY               YEAR       SALE       CSUM
--------------- ---------- ---------- ----------
France                1998    4900.25    4900.25
France                1999    5959.14   10859.39
France                2000    4275.03   15134.42
France                2001    5433.63   20568.05
Germany               1998   12943.98   12943.98
Germany               1999   14609.58   27553.56
Germany               2000   10012.77   37566.33
Germany               2001   15991.21   53557.54

8 rows selected.
```

行制限: 例

次の文は、employee_id値が低いほうから5人の従業員を返します。

```
SELECT employee_id, last_name
  FROM employees
  ORDER BY employee_id
  FETCH FIRST 5 ROWS ONLY;
EMPLOYEE_ID LAST_NAME
----------- -------------------------
        100 King
        101 Kochhar
        102 De Haan
```

```
        103 Hunold
        104 Ernst
```

次の文は、その次にemployee_id値が低いほうから5人の従業員を返します。

```
SELECT employee_id, last_name
  FROM employees
  ORDER BY employee_id
  OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY;
EMPLOYEE_ID LAST_NAME
----------- -------------------------
        105 Austin
        106 Pataballa
        107 Lorentz
        108 Greenberg
        109 Faviet
```

次の文は、給与が低いほうから5パーセントの従業員を返します。

```
SELECT employee_id, last_name, salary
  FROM employees
  ORDER BY salary
  FETCH FIRST 5 PERCENT ROWS ONLY;
EMPLOYEE_ID LAST_NAME                 SALARY
----------- ------------------------- ----------
        132 Olson                           2100
        128 Markle                          2200
        136 Philtanker                      2200
        127 Landry                          2400
        135 Gee                             2400
        119 Colmenares                      2500
```

WITH TIESが指定されているため、次の文は、給与が低いほうから5パーセントの従業員と、前述の例で最後にフェッチされた行と同じ給与の追加の従業員が返されれます。

```
SELECT employee_id, last_name, salary
  FROM employees
  ORDER BY salary
  FETCH FIRST 5 PERCENT ROWS WITH TIES;
EMPLOYEE_ID LAST_NAME                 SALARY
----------- ------------------------- ----------
        132 Olson                           2100
        128 Markle                          2200
        136 Philtanker                      2200
        127 Landry                          2400
        135 Gee                             2400
        119 Colmenares                      2500
        131 Marlow                          2500
        140 Patel                           2500
        144 Vargas                          2500
        182 Sullivan                        2500
        191 Perkins                         2500
```

FOR UPDATE句の使用方法: 例

次の文は、employees表中のオックスフォード勤務(location_idは2500)の購買係の行をロックし、departments表中の購買係が存在するオックスフォードの部門の行をロックします。

```
SELECT e.employee_id, e.salary, e.commission_pct
   FROM employees e, departments d
   WHERE job_id = 'SA_REP'
   AND e.department_id = d.department_id
   AND location_id = 2500
   ORDER BY e.employee_id
```

```
   FOR UPDATE;
```

次の文は、employees表中のオックスフォード勤務の購買係の行のみをロックします。departments表中の行はロックされません。

```
SELECT e.employee_id, e.salary, e.commission_pct
   FROM employees e JOIN departments d
   USING (department_id)
   WHERE job_id = 'SA_REP'
   AND location_id = 2500
   ORDER BY e.employee_id
   FOR UPDATE OF e.salary;
```

WITH CHECK OPTION句の使用方法: 例

次の文は、3番目に挿入される値が副問合せwhere_clauseの条件に違反していても有効です。

```
INSERT INTO (SELECT department_id, department_name, location_id
   FROM departments WHERE location_id < 2000)
   VALUES (9999, 'Entertainment', 2500);
```

ただし、次の文はWITH CHECK OPTION句を含むため、無効になります。

```
INSERT INTO (SELECT department_id, department_name, location_id
   FROM departments WHERE location_id < 2000 WITH CHECK OPTION)
   VALUES (9999, 'Entertainment', 2500);
     *
ERROR at line 2:
ORA-01402: view WITH CHECK OPTION where-clause violation
```

PIVOTおよびUNPIVOTの使用方法: 例

oe.orders表には、注文が発注された日時(order_date)、発注方法(order_mode)および注文の合計数(order_total)に関する情報とその他の情報が含まれています。次の例は、PIVOT句を使用して、order_mode値を列にピボットし、処理中にorder_totalデータを集計して発注モードごとの年間合計を取得する方法を示します。

```
CREATE TABLE pivot_table AS
SELECT * FROM
(SELECT EXTRACT(YEAR FROM order_date) year, order_mode, order_total FROM orders)
PIVOT
(SUM(order_total) FOR order_mode IN ('direct' AS Store, 'online' AS Internet));
SELECT * FROM pivot_table ORDER BY year;
      YEAR      STORE   INTERNET
---------- ---------- ----------
      2004     5546.6
      2006   371895.5   100056.6
      2007  1274078.8  1271019.5
      2008   252108.3   393349.4
```

UNPIVOT句では、入力列ヘッダーが1つ以上の記述子列の値として出力され、入力列値が1つ以上のメジャー列の値として出力されるように、指定した列を変換します。次に示す最初の問合せでは、デフォルトでNULLが除外されています。2番目の問合せは、INCLUDE NULLS句を使用してNULLを組み込むことができることを示しています。

```
SELECT * FROM pivot_table
  UNPIVOT (yearly_total FOR order_mode IN (store AS 'direct',
           internet AS 'online'))
  ORDER BY year, order_mode;
      YEAR ORDER_ YEARLY_TOTAL
---------- ------ ------------
      2004 direct       5546.6
      2006 direct     371895.5
      2006 online     100056.6
```

```
      2007 direct    1274078.8
      2007 online    1271019.5
      2008 direct     252108.3
      2008 online     393349.4
7 rows selected.
SELECT * FROM pivot_table
  UNPIVOT INCLUDE NULLS
    (yearly_total FOR order_mode IN (store AS 'direct', internet AS 'online'))
  ORDER BY year, order_mode;
      YEAR ORDER_ YEARLY_TOTAL
---------- ------ ------------
      2004 direct       5546.6
      2004 online
      2006 direct     371895.5
      2006 online     100056.6
      2007 direct    1274078.8
      2007 online    1271019.5
      2008 direct     252108.3
      2008 online     393349.4
8 rows selected.
```

結合問合せの使用方法: 例

次の例では、問合せで表を結合する様々な方法を示します。最初の例では、等価結合は、それぞれの従業員の名前と職種、およびその従業員が属する部門の番号と名前を戻します。

```
SELECT last_name, job_id, departments.department_id, department_name
   FROM employees, departments
   WHERE employees.department_id = departments.department_id
   ORDER BY last_name, job_id;
LAST_NAME           JOB_ID     DEPARTMENT_ID DEPARTMENT_NAME
------------------- ---------- ------------- ----------------------
Abel                 SA_REP               80 Sales
Ande                 SA_REP               80 Sales
Atkinson             ST_CLERK             50 Shipping
Austin               IT_PROG              60 IT
. . .
```

従業員名およびジョブは部門名と異なる表に格納されているため、このデータを戻すには結合を使用する必要があります。Oracle Databaseでは、次の結合条件に従って2つの表の行が結合されます。

```
employees.department_id = departments.department_id
```

次の等価結合は、すべての販売マネージャの名前、職種、部門番号および部門名を戻します。

```
SELECT last_name, job_id, departments.department_id, department_name
   FROM employees, departments
   WHERE employees.department_id = departments.department_id
   AND job_id = 'SA_MAN'
   ORDER BY last_name;
LAST_NAME           JOB_ID     DEPARTMENT_ID DEPARTMENT_NAME
------------------- ---------- ------------- ----------------------
Cambrault           SA_MAN               80 Sales
Errazuriz           SA_MAN               80 Sales
Partners            SA_MAN               80 Sales
Russell             SA_MAN               80 Sales
Zlotkey             SA_MAN               80 Sales
```

この問合せは、次のwhere_clause条件を使用して'SA_MAN'というjob値を持つ行のみを戻すこと以外は、前述の例と同じです。

副問合せの使用方法: 例

次の文は、従業員'Lorentz'と同じ部門で働く従業員を判断します。

```
SELECT last_name, department_id FROM employees
   WHERE department_id =
     (SELECT department_id FROM employees
      WHERE last_name = 'Lorentz')
   ORDER BY last_name, department_id;
```

次の文は、employees表の職種を変更した(job_history表に示される)すべての従業員の給与を10%上げます。

```
UPDATE employees
    SET salary = salary * 1.1
    WHERE employee_id IN (SELECT employee_id FROM job_history);
```

次の文は、departments表から3つの列のみを伴って新しい表new_departmentsを作成します。

```
CREATE TABLE new_departments
   (department_id, department_name, location_id)
   AS SELECT department_id, department_name, location_id
   FROM departments;
```

自己結合の使用方法: 例

次の問合せは、自己結合を使用して、それぞれの従業員の名前およびその従業員の上司の名前を戻します。出力を短くするためにWHERE句が追加されています。

```
SELECT e1.last_name||' works for '||e2.last_name
   "Employees and Their Managers"
   FROM employees e1, employees e2
   WHERE e1.manager_id = e2.employee_id
      AND e1.last_name LIKE 'R%'
   ORDER BY e1.last_name;
Employees and Their Managers
------------------------------
Rajs works for Mourgos
Raphaely works for King
Rogers works for Kaufling
Russell works for King
```

この問合せの結合条件では、サンプル表employeesに対する別名e1およびe2を使用します。

```
e1.manager_id = e2.employee_id
```

外部結合の使用方法: 例

次の例では、パーティション化された外部結合によって行のデータの欠損を補完し、分析ファンクションの指定および信頼性の高いレポートの書式設定を簡単にする方法を示します。この例では、結合で使用する小規模なデータ表を最初に作成します。

```
SELECT d.department_id, e.last_name
   FROM departments d LEFT OUTER JOIN employees e
   ON d.department_id = e.department_id
   ORDER BY d.department_id, e.last_name;
```

これは、次に示す以前のOracle Databaseの外部結合の構文と同じ問合せです。

```
SELECT d.department_id, e.last_name
   FROM departments d, employees e
   WHERE d.department_id = e.department_id(+)
   ORDER BY d.department_id, e.last_name;
```

この構文ではなく、前述の例で示した、より柔軟性が高いFROM句の結合構文を使用することをお薦めします。

左側外部結合では、従業員のいない部門を含むすべての部門が戻されます。右側外側結合が指定された同一文では、どの

部門にも割り当てられていない従業員を含むすべての従業員が戻されます。

ノート:

前述の例の従業員表には、従業員 Zeuss が追加されていますが、この従業員はサンプル・データには含まれません。

```
SELECT d.department_id, e.last_name
```

FROM departments d RIGHT OUTER JOIN employees e ON d.department_id = e.department_id ORDER BY d.department_id, e.last_name; DEPARTMENT_ID LAST_NAME ------------ ---------- ---------- . . . 110 Gietz 110 Higgins Grant Zeuss

この結果からは、GrantとZeussという従業員のdepartment_idがNULLであるのか、department_idがdepartments表に存在しないのかは不明です。これを確認するには、完全な外部結合が必要です。

```
SELECT d.department_id as d_dept_id, e.department_id as e_dept_id,
      e.last_name
   FROM departments d FULL OUTER JOIN employees e
   ON d.department_id = e.department_id
   ORDER BY d.department_id, e.last_name;
 D_DEPT_ID  E_DEPT_ID LAST_NAME
---------- ---------- -------------------------
  . . .
       110        110 Gietz
       110        110 Higgins
  . . .
       260
       270
                  999 Zeuss
                      Grant
```

この例の列名は、結合状態にある両方の表で同じであるため、結合構文のUSING句を指定することで、共通列機能も使用できます。出力は、一致する2つのdepartment_id列がUSING句によって結合されることを除き、前述の例と同じです。

```
SELECT department_id AS d_e_dept_id, e.last_name
   FROM departments d FULL OUTER JOIN employees e
   USING (department_id)
   ORDER BY department_id, e.last_name;
D_E_DEPT_ID LAST_NAME
----------- -------------------------
  . . .
        110 Higgins
        110 Gietz
  . . .
        260
        270
        999 Zeuss
            Grant
```

パーティション化された外部結合の使用方法: 例

次の例では、パーティション化された外部結合によって行の欠損を補完し、分析計算の指定および信頼性の高いレポートの書式設定を簡単にする方法を示します。この例では、結合で使用する単純な表を最初に作成し、移入します。

```
CREATE TABLE inventory (time_id    DATE,
                        product    VARCHAR2(10),
                        quantity   NUMBER);
INSERT INTO inventory VALUES (TO_DATE('01/04/01', 'DD/MM/YY'), 'bottle', 10);
INSERT INTO inventory VALUES (TO_DATE('06/04/01', 'DD/MM/YY'), 'bottle', 10);
```

```
INSERT INTO inventory VALUES (TO_DATE('01/04/01', 'DD/MM/YY'), 'can', 10);
INSERT INTO inventory VALUES (TO_DATE('04/04/01', 'DD/MM/YY'), 'can', 10);
SELECT times.time_id, product, quantity FROM inventory
   PARTITION BY  (product)
   RIGHT OUTER JOIN times ON (times.time_id = inventory.time_id)
   WHERE times.time_id BETWEEN TO_DATE('01/04/01', 'DD/MM/YY')
      AND TO_DATE('06/04/01', 'DD/MM/YY')
   ORDER BY  2,1;
TIME_ID   PRODUCT      QUANTITY
--------- ---------- ----------
01-APR-01 bottle             10
02-APR-01 bottle
03-APR-01 bottle
04-APR-01 bottle
05-APR-01 bottle
06-APR-01 bottle             10
01-APR-01 can                10
02-APR-01 can
03-APR-01 can
04-APR-01 can                10
05-APR-01 can
06-APR-01 can
12 rows selected.
```

これで、製品ディメンションの各パーティションのデータの密度は、時間ディメンションに沿ってより密になります。ただし、各パーティションに新しく追加された行のquantity列はそれぞれNULLです。さらに役立つようにするには、NULL値を、時間順でそれより前にあるNULL以外の値で置き換えます。このことを実現するには、分析ファンクションLAST_VALUEを問合せ結果に適用します。

```
SELECT time_id, product, LAST_VALUE(quantity IGNORE NULLS)
   OVER (PARTITION BY product ORDER BY time_id) quantity
   FROM ( SELECT times.time_id, product, quantity
             FROM inventory PARTITION BY  (product)
                RIGHT OUTER JOIN times ON (times.time_id = inventory.time_id)
   WHERE times.time_id BETWEEN TO_DATE('01/04/01', 'DD/MM/YY')
      AND TO_DATE('06/04/01', 'DD/MM/YY'))
   ORDER BY  2,1;
TIME_ID   PRODUCT      QUANTITY
--------- ---------- ----------
01-APR-01 bottle             10
02-APR-01 bottle             10
03-APR-01 bottle             10
04-APR-01 bottle             10
05-APR-01 bottle             10
06-APR-01 bottle             10
01-APR-01 can                10
02-APR-01 can                10
03-APR-01 can                10
04-APR-01 can                10
05-APR-01 can                10
06-APR-01 can                10
12 rows selected.
```

関連項目:

時系列計算における欠損補完の詳細および使用例については、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

アンチ結合の使用方法: 例

次の例では、特定の部門のセットに含まれない従業員のリストを選択します。

```
SELECT * FROM employees
   WHERE department_id NOT IN
   (SELECT department_id FROM departments
       WHERE location_id = 1700)
   ORDER BY last_name;
```

セミ結合の使用方法: 例

次の例では、副問合せに一致する行がemployees表に多数存在する場合でも、departments表から1つの行のみが戻されます。employeesのsalary列に索引が定義されていない場合、セミ結合を使用すると、問合せのパフォーマンスが向上します。

```
SELECT * FROM departments
   WHERE EXISTS
   (SELECT * FROM employees
       WHERE departments.department_id = employees.department_id
       AND employees.salary > 2500)
   ORDER BY department_name;
```

CROSS APPLYおよびOUTER APPLY結合の使用例

次の文は、cross_outer_apply_clauseのCROSS APPLY句を使用します。この結合は、結合(departments)の左側にある表の行のうち、結合の右側にあるインライン・ビューから結果を生成する行のみを返します。つまり、この結合は従業員が1人以上の部門のみを返します。WHERE句は、Marketing、OperationsおよびPublic Relations部門のみが含まれるように結果セットを制限します。ただし、Operations部門には従業員がいないため、この部門は結果セットに含まれません。

```
SELECT d.department_name, v.employee_id, v.last_name
  FROM departments d CROSS APPLY (SELECT * FROM employees e
                                  WHERE e.department_id = d.department_id) v
  WHERE d.department_name IN ('Marketing', 'Operations', 'Public Relations')
  ORDER BY d.department_name, v.employee_id;
DEPARTMENT_NAME                EMPLOYEE_ID LAST_NAME
------------------------------ ----------- -------------------------
Marketing                      201         Hartstein
Marketing                      202         Fay
Public Relations               204         Baer
```

次の文では、cross_outer_apply_clauseのOUTER APPLY句を使用します。この結合は、結合の右側にあるインライン・ビューから結果を生成するかどうかにかかわらず、結合(departments)の左側にある表のすべての行を返します。つまり、この結合は従業員がいるかどうかにかかわらず、すべての部門を返します。WHERE句は、Marketing、OperationsおよびPublic Relations部門のみが含まれるように結果セットを制限します。Operations部門には従業員がいませんが、この部門も結果セットに含まれます。

```
SELECT d.department_name, v.employee_id, v.last_name
  FROM departments d OUTER APPLY (SELECT * FROM employees e
                                  WHERE e.department_id = d.department_id) v
  WHERE d.department_name IN ('Marketing', 'Operations', 'Public Relations')
  ORDER by d.department_name, v.employee_id;
DEPARTMENT_NAME                EMPLOYEE_ID LAST_NAME
------------------------------ ----------- -------------------------
Marketing                      201         Hartstein
Marketing                      202         Fay
Operations
Public Relations               204         Baer
```

LATERALインライン・ビューの使用方法: 例

次の例では、2つのオペランドの結合を示します。2番目のオペランドは、WHERE句で最初のオペランドである表eを指定するインライン・ビューです。これは結果的にエラーとなります。

```
SELECT * FROM employees e, (SELECT * FROM departments d
                            WHERE e.department_id = d.department_id);
ORA-00904: "E"."DEPARTMENT_ID": invalid identifier
```

次の例では、2つのオペランドの結合を示します。2番目のオペランドは、WHERE句で最初オペランドである表eを指定するLATERALインライン・ビューであり、エラーが発生することなく成功します。

```
SELECT * FROM employees e, LATERAL(SELECT * FROM departments d
                                   WHERE e.department_id = d.department_id);
```

表のコレクション: 例

DML操作をネストした表に対して実行できるのは、これらの表が表の列として定義されている場合のみです。したがって、INSERT、DELETEまたはUPDATE文のquery_table_expr_clauseがtable_collection_expressionの場合、コレクション式は、TABLEコレクション式を使用して表のネストした表の列を選択する副問合せである必要があります。次の例は、次の使用例に基づいています。

データベースに、department_id列、location_id列、manager_id列を持つhr_info表と、マネージャごとのすべての従業員のlast_name列、department_id列およびsalary列を持つネストした表型peopleの列が含まれていると仮定します。

```
CREATE TYPE people_typ AS OBJECT (
   last_name      VARCHAR2(25),
   department_id  NUMBER(4),
   salary         NUMBER(8,2));
/
CREATE TYPE people_tab_typ AS TABLE OF people_typ;
/
CREATE TABLE hr_info (
   department_id   NUMBER(4),
   location_id     NUMBER(4),
   manager_id      NUMBER(6),
   people          people_tab_typ)
   NESTED TABLE people STORE AS people_stor_tab;
INSERT INTO hr_info VALUES (280, 1800, 999, people_tab_typ());
```

次の例では、hr_info表の部門番号280のネストした表peopleの列に値を挿入します。

```
INSERT INTO TABLE(SELECT h.people FROM hr_info h
   WHERE h.department_id = 280)
   VALUES ('Smith', 280, 1750);
```

次の例では、部門番号280のネストした表peopleを更新します。

```
UPDATE TABLE(SELECT h.people FROM hr_info h
   WHERE h.department_id = 280) p
   SET p.salary = p.salary + 100;
```

次の例では、部門番号280のネストした表peopleを削除します。

```
DELETE TABLE(SELECT h.people FROM hr_info h
   WHERE h.department_id = 280) p
   WHERE p.salary > 1700;
```

コレクション・ネスト解除: 例

ネストした表の列からデータを選択するには、TABLEコレクション式を使用して、ネストした表を表の列として扱います。このプロセスをコレクション・ネスト解除と呼びます。

次の文を使用すると、前述の例で作成したhr_infoからすべての行を取得し、hr_infoのネストした表peopleの列からす

べての行を取得できます。

```
SELECT t1.department_id, t2.* FROM hr_info t1, TABLE(t1.people) t2
   WHERE t2.department_id = t1.department_id;
```

peopleは、hr_infoのネストした表の列ではなく、last_name、department_id、address、hiredateおよびsalary列と別の表であると仮定します。次の文を使用して、前述の例と同じ行を抽出できます。

```
SELECT t1.department_id, t2.*
   FROM hr_info t1, TABLE(CAST(MULTISET(
      SELECT t3.last_name, t3.department_id, t3.salary
         FROM people t3
      WHERE t3.department_id = t1.department_id)
      AS people_tab_typ)) t2;
```

最後に、peopleはhr_info表のネストした表の列でも、表そのものでもないと仮定します。かわりに、すべての従業員の名前、部門および給与を様々な情報から抽出するpeople_funcファンクションを作成しておきます。次の問合せを使用して、前述の例と同様の情報を得ることができます。

```
SELECT t1.department_id, t2.* FROM hr_info t1, TABLE(CAST
   (people_func( ... ) AS people_tab_typ)) t2;
```

関連項目:

コレクション・ネスト解除の別の例は、『Oracle Databaseオブジェクト・リレーショナル開発者ガイド』を参照してください。

LEVEL疑似列の使用方法: 例

次の文は、階層順序内のすべての従業員を戻します。職種がAD_VPである従業員がルート行となるように定義されています。また、親である行の従業員番号が上司の従業員番号となるように、親である行の子である行が定義されています。

```
SELECT LPAD(' ',2*(LEVEL-1)) || last_name org_chart,
        employee_id, manager_id, job_id
    FROM employees
    START WITH job_id = 'AD_VP'
    CONNECT BY PRIOR employee_id = manager_id;
ORG_CHART          EMPLOYEE_ID MANAGER_ID JOB_ID
------------------ ----------- ---------- ----------
Kochhar                    101        100 AD_VP
  Greenberg                108        101 FI_MGR
    Faviet                 109        108 FI_ACCOUNT
    Chen                   110        108 FI_ACCOUNT
    Sciarra                111        108 FI_ACCOUNT
    Urman                  112        108 FI_ACCOUNT
    Popp                   113        108 FI_ACCOUNT
  Whalen                   200        101 AD_ASST
  Mavris                   203        101 HR_REP
  Baer                     204        101 PR_REP
  Higgins                  205        101 AC_MGR
    Gietz                  206        205 AC_ACCOUNT
De Haan                    102        100 AD_VP
  Hunold                   103        102 IT_PROG
    Ernst                  104        103 IT_PROG
    Austin                 105        103 IT_PROG
    Pataballa              106        103 IT_PROG
    Lorentz                107        103 IT_PROG
```

次の文は、前述の例とほぼ同じですが、職種がFI_MGRである従業員を選択しません。

```
SELECT LPAD(' ',2*(LEVEL-1)) || last_name org_chart,
        employee_id, manager_id, job_id
```

```
    FROM employees
    WHERE job_id != 'FI_MGR'
    START WITH job_id = 'AD_VP'
    CONNECT BY PRIOR employee_id = manager_id;
ORG_CHART          EMPLOYEE_ID MANAGER_ID JOB_ID
------------------ ----------- ---------- ----------
Kochhar                    101        100 AD_VP
    Faviet                 109        108 FI_ACCOUNT
    Chen                   110        108 FI_ACCOUNT
    Sciarra                111        108 FI_ACCOUNT
    Urman                  112        108 FI_ACCOUNT
    Popp                   113        108 FI_ACCOUNT
  Whalen                   200        101 AD_ASST
  Mavris                   203        101 HR_REP
  Baer                     204        101 PR_REP
  Higgins                  205        101 AC_MGR
    Gietz                  206        205 AC_ACCOUNT
De Haan                    102        100 AD_VP
  Hunold                   103        102 IT_PROG
    Ernst                  104        103 IT_PROG
    Austin                 105        103 IT_PROG
    Pataballa              106        103 IT_PROG
    Lorentz                107        103 IT_PROG
```

Greenbergが管理する従業員は戻されますが、マネージャGreenbergは戻されません。

次の文も、前述の例と同じですが、LEVEL疑似列を使用して管理階層の最初の2つのレベルのみを選択します。

```
SELECT LPAD(' ',2*(LEVEL-1)) || last_name org_chart,
employee_id, manager_id, job_id
    FROM employees
    START WITH job_id = 'AD_PRES'
    CONNECT BY PRIOR employee_id = manager_id AND LEVEL <= 2;
ORG_CHART          EMPLOYEE_ID MANAGER_ID JOB_ID
------------------ ----------- ---------- ----------
King                       100            AD_PRES
  Kochhar                  101        100 AD_VP
  De Haan                  102        100 AD_VP
  Raphaely                 114        100 PU_MAN
  Weiss                    120        100 ST_MAN
  Fripp                    121        100 ST_MAN
  Kaufling                 122        100 ST_MAN
  Vollman                  123        100 ST_MAN
  Mourgos                  124        100 ST_MAN
  Russell                  145        100 SA_MAN
  Partners                 146        100 SA_MAN
  Errazuriz                147        100 SA_MAN
  Cambrault                148        100 SA_MAN
  Zlotkey                  149        100 SA_MAN
  Hartstein                201        100 MK_MAN
```

分散問合せの使用方法: 例

この例では、ローカル・データベース上のdepartments表をremoteデータベース上のemployees表と結合する問合せを示します。

```
SELECT last_name, department_name
   FROM employees@remote, departments
   WHERE employees.department_id = departments.department_id;
```

相関副問合せの使用方法: 例

次の例では、相関副問合せの一般的な構文を示します。

```
SELECT select_list
```

```
    FROM table1 t_alias1
    WHERE expr operator
        (SELECT column_list
            FROM table2 t_alias2
            WHERE t_alias1.column
               operator t_alias2.column);
UPDATE table1 t_alias1
    SET column =
        (SELECT expr
            FROM table2 t_alias2
            WHERE t_alias1.column = t_alias2.column);
DELETE FROM table1 t_alias1
    WHERE column operator
        (SELECT expr
            FROM table2 t_alias2
            WHERE t_alias1.column = t_alias2.column);
```

次の文は、部門内の平均給与を超える給与を支給されている従業員の情報を戻します。給与情報が格納されている`employees`表に別名を割り当て、相関副問合せではその別名を使用します。

```
SELECT department_id, last_name, salary
   FROM employees x
   WHERE salary > (SELECT AVG(salary)
      FROM employees
      WHERE x.department_id = department_id)
   ORDER BY department_id;
```

親問合せでは、相関副問合せを使用して同一部門の従業員の平均給与を、`employees`表の行ごとに計算します。相関副問合せは、`employees`表の各行について次のステップを実行します。

- 行の`department_id`を判断します。
- `department_id`に基づいて親問合せが評価されます。
- 行の部門の平均給与より高い給与の行がある場合は、その行を戻します。

副問合せは、`employees`表の各行につき1回ずつ評価されます。

DUAL表からの選択: 例

次の文は、現在の日付を戻します。

```
SELECT SYSDATE FROM DUAL;
```

`employees`表から簡単にSYSDATEを選択できますが、このとき、`employees`表のすべての行に対して1件ずつ14行の同じSYSDATEが戻ります。このため、DUALから選択する方が便利です。

順序値の選択: 例

次の文は、`employees_seq`順序を増加させて新しい値を戻します。

```
SELECT employees_seq.nextval
    FROM DUAL;
```

次の文は、`employees_seq`の現在値を選択します。

```
SELECT employees_seq.currval
    FROM DUAL;
```

行パターン一致: 例

この例では、行パターン一致を使用して株価データを問い合せます。次の文では、表`Ticker`を作成し、株価データを表に挿

入します。

```
CREATE TABLE Ticker (SYMBOL VARCHAR2(10), tstamp DATE, price NUMBER);
INSERT INTO Ticker VALUES('ACME', '01-Apr-11', 12);
INSERT INTO Ticker VALUES('ACME', '02-Apr-11', 17);
INSERT INTO Ticker VALUES('ACME', '03-Apr-11', 19);
INSERT INTO Ticker VALUES('ACME', '04-Apr-11', 21);
INSERT INTO Ticker VALUES('ACME', '05-Apr-11', 25);
INSERT INTO Ticker VALUES('ACME', '06-Apr-11', 12);
INSERT INTO Ticker VALUES('ACME', '07-Apr-11', 15);
INSERT INTO Ticker VALUES('ACME', '08-Apr-11', 20);
INSERT INTO Ticker VALUES('ACME', '09-Apr-11', 24);
INSERT INTO Ticker VALUES('ACME', '10-Apr-11', 25);
INSERT INTO Ticker VALUES('ACME', '11-Apr-11', 19);
INSERT INTO Ticker VALUES('ACME', '12-Apr-11', 15);
INSERT INTO Ticker VALUES('ACME', '13-Apr-11', 25);
INSERT INTO Ticker VALUES('ACME', '14-Apr-11', 25);
INSERT INTO Ticker VALUES('ACME', '15-Apr-11', 14);
INSERT INTO Ticker VALUES('ACME', '16-Apr-11', 12);
INSERT INTO Ticker VALUES('ACME', '17-Apr-11', 14);
INSERT INTO Ticker VALUES('ACME', '18-Apr-11', 24);
INSERT INTO Ticker VALUES('ACME', '19-Apr-11', 23);
INSERT INTO Ticker VALUES('ACME', '20-Apr-11', 22);
```

次の問合せでは、行パターン一致を使用し、株価が底値まで下降した後に上昇したすべての事例を検索します。これは一般的にV字形と呼ばれます。問合せでONE ROW PER MATCHが指定され、3つの一致が見つかったため、出力結果には3つの行のみが含まれます。

```
SELECT *
FROM Ticker MATCH_RECOGNIZE (
     PARTITION BY symbol
     ORDER BY tstamp
     MEASURES STRT.tstamp AS start_tstamp,
              LAST(DOWN.tstamp) AS bottom_tstamp,
              LAST(UP.tstamp) AS end_tstamp
     ONE ROW PER MATCH
     AFTER MATCH SKIP TO LAST UP
     PATTERN (STRT DOWN+ UP+)
     DEFINE
        DOWN AS DOWN.price < PREV(DOWN.price),
        UP AS UP.price > PREV(UP.price)
     ) MR
ORDER BY MR.symbol, MR.start_tstamp;
SYMBOL     START_TST BOTTOM_TS END_TSTAM
---------- --------- --------- ---------
ACME       05-APR-11 06-APR-11 10-APR-11
ACME       10-APR-11 12-APR-11 13-APR-11
ACME       14-APR-11 16-APR-11 18-APR-11
```

SET CONSTRAINT[S]

目的

SET CONSTRAINTS文を使用すると、遅延可能な制約の検証を、各DML文の実行後に行うか(IMMEDIATE)、トランザクションのコミット時に行うか(DEFERRED)をトランザクションごとに指定できます。この文を使用して、制約名のリストまたはALL制約のモードを設定できます。

SET CONSTRAINTSモードは、トランザクションの存続期間中、または別のSET CONSTRAINTS文によってモードがリセットされるまで継続します。

> ノート:
>
> ALTER SESSION 文で SET CONSTRAINTS 句を使用して、すべての遅延可能制約を設定することもできます。これは、現在のセッションの各トランザクションの開始時に SET CONSTRAINTS 文を発行することと同等です。

トリガー定義の内部でこの文を指定することはできません。

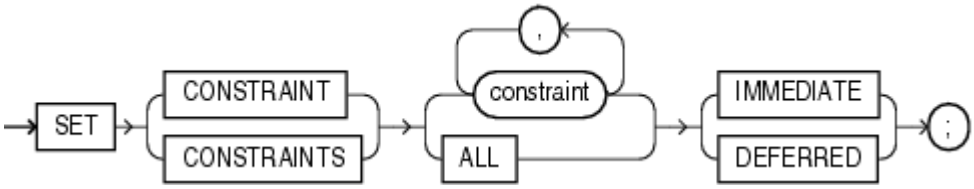
SET CONSTRAINTSは、分散型の文にすることができます。処理中のトランザクションを持つ既存のデータベース・リンクにはSET CONSTRAINTS ALL文の発行時にその発行が通知され、新しいリンクにはトランザクションの開始直後にその発行が通知されます。

前提条件

遅延可能な制約を検証する時期を指定する場合は、表が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、制約が適用される表に対するREADまたはSELECT権限が必要です。

構文

set_constraints::=



セマンティクス

constraint

1つ以上の整合性制約の名前を指定します。

ALL

ALLを指定すると、このトランザクションに対するすべての遅延可能な制約を設定できます。

IMMEDIATE

IMMEDIATEを指定すると、指定された制約が、各制約DML文の実行時に即時にチェックされます。Oracle Databaseでは、チェック済のすべての制約に一貫性があり、他のSET CONSTRAINTS文が発行されていない場合、トランザクションで以前

に遅延された制約が最初にチェックされ、その後すぐにそのトランザクションの他の文の制約チェックが継続してチェックされます。制約のチェックに失敗した場合は、エラーが通知されます。その時点で、COMMIT文を実行すると、トランザクション全体が元に戻されます。

COMMITを正常に実行できるかどうかをチェックする方法としてトランザクションの終了直後に制約を行います。制約をトランザクション内の最後の文としてIMMEDIATEに設定することで、予期しないロールバックを回避できます。いずれかの制約のチェックに失敗した場合は、トランザクションをコミットする前にエラーを解決できます。

DEFERRED

DEFERREDを指定すると、遅延可能な制約によって指定された条件が、トランザクションのコミット時に検証されます。

ノート:

SET CONSTRAINTS ALL IMMEDIATE 文を発行することによって、遅延可能な制約をコミットする前に、それらの制約が完全に適用されたかどうかを検証できます。

例

制約の設定: 例

次の文は、このトランザクション内のすべての遅延可能な制約が、各DML文の直後に検証されるように設定します。

```
SET CONSTRAINTS ALL IMMEDIATE;
```

次の文は、トランザクションのコミット時に3つの遅延制約を検証します。この例では、制約をNOT DEFERRABLEに設定すると失敗します。

```
SET CONSTRAINTS emp_job_nn, emp_salary_min,
   hr.jhist_dept_fk@remote DEFERRED;
```

SET ROLE

目的

データベースでは、ユーザー・ログイン時に、ユーザーに明示的に付与されたすべての権限およびユーザーのすべてのデフォルトのロールが使用可能になります。セッション中、ユーザーまたはアプリケーションは、SET ROLE文を使用して、そのセッションに対してロールを何度でも使用可能または使用禁止にできます。

148を超えるユーザー定義のロールを一度に使用可能にできません。

ノート:

- ほとんどのロールは、直接的に付与されているか、または他のロールを介して付与されていないかぎり、使用可能または使用禁止にできません。ただし、保護アプリケーション・ロールは、関連付けられている PL/SQL パッケージによって付与して使用可能にすることができます。保護アプリケーション・ロールについては、「USING package」の CREATE ROLE セマンティクスおよび『Oracle Database セキュリティ・ガイド』を参照してください。

- SET ROLE が正常に完了するのは、コール・スタックに定義者権限ユニットがない場合のみです。少なくとも 1 つの DR ユニットがコール・スタックにある場合は、SET ROLE コマンドを発行すると ORA-06565 が発生します。定義者権限ユニットの詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

- SET ROLE コマンドを PL/SQL から実行するには、動的 SQL を使用する必要があります(EXECUTE IMMEDIATE 文によって実行するのが望ましい)。この文の詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

SESSION_ROLESデータ・ディクショナリ・ビューを問い合せると、現在使用可能なロールを参照できます。

関連項目:

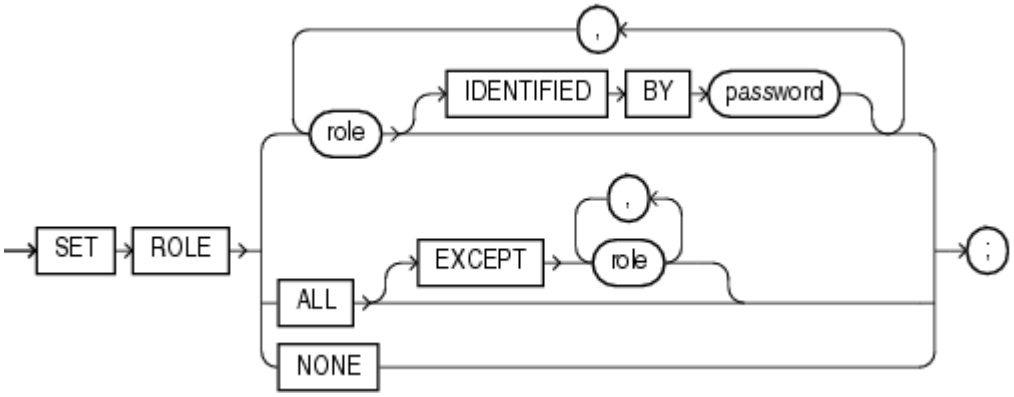
- ロールの作成の詳細は、「CREATE ROLE」を参照してください。
- ユーザーのデフォルト・ロールの変更については、「ALTER USER」を参照してください。
- SESSION_ROLESセッション・パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

前提条件

SET ROLE 文に指定するロールが付与されている必要があります。

構文

set_role::=



セマンティクス

role

現行のセッションで使用可能にするロールを1つ以上指定します。指定されていないすべてのロールは、現行のセッションの存続期間中、または現行のセッションで他のSET ROLE文が発行されるまで使用禁止になります。

IDENTIFIED BY password句では、ロールに対するパスワードを指定します。ロールにパスワードが設定されている場合は、指定する必要があります。

ロールの設定の制限事項

グローバルに識別されたロールを指定することはできません。グローバル・ロールは、ログイン時にデフォルトで使用可能になり、後で再度使用可能にすることはできません。

ALL句

ALLを指定すると、現行のセッションに対して付与されているすべてのロールを使用可能にできます。ただし、EXCEPT句に任意に指定されているロールは除きます。

EXCEPT句に指定するロールは、ユーザーに直接付与されている必要があります。他のロールによってユーザーに付与されたものは無効です。

直接付与されているロール、および他のロールを介してユーザーに付与されているロールをEXCEPT句に指定した場合、そのロールの付与先のロールにより、そのロールは使用可能のままになります。

ALL句の制限事項

ALL句には、次の制限事項が適用されます。

- この句を使用して、ユーザーに直接付与されているパスワード付きのロールを使用可能にすることはできません。
- この句を使用して、保護アプリケーション・ロールを使用可能にすることはできません。保護アプリケーション・ロールとは、認可済パッケージのみを使用するアプリケーションによって使用可能になるロールです。保護アプリケーション・ロールの作成の詳細は『Oracle Databaseセキュリティ・ガイド』、チュートリアルは『Oracle Database 2日でセキュリティ・ガイド』を参照してください。

NONE

NONEを指定すると、現行のセッションで、DEFAULTロールを含むすべてのロールを使用禁止にできます。

例

ロールの設定: 例

次の文は、現行のセッションのパスワードwarehouseによって識別されるロールdw_managerを有効にします。

```
SET ROLE dw_manager IDENTIFIED BY warehouse;
```

次の文は、現行のセッションで付与されているロールをすべて使用可能にします。

```
SET ROLE ALL;
```

次の文は、dw_managerを除くロールをすべて使用可能にします。

```
SET ROLE ALL EXCEPT dw_manager;
```

次の文は、現行のセッションで付与されているすべてのロールを使用禁止にします。

```
SET ROLE NONE;
```

SET TRANSACTION

目的

SET TRANSACTION文を使用すると、現行のトランザクションを読取り専用または読み書き両用として設定したり、分離レベルを設定したり、指定したロールバック・セグメントにトランザクションを割り当てたり、名前をトランザクションに割り当てることができます。

TXロックを取得する操作によって、トランザクションが暗黙的に開始されます。

- データを変更する文が発行されたとき
- SELECT ... FOR UPDATE文が発行されたとき
- SET TRANSACTION文またはDBMS_TRANSACTIONパッケージによってトランザクションが明示的に開始されたとき

COMMIT文またはROLLBACK文を発行すると、現行のトランザクションが明示的に終了されます。

SET TRANSACTION文によって実行される操作の影響が及ぶのは、現行トランザクションのみであり、他のユーザーや他のトランザクションには影響しません。トランザクションは、COMMIT文またはROLLBACK文が発行されると終了します。Oracle Databaseは、データ定義言語(DDL)文の実行前と後に、現行トランザクションを暗黙的にコミットします。

関連項目:

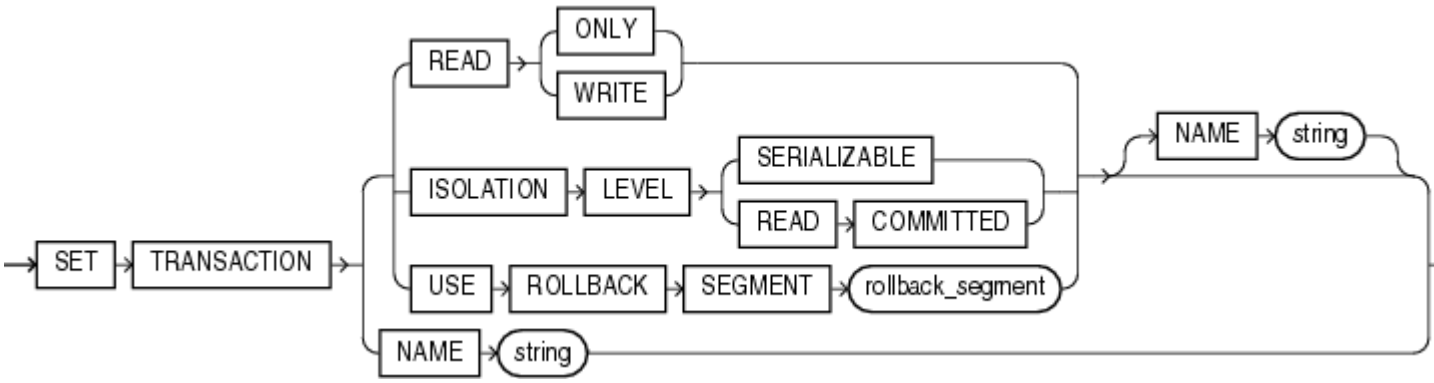
COMMIT and ROLLBACK

前提条件

SET TRANSACTION文を使用する場合、トランザクションの先頭に記述する必要があります。ただし、SET TRANSACTION文が必要ないトランザクションもあります。

構文

set_transaction::=



セマンティクス

READ ONLY

READ ONLY句を使用すると、現行のトランザクションを読取り専用に設定できます。この句では、トランザクション・レベルの読取り一貫性を設定します。

そのトランザクションの後続のすべての問合せでは、トランザクションの開始前にコミットされた変更のみが参照されます。読取り

専用トランザクションは、他のユーザーが更新中の1つ以上の表に対して、複数の問合せを実行するレポートに役立ちます。

ユーザーSYSは、この句を使用できません。SYSによる問合せでは、SYSがトランザクションをREAD ONLYに設定した場合でも、トランザクション中の変更が戻されます。

読取り専用トランザクションの制限事項

読取り専用トランザクションで許可される文は、次の文に限定されます。

- 副問合せ(for_update_clauseを指定しないSELECT文)
- LOCK TABLE
- SET ROLE
- ALTER SESSION
- ALTER SYSTEM

READ WRITE

READ WRITEを指定すると、現行のトランザクションは読み書き両用トランザクションに設定されます。この句によって、文レベルの読取り一貫性が設定されます(これがデフォルトです)。

読み書き両用トランザクションの制限事項

同じトランザクションで、トランザクション・レベルと文レベルの読取り一貫性を切り替えることはできません。

ISOLATION LEVEL句

- SQL規格に定義されているシリアライズ可能トランザクション分離モードを設定する場合に、SERIALIZABLEを指定します。シリアライズ可能トランザクションにデータ操作言語(DML)が含まれている場合、そのDMLがシリアライズ可能トランザクションの開始時にコミットされなかったトランザクション内の更新済のリソースを更新しようとすると、そのDML文は正常に実行されません。
- Oracle Databaseのトランザクションでは、デフォルトでREAD COMMITTEDが設定されています。別のトランザクションで行ロックを保持しておく必要があるDMLがトランザクションに指定されていると、DML文は行ロックが解除されるまで待ち状態になります。

USE ROLLBACK SEGMENT句

> ノート:
>
> この句は、UNDO 用のロールバック・セグメントを使用している場合にのみ有効です。自動 UNDO 管理モードで UNDO 領域を管理することをお薦めします。データベースを自動 UNDO モードで実行すると、この句は無視されます。

USE ROLLBACK SEGMENTを指定すると、現行のトランザクションを、指定したロールバック・セグメントに割り当てることができます。この句によって、現行のトランザクションは暗黙的に読み書き両用トランザクションに設定されます。

パラレルDMLでは、複数のロールバック・セグメントが必要です。したがって、トランザクションにパラレルDML操作が含まれている場合、この句は無視されます。

NAME句

NAME句を使用すると、現行のトランザクションに名前を割り当てることができます。この句は、分散データベース環境でインダウト・トランザクションを識別および変換する場合に便利です。string値の最大長は255バイトです。

分散トランザクションに対して名前を指定する場合、トランザクションのコミット時に、名前はコミットのコメントとなり、COMMIT文で明示的に指定した任意のコメントを上書きします。

関連項目:

トランザクションの名前指定の詳細は、『Oracle Database概要』を参照してください。

例

トランザクションの設定: 例

次の文は、サンプルの注文入力(oe)スキーマのトロントの倉庫にある在庫の製品と量を計算するもので、毎月の最終日の真夜中に実行されます。このレポートは、別の倉庫の在庫を追加および削除する他のユーザーの影響は受けません。

```
COMMIT;
SET TRANSACTION READ ONLY NAME 'Toronto';
SELECT product_id, quantity_on_hand FROM inventories
   WHERE warehouse_id = 5
   ORDER BY product_id;
COMMIT;
```

最初のCOMMIT文によって、SET TRANSACTIONが確実にトランザクション内の最初の文となります。最後のCOMMIT文は、実際にはデータベースに対する変更を永続化するものではありません。これは単に、この読取り専用トランザクションを終了するためのものです。

TRUNCATE CLUSTER

目的

ノート:

TRUNCATE CLUSTER 文はロールバックできません。

TRUNCATE CLUSTER文を使用すると、クラスタからすべての行を削除できます。デフォルトでは、次の処理も実行されます。

- 削除された行が使用していたすべての領域(ただし、MINEXTENTS記憶域パラメータで指定された領域は除く)の割当てが解除されます。
- NEXT記憶域パラメータが、切捨て処理によってセグメントから最後に削除されたエクステントのサイズに設定されます。

クラスタを削除して再作成するより、TRUNCATE文で行を削除する方が効果的です。クラスタを削除して再作成すると、そのクラスタに依存するオブジェクトが無効になり、クラスタに対するオブジェクト権限を再度付与する必要があるだけでなく、表の索引とクラスタを再作成し、その記憶域パラメータを再指定することも必要になります。切捨ての場合は、このような影響はありません。

TRUNCATE CLUSTER文を使用すると、DELETE文を使用してすべての行を削除するよりも迅速に削除できます。特に、クラスタに索引およびその他の依存オブジェクトが多数ある場合に有効です。

関連項目:

- クラスタからデータを削除する他の方法については、「DELETE」および「DROP CLUSTER」を参照してください。
- 表の切捨ての詳細は、「TRUNCATE TABLE」を参照してください。

前提条件

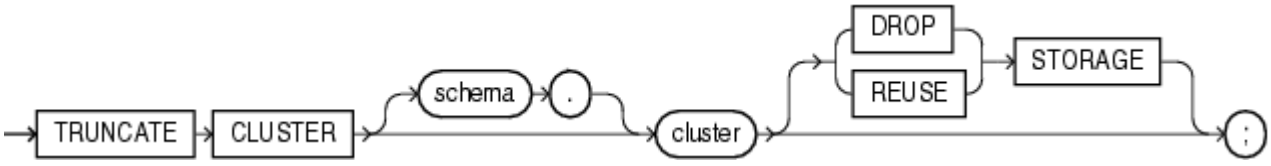
クラスタを切り捨てるには、自分のスキーマ内にそのクラスタがあるか、またはDROP ANY TABLEシステム権限が必要です。

関連項目:

表の切捨ての制限事項

構文

truncate_cluster::=



セマンティクス

CLUSTER句

切り捨てるクラスタが設定されているスキーマと、そのクラスタの名前を指定します。なお、索引クラスタは切り捨てられますが、

ハッシュ・クラスタは切り捨てられません。schemaを指定しない場合、クラスタは自分のスキーマ内にあるとみなされます。

クラスタを切り捨てた場合、そのクラスタにある表のすべての索引データも自動的に削除されます。

STORAGE句

STORAGE句を使用すると、行の切捨てによって解放された領域をどのようにするかを指定できます。DROP STORAGE句およびREUSE STORAGE句は、対応する索引から削除されたデータの空き領域にも適用されます。

DROP STORAGE

DROP STORAGEを指定すると、クラスタのMINEXTENTSパラメータで割り当てられた領域を除き、クラスタから削除された行からすべての領域の割当てを解除できます。この領域は、後で表領域内の他のオブジェクトで使用できます。また、NEXT記憶域パラメータが、切捨て処理によってセグメントから最後に削除されたエクステントのサイズに設定されます。これはデフォルトです。

REUSE STORAGE

REUSE STORAGEを指定すると、クラスタに割り当てられた削除行の領域を確保できます。STORAGEの値は、表またはクラスタを作成したときの値にリセットされません。この領域は、挿入操作または更新操作によってそのクラスタ内に作成される新規データによってのみ使用されます。記憶域パラメータは現行の設定のまま残ります。

切り捨てるオブジェクトに対して複数の空きリストを指定している場合は、REUSE STORAGE句によって、インスタンスへの空きリストのマッピングも削除され、最高水位標は第1エクステントの先頭までリセットされます。

例

クラスタの切捨て: 例

次の文は、personnelクラスタ内の表のすべての行を削除しますが、空き領域は表に割り当てられたままにしておきます。

```
TRUNCATE CLUSTER personnel REUSE STORAGE;
```

この文では、personnelクラスタにある表のすべての索引データも削除されます。

TRUNCATE TABLE

目的

ノート:

TRUNCATE TABLE 文をロールバックしたり、FLASHBACK TABLE 文を使用して、切り捨てられた表の内容を取得することはできません。

TRUNCATE TABLEを使用すると、表からすべての行を削除できます。デフォルトでは、次の処理も実行されます。

- 削除された行が使用していたすべての領域(ただし、MINEXTENTS記憶域パラメータで指定された領域は除く)の割当てが解除されます。
- NEXT記憶域パラメータが、切捨て処理によってセグメントから最後に削除されたエクステントのサイズに設定されます。

表を削除して再作成するより、TRUNCATE TABLE文で行を削除する方が効果的です。表を削除して再作成するとその表の依存オブジェクトが無効になるため、次のアクションを繰り返す必要があります。

- 表に対するオブジェクト権限の付与
- 表の索引、整合性制約およびトリガーの作成
- 表の記憶域パラメータの指定

切捨ての場合は、このような影響はありません。

TRUNCATE TABLE文を使用すると、DELETE文を使用してすべての行を削除するよりも迅速に削除できます。特に、表にトリガー、索引およびその他の依存オブジェクトが多数ある場合に有効です。

関連項目:

- 表からデータを削除する他の方法については、「DELETE」および「DROP TABLE」を参照してください。
- クラスタの切捨ての詳細は、「TRUNCATE CLUSTER」を参照してください。

前提条件

表を切り捨てるには、その表が自分のスキーマ内にあるか、自分にDROP ANY TABLEシステム権限が付与されている必要があります。

CASCADE句を指定するには、影響する子表が自分のスキーマにすべて含まれているか、自分にDROP ANY TABLEシステム権限が付与されている必要があります。

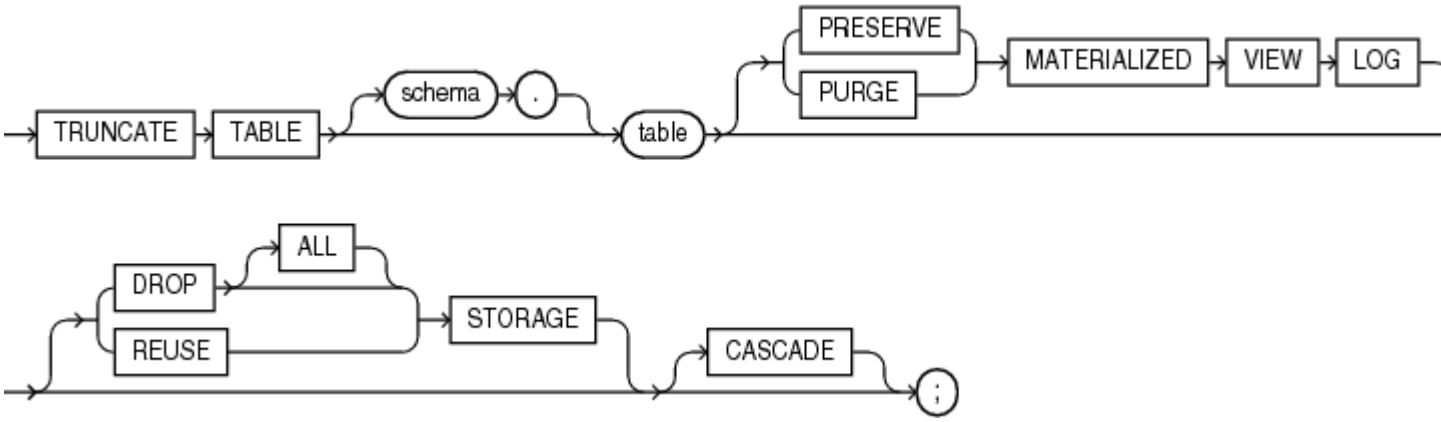
既存のTRUNCATE TABLEコマンドを使用して、プライベート一時表を切り捨てることができます。プライベート一時表を切り捨てても、既存のトランザクションはコミットされません。これは、トランザクション固有とセッション固有の両方のプライベート一時表に適用されます。切り捨てられたプライベート一時表は、RECYCLEBINに移動されないことに注意してください。

関連項目:

表の切捨ての制限事項

構文

truncate_table::=



セマンティクス

TABLE句

切り捨てる表が設定されているスキーマおよびその表の名前を指定します。クラスタを構成する表は、切り捨てることができません。schemaを指定しない場合、表は自分のスキーマ内にあるとみなされます。

- 索引構成表や一時表も切り捨てることができます。一時表を切り捨てた場合、現行のセッションで作成された行のみが削除されます。
- tableの記憶域パラメータNEXTが、切捨て処理中にセグメントから最後に削除されたエクステントのサイズに変更されます。
- tableに対する索引(ローカル索引のレンジ・パーティションとハッシュ・パーティション、およびローカル索引のサブパーティション)のUNUSABLEのインジケータも、自動的に切捨ておよびリセットされます。
- tableが空でない場合は、表中の非パーティション索引およびグローバル・パーティション索引のすべてのパーティションにUNUSABLEのマークが付けられます。ただし、表が切り捨てられると索引も切り捨てられ、索引セグメントに対して新しい最高水位標が計算されます。この操作は、索引に対して新しいセグメントを作成することと同じです。このため、切捨て操作の最後に、索引が再度USABLEになります。
- ドメイン索引の場合は、この文が、適切なTRUNCATEルーチンを起動し、ドメイン索引のデータを切り捨てます。

関連項目:

ドメイン索引の詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。

- 標準表および索引構成表がLOB列を含む場合、すべてのLOBデータおよびLOB索引セグメントは切り捨てられます。
- tableがパーティション化されている場合、各パーティションまたはサブパーティションのLOBデータ・セグメントおよびLOB索引セグメントと同様に、パーティションおよびサブパーティションも切り捨てられます。

ノート:

表を切り捨てた場合、表の索引データおよび表に対応付けられたマテリアライズド・ビューのダイレクト・パス・インサート情報はすべて、自動的に削除されます。この情報は、マテリアライズド・ビュー・ログのいずれにも依存していません。このダイレクト・パス・インサート情報を削除した場合、マテリアライズド・ビューの増分リフレッシュのデータが失われる場合があります。

- カーソルはすべて無効になります。

表の切捨ての制限事項

この文には、次の制限事項があります。

- TRUNCATE TABLE文はロールバックできません。
- 切捨て操作を行う前の表の状態にフラッシュバックすることはできません。
- クラスタを構成する表は、個別に切り捨てることはできません。これを行うには、クラスタを切り捨てるか、表のすべての行を削除するか、または表を削除して再作成する必要があります。
- 使用可能になっている外部キー制約の親である表は、切り捨てることはできません。その表を切り捨てる場合、制約を無効にしておく必要があります。整合性制約が自己参照型の場合は、例外として表を切り捨てることができます。
- tableでドメイン索引が定義されている場合、索引および索引パーティションにIN_PROGRESSのマークを付けることはできません。
- 参照パーティション表の親表は切り捨てることができません。まず、参照パーティション表の子表を削除する必要があります。
- 重複表を切り捨てることはできません。

MATERIALIZED VIEW LOG句

MATERIALIZED VIEW LOG句を使用すると、表が切り捨てられた場合に、この表に定義されているマテリアライズド・ビュー・ログを保存するか、または削除するかを指定できます。この句を使用した場合、マテリアライズド・ビューのマスター表を、エクスポートまたはインポートによって再編成できます。この場合、マスター表で定義された主キー・マテリアライズド・ビューを高速リフレッシュする機能は影響を受けません。主キー・マテリアライズド・ビューを継続的に高速リフレッシュできるようにするには、マテリアライズド・ビュー・ログに主キー情報を記録する必要があります。

ノート:

下位互換性を保つために、MATERIALIZED VIEW のかわりにキーワード SNAPSHOT もサポートされています。

PRESERVE

PRESERVEを指定すると、マスター表を切り捨てたときにマテリアライズド・ビュー・ログを保存できます。これはデフォルトです。

PURGE

PURGEを指定すると、マスター表を切り捨てたときにマテリアライズド・ビュー・ログを削除できます。

関連項目:

マテリアライズド・ビュー・ログおよびTRUNCATE文の詳細は、『Oracle Database管理者ガイド』を参照してください。

STORAGE句

STORAGE句を使用すると、行の切捨てによって解放された領域をどのようにするかを指定できます。DROP STORAGE句、DROP ALL STORAGE句およびREUSE STORAGE句は、対応する索引から削除されたデータの空き領域にも適用されます。

DROP STORAGE

DROP STORAGEを指定すると、表のMINEXTENTSパラメータで割り当てられた領域を除き、表から削除された行からすべての領域の割当てを解除できます。この領域は、後で表領域内の他のオブジェクトで使用できます。また、NEXT記憶域パラメータが、切捨て処理によってセグメントから最後に削除されたエクステントのサイズに設定されます。この設定(デフォルト)は、小規模や中規模のオブジェクトに有効です。このようなオブジェクトの場合、ローカル管理表領域のエクステント管理が非常に高速に行われるため、領域を確保する必要はありません。

DROP ALL STORAGE

DROP ALL STORAGEを指定すると、MINEXTENTSパラメータで割り当てられた領域も含め、表から削除された行からすべての領域の割当てを解除できます。表のすべてのセグメント、およびその依存オブジェクトのすべてのセグメントも、割当てが解除されます。

DROP ALL STORAGEの制限事項

この句には、「遅延セグメント作成の制限事項」に示されているものと同じ制限事項があります。

REUSE STORAGE

REUSE STORAGEを指定すると、表に割り当てられた削除行の領域を確保できます。STORAGEの値は、表を作成したときの値にリセットされません。この領域は、挿入操作または更新操作によってその表内に作成される新規データによってのみ使用されます。記憶域パラメータは現行の設定のまま残ります。

この設定は、行数が非常に多く、表に何千ものエクステントが必要な場合や、データを後で再挿入する予定がある場合に、大規模な表のすべての行を削除する別の方法として有効です。

この句は一時表では無効です。表が切り捨てられたときに、セッションは一時表からアンバインドされるので、記憶域が自動的に削除されます。

切り捨てるオブジェクトに対して複数の空きリストを指定している場合は、REUSE STORAGE句によって、インスタンスへの空きリストのマッピングも削除され、最高水位標は第1エクステントの先頭までリセットされます。

CASCADE

CASCADEを指定すると、ON DELETE CASCADE参照制約が有効化されていて、tableを参照する子表は、すべて切り捨てられます。これは、指定したオプションを使用して、すべての子表、孫表(およびそれ以下の子表)を切り捨てる再帰的な操作になります。

例

表の切捨て: 例

次の文は、サンプル表hr.employeesの仮想コピーのすべての行を削除して、解放された領域をemployees表が定義されている表領域に戻します。

```
TRUNCATE TABLE employees_demo;
```

ここでは、employees表の索引データもすべて削除され、解放された領域は、それらの索引が定義されていた表領域に戻さ

れます。

切捨て後のマテリアライズド・ビュー・ログの保存: 例

次の文は、マテリアライズド・ビュー・ログを保存するTRUNCATE文の例です。

```
TRUNCATE TABLE sales_demo PRESERVE MATERIALIZED VIEW LOG;
TRUNCATE TABLE orders_demo;
```

UPDATE

目的

UPDATE文を使用すると、表、ビューの実表またはマテリアライズド・ビューのマスター表の既存の値を変更できます。

前提条件

表の値を更新する場合は、表が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、その表に対するUPDATEオブジェクト権限が必要です。

ビューの実表の値を更新する場合は、次の条件を満たす必要があります。

- そのビューに対するUPDATEオブジェクト権限を持っている。
- そのビューが含まれているスキーマの所有者が、実表に対するUPDATEオブジェクト権限を持っている。

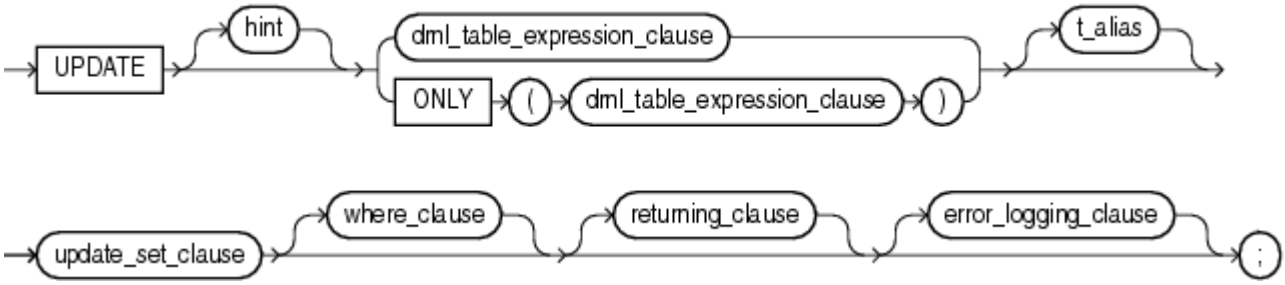
UPDATE ANY TABLEシステム権限を持っている場合は、任意の表またはビューの実表の値を更新できます。

リモート・データベースのオブジェクトの値を更新するには、オブジェクトのREADまたはSELECTオブジェクト権限も必要です。

SQL92_SECURITY初期化パラメータがTRUEに設定され、UPDATE操作がwhere_clauseの列などの表の列を参照する場合、更新するオブジェクトのSELECTオブジェクト権限も必要です。

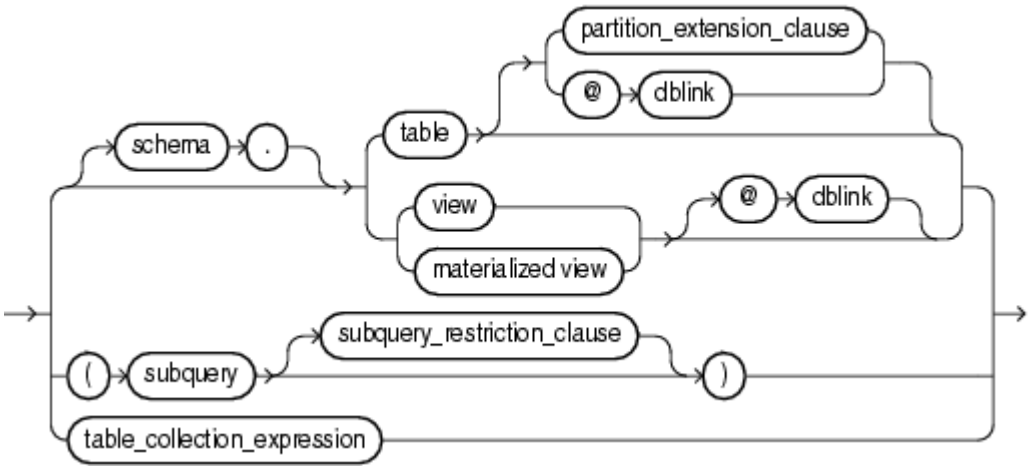
構文

update::=



(DML_table_expression_clause::=、update_set_clause::=、where_clause::=、returning_clause::=、error_logging_clause::=)

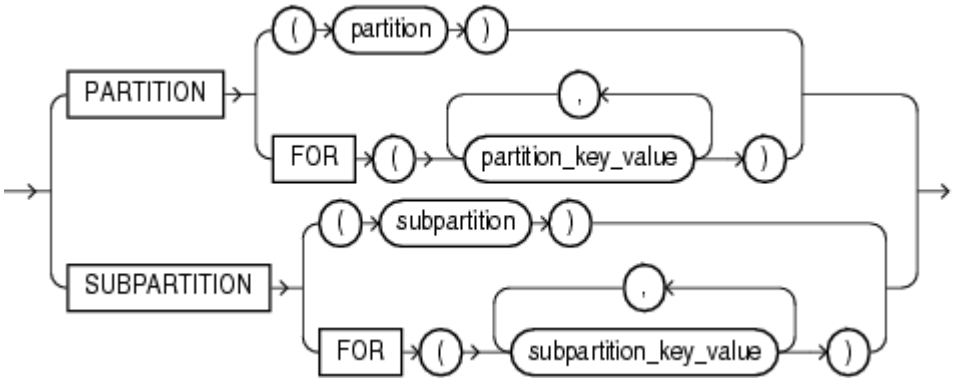
DML_table_expression_clause::=を参照



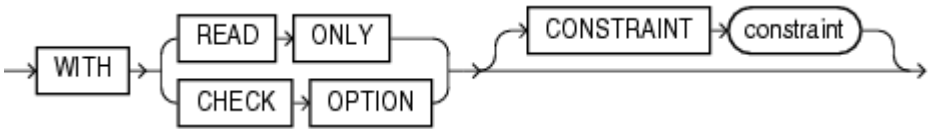
(partition_extension_clause::=、subquery::=(SELECTの一部)、subquery_restriction_clause::=、

table_collection_expression::=)

partition_extension_clause::=

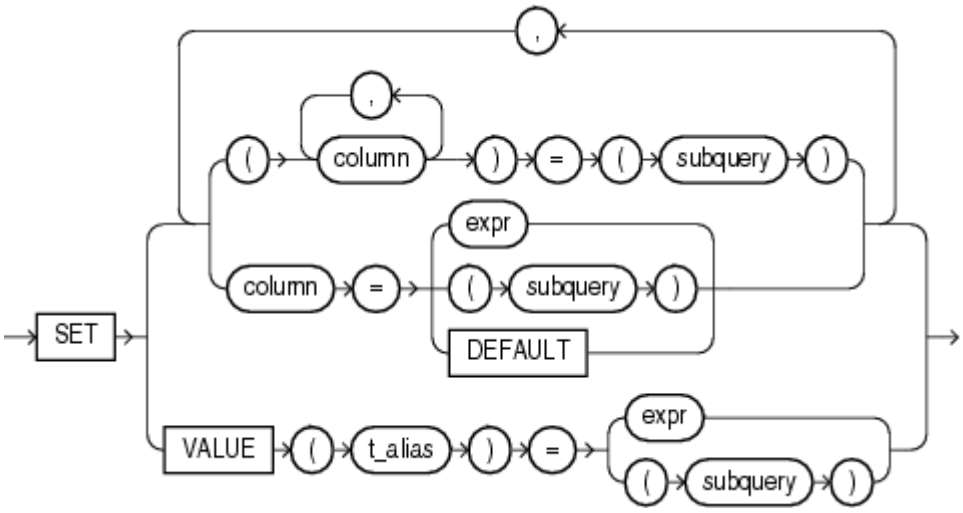


subquery_restriction_clause::=



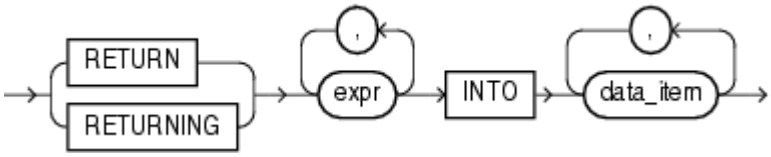
table_collection_expression::=

update_set_clause::=

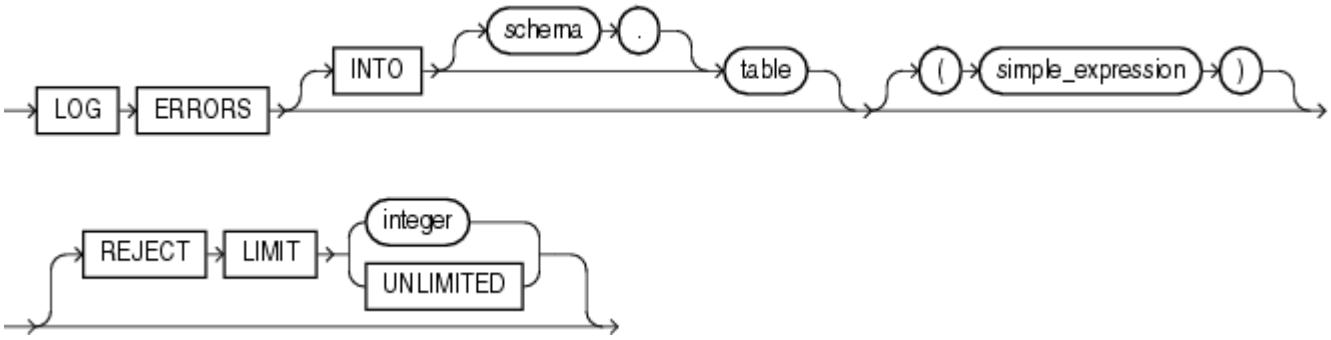


where_clause::=

returning_clause::=



error_logging_clause::=



セマンティクス

hint

文の実行計画を選択する場合に、オプティマイザに指示を与えるためのコメントを指定します。

UPDATEキーワードの直後にパラレル・ヒントを指定した場合、基礎となるスキャンおよびUPDATE操作の両方をパラレル化できます。

関連項目:

- ヒントの構文および説明は、「ヒント」を参照してください。
- パラレル実行の詳細は、『Oracle Database概要』を参照してください。

DML_table_expression_clause

ONLY句は、ビューのみに適用されます。UPDATE句のビューが階層に属し、そのどのサブビューの行も変更しない場合は、ONLY構文を指定します。

関連項目:

「DML_table_expression_clauseの制限事項」および「表の更新: 例」を参照してください。

schema

更新するオブジェクトが含まれているスキーマを指定します。schemaを指定しない場合、オブジェクトは自分のスキーマ内にあるとみなされます。

table | view | materialized_view |subquery

更新する対象となる、表、ビュー、マテリアライズド・ビュー、または副問合せから戻された列の名前を指定します。表に対してUPDATE文を実行した場合、その表に対応付けられたUPDATEトリガーが起動します。

- viewを指定した場合、ビューの実表が更新されます。ビューを定義する問合せに次のいずれかの要素が含まれる場合は、INSTEAD OFトリガーを除き、そのビューを更新することはできません。

- 集合演算子
- DISTINCT演算子
- 集計ファンクションまたは分析ファンクション
- GROUP BY、ORDER BY、MODEL、CONNECT BYまたはSTART WITH句
- SELECT構文のリストにあるコレクション式
- SELECT構文のリストにある副問合せ
- WITH READ ONLYが指定された副問合せ
- 再帰WITH句
- 結合(一部の例外を除く。詳細は、『Oracle Database管理者ガイド』を参照してください。)
-

- ビューから、複数の実表は更新できません。
- また、WITH CHECK OPTIONを指定してビューを作成した場合、実行結果がビューを定義する問合せの条件を満たす場合にのみ、ビューを更新できます。
- tableまたはviewの実表に、1つ以上のドメイン索引列がある場合は、この文によって適切な索引タイプの更新ルーチンが実行されます。
- 読取り専用のマテリアライズド・ビューの行は更新できません。書込み可能なマテリアライズド・ビューの行を更新すると、基礎となるコンテナ表の行も更新されます。ただし、その更新内容は次のリフレッシュ操作によって上書きされます。マテリアライズド・ビュー・グループ内の更新可能なマテリアライズド・ビューの行を更新すると、マスター表の対応する行も更新されます。

関連項目:

- 索引タイプの更新ルーチンの詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。
- 更新可能なマテリアライズド・ビューについては、「CREATE MATERIALIZED VIEW」を参照してください。

partition_extension_clause

更新対象のtable内にあるパーティションまたはサブパーティションの名前またはパーティション・キー値を指定します。パーティション表内の値を更新する場合は、パーティション名を指定する必要はありません。ただし、パーティション名を指定した方が、複雑なwhere_clauseを使用するよりも効果的な場合もあります。

関連項目:

「パーティション表と索引の参照」および「パーティションの更新: 例」を参照してください。

dblink

オブジェクトが格納されているリモート・データベースへのデータベース・リンクの完全名または部分名を指定します。Oracle Databaseの分散機能を使用している場合にかぎり、データベース・リンクを使用してリモート・オブジェクトを更新できます。

dblinkを省略した場合、オブジェクトがローカル・データベース上にあるとみなされます。

ノート:

Oracle Database 12c リリース 2 (12.2)以降では、UPDATE 文で、リモートの LOB ロケータをバインド変数として指定できます。詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』の「分散 LOB」の章を参照してください。

関連項目:

データベース・リンクの参照方法の詳細は、「リモート・データベース内のオブジェクトの参照」を参照してください。

subquery_restriction_clause

subquery_restriction_clauseを使用すると、次のいずれかの方法で副問合せを制限できます。

WITH READ ONLY

WITH READ ONLYを指定すると、表またはビューを更新禁止にできます。

WITH CHECK OPTION

WITH CHECK OPTIONを指定すると、副問合せに含まれない行を生成する表またはビューの変更を禁止できます。この句をDML文の副問合せ内で使用する場合、FROM句内の副問合せには指定できますが、WHERE句内の副問合せには指定できません。

CONSTRAINT constraint

CHECK OPTION制約の名前を指定します。この識別子を省略した場合は、Oracleによって自動的にSYS_Cnという形式の制約名が割り当てられます(nはデータベース内で制約名を一意にするための整数)。

関連項目:

WITH CHECK OPTION句の使用方法: 例

table_collection_expression

table_collection_expressionを使用すると、問合せおよびDML操作で、collection_expression値を表として扱うことができます。collection_expressionには、副問合せ、列、ファンクションまたはコレクション・コンストラクタのいずれかを指定できます。その形式にかかわらず、集合値(ネストした表型またはVARRAY型の値)を戻す必要があります。このようなコレクションの要素抽出プロセスをコレクション・ネスト解除といいます。

TABLEコレクション式を親表と結合する場合は、オプションのプラス(+)には大きな意味があります。+を指定すると、その2つの外部結合が作成され、コレクション式がNULLの場合でも、外部表の行が問合せで戻されるようになります。

ノート:

以前のリリースの Oracle では、collection_expression が副問合せの場合、table_collection_expression を THE subquery と表現していました。現在、このような表現方法は非推奨になっています。

table_collection_expressionを使用して、ある表の行を別の表の行を基にして更新できます。たとえば、四半期ごとの売上表を、年度ごとの売上表にまとめることができます。

t_alias

文中で参照する表、ビューまたは副問合せの相関名(別名)を指定します。DML_table_expression_clauseがいずれかのオブジェクト型属性またはオブジェクト型メソッドを参照する場合、この別名が必要です。

関連項目:

「相関更新: 例」

DML_table_expression_clauseの制限事項

この句には、次の制限事項があります。

- tableまたはviewの実表に、IN_PROGRESSまたはFAILEDとマークされたドメイン索引がある場合は、この文は実行できません。
- 関係する索引パーティションがUNUSABLEとマークされている場合は、パーティションに挿入できません。
- DML_table_expression_clauseの副問合せにはorder_by_clauseを指定できません。
- UNUSABLEのマークが付いている索引、索引パーティションまたは索引サブパーティションを指定する場合、SKIP_UNUSABLE_INDEXESセッション・パラメータがTRUEに設定されていないかぎり、UPDATE文は正常に実行されません。

関連項目:

SKIP_UNUSABLE_INDEXESセッション・パラメータの詳細は、「ALTER SESSION」を参照してください。

update_set_clause

update_set_clauseを使用すると、列の値を設定できます。

column

更新するオブジェクトの列の名前を指定します。update_set_clauseに表の列を指定しない場合、その列の値は変更されません。

columnがLOBオブジェクト属性を参照している場合、まず空またはNULLの値で初期化する必要があります。リテラルで更新はできません。また、UPDATE以外のSQL文を使用してLOB値を更新する場合は、LOBを含む行を最初にロックしておく必要があります。詳細は、「for_update_clause」を参照してください。

columnが仮想列である場合、ここで指定することはできません。この場合、仮想列の導出元となっている値を更新する必要があります。

columnがパーティション表のパーティション化キーに含まれる場合、別のパーティションまたはサブパーティションに行を移動する列の値を変更すると、行の移動を有効にしないかぎり、UPDATEは正常に実行されません。「CREATE TABLE」の「row_movement_clause」または「ALTER TABLE」を参照してください。

また、columnがリスト・パーティション表のパーティション化キーの一部である場合、パーティションのpartition_key_valueリストに存在していない列の値を指定すると、UPDATEは正常に実行されません。

subquery

更新される行ごとに1行ずつ戻す副問合せを指定します。

- update_set_clauseで1列のみを指定した場合、副問合せは1つの値のみを戻します。
- update_set_clauseで複数の列を指定した場合、副問合せは指定した列の数の値を戻します。
- 副問合せが行を戻さなかった場合は、列にはNULLが割り当てられます。
- subqueryがリモート・オブジェクトを参照する場合、参照がローカル・データベースのオブジェクトにループバックしないかぎり、UPDATEはパラレルで実行されます。ただし、DML_table_expression_clauseのsubqueryがリモート・オブジェクトを参照する場合は、UPDATEはシリアルで実行されます。

副問合せ内でflashback_query_clauseを使用すると、過去のデータでtableを更新できます。この句の詳細は、「SELECT」の「flashback_query_clause」 を参照してください。

関連項目:

- 「SELECT」および「副問合せの使用方法」を参照してください。
- 「CREATE TABLE」の「parallel_clause」を参照してください。

expr

対応する列に割り当てられた新しい値に変換する式を指定します。

ノート:

expr の構文は、「式」および「オブジェクト表の更新: 例」を参照してください。

DEFAULT

DEFAULTを指定すると、以前に列のデフォルト値として指定した値を列に設定できます。対応する列に対してデフォルト値を指定していない場合、列にNULLが設定されます。

デフォルト値に対する更新の制限事項

ビューを更新する場合は、DEFAULTを指定できません。

指定する表に対してOracle Label Securityポリシーが有効になっている場合、UPDATE文ではDEFAULT句を使用できません。

VALUE句

VALUE句を使用すると、オブジェクト表の行全体を指定できます。

VALUE句の制限事項

この句は、オブジェクト表にのみ指定できます。

ノート:

RAW 列に文字列リテラルを挿入する場合、後続の問合せ中に RAW 列にある索引は使用されずに、全表スキャンが行われます。

関連項目:

オブジェクト表の更新: 例

where_clause

where_clauseを使用すると、指定したconditionがtrueである行のみを更新するように制限できます。この句を指定しない場合、表またはビューのすべての行が更新されます。conditionの構文は、「条件」を参照してください。

where_clauseは、値を更新する行を決定します。where_clauseを指定しない場合、すべての行が更新されます。where_clauseの条件を満たす行ごとに、update_set_clauseの等号演算子(=)の左側にある列に、演算子の対応する右側の式の値が設定されます。式は行が更新される場合に評価されます。

returning_clause

この句を使用すると、DML文に影響される行を取り出すことができます。この句は、表、マテリアライズド・ビュー、および単一の実表を持つビューに指定できます。

returning_clauseを指定したDML文を単一行に実行すると、影響された行、ROWID、および処理された行へのREFを使用している列式が取り出され、ホスト変数またはPL/SQL変数に格納されます。

returning_clauseを指定したDML文を複数行に実行すると、式の値、ROWIDおよび処理された行に関連するREFがバインド配列に格納されます。

expr

exprリストの各項目は、適切な構文で表す必要があります。

INTO

INTO句を指定すると、変更された行の値を、data_itemリストに指定する変数に格納できます。

data_item

data_itemはそれぞれ、取り出したexpr値を格納するためのホスト変数またはPL/SQL変数です。

RETURNINGリストの各式については、INTOリストに、対応する型に互換性があるPL/SQL変数またはホスト変数を指定する必要があります。

制限事項

RETURNING句には、次の制限事項があります。

- exprに次の制限事項があります。
  - UPDATE文およびDELETE文の場合、各exprは、単純式または単一セットの集計ファンクション式である必要があります。1つのreturning_clause内に単純式と単一セットの集計ファンクション式を混在させることはできません。INSERT文の場合、各exprは単純式である必要があります。INSERT文のRETURNING句では、集計ファンクションはサポートされていません。
  - 単一セットの集計ファンクション式をDISTINCTキーワードに含めることはできません。
- exprのリストに主キー列または他のNOT NULL列が含まれている場合は、表にBEFORE UPDATEトリガーが定義されていると、UPDATE文の実行に失敗します。
- マルチテーブル・インサートではreturning_clauseを指定できません。
- パラレルDMLまたはリモート・オブジェクトにはこの句を使用できません。

- LONG型を取り出すことはできません。
- INSTEAD OFトリガーが定義されたビューに対して指定することはできません。

関連項目:

BULK COLLECT句を使用してコレクション変数に複数の値を戻す場合は、『Oracle Database PL/SQL言語リファレンス』を参照してください。

error_logging_clause

error_logging_clauseのUPDATE文での動作は、INSERT文の場合と同じです。詳細は、「INSERT」文の「error_logging_clause」を参照してください。

関連項目:

エラー・ロギングによる表への挿入：例

例

表の更新：例

次の文は、ジョブSH_CLERKを持つすべての従業員にNULLの歩合を付与します。

```
UPDATE employees
   SET commission_pct = NULL
   WHERE job_id = 'SH_CLERK';
```

次の文は、Douglas Grantを部門20の管理者に昇格させ、給与を$1,000引き上げます。

```
UPDATE employees SET
    job_id = 'SA_MAN', salary = salary + 1000, department_id = 120
    WHERE first_name||' '||last_name = 'Douglas Grant';
```

次の文は、remoteデータベースのemployees表の従業員の給与を増加します。

```
UPDATE employees@remote
   SET salary = salary*1.1
   WHERE last_name = 'Baer';
```

次の例は、UPDATE文の次の構文要素を示します。

- 単一文にまとめたupdate_set_clauseの2つの形式
- 相関副問合せ
- 更新された行を制限するwhere_clause

```
UPDATE employees a
    SET department_id =
        (SELECT department_id
            FROM departments
            WHERE location_id = '2100'),
        (salary, commission_pct) =
        (SELECT 1.1*AVG(salary), 1.5*AVG(commission_pct)
          FROM employees b
          WHERE a.department_id = b.department_id)
    WHERE department_id IN
        (SELECT department_id
          FROM departments
```

```
          WHERE location_id = 2900
              OR location_id = 2700);
```

このUPDATE文によって、次の処理が実行されます。

- ジュネーブまたはミュンヘン(location_idは2900または2700)で働く従業員のみを更新します。
- これらの従業員のdepartment_idをボンベイ(location_idは2100)の対応するdepartment_idに設定します。
- 各従業員の給与を、その部門の平均給与の110%に上げます。
- 各従業員の歩合を、その部門の平均歩合の150%に上げます。

パーティションの更新: 例

次の例では、sales表の1つのパーティションの値を更新します。

```
UPDATE sales PARTITION (sales_q1_1999) s
   SET s.promo_id = 494
   WHERE amount_sold > 1000;
```

オブジェクト表の更新: 例

次の文は、「表のコレクション: 例」で作成されたpeople_typオブジェクトのオブジェクト表として、people_demo1とpeople_demo2の2つを作成します。この例では、people_demo2から行を選択してpeople_demo1の行を更新する方法を示します。

```
CREATE TABLE people_demo1 OF people_typ;
CREATE TABLE people_demo2 OF people_typ;
UPDATE people_demo1 p SET VALUE(p) =
   (SELECT VALUE(q) FROM people_demo2 q
    WHERE p.department_id = q.department_id)
   WHERE p.department_id = 10;
```

この例では、SET句と副問合せの両方で、VALUEオブジェクト参照ファンクションを使用します。

相関更新: 例

相関副問合せを使用してネストした表の行を更新する例は、「表のコレクション: 例」を参照してください。

UPDATE操作中のRETURNING句の使用方法: 例

次の文は、更新された行の値を戻し、PL/SQL変数bnd1、bnd2、bnd3に結果を格納します。

```
UPDATE employees
  SET job_id ='SA_MAN', salary = salary + 1000, department_id = 140
  WHERE last_name = 'Jones'
  RETURNING salary*0.25, last_name, department_id
    INTO :bnd1, :bnd2, :bnd3;
```

次の文は、RETURNING句の式で単一セットの集計ファンクションを指定できることを示します。

```
UPDATE employees
   SET salary = salary * 1.1
   WHERE department_id = 100
   RETURNING SUM(salary) INTO :bnd1;
```

A 構文図の読み方

この付録では、構文図の読み方について説明します。

このマニュアルでは、OracleのSQL構文を図形とテキスト(バッカス正規形ーBNF)の両方で示しています。この付録の内容は次のとおりです。

- 図形構文図
- バッカス正規形構文

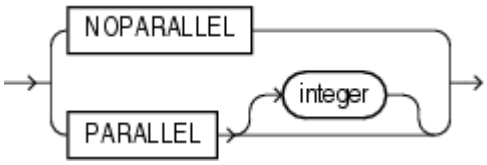
図形構文図

構文図とは、SQLの有効な構文を図で示したものです。構文図は、矢印が示す方向に左から右へ読んでください。

コマンドおよびキーワードは、四角形の中に大文字で書かれています。これらの文字は、四角形の中に表示されているとおり正確に入力してください。パラメータは、楕円形の中に小文字で書かれています。パラメータには変数が使用されます。句読点、デリミタ、終了記号は円の中に書かれています。

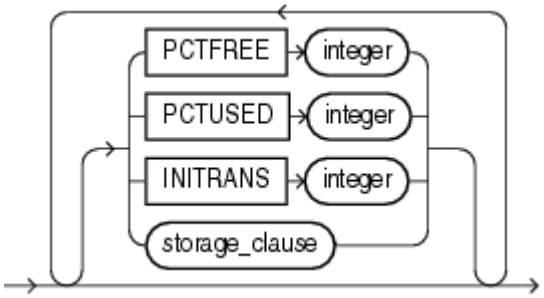
構文図の中にパスが複数ある場合は、ユーザーがパスを選択できます。たとえば、NOPARALLELまたはPARALLELのいずれかを指定できます:

parallel_clause::=



キーワード、演算子、パラメータに複数の選択肢がある場合は、選択できるオプションが縦に並べて書かれています。次の構文図では、スタックにある4つのパラメータから1つ以上を指定することができます。

physical_attributes_clause::=



構文図で使用されるパラメータと、各文でそのパラメータに代入する値の例を次の表に示します。

表A-1 構文のパラメータ

パラメータ	説明	例
`table`	パラメータによって指定された型のオブジェクト名で置き換える必要があります。オブジェクト型の一覧は、「スキーマ・オブジェクト」の項を参照してください。	`employees`
`c`	ご使用のデータベース文字セットの単一文字で置き換える必要があります。	`T` `s`
`'text'`	一重引用符で囲んだテキスト文字列で置き換える必要があります。`'text'`の構文は、「テキスト・リテラル」を参照してください。	`'Employee records'`

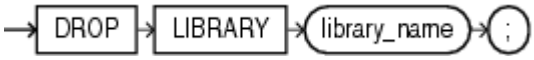
パラメータ	説明	例
char	CHAR または VARCHAR2 データ型の式か、一重引用符で囲んだ文字リテラルで置き換える必要があります。	last_name 'Smith'
condition	TRUE または FALSE に評価される条件で置き換える必要があります。condition の構文は、「条件」を参照してください。	last_name >'A'
date d	日付定数または DATE データ型の式で置き換える必要があります。	TO_DATE('01-Jan-2002', 'DD-MON-YYYY')
expr	「SQL 式」にある expr の構文の説明で定義されている任意のデータ型の式で置き換えることができます。	salary + 1000
integer	「整数リテラル」にある integer の構文の説明で定義されている整数で置き換える必要があります。	72
number m n	NUMBER データ型の式、または「数値リテラル」にある number の構文の説明で定義されている数値定数で置き換える必要があります。	AVG(salary) 15 * 7
raw	RAW データ型の式で置き換える必要があります。	HEXTORAW('7D')
subquery	SELECT 文で置き換える必要があります。この SELECT 文は、別の SQL 文中で使用されます。「SELECT」を参照してください。	SELECT last_name FROM employees
db_name	埋込み SQL プログラム内のデフォルト以外のデータベース名で置き換える必要があります。	sales_db
db_string	Oracle Net データベース接続のデータベース識別文字列で置き換える必要があります。詳細は、ご使用の Oracle Net プロトコルのユーザーズ・ガイドを参照してください。	—

必須のキーワードおよびパラメータ

必須キーワードおよびパラメータは、単独で示されているか、または選択肢のリストとして縦に並べて書かれています。必須キーワードおよびパラメータが1つの場合は、メイン・パス、つまり現在選択しているパスの線上に書かれています。次の例では、

library_nameが必須パラメータです。

drop_library::=

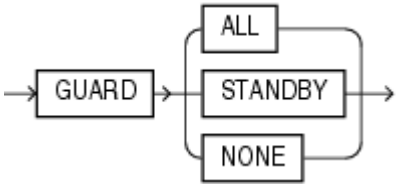


HQ_LIBという名前のライブラリがあるとすると、この図は、次の文を示しています。

```
DROP LIBRARY hq_lib;
```

メイン・パスに交わる縦に並んだリストの中に、複数のキーワードまたはパラメータがある場合、そのうちの1つが必須です。複数のキーワードまたはパラメータから、いずれか1つを選択する必要があります。ただし、メイン・パスに存在しなくてもかまいません。次の例では、ALL、STANDBY、またはNONEを選択する必要があります。

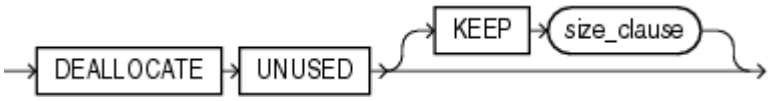
security_clause::=



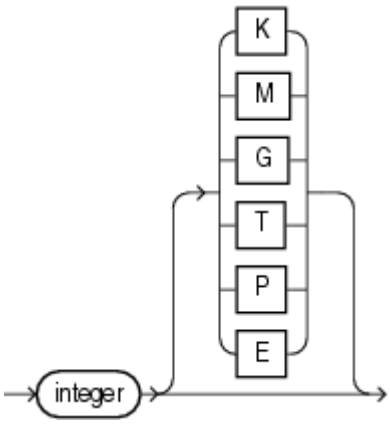
オプションのキーワードおよびパラメータ

キーワードおよびパラメータが、メイン・パスより上に縦に並べてリストされている場合は、それらのキーワードおよびパラメータがオプションであることを示しています。次の例では、上方向にたどらずに、メイン・パスを続けることができます。

deallocate_unused_clause::=



size_clause::=



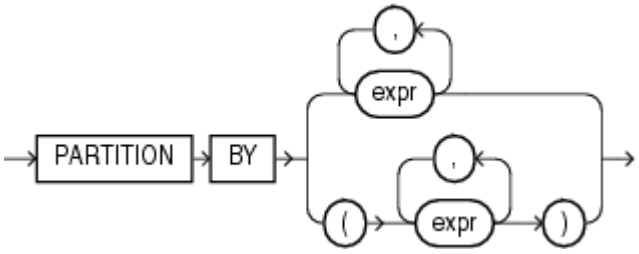
これらの図に従うと、次の文はすべて有効です。

```
DEALLOCATE UNUSED;
DEALLOCATE UNUSED KEEP 1000;
DEALLOCATE UNUSED KEEP 10G;
DEALLOCATE UNUSED 8T;
```

構文のループ

ループは、その中の構文を何回でも繰り返せることを示します。次の例では、値の式を1つ選択した後、繰り返し別の式を選択できます。式と式の間はカンマで区切ります。

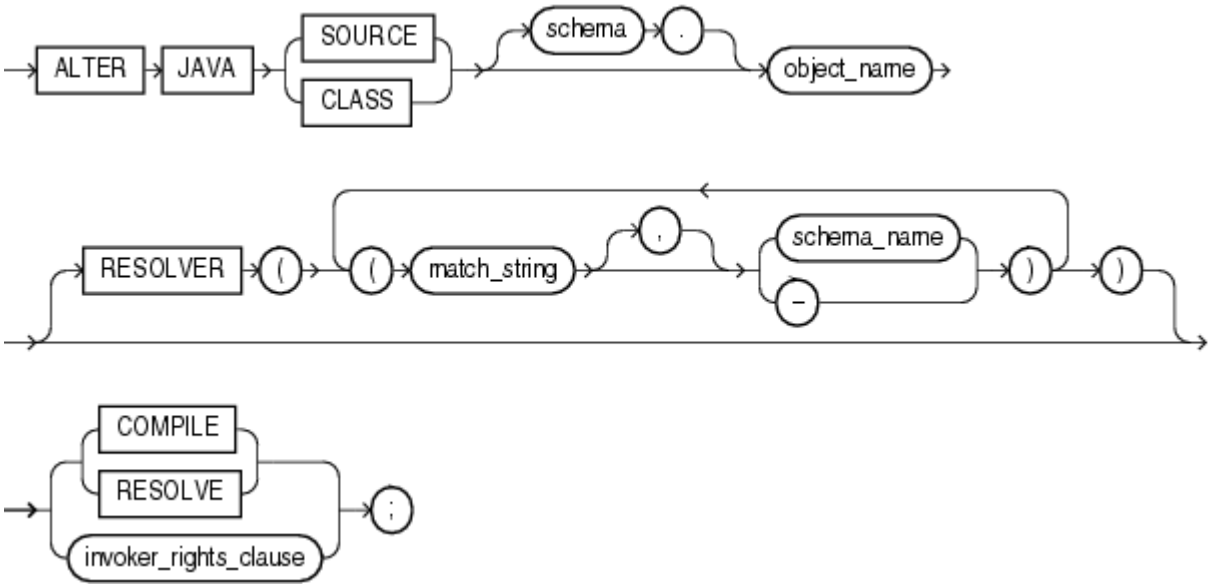
query_partition_clause::=



複数の部分に分割された構文図

複数の部分に分割された構文図は、すべてのメイン・パスの端と端が結合されているものとしてみてください。次の例は、3つに分割された構文図です。

alter_java::=



この図は、次の文が有効であることを示しています。

```
ALTER JAVA SOURCE jsource_1 COMPILE;
```

バッカス正規形構文

このマニュアルの各構文図の後には、テキストによる図の説明へのリンクが続きます。テキストによる説明は、次に示す記号や規則を含む、バッカス正規形(BNF)の単純な変形です。

記号または表記規則	意味
[ ]	大カッコは、カッコ内の項目を任意に選択することを表します。
{ }	中カッコは、カッコ内の項目のうち、1 つが必須であることを表します。
|	縦線は、大カッコまたは中カッコ内の選択項目を区切ります。
...	省略記号は、前述の構文要素を繰り返すことができることを示します。
デリミタ	大カッコ、中カッコ、縦線および省略記号以外のデリミタは、表示されているとおり入力する必要があります。
boldface	太字で示されている単語はキーワードです。キーワードは、表示されているとおり入力する必要があります。(オペレーティング・システムによっては、キーワードは大文字と小文字が区別されます。)太字以外の単語は、プレースホルダであり、名前または値を代入する必要があります。

B SQL操作時の自動ロックと手動ロックのメカニズム

この付録では、SQL文の実行時にデータを自動的にロックするメカニズムとユーザーの指定に従ってロックするメカニズムについて説明します。データの同時実行性および整合性という観点から説明したロック・メカニズムの概要は、『Oracle Database概要』を参照してください。

この付録の内容は次のとおりです。

- DML操作での自動ロック
- DDL操作での自動ロック
- 手動データ・ロック
- 非ブロッキングDDLのリスト

非ブロッキングDDLのリスト

11.2から12.2.0.2の各リリースで追加された非ブロッキングDDLを次にリストします。

リリース11.2

次の非ブロッキングDDLは、リリース11.2で追加されています。一部の非ブロッキングDDLは、サプリメンタル・ロギングが存在する場合にはブロッキングにダウングレードされます。

11.2で追加された非ブロッキングDDLのリスト

- create index online
- alter index rebuild online
- alter index rebuild partition online
- alter index rebuild subpartition online
- alter index visible / novisible

11.2で追加され、サプリメンタル・ロギング時にブロッキングにダウングレードされる非ブロッキングDDLのリスト

- alter table add column not null with default value
- alter table add constraint enable novalidate
- alter table modify constraint validate
- alter table add column (デフォルトなし)

リリース12.1

次の非ブロッキングDDLは、リリース12.1で追加されています。一部の非ブロッキングDDLは、サプリメンタル・ロギングが存在する場合にはブロッキングにダウングレードされます。

12.1で追加された非ブロッキングDDLのリスト

- drop index online
- alter index unusable online
- alter table move partition online

- alter table move subpartition online

12.1で追加され、サプリメンタル・ロギング時にブロッキングにダウングレードされる非ブロッキングDDLのリスト

- alter table set unused column online
- alter table drop constraint online
- alter table modify column visible / invisible
- alter table add nullable column with default value

リリース12.2.0.1

12.2.0.1で追加された非ブロッキングDDLのリスト

- alter table split partition [subpartition] online
- alter table move online (非パーティション表の移動)
- alter table modify partition by .. online (非パーティション表をパーティション状態に変換)

リリース12.2.0.2

12.2.0.2で追加された非ブロッキングDDLのリスト

- Alter table merge partition online
- alter table modify partition by .. online (表のパーティション化スキーマの変更)

DML操作での自動ロック

DMLロック(データ・ロックとも呼ぶ)の目的は、複数のユーザーが同時にアクセスするデータの整合性を保証することです。たとえば、DMLロックを使用すると、オンライン・ブックショップで在庫が残り1冊となった本を複数の顧客が購入してしまうことを回避できます。DMLロックは、同時に実行される矛盾する複数のDML操作またはDDL操作の破損を招く干渉を防ぎます。

DML文では、表レベルと行レベルの両方のロックが自動的に取得されます。次の各項で、それぞれのタイプのロックまたはロック・モードの後のカッコ内にある頭字語は、Oracle Enterprise Managerのロック・モニターで使用される略称です。Enterprise Managerでは、表ロックのモード(RS、SRXなど)が示されるかわりに、すべての表ロックについてTMと表示される場合があります。

次に、行ロックおよび表ロックのタイプの概要を示します。行ロックおよび表ロックのタイプの詳細は、『Oracle Database概要』を参照してください。

行ロック(TX)

行ロックは、TXロックとも呼ばれ、表の単一の行に対するロックです。INSERT、UPDATE、DELETE、MERGEおよびSELECT ... FOR UPDATE文のいずれかで変更される行ごとに、トランザクションは行ロックを取得します。行ロックは、トランザクションがコミットされるかロールバックされるまで保持されます。

ある行について行ロックを取得したトランザクションは、その行に対応する表についての表ロックも取得します。表ロックがあると、現行のトランザクションでのデータ変更をオーバーライドするDDL操作の競合が回避されます。

表ロック(TM)

INSERT、UPDATE、DELETE、MERGEおよびSELECT ... FOR UPDATE文で表が変更される場合、トランザクションは表ロック(TMロック)を自動的に取得します。これらのDML操作が表ロックを必要とするのは、トランザクションのために表へのDMLアクセスを確保すること、およびトランザクションと競合する可能性のあるDDL操作を防止するという2つの目的のためです。表ロックは、LOCK TABLE文(「手動データ・ロック」を参照)を使用して明示的に取得できます。

表ロックは、次のいずれかのモードで保持できます。

- 行共有ロック(RS)(半共有表ロック、SSとも呼ぶ): 表のロックを保持しているトランザクションがその表の行をロックしており、それらの行を更新する予定であることを示します。SSロックは最も制限の少ないモードの表ロックであり、表に対する同時実行性が最も高くなります。
- 行排他ロック(RX)(副排他表ロック、SXとも呼ぶ): ロックを保持しているトランザクションが表の行を更新したか、SELECT ... FOR UPDATEを発行したことを示します。SXロックでは、他のトランザクションが同じ表の行に対して問合せ、挿入、更新、削除またはロックを実行できます。このため、SXロックの場合は、同じ表に対して同時に行われているSXロックとSSロックを複数のトランザクションで取得できます。
- 共有表ロック(S): トランザクションが共有表ロックを保持している場合は、(SELECT ... FOR UPDATEを使用することなく)他のトランザクションによる同じ表への問合せが許可されます。ただし、表を更新できるのは、1つのトランザクションが共有表ロックを保持する場合のみです。共有表ロックは、複数のトランザクションで同時に保持されている場合があるため、このロックを保持していても、トランザクションで常に表を変更できるとはかぎりません。
- 共有行排他表ロック(SRX)(共有副排他表ロック、SSXとも呼ぶ): 共有表ロックよりも制限の厳しいロックです。特定の表に対して、同時に1つのトランザクションのみがSSXロックを取得できます。トランザクションがSSXロックを保持している場合、他のトランザクションは表への問合せが許可されます(SELECT ... FOR UPDATEの場合を除く)。ただし、表の更新は許可されません。
- 排他表ロック(X): 最も制限の厳しいモードの表ロックです。このロックを保持しているトランザクションには、表への排他

的な書込みアクセスが許可されます。表のXロックを取得できるのは、1つのトランザクションのみです。

関連項目:

「手動データ・ロック」

DML操作でのロック

Oracle Databaseでは、DML操作を実行すると行レベルと表レベルのロックが自動的に取得されます。ロック動作は操作の種類によって決まります。表B-1は、この項の情報をまとめたものです。

ノート:

表 B-1 の DML 文に示されている暗黙の SX ロックは、制約の影響により一時的に排他(X)ロックになることがあります。

表B-1 DML文で取得されるロックのまとめ

SQL文	行ロック	表ロック・モード	RS	RX	S	SRX	X
SELECT ... FROM table...	—	なし	可	可	可	可	可
INSERT INTO table ...	可	SX	可	可	不可	不可	不可
UPDATE table ...	可	SX	可脚注 1	可脚注 1	不可	不可	不可
MERGE INTO table ...	可	SX	可	可	不可	不可	不可
DELETE FROM table ...	可	SX	可脚注 1	可脚注 1	不可	不可	不可
SELECT ... FROM table FOR UPDATE OF ...	可	SX	可脚注 1	可脚注 1	不可	不可	不可
LOCK TABLE table IN ...	—						
ROW SHARE MODE		SS	可	可	可	可	不可
ROW EXCLUSIVE MODE		SX	可	可	不可	不可	不可

SQL文	行ロック	表ロック・モード	RS	RX	S	SRX	X
SHARE MODE		S	可	不可	可	不可	不可
SHARE ROW EXCLUSIVE MODE		SSX	可	不可	不可	不可	不可
EXCLUSIVE MODE		X	不可	不可	不可	不可	不可

脚注1

競合する行ロックが別のトランザクションによって保持されていない場合には、可。それ以外の場合は待機します。

行を問い合せるときのロック

問合せは、SELECT文の場合のように明示的に行われることも、ほとんどのINSERT、MERGE、UPDATEおよびDELETE文の場合のように暗黙的に行われることもあります。問合せコンポーネントを含まなくてもかまわない唯一のDML文は、VALUES句を伴うINSERT文です。問合せはデータを読み取るのみであるため、他のSQL文に対してほとんど干渉しないSQL文です。

次の特性は、FOR UPDATE句を伴わない問合せが該当します。

- 問合せはデータ・ロックを取得しません。そのため、特定の行に対して問合せが実行される場合も含め、問合せが実行されている表を、他のトランザクションが問い合せて更新できます。FOR UPDATE句を伴わない問合せでは、他の操作をブロックするデータ・ロックが取得されないため、このような問合せを非ブロック問合せと呼ぶことがあります。
- 問合せでは、データ・ロックが解除されるまで待機する必要はありません。そのため、問合せはいつでも実行できます。このルールの例外としては、分散トランザクションが保留中であるなど、きわめて特殊な場合にデータ・ロックの待機が必要になることがあります。

行を変更するときのロック

一部のデータベースでは、ロック・マネージャを使用してメモリー内のロック・リストをメンテナンスします。一方、Oracle Databaseでは、ロックされている行を含むデータ・ブロックにロック情報を格納します。各行ロックが影響するのは1つの行のみです。

Oracle Databaseでは、キューイング・メカニズムを使用して行ロックを取得します。行ロックを必要とするトランザクションは、まだ行がロックされていなければ、行のデータ・ブロックにロックを取得します。トランザクション自体は、ブロック・ヘッダーの対象トランザクション・リスト(ITL)・セクションにエントリが記録されます。このトランザクションで変更されるそれぞれの行は、ITLに格納されたトランザクションIDのコピーをポイントします。したがって、同じブロックに含まれる100行を1つのトランザクションで変更する場合は100個の行ロックが必要ですが、100行すべてが1つのトランザクションIDを参照します。

トランザクションが終了すると、トランザクションIDはデータ・ブロック・ヘッダーのITLセクションにそのまま残ります。新しいトランザクションで行の変更が必要になると、ロックがアクティブかどうかの判断は、そのトランザクションIDを使用して行われます。ロックがアクティブな場合、新しいトランザクションのセッションは、ロックが解除されたときの通知を要求します。アクティブでない場合は、新しいトランザクションがロックを取得します。

INSERT、UPDATE、DELETEおよびSELECT ... FOR UPDATE文の特性は次のとおりです。

20. DML文を含むトランザクションは、その文の変更対象の行に対して行排他ロックを取得します。そのため、行をロックしているトランザクションがコミットまたはロールバックされるまで、他のトランザクションは、ロックされている行を更新し

たり削除することはできません。

21. これらの行ロックに加え、データを変更するDML文を含むトランザクションは、変更対象の行を含む表に対して副排他表ロック(SX)以上のロックを取得することも必要です。その表に対してトランザクションが、S、SRXまたはX表ロック(SXロックよりも制限が厳しいロック)をすでに保持している場合、SXロックは不要であるため取得しません。ただし、このトランザクションがSSロックのみをすでに保持している場合、Oracle DatabaseはSSロックをSXロックに自動的に変換します。

22. DML文を含むトランザクションは、副問合せや暗黙的な問合せによって選択される行に対して行ロックを取得する必要はありません。

    次のUPDATE文の例では、カッコ内のSELECT文が副問合せであり、WHERE a > 5句が暗黙的な問合せです。

    ```
    UPDATE t SET x = ( SELECT y FROM t2 WHERE t2.z = t.z ) WHERE a > 5;
    ```

    DML文内にある副問合せや暗黙的な問合せは、問合せの開始時点での一貫性が保証されており、その問合せで構成されているDML文による影響を参照しません。

23. トランザクション内の問合せは、同じトランザクション内の前のDML文で行われた変更を参照できますが、他のトランザクションのコミットされていない変更は参照できません。

関連項目:

外部キーのロックの詳細は、『Oracle Database概要』を参照してください。

DDL操作での自動ロック

実行中のDDL操作でスキーマ・オブジェクトが影響を受けたり参照される間、そのオブジェクトの定義はデータ・ディクショナリ(DDL)・ロックで保護されます。たとえば、ユーザーがプロシージャを作成すると、そのプロシージャ定義で参照されるすべてのスキーマ・オブジェクトのDDLロックが自動的に取得されます。DDLロックにより、プロシージャのコンパイル完了前にそれらのオブジェクトの変更または削除が防止されます。

Oracle Databaseは、DDLロックを必要とするDDLトランザクションのために、DDLロックを自動的に取得します。ユーザーはDDLロックを明示的に要求できません。DDL操作中には、修正や参照の対象となる個々のスキーマ・オブジェクトのみがロックされます。データ・ディクショナリ全体がロックされることはありません。

DDL操作では、変更されるスキーマ・オブジェクトに対するDMLロックも取得されます。

排他DDLロック

排他DDLロックは、他のセッションにDDLロックまたはDMLロックが取得されないようにします。

ほとんどのDDL操作では、同じスキーマ・オブジェクトを変更または参照する可能性のある他のDDL操作によって破壊的な干渉が起きないように、排他DDLロックが必要です。たとえば、ALTER TABLE操作で表に列を追加している間は、DROP TABLE操作で表を削除できません。その逆も同様です。ただし、表に対する問合せはブロックされません。

排他ロックは、DDL文実行の継続中と自動コミット中ずっと存続します。排他DDLロックの取得では、別の操作によってすでにスキーマ・オブジェクトに対する別のDDLロックが保持されている場合、その取得は古いDDLロックが解除されるまで待機し、その後に処理されます。

共有DDLロック

リソースの共有DDLロックを使用すると、競合するDDL操作による破壊的な干渉を防ぐことができ、かつ一方では類似のDDL操作についてデータの同時実行性が許可されます。

たとえば、CREATE PROCEDURE文の実行時には、その文を含むトランザクションは、参照されるすべての表について共有DDLロックを取得します。他のトランザクションは同じ表を参照するプロシージャを同時に作成できるため、同じ表について同時実行の共有DDLロックの取得はできますが、参照中の表について排他DDLロックを取得できるトランザクションはありません。

共有DDLロックは、DDL文実行の継続中と自動コミット中ずっと存続します。そのため、共有DDLロックを保持するトランザクションには、トランザクションの継続中は参照しているスキーマ・オブジェクトの定義が変更されないことが保証されます。

ブレーク可能解析ロック

SQL文またはPL/SQLプログラム・ユニットは、参照する各スキーマ・オブジェクトについて解析ロックを保持します。解析ロックを取得する目的は、参照しているオブジェクトが変更または削除された場合に、対応する共有SQL領域を無効にできるようにするためです。解析ロックは、どのようなDDL操作も拒否せず、矛盾するDDL操作も許可するためにブレーク(中断)できるため、ブレーク可能解析ロックと呼ばれます。

解析ロックは、SQL文の実行解析フェーズの間に、共有プール内で取得されます。解析ロックは、その文の共有SQL領域が共有プールに残っているかぎり保持されます。

手動データ・ロック

データの同時実行性、データの整合性および文レベルの読取り一貫性を確保するために、Oracle Databaseは常に自動的にロックを実行します。ただし、このデフォルトのロック・メカニズムは上書きすることができます。デフォルト・ロックの上書きは、次のような状況で有効です。

- アプリケーションで、他のトランザクションによる変更を反映せず、トランザクションの継続時間にわたってデータの一貫性を維持する必要がある場合。トランザクション・レベルの読取り一貫性は、明示的なロック、読取り専用トランザクション、シリアライズ可能トランザクションを使用するか、デフォルトのロックを上書きすることで実現できます。
- アプリケーションで、あるトランザクションが他のトランザクションの完了まで待機せずに済むように、そのトランザクションがリソースに排他的アクセスできるようにする必要がある場合。トランザクションが継続している間、明示的にデータをロックできます。

自動ロックは、次の2つのレベルで上書きできます。

- トランザクション。トランザクション・レベルのロックは、次のSQL文で上書きできます。
  - `SET TRANSACTION ISOLATION LEVEL`
  - `LOCK TABLE`
  - `SELECT ... FOR UPDATE`

  これらの文で取得されたロックは、トランザクションがコミットまたはロールバックされた後で解除されます。
- セッション。必要なトランザクションの分離レベルは、`ALTER SESSION SET ISOLATION LEVEL`文を使用して設定できます。

ノート:

Oracle のデフォルト・ロックを上書きする場合、データベース管理者やアプリケーション開発者は、データの整合性が保証される、データの同時実行性が許容範囲内である、デッドロックは発生する可能性がないかまたは発生しても適切に処理される、などの条件を満たす必要があります。これらの条件の詳細は、『Oracle Database 概要』を参照してください。

C Oracleと標準SQL

この付録では、ANSI(米国規格協会)およびISO(国際標準化機構)によって確立されたSQL規格へのOracleの規格準拠について説明します。

ISOのSQL規格は、9つのパート(SQL/Framework、SQL/Foundation、SQL/CLI、SQL/PSM、SQL/MED、SQL/OLB、SQL/Schemata、SQL/JRTおよびSQL/XML)で構成されています。ANSI SQL標準は、同じ9つのパートで構成されています。

SQLの必須部分は、Core SQLとして知られ、SQL:2016のPart 2「Foundation」およびPart 11「Schemata」に記載されています。基礎的な機能は、Part 2のAnnex Fの「Feature taxonomy and definition for mandatory features」表で分析されています。スキーマ機能は、Part 11のAnnex Fの「Feature taxonomy and definition for mandatory features」表で分析されています。

この付録の内容は次のとおりです。

ANSI規格

SQLに関連するANSIのドキュメントは、次のとおりです。

- INCITS/ANSI/ISO/IEC 9075-1:2016, Information technology—Database languages—SQL—Part 1: Framework (SQL/Framework)
- INCITS/ANSI/ISO/IEC 9075-2:2016, Information technology—Database languages—SQL—Part 2: Foundation (SQL/Foundation)
- INCITS/ANSI/ISO/IEC 9075-3:2016, Information technology—Database languages—SQL—Part 3: Call-Level Interface (SQL/CLI)
- INCITS/ANSI/ISO/IEC 9075-4:2016, Information technology—Database languages—SQL—Part 4: Persistent Stored Modules (SQL/PSM)
- INCITS/ANSI/ISO/IEC 9075-9:2016, Information technology—Database languages—SQL—Part 9: Management of External Data (SQL/MED)
- INCITS/ANSI/ISO/IEC 9075-10:2016, Information technology—Database languages—SQL—Part 10: Object Language Bindings (SQL/OLB)
- INCITS/ANSI/ISO/IEC 9075-11:2016, Information technology—Database languages—SQL—Part 11: Information and Definition Schemas (SQL/Schemata)
- INCITS/ANSI/ISO/IEC 9075-13:2016, Information technology—Database languages—SQL—Part 13: SQL Routines and Types using the Java Programming Language (SQL/JRT)
- INCITS/ANSI/ISO/IEC 9075-14:2016, Information technology—Database languages—SQL—Part 14: XML-Related Specifications (SQL/XML)

これらの規格は、次の項に示す対応するISO規格と同一です。

ANSI規格のコピーについては、次の宛先に申し込んでください。

9. American National Standards Institute
10. 25 West 43rd Street, fourth floor
11. New York, NY 10036 USA
12. 電話番号: +1.212.642.4900
13. FAX番号: +1.212.398.0023
14. URL: http://www.ansi.org/

規格のコピーは、次のWebサイトからも入手できます。

```
http://webstore.ansi.org/default.aspx
```

SQL規格を含むANSI規格のサブセットは、INCITS規格です。これらは、INCITS(InterNational Committee for Information Technology Standards)から入手できます。URLは、次のとおりです。

```
http://www.incits.org/
```

ISO規格

SQLに関連するISOのドキュメントは、次のとおりです。

- ISO/IEC 9075-1:2016, Information technology--Database languages--SQL--Part 1: Framework (SQL/Framework)
- ISO/IEC 9075-2:2016, Information technology--Database languages--SQL--Part 2: Foundation (SQL/Foundation)
- ISO/IEC 9075-3:2016, Information technology--Database languages--SQL--Part 3: Call-Level Interface (SQL/CLI)
- ISO/IEC 9075-4:2016, Information technology--Database languages--SQL--Part 4: Persistent Stored Modules (SQL/PSM)
- ISO/IEC 9075-9:2016, Information technology--Database languages--SQL--Part 9: Management of External Data (SQL/MED)
- ISO/IEC 9075-10:2016, Information technology--Database languages--SQL--Part 10: Object Language Bindings (SQL/OLB)
- ISO/IEC 9075-11:2016, Information technology--Database languages--SQL--Part 11: Information and Definition Schemas (SQL/Schemata)
- ISO/IEC 9075-13:2016, Information technology--Database languages--SQL--Part 13: SQL Routines and Types using the Java Programming Language (SQL/JRT)
- ISO/IEC 9075-14:2016, Information technology--Database languages--SQL--Part 14: XML-Related Specifications (SQL/XML)

ISO規格のコピーについては、次の宛先に申し込んでください。

- International Organization for Standardization
- 1, ch. de la Voie-Creuse
- Case postale 56
- CH-1211, Geneva 20, Switzerland
- 電話番号: +41.22.749.0111
- FAX番号: +41.22.733.3430
- URL: http://www.iso.org/

Webストアからも入手できます。URLは、次のとおりです。

```
http://www.iso.org/iso/store.htm
```

Core SQLに対するOracleの準拠

ANSIおよびISOのSQL規格では、規格準拠性について明示する箇所に、準拠の種類および実装されている機能について記載することを義務付けています。規格準拠性についての最小限の明示をCore SQLといいます。これは、規格のPart 2「SQL/Foundation」およびPart 11「SQL/Schemata」に定義されています。次の製品は、後続の表に示すCore SQLに完全(または部分的)に準拠しています。

- Oracle Databaseサーバー、リリース12.2
- OTT (Oracle Type Translator)、リリース12.2
- Pro*C/C++リリース12.2
- Pro*COBOLリリース12.2

SQL規格準拠機能は、移植性の指針または機能性の指針として使用できます。移植性の観点からは、ユーザーは、規格の機能の正確な構文とセマンティクスの両方の準拠性に関心があります。機能性の観点からは、ユーザーは正確な構文よりもセマンティクスの問題に関心を持ちます。この付録の表では、規格の構文およびセマンティクスのサポートに関して次の用語を使用します。

- 完全なサポート: 機能は規格の構文およびセマンティクスによってサポートされます。
- 部分的なサポート: 規格の構文の一部がサポートされますが、完全にはサポートされません。サポートされる場合は規格のセマンティクスを持ちます。
- 拡張サポート: 規格のセマンティクスと、追加の機能がサポートされます。
- 同等のサポート: 規格とは異なる構文を使用して規格のセマンティクスがサポートされます。
- 類似のサポート: 規格の構文もセマンティクスも正確にはサポートされませんが、類似機能が提供されます。

表C-1に、Core SQLの機能に対するOracleのサポートを示します。

表C-1 Core SQLの機能に対するOracleのサポート

機能識別子、機能	サポート
E011、数値データ型	この機能を完全にサポートします。
E021、文字データ型	次の副機能を完全にサポートします。 12. E021-01、CHARACTER データ型 13. E021-07、文字の連結 14. E021-08、UPPER および LOWER ファンクション 15. E021-09、TRIM ファンクション 16. E021-10、文字データ型間の暗黙的な加算 次の副機能を部分的にサポートします。

機能識別子、機能	サポート
	● E021-02、CHARACTER VARYING データ型(Oracle は、長さ 0 の VARCHAR 文字列と NULL を区別しません) ● E021-03、文字リテラル(Oracle は、長さ 0 のリテラル''を NULL とみなします) ● E021-12、文字の比較(Oracle の規則は、比較する 2 つの文字列の短い方に埋込みを行うという点で規格とは異なります) Oracle は、次の副機能と同等の機能を持ちます。 ● E021-04、CHARACTER_LENGTH ファンクションのかわりに、LENGTH ファンクションを使用します。 ● E021-05、OCTET_LENGTH ファンクションのかわりに、LENGTHB ファンクションを使用します。 ● E021-06、SUBSTRING ファンクションのかわりに、SUBSTR ファンクションを使用します。 ● E021-11、POSITION ファンクションのかわりに、INSTR ファンクションを使用します。
E031、識別子	Oracle はこの機能をサポートしますが、次の例外があります。 ● 引用識別子内で二重引用符を使用するためのエスケープ・シーケンスはサポートされません。 ● 非引用識別子は Oracle の予約語とは異なる場合があります(Oracle の予約語のリストは規格のリストとは異なります)。 ● 引用識別子の場合でも、ROWID という列名は許可されていません。 この機能は次のように拡張されます。 ● 識別子は最大 128 文字まで指定可能です。 ● 非引用識別子にはドル記号($)またはポンド記号(#)を指定可能です。
E051、基本的な問合せ指定	次の副機能を完全にサポートします。 ● E051-01、SELECT DISTINCT ● E051-02、GROUP BY 句 ● E051-04、SELECT 構文のリストにない列を GROUP BY に含めることができます。

機能識別子、機能	サポート
	● E051-05、SELECT 構文のリスト項目の名前の変更 ● E051-06、HAVING 句 ● E051-07、SELECT 構文のリストでの*の修飾 次の副機能を部分的にサポートします。 ● E051-08、FROM 句の相関名(Oracle は、相関名をサポートしますが、オプションの AS キーワードはサポートしません) Oracle は、次の副機能と同等の機能を持ちます。 ● E051-09、FROM 句での列の名前の変更(FROM 句の副問合せで列名を変更できます)
E061、基本的な述語および検索条件	Oracle はこの機能を完全にサポートしますが、Oracle の文字列比較は規格とは異なります。規格では、長さが異なる 2 つの文字列を比較する場合、空白または実際のすべての文字よりも小さい仮想の文字が短い文字列に埋め込まれます。埋込みの判断は文字セットに基づいて行われます。Oracle では、この判断は、比較対象が固定長または可変長のどちらかによって決まります。
E071、基本的な問合せ式	次の副機能を完全にサポートします。 ● E071-01、UNION DISTINCT 表演算子 ● E071-02、UNION ALL 表演算子 ● E071-05、表演算子によって結合された列の型は同じでなくてもかまいません。 ● E071-06、副問合せの表演算子 Oracle は、次の副機能と同等の機能を持ちます。 ● E071-03、EXCEPT DISTINCT 表演算子(EXCEPT DISTINCT のかわりに MINUS を使用します)
E081、基本的な権限	Oracle は、E081-09、USAGE 権限を除いて、この機能のすべての副機能を完全にサポートします。規格では、USAGE 権限を持つユーザーは、ドメイン、照合、文字セット、文字変換、ユーザー定義型および順序ジェネレータを使用できます。Oracle は、ドメインまたは文字変換をサポートしません。照合および文字セットにアクセスするには権限は必要ありません。ユーザー定義型を使用するための Oracle の権限は、EXECUTE です。順序型を使用するための Oracle の権限は、SELECT です。

機能識別子、機能	サポート
E091、集合ファンクション	この機能を完全にサポートします。
E101、基本的なデータ操作	この機能を完全にサポートします。
E111、単一行の SELECT 文	この機能を完全にサポートします。
E121、基本的なカーソルのサポート	次の副機能を完全にサポートします。 ● E121-02、ORDER BY 列は、SELECT 構文のリストになくてもかまいません。 ● E121-03、ORDER BY 句の値の式 ● E121-04、OPEN 文 ● E121-06、位置付けされた UPDATE 文 ● E121-07、位置付けされた DELETE 文 ● E121-08、CLOSE 文 Oracle は、次の副機能の部分的なサポートを提供します。 ● E121-01、DECLARE CURSOR - FOR READ ONLY 構文の場合を除き、完全にサポートされます。 ● E121-10、FETCH 文、暗黙的な NEXT - ノイズ・ワード FROM の場合を除き、完全にサポートされます。 Oracle は、次の副機能の拡張サポートを提供します。 ● E121-17、WITH HOLD カーソル(規格では、カーソルは ROLLBACK 中は保持されませんが、Oracle では ROLLBACK 中も保持されます。)
E131、NULL 値のサポート	Oracle はこの機能を完全にサポートしますが、Oracle では文字列型の NULL を長さが 0(ゼロ)の文字列と区別できないという例外があります。
E141、基本的な整合性制約	この機能を完全にサポートします。

機能識別子、機能	サポート
E151、トランザクションのサポート	この機能を完全にサポートします。
E152、基本的な SET TRANSACTION 文	この機能を完全にサポートします。
E153、副問合せを持つ更新可能な問合せ	この機能を完全にサポートします。
E161、先頭に 2 つの負の符号(-)を付けた SQL 文のコメント	この機能を完全にサポートします。
E171、SQLSTATE のサポート	この機能を完全にサポートします。
E182、ホストの言語バインディング	Oracle は、Pro*C/C++および Pro*COBOL を介してこの機能を完全にサポートします。
F021、基本的な情報スキーマ	Oracle はこの機能のビューを持ちません。ただし、同じ情報を別のメタデータ・ビューで使用可能にします。 ● TABLES のかわりに、ALL_TABLES を使用します。 ● COLUMNS のかわりに、ALL_TAB_COLUMNS を使用します。 ● VIEWS のかわりに、ALL_VIEWS を使用します。 ただし、ユーザーのビューに WITH CHECK OPTION が定義されていたり、更新可能な場合は、Oracle の ALL_VIEWS は表示されません。ビューに WITH CHECK OPTION が定義されているかどうかを確認する場合は、ALL_CONSTRAINTS を使用し、TABLE_NAME をそのビュー名にして、CONSTRAINT_TYPE が'V'であるビューを検索します。 ● TABLE_CONSTRAINTS、REFERENTIAL_CONSTRAINTS および CHECK_CONSTRAINTS のかわりに、ALL_CONSTRAINTS を使用します。 ただし、Oracle の ALL_CONSTRAINTS は、制約が遅延可能か初期遅延かは表示しません。

機能識別子、機能	サポート
F031、基本的なスキーマ操作	次の副機能を完全にサポートします。 ● F031-01、実表を作成するための CREATE TABLE 文 ● F031-02、CREATE VIEW 文 ● F031-03、GRANT 文 Oracle は、次の副機能に対して同等のサポートを提供します。 ● F031-04、ALTER TABLE 文の ADD COLUMN 句(Oracle は、この構文のオプションのキーワード COLUMN をサポートしません。また、規格とは異なりますが、Oracle では列定義をカッコで囲む必要があります。) 次の副機能はサポートしません(Oracle は、キーワード RESTRICT をサポートしません)。 ● F031-13、DROP TABLE 文の RESTRICT 句 ● F031-16、DROP VIEW 文の RESTRICT 句 ● F031-19、REVOKE 文の RESTRICT 句 (Oracle の DROP コマンドは、すべての依存するオブジェクトを連鎖的に削除したり依存するオブジェクトがある場合に削除を禁止するかわりに、依存するオブジェクトを無効にすることで、ユーザー操作なしに後で再検証できるように規格を拡張しています。)
F041、基本的な結合表	この機能を完全にサポートします。
F051、基本的な日付および時刻	次の副機能を除き、この機能を完全にサポートします。 ● F051-02、TIME データ型 ● F051-07、LOCALTIME
F081、ビューの UNION および EXCEPT	Oracle はビューの UNION を完全にサポートします。規格の EXCEPT と同等な Oracle の機能は MINUS と呼ばれ、ビューで完全にサポートされます。
F131、グループ操作	この機能を完全にサポートします。
F181、複数モジュールのサポート	この機能を完全にサポートします。

機能識別子、機能	サポート
F201、CAST ファンクション	この機能を完全にサポートします。
F221、明示的なデフォルト	Oracle の列定義内での DEFAULT ON NULL 機能は、INSERT 文に同等の機能を提供しますが、UPDATE 文には同等の機能を提供しません。
F261、CASE 式	この機能を完全にサポートします。
F311、スキーマ定義文	この機能を完全にサポートします。
F471、スカラー副問合せの値	この機能を完全にサポートします。
F481、拡張 NULL 述語	この機能を完全にサポートします。
F501、機能および準拠するビュー	Oracle は、この機能をサポートしません。
F812、基本的なフラグ付け	Oracle は、フラガーを持っていますが、SQL:2011 よりも SQL-92 に準拠しています。
S011、固有型	固有型は、強い型指定のスカラー型です。固有型は、1 つの属性のみを持つオブジェクト型を使用して、Oracle でエミュレートできます。USER_DEFINED_TYPES と呼ばれる規格の情報スキーマ・ビューは、Oracle のメタデータ・ビューALL_TYPES と同じです。
T321、基本的な SQL 起動ルーチン	次の副機能を完全にサポートします。 ● T321-03、ファンクションの起動 ● T321-04、CALL 文 Oracle は、次の副機能を異なる構文でサポートします。 ● T321-01、オーバーロードしないユーザー定義ファンクション ● T321-02、オーバーロードしないユーザー定義プロシージャ Oracle の構文 CREATE FUNCTION および CREATE PROCEDURE と規格の構文の違いは、次のとおりです。

機能識別子、機能	サポート
	● 規格の構文では、パラメータのモード(IN、OUT または INOUT)はパラメータ名の前ですが、Oracle 構文では、パラメータ名の後です。 ● 規格の構文では、INOUT を使用しますが、Oracle の構文では、IN OUT を使用します。 ● Oracle の構文では、復帰型の後およびルーチン本体の定義の前に IS または AS が必要ですが、規格の構文では、必要ありません。 ● ルーチンの本体が C である場合、そのルーチンに名前を付けるとき、規格の構文では、キーワード LANGUAGE C EXTERNAL NAME を使用しますが、Oracle の構文では、LANGUAGE C NAME を使用します。 ● ルーチンの本体が SQL である場合、Oracle では、プロシージャを独自に拡張した PL/SQL を使用します。 Oracle は、次の副機能を PL/SQL ではサポートしますが、Oracle SQL ではサポートしません。 ● T321-05、RETURN 文 Oracle は、次の副機能と同等の機能を提供します。 ● T321-06、ROUTINES ビュー: ALL PROCEDURES メタデータ・ビューを使用します。 ● T321-07、PARAMETERS ビュー: ALL_ARGUMENTS メタデータ・ビューおよび ALL_METHOD_PARAMS メタデータ・ビューを使用します。
T631、1 つのリスト要素を使用した IN 述語	この機能を完全にサポートします。

SQL/Foundationのオプション機能に対するOracleのサポート

表C-2に、SQL/Foundationのオプション機能に対するOracleのサポートを示します。

表C-2 SQL/Foundationのオプション機能に対するOracleのサポート

機能識別子、機能	サポート
B012、埋込み C	この機能を完全にサポートします。
B013、埋込み COBOL	この機能を完全にサポートします。
B021、ダイレクト SQL	この機能を完全にサポートします(SQL*Plus)。
B031、基本的な動的 SQL	Oracle は、2 種類の動的 SQL をサポートします。埋込み言語のマニュアルの Oracle 動的 SQL および ANSI 動的 SQL に関する説明を参照してください。 ANSI 動的 SQL は規格の実装であり、次の制限事項があります。 ● Oracle は、記述子の項目のサブセットをサポートします。 ● <句による入力>の場合、Oracle は<入力記述子の使用>のみをサポートします。 ● <句による出力>の場合、Oracle は<記述子への出力>のみをサポートします。 ● 動的パラメータは、コロンの後に疑問符ではなく識別子を指定する形式で示されます。 Oracle 動的 SQL は規格の動的 SQL に似ていますが、次の変更が加えられています。 ● パラメータは、コロンの後に疑問符ではなく識別子を指定する形式で示されます。 ● 規格の DESCRIBE OUTPUT は、Oracle の DESCRIBE SELECT LIST FOR 文に置き換えられます。 ● Oracle では、動的 SQL 文を準備する PREPARE 文の前に、その動的 SQL 文を使用して物理的にカーソルを宣言する場合、DECLARE STATEMENT を使用できます。

機能識別子、機能	サポート
B032、拡張動的 SQL	ANSI 動的 SQL では、Oracle はこの機能からグローバル文およびグローバル・カーソルを宣言する機能のみを実装します。その他の機能はサポートしません。 Oracle 動的 SQL では、Oracle の DESCRIBE BIND VARIABLES は、規格の DESCRIBE INPUT と同等です。その他の機能はサポートしません。
B122、ルーチン言語 C	Oracle は、C で記述された外部ルーチンをサポートしますが、このようなルーチンを作成する規格構文はサポートしません。
B128、ルーチン言語 SQL	Oracle は PL/SQL で記述されたルーチンをサポートします。PL/SQL は規格の手続き型言語 SQL/PSM と同等な Oracle の機能です。
F032、CASCADE 削除動作	Oracle では、DROP コマンドによって削除されたオブジェクトのすべての依存オブジェクトが無効になります。無効になったオブジェクトを正常に再コンパイルできる方法を使用して、削除されたオブジェクトを再定義するまで、無効になったオブジェクトは完全に使用禁止になります。
F033、ALTER TABLE 文の DROP COLUMN 句	Oracle は DROP COLUMN 句を提供しますが、規格で使用される RESTRICT または CASCADE オプションは提供しません。
F034、拡張 REVOKE 文	Oracle は、この機能の次の部分をサポートします。 ● F034-01、スキーマ・オブジェクトの所有者以外のユーザーによって実行される REVOKE 文 ● F034-03、権限受領者が WITH GRANT OPTION を持つ権限を取り消す REVOKE 文 Oracle は、この機能のうち、次の部分と同等の機能を提供します。 ● CASCADE: Oracle では、REVOKE によってすべての依存オブジェクトが無効になります。これにより、この後に CREATE コマンドと GRANT コマンドによって、無効になったオブジェクトが正常に再コンパイルできるようになり、メタデータが変更されるまでは、これらのオブジェクトは完全に使用禁止になります。
F052、期間および日時の算術	Oracle は、INTERVAL YEAR TO MONTH および INTERVAL DAY TO SECOND データ型のみをサポートします。

機能識別子、機能	サポート
F111、SERIALIZABLE 以外の分離レベル	Oracle は、SERIALIZABLE に加えて、READ COMMITTED 分離レベルをサポートします。
F121、基本的な診断管理	この機能の大部分の機能は、埋込み言語の SQLCA によって提供されます。
F191、参照削除アクション	Oracle は、ON DELETE CASCADE および ON DELETE SET NULL をサポートします。
F200、TRUNCATE TABLE	Oracle はこの機能を完全にサポートしています。また、参照整合性制約で自身を参照する表の切捨てを可能にすることと、ON DELETE CASCADE 参照制約が有効にされている子表に向けてカスケードする機能により、この機能を拡張しています。
F231、権限表	Oracle は、この情報を次のメタデータ・ビューで使用可能にします。 ● TABLE_PRIVILEGES のかわりに、ALL_TAB_PRIVS を使用します。 ● COLUMN_PRIVILEGES のかわりに、ALL_COL_PRIVS を使用します。 ● Oracle は USAGE 権限をサポートしないため、USAGE_PRIVILEGES と同等の権限は存在しません。
F281、LIKE 拡張	この機能を完全にサポートします。

F291、UNIQUE 述語	IS A SET 条件は、多重集合が集合であるかどうか(各行が一意であるかどうか)のテストに使用できます。したがって、次の
	`UNIQUE <table subquery>`
	は、次と同じです。
	`CAST (<table subquery> AS MULTISET) IS A SET`

F302、INTERSECT 表演算子	Oracle の構文は、UNION、INTERSECT および MINUS の優先順位が同じであるという点で規格と異なります。
F312、MERGE 文	Oracle の MERGE 文は規格とほとんど同じですが、次の制限事項があります。 ● Oracle は、表の別名の前の AS キーワード(オプション)をサポートしません。

機能識別子、機能	サポート
	● Oracle は、表の別名の後にカッコで囲まれた列名のリストを使用して、USING 句で指定した表の列名を変更する機能をサポートしません。 ● Oracle は、<上書き句>をサポートしません。
F314、DELETE ブランチのある MERGE 文	Oracle には、同様の機能があります。ただし、Oracle では最初に行を更新してから、更新された行が条件を満たした場合に削除できるようになります。
F321、ユーザーの認可	Oracle は、次の副機能と同等の機能を提供します。 ● SESSION_USER のかわりに、SYS_CONTEXT ('USERENV', 'SESSION_USER')の使用 ● CURRENT_USER のかわりに、SYS_CONTEXT ('USERENV', 'CURRENT_USER')の使用 Oracle は、次の副機能をサポートしていません。 ● SYSTEM_USER ● SET SESSION AUTHORIZATION 文
F341、使用状況表	Oracle は、この情報を ALL_DEPENDENCIES、DBA_DEPENDENCIES および USER_DEPENDENCIES ビューで使用可能にします。
F381、拡張スキーマ操作	Oracle は、この機能の次の要素を完全にサポートします。 ● Oracle は、ALTER TABLE を使用して、表の制約を追加する規格の構文をサポートします。 Oracle は、この機能の次の要素を部分的にサポートします。 ● Oracle は、表の制約を削除する規格の構文をサポートしますが、RESTRICT はサポートしません。 Oracle は、この機能のうち、次の要素と同等の機能を提供します。 ● 列のデフォルト値を変更するには、ALTER TABLE の MODIFY オプションを使用します。 Oracle は、この機能の次の部分をサポートしません。 ● DROP SCHEMA 文

機能識別子、機能	サポート
	● ALTER ROUTINE 文
F382、列データ型の変更	Oracle は、規格とは異なる構文でこの機能をサポートします。規格を拡張しているため、Oracle では、列のサイズまたは精度を小さくできます。
F383、Set column not null 句	Oracle は、この機能の 2 つの副機能と同等の機能を提供します。 ● 既存の列に NOT NULL 制約を追加するための、ALTER TABLE ... MODIFY の使用 ● NOT NULL 制約を削除するための、ALTER TABLE を使用した名前による制約の削除
F384、Drop identity property 句	Oracle は、ALTER TABLE ... MODIFY (... DROP IDENTITY)を使用することで、同等の機能を提供します。
F386、Set identity column generation 句	Oracle は、同等の機能を提供します。Oracle の構文とセマンティクスは規格と同じですが、次の例外があります。 ● Oracle は、RESTART をサポートしていません。そのかわりに、START WITH を使用します。ID 列を再開すると、その ID 列の他のパラメータ値はデフォルトにリセットされます(ただし、ALTER TABLE 文で明示的に設定している場合を除く)。 Oracle の START WITH LIMIT VALUE オプションは、規格に対する拡張です。
F391、長い識別子	Oracle は、最大 128 文字の識別子をサポートします。
F393、リテラルの Unicode エスケープ	Oracle の UNISTR 機能は、すべての Unicode 文字に対する数値のエスケープ・シーケンスをサポートします。
F394、オプションの正規形指定	この機能により、NORMALIZE ファンクションおよび IS NORMAL 述語にキーワード NFC、NFD、NFKC および NKD が追加されます。これらのキーワードを指定しない場合、NFC がデフォルトです(「T061、UCS のサポート」を参照)。Oracle は、4 つすべての正規形を、次のように規格とは異なる構文でサポートします。 ● NFC については、COMPOSE を使用します。 ● NFD については、DECOMPOSE に CANONICAL オプションを付けて使用します。

機能識別子、機能	サポート
	● NFKD については、DECOMPOSE に COMPATIBILITY オプションを付けて使用します。 ● NFKC については、DECOMPOSE に CANONICAL オプションと COMPOSE を付けて使用します。 Oracle は、IS NORMAL 述語をサポートしません。
F401、拡張結合表	Oracle は、FULL 外部結合、CROSS 結合および NATURAL 結合をサポートします。
F402、LOB、配列および多重集合での名前付き列の結合	Oracle は、宣言した型がネストした表である列での名前付き列の結合をサポートします。Oracle は、LOB または配列での名前付き列の結合はサポートしません。
F403、パーティション結合表	Oracle は、FULL 外部結合を除くこの機能をサポートします。
F411、タイムゾーンの指定	Oracle は、TIMESTAMP WITH TIME ZONE を完全サポートしますが、TIME WITH TIME ZONE はサポートしません。
F421、各国語キャラクタ	この機能を完全にサポートします。
F431、読取り専用のスクロール可能なカーソル	この機能を完全にサポートします。
F441、拡張集合ファンクション・サポート	Oracle は、この機能の次の部分をサポートします。 ● ビューまたはインライン・ビューのいずれかで集計を使用して定義された列を参照する WHERE 句の機能 ● 式の DISTINCT を使用しない COUNT ● 集計問合せに対する外部参照となる列を参照する集計。ただし、Oracle は集計問合せの定義を、最も内側にある集計を含んでいる問合せとしています。これは、集計内で参照される範囲変数を定義する最も内側の問合せではありません。
F442、集合ファンクションでの複合列の参照	この機能を完全にサポートします。

機能識別子、機能	サポート
F461、名前付きの文字セット	Oracle は、Oracle 定義の名前が付いた複数の文字セットをサポートします。Oracle は、この機能の他の部分をサポートしません。
F491、制約管理	この機能を完全にサポートします。
F492、オプションの表の制約の強制	規格の ENFORCED は、Oracle の ENABLE VALIDATE と同じです。規格の NOT ENFORCED は、Oracle の DISABLE NOVALIDATE と同じです。それ以外の ENABLE | DISABLE オプション、VALIDATE | NOVALIDATE オプション、および RELY | NORELY オプションの組合せは、規格の拡張です。
F531、一時表	Oracle は、GLOBAL TEMPORARY 表をサポートします。
F555、拡張秒精度	Oracle はこの機能を拡張して、小数点以下 9 桁までをサポートします。
F561、完全評価式	この機能を完全にサポートします。
F571、真理値テスト	Oracle の LNNVL ファンクションは、規格の IS NOT TRUE 述語と同等です。
F591、導出表	Oracle は<導出表>をサポートしますが、次の制限事項があります。 ● Oracle は、表の別名の前の AS キーワード(オプション)をサポートしません。 ● Oracle は、<導出列リスト>をサポートしません。
F641、行コンストラクタおよび表コンストラクタ	Oracle では、他の行コンストラクタまたは副問合せとの等価性または非等価性の比較に行コンストラクタを使用できます。Oracle は、この機能の他の部分をサポートしません。
F690、照合サポート	Oracle の NLSSORT ファンクションは、文字式の照合の変更に使用できます。
F693、SQL セッションおよびクライアント・モジュールの照合	セッション照合を設定するには、ALTER SESSION SET NLS_COMP = 'LINGUISTIC'を使用するとともに、必要な照合に NLS_SORT を設定します。Oracle は、クライアント・モジュールの照合をサポートしません。
F695、変換のサポート	Oracle の CONVERT ファンクションでは、データベース文字セットと各国語文字セット間での変換が可能です。その他の文字セットについては、データを RAW データ型に格納して、PL/SQL パッケージ・ファンクション UTL_RAW.CONVERT を使用します。Oracle には、文字セットの変換を追加したり、削除する機能はありません。

機能識別子、機能	サポート
F721、遅延可能制約	この機能を完全にサポートします。
F731、列 INSERT 権限	この機能を完全にサポートします。
F761、セッション管理	Oracle は、この機能のうち、次の要素と同等の機能を提供します。 ● 規格の SET SESSION CHARACTERISTICS AS TRANSACTION SERIALIZABLE と同等な機能は、ALTER SESSION SET ISOLATION_LEVEL = SERIALIZABLE です。 ● 規格の SET SCHEMA と同等な機能は、ALTER SESSION SET CURRENT_SCHEMA です。 ● 規格の SET COLLATION と同等な機能は、ALTER SESSION SET NLS_SORT です。
F763、CURRENT_SCHEMA	Oracle の同等機能は、SYS_CONTEXT ('USERENV', 'CURRENT_SCHEMA')です。
F771、結合管理	Oracle の CONNECT 文は規格の CONNECT 文と同じ機能を持ちますが、構文は異なります。規格の SET CONNECTION を使用するかわりに、Oracle では SQL 文を実行する必要のある接続を指定する AT 句を提供します。Oracle の埋込み言語では、COMMIT または ROLLBACK の RELEASE オプションを使用して接続を切断できます。
F781、自己参照操作	この機能を完全にサポートします。
F801、完全集合ファンクション	この機能を完全にサポートします。
F831、完全カーソル更新	Oracle は、問合せでの FOR UPDATE 句と ORDER BY 句の組合せをサポートします。
F841、LIKE_REGEX 述語	Oracle では REGEXP_LIKE に相当します。Oracle のパターン構文は規格の機能の一部が欠落しています。Oracle の一致パラメータは一部の綴りが違いますが、規格と同じ機能を持っています。
F842、OCCURRENCES_REGEX ファンクション	Oracle では REGEXP_COUNT に相当します。Oracle のパターン構文は規格の機能の一部が欠落しています。Oracle の一致パラメータは一部の綴りが違いますが、規格と同じ機能を持っています。

機能識別子、機能	サポート
F843、POSITION_REGEX ファンクション	Oracle では REGEXP_INSTR に相当します。Oracle のパターン構文は規格の機能の一部が欠落しています。Oracle の一致パラメータは一部の綴りが違いますが、規格と同じ機能を持っています。
F844、SUBSTRING_REGEX ファンクション	Oracle では REGEXP_SUBSTR に相当します。Oracle のパターン構文は規格の機能の一部が欠落しています。Oracle の一致パラメータは一部の綴りが違いますが、規格と同じ機能を持っています。
F845、TRANSLATE_REGEX ファンクション	Oracle では REGEXP_REPLACE に相当します。Oracle のパターン構文は規格の機能の一部が欠落しています。Oracle の一致パラメータは一部の綴りが違いますが、規格と同じ機能を持っています。
F850、<問合せ式>でのトップレベルの<order by 句>	この機能を完全にサポートします。
F851、副問合せでの<order by 句>	この機能を完全にサポートします。
F852、ビューでのトップレベルの<order by 句>	この機能を完全にサポートします。
F855、<問合せ式>でのネストした<order by 句>	この機能を完全にサポートします。
F856、<問合せ式>でのネストした<fetch first 句>	この機能を完全にサポートします。
F857、<問合せ式>でのトップレベルの<fetch first 句>	この機能を完全にサポートします。
F858、副問合せでの<fetch first 句>	この機能を完全にサポートします。
F859、ビューでのトップレベルの<fetch first 句>	この機能を完全にサポートします。
F860、<fetch first 句>での動的<fetch first row count>	この機能を完全にサポートします。

機能識別子、機能	サポート
F861、<問合せ式>でのトップレベルの<result offset 句>	この機能を完全にサポートします。
F862、副問合せでの<result offset 句>	この機能を完全にサポートします。
F863、<問合せ式>でのネストした<result offset 句>	この機能を完全にサポートします。
F864、ビューでのトップレベルの<result offset 句>	この機能を完全にサポートします。
F865、<result offset 句>での動的<offset row count>	この機能を完全にサポートします。
F866、FETCH FIRST 句: PERCENT オプション	この機能を完全にサポートします。
F867、FETCH FIRST 句: WITH TIES オプション	この機能を完全にサポートします。
R010、行パターン認識: FROM 句	この機能を完全にサポートします。
S023、基本的な構造型	Oracle のオブジェクト型は、規格の構造型と同等です。
S024、拡張構造型	Oracle の構文は規格と異なりますが、次のものと同等の機能を提供します。 ● NOT INSTANTIABLE ● STATIC メソッド ● RELATIVE、MAP および STATE の順序付け。RELATIVE 順序付けに相当する Oracle のキーワードは ORDER です。STATE 順序付けに相当するキーワードはありません(他の順序付けが定義されていない場合、これがデフォルトになります)。規格とは異なり、Oracle は STATE 以外の順序付けの EQUALS ONLY をサポートしません。(「S251、ユーザー定義の順序付け」も参照。) ● コンストラクタ・メソッドのシグネチャの SELF AS RESULT

機能識別子、機能	サポート
S025、最終構造型	Oracle の最終オブジェクト型は、規格の最終構造型と同等です。
S026、自己参照構造型	Oracle では、オブジェクト型 OT は、OT を参照する参照を持つことができます。
S041、基本的な参照型	Oracle の参照型は、規格の参照型と同等です。参照を解除するには、規格の-> のかわりにドット表記法を使用します。
S043、拡張参照型	Oracle は、この機能の次の要素をサポートします。 ● 参照によって参照されるオブジェクトを返す DEREF 演算子 ● 表またはマテリアライズド・ビューの列に対する制約としての SCOPE 句 ● 列の有効範囲の追加および削除 ● システムが生成した参照または主キーから導出された参照(他の列のリストまたは型の属性のリストから導出されたものを除く)
S051、型の表の作成	Oracle のオブジェクト表は、規格の構造型の表と同等です。
S081、サブテーブル	Oracle はオブジェクト・ビューの階層をサポートしますが、オブジェクトの実表の階層はサポートしません。実表の階層をエミュレートするには、それらの実表にビューの階層を作成します。
S091、基本的な配列のサポート	Oracle の VARRAY 型は、規格の配列型と同等です。ただし、Oracle は、LOB の配列の記憶域をサポートしません。添字を使用して配列の単一要素にアクセスするには、PL/SQL を使用する必要があります。Oracle は、規格以外の構文を使用してこの機能の次の部分をサポートします。 ● 空の配列を含む VARRAY 型のインスタンスを構成するには、VARRAY 型コンストラクタを使用します。 ● FROM 句で VARRAY をネスト解除するには、TABLE 演算子を使用します。 ● VARRAY のカーディナリティを取得するには、PL/SQL の COUNT メソッドを使用します。
S092、ユーザー定義型の配列	Oracle は、オブジェクト型の VARRAY をサポートします。

機能識別子、機能	サポート
S094、参照型の配列	Oracle は、参照の VARRAY をサポートします。
S095、問合せ別の配列コンストラクタ	Oracle は、CAST (MULTISET (SELECT ...) AS varray_type)を使用して配列コンストラクタをサポートします。ORDER BY を使用して配列の要素を順序付ける機能はサポートされません。
S097、配列要素割当て	PL/SQL では、規格(SQL/PSM)に似た構文を使用して配列要素を割り当てることができます。
S098、ARRAY_AGG	Oracle には、VARRAY になる集計はありません。そのかわりに、COLLECT 集計を使用して多重集合を作成すると、配列の要素タイプにキャストできます。
S111、問合せ式での ONLY	Oracle は、ビューの階層に対する ONLY 句をサポートしますが、実表の階層はサポートしません。
S151、型述語	この機能を完全にサポートします。
S161、サブタイプ処理	この機能を完全にサポートします。
S162、参照のサブタイプ処理	構文が多少異なりますが、サポートします。規格では参照する型の名前をカッコで囲む必要がありますが、Oracle はこの位置でのカッコの使用をサポートしません。
S201、配列での SQL 起動ルーチン	PL/SQL は、配列をパラメータとして渡し、ファンクションの結果として配列を返す機能を提供します。C で記述されたプロシージャおよびファンクションは、Oracle Type Translator(OTT)を使用して配列を渡し、ファンクションの結果として配列を返すことができます。
S202、多重集合での SQL 起動ルーチン	PL/SQL ルーチンは、パラメータとしてネストした表を持ち、ネストした表を返すことができます。C で記述されたルーチンは、Oracle Type Translator を使用して配列を渡し、ファンクションの結果として配列を返すことができます。
S232、配列ロケータ	Oracle Type Translator は配列の記述子をサポートします。記述子はロケータと同じ用途に使用できます。
S233、多重集合ロケータ	Oracle は、ネストした表のロケータをサポートします。
S241、変換ファンクション	Oracle Type Translator は、変換と同じ機能を提供します。

機能識別子、機能	サポート
S251、ユーザー定義の順序付け	Oracle のオブジェクト型の順序付け機能は、次のとおり規格の機能に対応します。 ● Oracle の MAP 順序付けは、規格の ORDER FULL BY MAP 順序付けに対応します。 ● Oracle の ORDER 順序付けは、規格の ORDER FULL BY RELATIVE 順序付けに対応します。 ● Oracle のオブジェクト型で MAP または ORDER のいずれも宣言されていない場合、これは規格の EQUALS ONLY BY STATE に対応します。 ● Oracle には、順序付けられていないオブジェクト型は存在しません。そのため、順序付けは変更できますが、削除することはできません。
S261、特定の TYPE メソッド	ANYDATA 型の GetTypeName メソッドは、型名を調べるために使用できます。
S271、基本的な多重集合のサポート	Oracle では、ネストした表型として規格の多重集合がサポートされます。スカラー型(ST)に基づいた Oracle のネストした表のデータ型は、規格の用語では、ST 型の単一フィールドを持つ column_value という名前の行の多重集合と同じです。オブジェクト型に基づいた Oracle のネストした表型は、規格の構造型の多重集合と同等です。 Oracle は、ネストした表で使用するこの機能のうち、次の要素をサポートします。この場合、規格で多重集合に使用するのと同じ構文を使用します。 ● CARDINALITY ファンクション ● SET ファンクション ● MEMBER 述語 ● IS A SET 述語 ● COLLECT 集計 次のように、この機能の他のすべての部分が規格以外の構文でサポートされます。 ● 規格に MULTISET[]と表される、空の多重集合を作成するには、ネストした表型の空のコンストラクタを使用します。 ● 規格では、ELEMENT(<多重集合値の式>)と表される、要素が 1 つ付いた多重集合の要素を 1 つのみ取得するには、スカラー副問合せを使用して、ネストした表から単一要素を選択します。

機能識別子、機能	サポート
	● 多重集合を列挙ごとに構成するには、ネストした表型のコンストラクタを使用します。 ● 多重集合を問合せごとに構成するには、多重集合の引数を指定した CAST を使用して、ネストした表型にキャストします。 ● 多重集合をネスト解除するには、FROM 句の TABLE 演算子を使用します。
S272、ユーザー定義型の多重集合	Oracle のネストした表型では、構造型の多重集合が可能です。Oracle には固有型がないため、固有型の多重集合はサポートされません。
S274、参照型の多重集合	ネストした表型は、参照型の列を 1 つ以上持つことができます。
S275、拡張多重集合のサポート	Oracle は、ネストした表で使用するこの機能のうち、次の要素をサポートします。この場合、規格で多重集合に使用するのと同じ構文を使用します。 ● MULTISET UNION、MULTISET INTERSECTION および MULTISET EXCEPT 演算子 ● SUBMULTISET 述語 ● =および<>述語 Oracle は、FUSION または INTERSECTION 集計をサポートしません。
S281、ネストしたコレクション型	Oracle では、コレクション型のネストが可能です(VARRAY およびネストした表)。
S401、配列型に基づく固有型	Oracle の VARRAY 型は強い型指定です。
S403、ARRAY_MAX_CARDINALITY	PL/SQL では、VARRAY の LIMIT メソッドが最大カーディナリティを返します。
S404、TRIM_ARRAY	PL/SQL では、VARRAY の TRIM メソッドを VARRAY の切り捨てに使用できます。
T041、基本的な LOB データ型サポート	Oracle は、この機能の次の部分をサポートします。 ● キーワード BLOB、CLOB および NCLOB ● 連結(CLOB での UPPER、LOWER)

機能識別子、機能	サポート
	Oracle は、この機能のうち、次の部分と同等のサポートを提供します。 ● POSITION のかわりに INSTR の使用 ● CHAR_LENGTH のかわりに LENGTH の使用。 Oracle は、この機能の次の部分をサポートしません。 ● BLOB、CLOB および NCLOB のそれぞれのシノニムとしてのキーワード BINARY LARGE OBJECT、CHARACTER LARGE OBJECT および NATIONAL CHARACTER LARGE OBJECT ● <バイナリ文字列リテラル> ● BLOB や CLOB の長さの上限を指定する機能 ● BLOB の結合
T042、拡張 LOB のサポート	Oracle は、この機能の次の要素を完全にサポートします。 ● CLOB 引数の TRIM ファンクション Oracle は、この機能のうち、次の要素と同等の機能を提供します。 ● BLOB および CLOB サブストリング(SUBSTR を使用してサポート) ● SIMILAR 述語(REGEXPR_LIKE を使用し、Perl に似た構文とのパターン一致を実行してサポート) この機能の次の要素はサポートされません。 ● BLOB または CLOB オペランドの付いた比較述語 ● BLOB または CLOB オペランドの付いた CAST ● OVERLAY(SUBSTR および文字列連結を使用してエミュレート可能) ● BLOB または CLOB オペランドの付いた LIKE 述語
T051、行型	Oracle のオブジェクト型は、規格の行型のかわりに使用できます。
T061、UCS のサポート	Oracle は、この機能のうち、次の要素と同等の機能を提供します。

機能識別子、機能	サポート
	● Oracle は、文字データ型の宣言では、CHARACTERS と OCTETS のかわりに、それぞれ CHAR と BYTE のキーワードをサポートします。 ● Oracle の COMPOSE ファンクションは、規格の NORMALIZE ファンクションと同等です。 Oracle は、IS NORMALIZED 述語をサポートしません。
T071、BIGINT データ型	多くの実装では、BIGINT は、19 桁(10 進)の大部分をサポートする 64 ビットの 2 進整数型を参照します。Oracle の NUMBER 型は、39 桁(10 進)をサポートします。
T111、更新可能な結合、論理和および列	Oracle の更新可能な結合ビューは、規格の更新可能な結合機能と類似しています。規格とは異なり、Oracle では、SELECT 構文のリストに強力な候補キーを表示するために、更新可能な結合ビューは必要ありません。更新可能な結合ビューには複数のキー保存表が含まれることがありますが、UPDATE または DELETE を使用して更新されるキー保存表は、それらのうちの 1 つの表のみです。更新可能な結合に含まれるすべてのキー保存表が変更される規格とは異なります。
T121、問合せ式での WITH(RECURSIVE を除く)	この機能を完全にサポートします。
T122、副問合せでの WITH(RECURSIVE を除く)	この機能を完全にサポートします。
T131、再帰的問合せ	Oracle は自身を参照する WITH 句要素の使用をサポートしますが、RECURSIVE キーワードはサポートしません。または、Oracle の START WITH および CONNECT BY 句を使用すると、多くの再帰的問合せを実行できます。
T132、副問合せでの再帰的問合せ	Oracle は自身を参照する WITH 句要素の使用をサポートしますが、RECURSIVE キーワードはサポートしません。または、Oracle の START WITH および CONNECT BY 句を使用すると、多くの再帰的問合せを実行できます。
T141、SIMILAR 述語	Oracle は、Perl に似た構文とのパターン一致のための REGEXP_LIKE を提供します。
T172、表定義の AS 副問合せ句	Oracle の CREATE TABLE の AS 副問合せ機能は、規格とほとんど同じ機能を持ちますが、構文の一部が異なります。

機能識別子、機能	サポート
T174、ID 列	Oracle はこの機能をサポートしますが、構文には次のような相違点があります。 ● Oracle では、規格の NO MINVALUE および NO MAXVALUE のかわりに、NOMINVALUE および NOMAXVALUE を使用します。 ● ID 列を再開するには、ALTER TABLE MODIFY 文で START WITH LIMIT VALUE を使用して、最高値(増分 ID 列の場合)または最低値(減分 ID 列の場合)から再開します。特定の番号から再開するには、START WITH Number を使用します。 GENERATED BY DEFAULT ON NULL は、Oracle の拡張機能です。
T175、生成された列	Oracle はこの機能をサポートしますが、次の制限事項があります。 ● 一時表では生成された列はサポートされていません。 ● 生成された列のデータ型には LOB または XML を使用できません。
T176、順序ジェネレータのサポート	Oracle の順序は規格のものと同じ機能を持ちますが、構文は異なります。
T178、ID 列: 簡易再開オプション	Oracle の START WITH LIMIT VALUE は、ID 列が循環していない場合は規格の簡易再開と同じです。
T180、システムバージョン付きの表	Oracle のフラッシュバック機能は、規格のシステムバージョン付きの表と実質的に同じです。主な相違点は次のとおりです。 ● Oracle では、特定の表をジャーナリング用に指定する必要はありません。すべての表がジャーナリングされます。 ● Oracle では、LOB 列は個別にジャーナリング用に指定する必要があります。これは、データ量が巨大になる可能性があるためです。規格には、同様のプロビジョニングはありません。 ● Oracle では、履歴データを読み取るために権限が必要です。 ● 規格では、ジャーナリングされた表には、行の開始と終了のタイムスタンプを記録するための列があります。Oracle では、これは疑似列によってプロビジョニングされます。
T181、アプリケーション期間表	Oracle は、この機能の次の要素をサポートします。 ● CREATE TABLE 中のアプリケーション期間定義

機能識別子、機能	サポート
	● ALTER TABLE を使用したアプリケーション期間定義の追加および削除。構文が多少異なり、Oracle では期間指定をカッコで囲む必要がありますが、規格ではこの位置でのカッコの使用はサポートされません。 Oracle ではこの機能を次のように拡張しています。 ● 表ごとに複数のアプリケーション期間を指定できます。 ● 開始時間と終了時間の列をオプションにできます。この場合、これらの列は暗黙的に作成されます。 ● 開始時間列に NULL を指定することで、終了時間列の値以前の時点を示すすべての行が有効とみなされるようにできます。 ● 終了時間列に NULL を指定することで、開始時間列の値以降の時点を示すすべての行が有効とみなされるようにできます。 ● フラッシュバック問合せのオプション VERSIONS PERIOD FOR と AS OF PERIOD FOR を使用して、アプリケーション期間表を問い合せることができます。
T201、参照制約での比較可能データ型	この機能を完全にサポートします。
T211、基本的なトリガー機能	Oracle のトリガーと規格のトリガーの違いは、次のとおりです。 ● Oracle は、デフォルトではオプションの構文 FOR EACH STATEMENT(文トリガー)を提供しません。 ● Oracle は、OLD TABLE および NEW TABLE をサポートしません。規格で指定されている遷移表(影響される行のイメージの前後の多重集合)は使用できません。 ● トリガー本体は PL/SQL で記述されています。PL/SQL の機能は規格の手続き型言語 PSM と同等ですが、同じものではありません。 ● トリガー本体では、古い遷移変数と新しい遷移変数が、コロンで始まる形式で参照されます。 ● Oracle の行トリガーは、バッファリングおよびすべての行の処理後にまとめて実行されるのではなく、行の処理と同時に実行されます。規格のセマンティク

機能識別子、機能	サポート
	スは決定的ですが、Oracle のリアルタイムで実行される行トリガーはより実用的です。 ● Oracle の行および文の前のトリガーでは DML 文を実行できますが、規格では許可されていません。これに対し、Oracle の行の後の文では DML 文を実行できませんが、規格では許可されています。 ● 複数のトリガーが適用される場合、規格ではそれらのトリガーは定義順に実行されます。Oracle では、実行順序は、非決定的になります(FOLLOWS を使用して指定した場合を除く)。 ● Oracle では、規格の(表権限の)TRIGGER 権限のかわりに、システム権限 CREATE TRIGGER および CREATE ANY TRIGGER を使用してトリガーの作成を規制します。
T212、拡張トリガー機能	この機能によって、Oracle がサポートする文トリガーが許可されます(「T211、基本的なトリガー機能」を参照)。
T213、INSTEAD OF トリガー	Oracle はビューの INSTEAD OF トリガーをサポートします。「T211、基本的なトリガー機能」で説明した内容を除き、構文およびセマンティクスは規格に準拠します。Oracle は、WITH CHECK OPTION を指定したビューで、INSTEAD OF トリガーを使用できます。これは、規格とは異なります。
T241、START TRANSACTION 文	Oracle の SET TRANSACTION 文は、規格の SET TRANSACTION ではなく、規格の START TRANSACTION と同等のトランザクションを開始します。Oracle の READ ONLY トランザクションは SERIALIZABLE 分離レベルです。
T271、セーブポイント	Oracle はこの機能をサポートしますが、次の制限事項があります。 ● Oracle は、RELEASE SAVEPOINT をサポートしません。 ● Oracle は、セーブポイント・レベルをサポートしません。
T285、拡張導出列名	この機能は、列の別名を指定せず、SQL パラメータ参照で構成される SELECT 構文のリストの導出列にのみ適用されます。この場合、規格と同様に、列名のデフォルトはパラメータ名になります。
T323、外部ルーチンの明示的なセキュリティ	外部ファンクション、プロシージャまたはパッケージの作成時に使用される Oracle の構文 AUTHID { CURRENT USER | DEFINER }は、規格の EXTERNAL SECURITY { DEFINER | INVOKER }と同等です。

機能識別子、機能	サポート
T324、SQL ルーチンの明示的なセキュリティ	PL/SQL ファンクション、プロシージャまたはパッケージの作成時に使用される Oracle の構文 AUTHID { CURRENT USER | DEFINER }は、規格の SQL SECURITY { DEFINER | INVOKER }と同等です。
T325、修飾された SQL パラメータ参照	PL/SQL は、ルーチン名を使用したパラメータ名の修飾をサポートします。
T326、表ファンクション	Oracle は、この機能のうち、次の要素と同等の機能を提供します。 ● <問合せ別の多重集合値コンストラクタ>は、CAST(MULTISET(<問合せ式>)AS <ネストした表型>)を使用してサポートされます。 ● <表ファンクションの導出表>は、VARRAY またはネストした表を引数として指定した FROM 句の TABLE 演算子を使用してサポートされます。 ● <コレクション値の式>は、VARRAY またはネストした表になる Oracle の式と同等です。 ● <戻り値の表型>は、ネストした表を戻す PL/SQL ファンクションと同等です。
T331、基本的なロール	Oracle は、REVOKE ADMIN OPTION FOR の<ロール名>を除いて、この機能をサポートします。
T341、SQL 起動ファンクションおよびプロシージャのオーバーロード	Oracle は、ファンクションおよびプロシージャのオーバーロードをサポートします。ただし、特定のデータ型の組合せを処理するルールは、規格と異なります。たとえば、規格では、引数の数値型のみが異なる同一名の 2 つのファンクションの共存が許可されますが、Oracle では許可されません。
T351、大カッコで囲まれたコメント	この機能を完全にサポートします。
T431、拡張グループ化機能	この機能を完全にサポートします。
T432、ネストおよび連結した GROUPING SETS	Oracle は、連結した GROUPING SETS をサポートしますが、ネストした GROUPING SETS はサポートしません。
T433、複数引数ファンクション GROUPING	Oracle の GROUP_ID ファンクションを使用すると、グループ化した問合せによりグループを適切に区別できます。このファンクションは、規格の複数引数の GROUPING ファンクションの場合と同じ用途で使用できます。

機能識別子、機能	サポート
T441、ABS および MOD ファンクション	Oracle は ABS ファンクションをサポートします。Oracle の MOD ファンクションは規格に似ていますが、2 つの引数の符号が異なる場合の動作が異なります。
T471、結果セットの戻り値	PL/SQL REF カーソルは、規格の結果セット・カーソルのすべての機能を提供します。
T491、LATERAL 導出表	この機能を完全にサポートします。
T501、拡張 EXISTS 述語	この機能を完全にサポートします。
T511、トランザクションのカウント	Oracle は、コミットされたトランザクションとロールバックされたトランザクションのカウントを、システム・ビューの V$STATNAME と V$SESSTAT でサポートします。
T521、CALL 文の名前付き引数	この機能を完全にサポートします。
T522、SQL 起動プロシージャの IN パラメータのデフォルト値	この機能を完全にサポートします。
T524、CALL 文以外のルーチンの起動での名前付き引数	この機能を完全にサポートします。
T525、SQL 起動ファンクションのパラメータのデフォルト値	この機能を完全にサポートします。
T571、配列を戻す外部 SQL 起動ファンクション	VARRAY を戻す Oracle の表ファンクションは、外部のプログラミング言語で定義できます。SQL でこのようなファンクションを宣言する場合は、CREATE FUNCTION コマンドを PIPELINED USING 句とともに使用します。
T572、多重集合を戻す外部 SQL 起動ファンクション	ネストした表を戻す Oracle の表ファンクションは、外部のプログラミング言語で定義できます。SQL でこのようなファンクションを宣言する場合は、CREATE FUNCTION コマンドを PIPELINED USING 句とともに使用します。ファンクションの本体では、OCITable インタフェースを使用します。ファンクションは、FROM 句の TABLE 演算子内で起動する必要があります。
T581、正規表現のサブストリング・ファンクション	Oracle では、正規表現の一致を使用してサブストリング操作を実行する REGEXP_SUBSTR ファンクションを提供します。
T591、NULL の可能性のある列に対する UNIQUE 制約	Oracle では、NULL であることが可能な 1 つ以上の列に対する UNIQUE 制約が可能です。UNIQUE 制約の対象が単一列の場合、セマンティクスは規格と同じにな

機能識別子、機能	サポート
	ります(この制約には、指定した列の NULL である行の数に制限はありません)。UNIQUE 制約の対象が 2 つ以上の列の場合、セマンティクスは規格外になります。Oracle では、指定したすべての列の NULL である行の数に制限はありません。規格と異なり、指定した列の 1 つ以上の行が NULL 以外になる場合、制約対象の NULL 以外の列の値と同じ値を持つ別の行は、制約違反になり、許可されません。
T611、基本的な OLAP 演算子	Oracle はこの機能を完全にサポートします。ただし、DISTINCT は、ウィンドウ・パーティショニングと併用している場合にのみサポートされます(ウィンドウ・フレーミングと併用している場合はサポートされません)。
T612、拡張 OLAP 操作	Oracle は、この機能の次の要素をサポートします。PERCENT_RANK、CUME_DIST、WIDTH_BUCKET、仮想の集合ファンクション、PERCENTILE_CONT、PERCENTILE_DISC、および ROW_NUMBER です。 Oracle は、この機能の次の要素をサポートしません。 ● ORDER BY のない ROW_NUMBER
T613、サンプリング	Oracle では、規格のキーワード TABLESAMPLE のかわりに、キーワード SAMPLE を使用します。Oracle では、規格のキーワード SYSTEM のかわりに、キーワード BLOCK を使用します。Oracle では、行のベルヌーイ・サンプリングの指定に、BLOCK キーワードを使用しません。規格では、キーワード BERNOULLI でこのサンプリングを指定します。Oracle は、キー保存していない導出表や導出ビューサンプリングをサポートしていません。Oracle では、DELETE 文、UPDATE 文または MERGE 文の副問合せでのサンプリングはできません。
T614、NTILE ファンクション	この機能を完全にサポートします。
T615、LEAD ファンクションおよび LAG ファンクション	この機能を完全にサポートします。
T616、LEAD ファンクションおよび LAG ファンクションの NULL 処理オプション	この機能を完全にサポートします。
T617、FIRST_VALUE ファンクションおよび LAST_VALUE ファンクション	この機能を完全にサポートします。
T618、NTH_VALUE ファンクション	この機能を完全にサポートします。

機能識別子、機能	サポート
T621、拡張数値ファンクション	Oracle は、CEIL ファンクションの CEILING の綴りが異なることを除いて、この機能を完全にサポートします。
T622、三角法ファンクション	この機能を完全にサポートします。
T623、一般的な対数ファンクション	この機能を完全にサポートします。
T625、LISTAGG	この機能を完全にサポートします。
T641、複数列の割当て	割当てソースが副問合せの場合、複数の列を割り当てる規格の構文がサポートされます。
T652、SQL ルーチンでの SQL 動的文。	PL/SQL は、動的 SQL をサポートします。
T654、外部ルーチンでの SQL 動的文	Oracle は、埋込み C での動的 SQL をサポートします。これは、外部ルーチンの作成に使用できます。
T655、循環的な依存ルーチン	PL/SQL は、再帰型をサポートします。
T811、基本 SQL/JSON コンストラクタ・ファンクション	Oracle はこの機能を完全にサポートしています(問合せによる JSON_ARRAY コンストラクタを除く)。
T812、SQL/JSON: JSON_OBJECTAGG	この機能を完全にサポートします。
T813、SQL/JSON: JSON_ARRAYAGG と ORDER BY	この機能を完全にサポートします。
T821、基本的な SQL/JSON 問合せ演算子	この機能を完全にサポートします。
T822、SQL/JSON: IS JSON WITH UNIQUE KEYS 述語	この機能を完全にサポートします。
T823、SQL/JSON: PASSING 句	Oracle は、JSON_EXISTS に PASSING 句をサポートします。
T825、SQL/JSON: ON EMPTY および ON ERROR 句	Oracle は、次の制限事項を除き、この機能を完全にサポートします。

機能識別子、機能	サポート
	● JSON_EXISTS の ON ERROR 句は UNKNOWN をサポートしません。 ● JSON_TABLE は、列レベルの ON EMPTY 句をサポートしません。
T828、JSON_QUERY	この機能を完全にサポートします。
T829、JSON_QUERY: 配列ラッパー・オプション	この機能を完全にサポートします。
T832、SQL/JSON パス言語: アイテム・メソッド	Oracle は、次のアイテム・メソッドを完全にサポートします。 ● abs ● ceiling ● double ● floor Oracle は、次の同等のサポートを提供します。 ● date および timestamp は、標準の datetime に相当します Oracle では、次のアイテム・メソッドをサポートしてこの機能を拡張しています。 ● length ● lower ● number ● string ● upper
T833、SQL/JSON パス言語: 複数の添字	添字を厳密に単調増加する順序で指定する必要がある点を除き、Oracle はこの機能を完全にサポートします。
T834、SQL/JSON パス言語: ワイルドカード・メンバー・アクセサ	この機能を完全にサポートします。
T835、SQL/JSON パス言語: フィルタ式	Oracle では、JSON_EXISTS の SQL/JSON パス式の最後のステップとしてフィルタ式をサポートします。

機能識別子、機能	サポート
T839、文字列へ、または文字列からの datetime の書式指定されたキャスト	Oracle は、構文上の多少の相違を除いてこの機能をサポートします。Oracle では、キーワード FORMAT ではなくカンマを使用します。

SQL/CLIに対するOracleの準拠

OracleのODBCドライバは、SQL/CLIに準拠しています。

SQL/PSMに対するOracleの準拠

OracleのPL/SQLは、キーワードの綴りや構成などの構文上の小さな違いはありますが、SQL/PSMと同等の機能を提供します。

SQL/MEDに対するOracleの準拠

OracleはSQL/MEDに準拠しません。

SQL/OLBに対するOracleの準拠

Oracle SQLJは、SQL/OLB:1999に準拠していますが、SQL/OLB:2016にはまだ準拠していません。

SQL/JRTに対するOracleの準拠

Oracleは、Java(TM)で実装されるストアド・ルーチンおよびSQL型を完全にサポートしています。Oracleは、この型およびプロシージャの作成とメンテナンスについて同等のサポートを提供します。一般に、Oracleの機能は、規格で定義された機能にさらに多くの内容を盛り込んでいます。

SQL/XMLに対するOracleの準拠

規格のXMLデータ型はXMLです。Oracleの同等のデータ型は、XMLTypeです。Oracleと規格の相違点がデータ型の名前の綴りのみである場合、規格の機能はOracleで完全にサポートされているとみなします。

表C-3に、SQL/XMLの機能に対するOracleのサポートを示します。

表C-3 SQL/XMLの機能に対するOracleのサポート

機能識別子、機能	サポート
X010、XML 型	この機能を完全にサポートします。
X011、XML 型の配列	Oracle は、名前付きの配列型を使用してこの機能をサポートします。
X012、XML 型の多重集合	Oracle で XML 型の多重集合に相当するものは、XML 型の単一列を持つネストした表です。
X013、固有の XML 型	固有型は、1 つの属性のみを持つオブジェクト型を使用してエミュレートできます。
X014、XML 型の属性	Oracle では、オブジェクト型の属性は XMLType 型にできますが、この構文はオブジェクト型の作成には標準ではありません。
X015、XML 型のフィールド	Oracle のオブジェクト型は行型のかわりに使用できます。Oracle は、属性 XMLType を持つオブジェクト型をサポートしています。
X016、永続 XML 値	この機能を完全にサポートします。
X020、XMLConcat	この機能を完全にサポートします。
X025、XMLCast	Oracle はこの機能をサポートしますが、次の制限事項があります。 ● ソース式は XMLType にする必要がありますが、ターゲット・データ型は XMLType にする必要がありません。(Oracle は XML 型を 1 つしか持たないため、XML から XML にキャストする必要はありません。) ● Oracle は、<XML 受渡しメカニズム>をサポートしていません。この動作は、規格の BY VALUE と同じです。 Oracle は、REF XMLTYPE 型にキャストできるように、この機能を拡張しています。
X031、XMLElement	この機能を完全にサポートします。

機能識別子、機能	サポート
X032、XMLForest	この機能を完全にサポートします。
X034、XMLAgg	この機能を完全にサポートします。
X035、XMLAgg の ORDER BY オプション	この機能を完全にサポートします。
X036、XMLComment	この機能を完全にサポートします。
X036、XMLPi	この機能を完全にサポートします。
X038、XMLText	Oracle の XMLCData ファンクションは、テキスト・ノードの作成に使用できます。
X040、表の基本マッピング	Oracle の表のマッピングは、Java インタフェースおよびパッケージを介して実行できます。Oracle の表のマッピングは、表のみでなく問合せもマップするために汎用化されています。表のみをマップするには、SELECT * FROM table_name を指定します。これによって、この機能のうち、次の要素をサポートします。 ● X041、表の基本マッピング(NULL なし) ● X042、表の基本マッピング(NULL を NIL としてマッピング) ● X043、表の基本マッピング(表をフォレストとしてマッピング) ● X044、表の基本マッピング(表を要素としてマッピング) ● X045、表の基本マッピング(ターゲット・ネームスペースを含む) ● X046、表の基本マッピング(データのマッピング) ● X047、表の基本マッピング(メタデータのマッピング) ● X049、表の基本マッピング(16 進エンコーディング) Oracle は、この機能の次の要素をサポートしません。 ● X048、表の基本マッピング(BASE64 エンコーディング)
X041、表の基本マッピング(NULL なし)	X040 を参照してください。

機能識別子、機能	サポート
X042、表の基本マッピング(NULL を NIL としてマッピング)	X040 を参照してください。
X043、表の基本マッピング(表をフォレストとしてマッピング)	X040 を参照してください。
X044、表の基本マッピング(表を要素としてマッピング)	X040 を参照してください。
X045、表の基本マッピング(ターゲット・ネームスペースを含む)	X040 を参照してください。
X046、表の基本マッピング(データのマッピング)	X040 を参照してください。
X047、表の基本マッピング(メタデータのマッピング)	X040 を参照してください。
X049、表の基本マッピング(16 進エンコーディング)	X040 を参照してください。
X060、XMLParse(文字列の入力および CONTENT オプション)	Oracle は、{PRESERVE | STRIP} WHITESPACE 構文をサポートしません。動作は常に STRIP WHITESPACE になります。
X061、XMLParse(文字列の入力および DOCUMENT オプション)	Oracle は、{PRESERVE | STRIP} WHITESPACE 構文をサポートしません。動作は常に STRIP WHITESPACE になります。
X069、XMLSERIALIZE: INDENT	Oracle は、インデント・サイズを指定できるように、この機能を拡張しています。
X070、XMLSerialize(文字列のシリアライズおよび CONTENT オプション)	Oracle はこの機能をサポートしますが、次の制限があります。 ● 規格では、DOCUMENT または CONTENT の選択はオプションですが、Oracle では、どちらかを指定する必要があります。 Oracle はこの機能を拡張しています。規格ではターゲット・データ型が必要ですが、Oracle では CLOB にデフォルト設定します。

機能識別子、機能	サポート
X071、XMLSerialize(文字列のシリアライズおよび DOCUMENT オプション)	この機能を完全にサポートします。
X072、XMLSerialize(文字列のシリアライズ)	この機能を完全にサポートします。
X073、XMLSerialize: BLOB シリアライズ、および CONTENT オプション	この機能を完全にサポートします。
X074、XMLSerialize: BLOB シリアライズ、および DOCUMENT オプション	この機能を完全にサポートします。
X075、XMLSerialize: BLOB シリアライズ	この機能を完全にサポートします。
X076、XMLSerialize の VERSION オプション	シリアライズの前に XML バージョンを設定するには、XMLRoot を使用します。
X077、XMLSerialize: 明示的な ENCODING オプション	この機能を完全にサポートします。
X080、XML 公開時のネームスペース	XMLElement の Oracle 実装では、XMLAttributes を使用してネームスペースが定義されます(XMLNamespaces は実装されません)。ただし、XMLAttributes は、XMLForest ではサポートされません。
X086、XMLTable での XML ネームスペースの宣言	この機能を完全にサポートします。
X090、XML 文書の述語	Oracle では、ISFRAGMENT メソッドを使用して XML 値が文書かどうかをテストできます。
X096、XMLExists	Oracle はこの機能を完全にサポートします。ただし、Oracle は値渡しのみをサポートするため、PASSING 句の先頭でのキーワード BY VALUE はオプションであり、個別の引数はサポートしていません。
X120、SQL ルーチンでの XML パラメータ	この機能を完全にサポートします。

機能識別子、機能	サポート
X121、外部ルーチンでの XML パラメータ	Oracle は、規格以外のインタフェースを使用して外部ルーチンに渡される XML 値をサポートします。
X141、IS VALID 述語(データ駆動形式)	XMLISVALID メソッドは IS VALID 述語と同等で、データ駆動形式をサポートします。
X142、IS VALID 述語(ACCORDING TO 句)	XMLISVALID メソッドは IS VALID 述語と同等で、ACCORDING TO 句に相当するものを含みます。
X143、IS VALID 述語(ELEMENT 句)	XMLISVALID メソッドは IS VALID 述語と同等で、ELEMENT 句に相当するものを含みます。
X144、IS VALID 述語(スキーマの場所)	XMLISVALID メソッドは IS VALID 述語と同等で、登録済 XML スキーマのスキーマの場所の指定をサポートします。
X145、IS VALID 述語(CHECK 制約の外部)	XMLISVALID メソッドは IS VALID 述語と同等で、CHECK 制約の外部で使用できます。
X151、DOCUMENT オプション付きの IS VALID 述語	XMLISVALID メソッドは IS VALID 述語と同等で、DOCUMENT 句と同等の検証を実行します。(XMLISVALID は内容の検証をサポートしません。)
X156、IS VALID 述語(ELEMENT 句の付いたオプションの NAMESPACE)	XMLISVALID メソッドは IS VALID 述語と同等で、ネームスペースの要素の検証に使用できます。
X157、IS VALID 述語(ELEMENT 句の付いた NO NAMESPACE)	XMLISVALID メソッドは IS VALID 述語と同等で、名前のないネームスペースの要素の検証に使用できます。
X160、登録済 XML スキーマの基本情報スキーマ	Oracle の静的データ・ディクショナリ・ビューALL_XML_SCHEMAS は、現行のユーザーがアクセスできる登録済の XML スキーマのリストを提供します。ALL_XML_SCHEMAS.SCHEMA_URL 列は、規格の XML_SCHEMAS.XML_SCHEMA_LOCATION 列に対応します。登録済 XML スキーマのターゲット・ネームスペースは、ALL_XML_SCHEMAS.SCHEMA を調べることによって確認できます。Oracle は、規格の XML_SCHEMAS の他の列に相当する列を持ちません。
X161、登録済 XML スキーマの拡張情報スキーマ	Oracle は、規格の XML_SCHEMA_NAMESPACES および XML_SCHEMA_ELEMENTS に対応する静的データ・ディクショナリ・ビューを持ちません。ただし、登録済 XML スキーマに関するすべての情報は、ALL_XML_SCHEMAS.SCHEMA 列にある実際の XML スキーマを調べることによっ

機能識別子、機能	サポート
	て確認できます。これを調べると、登録済 XML スキーマが非決定的であるかどうかも確認できます。また、非決定的なネームスペースと要素も確認できます。
X191、XML(DOCUMENT(XMLSCHEMA))型	Oracle は、この構文をサポートしません。ただし、表の列は登録済 XML スキーマで制約できます。この場合、列のすべての値は XML(DOCUMENT(XMLSCHEMA)) 型になります。
X200、XMLQuery	Oracle は、この機能の次の要素を完全にサポートします。 ● X201、XMLQuery: RETURNING CONTENT ● X203、XMLQuery: コンテキスト項目の受渡し ● X204、XMLQuery: XQuery 変数の初期化 ● X206、XMLQuery: NULL ON EMPTY オプション Oracle は値渡しのみをサポートするため、PASSING 句の先頭でのキーワード BY VALUE はオプションであり、個別の引数はサポートしていません。
X201、XMLQuery: RETURNING CONTENT	X200 を参照してください。
X203、XMLQuery: コンテキスト項目の受渡し	X200 を参照してください。
X204、XMLQuery: XQuery 変数の初期化	X200 を参照してください。
X206、XMLQuery: NULL ON EMPTY オプション	X200 を参照してください。
X221、XML 受渡しメカニズム BY VALUE	Oracle は、XMLQuery、XMLTable および XMLExists での BY VALUE 句をサポートしています。これに該当する BY VALUE は、オプションの構文として引数リストの先頭でサポートされますが、個別の引数や列の修飾子としてはサポートされません。
X232、XML(CONTENT(ANY))型	Oracle は、この構文を型の修飾子としてサポートしませんが、Oracle の XMLType は、一時値でこのデータ型をサポートします。永続値は、

機能識別子、機能	サポート
	XML(DOCUMENT(ANY))型になります。この型は、XML(CONTENT(ANY))のサブセットです。
X241、XML 文書作成時の RETURNING CONTENT	Oracle は、この構文をサポートしません。Oracle では、文書を作成するファンクション(XMLAgg、XMLComment、XMLConcat、XMLElement、XMLForest および XMLPi)の動作は、常に RETURNING CONTENT になります。
X251、XML(DOCUMENT(UNTYPED))型の永続 XML 値	この機能を完全にサポートします。
X252、XML(DOCUMENT(ANY))型の永続値	この機能を完全にサポートします。
X256、XML(DOCUMENT(XMLSCHEMA))型の永続値	この機能を完全にサポートします。
X260、XML 型(ELEMENT 句)	Oracle は、この構文をサポートしません。ただし、表の列は登録済 XML スキーマの最上位の要素で制約できます。
X263、XML 型(ELEMENT 句の付いた NO NAMESPACE)	Oracle は、この構文をサポートしません。ただし、表の列は登録済 XML スキーマの名前のないネームスペースにある最上位の要素で制約できます。
X264、XML 型(スキーマの場所)	Oracle は、この構文をサポートしません。ただし、表の列はスキーマの場所で識別される登録済 XML スキーマで制約できます。
X271、XMLValidate(データ駆動形式)	SCHEMAVALIDATE メソッドは XMLValidate と同等で、データ駆動形式をサポートします。
X272、XMLValidate(ACCORDING TO 句)	SCHEMAVALIDATE メソッドは XMLValidate と同等で、特定の登録済 XML スキーマの指定に使用できます。
X273、XMLValidate(ELEMENT 句)	SCHEMAVALIDATE メソッドは XMLValidate と同等で、特定の登録済 XML スキーマの特定要素を指定するために使用できます。
X274、XMLValidate(スキーマの場所)	SCHEMAVALIDATE メソッドは XMLValidate と同等で、スキーマの場所を示す URL で特定の登録済 XML スキーマを指定するために使用できます。

機能識別子、機能	サポート
X281、DOCUMENT オプションの付いた XMLValidate	SCHEMAVALIDATE メソッドは XMLValidate と同等です。 SCHEMAVALIDATE は XML 文書の検証のみを実行します(内容の検証は実行しません)。
X286、XMLValidate(ELEMENT 句の付いた NO NAMESPACE)	SCHEMAVALIDATE メソッドは XMLValidate と同等で、特定の登録済 XML スキーマの名前のないネームスペースにある特定要素を指定するために使用できます。
X300、XMLTable	Oracle は、列パス式の逆軸をサポートしません。この制限事項を除き、Oracle は、この機能の次の要素を完全にサポートします。 ● X086、XMLTable での XML ネームスペースの宣言 ● X302、順序性列付きの XMLTable ● X303、XMLTable(列のデフォルト・オプション) ● X304、XMLTable(コンテキスト項目の受渡し) ● X305、XMLTable(XQuery 変数の初期化) Oracle は値渡しのみをサポートするため、PASSING 句の先頭でのキーワード BY VALUE はオプションであり、個別の引数はサポートしていません。
X302、順序性列付きの XMLTable	X300 を参照してください。
X303、XMLTable(列のデフォルト・オプション)	X300 を参照してください。
X304、XMLTable(コンテキスト項目の受渡し)	X300 を参照してください。
X305、XMLTable(XQuery 変数の初期化)	X300 を参照してください。

FIPS 127-2に対するOracleの準拠

Oracleは、最新のFIPS(Federal Information Processing Standard)であるFIPS PUB 127-2に完全に準拠しています。現在、この規格は、公開されていません。ただし、FIPS 127-2で定義されたデータベース要素のサイズに関する情報に依存するアプリケーションを使用するユーザーのために、準拠性についての詳細を表C-4に示します。

表C-4 データベース要素のサイズ設定

データベース要素	FIPS	Oracle Database
識別子の長さ(バイト単位)	18	128
CHARACTER データ型の長さ(バイト単位)	240	2,000
NUMERIC データ型の 10 進精度	15	38
DECIMAL データ型の 10 進精度	15	38
INTEGER データ型の 10 進精度	9	38
SMALLINT データ型の 10 進精度	4	38
FLOAT データ型の 2 進精度	20	126
REAL データ型の 2 進精度	20	63
DOUBLE PRECISION データ型の 2 進精度	30	126
表の中の列	100	1,000
INSERT 文の中の値	100	1,000
UPDATE 文内の SET 句(ノート 1)	20	1,000
行の長さ(ノート 2、ノート 3)	2,000	2,000,000
一意制約の中の列	6	32
一意制約の長さ(ノート 2)	120	(ノート 4)
外部キー列リストの長さ(ノート 2)	120	(ノート 4)

データベース要素	FIPS	Oracle Database
GROUP BY 句の中の列	6	255 (ノート 5)
GROUP BY 列のリストの長さ	120	(ノート 5)
ORDER BY 句の中のソート指定	6	255 (ノート 5)
ORDER BY 列のリストの長さ	120	(ノート 5)
参照整合性制約内の列	6	32
SQL 文で参照される表	15	無制限
同時にオープンできるカーソル	10	(ノート 6)
SELECT 構文のリストの項目	100	1,000

ノート1: UPDATE文のSET句の数とは、SETキーワードの後に続くカンマで区切られる項目の数のことです。

ノート2: FIPS PUBでは、列セットの長さを次の値の合計として規定しています。つまり、列の数を2倍した値、各文字列の長さ(バイト単位)、各真数値列の10進精度に1を加えた値、各概数値列の2進精度を4で割って1を加えた値の合計です。

ノート3: 行の最大長に対するOracleの制限は、長さ2GBのLONG値とそれぞれの長さが4000バイトである999のVARCHAR2値を含む行の最大長に基づいています。2(254) + 231 + (999(4000))

ノート4: 一意キー制約に対するOracleの制限は、Oracleデータ・ブロックのサイズ(初期化パラメータDB_BLOCK_SIZEによって指定される)の半分からオーバーヘッドを引いたものになります。

ノート5: Oracleは、GROUP BY句の列数やORDER BY句のソート指定の数に対して制限を設定しません。ただし、GROUP BY句やORDER BY句のすべての式のサイズの合計は、Oracleデータ・ブロックのサイズ(初期化パラメータDB_BLOCK_SIZEによって指定される)からオーバーヘッドを引いたサイズに制限されています。

ノート6: 同時にオープンできるカーソルの数に対するOracleの制限は、初期化パラメータOPEN_CURSORSによって指定されます。このパラメータの最大値は、使用しているオペレーティング・システムで使用可能なメモリーによって異なりますが、どんな場合でも100を超えます。

標準SQLに対するOracle拡張機能

Oracleでは、標準SQL以外でも様々な機能をサポートしています。他のSQL処理系へのアプリケーションの移植性を考慮する場合、OracleのFIPSフラガーを使用して、埋込みSQLプログラムでのEntry SQL-92にOracleの拡張機能を位置付けてください。FIPSフラガーは、OracleプリコンパイラとSQL*Moduleコンパイラの一部です。`ALTER SESSION SET FLAGGER = ENTRY`を使用すると、SQL*PlusでもFIPSフラガーを有効にできます。SQL:2016はSQL-92よりも優先されますが、SQL-92以降のどのバージョンのSQLにも規格準拠性の検証機関はありません。このため、Entry SQL-92が最も確実な移植性を保証します。

関連項目:

FIPSフラガーの使用方法の詳細は、『Pro*COBOLプログラマーズ・ガイド』および『Pro*C/C++プログラマーズ・ガイド』を参照してください。

以前の規格に対するOracleの準拠

今回のリリースのOracle Databaseは、この付録の前の項で説明したように、このガイドが公開されたときの最新版のSQL規格であるSQL:2016に準拠しています。SQL-92(特にSQL-92のエントリ・レベル)またはSQL:1999は、SQL:2016に置き換えられているため、Oracleは、データベースの今回のリリースがこれらの以前の規格に準拠していることを正式に明示していません。SQL規格の各エディション間にある変更(ほとんどは小さな変更)が、アプリケーションに影響することもあります。影響のある非互換の詳細は、SQL規格またはこの規格に関する説明資料を参照してください。重要な情報源の1つとして、SQL/Foundation:1999、SQL/Foundation:2003、SQL/Foundation:2008、SQL/Foundation:2011およびSQL/Foundation:2016のAnnex Eがあります。

今回のリリースのOracle Databaseでは、以前の版のSQL構造を引き続き使用できる場合があります。このようなサポートの多くは、ベンダーによる妥当な拡張機能として容認されています。これは、データベースのバージョン間の非互換性を最小限にするためのOracleの一般的な方針です。この方針に基づき、以前の形式が適している場合は、引き続き保持されます。いずれにしても、以前のSQLとSQL:2016の間の差異(前述)はあまり重要ではありません。

文字セット・サポート

ほとんどの各国語文字セット、国際文字セットおよびベンダー固有文字セットの標準がサポートされています。Oracleがサポートする文字セットの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

Unicodeは、エンコードされたユニバーサル文字セットで、単一文字セットを使用したすべての言語の情報を格納できます。Unicodeは、XML、Java、JavaScript、LDAPなどの最新の規格で必要です。Unicodeは、ISO/IEC規格10646に準拠しています。ISO規格についての情報は、国際標準化機構のWebサイトを参照してください。

```
http://www.iso.ch/
```

Oracle Database 18cは、Unicode規格のバージョン9.0に準拠しています。Unicode規格の最新情報については、次のUnicode ConsortiumのWebサイトを参照してください。

```
http://www.unicode.org
```

Oracleは、Unicode規格のUTF-8コード体系をサポートするためにAL32UTF8文字セットを使用し、UTF-16BEコード体系をサポートするためにAL16UTF16文字セットを使用し、UTF-16LEコード体系をサポートするためにAL16UTF16LE文字セットを使用します。AL32UTF8は、ASCIIベースのプラットフォーム上のクライアントとデータベースの文字セットとして有効です。AL16UTF16は、あらゆるプラットフォーム上の各国語(NCHAR)文字セットとして有効です。AL16UTF16LEは、クライアント、データベースの文字セット、または各国語文字セットとして有効ではありません。

Oracleは、非推奨のUnicode互換エンコード形式であるCESU-8(UTF8文字セットを使用)と、UTF-EBCDIC(UTFE文字セットを使用)を実装しています。UTF8とUTFEの文字セットは、Unicode規格バージョン3.0より以降の更新に含まれる保証はありません。UTF8は、ASCIIベースのプラットフォーム上のクライアントとデータベースの文字セットとして有効です。また、すべてのプラットフォーム上の各国語(NCHAR)文字セットとしても有効です。UTFEは、EBCDICベースのプラットフォーム上のデータベース文字セットとして有効です。

上記のOracle文字セットは、すべて変換ファンクションでサポートされます。

ASCIIベースのプラットフォーム上のデータベースは、AL32UTF8文字セットと、AL16UTF16各国語(NCHAR)文字セットを使用して作成することをお薦めします。NCHARデータ型と、それに関連する各国語文字セットは、一部のRDBMSコンポーネント(Oracle TextやOracle XDBなど)ではサポートされていないため、使用しないことをお薦めします。

関連項目:

Oracleの文字セットのサポートの詳細は、『Oracle Databaseグローバリゼーション・サポート・ガイド』を参照してください。

D Oracleの正規表現のサポート

Oracleによる正規表現の実装は、IEEEのPortable Operating System Interface(POSIX)正規表現規格およびUnicode ConsortiumのUnicode Regular Expression Guidelinesに準拠します。

この付録の内容は次のとおりです。

多言語の正規表現の構文

表D-1に、POSIX規格のExtended Regular Expression(ERE)の構文に定義されているすべての演算子を示します。Oracleは、POSIX規格にASCII(英語)データの検索用に定義されているこれらの演算子の完全な構文および検索セマンティクスに従います。演算子とその使用例、およびそれらの演算子のOracle多言語拡張の詳細は、『Oracle Database開発ガイド』を参照してください。表の後に示すノートでは、演算子とそれらの機能、およびそれらの演算子のOracle多言語拡張の詳細を示します。表D-2に、POSIX演算子のOracleのサポートおよび多言語拡張を示します。

表D-1 正規表現の演算子およびメタ記号

演算子	説明
¥	バックスラッシュは、そのコンテキストに応じて、次の 4 つの異なる意味を持つことができます。持つことができる意味は次のとおりです。 ● バックスラッシュ自体 ● 次の文字の引用 ● 演算子の導入 ● 意味なし
*	0(ゼロ)回以上の繰返しに一致します。
+	1 回以上の繰返しに一致します。
?	0 回または 1 回の繰返しに一致します。
|	代替一致を指定する代替演算子です。
^	デフォルトでは文字列の先頭に一致します。複数行モードでは、ソース文字列内の任意の行の先頭に一致します。
$	デフォルトでは文字列の末尾に一致します。複数行モードでは、ソース文字列内の任意の行の末尾に一致します。
.	サポートされる文字セットの任意の文字(NULL を除く)に一致します。
[ ]	大カッコで囲んだ一致リストのいずれかの正規表現に一致します。非一致リストの正規表現はキャレット(^)で始まり、リストに示されている正規表現以外の任意の文字に一致するリストを指定します。

演算子	説明
	大カッコの正規表現に右大カッコ(])を指定するには、リストの先頭(キャレット(^)が最初の文字である場合はその後)に配置します。 大カッコの正規表現にハイフンを指定するには、リストの先頭(キャレット(^)が最初の文字である場合はその後)、リストの最後または範囲を表す正規表現の場合は範囲の最後を表すポイントとして配置します。
( )	グループ化の正規表現で、1 つの部分正規表現として処理されます。
{m}	厳密に m 回の繰返しに一致します。
{m,}	m 回以上の繰返しに一致します。
{m,n}	m 回以上 n 回以下の繰返しに一致します。
¥n	後方参照表現(n は 1 から 9 の数字)は、¥n の前のカッコで囲まれた n 番目の部分正規表現に一致します。
[..]	1 つの照合要素を指定します。複数文字要素を指定できます(スペイン語の[.ch.]など)。
[: :]	文字クラスを指定します([:alpha:]など)。文字クラス内の任意の文字に一致します。
[==]	等価クラスを指定します。たとえば、[=a=]は基本文字「a」のすべての文字に一致します。

正規表現の演算子の多言語拡張

OracleによるPOSIX演算子の実装は、多言語データに適用される場合、POSIX規格に指定されている一致機能を拡張します。表D-2に、POSIX規格のコンテキストの演算子の関係を示します。

- 1列目に、サポートされる演算子を示します。
- 2列目と3列目に、POSIX規格(Basic Regular Expression(BRE)およびExtended Regular Expression(ERE))でその演算子が定義されているかどうかを示します。
- 4列目に、Oracleによる実装で、多言語データを処理するためにその演算子のセマンティクスが拡張されているかどうかを示します。

Oracleでは、マルチバイト文字を直接入力する手段があるか、またはファンクションを使用してマルチバイト文字を構成できる場合は、マルチバイト文字を直接入力できます。書式「¥xxxx」のUnicodeエンコーディング値の16進数値は使用できません。Oracleは、文字の図形的な表現ではなく、文字のエンコードに使用されるバイト値に基づいて文字を評価します。アクセント記号のあるすべての文字は、単語構成文字とみなされます。

表D-2 POSIXと多言語の演算子の関係

演算子	POSIX BRE構文	POSIX ERE構文	多言語拡張機能
¥	可	可	—
*	可	可	—
+	--	可	—
?	—	可	—
|	—	可	—
^	可	可	可
$	可	可	可
.	可	可	可
[ ]	可	可	可
( )	可	可	—
{m}	可	可	—

演算子	POSIX BRE構文	POSIX ERE構文	多言語拡張機能
{m,}	可	可	—
{m,n}	可	可	—
¥n	可	可	可
[..]	可	可	可
[::]	可	可	可
[==]	可	可	可

Oracleの正規表現におけるPerlによる拡張機能

Oracle Databaseの正規表現ファンクションおよびその条件では、一般的に使用されるPerlによる演算子の多くが使用可能です。ただし、POSIX規格の一部は使用できません。表D-3に、使用可能な演算子を示します。詳細とその例については、『Oracle Database開発ガイド』を参照してください。

表D-3 Oracleの正規表現におけるPerlによる演算子

演算子	説明
¥d	数字です。
¥D	数字以外です。
¥w	単語構成文字です。
¥W	単語構成文字以外です。
¥s	空白文字です。
¥S	空白文字以外です。
¥A	文字列の先頭、または文字列末尾の改行文字の直前のみに一致します。
¥Z	文字列の末尾のみに一致します。
*?	先行のパターン要素に 0 回以上一致します(最短マッチ)。
+?	先行のパターン要素に 1 回以上一致します(最短マッチ)。
??	先行のパターン要素に 0 回または 1 回一致します(最短マッチ)。
{n}?	先行のパターン要素に n 回一致します(最短マッチ)。
{n,}?	先行のパターン要素に n 回以上一致します(最短マッチ)。
{n,m}?	先行のパターン要素に n 回以上 m 回以下一致します(最短マッチ)。

E Oracle SQLの予約語とキーワード

この付録の内容は次のとおりです。

Oracle SQLの予約語

この項では、Oracle SQLの予約語を示します。Oracle SQLの予約語は非引用識別子として使用できません。引用識別子には、予約語を使用できますが、お薦めしません。

> ノート:
>
> Oracleでは、次の表に示す予約語の他に、暗黙的に生成されるスキーマ・オブジェクトとサブオブジェクトには「SYS_」で始まるシステム生成名を使用します。名前解決での競合を避けるため、この接頭辞は明示的に指定するスキーマ・オブジェクト名やサブオブジェクト名では使用しないでください。

V$RESERVED_WORDSデータ・ディクショナリ・ビューには、各予約語に関する追加情報が表示されます。この情報には、各予約語が常に予約されているか、特定の使用に対してのみ予約されているかの情報も含まれます。詳細は、『Oracle Databaseリファレンス』を参照してください。

アスタリスク(*)が付いた語は、ANSIの予約語でもあります。

- ACCESS
- ADD
- ALL *
- ALTER *
- AND *
- ANY *
- AS *
- ASC
- AUDIT
- BETWEEN *
- BY *
- CHAR *
- CHECK *
- CLUSTER
- COLUMN *
- COLUMN_VALUE (このリストの後の「ノート1」を参照)
- COMMENT
- COMPRESS
- CONNECT *
- CREATE *
- CURRENT *
- DATE *
- DECIMAL *
- DEFAULT *
- DELETE *
- DESC
- DISTINCT *
- DROP *
- ELSE *
- EXCLUSIVE
- EXISTS *
- FILE
- FLOAT *
- FOR *
- FROM *
- GRANT *
- GROUP *

- HAVING *
- IDENTIFIED
- IMMEDIATE
- IN *
- INCREMENT
- INDEX
- INITIAL
- INSERT *
- INTEGER *
- INTERSECT *
- INTO *
- IS *
- LEVEL
- LIKE *
- LOCK
- LONG
- MAXEXTENTS
- MINUS
- MLSLABEL
- MODE
- MODIFY
- NESTED_TABLE_ID (このリストの後の「ノート1」を参照)
- NOAUDIT
- NOCOMPRESS
- NOT *
- NOWAIT
- NULL *
- NUMBER
- OF *
- OFFLINE
- ON *
- ONLINE
- OPTION
- OR *
- ORDER *
- PCTFREE
- PRIOR
- PUBLIC
- RAW
- RENAME
- RESOURCE
- REVOKE *
- ROW *
- ROWID (このリストの後の「ノート2」を参照)
- ROWNUM
- ROWS *
- SELECT *
- SESSION
- SET *
- SHARE
- SIZE
- SMALLINT *
- START *
- SUCCESSFUL
- SYNONYM
- SYSDATE
- TABLE *
- THEN *

- TO *
- TRIGGER *
- UID
- UNION *
- UNIQUE *
- UPDATE *
- USER *
- VALIDATE
- VALUES *
- VARCHAR *
- VARCHAR2
- VIEW
- WHENEVER *
- WHERE *
- WITH *
-

ノート1: このキーワードは、属性名として使用するためにのみ予約されています。

ノート2: 引用識別子または非引用識別子のいずれであっても、大文字のROWIDを列名として使用することはできません。ただし、列名ではない引用識別子には大文字を使用できます。また列名を含む引用識別子には、1つ以上の小文字(たとえば、"Rowid"や"rowid")を使用できます。

Oracle SQLキーワード

Oracle SQLキーワードは予約されていません。ただし、Oracleは固有の方法でこれらの文字を内部的に使用します。したがって、これらの文字をオブジェクトおよびオブジェクトの部分の名前として使用した場合、使用しているSQL文が読みにくくなり、予期しない結果になることがあります。

V$RESERVED_WORDSデータ・ディクショナリ・ビューを問い合せることによって、キーワードのリストを取得できます。ビュー内のキーワードのうち、常に予約されているものや特定の使用に対して予約されているものとして示されていないキーワードは、すべてOracle SQLキーワードです。詳細は、『Oracle Databaseリファレンス』を参照してください。

F 詳細な例

SQL言語リファレンスの本文では、ほぼすべてのリファレンス項目に例を記載しています。この付録では、単一のSQL文のコンテキストに適さない詳細な例を示します。これらの例は、特定のOracle機能を活用するために使用する一連のステップを示すためのものです。このリファレンス本文の個々のSQL文の構文図およびセマンティクスにかわるものではありません。記載されている参照項目を使用して、必要な権限、制限事項、構文などの補足情報を参照してください。

この付録の内容は次のとおりです。

- 拡張索引作成機能の使用方法
- SQL文でのXMLの使用方法

拡張索引作成機能の使用方法

この項では、拡張索引作成機能の単純で現実的な使用例で必要となるステップの例を示します。

HR.employees表の給与をランク付けし、10から20にランク付けされる給与を検索するとします。この場合、次のようにDENSE_RANKファンクションを使用できます。

```
SELECT last_name, salary FROM
   (SELECT last_name, DENSE_RANK() OVER
      (ORDER BY salary DESC) rank_val, salary FROM employees)
   WHERE rank_val BETWEEN 10 AND 20;
```

関連項目:

DENSE_RANK

このネストした問合せは多少複雑で、employees表のソートのみではなく全体スキャンも必要です。同じ結果が得られるもう1つの方法には、拡張索引作成機能を使用する方法があります。この方法による問合せは単純になります。この問合せには、ROWIDによる索引スキャンおよび表アクセスのみ必要です。そのため、問合せが効率的に実行されます。

最初のステップでは、position_im実装タイプ(索引の定義、メンテナンスおよび作成を行うためのメソッド・ヘッダーを含む)を作成します。型本体のほとんどにはPL/SQLが使用されています。その部分はイタリック体で示しています。

ファンクションODCIINDEXCREATE()内にEXECUTE IMMEDIATE文が含まれるため、型は、AUTHID CURRENT_USER句を使用して作成する必要があります。デフォルトでは、このファンクションは定義者権限で実行されます。後続のドメイン索引の作成でこのファンクションがコールされる場合、実行者の権限は定義者権限とは異なります。

関連項目:

- 「CREATE TYPE」および「CREATE TYPE BODY」を参照してください。
- この文のODCIルーチンの詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。

```
CREATE OR REPLACE TYPE position_im AUTHID CURRENT_USER AS OBJECT

(
  curnum  NUMBER,
  howmany NUMBER,
  lower_bound NUMBER,
  upper_bound NUMBER,
/* lower_bound and upper_bound are used for the
index-based functional implementation */
  STATIC FUNCTION ODCIGETINTERFACES(ifclist OUT SYS.ODCIOBJECTLIST) RETURN NUMBER,
  STATIC FUNCTION ODCIINDEXCREATE
    (ia SYS.ODCIINDEXINFO, parms VARCHAR2, env SYS.ODCIEnv) RETURN NUMBER,
  STATIC FUNCTION ODCIINDEXTRUNCATE (ia SYS.ODCIINDEXINFO,
                                     env SYS.ODCIEnv) RETURN NUMBER,
  STATIC FUNCTION ODCIINDEXDROP(ia SYS.ODCIINDEXINFO,
                                env SYS.ODCIEnv) RETURN NUMBER,
  STATIC FUNCTION ODCIINDEXINSERT(ia SYS.ODCIINDEXINFO, rid ROWID,
                                  newval NUMBER, env SYS.ODCIEnv) RETURN NUMBER,
  STATIC FUNCTION ODCIINDEXDELETE(ia SYS.ODCIINDEXINFO, rid ROWID, oldval NUMBER,
                                  env SYS.ODCIEnv) RETURN NUMBER,
  STATIC FUNCTION ODCIINDEXUPDATE(ia SYS.ODCIINDEXINFO, rid ROWID, oldval NUMBER,
                                  newval NUMBER, env SYS.ODCIEnv) RETURN NUMBER,
  STATIC FUNCTION ODCIINDEXSTART(SCTX IN OUT position_im, ia SYS.ODCIINDEXINFO,
                                 op SYS.ODCIPREDINFO, qi SYS.ODCIQUERYINFO,
```

```
                                  strt NUMBER, stop NUMBER, lower_pos NUMBER,
                                  upper_pos NUMBER, env SYS.ODCIEnv) RETURN NUMBER,
  MEMBER FUNCTION ODCIINDEXFETCH(SELF IN OUT position_im, nrows NUMBER,
                                  rids OUT SYS.ODCIRIDLIST, env SYS.ODCIEnv)
                                  RETURN NUMBER,
  MEMBER FUNCTION ODCIINDEXCLOSE(env SYS.ODCIEnv) RETURN NUMBER
);
/
CREATE OR REPLACE TYPE BODY position_im
IS
   STATIC FUNCTION ODCIGETINTERFACES(ifclist OUT SYS.ODCIOBJECTLIST)
       RETURN NUMBER IS
   BEGIN
       ifclist := SYS.ODCIOBJECTLIST(SYS.ODCIOBJECT('SYS','ODCIINDEX2'));
       RETURN ODCICONST.SUCCESS;
   END ODCIGETINTERFACES;
 STATIC FUNCTION ODCIINDEXCREATE (ia SYS.ODCIINDEXINFO, parms VARCHAR2, env
SYS.ODCIEnv) RETURN
 NUMBER
  IS
   stmt   VARCHAR2(2000);
  BEGIN
/* Construct the SQL statement */
   stmt := 'Create Table ' || ia.INDEXSCHEMA || '.' || ia.INDEXNAME ||
           '_STORAGE_TAB' || '(col_val, base_rowid, constraint pk PRIMARY KEY ' ||
           '(col_val, base_rowid)) ORGANIZATION INDEX AS SELECT ' ||
           ia.INDEXCOLS(1).COLNAME || ', ROWID FROM ' ||
           ia.INDEXCOLS(1).TABLESCHEMA || '.' || ia.INDEXCOLS(1).TABLENAME;
   EXECUTE IMMEDIATE stmt;
   RETURN ODCICONST.SUCCESS;
  END;
  STATIC FUNCTION ODCIINDEXDROP(ia SYS.ODCIINDEXINFO, env SYS.ODCIEnv) RETURN NUMBER
IS
   stmt VARCHAR2(2000);
  BEGIN
/* Construct the SQL statement */
   stmt := 'DROP TABLE ' || ia.INDEXSCHEMA || '.' || ia.INDEXNAME ||
   '_STORAGE_TAB';
/* Execute the statement */
   EXECUTE IMMEDIATE stmt;
   RETURN ODCICONST.SUCCESS;
  END;
  STATIC FUNCTION ODCIINDEXTRUNCATE(ia SYS.ODCIINDEXINFO, env SYS.ODCIEnv) RETURN
NUMBER IS
   stmt VARCHAR2(2000);
  BEGIN
/* Construct the SQL statement */
   stmt := 'TRUNCATE TABLE ' || ia.INDEXSCHEMA || '.' || ia.INDEXNAME ||
'_STORAGE_TAB';

   EXECUTE IMMEDIATE stmt;
   RETURN ODCICONST.SUCCESS;
  END;
  STATIC FUNCTION ODCIINDEXINSERT(ia SYS.ODCIINDEXINFO, rid ROWID,
                           newval NUMBER, env SYS.ODCIEnv) RETURN NUMBER IS
   stmt VARCHAR2(2000);
  BEGIN
/* Construct the SQL statement */
   stmt := 'INSERT INTO ' || ia.INDEXSCHEMA || '.' || ia.INDEXNAME ||
          '_STORAGE_TAB  VALUES (''' || newval || ''' , ''' || rid || ''' )';
/* Execute the SQL statement */
   EXECUTE IMMEDIATE stmt;
   RETURN ODCICONST.SUCCESS;
  END;

  STATIC FUNCTION ODCIINDEXDELETE(ia SYS.ODCIINDEXINFO, rid ROWID, oldval NUMBER,
                                   env SYS.ODCIEnv)
```

```
     RETURN NUMBER IS
    stmt VARCHAR2(2000);
  BEGIN
/* Construct the SQL statement */
   stmt := 'DELETE FROM ' || ia.INDEXSCHEMA || '.' || ia.INDEXNAME ||
          '_STORAGE_TAB WHERE col_val = ''' || oldval || ''' AND base_rowid = ''' ||
rid || '''';
/* Execute the statement */
   EXECUTE IMMEDIATE stmt;
   RETURN ODCICONST.SUCCESS;
  END;
  STATIC FUNCTION ODCIINDEXUPDATE(ia SYS.ODCIINDEXINFO, rid ROWID, oldval NUMBER,
                           newval NUMBER, env SYS.ODCIEnv) RETURN NUMBER IS
   stmt VARCHAR2(2000);
  BEGIN
/* Construct the SQL statement */
   stmt := 'UPDATE ' || ia.INDEXSCHEMA || '.' || ia.INDEXNAME ||
          '_STORAGE_TAB SET col_val = ''' || newval || ''' WHERE f2 = '''|| rid
||'''';
/* Execute the statement */
   EXECUTE IMMEDIATE stmt;
   RETURN ODCICONST.SUCCESS;
  END;
  STATIC FUNCTION ODCIINDEXSTART(SCTX IN OUT position_im, ia SYS.ODCIINDEXINFO,
                         op SYS.ODCIPREDINFO, qi SYS.ODCIQUERYINFO,
                         strt NUMBER, stop NUMBER, lower_pos NUMBER,
                         upper_pos NUMBER, env SYS.ODCIEnv) RETURN NUMBER IS
    rid              VARCHAR2(5072);
    storage_tab_name VARCHAR2(65);
    lower_bound_stmt VARCHAR2(2000);
    upper_bound_stmt VARCHAR2(2000);
    range_query_stmt VARCHAR2(2000);
    lower_bound      NUMBER;
    upper_bound      NUMBER;
    cnum             INTEGER;
    nrows            INTEGER;

  BEGIN
/* Take care of some error cases.
    The only predicates in which position operator can appear are
       op() = 1     OR
       op() = 0     OR
       op() between 0 and 1
*/
    IF (((strt != 1) AND (strt != 0)) OR
        ((stop != 1) AND (stop != 0)) OR
        ((strt = 1) AND (stop = 0))) THEN
      RAISE_APPLICATION_ERROR(-20101,
                              'incorrect predicate for position_between operator');
    END IF;
    IF (lower_pos > upper_pos) THEN
      RAISE_APPLICATION_ERROR(-20101, 'Upper Position must be greater than or
      equal to Lower Position');
    END IF;
    IF (lower_pos <= 0) THEN
      RAISE_APPLICATION_ERROR(-20101, 'Both Positions must be greater than zero');
    END IF;
    storage_tab_name := ia.INDEXSCHEMA || '.' || ia.INDEXNAME ||
                        '_STORAGE_TAB';
    upper_bound_stmt := 'Select MIN(col_val) FROM (Select /*+ INDEX_DESC(' ||
                        storage_tab_name || ') */ DISTINCT ' ||
                        'col_val FROM ' || storage_tab_name || ' ORDER BY ' ||
                        'col_val DESC) WHERE rownum <= ' || lower_pos;
    EXECUTE IMMEDIATE upper_bound_stmt INTO upper_bound;
    IF (lower_pos != upper_pos) THEN
      lower_bound_stmt := 'Select MIN(col_val) FROM (Select /*+ INDEX_DESC(' ||
                          storage_tab_name || ') */ DISTINCT ' ||
```

```
                          'col_val FROM ' || storage_tab_name ||
                          ' WHERE col_val < ' || upper_bound || ' ORDER BY ' ||
                          'col_val DESC) WHERE rownum <= ' ||
                          (upper_pos - lower_pos);
      EXECUTE IMMEDIATE lower_bound_stmt INTO lower_bound;
    ELSE
      lower_bound := upper_bound;
    END IF;
    IF (lower_bound IS NULL) THEN
      lower_bound := upper_bound;
    END IF;
    range_query_stmt := 'Select base_rowid FROM ' || storage_tab_name ||
                        ' WHERE col_val BETWEEN ' || lower_bound || ' AND ' ||
                        upper_bound;
    cnum := DBMS_SQL.OPEN_CURSOR;
    DBMS_SQL.PARSE(cnum, range_query_stmt, DBMS_SQL.NATIVE);
/* set context as the cursor number */
    SCTX := position_im(cnum, 0, 0, 0);
/* return success */
    RETURN ODCICONST.SUCCESS;
  END;
  MEMBER FUNCTION ODCIINDEXFETCH(SELF IN OUT position_im, nrows NUMBER,
                                 rids OUT SYS.ODCIRIDLIST, env SYS.ODCIEnv)
   RETURN NUMBER IS
    cnum    INTEGER;
    rid_tab DBMS_SQL.Varchar2_table;
    rlist   SYS.ODCIRIDLIST := SYS.ODCIRIDLIST();
    i       INTEGER;
    d       INTEGER;
  BEGIN
    cnum := SELF.curnum;
    IF self.howmany = 0 THEN
      dbms_sql.define_array(cnum, 1, rid_tab, nrows, 1);
      d := DBMS_SQL.EXECUTE(cnum);
    END IF;
    d := DBMS_SQL.FETCH_ROWS(cnum);
    IF d = nrows THEN
      rlist.extend(d);
    ELSE
      rlist.extend(d+1);
    END IF;
    DBMS_SQL.COLUMN_VALUE(cnum, 1, rid_tab);
    for i in 1..d loop
      rlist(i) := rid_tab(i+SELF.howmany);
    end loop;
    SELF.howmany := SELF.howmany + d;
    rids := rlist;
    RETURN ODCICONST.SUCCESS;
  END;
  MEMBER FUNCTION ODCIINDEXCLOSE(env SYS.ODCIEnv) RETURN NUMBER IS
    cnum INTEGER;
  BEGIN
    cnum := SELF.curnum;
    DBMS_SQL.CLOSE_CURSOR(cnum);
    RETURN ODCICONST.SUCCESS;
  END;
END;
/
```

次のステップでは、索引タイプに関連付けられる演算子に必要なfunction_for_position_betweenファンクション実装を作成します。(PL/SQLブロックはカッコで囲んでいます。)

このファンクションは、索引ベースのファンクション評価で使用するためのものです。そのため、索引コンテキストおよびスキャン・コンテキストをパラメータとして指定します。

関連項目:

- 索引ベースのファンクション実装を作成する場合の詳細は、『Oracle Databaseデータ・カートリッジ開発者ガイド』を参照してください。
- 「CREATE FUNCTION」および『Oracle Database PL/SQL言語リファレンス』を参照してください。

```
CREATE OR REPLACE FUNCTION function_for_position_between

                           (col NUMBER, lower_pos NUMBER, upper_pos NUMBER,
                            indexctx IN SYS.ODCIIndexCtx,
                            scanctx IN OUT position_im,
                            scanflg IN NUMBER)
RETURN NUMBER AS
  rid              ROWID;
  storage_tab_name VARCHAR2(65);
  lower_bound_stmt VARCHAR2(2000);
  upper_bound_stmt VARCHAR2(2000);
  col_val_stmt     VARCHAR2(2000);
  lower_bound      NUMBER;
  upper_bound      NUMBER;
  column_value     NUMBER;
BEGIN
  IF (indexctx.IndexInfo IS NOT NULL) THEN
    storage_tab_name := indexctx.IndexInfo.INDEXSCHEMA || '.' ||
                        indexctx.IndexInfo.INDEXNAME || '_STORAGE_TAB';
    IF (scanctx IS NULL) THEN
/* This is the first call. Open a cursor for future calls.
   First, do some error checking
*/
      IF (lower_pos > upper_pos) THEN
        RAISE_APPLICATION_ERROR(-20101,
          'Upper Position must be greater than or equal to Lower Position');
      END IF;
      IF (lower_pos <= 0) THEN
        RAISE_APPLICATION_ERROR(-20101,
          'Both Positions must be greater than zero');
      END IF;
/* Obtain the upper and lower value bounds for the range we're interested in.
*/
      upper_bound_stmt := 'Select MIN(col_val) FROM (Select /*+ INDEX_DESC(' ||
                          storage_tab_name || ') */ DISTINCT ' ||
                          'col_val FROM ' || storage_tab_name || ' ORDER BY ' ||
                          'col_val DESC) WHERE rownum <= ' || lower_pos;
      EXECUTE IMMEDIATE upper_bound_stmt INTO upper_bound;
      IF (lower_pos != upper_pos) THEN
        lower_bound_stmt := 'Select MIN(col_val) FROM (Select /*+ INDEX_DESC(' ||
                            storage_tab_name || ') */ DISTINCT ' ||
                            'col_val FROM ' || storage_tab_name ||
                            ' WHERE col_val < ' || upper_bound || ' ORDER BY ' ||
                            'col_val DESC) WHERE rownum <= ' ||
                            (upper_pos - lower_pos);
        EXECUTE IMMEDIATE lower_bound_stmt INTO lower_bound;
      ELSE
        lower_bound := upper_bound;
      END IF;
      IF (lower_bound IS NULL) THEN
        lower_bound := upper_bound;
      END IF;
/* Store the lower and upper bounds for future function invocations for
   the positions.
*/
      scanctx := position_im(0, 0, lower_bound, upper_bound);
    END IF;
```

```
/* Fetch the column value corresponding to the rowid, and see if it falls
   within the determined range.
*/
    col_val_stmt := 'Select col_val FROM ' || storage_tab_name ||
                    ' WHERE base_rowid = ''' || indexctx.Rid || '''';
    EXECUTE IMMEDIATE col_val_stmt INTO column_value;
    IF (column_value <= scanctx.upper_bound AND
        column_value >= scanctx.lower_bound AND
        scanflg = ODCICONST.RegularCall) THEN
      RETURN 1;
    ELSE
      RETURN 0;
    END IF;
  ELSE
    RAISE_APPLICATION_ERROR(-20101, 'A column that has a domain index of' ||
                            'Position indextype must be the first argument');
  END IF;
END;
/
```

次の手順は、function_for_position_betweenファンクションを使用するposition_between演算子を作成します。この演算子には、索引列NUMBERを最初の引数として指定し、その後にNUMBERの下限および上限を2番目および3番目の引数として指定します。

関連項目:

CREATE OPERATOR

```
CREATE OR REPLACE OPERATOR position_between

   BINDING (NUMBER, NUMBER, NUMBER) RETURN NUMBER
   WITH INDEX CONTEXT, SCAN CONTEXT position_im
   USING function_for_position_between;
```

索引コンテキストおよびスキャン・コンテキストが、索引ベースのファンクション評価に渡されるように、このCREATE OPERATOR文にはWITH INDEX CONTEXT, SCAN CONTEXT position_im句が含まれています。

次の手順は、position_operatorに必要な索引タイプposition_indextypeを作成します。

関連項目:

CREATE INDEXTYPE

```
CREATE INDEXTYPE position_indextype

   FOR position_between(NUMBER, NUMBER, NUMBER)
   USING position_im;
```

演算子position_betweenは、索引ベースのファンクション実装を使用します。そのため、索引情報がファンクション評価に渡されるように、参照列にドメイン索引を定義する必要があります。そのため、最後のステップでは、索引タイプposition_indextypeを使用してドメイン索引salary_indexを作成します。

関連項目:

CREATE INDEX

```
CREATE INDEX salary_index ON employees(salary)
```

```
   INDEXTYPE IS position_indextype;
```

これで、演算子position_betweenを使用して、元の問合せを次のように書き換えることができます。

```
SELECT last_name, salary FROM employees
   WHERE position_between(salary, 10, 20)=1
   ORDER BY salary DESC, last_name;
LAST_NAME                      SALARY
-------------------------- ----------
Tucker                          10000
King                            10000
Baer                            10000
Bloom                           10000
Fox                              9600
Bernstein                        9500
Sully                            9500
Greene                           9500
Hunold                           9000
Faviet                           9000
McEwen                           9000
Hall                             9000
Hutton                           8800
Taylor                           8600
Livingston                       8400
Gietz                            8300
Chen                             8200
Fripp                            8200
Weiss                            8000
Olsen                            8000
Smith                            8000
Kaufling                         7900
```

SQL文でのXMLの使用方法

この項では、XMLTypeデータをデータベースで使用する方法を説明します。

XMLType表

サンプル・スキーマoeには表warehousesが含まれ、この表にはXMLType列warehouse_specが含まれます。warehouse_spec情報を持つ別の表を作成するとします。次の例は、CLOB列を1列のみ持つ非常に単純なXMLType表を作成します。

```
CREATE TABLE xwarehouses OF XMLTYPE
  XMLTYPE STORE AS CLOB;
```

このような表には、次の文に示すように、XMLType構文を使用してデータを挿入することができます。(この例で挿入されるデータは、サンプル表oe.warehousesのwarehouse_spec列にあるデータに対応します。warehouse_idは1です。)

```
INSERT INTO xwarehouses VALUES
  (xmltype('<?xml version="1.0"?>
  <Warehouse>
    <WarehouseId>1</WarehouseId>
    <WarehouseName>Southlake, Texas</WarehouseName>
    <Building>Owned</Building>
    <Area>25000</Area>
    <Docks>2</Docks>
    <DockType>Rear load</DockType>
    <WaterAccess>true</WaterAccess>
    <RailAccess>N</RailAccess>
    <Parking>Street</Parking>
    <VClearance>10</VClearance>
  </Warehouse>'));
```

関連項目:

XMLTypeおよびそのメンバー・メソッドの詳細は、『Oracle XML DB開発者ガイド』を参照してください。

この表の問合せには、次の文を使用します。

```
SELECT e.getClobVal() FROM xwarehouses e;
```

CLOB列には、LOB列に対する制限がすべて適用されます。これらの制限事項を回避するには、XMLスキーマベースの表を作成します。XMLスキーマは、XML要素を対応するオブジェクト・リレーショナル・データにマップします。次の例は、XMLスキーマをローカルに登録します。XMLスキーマ(xwarhouses.xsd)には、xwarehouses表と同じ構造が反映されます。(XMLスキーマの宣言ではPL/SQLおよびDBMS_XMLSCHEMAパッケージが使用されています。例では、これらをイタリック体で示しています。)

関連項目:

XMLスキーマの作成の詳細は、『Oracle XML DB開発者ガイド』を参照してください。

```
begin

 dbms_xmlschema.registerSchema(
  'http://www.example.com/xwarehouses.xsd',
  '<schema xmlns="http://www.w3.org/2001/XMLSchema"
       targetNamespace="http://www.example.com/xwarehouses.xsd"
       xmlns:who="http://www.example.com/xwarehouses.xsd"
```

```
       version="1.0">

  <simpleType name="RentalType">
   <restriction base="string">
    <enumeration value="Rented"/>
    <enumeration value="Owned"/>
   </restriction>
  </simpleType>

  <simpleType name="ParkingType">
   <restriction base="string">
    <enumeration value="Street"/>
    <enumeration value="Lot"/>
   </restriction>
  </simpleType>

  <element name = "Warehouse">
    <complexType>
     <sequence>
      <element name = "WarehouseId"   type = "positiveInteger"/>
      <element name = "WarehouseName" type = "string"/>
      <element name = "Building"      type = "who:RentalType"/>
      <element name = "Area"          type = "positiveInteger"/>
      <element name = "Docks"         type = "positiveInteger"/>
      <element name = "DockType"      type = "string"/>
      <element name = "WaterAccess"   type = "boolean"/>
      <element name = "RailAccess"    type = "boolean"/>
      <element name = "Parking"       type = "who:ParkingType"/>
      <element name = "VClearance"    type = "positiveInteger"/>
     </sequence>
    </complexType>
  </element>
</schema>',
   TRUE, TRUE, FALSE, FALSE);
end;
/
```

これで、次の例に示すように、XMLスキーマベースの表を作成できます。

```
CREATE TABLE xwarehouses OF XMLTYPE
   XMLSCHEMA "http://www.example.com/xwarehouses.xsd"
   ELEMENT "Warehouse";
```

デフォルトでは、この表はオブジェクト・リレーショナル表として格納されます。そのため、次の例に示すように、この表にデータを挿入できます。(この例で挿入されるデータは、サンプル表oe.warehousesのwarehouse_spec列にあるデータに対応します。warehouse_idは1です。)

```
INSERT INTO xwarehouses VALUES(   xmltype.createxml('<?xml version="1.0"?>
   <who:Warehouse xmlns:who="http://www.example.com/xwarehouses.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.example.com/xwarehouses.xsd
   http://www.example.com/xwarehouses.xsd">
      <WarehouseId>1</WarehouseId>
      <WarehouseName>Southlake, Texas</WarehouseName>
      <Building>Owned</Building>
      <Area>25000</Area>
      <Docks>2</Docks>
      <DockType>Rear load</DockType>
      <WaterAccess>true</WaterAccess>
      <RailAccess>false</RailAccess>
      <Parking>Street</Parking>
      <VClearance>10</VClearance>
      </who:Warehouse>'));
...
```

XMLスキーマベースの表には制約を定義できます。その場合は、XML要素Warehouse内の適切な属性を参照するように、XMLDATA疑似列を使用します。

```
ALTER TABLE xwarehouses ADD (PRIMARY KEY(XMLDATA."WarehouseId"));
```

xwarehousesのデータはオブジェクト・リレーショナルに格納されるため、可能な場合に基礎となる記憶域を参照できるように、このXMLType表への問合せが書き換えられます。そのため、次の問合せでは、前述の例の主キー制約によって作成された索引を使用します。

```
SELECT * FROM xwarehouses x
   WHERE EXISTSNODE(VALUE(x), '/Warehouse[WarehouseId="1"]',
   'xmlns:who="http://www.example.com/xwarehouses.xsd"') = 1;
SELECT * FROM xwarehouses x
   WHERE EXTRACTVALUE(VALUE(x), '/Warehouse/WarehouseId',
   'xmlns:who="http://www.example.com/xwarehouses.xsd"') = 1;
```

XMLスキーマベースの表に索引を明示的に作成すると、後続の問合せのパフォーマンスが大幅に向上します。XMLType表にオブジェクト・リレーショナル・ビューを作成することも、オブジェクト・リレーショナル表にXMLTypeビューを作成することもできます。

関連項目:

- XMLDATA疑似列の詳細は、「XMLDATA疑似列」を参照してください。
- 「XMLTypeビューの作成: 例」
- 「XMLType表の索引の作成: 例」

XMLType列

サンプル表oe.warehousesは、XMLType型のwarehouse_spec列を使用して作成されました。この項の例は、2つのタイプの記憶域を使用して、簡略化したoe.warehouses表を作成します。

最初の例は、CLOBとして格納されたXMLType表を持つ表を作成します。この表ではXMLスキーマが必要ないため、コンテンツ構造は事前に定義しません。

```
CREATE TABLE xwarehouses (
   warehouse_id        NUMBER,
   warehouse_spec      XMLTYPE)
   XMLTYPE warehouse_spec STORE AS CLOB
   (TABLESPACE example
    STORAGE (INITIAL 6144)
    CHUNK 4000
    NOCACHE LOGGING);
```

次の例でも前述の例とほぼ同じ表を作成しますが、指定されたXMLスキーマによって構造が決められているオブジェクト・リレーショナルXMLType列に、XMLTypeデータが格納されます。

```
CREATE TABLE xwarehouses (
   warehouse_id    NUMBER,
   warehouse_spec  XMLTYPE)
   XMLTYPE warehouse_spec STORE AS OBJECT RELATIONAL
      XMLSCHEMA "http://www.example.com/xwarehouses.xsd"
      ELEMENT "Warehouse";
```

索引

記号

数字

A

B

C

D

- DROP COLUMN句
    - ALTER TABLE 12.3
- DROP制約句
    - ALTER VIEW 12.9
- DROP CONSTRAINT句
    - ALTER TABLE 12.3
- DROP CONTEXT文 16.1
- DROP DATABASE LINK文 16.3
- DROP DATABASE文 16.2
- DROP DIMENSION文 16.4
- DROP DIRECTORY文 16.5
- DROP DISKGROUP文 16.6
- DROP FLASHBACK ARCHIVE文 16.8
- DROP FUNCTION文 16.9
- DROP HIERARCHY文 16.10
- DROP INDEX文 16.11
- DROP INDEXTYPE文 16.12
- DROP INMEMORY JOIN GROUP文 16.13
- DROP JAVA文 16.14
- DROP LIBRARY文 17.1
- DROP LOCKDOWN PROFILE文 17.2
- DROP LOGFILE句
    - ALTER DATABASE 10.8
- DROP LOGFILE MEMBER句
    - ALTER DATABASE 10.8
- DROP MATERIALIZED VIEW LOG文 17.4
- DROP MATERIALIZED VIEW文 17.3
- DROP MATERIALIZED ZONEMAP文 17.5
- DROP OPERATOR文 17.6
- DROP OUTLINE文 17.7
- DROP PACKAGE BODY文 17.8
- DROP PACKAGE文 17.8
- DROP PARTITION句
    - ALTER INDEX 10.16
    - ALTER TABLE 12.3
- DROP PLUGGABLE DATABASE文 17.9
- DROP PRIMARY制約句
    - ALTER TABLE 12.3
- DROP PROCEDURE文 17.10
- DROP PROFILE文 17.11
- DROP RESTORE POINT文 17.12
- DROP ROLE文 17.13
- DROP ROLLBACK SEGMENT文 17.14

E

F

G

H

- FRESH_MV 2.6.4.16
- FULL 2.6.4.17
- GATHER_OPTIMIZER_STATISTICS 2.6.4.18
- HASH 2.6.4.20
- INDEX 2.6.4.22
- INDEX_ASC 2.6.4.23
- INDEX_COMBINE 2.6.4.24
- INDEX_DESC 2.6.4.25
- INDEX_FFS 2.6.4.26
- INDEX_JOIN 2.6.4.27
- INDEX_SS 2.6.4.28
- INDEX_SS_ASC 2.6.4.29
- INDEX_SS_DESC 2.6.4.30
- INMEMORY 2.6.4.31
- INMEMORY_PRUNING 2.6.4.32
- SQL文 2.6.3
- LEADING 2.6.4.33
- 位置構文 2.6.3
- MERGE 2.6.4.34
- MODEL_MIN_ANALYSIS 2.6.4.35
- MONITOR 2.6.4.36
- NO_CLUSTERING 2.6.4.40
- NO_EXPAND 2.6.4.41
- NO_FACT 2.6.4.42
- NO_GATHER_OPTIMIZER_STATISTICS 2.6.4.43
- NO_INDEX 2.6.4.44
- NO_INDEX_FFS 2.6.4.45
- NO_INDEX_SS 2.6.4.46
- NO_INMEMORY 2.6.4.47
- NO_INMEMORY_PRUNING 2.6.4.48
- NO_MERGE 2.6.4.49
- NO_MONITOR 2.6.4.50
- NO_PARALLEL 2.6.4.52
- NO_PARALLEL_INDEX 2.6.4.54
- NO_PQ_CONCURRENT_UNION 2.6.4.56
- NO_PQ_SKEW 2.6.4.57
- NO_PUSH_PRED 2.6.4.58
- NO_PUSH_SUBQ 2.6.4.59
- NO_PX_JOIN_FILTER 2.6.4.60
- NO_QUERY_TRANSFORMATION 2.6.4.61
- NO_RESULT_CACHE 2.6.4.62
- NO_REWRITE 2.6.4.63
- NO_STAR_TRANSFORMATION 2.6.4.65

I

J

K

L

M

N

O

P

Q

R

S

- NUMTODSINTERVAL 7.151
- NUMTOYMINTERVAL 7.152
- NVL 7.153
- NVL2 7.154
- オブジェクト参照ファンクション 7.5
- OLAPファンクション 7.7
- ORA_DM_PARTITION_NAME 7.155
- ORA_DST_AFFECTED 7.156
- ORA_DST_CONVERT 7.157
- ORA_DST_ERROR 7.158
- ORA_HASH 7.159
- ORA_INVOKING_USER 7.160
- ORA_INVOKING_USERID 7.161
- PATH 7.162
- PERCENT_RANK 7.163
- PERCENTILE_CONT 7.164
- PERCENTILE_DISC 7.165
- POWER 7.166
- POWERMULTISET 7.167
- POWERMULTISET_BY_CARDINALITY 7.168
- PREDICTION 7.169
- PREDICTION_BOUNDS 7.170
- PREDICTION_COST 7.171
- PREDICTION_DETAILS 7.172
- PREDICTION_PROBABILITY 7.173
- PREDICTION_SET 7.174
- PRESENTNNV 7.175
- PRESENTV 7.176
- PREVIOUS 7.177
- RANK 7.178
- RATIO_TO_REPORT 7.179
- RAWTOHEX 7.180
- RAWTONHEX 7.181
- REF 7.182
- REFTOHEX 7.183
- REGEXP_COUNT 7.184
- REGEXP_INSTR 7.185
- REGEXP_REPLACE 7.186
- REGEXP_SUBSTR 7.187
- REGR_AVGX 7.188
- REGR_AVGY 7.188
- REGR_COUNT 7.188
- REGR_INTERCEPT 7.188

- SUBSTRC 7.220
- SUM 7.221
- SYS_CONNECT_BY_PATH7.222
- SYS_CONTEXT 7.223
- SYS_DBURIGEN 7.224
- SYS_EXTRACT_UTC 7.225
- SYS_GUID 7.226
- SYS_OP_ZONE_ID 7.227
- SYS_TYPEID 7.228
- SYS_XMLAGG 7.229
- SYS_XMLGEN 7.230
- SYSDATE 7.231
- SYSTIMESTAMP 7.232
- TAN 7.233
- TANH 7.234
- TIMESTAMP_TO_SCN 7.235
- TO_APPROX_COUNT_DISTINCT 7.236
- TO_APPROX_PERCENTILE 7.237
- TO_BINARY_DOUBLE 7.238
- TO_BINARY_FLOAT 7.239
- TO_BLOB (bfile) 7.240
- TO_BLOB (raw) 7.241
- TO_CHAR (bfile|blob)7.242
- TO_CHAR (文字)7.243
- TO_CHAR (日時) 7.244
- TO_CHAR (数値) 7.245
- TO_CLOB (bfile|blob) 7.246
- TO_CLOB (文字) 7.247
- TO_DATE 7.248
- TO_DSINTERVAL 7.249
- TO_LOB 7.250
- TO_MULTI_BYTE 7.251
- TO_NCHAR (文字) 7.252
- TO_NCHAR (日時) 7.253
- TO_NCHAR (数値) 7.254
- TO_NCLOB 7.255
- TO_NUMBER 7.256
- TO_SINGLE_BYTE 7.257
- TO_TIMESTAMP 7.258
- TO_TIMESTAMP_TZ 7.259
- TO_YMINTERVAL 7.261
- TRANSLATE 7.262
- TRANSLATE ... USING 7.263

T

U

V

W

X

Z