Oracle® Database

SQL 言語リファレンス

11*g* リリース 1（11.1）

部品番号 : E05750-03

2008 年 11 月

ORACLE®

Oracle Database SQL 言語リファレンス , 11*g* リリース 1（11.1）

部品番号 : E05750-03

Oracle Database SQL Language Reference, 11*g* Release 1 (11.0)

原本部品番号 : B28286-05

原本著者 : Diana Lorentz

原本協力者 : Bob Jenkins、Drew Adams、Nippun Agarwal、Shashaanka Agarwal、David Alpern、Patrick Amor、Rohan、Angrish、Geeta Arora、Lance Ashdown、David Austin、Thomas Baby、Hermann Baer、Prasad Bagal、Subhransu Basu、Mark Bauer、Eric Belden、Tugrul Bingol、Allen Brumm、Donna Carver、Sivasankaran Chandrasekar、Atif Chaudhry、Beethoven Cheng、Timothy Chien、Alan Choi、George Claborn、Benoit Dageville、Matthew Dombrowski、Jacco Draijjer、George Eadon、William Fisher、Steve Fogel、David Friedman、Amit Ganesh、Raymond Guzman、John Haydu、Chi Hoang、Pat Huey、Sam Idicula、Chandrasekharan Iyer、Ken Jacobs、Mark Jaeger、Balaji Krishnan、Ramkumar Krishnan、Vasudha Krishnaswamy、Ramesh Kumar、Joydip Kundu、Simon Law、Bill Lee、Geoff Lee、Nina Lewis、Zhen Liu、Bryn Llewellyn、Rich Long、Scott Lynn、Robert McGuirk、Ben Meng、Mughees Minhas、Krishna Mohan、Sheila Moore、Tony Morales、Ari Mozes、Niloy Mukherjee、Denis Mukhin、Gopal Mulagund、Ravi Murthy、Sujatha Muthulingam、DheerajPandey、Hanlin Qian、Kevin Quinn、Christopher Racicot、Venkatesh Radhakrishnan、Ananth Raghavan、Ashish Ray、Kathy Rich、Shrikanth Sankar、Vivian Schupmann、Laurent Schneider、Lei Sheng、Bipul Sinha、Wayne Smith、Kesavan Srinivasan、Peter Stengard、Gaby Stredie、Sankar Subramanian、Seema Sundara、Anh-Tuan Tran、Kothanda Umamageswaran、Randy Urbano、Mark van de Wiel、Badhri Varanasi、Srinivas Vemuri、Radek Vingralek、Guhan Viswanathan、William Waddington、Shaoyu Wang、Wei Wang、Steve Wertheimer、Charles Wetherell、Rajiv Wickremesinghe、Andrew Witkowski、Brian Wolf、Sergiusz Wolicki、Daniel Wong、Tsae-Feng Yu、Mohamed Zait、Mohammed Zaiuddin、Fred Zemke、Weiran Zhang

目次

はじめに xix

対象読者 xx
ドキュメントのアクセシビリティについて xx
関連ドキュメント xx
表記規則 xxi
サポートおよびサービス xxi

このリリースの新機能 xxiii

このマニュアルでの Oracle Database 11*g* リリース 1 の新機能 xxiv

1 概要

SQL の歴史 1-2
SQL 規格 1-2
SQL の特長 1-2
すべてのリレーショナル・データベースに共通の言語 1-3
最新の機能拡張 1-3
Enterprise Manager の使用方法 1-4
字句規則 1-4
ツール製品のサポート 1-4

2 Oracle SQL の基本要素

データ型 2-2
Oracle の組込みデータ型 2-7
CHAR データ型 2-9
NCHAR データ型 2-10
NVARCHAR2 データ型 2-10
VARCHAR2 データ型 2-10
VARCHAR データ型 2-11
NUMBER データ型 2-11
FLOAT データ型 2-13
浮動小数点数 2-13
BINARY_FLOAT 2-13
BINARY_DOUBLE 2-14
数値の優先順位 2-15
DATE データ型 2-17
ユリウス日の使用方法 2-17
TIMESTAMP データ型 2-18

TIMESTAMP WITH TIME ZONE データ型 2-18
TIMESTAMP WITH LOCAL TIME ZONE データ型 2-19
INTERVAL YEAR TO MONTH データ型 2-19
INTERVAL DAY TO SECOND データ型 2-20
日時および期間の演算 2-20
夏時間のサポート 2-22
日時および期間の例 2-22
RAW データ型と LONG RAW データ型 2-23
BFILE データ型 2-25
BLOB データ型 2-25
CLOB データ型 2-26
NCLOB データ型 2-26
ROWID データ型 2-26
ROWID データ型 2-26
UROWID データ型 2-27
ANSI、DB2、SQL/DS のデータ型 2-27
ユーザー定義型 2-29
オブジェクト型 2-29
REF データ型 2-30
VARRAY 2-30
ネストした表 2-30
Oracle が提供する型 2-31
任意型 2-31
ANYTYPE 2-31
ANYDATA 2-31
ANYDATASET 2-31
XML 型 2-32
XMLType 2-32
URI データ型 2-32
URIFactory パッケージ 2-33
空間型 2-33
SDO_GEOMETRY 2-33
SDO_TOPO_GEOMETRY 2-34
SDO_GEORASTER 2-34
メディア型 2-34
ORDAudio 2-34
ORDImage 2-34
ORDVideo 2-35
ORDDoc 2-35
ORDDicom 2-35
SI_StillImage 2-35
SI_Color 2-35
SI_AverageColor 2-35
SI_ColorHistogram 2-35
SI_PositionalColor 2-35
SI_Texture 2-35
SI_FeatureList 2-35
ORDImageSignature 2-35
Expression Filter 型 2-36
Expression 2-36

データ型の比較規則 2-36
数値 2-36
日付値 2-36
文字値 2-36
オブジェクト値 2-39
VARRAY とネストした表 2-39
データ型の優先順位 2-39
データ変換 2-39
暗黙的なデータ変換と明示的なデータ変換 2-39
暗黙的なデータ変換 2-39
暗黙的なデータ変換の例 2-41
明示的なデータ変換 2-42
データ変換のセキュリティ上の考慮事項 2-43
リテラル 2-44
テキスト・リテラル 2-44
数値リテラル 2-46
整数リテラル 2-46
NUMBER および浮動小数点リテラル 2-46
日時リテラル 2-48
期間リテラル 2-51
INTERVAL YEAR TO MONTH 2-51
INTERVAL DAY TO SECOND 2-52
書式モデル 2-54
数値書式モデル 2-54
数値書式要素 2-55
日時書式モデル 2-57
日時書式要素 2-58
日付書式要素における大文字 2-58
日時書式モデルにおける句読点と文字リテラル 2-58
日時書式要素およびグローバリゼーション・サポート 2-61
ISO 標準日付書式要素 2-62
RR 日時書式要素 2-62
RR 日時書式の例 2-62
日時書式要素の接尾辞 2-63
書式モデルの修飾子 2-63
書式モデルの例 2-64
文字列から日付への変換に関する規則 2-66
XML 書式モデル 2-66
NULL 2-67
SQL ファンクションでの NULL 2-68
比較条件での NULL 2-68
条件での NULL 2-68
コメント 2-69
SQL 文中のコメント 2-69
スキーマ・オブジェクトおよび非スキーマ・オブジェクトに関するコメント 2-70
ヒントの使用方法 2-70
ヒントのリスト（アルファベット順） 2-74
ALL_ROWS ヒント 2-74

APPEND ヒント 2-74
CACHE ヒント 2-75
CLUSTER ヒント 2-75
CURSOR_SHARING_EXACT ヒント 2-75
DRIVING_SITE ヒント 2-75
DYNAMIC_SAMPLING ヒント 2-76
FACT ヒント 2-76
FIRST_ROWS ヒント 2-77
FULL ヒント 2-77
HASH ヒント 2-77
INDEX ヒント 2-78
INDEX_ASC ヒント 2-78
INDEX_COMBINE ヒント 2-79
INDEX_DESC ヒント 2-79
INDEX_FFS ヒント 2-79
INDEX_JOIN ヒント 2-80
INDEX_SS ヒント 2-80
INDEX_SS_ASC ヒント 2-80
INDEX_SS_DESC ヒント 2-81
LEADING ヒント 2-81
MERGE ヒント 2-82
MODEL_MIN_ANALYSIS ヒント 2-82
MONITOR ヒント 2-82
NATIVE_FULL_OUTER_JOIN 2-82
NOAPPEND ヒント 2-83
NOCACHE ヒント 2-83
NO_EXPAND ヒント 2-83
NO_FACT ヒント 2-84
NO_INDEX ヒント 2-84
NO_INDEX_FFS ヒント 2-84
NO_INDEX_SS ヒント 2-85
NO_MERGE ヒント 2-85
NO_MONITOR ヒント 2-85
NO_NATIVE_FULL_OUTER_JOIN 2-85
NO_PARALLEL ヒント 2-86
NOPARALLEL ヒント 2-86
NO_PARALLEL_INDEX ヒント 2-86
NOPARALLEL_INDEX ヒント 2-86
NO_PUSH_PRED ヒント 2-86
NO_PUSH_SUBQ ヒント 2-87
NO_PX_JOIN_FILTER ヒント 2-87
NO_QUERY_TRANSFORMATION ヒント 2-87
NO_RESULT_CACHE ヒント 2-87
NO_REWRITE ヒント 2-87
NOREWRITE ヒント 2-88
NO_STAR_TRANSFORMATION ヒント 2-88
NO_UNNEST ヒント 2-88
NO_USE_HASH ヒント 2-88
NO_USE_MERGE ヒント 2-88
NO_USE_NL ヒント 2-89
NO_XMLINDEX_REWRITE ヒント 2-89

NO_XML_QUERY_REWRITE ヒント 2-89
OPT_PARAM ヒント 2-89
ORDERED ヒント 2-90
PARALLEL ヒント 2-90
PARALLEL_INDEX ヒント 2-91
PQ_DISTRIBUTE ヒント 2-91
PUSH_PRED ヒント 2-92
PUSH_SUBQ ヒント 2-93
PX_JOIN_FILTER ヒント 2-93
QB_NAME ヒント 2-93
RESULT_CACHE ヒント 2-93
REWRITE ヒント 2-94
STAR_TRANSFORMATION ヒント 2-94
UNNEST ヒント 2-95
USE_CONCAT ヒント 2-95
USE_HASH ヒント 2-96
USE_MERGE ヒント 2-96
USE_NL ヒント 2-96
USE_NL_WITH_INDEX ヒント 2-97
データベース・オブジェクト 2-97
スキーマ・オブジェクト 2-97
非スキーマ・オブジェクト 2-98
スキーマ・オブジェクト名および修飾子 2-98
スキーマ・オブジェクトのネーミング規則 2-98
スキーマ・オブジェクトのネーミング例 2-101
スキーマ・オブジェクトのネーミングのガイドライン 2-102
スキーマ・オブジェクトの構文および SQL 文の構成要素 2-102
Oracle Database によるスキーマ・オブジェクト参照の変換方法 2-103
他のスキーマ内のオブジェクトの参照 2-104
リモート・データベース内のオブジェクトの参照 2-104
データベース・リンクの作成 2-104
データベース・リンク名 2-104
ユーザー名およびパスワード 2-105
データベース接続文字列 2-105
データベース・リンクの参照 2-105
パーティション表と索引の参照 2-106
オブジェクト型の属性とメソッドの参照 2-108

3 疑似列

階層問合せ疑似列 3-2
CONNECT_BY_ISCYCLE 疑似列 3-2
CONNECT_BY_ISLEAF 疑似列 3-2
LEVEL 疑似列 3-3
順序疑似列 3-3
順序値の使用場所 3-4
順序値の使用方法 3-4
バージョン問合せ疑似列 3-6
COLUMN_VALUE 疑似列 3-6
OBJECT_ID 疑似列 3-7

OBJECT_VALUE 疑似列 3-7
ORA_ROWSCN 疑似列 3-8
ROWID 疑似列 3-8
ROWNUM 疑似列 3-9
XMLDATA 疑似列 3-10

4 演算子

SQL 演算子 4-2
単項演算子およびバイナリ演算子 4-2
演算子の優先順位 4-3
算術演算子 4-3
連結演算子 4-4
階層問合せ演算子 4-5
PRIOR 4-5
CONNECT_BY_ROOT 4-5
集合演算子 4-6
MULTISET 演算子 4-6
MULTISET EXCEPT 4-7
MULTISET INTERSECT 4-7
MULTISET UNION 4-8
ユーザー定義演算子 4-9

5 ファンクション

SQL ファンクション 5-2
単一行ファンクション 5-3
数値ファンクション 5-3
文字値を戻す文字ファンクション 5-4
NLS 文字ファンクション 5-4
数値を戻す文字ファンクション 5-4
日時ファンクション 5-5
一般的な比較ファンクション 5-5
変換ファンクション 5-5
ラージ・オブジェクト・ファンクション 5-6
収集ファンクション 5-6
階層ファンクション 5-6
データ・マイニング・ファンクション 5-7
XML ファンクション 5-7
エンコーディング・ファンクションおよびデコーディング・ファンクション 5-8
NULL 関連ファンクション 5-8
環境ファンクションおよび識別子ファンクション 5-8
集計ファンクション 5-8
分析ファンクション 5-10
オブジェクト参照ファンクション 5-15
モデル・ファンクション 5-15
SQL ファンクションのリスト（アルファベット順） 5-15
ABS 5-15
ACOS 5-16
ADD_MONTHS 5-16
APPENDCHILDXML 5-17

ASCIISTR 5-17
ASCII 5-18
ASIN 5-19
ATAN 5-19
ATAN2 5-20
AVG 5-20
BFILENAME 5-21
BIN_TO_NUM 5-22
BITAND 5-23
CARDINALITY 5-25
CAST 5-25
CEIL 5-28
CHARTOROWID 5-28
CHR 5-29
CLUSTER_ID 5-30
CLUSTER_PROBABILITY 5-31
CLUSTER_SET 5-32
COALESCE 5-35
COLLECT 5-36
COMPOSE 5-36
CONCAT 5-37
CONVERT 5-38
CORR 5-39
CORR_* 5-41
CORR_S 5-42
CORR_K 5-42
COS 5-42
COSH 5-43
COUNT 5-43
COVAR_POP 5-45
COVAR_SAMP 5-46
CUBE_TABLE 5-47
CUME_DIST 5-48
CURRENT_DATE 5-50
CURRENT_TIMESTAMP 5-50
CV 5-51
DATAOBJ_TO_PARTITION 5-52
DBTIMEZONE 5-53
DECODE 5-53
DECOMPOSE 5-54
DELETEXML 5-55
DENSE_RANK 5-56
DEPTH 5-58
DEREF 5-58
DUMP 5-59
EMPTY_BLOB、EMPTY_CLOB 5-60
EXISTSNODE 5-61
EXP 5-62
EXTRACT（日時） 5-62
EXTRACT（XML） 5-65
EXTRACTVALUE 5-65

FEATURE_ID 5-66
FEATURE_SET 5-67
FEATURE_VALUE 5-69
FIRST 5-71
FIRST_VALUE 5-72
FLOOR 5-74
FROM_TZ 5-74
GREATEST 5-75
GROUP_ID 5-75
GROUPING 5-76
GROUPING_ID 5-77
HEXTORAW 5-78
INITCAP 5-78
INSERTCHILDXML 5-79
INSERTCHILDXMLAFTER 5-80
INSERTCHILDXMLBEFORE 5-81
INSERTXMLAFTER 5-82
INSERTXMLBEFORE 5-82
INSTR 5-83
ITERATION_NUMBER 5-85
LAG 5-86
LAST 5-87
LAST_DAY 5-87
LAST_VALUE 5-88
LEAD 5-90
LEAST 5-91
LENGTH 5-91
LN 5-92
LNNVL 5-93
LOCALTIMESTAMP 5-94
LOG 5-95
LOWER 5-95
LPAD 5-96
LTRIM 5-96
MAKE_REF 5-97
MAX 5-98
MEDIAN 5-99
MIN 5-101
MOD 5-102
MONTHS_BETWEEN 5-103
NANVL 5-103
NCHR 5-104
NEW_TIME 5-105
NEXT_DAY 5-106
NLS_CHARSET_DECL_LEN 5-106
NLS_CHARSET_ID 5-107
NLS_CHARSET_NAME 5-107
NLS_INITCAP 5-108
NLS_LOWER 5-109
NLSSORT 5-109
NLS_UPPER 5-111

NTILE 5-111
NULLIF 5-112
NUMTODSINTERVAL 5-113
NUMTOYMINTERVAL 5-114
NVL 5-115
NVL2 5-115
ORA_HASH 5-116
PATH 5-117
PERCENT_RANK 5-118
PERCENTILE_CONT 5-119
PERCENTILE_DISC 5-121
POWER 5-122
POWERMULTISET 5-123
POWERMULTISET_BY_CARDINALITY 5-124
PREDICTION 5-125
PREDICTION_BOUNDS 5-127
PREDICTION_COST 5-128
PREDICTION_DETAILS 5-130
PREDICTION_PROBABILITY 5-131
PREDICTION_SET 5-133
PRESENTNNV 5-135
PRESENTV 5-136
PREVIOUS 5-137
RANK 5-138
RATIO_TO_REPORT 5-140
RAWTOHEX 5-141
RAWTONHEX 5-141
REF 5-142
REFTOHEX 5-142
REGEXP_COUNT 5-143
REGEXP_INSTR 5-144
REGEXP_REPLACE 5-147
REGEXP_SUBSTR 5-149
REGR_（線形回帰）ファンクション 5-150
REMAINDER 5-155
REPLACE 5-155
ROUND（数値） 5-156
ROUND（日付） 5-157
ROW_NUMBER 5-158
ROWIDTOCHAR 5-159
ROWIDTONCHAR 5-160
RPAD 5-160
RTRIM 5-161
SCN_TO_TIMESTAMP 5-162
SESSIONTIMEZONE 5-163
SET 5-163
SIGN 5-164
SIN 5-165
SINH 5-165
SOUNDEX 5-166
SQRT 5-167

STATS_BINOMIAL_TEST 5-167
STATS_CROSSTAB 5-168
STATS_F_TEST 5-169
STATS_KS_TEST 5-170
STATS_MODE 5-171
STATS_MW_TEST 5-172
STATS_ONE_WAY_ANOVA 5-173
STATS_T_TEST_* 5-174
STATS_T_TEST_ONE 5-175
STATS_T_TEST_PAIRED 5-176
STATS_T_TEST_INDEP および STATS_T_TEST_INDEPU 5-176
STATS_WSR_TEST 5-177
STDDEV 5-178
STDDEV_POP 5-179
STDDEV_SAMP 5-180
SUBSTR 5-181
SUM 5-182
SYS_CONNECT_BY_PATH 5-184
SYS_CONTEXT 5-184
SYS_DBURIGEN 5-190
SYS_EXTRACT_UTC 5-190
SYS_GUID 5-191
SYS_TYPEID 5-191
SYS_XMLAGG 5-192
SYS_XMLGEN 5-193
SYSDATE 5-194
SYSTIMESTAMP 5-194
TAN 5-195
TANH 5-195
TIMESTAMP_TO_SCN 5-196
TO_BINARY_DOUBLE 5-196
TO_BINARY_FLOAT 5-198
TO_BLOB 5-198
TO_CHAR（文字） 5-199
TO_CHAR（日時） 5-200
TO_CHAR（数値） 5-201
TO_CLOB 5-203
TO_DATE 5-203
TO_DSINTERVAL 5-205
TO_LOB 5-206
TO_MULTI_BYTE 5-206
TO_NCHAR（文字） 5-207
TO_NCHAR（日時） 5-207
TO_NCHAR（数値） 5-208
TO_NCLOB 5-209
TO_NUMBER 5-209
TO_SINGLE_BYTE 5-210
TO_TIMESTAMP 5-210
TO_TIMESTAMP_TZ 5-211
TO_YMINTERVAL 5-212
TRANSLATE 5-213

TRANSLATE ... USING 5-214
TREAT 5-215
TRIM 5-216
TRUNC（数値） 5-217
TRUNC（日付） 5-218
TZ_OFFSET 5-218
UID 5-219
UNISTR 5-219
UPDATEXML 5-220
UPPER 5-221
USER 5-221
USERENV 5-222
VALUE 5-223
VAR_POP 5-223
VAR_SAMP 5-224
VARIANCE 5-225
VSIZE 5-226
WIDTH_BUCKET 5-227
XMLAGG 5-228
XMLCAST 5-229
XMLCDATA 5-230
XMLCOLATTVAL 5-231
XMLCOMMENT 5-232
XMLCONCAT 5-232
XMLDIFF 5-233
XMLELEMENT 5-234
XMLEXISTS 5-237
XMLFOREST 5-237
XMLPARSE 5-238
XMLPATCH 5-239
XMLPI 5-240
XMLQUERY 5-241
XMLROOT 5-242
XMLSEQUENCE 5-243
XMLSERIALIZE 5-244
XMLTABLE 5-245
XMLTRANSFORM 5-247
ROUND および TRUNC 日付ファンクション 5-248
ユーザー定義ファンクション 5-249
前提条件 5-250
名前の優先順位 5-250
ネーミング規則 5-250

6 式

SQL 式 6-2
単純式 6-3
複合式 6-4
CASE 式 6-5
列式 6-6
CURSOR 式 6-7

日時式 6-8
ファンクション式 6-10
期間式 6-10
モデル式 6-11
オブジェクト・アクセス式 6-13
プレースホルダ式 6-13
スカラー副問合せ式 6-14
型コンストラクタ式 6-14
式のリスト 6-15

7 条件

SQL 条件 7-2
条件の優先順位 7-3
比較条件 7-4
単純比較条件 7-5
グループ比較条件 7-6
浮動小数点条件 7-7
論理条件 7-8
モデル条件 7-9
IS ANY 条件 7-9
IS PRESENT 条件 7-10
多重集合条件 7-11
IS A SET 条件 7-11
IS EMPTY 条件 7-12
MEMBER 条件 7-12
SUBMULTISET 条件 7-13
パターン一致条件 7-14
LIKE 条件 7-14
REGEXP_LIKE 条件 7-17
NULL 条件 7-18
XML 条件 7-19
EQUALS_PATH 条件 7-19
UNDER_PATH 条件 7-19
複合条件 7-20
BETWEEN 条件 7-20
EXISTS 条件 7-21
IN 条件 7-22
IS OF *type* 条件 7-23

8 共通の SQL DDL 句

allocate_extent_clause 8-2
constraint 8-4
deallocate_unused_clause 8-24
file_specification 8-26
logging_clause 8-34
parallel_clause 8-37
physical_attributes_clause 8-39
size_clause 8-42
storage_clause 8-43

9 SQL 問合せおよび副問合せ

問合せおよび副問合せ 9-2
単純な問合せの作成 9-2
階層問合せ 9-3
階層問合せの例 9-5
UNION [ALL]、INTERSECT および MINUS 演算子 9-7
問合せ結果のソート 9-10
結合 9-10
結合条件 9-10
等価結合 9-10
自己結合 9-11
デカルト積 9-11
内部結合 9-11
外部結合 9-11
アンチ結合 9-12
セミ結合 9-13
副問合せの使用方法 9-13
ネストされた副問合せのネスト解除 9-14
DUAL 表からの選択 9-14
分散問合せ 9-14

10 SQL 文 : ALTER CLUSTER ～ ALTER JAVA

様々な種類の SQL 文 10-2
データ定義言語（DDL）文 10-2
データ操作言語（DML）文 10-3
トランザクション制御文 10-3
セッション制御文 10-3
システム制御文 10-3
埋込み SQL 文 10-4
SQL 文に関する章の構成 10-4
ALTER CLUSTER 10-5
ALTER DATABASE 10-9
ALTER DIMENSION 10-43
ALTER DISKGROUP 10-46
ALTER FLASHBACK ARCHIVE 10-60
ALTER FUNCTION 10-63
ALTER INDEX 10-64
ALTER INDEXTYPE 10-82
ALTER JAVA 10-85

11 SQL 文 : ALTER MATERIALIZED VIEW ～ ALTER SYSTEM

ALTER MATERIALIZED VIEW 11-2
ALTER MATERIALIZED VIEW LOG 11-16
ALTER OPERATOR 11-22
ALTER OUTLINE 11-25
ALTER PACKAGE 11-27
ALTER PROCEDURE 11-28

ALTER PROFILE 11-29
ALTER RESOURCE COST 11-32
ALTER ROLE 11-34
ALTER ROLLBACK SEGMENT 11-36
ALTER SEQUENCE 11-39
ALTER SESSION 11-41
初期化パラメータおよび ALTER SESSION 11-45
セッション・パラメータおよび ALTER SESSION 11-46
ALTER SYSTEM 11-52

12 SQL 文 : ALTER TABLE ～ ALTER TABLESPACE

ALTER TABLE 12-2
ALTER TABLESPACE 12-82

13 SQL 文 : ALTER TRIGGER ～ COMMIT

ALTER TRIGGER 13-2
ALTER TYPE 13-4
ALTER USER 13-5
ALTER VIEW 13-12
ANALYZE 13-14
ASSOCIATE STATISTICS 13-21
AUDIT 13-25
CALL 13-37
COMMENT 13-41
COMMIT 13-44

14 SQL 文 : CREATE CLUSTER ～ CREATE JAVA

CREATE CLUSTER 14-2
CREATE CONTEXT 14-8
CREATE CONTROLFILE 14-10
CREATE DATABASE 14-16
CREATE DATABASE LINK 14-28
CREATE DIMENSION 14-33
CREATE DIRECTORY 14-38
CREATE DISKGROUP 14-40
CREATE FLASHBACK ARCHIVE 14-45
CREATE FUNCTION 14-48
CREATE INDEX 14-50
CREATE INDEXTYPE 14-73
CREATE JAVA 14-76

15 SQL 文 : CREATE LIBRARY ～ CREATE SPFILE

CREATE LIBRARY 15-2
CREATE MATERIALIZED VIEW 15-4
CREATE MATERIALIZED VIEW LOG 15-26
CREATE OPERATOR 15-32
CREATE OUTLINE 15-35
CREATE PACKAGE 15-39
CREATE PACKAGE BODY 15-41

CREATE PFILE 15-43
CREATE PROCEDURE 15-45
CREATE PROFILE 15-47
CREATE RESTORE POINT 15-53
CREATE ROLE 15-56
CREATE ROLLBACK SEGMENT 15-59
CREATE SCHEMA 15-62
CREATE SEQUENCE 15-64
CREATE SPFILE 15-68

16 SQL 文 : CREATE SYNONYM ～ CREATE TRIGGER

CREATE SYNONYM 16-2
CREATE TABLE 16-6
CREATE TABLESPACE 16-71
CREATE TRIGGER 16-85

17 SQL 文 : CREATE TYPE ～ DROP ROLLBACK SEGMENT

CREATE TYPE 17-3
CREATE TYPE BODY 17-5
CREATE USER 17-7
CREATE VIEW 17-13
DELETE 17-23
DISASSOCIATE STATISTICS 17-30
DROP CLUSTER 17-32
DROP CONTEXT 17-34
DROP DATABASE 17-35
DROP DATABASE LINK 17-36
DROP DIMENSION 17-37
DROP DIRECTORY 17-38
DROP DISKGROUP 17-39
DROP FLASHBACK ARCHIVE 17-41
DROP FUNCTION 17-42
DROP INDEX 17-44
DROP INDEXTYPE 17-46
DROP JAVA 17-47
DROP LIBRARY 17-48
DROP MATERIALIZED VIEW 17-49
DROP MATERIALIZED VIEW LOG 17-51
DROP OPERATOR 17-53
DROP OUTLINE 17-54
DROP PACKAGE 17-55
DROP PROCEDURE 17-57
DROP PROFILE 17-58
DROP RESTORE POINT 17-59
DROP ROLE 17-60
DROP ROLLBACK SEGMENT 17-61

18 SQL 文 : DROP SEQUENCE ～ ROLLBACK

DROP SEQUENCE 18-2
DROP SYNONYM 18-3
DROP TABLE 18-5
DROP TABLESPACE 18-8
DROP TRIGGER 18-11
DROP TYPE 18-12
DROP TYPE BODY 18-14
DROP USER 18-15
DROP VIEW 18-17
EXPLAIN PLAN 18-19
FLASHBACK DATABASE 18-22
FLASHBACK TABLE 18-25
GRANT 18-31
INSERT 18-53
LOCK TABLE 18-69
MERGE 18-72
NOAUDIT 18-76
PURGE 18-80
RENAME 18-82
REVOKE 18-84
ROLLBACK 18-92

19 SQL 文 : SAVEPOINT ～ UPDATE

SAVEPOINT 19-2
SELECT 19-4
SET CONSTRAINT[S] 19-49
SET ROLE 19-51
SET TRANSACTION 19-53
TRUNCATE CLUSTER 19-56
TRUNCATE TABLE 19-58
UPDATE 19-62

A 構文図の読み方

図形構文図 A-2
必須キーワードとパラメータ A-3
オプションのキーワードとパラメータ A-4
構文のループ A-4
複数の部分に分割された構文図 A-5
データベース・オブジェクト A-5

B Oracle と標準 SQL

ANSI 規格 B-2
ISO 規格 B-2
Core SQL:2003 に対する Oracle の準拠 B-3
SQL/Foundation:2003 のオプション機能に対する Oracle のサポート B-9
SQL/CLI:2003 に対する Oracle の準拠 B-18
SQL/PSM:2003 に対する Oracle の準拠 B-18

SQL/MED:2003 に対する Oracle の準拠 B-18
SQL/OLB:2003 に対する Oracle の準拠 B-18
SQL/XML:2006 に対する Oracle の準拠 B-18
FIPS 127-2 に対する Oracle の準拠 B-26
標準 SQL に対する Oracle 拡張機能 B-27
以前の規格に対する Oracle の準拠 B-28
キャラクタ・セットのサポート B-28

C Oracle の正規表現のサポート

多言語の正規表現の構文 C-2
正規表現の演算子の多言語拡張 C-3
Oracle の正規表現における Perl による拡張機能 C-4

D Oracle Database 予約語

E 詳細な例

拡張索引作成機能の使用方法 E-2
SQL 文での XML の使用方法 E-8

索引

はじめに

このマニュアルでは、Oracle Database の情報を管理するために使用される Structured Query Language（SQL）について説明します。Oracle SQL は、ANSI（米国規格協会）および ISO（国際標準化機構）の SQL:1999 規格にさらに多くの内容を盛り込んでいます。

ここでは、次の項目について説明します。

- 対象読者
- ドキュメントのアクセシビリティについて
- 関連ドキュメント
- 表記規則
- サポートおよびサービス

対象読者

このマニュアルは、すべての Oracle SQL ユーザーを対象としています。

ドキュメントのアクセシビリティについて

オラクル社は、障害のあるお客様にもオラクル社の製品、サービスおよびサポート・ドキュメントを簡単にご利用いただけることを目標としています。オラクル社のドキュメントには、ユーザーが障害支援技術を使用して情報を利用できる機能が組み込まれています。HTML 形式のドキュメントで用意されており、障害のあるお客様が簡単にアクセスできるようにマークアップされています。標準規格は改善されつつあります。オラクル社はドキュメントをすべてのお客様がご利用できるように、市場をリードする他の技術ベンダーと積極的に連携して技術的な問題に対応しています。オラクル社のアクセシビリティについての詳細情報は、Oracle Accessibility Program の Web サイト http://www.oracle.com/accessibility/ を参照してください。

ドキュメント内のサンプル・コードのアクセシビリティについて

スクリーン・リーダーは、ドキュメント内のサンプル・コードを正確に読めない場合があります。コード表記規則では閉じ括弧だけを行に記述する必要があります。しかし JAWS は括弧だけの行を読まない場合があります。

外部 Web サイトのドキュメントのアクセシビリティについて

このドキュメントにはオラクル社およびその関連会社が所有または管理しない Web サイトへのリンクが含まれている場合があります。オラクル社およびその関連会社は、それらの Web サイトのアクセシビリティに関しての評価や言及は行っておりません。

Oracle サポート・サービスへの TTY アクセス

アメリカ国内では、Oracle サポート・サービスへ 24 時間年中無休でテキスト電話（TTY）アクセスが提供されています。TTY サポートについては、(800)446-2398 にお電話ください。アメリカ国外からの場合は、+1-407-458-2479 にお電話ください。

関連ドキュメント

詳細は、次の Oracle ドキュメントを参照してください。

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

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

表記規則

このマニュアルでは次の表記規則を使用します。

規則	意味
太字	太字は、操作に関連する Graphical User Interface 要素、または本文中で定義されている用語および用語集に記載されている用語を示します。
イタリック体	イタリックは、ユーザーが特定の値を指定するプレースホルダ変数を示します。
固定幅フォント	固定幅フォントは、段落内のコマンド、URL、サンプル内のコード、画面に表示されるテキスト、または入力するテキストを示します。

サポートおよびサービス

次の各項に、各サービスに接続するための URL を記載します。

Oracle サポート・サービス

オラクル製品サポートの購入方法、および Oracle サポート・サービスへの連絡方法の詳細は、次の URL を参照してください。

```
http://www.oracle.com/lang/jp/support/index.html
```

製品マニュアル

製品のマニュアルは、次の URL にあります。

```
http://www.oracle.com/technology/global/jp/documentation/index.html
```

研修およびトレーニング

研修に関する情報とスケジュールは、次の URL で入手できます。

```
http://education.oracle.com/pls/web_prod-plq-dad/db_pages.getpage?page_id=3
```

その他の情報

オラクル製品やサービスに関するその他の情報については、次の URL から参照してください。

```
http://www.oracle.com/lang/jp/index.html
http://www.oracle.com/technology/global/jp/index.html
```

注意： ドキュメント内に記載されている URL や参照ドキュメントには、Oracle Corporation が提供する英語の情報も含まれています。日本語版の情報については、前述の URL を参照してください。

このリリースの新機能

ここでは、Oracle Database 11*g* の新機能について説明し、追加情報の参照先を示します。

以前のリリースの Oracle Database の新機能については、該当するリリースのマニュアルを参照してください。

このマニュアルでの Oracle Database 11*g* リリース 1 の新機能

このリリースでの構成の変更

多くの SQL 文は、ほぼ PL/SQL 要素のみで構成されています。これらの文はこのマニュアルでも取り上げられていますが、構文とセマンティクスの大部分は『Oracle Database PL/SQL 言語リファレンス』に移行されました。次の表に、このマニュアルに記載されている簡略化された SQL 構文とセマンティクス、および『Oracle Database PL/SQL 言語リファレンス』に記載されている完全な構文とセマンティクスへのリンクを示します。

簡略化された SQL の項	完全な構文とセマンティクス
CREATE FUNCTION（14-48 ページ）	CREATE FUNCTION
CREATE PACKAGE（15-39 ページ）	CREATE PACKAGE
CREATE PACKAGE BODY（15-41 ページ）	CREATE PACKAGE BODY
CREATE PROCEDURE（15-45 ページ）	CREATE PROCEDURE
CREATE TRIGGER（16-85 ページ）	CREATE TRIGGER
CREATE TYPE（17-3 ページ）	CREATE TYPE
CREATE TYPE BODY（17-5 ページ）	CREATE TYPE BODY
ALTER FUNCTION（10-63 ページ）	ALTER FUNCTION
ALTER PACKAGE（11-27 ページ）	ALTER PACKAGE
ALTER PROCEDURE（11-28 ページ）	ALTER PROCEDURE
ALTER TRIGGER（13-2 ページ）	ALTER TRIGGER
ALTER TYPE（13-4 ページ）	ALTER TYPE

このリリースの新機能

今回のリリースでは、次の SQL 文が追加または拡張されています。

- ALTER DATABASE（10-9 ページ）には、次の拡張が行われています。
  - managed_standby_recovery 句（10-22 ページ）が大幅に簡略化されました。ほとんどのリカバリ処理がデータベースで自動的に行われるようになったため、数多くの副次句が非推奨になりました。
  - supplemental_db_logging（10-31 ページ）には、PL/SQL コールのサプリメンタル・ロギングを使用可能または使用禁止にするための構文が新しく追加されています。
  - standby_database_clauses（10-33 ページ）には、物理スタンバイ・データベースをスナップショット・スタンバイ・データベースに変換したり、スナップショット・スタンバイ・データベースを物理スタンバイ・データベースに変換するための構文が新しく追加されています。
  - managed_standby_recovery 句（10-22 ページ）には、論理スタンバイで提供されるローリング・アップグレード機能を使用したり、プライマリ・データベースや物理スタンバイを元の構成に戻すための `KEEP IDENTITY` 構文が新しく追加されています。
- ALTER DISKGROUP（10-46 ページ）には、次の拡張が行われています。
  - check_diskgroup_clause（10-53 ページ）の、自動ストレージ管理環境におけるディスク・グループ、ディスクおよびファイルの一貫性をチェックするための構文が簡略化されました。
  - diskgroup_availability 句（10-57 ページ）には、ディスク・グループをマウントするためのオプションが新しく追加されています。

  - 修復するディスクをオフラインにし、後でオンラインに戻すための disk_offline_clause 句（10-52 ページ）および disk_online_clause 句（10-52 ページ）が新しく追加されています。

- ALTER INDEX（10-64 ページ）には、次の拡張が行われています。
  - ユーザー管理記憶表のドメイン索引をシステム管理記憶表に移行するための MIGRATE パラメータが新しく追加されています。
  - 索引をオプティマイザで参照できないように変更するための INVISIBLE パラメータが新しく追加されています。
  - 「PARAMETERS 句」（10-74 ページ）でドメイン索引と同様に XMLIndex 索引も再作成することができるようになりました。
- ALTER SYSTEM（11-52 ページ）には、次の拡張が行われています。
  - Oracle Real Application Clusters（Oracle RAC）環境で別のインスタンスのセッションを停止するための構文が新しく追加されています。
  - 移行用に自動ストレージ管理クラスタを作成し、すべてのノードが同じソフトウェア・バージョンに移行された後、通常の操作に戻すための rolling_migration_clauses（11-59 ページ）が新しく追加されています。
- ALTER TABLE（12-2 ページ）には、次の拡張が行われています。
  - DEFAULT 値を指定した場合の add_column_clause（12-39 ページ）の動作が拡張され、パフォーマンスが向上しています。
  - READ ONLY | READ WRITE（12-37 ページ）の構文によって、表のメンテナンス中に DDL または DML によって変更されないように表を読取り専用モードにした後で、読取り / 書込みモードにすることができます。
  - add_table_partition 句（12-58 ページ）の構文が拡張され、システム・パーティションを追加することができるようになりました。
  - flashback_archive_clause（12-37 ページ）を使用して、表の履歴追跡を使用可能または使用禁止にすることができます。
  - add_column_clause（12-39 ページ）を使用して、表に仮想列を追加することができるようになりました。
  - XMLSchema を追加または削除するために、XMLType 表を変更する構文が新しく追加されています。
  - レンジ・パーティション表を期間パーティション表に変換するための alter_interval_partitioning 句（12-53 ページ）が新しく追加されています。
  - データベースで、表におけるパーティションの様々なメンテナンス操作を参照パーティション表の子表に対してカスケードするための dependent_tables_clause（12-68 ページ）が新しく追加されています。
- ALTER TABLESPACE（12-82 ページ）に、一時表領域または個々の一時ファイルに使用されている領域を縮小するための構文が新しく追加されています。
- ASSOCIATE STATISTICS（13-21 ページ）の構文を使用して、システム管理されたドメイン索引で収集された統計情報の記憶域がデータベースによって管理されるように指定することができます。
- AUDIT（13-25 ページ）に、データ・マイニング・モデルでの様々なアクティビティを監査するための構文が新しく追加されています。
- CALL（13-37 ページ）では、ルーチンに引数が必要な場合に、コールされているルーチンに対する引数に位置表記法、名前表記法および混合表記法を使用することができるようになりました。
- COMMENT（13-41 ページ）に、データ・マイニング・モデルに関する説明コメントを指定するための MINING MODEL 句が新しく追加されています。

- CREATE DISKGROUP（14-40 ページ）および ALTER DISKGROUP（10-46 ページ）に、ディスク・グループの様々な属性を設定するための構文が新しく追加されています。
- 表の変更履歴を追跡できるフラッシュバック・データ・アーカイブを作成、変更および削除するための CREATE FLASHBACK ARCHIVE 文（14-45 ページ）、ALTER FLASHBACK ARCHIVE 文（10-60 ページ）および DROP FLASHBACK ARCHIVE 文（17-41 ページ）が新しく追加されています。
- CREATE INDEX（14-50 ページ）には、次の拡張が行われています。
  - ローカル・パーティション・ドメイン索引を作成するための local_domain_index_clause（14-66 ページ）が新しく追加されています。
  - index_attributes（14-60 ページ）が変更され、オプティマイザで参照できない索引を作成できるようになりました。
  - XML データ用の XMLIndex 索引を作成するための XMLIndex_clause（14-66 ページ）が新しく追加されています。
- CREATE INDEXTYPE（14-73 ページ）および ALTER INDEXTYPE（10-82 ページ）を使用して、サブジェクト索引タイプに基づいて作成されたドメイン索引をレンジ・パーティションにすることができ、このドメイン索引の記憶表およびパーティションのメンテナンス操作がデータベースによって管理されるよう指定することができます。
- CREATE PFILE（15-43 ページ）に、現行のシステム全体のパラメータ設定からパラメータ・ファイルを作成するための構文が新しく追加されています。
- CREATE RESTORE POINT（15-53 ページ）に、指定された日時または過去の SCN に対するリストア・ポイントを作成して、フラッシュバック・データベースを保持するための構文が新しく追加されています。
- CREATE SPFILE（15-68 ページ）に、現行のシステム全体のパラメータ設定からシステム・パラメータ・ファイルを作成するための構文が新しく追加されています。
- CREATE TABLE（16-6 ページ）には、次の拡張が行われています。
  - flashback_archive_clause（16-56 ページ）を使用して、変更履歴の追跡が使用可能な表を作成することができます。
  - system_partitioning 句（16-52 ページ）を使用して、`BY SYSTEM` 表をパーティション化することができます。
  - 仮想列を作成するための virtual_column_definition（16-27 ページ）が新しく追加されています。
  - XML データをバイナリ形式の XML 書式で格納するための XML 記憶域用の構文が新しく追加されています。
  - 別のパーティション表への参照ごとに表をパーティション化するための reference_partitioning 句（16-48 ページ）が新しく追加されています。
  - LOB_parameters（16-38 ページ）に、より高速かつ効率的で、LOB の圧縮、暗号化、複製などの新機能を使用できる LOB 用の記憶域を新しく指定するための `SECUREFILE` パラメータが新しく追加されています。
  - SecureFile 記憶域を使用した LOB に対するサーバー側の LOB 圧縮を使用可能または使用禁止にするための LOB_compression_clause（16-40 ページ）が新しく追加されています。
  - 重複データを単一の共有リポジトリ内で結合して、記憶域の消費を削減し、SecureFile 記憶域を使用した LOB の記憶域管理を簡略化するための LOB_deduplicate_clause（16-39 ページ）が新しく追加されています。
  - LOB_parameters（16-38 ページ）に、SecureFile 記憶域を使用した LOB の LOB 列の暗号化を使用可能または使用禁止にするための `ENCRYPT` 句および `DECRYPT` 句が新しく追加されています。

- CREATE TABLESPACE（16-71 ページ）に、storage_clause（8-43 ページ）に新しく追加された ENCRYPT キーワードとともに使用して表領域全体を暗号化するための構文が新しく追加されています。
- DROP DISKGROUP（17-39 ページ）に、自動ストレージ管理インスタンスによってマウントされなくなったディスク・グループを削除するための FORCE キーワードが新しく追加されています。
- GRANT（18-31 ページ）に、権限受領者がデータ・マイニング・モデルを使用して作業することができる様々なシステム権限およびオブジェクト権限が新しく追加されています。
- LOCK TABLE（18-69 ページ）に、表に対する DML ロックを取得するまでに文が待機する最大秒数を指定するための構文が新しく追加されています。
- MERGE（18-72 ページ）では、ドメイン索引が含まれる表での操作がサポートされるようになりました。
- SELECT（19-4 ページ）に、行を列に変換するための PIVOT 構文が新しく追加されています。また、列を行に変換するデータを問い合せるための UNPIVOT 操作も新しく追加されています。

次の SQL 組込みファンクションが追加または拡張されています。

- キューブまたはディメンションのデータを抽出し、2 次元形式のリレーショナル表に戻すための組込みファンクション CUBE_TABLE（5-47 ページ）が新しく追加されています。
- INSERTXMLAFTER（5-82 ページ）では、属性ノードでないターゲット・ノードの直後に任意の種類の 1 つ以上のノードを追加できます。
- REGEXP_INSTR（5-144 ページ）および REGEXP_SUBSTR（5-149 ページ）に、正規表現の特定のサブストリングを評価対象とするためのオプションの *subexpr* パラメータが追加されています。
- ソース文字列内で指定されている正規表現パターンの繰返し回数をカウントするための組込みファンクション REGEXP_COUNT（5-143 ページ）が新しく追加されています。
- PREDICTION（5-125 ページ）、PREDICTION_COST（5-128 ページ）および PREDICTION_SET（5-133 ページ）が拡張されています。格納されているコスト・マトリックスを使用するか（使用可能な場合のみ）、コスト・マトリックスを表内で指定するかを指定する構文が新しく追加されています。
- 予測に対する確度の下限および上限を戻す組込みファンクション PREDICTION_BOUNDS（5-127 ページ）が新しく追加されています。
- XML データを SQL スカラー・データ型にキャストして、XQuery 式が空でない XQuery シーケンスを戻すかどうかをそれぞれ確認するためのファンクション XMLCAST（5-229 ページ）および XMLEXISTS（5-237 ページ）が新しく追加されています。
- 対応する XMLDiff および XMLPatch C API への SQL インタフェースを提供するファンクション XMLDIFF（5-233 ページ）および XMLPATCH（5-239 ページ）が新しく追加されています。これらのファンクションを使用すると、2 つの XMLType 文書を比較し、diff ファイルを使用して XMLType 文書を修正できます。

その他の追加された変更は次のとおりです。

- 以前のリリースでは、第 6 章「式」の式の書式に**変数式**がありました。この書式の名前は、ドキュメント・セットの他のマニュアルに合わせて**プレースホルダ式**に変更されています。6-13 ページの「プレースホルダ式」を参照してください。
- 以前のリリースでは、TABLE および CLUSTER の区切られた構文のブランチを持つ単一文として TRUNCATE 文がありました。このコマンドは、他のトップレベル SQL 文に合わせて TRUNCATE CLUSTER（19-56 ページ）および TRUNCATE TABLE（19-58 ページ）に分割されています。実際の構文またはセマンティクスは、変更されていません。
- RESULT_CACHE_MODE 初期化パラメータの設定を上書きするためのヒント「RESULT_CACHE ヒント」（2-93 ページ）および「NO_RESULT_CACHE ヒント」（2-87 ページ）が新しく追加されています。

- 「ファンクション式」（6-10 ページ）では、式として使用されるユーザー定義ファンクションに対する引数に位置表記法、名前表記法および混合表記法を使用することができるようになりました。
- ALTER TABLE（12-2 ページ）および ALTER INDEX（10-64 ページ）の *index_partition_description* 構文を使用して、ドメイン索引のパーティションに対してパラメータを指定することができるようになりました。
- Oracle Multimedia でサポートされるオブジェクト型が新しく追加されています。2-35 ページの「ORDDicom」を参照してください。

1

概要

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

この章では、次の内容を説明します。

- SQL の歴史
- SQL 規格
- 最新の機能拡張
- Enterprise Manager の使用方法
- 字句規則
- ツール製品のサポート

SQL の歴史

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

SQL 規格

オラクル社は、業界標準に準拠するよう努力し、SQL 標準化委員会にも積極的に参加しています。業界で認知されている委員会には、ANSI（米国規格協会）、および IEC（国際電気標準会議）が電気・電子部門を担当している ISO（国際標準化機構）があります。ANSI と ISO/IEC はともに、SQL をリレーショナル・データベースの標準言語として認めています。新しい SQL 規格がこの両機関から同時に発表された場合、その規格の名前は、各機関の規則に従って付けられますが、技術的な詳細は同じです。

2003 年 7 月に採用された最新の SQL 規格を SQL:2003 といいます。SQL 規格の 1 つである Part 14: SQL/XML (ISO/IEC 9075-14) は 2006 年に改訂されており、SQL/XML:2006 と呼ばれます。SQL/XML を除いて、この規格の正式名称は、次のとおりです。

- ANSI/ISO/IEC 9075:2003、『Database Language SQL』、Part 1「SQL/Framework」、Part 2「SQL/Foundation」、Part 3「SQL/CLI」、Part 4「SQL/PSM」、Part 9「SQL/MED」、Part 10「SQL/OLB」、Part 11「SQL/Schemata」、Part 13「SQL/JRT」
- ISO/IEC 9075:2003、『Database Language SQL』、Part 1「SQL/Framework」、Part 2「SQL/Foundation」、Part 3「SQL/CLI」、Part 4「SQL/PSM」、Part 9「SQL/MED」、Part 10「SQL/OLB」、Part 11「SQL/Schemata」、Part 13「SQL/JRT」

> **参照：** SQL:2003 規格への Oracle Database の規格準拠については、付録 B「Oracle と標準 SQL」を参照してください。

改訂された Part14 の正式名称は、次のとおりです。

- ANSI/ISO/IEC 9075-14:2006、『Database Language SQL』、Part 14「SQL/XML」
- ISO/IEC 9075-14:2006、『Database Language SQL』、Part 14「SQL/XML」

SQL の特長

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

- 個々の単位としてではなく、グループとして一連のデータを処理します。
- データへの自動的なナビゲーション・アクセス（経路設定）を実行します。
- SQL で使用する文は、それぞれが複雑、強力で、スタンドアロン型の文です。これまで、SQL にはフロー制御文は含まれていませんでしたが、最近認められた SQL のオプション部分 ISO/IEC 9075-5 : 1996 にはフロー制御文が含まれています。フロー制御文は、永続保存モジュール（PSM）としてよく知られており、SQL の拡張機能である Oracle の PL/SQL は、PSM に似ています。

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

SQL 文を使用して、次の処理を行うことができます。

- データの問合せ
- 表の中の行の挿入、更新、削除
- オブジェクトの作成、置換、変更、削除
- データベースとデータベース・オブジェクトへのアクセス制御
- データベースの一貫性と整合性の保証

SQL では、前述のすべてのタスクを 1 つの一貫性のある言語に統一しました。

すべてのリレーショナル・データベースに共通の言語

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

最新の機能拡張

Oracle Database の SQL エンジンは、すべての Oracle Database アプリケーションの基礎となっています。Oracle SQL は、データベース・アプリケーションの多様化するニーズを満たし、先端の計算アーキテクチャ、API およびネットワーク・プロトコルをサポートするために発展し続けています。

SQL では、従来の構造化データに加え、より複雑なデータの格納、取出しおよび処理を実行できます。

- オブジェクト型、コレクション型および REF 型は、複雑な構造化データをサポートします。ネストした表コレクション型に、標準準拠の多くの MULTISET 演算子がサポートされています。
- ラージ・オブジェクト（LOB）は、構造化されていない文字データおよびバイナリ・データの両方をサポートします。1 つの LOB の最大サイズは、データベースのブロック・サイズに応じて 8 ～ 128TB になります。
- `XMLType` データ型は、半構造化 XML データをサポートします。

標準ベース機能のネイティブ・サポートに、次のものが含まれています。

- 正規表現のネイティブ・サポート。これによって、データベース内の自由形式のテキストをパターン検索したり、操作することができます。
- IEEE754 標準に準拠したネイティブ浮動小数点データ型。これによって、XML および Java 標準で一般的な浮動小数点の処理のパフォーマンスが向上し、数値データに必要な記憶域が減少します。
- 組込みの SQL 集計ファンクションおよび分析ファンクション。これらを使用すると、データ・ウェアハウスおよびデータ・マート内のデータに簡単にアクセスし、操作できます。

Oracle SQL は、多目的でスケーラブルかつ高パフォーマンスのデータベース・アプリケーションの開発を総合的にサポートするために、継続して機能が拡張されています。

Enterprise Manager の使用方法

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

字句規則

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

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

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

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

予約語、キーワード、識別子およびパラメータは、大文字と小文字を区別せずに記述できます。ただし、テキスト・リテラルと引用符で囲んだ名前では、大文字と小文字は区別されます。テキスト・リテラルの構文の詳細は、2-44 ページの「テキスト・リテラル」を参照してください。

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

ツール製品のサポート

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

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

- Oracle Application Express は、データベース関連の Web アプリケーションの開発およびデプロイ用のホストされた環境です。Oracle Application Express のコンポーネントである SQL Workshop を使用すると、Web ブラウザからデータベース・オブジェクトを表示および管理できます。SQL Workshop では、SQL コマンド・プロセッサおよび SQL スクリプト・リポジトリに素早くアクセスできます。

  参照： これらの製品の詳細は、『SQL*Plus ユーザーズ・ガイドおよびリファレンス』および『Oracle Database Application Express ユーザーズ・ガイド』を参照してください。

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

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

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

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

2

Oracle SQL の基本要素

この章では、Oracle SQL の基本要素に関する参照情報を説明します。これらの要素は、SQL 文の最も単純な構成ブロックです。したがって、第 10 章～第 19 章で説明されている文を使用する前に、この章で説明する概念を理解しておく必要があります。

この章では、次の内容を説明します。

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

データ型

Oracle Database が処理する値は、それぞれ**データ型**を持ちます。値のデータ型は、固定されたプロパティの集合をその値に対応付けます。このプロパティに応じて、Oracle は、あるデータ型の値を別のデータ型の値と区別して扱います。たとえば、NUMBER データ型の値は加算できますが、RAW データ型の値は加算できません。

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

Oracle Database には、多くの組込みデータ型、およびデータ型として使用できるいくつかのユーザー定義型のカテゴリがあります。Oracle のデータ型の構文については、次の構文図で示します。この項は、次の 4 つの項にわかれています。

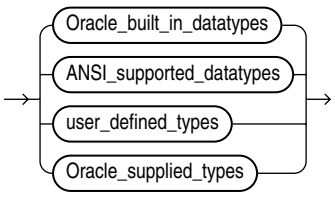
- Oracle の組込みデータ型
- ANSI、DB2、SQL/DS のデータ型
- ユーザー定義型
- Oracle が提供する型
- データ型の比較規則
- データ変換

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

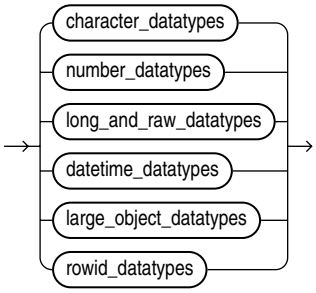
参照： 「LOB 列の制限事項 :」（2-25 ページ）

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

***datatypes*::=**

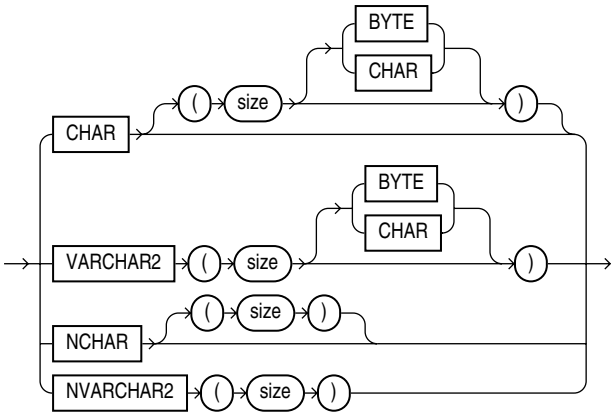


***Oracle_built_in_datatypes*::=**

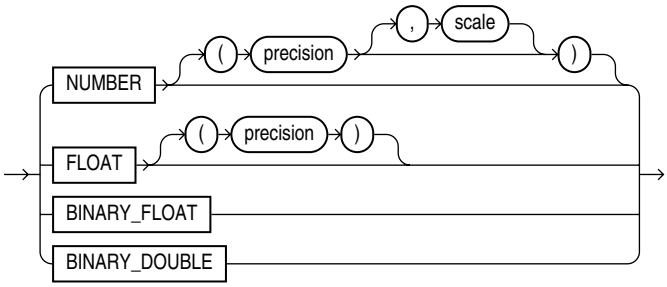


Oracle 組込みデータ型の詳細は、2-7 ページの「Oracle の組込みデータ型」を参照してください。

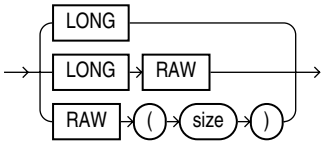
***character_datatypes*::=**



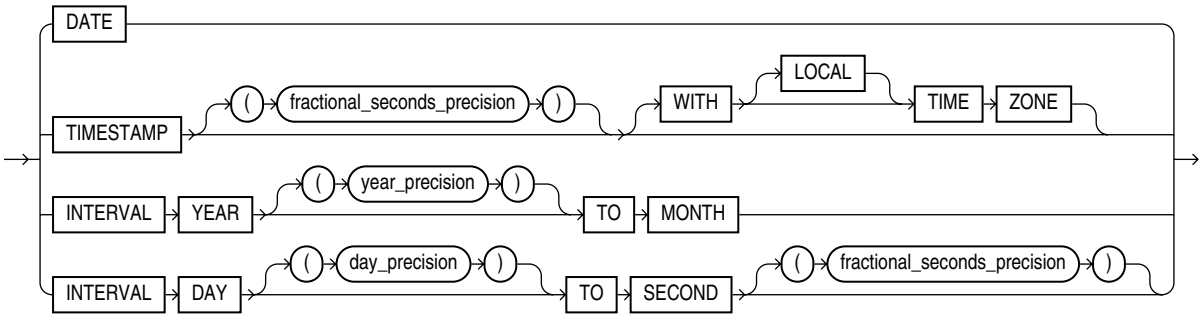
***number_datatypes*::=**



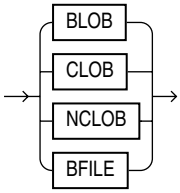
***long_and_raw_datatypes*::=**



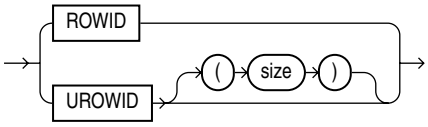
***datetime_datatypes*::=**



***large_object_datatypes*::=**

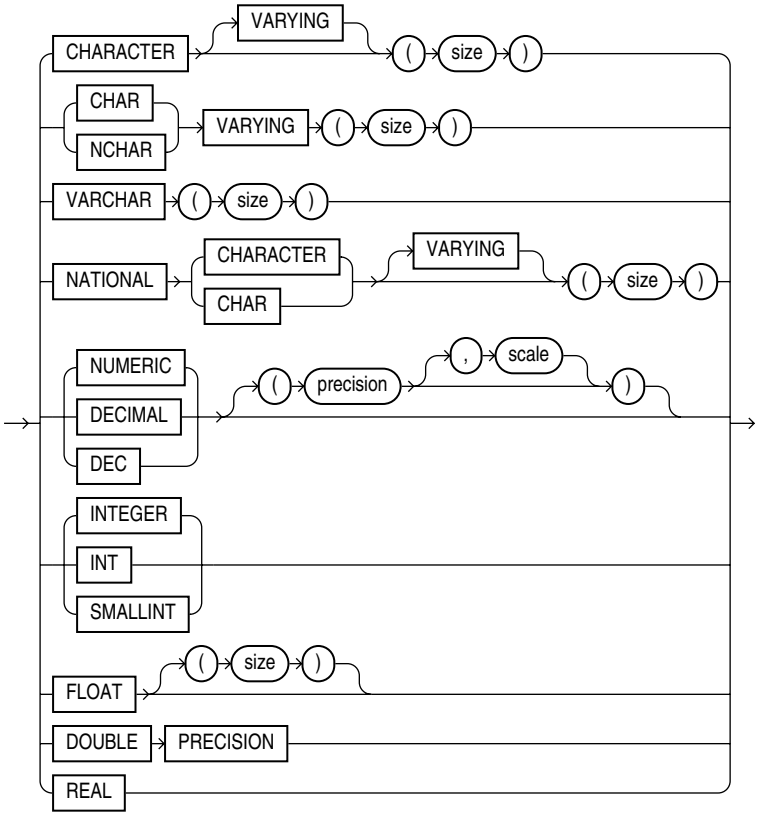


***rowid_datatypes*::=**

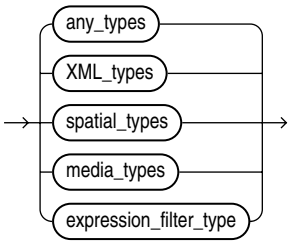


次に、ANSI がサポートしているデータ型について示します。また、2-27 ページの「ANSI、DB2、SQL/DS のデータ型」に、ANSI がサポートしているデータ型を Oracle 組込みデータ型にマップする方法を示します。

***ANSI_supported_datatypes*::=**

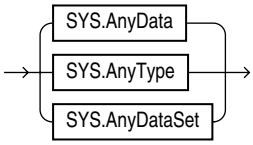


***Oracle_supplied_types*::=**



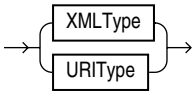
expression_filter_type の詳細は、2-36 ページの「Expression Filter 型」を参照してください。次に、Oracle が提供するその他の型を示します。

***any_types*::=**



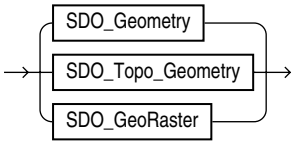
任意型の詳細は、2-31 ページの「任意型」を参照してください。

***XML_types*::=**



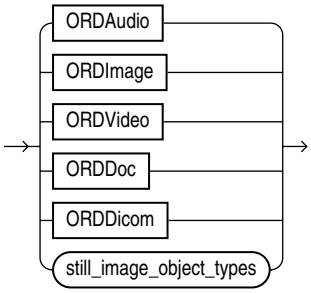
XML 型の詳細は、2-32 ページの「XML 型」を参照してください。

***spatial_types*::=**

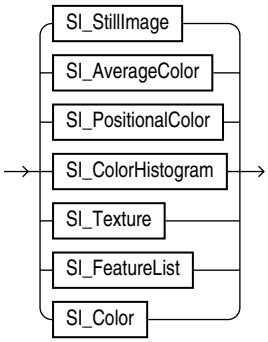


空間型の詳細は、2-33 ページの「空間型」を参照してください。

***media_types*::=**



***still_image_object_types*::=**



メディア型の詳細は、2-34 ページの「メディア型」を参照してください。

Oracle の組込みデータ型

次の表に Oracle 組込みデータ型の概要を示します。構文要素については、前の項の構文を参照してください。データ型のコードは、Oracle Database が内部的に使用します。DUMP ファンクションによって、列またはオブジェクト属性のデータ型コードが戻されます。

表 2-1 組込みデータ型の概要

コード	データ型	説明
1	VARCHAR2(*size* [BYTE | CHAR])	最大長が *size* バイトまたは size 文字の可変長文字列。最大サイズは 4000、最小サイズは 1 です。VARCHAR2 には、*size* を指定する必要があります。 BYTE は、列がバイト長セマンティクスを持つことを示します。CHAR は、列がキャラクタ・セマンティクスを持つことを示します。
1	NVARCHAR2(*size*)	最大長が *size* 文字の可変長 Unicode 文字列。バイト数は、AL16UTF16 エンコードの場合は *size* の 2 倍まで、UTF8 エンコードの場合は *size* の 3 倍までです。最大サイズは、各国語キャラクタ・セット定義（上限 4000 バイト）によって決定されます。NVARCHAR2 には、*size* を指定する必要があります。
2	NUMBER [(*p* [, *s*])]	精度 *p*、位取り *s* を持つ数。精度 *p* には 1 ～ 38 の値を指定できます。位取り *s* には -84 ～ 127 の値を指定できます。精度と位取りはともに 10 進数です。NUMBER の値は 1 ～ 22 バイトである必要があります。
2	FLOAT [(*p*)]	精度 *p* を持つ NUMBER データ型のサブタイプ。FLOAT の値は、内部的に NUMBER と表されます。精度 *p* の値は、1 ～ 126 の範囲の 2 進数で指定します。FLOAT の値は 1 ～ 22 バイトである必要があります。
8	LONG	最大 2GB（2^{31} から 1 を引いたバイト数）の可変長文字データ。下位互換性のために提供されています。
12	DATE	紀元前 4712 年 1 月 1 日～紀元 9999 年 12 月 31 日までの日付を指定します。デフォルトの書式は、NLS_DATE_FORMAT パラメータによって明示的に決まるか、または NLS_TERRITORY パラメータによって暗黙的に決まります。サイズは 7 バイトに固定されています。このデータ型には、YEAR、MONTH、DAY、HOUR、MINUTE および SECOND の日時フィールドがあります。秒の小数部とタイムゾーンはありません。
21	BINARY_FLOAT	32 ビットの浮動小数点数。このデータ型には、長さを示すバイトを含め、5 バイトが必要です。
22	BINARY_DOUBLE	64 ビットの浮動小数点数。このデータ型には、長さを示すバイトを含め、9 バイトが必要です。
180	TIMESTAMP [(*fractional_seconds_precision*)]	日付の年、月、日および時刻の時、分、秒の値。*fractional_seconds_precision* は、SECOND 日時フィールドの小数部の桁数です。*fractional_seconds_precision* の有効範囲は、0 ～ 9 です。デフォルトは 6 です。デフォルトの書式は、NLS_DATE_FORMAT パラメータによって明示的に決まるか、または NLS_TERRITORY パラメータによって暗黙的に決まります。サイズは、精度によって 7 ～ 11 バイトの長さで変化します。このデータ型には、YEAR、MONTH、DAY、HOUR、MINUTE および SECOND の日時フィールドがあります。秒の小数部はありますが、タイムゾーンはありません。

表 2-1 組込みデータ型の概要（続き）

コード	データ型	説明
181	TIMESTAMP [(*fractional_seconds*)] WITH TIME ZONE	タイムゾーンによる時差などのすべての TIMESTAMP の値。*fractional_seconds_precision* は、SECOND 日時フィールドの小数部の桁数です。有効範囲は 0 ～ 9 です。デフォルトは 6 です。デフォルトの書式は、NLS_DATE_FORMAT パラメータによって明示的に決まるか、または NLS_TERRITORY パラメータによって暗黙的に決まります。サイズは 13 バイトに固定されています。このデータ型には、YEAR、MONTH、DAY、HOUR、MINUTE、SECOND、TIMEZONE_HOUR および TIMEZONE_MINUTE の日時フィールドがあります。秒の小数部と明示的なタイムゾーンがあります。
231	TIMESTAMP [(*fractional_seconds*)] WITH LOCAL TIME ZONE	TIMESTAMP WITH TIME ZONE のすべての値。ただし、次に示す例外があります。 ■ データベースへの格納時、データがデータベースのタイムゾーンに正規化されている場合。 ■ データの検索時、ユーザーがセッションのタイムゾーンでデータを検索する場合。 デフォルトの書式は、NLS_DATE_FORMAT パラメータによって明示的に決まるか、または NLS_TERRITORY パラメータによって暗黙的に決まります。サイズは、精度によって 7 ～ 11 バイトの長さで変化します。
182	INTERVAL YEAR [(*year_precision*)] TO MONTH	年および月で期間を格納。*year_precision* は、YEAR 日時フィールドの桁数です。有効範囲は 0 ～ 9 です。デフォルトは 2 です。サイズは 5 バイトに固定されています。
183	INTERVAL DAY [(*day_precision*)] TO SECOND [(*fractional_seconds*)]	日、時、分および秒で期間を格納。 ■ *day_precision* は、DAY 日時フィールドの最大桁数です。有効範囲は 0 ～ 9 です。デフォルトは 2 です。 ■ *fractional_seconds_precision* は、SECOND フィールドの小数部の桁数です。有効範囲は 0 ～ 9 です。デフォルトは 6 です。 サイズは 11 バイトに固定されています。
23	RAW(*size*)	長さ *size* バイトのバイナリ・データ。最大サイズは 2000 バイトです。RAW 値には、*size* を指定する必要があります。
24	LONG RAW	最大 2GB の可変長バイナリ・データ。
69	ROWID	表の行のアドレスを一意に表す BASE64 文字列。主に、ROWID 疑似列によって戻される値のためのデータ型です。
208	UROWID [(*size*)]	索引構成表の行の論理アドレスを表す BASE64 文字列。オプションの *size* は、UROWID 型の列のサイズです。最大サイズおよびデフォルトは 4000 バイトです。
96	CHAR [(*size* [BYTE | CHAR])]	長さ *size* バイトまたは文字の固定長文字データ。最大サイズは 2000 です。デフォルトおよび最小サイズは 1 です。 BYTE および CHAR は、VARCHAR2 と同じセマンティクスを持ちます。
96	NCHAR[(*size*)]	長さ *size* 文字の固定長文字データ。バイト数は、AL16UTF16 エンコードの場合は *size* の 2 倍まで、UTF8 エンコードの場合は *size* の 3 倍までです。最大サイズは、各国語キャラクタ・セット定義（上限 2000 バイト）によって決定されます。デフォルトおよび最小サイズは 1 です。

表 2-1 組込みデータ型の概要（続き）

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

次の項では、Oracle Database に格納される Oracle データ型について説明します。これらのデータ型をリテラルとして指定する方法については、2-44 ページの「リテラル」を参照してください。

文字データ型

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

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

次のデータ型が、文字データに対して使用されます。

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

文字データ型をリテラルとして指定する方法については、2-44 ページの「テキスト・リテラル」を参照してください。

CHAR データ型

CHAR データ型は、固定長の文字列を指定します。Oracle では、CHAR 列に格納される値がすべて *size* で指定した長さを持つように調整されます。列の長さより短い値を挿入すると、列の長さにあわせるため、その値の後に空白が埋め込まれます。列に対して長すぎる値を挿入しようとすると、Oracle はエラーを戻します。

CHAR 列のデフォルトの長さは 1 バイトで、この許容最大値は 2000 バイトです。CHAR(10) 列には、1 バイトの文字列を挿入できますが、この文字列は 10 バイトまで空白埋めされてから格納されます。

CHAR 列を持つ表を作成する場合、デフォルトでは列の長さはバイト単位になります。BYTE 修飾子は、デフォルトと同じです。CHAR 修飾子（たとえば、CHAR(10 CHAR)）を使用すると、列の長さは文字単位になります。技術的ないい方をすると、文字は、データベース・キャラクタ・セットのコード・ポイントです。サイズは、データベース・キャラクタ・セットによって異なりますが、1 バイトから 4 バイトです。BYTE および CHAR 修飾子は、NLS_LENGTH_SEMANTICS パラメータ（デフォルトはバイト・セマンティクス）で指定したセマンティクスを上書きします。パフォーマンス上の理由から、長さセマンティクスの設定には NLS_LENGTH_SEMANTICS パラメータを使用し、このパラメータを上書きする必要があるときにのみ BYTE および CHAR 修飾子を使用することをお薦めします。

異なるキャラクタ・セットを持つデータベース間で適切にデータを変換するには、CHAR データが正しい書式の文字列で構成されていることを確認してください。

> **参照：** キャラクタ・セット・サポートの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。比較セマンティクスについては、2-36 ページの「データ型の比較規則」を参照してください。

NCHAR データ型

NCHAR データ型は Unicode のみのデータ型です。NCHAR 列を持つ表を作成する場合、列の長さを文字単位で指定します。使用する各国語キャラクタ・セットは、データベースを作成するときに指定します。

列の最大長は、各国語キャラクタ・セットの定義によって決まります。NCHAR 文字データ型の幅指定は、文字数を示します。許容最大列サイズは 2000 バイトです。

列の長さより短い値を挿入すると、列の長さにあわせるため、その値の後に空白が埋め込まれます。CHAR 値を NCHAR 列に挿入することや、NCHAR 値を CHAR 列に挿入することはできません。

次の例では、表 pm.product_descriptions の translated_description 列と各国語キャラクタ・セットの文字列を比較します。

```
SELECT translated_description FROM product_descriptions
   WHERE translated_name = N'LCD Monitor 11/PM';
```

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

NVARCHAR2 データ型

NVARCHAR2 データ型は Unicode のみのデータ型です。NVARCHAR2 列を持つ表を作成する場合、保持できる最大の文字数を指定します。Oracle では、列の最大長を超えないかぎり、各値を指定されたとおりに正確に列に格納します。

列の最大長は、各国語キャラクタ・セットの定義によって決まります。NVARCHAR2 文字データ型の幅指定は、文字数を示します。許容最大列サイズは 4000 バイトです。

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

VARCHAR2 データ型

VARCHAR2 データ型は、可変長の文字列を指定します。VARCHAR2 列を作成する場合、保持できるデータの最大バイト数または最大文字数を指定します。Oracle では、列の最大長を超えないかぎり、各値を指定されたとおりに正確に列に格納します。最大長を超える値を挿入しようとすると、Oracle はエラーを戻します。

VARCHAR2 列には最大長を指定する必要があります。保存される文字列の実際の長さは 0（ゼロ）にできますが、最大長は 1 バイト以上にする必要があります。CHAR 修飾子（たとえば、VARCHAR2(10 CHAR)）を使用すると、バイトではなく、文字で最大長を指定できます。技術的ないい方をすると、文字は、データベース・キャラクタ・セットのコード・ポイントです。BYTE 修飾子（たとえば、VARCHAR2(10 BYTE)）を使用すると、明示的にバイトで最大長を指定できます。列定義または属性定義に明示的な修飾子が含まれていないときにその列または属性を持つデータベース・オブジェクトを作成した場合、長さセマンティクスは、そのオブジェクトを作成するセッションの NLS_LENGTH_SEMANTICS パラメータの値によって決定されます。文字での最大長と関係なく、VARCHAR2 データの長さは 4000 バイト以下になります。Oracle は、非空白埋め比較セマンティクスを使用して VARCHAR2 値を比較します。

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

> **参照：** 比較セマンティクスについては、2-36 ページの「データ型の比較規則」を参照してください。

VARCHAR データ型

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

数値データ型

Oracle Database の数値データ型は、正と負の固定小数点数と浮動小数点数、0（ゼロ）、無限大、および操作の未定義の結果である値（非数値、つまり NAN）を格納します。数値データ型をリテラルとして指定する方法については、2-46 ページの「数値リテラル」を参照してください。

NUMBER データ型

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

次の書式で固定小数点数を指定できます。

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

それぞれの意味は、次のとおりです。

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

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

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

次の書式で整数を指定できます。

```
NUMBER(p)
```

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

次の書式で浮動小数点を指定できます。

```
NUMBER
```

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

参照： 「浮動小数点数」（2-13 ページ）

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

表 2-2 精度および位取りの格納

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

FLOAT データ型

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

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

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

```
CREATE TABLE test (col1 NUMBER(5,2), col2 FLOAT(5));

INSERT INTO test VALUES (1.23, 1.23);
INSERT INTO test VALUES (7.89, 7.89);
INSERT INTO test VALUES (12.79, 12.79);
INSERT INTO test VALUES (123.45, 123.45);

SELECT * FROM test;
      COL1       COL2
---------- ----------
      1.23        1.2
      7.89        7.9
     12.79         13
    123.45        120
```

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

ANSI の FLOAT データを変換する際、Oracle Database は、Oracle の FLOAT データ型を内部的に使用します。Oracle の FLOAT は使用可能ですが、BINARY_FLOAT および BINARY_DOUBLE データ型の方がより堅牢であるため、これらのデータ型を使用することをお薦めします。詳細は、2-13 ページの「浮動小数点数」を参照してください。

浮動小数点数

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

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

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

BINARY_FLOAT 32 ビットの単精度浮動小数点数データ型です。このデータ型の各値には、長さを示すバイトを含め、5 バイトが必要です。

BINARY_DOUBLE 64 ビットの倍精度浮動小数点数データ型です。このデータ型の各値には、長さを示すバイトを含め、9 バイトが必要です。

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

2-14 ページの表 2-3 に示す制限内で、浮動小数点数を指定できます。浮動小数点数を指定する書式については、2-46 ページの「数値リテラル」を参照してください。

表 2-3 浮動小数点数の制限

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

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

- SQL ファンクション SQRT は平方根を実装します。5-167 ページの「SQRT」を参照してください。
- SQL ファンクション REMAINDER は余りを実装します。5-155 ページの「REMAINDER」を参照してください。
- 算術演算子が準拠しています。4-3 ページの「算術演算子」を参照してください。
- 比較演算子が準拠しています。ただし、NaN と比較する場合は除きます。Oracle は、NaN が他のすべての値より大きいとみなし、NaN と NaN は等しいと評価します。7-7 ページの「浮動小数点条件」を参照してください。
- 変換演算子が準拠しています。5-5 ページの「変換ファンクション」を参照してください。
- デフォルトの丸めモードがサポートされています。
- デフォルトの例外処理モードがサポートされています。
- 特殊な値である INF、-INF および NaN がサポートされています。7-7 ページの「浮動小数点条件」を参照してください。
- BINARY_FLOAT および BINARY_DOUBLE の値を整数値の BINARY_FLOAT および BINARY_DOUBLE に丸める方法として、SQL ファンクション ROUND、TRUNC、CEIL および FLOOR が提供されています。
- BINARY_FLOAT および BINARY_DOUBLE を 10 進値に、10 進値を BINARY_FLOAT および BINARY_DOUBLE に丸める方法として、SQL ファンクション TO_CHAR、TO_NUMBER、TO_NCHAR、TO_BINARY_FLOAT、TO_BINARY_DOUBLE および CAST が提供されています。

浮動小数点データ型は次の点では IEEE754 に準拠していません。

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

数値の優先順位

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

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

暗黙的な変換が必要な場合に変換に失敗すると、操作は実行されません。暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

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

LONG データ型

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

LONG 列には、2GB（2^{31}）から 1 を引いたバイト数までの可変長の文字列を格納できます。LONG 列には、多くの点で VARCHAR2 列と同じ特長があります。LONG 列を使用すると、長いテキスト文字列を格納できます。LONG 値の長さは、ご使用のコンピュータで利用できるメモリーによって制限される場合もあります。LONG リテラルは、2-44 ページの「テキスト・リテラル」の説明のような形式になります。

既存の LONG 列も LOB 列に変換することをお薦めします。LOB 列は、LONG 列ほど制限は多くありません。LOB 機能はリリースごとに拡張されていますが、LONG 機能は最近のリリースでは変更されていません。LONG 列から LOB 列への変換については、12-2 ページの「ALTER TABLE」の「*modify_col_properties*」および 5-206 ページの「TO_LOB」を参照してください。

SQL 文の中の次の場所で LONG 列を参照できます。

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

LONG 値を使用する場合には、次の制限があります。

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

- LONG 列および LONG RAW 列は、分散型の SQL 文で使用できません。また、レプリケートできません。
- LONG と LOB の両方の列を持つ表では、1 つの SQL 文で両方に 4000 バイトより大きいデータをバインドすることはできません。ただし、いずれか片方にバインドすることは可能です。

また、LONG 列は SQL 文の次のような部分では使用できません。

- GROUP BY 句、ORDER BY 句、CONNECT BY 句または SELECT 文にある DISTINCT 演算子
- SELECT 文の一意演算子
- CREATE CLUSTER 文の列リスト
- CREATE MATERIALIZED VIEW 文の CLUSTER 句
- SQL 組込みファンクション、式または条件
- GROUP BY 句を含む問合せの SELECT 構文のリスト
- UNION、INTERSECT または MINUS 集合演算子によって結合されている副問合せまたは問合せの SELECT 構文のリスト
- CREATE TABLE ...AS SELECT 文の SELECT 構文のリスト
- ALTER TABLE ...MOVE 文
- INSERT 文の副問合せの SELECT 構文のリスト

トリガーでは、LONG データ型は次のように使用されます。

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

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

参照：『Oracle Call Interface プログラマーズ・ガイド』

日時および期間データ型

日時データ型には、DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONE および TIMESTAMP WITH LOCAL TIME ZONE があります。日時データ型の値は、「**日時**」とも呼ばれます。期間データ型には、INTERVAL YEAR TO MONTH および INTERVAL DAY TO SECOND があります。期間データ型の値は、「**期間**」とも呼ばれます。日時値と期間値をリテラルとして表す方法の詳細は、2-48 ページの「日時リテラル」および 2-51 ページの「期間リテラル」を参照してください。

日時および期間はいずれもフィールドで構成されます。これらのフィールドの値は、データ型の値によって決まります。表 2-4 に、日時フィールド、および日時と期間の有効な値を示します。

日時データの DML 操作で正しい結果を得るには、組込み SQL ファンクション DBTIMEZONE および SESSIONTIMEZONE で問い合せることによって、データベースおよびセッションのタイムゾーンを確認します。タイムゾーンを手動で設定していない場合、Oracle Database は、オペレーティング・システムのタイムゾーンをデフォルトで使用します。オペレーティング・システムのタイムゾーンが Oracle で有効でない場合は、Oracle は、協定世界時（UTC）（以前のグリニッジ標準時）をデフォルトの値として使用します。

表 2-4 日時フィールド、および日時と期間の値

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

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

DATE データ型

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

DATE 値をリテラルに指定するか、文字値や数値を TO_DATE ファンクションによって日付値に変換できます。これらの方法での DATE 値の表し方の例は 2-48 ページの「日時リテラル」を参照してください。

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

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

デフォルトの日付値は次のように決まります。

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

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

```
SELECT TO_DATE('2005', 'YYYY') FROM DUAL;

TO_DATE('
---------
01-MAY-05
```

例 次の文では、1997 年 1 月 1 日をユリウス日で戻します。

```
SELECT TO_CHAR(TO_DATE('01-01-1997', 'MM-DD-YYYY'),'J')
    FROM DUAL;

TO_CHAR
--------
2450450
```

> **参照：** DUAL 表については、「DUAL 表からの選択」を参照してください。

TIMESTAMP データ型

TIMESTAMP データ型は、DATE データ型の拡張機能です。DATE データ型の年、月および日に加えて、時、分および秒の値を格納します。このデータ型は、正確な時刻の値の格納に有効です。TIMESTAMP データ型は、次のように指定します。

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

fractional_seconds_precision には、オプションで、Oracle が格納する桁数を、SECOND 日時フィールドの小数部まで指定します。このデータ型の列を作成する場合、0 ～ 9 の値を指定できます。デフォルト値は 6 です。

> **参照：** 文字データの TIMESTAMP データへの変換については、5-210 ページの「TO_TIMESTAMP」を参照してください。

TIMESTAMP WITH TIME ZONE データ型

TIMESTAMP WITH TIME ZONE は、**タイムゾーン地域名**または**タイムゾーン・オフセット**を値に含む TIMESTAMP の変形です。タイムゾーン・オフセットは、ローカルの時刻と UTC（時および分）との差異です。このデータ型は複数の地域にまたがる日時情報の収集および評価に有効です。

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

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

fractional_seconds_precision には、オプションで、Oracle が格納する桁数を、SECOND 日時フィールドの小数部まで指定します。このデータ型の列を作成する場合、0 ～ 9 の値を指定できます。デフォルト値は 6 です。

参照 :

- Oracle のタイムゾーン・データの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。
- 夏時間のサポートについては、2-22 ページの「夏時間のサポート」および 2-65 ページの表 2-17「FX 書式モデル修飾子による文字データと書式モデルの一致」を参照してください。
- 文字データの TIMESTAMP WITH TIME ZONE データへの変換は、5-211 ページの「TO_TIMESTAMP_TZ」を参照してください。
- ERROR_ON_OVERLAP_TIME セッション・パラメータについては、11-41 ページの「ALTER SESSION」を参照してください。

TIMESTAMP WITH LOCAL TIME ZONE データ型

TIMESTAMP WITH LOCAL TIME ZONE は、**タイムゾーン・オフセット**を値に含む TIMESTAMP のもう 1 つの変形です。これは、TIMESTAMP WITH TIME ZONE とは異なり、データベースに格納されるデータはデータベース・タイムゾーンに対して正規化され、タイムゾーン・オフセットは列データの一部として格納されません。ユーザーがデータを検索すると、Oracle はユーザーのローカル・セッション・タイムゾーンのデータを戻します。タイムゾーン・オフセットは、ローカルの時刻と UTC（時および分）との差異です。このデータ型は、2 層アプリケーションでクライアント・システムのタイムゾーンの日時情報を表示する場合に有効です。

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

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

fractional_seconds_precision には、オプションで、Oracle が格納する桁数を、SECOND 日時フィールドの小数部まで指定します。このデータ型の列を作成する場合、0 ～ 9 の値を指定できます。デフォルト値は 6 です。

参照 :

- Oracle のタイムゾーン・データの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。
- データ型の使用例は、『Oracle Database アドバンスト・アプリケーション開発者ガイド』を参照してください。文字データの TIMESTAMP WITH LOCAL TIME ZONE データへの変換は、5-25 ページの「CAST」を参照してください。

INTERVAL YEAR TO MONTH データ型

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

INTERVAL YEAR TO MONTH は、次のように指定します。

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

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

期間値をリテラルとして指定すると、高い柔軟性が得られます。期間値をリテラルとして指定する方法の詳細は、2-51 ページの「期間リテラル」を参照してください。期間の使用例は、2-22 ページの「日時および期間の例」を参照してください。

INTERVAL DAY TO SECOND データ型

INTERVAL DAY TO SECOND は、日付、時、分および秒で期間を格納します。このデータ型は、2 つの日時の値の正確な違いを表す場合に有効です。

このデータ型は、次のように指定します。

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

それぞれの意味は、次のとおりです。

- *day_precision* は、DAY 日時フィールドの桁数です。有効範囲は 0 ～ 9 です。デフォルトは 2 です。
- *fractional_seconds_precision* は、SECOND 日時フィールドの小数部の桁数です。有効範囲は 0 ～ 9 です。デフォルトは 6 です。

期間値をリテラルとして指定すると、高い柔軟性が得られます。期間値をリテラルとして指定する方法の詳細は、2-51 ページの「期間リテラル」を参照してください。期間の使用例は、2-22 ページの「日時および期間の例」を参照してください。

日時および期間の演算

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

- 日付およびタイムスタンプ値の演算処理では NUMBER 定数を使用できますが、期間値では使用できません。Oracle は、タイムスタンプ値を内部的に日付値に変換し、日時の演算で使用される NUMBER 定数と期間式を日数として解析します。たとえば、SYSDATE+ 1 は明日です。SYSDATE -7 は 1 週間前です。SYSDATE + (10/1440) は 10 分後です。SYSDATE から employees サンプル表の hire_date 列を引くと、各従業員が雇用されてから経過した日数が戻ります。日付値またはタイムスタンプ値の乗算や除算はできません。
- Oracle は、BINARY_FLOAT オペランドおよび BINARY_DOUBLE オペランドを暗黙的に NUMBER に変換します。
- 各 DATE 値には時刻コンポーネントが含まれるため、多くの場合、日付操作の結果には小数部が含まれます。この小数部は、日を単位として表されています。たとえば、1.5 日は 36 時間です。小数部は、DATE データの一般的な操作を行う Oracle 組込みファンクションによっても戻されます。たとえば、MONTHS_BETWEEN ファンクションは、2 つの日付の間の月数を戻します。結果の小数部は、月（1 か月は 31 日）を単位として表されます。
- 1 つのオペランドが DATE 値または数値であり、タイムゾーンおよび小数部コンポーネントのいずれも含まない場合、次のようになります。
  - Oracle は、他方のオペランドを暗黙的に DATE データに変換します。ただし、数値と期間を掛けて期間を戻す乗算を行う場合は除きます。
  - 他方のオペランドがタイムゾーン値を持つ場合、Oracle は戻り値でセッション・タイムゾーンを使用します。
  - 他方のオペランドが小数部の値を持つ場合、その小数部の値は失われます。
- DATE データ型専用に設計された組込みファンクションにタイムスタンプ値、期間値または数値を渡すと、Oracle は DATE 値以外の値を暗黙的に DATE 値に変換します。DATE への暗黙的な変換を実行するファンクションの詳細は、5-5 ページの「日時ファンクション」を参照してください。

- 期間の計算が日時の値を戻す場合、その結果は実際の日時の値である必要があります。実際の日時の値でない場合、データベースはエラーを戻します。たとえば、次の 2 つの文はエラーを戻します。

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

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

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

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

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

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

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

表 2-5 日時の演算のマトリックス

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

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

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

夏時間のサポート

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

このような境界を解決するために、Oracle は TZR および TZD 書式要素を使用します。詳細は、表 2-17 を参照してください。TZR は、日時の入力文字列でタイムゾーン地域を表します。たとえば、Australia/North、UTC、Singapore などです。TZD は、夏時間情報を含むタイムゾーン地域の略称書式です。たとえば、米国 / 太平洋標準時は PST、米国 / 太平洋夏時間は PDT などです。TZR および TZD 書式要素の値を表示するには、V$TIMEZONE_NAMES 動的パフォーマンス・ビューの TZNAME および TZABBREV 列に問合せを実行してください。

注意：　夏時間機能には、タイムゾーン地域名が必要です。地域名は、2 つのタイムゾーン・ファイルに格納されます。デフォルトのタイムゾーン・ファイルは、パフォーマンスを最大にするために一般的なタイムゾーンのみの小さなファイルです。タイムゾーンがデフォルトのファイルに存在しない場合は、環境変数 ORA_TZFILE を使用して完全な（大きい）ファイルへのパスを指定するまで、夏時間はサポートされません。

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

参照：

- 書式要素の詳細は、2-57 ページの「日時書式モデル」を参照してください。また、11-46 ページの「ERROR_ON_OVERLAP_TIME」セッション・パラメータの説明も参照してください。
- Oracle のタイムゾーン・データの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。
- 動的パフォーマンス・ビューの詳細は、『Oracle Database リファレンス』を参照してください。

日時および期間の例

次の例では、日時および期間データ型の指定方法を示します。

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

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

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

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

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

```
SELECT last_name, EXTRACT(YEAR FROM (SYSDATE - hire_date) YEAR TO MONTH )
   || ' years '
   || EXTRACT(MONTH FROM (SYSDATE - hire_date) YEAR TO MONTH )
   || ' months'  "Interval"
FROM employees ;

LAST_NAME                 Interval
------------------------- --------------------
King                      17 years 11 months
Kochhar                   15 years 8 months
De Haan                   12 years 4 months
Hunold                    15 years 4 months
Ernst                     14 years 0 months
Austin                    7 years 11 months
Pataballa                 7 years 3 months
Lorentz                   6 years 3 months
Greenberg                 10 years 9 months
. . .

SELECT order_id,
   EXTRACT(DAY FROM (SYSDATE - order_date) DAY TO SECOND )
   || ' days '
   || EXTRACT(HOUR FROM (SYSDATE - order_date) DAY TO SECOND )
   || ' hours' "Interval"
FROM orders;

  ORDER_ID Interval
---------- --------------------
      2458 2095 days 18 hours
      2397 2000 days 17 hours
      2454 2048 days 16 hours
      2354 1762 days 16 hours
      2358 1950 days 15 hours
      2381 1823 days 13 hours
      2440 2080 days 12 hours
      2357 2680 days 11 hours
      2394 1917 days 10 hours
      2435 2078 days 10 hours
. . .
```

RAW データ型と LONG RAW データ型

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

LONG RAW 列をバイナリ LOB（BLOB）へ変換することをお薦めします。LOB 列は、LONG 列ほど制限は多くありません。詳細は、5-206 ページの「TO_LOB」を参照してください。

RAW は、VARCHAR2 と同様に可変長データ型ですが、Oracle Net（ユーザー・セッションとインスタンスを接続します）および Oracle のインポート / エクスポート・ユーティリティは、RAW または LONG RAW データの転送時に文字変換を行いません。これに対し、Oracle Net および Oracle のインポート / エクスポート・ユーティリティは、CHAR、VARCHAR2 および LONG データをデータベース・キャラクタ・セットからユーザー・セッションのキャラクタ・セットに自動的に変換します。2 つのキャラクタ・セットが異なる場合は、ALTER SESSION 文の NLS_LANGUAGE パラメータでユーザー・セッションのキャラクタ・セットを設定できます。

RAW データまたは LONG RAW データと CHAR データ間で、データを自動的に変換するときに、バイナリ・データは 16 進数で表されます。1 つの 16 進文字で 4 ビットの RAW データを表します。たとえば、ビット列が 11001011 で表示される 1 バイトの RAW データは、CB として表示または入力されます。

ラージ・オブジェクト（LOB）・データ型

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

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

LOB 列の作成時に行の記憶域を使用可能にしている場合、CLOB、NCLOB および BLOB の値は、約 4000 バイトを上限としてインラインに格納されます。4000 バイトを超える LOB は、常に外部ファイルに格納されます。詳細は、16-38 ページの「ENABLE STORAGE IN ROW」を参照してください。

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

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

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

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

LOB 列の制限事項： LOB 列には、多数のルールおよび制限事項があります。詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』を参照してください。

参照：

- これらのインタフェースおよび LOB の詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』および『Oracle Call Interface プログラマーズ・ガイド』を参照してください。
- LONG 列から LOB 列への変換については、12-2 ページの「ALTER TABLE」の *modify_col_properties* および 5-206 ページの「TO_LOB」を参照してください。

BFILE データ型

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

BFILENAME ファンクションを使用すると、実表のデータに影響を与えずに BFILE のファイル名およびパスを変更できます。この組込み SQL ファンクションの詳細は、5-21 ページの「BFILENAME」を参照してください。

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

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

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

参照： LOB の詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』および『Oracle Call Interface プログラマーズ・ガイド』を参照してください。また、14-38 ページの「CREATE DIRECTORY」も参照してください。

BLOB データ型

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

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

CLOB データ型

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

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

NCLOB データ型

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

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

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

ROWID データ型

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

ROWID データ型

Oracle Database 固有のヒープ構成表内の行は、**ROWID** という行のアドレスを持ちます。疑似列 ROWID を問い合せることによって、ROWID の行のアドレスを調べることができます。この疑似列の値は、各行のアドレスを表す文字列で、文字列のデータ型は ROWID です。また、ROWID データ型を持つ実際の列を含む表やクラスタを作成することもできます。Oracle Database では、このような列の値が有効な ROWID であることは保証されません。ROWID 疑似列の詳細は、第 3 章「疑似列」を参照してください。

注意： Oracle8 以降、Oracle SQL では、パーティション表、索引、および表領域関連のデータ・ブロック・アドレスを明確かつ効果的にサポートするために、ROWID の拡張形式が採用されています。バージョン 7 のデータベースを使用しており、これをアップグレードする場合は、DBMS_ROWID パッケージを使用してデータ内の ROWID を拡張形式に移行します。DBMS_ROWID については、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。Oracle7 からのアップグレードについては、『Oracle Database アップグレード・ガイド』を参照してください。

ROWID には、次の情報が含まれています。

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

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

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

UROWID データ型

物理アドレスまたは永続アドレス以外のアドレス、または Oracle Database が生成したものではないアドレスがある行を持つ表もあります。たとえば、索引構成表の行のアドレスは索引リーフに格納され、移動できます。外部キー表の ROWID（たとえば、ゲートウェイを介してアクセスされる DB2）は、標準の Oracle ROWID ではありません。

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

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

> **注意：** ヒープ構成表には物理 ROWID があります。データ型が UROWID の列をヒープ構成表に指定しないことをお薦めします。

ANSI、DB2、SQL/DS のデータ型

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

表 2-6 Oracle データ型に変換される ANSI データ型

ANSI SQL データ型	Oracle データ型
CHARACTER(n)	CHAR(n)
CHAR(n)	
CHARACTER VARYING(n)	VARCHAR2(n)
CHAR VARYING(n)	
NATIONAL CHARACTER(n)	NCHAR(n)
NATIONAL CHAR(n)	
NCHAR(n)	
NATIONAL CHARACTER VARYING(n)	NVARCHAR2(n)
NATIONAL CHAR VARYING(n)	
NCHAR VARYING(n)	
NUMERIC[(p,s)]	NUMBER(p,s)
DECIMAL[(p,s)]（**注意 a**）	
INTEGER	NUMBER(38)
INT	
SMALLINT	
FLOAT（**注意 b**）	FLOAT(126)
DOUBLE PRECISION（**注意 c**）	FLOAT(126)
REAL（**注意 d**）	FLOAT(63)

注意 :

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

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

- GRAPHIC
- LONG VARGRAPHIC
- VARGRAPHIC
- TIME

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

参照 : このマニュアルのデータ型の説明を参照してください。

表 2-7 Oracle データ型に変換される SQL/DS と DB2 のデータ型

SQL/DS と DB2 データ型	Oracle データ型
CHARACTER(n)	CHAR(n)
VARCHAR(n)	VARCHAR(n)
LONG VARCHAR	LONG
DECIMAL(p,s) (**a**)	NUMBER(p,s)
INTEGER SMALLINT	NUMBER(38)
FLOAT (**b**)	NUMBER

注意 :

a. DECIMAL データ型では、固定小数点数のみを指定できます。このデータ型では、*s* のデフォルトは 0 です。

b. FLOAT データ型は *b* の 2 進精度を持つ浮動小数点数です。このデータ型のデフォルト精度は 126 桁の 2 進精度（38 桁の 10 進精度）です。

ユーザー定義型

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

参照 :

- Oracle 組込みデータ型の詳細は、『Oracle Database 概要』を参照してください。
- ユーザー定義型の作成方法については、17-3 ページの「CREATE TYPE」および 17-5 ページの「CREATE TYPE BODY」を参照してください。
- ユーザー定義型の使用方法については、『Oracle Database オブジェクト・リレーショナル開発者ガイド』を参照してください。

オブジェクト型

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

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

REF データ型

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

REF 値が、存在しないオブジェクトを指している場合、その REF は DANGLING（参照先がない）状態であるといいます。DANGLING 状態の REF は、NULL である REF と異なります。REF が DANGLING 状態であるかどうかを確認するには、条件 IS [NOT] DANGLING を使用します。たとえば、列型が customer_typ（属性 cust_email を持つ）を指している REF である customer_ref 列があるとします。この customer_ref 列を含むオブジェクト・ビュー oc_orders（サンプル・スキーマ oe 内）は、次のように指定します。

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

VARRAY

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

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

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

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

通常、Oracle は、1 つの配列オブジェクトをインライン形式でその行の他のデータと同じ表領域に、またはアウトライン形式で LOB に格納します。この形式は配列オブジェクトのサイズによって決まります。ただし、VARRAY に個別の記憶特性を指定する場合、Oracle はこれをサイズに関係なくアウトライン形式で格納します。VARRAY の格納方法の詳細は、16-41 ページの「CREATE TABLE」の「varray_col_properties」を参照してください。

ネストした表

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

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

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

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

Oracle が提供する型

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

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

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

- 任意型
- XML 型
- 空間型
- メディア型

任意型

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

ANYTYPE

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

ANYDATA

この型には、指定した型のインスタンスをデータおよび型の記述とともに含めることができます。ANYDATA は、表の列のデータ型として使用できます。また、このデータ型によって、単一列に異機種間の値を格納できます。ユーザー定義型と同様に、SQL 組込み型の値も格納できます。

ANYDATASET

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

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

XML 型

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

XMLType

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

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

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

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

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

URI データ型

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

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

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

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

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

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

```
<HR>
  <EMPLOYEES>
    <ROW>
      <EMPLOYEE_ID>205</EMPLOYEE_ID>
      <LAST_NAME>Higgins</LAST_NAME>
      <SALARY>12000</SALARY>
      .. <!-- other columns -->
    </ROW>
    ... <!-- other rows -->
  </EMPLOYEES>
  <!-- other tables..-->
</HR>
<!-- other user schemas on which you have some privilege on..-->
```

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

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

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

URIFactory パッケージ

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

参照 :

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

空間型

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

SDO_GEOMETRY

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

SDO_GEOMETRY オブジェクト型の定義は、次のとおりです。

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

SDO_TOPO_GEOMETRY

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

SDO_TOPO_GEOMETRY オブジェクト型の定義は、次のとおりです。

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

SDO_GEORASTER

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

SDO_GEORASTER オブジェクト型の定義は、次のとおりです。

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

> **参照：** 空間データ型の完全な実装および使用のガイドラインは、『Oracle Spatial 開発者ガイド』、『Oracle Spatial トポロジおよびネットワーク・データ・モデル開発者ガイド』および『Oracle Spatial GeoRaster 開発者ガイド』を参照してください。

メディア型

Oracle Multimedia は、マルチメディア・データを記述するために、Java または C++ クラスに類似したオブジェクト型を使用します。これらのオブジェクト型のインスタンスは、メタデータおよびメディア・データを含む属性とメソッドで構成されます。Multimedia のデータ型は、ORDSYS スキーマで作成されます。パブリック・シノニムがすべてのデータ型に対して存在するため、スキーマ名を指定せずにアクセスできます。

> **参照：** これらの型の実装および使用のガイドラインは、『Oracle Multimedia リファレンス』を参照してください。

ORDAudio

ORDAudio オブジェクト型は、オーディオ・データの格納および管理をサポートします。

ORDImage

ORDImage オブジェクト型は、イメージ・データの格納および管理をサポートします。

ORDVideo

ORDVideo オブジェクト型は、ビデオ・データの格納および管理をサポートします。

ORDDoc

ORDDoc オブジェクト型は、オーディオ、イメージ、ビデオ・データなどすべての種類のメディア・データの格納および管理をサポートします。この型は、すべてのメディアを単一列に格納する場合に使用します。

ORDDicom

ORDDicom オブジェクト型は、医用画像の規格として広く認知されている形式である Digital Imaging and Communications in Medicine （DICOM）の格納および管理をサポートします。

次のデータ型は、ISO-IEC 13249-5 Still Image 規格（一般に SQL/MM StillImage と呼ばれる）に準拠しています。

SI_StillImage

SI_StillImage オブジェクト型は、高さ、幅、フォーマットなどの固有のイメージ特性を持つデジタル・イメージを表します。

SI_Color

SI_Color オブジェクト型は、色の値をカプセル化します。

SI_AverageColor

SI_AverageColor オブジェクト型は、イメージの平均の色によってそのイメージを特徴付けます。

SI_ColorHistogram

SI_ColorHistogram オブジェクト型は、RAW イメージのサンプルによって表示される色の相対頻度によって、イメージを特徴付けます。

SI_PositionalColor

イメージを n × m 個の四角形に分割すると考えた場合、SI_PositionalColor オブジェクト型は、その四角形の最も重要な n × m 個の色によってイメージを特徴付けます。

SI_Texture

SI_Texture オブジェクト型は、繰り返し出現している項目のサイズ（粗さ）、輝度の変化（コントラスト）および主な方向（方向性）によってイメージを特徴付けます。

SI_FeatureList

SI_FeatureList オブジェクト型は、前述したオブジェクト型（SI_AverageColor、SI_ColorHistogram、SI_PositionalColor および SI_Texture）が表すイメージの特徴のうち最大 4 つの特徴を含むリストです。各特徴は、重み付けされます。

ORDImageSignature

ORDImageSignature は非推奨になったため、コードに使用されることはありません。現存するこのオブジェクト型は、引き続き従来どおり機能します。

Expression Filter 型

Oracle Expression Filter を使用すると、アプリケーション開発者は、ユーザーの目的のデータを表す条件式を管理および評価できます。Expression Filter には、次のデータ型が含まれます。

Expression

Expression Filter は、Expression と呼ばれる仮想データ型を使用して、条件式をデータベース表のデータとして管理および評価します。Expression Filter では、VARCHAR2 列に属性セットを割り当てることで、この列から Expression データ型の列を作成します。この割当てによって、列に格納された式の妥当性を確認するデータ制約が有効になります。

Expression データ型で EVALUATE 演算子を使用して条件を定義すると、列に格納された式をデータに対して評価できます。Enterprise Edition を使用している場合は、Expression データ型の列に Expression Filter 索引を定義して、EVALUATE 演算子によって問合せを処理することもできます。

> **参照：** Expression Filter の詳細は、『Oracle Database ルール・マネージャおよび式フィルタ開発者ガイド』を参照してください。

データ型の比較規則

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

数値

大きい値は小さい値よりも大きいとみなされます。すべての負の数は、0（ゼロ）およびすべての正の数より小さいとみなされます。したがって、-1 は 100 より小さく、-100 は -1 より小さいとみなされます。

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

> **参照：** 比較セマンティクスの詳細は、2-15 ページの「数値の優先順位」および 2-13 ページの「浮動小数点数」を参照してください。

日付値

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

文字値

文字値は、2 つのメジャーに基づいて比較されます。

- バイナリ・ソートまたは言語ソート
- 空白埋め比較セマンティクスまたは非空白埋め比較セマンティクス

次の項で、2 つのメジャーについて説明します。

バイナリ比較または言語比較

デフォルトのバイナリ比較では、Oracle は、データベース・キャラクタ・セット内の文字の数値コードを連結した値に従って文字列を比較します。第 1 の文字の数値が第 2 の文字の数値よりも大きい場合、第 1 の文字は第 2 の文字よりも大きいとみなされます。Oracle は、空白はどの文字よりも小さいとみなします。これは、ほぼすべてのキャラクタ・セットでいえることです。

次に、一般的なキャラクタ・セットを示します。

- 7 ビット ASCII（情報交換用米国標準コード）
- EBCDIC コード（拡張 2 進化 10 進コード）
- ISO 8859/1（国際標準化機構）
- JEUC 日本語拡張 UNIX

数値コードのバイナリ順序と、比較する文字の言語順序が一致していない場合は、言語比較が有効です。NLS_SORT パラメータに BINARY 以外の設定があり、NLS_COMP パラメータが LINGUISTIC に設定されている場合は言語比較が使用されます。言語ソートでは、すべての SQL のソートおよび比較が NLS_SORT によって指定された言語規則に基づいて行われます。

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

空白埋め比較セマンティクスおよび非空白埋め比較セマンティクス

空白埋め比較セマンティクスでは、2 つの値の長さが異なる場合、Oracle はまず短い方の値の最後に空白を追加して、2 つの値が同じ長さになるようにします。次に、その 2 つの値を、最初に異なる文字まで 1 文字ずつ比較します。最初に異なる文字の位置で、大きい方の文字を持つ値の方が大きいとみなされます。2 つの値に異なる文字がない場合、その 2 つの値は等しいとみなされます。この規則では、2 つの値の後続空白数のみが異なる場合、その 2 つの値は等しいとみなされます。Oracle では、比較する両方の値が、CHAR データ型、NCHAR データ型、テキスト・リテラルのいずれかの式の場合、または USER ファンクションの戻り値の場合のみ空白埋め比較セマンティクスを使用します。

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

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

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

ASCII と EBCDIC のキャラクタ・セットの一部を表 2-8 と表 2-9 に示します。大文字と小文字は同じではありません。キャラクタ・セットの照合順番は、特定の言語の言語順序と一致しない場合があります。

表 2-8 ASCII キャラクタ・セット

記号	10 進値	記号	10 進値
空白	32	;	59
!	33	<	60
"	34	=	61
#	35	>	62
$	36	?	63
%	37	@	64
&	38	A-Z	65-90
'	39	[	91
(	40	\	92
)	41	]	93
*	42	^	94
+	43	_	95
,	44	'	96
-	45	a-z	97-122
.	46	{	123
/	47	|	124
0-9	48-57	}	125
:	58	~	126

表 2-9 EBCDIC キャラクタ・セット

記号	10 進値	記号	10 進値
空白	64	%	108
¢	74	_	109
.	75	>	110
<	76	?	111
(	77	:	122
+	78	#	123
|	79	@	124
&	80	'	125
!	90	=	126
$	91	"	127
*	92	a-i	129-137
)	93	j-r	145-153
;	94	s-z	162-169
?	95	A-I	193-201
-	96	J-R	209-217
/	97	S-Z	226-233

オブジェクト値

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

> **参照：** MAP メソッドと ORDER メソッド、およびこれらのメソッドが戻す値の詳細は、17-3 ページの「CREATE TYPE」を参照してください。

VARRAY とネストした表

ネストした表の比較の詳細は、7-4 ページの「比較条件」を参照してください。

データ型の優先順位

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

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

データ変換

一般に、式には異なるデータ型の値を含めることができません。たとえば、式では 10 に 5 を掛けた値に 'JAMES' を加えることはできません。ただし、Oracle では値をあるデータ型から別のデータ型へ変換する場合の、暗黙的な変換と明示的な変換をサポートしています。

暗黙的なデータ変換と明示的なデータ変換

次の理由から、暗黙的な変換または自動変換ではなく、明示的な変換を指定することをお薦めします。

- 明示的なデータ型変換ファンクションを使用すると、SQL 文がわかりやすくなります。
- 暗黙的なデータ型変換（特に列値のデータ型が定数に変換される場合）は、パフォーマンスに悪影響を及ぼす可能性があります。
- 暗黙的な変換はその変換が行われるコンテキストに依存し、どんな場合でも同様に機能するとはかぎりません。たとえば、日時値から VARCHAR2 型へ値を暗黙的に変換すると、NLS_DATE_FORMAT パラメータに指定されている値によっては、予期しない年が戻される場合があります。
- 暗黙的な変換のアルゴリズムは、ソフトウェア・リリースや Oracle 製品の変更によって変更されることがあります。明示的な変換を指定しておくと、その動作は将来的にも確実になります。

暗黙的なデータ変換

あるデータ型から別のデータ型への変換が意味を持つ場合、Oracle Database は値を自動的に変換します。文字データ型への暗黙的な変換はこれらのルールに従います。

表 2-10 に、Oracle の暗黙的な変換のマトリックスについて示します。この表は、変換の方向または変換されるコンテキストにかかわらず、すべての可能な変換を示します。詳細は、表の後の説明を参照してください。

表 2-10 暗黙的な型変換のマトリックス

	CHAR	VARCHAR2	NCHAR	NVARCHAR2	DATE	DATETIME/ INTERVAL	NUMBER	BINARY_FLOAT	BINARY_DOUBLE	LONG	RAW	ROWID	CLOB	BLOB	NCLOB
CHAR	--	X	X	X	X	X	X	X	X	X	X	--	X	X	X
VARCHAR2	X	--	X	X	X	X	X	X	X	X	X	X	X	--	X
NCHAR	X	X	--	X	X	X	X	X	X	X	X	X	X	--	X
NVARCHAR2	X	X	X	--	X	X	X	X	X	X	X	X	X	--	X
DATE	X	X	X	X	--	--	--	--	--	--	--	--	--	--	--
DATETIME/ INTERVAL	X	X	X	X	--	--	--	--	--	X	--	--	--	--	--
NUMBER	X	X	X	X	--	--	--	X	X	--	--	--	--	--	--
BINARY_FLOAT	X	X	X	X	--	--	X	--	X	--	--	--	--	--	--
BINARY_ DOUBLE	X	X	X	X	--	--	X	X	--	--	--	--	--	--	--
LONG	X	X	X	X	--	X[1]	--	--	--	--	X	--	X	--	X
RAW	X	X	X	X	--	--	--	--	--	X	--	--	--	X	--
ROWID	--	X	X	X	--	--	--	--	--	--	--	--	--	--	--
CLOB	X	X	X	X	--	--	--	--	--	X	--	--	--	--	X
BLOB	--	--	--	--	--	--	--	--	--	--	X	--	--	--	--
NCLOB	X	X	X	X	--	--	--	--	--	X	--	--	X	--	--

注意 1: LONG を INTERVAL に直接変換することはできませんが、TO_CHAR(*interval*) を使用して LONG を VARCHAR2 に変換し、その VARCHAR2 の値を INTERVAL に変換できます。

次に示す規則に従って、Oracle Database は、暗黙的なデータ型変換を実行する方向を確立します。

- INSERT および UPDATE 操作中に、Oracle は変更する列のデータ型に値を変換します。
- SELECT FROM 操作中に、Oracle は列からターゲット変数の型にデータを変換します。
- 数値を操作する際、Oracle は、通常、最大容量を確保するために精度および位取りを調整します。この場合、このような操作によって変換された数値データ型は、基礎となる表に含まれる数値データ型と異なることがあります。
- 数値と文字値を比較する場合、Oracle は文字データを数値に変換します。
- 文字値または NUMBER の値と浮動小数点数の値の間では、変換が正確に行われない場合があります。これは、文字型および NUMBER では 10 進精度で数値が示されるのに対し、浮動小数点数では 2 進精度が使用されているためです。
- CLOB 値を VARCHAR2 などの文字データ型に変換する場合、または BLOB を RAW データに変換する場合、変換するデータがターゲットのデータ型より大きいと、データベースはエラーを戻します。
- BINARY_FLOAT から BINARY_DOUBLE への変換は正確に行われます。
- BINARY_DOUBLE の値が BINARY_FLOAT でサポートされている精度のビット数よりも多いビット数を使用している場合、BINARY_DOUBLE から BINARY_FLOAT への変換は正確に行われません。
- DATE の値と文字の値を比較する場合、Oracle は文字データを DATE に変換します。

- SQL ファンクションまたは演算子に不当なデータ型の引数を指定して使用する場合、Oracle はその引数を正当なデータ型に変換します。
- 代入を実行する場合、Oracle は等号（=）の右側の値を左側の代入ターゲットのデータ型に変換します。
- 連結中に、Oracle は非文字データ型を CHAR または NCHAR に変換します。
- 文字データ型と非文字データ型に対する演算処理および比較中に、Oracle はすべての文字データ型を数値、日付または ROWID のいずれかの適切なデータ型に変換します。CHAR/VARCHAR2 と NCHAR/NVARCHAR2 の演算処理では、Oracle は NUMBER に変換します。
- CHAR と VARCHAR2、NCHAR と NVARCHAR2 の比較では、異なるキャラクタ・セットが必要な場合があります。このような場合のデフォルトの変換の方向は、データベース・キャラクタ・セットから各国語キャラクタ・セットです。表 2-11 に、異なるキャラクタ・タイプ間での暗黙的な変換の方向を示します。
- ほとんどの SQL 文字ファンクションは、CLOB をパラメータとして指定できます。また、Oracle は CLOB と文字型間で暗黙的な変換を実行します。このため、CLOB を使用できないファンクションは、暗黙的な変換を使用して CLOB を受け入れます。このような場合、Oracle はファンクションが起動される前に CLOB を CHAR または VARCHAR2 に変換します。CLOB が 4000 バイトより大きい場合、Oracle は最初の 4000 バイトのみを CHAR に変換します。

表 2-11 異なるキャラクタ・タイプの変換方向

	CHAR へ	VARCHAR2 へ	NCHAR へ	NVARCHAR2 へ
CHAR から	--	VARCHAR2	NCHAR	NVARCHAR2
VARCHAR2 から	VARCHAR2	--	NVARCHAR2	NVARCHAR2
NCHAR から	NCHAR	NCHAR	--	NVARCHAR2
NVARCHAR2 から	NVARCHAR2	NVARCHAR2	NVARCHAR2	--

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

暗黙的なデータ変換の例

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

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

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

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

日付の例　次の文では、Oracle がデフォルトの日付書式 'DD-MON-YY' を使用して、'03-MAR-97' を DATE 値に暗黙的に変換します。

```
SELECT last_name
    FROM employees
    WHERE hire_date = '03-MAR-97';
```

明示的なデータ変換

SQL 変換ファンクションを使用すると、データ型の変換を明示的に指定できます。表 2-12 に、値をあるデータ型から別のデータ型に明示的に変換する SQL ファンクションを示します。

Oracle が暗黙的なデータ型変換を行うことができる場合には、LONG および LONG RAW の値を指定できません。たとえば、ファンクションや演算子を含む式では、LONG と LONG RAW の値を使用できません。LONG データ型および LONG RAW データ型の制限については、2-15 ページの「LONG データ型」を参照してください。

表 2-12 明示的な型の変換

	CHAR、VARCHAR2、NCHAR、NVARCHAR2	NUMBER	Datetime/ Interval	RAW	ROWID	LONG、LONG RAW	CLOB、NCLOB、BLOB	BINARY_FLOAT	BINARY_DOUBLE
CHAR、VARCHAR2、NCHAR、NVARCHAR2 から	TO_CHAR（文字） TO_NCHAR（文字）	TO_NUMBER	TO_DATE TO_TIMESTAMP TO_TIMESTAMP_TZ TO_YMINTERVAL TO_DSINTERVAL	HEXTORAW	CHARTOROWID	--	TO_CLOB TO_NCLOB	TO_BINARY_FLOAT	TO_BINARY_DOUBLE
NUMBER から	TO_CHAR（数値） TO_NCHAR（数値）	--	TO_DATE NUMTOYMINTERVAL NUMTODSINTERVAL	--	--	--	--	TO_BINARY_FLOAT	TO_BINARY_DOUBLE
Datetime Interval から	TO_CHAR（日付） TO_NCHAR（日時）	--	--	--	--	--	--	--	--
RAW から	RAWTOHEX RAWTONHEX	--	--	--	--	--	TO_BLOB	--	--
ROWID から	ROWIDTOCHAR	--	--	--	--	--	--	--	--
LONG/ LONG RAW から	--	--	--	--	--	--	TO_LOB	--	--
CLOB、NCLOB、BLOB から	TO_CHAR TO_NCHAR	--	--	--	--	--	TO_CLOB TO_NCLOB	--	--

表 2-12 明示的な型の変換（続き）

	CHAR、VARCHAR2、NCHAR、NVARCHAR2	NUMBER	Datetime/ Interval	RAW	ROWID	LONG、LONG RAW	CLOB、NCLOB、BLOB	BINARY_FLOAT	BINARY_DOUBLE
CLOB、NCLOB、BLOB から	TO_CHAR TO_NCHAR	--	--	--	--	--	TO_CLOB TO_NCLOB	--	--
BINARY_FLOAT から	TO_CHAR（文字） TO_NCHAR（文字）	TO_NUMBER	--	--	--	--	--	TO_BINARY_FLOAT	TO_BINARY_DOUBLE
BINARY_DOUBLE から	TO_CHAR（文字） TO_NCHAR（文字）	TO_NUMBER	--	--	--	--	--	TO_BINARY_FLOAT	TO_BINARY_DOUBLE

参照： すべての明示的な変換ファンクションの詳細は、5-5 ページの「変換ファンクション」を参照してください。

データ変換のセキュリティ上の考慮事項

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

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

```
EXECUTE IMMEDIATE
'SELECT name FROM employee WHERE hiredate > ''' || start_date || '''';
```

start_date のデータ型は DATE です。

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

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

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

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

参照：

- セッションのグローバリゼーション・パラメータの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。
- 書式モデルの詳細は、2-54ページの「書式モデル」を参照してください。

リテラル

リテラルと**定数値**という用語の意味は同じで、固定データ値のことです。たとえば、'JACK'、'BLUE ISLAND'および'101'はすべて文字リテラルです。文字リテラルは、一重引用符で囲みます。一重引用符を付けることで、Oracleは文字リテラルとスキーマ・オブジェクト名を区別します。

この項では、次の内容を説明します。

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

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

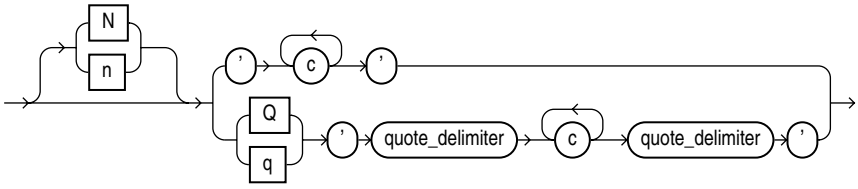
Datetime（日時）またはInterval（期間）のデータ型をリテラルとして指定する場合は、データ型に含まれているオプションの精度を考慮する必要があります。Datetime（日時）およびInterval（期間）のデータ型をリテラルとして指定する場合の例は、2-2ページの「データ型」の関連する項を参照してください。

テキスト・リテラル

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

テキスト・リテラルまたは文字列の構文は次のとおりです。

***string*::=**



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

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

構文の上の方のブランチでは、次のようになります。

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

構文の下の方のブランチでは、次のようになります。

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

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

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

- 式と条件の中のテキスト・リテラルは、Oracle によって CHAR データ型として扱われ、空白埋め比較セマンティクスで比較されます。
- テキスト・リテラルの最大長は 4000 バイトです。

有効なテキスト・リテラルの例を次に示します。

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

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

```
q'!name LIKE '%DBMS_%%'!'
q'<'So,' she said, 'It's finished.'>'
q'{SELECT * FROM employees WHERE last_name = 'Smith';}'
nq'Ô ü1234 Ô'
q'"name like '['"'
```

> **参照：** 「空白埋め比較セマンティクスおよび非空白埋め比較セマンティクス」（2-37 ページ）

数値リテラル

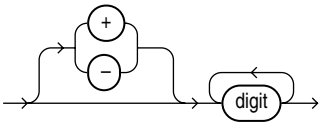
固定小数点数および浮動小数点数を指定するには、数値リテラルの表記法を使用します。

整数リテラル

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

integer（整数）の構文は次のとおりです。

***integer*::=**



digit は、0、1、2、3、4、5、6、7、8、9 のいずれかです。

整数は最大 38 桁の精度を記憶できます。

有効な integer（整数）の例を次に示します。

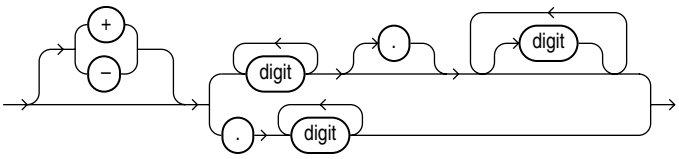
```
7
+255
```

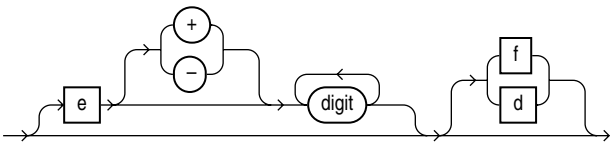
NUMBER および浮動小数点リテラル

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

number（数）の構文は次のとおりです。

***number*::=**





それぞれの意味は、次のとおりです。

- + または - は、正の値または負の値を示します。符号を指定しない場合、デフォルトは正の値です。
- *digit* は、0、1、2、3、4、5、6、7、8、9 のいずれかです。
- e または E は、数が科学表記法で指定されることを示します。E の後の数字が指数を示します。指数は -130 ～ 125 の範囲で指定します。
- f または F は、数が BINARY_FLOAT 型の 32 ビットの 2 進浮動小数点数であることを示します。
- d または D は、数が BINARY_DOUBLE 型の 64 ビットの 2 進浮動小数点数であることを示します。

  f（F）および d（D）を省略すると、数は NUMBER 型になります。

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

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

初期化パラメータ NLS_NUMERIC_CHARACTERS を使用してピリオド（.）以外の小数点文字を設定している場合は、*'text'* の表記法で数値リテラルを指定する必要があります。この場合、Oracle は自動的にテキスト・リテラルを数値に変換します。

注意： この表記法は、浮動小数点数リテラルでは使用できません。

たとえば、NLS_NUMERIC_CHARACTERS パラメータでカンマを小数点文字に設定している場合、数値 5.123 は次のように指定します。

```
'5,123'
```

参照： 11-41 ページの「ALTER SESSION」および『Oracle Database リファレンス』を参照してください。

有効な NUMBER リテラルの例を次に示します。

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

有効な浮動小数点数リテラルの例を次に示します。

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

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

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

日時リテラル

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

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

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

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

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

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

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

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

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

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

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

```
INSERT INTO my_table VALUES (1, SYSDATE);
INSERT INTO my_table VALUES (2, TRUNC(SYSDATE));

SELECT * FROM my_table;
```

```
   ROW_NUM DATECOL
---------- ---------
         1 03-OCT-02
         2 03-OCT-02

SELECT * FROM my_table
   WHERE datecol = TO_DATE('03-OCT-02','DD-MON-YY');

   ROW_NUM DATECOL
---------- ---------
         2 03-OCT-02

SELECT * FROM my_table
   WHERE datecol > TO_DATE('02-OCT-02', 'DD-MON-YY');

   ROW_NUM DATECOL
---------- ---------
         1 03-OCT-02
         2 03-OCT-02
```

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

```
SELECT * FROM my_table WHERE datecol = DATE '2002-10-03';
```

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

```
SELECT * FROM my_table WHERE TRUNC(datecol) = DATE '2002-10-03';
```

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

- TO_DATE ファンクションを使用して、時刻フィールドをマスクします。

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

- DATE リテラルを使用します。

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

- TRUNC ファンクションを使用します。

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

日付ファンクション SYSDATE は、現在のシステムの日付および時刻を戻します。CURRENT_DATE ファンクションは、現在のセッションの日付を戻します。SYSDATE、TO_* 日時ファンクションおよびデフォルト日付書式の詳細は、5-5 ページの「日時ファンクション」を参照してください。

TIMESTAMP リテラル TIMESTAMP データ型は、年、月、日、時、分、秒、および秒の小数部の値を格納します。TIMESTAMP をリテラルに指定する場合、*fractional_seconds_precision* 値には最大 9 桁を指定できます。次に例を示します。

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

TIMESTAMP WITH TIME ZONE リテラル TIMESTAMP WITH TIME ZONE データ型は、タイムゾーン地域名またはタイムゾーン・オフセットを含む TIMESTAMP の変形です。TIMESTAMP WITH TIME ZONE をリテラルに指定する場合、*fractional_seconds_precision* 値には最大 9 桁を指定できます。次に例を示します。

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

2 つの TIMESTAMP WITH TIME ZONE 値が UTC で同じ時刻を表す場合は、データに格納された TIME ZONE オフセットにかかわらず、同一であるとみなされます。次に例を示します。

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

前述の例文は次の例文と同じです。

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

太平洋標準時の午前 8 時は、東部標準時の午前 11 時と同じです。

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

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

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

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

タイムゾーン・オフセットは日時式を使用して表記することもできます。

```
SELECT TIMESTAMP '1999-10-29 01:30:00' AT TIME ZONE 'US/Pacific' FROM DUAL;
```

参照： 詳細は、6-8 ページの「日時式」を参照してください。

ERROR_ON_OVERLAP_TIME セッション・パラメータを TRUE に設定しておくと、TZD 書式要素を追加しなかったために日時値があいまいな場合、Oracle はエラーを戻します。パラメータを FALSE に設定しておくと、Oracle はあいまいな日時を、指定した地域の標準時刻として解析します。

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

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

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

期間リテラル

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

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

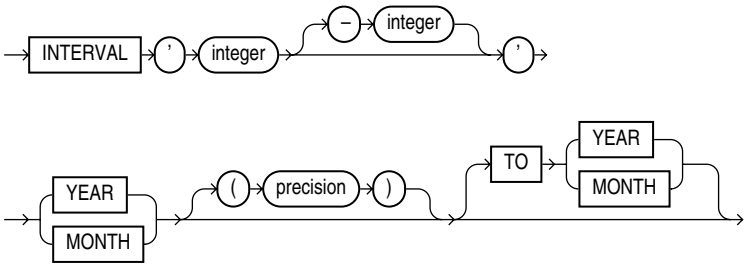
期間リテラルは、主に分析ファンクションとともに使用します。

> **参照：** 5-10 ページの「分析ファンクション」、5-113 ページの「NUMTODSINTERVAL」、5-114 ページの「NUMTOYMINTERVAL」および『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

INTERVAL YEAR TO MONTH

次の構文を使用して、YEAR TO MONTH 期間リテラルを指定します。

***interval_year_to_month*::=**



それぞれの意味は、次のとおりです。

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

先行フィールドの制限事項： 後続フィールドを指定する場合は、その桁数を先行フィールドの桁数より少なく設定する必要があります。たとえば、INTERVAL '0-1' MONTH TO YEAR は無効です。

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

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

このリテラルの他の書式の例を次に示します。省略バージョンも含みます。

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

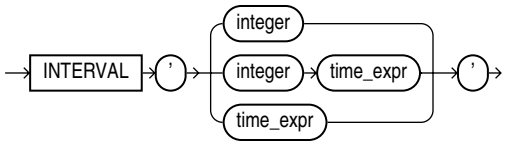
ある INTERVAL YEAR TO MONTH リテラルを別のリテラルに加算または減算して、新しい INTERVAL YEAR TO MONTH リテラルを作成できます。次に例を示します。

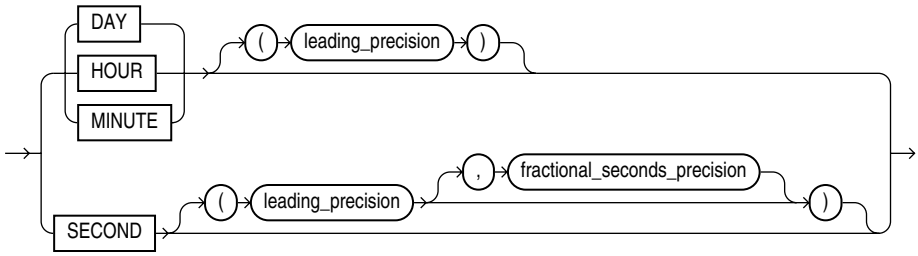
```
INTERVAL '5-3' YEAR TO MONTH + INTERVAL'20' MONTH =
INTERVAL '6-11' YEAR TO MONTH
```

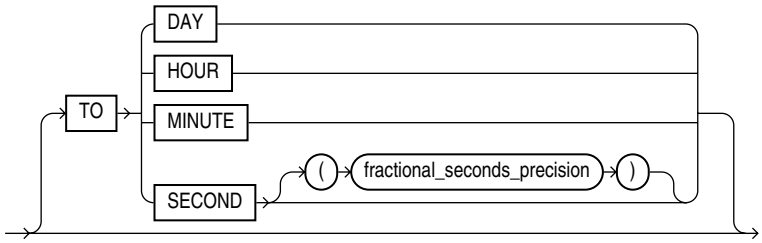
INTERVAL DAY TO SECOND

次の構文を使用して、DAY TO SECOND 期間リテラルを指定します。

***interval_day_to_second*::=**







それぞれの意味は、次のとおりです。

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

先行フィールドの制限事項： 後続フィールドを指定する場合は、その桁数を先行フィールドの桁数より少なく設定する必要があります。たとえば、INTERVAL MINUTE TO DAY は無効です。このため、SECOND が先行フィールドの場合、期間リテラルは後続フィールドを持つことができません。

後続フィールドの有効範囲は次のとおりです。

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

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

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

DAY TO SECOND 期間リテラルを別の DAY TO SECOND 期間リテラルに加算または減算できます。次に例を示します。

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

書式モデル

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

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

次に例を示します。

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

日時および数値書式モデルの要素のリストは、2-55 ページの表 2-13「数値書式要素」および 2-58 ページの表 2-15「日時書式要素」を参照してください。

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

参照：

- これらのパラメータの値の変更については、11-41 ページの「ALTER SESSION」を参照してください。書式モデルの使用例は、2-64 ページの「書式モデルの例」を参照してください。
- 5-200 ページの「TO_CHAR（日時）」、5-201 ページの「TO_CHAR（数値）」および 5-203 ページの「TO_DATE」を参照してください。
- これらのパラメータの詳細は、『Oracle Database リファレンス』および『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

この項の後半では、次の書式モデルの使用方法について説明します。

- 数値書式モデル
- 日時書式モデル
- 書式モデルの修飾子

数値書式モデル

次のファンクションで数値書式モデルを使用できます。

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

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

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

数値書式要素

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

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

表 2-13 数値書式要素

要素	例	説明
,（カンマ）	9,999	指定した位置にカンマを戻します。1 つの数値書式モデルに複数のカンマを指定できます。 **制限事項：** ■ 数値書式モデルは、カンマ要素で始めることはできません。 ■ 数値書式モデルでは、カンマを小数点文字やピリオドの右側に指定できません。
.（ピリオド）	99.99	指定した位置に小数点（ピリオド（.））を戻します。 **制限事項：**1 つの数値書式モデルには 1 つのピリオドのみ指定できます。
$	$9999	値の前にドル記号を付けて戻します。
0	0999	先行 0（ゼロ）を戻します。
	9990	後続 0（ゼロ）を戻します。
9	9999	正の値の場合は先頭に空白を埋め込み、負の値の場合は先頭に負の符号を埋め込んで、指定の桁数にしてから値を戻します。固定小数点数の整数部分の場合、先行 0（ゼロ）に対して空白を戻します。ただし、値 0（ゼロ）に対しては 0（ゼロ）を戻します。
B	B9999	整数部が 0（ゼロ）の場合、書式モデル内の 0（ゼロ）にかかわらず、固定小数点数の整数部に対して空白を戻します。
C	C999	指定した位置に ISO 通貨記号（NLS_ISO_CURRENCY パラメータの現在の値）を戻します。
D	99D99	指定した位置に小数点文字（NLS_NUMERIC_CHARACTER パラメータの現在の値）を戻します。デフォルトはピリオド（.）です。 **制限事項：**1 つの数値書式モデルには 1 つの小数点文字のみ指定できます。
EEEE	9.9EEEE	科学表記法で値を戻します。
G	9G999	指定した位置に桁区切り（NLS_NUMERIC_CHARACTER パラメータの現在の値）を戻します。1 つの数値書式モデルに複数の桁区切りを指定できます。 **制限事項：**数値書式モデルにおいて、桁区切りは小数点文字やピリオドの右側に指定できません。

表 2-13 数値書式要素（続き）

要素	例	説明
L	L999	指定した位置にローカル通貨記号（NLS_CURRENCY パラメータの現在の値）を戻します。
MI	9999MI	負の値の後に負の符号（-）を戻します。 正の値の後に空白を戻します。 **制限事項：**書式要素 MI は、数値書式モデルの最後の位置にのみ指定できます。
PR	9999PR	山カッコ <> の中に負の値を戻します。 正の値の前後に空白を付けて戻します。 **制限事項：**書式要素 PR は、数値書式モデルの最後の位置にのみ指定できます。
RN	RN	大文字のローマ数字で値を戻します。
rn	rn	小文字のローマ数字で値を戻します。 値は 1 ～ 3999 の整数となります。
S	S9999	負の値の前に負の符号（-）を戻します。 正の値の前に正の符号（+）を戻します。
	9999S	負の値の後に負の符号（-）を戻します。 正の値の後に正の符号（+）を戻します。 **制限事項：**書式要素 S は、数値書式モデルの最初または最後の位置にのみ指定できます。
TM	TM	テキストの最小値です。できるだけ少ない文字数を 10 進数で戻します。この要素は、大文字と小文字を区別しません。 デフォルトは TM9 です。デフォルトでは、戻り値が 64 文字を超えないかぎり、固定表記法で文字数を戻します。戻り値が 64 文字を超える場合、Oracle Database は自動的に科学表記法で文字数を戻します。 **制限事項：** ■ この要素の前に他の要素を指定することはできません。 ■ この要素の後には、1 つの 9、E または e のみを指定できます。ただし、これらを組み合せて指定することはできません。次の文は、エラーを戻します。 `SELECT TO_CHAR(1234, 'TM9e') FROM DUAL;`
U	U9999	指定した位置にユーロ（または他の）第 2 通貨記号（NLS_DUAL_CURRENCY パラメータの現在の値によって決定される）を戻します。
V	999V99	値を 10 の n 乗にして戻します（必要に応じて数値を丸めます）。ここで、n は V の後の 9 の数です。
X	XXXX xxxx	指定の桁数の 16 進数値を戻します。指定した値が整数でない場合、Oracle Database はその値を整数に丸めます。 **制限事項：** ■ この要素は、正の値または 0（ゼロ）のみを受け入れます。負の値の場合、エラーを戻します。 ■ この要素の前には、0（先行 0（ゼロ）を戻す）または FM のみを指定できます。その他の要素の場合、エラーを戻します。0（ゼロ）または FM を X と同時に指定しないと、戻り値の前に常に空白が 1 つ追加されます。

表 2-14 に、異なる *number* と *'fmt'* に対して次の問合せを行った場合の結果を示します。

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

表 2-14 数値変換の結果

number	'fmt'	結果
-1234567890	9999999999S	'1234567890-'
0	99.99	' .00'
+0.1	99.99	' .10'
-0.2	99.99	' -.20'
0	90.99	' 0.00'
+0.1	90.99	' 0.10'
-0.2	90.99	'-0.20'
0	9999	' 0'
1	9999	' 1'
0	B9999	' '
1	B9999	' 1'
0	B90.99	' '
+123.456	999.999	'123.456'
-123.456	999.999	'-123.456'
+123.456	FM999.009	'123.456'
+123.456	9.9EEEE	'1.2E+02'
+1E+123	9.9EEEE	'1.0E+123'
+123.456	FM9.9EEEE	'1.2E+02'
+123.45	FM999.009	'123.45'
+123.0	FM999.009	'123.00'
+123.45	L999.99	' $123.45'
+123.45	FML999.99	'$123.45'
+1234567890	9999999999S	'1234567890+'

日時書式モデル

次のファンクションで日時書式モデルを使用できます。

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

日時書式モデルの合計長は最大 22 文字です。

デフォルト日時書式は、NLS セッション・パラメータ NLS_DATE_FORMAT、NLS_TIMESTAMP_FORMAT および NLS_TIMESTAMP_TZ_FORMAT で明示的に指定することも、NLS セッション・パラメータ NLS_TERRITORY で暗黙的に指定することもできます。デフォルト日時書式をセッションごとに変更するには、ALTER SESSION 文を使用します。

参照： NLS パラメータの詳細は、11-41 ページの「ALTER SESSION」および『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

日時書式要素

日時書式モデルは、2-58 ページの表 2-15「日時書式要素」に示す、1 つ以上の日時書式要素で構成されます。

- 入力書式モデルの場合、同じ書式項目は 2 回指定できません。また、類似した情報を表す書式項目を組み合せることもできません。たとえば、'SYYYY' と 'BC' を同一の書式文字列内で使用することはできません。
- 2 番目の列は、書式要素が TO_* 日付ファンクションで使用可能かどうかを示します。TO_CHAR では、すべての書式要素が使用できます。
- 日時書式要素 FF、TZD、TZH、TZM および TZR は、タイムスタンプおよび期間書式モデルでは使用できますが、元の DATE 書式モデルでは使用できません。
- 多くの場合、日時書式要素は一定の長さになるまで空白または先行 0（ゼロ）が埋め込まれます。詳細は、2-63 ページの「FM」を参照してください。

日付書式要素における大文字 戻される日付値では、その大文字と小文字は対応する書式要素の表記に従います。たとえば、日付書式モデル「DAY」は「MONDAY」、「Day」は「Monday」、「day」は「monday」を生成します。

日時書式モデルにおける句読点と文字リテラル 日付書式モデルでは、次の文字を指定できます。

- ハイフン、スラッシュ、カンマ、ピリオド、コロンなどの句読点
- 文字リテラル（二重引用符で囲みます）

これらの文字は、戻り値の中で書式モデルに指定された位置と同じ位置に現れます。

表 2-15 日時書式要素

要素	TO_* 日時ファンクションで指定できるか	説明
- / , . ; : "text"	はい	結果に取り込まれる句読点とテキスト。
AD A.D.	はい	ピリオド付き / なしで西暦を示します。
AM A.M.	はい	ピリオド付き / なしで午前を示します。
BC B.C.	はい	ピリオド付き / なしで紀元前を示します。
CC SCC		世紀。 ■ 4 桁で表した年の下 2 桁が 01 ～ 99（01 および 99 を含む）の場合、世紀はその年の上 2 桁より 1 つ大きくなります。 ■ 4 桁で表した年の下 2 桁が 00 の場合、世紀はその年の上 2 桁と同一になります。 たとえば、2002 は 21 を戻し、2000 は 20 を戻します。

表 2-15 日時書式要素（続き）

要素	TO_* 日時ファンクションで指定できるか	説明
D	はい	曜日（1 ～ 7）。
DAY	はい	曜日。
DD	はい	月における日（1 ～ 31）。
DDD	はい	年における日（1 ～ 366）。
DL	はい	Oracle Database の DATE 書式の拡張である長い日付書式の値（NLS_DATE_FORMAT パラメータの現在の値によって決定される）を戻します。日付コンポーネント（曜日、月番号など）は、NLS_TERRITORY および NLS_LANGUAGE パラメータに応じて表示されます。たとえば、AMERICAN_AMERICA ロケールでは、これは書式 'fmDay, Month dd, yyyy' を指定することと同じです。GERMAN_GERMANY ロケールでは、書式 'fmDay, dd. Month yyyy' を指定することと同じです。 **制限事項 :** この書式は TS 要素とのみ指定できます。その場合は空白で区切ります。
DS	はい	短い日付書式の値を戻します。日付コンポーネント（曜日、月番号など）は、NLS_TERRITORY および NLS_LANGUAGE パラメータに応じて表示されます。たとえば、AMERICAN_AMERICA ロケールでは、これは書式 'MM/DD/RRRR' を指定することと同じです。ENGLISH_UNITED_KINGDOM ロケールでは、書式 'DD/MM/RRRR' を指定することと同じです。 **制限事項 :** この書式は TS 要素とのみ指定できます。その場合は空白で区切ります。
DY	はい	曜日の省略形。
E	はい	時代名の略称（日本、台湾、タイ）。
EE	はい	時代名の完全名称（日本、台湾、タイ）。
FF [1..9]	はい	小数部。基数は出力されません。基数文字の追加には、X 書式要素を使用します。戻される日時の値の小数部の桁数を指定するには、FF の後に、1 ～ 9 の数字を使用します。数字を指定しない場合は、日時データ型に指定された精度またはデータ型のデフォルトの精度が使用されます。タイムスタンプ書式および期間書式では有効ですが、DATE 書式では有効ではありません。 **例 :** 'HH:MI:SS.FF' SELECT TO_CHAR(SYSTIMESTAMP, 'SS.FF3') from dual;
FM	はい	前後に空白を付けずに値を戻します。 **参照** : この書式モデル修飾子の詳細は、このマニュアルを参照してください。
FX	はい	文字データと書式モデルが完全に一致する必要があります。 **参照** : この書式モデル修飾子の詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。
HH HH12	はい	時間（1 ～ 12）。
HH24	はい	時間（0 ～ 23）。
IW		ISO 規格に基づく、年における週（1 ～ 52 または 1 ～ 53）。
IYY IY I		それぞれ ISO 年の下 3 桁、2 桁、1 桁。
IYYY		ISO 規格に基づく 4 桁の年。
J	はい	ユリウス日。紀元前 4712 年 1 月 1 日から経過した日数。「J」を付けて指定する数値は、整数にしてください。

表 2-15 日時書式要素（続き）

要素	TO_* 日時ファンクションで指定できるか	説明
MI	はい	分（0 ～ 59）。
MM	はい	月（01~12、1 月 = 01）。
MON	はい	月の名前の省略形。
MONTH	はい	月の名前。
PM P.M.	はい	ピリオド付き / なしで午後を示します。
Q		年の四半期（1、2、3、4。1 月～ 3 月 =1）。
RM	はい	ローマ数字で表した月（I ～ XII。1 月 =I）。
RR	はい	2 桁のみを使用して、21 世紀に 20 世紀の日付を格納できます。 **参照** : RR 日時書式要素の詳細は、『Oracle Database SQL 言語リファレンス』を参照してください。
RRRR	はい	年を丸めます。4 桁または 2 桁で入力できます。2 桁の場合、RR の場合と同様の結果が戻ります。年を 4 桁で入力すると、この処理は行われません。
SS	はい	秒（0 ～ 59）。
SSSSS	はい	午前 0 時から経過した秒（0 ～ 86399）。
TS	はい	短い時刻書式の値を戻します。時刻コンポーネント（時、分など）は、NLS_TERRITORY および NLS_LANGUAGE 初期化パラメータに応じて表示されます。 **制限事項 :** この書式は DL 要素または DS 要素とのみ指定できます。その場合は空白で区切ります。
TZD	はい	夏時間の情報。TZD の値は、夏時間の情報を持つタイムゾーン文字列の省略形です。TZR で指定した地域と対応している必要があります。タイムスタンプ書式および期間書式では有効ですが、DATE 書式では有効ではありません。 **例 :** PST（米国 / 太平洋標準時）、PDT（米国 / 太平洋夏時間）
TZH	はい	タイムゾーンの時間（TZM 書式要素を参照）。タイムスタンプ書式および期間書式では有効ですが、DATE 書式では有効ではありません。 **例 :** 'HH:MI:SS.FFTZH:TZM'
TZM	はい	タイムゾーンの分（TZH 書式要素を参照）。タイムスタンプ書式および期間書式では有効ですが、DATE 書式では有効ではありません。 **例 :** 'HH:MI:SS.FFTZH:TZM'
TZR	はい	タイムゾーン地域の情報。値は、データベースでサポートされるタイムゾーン地域である必要があります。タイムスタンプ書式および期間書式では有効ですが、DATE 書式では有効ではありません。 **例 :** US/Pacific
WW		年における週（1 ～ 53）。第 1 週はその年の 1 月 1 日で始まり、1 月 7 日で終了します。
W		月における週（1 ～ 5）。第 1 週はその月の 1 日で始まり、7 日で終了します。
X	はい	ローカル基数文字。 **例 :** 'HH:MI:SSXFF'
Y,YYY	はい	指定した位置にカンマを付けた年。

表 2-15 日時書式要素（続き）

要素	TO_* 日時ファンクションで指定できるか	説明
YEAR SYEAR		フルスペルで表した年。S を指定すると紀元前の日付の先頭に負の符号（-）が付けられます。
YYYY SYYYY	はい	4 桁で表した年。S を指定すると紀元前の日付の先頭に負の符号（-）が付けられます。
YYY YY Y	はい	それぞれ年の下 3 桁、2 桁、1 桁。

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

```
SELECT TO_CHAR (TO_DATE('0297','MM/YY'), 'MM/YY') FROM DUAL;

TO_CH
-----
02/07
```

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

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

> **参照：** 詳細は、2-63 ページの「書式モデルの修飾子」および 2-66 ページの「文字列から日付への変換に関する規則」を参照してください。

日時書式要素およびグローバリゼーション・サポート

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

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

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

日時書式要素 D は、曜日の数（1 ～ 7）を戻します。この数が 1 である曜日は、初期化パラメータ NLS_TERRITORY によって暗黙的に指定されます。

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

ISO 標準日付書式要素

Oracle は、ISO 規格に従った日時書式要素 IYYY、IYY、IY、I および IW によって戻される値を計算します。これらの値と日時書式要素 YYYY、YYY、YY、Y および WW によって戻される値の相違点については、『Oracle Database グローバリゼーション・サポート・ガイド』の「グローバリゼーション・サポート」の章を参照してください。

RR 日時書式要素

RR 日時書式要素は、YY 日時書式要素に似ていますが、20 世紀以外の日付値を格納する場合の柔軟性が優れています。RR 日時書式要素によって、現在が 21 世紀でも、年の下 2 桁を指定するだけで、20 世紀の日付を格納できます。

YY 日時書式要素で TO_DATE ファンクションを使用した場合、日付値は常に現在の年と同じ上 2 桁で戻されます。そのかわりに RR 日時書式要素を使用した場合、年に指定した 2 桁の数と現在の年の下 2 桁の数によって戻り値の世紀が変化します。

したがって、次のようになります。

- 指定された 2 桁の年が 00 ～ 49 の場合
  - 現在の年の下 2 桁が 00 ～ 49 であれば、戻される年は、現在の年と同じ上 2 桁を持ちます。
  - 現在の年の下 2 桁が 50 ～ 99 であれば、戻される年の上 2 桁は、現在の年の上 2 桁より 1 多くなります。
- 指定された 2 桁の年が 50 ～ 99 の場合
  - 現在の年の下 2 桁が 00 ～ 49 であれば、戻される年の上 2 桁は、現在の年の上 2 桁より 1 少なくなります。
  - 現在の年の下 2 桁が 50 ～ 99 であれば、戻される年は、現在の年と同じ上 2 桁を持ちます。

次に、RR 日時書式要素の特長を具体的に説明します。

RR 日時書式の例

次の問合せが、1950 年～ 1999 年の間に発行されるとします。

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

Year
----
1998

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

Year
----
2017
```

次の問合せが、2000 年～ 2049 年の間に発行されるとします。

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

Year
----
1998

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

```
Year
----
2017
```

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

日時書式要素の接尾辞

表 2-16 に、日時書式要素に付加できる接尾辞を示します。

表 2-16 日付書式要素の接尾辞

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

日時書式要素の接尾辞の注意事項：

- これらの接尾辞を 1 つでも日時書式要素に付加した場合、戻り値は常に英語です。
- 日時の接尾辞は出力書式設定にのみ有効です。データベースに日付を挿入するためには使用できません。

書式モデルの修飾子

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

修飾子は書式モデルに複数指定できます。この場合、後の修飾子は前の修飾子の効果を逆にします。第 1 の修飾子は、それに後続するモデル部分に対して有効になり、第 2 の修飾子が指定されると、その後のモデル部分で無効になります。第 3 の修飾子が指定されると、その後のモデル部分で再び有効になります。以降、同様に続きます。

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

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

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

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

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

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

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

書式モデルの例

次の文は、日付書式モデルを使用して文字式を戻します。

```
SELECT TO_CHAR(SYSDATE, 'fmDDTH')||' of '||TO_CHAR
   (SYSDATE, 'fmMonth')||', '||TO_CHAR(SYSDATE, 'YYYY') "Ides"
    FROM DUAL;

Ides
------------------
3RD of April, 1998
```

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

```
SELECT TO_CHAR(SYSDATE, 'DDTH')||' of '||
   TO_CHAR(SYSDATE, 'Month')||', '||
   TO_CHAR(SYSDATE, 'YYYY') "Ides"
   FROM DUAL;

Ides
-----------------------
03RD of April    , 1998
```

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

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

Menu
-----------------
Tuesday's Special
```

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

表 2-17 に、*char* 値と '*fmt*' の様々な組合せに対し、次の文が FX を使用した一致条件を満たしているかどうかを示します（table という名前の表には DATE データ型の列 date_column があります）。

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

表 2-17 FX 書式モデル修飾子による文字データと書式モデルの一致

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

戻り値の書式例 書式モデルを使用して、データベースから値を戻す場合に Oracle が使用する書式を指定できます。

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

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

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

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

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

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

> **参照：** fm 書式要素の説明については、2-63 ページの「書式モデルの修飾子」を参照してください。

正しい書式モデルの付与例 列の値を挿入または更新する場合、指定する値のデータ型は列のデータ型と一致している必要があります。書式モデルを使用して、あるデータ型の値を列が必要とする別のデータ型の値に変換する書式を指定できます。

たとえば、DATE 列に挿入する値は、DATE データ型の値か、またはデフォルト日付書式の文字列値である必要があります（Oracle は、暗黙的にデフォルト日付書式の文字列を DATE データ型に変換します）。値が別の書式で与えられる場合、TO_DATE ファンクションを使用して値を DATE データ型に変換する必要があります。また、文字列の書式を指定する場合にも、書式モデルを使用する必要があります。

次の文は、TO_DATE ファンクションを使用して Hunold の入社日を更新します。文字列 '1998 05 20' を DATE 値に変換するために、書式マスク 'YYYY MM DD' を指定します。

```
UPDATE employees
  SET hire_date = TO_DATE('1998 05 20','YYYY MM DD')
  WHERE last_name = 'Hunold';
```

文字列から日付への変換に関する規則

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

- 先行 0（ゼロ）を含む数値書式要素の桁がすべて指定されている場合は、日付文字列から書式文字列に含まれる句読点を省略できます。たとえば、MM、DD、YY などの 2 桁の書式要素については、2 のかわりに 02 を指定した場合です。
- 日付文字列から、書式文字列の最後にある時刻フィールドを省略できます。
- 日時書式要素と日付文字列内の対応する文字の一致に失敗した場合、表 2-18 に示すとおり、元の書式要素のかわりに、別の書式要素の適用が試みられます。

表 2-18 Oracle の書式一致

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

XML 書式モデル

SYS_XMLGEN ファンクションは、XML 文書を含む XMLType 型のインスタンスを戻します。Oracle では、SYS_XMLGEN ファンクションの結果をフォーマットする XMLFormat オブジェクトを提供します。

表 2-19 に、XMLFormat オブジェクトの属性を示します。表に示すように、ファンクションはこの型を実装します。

参照：

- SYS_XMLGEN ファンクションの詳細は、5-193 ページの「SYS_XMLGEN」を参照してください。
- XMLFormat オブジェクトの実装およびその使用の詳細は、『Oracle XML Developer's Kit プログラマーズ・ガイド』を参照してください。

表 2-19 XMLFormat オブジェクトの属性

属性	データ型	用途
enclTag	VARCHAR2(100)	SYS_XMLGEN ファンクションの結果の囲みタグ名です。ファンクションへの入力が列名である場合、デフォルトはその列名です。それ以外の場合、デフォルトは ROW です。schemaType を USE_GIVEN_SCHEMA に設定すると、この属性によって XML Schema の要素名を指定することもできます。
schemaType	VARCHAR2(100)	出力された文書のスキーマ生成の型です。有効な値は、'NO_SCHEMA' および 'USE_GIVEN_SCHEMA' です。デフォルトは、'NO_SCHEMA' です。
schemaName	VARCHAR2(4000)	schemaType の値が 'USE_GIVEN_SCHEMA' のときに、Oracle が使用するターゲット・スキーマの名前です。schemaName を指定すると、囲みタグが要素名として使用されます。
targetNameSpace	VARCHAR2(4000)	スキーマが指定されている場合（つまり、schemaType が GEN_SCHEMA_* または USE_GIVEN_SCHEMA の場合）のターゲット・ネームスペースです。

表 2-19 XMLFormat オブジェクトの属性（続き）

属性	データ型	用途
dburl	VARCHAR2(2000)	WITH_SCHEMA が指定されている場合に使用するデータベースの URL です。この属性が指定されていない場合、Oracle はその型への URL を相対 URL 参照として宣言します。
processingIns	VARCHAR2(4000)	ファンクションの出力の最上位に、要素の前に追加されるユーザーの処理命令です。

XMLFormat オブジェクトを実装するファンクションは、次のとおりです。

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

NULL

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

Oracle Database は、長さが 0（ゼロ）の文字値を NULL として処理します。ただし、NULL は値 0（ゼロ）と同じではないため、0（ゼロ）の数値を表すために NULL 値を使用しないでください。

> **注意：** 現在、Oracle Database は、長さが 0（ゼロ）の文字値を NULL として処理します。ただし、この処理は Oracle の今後のバージョンでも継続されるとはかぎらないため、空の文字列を NULL として処理しないことをお薦めします。

NULL を含む算術式は、必ず NULL に評価されます。たとえば、NULL に 10 を加算しても結果は NULL です。実際、オペランドに NULL を指定した場合、（連結演算子を除く）すべての演算子は NULL を戻します。

SQL ファンクションでの NULL

ほとんどのスカラー・ファンクションでは、引数として NULL を指定すると NULL が戻されます。NVL ファンクションを使用した場合、NULL が発生したときに値を戻すことができます。たとえば、式 NVL(commission_pct,0) は、commission_pct が NULL の場合は 0（ゼロ）を戻し、commission_pct が NULL でなければその値を戻します。

集計ファンクションによる NULL の処理の詳細は、5-8 ページの「集計ファンクション」を参照してください。

比較条件での NULL

NULL を検査するには、比較条件 IS NULL および IS NOT NULL のみを使用します。NULL を他の条件で使用して、その結果が NULL の値に依存する場合、結果は UNKNOWN になります。NULL はデータの欠落を表すため、任意の値や別の NULL との関係で等号や不等号は成り立ちません。ただし、Oracle は DECODE ファンクションを評価するときに 2 つの NULL を等しい値とみなします。構文および追加情報については、5-53 ページの「DECODE」を参照してください。

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

条件での NULL

UNKNOWN として評価される条件は、FALSE と評価される場合とほとんど同じ働きをします。たとえば、UNKNOWN と評価される条件を WHERE 句に持つ SELECT 文からは、行が戻されません。ただし、UNKNOWN と評価される条件は FALSE 条件とは異なり、UNKNOWN 条件をさらに評価しても UNKNOWN と評価されます。したがって、NOT FALSE は TRUE と評価されますが、NOT UNKNOWN は UNKNOWN と評価されます。

表 2-20 は、条件に NULL を含む評価の例です。SELECT 文の WHERE 句で UNKNOWN と評価される条件が使用された場合、その問合せに対して行は戻されません。

表 2-20 NULL を含む条件

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

NULL を含む論理条件の結果を示した真理値表は、7-8 ページの表 7-5、7-8 ページの表 7-6、および 7-9 ページの表 7-7 を参照してください。

コメント

次の 2 種類のコメントを作成できます。

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

SQL 文中のコメント

コメントは、アプリケーションを読みやすく、メンテナンスしやすくします。たとえば、文にはアプリケーションでのその文の目的を記述したコメントを含めることができます。SQL 文中のコメントは文の実行には影響しませんが、ヒントは例外です。この特殊なコメント形式を使用する場合の詳細は、2-70 ページの「ヒントの使用方法」を参照してください。

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

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

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

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

例　次の文には多くのコメントが含まれています。

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

SELECT last_name,                    -- select the name
    salary + NVL(commission_pct, 0),-- total compensation
    job_id,                         -- job
    e.department_id                 -- and department
  FROM employees e,                 -- of all employees
       departments d
  WHERE e.department_id = d.department_id
    AND salary + NVL(commission_pct, 0) >  -- whose compensation
                                           -- is greater than
      (SELECT salary + NVL(commission_pct,0)  -- the compensation
    FROM employees
    WHERE last_name = 'Pataballa')        -- of Pataballa.
;
```

スキーマ・オブジェクトおよび非スキーマ・オブジェクトに関するコメント

COMMENT コマンドを使用して、スキーマ・オブジェクト（表、ビュー、マテリアライズド・ビュー、演算子、索引タイプ、マイニング・モデル）にコメントを付けることができます。表スキーマ・オブジェクトの一部である列にもコメントを作成できます。スキーマ・オブジェクトおよび非スキーマ・オブジェクトに付けたコメントは、データ・ディクショナリに格納されます。このコメントの形式の詳細は、13-41 ページの「COMMENT」を参照してください。

ヒントの使用方法

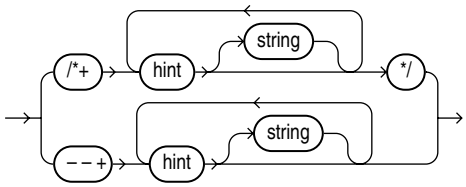
Oracle Database のオプティマイザに指示（**ヒント**）を与えるために、SQL 文中でコメントを使用できます。オプティマイザは、オプティマイザの動作を阻止する条件が存在しないかぎり、これらのヒントを使用して文の実行計画を選択します。

> **注意：** ヒントを使用する場合は注意が必要です。ヒントは、関連する表の統計情報を収集し、ヒントを使用せずに EXPLAIN PLAN 文を使用してオプティマイザ計画を評価した後で使用してください。今回のリリース以降では、データベースの条件の変更と問合せのパフォーマンス強化によって、コード内のヒントがパフォーマンスに重大な影響を与えることに注意してください。

文ブロックには、ヒントを含むコメントは 1 つのみ指定できます。このコメントは、SELECT、UPDATE、INSERT、MERGE または DELETE のいずれかのキーワードの後でのみ指定できます。INSERT 文では次の 2 つのヒントのみを使用します。APPEND ヒントは常に INSERT キーワードの後で指定し、PARALLEL ヒントは INSERT キーワードの後で指定できます。

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

***hint*::=**



それぞれの意味は、次のとおりです。

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

--+ 構文では、コメント全体を単一行で指定する必要があります。

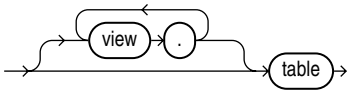
Oracle Database は、次の状況ではヒントを無視し、エラーを戻しません。

- ヒントにスペルの誤りまたは構文エラーがある場合。ただし、データベースは、同一コメント内に正しく指定された他のヒントがある場合はそれを採用します。
- ヒントを含むコメントが、DELETE、INSERT、MERGE、SELECT または UPDATE のいずれかのキーワードの後で指定していない場合。

- ヒントの組合せが互いに競合している場合。ただし、データベースは、同一コメント内に他のヒントがあればそれを採用します。
- データベース環境が、Forms バージョン 3 トリガー、Oracle Forms 4.5、Oracle Reports 2.5 など、PL/SQL バージョン 1 を使用している場合。

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

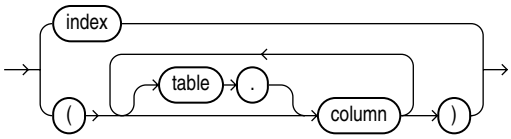
***tablespec*::=**



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

> **参照：** 次の内容の詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。
>
> - グローバル・ヒントを使用するタイミングとその解析方法
> - EXPLAIN PLAN を使用して、オプティマイザが問合せを実行する方法の確認
> - ビュー内の表へのヒントの参照

***indexspec*::=**



ヒントの仕様で *tablespec* の後に *indexspec* を指定する場合、表名と索引名をカンマで区切ることができます。ただし、このカンマは必須ではありません。*indexspec* を複数指定する場合もカンマで区切ることができます（必須ではありません）。

ヒントでの問合せブロックの指定

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

問合せブロックの引数の構文は、*@queryblock* の形式になります。ここで *queryblock* は問合せ内で問合せブロック指定する識別子です。*queryblock* 識別子は、システムで生成されるかまたはユーザーによって指定されます。ヒントが適用される問合せブロック内でヒントを指定する場合は、*@queryblock* 構文を指定しません。

- システム生成の識別子は、問合せに対する EXPLAIN PLAN 文を使用して取得できます。変換前の問合せブロック名は、NO_QUERY_TRANSFORMATION ヒントを使用している問合せに対して EXPLAIN PLAN を実行することで調べることができます。2-87 ページの「NO_QUERY_TRANSFORMATION ヒント」を参照してください。
- ユーザー指定の名前は QB_NAME ヒントで指定できます。2-93 ページの「QB_NAME ヒント」を参照してください。

表 2-21 に、機能のカテゴリに分類したヒントと、各ヒントの構文とセマンティクスの参照先を示します。表の後には、ヒントをアルファベット順に説明します。

> **参照：** 次の内容の詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。
>
> - ヒントを使用した SQL 文の最適化と、*tablespec* および *indexspec* 構文の使用方法の詳細。
> - ヒントでの問合せブロックの指定。
> - ヒントのカテゴリ分類の説明と、それを使用するタイミング。

表 2-21 機能のカテゴリに分類したヒント

ヒント	構文とセマンティクスの参照先
最適化目標と方法	ALL_ROWS ヒント（2-74 ページ） FIRST_ROWS ヒント（2-77 ページ）
アクセス・パスのヒント	CLUSTER ヒント（2-75 ページ）
--	FULL ヒント（2-77 ページ）
--	HASH ヒント（2-77 ページ）
--	INDEX ヒント（2-78 ページ） NO_INDEX ヒント（2-84 ページ）
--	INDEX_ASC ヒント（2-78 ページ） INDEX_DESC ヒント（2-79 ページ）
--	INDEX_COMBINE ヒント（2-79 ページ）
--	INDEX_JOIN ヒント（2-80 ページ）
--	INDEX_FFS ヒント（2-79 ページ）
--	INDEX_SS ヒント（2-80 ページ）
--	INDEX_SS_ASC ヒント（2-80 ページ）
--	INDEX_SS_DESC ヒント（2-81 ページ）
--	NATIVE_FULL_OUTER_JOIN（2-82 ページ） NO_NATIVE_FULL_OUTER_JOIN（2-85 ページ）
--	NO_INDEX_FFS ヒント（2-84 ページ）
--	NO_INDEX_SS ヒント（2-85 ページ）
結合順序のヒント	ORDERED ヒント（2-90 ページ）
--	LEADING ヒント（2-81 ページ）
結合操作のヒント	USE_HASH ヒント（2-96 ページ） NO_USE_HASH ヒント（2-88 ページ）
--	USE_MERGE ヒント（2-96 ページ） NO_USE_MERGE ヒント（2-88 ページ）
--	USE_NL ヒント（2-96 ページ） USE_NL_WITH_INDEX ヒント（2-97 ページ） NO_USE_NL ヒント（2-89 ページ）

表 2-21 機能のカテゴリに分類したヒント（続き）

ヒント	構文とセマンティクスの参照先
パラレル実行のヒント	PARALLEL ヒント（2-90 ページ） NO_PARALLEL ヒント（2-86 ページ）
--	PARALLEL_INDEX ヒント（2-91 ページ） NO_PARALLEL_INDEX ヒント（2-86 ページ）
--	PQ_DISTRIBUTE ヒント（2-91 ページ）
問合せ変換のヒント	FACT ヒント（2-76 ページ） NO_FACT ヒント（2-84 ページ）
--	MERGE ヒント（2-82 ページ） NO_MERGE ヒント（2-85 ページ）
--	NO_EXPAND ヒント（2-83 ページ） USE_CONCAT ヒント（2-95 ページ）
--	REWRITE ヒント（2-94 ページ） NO_REWRITE ヒント（2-87 ページ）
--	UNNEST ヒント（2-95 ページ） NO_UNNEST ヒント（2-88 ページ）
--	STAR_TRANSFORMATION ヒント（2-94 ページ） NO_STAR_TRANSFORMATION ヒント（2-88 ページ）
--	NO_QUERY_TRANSFORMATION ヒント（2-87 ページ）
XML のヒント	NO_XMLINDEX_REWRITE ヒント（2-89 ページ）
--	NO_XML_QUERY_REWRITE ヒント（2-89 ページ）
その他のヒント	APPEND ヒント（2-74 ページ） NOAPPEND ヒント（2-83 ページ）
--	CACHE ヒント（2-75 ページ） NOCACHE ヒント（2-83 ページ）
--	CURSOR_SHARING_EXACT ヒント（2-75 ページ）
--	DRIVING_SITE ヒント（2-75 ページ）
--	DYNAMIC_SAMPLING ヒント（2-76 ページ）
--	MODEL_MIN_ANALYSIS ヒント（2-82 ページ）
--	MONITOR ヒント（2-82 ページ）
--	NO_MONITOR ヒント（2-85 ページ）
--	OPT_PARAM ヒント（2-89 ページ）
--	PUSH_PRED ヒント（2-92 ページ） NO_PUSH_PRED ヒント（2-86 ページ）
--	PUSH_SUBQ ヒント（2-93 ページ） NO_PUSH_SUBQ ヒント（2-87 ページ）

表 2-21 機能のカテゴリに分類したヒント（続き）

ヒント	構文とセマンティクスの参照先
--	PX_JOIN_FILTER ヒント（2-93 ページ） NO_PX_JOIN_FILTER ヒント（2-87 ページ）
--	QB_NAME ヒント（2-93 ページ）
--	RESULT_CACHE ヒント（2-93 ページ） NO_RESULT_CACHE ヒント（2-87 ページ）

ヒントのリスト（アルファベット順）

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

ALL_ROWS ヒント

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

```
SELECT /*+ ALL_ROWS */ employee_id, last_name, salary, job_id
  FROM employees
  WHERE employee_id = 7566;
```

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

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

APPEND ヒント

APPEND ヒントは、ダイレクト・パス・インサート文を使用するようオプティマイザに指示します。

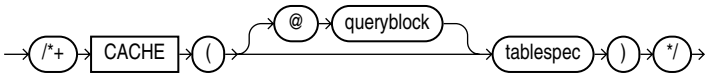
- 従来型の INSERT はシリアル・モードでのデフォルトです。シリアル・モードでは、ダイレクト・パスは APPEND ヒントを指定する場合にのみ使用できます。
- ダイレクト・パス・インサートはパラレル・モードでのデフォルトです。パラレル・モードでは、従来型 INSERT は NOAPPEND ヒントを指定する場合にのみ使用できます。

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

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

> **参照：** NOAPPEND ヒントの詳細は、2-83 ページの「NOAPPEND ヒント」を参照してください。ダイレクト・パス・インサートの詳細は、『Oracle Database 管理者ガイド』を参照してください。

CACHE ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照)

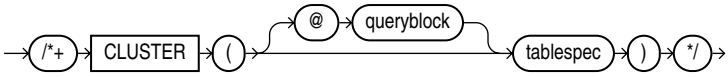
CACHE ヒントは、全表スキャンの実行時に、この表に対して取り出されたブロックを、バッファ・キャッシュ内の LRU リストの最高使用頻度側に入れるようオプティマイザに指定します。このヒントは、小規模な参照表で有効です。

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

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

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

CLUSTER ヒント



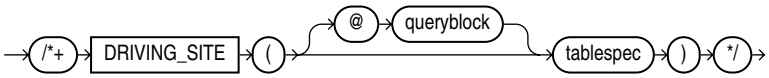
(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照)

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

CURSOR_SHARING_EXACT ヒント

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

DRIVING_SITE ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照)

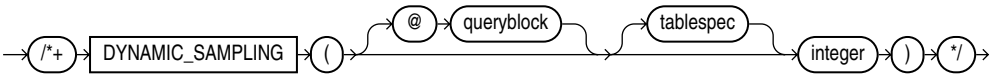
DRIVING_SITE ヒントは、データベースによって選択されたサイトとは異なるサイトで問合せを実行するようオプティマイザに指示します。このヒントは、分散化された問合せの最適化を使用している場合に有効です。

次に例を示します。

```
SELECT /*+ DRIVING_SITE(departments) */ *
  FROM employees, departments@rsite
  WHERE employees.department_id = departments.department_id;
```

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

DYNAMIC_SAMPLING ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照)

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

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

integer の値は 0 ～ 10 となり、サンプリングの度合いを示します。

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

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

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

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

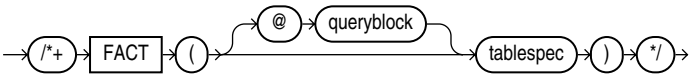
- 単一表述語がある場合、オプティマイザは既存のカーディナリティ統計情報を使用し、この既存の統計情報を使用して述語の選択性を推定します。

動的なサンプリングを特定の表に適用するには、次の形式のヒントを使用します。

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

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

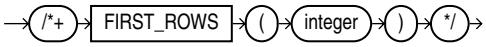
FACT ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照)

FACT ヒントは、スター型変換のコンテキストで使用されます。このヒントは、*tablespec* 内で指定されている表をファクト表とみなす必要があることをオプティマイザに指示します。

FIRST_ROWS ヒント



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

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

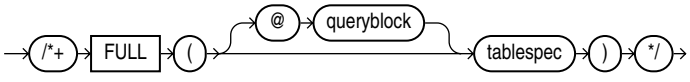
```
SELECT /*+ FIRST_ROWS(10) */ employee_id, last_name, salary, job_id
  FROM employees
  WHERE department_id = 20;
```

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

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

> **参照：** FIRST_ROWS ヒントと統計情報の詳細は、2-74 ページの「ALL_ROWS ヒント」を参照してください。

FULL ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照)

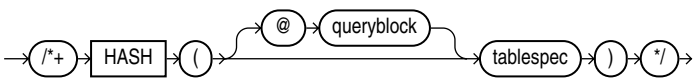
FULL ヒントは、指定した表に対して全表スキャンを実行するようオプティマイザに指示します。次に例を示します。

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

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

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

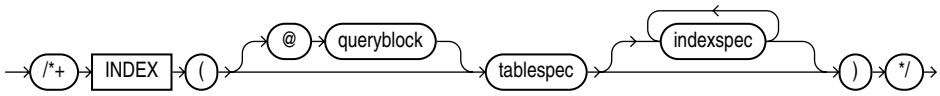
HASH ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照)

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

INDEX ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::=、2-71 ページの indexspec::= を参照)

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

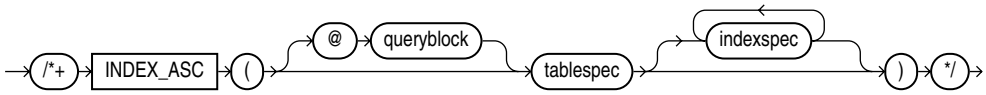
ヒントの動作は、*indexspec* の仕様によって異なります。

- INDEX ヒントが使用可能な単一の索引を指定すると、データベースはこの索引でスキャンを実行します。オプティマイザは、全表スキャンおよび表上の別の索引のスキャンを考慮しません。
- 複数の索引の組合せに対するヒントの場合、INDEX ではなく INDEX_COMBINE ヒントの使用をお薦めします。これは、後者のほうがより多目的なヒントであるためです。INDEX ヒントが使用可能な索引のリストを指定すると、オプティマイザは、リスト内の各索引でのスキャンのコストを検討し、最低のコストで索引スキャンを実行します。アクセス・パスのコストが最低となる場合、データベースは、このリストから複数の索引をスキャンするように選択し、結果をマージすることもあります。データベースは、全表スキャンおよびヒント内でリスト化されていない索引でのスキャンを検討しません。
- INDEX ヒントが索引を指定しない場合、オプティマイザは、表にある使用可能な各索引でのスキャンのコストを検討し、最低のコストで索引スキャンを実行します。アクセス・パスのコストが最低となる場合、データベースは複数の索引をスキャンするように選択し、結果をマージすることもあります。オプティマイザは全表スキャンを検討しません。

次に例を示します。

```
SELECT /*+ INDEX (employees emp_department_ix)*/
       employee_id, department_id
  FROM employees
  WHERE department_id > 50;
```

INDEX_ASC ヒント

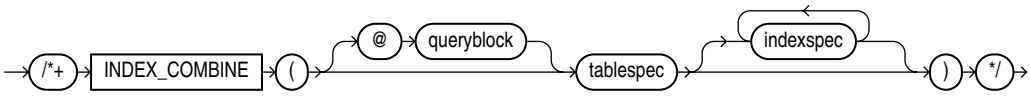


(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::=、2-71 ページの indexspec::= を参照)

INDEX_ASC ヒントは、指定した表について索引スキャンを使用するようオプティマイザに指示します。文で索引レンジ・スキャンを使用する場合、Oracle Database は索引付きの値の昇順で索引エントリをスキャンします。各パラメータは、2-78 ページの「INDEX ヒント」と同じ目的で使用されます。

レンジ・スキャンのデフォルトの動作は、索引付きの値の昇順で索引エントリをスキャンします。降順索引の場合は、降順でスキャンします。このヒントは索引のデフォルトの順序を変更しないため、INDEX ヒントにないものは指定しません。ただし、デフォルトの動作を変更する必要がある場合は、INDEX_ASC ヒントを使用して昇順レンジ・スキャンを明示的に指定できます。

INDEX_COMBINE ヒント

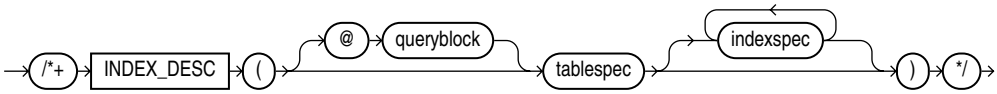


(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::=、2-71 ページの indexspec::= を参照)

INDEX_COMBINE ヒントは、表へのビットマップ・アクセス・パスを使用するようオプティマイザに指示します。*indexspec* が INDEX_COMBINE ヒントから省略されている場合、オプティマイザは、表のスキャンにかかるコスト効率が最大になる索引のブールの組合せを使用します。indexspec を指定すると、オプティマイザは、指定した索引のブールのいくつかの組合せの使用を試行します。各パラメータは、2-78 ページの「INDEX ヒント」と同じ目的で使用されます。次に例を示します。

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

INDEX_DESC ヒント



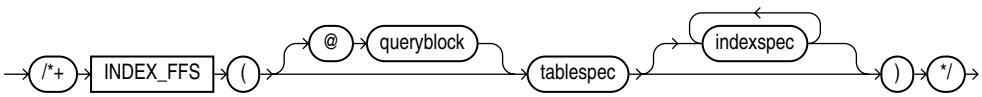
(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::=、2-71 ページの indexspec::= を参照)

INDEX_DESC ヒントは、指定した表に降順索引スキャンを使用するようオプティマイザに指示します。文で索引レンジ・スキャンを使用しており、索引が昇順の場合、Oracle は索引付きの値の降順で索引エントリをスキャンします。パーティション索引では、結果は各パーティション内で降順になります。降順索引の場合、このヒントは降順を効果的に取り消すため、索引エントリは昇順でスキャンされます。各パラメータは、2-78 ページの「INDEX ヒント」と同じ目的で使用されます。次に例を示します。

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

> **参照：** 全体スキャンの詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。

INDEX_FFS ヒント



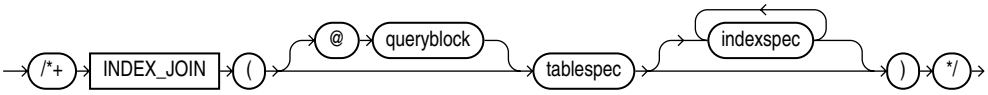
(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::=、2-71 ページの indexspec::= を参照)

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

各パラメータは、2-78 ページの「INDEX ヒント」と同じ目的で使用されます。次に例を示します。

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

INDEX_JOIN ヒント



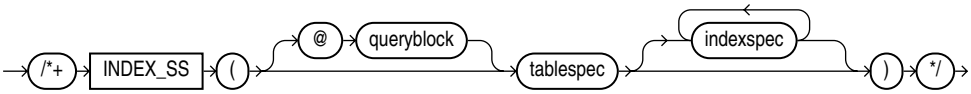
(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::=、2-71 ページの indexspec::= を参照)

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

各パラメータは、2-78 ページの「INDEX ヒント」と同じ目的で使用されます。たとえば、次の問合せは、索引結合を使用して manager_id および department_id 列にアクセスします。これらの列はどちらも、employees 表内で索引付けされています。

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

INDEX_SS ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::=、2-71 ページの indexspec::= を参照)

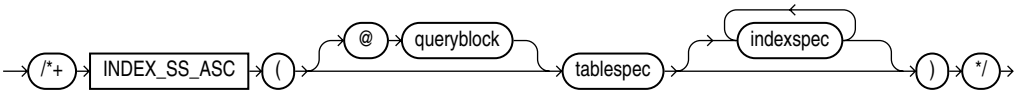
INDEX_SS ヒントは、指定した表について索引スキップ・スキャンを実行するようオプティマイザに指示します。文で索引レンジ・スキャンを使用する場合、Oracle は索引付きの値の昇順で索引エントリをスキャンします。パーティション索引では、結果は各パーティション内で昇順になります。

各パラメータは、2-78 ページの「INDEX ヒント」と同じ目的で使用されます。次に例を示します。

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

> **参照：** 索引スキップ・スキャンの詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。

INDEX_SS_ASC ヒント



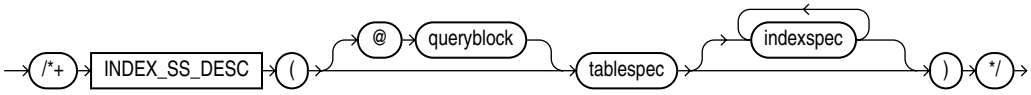
(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::=、2-71 ページの indexspec::= を参照)

INDEX_SS_ASC ヒントは、指定した表について索引スキップ・スキャンを実行するようオプティマイザに指示します。文で索引レンジ・スキャンを使用する場合、Oracle Database は索引付きの値の昇順で索引エントリをスキャンします。パーティション索引では、結果は各パーティション内で昇順になります。各パラメータは、2-78 ページの「INDEX ヒント」と同じ目的で使用されます。

レンジ・スキャンのデフォルトの動作は、索引付きの値の昇順で索引エントリをスキャンします。降順索引の場合は、降順でスキャンします。このヒントは索引のデフォルトの順序を変更しないため、INDEX_SS ヒントにないものは指定しません。ただし、デフォルトの動作を変更する必要がある場合は、INDEX_SS_ASC ヒントを使用して昇順レンジ・スキャンを明示的に指定できます。

> **参照：** 索引スキップ・スキャンの詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。

INDEX_SS_DESC ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::=、2-71 ページの indexspec::= を参照)

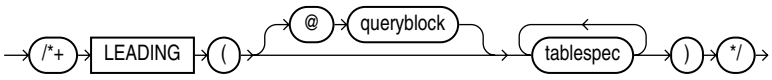
INDEX_SS_DESC ヒントは、指定した表について索引スキップ・スキャンを実行するようオプティマイザに指示します。文で索引レンジ・スキャンを使用しており、索引が昇順の場合、Oracle は索引付きの値の降順で索引エントリをスキャンします。パーティション索引では、結果は各パーティション内で降順になります。降順索引の場合、このヒントは降順を効果的に取り消すため、索引エントリは昇順でスキャンされます。

各パラメータは、2-78 ページの「INDEX ヒント」と同じ目的で使用されます。次に例を示します。

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

> **参照：** 索引スキップ・スキャンの詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。

LEADING ヒント



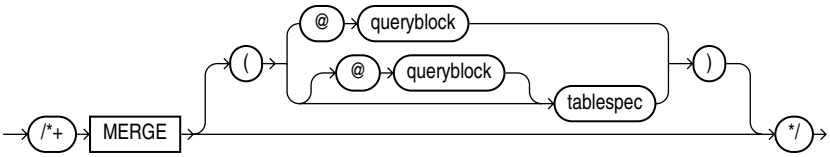
(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照)

LEADING ヒントは、実行計画において、指定した一連の表を接頭辞として使用するようオプティマイザに指示します。このヒントは、ORDERED ヒントより多目的なヒントです。次に例を示します。

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

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

MERGE ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照)

MERGE ヒントを使用すると、問合せ内のビューをマージできます。

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

次に例を示します。

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

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

MODEL_MIN_ANALYSIS ヒント

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

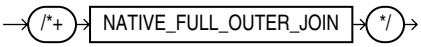
スプレッドシート・ルールの数が数百を超えると、スプレッドシート分析に長い時間がかかることがあるため、このヒントによってコンパイル時間を減らします。

MONITOR ヒント

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

> **参照：** リアルタイムの SQL 監視の詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。

NATIVE_FULL_OUTER_JOIN



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

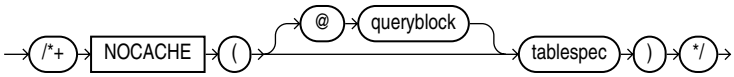
参照：

- 「NO_NATIVE_FULL_OUTER_JOIN」（2-85 ページ）
- ネイティブ完全外部結合の詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。

NOAPPEND ヒント

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

NOCACHE ヒント



（2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照）

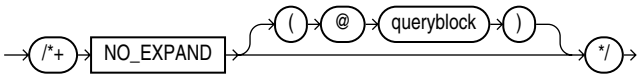
NOCACHE ヒントは、全表スキャンの実行時に、この表に対して取り出されたブロックを、バッファ・キャッシュ内の LRU リストの最低使用頻度側に入れるようオプティマイザに指定します。これは、バッファ・キャッシュ内のブロックの通常動作です。次に例を示します。

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

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

参照： 表のサイズによって異なる表の自動キャッシングの詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。

NO_EXPAND ヒント



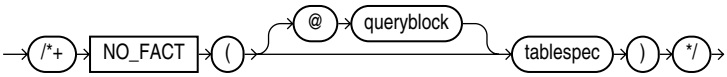
（2-71 ページの「ヒントでの問合せブロックの指定」を参照）

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

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

参照： 2-95 ページの「USE_CONCAT ヒント」を参照してください（このヒントの反対の機能です）。

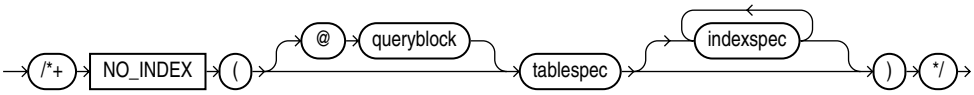
NO_FACT ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照)

NO_FACT ヒントは、スター型変換のコンテキストで使用されます。このヒントは、問合せ対象の表をファクト表とみなす必要がないことをオプティマイザに指示します。

NO_INDEX ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::=、2-71 ページの indexspec::= を参照)

NO_INDEX ヒントは、指定した表について 1 つ以上の索引を使用しないようオプティマイザに指示します。次に例を示します。

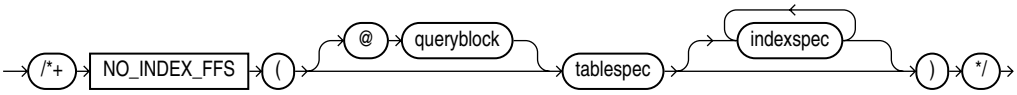
```
SELECT /*+ NO_INDEX(employees emp_empid) */ employee_id
  FROM employees
  WHERE employee_id > 200;
```

各パラメータは 2-78 ページの「INDEX ヒント」と同じ目的で使用されますが、次の変更が加えられています。

- このヒントが使用可能な単一の索引を指定すると、オプティマイザはこの索引でのスキャンを検討しません。その他の指定されていない索引については、スキャンを検討します。
- このヒントが使用可能な索引のリストを指定すると、オプティマイザは指定された索引でのスキャンを検討しません。リスト内に指定されていないその他の索引については、スキャンを検討します。
- このヒントが索引を指定しない場合、オプティマイザは表のいずれの索引でのスキャンを考慮しません。この動作は、表について使用可能なすべての索引のリストを指定する NO_INDEX ヒントと同様になります。

NO_INDEX ヒントは、ファンクション索引、B ツリー索引、ビットマップ索引、クラスタ索引およびドメイン索引に適用されます。NO_INDEX ヒントと索引ヒント（INDEX、INDEX_ASC、INDEX_DESC、INDEX_COMBINE または INDEX_FFS）の両方が同じ索引を指定する場合、データベースは、NO_INDEX ヒントと、指定した索引の索引ヒントの両方を無視し、これらの索引を文の実行中に使用することを検討します。

NO_INDEX_FFS ヒント

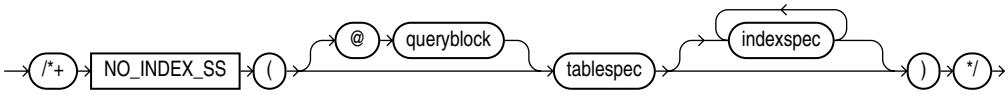


(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::=、2-71 ページの indexspec::= を参照)

NO_INDEX_FFS ヒントは、指定された表の指定された索引の高速全索引スキャンを除外するようオプティマイザに指示します。各パラメータは、2-78 ページの「INDEX ヒント」と同じ目的で使用されます。次に例を示します。

```
SELECT /*+ NO_INDEX_FFS(items item_order_ix) */ order_id
  FROM order_items items;
```

NO_INDEX_SS ヒント

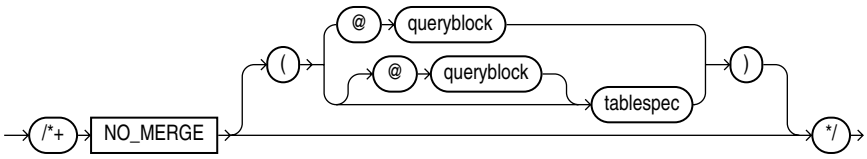


(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::=、2-71 ページの indexspec::= を参照)

NO_INDEX_SS ヒントは、指定された表の指定された索引のスキップ・スキャンを除外するようオプティマイザに指示します。各パラメータは、2-78 ページの「INDEX ヒント」と同じ目的で使用されます。

> **参照：** 索引スキップ・スキャンの詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。

NO_MERGE ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照)

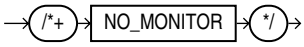
NO_MERGE ヒントは、外部問合せとインライン・ビュー問合せを結合して単一の問合せにしないようオプティマイザに指示します。

このヒントを使用すると、ビューへのアクセス方法に強い影響を与えることができます。たとえば、次の文は、ビュー seattle_dept がマージされないようにします。

```
SELECT /*+NO_MERGE(seattle_dept)*/ e1.last_name, seattle_dept.department_name
  FROM employees e1,
    (SELECT location_id, department_id, department_name
       FROM departments
       WHERE location_id = 1700) seattle_dept
  WHERE e1.department_id = seattle_dept.department_id;
```

ビューの問合せブロック内で NO_MERGE ヒントを使用する場合は、引数なしで指定します。また、周囲の問合せで NO_MERGE を指定する場合には、ビュー名を引数として指定します。

NO_MONITOR ヒント



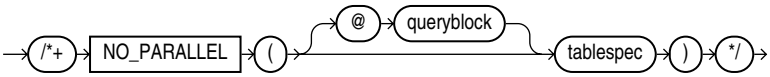
NO_MONITOR ヒントは、問合せの実行時間が長い場合でも、問合せの SQL のリアルタイム監視を無効にします。

NO_NATIVE_FULL_OUTER_JOIN

NO_NATIVE_FULL_OUTER_JOIN ヒントは、指定された各表を結合する場合にネイティブ実行メソッドを除外するようオプティマイザに指示します。かわりに、完全外部結合は、左側外部結合とアンチ結合の論理和として実行されます。

> **参照：** 「NATIVE_FULL_OUTER_JOIN」(2-82 ページ)

NO_PARALLEL ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照)

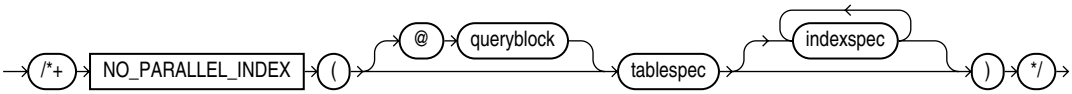
NO_PARALLEL ヒントは、表を作成するか変更した DDL 内の PARALLEL パラメータを上書きします。次に例を示します。

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

NOPARALLEL ヒント

NOPARALLEL ヒントは現在非推奨になっています。NO_PARALLEL ヒントをかわりに使用します。

NO_PARALLEL_INDEX ヒント



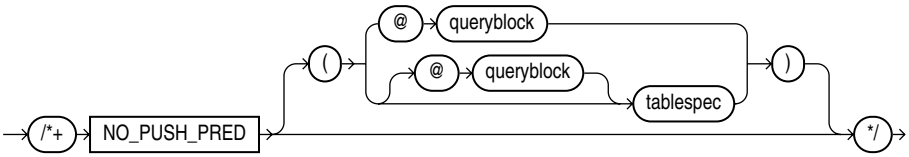
(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::=、2-71 ページの indexspec::= を参照)

NO_PARALLEL_INDEX ヒントは、索引を作成するか変更した DDL 内の PARALLEL パラメータを上書きし、パラレル索引スキャン操作を防止します。

NOPARALLEL_INDEX ヒント

NOPARALLEL_INDEX ヒントは現在非推奨になっています。NO_PARALLEL_INDEX ヒントをかわりに使用します。

NO_PUSH_PRED ヒント

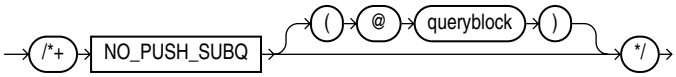


(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照)

NO_PUSH_PRED ヒントは、結合述語をビューにプッシュしないようオプティマイザに指示します。次に例を示します。

```
SELECT /*+ NO_MERGE(v) NO_PUSH_PRED(v) */ *
    FROM employees e,
         (SELECT manager_id
            FROM employees
         ) v
  WHERE e.manager_id = v.manager_id(+)
    AND e.employee_id = 100;
```

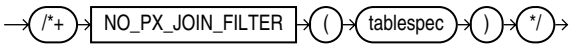
NO_PUSH_SUBQ ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」を参照)

NO_PUSH_SUBQ ヒントは、実行計画における最後の手順として、マージされていない副問合せを評価するようオプティマイザに指示します。これにより、副問合せのコストが比較的高い場合や、副問合せによって行数が大幅に減少しない場合に、パフォーマンスが向上する場合があります。

NO_PX_JOIN_FILTER ヒント



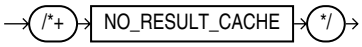
このヒントは、オプティマイザがパラレル結合のビットマップ・フィルタ処理を使用するのを防ぎます。

NO_QUERY_TRANSFORMATION ヒント

NO_QUERY_TRANSFORMATION ヒントは、すべての問合せ変換をスキップするようオプティマイザに指示します。この中には、OR 拡張、ビュー・マージ、副問合せのネスト解除、スター型変換、マテリアライズド・ビュー・リライトなどが含まれますが、これのみに限定されません。次に例を示します。

```
SELECT /*+ NO_QUERY_TRANSFORMATION */ employee_id, last_name
  FROM (SELECT *
        FROM employees e) v
  WHERE v.last_name = 'Smith';
```

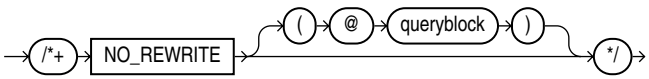
NO_RESULT_CACHE ヒント



RESULT_CACHE_MODE 初期化パラメータが FORCE に設定されていると、オプティマイザは、問合せ結果を結果キャッシュにキャッシュします。この場合、NO_RESULT_CACHE ヒントにより、このような現行の問合せのキャッシュが無効になります。

問合せが OCI クライアントから実行され、OCI クライアントの結果キャッシュが有効になっている場合、NO_RESULT_CACHE ヒントにより現行の問合せキャッシュが無効になります。

NO_REWRITE ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」を参照)

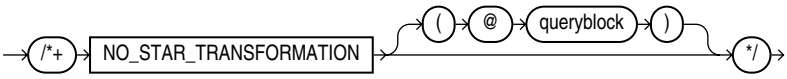
NO_REWRITE ヒントは、パラメータ QUERY_REWRITE_ENABLED の設定を上書きして、問合せブロック用のクエリー・リライトを無効にするようオプティマイザに指示します。次に例を示します。

```
SELECT /*+ NO_REWRITE */ sum(s.amount_sold) AS dollars
  FROM sales s, times t
  WHERE s.time_id = t.time_id
  GROUP BY t.calendar_month_desc;
```

NOREWRITE ヒント

NOREWRITE ヒントは現在非推奨になっています。NO_REWRITE ヒントをかわりに使用します。

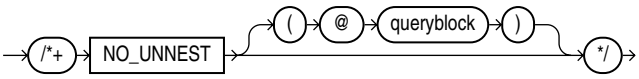
NO_STAR_TRANSFORMATION ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」を参照)

NO_STAR_TRANSFORMATION ヒントは、問合せのスター型問合せ変換を実行しないようオプティマイザに指示します。

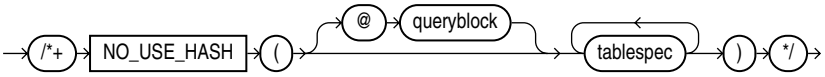
NO_UNNEST ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」を参照)

NO_UNNEST ヒントを使用するとネスト解除をオフに切り替えます。

NO_USE_HASH ヒント

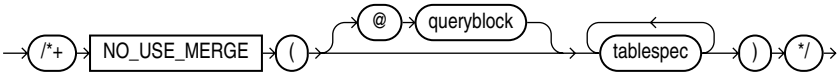


(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照)

NO_USE_HASH ヒントは、指定された表を内部表として使用して、指定された各表を別の行のソースに結合する際にハッシュ結合を除外するようオプティマイザに指示します。次に例を示します。

```
SELECT /*+ NO_USE_HASH(e d) */ *
  FROM employees e, departments d
  WHERE e.department_id = d.department_id;
```

NO_USE_MERGE ヒント

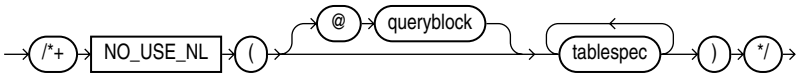


(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照)

NO_USE_MERGE ヒントは、指定された表を内部表として使用して、指定された各表を別の行のソースに結合する際にソート / マージ結合を除外するようオプティマイザに指示します。次に例を示します。

```
SELECT /*+ NO_USE_MERGE(e d) */ *
   FROM employees e, departments d
   WHERE e.department_id = d.department_id
   ORDER BY d.department_id;
```

NO_USE_NL ヒント



（2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照）

NO_USE_NL ヒントは、指定された表を内部表として使用して、指定された各表を別の行のソースに結合する際に、ネストしたループ結合を除外するようオプティマイザに指示します。次に例を示します。

```
SELECT /*+ NO_USE_NL(l h) */ *
  FROM orders h, order_items l
  WHERE l.order_id = h.order_id
    AND l.order_id > 3500;
```

このヒントを指定すると、指定された表についてハッシュ結合とソート / マージ結合のみが検討されます。ただし、ネストしたループのみを使用して表を結合する場合もあります。この場合、オプティマイザは、それらの表に関するヒントを無視します。

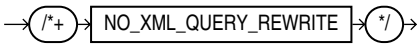
NO_XMLINDEX_REWRITE ヒント

NO_XMLINDEX_REWRITE ヒントは、現行の問合せに XMLIndex 索引を使用しないようにオプティマイザに指示します。次に例を示します。

```
SELECT /*+NO_XMLINDEX_REWRITE*/ count(*)
  FROM table WHERE existsNode(OBJECT_VALUE, '/*') = 1;
```

> **参照：** XMLIndex の使用を無効にするもう 1 つの方法については、2-89 ページの「NO_XML_QUERY_REWRITE ヒント」を参照してください。

NO_XML_QUERY_REWRITE ヒント

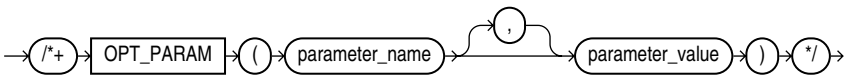


NO_XML_QUERY_REWRITE ヒントは、SQL 文の XPath 式のリライトを禁止するようオプティマイザに指示します。このヒントは、XPath 式のリライトを禁止することで、現行の問合せでの XMLIndex 索引の使用も禁止します。次に例を示します。

```
SELECT /*+NO_XML_QUERY_REWRITE*/ XMLQUERY('<A/>')
    FROM dual;
```

> **参照：** 「NO_XMLINDEX_REWRITE ヒント」（2-89 ページ）

OPT_PARAM ヒント



OPT_PARAM ヒントを使用すると、現行の問合せ中にのみ初期化パラメータを設定できます。このヒントは、パラメータ OPTIMIZER_DYNAMIC_SAMPLING、OPTIMIZER_INDEX_CACHING、OPTIMIZER_INDEX_COST_ADJ、OPTIMIZER_SECURE_VIEW_MERGING および STAR_TRANSFORMATION_ENABLED に対してのみ有効です。たとえば、次のヒントは、ヒントを追加した文のパラメータ STAR_TRANSFORMATION_ENABLED を TRUE に設定します。

```
SELECT /*+ OPT_PARAM('star_transformation_enabled' 'true') */ * FROM ... ;
```

文字列のパラメータ値は、一重引用符で囲まれます。数値のパラメータ値は、一重引用符で囲まずに指定されます。

ORDERED ヒント

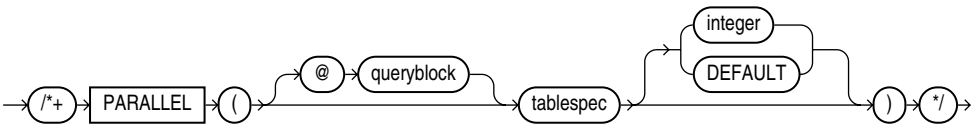
ORDERED ヒントは、FROM 句に現れる順序で表を結合するよう Oracle に指示します。ORDERED ヒントより多目的な LEADING ヒントを使用することをお薦めします。

結合を要求する SQL 文から ORDERED ヒントを削除すると、オプティマイザが表の結合順序を選択します。各表から選択した行数をオプティマイザが把握していないと考えられる場合、ORDERED ヒントを使用して結合順序を指定することがあります。この情報を使用すると、オプティマイザによる選択よりも効率良く内部表と外部表を選択できます。

次の問合せは、ORDERED ヒントの使用例です。

```
SELECT  /*+ORDERED */ o.order_id, c.customer_id, l.unit_price * l.quantity
  FROM customers c, order_items l, orders o
  WHERE c.cust_last_name = :b1
    AND o.customer_id = c.customer_id
    AND o.order_id = l.order_id;
```

PARALLEL ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照)

PARALLEL ヒントは、指定された数の同時サーバーをパラレル操作に使用するようオプティマイザに指示します。このヒントは、文の SELECT、INSERT、MERGE、UPDATE および DELETE 部分と表のスキャン部分に適用されます。

> **注意：** ソート操作またはグループ操作も実行する場合、使用可能なサーバー数は PARALLEL ヒントの値の 2 倍となります。

パラレル制限に違反すると、ヒントは無視されます。

integer 値は、指定された表の並列度を指定します。DEFAULT を指定するか、いかなる値も指定しない場合、問合せコーディネータはデフォルトの並列度を決定するために初期化パラメータの設定を検証する必要があります。次の例では、PARALLEL ヒントが、employees 表定義内で指定された並列度を上書きします。

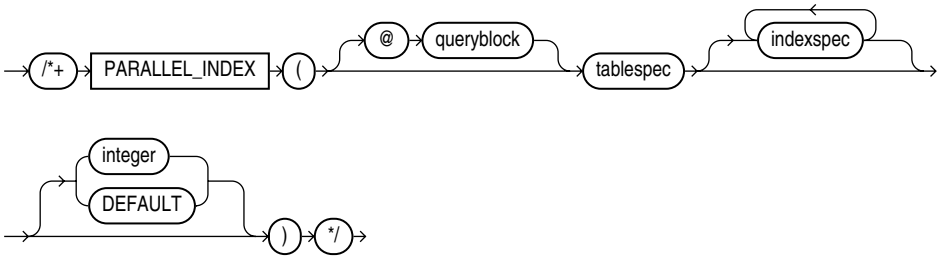
```
SELECT /*+ FULL(hr_emp) PARALLEL(hr_emp, 5) */ last_name
  FROM employees hr_emp;
```

次の例では、PARALLEL ヒントが、employees 表定義内で指定された並列度を上書きし、初期化パラメータが決定したデフォルトの並列度を使用するようオプティマイザに指示します。

```
SELECT /*+ FULL(hr_emp) PARALLEL(hr_emp, DEFAULT) */ last_name
  FROM employees hr_emp;
```

Oracle は、一時表に関するパラレル・ヒントを無視します。パラレル実行の詳細は、16-6 ページの「CREATE TABLE」および『Oracle Database 概要』を参照してください。

PARALLEL_INDEX ヒント



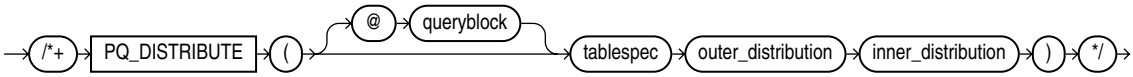
(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::=、2-71 ページの indexspec::= を参照)

PARALLEL_INDEX ヒントは、パーティション索引について索引レンジ・スキャン、全体スキャンおよび高速全体スキャンをパラレル化するために、指定された数の同時サーバーを使用するようオプティマイザに指示します。

integer 値は、指定された索引の並列度を示します。DEFAULT を指定するか、いかなる値も指定しない場合、問合せコーディネータはデフォルトの並列度を決定するために初期化パラメータの設定を検証する必要があります。たとえば、次のヒントは、3 つのパラレル実行プロセスが使用されることを示します。

```
SELECT /*+ PARALLEL_INDEX(table1, index1, 3) */
```

PQ_DISTRIBUTE ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照)

PQ_DISTRIBUTE ヒントは、結合表の行を、プロデューサおよびコンシューマ問合せサーバーに分散させる方法をオプティマイザに指示します。この分散処理により、パラレル結合操作のパフォーマンスが向上します。

- *outer_distribution* は、外部表への分散処理です。
- *inner_distribution* は、内部表への分散処理です。

分散の値は、HASH、BROADCAST、PARTITION および NONE です。表 2-22 で説明するとおり、6 つの組合せの表分散のみが有効です。

表 2-22 分散ヒントの組合せ

分散	説明
HASH、HASH	各表の行は、結合キーにハッシュ関数を使用して、コンシューマ問合せサーバーにマップされます。マッピングが完了すると、各問合せサーバーは、結果として生成されるパーティションの組で結合を実行します。この分散は、表のサイズがほぼ等しく、結合操作がハッシュ結合またはソート / マージ結合で実施される場合にお薦めします。
BROADCAST、NONE	外部表のすべての行が各問合せサーバーにブロードキャストされます。内部表の行は、ランダムにパーティション化されます。この分散は、外部表のサイズが内部表よりもきわめて小さい場合にお薦めします。一般的に、内部表のサイズに問合せサーバーの数を乗じた数値が外部表のサイズよりも大きい場合、この分散を使用します。
NONE、BROADCAST	内部表のすべての行が各コンシューマ問合せサーバーにブロードキャストされます。外部表の行は、ランダムにパーティション化されます。この分散は、内部表のサイズが外部表よりもきわめて小さい場合にお薦めします。一般的に、内部表のサイズに問合せサーバーの数を乗じた数値が外部表のサイズより小さい場合、この分散を使用します。

表 2-22 分散ヒントの組合せ（続き）

分散	説明
PARTITION、NONE	外部表の行は、内部表のパーティション化を使用してマップされます。内部表は、結合キーでパーティション化されている必要があります。この分散は、外部表のパーティションの数が問合せサーバーの数と等しいかほぼ等しい（たとえば、パーティション数が 14 で問合せサーバー数が 15）場合にお薦めします。 **注意：**オプティマイザは、内部表がパーティション化されていないか、パーティション化キーと等価結合関係にない場合、このヒントを無視します。
NONE、PARTITION	内部表の行は、外部表のパーティション化を使用してマップされます。外部表は、結合キーでパーティション化されている必要があります。この分散は、外部表のパーティションの数が問合せサーバーの数と等しいかほぼ等しい（たとえば、パーティション数が 14 で問合せサーバー数が 15）場合にお薦めします。 **注意：**オプティマイザは、外部表がパーティション化されていないか、パーティション化キーと等価結合関係にない場合、このヒントを無視します。
NONE、NONE	各問合せサーバーは、各表から抽出した一致パーティションの組で結合操作を実行します。両方の表は、結合キーに基づいて同一レベルでパーティション化されている必要があります。

たとえば、r と s という 2 つの表がハッシュ結合によって結合されている場合、次の問合せには、ハッシュ分散を使用するためのヒントが含まれます。

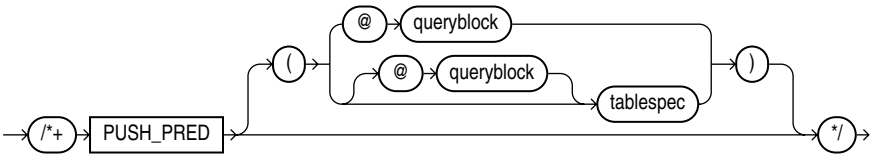
```
SELECT /*+ORDERED PQ_DISTRIBUTE(s HASH, HASH) USE_HASH (s)*/ column_list
  FROM r,s
  WHERE r.c=s.c;
```

外部表 r をブロードキャストするための問合せは、次のとおりです。

```
SELECT /*+ORDERED PQ_DISTRIBUTE(s BROADCAST, NONE) USE_HASH (s) */ column_list
  FROM r,s
  WHERE r.c=s.c;
```

> **参照：** Oracle での結合操作のパラレル化の詳細は、『Oracle Database 概要』を参照してください。

PUSH_PRED ヒント

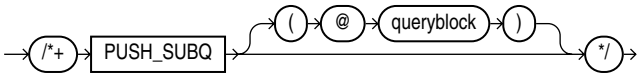


（2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照）

PUSH_PRED ヒントは、結合述語をビューにプッシュするようオプティマイザに指示します。次に例を示します。

```
SELECT /*+ NO_MERGE(v) PUSH_PRED(v) */ *
    FROM employees e,
         (SELECT manager_id
            FROM employees
         ) v
  WHERE e.manager_id = v.manager_id(+)
    AND e.employee_id = 100;
```

PUSH_SUBQ ヒント

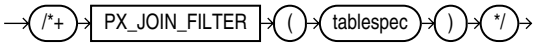


(2-71 ページの「ヒントでの問合せブロックの指定」を参照)

PUSH_SUBQ ヒントは、実行計画の初期段階で実行可能な手順で、マージされていない副問合せを評価するようオプティマイザに指示します。通常、マージされていない副問合せは、実行計画の最終手順で実行されます。副問合せのコストが比較的低く、副問合せによって行数が大幅に減少する場合、副問合せの早期評価によってパフォーマンスが向上する可能性があります。

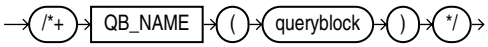
このヒントは、副問合せがリモート表か、マージ結合によって結合されている表に適用される場合には効果がありません。

PX_JOIN_FILTER ヒント



このヒントは、パラレル結合のビットマップ・フィルタ処理の使用をオプティマイザに強制します。

QB_NAME ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」を参照)

QB_NAME ヒントを使用すると、問合せブロックの名前を定義できます。この名前は外部問合せ内のヒントまたはインライン・ビュー内のヒントで使用でき、名前が付けられた問合せブロックにある表で実行する問合せに影響をおよぼします。

2 つ以上の問合せブロックが同じ名前の場合、または同じ問合せブロックが異なる名前でヒントが 2 回行われている場合、オプティマイザは、その問合せブロックを参照するすべての名前とヒントを無視します。このヒントを使用して名前が付けられていない問合せブロックには、システムが生成する一意の名前が付けられます。この名前は計画表に表示できます。また、問合せブロック内のヒントや問合せブロック・ヒントでも使用できます。次に例を示します。

```
SELECT /*+ QB_NAME(qb) FULL(@qb e) */ employee_id, last_name
  FROM employees e
  WHERE last_name = 'Smith';
```

RESULT_CACHE ヒント

RESULT_CACHE ヒントは、メモリー内の現行の問合せまたは問合せのフラグメントの結果をキャッシュし、今後の問合せまたは問合せのフラグメントの実行時にキャッシュした結果を使用するようデータベースに指示します。このヒントは、トップレベル問合せ、*subquery_factoring_clause* または FROM 句のインライン・ビューで認識されます。キャッシュ結果は、共有プールの結果のキャッシュ・メモリー部分に保存されます。

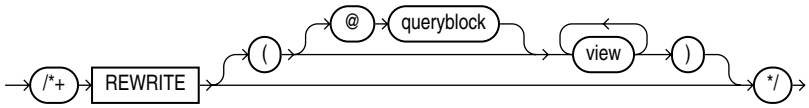
作成に使用されたデータベース・オブジェクトが正常に修正されると、キャッシュ結果は自動的に無効化されます。このヒントは、RESULT_CACHE_MODE 初期化パラメータの設定よりも優先されます。

問合せが結果キャッシュに使用できるのは、問合せで必要とされるすべてのファンクション(たとえば、組込みファンクション、ユーザー定義ファンクションまたは仮想列)が決定的である場合にかぎられます。

問合せが OCI クライアントから実行され、OCI クライアントの結果キャッシュが有効になっている場合、RESULT_CACHE ヒントにより現行の問合せのクライアントのキャッシュが有効になります。

> **参照：** このヒントの使用方法については、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。RESULT_CACHE_MODE 初期化パラメータについては、『Oracle Database リファレンス』を参照してください。OCI の結果のキャッシュの詳細および使用のガイドラインは、『Oracle Call Interface プログラマーズ・ガイド』を参照してください。

REWRITE ヒント



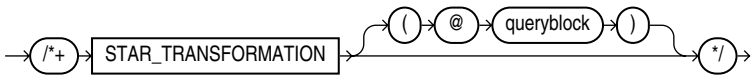
(2-71 ページの「ヒントでの問合せブロックの指定」を参照)

REWRITE ヒントは、可能な場合、コストを考慮することなく、マテリアライズド・ビューに関する問合せをリライトするようオプティマイザに指示します。REWRITE ヒントは、ビュー・リストとともに、またはビュー・リストなしで使用します。ビュー・リストとともに REWRITE を使用し、リストに適切なマテリアライズド・ビューが含まれている場合、Oracle はコストを考慮せずにそのビューを使用します。

Oracle では、リスト外のビューを検討しません。ビュー・リストを指定しない場合、Oracle は適切なマテリアライズド・ビューを検索し、最終計画のコストを考慮することなく常にそのビューを使用します。

> **参照：**
>
> - マテリアライズド・ビューの詳細は、『Oracle Database 概要』および『Oracle Database アドバンスト・レプリケーション』を参照してください。
> - マテリアライズド・ビューで REWRITE を使用する場合の詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

STAR_TRANSFORMATION ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」を参照)

STAR_TRANSFORMATION ヒントは、変換を行う際に最適な計画を使用するようオプティマイザに指示します。このヒントを使用しない場合、オプティマイザは、変換された問合せ用の最適な計画のかわりに、変換なしで生成された最適な計画を使用するという、問合せの最適化に関する決定を行う場合があります。次に例を示します。

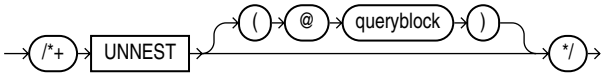
```
SELECT /*+ STAR_TRANSFORMATION */  *
  FROM sales s, times t, products p, channels c
  WHERE s.time_id = t.time_id
    AND s.prod_id = p.product_id
    AND s.channel_id = c.channel_id
    AND p.product_status = 'obsolete';
```

ヒントが指定された場合でも、変換が実行される保証はありません。オプティマイザは、妥当と考えられる場合にかぎって副問合せを生成します。副問合せが生成されない場合には、変換された問合せが存在しないため、ヒントに関係なく、未変換の問合せに関する最適な計画が使用されます。

参照：

- スター型変換の詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。
- STAR_TRANSFORMATION_ENABLED 初期化パラメータの詳細は、『Oracle Database リファレンス』を参照してください。

UNNEST ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」を参照)

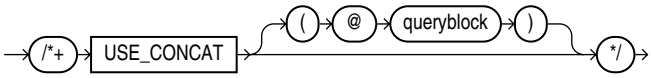
UNNEST ヒントは、副問合せの本体のネストを解除し、その副問合せを含む問合せブロック本体にマージするようオプティマイザに指示します。これによって、アクセス・パスおよび結合の評価時に、オプティマイザが副問合せと問合せブロックを総合して考慮できるようになります。

副問合せのネストを解除する前に、オプティマイザは、文が有効かどうかをまず検討します。文は、経験則に基づくテストと問合せ最適化テストに合格する必要があります。UNNEST ヒントは、副問合せブロックの有効性のみをチェックするようオプティマイザに指示します。副問合せブロックが有効な場合、経験則またはコストをチェックすることなく副問合せのネストを解除できます。

参照：

- ネスト化された副問合せのネスト解除、および副問合せブロックが有効となる条件については、19-45 ページの「コレクションのネスト解除例 :」を参照してください。
- 副問合せのネスト解除の詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。

USE_CONCAT ヒント



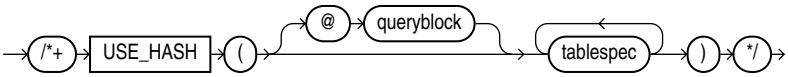
(2-71 ページの「ヒントでの問合せブロックの指定」を参照)

USE_CONCAT ヒントは、問合せの WHERE 句内で組み合わされた OR 条件を、集合演算子 UNION ALL を使用して複合問合せに変換するようオプティマイザに指示します。このヒントを使用しない場合、この変換は、連結を使用した問合せのコストが、使用しない場合よりも低い場合にのみ実行されます。USE_CONCAT ヒントは、コストより優先します。次に例を示します。

```
SELECT /*+ USE_CONCAT */ *
  FROM employees e
  WHERE manager_id = 108
     OR department_id = 110;
```

参照： OR 拡張の詳細については、このヒントの反対の機能である 2-83 ページの「NO_EXPAND ヒント」、および『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。

USE_HASH ヒント

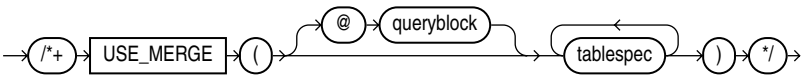


(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照)

USE_HASH ヒントは、指定された各表を、ハッシュ結合を使用して別の行のソースに結合するようオプティマイザに指示します。次に例を示します。

```
SELECT /*+ USE_HASH(l h) */ *
  FROM orders h, order_items l
  WHERE l.order_id = h.order_id
    AND l.order_id > 3500;
```

USE_MERGE ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照)

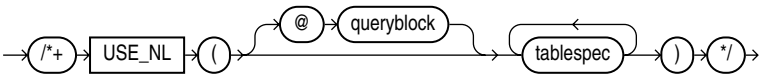
USE_MERGE ヒントは、指定された各表を、ソート / マージ結合を使用して別の行のソースに結合するようオプティマイザに指示します。次に例を示します。

```
SELECT /*+ USE_MERGE(employees departments) */ *
  FROM employees, departments
  WHERE employees.department_id = departments.department_id;
```

LEADING および ORDERED ヒントとともに、USE_NL および USE_MERGE ヒントを使用することをお薦めします。オプティマイザは、参照表を結合の内部表にする必要がある場合に、これらのヒントを使用します。参照表が外部表の場合、ヒントは無視されます。

USE_NL ヒント

USE_NL ヒントは、指定された表を内部表として使用し、指定された各表をネストしたループ結合とともに別の行のソースに結合するようオプティマイザに指示します。



(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::= を参照)

USE_NL ヒントは、指定された表を内部表として使用し、指定された各表をネストしたループ結合とともに別の行のソースに結合するようオプティマイザに指示します。

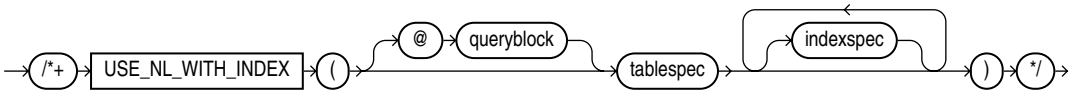
LEADING および ORDERED ヒントとともに、USE_NL および USE_MERGE ヒントを使用することをお薦めします。オプティマイザは、参照表を結合の内部表にする必要がある場合に、これらのヒントを使用します。参照表が外部表の場合、ヒントは無視されます。

次の例では、ネストしたループがヒントによって強制される場合、全表スキャンを介して orders がアクセスされ、各行にフィルタ条件 l.order_id = h.order_id が適用されます。フィルタ条件を満たす各行については、索引 order_id を介して order_items がアクセスされます。

```
SELECT /*+ USE_NL(l h) */ h.customer_id, l.unit_price * l.quantity
  FROM orders h ,order_items l
  WHERE l.order_id = h.order_id;
```

INDEX ヒントを問合せに追加すると、orders の全表スキャンを回避し、より大規模なシステムで使用される実行計画と同様の実行計画を生成できる場合があります。ただし、ここでは特に効果的ではない場合があります。

USE_NL_WITH_INDEX ヒント



(2-71 ページの「ヒントでの問合せブロックの指定」、2-71 ページの tablespec::=、2-71 ページの indexspec::= を参照)

USE_NL_WITH_INDEX ヒントは、指定された表を内部表として使用し、指定された表をネストしたループ結合とともに別の行のソースに結合するようオプティマイザに指示します。次に例を示します。

```
SELECT /*+ USE_NL_WITH_INDEX(l item_product_ix) */ *
  FROM orders h, order_items l
  WHERE l.order_id = h.order_id
    AND l.order_id > 3500;
```

次の条件が適用されます。

- 索引を指定しない場合、オプティマイザは、1 つ以上の結合述語とともに、索引キーとして一部の索引を使用できる必要があります。
- 索引を指定する場合、オプティマイザは、1 つ以上の結合述語とともに、索引キーとしてその索引を使用できる必要があります。

データベース・オブジェクト

次の項で説明するとおり、Oracle Database は、特定のスキーマに対応付けられたオブジェクトと、特定のスキーマに対応付けられていないオブジェクトを認識します。

スキーマ・オブジェクト

スキーマは、論理的なデータの構造（スキーマ・オブジェクト）の集まりです。スキーマは、データベース・ユーザーによって所有され、そのユーザーと同じ名前を持ちます。各ユーザーは、1 つのスキーマを所有します。スキーマ・オブジェクトは、SQL を使用して作成および操作できます。スキーマ・オブジェクトには次のタイプのオブジェクトがあります。

クラスタ
制約
データベース・リンク
データベース・トリガー
ディメンション
外部プロシージャ・ライブラリ
索引構成表
索引
索引タイプ
Java クラス、Java リソース、Java ソース
マテリアライズド・ビュー
マテリアライズド・ビュー・ログ
マイニング・モデル
オブジェクト表
オブジェクト型
オブジェクト・ビュー
演算子
パッケージ
順序
ストアド・ファンクション、ストアド・プロシージャ
シノニム
表
ビュー

非スキーマ・オブジェクト

次のタイプのオブジェクトもデータベースに格納され、SQL で作成および操作されますが、スキーマには含まれません。

コンテキスト
ディレクトリ
プロファイル
リストア・ポイント
ロール
ロールバック・セグメント
表領域
ユーザー

各タイプのオブジェクトは、このマニュアルの第 10 章～第 19 章のデータベース・オブジェクトを作成する文の項で説明されています。これらの文は、キーワード CREATE で始まります。たとえば、クラスタの定義については、14-2 ページの「CREATE CLUSTER」を参照してください。

> **参照：** データベース・オブジェクトの概要については、『Oracle Database 概要』を参照してください。

ほとんどのデータベース・オブジェクトでは、作成時に名前を指定する必要があります。名前は、この後の項に示す規則に従って付けてください。

スキーマ・オブジェクト名および修飾子

スキーマ・オブジェクトの中には、名前を付けることができる、または名前を付ける必要のある部分で構成されるものもあります。たとえば、表やビューの中の列、索引と表のパーティションおよびサブパーティション、表に対する整合性制約、パッケージ内に格納されるオブジェクト（プロシージャおよびストアド・ファンクションを含む）などです。この項では、次の内容について説明します。

- スキーマ・オブジェクトとスキーマ・オブジェクトの位置修飾子のネーミング規則
- スキーマ・オブジェクトと修飾子のネーミングのガイドライン

> **注意：** Oracle では、暗黙的に生成されるスキーマ・オブジェクトとサブオブジェクトには「SYS_」で始まるシステム生成名を使用し、Oracle が提供する一部のオブジェクトには、「ORA_」で始まる名前を使用します。名前解決での競合を避けるため、これらの接頭辞は明示的に指定するスキーマ・オブジェクト名やサブオブジェクト名では使用しないでください。

スキーマ・オブジェクトのネーミング規則

すべてのデータベース・オブジェクトには、名前があります。SQL 文では、**引用識別子**または**非引用識別子**を使用して、オブジェクトの名前を表します。

- 引用識別子は、二重引用符（"）で囲みます。引用識別子を使用してスキーマ・オブジェクトを指定した場合、そのオブジェクトを参照するときは、必ず二重引用符を使用します。
- 非引用識別子の前後に句読点は付けません。

データベース・オブジェクトを指定する場合、引用識別子または非引用識別子を使用できます。データベース名、グローバル・データベース名およびデータベース・リンク名では大 / 小文字は区別されませんが、大文字として保存されます。このような名前を引用識別子として指定する場合、引用符は特に警告もなく無視されます。ユーザー名とパスワードのネーミング規則の詳細は、17-7 ページの「CREATE USER」を参照してください。

特に指定がないかぎり、次の規則は、引用識別子と非引用識別子の両方に適用されます。

1. 名前は、1 ～ 30 バイトの長さで指定する必要があります。ただし、次の 2 つは例外です。
   - データベースの名前は、8 バイトまでに制限されています。
   - データベース・リンクの名前は、128 バイトまで指定できます。

   識別子にピリオドで区切られた複数の部分が含まれる場合、各属性の長さは最大 30 バイトにできます。ピリオドによる各セパレータおよび周囲の二重引用符は 1 バイトとしてカウントします。たとえば、次のように列を識別するとします。

   ```
   "schema"."table"."column"
   ```

   スキーマ名、表名、列名の長さはそれぞれ 30 バイトです。各引用符やピリオドはシングルバイト文字のため、この例での識別子の総バイト長は、最大 98 バイトになります。

2. 非引用識別子に Oracle Database の予約語は使用できません。引用識別子には、予約語を使用できますが、お薦めしません。

   名前は、データベース・オブジェクトにアクセスするために使用する Oracle 製品固有のその他の予約語によって、さらに制限されることもあります。

   注意： 予約語 ROWID は、この規則の例外です。引用識別子または非引用識別子のいずれであっても、大文字の ROWID を列名として使用することはできません。ただし、列名ではない引用識別子には大文字を使用できます。また列名を含む引用識別子には、1 つ以上の小文字（たとえば、"Rowid" や "rowid"）を使用できます。

   参照：
   - Oracle Database の予約語のすべてのリストは、付録 D「Oracle Database 予約語」を参照してください。
   - 製品の予約語のリストについては、『Oracle Database PL/SQL 言語リファレンス』などの各製品のマニュアルを参照してください。

3. Oracle の SQL 言語には、特別な意味を持つ文字が含まれています。これらの文字には、データ型、スキーマ名、ファンクション名、ダミーのシステム表 DUAL およびキーワード（DIMENSION、SEGMENT、ALLOCATE、DISABLE など、SQL 文中の大文字の単語）が含まれます。これらの文字は予約語ではありません。ただし、Oracle は固有の方法でこれらの文字を内部的に使用します。したがって、これらの文字をオブジェクトおよびオブジェクトの部分の名前として使用した場合、使用している SQL 文が読みにくくなり、予期しない結果になることがあります。

   特に、SYS_ または ORA_ で始まる文字をスキーマ・オブジェクト名として使用しないでください。また、SQL 組込みファンクションの名前を、スキーマ・オブジェクトまたはユーザー定義ファンクションの名前として使用しないでください。

   参照： 2-2 ページの「データ型」、5-2 ページの「SQL ファンクション」および 9-14 ページの「DUAL 表からの選択」を参照してください。

4. 異なるプラットフォームおよびオペレーティング・システム間では、ASCII 文字を使用することで最適な互換性を得ることができます。データベース名、グローバル・データベース名、データベース・リンク名には ASCII 文字を使用してください。

5. 非引用識別子は、データベース・キャラクタ・セットのアルファベット文字で開始する必要があります。引用識別子の開始文字には、任意の文字を使用できます。

6. 非引用識別子には、データベース・キャラクタ・セットの英数字、アンダースコア（_）、ドル記号（$）およびシャープ記号（#）のみ含めることができます。データベース・リンクの名前には、ピリオド（.）とアットマーク（@）を含めることもできます。非引用識別子では、$ と # はできるだけ使用しないでください。

   引用識別子には、すべての文字、句読点および空白を使用できます。ただし、引用識別子と非引用識別子のいずれにも、二重引用符または NULL 文字（たとえば \0）は使用できません。

7. ネームスペース内では、2 つのオブジェクトに同じ名前を付けることはできません。

   次のスキーマ・オブジェクトは、1 つのネームスペースを共有します。

   - 表
   - ビュー
   - 順序
   - プライベート・シノニム
   - スタンドアロン・プロシージャ
   - スタンドアロン・ストアド・ファンクション
   - パッケージ
   - マテリアライズド・ビュー
   - ユーザー定義型

   次の各スキーマ・オブジェクトは、固有のネームスペースを持ちます。

   - 索引
   - 制約
   - クラスタ
   - データベース・トリガー
   - プライベート・データベース・リンク
   - ディメンション

   表およびビューが同じネームスペースにあるため、同じスキーマの表およびビューが同じ名前を持つことはできません。ただし、表と索引は異なるネームスペースに存在します。このため、同じスキーマ内の表と索引には、同じ名前を付けることができます。

   データベース内の各スキーマには、その中のオブジェクトのために固有のネームスペースがあります。たとえば、異なるスキーマ内の 2 つの表は異なるネームスペースに存在し、同じ名前を付けることができます。

   次の各非スキーマ・オブジェクトは、固有のネームスペースを持ちます。

   - ユーザー・ロール
   - パブリック・シノニム
   - パブリック・データベース・リンク
   - 表領域
   - プロファイル
   - パラメータ・ファイル（PFILE）およびサーバー・パラメータ・ファイル（SPFILE）

   これらのネームスペース内のオブジェクトはスキーマに含まれないため、これらのネームスペースはデータベース全体で使用されます。

8. 非引用識別子は、大 / 小文字が区別されず、すべて大文字として解析されます。引用識別子は、大 / 小文字が区別されます。

   名前を二重引用符で囲むことによって、同じネームスペース内の異なるオブジェクトに対して次の名前を指定できます。

   ```
   employees
   "employees"
   "Employees"
   "EMPLOYEES"
   ```

   ただし、Oracle は次の名前を同じ名前として解析するため、同じネームスペース内の異なるオブジェクトには、次の名前を使用できません。

   ```
   employees
   EMPLOYEES
   "EMPLOYEES"
   ```

9. Oracle が識別子を大文字で格納または比較する場合、識別子の各文字の大文字形式は、データベース・キャラクタ・セットの大文字化規則を適用することによって決定されます。セッション設定 NLS_SORT によって決定される言語固有の規則は考慮されません。この動作は、ファンクション NLS_UPPER ではなく、SQL ファンクション UPPER を識別子に適用することに対応しています。

   データベース・キャラクタ・セットの大文字化規則によって、特定の自然言語としては不適切な結果となる場合があります。たとえば、データベース・キャラクタ・セットの大文字化規則に従うと、ドイツ語で使用される小文字のシャープ s（ß）に大文字形式はありません。この文字は識別子が大文字に変換されるときに変更されませんが、ドイツ語で予期される大文字形式は 2 文字連続の大文字 S（SS）です。同様に、小文字 i の大文字形式は、データベース・キャラクタ・セットの大文字化規則に従うと大文字 I ですが、トルコ語およびアゼルバイジャン語で予期される大文字形式は、上に点が付いた大文字 I です。

   データベース・キャラクタ・セットの大文字化規則では、識別子はセッションのすべての言語構成で同様に解釈されます。識別子が特定の自然言語で正しく表示されるようにする場合は、引用符で囲んで小文字形式を保持するか、またはその識別子を使用するときは常に言語的に正しい大文字形式を使用することができます。

10. 同じ表やビューでは、複数の列に同じ名前を付けることはできません。ただし、異なる表やビューでは、複数の列に同じ名前を付けることができます。

11. 引数の数およびデータ型が異なる場合、同じパッケージに含まれるプロシージャやファンクションに同じ名前を付けることができます。異なる引数を持ち、同じ名前のプロシージャやファンクションを同じパッケージ内に複数作成することを、**オーバーロード**といいます。

スキーマ・オブジェクトのネーミング例

次に、有効なスキーマ・オブジェクト名の例を示します。

```
last_name
horse
hr.hire_date
"EVEN THIS & THAT!"
a_very_long_and_valid_name
```

これらのすべての例は、2-98 ページの「スキーマ・オブジェクトのネーミング規則」に示す規則に従っています。次の例は、30 文字を超えているため、無効となります。

```
a_very_very_long_and_valid_name
```

列別名、表別名、ユーザー名およびパスワードは、オブジェクトまたはオブジェクトの部分ではありませんが、特に指定がないかぎり、同様にこれらのネーミング規則に従う必要があります。

スキーマ・オブジェクトのネーミングのガイドライン

オブジェクトとその部分に名前を付ける場合に有効なガイドラインを次に示します。

- わかりやすい名前（またはよく知られている省略形）を使用します。
- 一貫したネーミング規則を使用します。
- 複数の表にまたがる同一のエンティティや属性を記述するためには、同一の名前を使用します。

オブジェクトに名前を付ける場合は、短くて簡単な名前とわかりやすい名前のバランスを考えてください。迷ったときには、わかりやすい名前にしてください。これは、データベース内のオブジェクトは、多くの人々が長期間にわたって使用する可能性があるためです。`payment_due_date` のかわりに `pmdd` という名前を使用すると、10 年後の担当者は表の列の理解に苦労することになります。

一貫したネーミング規則を使用すると、アプリケーション上の各表の働きが理解しやすくなります。そのような規則の例として、`FINANCE` アプリケーションに属している表の名前をすべて `fin_` で始めるような場合が考えられます。

同一のエンティティや属性に対しては、複数の表にまたがっていても同じ名前を使用してください。たとえば、`employees` サンプル表と `departments` サンプル表の部門番号列には、どちらにも `department_id` という名前を付けます。

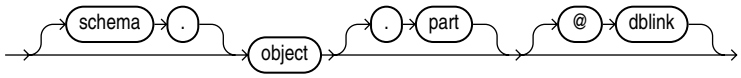
スキーマ・オブジェクトの構文および SQL 文の構成要素

SQL 文のコンテキストでスキーマ・オブジェクトとそれらの部分を参照する方法について説明します。次の項目について説明します。

- オブジェクトを参照するための一般的な構文
- Oracle がオブジェクトへの参照を変換する方法
- 自分のスキーマ以外のスキーマ内のオブジェクトを参照する方法
- リモート・データベース内のオブジェクトを参照する方法
- 表と索引のパーティションおよびサブパーティションを参照する方法

次に、オブジェクトやそれらの部分を参照するための一般的な構文を示します。

***database_object_or_part*::=**



それぞれの意味は、次のとおりです。

- *object* は、オブジェクトの名前です。
- *schema* は、オブジェクトを含むスキーマです。この修飾子を指定することによって、自分のスキーマ以外のスキーマ内のオブジェクトを参照できます。その場合には、自分のスキーマ以外のスキーマ内のオブジェクトを参照するための権限が必要です。この修飾子を指定しないと、自分自身のスキーマ内のオブジェクトを参照するものとみなされます。

  スキーマ・オブジェクトのみが *schema* で修飾できます。スキーマ・オブジェクトについては、2-100 ページの規則 7 を参照してください。規則 7 に示す非スキーマ・オブジェクトはスキーマ・オブジェクトではないため、*schema* では修飾できません。ただし、パブリック・シノニムは例外で、「`PUBLIC`」で修飾できます。この場合、引用符が必要です。
- *part* は、オブジェクトの部分です。この識別子によって、スキーマ・オブジェクトの部分（たとえば、表の列またはパーティション）を参照できます。なお、すべてのタイプのオブジェクトが部分を持っているとはかぎりません。

- *dblink* は、Oracle Database の分散オプションを使用している場合にのみ適用されます。オブジェクトを含むデータベースの名前です。この修飾子 *dblink* を指定することによって、ローカル・データベース以外のデータベース内のオブジェクトを参照できます。この *dblink* を指定しないと、自分自身のローカル・データベース内のオブジェクトを参照するものとみなされます。なお、すべての SQL 文でリモート・データベースのオブジェクトにアクセスできるとはかぎりません。

オブジェクトを参照する際のコンポーネントを区切っているピリオドの前後には、空白を入れることができます。ただし、通常は入れません。

Oracle Database によるスキーマ・オブジェクト参照の変換方法

SQL 文内のオブジェクトが参照される場合、Oracle はその SQL 文のコンテキストを検討して、該当するネームスペース内でそのオブジェクトの位置を確認します。そのオブジェクトの位置を確認してから、そのオブジェクトに対して文が指定する操作を実行します。指定した名前のオブジェクトが適切なネームスペース内に存在しない場合、Oracle はエラーを戻します。

次の例で、Oracle が SQL 文内のオブジェクト参照を変換する方法について説明します。名前 `departments` で識別される表にデータ行を追加する次の文を考えます。

```
INSERT INTO departments VALUES (
   280, 'ENTERTAINMENT_CLERK', 206, 1700);
```

文のコンテキストに基づいて、Oracle は、`departments` が次のようなオブジェクトであると判断します。

- 自分のスキーマ内の表
- 自分のスキーマ内のビュー
- 表またはビューに対するプライベート・シノニム
- パブリック・シノニム

Oracle は、文を発行したユーザーのスキーマ外のネームスペースを考慮する前に、そのユーザーのスキーマ内のネームスペースからオブジェクト参照を変換しようとします。この例では、Oracle は次の方法で名前 `departments` を変換しようとします。

1. Oracle は、最初に、表、ビューおよびプライベート・シノニムを含む文を発行したユーザーのスキーマ内のネームスペースで、オブジェクトの位置を確認しようとします。オブジェクトがプライベート・シノニムの場合、Oracle はそのシノニムが表すオブジェクトの位置を確認します。このオブジェクトは、ユーザー自身のスキーマ、他のスキーマ、または他のデータベースにあることもあります。このオブジェクトが別のシノニムである場合もあります。その場合、Oracle はそのシノニムが表すオブジェクトの位置を確認します。
2. オブジェクトがネームスペース内に存在する場合、Oracle はそのオブジェクトに対して文を実行しようとします。この例では、Oracle はデータ行を `departments` に追加しようとします。オブジェクトがその処理にとって正しい型でない場合、Oracle はエラーを戻します。この場合、`departments` は、表またはビュー、あるいは表またはビューとなるプライベート・シノニムである必要があります。`departments` が順序である場合、Oracle はエラーを戻します。
3. 前述の処理で検索されたネームスペースにオブジェクトが存在しない場合、Oracle はパブリック・シノニムを含むネームスペースを検索します。オブジェクトがそのネームスペースに存在する場合、Oracle はそのオブジェクトに対して文を実行しようとします。オブジェクトがその処理にとって正しい型でない場合、Oracle はエラーを戻します。この例では、`departments` が順序のパブリック・シノニムである場合、Oracle はエラーを戻します。

パブリック・シノニムに、依存表またはユーザー定義型がある場合は、依存オブジェクトと同じスキーマに、シノニムと同じ名前でオブジェクトを作成することはできません。

シノニムに、依存表またはユーザー定義型がない場合は、依存オブジェクトと同じスキーマに、依存オブジェクトと同じ名前で、オブジェクトを作成できます。すべての依存オブジェクトが無効になり、次のアクセス時に妥当性チェックが再実行されます。

参照： PL/SQL の識別子名の解決については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

他のスキーマ内のオブジェクトの参照

自分が所有するスキーマ以外のスキーマ内のオブジェクトを参照するには、次のように、オブジェクト名の前にスキーマ名を付けます。

```
schema.object
```

たとえば、次の文は、サンプル・スキーマ hr 内の employees 表を削除します。

```
DROP TABLE hr.employees;
```

リモート・データベース内のオブジェクトの参照

ローカル・データベース以外のデータベース内のオブジェクトを参照するには、オブジェクト名の後に、そのデータベースへのデータベース・リンクの名前を続けます。データベース・リンクはスキーマ・オブジェクトであり、これによって Oracle がリモート・データベースに接続され、そこにあるオブジェクトにアクセスします。この項では、次の項目について説明します。

- データベース・リンクを作成する方法
- SQL 文でデータベース・リンクを使用する方法

データベース・リンクの作成

14-28 ページの「CREATE DATABASE LINK」を使用して、データベース・リンクを作成します。この文では、データベース・リンクに関する次の情報を指定できます。

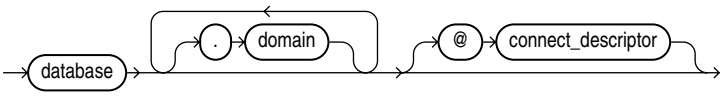
- データベース・リンク名
- リモート・データベースにアクセスするためのデータベース接続文字列
- リモート・データベースに接続するためのユーザー名およびパスワード

これらの情報はデータ・ディクショナリに格納されます。

データベース・リンク名 データベース・リンクを作成するとき、データベース・リンク名を指定する必要があります。データベース・リンク名は、他のオブジェクト型の名前とは異なります。データベース・リンク名は 128 バイト以内の長さで指定し、ピリオド（.）とアットマーク（@）を使用できます。

データベース・リンクに付ける名前は、データベース・リンクが参照するデータベースの名前、およびデータベース名の階層内のそのデータベースの位置に一致している必要があります。次に、データベース・リンク名の書式を示します。

***dblink*::=**



それぞれの意味は、次のとおりです。

- *database* には、データベース・リンクが接続するリモート・データベースのグローバル名の名前の部分を指定します。このグローバル名は、リモート・データベースのデータ・ディクショナリに格納されます。この名前は、GLOBAL_NAME データ・ディクショナリ・ビューで確認できます。
- *domain* には、データベース・リンクが接続するリモート・データベースのグローバル名のドメイン部分を指定します。データベース・リンクの名前に *domain* を指定しないと、Oracle は、現在、データ・ディクショナリに存在しているローカル・データベースのドメインに、データベース・リンク名を付加します。

- *connect_descriptor* によって、データベース・リンクをさらに修飾できます。接続修飾子を使用する場合、同じデータベースに複数のデータベース・リンクを作成できます。たとえば、接続修飾子を使用して、同じデータベースにアクセスする Oracle Real Application Clusters の異なるインスタンスに、複数のデータベース・リンクを作成できます。

database.domain の組合せは、**サービス名**と呼ばれることもあります。

> **参照：**『Oracle Database Net Services 管理者ガイド』

ユーザー名およびパスワード リモート・データベースに接続するために、ユーザー名およびパスワードを使用します。データベース・リンクでは、ユーザー名およびパスワードはオプションです。

データベース接続文字列 データベース接続文字列は、Oracle Net がリモート・データベースにアクセスするために使用する仕様です。データベース接続文字列の記述方法については、使用しているネットワーク・プロトコル用の Oracle Net のドキュメントを参照してください。データベース・リンク用のデータベース文字列はオプションです。

データベース・リンクの参照

データベース・リンクは、分散オプションを指定して Oracle を使用している場合にのみ利用できます。データベース・リンクを含む SQL 文の発行時に、次のいずれかの方法でデータベース・リンク名を指定します。

- データ・ディクショナリ内に格納される、*database*、*domain*、およびオプションの *connect_descriptor* コンポーネントを含む完全なデータベース・リンク名を指定します。
- *database* およびオプションの *connect_descriptor* コンポーネントを含むが、*domain* コンポーネントを含まない、部分的なデータベース・リンク名を指定します。

Oracle は、リモート・データベースに接続する前に次のタスクを実行します。

1. 文中に指定されているデータベース・リンク名が部分指定の場合、Oracle は、データ・ディクショナリ内に格納されているグローバル・データベース名に見られるとおり、そのリンク名にローカル・データベースのドメイン名を付加します。現在のグローバル・データベース名は、GLOBAL_NAME データ・ディクショナリ・ビューで見ることができます。
2. Oracle は、最初に、文を発行したユーザーのスキーマ内で、文の中のデータベース・リンクと同じ名前を持つプライベート・データベース・リンクを検索します。必要に応じて、同じ名前を持つパブリック・データベース・リンクを検索します。
   - Oracle は、必ず最初に一致したデータベース・リンク（プライベートまたはパブリック）のユーザー名およびパスワードを採用します。最初に一致したデータベース・リンクに対応付けられているユーザー名およびパスワードがあると、Oracle はそれを使用します。対応付けられているユーザー名およびパスワードがない場合、Oracle は、現在のユーザー名およびパスワードを使用します。
   - 最初に一致したデータベース・リンクに対応付けられているデータベース文字列が存在する場合、Oracle は、そのデータベース文字列を使用します。データベース文字列がない場合、Oracle は一致する次の（パブリック）データベース・リンクを検索します。一致するデータベース・リンクが存在しない場合、または一致するリンクに対応付けられているデータベース文字列が存在しない場合、Oracle はエラーを戻します。
3. Oracle は、リモート・データベースにアクセスするためにデータベース文字列を使用します。リモート・データベースにアクセスした後で、GLOBAL_NAMES パラメータの値が true の場合は、Oracle は、データベース・リンク名の *database.domain* 部分がリモート・データベースの完全なグローバル名に一致しているかどうかを確認します。この条件が満たされている場合、Oracle は手順 2 で選択したユーザー名とパスワードを使用して接続を続行します。それ以外の場合、Oracle はエラーを戻します。

4. データベース文字列、ユーザー名およびパスワードを使用した接続が成功した場合、Oracle は、リモート・データベース上の指定されたオブジェクトにアクセスしようとします。このとき、この項の前半で説明した、オブジェクト参照を変換するための規則、および他のスキーマ内のオブジェクトを参照するための規則が使用されます。

リモート・データベースの完全なグローバル名が、データベース・リンクの *database.domain* 部分と一致する必要があるという要件を無効にするには、初期化パラメータ GLOBAL_NAMES か、ALTER SYSTEM または ALTER SESSION 文の GLOBAL_NAMES パラメータに FALSE を設定します。

> **参照：** リモート・データベースの名前の変換の詳細は、『Oracle Database 管理者ガイド』を参照してください。

パーティション表と索引の参照

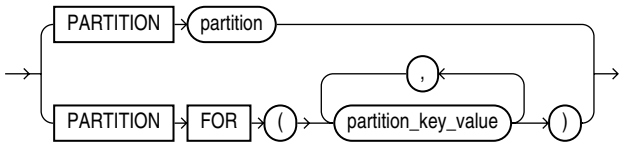
表および索引はパーティション化できます。パーティション化されたスキーマ・オブジェクトは、**パーティション**と呼ばれる多数の部分で構成され、各パーティションのすべての論理属性は同じです。たとえば、表のパーティションはすべて同じ列定義と制約定義を共有し、索引のパーティションはすべて同じ索引列を共有します。

拡張パーティションおよび拡張サブパーティション名を使用した場合、1 つのパーティションまたはサブパーティションのみ、パーティション・レベルおよびサブパーティション・レベルの操作（あるパーティションまたはサブパーティションからのすべての行の削除など）ができます。拡張された名前がない場合、そのような操作には述語（WHERE 句）を指定する必要があります。レンジおよびリスト・パーティション表では、パーティション・レベル操作を述語で表そうとすると（特にレンジ・パーティション・キーで複数の列を使用しているときは）、非常に複雑になる可能性があります。ハッシュ・パーティションおよびサブパーティションの場合、述語の使用はより難しくなります。これは、これらのパーティションおよびサブパーティションが、システムが定義するハッシュ・ファンクションに基づいているためです。

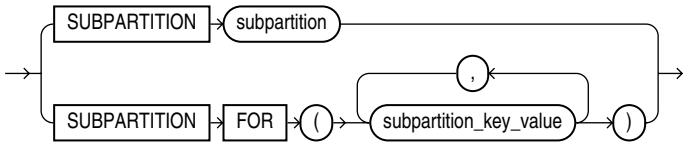
拡張パーティション名を使用した場合、パーティションを表のように使用できます。この方法のメリットは、これらのビューに対する権限を他のユーザーやロールに付与する（または取り消す）ことによって、パーティション・レベルのアクセス制御機構を構築できることです。このメリットは、レンジ・パーティション表に最も有効です。パーティションを表として使用するには、単一のパーティションからデータを選択してビューを作成し、そのビューを表として使用します。

構文 構文に *partition_extended_name* 要素または *subpartition_extended_name* 要素が指定されている SQL 文には、拡張パーティション表名および拡張サブパーティション表名を指定できます。

***partition_extended_name*::=**

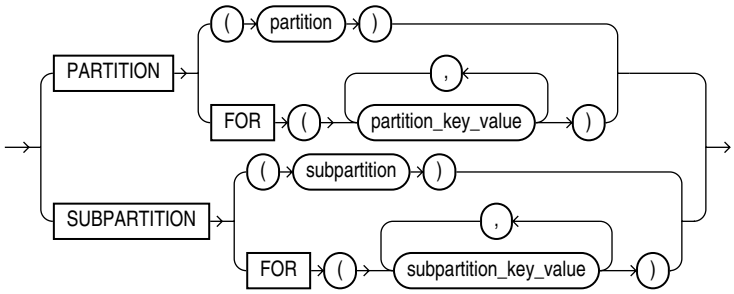


***subpartition_extended_name*::=**



DML 文 INSERT、UPDATE、DELETE、および ANALYZE 文では、パーティション名またはサブパーティション名をカッコで囲む必要があります。細かな違いですが、*partition_extension_clause* に影響します。

***partition_extension_clause*::=**



partition_extended_name、*subpartition_extended_name* および *partition_extension_clause* では、PARTITION FOR 句および SUBPARTITION FOR 句を使用すると、名前がなくてもパーティションを参照できます。これらの句は、すべてのタイプのパーティション化に有効であり、特に時間隔パーティションで役立ちます。データが表に挿入されると、必要に応じて時間隔パーティションが自動的に作成されます。

それぞれの *partition_value* には、パーティション化キー列ごとに 1 つの値を指定します。複数列のパーティション化キーには、パーティション化キーごとに 1 つの値を指定します。コンポジット・パーティションには、パーティション化キーごとに 1 つの値を指定し、続いてサブパーティション化キーごとに 1 つの値を指定します。すべてのパーティション化キーの値は、カンマで区切られます。時間隔パーティションには、1 つの *partition_value* のみを指定できます。この値は、有効な NUMBER または日時値にする必要があります。SQL 文は、指定した値を含むパーティションまたはサブパーティションで動作します。

> **参照：** 時間隔パーティションの詳細は、16-45 ページの「CREATE TABLE」の「INTERVAL 句」を参照してください。

拡張表名の制限事項： 現在、拡張パーティション表名および拡張サブパーティション表名を使用するときには、次の制限があります。

- リモート表は使用できません。拡張パーティション表名または拡張サブパーティション表名には、データベース・リンク（dblink）、または dblink を使用して表に変換するシノニムを含めることはできません。リモート・パーティションおよびリモート・サブパーティションを使用するには、拡張表名の構文を使用してリモート・サイトにビューを作成し、そのリモート・ビューを参照します。
- シノニムは使用できません。拡張パーティションまたは拡張サブパーティションは、実表を使用して指定する必要があります。シノニム、ビューまたはその他のオブジェクトは使用できません。
- PARTITION FOR 句および SUBPARTITION FOR 句は、ビューでの DDL 操作に対しては無効になります。

例　次の文の sales は、パーティション sales_q1_2000 を持つパーティション表です。単一パーティション sales_q1_2000 のビューを作成でき、それを表のように使用できます。この例では、パーティションから行が削除されます。

```
CREATE VIEW Q1_2000_sales AS
   SELECT * FROM sales PARTITION (SALES_Q1_2000);

DELETE FROM Q1_2000_sales WHERE amount_sold < 0;
```

オブジェクト型の属性とメソッドの参照

SQL 文のオブジェクト型の属性とメソッドを参照するには、参照を表の別名で完全に修飾する必要があります。次の例では、cust_address_typ 型、および cust_address_typ に基づく cust_address 列を持つ表 customers を含むサンプル・スキーマ oe について考えます。

```
CREATE TYPE cust_address_typ
  OID '82A4AF6A4CD1656DE034080020E0EE3D'
  AS OBJECT
    ( street_address     VARCHAR2(40)
    , postal_code        VARCHAR2(10)
    , city               VARCHAR2(30)
    , state_province     VARCHAR2(10)
    , country_id         CHAR(2)
    );
/
CREATE TABLE customers
    ( customer_id        NUMBER(6)
    , cust_first_name    VARCHAR2(20) CONSTRAINT cust_fname_nn NOT NULL
    , cust_last_name     VARCHAR2(20) CONSTRAINT cust_lname_nn NOT NULL
    , cust_address       cust_address_typ
.
.
.
```

次に示すとおり、SQL 文では、postal_code 属性への参照は表別名を使用して完全に修飾する必要があります。

```
SELECT c.cust_address.postal_code FROM customers c;

UPDATE customers c SET c.cust_address.postal_code = 'GU13 BE5'
   WHERE c.cust_address.city = 'Fleet';
```

引数を取らないメンバー・メソッドを参照する場合は、空のカッコを付ける必要があります。たとえば、サンプル・スキーマ oe には、メンバー・ファンクション getCatalogName を含む catalog_typ に基づくオブジェクト表 categories_tab が含まれます。SQL 文でこのメソッドをコールするには、次の例のように、空のカッコを付ける必要があります。

```
SELECT TREAT(VALUE(c) AS catalog_typ).getCatalogName() "Catalog Type"
   FROM categories_tab c
   WHERE category_id = 90;

Catalog Type
-----------------------------------
online catalog
```

3

疑似列

疑似列は表の列のように使用できますが、実際に表に格納されているわけではありません。疑似列から値を選択できますが、疑似列に対して値の挿入、更新、削除はできません。疑似列は、引数を指定しないファンクションとも似ています（詳細は第 5 章「ファンクション」を参照してください）。ただし、引数を指定しないファンクションは、通常、結果セット内の各行に対して同じ値を戻しますが、疑似列では各行に対して異なる値を戻します。

この章では、次の内容を説明します。

- 階層問合せ疑似列
- 順序疑似列
- バージョン問合せ疑似列
- COLUMN_VALUE 疑似列
- OBJECT_ID 疑似列
- OBJECT_VALUE 疑似列
- ORA_ROWSCN 疑似列
- ROWID 疑似列
- ROWNUM 疑似列
- XMLDATA 疑似列

階層問合せ疑似列

階層問合せ疑似列は、階層問合せでのみ有効です。階層問合せ疑似列には、次のものがあります。

- CONNECT_BY_ISCYCLE 疑似列
- CONNECT_BY_ISLEAF 疑似列
- LEVEL 疑似列

問合せの中で階層型の関連を定義するには、CONNECT BY 句を使用する必要があります。

CONNECT_BY_ISCYCLE 疑似列

CONNECT_BY_ISCYCLE 疑似列は、現在の行に自身の祖先でもある子がある場合に 1 を戻します。それ以外の場合は、0（ゼロ）を戻します。

CONNECT BY 句の NOCYCLE パラメータを指定した場合のみ、CONNECT_BY_ISCYCLE を指定できます。NOCYCLE によって、Oracle は問合せの結果を戻すことができます。このパラメータを指定しないと、データ内の CONNECT BY ループのため、問合せは失敗します。

> **参照：** NOCYCLE パラメータの詳細は、9-3 ページの「階層問合せ」を参照してください。CONNECT_BY_ISCYCLE 疑似列を使用する例については、9-5 ページの「階層問合せの例」を参照してください。

CONNECT_BY_ISLEAF 疑似列

CONNECT_BY_ISLEAF 疑似列は、現在の行が CONNECT BY 条件によって定義されるツリーのリーフである場合に 1 を戻します。それ以外の場合は、0（ゼロ）を戻します。この情報は、特定の行をさらに展開して階層の詳細を表示できるかどうかを示します。

CONNECT_BY_ISLEAF の例　次の例は、hr.employees 表の最初の 3 レベルです。各行がリーフ行か（IsLeaf 列が 1）、子である行を持つか（IsLeaf 列が 0）を示しています。

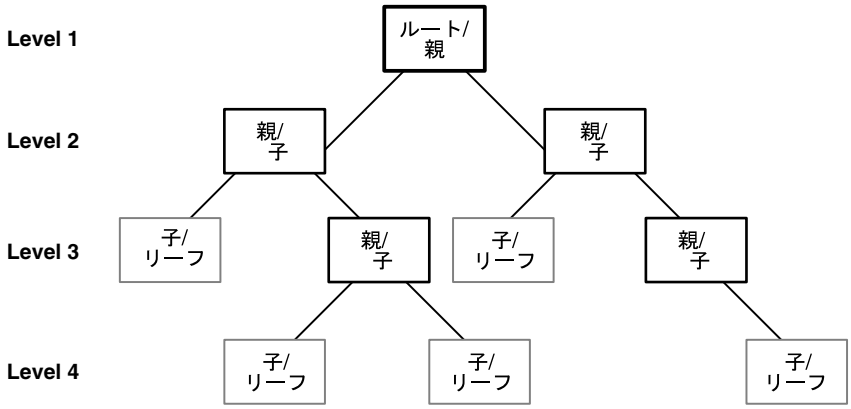
```
SELECT last_name "Employee", CONNECT_BY_ISLEAF "IsLeaf",
   LEVEL, SYS_CONNECT_BY_PATH(last_name, '/') "Path"
   FROM employees
   WHERE LEVEL <= 3 AND department_id = 80
   START WITH employee_id = 100
   CONNECT BY PRIOR employee_id = manager_id AND LEVEL <= 4;

Employee                       IsLeaf      LEVEL Path
------------------------- ---------- ---------- -------------------------
Abel                                1          3 /King/Zlotkey/Abel
Ande                                1          3 /King/Errazuriz/Ande
Banda                               1          3 /King/Errazuriz/Banda
Bates                               1          3 /King/Cambrault/Bates
Bernstein                           1          3 /King/Russell/Bernstein
Bloom                               1          3 /King/Cambrault/Bloom
Cambrault                           0          2 /King/Cambrault
Cambrault                           1          3 /King/Russell/Cambrault
Doran                               1          3 /King/Partners/Doran
Errazuriz                           0          2 /King/Errazuriz
Fox                                 1          3 /King/Cambrault/Fox
. . .
```

> **参照：** 9-3 ページの「階層問合せ」および 5-184 ページの「SYS_CONNECT_BY_PATH」を参照してください。

LEVEL 疑似列

階層問合せによって戻される各行について、LEVEL 疑似列は、ルート行に 1 を戻し、ルートの子には 2 を戻します（以降同様に続きます）。**ルート行**は逆ツリー構造の最上位行です。**子である行**は任意の非ルート行です。**親である行**は子を持つ任意の行です。**リーフ行**は子を持たない任意の行です。図 3-1 に、逆ツリーのノードとそれらの LEVEL 値を示します。

図 3-1 階層ツリー



参照： 階層問合せの概要については、9-3 ページの「階層問合せ」を参照してください。LEVEL 疑似列の使用に関する制限事項については、7-22 ページの「IN 条件」を参照してください。

順序疑似列

順序は、一意の連続値を生成できるスキーマ・オブジェクトです。これらの値は、主キーや一意のキーによく使用されます。次の疑似列を使用した SQL 文で、順序値を参照できます。

- CURRVAL: 順序の現在の値を戻します。
- NEXTVAL: 順序を増加させて次の値を戻します。

CURRVAL と NEXTVAL は、順序の名前で修飾する必要があります。

```
sequence.CURRVAL
sequence.NEXTVAL
```

別のユーザーのスキーマ内での順序の現在の値または次の値を参照するには、その順序に対する SELECT オブジェクト権限または SELECT ANY SEQUENCE システム権限のどちらかが必要です。さらに、その順序は、次に示すとおり、順序を含むスキーマで修飾する必要があります。

```
schema.sequence.CURRVAL
schema.sequence.NEXTVAL
```

リモート・データベース上の順序の値を参照するには、次のように、データベース・リンクの完全な名前または部分的な名前で順序を修飾する必要があります。

```
schema.sequence.CURRVAL@dblink
schema.sequence.NEXTVAL@dblink
```

順序には、待機またはロックすることなく多数のユーザーが同時にアクセスできます。

参照： データベース・リンクの参照方法の詳細は、2-104 ページの「リモート・データベース内のオブジェクトの参照」を参照してください。

順序値の使用場所

次の場所で CURRVAL と NEXTVAL を使用できます。

- 副問合せ、マテリアライズド・ビューまたはビューに含まれていない SELECT 文の SELECT 構文のリスト
- INSERT 文内の副問合せの SELECT 構文のリスト
- INSERT 文の VALUES 句
- UPDATE 文の SET 句

順序値の制限事項： 次の構造体では、CURRVAL および NEXTVAL は使用できません。

- DELETE 文、SELECT 文または UPDATE 文内の副問合せ
- ビューの問合せ、またはマテリアライズド・ビューの問合せ
- DISTINCT 演算子を持つ SELECT 文
- GROUP BY 句または ORDER BY 句を持つ SELECT 文
- 集合演算子 UNION、INTERSECT または MINUS によって別の SELECT 文と結合されている SELECT 文
- SELECT 文の WHERE 句
- CREATE TABLE 文または ALTER TABLE 文の列の DEFAULT 値
- CHECK 制約の条件

CURRVAL または NEXTVAL を使用する単一の SQL 文では、参照された LONG 列、更新された表、ロックされた表がすべて同じデータベース上にある必要があります。

順序値の使用方法

順序を作成するときに、初期値と増分値を定義できます。NEXTVAL の最初の参照によって、順序の初期値が戻されます。その後の参照によって、定義された NEXTVAL 増分値で順序が増加され、その新しい値が戻されます。CURRVAL を参照すると、NEXTVAL への最後の参照で戻された値である、順序の現在の値が常に戻されます。

セッションの順序に対して CURRVAL を使用する前に、まず NEXTVAL で順序を初期化してください。順序の詳細は、15-64 ページの「CREATE SEQUENCE」を参照してください。

NEXTVAL への参照が含まれる単一の SQL 文の中では、Oracle は、次の各行につき 1 回順序を増加させます。

- SELECT 文の外部問合せブロックによって戻される行。このような問合せブロックは、次の場所に指定できます。
  - トップレベルの SELECT 文
  - INSERT ... SELECT 文（単一表または複数表）。マルチテーブル・インサートの場合、NEXTVAL への参照が VALUES 句内に存在する必要があり、マルチテーブル・インサートの複数のブランチで NEXTVAL が参照される場合でも、副問合せによって戻される行ごとに、順序が 1 回更新されます。
  - CREATE TABLE ...AS SELECT 文
  - CREATE MATERIALIZED VIEW ...AS SELECT 文
- UPDATE 文で更新される行
- VALUES 句が含まれる INSERT 文

- INSERT ... [ALL | FIRST] 文（マルチテーブル・インサート）。マルチテーブル・インサートは、単一の SQL 文とみなされます。このため、順序の NEXTVAL への参照では、文の SELECT 部分からの各入力レコードに対して順序が 1 回のみ増加されます。INSERT ... [ALL | FIRST] 文のいずれかの部分で NEXTVAL が複数回指定されている場合は、指定されたレコードが挿入される回数に関係なく、値はすべての挿入ブランチに対して同じになります。
- MERGE 文でマージされる行。NEXTVAL への参照は、*merge_insert_clause* または *merge_update_clause* あるいはその両方に指定できます。NEXTVALUE 値は、更新操作または挿入操作に順序番号が使用されない場合でも、行が更新されるか挿入されるたびに増加されます。NEXTVAL がこれらの場所のいずれかで複数回指定されている場合、順序は各行に対して 1 回増加され、その行の NEXTVAL が検出されるたびにすべて同じ値を戻します。
- マルチテーブル INSERT ALL 文の入力行。NEXTVAL は、各行に対する *insert_into_clause* マップの回数に関係なく、副問合せによって戻される各行に対して 1 回のみ増加されます。

これらの場所のいずれかが、NEXTVAL を複数回参照している場合、Oracle は 1 回のみ順序を増加させ、NEXTVAL が検出されるたびにすべて同じ値を戻します。

これらの場所のいずれかが、CURRVAL と NEXTVAL の両方を参照している場合、Oracle は CURRVAL と NEXTVAL の両方について順序を増加させ同じ値を戻します。

順序の次の値の検索例　次の例では、サンプル・スキーマ hr の従業員順序の次の値を検索します。

```
SELECT employees_seq.nextval
    FROM DUAL;
```

表への順序値の挿入例　次の例では、従業員順序を増加させ、サンプル表 hr.employees に挿入される新しい従業員のためにその値を使用します。

```
INSERT INTO employees
   VALUES (employees_seq.nextval, 'John', 'Doe', 'jdoe',
   '555-1212', TO_DATE(SYSDATE), 'PU_CLERK', 2500, null, null,
   30);
```

順序の現在の値の再利用例　次の例では、次の注文番号を使用して新しい注文をマスター注文表に追加します。その後、この番号を使用して関連する注文をディテール注文表に追加します。

```
INSERT INTO orders (order_id, order_date, customer_id)
   VALUES (orders_seq.nextval, TO_DATE(SYSDATE), 106);

INSERT INTO order_items (order_id, line_item_id, product_id)
   VALUES (orders_seq.currval, 1, 2359);

INSERT INTO order_items (order_id, line_item_id, product_id)
   VALUES (orders_seq.currval, 2, 3290);

INSERT INTO order_items (order_id, line_item_id, product_id)
   VALUES (orders_seq.currval, 3, 2381);
```

バージョン問合せ疑似列

バージョン問合せ疑似列は、Oracle フラッシュバック問合せの一形態である Oracle Flashback Version Query でのみ有効です。バージョン問合せ疑似列には、次のものがあります。

- VERSIONS_STARTTIME: 問合せによって戻された行の最初のバージョンのタイムスタンプを戻します。
- VERSIONS_STARTSCN: 問合せによって戻された行の最初のバージョンの SCN を戻します。
- VERSIONS_ENDTIME: 問合せによって戻された行の最後のバージョンのタイムスタンプを戻します。
- VERSIONS_ENDSCN: 問合せによって戻された行の最後のバージョンの SCN を戻します。
- VERSIONS_XID: 各行のそれぞれのバージョンについて、その行バージョンを作成したトランザクションのトランザクション ID（RAW 番号）を戻します。
- VERSIONS_OPERATION: 各行のそれぞれのバージョンについて、その行バージョンを発生させた操作を表す単一の文字を戻します。戻される値は、I（挿入操作）、U（更新操作）または D（削除操作）です。

> **参照：** バージョン問合せの詳細は、19-15 ページの「flashback_query_clause」を参照してください。

COLUMN_VALUE 疑似列

COLUMNS 句を指定せずに XMLTable 構造体を参照する場合、または TABLE ファンクションを使用して表の型をネストしたスカラーを参照する場合、データベースは単一列を持つ仮想表を戻します。この疑似列の名前は、COLUMN_VALUE です。

XMLTable のコンテキストでは、戻り値のデータ型は XMLType です。たとえば、次の 2 つの文は同等で、どちらの出力も戻される列の名前として COLUMN_VALUE を示します。

```
SELECT * FROM XMLTABLE('<a>123</a>');

COLUMN_VALUE
----------------------------------------
<a>123</a>

SELECT COLUMN_VALUE FROM (XMLTable('<a>123</a>'));

COLUMN_VALUE
----------------------------------------
<a>123</a>
```

TABLE ファンクションのコンテキストでは、戻り値はコレクション要素のデータ型になります。次の文では、2 つのレベルにネストした表を作成します。このコンテキストでの COLUMN_VALUE の使用方法は、16-62 ページの「マルチレベル・コレクションの例 :」を参照してください。

```
CREATE TYPE phone AS TABLE OF NUMBER;
/
CREATE TYPE phone_list AS TABLE OF phone;
/
```

次の文では COLUMN_VALUE を使用して phone 型から選択します。

```
SELECT t.COLUMN_VALUE from table(phone(1,2,3)) t;

COLUMN_VALUE
------------
           1
           2
           3
```

ネストした型では、SELECT 構文のリストと TABLE ファンクションの両方で COLUMN_VALUE 疑似列を使用できます。

```
SELECT t.COLUMN_VALUE FROM
   TABLE(phone_list(phone(1,2,3))) p, TABLE(p.COLUMN_VALUE) t;
COLUMN_VALUE
------------
           1
           2
           3
```

次の例に示すように、キーワード COLUMN_VALUE は、列または属性名を持たない内部のネストした表のスカラー値に対して Oracle Database が生成する名前です。このコンテキストでは、COLUMN_VALUE は疑似列ではなく、実際の列の名前です。

```
CREATE TABLE my_customers (
   cust_id       NUMBER,
   name          VARCHAR2(25),
   phone_numbers phone_list,
   credit_limit  NUMBER)
   NESTED TABLE phone_numbers STORE AS outer_ntab
   (NESTED TABLE COLUMN_VALUE STORE AS inner_ntab);
```

参照 :

- このファンクションの詳細は、5-245 ページの「XMLTABLE」を参照してください。
- TABLE ファンクションの詳細は、18-56 ページの「table_collection_expression::=」を参照してください。
- ALTER TABLE の例は、12-80 ページの「ネストした表の例 :」を参照してください。

OBJECT_ID 疑似列

OBJECT_ID 疑似列は、オブジェクト表またはビューの列のオブジェクト識別子を戻します。Oracle はこの疑似列をオブジェクト表の主キーとして使用します。OBJECT_ID は、ビューの INSTEAD OF トリガーや、オブジェクト表の置換可能行の ID の識別に便利です。

注意 : 以前のリリースでは、この疑似列は SYS_NC_OID$ と呼ばれていました。下位互換性を保つため、この名称は引き続きサポートされます。ただし、より直感的な名前である OBJECT_ID を使用することをお薦めします。

参照 : この疑似列の使用例については、『Oracle Database オブジェクト・リレーショナル開発者ガイド』を参照してください。

OBJECT_VALUE 疑似列

OBJECT_VALUE 疑似列は、オブジェクト表、XMLType 表、オブジェクト・ビューまたは XMLType ビューの列のシステム生成名を戻します。この疑似列は、オブジェクト表の置換可能行の値の識別や、WITH OBJECT IDENTIFIER 句を使用したオブジェクト・ビューの作成に便利です。

注意 : 以前のリリースでは、この疑似列は SYS_NC_ROWINFO$ と呼ばれていました。下位互換性を保つため、この名称は引き続きサポートされます。ただし、より直感的な名前である OBJECT_VALUE を使用することをお薦めします。

参照：

- この疑似列の使用方法の詳細は、16-58 ページの「object_table」および 17-16 ページの「object_view_clause」を参照してください。
- この疑似列の使用例については、『Oracle Database オブジェクト・リレーショナル開発者ガイド』を参照してください。

ORA_ROWSCN 疑似列

ORA_ROWSCN は、各行について、その行に対する最新の変更のシステム変更番号（SCN）の上限（近似値）を戻します。この疑似列は、行が最後に更新されたおよその時期を判別するのに便利です。Oracle は行が存在するブロックをコミットしたトランザクションによって SCN を追跡するため、この値は正確でない場合があります。行レベル依存の追跡を行う表を作成すると、SCN のより正確な概数を取得できます。行レベル依存の追跡の詳細は、16-54 ページの「NOROWDEPENDENCIES | ROWDEPENDENCIES」を参照してください。

この疑似列は、ビューへの問合せでは使用できません。ただし、この疑似列を使用して、ビューの作成時に基礎となる表を参照することは可能です。また、UPDATE 文または DELETE 文の WHERE 句でこの疑似列を使用することもできます。

ORA_ROWSCN は、フラッシュバック問合せではサポートされません。かわりに、フラッシュバック問合せ専用に提供されているバージョン問合せ疑似列を使用してください。フラッシュバック問合せの詳細は、19-15 ページの「flashback_query_clause」を参照してください。バージョン問合せ疑似列の詳細は、3-6 ページの「バージョン問合せ疑似列」を参照してください。

ORA_ROWSCN の制限事項： この疑似列は、外部表ではサポートされません。

例　次の最初の文では、ORA_ROWSCN 疑似列を使用して、employees 表に対する最後の操作のシステム変更番号を取得します。2 番目の文では、この疑似列を SCN_TO_TIMESTAMP ファンクションとともに使用して、操作のタイムスタンプを判別します。

```
SELECT ORA_ROWSCN, last_name FROM employees WHERE employee_id = 188;

SELECT SCN_TO_TIMESTAMP(ORA_ROWSCN), last_name FROM employees
   WHERE employee_id = 188;
```

参照：　「SCN_TO_TIMESTAMP」（5-162 ページ）

ROWID 疑似列

ROWID 疑似列は、データベース内の各行について、行のアドレスを戻します。Oracle Database の ROWID 値には、行を検索するために必要な次の情報が含まれています。

- オブジェクトのデータ・オブジェクト番号。
- 行が存在するデータ・ファイルのデータ・ブロック。
- データ・ブロック内での行の位置（最初の行は 0）。
- 行が存在するデータ・ファイル（最初のファイルは 1）。ファイル番号は表領域に対して相対的です。

ほとんどの場合、ROWID 値ではデータベース内の行は一意に識別されます。ただし、同じクラスタに格納されている異なる表の行は、同じ ROWID を持つことができます。

ROWID 疑似列の値は ROWID または UROWID データ型を持ちます。詳細は、2-26 ページの「ROWID データ型」および 2-27 ページの「UROWID データ型」を参照してください。

ROWID 値には、次の重要な用途があります。

- 単一の行に最も速くアクセスする方法です。
- 表の行が格納されている様子を示すことができます。
- 表の行に対する一意の識別子です。

表の主キーとして ROWID を使用しないでください。たとえば、インポート・ユーティリティとエクスポート・ユーティリティで行を削除してから再挿入する場合、ROWID が変わる場合があります。行を削除した場合、Oracle は、後から新しく挿入される行にその ROWID を再度割り当てる可能性があります。

問合せの SELECT 句と WHERE 句で ROWID 疑似列を使用できますが、これらの疑似列の値が実際にデータベースに格納されるわけではありません。ROWID 疑似列の値に対して挿入、更新および削除はできません。

例　次の文は、部門 20 の従業員のデータを含むすべての行のアドレスを選択します。

```
SELECT ROWID, last_name
   FROM employees
   WHERE department_id = 20;
```

ROWNUM 疑似列

> **注意：** ROW_NUMBER 組込み SQL ファンクションは、問合せの結果の順序付けを強力にサポートします。詳細は、5-158 ページの「ROW_NUMBER」を参照してください。

ROWNUM 疑似列は、問合せによって戻される各行について、表や結合処理された行の集合から Oracle が行を選択する順序を示す番号を戻します。つまり、選択される最初の行の ROWNUM は 1、2 番目の行の ROWNUM は 2 です（以降同様に続きます）。

次の例のように、ROWNUM を使用して問合せによって戻される行数を制限できます。

```
SELECT * FROM employees WHERE ROWNUM < 11;
```

同じ問合せで ROWNUM に ORDER BY 句が続く場合、ORDER BY 句によって行が再び順序付けられます。結果は、行がアクセスされる方法によって異なります。たとえば、ORDER BY 句の指定によって Oracle が索引を使用してデータにアクセスする場合、索引なしの場合とは異なる順序で行が取り出されることがあります。このため、次の文では、前述の例と同じ行が戻されるとはかぎりません。

```
SELECT * FROM employees WHERE ROWNUM < 11 ORDER BY last_name;
```

ORDER BY 句を副問合せに埋め込んで ROWNUM 条件をトップレベル問合せに置いた場合、行の順序付けの後で ROWNUM 条件を強制的に適用させることができます。たとえば、次の問合せは、小さい順に 10 個の従業員番号を持つ従業員を戻します。これは、**上位 N 番のレポート**と呼ばれることがあります。

```
SELECT * FROM
   (SELECT * FROM employees ORDER BY employee_id)
   WHERE ROWNUM < 11;
```

前述の例では、ROWNUM 値はトップレベルの SELECT 文の値です。これらの値は、副問合せ内の employee_id によって行が順序付けられた後で生成されます。

比較条件「ROWNUM 値＞正の整数」は、常に偽となるため注意してください。たとえば、次の問合せでは行は戻されません。

```
SELECT * FROM employees
    WHERE ROWNUM > 1;
```

最初にフェッチされる行の ROWNUM には 1 が割り当てられるため、条件は偽と判断されます。2 番目にフェッチされる予定だった行は最初の行になるため、この ROWNUM にも 1 が割り当てられ、条件も偽と判断されます。このように、後続するすべての行が条件を満たさないため、行は戻されません。

また、次の例のように、ROWNUM を使用して表の各行に一意の値を割り当てることもできます。

```
UPDATE my_table
    SET column1 = ROWNUM;
```

行に一意の番号を割り当てる別の方法については、5-158 ページの「ROW_NUMBER」を参照してください。

注意： 問合せで ROWNUM を使用した場合、ビューの最適化に影響することがあります。詳細は、『Oracle Database 概要』を参照してください。

XMLDATA 疑似列

Oracle は、XMLSchema 情報および STORAGE 句の指定方法に基づいて、XMLType データを LOB またはオブジェクト・リレーショナル列に格納します。XMLDATA 疑似列を使用すると、基礎となる LOB またはオブジェクト・リレーショナル列にアクセスし、追加の STORAGE 句のパラメータ、制約、索引などを指定できます。

例 次の文は、この疑似列の使用方法を示しています。XMLType の簡単な表を作成するとします。

```
CREATE TABLE xml_lob_tab of XMLTYPE;
```

デフォルト記憶域は、CLOB 列にあります。基礎となる LOB 列の記憶特性を変更する場合は、次の文を使用できます。

```
ALTER TABLE xml_lob_tab MODIFY LOB (XMLDATA)
   (STORAGE (BUFFER_POOL DEFAULT) CACHE);
```

E-8 ページの「SQL 文での XML の使用方法」で作成した xwarehouses 表のような XML Schema ベースの表を作成したとします。その後、次の文に示すように、XMLDATA 列を使用して、基礎となる列のプロパティを設定できます。

```
ALTER TABLE xwarehouses ADD (UNIQUE(XMLDATA."WarehouseId"));
```

4

演算子

演算子は、データ項目を操作し、結果を戻すために使用します。構文では、演算子はオペランドの前後または 2 つのオペランドの間で使用します。

この章では、次の内容を説明します。

- SQL 演算子
- 算術演算子
- 連結演算子
- 階層問合せ演算子
- 集合演算子
- MULTISET 演算子
- ユーザー定義演算子

この章では、非論理（非ブール）演算子について説明します。これらの演算子は、問合せまたは副問合せ内で WHERE または HAVING 句の条件として、単独では使用できません。条件として使用できる論理演算子の詳細は、第 7 章「条件」を参照してください。

SQL 演算子

演算子は、**オペランド**または**引数**と呼ばれる個々のデータ項目を操作します。演算子は、特殊文字またはキーワードで表します。たとえば、乗算演算子は、アスタリスク（*）で表します。

Oracle Text がインストールされている場合は、Oracle Text 問合せで、この製品に含まれる SCORE 演算子を使用できます。また、CONTAINS、CATSEARCH、MATCHES などの組込み Text 演算子を使用して条件を作成することもできます。Oracle Text 要素の詳細は、『Oracle Text リファレンス』を参照してください。

Oracle Expression Filter を使用している場合、この製品に含まれる組込み EVALUATE 演算子を使用して条件を作成できます。詳細は、『Oracle Database ルール・マネージャおよび式フィルタ開発者ガイド』を参照してください。

注意： NLS_COMP と NLS_SORT の設定を組み合せた値によって、文字をソートおよび比較するルールが決まります。ご使用のデータベースの NLS_COMP に LINGUISTIC が設定されている場合、この章のエンティティはすべて NLS_SORT パラメータによって指定されるルールに従って解釈されます。NLS_COMP が LINGUISTIC に設定されていない場合、ファンクションは NLS_SORT の設定に関係なく解釈されます。NLS_SORT は、明示的に設定できます。明示的に設定されていない場合は、NLS_LANGUAGE から導出されます。これらの設定の詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

単項演算子およびバイナリ演算子

一般に、演算子には次の 2 つのクラスがあります。

- **単項**: 単項演算子は、1 つのみのオペランドについて操作します。単項演算子の書式は次のとおりです。

  ```
  operator operand
  ```

- **バイナリ**: バイナリ演算子は、2 つのオペランドについて操作します。バイナリ演算子の書式は次のとおりです。

  ```
  operand1 operator operand2
  ```

この他、特別な書式を持ち、3 つ以上のオペランドについて操作可能な演算子もあります。
演算子のオペランドに NULL が指定された場合、結果は常に NULL になります。この規則に従わない唯一の演算子が連結演算子（||）です。

演算子の優先順位

優先順位とは、同じ式の中の異なる演算子を Oracle Database が評価する順序を意味します。複数の演算子を含む式を評価するとき、Oracle は優先順位の高い演算子を評価した後で、優先順位の低い演算子を評価します。優先順位の等しい演算子は、式の中で左から右に評価されます。

表 4-1 に、SQL 演算子を優先順位の高い方から順に示します。同じ行にリストされている演算子の優先順位は同じです。

表 4-1 SQL 演算子の優先順位

演算子	操作
+、-（単項演算子）、PRIOR、CONNECT_BY_ROOT	同一、否定、階層内の位置
*、/	乗算、除算
+、-（バイナリ演算子）、||	加算、減算、連結
SQL 条件は、SQL 演算子の後で評価されます。	7-3 ページの「条件の優先順位」を参照してください。

優先順位の例　次の式では、乗算は加算よりも優先順位が高いため、2 と 3 を掛けた結果に 1 が加算されます。

```
1+2*3
```

式の中でカッコを使用して演算子の優先順位を上書きできます。Oracle は、カッコの内側の式を評価した後で、外側の式を評価します。

SQL では、集合演算子（UNION、UNION ALL、INTERSECT および MINUS）もサポートされます。集合演算子によって結合されるのは、問合せによって戻される行の集まりで、個々のデータ項目ではありません。集合演算子の優先順位はすべて同じです。

> **参照：** 階層問合せのみで使用する PRIOR 演算子については、4-5 ページの「階層問合せ演算子」および 9-3 ページの「階層問合せ」を参照してください。

算術演算子

1 つまたは 2 つの引数を持つ算術演算子を使用することによって、数値を否定（正負を反転）、加算、減算、乗算および除算できます。これらの演算子の中には、日時および時間隔の演算に使用されるものもあります。演算子の引数は、数値データ型、または数値データ型に暗黙的に変換可能な任意のデータ型に解決される必要があります。

単項算術演算子は、引数の数値データ型と同じデータ型を戻します。バイナリ算術演算子の場合、Oracle は、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。表 4-2 に、算術演算子を示します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。数値の優先順位の詳細は、2-15 ページの「数値の優先順位」を参照してください。また、2-20 ページの「日時および期間の演算」を参照してください。

表 4-2 算術演算子

演算子	用途	例
+ -	式の正負を示す場合、これらは単項演算子です。	`SELECT * FROM order_items WHERE quantity = -1 ORDER BY order_id, line_item_id, product_id;` `SELECT * FROM employees WHERE -salary < 0 ORDER BY employee_id;`
+ -	加算、減算を行う場合、これらはバイナリ演算子です。	`SELECT hire_date FROM employees WHERE SYSDATE - hire_date > 365 ORDER BY hire_date;`
* /	乗算、除算を行います。これらはバイナリ演算子です。	`UPDATE employees SET salary = salary * 1.1;`

二重否定や負の数の減算を表現する場合に、算術式で、連続した負の符号（--）は使用しないでください。文字 -- は、SQL 文ではコメントの開始を示す場合に使用します。連続した負の符号は、空白またはカッコで区切ってください。SQL 文中のコメントの詳細は、2-69 ページの「コメント」を参照してください。

連結演算子

連結演算子は、文字列および CLOB データを操作する場合に使用します。表 4-3 に、連結演算子を示します。

表 4-3 連結演算子

演算子	用途	例
||	文字列および CLOB データを連結します。	`SELECT 'Name is ' || last_name FROM employees ORDER BY last_name;`

2 つの文字列を連結した結果は別の文字列になります。両方の文字列が CHAR データ型の場合、結果は CHAR データ型の文字列になり、その最大文字数は 2000 です。どちらかの文字列が VARCHAR2 データ型の場合、結果は VARCHAR2 データ型の文字列になり、最大文字数は 4000 です。どちらかの引数が CLOB データ型の場合、結果は、一時 CLOB になります。データ型が文字列型か CLOB 型かにかかわらず、後続空白は連結後も文字列に残ります。

多くのプラットフォームでは、連結演算子は、表 4-3 に示すとおり 2 本の実線垂直バーで表されます。ただし、IBM 社のプラットフォームの中には、この演算子として破線垂直バーを使用するものもあります。異なるキャラクタ・セットを持つシステム間（たとえば ASCII と EBCDIC 間）で SQL スクリプト・ファイルを移動する場合、垂直バーが、移動先の Oracle Database 環境で必要な垂直バーに変換されない場合があります。オペレーティング・システムまたはネットワーク・ユーティリティによる変換の制御が困難または不可能である場合に備えて、Oracle では、垂直バー演算子にかわるものとして CONCAT 文字ファンクションが提供されています。異なるキャラクタ・セットを持つ環境間でアプリケーションを移動する場合は、この文字ファンクションを使用することをお薦めします。

Oracle は、長さが 0（ゼロ）の文字列を NULL として処理しますが、長さが 0（ゼロ）の文字列を別のオペランドと連結すると、その結果は常にもう一方のオペランドになります。結果が NULL になるのは、2 つの NULL 文字列を連結したときのみです。ただし、この処理は Oracle Database の今後のバージョンでも継続されるとはかぎりません。NULL になる可能性がある式を連結する場合は、NVL ファンクションを使用して、その式を長さが 0（ゼロ）の文字列に明示的に変換してください。

参照：

- CHAR データ型と VARCHAR2 データ型の違いの詳細は、2-9 ページの「文字データ型」を参照してください。
- 5-37 ページの CONCAT ファンクションおよび 5-115 ページの NVL ファンクションを参照してください。
- CLOB の詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』を参照してください。

連結の例　次の例では、CHAR 列および VARCHAR2 列を持つ表を作成し、後続空白のある値とない値を挿入してから、これらの値を選択し、連結します。なお、CHAR 列および VARCHAR2 列では、ともに後続空白が保存されます。

```
CREATE TABLE tab1 (col1 VARCHAR2(6), col2 CHAR(6),
        col3 VARCHAR2(6), col4 CHAR(6) );

INSERT INTO tab1 (col1,  col2,     col3,     col4)
        VALUES   ('abc', 'def   ', 'ghi   ', 'jkl');

SELECT col1||col2||col3||col4 "Concatenation"
        FROM tab1;

Concatenation
------------------------
abcdef   ghi   jkl
```

階層問合せ演算子

PRIOR および CONNECT_BY_ROOT の 2 つの演算子は、階層問合せでのみ有効です。

PRIOR

階層問合せでは、CONNECT BY *condition* 内の 1 つの式を PRIOR 演算子で修飾する必要があります。CONNECT BY *condition* が複合条件の場合、1 つの条件のみに PRIOR 演算子が必要です（複数の PRIOR 条件を使用することもできます）。PRIOR は、階層問合せ内でカレント行の親である行の直後にある式を評価します。

PRIOR は、等価演算子を使用して列の値を比較する場合によく使用されます（PRIOR キーワードは演算子のどちら側でもかまいません）。PRIOR を指定すると、列の親である行の値が使用されます。等号（=）以外の演算子は、理論上は CONNECT BY 句に指定できます。ただし、これらの他の演算子の組合せによっては、作成される条件は無限ループを発生させる場合があります。この場合、実行時にループが検出され、エラーが戻されます。この変数の詳細および例については、9-3 ページの「階層問合せ」を参照してください。

CONNECT_BY_ROOT

CONNECT_BY_ROOT は、階層問合せでのみ有効な単項演算子です。この演算子を使用して列を修飾すると、ルート行のデータを使用して列の値が戻されます。この演算子は、階層問合せの CONNECT BY [PRIOR] 条件の機能を拡張します。

CONNECT_BY_ROOT に関する制限事項　この演算子は、START WITH 条件または CONNECT BY 条件内で指定できません。

参照：「CONNECT_BY_ROOT の例」（9-6 ページ）

集合演算子

集合演算子は、2 つのコンポーネントの問合せ結果を 1 つの結果にまとめます。集合演算子を含む問合せを複合問合せと呼びます。表 4-4 に、SQL の集合演算子を示します。これらの演算子の例や制限事項などの詳細は、9-7 ページの「UNION [ALL]、INTERSECT および MINUS 演算子」を参照してください。

表 4-4 集合演算子

演算子	戻る結果
UNION	各問合せによって戻るすべての行（重複行は含まない）
UNION ALL	各問合せによって戻るすべての行（重複行を含む）
INTERSECT	両方の問合せによって戻るすべての行（重複行は含まない）
MINUS	最初の問合せによって戻る行で、2 番目の問合せでは戻されない行（重複行は含まない）

MULTISET 演算子

MULTISET 演算子は、2 つのネストした表の結果を 1 つのネストした表にまとめます。

MULTISET 演算子に関する例では、次のように 2 つのネストした表を作成し、データをロードする必要があります。

まず、customers_demo という名前で、oe.customers 表のコピーを作成します。

```
CREATE TABLE customers_demo AS
  SELECT * FROM customers;
```

次に、cust_address_tab_typ という表型を作成します。この型はネストした表の列を作成する際に使用します。

```
CREATE TYPE cust_address_tab_typ AS
  TABLE OF cust_address_typ
/
```

次に、customers_demo 表に、ネストした表の列を 2 つ作成します。

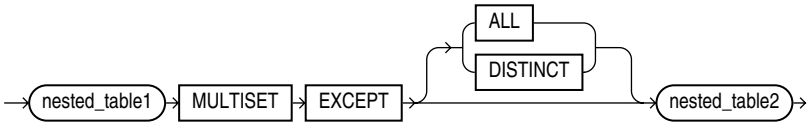
```
ALTER TABLE customers_demo
  ADD (cust_address_ntab cust_address_tab_typ,
       cust_address2_ntab cust_address_tab_typ)
    NESTED TABLE cust_address_ntab STORE AS cust_address_ntab_store
    NESTED TABLE cust_address2_ntab STORE AS cust_address2_ntab_store;
```

最後に、oe.customers 表の cust_address 列のデータを使用して、2 つの新しいネストした表の列にデータをロードします。

```
UPDATE CUSTOMERS_DEMO cd
  SET cust_address_ntab =
    CAST(MULTISET(SELECT cust_address
                  FROM customers c
                  WHERE c.customer_id =
                        cd.customer_id) as cust_address_tab_typ);

UPDATE CUSTOMERS_DEMO cd
  SET cust_address2_ntab =
    CAST(MULTISET(SELECT cust_address
                  FROM customers c
                  WHERE c.customer_id =
                        cd.customer_id) as cust_address_tab_typ);
```

MULTISET EXCEPT

MULTISET EXCEPT は、引数として 2 つのネストした表を取り、1 つ目のネストした表に存在し、2 つ目のネストした表に存在しない要素を持つネストした表を戻します。入力する 2 つのネストした表は同じ型である必要があり、戻されるネストした表も同じ型です。



- ALL キーワードは、*nested_table2* に存在せず、*nested_table1* に存在するすべての要素を戻します。たとえば、ある特定の要素が *nested_table1* で *m* 回、*nested_table2* で *n* 回出現すると想定します。結果には、この要素は、*m>n* の場合は *(m-n)* 回、*m<=n* の場合は 0（ゼロ）回出現します。ALL はデフォルトです。
- DISTINCT キーワードは、出現回数にかかわらず、*nested_table1* と *nested_table2* の両方に存在するすべての要素を排除します。
- ネストした表の要素型は比較可能なものである必要があります。スカラー型以外の型の比較の可能性は、7-4 ページの「比較条件」を参照してください。

例

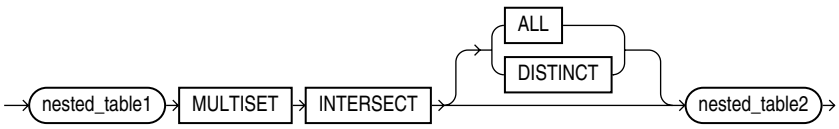
次の例では、2 つのネストした表を比較し、1 つ目のネストした表に存在し、2 つ目のネストした表に存在しない要素を持つネストした表を戻します。

```
SELECT customer_id, cust_address_ntab
  MULTISET EXCEPT DISTINCT cust_address2_ntab multiset_except
  FROM customers_demo;

CUSTOMER_ID MULTISET_EXCEPT(STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE, COUNTRY_ID)
----------- --------------------------------------------------------------------------------
        101 CUST_ADDRESS_TAB_TYP()
        102 CUST_ADDRESS_TAB_TYP()
        103 CUST_ADDRESS_TAB_TYP()
        104 CUST_ADDRESS_TAB_TYP()
        105 CUST_ADDRESS_TAB_TYP()
. . .
```

この例では、表 customers_demo と、データを含むネストした表の列が 2 つ必要です。この表およびネストした表の列を作成する方法については、4-6 ページの「MULTISET 演算子」を参照してください。

MULTISET INTERSECT

MULTISET INTERSECT は、引数として 2 つのネストした表を取り、入力する 2 つのネストした表に共通する値を持つネストした表を戻します。入力する 2 つのネストした表は同じ型である必要があり、戻されるネストした表も同じ型です。



- ALL キーワードは、入力する 2 つのネストした表に存在するすべての共通要素を戻します（重複する共通の値や、重複する共通の NULL も含まれます）。たとえば、ある特定の値が *nested_table1* で m 回、*nested_table2* で n 回出現する場合、結果には、この要素が min(m,n) 回出現します。ALL はデフォルトです。

- DISTINCT キーワードは、戻されるネストした表から重複を排除します（重複する NULL が存在する場合、これも排除されます）。
- ネストした表の要素型は比較可能なものである必要があります。スカラー型以外の型の比較の可能性は、7-4 ページの「比較条件」を参照してください。

例

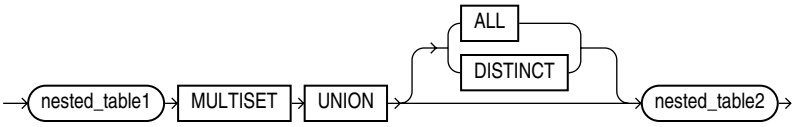
次の例では、2 つのネストした表を比較し、入力する両方のネストした表に存在する要素を持つネストした表を戻します。

```
SELECT customer_id, cust_address_ntab
  MULTISET INTERSECT DISTINCT cust_address2_ntab multiset_intersect
  FROM customers_demo
  ORDER BY customer_id;

CUSTOMER_ID MULTISET_INTERSECT(STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE, COUNTRY_ID
----------- --------------------------------------------------------------------------------
        101 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('514 W Superior St', '46901', 'Kokomo', 'IN', 'US'))
        102 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('2515 Bloyd Ave', '46218', 'Indianapolis', 'IN', 'US'))
        103 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('8768 N State Rd 37', '47404', 'Bloomington', 'IN', 'US'))
        104 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('6445 Bay Harbor Ln', '46254', 'Indianapolis', 'IN', 'US'))
        105 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('4019 W 3Rd St', '47404', 'Bloomington', 'IN', 'US'))
.
.
.
```

この例では、表 customers_demo と、データを含むネストした表の列が 2 つ必要です。この表およびネストした表の列を作成する方法については、4-6 ページの「MULTISET 演算子」を参照してください。

MULTISET UNION

MULTISET UNION は、引数として 2 つのネストした表を取り、入力する 2 つのネストした表に存在する値を持つネストした表を戻します。入力する 2 つのネストした表は同じ型である必要があり、戻されるネストした表も同じ型です。



- ALL キーワードは、入力する 2 つのネストした表に存在するすべての要素を戻します（重複する値や、重複する NULL も含まれます）。これはデフォルトです。
- DISTINCT キーワードは、戻されるネストした表から重複を排除します（重複する NULL が存在する場合、これも排除されます）。
- ネストした表の要素型は比較可能なものである必要があります。スカラー型以外の型の比較の可能性は、7-4 ページの「比較条件」を参照してください。

例

次の例では、2 つのネストした表を比較し、入力する両方のネストした表に存在する要素を持つネストした表を戻します。

```
SELECT customer_id, cust_address_ntab
  MULTISET UNION cust_address2_ntab multiset_union
  FROM customers_demo
  ORDER BY customer_id;

CUSTOMER_ID MULTISET_UNION(STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE, COUNTRY_ID)
----------- ------------------------------------------------------------------------------
        101 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('514 W Superior St', '46901', 'Kokomo', 'IN', 'US'),
                                 CUST_ADDRESS_TYP('514 W Superior St', '46901', 'Kokomo', 'IN', 'US'))
```

```
102 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('2515 Bloyd Ave', '46218', 'Indianapolis', 'IN', 'US'),
                         CUST_ADDRESS_TYP('2515 Bloyd Ave', '46218', 'Indianapolis', 'IN','US'))
103 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('8768 N State Rd 37', '47404', 'Bloomington', 'IN', 'US'),
                         CUST_ADDRESS_TYP('8768 N State Rd 37', '47404', 'Bloomington', 'IN', 'US'))
104 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('6445 Bay Harbor Ln', '46254', 'Indianapolis', 'IN', 'US'),
                         CUST_ADDRESS_TYP('6445 Bay Harbor Ln', '46254', 'Indianapolis', 'IN', 'US'))
105 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('4019 W 3Rd St', '47404', 'Bloomington', 'IN', 'US'),
                         CUST_ADDRESS_TYP('4019 W 3Rd St', '47404', 'Bloomington', 'IN', 'US'))
.
.
.
```

この例では、表 customers_demo と、データを含むネストした表の列が 2 つ必要です。この表およびネストした表の列を作成する方法については、4-6 ページの「MULTISET 演算子」を参照してください。

ユーザー定義演算子

ユーザー定義演算子は、組込み演算子のように、一連のオペランドを入力として受け取り、結果を戻します。ユーザー定義演算子は、ユーザーが CREATE OPERATOR 文で作成し、ユーザー定義の名前で識別されます。これらは、表、ビュー、型およびスタンドアロン・ファンクションと同じネームスペースに存在します。

新規の演算子を定義すると、他の組込み演算子のように SQL 文で使用できます。たとえば、SELECT 文の SELECT 構文のリスト、WHERE 句の条件、ORDER BY 句および GROUP BY 句でユーザー定義演算子を使用できます。ただし、これはユーザー定義オブジェクトであるため、演算子に対する EXECUTE 権限が必要です。

> **参照：** 演算子の作成例は、15-32 ページの「CREATE OPERATOR」を、ユーザー定義演算子の詳細は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。

5

ファンクション

ファンクションは、データ項目を操作し、結果を戻すという点で演算子に似ていますが、ファンクションと演算子は引数を指定する書式が異なります。次の書式によって、ファンクションでは 0（ゼロ）以上の引数を操作できます。

```
function(argument, argument, ...)
```

引数を指定しないファンクションは疑似列に似ています（第 3 章「疑似列」を参照）。ただし、疑似列は、通常、結果セット内の各行に対して異なる値を戻しますが、引数を指定しないファンクションは、各行に対して同じ値を戻します。

この章では、次の内容を説明します。

- SQL ファンクション
- ユーザー定義ファンクション

SQL ファンクション

SQL ファンクションは、Oracle Database に組み込まれており、適切な SQL 文で使用できます。SQL ファンクションと、PL/SQL で記述されたユーザー定義ファンクションを混同しないでください。

SQL ファンクションが戻す値のデータ型以外のデータ型の引数で SQL ファンクションをコールすると、Oracle は SQL ファンクションを実行する前に、その引数を必要なデータ型に変換します。

注意： NLS_COMP と NLS_SORT の設定を組み合せた値によって、文字をソートおよび比較するルールが決まります。ご使用のデータベースの NLS_COMP に LINGUISTIC が設定されている場合、この章のエンティティはすべて NLS_SORT パラメータによって指定されるルールに従って解釈されます。NLS_COMP が LINGUISTIC に設定されていない場合、ファンクションは NLS_SORT の設定に関係なく解釈されます。NLS_SORT は、明示的に設定できます。明示的に設定されていない場合は、NLS_LANGUAGE から導出されます。これらの設定の詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

SQL ファンクションの構文図では、データ型とともに引数が示されています。SQL 構文にパラメータ *function* が指定されている場合は、この項で説明するファンクションの 1 つに置き換えます。ファンクションは、引数のデータ型および戻り値によってグループ化されています。

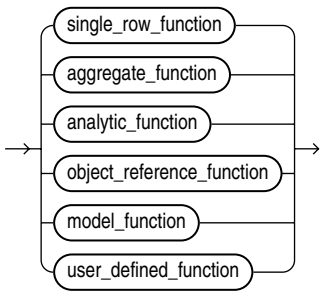
注意： LOB 列に SQL ファンクションを適用すると、Oracle Database は、SQL および PL/SQL の処理中に一時 LOB を作成します。ご使用のアプリケーションの一時 LOB を格納するために、十分な一時表領域が割り当てられていることを確認してください。

参照：

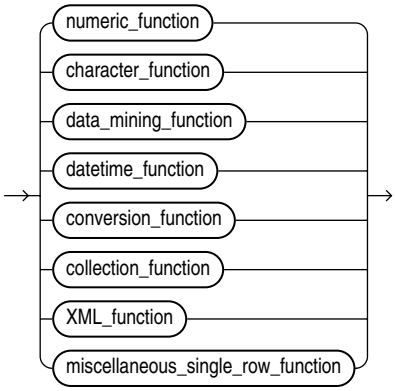
- ユーザー・ファンクションの詳細は、5-249 ページの「ユーザー定義ファンクション」を参照してください。データ型の暗黙的な変換の詳細は、2-39 ページの「データ変換」を参照してください。
- Oracle Text で使用するファンクションの詳細は、『Oracle Text リファレンス』を参照してください。
- Oracle Data Mining で使用するファンクションの詳細は、『Oracle Data Mining アプリケーション開発者ガイド』を参照してください。

次に、ファンクションのカテゴリを表す構文を示します。

***function*::=**



***single_row_function*::=**



次の項に、前述の図のユーザー定義ファンクション以外の各グループにおける組込み SQL ファンクションを示します。すべての組込み SQL ファンクションを、アルファベット順に説明します。

> **参照：** 5-249 ページの「ユーザー定義ファンクション」および 14-48 ページの「CREATE FUNCTION」を参照してください。

単一行ファンクション

単一行ファンクションは、問合せ対象の表またはビューの各行に対して 1 つの結果行を戻します。これらのファンクションは、SELECT 構文のリスト、WHERE 句、START WITH 句、CONNECT BY 句および HAVING 句に指定できます。

数値ファンクション

数値ファンクションは入力として数値を受け取り、結果として数値を戻します。数値ファンクションのほとんどは、38 桁（10 進）の NUMBER 値を戻します。超越関数（COS、COSH、EXP、LN、LOG、SIN、SINH、SQRT、TAN および TANH）は、36 桁（10 進）の値を戻します。超越関数の ACOS、ASIN、ATAN、ATAN2 は、30 桁（10 進）の値を戻します。数値ファンクションを次に示します。

ABS
ACOS
ASIN
ATAN
ATAN2
BITAND
CEIL
COS
COSH
EXP
FLOOR
LN
LOG
MOD
NANVL
POWER
REMAINDER
ROUND（数値）
SIGN
SIN
SINH
SQRT
TAN

TANH
TRUNC（数値）
WIDTH_BUCKET

文字値を戻す文字ファンクション

文字値を戻す文字ファンクションは、特に指定がないかぎり、次のデータ型の値を戻します。

- 入力引数が CHAR または VARCHAR2 の場合、戻される値は VARCHAR2 になります。
- 入力引数が NCHAR または NVARCHAR2 の場合、戻される値は NVARCHAR2 になります。

ファンクションによって戻される値の長さは、戻されるデータ型の最大長によって制限されます。

- CHAR または VARCHAR2 型の値を戻すファンクションの戻り値の長さが制限を超えた場合、Oracle Database は戻り値から制限を超えた部分を切り捨てて、エラー・メッセージを表示せずにその結果を戻します。
- CLOB 型の値を戻すファンクションの戻り値の長さが制限を超えた場合、Oracle はエラーを表示し、データを戻しません。

文字値を戻す文字ファンクションを次に示します。

CHR
CONCAT
INITCAP
LOWER
LPAD
LTRIM
NLS_INITCAP
NLS_LOWER
NLSSORT
NLS_UPPER
REGEXP_REPLACE
REGEXP_SUBSTR
REPLACE
RPAD
RTRIM
SOUNDEX
SUBSTR
TRANSLATE
TREAT
TRIM
UPPER

NLS 文字ファンクション

NLS 文字ファンクションは、キャラクタ・セットの情報を戻します。NLS 文字ファンクションを次に示します。

NLS_CHARSET_DECL_LEN
NLS_CHARSET_ID
NLS_CHARSET_NAME

数値を戻す文字ファンクション

数値を戻す文字ファンクションの引数には、すべての文字データ型を指定できます。

数値を戻す文字ファンクションを次に示します。

ASCII
INSTR
LENGTH
REGEXP_INSTR

日時ファンクション

日時ファンクションは、日付（DATE）、タイムスタンプ（TIMESTAMP、TIMESTAMP WITH TIME ZONE、TIMESTAMP WITH LOCAL TIME ZONE）および期間（INTERVAL DAY TO SECOND、INTERVAL YEAR TO MONTH）の値を操作します。

一部の日付ファンクションは、Oracle の DATE データ型（ADD_MONTHS、CURRENT_DATE、LAST_DAY、NEW_TIME および NEXT_DAY）用に設計されています。これらのファンクションの引数にタイムスタンプ値を指定すると、Oracle Database は入力された型を DATE 値に内部的に変換し、DATE 値を戻します。ただし、MONTHS_BETWEEN ファンクションは数値を戻し、ROUND および TRUNC ファンクションはタイムスタンプ値または期間値を受け入れません。

その他の日時ファンクションは、3 種類のすべてのデータ型（日付、タイムスタンプ、期間）を受け入れ、それらのいずれかのデータ型の値を戻すように設計されています。

SYSDATE、SYSTIMESTAMP、CURRENT_TIMESTAMP などの、現在のシステム日時情報を戻す日時ファンクションはすべて、SQL 文で参照される回数に関係なく、SQL 文ごとに 1 回評価されます。

日時ファンクションを次に示します。

ADD_MONTHS
CURRENT_DATE
CURRENT_TIMESTAMP
DBTIMEZONE
EXTRACT（日時）
FROM_TZ
LAST_DAY
LOCALTIMESTAMP
MONTHS_BETWEEN
NEW_TIME
NEXT_DAY
NUMTODSINTERVAL
NUMTOYMINTERVAL
ROUND（日付）
SESSIONTIMEZONE
SYS_EXTRACT_UTC
SYSDATE
SYSTIMESTAMP
TO_CHAR（日時）
TO_TIMESTAMP
TO_TIMESTAMP_TZ
TO_DSINTERVAL
TO_YMINTERVAL
TRUNC（日付）
TZ_OFFSET

一般的な比較ファンクション

一般的な比較ファンクションは、値の集合から最大値または最小値（あるいはその両方）を決定します。一般的な比較ファンクションを次に示します。

GREATEST
LEAST

変換ファンクション

変換ファンクションは、あるデータ型から他のデータ型に値を変換します。一般に、ファンクション名は *datatype* TO *datatype* の書式で指定されます。最初のデータ型は入力データ型です。2 番目のデータ型は出力データ型です。SQL 変換ファンクションを次に示します。

ASCIISTR
BIN_TO_NUM
CAST

CHARTOROWID
COMPOSE
CONVERT
DECOMPOSE
HEXTORAW
NUMTODSINTERVAL
NUMTOYMINTERVAL
RAWTOHEX
RAWTONHEX
ROWIDTOCHAR
ROWIDTONCHAR
SCN_TO_TIMESTAMP
TIMESTAMP_TO_SCN
TO_BINARY_DOUBLE
TO_BINARY_FLOAT
TO_CHAR（文字）
TO_CHAR（日時）
TO_CHAR（数値）
TO_CLOB
TO_DATE
TO_DSINTERVAL
TO_LOB
TO_MULTI_BYTE
TO_NCHAR（文字）
TO_NCHAR（日時）
TO_NCHAR（数値）
TO_NCLOB
TO_NUMBER
TO_DSINTERVAL
TO_SINGLE_BYTE
TO_TIMESTAMP
TO_TIMESTAMP_TZ
TO_YMINTERVAL
TO_YMINTERVAL
TRANSLATE ... USING
UNISTR

ラージ・オブジェクト・ファンクション

ラージ・オブジェクト・ファンクションは、LOB を操作します。ラージ・オブジェクト・ファンクションを次に示します。

BFILENAME
EMPTY_BLOB、EMPTY_CLOB

収集ファンクション

収集ファンクションは、ネストした表および VARRAY を操作します。SQL 収集ファンクションを次に示します。

CARDINALITY
COLLECT
POWERMULTISET
POWERMULTISET_BY_CARDINALITY
SET

階層ファンクション

階層ファンクションは、階層パス情報を結果セットに適用します。

SYS_CONNECT_BY_PATH

データ・マイニング・ファンクション

データ・マイニング・ファンクションは、DBMS_DATA_MINING パッケージまたは Oracle Data Mining Java API を使用して作成したモデルを操作します。SQL データ・マイニング・ファンクションを次に示します。

CLUSTER_ID
CLUSTER_PROBABILITY
CLUSTER_SET
FEATURE_ID
FEATURE_SET
FEATURE_VALUE
PREDICTION
PREDICTION_BOUNDS
PREDICTION_COST
PREDICTION_DETAILS
PREDICTION_PROBABILITY
PREDICTION_SET

XML ファンクション

XML ファンクションは、XML 文書またはフラグメントを操作または戻します。出力の書式設定など、これらのファンクションを使用した XML データの選択および問合せの詳細は、『Oracle XML DB 開発者ガイド』を参照してください。SQL XML ファンクションを次に示します。

APPENDCHILDXML
DELETEXML
DEPTH
EXTRACT（XML）
EXISTSNODE
EXTRACTVALUE
INSERTCHILDXML
INSERTXMLBEFORE
PATH
SYS_DBURIGEN
SYS_XMLAGG
SYS_XMLGEN
UPDATEXML
XMLAGG
XMLCAST
XMLCDATA
XMLCOLATTVAL
XMLCOMMENT
XMLCONCAT
XMLDIFF
XMLELEMENT
XMLEXISTS
XMLFOREST
XMLPARSE
XMLPATCH
XMLPI
XMLQUERY
XMLROOT
XMLSEQUENCE
XMLSERIALIZE
XMLTABLE
XMLTRANSFORM

エンコーディング・ファンクションおよびデコーディング・ファンクション

エンコーディング・ファンクションおよびデコーディング・ファンクションでは、データベース内のデータを調査およびデコードできます。

DECODE
DUMP
ORA_HASH
VSIZE

NULL 関連ファンクション

NULL 関連ファンクションは、NULL 処理を簡単にします。NULL 関連ファンクションを次に示します。

COALESCE
LNNVL
NULLIF
NVL
NVL2

環境ファンクションおよび識別子ファンクション

環境ファンクションおよび識別子ファンクションは、インスタンスとセッションの情報を提供します。これらのファンクションを次に示します。

SYS_CONTEXT
SYS_GUID
SYS_TYPEID
UID
USER
USERENV

集計ファンクション

集計ファンクションは、単一行に基づく結果行を戻すのではなく、行のグループに基づく単一結果行を戻します。集計ファンクションは、SELECT 構文のリスト、ORDER BY および HAVING 句に指定できます。集計ファンクションは、通常、SELECT 文の GROUP BY 句で使用され、この句の中で問合せ対象の表またはビューの行がグループ化されます。GROUP BY 句を含む問合せでは、SELECT 構文のリストの要素は、集計ファンクション、GROUP BY 式、定数またはこれらのいずれかを含む式になります。Oracle は、集計ファンクションを行の各グループに適用し、各グループに単一の結果行を戻します。

GROUP BY 句を指定しないと、SELECT 構文のリスト内の集計ファンクションは、問合せ対象の表またはビューのすべての行に適用されます。HAVING 句で集計ファンクションを使用して、グループを出力しないこともできます。このとき、出力は、問合せ対象の表またはビューの各行の値ではなく、集計ファンクションの結果に基づきます。

> **参照：** 問合せおよび副問合せにおける GROUP BY 句および HAVING 句の詳細は、19-34 ページの「GROUP BY 句の使用例 :」および 19-25 ページの「HAVING 句」を参照してください。

単一の引数を指定する多くの集計ファンクションには、次の句があります。

- DISTINCT 句を指定すると、集計ファンクションは引数式の重複行を排除した値のみを考慮します。
- ALL 句を指定すると、集計ファンクションは重複行を含むすべての値を考慮します。

たとえば、1、1、1、3 の平均値は DISTINCT では 2 となり、ALL では 1.5 となります。どの句も指定しない場合、デフォルトで ALL が使用されます。

集計ファンクションには、分析ファンクションの構文の一部である *windowing_clause* を使用できるものがあります。この句の詳細は、5-12 ページの「windowing_clause」を参照してください。この項の最後に示す集計ファンクションのリストでは、*windowing_clause* を使用できるファンクションにアスタリスク（*）が付いています。

COUNT(*)、GROUPING および GROUPING_ID を除くすべての集計ファンクションは、NULL を無視します。集計ファンクションに対する引数に NVL ファンクションを使用して、NULL をある値で置き換えることができます。COUNT および REGR_COUNT は NULL を戻しません。数字または 0（ゼロ）を戻します。その他の集計ファンクションでは、データ・セットに行がない場合、または集計ファンクションに対する引数として NULL を持つ行のみがある場合は NULL を戻します。

集計ファンクション MIN、MAX、SUM、AVG、COUNT、VARIANCE および STDDEV の後に、KEEP キーワードを指定すると、FIRST または LAST ファンクションと組み合せて使用することができ、与えられたソート指定に対して、FIRST または LAST としてランク付けされた一連の行の一連の値を操作できます。詳細は、5-71 ページの「FIRST」を参照してください。

集計ファンクションはネストできます。たとえば、次の例では、サンプル・スキーマ hr のすべての部門における最高額の給与の平均を計算します。

```
SELECT AVG(MAX(salary)) FROM employees GROUP BY department_id;

AVG(MAX(SALARY))
----------------
           10925
```

この計算では、GROUP BY 句（department_id）で定義されているグループごとの内部集計（MAX（salary））を評価し、その結果をもう一度集計しています。

集計ファンクションを次に示します。

AVG
COLLECT
CORR
CORR_*
COUNT
COVAR_POP
COVAR_SAMP
CUME_DIST
DENSE_RANK
FIRST
GROUP_ID
GROUPING
GROUPING_ID
LAST
MAX
MEDIAN
MIN
PERCENTILE_CONT
PERCENTILE_DISC
PERCENT_RANK
RANK
REGR_（線形回帰）ファンクション
STATS_BINOMIAL_TEST
STATS_CROSSTAB
STATS_F_TEST
STATS_KS_TEST
STATS_MODE
STATS_MW_TEST
STATS_ONE_WAY_ANOVA
STATS_T_TEST_*
STATS_WSR_TEST
STDDEV

STDDEV_POP
STDDEV_SAMP
SUM
SYS_XMLAGG
VAR_POP
VAR_SAMP
VARIANCE
XMLAGG

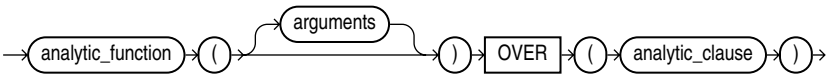
分析ファンクション

分析ファンクションは、行のグループに基づいて集計値を計算します。各グループに対して複数の行を戻す点で、集計ファンクションと異なります。行のグループを**ウィンドウ**といい、*analytic_clause* で定義されます。各行に対して、行のスライディング・ウィンドウが定義されます。このウィンドウによって、カレント行の計算に使用される行の範囲が決定されます。ウィンドウの大きさは、行の物理数値または時間などのロジカル・インターバルに基づきます。

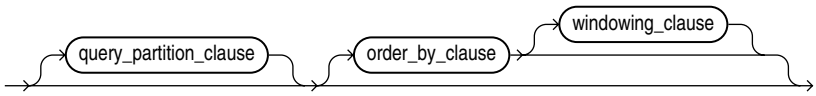
分析ファンクションは、問合せで最後に実行される演算（最後の ORDER BY 句を除く）の集合です。すべての結合およびすべての WHERE、GROUP BY および HAVING 句は、分析ファンクションが処理される前に実行されます。そのため、分析ファンクションは、SELECT 構文のリストまたは ORDER BY 句のみに指定できます。

通常、分析ファンクションは、累積集計、移動集計、センター集計およびレポート集計の実行に使用されます。

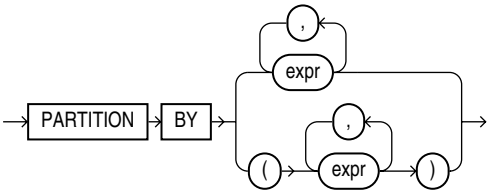
***analytic_function*::=**



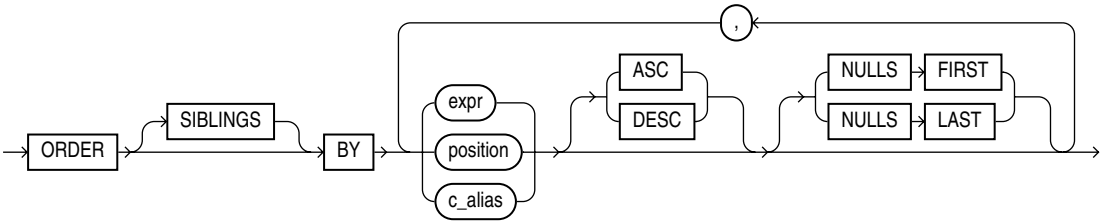
***analytic_clause*::=**



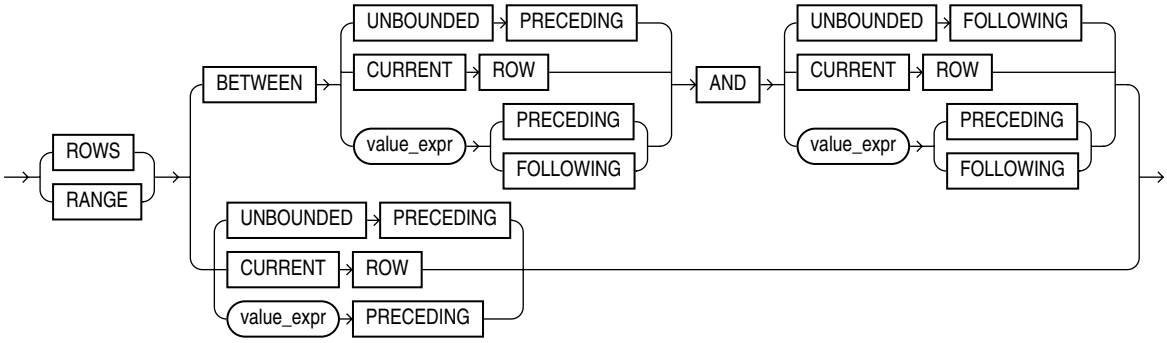
query_partition_clause::=



order_by_clause::=



windowing_clause::=



次に、この構文のセマンティクスを示します。

analytic_function

分析ファンクションの名前を指定します（セマンティクスの説明の後に示す分析ファンクションのリストを参照）。

引数

分析ファンクションには引数を 0 ～ 3 個指定します。引数には、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を指定できます。Oracle は、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換します。個々のファンクションに特に指定がないかぎり、戻り型もその引数のデータ型となります。

> **参照：** 数値の優先順位の詳細は、2-15 ページの「数値の優先順位」を参照してください。暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

analytic_clause

OVER *analytic_clause* 句は、ファンクションが問合せ結果セットを操作することを示します。この句は、FROM、WHERE、GROUP BY および HAVING 句の後に計算されます。SELECT 構文のリストのこの句または ORDER BY 句に分析ファンクションを指定できます。分析ファンクションに基づいて、問合せの結果をフィルタするには、これらのファンクションを親問合せ内でネストした後、ネストされた副問合せの結果をフィルタします。

***analytic_clause* の注意事項：** *analytic_clause* には、次の注意事項があります。

- *analytic_clause* のどの部分にも、分析ファンクションを指定して分析ファンクションをネストできません。ただし、副問合せで分析ファンクションを指定して、別の分析ファンクションを計算することはできます。
- OVER *analytic_clause* には、組込み分析ファンクションと同様に、ユーザー定義の分析ファンクションを指定できます。14-48 ページの「CREATE FUNCTION」を参照してください。

query_partition_clause

PARTITION BY 句を使用すると、1 つ以上の *value_expr* に基づいて、問合せ結果セットをグループに分割できます。この句を省略すると、ファンクションは問合せ結果セットのすべての行を単一のグループとして扱います。

分析ファンクションで *query_partition_clause* を使用するには、構文の上位ブランチ（カッコなし）を使用します。この句をモデルの問合せ（*model_column_clauses* 内）またはパーティション化された外部結合（*outer_join_clause* 内）で使用するには、構文の下位ブランチ（カッコ付き）を使用します。

同じまたは異なる PARTITION BY キーで、同じ問合せに複数の分析ファンクションを指定できます。

問い合せているオブジェクトにパラレル属性があり、*query_partition_clause* で分析ファンクションを指定する場合は、ファンクションの計算もパラレル化されます。

有効な値の *value_expr* は、定数、列、非分析ファンクション、ファンクション式、またはこれらのいずれかを含む式です。

order_by_clause

order_by_clause を使用すると、パーティション内でのデータの順序付け方法を指定できます。PERCENTILE_CONT および（単一キーのみを適用する）PERCENTILE_DISC 以外の分析ファンクションでは、各キーが *value_expr* で定義され、順序付けシーケンスで修飾された複数キーのパーティションの値を順序付けできます。

各ファンクションには、複数の順序式を指定できます。これは、2 番目の式が最初の式にある同一値との間の関連性を変換できるため、値をランク付けするファンクションを使用する場合に特に有効です。

order_by_clause の結果が複数行の個々の値である場合、ファンクションは各行の同じ値を戻します。この動作の詳細は、5-182 ページの「SUM」の分析例を参照してください。

ORDER BY 句の制限事項： ORDER BY 句には次の制限事項があります。

- *order_by_clause* を分析ファンクションで使用する場合、式（*expr*）が必要です。SIBLINGS キーワードは無効です（これは、階層問合せでのみ有効です）。位置（*position*）および列別名（*c_alias*）も無効です。それ以外で使用する場合、この *order_by_clause* は、問合せまたは副問合せ全体を順序付ける場合に使用するものと同じです。
- RANGE キーワードを使用する分析ファンクションでは、次の 2 つのウィンドウのいずれかを指定する場合に、ORDER BY 句で複数のソート・キーを使用できます。
  - RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW。この短縮形は、RANGE UNBOUNDED PRECEDING です。
  - RANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING。

  この 2 つ以外のウィンドウ境界では、分析ファンクションの ORDER BY 句でソート・キーを 1 つしか持てません。この制限事項は、ROW キーワードで指定したウィンドウ境界には適用されません。

ASC | DESC　順序付けシーケンス（昇順または降順）を指定します。デフォルトは ASC です。

NULLS FIRST | NULLS LAST　NULL 値を含む戻された行が順序の最初にくるか、最後にくるかを指定します。

NULLS LAST は昇順のデフォルトで、NULLS FIRST は降順のデフォルトです。

分析ファンクションは、常に、ファンクションの *order_by_clause* で指定された順序で行を操作します。ただし、ファンクションの *order_by_clause* は結果の順序を保証しません。最終結果の順序を保証するには、問合せの *order_by_clause* を使用してください。

> **参照：** この句の詳細は、19-29 ページの「SELECT」の「order_by_clause」を参照してください。

windowing_clause

一部の分析ファンクションでは、*windowing_clause* を使用できます。7-16 ページに示す分析ファンクションのリストでは、*windowing_clause* を使用できるファンクションにアスタリスク（*）が付いています。

ROWS | RANGE　これらのキーワードは、各行に対して、ファンクションの結果の計算に使用されるウィンドウ（行の物理集合または論理集合）を定義します。ファンクションは、ウィンドウのすべての行に適用されます。ウィンドウは、問合せ結果セット内またはパーティションの上から下まで移動します。

- ROWS は、物理単位（行）でウィンドウを指定します。
- RANGE は、論理オフセットとしてウィンドウを指定します。

order_by_clause を指定しないと、この句を指定できません。RANGE 句で定義したウィンドウ境界には、*order_by_clause* で指定できる式が 1 つのみのものもあります。詳細は、5-12 ページの「ORDER BY 句の制限事項 :」を参照してください。

分析ファンクションが論理オフセットで戻す値は、常に決定的なものです。ただし、分析ファンクションが物理オフセットで戻す値は、順序式の結果が一意の順序にならないかぎり、非決定的な結果を生成することがあります。*order_by_clause* に複数の列を指定して、結果の順序を一意にする必要があります。

BETWEEN ...AND　BETWEEN ... AND 句を使用すると、ウィンドウにスタート・ポイントおよびエンド・ポイントを指定できます。最初の式（AND の前）はスタート・ポイントを定義し、2 番目の式（AND の後）はエンド・ポイントを定義します。

BETWEEN を省略してエンド・ポイントを 1 つのみ指定すると、Oracle はそれをスタート・ポイントとみなし、デフォルトでカレント行をエンド・ポイントに指定します。

UNBOUNDED PRECEDING　UNBOUNDED PRECEDING を指定すると、パーティションの最初の行で、ウィンドウが開始します。これはスタート・ポイントの指定で、エンド・ポイントの指定としては使用できません。

UNBOUNDED FOLLOWING　UNBOUNDED FOLLOWING を指定すると、パーティションの最後の行で、ウィンドウが終了します。これはエンド・ポイントの指定で、スタート・ポイントの指定としては使用できません。

CURRENT ROW　スタート・ポイントとして、ウィンドウがカレント行または値（それぞれ ROW または RANGE を指定したかどうかに基づく）で開始することを指定します。この場合、*value_expr* PRECEDING をエンド・ポイントにできません。

エンド・ポイントとして、ウィンドウがカレント行または値（それぞれ ROW または RANGE を明示的に指定したかどうかに基づく）で終了することを指定します。この場合、*value_expr* FOLLOWING をスタート・ポイントにできません。

***value_expr* PRECEDING または *value_expr* FOLLOWING**　RANGE または ROW に対して、次のことがいえます。

- *value_expr* FOLLOWING がスタート・ポイントの場合、エンド・ポイントは *value_expr* FOLLOWING である必要があります。
- *value_expr* PRECEDING がエンド・ポイントの場合、スタート・ポイントは *value_expr* PRECEDING である必要があります。

数値形式の時間間隔で定義されている論理ウィンドウを定義する場合、変換ファンクションを使用する必要があります。

> **参照 :**　数値時間から間隔への変換の詳細は、5-114 ページの「NUMTOYMINTERVAL」および 5-113 ページの「NUMTODSINTERVAL」を参照してください。

ROWS を指定した場合、次のことがいえます。

- *value_expr* は物理オフセットになります。これは定数または式であり、正数値に評価する必要があります。
- *value_expr* がスタート・ポイントの一部の場合、エンド・ポイントの前にある行に評価する必要があります。

RANGE を指定した場合、次のことがいえます。

- *value_expr* は論理オフセットになります。これは、正数値または期間リテラルに評価する定数または式である必要があります。期間リテラルの詳細は、2-44 ページの「リテラル」を参照してください。
- *order_by_clause* には、式を 1 つのみ指定できます。
- *value_expr* が数値に対して評価を行う場合、ORDER BY *expr* は数値データ型または DATE データ型である必要があります。
- *value_expr* が間隔値に対して評価を行う場合、ORDER BY *expr* は DATE データ型である必要があります。

windowing_clause を完全に省略した場合、デフォルトで RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW になります。

分析ファンクションは、通常、データ・ウェアハウス環境で使用されます。次に示す分析ファンクションのリストでは、*windowing_clause* を含む完全な構文を使用できるファンクションには、アスタリスク（*）が付いています。

AVG *
CORR *
COVAR_POP *
COVAR_SAMP *
COUNT *
CUME_DIST
DENSE_RANK
FIRST
FIRST_VALUE *
LAG
LAST
LAST_VALUE *
LEAD
MAX *
MIN *
NTILE
PERCENT_RANK
PERCENTILE_CONT
PERCENTILE_DISC
RANK
RATIO_TO_REPORT
REGR_（線形回帰）ファンクション *
ROW_NUMBER
STDDEV *
STDDEV_POP *
STDDEV_SAMP *
SUM *
VAR_POP *
VAR_SAMP *
VARIANCE *

参照： これらのファンクションおよびその使用方法の詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

オブジェクト参照ファンクション

オブジェクト参照ファンクションは、指定されたオブジェクト型のオブジェクトへの参照となる REF 値を操作します。オブジェクト参照ファンクションを次に示します。

DEREF
MAKE_REF
REF
REFTOHEX
VALUE

> **参照：** REF データ型の詳細は、『Oracle Database オブジェクト・リレーショナル開発者ガイド』を参照してください。

モデル・ファンクション

モデル・ファンクションは、SELECT 文の *model_clause* でのみ使用できます。新しいモデル・ファンクションを次に示します。

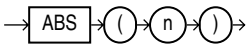
CV
ITERATION_NUMBER
PRESENTNNV
PRESENTV
PREVIOUS

SQL ファンクションのリスト（アルファベット順）

この項では、SQL ファンクションをアルファベット順に示し、説明します。

ABS

構文



用途

ABS は、*n* の絶対値を戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

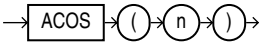
例

次の例では、-15 の絶対値を戻します。

```
SELECT ABS(-15) "Absolute" FROM DUAL;

  Absolute
----------
        15
```

ACOS

構文



用途

ACOS は、*n* のアーク・コサインを戻します。引数 *n* は -1 ～ 1 の範囲で、ファンクションによって戻される値は 0 ～ π（ラジアン）の範囲です。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数が BINARY_FLOAT の場合、このファンクションは BINARY_DOUBLE を戻します。それ以外の場合、引数と同じ数値データ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

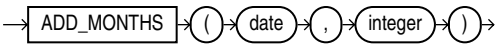
例

次の例では、.3 のアーク・コサインを戻します。

```
SELECT ACOS(.3)"Arc_Cosine" FROM DUAL;

Arc_Cosine
----------
1.26610367
```

ADD_MONTHS

構文



用途

ADD_MONTHS は、日付 *date* に月数 *integer* を加えて戻します。月は、セッション・パラメータ NLS_CALENDAR によって定義されます。引数 date には、日時値、または暗黙的に DATE に変換可能な任意の値を指定できます。引数 *integer* には、整数、または暗黙的に整数に変換可能な任意の値を指定できます。戻り型は、*date* のデータ型に関係なく常に DATE です。*date* が月の最終日の場合、または結果の月の日数が *date* の日付コンポーネントよりも少ない場合、戻される値は結果の月の最終日となります。それ以外の場合、結果には *date* と同じ日付コンポーネントが含まれます。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

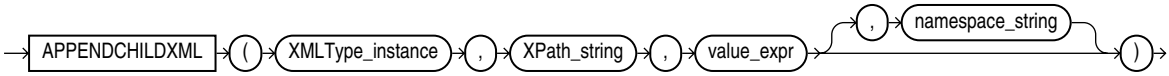
例

次の例では、サンプル表 employees の *hire_date* 後の月を戻します。

```
SELECT TO_CHAR(
     ADD_MONTHS(hire_date,1),
     'DD-MON-YYYY') "Next month"
     FROM employees
     WHERE last_name = 'Baer';

Next Month
-----------
07-JUL-1994
```

APPENDCHILDXML

構文



用途

APPENDCHILDXML は、ユーザー指定の値を、XPath 式で指定したノードの子としてターゲット XML に追加します。

- *XMLType_instance* は、XMLType のインスタンスです。
- *XPath_string* は、1 つ以上の子ノードが追加されるノードを 1 つ以上指定する XPath 式です。先頭にスラッシュを付けて絶対 *XPath_string* を指定したり、先頭のスラッシュを省略して相対 *XPath_string* を指定できます。先頭のスラッシュを省略した場合、相対パスのコンテキストは、デフォルトでルート・ノードに設定されます。
- *value_expr* は、XMLType の 1 つ以上のノードを指定します。これは文字列に変換する必要があります。
- オプションの *namespace_string* は、*XPath_string* のネームスペース情報を提供します。このパラメータは、VARCHAR2 型である必要があります。

> **参照：** このファンクションの詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

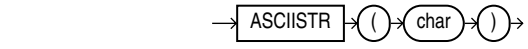
例

次の例では、/Building ノードの値が「Rented」の場合に、/Owner ノードを oe.warehouses 表の warehouse_spec の /Warehouse/Building ノードに追加します。

```
UPDATE warehouses SET warehouse_spec =
   APPENDCHILDXML(warehouse_spec,
   'Warehouse/Building',
   XMLType('<Owner>Grandco</Owner>'))
   WHERE EXTRACTVALUE(warehouse_spec, '/Warehouse/Building') = 'Rented';

SELECT warehouse_id, warehouse_name,
   EXTRACTVALUE(warehouse_spec, '/Warehouse/Building/Owner') "Prop.Owner"
   FROM warehouses
   WHERE EXISTSNODE(warehouse_spec, '/Warehouse/Building/Owner') = 1;

WAREHOUSE_ID WAREHOUSE_NAME  Prop.Owner
------------ --------------- ----------
           2 San Francisco   Grandco
           3 New Jersey      Grandco
```

ASCIISTR

構文



用途

ASCIISTR は、任意のキャラクタ・セットの文字列、または文字列に変換する式を引数として取り、データベース・キャラクタ・セットの ASCII 文字列を戻します。ASCII 文字以外は、\xxxx という書式に変換されます。xxxx は、UTF-16 のコード単位です。

参照： Unicode キャラクタ・セットおよびキャラクタ・セマンティクスの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

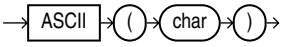
例

次の例では、テキスト文字列「ABÄCDE」を ASCII 文字列で戻します。

```
SELECT ASCIISTR('ABÄCDE') FROM DUAL;

ASCIISTR('
----------
AB\00C4CDE
```

ASCII

構文



用途

ASCII は、*char* の最初の文字の、データベース・キャラクタ・セットでの 10 進表記を戻します。

char のデータ型は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 です。戻り値のデータ型は NUMBER です。データベース・キャラクタ・セットが 7 ビットの ASCII の場合、このファンクションは ASCII 値を戻します。データベース・キャラクタ・セットが EBCDIC コードの場合、このファンクションは EBCDIC 値を戻します。このファンクションと一致する EBCDIC 文字ファンクションは存在しません。

このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。

参照： 詳細は、2-36 ページの「データ型の比較規則」を参照してください。

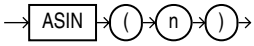
例

次の例では、姓が文字「L」で始まり、ASCII 表記が 76 の従業員を戻します。

```
SELECT last_name FROM employees
   WHERE ASCII(SUBSTR(last_name, 1, 1,)) = 76;

LAST_NAME
-------------------------
Ladwig
Landry
Lee
Livingston
```

ASIN

構文



用途

ASIN は、*n* のアーク・サインを戻します。引数 *n* は -1 ～ 1 の範囲で、ファンクションによって戻される値は - π/2 ～ π/2（ラジアン）の範囲です。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数が BINARY_FLOAT の場合、このファンクションは BINARY_DOUBLE を戻します。それ以外の場合、引数と同じ数値データ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

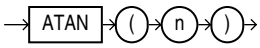
例

次の例では、.3 のアーク・サインを戻します。

```
SELECT ASIN(.3) "Arc_Sine" FROM DUAL;

 Arc_Sine
----------
.304692654
```

ATAN

構文



用途

ATAN は、*n* のアーク・タンジェントを戻します。引数 *n* の範囲に制限はなく、ファンクションによって戻される値は - π/2 ～ π/2（ラジアン）の範囲です。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数が BINARY_FLOAT の場合、このファンクションは BINARY_DOUBLE を戻します。それ以外の場合、引数と同じ数値データ型を戻します。

> **参照：** ATAN2 ファンクションの詳細は、5-20 ページの「ATAN2」を参照してください。暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

例

次の例では、.3 のアーク・タンジェントを戻します。

```
SELECT ATAN(.3) "Arc_Tangent" FROM DUAL;

Arc_Tangent
----------
.291456794
```

ATAN2

構文

用途

ATAN2 は、*n1* および *n2* のアーク・タンジェントを戻します。引数 *n1* の範囲に制限はなく、*n1* と *n2* の符号により、ファンクションによって戻される値は -π ～ π（ラジアン）の範囲です。ATAN2(*n1*,*n2*) と、ATAN(*n1*/*n2*) は同じです。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。いずれかの引数が BINARY_FLOAT または BINARY_DOUBLE の場合、このファンクションは BINARY_DOUBLE を戻します。それ以外の場合、NUMBER を戻します。

> **参照：** ATAN ファンクションの詳細は、5-19 ページの「ATAN」を参照してください。暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

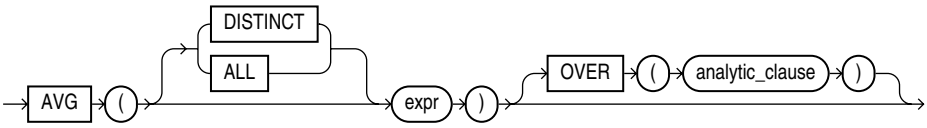
例

次の例では、.3 および .2 のアーク・タンジェントを戻します。

```
SELECT ATAN2(.3, .2) "Arc_Tangent2" FROM DUAL;

Arc_Tangent2
------------
  .982793723
```

AVG

構文



> **参照：** 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

AVG は、*expr* の平均値を戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

DISTINCT を指定する場合は、*analytic_clause* の *query_partition_clause* のみ指定できます。*order_by_clause* および *windowing_clause* は指定できません。

> **参照：** *expr* の書式の詳細は、6-2 ページの「SQL 式」を参照してください。5-8 ページの「集計ファンクション」も参照してください。

集計の例

次の例では、hr.employees 表にあるすべての従業員の平均給与を計算します。

```
SELECT AVG(salary) "Average" FROM employees;

       Average
--------------
    6461.68224
```

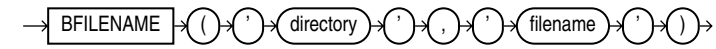
分析の例

次の例では、employees 表の各従業員について、ある期間内に雇用された従業員の平均給与を所属別に計算します。

```
SELECT manager_id, last_name, hire_date, salary,
   AVG(salary) OVER (PARTITION BY manager_id ORDER BY hire_date
   ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS c_mavg
   FROM employees
   ORDER BY manager_id, last_name, hire_date, salary, AVG(salary);

MANAGER_ID LAST_NAME                 HIRE_DATE     SALARY     C_MAVG
---------- ------------------------- --------- ---------- ----------
       100 Kochhar                   21-SEP-89      17000      17000
       100 De Haan                   13-JAN-93      17000      15000
       100 Raphaely                  07-DEC-94      11000 11966.6667
       100 Kaufling                  01-MAY-95       7900 10633.3333
       100 Hartstein                 17-FEB-96      13000 9633.33333
       100 Weiss                     18-JUL-96       8000 11666.6667
       100 Russell                   01-OCT-96      14000 11833.3333
       100 Partners                  05-JAN-97      13500 13166.6667
. . .
```

BFILENAME

構文



用途

BFILENAME は、サーバーのファイル・システムの物理 LOB バイナリ・ファイルに対応付けられている BFILE ロケータを戻します。

- '*directory*' は、実際にファイルが存在するサーバーのファイル・システム上のフルパス名に対する別名となるデータベース・オブジェクトです。
- '*filename*' は、サーバーのファイル・システムにあるファイルの名前です。

SQL 文、PL/SQL 文、DBMS_LOB パッケージまたは OCI の操作で、これらを BFILENAME への引数として使用するには、まずディレクトリ・オブジェクトを作成し、BFILE 値を物理ファイルと対応付ける必要があります。

次の 2 つの方法で、このファンクションを使用できます。

- DML 文で BFILE 列を初期化する場合
- プログラム・インタフェースで、BFILE ロケータに値を割り当てることによって BFILE データにアクセスする場合

ディレクトリの引数の大 / 小文字は区別されます。データ・ディクショナリ内に存在する名前と同じ名前でディレクトリ・オブジェクト名を指定しているかを確認する必要があります。たとえば、CREATE DIRECTORY 文で、大 / 小文字を組み合せた識別子を引用符で囲んで使用して Admin ディレクトリ・オブジェクトを作成すると、BFILENAME ファンクションを使用する場合、ディレクトリ・オブジェクトを 'Admin' として指定する必要があります。filename 引数は、ご使用のオペレーティング・システムの大 / 小文字および記号の表記規則に従って指定する必要があります。

参照 :

- LOB の詳細および BFILE データの検出例については、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』および『Oracle Call Interface プログラマーズ・ガイド』を参照してください。
- 「CREATE DIRECTORY」(14-38 ページ)

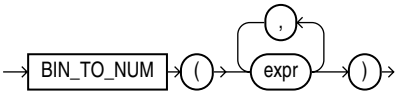
例

次の例では、サンプル表 pm.print_media に行を挿入します。BFILENAME を使用して、$ORACLE_HOME/demo/schema/product_media ディレクトリにある、サーバーのファイル・システムのバイナリ・ファイルを識別します。次の例では、PM スキーマでのディレクトリのデータベース・オブジェクト media_dir の作成方法を示します。

```
CREATE DIRECTORY media_dir AS '/demo/schema/product_media';

INSERT INTO print_media (product_id, ad_id, ad_graphic)
   VALUES (3000, 31001,
      BFILENAME('MEDIA_DIR', 'modem_comp_ad.gif'));
```

BIN_TO_NUM

構文



用途

BIN_TO_NUM は、ビット・ベクトルを同等の数値に変換します。このファンクションの各引数は、ビット・ベクトルのビットを表します。このファンクションは、引数として、任意の数値データ型、または暗黙的に NUMBER に変換可能な数値以外のデータ型を取ります。各 *expr* は、0 または 1 に評価される必要があります。このファンクションは Oracle の NUMBER 値を戻します。

BIN_TO_NUM は、データ・ウェアハウスのアプリケーションで、グルーピング・セットを使用して、マテリアライズド・ビューから対象グループを検索する場合に有効です。

参照 :

- GROUPING SETS 構文の詳細は、19-24 ページの「group_by_clause」を参照してください。
- 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。
- データ集計の詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

例

次の例では、バイナリの値を数値に変換します。

```
SELECT BIN_TO_NUM(1,0,1,0) FROM DUAL;

BIN_TO_NUM(1,0,1,0)
-------------------
                 10
```

次の例では、3 つの値を 1 つのバイナリの値に変換し、BIN_TO_NUM を使用してこのバイナリを数値に変換します。この例では、PL/SQL 宣言を使用して元の値を指定します。これらの値は、通常は実際のデータ・ソースから導出されます。

```
SELECT order_status FROM orders WHERE order_id = 2441;

ORDER_STATUS
------------
           5
DECLARE
  warehouse NUMBER := 1;
  ground    NUMBER := 1;
  insured   NUMBER := 1;
  result    NUMBER;
BEGIN
  SELECT BIN_TO_NUM(warehouse, ground, insured) INTO result FROM DUAL;
  UPDATE orders SET order_status = result WHERE order_id = 2441;
end;
/
PL/SQL procedure successfully completed.

SELECT order_status FROM orders WHERE order_id = 2441;

ORDER_STATUS
------------
           7
```

この逆の（1 つの列値から複数の値を抽出する）プロセスの詳細は、5-23 ページの「BITAND」の例を参照してください。

BITAND

構文

用途

BITAND ファンクションは、その入力と出力をビットのベクトルとして扱います。出力は、入力のビット単位 AND になります。

expr1 および *expr2* の型は NUMBER であり、結果は NUMBER 型になります。BITAND のいずれかの引数が NULL の場合、結果は NULL になります。

引数は、$-(2^{(n-1)})$ .. $((2^{(n-1)})-1)$ の範囲にする必要があります。引数をこの範囲外にすると、結果は未定義になります。

結果は、いくつかのステップで計算されます。まず、それぞれの引数 A が値 SIGN(A)*FLOOR(ABS(A)) に置き換えられます。この変換では、各引数の小数点以下が切り捨てられます。次に、整数値となったそれぞれの引数 A が、n ビットの 2 の補数のバイナリ整数値に変換されます。ビット単位 AND 演算を使用して、2 つのビット値が組み合せられます。最後に、生成された n ビットの 2 の補数値が NUMBER に変換されます。

BITAND ファンクションの注意事項

- BITAND の現行の実装では、*n* = 128 が定義されています。
- PL/SQL は、入力と結果の型がすべて BINARY_INTEGER で、*n* = 32 の BITAND のオーバーロードをサポートしています。

例

次の例では、数値 6（バイナリ 1,1,0）と数値 3（バイナリ 0,1,1）に AND 演算を実行します。

```
SELECT BITAND(6,3) FROM DUAL;

BITAND(6,3)
-----------
          2
```

これは、6 と 3 のバイナリの値を示す次の例と同じです。BITAND ファンクションは、バイナリの値の有効桁のみで演算を実行します。

```
SELECT BITAND(
   BIN_TO_NUM(1,1,0),
   BIN_TO_NUM(0,1,1)) "Binary"
FROM DUAL;

    Binary
----------
         2
```

単一の列値における複数の値のエンコードの詳細は、5-22 ページの「BIN_TO_NUM」の例を参照してください。

次の例では、サンプル表 oe.orders の *order_status* 列で複数の選択項目を 1 つの数値内の個別のビットとしてエンコードします。たとえば、まだ倉庫にある発注は、バイナリの値 001（10 進値 1）で表されます。陸上輸送で発送される発注は、バイナリの値 010（10 進値 2）で表されます。保険がかけられた荷物は、バイナリの値 100（10 進値 4）で表されます。この例では、DECODE ファンクションを使用して、*order_status* 値の 3 つのビットのそれぞれに 2 つの値（ビットがオンの場合の値とビットがオフの場合の値）が示されています。

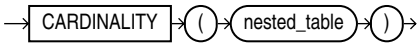
```
SELECT order_id, customer_id, order_status,
  DECODE(BITAND(order_status, 1), 1, 'Warehouse', 'PostOffice')
      "Location",
  DECODE(BITAND(order_status, 2), 2, 'Ground', 'Air') "Method",
  DECODE(BITAND(order_status, 4), 4, 'Insured', 'Certified') "Receipt"
  FROM orders
  WHERE sales_rep_id = 160
  ORDER BY order_id;

  ORDER_ID CUSTOMER_ID ORDER_STATUS Location   Method Receipt
---------- ----------- ------------ ---------- ------ ---------
      2455         145            7 Warehouse  Ground Insured
      2416         104            6 PostOffice Ground Insured
      2419         107            3 Warehouse  Ground Certified
      2420         108            2 PostOffice Ground Certified
      2423         145            3 Warehouse  Ground Certified
      2441         106            5 Warehouse  Air    Insured
```

Location 列では、まず BITAND によって *order_status* と 1（バイナリ 001）が比較されます。比較されるのは重要なビット値のみです。したがって、右端のビットが 1 であるバイナリの値（奇数）は正と評価され、1 が戻されます。偶数の場合は 0（ゼロ）が戻されます。DECODE ファンクションでは、BITAND によって戻された値と 1 が比較されます。両方とも 1 の場合、場所は Warehouse（倉庫）になります。これらの値が異なる場合、場所は PostOffice（郵便局）になります。

Method 列と Receipt 列は、同様に計算されます。Method では、BITAND によって *order_status* と 2（バイナリ 010）に AND 演算が実行されます。Receipt では、BITAND によって *order_status* と 4（バイナリ 100）に AND 演算が実行されます。

CARDINALITY

構文



用途

CARDINALITY は、ネストした表内の要素数を戻します。戻り型は、NUMBER です。ネストした表が空であるか NULL の集合である場合、CARDINALITY は NULL を戻します。

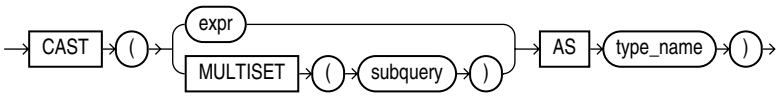
例

次の例では、pm.print_media サンプル表のネストした表の列 ad_textdocs_ntab 内の要素数を示します。

```
SELECT product_id, CARDINALITY(ad_textdocs_ntab) Cardinality
   FROM print_media
   ORDER BY product_id, cardinality;

PRODUCT_ID CARDINALITY
---------- -----------
      2056           3
      2268           3
      3060           3
      3106           3
```

CAST

構文



用途

CAST は、ある組込みデータ型またはコレクション型の値を、別の組込みデータ型またはコレクション型の値に変換します。

CAST を使用すると、ある組込みデータ型またはコレクション型の値を、別の組込みデータ型またはコレクション型に変換できます。名前のないオペランド（日付や副問合せの結果セットなど）または名前付きのコレクション（VARRAY やネストした表など）を型互換の名前付きコレクションにキャストできます。*type_name* は、組込みデータ型またはコレクション型の名前である必要があり、オペランドは、組込みデータ型であるか、またはその値がコレクション値である必要があります。

オペランドでは、*expr* は組込みデータ型、コレクション型または ANYDATA 型のインスタンスのいずれかです。*expr* が ANYDATA 型のインスタンスである場合、CAST は ANYDATA インスタンスの値を抽出し、その値がキャスト対象の型と一致する場合はその値を戻し、一致しない場合は NULL を戻します。MULTISET は、副問合せの結果セットを取り、コレクション値を戻すように Oracle Database に通知します。表 5-1 に、どの組込みデータ型が、どの組込みデータ型にキャストできるかを示します（CAST は、LONG 型、LONG RAW 型、または Oracle が提供する型をサポートしていません）。

CAST は、LOB データ型のいずれも直接的にサポートしていません。CAST を使用して、CLOB 値を文字データ型に変換するか、BLOB 値を RAW データ型に変換すると、データベースは暗黙的に LOB 値を文字または RAW データに変換し、結果値を明示的にターゲットのデータ型にキャストします。結果値がターゲットの型より大きい場合、エラーが戻されます。

CAST ... MULTISET を使用してコレクション値を取得すると、CAST ファンクションに渡される問合せ内の SELECT 構文のリストのアイテムは、ターゲットのコレクション要素型に対応する属性型に変換されます。

表 5-1 組込みデータ型のキャスト

	BINARY_FLOAT、BINARY_DOUBLE から	CHAR、VARCHAR2 から	NUMBER から	DATETIME/INTERVAL から（注 1）	RAW から	ROWID、UROWID から（注 2）	NCHAR、NVARCHAR2 から
BINARY_FLOAT、BINARY_DOUBLE へ	X	X	X	--	--	--	X
CHAR、VARCHAR2 へ	X	X	X	X	X	X	--
NUMBER へ	X	X	X	--	--	--	X
DATE、TIMESTAMP、INTERVAL へ	--	X	--	X	--	--	--
RAW へ	--	X	--	--	X	--	--
ROWID、UROWID へ	--	X	--	--	--	Xa	--
NCHAR、NVARCHAR2 へ	X	--	X	X	X	X	X

注 1: Datetime/Interval には、DATE、TIMESTAMP、TIMESTAMP WITH TIMEZONE、INTERVAL DAY TO SECOND および INTERVAL YEAR TO MONTH が含まれます。

注 2: UROWID が索引構成表の ROWID の値を含んでいる場合、UROWID を ROWID にキャストすることはできません。

名前付きコレクション型を別の名前付きコレクション型にキャストするには、両方のコレクションの要素が同じ型である必要があります。

> **参照：** Oracle Database で暗黙的にコレクション型データを文字データに変換する方法については、2-39 ページの「暗黙的なデータ変換」および 2-43 ページの「データ変換のセキュリティ上の考慮事項」を参照してください。

副問合せの結果セットが複数行に評価される可能性がある場合は、MULTISET キーワードを指定する必要があります。副問合せの結果である行は、それらの行がキャストされたコレクション値の要素を形成します。MULTISET キーワードを省略すると、副問合せはスカラー副問合せとして処理されます。

組込みデータ型の例

次の例では、CAST ファンクションをスカラー・データ型とともに使用します。

```
SELECT CAST('22-OCT-1997' AS TIMESTAMP WITH LOCAL TIME ZONE)
   FROM dual;

SELECT product_id,
   CAST(ad_sourcetext AS VARCHAR2(30)) Text
   FROM print_media
   ORDER BY product_id, text;
```

コレクションの例

後に続く CAST の例は、サンプルの注文入力スキーマ oe で使用されている cust_address_typ に基づいています。

```
CREATE TYPE address_book_t AS TABLE OF cust_address_typ;
/
CREATE TYPE address_array_t AS VARRAY(3) OF cust_address_typ;
/
CREATE TABLE cust_address (
   custno           NUMBER,
   street_address   VARCHAR2(40),
   postal_code      VARCHAR2(10),
   city             VARCHAR2(30),
   state_province   VARCHAR2(10),
   country_id       CHAR(2));

CREATE TABLE cust_short (custno NUMBER, name VARCHAR2(31));

CREATE TABLE states (state_id NUMBER, addresses address_array_t);
```

次の例では、副問合せをキャストします。

```
SELECT s.custno, s.name,
   CAST(MULTISET(SELECT ca.street_address,
                        ca.postal_code,
                        ca.city,
                        ca.state_province,
                        ca.country_id
                 FROM cust_address ca
                 WHERE s.custno = ca.custno)
   AS address_book_t)
FROM cust_short s
ORDER BY s.custno, s.name;
```

CAST では、VARRAY 型の列をネストした表に変換します。

```
SELECT CAST(s.addresses AS address_book_t)
   FROM states s
   WHERE s.state_id = 111;
```

次の例では、この後に示す例で使用するオブジェクトを作成します。

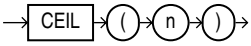
```
CREATE TABLE projects
   (employee_id NUMBER, project_name VARCHAR2(10));

CREATE TABLE emps_short
   (employee_id NUMBER, last_name VARCHAR2(10));

CREATE TYPE project_table_typ AS TABLE OF VARCHAR2(10);
   /
```

次の MULTISET 式の例は、前述のオブジェクトを使用します。

```
SELECT e.last_name,
   CAST(MULTISET(SELECT p.project_name
   FROM projects p
   WHERE p.employee_id = e.employee_id
   ORDER BY p.project_name)
   AS project_table_typ)
FROM emps_short e
ORDER BY e.last_name;
```

CEIL

構文



用途

CEIL は、*n* 以上の最も小さい整数を戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

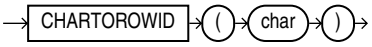
例

次の例では、指定した注文の合計以上である最小の整数を戻します。

```
SELECT order_total, CEIL(order_total) FROM orders
   WHERE order_id = 2434;

ORDER_TOTAL CEIL(ORDER_TOTAL)
----------- -----------------
   268651.8            268652
```

CHARTOROWID

構文



用途

CHARTOROWID は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型の値を ROWID データ型に変換します。

このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。

> **参照：** 詳細は、2-36 ページの「データ型の比較規則」を参照してください。

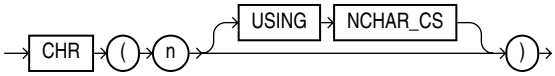
例

次の例では、文字表記の ROWID を ROWID に変換します。（実際の ROWID は、各データベース・インスタンスによって異なります。）

```
SELECT last_name FROM employees
   WHERE ROWID = CHARTOROWID('AAAFd1AAFAAAABSAA/');

LAST_NAME
-------------------------
Greene
```

CHR

構文



用途

CHR は、データベース・キャラクタ・セットまたは各国語キャラクタ・セット（USING NCHAR_CS を指定している場合）の中の *n* に等しい 2 進数を持つ文字を、VARCHAR2 値として戻します。

シングルバイト・キャラクタ・セットの場合、Oracle Database は、n > 256 に対して n mod 256 に等しい 2 進数を戻します。マルチバイト・キャラクタ・セットの場合、n は 1 つのコードポイント全体として解決される必要があります。無効なコードポイントは検証されないため、無効なコードポイントを指定した場合の結果は、予測不能です。

このファンクションは、引数として、NUMBER 値、または暗黙的に NUMBER 型に変換可能な任意の値を取り、文字を戻します。

注意：（オプションの USING NCHAR_CS 句の有無にかかわらず）CHR ファンクションを使用すると、ASCII および EBCDIC ベースのマシン・アーキテクチャ間で移植不可能なコードが戻されます。

参照： 5-104 ページの「NCHR」を参照してください。暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

例

次の例は、データベース・キャラクタ・セットが WE8ISO8859P1 と定義されている ASCII ベースのマシンで実行されています。

```
SELECT CHR(67)||CHR(65)||CHR(84) "Dog" FROM DUAL;

Dog
---
CAT
```

キャラクタ・セットが WE8EBCDIC1047 の EBCDIC ベースのマシンでも同じ結果を戻すには、前述の例を次のように修正する必要があります。

```
SELECT CHR(195)||CHR(193)||CHR(227) "Dog"
   FROM DUAL;

Dog
---
CAT
```

マルチバイト・キャラクタ・セットの場合、このように連結しても異なる結果となります。たとえば、a1a2（a1 は 1 つ目のバイト、a2 は 2 つ目のバイト）という 16 進数の値を持つマルチバイト文字の場合、n に対して「a1a2」に等しい 2 進数または 41378 を指定する必要があります。

```
SELECT CHR(41378) FROM DUAL;
```

次の例のように、a2 に等しい 2 進数と連結された a1 に等しい 2 進数を指定することはできません。

```
SELECT CHR(161)||CHR(162) FROM DUAL;
```

ただし、次の例のように、マルチバイト文字を連結するマルチバイトのコードポイント全体を連結することは可能です。この例では、a1a2 と a1a3 の 16 進数を持つマルチバイト文字を連結しています。

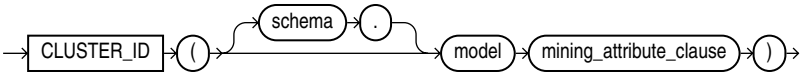
```
SELECT CHR(41378)||CHR(41379) FROM DUAL;
```

次の例では、各国語キャラクタ・セットが UTF16 であると仮定します。

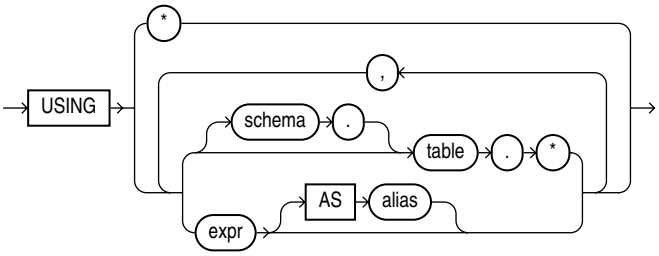
```
SELECT CHR (196 USING NCHAR_CS) FROM DUAL;

CH
--
Ä
```

CLUSTER_ID

構文



***mining_attribute_clause*::=**



用途

このファンクションは、DBMS_DATA_MINING パッケージまたは Oracle Data Mining Java API を使用して作成したモデルのクラスタリングで使用するためのものです。このファンクションは、*mining_attribute_clause* で指定した一連の予測子のうち、最も確率の高い予測クラスタのクラスタ識別子を戻します。戻り値は、Oracle の NUMBER です。

mining_attribute_clause は、PREDICTION ファンクションと同様に動作します。詳細は、5-126 ページの「mining_attribute_clause」を参照してください。

参照：

- Oracle Data Mining の詳細は、『Oracle Data Mining 概要』を参照してください。
- コードで使用可能なデモ・プログラムの詳細は、『Oracle Data Mining 管理者ガイド』を参照してください。
- SQL データ・マイニング・ファンクションを使用したリアルタイム・スコアリングの詳細は、『Oracle Data Mining アプリケーション開発者ガイド』を参照してください。
- 「PREDICTION」（5-125 ページ）

例

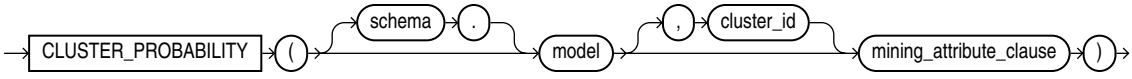
次の例では、指定したデータセットの顧客がグループ化されたクラスタを示します。

この例と前提条件のデータ・マイニング操作（dm_sh_clus_sample モデルおよび dm_sh_sample_apply_prepared ビューの作成など）は、デモ・ファイル $ORACLE_HOME/rdbms/demo/dmkmdemo.sql で確認できます。データ・マイニングのデモ・ファイルの一般情報は、『Oracle Data Mining 管理者ガイド』を参照してください。次に、このファンクションの構文の使用例を示します。

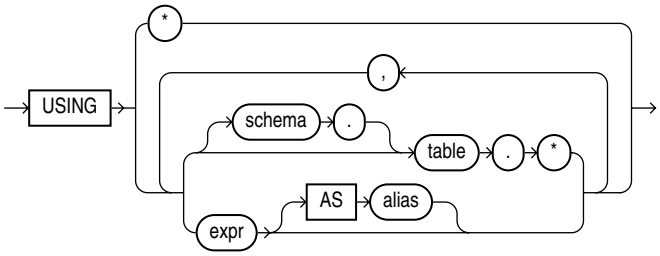
```
SELECT CLUSTER_ID(km_sh_clus_sample USING *) AS clus, COUNT(*) AS cnt
  FROM km_sh_sample_apply_prepared
GROUP BY CLUSTER_ID(km_sh_clus_sample USING *)
ORDER BY cnt DESC;

      CLUS        CNT
---------- ----------
         2        580
        10        199
         6        185
         8        115
        12         98
        16         82
        19         81
        15         68
        18         65
        14         27

10 rows selected.
```

CLUSTER_PROBABILITY

構文



***mining_attribute_clause*::=**



用途

このファンクションは、DBMS_DATA_MINING パッケージまたは Oracle Data Mining Java API を使用して作成したモデルのクラスタリングで使用するためのものです。このファンクションは、指定したモデルに関連付けられたクラスタにある入力行のメンバーシップの確信度のメジャーを戻します。

- *cluster_id* には、モデル内のクラスタの識別子を指定します。このファンクションでは、指定したクラスタの確率を戻します。この句を指定しない場合、最適な予測クラスタに関連付けられた確率が戻ります。*cluster_id* と CLUSTER_ID ファンクションの組合せのない形式を使用して、クラスタ ID と確率の最適な予測の組を取得できます。

- *mining_attribute_clause* は、PREDICTION ファンクションと同様に動作します。詳細は、5-126 ページの「mining_attribute_clause」を参照してください。

参照：

- Oracle Data Mining の詳細は、『Oracle Data Mining 概要』を参照してください。
- コードで使用可能なデモ・プログラムの詳細は、『Oracle Data Mining 管理者ガイド』を参照してください。
- SQL データ・マイニング・ファンクションを使用したリアルタイム・スコアリングの詳細は、『Oracle Data Mining アプリケーション開発者ガイド』を参照してください。
- 関連するデータ・マイニング・ファンクションの詳細は、5-30 ページの「CLUSTER_ID」および 5-125 ページの「PREDICTION」を参照してください。

例

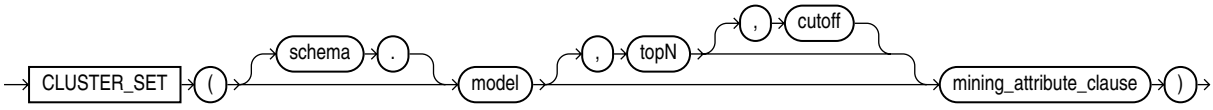
次の例では、可能性に基づいて、クラスタ 2 で最も代表的な顧客を 10 人決定します。

この例と前提条件のデータ・マイニング操作（dm_sh_clus_sample モデルおよび dm_sh_sample_apply_prepared ビューの作成など）は、デモ・ファイル $ORACLE_HOME/rdbms/demo/dmkmdemo.sql で確認できます。データ・マイニングのデモ・ファイルの一般情報は、『Oracle Data Mining 管理者ガイド』を参照してください。次に、このファンクションの構文の使用例を示します。

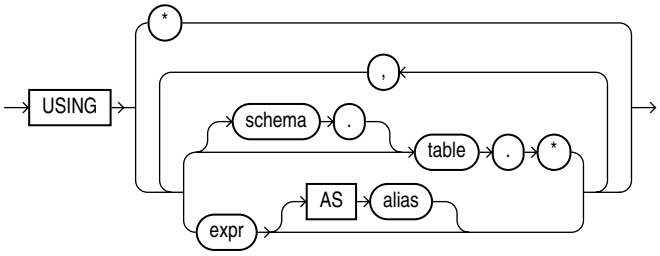
```
SELECT *
  FROM (SELECT cust_id, CLUSTER_PROBABILITY(km_sh_clus_sample, 2 USING *) prob
          FROM km_sh_sample_apply_prepared
        ORDER BY prob DESC)
 WHERE ROWNUM < 11;

   CUST_ID   PROB
---------- ------
    100052  .9993
    100962  .9993
    101208  .9993
    100281  .9993
    100012  .9993
    101009  .9992
    100173  .9992
    101176  .9991
    100672  .9991
    101420  .9991

10 rows selected.
```

CLUSTER_SET

構文



***mining_attribute_clause*::=**



用途

このファンクションは、DBMS_DATA_MINING パッケージまたは Oracle Data Mining Java API を使用して作成したモデルのクラスタリングで使用するためのものです。このファンクションは、指定した行が属している可能性のあるクラスタを含むオブジェクトの VARRAY を戻します。VARRAY の各オブジェクトは、クラスタ ID とクラスタ確率を含むスカラー値の組です。オブジェクト・フィールドには、CLUSTER_ID と PROBABILITY の名前が付き、両方とも Oracle の NUMBER になります。

- オプションの *topN* 引数には、正の整数を指定します。指定すると、予測クラスタの集合は、上位 N の確率値のいずれか持つ予測クラスタに制限されます。*topN* を指定しないか、NULL に設定すると、すべてのクラスタはコレクションに戻されます。複数のクラスタが N 番目の値に一致しても、N 個の値のみが戻されます。
- オプションの *cutoff* 引数には、正の整数を指定し、戻されるクラスタを指定したカットオフ以上の確率を持つクラスタに制限します。NULL を *topN* に指定し、必要なカットオフ値を *cutoff* に指定すると、*cutoff* のみをフィルタ処理できます。

topN と *cutoff* をともに指定すると、戻されるクラスタを上位 N の中で、しきい値を超える確率を持つクラスタに制限できます。

mining_attribute_clause は、PREDICTION ファンクションと同様に動作します。詳細は、5-126 ページの「mining_attribute_clause」を参照してください。

参照 :

- Oracle Data Mining の詳細は、『Oracle Data Mining 概要』を参照してください。
- コードで使用可能なデモ・プログラムの詳細は、『Oracle Data Mining 管理者ガイド』を参照してください。
- SQL データ・マイニング・ファンクションを使用したリアルタイム・スコアリングの詳細は、『Oracle Data Mining アプリケーション開発者ガイド』を参照してください。

例

次の例では、20% を超える可能性で顧客 101362 が属する各クラスタで、最も関連性の高い属性（確信度が 55% を超える）を示します。

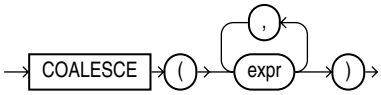
この例と前提条件のデータ・マイニング操作（dm_sh_clus_sample モデル、ビューおよび型の作成など）は、デモ・ファイル $ORACLE_HOME/rdbms/demo/dmkmdemo.sql で確認できます。データ・マイニングのデモ・ファイルの一般情報は、『Oracle Data Mining 管理者ガイド』を参照してください。次に、このファンクションの構文の使用例を示します。

```
WITH
clus_tab AS (
SELECT id,
       A.attribute_name aname,
       A.conditional_operator op,
       NVL(A.attribute_str_value,
```

```
        ROUND(DECODE(A.attribute_name, N.col,
                     A.attribute_num_value * N.scale + N.shift,
                     A.attribute_num_value),4)) val,
       A.attribute_support support,
       A.attribute_confidence confidence
  FROM TABLE(DBMS_DATA_MINING.GET_MODEL_DETAILS_KM('km_sh_clus_sample')) T,
       TABLE(T.rule.antecedent) A,
       km_sh_sample_norm N
 WHERE A.attribute_name = N.col (+) AND A.attribute_confidence > 0.55
),
clust AS (
SELECT id,
       CAST(COLLECT(Cattr(aname, op, TO_CHAR(val), support, confidence))
         AS Cattrs) cl_attrs
  FROM clus_tab
GROUP BY id
),
custclus AS (
SELECT T.cust_id, S.cluster_id, S.probability
  FROM (SELECT cust_id, CLUSTER_SET(km_sh_clus_sample, NULL, 0.2 USING *) pset
          FROM km_sh_sample_apply_prepared
         WHERE cust_id = 101362) T,
       TABLE(T.pset) S
)
SELECT A.probability prob, A.cluster_id cl_id,
       B.attr, B.op, B.val, B.supp, B.conf
  FROM custclus A,
       (SELECT T.id, C.*
          FROM clust T,
               TABLE(T.cl_attrs) C) B
 WHERE A.cluster_id = B.id
ORDER BY prob DESC, cl_id ASC, conf DESC, attr ASC, val ASC;

   PROB      CL_ID ATTR            OP  VAL                  SUPP    CONF
------- ---------- --------------- --- --------------- ---------- -------
  .7873          8 HOUSEHOLD_SIZE  IN  9+                    126   .7500
  .7873          8 CUST_MARITAL_ST IN  Divorc.                118   .6000
                   ATUS

  .7873          8 CUST_MARITAL_ST IN  NeverM                 118   .6000
                   ATUS

  .7873          8 CUST_MARITAL_ST IN  Separ.                 118   .6000
                   ATUS

  .7873          8 CUST_MARITAL_ST IN  Widowed                118   .6000
                   ATUS

  .2016          6 AGE             >=  17                     152   .6667
  .2016          6 AGE             <=  31.6                   152   .6667
  .2016          6 CUST_MARITAL_ST IN  NeverM                 168   .6667
                   ATUS

8 rows selected.
```

COALESCE

構文



用途

COALESCE は、式のリストの最初の NULL でない *expr* を戻します。2 つ以上の式を指定する必要があります。すべての *expr* が NULL と評価された場合、このファンクションは NULL を戻します。

Oracle Database では、**短絡評価**を使用します。データベースは、NULL かどうかを判断する前に *expr* 値のすべてを評価するのではなく、各 *expr* 値を評価して、NULL かどうかを判断します。

すべての *expr* が数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型である場合、Oracle Database は、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。数値の優先順位の詳細は、2-15 ページの「数値の優先順位」を参照してください。

このファンクションは NVL ファンクションを一般化したファンクションです。

COALESCE は、CASE 式の変形として使用できます。次に例を示します。

```
COALESCE (expr1, expr2)
```

これは、次の CASE 式と同じです。

```
CASE WHEN expr1 IS NOT NULL THEN expr1 ELSE expr2 END
```

次の例も同様です。

```
COALESCE (expr1, expr2, ..., exprn), for n>=3
```

これは、次の CASE 式と同じです。

```
CASE WHEN expr1 IS NOT NULL THEN expr1
   ELSE COALESCE (expr2, ..., exprn) END
```

> **参照：** 5-115 ページの「NVL」および 6-5 ページの「CASE 式」を参照してください。

例

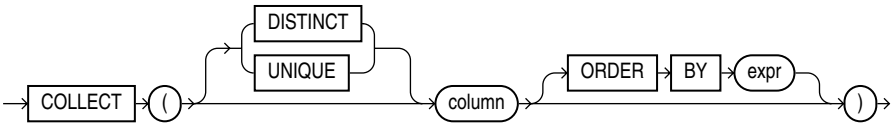
次の例では、サンプル表 oe.product_information を使用して、製品のクリアランス・セールを企画します。製品の表示価格から 10% 値引きします。表示価格がない場合は、最小価格はセール価格となります。最小価格がない場合、セール価格は「5」となります。

```
SELECT product_id, list_price, min_price,
   COALESCE(0.9*list_price, min_price, 5) "Sale"
   FROM product_information
   WHERE supplier_id = 102050
   ORDER BY product_id, list_price, min_price, "Sale";

PRODUCT_ID LIST_PRICE  MIN_PRICE       Sale
---------- ---------- ---------- ----------
      1769         48                 43.2
      1770                    73         73
```

```
      2378        305        247      274.5
      2382        850        731        765
      3355                                 5
```

COLLECT

構文



用途

COLLECT は、任意の型の列を引数に取り、選択された行から、入力された型のネストした表を作成する集計ファンクションです。このファンクションから正確な結果を取得するには、このファンクションを CAST ファンクション内で使用する必要があります。

column 自体がコレクションである場合、COLLECT の出力はコレクションのネストした表になります。*column* がユーザー定義型である場合は、オプションの DISTINCT、UNIQUE および ORDER BY 句を使用できるように、*column* に MAP または ORDER メソッドが定義されている必要があります。

参照： 「CAST」（5-25 ページ）

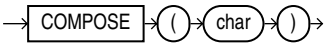
例

次の例では、oe.customers サンプル表の電話番号の VARRAY 列からネストした表を作成します。

```
CREATE TYPE phone_book_t AS TABLE OF phone_list_typ;
/
SELECT CAST(COLLECT(phone_numbers) AS phone_book_t) Phone_Book
   FROM customers
   ORDER BY phone_book;
```

COMPOSE

構文



用途

COMPOSE は、引数として任意のデータ型の文字列または文字列に変換する式を取り、入力されたものと同じキャラクタ・セットで Unicode 文字列を戻します。*char* のデータ型は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB です。たとえば、o ウムラウト・コードポイントは、ウムラウト・コードポイントによって修飾された o コードポイントとして戻されます。

COMPOSE は、NFC 正規形の文字列を戻します。より排他的な設定の場合は、まず CANONICAL 設定を使用して DECOMPOSE を呼び出し、次に COMPOSE を呼び出すことができます。この組合せによって、NFKC 正規形の文字列が戻されます。

暗黙的な変換を使用して、CLOB および NCLOB の値がサポートされます。*char* が文字の LOB 値の場合、COMPOSE 演算の前に VARCHAR 値に変換されます。特定の開発環境で、LOB 値のサイズが VARCHAR のサポートする長さを超える場合、この演算は失敗します。

参照：

- Unicode キャラクタ・セットおよびキャラクタ・セマンティクスの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。
- 「DECOMPOSE」（5-54 ページ）

例

次の例は、o ウムラウトのコードポイントを戻します。

```
SELECT COMPOSE ( 'o' || UNISTR('\0308') ) FROM DUAL;

CO
--
ö
```

参照： 「UNISTR」（5-219 ページ）

CONCAT

構文

用途

CONCAT は、*char2* に連結されている *char1* を戻します。*char1* および *char2* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。*char1* と同じキャラクタ・セットの文字列が戻されます。そのデータ型は、引数のデータ型によって決まります。

2 つの異なるデータ型を連結すると、可逆式変換となるデータ型が戻されます。したがって、引数の 1 つが LOB の場合、戻り値は LOB となります。引数の 1 つが各国語データ型の場合は、戻り値は各国語データ型となります。たとえば、次のようになります。

- CONCAT(CLOB, NCLOB) は NCLOB を戻します。
- CONCAT(NCLOB, NCHAR) は NCLOB を戻します。
- CONCAT(NCLOB, CHAR) は NCLOB を戻します。
- CONCAT(NCHAR, CLOB) は NCLOB を戻します。

このファンクションは、連結演算子（||）と同等です。

参照： CONCAT 演算子の詳細は、4-4 ページの「連結演算子」を参照してください。

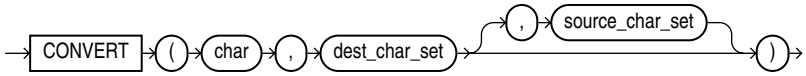
例

次の例では、ネストを使用して 3 つの文字列を連結します。

```
SELECT CONCAT(CONCAT(last_name, '''s job category is '),
      job_id) "Job"
   FROM employees
   WHERE employee_id = 152
   ORDER BY "Job";

Job
----------------------------------------------------
Hall's job category is SA_REP
```

CONVERT

構文



用途

CONVERT は、文字列を、あるキャラクタ・セットから別のキャラクタ・セットに変換します。

- 引数 *char* は変換する値です。char は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。
- 引数 *dest_char_set* は、*char* が変換されるキャラクタ・セットの名前です。
- 引数 *source_char_set* は、*char* をデータベースに格納しているキャラクタ・セットの名前です。デフォルト値はデータベース・キャラクタ・セットです。

CHAR と VARCHAR2 の戻り値は、VARCHAR2 です。NCHAR と NVARCHAR2 の戻り値は、NVARCHAR2 です。CLOB の戻り値は CLOB、NCLOB の戻り値は NCLOB です。

変換先キャラクタ・セットと変換元キャラクタ・セットの引数として、リテラルまたはキャラクタ・セットの名前を含んでいる列を指定できます。

完全に文字を変換するには、変換先キャラクタ・セットが変換元キャラクタ・セットで定義されているすべての文字を表現できる必要があります。文字が変換先キャラクタ・セットに存在しないと、置換文字が使用されます。置換文字は、キャラクタ・セット定義の一部として定義できます。

注意： Oracle Database の現行のリリースでは、CONVERT ファンクションを使用しないことをお薦めします。CONVERT の戻り値は文字データ型であるため、そのデータ型に応じてデータベース・キャラクタ・セットまたは各国語キャラクタ・セットのいずれかになります。この 2 つのキャラクタ・セットのどちらでもない *dest_char_set* は、サポートされません。*char* 引数と *source_char_set* の要件は同じです。このため、ファンクションの実用的な用途は、誤ったキャラクタ・セットで格納されているデータの修正にかぎられます。

データベース・キャラクタ・セットと各国語キャラクタ・セットのどちらでもない値は、RAW または BLOB として処理して格納する必要があります。PL/SQL パッケージ UTL_RAW および UTL_I18N のプロシージャ（たとえば、UTL_RAW.CONVERT）では、このような値の限定的な処理が可能です。パッケージ UTL_FILE、UTL_TCP、UTL_HTTP および UTL_SMTP 内で RAW 引数を受け入れるプロシージャを使用すると、処理されたデータを出力できます。

例

次の例では、Latin-1 文字列を ASCII に変換するキャラクタ・セットの変換を示します。これは、同じ文字列を WE8ISO8859P1 データベースから US7ASCII データベースへインポートした場合と同じ結果が得られます。

```
SELECT CONVERT('Ä Ê Í Õ Ø A B C D E ', 'US7ASCII', 'WE8ISO8859P1')
   FROM DUAL;

CONVERT('ÄÊÍÕØABCDE'
--------------------
A E I ? ? A B C D E ?
```

一般的なキャラクタ・セットを次に示します。

- US7ASCII: US7 ビット ASCII キャラクタ・セット
- WE8ISO8859P1: ISO 8859-1 西ヨーロッパ 8 ビット・キャラクタ・セット
- EE8MSWIN1250: Microsoft Windows 西ヨーロッパ・コード・ページ 1250
- WE8MSWIN1252: Microsoft Windows 西ヨーロッパ・コード・ページ 1252
- WE8EBCDIC1047: IBM 西ヨーロッパ EBCDIC コード・ページ 1047
- JA16SJISTILDE: MS コード・ページ 932 と互換性のある日本語 Shift-JIS キャラクタ・セット
- ZHT16MSWIN950: Microsoft Windows 繁体字中国語コード・ページ 950
- UTF8: Unicode 3.0 ユニバーサル・キャラクタ・セット CESU-8 エンコード形式
- AL32UTF8: Unicode 5.0 ユニバーサル・キャラクタ・セット UTF-8 エンコード形式

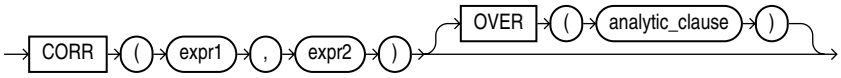
次のように V$NLS_VALID_VALUES ビューを問い合せて、有効なキャラクタ・セットのリストを取得できます。

```
SELECT * FROM V$NLS_VALID_VALUES WHERE parameter = 'CHARACTERSET'
```

> **参照：** サポートされているキャラクタ・セットの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。V$NLS_VALID_VALUES ビューの詳細は、『Oracle Database リファレンス』を参照してください。

CORR

構文



> **参照：** 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

CORR は、数値の組の集合に対する相関係数を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。Oracle は、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。数値の優先順位の詳細は、2-15 ページの「数値の優先順位」を参照してください。

Oracle Database は、*expr1* または *expr2* が NULL である組を排除した後、このファンクションを (*expr1*, *expr2*) の集合に適用します。その後、Oracle は次の計算を行います。

```
COVAR_POP(expr1, expr2) / (STDDEV_POP(expr1) * STDDEV_POP(expr2))
```

ファンクションは、NUMBER 型の値を戻します。ファンクションが空の集合に適用されると、NULL を戻します。

注意： CORR ファンクションは、ピアソンの相関係数を計算します。この計算を行うには、数式を引数として指定する必要があります。Oracle は、ノンパラメトリックまたは順位相関をサポートするための CORR_S（スピアマンのロー係数）および CORR_K（ケンドールのタウ b 係数）も提供します。

参照： *expr* の書式の詳細は、5-8 ページの「集計ファンクション」および 6-2 ページの「SQL 式」を参照してください。5-41 ページの「CORR_*」と 5-42 ページの「CORR_S」も参照してください。

集計の例

次の例では、oe.product_information サンプル表の重さクラスごとの製品の表示価格と最小価格の相関係数を計算します。

```
SELECT weight_class, CORR(list_price, min_price) "Correlation"
   FROM product_information
   GROUP BY weight_class
   ORDER BY weight_class, "Correlation";

WEIGHT_CLASS Correlation
------------ -----------
           1  .999149795
           2  .999022941
           3  .998484472
           4  .999359909
           5  .999536087
```

分析の例

次の例では、会社での勤務年数と給与の相関を従業の役職別に示します。結果セットでは、指定した業務の従業員ごとに同じ相関を示します。

```
SELECT employee_id, job_id,
   TO_CHAR((SYSDATE - hire_date) YEAR TO MONTH ) "Yrs-Mns",     salary,
   CORR(SYSDATE-hire_date, salary)
   OVER(PARTITION BY job_id) AS "Correlation"
FROM employees
WHERE department_id in (50, 80)
ORDER BY job_id, employee_id;

EMPLOYEE_ID JOB_ID     Yrs-Mns     SALARY Correlation
----------- ---------- ------- ---------- -----------
        145 SA_MAN     +08-07       14000  .912385598
        146 SA_MAN     +08-04       13500  .912385598
        147 SA_MAN     +08-02       12000  .912385598
        148 SA_MAN     +05-07       11000  .912385598
        149 SA_MAN     +05-03       10500  .912385598
        150 SA_REP     +08-03       10000   .80436755
        151 SA_REP     +08-02        9500   .80436755
        152 SA_REP     +07-09        9000   .80436755
        153 SA_REP     +07-01        8000   .80436755
        154 SA_REP     +06-05        7500   .80436755
        155 SA_REP     +05-06        7000   .80436755
...
```

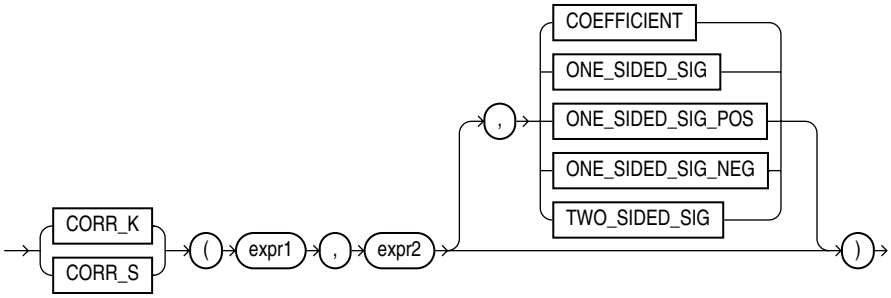
CORR_*

CORR_* ファンクションを次に示します。

- CORR_S
- CORR_K

構文

***correlation*::=**



用途

CORR ファンクション（5-39 ページの「CORR」を参照）は、ピアソンの相関係数を計算し、入力として数式を必要とします。CORR_* ファンクションは、ノンパラメトリックまたは順位相関をサポートします。これらのファンクションを使用すると、式間の相関を順序尺度化して求めることができます（値の順序付けが可能な場合）。相関係数は -1 ～ 1 の範囲の値となります。1 は完全な正相関、-1 は完全な逆相関（一方の変数が減少すると他方の変数が増加する）、および 0（ゼロ）に近い値は無相関を表します。

これらのファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。Oracle Database は、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、計算を実行して NUMBER を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。数値の優先順位の詳細は、2-15 ページの「数値の優先順位」を参照してください。

expr1 および *expr2* は、分析対象の 2 つの変数です。3 つ目の引数は VARCHAR2 型の戻り値です。3 つ目の引数を指定しない場合、デフォルトで COEFFICIENT が戻り値になります。戻り値の意味を次の表に示します。

表 5-2 CORR_* の戻り値

戻り値	意味
COEFFICIENT	相関の係数
ONE_SIDED_SIG	相関の正の片側有意
ONE_SIDED_SIG_POS	ONE_SIDED_SIG に同じ
ONE_SIDED_SIG_NEG	相関の負の片側有意
TWO_SIDED_SIG	相関の両側有意

CORR_S

CORR_S は、スピアマンの順位相関係数（ロー）を計算します。入力式は、観測値の組 (x_i, y_i) の集合である必要があります。このファンクションは、最初に各値を順位に置き換えます。各 x_i の値は、標本内の他のすべての x_i 中での順位に置き換えられ、各 y_i の値は、他のすべての y_i 中での順位に置き換えられます。したがって、各 x_i および各 y_i は 1 ～ *n* の値となります。ここで *n* は、値の組の合計数です。同順位の場合は、値がわずかに異なっていた場合に付けられる順位の平均が割り当てられます。その後このファンクションは、順位の線形相関係数を計算します。

CORR_S の例　次の例では、スピアマンのロー相関係数を使用して、salary と commission_pct、salary と employee_id の 2 つの異なる比較ごとに相関係数を導出します。

```
SELECT COUNT(*) count,
       CORR_S(salary, commission_pct) commission,
       CORR_S(salary, employee_id) empid
FROM employees;

     COUNT COMMISSION      EMPID
---------- ---------- ----------
       107 .735837022 -.04482358
```

CORR_K

CORR_K は、ケンドールの順位相関係数（タウ b）を計算します。CORR_S と同様に、入力式は観測値の組 (x_i, y_i) の集合です。係数を計算するために、このファンクションは一致した組と一致しない組の数をカウントします。x と y の値がいずれも大きい観測値の組は一致しています。x の値が大きく y の値が小さい観測値の組は一致していません。

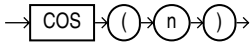
タウ b での有意性は、タウ b によって示される相関が偶然の結果である確率です（0 ～ 1 の値）。この値が小さい場合、タウ b が正の値であれば有意な相関が存在します（タウ b が負の値の場合は逆相関）。

CORR_K の例　次の例では、ケンドールのタウ b 相関係数を使用して、従業員の給与と歩合の割合（パーセント）間に相関があるかどうかを判断します。

```
SELECT CORR_K(salary, commission_pct, 'COEFFICIENT') coefficient,
       CORR_K(salary, commission_pct, 'TWO_SIDED_SIG') two_sided_p_value
  FROM hr.employees;

COEFFICIENT TWO_SIDED_P_VALUE
----------- -----------------
 .603079768        3.4702E-07
```

COS

構文



用途

COS は、*n*（ラジアンで表された角度）のコサインを戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数が BINARY_FLOAT の場合、このファンクションは BINARY_DOUBLE を戻します。それ以外の場合、引数と同じ数値データ型を戻します。

参照： 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

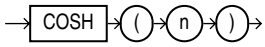
例

次の例では、180 度のコサインを戻します。

```
SELECT COS(180 * 3.14159265359/180)
   "Cosine of 180 degrees" FROM DUAL;

Cosine of 180 degrees
---------------------
                   -1
```

COSH

構文



用途

COSH は、*n* の双曲線コサインを戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数が BINARY_FLOAT の場合、このファンクションは BINARY_DOUBLE を戻します。それ以外の場合、引数と同じ数値データ型を戻します。

参照： 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

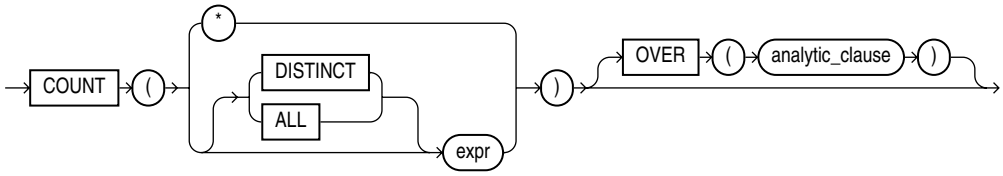
例

次の例では、0 の双曲線コサインを戻します。

```
SELECT COSH(0) "Hyperbolic cosine of 0" FROM DUAL;

Hyperbolic cosine of 0
----------------------
                     1
```

COUNT

構文



参照： 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

COUNT は、問合せによって戻された行の数を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

DISTINCT を指定する場合は、*analytic_clause* の *query_partition_clause* のみ指定できます。*order_by_clause* および *windowing_clause* は指定できません。

expr を指定すると、COUNT は *expr* が NULL でない行数を戻します。*expr* のすべての行を数えるか、または異なる値のみを数えることができます。

アスタリスク（*）を指定すると、このファンクションは重複値および NULL 値を含むすべての行を戻します。COUNT は NULL を戻しません。

> **参照：** *expr* の書式の詳細は、6-2 ページの「SQL 式」を参照してください。5-8 ページの「集計ファンクション」も参照してください。

集計の例

次の例では、COUNT を集計ファンクションとして使用します。

```
SELECT COUNT(*) "Total" FROM employees;

     Total
----------
       107

SELECT COUNT(*) "Allstars" FROM employees
   WHERE commission_pct > 0;

 Allstars
---------
       35

SELECT COUNT(commission_pct) "Count" FROM employees;

     Count
----------
        35

SELECT COUNT(DISTINCT manager_id) "Managers" FROM employees;

  Managers
----------
        18
```

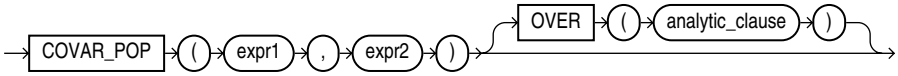
分析の例

次の例では、employees 表の各従業員について、その従業員の給与より 50 ドル少ない金額から 150 ドル多い金額の範囲の給与を得ている従業員の数を計算します。

```
SELECT last_name, salary,
   COUNT(*) OVER (ORDER BY salary RANGE BETWEEN 50 PRECEDING
      AND 150 FOLLOWING) AS mov_count FROM employees
   ORDER BY last_name, salary, COUNT(*);

LAST_NAME                     SALARY  MOV_COUNT
------------------------- ---------- ----------
Olson                           2100          3
Markle                          2200          2
Philtanker                      2200          2
Landry                          2400          8
Gee                             2400          8
Colmenares                      2500         10
Marlow                          2500         10
Patel                           2500         10
. . .
```

COVAR_POP

構文



参照： 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

COVAR_POP は、数値の組の集合に対する母集団共分散を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。Oracle は、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

参照： 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。数値の優先順位の詳細は、2-15 ページの「数値の優先順位」を参照してください。

Oracle Database は、*expr1* または *expr2* が NULL であるすべての組を排除した後、このファンクションを (*expr1*, *expr2*) の組の集合に適用します。その後、Oracle は次の計算を行います。

```
(SUM(expr1 * expr2) - SUM(expr2) * SUM(expr1) / n) / n
```

ここで、*n* は (*expr1*, *expr2*) の組の数です（ただし、*expr1* および *expr2* の両方が NULL ではない場合です）。

ファンクションは、NUMBER 型の値を戻します。ファンクションが空の集合に適用されると、NULL を戻します。

参照： *expr* の書式の詳細は、6-2 ページの「SQL 式」を参照してください。5-8 ページの「集計ファンクション」も参照してください。

集計の例

次の例では、サンプル表 hr.employees を使用して、勤務時間（SYSDATE - hire_date）と給与の母集団共分散と標本共分散を計算します。

```
SELECT job_id,
   COVAR_POP(SYSDATE-hire_date, salary) AS covar_pop,
   COVAR_SAMP(SYSDATE-hire_date, salary) AS covar_samp
   FROM employees
   WHERE department_id in (50, 80)
   GROUP BY job_id
   ORDER BY job_id, covar_pop, covar_samp;

JOB_ID       COVAR_POP  COVAR_SAMP
---------- ----------- -----------
SA_MAN         660700     825875
SA_REP     579988.466  600702.34
SH_CLERK     212432.5 223613.158
ST_CLERK    176577.25 185870.789
ST_MAN         436092     545115
```

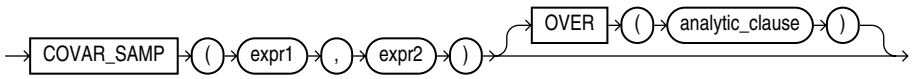
分析の例

次の例では、デモ・スキーマ oe の製品の表示価格および最小価格の累積標本共分散を計算します。

```
SELECT product_id, supplier_id,
   COVAR_POP(list_price, min_price)
      OVER (ORDER BY product_id, supplier_id)
         AS CUM_COVP,
   COVAR_SAMP(list_price, min_price)
      OVER (ORDER BY product_id, supplier_id)
        AS CUM_COVS
   FROM product_information p
   WHERE category_id = 29
   ORDER BY product_id, supplier_id;

PRODUCT_ID SUPPLIER_ID   CUM_COVP   CUM_COVS
---------- ----------- ---------- ----------
      1774      103088          0
      1775      103087    1473.25     2946.5
      1794      103096 1702.77778 2554.16667
      1825      103093    1926.25 2568.33333
      2004      103086     1591.4    1989.25
      2005      103086     1512.5       1815
      2416      103088 1475.97959 1721.97619
. . .
```

COVAR_SAMP

構文



参照： 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

COVAR_SAMP は、数値の組の集合の標本共分散を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。Oracle は、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

参照： 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。数値の優先順位の詳細は、2-15 ページの「数値の優先順位」を参照してください。

Oracle Database は、*expr1* または *expr2* が NULL であるすべての組を排除した後、このファンクションを (*expr1*, *expr2*) の組の集合に適用します。その後、Oracle は次の計算を行います。

```
(SUM(expr1 * expr2) - SUM(expr1) * SUM(expr2) / n) / (n-1)
```

ここで、*n* は (*expr1*, *expr2*) の組の数です（ただし、*expr1* および *expr2* の両方が NULL ではない場合です）。

ファンクションは、NUMBER 型の値を戻します。ファンクションが空の集合に適用されると、NULL を戻します。

> **参照：** *expr* の書式の詳細は、6-2 ページの「SQL 式」を参照してください。5-8 ページの「集計ファンクション」も参照してください。

集計の例

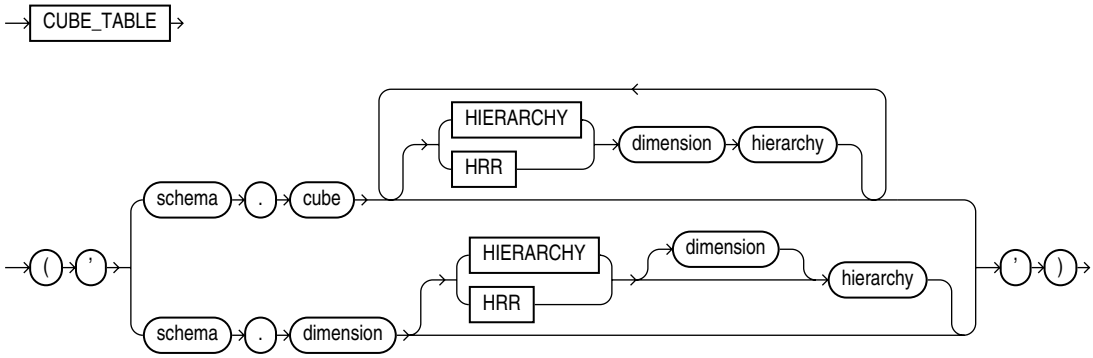
5-45 ページの「COVAR_POP」の集計の例を参照してください。

分析の例

5-45 ページの「COVAR_POP」の分析の例を参照してください。

CUBE_TABLE

構文



用途

CUBE_TABLE は、キューブまたはディメンションのデータを抽出し、SQL ベースのアプリケーションで使用可能な 2 次元形式のリレーショナル表に戻します。

このファンクションは、1 つの VARCHAR2 引数を取ります。オプションの HIERARCHY 句を使用すると、ディメンション階層を指定できます。キューブでは複数の HIERARCHY 句（ディメンションごとに 1 つ）を使用できます。

次の様々なタイプの表を生成できます。

- キューブ表には、各ディメンションのキー列およびキューブ内の各メジャーと計算されたメジャーの列が含まれています。キューブ表を作成するには、キューブを指定します。キューブの HIERARCHY 句は、使用することも使用しないこともできます。複数の階層のあるディメンションの場合は、この句によって戻り値を指定された階層内のディメンション・メンバーとレベルに制限します。HIERARCHY 句を使用しない場合は、すべてのディメンション・メンバーとすべてのレベルが含められます。
- ディメンション表には、キー列、および各レベルと各属性の列が含まれています。この表には、すべてのディメンション・メンバーとすべてのレベルが含められます。ディメンション表を作成するには、ディメンションの HIERARCHY 句を**使用しないで**ディメンションを指定します。
- 階層表には、ディメンション表のすべての列に加えて、親メンバーの列と各ソース・レベルの列が含まれています。指定された階層に含まれないディメンション・メンバーとレベルはすべて、表から除外されます。階層表を作成する場合は、ディメンションの HIERARCHY 句を**使用して**ディメンションを指定します。

CUBE_TABLE は表ファンクションであり、SELECT 文のコンテキストで常に次の構文で使用されます。

```
SELECT ... FROM TABLE(CUBE_TABLE('arg'));
```

> **参照：** ディメンション・オブジェクトの詳細および CUBE_TABLE で生成される表の詳細は、『Oracle OLAP ユーザーズ・ガイド』を参照してください。

例

次の SELECT 文は、GLOBAL スキーマの CHANNEL のディメンション表を生成します。

```
SELECT * FROM TABLE(CUBE_TABLE('global.channel'));

DIM_KEY  LEVEL_NAME      LONG_DESCRIP SHORT_DESCRI TOTAL_CHANNEL_ID CHANNEL_ID
-------- --------------- ------------ ------------ ---------------- ----------
1        TOTAL_CHANNEL   All Channels All Channels               1
2        CHANNEL         Direct Sales Direct Sales               1          2
3        CHANNEL         Catalog      Catalog                    1          3
4        CHANNEL         Internet     Internet                   1          4
```

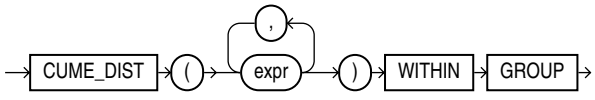
次の文は、UNITS_CUBE のキューブ表を生成します。この文は、表を MARKET_ROLLUP 階層および CALENDAR 階層に制限します。

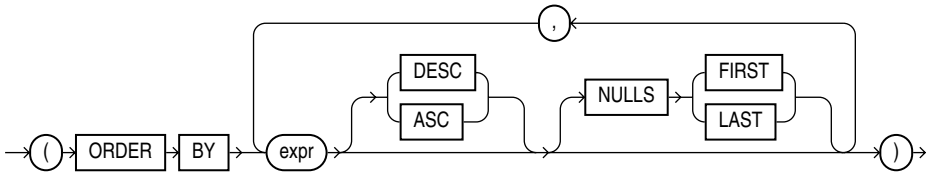
```
SELECT * FROM TABLE(CUBE_TABLE(
   'global.units_cube HIERARCHY customer market_rollup HIERARCHY time calendar'));

     SALES      UNITS       COST TIME     CUSTOMER PRODUCT  CHANNEL
---------- ---------- ---------- -------- -------- -------- --------
 134109248     330425  124918967 2        7        1        1
32275009.5      77425   30255208 10       7        1        1
10768750.7      25780 10058324.5 36       7        1        1
 109261.64        278  101798.32 36       5        1        1
  22371.47         53   20887.54 36       36       1        1
      .
      .
      .
```

CUME_DIST

集計の構文

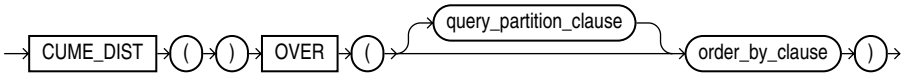
***cume_dist_aggregate*::=**





分析の構文

***cume_dist_analytic*::=**



> **参照：** 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

CUME_DIST は、値のグループにある値の累積分布値を計算します。CUME_DIST が戻す値の範囲は、0 より大きく 1 以下です。連結値は、常に同じ累積分布値に対して評価を行います。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。Oracle Database は、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、計算を実行して NUMBER を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。数値の優先順位の詳細は、2-15 ページの「数値の優先順位」を参照してください。

- 集計ファンクションとしての CUME_DIST は、ファンクションの引数および対応するソート指定によって識別される不確定な行 *r* に対して、集計グループ内の行の中での行 *r* の関連する位置を計算します。Oracle は、不確定な行 *r* を集計される行のグループに挿入するように計算します。このファンクションの引数は、各集計グループ内の 1 つの不確定行を識別します。このため、すべての引数は、集計グループ内で定数式と評価される必要があります。定数引数式および集計の ORDER BY 句の式の位置は、一致します。このため、引数の数は同じであり、その型は互換性がある必要があります。
- 分析ファンクションとしての CUME_DIST は、値のグループにある特定の値の相対位置を計算します。行 *r* について、昇順で順序付けられているとします。*r* の CUME_DIST は、*r* の値以下の値の行数を、評価される行の数（問合せ結果セットまたはパーティション）で割った数です。

集計の例

次の例では、サンプル表 oe.employees の従業員の中から、給与が 15,500 ドルであり、歩合が 5% の不確定な従業員の累積分布を計算します。

```
SELECT CUME_DIST(15500, .05) WITHIN GROUP
   (ORDER BY salary, commission_pct) "Cume-Dist of 15500"
   FROM employees;

Cume-Dist of 15500
------------------
        .972222222
```

分析の例

次の例では、購買部門の各従業員の給与のパーセンタイルを計算します。たとえば、事務員の 40% が、Himuro の給与以下の給与を得ていることがわかります。

```
SELECT job_id, last_name, salary, CUME_DIST()
   OVER (PARTITION BY job_id ORDER BY salary) AS cume_dist
   FROM employees
   WHERE job_id LIKE 'PU%'
   ORDER BY job_id, last_name, salary, cume_dist;
```

```
JOB_ID     LAST_NAME                     SALARY  CUME_DIST
---------- ------------------------- ---------- ----------
PU_CLERK   Baida                           2900         .8
PU_CLERK   Colmenares                      2500         .2
PU_CLERK   Himuro                          2600         .4
PU_CLERK   Khoo                            3100          1
PU_CLERK   Tobias                          2800         .6
PU_MAN     Raphaely                       11000          1
```

CURRENT_DATE

構文

用途

CURRENT_DATE は、セッション・タイムゾーンの現在の日付を DATE データ型のグレゴリオ暦の値で戻します。

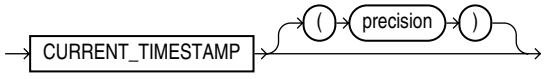
例

次の例では、CURRENT_DATE がセッション・タイムゾーンによって異なることを示します。

```
ALTER SESSION SET TIME_ZONE = '-5:0';
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;

SESSIONTIMEZONE CURRENT_DATE
--------------- --------------------
-05:00          29-MAY-2000 13:14:03

ALTER SESSION SET TIME_ZONE = '-8:0';
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;

SESSIONTIMEZONE CURRENT_DATE
--------------- --------------------
-08:00          29-MAY-2000 10:14:33
```

CURRENT_TIMESTAMP

構文



用途

CURRENT_TIMESTAMP は、セッション・タイムゾーンの現在の日付および時刻を TIMESTAMP WITH TIME ZONE データ型の値で戻します。タイムゾーン・オフセットは、SQL セッションの現在のローカル時刻を反映します。精度の指定を省略した場合のデフォルトは 6 です。このファンクションと LOCALTIMESTAMP との違いは、CURRENT_TIMESTAMP は、TIMESTAMP WITH TIME ZONE の値を戻し、LOCALTIMESTAMP は TIMESTAMP の値を戻す点です。

オプションの引数では、*precision* は、戻される時刻の値の小数秒の精度を指定します。

参照： 「LOCALTIMESTAMP」（5-94 ページ）

例

次の例では、CURRENT_TIMESTAMP がセッション・タイムゾーンによって異なることを示します。

```
ALTER SESSION SET TIME_ZONE = '-5:0';
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
SELECT SESSIONTIMEZONE, CURRENT_TIMESTAMP FROM DUAL;

SESSIONTIMEZONE CURRENT_TIMESTAMP
--------------- ---------------------------------------------------
-05:00          04-APR-00 01.17.56.917550 PM -05:00

ALTER SESSION SET TIME_ZONE = '-8:0';
SELECT SESSIONTIMEZONE, CURRENT_TIMESTAMP FROM DUAL;

SESSIONTIMEZONE CURRENT_TIMESTAMP
--------------- ---------------------------------------------------
-08:00          04-APR-00 10.18.21.366065 AM -08:00
```

CURRENT_TIMESTAMP で書式マスクを使用する場合は、ファンクションが戻す値と書式マスクを一致させてください。たとえば、次の表の場合を考えます。

```
CREATE TABLE current_test (col1 TIMESTAMP WITH TIME ZONE);
```

ファンクションが戻す型の TIME ZONE の部分がマスクに含まれていないため、次の文は正常に実行されません。

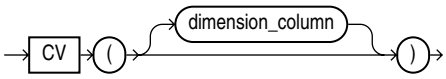
```
INSERT INTO current_test VALUES
  (TO_TIMESTAMP_TZ(CURRENT_TIMESTAMP, 'DD-MON-RR HH.MI.SSXFF PM'));
```

次の文では、CURRENT_TIMESTAMP の戻り値の型と一致する正しい書式マスクが使用されています。

```
INSERT INTO current_test VALUES (TO_TIMESTAMP_TZ
    (CURRENT_TIMESTAMP, 'DD-MON-RR HH.MI.SSXFF PM TZH:TZM'));
```

CV

構文



用途

CV ファンクションは、SELECT 文の *model_clause* でのみ、かつモデル・ルールの右側でのみ使用できます。戻り値は、ルールの左側から右側に送られたディメンション列またはパーティション列の現在の値です。このファンクションを *model_clause* で使用するとディメンション列に対する相対索引を作成できます。戻り型は、ディメンション列のデータ型です。引数を指定しない場合、セル参照内のファンクションの相対位置に対応付けられたディメンション列がデフォルトで使用されます。

CV ファンクションは、セル参照外でも使用できます。その場合は、*dimension_column* が必要です。

> **参照：** 構文およびセマンティクスの詳細は、19-25 ページの「model_clause」および 6-11 ページの「モデル式」を参照してください。

例

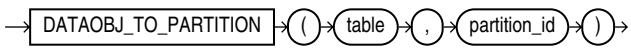
次の例では、ディメンション列（マウス・パッドまたはスタンダード・マウス）の 1999 年および 2000 年の現在の値が表す製品の売上の合計を、その製品の 2001 年の売上に割り当てます。

```
SELECT country, prod, year, s
  FROM sales_view_ref
  MODEL
    PARTITION BY (country)
    DIMENSION BY (prod, year)
    MEASURES (sale s)
    IGNORE NAV
    UNIQUE DIMENSION
    RULES UPSERT SEQUENTIAL ORDER
    (
      s[FOR prod IN ('Mouse Pad', 'Standard Mouse'), 2001] =
        s[CV( ), 1999] + s[CV( ), 2000]
    )
  ORDER BY country, prod, year;

COUNTRY       PROD                                    YEAR          S
----------    ----------------------------------      --------   ---------
France        Mouse Pad                                  1998     2509.42
France        Mouse Pad                                  1999     3678.69
France        Mouse Pad                                  2000     3000.72
France        Mouse Pad                                  2001     6679.41
France        Standard Mouse                             1998     2390.83
France        Standard Mouse                             1999     2280.45
France        Standard Mouse                             2000     1274.31
France        Standard Mouse                             2001     3554.76
Germany       Mouse Pad                                  1998     5827.87
Germany       Mouse Pad                                  1999     8346.44
Germany       Mouse Pad                                  2000     7375.46
Germany       Mouse Pad                                  2001     15721.9
Germany       Standard Mouse                             1998     7116.11
Germany       Standard Mouse                             1999     6263.14
Germany       Standard Mouse                             2000     2637.31
Germany       Standard Mouse                             2001     8900.45

16 rows selected.
```

この例では、ビュー sales_view_ref が必要です。このビューを作成する方法については、19-36 ページの「MODEL 句の例 :」を参照してください。

DATAOBJ_TO_PARTITION

構文



用途

DATAOBJ_TO_PARTITION は、ドメイン索引データの格納に使用するシステム・パーティション表でデータ・メンテナンスまたは問合せ操作を実行するデータ・カートリッジ開発者にのみ役立ちます。DML 操作や問合せ操作は、ドメイン索引の実表での対応する操作によってトリガーされます。

このファンクションは、引数として実表の名前と実表のパーティションのパーティション ID を取ります。これらはともに、適切な ODCIIndex メソッドによってファンクションに渡されます。ファンクションは、対応するシステムパーティション表のパーティション ID を戻します。

この ID は、システム・パーティション表の該当パーティションで操作（DML または問合せ）を実行する際に使用できます。

> **参照：** このファンクションの使用の詳細および例は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。

DBTIMEZONE

構文

用途

DBTIMEZONE ファンクションは、データベース・タイムゾーンの値を戻します。戻り型は、タイムゾーン・オフセット（'[+|-]TZH:TZM' という書式の文字列型）またはタイムゾーン地域名です。これは、最近の CREATE DATABASE または ALTER DATABASE 文でユーザーが指定したデータベース・タイムゾーンの値によって異なります。

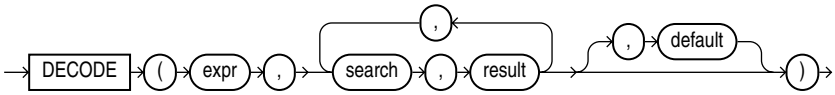
例

次の例では、データベース・タイムゾーンが UTC タイムゾーンに設定されていると想定します。

```
SELECT DBTIMEZONE FROM DUAL;

DBTIME
------
+00:00
```

DECODE

構文



用途

DECODE ファンクションは、*expr* と各 *search* の値を 1 つずつ比較します。*expr* が *search* と等しい場合、Oracle Database は対応する *result* を戻します。一致する値が見つからない場合は、*default* を戻します。*default* が省略されている場合は、NULL を戻します。

引数は、任意の数値型（NUMBER、BINARY_FLOAT、BINARY_DOUBLE）または文字列型です。

- *expr* および *search* が文字データである場合、Oracle は、非空白埋め比較セマンティクスを使用してそれらを比較します。*expr*、*search* および *result* のデータ型は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 です。戻される文字列は、VARCHAR2 データ型で、最初の *result* パラメータと同じキャラクタ・セットの文字列です。
- 最初の *search-result* の組が数値である場合、Oracle はすべての *search-result* の式と最初の *expr* を比較して、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

search、*result* および *default* の値は、式から導出できます。Oracle Database では、**短絡評価**を使用します。データベースは、*search* 値のいずれかと *expr* を比較する前にすべての search 値を評価するのではなく、各 search 値と *expr* を比較する前にのみ、各 *search* 値を評価します。その結果、*expr* と等しい *search* が見つかると、Oracle はその後の *search* を評価しません。

比較する前に、Oracle は *expr* と各 *search* 値を、最初の *search* 値のデータ型に自動的に変換します。Oracle は、戻り値を最初の *result* と同じデータ型に自動的に変換します。最初の *result* のデータ型が CHAR の場合、または最初の *result* が NULL の場合、Oracle は戻り値を VARCHAR2 データ型の値に変換します。

DECODE ファンクションでは、Oracle は 2 つの NULL を同等とみなします。*expr* が NULL の場合、Oracle は最初の *search* 値の *result* も NULL として戻します。

DECODE ファンクションのコンポーネントの最大数は、*expr*、*search*、*result*、*default* を含めて 255 です。

参照：

- 比較セマンティクスについては、2-36 ページの「データ型の比較規則」を参照してください。
- データ型変換については、2-39 ページの「データ変換」を参照してください。
- 浮動小数点の比較セマンティクスについては、2-13 ページの「浮動小数点数」を参照してください。
- 暗黙的な変換のデメリットについては、2-39 ページの「暗黙的なデータ変換と明示的なデータ変換」を参照してください。

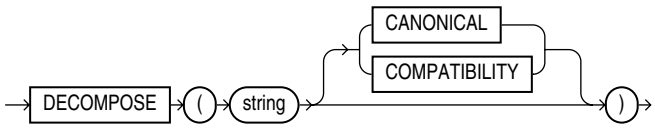
例

この例では、warehouse_id の値をデコードします。warehouse_id が 1 の場合「Southlake」を、warehouse_id が 2 の場合は「San Francisco」を戻します。warehouse_id が 1、2、3、4 のいずれでもない場合、ファンクションは「Non domestic」を戻します。

```
SELECT product_id,
       DECODE (warehouse_id, 1, 'Southlake',
                             2, 'San Francisco',
                             3, 'New Jersey',
                             4, 'Seattle',
                                'Non domestic') "Location"
       FROM inventories
       WHERE product_id < 1775
       ORDER BY product_id, "Location";
```

DECOMPOSE

構文



用途

DECOMPOSE は、Unicode キャラクタに対してのみ有効です。DECOMPOSE は、任意のデータ型の文字列を引数として取り、入力と同じキャラクタ・セットで分解された Unicode の文字列を戻します。たとえば、o ウムラウト・コードポイントは、ウムラウト・コードポイントが続く「o」コードポイントとして戻されます。

- *string* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。
- CANONICAL を使用すると標準的な分解が行われ、（たとえば、COMPOSE ファンクションを使用して）元の文字列への再構成が可能になります。これはデフォルトであり、NFD 正規形の文字列が戻されます。

- COMPATIBILITY を使用すると互換モードで分解が行われます。このモードでは、再構成はできません。このモードは、半角および全角のカタカナ文字を分解する場合など、外的な書式またはスタイル情報なしに再構成を行うことが望ましくない場合に有効です。このモードでは、NFKD 正規形の文字列が戻されます。

暗黙的な変換を使用して、CLOB および NCLOB の値がサポートされます。*char* が文字の LOB 値の場合、COMPOSE 演算の前に VARCHAR 値に変換されます。特定の開発環境で、LOB 値のサイズが VARCHAR のサポートする長さを超える場合、この演算は失敗します。

参照：

- Unicode キャラクタ・セットおよびキャラクタ・セマンティクスの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。
- 「COMPOSE」（5-36 ページ）

例

次の例では、文字列「Châteaux」をその要素のコードポイントに分解します。

```
SELECT DECOMPOSE ('Châteaux') FROM DUAL;

DECOMPOSE
---------
Cha^teaux
```

注意： この例の結果は、ご使用のオペレーティング・システムのキャラクタ・セットによって異なる場合があります。

DELETEXML

構文

用途

DELETEXML は、ターゲット XML の XPath 式で一致する単一または複数のノードを削除します。

- *XMLType_instance* は、XMLType のインスタンスです。
- *XPath_string* は、削除するノードを 1 つ以上指定する XPath 式です。先頭にスラッシュを付けて絶対 *XPath_string* を指定したり、先頭のスラッシュを省略して相対 *XPath_string* を指定できます。先頭のスラッシュを省略した場合、相対パスのコンテキストは、デフォルトでルート・ノードに設定されます。*XPath_string* で指定したノードの子ノードも削除されます。
- オプションの *namespace_string* は、*XPath_string* のネームスペース情報を提供します。このパラメータは、VARCHAR2 型である必要があります。

参照： このファンクションの詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

例

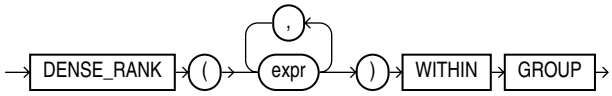
次の例では、5-17 ページの「APPENDCHILDXML」の例で変更したウェアハウスの 1 つの warehouse_spec から /Owner ノードを削除します。

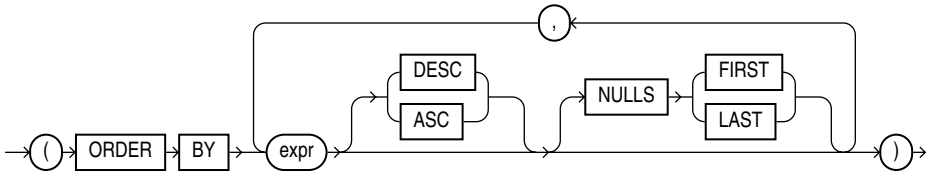
```
UPDATE warehouses SET warehouse_spec =
   DELETEXML(warehouse_spec,
   '/Warehouse/Building/Owner')
   WHERE warehouse_id = 2;

SELECT warehouse_id, warehouse_spec FROM warehouses
   WHERE warehouse_id in (2,3);

        ID WAREHOUSE_SPEC
---------- ---------------------------------
         2 <?xml version="1.0"?>
           <Warehouse>
             <Building>Rented</Building>
             <Area>50000</Area>
             <Docks>1</Docks>
             <DockType>Side load</DockType>
             <WaterAccess>Y</WaterAccess>
             <RailAccess>N</RailAccess>
             <Parking>Lot</Parking>
             <VClearance>12 ft</VClearance>
           </Warehouse>

         3 <?xml version="1.0"?>
           <Warehouse>
             <Building>Rented
               <Owner>Grandco</Owner>
               <Owner>ThirdOwner</Owner>
               <Owner>LesserCo</Owner>
             </Building>
             <Area>85700</Area>
             <DockType/>
             <WaterAccess>N</WaterAccess>
             <RailAccess>N</RailAccess>
             <Parking>Street</Parking>
             <VClearance>11.5 ft</VClearance>
           </Warehouse>
```

DENSE_RANK

集計の構文

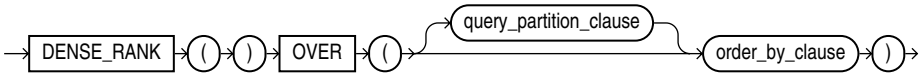
***dense_rank_aggregate*::=**





分析の構文

***dense_rank_analytic*::=**



参照： 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

DENSE_RANK は、順序付けされた行のグループ内の行のランクを計算し、そのランクを NUMBER として戻します。ランクは 1 から始まる連続した整数です。ランクの最大値は、問合せが戻す一意の数値です。ランクの値は、連続した整数です。ランク付け基準と同じ値を持つ行は、同じランクになります。このファンクションは、上位 N 番および下位 N 番のレポートに有効です。

このファンクションは、引数に任意の数値データ型を受け入れ、NUMBER を戻します。

- 集計ファンクションとして使用する DENSE_RANK は、ファンクションの引数によって識別される不確定な行の稠密ランクを、与えられたソート指定で計算します。ファンクションの引数は、各集計グループの単一行を識別するため、すべての引数は各集計グループ内で定数式に評価される必要があります。定数引数式および集計の *order_by_clause* の式の位置は、一致します。このため、引数の数は同じであり、型は互換性がある必要があります。
- 分析ファンクションとして使用する DENSE_RANK は、他の行について、問合せで戻される各行のランクを計算します。この計算は、*order_by_clause* にある *value_exprs* の値に基づいて行われます。

集計の例

次の例では、サンプル表 oe.employees から、給与が 15,500 ドルであり、歩合が 5% の仮想の従業員のランクを計算します。

```
SELECT DENSE_RANK(15500, .05) WITHIN GROUP
   (ORDER BY salary DESC, commission_pct) "Dense Rank"
   FROM employees;

         Dense Rank
-------------------
                  3
```

分析の例

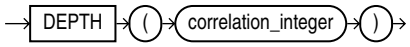
次の文は、人事部門または購買部門で働くすべての従業員の部門名、従業員名および給与を選択します。その後、この 2 つの部門それぞれについて、一意の各給与に対するランクを計算します。同じ給与は同じランクになります。この例と、5-138 ページの「RANK」の例を比較してください。

```
SELECT d.department_name, e.last_name, e.salary, DENSE_RANK()
   OVER (PARTITION BY e.department_id ORDER BY e.salary) AS drank
   FROM employees e, departments d
   WHERE e.department_id = d.department_id
   AND d.department_id IN ('30', '40')
   ORDER BY e.last_name, e.salary, d.department_name, drank;

DEPARTMENT_NAME                LAST_NAME                     SALARY      DRANK
c------------------------------ ------------------------- ---------- ----------
Purchasing                     Baida                           2900          4
Purchasing                     Colmenares                      2500          1
Purchasing                     Himuro                          2600          2
```

```
Purchasing                Khoo                     3100          5
Human Resources           Mavris                   6500          1
Purchasing                Raphaely                11000          6
Purchasing                Tobias                   2800          3
```

DEPTH

構文



用途

DEPTH は、UNDER_PATH および EQUALS_PATH 条件でのみ使用される補助ファンクションです。このファンクションは、同じ相関変数を持つ UNDER_PATH 条件によって指定されるパスのレベル数を戻します。

correlation_integer は任意の NUMBER 整数です。文に複数の一次条件が含まれている場合に、この補助ファンクションをその一次条件と関連付けるために使用します。1 未満の値は 1 として扱われます。

> **参照：** 7-19 ページの「EQUALS_PATH 条件」および 7-19 ページの「UNDER_PATH 条件」を参照してください。関連するファンクションについては、5-117 ページの「PATH」を参照してください。

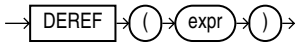
例

EQUALS_PATH および UNDER_PATH 条件は、2 つの補助ファンクション DEPTH および PATH を取ることができます。次に、その 2 つの補助ファンクションの使用方法を示します。この例では、XML Schema である warehouses.xsd（E-8 ページの「SQL 文での XML の使用方法」で作成）が存在することを前提としています。

```
SELECT PATH(1), DEPTH(2)
   FROM RESOURCE_VIEW
   WHERE UNDER_PATH(res, '/sys/schemas/OE', 1)=1
     AND UNDER_PATH(res, '/sys/schemas/OE', 2)=1;

PATH(1)                               DEPTH(2)
--------------------------------      --------
/www.example.com                       1

/www.example.com/xwarehouses.xsd       2
```

DEREF

構文



用途

DEREF は、引数 *expr* のオブジェクト参照を戻します。この場合、*expr* はオブジェクトに REF を戻す必要があります。問合せでこのファンクションを使用しない場合、次の例で示すとおり、かわりに REF のオブジェクト ID を戻します。

> **参照：** 「MAKE_REF」（5-97 ページ）

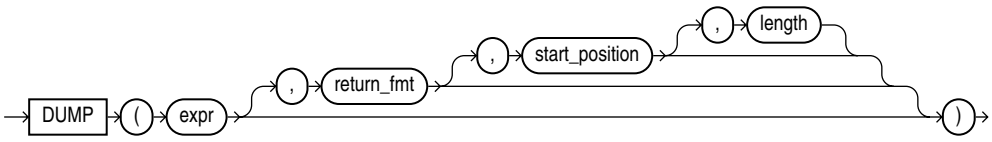
例

サンプル・スキーマ oe には、cust_address_typ というオブジェクト型が含まれます。8-22 ページの「REF 制約の例」では、類似する型 cust_address_typ_new、およびその型への REF である 1 つの列を含む表を作成します。次の例では、その列に挿入を行う方法、および DEREF を使用して列から情報を抽出する方法を示します。

```
INSERT INTO address_table VALUES
   ('1 First', 'G45 EU8', 'Paris', 'CA', 'US');

INSERT INTO customer_addresses
  SELECT 999, REF(a) FROM address_table a;

SELECT address FROM customer_addresses
   ORDER BY address;

ADDRESS
--------------------------------------------------------------------------------
000022020876B2245DBE325C5FE03400400B40DCB176B2245DBE305C5FE03400400B40DCB1

SELECT DEREF(address) FROM customer_addresses;

DEREF(ADDRESS)(STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE, COUNTRY_ID)
--------------------------------------------------------------------------------
CUST_ADDRESS_TYP('1 First', 'G45 EU8', 'Paris', 'CA', 'US')
```

DUMP

構文



用途

DUMP は、*expr* のデータ型コード、長さ（バイト単位）および内部表現を含む VARCHAR2 値を戻します。戻される結果は、常にデータベース・キャラクタ・セットの文字です。各コードに対応するデータ型については、2-7 ページの表 2-1「組込みデータ型の概要」を参照してください。

引数 *return_fmt* には戻り値の書式として、次の値のいずれかを指定します。

- 8 は、結果を 8 進表記で戻します。
- 10 は、結果を 10 進表記で戻します。
- 16 は、結果を 16 進表記で戻します。
- 17 は、各バイトがコンパイラのキャラクタ・セット内の出力可能な文字（通常は ASCII または EBCDIC）として解釈される場合にのみ、文字として出力された各バイトを戻します。一部の ASCII 制御文字は、^X 形式で出力される場合もあります。それ以外の場合、文字は 16 進表記で出力されます。NLS パラメータはすべて無視されます。*return_fmt* 17 を指定した DUMP の場合、特定の出力書式には依存しないでください。

デフォルトでは、戻り値にキャラクタ・セット情報が含まれません。*expr* のキャラクタ・セット名を取り出すには、前述の書式のいずれかの値に 1000 を加えます。たとえば、*return_fmt* に 1008 を指定すると、8 進表記で結果が戻り、さらに *expr* のキャラクタ・セット名が得られます。

引数 *start_position* と *length* を組み合せて、内部表現の戻す部分を指定します。デフォルトでは、10 進表記で全体の内部表現が戻されます。

expr が NULL の場合は NULL を戻します。

このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。

> **参照：** 詳細は、2-36 ページの「データ型の比較規則」を参照してください。

例

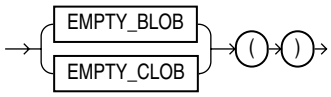
次の例では、文字列式および列からダンプ情報を抽出する方法を示します。

```
SELECT DUMP('abc', 1016)
   FROM DUAL;

DUMP('ABC',1016)
----------------------------------------
Typ=96 Len=3 CharacterSet=WE8DEC: 61,62,63

SELECT DUMP(last_name, 8, 3, 2) "OCTAL"
   FROM employees
   WHERE last_name = 'Hunold'
   ORDER BY employee_id;

OCTAL
----------------------------------------------------------------------
Typ=1 Len=6: 156,157

SELECT DUMP(last_name, 10, 3, 2) "ASCII"
   FROM employees
   WHERE last_name = 'Hunold'
   ORDER BY employee_id;

ASCII
----------------------------------------------------------------------
Typ=1 Len=6: 110,111
```

EMPTY_BLOB、EMPTY_CLOB

構文

***empty_LOB*::=**



用途

EMPTY_BLOB および EMPTY_CLOB は、LOB 変数を初期化したり、INSERT または UPDATE 文で LOB 列または属性を EMPTY に初期化できる空の LOB ロケータを戻します。EMPTY とは、LOB は初期化されていても、データが移入されていない状態をいいます。

> **注意：** 空の LOB と、NULL の LOB は、同じではありません。また、空の CLOB と、長さが 0（ゼロ）の文字列を含む LOB は、同じではありません。詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』を参照してください。

LOB ロケータの制限 このファンクションから戻されるロケータは、DBMS_LOB パッケージまたは OCI へのパラメータとして使用することはできません。

例

次の例では、サンプル表 pm.print_media の ad_photo 列を EMPTY に初期化します。

```
UPDATE print_media SET ad_photo = EMPTY_BLOB();
```

EXISTSNODE

構文

用途

EXISTSNODE は、指定されたパスを使用して XML 文書をトラバースした後にノードが存在するかを判断します。XML 文書、およびパスを指定する XPath 文字列 VARCHAR2 を含む XMLType インスタンスを引数として指定します。オプションの *namespace_string* は、接頭辞のデフォルト・マッピングまたはネームスペース・マッピング（Oracle Database が XPath 式を評価する場合に使用）を指定する VARCHAR2 値に解決される必要があります。

namespace_string 引数のデフォルトは、ルート要素の名前空間になります。*Xpath_string* のサブ要素を参照する場合は、*namespace_string* を指定する必要があります。また、これらの引数の両方に who 接頭辞を指定する必要があります。

> **参照：** *namespace_string* の指定例および who 接頭辞の使用例については、E-8 ページの「SQL 文での XML の使用方法」を参照してください。

戻り値は、NUMBER です。

- ドキュメントに XPath トラバースを適用した後にノードが存在しない場合は、0（ゼロ）が戻ります。
- ノードが存在する場合は、1 が戻ります。

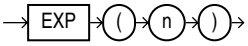
例

次の例では、サンプル表 oe.warehouses の warehouse_spec 列の XML パスに /Warehouse/Dock ノードが存在するかを判断します。

```
SELECT warehouse_id, warehouse_name
   FROM warehouses
   WHERE EXISTSNODE(warehouse_spec, '/Warehouse/Docks') = 1
   ORDER BY warehouse_id, warehouse_name;

WAREHOUSE_ID WAREHOUSE_NAME
------------ ----------------------------------
           1 Southlake, Texas
           2 San Francisco
           4 Seattle, Washington
```

EXP

構文



用途

EXP は、e を *n* 乗した値（e = 2.71828183 ...）を戻します。このファンクションは、引数と同じ型の値を戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数が BINARY_FLOAT の場合、このファンクションは BINARY_DOUBLE を戻します。それ以外の場合、引数と同じ数値データ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

例

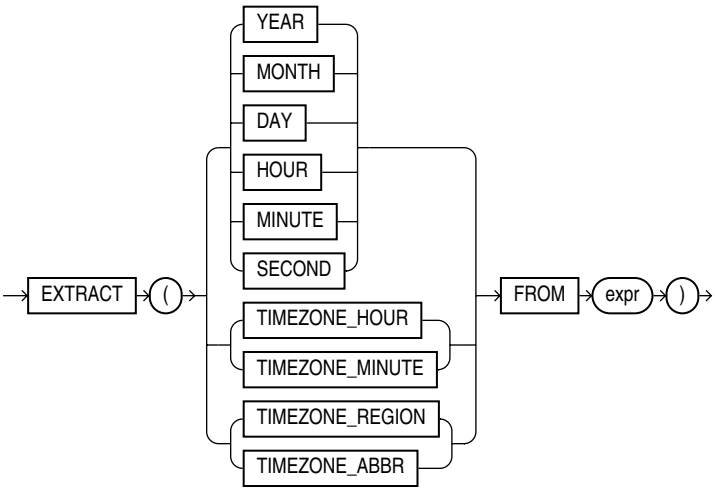
次の例では、e を 4 乗した値を戻します。

```
SELECT EXP(4) "e to the 4th power" FROM DUAL;

e to the 4th power
------------------
          54.59815
```

EXTRACT（日時）

構文

***extract_datetime*::=**



用途

EXTRACT は、日時式または期間式から、指定された日時フィールドの値を抽出して戻します。*expr* には、要求されたフィールドと互換性のある日時または期間データ型に評価される任意の式を指定できます。

- YEAR または MONTH が要求された場合、*expr* はデータ型 DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONE、TIMESTAMP WITH LOCAL TIME ZONE または INTERVAL YEAR TO MONTH の式に評価される必要があります。
- DAY が要求された場合、*expr* はデータ型 DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONE、TIMESTAMP WITH LOCAL TIME ZONE または INTERVAL DAY TO SECOND の式に評価される必要があります。
- HOUR、MINUTE または SECOND が要求された場合、*expr* はデータ型 TIMESTAMP、TIMESTAMP WITH TIME ZONE、TIMESTAMP WITH LOCAL TIME ZONE または INTERVAL DAY TO SECOND の式に評価される必要があります。DATE は、Oracle Database によって時刻フィールドを持たない ANSI DATE データ型として処理されるため、ここでは有効ではありません。
- TIMEZONE_HOUR、TIMEZONE_MINUTE、TIMEZONE_ABBR、TIMEZONE_REGION または TIMEZONE_OFFSET が要求された場合、*expr* はデータ型 TIMESTAMP WITH TIME ZONE または TIMESTAMP WITH LOCAL TIME ZONE の式に評価される必要があります。

EXTRACT は、*expr* を ANSI 日時データ型として解釈します。たとえば、EXTRACT は DATE をレガシー Oracle DATE ではなく、時刻要素を持たない ANSI DATE として処理します。したがって、DATE 値からは、YEAR、MONTH および DAY のみを抽出できます。同様に、TIMESTAMP WITH TIME ZONE データ型からは、TIMEZONE_HOUR および TIMEZONE_MINUTE のみを抽出できます。

TIMEZONE_REGION を抽出する場合、戻り値は、適切なタイムゾーン名を含む文字列です。また、TIMEZONE_ABBR（略称）を抽出する場合、戻り値は、適切な略称を含む文字列です。その他の値を抽出する場合、戻り値はグレゴリオ暦での日時値を表す整数です。タイムゾーン値を持つ日時から抽出する場合、戻り値は UTC です。有効なタイムゾーン名およびそれに対する略称を表示するには、V$TIMEZONE_NAMES 動的パフォーマンス・ビューに問合せを実行してください。

このファンクションは、次に示す最初の例のように、非常に大規模な表の日時フィールド値を操作する場合に特に有効です。

注意： 夏時間機能には、タイムゾーン地域名が必要です。地域名は、2 つのタイムゾーン・ファイルに格納されます。デフォルトのタイムゾーン・ファイルは、パフォーマンスを最大にするために一般的なタイムゾーンのみの小さなファイルです。タイムゾーンがデフォルトのファイルに存在しない場合は、環境変数 ORA_TZFILE を使用して完全な（大きい）ファイルへのパスを指定するまで、夏時間はサポートされません。

日時フィールドと日時または期間値の式を組み合せると、あいまいな結果になる場合があります。この場合、Oracle Database は、UNKNOWN を戻します（詳細は、次の例を参照してください）。

参照：

- 両方のファイルに含まれるすべてのタイムゾーン地域名のリストは、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。
- *datetime_value_expr* および *interval_value_expr* の詳細は、2-20 ページの「日時および期間の演算」を参照してください。
- 動的パフォーマンス・ビューの詳細は、『Oracle Database リファレンス』を参照してください。

例

次の例では、oe.orders 表から、各月の注文数を戻します。

```
SELECT EXTRACT(month FROM order_date) "Month",
  COUNT(order_date) "No. of Orders"
  FROM orders
  GROUP BY EXTRACT(month FROM order_date)
  ORDER BY "No. of Orders" DESC;

     Month No. of Orders
---------- -------------
        11            15
         7            14
         6            14
         3            11
         5            10
         9             9
         2             9
         8             7
        10             6
         1             5
        12             4
         4             1

12 rows selected.
```

次の例では、1998 年を戻します。

```
SELECT EXTRACT(YEAR FROM DATE '1998-03-07') FROM DUAL;

EXTRACT(YEARFROMDATE'1998-03-07')
---------------------------------
                             1998
```

次の例では、サンプル表 hr.employees から、1998 以降に雇用されたすべての従業員を選択します。

```
SELECT last_name, employee_id, hire_date
   FROM employees
   WHERE EXTRACT(YEAR FROM
   TO_DATE(hire_date, 'DD-MON-RR')) > 1998
   ORDER BY hire_date;

LAST_NAME                 EMPLOYEE_ID HIRE_DATE
------------------------- ----------- ---------
Landry                            127 14-JAN-99
Lorentz                           107 07-FEB-99
Cabrio                            187 07-FEB-99
. . .
```

次の例では、結果があいまいになるため、Oracle は UNKNOWN を戻します。

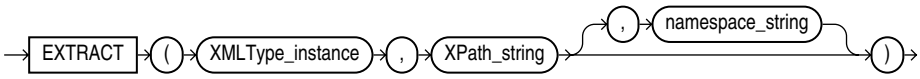
```
SELECT EXTRACT(TIMEZONE_REGION
      FROM TIMESTAMP '1999-01-01 10:00:00 -08:00')
   FROM DUAL;

EXTRACT(TIMEZONE_REGIONFROMTIMESTAMP'1999-01-0110:00:00-08:00')
---------------------------------------------------------------
UNKNOWN
```

タイムゾーン数値オフセットが式に指定され、その数値オフセットが複数のタイムゾーン地域をマップする場合、結果があいまいになります。

EXTRACT（XML）

構文

***extract_xml*::=**



用途

EXTRACT（XML）は、EXISTSNODE ファンクションに似ています。VARCHAR2 の XPath 文字列を適用し、XML のフラグメントを含む XMLType インスタンスを戻します。先頭にスラッシュを付けて絶対 *XPath_string* を指定したり、先頭のスラッシュを省略して相対 *XPath_string* を指定できます。先頭のスラッシュを省略した場合、相対パスのコンテキストは、デフォルトでルート・ノードに設定されます。処理している XML でネームスペース接頭辞が使用される場合、オプションの *namespace_string* が必要です。この引数は、接頭辞のデフォルト・マッピングまたはネームスペース・マッピング（Oracle Database が XPath 式を評価する場合に使用）を指定する VARCHAR2 値に解決される必要があります。

例

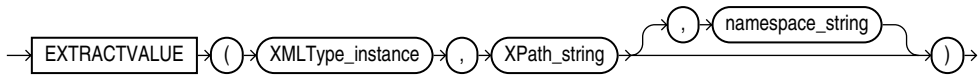
次の例では、サンプル表 oe.warehouses の warehouse_spec 列の XML パスの /Warehouse/Dock ノードの値を抽出します。

```
SELECT warehouse_name, EXTRACT(warehouse_spec, '/Warehouse/Docks')
   "Number of Docks"
   FROM warehouses
   WHERE warehouse_spec IS NOT NULL
   ORDER BY warehouse_name, "Number of Docks";

WAREHOUSE_NAME            Number of Docks
------------------------- -------------------------
New Jersey
San Francisco             <Docks>1</Docks>
Seattle, Washington       <Docks>3</Docks>
Southlake, Texas          <Docks>2</Docks>
```

この例と、XML のフラグメントのスカラー値を戻す、5-65 ページの「EXTRACTVALUE」の例を比較してみてください。

EXTRACTVALUE

構文



EXTRACTVALUE は、引数として XMLType インスタンスと XPath 式を取り、結果として得られるノードのスカラー値を戻します。結果はシングルノードであり、テキスト・ノード、属性または要素のいずれかである必要があります。結果が要素である場合、その要素はシングル・テキスト・ノードを子ノードとして持つ必要があり、このファンクションが戻す値は、その子ノードの値になります。先頭にスラッシュを付けて絶対 *XPath_string* を指定したり、先頭のスラッシュを省略して相対 *XPath_string* を指定できます。先頭のスラッシュを省略した場合、相対パスのコンテキストは、デフォルトでルート・ノードに設定されます。

指定された XPath が複数の子ノードを持つノードを指す場合、または指されたノードが非テキスト・ノードの子を持つ場合は、エラーが戻されます。オプションの *namespace_string* は、接頭辞のデフォルト・マッピングまたはネームスペース・マッピング（Oracle が XPath 式を評価する場合に使用）を指定する VARCHAR2 値に解決される必要があります。

XML Schema に基づくドキュメントで戻り値の型が推測できる場合は、適切な型のスカラー値が戻ります。その他の場合は、VARCHAR2 型の結果が戻ります。XML Schema に基づかないドキュメントの場合、戻り値は常に VARCHAR2 です。

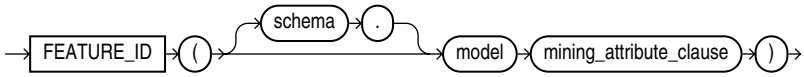
例

次の例では、入力として、5-65 ページの「EXTRACT（XML）」の例と同じ引数を取ります。このファンクションは、EXTRACT ファンクションとは異なり、XML のフラグメントを戻すのではなく、XML のフラグメントのスカラー値を戻します。

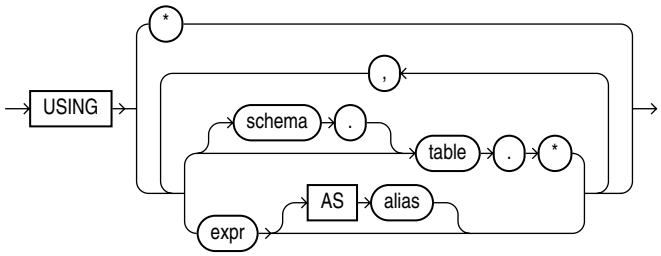
```
SELECT warehouse_name,
   EXTRACTVALUE(e.warehouse_spec, '/Warehouse/Docks')
   "Docks"
   FROM warehouses e
   WHERE warehouse_spec IS NOT NULL;

WAREHOUSE_NAME       Docks
-------------------- ------------
Southlake, Texas     2
San Francisco        1
New Jersey
Seattle, Washington  3
```

FEATURE_ID

構文



***mining_attribute_clause*::=**



用途

このファンクションは、DBMS_DATA_MINING パッケージまたは Oracle Data Mining Java API を使用して作成した特徴抽出モデルで使用するためのものです。このファンクションは、行内で最も高い数値を持つ特徴の識別子を Oracle の NUMBER で戻します。

mining_attribute_clause は、PREDICTION ファンクションと同様に動作します。詳細は、5-126 ページの「mining_attribute_clause」を参照してください。

参照 :

- Oracle Data Mining の詳細は、『Oracle Data Mining 概要』を参照してください。
- コードで使用可能なデモ・プログラムの詳細は、『Oracle Data Mining 管理者ガイド』を参照してください。
- SQL データ・マイニング・ファンクションを使用したリアルタイム・スコアリングの詳細は、『Oracle Data Mining アプリケーション開発者ガイド』を参照してください。

例

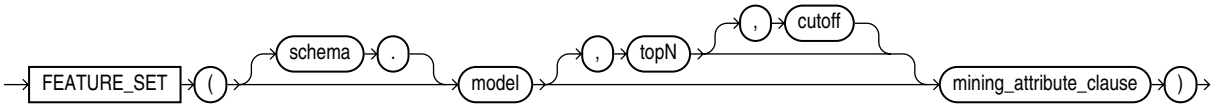
次の例では、データセット内の特徴と、それに対応する顧客の件数を示します。

この例と前提条件のデータ・マイニング操作（nmf_sh_sample モデルおよび nmf_sh_sample_apply_prepared ビューの作成など）は、デモ・ファイル $ORACLE_HOME/rdbms/demo/dmnmdemo.sql で確認できます。データ・マイニングのデモ・ファイルの一般情報は、『Oracle Data Mining 管理者ガイド』を参照してください。次に、このファンクションの構文の使用例を示します。

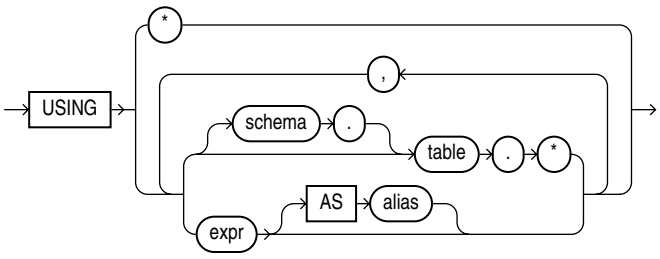
```
SELECT FEATURE_ID(nmf_sh_sample USING *) AS feat, COUNT(*) AS cnt
  FROM nmf_sh_sample_apply_prepared
GROUP BY FEATURE_ID(nmf_sh_sample USING *)
ORDER BY cnt DESC;

      FEAT        CNT
---------- ----------
         7       1443
         2         49
         3          6
         1          1
         6          1
```

FEATURE_SET

構文



***mining_attribute_clause*:=**



用途

このファンクションは、DBMS_DATA_MINING パッケージまたは Oracle Data Mining Java API を使用して作成した特徴抽出モデルで使用するためのものです。このファンクションは、有効な特徴を含むオブジェクトの VARRAY を戻します。VARRAY の各オブジェクトは、特徴 ID と特徴値を含むスカラー値の組です。オブジェクト・フィールドには、FEATURE_ID と VALUE の名前が付き、両方とも Oracle の NUMBER になります。

オプションの *topN* 引数は、特徴セットを上位 N の値のいずれかを持つ特徴セットに制限する正の整数です。N 番目の値に同順位がある場合でも、N 個の値のみが戻されます。この引数を指定しない場合、このファンクションはすべての特徴を戻します。

オプションの *cutoff* 引数は、戻される特徴を、指定したカットオフ以上の特徴値を持つ特徴のみに制限します。*cutoff* のみをフィルタ処理するには、NULL を *topN* に指定し、必要なカットオフ値を *cutoff* に指定します。

mining_attribute_clause は、PREDICTION ファンクションと同様に動作します。詳細は、5-126 ページの「mining_attribute_clause」を参照してください。

参照：

- Oracle Data Mining の詳細は、『Oracle Data Mining 概要』を参照してください。
- コードで使用可能なデモ・プログラムの詳細は、『Oracle Data Mining 管理者ガイド』を参照してください。
- SQL データ・マイニング・ファンクションを使用したリアルタイム・スコアリングの詳細は、『Oracle Data Mining アプリケーション開発者ガイド』を参照してください。

例

次の例では、指定した顧客レコードに対応する上位の特徴を示し（一致する品質を基準）、各特徴に上位の属性を決定します（0.25 を超える係数を基準）。

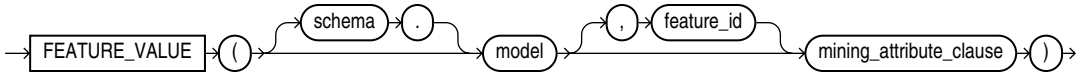
この例と前提条件のデータ・マイニング操作（モデル、ビューおよび型の作成など）は、デモ・ファイル $ORACLE_HOME/rdbms/demo/dmkmdemo.sql で確認できます。データ・マイニングのデモ・ファイルの一般情報は、『Oracle Data Mining 管理者ガイド』を参照してください。次に、このファンクションの構文の使用例を示します。

```
WITH
feat_tab AS (
SELECT F.feature_id fid,
       A.attribute_name attr,
       TO_CHAR(A.attribute_value) val,
       A.coefficient coeff
  FROM TABLE(DBMS_DATA_MINING.GET_MODEL_DETAILS_NMF('nmf_sh_sample')) F,
       TABLE(F.attribute_set) A
 WHERE A.coefficient > 0.25
),
feat AS (
SELECT fid,
       CAST(COLLECT(Featattr(attr, val, coeff))
         AS Featattrs) f_attrs
  FROM feat_tab
GROUP BY fid
),
cust_10_features AS (
SELECT T.cust_id, S.feature_id, S.value
  FROM (SELECT cust_id, FEATURE_SET(nmf_sh_sample, 10 USING *) pset
          FROM nmf_sh_sample_apply_prepared
         WHERE cust_id = 100002) T,
```

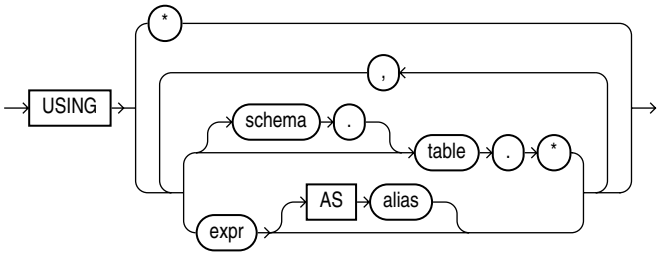
```
      TABLE(T.pset) S
)
SELECT A.value, A.feature_id fid,
       B.attr, B.val, B.coeff
  FROM cust_10_features A,
       (SELECT T.fid, F.*
          FROM feat T,
               TABLE(T.f_attrs) F) B
 WHERE A.feature_id = B.fid
ORDER BY A.value DESC, A.feature_id ASC, coeff DESC, attr ASC, val ASC;

   VALUE  FID ATTR                      VAL                  COEFF
-------- ---- ------------------------ ------------------- -------
  6.8409    7 YRS_RESIDENCE                                1.3879
  6.8409    7 BOOKKEEPING_APPLICATION                       .4388
  6.8409    7 CUST_GENDER              M                    .2956
  6.8409    7 COUNTRY_NAME             United States of Ame  .2848
                                       rica

  6.4975    3 YRS_RESIDENCE                                1.2668
  6.4975    3 BOOKKEEPING_APPLICATION                       .3465
  6.4975    3 COUNTRY_NAME             United States of Ame  .2927
                                       rica

  6.4886    2 YRS_RESIDENCE                                1.3285
  6.4886    2 CUST_GENDER              M                    .2819
  6.4886    2 PRINTER_SUPPLIES                              .2704
  6.3953    4 YRS_RESIDENCE                                1.2931
  5.9640    6 YRS_RESIDENCE                                1.1585
  5.9640    6 HOME_THEATER_PACKAGE                          .2576
  5.2424    5 YRS_RESIDENCE                                1.0067
  2.4714    8 YRS_RESIDENCE                                 .3297
  2.3559    1 YRS_RESIDENCE                                 .2768
  2.3559    1 FLAT_PANEL_MONITOR                            .2593

17 rows selected.
```

FEATURE_VALUE

構文



***mining_attribute_clause*:=**



用途

このファンクションは、DBMS_DATA_MINING パッケージまたは Oracle Data Mining Java API を使用して作成した特徴抽出モデルで使用するためのものです。このファンクションは、指定した特徴の値を戻します。*feature_id* 引数を指定しない場合、このファンクションは最も高い特徴値を戻します。この形式と FEATURE_ID ファンクションを組み合せて使用すると、特徴と値の最大の組合せを取得できます。

mining_attribute_clause は、PREDICTION ファンクションと同様に動作します。詳細は、5-126 ページの「mining_attribute_clause」を参照してください。

参照 :

- Oracle Data Mining の詳細は、『Oracle Data Mining 概要』を参照してください。
- コードで使用可能なデモ・プログラムの詳細は、『Oracle Data Mining 管理者ガイド』を参照してください。
- SQL データ・マイニング・ファンクションを使用したリアルタイム・スコアリングの詳細は、『Oracle Data Mining アプリケーション開発者ガイド』を参照してください。

例

次の例では、特徴 3 に対応する顧客を、一致する品質の順序で示します。

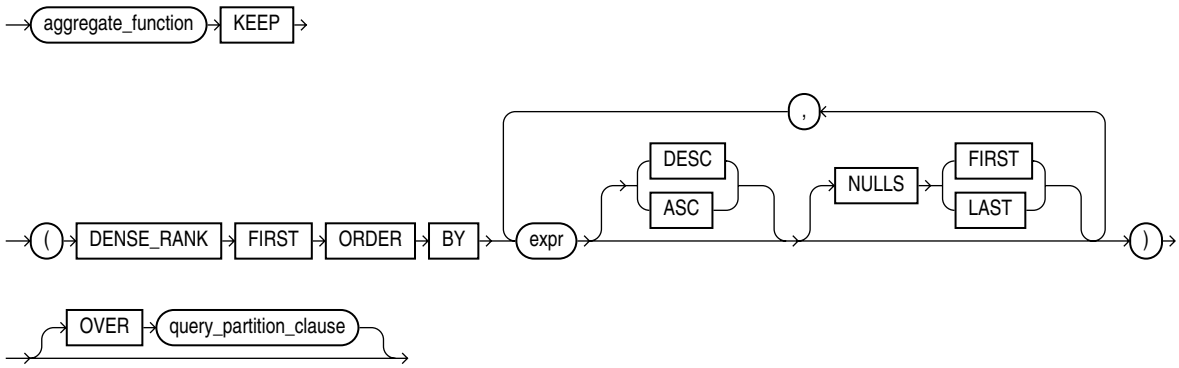
この例と前提条件のデータ・マイニング操作（モデルおよびビューの作成など）は、デモ・ファイル $ORACLE_HOME/rdbms/demo/dmkmdemo.sql で確認できます。データ・マイニングのデモ・ファイルの一般情報は、『Oracle Data Mining 管理者ガイド』を参照してください。次に、このファンクションの構文の使用例を示します。

```
SELECT *
  FROM (SELECT cust_id, FEATURE_VALUE(nmf_sh_sample, 3 USING *) match_quality
          FROM nmf_sh_sample_apply_prepared
         ORDER BY match_quality DESC)
 WHERE ROWNUM < 11;

   CUST_ID MATCH_QUALITY
---------- -------------
    100210    19.4101627
    100962    15.2482251
    101151    14.5685197
    101499    14.4186292
    100363    14.4037396
    100372    14.3335148
    100982    14.1716545
    101039    14.1079914
    100759    14.0913761
    100953    14.0799737

10 rows selected.
```

FIRST

構文

***first*::=**



参照： ORDER BY 句および OVER 句の構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

FIRST ファンクションと LAST ファンクションは類似しています。両方のファンクションとも、与えられたソート指定に対して、FIRST または LAST としてランク付けされた一連の行の一連の値を操作する集計ファンクションおよび分析ファンクションです。1 つの行のみが FIRST または LAST としてランク付けされている場合、集計は、1 つの要素のみを持つセットを操作します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を戻します。

ソートされたグループの最初または最後の行の値が必要であり、その値がソート・キーでない場合、FIRST および LAST ファンクションは、自己結合またはビューの必要性を排除し、パフォーマンスを向上させます。

- *aggregate_function* は、MIN、MAX、SUM、AVG、COUNT、VARIANCE または STDDEV ファンクションのいずれかです。FIRST または LAST のいずれかにランク付けされた行の値を操作します。1 つの行のみが FIRST または LAST としてランク付けされている場合、集計は、単一（集計ではない）のセットを操作します。
- KEEP キーワードはセマンティクスを明確にするためのものです。これは *aggregate_function* を修飾し、戻される *aggregate_function* の値が FIRST または LAST のみであることを示します。
- DENSE_RANK FIRST または DENSE_RANK LAST は、最小（FIRST）または最大（LAST）稠密ランク（「オリンピック・ランク」）を持つ行のみを集計することを示します。

OVER 句を指定すると、FIRST および LAST ファンクションを分析ファンクションとして使用できます。OVER 句のうち、*query_partitioning_clause* の部分のみがこれらのファンクションで有効です。

参照： 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。5-87 ページの「LAST」も参照してください。

集計の例

次の例では、サンプル表 hr.employees の各部門で、歩合が最低である従業員の中での最低給与、および歩合が最高の従業員の中での最高給与を戻します。

```
SELECT department_id,
MIN(salary) KEEP (DENSE_RANK FIRST ORDER BY commission_pct) "Worst",
MAX(salary) KEEP (DENSE_RANK LAST ORDER BY commission_pct) "Best"
   FROM employees
   GROUP BY department_id
   ORDER BY department_id, "Worst", "Best";

DEPARTMENT_ID      Worst       Best
------------- ---------- ----------
           10       4400       4400
           20       6000      13000
           30       2500      11000
           40       6500       6500
           50       2100       8200
           60       4200       9000
           70      10000      10000
           80       6100      14000
           90      17000      24000
          100       6900      12000
          110       8300      12000
                    7000       7000
```

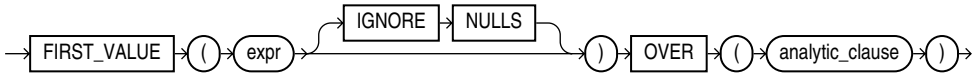
分析の例

次の例では、前述の例と同じ計算をしますが、当該部門の各従業員の結果を戻します。

```
SELECT last_name, department_id, salary,
   MIN(salary) KEEP (DENSE_RANK FIRST ORDER BY commission_pct)
      OVER (PARTITION BY department_id) "Worst",
   MAX(salary) KEEP (DENSE_RANK LAST ORDER BY commission_pct)
      OVER (PARTITION BY department_id) "Best"
    FROM employees
    ORDER BY department_id, salary;

LAST_NAME           DEPARTMENT_ID     SALARY      Worst       Best
------------------- ------------- ---------- ---------- ----------
Whalen                         10       4400       4400       4400
Fay                            20       6000       6000      13000
Hartstein                      20      13000       6000      13000
. . .
Gietz                         110       8300       8300      12000
Higgins                       110      12000       8300      12000
Grant                                   7000       7000       7000
```

FIRST_VALUE

構文



参照： 構文、セマンティクス、制限事項および *expr* の書式の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

FIRST_VALUE は分析ファンクションです。これは、順序付けられた値の集合にある最初の値を戻します。集合内の最初の値が NULL の場合、IGNORE NULLS を指定していないかぎり、ファンクションは NULL を戻します。この設定は、データの稠密化に役立ちます。IGNORE NULLS を指定すると、FIRST_VALUE は集合内の最初の NULL ではない値を戻します。すべての値が NULL の場合は NULL を戻します。データの稠密化の例は、19-43 ページの「パーティション化された外部結合の使用例 :」を参照してください。

expr には、FIRST_VALUE または他の分析ファンクションを使用して分析ファンクションをネストできません。ただし、他の組込みファンクション式を *expr* で使用できます。*expr* の書式の詳細は、6-2 ページの「SQL 式」を参照してください。

例

次の例では、部門 90 の各従業員について、その部門で給与が一番少ない従業員の名前を選択します。

```
SELECT department_id, last_name, salary, FIRST_VALUE(last_name)
  OVER (ORDER BY salary ASC ROWS UNBOUNDED PRECEDING) AS lowest_sal
  FROM (SELECT * FROM employees WHERE department_id = 90
    ORDER BY employee_id)
  ORDER BY department_id, last_name, salary, lowest_sal;

DEPARTMENT_ID LAST_NAME                 SALARY LOWEST_SAL
------------- ------------------------- ---------- -------------------------
           90 De Haan                    17000 Kochhar
           90 King                       24000 Kochhar
           90 Kochhar                    17000 Kochhar
```

例では、FIRST_VALUE ファンクションの非決定的な性質が示されています。Kochhar と De Haan の給与は同じであるため、Kochhar の次の行に De Haan があります。Kochhar が最初に表示されているのは、副問合せが戻す行が employee_id で順序付けられているためです。ただし、副問合せが戻す行が employee_id で降順に順序付けられている場合は、次の例に示すとおり、ファンクションは異なる値を戻します。

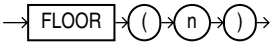
```
SELECT department_id, last_name, salary, FIRST_VALUE(last_name)
  OVER (ORDER BY salary ASC ROWS UNBOUNDED PRECEDING) as fv
    FROM (SELECT * FROM employees WHERE department_id = 90
      ORDER by employee_id DESC)
  ORDER BY department_id, last_name, salary, fv;

DEPARTMENT_ID LAST_NAME                 SALARY FV
------------- ------------------------- ---------- -------------------------
           90 De Haan                    17000 De Haan
           90 King                       24000 De Haan
           90 Kochhar                    17000 De Haan
```

次の例では、一意キーで順序付けることによって、FIRST_VALUE ファンクションを決定的にする方法を示します。

```
SELECT department_id, last_name, salary, hire_date,
   FIRST_VALUE(last_name) OVER
   (ORDER BY salary ASC, hire_date ROWS UNBOUNDED PRECEDING) AS fv
   FROM (SELECT * FROM employees
   WHERE department_id = 90 ORDER BY employee_id DESC)
   ORDER BY department_id, last_name, salary, hire_date;

DEPARTMENT_ID LAST_NAME           SALARY HIRE_DATE FV
------------- --------------- ---------- --------- -------------------------
           90 De Haan              17000 13-JAN-93 Kochhar
           90 King                 24000 17-JUN-87 Kochhar
           90 Kochhar              17000 21-SEP-89 Kochhar
```

FLOOR

構文



用途

FLOOR は *n* 以下の最大整数を戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

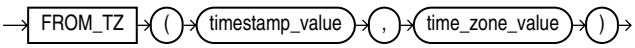
例

次の例では、15.7 以下である最大の整数を戻します。

```
SELECT FLOOR(15.7) "Floor" FROM DUAL;

     Floor
----------
        15
```

FROM_TZ

構文



用途

FROM_TZ ファンクションは、タイムスタンプ値およびタイムゾーンを TIMESTAMP WITH TIME ZONE 値に変換します。*time_zone_value* は、'TZH:TZM' 書式の文字列、またはオプションの TZD 書式を持つ TZR 書式で文字列を戻す文字式です。

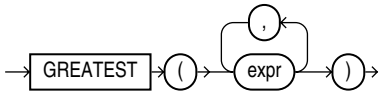
例

次の例では、タイムスタンプ値を TIMESTAMP WITH TIME ZONE に戻します。

```
SELECT FROM_TZ(TIMESTAMP '2000-03-28 08:00:00', '3:00')
   FROM DUAL;

FROM_TZ(TIMESTAMP'2000-03-2808:00:00','3:00')
---------------------------------------------------------------
28-MAR-00 08.00.00 AM +03:00
```

GREATEST

構文



用途

GREATEST は、1 つ以上の式のリストの最大値を戻します。Oracle Database は、最初の *expr* を使用して戻り型を判断します。最初の *expr* が数値である場合、Oracle は、数値の優先順位が最も高い引数を判断し、比較の前に残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。最初の *expr* が数値ではない場合、比較の前に、2 番目以降の各 *expr* が最初の *expr* のデータ型に暗黙的に変換されます。

Oracle Database は、非空白埋め比較セマンティクスを使用して各 *expr* を比較します。比較では、デフォルトの場合はバイナリが使用され、NLS_COMP パラメータが LINGUISTIC に設定され、NLS_SORT パラメータに BINARY 以外の設定がある場合は言語が使用されます。文字の比較は、データベース・キャラクタ・セットの文字の数値コードに基づいて行われます。また、文字ごとではなく、一連のバイトとして扱われる文字列全体で行われます。このファンクションによって戻される値が文字データの場合、そのデータ型は、最初の *expr* が文字データ型の場合は VARCHAR2、最初の *expr* が各国語キャラクタ・データ型の場合は NVARCHAR2 になります。

参照 :

- 文字の比較の詳細は、2-36 ページの「データ型の比較規則」を参照してください。
- 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。2 進浮動小数点の比較セマンティクスの詳細は、2-13 ページの「浮動小数点数」を参照してください。

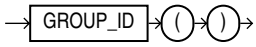
例

次の文では、最大値を持つ文字列を検索します。

```
SELECT GREATEST ('HARRY', 'HARRIOT', 'HAROLD')
   "Greatest" FROM DUAL;

Greatest
--------
HARRY
```

GROUP_ID

構文



用途

GROUP_ID は、GROUP BY 指定の結果から、重複するグループを識別します。このファンクションは、問合せ結果から重複グループを除外する場合に有効です。このファンクションは、重複グループを一意に識別するために、Oracle の NUMBER 値を戻します。このファンクションは、1 つの GROUP BY 句を含む SELECT 文のみに適用できます。

特定のグループ化で *n* 個の重複が存在する場合、GROUP_ID は 0 ～ *n*-1 の範囲の数値を戻します。

例

次の例では、サンプル表 sh.countries および sh.sales の問合せの結果、重複する co.country_region グループ化に値 1 を割り当てます。

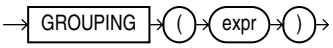
```
SELECT co.country_region, co.country_subregion,
   SUM(s.amount_sold) "Revenue",
   GROUP_ID() g
FROM sales s, customers c, countries co
WHERE s.cust_id = c.cust_id AND
   c.country_id = co.country_id AND
   s.time_id = '1-JAN-00' AND
   co.country_region IN ('Americas', 'Europe')
GROUP BY co.country_region,
   ROLLUP (co.country_region, co.country_subregion)
ORDER BY co.country_region, co.country_subregion, "Revenue", g;

COUNTRY_REGION       COUNTRY_SUBREGION                Revenue          G
-------------------- ------------------------------ ---------- ----------
Americas             Northern America                   944.6          0
Americas                                                944.6          0
Americas                                                944.6          1
Europe               Western Europe                     566.39          0
Europe                                                  566.39          0
Europe                                                  566.39          1
```

GROUP_ID が 1 より小さい行のみを戻すには、文の最後に次の HAVING 句を追加します。

```
HAVING GROUP_ID() < 1
```

GROUPING

構文



用途

GROUPING は、通常のグループ化された行と超集合行を区別します。ROLLUP や CUBE などの GROUP BY の拡張機能は、すべての値の集合が NULL で表される超集合行を生成します。GROUPING ファンクションを使用すると、通常の行に含まれる NULL と超集合行ですべての値の集合を表す NULL を区別できます。

GROUPING ファンクションの *expr* は、GROUP BY 句の式のいずれかと一致する必要があります。行の *expr* の値がすべての値の集合を表す NULL の場合、このファンクションは 1 の値を戻します。それ以外の場合は、0（ゼロ）を戻します。GROUPING ファンクションは、Oracle の NUMBER 値を戻します。これらの用語の詳細は、19-24 ページの「SELECT」の「group_by_clause」を参照してください。

例

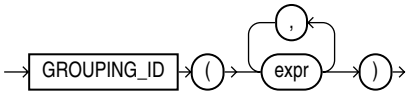
次の例では、サンプル表 hr.departments および hr.employees で、GROUPING ファンクションが 1（表の通常の行ではなく超集合行）を戻す場合、それ以外の場合に表示される NULL のかわりに、文字列「All Jobs」が「JOB」列に表示されます。

```
SELECT
   DECODE(GROUPING(department_name), 1, 'All Departments', department_name) AS
department,
   DECODE(GROUPING(job_id), 1, 'All Jobs', job_id) AS job,
   COUNT(*) "Total Empl",
   AVG(salary) * 12 "Average Sal"
FROM employees e, departments d
```

```
WHERE d.department_id = e.department_id
GROUP BY ROLLUP (department_name, job_id)
ORDER BY department, job, "Total Empl", "Average Sal";

DEPARTMENT                     JOB        Total Empl Average Sal
------------------------------ ---------- ---------- -----------
Accounting                     AC_ACCOUNT          1       99600
Accounting                     AC_MGR              1      144000
Accounting                     All Jobs            2      121800
Administration                 AD_ASST             1       52800
Administration                 All Jobs            1       52800
All Departments                All Jobs          106  77479.2453
Executive                      AD_PRES             1      288000
Executive                      AD_VP               2      204000
Executive                      All Jobs            3      232000
Finance                        All Jobs            6      103200
Finance                        FI_ACCOUNT          5       95040
. . .
```

GROUPING_ID

構文



用途

GROUPING_ID は、行に関連する GROUPING ビット・ベクトルに対応する数値を戻します。GROUPING_ID は、ROLLUP や CUBE など、GROUP BY の拡張機能および GROUPING ファンクションを含む SELECT 文にのみ適用できます。多くの GROUP BY 式を含む問合せでは、多くの GROUPING ファンクションを必要とする特定の行の GROUP BY レベルを指定するため、非常に複雑な SQL になります。GROUPING_ID は、このような場合に有効です。

GROUPING_ID は、複数の GROUPING ファンクションの結果をビット・ベクトル（1 と 0 を組み合せた文字列）に連結したものと同じです。GROUPING_ID を使用すると、複数の GROUPING ファンクションを使用する必要がなくなり、行のフィルタ条件の表記が簡単になります。GROUPING_ID を使用すると、要求する行が単一の条件（GROUPING_ID = *n*）によって識別されるため、行のフィルタ処理が簡単になります。このファンクションは、1 つの表に複数のレベルの集計を格納する場合に、特に有効です。

例

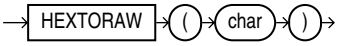
次の例では、サンプル表 sh.sales の問合せからグループ化 ID を抽出する方法を示します。

```
SELECT channel_id, promo_id, sum(amount_sold) s_sales,
   GROUPING(channel_id) gc,
   GROUPING(promo_id) gp,
   GROUPING_ID(channel_id, promo_id) gcp,
   GROUPING_ID(promo_id, channel_id) gpc
   FROM sales
   WHERE promo_id > 496
   GROUP BY CUBE(channel_id, promo_id)
   ORDER BY channel_id, promo_id, s_sales, gc;

CHANNEL_ID   PROMO_ID    S_SALES         GC         GP        GCP        GPC
---------- ---------- ---------- ---------- ---------- ---------- ----------
         2        999 25797563.2          0          0          0          0
         2            25797563.2          0          1          1          2
         3        999 55336945.1          0          0          0          0
```

```
3           55336945.1          0          1          1          2
4       999 13370012.5          0          0          0          0
4           13370012.5          0          1          1          2
        999 94504520.8          1          0          2          1
            94504520.8          1          1          3          3
```

HEXTORAW

構文



用途

HEXTORAW は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 キャラクタ・セットで 16 進数を含む *char* を RAW 値に変換します。

このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。

> **参照：** 詳細は、2-36 ページの「データ型の比較規則」を参照してください。

例

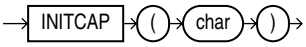
次の例では、RAW 列を含む簡単な表を作成し、RAW に変換された 16 進数値を挿入します。

```
CREATE TABLE test (raw_col RAW(10));

INSERT INTO test VALUES (HEXTORAW('7D'));
```

> **参照：** 2-23 ページの「RAW データ型と LONG RAW データ型」および 5-141 ページの「RAWTOHEX」を参照してください。

INITCAP

構文



用途

INITCAP は、各単語の最初の文字を大文字、残りの文字を小文字にして *char* を戻します。単語は空白または英数字以外の文字で区切ります。

char は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型です。戻り値は、*char* と同じデータ型です。データベースは、基礎となるキャラクタ・セットに対して定義したバイナリ・マッピングに基づいて先頭文字の形式を設定します。大文字と小文字の区別については、5-108 ページの「NLS_INITCAP」を参照してください。

このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。

> **参照：** 詳細は、2-36 ページの「データ型の比較規則」を参照してください。

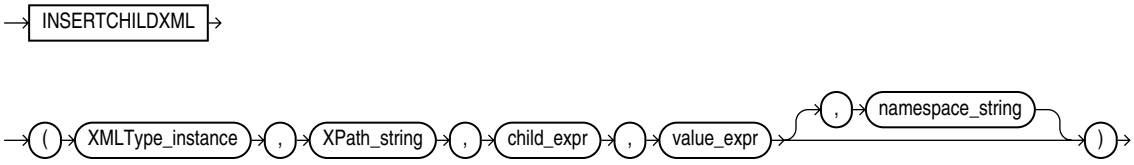
例

次の例では、文字列にある各単語を大文字で始めます。

```
SELECT INITCAP('the soap') "Capitals" FROM DUAL;

Capitals
---------
The Soap
```

INSERTCHILDXML

構文



用途

INSERTCHILDXML は、ユーザー指定の値を、XPath 式で指定したノードのターゲット XML に挿入します。このファンクションと 5-82 ページの「INSERTXMLBEFORE」を比較してください。

> **参照：** このファンクションの詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

- *XMLType_instance* は、XMLType のインスタンスです。
- *XPath_string* は、1 つ以上の子ノードが挿入されるノードを 1 つ以上示す XPath 式です。先頭にスラッシュを付けて絶対 *XPath_string* を指定したり、先頭のスラッシュを省略して相対 *XPath_string* を指定できます。先頭のスラッシュを省略した場合、相対パスのコンテキストは、デフォルトでルート・ノードに設定されます。
- *child_expr* は、挿入する要素または属性のノードを 1 つ以上指定します。
- *value_expr* は、挿入されるノードを 1 つ以上指定する XMLType のフラグメントです。これは文字列に変換する必要があります。
- オプションの *namespace_string* は、*XPath_string* のネームスペース情報を提供します。このパラメータは、VARCHAR2 型である必要があります。

例

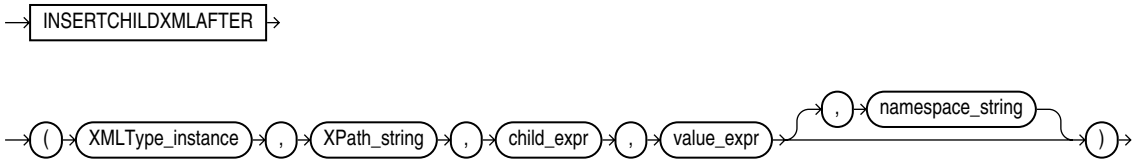
次の例では、2 番目の /Owner ノードを、5-17 ページの「APPENDCHILDXML」の例で更新したウェアハウスの 1 つの warehouse_spec に追加します。

```
UPDATE warehouses SET warehouse_spec =
   INSERTCHILDXML(warehouse_spec,
   '/Warehouse/Building', 'Owner',
   XMLType('<Owner>LesserCo</Owner>'))
   WHERE warehouse_id = 3;

SELECT warehouse_spec FROM warehouses
   WHERE warehouse_id = 3;

WAREHOUSE_SPEC
--------------------------------------------------------------------------------
<?xml version="1.0"?>
<Warehouse>
  <Building>Rented
```

```
    <Owner>Grandco</Owner>
    <Owner>LesserCo</Owner>
  </Building>
  <Area>85700</Area>
  <DockType/>
  <WaterAccess>N</WaterAccess>
  <RailAccess>N</RailAccess>
  <Parking>Street</Parking>
  <VClearance>11.5 ft</VClearance>
</Warehouse>
```

INSERTCHILDXMLAFTER

構文



用途

INSERTXMLCHILDAFTER は、1 つ以上のコレクション要素をターゲット親要素の子として挿入します。各ターゲットは、指定された既存のコレクション要素の直後に挿入されます。挿入の対象となる既存の XML 文書は、スキーマベースまたは非スキーマベースにすることができます。

- *XMLType_instance* は、挿入の対象となる XML データを識別します。
- *XPath_string* は、対象データ内の親要素の位置を特定します。子データは、それぞれの親要素の下に挿入されます。
- *child_expr* は、挿入される子データの前に配置される既存の子の位置を指定する相対 XPath 1.0 式です。この式は、親 XPath によって示された要素の子要素を指定する必要があります。また、条件を含むこともできます。
- *value_expr* は、挿入する XMLType 子要素データです。この引数の各トップレベル要素のノードは、child_expr によって示された要素と同じデータ型である必要があります。
- オプションの *namespace_string* は、親要素、既存の子要素および挿入される子要素 XML データの名前空間を指定します。

このファンクション、その使用方法および例の詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

例

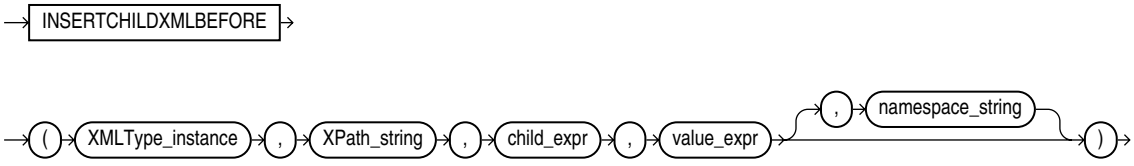
次の例は、INSERTCHILDXML の例と似ていますが、この例では、INSERTCHILDXML の例で追加された /Owner ノードの後に、3 番目の /Owner ノードが追加されています。問合せの出力は読み取りやすいように整えられています。

```
UPDATE warehouses SET warehouse_spec =
   INSERTCHILDXMLAFTER(warehouse_spec,
   '/Warehouse/Building','Owner[2]',
   XMLType('<Owner>ThirdOwner</Owner>'))
   WHERE warehouse_id = 3;

SELECT warehouse_name, EXTRACT(warehouse_spec,
   '/Warehouse/Building/Owner') "Owners"
   FROM warehouses
   WHERE warehouse_id = 3;
```

```
WAREHOUSE_NAME                     Owners
---------------------------------- ------------------------------
New Jersey                         <Owner>LesserCo</Owner>
                                   <Owner>GrandCo</Owner>
                                   <Owner>ThirdOwner</Owner>
```

INSERTCHILDXMLBEFORE

構文



用途

INSERTXMLCHILDBEFORE は、1 つ以上のコレクション要素をターゲット親要素の子として挿入します。各ターゲットは、指定された既存のコレクション要素の直前に挿入されます。挿入の対象となる既存の XML 文書は、スキーマベースまたは非スキーマベースにすることができます。

- *XMLType_instance* は、挿入の対象となる XML データを識別します。
- *XPath_string* は、対象データ内の親要素の位置を特定します。子データは、それぞれの親要素の下に挿入されます。
- *child_expr* は、挿入される子データの後に配置される既存の子の位置を指定する相対 XPath 1.0 式です。この式は、親 XPath によって示された要素の子要素を指定する必要があります。また、条件を含むこともできます。
- *value_expr* は、挿入する XMLType 子要素データです。この引数の各トップレベル要素のノードは、child_expr によって示された要素と同じデータ型である必要があります。
- オプションの *namespace_string* は、親要素、既存の子要素および挿入される子要素 XML データの名前空間を指定します。

このファンクション、その使用方法および例の詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

例

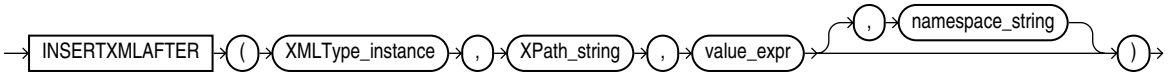
次の例は、INSERTCHILDXML の例と似ていますが、この例では、INSERTCHILDXML の例で追加された /Owner ノードの前に、3 番目の /Owner ノードが追加されています。問合せの出力は読み取りやすいように整えられています。

```
UPDATE warehouses SET warehouse_spec =
   INSERTCHILDXMLBEFORE(warehouse_spec,
   '/Warehouse/Building','Owner[2]',
   XMLType('<Owner>ThirdOwner</Owner>'))
   WHERE warehouse_id = 3;

SELECT warehouse_name, EXTRACT(warehouse_spec,
   '/Warehouse/Building/Owner') "Owners"
   FROM warehouses
   WHERE warehouse_id = 3;
```

```
WAREHOUSE_NAME                     Owners
---------------------------------- ------------------------------
New Jersey                         <Owner>LesserCo</Owner>
                                   <Owner>ThirdOwner</Owner>
                                   <Owner>GrandCo</Owner>
```

INSERTXMLAFTER

構文



用途

INSERTXMLAFTER は、属性のノードではないターゲット・ノードの直後に任意の種類の 1 つ以上のノードを挿入します。挿入の対象となる XML 文書は、スキーマベースまたは非スキーマベースにすることができます。このファンクションは insertXMLbefore と似ていますが、ターゲット・ノードの前ではなく後に挿入します。

- *XMLType_instance* は、挿入のターゲット・ノードを指定します。
- *XPath_string* は、属性のノードを除く任意の 0 個以上のノードの位置をターゲット・ノード内で指定する XPath 1.0 式です。XML データは、これらの各ノードの直後に挿入されます。つまり、指定された各ノードは、*value_expr* で指定されたノードの前に配置される兄弟ノードとなります。
- *value_expr* は、挿入される XML データです。任意の種類の 1 つ以上のノードを指定できます。ノードの順序は挿入後も保持されます。
- *namespace_string* は、ターゲット・ノードのネームスペースです。

このファンクション、その使用方法および例の詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

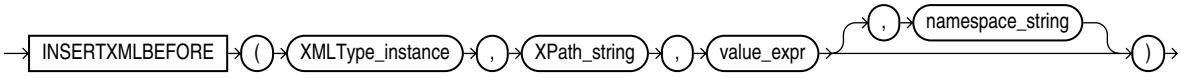
例

次の例は、INSERTCHILDXML の例と似ていますが、この例では、INSERTCHILDXML の例で追加された /Owner ノードの後に、3 番目の /Owner ノードが追加されています。問合せの出力は読み取りやすいように整えられています。

```
UPDATE warehouses SET warehouse_spec =
   INSERTXMLAFTER(warehouse_spec,
   '/Warehouse/Building/Owner[1]',
   XMLType('<Owner>SecondOwner</Owner>'))
   WHERE warehouse_id = 3;

SELECT warehouse_name, EXTRACT(warehouse_spec,
   '/Warehouse/Building/Owner') "Owners"
   FROM warehouses
   WHERE warehouse_id = 3;
```

INSERTXMLBEFORE

構文



用途

INSERTXMLBEFORE は、ユーザー指定の値を、XPath 式で指定したノードの前のターゲット XML に挿入します。このファンクションは INSERTXMLAFTER と似ていますが、ターゲット・ノードの後ではなく前に挿入します。このファンクションと 5-79 ページの「INSERTCHILDXML」を比較してください。

- *XMLType_instance* は、XMLType のインスタンスです。
- *XPath_string* は、1 つ以上の子ノードが挿入されるノードを 1 つ以上示す XPath 式です。先頭にスラッシュを付けて絶対 *XPath_string* を指定したり、先頭のスラッシュを省略して相対 *XPath_string* を指定できます。先頭のスラッシュを省略した場合、相対パスのコンテキストは、デフォルトでルート・ノードに設定されます。
- *value_expr* は、挿入される 1 つ以上のノードと親ノード内での位置を定義する XMLType のフラグメントです。これは文字列に変換する必要があります。
- オプションの *namespace_string* は、*XPath_string* のネームスペース情報を提供します。このパラメータは、VARCHAR2 型である必要があります。

> **参照：** このファンクションの詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

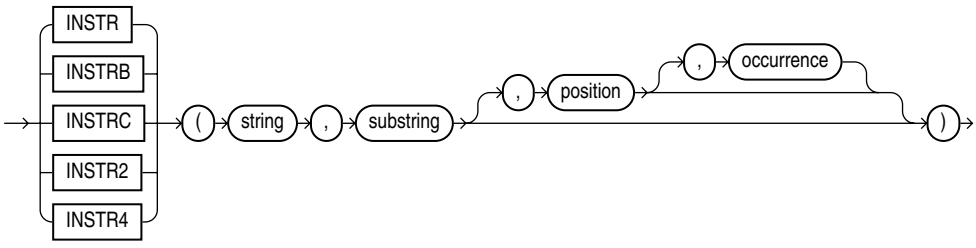
例

次の例は、5-79 ページの「INSERTCHILDXML」の例と似ていますが、この例では、「INSERTCHILDXML」の例で追加した /Owner ノードの前に、3 番目の /Owner ノードを追加します。問合せの出力は読み取りやすいように整えられています。

```
UPDATE warehouses SET warehouse_spec =
   INSERTXMLBEFORE(warehouse_spec,
   '/Warehouse/Building/Owner[2]',
   XMLType('<Owner>ThirdOwner</Owner>'))
   WHERE warehouse_id = 3;

SELECT warehouse_name, EXTRACT(warehouse_spec,
   '/Warehouse/Building/Owner') "Owners"
   FROM warehouses
   WHERE warehouse_id = 3;

Name         Owners
------------ --------------------------------------------------------------------
New Jersey <Owner>Grandco</Owner>
           <Owner>ThirdOwner</Owner>
           <Owner>LesserCo</Owner>
```

INSTR

構文



用途

INSTR ファンクションは、*string* の *substring* を検索します。このファンクションは、最初に現れた文字 *string* の位置を示す整数を戻します。INSTR は、入力キャラクタ・セットによって定義された文字を使用して、文字列を算出します。INSTRB は、文字のかわりにバイトを使用します。INSTRC は、完全な Unicode キャラクタを使用します。INSTR2 は、UCS2 コードポイントを使用します。INSTR4 は、UCS4 コードポイントを使用します。

- *position* は、Oracle Database が検索を開始する文字 *string* の位置を示す 0（ゼロ）以外の整数です。*position* が負の場合、Oracle は *string* の終わりから逆方向にカウントし、結果位置から逆方向に検索します。
- *occurrence* は、検索する *string* が現れたことを示す整数です。*occurrence* の値は正である必要があります。*occurrence* が 1 より大きい場合、データベースでは、最初に出現する *string* の 2 番目の文字以降で 2 番目の出現を検索します。

string および *substring* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻り値のデータ型は NUMBER です。

position および *occurrence* は、NUMBER データ型か、または暗黙的に NUMBER に変換可能な任意のデータ型で、整数に変換されるものである必要があります。*position* および *occurrence* のデフォルト値は 1 です。この場合、Oracle は *string* の最初の文字から検索を開始します。検索対象は *substring* が最初に現れる位置です。戻り値は、*position* の値にかかわらず、*string* の先頭に関係し、文字で表されます。検索が失敗した（*string* の *position* 番目の文字の後に *substring* が *occurrence* 回現れない）場合、戻り値は 0 となります。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

例

次の例では、文字列 CORPORATE FLOOR で文字列 OR の検索を 3 番目の文字から開始します。文字列 CORPORATE FLOOR で 2 回目に現れる「OR」の開始位置を戻します。

```
SELECT INSTR('CORPORATE FLOOR','OR', 3, 2)
  "Instring" FROM DUAL;

  Instring
----------
        14
```

次の例では、最後の文字から、最後から 3 番目の文字まで、つまり FLOOR の最初の O まで逆方向にカウントします。次に、2 回目に現れる「OR」を逆方向に検索し、2 回目に現れるこの文字列が、検索文字列の 2 番目の文字で始まることを認識します。

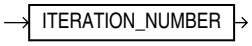
```
SELECT INSTR('CORPORATE FLOOR','OR', -3, 2)
"Reversed Instring"
     FROM DUAL;

Reversed Instring
-----------------
                2
```

次の例では、データベース・キャラクタ・セットがダブルバイトの場合を想定しています。

```
SELECT INSTRB('CORPORATE FLOOR','OR',5,2) "Instring in bytes"
   FROM DUAL;

Instring in bytes
-----------------
               27
```

ITERATION_NUMBER

構文



用途

ITERATION_NUMBER ファンクションは、SELECT 文の *model_clause* にのみ指定でき、*model_rules_clause* に ITERATE(*number*) が指定されている場合にのみ使用できます。このファンクションは、モデル・ルールに従って完了した反復を示す整数を戻します。最初の反復では 0（ゼロ）を戻します。2 回目以降の各反復では、*iteration_number* に 1 を足した整数を戻します。

> **参照：** 構文およびセマンティクスの詳細は、19-25 ページの「model_clause」および 6-11 ページの「モデル式」を参照してください。

例

次の例では、1998 年および 1999 年のマウス・パッドの売上を、それぞれ 2001 年および 2002 年のマウス・パッドの売上に割り当てます。

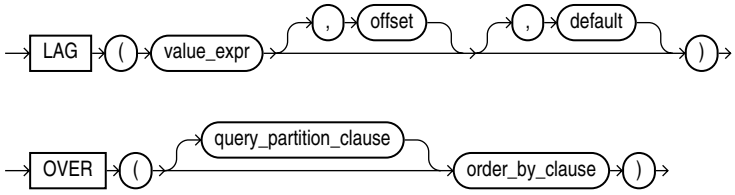
```
SELECT country, prod, year, s
  FROM sales_view_ref
  MODEL
    PARTITION BY (country)
    DIMENSION BY (prod, year)
    MEASURES (sale s)
    IGNORE NAV
    UNIQUE DIMENSION
    RULES UPSERT SEQUENTIAL ORDER ITERATE(2)
      (
        s['Mouse Pad', 2001 + ITERATION_NUMBER] =
          s['Mouse Pad', 1998 + ITERATION_NUMBER]
      )
  ORDER BY country, prod, year;
```

```
COUNTRY      PROD                                    YEAR          S
----------   ----------------------------------   --------   ---------
France       Mouse Pad                               1998     2509.42
France       Mouse Pad                               1999     3678.69
France       Mouse Pad                               2000     3000.72
France       Mouse Pad                               2001     2509.42
France       Mouse Pad                               2002     3678.69
France       Standard Mouse                          1998     2390.83
France       Standard Mouse                          1999     2280.45
France       Standard Mouse                          2000     1274.31
France       Standard Mouse                          2001     2164.54
Germany      Mouse Pad                               1998     5827.87
Germany      Mouse Pad                               1999     8346.44
Germany      Mouse Pad                               2000     7375.46
Germany      Mouse Pad                               2001     5827.87
Germany      Mouse Pad                               2002     8346.44
Germany      Standard Mouse                          1998     7116.11
Germany      Standard Mouse                          1999     6263.14
Germany      Standard Mouse                          2000     2637.31
Germany      Standard Mouse                          2001     6456.13

18 rows selected.
```

この例では、ビュー sales_view_ref が必要です。このビューを作成する方法については、19-36 ページの「MODEL 句の例 :」を参照してください。

LAG

構文



参照： 構文、セマンティクス、制限事項および *value_expr* の書式の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

LAG は分析ファンクションです。これは、自己結合せずに、表の 1 つ以上の行へ同時アクセスを行います。問合せから戻される一連の行およびカーソル位置を指定すると、LAG は、その位置より前にある指定された物理オフセットにある行へアクセスします。

offset を指定しない場合、デフォルト値は 1 です。オフセットがウィンドウの有効範囲を超えた場合、オプションの *default* 値が戻されます。*default* を指定しない場合、デフォルトは NULL です。

value_expr には、LAG または他の分析ファンクションを使用して分析ファンクションをネストできません。ただし、他の組込みファンクション式を *value_expr* で使用できます。

参照： *expr* の書式の詳細は、6-2 ページの「SQL 式」を参照してください。5-90 ページの「LEAD」も参照してください。

例

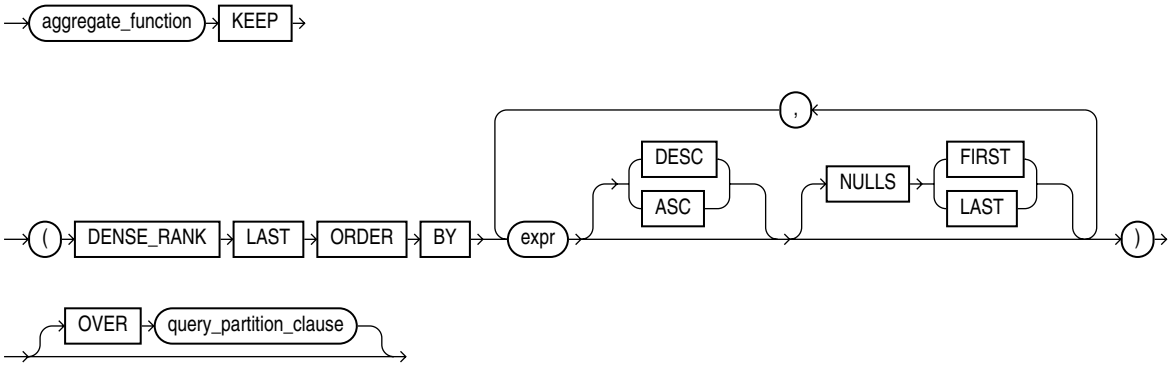
次の例では、employees 表の各販売員について、その販売員の直前に雇用された従業員の給与を示します。

```
SELECT last_name, hire_date, salary,
   LAG(salary, 1, 0) OVER (ORDER BY hire_date) AS prev_sal
   FROM employees
   WHERE job_id = 'PU_CLERK'
   ORDER BY last_name, hire_date, salary, prev_sal;

LAST_NAME                 HIRE_DATE     SALARY   PREV_SAL
------------------------- --------- ---------- ----------
Baida                     24-DEC-97       2900       2800
Colmenares                10-AUG-99       2500       2600
Himuro                    15-NOV-98       2600       2900
Khoo                      18-MAY-95       3100          0
Tobias                    24-JUL-97       2800       3100
```

LAST

構文

***last*::=**



参照： *query_partitioning_clause* の構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

FIRST ファンクションと LAST ファンクションは類似しています。両方のファンクションとも、与えられたソート指定に対して、FIRST または LAST としてランク付けされた一連の行の一連の値を操作する集計ファンクションおよび分析ファンクションです。1 つの行のみが FIRST または LAST としてランク付けされている場合、集計は、1 つの要素のみを持つセットを操作します。

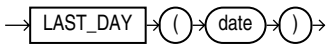
このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を戻します。

参照： 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

このファンクションの詳細および使用例は、5-71 ページの「FIRST」を参照してください。

LAST_DAY

構文



用途

LAST_DAY は、*date* を含む月の最終日の日付を戻します。月の最終日は、セッション・パラメータ NLS_CALENDAR によって定義されます。戻り型は、*date* のデータ型に関係なく常に DATE です。

例

次の文は、現在の月の残りの日数を確認します。

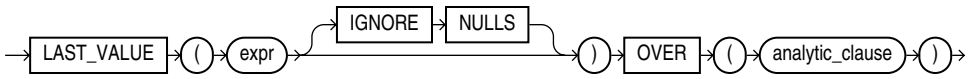
```
SELECT SYSDATE,
   LAST_DAY(SYSDATE) "Last",
   LAST_DAY(SYSDATE) - SYSDATE "Days Left"
   FROM DUAL;

SYSDATE   Last       Days Left
--------- --------- ----------
30-MAY-01 31-MAY-01          1
```

次の例では、各従業員の雇用開始日に 5 か月を加えて、評価日付を戻します。

```
SELECT last_name, hire_date, TO_CHAR(
   ADD_MONTHS(LAST_DAY(hire_date), 5)) "Eval Date"
   FROM employees;

LAST_NAME                 HIRE_DATE Eval Date
------------------------- --------- ---------
King                      17-JUN-87 30-NOV-87
Kochhar                   21-SEP-89 28-FEB-90
De Haan                   13-JAN-93 30-JUN-93
Hunold                    03-JAN-90 30-JUN-90
Ernst                     21-MAY-91 31-OCT-91
Austin                    25-JUN-97 30-NOV-97
Pataballa                 05-FEB-98 31-JUL-98
Lorentz                   07-FEB-99 31-JUL-99
. . .
```

LAST_VALUE

構文



参照： 構文、セマンティクス、制限事項、および *expr* の書式の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

LAST_VALUE は分析ファンクションです。これは、順序付けられた値の集合にある最後の値を戻します。集合内の最後の値が NULL の場合、IGNORE NULLS を指定していないかぎり、ファンクションは NULL を戻します。この設定は、データの稠密化に役立ちます。IGNORE NULLS を指定すると、LAST_VALUE は集合内の最初の NULL ではない値を戻します。すべての値が NULL の場合は NULL を戻します。データの稠密化の例は、19-43 ページの「パーティション化された外部結合の使用例 :」を参照してください。

expr には、LAST_VALUE または他の分析ファンクションを使用して分析ファンクションをネストできません。ただし、他の組込みファンクション式を *expr* で使用できます。*expr* の書式の詳細は、6-2 ページの「SQL 式」を参照してください。

analytic_clause の *windowing_clause* を省略した場合、デフォルトで RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW になります。ウィンドウの最後の値はウィンドウの下部にあり、修復されないため、このデフォルトは予期しない値を戻す場合があります。変更は、現行の行の変更として保持されます。正しい結果を得るには、*windowing_clause* を RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING として指定します。または、*windowing_clause* を RANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING として指定することもできます。

例

次の例では、給与が一番高い従業員の雇用開始日を各行に戻します。

```
SELECT last_name, salary, hire_date, LAST_VALUE(hire_date) OVER
   (ORDER BY salary
   ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS lv
FROM (SELECT * FROM employees WHERE department_id = 90
   ORDER BY hire_date)
ORDER BY last_name, salary, hire_date, lv;

LAST_NAME           SALARY HIRE_DATE LV
--------------- ---------- --------- ---------
De Haan              17000 13-JAN-93 17-JUN-87
King                 24000 17-JUN-87 17-JUN-87
Kochhar              17000 21-SEP-89 17-JUN-87
```

この例では、LAST_VALUE ファンクションの非決定的な性質を示しています。Kochhar と De Haan の給与は同じであるため、Kochhar の次の行に De Haan があります。Kochhar が最初に表示されているのは、副問合せの行が hire_date で順序付けられているためです。ただし、行が hire_date で降順に順序付けられている場合は、次の例に示すとおり、ファンクションは異なる値を戻します。

```
SELECT last_name, salary, hire_date, LAST_VALUE(hire_date) OVER
   (ORDER BY salary
   ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS lv
FROM (SELECT * FROM employees WHERE department_id = 90
   ORDER BY hire_date DESC)
ORDER BY last_name, salary, hire_datea, lv;

LAST_NAME           SALARY HIRE_DATE LV
--------------- ---------- --------- ---------
De Haan              17000 13-JAN-93 17-JUN-87
King                 24000 17-JUN-87 17-JUN-87
Kochhar              17000 21-SEP-89 17-JUN-87
```

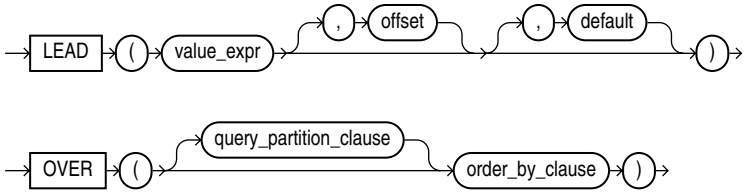
次の 2 つの例では、一意キーで順序付けることによって、LAST_VALUE ファンクションを決定的にする方法を示しています。salary および hire_date でファンクション内を順序付けると、副問合せの順序付けにかかわらず、同じ結果が戻されます。

```
SELECT last_name, salary, hire_date, LAST_VALUE(hire_date) OVER
(ORDER BY salary, hire_date
  ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS lv
FROM (SELECT * FROM employees WHERE department_id = 90
   ORDER BY hire_date)
ORDER BY last_name, salary, hire_date, lv;

LAST_NAME           SALARY HIRE_DATE LV
--------------- ---------- --------- ---------
De Haan              17000 13-JAN-93 17-JUN-87
King                 24000 17-JUN-87 17-JUN-87
Kochhar              17000 21-SEP-89 17-JUN-87

SELECT last_name, salary, hire_date, LAST_VALUE(hire_date) OVER
   (ORDER BY salary, hire_date
```

```
   ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS lv
FROM (SELECT * FROM employees WHERE department_id = 90
   ORDER BY hire_date DESC)
ORDER BY last_name, salary, hire_date, lv;

LAST_NAME           SALARY HIRE_DATE LV
--------------- ---------- --------- ---------
De Haan              17000 13-JAN-93 17-JUN-87
King                 24000 17-JUN-87 17-JUN-87
Kochhar              17000 21-SEP-89 17-JUN-87
```

LEAD

構文



参照： 構文、セマンティクス、制限事項、および *value_expr* の書式の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

LEAD は分析ファンクションです。これは、自己結合せずに、表の 1 つ以上の行へ同時アクセスを行います。問合せから戻される一連の行およびカーソル位置を指定すると、LEAD は、その位置より後にある指定された物理オフセットの行へアクセスします。

offset を指定しない場合、デフォルト値は 1 です。オフセットが表の有効範囲を超えた場合、オプションの *default* 値が戻されます。*default* を指定しない場合、デフォルト値は NULL です。

value_expr には、LEAD または他の分析ファンクションを使用して分析ファンクションをネストできません。ただし、他の組込みファンクション式を *value_expr* で使用できます。

参照： *expr* の書式の詳細は、6-2 ページの「SQL 式」を参照してください。5-86 ページの「LAG」も参照してください。

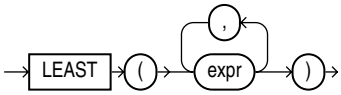
例

次の例では、employees 表の各従業員について、その従業員の直後に雇用された従業員の雇用開始日を示します。

```
SELECT last_name, hire_date,
   LEAD(hire_date, 1) OVER (ORDER BY hire_date) AS "NextHired"
   FROM employees WHERE department_id = 30
   ORDER BY last_name, hire_date, "NextHired";

LAST_NAME                 HIRE_DATE NextHired
------------------------- --------- ---------
Baida                     24-DEC-97 15-NOV-98
Colmenares                10-AUG-99
Himuro                    15-NOV-98 10-AUG-99
Khoo                      18-MAY-95 24-JUL-97
Raphaely                  07-DEC-94 18-MAY-95
Tobias                    24-JUL-97 24-DEC-97
```

LEAST

構文



用途

LEAST は、リストされた *expr* 内の最小値を戻します。比較の前に、2 番目以降のすべての *expr* は最初の *expr* のデータ型に暗黙的に変換されます。Oracle Database は、非空白埋め比較セマンティクスを使用して *expr* を比較します。このファンクションによって戻される値が文字データの場合、そのデータ型は常に VARCHAR2 になります。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。2 進浮動小数点の比較セマンティクスの詳細は、2-13 ページの「浮動小数点数」を参照してください。2-36 ページの「データ型の比較規則」も参照してください。

例

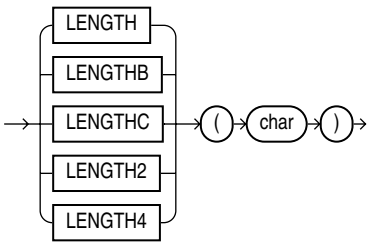
次の文では、最小値を持つ文字列を検索します。

```
SELECT LEAST('HARRY','HARRIOT','HAROLD') "LEAST"
     FROM DUAL;

LEAST
------
HAROLD
```

LENGTH

構文

***length*::=**



用途

LENGTH の各ファンクションは、*char* の長さを戻します。LENGTH は、入力キャラクタ・セットによって定義された文字を使用して、長さを算出します。LENGTHB は、文字のかわりにバイトを使用します。LENGTHC は、完全な Unicode キャラクタを使用します。LENGTH2 は、UCS2 コードポイントを使用します。LENGTH4 は、UCS4 コードポイントを使用します。

char は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻り値のデータ型は NUMBER です。*char* のデータ型が CHAR の場合、その長さにはすべての後続空白が含まれます。*char* が NULL の場合、このファンクションは NULL を戻します。

LENGTHB の制限事項 LENGTHB ファンクションは、シングルバイトの LOB でのみサポートされます。このファンクションは、マルチバイトのキャラクタ・セットの CLOB と NCLOB では使用できません。

例

次の例では、シングルバイトおよびマルチバイトのデータベース・キャラクタ・セットを使用する LENGTH ファンクションを使用します。

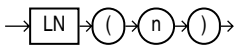
```
SELECT LENGTH('CANDIDE') "Length in characters"
   FROM DUAL;

Length in characters
--------------------
                   7
```

次の例では、データベース・キャラクタ・セットがダブルバイトの場合を想定しています。

```
SELECT LENGTHB ('CANDIDE') "Length in bytes"
   FROM DUAL;

Length in bytes
---------------
             14
```

LN

構文



用途

LN は、*n* の自然対数を戻します（*n* は正の数）。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数が BINARY_FLOAT の場合、このファンクションは BINARY_DOUBLE を戻します。それ以外の場合、引数と同じ数値データ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

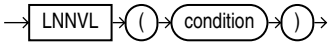
例

次の例では、95 の自然対数を戻します。

```
SELECT LN(95) "Natural log of 95" FROM DUAL;

Natural log of 95
-----------------
       4.55387689
```

LNNVL

構文



用途

LNNVL は、条件のオペランドの 1 つまたは両方が NULL の可能性がある場合にその条件を簡単に評価する方法を提供します。このファンクションは、問合せの WHERE 句でのみ使用できます。このファンクションは、引数として条件を取り、その条件が FALSE または UNKNOWN の場合は TRUE を戻し、TRUE の場合は FALSE を戻します。LNNVL は、スカラー式を指定できる場所であればどこでも指定でき、有効ではないが、発生する可能性がある NULL を評価するために IS [NOT] NULL、AND または OR 条件が必要であるようなコンテキストでも指定できます。

Oracle Database は、LNNVL ファンクションをこの方法で内部的に使用して、NOT IN 条件を NOT EXISTS 条件として書き換える場合があります。この場合、EXPLAIN PLAN からの出力によって、この操作が PLAN TABLE に出力されます。*condition* では、どのようなスカラー値でも評価できますが、AND、OR または BETWEEN を含む複合条件は指定できません。

a = 2 および b=NULL の場合の LNNVL からの戻り値を次の表に示します。

条件	条件の真偽	LNNVL の戻り値
a = 1	FALSE	TRUE
a = 2	TRUE	FALSE
a IS NULL	FALSE	TRUE
b = 1	UNKNOWN	TRUE
b IS NULL	TRUE	FALSE
a = b	UNKNOWN	TRUE

例

歩合を受け取らない従業員を含めて、歩合率が 20% 未満の従業員数を調べるとします。次の問合せは、20% 未満の歩合を実際に受け取る従業員のみを戻します。

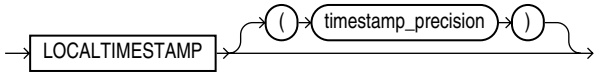
```
SELECT COUNT(*) FROM employees WHERE commission_pct < .2;

  COUNT(*)
----------
        11
```

歩合を受け取らない 72 人の従業員も含めるには、LNNVL ファンクションを次のように使用して、この問合せを書き換えます。

```
SELECT COUNT(*) FROM employees WHERE LNNVL(commission_pct >= .2);

  COUNT(*)
----------
        83
```

LOCALTIMESTAMP

構文



用途

LOCALTIMESTAMP は、セッションのタイムゾーンの現在の日付および時刻を TIMESTAMP データ型の値で戻します。これに対して、CURRENT_TIMESTAMP は、TIMESTAMP WITH TIME ZONE 値を戻します。

オプションの引数 *timestamp_precision* には、戻される時刻値の秒の小数部の精度を指定します。

参照： 「CURRENT_TIMESTAMP」（5-50 ページ）

例

次の例では、LOCALTIMESTAMP と CURRENT_TIMESTAMP の相違を示します。

```
ALTER SESSION SET TIME_ZONE = '-5:00';
SELECT CURRENT_TIMESTAMP, LOCALTIMESTAMP FROM DUAL;

CURRENT_TIMESTAMP                       LOCALTIMESTAMP
--------------------------------------------------------------------
04-APR-00 01.27.18.999220 PM -05:00  04-APR-00 01.27.19 PM

ALTER SESSION SET TIME_ZONE = '-8:00';
SELECT CURRENT_TIMESTAMP, LOCALTIMESTAMP FROM DUAL;

CURRENT_TIMESTAMP                       LOCALTIMESTAMP
----------------------------------  ----------------------------
04-APR-00 10.27.45.132474 AM -08:00  04-APR-00 10.27.451 AM
```

LOCALTIMESTAMP で書式マスクを使用する場合は、ファンクションが戻す値と書式マスクを一致させてください。たとえば、次の表の場合を考えます。

```
CREATE TABLE local_test (col1 TIMESTAMP WITH LOCAL TIME ZONE);
```

ファンクションが戻す型の TIME ZONE の部分がマスクに含まれていないため、次の文は正常に実行されません。

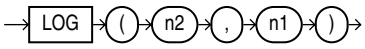
```
INSERT INTO local_test VALUES
   (TO_TIMESTAMP(LOCALTIMESTAMP, 'DD-MON-RR HH.MI.SSXFF'));
```

次の文では、LOCALTIMESTAMP の戻り値の型と一致する正しい書式マスクが使用されています。

```
INSERT INTO local_test VALUES
   (TO_TIMESTAMP(LOCALTIMESTAMP, 'DD-MON-RR HH.MI.SSXFF PM'));
```

LOG

構文



用途

LOG は、*n2* を底とする *n1* の対数を戻します。底 *n1* は 0（ゼロ）または 1 以外の任意の正の値で、*n2* は任意の正の値です。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。いずれかの引数が BINARY_FLOAT または BINARY_DOUBLE の場合、このファンクションは BINARY_DOUBLE を戻します。それ以外の場合、NUMBER を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

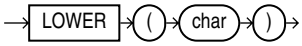
例

次の例では、100 の対数を戻します。

```
SELECT LOG(10,100) "Log base 10 of 100" FROM DUAL;

Log base 10 of 100
------------------
                 2
```

LOWER

構文



用途

LOWER は、すべての文字を小文字にして *char* を戻します。*char* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻り値は、*char* と同じデータ型です。データベースは、基礎となるキャラクタ・セットに対して定義したバイナリ・マッピングに基づいて文字の形式を設定します。小文字の区別については、5-109 ページの「NLS_LOWER」を参照してください。

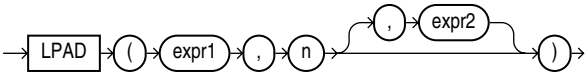
例

次の例では、文字列を小文字にして戻します。

```
SELECT LOWER('MR. SCOTT MCMILLAN') "Lowercase"
   FROM DUAL;

Lowercase
--------------------
mr. scott mcmillan
```

LPAD

構文



用途

LPAD は、*expr1* の左側に *expr2* に指定した文字を連続的に埋め込んで *n* 桁にして戻します。このファンクションは、問合せの出力の書式設定に役立ちます。

expr1 および *expr2* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻される文字列は、*expr1* が文字データ型の場合は VARCHAR2 データ型、*expr1* が各国語キャラクタ・データ型の場合は NVARCHAR2 データ型、*expr1* が LOB データ型の場合は LOB データ型になります。*expr1* と同じキャラクタ・セットの文字列が戻されます。引数 *n* は、NUMBER 型の整数か、または NUMBER 型の整数に暗黙的に変換可能な値である必要があります。

expr2 を指定しない場合、デフォルトで空白 1 個が指定されます。*expr1* が *n* より長い場合、このファンクションは *n* に収まる *expr1* の一部を戻します。

引数 *n* は、戻り値が画面に表示される場合の全体の長さです。多くのキャラクタ・セットでは、これは戻り値の文字数でもあります。ただし、マルチバイトのキャラクタ・セットでは、表示される文字列の長さが文字列の文字数と異なる場合もあります。

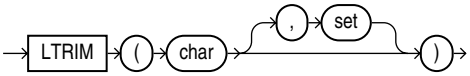
例

次の例では、文字列の左側にアスタリスクとピリオド（*.）を埋め込みます。

```
SELECT LPAD('Page 1',15,'*.') "LPAD example"
   FROM DUAL;

LPAD example
---------------
*.*.*.*.*Page 1
```

LTRIM

構文



用途

LTRIM は、*char* の左端から、*set* に指定されたすべての文字を削除します。*set* を指定しない場合、デフォルトで空白 1 個が指定されます。*char* が文字リテラルの場合、一重引用符で囲む必要があります。Oracle Database は *char* の先頭文字からスキャンし始め、*set* に指定された文字をすべて削除します。*set* に指定された文字以外の文字が見つかった時点で結果を戻します。

char および *set* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻される文字列は、*char* が文字データ型の場合は VARCHAR2 データ型、*char* が各国語キャラクタデータ型の場合は NVARCHAR2 データ型、*char* が LOB データ型の場合は LOB データ型になります。

参照： 「RTRIM」（5-161 ページ）

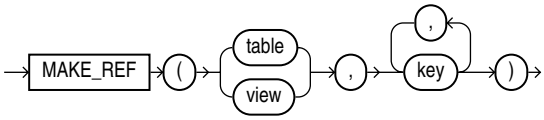
例

次の例では、oe.products 表内の製品名のグループから、冗長な最初の語を削除します。

```
SELECT product_name, LTRIM(product_name, 'Monitor ') "Short Name"
   FROM products
   WHERE product_name LIKE 'Monitor%';

PRODUCT_NAME         Short Name
-------------------- ---------------
Monitor 17/HR        17/HR
Monitor 17/HR/F      17/HR/F
Monitor 17/SD        17/SD
Monitor 19/SD        19/SD
Monitor 19/SD/M      19/SD/M
Monitor 21/D         21/D
Monitor 21/HR        21/HR
Monitor 21/HR/M      21/HR/M
Monitor 21/SD        21/SD
Monitor Hinge - HD   Hinge - HD
Monitor Hinge - STD  Hinge - STD
```

MAKE_REF

構文



用途

MAKE_REF は、オブジェクト識別子が主キーに基づいている、オブジェクト・ビューの行またはオブジェクト表の行に対する REF を作成します。このファンクションは、オブジェクト・ビューを作成する場合などに有効です。

> **参照：** オブジェクト・ビューの詳細は、『Oracle Database オブジェクト・リレーショナル開発者ガイド』を参照してください。5-58 ページの「DEREF」も参照してください。

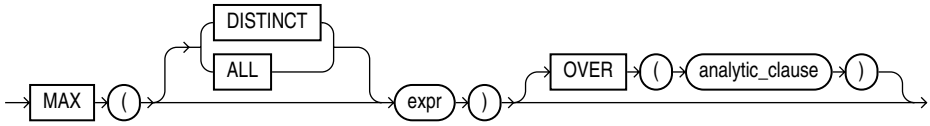
例

サンプル・スキーマ oe には、inventory_typ に基づくオブジェクト・ビュー oc_inventories が格納されています。オブジェクト識別子は product_id です。次の例では、3003 という product_id を含むオブジェクト・ビュー oc_inventories にある行への REF を作成します。

```
SELECT MAKE_REF (oc_inventories, 3003) FROM DUAL;

MAKE_REF(OC_INVENTORIES,3003)
--------------------------------------------------------------------
00004A038A0046857C14617141109EE034080020825436000000142601000100010
0029090606002A00078401FE0000000B03C21F040000000000000000000000000000
0000000000
```

MAX

構文



参照： 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

MAX は *expr* の最大値を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

参照： *expr* の書式の詳細は、6-2 ページの「SQL 式」を参照してください。2 進浮動小数点の比較セマンティクスの詳細は、2-13 ページの「浮動小数点数」を参照してください。5-8 ページの「集計ファンクション」も参照してください。

集計の例

次の例では、hr.employees 表の最高給与を検索します。

```
SELECT MAX(salary) "Maximum" FROM employees;

   Maximum
----------
      24000
```

分析の例

次の例では、各従業員について、その従業員と所属が同じ従業員のうち、一番高い給与を計算します。

```
SELECT manager_id, last_name, salary,
   MAX(salary) OVER (PARTITION BY manager_id) AS mgr_max
   FROM employees
   ORDER BY manager_id, last_name, salary, mgr_max;

MANAGER_ID LAST_NAME                 SALARY    MGR_MAX
---------- ------------------------- ---------- ----------
       100 Cambrault                  11000      17000
       100 De Haan                    17000      17000
       100 Errazuriz                  12000      17000
       100 Fripp                       8200      17000
       100 Hartstein                  13000      17000
       100 Kaufling                    7900      17000
       100 Kochhar                    17000      17000
. . .
```

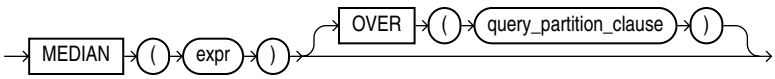
この問合せを述語のある親問合せで囲むと、各部門で給与の一番高い従業員がわかります。

```
SELECT manager_id, last_name, salary
   FROM (SELECT manager_id, last_name, salary,
      MAX(salary) OVER (PARTITION BY manager_id) AS rmax_sal
      FROM employees) WHERE salary = rmax_sal
   ORDER BY manager_id, last_name, salary;
```

```
MANAGER_ID LAST_NAME                 SALARY
---------- ------------------------- ----------
       100 De Haan                        17000
       100 Kochhar                        17000
       101 Greenberg                      12000
       101 Higgins                        12000
       102 Hunold                          9000
       103 Ernst                           6000
       108 Faviet                          9000
       114 Khoo                            3100
       120 Nayer                           3200
       120 Taylor                          3200
       121 Sarchand                        4200
       122 Chung                           3800
       123 Bell                            4000
       124 Rajs                            3500
       145 Tucker                         10000
       146 King                           10000
       147 Vishney                        10500
       148 Ozer                           11500
       149 Abel                           11000
       201 Fay                             6000
       205 Gietz                           8300
           King                           24000

22 rows selected.
```

MEDIAN

構文



参照： 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

MEDIAN は、連続分散モデルを想定する逆分散関数です。このファンクションは、数値または日時値を取り、その中央値、または値のソート後に中央値となる補間された値を戻します。計算では、NULL は無視されます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。*expr* のみを指定した場合、このファンクションは引数の数値データ型と同じデータ型を戻します。OVER 句を指定した場合、Oracle Database は、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

参照： 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。数値の優先順位の詳細は、2-15 ページの「数値の優先順位」を参照してください。

MEDIAN の結果の計算では、まず行が順序付けされます。グループ内の行数として N が使用され、対象の行番号（RN）が RN = (1 + (0.5 × (N-1)) という式で計算されます。集計ファンクションの最終結果は、行番号が CRN = CEILING(RN) および FRN = FLOOR(RN) の行の値間の直線補間によって計算されます。

最終結果は次のとおりです。

```
   if (CRN = FRN = RN) then
      (value of expression from row at RN)
   else
      (CRN - RN) * (value of expression for row at FRN) +
      (RN - FRN) * (value of expression for row at CRN)
```

MEDIAN は、分析ファンクションとして使用できます。その場合、OVER 句には、*query_partition_clause* のみを指定できます。各行に対して、各パーティション内の一連の値の中央値に該当する値が戻されます。

このファンクションと次のファンクションを比較してください。

- PERCENTILE_CONT（5-119 ページ）: 指定したパーセンタイルに一致するように補間された値が戻されます。MEDIAN は、パーセンタイル値がデフォルトで 0.5 に指定される特別な PERCENTILE_CONT です。
- PERCENTILE_DISC（5-121 ページ）: 指定したパーセンタイルに一致する値を補間なしで検索する場合に役立ちます。

集計の例

次の問合せは、hr.employees 表内の各従業員の給与の中央値を戻します。

```
SELECT department_id, MEDIAN(salary)
   FROM employees
   GROUP BY department_id
   ORDER BY department_id, median(salary);

DEPARTMENT_ID MEDIAN(SALARY)
------------- --------------
           10           4400
           20           9500
           30           2850
           40           6500
           50           3100
           60           4800
           70          10000
           80           8900
           90          17000
          100           8000
          110          10150
                        7000
```

分析の例

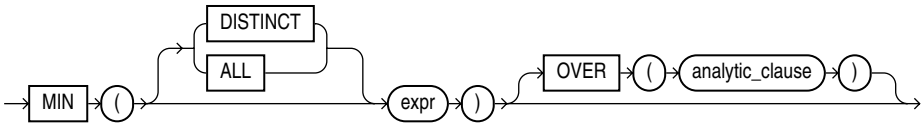
次の問合せは、hr.employees 表内の部門のサブセットのマネージャごとに給与の中央値を戻します。

```
SELECT manager_id, employee_id, salary,
   MEDIAN(salary) OVER (PARTITION BY manager_id) "Median by Mgr"
   FROM employees
   WHERE department_id > 60
   ORDER BY manager_id, employee_id, salary, "Median by Mgr";

MANAGER_ID EMPLOYEE_ID     SALARY Median by Mgr
---------- ----------- ---------- -------------
       100         101      17000         13500
       100         102      17000         13500
       100         145      14000         13500
       100         146      13500         13500
       100         147      12000         13500
       100         148      11000         13500
       100         149      10500         13500
```

```
       101         108       12000         12000
       101         204       10000         12000
       101         205       12000         12000
       108         109        9000          7800
       108         110        8200          7800
       108         111        7700          7800
       108         112        7800          7800
       108         113        6900          7800
       145         150       10000          8500
       145         151        9500          8500
       145         152        9000          8500
. . .
```

MIN

構文



参照： 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

MIN は、*expr* の最小値を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

参照： *expr* の書式の詳細は、6-2 ページの「SQL 式」を参照してください。2 進浮動小数点の比較セマンティクスの詳細は、2-13 ページの「浮動小数点数」を参照してください。5-8 ページの「集計ファンクション」も参照してください。

集計の例

次の例では、hr.employees 表の最初の雇用開始日を戻します。

```
SELECT MIN(hire_date) "Earliest" FROM employees;

Earliest
---------
17-JUN-87
```

分析の例

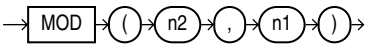
次の例では、各従業員について、その従業員が雇用された日以前に雇用された従業員を検索します。その従業員と所属が同じ従業員のサブセットを決定し、そのサブセット内で一番低い給与を戻します。

```
SELECT manager_id, last_name, hire_date, salary,
   MIN(salary) OVER(PARTITION BY manager_id ORDER BY hire_date
   RANGE UNBOUNDED PRECEDING) AS p_cmin
   FROM employees
   ORDER BY manager_id, last_name, hire_date, salary, p_cmin;

MANAGER_ID LAST_NAME                 HIRE_DATE     SALARY     P_CMIN
---------- ------------------------- --------- ---------- ----------
       100 Cambrault                 15-OCT-99      11000       6500
       100 De Haan                   13-JAN-93      17000      17000
```

```
       100 Errazuriz                 10-MAR-97      12000       7900
       100 Fripp                     10-APR-97       8200       7900
       100 Hartstein                 17-FEB-96      13000       7900
       100 Kaufling                  01-MAY-95       7900       7900
       100 Kochhar                   21-SEP-89      17000      17000
       100 Mourgos                   16-NOV-99       5800       5800
       100 Partners                  05-JAN-97      13500       7900
       100 Raphaely                  07-DEC-94      11000      11000
       100 Russell                   01-OCT-96      14000       7900
. . .
```

MOD

構文



用途

MOD は *n2* を *n1* で割った余りを戻します。*n1* が 0 の場合は、*n2* を戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。Oracle は、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。数値の優先順位の詳細は、2-15 ページの「数値の優先順位」を参照してください。

例

次の例では、11 を 4 で割ったときの余りを戻します。

```
SELECT MOD(11,4) "Modulus" FROM DUAL;

   Modulus
----------
         3
```

このファンクションは、*m* が負の場合には古典数学のモジュール・ファンクションとは異なる動作をします。古典数学のモジュール・ファンクションは、次の公式を用いた MOD ファンクションで表すことができます。

```
m - n * FLOOR(m/n)
```

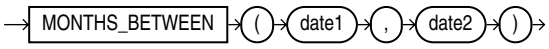
次の表に、MOD ファンクションと古典数学のモジュール・ファンクションの相違を示します。

m	n	MOD(m,n)	古典数学のモジュール・ファンクション
11	4	3	3
11	-4	3	-1
-11	4	-3	1
-11	-4	-3	-3

> **参照：** 5-74 ページの「FLOOR」および 5-155 ページの「REMAINDER」を参照してください。REMAINDER は MOD と似ていますが、FLOOR ではなく ROUND を式に使用します。

MONTHS_BETWEEN

構文



用途

MONTHS_BETWEEN は、日付 *date1* と *date2* の間の月数を戻します。月および月の最終日は、パラメータ NLS_CALENDAR によって定義されます。*date1* が *date2* より後の日付の場合、結果は正の値になります。*date1* が *date2* より前の日付の場合、結果は負の値になります。*date1* および *date2* が、月の同じ日または月の最終日の場合、結果は常に整数になります。それ以外の場合、Oracle Database は結果の小数部を 1 か月 31 日として計算し、*date1* と *date2* の差を割り出します。

例

次の例では、2 つの日付間の月数を計算します。

```
SELECT MONTHS_BETWEEN
   (TO_DATE('02-02-1995','MM-DD-YYYY'),
    TO_DATE('01-01-1995','MM-DD-YYYY') ) "Months"
    FROM DUAL;

    Months
----------
1.03225806
```

NANVL

構文

用途

NANVL ファンクションは、BINARY_FLOAT 型または BINARY_DOUBLE 型の浮動小数点数のみに有効です。このファンクションは、入力値 *n2* が NaN（非数値）の場合は代替値 *n1* を戻すように Oracle Database に指示します。*n2* が NaN **でない**場合、Oracle は *n2* を戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。Oracle は、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。2 進浮動小数点の比較セマンティクスの詳細は、2-13 ページの「浮動小数点数」を参照してください。数値の優先順位詳細は、2-15 ページの「数値の優先順位」を参照してください。

例

5-196 ページの「TO_BINARY_DOUBLE」で作成した float_point_demo を使用して、表に 2 つ目のエントリを挿入します。

```
Insert INTO float_point_demo
  VALUES (0,'NaN','NaN');

SELECT * FROM float_point_demo;
```

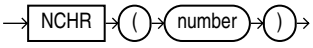
```
   DEC_NUM BIN_DOUBLE  BIN_FLOAT
---------- ---------- ----------
   1234.56 1.235E+003 1.235E+003
         0        Nan        Nan
```

次の例では、数値の場合は *bin_float* を戻します。数値以外の場合は、0（ゼロ）を戻します。

```
SELECT bin_float, NANVL(bin_float,0)
  FROM float_point_demo;

 BIN_FLOAT NANVL(BIN_FLOAT,0)
---------- ------------------
1.235E+003         1.235E+003
       Nan                  0
```

NCHR

構文



用途

NCHR は、各国語キャラクタ・セットの *number* と同等のバイナリを持つ文字を戻します。戻り値は常に NVARCHAR2 です。このファンクションは、CHR ファンクションに USING NCHAR_CS 句を指定して使用した場合と同じ結果を戻します。

このファンクションは、引数として、NUMBER 値、または暗黙的に NUMBER 型に変換可能な任意の値を取り、文字を戻します。

参照： 「CHR」（5-29 ページ）

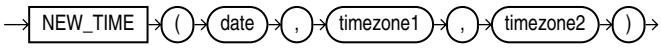
例

次の例では、NCHAR 文字 187 を戻します。

```
SELECT NCHR(187) FROM DUAL;

NC
--
>

SELECT CHR(187 USING NCHAR_CS) FROM DUAL;

CH
--
>
```

NEW_TIME

構文



用途

NEW_TIME は、タイムゾーン *timezone1* の日時が *date* の時点のタイムゾーン *timezone2* の日時を戻します。このファンクションを使用する前に、24 時間で表示されるように、NLS_DATE_FORMAT パラメータを設定する必要があります。戻り型は、*date* のデータ型に関係なく常に DATE です。

> **注意：** このファンクションが入力として取ることのできるタイムゾーンの数には制限があります。FROM_TZ ファンクションと日時式を組み合せることによって、より多くのタイムゾーンにアクセスできます。5-74 ページの「FROM_TZ」および 6-8 ページの「日時式」の例を参照してください。

引数 *timezone1* および *timezone2* には、次のテキスト文字列のいずれかを指定できます。

- AST、ADT: 大西洋標準時および大西洋夏時間
- BST、BDT: ベーリング標準時およびベーリング夏時間
- CST、CDT: 中央標準時および中央夏時間
- EST、EDT: 東部標準時および東部夏時間
- GMT: グリニッジ標準時
- HST、HDT: アラスカ - ハワイ標準時およびアラスカ - ハワイ夏時間
- MST、MDT: 山岳部標準時および東部夏時間
- NST: ニューファンドランド標準時
- PST、PDT: 太平洋標準時および太平洋夏時間
- YST、YDT: ユーコン標準時およびユーコン夏時間

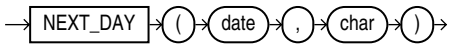
例

次の例では、指定された太平洋標準時と同等の大西洋標準時を戻します。

```
ALTER SESSION SET NLS_DATE_FORMAT =
   'DD-MON-YYYY HH24:MI:SS';

SELECT NEW_TIME(TO_DATE(
   '11-10-99 01:23:45', 'MM-DD-YY HH24:MI:SS'),
   'AST', 'PST') "New Date and Time" FROM DUAL;

New Date and Time
--------------------
09-NOV-1999 21:23:45
```

NEXT_DAY

構文



用途

NEXT_DAY は、*char* で指定した曜日で、日付 *date* より後の最初の日付を戻します。戻り型は、*date* のデータ型に関係なく常に DATE です。引数 *char* は、セッションの日付言語での曜日である必要があります（フルネームでも省略形でも可）。必要最小限の文字数は、省略形の文字数です。有効な省略形の後に続けて文字が入力されていても、それらの文字は無視されます。戻り値は、引数 *date* と同じ時、分および秒のコンポーネントを持っています。

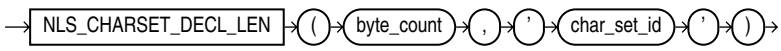
例

次の例では、2001 年 2 月 2 日以降の最初の火曜日の日付を戻します。

```
SELECT NEXT_DAY('02-FEB-2001','TUESDAY') "NEXT DAY"
     FROM DUAL;

NEXT DAY
-----------
06-FEB-2001
```

NLS_CHARSET_DECL_LEN

構文



用途

NLS_CHARSET_DECL_LEN は、NCHAR 列の宣言の長さを（文字数で）戻します。*byte_count* 引数は、列の幅です。*char_set_id* 引数は、列のキャラクタ・セット ID です。

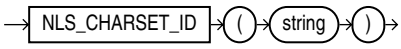
例

次の例では、マルチバイト・キャラクタ・セットを使用している場合に、列の幅が 200 バイトである文字の数を戻します。

```
SELECT NLS_CHARSET_DECL_LEN
  (200, nls_charset_id('ja16eucfixed'))
   FROM DUAL;

NLS_CHARSET_DECL_LEN(200,NLS_CHARSET_ID('JA16EUCFIXED'))
-------------------------------------------------------
                                                    100
```

NLS_CHARSET_ID

構文



用途

NLS_CHARSET_ID は、キャラクタ・セット名 *string* に対応するキャラクタ・セットの ID 番号を戻します。引数 *string* は、実行時の VARCHAR2 値です。*string* 値 'CHAR_CS' は、サーバーのデータベース・キャラクタ・セットの ID 番号を戻します。*string* 値 'NCHAR_CS' は、サーバーの各国語キャラクタ・セットの ID 番号を戻します。

無効なキャラクタ・セット名を指定すると、NULL が戻されます。

例

次の例では、キャラクタ・セットの ID を戻します。

```
SELECT NLS_CHARSET_ID('ja16euc')
  FROM DUAL;

NLS_CHARSET_ID('JA16EUC')
-------------------------
                      830
```

参照： キャラクタ・セット名のリストについては、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

NLS_CHARSET_NAME

構文

用途

NLS_CHARSET_NAME は、ID 番号 *number* に対応するキャラクタ・セット名を戻します。キャラクタ・セット名は、データベース・キャラクタ・セットの VARCHAR2 値として戻されます。

number が有効なキャラクタ・セット ID として認識されない場合は、このファンクションは NULL を戻します。

例

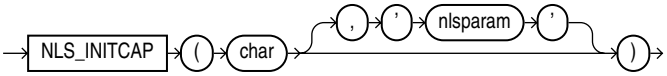
次の例では、キャラクタ・セットの ID 番号 2 に対応するキャラクタ・セットを戻します。

```
SELECT NLS_CHARSET_NAME(2)
  FROM DUAL;

NLS_CH
------
WE8DEC
```

参照： キャラクタ・セット ID のリストについては、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

NLS_INITCAP

構文



用途

NLS_INITCAP は、各単語の最初の文字を大文字、残りの文字を小文字にして *char* を戻します。単語は空白または英数字以外の文字で区切ります。

char および *'nlsparam'* は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型です。戻される文字列は、VARCHAR2 データ型であり、*char* と同じキャラクタ・セットです。

'nlsparam' の値は次の書式で指定します。

```
'NLS_SORT = sort'
```

sort は、言語ソート基準または BINARY のいずれかです。言語ソート基準は、大文字と小文字の変換のために特別な言語要件を処理します。これらの要件によって、*char* と異なる長さの値が戻される場合があります。*'nlsparam'* を省略すると、このファンクションはセッションに対してデフォルトのソート基準を使用します。

このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。

> **参照：** 詳細は、2-36 ページの「データ型の比較規則」を参照してください。

例

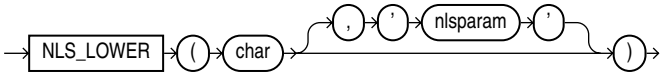
次に、ファンクションによって言語ソート基準がどのように異なる値を戻すかを示します。

```
SELECT NLS_INITCAP
   ('ijsland') "InitCap" FROM DUAL;

InitCap
-------
Ijsland

SELECT NLS_INITCAP
   ('ijsland', 'NLS_SORT = XDutch') "InitCap"
   FROM DUAL;

InitCap
-------
IJsland
```

> **参照：** ソート基準の詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

NLS_LOWER

構文



用途

NLS_LOWER は、すべての文字を小文字にして *char* を戻します。

char および *'nlsparam'* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻される文字列は、*char* が文字データ型の場合は VARCHAR2 データ型になり、*char* が LOB データ型の場合は LOB になります。*char* と同じキャラクタ・セットの文字列が戻されます。

'nlsparam' の書式および用途は、NLS_INITCAP ファンクションと同じです。

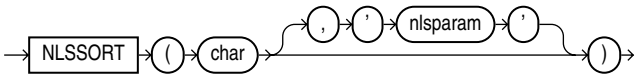
例

次の例では、XGerman 言語ソート順序を使用する文字列 'citta' を戻します。

```
SELECT NLS_LOWER
   ('CITTA''', 'NLS_SORT = XGerman') "Lowercase"
   FROM DUAL;

Lowerc
------
citta'
```

NLSSORT

構文



用途

NLSSORT は、*char* のソートに使用される文字列のバイトを戻します。

char および *'nlsparam'* は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型です。戻される文字列は RAW データ型です。

'nlsparam' の値は次の書式で指定します。

```
'NLS_SORT = sort'
```

sort は、言語ソート基準または BINARY のいずれかです。*'nlsparam'* を省略すると、このファンクションはセッションに対してデフォルトのソート基準を使用します。BINARY を指定すると、このファンクションは *char* を戻します。

'nlsparam' を指定した場合、言語ソート名に接尾辞 _ai を追加してアクセント記号の有無を区別しないソートを行うか、または _ci を追加して大 / 小文字を区別しないソートを行うことができます。アクセント記号の有無および大 / 小文字を区別しないソートの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。

参照： 詳細は、2-36 ページの「データ型の比較規則」を参照してください。

例

このファンクションを使用すると、文字列の 2 進値を基にしたソートおよび比較ではなく、言語ソート基準を基にしたソートおよび比較を指定できます。次の例では、2 つの値を含むテスト表を作成し、戻される値が NLSSORT ファンクションによってどのように順序付けられるかを示します。

```
CREATE TABLE test (name VARCHAR2(15));
INSERT INTO test VALUES ('Gaardiner');
INSERT INTO test VALUES ('Gaberd');
INSERT INTO test VALUES ('Gaasten');

SELECT * FROM test ORDER BY name;

NAME
---------------
Gaardiner
Gaasten
Gaberd

SELECT * FROM test ORDER BY NLSSORT(name, 'NLS_SORT = XDanish');

NAME
---------------
Gaberd
Gaardiner
Gaasten
```

次の例では、比較操作での NLSSORT ファンクションの使用方法を示します。

```
SELECT * FROM test WHERE name > 'Gaberd'
  ORDER BY name;

no rows selected

SELECT * FROM test WHERE NLSSORT(name, 'NLS_SORT = XDanish') >
   NLSSORT('Gaberd', 'NLS_SORT = XDanish')
   ORDER BY name;

NAME
---------------
Gaardiner
Gaasten
```

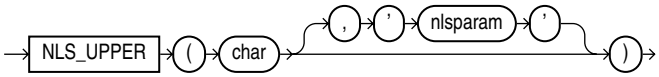
同一の言語ソート基準を使用する比較操作で頻繁に NLSSORT を使用する場合、NLS_COMP パラメータ（データベース用か現行のセッション用のいずれか）に LINGUISTIC を設定し、セッションの NLS_SORT パラメータに必要なソート基準を設定するとより効率的です。これによって、現行のセッション中のすべてのソート操作および比較操作において、デフォルトでそのソート基準が使用されるようになります。

```
ALTER SESSION SET NLS_COMP = 'LINGUISTIC';
ALTER SESSION SET NLS_SORT = 'XDanish';

SELECT * FROM test WHERE name > 'Gaberd'
  ORDER BY name;

NAME
---------------
Gaardiner
Gaasten
```

参照： ソート基準の詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

NLS_UPPER

構文



用途

NLS_UPPER は、すべての文字を大文字にして *char* を戻します。

char および *'nlsparam'* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻される文字列は、*char* が文字データ型の場合は VARCHAR2 データ型になり、*char* が LOB データ型の場合は LOB になります。*char* と同じキャラクタ・セットの文字列が戻されます。

'nlsparam' の書式および用途は、NLS_INITCAP ファンクションと同じです。

例

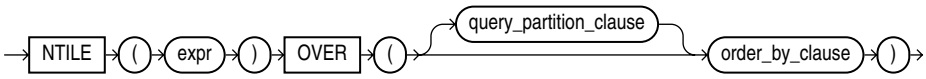
次の例では、すべての文字を大文字に変換して文字列を戻します。

```
SELECT NLS_UPPER ('große') "Uppercase"
     FROM DUAL;

Upper
-----
GROßE

SELECT NLS_UPPER ('große', 'NLS_SORT = XGerman') "Uppercase"
   FROM DUAL;

Upperc
------
GROSSE
```

参照： 「NLS_INITCAP」（5-108 ページ）

NTILE

構文



参照： 構文、セマンティクス、制限事項、および *expr* の書式の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

NTILE は分析ファンクションです。これは、順序付けられたデータセットを *expr* に指定した数のバケットに分割し、適切なバケット番号を各行に割り当てます。バケットには 1 ～ *expr* の番号が付けられます。*expr* 値は、パーティションごとに、正の定数に変換される必要があります。Oracle Database では *expr* は整数とみなされるため、この値が整数ではない定数の場合は整数に切り捨てられます。戻り値は、NUMBER です。

バケット内の行数は、最大で 1 異なります。残りの値（バケットで割った行数の余り）は、バケット 1 から順に、1 行ずつ分割されます。

expr が行数より大きい場合、行数と等しい数のバケットに行が入れられ、余りのバケットは空になります。

expr には、NTILE または他の分析ファンクションを使用して分析ファンクションをネストできません。ただし、他の組込みファンクション式を *expr* で使用できます。

> **参照：** *expr* の書式の詳細は、6-2 ページの「SQL 式」を参照してください。暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

例

次の例では、部門 100 の oe.employees 表の salary 列の値を 4 つのバケットに分割します。この部門の salary 列には 6 つの値が存在するため、2 つの余分な値（6 を 4 で割った余り）は、バケット 1 および 2 に割り当てられます。そのため、バケット 1 および 2 は、バケット 3 または 4 より値が 1 つ多くなります。

```
SELECT last_name, salary, NTILE(4) OVER (ORDER BY salary DESC)
   AS quartile FROM employees
   WHERE department_id = 100
   ORDER BY last_name, salary, quartile;

LAST_NAME                      SALARY   QUARTILE
------------------------- ---------- ----------
Chen                             8200          2
Faviet                           9000          1
Greenberg                       12000          1
Popp                             6900          4
Sciarra                          7700          3
Urman                            7800          2
```

NULLIF

構文

用途

NULLIF は、*expr1* と *expr2* を比較します。同じである場合は、NULL を戻します。異なる場合は、*expr1* を戻します。*expr1* には、リテラル NULL を指定できません。

両方の引数が数値データ型である場合、Oracle Database は、数値の優先順位が高い方の引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。2 つの引数が数値ではない場合、それらのデータ型が同じである必要があります。データ型が異なる場合、Oracle はエラーを戻します。

NULLIF ファンクションは、次の CASE 式と論理的に同じです。

```
CASE WHEN expr1 = expr 2 THEN NULL ELSE expr1 END
```

> **参照：** 「CASE 式」（6-5 ページ）

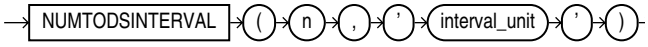
例

次の例では、サンプル・スキーマ hr から、雇用後に職種が変更した従業員を検索します。これは、employees 表の現行の job_id が job_history 表の job_id と異なるかどうかによって識別されます。

```
SELECT e.last_name, NULLIF(e.job_id, j.job_id) "Old Job ID"
   FROM employees e, job_history j
   WHERE e.employee_id = j.employee_id
   ORDER BY last_name, "Old Job ID";

LAST_NAME                 Old Job ID
------------------------- ----------
De Haan                   AD_VP
Hartstein                 MK_MAN
Kaufling                  ST_MAN
Kochhar                   AD_VP
Kochhar                   AD_VP
Raphaely                  PU_MAN
Taylor                    SA_REP
Taylor
Whalen                    AD_ASST
Whalen
```

NUMTODSINTERVAL

構文



用途

NUMTODSINTERVAL は、*n* を INTERVAL DAY TO SECOND リテラルに変換します。引数 *n* には、任意の NUMBER 値か、または NUMBER 値に暗黙的に変換可能な式を指定できます。引数 *interval_unit* のデータ型は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 です。*interval_unit* の値には *n* の単位を指定します。この値は次の文字列値のいずれかである必要があります。

- 'DAY'
- 'HOUR'
- 'MINUTE'
- 'SECOND'

interval_unit では、大 / 小文字は区別されません。カッコ内の先行値および後続値は無視されます。デフォルトでは、戻り値の精度は 9 です。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

例

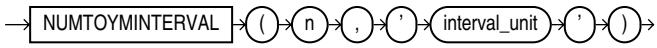
次の例では、COUNT 分析ファンクションに NUMTODSINTERVAL を使用し、各従業員について、雇用開始日から過去 100 日以内に同じマネージャの部下として配属された従業員の人数を計算します。分析ファンクションの構文の詳細は、5-10 ページの「分析ファンクション」を参照してください。

```
SELECT manager_id, last_name, hire_date,
   COUNT(*) OVER (PARTITION BY manager_id ORDER BY hire_date
   RANGE NUMTODSINTERVAL(100, 'day') PRECEDING) AS t_count
   FROM employees;
```

```
MANAGER_ID LAST_NAME                 HIRE_DATE    T_COUNT
---------- ------------------------- --------- ----------
       100 Kochhar                   21-SEP-89          1
       100 De Haan                   13-JAN-93          1
       100 Raphaely                  07-DEC-94          1
       100 Kaufling                  01-MAY-95          1
       100 Hartstein                 17-FEB-96          1
. . .
       149 Grant                     24-MAY-99          1
       149 Johnson                   04-JAN-00          1
       201 Goyal                     17-AUG-97          1
       205 Gietz                     07-JUN-94          1
           King                      17-JUN-87          1
```

NUMTOYMINTERVAL

構文



用途

NUMTOYMINTERVAL は、数値 *n* を INTERVAL YEAR TO MONTH リテラルに変換します。引数 *n* には、任意の NUMBER 値か、または NUMBER 値に暗黙的に変換可能な式を指定できます。引数 *interval_unit* のデータ型は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 です。*interval_unit* の値には *n* の単位を指定します。この値は次の文字列値のいずれかである必要があります。

- 'YEAR'
- 'MONTH'

interval_unit では、大 / 小文字は区別されません。カッコ内の先行値および後続値は無視されます。デフォルトでは、戻り値の精度は 9 です。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

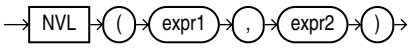
例

次の例では、SUM 分析ファンクションに NUMTOYMINTERVAL を使用し、各従業員について、その従業員の雇用日から過去 1 年の間に雇用された従業員の給与の合計を計算します。分析ファンクションの構文の詳細は、5-10 ページの「分析ファンクション」を参照してください。

```
SELECT last_name, hire_date, salary, SUM(salary)
   OVER (ORDER BY hire_date
   RANGE NUMTOYMINTERVAL(1,'year') PRECEDING) AS t_sal
   FROM employees
   ORDER BY last_name, hire_date;

LAST_NAME                 HIRE_DATE     SALARY      T_SAL
------------------------- --------- ---------- ----------
King                      17-JUN-87      24000      24000
Whalen                    17-SEP-87       4400      28400
Kochhar                   21-SEP-89      17000      17000
. . .
Markle                    08-MAR-00       2200     112400
Ande                      24-MAR-00       6400     106500
Banda                     21-APR-00       6200     109400
Kumar                     21-APR-00       6100     109400
```

NVL

構文



用途

NVL を使用すると、NULL（空白として戻される）を文字列に置換して問合せの結果に含めることができます。*expr1* が NULL の場合、NVL は *expr2* を戻します。*expr1* が NULL でない場合、NVL は *expr1* を戻します。

引数 *expr1* および *expr2* は、任意のデータ型を持つことができます。2 つの引数のデータ型が異なる場合、一方のデータ型が他方のデータ型に暗黙的に変換されます。暗黙的に変換できない場合、データベースはエラーを戻します。暗黙的な変換は、次のように実行されます。

- *expr1* が文字データの場合、Oracle Database は 2 つの引数を比較する前に *expr2* を *expr1* のデータ型に変換し、*expr1* のキャラクタ・セットで VARCHAR2 を戻します。
- *expr1* が数値である場合、Oracle は数値の優先順位が最も高い引数を判断し、その引数のデータ型に他方の引数を暗黙的に変換して、そのデータ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。数値の優先順位の詳細は、2-15 ページの「数値の優先順位」を参照してください。

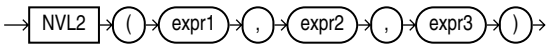
例

次の例では、従業員が歩合を受け取らない場合、従業員名および従業員の歩合「Not Applicable」を表示します。

```
SELECT last_name, NVL(TO_CHAR(commission_pct), 'Not Applicable')
   "COMMISSION" FROM employees
   WHERE last_name LIKE 'B%'
   ORDER BY last_name;

LAST_NAME                 COMMISSION
------------------------- ----------------------------------------
Baer                      Not Applicable
Baida                     Not Applicable
Banda                     .1
Bates                     .15
Bell                      Not Applicable
Bernstein                 .25
Bissot                    Not Applicable
Bloom                     .2
Bull                      Not Applicable
```

NVL2

構文



用途

NVL2 を使用すると、指定された式が NULL かどうかに基づく問合せによって戻される値を判断できます。*expr1* が NULL でない場合、NVL2 は *expr2* を戻します。*expr1* が NULL の場合、NVL2 は *expr3* を戻します。

引数 *expr1* は、任意のデータ型を持つことができます。引数 *expr2* および *expr3* は、LONG 以外の任意のデータ型を持つことができます。

expr2 と *expr3* のデータ型が異なる場合、次のようになります。

- *expr2* が文字データの場合、Oracle Database は、*expr3* が NULL の定数ではないかぎり、2 つの引数を比較する前に *expr3* を *expr2* のデータ型に変換します。*expr3* が NULL 定数の場合は、データ型の変換は行われません。Oracle は、*expr2* のキャラクタ・セットで VARCHAR2 を戻します。
- *expr2* が数値である場合、Oracle は、数値の優先順位が最も高い引数を判断し、その引数のデータ型に他方の引数を暗黙的に変換して、そのデータ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。数値の優先順位の詳細は、2-15 ページの「数値の優先順位」を参照してください。

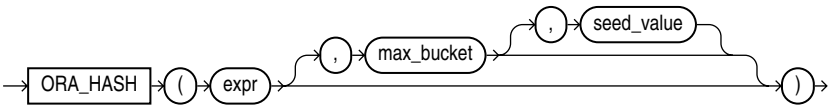
例

次の例では、employees の commission_pct 列が NULL かどうかによって、従業員の収入が給与と歩合か、または給与のみかを示します。

```
SELECT last_name, salary, NVL2(commission_pct,
   salary + (salary * commission_pct), salary) income
   FROM employees WHERE last_name like 'B%'
   ORDER BY last_name;

LAST_NAME                 SALARY     INCOME
------------------------- ---------- ----------
Baer                       10000      10000
Baida                       2900       2900
Banda                       6200       6882
Bates                       7300       8468
Bell                        4000       4000
Bernstein                   9500      11970
Bissot                      3300       3300
Bloom                      10000      12100
Bull                        4100       4100
```

ORA_HASH

構文



用途

ORA_HASH ファンクションは、指定された式のハッシュ値を計算します。このファンクションは、データのサブセットの分析や、ランダムな標本の生成などの操作に有効です。

- *expr* 引数には、Oracle Database でハッシュ値を計算するデータを指定します。*expr* に指定できるデータの長さの制限はありません。このデータは通常、列名です。*expr* は、LONG 型または LOB 型にはできません。また、ネストした表型でない場合は、ユーザー定義オブジェクト型にできません。ネストした表型のハッシュ値は、コレクション内の要素の順序に依存しません。その他のデータ型はすべて、*expr* でサポートされています。
- オプションの *max_bucket* 引数には、ハッシュ・ファンクションから戻される最大バケット値を指定します。0（ゼロ）～ 4294967295 の任意の値を指定できます。デフォルトは 4294967295 です。

- オプションの *seed_value* 引数を指定すると、同じデータ・セットに対して様々な結果を生成できます。Oracle は、ハッシュ・ファンクションを *expr* と *seed_value* の組合せに適用します。0（ゼロ）～ 4294967295 の任意の値を指定できます。デフォルトは 0（ゼロ）です。

戻り値は NUMBER です。

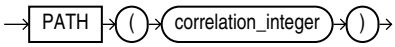
例

次の例では、sh.sales 表内の顧客 ID と製品 ID の各組合せに対してハッシュ値を作成し、そのハッシュ値を最大 100 個のバケットに分割して、最初のバケット（バケット 0（ゼロ））で amount_sold 値の合計を戻します。3 つ目の引数（5）には、ハッシュ・ファンクションのシード値を指定しています。このシード値を変更すると、同じ問合せで異なるハッシュ結果を得ることができます。

```
SELECT SUM(amount_sold) FROM sales
   WHERE ORA_HASH(CONCAT(cust_id, prod_id), 99, 5) = 0;

SUM(AMOUNT_SOLD)
----------------
       989431.14
```

PATH

構文



用途

PATH は、UNDER_PATH および EQUALS_PATH 条件でのみ使用される補助ファンクションです。一次条件で指定されたリソースへの相対パスを戻します。

correlation_integer は NUMBER 型の任意の整数で、この補助ファンクションをその一次条件と関連付けるために使用します。1 未満の値は 1 として扱われます。

> **参照：** 7-19 ページの「EQUALS_PATH 条件」および 7-19 ページの「UNDER_PATH 条件」を参照してください。

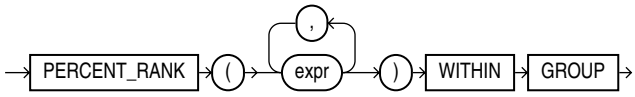
例

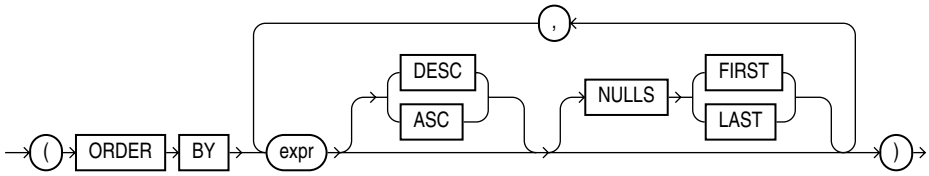
EQUALS_PATH 条件および UNDER_PATH 条件でこの補助ファンクションを使用する例は、5-58 ページの関連ファンクション「DEPTH」を参照してください。

PERCENT_RANK

集計の構文

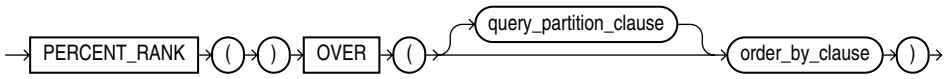
***percent_rank_aggregate*::=**





分析の構文

***percent_rank_analytic*::=**



> **参照：** 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

PERCENT_RANK は、CUME_DIST（累積分布）ファンクションと似ています。PERCENT_RANK が戻す値の範囲は、0 ～ 1（0 および 1 を含む）です。すべての集合の最初の行の PERCENT_RANK は 0（ゼロ）になります。戻り値は NUMBER です。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

- 集計ファンクションとしての PERCENT_RANK は、ファンクションの引数および対応するソート指定によって識別される不確定な行 *r* を計算し、行 *r* のランクから 1 を引いて、集計グループ内の行の数で割ります。不確定な行 *r* を Oracle Database が集計する行のグループに挿入するように計算します。

  このファンクションの引数は、各集計グループ内の 1 つの不確定行を識別します。このため、すべての引数は、集計グループ内で定数式と評価される必要があります。定数引数式および集計の ORDER BY 句の式の位置は、一致します。このため、引数の数は同じであり、その型は互換性がある必要があります。

- 分析ファンクションとしての PERCENT_RANK は、行 *r* に対して、*r* のランクから 1 引いた数を評価される行数（問合せ結果セット全体またはパーティション）より 1 少ない数で割ります。

集計の例

次の例では、サンプル表 hr.employees から、給与が 15,500 ドルであり、歩合が 5% である不確定な従業員のパーセント・ランクを計算します。

```
SELECT PERCENT_RANK(15000, .05) WITHIN GROUP
   (ORDER BY salary, commission_pct) "Percent-Rank"
   FROM employees;
```

```
Percent-Rank
------------
  .971962617
```

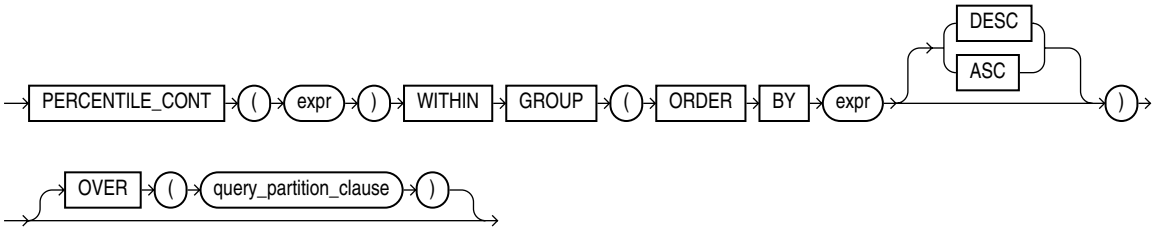
分析の例

次の例では、従業員ごとに、部門内での給与のパーセント・ランクを計算します。

```
SELECT department_id, last_name, salary,
   PERCENT_RANK()
      OVER (PARTITION BY department_id ORDER BY salary DESC) AS pr
   FROM employees
   ORDER BY pr, salary;

DEPARTMENT_ID LAST_NAME                 SALARY         PR
------------- ------------------------- ---------- ----------
           10 Whalen                          4400          0
           40 Marvis                          6500          0
. . .
           80 Vishney                        10500 .176470588
           50 Everett                         3900 .181818182
           30 Khoo                            3100         .2
. . .
           80 Johnson                         6200 .941176471
           50 Markle                          2200 .954545455
           50 Philtanker                      2200 .954545455
           50 Olson                           2100          1
. . .
```

PERCENTILE_CONT

構文



参照： OVER 句の構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

PERCENTILE_CONT は、連続分散モデルを想定する逆分散関数です。このファンクションは、パーセンタイル値およびソート指定を用い、そのソート指定に従ってそのパーセンタイル値に該当する補間された値を戻します。計算では、NULL は無視されます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を戻します。

参照： 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

最初の *expr* は、パーセンタイル値であるため、0 ～ 1 の数値で評価します。この *expr* は、各集計グループ内の定数である必要があります。ORDER BY 句には、Oracle が補間を実行できる型である数値または日時値の単一式を指定します。

PERCENTILE_CONT の結果は、順序付けされた後の値間の直線補間によって計算されます。集計グループで、パーセンタイル値（P）および行数（N）を使用すると、ソート指定に従って行を順序付けた後の行数を計算できます。この行数（RN）は、計算式 RN = (1+ (P*(N-1)) に従って計算されます。集計ファンクションの最終結果は、行番号が CRN = CEILING(RN) および FRN = FLOOR(RN) の行の値間の直線補間によって計算されます。

最終結果は次のとおりです。

```
If (CRN = FRN = RN) then the result is
  (value of expression from row at RN)
Otherwise the result is
  (CRN - RN) * (value of expression for row at FRN) +
  (RN - FRN) * (value of expression for row at CRN)
```

PERCENTILE_CONT ファンクションは、分析ファンクションとしても使用できます。その場合、OVER 句には、*query_partitioning_clause* のみを指定できます。各行に対して、各パーティション内の一連の値から、指定されたパーセンタイルに該当する値を戻します。

MEDIAN ファンクションは、パーセンタイル値がデフォルトで 0.5 に指定される特別な PERCENTILE_CONT です。詳細は、5-99 ページの「MEDIAN」を参照してください。

集計の例

次の例では、各部門の給与の中央値を計算します。

```
SELECT department_id,
   PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY salary DESC)
      "Median cont",
   PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY salary DESC)
      "Median disc"
   FROM employees GROUP BY department_id
   ORDER BY department_id, "Median cont", "Median disc;

DEPARTMENT_ID Median cont Median disc
------------- ----------- -----------
           10        4400        4400
           20        9500       13000
           30        2850        2900
           40        6500        6500
           50        3100        3100
           60        4800        4800
           70       10000       10000
           80        8900        9000
           90       17000       17000
          100        8000        8200
          110       10150       12000
                     7000        7000
```

PERCENTILE_CONT および PERCENTILE_DISC は、異なる結果を戻す場合があります。PERCENTILE_CONT は、直線補間後の結果を計算します。PERCENTILE_DISC は、集計された一連の値から値のみを戻します。この例に示すように、パーセンタイル値が 0.5 の場合、PERCENTILE_CONT は、偶数の要素を持つグループの中間の 2 つの値の平均を戻します。それに対して、PERCENTILE_DISC は、中間の 2 つの値の最初の値を戻します。奇数の要素を持つ集計グループでは、どちらのファンクションも中間の要素の値を戻します。

分析の例

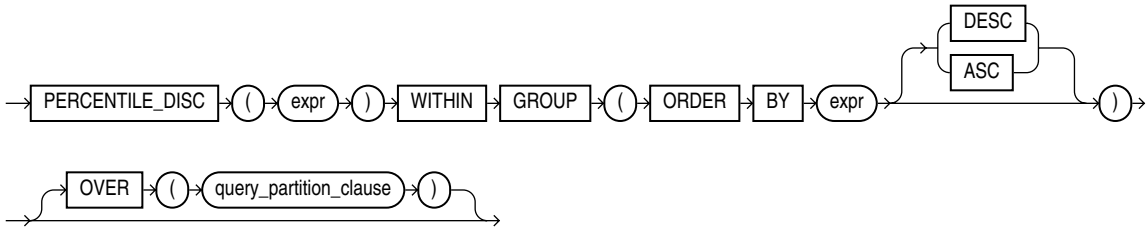
次の例では、0.5 のパーセンタイル（Percent_Rank）に対応する部門 60 の中央値は 4800 です。部門 30 の給与にパーセンタイル 0.5 がないため、2900（パーセンタイル 0.4）～ 2800（パーセンタイル 0.6）の範囲で 2850 に評価される中央値が補間される必要があります。

```
SELECT last_name, salary, department_id,
   PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY salary DESC)
      OVER (PARTITION BY department_id) "Percentile_Cont",
```

```
   PERCENT_RANK()
      OVER (PARTITION BY department_id ORDER BY salary DESC) "Percent_Rank"
FROM employees WHERE department_id IN (30, 60)
ORDER BY last_name, salary, department_id, "Percentile_Cont", "Percent_Rank";

LAST_NAME                 SALARY DEPARTMENT_ID Percentile_Cont Percent_Rank
------------------------- ---------- ------------- --------------- ------------
Austin                        4800            60            4800           .5
Baida                         2900            30            2850           .4
Colmenares                    2500            30            2850            1
Ernst                         6000            60            4800          .25
Himuro                        2600            30            2850           .8
Hunold                        9000            60            4800            0
Khoo                          3100            30            2850           .2
Lorentz                       4200            60            4800            1
Pataballa                     4800            60            4800           .5
Raphaely                     11000            30            2850            0
Tobias                        2800            30            2850           .6
```

PERCENTILE_DISC

構文



参照： OVER 句の構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

PERCENTILE_DISC は、不連続分散モデルを想定する逆分散関数です。このファンクションでは、パーセンタイル値およびソート指定を指定し、そのセットから要素を戻します。計算では、NULL は無視されます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を戻します。

参照： 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

最初の *expr* は、パーセンタイル値であるため、0 ～ 1 の数値で評価します。この expr は、各集計グループ内の定数である必要があります。ORDER BY 句には、ソート可能な型の単一式を指定します。

指定されたパーセンタイル値 P に対して、PERCENTILE_DISC は、ORDER BY 句の式の値をソートし、P 以上である（同じソート指定に従う）最小 CUME_DIST 値を持つ値を戻します。

集計の例

5-119 ページの「PERCENTILE_CONT」の例を参照してください。

分析の例

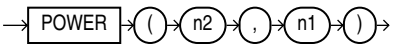
次の例では、サンプル表 hr.employees の各従業員の給与の中央値不連続パーセンタイルを計算します。

```
SELECT last_name, salary, department_id,
   PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY salary DESC)
      OVER (PARTITION BY department_id) "Percentile_Disc",
   CUME_DIST() OVER (PARTITION BY department_id
      ORDER BY salary DESC) "Cume_Dist"
FROM employees where department_id in (30, 60)
ORDER BY last_name, salary, department_id, "Percentile_Disc", "Cume_Dist";

LAST_NAME                     SALARY DEPARTMENT_ID Percentile_Disc  Cume_Dist
------------------------- ---------- ------------- --------------- ----------
Austin                          4800            60            4800         .8
Baida                           2900            30            2900         .5
Colmenares                      2500            30            2900          1
Ernst                           6000            60            4800         .4
Himuro                          2600            30            2900 .833333333
Hunold                          9000            60            4800         .2
Khoo                            3100            30            2900 .333333333
Lorentz                         4200            60            4800          1
Pataballa                       4800            60            4800         .8
Raphaely                       11000            30            2900 .166666667
Tobias                          2800            30            2900 .666666667
```

部門 30 の中央値の値は 2900 です。この値の対応するパーセンタイル（Cume_Dist）は、0.5 以上の最小値です。部門 60 の中央値の値は 4800 です。この値の対応するパーセンタイルは、0.5 以上の最小値です。

POWER

構文



用途

POWER は、*n2* を *n1* 乗した値を戻します。底 *n2* および指数 *n1* は任意の数です。ただし、*n2* が負の場合、*n1* は整数である必要があります。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。いずれかの引数が BINARY_FLOAT または BINARY_DOUBLE の場合、このファンクションは BINARY_DOUBLE を戻します。それ以外の場合、NUMBER を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

例

次の例では、3 の 2 乗を戻します。

```
SELECT POWER(3,2) "Raised" FROM DUAL;

    Raised
----------
         9
```

POWERMULTISET

構文

用途

POWERMULTISET は、入力としてネストした表を取り、入力されたネストした表のすべての空でないサブセットを含む、ネストした表のネストした表（サブ多重集合という）を戻します。

- *expr* には、ネストした表に評価される任意の式を指定できます。
- *expr* が NULL である場合、Oracle Database は NULL を戻します。
- *expr* がネストした表である場合、Oracle はエラーを戻します。
- ネストした表の要素の型は、比較可能な型である必要があります。スカラー型以外の型の比較の可能性は、7-4 ページの「比較条件」を参照してください。

注意： このファンクションは、PL/SQL ではサポートされていません。

例

まず、cust_address_tab_type データ型のネストした表であるデータ型を作成します。

```
CREATE TYPE cust_address_tab_tab_typ
  AS TABLE OF cust_address_tab_typ;
```

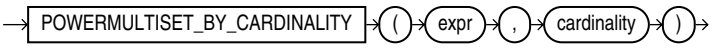
次に、POWERMULTISET ファンクションを使用して、customers_demo 表から、ネストした表の列 cust_address_ntab を選択します。

```
SELECT CAST(POWERMULTISET(cust_address_ntab)
         AS cust_address_tab_tab_typ)
  FROM customers_demo;

CAST(POWERMULTISET(CUST_ADDRESS_NTAB) AS CUST_ADDRESS_TAB_TAB_TYP)
  (STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE, COUNTRY_ID)
--------------------------------------------------------------------
CUST_ADDRESS_TAB_TAB_TYP(CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP
  ('514 W Superior St', '46901', 'Kokomo', 'IN', 'US')))
CUST_ADDRESS_TAB_TAB_TYP(CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP
  ('2515 Bloyd Ave', '46218', 'Indianapolis', 'IN', 'US')))
CUST_ADDRESS_TAB_TAB_TYP(CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP
  ('8768 N State Rd 37', '47404', 'Bloomington', 'IN', 'US')))
CUST_ADDRESS_TAB_TAB_TYP(CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP
  ('6445 Bay Harbor Ln', '46254', 'Indianapolis', 'IN', 'US')))
. . .
```

前述の例では、customers_demo 表、およびデータを含むネストした表の列が必要です。この表およびネストした表の列を作成する方法については、4-6 ページの「MULTISET 演算子」を参照してください。

POWERMULTISET_BY_CARDINALITY

構文



用途

POWERMULTISET_BY_CARDINALITY は、入力としてネストした表およびカーディナリティを取り、指定されたカーディナリティを持つネストした表のすべての空でないサブセットを含む、ネストした表のネストした表（サブ多重集合という）を戻します。

- *expr* には、ネストした表に評価される任意の式を指定できます。
- *cardinality* には、任意の整数を指定できます。
- *expr* が NULL である場合、Oracle Database は NULL を戻します。
- *expr* がネストした表である場合、Oracle はエラーを戻します。
- ネストした表の要素の型は、比較可能な型である必要があります。スカラー型以外の型の比較の可能性は、7-4 ページの「比較条件」を参照してください。

> **注意：** このファンクションは、PL/SQL ではサポートされていません。

例

まず、ネストした表のすべての行内の要素を複製し、ネストした表の行のカーディナリティを 2 に増やします。

```
UPDATE customers_demo
  SET cust_address_ntab = cust_address_ntab MULTISET UNION cust_address_ntab;
```

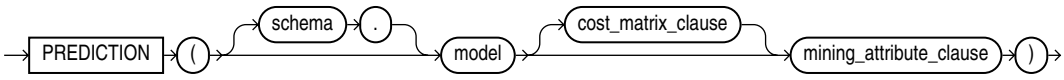
次に、POWERMULTISET_BY_CARDINALITY ファンクションを使用して、customers_demo 表から、ネストした表の列 cust_address_ntab を選択します。

```
SELECT CAST(POWERMULTISET_BY_CARDINALITY(cust_address_ntab, 2)
         AS cust_address_tab_tab_typ)
  FROM customers_demo;

CAST(POWERMULTISET_BY_CARDINALITY(CUST_ADDRESS_NTAB,2) AS CUST_ADDRESS_TAB_TAB_TYP)
  (STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE, COUNTRY_ID)
--------------------------------------------------------------------------------
CUST_ADDRESS_TAB_TAB_TYP(CUST_ADDRESS_TAB_TYP
  (CUST_ADDRESS_TYP('514 W Superior St', '46901', 'Kokomo', 'IN', 'US'),
   CUST_ADDRESS_TYP('514 W Superior St', '46901', 'Kokomo', 'IN', 'US')))
CUST_ADDRESS_TAB_TAB_TYP(CUST_ADDRESS_TAB_TYP
  (CUST_ADDRESS_TYP('2515 Bloyd Ave', '46218', 'Indianapolis', 'IN', 'US'),
   CUST_ADDRESS_TYP('2515 Bloyd Ave', '46218', 'Indianapolis', 'IN', 'US')))
CUST_ADDRESS_TAB_TAB_TYP(CUST_ADDRESS_TAB_TYP
  (CUST_ADDRESS_TYP('8768 N State Rd 37', '47404', 'Bloomington', 'IN', 'US'),
   CUST_ADDRESS_TYP('8768 N State Rd 37', '47404', 'Bloomington', 'IN', 'US')))
. . .
```

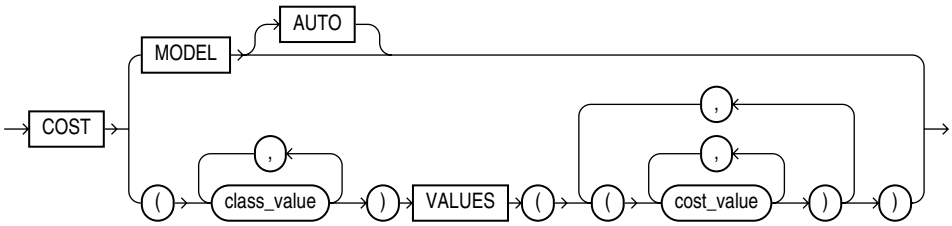
前述の例では、customers_demo 表、およびデータを含むネストした表の列が必要です。この表およびネストした表の列を作成する方法については、4-6 ページの「MULTISET 演算子」を参照してください。

PREDICTION

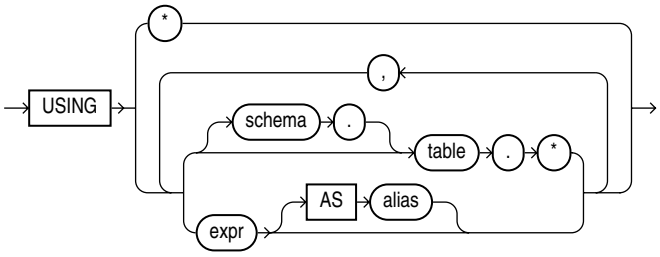
構文



***cost_matrix_clause*::=**



***mining_attribute_clause*::=**



用途

このファンクションは、DBMS_DATA_MINING パッケージまたは Oracle Data Mining Java API で作成したモデルで使用するためのものです。このファンクションは、モデルの最適な予測を戻します。戻されるデータ型は、モデルの作成中に使用するターゲット値の型によって異なります。回帰モデルの場合、このファンクションは期待値を戻します。

cost_matrix_clause COST 句は、すべての分類モデルに対して有効です。

- モデルに関連付けられたコスト・マトリックスを考慮して、スコアリングを実行する必要があることを示すには、COST MODEL を指定します。このようなスコアリングのコスト・マトリックスが存在しない場合は、エラーが戻されます。
- コスト・マトリックスが存在するかどうかわからない場合は、COST MODEL AUTO を指定します。このとき、格納されたコスト・マトリックスが存在する場合は、このマトリックスを使用して最低のコスト予測が戻されます。格納されたコスト・マトリックスが存在しない場合は、最も高い確率の予測が戻されます。
- 表内コスト・マトリックスを指定するには、VALUES 句（*cost_matrix_clause* の下の方のブランチ）を使用します。表内コスト・マトリックスは、モデルがスコアリングのコスト・マトリックスに関連付けられているかどうかにかかわらず使用できます。

cost_matrix_clause 句を指定しない場合、最適な予測は最も高い確率を持つターゲット・クラスになります。2 つ以上のクラスが最も高い確率にある場合、そのうちの 1 つのクラスが選択されます。

mining_attribute_clause これは、モデルの作成時に提供された予測子をマップします。USING * を指定すると、基礎となる入力（表、ビューなど）から取り出すことができる列と式にすべての予測子がマップされます。

- モデルで使用される予測子より多くの予測子を *mining_attribute_clause* で指定すると、余分な式が自動的に無視されます。
- 作成中に使用される予測子より少ない予測子を指定すると、指定した予測子のサブセットで操作が処理され、ベストエフォート・ベースで情報が戻されます。この句で指定した予測子の数に関係なく、すべての型のモデルで結果が戻されます。
- 作成中に使用される予測子と同じ名前で異なるデータ型の予測子を指定すると、データベースは暗黙的に変換して、元のビルドと同じ型の予測子の値を生成します。

参照：

- Oracle Data Mining の詳細は、『Oracle Data Mining 概要』を参照してください。
- コードで使用可能なデモ・プログラムの詳細は、『Oracle Data Mining 管理者ガイド』を参照してください。
- SQL データ・マイニング・ファンクションを使用したリアルタイム・スコアリングの詳細は、『Oracle Data Mining アプリケーション開発者ガイド』を参照してください。
- DBMS_DATA_MINING パッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

例

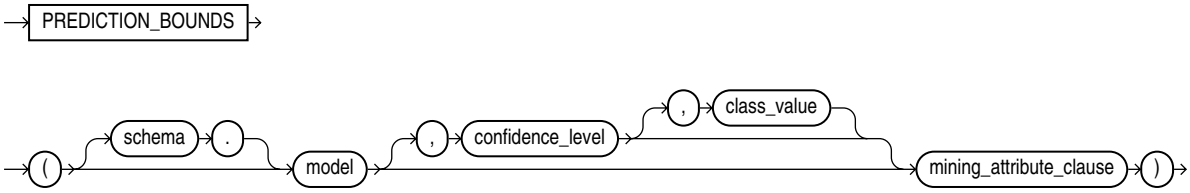
次の例では、提携カードを使用している可能性のある顧客の平均年齢を性別ごとに戻します。PREDICTION ファンクションでは、cust_marital_status、education および household_size の予測子のみを考慮します。

この例と前提条件のデータ・マイニング操作（ビューの作成など）は、デモ・ファイル $ORACLE_HOME/rdbms/demo/dmdtdemo.sql で確認できます。データ・マイニングのデモ・ファイルの一般情報は、『Oracle Data Mining 管理者ガイド』を参照してください。次に、このファンクションの構文の使用例を示します。

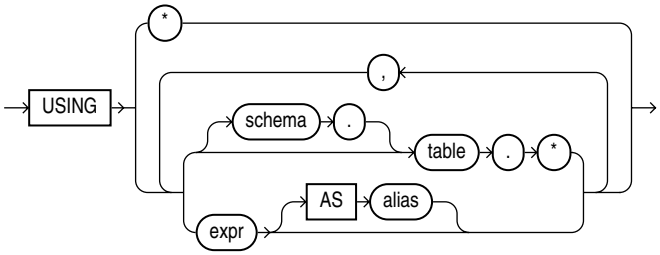
```
SELECT cust_gender, COUNT(*) AS cnt, ROUND(AVG(age)) AS avg_age
   FROM mining_data_apply_v
   WHERE PREDICTION(DT_SH_Clas_sample COST MODEL
      USING cust_marital_status, education, household_size) = 1
   GROUP BY cust_gender
   ORDER BY cust_gender;

C        CNT    AVG_AGE
- ---------- ----------
F        170         38
M        685         42
```

PREDICTION_BOUNDS

構文



***mining_attribute_clause*::=**



用途

PREDICTION_BOUNDS ファンクションは、汎用化された線形モデルにのみ使用します。このファンクションは、LOWER と UPPER の 2 つの NUMBER フィールドを持つオブジェクトを戻します。回帰マイニング・ファンクションの場合は、予測の値に限度が適用されます。分類マイニング・ファンクションの場合は、確率値に限度が適用されます。リッジ回帰を使用して GLM が構築されたか、または構築中に共分散マトリックスに異常があると認識された場合は、両方のフィールドに NULL が戻されます。

- *confidence_level* には、(0,1) の範囲の数値を指定します。この句を省略した場合、デフォルト値は 0.95 になります。
- *class_value* 引数は、分類モデルに対して有効ですが、回帰モデルには有効ではありません。デフォルトでは、最も高い確率の予測の限度が戻されます。*class_value* 引数を使用すると、ターゲット値に固有の限度値を除外できます。

  confidence_level に NULL を指定すると、デフォルトで *confidence_level* を残したまま *class_value* を指定できます。
- PREDICTION_BOUNDS に対する *mining_attribute_clause* の動作は、PREDICTION の場合と同じです。詳細は、5-126 ページの「mining_attribute_clause」を参照してください。

参照 :

- Oracle Data Mining および汎用化された線形モデルの詳細は、『Oracle Data Mining 概要』を参照してください。
- コードで使用可能なデモ・プログラムの詳細は、『Oracle Data Mining 管理者ガイド』を参照してください。
- SQL データ・マイニング・ファンクションを使用したリアルタイム・スコアリングの詳細は、『Oracle Data Mining アプリケーション開発者ガイド』を参照してください。
- DBMS_DATA_MINING パッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

例

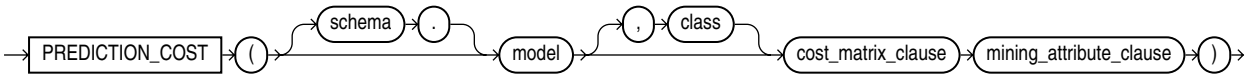
次の例では、98% の確度で年齢が 25 ～ 45 才と予測される顧客の分布を戻します。

この例と前提条件のデータ・マイニング操作は、デモ・ファイル $ORACLE_HOME/rdbms/demo/dmglcdem.sql で確認できます。次に、このファンクションの構文の使用例を示します。データ・マイニングのデモ・ファイルの一般情報は、『Oracle Data Mining 管理者ガイド』を参照してください。

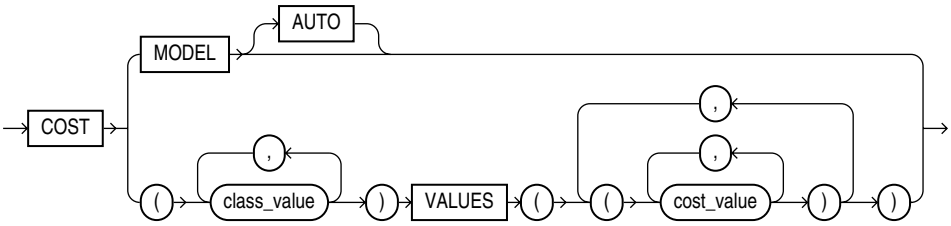
```
SELECT count(cust_id) cust_count, cust_marital_status
  FROM (SELECT cust_id, cust_marital_status
    FROM mining_data_apply_v
    WHERE PREDICTION_BOUNDS(glmr_sh_regr_sample,0.98 USING *).LOWER > 24 AND
          PREDICTION_BOUNDS(glmr_sh_regr_sample,0.98 USING *).UPPER < 46)
    GROUP BY cust_marital_status;

    CUST_COUNT CUST_MARITAL_STATUS
-------------- --------------------
            46 NeverM
             7 Mabsent
             5 Separ.
            35 Divorc.
            72 Married
```

PREDICTION_COST

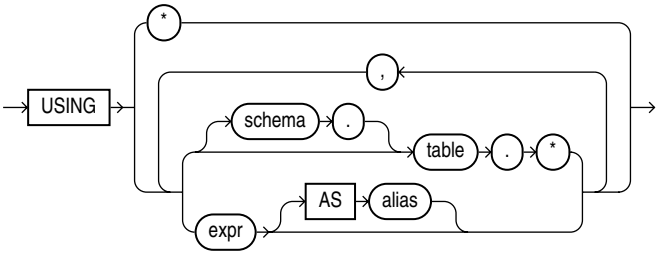
構文



***cost_matrix_clause*::=**



***mining_attribute_clause*::=**



用途

このファンクションは、DBMS_DATA_MINING パッケージまたは Oracle Data Mining Java API で作成した任意の分類モデルで使用するためのものです。また、指定した予測のコストのメジャーを Oracle の NUMBER として戻します。

オプションの *class* パラメータを指定すると、このファンクションは指定したクラスのコストを戻します。*class* パラメータを指定しない場合、最適な予測に関連付けられたコストが戻されます。この形式と PREDICTION ファンクションを組み合せて使用すると、予測値とコストの最適な組合せを取得できます。

COST 句は、すべての分類モデルに対して有効です。

- モデルに関連付けられたコスト・マトリックスを考慮して、スコアリングを実行する必要があることを示すには、COST MODEL を指定します。このようなスコアリングのコスト・マトリックスが存在しない場合は、エラーが戻されます。
- コスト・マトリックスが存在するかどうかわからない場合は、COST MODEL AUTO を指定します。このとき、格納されたコスト・マトリックスを使用してコストが戻されます。格納されたコスト・マトリックスが存在しない場合は、単位コスト・マトリックス（対角線上は 0 の値、それ以外の場所は 1 の値）が適用されます。これは、1 から指定されたクラスの確率を引いた値と同じです。
- 表内コスト・マトリックスを指定するには、VALUES 句（*cost_matrix_clause* の下の方のブランチ）を使用します。表内コスト・マトリックスは、モデルがスコアリングのコスト・マトリックスに関連付けられているかどうかにかかわらず使用できます。

mining_attribute_clause は、PREDICTION ファンクションと同様に動作します。詳細は、5-126 ページの「mining_attribute_clause」を参照してください。

参照 :

- Oracle Data Mining の概要およびコストの詳細は、『Oracle Data Mining 概要』を参照してください。
- コードで使用可能なデモ・プログラムの詳細は、『Oracle Data Mining 管理者ガイド』を参照してください。
- SQL データ・マイニング・ファンクションを使用したリアルタイム・スコアリングの詳細は、『Oracle Data Mining アプリケーション開発者ガイド』を参照してください。
- DBMS_DATA_MINING パッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

例

次の例では、イタリア在住で、提携カードの使用を薦める上で最もコストが低い顧客を 10 人検索します。

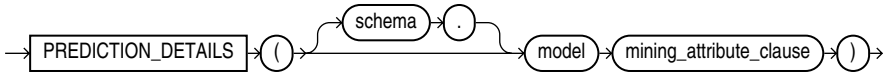
この例と前提条件のデータ・マイニング操作は、デモ・ファイル $ORACLE_HOME/rdbms/demo/dmdtdemo.sql で確認できます。データ・マイニングのデモ・ファイルの一般情報は、『Oracle Data Mining 管理者ガイド』を参照してください。次に、このファンクションの構文の使用例を示します。

```
WITH
cust_italy AS (
SELECT cust_id
  FROM mining_data_apply_v
 WHERE country_name = 'Italy'
ORDER BY PREDICTION_COST(DT_SH_Clas_sample, 1 COST MODEL USING *) ASC, 1
)
SELECT cust_id
  FROM cust_italy
 WHERE rownum < 11;
```

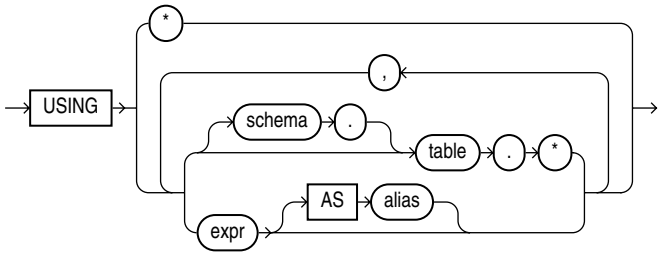
```
   CUST_ID
----------
    100081
    100179
    100185
    100324
    100344
    100554
    100662
    100733
    101250
    101306

10 rows selected.
```

PREDICTION_DETAILS

構文



***mining_attribute_clause*::=**



用途

このファンクションは、DBMS_DATA_MINING パッケージまたは Oracle Data Mining Java API で作成した決定ツリー・モデルと単一機能の Adaptive Bayes Network（ABN）モデルで使用するためのものです。このファンクションは、入力行のスコアリングに関連するモデル固有の情報を含む XML 文字列を戻します。今回のリリースでは、戻り値は次の書式になります。

```
<Node id= "integer"/>
```

ここで、*integer* はデータ・マイニングのツリー・ノードの識別子です。出力の形式は、変更される可能性があります。今後のリリースでは、追加の予測情報を提供するように拡張される可能性があります。

mining_attribute_clause は、PREDICTION ファンクションと同様に動作します。詳細は、5-126 ページの「mining_attribute_clause」を参照してください。

参照：

- Oracle Data Mining の詳細は、『Oracle Data Mining 概要』を参照してください。
- コードで使用可能なデモ・プログラムの詳細は、『Oracle Data Mining 管理者ガイド』を参照してください。
- SQL データ・マイニング・ファンクションを使用したリアルタイム・スコアリングの詳細は、『Oracle Data Mining アプリケーション開発者ガイド』を参照してください。

例

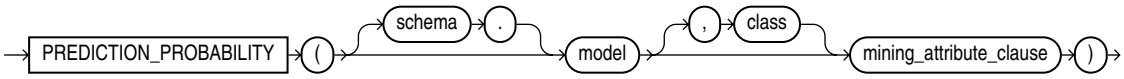
次の例では、DT_SH_Clas_sample 決定ツリー・モデルの関連する予測子である mining_data_apply_v ビューから、すべての属性を使用します。テクニカル・サポートで働く 25 才未満の顧客の場合、DT_SH_Clas_sample モデルを持つレコードのスコアリングの結果からツリー・ノードが戻されます。

この例と前提条件のデータ・マイニング操作（ビューの作成など）は、デモ・ファイル $ORACLE_HOME/rdbms/demo/dmdtdemo.sql で確認できます。データ・マイニングのデモ・ファイルの一般情報は、『Oracle Data Mining 管理者ガイド』を参照してください。次に、このファンクションの構文の使用例を示します。

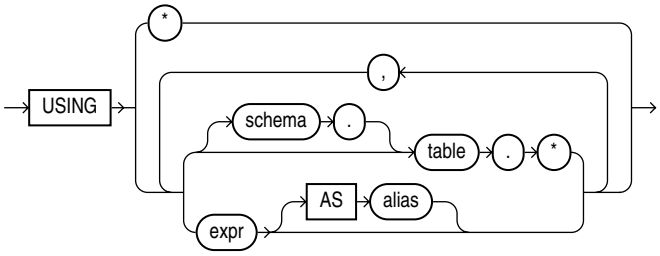
```
SELECT cust_id, education,
   PREDICTION_DETAILS(DT_SH_Clas_sample using *) treenode
   FROM mining_data_apply_v
   WHERE occupation = 'TechSup' AND age < 25
   ORDER BY cust_id;

   CUST_ID EDUCATION            TREENODE
---------- -------------------- -------------------------
    100234 < Bach.              <Node id="21"/>
    100320 < Bach.              <Node id="21"/>
    100349 < Bach.              <Node id="21"/>
    100419 < Bach.              <Node id="21"/>
    100583 < Bach.              <Node id="13"/>
    100657 HS-grad              <Node id="21"/>
    101171 < Bach.              <Node id="21"/>
    101225 < Bach.              <Node id="21"/>
    101338 < Bach.              <Node id="21"/>

9 rows selected.
```

PREDICTION_PROBABILITY

構文



***mining_attribute_clause*::=**



用途

このファンクションは、DBMS_DATA_MINING パッケージまたは Oracle Data Mining Java API で作成した分類モデルで使用するためのものです。このファンクションは、他の型のモデルに対しては無効です。また、指定した予測の確率を Oracle の NUMBER として戻します。

オプションの *class* パラメータを指定すると、このファンクションは指定したクラスの確率を戻します。これは、指定したターゲット・クラス値の選択に関連付けられた確率と同じです。

class パラメータを指定しない場合、最適な予測に関連付けられた確率が戻されます。この形式と PREDICTION ファンクションを組み合せて使用すると、予測値と確率の最適な組合せを取得できます。

mining_attribute_clause は、PREDICTION ファンクションと同様に動作します。詳細は、5-126 ページの「mining_attribute_clause」を参照してください。

参照：

- Oracle Data Mining の詳細は、『Oracle Data Mining 概要』を参照してください。
- コードで使用可能なデモ・プログラムの詳細は、『Oracle Data Mining 管理者ガイド』を参照してください。
- SQL データ・マイニング・ファンクションを使用したリアルタイム・スコアリングの詳細は、『Oracle Data Mining アプリケーション開発者ガイド』を参照してください。

例

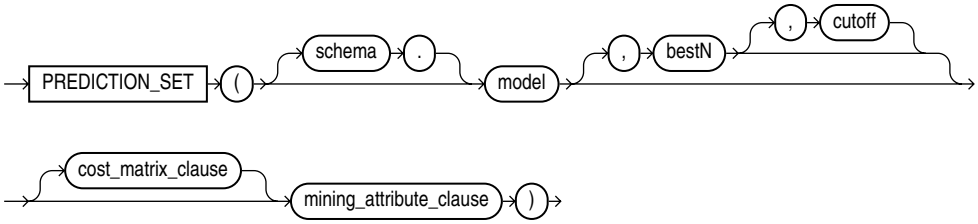
次の例では、提携カードを使用している可能性が最も高いイタリア在住の顧客を 10 人戻します。

この例と前提条件のデータ・マイニング操作（ビューの作成など）は、デモ・ファイル $ORACLE_HOME/rdbms/demo/dmdtdemo.sql で確認できます。データ・マイニングのデモ・ファイルの一般情報は、『Oracle Data Mining 管理者ガイド』を参照してください。次に、このファンクションの構文の使用例を示します。

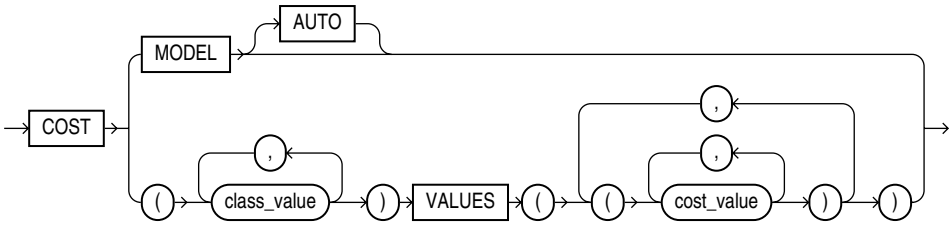
```
SELECT cust_id FROM (
   SELECT cust_id
   FROM mining_data_apply_v
   WHERE country_name = 'Italy'
   ORDER BY PREDICTION_PROBABILITY(DT_SH_Clas_sample, 1 USING *)
      DESC, cust_id)
   WHERE rownum < 11;

   CUST_ID
----------
    100081
    100179
    100185
    100324
    100344
    100554
    100662
    100733
    101250
    101306

10 rows selected.
```

PREDICTION_SET

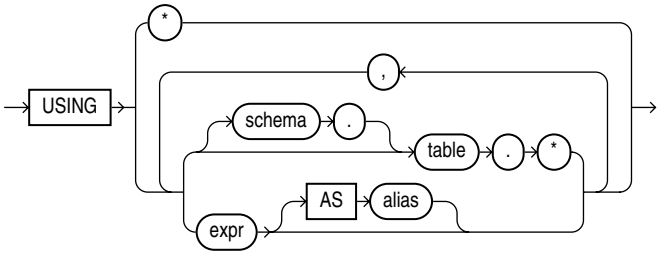
構文



***cost_matrix_clause*::=**



***mining_attribute_clause*::=**



用途

このファンクションは、DBMS_DATA_MINING パッケージまたは Oracle Data Mining Java API を使用して作成した分類モデルで使用するためのものです。このファンクションは、他の型のモデルに対しては無効です。このファンクションは、複数クラスの分類の使用例で、すべてのクラスを含むオブジェクトの VARRAY を戻します。オブジェクト・フィールドには、PREDICTION、PROBABILITY および COST の名前が付きます。PREDICTION フィールドのデータ型は、モデルの作成中に使用するターゲット値の型によって異なります。他の 2 つのフィールドは、両方とも Oracle の NUMBER になります。要素は最適な予測から最低の予測の順序で戻されます。

- *bestN* には、戻されるターゲット・クラスを最も高い確率（コスト・マトリックス句が指定されている場合は、最も低いコスト）の N に制限するために、正の整数を指定します。複数のクラスが N 番目の値にあっても、N 個の値のみが戻されます。*cutoff* のみでフィルタ処理するには、このパラメータに NULL を指定します。
- *cutoff* には、戻されるターゲット・クラスを、指定したカットオフ値以上（コスト・マトリックス句が指定されている場合は、指定したコスト以下）の確率を持つターゲット・クラスに制限するために、NUMBER 値を指定します。*bestN* に NULL を指定すると、*cutoff* のみでフィルタ処理できます。

  bestN と *cutoff* の両方に値を指定すると、戻される予測は、*bestN* で、かつ確率（または *cost_matrix_clause* 指定時のコスト）がしきい値を超えるもののみに制限されます。

cost_matrix_clause 句は、すべての分類モデルに対して有効です。この句を指定すると、*bestN* と *cutoff* の両方は、予測の確率ではなく予測コストに関して処理されます。*bestN* の値は、結果を N の最適（最低）コストを持つターゲット・クラスに制限し、*cutoff* は、ターゲット・クラスを、指定したカットオフ以下のコストを持つターゲット・クラスに制限します。

この句を指定すると、コレクションの各オブジェクトは、予測値を含むスカラー値（データ型はモデルの作成時に使用されるターゲット値の型によって異なる）、予測確率および予測コスト（両方とも Oracle の NUMBER）になります。

この句を指定しない場合、VARRAY の各オブジェクトは、予測値と予測確率を含むスカラーの組になります。戻されるデータ型は、前述の説明と同様です。

- モデルに関連付けられたコスト・マトリックスを考慮して、スコアリングを実行する必要があることを示すには、COST MODEL を指定します。このようなコスト・マトリックスが存在しない場合は、エラーが戻されます。
- コスト・マトリックスが存在するかどうかわからない場合は、COST MODEL AUTO を指定します。この場合、次のようになります。
  - 格納されたコスト・マトリックスが存在する場合、結果は COST MODEL の場合と同じになります。
  - 格納されたコスト・マトリックスが存在しない場合は、結果は *cost_matrix_clause* を使用しない場合とほとんど同じになります。ただし、コレクションのオブジェクトが 3 つあり、コスト値が単位コスト・マトリックス（対角線上は 0 の値、それ以外の場所は 1 の値）に基づいて計算されている場合は除きます。これは、1 から指定されたクラスの確率を引いた値と同じです。格納されたコスト・マトリックスが存在しない場合は、cutoff パラメータは無視されます。
- 表内コスト・マトリックスを指定するには、VALUES 句（*cost_matrix_clause* の下の方のブランチ）を使用します。表内コスト・マトリックスは、モデルがスコアリングのコスト・マトリックスに関連付けられているかどうかにかかわらず使用できます。

mining_attribute_clause は、PREDICTION ファンクションと同様に動作します。詳細は、5-126 ページの「mining_attribute_clause」を参照してください。

参照：

- Oracle Data Mining の詳細は、『Oracle Data Mining 概要』を参照してください。
- コードで使用可能なデモ・プログラムの詳細は、『Oracle Data Mining 管理者ガイド』を参照してください。
- SQL データ・マイニング・ファンクションを使用したリアルタイム・スコアリングの詳細は、『Oracle Data Mining アプリケーション開発者ガイド』を参照してください。
- DBMS_DATA_MINING パッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

例

次の例では、提携カードの使用および不使用の可能性とコストを、10 人の顧客それぞれで示します。この例はバイナリ・ターゲットを持ちますが、このような問合せは「Low」、「Med」、「High」などの複数クラスの分類でも有効です。

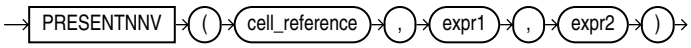
この例と前提条件のデータ・マイニング操作は、デモ・ファイル $ORACLE_HOME/rdbms/demo/dmdtdemo.sql で確認できます。データ・マイニングのデモ・ファイルの一般情報は、『Oracle Data Mining 管理者ガイド』を参照してください。次に、このファンクションの構文の使用例を示します。

```
SELECT T.cust_id, S.prediction, S.probability, S.cost
  FROM (SELECT cust_id,
               PREDICTION_SET(dt_sh_clas_sample COST MODEL USING *) pset
          FROM mining_data_apply_v
```

```
        WHERE cust_id < 100011) T,
      TABLE(T.pset) S
ORDER BY cust_id, S.prediction;

   CUST_ID PREDICTION PROBABILITY  COST
---------- ---------- ----------- -----
    100001          0      .96682   .27
    100001          1      .03318   .97
    100002          0      .74038  2.08
    100002          1      .25962   .74
    100003          0      .90909   .73
    100003          1      .09091   .91
    100004          0      .90909   .73
    100004          1      .09091   .91
    100005          0      .27236  5.82
    100005          1      .72764   .27
    100006          0     1.00000   .00
    100006          1      .00000  1.00
    100007          0      .90909   .73
    100007          1      .09091   .91
    100008          0      .90909   .73
    100008          1      .09091   .91
    100009          0      .27236  5.82
    100009          1      .72764   .27
    100010          0      .80808  1.54
    100010          1      .19192   .81

20 rows selected.
```

PRESENTNNV

構文



用途

PRESENTNNV ファンクションは、SELECT 文の *model_clause* でのみ、およびモデル・ルールの右側でのみ使用できます。このファンクションは、*cell_reference* が *model_clause* の実行前に存在し、PRESENTNNV の評価時に NULL ではない場合に *expr1* を戻します。それ以外の場合は *expr2* を戻します。このファンクションは、NVL2 とは異なります。NVL2 は、*model_clause* の実行前の状態のデータを評価するのではなく、実行時のデータを評価します。

参照：

- 構文およびセマンティクスの詳細は、19-25 ページの「model_clause」および 6-11 ページの「モデル式」を参照してください。
- 比較については、5-115 ページの「NVL2」を参照してください。

例

次の例では、2002 年のマウス・パッドの売上を含む行が存在し、その売上値が NULL ではない場合、その売上値を変更しません。その行が存在し、売上値が NULL の場合、その売上値を 10 に設定します。その行が存在しない場合、売上値を 10 に設定して行を作成します。

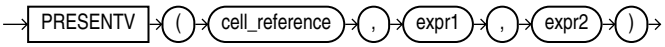
```
SELECT country, prod, year, s
  FROM sales_view_ref
  MODEL
    PARTITION BY (country)
```

```
    DIMENSION BY (prod, year)
    MEASURES (sale s)
    IGNORE NAV
    UNIQUE DIMENSION
    RULES UPSERT SEQUENTIAL ORDER
    ( s['Mouse Pad', 2002] =
        PRESENTNNV(s['Mouse Pad', 2002], s['Mouse Pad', 2002], 10)
    )
  ORDER BY country, prod, year;

COUNTRY      PROD                                    YEAR          S
----------   ----------------------------------   --------   ---------
France       Mouse Pad                               1998     2509.42
France       Mouse Pad                               1999     3678.69
France       Mouse Pad                               2000     3000.72
France       Mouse Pad                               2001     3269.09
France       Mouse Pad                               2002          10
France       Standard Mouse                          1998     2390.83
France       Standard Mouse                          1999     2280.45
France       Standard Mouse                          2000     1274.31
France       Standard Mouse                          2001     2164.54
Germany      Mouse Pad                               1998     5827.87
Germany      Mouse Pad                               1999     8346.44
Germany      Mouse Pad                               2000     7375.46
Germany      Mouse Pad                               2001     9535.08
Germany      Mouse Pad                               2002          10
Germany      Standard Mouse                          1998     7116.11
Germany      Standard Mouse                          1999     6263.14
Germany      Standard Mouse                          2000     2637.31
Germany      Standard Mouse                          2001     6456.13

18 rows selected.
```

この例では、ビュー sales_view_ref が必要です。このビューを作成する方法については、19-31 ページの「例」を参照してください。

PRESENTV

構文



用途

PRESENTV ファンクションは、SELECT 文の *model_clause* でのみ、およびモデル・ルールの右側でのみ使用できます。このファンクションは、*cell_reference* が存在する場合、*model_clause* を実行する前に *expr1* を戻します。それ以外の場合は *expr2* を戻します。

> **参照：** 構文およびセマンティクスの詳細は、19-25 ページの「model_clause」および 6-11 ページの「モデル式」を参照してください。

例

次の例では、2000 年のマウス・パッドの売上を含む行が存在する場合、2001 年のマウス・パッドの売上を 2000 年のマウス・パッドの売上値に設定します。その行が存在しない場合、2001 年のマウス・パッドの売上値を 0（ゼロ）に設定して行を作成します。

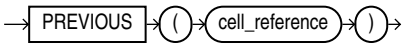
```
SELECT country, prod, year, s
  FROM sales_view_ref
  MODEL
```

```
    PARTITION BY (country)
    DIMENSION BY (prod, year)
    MEASURES (sale s)
    IGNORE NAV
    UNIQUE DIMENSION
    RULES UPSERT SEQUENTIAL ORDER
    (
      s['Mouse Pad', 2001] =
        PRESENTV(s['Mouse Pad', 2000], s['Mouse Pad', 2000], 0)
    )
  ORDER BY country, prod, year;

COUNTRY      PROD                                    YEAR          S
-----------  ----------------------------------  --------  ---------
France       Mouse Pad                               1998    2509.42
France       Mouse Pad                               1999    3678.69
France       Mouse Pad                               2000    3000.72
France       Mouse Pad                               2001    3000.72
France       Standard Mouse                          1998    2390.83
France       Standard Mouse                          1999    2280.45
France       Standard Mouse                          2000    1274.31
France       Standard Mouse                          2001    2164.54
Germany      Mouse Pad                               1998    5827.87
Germany      Mouse Pad                               1999    8346.44
Germany      Mouse Pad                               2000    7375.46
Germany      Mouse Pad                               2001    7375.46
Germany      Standard Mouse                          1998    7116.11
Germany      Standard Mouse                          1999    6263.14
Germany      Standard Mouse                          2000    2637.31
Germany      Standard Mouse                          2001    6456.13

16 rows selected.
```

この例では、ビュー sales_view_ref が必要です。このビューを作成する方法については、19-36 ページの「MODEL 句の例 :」を参照してください。

PREVIOUS

構文



用途

PREVIOUS ファンクションは、SELECT 文の *model_clause* でのみ、および *model_rules_clause* の ITERATE ... [UNTIL] 句でのみ使用できます。このファンクションは、各反復の最初に *cell_reference* の値を戻します。

> **参照 :** 構文およびセマンティクスの詳細は、19-25 ページの「model_clause」および 6-11 ページの「モデル式」を参照してください。

例

次の例では、反復の最初と最後にある cur_val の値の差が 1 未満になるまで、最大 1000 回ルールを繰り返します。

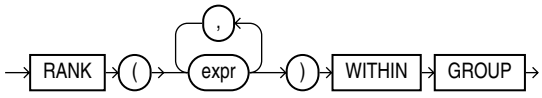
```
SELECT dim_col, cur_val, num_of_iterations
  FROM (SELECT 1 AS dim_col, 10 AS cur_val FROM dual)
  MODEL
    DIMENSION BY (dim_col)
```

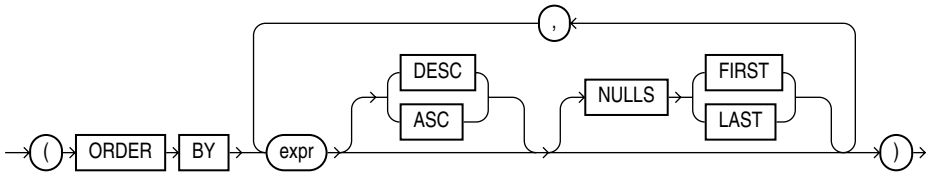
```
    MEASURES (cur_val, 0 num_of_iterations)
    IGNORE NAV
    UNIQUE DIMENSION
    RULES ITERATE (1000) UNTIL (PREVIOUS(cur_val[1]) - cur_val[1] < 1)
    (
      cur_val[1] = cur_val[1]/2,
      num_of_iterations[1] = num_of_iterations[1] + 1
    );

   DIM_COL    CUR_VAL NUM_OF_ITERATIONS
---------- ---------- -----------------
         1       .625                 4
```

RANK

集計の構文

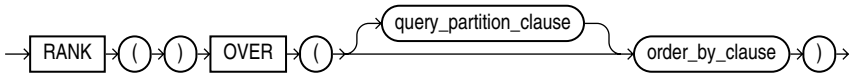
***rank_aggregate*::=**





分析の構文

***rank_analytic*::=**



> **参照：** 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

RANK は、一連の値における値のランクを計算します。戻り型は、NUMBER です。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。数値の優先順位の詳細は、2-15 ページの「数値の優先順位」を参照してください。

ランク付け基準と同じ値を持つ行は、同じランクになります。Oracle Database は連結行の数を連結ランクに追加して、次のランクを計算します。そのため、ランクは連続した数値でない場合があります。このファンクションは、上位 N 番および下位 N 番のレポートに有効です。

- 集計ファンクションとして使用する RANK は、ファンクションの引数として識別される不確定な行のランクを、指定されたソート指定に従って計算します。ファンクションの引数は、各集計グループの単一行を識別するため、すべての引数は各集計グループ内で定数式に評価される必要があります。定数引数式および集計の ORDER BY 句の式の位置は、一致します。このため、引数の数は同じであり、その型は互換性がある必要があります。

- 分析ファンクションとして使用する RANK は、ある問合せが戻す他の行について、その問合せが戻す各行のランクを計算します。この計算は、*order_by_clause* にある *value_exprs* の値に基づいて行われます。

集計の例

次の例では、サンプル表 hr.employees から、給与が 15,500 ドルであり、歩合が 5% である不確定な従業員のランクを計算します。

```
SELECT RANK(15500, .05) WITHIN GROUP
   (ORDER BY salary, commission_pct) "Rank"
   FROM employees;

      Rank
----------
       105
```

同様に、次の問合せは、従業員の給与から給与が 15,500 ドルのランクを戻します。

```
SELECT RANK(15500) WITHIN GROUP
   (ORDER BY salary DESC) "Rank of 15500"
   FROM employees;

Rank of 15500
--------------
             4
```

分析の例

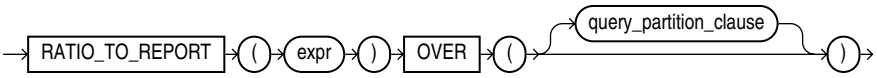
次の例では、サンプル・スキーマ hr の部門 80 の従業員を、その給与および歩合に基づいてランク付けします。給与が同じ場合は同じランクになるため、連続しないランクになります。この例と、5-56 ページの「DENSE_RANK」の例を比較してください。

```
SELECT department_id, last_name, salary, commission_pct,
   RANK() OVER (PARTITION BY department_id
   ORDER BY salary DESC, commission_pct) "Rank"
FROM employees WHERE department_id = 80
ORDER BY department_id, last_name, salary, commission_pct, "Rank";

DEPARTMENT_ID LAST_NAME                     SALARY COMMISSION_PCT       Rank
------------- ------------------------- ---------- -------------- ----------
           80 Abel                           11000             .3          5
           80 Ande                            6400             .1         31
           80 Banda                           6200             .1         32
           80 Bates                           7300            .15         26
           80 Bernstein                       9500            .25         14
           80 Bloom                          10000             .2          9
           80 Cambrault                       7500             .2         23
           80 Cambrault                      11000             .3          5
           80 Doran                           7500             .3         24
           80 Errazuriz                      12000             .3          3
           80 Fox                             9600             .2         12
           80 Greene                          9500            .15         13
           80 Hall                            9000            .25         16
           80 Hutton                          8800            .25         18
           80 Johnson                         6200             .1         32
           80 King                           10000            .35         11
           80 Kumar                           6100             .1         34
           80 Lee                             6800             .1         30
           80 Livingston                      8400             .2         20
           80 Marvins                         7200             .1         27
           80 McEwen                          9000            .35         17
           80 Olsen                           8000             .2         21
```

```
80 Ozer                    11500          .25          4
80 Partners                13500           .3          2
80 Russell                 14000           .4          1
80 Sewall                   7000          .25         29
80 Smith                    7400          .15         25
80 Smith                    8000           .3         22
80 Sully                    9500          .35         15
80 Taylor                   8600           .2         19
80 Tucker                  10000           .3         10
80 Tuvault                  7000          .15         28
80 Vishney                 10500          .25          8
80 Zlotkey                 10500           .2          7
```

RATIO_TO_REPORT

構文



参照： 構文、セマンティクス、制限事項、および *expr* の書式の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

RATIO_TO_REPORT は分析ファンクションです。このファンクションは、ある値の集合の合計に対する、その値の比率を計算します。*expr* が NULL の場合は、値も NULL になります。

値の集合は、*query_partition_clause* によって決まります。この句を省略すると、比率は、問合せによって戻されるすべての行で計算されます。

expr には、RATIO_TO_REPORT または他の分析ファンクションを使用して分析ファンクションをネストできません。ただし、他の組込みファンクション式を *expr* で使用できます。*expr* の書式の詳細は、6-2 ページの「SQL 式」を参照してください。

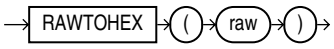
例

次の例では、すべての事務員の給与の合計に対する各事務員の給与の割合の値を計算します。

```
SELECT last_name, salary, RATIO_TO_REPORT(salary) OVER () AS rr
   FROM employees
   WHERE job_id = 'PU_CLERK'
   ORDER BY last_name, salary, rr;

LAST_NAME                 SALARY         RR
------------------------- ---------- ----------
Baida                       2900 .208633094
Colmenares                  2500 .179856115
Himuro                      2600  .18705036
Khoo                        3100 .223021583
Tobias                      2800 .201438849
```

RAWTOHEX

構文



用途

RAWTOHEX は、*raw* を 16 進表記で表した文字値に変換します。

SQL 組込みファンクションとして使用される場合、RAWTOHEX は、LONG、LONG RAW、CLOB、BLOB または BFILE 以外のすべてのスカラー・データ型の引数を取ります。このファンクションは、*raw* の値を構成するバイトの 16 進表記で VARCHAR2 値を戻します。各バイトは、2 桁の 16 進数で表されます。

> **注意：** RAWTOHEX は、PL/SQL 組込みファンクションとして使用される場合は異なる動作をします。詳細は、『Oracle Database アドバンスト・アプリケーション開発者ガイド』を参照してください。

例

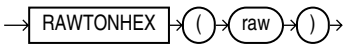
次の例は、RAW 列の値に等しい 16 進を戻します。

```
SELECT RAWTOHEX(raw_column) "Graphics"
   FROM graphics;

Graphics
--------
7D
```

> **参照：** 2-23 ページの「RAW データ型と LONG RAW データ型」および 5-78 ページの「HEXTORAW」を参照してください。

RAWTONHEX

構文



用途

RAWTONHEX は、*raw* を 16 進表記で表した文字値に変換します。RAWTONHEX (*raw*) は、TO_NCHAR(RAWTOHEX(*raw*)) と同じです。戻り値は、常に各国語キャラクタ・セットです。

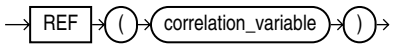
例

次の例は、RAW 列の値に等しい 16 進を戻します。

```
SELECT RAWTONHEX(raw_column),
   DUMP ( RAWTONHEX (raw_column) ) "DUMP"
   FROM graphics;

RAWTONHEX(RA)            DUMP
------------------------ ------------------------------
7D                       Typ=1 Len=4: 0,55,0,68
```

REF

構文



用途

REF は、引数としてオブジェクト表またはオブジェクト・ビューの行に関連付けられた相関変数（表別名）を取ります。変数または行にバインドされたオブジェクト・インスタンスについての REF 値が戻ります。

例

サンプル・スキーマ oe には、次のように定義された cust_address_typ という型が含まれます。

```
Attribute                           Type
----------------------------- ----------------
STREET_ADDRESS                VARCHAR2(40)
POSTAL_CODE                   VARCHAR2(10)
CITY                          VARCHAR2(30)
STATE_PROVINCE                VARCHAR2(10)
COUNTRY_ID                         CHAR(2)
```

次の例では、サンプル型 oe.cust_address_typ に基づく表（addresses 表）を作成した後、その表に行を挿入し、その表からその型のオブジェクト・インスタンスの REF 値を取り出します。

```
CREATE TABLE addresses OF cust_address_typ;

INSERT INTO addresses VALUES (
   '123 First Street', '4GF H1J', 'Our Town', 'Ourcounty', 'US');

SELECT REF(e) FROM addresses e;

REF(E)
--------------------------------------------------------------------------------
00002802097CD1261E51925B60E0340800208254367CD1261E51905B60E034080020825436010101820000
```

参照： REF の詳細は、『Oracle Database オブジェクト・リレーショナル開発者ガイド』を参照してください。

REFTOHEX

構文

用途

REFTOHEX は、引数 *expr* を 16 進で表した文字値に変換します。*expr* は、REF を戻す必要があります。

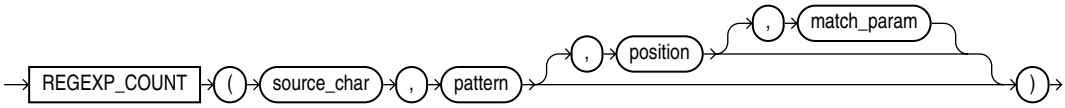
例

サンプル・スキーマ oe には warehouse_typ が含まれています。次の例では、その型を基に、ある列の REF 値をそれに等しい 16 進を含む文字値に変換する方法を示します。

```
CREATE TABLE warehouse_table OF warehouse_typ
   (PRIMARY KEY (warehouse_id));

CREATE TABLE location_table
   (location_number NUMBER, building REF warehouse_typ
   SCOPE IS warehouse_table);

INSERT INTO warehouse_table VALUES (1, 'Downtown', 99);

INSERT INTO location_table SELECT 10, REF(w) FROM warehouse_table w;

SELECT REFTOHEX(building) FROM location_table;

REFTOHEX(BUILDING)
--------------------------------------------------------------------------------
0000220208859B5E9255C31760E03408002082543685 9B5E9255C21760E034080020825436
```

REGEXP_COUNT

構文



用途

REGEXP_COUNT は、ソース文字列でパターンが発生した回数を戻すことによって、REGEXP_INSTR ファンクションの機能を補完します。このファンクションは、入力キャラクタ・セットによって定義された文字を使用して、文字列を評価します。また、*pattern* の発生回数を示す整数を戻します。一致する値が見つからない場合は 0（ゼロ）を戻します。

- *source_char* は、検索値として使用される文字式です。これは通常は文字列であり、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。
- *pattern* は正規表現です。これは通常はテキスト・リテラルであり、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型です。最大 512 バイトを指定できます。*pattern* のデータ型が *source_char* のデータ型と異なる場合、Oracle Database は *pattern* を *source_char* のデータ型に変換します。

  REGEXP_COUNT は、*pattern* 内の部分正規表現のカッコを無視します。たとえば、パターン '(123(45))' は、'12345' と同じです。*pattern* で指定できる演算子のリストは、付録 C「Oracle の正規表現のサポート」を参照してください。
- *position* は、Oracle が検索を開始する文字 *source_char* の位置を示す正の整数です。デフォルトは 1 で、*source_char* の最初の文字から検索が開始されます。1 番目の *pattern* の出現が検出されると、その後に続く文字以降の 2 番目の出現が検索されます。
- *match_param* は、ファンクションのデフォルトの検索動作を変更するためのテキスト・リテラルです。*match_param* には次の値を 1 つ以上指定できます。
  - 'i': 大 / 小文字を区別せずに検索します。
  - 'c': 大 / 小文字を区別して検索します。
  - 'n': 任意の文字に一致する文字であるピリオド（.）を、改行文字の一致に使用します。このパラメータを指定しない場合、ピリオドは改行文字には一致しません。

- 'm': ソース文字列を複数行として処理します。Oracle は、キャレット（^）およびドル記号（$）を、それぞれ、ソース文字列全体の開始または終了としてのみではなく、ソース文字列内の任意の場所にある任意の行の開始または終了としても解釈します。このパラメータを指定しない場合、Oracle はソース文字列を単一行として処理します。
- 'x' は空白文字を無視します。デフォルトでは、空白文字は空白文字として一致します。

複数の矛盾する値を指定すると、最後の値が使用されます。たとえば、'ic' を指定すると、大 / 小文字を区別する検索が行われます。前述以外の文字を指定すると、エラーが戻されます。

match_param を指定しない場合、次のようになります。

- 大 / 小文字を区別するかどうかのデフォルトは、NLS_SORT パラメータの値によって決まります。
- ピリオド（.）は改行文字に一致しません。
- ソース文字列は単一行として処理されます。

例

次の例では、pattern 内の部分正規表現のカッコが無視されることを示します。

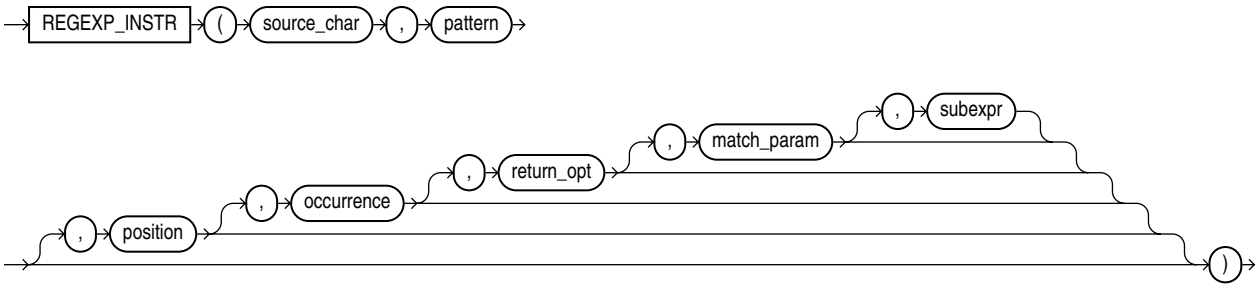
```
SELECT REGEXP_COUNT('123123123123123', '(12)3', 1, 'i') REGEXP_COUNT
   FROM DUAL;

REGEXP_COUNT
------------
           5
```

次の例では、ファンクションは 3 番目の文字でソース文字列の評価を開始するため、pattern の最初の出現はスキップされます。

```
SELECT REGEXP_COUNT('123123123123', '123', 3, 'i') COUNT FROM DUAL;

     COUNT
----------
         3
```

REGEXP_INSTR

構文



用途

REGEXP_INSTR は、正規表現パターンで文字列を検索できるように INSTR の機能を拡張したものです。このファンクションは、入力キャラクタ・セットによって定義された文字を使用して、文字列を評価します。また、*return_option* 引数の値に基づいて、一致したサブストリングの開始位置または終了位置を示す整数を戻します。一致する値が見つからない場合は 0（ゼロ）を戻します。

このファンクションは、POSIX 正規表現規格および Unicode Regular Expression Guidelines に準拠しています。詳細は、付録 C「Oracle の正規表現のサポート」を参照してください。

- *source_char* は、検索値として使用される文字式です。これは通常は文字列であり、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。
- *pattern* は正規表現です。これは通常はテキスト・リテラルであり、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型です。最大 512 バイトを指定できます。*pattern* のデータ型が *source_char* のデータ型と異なる場合、Oracle Database は *pattern* を *source_char* のデータ型に変換します。*pattern* で指定できる演算子のリストは、付録 C「Oracle の正規表現のサポート」を参照してください。
- *position* は、Oracle が検索を開始する文字 *source_char* の位置を示す正の整数です。デフォルトは 1 で、*source_char* の最初の文字から検索が開始されます。
- *occurrence* は、*source_char* に出現する *pattern* のうちのどれを検索するかを示す正の整数です。デフォルトは 1 で、最初に出現する *pattern* が検索されます。*occurrence* が 1 より大きい場合、1 番目の *pattern* が検出された後に続く 1 文字目から、2 番目（以降）の出現を検索します。この動作は、最初の出現の 2 番目の文字から 2 番目の出現を検索する INSTR ファンクションとは異なります。
- *return_option* には、出現した文字と関連して戻すものを指定できます。
  - 0（ゼロ）を指定すると、出現した文字の最初の文字の位置が戻されます。これはデフォルトです。
  - 1 を指定すると、出現した文字の次の文字の位置が戻されます。
- *match_parameter* は、ファンクションのデフォルトの検索動作を変更するためのテキスト・リテラルです。このパラメータの動作は、REGEXP_COUNT のこのファンクションの動作と同じです。詳細は、5-143 ページの「REGEXP_COUNT」を参照してください。
- 部分正規表現のある *pattern* の場合、*subexpr* は、ファンクションのターゲットとなる *pattern* 内の部分正規表現を示す 0 から 9 までの整数になります。*subexpr* は、カッコで囲まれた pattern のフラグメントです。部分正規表現はネストできます。部分正規表現は、その左側のカッコが pattern に出現する順に番号付けされます。たとえば、次の正規表現を考えます。

  ```
  0123(((abc)(de)f)ghi)45(678)
  ```

  この正規表現には、abcdefghi、abcdef、abc、de、678 の順に 5 つの部分正規表現があります。

  subexpr が 0（ゼロ）の場合、*pattern* に一致するサブストリング全体の位置が戻されます。*subexpr* が 0（ゼロ）より大きい場合、一致したサブストリング内の部分正規表現番号 *subexpr* に対応するサブストリング・フラグメントの位置が戻されます。*pattern* に *subexpr* の部分正規表現が 1 つもない場合、このファンクションは 0（ゼロ）を戻します。NULL の *subexpr* 値は、NULL を戻します。*subexpr* のデフォルト値は 0（ゼロ）です。

  参照：

  - 5-83 ページの「INSTR」および 5-149 ページの「REGEXP_SUBSTR」を参照してください。
  - 5-147 ページの「REGEXP_REPLACE」および 7-17 ページの「REGEXP_LIKE 条件」を参照してください。

例

次の例では、文字列を調べて、空白以外の文字を検索します。Oracle は、文字列の最初の文字から検索を開始し、空白以外の文字が 6 つ目に出現する開始位置（デフォルト）を戻します。

```
SELECT
  REGEXP_INSTR('500 Oracle Parkway, Redwood Shores, CA',
               '[^ ]+', 1, 6) "REGEXP_INSTR"
  FROM DUAL;

REGEXP_INSTR
------------
          37
```

次の例では、文字列を調べて、大 / 小文字を区別せずに s、r または p で始まり、任意の 6 つのアルファベット文字が続く単語を検索します。Oracle は、文字列の 3 つ目の文字から検索を開始し、大文字か小文字の s、r または p で始まる 7 文字の単語が 2 つ目に出現した後の文字の、文字列内の位置を戻します。

```
SELECT
  REGEXP_INSTR('500 Oracle Parkway, Redwood Shores, CA',
               '[s|r|p][[:alpha:]]{6}', 3, 2, 1, 'i') "REGEXP_INSTR"
  FROM DUAL;

REGEXP_INSTR
------------
          28
```

次の例では、*subexpr* 引数を使用して *pattern* 内の特定の部分正規表現を検索します。最初の文は、最初の部分正規表現にある最初の文字のソース文字列（123）の位置を戻します。

```
SELECT REGEXP_INSTR('1234567890', '(123)(4(56)(78))', 1, 1, 0, 'i', 1)
"REGEXP_INSTR" FROM DUAL;

REGEXP_INSTR
--------------------
1
```

次の文は、2 番目の部分正規表現にある最初の文字のソース文字列（45678）の位置を戻します。

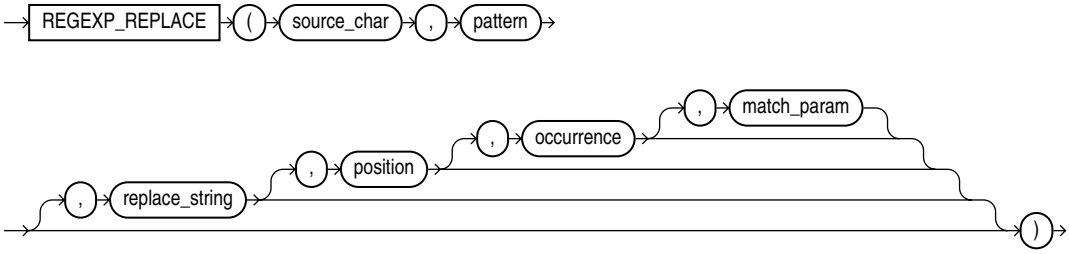
```
SELECT REGEXP_INSTR('1234567890', '(123)(4(56)(78))', 1, 1, 0, 'i', 2)
"REGEXP_INSTR" FROM DUAL;

REGEXP_INSTR
--------------------
4
```

次の文は、4 番目の部分正規表現にある最初の文字のソース文字列（78）の位置を戻します。

```
SELECT REGEXP_INSTR('1234567890', '(123)(4(56)(78))', 1, 1, 0, 'i', 4)
"REGEXP_INSTR" FROM DUAL;

REGEXP_INSTR
--------------------
7
```

REGEXP_REPLACE

構文



用途

REGEXP_REPLACE は、正規表現パターンで文字列を検索できるように REPLACE の機能を拡張したものです。デフォルトでは、このファンクションは、正規表現パターンに一致するすべての文字を *replace_string* に置き換えて *source_char* を戻します。*source_char* と同じキャラクタ・セットの文字列が戻されます。このファンクションは、1 つ目の引数が LOB ではない場合は VARCHAR2 を戻し、1 つ目の引数が LOB の場合は CLOB を戻します。

このファンクションは、POSIX 正規表現規格および Unicode Regular Expression Guidelines に準拠しています。詳細は、付録 C「Oracle の正規表現のサポート」を参照してください。

- *source_char* は、検索値として使用される文字式です。これは通常は文字列であり、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。
- *pattern* は正規表現です。これは通常はテキスト・リテラルであり、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型です。最大 512 バイトを指定できます。*pattern* のデータ型が *source_char* のデータ型と異なる場合、Oracle Database は *pattern* を *source_char* のデータ型に変換します。*pattern* で指定できる演算子のリストは、付録 C「Oracle の正規表現のサポート」を参照してください。
- *replace_string* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。*replace_string* が CLOB または NCLOB の場合、*replace_string* は 32KB に切り捨てられます。*replace_string* には、最大 500 個の部分正規表現への後方参照を \n という書式で指定できます。n は、1 ～ 9 の数値です。*replace_string* 内で n がバックスラッシュ文字に評価される場合、その前にエスケープ文字（\\）を付ける必要があります。後方参照表現の詳細は、「Oracle の正規表現のサポート」の C-2 ページの表 C-1 に示す説明を参照してください。
- *position* は、Oracle が検索を開始する文字 *source_char* の位置を示す正の整数です。デフォルトは 1 で、*source_char* の最初の文字から検索が開始されます。
- *occurrence* は、置換操作の対象となる文字を示す、負ではない整数です。
  - 0（ゼロ）を指定すると、一致したすべての文字が置換されます。
  - 正の整数 *n* を指定すると、*n* 番目に一致した文字が置換されます。

  occurrence が 1 より大きい場合、1 番目の *pattern* が検出された後に続く 1 文字目から、2 番目（以降）の出現を検索します。この動作は、最初の出現の 2 番目の文字から 2 番目の出現を検索する INSTR ファンクションとは異なります。
- *match_parameter* は、ファンクションのデフォルトの検索動作を変更するためのテキスト・リテラルです。このパラメータの動作は、REGEXP_COUNT のこのファンクションの動作と同じです。詳細は、5-143 ページの「REGEXP_COUNT」を参照してください。

参照：

- 「REPLACE」（5-155 ページ）
- 5-144 ページの「REGEXP_INSTR」、5-149 ページの「REGEXP_SUBSTR」および 7-17 ページの「REGEXP_LIKE 条件」を参照してください。

例

次の例では、phone_number を調べて、*xxx.xxx.xxxx* というパターンを検索します。その後、このパターンを (*xxx*) *xxx-xxxx* という書式に変更します。

```
SELECT
  REGEXP_REPLACE(phone_number,
                 '([[:digit:]]{3})\.([[:digit:]]{3})\.([[:digit:]]{4})',
                 '(\1) \2-\3') "REGEXP_REPLACE"
  FROM employees
  ORDER BY "REGEXP_REPLACE";

REGEXP_REPLACE
--------------------------------------------------------------------------------
(515) 123-4444
(515) 123-4567
(515) 123-4568
(515) 123-4569
(515) 123-5555
. . .
```

次の例では、country_name を調べます。文字列内の NULL でない各文字の後に空白を挿入します。

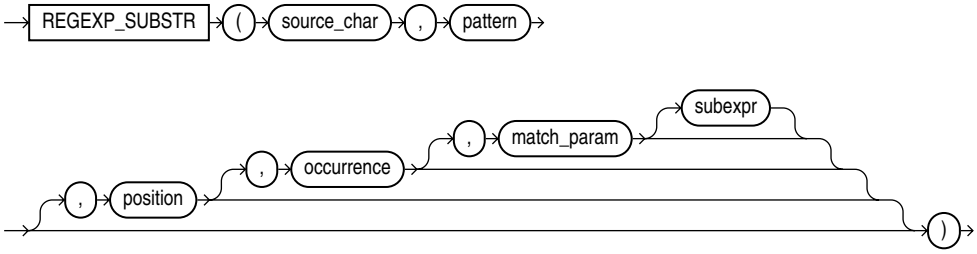
```
SELECT
  REGEXP_REPLACE(country_name, '(.)', '\1 ') "REGEXP_REPLACE"
  FROM countries;

REGEXP_REPLACE
--------------------------------------------------------------------------------
A r g e n t i n a
A u s t r a l i a
B e l g i u m
B r a z i l
C a n a d a
. . .
```

次の例では、文字列を調べて、2 つ以上の空白を検索します。検索された 2 つ以上の空白を、それぞれ 1 つの空白に置換します。

```
SELECT
  REGEXP_REPLACE('500   Oracle     Parkway,    Redwood  Shores, CA',
                 '( ){2,}', ' ') "REGEXP_REPLACE"
  FROM DUAL;

REGEXP_REPLACE
---------------------------------------
500 Oracle Parkway, Redwood Shores, CA
```

REGEXP_SUBSTR

構文



用途

REGEXP_SUBSTR は、正規表現パターンで文字列を検索できるように SUBSTR の機能を拡張したものです。REGEXP_INSTR と似ていますが、REGEXP_INSTR はサブストリングの位置ではなくサブストリング自体を戻します。このファンクションは、一致文字列の内容のみが必要で、ソース文字列内での位置は必要ない場合に有効です。このファンクションは、文字列を VARCHAR2 または CLOB データとして、*source_char* と同じキャラクタ・セットで戻します。

このファンクションは、POSIX 正規表現規格および Unicode Regular Expression Guidelines に準拠しています。詳細は、付録 C「Oracle の正規表現のサポート」を参照してください。

- *source_char* は、検索値として使用される文字式です。これは通常は文字列であり、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。
- *pattern* は正規表現です。これは通常はテキスト・リテラルであり、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型です。最大 512 バイトを指定できます。*pattern* のデータ型が *source_char* のデータ型と異なる場合、Oracle Database は *pattern* を *source_char* のデータ型に変換します。*pattern* で指定できる演算子のリストは、付録 C「Oracle の正規表現のサポート」を参照してください。
- *position* は、Oracle が検索を開始する文字 *source_char* の位置を示す正の整数です。デフォルトは 1 で、*source_char* の最初の文字から検索が開始されます。
- *occurrence* は、*source_char* に出現する *pattern* のうちのどれを検索するかを示す正の整数です。デフォルトは 1 で、最初に出現する *pattern* が検索されます。

  occurrence が 1 より大きい場合、1 番目の *pattern* が検出された後に続く 1 文字目から、2 番目（以降）の出現を検索します。この動作は、最初の出現の 2 番目の文字から 2 番目の出現を検索する SUBSTR ファンクションとは異なります。
- *match_parameter* は、ファンクションのデフォルトの検索動作を変更するためのテキスト・リテラルです。このパラメータの動作は、REGEXP_COUNT のこのファンクションの動作と同じです。詳細は、5-143 ページの「REGEXP_COUNT」を参照してください。
- 部分正規表現のある *pattern* の場合、*subexpr* は、ファンクションによって戻される *pattern* 内の部分正規表現を示す 0 から 9 までの負でない整数になります。このパラメータは、REGEXP_INSTR ファンクションの場合と同じセマンティクスを持ちます。詳細は、5-144 ページの「REGEXP_INSTR」を参照してください。

参照：

- 5-181 ページの「SUBSTR」および 5-144 ページの「REGEXP_INSTR」を参照してください。
- 5-147 ページの「REGEXP_REPLACE」および 7-17 ページの「REGEXP_LIKE 条件」を参照してください。

例

次の例では、文字列を調べて、カンマで区切られた最初のサブストリングを検索します。Oracle Database は、後にカンマが付いているカンマでない 1 つ以上の文字の前にあるカンマを検索します。該当するサブストリングが、前後のカンマを含めて戻されます。

```
SELECT
  REGEXP_SUBSTR('500 Oracle Parkway, Redwood Shores, CA',
                ',[^,]+,') "REGEXPR_SUBSTR"
  FROM DUAL;

REGEXPR_SUBSTR
-----------------
, Redwood Shores,
```

次の例では、文字列を調べて、1 つ以上の英数字を含むサブストリング、および任意でピリオド（.）が続く http:// を検索します。Oracle は、http:// と、スラッシュ（/）または文字列の末尾のいずれかとの間で、3 ～ 4 つのこのサブストリングを検索します。

```
SELECT
  REGEXP_SUBSTR('http://www.example.com/products',
                'http://([[:alnum:]]+\.?){3,4}/?') "REGEXP_SUBSTR"
  FROM DUAL;

REGEXP_SUBSTR
----------------------
http://www.example.com/
```

次の 2 つの例では、*subexpr* 引数を使用して *pattern* 内の特定の部分正規表現を戻します。最初の文は、*pattern* 内の最初の部分正規表現を戻します。

```
SELECT REGEXP_SUBSTR('1234567890', '(123)(4(56)(78))', 1, 1, 'i', 1)
"REGEXP_SUBSTR" FROM DUAL;

REGEXP_SUBSTR
-------------------
123
```

次の文は、*pattern* 内の 4 番目の部分正規表現を戻します。

```
SELECT REGEXP_SUBSTR('1234567890', '(123)(4(56)(78))', 1, 1, 'i', 4)
"REGEXP_SUBSTR" FROM DUAL;

REGEXP_SUBSTR
-------------------
78
```

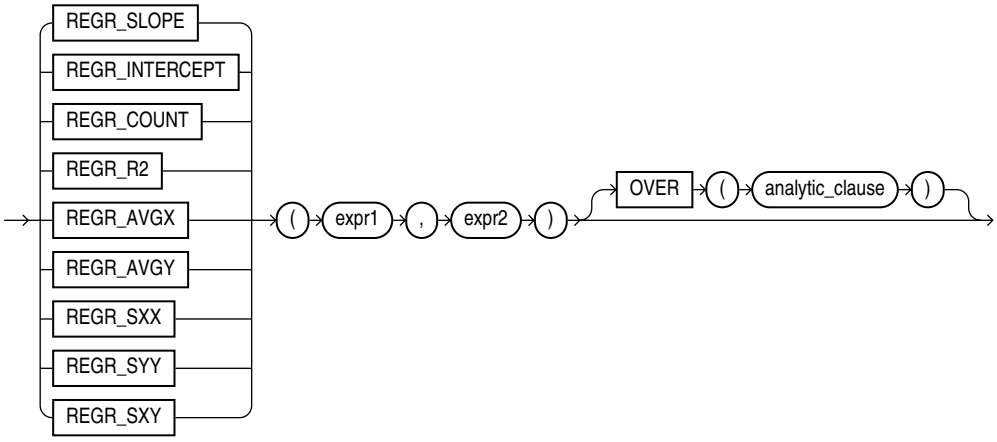
REGR_（線形回帰）ファンクション

線形回帰ファンクションは次のとおりです。

- REGR_SLOPE
- REGR_INTERCEPT
- REGR_COUNT
- REGR_R2
- REGR_AVGX
- REGR_AVGY
- REGR_SXX
- REGR_SYY
- REGR_SXY

構文

***linear_regr*::=**



> **参照：** 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

線形回帰ファンクションは、微分最小 2 乗法で求めた回帰直線を数値の組の集合に対応付けます。これは、集計ファンクションまたは分析ファンクションとして使用できます。

> **参照：** *expr* の書式の詳細は、5-8 ページの「集計ファンクション」および 6-2 ページの「SQL 式」を参照してください。

これらのファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。Oracle は、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。数値の優先順位の詳細は、2-15 ページの「数値の優先順位」を参照してください。

Oracle は、*expr1* または *expr2* が NULL であるすべての組を排除した後、このファンクションを (*expr1* , *expr2*) の集合に適用します。Oracle は、データでの引渡し中、同時にすべての回帰ファンクションを計算します。

expr1 は、従属変数の値（y 値）として解析されます。*expr2* は、独立変数の値（x 値）として解析されます。

- REGR_SLOPE は、直線の傾きを戻します。戻り値は数値データ型で、NULL になる場合もあります。NULL (*expr1*, *expr2*) の組を排除した後、このファンクションは次の計算を行います。

  ```
  COVAR_POP(expr1, expr2) / VAR_POP(expr2)
  ```

- REGR_INTERCEPT は、回帰ファンクションの y 切片を戻します。戻り値は数値データ型で、NULL になる場合もあります。NULL (*expr1*, *expr2*) の組を排除した後、このファンクションは次の計算を行います。

  ```
  AVG(expr1) - REGR_SLOPE(expr1, expr2) * AVG(expr2)
  ```

- REGR_COUNT は整数を戻します。この整数は、回帰ファンクションに適応させるために使用される NULL でない数値の組です。

- REGR_R2 は、回帰に対する決定係数（R の 2 乗または適合度とも呼ぶ）を戻します。戻り値は数値データ型で、NULL になる場合もあります。VAR_POP (*expr1*) および VAR_POP (*expr2*) は、NULL の組が排除された後に評価されます。戻り値は次のとおりです。

```
                    NULL if VAR_POP(expr2)  = 0

                       1 if VAR_POP(expr1)  = 0 and
                            VAR_POP(expr2) != 0

POWER(CORR(expr1,expr),2) if VAR_POP(expr1)  > 0 and
                            VAR_POP(expr2  != 0
```

これ以外のすべての回帰ファンクションの戻り値は数値データ型で、NULL になる場合もあります。

- REGR_AVGX は、回帰直線の独立変数（*expr2*）の平均を求めます。NULL (*expr1*、*expr2*) の組を排除した後、このファンクションは次の計算を行います。

```
AVG(expr2)
```

- REGR_AVGY は、回帰直線の従属変数（*expr1*）の平均を求めます。NULL (*expr1*、*expr2*) の組を排除した後、このファンクションは次の計算を行います。

```
AVG(expr1)
```

REGR_SXY、REGR_SXX、REGR_SYY は補助ファンクションです。これらは、様々な診断統計の計算に使用されます。

- NULL (*expr1*、*expr2*) の組を排除した後、REGR_SXX は次の計算を行います。

```
REGR_COUNT(expr1, expr2) * VAR_POP(expr2)
```

- NULL (*expr1*、*expr2*) の組を排除した後、REGR_SYY は次の計算を行います。

```
REGR_COUNT(expr1, expr2) * VAR_POP(expr1)
```

- NULL (*expr1*、*expr2*) の組を排除した後、REGR_SXY は次の計算を行います。

```
REGR_COUNT(expr1, expr2) * COVAR_POP(expr1, expr2)
```

次の例は、サンプル表 sh.sales および sh.products に基づいています。

一般的な線形回帰の例

次の例では、分析書式で使用される様々な線形回帰ファンクションを比較します。これらのファンクションの分析書式は、モデル別で従業員ごとに予測される給与の検索などの計算で、回帰統計を使用する場合に有効です。次の個々の線形回帰ファンクションについての項では、これらのファンクションの集計書式の例を示します。

```
SELECT job_id, employee_id ID, salary,
REGR_SLOPE(SYSDATE-hire_date, salary)
   OVER (PARTITION BY job_id) slope,
REGR_INTERCEPT(SYSDATE-hire_date, salary)
   OVER (PARTITION BY job_id) intcpt,
REGR_R2(SYSDATE-hire_date, salary)
   OVER (PARTITION BY job_id) rsqr,
REGR_COUNT(SYSDATE-hire_date, salary)
   OVER (PARTITION BY job_id) count,
REGR_AVGX(SYSDATE-hire_date, salary)
   OVER (PARTITION BY job_id) avgx,
REGR_AVGY(SYSDATE-hire_date, salary)
   OVER (PARTITION BY job_id) avgy
   FROM employees
   WHERE department_id in (50, 80)
   ORDER BY job_id, employee_id;
```

```
JOB_ID        ID     SALARY SLOPE    INTCPT  RSQR  COUNT       AVGX      AVGY
---------- ----- ---------- ----- --------- ----- ------ ---------- ---------
SA_MAN       145      14000  .355 -1707.035  .832      5  12200.000  2626.589
SA_MAN       146      13500  .355 -1707.035  .832      5  12200.000  2626.589
SA_MAN       147      12000  .355 -1707.035  .832      5  12200.000  2626.589
SA_MAN       148      11000  .355 -1707.035  .832      5  12200.000  2626.589
SA_MAN       149      10500  .355 -1707.035  .832      5  12200.000  2626.589
SA_REP       150      10000  .257   404.763  .647     29   8396.552  2561.244
SA_REP       151       9500  .257   404.763  .647     29   8396.552  2561.244
SA_REP       152       9000  .257   404.763  .647     29   8396.552  2561.244
SA_REP       153       8000  .257   404.763  .647     29   8396.552  2561.244
SA_REP       154       7500  .257   404.763  .647     29   8396.552  2561.244
SA_REP       155       7000  .257   404.763  .647     29   8396.552  2561.244
SA_REP       156      10000  .257   404.763  .647     29   8396.552  2561.244
...
```

REGR_SLOPE および REGR_INTERCEPT の例

次の例では、サンプル表 hr.employees を使用して、勤務時間（SYSDATE - hire_date）と給与に関する線形回帰モデルの傾きと回帰を計算します。結果は job_id 別にグループ化されます。

```
SELECT job_id,
REGR_SLOPE(SYSDATE-hire_date, salary) slope,
REGR_INTERCEPT(SYSDATE-hire_date, salary) intercept
   FROM employees
   WHERE department_id in (50,80)
   GROUP BY job_id
   ORDER BY job_id;

JOB_ID     SLOPE    INTERCEPT
---------- ----- ------------
SA_MAN      .355 -1707.030762
SA_REP      .257   404.767151
SH_CLERK    .745   159.015293
ST_CLERK    .904   134.409050
ST_MAN      .479  -570.077291
```

REGR_COUNT の例

次の例では、サンプル表 hr.employees を使用して、勤務時間（SYSDATE - hire_date）と給与について、job_id 別の数を計算します。結果は job_id 別にグループ化されます。

```
SELECT job_id,
REGR_COUNT(SYSDATE-hire_date, salary) count
   FROM employees
   WHERE department_id in (30, 50)
   GROUP BY job_id
   ORDER BY job_id, count;

JOB_ID          COUNT
---------- ----------
PU_CLERK             5
PU_MAN               1
SH_CLERK            20
ST_CLERK            20
ST_MAN               5
```

REGR_R2 の例

次の例では、サンプル表 hr.employees を使用して、勤務時間（SYSDATE - hire_date）と給与の線形回帰に対する決定係数を計算します。

```
SELECT job_id,
REGR_R2(SYSDATE-hire_date, salary) Regr_R2
   FROM employees
   WHERE department_id in (80, 50)
   GROUP by job_id
  ORDER BY job_id, Regr_R2;

JOB_ID        REGR_R2
---------- ----------
SA_MAN      .83244748
SA_REP     .647007156
SH_CLERK   .879799698
ST_CLERK   .742808493
ST_MAN      .69418508
```

REGR_AVGY および REGR_AVGX の例

次の例では、サンプル表 hr.employees を使用して、勤務時間（SYSDATE - hire_date）と給与の平均値を計算します。結果は job_id 別にグループ化されます。

```
SELECT job_id,
REGR_AVGY(SYSDATE-hire_date, salary) avgy,
REGR_AVGX(SYSDATE-hire_date, salary) avgx
   FROM employees
   WHERE department_id in (30,50)
   GROUP BY job_id
   ORDER BY job_id, avgy, avgx;

JOB_ID           AVGY       AVGX
---------- ---------- ----------
PU_CLERK    2950.3778       2780
PU_MAN      4026.5778      11000
SH_CLERK    2773.0778       3215
ST_CLERK    2872.7278       2785
ST_MAN      3140.1778       7280
```

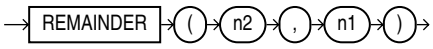
REGR_SXY、REGR_SXX および REGR_SYY の例

次の例では、サンプル表 hr.employees を使用して、勤務時間（SYSDATE - hire_date）と給与の線形回帰について、3 種類の診断統計を計算します。

```
SELECT job_id,
REGR_SXY(SYSDATE-hire_date, salary) regr_sxy,
REGR_SXX(SYSDATE-hire_date, salary) regr_sxx,
REGR_SYY(SYSDATE-hire_date, salary) regr_syy
   FROM employees
   WHERE department_id in (80, 50)
   GROUP BY job_id
   ORDER BY job_id;

JOB_ID       REGR_SXY    REGR_SXX   REGR_SYY
---------- ---------- ----------- ----------
SA_MAN        3303500   9300000.0    1409642
SA_REP     16819665.5  65489655.2 6676562.55
SH_CLERK      4248650   5705500.0    3596039
ST_CLERK      3531545   3905500.0 4299084.55
ST_MAN        2180460   4548000.0  1505915.2
```

REMAINDER

構文



用途

REMAINDER は *n2* を *n1* で割った余りを戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。Oracle は、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

MOD ファンクションは REMAINDER と似ていますが、MOD の式では FLOOR を使用します。これに対して REMAINDER では ROUND を使用します。詳細は、5-102 ページの「MOD」を参照してください。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。数値の優先順位の詳細は、2-15 ページの「数値の優先順位」を参照してください。

- *n1* が 0（ゼロ）であるか、または *n2* が無限大の場合、Oracle が戻すものは次のようになります。
  - 両方の引数が NUMBER 型の場合はエラー
  - 両方の引数が BINARY_FLOAT または BINARY_DOUBLE の場合は NaN
- *n1* が 0 ではない場合、余りは *n2* - (*n1***N*) となります。*N* は、*n2*/*n1* に最も近い整数です。*n2*/*n1* が x.5 と等しい場合、*N* は最も近い偶数の整数です。
- *n2* が浮動小数点数で、余りが 0（ゼロ）の場合、余りの符号は *n2* の符合になります。NUMBER 値の場合、0 の余りには符号は付きません。

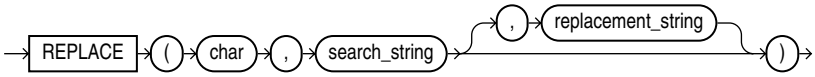
例

次の例では、5-197 ページの「例」で作成された float_point_demo 表を使用して、2 つの浮動小数点数を割り、算出された余りを戻します。

```
SELECT bin_float, bin_double, REMAINDER(bin_float, bin_double)
  FROM float_point_demo;

 BIN_FLOAT BIN_DOUBLE REMAINDER(BIN_FLOAT,BIN_DOUBLE)
---------- ---------- --------------------------------
1.235E+003 1.235E+003                      5.859E-005
```

REPLACE

構文



用途

REPLACE は、*replacement_string* ですべての *search_string* を変換して *char* を戻します。*replacement_string* を指定しない場合または NULL の場合、すべての *search_string* が削除されます。*search_string* が NULL の場合、*char* が戻されます。

char と同様に、*search_string* および *replacement_string* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。*char* と同じキャラクタ・セットの文字列が戻されます。このファンクションは、1 つ目の引数が LOB ではない場合は VARCHAR2 を戻し、1 つ目の引数が LOB の場合は CLOB を戻します。

REPLACE は、TRANSLATE ファンクションに関連する機能を提供します。TRANSLATE は、単一文字を 1 対 1 で置き換えます。REPLACE ファンクションでは、1 つの文字列の置換および複数の文字列の削除を実行できます。

参照： 「TRANSLATE」（5-213 ページ）

例

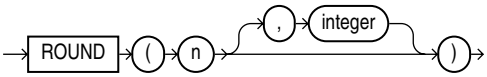
次の例では、J を BL に置換します。

```
SELECT REPLACE('JACK and JUE','J','BL') "Changes"
     FROM DUAL;

Changes
--------------
BLACK and BLUE
```

ROUND（数値）

構文

***round_number*::=**



用途

ROUND は、*n* を小数点以下 *integer* 桁に丸めた値を戻します。*integer* を指定しない場合、*n* は小数点以下が丸められます。*integer* が負の場合、*n* は小数点の左側に丸められます。

n には、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を指定できます。*integer* を指定しない場合、このファンクションは、*n* の数値データ型と同じデータ型で値 ROUND(n, 0) を戻します。*integer* を指定すると、このファンクションは NUMBER を戻します。

ROUND は、次の規則を使用して実行されます。

1. *n* が 0 の場合、ROUND は *integer* に関係なく常に 0 を戻します。
2. *n* が負の場合、ROUND(n, integer) は、-ROUND(-n, integer) を戻します。
3. *n* が正の場合は、次のとおりです。

```
ROUND(n, integer) = FLOOR(n * POWER(10, integer) + 0.5) * POWER(10, -integer)
```

NUMBER 値に適用された ROUND が、浮動小数点で表現された同じ値に適用された ROUND と少し異なる場合があります。この結果の相違は、NUMBER と浮動小数点値の内部表現の違いから発生します。相違が発生した場合、その相違は丸められた桁で 1 になります。

参照： 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。Oracle Database による BINARY_FLOAT および BINARY_DOUBLE 値の処理方法の詳細は、2-13 ページの「浮動小数点数」を参照してください。

例

次の例では、数値を小数点以下 1 桁に丸めます。

```
SELECT ROUND(15.193,1) "Round" FROM DUAL;

     Round
----------
      15.2
```

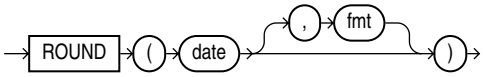
次の例では、数値の小数点の左 1 桁を丸めます。

```
SELECT ROUND(15.193,-1) "Round" FROM DUAL;

     Round
----------
        20
```

ROUND（日付）

構文

***round_date*::=**



用途

ROUND は、*date* を書式モデル *fmt* で指定した単位に丸めた結果を戻します。このファンクションは、NLS_CALENDAR セッション・パラメータの影響を受けません。このファンクションはグレゴリオ暦の規則に従って動作します。戻される値は、*date* に異なる日時データ型を指定した場合でも、常に DATE データ型です。*fmt* を省略すると、*date* は最も近い日に丸められます。*date* 式は、DATE 値に変換される必要があります。

> **参照：** *fmt* で使用できる書式モデルについては、5-248 ページの「ROUND および TRUNC 日付ファンクション」を参照してください。

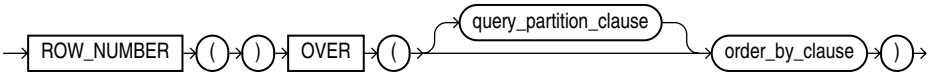
例

次の例では、次の年の最も近い日に丸めます。

```
SELECT ROUND (TO_DATE ('27-OCT-00'),'YEAR')
   "New Year" FROM DUAL;

New Year
---------
01-JAN-01
```

ROW_NUMBER

構文



参照： 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

ROW_NUMBER は分析ファンクションです。このファンクションは、*order_by_clause* に指定された行の、1 から始まる順序シーケンスで、このファンクションが適用される各行（パーティションの各行、または問合せが戻す各行）に一意の数値を割り当てます。

指定された範囲の ROW_NUMBER 値を取得する問合せ内に、ROW_NUMBER を使用した副問合せをネストすることで、内側の問合せの結果から正確な行のサブセットを得ることができます。この方法でこのファンクションを使用すると、上位 N 番、下位 N 番および内部 N 番のレポートを実行できます。一貫した結果を戻すには、問合せで決定的なソート順序を使用する必要があります。

expr には、ROW_NUMBER または他の分析ファンクションを使用して分析ファンクションをネストできません。ただし、他の組込みファンクション式を *expr* で使用できます。*expr* の書式の詳細は、6-2 ページの「SQL 式」を参照してください。

例

次の例では、hr.employees 表で各部門の最も支給額の高い上位 3 人の従業員を検索します。従業員が 3 人未満の部門については、3 行未満の行が戻されます。

```
SELECT department_id, first_name, last_name, salary
FROM
(
  SELECT
    department_id, first_name, last_name, salary,
    ROW_NUMBER() OVER (PARTITION BY department_id ORDER BY salary desc) rn
  FROM employees
)
WHERE rn <= 3
ORDER BY department_id, salary DESC, last_name;
```

次の例は、sh.sales 表の結合問合せです。1999 年の売上高が上位 5 つの製品について 2000 年の売上高を検索し、2000 年と 1999 年の差を比較します。各販売チャネルにおいて売上高上位の 10 製品が計算されます。

```
SELECT sales_2000.channel_desc, sales_2000.prod_name,
       sales_2000.amt amt_2000,  top_5_prods_1999_year.amt amt_1999,
       sales_2000.amt  - top_5_prods_1999_year.amt amt_diff
FROM
/* The first subquery finds the 5 top-selling products per channel in year 1999. */
  (SELECT channel_desc, prod_name, amt
   FROM
   (
     SELECT channel_desc, prod_name, sum(amount_sold) amt,
       ROW_NUMBER () OVER (PARTITION BY channel_desc
                           ORDER BY SUM(amount_sold) DESC) rn
     FROM sales, times, channels, products
     WHERE sales.time_id = times.time_id
       AND times.calendar_year = 1999
       AND channels.channel_id = sales.channel_id
       AND products.prod_id = sales.prod_id
```

```
    GROUP BY channel_desc, prod_name
   )
   WHERE rn <= 5
  ) top_5_prods_1999_year,
/* The next subquery finds sales per product and per channel in 2000. */
  (SELECT channel_desc, prod_name, sum(amount_sold) amt
     FROM sales, times, channels, products
     WHERE sales.time_id = times.time_id
       AND times.calendar_year = 2000
       AND channels.channel_id = sales.channel_id
       AND products.prod_id = sales.prod_id
     GROUP BY channel_desc, prod_name
  ) sales_2000
WHERE sales_2000.channel_desc = top_5_prods_1999_year.channel_desc
  AND sales_2000.prod_name = top_5_prods_1999_year.prod_name
ORDER BY sales_2000.channel_desc, sales_2000.prod_name
;
CHANNEL_DESC    PROD_NAME                                  AMT_2000   AMT_1999   AMT_DIFF
--------------- --------------==-------------------------------- ---------- ---------- ----------
Direct Sales     17" LCD w/built-in HDTV Tuner              628855.7 1163645.78 -534790.08
Direct Sales     Envoy 256MB - 40GB                         502938.54  843377.88 -340439.34
Direct Sales     Envoy Ambassador                          2259566.96 1770349.25  489217.71
Direct Sales     Home Theatre Package with DVD-Audio/Video Play  1235674.15 1260791.44  -25117.29
Direct Sales     Mini DV Camcorder with 3.5" Swivel LCD     775851.87 1326302.51 -550450.64
Internet         17" LCD w/built-in HDTV Tuner               31707.48   160974.7 -129267.22
Internet         8.3 Minitower Speaker                      404090.32  155235.25  248855.07
Internet         Envoy 256MB - 40GB                          28293.87  154072.02 -125778.15
Internet         Home Theatre Package with DVD-Audio/Video Play   155405.54  153175.04     2230.5
Internet         Mini DV Camcorder with 3.5" Swivel LCD      39726.23  189921.97 -150195.74
Partners         17" LCD w/built-in HDTV Tuner              269973.97  325504.75  -55530.78
Partners         Envoy Ambassador                          1213063.59  614857.93  598205.66
Partners         Home Theatre Package with DVD-Audio/Video Play   700266.58  520166.26  180100.32
Partners         Mini DV Camcorder with 3.5" Swivel LCD     404265.85  520544.11 -116278.26
Partners         Unix/Windows 1-user pack                   374002.51  340123.02   33879.49

15 rows selected.
```

ROWIDTOCHAR

構文

用途

ROWIDTOCHAR は、ROWID の値を VARCHAR2 データ型に変換します。この変換の結果は常に 18 文字です。

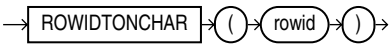
例

次の例では、employees 表の ROWID 値を文字値に変換します。（結果はサンプル・データベースのビルドごとに異なります。）

```
SELECT ROWID FROM employees
   WHERE ROWIDTOCHAR(ROWID) LIKE '%JAAB%'
   ORDER BY ROWID;

ROWID
------------------
AAAFfIAAFAAAABSAAb
```

ROWIDTONCHAR

構文



用途

ROWIDTONCHAR は、ROWID 値を NVARCHAR2 データ型に変換します。この変換の結果は、常に、各国語キャラクタ・セットの文字で、18 文字です。

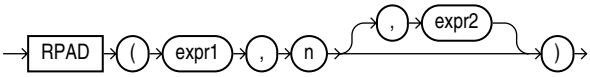
例

次の例では、ROWID 値を NVARCHAR2 文字列に変換します。

```
SELECT LENGTHB( ROWIDTONCHAR(ROWID) ) Length, ROWIDTONCHAR(ROWID)
   FROM employees
   ORDER BY length;

    LENGTH ROWIDTONCHAR(ROWID
---------- ------------------
        36 AAAL52AAFAAAABSABD
        36 AAAL52AAFAAAABSABV
. . .
```

RPAD

構文



用途

RPAD は、*expr1* の右に *expr2* で指定した文字を必要に応じて連続的に埋め込み、長さ *n* にして戻します。このファンクションは、問合せの出力の書式設定に役立ちます。

expr1 および *expr2* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻される文字列は、*expr1* が文字データ型の場合は VARCHAR2 データ型、*expr1* が各国語キャラクタ・データ型の場合は NVARCHAR2 データ型、*expr1* が LOB データ型の場合は LOB データ型になります。*expr1* と同じキャラクタ・セットの文字列が戻されます。引数 *n* は、NUMBER 型の整数か、または NUMBER 型の整数に暗黙的に変換可能な値である必要があります。

expr1 は NULL 以外である必要があります。*expr2* を指定しない場合、デフォルトで空白 1 個が指定されます。*expr1* が *n* より長い場合、このファンクションは *n* に収まる *expr1* の一部を戻します。

引数 *n* は、戻り値が画面に表示される場合の全体の長さです。多くのキャラクタ・セットでは、これは戻り値の文字数でもあります。ただし、マルチバイトのキャラクタ・セットでは、表示される文字列の長さが文字列の文字数と異なる場合もあります。

例

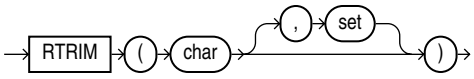
次の例では、空白にアスタリスクを埋め込んで、給与額の単純なチャートを作成します。

```
SELECT last_name, RPAD(' ', salary/1000/1, '*') "Salary"
   FROM employees
   WHERE department_id = 80
   ORDER BY last_name, "Salary";
```

```
LAST_NAME                 Salary
------------------------- ---------------
Abel                       **********
Ande                       *****
Banda                      *****
Bates                      ******
Bernstein                  ********
Bloom                      *********
Cambrault                  **********
Cambrault                  ******
Doran                      ******
Errazuriz                  ***********
Fox                        ********
Greene                     ********
Hall                       ********
Hutton                     *******
Johnson                    *****
King                       *********
. . .
```

RTRIM

構文



用途

RTRIM は、*char* の右端から、*set* に指定されたすべての文字を削除します。このファンクションは、問合せの出力の書式設定に役立ちます。

set を指定しない場合、デフォルトで空白 1 個が指定されます。*char* が文字リテラルの場合、一重引用符で囲む必要があります。RTRIM は LTRIM と同様の働きをします。

char および *set* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻される文字列は、*char* が文字データ型の場合は VARCHAR2 データ型、*char* が各国語キャラクタデータ型の場合は NVARCHAR2 データ型、*char* が LOB データ型の場合は LOB データ型になります。

参照： 「LTRIM」（5-96 ページ）

例

次の例では、文字列の右端にあるすべてのピリオド、スラッシュおよび等号を文字列から削除します。

```
SELECT RTRIM('BROWNING: ./=./=./=./=./=.=','/=.') "RTRIM example" FROM DUAL;

RTRIM exam
----------
BROWNING:
```

SCN_TO_TIMESTAMP

構文

用途

SCN_TO_TIMESTAMP は、引数として、システム変更番号（SCN）と評価される数値を取り、その SCN に関連付けられた概数のタイムスタンプを戻します。戻り値のデータ型は TIMESTAMP です。このファンクションは、SCN に関連付けられたタイムスタンプを調べる場合に有効です。たとえば、このファンクションを ORA_ROWSCN 疑似列とともに使用して、行への最新の変更にタイムスタンプを関連付けることができます。

注意：

- 結果値の通常の精度は 3 秒です。
- SCN と SCN 生成時のタイムスタンプの関連は、一定期間データベースで記憶されます。この期間は、最大で自動調整された UNDO 保存期間（データベースが自動 UNDO 管理モードで実行されている場合）およびデータベース内のすべてのフラッシュバック・アーカイブの保存時間となりますが、120 時間以上になります。関連が不要となるまでの経過時間には、データベースが開かれているときの時間のみが加算されます。SCN_TO_TIMESTAMP の引数に対して指定された SCN が古すぎる場合は、エラーが戻されます。

参照： 3-8 ページの「ORA_ROWSCN 疑似列」および 5-196 ページの「TIMESTAMP_TO_SCN」を参照してください。

例

次の例では、ORA_ROWSCN 疑似列を使用して行への最新の変更のシステム変更番号を判断し、SCN_TO_TIMESTAMP を使用してその SCN をタイムスタンプに変換します。

```
SELECT SCN_TO_TIMESTAMP(ORA_ROWSCN) FROM employees
   WHERE employee_id = 188;
```

このような問合せを使用して、Oracle フラッシュバック問合せで使用するためにシステム変更番号をタイムスタンプに変換することもできます。

```
SELECT salary FROM employees WHERE employee_id = 188;
    SALARY
----------
      3800

UPDATE employees SET salary = salary*10 WHERE employee_id = 188;
COMMIT;

SELECT salary FROM employees WHERE employee_id = 188;
    SALARY
----------
     38000

SELECT SCN_TO_TIMESTAMP(ORA_ROWSCN) FROM employees
   WHERE employee_id = 188;
SCN_TO_TIMESTAMP(ORA_ROWSCN)
---------------------------------------------------------------------------
28-AUG-03 01.58.01.000000000 PM
```

```
FLASHBACK TABLE employees TO TIMESTAMP
   TO_TIMESTAMP('28-AUG-03 01.00.00.000000000 PM');

SELECT salary FROM employees WHERE employee_id = 188;
    SALARY
----------
      3800
```

SESSIONTIMEZONE

構文

用途

SESSIONTIMEZONE は、現行のセッションのタイムゾーンを戻します。戻り型は、タイムゾーン・オフセット（'[+|]TZH:TZM' という書式の文字列型）またはタイムゾーン地域名です。これは、最近の ALTER SESSION 文でユーザーが指定したセッション・タイムゾーンの値によって異なります。

> **注意：** 環境変数 ORA_SDTZ を使用して、デフォルトのクライアント・セッションのタイムゾーンを設定できます。この変数の詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

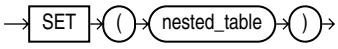
例

次の例では、現行のセッションのタイムゾーンを戻します。

```
SELECT SESSIONTIMEZONE FROM DUAL;

SESSION
-------
-08:00
```

SET

構文



用途

SET は、重複を排除することによってネストした表を単一の集合に変換します。このファンクションは、各要素が一意であるネストした表を戻します。戻されるネストした表のデータ型は、入力されたネストした表と同じです。

ネストした表の要素の型は、比較可能な型である必要があります。スカラー型以外の型の比較の可能性は、7-4 ページの「比較条件」を参照してください。

例

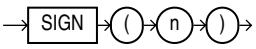
次の例では、customers_demo 表から、ネストした表列 cust_address_ntab の一意の要素を選択します。

```
SELECT customer_id, SET(cust_address_ntab) address
  FROM customers_demo
  ORDER BY customer_id;

CUSTOMER_ID ADDRESS(STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE, COUNTRY_ID)
----------- --------------------------------------------------------------------
        101 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('514 W Superior St', '46901', 'Kokomo', 'IN', 'US'))
        102 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('2515 Bloyd Ave', '46218', 'Indianapolis', 'IN', 'US'))
        103 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('8768 N State Rd 37', '47404', 'Bloomington', 'IN', 'US'))
        104 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('6445 Bay Harbor Ln', '46254', 'Indianapolis', 'IN', 'US'))
        105 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('4019 W 3Rd St', '47404', 'Bloomington', 'IN', 'US'))
. . .
```

この例では、表 customers_demo と、データを含むネストした表の列が 1 つ必要です。この表およびネストした表の列を作成する方法については、4-2 ページの「MULTISET 演算子」を参照してください。

SIGN

構文



用途

SIGN は、*n* の符号を戻します。このファンクションは、引数として、任意の数値データ型、または暗黙的に NUMBER に変換可能な数値以外のデータ型を取り、NUMBER を戻します。

NUMBER 型の値の場合、符号は次のとおりです。

- *n*<0（ゼロ）の場合、-1
- *n*=0 の場合、0（ゼロ）
- *n*>0 の場合、1

浮動小数点数（BINARY_FLOAT および BINARY_DOUBLE）の場合、このファンクションは数値の符号ビットを戻します。符号ビットは次のとおりです。

- *n*<0（ゼロ）の場合、-1
- *n*>=0 または *n*=NaN の場合、+1

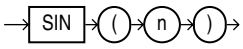
例

次の例では、ファンクションの引数（-15）が 0 より小さいことを示します。

```
SELECT SIGN(-15) "Sign" FROM DUAL;

      Sign
----------
        -1
```

SIN

構文



用途

SIN は、*n*（ラジアンで表された角度）のサインを戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数が BINARY_FLOAT の場合、このファンクションは BINARY_DOUBLE を戻します。それ以外の場合、引数と同じ数値データ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

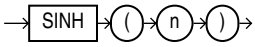
例

次の例では、30 度のサインを戻します。

```
SELECT SIN(30 * 3.14159265359/180)
 "Sine of 30 degrees" FROM DUAL;

Sine of 30 degrees
------------------
                .5
```

SINH

構文



用途

SINH は、*n* の双曲線サインを戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数が BINARY_FLOAT の場合、このファンクションは BINARY_DOUBLE を戻します。それ以外の場合、引数と同じ数値データ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

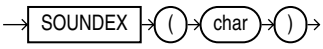
例

次の例では、1 の双曲線サインを戻します。

```
SELECT SINH(1) "Hyperbolic sine of 1" FROM DUAL;

Hyperbolic sine of 1
--------------------
          1.17520119
```

SOUNDEX

構文



用途

SOUNDEX は、*char* と同じ音声表現を持つ文字列を戻します。このファンクションによって、綴りが異なっても発音が類似した英単語を比較できます。

音声表現については、『The Art of Computer Programming, Volume 3: Sorting and Searching』（Donald E. Knuth 著）で次のように定義されています。

1. 文字列の最初の文字を残し、a、e、h、i、o、u、w、y の文字が出てきた場合にはすべて削除します。
2. 残った 2 文字目以降の文字に対し、次のように数値を割り当てます。

```
b, f, p, v = 1
c, g, j, k, q, s, x, z = 2
d, t = 3
l = 4
m, n = 5
r = 6
```

3. 元の名前（1 つ目の定義を行う前）で、同じ数値を持つ 2 つ以上の文字が並んでいるか、または h と w の間の文字以外と並んでいる場合は、最初の文字以外のすべての文字を削除します。
4. 最初の 4 バイトを 0（ゼロ）で埋めて戻します。

char は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型です。戻り値は、*char* と同じデータ型です。

このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。

> **参照：** 詳細は、2-36 ページの「データ型の比較規則」を参照してください。

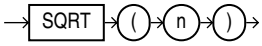
例

次の例では、「Smyth」と同じ音声表現を持つ従業員を戻します。

```
SELECT last_name, first_name
     FROM hr.employees
     WHERE SOUNDEX(last_name)
         = SOUNDEX('SMYTHE')
     ORDER BY last_name, first_name;

LAST_NAME  FIRST_NAME
---------- ----------
Smith      Lindsey
Smith      William
```

SQRT

構文



用途

SQRT は、*n* の平方根を戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

- *n* が NUMBER になる場合、値 *n* は負にはなりません。SQRT は実数を戻します。
- *n* が浮動小数点数（BINARY_FLOAT または BINARY_DOUBLE）になる場合、結果は次のとおりです。
  - *n* >= 0 の場合、結果は正
  - *n* = -0 の場合、結果は -0
  - *n* < 0 の場合、結果は NaN

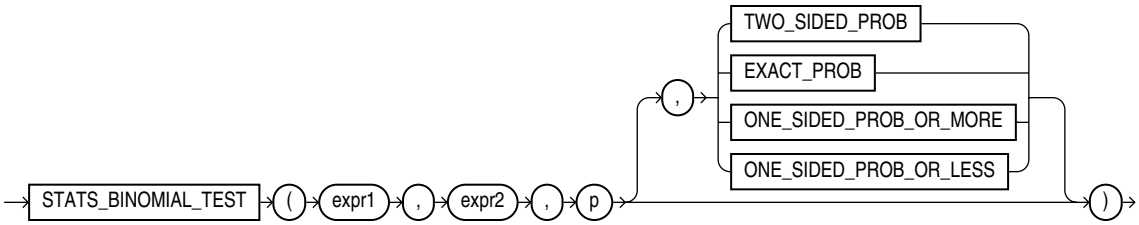
例

次の例では、26 の平方根を戻します。

```
SELECT SQRT(26) "Square root" FROM DUAL;

Square root
-----------
5.09901951
```

STATS_BINOMIAL_TEST

構文



用途

STATS_BINOMIAL_TEST は、有効な値が 2 つのみである二値変数に使用する直接確立法です。このファンクションは、標本の割合と指定された割合との差をテストします。このテストでは通常、小さいサイズの標本が使用されます。

このファンクションは 4 つの引数を取ります。*expr1* はテスト対象の標本、*expr2* は割合を求める値、*p* はテストの基準となる割合です。4 つ目の引数は VARCHAR2 型の戻り値です。4 つ目の引数を指定しない場合、デフォルトで TWO_SIDED_PROB が戻り値になります。表 5-3 に、戻り値の意味を示します。

表 5-3 STATS_BINOMIAL_TEST の戻り値

戻り値	意味
TWO_SIDED_PROB	指定された母集団の割合 *p* が、観測された割合より外側になる確率
EXACT_PROB	指定された母集団の割合 *p* が、観測された割合と正確に同じ値になる確率
ONE_SIDED_PROB_OR_MORE	指定された母集団の割合 *p* が、観測された割合以上になる確率
ONE_SIDED_PROB_OR_LESS	指定された母集団の割合 *p* が、観測された割合以下になる確率

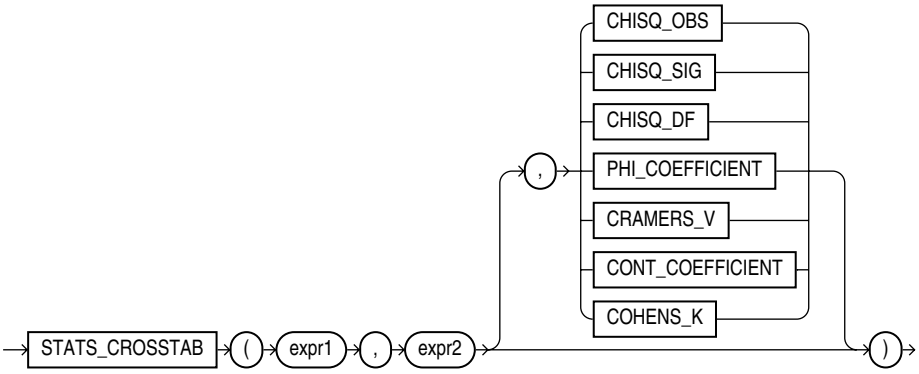
EXACT_PROB では、*p* と一致する割合が戻される確率が戻されます。標本における割合が、50% と大幅に異なるか（有意差があるか）どうかをテストする場合、通常、*p* を 0.50 に設定します。割合が異なるかどうかのみをテストする場合、戻り値に TWO_SIDED_PROB を使用します。割合が *expr2* の値より大きいかどうかをテストする場合、戻り値に ONE_SIDED_PROB_OR_MORE を使用します。割合が *expr2* より小さいかどうかをテストする場合、戻り値に ONE_SIDED_PROB_OR_LESS を使用します。

STATS_BINOMIAL_TEST の例　次の例では、母集団の 69% が男性であるという仮定に基づいて観測された男性数が、実際の男性数と正確に同一となる確率を判断します。

```
SELECT AVG(DECODE(cust_gender, 'M', 1, 0)) real_proportion,
       STATS_BINOMIAL_TEST
         (cust_gender, 'M', 0.68, 'EXACT_PROB') exact,
       STATS_BINOMIAL_TEST
         (cust_gender, 'M', 0.68, 'ONE_SIDED_PROB_OR_LESS') prob_or_less
  FROM sh.customers;
```

STATS_CROSSTAB

構文



用途

クロス集計は、2 つの名義変数の分析に使用する方法です。STATS_CROSSTAB ファンクションは、2 つの式と VARCHAR2 型の戻り値の 3 つの引数を取ります。*expr1* および *expr2* は、分析対象の 2 つの変数です。このファンクションは、3 つ目の引数に従って 1 つの数値を戻します。3 つ目の引数を指定しない場合、デフォルトで CHISQ_SIG が戻り値になります。表 5-4 に、戻り値の意味を示します。

表 5-4 STATS_CROSSTAB の戻り値

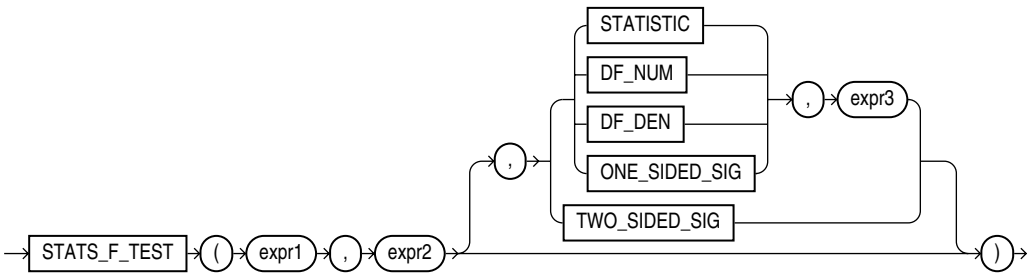
戻り値	意味
CHISQ_OBS	カイ 2 乗の観測値
CHISQ_SIG	カイ 2 乗の観測値の有意性
CHISQ_DF	カイ 2 乗の自由度
PHI_COEFFICIENT	ファイ係数
CRAMERS_V	クラメールの V 統計
CONT_COEFFICIENT	一致係数
COHENS_K	コーエンのカッパ

STATS_CROSSTAB の例　次の例では、性別と収入水準の関連の強さを判断します。

```
SELECT STATS_CROSSTAB
         (cust_gender, cust_income_level, 'CHISQ_OBS') chi_squared,
       STATS_CROSSTAB
         (cust_gender, cust_income_level, 'CHISQ_SIG') p_value,
       STATS_CROSSTAB
         (cust_gender, cust_income_level, 'PHI_COEFFICIENT') phi_coefficient
  FROM sh.customers;

CHI_SQUARED    P_VALUE PHI_COEFFICIENT
----------- ---------- ---------------
 251.690705 1.2364E-47      .067367056
```

STATS_F_TEST

構文



用途

STATS_F_TEST は、2 つの分散に有意差があるかどうかをテストします。*f* の観測値は、他方の分散に対する一方の分散の比率です。この値が 1 と大きく異なる場合は、通常、有意差があることを示しています。

このファンクションは 3 つの引数を取ります。*expr1* はグループ変数または独立変数で、*expr2* は値の標本です。このファンクションは、3 つ目の引数に従って 1 つの数値を戻します。3 つ目の引数を指定しない場合、デフォルトで TWO_SIDED_SIG が戻り値になります。表 5-5 に、戻り値の意味を示します。

表 5-5 STATS_F_TEST の戻り値

戻り値	意味
STATISTIC	*f* の観測値
DF_NUM	分子の自由度
DF_DEN	分母の自由度
ONE_SIDED_SIG	*f* の片側有意
TWO_SIDED_SIG	*f* の両側有意

片側有意は常に上側確率に関連します。最後の引数 *expr3* は、expr1 で指定した 2 つのグループのうち、大きい値または分子（棄却域が上側確率の値）のグループを示します。

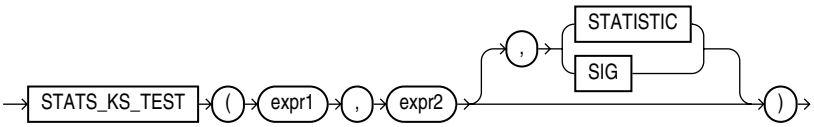
f の観測値は、2 つ目のグループの分散に対する 1 つ目のグループの分散の比率です。*f* の観測値の有意性は、2 つの分散が偶然に異なる確率で、0（ゼロ）～ 1 の数値です。この有意性の値が小さい場合、2 つの分散に有意差があることを示しています。各分散の自由度は、標本における観測数から 1 を引いた値です。

STATS_F_TEST の例　次の例では、男性と女性のクレジット利用限度額に有意差があるかどうかを判断します。p_value が 0（ゼロ）に近くなく、f_statistic が 1 に近いという結果は、男性と女性のクレジット利用限度額の差が有意でないことを示しています。

```
SELECT VARIANCE(DECODE(cust_gender, 'M', cust_credit_limit, null)) var_men,
       VARIANCE(DECODE(cust_gender, 'F', cust_credit_limit, null)) var_women,
       STATS_F_TEST(cust_gender, cust_credit_limit, 'STATISTIC', 'F') f_statistic,
       STATS_F_TEST(cust_gender, cust_credit_limit) two_sided_p_value
  FROM sh.customers;

   VAR_MEN  VAR_WOMEN F_STATISTIC TWO_SIDED_P_VALUE
---------- ---------- ----------- -----------------
12879896.7   13046865  1.01296348        .311928071
```

STATS_KS_TEST

構文



用途

STATS_KS_TEST は、2 つの標本を比較して、それらが同じ母集団に属しているか、または同じ分布を持つ母集団に属しているかをテストする Kolmogorov-Smirnov ファンクションです。このファンクションでは、標本の母集団が正規分布に従うとは仮定されません。

このファンクションは、2 つの式と VARCHAR2 型の戻り値の 3 つの引数を取ります。*expr1* には、データを 2 つのサンプルに分類する値を指定します。*expr2* には、各サンプルの値を指定します。*expr1* に、行を 1 つのみか 3 つ以上のサンプルに分類する値を指定すると、エラーが発生します。このファンクションは、3 つ目の引数に従って 1 つの値を戻します。3 つ目の引数を指定しない場合、デフォルトで SIG が戻り値になります。表 5-6 に、戻り値の意味を示します。

表 5-6 STATS_KS_TEST の戻り値

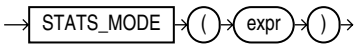
戻り値	意味
STATISTIC	*D* の観測値
SIG	*D* の有意性

STATS_KS_TEST の例 次の例では、Kolmogorov-Smirnov 検定を使用して、男性と女性に対する売上の分布が偶然であるかどうかを判断します。

```
SELECT stats_ks_test(cust_gender, amount_sold, 'STATISTIC') ks_statistic,
       stats_ks_test(cust_gender, amount_sold) p_value
  FROM sh.customers c, sh.sales s
  WHERE c.cust_id = s.cust_id;

KS_STATISTIC    P_VALUE
------------ ----------
  .003841396 .004080006
```

STATS_MODE

構文



用途

STATS_MODE は、引数として値の集合を取り、最も出現頻度の高い値を戻します。複数の最頻値が存在する場合、Oracle Database は 1 つの最頻値を選択し、その値のみを戻します。

複数の最頻値（存在する場合）を取得する場合は、次の不確定な問合せに示すとおり、他のファンクションの組合せを使用する必要があります。

```
SELECT x FROM (SELECT x, COUNT(x) AS cnt1
   FROM t GROUP BY x)
   WHERE cnt1 =
      (SELECT MAX(cnt2) FROM (SELECT COUNT(x) AS cnt2 FROM t GROUP BY x));
```

例

次の例では、hr.employees 表の部門ごとの給与の最頻値を戻します。

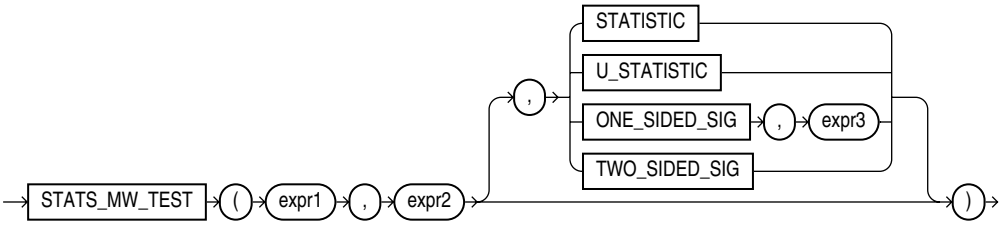
```
SELECT department_id, STATS_MODE(salary) FROM employees
   GROUP BY department_id
   ORDER BY department_id, stats_mode(salary);

DEPARTMENT_ID STATS_MODE(SALARY)
------------- ------------------
           10               4400
           20               6000
           30               2500
           40               6500
           50               2500
           60               4800
           70              10000
           80               9500
           90              17000
          100               6900
          110               8300
                            7000
```

複数の最頻値が存在し、そのすべての最頻値を取得する必要がある場合は、次の例に示すとおり、他のファンクションの組合せを使用します。

```
SELECT commission_pct FROM
   (SELECT commission_pct, COUNT(commission_pct) AS cnt1 FROM employees
      GROUP BY commission_pct)
   WHERE cnt1 =
      (SELECT MAX (cnt2) FROM
         (SELECT COUNT(commission_pct) AS cnt2
         FROM employees GROUP BY commission_pct))
   ORDER BY commission_pct;

COMMISSION_PCT
--------------
            .2
            .3
```

STATS_MW_TEST

構文



用途

Mann-Whitney 検定では、2 つの独立した標本を比較して、2 つの母集団が同じ分布ファンクションを持つという帰無仮説を、2 つの分布ファンクションは異なるという対立仮説に対してテストします。

STATS_MW_TEST では、STATS_T_TEST_* ファンクションとは異なり、標本間の差が正規分布するとは仮定されません。このファンクションは、3 つの引数と VARCHAR2 型の戻り値を取ります。*expr1* には、データをグループに分類する値を指定します。*expr2* には、各グループの値を指定します。このファンクションは、3 つ目の引数に従って 1 つの値を戻します。3 つ目の引数を指定しない場合、デフォルトで TWO_SIDED_SIG が戻り値になります。表 7-7 に、戻り値の意味を示します。

Z または *U* の観測値の有意性は、2 つの分散が偶然に異なる確率で、0（ゼロ）～ 1 の数値です。この有意性の値が小さい場合、2 つの分散に有意差があることを示しています。各分散の自由度は、標本における観測数から 1 を引いた値です。

表 5-7 STATS_MW_TEST の戻り値

戻り値	意味
STATISTIC	*Z* の観測値
U_STATISTIC	*U* の観測値
ONE_SIDED_SIG	*Z* の片側有意
TWO_SIDED_SIG	*Z* の両側有意

片側有意は常に上側確率に関連します。最後の引数 *expr3* は、expr1 で指定した 2 つのグループのうち、大きい値（棄却域が上側確率の値）のグループを示します。

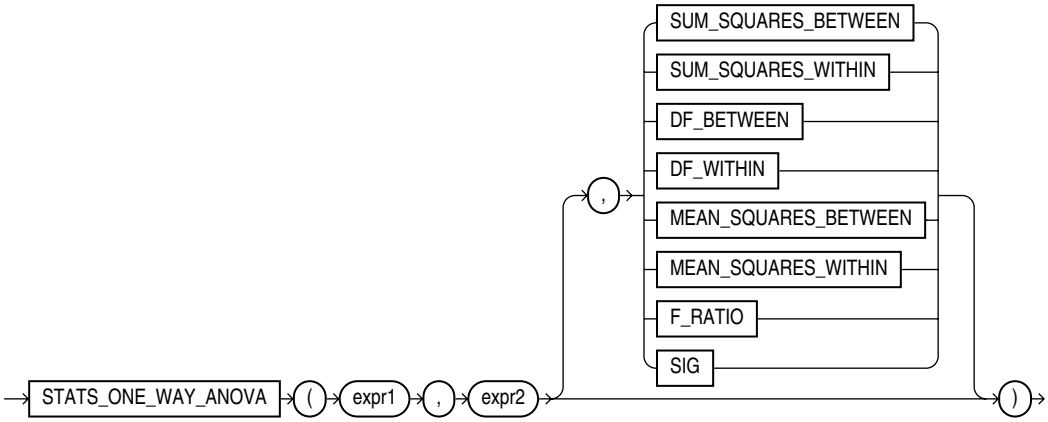
STATS_MW_TEST は、値の順位の合計における差を確認して、標本が同じ分布からのものである確率を計算します。標本が同じ分布に属している場合、各標本の合計値は近くなります。

STATS_MW_TEST の例　次の例では、Mann-Whitney 検定を使用して、男性と女性に対する売上の分布が偶然であるかどうかを判断します。

```
SELECT STATS_MW_TEST
         (cust_gender, amount_sold, 'STATISTIC') z_statistic,
       STATS_MW_TEST
         (cust_gender, amount_sold, 'ONE_SIDED_SIG', 'F') one_sided_p_value
  FROM sh.customers c, sh.sales s
  WHERE c.cust_id = s.cust_id;

Z_STATISTIC ONE_SIDED_P_VALUE
----------- -----------------
 -1.4011509        .080584471
```

STATS_ONE_WAY_ANOVA

構文



用途

STATS_ONE_WAY_ANOVA ファンクションによる一元配置分散分析では、分散の 2 つの異なる推定値を比較して、（複数のグループまたは変数の）平均値の差をテストして統計学的有意差を求めます。1 つ目の推定値は、各グループまたはカテゴリ内の分散に基づきます。これは、**群内平均平方**または**平均平方誤差**と呼ばれます。2 つ目の推定値は、グループの平均値の分散に基づきます。これは、**群間平均平方**と呼ばれます。グループの平均値に有意差がある場合、その群間平均平方は期待値より大きくなり、群内平均平方に一致しません。グループの平均平方が一貫している場合、2 つの分散の推定値はほぼ同じになります。

STATS_ONE_WAY_ANOVA は、2 つの式と VARCHAR2 型の戻り値の 3 つの引数を取ります。*expr1* には、データをグループの集合に分割する独立変数またはグループ変数を指定します。*expr2* には、グループの各構成要素に対応する値を含む従属変数（数式）を指定します。このファンクションは、3 つ目の引数に従って 1 つの数値を戻します。3 つ目の引数を指定しない場合、デフォルトで SIG が戻り値になります。表 5-8 に、戻り値の意味を示します。

表 5-8 STATS_ONE_WAY_ANOVA の戻り値

戻り値	意味
SUM_SQUARES_BETEEN	グループ間平方和
SUM_SQUARES_WITHIN	グループ内平方和
DF_BETWEEN	グループ間の自由度
DF_WITHIN	グループ内の自由度
MEAN_SQUARES_BETWEEN	グループ間平均平方
MEAN_SQUARES_WITHIN	グループ内平均平方
F_RATIO	群内平方和に対する群間平方和の比率（MSB（グループ間平均平方）/MSW（グループ内平均平方））
SIG	有意性

一元配置分散分析の有意性は群間平均平方と群内平均平方の比率に対する *f* 検定の片側有意を求めることによって判断されます。*f* 検定では片側有意を使用する必要があります。これは、群間平均平方は、群内平均平方以上のみになるためです。そのため、STATS_ONE_WAY_ANOVA が戻す有意性は、グループ間の差が偶然である確率で、0（ゼロ）～ 1 の数値です。この数値が小さいほど、グループ間の有意差が大きくなります。*f* 検定の実行の詳細は、5-169 ページの「STATS_F_TEST」を参照してください。

STATS_ONE_WAY_ANOVA の例　次の例では、収入水準内での売上平均の差と、収入水準間での売上平均の差の有意性を判断します。p_values が 0（ゼロ）に近いという結果は、男性と女性の両方に対して、様々な収入水準の人々に販売された商品の金額には有意差があることを示しています。

```
SELECT cust_gender,
       STATS_ONE_WAY_ANOVA(cust_income_level, amount_sold, 'F_RATIO') f_ratio,
       STATS_ONE_WAY_ANOVA(cust_income_level, amount_sold, 'SIG') p_value
  FROM sh.customers c, sh.sales s
  WHERE c.cust_id = s.cust_id
  GROUP BY cust_gender
  ORDER BY cust_gender;

C    F_RATIO    P_VALUE
- ---------- ----------
F 5.59536943 4.7840E-09
M  9.2865001 6.7139E-17
```

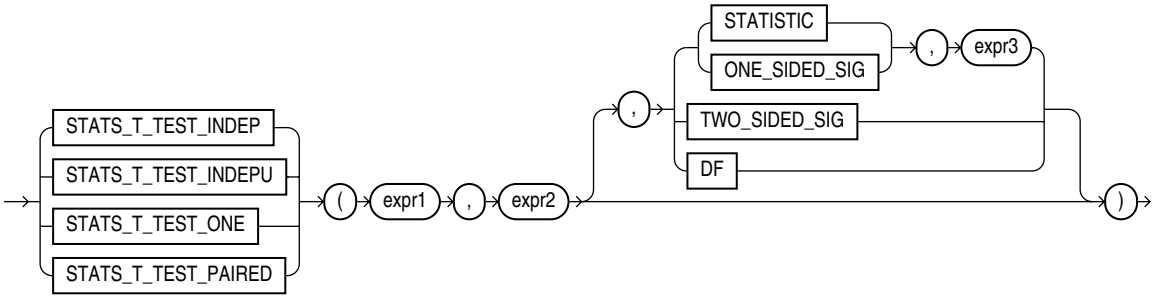
STATS_T_TEST_*

t 検定ファンクションを次に示します。

- STATS_T_TEST_ONE: 1 標本 *t* 検定
- STATS_T_TEST_PAIRED: 対応のある 2 標本 *t* 検定（クロス *t* 検定とも呼ばれる）
- STATS_T_TEST_INDEP: 同じ分散（併合分散）を持つ独立した 2 つのグループの *t* 検定
- STATS_T_TEST_INDEPU: 異なる分散（非併合分散）を持つ独立した 2 つのグループの *t* 検定

構文

***stats_t_test*::=**



用途

t 検定では、平均値の差の有意性を測定します。この検定を使用して、2 つのグループの平均値の比較、または 1 つのグループの平均値と定数の比較を行えます。1 標本および 2 標本 STATS_T_TEST_* ファンクションは、2 つの式と VARCHAR2 型の戻り値の 3 つの引数を取ります。このファンクションは、3 つ目の引数に従って 1 つの数値を戻します。3 つ目の引数を指定しない場合、デフォルトで TWO_SIDED_SIG が戻り値になります。表 5-9 に、戻り値の意味を示します。

表 5-9 STATS_T_TEST_* の戻り値

戻り値	意味
STATISTIC	*t* の観測値
DF	自由度
ONE_SIDED_SIG	*t* の片側有意
TWO_SIDED_SIG	*t* の両側有意

2 つの独立した STATS_T_TEST_* ファンクションは、3 つ目の引数が STATISTIC または ONE_SIDED_SIG として指定されている場合、4 つ目の引数（*expr3*）を取ることができます。この場合、*expr3* は、*expr1* のどちらの値が大きい値（棄却域が上側確率の値）であるかを示します。

t の観測値の有意性は、*t* の値が偶然得られた確率で、0（ゼロ）～ 1 の数値です。この値が小さいほど、平均値間の有意差が大きくなります。片側有意は常に上側確率に関連します。1 標本および対応のある *t*-test の場合、最初の式が大きい値になります。独立した *t*-test の場合、*expr3* で指定した値が大きい値になります。

自由度は、*t* の観測値を求めるために使用した *t* 検定の種類によって異なります。たとえば、1 標本 *t* 検定（STATS_T_TEST_ONE）の場合、自由度は標本における観測数から 1 を引いた値です。

STATS_T_TEST_ONE

STATS_T_TEST_ONE ファンクションでは、*expr1* には標本を指定し、*expr2* には標本平均値の比較対象となる定数平均値を指定します。この *t*-test の場合にのみ、*expr2* はオプションとなります。定数平均値は、デフォルトで 0 になります。このファンクションは、標本平均値と既知の平均値の差を、平均値の標準誤差で割って *t* 値を求めます（STATS_T_TEST_PAIRED では、平均値の差の標準誤差で割ります）。

STATS_T_TEST_ONE の例　次の例では、平均表示価格と定数値 60 の差の有意性を判断します。

```
SELECT AVG(prod_list_price) group_mean,
       STATS_T_TEST_ONE(prod_list_price, 60, 'STATISTIC') t_observed,
       STATS_T_TEST_ONE(prod_list_price, 60) two_sided_p_value
  FROM sh.products;

GROUP_MEAN T_OBSERVED TWO_SIDED_P_VALUE
---------- ---------- -----------------
139.545556 2.32107746        .023158537
```

STATS_T_TEST_PAIRED

STATS_T_TEST_PAIRED ファンクションでは、*expr1* および *expr2* には、比較する平均値の計算元の 2 つの標本をそれぞれ指定します。このファンクションは、標本平均値の差を、平均値の差の標準誤差で割って *t* 値を求めます（STATS_T_TEST_ONE では、平均値の標準誤差で割ります）。

STATS_T_TEST_INDEP および STATS_T_TEST_INDEPU

STATS_T_TEST_INDEP および STATS_T_TEST_INDEPU ファンクションでは、*expr1* はグループ列で、*expr2* は値の標本です。併合分散用のファンクション（STATS_T_TEST_INDEP）は、同様の分散を持つ 2 つの分布において、平均値が同じか異なるかをテストします。非併合分散用のファンクション（STATS_T_TEST_INDEPU）は、既知の有意差のある分散を持つ 2 つの分布においても、平均値が同じか異なるかをテストします。

これらのファンクションを使用する前に、標本の分散に有意差があるかどうかを判断しておくことをお薦めします。有意差がある場合、そのデータは異なる形状の分布に属している可能性があり、平均値の差が有効でない場合があります。*f* 検定を実行して、分散の差を判断できます。分散に有意差がない場合、STATS_T_TEST_INDEP を使用します。有意差がある場合は、STATS_T_TEST_INDEPU を使用します。*f* 検定の実行の詳細は、5-169 ページの「STATS_F_TEST」を参照してください。

STATS_T_TEST_INDEP の例　次の例では、各分布の分散が同様である（併合分散）として、男性と女性に対する平均売上間の差の有意性を判断します。

```
SELECT SUBSTR(cust_income_level, 1, 22) income_level,
     AVG(DECODE(cust_gender, 'M', amount_sold, null)) sold_to_men,
     AVG(DECODE(cust_gender, 'F', amount_sold, null)) sold_to_women,
     STATS_T_TEST_INDEP(cust_gender, amount_sold, 'STATISTIC', 'F') t_observed,
     STATS_T_TEST_INDEP(cust_gender, amount_sold) two_sided_p_value
  FROM sh.customers c, sh.sales s
  WHERE c.cust_id = s.cust_id
  GROUP BY ROLLUP(cust_income_level)
  ORDER BY income_level, sold_to_men, sold_to_women, t_observed;

INCOME_LEVEL           SOLD_TO_MEN SOLD_TO_WOMEN T_OBSERVED TWO_SIDED_P_VALUE
---------------------- ----------- ------------- ---------- -----------------
A: Below 30,000          105.28349    99.4281447 -1.9880629        .046811482
B: 30,000 - 49,999       102.59651    109.829642 3.04330875        .002341053
C: 50,000 - 69,999      105.627588    110.127931 2.36148671        .018204221
D: 70,000 - 89,999      106.630299     110.47287 2.28496443        .022316997
E: 90,000 - 109,999     103.396741    101.610416 -1.2544577        .209677823
F: 110,000 - 129,999     106.76476    105.981312 -.60444998        .545545304
G: 130,000 - 149,999    108.877532     107.31377 -.85298245        .393671218
H: 150,000 - 169,999    110.987258    107.152191 -1.9062363        .056622983
I: 170,000 - 189,999    102.808238     107.43556 2.18477851        .028908566
J: 190,000 - 249,999    108.040564    115.343356 2.58313425        .009794516
```

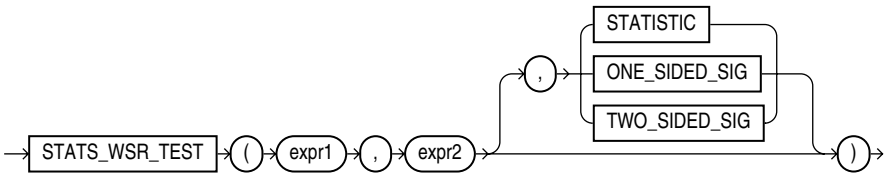
```
K: 250,000 - 299,999    112.377993    108.196097 -1.4107871        .158316973
L: 300,000 and above    120.970235    112.216342 -2.0642868        .039003862
                        107.121845     113.80441 .686144393        .492670059
                        106.663769    107.276386 1.08013499        .280082357
14 rows selected.
```

STATS_T_TEST_INDEPU の例　次の例では、各分布の分散に既知の有意差がある（非併合分散）として、男性と女性に対する平均売上間の差の有意性を判断します。

```
SELECT SUBSTR(cust_income_level, 1, 22) income_level,
       AVG(DECODE(cust_gender, 'M', amount_sold, null)) sold_to_men,
       AVG(DECODE(cust_gender, 'F', amount_sold, null)) sold_to_women,
       STATS_T_TEST_INDEPU(cust_gender, amount_sold, 'STATISTIC', 'F') t_observed,
       STATS_T_TEST_INDEPU(cust_gender, amount_sold) two_sided_p_value
  FROM sh.customers c, sh.sales s
  WHERE c.cust_id = s.cust_id
  GROUP BY ROLLUP(cust_income_level)
  ORDER BY income_level, sold_to_men, sold_to_women, t_observed;

INCOME_LEVEL           SOLD_TO_MEN SOLD_TO_WOMEN T_OBSERVED TWO_SIDED_P_VALUE
---------------------- ----------- ------------- ---------- -----------------
A: Below 30,000          105.28349    99.4281447 -2.0542592        .039964704
B: 30,000 - 49,999       102.59651    109.829642 2.96922332        .002987742
C: 50,000 - 69,999      105.627588    110.127931  2.3496854        .018792277
D: 70,000 - 89,999      106.630299     110.47287 2.26839281        .023307831
E: 90,000 - 109,999     103.396741    101.610416 -1.2603509        .207545662
F: 110,000 - 129,999     106.76476    105.981312 -.60580011        .544648553
G: 130,000 - 149,999    108.877532     107.31377 -.85219781        .394107755
H: 150,000 - 169,999    110.987258    107.152191 -1.9451486        .051762624
I: 170,000 - 189,999    102.808238     107.43556 2.14966921        .031587875
J: 190,000 - 249,999    108.040564    115.343356 2.54749867        .010854966
K: 250,000 - 299,999    112.377993    108.196097 -1.4115514        .158091676
L: 300,000 and above    120.970235    112.216342 -2.0726194        .038225611
                        107.121845     113.80441 .689462437        .490595765
                        106.663769    107.276386 1.07853782        .280794207
14 rows selected.
```

STATS_WSR_TEST

構文



用途

STATS_WSR_TEST は、対応のある標本のウィルコクソンの符号順位検定であり、標本間の差の中央値と 0（ゼロ）に有意差があるかどうかを判断します。差の絶対値は、標本を順序付けし、順位に符号を付けて求められます。また、帰無仮説によって、正の差の順位の合計が、負の差の順位の合計に等しいと仮定されます。

このファンクションは 3 つの引数を取ります。*expr1* および *expr2* は分析対象の 2 つの標本で、3 つ目の引数は VARCHAR2 型の戻り値です。3 つ目の引数を指定しない場合、デフォルトで TWO_SIDED_SIG が戻り値になります。表 5-10 に、戻り値の意味を示します。

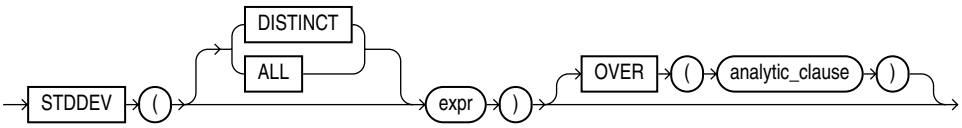
表 5-10 STATS_WSR_TEST_* の戻り値

戻り値	意味
STATISTIC	Z の観測値
ONE_SIDED_SIG	Z の片側有意
TWO_SIDED_SIG	Z の両側有意

片側有意は常に上側確率に関連します。*expr1* が大きい値（棄却域が上側確率の値）になります。

STDDEV

構文



参照： 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

STDDEV は、数値の集合である *expr* の標本標準偏差を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。このファンクションは、STDDEV_SAMP が NULL を戻すことに対して、入力データが 1 行のみの場合に STDDEV が 0（ゼロ）を戻すという点で、STDDEV_SAMP と異なります。

Oracle Database は、VARIANCE 集計ファンクションに対して定義された分散の平方根として標準偏差を計算します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を戻します。

参照： 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

DISTINCT を指定する場合は、*analytic_clause* の *query_partition_clause* のみ指定できます。*order_by_clause* および *windowing_clause* は指定できません。

参照：

- 5-8 ページの「集計ファンクション」、5-225 ページの「VARIANCE」および 5-180 ページの「STDDEV_SAMP」を参照してください。
- *expr* の書式の詳細は、6-2 ページの「SQL 式」を参照してください。

集計の例

次の例では、サンプル表 hr.employees の給与値の標本標準偏差を戻します。

```
SELECT STDDEV(salary) "Deviation"
   FROM employees;

 Deviation
----------
3909.36575
```

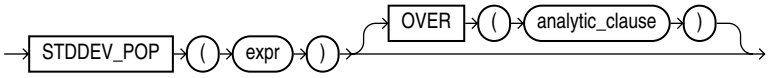
分析の例

次の例では、hire_date で順序付けられたサンプル表 hr.employees の部門 80 の給与の値の累積標準偏差を戻します。

```
SELECT last_name, salary,
   STDDEV(salary) OVER (ORDER BY hire_date) "StdDev"
   FROM employees
   WHERE department_id = 30
   ORDER BY last_name, salary, "StdDev";

LAST_NAME                     SALARY     StdDev
------------------------- ---------- ----------
Baida                           2900 4035.26125
Colmenares                      2500 3362.58829
Himuro                          2600  3649.2465
Khoo                            3100 5586.14357
Raphaely                       11000          0
Tobias                          2800  4650.0896
```

STDDEV_POP

構文



> **参照：** 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

STDDEV_POP は母集団標準偏差を計算し、母集団分散の平方根を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

このファンクションは、VAR_POP ファンクションの平方根と同じです。VAR_POP が NULL を戻す場合、このファンクションも NULL を戻します。

> **参照：**
>
> - 5-8 ページの「集計ファンクション」および 5-223 ページの「VAR_POP」を参照してください。
> - *expr* の書式の詳細は、6-2 ページの「SQL 式」を参照してください。

集計の例

次の例では、サンプル表 sh.sales にある売上高の母集団標準偏差および標本標準偏差を戻します。

```
SELECT STDDEV_POP(amount_sold) "Pop",
   STDDEV_SAMP(amount_sold) "Samp"
   FROM sales;
```

```
       Pop       Samp
---------- ----------
896.355151 896.355592
```

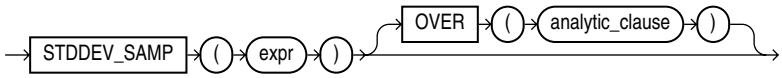
分析の例

次の例では、サンプル表 hr.employees の部門ごとの給与の母集団標準偏差を戻します。

```
SELECT department_id, last_name, salary,
   STDDEV_POP(salary) OVER (PARTITION BY department_id) AS pop_std
   FROM employees
   ORDER BY department_id, last_name, salary, pop_std;

DEPARTMENT_ID LAST_NAME                 SALARY    POP_STD
------------- ------------------------- ---------- ----------
           10 Whalen                          4400          0
           20 Fay                             6000       3500
           20 Hartstein                      13000       3500
           30 Baida                           2900  3069.6091
. . .
          100 Urman                           7800 1644.18166
          110 Gietz                           8300       1850
          110 Higgins                        12000       1850
              Grant                           7000          0
```

STDDEV_SAMP

構文



参照： 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

STDDEV_SAMP は標本累積標準偏差を計算し、標本分散の平方根を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を戻します。

参照： 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

このファンクションは、VAR_SAMP ファンクションの平方根と同じです。VAR_SAMP が NULL を戻す場合、このファンクションも NULL を戻します。

参照：

- 5-8 ページの「集計ファンクション」および 5-224 ページの「VAR_SAMP」を参照してください。
- *expr* の書式の詳細は、6-2 ページの「SQL 式」を参照してください。

集計の例

5-179 ページの「STDDEV_POP」の集計の例を参照してください。

分析の例

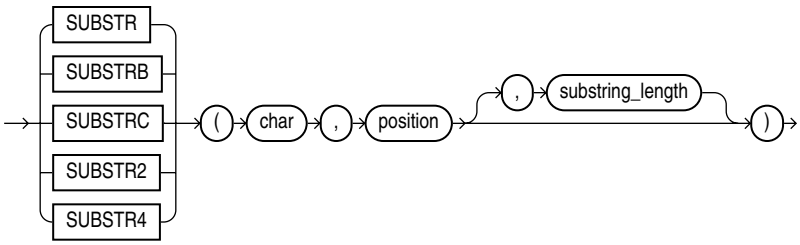
次の例では、employees 表の部門ごとの給与の標本標準偏差を戻します。

```
SELECT department_id, last_name, hire_date, salary,
   STDDEV_SAMP(salary) OVER (PARTITION BY department_id
      ORDER BY hire_date
      ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS cum_sdev
   FROM employees
   ORDER BY department_id, last_name, hire_date, salary, cum_sdev;

DEPARTMENT_ID LAST_NAME       HIRE_DATE     SALARY   CUM_SDEV
------------- --------------- --------- ---------- ----------
           10 Whalen                    17-SEP-87       4400
           20 Fay                       17-AUG-97       6000 4949.74747
           20 Hartstein                 17-FEB-96      13000
           30 Baida                     24-DEC-97       2900 4035.26125
           30 Colmenares                10-AUG-99       2500 3362.58829
           30 Himuro                    15-NOV-98       2600  3649.2465
           30 Khoo                      18-MAY-95       3100 5586.14357
           30 Raphaely                  07-DEC-94      11000
. . .
          100 Greenberg                 17-AUG-94      12000 2121.32034
          100 Popp                      07-DEC-99       6900 1801.11077
          100 Sciarra                   30-SEP-97       7700 1925.91969
          100 Urman                     07-MAR-98       7800 1785.49713
          110 Gietz                     07-JUN-94       8300 2616.29509
          110 Higgins                   07-JUN-94      12000
              Grant                     24-MAY-99       7000
```

SUBSTR

構文

***substr*::=**



用途

SUBSTR の各ファンクションは、*char* の *position* の文字から *substring_length* 文字分の文字列を抜き出して戻します。SUBSTR は、入力キャラクタ・セットによって定義された文字を使用して、長さを計算します。SUBSTRB は、文字のかわりにバイトを使用します。SUBSTRC は、完全な Unicode キャラクタを使用します。SUBSTR2 は、UCS2 コードポイントを使用します。SUBSTR4 は、UCS4 コードポイントを使用します。

- *position* が 0 の場合、1 として処理されます。
- *position* が正の場合、Oracle Database は *char* の始めから数えて最初の文字を検索します。
- *position* が負の場合、Oracle は *char* の終わりから逆方向に数えます。
- *substring_length* を指定しないと、Oracle は *char* の終わりまでのすべての文字を戻します。*substring_length* が 1 より小さい場合、NULL を戻します。

char は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。*position* および *substring_length* は、NUMBER データ型か、または暗黙的に NUMBER に変換可能な任意のデータ型で、整数である必要があります。戻り値は、*char* と同じデータ型です。引数として SUBSTR に渡された浮動小数点数は、自動的に整数に変換されます。

> **参照：** 異なるロケールでの SUBSTR ファンクションおよび長さセマンティクスの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

例

次の例では、「ABCDEFG」の指定されたサブストリングを戻します。

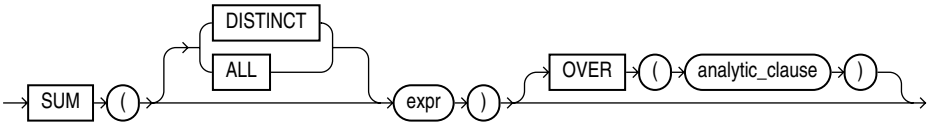
```
SELECT SUBSTR('ABCDEFG',3,4) "Substring"
     FROM DUAL;

Substring
---------
CDEF

SELECT SUBSTR('ABCDEFG',-5,4) "Substring"
     FROM DUAL;

Substring
---------
CDEF
```

データベース・キャラクタ・セットがダブルバイトの場合を想定します。

```
SELECT SUBSTRB('ABCDEFG',5,4.2) "Substring with bytes"
     FROM DUAL;

Substring with bytes
--------------------
CD
```

SUM

構文



> **参照：** 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

SUM は、*expr* の値の合計を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

DISTINCT を指定する場合は、*analytic_clause* の *query_partition_clause* のみ指定できます。*order_by_clause* および *windowing_clause* は指定できません。

> **参照：** *expr* の書式の詳細は、6-2 ページの「SQL 式」を参照してください。5-8 ページの「集計ファンクション」も参照してください。

集計の例

次の例では、サンプル表 hr.employees にあるすべての給与の合計を計算します。

```
SELECT SUM(salary) "Total"
     FROM employees;

     Total
----------
    691400
```

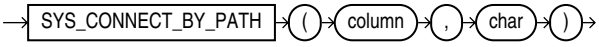
分析の例

次の例では、サンプル表 hr.employees の各マネージャについて、そのマネージャの下で働く従業員の現在の給与以下の給与の累積合計を計算します。Raphaely および Cambrault が同じ累積を持っています。これは、Raphaely および Cambrault が同じ給与であるため、Oracle Database が給与の値を同時に追加し、同じ累積合計を両方の行に対して適用したためです。

```
SELECT manager_id, last_name, salary,
   SUM(salary) OVER (PARTITION BY manager_id ORDER BY salary
   RANGE UNBOUNDED PRECEDING) l_csum
   FROM employees
   ORDER BY manager_id, last_name, salary, l_csum;

MANAGER_ID LAST_NAME           SALARY     L_CSUM
---------- --------------- ---------- ----------
MANAGER_ID LAST_NAME                    SALARY     L_CSUM
---------- ------------------------ ---------- ----------
       100 Cambrault                     11000      68900
       100 De Haan                       17000     155400
       100 Errazuriz                     12000      80900
       100 Fripp                          8200      36400
       100 Hartstein                     13000      93900
       100 Kaufling                       7900      20200
       100 Kochhar                       17000     155400
       100 Mourgos                        5800       5800
       100 Partners                      13500     107400
       100 Raphaely                      11000      68900
       100 Russell                       14000     121400
. . .
       149 Hutton                         8800      39000
       149 Johnson                        6200       6200
       149 Livingston                     8400      21600
       149 Taylor                         8600      30200
       201 Fay                            6000       6000
       205 Gietz                          8300       8300
           King                          24000      24000
```

SYS_CONNECT_BY_PATH

構文



用途

SYS_CONNECT_BY_PATH は、階層問合せのみで有効です。このファンクションは、ルートからノードへの列の値のパスを、CONNECT BY 条件によって戻された各行を *char* で区切った列の値とともに戻します。

column および *char* は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型です。戻される文字列は、VARCHAR2 データ型であり、*column* と同じキャラクタ・セットです。

> **参照：** 階層問合せおよび CONNECT BY 条件の詳細は、9-3 ページの「階層問合せ」を参照してください。

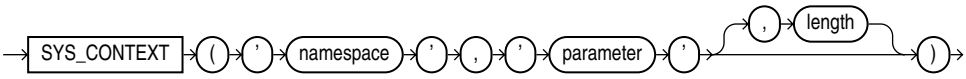
例

次の例では、従業員 Kochhar から Kochhar のすべての従業員（およびそれらの従業員の従業員）への従業員名のパスを戻します。

```
SELECT LPAD(' ', 2*level-1)||SYS_CONNECT_BY_PATH(last_name, '/') "Path"
   FROM employees
   START WITH last_name = 'Kochhar'
   CONNECT BY PRIOR employee_id = manager_id;

Path
------------------------------
     /Kochhar/Greenberg/Chen
     /Kochhar/Greenberg/Faviet
     /Kochhar/Greenberg/Popp
     /Kochhar/Greenberg/Sciarra
     /Kochhar/Greenberg/Urman
     /Kochhar/Higgins/Gietz
   /Kochhar/Baer
   /Kochhar/Greenberg
   /Kochhar/Higgins
   /Kochhar/Mavris
   /Kochhar/Whalen
 /Kochhar
```

SYS_CONTEXT

構文



用途

SYS_CONTEXT は、コンテキスト *namespace* に関連付けられた *parameter* の値を戻します。このファンクションは、SQL 文および PL/SQL 文で使用できます。

namespace および *parameter* には、文字列、またはネームスペースまたは属性を指定する文字列に変換する式を指定できます。コンテキスト *namespace* はすでに作成されている必要があり、関連付けられた *parameter* およびその値は、DBMS_SESSION.set_context プロシージャを使用して設定されている必要があります。*namespace* は有効な SQL 識別子である必要があります。*parameter* 名には、すべての文字列を指定できます。大 / 小文字を区別しますが、長さは 30 バイト以下です。

戻り値のデータ型は VARCHAR2 です。戻り値のデフォルトの最大サイズは、256 バイトです。オプションの *length* パラメータを指定して、このデフォルトを上書きできます。このパラメータは、NUMBER か、または暗黙的に NUMBER に変換可能な値である必要があります。値の有効範囲は 1 ～ 4000 バイトです。無効な値を指定すると、Oracle Database はその値を無視してデフォルト値を使用します。

Oracle では、現行のセッションを記述する USERENV という組込みネームスペースを提供しています。ネームスペース USERENV の事前定義パラメータについては、5-186 ページの表 5-11 を参照してください。

参照：

- アプリケーション開発でのアプリケーション・コンテキスト機能の使用方法については、『Oracle Database セキュリティ・ガイド』を参照してください。
- ユーザー定義のコンテキスト・ネームスペースの作成方法については、14-8 ページの「CREATE CONTEXT」を参照してください。
- DBMS_SESSION.set_context プロシージャについては、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

例

次の例では、データベースにログインしたユーザー名を戻します。

```
CONNECT OE/password
SELECT SYS_CONTEXT ('USERENV', 'SESSION_USER')
   FROM DUAL;

SYS_CONTEXT ('USERENV', 'SESSION_USER')
----------------------------------------------------
OE
```

注意： この機能の説明を簡単にするために、この例ではデプロイされたシステムで通常使用されるパスワード管理技術を実行しません。本番環境では、Oracle Database のパスワード管理ガイドラインに従って、サンプル・アカウントを無効にします。パスワード管理ガイドラインおよびその他のセキュリティ推奨事項については、『Oracle Database セキュリティ・ガイド』を参照してください。

次の例では、hr_apps の作成時に、コンテキスト hr_apps に関連付けられた PL/SQL パッケージの属性 group_no に対する値として設定されたグループ番号を戻します。

```
SELECT SYS_CONTEXT ('hr_apps', 'group_no') "User Group"
   FROM DUAL;
```

表 5-11 ネームスペース USERENV の事前定義パラメータ

パラメータ	戻り値
ACTION	モジュール（アプリケーション名）内の位置を識別します。 DBMS_APPLICATION_INFO パッケージまたは OCI を使用して設定します。
AUDITED_CURSORID	監査をトリガーによって実行した SQL のカーソル ID を戻します。このパラメータは、ファイングレイン監査環境では無効です。このパラメータをファイングレイン監査環境で指定すると、Oracle Database は常に NULL を戻します。
AUTHENTICATED_ IDENTITY	認証で使用する識別を戻します。次のリストでは、ユーザーのタイプの後に、戻される値を示します。 ■ Kerberos 認証済のエンタープライズ・ユーザー : Kerberos プリンシパル名 ■ Kerberos 認証済の外部ユーザー : Kerberos プリンシパル名（スキーマ名と同一） ■ SSL 認証済のエンタープライズ・ユーザー : ユーザーの PKI 証明書の DN ■ SSL 認証済の外部ユーザー : ユーザーの PKI 証明書の DN ■ パスワード認証済のエンタープライズ・ユーザー : ニックネーム（ログイン名と同一） ■ パスワード認証済のデータベース・ユーザー : データベースのユーザー名（スキーマ名と同一） ■ OS 認証済の外部ユーザー : 外部オペレーティング・システムのユーザー名 ■ RADIUS/DCE 認証済の外部ユーザー : スキーマ名 ■ DN 付きプロキシ : クライアントの Oracle Internet Directory DN ■ 証明書付きプロキシ : クライアントの証明書 DN ■ ユーザー名付きプロキシ : クライアントがローカル・データベースのユーザーの場合はデータベース・ユーザー名、クライアントがエンタープライズ・ユーザーの場合はニックネーム ■ パスワード・ファイルを使用する SYSDBA/SYSOPER: ログイン名 ■ OS 認証を使用する SYSDBA/SYSOPER: オペレーティング・システムのユーザー名
AUTHENTICATION_DATA	ログイン・ユーザーの認証に使用されるデータを戻します。X.503 認証セッションでは、このフィールドは HEX2 形式での認証のコンテキストを戻します。 **注意 :** 構文の *length* パラメータを使用して、AUTHENTICATION_DATA 属性の戻り値を変更できます。最大 4000 までの値を指定できます。この属性は、Oracle Database がこのような変更を実行する USERENV の唯一の属性です。
AUTHENTICATION_METHOD	認証方式を戻します。次のリストでは、ユーザーのタイプの後に、戻される方法を示します。 ■ パスワード認証済のエンタープライズ・ユーザー、ローカル・データベースのユーザー、またはパスワード・ファイルを使用する SYSDBA/SYSOPER（パスワードを使用するユーザー名付きのプロキシ）: PASSWORD ■ Kerberos 認証済のエンタープライズ・ユーザーまたは外部ユーザー : KERBEROS ■ SSL 認証済のエンタープライズ・ユーザーまたは外部ユーザー : SSL ■ RADIUS 認証済の外部ユーザー : RADIUS ■ OS 認証済の外部ユーザーまたは SYSDBA/SYSOPER: OS ■ DCE 認証済の外部ユーザー : DCE ■ 証明書付きプロキシ、DN、またはパスワードを使用しないユーザー名 : NONE ■ バックグラウンド・プロセス（ジョブ・キュー・スレーブ・プロセス）: JOB IDENTIFICATION_TYPE を使用すると、認証方式が PASSWORD、KERBEROS または SSL の場合に、外部ユーザーとエンタープライズ・ユーザーを区別できます。

表 5-11 ネームスペース USERENV の事前定義パラメータ（続き）

パラメータ	戻り値
BG_JOB_ID	現行のセッションが Oracle Database のバックグラウンド・プロセスで確立された場合、そのセッションのジョブ ID を戻します。セッションがバックグラウンド・プロセスで確立されていない場合は、NULL を戻します。
CLIENT_IDENTIFIER	アプリケーションによって設定された識別子を、DBMS_SESSION.SET_IDENTIFIER プロシージャ、OCI 属性 OCI_ATTR_CLIENT_IDENTIFIER または Java クラス Oracle.jdbc.OracleConnection.setClientIdentifier を使用して戻します。この属性は、同じデータベース・ユーザーとして認証される複数の軽量アプリケーション・ユーザーを識別するために、様々なデータベース・コンポーネントによって使用されます。
CLIENT_INFO	DBMS_APPLICATION_INFO パッケージを使用するアプリケーションが格納できる 64 バイトまでのユーザー・セッション情報を戻します。
CURRENT_BIND	ファイングレイン監査のバインド変数。
CURRENT_SCHEMA	現在アクティブなデフォルトのスキーマの名前。この値は、セッションの存続期間中に ALTER SESSION SET CURRENT_SCHEMA 文を使用して変更できます。また、この値は、セッションの存続期間中に、アクティブな定義者権限オブジェクトの所有者を反映するために変更することもできます。この値をビュー定義の本体で直接使用すると、そのビューを使用しているカーソルを実行するときに使用されるデフォルトのスキーマが戻されます。定義者権限としてカーソルで使用されるビューは考慮されません。 **注意：**ストアド PL/SQL ユニット内から SQL 文 ALTER SESSION SET CURRENT_SCHEMA を発行しないことをお薦めします。
CURRENT_SCHEMAID	現在アクティブなデフォルトのスキーマの識別子。
CURRENT_SQL CURRENT_SQL*n*	CURRENT_SQL は、ファイングレイン監査イベントをトリガーによって実行した現行の SQL の最初の 4KB を戻します。CURRENT_SQL*n* 属性は、後続の 4KB ずつを戻します。*n* は、1 ～ 7（1 および 7 を含む）の整数です。CURRENT_SQL1 は 5 ～ 8KB、CURRENT_SQL2 は 9 ～ 12KB のように戻します。これらの属性は、ファイングレイン監査機能のイベント・ハンドラ内のみで指定できます。
CURRENT_SQL_LENGTH	ファイングレイン監査または行レベルのセキュリティ（RLS）ポリシーのファンクションまたはイベント・ハンドラをトリガーする現行の SQL 文の長さ。ファンクションまたはイベント・ハンドラ内のみで有効です。
CURRENT_USER	権限が現在アクティブになっているデータベース・ユーザーの名前。この値は、セッションの存続期間中に、アクティブな定義者の権限オブジェクトの所有者を反映するために変更できます。定義者権限オブジェクトがアクティブでない場合、CURRENT_USER は SESSION_USER と同じ値を戻します。この値をビュー定義の本体で直接使用すると、そのビューを使用しているカーソルを実行しているユーザーが戻されます。定義者権限としてカーソルで使用されるビューは考慮されません。
CURRENT_USERID	権限が現在アクティブになっているデータベース・ユーザーの識別子。
DB_DOMAIN	DB_DOMAIN 初期化パラメータで指定されたデータベースのドメインを戻します。
DB_NAME	DB_NAME 初期化パラメータで指定されたデータベース名を戻します。
DB_UNIQUE_NAME	DB_UNIQUE_NAME 初期化パラメータで指定されたデータベース名を戻します。
ENTRYID	現行の監査エントリ番号を戻します。監査エントリ ID の順序は、ファイングレイン監査レコードと通常の監査レコードで共通です。この属性を分散 SQL 文で使用することはできません。正しい監査エントリ識別子は、標準またはファイングレイン監査の監査ハンドラを介してのみ参照できます。

表 5-11 ネームスペース USERENV の事前定義パラメータ（続き）

パラメータ	戻り値
ENTERPRISE_IDENTITY	ユーザーのエンタープライズ全体の識別を戻します。 ■ エンタープライズ・ユーザーの場合 : Oracle Internet Directory DN ■ 外部ユーザーの場合 : 外部識別（Kerberos プリンシパル名、RADIUS および DCE スキーマ名、OS ユーザー名、証明書 DN） ■ ローカル・ユーザーおよび SYSDBA/SYSOPER ログインの場合 : NULL 属性の値はプロキシ方式によって異なります。 ■ DN 付きプロキシの場合 : クライアントの Oracle Internet Directory DN ■ 証明書付きプロキシの場合 : クライアントの証明書 DN（外部ユーザー）、Oracle Internet Directory DN（グローバル・ユーザー） ■ ユーザー名付きプロキシの場合 : クライアントがエンタープライズ・ユーザーの場合は Oracle Internet Directory DN、クライアントがローカル・データベースのユーザーの場合は NULL
FG_JOB_ID	現行のセッションが Oracle のフォアグラウンド・プロセスで確立された場合、そのセッションのジョブ ID を戻します。セッションがフォアグラウンド・プロセスで確立されていない場合は、NULL を戻します。
GLOBAL_CONTEXT_MEMORY	コンテキストへのグローバルなアクセスによって、システム・グローバル領域で使用された数値を戻します。
GLOBAL_UID	エンタープライズ・ユーザー・セキュリティ（EUS）ログインの場合は、Oracle Internet Directory からグローバル・ユーザー ID を戻します。その他のすべてのログインの場合は、NULL を戻します。
HOST	接続中のクライアントのホスト・マシン名を戻します。
IDENTIFICATION_TYPE	データベースでユーザーのスキーマを作成した方法を戻します。特に、CREATE/ALTER USER 構文に、IDENTIFIED 句が反映されます。次のリストでは、スキーマの作成中に使用する構文の後に、戻される識別タイプが続きます。 ■ IDENTIFIED BY パスワード : LOCAL ■ IDENTIFIED EXTERNALLY: EXTERNAL ■ IDENTIFIED GLOBALLY: GLOBAL SHARED ■ IDENTIFIED GLOBALLY AS *DN*: GLOBAL PRIVATE
INSTANCE	現行のインスタンスのインスタンス識別番号を戻します。
INSTANCE_NAME	インスタンス名。
IP_ADDRESS	接続中のクライアントのマシンの IP アドレスを戻します。
ISDBA	オペレーティング・システムまたはパスワード・ファイルによって、ユーザーが DBA 権限を持っていると認証された場合、TRUE を戻します。
LANG	言語名の ISO 略称を戻します。これは、既存の 'LANGUAGE' パラメータを短縮したものです。
LANGUAGE	現行のセッションで使用している言語（language）および地域（territory）を、データベース・キャラクタ・セット（character set）も含めて次の書式で戻します。 language_territory.characterset
MODULE	DBMS_APPLICATION_INFO パッケージまたは OCI を使用して設定されたアプリケーション名（モジュール）を戻します。
NETWORK_PROTOCOL	接続文字列の 'PROTOCOL=*protocol*' の部分で指定された、通信に使用されるネットワーク・プロトコルを戻します。
NLS_CALENDAR	現行のセッションの現行のカレンダを戻します。
NLS_CURRENCY	現行のセッションの通貨を戻します。

表 5-11 ネームスペース USERENV の事前定義パラメータ（続き）

パラメータ	戻り値
NLS_DATE_FORMAT	セッションの日付書式を戻します。
NLS_DATE_LANGUAGE	日付の表示に使用される言語を戻します。
NLS_SORT	BINARY または言語ソート基準を戻します。
NLS_TERRITORY	現行のセッションの地域を戻します。
OS_USER	データベース・セッションを初期化するクライアント・プロセスのオペレーティング・システム・ユーザー名を戻します。
POLICY_INVOKER	行レベルのセキュリティ（RLS）・ポリシーのファンクションの実行者。
PROXY_ENTERPRISE_IDENTITY	プロキシ・ユーザーがエンタープライズ・ユーザーの場合に、Oracle Internet Directory DN を戻します。
PROXY_USER	SESSION_USER のかわりに現行のセッションを開いたデータベース・ユーザー名を戻します。
PROXY_USERID	SESSION_USER のかわりに現行のセッションを開いたデータベース・ユーザーの ID を戻します。
SERVER_HOST	インスタンスを実行しているマシンのホスト名。
SERVICE_NAME	任意のセッションで接続しているサービスの名前を戻します。
SESSION_USER	ログオン時のデータベース・ユーザーの名前。エンタープライズ・ユーザーの場合、スキーマを戻します。その他のユーザーの場合、データベース・ユーザー名を戻します。この値は、セッションの存続期間中は同じです。
SESSION_USERID	ログオン時のデータベース・ユーザーの識別子。
SESSIONID	監査セッション識別子を戻します。この属性を分散 SQL 文で使用することはできません。
SID	セッション番号（セッション ID とは異なります）。
STATEMENTID	文の監査の識別子を戻します。STATEMENTID は、任意のセッションで監査された SQL 文の番号を示します。この属性を分散 SQL 文で使用することはできません。正しい文の監査の識別子は、標準またはファイングレイン監査の監査ハンドラを介してのみ参照できます。
TERMINAL	現行のセッションのクライアントに対するオペレーティング・システムの識別子を戻します。分散 SQL 文では、この属性はローカル・セッションの識別子を戻します。分散環境では、リモートの SELECT に対してのみこのオプションを使用でき、リモートの INSERT、UPDATE または DELETE には使用できません。（このパラメータの戻り値の長さはオペレーティング・システムによって異なります。）

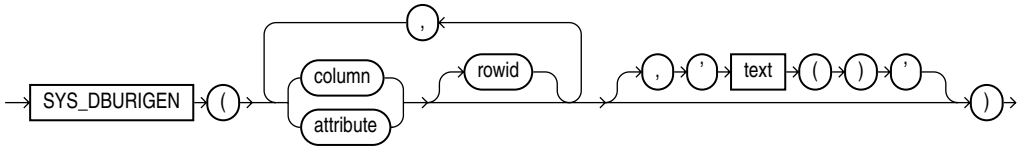
表 5-12 に、非推奨になったネームスペース USERENV のパラメータを示します。これらのパラメータはどれも指定しないでください。かわりに、「コメント」列で提案する代替手段を使用します。

表 5-12 非推奨になったネームスペース USERENV のパラメータ

パラメータ	コメント
AUTHENTICATION_TYPE	このパラメータは、ユーザーの認証方法を示す値を戻しました。現在、同じ情報は、IDENTIFICATION_TYPE と組み合せた新しい AUTHENTICATION_METHOD パラメータから取得できます。
EXTERNAL_NAME	このパラメータは、ユーザーの外部名を戻しました。詳細な情報は、AUTHENTICATED_IDENTITY および ENTERPRISE_IDENTITY パラメータから取得できます。

SYS_DBURIGEN

構文



用途

SYS_DBURIGen は、引数として 1 つ以上の列または属性、およびオプションで ROWID を取り、特定の列または行オブジェクトへの DBUriType データ型の URL を生成します。これによって、データベースから XML 文書を検索するための URL を使用できるようになります。

参照されるすべての列または属性は、サンプル表に存在する必要があります。これらは、主キーの役割を果たす必要があります。実際に表の主キーに一致する必要はありませんが、一意の値を参照する必要があります。複数の列を指定すると、最後の列以外のすべての列はデータベースの行を識別し、指定された最後の列は行にある列を識別します。

デフォルトでは、URL はフォーマットされた XML 文書を指します。ドキュメントのテキストのみを指す場合は、オプションの 'text()' を指定します

> **注意：** この XML コンテキストでは、小文字の text はキーワードであり、構文のプレースホルダではありません。

列または属性を含む表またはビューが、問合せのコンテキストで指定されるスキーマを持たない場合、Oracle Database は、表名またはビュー名をパブリック・シノニムとして解析します。

> **参照：** データベースの URIType データ型および XML 文書の詳細は、『Oracle XML Developer's Kit プログラマーズ・ガイド』を参照してください。

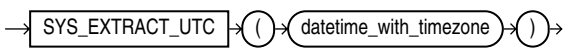
例

次の例では、SYS_DBURIGen ファンクションを使用して、サンプル表 hr.employees の employee_id = 206 である行の email 列への DBUriType データ型の URL を生成します。

```
SELECT SYS_DBURIGEN(employee_id, email)
   FROM employees
   WHERE employee_id = 206;

SYS_DBURIGEN(EMPLOYEE_ID,EMAIL)(URL, SPARE)
--------------------------------------------------------------------------------
DBURITYPE('/PUBLIC/EMPLOYEES/ROW[EMPLOYEE_ID=''206'']/EMAIL', NULL)
```

SYS_EXTRACT_UTC

構文



用途

SYS_EXTRACT_UTC は、タイムゾーン・オフセットまたはタイムゾーン地域名を含む日時値から協定世界時（UTC）（以前のグリニッジ標準時）を抽出します。

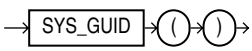
例

次の例では、指定された日時から UTC を抽出します。

```
SELECT SYS_EXTRACT_UTC(TIMESTAMP '2000-03-28 11:30:00.00 -08:00')
   FROM DUAL;

SYS_EXTRACT_UTC(TIMESTAMP'2000-03-2811:30:00.00-08:00')
------------------------------------------------------------------
28-MAR-00 07.30.00 PM
```

SYS_GUID

構文



用途

SYS_GUID は、16 バイトで構成されたグローバルな一意の識別子（RAW 値）を生成して戻します。多くのプラットフォームでは、生成された識別子は、ホスト識別子、プロセス、またはファンクションをコールするプロセスやスレッドのスレッド識別子、およびそのプロセスやスレッドに対する非反復値（バイトの順序）で構成されています。

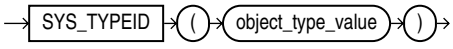
例

次の例では、サンプル表 hr.locations に列を追加後、一意の識別子を各行に挿入し、グローバルな一意識別子の 16 バイトの RAW 値を 32 文字の 16 進表記で戻します。

```
ALTER TABLE locations ADD (uid_col RAW(32));

UPDATE locations SET uid_col = SYS_GUID();

SELECT location_id, uid_col FROM locations
   ORDER BY location_id, uid_col;

LOCATION_ID UID_COL
----------- --------------------------------------------------------------
       1000 09F686761827CF8AE040578CB20B7491
       1100 09F686761828CF8AE040578CB20B7491
       1200 09F686761829CF8AE040578CB20B7491
       1300 09F68676182ACF8AE040578CB20B7491
       1400 09F68676182BCF8AE040578CB20B7491
       1500 09F68676182CCF8AE040578CB20B7491
. . .
```

SYS_TYPEID

構文



用途

SYS_TYPEID は、オペランドで最も指定される型の型 ID を戻します。この値は、主に、置換可能な列の基礎となる型判別式の列を識別するために使用されます。たとえば、型判別式の列に索引を構築するために、SYS_TYPEID によって戻される値を使用できます。

このファンクションは、オブジェクト型のオペランドのみで使用してください。すべての最終ルート・オブジェクト型（型階層に属さない最終型）は、NULL 型の ID を持ちます。Oracle Database は、型階層に属するすべての型に、NULL 以外の一意の型 ID を割り当てます。

> **参照：** 型 ID の詳細は、『Oracle Database オブジェクト・リレーショナル開発者ガイド』を参照してください。

例

次の例では、16-61 ページの「置換可能な表および列のサンプル :」で作成された表 persons および books を使用します。最初の問合せは、persons 表に格納されたオブジェクト・インスタンスの最も指定される型を戻します。

```
SELECT name, SYS_TYPEID(VALUE(p)) "Type_id" FROM persons p;

NAME                      Type_id
------------------------- --------------------------------
Bob                       01
Joe                       02
Tim                       03
```

次の問合せは、books 表に格納された作者の最も指定される型を戻します。

```
SELECT b.title, b.author.name, SYS_TYPEID(author)
   "Type_ID" FROM books b;

TITLE                     AUTHOR.NAME          Type_ID
------------------------- -------------------- --------------------
An Autobiography          Bob                  01
Business Rules            Joe                  02
Mixing School and Work    Tim                  03
```

SYS_TYPEID ファンクションを使用すると、表の型判別式の列に索引を作成できます。14-72 ページの「置換可能な列の索引の作成例 :」を参照してください。

SYS_XMLAGG

構文

用途

SYS_XMLAGG は、*expr* によって表されるすべての XML 文書または XML フラグメントを集約し、単一の XML 文書を生成します。このファンクションは、デフォルト名 ROWSET の新しい囲み要素を追加します。XML 文書を別の方法でフォーマットする場合は、XMLFormat オブジェクトのインスタンスである *fmt* を指定します。

> **参照：** SYS_XMLAgg の結果をフォーマットするための XMLFormat 型の属性の使用については、5-193 ページの「SYS_XMLGEN」および 2-66 ページの「XML 書式モデル」を参照してください。

例

次の例では、SYS_XMLGEN ファンクションを使用して、従業員名の最初の文字が R であるサンプル表 employees の各行に対して、XML 文書を生成した後、デフォルトの囲み要素 ROWSET の 1 つの XML 文書にすべての行を集約します。

```
SELECT SYS_XMLAGG(SYS_XMLGEN(last_name)) XMLAGG
   FROM employees
   WHERE last_name LIKE 'R%'
   ORDER BY xmlagg;

XMLAGG
--------------------------------------------------------------------------------
<?xml version="1.0"?>
<ROWSET>
<LAST_NAME>Rajs</LAST_NAME>
<LAST_NAME>Raphaely</LAST_NAME>
<LAST_NAME>Rogers</LAST_NAME>
<LAST_NAME>Russell</LAST_NAME>
</ROWSET>
```

SYS_XMLGEN

構文

用途

SYS_XMLGEN は、データベースの特定の行および列を評価する式を取り、XML 文書を含む XMLType 型のインスタンスを戻します。*expr* は、スカラー値、ユーザー定義型または XMLType インスタンスです。

- *expr* がスカラー値である場合、ファンクションはスカラー値を含む XML 要素を戻します。
- *expr* が型である場合、ファンクションは XML 要素へユーザー定義型の属性をマップします。
- *expr* が XMLType インスタンスである場合、ファンクションはデフォルトのタグ名が ROW である XML 要素でドキュメントを囲みます。

デフォルトでは、XML 文書の要素は *expr* の要素と一致します。たとえば、*expr* が列名に変換される場合、XML の囲み要素は同じ列名になります。XML 文書を別の方法でフォーマットする場合は、XMLFormat オブジェクトのインスタンスである *fmt* を指定します。

参照：

- XMLFormat 型の詳細および SYS_XMLGEN の結果を書式化するための属性の使用方法については、2-66 ページの「XML 書式モデル」を参照してください。
- XML 型およびそれらの使用の概要については、『Oracle Database 概要』を参照してください。

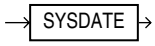
例

次の例では、サンプル表 oe.employees から employee_id 値が 205 の従業員の電子メール ID を検出し、EMAIL 要素を持つ XML 文書を含む XMLType のインスタンスを生成します。

```
SELECT SYS_XMLGEN(email)
   FROM employees
   WHERE employee_id = 205;

SYS_XMLGEN(EMAIL)
--------------------------------------------------------------------
<?xml version="1.0"?>
<EMAIL>SHIGGINS</EMAIL>
```

SYSDATE

構文



用途

SYSDATE は、データベースが存在するオペレーティング・システムの現在の日付と時刻のセットを戻します。戻り値のデータ型は DATE です。戻り値の書式は、NLS_DATE_FORMAT 初期化パラメータの値によって異なります。このファンクションに引数は不要です。分散 SQL 文では、このファンクションはローカル・データベースのオペレーティング・システムの日付と時刻のセットを戻します。CHECK 制約の条件でこのファンクションは使用できません。

例

次の例では、オペレーティング・システムの現在の日付および時刻を戻します。

```
SELECT TO_CHAR
    (SYSDATE, 'MM-DD-YYYY HH24:MI:SS') "NOW"
     FROM DUAL;

NOW
-------------------
04-13-2001 09:45:51
```

SYSTIMESTAMP

構文

用途

SYSTIMESTAMP は、データベースが存在するシステムの、秒の小数部とタイムゾーンを含む日付を戻します。戻り型は、TIMESTAMP WITH TIME ZONE です。

例

次の例では、システムのタイムスタンプを戻します。

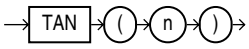
```
SELECT SYSTIMESTAMP FROM DUAL;

SYSTIMESTAMP
---------------------------------------------------------------------
28-MAR-00 12.38.55.538741 PM -08:00
```

次の例では、秒の小数部を明示的に指定する方法を示します。

```
SELECT TO_CHAR(SYSTIMESTAMP, 'SSSSS.FF') FROM DUAL;

TO_CHAR(SYSTIME
---------------
55615.449255
```

TAN

構文



用途

TAN は、*n*（ラジアンで表された角度）のタンジェントを戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数が BINARY_FLOAT の場合、このファンクションは BINARY_DOUBLE を戻します。それ以外の場合、引数と同じ数値データ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

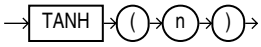
例

次の例では、135 度のタンジェントを戻します。

```
SELECT TAN(135 * 3.14159265359/180)
   "Tangent of 135 degrees"  FROM DUAL;

Tangent of 135 degrees
----------------------
                    - 1
```

TANH

構文



用途

TANH は、*n* の双曲線タンジェントを戻します。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。引数が BINARY_FLOAT の場合、このファンクションは BINARY_DOUBLE を戻します。それ以外の場合、引数と同じ数値データ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

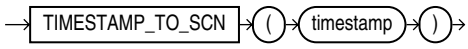
例

次の例では、5 の双曲線タンジェントを戻します。

```
SELECT TANH(.5) "Hyperbolic tangent of .5"
   FROM DUAL;

Hyperbolic tangent of .5
------------------------
              .462117157
```

TIMESTAMP_TO_SCN

構文



用途

TIMESTAMP_TO_SCN は、引数としてタイムスタンプ値を取り、そのタイムスタンプに関連付けられたシステム変更番号（SCN）の概数を戻します。戻り値のデータ型は NUMBER です。このファンクションは、特定のタイムスタンプに関連付けられた SCN を調べる場合に有効です。

> **注意：** SCN と SCN 生成時のタイムスタンプの関連は、一定期間データベースで記憶されます。この期間は、最大で自動調整された UNDO 保存期間（データベースが自動 UNDO 管理モードで実行されている場合）およびデータベース内のすべてのフラッシュバック・アーカイブの保存時間となりますが、120 時間以上になります。関連が不要となるまでの経過時間には、データベースが開かれているときの時間のみが加算されます。TIMESTAMP_TO_SCN の引数に対して指定された SCN が古すぎる場合は、エラーが戻されます。

> **参照：** SCN をタイムスタンプへ変換する方法については、5-162 ページの「SCN_TO_TIMESTAMP」を参照してください。

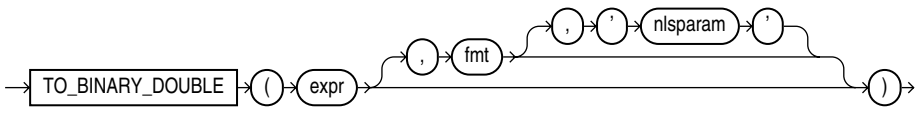
例

次の例では、行を oe.orders 表に挿入し、TIMESTAMP_TO_SCN を使用して、挿入操作のシステム変更番号を判断します。（実際に戻される SCN は、システムごとに異なります。）

```
INSERT INTO orders (order_id, order_date, customer_id, order_total)
   VALUES (5000, SYSTIMESTAMP, 188, 2345);
1 row created.

COMMIT;
Commit complete.

SELECT TIMESTAMP_TO_SCN(order_date) FROM orders
   WHERE order_id = 5000;

TIMESTAMP_TO_SCN(ORDER_DATE)
----------------------------
                      574100
```

TO_BINARY_DOUBLE

構文



用途

TO_BINARY_DOUBLE は、倍精度の浮動小数点数を戻します。

- *expr* には、文字列か、または NUMBER、BINARY_FLOAT または BINARY_DOUBLE 型の数値を指定できます。*expr* が BINARY_DOUBLE の場合、このファンクションは *expr* を戻します。
- オプションの '*fmt*' 引数および '*nlsparam*' 引数は *expr* が文字列の場合にのみ有効です。これらの引数は、TO_CHAR（数値）ファンクションの場合と同じ用途に使用されます。
  - 文字列 'INF'（大 / 小文字は区別されない）は、正の無限大に変換されます。
  - 文字列 '-INF'（大 / 小文字は区別されない）は、負の無限大に変換されます。
  - 文字列 'NaN'（大 / 小文字は区別されない）は、NaN（非数値）に変換されます。

expr 文字列には、浮動小数点数の書式要素（F、f、D または d）は使用できません。

文字列または NUMBER から BINARY_DOUBLE への変換は、正確に行われない場合があります。これは、NUMBER および文字列型では単精度、BINARY_DOUBLE で倍精度を使用して数値を表現するためです。

BINARY_FLOAT から BINARY_DOUBLE への変換は正確に行われます。

> **参照：** 5-201 ページの「TO_CHAR（数値）」および 2-13 ページの「浮動小数点数」を参照してください。

例

次の例では、それぞれ異なる数値データ型の 3 つの列を持つ次の表を使用します。

```
CREATE TABLE float_point_demo
  (dec_num NUMBER(10,2), bin_double BINARY_DOUBLE, bin_float BINARY_FLOAT);

INSERT INTO float_point_demo
  VALUES (1234.56,1234.56,1234.56);

SELECT * FROM float_point_demo;

   DEC_NUM BIN_DOUBLE  BIN_FLOAT
---------- ---------- ----------
   1234.56 1.235E+003 1.235E+003
```

次の例では、NUMBER データ型の値を BINARY_DOUBLE データ型の値に変換します。

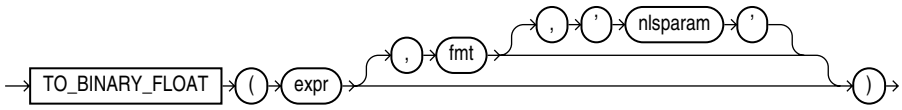
```
SELECT dec_num, TO_BINARY_DOUBLE(dec_num)
  FROM float_point_demo;

   DEC_NUM TO_BINARY_DOUBLE(DEC_NUM)
---------- -------------------------
   1234.56                1.235E+003
```

次の例では、dec_num 列および bin_double 列から抽出されたダンプ情報を比較します。

```
SELECT DUMP(dec_num) "Decimal",
   DUMP(bin_double) "Double"
   FROM float_point_demo;

Decimal                    Double
-------------------------- ---------------------------------------------
Typ=2 Len=4: 194,13,35,57  Typ=101 Len=8: 192,147,74,61,112,163,215,10
```

TO_BINARY_FLOAT

構文



用途

TO_BINARY_FLOAT は、単精度の浮動小数点数を戻します。

- *expr* には、文字列か、または NUMBER、BINARY_FLOAT または BINARY_DOUBLE 型の数値を指定できます。*expr* が BINARY_FLOAT の場合、このファンクションは *expr* を戻します。
- オプションの '*fmt*' 引数および '*nlsparam*' 引数は *expr* が文字列の場合にのみ有効です。これらの引数は、TO_CHAR（数値）ファンクションの場合と同じ用途に使用されます。
  - 文字列 'INF'（大 / 小文字は区別されない）は、正の無限大に変換されます。
  - 文字列 '-INF'（大 / 小文字は区別されない）は、負の無限大に変換されます。
  - 文字列 'NaN'（大 / 小文字は区別されない）は、NaN（非数値）に変換されます。

expr 文字列には、浮動小数点数の書式要素（F、f、D または d）は使用できません。

文字列または NUMBER から BINARY_FLOAT への変換は、正確に行われない場合があります。これは、NUMBER および文字列型では単精度、BINARY_FLOAT では倍精度を使用して数値を表現するためです。

BINARY_DOUBLE 値に、BINARY_FLOAT がサポートする数を超える精度ビットが使用されている場合、BINARY_DOUBLE から BINARY_FLOAT への変換は正確に行われません。

> **参照：** 5-201 ページの「TO_CHAR（数値）」および 2-13 ページの「浮動小数点数」を参照してください。

例

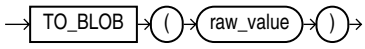
次の例では、5-196 ページの「TO_BINARY_DOUBLE」で作成した float_point_demo 表を使用して、NUMBER データ型の値を BINARY_FLOAT データ型の値に変換します。

```
SELECT dec_num, TO_BINARY_FLOAT(dec_num)
  FROM float_point_demo;

   DEC_NUM TO_BINARY_FLOAT(DEC_NUM)
---------- ------------------------
   1234.56                1.235E+003
```

TO_BLOB

構文

***to_blob*::=**



用途

TO_BLOB は、LONG RAW および RAW 値を BLOB 値に変換します。

例

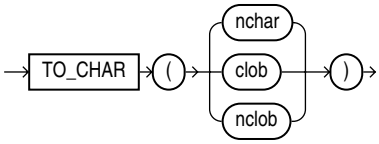
次の例は、RAW 列の値として BLOB を戻します。

```
SELECT TO_BLOB(raw_column) blob FROM raw_table;

BLOB
------------------------
00AADD343CDBBD
```

TO_CHAR（文字）

構文

***to_char_char*::=**



用途

TO_CHAR（文字）は、NCHAR、NVARCHAR2、CLOB または NCLOB データをデータベース・キャラクタ・セットに変換します。戻り値は常に VARCHAR2 です。

このファンクションを使用してキャラクタ LOB をデータベース・キャラクタ・セットに変換すると、変換する LOB 値がターゲットの型よりも大きい場合、エラーが戻されます。

このファンクションは任意の XML ファンクションと組み合せて使用できますが、生成される日付の書式は、XML スキーマの標準書式ではなく、データベースの書式になります。

参照 :

- XML の日付およびタイムスタンプの書式設定とその例については、『Oracle XML DB 開発者ガイド』を参照してください。
- XML ファンクションのリストについては、5-7 ページの「XML ファンクション」を参照してください。

例

次の例では、単純な文字列を文字データとして解析します。

```
SELECT TO_CHAR('01110') FROM DUAL;

TO_CH
-----
01110
```

この例と、5-201 ページの「TO_CHAR（数値）」の最初の例を比較してください。

次の例では、pm.print_media 表の CLOB データをデータベース・キャラクタ・セットに変換します。

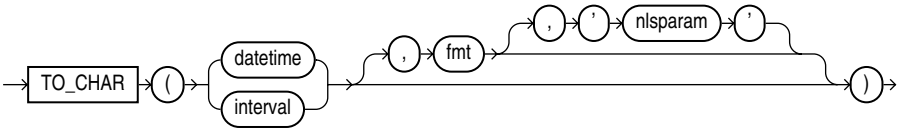
```
SELECT TO_CHAR(ad_sourcetext) FROM print_media
      WHERE product_id = 2268;

TO_CHAR(AD_SOURCETEXT)
--------------------------------------------------------------------------------
*****************************
TIGER2 2268...Standard Hayes Compatible Modem
```

```
Product ID: 2268
The #1 selling modem in the universe! Tiger2's modem includes call management
and Internet voicing. Make real-time full duplex phone calls at the same time
you're online.
**********
```

TO_CHAR（日時）

構文

***to_char_date*::=**



用途

TO_CHAR（日時）は、DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONE または TIMESTAMP WITH LOCAL TIME ZONE データ型の日時値または期間値を日付書式 *fmt* で指定された書式の VARCHAR2 データ型に変換します。*fmt* を省略すると、次のように、*date* は VARCHAR2 値に変換されます。

- DATE 値は、デフォルトの日付書式の値に変換されます。
- TIMESTAMP および TIMESTAMP WITH LOCAL TIME ZONE 値は、デフォルトのタイムスタンプ書式の値に変換されます。
- TIMESTAMP WITH TIME ZONE 値は、タイムゾーン書式のデフォルトのタイムスタンプの値に変換されます。

日時書式の詳細は、2-54 ページの「書式モデル」を参照してください。

'nlsparam' 引数には、月と日の名前および略称が戻される言語を指定します。この引数は、次の書式で指定します。

```
'NLS_DATE_LANGUAGE = language'
```

'nlsparam' を指定しないと、このファンクションはセッションのデフォルト日付言語を使用します。

> **参照：** 「データ変換のセキュリティ上の考慮事項」（2-43 ページ）

例

次の例で使用する表は、次のとおりです。

```
CREATE TABLE date_tab (
   ts_col      TIMESTAMP,
   tsltz_col   TIMESTAMP WITH LOCAL TIME ZONE,
   tstz_col    TIMESTAMP WITH TIME ZONE);
```

次の例では、TO_CHAR を別の TIMESTAMP データ型に適用した結果を示します。TIMESTAMP WITH LOCAL TIME ZONE 列の結果は、セッションのタイムゾーンを識別します。これに対して、TIMESTAMP および TIMESTAMP WITH TIME ZONE 列の結果は、セッションのタイムゾーンを識別しません。

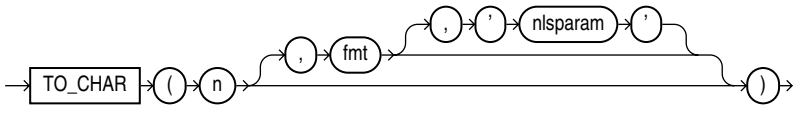
```
ALTER SESSION SET TIME_ZONE = '-8:00';
INSERT INTO date_tab VALUES (
   TIMESTAMP'1999-12-01 10:00:00',
   TIMESTAMP'1999-12-01 10:00:00',
   TIMESTAMP'1999-12-01 10:00:00');
```

```
INSERT INTO date_tab VALUES (
   TIMESTAMP'1999-12-02 10:00:00 -8:00',
   TIMESTAMP'1999-12-02 10:00:00 -8:00',
   TIMESTAMP'1999-12-02 10:00:00 -8:00');

SELECT TO_CHAR(ts_col, 'DD-MON-YYYY HH24:MI:SSxFF') AS ts_date,
   TO_CHAR(tstz_col, 'DD-MON-YYYY HH24:MI:SSxFF TZH:TZM') AS tstz_date
   FROM date_tab
   ORDER BY ts_date, tstz_date;

TS_DATE                       TSTZ_DATE
----------------------------- -----------------------------------
01-DEC-1999 10:00:00.000000    01-DEC-1999 10:00:00.000000 -08:00
02-DEC-1999 10:00:00.000000    02-DEC-1999 10:00:00.000000 -08:00

SELECT SESSIONTIMEZONE,
   TO_CHAR(tsltz_col, 'DD-MON-YYYY HH24:MI:SSxFF') AS tsltz
   FROM date_tab
   ORDER BY sessiontimezone, tsltz;

SESSIONTIM TSLTZ
---------- ------------------------------
-08:00     01-DEC-1999 10:00:00.000000
-08:00     02-DEC-1999 10:00:00.000000

ALTER SESSION SET TIME_ZONE = '-5:00';
SELECT TO_CHAR(ts_col, 'DD-MON-YYYY HH24:MI:SSxFF') AS ts_col,
   TO_CHAR(tstz_col, 'DD-MON-YYYY HH24:MI:SSxFF TZH:TZM') AS tstz_col
   FROM date_tab
   ORDER BY ts_col, tstz_col;

TS_COL                        TSTZ_COL
----------------------------- -----------------------------------
01-DEC-1999 10:00:00.000000    01-DEC-1999 10:00:00.000000 -08:00
02-DEC-1999 10:00:00.000000    02-DEC-1999 10:00:00.000000 -08:00

SELECT SESSIONTIMEZONE,
TO_CHAR(tsltz_col, 'DD-MON-YYYY HH24:MI:SSxFF') AS tsltz_col
   FROM date_tab
   ORDER BY sessiontimezone, tsltz_col;
  2    3    4
SESSIONTIM TSLTZ_COL
---------- ------------------------------
-05:00     01-DEC-1999 13:00:00.000000
-05:00     02-DEC-1999 13:00:00.000000
```

TO_CHAR（数値）

構文

***to_char_number*::=**



用途

TO_CHAR（数値）は、*n* を、オプションの数値書式 *fmt* を使用して VARCHAR2 データ型の値に変換します。*n* 値には、NUMBER、BINARY_FLOAT または BINARY_DOUBLE 型の値を指定できます。*fmt* を指定しないと、*n* の有効桁数を保持するために十分な長さの VARCHAR2 値に変換されます。

n が負の場合、書式が適用された後で符号が適用されます。したがって、TO_CHAR(-1, '$9') は、$-1 ではなく -$1 を戻します。

日時書式の詳細は、2-54 ページの「書式モデル」を参照してください。

'nlsparam' 引数には、数値書式要素によって戻される次の文字を指定します。

- 小数点文字
- 桁区切り
- 各国通貨記号
- 国際通貨記号

この引数は、次の書式で指定します。

```
'NLS_NUMERIC_CHARACTERS = ''dg''
   NLS_CURRENCY = ''text''
   NLS_ISO_CURRENCY = territory '
```

文字 *d* および *g* は、それぞれ小数点文字および桁区切りを表します。これらは、異なるシングルバイト文字である必要があります。引用符付き文字列の中では、パラメータ値を囲む一重引用符を 2 つ使用する必要があります。通貨記号には 10 文字使用できます。

'nlsparam' またはパラメータのいずれか 1 つを省略すると、このファンクションはセッションのデフォルト・パラメータ値を使用します。

> **参照：** 「データ変換のセキュリティ上の考慮事項」（2-43 ページ）

例

次の文は、暗黙的な変換を使用して、文字列と数値を数値に結合します。

```
SELECT TO_CHAR('01110' + 1) FROM dual;

TO_C
----
1111
```

この例と、5-199 ページの「TO_CHAR（文字）」の最初の例を比較してください。

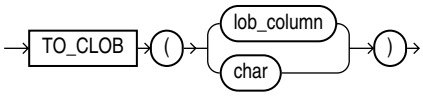
次の例では、出力で通貨記号の左側に空白埋めが行われます。

```
SELECT TO_CHAR(-10000,'L99G999D99MI') "Amount"
     FROM DUAL;

Amount
--------------
  $10,000.00-

SELECT TO_CHAR(-10000,'L99G999D99MI',
   'NLS_NUMERIC_CHARACTERS = '',.''
   NLS_CURRENCY = ''AusDollars'' ') "Amount"
     FROM DUAL;

Amount
-------------------
AusDollars10.000,00-
```

オプションの数値書式 *fmt* では、L は各国通貨記号を、MI は後に付くマイナス記号（-）を表します。すべての数値書式要素のリストは、2-65 ページの表 2-17「FX 書式モデル修飾子による文字データと書式モデルの一致」を参照してください。

TO_CLOB

構文



用途

TO_CLOB は、LOB 列またはその他の文字列の NCLOB 値を CLOB 値に変換します。*char* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。Oracle Database は、基礎となる LOB データを各国語キャラクタ・セットからデータベース・キャラクタ・セットに変換することによって、このファンクションを実行します。

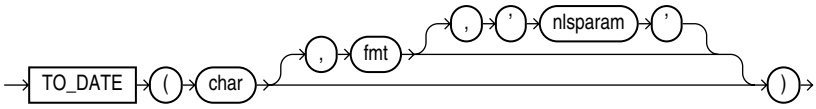
例

次の文は、サンプル表 pm.print_media の NCLOB データを CLOB に変換し、CLOB 列に挿入します。

```
UPDATE PRINT_MEDIA
   SET AD_FINALTEXT = TO_CLOB (AD_FLTEXTN);
```

TO_DATE

構文



用途

TO_DATE は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型の *char* を、DATE データ型の値に変換します。

> **注意：** このファンクションは、データを別の日時データ型へは変換しません。他の日時変換の詳細は、5-210 ページの「TO_TIMESTAMP」、5-211 ページの「TO_TIMESTAMP_TZ」、5-205 ページの「TO_DSINTERVAL」および 5-212 ページの「TO_YMINTERVAL」を参照してください。

fmt は、*char* の書式を指定する日時書式モデルです。*fmt* を指定しない場合、*char* はデフォルトの日付書式である必要があります。デフォルトの日付書式は、NLS_TERRITORY 初期化パラメータによって暗黙的に決まります。NLS_DATE_FORMAT パラメータによって明示的に設定することもできます。*fmt* が J（ユリウス日）の場合、*char* は整数である必要があります。

注意： 次の項の例に示すとおり、TO_DATE では、常に書式マスク（*fmt*）を指定することをお薦めします。書式マスクを指定しないと、このファンクションは、*char* が NLS_TERRITORY または NLS_DATE_FORMAT パラメータで指定されたものと同じ書式を使用している場合にのみ有効になります。さらに、依存性を回避するために明示的な書式マスクが指定されていない場合は、データベース間でファンクションが不安定になることがあります。

'nlsparam' 引数は、日付変換の TO_CHAR ファンクションの場合と同じ用途で使用されます。

引数 *char* に DATE 値を持つ TO_DATE ファンクションは使用しないでください。戻される DATE 値は、*fmt* またはデフォルトの日付書式によって、元の *char* とは異なる世紀の値を持つことがあります。

このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。

参照： 詳細は、2-57 ページの「日時書式モデル」および 2-36 ページの「データ型の比較規則」を参照してください。

例

次の例では、文字列をタイムスタンプに変換します。

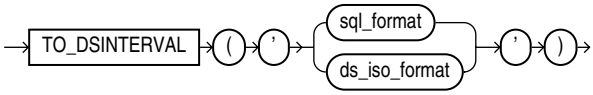
```
SELECT TO_DATE(
    'January 15, 1989, 11:00 A.M.',
    'Month dd, YYYY, HH:MI A.M.',
     'NLS_DATE_LANGUAGE = American')
     FROM DUAL;

TO_DATE('
---------
15-JAN-89
```

戻された値は、NLS_TERRITORY パラメータが 'AMERICA' に設定されている場合は、デフォルトの日付書式を反映します。異なる NLS_TERRITORY の値が設定されている場合は、異なるデフォルトの日付書式となります。

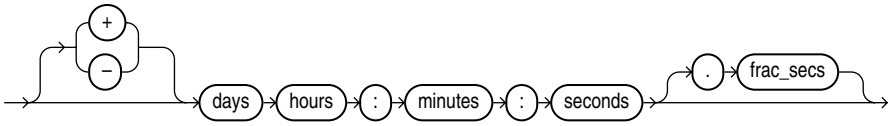
```
ALTER SESSION SET NLS_TERRITORY = 'KOREAN';

SELECT TO_DATE(
    'January 15, 1989, 11:00 A.M.',
    'Month dd, YYYY, HH:MI A.M.',
     'NLS_DATE_LANGUAGE = American')
     FROM DUAL;

TO_DATE(
--------
89/01/15
```

TO_DSINTERVAL

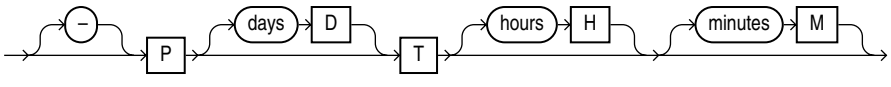
構文

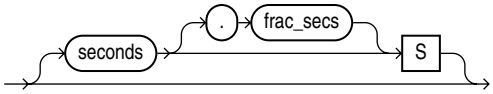


***sql_format*::=**



***ds_iso_format*::=**





用途

TO_DSINTERVAL は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型の文字列を、INTERVAL DAY TO SECOND 型に変換します。

TO_DSINTERVAL は、2 つの書式のいずれかの引数を取ります。

- SQL 規格（ISO/IEC 9075:2003）に準拠した SQL 期間書式（ISO/IEC 9075:2003）
- ISO 8601:2004 規格に準拠した ISO 存続期間書式

SQL 書式では、*days* は 0（ゼロ）～ 999999999 の整数、*hours* は 0（ゼロ）～ 23 の整数、*minutes* と *seconds* は 0（ゼロ）～ 59 の整数になります。*frac_secs* は秒の小数部であり、.0 ～ .999999999 になります。日付と時間は 1 つ以上の空白で区切ります。これ以外に、書式要素の間にも空白を使用できます。

ISO 書式では、*days*、*hours*、*minutes* および *seconds* は、0（ゼロ）～ 999999999 の整数になります。*frac_secs* は秒の小数部であり、.0 ～ .999999999 になります。値に空白は使用できません。

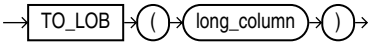
例

次の例では、1990 年 1 月 1 日から 100 日以上勤務している従業員を hr.employees 表から検索します。

```
SELECT employee_id, last_name FROM employees
   WHERE hire_date + TO_DSINTERVAL('100 00:00:00')
   <= DATE '1990-01-01'
   ORDER BY employee_id;

EMPLOYEE_ID LAST_NAME
----------- ---------------
        100 King
        101 Kochhar
        200 Whalen
```

TO_LOB

構文



用途

TO_LOB は、*long_column* 列の LONG または LONG RAW 値を LOB 値に変換します。このファンクションは LONG または LONG RAW 列に対してのみ、および INSERT 文における副問合せの SELECT 構文のリストにおいてのみ適用できます。

このファンクションを使用する前に、LOB 列を作成して変換された LONG 値を受け取る必要があります。LONG 値を変換する場合は、CLOB 列を作成します。LONG RAW 値を変換する場合は、BLOB 列を作成します。

索引構成表を作成する場合は、CREATE TABLE ... AS SELECT 文の副問合せで、TO_LOB ファンクションを使用して LONG 列を LOB 列に変換することはできません。LONG 列を含まない索引構成表を作成し、INSERT ...AS SELECT 文で TO_LOB ファンクションを使用してください。

参照：

- LONG 列を LOB に変換する別の方法については、12-2 ページの「ALTER TABLE」の「*modify_col_properties*」を参照してください。
- INSERT 文の副問合せについては、18-53 ページの「INSERT」を参照してください。

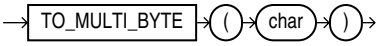
例

次の例では、不確定な表 *old_table* の LONG データに対する TO_LOB ファンクションの使用方法を示します。

```
CREATE TABLE new_table (col1, col2, ... lob_col CLOB);
INSERT INTO new_table (select o.col1, o.col2, ... TO_LOB(o.old_long_col)
   FROM old_table o;
```

TO_MULTI_BYTE

構文



用途

TO_MULTI_BYTE は、シングルバイト文字を、対応するマルチバイト文字に変換して *char* を戻します。*char* のデータ型は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 です。戻り値は、*char* と同じデータ型です。

char 内に同等のマルチバイト文字がないシングルバイト文字は、シングルバイト文字として出力されます。このファンクションは、ご使用のデータベース・キャラクタ・セットに、シングルバイト文字およびマルチバイト文字の両方が含まれている場合にのみ有効です。

このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。

参照： 詳細は、2-36 ページの「データ型の比較規則」を参照してください。

例

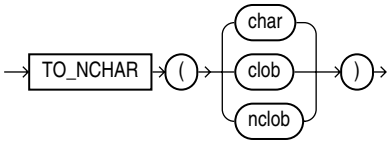
次の例では、シングルバイト A から UTF8 のマルチバイト A への変換を示します。

```
SELECT dump(TO_MULTI_BYTE( 'A')) FROM DUAL;

DUMP(TO_MULTI_BYTE('A'))
------------------------
Typ=1 Len=3: 239,188,161
```

TO_NCHAR（文字）

構文

***to_nchar_char*::=**



用途

TO_NCHAR（文字）は、文字列、CHAR、VARCHAR2、CLOB または NCLOB 値を各国語キャラクタ・セットに変換します。戻り値は常に NVARCHAR2 です。このファンクションは、各国語キャラクタ・セットで USING 句を指定した TRANSLATE ... USING ファンクションと同じです。

> **参照：** 2-39 ページの「データ変換」および 5-214 ページの「TRANSLATE ... USING」を参照してください。

例

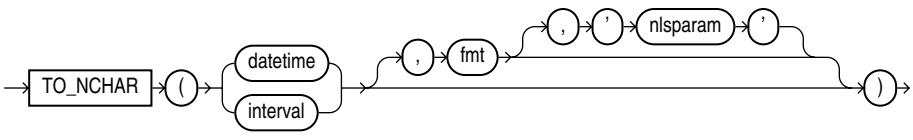
次の例では、oe.customers 表の VARCHAR2 データを各国語キャラクタ・セットに変換します。

```
SELECT TO_NCHAR(cust_last_name) FROM customers
   WHERE customer_id=103;

TO_NCHAR(CUST_LAST_NAME)
------------------------------------------------
Taylor
```

TO_NCHAR（日時）

構文

***to_nchar_date*::=**



用途

TO_NCHAR（日時）は、DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONE、TIMESTAMP WITH LOCAL TIME ZONE、INTERVAL MONTH TO YEAR または INTERVAL DAY TO SECOND データ型の文字列をデータベース・キャラクタ・セットから各国語キャラクタ・セットに変換します。

参照： 「データ変換のセキュリティ上の考慮事項」（2-43 ページ）

例

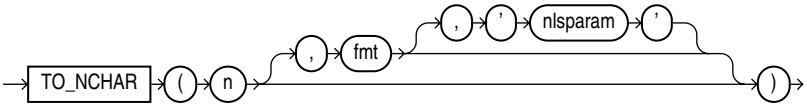
次の例では、状態が 9 であるすべての注文の order_date を各国語キャラクタ・セットに変換します。

```
SELECT TO_NCHAR(ORDER_DATE) AS order_date
   FROM ORDERS
   WHERE ORDER_STATUS > 9
   ORDER BY order_date;

ORDER_DATE
---------------------------------------------------------------------------
06-DEC-99 02.22.34.225609 PM
13-SEP-99 10.19.00.654279 AM
14-SEP-99 09.53.40.223345 AM
26-JUN-00 10.19.43.190089 PM
27-JUN-00 09.53.32.335522 PM
```

TO_NCHAR（数値）

構文

***to_nchar_number*::=**



用途

TO_NCHAR（数値）は、*n* を各国語キャラクタ・セットの文字列に変換します。*n* 値には、NUMBER、BINARY_FLOAT または BINARY_DOUBLE 型の値を指定できます。このファンクションは、引数と同じ型の値を戻します。オプションの *fmt*、*n* に対応する *'nlsparam'* は、DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONE、TIMESTAMP WITH LOCAL TIME ZONE、INTERVAL MONTH TO YEAR または INTERVAL DAY TO SECOND データ型です。

参照： 「データ変換のセキュリティ上の考慮事項」（2-43 ページ）

例

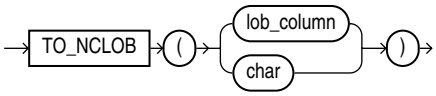
次の例では、oe.orders サンプル表の customer_id 値を各国語キャラクタ・セットに変換します。

```
SELECT TO_NCHAR(customer_id) "NCHAR_Customer_ID"  FROM orders
   WHERE order_status > 9
   ORDER BY "NCHAR_Customer_ID";

NCHAR_Customer_ID
----------------------------------------
102
103
```

```
148
148
149
```

TO_NCLOB

構文



用途

TO_NCLOB は、LOB 列またはその他の文字列の CLOB 値を NCLOB 値に変換します。*char* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。Oracle Database は、*char* のキャラクタ・セットをデータベース・キャラクタ・セットから各国語キャラクタ・セットに変換することによって、このファンクションを実行します。

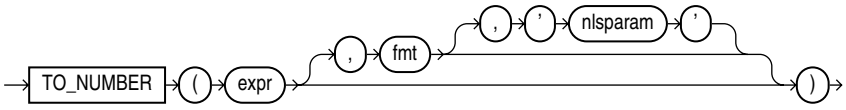
例

次の例では、TO_NCLOB ファンクションでデータを変換後、pm.print_media 表の NCLOB 列に文字データを挿入します。

```
INSERT INTO print_media (product_id, ad_id, ad_fltextn)
   VALUES (3502, 31001,
      TO_NCLOB('Placeholder for new product description'));
```

TO_NUMBER

構文



用途

TO_NUMBER は、*expr* を、NUMBER データ型の値に変換します。*expr* には、BINARY_DOUBLE 値、またはオプションの書式モデル *fmt* によって指定された書式の数値を含む CHAR、VARCHAR2、NCHAR、NVARCHAR2 データ型の値を指定できます。

BINARY_FLOAT の *expr* を指定できます。ただし、浮動小数点はその内部表示によってのみ解釈されるため、この指定は意味がありません。

このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。

> **参照：** 詳細は、2-36 ページの「データ型の比較規則」を参照してください。

例

次の例では、文字列データを数値に変換します。

```
UPDATE employees SET salary = salary +
   TO_NUMBER('100.00', '9G999D99')
   WHERE last_name = 'Perkins';
```

このファンクションの *'nlsparam'* 引数は、数値変換の TO_CHAR ファンクションの場合と同じ用途に使用されます。詳細は、5-201 ページの「TO_CHAR（数値）」を参照してください。

```
SELECT TO_NUMBER('-AusDollars100','L9G999D99',
   ' NLS_NUMERIC_CHARACTERS = ''.,''
     NLS_CURRENCY           = ''AusDollars''
   ') "Amount"
     FROM DUAL;

    Amount
----------
      -100
```

TO_SINGLE_BYTE

構文

用途

TO_SINGLE_BYTE は、マルチバイト文字を、対応するシングルバイト文字に変換して *char* を戻します。*char* のデータ型は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 です。戻り値は、*char* と同じデータ型です。

char 内に同等のシングルバイト文字がないマルチバイト文字は、マルチバイト文字として出力されます。このファンクションは、ご使用のデータベース・キャラクタ・セットに、シングルバイト文字およびマルチバイト文字の両方が含まれている場合にのみ有効です。

このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。

> **参照：** 詳細は、2-36 ページの「データ型の比較規則」を参照してください。

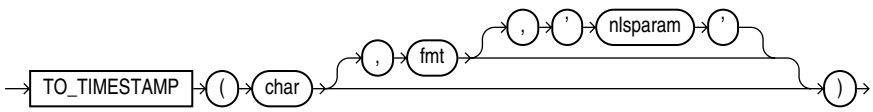
例

次の例では、UTF8 のマルチバイト A からシングルバイトの ASCII の A への変換を示します。

```
SELECT TO_SINGLE_BYTE( CHR(15711393)) FROM DUAL;

T
-
A
```

TO_TIMESTAMP

構文



用途

TO_TIMESTAMP は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型の *char* を、TIMESTAMP データ型の値に変換します。

オプションの *fmt* は、*char* の書式を指定します。*fmt* を指定しない場合、*char* はデフォルト書式の TIMESTAMP データ型である必要があります。このデータ型は、NLS_TIMESTAMP_FORMAT 初期化パラメータによって決まります。オプションの *'nlsparam'* 引数は、日付変換の TO_CHAR ファンクションの場合と同じ用途で使用されます。

このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。

> **参照：** 詳細は、2-36 ページの「データ型の比較規則」を参照してください。

例

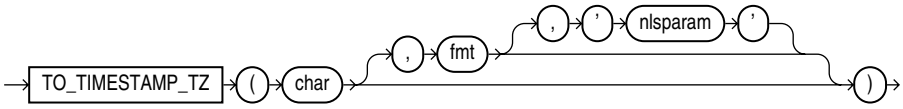
次の例では、文字列をタイムスタンプに変換します。文字列がデフォルトの TIMESTAMP 書式ではないため、書式マスクを指定する必要があります。

```
SELECT TO_TIMESTAMP ('10-Sep-02 14:10:10.123000', 'DD-Mon-RR HH24:MI:SS.FF')
   FROM DUAL;

TO_TIMESTAMP('10-SEP-0214:10:10.123000','DD-MON-RRHH24:MI:SS.FF')
---------------------------------------------------------------------------
10-SEP-02 02.10.10.123000000 PM
```

> **参照：** デフォルトの TIMESTAMP 書式の詳細は、『Oracle Database リファレンス』の NLS_TIMESTAMP_FORMAT パラメータに関する項を参照してください。書式マスクの指定の詳細は、2-57 ページの「日時書式モデル」を参照してください。

TO_TIMESTAMP_TZ

構文



用途

TO_TIMESTAMP_TZ は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型の *char* を、TIMESTAMP WITH TIME ZONE データ型の値に変換します。

> **注意：** このファンクションは、文字列を TIMESTAMP WITH LOCAL TIME ZONE に変換しません。変換するには、5-25 ページの「CAST」で示すように、CAST ファンクションを使用してください。

オプションの *fmt* は、*char* の書式を指定します。*fmt* を指定しない場合、*char* はデフォルト書式の TIMESTAMP WITH TIME ZONE データ型である必要があります。オプションの *'nlsparam'* は、日付変換の TO_CHAR ファンクションの場合と同じ用途で使用されます。

例

次の例では、文字列を TIMESTAMP WITH TIME ZONE 値に変換します。

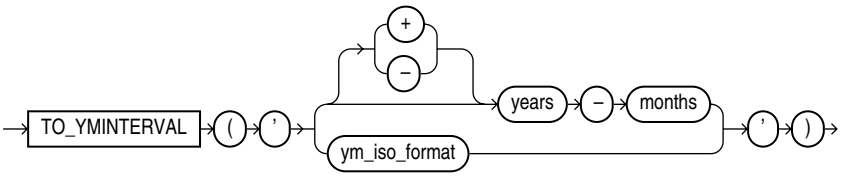
```
SELECT TO_TIMESTAMP_TZ('1999-12-01 11:00:00 -8:00',
   'YYYY-MM-DD HH:MI:SS TZH:TZM') FROM DUAL;

TO_TIMESTAMP_TZ('1999-12-0111:00:00-08:00','YYYY-MM-DDHH:MI:SSTZH:TZM')
---------------------------------------------------------------------
01-DEC-99 11.00.00.000000000 AM -08:00
```

次の例では、サンプル表 oe.order_items および oe.orders を使用して、UNION 演算の NULL 列を TIMESTAMP WITH LOCAL TIME ZONE としてキャストします。

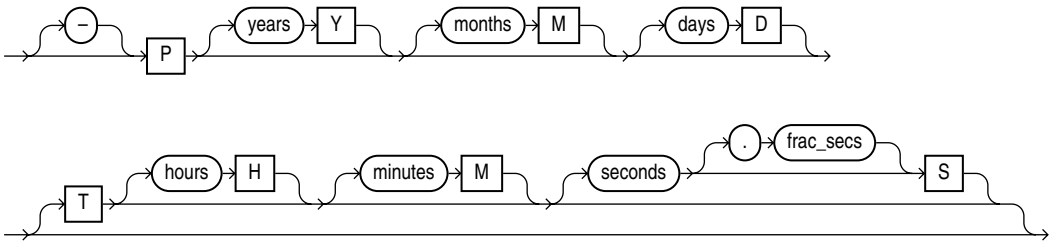
```
SELECT order_id, line_item_id,
   CAST(NULL AS TIMESTAMP WITH LOCAL TIME ZONE) order_date
   FROM order_items
UNION
SELECT order_id, to_number(null), order_date
   FROM orders;

  ORDER_ID LINE_ITEM_ID ORDER_DATE
---------- ------------ ---------------------------------
      2354            1
      2354            2
      2354            3
      2354            4
      2354            5
      2354            6
      2354            7
      2354            8
      2354            9
      2354           10
      2354           11
      2354           12
      2354           13
      2354              14-JUL-00 05.18.23.234567 PM
      2355            1
      2355            2
...
```

TO_YMINTERVAL

構文



***ym_iso_format*::=**



用途

TO_YMINTERVAL は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型の文字列を、INTERVAL YEAR TO MONTH 型に変換します。

TO_YMINTERVAL は、2 つの書式のいずれかの引数を取ります。

- SQL 規格（ISO/IEC 9075:2003）に準拠した SQL 期間書式（ISO/IEC 9075:2003）
- ISO 8601:2004 規格に準拠した ISO 存続期間書式

SQL 書式では、*years* は 0（ゼロ）～ 999999999 の整数、*months* は 0（ゼロ）～ 11 の整数になります。これ以外に、書式要素の間に空白を使用できます。

ISO 書式では、years と months は、0（ゼロ）～ 999999999 の整数になります。Days、*hours*、*minutes*、*seconds* および frac_secs は、負でない整数になります。負の整数を指定した場合は、無視されます。値に空白は使用できません。

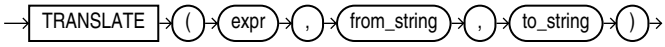
例

次の例では、サンプル表 hr.employees の各従業員の雇用された後から 1 年と 2 か月後の日付を計算します。

```
SELECT hire_date, hire_date + TO_YMINTERVAL('01-02') "14 months"
   FROM employees;

HIRE_DATE 14 months
--------- ---------
17-JUN-87 17-AUG-88
21-SEP-89 21-NOV-90
13-JAN-93 13-MAR-94
03-JAN-90 03-MAR-91
21-MAY-91 21-JUL-92
. . .
```

TRANSLATE

構文



用途

TRANSLATE は、*from_string* 内のすべての文字を *to_string* 内の対応する文字に置換して *expr* を戻します。*from_string* 内に存在しない *expr* 内の文字は置換されません。引数 *from_string* には、*to_string* より多い文字を指定できます。この場合、*from_string* の終わりにある余分な文字には、*to_string* 内に対応する文字がありません。これらの余分な文字が *expr* 内にある場合、それらの文字は戻り値から削除されます。

from_string 内に 1 つの文字が複数回現れた場合は、最初の出現に対応する *to_string* マッピングが使用されます。

戻り値から *from_string* 内のすべての文字を削除するために、*to_string* に空の文字列を使用することはできません。Oracle Database は空の文字列を NULL と解析するため、このファンクションに NULL の引数がある場合、NULL が戻されます。*from_string* 内の文字をすべて削除するには、*from_string* の先頭に別の文字を結合し、この文字を *to_string* として指定します。たとえば、TRANSLATE(*expr*,'x0123456789','x') では、すべての数字が *expr* から削除されます。

TRANSLATE は、REPLACE ファンクションに関連する機能を提供します。REPLACE ファンクションでは、単一の文字列から別の単一の文字列への置換、および文字列の削除を実行できます。TRANSLATE では、1 回の操作で複数の単一文字を 1 対 1 で置き換えることができます。

このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。

> **参照：** 詳細は、2-36 ページの「データ型の比較規則」を参照してください。5-155 ページの「REPLACE」も参照してください。

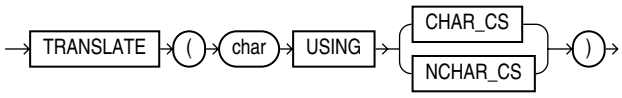
例

次の文は、本のタイトルを、ファイル名などに使用するために文字列に変換します。*from_string* には、空白、アスタリスク、スラッシュおよびアポストロフィの 4 つの文字（およびエスケープ文字のアポストロフィ）が含まれています。*to_string* には、3 つのアンダースコアのみが含まれています。*from_string* の 4 つ目の文字に対応する置換文字がないため、戻り値ではアポストロフィが削除されています。

```
SELECT TRANSLATE('SQL*Plus User''s Guide', ' */''', '___') FROM DUAL;

TRANSLATE('SQL*PLUSU
--------------------
SQL_Plus_Users_Guide
```

TRANSLATE ... USING

構文



用途

TRANSLATE ... USING は、*char* をデータベース・キャラクタ・セットと各国語キャラクタ・セット間の変換に指定されたキャラクタ・セットに変換します。

> **注意：** TRANSLATE ... USING は、主に ANSI との互換性のためにサポートされているファンクションです。TO_CHAR および TO_NCHAR ファンクションを使用してデータをデータベース・キャラクタ・セットまたは各国語キャラクタ・セットに適切に変換することをお薦めします。TO_CHAR および TO_NCHAR は、文字データのみを受け入れる TRANSLATE ... USING に比べ、より多くのデータ型を引数として取ることができます。

引数 *char* は変換する式です。

- USING CHAR_CS 引数を指定すると、*char* がデータベース・キャラクタ・セットに変換されます。出力データ型は VARCHAR2 です。
- USING NCHAR_CS 引数を指定すると、*char* が各国語キャラクタ・セットに変換されます。出力データ型は NVARCHAR2 です。

このファンクションは、Oracle の CONVERT ファンクションに似ていますが、入力または出力のデータ型に NCHAR または NVARCHAR2 を使用する場合は、CONVERT ではなくこのファンクションを使用する必要があります。入力に UCS2 コードポイントまたはバックスラッシュ（\）が含まれる場合は、UNISTR ファンクションを使用してください。

> **参照：** 5-38 ページの「CONVERT」および 5-219 ページの「UNISTR」を参照してください。

例

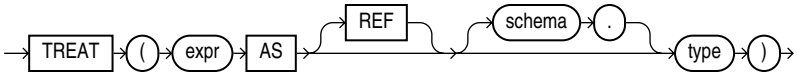
次の文は、サンプル表 oe.product_descriptions のデータを使用して、TRANSLATE ... USING ファンクションの使用方法を示しています。

```
CREATE TABLE translate_tab (char_col  VARCHAR2(100),
                            nchar_col NVARCHAR2(50));
INSERT INTO translate_tab
   SELECT NULL, translated_name
      FROM product_descriptions
      WHERE product_id = 3501;
```

```
SELECT * FROM translate_tab;

CHAR_COL             NCHAR_COL
-------------------- --------------------------------------------------
. . .
                     C pre SPNIX4.0 - Sys
                     C pro SPNIX4.0 - Sys
                     C til SPNIX4.0 - Sys
                     C voor SPNIX4.0 - Sys
. . .

UPDATE translate_tab
   SET char_col = TRANSLATE (nchar_col USING CHAR_CS);

SELECT * FROM translate_tab;

CHAR_COL                 NCHAR_COL
------------------------ ------------------------
. . .
C per a SPNIX4.0 - Sys   C per a SPNIX4.0 - Sys
C pro SPNIX4.0 - Sys     C pro SPNIX4.0 - Sys
C for SPNIX4.0 - Sys     C for SPNIX4.0 - Sys
C til SPNIX4.0 - Sys     C til SPNIX4.0 - Sys
. . .
```

TREAT

構文



用途

TREAT は式の宣言型を変更します。

このファンクションを使用する場合は、*type* に対する EXECUTE オブジェクト権限が必要です。

- *type* は、スーパータイプまたは *expr* の宣言型のサブタイプである必要があります。*expr* の最も指定される型が *type*（または *type* のサブタイプ）である場合、TREAT は *expr* を戻します。*expr* の最も指定される型が *type*（または *type* のサブタイプ）ではない場合、TREAT は NULL を戻します。
- *expr* の宣言型が REF 型の場合にのみ、REF を指定できます。
- *expr* の宣言型が *expr* のソース型への REF である場合、*type* はサブタイプまたは *expr* のソース型のスーパータイプである必要があります。DEREF(*expr*) の最も指定される型が *type*（または *type* のサブタイプ）である場合、TREAT は *expr* を戻します。DEREF(*expr*) の最も指定される型が *type*（または *type* のサブタイプ）ではない場合、TREAT は NULL を戻します。

このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。

> **参照：** 詳細は、2-36 ページの「データ型の比較規則」を参照してください。

例

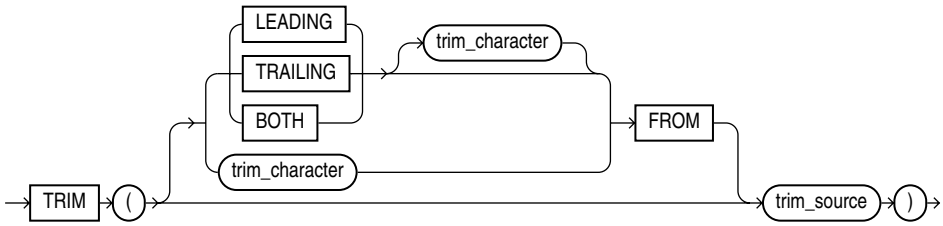
次の例では、16-61 ページの「置換可能な表および列のサンプル :」で作成された表 oe.persons を使用します。次の例では、persons 表のすべての人物の給与属性を検索します。従業員ではない人物のインスタンスの値は NULL です。

```
SELECT name, TREAT(VALUE(p) AS employee_t).salary salary
   FROM persons p;

NAME                          SALARY
----------------------- ----------
Bob
Joe                           100000
Tim                             1000
```

TREAT ファンクションを使用すると、置換可能列のサブタイプ属性に索引を作成できます。14-72 ページの「置換可能な列の索引の作成例 :」を参照してください。

TRIM

構文



用途

TRIM によって、文字列の先行または後続文字（またはその両方）を切り捨てることができます。*trim_character* または *trim_source* が文字リテラルの場合、一重引用符で囲む必要があります。

- LEADING を指定すると、Oracle Database は *trim_character* と等しい先行文字を削除します。
- TRAILING を指定すると、Oracle は *trim_character* と等しい後続文字を削除します。
- BOTH を指定するか、またはいずれも指定しない場合、Oracle は *trim_character* と等しい先行および後続文字を削除します。
- *trim_character* を指定しないと、デフォルト値は空白になります。
- *trim_source* のみを指定すると、Oracle は先行および後続空白を削除します。
- このファンクションは、値を VARCHAR2 データ型で戻します。値の最大長は、*trim_source* の長さです。
- *trim_source* または *trim_character* のいずれかが NULL の場合、TRIM は NULL を戻します。

trim_character と *trim_source* は両方とも、VARCHAR2 または VARCHAR2 に暗黙的に変換できる任意のデータ型です。戻される文字列は、*trim_source* が文字データ型の場合は VARCHAR2 データ型になり、*trim_source* が LOB データ型の場合は LOB になります。*trim_source* と同じキャラクタ・セットの文字列が戻されます。

例

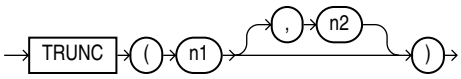
次の例では、hr スキーマの従業員の雇用日から先行 0（ゼロ）を切り捨てます。

```
SELECT employee_id,
      TO_CHAR(TRIM(LEADING 0 FROM hire_date))
      FROM employees
      WHERE department_id = 60
      ORDER BY employee_id;

EMPLOYEE_ID TO_CHAR(T
----------- ---------
        103 3-JAN-90
        104 21-MAY-91
        105 25-JUN-97
        106 5-FEB-98
        107 7-FEB-99
```

TRUNC（数値）

構文

***trunc_number*::=**



用途

TRUNC（数値）ファンクションは、*n1* を小数第 *n2* 位までに切り捨てた値を戻します。*n2* を指定しない場合、*n1* の小数点以下を切り捨てます。*n2* が負の場合は、小数点の左 *n2* 桁を切り捨てて、0（ゼロ）にします。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。*n2* を指定しない場合、このファンクションは、引数の数値データ型と同じデータ型を戻します。*n2* を指定すると、このファンクションは NUMBER を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

例

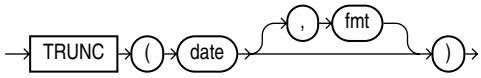
次の例では、数値を切り捨てます。

```
SELECT TRUNC(15.79,1) "Truncate" FROM DUAL;

  Truncate
----------
      15.7

SELECT TRUNC(15.79,-1) "Truncate" FROM DUAL;

  Truncate
----------
        10
```

TRUNC（日付）

構文

***trunc_date*::=**



用途

TRUNC（日付）ファンクションは、時刻部分を書式モデル *fmt* で指定された単位まで切り捨てた *date* を戻します。このファンクションは、NLS_CALENDAR セッション・パラメータの影響を受けません。このファンクションはグレゴリオ暦の規則に従って動作します。戻される値は、*date* に異なる日時データ型を指定した場合でも、常に DATE データ型です。*fmt* を省略すると、*date* は最も近い日に切り捨てられます。*fmt* で使用できる書式モデルについては、5-248 ページの「ROUND および TRUNC 日付ファンクション」を参照してください。

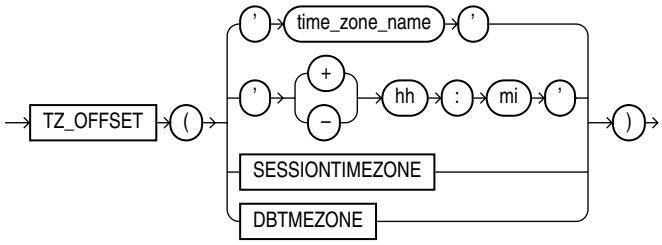
例

次の例では、日付を切り捨てます。

```
SELECT TRUNC(TO_DATE('27-OCT-92','DD-MON-YY'), 'YEAR')
  "New Year" FROM DUAL;

New Year
---------
01-JAN-92
```

TZ_OFFSET

構文



用途

TZ_OFFSET は、文が実行された日付に基づいた引数に対応するタイムゾーン・オフセットを戻します。有効なタイムゾーン名、UTC からのタイムゾーン・オフセット（それ自体を戻します）、またはキーワード SESSIONTIMEZONE または DBTIMEZONE を入力できます。*time_zone_name* の有効な値を表示するには、V$TIMEZONE_NAMES 動的パフォーマンス・ビューの TZNAME 列を問い合せます。

注意： 夏時間機能には、タイムゾーン地域名が必要です。地域名は、2 つのタイムゾーン・ファイルに格納されます。デフォルトのタイムゾーン・ファイルは、パフォーマンスを最大にするために一般的なタイムゾーンのみの小さなファイルです。タイムゾーンがデフォルトのファイルに存在しない場合は、環境変数 ORA_TZFILE を使用して完全な（大きい）ファイルへのパスを指定するまで、夏時間はサポートされません。

例

次の例では、UTC から US/ 東部タイムゾーンのタイムゾーン・オフセットを戻します。

```
SELECT TZ_OFFSET('US/Eastern') FROM DUAL;

TZ_OFFS
-------
-04:00
```

UID

構文

用途

UID は、セッション・ユーザー（ログインしているユーザー）を一意に識別する整数を戻します。

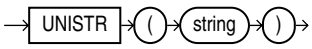
例

次の例では、現行のユーザーの UID を戻します。

```
SELECT UID FROM DUAL;
```

UNISTR

構文



用途

UNISTR は、引数としてテキスト・リテラルまたは文字データに変換する式を取り、各国語キャラクタ・セットで戻します。データベースの各国語キャラクタ・セットは、AL16UTF16 または UTF8 です。UNISTR では、文字列への文字の Unicode エンコーディング値の指定を可能にすることによって、Unicode 文字列リテラルがサポートされます。これは、NCHAR 列にデータを挿入する場合などに有効です。

Unicode エンコーディング値は、\xxxx という形式です。xxxx は、文字の UCS-2 エンコーディング形式での 16 進数値です。付加される文字は 2 つのコード単位としてエンコードされます。1 つ目は高サロゲート範囲（U+D800 ～ U+DBFF）からエンコードされ、2 つ目は低サロゲート範囲（U+DC00 ～ U+DFFF）からエンコードされます。バックスラッシュ自体を文字列に含めるには、バックスラッシュをもう 1 つ追加します（\\）。

移植性およびデータ保護のために、UNISTR 文字列の引数には ASCII 文字および Unicode エンコーディング値のみを指定することをお薦めします。

> **参照：** Unicode および各国語キャラクタ・セットの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

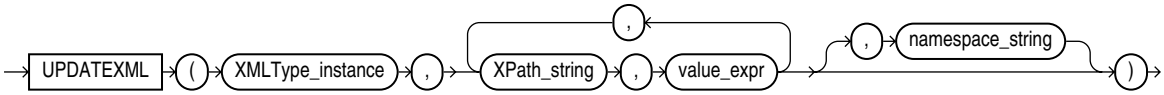
例

次の例では、ASCII 文字および Unicode エンコーディング値を UNISTR ファンクションに渡した後、その UNISTR ファンクションによって各国語キャラクタ・セットで文字列が戻されます。

```
SELECT UNISTR('abc\00e5\00f1\00f6') FROM DUAL;

UNISTR
------
abcåñö
```

UPDATEXML

構文



用途

UPDATEXML は、引数として XMLType インスタンスおよび XPath 値の組を取り、更新された値を持つ XMLType インスタンスを戻します。*XPath_string* が XML 要素の場合、対応する *value_expr* は XMLType インスタンスです。*XPath_string* が属性またはテキスト・ノードの場合、*value_expr* は任意のスカラー・データ型となります。先頭にスラッシュを付けて絶対 *XPath_string* を指定したり、先頭のスラッシュを省略して相対 *XPath_string* を指定できます。先頭のスラッシュを省略した場合、相対パスのコンテキストは、デフォルトでルート・ノードに設定されます。

各 *XPath_string* およびこれに対応する *value_expr* のターゲットのデータ型は一致する必要があります。オプションの *namespace_string* は、接頭辞のデフォルト・マッピングまたはネームスペース・マッピング（Oracle Database が XPath 式を評価する場合に使用）を指定する VARCHAR2 値に解決される必要があります。

XML 要素を更新して NULL にした場合、属性および子要素が削除されるため、要素は空になります。要素のテキスト・ノードを更新して NULL にした場合、その要素のテキスト値が削除されるため、要素自体は残りますが空になります。

多くの場合、このファンクションはメモリー内の XML 文書を実体化して、その値を更新します。ただし、UPDATEXML は、直接オブジェクト・リレーショナル列の値を更新するように、その列での UPDATE 文に対して最適化されます。この最適化には、次の条件があります。

- *XMLType_instance* は、UPDATE ... SET 句の列と同じである必要があります。
- *XPath_string* は、スカラー・コンテンツとなる必要があります。

例

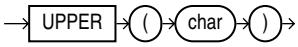
次の例では、サンプル・スキーマ OE（XMLType 型の warehouse_spec 列を持つ）にある San Francisco 倉庫のドック数を 4 に更新します。

```
SELECT warehouse_name,
   EXTRACT(warehouse_spec, '/Warehouse/Docks')
   "Number of Docks"
   FROM warehouses
   WHERE warehouse_name = 'San Francisco';

WAREHOUSE_NAME       Number of Docks
-------------------- --------------------
San Francisco        <Docks>1</Docks>

UPDATE warehouses SET warehouse_spec =
```

```
   UPDATEXML(warehouse_spec,
   '/Warehouse/Docks/text()',4)
   WHERE warehouse_name = 'San Francisco';

1 row updated.

SELECT warehouse_name,
   EXTRACT(warehouse_spec, '/Warehouse/Docks')
   "Number of Docks"
   FROM warehouses
   WHERE warehouse_name = 'San Francisco';

WAREHOUSE_NAME       Number of Docks
-------------------- --------------------
San Francisco        <Docks>4</Docks>
```

UPPER

構文



用途

UPPER は、すべての文字を大文字にして *char* を戻します。*char* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻り値は、*char* と同じデータ型です。データベースは、基礎となるキャラクタ・セットに対して定義したバイナリ・マッピングに基づいて文字の形式を設定します。大文字の区別については、5-111 ページの「NLS_UPPER」を参照してください。

例

次の例では、各従業員の姓を大文字で戻します。

```
SELECT UPPER(last_name) "Uppercase"
   FROM employees;
```

USER

構文

用途

USER は、セッション・ユーザー（ログインしているユーザー）の名前を VARCHAR2 データ型で戻します。Oracle Database は、空白埋め比較セマンティクスでこのファンクションの値を比較します。

分散 SQL 文では、UID ファンクションおよび USER ファンクションは、ローカル・データベース上のユーザーを識別します。CHECK 制約の条件でこれらのファンクションは使用できません。

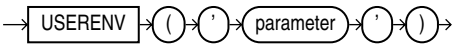
例

次の例では、現行のユーザーおよびユーザーの UID を戻します。

```
SELECT USER, UID FROM DUAL;
```

USERENV

構文



用途

> **注意：** USERENV は、下位互換用に保持されるレガシー・ファンクションです。現行の機能に対して組込み USERENV ネームスペースとともに SYS_CONTEXT ファンクションを使用することをお薦めします。詳細は、5-184 ページの「SYS_CONTEXT」を参照してください。

USERENV は現行のセッションについての情報を戻します。この情報は、アプリケーション固有の監査証跡表を書き込む場合、またはセッションで現在使用されている言語固有の文字を判断する場合に有効です。CHECK 制約の条件で、USERENV は使用できません。表 5-13 に、*parameter* 引数の値を示します。

SESSIONID および ENTRYID パラメータを使用するコール（NUMBER を戻す）以外の USERENV へのすべてのコールは、VARCHAR2 データを戻します。

表 5-13 USERENV ファンクションのパラメータ

パラメータ	戻り値
CLIENT_INFO	DBMS_APPLICATION_INFO パッケージを使用するアプリケーションが格納できる 64 バイトまでのユーザー・セッション情報を戻します。 **注意：**商業用のアプリケーションによっては、このコンテキスト値を使用する可能性があります。このコンテキスト領域の使用に対する制限については、これらのアプリケーションのドキュメントを参照してください。 **参照：** ■ アプリケーション・コンテキストの詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。 ■ 14-8 ページの「CREATE CONTEXT」および 5-184 ページの「SYS_CONTEXT」を参照してください。
ENTRYID	現行の監査エントリ番号を戻します。監査エントリ ID の順序は、ファイングレイン監査レコードと通常の監査レコードで共通です。この属性を分散 SQL 文で使用することはできません。
ISDBA	ISDBA は、オペレーティング・システムまたはパスワード・ファイルによって、ユーザーが DBA 権限を持っていると認証された場合、'TRUE' を戻します。
LANG	言語名の ISO 略称を戻します。これは、既存の 'LANGUAGE' パラメータを短縮したものです。
LANGUAGE	現行のセッションで使用している言語（language）および地域（territory）を、データベース・キャラクタ・セット（character set）も含めた次の書式で戻します。 language_territory.characterset
SESSIONID	監査セッション識別子を戻します。この属性を分散 SQL 文で使用することはできません。
TERMINAL	現行のセッションの端末に対するオペレーティング・システム識別子を戻します。分散 SQL 文では、このパラメータはローカル・セッションの識別子を戻します。分散環境では、リモートの SELECT に対してのみこのパラメータを使用でき、リモートの INSERT、UPDATE または DELETE には使用できません。

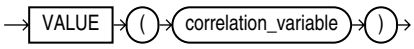
例

次の例では、現行のセッションの LANGUAGE パラメータを戻します。

```
SELECT USERENV('LANGUAGE') "Language" FROM DUAL;

Language
----------------------------------
AMERICAN_AMERICA.WE8ISO8859P1
```

VALUE

構文



用途

VALUE は、引数としてオブジェクト表の行に関連付けられた相関変数（表別名）を取り、オブジェクト表に格納されたオブジェクト・インスタンスを戻します。オブジェクト・インスタンスの型は、オブジェクト表と同じ型です。

例

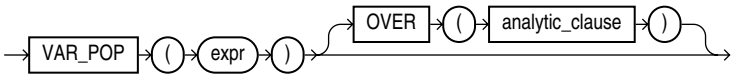
次の例では、16-61 ページの「置換可能な表および列のサンプル :」で作成された oe.persons サンプル表を使用します。

```
SELECT VALUE(p) FROM persons p;

VALUE(P)(NAME, SSN)
-----------------------------------------------------------
PERSON_T('Bob', 1234)
EMPLOYEE_T('Joe', 32456, 12, 100000)
PART_TIME_EMP_T('Tim', 5678, 13, 1000, 20)
```

> **参照 :** VALUE ファンクションでの IS OF 型の使用方法については、7-23 ページの「IS OF type 条件」を参照してください。

VAR_POP

構文



> **参照 :** 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

VAR_POP は、数値の集合にある NULL を削除した後、この集合の母集団分散を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を戻します。

> **参照 :** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

ファンクションが空の集合に適用されると、NULL を戻します。このファンクションは、次の計算を行います。

```
(SUM(expr2) - SUM(expr)2 / COUNT(expr)) / COUNT(expr)
```

> **参照：** *expr* の書式の詳細は、6-2 ページの「SQL 式」を参照してください。5-8 ページの「集計ファンクション」も参照してください。

集計の例

次の例では、employees 表にある給与の人口分散を計算します。

```
SELECT VAR_POP(salary) FROM employees;

VAR_POP(SALARY)
---------------
     15140307.5
```

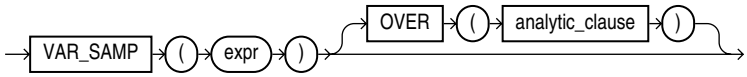
分析の例

次の例では、sh.sales 表での 1998 年における月ごとの売上について、累積の母集団および標本分散を計算します。

```
SELECT t.calendar_month_desc,
   VAR_POP(SUM(s.amount_sold))
      OVER (ORDER BY t.calendar_month_desc) "Var_Pop",
   VAR_SAMP(SUM(s.amount_sold))
      OVER (ORDER BY t.calendar_month_desc) "Var_Samp"
  FROM sales s, times t
  WHERE s.time_id = t.time_id AND t.calendar_year = 1998
  GROUP BY t.calendar_month_desc
  ORDER BY t.calendar_month_desc, "Var_Pop", "Var_Samp";

CALENDAR    Var_Pop   Var_Samp
-------- ---------- ----------
1998-01           0
1998-02  2269111326 4538222653
1998-03  5.5849E+10 8.3774E+10
1998-04  4.8252E+10 6.4336E+10
1998-05  6.0020E+10 7.5025E+10
1998-06  5.4091E+10 6.4909E+10
1998-07  4.7150E+10 5.5009E+10
1998-08  4.1345E+10 4.7252E+10
1998-09  3.9591E+10 4.4540E+10
1998-10  3.9995E+10 4.4439E+10
1998-11  3.6870E+10 4.0558E+10
1998-12  4.0216E+10 4.3872E+10
```

VAR_SAMP

構文



> **参照：** 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

VAR_SAMP は、数値の集合にある NULL を削除した後、この集合の標本分散を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

ファンクションが空の集合に適用されると、NULL を戻します。このファンクションは、次の計算を行います。

```
(SUM(expr2) - SUM(expr)2 / COUNT(expr)) / (COUNT(expr) - 1)
```

このファンクションは、1 つの要素の集合を入力すると VARIANCE は 0 を戻し、VAR_SAMP は NULL を戻すという点を除いては、VARIANCE に似ています。

> **参照：** *expr* の書式の詳細は、6-2 ページの「SQL 式」を参照してください。5-8 ページの「集計ファンクション」も参照してください。

集計の例

次の例では、サンプル表 employees にある給与の標本分散を計算します。

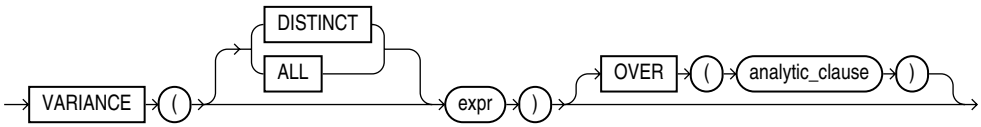
```
SELECT VAR_SAMP(salary) FROM employees;

VAR_SAMP(SALARY)
----------------
      15283140.5
```

分析の例

5-223 ページの「VAR_POP」の分析の例を参照してください。

VARIANCE

構文



> **参照：** 構文、セマンティクスおよび制限事項の詳細は、5-10 ページの「分析ファンクション」を参照してください。

用途

VARIANCE は *expr* の分散を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

expr の分散の計算結果は、次のようになります。

- *expr* の行数が 1 の場合は 0（ゼロ）
- *expr* の行数が 1 を超える場合は VAR_SAMP

DISTINCT を指定する場合は、*analytic_clause* の *query_partition_clause* のみ指定できます。*order_by_clause* および *windowing_clause* は指定できません。

このファンクションは、引数として、任意の数値データ型、または暗黙的に数値データ型に変換可能な数値以外のデータ型を取ります。また、引数の数値データ型と同じデータ型を戻します。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。*expr* の書式の詳細は、6-2 ページの「SQL 式」を参照してください。5-8 ページの「集計ファンクション」も参照してください。

集計の例

次の例では、サンプル表 employees にあるすべての給与の合計を計算します。

```
SELECT VARIANCE(salary) "Variance"
   FROM employees;

  Variance
----------
15283140.5
```

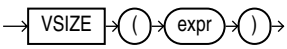
分析の例

次の例では、雇用日で順序付けられた部門 30 の給与の値の累積分散を戻します。

```
SELECT last_name, salary, VARIANCE(salary)
      OVER (ORDER BY hire_date) "Variance"
   FROM employees
   WHERE department_id = 30
   ORDER BY last_name, salary, "Variance";

LAST_NAME                     SALARY   Variance
------------------------- ---------- ----------
Baida                           2900 16283333.3
Colmenares                      2500   11307000
Himuro                          2600   13317000
Khoo                            3100   31205000
Raphaely                       11000          0
Tobias                          2800 21623333.3
```

VSIZE

構文



用途

VSIZE は、*expr* の内部表現でのバイト数を戻します。*expr* が NULL の場合は NULL を戻します。

このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。

> **参照：** 詳細は、2-36 ページの「データ型の比較規則」を参照してください。

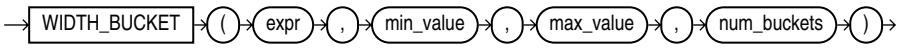
例

次の例では、部門 10 の従業員の last_name 列のバイト数を戻します。

```
SELECT last_name, VSIZE (last_name) "BYTES"
  FROM employees
  WHERE department_id = 10
  ORDER BY employee_id;

LAST_NAME            BYTES
--------------- ----------
Whalen                    6
```

WIDTH_BUCKET

構文



用途

WIDTH_BUCKET を使用すると、ヒストグラムの幅が同じサイズに分割された等幅ヒストグラムを作成できます。このファンクションと、等高ヒストグラムを作成する NTILE を比較してください。各バケットが実際の数値線幅の closed-open 間隔であることが理想です。たとえば、間隔に 10 が含まれ 20 が排除されることを示すために、バケットは 10.00 ～ 19.999... のスコアに割り当てられます。これは、[10, 20) と表すこともできます。

指定された式に対して、WIDTH_BUCKET は、この式の値が評価された後に該当するバケット数を戻します。

- *expr* は、ヒストグラムが作成される式です。この式は、数値または日時値、あるいは数値または日時値に暗黙的に変換可能な値に評価される必要があります。*expr* が NULL と評価された場合、式は NULL を戻します。
- *min_value* および *max_value* は、*expr* の許容域のエンド・ポイントに解決される式です。これらの式は両方とも数値または日時値に評価される必要があり、いずれも NULL に評価されません。
- *num_buckets* は、バケット数を示す定数に解決される式です。この式は、正の整数に評価される必要があります。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。

Oracle Database は、必要に応じて、0（ゼロ）の下位バケットおよび *num_buckets*+1 の上位バケットを作成します。これらのバケットは、*min_value* 未満および *max_value* より大きい値を処理し、エンド・ポイントの妥当性チェックに有効です。

例

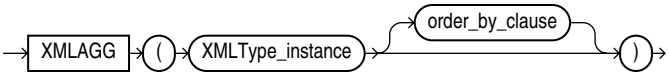
次の例では、サンプル表 oe.customers のスイスの顧客の credit_limit 列に 10 バケットのヒストグラムを作成し、各顧客のバケット数（「クレジット・グループ」）を戻します。最大値を超えるクレジット利用限度額を持つ顧客は、上位バケット 11 に割り当てられます。

```
SELECT customer_id, cust_last_name, credit_limit,
   WIDTH_BUCKET(credit_limit, 100, 5000, 10) "Credit Group"
   FROM customers WHERE nls_territory = 'SWITZERLAND'
   ORDER BY "Credit Group", customer_id, cust_last_name, credit_limit;

CUSTOMER_ID CUST_LAST_NAME       CREDIT_LIMIT Credit Group
----------- -------------------- ------------ ------------
        825 Dreyfuss                      500            1
```

```
826 Barkin                     500          1
827 Siegel                     500          1
853 Palin                      400          1
843 Oates                      700          2
844 Julius                     700          2
835 Eastwood                  1200          3
836 Berenger                  1200          3
837 Stanton                   1200          3
840 Elliott                   1400          3
841 Boyer                     1400          3
842 Stern                     1400          3
848 Olmos                     1800          4
849 Kaurusmdki                1800          4
828 Minnelli                  2300          5
829 Hunter                    2300          5
850 Finney                    2300          5
851 Brown                     2300          5
852 Tanner                    2300          5
830 Dutt                      3500          7
831 Bel Geddes                3500          7
832 Spacek                    3500          7
833 Moranis                   3500          7
834 Idle                      3500          7
838 Nicholson                 3500          7
839 Johnson                   3500          7
845 Fawcett                   5000         11
846 Brando                    5000         11
847 Streep                    5000         11
```

XMLAGG

構文



用途

XMLAGG は集計ファンクションです。XML フラグメントのコレクションを取り、集計された XML 文書を戻します。NULL を戻す引数は結果から排除されます。

XMLAGG は、ノードのコレクションを戻す点を除いて、SYS_XMLAGG に似ていますが、XMLFormat オブジェクトを使用した書式設定は受け入れません。また、XMLAGG は、SYS_XMLAGG とは異なり、出力を要素タグで囲みません。

この *order_by_clause* にかぎり、他の使用方法とは異なり、数値リテラルは列の位置として認識されず、単に数値リテラルとして解釈されます。

> **参照：** 5-234 ページの「XMLELEMENT」および 5-192 ページの「SYS_XMLAGG」を参照してください。

例

次の例では、従業員のジョブ ID と姓を要素の内容とする Employee 要素を含む Department 要素を生成します。

```
SELECT XMLELEMENT("Department",
   XMLAGG(XMLELEMENT("Employee",
   e.job_id||' '||e.last_name)
   ORDER BY last_name))
   as "Dept_list"
```

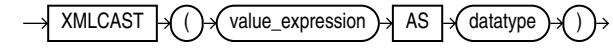
```
   FROM employees e
   WHERE e.department_id = 30;

Dept_list
------------------------------------------------------------
<Department>
  <Employee>PU_CLERK Baida</Employee>
  <Employee>PU_CLERK Colmenares</Employee>
  <Employee>PU_CLERK Himuro</Employee>
  <Employee>PU_CLERK Khoo</Employee>
  <Employee>PU_MAN Raphaely</Employee>
  <Employee>PU_CLERK Tobias</Employee>
</Department>
```

XMLAGG が行を集計するため、結果は単一行となります。GROUP BY 句を使用して、戻された行の集合を複数のグループにまとめることができます。

```
SELECT XMLELEMENT("Department",
      XMLAGG(XMLELEMENT("Employee", e.job_id||' '||e.last_name)))
   AS "Dept_list"
   FROM employees e
   GROUP BY e.department_id;

Dept_list
------------------------------------------------------------
<Department>
  <Employee>AD_ASST Whalen</Employee>
</Department>

<Department>
  <Employee>MK_MAN Hartstein</Employee>
  <Employee>MK_REP Fay</Employee>
</Department>

<Department>
  <Employee>PU_MAN Raphaely</Employee>
  <Employee>PU_CLERK Khoo</Employee>
  <Employee>PU_CLERK Tobias</Employee>
  <Employee>PU_CLERK Baida</Employee>
  <Employee>PU_CLERK Colmenares</Employee>
  <Employee>PU_CLERK Himuro</Employee>
</Department>
. . .
```

XMLCAST

構文



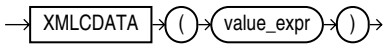
用途

XMLCast は、*value_expression* を *datatype* で指定されたスカラー SQL データ型にキャストします。*value_expression* 引数は、評価される SQL 式です。*datatype* 引数のデータ型は、NUMBER、VARCHAR2、および任意の日時データ型です。

> **参照：** このファンクションの使用の詳細および例は、『Oracle XML DB 開発者ガイド』を参照してください。

XMLCDATA

構文



用途

XMLCData は、*value_expr* を評価して CDATA セクションを生成します。*value_expr* は文字列に変換する必要があります。このファンクションの戻り値は、次の書式になります。

```
<![CDATA[string]]>
```

結果値が無効な XML CDATA セクションの場合は、エラーが戻されます。

次の条件が XMLCData に適用されます。

- *value_expr* には、サブストリング]]> を含めることはできません。
- *value_expr* が NULL と評価された場合、このファンクションは NULL を戻します。

> **参照：** このファンクションの詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

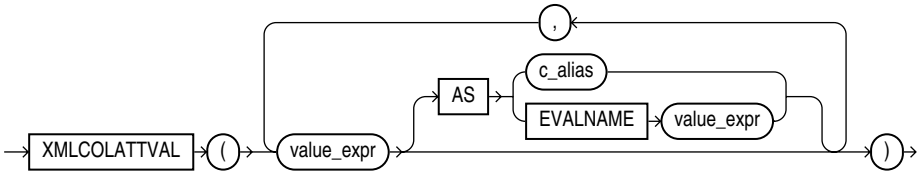
例

次の文は、DUAL 表を使用して、XMLCData の構文を示します。

```
SELECT XMLELEMENT("PurchaseOrder",
   XMLAttributes(dummy as "pono"),
   XMLCdata('<!DOCTYPE po_dom_group [
   <!ELEMENT po_dom_group(student_name)*>
   <!ELEMENT po_purch_name (#PCDATA)>
   <!ATTLIST po_name po_no ID #REQUIRED>
   <!ATTLIST po_name trust_1 IDREF #IMPLIED>
   <!ATTLIST po_name trust_2 IDREF #IMPLIED>
   ]>')) "XMLCData" FROM DUAL;

XMLCData
------------------------------------------------------------
<PurchaseOrder pono="X"><![CDATA[
<!DOCTYPE po_dom_group [
   <!ELEMENT po_dom_group(student_name)*>
   <!ELEMENT po_purch_name (#PCDATA)>
   <!ATTLIST po_name po_no ID #REQUIRED>
   <!ATTLIST po_name trust_1 IDREF #IMPLIED>
   <!ATTLIST po_name trust_2 IDREF #IMPLIED>
   ]>
  ]]>
</PurchaseOrder>
```

XMLCOLATTVAL

構文



用途

XMLCOLATTVAL は、XML フラグメントを作成し、各 XML フラグメントの名前が column、属性が name となるように、結果としてできた XML を展開します。

AS 句を使用して、name 属性の値を列名以外の値に変更できます。この場合、文字列リテラルの *c_alias* を指定するか、または EVALNAME *value_expr* を指定します。後者の場合は、値の式が評価され、その結果（文字列リテラル）が別名として使用されます。別名は、最大 4000 文字まで指定可能です。

value_expr の値を指定する必要があります。*value_expr* が NULL の場合、要素は戻されません。

XMLColAttVal の制限事項： *value_expr* にオブジェクト型の列を指定することはできません。

例

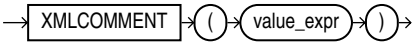
次の例では、従業員のサブセットに対して Emp 要素を作成します。このとき、Emp の内容として、employee_id、last_name および salary の各要素がネストされます。ネストされた各要素には column という名前が付き、属性値として列名を持つ name 属性が与えられます。

```
SELECT XMLELEMENT("Emp",
   XMLCOLATTVAL(e.employee_id, e.last_name, e.salary)) "Emp Element"
   FROM employees e
   WHERE employee_id = 204;

Emp Element
--------------------------------------------------------------------
<Emp>
  <column name="EMPLOYEE_ID">204</column>
  <column name="LAST_NAME">Baer</column>
  <column name="SALARY">10000</column>
</Emp>
```

これらの 2 つのファンクションの出力の比較については、5-237 ページの「XMLFOREST」を参照してください。

XMLCOMMENT

構文



用途

XMLComment は、*value_expr* の評価結果を使用して XML コメントを生成します。*value_expr* は文字列に変換する必要があります。このファンクションには、2 つの連続したダッシュ（ハイフン）を含めることはできません。このファンクションの戻り値は、次の書式になります。

```
<!--string-->
```

value_expr が NULL である場合、このファンクションは NULL を戻します。

> **参照：** このファンクションの詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

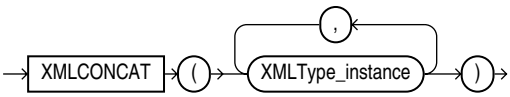
例

次の例は、DUAL 表を使用して、XMLComment 構文を示します。

```
SELECT XMLCOMMENT('OrderAnalysisComp imported, reconfigured, disassembled')
   AS "XMLCOMMENT" FROM DUAL;

XMLCOMMENT
--------------------------------------------------------------------------------
<!--OrderAnalysisComp imported, reconfigured, disassembled-->
```

XMLCONCAT

構文



用途

XMLCONCAT は、入力として一連の XMLType インスタンスを取り、各行について一連の要素を連結し、連結した結果を戻します。XMLCONCAT は XMLSEQUENCE の逆の処理をします。

NULL 式は結果から排除されます。値のすべての式が NULL の場合、このファンクションは NULL を戻します。

> **参照：** 「XMLSEQUENCE」（5-243 ページ）

例

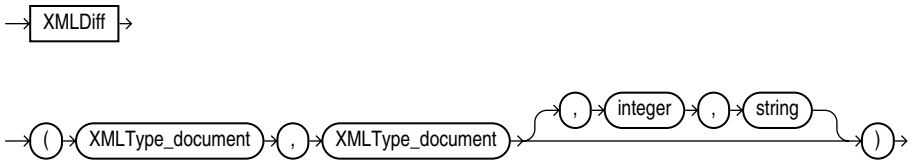
次の例では、従業員のサブセットの名前と姓に対して XML 要素を作成し、作成した要素を連結して戻します。

```
SELECT XMLCONCAT(XMLELEMENT("First", e.first_name),
   XMLELEMENT("Last", e.last_name)) AS "Result"
   FROM employees e
   WHERE e.employee_id > 202;
```

```
Result
--------------------------------------------------------------------
<First>Susan</First>
<Last>Mavris</Last>

<First>Hermann</First>
<Last>Baer</Last>

<First>Shelley</First>
<Last>Higgins</Last>

<First>William</First>
<Last>Gietz</Last>

4 rows selected.
```

XMLDIFF

構文



用途

XMLDiff ファンクションは、XmlDiff C API の SQL インタフェースです。このファンクションは、2 つの XML 文書を比較し、Xdiff スキーマに基づいた XML の差異を取得します。diff 文書が XMLType 文書として戻されます。

- 最初の 2 つの引数には、2 つの XMLType 文書の名前を指定します。
- *integer* には、C ファンクション XmlDiff の hashLevel を表す数値を指定します。ハッシュを使用しない場合は、この引数を 0（ゼロ）に設定するか、または引数全体を省略します。ハッシュを使用しないでフラグを指定する場合は、この引数を 0（ゼロ）に設定する必要があります。
- *string* には、ファンクションの動作を制御するフラグを指定します。これらのフラグは、セミコロンで区切られた 1 つ以上の名前で指定します。これらの名前は、XmlDiff ファンクションの定数の名前と同じです。

> **参照：** このファンクションの使用方法と例については、『Oracle XML Developer's Kit プログラマーズ・ガイド』を参照してください。C の XML API の詳細は、『Oracle Database XML C API リファレンス』を参照してください。

例

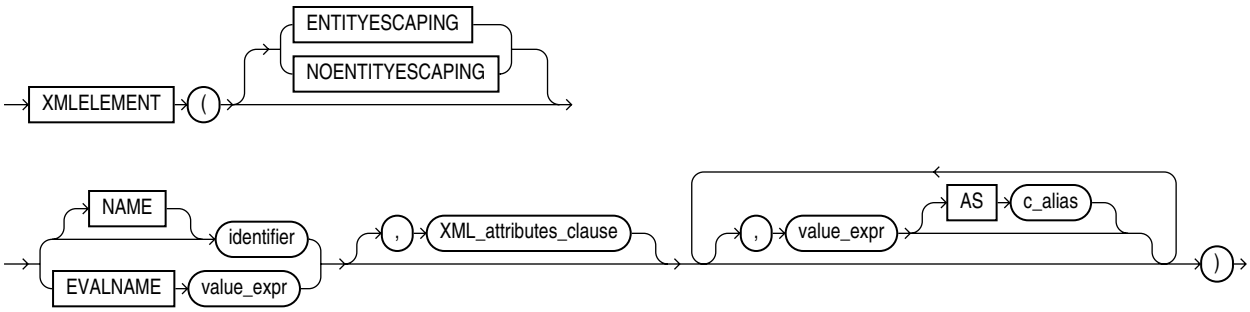
次の例では、2 つの XML 文書を比較し、差異を XMLType 文書として戻します。

```
SELECT XMLDIFF(
XMLTYPE('<?xml version="1.0"?>
<bk:book xmlns:bk="http://nosuchsite.com">
   <bk:tr>
        <bk:td>
                <bk:chapter>
                        Chapter 1.
                </bk:chapter>
        </bk:td>
```

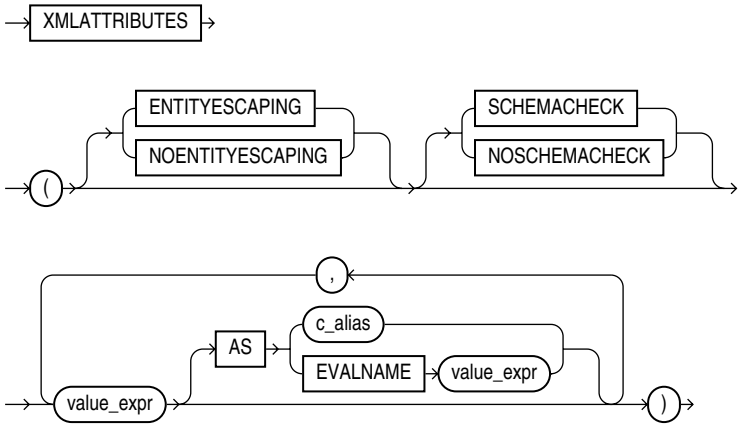
```
        <bk:td>
                 <bk:chapter>
                        Chapter 2.
                </bk:chapter>
        </bk:td>
   </bk:tr>
</bk:book>'),
XMLTYPE('<?xml version="1.0"?>
<bk:book xmlns:bk="http://nosuchsite.com">
   <bk:tr>
        <bk:td>
                <bk:chapter>
                        Chapter 1.
                </bk:chapter>
        </bk:td>
        <bk:td/>
   </bk:tr>
</bk:book>')
)
FROM DUAL;
```

XMLELEMENT

構文



***XML_attributes_clause*::=**



用途

XMLELEMENT は、*identifier* に対する要素名を取るか、または EVALNAME *value_expr* の要素名、要素に対する属性のオプションのコレクション、およびその要素の内容を構成する引数を評価します。このファンクションは XMLType 型のインスタンスを戻します。XMLELEMENT は、戻された XML に属性を格納できる点を除いて SYS_XMLGEN に似ていますが、XMLFormat オブジェクトを使用した書式設定は受け入れません。

次の項に示すとおり、通常は、XMLELEMENT ファンクションはネストされており、ネスト構造の XML 文書を生成します。

ENTITYESCAPING および NONENTITYESCAPING キーワードについては、『Oracle XML DB 開発者ガイド』を参照してください。

囲みタグに使用する Oracle Database の値を指定する必要があります。この場合、文字列リテラルの *identifier* を指定するか、または EVALNAME *value_expr* を指定します。後者の場合は、値の式が評価され、その結果（文字列リテラル）が識別子として使用されます。この識別子は、最大 4000 文字まで指定可能で、列名または列の参照である必要はありません。式または NULL は指定できません。

要素の内容を構成するオブジェクトは、XMLATTRIBUTES キーワードの後に指定します。*XML_attributes_clause* では、*value_expr* が NULL の場合、その値の式に対する属性は作成されません。*value_expr* の型に、オブジェクト型またはコレクションは指定できません。AS 句を使用して *value_expr* に別名を指定すると、*c_alias* または評価された値の式（EVALNAME *value_expr*）は、最大 4000 文字まで指定可能になります。

構文図の *XML_attributes_clause* に続くオプションの *value_expr* は、次のようになります。

- *value_expr* がスカラー式である場合、AS 句は省略できます。この場合、Oracle は列名を要素名として使用します。
- *value_expr* がオブジェクト型またはコレクションである場合、AS 句は必須です。この場合、Oracle は指定された *c_alias* を囲みタグとして使用します。
- *value_expr* が NULL の場合、その値の式に対する要素は作成されません。

参照：「SYS_XMLGEN」（5-193 ページ）

例

次の例では、一連の従業員について、従業員の名前と雇用日を指定する、ネストされた要素を持つ Emp 要素を生成します。

```
SELECT XMLELEMENT("Emp", XMLELEMENT("Name",
   e.job_id||' '||e.last_name),
   XMLELEMENT("Hiredate", e.hire_date)) as "Result"
   FROM employees e WHERE employee_id > 200;

Result
--------------------------------------------------------------------
<Emp>
  <Name>MK_MAN Hartstein</Name>
  <Hiredate>17-FEB-96</Hiredate>
</Emp>

<Emp>
  <Name>MK_REP Fay</Name>
  <Hiredate>17-AUG-97</Hiredate>
</Emp>

<Emp>
  <Name>HR_REP Mavris</Name>
  <Hiredate>07-JUN-94</Hiredate>
</Emp>
```

```
<Emp>
  <Name>PR_REP Baer</Name>
  <Hiredate>07-JUN-94</Hiredate>
</Emp>

<Emp>
  <Name>AC_MGR Higgins</Name>
  <Hiredate>07-JUN-94</Hiredate>
</Emp>

<Emp>
  <Name>AC_ACCOUNT Gietz</Name>
  <Hiredate>07-JUN-94</Hiredate>
</Emp>

6 rows selected.
```

次の例では、XMLELEMENT ファンクションに *XML_attributes_clause* を使用して、トップレベル要素に対する属性値を持つ、ネストされた XML 要素を作成します。

```
SELECT XMLELEMENT("Emp",
      XMLATTRIBUTES(e.employee_id AS "ID", e.last_name),
      XMLELEMENT("Dept", e.department_id),
      XMLELEMENT("Salary", e.salary)) AS "Emp Element"
   FROM employees e
   WHERE e.employee_id = 206;

Emp Element
--------------------------------------------------------------
<Emp ID="206" LAST_NAME="Gietz">
  <Dept>110</Dept>
  <Salary>8300</Salary>
</Emp>
```

last_name には AS *identifier* 句が指定されていません。その結果、戻された XML は、デフォルトで列名 last_name を使用します。

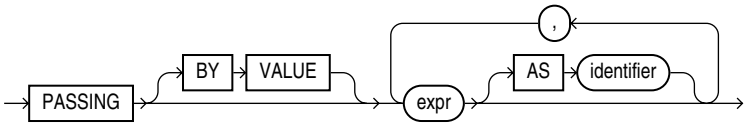
最後に、次の例では、*XML_attributes_clause* に副問合せを使用して、別の表から要素の属性に情報を取り出します。

```
SELECT XMLELEMENT("Emp", XMLATTRIBUTES(e.employee_id, e.last_name),
   XMLELEMENT("Dept", XMLATTRIBUTES(e.department_id,
   (SELECT d.department_name FROM departments d
   WHERE d.department_id = e.department_id) as "Dept_name")),
   XMLELEMENT("salary", e.salary),
   XMLELEMENT("Hiredate", e.hire_date)) AS "Emp Element"
   FROM employees e
   WHERE employee_id = 205;

Emp Element
---------------------------------------------------------------------
<Emp EMPLOYEE_ID="205" LAST_NAME="Higgins">
  <Dept DEPARTMENT_ID="110" Dept_name="Accounting"/>
  <salary>12000</salary>
  <Hiredate>07-JUN-94</Hiredate>
</Emp>
```

XMLEXISTS

構文

***XML_passing_clause*::=**



用途

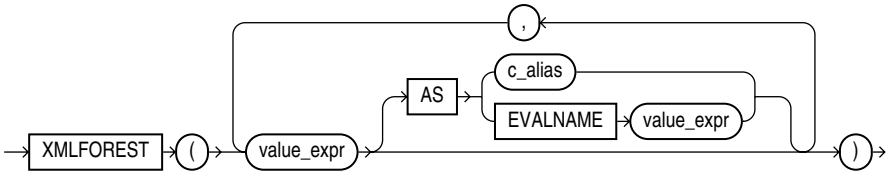
XMLExists は、指定された XQuery 式から空でない XQuery シーケンスが戻されるかどうかをチェックします。空でない XQuery シーケンスが戻される場合、このファンクションは TRUE を戻し、それ以外の場合は FALSE を戻します。引数 *XQuery_string* はリテラル文字列ですが、*XML_passing_clause* を使用してバインドする XQuery 変数を含めることができます。

XML_passing_clause の *expr* は、XMLType または SQL スカラー・データ型のインスタンスを戻し、XQuery 式を評価するためのコンテキストとして使用されます。PASSING 句には、識別子を指定せずに 1 つの *expr* のみを指定できます。各 *expr* の評価結果は、*XQuery_string* の対応する識別子にバインドされます。*expr* の後に AS 句が続かない場合、式の評価結果は *XQuery_string* の評価用のコンテキスト項目として使用されます。

> **参照：** このファンクションの使用の詳細および例は、『Oracle XML DB 開発者ガイド』を参照してください。

XMLFOREST

構文



用途

XMLFOREST は、各引数パラメータを XML に変換し、変換された引数を連結した XML フラグメントを戻します。

- *value_expr* がスカラー式である場合、AS 句は省略できます。この場合、Oracle Database は列名を要素名として使用します。
- *value_expr* がオブジェクト型またはコレクションである場合、AS 句は必須です。この場合、Oracle は指定された式を囲みタグとして使用します。

  この場合、文字列リテラルの *c_alias* を指定するか、または EVALNAME *value_expr* を指定します。後者の場合は、値の式が評価され、その結果（文字列リテラル）が識別子として使用されます。この識別子は、最大 4000 文字まで指定可能で、列名または列の参照である必要はありません。式または NULL は指定できません。
- *value_expr* が NULL の場合、その *value_expr* に対する要素は作成されません。

例

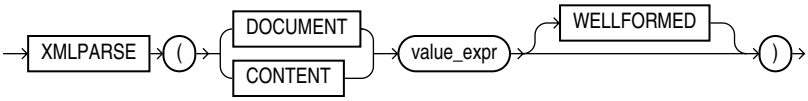
次の例では、従業員のサブセットに対して Emp 要素を作成します。このとき、Emp の内容として、employee_id、last_name および salary の各要素がネストされます。

```
SELECT XMLELEMENT("Emp",
   XMLFOREST(e.employee_id, e.last_name, e.salary))
   "Emp Element"
   FROM employees e WHERE employee_id = 204;

Emp Element
--------------------------------------------------------------
<Emp>
  <EMPLOYEE_ID>204</EMPLOYEE_ID>
  <LAST_NAME>Baer</LAST_NAME>
  <SALARY>10000</SALARY>
</Emp>
```

これらの 2 つのファンクションの出力の比較については、5-231 ページの「XMLCOLATTVAL」を参照してください。

XMLPARSE

構文



用途

XMLParse は、*value_expr* の評価結果から XML インスタンスを解析して生成します。*value_expr* は文字列に変換する必要があります。*value_expr* が NULL である場合、このファンクションは NULL を戻します。

- DOCUMENT を指定する場合、*value_expr* は単一ルートの XML 文書である必要があります。
- CONTENT を指定する場合、*value_expr* は有効な XML 値である必要があります。
- WELLFORMED を指定する場合、*value_expr* は整形式 XML 文書である必要があります。これは、入力が正しい書式であることを確認する、データベースによる妥当性チェックが実行されないためです。

> **参照：** このファンクションの詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

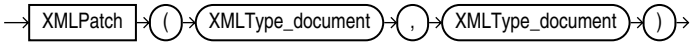
例

次の例は、DUAL 表を使用して、XMLParse の構文を示します。

```
SELECT XMLPARSE(CONTENT '124 <purchaseOrder poNo="12435">
   <customerName> Acme Enterprises</customerName>
   <itemNo>32987457</itemNo>
   </purchaseOrder>'
WELLFORMED) AS PO FROM DUAL;

PO
--------------------------------------------------------------
124 <purchaseOrder poNo="12435">
   <customerName> Acme Enterprises</customerName>
   <itemNo>32987457</itemNo>
   </purchaseOrder>
```

XMLPATCH

構文



用途

XMLPatch ファンクションは、XmlPatch C API の SQL インタフェースです。このファンクションは、指定された変更を適用して XML 文書を修正します。修正された XMLType 文書が戻されます。

- 最初の引数には、入力 XMLType 文書の名前を指定します。

  2 番目の引数には、最初の文書に適用する変更を含む XMLType 文書を指定します。変更は、Xdiff XML スキーマに基づいている必要があります。

- *string* には、ファンクションの動作を制御するフラグを指定します。これらのフラグは、セミコロンで区切られた 1 つ以上の名前で指定します。これらの名前は、XmlPatch C ファンクションの定数の名前と同じです。

> **参照：** このファンクションの使用方法と例については、『Oracle XML Developer's Kit プログラマーズ・ガイド』を参照してください。C の XML API の詳細は、『Oracle Database XML C API リファレンス』を参照してください。

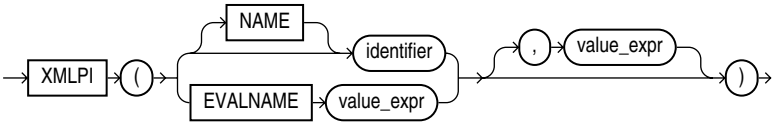
例

次の例では、別の XMLType 文書に指定された変更を適用して XMLType 文書を修正し、修正した XMLType 文書を戻します。

```
SELECT XMLPATCH(
XMLTYPE('<?xml version="1.0"?>
<bk:book xmlns:bk="http://nosuchsite.com">
   <bk:tr>
        <bk:td>
                <bk:chapter>
                        Chapter 1.
                </bk:chapter>
        </bk:td>
        <bk:td>
                 <bk:chapter>
                        Chapter 2.
                </bk:chapter>
        </bk:td>
   </bk:tr>
</bk:book>'),
XMLTYPE('<?xml version="1.0"?>
<xd:xdiff xsi:schemaLocation="http://xmlns.example.com/xdb/xdiff.xsd
  http://xmlns.example.com/xdb/xdiff.xsd"
  xmlns:xd="http://xmlns.example.com/xdb/xdiff.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:bk="http://nosuchsite.com">
  <?oracle-xmldiff operations-in-docorder="true" output-model="snapshot"
    diff-algorithm="global"?>
  <xd:delete-node xd:node-type="element"
   xd:xpath="/bk:book[1]/bk:tr[1]/bk:td[2]/bk:chapter[1]"/>
</xd:xdiff>')
)
FROM DUAL;
```

XMLPI

構文



用途

XMLPI は、*identifier* と *value_expr* の評価結果（オプション）を使用して、XML の処理命令を生成します。通常、処理命令は、XML 文書のすべてまたは一部に関連付けられたアプリケーション情報に使用されます。アプリケーションはこの処理命令を使用して、XML 文書の最適な処理方法を決定します。

囲みタグに使用する Oracle Database の値を指定する必要があります。この場合、文字列リテラルの *identifier* を指定するか、または EVALNAME *value_expr* を指定します。後者の場合は、値の式が評価され、その結果（文字列リテラル）が識別子として使用されます。この識別子は、最大 4000 文字まで指定可能で、列名または列の参照である必要はありません。式または NULL は指定できません。

オプションの *value_expr* は文字列に変換する必要があります。オプションの *value_expr* を指定しない場合、デフォルトは長さが 0（ゼロ）の文字列です。このファンクションの戻り値は、次の書式になります。

```
<?identifier string?>
```

XMLPI には、次の制限事項があります。

- *identifier* は、処理命令の有効な対象である必要があります。
- xml は、*identifier* と組み合せて指定することはできません。
- *identifier* には、連続文字 ?> を含めることはできません。

> **参照：** このファンクションの詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

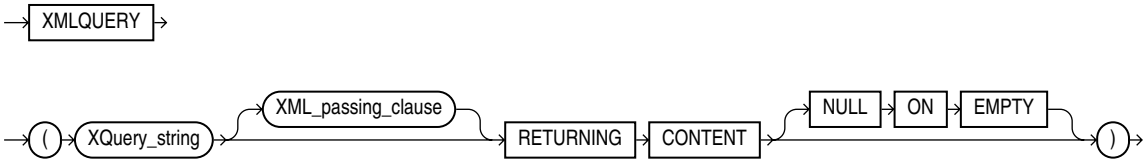
例

次の文は、DUAL 表を使用して、XMLPI の構文の使用を示します。

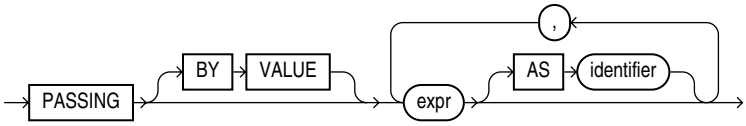
```
SELECT XMLPI(NAME "Order analysisComp", 'imported, reconfigured, disassembled')
   AS "XMLPI" FROM DUAL;

XMLPI
--------------------------------------------------------------------------------
<?Order analysisComp imported, reconfigured, disassembled?>
```

XMLQUERY

構文



XML_passing_clause::=



用途

XMLQUERY では、SQL 文の XML データを問い合せることができます。このファンクションは、文字列リテラル、オプションのコンテキスト項目、および他のバインド変数として XQuery 式を取り、これらの入力値を使用して XQuery 式の評価結果を戻します。

- *XQuery_string* は、プロローグを含む完全な XQuery 式です。
- *XML_passing_clause* の *expr* は、XMLType または SQL スカラー・データ型のインスタンスを戻し、XQuery 式を評価するためのコンテキストとして使用されます。PASSING 句には、識別子を指定せずに 1 つの *expr* のみを指定できます。各 *expr* の評価結果は、*XQuery_string* の対応する識別子にバインドされます。*expr* の後に AS 句が続かない場合、式の評価結果は *XQuery_string* の評価用のコンテキスト項目として使用されます。
- RETURNING CONTENT は、XQuery 式の評価結果が XML 1.0 文書か、または XML 1.0 セマンティクスに準拠したドキュメント・フラグメントのいずれかであることを示します。
- 結果セットが空の場合、このファンクションは、SQL NULL 値を戻します。NULL ON EMPTY キーワードがデフォルトで実装されます。このキーワードは、意味を明確にするために示されます。

> **参照：** このファンクションの詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

例

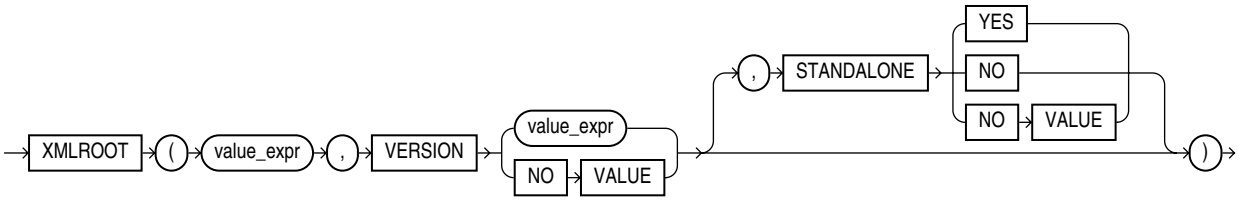
次の文では、*XML_passing_clause* の oe.warehouses 表の *warehouse_spec* 列をコンテキスト項目として指定します。この文は、領域が 50K より大きいウェアハウスに関する特定の情報を戻します。

```
SELECT warehouse_name,
EXTRACTVALUE(warehouse_spec, '/Warehouse/Area'),
XMLQuery(
   'for $i in /Warehouse
   where $i/Area > 50000
   return <Details>
             <Docks num="{$i/Docks}"/>
             <Rail>
               {
               if ($i/RailAccess = "Y") then "true" else "false"
               }
             </Rail>
          </Details>' PASSING warehouse_spec RETURNING CONTENT) "Big_warehouses"
   FROM warehouses;
```

```
WAREHOUSE_ID Area      Big_warehouses
------------ --------- ------------------------------------------------------
           1     25000
           2     50000
           3     85700 <Details><Docks></Docks><Rail>false</Rail></Details>
           4    103000 <Details><Docks num="3"></Docks><Rail>true</Rail></Details>
 . . .
```

XMLROOT

構文



用途

XMLROOT では、既存の XML 値の XML ルート情報（プロローグ）のバージョンとスタンドアロンのプロパティを指定して、新しい XML 値を作成できます。*value_expr* がすでにプロローグを持っている場合、エラーが戻されます。入力が NULL の場合、このファンクションは NULL を戻します。

戻り値は次の書式になります。

```
<?xml version = "version" [ STANDALONE = "{yes | no}" ]?>
```

- 最初の *value_expr* では、XML 値を指定します。この値に対してプロローグ情報を指定します。
- VERSION 句の *value_expr* は、有効な XML バージョンを表す文字列である必要があります。VERSION に NO VALUE を指定すると、バージョンはデフォルトで 1.0 になります。
- オプションの STANDALONE 句を指定しない場合や、NO VALUE を指定した場合は、ファンクションの戻り値にスタンドアロンのプロパティは存在しません。

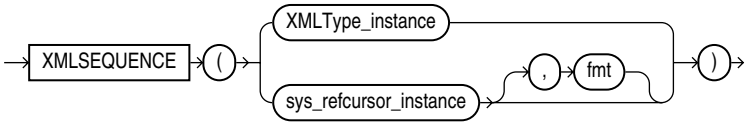
例

次の文は、DUAL 表を使用して、XMLROOT の構文を示します。

```
SELECT XMLROOT ( XMLType('<poid>143598</poid>'), VERSION '1.0', STANDALONE YES)
   AS "XMLROOT" FROM DUAL;

XMLROOT
--------------------------------------------------------------------------------
<?xml version="1.0" standalone="yes"?>
<poid>143598</poid>
```

XMLSEQUENCE

構文



用途

XMLSEQUENCE には 2 つの書式があります。

- 1 つ目の書式は、入力として XMLType インスタンスを取り、XMLType にあるトップレベルのノードの VARRAY を戻します。この書式のかわりに、SQL/XML 標準関数 XMLTable を使用すると効果的です。この関数を使用すると、SQL コードがより読みやすくなります。Oracle Database10*g* リリース 2（10.2）より前のリリースでは、このリリースの XMLTable 関数でより効果的に実行されるいくつかの機能を、XMLSequence が SQL 関数 TABLE を使用して実行していました。
- 2 つ目の書式は、入力として REFCURSOR インスタンスおよびオプションの XMLFormat オブジェクトのインスタンスを取り、カーソルの各行に対して、XMLSequence 型として XML 文書を戻します。

XMLSEQUENCE は XMLType のコレクションを戻すため、このファンクションを TABLE 句で使用して、コレクション値をネスト解除することで複数行にし、SQL 問合せでの処理をさらに進めることができます。

> **参照：** このファンクションの詳細は、『Oracle XML DB 開発者ガイド』を参照してください。また 5-245 ページの「XMLTABLE」も参照してください。

例

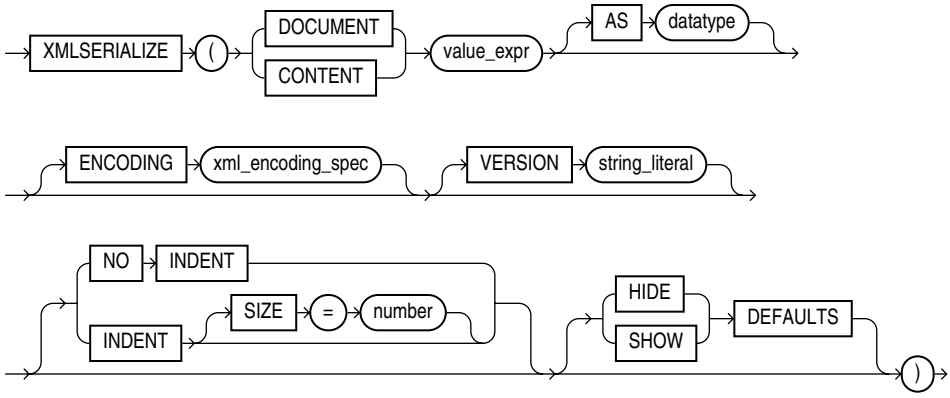
次の例では、XMLSEQUENCE が複数の要素を持つ XML 文書を VARRAY 型の単一要素ドキュメントに分割する方法を示しています。この例では、TABLE キーワードが、コレクションを副問合せの FROM 句で使用できる表の値とみなすように、Oracle Database に指示しています。

```
SELECT EXTRACT(warehouse_spec, '/Warehouse') as "Warehouse"
   FROM warehouses WHERE warehouse_name = 'San Francisco';

Warehouse
------------------------------------------------------------
<Warehouse>
  <Building>Rented</Building>
  <Area>50000</Area>
  <Docks>1</Docks>
  <DockType>Side load</DockType>
  <WaterAccess>Y</WaterAccess>
  <RailAccess>N</RailAccess>
  <Parking>Lot</Parking>
  <VClearance>12 ft</VClearance>
</Warehouse>

1 row selected.

SELECT VALUE(p)
   FROM warehouses w,
   TABLE(XMLSEQUENCE(EXTRACT(warehouse_spec, '/Warehouse/*'))) p
   WHERE w.warehouse_name = 'San Francisco';
```

```
VALUE(P)
--------------------------------------------------------------
<Building>Rented</Building>
<Area>50000</Area>
<Docks>1</Docks>
<DockType>Side load</DockType>
<WaterAccess>Y</WaterAccess>
<RailAccess>N</RailAccess>
<Parking>Lot</Parking>
<VClearance>12 ft</VClearance>

8 rows selected.
```

XMLSERIALIZE

構文



用途

XMLSerialize は、*value_expr* の内容を含む文字列または LOB を作成します。

- DOCUMENT を指定する場合、*value_expr* は有効な XML 文書である必要があります。
- CONTENT を指定する場合、*value_expr* は単一ルートの XML 文書である必要はありません。ただし、有効な XML コンテンツである必要があります。
- 指定する *datatype* は、文字列型（VARCHAR2 または VARCHAR は使用可能、NVARCHAR または NVARCHAR2 は使用不可）、BLOB または CLOB です。デフォルトは CLOB です。
- *datatype* が BLOB の場合は、ENCODING 句を指定して、指定したエンコーディングをプロローグで使用できます。
- XML 宣言で *string_literal* として指定したバージョン（<?xml version="..." ...?>）を使用するには、VERSION 句を指定します。
- 意味のないすべての空白を出力から取り除くには、NO INDENT を指定します。出力を *N* 個の空白の相対インデントを使用して出力整形するには、INDENT SIZE = *N* を指定します。*N* は整数です。*N* が 0 の場合、出力整形によって各要素の後に改行文字が挿入され、各要素は独自の行に配置されますが、その他の意味のないすべての空白は出力で省略されます。SIZE を指定せずに INDENT を指定すると、2 個の空白インデントが使用されます。この句を指定しない場合、動作（出力整形されるかどうか）は予測不能です。
- HIDE DEFAULTS および SHOW DEFAULTS は、XML スキーマベースのデータにのみ適用されます。SHOW DEFAULTS を指定し、XML スキーマがデフォルト値を定義するオプションの要素または属性が入力データにない場合、それらの要素または属性はデフォルト値を使用して出力に含まれます。HIDE DEFAULTS を指定した場合、そのような要素または属性は出力に含まれません。HIDE DEFAULTS がデフォルトの動作です。

参照： このファンクションの詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

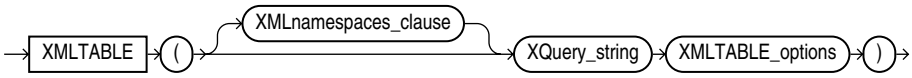
例

次の文は、DUAL 表を使用して、XMLSerialize の構文を示します。

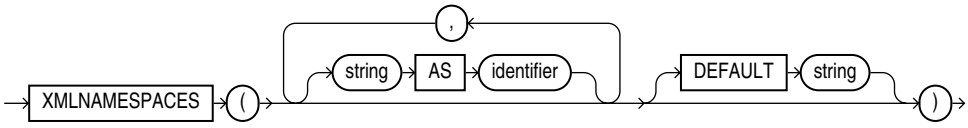
```
SELECT XMLSERIALIZE(CONTENT XMLTYPE('<Owner>Grandco</Owner>'))
   FROM DUAL;
```

XMLTABLE

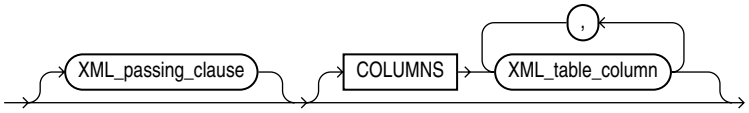
構文



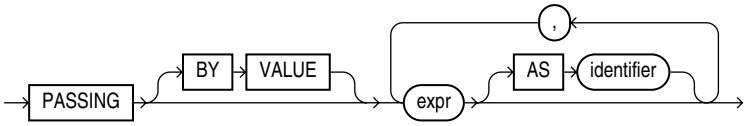
***XMLnamespaces_clause*::=**



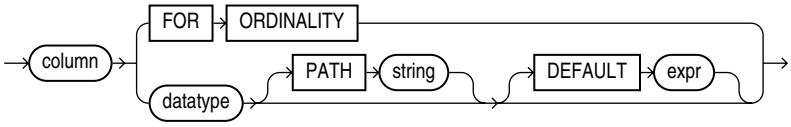
***XMLTABLE_options*::=**



***XML_passing_clause*::=**



***XML_table_column*::=**



用途

XMLTable は、XQuery の評価結果をリレーショナル行および列にマップします。SQL を使用して、このファンクションで戻される結果を仮想リレーショナル表として問い合せることができます。

- XMLNAMESPACES 句には、XML ネームスペースの宣言が含まれます。これらの宣言は、XQuery 式（評価された *XQuery_string*）と、*XML_table_column* の PATH 句の XPath 式で参照されます。XQuery 式は行を計算し、XPath 式は XMLTable ファンクション全体の列を計算します。COLUMNS 句の PATH 式に修飾名を使用する場合は、XMLNAMESPACES 句を指定する必要があります。
- *XQuery_string* は、完全な XQuery 式で、プロローグ宣言を含むことができます。

- *XML_passing_clause* の *expr* は、XMLType または SQL スカラー・データ型のインスタンスを戻し、XQuery 式を評価するためのコンテキストとして使用されます。PASSING 句には、識別子を指定せずに 1 つの *expr* のみを指定できます。各 *expr* の評価結果は、*XQuery_string* の対応する識別子にバインドされます。*expr* の後に AS 句が続かない場合、式の評価結果は *XQuery_string* の評価用のコンテキスト項目として使用されます。
- オプションの COLUMNS 句では、XMLTable で作成する仮想表の列を定義します。
  - COLUMNS 句を指定しない場合、XMLTable は、COLUMN_VALUE という名前の単一の XMLType 疑似列を戻します。
  - XMLTable が XMLType, *datatype* という XML スキーマ・ベースの記憶域とともに使用される場合以外は、*datatype* が必要です。この場合、*datatype* の指定を省略すると、Oracle XML DB は、XML スキーマに基づいてデータ型を推測します。データベース側で、ノードの正しい型が判断できないときは、デフォルト型である VARCHAR2(4000) が使用されます。
  - FOR ORDINALITY は、生成された行番号の列になるように指定します。FOR ORDINALITY 句は、最大で 1 つにする必要があります。これは、NUMBER 列として作成されます。
  - オプションの PATH 句では、XQuery 式の文字列でアドレス指定される XQuery 結果の部分を列の内容として使用するように指定します。PATH を指定しない場合、XQuery 式の *column* とみなされます。たとえば、次のようになります。

    ```
    XMLTable(... COLUMNS xyz
    ```

    これは、次の式と同じです。

    ```
    XMLTable(... COLUMNS xyz PATH 'XYZ')
    ```

    異なる PATH 句を使用すると、XQuery 結果を異なる仮想表の列に分割できます。
  - オプションの DEFAULT 句では、PATH 式の結果が空のシーケンスの場合に使用する値を指定します。*expr* は、デフォルト値の生成用に評価する XQuery 式です。

    参照： 追加の例を含む XMLTable ファンクションおよび XQuery の概要については、『Oracle XML DB 開発者ガイド』を参照してください。

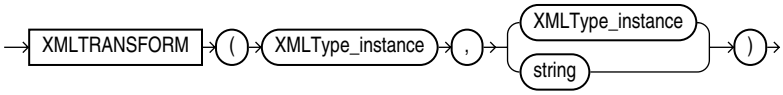
例

次の例では、warehouses 表の warehouse_spec 列の各値に XQuery '/Warehouse' を適用した結果を、列 Water および Rail のある仮想リレーショナル表に変換します。

```
SELECT warehouse_name warehouse,
   warehouse2."Water", warehouse2."Rail"
   FROM warehouses,
   XMLTABLE('/Warehouse'
      PASSING warehouses.warehouse_spec
      COLUMNS
         "Water" varchar2(6) PATH '/Warehouse/WaterAccess',
         "Rail" varchar2(6) PATH '/Warehouse/RailAccess')
      warehouse2;

WAREHOUSE                          Water  Rail
---------------------------------- ------ ------
Southlake, Texas                   Y      N
San Francisco                      Y      N
New Jersey                         N      N
Seattle, Washington                N      Y
```

XMLTRANSFORM

構文



用途

XMLTRANSFORM は、引数として XMLType インスタンスおよび XSL スタイルシート（それ自体が XMLType インスタンス）を取ります。このファンクションは、スタイルシートをインスタンスに適用して、XMLType を戻します。

このファンクションは、データをデータベースから取得するように、スタイルシートに従ってデータを編成する場合に有効です。

> **参照：** このファンクションの詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

例

XMLTRANSFORM ファンクションを使用するには、XSL スタイルシートが必要です。次に、ノード内の要素をアルファベット順に並べる単純なスタイルシートの例を示します。

```
CREATE TABLE xsl_tab (col1 XMLTYPE);

INSERT INTO xsl_tab VALUES (
   XMLTYPE.createxml(
   '<?xml version="1.0"?>
    <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
      <xsl:output encoding="utf-8"/>
      <!-- alphabetizes an xml tree -->
      <xsl:template match="*">
        <xsl:copy>
          <xsl:apply-templates select="*|text()">
            <xsl:sort select="name(.)" data-type="text" order="ascending"/>
          </xsl:apply-templates>
        </xsl:copy>
      </xsl:template>
      <xsl:template match="text()">
        <xsl:value-of select="normalize-space(.)"/>
      </xsl:template>
    </xsl:stylesheet>  '));

1 row created.
```

次の例では、XSL スタイルシート xsl_tab を使用して、サンプル表 oe.warehouses にある warehouse_spec の要素をアルファベット順に並べます。

```
SELECT XMLTRANSFORM(w.warehouse_spec, x.col1).GetClobVal()
   FROM warehouses w, xsl_tab x
   WHERE w.warehouse_name = 'San Francisco';

XMLTRANSFORM(W.WAREHOUSE_SPEC,X.COL1).GETCLOBVAL()
--------------------------------------------------------------------------------
<Warehouse>
  <Area>50000</Area>
  <Building>Rented</Building>
  <DockType>Side load</DockType>
  <Docks>1</Docks>
  <Parking>Lot</Parking>
```

```
  <RailAccess>N</RailAccess>
  <VClearance>12 ft</VClearance>
  <WaterAccess>Y</WaterAccess>
</Warehouse>
```

ROUND および TRUNC 日付ファンクション

表 5-14 に、ROUND および TRUNC 日付ファンクションで使用できる書式モデル、および日付の丸めと切捨ての単位を示します。デフォルトのモデル DD では、午前 0 時（真夜中）を基準に丸めおよび切捨てを行い、日付を戻します。

表 5-14 ROUND および TRUNC 日付ファンクションの日付書式モデル

書式モデル	丸め単位または切捨て単位
CC SCC	4 桁の年号の上 2 桁より 1 大きい数
SYYYY YYYY YEAR SYEAR YYY YY Y	年（7 月 1 日に切上げ）
IYYY IY IY I	ISO 年
Q	四半期（その四半期の 2 番目の月の 16 日に切上げ）
MONTH MON MM RM	月（16 日に切上げ）
WW	年の最初の日と同じ曜日
IW	ISO 年の最初の日と同じ曜日
W	月の最初の日と同じ曜日
DDD DD J	日
DAY DY D	週の開始日
HH HH12 HH24	時
MI	分

書式モデル DAY、DY および D によって使用される週の開始日は、NLS_TERRITORY 初期化パラメータによって暗黙的に指定されています。

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

ユーザー定義ファンクション

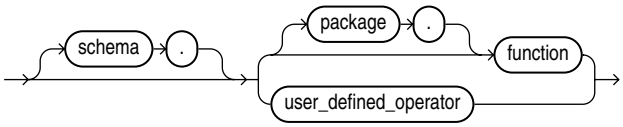
PL/SQL または Java でユーザー定義ファンクションを作成し、SQL または SQL 組込みファンクションにはない機能を持たせることができます。ユーザー定義ファンクションは、式を指定できる場所であればどこでも、SQL 文に指定できます。

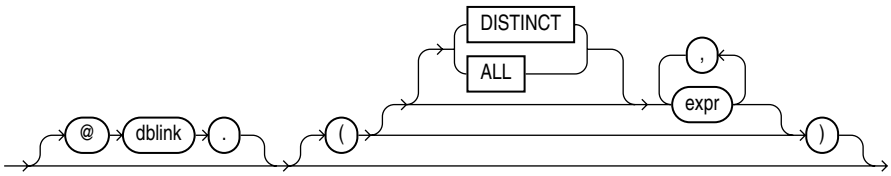
たとえば、SQL 文の次の場所でユーザー定義ファンクションを使用できます。

- SELECT 文の SELECT 構文のリスト
- WHERE 句の条件
- CONNECT BY 句、START WITH 句、ORDER BY 句および GROUP BY 句
- INSERT 文の VALUES 句
- UPDATE 文の SET 句

> **注意：** Oracle SQL は、ブール値のパラメータまたは戻り値を持つファンクションのコールをサポートしていません。したがって、ユーザー定義ファンクションが SQL 文からコールされる場合は、数値（0 または 1）または文字列（'TRUE' または 'FALSE'）を戻すように設計する必要があります。

***user_defined_function*::=**





オプションの式のリストは、ファンクション、パッケージまたは演算子の属性と一致する必要があります。

ユーザー定義ファンクションの制限事項： DISTINCT および ALL キーワードは、ユーザー定義集計ファンクションのみで有効です。

参照：

- ファンクションの作成（ユーザー定義ファンクションの制限を含む）の詳細は、14-48 ページの「CREATE FUNCTION」を参照してください。
- ユーザー・ファンクションの作成および使用方法の詳細は、『Oracle Database アドバンスト・アプリケーション開発者ガイド』を参照してください。

前提条件

ユーザー定義ファンクションを SQL 文で使用するには、トップレベル・ファンクションとして作成するか、またはパッケージ仕様部で宣言する必要があります。

SQL の式の中でユーザー・ファンクションを使用するには、ユーザーがユーザー・ファンクションの EXECUTE 権限を持っている必要があります。ユーザー・ファンクションで定義したビューを問い合せるには、そのビューに対する SELECT 権限が必要です。ビューから選択するには、個別の EXECUTE 権限は必要ありません。

> **参照：** トップレベル・ファンクションの詳細は、14-48 ページの「CREATE FUNCTION」を参照してください。パッケージ・ファンクションの詳細は、15-39 ページの「CREATE PACKAGE」を参照してください。

名前の優先順位

SQL 文内では、データベースの列名は、パラメータなしのファンクション名より優先順位が高くなります。たとえば、Human Resources のマネージャが、hr スキーマに次の 2 つのオブジェクトを作成する場合は、次のようにします。

```
CREATE TABLE new_emps (new_sal NUMBER, ...);
CREATE FUNCTION new_sal RETURN NUMBER IS BEGIN ... END;
```

その後、次の 2 つの文のように、new_sal を参照すると new_emps.new_sal 列を参照することになります。

```
SELECT new_sal FROM new_emps;
SELECT new_emps.new_sal FROM new_emps;
```

new_sal ファンクションにアクセスするには、次のように入力します。

```
SELECT hr.new_sal FROM new_emps;
```

SQL の式内で使用できるユーザー・ファンクションのコール例を次に示します。

```
circle_area (radius)
payroll.tax_rate (empno)
hr.employees.tax_rate (dependent, empno)@remote
```

例 hr スキーマから tax_rate ユーザー・ファンクションをコールし、tax_table 内の ss_no 列と sal 列に対してこのファンクションを実行するには、次のように指定します。

```
SELECT hr.tax_rate (ss_no, sal)
    INTO income_tax
    FROM tax_table WHERE ss_no = tax_id;
```

INTO 句は、結果を income_tax 変数に配置するために使用できる PL/SQL です。

ネーミング規則

オプションのスキーマ名またはパッケージ名を 1 つのみ指定すると、最初の識別子はスキーマ名またはパッケージ名のいずれかになります。たとえば、PAYROLL.TAX_RATE という参照内の PAYROLL がスキーマ名かパッケージ名かを判断するには、Oracle Database は次の手順を実行します。

1. カレント・スキーマ内の PAYROLL パッケージをチェックします。
2. PAYROLL パッケージが検出されない場合は、トップレベルの TAX_RATE ファンクションを含むスキーマ名 PAYROLL を検索します。このようなファンクションが検出されない場合は、エラーを戻します。

3. カレント・スキーマ内で PAYROLL パッケージが検出されると、PAYROLL パッケージの中で TAX_RATE ファンクションを検索します。このようなファンクションが検出されない場合は、エラーを戻します。

また、ユーザーが定義したシノニムを使用して、ストアド・トップレベル・ファンクションを参照することもできます。

6
式

この章では、値、演算子およびファンクションを式の中で組み合せて使用する方法について説明します。

この章では、次の内容を説明します。

- SQL 式
- 単純式
- 複合式
- CASE 式
- 列式
- CURSOR 式
- 日時式
- ファンクション式
- 期間式
- モデル式
- オブジェクト・アクセス式
- プレースホルダ式
- スカラー副問合せ式
- 型コンストラクタ式
- 式のリスト

SQL 式

式は、1 つ以上の値、演算子、および値に評価される SQL ファンクションの組合せです。一般に、式のデータ型は、そのコンポーネントのデータ型になります。

注意： NLS_COMP と NLS_SORT の設定を組み合せた値によって、文字をソートおよび比較するルールが決まります。ご使用のデータベースの NLS_COMP に LINGUISTIC が設定されている場合、この章のエンティティはすべて NLS_SORT パラメータによって指定されるルールに従って解釈されます。NLS_COMP が LINGUISTIC に設定されていない場合、ファンクションは NLS_SORT の設定に関係なく解釈されます。NLS_SORT は、明示的に設定できます。明示的に設定されていない場合は、NLS_LANGUAGE から導出されます。これらの設定の詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

次の単純式は、値が 4 になり、データ型は NUMBER（コンポーネントと同じデータ型）になります。

```
2*2
```

次の例は、ファンクションと演算子を使用した複雑な式です。この式は、現在の日付に 7 日を加算し、その合計から時間コンポーネントを削除し、結果を CHAR データ型に変換します。

```
TO_CHAR(TRUNC(SYSDATE+7))
```

次の場所で式を使用できます。

- SELECT 文の SELECT 構文のリスト
- WHERE 句および HAVING 句の条件
- CONNECT BY 句、START WITH 句および ORDER BY 句
- INSERT 文の VALUES 句
- UPDATE 文の SET 句

たとえば、次の UPDATE 文の SET 句で、引用符で囲まれた文字列 'Smith' のかわりに式を使用することもできます。

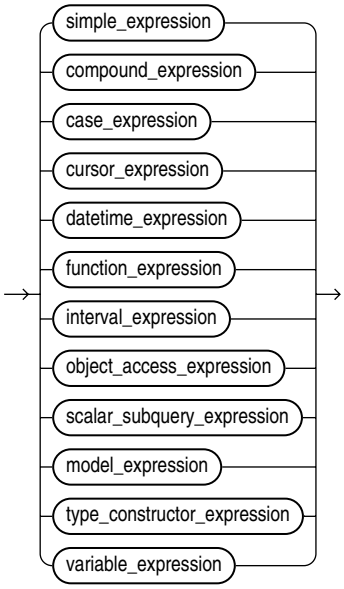
```
SET last_name = 'Smith';
```

この SET 句では、引用符で囲まれた文字列 'Smith' のかわりに、INITCAP(last_name) を使用しています。

```
SET last_name = INITCAP(last_name);
```

次の構文に示すとおり、式にはいくつかの書式があります。

***expr*::=**



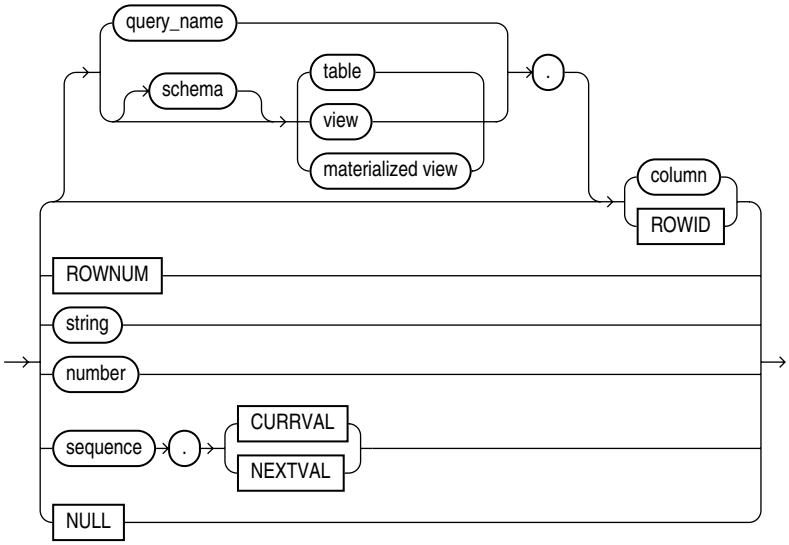
Oracle Database は、すべての SQL 文のすべての部分で、式のすべての書式を受け入れるわけではありません。文に指定する式の制限については、第 10 章～第 19 章の個々の SQL 文の説明を参照してください。

このマニュアルの他の箇所で、条件、SQL ファンクションまたは SQL 文に *expr* が示されている場合は、必ず適切な式の表記法を使用してください。次の項では、いくつかの例をあげて、様々な式の書式を説明します。

単純式

単純式は、列、疑似列、定数、順序番号または NULL を指定します。

***simple_expression*::=**



スキーマは各ユーザー用の他に、"PUBLIC"（二重引用符が必要）にもなり得ます。その場合、スキーマは表、ビューまたはマテリアライズド・ビューのパブリック・シノニムを修飾する必要があります。"PUBLIC" でのパブリック・シノニムの修飾は、データ操作言語（DML）文でのみサポートされています。データ定義言語（DDL）文ではサポートされていません。

列の後ろに続くオプションの PRECEDING キーワードおよび INITIAL キーワードは、バージョン問合せを発行する場合にのみ有効です。

ROWID は、表でのみ使用でき、ビューまたはマテリアライズド・ビューでは使用できません。NCHAR および NVARCHAR2 は、有効な疑似列データ型ではありません。

> **参照：** 疑似列の詳細は、第 3 章「疑似列」を参照してください。*query_name* の詳細は、19-13 ページの「subquery_factoring_clause」を参照してください。

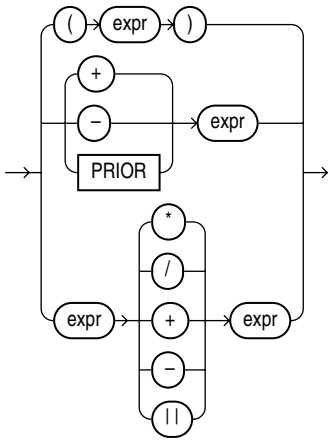
有効な単純式の例を次に示します。

```
employees.last_name
'this is a text string'
10
N'this is an NCHAR string'
```

複合式

複合式は、その他の式の組合せを指定します。

***compound_expression*::=**



組込みファンクションを式として使用できます（6-10 ページの「ファンクション式」を参照してください）。ただし、複合式では、ファンクションの組合せによっては、適切でないものや拒否されるものもあります。たとえば、LENGTH ファンクションは集計ファンクション内では使用できません。

PRIOR 演算子は、階層問合せの CONNECT BY 句で使用されます。

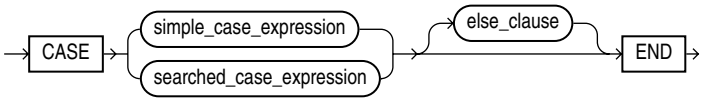
> **参照：** 4-3 ページの「演算子の優先順位」および 9-3 ページの「階層問合せ」を参照してください。

有効な複合式の例を次に示します。

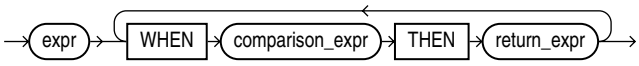
```
('CLARK' || 'SMITH')
LENGTH('MOOSE') * 57
SQRT(144) + 72
my_fun(TO_CHAR(sysdate,'DD-MMM-YY'))
```

CASE 式

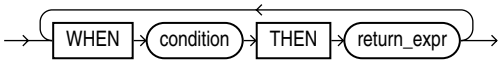
CASE 式を使用すると、プロシージャを起動せずに、SQL 文で IF ... THEN ... ELSE 論理を使用できます。構文は次のとおりです。



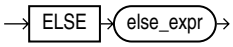
***simple_case_expression*::=**



***searched_case_expression*::=**



***else_clause*::=**



単純 CASE 式では、Oracle Database は、*expr* と *comparison_expr* が一致する最初の WHEN ... THEN の組合せを検索し、*return_expr* を戻します。WHEN ... THEN の組合せが条件に一致せず、ELSE 句が存在する場合、Oracle は *else_expr* を戻します。それ以外の場合、Oracle は NULL を戻します。*return_expr* および *else_expr* には、リテラル NULL を指定できません。

検索 CASE 式では、Oracle は、*condition* が真である項目を左から右へ検索し、*return_expr* を戻します。真である *condition* がなく、ELSE 句が存在する場合、Oracle は *else_expr* を戻します。それ以外の場合、Oracle は NULL を戻します。

Oracle Database では、**短絡評価**を使用します。単純 CASE 式の場合、データベースは、*comparison_expr* 値のいずれかと *expr* を比較する前にすべての *comparison_expr* 値を評価するのではなく、各 *comparison_expr* 値と *expr* を比較する前にのみ、各 *comparison_expr* 値を評価します。その結果、*expr* と等しい *comparison_expr* が見つかると、Oracle はその後の *comparison_expr* を評価しません。検索 CASE 式の場合、データベースは各 *condition* を評価してこれが真であるかどうかを判断します。ただし、ある *condition* が真の場合は、次の *condition* を評価しません。

単純 CASE 式では、*expr* およびすべての *comparison_expr* 値は、同じデータ型（CHAR、VARCHAR2、NCHAR、NVARCHAR2、NUMBER、BINARY_FLOAT または BINARY_DOUBLE）であるか、またはすべて数値データ型である必要があります。すべての式が数値データ型の場合、Oracle は、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

単純および検索 CASE 式では、すべての *return_expr* は、同じデータ型（CHAR、VARCHAR2、NCHAR、NVARCHAR2、NUMBER、BINARY_FLOAT または BINARY_DOUBLE）であるか、またはすべて数値データ型である必要があります。すべての戻り式が数値データ型の場合、Oracle は、数値の優先順位が最も高い引数を判断し、残りの引数をそのデータ型に暗黙的に変換して、そのデータ型を戻します。

1 つの CASE 式では、引数の最大数は 255 です。単純 CASE 式の初期式やオプションの ELSE 式を含むすべての式が、この上限の対象となります。WHEN ... THEN の各組は、2 つの引数としてカウントします。この上限を超えないように、CASE 式をネストし、*return_expr* 自体が CASE 式となるようにすることができます。

参照：

- 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。
- 数値の優先順位については、2-15 ページの「数値の優先順位」を参照してください。
- その他の CASE 式の書式については、5-35 ページの「COALESCE」および 5-112 ページの「NULLIF」を参照してください。
- CASE 式の様々な書式を使用した例は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

単純 CASE 式の例　次の文は、サンプル表 oe.customers のそれぞれの顧客について、クレジット利用限度額を、100 ドルの場合は「Low」、5,000 ドルの場合は「High」、それ以外の場合は「Medium」で表示します。

```
SELECT cust_last_name,
   CASE credit_limit WHEN 100 THEN 'Low'
   WHEN 5000 THEN 'High'
   ELSE 'Medium' END AS credit
   FROM customers
   ORDER BY cust_last_name, credit;

CUST_LAST_NAME       CREDIT
-------------------- ------
Adjani               Medium
Adjani               Medium
Alexander            Medium
Alexander            Medium
Altman               High
Altman               Medium
. . .
```

検索 CASE 式の例　次の文は、2,000 ドルを最少額の給与として、サンプル表 oe.employees の従業員の給与の平均を検出します。

```
SELECT AVG(CASE WHEN e.salary > 2000 THEN e.salary
   ELSE 2000 END) "Average Salary" FROM employees e;

Average Salary
--------------
    6461.68224
```

列式

後述の構文図で *column_expr* で示される列式は、*expr* の書式の一部です。列式は単純式、複合式、ファンクション式または式リストにできますが、列式に含めることができるのは、次の書式の式にかぎられます。

- 対象とする表（作成、変更または索引作成される表）の列
- 定数（文字列または数値）
- DETERMINISTIC ファンクション：SQL 組込みファンクションまたはユーザー定義ファンクションのいずれか

これ以外に、この章で説明されている書式で使用できるものはありません。また、PRIOR キーワードを使用した複合式および集計ファンクションはサポートされません。

列式は、次の目的で使用できます。

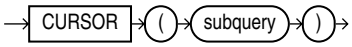
- ファンクション索引を作成する場合。
- 明示的または暗黙的に仮想列を定義する場合。仮想列を定義する際、*column_expr* の定義では、現行の文または以前の文ですでに定義されている対象の表の列のみを参照する必要があります。

列式を組み合せた要素は、決定的である必要があります。つまり、同じ入力値のセットによって、同じ出力値のセットが戻される必要があります。

> **参照：** これらの *expr* の書式の詳細は、6-3 ページの「単純式」、6-4 ページの「複合式」、6-10 ページの「ファンクション式」および 6-15 ページの「式のリスト」を参照してください。

CURSOR 式

CURSOR 式は、ネステッド・カーソルを戻します。この式の書式は、PL/SQL の REF CURSOR と同じで、REF CURSOR 引数としてファンクションに渡すことができます。



カーソル式が評価されるときに、ネステッド・カーソルは暗黙的にオープンされます。たとえば、カーソル式が SELECT 構文のリストにある場合、問合せによってフェッチされた各行に対して、ネステッド・カーソルがオープンされます。ネステッド・カーソルは、次の場合にのみクローズされます。

- ユーザーによって明示的にクローズされたとき
- 親カーソルが再実行されたとき
- 親カーソルがクローズされたとき
- 親カーソルが取り消されたとき
- 親カーソルの 1 つでのフェッチ時にエラーが発生したとき（クリーンアップの一部としてクローズされる）

CURSOR 式の制限事項： CURSOR 式には、次の制限事項があります。

- 囲まれる文が SELECT 文以外の文である場合、ネステッド・カーソルは、プロシージャの REF CURSOR 引数としてのみ表示されます。
- 囲まれる文が SELECT 文である場合、ネステッド・カーソルは、問合せ指定の一番外側の SELECT 構文のリスト、または別のネステッド・カーソルの一番外側の SELECT 構文のリストに表示されます。
- ネステッド・カーソルはビューに表示できません。
- ネステッド・カーソルに対して、BIND 操作および EXECUTE 操作は実行できません。

例 問合せの SELECT 構文のリストでの CURSOR 式の使用方法について、次に例を示します。

```
SELECT department_name, CURSOR(SELECT salary, commission_pct
   FROM employees e
   WHERE e.department_id = d.department_id)
   FROM departments d
   ORDER BY department_name;
```

ファンクションの引数としての CURSOR の使用方法について、次に例を示します。例では、まず、サンプル・スキーマ OE に REF CURSOR 引数を受け入れるファンクションを作成します。（イタリック体は、PL/SQL ファンクションの本体です。）

```
CREATE FUNCTION f(cur SYS_REFCURSOR, mgr_hiredate DATE)
   RETURN NUMBER IS
   emp_hiredate DATE;
```

```
   before number :=0;
   after number:=0;
begin
  loop
    fetch cur into emp_hiredate;
    exit when cur%NOTFOUND;
    if emp_hiredate > mgr_hiredate then
      after:=after+1;
    else
      before:=before+1;
    end if;
  end loop;
  close cur;
  if before > after then
    return 1;
  else
    return 0;
  end if;
end;
/
```

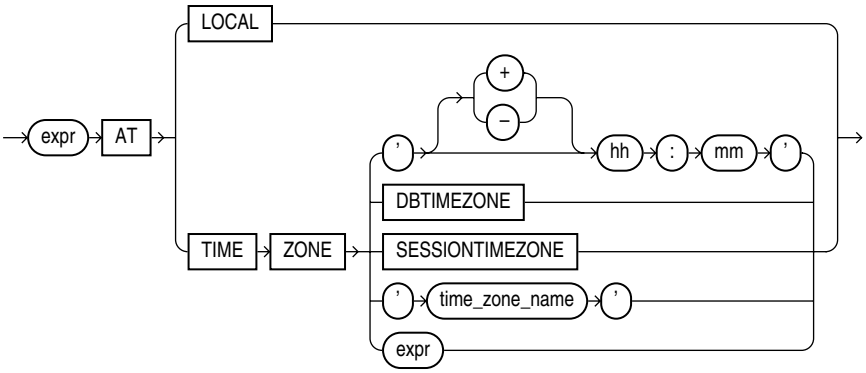
ファンクションには、カーソルおよび日付を指定できます。ファンクションは、カーソルが日付セットを戻す問合せであることを想定します。次の問合せでは、ファンクションを使用して、サンプル表 employees から、ほとんどの従業員がマネージャよりも前に雇用されているマネージャを検索します。

```
SELECT e1.last_name FROM employees e1
   WHERE f(
   CURSOR(SELECT e2.hire_date FROM employees e2
   WHERE e1.employee_id = e2.manager_id),
   e1.hire_date) = 1
   ORDER BY last_name;

LAST_NAME
-------------------------
Cambrault
De Haan
Higgins
Mourgos
Zlotkey
```

日時式

日時式は、日時データ型の値を戻します。

***datetime_expression*::=**



初期の *expr* は、データ型 TIMESTAMP、TIMESTAMP WITH TIME ZONE または TIMESTAMP WITH LOCAL TIME ZONE の値に評価される任意の式（スカラー副問合せ式を除く）です。DATE データ型はサポートされません。この *expr* 自体が *datetime_expression* である場合は、カッコで囲む必要があります。

2-21 ページの表 2-5 で定義される規則に従って、日時および期間を組み合せることができます。日時の値を戻す 3 つの組合せは、日時式で有効です。

AT LOCAL を指定すると、Oracle は現行のセッションのタイムゾーンを使用します。

AT TIME ZONE の設定は、次のように解析されます。

- 文字列 '(+|-)HH:MM': UTC のオフセットとしてタイムゾーンを指定します。
- DBTIMEZONE: Oracle は、データベースの作成中に（明示的またはデフォルトで）構築されたデータベース・タイムゾーンを使用します。
- SESSIONTIMEZONE: Oracle は、デフォルトで構築されたセッション・タイムゾーン、または最新の ALTER SESSION 文で構築されたセッション・タイムゾーンを使用します。
- *time_zone_name*: Oracle は、*time_zone_name* で指定されたタイムゾーンの *datetime_value_expr* を戻します。有効なタイムゾーン名を表示するには、V$TIMEZONE_NAMES 動的パフォーマンス・ビューに問合せを実行してください。

  注意： 夏時間機能には、タイムゾーン地域名が必要です。地域名は、2 つのタイムゾーン・ファイルに格納されます。デフォルトのタイムゾーン・ファイルは、パフォーマンスを最大にするために一般的なタイムゾーンのみの小さなファイルです。タイムゾーンがデフォルトのファイルに存在しない場合は、環境変数 ORA_TZFILE を使用して完全な（大きい）ファイルへのパスを指定するまで、夏時間はサポートされません。

  参照：

  - 両方のファイルに含まれるすべてのタイムゾーン地域名のリストは、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。
  - 動的パフォーマンス・ビューの詳細は、『Oracle Database リファレンス』を参照してください。

- *expr*: *expr* が有効なタイムゾーン書式で文字列を戻す場合、Oracle は、そのタイムゾーンで入力を戻します。そうでない場合は、エラーが戻ります。

例 次の例は、タイムゾーンの日時の値を別のタイムゾーンに変換します。

```
SELECT FROM_TZ(CAST(TO_DATE('1999-12-01 11:00:00',
      'YYYY-MM-DD HH:MI:SS') AS TIMESTAMP), 'America/New_York')
   AT TIME ZONE 'America/Los_Angeles' "West Coast Time"
   FROM DUAL;

West Coast Time
---------------------------------------------
01-DEC-99 08.00.00.000000 AM AMERICA/LOS_ANGELES
```

ファンクション式

組込み SQL ファンクションまたはユーザー定義ファンクションを式として使用できます。有効な組込みファンクション式の例を次に示します。

```
LENGTH('BLAKE')
ROUND(1234.567*43)
SYSDATE
```

> **参照：** 組込みファンクションの詳細は、5-2 ページの「SQL ファンクション」および 5-8 ページの「集計ファンクション」を参照してください。

ユーザー定義ファンクション式は、次のものへのコールを指定します。

- オラクル社が提供するパッケージにあるファンクション（『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照）
- ユーザー定義パッケージにあるファンクションまたはスタンドアロン・ユーザー定義ファンクション（5-249 ページの「ユーザー定義ファンクション」を参照）
- ユーザー定義ファンクションおよび演算子（15-32 ページの「CREATE OPERATOR」、14-48 ページの「CREATE FUNCTION」および『Oracle Database データ・カートリッジ開発者ガイド』を参照）

有効なユーザー定義ファンクション式の例を次に示します。

```
circle_area(radius)
payroll.tax_rate(empno)
hr.employees.comm_pct@remote(dependents, empno)
DBMS_LOB.getlength(column_name)
my_function(a_column)
```

式として使用されるユーザー定義ファンクションでは、位置表記法、名前付き表記法および複合表記法がサポートされています。たとえば、次の表記はすべて正しい表記です。

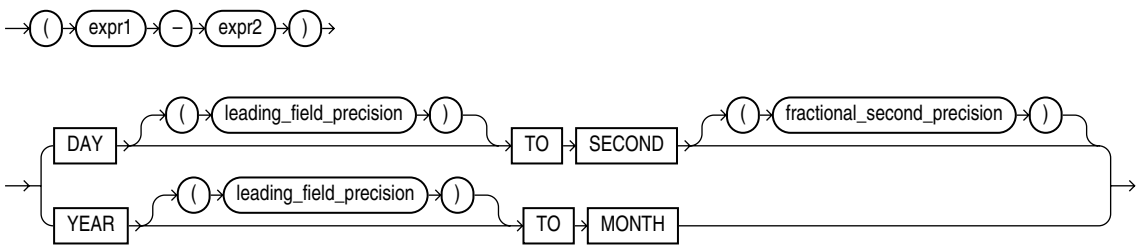
```
CALL my_function(arg1 => 3, arg2 => 4) ...

CALL my_function(3, 4) ...

CALL my_function(3, arg2 => 4) ...
```

ユーザー定義ファンクション式の制限事項： オブジェクト型または XMLType の引数をリモート・ファンクションおよびリモート・プロシージャに渡すことはできません。

期間式

期間式は、INTERVAL YEAR TO MONTH または INTERVAL DAY TO SECOND の値を戻します。

***interval_expression*::=**



式 *expr1* および *expr2* は、データ型 DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONE または TIMESTAMP WITH LOCAL TIME ZONE の値に評価される任意の式にすることができます。

2-21 ページの表 2-5 で定義される規則に従って、日時および期間を組み合せることができます。期間の値を戻す 6 つの組合せは、期間式で有効です。

leading_field_precision および *fractional_second_precision* は、0 ～ 9 の任意の整数です。DAY または YEAR のいずれかで *leading_field_precision* を省略すると、Oracle Database はデフォルト値である 2 を使用します。
秒で *fractional_second_precision* を省略すると、データベースはデフォルト値である 6 を使用します。問合せで戻された値にデフォルトの精度を超える桁数が含まれる場合、Oracle Database はエラーを戻します。したがって、問合せで戻されるとわかっている値以上の精度を指定することをお薦めします。

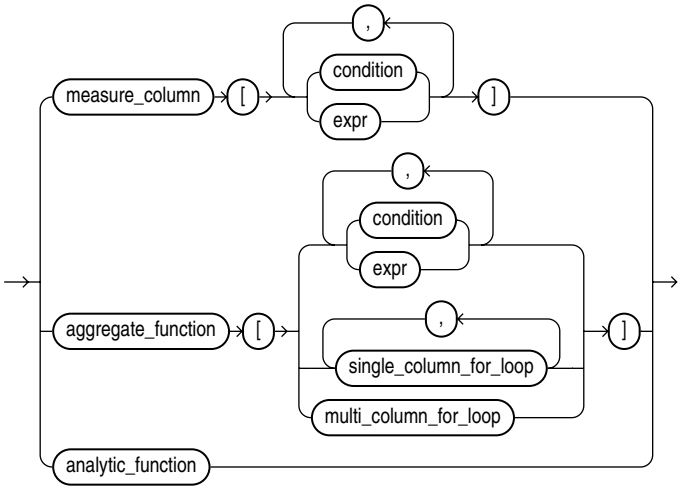
たとえば、次の文は、システム・タイムスタンプ（日時の値）からサンプル表 orders の order_date 列の値（別の日時の値）を減算して、期間値の式を戻します。一番古い注文が発注されたのが何日前かわからないため、DAY の先行フィールド精度に最大値である 9 を指定します。

```
SELECT (SYSTIMESTAMP - order_date) DAY(9) TO SECOND FROM orders
   WHERE order_id = 2458;
```

モデル式

モデル式は、SELECT 文の *model_clause* でのみ、さらにモデル・ルールの右側でのみ使用されます。モデル式は、*model_clause* で事前定義されたメジャー列のセルに値を戻します。詳細は、19-25 ページの「model_clause」を参照してください。

***model_expression*::=**



モデル式でメジャー列を指定する場合、指定する任意の条件と式は単一の値に変換される必要があります。

モデル式で集計ファンクションを指定する場合、ファンクションの引数は *model_clause* で事前定義したメジャー列です。集計ファンクションは、モデル・ルールの右側でのみ使用できます。

モデル・ルールの右側で分析ファンクションを指定すると、*model_clause* に複雑な計算を直接表記することができます。モデル式で分析ファンクションを使用するときには、次の制限事項が適用されます。

- 分析ファンクションは、UPDATE ルールのみで使用できます。
- モデル・ルールの左側に FOR ループまたは ORDER BY 句が含まれている場合は、モデル・ルールの右側で分析ファンクションを指定できません。
- 分析ファンクションの OVER 句の引数に集計を含めることはできません。
- 分析ファンクションの OVER 句の前の引数にセル参照を含めることはできません。

> **参照：** モデル・ルールの右側で分析ファンクションを使用する例は、19-36 ページの「MODEL 句の例 :」を参照してください。

expr 自体がモデル式である場合、**ネストしたセル参照**と呼ばれます。ネストしたセル参照には、次の制限事項が適用されます。

- ネストは 1 レベルのみ可能です。
- ネストしたセル参照は、単一セル参照である必要があります。
- *model_rules_clause* に AUTOMATIC ORDER が指定されているとき、ネストしたセル参照で使用されるメジャーがスプレッドシート句内のいずれのセルについても更新されない場合のみ、ネストしたセル参照をモデル・ルールの左側で使用できます。

後述するモデル式は、次に示す SELECT 文の *model_clause* に基づいています。

```
SELECT country,prod,year,s
  FROM sales_view_ref
  MODEL
    PARTITION BY (country)
    DIMENSION BY (prod, year)
    MEASURES (sale s)
    IGNORE NAV
    UNIQUE DIMENSION
    RULES UPSERT SEQUENTIAL ORDER
    (
      s[prod='Mouse Pad', year=2000] =
        s['Mouse Pad', 1998] + s['Mouse Pad', 1999],
      s['Standard Mouse', 2001] = s['Standard Mouse', 2000]
    )
  ORDER BY country, prod, year;
```

次のモデル式は、記号表記法を使用して単一セル参照を表しています。この式は、2000 年のマウス・パッドの売上を示します。

```
s[prod='Mouse Pad',year=2000]
```

次のモデル式は、CV ファンクションを使用して、位置表記法を使用する複数セル参照を表しています。この式は、prod ディメンション列の現在値の 2001 年における売上を示します。

```
s[CV(prod), 2001]
```

次のモデル式は、集計ファンクションを表しています。この式は、year ディメンション列の現在の値 -2 から year ディメンション列の現在の値 -1 までの年のマウス・パッドの売上合計を示します。

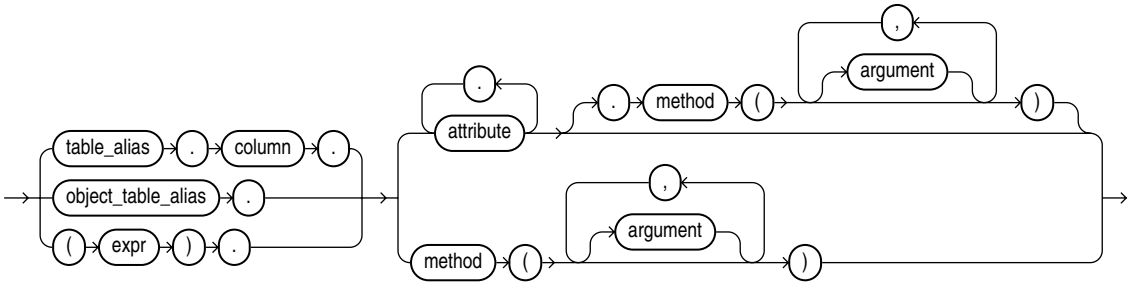
```
SUM(s)['Mouse Pad',year BETWEEN CV()-2 AND CV()-1]
```

> **参照：** 5-51 ページの「CV」および 19-25 ページの「model_clause」を参照してください。

オブジェクト・アクセス式

オブジェクト・アクセス式は、属性の参照およびメソッドの起動を指定します。

***object_access_expression*::=**



column パラメータはオブジェクトまたは REF 列です。*expr* を指定する場合、オブジェクト型に変換する必要があります。

ある型のメンバー・ファンクションが SQL 文のコンテキストでコールされると、SELF 引数が NULL の場合に、Oracle は NULL を戻し、ファンクションは起動されません。

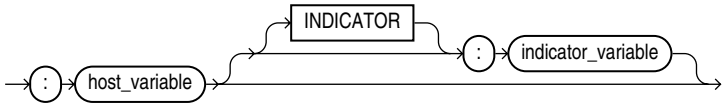
例　次の例では、サンプル・オブジェクト型 oe.order_item_typ に基づく表を作成し、オブジェクト列属性から更新および検索する方法を示します。

```
CREATE TABLE short_orders (
   sales_rep VARCHAR2(25), item order_item_typ);

UPDATE short_orders s SET sales_rep = 'Unassigned';

SELECT o.item.line_item_id, o.item.quantity FROM short_orders o;
```

プレースホルダ式

プレースホルダ式は、SQL 文における位置を指定します。この位置に対して、第三世代言語のバインド変数によって値が指定されます。プレースホルダ式には、オプションで標識変数を指定できます。この書式の式は、埋込み SQL 文または Oracle Call Interface（OCI）プログラムで処理される SQL 文のみで指定できます。

***placeholder_expression*::=**



有効なプレースホルダ式の例を次に示します。

```
:employee_name INDICATOR :employee_name_indicator_var
:department_location
```

スカラー副問合せ式

スカラー副問合せ式は、1 つの行から 1 つの列値のみを戻す副問合せです。スカラー副問合せ式の値は、副問合せの SELECT 構文リスト項目の値です。副問合せが 0 行を戻す場合、スカラー副問合せ式の値は NULL です。副問合せが 2 つ以上の行を戻す場合、Oracle はエラーを戻します。

スカラー副問合せ式は、式（*expr*）をコールするほとんどの構文で使用できます。すべての場合、スカラー副問合せは、その構文の場所がすでにカッコ内であっても（組込みファンクションの引数として使用されている場合など）、独自のカッコで囲む必要があります。

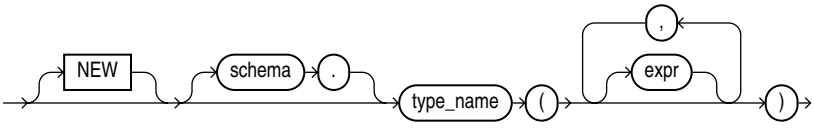
スカラー副問合せは、次の場所では無効です。

- 列のデフォルト値
- クラスタのハッシュ式
- DML 文 RETURNING 句
- ファンクション索引の基礎
- CHECK 制約
- GROUP BY 句
- CREATE PROFILE などの問合せに関連しない文

型コンストラクタ式

型コンストラクタ式は、コンストラクタ・メソッドへのコールを指定します。型コンストラクタの引数は、任意の式です。型コンストラクタは、ファンクションが起動されるすべての場所で起動できます。

***type_constructor_expression*::=**



NEW キーワードによって、コレクション型ではなくオブジェクトに、コンストラクタが適用されます。これによって、適切なコンストラクタを起動して新しいオブジェクトを作成するように Oracle に指示します。NEW キーワードはオプションですが、指定することをお薦めします。

type_name が**オブジェクト型**の場合、最初の引数の値の型がオブジェクト型の最初の属性と一致する値を取り、2 番目の引数の値の型がオブジェクト型の 2 番目の属性と一致するというように、式は順序付けられたリストになっている必要があります。コンストラクタの引数の合計数は、オブジェクト型の属性の合計数と一致する必要があります。

type_name が **VARRAY 型**または**ネストした表型**の場合、式のリストには 0 個以上の引数を含めることができます。引数が 0 個の場合は、空コレクションの構造であることを示します。それ以外の場合は、各引数が、型がコレクション型の要素型である要素値に対応します。

型コンストラクタの起動の制限事項　型コンストラクタ・メソッドの起動では、オブジェクト型に 1000 以上の属性がある場合でも、指定できるパラメータの数（*expr*）は最大で 999 です。この制限は、コンストラクタが SQL からコールされる場合にのみ適用されます。PL/SQL からコールされる場合には、PL/SQL の制限が適用されます。

> **参照：**　コンストラクタ・メソッドの詳細は、『Oracle Database オブジェクト・リレーショナル開発者ガイド』を参照してください。型コンストラクタへのコールに関する PL/SQL の制限の詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

式の例　次の例では、サンプル・スキーマ oe の cust_address_typ 型を使用して、コンストラクタ・メソッドへのコールに含まれる式の使用方法を示します（PL/SQL はイタリック体で示しています）。

```
CREATE TYPE address_book_t AS TABLE OF cust_address_typ;
DECLARE
   myaddr cust_address_typ := cust_address_typ(
     '500 Oracle Parkway', 94065, 'Redwood Shores', 'CA','USA');
   alladdr address_book_t := address_book_t();
BEGIN
   INSERT INTO customers VALUES (
      666999, 'Joe', 'Smith', myaddr, NULL, NULL, NULL, NULL,
      NULL, NULL, NULL, NULL, NULL, NULL, NULL);
END;
/
```

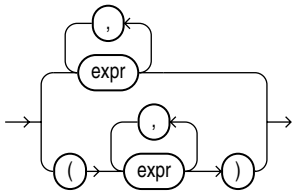
副問合せ例　次の例では、サンプル・スキーマ oe の warehouse_typ 型を使用して、コンストラクタ・メソッドへのコールに含まれる副問合せの使用方法を示します。

```
CREATE TABLE warehouse_tab OF warehouse_typ;

INSERT INTO warehouse_tab
   VALUES (warehouse_typ(101, 'new_wh', 201));

CREATE TYPE facility_typ AS OBJECT (
   facility_id NUMBER,
   warehouse_ref REF warehouse_typ);

CREATE TABLE buildings (b_id NUMBER, building facility_typ);

INSERT INTO buildings VALUES (10, facility_typ(102,
   (SELECT REF(w) FROM warehouse_tab w
      WHERE warehouse_name = 'new_wh')));

SELECT b.b_id, b.building.facility_id "FAC_ID",
   DEREF(b.building.warehouse_ref) "WH" FROM buildings b;

      B_ID     FAC_ID WH(WAREHOUSE_ID, WAREHOUSE_NAME, LOCATION_ID)
---------- ---------- -------------------------------------------
        10        102 WAREHOUSE_TYP(101, 'new_wh', 201)
```

式のリスト

式のリストは、その他の式の組合せです。

***expression_list*::=**



式のリストは、比較条件、メンバーシップ条件、および問合せと副問合せの GROUP BY 句に指定できます。比較条件またはメンバーシップ条件内の式のリストは、**行値コンストラクタ**または**行コンストラクタ**と呼ばれる場合があります。

比較条件およびメンバーシップ条件は、WHERE 句の条件として指定します。これらの条件には、カンマで区切られた 1 つ以上の式、または 1 つ以上の式の集合（各集合には、カンマで区切られた式が 1 つ以上含まれる）を含めることができます。式の集合が複数存在する場合は、次のように指定します。

- 各集合をカッコで区切ります。
- 各集合に含まれる式の数を同じにします。
- 各集合内の式の数を、比較条件の演算子の前またはメンバーシップ条件の IN キーワードの前にある式の数と同じにします。

カンマで区切られた式のリストには、最大 1000 個の式を指定できます。カンマで区切られた式の集合のリストには、任意数の式の集合を含めることができますが、各集合に指定できる式は最大 1000 個です。

次に、条件に含める有効な式のリストを示します。

```
(10, 20, 40)
('SCOTT', 'BLAKE', 'TAYLOR')
( ('Guy', 'Himuro', 'GHIMURO'),('Karen', 'Colmenares', 'KCOLMENA') )
```

3 番目の例の場合、各集合の式の数は、条件の最初の部分の式の数と同じである必要があります。たとえば、次のようになります。

```
SELECT * FROM employees
  WHERE (first_name, last_name, email) IN
  (('Guy', 'Himuro', 'GHIMURO'),('Karen', 'Colmenares', 'KCOLMENA'))
```

参照： 7-4 ページの「比較条件」および 7-22 ページの「IN 条件」条件を参照してください。

単純な GROUP BY 句では、式のリストの書式として、次のいずれかを使用できます。

```
SELECT department_id, MIN(salary) min, MAX(salary) max FROM employees
   GROUP BY department_id, salary
   ORDER BY department_id, min, max;

SELECT department_id, MIN(salary) min, MAX(salary) max FROM employees
   GROUP BY (department_id, salary)
   ORDER BY department_id, min, max;
```

ROLLUP、CUBE、および GROUP BY 句の GROUPING SETS 句では、個々の式と同じリストにある式の集合を組み合せることができます。次に、1 つの SQL 文に含まれている有効なグルーピング・セットと式のリストを示します。

```
SELECT
prod_category, prod_subcategory, country_id, cust_city, count(*)
   FROM  products, sales, customers
   WHERE sales.prod_id = products.prod_id
   AND sales.cust_id=customers.cust_id
   AND sales.time_id = '01-oct-00'
   AND customers.cust_year_of_birth BETWEEN 1960 and 1970
GROUP BY GROUPING SETS
  (
   (prod_category, prod_subcategory, country_id, cust_city),
   (prod_category, prod_subcategory, country_id),
   (prod_category, prod_subcategory),
    country_id
  )
ORDER BY prod_category, prod_subcategory, country_id, cust_city;
```

参照： 「SELECT」（19-4 ページ）

7

条件

条件は、1 つ以上の式と論理（ブール）演算子の組合せで指定し、TRUE、FALSE または UNKNOWN のいずれかの値を戻します。

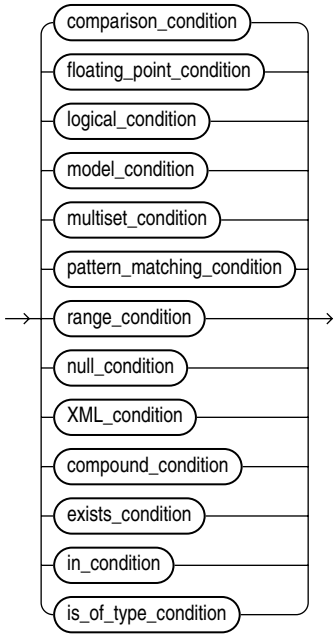
この章では、次の内容を説明します。

- SQL 条件
- 比較条件
- 浮動小数点条件
- 論理条件
- モデル条件
- 多重集合条件
- パターン一致条件
- NULL 条件
- XML 条件
- 複合条件
- BETWEEN 条件
- EXISTS 条件
- IN 条件
- IS OF type 条件

SQL 条件

条件には、次の構文で示すとおり、複数の書式があります。

***condition*::=**



Oracle Text がインストールされている場合、CONTAINS、CATSEARCH、MATCHES など、この製品に含まれる組込み条件を使用して条件を作成できます。Oracle Text 要素の詳細は、『Oracle Text リファレンス』を参照してください。

Oracle Expression Filter を使用している場合、この製品に含まれる組込み EVALUATE 演算子を使用して条件を作成できます。詳細は、『Oracle Database ルール・マネージャおよび式フィルタ開発者ガイド』を参照してください。

次の項では、様々な書式の条件を説明します。SQL 文に *condition* が含まれる場合は、適切な条件構文を使用する必要があります。

条件は、次の文の WHERE 句で使用できます。

- DELETE
- SELECT
- UPDATE

また、SELECT 文の次の句で使用することもできます。

- WHERE
- START WITH
- CONNECT BY
- HAVING

注意： NLS_COMP と NLS_SORT の設定を組み合せた値によって、文字をソートおよび比較するルールが決まります。ご使用のデータベースの NLS_COMP に LINGUISTIC が設定されている場合、この章のエンティティはすべて NLS_SORT パラメータによって指定されるルールに従って解釈されます。NLS_COMP が LINGUISTIC に設定されていない場合、ファンクションは NLS_SORT の設定に関係なく解釈されます。NLS_SORT は、明示的に設定できます。明示的に設定されていない場合は、NLS_LANGUAGE から導出されます。これらの設定の詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

条件は、論理データ型であるともいえます。ただし、Oracle Database で、正式にこのようなデータ型をサポートしているわけではありません。

次のような単純な条件は、常に TRUE に評価されます。

```
1 = 1
```

次のやや複雑な条件は、salary の値を salary*commission_pct の値に加算し（NULL は 0 で置き換える）、その合計が定数 25000 より大きいかどうかを判断します。

```
NVL(salary, 0) + NVL(salary + (salary*commission_pct, 0) > 25000)
```

論理条件を使用すると、複数の条件を単一の条件に結合できます。たとえば、次のように AND 条件を使用して 2 つの条件を結合できます。

```
(1 = 1) AND (5 < 7)
```

有効な条件の例を次に示します。

```
name = 'SMITH'
employees.department_id = departments.department_id
hire_date > '01-JAN-88'
job_id IN ('SA_MAN', 'SA_REP')
salary BETWEEN 5000 AND 10000
commission_pct IS NULL AND salary = 2100
```

参照： 文に指定する条件の制限については、第 10 章～第 19 章にある各文の説明を参照してください。

条件の優先順位

優先順位とは、同じ式の中の異なる条件を Oracle Database が評価する順序を意味します。複数の条件を含む式を評価するとき、Oracle は優先順位の高い条件を評価した後で、優先順位の低い条件を評価します。優先順位の等しい条件は、式の中で左から右に評価されます。

表 7-1 に、SQL 条件を優先順位の高い方から順に示します。同じ行に示されている条件の優先順位は同じです。表に示すとおり、Oracle は条件の前に演算子を評価します。

表 7-1 SQL 条件の優先順位

条件の種類	用途
SQL 演算子は、SQL 条件の前に評価されます。	4-3 ページの「演算子の優先順位」を参照してください。
=、!=、<、>、<=、>=	比較
IS [NOT] NULL、LIKE、[NOT] BETWEEN、[NOT] IN、EXISTS、IS OF *type*	比較
NOT	指数、論理否定
AND	論理積
OR	論理和

比較条件

比較条件は、2 つの式を比較します。比較の結果は、TRUE、FALSE または NULL になります。

ラージ・オブジェクト（LOB）は、比較条件ではサポートされていません。ただし、CLOB データの比較では、PL/SQL プログラムを使用できます。

数式を比較する場合、Oracle は数値の優先順位を使用して、その条件が NUMBER、BINARY_FLOAT または BINARY_DOUBLE のうちどの値を比較するかを判断します。数値の優先順位の詳細は、2-15 ページの「数値の優先順位」を参照してください。

非スカラー型の 2 つのオブジェクトが比較可能なのは、これらが同じ名前付きの型であり、要素が 1 対 1 で対応している場合です。また、ユーザー定義オブジェクト型のネストした表では、要素が比較可能な場合も、等式や IN 条件で使用する MAP メソッドが定義されている必要があります。

参照：

- MAP メソッドの詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。
- PL/SQL のユーザー定義オブジェクト型の比較要件は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

表 7-2 に、比較条件を示します。

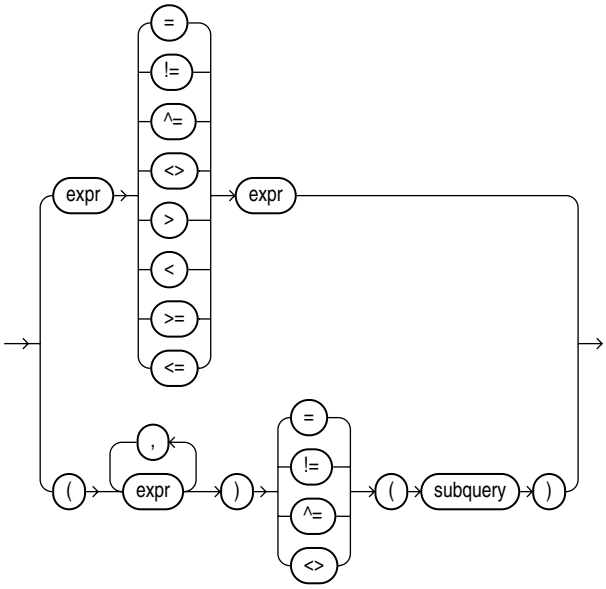
表 7-2 比較条件

条件の種類	用途	例
=	等価性をテストします。	SELECT * FROM employees WHERE salary = 2500 ORDER BY employee_id;
!= ^= < > ¬=	不等性をテストします。プラットフォームによっては、一部の不等号条件を使用できない場合もあります。	SELECT * FROM employees WHERE salary != 2500 ORDER BY employee_id;
> <	大 / 小をテストします。	SELECT * FROM employees WHERE salary > 2500 ORDER BY employee_id; SELECT * FROM employees WHERE salary < 2500 ORDER BY employee_id;
>= <=	以上 / 以下をテストします。	SELECT * FROM employees WHERE salary >= 2500 ORDER BY employee_id; SELECT * FROM employees WHERE salary <= 2500 ORDER BY employee_id;
ANY SOME	リスト内の各値または問合せによって戻される各値と、ある値を比較します。=、!=、>、<、<=、>= のいずれかを先に指定する必要があります。1 つ以上の値を戻す任意の式または副問合せを後に指定できます。 問合せによって行が戻されない場合には、FALSE と評価されます。	SELECT * FROM employees WHERE salary = ANY (SELECT salary FROM employees WHERE department_id = 30) ORDER BY employee_id;
ALL	リスト内のすべての値または問合せによって戻されるすべての値と、ある値を比較します。=、!=、>、<、<=、>= のいずれかを先に指定する必要があります。1 つ以上の値を戻す任意の式または副問合せを後に指定できます。 問合せによって行が戻されない場合には、TRUE と評価されます。	SELECT * FROM employees WHERE salary >= ALL (1400, 3000) ORDER BY employee_id;

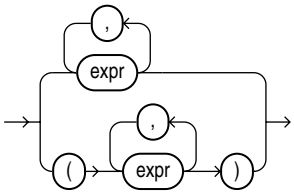
単純比較条件

単純比較条件は、式または副問合せの結果の比較方法を指定します。

***simple_comparison_condition*::=**



***expression_list*::=**



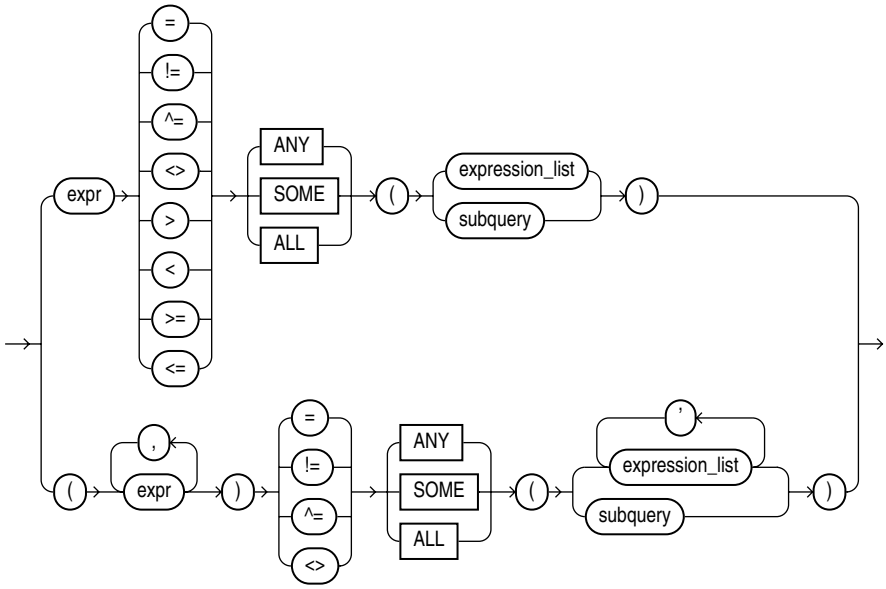
この条件の下の方の書式（演算子の左辺に複数の式を指定する書式）を使用する場合は、*expression_list* の下の方の書式を使用し、副問合せによって戻される値は、*expression_list* 内の式と同じ数とデータ型で構成されている必要があります。

> **参照：** 式の組合せの詳細は、6-15 ページの「式のリスト」を参照してください。副問合せの詳細は、19-4 ページの「SELECT」を参照してください。

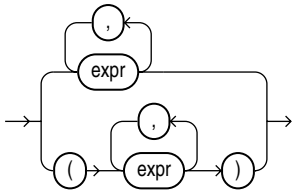
グループ比較条件

グループ比較条件は、リストまたは副問合せ内の任意またはすべてのメンバーの比較方法を指定します。

***group_comparison_condition*::=**



***expression_list*::=**



この条件の上の方の書式（演算子の左辺に 1 つの式を指定する書式）を使用する場合は、*expression_list* の上の方の書式を使用する必要があります。この条件の下の方の書式（演算子の左辺に複数の式を指定する書式）を使用する場合は、*expression_list* の下の方の書式を使用し、各 *expression_list* 内の式は、演算子の左辺にある式と同じ数とデータ型で構成されている必要があります。

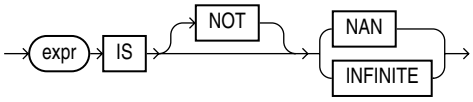
参照：

- 「式のリスト」（6-15 ページ）
- 「SELECT」（19-4 ページ）

浮動小数点条件

浮動小数点条件では、式が無限か、または演算の未定義の結果（非数値（NaN））かを判断します。

***floating_point_conditions*::=**



どちらの浮動小数点条件でも、*expr* は、数値データ型、または数値データ型に暗黙的に変換可能な任意のデータ型に解決される必要があります。表 7-3 に、浮動小数点条件を示します。

表 7-3 浮動小数点数条件

条件の種類	操作	例
IS [NOT] NAN	NOT が指定されているときは、*expr* が特殊な値である NaN の場合に TRUE を戻します。NOT が指定されているときは、*expr* が特殊な値である NaN ではない場合に TRUE を戻します。	SELECT COUNT(*) FROM employees WHERE commission_pct IS NOT NAN;
IS [NOT] INFINITE	NOT が指定されていないときは、*expr* が特殊な値である +INF または -INF の場合に TRUE を戻します。NOT が指定されているときは、*expr* が特殊な値である +INF でも -INF でもない場合に TRUE を戻します。	SELECT last_name FROM employees WHERE salary IS NOT INFINITE;

参照：

- Oracle での浮動小数点数の実装の詳細は、2-13 ページの「浮動小数点数」を参照してください。
- Oracle での浮動小数点データ型の変換方法の詳細は、2-39 ページの「暗黙的なデータ変換」を参照してください。

論理条件

論理条件は、2 つのコンポーネント条件の結果を組み合せ、それらに基づいて単一の結果を生成したり、単一の条件の結果を反転させます。表 7-4 に、論理条件を示します。

表 7-4 論理条件

条件の種類	操作	例
NOT	後続する条件が FALSE の場合に TRUE を戻します。TRUE の場合には FALSE を戻します。UNKNOWN の場合には UNKNOWN を戻します。	SELECT * FROM employees WHERE NOT (job_id IS NULL) ORDER BY employee_id; SELECT * FROM employees WHERE NOT (salary BETWEEN 1000 AND 2000) ORDER BY employee_id;
AND	コンポーネントの条件が両方とも TRUE の場合に TRUE を戻します。どちらかが FALSE の場合には FALSE を戻します。それ以外の場合は UNKNOWN を戻します。	SELECT * FROM employees WHERE job_id = 'PU_CLERK' AND department_id = 30 ORDER BY employee_id;
OR	コンポーネントの条件のどちらかが TRUE の場合に TRUE を戻します。両方とも FALSE の場合は FALSE を戻します。それ以外の場合は UNKNOWN を戻します。	SELECT * FROM employees WHERE job_id = 'PU_CLERK' OR department_id = 10 ORDER BY employee_id;

表 7-5 に、式に NOT 条件を適用した結果を示します。

表 7-5 NOT 真理値表

--	TRUE	FALSE	UNKNOWN
NOT	FALSE	TRUE	UNKNOWN

表 7-6 に、2 つの式に AND 条件を組み合せた結果を示します。

表 7-6 AND 真理値表

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

たとえば、次の SELECT 文の WHERE 句は、AND 論理条件を使用して、1989 年より前に入社し、給与が 2,500 ドルを超える従業員のみを戻します。

```
SELECT * FROM employees
WHERE hire_date < TO_DATE('01-JAN-1989', 'DD-MON-YYYY')
  AND salary > 2500
  ORDER BY employee_id;
```

表 7-7 に、2 つの式に OR を適用した結果を示します。

表 7-7 OR 真理値表

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

たとえば、次の問合せは、歩合率が 40% または給与が 20,000 ドルを超える従業員を戻します。

```
SELECT employee_id FROM employees
   WHERE commission_pct = .4 OR salary > 20000
   ORDER BY employee_id;
```

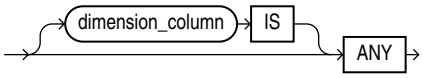
モデル条件

モデル条件は、SELECT 文の MODEL 句でのみ使用できます。

IS ANY 条件

IS ANY 条件は、SELECT 文の *model_clause* でのみ使用できます。この条件を使用して、ディメンション列のすべての値（NULL を含む）を修飾します。

***is_any_condition*::=**



この条件は、常にブール値の TRUE を戻し、列内のすべての値を修飾します。

> **参照：** 19-25 ページの「model_clause」および 6-11 ページの「モデル式」を参照してください。

例

次の例は、2000 年の各製品の売上を 0（ゼロ）に設定します。

```
SELECT country, prod, year, s
  FROM sales_view_ref
  MODEL
    PARTITION BY (country)
    DIMENSION BY (prod, year)
    MEASURES (sale s)
    IGNORE NAV
    UNIQUE DIMENSION
    RULES UPSERT SEQUENTIAL ORDER
    (
      s[ANY, 2000] = 0
    )
  ORDER BY country, prod, year;

COUNTRY      PROD                                     YEAR          S
----------   ----------------------------------    --------   ---------
France       Mouse Pad                                1998     2509.42
France       Mouse Pad                                1999     3678.69
France       Mouse Pad                                2000           0
France       Mouse Pad                                2001     3269.09
France       Standard Mouse                           1998     2390.83
```

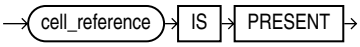
```
France          Standard Mouse                          1999       2280.45
France          Standard Mouse                          2000             0
France          Standard Mouse                          2001       2164.54
Germany         Mouse Pad                               1998       5827.87
Germany         Mouse Pad                               1999       8346.44
Germany         Mouse Pad                               2000             0
Germany         Mouse Pad                               2001       9535.08
Germany         Standard Mouse                          1998       7116.11
Germany         Standard Mouse                          1999       6263.14
Germany         Standard Mouse                          2000             0
Germany         Standard Mouse                          2001       6456.13

16 rows selected.
```

この例では、ビュー sales_view_ref が必要です。このビューを作成する方法については、19-36 ページの「MODEL 句の例 :」を参照してください。

IS PRESENT 条件

***is_present_condition*::=**

IS PRESENT 条件は、SELECT 文の *model_clause* でのみ使用できます。この条件を使用すると、*model_clause* の実行前に、参照されるセルが存在するかどうかをテストできます。



この条件は、*model_clause* の実行前にセルが存在する場合に TRUE、存在しない場合に FALSE を戻します。

> **参照：** 19-25 ページの「model_clause」および 6-11 ページの「モデル式」を参照してください。

例

次の例では、1999 年のマウス・パッドの売上が存在する場合、2000 年のマウス・パッドの売上が 1999 年のマウス・パッドの売上に設定されます。1999 年のマウス・パッドの売上が存在しない場合は、2000 年のマウス・パッドの売上は 0（ゼロ）に設定されます。

```
SELECT country, prod, year, s
  FROM sales_view_ref
  MODEL
    PARTITION BY (country)
    DIMENSION BY (prod, year)
    MEASURES (sale s)
    IGNORE NAV
    UNIQUE DIMENSION
    RULES UPSERT SEQUENTIAL ORDER
    (
      s['Mouse Pad', 2000] =
        CASE WHEN s['Mouse Pad', 1999] IS PRESENT
             THEN s['Mouse Pad', 1999]
             ELSE 0
        END
    )
  ORDER BY country, prod, year;

COUNTRY       PROD                                    YEAR          S
----------    ----------------------------------      --------   ---------
France        Mouse Pad                                  1998     2509.42
France        Mouse Pad                                  1999     3678.69
France        Mouse Pad                                  2000     3678.69
France        Mouse Pad                                  2001     3269.09
```

```
France          Standard Mouse                          1998     2390.83
France          Standard Mouse                          1999     2280.45
France          Standard Mouse                          2000     1274.31
France          Standard Mouse                          2001     2164.54
Germany         Mouse Pad                               1998     5827.87
Germany         Mouse Pad                               1999     8346.44
Germany         Mouse Pad                               2000     8346.44
Germany         Mouse Pad                               2001     9535.08
Germany         Standard Mouse                          1998     7116.11
Germany         Standard Mouse                          1999     6263.14
Germany         Standard Mouse                          2000     2637.31
Germany         Standard Mouse                          2001     6456.13
16 rows selected.
```

この例では、ビュー sales_view_ref が必要です。このビューを作成する方法については、19-36 ページの「MODEL 句の例 :」を参照してください。

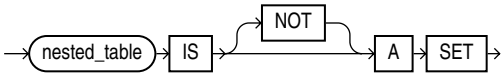
多重集合条件

多重集合条件は、ネストした表を様々な側面からテストします。

IS A SET 条件

IS A SET 条件を使用すると、指定されたネストした表が一意の要素で構成されているかどうかをテストできます。この条件は、ネストした表が NULL の場合に、NULL を戻します。それ以外では、ネストした表がセットである場合（ネストした表の長さが 0（ゼロ）の場合も含む）に TRUE を戻し、その他の場合は FALSE を戻します。

***is_a_set_conditions*::=**



例

次の例は、表 customers_demo から、cust_address_ntab というネストした表の列に一意の要素が含まれる行を選択します。

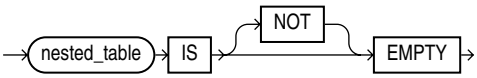
```
SELECT customer_id, cust_address_ntab
  FROM customers_demo
  WHERE cust_address_ntab IS A SET
  ORDER BY customer_id;

CUSTOMER_ID CUST_ADDRESS_NTAB(STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE, COUNTRY_ID)
-----------------------------------------------------------------------------------------
        101 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('514 W Superior St', '46901', 'Kokomo', 'IN', 'US'))
        102 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('2515 Bloyd Ave', '46218', 'Indianapolis', 'IN', 'US'))
        103 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('8768 N State Rd 37', '47404', 'Bloomington', 'IN', 'US'))
        104 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('6445 Bay Harbor Ln', '46254', 'Indianapolis', 'IN', 'US'))
        105 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('4019 W 3Rd St', '47404', 'Bloomington', 'IN', 'US'))
```

この例では、表 customers_demo と、データを含むネストした表の列が 1 つ必要です。この表およびネストした表の列を作成する方法については、4-6 ページの「MULTISET 演算子」を参照してください。

IS EMPTY 条件

IS [NOT] EMPTY 条件を使用すると、指定されたネストした表が空かどうかをテストできます。単一の値（NULL）で構成されたネストした表は、空のネストした表とはみなされません。

***is_empty_conditions*::=**



この条件は、コレクションが空の場合に IS EMPTY 条件にブール値 TRUE を戻し、コレクションが空でない場合に IS NOT EMPTY 条件にブール値 TRUE を戻します。ネストした表または VARRAY に NULL を指定すると、結果は NULL になります。

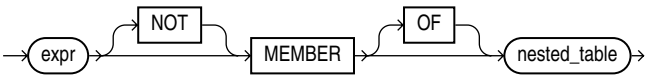
例

次の例は、サンプル表 pm.print_media から、ad_textdocs_ntab というネストした表の列が空ではない行を選択します。

```
SELECT product_id, TO_CHAR(ad_finaltext) AS text
   FROM print_media
   WHERE ad_textdocs_ntab IS NOT EMPTY
   ORDER BY product_id, text;
```

MEMBER 条件

***member_condition*::=**



member_condition は、要素がネストした表のメンバーかどうかをテストするメンバーシップ条件です。*expr* が指定されたネストした表または VARRAY のメンバーと等しい場合、TRUE が戻されます。*expr* が NULL またはネストした表が空の場合、NULL が戻されます。

- *expr* の型は、ネストした表の要素型と同じである必要があります。
- OF キーワードはオプションであり、条件の動作に影響しません。
- NOT キーワードはブール出力を逆にします。*expr* が指定されたネストした表のメンバーである場合、FALSE が戻されます。
- ネストした表の要素の型は、比較可能な型である必要があります。スカラー型以外の型の比較の可能性は、7-4 ページの「比較条件」を参照してください。

例

次の例は、表 customers_demo から、cust_address_ntab というネストした表の列に、WHERE 句で指定された値が含まれる行を選択します。

```
SELECT customer_id, cust_address_ntab
  FROM customers_demo
  WHERE cust_address_typ('8768 N State Rd 37', 47404,
                         'Bloomington', 'IN', 'US')
  MEMBER OF cust_address_ntab
  ORDER BY customer_id;

CUSTOMER_ID CUST_ADDRESS_NTAB(STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE, COUNTRY_ID)
----------- --------------------------------------------------------------------------------
        103 CUST_ADDRESS_TAB_TYP(CUST_ADDRESS_TYP('8768 N State Rd 37', '47404', 'Bloomington', 'IN', 'US'))
```

この例では、表 customers_demo と、データを含むネストした表の列が 1 つ必要です。この表およびネストした表の列を作成する方法については、4-6 ページの「MULTISET 演算子」を参照してください。

SUBMULTISET 条件

SUBMULTISET 条件は、指定されたネストした表が他の指定されたネストした表のサブ多重集合かどうかをテストします。

この演算子はブール値を戻します。*nested_table1* が *nested_table2* のサブ多重集合である場合、TRUE が戻されます。次の条件のいずれかが発生する場合、*nested_table1* は *nested_table2* のサブ多重集合となります。

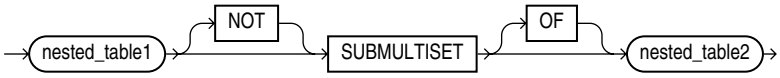
- *nested_table1* が NULL でなく、また行を含まない場合。空の多重集合は *nested_table2* に対して NULL 以外の値を示すサブ多重集合であるため、*nested_table2* が NULL の場合でも、TRUE が戻されます。
- *nested_table1* と *nested_table2* がともに NULL でなく、*nested_table1* に NULL 要素が含まれず、また *nested_table1* 内の各要素が *nested_table2* 内の等しい要素と 1 対 1 でマップされている場合。

次の条件のいずれかが発生する場合、NULL が戻されます。

- *nested_table1* が NULL の場合。
- *nested_table2* が NULL で、また *nested_table1* が NULL でもなく空でもない場合。
- *nested_table1* および *nested_table2* の各 NULL 要素を NULL 以外の値に変更し、*nested_table1* 内の各要素と等しい *nested_table2* 内の要素の 1 対 1 マッピングを有効にすることで、*nested_table1* が *nested_table2* のサブ多重集合となっている場合。

前述の条件のいずれも発生していない場合は、FALSE が戻されます。

***submultiset_conditions*::=**



- OF キーワードはオプションであり、演算子の動作に影響しません。
- NOT キーワードはブール出力を逆にします。*nested_table1* が *nested_table2* のサブセットである場合、FALSE が戻されます。
- ネストした表の要素の型は、比較可能な型である必要があります。スカラー型以外の型の比較の可能性は、7-4 ページの「比較条件」を参照してください。

例

次の例は、customers_demo 表から、cust_address_ntab というネストした表が cust_address2_ntab というネストした表のサブ多重集合である行を選択します。

```
SELECT customer_id, cust_address_ntab
  FROM customers_demo
  WHERE cust_address_ntab SUBMULTISET OF cust_address2_ntab
  ORDER BY customer_id;
```

この例では、表 customers_demo と、データを含むネストした表の列が 2 つ必要です。この表およびネストした表の列を作成する方法については、4-6 ページの「MULTISET 演算子」を参照してください。

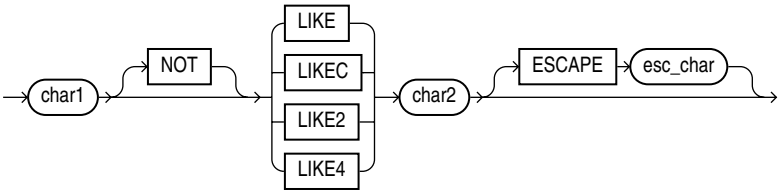
パターン一致条件

パターン一致条件は文字データを比較します。

LIKE 条件

LIKE 条件は、パターン一致を含むかどうかをテストします。等号演算子（=）は、ある文字値を別の文字値と一致させますが、LIKE 条件は、ある文字値の一部を別の文字値と一致させます（ある値が指定したパターンの検索を、もう一方の値に対して行います）。LIKE は、入力キャラクタ・セットによって定義された文字を使用して、文字列を算出します。LIKEC は、完全な Unicode キャラクタを使用します。LIKE2 は、UCS2 コードポイントを使用します。LIKE4 は、UCS4 コードポイントを使用します。

***like_condition*::=**



この構文は、次の特長があります。

- *char1* は、キャラクタ列などの文字式で、**検索値**と呼ばれます。
- *char2* は、通常はリテラルの文字式で、**パターン**と呼ばれます。
- *esc_char* は、通常はリテラルの文字式で、**エスケープ文字**と呼ばれます。

LIKE 条件は、ほぼすべての場合において最適な選択です。次のガイドラインを使用して、ご使用の環境に有効な例を判断してください。

- LIKE2 を使用して、UCS-2 セマンティクスで文字列を処理します。LIKE2 は、Unicode 補助文字を 2 文字として扱います。
- LIKE4 を使用して、UCS-4 セマンティクスで文字列を処理します。LIKE4 は、Unicode 補助文字を 1 文字として扱います。
- LIKEC を使用して、完全な Unicode キャラクタ・セマンティクスで文字列を処理します。LIKEC は、複合文字を 1 文字として扱います。

esc_char が指定されていない場合、デフォルトのエスケープ文字はありません。*char1*、*char2* または *esc_char* のいずれかが NULL である場合、結果は不明になります。それ以外の場合は、エスケープ文字（指定されている場合）は、長さが 1 の文字列です。

すべての文字式（*char1*、*char2* および *esc_char*）は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型です。文字式のデータ型が異なる場合、Oracle はすべての文字式を *char1* のデータ型に変換します。

パターンは、特殊パターン一致文字を含むことができます。

- パターン中のアンダースコア（_）は、値の中の 1 文字（マルチバイトのキャラクタ・セットでの 1 バイトとは異なる）に置き換えることができます。
- パターン中のパーセント記号（%）は、値の中の 0（ゼロ）を含む任意の数の文字（マルチバイトのキャラクタ・セットでの 1 バイトとは異なる）に置き換えることができます。ただし、パターン「%」は、NULL に置き換えることはできません。

エスケープ文字を識別する ESCAPE 句を使用すると、パターン中に % または _ を実際の文字として含めることができます。エスケープ文字がパターンの中で文字 % または _ の前に指定されている場合、Oracle は、この文字を特殊パターン一致文字としてではなく、リテラル文字として解析します。また、エスケープ文字自体を検索する場合は、その文字を続けて入力してください。たとえば、アットマーク（@）がエスケープ文字である場合、@@ を使用してアットマーク（@）を検索できます。

表 7-8 に、LIKE 条件を示します。

表 7-8 LIKE 条件

条件の種類	操作	例
x [NOT] LIKE y [ESCAPE 'z']	x がパターン y に一致する場合（NOT を指定すると一致しない場合）は TRUE と評価されます。y 内で、文字 % は 0 以上の NULL 以外の文字を含む文字列と一致します。文字 _ は、任意の 1 文字に一致します。パーセント（%）およびアンダースコア（_）を除く任意の文字を ESCAPE の後に指定できます。ワイルド・カード文字は、エスケープ文字が前に付いている場合はリテラルとして扱われます。	SELECT last_name FROM employees WHERE last_name LIKE '%A_B%' ESCAPE '\' ORDER BY last_name;

LIKE 条件を処理するために、Oracle は、パターンを 1 つまたは 2 つの文字で構成されるサブパターンに分割します。2 文字のサブパターンは、エスケープ文字で始まり、もう 1 つの文字はパーセント（%）、アンダースコア（_）またはエスケープ文字です。

P_1、P_2、...、P_n を、このようなサブパターンと想定します。1 ～ *n* のすべての *i* において、次の条件を満たすように検索値を部分文字列 S_1、S_2、...、S_n に分割できる場合、LIKE 条件は TRUE です。

- P_i がアンダースコア（_）の場合、S_i は単一文字である。
- P_i がパーセント（%）の場合、S_i は任意の文字列である。
- P_i がエスケープ文字で始まる 2 文字の場合、S_i は P_i の 2 番目の文字である。
- それ以外の場合、P_i と S_i は一致する。

LIKE 条件では、値を定数ではなくパターンと比較できます。必ず LIKE キーワードの直後に、パターンを指定してください。たとえば、次の問合せを発行することによって、名前が R で始まるすべての従業員の給与を検索できます。

```
SELECT salary
    FROM employees
    WHERE last_name LIKE 'R%'
    ORDER BY salary;
```

次の問合せは、LIKE 条件ではなく = 演算子を使用しているため、名前が「R%」のすべての従業員の給与が検索されます。

```
SELECT salary
    FROM employees
    WHERE last_name = 'R%'
    ORDER BY salary;
```

次の問合せでは、名前が「SM%」のすべての従業員の給与が検索されます。この場合、'SM%' が LIKE キーワードの前にあるため、Oracle は、'SM%' をパターンとしてではなく、テキスト・リテラルとして解析します。

```
SELECT salary
    FROM employees
    WHERE 'SM%' LIKE last_name
    ORDER BY salary;
```

英語の大 / 小文字の区別

LIKE 条件および等号（=）演算子を使用する文字式を比較するすべての条件において、大 / 小文字は区別されます。大 / 小文字の区別、アクセント記号の有無の区別をせずに LIKE 検索をするには、NLS_SORT および NLS_COMP のセッション・パラメータを設定します。

> **参照：** 大 / 小文字およびアクセント記号の有無を区別しない言語ソートの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

索引列でのパターン一致

LIKE を使用して索引列をパターン検索する場合、パターンの先頭文字が % または _ でなければ、Oracle は索引を利用して問合せのパフォーマンスを向上させることができます。この場合、Oracle はこの先頭文字によって索引をスキャンできます。パターンの先頭文字が % または _ の場合、Oracle は索引をスキャンできないため、パフォーマンスは向上しません。

LIKE 条件 : 一般的な例

次の条件は、「Ma」で始まるすべての last_name 値について TRUE（真）となります。

```
last_name LIKE 'Ma%'
```

次の last_name 値はすべて、条件を TRUE（真）にします。

```
Mallin, Markle, Marlow, Marvins, Marvis, Matos
```

大 / 小文字は区別されるため、MA、ma および mA で始まる last_name 値では条件が FALSE（偽）になります。

次の条件を考えます。

```
last_name LIKE 'SMITH_'
```

この条件は、次の last_name 値について TRUE（真）となります。

```
SMITHE, SMITHY, SMITHS
```

特殊文字であるアンダースコア（_）は、last_name 値の 1 つの文字に置き換えることができるため、この条件は「SMITH」について FALSE（偽）となります。

ESCAPE 句の例　次の例は、名前の中に文字列 A_B を持つ従業員を検索します。

```
SELECT last_name
    FROM employees
    WHERE last_name LIKE '%A\_B%' ESCAPE '\'
    ORDER BY last_name;
```

ESCAPE 句は、エスケープ文字としてバックスラッシュ（\）を識別します。パターンの中でエスケープ文字はアンダースコア（_）に先行します。これによって、Oracle は、アンダースコアを特殊なパターン一致文字としてではなく、リテラルとして解析します。

パーセント（%）なしのパターンの例　パターンに文字 % が含まれていない場合、両方のオペランドの長さが同じ場合にのみ、条件が TRUE（真）になります。次の表の定義および挿入される値について考えます。

```
CREATE TABLE ducks (f CHAR(6), v VARCHAR2(6));
INSERT INTO ducks VALUES ('DUCK', 'DUCK');
SELECT '*'||f||'*' "char",
   '*'||v||'*' "varchar"
   FROM ducks;

char     varchar
-------- --------
*DUCK  * *DUCK*
```

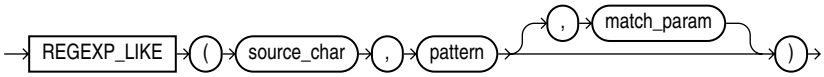
Oracle は、CHAR 値に空白埋めを行うため、f の値は空白埋めによって 6 バイトになります。v の値は空白埋めされず、4 文字長のままです。

REGEXP_LIKE 条件

REGEXP_LIKE は LIKE 条件と似ています。ただし、REGEXP_LIKE は、単純なパターン一致を実行する LIKE とは異なり、正規表現一致を実行します。この条件は、入力キャラクタ・セットによって定義された文字を使用して、文字列を評価します。

この条件は、POSIX 正規表現規格および Unicode Regular Expression Guidelines に準拠します。詳細は、付録 C「Oracle の正規表現のサポート」を参照してください。

***regexp_like_condition*::=**



- *source_char* は、検索値として使用される文字式です。これは通常は文字列であり、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。
- *pattern* は**正規表現**です。これは通常はテキスト・リテラルであり、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型です。最大 512 バイトを指定できます。*pattern* のデータ型が *source_char* のデータ型と異なる場合、Oracle は *pattern* を *source_char* のデータ型に変換します。*pattern* で指定できる演算子のリストは、付録 C「Oracle の正規表現のサポート」を参照してください。
- *match_parameter* は、ファンクションのデフォルトの検索動作を変更するためのテキスト・リテラルです。*match_parameter* には次の値を 1 つ以上指定できます。
  - 'i': 大 / 小文字を区別せずに検索します。
  - 'c': 大 / 小文字を区別して検索します。
  - 'n' は、ピリオド（.）の使用を許可します。ピリオドは、改行文字を含む任意の文字と一致するワイルド・カード文字です。このパラメータを指定しない場合、ピリオドは改行文字には一致しません。
  - 'm': ソース文字列を複数行として処理します。Oracle は、^ および $ を、それぞれ、ソース文字列全体の開始または終了としてのみではなく、ソース文字列内の任意の場所にある任意の行の開始または終了としてとして解釈します。このパラメータを指定しない場合、Oracle はソース文字列を単一行として処理します。
  - 'x' は空白文字を無視します。デフォルトでは、空白文字は空白文字として一致します。

  複数の矛盾する値を指定すると、最後の値が使用されます。たとえば、'ic' を指定すると、大 / 小文字を区別する検索が行われます。前述以外の文字を指定すると、エラーが戻されます。

  match_parameter を指定しない場合、次のようになります。
  - 大 / 小文字を区別するかどうかのデフォルトは、NLS_SORT パラメータの値によって決まります。
  - ピリオド（.）は改行文字に一致しません。
  - ソース文字列は単一行として処理されます。

  参照：
  - 「LIKE 条件」（7-14 ページ）
  - 正規表現をサポートするファンクションについては、5-144 ページの「REGEXP_INSTR」、5-147 ページの「REGEXP_REPLACE」および 5-149 ページの「REGEXP_SUBSTR」を参照してください。

例

次の問合せは、名前が Steven または Stephen である（first_name が Ste で始まって en で終わり、間が v または ph がある）従業員の姓と名を戻します。

```
SELECT first_name, last_name
FROM employees
WHERE REGEXP_LIKE (first_name, '^Ste(v|ph)en$')
ORDER BY first_name, last_name;

FIRST_NAME           LAST_NAME
-------------------- -------------------------
Steven               King
Steven               Markle
Stephen              Stiles
```

次の問合せは、姓に母音を 2 つ含む（大文字か小文字にかかわらず、last_name で a、e、i、o または u のいずれかが 2 つ連続している）従業員の姓を戻します。

```
SELECT last_name
FROM employees
WHERE REGEXP_LIKE (last_name, '([aeiou])\1', 'i')
ORDER BY last_name;

LAST_NAME
-------------------------
De Haan
Greenberg
Khoo
Gee
Greene
Lee
Bloom
Feeney
```

NULL 条件

NULL 条件は、NULL かどうかをテストします。NULL のテストに使用する必要がある唯一の条件です。

***null_conditions*::=**

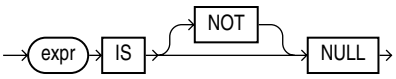


表 7-9 に、NULL 条件を示します。

表 7-9 NULL 条件

条件の種類	操作	例
IS [NOT] NULL	NULL をテストします。 **参照：**「NULL」（2-67 ページ）	SELECT last_name FROM employees WHERE commission_pct IS NULL ORDER BY last_name;

XML 条件

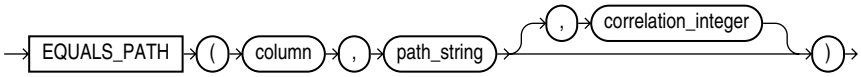
XML 条件は、指定した XML リソースが、指定されたパスにあるかどうかを判断します。

EQUALS_PATH 条件

EQUALS_PATH 条件は、Oracle XML データベース内のリソースが、指定されたパスのデータベースにあるかどうかを判断します。

この条件は、RESOURCE_VIEW および PATH_VIEW の問合せで使用します。これらのパブリック・ビューは、XML データベース・リポジトリ内に格納されているデータに対して SQL でアクセスするためのメカニズムを提供します。RESOURCE_VIEW には、リポジトリ内のリソースごとに行が 1 行あり、PATH_VIEW には、リポジトリ内の一意パスごとに行が 1 行あります。

equals_path_condition::=



この条件は、指定どおりパスにのみ適用されます。この条件は、UNDER_PATH と似ていますが、制限は多くなります。

pathname には、変換する（絶対）パス名を指定します。これには、ハード・リソース・リンクまたはウィーク・リソース・リンクである構成要素を含めることができます。

オプションの *correlation_integer* 引数は、EQUALS_PATH 条件を補助ファンクション DEPTH および PATH と関連付けます。

> **参照：** 7-19 ページの「UNDER_PATH 条件」、5-58 ページの「DEPTH」および 5-117 ページの「PATH」を参照してください。

例

ビュー RESOURCE_VIEW は、データベース・リポジトリ内の（res 列にある）すべての XML リソースへの（any_path 列にある）パスを計算します。次の例は、RESOURCE_VIEW ビューに問い合せて、サンプル・スキーマ oe にあるリソースへのパスを検索します。EQUALS_PATH 条件によって、問合せは指定されたパスのみを戻します。

```
SELECT ANY_PATH FROM RESOURCE_VIEW
   WHERE EQUALS_PATH(res, '/sys/schemas/OE/www.example.com')=1;

ANY_PATH
------------------------------------------------
/sys/schemas/OE/www.example.com
```

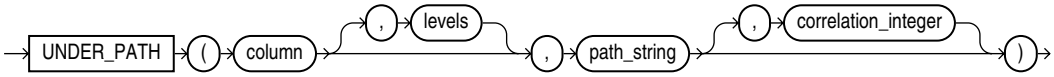
この例と、7-19 ページの「UNDER_PATH 条件」の例を比較してください。

UNDER_PATH 条件

UNDER_PATH 条件は、列で指定されたリソースが、Oracle XML データベース・リポジトリの *path_string* で指定された特定のパスにあるかどうかを判断します。パス情報は、この条件を使用する場合に問い合せる RESOURCE_VIEW ビューによって計算されます。

この条件は、RESOURCE_VIEW および PATH_VIEW の問合せで使用します。これらのパブリック・ビューは、XML データベース・リポジトリ内に格納されているデータに対して SQL でアクセスするためのメカニズムを提供します。RESOURCE_VIEW には、リポジトリ内のリソースごとに行が 1 行あり、PATH_VIEW には、リポジトリ内の一意パスごとに行が 1 行あります。

***under_path_condition*::=**



オプションの *levels* 引数は、検索対象となる *path_string* 以下のレベル数を示します。*levels* には負ではない整数を指定します。

オプションの *correlation_integer* 引数は、UNDER_PATH 条件を補助ファンクション PATH および DEPTH と関連付けます。

> **参照：** 関連する条件については、7-19 ページの「EQUALS_PATH 条件」を参照してください。補助ファンクションについては、5-58 ページの「DEPTH」および 5-117 ページの「PATH」を参照してください。

例

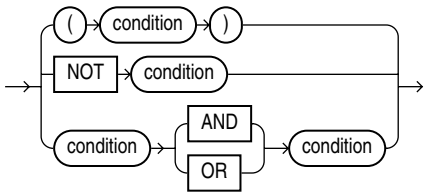
ビュー RESOURCE_VIEW は、データベース・リポジトリ内の（res 列にある）すべての XML リソースへの（any_path 列にある）パスを計算します。次の例は、RESOURCE_VIEW ビューに問い合せて、サンプル・スキーマ oe にあるリソースへのパスを検索します。この問合せは、16-64 ページの「XMLType 表の例 :」で作成された XML Schema のパスを戻します。

```
SELECT ANY_PATH FROM RESOURCE_VIEW
   WHERE UNDER_PATH(res, '/sys/schemas/OE/www.example.com')=1;

ANY_PATH
--------------------------------------------
/sys/schemas/OE/www.example.com/xwarehouses.xsd
```

複合条件

複合条件は、異なる条件の組合せを指定します。

***compound_conditions*::=**

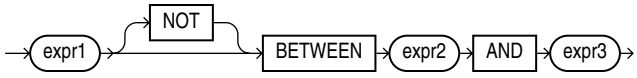


> **参照：** NOT、AND および OR 条件の詳細は、7-8 ページの「論理条件」を参照してください。

BETWEEN 条件

BETWEEN 条件は、1 つの式の値が、他の 2 つの式によって定義された期間内に存在するかどうかを決定します。

***between_condition*::=**



3 つのすべての式は、数式、文字式または日時式である必要があります。SQL では、*expr1* を複数回評価できます。BETWEEN 式が PL/SQL で使用されている場合、*expr1* は 1 回のみ評価されることが保証されます。すべての式が同じデータ型ではない場合、式は共通のデータ型に暗黙的に変換されます。それが不可能な場合は、エラーが戻されます。

注意： PL/SQL の暗黙的なデータ型変換規則は SQL の規則とは異なります。詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

参照： SQL のデータ型変換の詳細は、2-39 ページの「暗黙的なデータ変換」を参照してください。

次の式の値について考えてみます。

```
expr1 NOT BETWEEN expr2 AND expr3
```

この式の値は次の式の値と同じです。

```
NOT (expr1 BETWEEN expr2 AND expr3)
```

さらに次の式の値について考えてみます。

```
expr1 BETWEEN expr2 AND expr3
```

この式の値は次のブール式の値と同じです。

```
expr2 <= expr1 AND expr1 <= expr3
```

expr3 < *expr2* の場合、この期間は空です。*expr1* が NULL の場合、結果は NULL となります。*expr1* が NULL ではない場合、値は通常は FALSE であり、キーワード NOT が使用された場合は TRUE となります。

ブール演算子 AND を使用すると、予期しない結果となる場合があります。特に、式 `x AND y` では、条件 `x IS NULL` は式の値を決定するのに十分ではありません。2 番目のオペランドも評価する必要があります。2 番目のオペランドの値が FALSE の場合、結果は FALSE であり、そうではない場合は NULL となります。AND の詳細は、7-8 ページの「論理条件」を参照してください。

表 7-10 BETWEEN 条件

条件の種類	操作	例
[NOT] BETWEEN x AND y	[NOT]（*expr2* が *expr1* 以下かつ *expr1* が *expr3* 以下）	SELECT * FROM employees WHERE salary BETWEEN 2000 AND 3000 ORDER BY employee_id;

EXISTS 条件

EXISTS 条件は、副問合せに行が存在するかどうかをテストします。

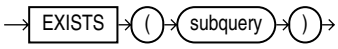


表 7-11 に、EXISTS 条件を示します。

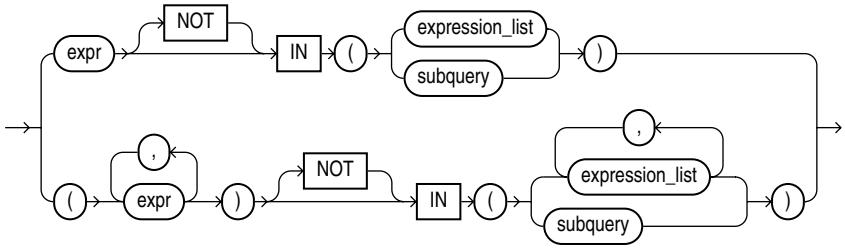
表 7-11 EXISTS 条件

条件の種類	操作	例
EXISTS	副問合せによって行が 1 行以上戻される場合には、TRUE と評価されます。	SELECT department_id FROM departments d WHERE EXISTS (SELECT * FROM employees e WHERE d.department_id = e.department_id) ORDER BY department_id;

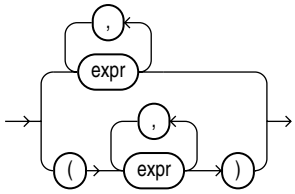
IN 条件

in_condition はメンバーシップ条件です。メンバーシップ条件は、値のリストまたは副問合せ内のメンバーシップをテストします。

***in_conditions*::=**



***expression_list*::=**



in_condition 条件の上の方の書式（演算子の左辺に 1 つの式を指定する書式）を使用する場合は、*expression_list* の上の方の書式を使用する必要があります。この条件の下の方の書式（演算子の左辺に複数の式を指定する書式）を使用する場合は、*expression_list* の下の方の書式を使用し、各 *expression_list* 内の式は、演算子の左辺にある式と同じ数とデータ型で構成されている必要があります。

参照： 「式のリスト」（6-15 ページ）

表 7-12 に、IN 条件の書式を示します。

表 7-12 IN 条件

条件の種類	操作	例
IN	メンバーとの等価性をテストします。=ANY と同じです。	SELECT * FROM employees WHERE job_id IN ('PU_CLERK','SH_CLERK') ORDER BY employee_id; SELECT * FROM employees WHERE salary IN (SELECT salary FROM employees WHERE department_id =30) ORDER BY employee_id;
NOT IN	!=ALL と同じです。メンバーのいずれかが NULL の場合には、FALSE と評価されます。	SELECT * FROM employees WHERE salary NOT IN (SELECT salary FROM employees WHERE department_id = 30) ORDER BY employee_id; SELECT * FROM employees WHERE job_id NOT IN ('PU_CLERK', 'SH_CLERK') ORDER BY employee_id;

NOT IN 演算子に続くリストの中のいずれかの項目が NULL の場合は、すべての行は FALSE または不明（UNKNOWN）と評価されます（行は戻されません）。たとえば、次の文ではそれぞれの行に対して文字列 'True' が戻されます。

```
SELECT 'True' FROM employees
   WHERE department_id NOT IN (10, 20);
```

ただし、次の文では行は戻されません。

```
SELECT 'True' FROM employees
    WHERE department_id NOT IN (10, 20, NULL);
```

この例で行が戻されないのは、WHERE 句の条件が次のように評価されるためです。

```
department_id != 10 AND department_id != 20 AND department_id != null
```

3 番目の条件で department_id と NULL の比較が行われるため、この条件の結果が UNKNOWN となり、式全体の結果が FALSE（department_id を持つ行が 10 または 20 と等しいため）となります。特に、NOT IN 演算子が副問合せを参照するときは、このような動作を見逃してしまう可能性があることに注意してください。

また、NOT IN 条件が、行を戻さない副問合せを参照する場合は、次のように、すべての行が戻されます。

```
SELECT 'True' FROM employees
   WHERE department_id NOT IN (SELECT 0 FROM DUAL WHERE 1=2);
```

WHERE 句の LEVEL の制限事項： WHERE 句の [NOT] IN 条件の右辺が副問合せになっている場合、条件の左辺で LEVEL は使用できません。ただし、FROM 句の副問合せで LEVEL を指定すると、同じ結果が得られます。たとえば、次の文は無効です。

```
SELECT employee_id, last_name FROM employees
   WHERE (employee_id, LEVEL)
      IN (SELECT employee_id, 2 FROM employees)
   START WITH employee_id = 2
   CONNECT BY PRIOR employee_id = manager_id;
```

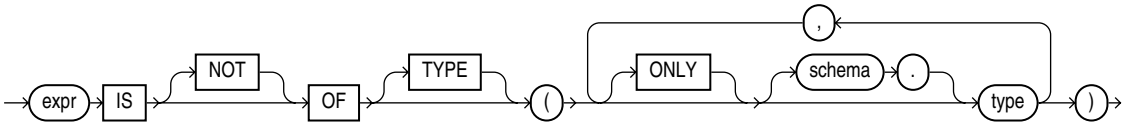
ただし、次の文は、LEVEL 情報を含む問合せを FROM 句にカプセル化するため有効です。

```
SELECT v.employee_id, v.last_name, v.lev
   FROM
      (SELECT employee_id, last_name, LEVEL lev
      FROM employees v
      START WITH employee_id = 100
      CONNECT BY PRIOR employee_id = manager_id) v
   WHERE (v.employee_id, v.lev) IN
      (SELECT employee_id, 2 FROM employees);
```

IS OF *type* 条件

IS OF *type* 条件を使用すると、固有の型情報に基づくオブジェクト・インスタンスをテストできます。

***is_of_type_conditions*::=**



type によって参照されるすべての型に対する EXECUTE 権限が必要です。また、すべての *type* は、同じ型ファミリに属している必要があります。

expr が NULL である場合、この条件は NULL と評価されます。*expr* が NULL でない場合、次に示す環境では、この条件は TRUE（NOT キーワードを指定した場合は FALSE）と評価されます。

- *expr* の型が、*type* リストで指定された型のサブタイプであり、ONLY を指定していない場合
- *expr* の型が *type* リストで明示的に指定されている場合

expr は、相関変数を伴う VALUE ファンクションの書式をとる場合があります。

次の例では、16-61 ページの「置換可能な表および列のサンプル :」の型階層に基づいて作成されたサンプル表 oe.persons を使用します。この例は、IS OF *type* 条件を使用して、問合せを特定のサブタイプに制限します。

```
SELECT * FROM persons p
   WHERE VALUE(p) IS OF TYPE (employee_t);

NAME                          SSN
----------------------------
Joe                         32456
Tim                          5678

SELECT * FROM persons p
   WHERE VALUE(p) IS OF (ONLY part_time_emp_t);

NAME                          SSN
----------------------------
Tim                         5678
```

8

共通の SQL DDL 句

この章では、複数の SQL 文で使用される SQL データ定義句について説明します。

この章では、次の内容を説明します。

- allocate_extent_clause
- constraint
- deallocate_unused_clause
- file_specification
- logging_clause
- parallel_clause
- physical_attributes_clause
- size_clause
- storage_clause

allocate_extent_clause

用途

allocate_extent_clause 句を使用すると、データベース・オブジェクトの新しいエクステントを明示的に割り当てることができます。

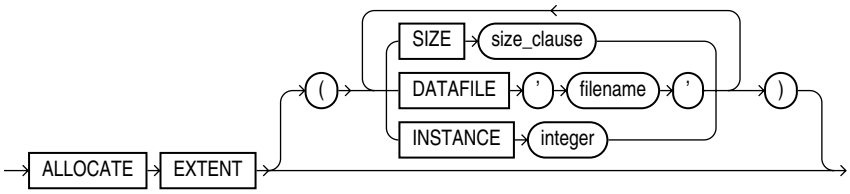
この句を使用してエクステントを明示的に割り当てた場合、NEXT および PCTINCREASE 記憶域パラメータの値は変更されません。したがって、Oracle Database で暗黙的に割り当てられる次のエクステントのサイズには影響ありません。NEXT および PCTINCREASE 記憶域パラメータの詳細は、8-43 ページの「storage_clause」を参照してください。

エクステントは次の SQL 文で割り当てることができます。

- ALTER CLUSTER（10-5 ページの「ALTER CLUSTER」を参照）
- ALTER INDEX: エクステントを索引、索引パーティションまたは索引サブパーティションに割り当てる場合（10-64 ページの「ALTER INDEX」を参照）
- ALTER MATERIALIZED VIEW: エクステントをマテリアライズド・ビュー、そのパーティションまたはサブパーティションの 1 つ、あるいは索引構成マテリアライズド・ビューのオーバーフロー・セグメントに割り当てる場合（11-2 ページの「ALTER MATERIALIZED VIEW」を参照）
- ALTER MATERIALIZED VIEW LOG（11-16 ページの「ALTER MATERIALIZED VIEW LOG」を参照）
- ALTER TABLE: エクステントを表、表パーティション、表サブパーティション、索引構成表のマッピング表、索引構成表のオーバーフロー・セグメントまたは LOB 記憶域セグメントに割り当てる場合（12-2 ページの「ALTER TABLE」を参照）

構文

***allocate_extent_clause*::=**



（8-42 ページの size_clause::= を参照）

セマンティクス

この項では、*allocate_extent_clause* のパラメータについて説明します。詳細は、特定のデータベース・オブジェクトに対してこれらのパラメータを設定または再設定する SQL 文の説明を参照してください。

同じ文で *allocate_extent_clause* と *deallocate_unused_clause* を指定することはできません。

SIZE

エクステント・サイズをバイト単位で指定します。*integer* の値は 0 ～ 2147483647 となります。これより大きいエクステント・サイズを指定するには、この範囲の整数と K、M、G または T を使用して、エクステント・サイズを KB、MB、GB または TB 単位で指定します。

表、索引、マテリアライズド・ビューまたはマテリアライズド・ビュー・ログでは、SIZE を指定しないと、Oracle Database ではオブジェクトの記憶域パラメータの値に基づいてサイズが決定されます。ただし、クラスタの場合、クラスタの記憶域パラメータは評価されないため、デフォルト値を使用しない場合は SIZE を指定する必要があります。

DATAFILE '*filename*'

新しいエクステントを割り当てるデータ・ファイルを、表、クラスタ、索引、マテリアライズド・ビューまたはマテリアライズド・ビュー・ログの表領域から 1 つ指定します。DATAFILE を指定しないと、Oracle がデータ・ファイルを選択します。

INSTANCE *integer*

Oracle Real Application Clusters を使用する場合のみ、このパラメータを使用します。

INSTANCE *integer* を指定すると、指定したインスタンスに対応付けられた空きリスト・グループが新しいエクステントを使用できるようになります。インスタンス数が空きリスト・グループの最大数を超えた場合、指定したインスタンス数 / 最大数という除算が行われ、その余りから、使用する空きリスト・グループが識別されます。インスタンスは初期化パラメータ INSTANCE_NUMBER の値で識別されます。

このパラメータを指定しない場合、表、クラスタ、索引、マテリアライズド・ビューまたはマテリアライズド・ビュー・ログに領域が割り当てられますが、特定の空きリスト・グループからは割り当てられません。かわりにマスター空きリストが使用され、必要に応じて領域が割り当てられます。

注意： 自動セグメント領域管理の使用時は、*allocate_extent_clause* の INSTANCE パラメータを設定しても、指定したインスタンスに新しく割り当てられた領域が確保されない場合があります。これは、自動セグメント領域管理では、エクステントとインスタンス間の固定したアフィニティが保持されないためです。

---

constraint

用途

constraint を使用すると、**整合性制約**（データベース内の値を制限する規則）を定義できます。Oracle Database では、6 つの制約を作成し、それを 2 つの方法で宣言することができます。

次に、6 つの整合性制約について簡単に説明します。詳細は、8-8 ページの「セマンティクス」を参照してください。

- **NOT NULL 制約**は、データベースの値が NULL になることを禁止します。
- **一意制約**は、複数の行が同じ列または列の組合せで同じ値を持つことを禁止しますが、一部の値が NULL になることを許可します。
- **主キー制約**は、1 つの宣言で NOT NULL 制約と一意制約を組み合せたものです。これにより、同じ列または列の組合せで、複数の行の値が同じにならないように、また値が NULL にならないようにします。
- **外部キー制約**は、ある表の値が別の表の値と一致することを必要とします。
- **CHECK 制約**は、データベースの値が、指定された条件を満たすことを必要とします。
- REF 列は、定義上は別のオブジェクト型またはリレーショナル表の中のオブジェクトを参照します。**REF 制約**を使用すると、REF 列と参照先のオブジェクトの関係をさらに詳しく指定できます。

制約は、次の 2 つの構文で定義できます。

- 個々の列または属性の定義の一部として定義できます。これを、**表内**指定といいます。
- 表定義の一部として定義できます。これを、**表外**指定といいます。

NOT NULL 制約は、表内に宣言する必要があります。その他のすべての制約は、表内または表外に宣言できます。

制約句は、次の文に指定できます。

- CREATE TABLE（16-6 ページの「CREATE TABLE」を参照）
- ALTER TABLE（12-2 ページの「ALTER TABLE」を参照）
- CREATE VIEW（17-13 ページの「CREATE VIEW」を参照）
- ALTER VIEW（13-12 ページの「ALTER VIEW」を参照）

ビュー制約　Oracle Database では、ビュー制約を適用していません。ただし、実表に対する制約によってビューに制約を適用できます。

ビューには、一意制約、主キー制約および外部キー制約のみを指定でき、これらの制約は DISABL NOVALIDATE モードのみでサポートされています。オブジェクト列の属性にビュー制約を定義することはできません。

> **参照：**　ビュー制約の詳細は、8-17 ページの「ビュー制約」を参照してください。DISABLE NOVALIDATE モードの詳細は、8-15 ページの「DISABLE 句」を参照してください。

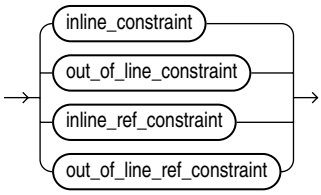
前提条件

制約を定義する文を発行できる権限が必要です。

外部キー制約を作成する場合は、この条件に加えて、親表またはビューが自分のスキーマ内に設定されている必要があります。設定されていないかぎり、親表またはビューの参照キー列に対する REFERENCES 権限が必要です。

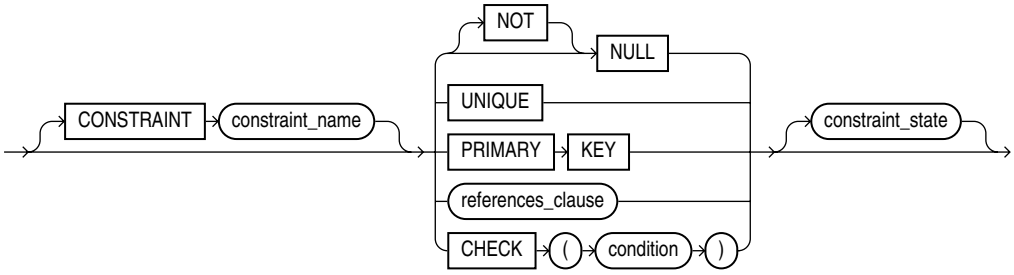
構文

***constraint*::=**



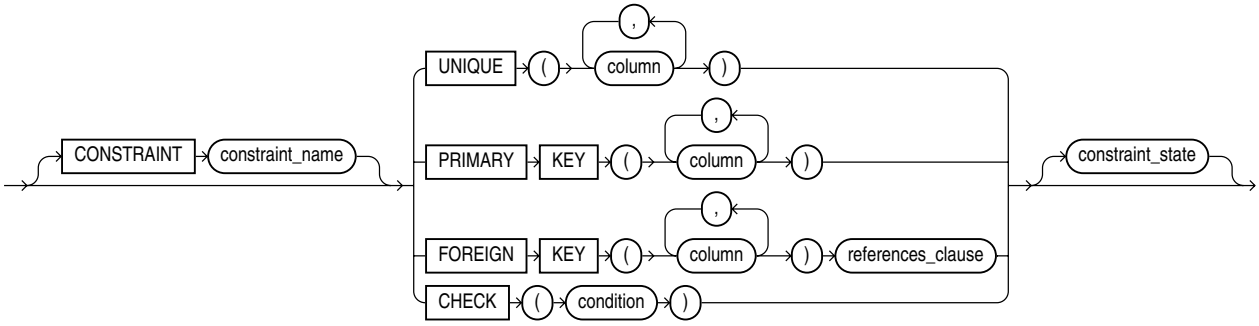
（8-5 ページの inline_constraint::=、8-5 ページの out_of_line_constraint::=、8-5 ページの inline_ref_constraint::=、8-6 ページの out_of_line_ref_constraint::= を参照）

***inline_constraint*::=**



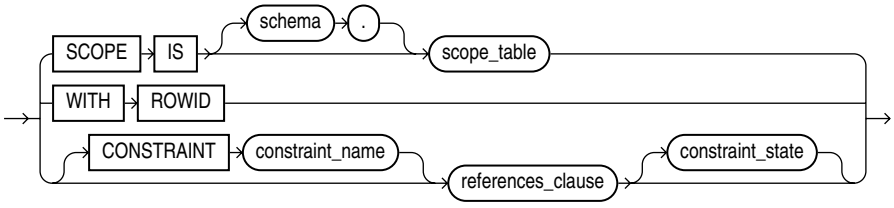
（8-6 ページの references_clause::= を参照）

***out_of_line_constraint*::=**



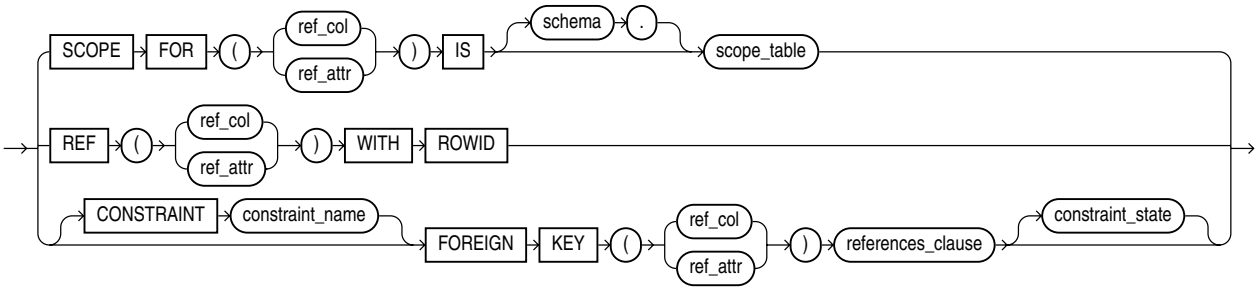
（8-6 ページの references_clause::=、8-6 ページの constraint_state::= を参照）

***inline_ref_constraint*::=**



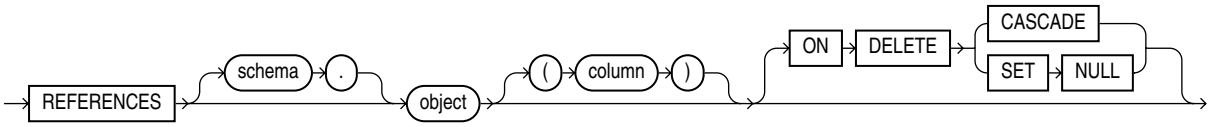
（8-6 ページの references_clause::=、8-6 ページの constraint_state::= を参照）

***out_of_line_ref_constraint*::=**

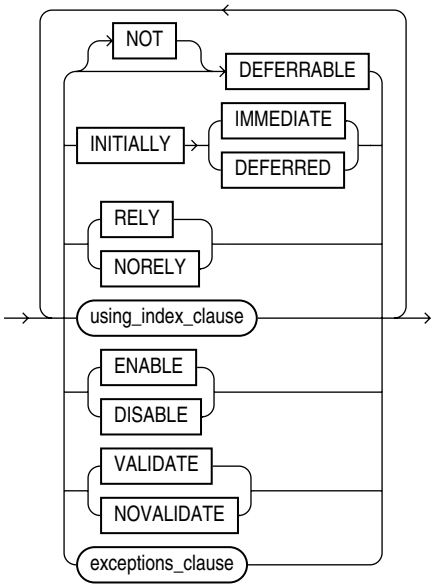


(8-6 ページの references_clause::=、8-6 ページの constraint_state::= を参照)

references_clause::=

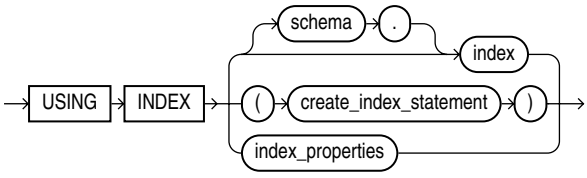


constraint_state::=



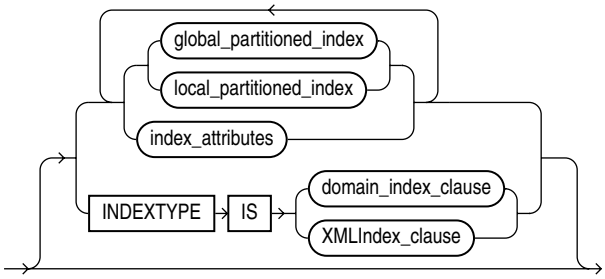
(8-6 ページの using_index_clause::=、8-7 ページの exceptions_clause::= を参照)

using_index_clause::=



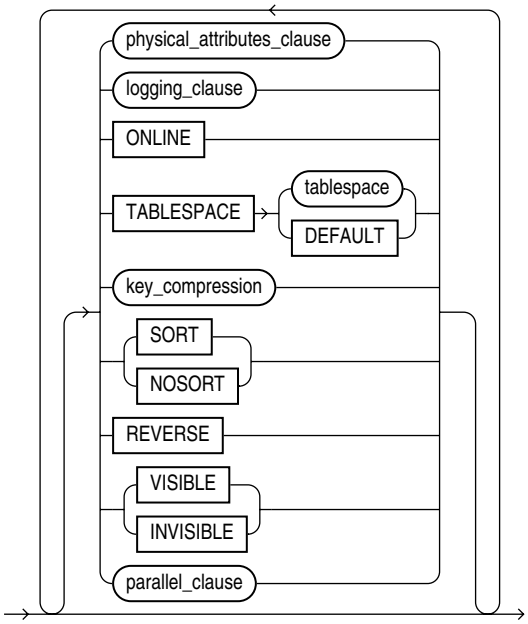
(14-51 ページの create_index::=、8-7 ページの index_properties::= を参照)

***index_properties*::=**



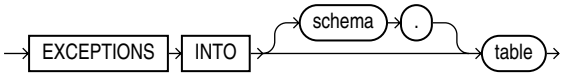
（14-53 ページの global_partitioned_index::=、「CREATE INDEX」の項にある 14-55 ページの local_partitioned_index::=、8-7 ページの index_attributes::= を参照。INDEXTYPE IS ... 句は、制約を定義すると無効になります。）

***index_attributes*::=**



（14-3 ページの physical_attributes_clause::=、8-34 ページの logging_clause::=、「CREATE INDEX」の項にある 14-53 ページの key_compression::= を参照。*parallel_clause* は、*using_index_clause* でサポートされていません。）

exceptions_clause::=



セマンティクス

この項では、*constraint* のセマンティクスについて説明します。詳細は、表またはビューの制約を定義または再定義する SQL 文の説明を参照してください。

Oracle Database では、ユーザー定義オブジェクト、ネストした表、VARRAY、REF または LOB 型の列または属性に対して制約を使用することはできません。ただし、次の 2 つの例外があります。

- NOT NULL 制約は、ユーザー定義オブジェクト、VARRAY、REF または LOB 型の列や属性に使用できます。
- NOT NULL、外部キーおよび REF 制約は、REF 型の列に使用できます。

CONSTRAINT *constraint_name* 制約名を指定します。この識別子を指定しない場合、SYS_C*n* の形式で名前が生成されます。整合性制約の名前と定義は、USER_、ALL_ および DBA_CONSTRAINTS データ・ディクショナリ・ビュー（それぞれ CONSTRAINT_NAME 列および SEARCH_CONDITION 列）に格納されます。

> **参照：** データ・ディクショナリ・ビューの詳細は、『Oracle Database リファレンス』を参照してください。

NOT NULL 制約

NOT NULL 制約は、列に NULL が含まれることを禁止します。NULL キーワード自体は、実際に整合性制約を定義するものではありませんが、これを指定すると、列に NULL が含まれることが許可されます。NOT NULL および NULL は、表内指定で定義する必要があります。NOT NULL または NULL を指定しない場合、NULL がデフォルトになります。

NOT NULL 制約は、XMLType 列および VARRAY 列で表内指定できる唯一の制約です。

NOT NULL 制約を満たすには、表のすべての行がその列の値を持つ必要があります。

> **注意：** Oracle Database では、すべてのキー列が NULL の表の行には索引を付けません（ただし、ビットマップ索引の場合を除きます）。このため、表のすべての行に索引を付けるには、1 つ以上の索引キー列に NOT NULL 制約を指定するか、ビットマップ索引を作成する必要があります。

NOT NULL 制約の制限事項： NOT NULL 制約には次の制限事項があります。

- ビュー制約には NULL または NOT NULL を指定できません。
- オブジェクト属性には NULL または NOT NULL を指定できません。そのかわりに、IS [NOT] NULL 条件で CHECK 制約を使用してください。

> **参照：** 8-22 ページの「属性レベル制約の例」および 8-19 ページの「NOT NULL の例」を参照してください。

一意制約

一意制約は、列を一意キーとして指定します。**複合一意キー**は、列の組合せを一意キーとして指定します。一意制約を表内に定義する場合に必要となるのは、UNIQUE キーワードのみです。一意制約を表外に定義する場合は、1 つ以上の列も指定する必要があります。複合一意キーは、表外に定義する必要があります。

一意制約を満たすには、表の中の 2 つの行が一意キーに対して同じ値を持たないようにする必要があります。ただし、単一の列で構成される一意キーの場合は、複数の NULL を持つことができます。複合一意キーを満たすには、表またはビューの 2 つの行がキー列に対して同じ組合せの値を持たないようにする必要があります。すべてのキー列に対して NULL を持つ行は、自動的にその制約を満たすことになります。ただし、1 つ以上のキー列に対して NULL を持ち、その他のキー列に対して同じ組合せの値を持つ 2 つの行は、制約に違反します。

Oracle では、1 つ以上の列に一意制約を指定すると、暗黙的に一意キーに索引が作成されます。問合せのパフォーマンスを目的に一意性を定義する場合は、かわりに CREATE UNIQUE INDEX 文を使用して明示的に一意索引を作成することをお薦めします。また、CREATE UNIQUE INDEX 文で、条件付きの一意制約を定義する一意のファンクション索引を作成することもできます。詳細は、14-70 ページの「ファンクション索引の使用による条件付き一意性の定義例 :」を参照してください。

一意制約の制限事項 : 一意制約には、次の制限事項があります。

- 一意キー列は、LOB、LONG、LONG RAW、VARRAY、NESTED TABLE、OBJECT、REF、TIMESTAMPWITH TIME ZONE またはユーザー定義型を含むことができません。ただし、一意キーは TIMESTAMP WITH LOCAL TIME ZONE の列を含むことができます。
- 1 つの複合一意キーは、33 以上の列を持つことはできません。
- 同一の列または列の組合せを一意キーと主キーの両方には指定できません。
- 継承階層でサブビューを作成する場合は、一意キーを指定できません。一意キーは、トップレベル（ルート）のビューのみに指定できます。

> **参照 :** 8-17 ページの「一意キーの例」および 8-18 ページの「複合一意キーの例」を参照してください。

主キー制約

主キー制約は、列を表またはビューの主キーとして指定します。**複合主キー**は、列の組合せを主キーとして指定します。主キー制約を表内に定義する場合に必要となるのは、PRIMARY KEY キーワードのみです。主キー制約を表外に定義する場合は、1 つ以上の列も指定する必要があります。複合主キーは表外に定義する必要があります。

主キー制約は、1 つの宣言で NOT NULL 制約と一意制約を組み合せたものです。このため、主キー制約を満たすには次の条件があります。

- 主キーの値が、表の中の複数行に存在することはできません。
- 主キーを構成する列に、NULL を持たせることはできません。

主キー制約の制限事項 : 主キー制約には、次の制限事項があります。

- 表またはビューには、主キーを 1 つのみ指定できます。
- 主キー列は、LOB、LONG、LONG RAW、VARRAY、NESTED TABLE、BFILE、REF、TIMESTAMP WITH TIME ZONE またはユーザー定義型を含むことができません。ただし、主キーは TIMESTAMP WITH LOCAL TIME ZONE の列を含むことができます。
- 主キーのサイズは、1 データベース・ブロックを超えることはできません。
- 1 つの複合主キーは、33 以上の列を持つことはできません。
- 同一の列または列の組合せを一意キーと主キーの両方には指定できません。
- 継承階層でサブビューを作成する場合は、主キーを指定できません。主キーは、トップレベル（ルート）のビューのみに指定できます。

> **参照 :** 8-18 ページの「主キーの例」および 8-19 ページの「複合主キーの例」を参照してください。

外部キー制約

外部キー制約（**参照整合性制約**ともいう）は、列を外部キーとして指定し、その外部キーと指定した主キーまたは一意キー（**参照キー**）との関係を設定します。**複合外部キー**は、列の組合せを外部キーとして指定します。

外部キーを持つ表またはビューを**子**オブジェクトといい、参照キーを持つ表またはビューを**親**オブジェクトといいます。外部キーと参照キーを、同一の表またはビューに設定することができます。この場合、親表と子表は同一の表になります。親表または親ビューのみを指定して、列名を指定しない場合、外部キーは自動的に親表または親ビューの主キーを参照します。外部キーと参照キーの対応する列は、同じ順序とデータ型で構成されている必要があります。

単一キー列の外部キー制約は、表内または表外に定義できます。複合外部キーと属性の外部キーは、表外に指定します。

複合外部キー制約を満たすには、複合外部キーが親表または親ビューの複合一意キーまたは複合主キーを参照するか、外部キーの 1 つ以上の列の値が NULL である必要があります。

外部キーと主キーの両方、または外部キーと一意キーの両方に、同一の列または列の組合せを指定できます。また、外部キーとクラスタ・キーの両方にも同じ列または列の組合せを指定できます。

1 つの表またはビューで複数の外部キーを定義できます。また、1 つの列が複数の外部キーを構成することもできます。

外部キー制約の制限事項： 外部キー制約には、次の制限事項があります。

- 外部キー列は、LOB、LONG、LONG RAW、VARRAY、NESTED TABLE、BFILE、REF、TIMESTAMP WITH TIME ZONE またはユーザー定義型を含むことができません。ただし、主キーは TIMESTAMP WITH LOCAL TIME ZONE の列を含むことができます。
- 親表または親ビューの参照一意制約または参照主キー制約が事前に定義されている必要があります。
- 1 つの複合外部キーは、33 以上の列を持つことはできません。
- 子表と親表は、同一データベース上に設定されている必要があります。分散データベースのノード間の参照整合性制約を使用可能にする場合、データベース・トリガーを使用する必要があります。16-85 ページの「CREATE TRIGGER」を参照してください。
- 子オブジェクトと親オブジェクトのどちらかがビューの場合、ビュー制約のすべての制限事項が制約に適用されます。8-17 ページの「ビュー制約」を参照してください。
- AS *subquery* 句を含む CREATE TABLE 文には、外部キー制約を定義できません。そのかわりに、制約を指定せずに表を作成し、後で ALTER TABLE 文を使用してその制約を追加できます。

  参照：

  - 制約の使用方法の詳細は、『Oracle Database アドバンスト・アプリケーション開発者ガイド』を参照してください。
  - 8-19 ページの「外部キー制約の例」および 8-20 ページの「複合外部キー制約の例」を参照してください。

references_clause 外部キー制約は *references_clause* 構文を使用します。外部キー制約を表内に指定する場合に必要となるのは、*references_clause* のみです。外部キー制約を表外に指定する場合は、FOREIGN KEY キーワードと 1 つ以上の列も指定する必要があります。

ON DELETE 句 ON DELETE 句を指定すると、参照主キーまたは参照一意キーの値を削除した場合に参照整合性がどのように自動処理されるかを指定できます。この句を指定しない場合、子表に依存する行を持つ親表の中の参照キーの値は削除できません。

- 依存する外部キーの値を削除する場合は、CASCADE を指定します。
- 依存する外部キーの値を NULL に変換する場合は、SET NULL を指定します。仮想列の値は直接更新できないため、仮想列に対してこの句を指定することはできません。仮想列が導出される値を更新する必要があります。

ON DELETE の制限事項： この句は、ビュー制約に対して指定できません。

参照：「ON DELETE の例」（8-20 ページ）

CHECK 制約

CHECK 制約によって、表の各行に必要な条件を指定できます。制約を満たすためには、表のそれぞれの行が、その条件に対して TRUE または不明（NULL のため）のいずれかである必要があります。特定の行に対する CHECK 制約条件が評価される場合、条件にある列名に、その行の列値が適用されます。

CHECK 制約の表内指定と表外指定の構文は同じです。ただし、表内指定では現在定義されている列（または、オブジェクト列の場合は列の属性）のみを参照でき、表外指定では複数の列または属性を参照できます。

Oracle では、CHECK 制約の条件が相互に排他的かどうかは検証しません。このため、1 つの列に対して複数の CHECK 制約を作成する場合は、制約の用途が矛盾しないように注意する必要があります。また、条件の評価について特別な順序を想定しないでください。

参照：

- その他の詳細および構文は、第 7 章「条件」を参照してください。
- 8-21 ページの「CHECK 制約の例」および 8-22 ページの「属性レベル制約の例」を参照してください。

CHECK 制約の制限事項： CHECK 制約には、次の制限事項があります。

- ビューに対しては、CHECK 制約を指定できません。ただし、WITH CHECK OPTION 句を使用してビューを定義することは可能です。これは、ビューに CHECK 制約を指定することと同じです。
- CHECK 制約の条件では、その表の中のすべての列を参照できますが、他の表の列は参照できません。
- CHECK 制約の条件は、次の構造を持つことができません。
  - 副問合せおよびスカラー副問合せ式
  - 決定的でないファンクションへのコール（CURRENT_DATE、CURRENT_TIMESTAMP、DBTIMEZONE、LOCALTIMESTAMP、SESSIONTIMEZONE、SYSDATE、SYSTIMESTAMP、UID、USER および USERENV）
  - ユーザー定義ファンクションへのコール
  - REF 列の参照解除（DEREF ファンクションを使用する場合など）
  - ネストした表の列または属性
  - 疑似列 CURRVAL、NEXTVAL、LEVEL または ROWNUM
  - 完全に指定されていない日付定数

REF 制約

REF 制約を使用すると、REF 型の列とそれが参照するオブジェクトとの関係を指定できます。

ref_constraint REF 制約は、*ref_constraint* 構文を使用します。REF 制約は表内または表外に定義します。表外指定の場合は、指定している REF 列または属性を指定する必要があります。

- *ref_column* には、オブジェクトまたはリレーショナル表の REF 列の名前を指定します。
- *ref_attribute* には、リレーショナル表のオブジェクト列内の埋込み REF 属性を指定します。

表内指定と表外指定のどちらでも、有効範囲制約、ROWID 制約または参照整合性制約を REF 列に定義できます。

REF 列の有効範囲表または参照表が、主キーに基づくオブジェクト識別子を持っている場合、その REF 列は**ユーザー定義 REF 列**です。

参照：

- REF データ型の詳細は、『Oracle Database オブジェクト・リレーショナル開発者ガイド』を参照してください。
- 8-9 ページの「外部キー制約」および 8-22 ページの「REF 制約の例」を参照してください。

REF 列の有効範囲制約

表が REF 列を持つ場合は、この列のそれぞれの REF 値が別のオブジェクト表内の行を参照する場合があります。SCOPE 句は、参照の有効範囲を 1 つの表 *scope_table* に制限します。REF 列または属性の値は *scope_table* 内のオブジェクトを指し、その場所に REF 列と同じ型のオブジェクト・インスタンスが格納されます。

REF 列内の参照の有効範囲を 1 つの表に制限する場合に、SCOPE 句を指定します。この句を指定するには、*scope_table* が自分のスキーマ内にあるか、または *scope_table* に対する SELECT 権限または SELECT ANY TABLE システム権限が必要です。REF 列ごとに有効範囲表を 1 つのみ指定できます。

有効範囲制約の制限事項： 有効範囲制約には、次の制限事項があります。

- 表が空でない場合は、有効範囲制約を既存の列に追加できません。
- VARRAY 列の REF 要素に対しては有効範囲制約を指定できません。
- AS *subquery* を指定し、この副問合せがユーザー定義 REF データ型を戻す場合、この句を指定する必要があります。
- 有効範囲制約を、後で REF 列から削除することはできません。

REF 列の ROWID 制約

WITH ROWID を指定して、*ref_column* または *ref_attribute* の REF の値とともに ROWID を格納します。ROWID とともに REF 値を格納した場合、参照解除操作のパフォーマンスは向上しますが、使用する領域も多くなります。デフォルトの REF 値の記憶域では、ROWID は格納されません。

参照： 参照解除の例は、5-58 ページの「DEREF」を参照してください。

ROWID 制約の制限事項： ROWID 制約には、次の制限事項があります。

- VARRAY 列の REF 要素に対しては ROWID 制約を定義できません。
- ROWID 制約を、後で REF 列から削除することはできません。
- REF 列または属性の範囲が限定される場合、この句は無視され、ROWID は REF とともには格納されません。

REF 列の参照整合性制約

ref_constraint 構文の *references_clause* を使用すると、外部キー制約を REF 列に定義できます。また、この句は REF 列や属性の有効範囲を参照表に暗黙的に制限します。ただし、REF 列以外の外部キー制約が親表の実際の列を参照することに対し、REF 列の外部キー制約は親表の暗黙のオブジェクト識別子を参照します。

制約名を指定しない場合、制約に対するシステム名が SYS_Cn という形式で生成されます。

範囲が限定されている既存の REF 列に参照整合性制約を追加する場合、参照表は、REF 列の有効範囲表と同じである必要があります。後で参照整合性制約を削除する場合、REF 列の範囲が参照表に限定されたままになります。

他の型の列に対する外部キー制約と同様に、表内で宣言する場合に必要となるのは *references_clause* のみです。表外で宣言する場合は、FOREIGN KEY キーワード、および 1 つ以上の REF 列または属性も指定する必要があります。

> **参照：** オブジェクト識別子の詳細は、『Oracle Database オブジェクト・リレーショナル開発者ガイド』を参照してください。

REF 列の参照整合性制約の制限事項： REF 列における外部キー制約には、次の追加の制限事項があります。

- 範囲が限定されていない既存の REF 列に参照整合性制約を追加する場合、有効範囲制約が暗黙的に追加されます。したがって、有効範囲制約に適用されるすべての制限事項は、この場合にも適用されます。
- *references_clause* 内のオブジェクト名の後には列を指定できません。

制約状態の指定

制約定義の一部として、制約が適用される方法およびタイミングを指定できます。

constraint_state *constraint_state* は、表内指定と表外指定の両方に使用できます。*constraint_state* の句を表示される順序（上から下）で指定します。いずれの句も複数回は指定できません。

DEFERRABLE 句 DEFERRABLE および NOT DEFERRABLE パラメータは、後続のトランザクションで、SET CONSTRAINT(S) 文を使用して、トランザクションが終わるまで制約のチェックを遅延できるかどうかを指定します。この句を指定しない場合、NOT DEFERRABLE がデフォルトになります。

- NOT DEFERRABLE を指定すると、後続のトランザクションで、SET CONSTRAINT[S] 句を使用して、トランザクションがコミットされるまでこの制約のチェックを遅延させることができません。NOT DEFERRABLE 制約のチェックは、トランザクションの終わりまで遅延させることはできません。

  新しい NOT DEFERRABLE 制約を宣言する場合、CREATE TABLE または ALTER TABLE 文のコミット時にその制約が有効である必要があります。有効でない場合、これらの文は正常に実行されません。

- DEFERRABLE を指定すると、後続のトランザクションで、SET CONSTRAINT[S] 句を使用して、トランザクションがコミットされるまでこの制約のチェックを遅延させることができます。この設定によって、制約に違反する可能性のある変更をデータベースに行っている場合に、すべての変更が完了するまで、制約を一時的に使用禁止にすることが可能になります。

制約の遅延可能状態は変更できません。これらのパラメータのいずれを指定するか、どちらも指定せずに NOT DEFERRABLE 制約を暗黙的に有効にするかに関係なく、この句を ALTER TABLE 文に指定することはできません。制約を削除してから再作成する必要があります。

> **参照：**
>
> - トランザクションに対する制約のチェックの設定の詳細は、19-49 ページの「SET CONSTRAINT[S]」を参照してください。
> - 遅延制約の詳細は、『Oracle Database 管理者ガイド』および『Oracle Database 概要』を参照してください。
> - 「DEFERRABLE 制約の例」（8-23 ページ）

[NOT] DEFERRABLE の制限事項 : ビュー制約にはこれらのパラメータを指定できません。

INITIALLY 句 INITIALLY 句は、DEFERRABLE 句が指定されている制約に対するデフォルトのチェック動作を指定します。INITIALLY 設定は、後続のトランザクションに SET CONSTRAINT(S) 文を指定することで上書きできます。

- INITIALLY IMMEDIATE を指定すると、この制約は後続の各 SQL 文の終わりでチェックされます。INITIALLY を指定しない場合、INITIALLY IMMEDIATE がデフォルトになります。

  新しい INITIALLY IMMEDIATE 制約を宣言する場合、CREATE TABLE または ALTER TABLE 文のコミット時にその制約が有効である必要があります。有効でない場合、これらの文は正常に実行されません。

- INITIALLY DEFERRED を指定すると、この制約は後続のトランザクションの終わりでチェックされます。

制約を NOT DEFERRABLE として宣言した場合、この句は無効です。NOT DEFERRABLE 制約は自動的に INITIALLY IMMEDIATE になり、INITIALLY DEFERRED に変更することはできないためです。

VALIDATE | NOVALIDATE VALIDATE および NOVALIDATE の動作は、制約が明示的にまたはデフォルトで使用可能 / 使用禁止のどちらになっているかで異なります。詳細は、8-14 ページの「ENABLE 句」および 8-15 ページの「DISABLE 句」を参照してください。

ENABLE 句 表のデータに制約を適用するには、ENABLE を指定します。

一意制約または主キー制約を使用可能にした場合、キーに索引が存在しないと、一意索引が作成されます。KEEP INDEX を指定しないかぎり、その後で制約が使用禁止になった場合にこの索引は削除されます。そのため、制約が使用可能になるたびに索引が再作成されます。

索引の再作成を避け、余分な索引を削除するには、最初に使用禁止にした主キー制約および一意制約を新しく作成します。その後、一意でない索引を作成して（または、既存の一意でない索引を使用して）制約を適用してください。制約が使用禁止の場合、一意でない索引は削除されず、後続の ENABLE 操作が容易になります。

- ENABLE VALIDATE を使用すると、すべての旧データと新データが制約に従うことを指定できます。制約が使用可能で、妥当性チェック済の場合は、すべてのデータが現在有効で、今後も有効であることが保証されます。

  整合性制約に違反する行が表にある場合、制約は使用禁止のままエラーを戻します。すべての行が制約に従っている場合、Oracle は制約を使用可能にします。新規データが整合性制約に違反する場合、その後の文は実行されず、整合性制約の違反を示すエラーが戻されます。

  主キー制約を ENABLE VALIDATE モードに設定すると、妥当性チェック・プロセスによって主キー列に NULL が含まれないことが検証されます。これによるオーバーヘッドを回避するには、データを列に入力する前およびこの表の主キー制約を使用可能にする前に、主キーの各列に NOT NULL のマークを付けます。

- ENABLE NOVALIDATE を使用すると、制約データに対して新しく行うすべての DML 操作が制約に従うことが保証されます。表の既存データが制約に従っていることは保証されません。

VALIDATE も NOVALIDATE も指定しない場合、VALIDATE がデフォルトになります。

ENABLE NOVALIDATE から ENABLE VALIDATE に単一制約状態を変更すると、パラレルで操作が実行できるため、読込み、書込みまたはその他の DDL 操作が中断されません。

ENABLE 句の制限事項 : 使用禁止になっている一意キーまたは主キーを参照する外部キーを使用可能にすることはできません。

DISABLE 句 整合性制約を使用禁止にする場合は、DISABLE を指定します。データ・ディクショナリでは、使用禁止になっている整合性制約は、使用可能な制約とともに表示されます。この句を指定せずに制約を作成した場合は、その制約は自動的に使用可能になります。

- DISABLE VALIDATE は、制約を使用禁止にして制約の索引を削除しますが、制約は有効のままです。この機能を使用すると、大量のデータをロードでき、また索引用の領域が削減されるため、データ・ウェアハウスで非常に有効です。この設定を使用することで、ALTER TABLE 文の *exchange_partition_clause* または SQL*Loader を使用して、データを非パーティション表からパーティション表にロードすることもできます。他の SQL 文を使用した表に対するすべての変更（挿入、更新および削除）は禁止されます。

  参照： この設定の使用方法の詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

- DISABLE NOVALIDATE は、Oracle によって制約がメンテナンスされないこと（使用禁止になっているため）、および制約が真であると保証されないこと（妥当性チェックが行われていないため）を示します。

  外部キー制約が DISABLE NOVALIDATE 状態であっても、外部キーが参照している主キーを持つ表を削除できません。また、オプティマイザは、DISABLE NOVALIDATE 状態でも制約を使用できます。

  参照： この設定の使用方法の詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。

VALIDATE も NOVALIDATE も指定しない場合、NOVALIDATE がデフォルトになります。

一意索引を使用している一意制約または主キー制約を使用禁止にすると、一意索引は削除されます。その他の注意事項や制限事項は、16-54 ページの「CREATE TABLE」の「enable_disable_clause」を参照してください。

RELY 句 RELY および NORELY は、既存の制約を変更する場合のみ（ALTER TABLE ... MODIFY 制約構文内でのみ）有効です。これらのパラメータは、クエリー・リライトで NOVALIDATE モードの制約を考慮するかどうかを指定します。RELY を指定すると、適用されていないクエリー・リライトに対する NOVALIDATE モードの既存の制約は、アクティブになります。その制約は NOVALIDATE モードであるため、適用されません。デフォルト値は NORELY です。

適用されない制約は、通常、マテリアライズド・ビューおよびクエリー・リライトにのみ有効です。QUERY_REWRITE_INTEGRITY モードに従って、クエリー・リライトは、VALIDATE モードの制約、または RELY パラメータが設定された NOVALIDATE モードの制約を使用して、結合情報を確認します。

RELY 句の制限事項： RELY には、遅延不可の NOT NULL 制約を設定できません。

参照： マテリアライズド・ビューおよびクエリー・リライトの詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

索引による制約の適用

一意制約または主キー制約の状態を定義する場合は、制約の適用に使用する索引を指定すること、または制約の適用に使用する索引を Oracle で自動作成することが可能です。

using_index_clause *using_index_clause* は、一意制約または主キー制約を使用可能にしている場合にのみ指定できます。*using_index_clause* の句はどのような順序でも指定できますが、各句を指定できるのは 1 回のみです。

- *schema.index* を指定すると、指定した索引を使用して制約が適用されます。索引がない、または制約の適用に索引が使用できない場合、エラーが戻されます。
- *create_index_statement* を指定すると、索引が作成され、制約の適用に使用されます。索引が作成できない、または制約の適用に索引が使用できない場合、エラーが戻されます。

- 既存の索引の指定も、新しい索引の作成も行わない場合は、索引が自動で作成されます。この場合、次のようになります。
  - 索引には制約と同じ名前が割り当てられます。
  - 表がパーティション化されている場合、一意制約または主キー制約にローカル・パーティション索引またはグローバル・パーティション索引を指定できます。

***using_index_clause* の制限事項：** *using_index_clause* には、次の制限事項があります。

- この句は、ビュー制約に対して指定できません。
- この句は、NOT NULL、外部キーまたは CHECK 制約には指定できません。
- 索引構成表の主キーが使用可能な場合は、索引の指定（*schema.index*）および作成（*create_index_statement*）はできません。
- *index_attributes* の *parallel_clause* は指定できません。
- *index_properties* の INDEXTYPE IS ... 句は、制約の定義では無効です。

参照：

- index_attributes、global_partitioned_index、local_partitioned_index 句の詳細、および索引に関連する NOSORT と *logging_clause* の詳細は、14-50 ページの「CREATE INDEX」を参照してください。
- 8-39 ページの「physical_attributes_clause」と「PCTFREE パラメータ」および 8-43 ページの「storage_clause」を参照してください。
- 「明示的な索引の制御例」（8-23 ページ）

制約の例外の処理

制約の状態を定義する場合は、制約に違反するすべての行の ROWID を格納する表を指定できます。

exceptions_clause *exceptions_clause* 構文を使用すると、例外の処理を定義できます。*schema* を指定しない場合、自分のスキーマ内に例外表があるとみなされます。この句自体を指定しない場合、表の名前は EXCEPTIONS になります。EXCEPTIONS 表または指定した表は、ローカル・データベースに存在する必要があります。

次のいずれかのスクリプトを使用して、EXCEPTIONS 表を作成できます。

- UTLEXCPT.SQL は、物理 ROWID を使用します。そのため、行は、索引構成表からではなく従来表から収集されます（次の注意を参照）。
- UTLEXPT1.SQL は、ユニバーサル ROWID を使用します。そのため、行は、従来表と索引構成表の両方から収集されます。

独自の例外表を作成する場合、これら 2 つのスクリプトのいずれかで規定される形式に従う必要があります。

ユニバーサル ROWID ではなく、主キーに基づく索引構成表から例外を収集する場合、索引構成表ごとに別の例外表を作成し、主キー記憶域を確保する必要があります。スクリプトを変更および再発行することによって、別の名前の例外表を複数作成できます。

***exceptions_clause* の制限事項：** *exceptions_clause* には、次の制限事項があります。

- この句は、ビュー制約に対して指定できません。
- この文が正常に終了されるまで ROWID は存在しないため、この句を CREATE TABLE 文に指定することはできません。

参照：

- SQL スクリプトの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の DBMS_IOT パッケージを参照してください。
- 移行行および連鎖行の削除については、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。

ビュー制約

データ・ウェアハウスのアプリケーションは、リレーショナル・スキーマ内の参照整合性制約を識別することによって、Oracle Database 内の多次元データを認識します。これらの制約によって、表間の主キーと外部キーの関係が表されます。Oracle Database データ・ディクショナリを問い合せることによって、アプリケーションはそのような制約を認識できるようになるため、データベース内の多次元データを認識できます。スキーマの複雑さまたはセキュリティ上の理由のため、ファクト表およびディメンション表にビューを定義する必要がある場合があります。Oracle Database には、これらのビューを制約する機能があります。ビュー間で制約を定義できるようにすることによって、実表制約をビューに伝播できます。これによって、ビューで提供される制限された環境でも、アプリケーションで多次元データを認識できるようにすることができます。

Oracle では、ビュー制約を適用していません。ただし、ビューに対する操作には、基となる実表に定義されている整合性制約が適用されます。つまり、実表に対する制約によって、ビューに制約を適用できます。

ビュー制約の注意事項： ビュー制約は表制約のサブセットで、次の制限事項があります。

- ビューには、一意制約、主キー制約および外部キー制約のみ指定できます。ただし、WITH CHECK OPTION 句を使用してビューを定義することは可能です。これは、ビューに CHECK 制約を指定することと同じです。
- ビュー制約は、DISABLE NOVALIDATE モードのみでサポートされています。他のモードは指定できません。ビュー制約を宣言する場合は、キーワード DISABLE を指定する必要があります。NOVALIDATE はデフォルトのため、明示的に指定する必要はありません。
- RELY および NORELY パラメータはオプションです。ビュー制約は適用されないため、通常は RELY パラメータで指定し、より有効に使用します。RELY または NORELY キーワードは、DISABLE キーワードより前に指定する必要があります。詳細は、8-15 ページの「RELY 句」を参照してください。
- ビュー制約は直接適用されないため、INITIALLY DEFERRED または DEFERRABLE は指定できません。
- *using_index_clause*、*exceptions_clause* 句、または *references_clause* の ON DELETE 句は指定できません。
- オブジェクト列の属性にビュー制約を定義することはできません。

例

一意キーの例 次の文は、サンプル表 sh.promotions を作成する文の例です。この文は制約を表内に定義し、promo_id 列で一意キーを暗黙的に使用可能にします（この例では、他の制約は省略されています）。

```
CREATE TABLE promotions_var1
    ( promo_id          NUMBER(6)
                        CONSTRAINT promo_id_u  UNIQUE
    , promo_name        VARCHAR2(20)
    , promo_category    VARCHAR2(15)
    , promo_cost        NUMBER(10,2)
    , promo_begin_date DATE
    , promo_end_date    DATE
    ) ;
```

制約 promo_id_u は、一意キーとして promo_id 列を識別します。この制約によって、表に同じ ID を持つ複数の販売促進の行が存在しないことが保証されます。ただし、識別子のない行は許可されます。

この制約を表外に定義して使用可能にすることもできます。

```
CREATE TABLE promotions_var2
    ( promo_id         NUMBER(6)
    , promo_name       VARCHAR2(20)
    , promo_category   VARCHAR2(15)
    , promo_cost       NUMBER(10,2)
    , promo_begin_date DATE
    , promo_end_date   DATE
    , CONSTRAINT promo_id_u UNIQUE (promo_id)
   USING INDEX PCTFREE 20
      TABLESPACE stocks
      STORAGE (INITIAL 8K  NEXT 6K) );
```

この文には *using_index_clause* も含まれています。この句は、制約を使用可能にするために作成される索引の記憶特性を指定します。

複合一意キーの例　次の文は、oe.warehouses 表の warehouse_id 列と warehouse_name 列を組み合せて複合一意キーを定義し、使用可能にします。

```
ALTER TABLE warehouses
   ADD CONSTRAINT wh_unq UNIQUE (warehouse_id, warehouse_name)
   USING INDEX PCTFREE 5
   EXCEPTIONS INTO wrong_id;
```

wh_unq 制約によって、warehouse_id と warehouse_name を組み合せた値が表の中に複数存在しないことが保証されます。

この ADD CONSTRAINT 句には、制約以外のプロパティも指定できます。

- USING INDEX 句は、制約を使用可能にするために作成される索引の記憶特性を指定します。
- EXCEPTIONS INTO 句によって、制約に違反する行が warehouses 表に含まれている場合、その行に関する情報が wrong_id 表に書き込まれます。wrong_id 例外表が存在しない場合、この文は正常に実行されません。

主キーの例　次の文は、サンプル表 hr.locations を作成する文の例です。locations_demo 表を作成し、location_id 列に主キーを定義して使用可能にします（この例では、hr.locations 表の他の制約は省略されています）。

```
CREATE TABLE locations_demo
    ( location_id    NUMBER(4) CONSTRAINT loc_id_pk PRIMARY KEY
    , street_address VARCHAR2(40)
    , postal_code    VARCHAR2(12)
    , city           VARCHAR2(30)
    , state_province VARCHAR2(25)
    , country_id     CHAR(2)
    ) ;
```

表内に指定されている loc_id_pk 制約は、location_id 列を locations_demo 表の主キーとして識別します。この制約によって、表の中の複数の所在地が同一の所在地識別子を持つことはなく、かつ所在地識別子が NULL にならないことが保証されます。

この制約を表外に定義して使用可能にすることもできます。

```
CREATE TABLE locations_demo
    ( location_id    NUMBER(4)
    , street_address VARCHAR2(40)
    , postal_code    VARCHAR2(12)
```

```
    , city           VARCHAR2(30)
    , state_province VARCHAR2(25)
    , country_id     CHAR(2)
    , CONSTRAINT loc_id_pk PRIMARY KEY (location_id));
```

NOT NULL の例　次の文は、(8-18 ページの「主キーの例」で作成した) locations_demo 表を変更し、country_id 列に NOT NULL 制約を定義して使用可能にします。

```
ALTER TABLE locations_demo
   MODIFY (country_id CONSTRAINT country_nn NOT NULL);
```

制約 country_nn によって、country_id が NULL の所在地の行が表にないことが保証されます。

複合主キーの例　次の文は、sh.sales サンプル表の prod_id 列および cust_id 列を組み合せて複合主キーを定義します。

```
ALTER TABLE sales
    ADD CONSTRAINT sales_pk PRIMARY KEY (prod_id, cust_id) DISABLE;
```

この制約は、sales 表の主キーとして prod_id 列と cust_id 列の組合せを識別します。この制約によって、表の中の複数の行が prod_id 列と cust_id 列に同じの組合せの値を持たないことが保証されます。

この制約句（PRIMARY KEY）では、次の制約のプロパティも指定しています。

- 制約定義によって制約名が指定されていないため、この制約に対する名前が Oracle によって自動的に生成されます。
- DISABLE 句によって制約が定義されますが、使用可能にはなりません。

外部キー制約の例　次の文は、dept_20 表を作成し、departments 表の department_id 列の主キーを参照する department_id 列に、外部キーを定義して使用可能にします。

```
CREATE TABLE dept_20
   (employee_id     NUMBER(4),
    last_name       VARCHAR2(10),
    job_id          VARCHAR2(9),
    manager_id      NUMBER(4),
    hire_date       DATE,
    salary          NUMBER(7,2),
    commission_pct  NUMBER(7,2),
    department_id   CONSTRAINT fk_deptno
                    REFERENCES departments(department_id) );
```

fk_deptno 制約によって、dept_20 表の従業員が属しているすべての部門が departments 表に含まれることが保証されます。ただし、部門番号が NULL 値の従業員（どの部門にも属さない従業員）がいてもかまいません。すべての従業員がいずれかの部門に割り当てられるようにするには、dept_20 表の department_id 列に対して、REFERENCES 制約の他に NOT NULL 制約を作成します。

この制約を定義して使用可能にする前に、departments 表の department_id 列を主キーまたは一意キーとして指定する制約を定義して使用可能にする必要があります。

制約が表内に定義されているため、この外部キー制約定義は FOREIGN KEY 句を使用しません。また、この列には参照キーのデータ型が自動的に割り当てられるため、*department_id* 列のデータ型は必要ありません。

制約定義によって親表と参照キーの列の両方が指定されます。参照キーは親表の主キーであるため、参照キーの列名の指定は任意です。

この外部キー制約を表外に定義することもできます。

```
CREATE TABLE dept_20
   (employee_id     NUMBER(4),
    last_name       VARCHAR2(10),
    job_id          VARCHAR2(9),
    manager_id      NUMBER(4),
    hire_date       DATE,
    salary          NUMBER(7,2),
    commission_pct  NUMBER(7,2),
    department_id,
   CONSTRAINT fk_deptno
      FOREIGN  KEY (department_id)
      REFERENCES  departments(department_id) );
```

これらの外部キー定義は、両方とも ON DELETE 句を指定していないため、従業員がいる部門は削除できません。

ON DELETE の例　次の文では、dept_20 表を作成し、2 つの参照整合性制約を定義して使用可能にした後、ON DELETE 句を使用します。

```
CREATE TABLE dept_20
   (employee_id     NUMBER(4) PRIMARY KEY,
    last_name       VARCHAR2(10),
    job_id          VARCHAR2(9),
    manager_id      NUMBER(4) CONSTRAINT fk_mgr
                    REFERENCES employees ON DELETE SET NULL,
    hire_date       DATE,
    salary          NUMBER(7,2),
    commission_pct  NUMBER(7,2),
    department_id   NUMBER(2)   CONSTRAINT fk_deptno
                    REFERENCES departments(department_id)
                    ON DELETE CASCADE );
```

最初の ON DELETE 句によって、従業員番号 2332 の上司が employees 表から削除された場合、その上司の部下だった dept_20 表のすべての従業員の manager_id 値が NULL になります。

次の ON DELETE 句によって、departments 表の department_id 値が削除されると、これに依存する dept_20 表の行の department_id 値も同時に削除されます。たとえば、部門番号 20 が departments 表から削除されると、部門番号 20 の全従業員が dept_20 表から削除されます。

複合外部キー制約の例　次の文は、dept_20 表の employee_id 列および hire_date 列を組み合せて外部キーを定義して使用可能にします。

```
ALTER TABLE dept_20
   ADD CONSTRAINT fk_empid_hiredate
   FOREIGN KEY (employee_id, hire_date)
   REFERENCES hr.job_history(employee_id, start_date)
   EXCEPTIONS INTO wrong_emp;
```

fk_empid_hiredate 制約によって、dept_20 表の中すべての従業員が、employees 表に存在する employee_id と hire_date の組合せを持つことが保証されます。この制約を定義して使用可能にする前に、employees 表の employee_id 列と hire_date 列の組合せを主キーまたは一意キーとして指定する制約を定義して使用可能にする必要があります。

EXCEPTIONS INTO 句によって、制約に違反する行が dept_20 表に含まれている場合、その行に関する情報が wrong_emp 表に書き込まれます。wrong_emp 例外表が存在しない場合、この文は正常に実行されません。

CHECK 制約の例 次の文は、divisions 表を作成し、その表の各列に check 制約を定義します。

```
CREATE TABLE divisions
   (div_no    NUMBER  CONSTRAINT check_divno
              CHECK (div_no BETWEEN 10 AND 99)
              DISABLE,
    div_name  VARCHAR2(9)  CONSTRAINT check_divname
              CHECK (div_name = UPPER(div_name))
              DISABLE,
    office    VARCHAR2(10)  CONSTRAINT check_office
              CHECK (office IN ('DALLAS','BOSTON',
              'PARIS','TOKYO'))
              DISABLE);
```

列に定義されている各制約によって、列の値が次のように制限されます。

- check_divno によって、部門番号は必ず 10 ～ 99 の範囲内になります。
- check_divname によって、部門名はすべて大文字になります。
- check_office によって、事務所の所在地は Dallas、Boston、Paris または Tokyo のいずれかに制限されます。

それぞれの CONSTRAINT 句に DISABLE 句が指定されているため、これらの制約は定義されるのみで、使用可能にはされません。

次の文は、dept_20 表を作成し、CHECK 制約を表外に定義して暗黙的に使用可能にします。

```
CREATE TABLE dept_20
   (employee_id     NUMBER(4) PRIMARY KEY,
    last_name       VARCHAR2(10),
    job_id          VARCHAR2(9),
    manager_id      NUMBER(4),
    salary          NUMBER(7,2),
    commission_pct  NUMBER(7,2),
    department_id   NUMBER(2),
    CONSTRAINT check_sal CHECK (salary * commission_pct <= 5000));
```

この制約は、不等式の条件を使用して、従業員の歩合総額（salary と commission_pct を掛けた金額）を 5,000 ドルに制限します。

- 従業員の給与と歩合が NULL 以外の値の場合、制約を満たすには、この金額が 5,000 ドルを超えてはいけません。
- 従業員の給与または歩合が NULL 値の場合、条件の結果は不明となり、その従業員は自動的に制約を満たします。

この例の制約句には制約名が指定されていないため、制約の名前が自動的に生成されます。

次の文は、1 つの主キー制約、2 つの外部キー制約、1 つの NOT NULL 制約および 2 つの CHECK 制約を定義して使用可能にします。

```
CREATE TABLE order_detail
  (CONSTRAINT pk_od PRIMARY KEY (order_id, part_no),
   order_id    NUMBER
      CONSTRAINT fk_oid
         REFERENCES oe.orders(order_id),
   part_no     NUMBER
      CONSTRAINT fk_pno
         REFERENCES oe.product_information(product_id),
   quantity    NUMBER
      CONSTRAINT nn_qty NOT NULL
      CONSTRAINT check_qty CHECK (quantity > 0),
   cost        NUMBER
      CONSTRAINT check_cost CHECK (cost > 0) );
```

この制約によって、表のデータに対して次の規則を適用できます。

- pk_od は、order_id 列と part_no 列の組合せを表の主キーとして指定します。この制約を満たすためには、order_id 列および part_no 列の組合せが同じである行が、表内に 2 つあってはいけません。また、order_id 列および part_no 列では、行に NULL を入れることはできません。
- fk_oid は、サンプル・スキーマ oe 内の orders 表にある order_id 列を参照する外部キーとして、order_id 列を指定します。order_detail.order_id 列に追加されるすべての新しい値は、oe.orders.order_id 列にあらかじめ存在する必要があります。
- fk_pno は、oe が所有する product_information 表にある product_id 列を参照する外部キーとして、product_id 列を指定します。order_detail.product_id 列に追加されるすべての新しい値は、oe.product_information.product_id 列にあらかじめ存在する必要があります。
- nn_qty は、quantity 列に対して NULL を禁止します。
- check_qty によって、quantity 列の値は必ず 0（ゼロ）より大きくなります。
- check_cost によって、cost 列の値は必ず 0（ゼロ）より大きくなります。

この例は、制約句と列定義について、次の点についても示しています。

- 表外制約定義は、列定義の前と後のどちらにも指定できます。この例では、pk_od 制約の表外定義が、列定義の前にあります。
- 列定義には、複数の表内制約定義を含めることができます。この例では、quantity 列の定義は nn_qty 制約と check_qty 制約の両方の定義を含んでいます。
- 表には、複数の CHECK 制約を指定できます。複数のビジネス・ルールを適用する複雑な条件を持つ 1 つの CHECK 制約よりも、それぞれ 1 つのビジネス・ルールのみを適用する単純な条件を持つ複数の CHECK 制約を使用してください。矛盾している制約があると、その制約を識別するエラー・メッセージが戻されます。エラーが検出された制約が 1 つのビジネス・ルールのみを有効にする場合、このようなエラー・メッセージの方が、矛盾のあるビジネス・ルールを正確に識別できます。

属性レベル制約の例　次の文は、students 表の name 列の first_name 属性と last_name 属性の両方に対して値が存在することを保証します。

```
CREATE TYPE person_name AS OBJECT
   (first_name VARCHAR2(30), last_name VARCHAR2(30));
/

CREATE TABLE students (name person_name, age INTEGER,
   CHECK (name.first_name IS NOT NULL AND
          name.last_name IS NOT NULL));
```

REF 制約の例　次の例は、サンプル・スキーマのオブジェクト型である cust_address_typ の複製を作成し、SCOPE 制約が指定された REF 列を含む表を作成します。

```
CREATE TYPE cust_address_typ_new AS OBJECT
    ( street_address     VARCHAR2(40)
    , postal_code        VARCHAR2(10)
    , city               VARCHAR2(30)
    , state_province     VARCHAR2(10)
    , country_id         CHAR(2)
    );
/
CREATE TABLE address_table OF cust_address_typ_new;

CREATE TABLE customer_addresses (
   add_id NUMBER,
   address REF cust_address_typ_new
   SCOPE IS address_table);
```

次の例は、同じ表を作成しますが、親表のオブジェクト識別子列を参照する REF 列に参照整合性制約が指定されています。

```
CREATE TABLE customer_addresses (
   add_id NUMBER,
   address REF cust_address_typ REFERENCES address_table);
```

次の例では、department_typ 型および departments_obj_t 表（16-69 ページの「オブジェクト表の作成例 :」で作成）を使用します。この文を実行すると、有効範囲付き REF を持つ表が作成されます。

```
CREATE TABLE employees_obj
   ( e_name   VARCHAR2(100),
     e_number NUMBER,
     e_dept   REF department_typ SCOPE IS departments_obj_t );
```

次の文は、参照整合性制約が定義されている REF 列を含む表を作成します。

```
CREATE TABLE employees_obj
   ( e_name   VARCHAR2(100),
     e_number NUMBER,
     e_dept   REF department_typ REFERENCES departments_obj_t);
```

明示的な索引の制御例　次の文は、Oracle が制約を適用する場合に使用する索引を、明示的に制御する一意制約または主キー制約を作成する別の方法を示します。

```
CREATE TABLE promotions_var3
    ( promo_id          NUMBER(6)
    , promo_name        VARCHAR2(20)
    , promo_category    VARCHAR2(15)
    , promo_cost        NUMBER(10,2)
    , promo_begin_date DATE
    , promo_end_date    DATE
    , CONSTRAINT promo_id_u UNIQUE (promo_id, promo_cost)
         USING INDEX (CREATE UNIQUE INDEX promo_ix1
            ON promotions_var3 (promo_id, promo_cost))
    , CONSTRAINT promo_id_u2 UNIQUE (promo_cost, promo_id)
         USING INDEX promo_ix1);
```

この例は、1 つの制約に対して 1 つの索引を作成し、その索引を使用して同じ文に別の制約を作成して使用可能にできることも示しています。

DEFERRABLE 制約の例　次の文は、scores 列に CHECK 制約 NOT DEFERRABLE INITIALLY IMMEDIATE（デフォルト）を使用した games 表を作成します。

```
CREATE TABLE games (scores NUMBER CHECK (scores >= 0));
```

次の文は、列に対する一意制約を INITIALLY DEFERRED DEFERRABLE として定義します。

```
CREATE TABLE games
  (scores NUMBER, CONSTRAINT unq_num UNIQUE (scores)
   INITIALLY DEFERRED DEFERRABLE);
```

deallocate_unused_clause

用途

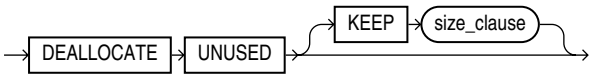
deallocate_unused_clause を使用すると、データベース・オブジェクト・セグメントの終わりにある未使用領域の割当てを明示的に解除し、解放された領域を表領域の他のセグメントで使用できるようになります。

未使用領域の割当ては、次の文で解除できます。

- ALTER CLUSTER（10-5 ページの「ALTER CLUSTER」を参照）
- ALTER INDEX: 未使用領域の割当てを索引、索引パーティションまたは索引サブパーティションから解除する場合（10-64 ページの「ALTER INDEX」を参照）
- ALTER MATERIALIZED VIEW: 未使用領域の割当てを索引構成マテリアライズド・ビューのオーバーフロー・セグメントから解除する場合（11-2 ページの「ALTER MATERIALIZED VIEW」を参照）
- ALTER TABLE: 未使用領域の割当てを表、表パーティション、表サブパーティション、索引構成表のマッピング表、索引構成表のオーバーフロー・セグメントまたは LOB 記憶域セグメントから解除する場合（12-2 ページの「ALTER TABLE」を参照）

構文

***deallocate_unused_clause*::=**



（8-42 ページの size_clause::= を参照）

セマンティクス

この項では、*deallocate_unused_clause* のセマンティクスについて説明します。詳細は、特定のデータベース・オブジェクトに対してこの句を設定または再設定する SQL 文の説明を参照してください。

同じ文で *deallocate_unused_clause* と *allocate_extent_clause* の両方を指定することはできません。

最高水位標を超える（データベース・ブロックがデータを受け取るためにフォーマットされていない）未使用領域のみ解放できます。未使用領域の割当て解除は、オブジェクトの終わりから開始し、オブジェクトの先頭にある最高水位標に向かって進行します。

エクステントが割当て解除の範囲内に完全に含まれる場合、エクステント全体が解放されて再利用可能となります。エクステントの一部が含まれる場合、最高水位標までのすでに使用されている部分がエクステントになり、残りの未使用領域が解放されて再利用可能になります。

割当てが解除される表領域のユーザー割当て制限は、解放される領域の量のみ増加します。

解放される領域の実際の量は、INITIAL、MINEXTENTS および NEXT 記憶域パラメータによって異なります。これらのパラメータの詳細は、8-43 ページの「storage_clause」を参照してください。

KEEP *integer*

割当て解除後に、データベース・オブジェクトのセグメントが保持する、最高水位標を超えるバイト数を指定します。

- KEEP を指定しなかった場合に、最高水位標が INITIAL および MINEXTENTS のサイズを超えると、最高水位標を超えるすべての未使用領域が解放されます。最高水位標が INITIAL または MINEXTENTS のサイズより小さい場合は、MINEXTENTS を超えるすべての未使用領域が解放されます。
- KEEP オプションを指定した場合、指定した量の領域が保持され、残りの領域のみが解放されます。このとき、残りのエクステント数が MINEXTENTS より小さくなった場合、MINEXTENTS はそのエクステント数に変更されます。また、初期エクステントが INITIAL より小さくなった場合、INITIAL がそのサイズに変更されます。
- どちらの場合にも、NEXT 記憶域パラメータの値は、割当て解除された最後のエクステントのサイズに設定されます。

file_specification

用途

file_specification 構文のいずれかを使用すると、ファイルをデータ・ファイルまたは一時ファイルとして指定できます。また、1 つ以上のファイルのグループを 1 つの REDO ログ・ファイル・グループとして指定することもできます。自動ストレージ管理ディスク・グループにファイルを格納する場合、そのファイルをディスク・グループ・ファイルとしても指定できます。

file_specification は次の文に指定できます。

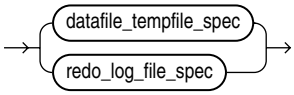
- CREATE CONTROLFILE（14-10 ページの「CREATE CONTROLFILE」を参照）
- CREATE DATABASE（14-16 ページの「CREATE DATABASE」を参照）
- ALTER DATABASE（10-9 ページの「ALTER DATABASE」を参照）
- CREATE TABLESPACE（16-71 ページの「CREATE TABLESPACE」を参照）
- ALTER TABLESPACE（12-82 ページの「ALTER TABLESPACE」を参照）
- ALTER DISKGROUP（10-46 ページの「ALTER DISKGROUP」を参照）

前提条件

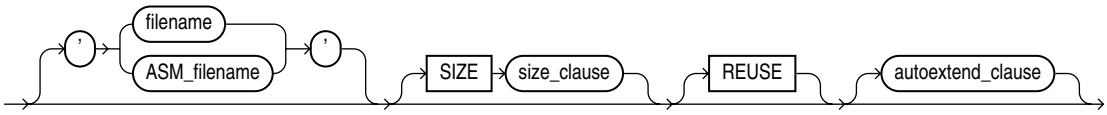
ファイルを指定する文を発行する権限が必要です。

構文

***file_specification*::=**

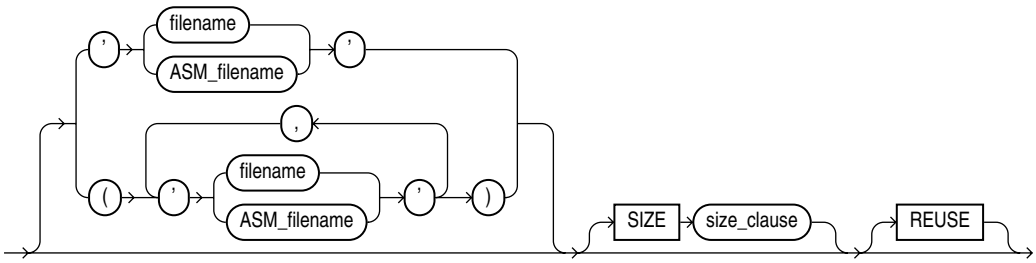


datafile_tempfile_spec::=



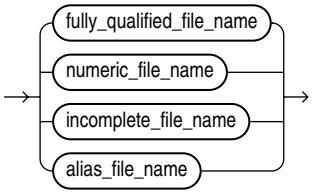
（8-42 ページの size_clause::= を参照）

redo_log_file_spec::=

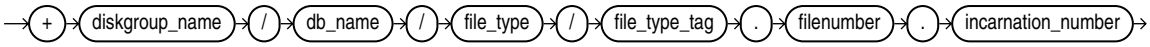


（8-42 ページの size_clause::= を参照）

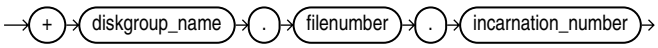
ASM_filename::=



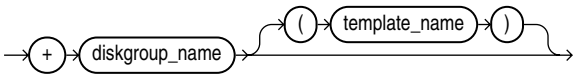
fully_qualified_file_name::=



numeric_file_name::=

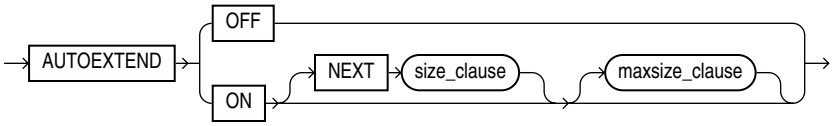


incomplete_file_name::=



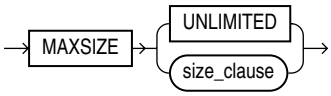
alias_file_name::=

***autoextend_clause*::=**



(8-42 ページの size_clause::= を参照)

***maxsize_clause*::=**



(8-42 ページの size_clause::= を参照)

セマンティクス

この項では、*file_specification* のセマンティクスについて説明します。詳細は、データ・ファイル、一時ファイル、REDO ログ・ファイルまたは自動ストレージ管理ディスク・グループやディスク・グループ・ファイルを指定する SQL 文の説明を参照してください。

datafile_tempfile_spec

データベース記憶域がファイル・システム内、RAW デバイス上または自動ストレージ管理ディスク・グループ内に存在する場合、この句を使用してデータ・ファイルと一時ファイルの属性を指定します。

redo_log_file_spec

データベース記憶域がファイル・システム内、RAW デバイス上または自動ストレージ管理ディスク・グループ内に存在する場合、この句を使用して REDO ログ・ファイルの属性を指定します。

filename

filename は、ファイル・システム内または RAW デバイス上に格納されたファイルに使用します。*filename* は、新しいファイルまたは既存のファイルを指定できます。新しいファイルを指定する場合、次のことに注意します。

- Oracle Managed Files を使用していない場合は、*filename* と SIZE 句の両方を指定しないと、文は失敗します。サイズを指定せずにファイル名を指定すると、Oracle は既存のファイルを再利用し、ファイルが存在しない場合はエラーを戻そうとします。
- Oracle Managed Files を使用している場合は、*filename* や残りの句は必須ではありません。この場合、Oracle Database によってファイルに一意の名前が作成され、そのファイルが次のいずれかの初期化パラメータで指定されるディレクトリに保存されます。
  - DB_RECOVERY_FILE_DEST（ログ・ファイルと制御ファイルに有効）
  - DB_CREATE_FILE_DEST 初期化パラメータ（すべてのファイル・タイプに有効）
  - DB_CREATE_ONLINE_LOG_DEST_*n* 初期化パラメータ（ログ・ファイルに対しては DB_CREATE_FILE_DEST と DB_RECOVERY_FILE_DEST より優先される）

既存のファイルを指定する場合は、データ・ファイル、一時ファイルまたは REDO ログ・ファイル・メンバーの名前を指定します。*filename* には、7 ビット ASCII キャラクタ・セットまたは EBCDIC キャラクタ・セットのシングルバイト文字のみを使用できます。マルチバイト文字は無効です。

filename には、パス接頭辞を含めることができます。パス接頭辞を指定しない場合は、データベースによってデフォルトの格納場所（プラットフォームによって異なる）のパス接頭辞が追加されます。

REDO ログ・ファイル・グループは、1 つ以上のメンバー（コピー）で構成されます。*filename* は、使用するオペレーティング・システムの表記規則に従って、完全な名前で指定する必要があります。

データベースが *filename* を解析する方法は、SIZE 句および REUSE 句の指定によっても異なります。

- *filename* のみを指定する場合、または REUSE 句を指定して SIZE 句を指定しない場合、そのファイルはすでに存在している必要があります。
- *filename* の指定に、SIZE 句を指定して、REUSE 句を指定しない場合、そのファイルは新しいファイルである必要があります。
- *filename* の指定に、SIZE 句と REUSE 句の両方を指定する場合、そのファイルは新しいファイル、既存のファイルのどちらでもかまいません。ファイルがすでに存在している場合は、そのファイルが新しいサイズで再利用されます。また、ファイルが存在しない場合は、データベースは REUSE キーワードを無視し、指定のサイズで新しいファイルを作成します。

> **参照：** Oracle Managed Files の詳細は、『Oracle Database ストレージ管理者ガイド』を参照してください。また、8-33 ページの「データ・ファイルの指定例」および 8-32 ページの「ログ・ファイルの指定例」も参照してください。

ASM_filename

自動ストレージ管理ディスク・グループに格納されたファイルには、ASM_filename の形式を使用します。この構文でデータ・ファイル、一時ファイルおよび REDO ログ・ファイルを作成または参照できます。

すべての ASM_filename 形式は、プラス記号（+）で始まりディスク・グループ名が続きます。すべての自動ストレージ管理ディスク・グループの名前を確認するには、V$ASM_DISKGROUP ビューを問い合せます。

> **参照：** 自動ストレージ管理の使用については、『Oracle Database ストレージ管理者ガイド』を参照してください。

fully_qualified_file_name

自動ストレージ管理ディスク・グループにファイルを作成する場合、そのファイルにはシステム生成の完全修飾された自動ストレージ管理ファイル名が付けられます。既存の自動ストレージ管理ファイルを参照する場合にのみ、この形式を使用できます。そのため、ファイル作成時にこの形式を使用する場合は、REUSE も指定する必要があります。

- *db_name* は、DB_UNIQUE_NAME 初期化パラメータの値です。この名前はファイルが存在するデータベースの名前と同じです。ただし、プライマリ・データベースとスタンバイ・データベースの両方が存在する場合は、パラメータはこれらを区別して異なる値になります。
- *file_type* と *file_type_tag* は、データベース・ファイル・タイプです。8-29 ページの表 8-1 に、すべてのファイル・タイプとそれに対応する自動ストレージ管理タグを示します。
- *filenumber* と *incarnation_number* は、システム生成の識別子で、これによって一意性が保証されます。

完全修飾された自動ストレージ管理ファイル名を確認するには、そのファイル・タイプに適切な動的パフォーマンス・ビュー（データ・ファイルでは V$DATAFILE、制御ファイルでは V$CONTROLFILE など）を問い合せます。また、V$ASM_FILE ビューを問い合せることによって、完全修飾された名前の *filenumber* と *incarnation_number* の部分を取得できます。

表 8-1 Oracle ファイル・タイプと自動ストレージ管理ファイル・タイプ・タグ

自動ストレージ管理 *file_type*	説明	自動ストレージ管理 *file_type_tag*	コメント
CONTROLFILE	制御ファイルとバックアップ制御ファイル	Current Backup	—
DATAFILE	データ・ファイルとデータ・ファイル・コピー	*tsname*	ファイルが追加される表領域です。
ONLINELOG	オンライン・ログ	group_*group#*	—
ARCHIVELOG	アーカイブ・ログ	thread_*thread#*_seq_*sequence#*	—
TEMPFILE	一時ファイル	*tsname*	ファイルが追加される表領域です。
BACKUPSET	データ・ファイルとアーカイブ・ログのバックアップ・ピース（データ・ファイルの増分バックアップ・ピース）	*hasspfile_timestamp*	*hasspfile* の値は、*s*（バックアップ・セットが spfile を含む）または *n*（バックアップ・セットが spfile を含まない）のいずれかになります。
PARAMETERFILE	永続パラメータ・ファイル	spfile	—
DAATAGUARDCONFIG	Data Guard 構成ファイル	*db_unique_name*	サービス・プロバイダ名が設定されている場合、Data Guard はその名前を使用しようとします。設定されていない場合、タグのデフォルトは *DRCname* です。
FLASHBACK	フラッシュバック・ログ	log_*log#*	—

表 8-1 Oracle ファイル・タイプと自動ストレージ管理ファイル・タイプ・タグ（続き）

自動ストレージ管理 *file_type*	説明	自動ストレージ管理 *file_type_tag*	コメント
CHANGETRACKING	ブロック・チェンジ・トラッキング・データ	ctf	増分バックアップ中に使用されます。
DUMPSET	データ・ポンプ・ダンプ・セット	user_*obj#_file#*	ダンプ・セット・ファイルは、ユーザー名、ダンプ・セットが作成されたジョブ番号およびファイル番号をタグの一部としてエンコードします。
XTRANSPORT	データ・ファイル変換	*tsname*	—
AUTOBACKUP	自動バックアップ・ファイル	*hasspfile_timestamp*	*hasspfile* の値は、*s*（バックアップ・セットが spfile を含む）または *n*（バックアップ・セットが spfile を含まない）のいずれかになります。

numeric_file_name

数値の自動ストレージ管理ファイル名は、一意の *filenumber.incarnation_number* 文字列のみが使用されることを除き、完全修飾されたファイル名に類似しています。既存のファイルを参照するためにのみ、この形式を使用できます。そのため、ファイル作成時にこの形式を使用する場合は、REUSE も指定する必要があります。

incomplete_file_name

不完全な自動ストレージ管理ファイル名は、ファイルの作成時にのみ使用されます。ディスク・グループ名のみを指定した場合、自動ストレージ管理ではそのファイル・タイプに適したデフォルトのテンプレートが使用されます。たとえば、CREATE TABLESPACE 文でデータ・ファイルを作成する場合、自動ストレージ管理ではデフォルトの DATAFILE テンプレートを使用して自動ストレージ管理データ・ファイルが作成されます。ディスク・グループ名とテンプレートを指定した場合、自動ストレージ管理では指定したテンプレートを使用してファイルが作成されます。いずれの場合も、自動ストレージ管理によって完全修飾されたファイル名が作成されます。

template_name テンプレートは、属性の名前付きコレクションです。テンプレートを作成して、ディスク・グループのファイルに適用できます。すべての自動ストレージ管理テンプレートの名前を確認するには、V$ASM_TEMPLATE データ・ディクショナリ・ビューを問い合せます。自動ストレージ管理テンプレートを作成する手順は、10-54 ページの「diskgroup_template_clauses」を参照してください。

template は、ファイルの作成時にのみ指定できます。これは、*ASM_filename* の構文図の不完全なファイル名形式および別名形式に示されています。

- ディスク・グループ名の直後に *template* を指定した場合、自動ストレージ管理では指定したテンプレートを使用してファイルが作成され、そのファイルに完全修飾されたファイル名が付けられます。
- 別名を指定した後に *template* を指定した場合、自動ストレージ管理では指定したテンプレートを使用してファイルが作成され、そのファイルに完全修飾されたファイル名が付けられます。また、後でファイルの参照に使用できる別名も作成されます。指定した別名が既存のファイルを参照している場合は、REUSE を指定しないかぎり、自動ストレージ管理ではテンプレートの指定が無視されます。

参照： デフォルト・テンプレートについては、10-54 ページの「diskgroup_template_clauses」を参照してください。

alias_file_name

別名は、自動ストレージ管理ファイルのわかりやすい名前です。ファイルの別名は、ファイルの作成または参照時に使用できます。ファイルの作成時にのみ、別名とともにテンプレートを指定できます。自動ストレージ管理ファイルの別名を確認するには、V$ASM_ALIAS データ・ディクショナリ・ビューを問い合せます。

ファイルの作成時に別名を指定する場合、完全な別名を指定する手順は、10-56 ページの「diskgroup_directory_clauses」および 10-56 ページの「diskgroup_alias_clauses」を参照してください。

SIZE 句

ファイルのサイズをバイト単位で指定します。K、M、G または T を使用して、KB、MB、GB または TB 単位で指定することもできます。

- UNDO 表領域の場合、各データ・ファイルに対して SIZE 句を指定する必要があります。その他の表領域の場合、ファイルがすでに存在するとき、または Oracle Managed Files を作成するときには、このパラメータを省略できます。
- Oracle Managed Files の作成時にこの句を指定しないと、100MB のファイルが作成されます。
- 表領域は、中に含まれるオブジェクトのサイズの合計より 1 ブロック大きいサイズである必要があります。

> **参照：** 自動 UNDO 管理および UNDO 表領域の詳細は、『Oracle Database 管理者ガイド』を参照してください。また、8-32 ページの「ログ・ファイルの追加例」も参照してください。

REUSE

REUSE は、既存ファイルの再利用を可能にします。

- ファイルがすでに存在する場合、ファイル名が再利用され、新しいサイズが適用されるか（SIZE を指定する場合）、または元のサイズのままとなります。
- ファイルが存在していない場合、この句は無視され、ファイルが作成されます。

REUSE 句の制限事項： *filename* を指定しないかぎり、REUSE は指定できません。

既存のファイルが使用される場合は、そのファイルに入っていた内容は失われます。

> **参照：** 8-33 ページの「データ・ファイルの追加例」および 8-32 ページの「ログ・ファイルの追加例」を参照してください。

autoextend_clause

autoextend_clause はデータ・ファイルと一時ファイルに有効ですが、REDO ログ・ファイルには有効ではありません。この句は、新しいデータ・ファイル、既存のデータ・ファイルまたは一時ファイルの自動拡張を使用可能または使用禁止にします。この句を指定しない場合、次のようになります。

- Oracle Managed Files の場合
  - SIZE を指定すると、指定したサイズのファイルが作成され、AUTOEXTEND が使用禁止になります。
  - SIZE を指定しないと、100MB のファイルが作成され、AUTOEXTEND が使用可能になります。自動拡張が必要な場合、データベースは元のサイズまたは 100MB（いずれか小さい方）のサイズ単位でファイルを拡張します。NEXT 句を指定して、このデフォルトの動作を上書きできます。
- ユーザー管理ファイルの場合、SIZE の指定 / 未指定にかかわらず、AUTOEXTEND が使用禁止にされたファイルが作成されます。

ON ON を指定すると、自動拡張を使用可能にします。

OFF OFF を指定すると、自動拡張を使用禁止にします。自動拡張を使用禁止にする場合は、NEXT および MAXSIZE の値を 0（ゼロ）に設定します。後続の文で自動拡張を使用可能に戻す場合は、これらの値を設定しなおす必要があります。

NEXT NEXT 句を使用すると、エクステントがさらに必要になった場合にデータ・ファイルに自動的に割り当てられるディスク領域の増分サイズを、バイト単位で指定できます。デフォルトのサイズは 1 データ・ブロックです。

MAXSIZE MAXSIZE 句を使用すると、データ・ファイルの自動拡張で使用されるディスク領域の最大サイズを指定できます。

UNLIMITED データ・ファイルまたは一時ファイルに割り当てられるディスク領域を制限しない場合は、UNLIMITED 句を使用します。

***autoextend_clause* の制限事項：** この句は CREATE CONTROLFILE 文の *datafile_tempfile_spec* の一部、または ALTER DATABASE CREATE DATAFILE 句には指定できません。

例

ログ・ファイルの指定例 次の文は、payable という名前のデータベースを作成します。このデータベースには各グループにメンバーを 2 つ持つ REDO ログ・ファイル・グループが 2 つと、データ・ファイルが 1 つ設定されています。

```
CREATE DATABASE payable
   LOGFILE GROUP 1 ('diska:log1.log', 'diskb:log1.log') SIZE 50K,
           GROUP 2 ('diska:log2.log', 'diskb:log2.log') SIZE 50K
   DATAFILE 'diskc:dbone.dat' SIZE 30M;
```

LOGFILE 句の最初のファイル指定は、REDO ログ・ファイル・グループに GROUP 1 の値を指定します。このグループには、'diska:log1.log' および 'diskb:log1.log' というメンバーがあり、サイズはそれぞれ 50KB です。

LOGFILE 句の 2 番目のファイル指定は、REDO ログ・ファイル・グループに GROUP 2 の値を指定します。このグループには、'diska:log2.log' および 'diskb:log2.log' というメンバーがあり、サイズはそれぞれ 50KB です。

DATAFILE 句のファイル指定では、'diskc:dbone.dat' という名前のデータ・ファイルが指定されています。このファイルのサイズは 30MB です。

各ファイル指定は SIZE パラメータの値を指定し、REUSE 句は指定していないため、指定のファイルがすでに定義されていることはありません。Oracle が新しくファイルを作成します。

ログ・ファイルの追加例 次の文は、2 つのメンバーで構成される新しい REDO ログ・ファイル・グループを、payable データベースに追加します。

```
ALTER DATABASE payable
   ADD LOGFILE GROUP 3 ('diska:log3.log', 'diskb:log3.log')
   SIZE 50K REUSE;
```

ADD LOGFILE 句のファイル指定は、REDO ログ・ファイル・グループに GROUP 3 の値を指定します。このグループには、'diska:log3.log' および 'diskb:log3.log' というメンバーがあり、サイズはそれぞれ 50KB です。このファイル指定では REUSE 句が指定されているため、各メンバーはすでに存在している可能性があります（ただし、存在していなくてもかまいません）。

データ・ファイルの指定例　次の文は、stocks という名前の表領域を作成します。また、この表領域にはデータ・ファイルが 3 つあります。

```
CREATE TABLESPACE stocks
   DATAFILE 'stock1.dat' SIZE 10M,
            'stock2.dat' SIZE 10M,
            'stock3.dat' SIZE 10M;
```

このデータ・ファイルのファイル指定は、'diskc:stock1.dat'、'diskc:stock2.dat' および 'diskc:stock3.dat' という名前のファイルを指定します。

データ・ファイルの追加例　次の文は、stocks 表領域を変更し、新しいデータ・ファイルを追加します。

```
ALTER TABLESPACE stocks
   ADD DATAFILE 'stock4.dat' SIZE 10M REUSE;
```

このファイル指定は、'stock4.dat' という名前のデータ・ファイルを指定します。このファイル名が存在しない場合、REUSE キーワードは無視されます。

完全修飾された自動ストレージ管理データ・ファイル名の使用例　次の文は、自動ストレージ管理を使用する際に、*fully_qualified_file_name* 句を使用して、仮想データベース testdb のデータ・ファイルをオンラインにする方法を示します。

```
ALTER DATABASE testdb
   DATAFILE '+dgroup_01/testdb/datafile/system.261.1' ONLINE;
```

logging_clause

用途

logging_clause を使用すると、データベース・オブジェクトの作成が REDO ログ・ファイルに記録されるか（LOGGING）、記録されないか（NOLOGGING）を指定できます。

logging_clause は次の文に指定できます。

- CREATE TABLE および ALTER TABLE: 表、表のパーティションの 1 つ、LOB セグメント、または索引構成表のオーバーフロー・セグメントのログを記録する場合（16-6 ページの「CREATE TABLE」および 12-2 ページの「ALTER TABLE」を参照）

  注意： LOB 列に指定されたロギングは、表レベルで設定されたロギングとは異なる場合があります。表レベルで LOGGING を指定し、LOB 列に NOLOGGING を指定する場合は、実表の行に対する DML の変更は記録されますが、LOB データに対する DML の変更は記録されません。

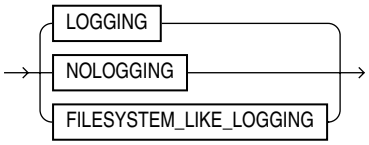
- CREATE INDEX および ALTER INDEX: 索引または索引のパーティションの 1 つのログを記録する場合（14-50 ページの「CREATE INDEX」および 10-64 ページの「ALTER INDEX」を参照）
- CREATE MATERIALIZED VIEW および ALTER MATERIALIZED VIEW: マテリアライズド・ビュー、マテリアライズド・ビューのパーティションの 1 つまたは LOB セグメントのログを記録する場合（15-4 ページの「CREATE MATERIALIZED VIEW」および 11-2 ページの「ALTER MATERIALIZED VIEW」を参照）
- CREATE MATERIALIZED VIEW LOG および ALTER MATERIALIZED VIEW LOG: マテリアライズド・ビュー・ログまたはマテリアライズド・ビュー・ログのパーティションの 1 つのログを記録する場合（15-26 ページの「CREATE MATERIALIZED VIEW LOG」および 11-16 ページの「ALTER MATERIALIZED VIEW LOG」を参照）
- CREATE TABLESPACE および ALTER TABLESPACE: 表領域に作成されたすべてのオブジェクトに対するデフォルトのロギング特性を設定または変更する場合（16-71 ページの「CREATE TABLESPACE」および 12-82 ページの「ALTER TABLESPACE」を参照）

次の操作にも、LOGGING または NOLOGGING を指定できます。

- 索引の再構築（CREATE INDEX ... REBUILD を使用）
- 表の移動（ALTER TABLE ... MOVE を使用）

構文

***logging_clause*::=**



セマンティクス

この項では、*logging_clause* のセマンティクスについて説明します。詳細は、特定のデータベース・オブジェクトに対してロギング特性を設定または再設定する SQL 文の説明を参照してください。

- データベース・オブジェクトの作成、およびその後のオブジェクトへの挿入を REDO ログ・ファイルに記録する場合は、LOGGING を指定します。
- これらの操作を記録しない場合は、NOLOGGING を指定します。
  - **非パーティション・オブジェクト**の場合、この句に指定する値は、オブジェクトに関連付けられたセグメントの実際の物理属性となります。
  - **パーティション・オブジェクト**の場合、この句に指定する値は、PARTITION 記述でロギング属性を指定しないかぎり、CREATE 文（および後続の ALTER ... ADD PARTITION 文）で指定するすべてのパーティションに関連付けられたセグメントのデフォルトの物理属性となります。
  - SecureFile LOB の場合、NOLOGGING 設定は FILESYSTEM_LIKE_LOGGING に内部的に変換されます。
- FILESYSTEM_LIKE_LOGGING 句は、SecureFile LOB セグメントのログを記録する場合にのみ有効です。この設定は、BasicFile LOB には指定できません。メタデータ変更のログのみを記録する場合は、この設定を指定します。この設定は、障害からリカバリする平均時間を短縮するファイル・システムのメタデータ・ジャーナリングに似ています。SecureFile LOB に対する LOGGING 設定は、ファイル・システムのデータ・ジャーナリングに似ています。LOGGING と FILESYSTEM_LIKE_LOGGING の両方の設定によって、SecureFile を使用した完全なトランザクション・ファイル・システムが実現します。

> **注意：** LOB セグメントの場合は、NOLOGGING 設定と FILESYSTEM_LIKE_LOGGING 設定を使用すると、バックアップ操作中にディスク上でデータが変更され、読取りの非一貫性が生じる場合があります。この状況を回避するには、LOB 記憶域に LOGGING を設定し、LOB セグメントに対する変更が REDO ログ・ファイルに保存されるようにします。または、データベースを FORCE LOGGING モードに変更し、すべての LOB セグメントに対する変更が REDO ログ・ファイルに保存されるようにします。

ロギング属性を指定しているオブジェクトが強制ロギング・モードのデータベースまたは表領域に存在している場合、そのデータベースまたは表領域が強制ロギング・モードから別のモードに変わるまで、NOLOGGING 設定は無視されます。

データベースを ARCHIVELOG モードで運用する場合、LOGGING 操作の前に取ったバックアップからのメディア・リカバリによって、オブジェクトが再作成されます。ただし、NOLOGGING 操作の前に取ったバックアップからのメディア・リカバリでは、オブジェクトは再作成されません。

NOLOGGING モードでの操作で生成される REDO ログのサイズは、LOGGING モードで生成されるログより非常に小さくなります。

NOLOGGING モードでは、データの変更時に、（新しいエクステントを INVALID としてマーク設定し、ディクショナリの変更を記録するために）最小限のログが記録されます。メディア・リカバリ中に NOLOGGING が適用された場合、REDO データのログ記録が中断されるため、エクステント無効化レコードでは、一定のブロック範囲に「論理的に無効」というマークが付きます。このため、損失してはならないデータベース・オブジェクトの場合は、NOLOGGING 操作の後にバックアップを取る必要があります。

NOLOGGING は、LOGGING をサポートする場所のサブセットのみでサポートされます。次の操作でのみ、NOLOGGING モードがサポートされます。

DML:

- INSERT 文または MERGE 文のいずれかの結果として実行されるダイレクト・パス・インサート（シリアルまたはパラレル）。NOLOGGING は、MERGE 文の結果として実行される UPDATE 操作には適用できません。
- ダイレクト・ローダー（SQL*Loader）

DDL:

- CREATE TABLE ...AS SELECT
- CREATE TABLE ...*LOB_storage_clause* ...*LOB_parameters* ...NOCACHE | CACHE READS
- ALTER TABLE ...*LOB_storage_clause* ...*LOB_parameters* ...NOCACHE | CACHE READS（新しく作成した LOB 列のロギングを指定する場合）
- ALTER TABLE ... *modify_LOB_storage_clause* ... *modify_LOB_parameters* ...NOCACHE | CACHE READS（既存の LOB 列のロギングを変更する場合）
- ALTER TABLE ...MOVE
- ALTER TABLE ...（データ移動を伴うすべてのパーティション操作）
  - ALTER TABLE ...ADD PARTITION（ハッシュ・パーティションのみ）
  - ALTER TABLE ...MERGE PARTITIONS
  - ALTER TABLE ...SPLIT PARTITION
  - ALTER TABLE ...MOVE PARTITION
  - ALTER TABLE ...MODIFY PARTITION ...ADD SUBPARTITION
  - ALTER TABLE ...MODIFY PARTITION ...COALESCE SUBPARTITION
- CREATE INDEX
- ALTER INDEX ...REBUILD
- ALTER INDEX ...REBUILD [SUB] PARTITION
- ALTER INDEX ...SPLIT PARTITION

LOB 以外のオブジェクトの場合、この句を指定しないと、オブジェクトが存在する表領域のロギング属性がオブジェクトのデフォルトのロギング属性になります。

LOB に対してこの句を省略した場合は、次のようになります。

- CACHE を指定した場合は、LOGGING が使用されます（CACHE NOLOGGING は指定できないため）。
- NOCACHE または CACHE READS を指定した場合は、表が存在する表領域の属性がデフォルトのロギング属性として使用されます。

NOLOGGING は、内部に（行データとともに表に）格納された LOB には適用されません。LOB に対する NOLOGGING を 4000 バイト未満の値に指定し、STORAGE IN ROW を使用禁止にしていなかった場合、NOLOGGING の指定は無視され、LOB データは他の表データと同様に扱われます。

parallel_clause

用途

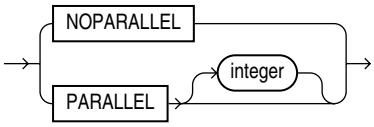
parallel_clause を使用すると、データベース・オブジェクト作成のパラレル化、およびその後のオブジェクトに対する問合せのデフォルトの並列度と、オブジェクトに対する DML 操作の設定が可能になります。

parallel_clause は次の文に指定できます。

- CREATE TABLE: 表のパラレル化を設定する場合（16-6 ページの「CREATE TABLE」を参照）
- ALTER TABLE（12-2 ページの「ALTER TABLE」を参照）
  - 表のパラレル化を変更する場合
  - 表パーティションの追加、結合、交換、マージ、分割、切捨て、削除または移動の操作をパラレル化する場合
- CREATE CLUSTER および ALTER CLUSTER: クラスタのパラレル化を設定または変更する場合（14-2 ページの「CREATE CLUSTER」および 10-5 ページの「ALTER CLUSTER」を参照）
- CREATE INDEX: 索引のパラレル化を設定する場合（14-50 ページの「CREATE INDEX」を参照）
- ALTER INDEX（10-64 ページの「ALTER INDEX」を参照）
  - 索引のパラレル化を変更する場合
  - 索引の再構築または索引パーティションの分割をパラレル化する場合
- CREATE MATERIALIZED VIEW: マテリアライズド・ビューのパラレル化を設定する場合（15-4 ページの「CREATE MATERIALIZED VIEW」を参照）
- ALTER MATERIALIZED VIEW（11-2 ページの「ALTER MATERIALIZED VIEW」を参照）
  - マテリアライズド・ビューのパラレル化を変更する場合
  - マテリアライズド・ビュー・パーティションの追加、結合、交換、マージ、分割、切捨て、削除または移動の操作をパラレル化する場合
  - マテリアライズド・ビュー・サブパーティションの追加または移動の操作をパラレル化する場合
- CREATE MATERIALIZED VIEW LOG: マテリアライズド・ビュー・ログのパラレル化を設定する場合（15-26 ページの「CREATE MATERIALIZED VIEW LOG」を参照）
- ALTER MATERIALIZED VIEW LOG（11-16 ページの「ALTER MATERIALIZED VIEW LOG」を参照）
  - マテリアライズド・ビュー・ログのパラレル化を変更する場合
  - マテリアライズド・ビュー・ログ・パーティションの追加、結合、交換、マージ、分割、切捨て、削除または移動の操作をパラレル化する場合
- ALTER DATABASE ... RECOVER: データベースをリカバリする場合（10-9 ページの「ALTER DATABASE」を参照）
- ALTER DATABASE ... *standby_database_clauses*: スタンバイ・データベースに対する操作をパラレル化する場合（10-9 ページの「ALTER DATABASE」を参照）

構文

***parallel_clause*::=**



セマンティクス

この項では、*parallel_clause* のセマンティクスについて説明します。詳細は、特定のデータベース・オブジェクトまたは操作に対してパラレル化を設定または再設定する SQL 文の説明を参照してください。

注意： *parallel_clause* 構文は、以前のリリースの構文にかわるものです。以前のリリースの構文は下位互換用にサポートされていますが、動作がわずかに異なることがあります。

NOPARALLEL NOPARALLEL を指定すると、シリアル実行が行われます。これはデフォルトです。

PARALLEL 並列度を選択する場合は、PARALLEL を指定します。並列度とは、すべての関係するインスタンスで使用可能な CPU の数に、初期化パラメータ PARALLEL_THREADS_PER_CPU の値を掛けたものです。

PARALLEL *integer* *integer* には、パラレル操作で使用するパラレル・スレッド数である**並列度**を指定します。各パラレル・スレッドは、1、2 個のパラレル実行サーバーを使用します。通常、最適な並列度が計算されるため、*integer* に値を指定する必要はありません。

***parallel_clause* の注意事項：** *parallel_clause* には、次の注意事項があります。

- トリガーまたは参照整合性制約を定義した表に対する DML 操作では、パラレル化を使用できません。
- 索引構成表での UPDATE または DELETE 操作に対して、パラレル化はサポートされません。
- 表の作成中に *parallel_clause* を指定する際に、表に LOB 型またはユーザー定義オブジェクト型の列が含まれている場合、この LOB 型またはオブジェクト型の列を変更する後続の INSERT、UPDATE、DELETE および MERGE 操作は、通知なしにシリアル実行されます。ただし、後続の問合せはパラレルで実行されます。
- *parallel_clause* の効果はパラレル・ヒントによって上書きされます。
- リモート・オブジェクトを参照する DML 文および CREATE TABLE ... AS SELECT 文は、パラレルで実行されます。ただし、リモート・オブジェクトはリモート・データベースにある必要があります。参照は、ローカル・データベースにあるオブジェクトにループバックできません。たとえば、ローカル・データベースのオブジェクトを指定するリモート・データベースのシノニムから参照することはできません。
- LOB 列を含む表での DML 操作はパラレル化できます。ただし、パーティション内並列性はサポートされていません。

参照： パラレル化操作の詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。また 16-61 ページの「PARALLEL の例 :」も参照してください。

physical_attributes_clause

用途

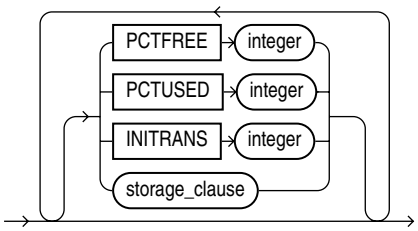
physical_attributes_clause を使用すると、PCTFREE、PCTUSED および INITRANS パラメータの値、および表、クラスタ、索引またはマテリアライズド・ビューの記憶特性を指定できます。

physical_attributes_clause は次の文に指定できます。

- CREATE CLUSTER および ALTER CLUSTER: クラスタおよびクラスタ内のすべての表の物理属性を設定または変更する場合（14-2 ページの「CREATE CLUSTER」および 10-5 ページの「ALTER CLUSTER」を参照）
- CREATE TABLE: 表、表パーティション、オブジェクト表の OIDINDEX または索引構成表のオーバーフロー・セグメントの物理属性を設定する場合（16-6 ページの「CREATE TABLE」を参照）
- ALTER TABLE: 表の物理属性、将来追加される表パーティションのデフォルトの物理属性、または既存の表パーティションの物理属性を変更する場合（12-2 ページの「ALTER TABLE」を参照）。次の制限があります。
  - 物理属性は、一時表には指定できません。
  - 物理属性は、クラスタ化表には指定できません。クラスタ内の表はクラスタの物理属性を継承します。
- CREATE INDEX: 索引または索引パーティションの物理属性を設定する場合（14-50 ページの「CREATE INDEX」を参照）
- ALTER INDEX: 索引の物理属性、将来追加される索引パーティションのデフォルトの物理属性、または既存の索引パーティションの物理属性を変更する場合（10-64 ページの「ALTER INDEX」を参照）
- CREATE MATERIALIZED VIEW: マテリアライズド・ビュー、そのパーティションの 1 つ、またはマテリアライズド・ビューを保持するために生成される索引の物理属性を設定する場合（15-4 ページの「CREATE MATERIALIZED VIEW」を参照）
- ALTER MATERIALIZED VIEW: マテリアライズド・ビューの物理属性、将来追加されるパーティションのデフォルトの物理属性、および既存のパーティション、またはマテリアライズド・ビューを保持するために作成される索引の物理属性を変更する場合（11-2 ページの「ALTER MATERIALIZED VIEW」を参照）
- CREATE MATERIALIZED VIEW LOG および ALTER MATERIALIZED VIEW LOG: マテリアライズド・ビュー・ログの物理属性を設定または変更する場合（15-26 ページの「CREATE MATERIALIZED VIEW LOG」および 11-16 ページの「ALTER MATERIALIZED VIEW LOG」を参照）

構文

***physical_attributes_clause*::=**



（8-44 ページの storage_clause::= を参照）

セマンティクス

この項では、*physical_attributes_clause* のパラメータについて説明します。詳細は、特定のデータベース・オブジェクトに対してこれらのパラメータを設定または再設定する SQL 文の説明を参照してください。

PCTFREE *integer*

データベース・オブジェクトの各データ・ブロック内で、オブジェクトの行を将来更新するために確保しておく領域の割合を表す整数値を指定します。PCTFREE の値は、0 ～ 99 の値にする必要があります。値に 0 を指定した場合は、ブロック全体が一杯になるまで新しい行を挿入できます。デフォルト値は 10 です。10 を指定した場合、既存の行に対して行われる更新用に各ブロックの 10% が確保されるため、各ブロックでは最大 90% まで表に新しい行を挿入できます。

PCTFREE は、表、パーティション、クラスタ、索引、マテリアライズド・ビューおよびマテリアライズド・ビュー・ログを作成および変更する文の中で同様に機能します。PCTFREE と PCTUSED の組合せによって、新しい行を既存のデータ・ブロックと新しいデータ・ブロックのどちらに挿入するかが決まります。8-40 ページの「PCTFREE と PCTUSED の連携」を参照してください。

PCTFREE 句の制限事項： 索引を変更する場合、このパラメータは、*modify_index_default_attrs* 句および *split_partition_clause* にのみ指定できます。

PCTUSED *integer*

使用済領域のうち、データベース・オブジェクトのデータ・ブロックごとに確保される最小限の割合を表す整数値を指定します。PCTUSED は 0 ～ 99 までの正の整数で指定し、デフォルト値は 40 です。

PCTUSED は、表、パーティション、クラスタ、マテリアライズド・ビューおよびマテリアライズド・ビュー・ログを作成および変更する文の中で同様に機能します。

PCTUSED は、索引構成表には無効な表記憶特性です。

PCTFREE および PCTUSED の合計は 100 以下である必要があります。PCTFREE と PCTUSED をともに使用して、データベース・オブジェクト内の領域を効果的に利用できます。8-40 ページの「PCTFREE と PCTUSED の連携」を参照してください。

PCTUSED 句の制限事項： PCTUSED パラメータには、次の制限事項があります。

- このパラメータは、索引または索引構成表の索引セグメントには指定できません。
- このパラメータは、自動セグメント領域管理のオブジェクトに対しては有効ではなく、無視されます。

> **参照：** PCTUSED および PCTFREE の各値によるパフォーマンスへの効果については『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。自動セグメント領域管理については 16-79 ページの「CREATE TABLESPACE」の「segment_management_clause」を参照してください。

PCTFREE と PCTUSED の連携

新しく割り当てられたデータ・ブロックでは、挿入に使用できる領域は、ブロック・サイズからブロック・オーバーヘッドと空き領域（PCTFREE）の合計を引いたものになります。既存のデータを更新する場合は、ブロック内の任意の利用可能な領域を使用できます。このため、更新によってブロックの利用可能な領域が PCTFREE より小さくなる可能性があります。

データ・ブロックが PCTFREE によって決定された制限に達すると、そのブロックの割合（パーセント）がパラメータ PCTUSED を下回るまで、ブロックに新しい行は挿入はできないと Oracle Database によって判断されます。この値になるまで、Oracle Database では、データ・ブロックにすでに含まれている行の更新にのみデータ・ブロックの空き領域が使用されます。ブロックは、使用済領域が PCTUSED の値を下回ると、行挿入の対象となります。

> **参照：** PCTUSED および PCTFREE による空きリストのセグメント領域管理処理の詳細は、8-47 ページの「FREELISTS」を参照してください。

INITRANS *integer*

データベース・オブジェクトに割り当てられた各データ・ブロックに割り当てられる、同時実行トランザクション・エントリの初期数を指定します。この値の範囲は 1 ～ 255 で、デフォルト値は 1 ですが、次の例外があります。

- クラスタの INITRANS のデフォルト値は、クラスタが存在する表領域の INITRANS のデフォルト値と 2 のいずれか大きい方の値です。
- 索引のデフォルト値は 2 です。

通常、INITRANS 値は、変更せずにデフォルトのまま使用してください。

ブロックを更新するトランザクションごとに、ブロックのトランザクション・エントリが必要です。このパラメータを指定した場合、最小数の同時実行トランザクションでブロックを更新することができます。さらに、トランザクション・エントリを動的に割り当てるときのオーバーヘッドを回避できます。

INITRANS パラメータは、表、パーティション、クラスタ、索引、マテリアライズド・ビューおよびマテリアライズド・ビュー・ログを作成および変更する文の中で同様に機能します。

MAXTRANS パラメータ

MAXTRANS は、セグメント内の各データ・ブロックで実行可能な同時実行更新トランザクションの最大数を決定する以前のリリースのパラメータです。このパラメータは現在非推奨になっています。今回のリリースでは、Oracle は、ブロック内の未使用領域に応じて、任意のデータ・ブロックに対して最大 255 の同時実行更新トランザクションを自動的に許可します。

MAXTRANS 値が設定された既存のオブジェクトは、その設定を保持します。ただし、MAXTRANS 値を変更しようとすると、新しい指定は無視され、エラーを戻さずに値は 255 に置き換えられます。

storage_clause

storage_clause によって、表、オブジェクト表 OIDINDEX、パーティション、LOB データ・セグメントまたは索引構成表のオーバーフロー・データ・セグメントの記憶特性を指定できます。この句は、大規模な表のパフォーマンスに影響します。記憶域は、追加領域の動的割当てを最小限に抑えるように割り当てられます。詳細は、8-43 ページの「storage_clause」を参照してください。

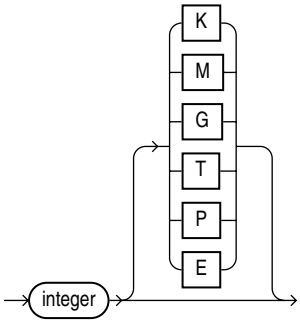
size_clause

用途

size_clause を使用すると、すべての文にバイト、KB（K）、MB（M）、GB（G）、TB（T）、PB（P）、EB（E）の各単位でバイト数を指定して、ディスク容量またはメモリー容量を設定できます。

構文

***size_clause*::=**



セマンティクス

size_clause を使用すると、バイト数を様々な単位で指定できます。単位の略語を指定しない場合、*integer* はバイト単位で解釈されます。

注意： バイトのすべての単位が常に適切であるとはかぎりません。状況依存の制限が適用されます。後者の場合、Oracle はエラー・メッセージを発行します。

storage_clause

用途

storage_clause を使用すると、Oracle Database による永続的なデータベース・オブジェクトの格納方法を指定できます。一時セグメントの記憶域パラメータでは、常に、関連付けられた表領域のデフォルトの記憶域パラメータが使用されます。記憶域パラメータは、データベースのデータへのアクセス時間およびデータベース内での領域の効率的な利用に影響します。

> **参照：** 記憶域パラメータの影響の詳細は、『Oracle Database ストレージ管理者ガイド』を参照してください。

クラスタ、索引、マテリアライズド・ビュー、マテリアライズド・ビュー・ログ、ロールバック・セグメント、表、LOB、VARRAY、ネストした表またはパーティションを作成する場合、これらのオブジェクトに割り当てられるセグメントに、記憶域パラメータの値を指定できます。記憶域パラメータを指定しない場合、オブジェクトが存在する表領域に指定されている記憶域パラメータの値が使用されます。表領域に対して値を指定しなかった場合、データベースによってデフォルト値が使用されます。

注意： ローカル管理表領域内のオブジェクトへの記憶域パラメータの指定は、下位互換性のためにサポートされています。ローカル管理表領域を使用している場合、ローカル管理表領域にオブジェクトを作成する際にこれらの記憶域パラメータを省略できます。

クラスタ、索引、マテリアライズド・ビュー、マテリアライズド・ビュー・ログ、ロールバック・セグメント、表、VARRAY、ネストした表またはパーティションを変更する場合、記憶域パラメータの値を変更できます。この値の変更は、それ以降のエクステントの割当てにのみ影響します。

storage_clause は *physical_attributes_clause* の一部であるため、この句は、物理属性句を指定できる任意の文に指定できます（8-39 ページの「physical_attributes_clause」を参照）。*storage_clause* は次の文にも指定できます。

- CREATE CLUSTER および ALTER CLUSTER: **クラスタおよびクラスタ内のすべての表**の記憶特性を設定または変更する場合（14-2 ページの「CREATE CLUSTER」および 10-5 ページの「ALTER CLUSTER」を参照）
- CREATE INDEX および ALTER INDEX: **表索引または索引パーティション**に対して作成された索引セグメントまたは**主キー制約または一意制約を適用するために使用される索引に対して作成された索引セグメント**の記憶特性を設定または変更する場合（14-50 ページの「CREATE INDEX」および 10-64 ページの「ALTER INDEX」を参照）
- CREATE TABLE または ALTER TABLE の ENABLE ... USING INDEX 句: **主キー制約または一意制約を適用するためにシステムによって作成された索引**の記憶特性を設定または変更する場合
- CREATE MATERIALIZED VIEW および ALTER MATERIALIZED VIEW: **マテリアライズド・ビュー、そのパーティションの 1 つ、またはマテリアライズド・ビューを保持するために生成される索引**の記憶特性を設定または変更する場合（15-4 ページの「CREATE MATERIALIZED VIEW」および 11-2 ページの「ALTER MATERIALIZED VIEW」を参照）
- CREATE MATERIALIZED VIEW LOG および ALTER MATERIALIZED VIEW LOG: **マテリアライズド・ビュー・ログ**の記憶特性を設定または変更する場合（15-26 ページの「CREATE MATERIALIZED VIEW LOG」および 11-16 ページの「ALTER MATERIALIZED VIEW LOG」を参照）
- CREATE ROLLBACK SEGMENT および ALTER ROLLBACK SEGMENT: **ロールバック・セグメント**の記憶特性を設定または変更する場合（15-59 ページの「CREATE ROLLBACK SEGMENT」および 11-36 ページの「ALTER ROLLBACK SEGMENT」を参照）

- CREATE TABLE および ALTER TABLE: クラスタ化されていない表またはそのパーティションまたはサブパーティションの 1 つの **LOB または VARRAY データ・セグメント**、または**ネストした表の記憶表**の記憶特性を設定する場合（16-6 ページの「CREATE TABLE」および 12-2 ページの「ALTER TABLE」を参照）
- CREATE TABLESPACE および ALTER TABLESPACE: **表領域に作成されたすべてのオブジェクト**に対するデフォルトの記憶特性を設定または変更する場合（16-71 ページの「CREATE TABLESPACE」および 12-82 ページの「ALTER TABLESPACE」を参照）

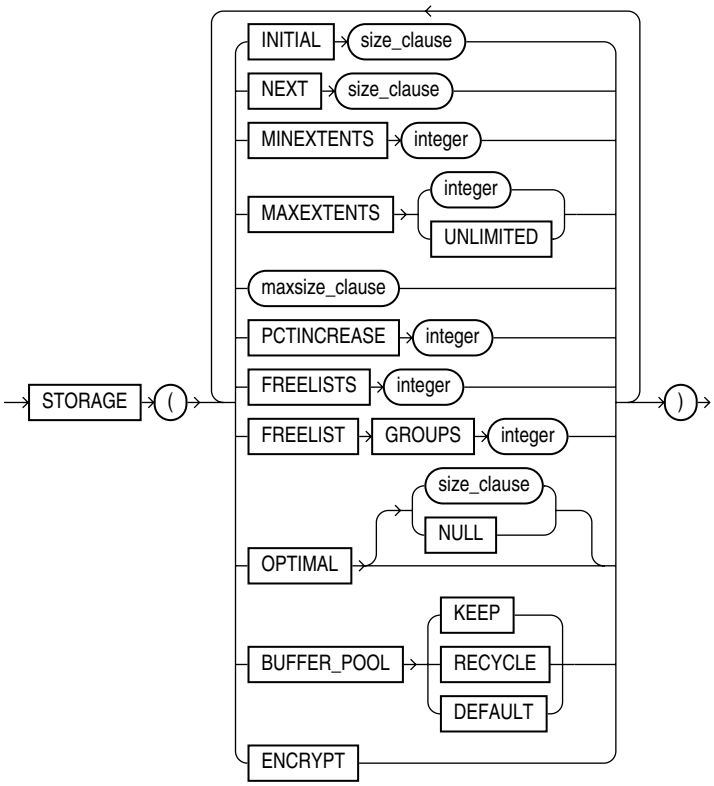
  表領域の記憶域パラメータに対する変更は、表領域に作成される新しいオブジェクトまたはセグメントに割り当てられる新しいエクステントにのみ影響します。
- *constraint*: **制約の適用に使用される索引（パーティション索引の場合は索引のパーティション）**の記憶域を指定する場合（8-4 ページの「constraint」を参照）

前提条件

STORAGE パラメータの値を変更する場合は、適切な CREATE 文または ALTER 文を使用するための権限が必要です。

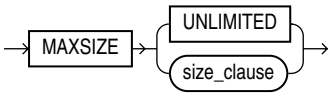
構文

***storage_clause*::=**



（8-42 ページの size_clause::= を参照）

***maxsize_clause*::=**



（8-42 ページの size_clause::= を参照）

セマンティクス

この項では、*storage_clause* のパラメータについて説明します。詳細は、特定のデータベース・オブジェクトに対してこれらの記憶域パラメータを設定または再設定する SQL 文の説明を参照してください。

注意： ローカル管理表領域の場合、*storage_clause* の解析方法は異なります。ローカル管理表領域の場合、Oracle Database では、オブジェクトの最初の作成時に割り当てられるエクステント数を計算するために、INITIAL、NEXT、PCTINCREASE および MINEXTENTS が使用されます。オブジェクトの作成後、これらのパラメータは無視されます。詳細は、16-71 ページの「CREATE TABLESPACE」を参照してください。

参照： 「表の記憶域属性の指定例」（8-49 ページ）

INITIAL

オブジェクトの第 1 エクステントのサイズを指定します。スキーマ・オブジェクトの作成時に、このエクステントに領域が割り当てられます。この句の詳細は、8-42 ページの「size_clause」を参照してください。

ローカル管理表領域では、INITIAL の値は、MINEXTENTS、NEXT および PCTINCREASE の値とともに、セグメントの初期サイズを決定するために使用されます。自動割当てエクステント管理の場合、データベースによって、初期セグメント・サイズを構成するエクステントのサイズが決定されます。均一エクステント管理の場合、エクステント数は、初期セグメント・サイズと表領域作成時に指定された均一エクステント・サイズで決定されます。たとえば、エクステントが 1M の均一なローカル管理表領域の場合、INITIAL の値を 5M に指定すると、1M のエクステントが 5 つ作成されます。

ディクショナリ管理表領域では、デフォルトの初期エクステント・サイズは 5 ブロックであり、後続のすべてのエクステントは 5 ブロックに丸められます。表領域作成時に MINIMUM EXTENT が指定された場合、エクステント・サイズは MINIMUM EXTENT の値に丸められます。

INITIAL の制限事項： ALTER 文では、INITIAL を指定できません。

NEXT

オブジェクトに割り当てる次のエクステント・サイズをバイト単位で指定します。この句の詳細は、8-42 ページの「size_clause」を参照してください。

ローカル管理表領域では、表領域が自動割当てエクステント管理用に設定されている場合、NEXT のサイズは Oracle によって決定されます。UNIFORM 表領域では、NEXT のサイズは表領域作成時に指定された均一エクステント・サイズです。

ディクショナリ管理表領域では、デフォルトのサイズは 5 データ・ブロックです。最小のサイズは 1 データ・ブロックです。最大値は、ご使用のオペレーティング・システムによって異なります。5 データ・ブロックより小さい値が指定された場合、データ・ブロック・サイズの一番近い倍数に丸められます。5 データ・ブロックを超える値の場合は、断片化を最小限に抑える値に切り上げられます。

参照： 断片化の最小化の詳細は、『Oracle Database 管理者ガイド』を参照してください。

PCTINCREASE

ローカル管理表領域では、初期セグメント・サイズを決定するためにセグメント作成時にPCTINCREASE の値が使用され、その後の領域割当て時にはこのパラメータは無視されます。

ディクショナリ管理表領域では、3 番目以降の各エクステントが直前のエクステントに対して増加する割合（パーセント）を指定します。デフォルト値は 50 です。この場合、3 番目以降のエクステントは、それぞれその直前のエクステントより 50% ずつ大きくなります。最小値は 0 で、この場合、第 2 エクステント以降のエクステントのサイズはすべて同じになります。最大値は、ご使用のオペレーティング・システムによって異なります。計算された各エクステントのサイズは、データ・ブロック・サイズの一番近い倍数に丸められます。ALTER 文に PCTINCREASE パラメータの値を指定してこの値を変更すると、この変更した値と直前に割り当てられたエクステントのサイズを使用して、次に割り当てるエクステントのサイズが計算されます。

> **提案：** すべてのエクステントを同じサイズにする場合は、PCTINCREASE の設定を 0（ゼロ）にして、SMON バックグラウンド・プロセスによるエクステントの結合を回避します。通常、設定は 0（ゼロ）にすることをお薦めします。そうすることで、断片化を最小限に抑え、処理中に一時セグメントの極端な拡大化を防ぐことができます。

PCTINCREASE の制限事項： ロールバック・セグメントに PCTINCREASE は指定できません。ロールバック・セグメントでは、PCTINCREASE の値は常に 0（ゼロ）です。

MINEXTENTS

ローカル管理表領域では、初期セグメント・サイズを決定するために、MINEXTENTS の値が PCTINCREASE、INITIAL および NEXT の値とともに使用されます。

ディクショナリ管理表領域では、オブジェクトの作成時に割り当てられる合計エクステント数を指定します。最小値（デフォルト）は 1 です。この場合、第 1 エクステントのみが割り当てられます。ただし、ロールバック・セグメントの場合、最小値（デフォルト）は 2 です。最大値は、ご使用のオペレーティング・システムによって異なります。

- ローカル管理表領域では、領域の初期割当て量を計算するために MINEXTENTS が使用されます。INITIAL × MINEXTENTS で算出されます。その後、この値は 1 に設定され、DBA_SEGMENTS ビューに反映されます。
- ディクショナリ管理表領域では、MINEXTENTS は単純に、セグメントに割り当てる必要があるエクステントの最小数です。

MINEXTENTS の値が 1 より大きい場合、INITIAL、NEXT および PCTINCREASE 記憶域パラメータの値に基づいて、次のエクステントのサイズが計算されます。

ALTER 文に MINEXTENTS の値を指定して変更する際は、現行の値より小さくすることはできますが、大きくすることはできません。MINEXTENTS をより小さい値に再設定すると便利な場合があります。たとえば、TRUNCATE ... DROP STORAGE 文の前で、TRUNCATE 操作の後もセグメントがエクステントの最小数を変更しない場合です。

MINEXTENTS の制限事項： MINEXTENTS 記憶域パラメータには、次の制限事項があります。

- MINEXTENTS は、表領域レベルでは適用できません。
- ALTER 文の MINEXTENTS の値、またはローカル管理表領域に存在するオブジェクトに対する MINEXTENTS の値は変更できません。

MAXEXTENTS

この記憶域パラメータは、ディクショナリ管理表領域のオブジェクトに対してのみ有効です。第 1 エクステントを含めて、Oracle がオブジェクトに割り当てることができるエクステントの総数を指定します。最小値は 1 です（最小値が常に 2 のロールバック・セグメントは除きます）。デフォルト値は、データ・ブロックのサイズによって異なります。

MAXEXTENTS の制限事項： MAXEXTENTS は、ローカル管理表領域に存在するオブジェクトに対しては無視されます。

UNLIMITED 必要に応じてエクステントが自動的に割り当てられるようにする場合に、UNLIMITED を指定します。断片化を最小限に抑えるため、この設定をお薦めします。

ロールバック・セグメントにこの句は使用しないでください。この句をロールバック・セグメントに使用すると、長時間実行される特殊な DML トランザクションによって、ディスクが一杯になるまで新しいエクステントが作成され続ける可能性があります。

注意： *storage_clause* を指定せずに作成したロールバック・セグメントは、ロールバック・セグメントを作成した表領域の記憶域パラメータと同じ設定になります。MAXEXTENTS UNLIMITED を指定して表領域を作成する場合、ロールバック・セグメントのデフォルトは同じ設定になります。

MAXSIZE

MAXSIZE 句を使用すると、記憶域要素の最大サイズを指定できます。LOB 記憶域では、MAXSIZE を使用すると、次のようになります。

- *LOB_parameters* に RETENTION MAX を指定すると、指定されたサイズに LOB セグメントが拡張され、その後、UNDO 領域から任意の領域を再利用できます。
- *LOB_parameters* に RETENTION AUTO、MIN または NONE を指定すると、指定されたサイズは LOB セグメントのサイズの上限値となり、UNDO の保持においてその指定の影響はありません。

UNLIMITED 記憶域要素のディスク領域を制限しない場合は、UNLIMITED 句を使用します。この句は、*LOB_parameters* での RETENTION MAX の指定と同時には指定できません。両方を指定すると、RETENTION AUTO および MAXSIZE UNLIMITED が使用されます。

FREELISTS

セグメントを手動で領域管理する表領域の場合、セグメント内の挿入ポイント数を増やして OLTP システムの領域管理のパフォーマンスを向上させるために、FREELISTS 記憶域パラメータが使用されます。セグメントを自動的に領域管理する表領域の場合、変化するワークロードにデータベースが適応するため、このパラメータは無視されます。

セグメントを手動で領域管理する表領域の場合、表領域およびロールバック・セグメント以外のオブジェクトに対して、表、パーティション、クラスタまたは索引の各空きリスト・グループの空きリスト数を指定します。このパラメータの最小値（デフォルト）は 1 です。各空きリスト・グループには、空きリストが 1 つ割り当てられます。最大値は、データ・ブロックのサイズによって異なります。FREELISTS に指定した値が大きすぎた場合、最大値を示すエラーが戻ります。

16-38 ページの「LOB_parameters」の SECUREFILE パラメータを指定している場合、この句は無効です。SECUREFILE パラメータおよび FREELISTS の両方を指定した場合、FREELISTS の指定は無視されます。

FREELISTS の制限事項： 表領域またはロールバック・セグメントを作成または変更するとき以外は、すべての文の *storage_clause* に FREELISTS を指定できます。

FREELIST GROUPS

セグメントを手動で領域管理する表領域の場合、Oracle Real Application Clusters 環境でセグメント空き領域を静的にパーティション化するために、この記憶域パラメータの値が使用されます。このパーティション化によって、セグメント・メタデータがインスタンス間で転送されなくなり、領域の割当ておよび割当て解除のパフォーマンスが向上します。セグメントを自動的に領域管理する表領域の場合、インスタンス間ワークロードに Oracle が動的に適応するため、このパラメータは無視されます。

セグメントを手動で領域管理する表領域の場合、作成するデータベース・オブジェクトに対する空きリスト・グループ数を指定します。このパラメータの最小値（デフォルト）は 1 です。Oracle は、Oracle Real Application Clusters（RAC）インスタンスのインスタンス番号を使用して、各インスタンスを空きリスト・グループにマップします。

1 つの空きリスト・グループに、それぞれデータベース・ブロックを 1 つずつ使用します。したがって、次のことがいえます。

- 各空きリスト・グループの最小値に、1 データ・ブロックを加えた値を格納できるだけの十分な大きさの値を INITIAL の値に指定していない場合、INITIAL の値は必要な分だけ引き上げられます。
- 均一なローカル管理表領域にオブジェクトを作成する場合、空きリスト・グループ数に適応するだけの十分なエクステント・サイズがないと、作成操作は正常に実行されません。

16-38 ページの「LOB_parameters」の SECUREFILE パラメータを指定している場合、この句は無効です。SECUREFILE パラメータおよび FREELIST GROUPS の両方を指定した場合、FREELIST GROUPS の指定は無視されます。

FREELIST GROUPS の制限事項： FREELIST GROUPS パラメータは、CREATE TABLE、CREATE CLUSTER、CREATE MATERIALIZED VIEW、CREATE MATERIALIZED VIEW LOG および CREATE INDEX 文でのみ指定できます。

BUFFER_POOL

BUFFER_POOL 句では、スキーマ・オブジェクト用のデフォルトのバッファ・プール（キャッシュ）を定義します。オブジェクトのすべてのブロックは、指定されたキャッシュに格納されます。

- バッファ・プールがパーティション表またはパーティション索引用に定義されている場合は、パーティションは、パーティション・レベル定義で変更されないかぎり、表定義または索引定義のバッファ・プールを継承します。
- 索引構成表の場合、索引セグメントおよびオーバーフロー・セグメントに対して個々にバッファ・プールを指定できます。

BUFFER_POOL の制限事項： BUFFER_POOL には、次の制限事項があります。

- この句は、クラスタ化表には指定できません。ただし、クラスタには指定できます。
- この句は、表領域またはロールバック・セグメントには指定できません。

KEEP KEEP を指定すると、ブロックがセグメントから KEEP バッファ・プールへ移されます。KEEP バッファ・プールに適切なサイズを維持すると、メモリーのスキーマ・オブジェクトが保持され、I/O 操作を避けることができます。KEEP は、表、クラスタ、マテリアライズド・ビューまたはマテリアライズド・ビュー・ログに指定する NOCACHE 句より優先されます。

RECYCLE RECYCLE を指定すると、ブロックがセグメントから RECYCLE プールへ移されます。RECYCLE プールに適切なサイズを指定すると、不要なキャッシュ領域が利用されず、デフォルト・プールが RECYCLE プールであるオブジェクト数が削減されます。

DEFAULT デフォルトのバッファ・プールを識別する場合に、DEFAULT を指定します。これは、KEEP も RECYCLE も指定しないオブジェクトのデフォルトです。

> **参照：** 複数のバッファ・プールの使用方法については、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。

OPTIMAL

OPTIMAL キーワードは、ロールバック・セグメントのみに指定します。ロールバック・セグメントの最適なサイズをバイト単位で指定します。この句の詳細は、8-42 ページの「size_clause」を参照してください。

エクステントのデータがアクティブ・トランザクションで不要になった場合、Oracle は、そのエクステントの割当てを動的に解除することによって、指定されたロールバック・セグメントのサイズを維持します。ロールバック・セグメントのサイズの合計を OPTIMAL 値より小さくせずに、できるだけ多くのエクステントの割当てを解除します。

OPTIMAL には、MINEXTENTS、INITIAL、NEXT および PCTINCREASE パラメータで最初に割り当てた領域より小さい値を指定できません。最大値は、ご使用のオペレーティング・システムによって異なります。値は、データ・ブロック・サイズの一番近い倍数に丸められます。

NULL ロールバック・セグメントに対する最適なサイズがないことを示す場合に NULL を指定します。これは、ロールバック・セグメントのエクステントの割当てが解除されないことを示します。これはデフォルトの動作です。

ENCRYPT

この句は、表領域を作成する場合にのみ有効です。表領域全体を暗号化するには、ENCRYPT を指定します。CREATE TABLESPACE 文に ENCRYPTION 句を指定する必要もあります。

> **参照：** 16-77 ページの「CREATE TABLESPACE」の「ENCRYPTION 句」を参照してください。

例

表の記憶域属性の指定例 次の文は、表の作成時に記憶域パラメータの値を指定します。

```
CREATE TABLE divisions
    (div_no     NUMBER(2),
     div_name   VARCHAR2(14),
     location   VARCHAR2(13) )
     STORAGE  ( INITIAL 100K  NEXT     50K
                MINEXTENTS 1  MAXEXTENTS 50  PCTINCREASE 5);
```

STORAGE パラメータに指定した値に基づいて、次に示すとおり表に領域が割り当てられます。

- MINEXTENTS の値は 1 のため、表作成時にエクステントが 1 つ割り当てられます。
- INITIAL の値は 100KB のため、第 1 エクステントのサイズは 100KB になります。
- 表データが第 1 エクステントを超えた場合、第 2 エクステントが割り当てられます。NEXT の値が 50KB のため、第 2 エクステントのサイズは 50KB になります。
- 表データが増加して最初の 2 つのエクステントを超えた場合、第 3 エクステントが割り当てられます。PCTINCREASE の値は 5 のため、第 3 エクステントの値は第 2 エクステントの 5% 増の 52.5KB になります。このとき、データ・ブロック・サイズが 2KB の場合は、値は丸められて 52KB になります。

  これ以降、表データの増加に応じて、直前に割り当てられたエクステントのサイズより 5% 大きいサイズのエクステントが割り当てられます。
- MAXEXTENTS の値は 50 のため、表に割り当てられるエクステントの最大数は 50 になります。

9

SQL 問合せおよび副問合せ

この章では、SQL 問合せおよび副問合せについて説明します。

この章では、次の内容を説明します。

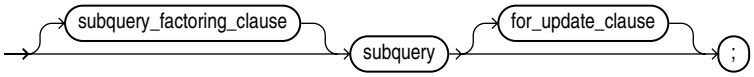
- 問合せおよび副問合せ
- 単純な問合せの作成
- 階層問合せ
- UNION [ALL]、INTERSECT および MINUS 演算子
- 問合せ結果のソート
- 結合
- 副問合せの使用方法
- ネストされた副問合せのネスト解除
- DUAL 表からの選択
- 分散問合せ

問合せおよび副問合せ

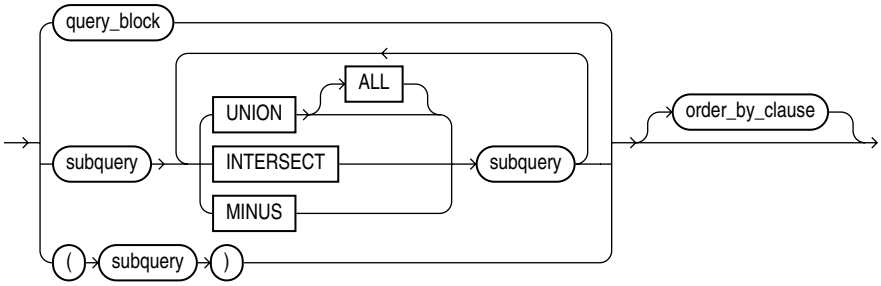
問合せとは、1 つ以上の表またはビューからデータを検索する操作のことです。このマニュアルでは、トップレベルの SELECT 文を**問合せ**といい、他の SQL 文の中でネストされた問合せを**副問合せ**といいます。

この項では、問合せおよび副問合せの種類およびその使用方法について説明します。この章では、トップレベルの構文について説明します。すべての句のすべての構文およびこの文のセマンティクスについては、19-4 ページの「SELECT」を参照してください。

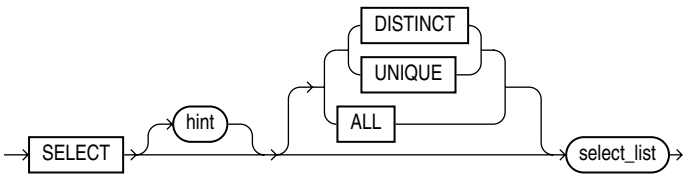
***select*::=**

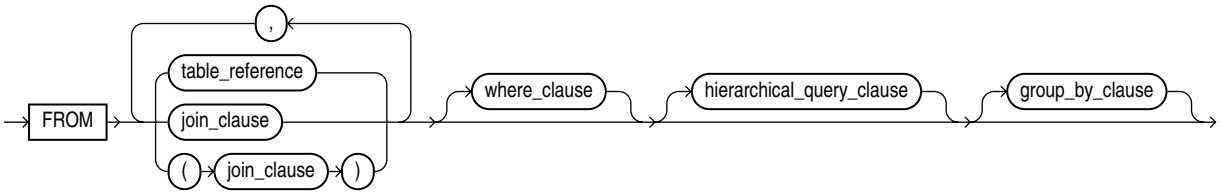


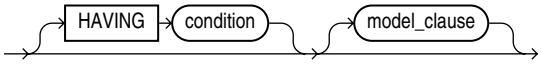
***subquery*::=**



***query_block*::=**







単純な問合せの作成

SELECT キーワードの後、FROM 句の前にある式のリストを、**SELECT 構文のリスト**といいます。SELECT 構文のリストに、1 つ以上の表、ビューおよびマテリアライズド・ビューから Oracle Database が戻す行に含まれる 1 つ以上の列を指定します。SELECT 構文のリストの要素によって、列のデータ型、長さおよび数が決定されます。

複数の表に同じ名前の列がある場合、表の名前でその列名を修飾する必要があります。それ以外の場合は、完全に修飾した列名はオプションとなります。ただし、明示的に表および列の参照を修飾することをお薦めします。表および列名を完全に修飾することで、Oracle の作業が少なくなります。

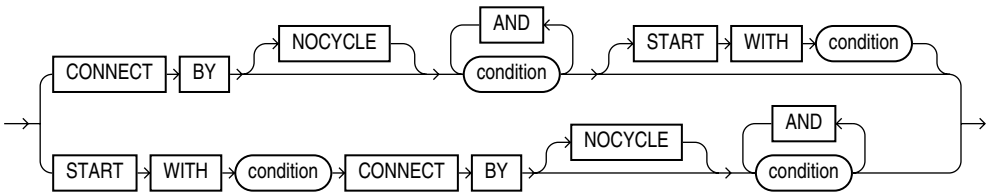
列の別名 *c_alias* を使用して、SELECT 構文のリストの直前の式にラベルを付けると、列が新しい見出し付きで表示されます。別名によって、問合せ中に SELECT 構文のリストの項目名を効果的に変更できます。別名は ORDER BY 句の中で使用できますが、問合せ内のその他の句には使用できません。

Oracle Database のオプティマイザに指示（**ヒント**）を与えるために、SELECT 文中でコメントを使用できます。オプティマイザは、これらのヒントを使用して文の実行計画を選択します。ヒントの詳細は、2-70 ページの「ヒントの使用方法」を参照してください。

階層問合せ

表に階層データが含まれる場合、階層問合せ句を使用して階層順に行を選択することができます。

***hierarchical_query_clause*::=**



START WITH 句では、階層のルート行を指定します。

CONNECT BY 句では、階層の親 / 子の行の関連を指定します。

- NOCYCLE パラメータは、データ内に CONNECT BY ループが存在する場合でも問合せで行を戻すように Oracle Database に指示します。このパラメータを CONNECT_BY_ISCYCLE 疑似列とともに使用すると、ループが含まれている行を確認できます。詳細は、3-2 ページの「CONNECT_BY_ISCYCLE 疑似列」を参照してください。
- 階層問合せでは、*condition* 内の 1 つの式を、親である行を参照するために PRIOR 演算子で修飾する必要があります。次に例を示します。

  ```
  ... PRIOR expr = expr
  or
  ... expr = PRIOR expr
  ```

  CONNECT BY *condition* が複合条件の場合、1 つの条件のみに PRIOR 演算子が必要です（複数の PRIOR 条件を使用することもできます）。たとえば、次のようになります。

  ```
  CONNECT BY last_name != 'King' AND PRIOR employee_id = manager_id ...
  CONNECT BY PRIOR employee_id = manager_id and
             PRIOR account_mgr_id = customer_id ...
  ```

  PRIOR は単項演算子であり、単項算術演算子の + および - と同じ優先順位を持っています。この演算子は、階層問合せ内でカレント行の親である行の直後にある式を評価します。

  PRIOR は、等価演算子を使用して列の値を比較する場合によく使用されます（PRIOR キーワードは演算子のどちら側でもかまいません）。PRIOR を指定すると、列の親である行の値が使用されます。等号（=）以外の演算子は、理論上は CONNECT BY 句に指定できます。ただし、これらの他の演算子の組合せによっては、作成される条件は無限ループを発生させる場合があります。この場合、実行時にループが検出され、エラーが戻されます。

CONNECT BY 条件と PRIOR 式は、いずれも相関関係のない副問合せの形式で指定できます。ただし、CURRVAL および NEXTVAL は、無効な PRIOR 式であるため、PRIOR 式は順序を参照できません。

CONNECT_BY_ROOT 演算子を使用して SELECT 構文のリスト内の列を問い合せることによって、階層問合せをさらに向上できます。この演算子は、親の行のみでなく、階層内のすべての祖先の行を戻すことによって、階層問合せの CONNECT BY [PRIOR] 条件の機能を拡張します。

参照： この演算子の詳細は、4-5 ページの「CONNECT_BY_ROOT」および 9-5 ページの「階層問合せの例」を参照してください。

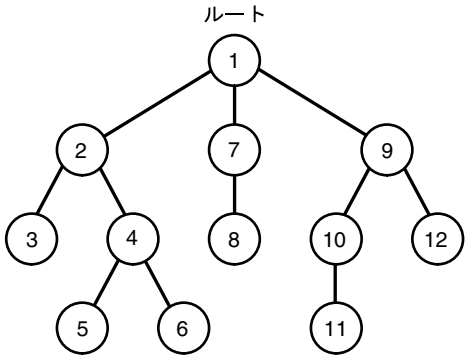
Oracle は次のように階層問合せを処理します。

- 最初に、結合（指定されている場合）が、FROM 句で指定されているか、または WHERE 句述語で指定されているかが評価されます。
- CONNECT BY 条件が評価されます。
- 残りの WHERE 句述語が評価されます。

次に、Oracle はこれらの評価からの情報を使用して、次の手順で階層を形成します。

1. Oracle は、階層のルート行を選択します。これらの行は、START WITH 条件を満たすものです。
2. Oracle は、各ルート行の子である行を選択します。子である各行は、1 つのルート行に関して CONNECT BY 条件を満たす必要があります。
3. Oracle は、子である行の連続生成を選択します。まず、手順 2 で戻された子である行を選択し、その行にある子を選択します（以降同様に続きます）。現在の親である行に関する CONNECT BY 条件を評価することによって、常に子を選択します。
4. 問合せに結合を含まない WHERE 句が含まれる場合、Oracle は、階層から WHERE 句の条件を満たさないすべての行を排除します。条件を満たさない子である行をすべて排除するのではなく、各行に関してこの条件をそれぞれ評価します。
5. Oracle は、図 9-1 に示す順序で行を戻します。この図では、親である行の下に子である行が表示されます。階層ツリーの詳細は、3-3 ページの図 3-1「階層ツリー」を参照してください。

図 9-1 階層問合せ



親である行に対する子を検索するために、Oracle は、親である行の CONNECT BY 条件の PRIOR 式、および各行の他の式を表の中で評価します。条件が TRUE となる行が、その親である行の子です。CONNECT BY 条件に、問合せによって選択された行をさらにフィルタ処理するための他の条件を含めることができます。

CONNECT BY 条件が階層のループになった場合、Oracle はエラーを戻します。1 つの行が別の行の親（または親の親または祖先）および子（または子の子または子孫）の場合、ループが発生します。

注意： 階層問合せでは、ORDER BY または GROUP BY を指定しないでください。指定すると、CONNECT BY の結果の階層順序が上書きされます。同じ親の兄弟である行を順序付ける場合は、ORDER SIBLINGS BY 句を使用します。19-29 ページの「order_by_clause」を参照してください。

階層問合せの例

CONNECT BY 句の例　次の階層問合せは、CONNECT BY を使用して従業員とマネージャの関係を定義しています。

```
SELECT employee_id, last_name, manager_id
   FROM employees
   CONNECT BY PRIOR employee_id = manager_id;

EMPLOYEE_ID LAST_NAME                 MANAGER_ID
----------- ------------------------- ----------
        101 Kochhar                           100
        108 Greenberg                         101
        109 Faviet                            108
        110 Chen                              108
        111 Sciarra                           108
        112 Urman                             108
        113 Popp                              108
        200 Whalen                            101
        203 Mavris                            101
        204 Baer                              101
. . .
```

LEVEL の例　次の例は、前述の例と似ていますが、LEVEL 疑似列を使用して、親および子である行を表示しています。

```
SELECT employee_id, last_name, manager_id, LEVEL
   FROM employees
   CONNECT BY PRIOR employee_id = manager_id;

EMPLOYEE_ID LAST_NAME                 MANAGER_ID      LEVEL
----------- ------------------------- ---------- ----------
        101 Kochhar                           100          1
        108 Greenberg                         101          2
        109 Faviet                            108          3
        110 Chen                              108          3
        111 Sciarra                           108          3
        112 Urman                             108          3
        113 Popp                              108          3
        200 Whalen                            101          2
        203 Mavris                            101          2
        204 Baer                              101          2
        205 Higgins                           101          2
        206 Gietz                             205          3
        102 De Haan                           100          1
...
```

START WITH 句の例　次の例は、START WITH 句を追加して階層にルート行を指定し、SIBLINGS キーワードを使用した ORDER BY 句を追加して階層の順序を保持しています。

```
SELECT last_name, employee_id, manager_id, LEVEL
      FROM employees
      START WITH employee_id = 100
      CONNECT BY PRIOR employee_id = manager_id
      ORDER SIBLINGS BY last_name;

LAST_NAME                 EMPLOYEE_ID MANAGER_ID      LEVEL
------------------------- ----------- ---------- ----------
King                              100                     1
Cambrault                         148        100          2
Bates                             172        148          3
Bloom                             169        148          3
Fox                               170        148          3
```

```
Kumar                          173       148         3
Ozer                           168       148         3
Smith                          171       148         3
De Haan                        102       100         2
Hunold                         103       102         3
Austin                         105       103         4
Ernst                          104       103         4
Lorentz                        107       103         4
Pataballa                      106       103         4
Errazuriz                      147       100         2
Ande                           166       147         3
Banda                          167       147         3
...
```

hr.employees 表で、Steven King は会社の最高責任者であるため、マネージャはいません。彼の従業員には、部門 80 のマネージャである John Russell がいます。employees 表を更新して Russell を King のマネージャとして設定する場合は、データ内にループを作成します。

```
UPDATE employees SET manager_id = 145
   WHERE employee_id = 100;

SELECT last_name "Employee",
   LEVEL, SYS_CONNECT_BY_PATH(last_name, '/') "Path"
   FROM employees
   WHERE level <= 3 AND department_id = 80
   START WITH last_name = 'King'
   CONNECT BY PRIOR employee_id = manager_id AND LEVEL <= 4;

ERROR:
ORA-01436: CONNECT BY loop in user data
```

CONNECT BY 条件に NOCYCLE パラメータを指定すると、Oracle はループでも行を戻します。CONNECT_BY_ISCYCLE 疑似列には、サイクルを含む行が表示されます。

```
SELECT last_name "Employee", CONNECT_BY_ISCYCLE "Cycle",
   LEVEL, SYS_CONNECT_BY_PATH(last_name, '/') "Path"
   FROM employees
   WHERE level <= 3 AND department_id = 80
   START WITH last_name = 'King'
   CONNECT BY NOCYCLE PRIOR employee_id = manager_id AND LEVEL <= 4
   ORDER BY "Employee", "Cycle", LEVEL, "Path";

Employee                       Cycle      LEVEL Path
------------------------- ---------- ---------- -------------------------
Abel                               0          3 /King/Zlotkey/Abel
Ande                               0          3 /King/Errazuriz/Ande
Banda                              0          3 /King/Errazuriz/Banda
Bates                              0          3 /King/Cambrault/Bates
Bernstein                          0          3 /King/Russell/Bernstein
Bloom                              0          3 /King/Cambrault/Bloom
Cambrault                          0          2 /King/Cambrault
Cambrault                          0          3 /King/Russell/Cambrault
Doran                              0          3 /King/Partners/Doran
Errazuriz                          0          2 /King/Errazuriz
Fox                                0          3 /King/Cambrault/Fox
...
```

CONNECT_BY_ROOT の例　次の例では、部門 110 の各従業員の名字、階層内で各従業員の上位に位置するマネージャ、マネージャと従業員間のレベル数および両者間のパスを戻します。

```
SELECT last_name "Employee", CONNECT_BY_ROOT last_name "Manager",
   LEVEL-1 "Pathlen", SYS_CONNECT_BY_PATH(last_name, '/') "Path"
   FROM employees
   WHERE LEVEL > 1 and department_id = 110
```

```
   CONNECT BY PRIOR employee_id = manager_id
   ORDER BY "Employee", "Manager", "Pathlen", "Path";

Employee        Manager            Pathlen Path
--------------- --------------- ---------- ------------------------------
Gietz           Higgins                  1 /Higgins/Gietz
Gietz           King                     3 /King/Kochhar/Higgins/Gietz
Gietz           Kochhar                  2 /Kochhar/Higgins/Gietz
Higgins         King                     2 /King/Kochhar/Higgins
Higgins         Kochhar                  1 /Kochhar/Higgins
```

次の例では、GROUP BY 句を使用して、部門 110 の各従業員と階層内でその従業員の下位に位置する従業員の合計の給与を戻します。

```
SELECT name, SUM(salary) "Total_Salary" FROM (
   SELECT CONNECT_BY_ROOT last_name as name, Salary
      FROM employees
      WHERE department_id = 110
      CONNECT BY PRIOR employee_id = manager_id)
      GROUP BY name
   ORDER BY name, "Total_Salary";

NAME                     Total_Salary
------------------------ ------------
Gietz                            8300
Higgins                         20300
King                            20300
Kochhar                         20300
```

参照：

- 階層問合せでのこれらの疑似列の処理方法については、3-3 ページの「LEVEL 疑似列」および 3-2 ページの「CONNECT_BY_ISCYCLE 疑似列」を参照してください。
- ルートからノードへの列の値のパスの検索については、5-184 ページの「SYS_CONNECT_BY_PATH」を参照してください。
- ORDER BY 句の SIBLINGS キーワードについては、19-29 ページの「order_by_clause」を参照してください。

UNION [ALL]、INTERSECT および MINUS 演算子

集合演算子 UNION、UNION ALL、INTERSECT および MINUS を使用して、複数の問合せを組み合せることができます。集合演算子の優先順位はすべて同じです。SQL 文に複数の集合演算子がある場合、カッコによって明示的に別の順序が指定されないかぎり、Oracle Database は左から右の順に評価します。

複合問合せを構成する各問合せと、それに対応する SELECT 構文のリスト内の各式は、数値が一致し、データ型グループ（数値や文字など）が同じである必要があります。

集合演算子によって結合された 2 つの問合せが文字データを選択する場合、戻される値のデータ型は次のようにして決定されます。

- 両方の問合せが同じ長さの CHAR データ型の値を選択する場合、戻される値のデータ型はその長さの CHAR になります。両方の問合せが異なる長さの CHAR データ型の値を選択する場合、戻される値は、長い方の CHAR 値の長さを使用した VARCHAR2 になります。
- 問合せのどちらか一方または両方が、VARCHAR2 データ型の値を選択する場合、戻される値のデータ型は VARCHAR2 になります。

集合演算子によって結合された 2 つの問合せが数値データを選択する場合、戻される値のデータ型は数値の優先順位によって決定されます。

- すべての問合せが BINARY_DOUBLE 型の値を選択する場合、戻される値のデータ型は BINARY_DOUBLE になります。
- いずれの問合せも BINARY_DOUBLE 型の値を選択せず、BINARY_FLOAT 型の値を選択する場合、戻される値のデータ型は BINARY_FLOAT になります。
- すべての問合せが NUMBER 型の値を選択する場合、戻される値のデータ型は NUMBER になります。

集合演算子を使用する問合せでは、データ型グループ間の暗黙的な変換は行われません。そのため、複合問合せの対応する式が文字データと数値データの両方になる場合は、エラーが戻されます。

> **参照：** 暗黙的な変換の詳細は、2-40 ページの表 2-10「暗黙的な型変換のマトリックス」を参照してください。数値の優先順位の詳細は、2-15 ページの「数値の優先順位」を参照してください。

例 次の問合せは有効です。

```
SELECT 3 FROM DUAL
   INTERSECT
SELECT 3f FROM DUAL;
```

この問合せは、次の複合問合せに暗黙的に変換されます。

```
SELECT TO_BINARY_FLOAT(3) FROM DUAL
   INTERSECT
SELECT 3f FROM DUAL;
```

次の問合せはエラーを戻します。

```
SELECT '3' FROM DUAL
   INTERSECT
SELECT 3f FROM DUAL;
```

集合演算子の制限事項： 集合演算子には、次の制限事項があります。

- 集合演算子は、データ型が BLOB、CLOB、BFILE、VARRAY またはネストした表である列に対しては無効になります。
- UNION、INTERSECT および MINUS 演算子は、LONG 列に対しては無効になります。
- 集合演算子の前の SELECT 構文のリストに式が含まれている場合、*order_by_clause* でその式を参照するには、式に列の別名を指定する必要があります。
- *for_update_clause* は、集合演算子とともに指定できません。
- これらの演算子の副問合せには、*order_by_clause* を指定できません。
- TABLE コレクション式を含む SELECT 文では、これらの演算子を使用できません。

> **注意：** SQL 規格に準拠するために、Oracle の今後のリリースでは、他の集合演算子より優先順位の高い INTERSECT 演算子が提供されます。したがって、INTERSECT 演算子と他の集合演算子を使用する問合せでは、カッコを使用して評価順序を指定してください。

UNION の例　次の文は、UNION 演算子によって 2 つの問合せの結果を結合しています。結果に重複行は含まれません。次の文は、他の表に存在していない列がある場合に、（TO_CHAR ファンクションを使用して）データ型を一致させる必要があることを示しています。

```
SELECT location_id, department_name "Department",
   TO_CHAR(NULL) "Warehouse"  FROM departments
   UNION
   SELECT location_id, TO_CHAR(NULL) "Department", warehouse_name
   FROM warehouses;

LOCATION_ID Department                     Warehouse
----------- ------------------------------ --------------------------
       1400 IT
       1400                                Southlake, Texas
       1500 Shipping
       1500                                San Francisco
       1600                                New Jersey
       1700 Accounting
       1700 Administration
       1700 Benefits
       1700 Construction
       1700 Contracting
       1700 Control And Credit
...
```

UNION ALL の例　UNION ALL 演算子がすべての行を戻すのに対して、UNION 演算子は重複しない行のみを戻します。UNION ALL 演算子は、重複行も対象に含めます。

```
SELECT product_id FROM order_items
UNION
SELECT product_id FROM inventories
ORDER BY product_id;

SELECT location_id  FROM locations
UNION ALL
SELECT location_id  FROM departments
ORDER BY location_id;
```

問合せで複数回戻される location_id 値（1700 など）は、UNION 演算子では 1 回のみ戻されますが、UNION ALL 演算子では複数回戻されています。

INTERSECT の例　次の文は、INTERSECT 演算子によって 2 つの結果を結合しています。この場合、両方の問合せによって共通に戻される一意の行のみが戻されます。

```
SELECT product_id FROM inventories
INTERSECT
SELECT product_id FROM order_items
ORDER BY product_id;
```

MINUS の例　次の文は、MINUS 演算子を使用して 2 つの結果を結合します。この場合、最初の問合せでは戻されるが、2 番目の問合せでは戻されない一意の行のみが戻されます。

```
SELECT product_id FROM inventories
MINUS
SELECT product_id FROM order_items
ORDER BY product_id;
```

問合せ結果のソート

ORDER BY 句を使用して、問合せによって選択された行を順序付けます。位置のソートは次のような場合に有効です。

- 長い SELECT 構文のリストの式によって順序付けるためには、全体の式を複製するのではなく、ORDER BY 句でその位置を指定することができます。
- 集合演算子 UNION、INTERSECT、MINUS または UNION ALL を含む複合問合せでは、ORDER BY 句に明示的な式ではなく、位置または別名を指定する必要があります。また ORDER BY 句は、最後のコンポーネントの問合せにのみ使用できます。ORDER BY 句は、複合問合せ全体によって戻されたすべての行を順序付けます。

ORDER BY 句による値のソートは、NLS_SORT 初期化パラメータによって明示的に指定するか、NLS_LANGUAGE 初期化パラメータによって暗黙的に指定します。ALTER SESSION 文を使用すると、ソート方法を 1 つの言語ソート基準から別の言語ソート基準に動的に変更できます。

> **参照：** NLS パラメータの詳細は、5-109 ページの「NLSSORT」および『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

結合

結合とは、2 つ以上の表、ビューまたはマテリアライズド・ビューの行を結合する問合せです。複数の表が問合せの FROM 句に指定される場合、Oracle Database は結合を実行します。問合せの SELECT 構文のリストは、これらの表のいずれかの任意の列を選択することができます。これらの表のいずれか 2 つに共通の列名を持つものがある場合、問合せの間、これらの列に対してすべての参照を明確にするために表の名前を付けて修飾する必要があります。

結合条件

ほとんどの結合問合せには、FROM 句または WHERE 句のいずれかに 1 つ以上の**結合条件**が含まれます。結合条件によって、異なる表から 2 つの列が比較されます。結合を実行するために、Oracle Database は各表に 1 つずつ含まれている列を結合し、結合条件が TRUE になるようにします。結合条件の列を SELECT 構文のリストに表示する必要はありません。

3 つ以上の表を結合するために、Oracle はまず列を比較する結合条件に基づいて 2 つの表を結合し、結合された表と新規の表の列を含む結合条件に基づいて、さらにもう 1 つの表を結合します。すべての表が結果に結合されるまで、このプロセスを継続します。オプティマイザは、Oracle が結合条件に基づいて表を結合する順序、表の索引、および任意の使用可能な表の統計を決定します。

結合条件を含む IA WHERE 句には、1 つの表のみの列を参照する別の条件も含めることができます。これらの条件は、結合問合せによって戻された列をさらに制限することができます。

> **注意：** 結合条件を含む WHERE 句には、LOB 列を指定できません。WHERE 句での LOB の使用については、他にも制限事項があります。詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』を参照してください。

等価結合

等価結合とは、等価演算子を含む結合条件での結合のことです。等価結合は、指定した列に同等の値を持つ行を結合します。オプティマイザが結合の実行を選択する内部アルゴリズムによって、1 つの表の等価結合条件における列の合計サイズは、データ・ブロックのサイズ以下に制限される可能性があります。データ・ブロックのサイズは、初期化パラメータ DB_BLOCK_SIZE によって指定されます。

> **参照：** 「結合問合せの使用例 :」（19-40 ページ）

自己結合

自己結合とは、自己の表結合のことです。この表は FROM 句に 2 回指定され、結合条件の列名を修飾する表の別名が続きます。自己結合を実行するために、Oracle Database は結合条件を満たす表の行を結合して戻します。

参照：「自己結合の使用例 :」（19-41 ページ）

デカルト積

結合問合せの 2 つの表に結合条件がない場合、Oracle Database は**デカルト積**を戻します。この場合、1 つの表の各行が別の表の各行に結合されます。デカルト積は常に多数の行を生成するため、有効ではありません。たとえば、それぞれが 100 行を持つ 2 つの表のデカルト積は 10,000 行を生成します。特にデカルト積を必要としないかぎり、必ず結合条件を指定してください。問合せが 3 つ以上の表を結合し、特定の組に対して結合条件を指定しない場合、オプティマイザは、中間のデカルト積を生成しないように結合順序を選択する可能性があります。

内部結合

内部結合（**単純結合**）とは、結合条件を満たす行のみを戻す、複数の表の結合です。

外部結合

外部結合は、単純結合の結果を拡張します。外部結合は、結合条件を満たすすべての行と、結合条件を満たす行を除いた、一方の表のすべての行を戻します。

- 表 A および B の外部結合を行い、すべての行を A から戻す問合せ（**左側外部結合**）を記述するには、FROM 句で LEFT [OUTER] JOIN 構文を使用するか、WHERE 句の結合条件で外部結合演算子（+）を B のすべての列に適用します。B に一致する行のない A のすべての行に関して、Oracle Database は、B の列を含む任意の SELECT 構文のリストの式に NULL を戻します。
- 表 A および B の外部結合を行い、すべての行を B から戻す問合せ（**右側外部結合**）を記述するには、FROM 句で RIGHT [OUTER] JOIN 構文を使用するか、WHERE 句の結合条件で外部結合演算子（+）を A のすべての列に適用します。A に一致する行のない B のすべての行に関して、Oracle は、A の列を含む任意の SELECT 構文のリストの式に NULL を戻します。
- 外部結合を行い、結合条件を満たさない場合にすべての行を A および B から NULL で拡張して戻す問合せ（**完全外部結合**）を記述するには、FROM 句で FULL [OUTER] JOIN 構文を使用します。

指定する書式にかかわらず、外部結合の WHERE 句の副問合せでは列を比較できません。

外部結合を使用すると、疎データ内の欠損を補完できます。このような結合はパーティション化された外部結合と呼ばれ、*join_clause* 構文の *query_partition_clause* を使用して形成されます。疎データとは、時刻や部門などのディメンションの一部の値に対する行を持たないデータです。たとえば、販売データの表には通常、売上のない任意の日付の製品に対する行は存在しません。データの欠損の補完は、データの欠損によって分析計算が複雑になる場合や、疎データを直接問い合せた場合に一部のデータを見逃す可能性がある際に役立ちます。

参照：

- 外部結合を使用した疎データの欠損の補完方法の詳細は、19-20 ページの「join_clause」を参照してください。
- グループ外部結合および疎データの欠損補完の詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

Oracle の結合演算子よりも、FROM 句の OUTER JOIN 構文を使用することをお薦めします。Oracle の結合演算子（+）を使用した外部結合問合せには、次の規則と制限事項があります（これらの規則や制限事項は、FROM 句の OUTER JOIN 構文にはありません）。

- FROM 句の結合構文を含む問合せブロックには結合演算子（+）を指定できません。
- 結合演算子（+）は、WHERE 句または FROM 句の左相関のコンテキスト（TABLE 句を指定する場合）にのみ指定でき、表またはビューの列にのみ適用されます。
- A および B が複数の結合条件によって結合される場合、これらの条件のすべてにおいて結合演算子（+）を使用する必要があります。使用しない場合、Oracle Database は単純結合の結果である行のみを戻しますが、外部結合の結果がないことを示す警告やエラーは出力しません。
- ある表を外部問合せに指定して別の表を内部問合せに指定した場合、結合演算子（+）は外部結合を生成しません。
- 自己結合が有効であっても、結合演算子（+）を使用して表を自己に外部結合することはできません。たとえば、次の文は無効です。

```
-- The following statement is not valid:
SELECT employee_id, manager_id
   FROM employees
   WHERE employees.manager_id(+) = employees.employee_id;
```

ただし、次の自己結合は有効です。

```
SELECT e1.employee_id, e1.manager_id, e2.employee_id
   FROM employees e1, employees e2
   WHERE e1.manager_id(+) = e2.employee_id
   ORDER BY e1.employee_id, e1.manager_id, e2.employee_id;
```

- 結合演算子（+）は任意の式ではなく、列にのみ適用することができます。ただし、任意の式には結合演算子（+）でマークされた 1 つ以上の列を含めることができます。
- 結合演算子（+）を含む WHERE 条件は、OR 論理演算子を使用する他の条件と結合できません。
- WHERE 条件は、IN 比較条件を使用して、結合演算子（+）でマークされた列を式と比較できません。

WHERE 句に表 B の列と定数を比較する条件が含まれる場合、Oracle がこの列に対して NULL を生成する表 A の列を戻すように、結合演算子（+）をこの列に適用する必要があります。それ以外の場合、Oracle は単純結合の結果のみを戻します。

2 組以上の表の外部結合を行う問合せにおいて、単一表は他の 1 つの表のみに対して NULL 生成された表になることができます。そのため、A と B の結合条件および B と C の結合条件における B の列に、結合演算子（+）を適用することはできません。外部結合の構文については、19-4 ページの「SELECT」を参照してください。

アンチ結合

アンチ結合は、述語の右側に対応する行を持たない述語の左側の行を戻します。この結合は、右側の副問合せに一致しない（NOT IN）行を戻します。

参照：「アンチ結合の使用例：」（19-44 ページ）

セミ結合

セミ結合は、述語の右側の複数の行が副問合せの条件を満たす場合に、述語の左側から行を重複させずに EXISTS 副問合せに一致する行を戻します。

副問合せが WHERE 句の OR ブランチに指定されている場合、セミ結合およびアンチ結合変換は実行できません。

参照：「セミ結合の使用例 :」（19-44 ページ）

副問合せの使用方法

副問合せは、複数部分の問合せに応答します。たとえば、Taylor の部門で働いている人を判断するには、まず Taylor が働く部門を判断する副問合せを使用できます。その後、親 SELECT 文で元の問合せに応答することができます。SELECT 文の FROM 句の副問合せは、**インライン・ビュー**とも呼ばれます。任意の数の副問合せをインライン・ビュー内にネストできます。また、SELECT 文の WHERE 句の副問合せは、**ネストした副問合せ**とも呼ばれます。ネストした副問合せには、最大 255 レベルの副問合せをネストできます。

副問合せは、別の副問合せを含むことができます。トップレベル問合せの FROM 句内の副問合せレベルの数には、制限がありません。WHERE 句には、最大 255 レベルの副問合せをネストできます。

副問合せにある列が、含まれる文の列と同じ名前を持つ場合、含まれる文の表の列に表名または別名で参照の接頭辞を付ける必要があります。文をさらに読みやすくするには、常に、表、ビューまたはマテリアライズド・ビューの名前または別名で副問合せの列を修飾します。

ネストした副問合せが、その副問合せから任意のレベル上位の親である文で参照する表の列を参照する場合、Oracle は**相関副問合せ**を行います。親である文は、副問合せがネストしている SELECT、UPDATE または DELETE 文のいずれかです。相関副問合せは、親である文によって処理された各行を 1 回評価します。Oracle は、副問合せで指定された表内を検索した後、親である文で指定された表内を検索することによって、副問合せ内の未修飾列を解決します。

相関副問合せは、応答が親である文によって処理された各列の値に依存する複数部分の問合せに応答します。たとえば、相関副問合せを使用して、部門内で給与が平均給与以上の従業員を判断することができます。この場合、相関副問合せは独自で各部門の平均給与を計算します。

参照：「相関副問合せの使用例 :」（19-47 ページ）

副問合せは、次の用途に使用します。

- INSERT または CREATE TABLE 文のターゲット表に挿入する一連の行を定義します。
- CREATE VIEW または CREATE MATERIALIZED VIEW 文のビューまたはマテリアライズド・ビューに含める一連の行を定義します。
- UPDATE 文の既存の行に割り当てる 1 つ以上の値を定義します。
- SELECT、UPDATE および DELETE 文の WHERE 句、HAVING 句または START WITH 句における条件に対する値を定義します。
- 含まれる問合せによって操作される表を定義します。

  表名を指定する場合と同様に、問合せを含む FROM 句に副問合せを指定することによってこれらのことを行います。INSERT、UPDATE および DELETE 文においても、このようにして表のかわりに副問合せを使用することができます。

  このように使用された副問合せには、相関変数を指定できますが、外部参照ではなく、その副問合せ内に定義された相関変数のみを指定できます。詳細は、19-18 ページの「table_collection_expression」を参照してください。

  1 つの行から 1 つの列の値を戻すスカラー副問合せは、有効な書式の式です。構文で *expr* をコールするほとんどの場合に、スカラー副問合せ式を使用できます。詳細は、6-14 ページの「スカラー副問合せ式」を参照してください。

ネストされた副問合せのネスト解除

副問合せは、親である文の WHERE 句内にあるときは**ネスト**されています。ネストされた副問合せを持つ文を評価する場合、Oracle Database は、副問合せ部分を複数回評価する必要があり、効果的なアクセス・パスまたは結合を見逃してしまう可能性があります。

副問合せのネスト解除によって、副問合せの本体がネスト解除され、その副問合せを含む文の本体に結合されます。これによって、アクセス・パスおよび結合の評価時に、オプティマイザが副問合せと文を 1 つのものと判断します。オプティマイザは、ほぼすべての副問合せをネスト解除できますが、いくつか例外があります。これらの例外としては、階層副問合せ、および ROWNUM 疑似列、集合演算子の 1 つ、ネストした集計ファンクション、副問合せの直接的な外部問合せブロックではない問合せブロックへの相関参照を含む副問合せなどがあります。

制約がない場合、オプティマイザは、次のネストされた副問合せを自動的にネスト解除します（ただし、ネスト解除しない場合もあります）。

- 相関関係のない IN 副問合せ
- IN および EXISTS 相関副問合せ（集計ファンクションまたは GROUP BY 句を含まない場合）

ネスト解除された拡張副問合せを行うには、次のタイプの副問合せをネスト解除するようにオプティマイザに指示します。

- 副問合せに HASH_AJ または MERGE_AJ ヒントを指定して、NOT IN 副問合せをネスト解除します。
- 副問合せに UNNEST ヒントを指定して、その他の副問合せをネスト解除します。

> **参照：** ヒントの詳細は、2-70 ページの「ヒントの使用方法」を参照してください。

DUAL 表からの選択

DUAL は、データ・ディクショナリとともに Oracle Database によって自動的に作成された表です。DUAL は、ユーザー SYS のスキーマにありますが、すべてのユーザーが DUAL という名前でアクセスすることができます。DUAL は、VARCHAR2(1) として定義されている DUMMY 列を持ち、「X」値を持つ行を含みます。DUAL 表から選択することは、定数式を SELECT 文で計算する場合に便利です。DUAL には 1 行以外存在しないため、定数は 1 回のみ戻されます。一方で、任意の表から定数、疑似列または式を選択できますが、値は表の行の数のみ戻されます。DUAL から定数値を選択する例は、5-2 ページの「SQL ファンクション」を参照してください。

分散問合せ

Oracle 分散データベース管理システム・アーキテクチャによって、Oracle Net および Oracle Database サーバーを使用するリモート・データベースにアクセスできます。名前の最後に *@dblink* を追加して、リモート表、ビューまたはマテリアライズド・ビューを識別できます。*dblink* は、リモート表、ビューまたはマテリアライズド・ビューを含むデータベースへのデータベース・リンクの完全な名前または部分的な名前である必要があります。

> **参照：**
>
> - データベース・リンクの参照方法の詳細は、2-104 ページの「リモート・データベース内のオブジェクトの参照」を参照してください。
> - リモート・データベースへのアクセスについては、『Oracle Database Net Services 管理者ガイド』を参照してください。

分散問合せの制限事項： 現在、分散問合せには、FOR UPDATE 句によってロックされたすべての表、および問合せによって選択された LONG 列を持つすべての表が、同じデータベース上に位置している必要があるという制限があります。たとえば、次の文は、remote データベースの print_media 表から press_release（LONG 値）を選択し、local データベースの print_media 表をロックするため、エラーになります。

```
SELECT r.product_id, l.ad_id, r.press_release
   FROM pm.print_media@remote r, pm.print_media l
   FOR UPDATE OF l.ad_id;
```

また、Oracle Database は現在、リモート表にあるユーザー定義型またはオブジェクト REF のデータ型を選択する分散問合せをサポートしていません。

10

SQL 文 : ALTER CLUSTER ～ ALTER JAVA

この章では、様々な SQL 文についてアルファベット順に説明します。その他の SQL 文については、第 11 章～第 19 章を参照してください。

この章では、次の内容を説明します。

- 様々な種類の SQL 文
- SQL 文に関する章の構成
- ALTER CLUSTER
- ALTER DATABASE
- ALTER DIMENSION
- ALTER DISKGROUP
- ALTER FLASHBACK ARCHIVE
- ALTER FUNCTION
- ALTER INDEX
- ALTER INDEXTYPE
- ALTER JAVA

様々な種類の SQL 文

次の項にあるリストは、SQL 文の機能の概要について、次のカテゴリに分類して説明しています。

- データ定義言語（DDL）文
- データ操作言語（DML）文
- トランザクション制御文
- セッション制御文
- システム制御文
- 埋込み SQL 文

データ定義言語（DDL）文

データ定義言語（DDL）文によって、次のタスクを実行できます。

- スキーマ・オブジェクトの作成、変更および削除
- 権限およびロールの付与および取消し
- 表、索引またはクラスタ上の情報の分析
- 監査オプションの構築
- データ・ディクショナリへのコメントの追加

CREATE、ALTER および DROP コマンドは、特定のオブジェクトに対して排他的アクセスを必要とします。たとえば、別のユーザーが特定の表でトランザクションをオープンしている場合、ALTER TABLE 文は実行できません。

GRANT、REVOKE、ANALYZE、AUDIT および COMMENT コマンドは、特定のオブジェクトに対する排他的アクセスを必要としません。たとえば、他のユーザーが表を更新しているときでも、その表を分析できます。

Oracle Database は、暗黙的にすべての DDL 文の前後で現在のトランザクションをコミットします。

DDL 文の多くは、Oracle Database にスキーマ・オブジェクトを再コンパイルまたは再認可させることができます。Oracle Database がスキーマ・オブジェクトを再コンパイルまたは再認可する方法、および DDL 文によってそれを実行する環境については、『Oracle Database 概要』を参照してください。

DDL 文は、DBMS_SQL パッケージを使用した PL/SQL によってサポートされます。

> **参照：** このパッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

次に、DDL 文を示します。

ALTER （ALTER SESSION および ALTER SYSTEM を除く ALTER で始まるすべての文。10-3 ページの「セッション制御文」および 10-3 ページの「システム制御文」を参照）
ANALYZE
ASSOCIATE STATISTICS
AUDIT
COMMENT
CREATE（CREATE で始まるすべての文）
DISASSOCIATE STATISTICS
DROP（DROP で始まるすべての文）
FLASHBACK（FLASHBACK で始まるすべての文）
GRANT

```
NOAUDIT
PURGE
RENAME
REVOKE
TRUNCATE
```

データ操作言語（DML）文

データ操作言語（DML）文は、既存スキーマ・オブジェクトのデータにアクセスし、操作します。次の文は、現在のトランザクションを暗黙的にコミットしません。次に、データ操作言語文を示します。

```
CALL
DELETE
EXPLAIN PLAN
INSERT
LOCK TABLE
MERGE
SELECT
UPDATE
```

SELECT 文は、DML 文の制限された形式であり、データベース内のデータへのアクセスのみが可能です。アクセスしたデータを操作してから問合せの結果を戻すことはできますが、データベースに格納されたデータを操作することはできません。

CALL および EXPLAIN PLAN 文は、動的に実行されるときにのみ PL/SQL でサポートされます。他のすべての DML 文は、PL/SQL で完全にサポートされます。

トランザクション制御文

トランザクション制御文は、DML 文で行った変更を管理します。次に、トランザクション制御文を示します。

```
COMMIT
ROLLBACK
SAVEPOINT
SET TRANSACTION
SET CONSTRAINT
```

COMMIT および ROLLBACK コマンドの特定書式以外のトランザクション制御文は、PL/SQL でサポートされます。制限については、13-44 ページの「COMMIT」および 18-92 ページの「ROLLBACK」を参照してください。

セッション制御文

セッション制御文は、ユーザー・セッションのプロパティを動的に管理します。次の文は、現在のトランザクションを暗黙的にコミットしません。

PL/SQL は、セッション制御文をサポートしません。次に、セッション制御文を示します。

```
ALTER SESSION
SET ROLE
```

システム制御文

単一システム制御文 ALTER SYSTEM は、Oracle Database インスタンスのプロパティを動的に管理します。この文は、現在のトランザクションを暗黙的にコミットしません。また、PL/SQL ではサポートされません。

埋込み SQL 文

埋込み SQL 文は、DDL、DML およびトランザクション制御文を手続き型言語プログラム内に入れます。埋込み SQL は、Oracle プリコンパイラでサポートされており、次のマニュアルに記載されています。

- 『Pro*COBOL プログラマーズ・ガイド』
- 『Pro*C/C++ プログラマーズ・ガイド』
- 『Oracle SQL*Module for Ada Programmer's Guide』

SQL 文に関する章の構成

この章および第 11 章～第 19 章のすべての SQL 文は、次の項で編成されています。

構文　構文図には、文を構成するキーワードおよびパラメータを示します。

> **注意：**　すべてのキーワードおよびパラメータがあらゆる環境において有効なわけではありません。構文の制限事項については、文および句の「セマンティクス」を参照してください。

用途　文の基本的な使用方法を説明します。

前提条件　文の実行に必要な権限と、文を使用する前に実行する手順を示します。特に指定がないかぎり、ご使用のインスタンスでデータベースがオープンされている必要があります。

セマンティクス　構文を構成するキーワード、パラメータおよび句の用途について説明します。また、制限事項およびその他の注意事項についても説明します。（この章および第 11 章～第 19 章で使用するキーワードおよびパラメータの表記規則の詳細は、「はじめに」を参照してください。）

例　文の様々な句およびパラメータの使用方法を示します。

ALTER CLUSTER

用途

ALTER CLUSTER 文を使用すると、クラスタの記憶特性および並列特性を再定義できます。

注意： クラスタ・キーの列番号および列名を変更するためにこの文を使用することはできません。また、クラスタを格納する表領域を変更することはできません。

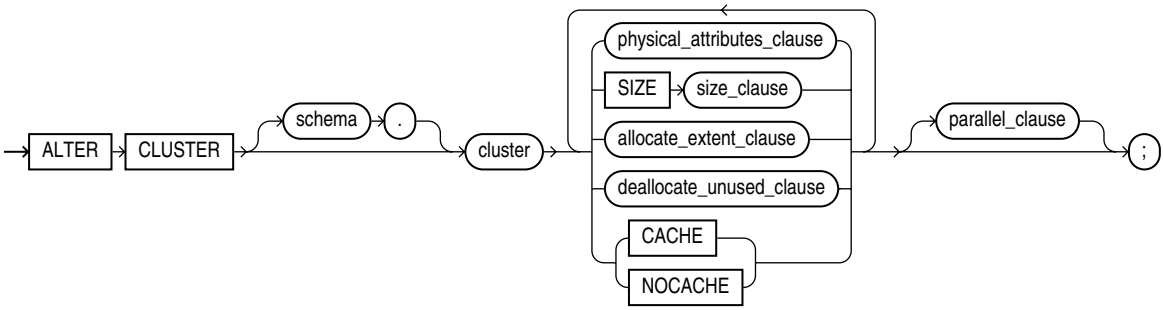
参照： クラスタの作成については、14-2 ページの「CREATE CLUSTER」を参照してください。クラスタからの表の削除については、17-32 ページの「DROP CLUSTER」および 18-5 ページの「DROP TABLE」を参照してください。クラスタへの表の追加については、16-30 ページの「CREATE TABLE」の「physical_properties」を参照してください。

前提条件

クラスタが自分のスキーマ内にあるか、または ALTER ANY CLUSTER システム権限が必要です。

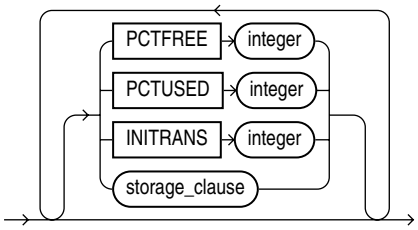
構文

***alter_cluster*::=**



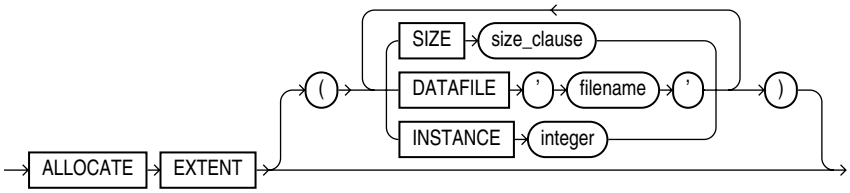
（10-5 ページの physical_attributes_clause::=、8-42 ページの size_clause::=、10-6 ページの allocate_extent_clause::=、10-6 ページの deallocate_unused_clause::=、10-6 ページの parallel_clause::= を参照）

physical_attributes_clause::=



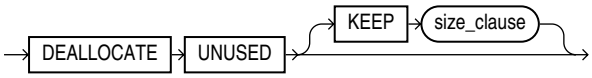
（8-44 ページの storage_clause::= を参照）

allocate_extent_clause::=



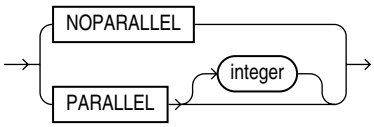
(8-42 ページの size_clause::= を参照)

deallocate_unused_clause::=



(8-42 ページの size_clause::= を参照)

parallel_clause::=



セマンティクス

schema

クラスタが含まれているスキーマを指定します。*schema* を指定しない場合、そのクラスタは自分のスキーマにあるとみなされます。

cluster

変更するクラスタの名前を指定します。

physical_attributes_clause

クラスタの PCTUSED パラメータ、PCTFREE パラメータおよび INITRANS パラメータの値を変更します。

クラスタの記憶特性を変更するには、STORAGE 句を使用します。

参照 :

- パラメータの詳細は、8-39 ページの「physical_attributes_clause」を参照してください。
- その句の詳細は、8-41 ページの「storage_clause」を参照してください。

物理属性の制限事項 : クラスタの記憶域パラメータ INITIAL および MINEXTENTS の値は変更できません。

SIZE *integer*

SIZE 句を使用すると、クラスタに割り当てられたデータ・ブロック中に格納されるクラスタ・キーの数を指定できます。

SIZE の制限事項： ハッシュ・クラスタではなく、索引クラスタの SIZE パラメータのみを変更できます。

> **参照：** SIZE パラメータについては、14-2 ページの「CREATE CLUSTER」および 10-7 ページの「クラスタの変更例 :」を参照してください。

allocate_extent_clause

allocate_extent_clause を指定すると、クラスタの新しいエクステントを明示的に割り当てることができます。

この句で明示的にエクステントを割り当てる場合、Oracle Database は、クラスタの記憶域パラメータを評価しません。割り当てられる新しいエクステントの新しいサイズも決定しません（表を作成する際に行います）。したがって、Oracle Database がデフォルト値を使用しないようにするには、SIZE を指定してください。

エクステントの割当ての制限事項： ハッシュ・クラスタではなく、索引クラスタのみに新しいエクステントを割り当てることができます。

> **参照：** この句の詳細は、8-2 ページの「allocate_extent_clause」および 10-8 ページの「未使用領域の解放例 :」を参照してください。

deallocate_unused_clause

deallocate_unused_clause 句を使用すると、クラスタの終わりの未使用領域の割当てを明示的に解除し、解放された領域が他のセグメントで使用可能になります。

> **参照：** この句の詳細は、8-24 ページの「deallocate_unused_clause」を参照してください。

CACHE | NOCACHE

この句の動作は、CREATE CLUSTER 文および ALTER CLUSTER 文で同じです。

> **参照：** この句の詳細は、14-6 ページの「CACHE | NOCACHE」を参照してください。

parallel_clause

parallel_clause を指定すると、クラスタの DML および問合せのデフォルト並列度を変更できます。

並列化するクラスタの制限事項： *cluster* の表が LOB 型またはユーザー定義オブジェクト型の列を含む場合、*cluster* でその後に行う INSERT、UPDATE または DELETE 操作と同様、この文は通知なしに逐次実行されます。

> **参照：** この句の詳細は、16-54 ページの「CREATE TABLE」の「parallel_clause」を参照してください。

例

次に、14-7 ページの「CREATE CLUSTER」の「例」で作成したクラスタを変更する例を示します。

クラスタの変更例： 次の文は、personnel クラスタを変更します。

```
ALTER CLUSTER personnel
   SIZE 1024 CACHE;
```

この結果、各クラスタ・キー値に 1024 バイトが割り当てられ、CACHE 属性が有効になります。データ・ブロックのサイズを 2KB と想定した場合、このクラスタ内の今後のデータ・ブロックには、各ブロックに 2 つのクラスタ・キー（2KB を 1024 バイトで割った値）が含まれます。

未使用領域の解放例： 次の文は、language クラスタから未使用領域の割当てを解除し、後で使用できるように 30KB の未使用領域を保持します。

```
ALTER CLUSTER language
   DEALLOCATE UNUSED KEEP 30 K;
```

ALTER DATABASE

用途

ALTER DATABASE 文を使用すると、既存のデータベースを変更、メンテナンスまたはリカバリできます。

注意： 以前のバージョンの Oracle Database では、ALTER DATABASE を使用して次の 2 種類の変換操作を実行できました。

- RESET COMPATIBILITY 句 : 次のインスタンス起動時に、データベースを以前のバージョンにリセットします。
- CONVERT 句 : Oracle7 のデータ・ディクショナリを、Oracle8*i* または Oracle9*i* のデータ・ディクショナリにアップグレードします。

今回のバージョンでは、これらの句はサポートされていません。移行や相互運用性の問題については、『Oracle Database アップグレード・ガイド』を参照してください。

参照 :

- メディア・リカバリの実行例については、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- スタンバイ・データベースをメンテナンスするために ALTER DATABASE 文を使用する場合の詳細は、『Oracle Data Guard 概要および管理』を参照してください。
- データベースの作成の詳細は、14-16 ページの「CREATE DATABASE」を参照してください。

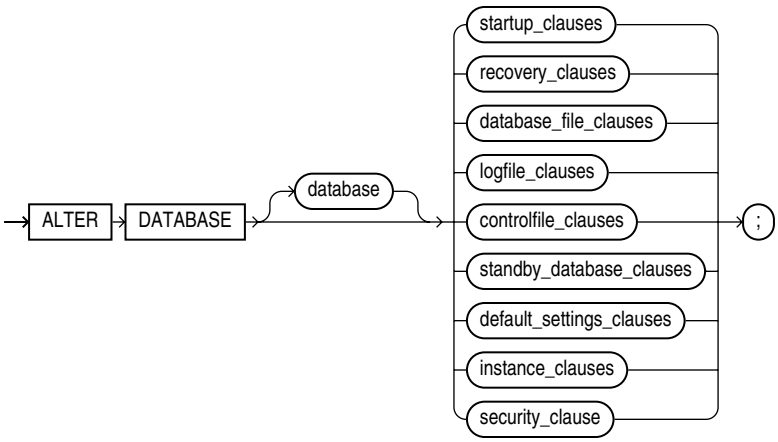
前提条件

ALTER DATABASE システム権限が必要です。

RECOVER 句を指定する場合は、SYSDBA システム権限が必要です。

構文

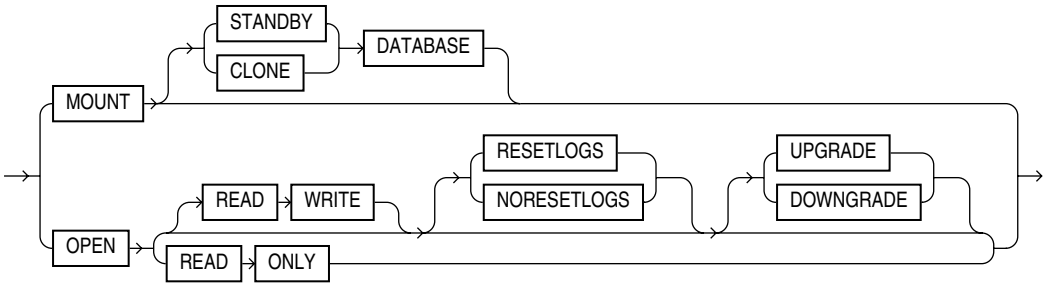
***alter_database*::=**



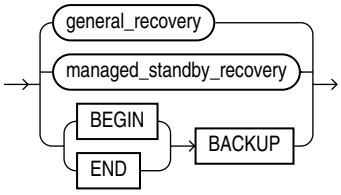
ALTER DATABASE 構文のグループは、次のとおりです。

- **startup_clauses::=**（10-10 ページ）
- **recovery_clauses::=**（10-10 ページ）
- **database_file_clauses::=**（10-12 ページ）
- **logfile_clauses::=**（10-13 ページ）
- **controlfile_clauses::=**（10-14 ページ）
- **standby_database_clauses::=**（10-15 ページ）
- **default_settings_clauses::=**（10-17 ページ）
- **instance_clauses::=**（10-17 ページ）
- **security_clause::=**（10-17 ページ）

startup_clauses::=

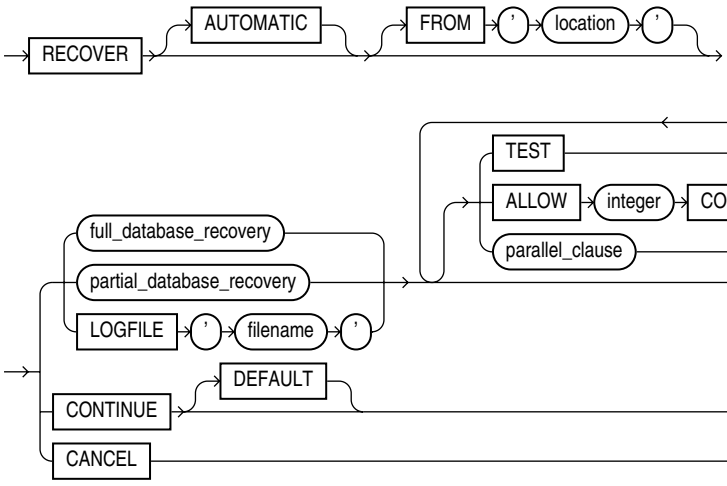


recovery_clauses::=



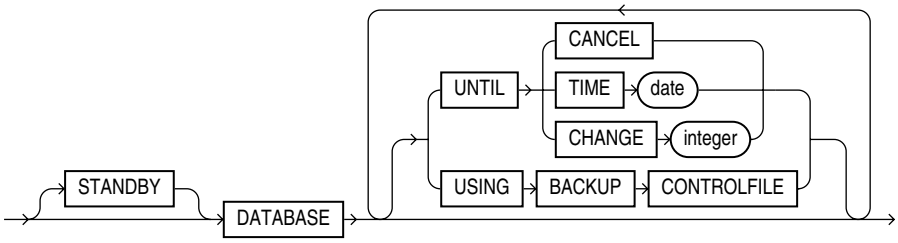
（10-10 ページの general_recovery::=、10-11 ページの managed_standby_recovery::= を参照）

general_recovery::=

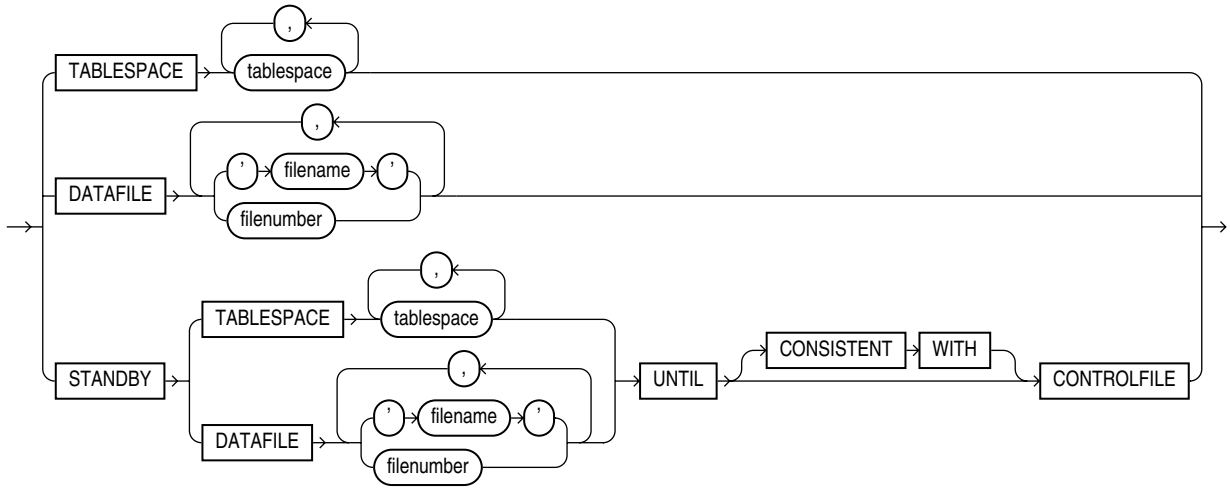


（10-11 ページの full_database_recovery::=、10-11 ページの partial_database_recovery::= を参照）

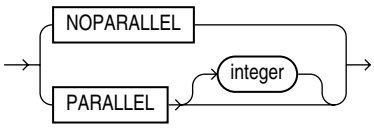
full_database_recovery::=



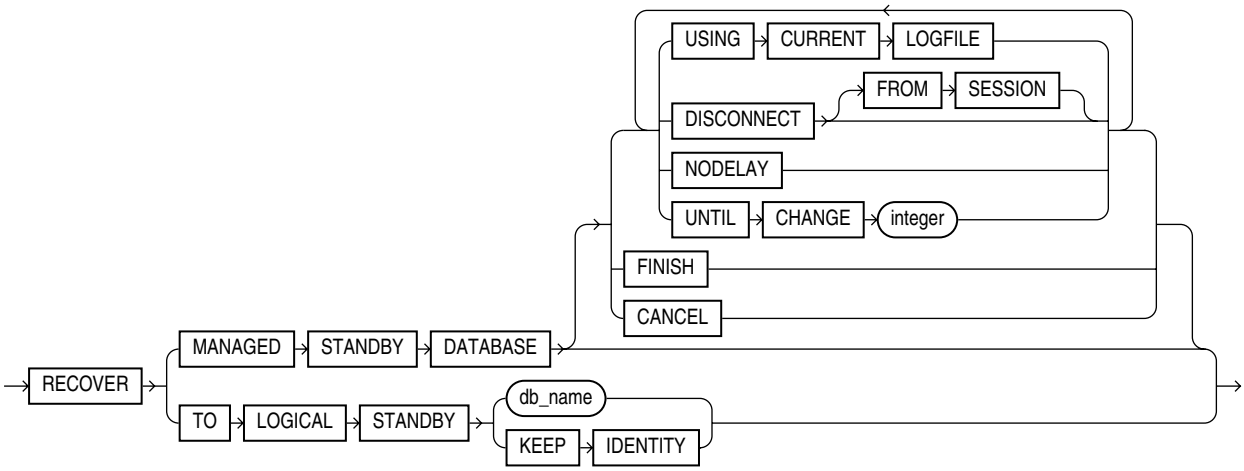
partial_database_recovery::=



***parallel_clause*::=**

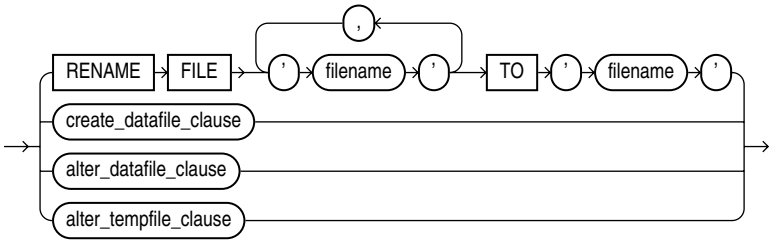


managed_standby_recovery::=



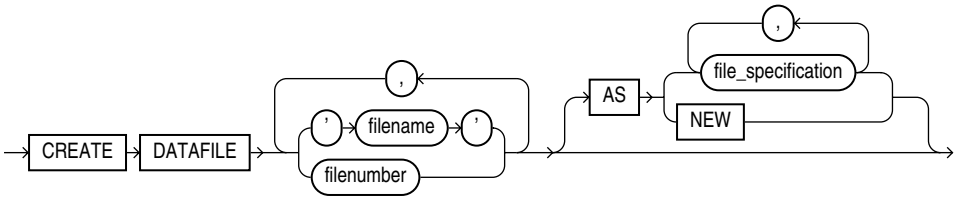
注意： *managed_standby_recovery* のいくつかの副次句は不要であり、その使用は非推奨になっています。これらの副次句は、構文図にも記載されていません。10-22 ページの「managed_standby_recovery」のセマンティクスを参照してください。

database_file_clauses::=



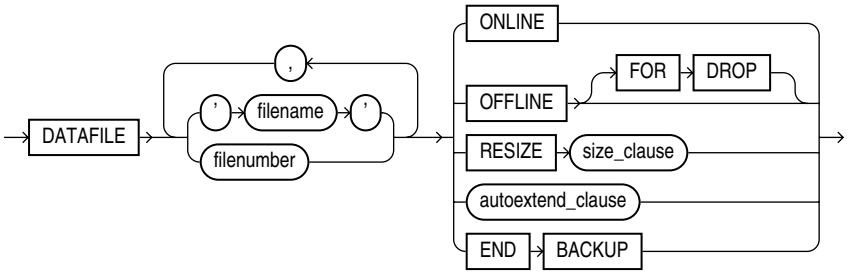
（10-12 ページの create_datafile_clause::=、10-12 ページの alter_datafile_clause::=、10-12 ページの alter_tempfile_clause::= を参照）

create_datafile_clause::=



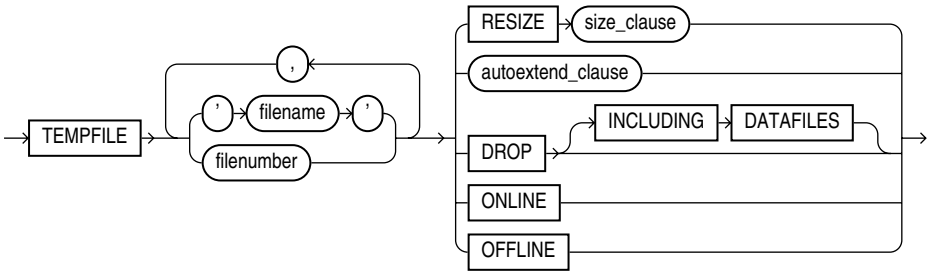
（8-26 ページの file_specification::= を参照）

alter_datafile_clause::=



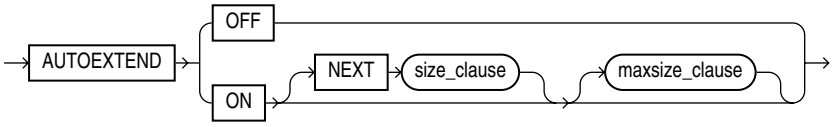
（10-13 ページの autoextend_clause::=、8-42 ページの size_clause::= を参照）

alter_tempfile_clause::=

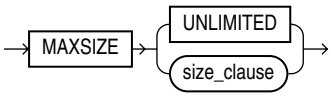


（10-13 ページの autoextend_clause::=、8-42 ページの size_clause::= を参照）

autoextend_clause::=

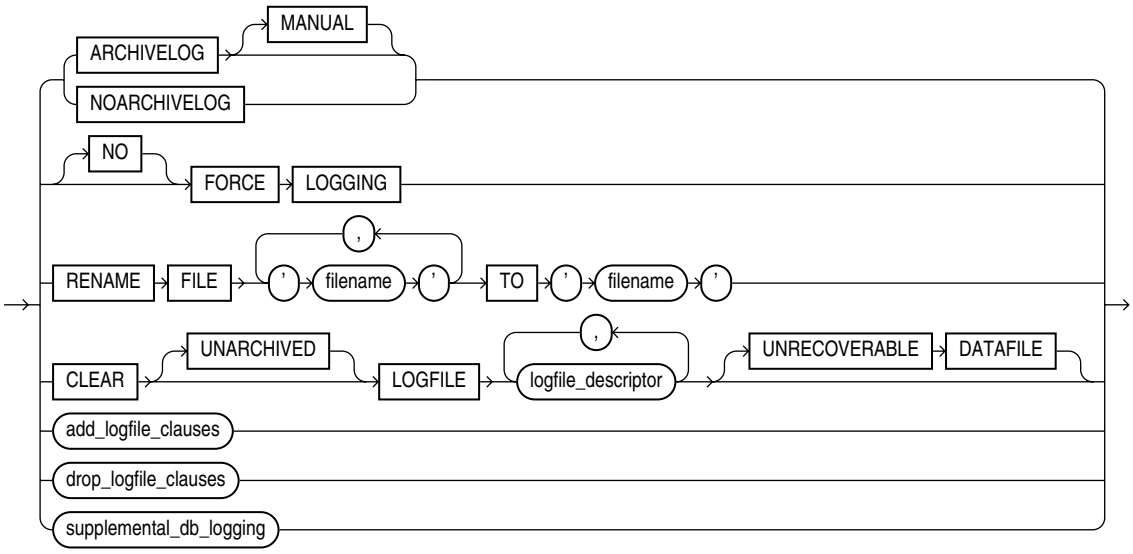


***maxsize_clause*::=**



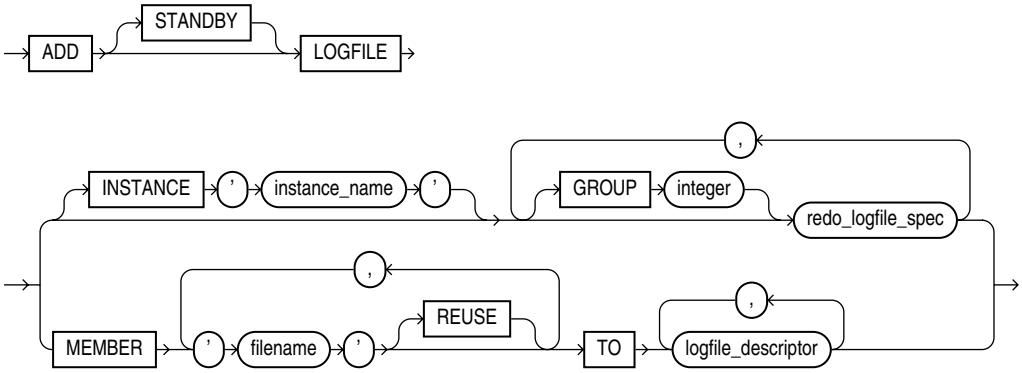
（8-42 ページの size_clause::= を参照）

logfile_clauses::=



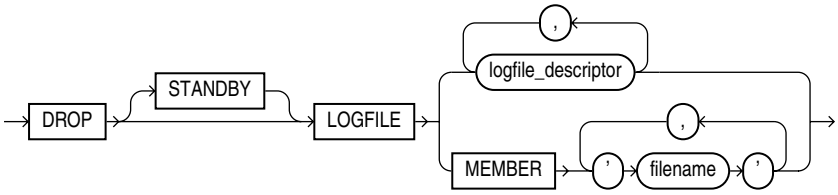
（10-14 ページの logfile_descriptor::=、10-13 ページの add_logfile_clauses::=、
10-14 ページの drop_logfile_clauses::=、10-14 ページの supplemental_db_logging::= を参照）

add_logfile_clauses::=



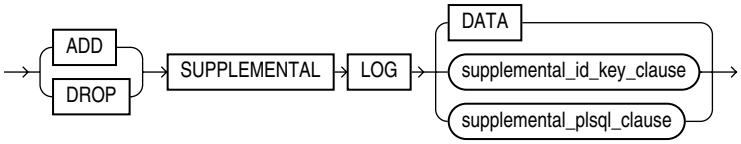
（8-26 ページの redo_log_file_spec::=、10-14 ページの logfile_descriptor::= を参照）

drop_logfile_clauses::=



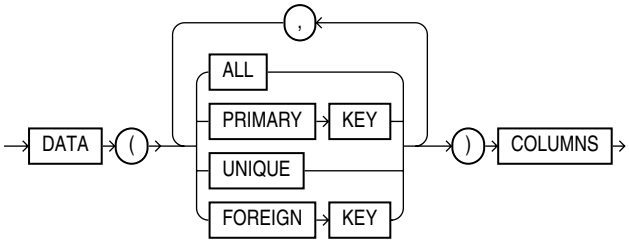
(10-14 ページの logfile_descriptor::= を参照)

supplemental_db_logging::=

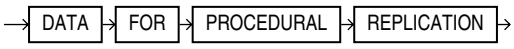


(10-14 ページの supplemental_id_key_clause::= を参照)

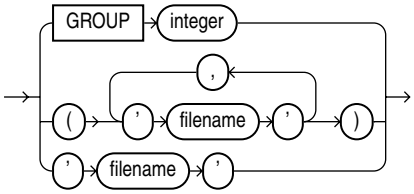
***supplemental_id_key_clause*::=**



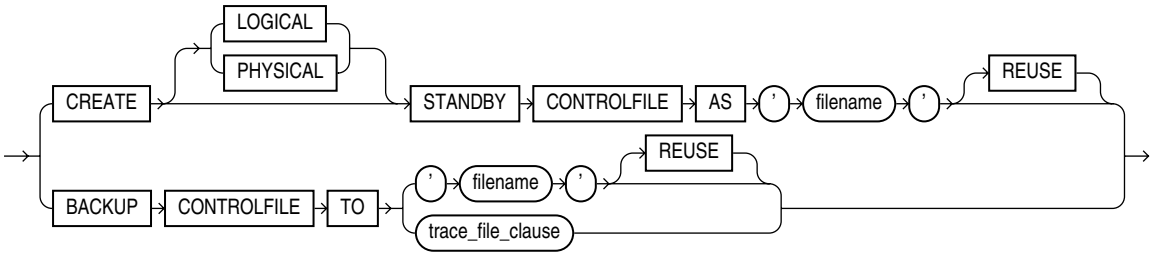
***supplemental_plsql_clause*::=**



***logfile_descriptor*::=**

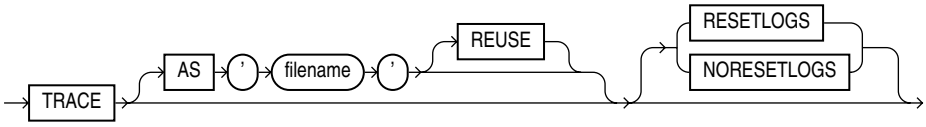


controlfile_clauses::=

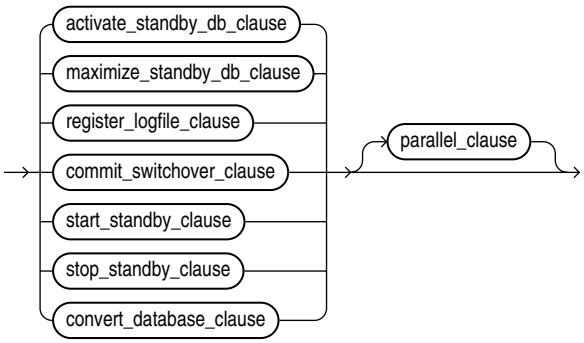


(10-15 ページの trace_file_clause::= を参照)

***trace_file_clause*::=**

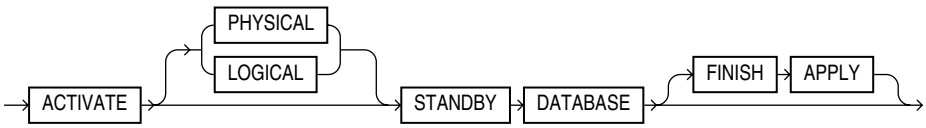


standby_database_clauses::=

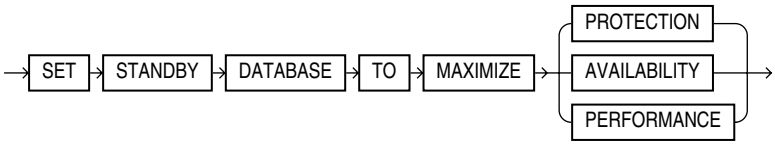


（10-15 ページの activate_standby_db_clause::=、
10-15 ページの maximize_standby_db_clause::=、10-15 ページの register_logfile_clause::=、
10-16 ページの commit_switchover_clause::=、10-16 ページの start_standby_clause::=、
10-16 ページの stop_standby_clause::=、10-16 ページの convert_database_clause::=、
10-11 ページの parallel_clause::= を参照）

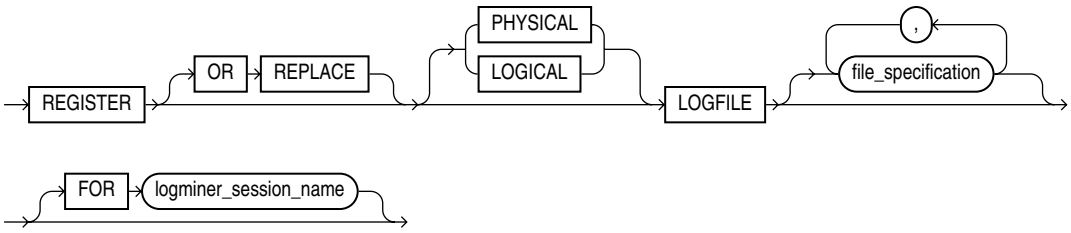
activate_standby_db_clause::=



maximize_standby_db_clause::=

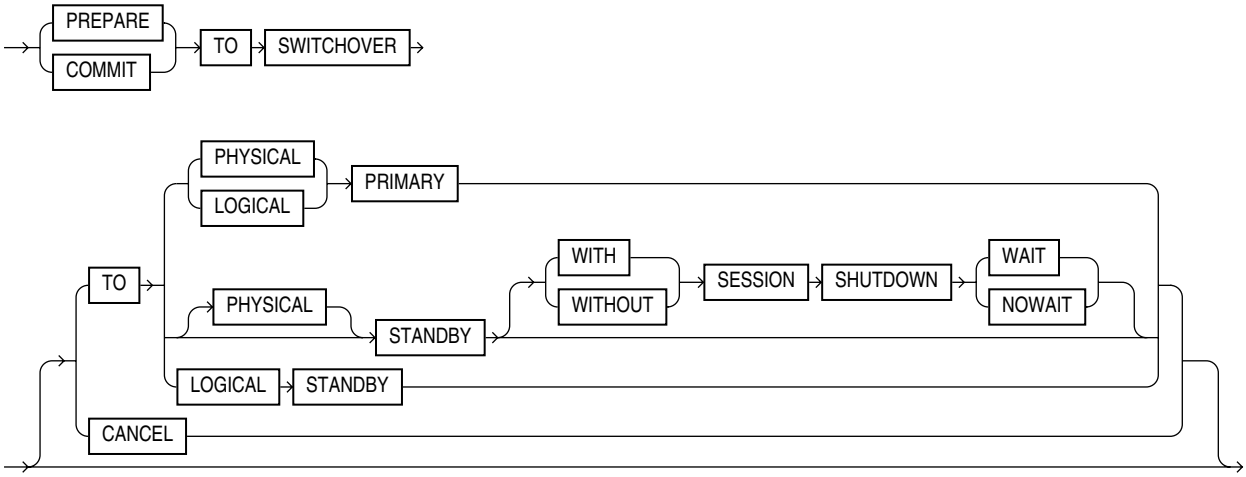


register_logfile_clause::=

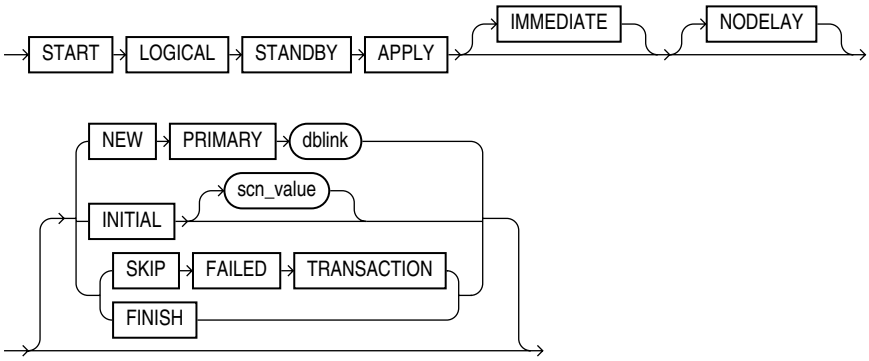


（8-26 ページの file_specification::= を参照）

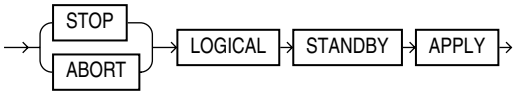
commit_switchover_clause::=



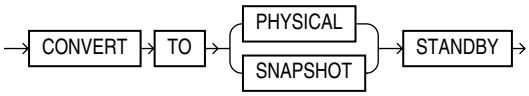
start_standby_clause::=



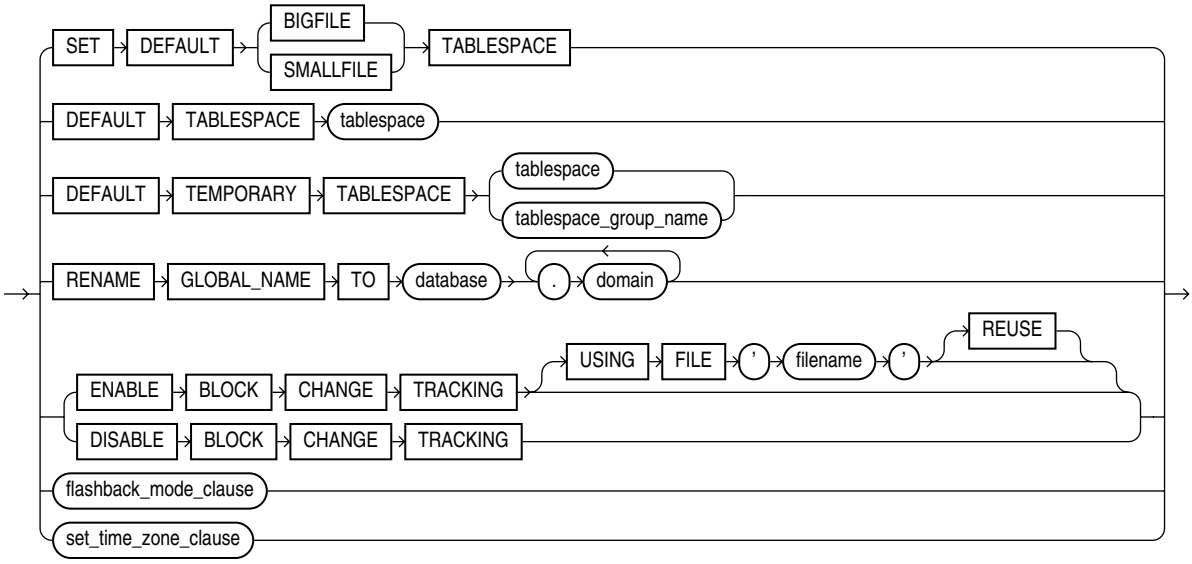
stop_standby_clause::=



convert_database_clause::=

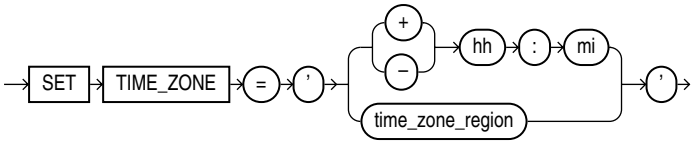


default_settings_clauses::=

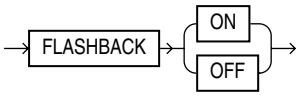


（10-17 ページの flashback_mode_clause::=、10-17 ページの set_time_zone_clause::= を参照）

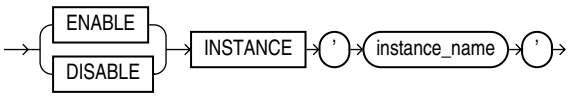
set_time_zone_clause::=



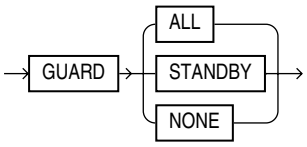
flashback_mode_clause::=



instance_clauses::=



security_clause::=



セマンティクス

database

変更するデータベースの名前を指定します。データベース名には ASCII 文字のみが使用できます。データベース名を指定しないと、初期化パラメータ DB_NAME に指定されているデータベースが変更されます。なお、データベースの制御ファイルが初期化パラメータ CONTROL_FILES に指定されている場合にのみ、そのデータベースを変更できます。データベース識別子は、Oracle Net のデータベース指定とは関係ありません。

startup_clauses

startup_clauses を使用すると、データベースをマウントおよびオープンして、アクセス可能にできます。

MOUNT 句

MOUNT 句を使用すると、データベースをマウントできます。データベースがすでにマウントされている場合、この句は使用できません。

MOUNT STANDBY DATABASE MOUNT STANDBY DATABASE を指定すると、物理スタンバイ・データベースをマウントできます。キーワード STANDBY DATABASE はオプションです。Oracle Database は、マウント対象のデータベースがプライマリ・データベースかスタンバイ・データベースかを自動的に判別します。この文の実行直後、スタンバイ・インスタンスはプライマリ・インスタンスから REDO データを受信できるようになります。

> **参照：** スタンバイ・データベースの詳細は、『Oracle Data Guard 概要および管理』を参照してください。

MOUNT CLONE DATABASE MOUNT CLONE DATABASE を指定すると、クローン・データベースをマウントできます。

OPEN 句

OPEN 句を使用すると、データベースを使用可能な状態にできます。データベースをオープンするには、マウントしておく必要があります。

OPEN を他のキーワードを指定せずに単独で指定した場合のデフォルトは、プライマリ・データベース、論理スタンバイ・データベースまたはスナップショット・スタンバイ・データベースでは OPEN READ WRITE NORESETLOGS、物理スタンバイ・データベースでは OPEN READ ONLY です。

OPEN READ WRITE OPEN READ WRITE を指定すると、読取り / 書込みモードでデータベースがオープンされ、ユーザーは REDO ログを生成できるようになります。これは、プライマリ・データベースをオープンするときのデフォルトです。この句は、物理スタンバイ・データベースに対しては指定できません。

> **参照：** 「READ ONLY | READ WRITE の例 :」（10-40 ページ）

RESETLOGS | NORESETLOGS 現行のログ順序番号を 1 にリセットし、アーカイブされていないログ（現行のログを含む）をアーカイブして、リカバリ時に適用されなかった REDO 情報を今後適用されないように破棄するかどうかを指定します。Oracle Database は、この句の設定が必要な次の状況を除き、NORESETLOGS を自動的に使用します。

- 次の場合は、RESETLOGS を指定する必要があります。
  - 不完全メディア・リカバリ、またはバックアップ制御ファイルを使用してメディア・リカバリを実行した後
  - 前回の不完全な OPEN RESETLOGS 操作の後
  - FLASHBACK DATABASE 処理の後

- 作成した制御ファイルをマウントした場合、オンライン・ログが失われたときは RESETLOGS を、失われていないときは NORESETLOGS を指定する必要があります。

UPGRADE | DOWNGRADE データベースをアップグレードまたはダウングレードする場合にかぎり、これらの OPEN 句パラメータを使用します。この句は、Oracle Database に対して、アップグレードとダウングレードにそれぞれ必要なシステム・パラメータを動的に変更するように指示します。SQL*Plus の STARTUP UPGRADE コマンドまたは STARTUP DOWNGRADE コマンドでも、同じ結果が得られます。

参照 :

- データベースをあるリリースから別のリリースにアップグレードまたはダウングレードする手順については、『Oracle Database アップグレード・ガイド』を参照してください。
- SQL*Plus の STARTUP コマンドの詳細は、『SQL*Plus ユーザーズ・ガイドおよびリファレンス』を参照してください。

OPEN READ ONLY OPEN READ ONLY を指定すると、トランザクションが読取り専用に制限され、ユーザーは REDO ログを生成できなくなります。この設定は、物理スタンバイ・データベースをオープンするときのデフォルトです。プライマリ・データベース・サイトからのアーカイブ・ログのコピー中でも、物理スタンバイ・データベースが問合せに使用可能になります。

データベースのオープンの制限事項 : データベースのオープンには、次の制限事項があります。

- データベースが現在他のインスタンスで READ WRITE モードでオープンされている場合、READ ONLY モードではオープンできません。
- データベースをリカバリする必要がある場合、READ ONLY モードではオープンできません。
- データベースが READ ONLY モードでオープンされている間は、表領域をオフラインに切り替えることはできません。ただし、データ・ファイルのオフラインとオンラインの切替えはできます。また、データベースが READ ONLY モードでオープンされている間は、オフラインのデータ・ファイルおよび表領域をリカバリできます。

参照 : 物理スタンバイ・データベースのオープン方法の詳細は、『Oracle Data Guard 概要および管理』を参照してください。

recovery_clauses

recovery_clauses を使用すると、バックアップ後の操作を指定できます。これらすべての句では、Oracle Database は、現行の制御ファイルが認識しているデータ・ファイルおよびログ・ファイルのインカネーションを使用して、データベースをリカバリします。

参照 : データベースのバックアップの詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』および 10-42 ページの「データベースのリカバリ例 :」を参照してください。

general_recovery

general_recovery 句を指定すると、データベース、スタンバイ・データベース、または指定した表領域やファイルのメディア・リカバリを制御できます。インスタンスで、データベースがマウント済（オープン状態またはクローズ状態）の場合、関連ファイルが使用中でなければ、この句を使用できます。

注意 : 全体的または部分的なデータベース・リカバリおよびログファイルのリカバリでは、デフォルトでパラレル化が有効になっています。これらの操作でのパラレル化は、それぞれの構文図に示されているように NOPARALLEL を指定して無効にできます。

一般的なデータベース・リカバリの制限事項： 一般的なリカバリには、次の制限事項があります。

- データベースがクローズ状態の場合にのみ、データベース全体をリカバリできます。
- インスタンスで、データベースが排他モードでマウントされている必要があります。
- リカバリ対象の表領域またはデータ・ファイルがオフラインの場合、データベースがオープン状態でもクローズ状態でも、表領域またはデータ・ファイルをリカバリできます。
- 共有サーバー・アーキテクチャで Oracle Database に接続している場合、メディア・リカバリは実行できません。

注意： メディアについて特別な要件がない場合は、*general_recovery* 句ではなく、SQL*Plus の RECOVER コマンドを使用することをお薦めします。

参照：

- Recovery Manager のメディア・リカバリおよびユーザー定義のメディア・リカバリの詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- SQL*Plus の RECOVER コマンドの詳細は、『SQL*Plus ユーザーズ・ガイドおよびリファレンス』を参照してください。

AUTOMATIC

AUTOMATIC を指定すると、リカバリ操作を続けるために必要な新規アーカイブ REDO ログ・ファイルの名前が自動的に生成されます。LOG_ARCHIVE_DEST_*n* パラメータが定義されている場合、最初のローカル接続先に対して有効で使用可能なパラメータがスキャンされます。その接続先は LOG_ARCHIVE_FORMAT と結合して使用され、ターゲットの REDO ログ・ファイル名が生成されます。LOG_ARCHIVE_DEST_*n* が定義されていない場合は、かわりに LOG_ARCHIVE_DEST パラメータ値が使用されます。

ファイルが検出されると、そのファイルに含まれている REDO が適用されます。ファイルが検出されない場合は、ファイル名の入力を求めるプロンプトが表示されます。この場合、提案として生成されたファイル名が表示されます。

AUTOMATIC も LOGFILE も指定しない場合、ファイル名の入力を求めるプロンプトが表示されます。この場合、提案として生成されたファイル名が表示されます。この場合、生成されたファイル名を受け入れるか、または完全修飾されたファイル名に置き換えます。生成されるファイル名とアーカイブ済のファイル名が異なることがわかっている場合、LOGFILE 句を使用することによって時間を節約できます。

FROM *'location'*

FROM *'location'* を指定すると、アーカイブ REDO ログ・ファイル・グループを読み取る位置を指定できます。*location* には、使用するオペレーティング・システムの表記規則に従って、ファイルの位置を完全に指定する必要があります。このパラメータを指定しないと、そのアーカイブ REDO ログ・グループは、初期化パラメータ LOG_ARCHIVE_DEST または LOG_ARCHIVE_DEST_1 に指定された位置にあるとみなされます。

full_database_recovery

full_database_recovery 句を指定すると、データベース全体をリカバリできます。

DATABASE DATABASE 句を指定すると、データベース全体をリカバリできます。これはデフォルトです。データベースがクローズされている場合にのみ、このオプションを使用できます。

STANDBY DATABASE STANDBY DATABASE 句を指定すると、プライマリ・データベースからコピーされたアーカイブ REDO ログ・ファイルおよび制御ファイルを使用して、物理スタンバイ・データベースを手動でリカバリできます。スタンバイ・データベースは、マウントされているがオープンされていない状態である必要があります。

この句は、オンライン・データ・ファイルのみをリカバリします。

- UNTIL を使用すると、リカバリ操作の存続期間を指定できます。
  - CANCEL は、取消しベースのリカバリを示します。RECOVER CANCEL 句を指定した ALTER DATABASE 文を発行するまで、データベース・リカバリが続行されます。
  - TIME は、時間ベースのリカバリを示します。このパラメータは、date に指定した時点までデータベースをリカバリします。date は、'YYYY-MM-DD:HH24:MI:SS' の書式の文字リテラルである必要があります。
  - CHANGE は、変更ベースのリカバリを示します。*integer* に指定したシステム変更番号の直前の、トランザクションの一貫性が保たれるところまでデータベースをリカバリします。
- 現行の制御ファイルのかわりにバックアップ制御ファイルを使用する場合、USING BACKUP CONTROLFILE を指定します。

partial_database_recovery

partial_database_recovery 句を指定すると、個々の表領域およびデータ・ファイルをリカバリできます。

TABLESPACE TABLESPACE 句を指定すると、指定した表領域のみをリカバリできます。リカバリの対象となる表領域がオフラインの場合、データベースがオープン状態でもクローズ状態でも、この句を使用できます。

> **参照 :** 「パラレル・リカバリ処理の使用例 :」(10-40 ページ)

DATAFILE DATAFILE 句を指定すると、指定したデータ・ファイルをリカバリできます。リカバリ対象のデータ・ファイルがオフラインの場合、データベースがオープン状態でもクローズ状態でも、この句を使用できます。

データ・ファイルは、名前または番号で識別できます。番号で識別した場合、*filenumber* は、V$DATAFILE 動的パフォーマンス・ビューの FILE# 列、または DBA_DATA_FILES データ・ディクショナリ・ビューの FILE_ID 列の数を表す整数です。

STANDBY TABLESPACE 以前のリリースでは、STANDBY TABLESPACE を指定すると、プライマリ・データベースおよび制御ファイルからコピーしたアーカイブ REDO ログ・ファイルを使用して、スタンバイ・データベース内に損失または破損した表領域を再構成できました。この句は現在非推奨になっています。かわりに、古い表領域をスタンバイ制御ファイルにリカバリする（それ以上の処理は行わない）場合は、文 ALTER DATABASE RECOVER MANAGED STANDBY DATABASE UNTIL CHANGE *scn* を使用します。*scn* は、V$DATABASE の CURRENT_SCN フィールドに 1 を加えた値です。

STANDBY DATAFILE 以前のリリースでは、STANDBY DATAFILE を指定すると、プライマリ・データベースおよび制御ファイルからコピーされたアーカイブ REDO ログ・ファイルを使用して、損失または破損したデータ・ファイルを物理スタンバイ・データベース内で手動で再構成できました。この句は現在非推奨になっています。かわりに、古いデータ・ファイルをスタンバイ制御ファイルにリカバリする（それ以上の処理は行わない）場合は、文 ALTER DATABASE RECOVER MANAGED STANDBY DATABASE UNTIL CHANGE *scn* を使用します。*scn* は、V$DATABASE の CURRENT_SCN フィールドに 1 を加えた値です。

LOGFILE

LOGFILE '*filename*' を指定すると、指定した REDO ログ・ファイルを使用して、メディア・リカバリを続行できます。

TEST

TEST 句を使用すると、試行リカバリを実行できます。試行リカバリは、通常のリカバリ手順に問題があった場合に有効です。REDO ストリームを見ると、続いて発生する可能性のある問題を検出することができます。試行リカバリは、通常のリカバリと同様に REDO を適用しますが、ディスクに変更を書き込みません。変更は、試行リカバリの最後にロールバックされます。

この句を使用できるのは、最後の RESETLOGS 操作以降に取ったバックアップをリストアする場合のみです。そうでない場合は、エラーが戻ります。

ALLOW ... CORRUPTION

ALLOW *integer* CORRUPTION 句を指定すると、ログ・ファイルが破損した場合に、許容リカバリの続行で許容する破損ブロックの数を指定することができます。

参照 :

- 一般的なデータベースのリカバリの詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- スタンバイ・データベースの管理リカバリの詳細は、『Oracle Data Guard 概要および管理』を参照してください。

CONTINUE

CONTINUE を指定すると、スレッドを使用禁止にするために中断されていた複数インスタンス・リカバリを再開できます。

CONTINUE DEFAULT を指定すると、他に指定されたログ・ファイルがない場合、自動的に生成された REDO ログ・ファイルを使用して、リカバリが再開されます。ファイル名の入力を求めるプロンプトが表示されないこと以外は、AUTOMATIC の指定と同じです。

CANCEL

CANCEL を指定すると、取消しベースのリカバリを終了できます。

managed_standby_recovery

managed_standby_recovery 句を使用すると、物理スタンバイ・データベースでの REDO Apply を開始および停止できます。REDO Apply は、プライマリ・データベースから受け取る REDO を継続的に適用することで、スタンバイ・データベースとプライマリ・データベースでのトランザクションの一貫性を確保します。

プライマリ・データベースは、その REDO データをスタンバイ・サイトに転送します。REDO データが、物理スタンバイ・サイトの REDO ログ・ファイルに書き込まれると、REDO Apply でそのログ・ファイルを使用できるようになります。*managed_standby_recovery* 句は、スタンバイ・インスタンスにデータベースがマウントされているか、またはスタンバイ・インスタンスが読取り専用でオープンされている場合にのみ使用できます。

管理スタンバイ・リカバリの制限事項 : この句には、10-19 ページの「general_recovery」に記載されている内容と同じ制限が適用されます。

参照 : この句の使用の詳細は、『Oracle Data Guard 概要および管理』を参照してください。

USING CURRENT LOGFILE 句 USING CURRENT LOGFILE を指定すると、**リアルタイム適用**を開始できます。これによって、スタンバイ REDO ログ・ファイルに書込みが行われると、このファイルからすぐにリカバリが実行されます。このとき、REDO ファイルを物理スタンバイ・データベースにアーカイブしておく必要はありません。

参照 : リアルタイム適用の詳細は、『Oracle Data Guard 概要および管理』を参照してください。

DISCONNECT DISCONNECT を指定すると、現行のセッションを他のタスクで使用可能にしたまま、REDO Apply をバックグラウンドで実行するよう指示できます。FROM SESSION キーワードはオプションであり、意味を明確にするためのものです。

NODELAY NODELAY 句は、プライマリ・データベースの LOG_ARCHIVE_DEST_n パラメータの DELAY 属性より優先されます。NODELAY 句を指定しない場合は、LOG_ARCHIVE_DEST_*n* 設定の DELAY 属性に従って、アーカイブ REDO ログ・ファイルの適用が遅延されます（属性が設定されている場合）。このパラメータに DELAY 属性が指定されていない場合、アーカイブ REDO ログ・ファイルは、すぐにスタンバイ・データベースに適用されます。

リアルタイム適用を USING CURRENT LOGFILE 句とともに指定すると、このスタンバイのプライマリで LOG_ARCHIVE_DEST_*n* パラメータに対して指定された DELAY 値は無視されます。デフォルトは NODELAY です。

UNTIL CHANGE 句 この句を使用すると、特定のシステム変更番号より前の REDO データをリカバリするよう REDO Apply に指示できます。

FINISH FINISH を指定すると、フェイルオーバー用に準備されている使用可能なすべての REDO データを完全に適用できます。

FINISH 句は、プライマリ・データベースに障害が発生した場合にのみ使用してください。この句は、設定済のいずれの遅延間隔より優先され、使用可能なすべての REDO が即座に適用されます。FINISH コマンドが完了すると、このデータベースをスタンバイ・データベース・ロールで実行することができなくなります。このデータベースは、ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY 文を発行してプライマリ・データベースに変換する必要があります。

CANCEL CANCEL を指定すると、REDO Apply をすぐに停止できます。REDO Apply が停止すると、すぐに制御が戻されます。

TO LOGICAL STANDBY 句 この句を使用すると、物理スタンバイ・データベースを論理スタンバイ・データベースに変換できます。

db_name データベース名を指定して、新しい論理スタンバイ・データベースを識別します。この文を発行するときにサーバー・パラメータ・ファイル（spfile）を使用する場合は、新しい論理スタンバイ・データベースの情報でファイルが適切に更新されます。spfile を使用しない場合は、データベースのシャットダウン後に DB_NAME パラメータの名前を設定するように求めるメッセージが発行されます。さらに、この句をスタンバイ・データベースで使用する前に、プライマリ・データベースで DBMS_LOGSTDBY.BUILD PL/SQL プロシージャを起動する必要もあります。

> **参照：** DBMS_LOGSTDBY.BUILD プロシージャの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

KEEP IDENTITY 論理スタンバイによって提供されるローリング・アップグレード機能を使用し、さらにプライマリ・データベースおよび物理スタンバイの元の構成に戻す場合は、この句を使用します。この句を使用して作成された論理スタンバイ・データベースでは、スイッチオーバーおよびフェイルオーバーのサポートは制限されます。そのため、この句を使用して汎用論理スタンバイ・データベースを作成しないでください。

> **参照：** ローリング・アップグレードの詳細は、『Oracle Data Guard 概要および管理』を参照してください。

非推奨になる予定の管理スタンバイ・リカバリ句

以前のリリースの構文には、次の句が記載されていました。これらの句は非推奨になる予定であり、使用する必要はありません。これらの句は使用しないことをお薦めします。

NOPARALLEL NOPARALLEL 句は、非推奨になる予定です。すべての REDO Apply はパラレル・モードで実行されます。指定されている場合、この句は無視されます。

FINISH FORCE、FINISH WAIT、FINISH NOWAIT FINISH 句のこれらのオプションの書式は、非推奨になる予定です。ここでは下位互換性のために、これらのセマンティクスについて説明します。

- FORCE を指定すると、FINISH プロセスを開始する妨げになる非アクティブな REDO 転送セッションを終了できます。
- NOWAIT を指定すると、リカバリが完了する前に、フォアグラウンド・プロセスに制御が戻ります。
- WAIT（デフォルト）を指定すると、リカバリが完了してから、フォアグラウンド・プロセスに制御が戻ります。

指定されている場合、これらの句は無視されます。ターミナル・リカバリはフォアグラウンドで実行され、常にすべての REDO 転送セッションを終了します。そのため、リカバリが完了するまでユーザーに制御は戻りません。

CANCEL IMMEDIATE、CANCEL WAIT、CANCEL NOWAIT CANCEL 句のこれらのオプションの書式は、非推奨になる予定です。ここでは下位互換性のために、これらのセマンティクスについて説明します。

- IMMEDIATE キーワードを含めると、現在の REDO ログ・ファイルが完全に適用される前に、REDO Apply を停止できます。セッション制御は、REDO Apply が実際に停止した時点で戻ります。
- NOWAIT キーワードを含めると、CANCEL 操作が終了するのを待たずに、セッション制御が戻ります。

指定されている場合、これらの句は無視されます。REDO Apply は常にすぐに取り消され、操作の完了後にのみ制御がセッションに戻されます。

BACKUP 句

この句を使用すると、データベース内のすべてのデータ・ファイルをオンライン・バックアップ・モード（ホット・バックアップ・モードともいう）にしたり、このモードから戻すことができます。

> **参照：** 個々の表領域のすべてのデータ・ファイルに対するオンライン・バックアップ・モードの設定または解除の詳細は、12-82 ページの「ALTER TABLESPACE」を参照してください。

BEGIN BACKUP 句

BEGIN BACKUP を指定すると、データベース内のすべてのデータ・ファイルをオンライン・バックアップ・モードに設定できます。データベースをマウントしてオープンしておく必要があります。また、メディア・リカバリを使用可能にする（データベースを ARCHIVELOG モードにする）必要があります。

データベースがオンライン・バックアップ・モードになっている間は、インスタンスの通常停止、個々の表領域のバックアップの開始、表領域のオフラインへの切替え、または表領域の読取り専用設定は実行できません。

この句は、オフラインまたは読取り専用の表領域のデータ・ファイルには影響しません。

END BACKUP 句

END BACKUP 指定すると、オンライン・バックアップ・モードの現行のデータベースにあるすべてのデータ・ファイルが、オンライン・バックアップ・モードから戻されます。この操作を実行するとき、データベースはマウント済（オープン状態またはクローズ状態）である必要があります。

システム障害、インスタンス障害、または SHUTDOWN ABORT 操作の後は、オンライン・バックアップ・モードのファイルがシステム・クラッシュ時のファイルと一致するかどうかは識別されません。ファイルに一貫性がある場合、個々のデータ・ファイル、またはすべてのデータ・ファイルをオンライン・バックアップ・モードから戻すことができます。これによって、起動時にファイルのメディア・リカバリを回避できます。

- ALTER DATABASE DATAFILE ... END BACKUP 文を使用すると、個々のデータ・ファイルをオンライン・バックアップ・モードから戻すことができます。10-25 ページの「database_file_clauses」を参照してください。
- ALTER TABLESPACE ... END BACKUP 文を使用すると、表領域のすべてのデータ・ファイルをオンライン・バックアップ・モードから戻すことができます。

database_file_clauses

database_file_clauses を指定すると、データ・ファイルおよび一時ファイルを変更できます。インスタンスで、データベースがマウント済（オープン状態またはクローズ状態）の場合、関連ファイルが使用中でなければ、次の句はどれでも使用できます。

RENAME FILE 句

RENAME FILE 句を使用すると、データ・ファイル、一時ファイルまたは REDO ログ・ファイル・メンバーの名前を変更できます。この句を指定する前に、オペレーティング・システムのファイル名の表記規則に従って、各ファイル名を指定してください。

- データ・ファイルまたは一時ファイルにこの句を使用するには、データベースをマウントしておく必要があります。データベースはオープンしていてもかまいませんが、名前を変更するデータ・ファイルまたは一時ファイルはオフラインにしておく必要があります。また、最初にファイル・システム上のファイルを新しい名前に変更する必要があります。
- ログ・ファイル用にこの句を使用するには、データベースはマウントされているがオープンされていない状態である必要があります。
- ブロック・チェンジ・トラッキングを使用可能にしている場合、この句を使用してブロック・チェンジ・トラッキング・ファイルの名前を変更できます。ブロック・チェンジ・トラッキング・ファイルの名前を変更する場合、データベースはマウントされているがオープンされていない状態である必要があります。

この句は、制御ファイルのファイル名のみを変更します。オペレーティング・システムのファイルの名前は、実際には変更されません。オペレーティング・システムのファイル名は存在したままですが、使用されません。

参照 :

- データ・ファイルと一時ファイルのリカバリの詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- 10-41 ページの「ログ・ファイル・メンバーの名前の変更例 :」および 10-42 ページの「一時ファイルの操作例 :」を参照してください。

create_datafile_clause

CREATE DATAFILE 句を使用すると、元のデータ・ファイルのかわりに新しい空のデータ・ファイルを作成できます。この句を使用すると、バックアップを取らずに失われたデータ・ファイルを再作成できます。*filename* または *filenumber* には、データベースの一部であるファイルまたは以前一部であったファイルを指定します。番号で識別した場合、*filenumber* は、V$DATAFILE 動的パフォーマンス・ビューの FILE# 列、または DBA_DATA_FILES データ・ディクショナリ・ビューの FILE_ID 列の数を表す整数です。

- AS NEW を指定すると、データ・ファイル用のデフォルトのファイル・システム位置に、システムが生成するファイル名で、置き換えるファイルと同じサイズの Oracle Managed Files のデータ・ファイルが作成されます。

- AS *file_specification* を指定すると、新しいデータ・ファイルにファイル名（およびオプションでサイズ）を割り当てることができます。オペレーティング・システムのファイル・システム内の標準データ・ファイルと一時ファイル、または自動ストレージ管理ディスク・グループのファイルを指定するには、*file_specification* の *datafile_tempfile_spec* 書式（8-26 ページの「file_specification」を参照）を使用します。

元のファイル（*filename* または *filenumber*）が既存の Oracle Managed Files のデータ・ファイルの場合、Oracle Database は新しいファイルを作成した後、元のファイルを削除しようとします。元のファイルが既存のユーザー管理データ・ファイルの場合、Oracle Database は元のファイルを削除しません。

AS 句を指定しない場合、Oracle Database によって、*filename* または *filenumber* に指定したファイルと同じ名前およびサイズのファイルが新しく作成されます。

リカバリ時には、元のデータ・ファイルの作成後に書き込まれたアーカイブ REDO ログを、失われたデータ・ファイルにかわる新しい空のデータ・ファイルに適用する必要があります。

新しいファイルは、元のファイルの作成時と同じ状態で作成されます。新しいファイルを元のファイルが失われた時点の状態に戻すには、メディア・リカバリを行ってください。

データ・ファイルの新規作成の制限事項： 新規データ・ファイルの作成には、次の制限事項があります。

- SYSTEM 表領域の最初のデータ・ファイルに基づいて新しいファイルを作成することはできません。
- この CREATE DATAFILE 句には、*datafile_tempfile_spec* の *autoextend_clause* は指定できません。

参照：

- 新しいデータ・ファイル名を指定しない場合にこの句によって戻される結果の詳細は、14-13 ページの「CREATE DATABASE」の「DATAFILE 句」を参照してください。
- ファイル仕様（*datafile_tempfile_spec*）の詳細は、8-26 ページの「file_specification」および 10-41 ページの「新しいデータ・ファイルの作成例 :」を参照してください。

alter_datafile_clause

DATAFILE 句を使用すると、名前または番号で識別するファイルを操作できます。番号で識別した場合、*filenumber* は、V$DATAFILE 動的パフォーマンス・ビューの FILE# 列、または DBA_DATA_FILES データ・ディクショナリ・ビューの FILE_ID 列の数を表す整数です。DATAFILE 句を使用すると、次のようにデータベース・ファイルを変更できます。

ONLINE ONLINE を指定すると、データ・ファイルをオンラインにできます。

OFFLINE OFFLINE を指定すると、データ・ファイルをオフラインにできます。データベースがオープンされている場合、データ・ファイルをオンラインに戻す前に、データ・ファイルのメディア・リカバリを行う必要があります。これは、データ・ファイルがオフラインになる前に、チェックポイントが実行されないためです。

FOR DROP データベースが NOARCHIVELOG モードの場合は、FOR DROP 句を指定して、データ・ファイルをオフラインにする必要があります。ただし、この句は、データベースからデータ・ファイルを削除しません。そのため、オペレーティング・システムのコマンドを使用するか、またはデータ・ファイルが存在する表領域を削除する必要があります。削除するまで、データ・ファイルは RECOVER または OFFLINE の状態でデータ・ディクショナリに残ります。

データベースが ARCHIVELOG モードの場合は、FOR DROP 句は無視されます。

RESIZE RESIZE を指定すると、データ・ファイルのサイズを、指定した絶対サイズ（バイト単位）まで増やしたり減らすことができます。デフォルト値はないため、必ずサイズを指定してください。

増やしたサイズに対して十分なディスク領域がない場合、または減らしたサイズを超えるデータがファイルに含まれる場合、エラー・メッセージが戻されます。

参照：「データ・ファイルのサイズの変更例 :」（10-42 ページ）

END BACKUP END BACKUP を指定すると、データ・ファイルがオンライン・バックアップ・モードから戻されます。END BACKUP 句の詳細は、ALTER DATABASE 構文の最上位で説明しています。10-24 ページの「END BACKUP 句」を参照してください。

alter_tempfile_clause

TEMPFILE 句を使用すると、一時データ・ファイルのサイズを変更できます。また、*autoextend_clause* を指定すると、永続データ・ファイルと同じ効果を持たせることができます。データベースは、オープンされている必要があります。一時ファイルは、名前または番号で識別します。番号で識別した場合、*filenumber* は、V$TEMPFILE 動的パフォーマンス・ビューの FILE# 列の数を表す整数です。

注意： オペレーティング・システムによっては、一時ファイルのブロックが実際にアクセスされるまで、一時ファイル用の領域が割り当てられない場合があります。領域の割当ての遅延のため、一時ファイルの作成およびサイズ変更が速くなります。ただし、後で一時ファイルが使用されるときに、十分なディスク領域を使用可能にする必要があります。発生する可能性がある問題を回避するには、一時ファイルを作成またはサイズ変更する前に、ディスク領域が、新しく作成する一時ファイルまたはサイズ変更後の一時ファイルのサイズより大きいことを確認してください。ディスク領域に余裕を持たせることによって、関連のない操作が原因で増加が予想されるディスク使用量に対応できます。十分な領域があることを確認した後で、作成またはサイズ変更操作を実行してください。

DROP DROP を指定すると、データベースから一時ファイルを削除できます。表領域はそのまま残ります。

INCLUDING DATAFILES を指定すると、関連付けられたオペレーティング・システム・ファイルは削除され、削除された各ファイルのメッセージがアラート・ログに書き込まれます。同じ結果は、ALTER TABLESPACE ... DROP TEMPFILE 文を使用しても得られます。詳細は、12-88 ページの「ALTER TABLESPACE」の「DROP 句」を参照してください。

autoextend_clause

autoextend_clause を使用すると、新規または既存のデータ・ファイルまたは一時ファイルの自動拡張を使用可能または使用禁止にできます。この句の詳細は、8-26 ページの「file_specification」を参照してください。

logfile_clauses

logfile_clauses を指定すると、ログ・ファイルを追加、削除または変更できます。

ARCHIVELOG

ARCHIVELOG を指定すると、REDO ログ・ファイル・グループを再利用する前に、グループの内容をアーカイブできます。このモードでは、メディア・リカバリができるようになります。この句は、インスタンスを正常に停止したか、またはエラーなしで即時停止した後に、再起動してデータベースをマウントした後にのみ使用します。

MANUAL MANUAL を指定すると、Oracle Database によって作成された REDO ログ・ファイルのアーカイブを、ユーザーが制御することができます。この句は、テープに直接アーカイブするユーザーなどに対して、下位互換性を保つために提供されています。MANUAL を指定する場合、次のことに注意します。

- ログの切替えが発生した場合、Oracle Database は REDO ログ・ファイルをアーカイブしません。これは手動で行う必要があります。
- アーカイブ・ログの宛先として、スタンバイ・データベースを指定することはできません。したがって、データベースを、MAXIMUM PROTECTION または MAXIMUM AVAILABILITY スタンバイ保護モードにすることはできません。

この句を指定しない場合、REDO ログ・ファイルは、初期化パラメータ LOG_ARCHIVE_DEST_*n* に指定された宛先に自動的にアーカイブされます。

NOARCHIVELOG

REDO ログ・ファイル・グループを再利用する前に、内容をアーカイブする必要がない場合は、NOARCHIVELOG を指定します。このモードでは、メディア障害後のリカバリはできません。インスタンスでデータベースがマウントされているがオープンされていない場合にのみ、この句を使用します。

[NO] FORCE LOGGING

この句を使用すると、データベースを FORCE LOGGING モードにしたり、このモードから戻すことができます。データベースは、マウントまたはオープンされている必要があります。

FORCE LOGGING モードでは、一時表領域および一時セグメントの変更を除くデータベースのすべての変更が記録されます。この設定は、各表領域で指定する NOLOGGING または FORCE LOGGING 設定、および各データベース・オブジェクトで指定する NOLOGGING 設定より優先され、これらの設定には影響されません。

FORCE LOGGING を指定すると、Oracle Database は、まだ記録されていない実行中のすべての操作が終了するまで待機します。

> **参照：** FORCE LOGGING モードの使用の詳細は、『Oracle Database 管理者ガイド』を参照してください。

RENAME FILE 句

ログ・ファイルに対するこの句の機能は、データ・ファイルおよび一時ファイルの場合と同じです。10-25 ページの「RENAME FILE 句」を参照してください。

CLEAR LOGFILE 句

CLEAR LOGFILE 句を使用すると、オンライン REDO ログを再初期化できます。REDO ログをアーカイブしないオプションもあります。CLEAR LOGFILE は、REDO ログの追加および削除と似ていますが、スレッドに 2 つ以外ログがない場合や、クローズ状態のスレッドの現行の REDO ログ・ファイルに対しても発行できます。

スタンバイ・データベースでは、STANDBY_FILE_MANAGEMENT 初期化パラメータが AUTO に設定されており、いずれかのログ・ファイルが Oracle Managed Files である場合、Oracle Database によって、制御ファイル内のファイルと同数の Oracle Managed Files のログ・ファイルが作成されます。ログ・ファイル・メンバーは、ログ・ファイルの現行のデフォルト宛先に格納されます。

- アーカイブされていない REDO ログを再利用する場合は、UNARCHIVED を指定する必要があります。

> **注意：** リカバリのために REDO ログが必要な場合に UNARCHIVED を指定すると、バックアップが使用できなくなります。

- ARCHIVELOG モードのデータベースでデータ・ファイルをオフラインにする（DROP キーワードを使用せずに ALTER DATABASE ... DATAFILE OFFLINE を指定する）場合、およびそのデータ・ファイルをオンラインに戻す前に、消去するアーカイブされていないログがデータ・ファイルのリカバリに必要な場合は、UNRECOVERABLE DATAFILE を指定する必要があります。この場合、CLEAR LOGFILE 文の完了後にデータ・ファイルおよび表領域全体を削除する必要があります。

  メディア・リカバリで必要なログを、CLEAR LOGFILE を使用して消去しないでください。データベースのチェックポイント後の REDO を含むログを消去する必要がある場合は、不完全メディア・リカバリを最初に実行する必要があります。オープンしているスレッドの現行の REDO ログは消去できます。クローズしているスレッドの現行のログは、そのスレッド内でログを切り替えれば消去できます。

  CLEAR LOGFILE 文が、システム障害またはインスタンス障害による割込みを受けると、データベースがハングする場合があります。このような状況になった場合は、データベースの再起動後に、この文を再発行します。ログ・グループのあるメンバーにアクセスしようとした際、I/O エラーによる障害が発生した場合は、そのメンバーを削除して他のメンバーを追加できます。

  参照 : 「ログ・ファイルの消去例 :」（10-42 ページ）

add_logfile_clauses

この句を使用すると、データベースに REDO ログ・ファイル・グループを追加したり、既存の REDO ログ・ファイル・グループに新しいメンバーを追加することができます。

ADD LOGFILE 句 ADD LOGFILE 句を使用すると、指定したスレッドまたはインスタンスに 1 つ以上の REDO ログ・ファイル・グループが追加され、そのスレッドが割り当てられているインスタンスで、そのグループを使用できるようになります。

ログ・ファイルがオンライン用、スタンバイ・データベース用のどちらに設計されたかを確認するには、V$LOGFILE 動的パフォーマンス・ビューの TYPE 列を問い合せます。

参照 :

- 新しいログ・ファイル・グループ名を指定しない場合にこの句によって Oracle Managed Files に戻される結果の詳細は、14-20 ページの「CREATE DATABASE」の「LOGFILE 句」を参照してください。
- 「REDO ログ・ファイル・グループの追加例 :」（10-40 ページ）
- V$LOGFILE の詳細は、『Oracle Database リファレンス』を参照してください。

INSTANCE INSTANCE 句は、パラレル・モードの Real Application Clusters で Oracle Database を使用している場合にのみ適用可能です。ログ・ファイルを追加するインスタンス名を指定します。インスタンス名は最大 80 文字の文字列です。指定したインスタンスにマップするスレッドは Oracle Database によって自動的に使用されます。指定されたインスタンスにスレッドがマップされていない場合、Oracle Database はマップされていない使用可能なスレッドを取得して、そのインスタンスに割り当てます。この句も THREAD 句も指定しない場合、Oracle Database は、現行のインスタンスを指定した場合と同じコマンドを実行します。指定されたインスタンスにマップされた現行のスレッドが存在せず、マップされていない使用可能なスレッドも存在しない場合、エラーが戻されます。

GROUP GROUP 句を指定すると、すべてのスレッドのすべてのグループ間で、REDO ログ・ファイル・グループを一意に識別できます。この値は、1 から CREATE DATABASE 文の MAXLOGFILES に指定された値までの範囲で指定できます。同一の GROUP 値を持つ REDO ログ・ファイル・グループは、複数追加できません。このパラメータを指定しない場合、値が自動的に生成されます。REDO ログ・ファイル・グループの GROUP 値は、動的パフォーマンス・ビュー V$LOG で確認できます。

redo_log_file_spec 各 *redo_log_file_spec* には、1 つ以上のメンバー（コピー）を含む REDO ログ・ファイル・グループを指定します。新しいログ・ファイルにファイル名を指定しない場合、14-20 ページの「CREATE DATABASE」の「LOGFILE 句」に示すルールに従って、Oracle Managed Files が作成されます。

参照：

- 「file_specification」（8-26 ページ）
- 動的パフォーマンス・ビューの詳細は、『Oracle Database リファレンス』を参照してください。

ADD [STANDBY] LOGFILE MEMBER 句 ADD LOGFILE MEMBER 句を使用すると、既存の REDO ログ・ファイル・グループに新しいメンバーを追加できます。新しいメンバーをそれぞれ *'filename'* に指定します。すでにファイルが存在する場合、追加するメンバーはグループ内の他のメンバーと同じサイズである必要があり、REUSE を指定する必要があります。ファイルが存在しない場合、適切なサイズのファイルが作成されます。メディア障害によってグループのすべてのメンバーを失った場合は、そのグループにメンバーを追加することはできません。

STANDBY を指定すると、ログ・ファイル・メンバーが物理スタンバイ・データベースのみで使用可能になります。ただし、このキーワードは必須ではありません。グループ *integer* がスタンバイ・データベース用に追加された場合は、すべてのメンバーも同様にスタンバイ・データベースのみに使用されます。

次のいずれかの方法で、既存の REDO ログ・ファイル・グループを指定できます。

GROUP *integer* integer には REDO ログ・ファイル・グループを識別する GROUP パラメータの値を指定します。

filename REDO ログ・ファイル・グループのすべてのメンバーを指定します。ご使用のオペレーティング・システムの表記規則に従って、ファイル名を完全に指定する必要があります。

参照：

- 新しいログ・ファイル・グループ名を指定しない場合にこの句によって Oracle Managed Files に戻される結果の詳細は、14-20 ページの「CREATE DATABASE」の「LOGFILE 句」を参照してください。
- 「REDO ログ・ファイル・グループ・メンバーの追加例 :」（10-41 ページ）

drop_logfile_clauses

この句を使用すると、REDO ログ・ファイル・グループまたは REDO ログ・ファイル・メンバーを削除できます。

DROP LOGFILE 句 DROP LOGFILE 句を使用すると、REDO ログ・ファイル・グループのすべてのメンバーを削除できます。この句を使用して Oracle Managed Files を削除する場合、ディスクからもすべてのログ・ファイル・メンバーが削除されます。ADD LOGFILE MEMBER 句と同様に、REDO ログ・ファイル・グループを指定します。

- 現行のログ・ファイル・グループを削除する場合、最初に ALTER SYSTEM SWITCH LOGFILE 文を発行する必要があります。
- アーカイブが必要な REDO ログ・ファイル・グループは、削除できません。
- REDO ログ・ファイル・グループを削除すると、その REDO スレッドの REDO ログ・ファイル・グループが 2 つ未満になる場合は、削除できません。

参照： 11-52 ページの「ALTER SYSTEM」および 10-41 ページの「ログ・ファイル・メンバーの削除例 :」を参照してください。

DROP LOGFILE MEMBER 句 DROP LOGFILE MEMBER 句を使用すると、1 つ以上の REDO ログ・ファイル・メンバーを削除できます。各 *'filename'* には、ご使用のオペレーティング・システムのファイル名の表記規則に従って、メンバーを完全に指定する必要があります。

- 現行のログのログ・ファイルを削除する場合、最初に ALTER SYSTEM SWITCH LOGFILE 文を発行する必要があります。詳細は、11-52 ページの「ALTER SYSTEM」を参照してください。
- この句では、有効なデータを含む REDO ログ・ファイル・グループのすべてのメンバーを削除できません。このような操作には、DROP LOGFILE 句を使用してください。

参照 : 「ログ・ファイル・メンバーの削除例 :」（10-41 ページ）

supplemental_db_logging

この句を使用すると、ログ・ストリームへのサプリメンタル・データの追加を実行または停止するように Oracle Database に指示できます。

ADD SUPPLEMENTAL LOG 句 ADD SUPPLEMENTAL LOG DATA を指定すると、**最小サプリメンタル・ロギング**を使用可能にできます。最小サプリメンタル・ロギングに加え、列データ・ロギングを使用可能にする場合、ADD SUPPLEMENTAL LOG *supplemental_id_key_clause* を指定します。PL/SQL コールのサプリメンタル・ロギングを使用可能にする場合、ADD SUPPLEMENTAL LOG *supplemental_plsql_clause* を指定します。デフォルトでは、最小サプリメンタル・ロギングもサプリメンタル・ロギングも使用できません。

最小サプリメンタル・ロギングによって、LogMiner（および LogMiner の技術に基づいた製品）には、連鎖行および様々な記憶域構成（クラスタ表など）をサポートするために十分な情報が確保されます。

あるデータベースで生成された REDO が、別のデータベースでの変更の基となる（マイニングおよび適用される）場合（論理スタンバイ・データベースなど）、ROWID ではなく列データを使用して、影響を受ける行を識別する必要があります。この場合、*supplemental_id_key_clause* を指定する必要があります。

サプリメンタル・ロギングが使用可能になっているかどうかを確認するには、V$DATABASE ビューの適切な列を問い合せます。

データベースのオープン時にこの文を発行できます。ただし、パフォーマンスに影響するカーソル・キャッシュのすべての DML カーソルが、キャッシュが再移入されるまで無効になります。

supplemental_id_clause の詳細は、16-29 ページの「CREATE TABLE」の「supplemental_id_key_clause」を参照してください。

参照 :

論理スタンバイ・データベースをサポートするためのプライマリ・データベースでのサプリメンタル・ロギングの詳細は、『Oracle Data Guard 概要および管理』を参照してください。

supplemental_db_logging 句の構文の使用例は、『Oracle Database ユーティリティ』を参照してください。

DROP SUPPLEMENTAL LOG 句

この句を使用すると、サプリメンタル・ロギングを停止できます。

- DROP SUPPLEMENTAL LOG DATA を指定すると、更新操作が発生するたびに REDO ログ・ストリームに追加の最小ログ情報が置かれることを停止できます。Oracle Database が *supplemental_id_key_clause* で指定された列データ・サプリメンタル・ロギングを実行している場合、まず DROP SUPPLEMENTAL LOG *supplemental_id_key_clause* で列データ・サプリメンタル・ロギングを停止してから、この句を指定する必要があります。

- DROP SUPPLEMENTAL LOG *supplemental_id_key_clause* を指定すると、システム生成の補助ログ・グループの一部またはすべてを削除できます。削除する補助ログ・グループが *supplemental_id_key_clause* を使用して追加された場合、この句を指定する必要があります。
- PL/SQL コールのサプリメンタル・ロギングを使用禁止にする場合は、DROP SUPPLEMENTAL LOG *supplemental_plsql_clause* を指定します。

> **参照：** サプリメンタル・ロギングの詳細は、『Oracle Data Guard 概要および管理』を参照してください。

controlfile_clauses

controlfile_clauses を指定すると、制御ファイルを作成またはバックアップできます。

CREATE STANDBY CONTROLFILE 句

CREATE STANDBY CONTROLFILE 句を使用すると、物理または論理スタンバイ・データベースを管理するための制御ファイルを作成できます。ファイルがすでに存在している場合は、REUSE を指定してください。

> **参照：** 『Oracle Data Guard 概要および管理』

BACKUP CONTROLFILE 句

BACKUP CONTROLFILE 句を使用すると、現行の制御ファイルのバックアップを取ることができます。この句を使用するとき、データベースをオープンまたはマウントしておく必要があります。

TO '*filename*' この句を使用して制御ファイルのバイナリ・バックアップを指定します。ご使用のオペレーティング・システムの表記規則に従って、ファイル名を完全に指定する必要があります。指定したファイルがすでに存在している場合は、REUSE を指定します。

バイナリ・バックアップには、TO TRACE を指定した場合に取得されない情報が含まれています。アーカイブ・ログ履歴、読取り専用およびオフライン表領域のオフライン範囲、バックアップ・セットおよびコピー（Recovery Manager を使用する場合）などです。COMPATIBLE 初期化パラメータが 10.2 以上の場合、バイナリ制御ファイルのバックアップには一時ファイルのエントリが含まれます。

TO TRACE TO TRACE を指定すると、Oracle Database では、制御ファイルの物理バックアップが作成されるかわりに、トレース・ファイルに SQL 文が書き込まれます。トレース・ファイルは、DIAGNOSTIC_DEST 初期化パラメータによって決まるサブディレクトリに格納されます。CREATE CONTROLFILE 文が書き込まれたトレース・ファイルの名前と場所は、アラート・ログで確認できます。V$DIAG_INFO 動的パフォーマンス・ビューの NAME および VALUE 列を問い合せて、トレース・ファイルのディレクトリを検索することもできます。トレース・ファイルに記述された SQL 文を使用すると、データベースの起動、制御ファイルの再作成、データベースのリカバリやオープンなどの操作を、作成した制御ファイルに基づいて正しく実行できます。ブロック・チェンジ・トラッキングを使用可能にしているときに ALTER DATABASE BACKUP CONTROLFILE TO TRACE 文を発行すると、生成されるスクリプトには、ブロック・チェンジ・トラッキングを再度使用可能にするコマンドが記述されます。

> **参照：** アラート・ログの表示方法の詳細は、『Oracle Database 管理者ガイド』を参照してください。

この文は暗黙的な ALTER DATABASE REGISTER LOGFILE 文を発行します。アーカイブ・ログ・ファイルが現行のアーカイブ・ログの宛先に存在する場合、この文によって、インカネーション・レコードが作成されます。

制御ファイルのコピーがすべて失われた場合（または、制御ファイルのサイズを変更する場合）は、トレース・ファイルからスクリプト・ファイルに文をコピーし、必要に応じて文を編集できます。

- AS *filename* を指定すると、標準トレース・ファイルではなく *filename* というファイルにスクリプトを格納できます。
- REUSE を指定すると、*filename* と呼ばれる既存ファイルを上書きできます。
- RESETLOGS は、データベースの起動用としてトレース・ファイルに書き込まれた SQL 文が、ALTER DATABASE OPEN RESETLOGS であることを示します。オンライン・ログが使用不可能な場合のみ、この設定は有効です。
- NORESETLOGS は、データベースの起動用としてトレース・ファイルに書き込まれた SQL 文が、ALTER DATABASE OPEN NORESETLOGS であることを示します。オンライン・ログが使用可能な場合のみ、この設定は有効です。

オンライン・ログの今後の状態が予測できない場合は、RESETLOGS も NORESETLOGS も指定しないでください。この場合、両方のバージョンのスクリプトがトレース・ファイルに入れられるため、スクリプトが必要になった時点で、適切なバージョンを選択できます。

standby_database_clauses

この句を使用すると、スタンバイ・データベースをアクティブにする、または保護モードと非保護モードのいずれかを指定することができます。

> **参照：** 物理および論理スタンバイ・データベース、およびスタンバイ・データベースの管理と使用の詳細は、『Oracle Data Guard 概要および管理』を参照してください。

activate_standby_db_clause

ACTIVATE STANDBY DATABASE 句を使用すると、スタンバイ・データベースをプライマリ・データベースに変換できます。

> **注意：** このコマンドを使用する前に、『Oracle Data Guard 概要および管理』を参照して使用方法を確認してください。

PHYSICAL PHYSICAL を指定すると、物理スタンバイ・データベースをアクティブにできます。これはデフォルトです。

LOGICAL LOGICAL を指定すると、論理スタンバイ・データベースをアクティブにできます。論理スタンバイ・データベースが複数ある場合は、すべてのスタンバイ・システムで同じログ・データが使用可能であることを確認する必要があります。

FINISH APPLY この句は、論理スタンバイ・データベースのみに適用されます。この句は、**ターミナル適用**を開始するために使用します（ターミナル適用を実行すると、残りの REDO が適用され、論理スタンバイ・データベースがプライマリ・データベースと同じ状態になります）。ターミナル適用が完了すると、データベースは、論理スタンバイからプライマリ・データベースへのスイッチオーバーを完了します。

データの消失があってもデータベースをすぐにリストアする必要がある場合は、この句を指定しないでください。データベースは、ターミナル適用を行わずに、論理スタンバイからプライマリ・データベースへのスイッチオーバーを実行します。

maximize_standby_db_clause

この句を使用すると、データベース環境でのデータの保護レベルを指定できます。この句は、プライマリ・データベースから指定します。この場合、プライマリ・データベースはマウントされている状態で、オープンされていない状態である必要があります。

注意： PROTECTED および UNPROTECTED キーワードは、意味を明確にするために変更されていますが、引き続きサポートされています。PROTECTED は、TO MAXIMIZE PROTECTION と同等です。UNPROTECTED は、TO MAXIMIZE PERFORMANCE と同等です。

TO MAXIMIZE PROTECTION この設定は、**最大保護モード**を確立し、最高レベルのデータ保護を実現します。トランザクションのリカバリに必要なすべてのデータが、SYNC ログ転送モードを使用するように構成された 1 つ以上の物理スタンバイ・データベースに書き込まれるまで、そのトランザクションはコミットされません。1 つ以上のスタンバイ・データベースに REDO レコードを書き込めない場合、プライマリ・データベースは停止します。このモードでは、データ非消失は保証されますが、プライマリ・データベースのパフォーマンスおよび可用性に最大の影響を与える可能性があります。

TO MAXIMIZE AVAILABILITY この設定は、**最大可用性モード**を確立し、2 番目に高いレベルのデータ保護を実現します。トランザクションのリカバリに必要なすべてのデータが、SYNC ログ転送モードを使用するように構成された 1 つ以上の物理または論理スタンバイ・データベースに書き込まれるまで、そのトランザクションはコミットされません。最大保護モードとは異なり、1 つ以上のスタンバイ・データベースに REDO レコードを書き込めない場合でも、プライマリ・データベースは停止しません。障害が修正され、スタンバイ・データベースがプライマリ・データベースと同一になるまで、データ保護レベルは最大パフォーマンス・モードまで下げられます。最大パフォーマンス・モードのプライマリ・データベースに障害が発生しないかぎり、このモードによってデータ非消失が保証されます。最大可用性モードは、プライマリ・データベースの可用性に影響を与えずに、最大レベルのデータ保護を実現します。

TO MAXIMIZE PERFORMANCE この設定は、**最大パフォーマンス・モード**を確立します。これがデフォルトの設定です。トランザクションのリカバリに必要なデータが、スタンバイ・データベースにすべて書き込まれる前に、トランザクションはコミットされます。そのため、プライマリ・データベースに障害が発生し、プライマリ・データベースから REDO レコードをリカバリできない場合は、一部のトランザクションが失われることがあります。このモードは、プライマリ・データベースのパフォーマンスに影響を与えずに、最大レベルのデータ保護を実現します。

データベースの現行モードを確認するには、V$DATABASE 動的パフォーマンス・ビューの PROTECTION_MODE 列を問い合せます。

参照： スタンバイ・データベースのこれらの設定については、『Oracle Data Guard 概要および管理』を参照してください。

register_logfile_clause

スタンバイ・データベースで REGISTER LOGFILE 句を指定すると、障害が発生したプライマリ・データベースからログ・ファイルを手動で登録することができます。オペレーティング・システムのファイル・システム内の標準 REDO ログ・ファイル、または自動ストレージ管理ディスク・グループの REDO ログ・ファイルをリスト表示するには、*file_specification* の *redo_log_file_spec* 書式（8-26 ページの「file_specification」を参照）を使用します。

ログ・ファイルのインカーネーションが不明である場合、REGISTER LOGFILE 句は、インカネーション・レコードを V$DATABASE_INCARNATION ビューに追加します。新しく登録されたログ・ファイルのインカネーションが現行の RECOVERY_TARGET_INCARNATION# よりも高い RESETLOGS_TIME を保持している場合、REGISTER LOGFILE 句は、新しく追加されたインカネーション・レコードに対応するように RECOVERY_TARGET_INCARNATION# も変更します。

OR REPLACE OR REPLACE を指定すると、たとえば位置やファイル指定が変更された場合に、スタンバイ・データベースの既存のアーカイブ・ログ・エントリが更新されます。エントリのシステム変更番号は、正確に一致している必要があります。また、元のエントリは、管理スタンバイ・ログの送信メカニズムによって作成される必要があります。

FOR *logminer_session_name* この句は、Streams 環境で便利です。この句によって、指定した 1 つの LogMiner セッションで、ログ・ファイルを登録できます。

commit_switchover_clause

この句を使用すると、Data Guard 構成でのデータベース・ロールの遷移を実行できます。

> **注意：** このコマンドを使用する前に、『Oracle Data Guard 概要および管理』を参照して使用方法を確認してください。

PREPARE TO SWITCHOVER この句は、プライマリ・データベースが論理スタンバイ・データベースになる準備または論理スタンバイ・データベースがプライマリ・データベースになる準備を行います。

- プライマリ・データベースで PREPARE TO SWITCHOVER TO LOGICAL STANDBY を指定します。
- 論理スタンバイ・データベースで PREPARE TO SWITCHOVER TO PRIMARY DATABASE を指定します。

COMMIT TO SWITCHOVER この句は、プライマリ・データベースをスタンバイ・データベース・ロールに切り替えるか、またはスタンバイ・データベースをプライマリ・データベース・ロールに切り替えます。

- プライマリ・データベースで COMMIT TO SWITCHOVER TO PHYSICAL STANDBY または COMMIT TO SWITCHOVER TO LOGICAL STANDBY を指定します。
- スタンバイ・データベースで COMMIT TO SWITCHOVER TO PRIMARY DATABASE を指定します。

PHYSICAL この句は常にオプションです。この句を COMMIT TO SWITCHOVER TO PRIMARY 句とともに使用することは、非推奨になりました。

LOGICAL この句は、プライマリ・データベースが論理スタンバイ・データベース・ロールに遷移されるときに、PREPARE TO SWITCHOVER 句または COMMIT TO SWITCHOVER 句とともに指定します。この句を COMMIT TO SWITCHOVER TO PRIMARY 句とともに使用することは、非推奨になりました。

WITH SESSION SHUTDOWN この句は、データベース・ロールの遷移を実行する前に、すべてのデータベース・セッションをクローズし、コミットされていないトランザクションをロールバックします。

WITHOUT SESSION SHUTDOWN この句は、データベース・セッションが存在する場合、要求されたロールの遷移が発生しないようにします。これはデフォルトです。

WAIT この句を指定すると、ロールの遷移が完了するのを待機してから制御がユーザーに戻されます。

NOWAIT この句を指定すると、ロールの遷移が完了するのを待機しないで制御がユーザーに戻されます。これはデフォルトです。

CANCEL この句を指定すると、以前に指定した PREPARE TO SWITCHOVER 文の影響が無効にされます。

> **参照：** プライマリ・データベースとスタンバイ・データベース間のスイッチオーバーの詳細は、『Oracle Data Guard 概要および管理』を参照してください。

start_standby_clause

START LOGICAL STANDBY APPLY 句を指定すると、論理スタンバイ・データベースへの REDO ログの適用を開始できます。この句では、PL/SQL コールのロギング以外に、主キー、一意索引および一意制約のサプリメンタル・ロギングが使用可能になります。

- 現在のスタンバイ REDO ログ・ファイルの REDO データを適用する場合は、IMMEDIATE を指定します。
- この適用において遅延を無視するように Oracle Database に指示する場合は、NODELAY を指定します。これは、プライマリ・データベースが存在せず、PL/SQL コールの実行が必要になる場合に便利です。
- 初めてスタンバイ・データベースにログを適用する場合は、INITIAL を指定します。
- 次の 2 つの状況では、
- NEW PRIMARY 句が必要となります。
    - 論理スタンバイへのフェイルオーバー時に、フェイルオーバー操作に加わっていない論理スタンバイ、および論理スタンバイ・データベースとして回復された後の古いデータベースでこの句を指定します。
    - （準備されていないスイッチオーバー操作を使用する）論理スタンバイ・データベースを使用してローリング・アップグレードを実行している場合は、元のプライマリ・データベースが新しいデータベース・ソフトウェアにアップグレードされた後でこの句を指定します。
- イベント表の最後のトランザクションをスキップして適用を再開する場合は、SKIP FAILED [TRANSACTION] を指定します。
- スタンバイ REDO ログ・ファイル情報をアーカイブ・ログに強制処理する場合は、FINISH を指定します。プライマリ・データベースが使用できなくなった場合、REDO ログ・ファイルのデータを適用できます。

stop_standby_clause

この句を使用すると、ログ適用サービスを停止できます。この句は、物理スタンバイ・データベースではなく、論理スタンバイ・データベースのみに適用されます。STOP 句を使用すると、適用が正常に停止されます。

convert_database_clause

この句を使用すると、データベースを別の形式に変換できます。

- CONVERT TO PHYSICAL STANDBY を使用すると、プライマリ・データベースまたはスナップショット・スタンバイ・データベースを物理スタンバイ・データベースへ変換できます。
- CONVERT TO SNAPSHOT STANDBY を使用すると、物理スタンバイ・データベースをスナップショット・スタンバイ・データベースへ変換できます。

この句を指定する前に、次の手順を実行します。

- 物理スタンバイ・データベースで、REDO Apply がアクティブな場合は停止します。
- Real Applications Cluster（RAC）データベースで、1 つを除いてすべてのインスタンスを停止します。
- データベースはマウントされているがオープンされていないことを確認します。

データベースは変換後にディスマウントされるため、再起動する必要があります。

注意： スナップショット・スタンバイ・データベースは、物理スタンバイ・データベースに変換される前に読取り / 書込みモードで 1 回以上オープンされている必要があります。

参照： スタンバイ・データベースの詳細は、『Oracle Data Guard 概要および管理』を参照してください。

default_settings_clauses

データベースのデフォルト設定を変更できます。

CHARACTER SET、NATIONAL CHARACTER SET

ALTER DATABASE 文を使用して、データベース・キャラクタ・セットまたは各国語キャラクタ・セットを変更することはできなくなりました。データベース・キャラクタ・セットの移行の詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

SET DEFAULT TABLESPACE 句

この句を使用すると、以降に作成する表領域のデフォルト・タイプを指定または変更できます。表領域が bigfile か smallfile かを指定するには、BIGFILE または SMALLFILE を指定します。

- **bigfile 表領域**には、1 つのデータファイルまたは一時ファイルのみが含まれます。ファイルは、最大で約 40 億（2^{32}）のブロックを格納できます。1 つのデータ・ファイルまたは一時ファイルの最大サイズは、32K ブロックの表領域で 128TB、8K ブロックの表領域で 32TB です。
- **smallfile 表領域**は、Oracle の従来の表領域であり、1022 のデータ・ファイルまたは一時ファイルを含めることができます。それぞれのファイルは、最大で約 400 万（2^{22}）のブロックを格納できます。

  参照：

  - bigfile 表領域の詳細は、『Oracle Database 管理者ガイド』を参照してください。
  - 「表領域のデフォルト・タイプの設定例：」（10-41 ページ）

DEFAULT TABLESPACE 句

この句を使用すると、データベースのデフォルトの永続表領域を構築または変更できます。作成済の表領域を指定する必要があります。この操作の完了後、すべての非 SYSTEM ユーザーに、新しいデフォルトの一時表領域が再度割り当てられます。これらのユーザーがこれ以降作成するすべてのオブジェクトは、デフォルトで、新しいデフォルトの表領域に格納されます。以前に指定したデフォルトの表領域を置き換える場合、以前作成したオブジェクトを古いデフォルトの表領域から新しいデフォルトの表領域に移動した後で、必要に応じて古いデフォルトの表領域を削除することができます。

DEFAULT TEMPORARY TABLESPACE 句

この句を使用すると、データベースのデフォルトの一時表領域を、新しい表領域または表領域グループに変更できます。

- *tablespace* を指定すると、データベースの新しいデフォルトの一時表領域を指定できます。この操作の完了後、以前のデフォルトの一時表領域が割り当てられているすべてのユーザーに、新しいデフォルトの一時表領域が再度割り当てられます。その後、必要に応じて以前のデフォルトの一時表領域を削除することができます。
- *tablespace_group_name* を指定すると、*tablespace_group_name* で指定されている表領域グループのすべての表領域を、データベースのデフォルトの一時表領域として指定できます。この操作の完了後、デフォルトの一時表領域を明示的に割り当てられていないユーザーは、*tablespace_group_name* に含まれる任意の表領域に一時セグメントを作成できます。以前のデフォルトの一時表領域がデフォルトの一時表領域グループに含まれる場合、以前の一時表領域を削除することはできません。

現行のデフォルトの一時表領域またはデフォルトの一時表領域グループの名前を確認するには、ALL_、DBA- または USER_USERS データ・ディクショナリ・ビューの TEMPORARY_TABLESPACE 列を問い合せます。

デフォルトの一時表領域の制限事項： デフォルトの一時表領域には、次の制限事項があります。

- デフォルトの一時表領域として割り当てる表領域、または再度割り当てる表領域は、標準的なブロック・サイズである必要があります。
- SYSTEM 表領域がローカル管理される場合、デフォルトの一時表領域として指定する表領域も、ローカル管理される必要があります。

参照：

- 表領域グループの詳細は、『Oracle Database 管理者ガイド』を参照してください。
- 「デフォルトの一時表領域の変更例：」（10-41 ページ）

instance_clauses

Oracle Real Application Clusters 環境において、ENABLE INSTANCE を指定すると、指定したデータベース・インスタンスにマップされたスレッドを使用可能にできます。使用可能にできるスレッドは、2 つ以上の REDO ログ・ファイル・グループを持つスレッドのみです。また、データベースはオープンされている必要があります。

DISABLE INSTANCE を指定すると、指定したデータベース・インスタンスにマップされたスレッドを使用禁止にできます。インスタンス名は最大 80 文字の文字列です。指定したインスタンスにマップされているスレッドがない場合、エラーが戻されます。データベースは、オープンされている必要がありますが、指定したスレッドを使用しているインスタンスでデータベースをマウント済の場合は、そのスレッドを使用禁止にできません。

参照： インスタンスを使用可能または使用禁止にする場合の詳細は、『Oracle Real Application Clusters 管理およびデプロイメント・ガイド』を参照してください。

RENAME GLOBAL_NAME 句

RENAME GLOBAL_NAME を指定すると、データベースのグローバル名を変更できます。*database* には、データベースの新しい名前を 8 バイト以内の長さで指定します。オプションの *domain* には、ネットワーク階層におけるデータベースの有効な位置を指定します。データベースは、オープンされている必要があります。

注意： データベース名を変更しても、リモート・データベース内の既存のデータベース・リンク、シノニム、ストアド・プロシージャ、ストアド・ファンクションからのユーザーのデータベースに対するグローバル参照は変更されません。これらの参照を変更するのは、リモート・データベース管理者の責任です。

参照： 「グローバル・データベース名の変更例：」（10-42 ページ）

BLOCK CHANGE TRACKING 句

ブロック・チェンジ・トラッキング機能を使用すると、Oracle Database では、プライマリ・データベースおよび任意の物理スタンバイ・データベースの両方で、すべてのデータベース更新の物理的な場所を追跡できます。トラッキングを実行する各データベースで、ブロック・チェンジ・トラッキングを使用可能にする必要があります。トラッキング情報は、ブロック・チェンジ・トラッキング・ファイルという別のファイルに保持されます。Oracle Managed Files を使用している場合、Oracle Database は、DB_CREATE_FILE_DEST で指定した場所にブロック・チェンジ・トラッキング・ファイルを自動的に作成します。Oracle Managed Files を使用していない場合、チェンジ・トラッキング・ファイルの名前を指定する必要があります。Oracle Database は、増分バックアップのパフォーマンス向上などの内部タスクのために、チェンジ・トラッキング・データを使用します。ブロック・チェンジ・トラッキングを使用可能にする場合、データベースは、ARCHIVELOG モードまたは NOARCHIVELOG モードで、オープンされているかマウント済である必要があります。

ENABLE BLOCK CHANGE TRACKING この句を指定すると、ブロック・チェンジ・トラッキングが使用可能になり、Oracle Database によってブロック・チェンジ・トラッキング・ファイルが作成されます。

- USING FILE '*filename*' を指定すると、ブロック・チェンジ・トラッキング・ファイルの名前を（Oracle Database が生成するのではなく）自分で指定できます。Oracle Managed Files を使用していない場合、この句を指定する必要があります。
- REUSE を指定すると、同じ名前の既存のブロック・チェンジ・トラッキング・ファイルを上書きできます。

DISABLE BLOCK CHANGE TRACKING この句を指定すると、Oracle Database によるチェンジ・トラッキングを停止し、既存のブロック・チェンジ・トラッキング・ファイルを削除できます。

> **参照：** ブロック・チェンジ・トラッキングの設定の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』および 10-42 ページの「ブロック・チェンジ・トラッキングを使用可能および使用禁止にする例 :」を参照してください。

flashback_mode_clause

この句を使用すると、データベースを FLASHBACK モードにしたり、そのモードから戻すことができます。この句を指定できるのはデータベースが ARCHIVELOG モードであり、データベースのフラッシュ・リカバリ領域をすでに準備している場合のみです。この句は、データベースがマウントされているがオープンされていない場合に指定できます。REDO Apply がアクティブな場合、物理スタンバイ・データベースでこの句は指定できません。

> **参照：** フラッシュバック操作のためのフラッシュ・リカバリ領域の準備の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

FLASHBACK ON この句を使用すると、データベースは FLASHBACK モードになります。データベースが FLASHBACK モードのとき、Oracle Database は、フラッシュ・リカバリ領域に自動的にフラッシュバック・データベース・ログを作成し、これを管理します。その後、SYSDBA システム権限を持つユーザーは、FLASHBACK DATABASE 文を発行できます。

FLASHBACK OFF この句を使用すると、データベースは FLASHBACK モードから戻されます。Oracle Database は、フラッシュバック・データのロギングを停止し、既存のすべてのフラッシュバック・データベース・ログを削除します。FLASHBACK DATABASE を発行すると、エラーが発生して実行できません。

set_time_zone_clause

この句のセマンティクスは、CREATE DATABASE 文および ALTER DATABASE 文で同じです。この句を ALTER DATABASE と併用すると、データベースのタイムゾーンがリセットされます。データベースのタイムゾーンを確認するには、組込み関数 DBTIMEZONE を問い合せます（5-53 ページの「DBTIMEZONE」を参照）。この句を使用してタイムゾーンを設定または変更した後、新しいタイムゾーンを有効にするためにデータベースを再起動する必要があります。

すべての新しい TIMESTAMP WITH LOCAL TIME ZONE データは、ディスクに格納されるときにデータベースのタイムゾーンに正規化されます。データベースの既存のデータは、自動的に新しいタイムゾーンに更新されません。したがって、データベースに TIMESTAMP WITH LOCAL TIME ZONE データが存在する場合、データベースのタイムゾーンをリセットすることはできません。データベースのタイムゾーンをリセットするには、まず TIMESTAMP WITH LOCAL TIME ZONE データを削除またはエクスポートする必要があります。そのため、データを格納しているデータベースのタイムゾーンを変更しないことをお薦めします。

この句の詳細は、14-26 ページの「CREATE DATABASE」の「set_time_zone_clause」を参照してください。

security_clause

security_clause（GUARD）を使用すると、データベースのデータが変更されないように保護できます。現行のセッションでこの設定を上書きするには、ALTER SESSION DISABLE GUARD 文を使用します。詳細は、11-41 ページの「ALTER SESSION」を参照してください。

ALL ALL を指定すると、SYS 以外のすべてのユーザーが、データベースの内容を変更できなくなります。

STANDBY STANDBY を指定すると、SYS 以外のすべてのユーザーは、論理スタンバイで管理されるデータベース・オブジェクトを変更できなくなります。この設定は、論理スタンバイによってレプリケートされる前に、データをレポート操作によって変更できるようにする場合に便利です。

> **参照：** 論理スタンバイの詳細は、『Oracle Data Guard 概要および管理』を参照してください。

NONE NONE を指定すると、データベースのすべてのデータについて、通常のセキュリティを設定できます。

> **注意：** 論理スタンバイ・データベースには、この設定をできるだけ使用しないでください。

例

READ ONLY | READ WRITE の例： 次の文は、データベースを読取り専用モードでオープンします。

```
ALTER DATABASE OPEN READ ONLY;
```

次の文は、データベースを読み書き両用モードでオープンし、オンライン REDO ログを消去します。

```
ALTER DATABASE OPEN READ WRITE RESETLOGS;
```

パラレル・リカバリ処理の使用例： 次の文は、パラレル・リカバリ処理を使用して表領域のリカバリを行います。

```
ALTER DATABASE
   RECOVER TABLESPACE tbs_03
   PARALLEL;
```

REDO ログ・ファイル・グループの追加例： 次の文は、2 つのメンバーを含む REDO ログ・ファイル・グループを追加し、GROUP パラメータの値に 3 を指定してこのグループを識別します。

```
ALTER DATABASE
  ADD LOGFILE GROUP 3
    ('diska:log3.log' ,
     'diskb:log3.log') SIZE 50K;
```

次の文は、2 つのメンバーを含む REDO ログ・ファイル・グループをスレッド 5（Real Application Clusters 環境内）に追加して、このグループに GROUP パラメータ値 4 を割り当てます。

```
ALTER DATABASE
    ADD LOGFILE THREAD 5 GROUP 4
        ('diska:log4.log',
         'diskb:log4:log');
```

REDO ログ・ファイル・グループ・メンバーの追加例： 次の文は、前述の例で追加した REDO ログ・ファイル・グループに 1 つのメンバーを追加します。

```
ALTER DATABASE
   ADD LOGFILE MEMBER 'diskc:log3.log'
   TO GROUP 3;
```

ログ・ファイル・メンバーの削除例： 次の文は、前述の例で追加した REDO ログ・ファイル・メンバーの 1 つを削除します。

```
ALTER DATABASE
    DROP LOGFILE MEMBER 'diskb:log3.log';
```

次の文は、REDO ログ・ファイル・グループ 3 のすべてのメンバーを削除します。

```
ALTER DATABASE DROP LOGFILE GROUP 3;
```

ログ・ファイル・メンバーの名前の変更例： 次の文は、REDO ログ・ファイル・メンバーの名前を変更します。

```
ALTER DATABASE
    RENAME FILE 'diskc:log3.log' TO 'diskb:log3.log';
```

この例では、REDO ログ・グループ・メンバーのファイルが、別のファイルに変更されただけです。ファイル名が、実際に `diskc:log3.log` から `diskb:log3.log` に変更されたわけではありません。この文を発行する前に、オペレーティング・システムでこのファイル名を変更する必要があります。

表領域のデフォルト・タイプの設定例： 次の文は、これ以降作成される表領域の種類がデフォルトで bigfile 表領域になるように指定します。

```
ALTER DATABASE
    SET DEFAULT BIGFILE TABLESPACE;
```

デフォルトの一時表領域の変更例： 次の文は、`tbs_5`（16-82 ページの「一時表領域の作成例 :」で作成）をデータベースのデフォルトの一時表領域にします。この文は、作成時に何も指定されていない場合にデフォルトの一時表領域を作成するか、既存のデフォルトの一時表領域を `tbs_05` に置き換えます。

```
ALTER DATABASE
   DEFAULT TEMPORARY TABLESPACE tbs_05;
```

または、表領域グループを使用して、表領域グループをデフォルトの一時表領域に定義することもできます。次の文は、表領域グループ `tbs_group_01` の表領域（16-83 ページの「表領域グループへの一時表領域の追加例 :」で作成）をデータベースのデフォルトの一時表領域にします。

```
ALTER DATABASE
   DEFAULT TEMPORARY TABLESPACE tbs_grp_01;
```

新しいデータ・ファイルの作成例： 次の文は、ファイル `tbs_f03.dbf` を基に、新しいデータ・ファイル `tbs_f04.dbf` を作成します。新しいデータ・ファイルを作成する前に、既存のデータ・ファイル（またはこのデータ・ファイルが存在する表領域）をオフラインにする必要があります。

```
ALTER DATABASE
    CREATE DATAFILE 'tbs_f03.dbf'
                 AS 'tbs_f04.dbf';
```

一時ファイルの操作例： 次の例では、12-92 ページの「データ・ファイルおよび一時ファイルの追加例と削除例：」で作成された一時ファイル temp02.dbf をオフラインにして、名前を変更します。

```
ALTER DATABASE TEMPFILE 'temp02.dbf' OFFLINE;

ALTER DATABASE RENAME FILE 'temp02.dbf' TO 'temp03.dbf';
```

一時ファイルの名前を変更する文では、まずファイル temp03.dbf をオペレーティング・システム上に作成する必要があります。

グローバル・データベース名の変更例： 次の文は、データベースのグローバル名を変更し、データベース名とドメインの両方を指定します。

```
ALTER DATABASE
    RENAME GLOBAL_NAME TO demo.world.example.com;
```

ブロック・チェンジ・トラッキングを使用可能および使用禁止にする例： 次の文は、ブロック・チェンジ・トラッキングを使用可能にし、tracking_file というブロック・チェンジ・トラッキング・ファイルを作成（すでに存在する場合は上書き）します。

```
ALTER DATABASE
  ENABLE BLOCK CHANGE TRACKING
    USING FILE 'tracking_file' REUSE;
```

次の文は、ブロック・チェンジ・トラッキングを使用禁止にし、既存のブロック・チェンジ・トラッキング・ファイルを削除します。

```
ALTER DATABASE
  DISABLE BLOCK CHANGE TRACKING;
```

データ・ファイルのサイズの変更例： 次の文は、データ・ファイル diskb:tbs_f5.dat のサイズを変更します。

```
ALTER DATABASE
    DATAFILE 'diskb:tbs_f5.dat' RESIZE 10 M;
```

ログ・ファイルの消去例： 次の文は、ログ・ファイルを消去します。

```
ALTER DATABASE
    CLEAR LOGFILE 'diskc:log3.log';
```

データベースのリカバリ例： 次の文は、データベース全体を完全にリカバリし、必要な新しいアーカイブ REDO ログ・ファイル名を生成します。

```
ALTER DATABASE
  RECOVER AUTOMATIC DATABASE;
```

次の文は、Oracle Database が適用する REDO ログ・ファイル名を明示的に指定します。

```
ALTER DATABASE
    RECOVER LOGFILE 'diskc:log3.log';
```

次の文は、時間ベースのデータベース・リカバリを実行します。

```
ALTER DATABASE
    RECOVER AUTOMATIC UNTIL TIME '2001-10-27:14:00:00';
```

データベースは、2001 年 10 月 27 日の午後 2 時の状態にリカバリされます。

表領域のリカバリ例は、10-40 ページの「パラレル・リカバリ処理の使用例：」を参照してください。

ALTER DIMENSION

用途

ALTER DIMENSION 文を使用すると、ディメンションの階層関係またはディメンション属性を変更できます。

> **参照：** 14-33 ページの「CREATE DIMENSION」および 17-37 ページの「DROP DIMENSION」を参照してください。

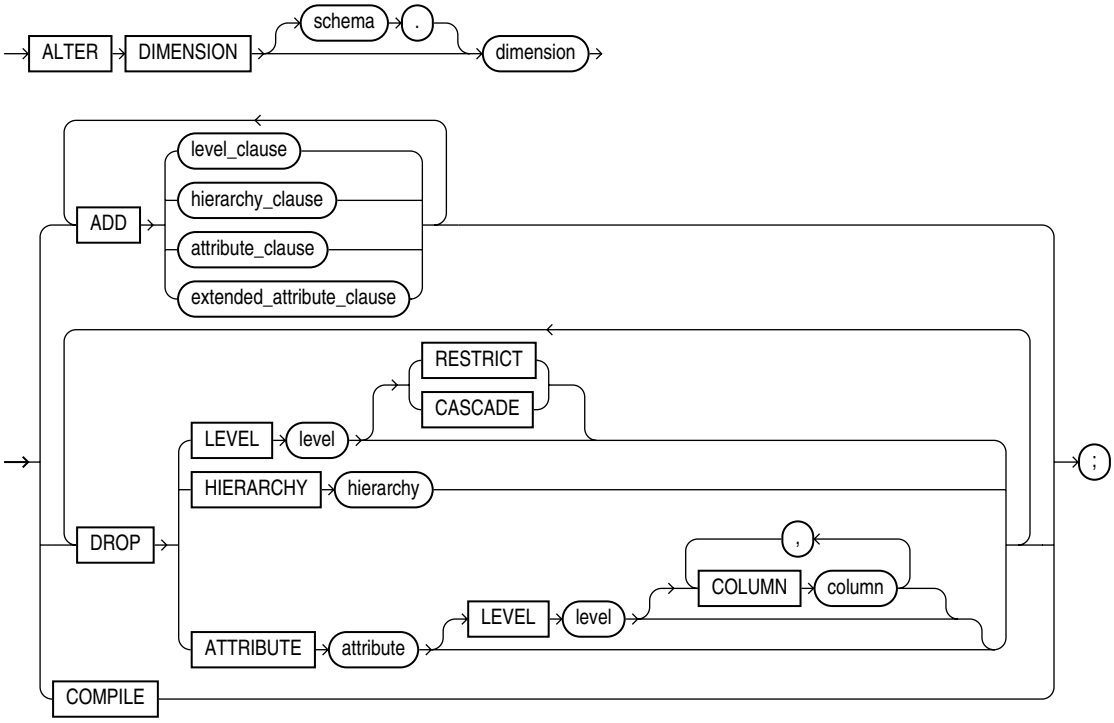
前提条件

ディメンションが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY DIMENSION システム権限が必要です。

所有者の権限があれば、ディメンションはいつでも変更されます。

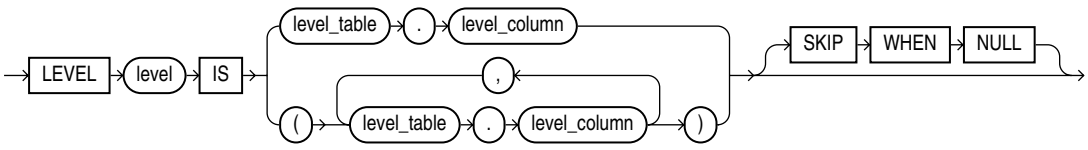
構文

***alter_dimension*::=**

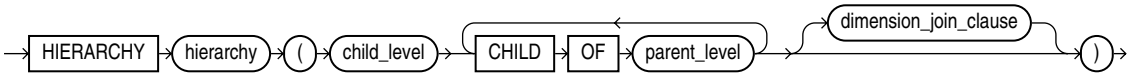


（10-43 ページの level_clause::=、10-44 ページの hierarchy_clause::=、10-44 ページの attribute_clause::=、10-44 ページの extended_attribute_clause::= を参照）

***level_clause*::=**

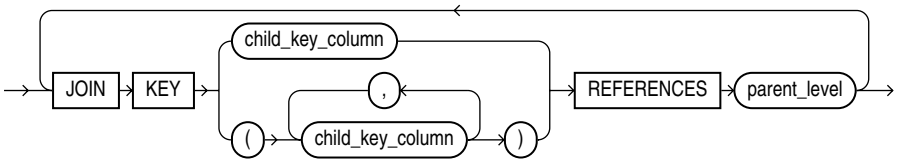


***hierarchy_clause*::=**

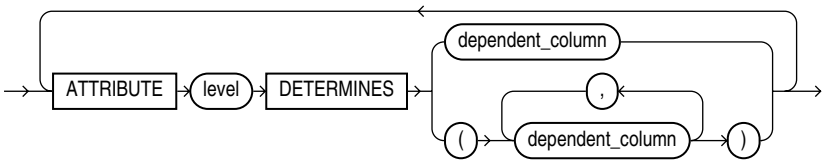


(10-44 ページの dimension_join_clause::= を参照)

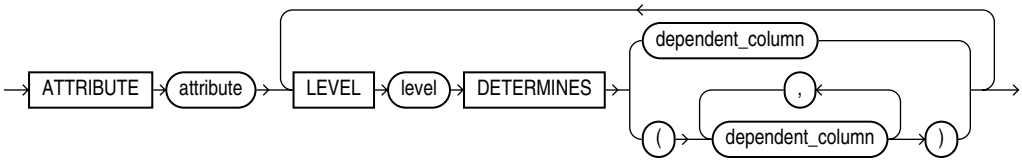
***dimension_join_clause*::=**



***attribute_clause*::=**



***extended_attribute_clause*::=**



セマンティクス

次のキーワード、パラメータおよび句の意味は、ALTER DIMENSION に対してのみ有効です。その他のキーワード、パラメータおよび句には、CREATE DIMENSION 文での機能と同じ機能があります。詳細は、14-33 ページの「CREATE DIMENSION」を参照してください。

schema

変更するディメンションのスキーマを指定します。*schema* を指定しない場合、ディメンションは自分のスキーマ内にあるとみなされます。

dimension

ディメンション名を指定します。ディメンションは、すでに存在している必要があります。

ADD

ADD 句を使用すると、ディメンションにレベル、階層または属性を追加できます。これらの要素の 1 つを追加しても、既存のマテリアライズド・ビューは無効になりません。

ADD LEVEL 句は、他の ADD 句より前に処理されます。

DROP

DROP 句を使用すると、ディメンションからレベル、階層または属性を削除できます。指定するすべてのレベル、階層または属性は、すでに存在している必要があります。

1 つの属性で、1 つのレベルに関連付けられている 1 つ以上のレベルと列の関係を削除できます。

DROP の制限事項 : 属性または階層がレベルを参照する場合は、すべての参照している属性および階層を削除するか、または CASCADE を指定しないかぎり、このレベルを削除することはできません。

CASCADE CASCADE を指定すると、レベルを参照する属性または階層をレベルとともに削除できます。

RESTRICT RESTRICT を指定すると、属性または階層が参照するレベルを削除しないように Oracle Database に指示できます。これはデフォルトです。

COMPILE

COMPILE を指定すると、無効のディメンションを明示的に再コンパイルできます。ADD 句または DROP 句が発行されると、ディメンションは自動的にコンパイルされます。ただし、ディメンションが参照するオブジェクトを変更する（たとえば、ディメンションで参照された表を削除した後再作成する）と、ディメンションが無効になるため、これを明示的に再コンパイルする必要があります。

例

ディメンションの変更例 : 次の例は、サンプル・スキーマ sh の customers_dim ディメンションを変更します。

```
ALTER DIMENSION customers_dim
   DROP ATTRIBUTE country;

ALTER DIMENSION customers_dim
   ADD LEVEL zone IS customers.cust_postal_code
   ADD ATTRIBUTE zone DETERMINES (cust_city);
```

ALTER DISKGROUP

注意： この SQL 文は、自動ストレージ管理を使用しており、自動ストレージ管理インスタンスを起動している場合にのみ有効です。この文の発行は、通常のデータベース・インスタンスからではなく、自動ストレージ管理インスタンスから行う必要があります。自動ストレージ管理インスタンスの起動の詳細は、『Oracle Database ストレージ管理者ガイド』を参照してください。

用途

ALTER DISKGROUP 文を使用すると、ディスク・グループまたはディスク・グループ内のディスクに対して多数の操作を実行できます。

参照：

- ディスク・グループの作成については、14-40 ページの「CREATE DISKGROUP」を参照してください。
- 自動ストレージ管理およびディスク・グループを使用してデータベース管理を簡略化する方法については、『Oracle Database ストレージ管理者ガイド』を参照してください。

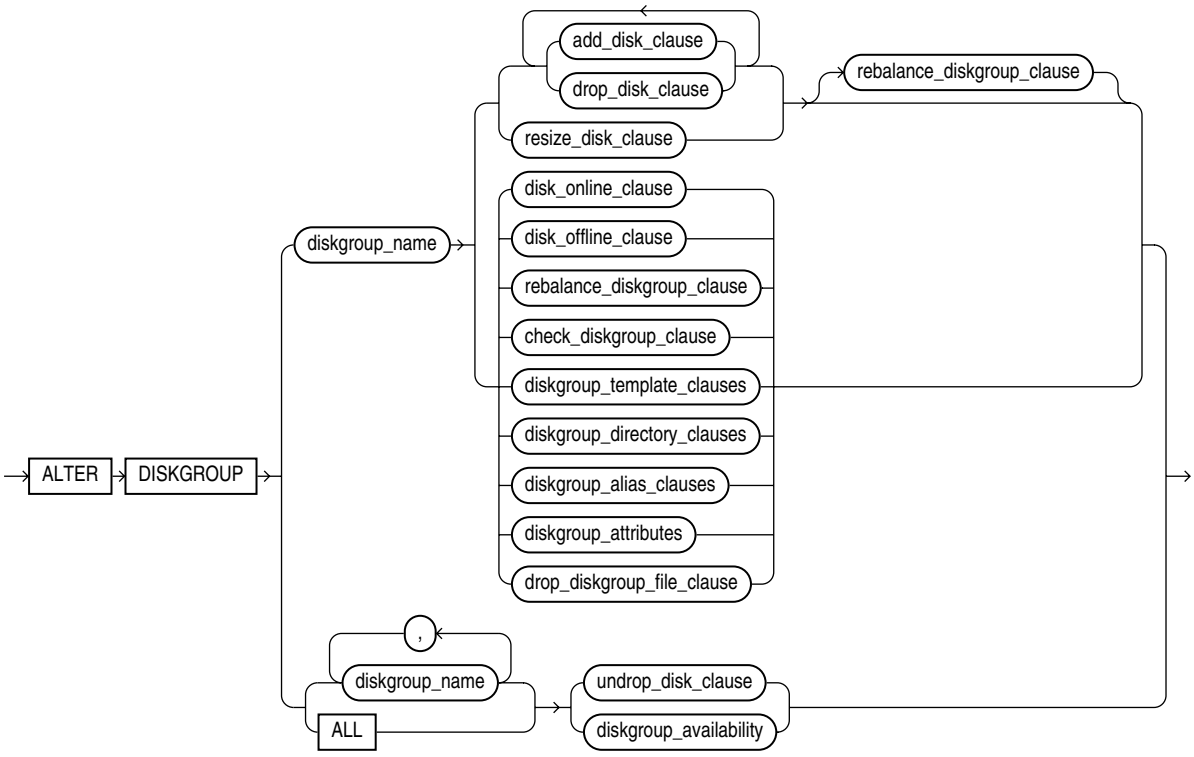
前提条件

この文を発行するには、SYSDBA システム権限が必要です。また、この文を発行する自動ストレージ管理インスタンスが起動されている必要があります。変更するディスク・グループはマウントされている必要があります。

SYSOPER ロールで実行可能な ALTER DISKGROUP 操作は、*diskgroup_availability_clause*、*balance_diskgroup_clause* です。

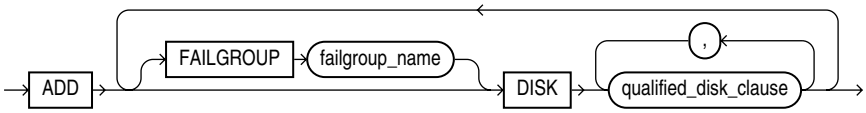
構文

***alter_diskgroup*::=**



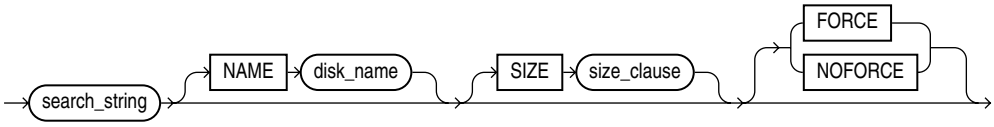
（10-47 ページの add_disk_clause::=、10-48 ページの drop_disk_clauses::=、
10-48 ページの resize_disk_clauses::=、10-48 ページの disk_online_clause::=、
10-48 ページの disk_offline_clause::=、10-49 ページの rebalance_diskgroup_clause::=、
10-49 ページの check_diskgroup_clause::=、10-49 ページの diskgroup_template_clauses::=、
10-49 ページの diskgroup_directory_clauses::=、10-50 ページの diskgroup_alias_clauses::=、
10-50 ページの diskgroup_attributes::=、10-50 ページの drop_diskgroup_file_clause::=、
10-49 ページの undrop_disk_clause::=、10-50 ページの diskgroup_availability::= を参照）

add_disk_clause::=



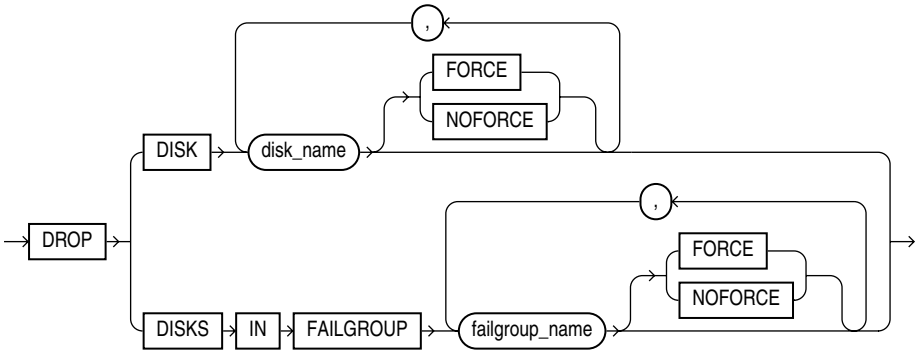
（10-47 ページの qualified_disk_clause::= を参照）

qualified_disk_clause::=

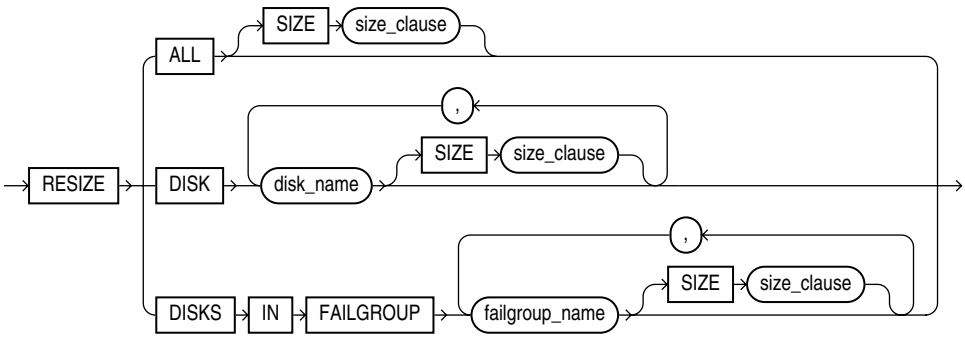


（8-42 ページの size_clause::= を参照）

drop_disk_clauses::=

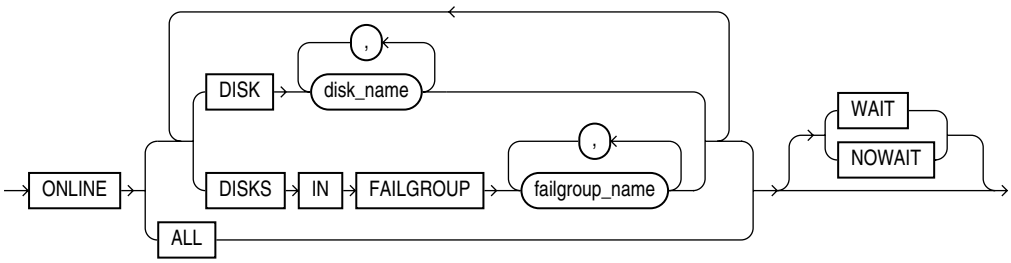


resize_disk_clauses::=

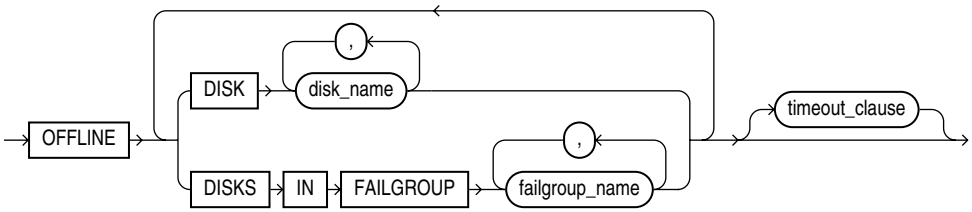


（8-42 ページの size_clause::= を参照）

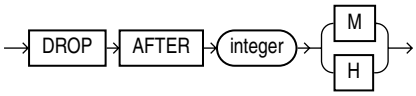
disk_online_clause::=



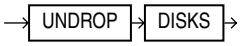
disk_offline_clause::=



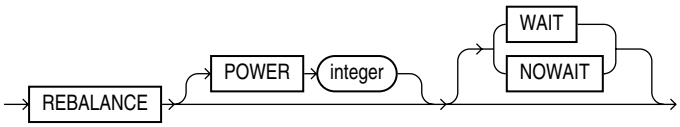
***timeout_clause*::=**



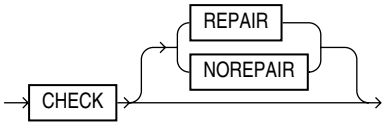
undrop_disk_clause::=



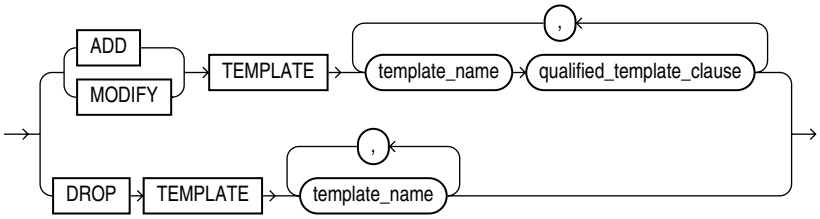
rebalance_diskgroup_clause::=



check_diskgroup_clause::=

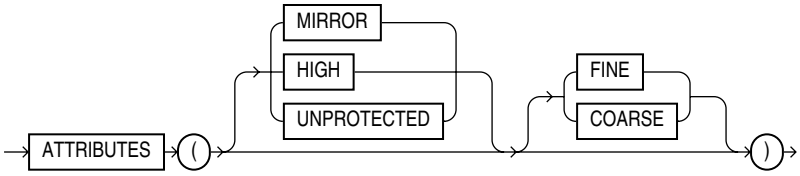


diskgroup_template_clauses::=

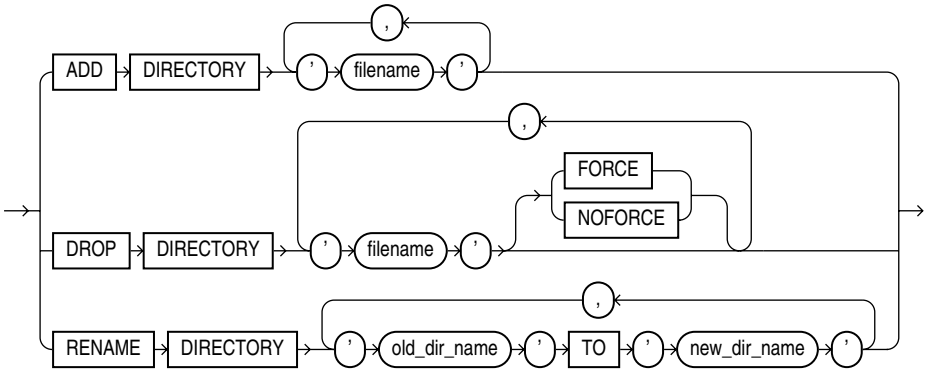


（10-49 ページの qualified_template_clause::= を参照）

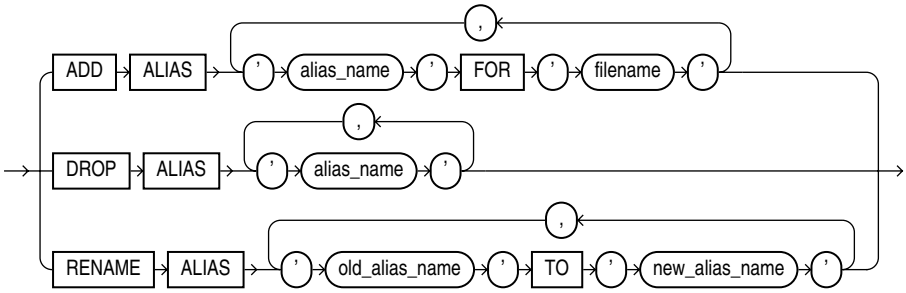
***qualified_template_clause*::=**



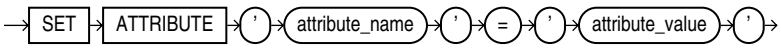
diskgroup_directory_clauses::=



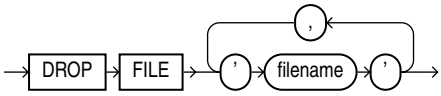
diskgroup_alias_clauses::=



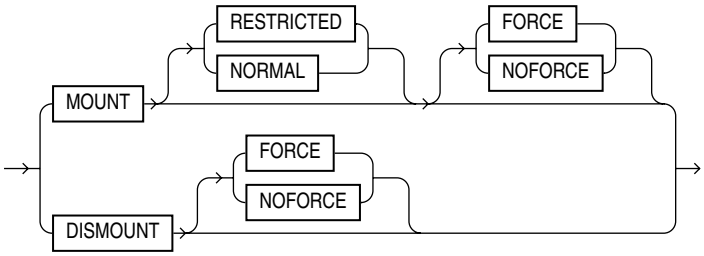
diskgroup_attributes::=



drop_diskgroup_file_clause::=



diskgroup_availability::=



セマンティクス

diskgroup_name

変更するディスク・グループの名前を指定します。既存のディスク・グループの名前を確認するには、V$ASM_DISKGROUP 動的パフォーマンス・ビューを問い合せます。

add_disk_clause

この句を使用すると、ディスク・グループに 1 つ以上のディスクを追加し、新しく追加したディスクの属性を指定できます。この操作を実行すると、自動ストレージ管理によって、自動的にディスク・グループの均衡が再調整されます。

この句を使用して、ディスクの障害グループを変更することはできません。これを行うには、ディスク・グループからディスクを削除した後で、新しい障害グループの一部として、ディスク・グループにディスクを再度追加します。

ディスク・グループの既存のディスク名を確認するには、V$ASM_DISK 動的パフォーマンス・ビューを問い合せます。

FAILGROUP 句 この句を使用すると、新しく追加したディスクを障害グループに割り当てることができます。この句を省略して、標準または高冗長性ディスク・グループにディスクを追加すると、Oracle Database は、新しく追加されたディスクを自動的に障害グループに追加します。障害グループの暗黙的な名前は、オペレーティング・システムに依存しないディスク名と同じです（14-42 ページの「NAME 句」を参照）。

外部冗長性ディスク・グループを作成している場合、この句は指定できません。

qualified_disk_clause

この句のセマンティクスは、CREATE DISKGROUP 文および ALTER DISKGROUP 文で同じです。この句の詳細は、14-42 ページの「CREATE DISKGROUP」の「qualified_disk_clause」を参照してください。

drop_disk_clauses

この句を使用すると、ディスク・グループから 1 つ以上のディスクを削除できます。

DROP DISK DROP DISK 句を使用すると、ディスク・グループから 1 つ以上のディスクを削除し、自動的にディスク・グループの均衡を再調整できます。ディスクを削除すると、自動ストレージ管理によって、ディスクのすべてのデータが再配置され、ディスクをディスク・グループから除外するためにディスク・ヘッダーが消去されます。

DROP DISKS IN FAILGROUP DROP DISKS IN FAILGROUP 句を使用すると、指定した障害グループからすべてのディスクを削除できます。他の動作は、DROP DISK 句と同じです。

FORCE | NOFORCE これらのキーワードを使用すると、ディスクがディスク・グループから除外されたとみなされるタイミングを指定できます。デフォルト設定である NOFORCE を使用することをお薦めします。

- NOFORCE を指定すると、自動ストレージ管理によって、ディスクのすべてのエクステントが他のディスクに再配置された後で、ディスク・グループからそのディスクが除外され、ディスク・グループの均衡が再調整されます。

  注意： DROP DISK ... NOFORCE を指定すると、ディスクが安全に再利用される前またはディスクがシステムから削除される前に、制御がユーザーに戻ります。ディスクの削除操作が完了したことを確認するには、V$ASM_DISK ビューを問い合せて、HEADER_STATUS の値が FORMER であることを確認します。STATE の値が DROPPING の場合は、ディスクの削除や再利用はしないでください。V$ASM_OPERATION ビューを問い合せると、ディスクの削除によって実行される均衡の再調整に必要な時間の概略がわかります。REBALANCE...WAIT（10-53 ページの「rebalance_diskgroup_clause」を参照）も指定すると、均衡の再調整操作が完了しディスクがクリアされるまで、この文は戻りません。ただし、予想外のイベントによって均衡の再調整が失敗することがあるため、V$ASM_DISK の HEADER_STATUS 列が FORMER になっていることを必ず確認してください。

- FORCE を指定すると、ディスクは、ディスク・グループからすぐに除外されます。次に、他のディスク上の冗長コピーからデータが再構築され、そのデータが他のディスクに再配置されて、ディスク・グループの均衡が再調整されます。

  FORCE 句は、削除するディスクを自動ストレージ管理で読み取れなくなった場合などに便利です。ただし、FORCE 句は、NOFORCE による削除より長い時間がかかり、ファイルの一部の保護が低下することがあります。外部冗長性ディスク・グループには FORCE を指定できません。自動ストレージ管理では、ディスク上に冗長データが存在しない場合、ディスクを削除する前にそのディスクからデータを読み取る必要があるためです。

FORCE や NOFORCE を指定したかどうかにかかわらず、ディスクを削除する際の均衡の再調整操作には時間がかかります。進捗を監視するには、V$ASM_OPERATION 動的パフォーマンス・ビューを問い合せます。均衡の再調整操作の詳細は、10-53 ページの「rebalance_diskgroup_clause」を参照してください。

resize_disk_clauses

この句を使用すると、ディスク・グループ内の 1 つ以上のディスクの新しいサイズを指定できます。この句は、オペレーティング・システムによって戻されるサイズや、以前にディスクに指定したサイズを上書きします。

RESIZE ALL この句を指定すると、ディスク・グループ内のすべてのディスクのサイズ変更操作を実行できます。

RESIZE DISK この句を指定すると、指定したディスクのみのサイズを変更できます。

RESIZE DISKS IN FAILGROUP この句を指定すると、指定した障害グループ内のすべてのディスクのサイズを変更できます。

SIZE ディスクのサイズを KB、MB、GB または TB 単位で指定します。ディスク容量を超えるサイズは指定できません。ディスク容量よりも小さいサイズを指定した場合、自動ストレージ管理で使用されるディスク容量が制限されます。この句を省略した場合、自動ストレージ管理ではオペレーティング・システムによって戻されたサイズが使用されます。

disk_offline_clause

disk_offline_clause を使用すると、1 つ以上のディスクをオフラインに切り替えることができます。指定したディスクをオフラインに切り替えることによってディスク・グループの冗長性レベルに違反する場合、この句は失敗します。

デフォルトでは、オフラインに切り替えられるとすぐに、ディスクは自動ストレージ管理によって削除されます。*timeout_clause* を指定してこの操作を遅らせることにより、ディスクを修復してオンラインに戻すことができます。分単位または時間単位でタイムアウト値を指定できます。単位を指定しない場合、デフォルトは時間です。

この句を複数回指定することによって、タイムアウトの期間を変更できます。指定するたびに、自動ストレージ管理によって、ディスク・グループがマウントされている間、直近の *disk_offline_clause* からの時間が測定されます。自動ストレージ管理によってオフライン・ディスクが削除されるまでの残り時間を知るには、V$ASM_DISK の repair_timer 列を問い合せます。

この句によって、disk_repair_time 属性の以前の設定が上書きされます。ディスク・グループの属性の詳細は、表 14-1「ディスク・グループの属性」を参照してください。

disk_online_clause

disk_online_clause を使用すると、1 つ以上のディスクをオンラインにし、ディスク・グループの均衡を再調整できます。WAIT および NOWAIT 句は、ディスク・グループの均衡の手動再調整の場合と同じセマンティクスを持ちます。詳細は、10-53 ページの「POWER」および 10-53 ページの「WAIT | NOWAIT」の説明を参照してください。

> **参照：** ASM ディスクのオンラインおよびオフラインへの切替えの詳細は、『Oracle Database ストレージ管理者ガイド』を参照してください。

undrop_disk_clause

この句を使用すると、ディスク・グループからのディスクの削除を取り消すことができます。1 つ以上のディスク・グループ内のすべてのディスク（*diskgroup_name* を指定）、またはすべてのディスク・グループ内のすべてのディスク（ALL を指定）の保留中の削除を取り消すことができます。

ディスク・グループから完全に削除されたディスクや、完全に削除されたディスク・グループに対しては、この句は無効です。この句を指定すると、実行時間が長い操作が行われます。操作の状態を確認するには、V$ASM_OPERATION 動的パフォーマンス・ビューを問い合せます。

> **参照：** 実行時間が長い自動ストレージ管理操作の詳細は、V$ASM_OPERATION を参照してください。

diskgroup_clauses

この句を使用すると、ディスク・グループ全体に対して操作を実行できます。

rebalance_diskgroup_clause

この句を使用すると、ディスク・グループの均衡を手動で再調整できます。自動ストレージ管理では、すべてのドライブ間で均等にデータ・ファイルが再分散されます。記憶域構成が変化すると、自動ストレージ管理によって、ファイルが均等に配置され、ディスク・グループの均衡が自動的に再調整されるため、この句が必要なことはほとんどありません。ただし、POWER 句を使用して均衡の自動再調整操作の速度を制御する場合、この句は便利です。

POWER POWER 句には、0 ～ 11 の値を指定します。0 を指定すると均衡の再調整操作は停止し、11 を指定すると自動ストレージ管理による均衡の再調整は最高速度で実行されます。POWER 句に指定する値のデフォルトは、ASM_POWER_LIMIT 初期化パラメータの値です。

POWER 句を省略すると、自動ストレージ管理による均衡の自動および指定の再調整操作は、ASM_POWER_LIMIT 初期化パラメータの値に基づいて実行されます。

WAIT | NOWAIT この句を使用すると、制御を、均衡の再調整操作のどの段階でユーザーに戻すか指定できます。

- WAIT を指定すると、ディスクを追加または削除するスクリプトは、ディスク・グループの均衡が再調整されるのを待機してからユーザーに制御を戻します。WAIT モードで実行している再調整操作は、明示的に終了させることができます。ただし、その場合、同じ文で完了しているディスクの追加操作や削除操作が元に戻ることはありません。
- NOWAIT は、文の発行後すぐに制御をユーザーに戻す場合に指定します。これはデフォルトです。

均衡の再調整操作の進捗を監視するには、V$ASM_OPERATION 動的パフォーマンス・ビューを問い合せます。

> **参照：** ディスク・グループの均衡の再調整の詳細は、ASM_POWER_LIMIT および『Oracle Database ストレージ管理者ガイド』を参照してください。また、10-58 ページの「ディスク・グループの均衡の再調整例 :」も参照してください。

check_diskgroup_clause

check_diskgroup_clauses を使用すると、自動ストレージ管理ディスク・グループのメタデータの内部一貫性を検証できます。ディスク・グループは、マウントされている必要があります。自動ストレージ管理によってサマリー・エラーが表示され、検出されたエラーの詳細がアラート・ログに書き込まれます。

CHECK キーワードによって、次の処理が実行されます。

- ディスクの一貫性をチェックします。
- すべてのファイル・エクステント・マップおよび割当て表の一貫性をクロスチェックします。
- 別名メタデータ・ディレクトリおよびファイル・ディレクトリが正しくリンクされていることをチェックします。
- 別名ディレクトリ・ツリーが正しくリンクされていることをチェックします。
- ASM メタデータ・ディレクトリに、到達不可能なブロックが割り当てられていないことをチェックします。

REPAIR | NOREPAIR この句を使用すると、一貫性チェックで検出されたエラーを修正するかどうかを自動ストレージ管理に指示できます。デフォルトは NOREPAIR です。NOREPAIR 設定は、非一貫性が検出された場合には警告を受け取るが、自動ストレージ管理による修正処理を自動的に行わない場合に便利です。

非推奨になる予定の句 以前のリリースでは、ALL、DISK、DISKS IN FAILGROUP または FILE に対して CHECK を指定できました。これらの句は不要になったため、非推奨になる予定です。指定すると、動作は以前のリリースと同じで、アラート・ログにメッセージが追加されます。ただし、これらの句はサポートされなくなる予定であるため、新しいコードには使用しないことをお薦めします。非推奨になる予定の句は、次のとおりです。

- ALL: ディスク・グループ内のすべてのディスクとファイルをチェックします。
- DISK: ディスク・グループ内の 1 つ以上の指定したディスクをチェックします。
- DISKS IN FAILGROUP: 指定した障害グループ内のすべてのディスクをチェックします。
- FILE: ディスク・グループ内の 1 つ以上の指定したファイルをチェックします。ファイル名の参照書式のいずれかを使用する必要があります。自動ストレージ管理のファイル名の参照書式の詳細は、8-28 ページの「ASM_filename」を参照してください。

diskgroup_template_clauses

テンプレートは、属性の名前付きコレクションです。ディスク・グループを作成すると、自動ストレージ管理によって、システムの一連の初期デフォルト・テンプレートがそのディスク・グループに関連付けられます。テンプレートで定義される属性は、そのディスク・グループ内のすべてのファイルに適用されます。次の表に、システムのデフォルト・テンプレートと、様々な種類のファイルに適用される属性を示します。表の後に説明する *diskgroup_template_clauses* を使用すると、テンプレートの属性を変更して、新しいテンプレートを作成できます。

ディスク・グループ・ファイルを作成した後は、この句を使用して属性を変更することはできません。これを行うには、Recovery Manager（RMAN）を使用して、新しい属性を持つ新しいファイルにそのファイルをコピーする必要があります。

表 10-1 自動ストレージ管理システムのファイル・グループのデフォルト・テンプレート

テンプレート名	ファイルの種類	外部冗長性	標準冗長性	高冗長性	ストライプ化
CONTROL	制御ファイル	保護なし	双方向ミラー	3 方向ミラー	密
DATAFILE	データ・ファイルとコピー	保護なし	双方向ミラー	3 方向ミラー	粗
ONLINELOG	オンライン・ログ	保護なし	双方向ミラー	3 方向ミラー	密
ARCHIVELOG	アーカイブ・ログ	保護なし	双方向ミラー	3 方向ミラー	粗
TEMPFILE	一時ファイル	保護なし	双方向ミラー	3 方向ミラー	粗
BACKUPSET	データ・ファイルのバックアップ・ピース、データ・ファイルの増分バックアップ・ピース、およびアーカイブ・ログのバックアップ・ピース	保護なし	双方向ミラー	3 方向ミラー	粗
PARAMETERFILE	SPFILE	保護なし	双方向ミラー	3 方向ミラー	粗
DATAGUARDCONFIG	障害回復構成（スタンバイ・データベースで使用）	保護なし	双方向ミラー	3 方向ミラー	粗
FLASHBACK	フラッシュバック・ログ	保護なし	双方向ミラー	3 方向ミラー	密
CHANGETRACKING	ブロック・チェンジ・トラッキング・データ（増分バックアップで使用）	保護なし	双方向ミラー	3 方向ミラー	粗
DUMPSET	データ・ポンプ・ダンプ・セット	保護なし	双方向ミラー	3 方向ミラー	粗
XTRANSPORT	クロス・プラットフォーム変換データ・ファイル	保護なし	双方向ミラー	3 方向ミラー	粗
AUTOBACKUP	自動バックアップ・ファイル	保護なし	双方向ミラー	3 方向ミラー	粗

ADD TEMPLATE　この句を使用すると、ディスク・グループに 1 つ以上の名前付きテンプレートを追加できます。既存のテンプレート名を確認するには、V$ASM_TEMPLATE 動的パフォーマンス・ビューを問い合せます。

MODIFY TEMPLATE　この句を使用すると、システムのデフォルト・ディスク・グループ・テンプレートまたはユーザー定義のディスク・グループ・テンプレートの属性を変更できます。指定した属性のみが変更されます。指定していないプロパティは、現在の値のまま変更されません。

注意： 以前のリリースでは、キーワード ALTER TEMPLATE が MODIFY TEMPLATE のかわりに使用されました。ALTER キーワードは、下位互換性のためにまだサポートされていますが、他の Oracle SQL との一貫性のために MODIFY に置き換えられました。

---

template_name　追加または変更するテンプレートの名前を指定します。テンプレート名には、データベース・スキーマ・オブジェクトと同じネーミング規則および制限が適用されます。データベース・オブジェクト名の詳細は、2-98 ページの「スキーマ・オブジェクトのネーミング規則」を参照してください。

冗長性レベル　新しく追加または変更するテンプレートの冗長性レベルを指定します。

- MIRROR: このテンプレートが適用されるファイルは、データ・ブロックのミラー化によって保護されます。標準冗長性ディスク・グループでは、プライマリ・エクステントごとに 1 つのミラー・エクステントが存在します（双方向ミラー化）。高冗長性ディスク・グループでは、プライマリ・エクステントごとに 2 つのミラー・エクステントが存在します（3 方向ミラー化）。外部冗長性ディスク・グループのテンプレートには、MIRROR を指定することはできません。
- HIGH: このテンプレートが適用されるファイルは、データ・ブロックのミラー化によって保護されます。標準冗長性および高冗長性ディスク・グループでは、プライマリ・エクステントごとに 2 つのミラー・エクステントが存在します（3 方向ミラー化）。外部冗長性ディスク・グループのテンプレートには、HIGH を指定することはできません。
- UNPROTECTED: このテンプレートが適用されるファイルは、自動記憶域管理によってメディア障害から保護されません。システムのアクションまたはユーザー・コマンドによってディスクがオフラインになると、保護されていないファイルが失われる可能性があります。外部冗長性ディスク・グループには、UNPROTECTED のみが有効な設定です。高冗長性ディスク・グループのテンプレートには、UNPROTECTED を指定することはできません。標準または高冗長性ディスク・グループでは、保護されていないファイルを使用しないことをお薦めします。

この句を指定しない場合、デフォルト値は、標準冗長性ディスク・グループでは MIRROR、高冗長性ディスク・グループでは HIGH、外部冗長性ディスク・グループでは UNPROTECTED になります。

ディスクのストライプ化　このテンプレートが適用されるファイルのストライプ化の方法を指定します。

- FINE: このテンプレートが適用されるファイルは、128KB 単位でストライプ化されます。
- COARSE: このテンプレートが適用されるファイルは、1MB 単位でストライプ化されます。これはデフォルト値です。

DROP TEMPLATE　この句を使用すると、ディスク・グループから 1 つ以上のテンプレートを削除できます。この句で削除できるのはユーザー定義テンプレートのみであり、システム・デフォルト・テンプレートは削除できません。

diskgroup_directory_clauses

自動ストレージ管理ファイル名の別名（10-56 ページの「diskgroup_alias_clauses」を参照）を作成する前に、別名が存在する場所の完全なディレクトリ構造を指定する必要があります。*diskgroup_directory_clauses* を使用すると、そのようなディレクトリ構造を作成および操作できます。

ADD DIRECTORY　この句を使用すると、階層的に名付けられた別名の新しいディレクトリ・パスを作成できます。ディレクトリの各コンポーネントを区切るには、スラッシュ（/）を使用します。各ディレクトリ・コンポーネントの最大長は 48 バイトであり、スラッシュ文字を含むことはできません。コンポーネントの最初または最後の文字に空白を使用することはできません。ディレクトリ・パス全体の長さは、256 バイトから、このディレクトリに作成する別名（10-56 ページの「diskgroup_alias_clauses」を参照）の長さを引いた長さを超えることはできません。

DROP DIRECTORY　この句を使用すると、階層的に名付けられた別名のディレクトリを削除できます。FORCE が指定されていないかぎり、自動ストレージ管理では、別名の定義を含むディレクトリは削除されません。この句は、システム別名の一部として作成されたディレクトリの削除には無効です。そのようなディレクトリは、V$ASM_ALIAS 動的パフォーマンス・ビューの SYSTEM_CREATED 列に Y という値が示されています。

RENAME DIRECTORY　この句を使用すると、階層的に名付けられた別名のディレクトリ名を変更できます。この句は、システム別名の一部として作成されたディレクトリ名の変更には無効です。そのようなディレクトリは、V$ASM_ALIAS 動的パフォーマンス・ビューの SYSTEM_CREATED 列に Y という値が示されています。

diskgroup_alias_clauses

自動ストレージ管理ファイルが暗黙的またはユーザー指定によって作成されると、そのファイルには、ドット付きの数値の組で終わる完全修飾名が割り当てられます（8-26 ページの「file_specification」を参照）。*diskgroup_alias_clauses* を使用すると、自動ストレージ管理ファイル名に、よりわかりやすい別名を作成できます。ドット付きの数値の組で終わる別名は指定できません。このような書式を使用すると、自動ストレージ管理ファイル名と区別できないためです。

この句を指定する前に、まずネーミング規則に従ってディレクトリ構造を作成する必要があります（10-56 ページの「diskgroup_directory_clauses」を参照）。別名全体の長さは最大 256 バイトです（ディレクトリの接頭辞を含む）。別名では大 / 小文字は区別されませんが、大 / 小文字の区別は保持されます。

ADD ALIAS　この句を使用すると、自動ストレージ管理ファイル名の別名を作成できます。*alias_name* には、ディレクトリのフルパスと別名を指定します。自動ストレージ管理の既存の別名を確認するには、V$ASM_ALIAS 動的パフォーマンス・ビューを問い合せます。自動ストレージ管理ファイル名の詳細は、8-28 ページの「ASM_filename」を参照してください。

DROP ALIAS　この句を使用すると、ディスク・グループ・ディレクトリから別名を削除できます。各別名には、ディレクトリのフルパスと別名を指定します。別名が参照する元のファイルは変更されません。

RENAME ALIAS　この句を使用すると、既存の別名を変更できます。*alias_name* には、ディレクトリのフルパスと別名を指定します。

別名の削除および名前の変更の制限事項：　システム生成された別名を削除したり、名前を変更することはできません。別名がシステム生成されたものかどうかを確認するには、V$ASM_ALIAS 動的パフォーマンス・ビューの SYSTEM_CREATED 列を問い合せます。

diskgroup_attributes

この句を使用すると、ディスク・グループの属性を指定できます。14-43 ページの表 14-1「ディスク・グループの属性」に、この句で設定できる属性を示します。この句の動作の詳細は、14-43 ページの「CREATE DISKGROUP」の「ATTRIBUTE 句」を参照してください。

drop_diskgroup_file_clause

この句を使用すると、ディスク・グループからファイルを削除できます。自動ストレージ管理によって、削除するファイルに関連付けられたすべての別名も削除されます。ファイル名の参照書式のいずれかを使用する必要があります。ほとんどの自動ストレージ管理ファイルは Oracle Managed Files であり、不要になると自動的に削除されるため、手動で削除する必要はありません。自動ストレージ管理のファイル名の参照書式の詳細は、8-28 ページの「ASM_filename」を参照してください。

diskgroup_availability

この句を使用すると、自動ストレージ管理インスタンスと同じノードで実行されているデータベース・インスタンスに対して、1 つ以上のディスク・グループを使用可能または使用禁止にできます。この句は、クラスタ内の他のノードのディスク・グループの状態には影響しません。

MOUNT　MOUNT を指定すると、ローカル自動ストレージ管理インスタンスのディスク・グループをマウントできます。ALL MOUNT を指定すると、ASM_DISKGROUPS 初期化パラメータで指定されたすべてのディスク・グループがマウントされます。ファイル操作は、ディスク・グループがマウントされている場合のみ可能です。

RESTRICTED | NORMAL　これらの句を使用すると、ディスク・グループがマウントされる方式を決定できます。

- RESTRICTED モードでは、ディスク・グループは単一インスタンス排他モードでマウントされます。同じクラスタ内の他の ASM インスタンスは、そのディスク・グループをマウントできません。このモードでは、ASM クライアントはディスク・グループを使用できません。
- NORMAL モードでは、ディスク・グループは共有モードでマウントされます。そのため、他の ASM インスタンスおよびクライアントがディスク・グループにアクセスできます。これはデフォルトです。

FORCE | NOFORCE　これらの句を使用すると、ディスク・グループがマウントされる環境を決定できます。

- FORCE モードでは、ASM は、ディスク・グループに属するすべてのデバイスを検出できない場合でも、そのディスク・グループをマウントしようとします。この設定は、標準冗長性または高冗長性ディスク・グループのディスマウント中に、そのディスクの一部が使用不可になった場合に役立ちます。MOUNT FORCE が成功した場合、ASM は欠落しているディスクをオフラインにします。

  ASM がディスク・グループ内のすべてのディスクを検出した場合、MOUNT FORCE は失敗します。そのため、MOUNT FORCE 設定は、一部のディスクが使用不可の場合にのみ使用します。それ以外の場合は、NOFORCE を使用します。

  標準冗長性および高冗長性ディスク・グループでは、1 つの障害グループのディスクが使用不可になる場合があり、MOUNT FORCE は正常に実行されます。また、高冗長性ディスク・グループでは、2 つの異なる障害グループの 2 つのディスクが使用不可になる場合があり、MOUNT FORCE は正常に実行されます。使用不可のディスクのその他の組合せでは、ASM はすべてのユーザー・データまたはメタデータの有効なコピーが使用可能なディスク上にあることを保証できないため、操作は失敗します。
- NOFORCE モードでは、ASM は、すべてのメンバー・ディスクが検出されないかぎり、ディスク・グループをマウントしようとしません。これはデフォルトです。

> **参照：**　初期化パラメータ・ファイルへのディスク・グループ名の追加の詳細は、ASM_DISKGROUPS を参照してください。

DISMOUNT DISMOUNT を指定すると、指定したディスク・グループをディスマウントできます。FORCE が指定されていないかぎり、ディスク・グループのいずれかのファイルがオープンされていると、エラーが戻されます。ALL DISMOUNT を指定すると、現在マウントされているすべてのディスク・グループがディスマウントされます。ファイル操作は、ディスク・グループがマウントされている場合のみ可能です。

FORCE FORCE を指定すると、ディスク・グループのいずれかのファイルがオープンされていてもディスク・グループをディスマウントするように、自動ストレージ管理に指示できます。

例

次の例では、dgroup_01 というディスク・グループが必要です。ここでは、ASM_DISKSTRING が $ORACLE_HOME/disks/* に設定されていることを想定しています。また、Oracle ユーザーが $ORACLE_HOME/disks/d100 への読取り / 書込み権限を持っていると想定しています。dgroup_01 の作成方法については、14-44 ページの「ディスク・グループの作成例 :」を参照してください。

ディスク・グループへのディスクの追加例 : 次の文は、ディスク d100 をディスク・グループ dgroup_01 に追加します。

```
ALTER DISKGROUP dgroup_01
  ADD DISK '$ORACLE_HOME/disks/d100';
```

ディスク・グループからのディスクの削除例 : 次の文は、ディスク dgroup_01_0000 をディスク・グループ dgroup_01 から削除します。

```
ALTER DISKGROUP dgroup_01
  DROP DISK dgroup_01_0000;
```

ディスク・グループからのディスクの削除の取消し例 : 次の文は、ディスク・グループ dgroup_01 からのディスクの削除を取り消します。

```
ALTER DISKGROUP dgroup_01
  UNDROP DISKS;
```

ディスク・グループのサイズの変更例 : 次の文は、ディスク・グループ dgroup_01 のすべてのディスクのサイズを変更します。

```
ALTER DISKGROUP dgroup_01
  RESIZE ALL
  SIZE 36G;
```

ディスク・グループの均衡の再調整例 : 次の文は、ディスク・グループ dgroup_01 の均衡を手動で再調整し、自動ストレージ管理による均衡の再調整を最高速度で実行します。

```
ALTER DISKGROUP dgroup_01
  REBALANCE POWER 11 WAIT;
```

WAIT キーワードを指定すると、データベースでは、ディスク・グループの均衡が再調整されるのを待機してからユーザーに制御を戻します。

ディスク・グループのメタデータの内部一貫性の検証例 : 次の文は、自動ストレージ管理ディスク・グループのメタデータの内部一貫性を検証し、検出されたエラーの修正を自動ストレージ管理に指示します。

```
ALTER DISKGROUP dgroup_01
  CHECK ALL
  REPAIR;
```

ディスク・グループへの名前付きテンプレートの追加例： 次の文は、名前付きテンプレート template_01 をディスク・グループ dgroup_01 に追加します。

```
ALTER DISKGROUP dgroup_01
  ADD TEMPLATE template_01
    ATTRIBUTES (UNPROTECTED COARSE);
```

ディスク・グループ・テンプレートの属性の変更例： 次の文は、システムのデフォルト・ディスク・グループ・テンプレートまたはユーザー定義のディスク・グループ・テンプレート template_01 の属性を変更します。

```
ALTER DISKGROUP dgroup_01
  MODIFY TEMPLATE template_01
    ATTRIBUTES (FINE);
```

ディスク・グループからのユーザー定義テンプレートの削除例： 次の文は、ユーザー定義テンプレート template_01 をディスク・グループ dgroup_01 から削除します。

```
ALTER DISKGROUP dgroup_01
  DROP TEMPLATE template_01;
```

階層的に名付けられた別名のディレクトリ・パスの作成例： 次の文は、別名が存在する場所のディレクトリ構造を指定します。

```
ALTER DISKGROUP dgroup_01
  ADD DIRECTORY '+dgroup_01/alias_dir';
```

自動ストレージ管理ファイル名の別名の作成例： 次の文は、自動ストレージ管理の数値のファイル名を指定してユーザー別名を作成します。

```
ALTER DISKGROUP dgroup_01
  ADD ALIAS '+dgroup_01/alias_dir/datafile.dbf'
    FOR '+dgroup_01.261.1';
```

ディスク・グループのディスマウント例： 次の文は、ディスク・グループ dgroup_01 をディスマウントします。この文は、アクティブなファイルが 1 つ以上ある場合でも、ディスク・グループをディスマウントします。

```
ALTER DISKGROUP dgroup_01
  DISMOUNT FORCE;
```

ディスク・グループのマウント例： 次の文は、ディスク・グループ dgroup_01 をマウントします。

```
ALTER DISKGROUP dgroup_01
  MOUNT;
```

ALTER FLASHBACK ARCHIVE

用途

ALTER FLASHBACK ARCHIVE 文を使用すると、次の操作を実行できます。

- システムのデフォルトのフラッシュバック・データ・アーカイブとしてのフラッシュバック・データ・アーカイブの指定
- フラッシュバック・データ・アーカイブによって使用される表領域の追加
- フラッシュバック・データ・アーカイブによって使用される表領域の割当て制限の変更
- フラッシュバック・データ・アーカイブによる使用からの表領域の削除
- フラッシュバック・データ・アーカイブの保存期間の変更
- 不要な古いデータのフラッシュバック・データ・アーカイブの消去

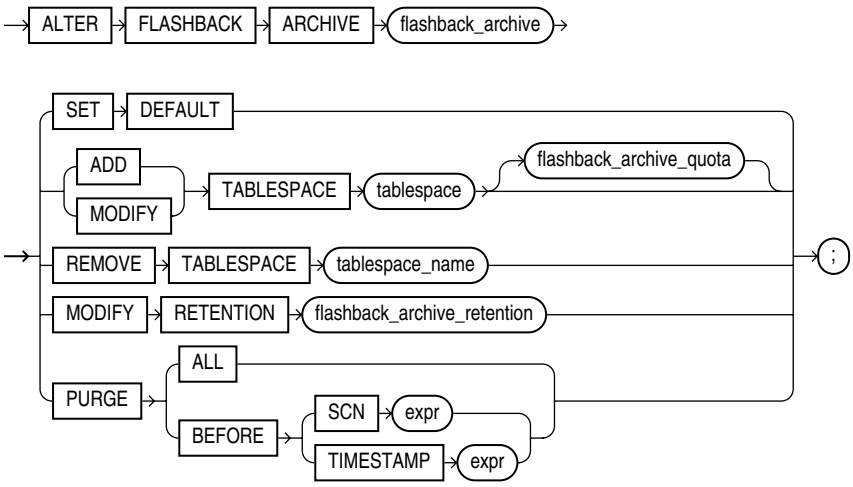
> **参照：** フラッシュバック・データ・アーカイブの使用方法の詳細は、『Oracle Database アドバンスト・アプリケーション開発者ガイド』および 14-45 ページの「CREATE FLASHBACK ARCHIVE」を参照してください。

前提条件

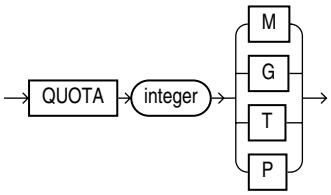
フラッシュバック・データ・アーカイブを変更するには、FLASHBACK ARCHIVE ADMINISTER システム権限が必要です。フラッシュバック・データ・アーカイブ表領域を追加、変更または削除するには、影響を受ける表領域に対する適切な権限も必要です。

構文

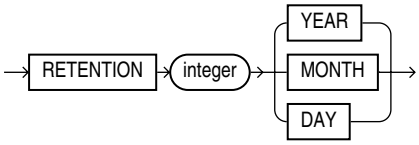
***alter_flashback_archive*::=**



***flashback_archive_quota*::=**



***flashback_archive_retention*::=**



セマンティクス

flashback_archive

既存のフラッシュバック・データ・アーカイブの名前を指定します。

SET DEFAULT

この句を指定するには、SYSDBA としてログインしている必要があります。この句を使用すると、このフラッシュバック・データ・アーカイブをシステムのデフォルトのフラッシュバック・データ・アーカイブとして指定できます。CREATE TABLE 文または ALTER TABLE 文に、フラッシュバック・データ・アーカイブ名を指定しないで *flashback_archive_clause* が指定されている場合は、データベースはデフォルトのフラッシュバック・データ・アーカイブを使用して、該当する表のデータを格納します。

この文によって、異なるフラッシュバック・データ・アーカイブのデフォルトとしての以前の指定は上書きされます。

> **参照：** 詳細は、16-56 ページの「CREATE TABLE」の「flashback_archive_clause」を参照してください。

ADD TABLESPACE

この句を使用すると、フラッシュバック・データ・アーカイブに表領域を追加できます。*flashback_archive_quota* 句を使用して、新規表領域内のフラッシュバック・データ・アーカイブによって使用できる領域の量を指定できます。この句を省略すると、新しく追加された表領域内でフラッシュバック・データ・アーカイブの領域の制限はありません。

MODIFY TABLESPACE

この句を使用すると、フラッシュバック・データ・アーカイブによってすでに使用されている表領域の割当て制限を変更できます。

REMOVE TABLESPACE

この句を使用すると、フラッシュバック・データ・アーカイブによる使用から表領域を削除できます。フラッシュバック・データ・アーカイブによって使用される最後に残った表領域は削除できません。

削除する表領域にフラッシュバック・アーカイブの保存期間内のデータが含まれている場合は、そのデータも削除されます。そのため、この句を使用して表領域を削除する前に、データを別の表領域に移動する必要があります。

MODIFY RETENTION

この句を使用すると、フラッシュバック・データ・アーカイブの保存期間を変更できます。

PURGE

この句を使用すると、フラッシュバック・データ・アーカイブからデータを消去できます。

- PURGE ALL を指定すると、フラッシュバック・データ・アーカイブからすべてのデータが削除されます。この履歴情報は、フラッシュバック問合せで指定された SCN またはタイムスタンプが UNDO 保存期間内である場合にのみ、フラッシュバック問合せを使用して取得できます。

- PURGE BEFORE SCN を指定すると、指定したシステム変更番号より前のフラッシュバック・データ・アーカイブからすべてのデータが削除されます。
- PURGE BEFORE TIMESTAMP を指定すると、指定したタイムスタンプより前のフラッシュバック・データ・アーカイブからすべてのデータが削除されます。

参照： フラッシュバック・データ・アーカイブの作成の詳細およびフラッシュバック・データ・アーカイブの簡単な使用例は、14-45 ページの「CREATE FLASHBACK ARCHIVE」を参照してください。

ALTER FUNCTION

用途

ファンクションは PL/SQL を使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

ALTER FUNCTION 文を使用すると、無効なスタンドアロンのストアド・ファンクションを再コンパイルできます。明示的に再コンパイルすることによって、実行時に暗黙的に再コンパイルする必要がなくなり、また、実行時のコンパイル・エラーとパフォーマンス上のオーバーヘッドもなくなります。

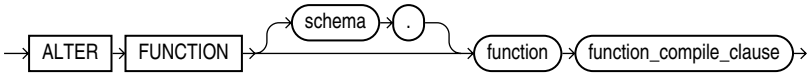
この文を使用して既存のファンクションの宣言や定義を変更することはできません。ファンクションを再宣言または再定義する場合は、OR REPLACE 句を指定して CREATE FUNCTION 文を使用します。14-48 ページの「CREATE FUNCTION」を参照してください。

前提条件

ファンクションが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY PROCEDURE システム権限が必要です。

構文

***alter_function*::=**



（*function_compile_clause*: この句の構文の詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。）

セマンティクス

schema

ファンクションが含まれているスキーマを指定します。*schema* を指定しない場合、ファンクションは自分のスキーマ内にあるとみなされます。

function

再コンパイルするファンクション名を指定します。

function_compile_clause

この句の構文とセマンティクスの詳細およびファンクションの作成とコンパイルの詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

ALTER INDEX

用途

ALTER INDEX 文を使用すると、既存の索引を変更または再作成できます。

> **参照：** 索引の作成については、14-50 ページの「CREATE INDEX」を参照してください。

前提条件

索引が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY INDEX システム権限が必要です。

MONITORING USAGE 句を実行する場合は、索引は自分のスキーマ内に存在する必要があります。

ドメイン索引を変更する場合は、索引の索引タイプに対して EXECUTE オブジェクト権限が必要です。

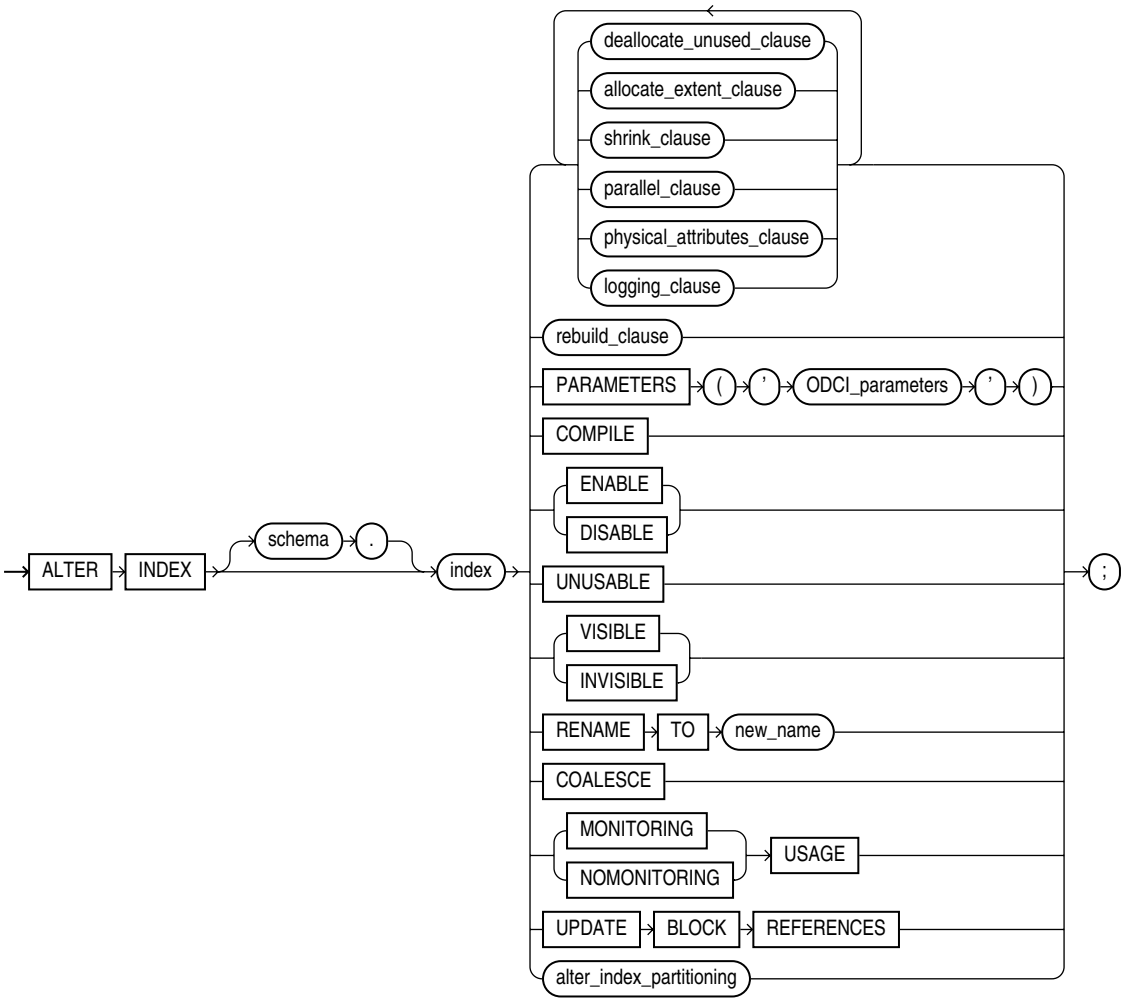
スキーマ・オブジェクト権限は、個々の索引パーティションまたはサブパーティションではなく、親索引に付与されている必要があります。

索引パーティションの変更、再作成または分割、索引サブパーティションの変更または再作成を行う場合は、表領域割当て制限が必要です。

> **参照：** ドメイン索引については、14-50 ページの「CREATE INDEX」および『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。

構文

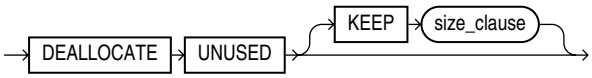
***alter_index*::=**



（10-65 ページの deallocate_unused_clause::=、10-66 ページの allocate_extent_clause::=、
10-66 ページの shrink_clause::=、10-66 ページの parallel_clause::=、
10-66 ページの physical_attributes_clause::=、8-34 ページの logging_clause::=、
10-67 ページの rebuild_clause::=、10-67 ページの alter_index_partitioning::= を参照）

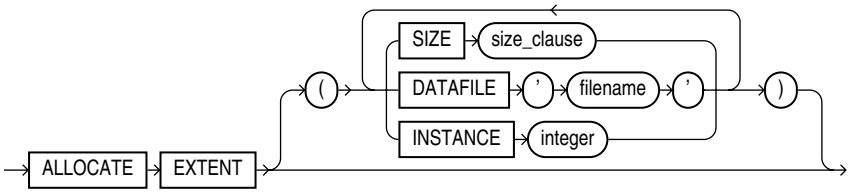
（*XMLIndex_parameters* については、『Oracle XML DB 開発者ガイド』を参照してください。）

deallocate_unused_clause::=



（8-42 ページの size_clause::= を参照）

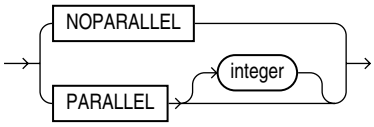
allocate_extent_clause::=



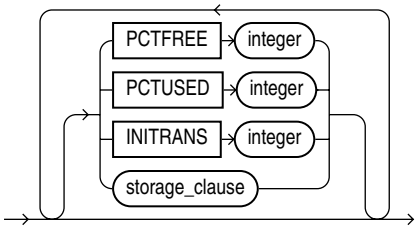
(8-42 ページの size_clause::= を参照)

shrink_clause::=

parallel_clause::=

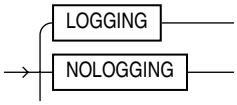


physical_attributes_clause::=

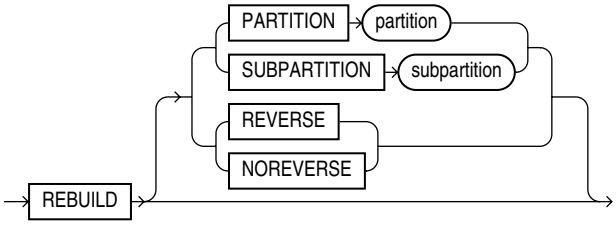


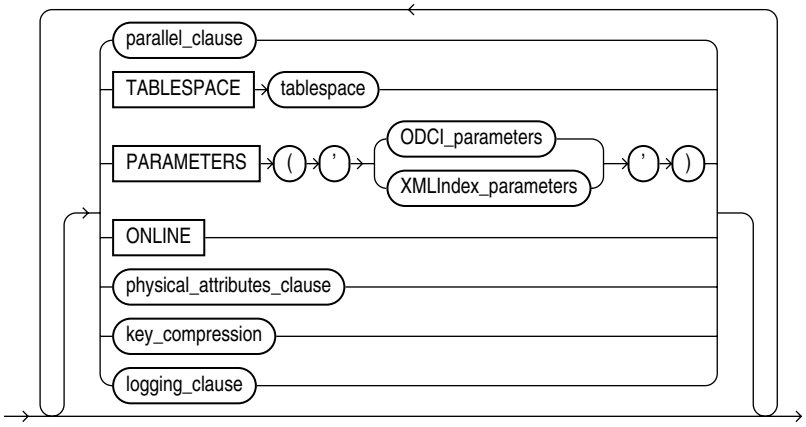
(8-44 ページの storage_clause::= を参照)

logging_clause::=



rebuild_clause::=

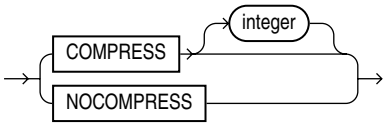




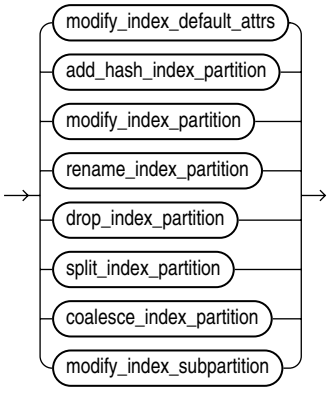
（10-66 ページの parallel_clause::=、10-66 ページの physical_attributes_clause::=、10-67 ページの key_compression::=、8-34 ページの logging_clause::= を参照）

（*XMLIndex_parameters* については、『Oracle XML DB 開発者ガイド』を参照してください。）

key_compression::=

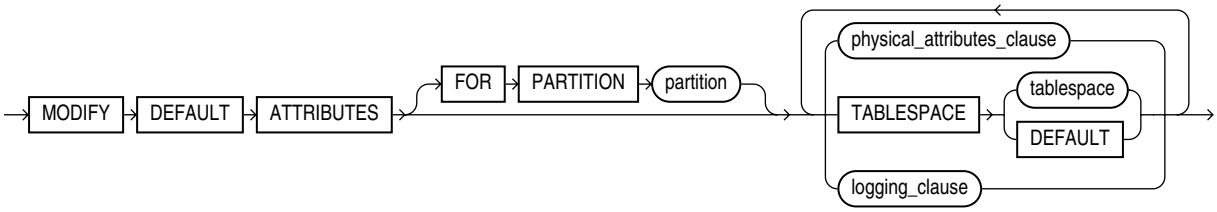


alter_index_partitioning::=



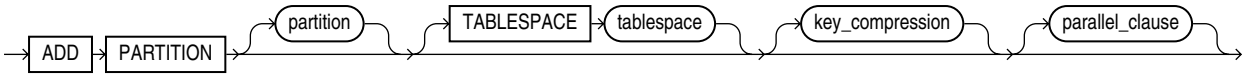
（10-68 ページの modify_index_default_attrs::=、10-68 ページの add_hash_index_partition::=、10-68 ページの modify_index_partition::=、10-68 ページの rename_index_partition::=、10-69 ページの drop_index_partition::=、10-69 ページの split_index_partition::=、10-68 ページの coalesce_index_partition::=、10-69 ページの modify_index_subpartition::= を参照）

modify_index_default_attrs::=



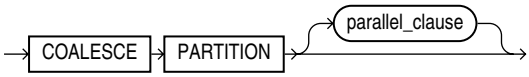
(10-66 ページの physical_attributes_clause::=、8-34 ページの logging_clause::= を参照)

add_hash_index_partition::=



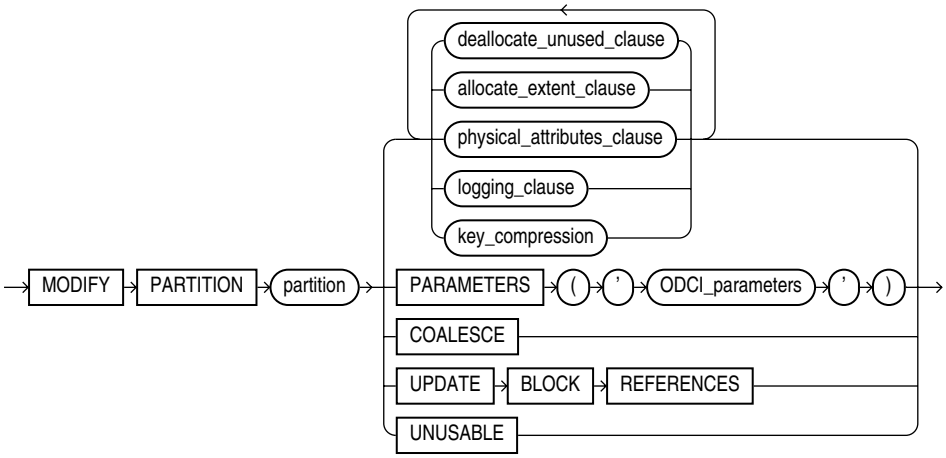
(10-66 ページの parallel_clause::= を参照)

coalesce_index_partition::=



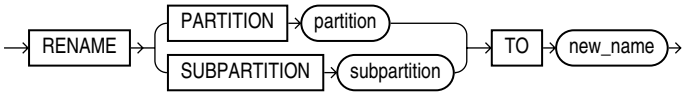
(10-66 ページの parallel_clause::= を参照)

modify_index_partition::=



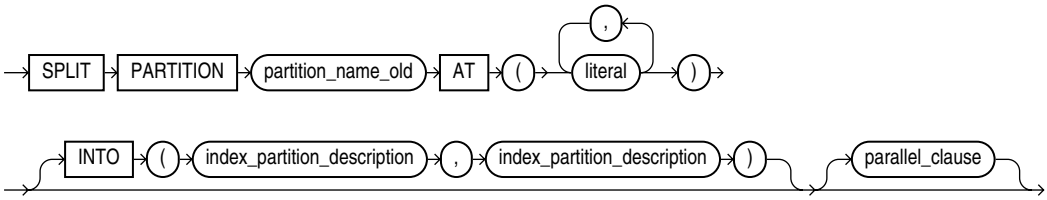
(10-65 ページの deallocate_unused_clause::=、10-66 ページの allocate_extent_clause::=、
10-66 ページの physical_attributes_clause::=、8-34 ページの logging_clause::=、
10-67 ページの key_compression::= を参照)

rename_index_partition::=



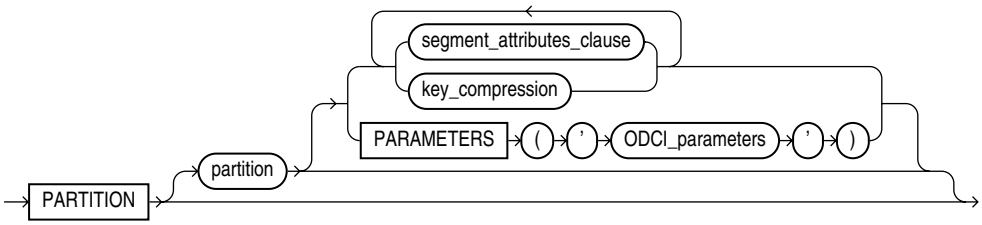
drop_index_partition::=

split_index_partition::=



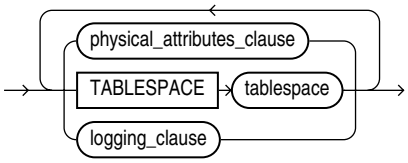
（10-66 ページの parallel_clause::= を参照）

***index_partition_description*::=**



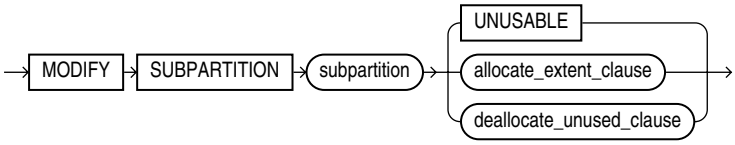
（10-69 ページの segment_attributes_clause::=、10-67 ページの key_compression::= を参照）

***segment_attributes_clause*::=**



（10-66 ページの physical_attributes_clause::=、8-34 ページの logging_clause::= を参照）

modify_index_subpartition::=



（10-66 ページの allocate_extent_clause::=、10-65 ページの deallocate_unused_clause::= を参照）

セマンティクス

schema

索引が含まれているスキーマを指定します。*schema* を指定しない場合、索引は自分のスキーマ内にあるとみなされます。

index

変更する索引の名前を指定します。

索引変更の制限事項： 索引の変更には、次の制限事項があります。

- *index* がドメイン索引である場合は、PARAMETERS 句、RENAME 句、*rebuild_clause*（PARAMETERS 句の有無に関係なく）、*parallel_clause* または UNUSABLE 句のみ指定できます。その他のすべての句は無効です。
- LOADING または FAILED のマークが付いているドメイン索引は、変更または名前の変更ができません。索引に FAILED のマークが付いている場合、REBUILD 句のみ指定できます。

> **参照：** ドメイン索引の LOADING および FAILED 状態については、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。

deallocate_unused_clause

deallocate_unused_clause 句を使用すると、索引の終わりの未使用領域の割当てを明示的に解除し、解放された領域が表領域内の他のセグメントで使用可能になります。

index がレンジ・パーティションまたはハッシュ・パーティションである場合、各索引パーティションの未使用領域の割当てが解除されます。*index* がコンポジット・パーティション表のローカル索引である場合、各索引サブパーティションの未使用領域の割当てが解除されます。

領域の割当て解除の制限事項： 領域の割当て解除には、次の制限事項があります。

- この句は、一時表の索引に対して指定できません。
- この句および *rebuild_clause* は、指定できません。

この句の詳細は、8-24 ページの「deallocate_unused_clause」を参照してください。

KEEP *integer* KEEP 句を使用すると、割当てを解除した後に索引に残す、最高水位標を超えるバイト数を指定できます。残りのエクステント数が MINEXTENTS より少ない場合、MINEXTENTS は現行のエクステント数に設定されます。初期エクステントが INITIAL より小さくなると、INITIAL は初期エクステントの現行の値に設定されます。KEEP を指定しないと、すべての未使用領域が解放されます。

この句の詳細は、12-2 ページの「ALTER TABLE」を参照してください。

allocate_extent_clause

allocate_extent_clause を使用すると、索引の新しいエクステントを明示的に割り当てることができます。ハッシュ・パーティション表のローカル索引に対して、新規エクステントが索引の各パーティションに割り当てられます。

エクステントの割当ての制限事項： この句は、一時表の索引、レンジ・パーティションまたはコンポジット・パーティション索引に対して指定できません。

この句の詳細は、8-2 ページの「allocate_extent_clause」を参照してください。

shrink_clause

この句を使用すると、索引セグメントを縮小化できます。ALTER INDEX ... SHRINK SPACE COMPACT を指定することは、ALTER INDEX ... COALESCE を指定することと同じです。

この句の詳細は、12-35 ページの「CREATE TABLE」の「shrink_clause」を参照してください。

索引セグメントの縮小の制限事項： この句は、ビットマップ結合索引またはファンクション索引に対して指定できません。

parallel_clause

PARALLEL 句を使用すると、索引の問合せおよび DML に対するデフォルトの並列度を変更できます。

並列化する索引の制限事項 : この句は、一時表の索引に対して指定できません。

この句の詳細は、16-54 ページの「CREATE TABLE」の「parallel_clause」を参照してください。

参照 : 「パラレル問合せを使用可能にする例 :」(10-80 ページ)

physical_attributes_clause

physical_attributes_clause を使用すると、非パーティション索引、パーティション索引のすべてのパーティションおよびサブパーティション、指定されたパーティション、または指定されたパーティションのすべてのサブパーティションに対するパラメータの値を変更できます。

参照 :

- この句のパラメータの詳細は、16-6 ページの「CREATE TABLE」を参照してください。
- 10-80 ページの「索引の実属性の変更例 :」および 10-80 ページの「MAXEXTENTS の変更例 :」を参照してください。

索引の物理属性の制限事項 : 索引の物理属性には、次の制限事項があります。

- この句は、一時表の索引に対して指定できません。
- 索引の変更中は、PCTUSED パラメータを指定できません。
- PCTFREE パラメータは、*rebuild_clause*、*modify_index_default_attrs* 句または *split_partition_clause* の一部としてのみ指定できます。

storage_clause

storage_clause を使用すると、非パーティション索引、索引パーティション、またはパーティション索引のすべてのパーティションの記憶域パラメータ、あるいはパーティション索引の記憶域パラメータのデフォルト値を変更できます。この句の詳細は、8-41 ページの「storage_clause」を参照してください。

logging_clause

logging_clause を使用すると、索引のロギング属性を変更できます。REBUILD 句も指定すると、この新しい設定は再構築操作に影響します。REBUILD 句でロギングに異なる値を指定した場合、索引および再構築操作のロギング属性として指定された最後のロギング値が使用されます。

索引セグメントには、実表の属性と異なるロギング属性、および同じ実表の他の索引セグメントと異なるロギング属性を指定できます。

索引のログの制限事項 : この句は、一時表の索引に対して指定できません。

参照 :

- この句の詳細は、8-34 ページの「logging_clause」を参照してください。
- パラレル DML の詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

RECOVERABLE | UNRECOVERABLE

これらのキーワードは以前のリリースで非推奨になったもので、それぞれ LOGGING および NOLOGGING に置き換えられています。RECOVERABLE および UNRECOVERABLE は、下位互換性のためにサポートされていますが、LOGGING および NOLOGGING キーワードを使用することをお薦めします。

RECOVERABLE は、パーティション表または LOB の記憶特性の作成時には無効なキーワードです。UNRECOVERABLE は、パーティション表または索引構成表の作成時には無効なキーワードです。また、CREATE INDEX の AS 副問合せ句を使用してのみ指定できます。

rebuild_clause

rebuild_clause を使用すると、既存の索引、あるいはパーティションまたはサブパーティションのいずれかを再構築できます。索引に UNUSABLE のマークが付いている場合、正常に再構築すると USABLE になります。ファンクション索引も使用可能にします。索引の基になるファンクションが存在しない場合、再構築文は正常に実行されません。

> **注意：** 索引構成表の 2 次索引を再構築する場合、Oracle Database は、索引が作成されたときの論理 ROWID に含まれる主キー列を保持します。したがって、COMPATIBLE 初期化パラメータが 10.0.0 未満に設定された状態で索引が作成された場合、再構築された索引には、索引キーと、索引キーには含まれない表の主キー列が含まれます。COMPATIBLE 初期化パラメータが 10.0.0 以上に設定された状態で索引が作成された場合、再構築された索引には、索引キーと、索引キーに含まれる主キー列を含む、表のすべての主キー列が含まれます。

索引の再構築の制限事項： 索引の再構築には、次の制限事項があります。

- 一時表の索引は、再構築できません。
- INVALID のマークが付いているビットマップ索引は再構築できません。制約を削除してからそれを再作成する必要があります。
- パーティション索引全体は、再構築できません。PARTITION 句で説明するとおり、各パーティションまたはサブパーティションを再構築する必要があります。
- 同じ文で *deallocate_unused_clause* と *rebuild_clause* を指定することはできません。
- 索引全体（ALTER INDEX）またはパーティション（ALTER INDEX ... MODIFY PARTITION）に対して、PCTFREE パラメータ値を変更できません。ALTER INDEX 文の他のすべての形式では、PCTFREE を指定できます。
- ドメイン索引の場合
  - 指定できるのは、PARAMETERS 句（索引または索引のパーティション用）または *parallel_clause* のみです。その他の再構築の句は無効です。
  - 索引に IN_PROGRESS のマークが付いていない場合にのみ、索引を再構築できます。
  - 索引に IN_PROGRESS または FAILED のマーク、パーティションに IN_PROGRESS のマークが付いていない場合にのみ、索引パーティションを再構築できます。
- ローカル索引は再構築できませんが、ALTER INDEX ... REBUILD PARTITION でローカル索引のパーティションを再構築できます。
- ハッシュ・パーティションまたはハッシュ・サブパーティションのローカル索引に指定できるパラメータは、TABLESPACE のみです。

PARTITION 句

PARTITION 句を使用すると、索引の 1 つのパーティションを再構築できます。この句は、索引パーティションを別の表領域に移動したり、作成時の物理属性を変更するために使用できます。

ブロック・サイズが異なる表領域のパーティション化されたデータベース・エンティティの記憶域には、制限事項があります。これらの制限事項については、『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。

パーティションの再構築の制限事項： この句は、コンポジット・パーティション表のローカル索引に対して指定できません。かわりに、REBUILD SUBPARTITION 句を使用してください。

> **参照：** パーティションのメンテナンス操作については、『Oracle Database VLDB およびパーティショニング・ガイド』および 10-80 ページの「使用禁止の索引パーティションの再構築例 :」を参照してください。

SUBPARTITION 句

SUBPARTITION 句を使用すると、索引の 1 つのサブパーティションを再構築できます。この句を使用して、索引サブパーティションを他の表領域に移動することもできます。TABLESPACE を指定しないと、サブパーティションは同じ表領域に再構築されます。

ブロック・サイズが異なる表領域のパーティション化されたデータベース・エンティティの記憶域には、制限事項があります。これらの制限事項については、『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。

索引のサブパーティション変更の制限事項： 索引のサブパーティションの変更には、次の制限事項があります。

- パラメータ TABLESPACE、ONLINE および *parallel_clause* 以外は、サブパーティションに対して指定できません。
- リスト・パーティションのサブパーティションは再構築できません。

REVERSE | NOREVERSE

索引ブロックのバイトを逆順に格納するかどうかを示します。

- REVERSE を指定すると、索引の再構築時に、索引ブロックのバイトが逆順で格納され、ROWID が除外されます。
- NOREVERSE を指定すると、索引の再構築時に、逆順にせずに索引ブロックのバイトが格納されます。NOREVERSE キーワードを指定せずに REVERSE 索引を再構築すると、再構築された索引は、逆キーの索引になります。

逆索引の制限事項： 逆索引には、次の制限事項があります。

- ビットマップ索引または索引構成表は逆順には格納できません。
- パーティションまたはサブパーティションに対して、REVERSE または NOREVERSE を指定できません。

> **参照：** 「索引ブロックの逆順格納例 :」（10-79 ページ）

parallel_clause

parallel_clause を使用すると、索引の再構築をパラレル化できます。

> **参照：** 「索引のパラレル再構築例 :」（10-79 ページ）

TABLESPACE 句

再構築された索引、索引パーティションまたは索引サブパーティションが格納される表領域を指定します。デフォルトは、再構築の前に索引またはパーティションが格納されていた表領域です。

key_compression

COMPRESS を指定すると、キー列値の繰返し項目を排除するキー圧縮が使用可能になります。*integer* を使用して、接頭辞の長さ（圧縮する接頭辞列数）を指定します。

- 一意の索引の場合、接頭辞の長さの有効範囲は、1 ～キー列の数から 1 を引いた数までです。接頭辞の長さのデフォルト値は、キー列の数から 1 を引いた数です。
- 一意でない索引の場合、接頭辞の長さの有効範囲は、1 ～キー列の数までです。接頭辞の長さのデフォルト値は、キー列の数です。

Oracle Database では、索引（一意でない索引または 2 列以上の一意索引）が圧縮されます。パーティション索引の圧縮を使用する場合、索引では圧縮が索引レベルで有効になっている必要があります。

NOCOMPRESS を指定すると、キー圧縮が使用禁止になります。これはデフォルトです。

キー圧縮の制限事項： COMPRESS は、ビットマップ索引に対して指定できません。

ONLINE 句

ONLINE を指定すると、表またはパーティションの DML 操作を索引の再構築中に可能にするかどうかを指定できます。

オンライン索引の制限事項： オンライン索引には、次の制限事項があります。

- オンライン索引の作成中は、パラレル DML はサポートされません。ONLINE を指定し、続いてパラレル DML 文を発行すると、Oracle Database はエラーを戻します。
- ビットマップ結合索引またはクラスタ索引には、ONLINE を指定できません。
- 索引構成表の一意でない 2 次索引の場合、索引構成表内の索引キー列の数と論理 ROWID の主キー列の数の合計は、32 以下にする必要があります。論理 ROWID は、索引キーに含まれる列を除外します。

logging_clause

ALTER INDEX ... REBUILD 操作をログに記録するかどうかを指定します。

この句の詳細は、8-34 ページの「logging_clause」を参照してください。

PARAMETERS 句

この句は、トップレベルの ALTER INDEX 文のドメイン索引および *rebuild_clause* で使用される場合のドメイン索引および XMLIndex 索引に対してのみ有効です。

- ドメイン索引の場合、PARAMETERS 句によって、未解析のまま適切な ODCI 索引タイプ・ルーチンに渡されたパラメータ文字列が指定されます。
- XMLIndex 索引の場合、PARAMETERS 句によって、XMLIndex 実装を定義するパラメータ文字列が指定されます。

パラメータ文字列の最大長は 1,000 文字です。

索引全体を変更または再構築する場合、文字列は索引レベルのパラメータを参照する必要があります。索引のパーティションを再構築する場合、文字列はパーティション・レベルのパラメータを参照する必要があります。

index に UNUSABLE のマークが付いている場合、パラメータを変更するのみでは USABLE にはなりません。UNUSABLE のマークが付いた索引を使用可能にするには、その索引を再構築する必要があります。

Oracle Text がインストール済の場合、Oracle Text 固有のパラメータを使用する Oracle Text のドメイン索引を再構築することができます。パラメータの詳細は、『Oracle Text リファレンス』を参照してください。

PARAMETERS 句の制限事項： PARAMETERS 句には、次の制限事項があります。

- この句は、ドメイン索引または XMLIndex 索引に対してのみ指定できます。
- *index* に IN_PROGRESS または FAILED のマークが付いておらず、IN_PROGRESS のマークが付いた索引パーティションが存在せず、変更対象のパーティションに FAILED のマークが付いていない場合にのみ、索引パーティションを変更できます。

参照：

- ドメイン索引の索引タイプ・ルーチンの詳細は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。
- *XMLIndex_parameters_clause* の構文およびセマンティクスを含む XMLIndex の詳細は、『Oracle XML DB 開発者ガイド』を参照してください。
- ドメイン索引の詳細は、14-50 ページの「CREATE INDEX」を参照してください。

COMPILE 句

この句は、ドメイン索引に対してのみ有効です。この句を使用すると、無効なドメイン索引を明示的に再コンパイルできます。この句は、主に、システム管理されたドメイン索引をサポートするように基礎となる索引タイプが変更され、その結果として既存のドメイン索引が INVALID とマークされた場合に有効です。これにより、この ALTER INDEX 文によって、ドメイン索引はユーザー管理されたドメイン索引からシステム管理されたドメイン索引に移行されます。

参照： システム管理されたドメイン索引の作成の詳細は、14-75 ページの「CREATE INDEXTYPE」の「storage_table_clause」および『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。

ENABLE 句

ENABLE は、索引が使用するユーザー定義ファンクションが削除または置換されたために使用禁止になったファンクション索引のみに適用されます。次の条件が該当する場合、この句によって、このような索引が使用可能になります。

- ファンクションが現在有効な場合
- 現行のファンクションの署名が、索引が作成された場合のファンクションの署名と一致する場合
- ファンクションに現在 DETERMINISTIC のマークが付いている場合

ファンクション索引を使用可能にする場合の制限事項： ENABLE と同じ文には、ALTER INDEX の他の句を指定できません。

DISABLE 句

DISABLE は、ファンクション索引のみに適用されます。この句を使用してファンクション索引を使用禁止にします。たとえば、ファンクションの本体を処理する場合に、これを行います。その後、ENABLE キーワードを使用して、索引を再構築、または別の ALTER INDEX 文を指定できます。

UNUSABLE

索引、索引パーティションまたは索引サブパーティションに UNUSABLE のマークを付けるには、UNUSABLE を指定します。使用禁止の索引を使用可能にする場合、再構築するか、または削除して再作成する必要があります。1 つのパーティションに UNUSABLE のマークが付いている場合も、同じ索引の他のパーティションは有効です。その索引を必要とする文が使用禁止のパーティションにアクセスしない場合、その文を実行できます。また、使用禁止のパーティションは、分割または名前を変更してから再構築できます。詳細は、14-67 ページの「CREATE INDEX ... UNUSABLE」を参照してください。

索引への使用禁止のマーク付けの制限事項： この句は、一時表の索引に対して指定できません。

VISIBLE | INVISIBLE

この句を使用すると、オプティマイザで索引を参照可能にするかどうかを指定できます。この句の詳細は、14-61 ページの CREATE INDEX の「VISIBLE | INVISIBLE」を参照してください。

RENAME 句

この句を指定すると、索引の名前を変更できます。*new_index_name* は単一の識別子で、スキーマ名は含まれません。

索引の名前の変更の制限事項： ドメイン索引の場合、*index* および *index* のパーティションに IN_PROGRESS または FAILED のマークが付いていると無効になります。

> **参照：** 「索引の名前の変更例：」（10-80 ページ）

COALESCE 句

COALESCE を指定すると、ブロックを再利用するために、索引ブロックの内容を空きブロックにマージできます（可能な場合）。

索引ブロックの結合の制限事項： 索引ブロックの結合には、次の制限事項があります。

- この句は、一時表の索引に対して指定できません。
- この句は、索引構成表の主キー索引に対して指定できません。かわりに ALTER TABLE の COALESCE 句を使用します。

> **参照：**
>
> - 領域管理および索引の結合の詳細は、『Oracle Database 管理者ガイド』を参照してください。
> - 索引構成表の領域の結合の詳細は、12-39 ページの「COALESCE 句」を参照してください。
> - 索引セグメントを縮小化する他の方法については、12-35 ページの「shrink_clause」を参照してください。

MONITORING USAGE | NOMONITORING USAGE

この句を使用すると、索引の使用を監視するかどうかを決定できます。

- MONITORING USAGE を指定すると、索引の監視が開始されます。まず、索引の使用に関する既存の情報が削除され、ALTER INDEX ... NOMONITORING USAGE 文が次に実行されるまで、索引の使用が監視されます。
- NOMONITORING USAGE を指定すると、索引の監視を終了できます。

この ALTER INDEX ... NOMONITORING USAGE 文が発行された後、索引が使用されたかどうかを調べるには、V$OBJECT_USAGE 動的パフォーマンス・ビューの USED 列を問い合せます。

> **参照：** データ・ディクショナリおよび動的パフォーマンス・ビューの詳細は、『Oracle Database リファレンス』を参照してください。

UPDATE BLOCK REFERENCES 句

UPDATE BLOCK REFERENCES 句は、索引構成表の通常のドメイン索引に対してのみ有効です。この句を指定すると、主キーによって識別されるブロックの適切なデータベース・アドレスとともに索引行の一部として格納され、失効したと推測されるすべてのデータ・ブロック・アドレスを更新することができます。

ドメイン索引では、AlterIndexUpdBlockRefs に設定された alter_option パラメータで ODCIIndexAlter ルーチンが実行されます。このルーチンはカートリッジ・コードを使用可能にし、失効したと推測される索引のデータ・ブロック・アドレスを更新します。

UPDATE BLOCK REFERENCES の制限事項 : この句を ALTER INDEX の他の句と組み合せることはできません。

alter_index_partitioning

ALTER INDEX 文のパーティション化句は、パーティション索引に対してのみ有効です。

ブロック・サイズが異なる表領域のパーティション化されたデータベース・エンティティの記憶域には、制限事項があります。これらの制限事項については、『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。

索引のパーティション変更の制限事項 : 索引のパーティション変更には、次の制限事項があります。

- これらの句は、一時表の索引に対して指定できません。
- ベース索引に対するいくつかの操作を 1 つの ALTER INDEX 文にまとめることはできますが (RENAME および REBUILD は除く)、パーティション操作を、他のパーティション操作またはベース索引に対する操作と組み合せることはできません。

modify_index_default_attrs

パーティション索引のデフォルト属性に新しい値を指定します。

パーティションのデフォルト属性の変更の制限事項 : ハッシュ・パーティション・グローバル索引またはハッシュ・パーティション表の索引の属性には、TABLESPACE のみを指定できます。

TABLESPACE 索引の新規パーティション、または索引パーティションのサブパーティションに対して、デフォルトの表領域を指定します。

logging_clause パーティション索引または索引パーティションのデフォルトのロギング属性を指定します。

この句の詳細は、8-34 ページの「logging_clause」を参照してください。

FOR PARTITION FOR PARTITION 句を使用すると、コンポジット・パーティション表にあるローカル索引のパーティションのサブパーティションに対して、デフォルトの属性を指定できます。

FOR PARTITION の制限事項 : リスト・パーティションに対して FOR PARTITION を指定することはできません。

参照 : 「デフォルト属性の変更例 :」(10-81 ページ)

add_hash_index_partition

この句を使用すると、ハッシュ・パーティション・グローバル索引にパーティションを追加できます。Oracle Database は、ハッシュ・パーティションを追加し、ハッシュ・ファンクションによって索引の既存のハッシュ・パーティションから再ハッシュされた索引エントリをそのハッシュ・パーティションに移入します。パーティション名を省略すると、SYS_Pn の形式でパーティション名が割り当てられます。TABLESPACE 句を省略すると、その索引に対して指定された表領域にパーティションが配置されます。索引の表領域が指定されていない場合、パーティションは、ユーザーのデフォルトの表領域 (指定されている場合) またはシステムのデフォルトの表領域に配置されます。

modify_index_partition

modify_index_partition 句を使用すると、索引パーティション *partition* またはそのサブパーティションの実物理属性、ロギング属性または記憶特性を変更できます。ハッシュ・パーティション・グローバル索引の場合、この句に指定できる副次句は UNUSABLE のみです。

COALESCE この句を指定すると、ブロックを再利用するために、索引パーティション・ブロックの内容を空きブロックにマージできます（可能な場合）。

UPDATE BLOCK REFERENCES UPDATE BLOCK REFERENCES 句は、索引構成表の通常の索引に対してのみ有効です。この句を使用すると、2 次索引パーティションに格納されている、失効したと推測されるすべてのデータ・ブロック・アドレスを更新することができます。

UPDATE BLOCK REFERENCES の制限事項 : この句には、次の制限事項があります。

- ハッシュ・パーティション表の索引に対して、*physical_attributes_clause* を指定することはできません。
- ALTER INDEX の他の句とともに、UPDATE BLOCK REFERENCES を指定することはできません。

注意 : 索引がコンポジット・パーティション表のローカル索引である場合、ここで指定した変更は、以前に索引のサブパーティションに対して指定したすべての属性を上書きします。また、このパーティションの将来のサブパーティションに対する属性のデフォルト値を設定します。サブパーティションの属性を上書きせずにパーティションのデフォルトの属性を変更する場合は、ALTER TABLE ... MODIFY DEFAULT ATTRIBUTES OF PARTITION を使用します。

参照 : 「索引への使用禁止のマーク付けの例 :」（10-80 ページ）

UNUSABLE 句 索引パーティションに対するこの句の機能は、索引全体の場合と同じです。詳細は、10-75 ページの「UNUSABLE」を参照してください。

key_compression この句は、コンポジット・パーティション索引に対してのみ有効です。この句を使用すると、パーティションおよびそのパーティションのすべてのサブパーティションの圧縮属性を変更できます。Oracle Database は、パーティション内の各索引サブパーティションに UNUSABLE のマークを付けます。その後、これらのサブパーティションを再構築する必要があります。パーティションにキー圧縮を指定するには、まず表にキー圧縮を指定しておく必要があります。この句は、パーティション・レベルでのみ指定できます。個々のサブパーティションの圧縮属性は変更できません。

非コンポジット索引パーティションにもこの句を指定できます。ただし、非コンポジット・パーティションには、再構築と圧縮属性の設定を 1 つの手順で行う *rebuild_clause* を使用する方が効率的です。

rename_index_partition

rename_index_partition 句を使用すると、索引パーティションまたは索引サブパーティションの名前を *new_name* に変更できます。

索引パーティションの名前の変更の制限事項 : 索引パーティションの名前の変更には、次の制限事項があります。

- リスト・パーティションのサブパーティションは名前を変更できません。
- ドメイン索引のパーティションの場合、*index* に IN_PROGRESS または FAILED のマークを付けることはできません。また、パーティションに IN_PROGRESS、名前を変更するパーティションに FAILED のマークを付けることはできません。

参照 : 「索引パーティションの名前の変更例 :」（10-80 ページ）

drop_index_partition

drop_index_partition 句を使用すると、グローバル・パーティション索引からパーティションとその中のデータを削除できます。グローバル索引のパーティションを削除する場合、その索引の次のパーティションに UNUSABLE のマークが付けられます。グローバル索引の最上位のパーティションは削除できません。

参照：「索引パーティションの削除例：」（10-81 ページ）

split_index_partition

split_index_partition 句を使用すると、レンジ・パーティション・グローバル索引のパーティションを 2 つのパーティションに分割し、新しいパーティションを索引に追加できます。この句は、ハッシュ・パーティション・グローバル索引に対しては無効です。かわりに、*add_hash_index_partition* 句を使用してください。

UNUSABLE のマークが付いたパーティションを分割すると、2 つのパーティションが生成されますが、その両方に UNUSABLE のマークが付けられます。このようなパーティションは、使用前に再構築する必要があります。

使用可能なパーティションを分割した場合、索引データが入っている 2 つのパーティションが生成されます。両方の新規パーティションは使用可能です。

AT 句　*split_partition_1* に新しい上限（境界は含まない）を指定します。*value_list* の値は、*partition_name_old* の分割前のパーティション境界より小さく、その次の最小のパーティション（そのようなパーティションがある場合）のパーティション境界より大きい値である必要があります。

INTO 句　分割の結果、生成される 2 つのパーティションの名前と物理属性を任意に指定します。

参照：「パーティションの分割例：」（10-81 ページ）

coalesce_index_partition

この句は、ハッシュ・パーティション・グローバル索引に対してのみ有効です。索引パーティションの数が 1 つ減少します。結合するパーティションは、ハッシュ・ファンクションの要件に基づいて選択されます。この句を使用するのは、選択されたパーティションの索引エントリを残りのパーティションのいずれかに分散した後で、その選択されたパーティションを削除する場合です。

modify_index_subpartition

modify_index_subpartition 句を使用すると、コンポジット・パーティション表にあるローカル索引のサブパーティションに対する領域の UNUSABLE のマーク付け、割当てまたは割当て解除が可能になります。このようなサブパーティションの他のすべての属性は、パーティション・レベルのデフォルトの属性から継承されます。

例

索引ブロックの逆順格納例：　次の文は、索引ブロックのバイトが逆順に格納されるように、索引 ord_customer_ix（14-68 ページの「索引の作成例：」で作成）を再構築します。

```
ALTER INDEX ord_customer_ix REBUILD REVERSE;
```

索引のパラレル再構築例：　次の文は、パラレル実行プロセスを使用して、既存の索引のスキャンおよび新しい索引の構築を行い、既存の索引から索引を再構築します。

```
ALTER INDEX ord_customer_ix REBUILD PARALLEL;
```

索引の実属性の変更例： 次の文は、同じ索引に将来追加されるデータ・ブロックが、5 つの初期トランザクション・エントリと 100KB の増分エクステントを使用するように、索引 oe.cust_lname_ix を変更します。

```
/* Unless you change the default tablespace of sample user oe,
   or specify different tablespace storage for the index, this
   example fails because the default tablespace originally assigned
   to oe is locally managed.
*/
ALTER INDEX oe.cust_lname_ix
    INITRANS 5
    STORAGE (NEXT 100K);
```

索引 oe.cust_lname_ix がパーティション化されている場合、この文は将来追加される索引のパーティションのデフォルト属性も変更します。将来追加されるパーティションでは、5 つの初期トランザクション・エントリと 100KB の増分エクステントが使用されます。

パラレル問合せを使用可能にする例： 次の文は、索引 upper_ix（14-69 ページの「ファンクション索引の作成例：」で作成）に対するスキャンがパラレル化されるように、索引のパラレル属性を設定します。

```
ALTER INDEX upper_ix PARALLEL;
```

索引の名前の変更例： 次の文は、索引の名前を変更します。

```
ALTER INDEX upper_ix RENAME TO upper_name_ix;
```

索引への使用禁止のマーク付けの例： 次の例は、cost_ix 索引（14-70 ページの「レンジ・パーティション・グローバル索引の作成例：」で作成）を使用します。この索引のパーティション p1 は、10-81 ページの「索引パーティションの削除例：」で削除されています。最初の文は、索引パーティション p2 に UNUSABLE のマークを付けます。

```
ALTER INDEX cost_ix
   MODIFY PARTITION p2 UNUSABLE;
```

次の文は、索引 cost_ix 全体に UNUSABLE のマークを付けます。

```
ALTER INDEX cost_ix UNUSABLE;
```

使用禁止の索引パーティションの再構築例： 次の文は、cost_ix 索引のパーティション p2 および p3 を再構築し、索引をもう一度使用可能にします。パーティション p3 の再構築はログに記録されません。

```
ALTER INDEX cost_ix
   REBUILD PARTITION p2;
ALTER INDEX cost_ix
   REBUILD PARTITION p3 NOLOGGING;
```

MAXEXTENTS の変更例： 次の文は、パーティション p3 のエクステントの最大数を変更し、ロギング属性を変更します。

```
/* This example will fail if the tablespace in which partition p3
   resides is locally managed.
*/
ALTER INDEX cost_ix MODIFY PARTITION p3
   STORAGE(MAXEXTENTS 30) LOGGING;
```

索引パーティションの名前の変更例： 次の文は、cost_ix 索引（14-70 ページの「レンジ・パーティション・グローバル索引の作成例：」で作成）の索引パーティションの名前を変更します。

```
ALTER INDEX cost_ix
   RENAME PARTITION p3 TO p3_Q3;
```

パーティションの分割例： 次の文は、索引 cost_ix のパーティション p2（14-70 ページの「レンジ・パーティション・グローバル索引の作成例：」で作成）を p2a と p2b に分割します。

```
ALTER INDEX cost_ix
   SPLIT PARTITION p2 AT (1500)
   INTO ( PARTITION p2a TABLESPACE tbs_01 LOGGING,
          PARTITION p2b TABLESPACE tbs_02);
```

索引パーティションの削除例： 次の文は、cost_ix 索引から索引パーティション p1 を削除します。

```
ALTER INDEX cost_ix
   DROP PARTITION p1;
```

デフォルト属性の変更例： 次の文は、ローカル・パーティション索引 prod_idx（14-71 ページの「ハッシュ・パーティション表の索引の作成例：」で作成）のデフォルトの属性を変更します。将来追加されるパーティションでは、5 つの初期トランザクション・エントリと 100KB の増分エクステントが使用されます。

```
ALTER INDEX prod_idx
      MODIFY DEFAULT ATTRIBUTES INITRANS 5 STORAGE (NEXT 100K);
```

ALTER INDEXTYPE

用途

ALTER INDEXTYPE 文を使用すると、索引タイプの演算子を追加または削除したり、実装タイプを変更したり、索引タイプのプロパティを変更することができます。

前提条件

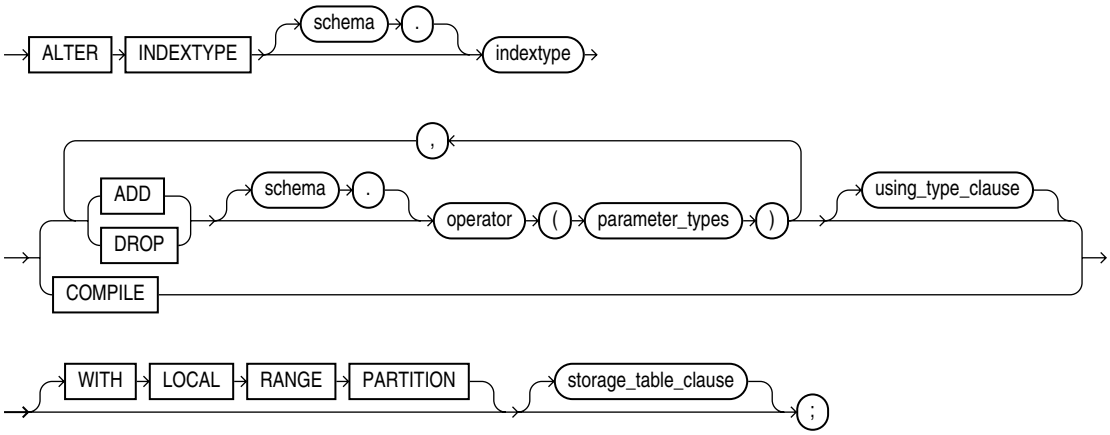
索引タイプが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY INDEXTYPE システム権限が必要です。

新しい演算子を追加する場合は、その演算子に対する EXECUTE オブジェクト権限が必要です。

実装タイプを変更する場合は、新しい実装タイプに対する EXECUTE オブジェクト権限が必要です。

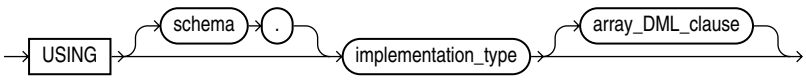
構文

***alter_indextype*::=**



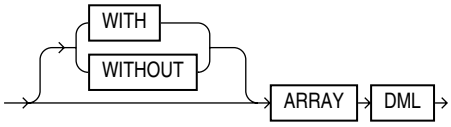
(10-82 ページの using_type_clause::=、10-84 ページの storage_table_clause を参照)

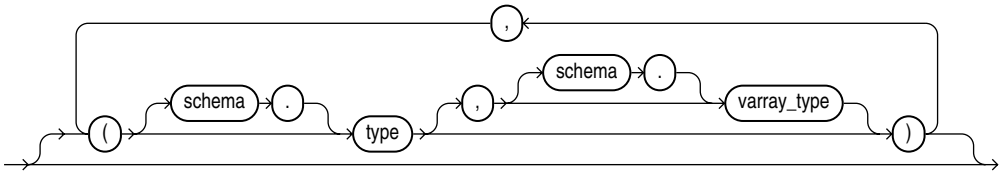
using_type_clause::=



(10-83 ページの array_DML_clause::= を参照)

array_DML_clause::=





storage_table_clause::=

セマンティクス

schema

索引タイプが存在するスキーマ名を指定します。*schema* を指定しない場合、索引タイプは自分のスキーマ内にあるとみなされます。

indextype

変更する索引タイプの名前を指定します。

ADD | DROP

ADD または DROP 句を使用すると、演算子を追加または削除できます。

削除には特別な権限は必要ありません。

- *schema* には、演算子を含むスキーマを指定します。*schema* を指定しない場合、その演算子は自分のスキーマにあるとみなされます。
- *operator* には、索引タイプによってサポートされる演算子の名前を指定します。

  この句に指定するすべての演算子は有効な演算子である必要があります。
- *parameter_type* には、演算子へのパラメータ・タイプを指定します。

using_type_clause

USING 句を使用すると、索引タイプを実装する新しいタイプを指定できます。

array_DML_clause

この句を使用すると、ODCIIndexInsert メソッドで配列インタフェースをサポートする索引タイプを変更できます。

type* および *varray_type 索引付けする列のデータ型がユーザー定義のオブジェクト型である場合、この句を指定して、Oracle が *type* の列値を保持するために使用する VARRAY の *varray_type* を識別する必要があります。索引タイプで型のリストがサポートされている場合、対応する VARRAY 型のリストを指定できます。*type* または *varray_type* で *schema* を省略した場合、型が自分のスキーマ内に定義されているとみなされます。

索引付けする列のデータ型が組込みシステム型である場合、その索引タイプに指定された VARRAY 型は、システムで定義された ODCI 型よりも優先されます。

COMPILE

この句を使用すると、索引タイプを明示的に再コンパイルできます。通常、索引タイプは自動的に再コンパイルされるため、この句は一部のアップグレード操作の後でのみ指定する必要があります。

storage_table_clause

この句では、索引タイプを変更する場合に、索引タイプを作成するときと同じ動作になります。詳細は、14-75 ページの「CREATE INDEXTYPE」の「storage_table_clause」を参照してください。

WITH LOCAL RANGE PARTITION

この句では、索引タイプを変更する場合に、索引タイプを作成するときと同じ動作になります。詳細は、14-75 ページの「CREATE INDEXTYPE」の句「WITH LOCAL RANGE PARTITION」を参照してください。

例

索引タイプの変更例： 次の例は、position_indextype 索引タイプ（14-75 ページの「索引タイプの作成例：」で作成）をコンパイルします。

```
ALTER INDEXTYPE position_indextype COMPILE;
```

ALTER JAVA

用途

ALTER JAVA 文を使用すると、Java クラス・スキーマ・オブジェクトの変換、または Java ソース・スキーマ・オブジェクトのコンパイルを強制実行できます （Java 名に対するすべての外部参照を他のクラスと対応付ける前に、Java クラスのメソッドをコールすることはできません）。

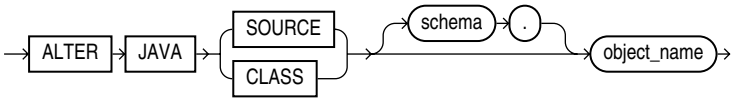
> **参照：** Java クラスの変換および Java ソースのコンパイルの詳細は、『Oracle Database Java 開発者ガイド』を参照してください。

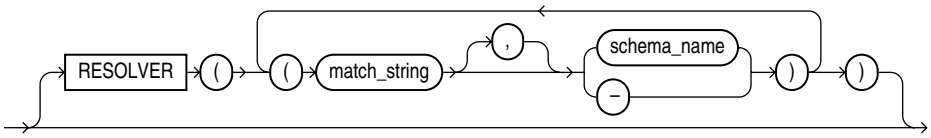
前提条件

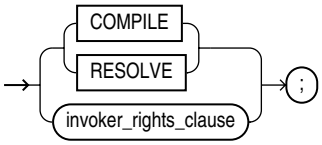
Java ソースまたはクラスが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY PROCEDURE システム権限が必要です。さらに、Java クラスに対する EXECUTE オブジェクト権限も必要です。

構文

***alter_java*::=**

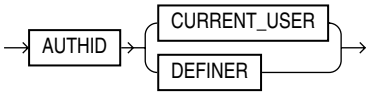






（10-85 ページの invoker_rights_clause::= を参照）

invoker_rights_clause::=



セマンティクス

JAVA SOURCE

ALTER JAVA SOURCE を使用すると、Java ソース・スキーマ・オブジェクトをコンパイルできます。

JAVA CLASS

ALTER JAVA CLASS を使用すると、Java ソース・スキーマ・オブジェクトを変換できます。

object_name

以前作成した Java クラスまたはソース・スキーマ・オブジェクトを指定します。小文字、または大文字と小文字を組み合せた名前を付けるには、二重引用符を使用してください。

RESOLVER

RESOLVER 句を使用すると、Java クラスまたはソースが作成されたときに指定したマッピング・ペアを使用して、完全に指定された参照用の Java 名に対するスキーマの検索方法を指定できます。

> **参照：** 14-76 ページの「CREATE JAVA」および 10-86 ページの「Java クラスの変換例 :」を参照してください。

RESOLVE | COMPILE

RESOLVE および COMPILE は、同義のキーワードです。これらの句を使用すると、プライマリ Java クラス・スキーマ・オブジェクトの変換を指定できます。

- クラスに適用された場合、他のクラス・スキーマ・オブジェクトに対する参照名に変換されます。
- ソースに適用された場合、ソースがコンパイルされます。

invoker_rights_clause

invoker_rights_clause を使用すると、クラスを定義したユーザーのスキーマ内で、そのユーザーの権限を使用してクラスのメソッドを実行するか、または、CURRENT_USER のスキーマ内で、そのユーザーの権限を使用してクラスのメソッドを実行するかを指定できます。

また、この句は、問合せ、DML 操作、およびその型のメンバー・ファンクションおよびプロシージャ内の動的 SQL 文の外部名の変換方法も定義します。

AUTHID CURRENT_USER CURRENT_USER を指定すると、クラスのメソッドが CURRENT_USER 権限で実行されることを指定できます。この句はデフォルトで、**実行者権限クラス**を作成します。

また、この句は、問合せ、DML 操作、および動的 SQL 文の外部名を CURRENT_USER のスキーマで変換することも指定します。他のすべての文における外部名は、メソッドを含むスキーマで変換します。

AUTHID DEFINER DEFINER を指定すると、クラスを定義したユーザーの権限を使用してそのクラスのメソッドを実行できます。

さらに、メソッドのあるスキーマ内で外部名を変換するかどうかを指定します。

> **参照：** CURRENT_USER の判断方法については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

例

Java クラスの変換例： 次の文は、Java クラスを強制変換します。

```
ALTER JAVA CLASS "Agent"
   RESOLVER (("/usr/bin/bfile_dir/*" pm)(* public))
   RESOLVE;
```

11

SQL 文 : ALTER MATERIALIZED VIEW ～ ALTER SYSTEM

この章では、次の SQL 文について説明します。

- ALTER MATERIALIZED VIEW
- ALTER MATERIALIZED VIEW LOG
- ALTER OPERATOR
- ALTER OUTLINE
- ALTER PACKAGE
- ALTER PROCEDURE
- ALTER PROFILE
- ALTER RESOURCE COST
- ALTER ROLE
- ALTER ROLLBACK SEGMENT
- ALTER SEQUENCE
- ALTER SESSION
- ALTER SYSTEM

ALTER MATERIALIZED VIEW

用途

マテリアライズド・ビューは、問合せ結果を含むデータベース・オブジェクトです。問合せの FROM 句には、表、ビューおよびその他のマテリアライズド・ビューを指定できます。これらをあわせて、**マスター表**（レプリケーション用語）または**ディテール表**（データ・ウェアハウス用語）といいます。このマニュアルでは、マスター表という用語を使用します。マスター表が格納されているデータベースを**マスター・データベース**といいます。

ALTER MATERIALIZED VIEW 文を使用すると、既存のマテリアライズド・ビューを次の方法で変更できます。

- 記憶特性を変更します。
- リフレッシュ方法、モードまたは時間を変更します。
- 別のタイプのマテリアライズド・ビューになるように構造を変更します。
- クエリー・リライトを使用可能または使用禁止にします。

注意： 下位互換性を保つために、MATERIALIZED VIEW のかわりにキーワード SNAPSHOT もサポートされています。

参照：

- マテリアライズド・ビューの作成の詳細は、15-4 ページの「CREATE MATERIALIZED VIEW」を参照してください。
- レプリケーション環境でのマテリアライズド・ビューの詳細は、『Oracle Database アドバンスト・レプリケーション』を参照してください。
- データ・ウェアハウス環境でのマテリアライズド・ビューの詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

前提条件

マテリアライズド・ビューを変更するために必要な権限は、次のように直接付与される必要があります。

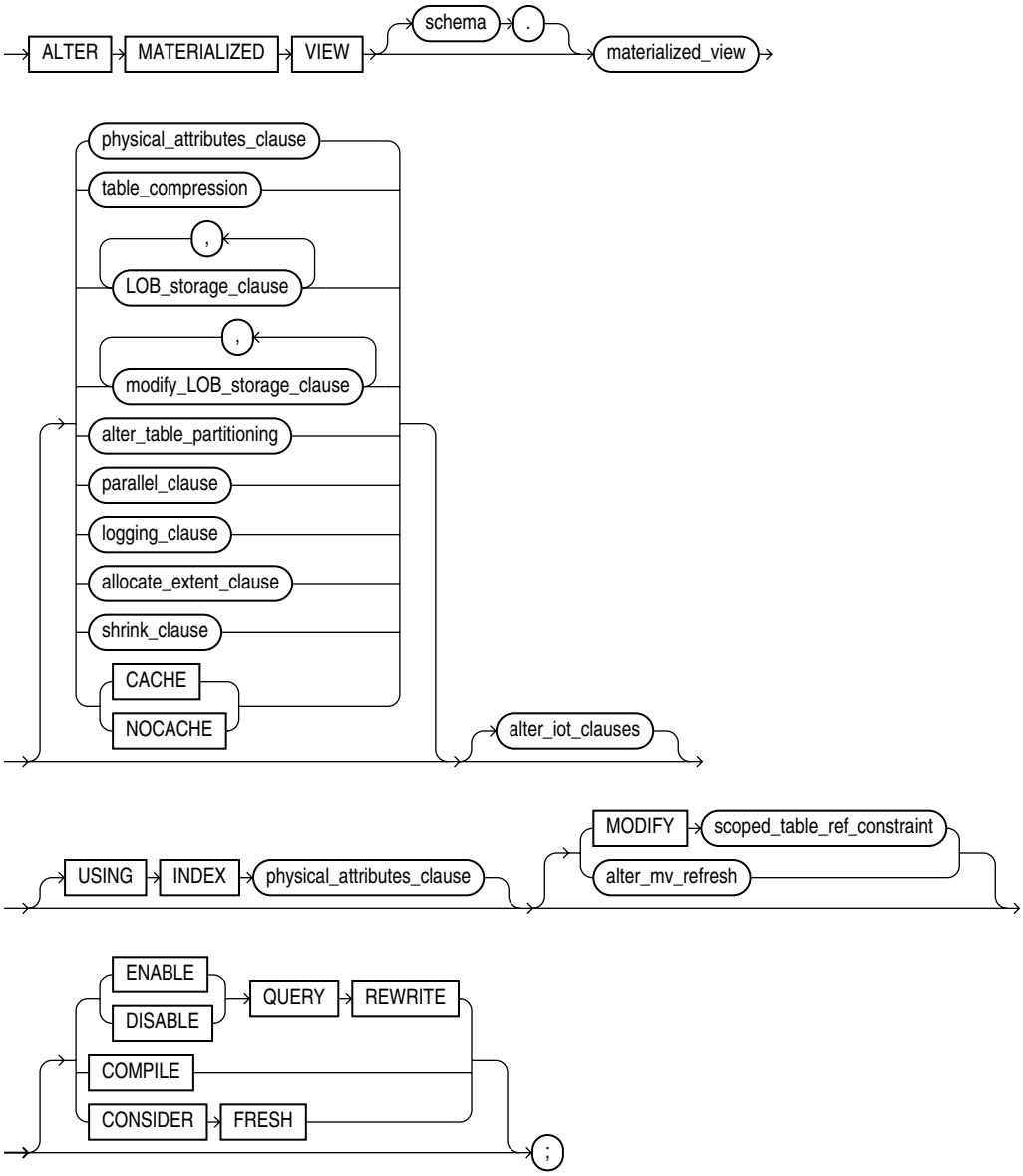
マテリアライズド・ビューが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY MATERIALIZED VIEW システム権限が必要です。

クエリー・リライトでマテリアライズド・ビューを使用可能にする場合、次の条件が必要です。

- マテリアライズド・ビュー内のすべてのマスター表が自分のスキーマ内にある場合、QUERY REWRITE 権限が必要です。
- いずれかのマスター表が別のスキーマ内にある場合、GLOBAL QUERY REWRITE 権限が必要です。
- マテリアライズド・ビューが別のユーザーのスキーマ内にある場合、ユーザーおよびそのスキーマ所有者の両方に、前述の適切な QUERY REWRITE 権限が必要です。また、マテリアライズド・ビューの所有者は、マテリアライズド・ビュー所有者が所有しないすべてのマスター表への SELECT 権限を持っている必要があります。

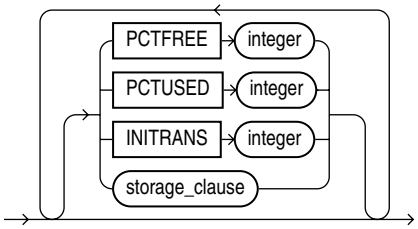
構文

***alter_materialized_view*::=**



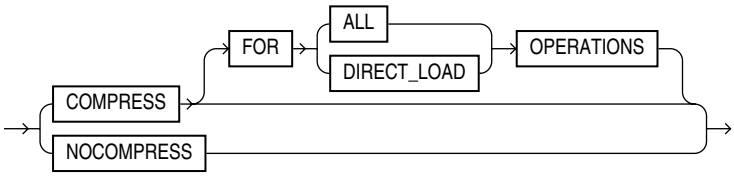
（「ALTER TABLE」の項にある 11-4 ページの physical_attributes_clause::=、
11-4 ページの table_compression::=、11-4 ページの LOB_storage_clause::=、
11-5 ページの modify_LOB_storage_clause::=、12-18 ページの alter_table_partitioning::=、
11-6 ページの parallel_clause::=、11-6 ページの logging_clause::=、
11-6 ページの allocate_extent_clause::=、11-7 ページの alter_iot_clauses::=、
11-8 ページの scoped_table_ref_constraint::=、11-8 ページの alter_mv_refresh::= を参照）

physical_attributes_clause::=

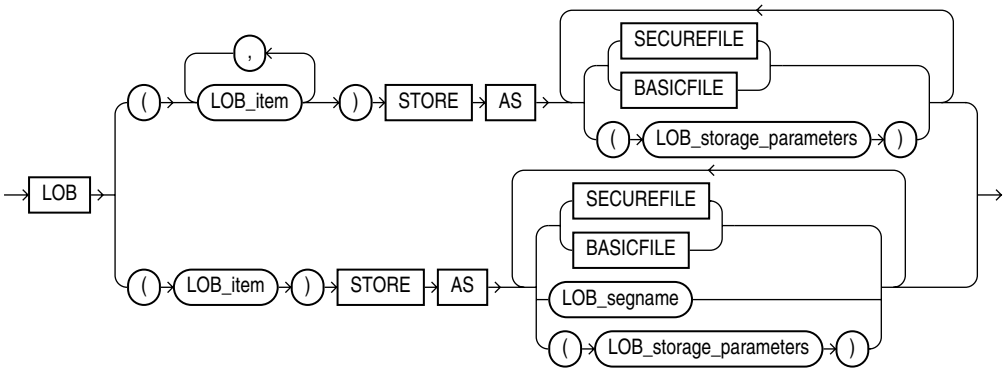


(8-44 ページの storage_clause::= を参照)

table_compression::=

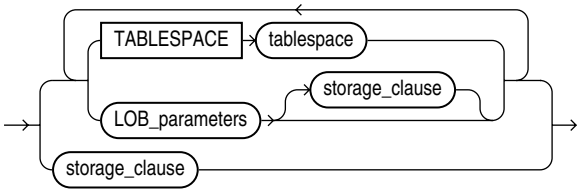


LOB_storage_clause::=



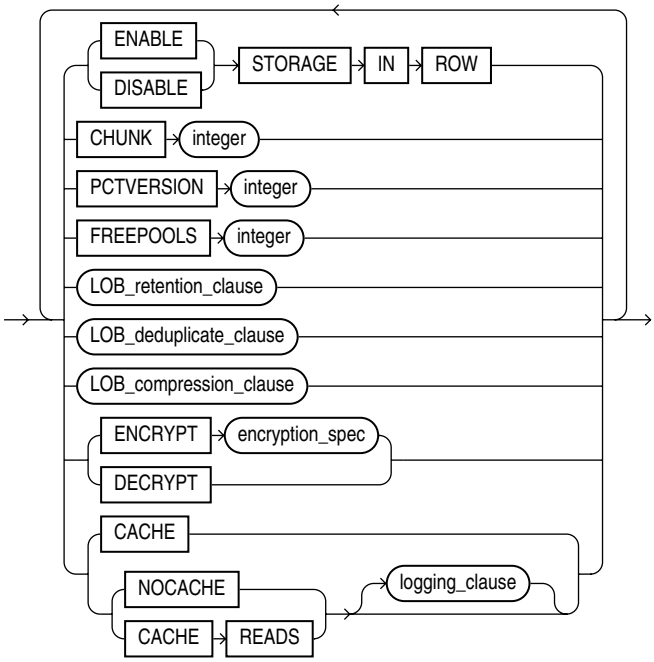
(11-4 ページの LOB_storage_parameters::= を参照)

***LOB_storage_parameters*::=**



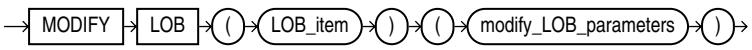
(11-5 ページの LOB_parameters::=、8-44 ページの storage_clause::= を参照)

***LOB_parameters*::=**



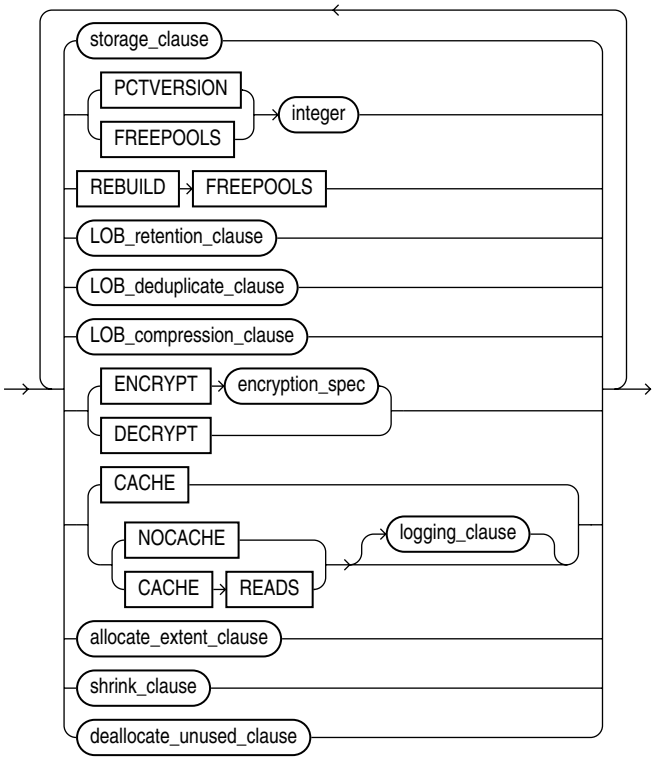
（8-44 ページの storage_clause::=、8-34 ページの logging_clause::= を参照）

modify_LOB_storage_clause::=



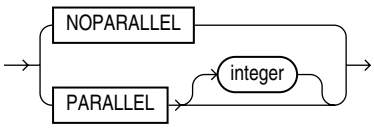
（11-6 ページの modify_LOB_parameters::= を参照）

***modify_LOB_parameters*::=**

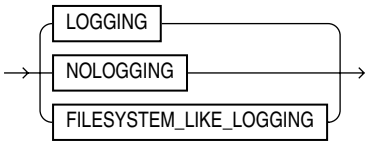


（8-44 ページの storage_clause::=、16-13 ページの LOB_retention_clause::=、16-13 ページの LOB_compression_clause::=、8-34 ページの logging_clause::=、11-6 ページの allocate_extent_clause::=、11-7 ページの shrink_clause::=、11-7 ページの deallocate_unused_clause::= を参照）

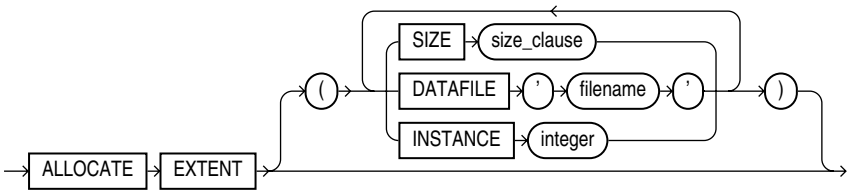
parallel_clause::=



***logging_clause*::=**

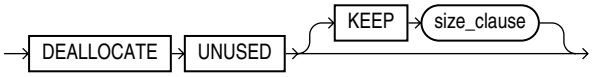


allocate_extent_clause::=



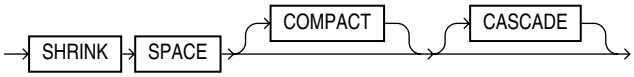
（8-42 ページの size_clause::= を参照）

***deallocate_unused_clause*::=**

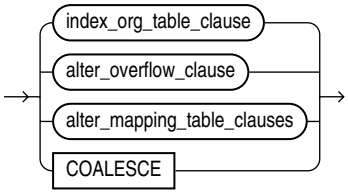


（8-42 ページの size_clause::= を参照）

shrink_clause::=

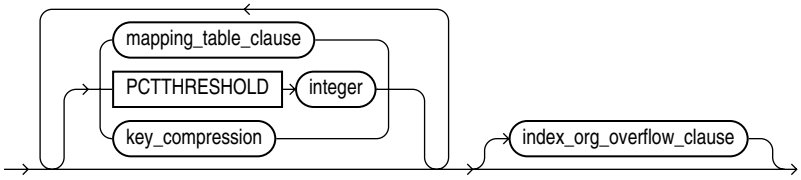


alter_iot_clauses::=



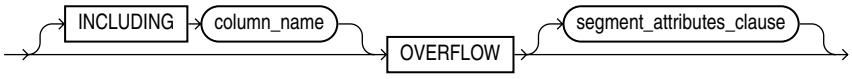
（11-7 ページの index_org_table_clause::=、11-7 ページの alter_overflow_clause::= を参照。*alter_mapping_table_clauses* は、マテリアライズド・ビューではサポートされていません。）

***index_org_table_clause*::=**



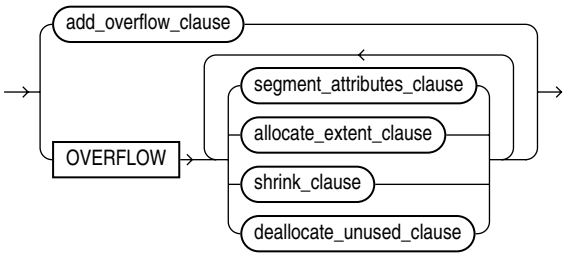
（*mapping_table_clause* および *key_compression* は、マテリアライズド・ビューではサポートされていません。11-7 ページの index_org_overflow_clause::= を参照）

***index_org_overflow_clause*::=**



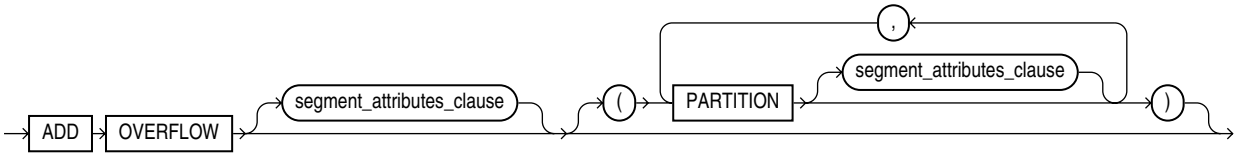
（「ALTER TABLE」の項にある 12-7 ページの segment_attributes_clause::= を参照）

***alter_overflow_clause*::=**



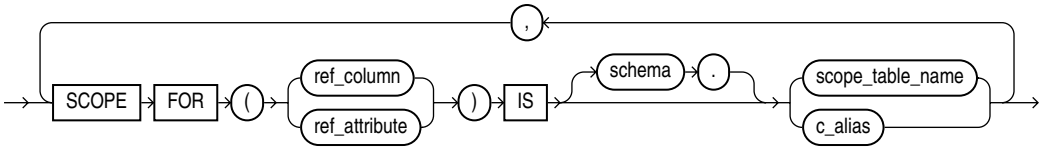
（11-6 ページの allocate_extent_clause::=、11-7 ページの shrink_clause::=、11-7 ページの deallocate_unused_clause::= を参照）

***add_overflow_clause*::=**

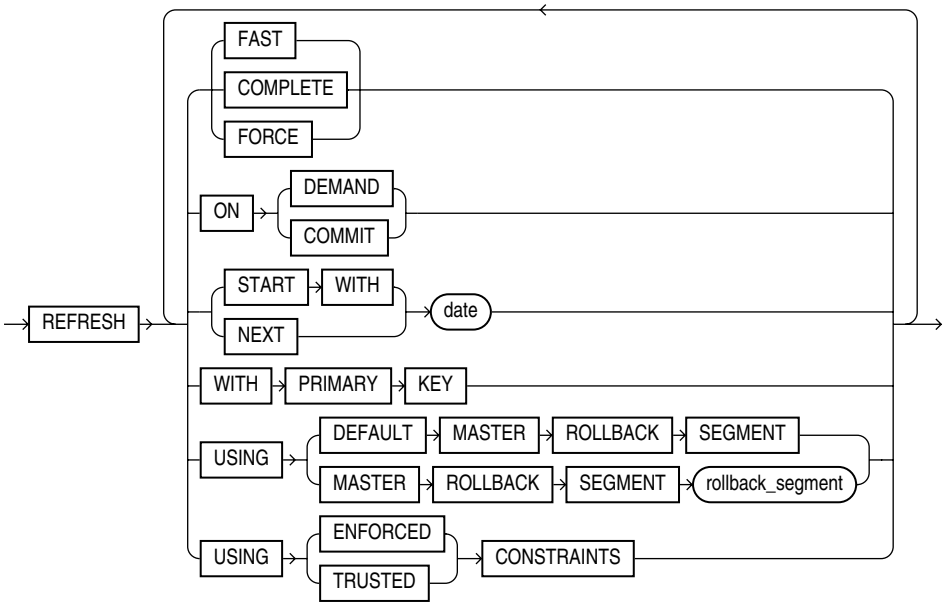


（「ALTER TABLE」の項にある 12-7 ページの segment_attributes_clause::= を参照）

***scoped_table_ref_constraint*::=**



alter_mv_refresh::=



セマンティクス

schema

マテリアライズド・ビューが含まれているスキーマを指定します。*schema* を指定しない場合、マテリアライズド・ビューは自分のスキーマ内にあるとみなされます。

materialized_view

変更するマテリアライズド・ビューの名前を指定します。

physical_attributes_clause

PCTFREE、PCTUSED、INITRANS パラメータの値（USING INDEX 句で使用する場合は、INITRANS パラメータ値のみ）、およびマテリアライズド・ビューの記憶特性を指定します。PCTFREE、PCTUSED および INITRANS パラメータの詳細は、12-2 ページの「ALTER TABLE」を参照してください。記憶特性の詳細は、8-41 ページの「storage_clause」を参照してください。

table_compression

table_compression 句を使用すると、ディスクおよびメモリーの使用量を削減するために、データ・セグメントを圧縮するかどうかを指定できます。この句のセマンティクスの詳細は、16-31 ページの「CREATE TABLE」の table_compression 句を参照してください。

LOB_storage_clause

LOB_storage_clause を使用すると、新しい LOB の記憶特性を指定できます。マテリアライズド・ビューの LOB 記憶域は、表の場合と同様に動作します。LOB 記憶域パラメータの詳細は、16-37 ページの「CREATE TABLE」の「LOB_storage_clause」を参照してください。

modify_LOB_storage_clause

modify_LOB_storage_clause を使用すると、LOB 属性 *LOB_item* の物理属性または LOB オブジェクト属性を変更できます。マテリアライズド・ビューの LOB 記憶域の変更は、表の場合と同様に動作します。

> **参照：** 変更可能な LOB 記憶域パラメータの詳細は、12-49 ページの「ALTER TABLE」の「modify_LOB_storage_clause」を参照してください。

alter_table_partitioning

マテリアライズド・ビューの alter_table_partitioning の構文および一般的な機能は、パーティション表と同じです。12-52 ページの「ALTER TABLE」の「alter_table_partitioning」を参照してください。

マテリアライズド・ビューのパーティション変更の制限事項： *partitioning_clauses* では、*LOB_storage_clause* または *modify_LOB_storage_clause* を指定できません。

> **注意：** マテリアライズド・ビューの内容をマスター表の内容と同期させて保持するには、表パーティションを削除または切り捨てた後、表に依存しているすべてのマテリアライズド・ビューを手動で完全リフレッシュすることをお薦めします。

MODIFY PARTITION UNUSABLE LOCAL INDEXES この句を使用すると、*partition* に関連付けられたすべてのローカル索引パーティションに、UNUSABLE のマークが付きます。

MODIFY PARTITION REBUILD UNUSABLE LOCAL INDEXES この句を使用すると、*partition* に関連付けられた、使用禁止のローカル索引パーティションを再構築できます。

parallel_clause

parallel_clause を使用すると、マテリアライズド・ビューのデフォルトの並列度を変更できます。

この句の詳細は、16-54 ページの「CREATE TABLE」の「parallel_clause」を参照してください。

logging_clause

この句を使用すると、マテリアライズド・ビューのロギング特性を指定または変更できます。この句の詳細は、8-34 ページの「logging_clause」を参照してください。

allocate_extent_clause

allocate_extent_clause を使用すると、マテリアライズド・ビューの新しいエクステントを明示的に割り当てることができます。この句の詳細は、8-2 ページの「allocate_extent_clause」を参照してください。

shrink_clause

この句を使用すると、マテリアライズド・ビューのセグメントを縮小化できます。この句の詳細は、12-35 ページの「CREATE TABLE」の「shrink_clause」を参照してください。

CACHE | NOCACHE

アクセス頻度の高いデータについて、CACHE は、全表スキャンの実行時にこの表に対して取り出された各ブロックを、バッファ・キャッシュの LRU リストの最高使用頻度側に入れることを指定します。この属性は、小規模な参照表で有効です。NOCACHE は、ブロックを LRU リストの最低使用頻度側に入れることを指定します。この句の詳細は、16-53 ページの「CREATE TABLE」の「CACHE | NOCACHE | CACHE READS」を参照してください。

alter_iot_clauses

alter_iot_clauses を使用すると、索引構成マテリアライズド・ビューの特性を変更できます。*alter_iot_clauses* コンポーネントのキーワードおよびパラメータは、ALTER TABLE と同じです。また、次の制限事項があります。

索引構成マテリアライズド・ビューの変更の制限事項： *index_org_table_clause* の *mapping_table_clauses* または *key_compression* 句は指定できません。

> **参照：** 索引構成マテリアライズド・ビューの作成については、15-14 ページの「CREATE MATERIALIZED VIEW」の「index_org_table_clause」を参照してください。

USING INDEX 句

この句を使用すると、マテリアライズド・ビューのデータをメンテナンスするために使用される索引の INITRANS パラメータおよび STORAGE パラメータの値を変更できます。

USING INDEX 句の制限事項： この句では、PCTUSED または PCTFREE パラメータは指定できません。

MODIFY *scoped_table_ref_constraint*

MODIFY *scoped_table_ref_constraint* 句を使用すると、新しい表または新しい列の別名に REF 列または属性の有効範囲を再指定できます。

REF 列の有効範囲の再指定の制限事項： ALTER MATERIALIZED VIEW 文では、1 つの REF 列または属性の有効範囲のみを再指定することができます。この句は、この文以外では使用できません。

alter_mv_refresh

alter_mv_refresh 句を使用すると、自動リフレッシュの方法、モードおよび日時のデフォルト値を変更できます。マテリアライズド・ビューのマスター表の内容が変更された場合、マテリアライズド・ビューのデータを更新し、現在マスター表にあるデータを正確に反映させる必要があります。この句によって、自動的にマテリアライズド・ビューをリフレッシュする日時をスケジューリングし、リフレッシュの方法およびモードを指定できます。

注意： この句では、デフォルトのリフレッシュ・オプションのみを設定します。リフレッシュを実際に実装する手順は、『Oracle Database アドバンスト・レプリケーション』および『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

FAST 句

FAST を指定すると、増分リフレッシュ方法を指定できます。これはマスター表に対して行った変更に従ってリフレッシュを行います。この変更は、マスター表に関連付けられたマテリアライズド・ビュー・ログ（従来型 DML 変更の場合）またはダイレクト・ローダー・ログ（ダイレクト・パス・インサート操作の場合）に格納されます。

従来型 DML の変更の場合も、ダイレクト・パス・インサート操作の場合も、他の条件によって、高速リフレッシュへのマテリアライズド・ビューの適応性が制限されることがあります。

参照 :

- レプリケーション環境における高速リフレッシュの制限については、『Oracle Database アドバンスト・レプリケーション』を参照してください。
- データ・ウェアハウス環境における高速リフレッシュの制限については、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。
- 「自動リフレッシュの例 :」（11-14 ページ）

FAST リフレッシュの制限事項 : FAST リフレッシュには、次の制限事項があります。

- 作成時に FAST リフレッシュを指定した場合、作成するマテリアライズド・ビューは高速リフレッシュに適応することが検証されています。ALTER MATERIALIZED VIEW 文でリフレッシュ方法を FAST に変更した場合、これは検証されていません。マテリアライズド・ビューが高速リフレッシュに適応しない場合、このビューをリフレッシュしようとするとエラーが戻されます。
- 定義する問合せに分析ファンクションが含まれている場合、マテリアライズド・ビューは高速リフレッシュに適応しません。
- 暗号化されている列がある場合は、マテリアライズド・ビューを高速リフレッシュできません。

参照 : 「分析ファンクション」（5-10 ページ）

COMPLETE 句

COMPLETE を指定すると、完全リフレッシュ方法を指定できます。これは、マテリアライズド・ビューを定義する問合せを実行することによって実装されます。完全リフレッシュを指定すると、高速リフレッシュが実行可能であっても、完全リフレッシュが実行されます。

参照 : 「完全リフレッシュの例 :」（11-15 ページ）

FORCE 句

FORCE を指定すると、リフレッシュ時に、高速リフレッシュが可能な場合は高速リフレッシュを実行し、そうでない場合は完全リフレッシュを実行できます。

ON COMMIT 句

ON COMMIT を指定すると、マテリアライズド・ビューのマスター表に対するトランザクションをコミットするときに必ず高速リフレッシュが実行されます。

ON COMMIT の制限事項 : この句は、マテリアライズド結合ビューおよびマテリアライズド集計ビューにのみサポートされます。

参照 : 『Oracle Database アドバンスト・レプリケーション』および『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

ON DEMAND 句

ON DEMAND を指定すると、マテリアライズド・ビューは、3 つの DBMS_MVIEW リフレッシュ・プロシージャのいずれかのコールによる要求でリフレッシュされます。ON COMMIT および ON DEMAND のどちらも指定しなかった場合、ON DEMAND がデフォルトになります。

ON COMMIT または ON DEMAND を指定した場合、START WITH または NEXT を同時に指定できません。

参照 :

- これらのプロシージャの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。
- REFRESH ON DEMAND を指定することによって作成できるマテリアライズド・ビューのタイプについては、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

START WITH 句

START WITH *date* を指定すると、最初の自動リフレッシュ時間を表す日付を指定できます。

NEXT 句

NEXT を指定すると、自動リフレッシュの間隔を計算するための日付式を指定できます。

START WITH 値および NEXT 値は、将来の時刻に評価される値です。START WITH 値を省略した場合、Oracle Database はマテリアライズド・ビューの作成時刻に対して NEXT 式を評価することによって、最初の自動リフレッシュ時刻を判断します。START WITH 値を指定し、NEXT 値を指定しない場合、Oracle Database は 1 回のみマテリアライズド・ビューをリフレッシュします。START WITH 値および NEXT 値のどちらも指定しない場合、または *alter_mv_refresh* を指定しない場合、Oracle Database はマテリアライズド・ビューを自動リフレッシュしません。

WITH PRIMARY KEY 句

WITH PRIMARY KEY を指定すると、ROWID マテリアライズド・ビューを主キー・マテリアライズド・ビューに変更できます。主キー・マテリアライズド・ビューを使用すると、高速リフレッシュを継続できるマテリアライズド・ビューの機能に影響せずに、マテリアライズド・ビュー・マスター表を再編成できます。

この句を指定するには、マスター表に、使用可能な主キー制約が定義され、この制約に基づき、主キー情報を記録するマテリアライズド・ビュー・ログが定義されている必要があります。

参照 :

- 主キー・マテリアライズド・ビューの詳細は、『Oracle Database アドバンスト・レプリケーション』を参照してください。
- 「主キー・マテリアライズド・ビューの例 :」(11-15 ページ)

USING ROLLBACK SEGMENT 句

自動 UNDO モードでは、ロールバック・セグメントではなく UNDO 表領域が使用されるため、データベースが自動 UNDO モードの場合、この句は無効です。自動 UNDO モードを使用することをお薦めします。この句は、ロールバック・セグメントが使用される以前のバージョンの Oracle Database が含まれるレプリケーション環境との下位互換性のためにサポートされています。

この句の詳細は、15-18 ページの「CREATE MATERIALIZED VIEW」の「USING ROLLBACK SEGMENT 句」を参照してください。

USING ...CONSTRAINTS 句

この句のセマンティクスは、CREATE MATERIALIZED VIEW および ALTER MATERIALIZED VIEW 文で同じです。この句の詳細は、15-19 ページの「CREATE MATERIALIZED VIEW」の「USING ...CONSTRAINTS 句」を参照してください。

QUERY REWRITE 句

この句を使用すると、マテリアライズド・ビューをクエリー・リライトで使用できるかどうかを指定できます。

ENABLE 句

ENABLE を指定すると、クエリー・リライトでマテリアライズド・ビューを使用可能にできます。

参照：「クエリー・リライトを使用可能にする例：」（11-15 ページ）

マテリアライズド・ビューを使用可能にする場合の制限事項： マテリアライズド・ビューを使用可能にする処理には、次の制限事項があります。

- マテリアライズド・ビューが無効または使用禁止の場合、ENABLE モードでもクエリー・リライトに適応しません。
- マテリアライズド・ビューの全体または一部がビューから作成されている場合、クエリー・リライトを使用可能にできません。
- マテリアライズド・ビューのすべてのユーザー定義ファンクションが DETERMINISTIC である場合のみ、クエリー・リライトを使用可能にできます。

  参照：「CREATE FUNCTION」（14-48 ページ）

- 文内の式が反復可能な場合のみ、クエリー・リライトを使用可能にできます。たとえば、CURRENT_TIME または USER を含めることはできません。

  参照： クエリー・リライトの詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

DISABLE 句

DISABLE を指定すると、クエリー・リライトでマテリアライズド・ビューを使用禁止にできます。マテリアライズド・ビューが無効な場合、使用禁止であるかどうかにかかわらず、クエリー・リライトの使用には適応しません。ただし、使用禁止にされたマテリアライズド・ビューをリフレッシュすることはできます。

COMPILE

COMPILE を指定すると、マテリアライズド・ビューを明示的に再検証できます。マテリアライズド・ビューが依存するオブジェクトを削除または変更した場合、マテリアライズド・ビューはアクセス可能のままですが、クエリー・リライトに対しては無効です。再度、明示的にマテリアライズド・ビューの妥当性チェックを行い、クエリー・リライトの使用に適応させるには、この句を使用します。

マテリアライズド・ビューの再妥当性チェックに失敗すると、リフレッシュできなくなるか、またはクエリー・リライトに使用できなくなります。

参照：「マテリアライズド・ビューのコンパイル例：」（11-15 ページ）

CONSIDER FRESH

この句を使用すると、マスター表が変更された後のマテリアライズド・ビューの失効状態を管理することができます。CONSIDER FRESH は、マテリアライズド・ビューが最新であり、TRUSTED または STALE_TOLERATED モードでのクエリー・リライトに適応するとみなされるように指定します。Oracle Database は、マテリアライズド・ビューが最新であるかどうかを保証できないため、ENFORCED モードでのクエリー・リライトはサポートしません。また、この句はマテリアライズド・ビューの失効状態を UNKNOWN に設定します。失効状態は、ALL_MVIEWS、DBA_MVIEWS および USER_MVIEWS の各データ・ディクショナリ・ビューの STALENESS 列に表示されます。

いずれかのマスター表の内容が変更された場合、マテリアライズド・ビューは失効します。この句は、Oracle Database に対して、マテリアライズド・ビューが最新で、変更されていないものと仮定するように指示します。そのため、リフレッシュが保留されているこれらの表に対する実際の更新内容は、マテリアライズド・ビューから削除されます。

参照：

- クエリー・リライトの詳細、およびマスター表へのパーティション・メンテナンス操作の影響については、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。
- 「CONSIDER FRESH の例 :」（11-15 ページ）

例

自動リフレッシュの例： 次の文は、マテリアライズド・ビュー sales_by_month_by_state（15-22 ページの「マテリアライズド集計ビューの作成例 :」で作成）のリフレッシュ方法のデフォルトを FAST に変更します。

```
ALTER MATERIALIZED VIEW sales_by_month_by_state
   REFRESH FAST;
```

これ以降のマテリアライズド・ビューの自動リフレッシュは、高速リフレッシュになります。これは、単純なマテリアライズド・ビューであり、そのマスター表には、マテリアライズド・ビューの作成前または最後のリフレッシュ前に作成されたマテリアライズド・ビュー・ログがあります。

REFRESH 句に START WITH または NEXT の値が指定されていないため、マテリアライズド・ビュー sales_by_month_by_state を作成したとき、または最後に変更したときに REFRESH 句で設定されたリフレッシュ間隔がそのまま使用されます。

次の文は、マテリアライズド・ビュー sales_by_month_by_state の新しい自動リフレッシュ間隔を設定します。

```
ALTER MATERIALIZED VIEW sales_by_month_by_state
   REFRESH NEXT SYSDATE+7;
```

REFRESH に START WITH の値が指定されていないため、マテリアライズド・ビュー sales_by_month_by_state が作成されたとき、または最後に変更されたときに指定された START WITH と NEXT の値によって設定された日時に次の自動リフレッシュが行われます。

このマテリアライズド・ビューは、次に自動リフレッシュが行われる際に、リフレッシュされます。次に自動リフレッシュが行われる日時は、NEXT に設定した式 SYSDATE+7 が計算されて決まります。その後は、週に 1 回リフレッシュが自動的に行われます。REFRESH 句にリフレッシュ方法が明示的に指定されていないため、CREATE MATERIALIZED VIEW 文または直前の ALTER MATERIALIZED VIEW 文の REFRESH 句で指定されたリフレッシュ方法が引き続き使用されます。

CONSIDER FRESH の例： 次の文は、Oracle Database がマテリアライズド・ビュー sales_by_month_by_state を最新であるとみなすように指定します。この文を使用すると、sales_by_month_by_state のマスター表に対してパーティション・メンテナンス操作を実行した後でも、sales_by_month_by_state に対して TRUSTED モードでのクエリー・リライトが可能です。

```
ALTER MATERIALIZED VIEW sales_by_month_by_state CONSIDER FRESH;
```

> **参照：** この ALTER MATERIALIZED VIEW の例を必要とするパーティション・メンテナンスの例は、12-75 ページの「表のパーティションの分割例 :」を参照してください。

完全リフレッシュの例： 次の文は、マテリアライズド・ビュー emp_data（15-23 ページの「マテリアライズド・ビューの定期的リフレッシュ例 :」で作成）の新しいリフレッシュ方法、NEXT で示す新しいリフレッシュ日時および新しい自動リフレッシュ間隔を指定します。

```
ALTER MATERIALIZED VIEW emp_data
   REFRESH COMPLETE
   START WITH TRUNC(SYSDATE+1) + 9/24
   NEXT SYSDATE+7;
```

START WITH 句に指定した値によって、このマテリアライズド・ビューの次の自動リフレッシュは翌日の午前 9 時に発生するように設定されます。この日時にマテリアライズド・ビューの完全リフレッシュが実行され、NEXT に設定した式が計算されます。その後は、このマテリアライズド・ビューは毎週リフレッシュされます。

クエリー・リライトを使用可能にする例： 次の文は、マテリアライズド・ビュー emp_data のクエリー・リライトを使用可能にし、暗黙的に再妥当性チェックを行います。

```
ALTER MATERIALIZED VIEW emp_data
   ENABLE QUERY REWRITE;
```

主キー・マテリアライズド・ビューの例： 次の文は、ROWID マテリアライズド・ビュー order_data（15-23 ページの「ROWID マテリアライズド・ビューの作成例 :」で作成）を主キー・マテリアライズド・ビューに変更します。この例では、order_data に主キーを持つマテリアライズド・ビュー・ログを定義していることが必要です。

```
ALTER MATERIALIZED VIEW order_data
   REFRESH WITH PRIMARY KEY;
```

マテリアライズド・ビューのコンパイル例： 次の文は、マテリアライズド・ビュー store_mv を再検証します。

```
ALTER MATERIALIZED VIEW order_data COMPILE;
```

ALTER MATERIALIZED VIEW LOG

用途

マテリアライズド・ビュー・ログとは、マテリアライズド・ビューのマスター表に関連付けられる表です。ALTER MATERIALIZED VIEW LOG 文を使用すると、既存のマテリアライズド・ビュー・ログの記憶特性またはタイプを変更できます。

注意： 下位互換性を保つために、MATERIALIZED VIEW のかわりにキーワード SNAPSHOT もサポートされています。

参照：

- マテリアライズド・ビュー・ログの作成については、15-26 ページの「CREATE MATERIALIZED VIEW LOG」を参照してください。
- マテリアライズド・ビューのリフレッシュ方法など、マテリアライズド・ビューの詳細は、11-2 ページの「ALTER MATERIALIZED VIEW」を参照してください。
- マテリアライズド・ビューの様々なタイプの詳細は、15-4 ページの「CREATE MATERIALIZED VIEW」を参照してください。

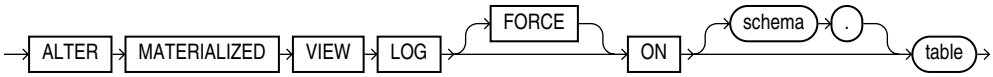
前提条件

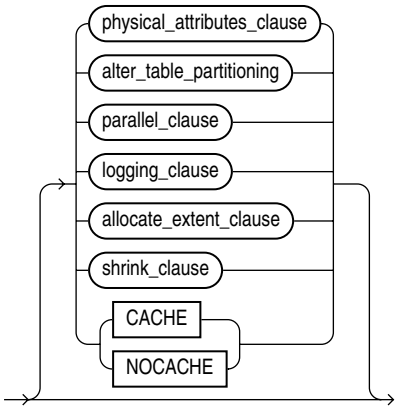
マスター表の所有者であるか、またはマスター表に対する SELECT 権限およびマテリアライズド・ビュー・ログに対する ALTER 権限が必要です。

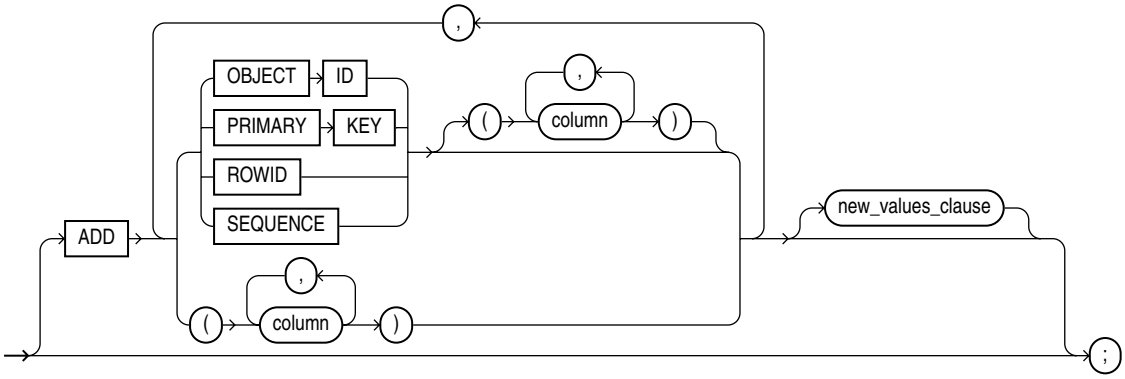
参照： ALTER MATERIALIZED VIEW LOG の前提条件の詳細は、『Oracle Database アドバンスト・レプリケーション』を参照してください。

構文

***alter_materialized_view_log*::=**

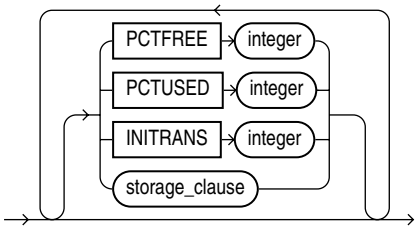






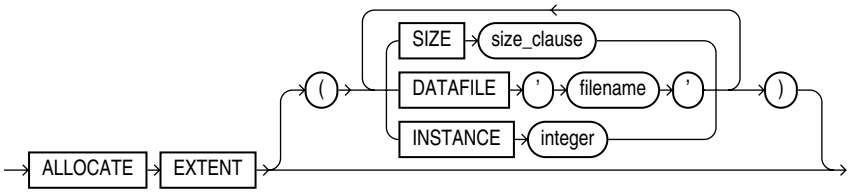
（11-17 ページの physical_attributes_clause::=、12-18 ページの alter_table_partitioning::=（「ALTER TABLE」の項）、11-18 ページの parallel_clause::=、8-34 ページの logging_clause::=、11-18 ページの allocate_extent_clause::= および 11-18 ページの new_values_clause::= を参照）

physical_attributes_clause::=



（8-44 ページの storage_clause::= を参照）

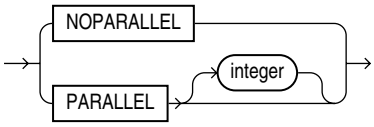
allocate_extent_clause::=



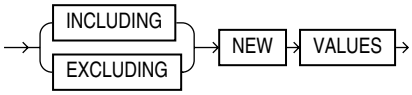
(8-42 ページの size_clause::= を参照)

shrink_clause::=

parallel_clause::=



***new_values_clause*::=**



セマンティクス

FORCE

FORCE を指定すると、ADD 句で指定したいずれかの項目がすでにマテリアライズド・ビュー・ログに指定されている場合、エラーは戻されませんが、既存の要素は無視され、マテリアライズド・ビュー・ログに存在しないすべての項目が追加されます。同様に、INCLUDING NEW VALUES を指定すると、この属性がすでにマテリアライズド・ビュー・ログに指定されている場合、冗長は無視され、エラーも戻されません。

schema

マスター表が定義されているスキーマを指定します。*schema* を指定しない場合、マテリアライズド・ビュー・ログは自分のスキーマ内にあるとみなされます。

table

変更するマテリアライズド・ビュー・ログに関連付けられたマスター表の名前を指定します。

physical_attributes_clause

physical_attributes_clause を使用すると、PCTFREE、PCTUSED および INITRANS の各パラメータの値、マテリアライズド・ビュー・ログ、パーティションおよびオーバーフロー・データ・セグメントの記憶特性、またはパーティション・マテリアライズド・ビュー・ログのデフォルト特性を変更できます。

マテリアライズド・ビュー・ログの物理属性の制限事項： マテリアライズド・ビュー・ログがローカル管理表領域に存在する場合は、*storage_clause* を使用して、エクステント・パラメータを変更することはできません。このパラメータについては、16-6 ページの「CREATE TABLE」を参照してください。

alter_table_partitioning

alter_table_partitioning の構文および一般的な機能は、ALTER TABLE 文の場合と同じです。12-52 ページの「ALTER TABLE」の「alter_table_partitioning」を参照してください。

マテリアライズド・ビュー・ログのパーティション変更の制限事項： マテリアライズド・ビュー・ログのパーティションの変更には、次の制限事項があります。

- マテリアライズド・ビュー・ログのパーティションを変更する場合、*LOB_storage_clause* または *modify_LOB_storage_clause* は使用できません。
- マテリアライズド・ビュー・ログのパーティションを削除、切捨てまたは交換しようとすると、Oracle Database はエラーを戻します。

parallel_clause

parallel_clause を使用すると、マテリアライズド・ビュー・ログへのパラレル操作がサポートされているかどうかを指定できます。

この句の詳細は、16-54 ページの「CREATE TABLE」の「parallel_clause」を参照してください。

logging_clause

マテリアライズド・ビュー・ログに対するロギング属性を指定します。この句の詳細は、8-34 ページの「logging_clause」を参照してください。

allocate_extent_clause

allocate_extent_clause を使用すると、マテリアライズド・ビュー・ログの新しいエクステントを明示的に割り当てることができます。この句の詳細は、8-2 ページの「allocate_extent_clause」を参照してください。

shrink_clause

この句を使用すると、マテリアライズド・ビュー・ログのセグメントを縮小化できます。この句の詳細は、12-35 ページの「CREATE TABLE」の「shrink_clause」を参照してください。

CACHE | NOCACHE 句

アクセス頻度が高いデータについて、CACHE は、全表スキャンの実行時にこのログ用に取り出された各ブロックを、バッファ・キャッシュ内の LRU リストの最高使用頻度側に入れることを指定します。この属性は、小規模な参照表で有効です。NOCACHE は、ブロックを LRU リストの最低使用頻度側に入れることを指定します。この句の詳細は、16-53 ページの「CREATE TABLE」の「CACHE | NOCACHE | CACHE READS」を参照してください。

ADD 句

ADD 句を使用すると、マテリアライズド・ビュー・マスター表内の行が変更される際に、主キー値、ROWID 値、オブジェクト ID 値または順序も記録できるようにマテリアライズド・ビュー・ログを拡張できます。また、この句は、新しく列を記録するためにも使用できます。

これらの情報の記録を停止する場合は、マテリアライズド・ビュー・ログを削除してから、再作成する必要があります。マテリアライズド・ビュー・ログを削除した後再作成した場合、マスター表に依存するすべての既存マテリアライズド・ビューが、次回のリフレッシュ時に強制的に完全リフレッシュされます。

マテリアライズド・ビュー・ログの拡張の制限事項： 各マテリアライズド・ビュー・ログに指定できるのは、PRIMARY KEY、ROWID、OBJECT ID、SEQUENCE および列リストの列を 1 つずつです。この ALTER 文に PRIMARY KEY、ROWID、OBJECT ID、SEQUENCE および列リストを指定できるのはそれぞれ 1 回のみです。また、FORCE オプションを指定しないかぎり、これらの値のいずれかが作成時に（暗黙的または明示的に）指定された場合、この ALTER 文にはそれらの値を指定できません。

OBJECT ID OBJECT ID を指定すると、更新されるすべての行の適切なオブジェクト識別子をマテリアライズド・ビュー・ログに記録できます。

OBJECT ID 句の制限事項 : OBJECT ID は、オブジェクト表のログ用のみに指定でき、記憶表用には指定できません。

PRIMARY KEY PRIMARY KEY を指定すると、更新されるすべての行の主キーをマテリアライズド・ビュー・ログに記録できます。

ROWID ROWID を指定すると、更新されるすべての行の ROWID 値をマテリアライズド・ビュー・ログに記録できます。

SEQUENCE SEQUENCE を指定すると、追加の順序情報を提供する順序値をマテリアライズド・ビュー・ログに記録できます。

column 更新されるすべての行に対して、マテリアライズド・ビュー・ログに記録する値を持つ新しい列を指定します。通常、フィルタ列（副問合せマテリアライズド・ビューが参照する主キー以外の列）および結合列（副問合せの WHERE 句で結合を定義する主キー以外の列）を指定します。

参照 :

- マテリアライズド・ビュー・ログに値を明示的および暗黙的に含める方法については、15-4 ページの「CREATE MATERIALIZED VIEW」を参照してください。
- フィルタ列および結合列については、『Oracle Database アドバンスト・レプリケーション』を参照してください。
- 「ROWID マテリアライズド・ビュー・ログの例 :」（11-20 ページ）

NEW VALUES 句

NEW VALUES 句を使用すると、更新 DML 操作で、古い値と新しい値の両方をマテリアライズド・ビュー・ログに保存するかどうかを指定できます。ALTER MATERIALIZED VIEW LOG 文で追加した列のみでなく、ログのすべての列にこの句で設定した値を適用します。

INCLUDING INCLUDING を指定すると、古い値と新しい値の両方を保存できます。このログが単一表マテリアライズド集計ビューの表用で、マテリアライズド・ビューに高速リフレッシュを実行する場合、INCLUDING を指定してください。

EXCLUDING EXCLUDING を指定すると、ログに新しい値が記録されなくなります。この句を使用すると、新しい値の記録によるオーバーヘッドを回避できます。

マテリアライズド・ビューのリフレッシュ・モードを FAST 以外のモードに変更した場合を除き、高速リフレッシュが可能な単一表マテリアライズド集計ビューが定義されている場合は、EXCLUDING NEW VALUES を使用しないでください。

参照 : 「マテリアライズド・ビュー・ログ EXCLUDING NEW VALUES の例 :」（11-21 ページ）

例

ROWID マテリアライズド・ビュー・ログの例 : 次の文は、ROWID 情報も記録するように既存の主キー・マテリアライズド・ビュー・ログを変更します。

```
ALTER MATERIALIZED VIEW LOG ON order_items ADD ROWID;
```

マテリアライズド・ビュー・ログ EXCLUDING NEW VALUES の例 : 次の文は、フィルタ列を追加し、新しい値を除外することで、hr.employees のマテリアライズド・ビュー・ログを変更します。このログを使用するマテリアライズド集計ビューは、これ以降高速リフレッシュされません。ただし、高速リフレッシュが不要になる場合は、この処理によって新しい値の記録によるオーバーヘッドを回避できます。

```
ALTER MATERIALIZED VIEW LOG ON employees
   ADD (commission_pct)
   EXCLUDING NEW VALUES;
```

ALTER OPERATOR

用途

ALTER OPERATOR 文を使用すると、既存の演算子に対するバインドを追加または削除したり、既存の演算子をコンパイルすることができます。

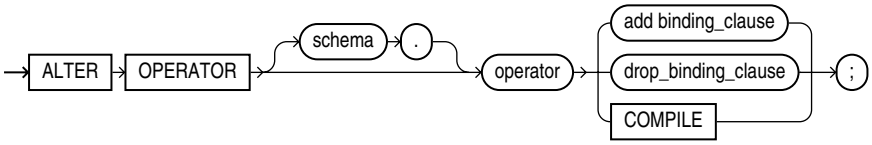
参照： 「CREATE OPERATOR」（15-32 ページ）

前提条件

事前に CREATE OPERATOR 文によって演算子が作成されている必要があります。演算子が自分のスキーマ内にあるか、または ALTER ANY OPERATOR システム権限が必要です。ALTER OPERATOR 文で参照される演算子およびファンクションに対する EXECUTE オブジェクト権限が必要です。

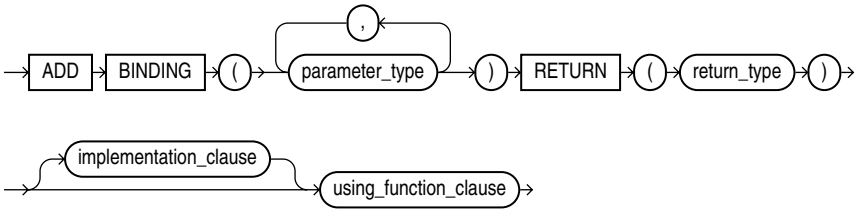
構文

***alter_operator*::=**



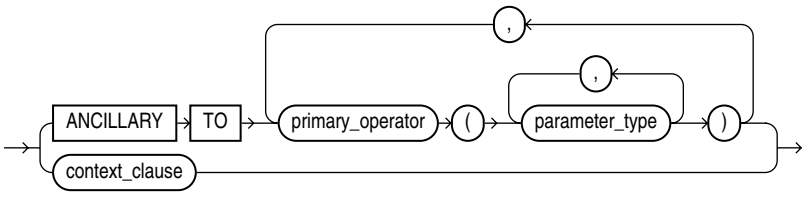
（11-22 ページの add_binding_clause::=、11-23 ページの drop_binding_clause::= を参照）

add_binding_clause::=



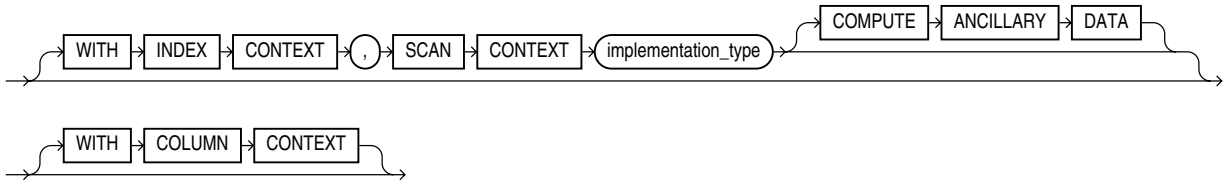
（11-22 ページの implementation_clause::=、11-23 ページの using_function_clause::= を参照）

implementation_clause::=

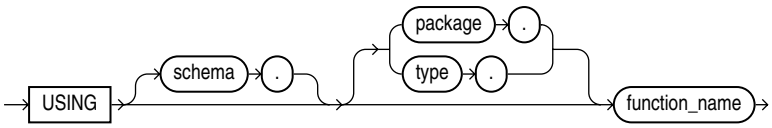


（11-23 ページの context_clause::= を参照）

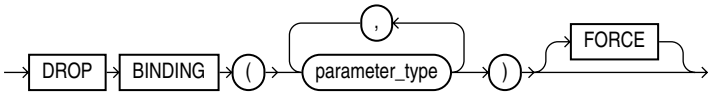
context_clause::=



using_function_clause::=



drop_binding_clause::=



セマンティクス

schema

演算子が含まれているスキーマを指定します。この句を指定しない場合、その演算子は自分のスキーマにあるとみなされます。

operator

変更する演算子の名前を指定します。

add_binding_clause

この句を使用すると、演算子バインドを追加し、パラメータ・データ型および戻り型を指定できます。この演算子の既存のバインドと異なる署名を使用する必要があります。

演算子のバインドが索引タイプに関連付けられており、演算子に別のバインドを追加する場合、Oracle Database は新しいバインドと索引タイプの関連付けを自動的に行いません。関連付けを行うには、明示的な ALTER INDEXTYPE ...ADD OPERATOR 文を発行する必要があります。

implementation_clause

この句のセマンティクスは、CREATE OPERATOR および ALTER OPERATOR 文で同じです。この句の詳細は、15-33 ページの「CREATE OPERATOR」の「implementation_clause」を参照してください。

context_clause

この句のセマンティクスは、CREATE OPERATOR および ALTER OPERATOR 文で同じです。この句の詳細は、15-34 ページの「CREATE OPERATOR」の「context_clause」を参照してください。

using_function_clause

この句のセマンティクスは、CREATE OPERATOR および ALTER OPERATOR 文で同じです。この句の詳細は、15-34 ページの「CREATE OPERATOR」の「using_function_clause」を参照してください。

drop_binding_clause

この句を使用すると、演算子から削除するバインドのパラメータ・データ型のリストを指定できます。バインドに索引タイプや補助演算子バインドなどの依存オブジェクトが含まれる場合、FORCE を指定する必要があります。FORCE を指定すると、そのバインドに依存するすべてのオブジェクトに INVALID のマークが付きます。依存オブジェクトは、DDL 文、DML 文または問合せで次に参照された際に再検証されます。

この句を使用して、この演算子に関連付けられた唯一のバインドを削除することはできません。この演算子に関連付けられた唯一のバインドを削除するには、DROP OPERATOR 文を使用する必要があります。詳細は、17-53 ページの「DROP OPERATOR」を参照してください。

COMPILE

COMPILE を指定すると、演算子を再コンパイルできます。

例

ユーザー定義演算子のコンパイル例： 次の例は、演算子 eq_op（15-34 ページの「ユーザー定義演算子の作成例：」で作成）をコンパイルします。

```
ALTER OPERATOR eq_op COMPILE;
```

ALTER OUTLINE

用途

注意： 新規のアプリケーションには SQL 計画管理を使用することをお薦めします。SQL 計画管理では、ストアド・アウトラインよりも非常に安定した SQL パフォーマンスを実現する SQL 計画ベースラインが作成されます。

DBMS_SPM パッケージの LOAD_PLANS_FROM_CURSOR_CACHE プロシージャまたは LOAD_PLANS_FROM_SQLSET プロシージャを使用して、既存のストアド・アウトラインを SQL 計画ベースラインに移行できます。移行が完了したら、ストアド・アウトラインを無効にするか、または削除する必要があります。

参照： SQL 計画管理の詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。DBMS_SPM パッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

ALTER OUTLINE 文を使用すると、ストアド・アウトラインの名前を変更したり、ストアド・アウトラインを異なるカテゴリに再度割り当てることができます。また、アウトラインの SQL 文をコンパイルし、古いアウトライン・データを現行の条件で作成したアウトラインと置き換えて、ストアド・アウトラインを再生成できます。

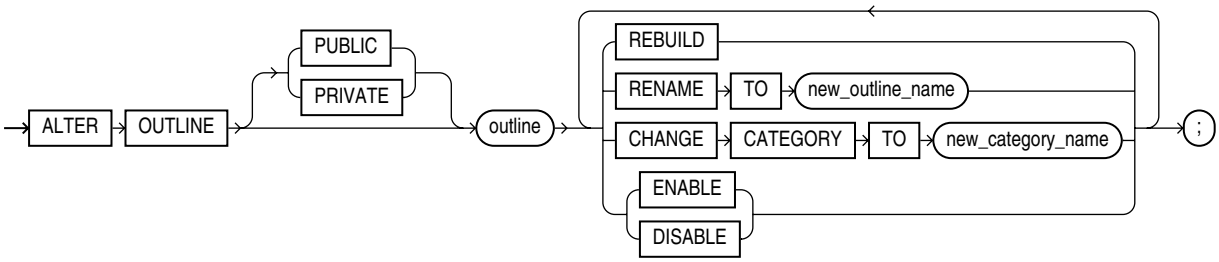
参照： アウトラインの詳細は、15-35 ページの「CREATE OUTLINE」および『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。

前提条件

アウトラインを変更する場合は、ALTER ANY OUTLINE システム権限が必要です。

構文

***alter_outline*::=**



セマンティクス

PUBLIC | PRIVATE

PUBLIC を指定すると、アウトラインのパブリック・バージョンを変更できます。これはデフォルトです。

PRIVATE を指定すると、現行のセッションに対してプライベートで、現行の解析スキーマにデータが格納されているアウトラインを変更できます。

outline

変更するアウトラインの名前を指定します。

REBUILD

REBUILD を指定すると、現行の条件で、*outline* の実行計画が再生成されます。

参照： 「アウトラインの再構築例 :」（11-26 ページ）

RENAME TO 句

RENAME TO 句を使用すると、*outline* の値と置き換えるアウトライン名を指定できます。

CHANGE CATEGORY TO 句

CHANGE CATEGORY TO 句を使用すると、*outline* の移動先となるカテゴリ名を指定できます。

ENABLE | DISABLE

この句を使用すると、このアウトラインを選択的に使用可能または使用禁止にできます。デフォルトでは、アウトラインは使用可能になっています。DISABLE キーワードを使用すると、他のアウトラインの使用に影響を与えずに、1 つのアウトラインを使用禁止にできます。

例

アウトラインの再構築例： 次の文は、アウトラインのテキストをコンパイルし、古いアウトライン・データを現行の条件で作成したアウトラインと置き換えて、salaries というストアド・アウトラインを再生成します。

```
ALTER OUTLINE salaries REBUILD;
```

ALTER PACKAGE

用途

パッケージは PL/SQL を使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

ALTER PACKAGE 文を使用すると、パッケージ仕様部またはパッケージ本体（あるいはその両方）を明示的に再コンパイルできます。明示的に再コンパイルすることによって、実行時に暗黙的に再コンパイルする必要がなくなり、また、実行時のコンパイル・エラーとパフォーマンス上のオーバーヘッドもなくなります。

パッケージ中のすべてのオブジェクトは、1 つの単位として格納されているため、ALTER PACKAGE 文によって、すべてのパッケージ・オブジェクトがまとめて再コンパイルされます。ALTER PROCEDURE 文または ALTER FUNCTION 文を使用して、パッケージ中の一部のプロシージャまたはファンクションを再コンパイルすることはできません。

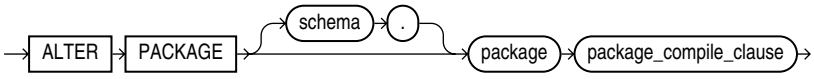
> **注意：** この文を使用して、既存のパッケージの宣言や定義を変更することはできません。パッケージを再宣言または再定義する場合は、15-39 ページの「CREATE PACKAGE」または「CREATE PACKAGE BODY」に OR REPLACE 句を指定します。

前提条件

パッケージを変更するには、パッケージが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY PROCEDURE システム権限が必要です。

構文

***alter_package*::=**



（*package_compile_clause*: この句の構文の詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。）

セマンティクス

schema

パッケージが含まれているスキーマを指定します。*schema* を指定しない場合、パッケージは自分のスキーマ内にあるとみなされます。

package

再コンパイルするパッケージの名前を指定します。

package_compile_clause

この句の構文とセマンティクスの詳細およびパッケージの作成とコンパイルの詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

ALTER PROCEDURE

用途

パッケージは PL/SQL を使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

ALTER PROCEDURE 文を使用すると、スタンドアロンのストアド・プロシージャを明示的に再コンパイルできます。明示的に再コンパイルすることによって、実行時に暗黙的に再コンパイルする必要がなくなり、また、実行時のコンパイル・エラーとパフォーマンス上のオーバーヘッドもなくなります。

パッケージの一部であるプロシージャを再コンパイルする場合、ALTER PACKAGE 文を使用して、そのパッケージ全体を再コンパイルします（11-27 ページの「ALTER PACKAGE」を参照）。

> **注意：** この文を使用して、既存のプロシージャの宣言または定義を変更することはできません。プロシージャを再宣言または再定義する場合は、OR REPLACE を指定して CREATE PROCEDURE 文を使用します（15-45 ページの「CREATE PROCEDURE」を参照）。

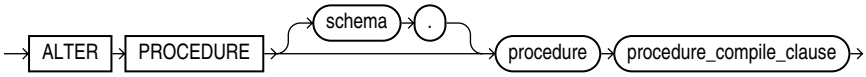
ALTER PROCEDURE 文は、ALTER FUNCTION 文と似ています。詳細は、10-63 ページの「ALTER FUNCTION」を参照してください。

前提条件

プロシージャは、自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY PROCEDURE システム権限が必要です。

構文

***alter_procedure*::=**



（*procedure_compile_clause*: この句の構文の詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。）

セマンティクス

schema

プロシージャが含まれているスキーマを指定します。*schema* を指定しない場合、プロシージャは自分のスキーマ内にあるとみなされます。

procedure

再コンパイルするプロシージャの名前を指定します。

procedure_compile_clause

この句の構文とセマンティクスの詳細およびプロシージャの作成とコンパイルの詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

ALTER PROFILE

用途

ALTER PROFILE 文を使用すると、プロファイルのリソース制限またはパスワード管理パラメータを追加、変更または削除できます。

ALTER PROFILE 文を使用すると、プロファイルに対して行った変更は、このプロファイルの現行のセッションのユーザーには影響しません。後続セッションのユーザーのみに影響します。

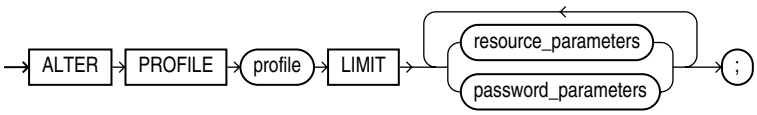
> **参照：** プロファイルの作成の詳細は、15-47 ページの「CREATE PROFILE」を参照してください。

前提条件

プロファイルのリソース制限を変更する場合は、ALTER PROFILE システム権限が必要です。パスワード制限および保護を変更する場合は、ALTER PROFILE および ALTER USER システム権限が必要です。

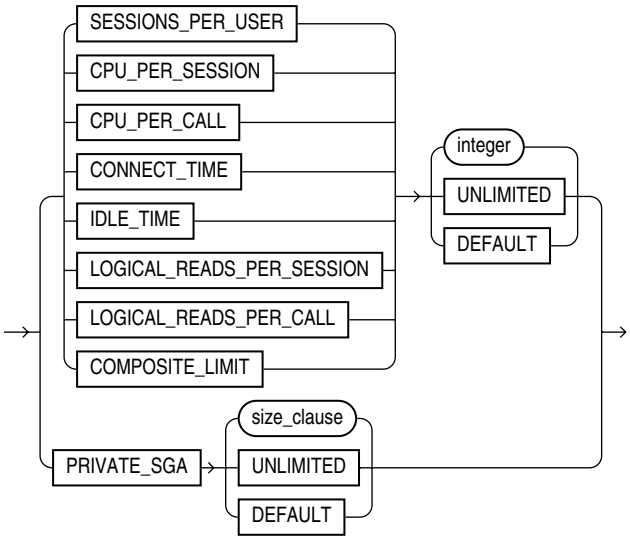
構文

***alter_profile*::=**



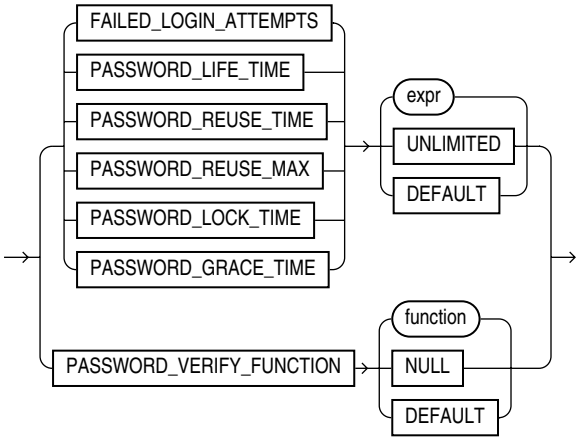
（11-29 ページの resource_parameters::=、11-30 ページの password_parameters::= を参照）

***resource_parameters*::=**



（8-42 ページの size_clause::= を参照）

***password_parameters*::=**



セマンティクス

ALTER PROFILE 文のキーワード、パラメータおよび句の意味は、すべて CREATE PROFILE 文のキーワード、パラメータおよび句と同じです。

DEFAULT プロファイルから制限を削除することはできません。

詳細は、15-47 ページの「CREATE PROFILE」および次の例を参照してください。

例

パスワードを無効にする例： 次の文は、new_profile プロファイル（15-51 ページの「プロファイルの作成例：」で作成）のパスワードを 90 日間再利用できないようにします。

```
ALTER PROFILE new_profile
   LIMIT PASSWORD_REUSE_TIME 90
   PASSWORD_REUSE_MAX UNLIMITED;
```

デフォルトのパスワード値の設定例： 次の文は、app_user プロファイル（15-51 ページの「プロファイルのリソース制限の設定の例：」で作成）の PASSWORD_REUSE_TIME 値を DEFAULT プロファイルに定義された値にデフォルト設定します。

```
ALTER PROFILE app_user
   LIMIT PASSWORD_REUSE_TIME DEFAULT
   PASSWORD_REUSE_MAX UNLIMITED;
```

ログインおよびパスワード・ロック時間の制限例： 次の文は、プロファイル app_user の FAILED_LOGIN_ATTEMPTS を 5 に、PASSWORD_LOCK_TIME を 1 に変更します。

```
ALTER PROFILE app_user LIMIT
   FAILED_LOGIN_ATTEMPTS 5
   PASSWORD_LOCK_TIME 1;
```

この文を使用すると、ログインに 5 回失敗した場合に、app_user のアカウントは 1 日ロックされます。

パスワードの有効期限および猶予期間の変更例： 次の文は、プロファイル app_user2 の PASSWORD_LIFE_TIME を 90 日に、PASSWORD_GRACE_TIME を 5 日に変更します。

```
ALTER PROFILE app_user2 LIMIT
   PASSWORD_LIFE_TIME 90
   PASSWORD_GRACE_TIME 5;
```

同時セッションの制限例： 次の文は、プロファイル app_user に同時実行セッションの新しい制限 5 を指定します。

```
ALTER PROFILE app_user LIMIT SESSIONS_PER_USER 5;
```

現在、プロファイル app_user に SESSIONS_PER_USER の制限が定義されていない場合は、このプロファイルに制限 5 が追加されます。プロファイルに制限が定義されている場合は、前述の文によってその制限が 5 に再定義されます。プロファイル app_user が割り当てられているすべてのユーザーは、同時実行のセッションが 5 件に制限されます。

プロファイル制限の削除例： 次の文は、プロファイル app_user の IDLE_TIME 制限を削除します。

```
ALTER PROFILE app_user LIMIT IDLE_TIME DEFAULT;
```

プロファイル app_user が割り当てられているユーザーは、以降のセッションからプロファイル DEFAULT に定義された IDLE_TIME 制限に従います。

プロファイル・アイドル時間の制限例： 次の文は、プロファイル DEFAULT にアイドル時間の制限（2 分）を定義します。

```
ALTER PROFILE default LIMIT IDLE_TIME  2;
```

IDLE_TIME の制限は、次のユーザーに適用されます。

- プロファイルが明示的に割り当てられていないユーザー
- IDLE_TIME の制限が定義されていないプロファイルが明示的に割り当てられているユーザー

次の文は、プロファイル app_user2 に無制限のアイドル時間を設定します。

```
ALTER PROFILE app_user2 LIMIT IDLE_TIME UNLIMITED;
```

プロファイル app_user2 が割り当てられているすべてのユーザーは、以降のセッションから無制限のアイドル時間が許可されます。

ALTER RESOURCE COST

用途

ALTER RESOURCE COST 文を使用すると、セッションで使用するリソース・コストの合計を算出するための式を指定または変更できます。

Oracle Database は、その他のリソースの使用も監視しますが、セッションに対するリソース・コストの合計は、この構文の 4 種類のリソースに基づいて算出されます。

この文を使用すると、4 種類のリソースに重みを適用できます。Oracle Database は、プロファイルに指定されたこれらのリソースの値に重みを適用し、リソース・コストの合計を算出する計算式を設定します。CREATE PROFILE 文の COMPOSITE_LIMIT パラメータを使用して、セッションに対するコストを制限できます。セッションのリソース・コストが制限を超えた場合、セッションは異常終了し、エラーが戻ります。各リソースに割り当てた重みを変更するために ALTER RESOURCE COST 文を使用した場合、現行のセッション以降のすべてのセッションで、その新しい重みを基にリソース・コストが計算されます。

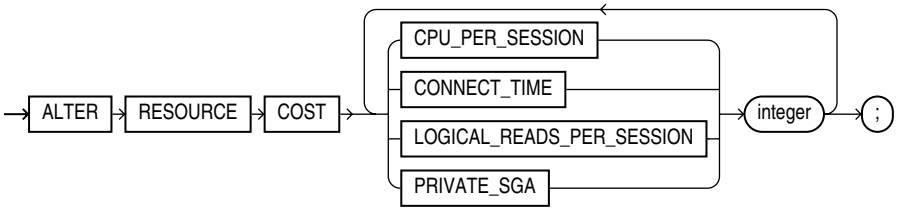
> **参照：** すべてのリソースおよびリソース制限の設定については、15-47 ページの「CREATE PROFILE」を参照してください。

前提条件

ALTER RESOURCE COST システム権限が必要です。

構文

***alter_resource_cost*::=**



セマンティクス

最初にセッションで使用された各リソースの量にそのリソースの重みを乗算し、次に、4 種類のリソースの乗算結果を加算することによって、リソース・コストの合計が計算されます。どのセッションについても、このコストは、ユーザーのプロファイル内の COMPOSITE_LIMIT パラメータの値によって制限されます。乗算結果と総コストは、ともに**サービス単位**と呼ばれる単位で表されます。

CPU_PER_SESSION

このキーワードを使用すると、CPU_PER_SESSION リソースに重みを適用できます。

CONNECT_TIME

このキーワードを使用すると、CONNECT_TIME リソースに重みを適用できます。

LOGICAL_READS_PER_SESSION

この句を使用すると、LOGICAL_READS_PER_SESSION リソースに重みを適用できます。論理読取りには、メモリーおよびディスクの両方から読み取られたブロックが含まれます。

PRIVATE_SGA

この句を使用すると、PRIVATE_SGA リソースに重みを適用できます。共有サーバー・アーキテクチャを使用して、セッション用として SGA 内でプライベート領域を割り当てている場合のみ、この制限が適用されます。

integer

各リソースの重みを指定します。各リソースに割り当てる重みによって、各リソースがリソース・コストの合計に影響する程度が決定されます。リソースに重みを割り当てない場合は、デフォルト値の 0（ゼロ）が適用され、コストへの影響はありません。重みを割り当てた場合は、データベースの次のセッション以降のすべてのセッションで、その重みが適用されます。

例

リソース・コストの変更例： 次の文は、リソース CPU_PER_SESSION と CONNECT_TIME に重みを割り当てます。

```
ALTER RESOURCE COST
   CPU_PER_SESSION 100
   CONNECT_TIME      1;
```

この重みによって、セッションごとに次のコスト計算式が設定されます。

```
cost = (100 * CPU_PER_SESSION) + (1 * CONNECT_TIME)
```

この例では、CPU_PER_SESSION および CONNECT_TIME の値は、DEFAULT プロファイルまたはセッションのユーザーのプロファイルにある値のいずれかです。

ここでは、リソース LOGICAL_READS_PER_SESSION および PRIVATE_SGA に重みを割り当てていないため、これらのリソースは式に含まれません。

プロファイルで COMPOSITE_LIMIT 値として 500 を割り当てた場合、cost が 500 を超えると、必ず、セッションはこの制限を超えます。たとえば、CPU 時間 0.04 秒、経過時間 101 分を使用するセッションは、この制限を超えます。同様に、CPU 時間が 0.0301 秒、経過時間が 200 分のセッションもこの制限を超えます。

一度割り当てた重みは、次のように、別の ALTER RESOURCE 文を発行することによって変更できます。

```
ALTER RESOURCE COST
   LOGICAL_READS_PER_SESSION 2
   CONNECT_TIME 0;
```

新しく割り当てた重みによって、新しいコスト計算式が設定されます。

```
cost = (100 * CPU_PER_SESSION) + (2 * LOGICAL_READ_PER_SECOND)
```

CPU_PER_SESSION および LOGICAL_READS_PER_SECOND の値は、DEFAULT プロファイルまたはセッションのユーザーのプロファイルにある値のいずれかです。

この ALTER RESOURCE COST 文によって、式は次のように変更されます。

- CPU_PER_SESSION リソースの重みは指定しません。このリソースにはすでに重みが割り当てられているため、式では先に指定した重みがそのまま使用されます。
- LOGICAL_READS_PER_SESSION リソースに重みを割り当てたため、このリソースが式で使用されます。
- CONNECT_TIME リソースに 0（ゼロ）を割り当てたため、このリソースは式に含まれていません。
- PRIVATE_SGA リソースの重みは指定しません。このリソースには重みを割り当てていないため、式に含まれていません。

ALTER ROLE

用途

ALTER ROLE 文を使用すると、ロールを使用可能にするために必要な許可を変更できます。

参照：

- ロールの作成の詳細は、15-56 ページの「CREATE ROLE」を参照してください。
- セッションのロールを使用可能または使用禁止にする方法については、19-51 ページの「SET ROLE」を参照してください。

前提条件

ロールに ADMIN OPTION が付与されている必要があります。付与されていない場合は、ALTER ANY ROLE システム権限が必要です。

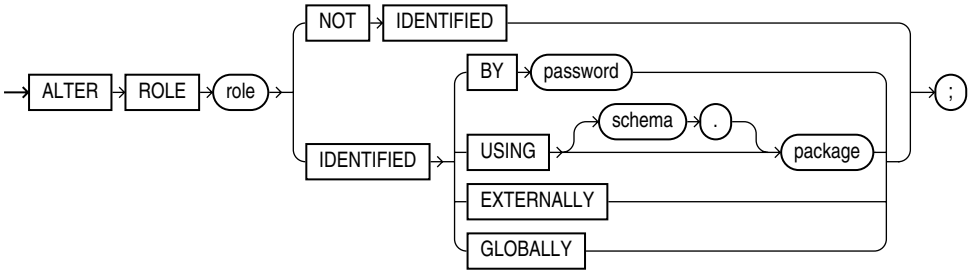
ロールを IDENTIFIED GLOBALLY に変更する前に、次の作業が必要です。

- ロールに対して外部的に識別されたロール権限をすべて取り消します。
- すべてのユーザー、ロールおよび PUBLIC からロールの付与を取り消します。

この規則の唯一の例外として、現在ロールを変更しているユーザーからはそのロールを取り消さないでください。

構文

***alter_role*::=**



セマンティクス

ALTER ROLE 文のキーワード、パラメータおよび句の意味は、すべて CREATE ROLE 文のキーワード、パラメータおよび句と同じです。

ロールの変更の注意事項： ロールの変更には、次の注意事項があります。

- すでに使用可能なロールのユーザー・セッションには影響しません。
- パスワードによって識別されるロールをアプリケーション・ロールに変更する場合（USING *package* 句を使用）、ロールに対応付けられたパスワード情報は失われます。次にロールが使用可能になるときから、新しい認証方式が使用されます。
- ALTER ANY ROLE システム権限を持つユーザーが、IDENTIFIED GLOBALLY ロールを IDENTIFIED BY *password*、IDENTIFIED EXTERNALLY または NOT IDENTIFIED に変更すると、非グローバルなロールを作成した場合と同様に、そのユーザーに変更されたロールが ADMIN OPTION 付きで付与されます。

詳細は、15-56 ページの「CREATE ROLE」および次の例を参照してください。

例

ロール識別機能の変更例： 次の文は、ロール warehouse_user（15-58 ページの「ロールの作成例：」で作成）を NOT IDENTIFIED に変更します。

```
ALTER ROLE warehouse_user NOT IDENTIFIED;
```

ロール・パスワードの変更例： 次の文は、ロール dw_manager（15-58 ページの「ロールの作成例：」で作成）のパスワードを data に変更します。

```
ALTER ROLE dw_manager
   IDENTIFIED BY data;
```

パスワードの変更後、ロール dw_manager が付与されているユーザーは、新しいパスワード data を使用してこのロールを使用可能にする必要があります。

アプリケーション・ロールの例： 次の例は、ロール dw_manager を hr.admin パッケージを使用してアプリケーション・ロールに変更します。

```
ALTER ROLE dw_manager IDENTIFIED USING hr.admin;
```

ALTER ROLLBACK SEGMENT

> **注意：** ロールバック・セグメントを使用せずに、自動 UNDO 管理モードでデータベースを実行することを強くお薦めします。ロールバック・セグメントは、以前のバージョンの Oracle Database との互換性に必要な場合以外は使用しないでください。自動 UNDO 管理の詳細は、『Oracle Database 管理者ガイド』を参照してください。

ALTER ROLLBACK SEGMENT 文を使用すると、ロールバック・セグメントのオンライン / オフライン切替え、記憶特性の変更、またはロールバック・セグメントの最適サイズまたは指定サイズへの縮小を行うことができます。

ここでは、データベースがロールバック UNDO モードで実行されている（初期化パラメータ UNDO_MANAGEMENT に MANUAL を設定、またはすべて設定しない）ことを前提としています。データベースが自動 UNDO モードで実行されている場合（初期化パラメータ UNDO_MANAGEMENT にデフォルト値の AUTO を設定）、ユーザーが作成したロールバック・セグメントは意味を持ちません。

参照：

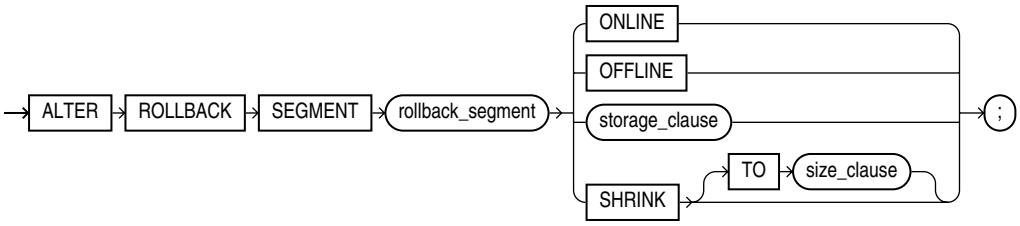
- ロールバック・セグメントの作成については、15-59 ページの「CREATE ROLLBACK SEGMENT」を参照してください。
- UNDO_MANAGEMENT パラメータの詳細は、『Oracle Database リファレンス』を参照してください。

前提条件

ALTER ROLLBACK SEGMENT システム権限が必要です。

構文

***alter_rollback_segment*::=**



（8-43 ページの storage_clause、8-42 ページの size_clause::= を参照）

セマンティクス

rollback_segment

既存のロールバック・セグメントの名前を指定します。

ONLINE

ONLINE を指定すると、ロールバック・セグメントをオンラインにできます。ロールバック・セグメントを作成した場合、最初はオフライン状態になり、トランザクションに使用できなくなります。この句を指定した場合、ロールバック・セグメントはオンラインになり、インスタンスは、トランザクションに対してそのロールバック・セグメントを使用できるようになります。また、初期化パラメータ ROLLBACK_SEGMENTS を使用すると、インスタンスの起動時にロールバック・セグメントをオンラインにできます。

> **参照：** 「ロールバック・セグメントをオンラインにする例：」(11-38 ページ)

OFFLINE

OFFLINE を指定すると、ロールバック・セグメントをオフラインにできます。

- ロールバック・セグメント内に、アクティブ・トランザクションのロールバックに必要な情報が含まれていない場合は、すぐにオフラインになります。
- ロールバック・セグメントにアクティブ・トランザクションについての情報が含まれている場合、このロールバック・セグメントをその後のトランザクションに対して使用できないようにします。また、そのすべてのアクティブ・トランザクションがコミットまたはロールバックされた後、ロールバック・セグメントはオフラインになります。

オフラインになっているロールバック・セグメントは、どのインスタンスからもオンラインにできます。

ロールバック・セグメントがオンラインかオフラインかを確認するには、データ・ディクショナリ・ビュー DBA_ROLLBACK_SEGS の STATUS 列を問い合せます。オンライン・ロールバック・セグメントの値は IN_USE です。オフライン・ロールバック・セグメントの値は AVAILABLE です。

ロールバック・セグメントをオフラインにする場合の制限事項： SYSTEM ロールバック・セグメントをオフラインにすることはできません。

storage_clause

storage_clause を使用すると、ロールバック・セグメントの記憶特性を変更できます。

ロールバック・セグメントの記憶域の制限事項： INITIAL パラメータの値は変更できません。ロールバック・セグメントがローカル管理表領域にある場合、変更可能な記憶域パラメータは OPTIMAL のみです。ロールバック・セグメントがディクショナリ管理表領域にある場合、変更可能な記憶域パラメータは、NEXT、MINEXTENTS、MAXEXTENTS および OPTIMAL のみです。

> **参照：** 構文および追加情報については、8-43 ページの「storage_clause」を参照してください。

SHRINK 句

SHRINK を指定すると、ロールバック・セグメントを最適サイズまたは指定サイズに縮小できます。縮小されるかどうか、および縮小量は、ロールバック・セグメントの使用可能領域およびアクティブ・トランザクションのロールバック・セグメント内での領域保持状態の状況によって異なります。

TO *size_clause* に値を指定しなかった場合、ロールバック・セグメントを作成した CREATE ROLLBACK SEGMENT 文の *storage_clause* の OPTIMAL で指定した値が、デフォルトのサイズになります。OPTIMAL 値を指定しなかった場合、CREATE ROLLBACK SEGMENT 文の *storage_clause* の MINEXTENTS で指定した値がデフォルトのサイズになります。

TO *size_clause* に値を指定するかどうかにかかわらず、次のことがいえます。

- この文の実行時には、ロールバック・セグメントの縮小値が有効です。その後、サイズは CREATE ROLLBACK SEGMENT 文の OPTIMAL 値に戻ります。
- ロールバック・セグメントは、エクステント数 2 未満には縮小できません。

ロールバック・セグメントを縮小した後でロールバック・セグメントの実際のサイズを確認する場合は、DBA_SEGMENTS ビューの BYTES 列、BLOCKS 列および EXTENTS 列を問い合せます。

ロールバック・セグメントの縮小の制限事項： Oracle Real Application Clusters 環境では、インスタンスに対してオンライン状態のロールバック・セグメントのみを縮小できます。

> **参照：** この句の詳細は、8-42 ページの「size_clause」および 11-38 ページの「ロールバック・セグメントのサイズの変更例 :」を参照してください。

例

次の例では、ロールバック・セグメント rbs_one（15-61 ページの「ロールバック・セグメントの作成例 :」で作成）を使用します。

ロールバック・セグメントをオンラインにする例： 次の文は、ロールバック・セグメント rbs_one をオンラインにします。

```
ALTER ROLLBACK SEGMENT rbs_one ONLINE;
```

ロールバック・セグメントのサイズの変更例： 次の文は、ロールバック・セグメント rbs_one を縮小します。

```
ALTER ROLLBACK SEGMENT rbs_one
   SHRINK TO 100M;
```

ALTER SEQUENCE

用途

ALTER SEQUENCE 文を使用すると、既存の順序の増分値、最小値および最大値、キャッシュされる数および動作を変更できます。この文は、順序番号に影響します。

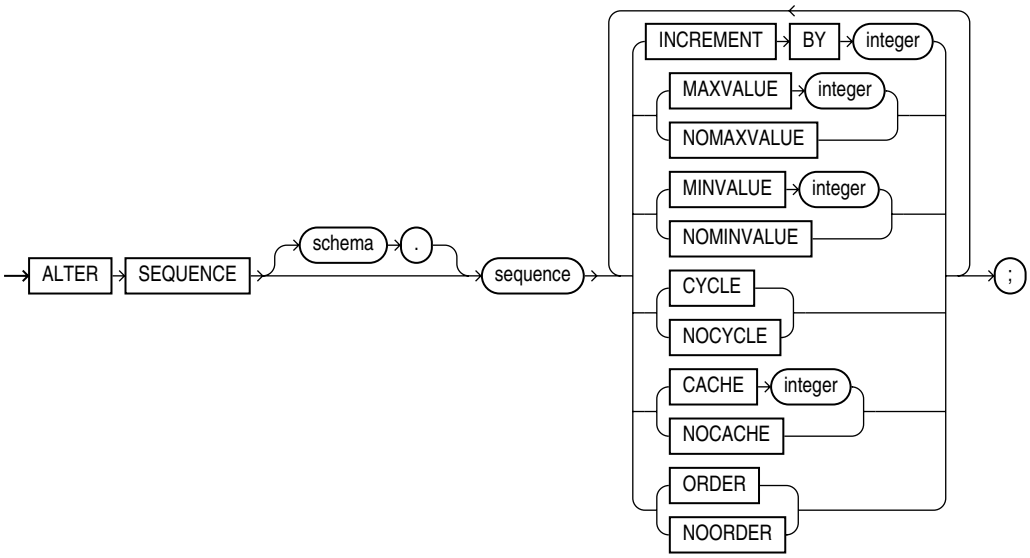
> **参照：** 順序の詳細は、15-64 ページの「CREATE SEQUENCE」を参照してください。

前提条件

順序が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、順序に対する ALTER オブジェクト権限または ALTER ANY SEQUENCE システム権限が必要です。

構文

***alter_sequence*::=**



セマンティクス

この文のキーワードおよびパラメータの意味は、順序を作成する場合と同じです。

- 異なる順序番号で再開する場合、順序を削除して再作成する必要があります。
- NEXTVAL を最初に呼び出す前に、INCREMENT BY の値を変更する場合、いくつかの順序番号がスキップされます。このため、元の START WITH の値を保持するには、順序を削除し、これを元の START WITH の値および新しい INCREMENT BY の値を使用して再作成する必要があります。
- いくつかの妥当性チェックが行われます。たとえば、MAXVALUE の値に現行の順序番号より小さい値は指定できません。

> **参照：** 順序の作成については、15-64 ページの「CREATE SEQUENCE」を参照してください。順序の削除および再作成については、18-2 ページの「DROP SEQUENCE」を参照してください。

例

順序の変更例： 次の文は、customers_seq 順序（15-67 ページの「順序の作成例：」で作成）に新しい最大値を設定します。

```
ALTER SEQUENCE customers_seq
   MAXVALUE 1500;
```

次の文は、customers_seq 順序に CYCLE および CACHE オプションを指定します。

```
ALTER SEQUENCE customers_seq
   CYCLE
   CACHE 5;
```

ALTER SESSION

用途

ALTER SESSION 文を使用すると、データベースへの接続に影響するすべての条件またはパラメータを、設定または変更できます。この文は、データベースとの接続を切断するまで有効です。

前提条件

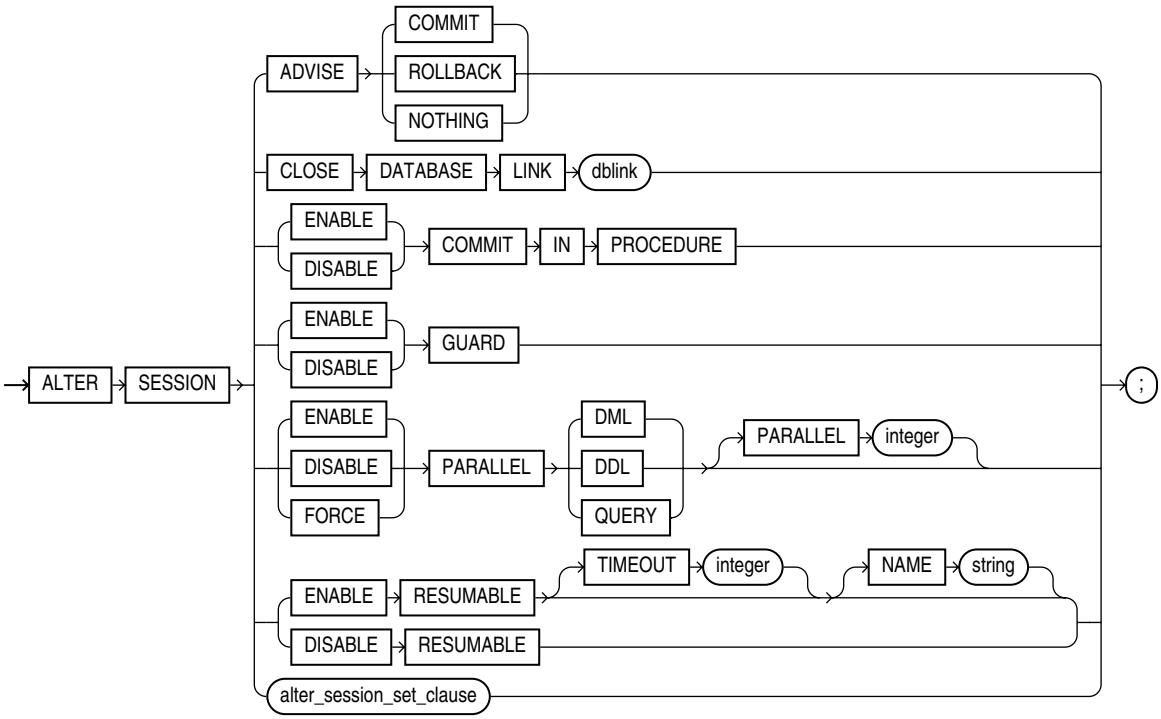
SQL トレース機能を使用可能または使用禁止にするには、ALTER SESSION システム権限が必要です。

再開可能な領域割当てを使用可能または使用禁止にするには、RESUMABLE システム権限が必要です。

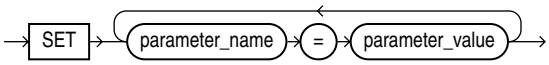
特に指定がないかぎり、これ以外の操作についての権限は必要ありません。

構文

***alter_session*::=**



alter_session_set_clause::=



セマンティクス

ADVISE 句

ADVISE 句を指定すると、分散トランザクションを強制処理するためのアドバイスをリモート・データベースに送ることができます。リモート・データベース内の DBA_2PC_PENDING ビューの ADVICE 列に、アドバイスが表示されます（値 'C' が COMMIT、'R' が ROLLBACK、' ' が NOTHING を示します）。トランザクションの状態がインダウトになった場合、データベース管理者は、このアドバイスを使用してトランザクションをコミットするか、ロールバックするかを決定できます。

単一トランザクションにおいて、ADVISE 句を指定した ALTER SESSION 文を複数発行し、リモート・データベースごとに異なるアドバイスを送ることができます。ADVISE 句を指定した文はそれぞれ、ADVISE 句を指定した別の文が発行されるまで、トランザクション内の後続する文で参照されるデータベースに対してアドバイスを送ります。

参照： 「分散トランザクションを強制的に実行する例 :」（11-49 ページ）

CLOSE DATABASE LINK 句

CLOSE DATABASE LINK を指定すると、データベース・リンク *dblink* をクローズできます。データベース・リンクを使用する SQL 文を発行した場合、Oracle Database は、このデータベース・リンクを使用してリモート・データベース上にセッションを作成します。この接続は、セッションの終了またはデータベース・リンクの数が初期化パラメータ OPEN_LINKS の値を超えるまでオープンされています。リンクをオープンしたままにしておくことによって発生するネットワークのオーバーヘッドを減らすには、セッションでデータベース・リンクを再度使用しない場合に、この句を使用してデータベース・リンクを明示的にクローズします。

参照： 「データベース・リンクをクローズする例 :」（11-50 ページ）

ENABLE | DISABLE COMMIT IN PROCEDURE

プロシージャおよびストアド・ファンクションは PL/SQL で記述されるため、COMMIT 文と ROLLBACK 文を発行できます。アプリケーション自体が直接発行していない COMMIT 文や ROLLBACK 文によって、アプリケーションが中断される場合、DISABLE COMMIT IN PROCEDURE を指定して、セッション中にコールされるプロシージャおよびストアド・ファンクションがこれらの文を発行しないように制御します。

その後、ENABLE COMMIT IN PROCEDURE を指定することによって、セッションでプロシージャおよびストアド・ファンクションが COMMIT および ROLLBACK 文を発行できるようになります。

一部のアプリケーションは、自動的にプロシージャおよびストアド・ファンクションでの COMMIT 文や ROLLBACK 文を禁止します。詳細は、ご使用のアプリケーションのドキュメントを参照してください。

ENABLE | DISABLE GUARD

ALTER DATABASE の *security_clause* を使用すると、SYS ユーザー以外のユーザーは、プライマリ・データベースまたはスタンバイ・データベース上のデータまたはデータベース・オブジェクトを変更できなくなります。この句を使用すると、現行のセッションの設定を上書きできます。

参照： GUARD の設定の詳細は、10-40 ページの「security_clause」を参照してください。

PARALLEL DML | DDL | QUERY

PARALLEL パラメータを使用すると、そのセッションの後続の DML、DDL または問合せ文をパラレル実行するかどうかを指定できます。この句は、現行のセッション中に表自体を変更せずに、表の並列度を上書き可能にします。コミットされていないトランザクションは、DML に対してこの句を実行する前に、コミットまたはロールバックされる必要があります。

参照 : 「パラレル DML を使用可能にする例 :」（11-49 ページ）

ENABLE 句

ENABLE を指定すると、セッション内の後続文をパラレルで実行できます。これは、DDL 文および問合せ文のデフォルトです。

- DML: パラレル・ヒントまたはパラレル句が指定されている場合に、DML 文をパラレル・モードで実行します。
- DDL: パラレル句が指定されている場合に、DDL 文をパラレル・モードで実行します。
- QUERY: パラレル・ヒントまたはパラレル句が指定されている場合に、問合せをパラレル・モードで実行します。

ENABLE 句の制限事項 : オプションの PARALLEL *integer* に ENABLE を指定することはできません。

DISABLE 句

DISABLE を指定すると、セッション内の後続文をシリアルで実行できます。これは、DML 文のデフォルトです。

- DML: DML 文をシリアルで実行します。
- DDL: DDL 文をシリアルで実行します。
- QUERY: 問合せをシリアルで実行します。

DISABLE 句の制限事項 : オプションの PARALLEL *integer* に DISABLE を指定することはできません。

FORCE 句

FORCE を使用すると、セッションの後続文を強制的にパラレル実行できます。パラレル句もパラレル・ヒントも指定されていない場合は、デフォルトの並列度が使用されます。この句は、セッションの後続文に指定されたすべての *parallel_clause* を上書きしますが、パラレル・ヒントによって上書きされます。

- DML: パラレル DML 制限のどれにも違反していない場合、特定の並列度がこの句に指定されていないかぎり、セッションの後続の DML 文は、デフォルトの並列度で実行されます。
- DDL: 特定の並列度がこの句に指定されていないかぎり、セッションの後続の DDL 文は、デフォルトの並列度で実行されます。結果のデータベース・オブジェクトは、通常の並列度に対応します。

  FORCE DDL を指定した場合、そのセッションで作成されるすべての表は、自動的にデフォルトの並列度で作成されます。結果は、CREATE TABLE 文で（デフォルトの並列度を使用して）*parallel_clause* を指定した場合と同じです。
- QUERY: 特定の並列度がこの句に指定されていないかぎり、後続の問合せは、デフォルトの並列度で実行されます。

PARALLEL *integer* 並列度を明示的に指定する整数を指定します。

- FORCE DDL では、並列度は後続の DDL 文のパラレル句を上書きします。
- FORCE DML および FORCE QUERY では、並列度は、データ・ディクショナリの表に格納されている現行の並列度を上書きします。
- ヒントによって文に指定される並列度は、強制的に実行される並列度を上書きします。

次の DML 操作は、この句に関係なくパラレル化されません。

- クラスタ化表に対する操作
- データベースまたはパッケージの状態を読み書きする埋込みファンクションを使用した操作
- 起動する可能性のあるトリガーを使用した表に対する操作
- オブジェクト型、LONG または LOB データ型が含まれている表またはスキーマ・オブジェクトでの操作

RESUMABLE 句

これらの句を使用すると、再開可能な領域割当てを使用可能および使用禁止にできます。この機能によって、領域不足のエラー条件が発生した場合に操作は停止され、エラー条件が修復されたときに中断したところから自動的に再開されます。

> **注意：** 再開可能な領域割当ては、ローカル管理表領域での操作で、完全にサポートされます。ディクショナリ管理表領域を使用する場合は、制限事項があります。制限事項の詳細は、『Oracle Database 管理者ガイド』を参照してください。

ENABLE RESUMABLE

この句を使用すると、セッションに対する再開可能な領域割当てを使用可能にできます。

TIMEOUT TIMEOUT を使用すると、エラー条件が修復されるまで操作を停止する時間を秒単位で指定できます。エラー条件が TIMEOUT で指定した時間までに修復されない場合は、停止操作は異常終了します。

NAME NAME を使用すると、ユーザー定義のテキスト文字列を指定することができ、再開可能モードのセッション中に発行される文の識別に有効です。USER_RESUMABLE データ・ディクショナリ・ビューおよび DBA_RESUMABLE データ・ディクショナリ・ビューに、テキスト文字列が挿入されます。NAME を指定しない場合は、デフォルト文字列「User *username*(*userid*), Session *sessionid*, Instance *instanceid*」が挿入されます。

> **参照：** データ・ディクショナリ・ビューの詳細は、『Oracle Database リファレンス』を参照してください。

DISABLE RESUMABLE

この句を使用すると、セッションに対する再開可能な領域割当てを使用禁止にできます。

alter_session_set_clause

alter_session_set_clause を使用すると、セッションの初期化パラメータの値を設定できます。

初期化パラメータ この句では、2 種類のパラメータを設定できます。

- ALTER SESSION 文の有効範囲内で動的な初期化パラメータ（11-45 ページの「初期化パラメータおよび ALTER SESSION」を参照）
- セッション・パラメータ（11-46 ページの「セッション・パラメータおよび ALTER SESSION」を参照）

同じ *alter_session_set_clause* で複数のパラメータに対する値を設定できます。

初期化パラメータおよび ALTER SESSION

一部の初期化パラメータは、ALTER SESSION の有効範囲内で動的です。ALTER SESSION を使用してこれらのパラメータを設定すると、設定した値は現行セッションでのみ保持されます。パラメータを ALTER SESSION 文で変更できるかどうかを確認するには、V$PARAMETER 動的パフォーマンス・ビューの ISSES_MODIFIABLE 列を問い合せます。

注意： 初期化パラメータの値を変更する前に、『Oracle Database リファレンス』を参照してください。

ALTER SESSION で設定可能な一部のパラメータは、初期化パラメータではありません。初期化パラメータ・ファイルではなく、ALTER SESSION でのみ設定可能です。これらのセッション・パラメータの詳細は、11-46 ページの「セッション・パラメータおよび ALTER SESSION」を参照してください。

セッション・パラメータおよび ALTER SESSION

次のパラメータは、セッション・パラメータであり、初期化パラメータではありません。

CONSTRAINT[S]

構文 :

```
CONSTRAINT[S] = { IMMEDIATE | DEFERRED | DEFAULT }
```

CONSTRAINT[S] は、遅延可能制約によって指定された条件を、いつ適用するかを指定します。

- IMMEDIATE を設定すると、遅延可能な制約によって指定される条件は、各 DML 文の直後にチェックされます。これは、セッションの各トランザクションの開始時に、SET CONSTRAINTS ALL IMMEDIATE 文を発行することと同じです。
- DEFERRED を設定すると、遅延可能な制約によって指定される条件は、トランザクションのコミット時にチェックされます。これは、セッションの各トランザクションの開始時に、SET CONSTRAINTS ALL DEFERRED 文を発行することと同じです。
- DEFAULT を設定すると、すべての制約は各トランザクションの開始時に、DEFERRED または IMMEDIATE の初期状態にリストアされます。

CURRENT_SCHEMA

構文 :

```
CURRENT_SCHEMA = schema
```

CURRENT_SCHEMA は、セッションの現行のスキーマを、指定したスキーマに変更します。セッション中のスキーマ・オブジェクトに対する後続の未修飾の参照は、この指定したスキーマ内でオブジェクトに変換されます。この設定は、現行のセッションの存続期間中、または ALTER SESSION SET CURRENT_SCHEMA 文を再発行するまで保持されます。

この設定によって、現行のユーザーのスキーマ以外にあるオブジェクトに対する操作を、オブジェクトをスキーマ名で修飾することなく簡単に行えます。この設定によって、現行のスキーマは変更されますが、このセッションのユーザーまたは現行のユーザーは変更されません。また、セッション・ユーザーには、このセッションに対する追加のシステム権限またはオブジェクト権限は付与されません。

ERROR_ON_OVERLAP_TIME

構文 :

```
ERROR_ON_OVERLAP_TIME = {TRUE | FALSE}
```

ERROR_ON_OVERLAP_TIME パラメータには、Oracle Database が不明瞭な境界日時値（日時が標準か夏時間かが明確でない場合）を処理する方法を指定します。

- TRUE を指定すると、不明瞭なオーバラップ・タイムスタンプに対してエラーが戻されます。
- FALSE を指定すると、不明瞭なオーバラップ・タイムスタンプは標準時刻のデフォルトになります。これはデフォルトです。

境界日時値の詳細は、2-22 ページの「夏時間のサポート」を参照してください。

FLAGGER

構文 :

```
FLAGGER = { ENTRY | INTERMEDIATE | FULL | OFF }
```

FLAGGER パラメータは、FIPS のフラグ付けを指定します。このフラグ付けを使用した場合、ANSI SQL92 の拡張要素である SQL 文が発行されたときに、エラー・メッセージが生成されます。FLAGGER は、セッション・パラメータであり、初期化パラメータではありません。

Oracle Database では、ENTRY レベル、INTERMEDIATE レベル、FULL レベルのそれぞれのフラグ付けに違いはありません。セッションでフラグ付けが設定されると、これに続く ALTER SESSION SET FLAGGER 文は成功しますが、ORA-00097 のメッセージが生成されます。このため、セッションを切断しなくても FIPS のフラグ付けを変更できます。OFF を設定した場合、フラグ付けの使用は停止されます。

> **参照 :** 現行の ANSI SQL 規格に対する Oracle の準拠については、付録 B「Oracle と標準 SQL」を参照してください。

INSTANCE

構文 :

```
INSTANCE = integer
```

INSTANCE パラメータを設定すると、自分のインスタンスに接続している場合と同様に、別のインスタンスにもアクセスできます。INSTANCE はセッション・パラメータであり、初期化パラメータではありません。Oracle Real Application Clusters（RAC）環境では、このパラメータの設定に基づき、各 RAC インスタンスで、最適な DML パフォーマンスを実現するようにディスク領域の静的または動的な所有権が保持されます。

ISOLATION_LEVEL

構文 :

```
ISOLATION_LEVEL = {SERIALIZABLE | READ COMMITTED}
```

ISOLATION_LEVEL パラメータは、データベースを変更するトランザクションがどのように処理されるかを指定します。ISOLATION_LEVEL はセッション・パラメータであり、初期化パラメータではありません。

- SERIALIZABLE を設定すると、セッション内のトランザクションは、SQL92 に規定されているとおりシリアル化可能トランザクション分離モードを使用します。シリアル化可能トランザクションが行を更新する DML 文を実行する場合、現在の更新対象の行がそのシリアル化可能トランザクションの開始時にコミットされていない別のトランザクションによって更新されていたときは、その DML 文は失敗します。シリアル化可能トランザクションは、同一トランザクション内で行った更新を確認できます。
- READ COMMITTED を設定すると、セッション内のトランザクションは、Oracle Database トランザクションのデフォルトの動作を行います。別のトランザクションで行ロックを保持しておく必要がある DML がトランザクションに指定されていると、DML 文は行ロックが解除されるまで待ち状態になります。

TIME_ZONE

構文 :

```
TIME_ZONE =  '[+ | -] hh:mm'
             | LOCAL
             | DBTIMEZONE
             | 'time_zone_region'
```

TIME_ZONE パラメータには、現行の SQL セッションのデフォルトのローカル・タイムゾーン・オフセットまたは地域名を指定します。TIME_ZONE はセッション・パラメータであり、初期化パラメータではありません。現行のセッションのタイムゾーンを確認するには、組込み関数 SESSIONTIMEZONE を問い合せます（5-163 ページの「SESSIONTIMEZONE」を参照）。

- UTC の前または後の時および分を示す書式マスク（'[+|-]*hh:mm*'）を指定します。*hh:mm* の有効範囲は、-12:00 ～ +14:00 です。
- LOCAL を指定すると、現行の SQL セッションのデフォルトのローカル・タイムゾーン・オフセットが、現行の SQL セッションを起動したときに構築された、元のデフォルトのローカル・タイムゾーン・オフセットに設定されます。
- DBTIMEZONE を指定すると、現行のセッションのタイムゾーンにデータベースのタイムゾーンと一致する値が設定されます。この設定を指定する場合、DBTIMEZONE ファンクションは、データベースのタイムゾーンを UTC オフセットまたはタイムゾーン地域として戻します。これはデータベースのタイムゾーンの設定に依存します。
- 有効な *time_zone_region* を指定します。有効な地域名を表示するには、V$TIMEZONE_NAMES 動的パフォーマンス・ビューの TZNAME 列を問い合せます。この設定を指定する場合、SESSIONTIMEZONE ファンクションは地域の名前を戻します。

注意： 夏時間機能には、タイムゾーン地域名が必要です。地域名は、2 つのタイムゾーン・ファイルに格納されます。デフォルトのタイムゾーン・ファイルは、パフォーマンスを最大にするために一般的なタイムゾーンのみの小さなファイルです。タイムゾーンがデフォルトのファイルに存在しない場合は、環境変数 ORA_TZFILE を使用して完全な（大きい）ファイルへのパスを指定するまで、夏時間はサポートされません。

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

注意： 環境変数 ORA_SDTZ を使用して、クライアント・セッションのデフォルトのタイムゾーンを設定することもできます。この変数の詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

USE_PRIVATE_OUTLINES

構文：

```
USE_PRIVATE_OUTLINES = { TRUE | FALSE | category_name }
```

USE_PRIVATE_OUTLINES パラメータを使用すると、プライベート・アウトラインの使用を制御することができます。このパラメータが使用可能で、アウトライン化された SQL 文が発行された場合、オプティマイザは、USE_STORED_OUTLINES が使用可能なときに使用されるパブリック領域ではなく、そのセッションのプライベート領域からアウトラインを検索します。そのセッションのプライベート領域にアウトラインが存在しない場合、オプティマイザは、文のコンパイルにアウトラインを使用しません。USE_PRIVATE_OUTLINES は、初期化パラメータではありません。

- TRUE に設定すると、要求をコンパイルするときに、オプティマイザは DEFAULT カテゴリのストアド・プライベート・アウトラインを使用します。
- FALSE に設定すると、オプティマイザはストアド・プライベート・アウトラインを使用しません。これはデフォルトです。USE_STORED_OUTLINES が使用可能な場合、オプティマイザはストアド・パブリック・アウトラインを使用します。
- *category_name* に設定すると、要求をコンパイルするときに、オプティマイザは *category_name* カテゴリのストアド・アウトラインを使用します。

USE_PRIVATE_OUTLINES の制限事項 : USE_STORED_OUTLINES が使用可能な場合は、このパラメータを使用可能にできません。

USE_STORED_OUTLINES

構文 :

```
USE_STORED_OUTLINES = { TRUE | FALSE | category_name }
```

USE_STORED_OUTLINES パラメータは、オプティマイザが実行計画を生成するためにストアド・パブリック・アウトラインを使用するかどうかを決定します。USE_STORED_OUTLINES は、初期化パラメータではありません。

- TRUE に設定すると、要求をコンパイルするときに、オプティマイザは DEFAULT カテゴリのストアド・アウトラインを使用します。
- FALSE に設定すると、オプティマイザはストアド・アウトラインを使用しません。これはデフォルトです。
- *category_name* に設定すると、要求をコンパイルするときに、オプティマイザは *category_name* カテゴリのストアド・アウトラインを使用します。

USED_STORED_OUTLINES の制限事項 : USE_PRIVATE_OUTLINES が使用可能な場合は、このパラメータを使用可能にできません。

例

パラレル DML を使用可能にする例 : 次の文は、現行のセッションでパラレル DML モードを使用可能にします。

```
ALTER SESSION ENABLE PARALLEL DML;
```

分散トランザクションを強制的に実行する例 : 次のトランザクションは、データベース・リンク remote によって識別されるデータベース上の employees 表に従業員のレコードを挿入し、local によって識別されるデータベース上の employees 表から従業員のレコードを削除します。

```
ALTER SESSION
   ADVISE COMMIT;

INSERT INTO employees@remote
   VALUES (8002, 'Juan', 'Fernandez', 'juanf@hr.com', NULL,
   TO_DATE('04-OCT-1992', 'DD-MON-YYYY'), 'SA_CLERK', 3000,
   NULL, 121, 20);

ALTER SESSION
   ADVISE ROLLBACK;

DELETE FROM employees@local
   WHERE employee_id = 8002;

COMMIT;
```

このトランザクションには、ADVISE 句を指定した ALTER SESSION 文が 2 つあります。このトランザクションが状態不明（インダウト）になった場合、remote には、最初に指定した ALTER SESSION 文によってアドバイス 'COMMIT' が送信され、local には、2 番目の文によってアドバイス 'ROLLBACK' が送信されます。

データベース・リンクをクローズする例： 次の文は、データベース・リンクを使用している local データベース上の jobs 表を更新し、このトランザクションをコミットして、データベース・リンクを明示的にクローズします。

```
UPDATE jobs@local SET min_salary = 3000
   WHERE job_id = 'SH_CLERK';

COMMIT;

ALTER SESSION
   CLOSE DATABASE LINK local;
```

日付書式の動的な変更例： 次の文は、セッションのデフォルトの日付書式を 'YYYY MM DD-HH24:MI:SS' に動的に変更します。

```
ALTER SESSION
   SET NLS_DATE_FORMAT = 'YYYY MM DD HH24:MI:SS';
```

変更後は、新しい日付書式が次のように適用されます。

```
SELECT TO_CHAR(SYSDATE) Today
   FROM DUAL;

TODAY
-------------------
2001 04 12 12:30:38
```

日付言語の動的な変更例： 次の文は、日付書式要素の言語をフランス語に変更します。

```
ALTER SESSION
   SET NLS_DATE_LANGUAGE = French;

SELECT TO_CHAR(SYSDATE, 'Day DD Month YYYY') Today
   FROM DUAL;

TODAY
---------------------------
Jeudi    12 Avril     2001
```

ISO 通貨の変更例： 次の文は、ISO 通貨記号をアメリカ合衆国の ISO 通貨記号に動的に変更します。

```
ALTER SESSION
   SET NLS_ISO_CURRENCY = America;

SELECT TO_CHAR( SUM(salary), 'C999G999D99') Total
   FROM employees;

TOTAL
------------------
     USD694,900.00
```

小数点文字と桁区切りの変更例： 次の文は、小数点文字をカンマ（,）に、桁区切りをピリオド（.）に動的に変更します。

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

これらの数値書式要素を使用した場合、新しい文字が戻ります。

```
ALTER SESSION SET NLS_CURRENCY = 'FF';

SELECT TO_CHAR( SUM(salary), 'L999G999D99') Total FROM employees;
```

```
TOTAL
---------------------
         FF694.900,00
```

NLS 通貨の変更例 : 次の文は、各国通貨記号を 'DM' に動的に変更します。

```
ALTER SESSION
   SET NLS_CURRENCY = 'DM';

SELECT TO_CHAR( SUM(salary), 'L999G999D99') Total
   FROM employees;

TOTAL
---------------------
         DM694.900,00
```

NLS 言語の変更例 : 次の文は、表示されたエラー・メッセージの言語をフランス語に動的に変更します。

```
ALTER SESSION
   SET NLS_LANGUAGE = FRENCH;

Session modifiee.

SELECT * FROM DMP;

ORA-00942: Table ou vue inexistante
```

言語のソート順の変更例 : 次の文は、言語ソート順序をスペイン語に動的に変更します。

```
ALTER SESSION
   SET NLS_SORT = XSpanish;
```

これによって、文字の値はスペイン語のソート順序に基づいてソートされます。

SQL トレースを使用可能にする例 : 次の文は、セッションに対して SQL トレース機能を使用可能にします。

```
ALTER SESSION
   SET SQL_TRACE = TRUE;
```

クエリー・リライトを使用可能にする例 : 次の文は、明示的に使用禁止にされていないすべてのマテリアライズド・ビューに対する現行のセッションのクエリー・リライトを使用可能にします。

```
ALTER SESSION SET QUERY_REWRITE_ENABLED = TRUE;
```

ALTER SYSTEM

用途

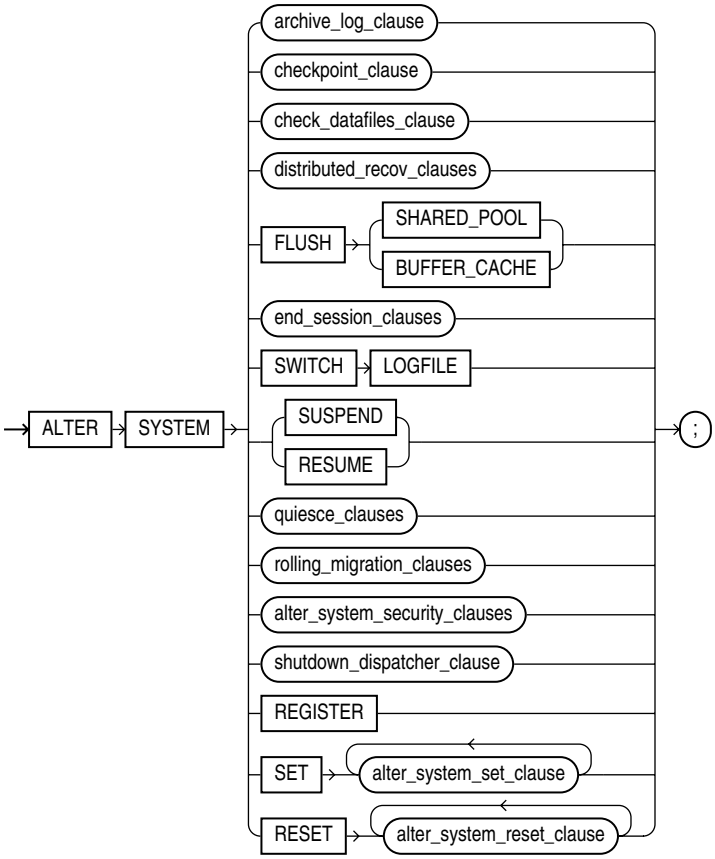
ALTER SYSTEM 文を使用すると、Oracle Database インスタンスを動的に変更できます。この設定は、データベースがマウントされているかぎり有効です。

前提条件

ALTER SYSTEM システム権限が必要です。

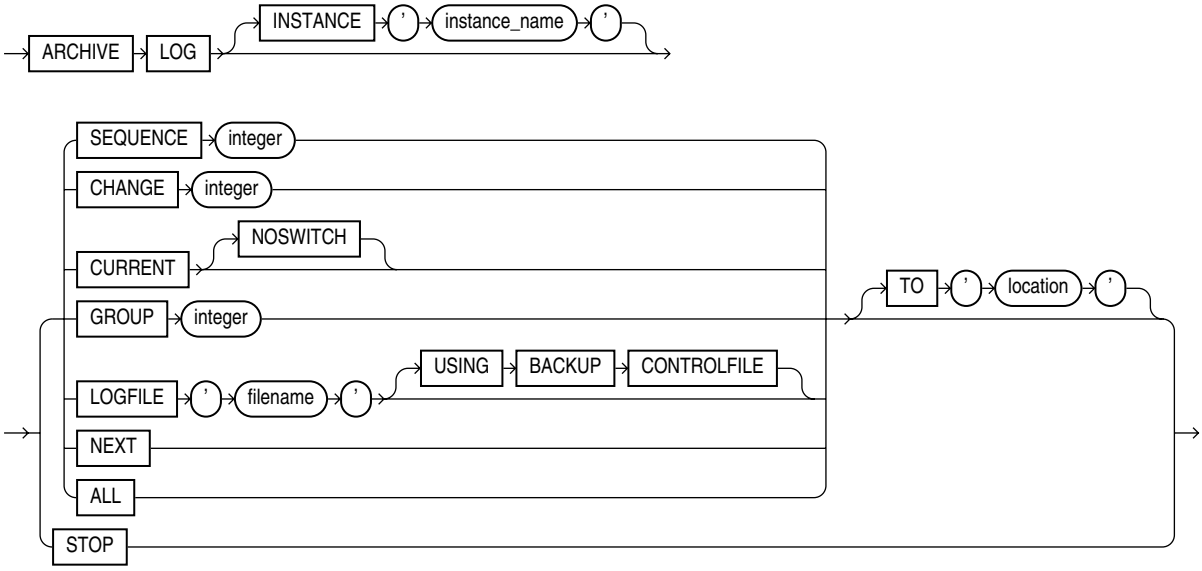
構文

***alter_system*::=**

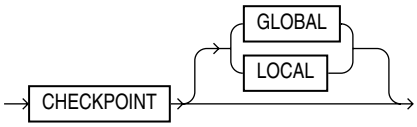


(11-53 ページの archive_log_clause::=、11-53 ページの checkpoint_clause::=、
11-53 ページの check_datafiles_clause::=、11-53 ページの distributed_recov_clauses::=、
11-53 ページの end_session_clauses::=、11-53 ページの quiesce_clauses::=、
11-54 ページの rolling_migration_clauses::=、11-54 ページの alter_system_security_clauses::=、
11-54 ページの shutdown_dispatcher_clause::=、11-54 ページの alter_system_set_clause::=、
11-55 ページの alter_system_reset_clause::= を参照)

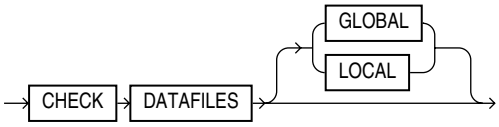
archive_log_clause::=



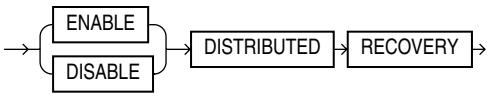
checkpoint_clause::=



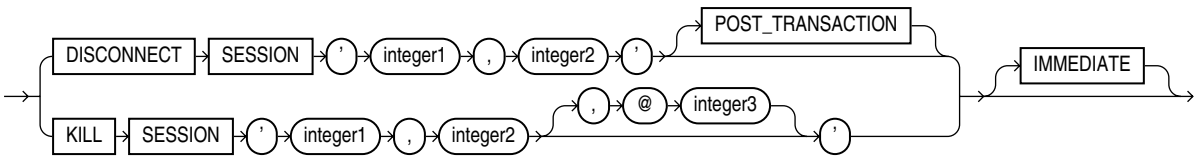
check_datafiles_clause::=



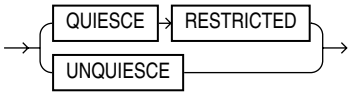
distributed_recov_clauses::=



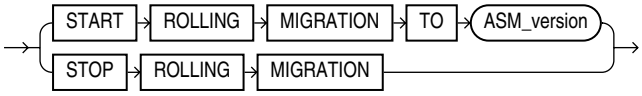
end_session_clauses::=



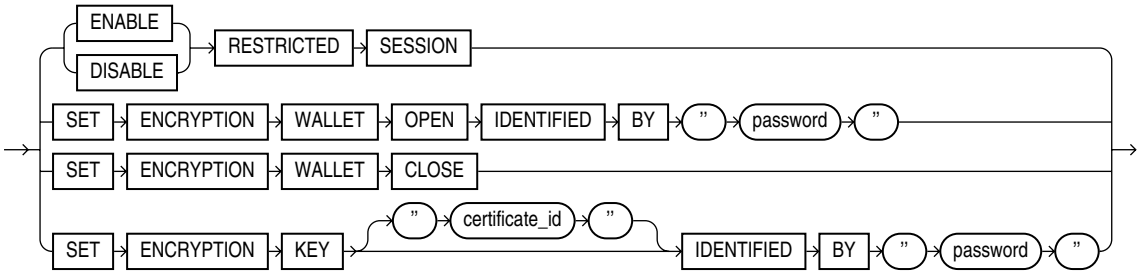
quiesce_clauses::=



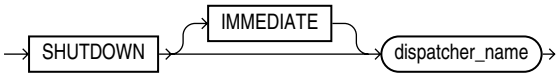
rolling_migration_clauses::=



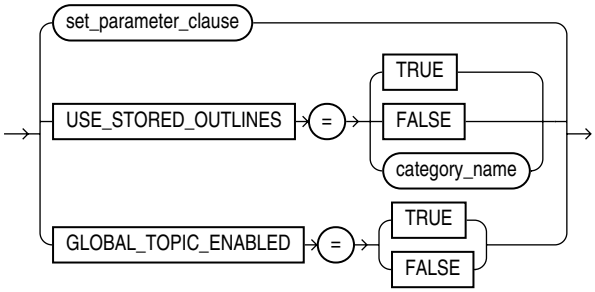
alter_system_security_clauses::=



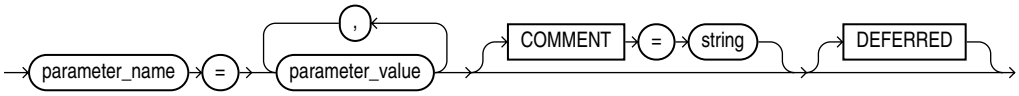
shutdown_dispatcher_clause::=

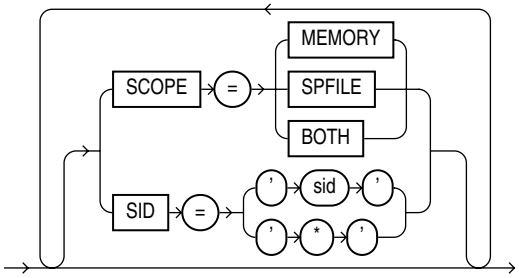


alter_system_set_clause::=

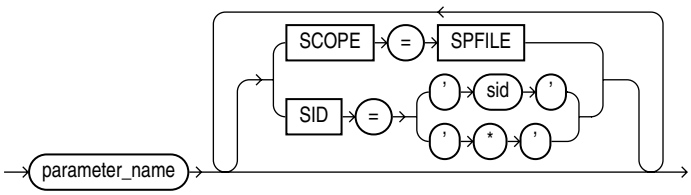


***set_parameter_clause*::=**





alter_system_reset_clause::=



セマンティクス

archive_log_clause

archive_log_clause を使用すると、REDO ログ・ファイルを手動でアーカイブしたり、自動アーカイブを使用可能または使用禁止にすることができます。この句を使用する場合、インスタンスでデータベースをマウントする必要があります。特に指定がないかぎり、データベースはオープンまたはクローズできます。

INSTANCE 句

この句が関係するのは、Oracle Real Application Clusters（RAC）を使用している場合のみです。REDO ログ・ファイル・グループをアーカイブするインスタンスの名前を指定します。インスタンス名は最大 80 文字の文字列です。指定したインスタンスにマップするスレッドは Oracle Database によって自動的に決定され、対応する REDO ログ・ファイル・グループがアーカイブされます。指定したインスタンスにマップされているスレッドがない場合、エラーが戻されます。

SEQUENCE 句

SEQUENCE を指定すると、指定したスレッド内のログ順序番号 *integer* によって識別されるオンライン REDO ログ・ファイル・グループを手動でアーカイブできます。THREAD パラメータを指定しなかった場合、インスタンスに割り当てられているスレッドから、指定したグループがアーカイブされます。

CHANGE 句

CHANGE を指定すると、オンライン REDO ログ・ファイル・グループを、手動でアーカイブできます。このグループには、指定したスレッド内の *integer* によって識別される SCN を持つ REDO ログ・エントリが含まれます。この SCN が現行の REDO ログ・ファイル・グループ内にある場合、ログ・スイッチが実行されます。THREAD パラメータを指定しない場合、使用可能な状態にあるすべてのスレッドから、この SCN を含むグループがアーカイブされます。

インスタンスでデータベースをオープンしている場合にのみ、この句を使用できます。

CURRENT 句

CURRENT を指定すると、ログ・スイッチを強制的に発生させ、指定したスレッドの現行の REDO ログ・ファイル・グループを手動でアーカイブできます。THREAD パラメータを指定しない場合、すべての使用可能なスレッドから、現行のログ以前のログも含むすべての REDO ログ・ファイル・グループがアーカイブされます。データベースがオープンしているときのみ、CURRENT を指定できます。

NOSWITCH NOSWITCH を指定すると、ログ・スイッチの強制実行なしで現行の REDO ログ・ファイル・グループを手動でアーカイブできます。この設定は、プライマリ・データベースが停止したときに、データ分岐が発生しないようにするために、主にスタンバイ・データベースで使用されます。データ分岐は、プライマリ・データベースに障害が発生した場合に、データが消失する可能性があることを意味します。

インスタンスでデータベースがマウントされているがオープンされていない場合にのみ、NOSWITCH 句を使用できます。データベースがオープンしている場合は、この操作によってデータベースは自動的にクローズされます。再オープンする前にデータベースを手動で停止する必要があります。

GROUP 句

GROUP を指定すると、オンライン REDO ログ・ファイル・グループを手動でアーカイブできます。このグループには、*integer* によって識別される GROUP 値が含まれます。REDO ログ・ファイル・グループの GROUP 値を確認するには、データ・ディクショナリ・ビュー DBA_LOG_GROUPS を問い合せます。THREAD パラメータと GROUP パラメータの両方を指定する場合は、指定する REDO ログ・ファイル・グループが、指定するスレッド内に含まれている必要があります。

LOGFILE 句

LOGFILE を指定すると、オンライン REDO ログ・ファイル・グループを手動でアーカイブできます。このグループには、'*filename*' によって識別される REDO ログ・ファイル・メンバーが含まれます。THREAD パラメータと LOGFILE パラメータの両方を指定する場合は、指定する REDO ログ・ファイル・グループが、指定するスレッド内に含まれている必要があります。

データベースがバックアップ制御ファイルでマウントされている場合は、USING BACKUP CONTROLFILE を指定し、現行のログ・ファイルを含むすべてのオンライン・ログ・ファイルのアーカイブを許可します。

LOGFILE 句の制限事項： REDO ログ・ファイル・グループは、一杯になった順にアーカイブする必要があります。LOGFILE パラメータを使用して REDO ログ・ファイル・グループのアーカイブを指定した場合、それ以前の REDO ログ・ファイル・グループがアーカイブされていないとエラー・メッセージが戻ります。

NEXT 句

NEXT を指定すると、一杯になってもアーカイブされていない次のオンライン REDO ログ・ファイルを、指定したスレッドから手動でアーカイブできます。THREAD パラメータを指定しない場合、使用可能な任意のスレッド上の、アーカイブされていない最初の REDO ログ・ファイル・グループがアーカイブされます。

ALL 句

ALL を指定すると、一杯になってもアーカイブされていないすべてのオンライン REDO ログ・ファイルを、指定したスレッドから手動でアーカイブできます。THREAD パラメータを指定しない場合、使用可能なすべてのスレッドから、一杯でアーカイブされていないすべての REDO ログ・ファイル・グループがアーカイブされます。

START 句

以前のリリースでは、この句を使用すると、インスタンスに割り当てられたスレッドについて、REDO ログ・ファイル・グループの自動アーカイブが使用可能になります。この句は現在非推奨になっています。Oracle Database は、REDO ログ・ファイル・グループの自動アーカイブを自動的に使用可能にします。この句は無効です。この句を指定すると、アラート・ログにメッセージが書き込まれます。

TO *location* 句

TO '*location*' を指定すると、REDO ログ・ファイル・グループがアーカイブされる位置を指定できます。このパラメータの値には、オペレーティング・システムの規則に従って、ファイルの位置を完全に指定する必要があります。このパラメータを指定しない場合、REDO ログ・ファイル・グループは初期化パラメータ LOG_ARCHIVE_DEST または LOG_ARCHIVE_DEST_*n* に指定された場所に格納されます。

STOP 句

以前のリリースでは、この句を使用すると、インスタンスに割り当てられたスレッドについて、REDO ログ・ファイル・グループの自動アーカイブが使用禁止になります。この句は現在非推奨になっています。この句は無効です。この句を指定すると、アラート・ログにメッセージが書き込まれます。

checkpoint_clause

CHECKPOINT を指定すると、チェックポイントを明示的に強制処理して、コミット済のトランザクションによる変更をディスク上のデータ・ファイルに書き込むことができます。インスタンスでデータベースがオープンしている場合にのみ、この句を指定できます。チェックポイントが完了するまで、ユーザーに制御は戻りません。

GLOBAL Oracle Real Application Clusters（RAC）環境で、データベースをオープンしているすべてのインスタンスに対してチェックポイントを実行します。これはデフォルトです。

LOCAL Oracle RAC 環境で、文を発行するインスタンスの REDO ログ・ファイル・グループのスレッドに対してのみチェックポイントを実行します。

参照：「チェックポイントの発生例：」（11-66 ページ）

check_datafiles_clause

Oracle RAC 環境などの分散データベース・システムで、データベース制御ファイルからインスタンスの SGA を更新し、すべてのオンライン・データ・ファイルに情報を反映します。

- GLOBAL を指定すると、データベースをオープンしているすべてのインスタンスに対して、この同期化を実行できます。これはデフォルトです。
- LOCAL を指定すると、ローカル・インスタンスに対してのみこの同期化を実行できます。

インスタンスでデータベースをオープンしておく必要があります。

end_session_clauses

end_session_clauses を使用すると、現行のセッションを終了することができます。

DISCONNECT SESSION 句

DISCONNECT SESSION 句を使用すると、専用サーバー・プロセス（共有サーバーによって接続が確立されていた場合は、仮想回路）を破棄することによって、現行のセッションが切断されます。この句を使用する場合、インスタンスでデータベースをオープンする必要があります。次の両方の値を V$SESSION ビューで確認して、このセッションを識別する必要があります。

- *integer1* には、SID 列の値を指定します。
- *integer2* には、SERIAL# 列の値を指定します。

システム・パラメータを適切に設定した場合、アプリケーション・フェイルオーバーが有効になります。

- POST_TRANSACTION を設定すると、セッションが切断される前に、実行中のトランザクションを完了できます。セッションに実行中のトランザクションがない場合、この句は、後述の KILL SESSION と同様の効果があります。
- IMMEDIATE を設定すると、実行中のトランザクションの完了を待たずにセッションを切断し、すぐにセッション全体の状態をリカバリできます。
  - POST_TRANSACTION を指定した場合、セッションに実行中のトランザクションがあれば、IMMEDIATE キーワードは無視されます。
  - POST_TRANSACTION を指定しない場合、または POST_TRANSACTION を指定していてもセッションに実行中のトランザクションがない場合、この句は、後述の KILL SESSION IMMEDIATE と同様の効果があります。

参照：「セッションの切断例：」（11-68 ページ）

KILL SESSION 句

KILL SESSION を指定すると、セッションに終了済のマークが付き、実行中のトランザクションがロールバックされ、すべてのセッション・ロックが解放され、セッション・リソースが一部リカバリされます。この句を使用する場合、インスタンスでデータベースをオープンする必要があります。*integer3* を指定しない場合は、自セッションおよび終了されるセッションは、同じインスタンスにある必要があります。次の値を V$SESSION ビューで確認して、このセッションを識別する必要があります。

- *integer1* には、SID 列の値を指定します。
- *integer2* には、SERIAL# 列の値を指定します。
- オプションの *integer3* には、終了されるターゲット・セッションが存在するインスタンスの ID を指定します。GV$ 表を問い合せることによって、インスタンス ID を確認することができます。

リモート・データベースからの応答を待ったり、トランザクションをロールバックするなど、最後まで完了する必要があるアクティビティをセッションが実行している場合、Oracle Database はこのアクティビティが完了するまで待機し、セッションに終了済のマークを付け、その後、ユーザーに制御を戻します。待ち時間が 1 分以上続く場合は、終了されるセッションにマークが付けられ、マークが付けられたセッションが終了されることを示すメッセージとともにユーザーに制御が戻されます。アクティビティが完了すると、PMON バックグラウンド・プロセスは、セッションに終了済のマークを付けます。

セッションに実行中のトランザクションがあるかどうかにかかわらず、セッション・ユーザーがセッションに要求を発行してセッションが終了されたことを示すメッセージを受け取るまで、Oracle Database は、セッション全体の状態をリカバリしません。

参照： 「セッションの終了例：」（11-67 ページ）

IMMEDIATE IMMEDIATE を指定すると、実行中のトランザクションをロールバックしてすべてのセッション・ロックを解放し、セッション全体の状態をリカバリしてから、すぐにユーザーに制御を戻すように Oracle Database に指示できます。

distributed_recov_clauses

DISTRIBUTED RECOVERY 句を使用すると、分散リカバリを使用可能または使用禁止にできます。この句を使用する場合、インスタンスでデータベースをオープンする必要があります。

ENABLE ENABLE を指定すると、分散リカバリを使用可能にできます。シングルプロセス環境では、分散リカバリを開始する場合にこの句を使用する必要があります。

トランザクションに関係があるリモート・ノードにアクセスできない場合、インダウト・トランザクションをリカバリするには、ENABLE DISTRIBUTED RECOVERY 文を複数回発行する必要がある場合もあります。インダウト・トランザクションは、データ・ディクショナリ・ビュー DBA_2PC_PENDING に表示されます。

参照： 「分散リカバリを使用可能にする例：」（11-67 ページ）

DISABLE DISABLE を指定すると、分散リカバリを使用禁止にできます。

FLUSH SHARED_POOL 句

FLUSH SHARED_POOL 句を指定すると、SGA の共有プール上のすべてのデータが消去されます。共有プールは次のものを格納します。

- キャッシュされたデータ・ディクショナリ情報
- SQL 文の共有 SQL 領域、共有 PL/SQL 領域、ストアド・プロシージャ、ファンクション、パッケージおよびトリガー

この文は、現在実行中の項目に対する共有 SQL 領域および共有 PL/SQL 領域を消去しません。インスタンスでデータベースがマウントされていてもディスマウントされていても、またはオープン状態でもクローズ状態でも、この句を使用できます。

参照：「共有プールの消去例：」（11-66 ページ）

FLUSH BUFFER_CACHE 句

FLUSH BUFFER_CACHE 句を指定すると、システム・グローバル領域（SGA）のバッファ・キャッシュ上のすべてのデータを消去できます。

注意： この句は、テスト・データベース上のみで使用します。この句を本番データベース上で使用しないでください。この文を実行すると、後続の問合せで結果が戻らなくなります。

この句は、リライトされた問合せ、または同一の開始点からの一連の問合せのパフォーマンスを測定する必要がある場合に有効です。

SWITCH LOGFILE 句

SWITCH LOGFILE 句を指定すると、現行の REDO ログ・ファイル・グループのファイルが一杯であるかどうかにかかわらず、新しい REDO ログ・ファイル・グループへの書込みを明示的かつ強制的に開始できます。ログ・スイッチを強制的に発生させると、チェックポイントが実行されますが、チェックポイントが完了する前に、すぐに制御が戻されます。この句を使用する場合、インスタンスでデータベースをオープンする必要があります。

参照：「ログ・スイッチの発生例：」（11-67 ページ）

SUSPEND | RESUME

SUSPEND 句を指定すると、すべての I/O（データ・ファイル、制御ファイルおよびファイル・ヘッダー）および問合せを停止し、すべてのインスタンスで実行中のトランザクションを処理せずにデータベースのコピーが作成可能になります。

SUSPEND および RESUME の制限事項： SUSPEND と RESUME には、次の制限事項があります。

- ホット・バックアップ・モードでデータベースの表領域を確保するまで、この句は使用できません。
- ALTER SYSTEM SUSPEND 文を発行したセッションは終了しないでください。システムが一時停止しているときに再接続しようとすると、SYS ログイン中に実行される再帰的 SQL が原因となって接続が失敗することがあります。
- システムが停止中に新しいインスタンスを起動する場合、この新しいインスタンスは停止しません。

RESUME 句を指定すると、問合せおよび I/O に対して、再度、データベースが使用可能になります。

rolling_migration_clauses

クラスタ化された自動ストレージ管理（ASM）環境内でこれらの句を使用すると、ASM クラスタまたはストレージに ASM を使用するデータベース・クラスタの全体的な可用性に影響することなく、ノードを一度に 1 つずつ別の ASM バージョンに移行できます。

START ROLLING MIGRATION　ローリング・アップグレードを開始するときに、*ASM_version* について、次の 5 つの文字列を指定する必要があります。

```
<version_num>, <release_num>, <update_num>,<port_release_num>,<port_update_num>
```

ASM_version は、11.1.0.0.0 以上である必要があります。次に、自動ストレージ管理は、最初に、指定されたリリースへの移行について現在のリリースに互換性があることを確認し、制限された機能モードになります。自動ストレージ管理は、次に、クラスタ内で均衡の再調整処理が進行中かどうかを確認します。そのような操作がある場合、文は失敗し、均衡の再調整操作の完了後に文を再発行する必要があります。

ローリング・アップグレード・モードは、クラスタ全体でメモリー内に永続的に保持される状態です。クラスタは、少なくとも 1 つの ASM インスタンスがクラスタ内で実行中になるまで、この状態であり続けます。クラスタに参加する新しいインスタンスは、起動時にすぐに移行モードに切り替わります。クラスタ内のすべてのインスタンスが終了すると、この後に自動ストレージ管理インスタンスを起動しても、この文を再発行して自動ストレージ管理インスタンスのローリング・アップグレードを再開するまでは、ローリング・アップグレード・モードにはなりません。

STOP ROLLING MIGRATION　この句を使用すると、ローリング・アップグレードを停止して、クラスタを通常の操作に戻すことができます。クラスタ内のすべてのインスタンスが同じソフトウェア・バージョンに移行した後にのみ、この句を指定します。クラスタがローリング・アップグレード・モードではない場合、文は失敗します。

この句を指定すると、自動ストレージ管理インスタンスはクラスタのすべてのメンバーが同じソフトウェア・バージョンであることを検証し、インスタンスをローリング・アップグレード・モードから除外して、自動ストレージ管理クラスタの完全な機能に戻ります。ディスクがオフラインであるために均衡の再調整操作が保留中の場合、操作は再起動されます。ただし、そのような再起動によって ASM_POWER_LIMIT パラメータに違反することにならない場合にかぎります。

> **参照：** ローリング・アップグレードの詳細は、『Oracle Database ストレージ管理者ガイド』を参照してください。

quiesce_clauses

QUIESCE RESTRICTED 句および UNQUIESCE 句を使用すると、データベースを**静止状態**にしたり、静止状態から戻すことができます。この状態では、データベース管理者は、トランザクション、問合せまたは PL/SQL 操作が同時に存在する状態では安全に実行できない管理操作を実行することができます。

> **注意：** QUIESCE RESTRICTED 句は、データベース・リソース・マネージャがインストールされている場合、およびデータベースをオープンしたインスタンスでデータベースが起動された後、リソース・マネージャが継続的にアクティブになっている場合のみ有効です。

複数の QUIESCE RESTRICTED 文または UNQUIESCE 文が異なるセッションまたはインスタンスで同時に発行された場合、1 つを除いた他のすべてのセッションまたはインスタンスにエラーが戻されます。

QUIESCE RESTRICTED

QUIESCE RESTRICTED を指定すると、データベースを静止状態にできます。この句は、データベースがオープン中のすべてのインスタンスに次の影響を与えます。

- Oracle Database は、すべてのインスタンスのデータベース・リソース・マネージャに、アクティブでないすべてのセッション（SYS および SYSTEM 以外）をアクティブにしないように指示します。SYS および SYSTEM 以外のユーザーは、新しいトランザクション、問合せ、フェッチまたは PL/SQL 操作を開始できません。

- Oracle Database は、SYS または SYSTEM 以外のユーザーが開始した、すべてのインスタンスの既存のトランザクションが終了するまで（コミットまたは異常終了するまで）待機します。また、Oracle Database は、SYS または SYSTEM 以外のユーザーが開始した、内部トランザクションにない、すべてのインスタンスで実行中のすべての問合せ、フェッチおよび PL/SQL プロシージャが終了するまで待機します。連続する複数の Oracle Call Interface（OCI）のフェッチによって問合せが実行される場合、Oracle Database はすべてのフェッチが終了するまで待機しません。現行のフェッチが終了するまで待機しますが、次のフェッチは行われません。Oracle Database は、エンキューなどの共有リソースを保持するすべてのセッション（SYS および SYSTEM 以外のセッション）がリソースを解放するまで待機します。すべての操作が完了した後、Oracle Database はデータベースを静止状態にし、QUIESCE RESTRICTED 文の実行を終了します。
- 共有サーバー・モードでインスタンスを実行している場合、Oracle Database は、SYS または SYSTEM 以外のユーザーがそのインスタンスにログインすることを阻止するようにデータベース・リソース・マネージャに指示します。非共有サーバー・モードでインスタンスを実行している場合、そのインスタンスへのユーザー・ログインに制限はありません。

静止状態中、すべてのインスタンスにおいてリソース・マネージャのプランは変更できません。

UNQUIESCE

UNQUIESCE を指定すると、データベースを静止状態から戻すことができます。これによって、SYS または SYSTEM 以外のユーザーによって開始された、トランザクション、問合せ、フェッチおよび PL/SQL プロシージャを再開できます。UNQUIESCE 文は、QUIESCE RESTRICTED 文を発行したセッションと同じセッションで起動する必要はありません。

alter_system_security_clauses

alter_system_security_clauses を使用すると、インスタンスへのアクセスを制御できます。

RESTRICTED SESSION

RESTRICTED SESSION 句を指定すると、Oracle Database にログインできるユーザーを制限できます。インスタンスでデータベースがマウントされていてもディスマウントされていても、またはオープン状態でもクローズ状態でも、この句を使用できます。

- ENABLE を指定すると、RESTRICTED SESSION システム権限が付与されているユーザーのみが Oracle Database にログインできます。既存のセッションは終了しません。

  この句は、現行のインスタンスのみに適用されます。そのため、Oracle RAC 環境では、RESTRICTED SESSION システム権限を持たない認可済ユーザーも他のインスタンスでデータベースにアクセスできます。
- DISABLE を指定すると、ENABLE RESTRICTED SESSION 句の効果を無効にできます。つまり、CREATE SESSION システム権限が付与されているすべてのユーザーが Oracle Database にログインできるようになります。これはデフォルトです。

参照： 「セッションを制限する例：」（11-66 ページ）

SET ENCRYPTION WALLET 句

この句を使用すると、サーバー・ウォレット内の情報へのデータベース・アクセスを管理できます。この文は、キーワード ALTER で始まりますが、ALTER SYSTEM SET ENCRYPTION WALLET 文は DDL 句ではありません。ただし、このような文はロールバックすることはできません。

OPEN　この句を指定すると、データベースは指定されたパスワードを使用してサーバー・ウォレット内の情報をメモリーにロードし、インスタンスの存続期間中にデータベース・アクセスができるようになります。この句を使用すると、データベースは、SSO ウォレットなしにサーバー・ウォレットからキーを取得できます。サーバー・ウォレットが使用できないか、すでに開いている場合は、データベースからエラーが戻ります。このコンテキストでは、パスワードを二重引用符で囲む必要があります。

CLOSE この句を使用すると、メモリーからサーバー・ウォレットの情報が削除されます。

> **参照：** Oracle Real Application Clusters（Oracle RAC）環境での暗号化ウォレットの設定については、『Oracle Real Application Clusters 管理およびデプロイメント・ガイド』を参照してください。

SET ENCRYPTION KEY 句

この句を使用すると、新しい暗号化キーを生成し、現行のデータの透過的暗号化のマスター鍵として設定できます。また、この句はサーバー・ウォレットのデータベース・アクセス情報をメモリーにロードします。*certificate_id* は、証明書を識別する整数です。これは、基本鍵を使用する場合は不要ですが、PKI ベースの鍵を使用する場合は必要になります。この値は、V$WALLET 動的パフォーマンス・ビューの CERT_ID 列を問い合せて確認できます。*password* には、セキュリティ・モジュールへの接続に使用するパスワードを指定します。指定した *certificate_id* またはパスワードが無効な場合、データベースからエラーが戻ります。このコンテキストでは、パスワードを二重引用符で囲む必要があります。

ALTER SYSTEM SET KEY 文は DDL 文です。この文は、スキーマ内で保留中のトランザクションを自動的にコミットします。

データの透過的暗号化機能を使用するには、暗号化ウォレットと暗号化キーの両方を設定する必要があります。

> **参照：**
>
> - サーバー・ウォレットと暗号化キーの使用方法およびデータの透過的暗号化の詳細は、『Oracle Database Advanced Security 管理者ガイド』を参照してください。
> - この機能を利用して表の列を暗号化する方法の詳細は、16-26 ページの「CREATE TABLE」の「encryption_spec」を参照してください。
> - 「ウォレットおよび暗号化キーの確立」（11-66 ページ）

shutdown_dispatcher_clause

SHUTDOWN 句は、システムに Oracle Database の共有サーバー・アーキテクチャを使用している場合のみ有効です。*dispatcher_name* で識別されたディスパッチャを停止します。

> **注意：** この句を SQL*Plus コマンド SHUTDOWN（データベース全体を停止するために使用する）と混同しないでください。

dispatcher_name は、'D*xxx*' という形式の文字列である必要があります。*xxx* はディスパッチャの番号です。ディスパッチャ名のリストを取得するには、V$DISPATCHER 動的パフォーマンス・ビューの NAME 列を問い合せます。

- IMMEDIATE を指定した場合、ディスパッチャは新しい接続の受入れをすぐに中止し、そのディスパッチャによる既存の接続はすべて終了されます。すべてのセッションがクリーンアップされてから、ディスパッチャ・プロセスは停止します。
- IMMEDIATE を指定しない場合、ディスパッチャは新しい接続の受入れをすぐに中止しますが、すべてのユーザーが切断し、すべてのデータベース・リンクが終了されるのを待ちます。その後、ディスパッチャは停止されます。

REGISTER 句

REGISTER を指定すると、PMON バックグラウンド・プロセスによってリスナーにインスタンスがすぐに登録されます。この句を指定しない場合、PMON が次に検出ルーチンを実行するまでインスタンスの登録は行われません。その結果、クライアントは、リスナー起動後最大 60 秒間サービスにアクセスできない可能性があります。

参照： PMON バックグラウンド・プロセスおよびリスナーの詳細は、『Oracle Database 概要』および『Oracle Database Net Services 管理者ガイド』を参照してください。

alter_system_set_clause

従来のプレーン・テキストのパラメータ・ファイル（PFILE）を使用してデータベースを起動したか、サーバー・パラメータ・ファイル（SPFILE）を使用してデータベースを起動したかに応じて、現行のインスタンスの多くの初期化パラメータ値を変更できます。『Oracle Database リファレンス』では、各パラメータの説明で、これらのパラメータが「変更可能」というカテゴリに分類されています。PFILE を使用した場合、変更はインスタンスの存続期間中のみ保持されます。一方、SPFILE を使用してデータベースを起動した場合、SPFILE 自体のパラメータの値を変更できるため、後続のインスタンスで新しい値が使用されます。

『Oracle Database リファレンス』には、すべての初期化パラメータが詳細に記載されています。パラメータは、次の 3 つのカテゴリに分類されます。

- **基本パラメータ :** データベース管理者は、すべての基本パラメータについて熟知し、これらのパラメータの設定を考慮する必要があります。
- **機能のカテゴリ :** 『Oracle Database リファレンス』には、初期化パラメータが機能のカテゴリ別にも示されています。
- **アルファベット順 :** 『Oracle Database リファレンス』の目次には、すべての初期化パラメータがアルファベット順に示されています。

初期化パラメータ値を変更する権限は、従来のプレーン・テキストの初期化パラメータ・ファイル（pfile）を使用してデータベースを起動したか、サーバー・パラメータ・ファイル（spfile）を使用してデータベースを起動したかによって異なります。特定のパラメータ値を変更する権限を持っているかどうかを確認するには、V$PARAMETER 動的パフォーマンス・ビューの ISSYS_MODIFIABLE 列を問い合せます。

set_parameter_clause

パラメータ値を設定するときに、次の設定も行えます。

COMMENT COMMENT 句を使用すると、コメント文字列をパラメータ値の変更に対応付けることができます。SPFILE をあわせて指定すると、パラメータ・ファイルにコメントが表示され、そのパラメータに対する直前の変更がわかります。

DEFERRED DEFERRED キーワードを指定すると、データベースに接続するその後のセッションに対するパラメータの値を設定または変更できます。現行のセッションでは変更前の値が残ります。

このパラメータの V$PARAMETER の ISSYS_MODIFIABLE 列の値が DEFERRED の場合は、DEFERRED を指定する必要があります。この列の値が IMMEDIATE の場合、この句の DEFERRED キーワードはオプションです。この列の値が FALSE の場合、この ALTER SYSTEM 文では DEFERRED を指定できません。

参照： V$PARAMETER 動的パフォーマンス・ビューの詳細は、『Oracle Database リファレンス』を参照してください。

SCOPE SCOPE 句を使用すると、変更が有効になるタイミングを指定できます。有効範囲は、データベースの起動に使用したファイルが従来のプレーン・テキストのパラメータ・ファイル（pfile）か、サーバー・パラメータ・ファイル（spfile）かによって異なります。

- MEMORY を指定すると、変更がメモリーで行われ、すぐに有効になり、データベースが停止するまで持続されます。パラメータ・ファイル（pfile）を使用してデータベースを起動した場合、この有効範囲のみを指定できます。

- SPFILE を指定すると、変更がサーバー・パラメータ・ファイルで行われます。新しい設定は、データベースが次に停止し、再起動されたときに有効になります。『Oracle Database リファレンス』に変更不可能と示されている静的パラメータ値を変更する場合は、SPFILE を指定する必要があります。
- BOTH を指定すると、変更がメモリーとサーバー・パラメータ・ファイルの両方で行われます。新しい設定はすぐに有効になり、データベースが停止し、再起動された後も持続します。

データベースの起動でサーバー・パラメータ・ファイルを使用した場合は、BOTH がデフォルトです。データベースの起動でパラメータ・ファイルを使用した場合は、MEMORY がデフォルトで、これ以外の有効範囲は指定できません。

SID SID 句を使用すると、値を有効にするインスタンスの SID を指定できます。

- このパラメータがまだ明示的に設定されていないすべてのインスタンスに対してパラメータ値を Oracle Database で変更する場合は、SID = '*' を指定します。
- *sid* のインスタンスのみでパラメータ値を変更する場合、SID = *'sid'* を指定します。この設定は、SID = '*' を指定する前後の ALTER SYSTEM SET 文より優先されます。

この句を指定しない場合は、次のようになります。

- pfile（従来のプレーン・テキストの初期化パラメータ・ファイル）を使用してインスタンスを起動する場合、現行のインスタンスの SID を指定したとみなされます。
- spfile（サーバー・パラメータ・ファイル）を使用してインスタンスを起動する場合、SID = '*' を指定したとみなされます。

現行のインスタンス以外のインスタンスを指定すると、そのインスタンスに、メモリーのパラメータ値の変更を通知するメッセージが送信されます。

> **参照：** V$PARAMETER ビューの詳細は、『Oracle Database リファレンス』を参照してください。

USE_STORED_OUTLINES 句

USE_STORED_OUTLINES は、システム・パラメータであり、初期化パラメータではありません。pfile または spfile 内に設定することはできませんが、ALTER SYSTEM 文とともに設定できます。このパラメータは、オプティマイザが実行計画を生成するためにストアド・パブリック・アウトラインを使用するかどうかを指定します。

- TRUE に設定すると、要求をコンパイルするときに、オプティマイザは DEFAULT カテゴリのストアド・アウトラインを使用します。
- FALSE に設定すると、オプティマイザはストアド・アウトラインを使用しません。これはデフォルトです。
- *category_name* に設定すると、要求をコンパイルするときに、オプティマイザは *category_name* カテゴリのストアド・アウトラインを使用します。

GLOBAL_TOPIC_ENABLED

GLOBAL_TOPIC_ENABLED は、システム・パラメータであり、初期化パラメータではありません。pfile または spfile 内に設定することはできませんが、ALTER SYSTEM 文とともに設定できます。このパラメータは、Oracle Streams AQ で作成されたすべてのキューおよびトピックが LDAP サーバーに自動的に登録されるかどうかを指定します。GLOBAL_TOPIC_ENABLED = TRUE の場合、キュー表が作成、変更または削除されると、対応する Lightweight Directory Access Protocol（LDAP）エントリも作成、変更または削除されます。

このパラメータは、Java Message Service（JMS）に対しても同様に機能します。LDAP を使用するようにデータベースが構成されており、GLOBAL_TOPIC_ENABLED パラメータが TRUE に設定されている場合、すべての JMS キューおよびトピックは作成時に LDAP サーバーに自動的に登録されます。管理者は、LDAP に登録されたキューおよびトピックの別名を作成することもできます。LDAP に登録されたキューおよびトピックは、JNDI を介してキューまたはトピックの名前または別名を使用してルックアップできます。

共有サーバー・パラメータ

インスタンスを起動すると、Oracle Database は SHARED_SERVERS および DISPATCHERS 初期化パラメータの値に基づく共有サーバー・アーキテクチャの共有サーバー・プロセスおよびディスパッチャ・プロセスを作成します。ALTER SYSTEM 文で SHARED_SERVERS および DISPATCHERS パラメータを設定し、インスタンスの実行中に次の操作のいずれかを実行できます。

- 共有サーバー・プロセスの最小値を増やして、追加の共有サーバー・プロセスを作成します。
- 現行のコールが処理を終了した後、既存の共有サーバー・プロセスを終了します。
- 特定のプロトコルに対するディスパッチャ・プロセスをより多く作成します。プロトコル全体で、初期化パラメータ MAX_DISPATCHERS によって指定される数まで作成できます。
- 現行のユーザー・プロセスがインスタンスから切断した後、特定のプロトコルに対する既存のディスパッチャ・プロセスを終了します。

alter_system_reset_clause

この句によって、任意のインスタンスについて、インスタンスの起動に使用された spfile 内の初期化パラメータの設定を削除できます。SCOPE=MEMORY も SCOPE=BOTH も許可されません。SCOPE = SPFILE 句は必須ではありませんが、構文を明確にするために含まれています。この句は単一インスタンス環境で使用できますが、pfile ではなく spfile を使用してインスタンスが起動された場合のみです。

SID 句を使用すると、指定したインスタンスの spfile パラメータ設定を削除できます。RAC 環境以外では、インスタンスは 1 つのみであるため、この句を省略できます。RAC 環境では、この句を省略すると、デフォルトの SID = '*' が使用されます。つまり、*.*parameter* = *value* 形式のパラメータのすべての設定が削除されます。

参照 :

- Oracle Real Application Clusters 環境の各インスタンス用のパラメータ値設定の詳細は、『Oracle Real Application Clusters 管理およびデプロイメント・ガイド』を参照してください。
- ALTER SYSTEM 文の使用例は、11-67 ページの「ライセンス・パラメータの変更例 :」、11-65 ページの「クエリー・リライトを使用可能にする例 :」、11-66 ページの「リソース制限を使用可能にする例 :」、11-65 ページの「共有サーバー・パラメータ」、11-66 ページの「共有サーバーの設定の変更例 :」 を参照してください。

例

REDO ログの手動アーカイブの例 : 次の文は、SCN 9356083 の REDO ログ・エントリを含む REDO ログ・ファイル・グループを手動でアーカイブします。

```
ALTER SYSTEM ARCHIVE LOG CHANGE 9356083;
```

次の文は、メンバー 'diskl:log6.log' を含む REDO ログ・ファイル・グループを、'diska:[arch$]' という場所にあるアーカイブ REDO ログ・ファイルに手動でアーカイブします。

```
ALTER SYSTEM ARCHIVE LOG
    LOGFILE 'diskl:log6.log'
    TO 'diska:[arch$]';
```

クエリー・リライトを使用可能にする例 : 次の文は、クエリー・リライトが明示的に使用禁止にされていないすべてのマテリアライズド・ビューに対するすべてのセッションで、クエリー・リライトを使用可能にします。

```
ALTER SYSTEM SET QUERY_REWRITE_ENABLED = TRUE;
```

セッションを制限する例： たとえば、アプリケーションのメンテナンス中は、RESTRICTED SESSION システム権限が付与されているアプリケーション開発者のみがログインできるようにセッションを制限できます。このためには、次の文を発行します。

```
ALTER SYSTEM
   ENABLE RESTRICTED SESSION;
```

次に、ALTER SYSTEM 文の KILL SESSION 句を使用すると、既存のセッションをどれでも終了できます。

アプリケーションのメンテナンスが終了した後で、次の文を発行することによって、CREATE SESSION システム権限が付与されているユーザーもログインできるようになります。

```
ALTER SYSTEM
   DISABLE RESTRICTED SESSION;
```

ウォレットおよび暗号化キーの確立 次の文を使用すると、サーバー・ウォレットの情報をメモリーにロードして、データの透過的暗号化のマスター鍵を設定できます。

```
ALTER SYSTEM SET ENCRYPTION WALLET OPEN IDENTIFIED BY "welcome1";
ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY "welcome1";
```

これらの文では、セキュリティ・モジュールが初期化済で、パスワード welcome1 を使用してウォレットが作成済であることを前提としています。

共有プールの消去例： 共有プールを消去してから、パフォーマンス分析を開始します。共有プールを消去する場合、次の文を発行します。

```
ALTER SYSTEM FLUSH SHARED_POOL;
```

チェックポイントの発生例： 次の文は、チェックポイントを強制的に発生させます。

```
ALTER SYSTEM CHECKPOINT;
```

リソース制限を使用可能にする例： 次の ALTER SYSTEM 文は、リソース制限を動的に使用可能にします。

```
ALTER SYSTEM SET RESOURCE_LIMIT = TRUE;
```

共有サーバーの設定の変更例： 次の文は、共有サーバー・プロセスの最小数を 25 に変更します。

```
ALTER SYSTEM SET SHARED_SERVERS = 25;
```

共有サーバー・プロセスの数が 25 より少ない場合は追加作成されます。共有サーバー・プロセスが 25 より多く、25 あれば負荷を管理できる場合は、現行のコールによる処理が終了した時点で、25 を超える分の共有サーバー・プロセスは終了します。

次の文は、TCP/IP プロトコルのディスパッチャ・プロセス数を 5 に、ipc プロトコルのディスパッチャ・プロセス数を 10 に動的に変更します。

```
ALTER SYSTEM
   SET DISPATCHERS =
      '(INDEX=0)(PROTOCOL=TCP)(DISPATCHERS=5)',
      '(INDEX=1)(PROTOCOL=ipc)(DISPATCHERS=10)';
```

TCP のディスパッチャ・プロセスの数が 5 より少ない場合、ディスパッチャ・プロセスが新しく作成されます。5 より多い場合は、接続されているユーザーが接続を切断した後に、5 を超える分のディスパッチャ・プロセスは終了します。

ipc のディスパッチャ・プロセスの数が 10 より少ない場合、ディスパッチャ・プロセスが新しく作成されます。10 より多い場合は、接続されているユーザーが接続を切断した後に、10 を超える分のディスパッチャ・プロセスは終了します。

これ以外のプロトコル用として既存ディスパッチャがある場合、この文は、そのディスパッチャの数に影響しません。

ライセンス・パラメータの変更例： 次の文は、インスタンスにおけるセッションの最大数を 64 に、セッションの警告しきい値を 54 に動的に変更します。

```
ALTER SYSTEM
   SET LICENSE_MAX_SESSIONS = 64
   LICENSE_SESSIONS_WARNING = 54;
```

セッション数が 54 に達した場合、後続の各セッションの ALERT ファイルに警告メッセージが書き込まれます。RESTRICTED SESSION システム権限を持つユーザーも、後続セッションを開始する場合に、警告メッセージを受け取ります。

セッション数が 64 に達した場合、セッション数が再び 64 を下回るまでは、RESTRICTED SESSION システム権限を持つユーザー以外は新しいセッションを開始できません。

次の文は、インスタンスのセッションの最大数を動的に使用禁止にします。この文の実行後は、インスタンスのセッション数に制限がなくなります。

```
ALTER SYSTEM SET LICENSE_MAX_SESSIONS = 0;
```

次の文は、データベースのユーザー数の制限を 200 に動的に変更します。この文の実行後は、データベース・ユーザー数が 200 を超えることはありません。

```
ALTER SYSTEM SET LICENSE_MAX_USERS = 200;
```

ログ・スイッチの発生例： 書込み中のファイルの削除および名前の変更はできません。ただし、ログ・スイッチを強制的に発生させることによって、現行の REDO ログ・ファイル・グループまたはそのメンバーの 1 つを削除したり、その名前を変更できます。強制的に発生したログ・スイッチは、インスタンスの REDO ログ・スレッドのみに影響します。次の文は、ログ・スイッチを強制的に発生させます。

```
ALTER SYSTEM SWITCH LOGFILE;
```

分散リカバリを使用可能にする例： 次の文は、分散リカバリを使用可能にします。

```
ALTER SYSTEM ENABLE DISTRIBUTED RECOVERY;
```

デモンストレーションまたはテストの目的で、分散リカバリを使用禁止にする場合があります。次の文は、シングルプロセス・モードとマルチプロセス・モードの両方で分散リカバリを使用禁止にします。

```
ALTER SYSTEM DISABLE DISTRIBUTED RECOVERY;
```

デモンストレーションまたはテストが終了した場合、ALTER SYSTEM 文に ENABLE DISTRIBUTED RECOVERY 句を指定して実行すると、分散リカバリを再び使用可能にできます。

セッションの終了例： あるユーザーのセッションで、他のユーザーが必要とするリソースを使用している場合、そのユーザーのセッションを終了させる場合があります。このユーザーは、セッションが終了したことを示すエラー・メッセージを受け取ります。このユーザーは、新しいセッションを開始しないかぎり、このデータベースをコールできません。次の V$SESSION 動的パフォーマンス表のデータについて考えます。ここでは、ユーザー SYS と oe が両方ともセッションを開いているものとします。

```
SELECT sid, serial#, username
   FROM V$SESSION;

       SID    SERIAL# USERNAME
---------- ---------- ------------------------------
        29         85 SYS
        33          1
        35          8
```

```
       39         23 OE
       40          1
. . .
```

次の文は、V$SESSION の SID 値と SERIAL# 値を使用して、ユーザー scott のセッションを終了します。

```
ALTER SYSTEM KILL SESSION '39, 23';
```

セッションの切断例： 次の文は、V$SESSION の SID と SERIAL# の値を使用して、ユーザー scott のセッションを切断します。

```
ALTER SYSTEM DISCONNECT SESSION '13, 8' POST_TRANSACTION;
```

12

SQL 文 : ALTER TABLE ～ ALTER TABLESPACE

この章では、次の SQL 文について説明します。

- ALTER TABLE
- ALTER TABLESPACE

ALTER TABLE

用途

ALTER TABLE を使用すると、非パーティション表、パーティション表、表パーティションおよび表サブパーティションの定義を変更できます。オブジェクト表またはオブジェクト列を含むリレーショナル表の場合は、ALTER TABLE を使用して型が変更された後に、表を参照する型の最新の定義に変換します。

参照：

- 表の作成については、16-6 ページの「CREATE TABLE」を参照してください。
- Oracle Text とともに使用する ALTER TABLE については、『Oracle Text リファレンス』を参照してください。

前提条件

表が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、その表に対する ALTER オブジェクト権限または ALTER ANY TABLE システム権限が必要です。

パーティション化操作におけるその他の前提条件　表の所有者でない場合、*drop_table_partition* または *truncate_table_partition* 句を使用するには、DROP ANY TABLE 権限が必要です。

add_table_partition、*modify_table_partition*、*move_table_partition* および *split_table_partition* 句を使用する場合、領域を確保する表領域に領域割当て制限が必要です。

制約およびトリガーにおけるその他の前提条件　一意キー制約または主キー制約を有効にする場合は、表に索引を作成するための権限が必要です。Oracle Database では、表を含むスキーマにおいて、一意キーまたは主キーの列に索引を作成するため、この権限が必要になります。

トリガーを使用可能または使用禁止にする場合、トリガーが自分のスキーマ内にある必要があります。自分のスキーマにない場合は、ALTER ANY TRIGGER システム権限が必要です。

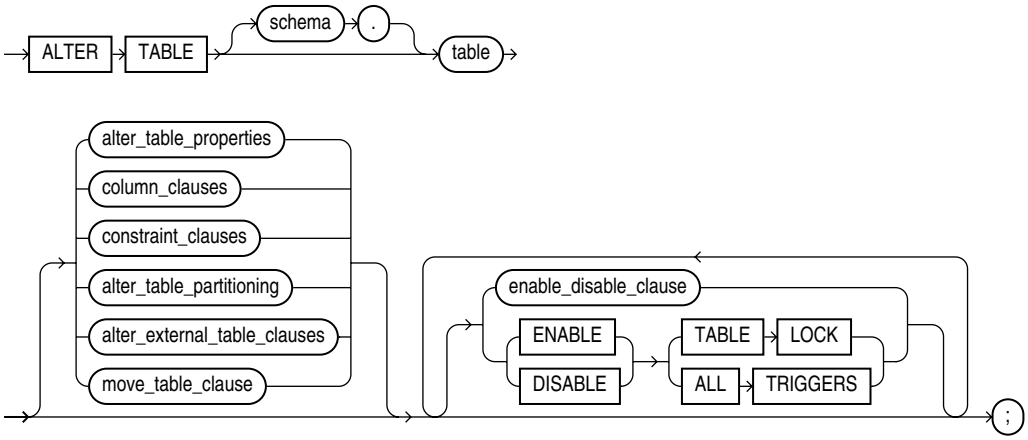
オブジェクト型を使用する場合のその他の前提条件　表を変更するときに列定義でオブジェクト型を使用する場合、そのオブジェクトが、変更する表と同じスキーマに属している必要があります。または、EXECUTE ANY TYPE システム権限またはそのオブジェクト型に対する EXECUTE スキーマ・オブジェクト権限が必要です。

フラッシュバック・データ・アーカイブ操作におけるその他の前提条件　*flashback_archive_clause* を使用して表の履歴追跡を有効にするには、履歴データが含まれるフラッシュバック・データ・アーカイブに対する FLASHBACK ARCHIVE オブジェクト権限が必要です。*flashback_archive_clause* を使用して表の履歴追跡を無効にするには、FLASHBACK ARCHIVE ADMINSTER システム権限を持っているか、または SYSDBA としてログインしている必要があります。

参照：　索引を作成する場合に必要な権限については、14-50 ページの「CREATE INDEX」を参照してください。

構文

***alter_table*::=**



注意： *table* の後に句を指定する必要があります。必須の句はありませんが、1 つ以上の句を指定する必要があります。

ALTER TABLE 構文のグループは、次のとおりです。

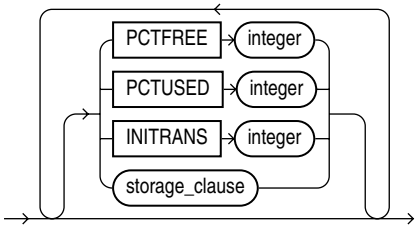
- **alter_table_properties::=**（12-4 ページ）
- **column_clauses::=**（12-8 ページ）
- **constraint_clauses::=**（12-10 ページ）
- **alter_table_partitioning::=**（12-18 ページ）
- **alter_external_table::=**（12-17 ページ）
- **move_table_clause::=**（12-30 ページ）
- **enable_disable_clause::=**（12-30 ページ）

各句の後には、そのコンポーネントの副次句の参照先が記載されています。

alter_table_properties::=

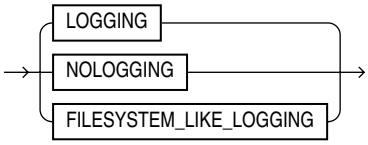
(12-4 ページの physical_attributes_clause::=、8-34 ページの logging_clause::=、
12-5 ページの table_compression::=、12-5 ページの supplemental_table_logging::=、
12-5 ページの allocate_extent_clause::=、12-6 ページの deallocate_unused_clause::= 、
12-6 ページの upgrade_table_clause::=、12-6 ページの records_per_block_clause::=、
12-6 ページの parallel_clause::=、12-6 ページの row_movement_clause::=、
12-30 ページの flashback_archive_clause::=、12-6 ページの shrink_clause::=、
12-6 ページの alter_iot_clauses::=、12-43 ページの alter_XMLSchemas_clause を参照)

physical_attributes_clause::=

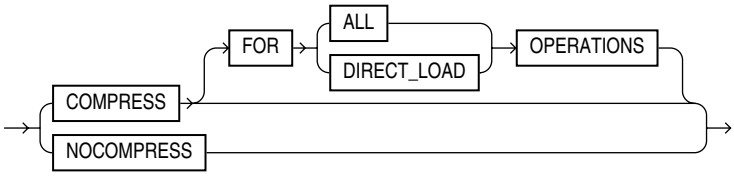


(8-44 ページの storage_clause::= を参照)

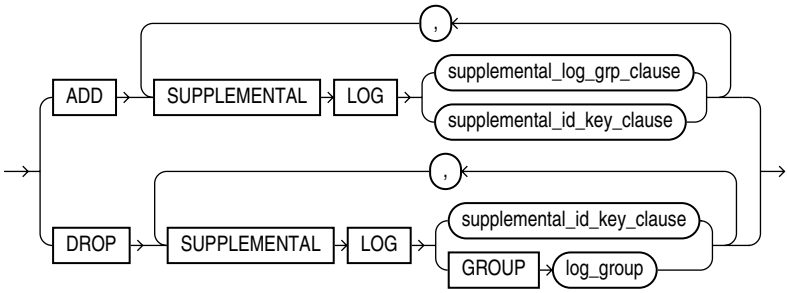
logging_clause::=



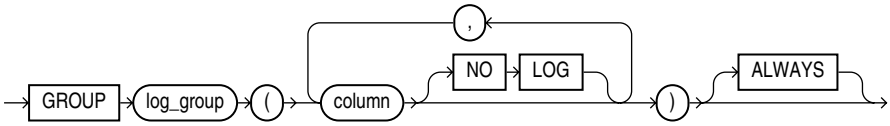
table_compression::=



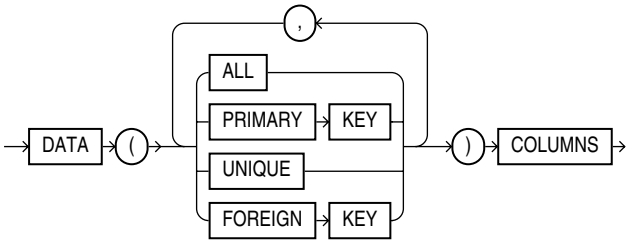
supplemental_table_logging::=



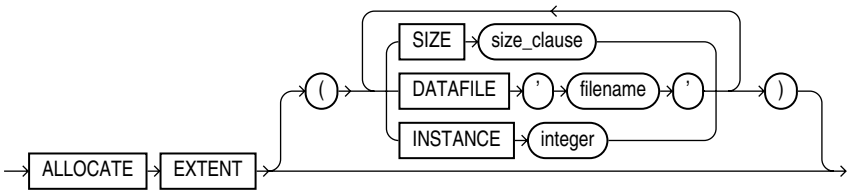
***supplemental_log_grp_clause*::=**



***supplemental_id_key_clause*::=**

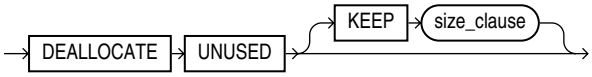


allocate_extent_clause::=



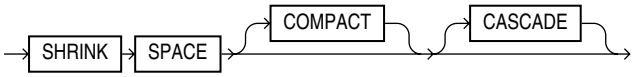
(8-42 ページの size_clause::= を参照)

deallocate_unused_clause::=

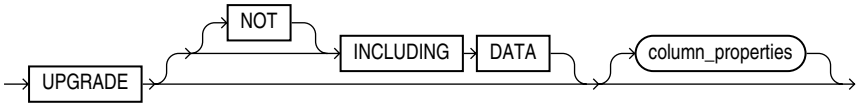


(8-42 ページの size_clause::= を参照)

shrink_clause::=

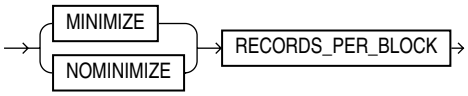


upgrade_table_clause::=

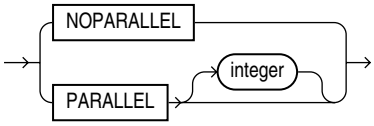


(12-11 ページの column_properties::= を参照)

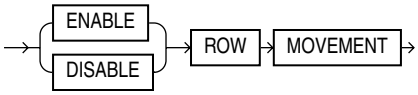
records_per_block_clause::=



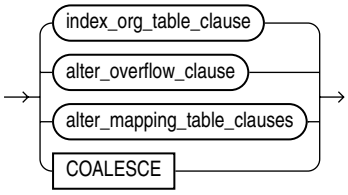
***parallel_clause*::=**



row_movement_clause::=

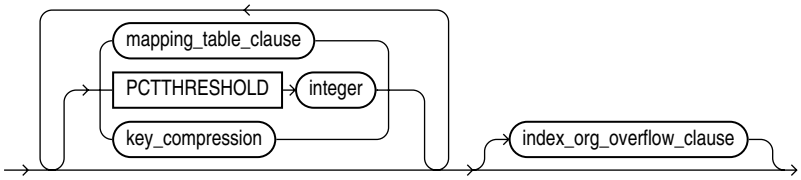


alter_iot_clauses::=

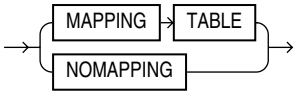


(12-8 ページの alter_overflow_clause::=、12-8 ページの alter_mapping_table_clauses::= を参照)

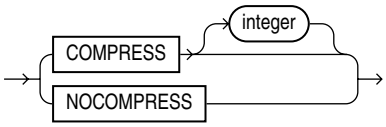
index_org_table_clause::=



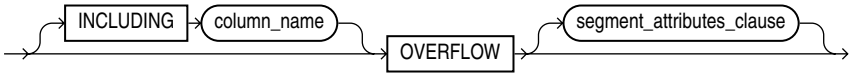
***mapping_table_clauses*::=**



***key_compression*::=**

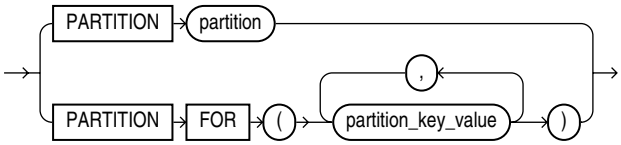


***index_org_overflow_clause*::=**

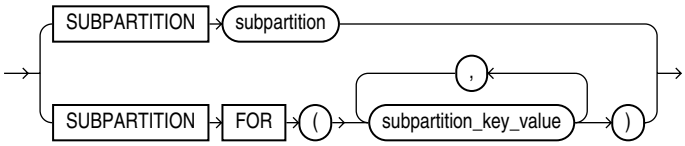


（12-7 ページの segment_attributes_clause::= を参照）

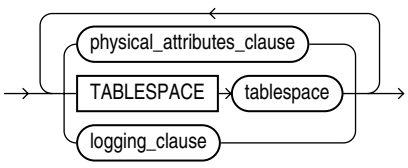
***partition_extended_name*::=**



***subpartition_extended_name*::=**

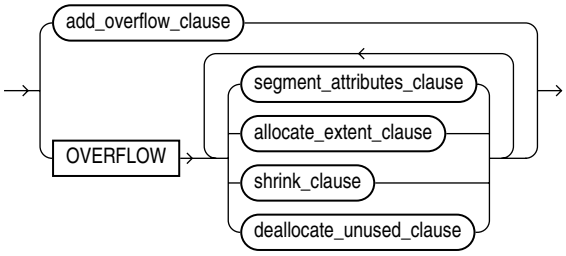


***segment_attributes_clause*::=**



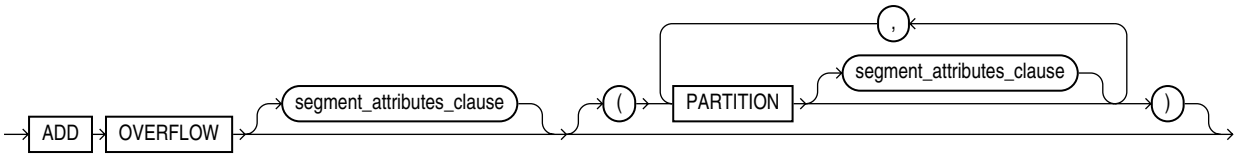
（12-4 ページの physical_attributes_clause::=、8-34 ページの logging_clause::= を参照）

alter_overflow_clause::=



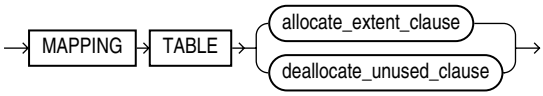
（12-7 ページの segment_attributes_clause::=、12-5 ページの allocate_extent_clause::=、12-6 ページの shrink_clause::=、12-6 ページの deallocate_unused_clause::= を参照）

add_overflow_clause::=



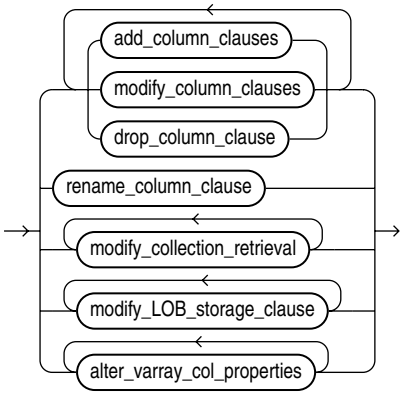
（12-7 ページの segment_attributes_clause::= を参照）

alter_mapping_table_clauses::=



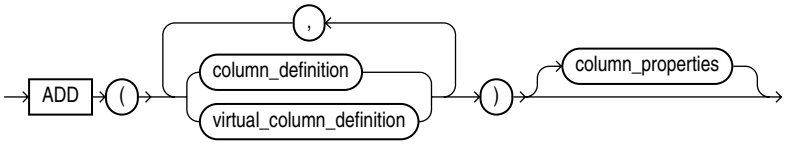
（12-5 ページの allocate_extent_clause::=、12-6 ページの deallocate_unused_clause::= を参照）

column_clauses::=



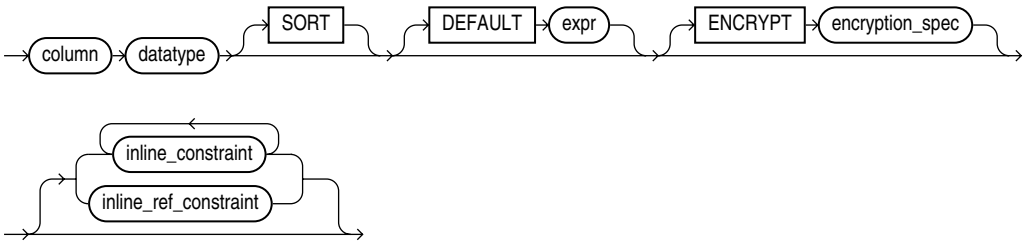
（12-9 ページの add_column_clause::=、12-9 ページの modify_column_clauses::=、12-10 ページの drop_column_clause::=、12-10 ページの rename_column_clause::=、12-10 ページの modify_collection_retrieval::=、12-14 ページの modify_LOB_storage_clause::=、12-16 ページの alter_varray_col_properties::=、12-10 ページの encryption_spec::= を参照）

add_column_clause::=



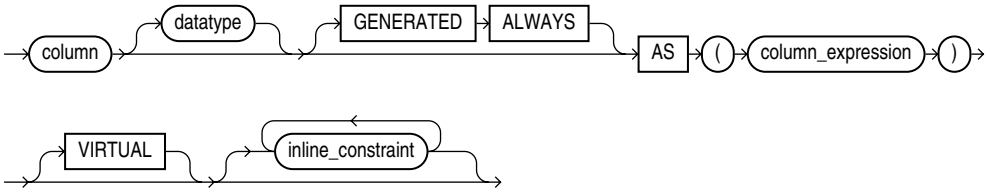
（12-9 ページの column_definition::=、12-11 ページの column_properties::= を参照）

column_definition::=

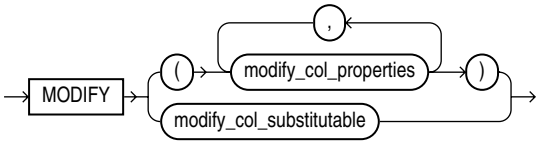


（12-10 ページの encryption_spec::=、8-5 ページの constraint::= の「*inline_constraint*」および「*inline_ref_constraint* を参照）

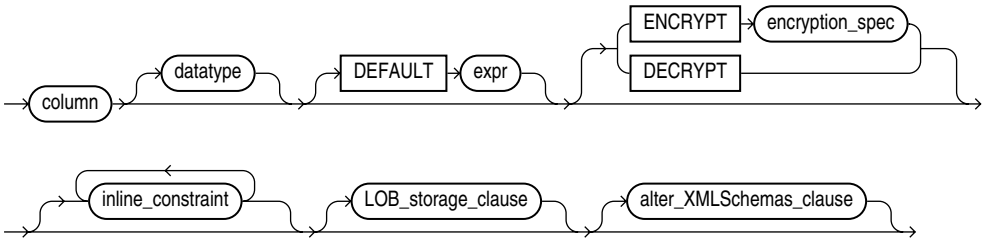
***virtual_column_definition*::=**



modify_column_clauses::=

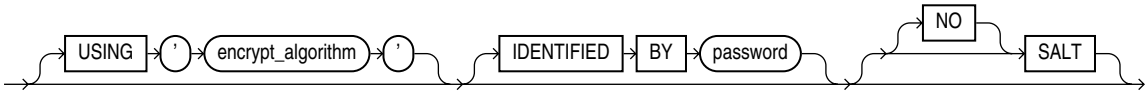


modify_col_properties::=

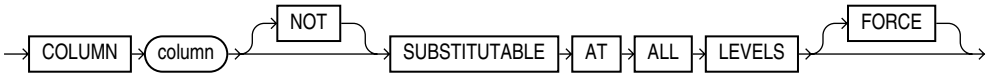


（12-10 ページの *encryption_spec::=*、8-5 ページの constraint::= の「*inline_constraint*」
および 12-13 ページの LOB_storage_clause::= を参照）

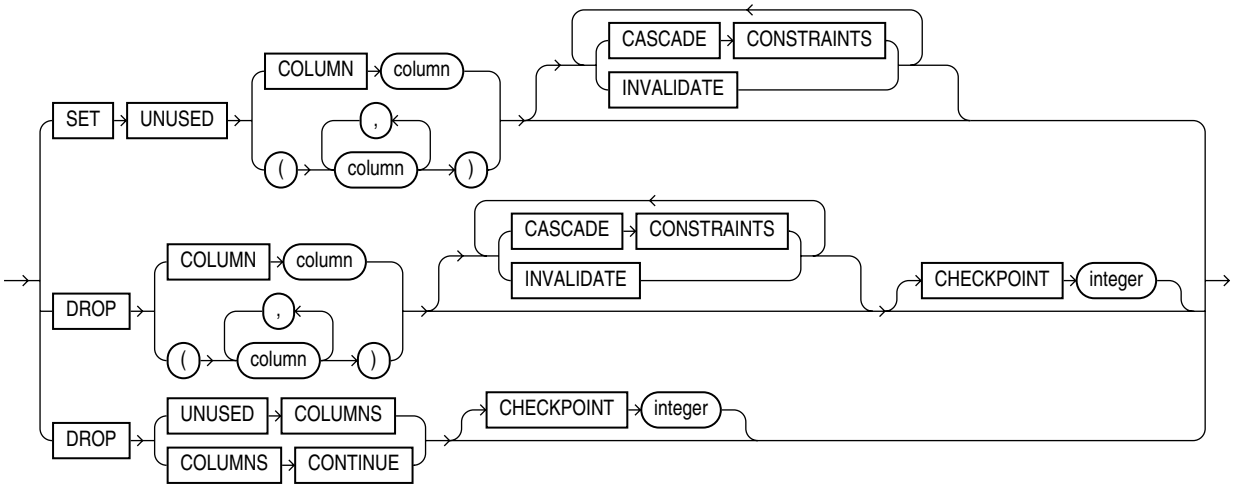
***encryption_spec*::=**



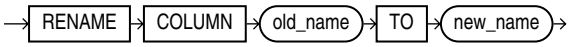
modify_col_substitutable::=



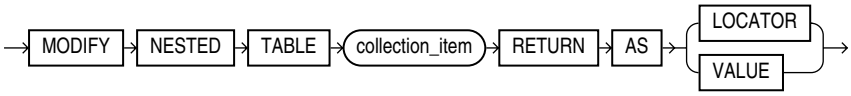
drop_column_clause::=



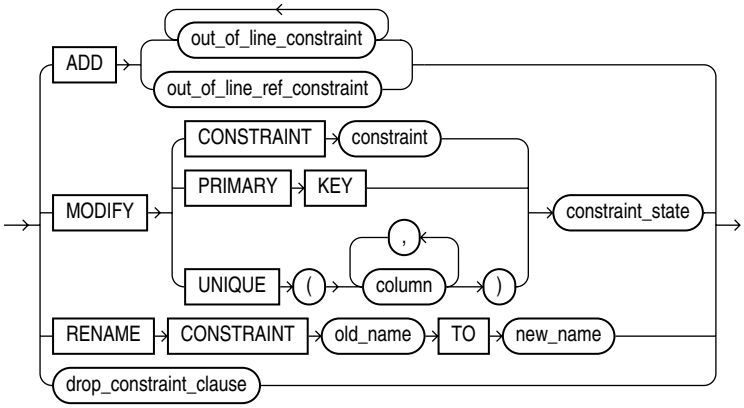
rename_column_clause::=



modify_collection_retrieval::=

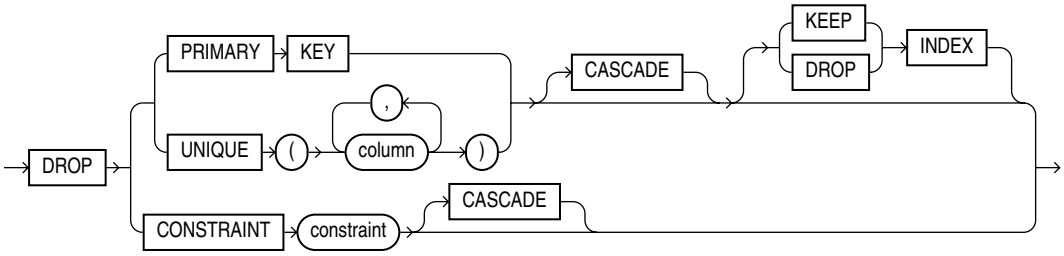


constraint_clauses::=

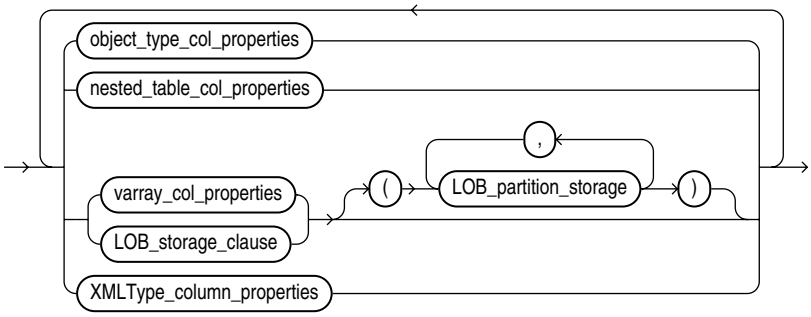


(8-5 ページの constraint::= の「*constraint_state*:」を参照)

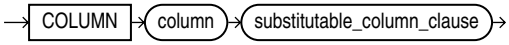
drop_constraint_clause::=



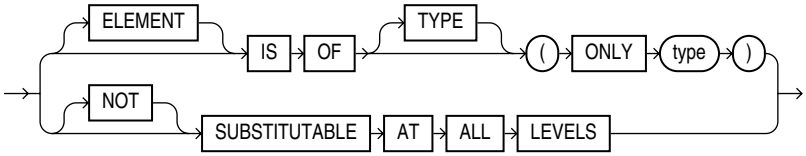
column_properties::=



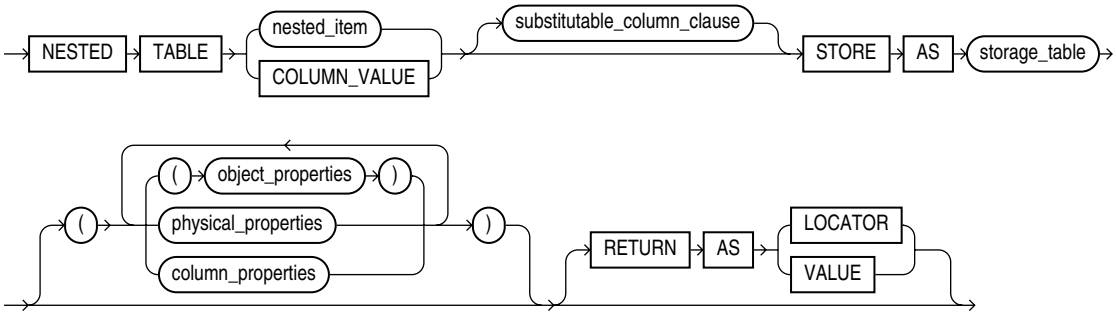
object_type_col_properties::=



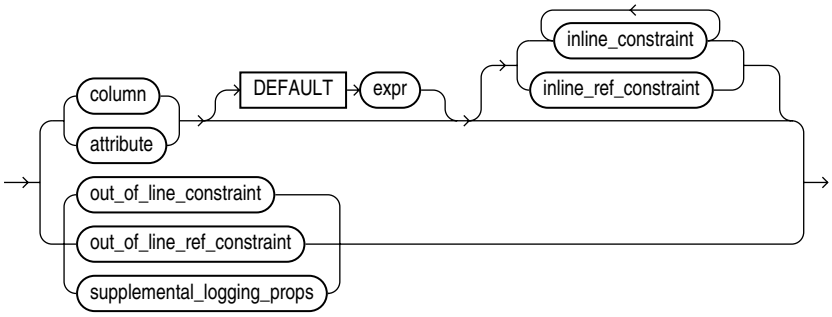
***substitutable_column_clause*::=**



nested_table_col_properties::=

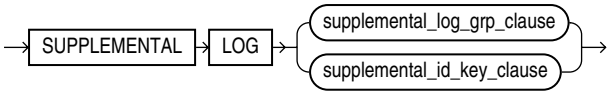


***object_properties*::=**



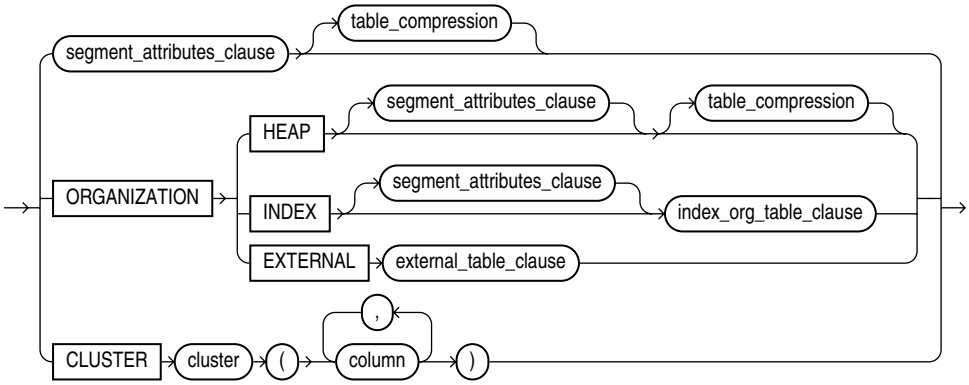
(8-5 ページの constraint::= の「*inline_constraint*」、「*inline_ref_constraint*」、「*out_of_line_constraint*」および「*out_of_line_ref_constraint*」を参照)

***supplemental_logging_props*::=**



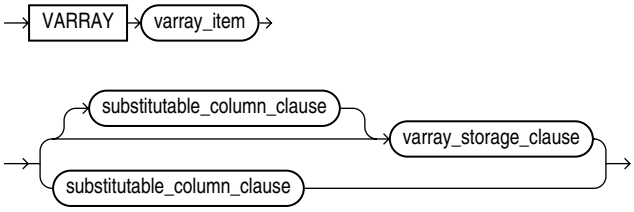
(12-5 ページの supplemental_log_grp_clause::=、12-5 ページの supplemental_id_key_clause::= を参照)

***physical_properties*::=**



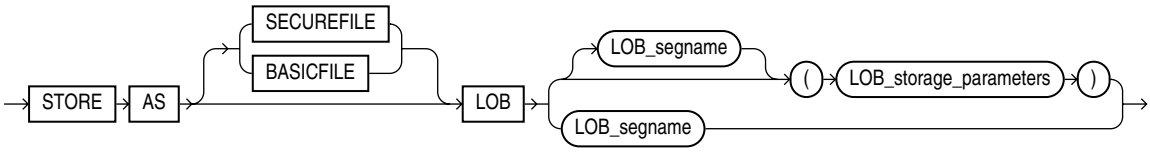
(12-7 ページの segment_attributes_clause::=、12-7 ページの index_org_table_clause::=、12-18 ページの external_data_properties::= を参照)

varray_col_properties::=



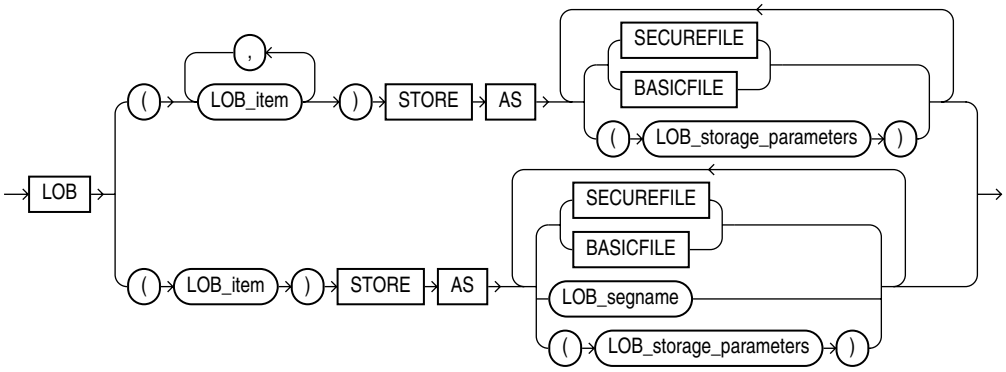
（12-11 ページの substitutable_column_clause::=、12-13 ページの varray_storage_clause::= を参照）

***varray_storage_clause*::=**



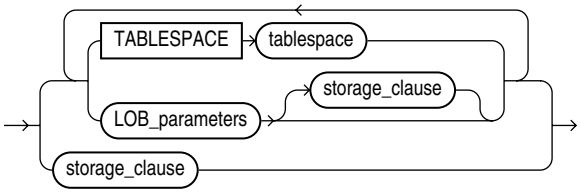
（12-14 ページの LOB_parameters::= を参照）

LOB_storage_clause::=



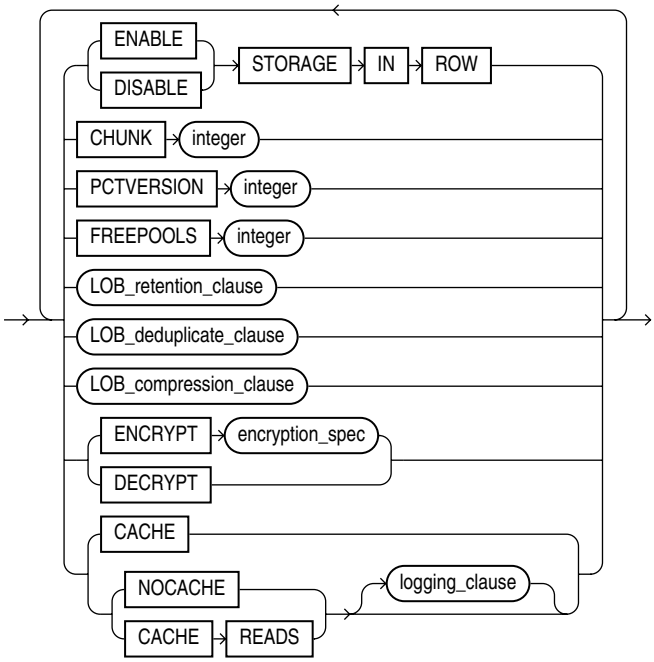
（12-14 ページの LOB_storage_parameters::= を参照）

***LOB_storage_parameters*::=**



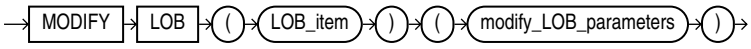
（12-14 ページの LOB_parameters::=、8-44 ページの storage_clause::= を参照）

***LOB_parameters*::=**

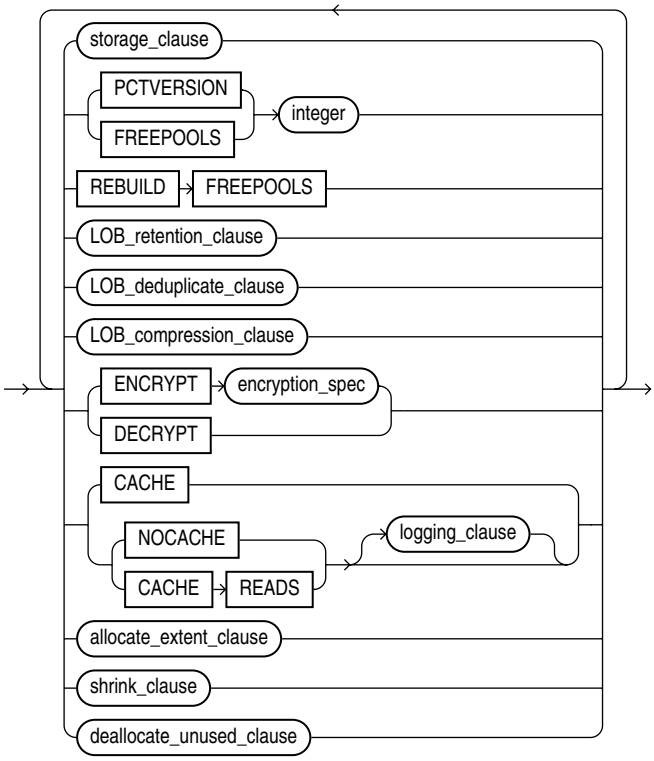


（12-15 ページの LOB_retention_clause::=、12-15 ページの LOB_deduplicate_clause::=、12-15 ページの LOB_compression_clause::=、12-10 ページの encryption_spec::=、8-34 ページの logging_clause::= を参照）

modify_LOB_storage_clause::=

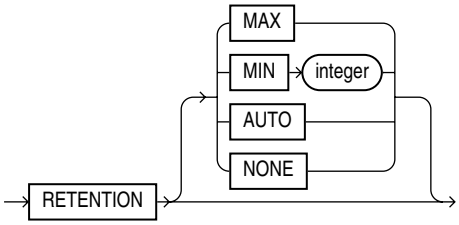


***modify_LOB_parameters*::=**

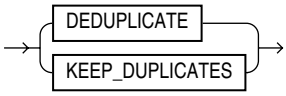


（8-44 ページの storage_clause::=、12-15 ページの LOB_retention_clause::=、
12-15 ページの LOB_compression_clause::=、12-10 ページの encryption_spec::=、
8-34 ページの logging_clause::=、12-5 ページの allocate_extent_clause::=、
12-6 ページの shrink_clause::=、12-6 ページの deallocate_unused_clause::= を参照）

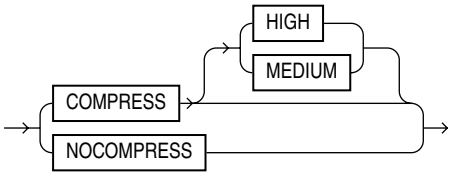
LOB_retention_clause::=



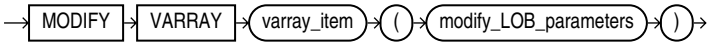
LOB_deduplicate_clause::=



LOB_compression_clause::=

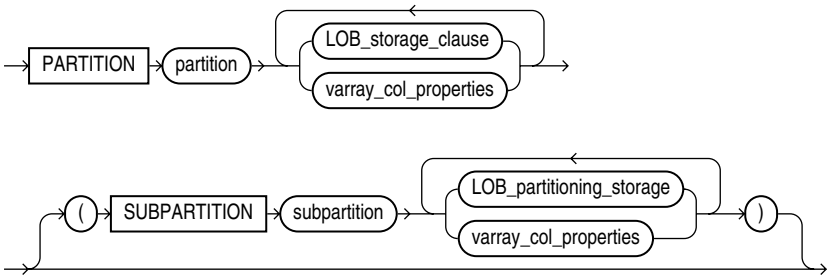


alter_varray_col_properties::=



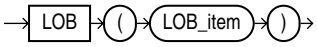
(12-15 ページの modify_LOB_parameters::= を参照)

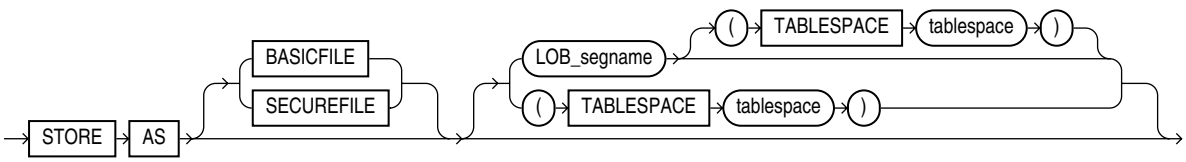
LOB_partition_storage::=



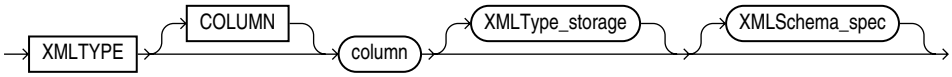
(12-13 ページの LOB_storage_clause::=、12-13 ページの varray_col_properties::=、
12-16 ページの LOB_partitioning_storage::= を参照)

LOB_partitioning_storage::=

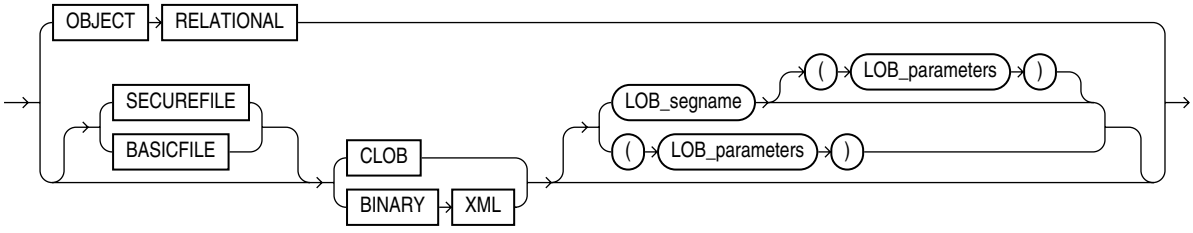




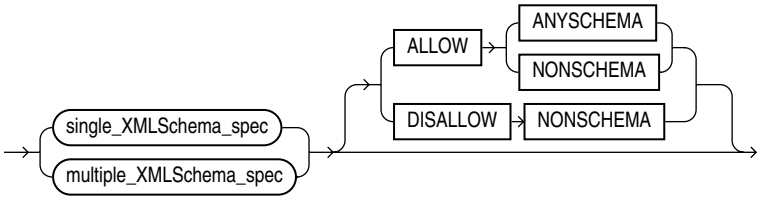
XMLType_column_properties::=



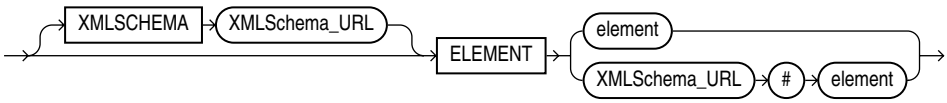
XMLType_storage::=



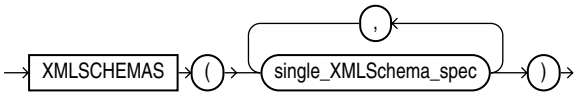
***XMLSchema_spec*::=**



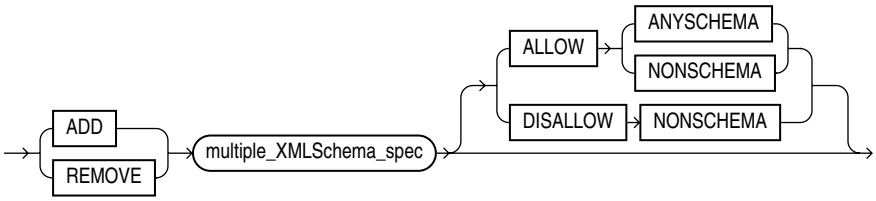
***single_XMLSchema_spec*::=**



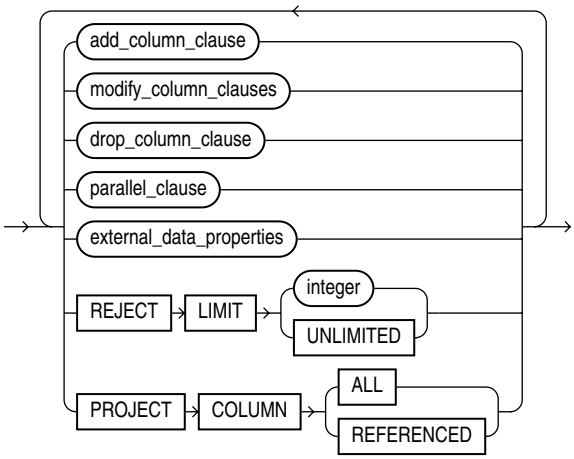
***multiple_XMLSchema_spec*::=**



***alter_XMLSchemas_clause*::=**

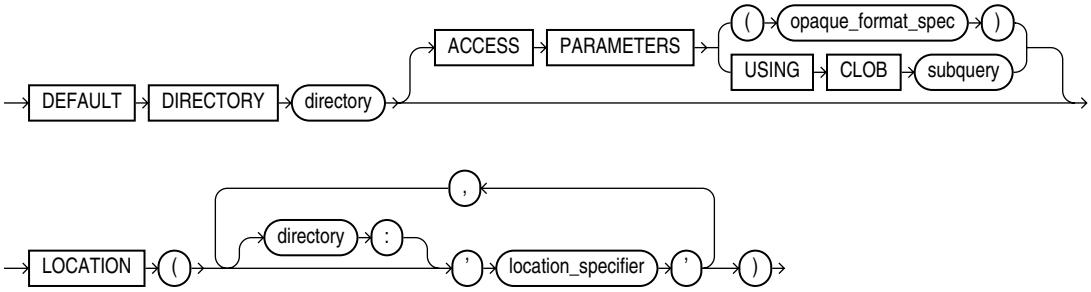


alter_external_table::=

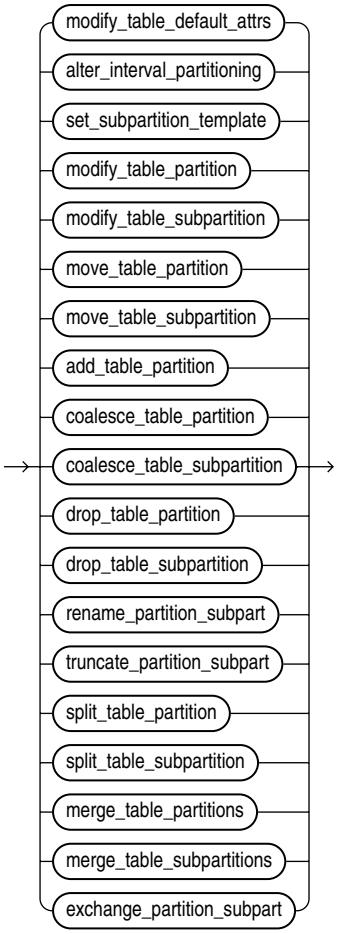


（12-9 ページの add_column_clause::=、12-9 ページの modify_column_clauses::=、
12-10 ページの drop_column_clause::=、12-11 ページの drop_constraint_clause::=、
12-6 ページの parallel_clause::= を参照）

***external_data_properties*::=**

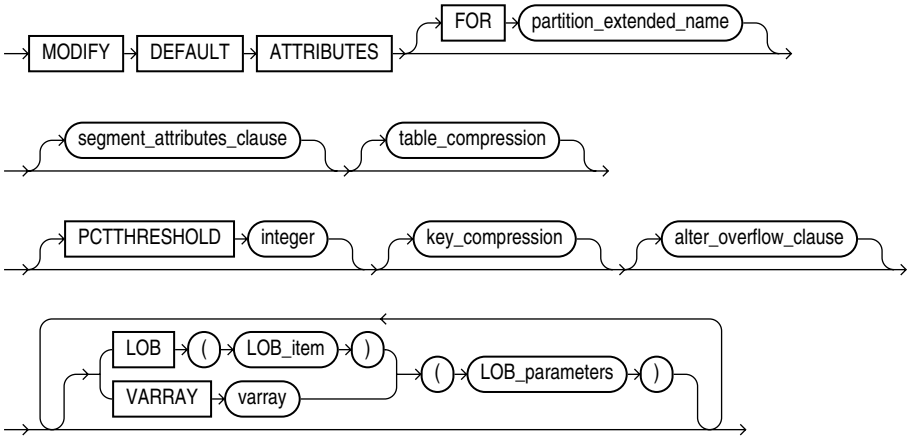


alter_table_partitioning::=



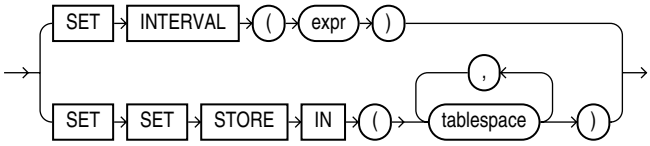
(12-19 ページの modify_table_default_attrs::=、12-19 ページの alter_interval_partitioning::=、
12-19 ページの set_subpartition_template::=、12-19 ページの modify_table_partition::=、
12-21 ページの modify_table_subpartition::=、12-21 ページの move_table_partition::=、
12-21 ページの move_table_subpartition::=、12-21 ページの add_table_partition::=、
12-23 ページの coalesce_table_partition::=、12-23 ページの coalesce_table_subpartition::=、
12-23 ページの drop_table_partition::=、12-23 ページの drop_table_subpartition::=、
12-23 ページの rename_partition_subpart::=、12-24 ページの truncate_partition_subpart::=、
12-24 ページの split_table_partition::=、12-24 ページの split_table_subpartition::=、
12-25 ページの merge_table_partitions::=、12-25 ページの merge_table_subpartitions::=、
12-25 ページの exchange_partition_subpart::= を参照)

modify_table_default_attrs::=

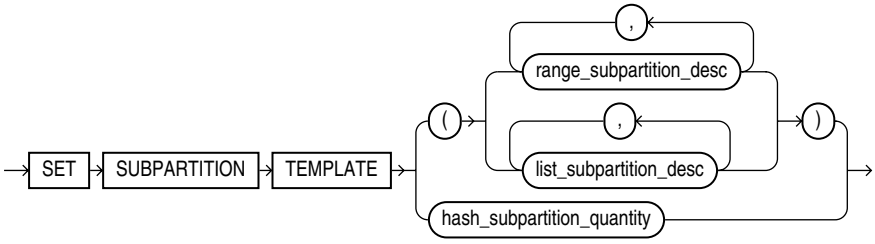


（12-7 ページの partition_extended_name::=、12-7 ページの segment_attributes_clause::=、12-5 ページの table_compression::=、12-7 ページの key_compression::=、12-8 ページの alter_overflow_clause::=、12-14 ページの LOB_parameters::= を参照）

alter_interval_partitioning::=

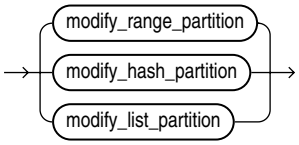


set_subpartition_template::=



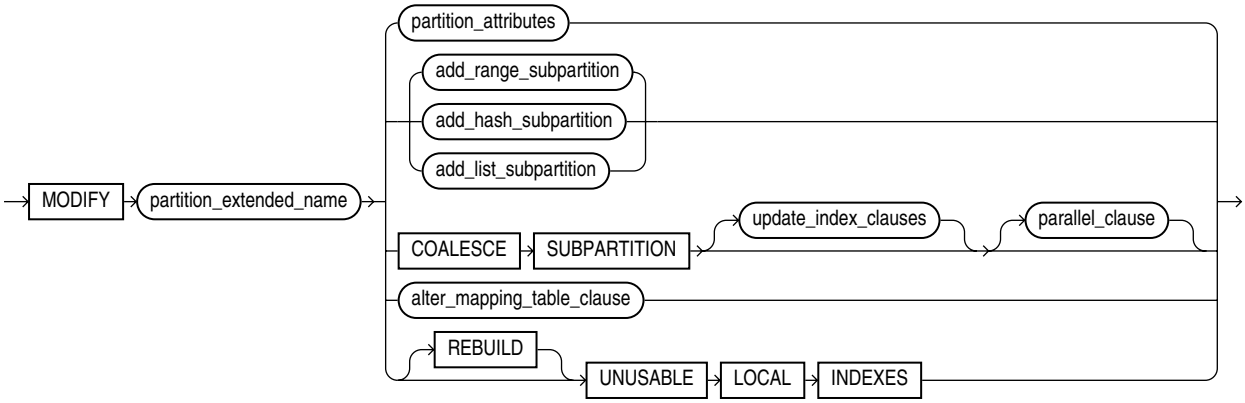
（12-27 ページの range_subpartition_desc::=、12-27 ページの list_subpartition_desc::= を参照）

modify_table_partition::=



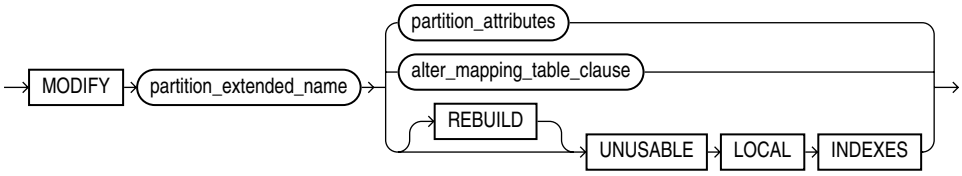
（12-20 ページの modify_range_partition::=、12-20 ページの modify_hash_partition::=、12-20 ページの modify_list_partition::= を参照）

modify_range_partition::=



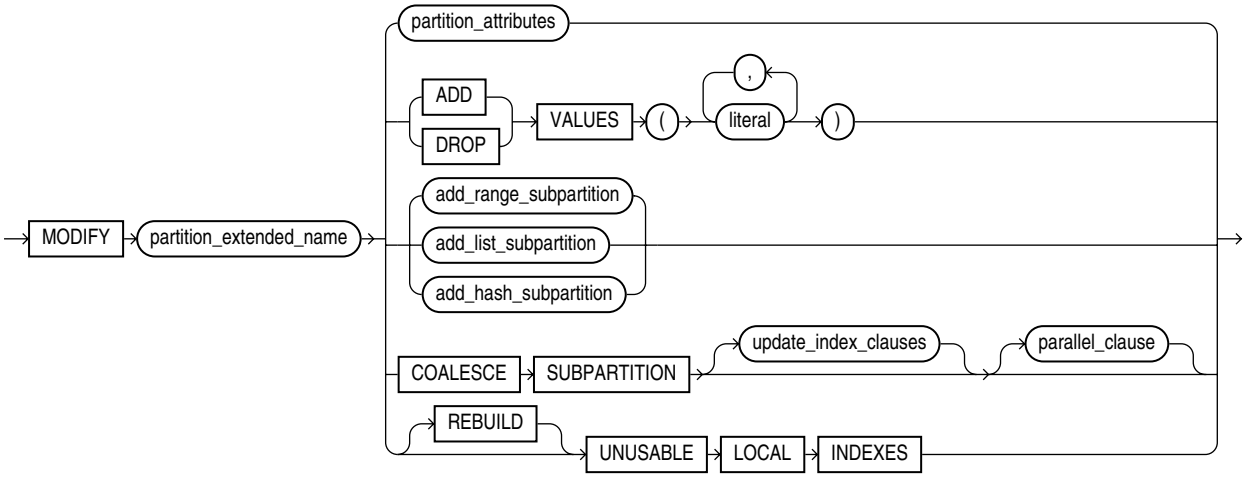
(12-7 ページの partition_extended_name::=、12-28 ページの partition_attributes::=、
12-22 ページの add_range_subpartition::=、12-22 ページの add_hash_subpartition::=、
12-23 ページの add_list_subpartition::=、12-29 ページの update_index_clauses::=、
12-6 ページの parallel_clause::=、12-8 ページの alter_mapping_table_clauses::= を参照)

modify_hash_partition::=



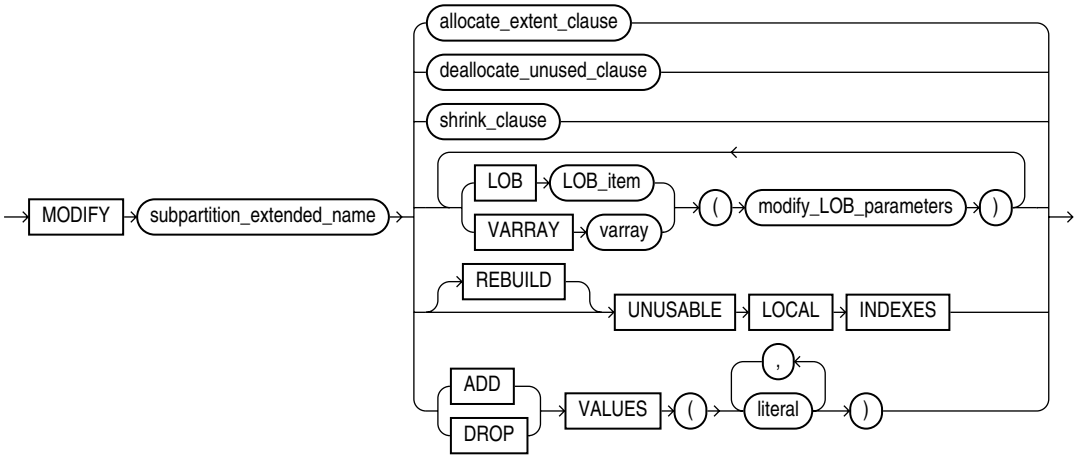
(12-7 ページの partition_extended_name::=、12-28 ページの partition_attributes::=、
12-8 ページの alter_mapping_table_clauses::= を参照)

***modify_list_partition*::=**



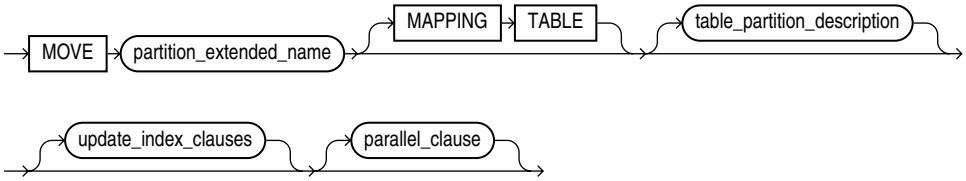
(12-7 ページの partition_extended_name::=、12-28 ページの partition_attributes::=、
12-22 ページの add_range_subpartition::=、12-23 ページの add_list_subpartition::=、
12-22 ページの add_hash_subpartition::= を参照)

modify_table_subpartition::=



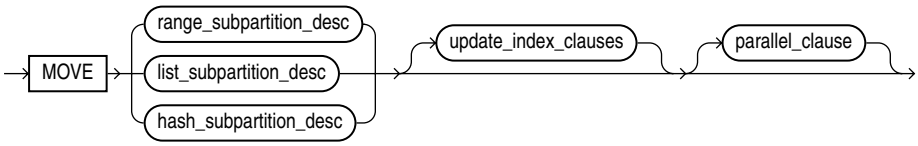
（12-7 ページの subpartition_extended_name::=、12-5 ページの allocate_extent_clause::=、12-6 ページの deallocate_unused_clause::=、12-6 ページの shrink_clause::=、12-15 ページの modify_LOB_parameters::= を参照）

move_table_partition::=



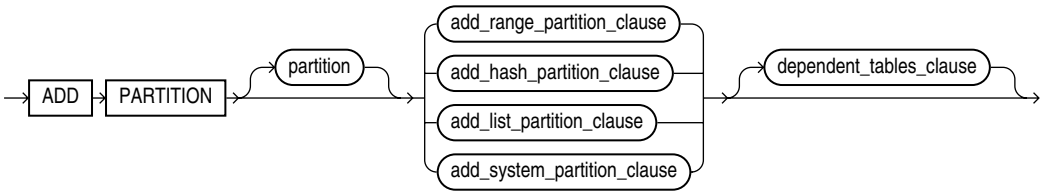
（12-7 ページの partition_extended_name::=、12-26 ページの table_partition_description::=、12-29 ページの update_index_clauses::=、12-6 ページの parallel_clause::= を参照）

move_table_subpartition::=



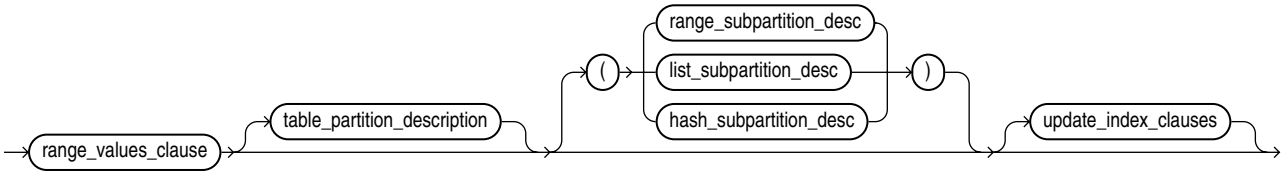
（12-27 ページの range_subpartition_desc::=、12-27 ページの list_subpartition_desc::=、12-27 ページの hash_subparts_by_quantity::=、12-29 ページの update_index_clauses::=、12-6 ページの parallel_clause::= を参照）

add_table_partition::=



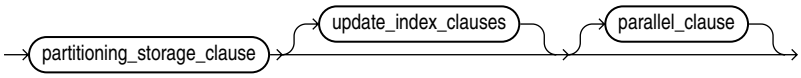
（12-22 ページの add_range_partition_clause::=、12-22 ページの add_hash_partition_clause::=、12-22 ページの add_list_partition_clause::=、12-22 ページの add_system_partition_clause::=、12-23 ページの dependent_tables_clause:= を参照）

add_range_partition_clause::=



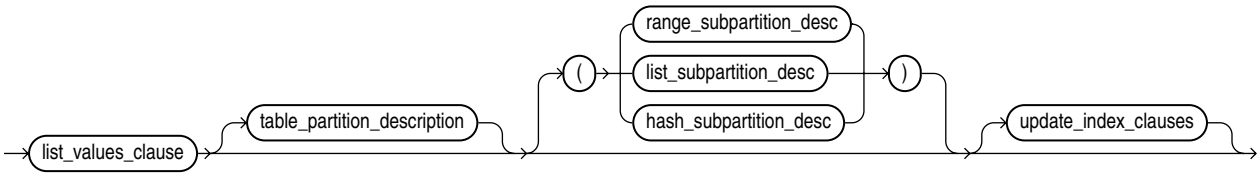
(12-26 ページの range_values_clause::=、12-26 ページの table_partition_description::=、12-27 ページの range_subpartition_desc::=、12-27 ページの list_subpartition_desc::=、12-27 ページの hash_subparts_by_quantity::=、12-29 ページの update_index_clauses::= を参照)

add_hash_partition_clause::=



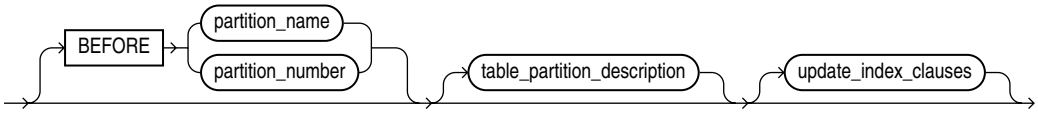
(12-28 ページの partitioning_storage_clause::=、12-29 ページの update_index_clauses::=、12-6 ページの parallel_clause::= を参照)

add_list_partition_clause::=



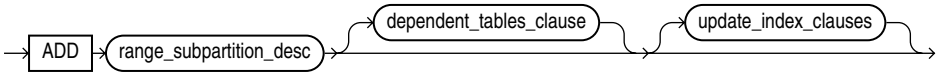
(12-26 ページの list_values_clause::=、12-26 ページの table_partition_description::=、12-27 ページの range_subpartition_desc::=、12-27 ページの list_subpartition_desc::=、12-27 ページの hash_subparts_by_quantity::=、12-29 ページの update_index_clauses::= を参照)

add_system_partition_clause::=



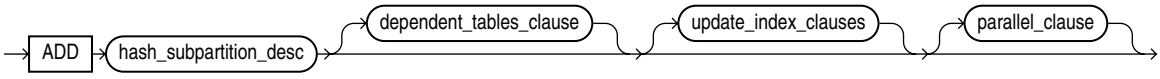
(12-26 ページの table_partition_description::=、12-29 ページの update_index_clauses::= を参照)

***add_range_subpartition*::=**



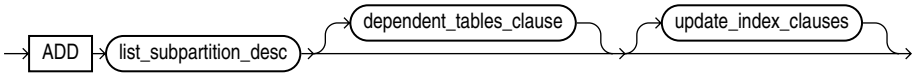
(12-27 ページの range_subpartition_desc::=、12-29 ページの update_index_clauses::=、12-6 ページの parallel_clause::= を参照)

add_hash_subpartition::=



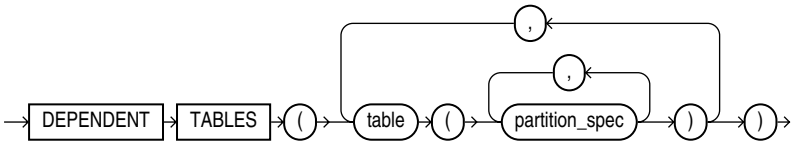
(12-27 ページの hash_subparts_by_quantity::=、12-29 ページの update_index_clauses::=、12-6 ページの parallel_clause::= を参照)

add_list_subpartition::=



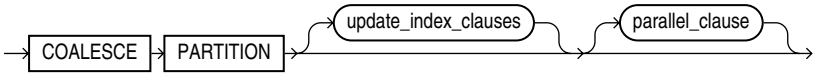
（12-27 ページの list_subpartition_desc::=、12-29 ページの update_index_clauses::= を参照）

dependent_tables_clause:=



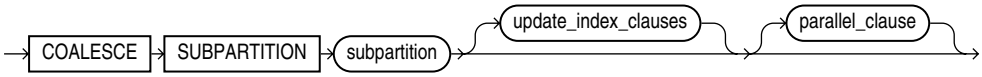
（12-28 ページの partition_spec::= を参照）

coalesce_table_partition::=



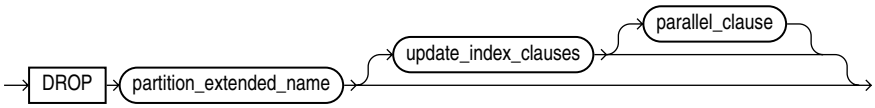
（12-29 ページの update_index_clauses::=、12-6 ページの parallel_clause::= を参照）

***coalesce_table_subpartition*::=**



（12-29 ページの update_index_clauses::=、12-6 ページの parallel_clause::= を参照）

drop_table_partition::=

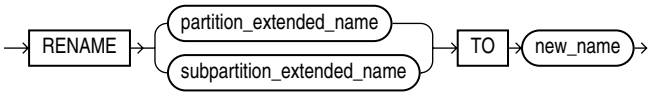


（12-7 ページの partition_extended_name::=、12-29 ページの update_index_clauses::=、12-6 ページの parallel_clause::= を参照）

drop_table_subpartition::=

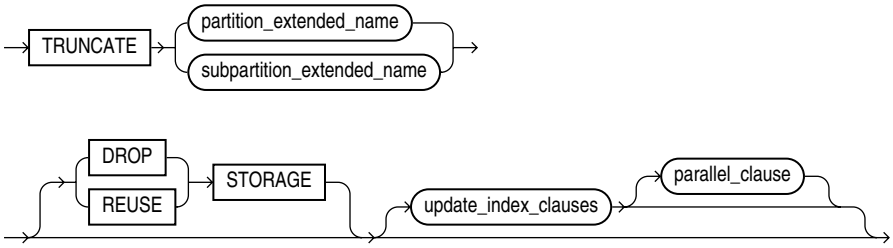
（12-7 ページの subpartition_extended_name::=、12-29 ページの update_index_clauses::=、12-6 ページの parallel_clause::= を参照）

rename_partition_subpart::=



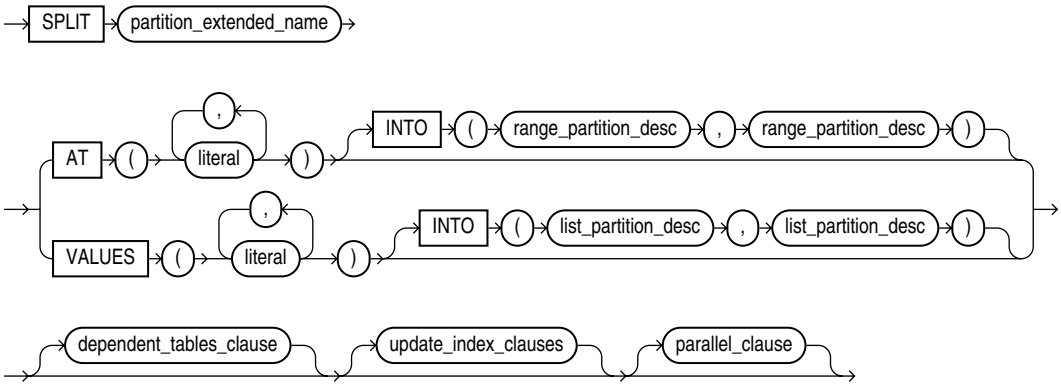
（12-7 ページの partition_extended_name::=、12-7 ページの subpartition_extended_name::= を参照）

truncate_partition_subpart::=



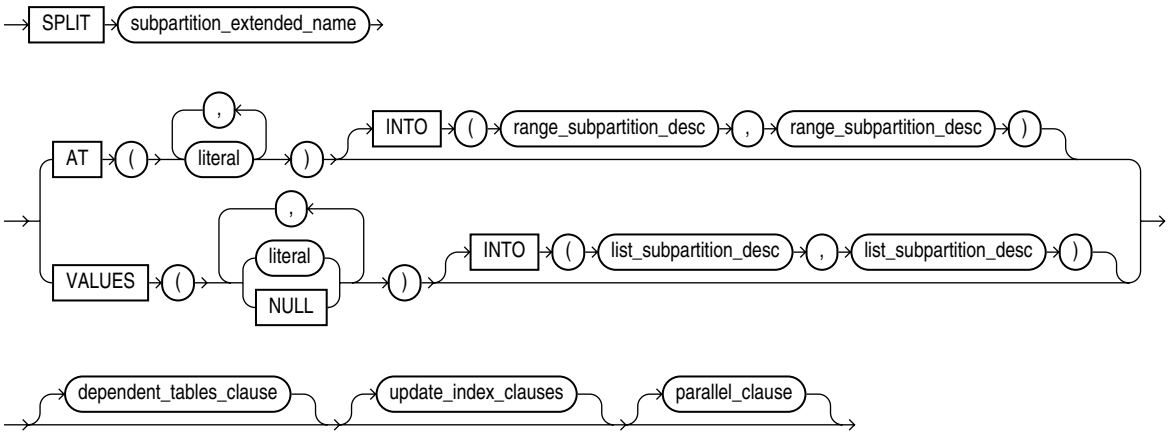
（12-7 ページの partition_extended_name::=、12-7 ページの subpartition_extended_name::=、12-29 ページの update_index_clauses::=、12-6 ページの parallel_clause::= を参照）

split_table_partition::=



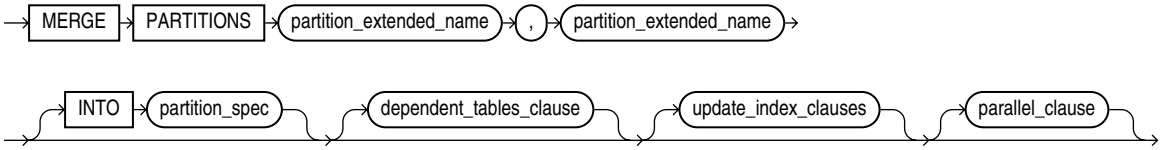
（12-7 ページの partition_extended_name::=、12-26 ページの range_partition_desc::=、12-27 ページの list_partition_desc::=、12-23 ページの dependent_tables_clause:=、12-29 ページの update_index_clauses::=、12-6 ページの parallel_clause::= を参照）

split_table_subpartition::=



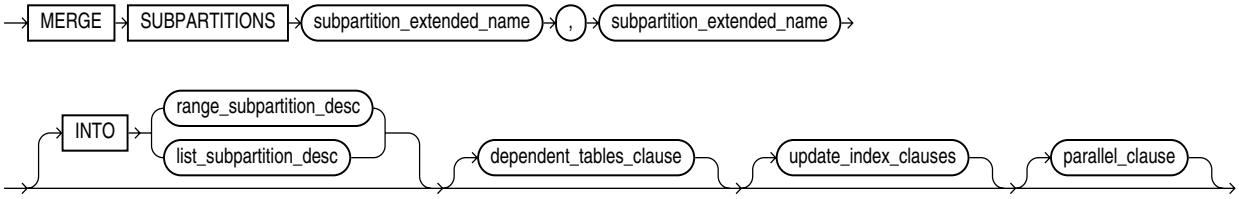
（12-7 ページの subpartition_extended_name::=、12-27 ページの range_subpartition_desc::=、12-27 ページの list_subpartition_desc::=、12-29 ページの update_index_clauses::=、12-6 ページの parallel_clause::= を参照）

merge_table_partitions::=



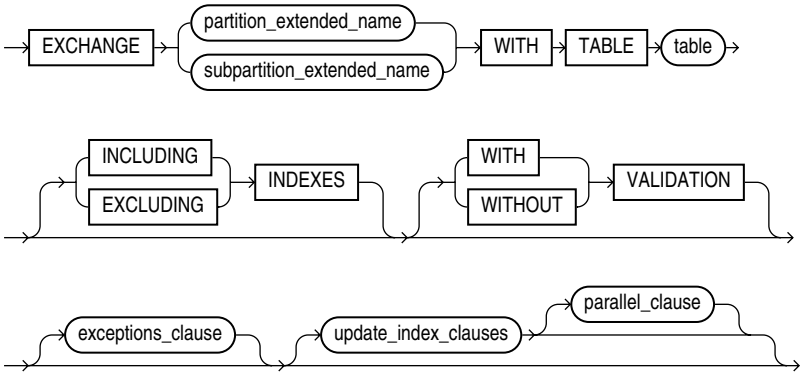
（12-7 ページの partition_extended_name::=、12-28 ページの partition_spec::=、
12-23 ページの dependent_tables_clause:=、12-29 ページの update_index_clauses::=、
12-6 ページの parallel_clause::= を参照）

merge_table_subpartitions::=



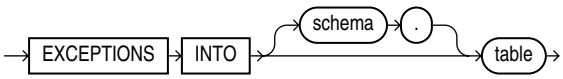
（12-7 ページの subpartition_extended_name::=、12-27 ページの range_subpartition_desc::=、
12-27 ページの list_subpartition_desc::=、12-29 ページの update_index_clauses::=、
12-6 ページの parallel_clause::= を参照）

exchange_partition_subpart::=

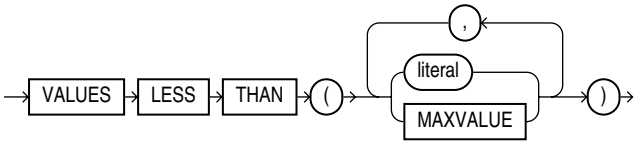


（12-7 ページの partition_extended_name::=、12-7 ページの subpartition_extended_name::=、
12-25 ページの exceptions_clause::=、12-29 ページの update_index_clauses::=、
12-6 ページの parallel_clause::= を参照）

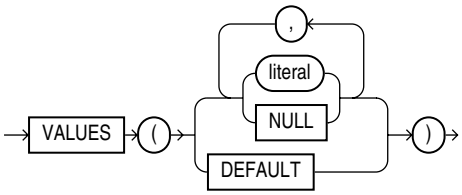
exceptions_clause::=



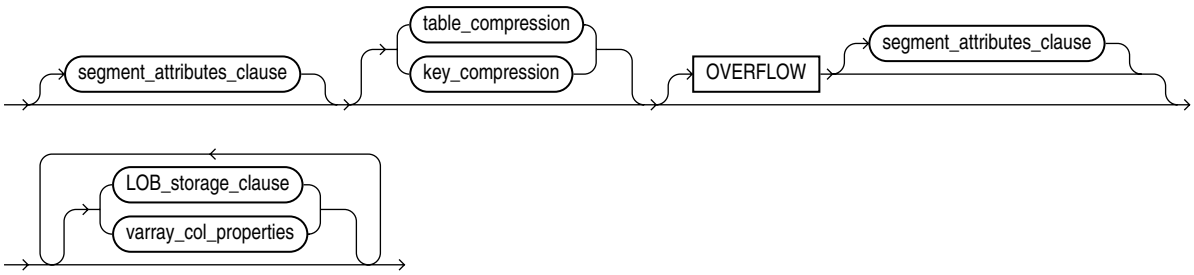
***range_values_clause*::=**



***list_values_clause*::=**

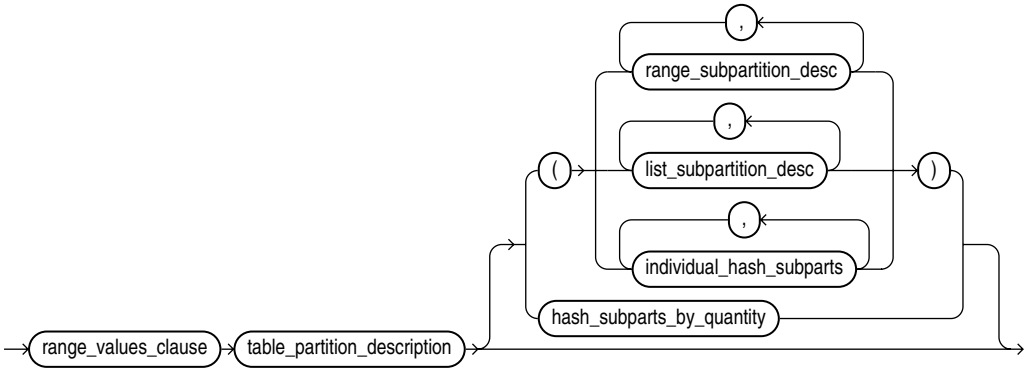


***table_partition_description*::=**



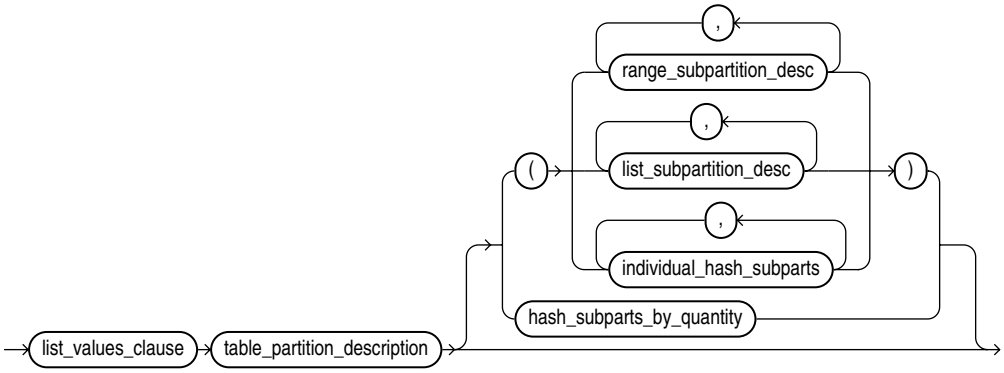
（12-7 ページの segment_attributes_clause::=、12-5 ページの table_compression::=、12-7 ページの key_compression::=、12-13 ページの LOB_storage_clause::=、12-13 ページの varray_col_properties::= を参照）

***range_partition_desc*::=**



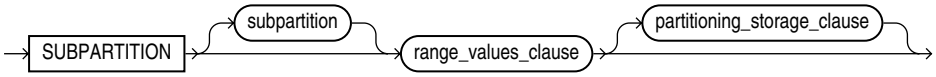
（12-26 ページの range_values_clause::=、12-26 ページの table_partition_description::=、12-27 ページの range_subpartition_desc::=、12-27 ページの list_subpartition_desc::= を参照）

***list_partition_desc*::=**



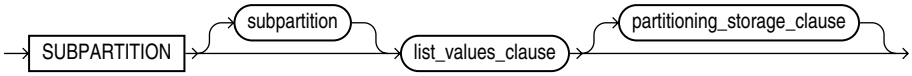
（12-26 ページの list_values_clause::=、12-26 ページの table_partition_description::=、12-27 ページの range_subpartition_desc::=、12-27 ページの list_subpartition_desc::= を参照）

***range_subpartition_desc*::=**



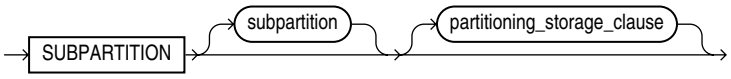
（12-26 ページの range_values_clause::=、12-28 ページの partitioning_storage_clause::= を参照）

***list_subpartition_desc*::=**

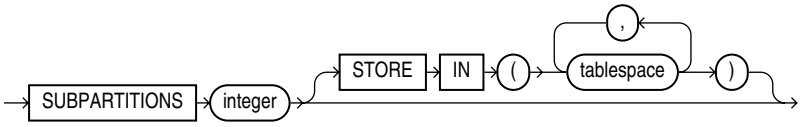


（12-26 ページの list_values_clause::=、12-28 ページの partitioning_storage_clause::= を参照）

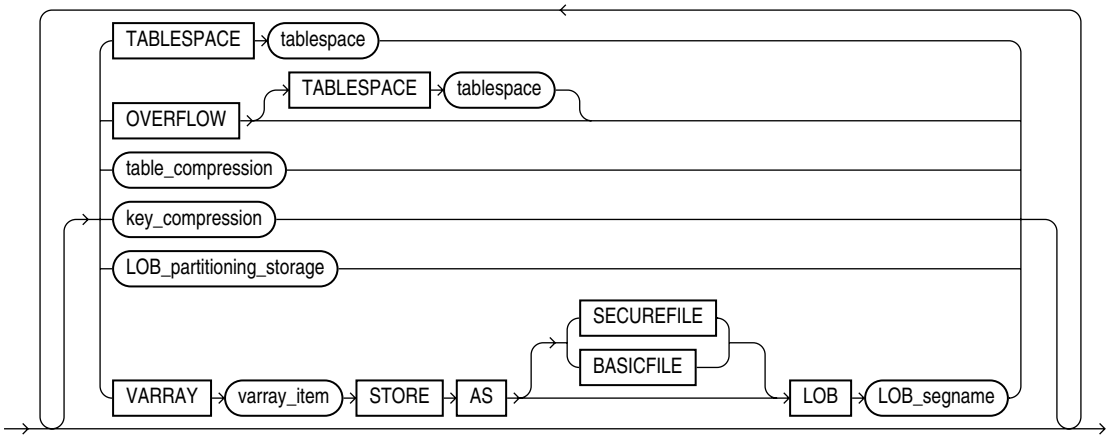
***individual_hash_subparts*::=**



***hash_subparts_by_quantity*::=**

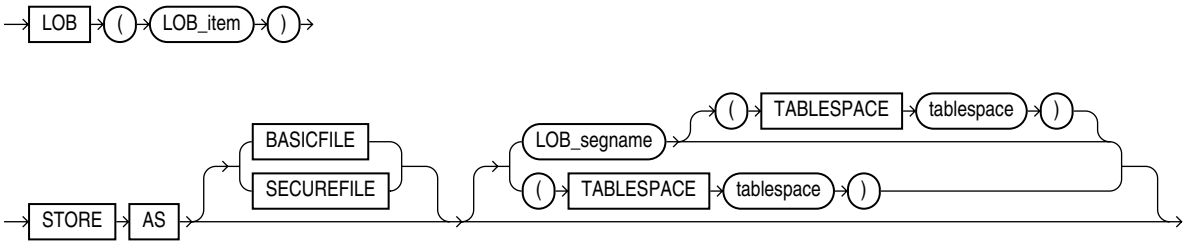


***partitioning_storage_clause*::=**

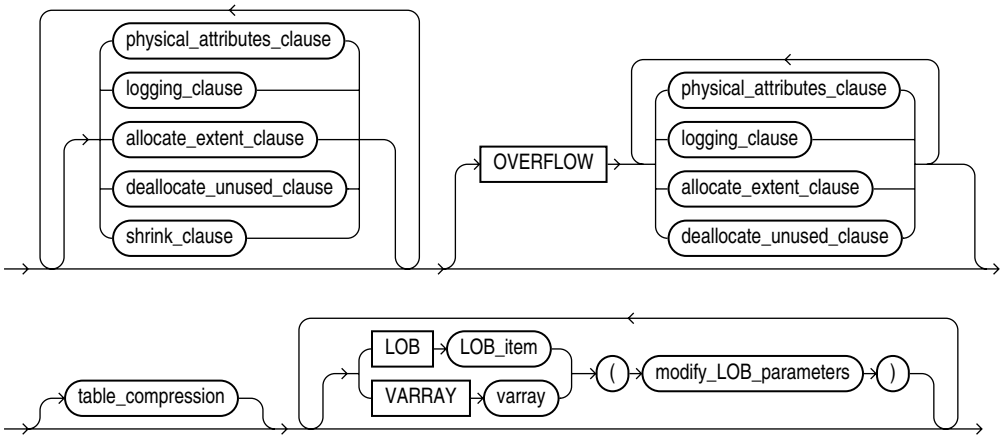


（12-5 ページの table_compression::=、12-28 ページの LOB_partitioning_storage::= を参照）

***LOB_partitioning_storage*::=**

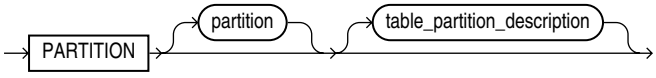


***partition_attributes*::=**



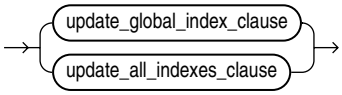
（12-4 ページの physical_attributes_clause::=、8-34 ページの logging_clause::=、
12-5 ページの allocate_extent_clause::=、12-6 ページの deallocate_unused_clause::=、
12-6 ページの shrink_clause::=、12-5 ページの table_compression::=、
12-15 ページの modify_LOB_parameters::= を参照）

***partition_spec*::=**



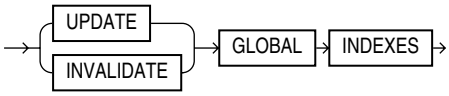
（12-26 ページの table_partition_description::= を参照）

update_index_clauses::=

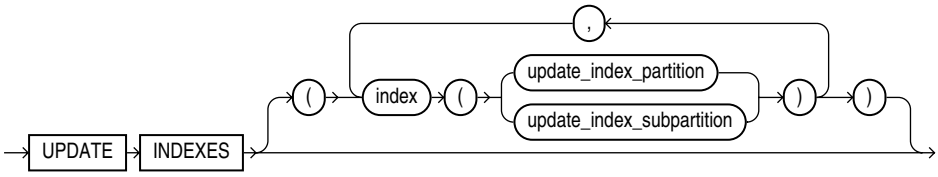


（12-29 ページの update_global_index_clause::=、12-29 ページの update_all_indexes_clause::= を参照）

update_global_index_clause::=

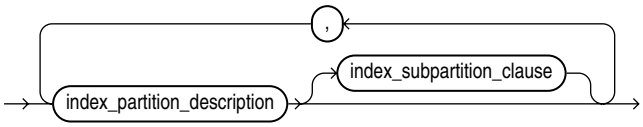


update_all_indexes_clause::=



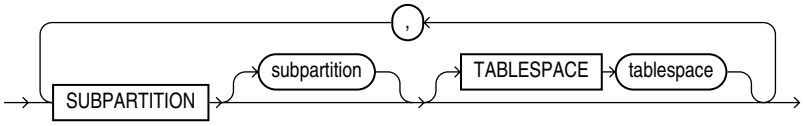
（12-29 ページの update_index_partition::=、12-29 ページの update_index_subpartition::= を参照）

***update_index_partition*::=**

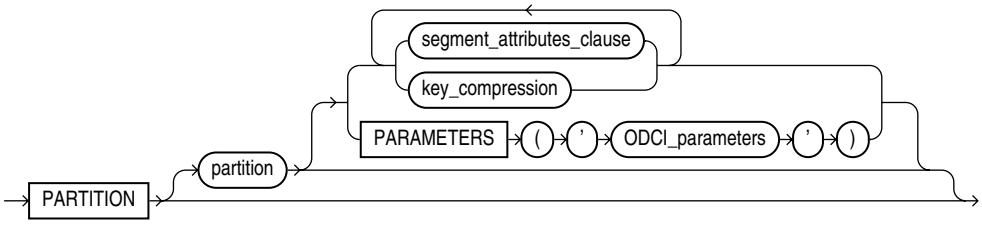


（12-29 ページの index_partition_description::=、12-30 ページの index_subpartition_clause::= を参照）

***update_index_subpartition*::=**

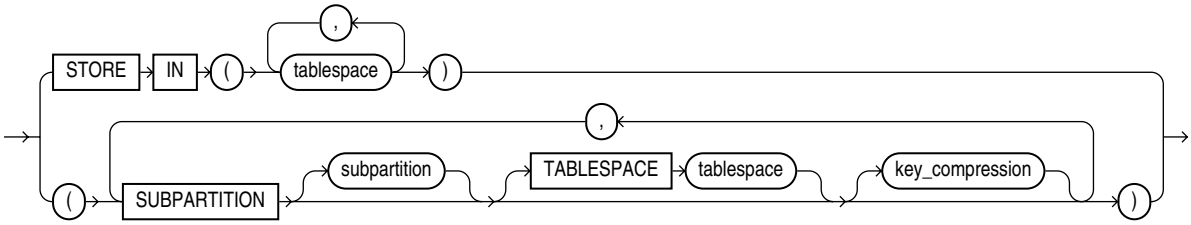


***index_partition_description*::=**

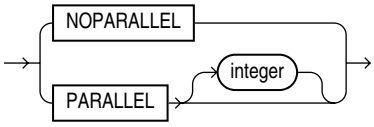


（12-7 ページの segment_attributes_clause::=、12-7 ページの key_compression::= を参照）

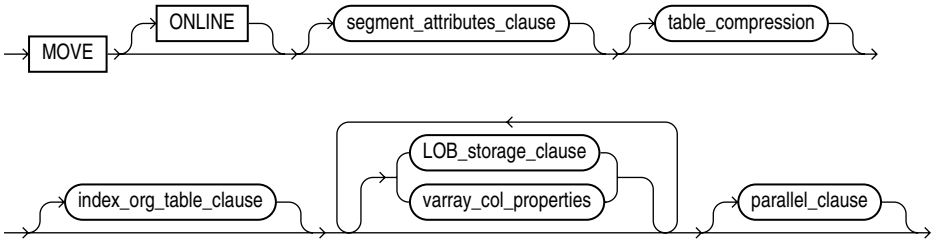
***index_subpartition_clause*::=**



parallel_clause::=

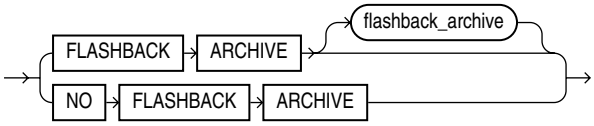


move_table_clause::=

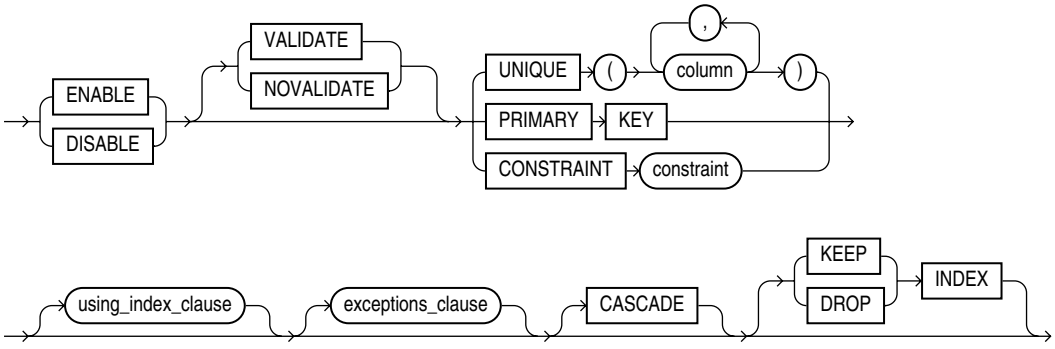


（12-7 ページの segment_attributes_clause::=、12-5 ページの table_compression::=、12-7 ページの index_org_table_clause::=、12-13 ページの LOB_storage_clause::=、12-13 ページの varray_col_properties::= を参照）

flashback_archive_clause::=

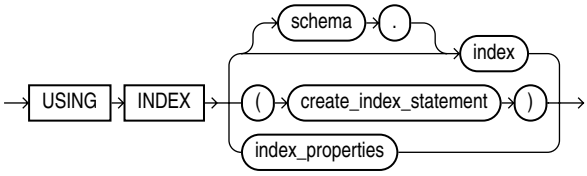


enable_disable_clause::=



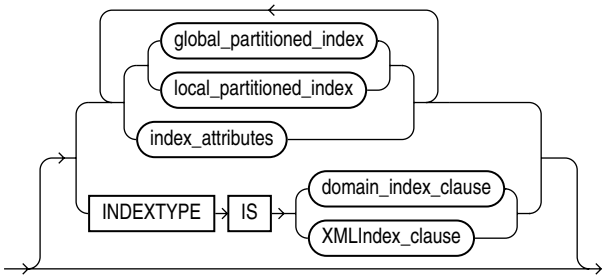
（12-31 ページの using_index_clause::=、12-25 ページの exceptions_clause::= を参照）

***using_index_clause*::=**



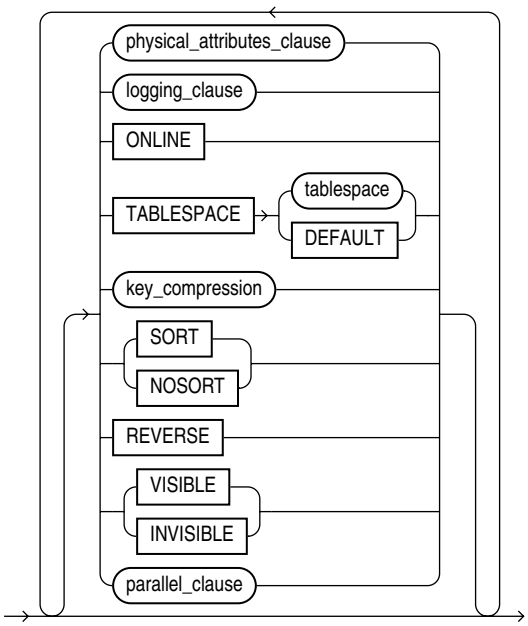
（14-51 ページの create_index::=、12-31 ページの index_properties::= を参照）

***index_properties*::=**



（「CREATE INDEX」の項にある 14-53 ページの global_partitioned_index::= と
14-55 ページの local_partitioned_index::=、および 12-31 ページの index_attributes::= を参照。
domain_index_clause は、*using_index_clause* ではサポートされていません。）

***index_attributes*::=**



（12-4 ページの physical_attributes_clause::=、8-34 ページの logging_clause::=、
12-7 ページの key_compression::= を参照。*parallel_clause* は、*using_index_clause*
ではサポートされていません。）

セマンティクス

ALTER TABLE 文の多くの句の機能は、CREATE TABLE 文での機能と同じです。これらの句の詳細は、16-6 ページの「CREATE TABLE」を参照してください。

注意： ALTER TABLE 文を実行すると、変更した表にアクセスするプロシージャおよびストアド・ファンクションが無効になる場合があります。無効になる条件および具体的な状況については、『Oracle Database アドバンスト・アプリケーション開発者ガイド』を参照してください。

---

schema

表が含まれているスキーマを指定します。*schema* の指定を省略した場合、表は自分のスキーマ内にあるとみなされます。

table

変更する表の名前を指定します。

注意： 1 つ以上のマテリアライズド・ビューのマスター表となっている表を変更した場合、マテリアライズド・ビューには INVALID のマークが付けられます。無効なマテリアライズド・ビューは、クエリー・リライトによって使用できません。また、リフレッシュすることもできません。マテリアライズド・ビューを再検証する場合は、11-2 ページの「ALTER MATERIALIZED VIEW」を参照してください。

参照： マテリアライズド・ビューに関する一般的な情報については、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

一時表の変更の制限事項： 一時表の変更、一時表内の列の削除または一時表の名前の変更を実行できます。ただし、一時表に対して次のことはできません。

- ネストした表型の列の追加（他の型の列は追加できます）
- 追加または変更された列の参照整合性（外部キー）制約の指定
- 追加または変更された LOB 列に対する次の *LOB_storage_clause* の句の指定：TABLESPACE、*storage_clause*、*logging_clause*、*allocate_extent_clause* または *deallocate_unused_clause*
- *physical_attributes_clause*、*nested_table_col_properties*、*parallel_clause*、*allocate_extent_clause*、*deallocate_unused_clause* または索引構成表（IOT）句の指定
- パーティション表と一時表の間でのパーティションの交換
- *logging_clause* の指定
- MOVE の指定

外部表の変更の制限事項： 外部表の列を追加、削除または変更できます。ただし、外部表に対して次のことはできません。

- LONG、LOB またはオブジェクト型の列の追加、または外部表の列のデータ型をこれらのデータ型のいずれかに変換
- 外部表への制約の追加
- 外部表の記憶域パラメータの変更
- *logging_clause* の指定
- MOVE の指定

alter_table_properties

alter_table_clauses を使用すると、データベースの表を変更できます。

physical_attributes_clause

physical_attributes_clause を使用すると、PCTFREE、PCTUSED および INITRANS パラメータの値と記憶特性を変更できます。これらのパラメータおよび特性の詳細は、8-39 ページの「physical_attributes_clause」および 8-41 ページの「storage_clause」を参照してください。

表の物理属性の変更の制限事項 : 物理属性の変更には、次の制限事項があります。

- PCTUSED パラメータは、索引構成表の索引セグメントに対して指定できません。
- ローカル管理表領域の表の記憶域属性を変更しようとすると、エラーが発生します。ただし、ローカル管理表領域にパーティション表のセグメントがあり、ディクショナリ管理表領域に他のセグメントがある場合、ディクショナリ管理表領域のセグメントの記憶域属性は変更されますが、ローカル管理表領域のセグメントの記憶域属性は変更されません。また、エラーも発生しません。
- 自動セグメント領域管理のセグメントの場合は、PCTUSED 設定の変更は無視されます。PCTFREE 設定を変更した場合、その変更をセグメントに割当て済のブロックに実装するには、DBMS_REPAIR.SEGMENT_FIX_STATUS プロシージャを実行する必要があります。

表の物理属性の変更の注意事項 : この句で指定する値が表に与える影響を次に示します。

- 非パーティション表の場合、作成時に表に指定した値は新しく指定した値によって上書きされます。
- レンジ・パーティション表、リスト・パーティション表またはハッシュ・パーティション表の場合、新しく指定した値がその表のデフォルト値およびすべての既存パーティションに対する実際の値となり、そのパーティションにすでに設定されていた値は上書きされます。既存のパーティション値を上書きせずにデフォルトの表属性を変更する場合は、*modify_table_default_attrs* を使用してください。
- コンポジット・パーティション表の場合、新しく指定した値がその表とその表のすべてのパーティションのデフォルト値、およびその表のすべてのサブパーティションに対する実際の値となり、そのサブパーティションにすでに設定されていた値は上書きされます。既存のサブパーティションの値を上書きせずにデフォルトのパーティション属性を変更する場合は、FOR PARTITION 句とともに *modify_table_default_attrs* を使用してください。

logging_clause

logging_clause を使用すると、表のロギング属性を変更できます。また、*logging_clause* では、後続の ALTER TABLE ... MOVE および ALTER TABLE ... SPLIT 操作のログを記録するかどうかも指定します。

modify_table_default_attrs 句とともにこの句を使用した場合、パーティション表のロギング属性が影響を受けます。

参照 :

- この句の詳細は、8-34 ページの「logging_clause」を参照してください。
- *logging_clause* およびパラレル DML の詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

table_compression

table_compression 句は、ヒープ構成表に対してのみ有効です。この句を使用すると、ディスクおよびメモリーの使用量を削減するために、データ・セグメントを圧縮するかどうかを指定できます。この句のセマンティクスの詳細および表の圧縮を使用したオブジェクトの作成方法の詳細は、16-31 ページの「CREATE TABLE」の「table_compression」を参照してください。

注意： 初めて、圧縮データが追加されるように表を変更する場合は、表のすべてのビットマップ索引およびビットマップ索引パーティションに UNUSABLE のマークを付ける必要があります。

参照： 表の圧縮の使用例は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

supplemental_table_logging

supplemental_table_logging 句を使用すると、REDO ログ・グループ、またはサプリメンタル・ログが記録される REDO ログ・グループの 1 つ以上の列を追加または削除できます。

- ADD 句で *supplemental_log_grp_clause* を使用すると、指定したサプリメンタル・ログ・グループを作成できます。*supplemental_id_key_clause* を使用すると、システム生成のログ・グループを作成できます。
- DROP 句で GROUP *log_group* 構文を使用すると、指定したサプリメンタル・ログ・グループを削除できます。*supplemental_id_key_clause* を使用すると、システム生成のログ・グループを削除できます。

supplemental_log_grp_clause および *supplemental_id_key_clause* は、CREATE TABLE 文および ALTER TABLE 文で同じセマンティクスを持ちます。これらの句の詳細は、「CREATE TABLE」の 16-29 ページの「supplemental_log_grp_clause」および 16-29 ページの「supplemental_id_key_clause」を参照してください。

参照： サプリメンタル REDO ログ・グループの詳細は、『Oracle Data Guard 概要および管理』を参照してください。

allocate_extent_clause

allocate_extent_clause を使用すると、表、パーティション、サブパーティション、オーバーフロー・データ・セグメント、LOB データ・セグメントまたは LOB 索引に新しいエクステントを明示的に割り当てることができます。

表のエクステントの割当ての制限事項： 一時表、レンジ・パーティション表またはコンポジット・パーティション表にエクステントを割り当てることはできません。

参照： この句の詳細は、8-2 ページの「allocate_extent_clause」および 12-78 ページの「エクステントの割当て例 :」を参照してください。

deallocate_unused_clause

deallocate_unused_clause を使用すると、表、パーティション、サブパーティション、オーバーフロー・データ・セグメント、LOB データ・セグメントまたは LOB 索引の最後にある未使用領域の割当てを明示的に解除できます。解放された領域は、表領域の他のセグメントから利用できます。

参照： この句の詳細は、8-24 ページの「deallocate_unused_clause」および 12-74 ページの「未使用領域の解放例 :」を参照してください。

shrink_clause

shrink_clause を使用すると、表、索引構成表またはそのオーバーフロー・セグメント、索引、パーティション、サブパーティション、LOB セグメント、マテリアライズド・ビューまたはマテリアライズド・ビュー・ログの領域を手動で縮小できます。この句は、自動セグメント管理を使用した表領域のセグメントに対してのみ有効です。デフォルトでは、セグメントが縮小されて最高水位標が調整され、再生領域がすぐに解放されます。

セグメントの縮小では、行移動が必要です。したがって、この句を指定する前に、縮小するオブジェクトの行移動を使用可能にする必要があります。また、アプリケーションで ROWID に基づいたトリガーを使用している場合、この句を発行する前にこのトリガーを使用禁止にする必要があります。

> **注意：** *shrink_clause* を指定する前に、索引構成表に対して行の移動を有効にしないでください。索引構成表の ROWID は、その主キーであり、変更できません。このため、そのような表に対して行の移動は関係なく、有効でもありません。

COMPACT COMPACT を指定すると、セグメント領域のデフラグのみが行われ、後で領域を解放できるように表の行が縮小されます。最高水位標の再調整および領域の解放は、すぐに行われません。操作を完了するには、後で別の ALTER TABLE ... SHRINK SPACE 文を発行する必要があります。この句は、1 回の長い手順のかわりに 2 回の短い手順で縮小操作を実行する場合に便利です。

索引または索引構成表では、ALTER [INDEX | TABLE] ... SHRINK SPACE COMPACT を指定することは、ALTER [INDEX | TABLE] ... COALESCE を指定することと同じです。*shrink_clause* はカスケード実行できます（次の CASCADE を参照）。また、結合操作の場合よりセグメントを高密度に圧縮できるため、パフォーマンスが向上します。ただし、未使用領域を解放しない場合は、適切な COALESCE 句を使用します。

CASCADE CASCADE を指定すると、*table* のすべての依存オブジェクト（索引構成表の 2 次索引を含む）に対して同じ操作を実行できます。

***shrink_clause* の制限事項：** *shrink_clause* には、次の制限事項があります。

- 同じ ALTER TABLE 文で、この句と別の句を組み合せて使用することはできません。

  この句は、クラスタ、クラスタ化表または LONG 列を含むすべてのオブジェクトには指定できません。
- セグメントの縮小は、ファンクション索引またはビットマップ結合索引を含む表ではサポートされません。
- CASCADE を指定しても、この句では索引構成表のマッピング表を縮小できません。
- この句は、圧縮表には指定できません。
- ON COMMIT マテリアライズド・ビューのマスター表は縮小できません。ROWID マテリアライズド・ビューは、縮小操作の実行後に再構築する必要があります。

CACHE | NOCACHE

CACHE 句および NOCACHE 句のセマンティクスは、CREATE TABLE 文および ALTER TABLE 文で同じです。これらの句の詳細は、16-53 ページの「CREATE TABLE」の「CACHE | NOCACHE | CACHE READS」を参照してください。ALTER TABLE 文でこれらの句をいずれも省略した場合、既存の値は変更されません。

upgrade_table_clause

upgrade_table_clause は、オブジェクト表およびオブジェクト列を含むリレーショナル表に対して有効です。この句を使用すると、ターゲットとなる表のメタデータが参照される各型の最新バージョンに準拠する型に変換されます。表が有効な場合、表のメタデータは変更されないままとなります。

オブジェクト表およびオブジェクト列の更新の制限事項： この句の中では、*object_type_col_properties* を *column_properties* 句として指定できません。

INCLUDING DATA INCLUDING DATA を指定すると、表のデータを型の最新バージョンの形式に変換できます。column_properties および LOB_partition_storage を使用して表を更新する間に、新しいすべての列の記憶域を定義できます。これはデフォルトです。

ALTER TYPE 文の *dependent_handling_clause* に CASCADE INCLUDING TABLE DATA を指定すると、型の更新時に表のデータを変換できます。この句の詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。表が古いバージョンの型に基づくデータを含むかどうかを確認するには、USER_TAB_COLUMNS データ・ディクショナリ・ビューの DATA_UPGRADED 列を参照します。

NOT INCLUDING DATA NOT INCLUDING DATA を指定すると、列データは変更されません。

NOT INCLUDING DATA の制限事項： 表が Oracle8 リリース 8.0.x のイメージ・フォーマットを含む場合、NOT INCLUDING DATA を指定できません。表がこのような列を含むかどうかを確認するには、USER_TAB_COLUMNS データ・ディクショナリ・ビューの V80_FMT_IMAGE 列を参照します。

参照：

- データ・ディクショナリ・ビューの詳細は、『Oracle Database リファレンス』を参照してください。
- 表が依存する型を変更する際に、依存表のデータを変換する場合の詳細は、13-4 ページの「ALTER TYPE」を参照してください。

records_per_block_clause

records_per_block_clause を使用すると、1 ブロックに格納できるレコード数を制限するかどうかを指定できます。この句によって、この後、表に作成されるビットマップ索引はできるだけ縮小されます。

ブロック内のレコードの制限事項： *record_per_block_clause* には、次の制限事項があります。

- 表にすでにビットマップ索引が定義されている場合は、MINIMIZE または NOMINIMIZE のいずれも指定できません。まず、ビットマップ索引を削除する必要があります。
- この句は、索引構成表またはネストした表に対して指定できません。

MINIMIZE MINIMIZE を指定すると、表の各ブロックの最大レコード数を計算し、ブロックに含まれるレコード数がその数を超えないように挿入操作を制限できます。

MINIMIZE を指定する前に、表にデータのサンプル・セットを定義しておくことをお薦めします。表の圧縮を使用している場合（12-34 ページの「table_compression」を参照）、圧縮データのサンプル・セットは、すでに表に存在している必要があります。

MINIMIZE の制限事項： MINIMIZE は、空の表に対して指定できません。

NOMINIMIZE NOMINIMIZE を指定すると、MINIMIZE 機能が無効になります。これはデフォルトです。

row_movement_clause

親表でも行の移動が無効化されている場合を除き、参照パーティション表で行の移動を無効にすることはできません。それ以外では、この句のセマンティクスは、CREATE TABLE 文および ALTER TABLE 文で同じです。これらの句の詳細は、16-55 ページの「CREATE TABLE」の「row_movement_clause」を参照してください。

flashback_archive_clause

この句を指定するには、指定されたフラッシュバック・データ・アーカイブに対する FLASHBACK ARCHIVE オブジェクト権限が必要です。この句を使用すると、表の履歴追跡を有効または無効にできます。

- 表の追跡を有効にするには、FLASHBACK ARCHIVE を指定します。*flashback_archive* を指定すると、この表に特定のフラッシュバック・データ・アーカイブを指定できます。指定するフラッシュバック・データ・アーカイブは、すでに存在している必要があります。

  アーカイブ名を指定しない場合、データベースは、システムに指定されたデフォルトのフラッシュバック・データ・アーカイブを使用します。システムにデフォルトのフラッシュバック・データ・アーカイブが指定されていない場合は、*flashback_archive* を指定する必要があります。

  FLASHBACK ARCHIVE を指定して、この表のフラッシュバック・データ・アーカイブを変更することはできません。かわりに、まず ALTER TABLE 文に NO FLASHBACK ARCHIVE 句を付けて実行し、次に、ALTER TABLE 文に FLASHBACK ARCHIVE 句を付けて実行します。

- 表の追跡を無効にするには、NO FLASHBACK ARCHIVE を指定します。

> **参照：** 追跡が有効な表の作成の詳細は、16-56 ページの「CREATE TABLE」の「flashback_archive_clause」を参照してください。デフォルトのフラッシュバック・データ・アーカイブの作成の詳細は、14-45 ページの「CREATE FLASHBACK ARCHIVE」を参照してください。

RENAME TO

RENAME 句を使用すると、表の名前を *new_table_name* に変更できます。

この句を使用した場合、依存するすべてのマテリアライズド・ビューは無効になります。マテリアライズド・ビューの詳細は、15-4 ページの「CREATE MATERIALIZED VIEW」および『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

表にドメイン索引が定義されている場合、ODCIIndexAlter() メソッドが RENAME オプション付きで起動されます。この操作によって、索引タイプ・メタデータと実表の間の対応が確立されます。

READ ONLY | READ WRITE

READ ONLY を指定すると、表を読取り専用モードに設定できます。表が READ ONLY モードの場合は、表に影響する DML 文または SELECT ... FOR UPDATE 文は発行できません。表データを変更しない DDL 文は発行できます。表が READ ONLY モードの場合は、表に関連する索引の操作は可能です。

READ WRITE を指定すると、読取り専用表を読取り / 書込みモードに戻すことができます。

REKEY *encryption_spec*

REKEY 句を使用すると、新しい暗号化キーの生成または異なるアルゴリズム間の切替えができます。この操作は、表内の LOB 列を含むすべての暗号化された列が再度暗号化された後にのみ戻されます。

alter_iot_clauses

index_org_table_clause

この句を使用すると、既存の索引構成表の特性の一部を変更できます。索引構成表は、主キーによってソートされたデータを保持する表であり、主キーに基づくアクセスおよび操作には最適です。詳細は、16-33 ページの「CREATE TABLE」の「index_org_table_clause」を参照してください。

> **参照：** 「索引構成表の変更例 :」（12-75 ページ）

key_compression

この句は、*table* が索引構成されている場合のみ有効です。COMPRESS を指定すると、ブロックを再利用するために、索引構成表の主キー索引ブロックを空きブロックに結合できます（可能な場合）。この句は *parallel_clause* とあわせて指定できます。

PCTTHRESHOLD *integer* 16-33 ページの「CREATE TABLE」の「PCTTHRESHOLD integer」を参照してください。

INCLUDING *column_name* 16-34 ページの「CREATE TABLE」の「INCLUDING column_name」を参照してください。

overflow_attributes

overflow_attributes を使用すると、索引構成表に対して、変更するオーバーフロー・データ・セグメントの物理記憶域属性およびロギング属性を指定できます。この句に指定するパラメータの値は、オーバーフロー・データ・セグメントにのみ適用されます。

> **参照：** 「CREATE TABLE」（16-6 ページ）

add_overflow_clause

add_overflow_clause を使用すると、指定した索引構成表にオーバーフロー・データ・セグメントを追加できます。また、この句を使用すると、エクステントを明示的に割り当てたり、既存のオーバーフロー・セグメントから未使用領域の割当てを解除することができます。

STORE IN *tablespace* 句を使用すると、オーバーフロー・セグメント全体に対する表領域の記憶域を指定できます。PARTITION 句を使用すると、パーティションによるセグメントに対する表領域の記憶域を指定できます。

パーティション化された索引構成表の場合、次の点に注意してください。

- PARTITION を指定しない場合、それぞれのパーティションに自動的にオーバーフロー・セグメントが割り当てられます。これらのセグメントの物理属性は表のレベルから継承されます。
- 1 つ以上のパーティションに別々の物理属性を指定する場合、表の各パーティションに属性を指定する必要があります。パーティションの名前を指定する必要はありませんが、パーティションが作成された順番で属性を指定する必要があります。

パーティションの順番を確認するには、USER_IND_PARTITIONS ビューの PARTITION_NAME および PARTITION_POSITION 列を問い合せます。

TABLESPACE を指定していないパーティションがある場合、表に対して指定された表領域が使用されます。表レベルで TABLESPACE を指定していない場合は、そのパーティションの主キー索引セグメントの表領域が使用されます。

オーバーフロー属性の制限事項： *segment_attributes_clause* では、次のことに注意します。

- *physical_attributes_clause* の OPTIMAL パラメータを指定できません。
- この句を使用して、オーバーフロー・セグメントの表領域の記憶域を指定できません。非パーティション表の場合、ALTER TABLE ... MOVE ... OVERFLOW を使用して、セグメントを異なる表領域に移動します。パーティション表の場合、ALTER TABLE ... MODIFY DEFAULT ATTRIBUTES ... OVERFLOW を使用して、オーバーフロー・セグメントのデフォルト表領域を変更します。

table がローカル管理表領域にある場合、表領域のセグメント属性の中にはデータベースによって自動的に管理されるものがあるため、制限事項が追加されます。

> **参照：** *add_overflow_clause* 句の詳細は、8-2 ページの「allocate_extent_clause」および 8-24 ページの「deallocate_unused_clause」を参照してください。

alter_overflow_clause

alter_overflow_clause を使用すると、既存の索引構成表のオーバーフロー・セグメントの定義を変更できます。

add_overflow_clause の制限事項は、*alter_overflow_clause* にも適用されます。

> **注意：** 索引構成表に列を追加した場合、各列の最大サイズが評価され、行の最大値が計算されます。オーバーフロー・セグメントが必要で、OVERFLOW を指定していない場合は、エラーが発生し ALTER TABLE 文は実行されません。このチェック機能によって、索引構成表に対する後続の DML 操作が、オーバーフロー・セグメントがないために失敗することを回避できます。

alter_mapping_table_clauses

alter_mapping_table_clauses は、*table* が索引構成されており、マッピング表を持つ場合にのみ有効です。

allocate_extent_clause *allocate_extent_clause* を使用すると、新しいエクステントを索引構成表のマッピング表の終わりに割り当てることができます。この句の詳細は、8-2 ページの「allocate_extent_clause」を参照してください。

deallocate_unused_clause *deallocate_unused_clause* を指定すると、索引構成表のマッピング表の終わりにある未使用領域の割当てを解除できます。この句の詳細は、8-24 ページの「deallocate_unused_clause」を参照してください。

マッピング表またはそのパーティションに関する他のすべての属性は、自動的に管理されます。

COALESCE 句

COALESCE を指定すると、ブロックを再利用するために、索引構成表の管理に使用される索引の索引ブロックの内容を空きブロックにマージできます（可能な場合）。この句と shrink_clause の関係については、12-35 ページの「shrink_clause」を参照してください。

alter_XMLSchemas_clause

この句は、XMLType 表を変更する場合にのみ、*alter_table_properties* の一部として有効です。詳細は、12-43 ページの「alter_XMLSchemas_clause」を参照してください。

column_clauses

これらの句を指定すると、列を追加、削除または変更できます。

add_column_clause

add_column_clause を使用すると、表に列を追加できます。

> **参照：** この句のキーワードとパラメータの詳細は、16-6 ページの「CREATE TABLE」および 12-78 ページの「表の列の追加例 :」を参照してください。

column_definition

既存の表に列を追加するときの *column_definition* の要素の動作は、この項で特に述べられていないかぎり、新しい表の作成時に列を追加するときの動作と同じです。詳細は、16-25 ページの「column_definition」を参照してください。

***column_definition* の制限事項** SORT パラメータは、新しい表を作成する場合にのみ有効です。ALTER TABLE ... ADD 文の column_definition では、SORT を指定できません。

列を追加すると、新しい列の各行に初期値として NULL が設定されます。NOT NULL 列に DEFAULT 句を指定すると、デフォルト値はメタデータとして格納されますが、列自体にはデータは移入されません。ただし、デフォルト値が結果セットに戻されるように、新しい列を指定する後続の問合せは再書込みされます。

この最適化された動作は、以前のリリースとは異なります。以前のリリースでは、ALTER TABLE 操作の一部として、Oracle Database は新しく作成された列内の行をデフォルト値で更新し、表に定義された AFTER UPDATE トリガーを起動していました。ただし、最適化された動作には、次の制限事項があります。

- 表に LOB 列を持つことはできません。対象となる表は、索引構成化したり、一時表またはクラスタ化された表にすることはできません。また、キュー表、オブジェクト表またはマテリアライズド・ビューのコンテナ表にすることもできません。
- 追加する列は暗号化できず、オブジェクト列、ネストした表の列または LOB 列にすることはできません。

注意： 列にデフォルト値がある場合、DEFAULT 句を使用してデフォルトを NULL に変更することができますが、デフォルト値を完全に削除することはできません。列に割り当てられたデフォルト値がある場合、USER_TAB_COLUMNS データ・ディクショナリ・ビューの DATA_DEFAULT 列にはデフォルト値または NULL のいずれかが表示されます。

パーティション化された索引構成表の各パーティションには、オーバーフロー・データ・セグメントを追加できます。

非パーティションおよびパーティション表に LOB 列を追加できます。表、およびパーティションまたはサブパーティションのレベルで LOB 記憶域を指定できます。

SELECT * 構文を使用して、表からすべての列を選択するように指定した問合せを使用してビューを作成した場合、*table* に列を追加しても、新しい列がビューに自動的に追加されることはありません。ビューに新しい列を追加する場合、CREATE VIEW 文に OR REPLACE 句を指定してビューを再作成してください。詳細は、17-13 ページの「CREATE VIEW」を参照してください。

virtual_column_definition

virtual_column_definition は列を追加する場合、列を作成する場合と同じセマンティクスを持ちます。

> **参照：** 詳細は、16-27 ページの「CREATE TABLE」の「virtual_column_definition」および 12-78 ページの「仮想表の列の追加 例 :」を参照してください。

列の追加の制限事項： 列の追加には、次の制限事項があります。

- クラスタ化表には、LOB 列を追加できません。
- LOB 列をハッシュ・パーティション表に追加する場合、新しいパーティションに対して指定できる属性は、TABLESPACE のみです。
- *table* に行がある場合、DEFAULT 句を指定しないかぎり、NOT NULL 制約のある列を追加できません。
- 索引構成表にこの句を指定した場合、同じ文では他の句を指定できません。

DEFAULT

DEFAULT 句を使用すると、新しい列にデフォルト値を指定したり、既存の列に新しいデフォルト値を指定することができます。後続の INSERT 文で列に値を指定しない場合、この値が自動的に割り当てられます。表に新しい列を追加する場合、デフォルト値を指定すると、その表のすべての行にデフォルトの列値が挿入されます。

デフォルト値のデータ型は、列に対して指定したデータ型と一致している必要があります。また、列には、このデフォルト値を保持できる十分な大きさが必要です。

列のデフォルト値の制限事項： 列のデフォルト値には、次の制限事項があります。

- DEFAULT 式に、他の列、疑似列 CURRVAL、NEXTVAL、LEVEL および ROWNUM、または完全には指定されていない日付定数に対する参照を指定することはできません。
- 式には、スカラー副問合せ式を除くすべての書式を使用できます。

参照： 「デフォルト列値の指定例 :」（12-78 ページ）

inline_constraint

inline_constraint を使用すると、新しい列に制約を追加できます。

inline_ref_constraint

inline_ref_constraint を使用すると、新しい REF 型の列を定義できます。制約の型の構文や制限などの詳細は、8-4 ページの「constraint」を参照してください。

column_properties

column_properties の句を使用すると、オブジェクト型、ネストした表、VARRAY または LOB 列の記憶特性を指定できます。

object_type_col_properties この句は新しいオブジェクト型列や属性を追加する場合にのみ有効です。*modify_column_clauses* を使用すると、既存のオブジェクト型列のプロパティを変更できます。この句のセマンティクスは、特に指定がないかぎり、CREATE TABLE と同じです。

object_type_col_properties 句を使用すると、新しいオブジェクト列、オブジェクト属性、コレクション列およびコレクション属性の要素に対する記憶特性を指定できます。

この句の詳細は、16-36 ページの「CREATE TABLE」の「object_type_col_properties」を参照してください。

nested_table_col_properties *nested_table_col_properties* 句を使用すると、ネストした表に対して別の記憶特性を指定し、そのネストした表を索引構成表として定義できます。ネストした表の型を持つ列または列属性付きで表を作成する場合は、この句を挿入する必要があります。（この句の中で、親オブジェクト表に対する場合と同じ働きをする句は、ここでは繰り返されません）。

- *nested_item* には、型がネストした表である列（または、ネストした表のオブジェクト型の最上位の属性）の名前を指定します。

  ネストした表がマルチレベル・コレクションで、内部のネストした表には名前が割り当てられていない場合、*nested_item* 名のかわりに COLUMN_VALUE を指定します。

- *storage_table* には、*nested_item* の行を含む表の名前を指定します。記憶表は、親表と同じスキーマ、および親表と同じ表領域内に作成されます。

ネストした表の列のプロパティの制限事項： ネストした表の列のプロパティには、次の制限事項があります。

- *parallel_clause* は指定できません。
- *physical_properties* 句の一部として CLUSTER を指定できません。

参照：「ネストした表の例：」（12-80 ページ）

varray_col_properties *varray_col_properties* を使用すると、VARRAY 型のデータが格納されている LOB に対して、別の記憶特性を指定できます。この句を指定する場合、インラインに格納できるほど小さい値でも、VARRAY は必ず LOB に格納されます。*varray_item* がマルチレベル・コレクションの場合、*varray_item* 内にネストされたすべてのコレクション項目は、常に *varray_item* と同じ LOB に格納されます。

VARRAY 列のプロパティの制限事項： VARRAY 列に対し、*LOB_parameters* の一部として TABLESPACE を指定することはできません。VARRAY に対する LOB 表領域のデフォルトは、表を含む表領域になります。

LOB_storage_clause

LOB_storage_clause を使用すると、新しく追加した LOB 列、LOB パーティション、LOB サブパーティションまたは LONG 列から LOB 列への変換時の LOB 記憶特性を指定できます。この句では、既存の LOB を変更できません。かわりに、12-49 ページの「modify_LOB_storage_clause」を使用する必要があります。

この項で特に指定がない場合は、*LOB_storage_clause* および *modify_LOB_storage_clause* 内のすべての LOB パラメータは、ALTER TABLE 文で、CREATE TABLE 文と同じセマンティクスを持ちます。この句の詳細は、16-37 ページの「CREATE TABLE」の「LOB_storage_clause」を参照してください。

LOB パラメータの制限事項： ハッシュ・パーティションまたはハッシュ・サブパーティションに指定できる *LOB_parameters* のパラメータは、TABLESPACE のみです。

CACHE READS 句 新しい LOB 列を追加するときに、CACHE READS でロギング属性を指定できます。作成時に LOB 列を定義するときにも指定できます。この句の詳細は、16-53 ページの「CREATE TABLE」の句「CACHE READS」を参照してください。

ENABLE | DISABLE STORAGE IN ROW STORAGE IN ROW は、一度設定すると変更できません。したがって、この句は *modify_col_properties* 句の一部には指定できません。ただし、新しい列を追加するとき（add_column_clause）、または表を移動するとき（move_table_clause）に、この設定を変更することはできます。この句の詳細は、16-38 ページの「CREATE TABLE」の句「ENABLE STORAGE IN ROW」を参照してください。

CHUNK *integer* *modify_col_properties* 句を使用して、設定後の CHUNK の値を変更することはできません。作成後の列に異なる CHUNK 値が必要な場合は、ALTER TABLE … MOVE を使用します。詳細は、16-38 ページの「CREATE TABLE」の句「CHUNK integer」を参照してください。

RETENTION BasicFile LOB については、データベースが自動 UNDO モードで稼働している場合、旧バージョンの LOB を保持するには、PCTVERSION ではなく RETENTION を指定します。この句によって、PCTVERSION のこれまでの設定が上書きされます。このパラメータの詳細は、16-39 ページの「CREATE TABLE」の句「LOB_retention_clause」を参照してください。

FREEPOOLS *integer* データベースが自動 UNDO モードで稼働している場合、BasicFile LOB に対してこの句を使用すると LOB の空きリスト・グループ数を指定できます。この句によって、FREELIST GROUPS のこれまでの設定が上書きされます。このパラメータの詳細は、16-39 ページの「CREATE TABLE」の句「FREEPOOLS integer」を参照してください。SecureFile LOB の場合は、データベースはこのパラメータを無視します。

LOB_partition_storage

1 つの ALTER TABLE 文では、*LOB_partition_storage* のリストを 1 つのみ指定できます。すべての *LOB_storage_clauses* および *varray_col_properties* 句は、*LOB_partition_storage* 句のリストの前に指定する必要があります。制限を含むこの句の詳細は、16-41 ページの「CREATE TABLE」の句「LOB_partition_storage」を参照してください。

XMLType_column_properties この句の詳細は、16-43 ページの「CREATE TABLE」の句「XMLType_column_properties」を参照してください。

XMLSchema_spec この句の詳細は、16-59 ページの「CREATE TABLE」の句「XMLSchema_spec」を参照してください。

alter_XMLSchemas_clause この句を使用すると、1 つ以上の XMLSchema 指定を XMLType 表に追加または削除できます。

- XMLSchema を削除すると、表がスキャンされ、その特定のスキーマを使用してエンコードされた XMLType 行が削除されます。
- 表で非スキーマ・バイナリ XML データが使用可能である場合、表が空でないときは、XMLSchema を追加できません。
- オプションの ALLOW | DISALLOW 句は、BINARY XML 記憶域を指定した XMLType 列に対してのみ有効です。これらの句の詳細は、16-59 ページの「CREATE TABLE」の「XMLSchema_spec」を参照してください。

参照 :

- *LOB_segname* および *LOB_parameters* 句の詳細は、12-42 ページの「LOB_storage_clause」を参照してください。
- オブジェクト・リレーショナル表における XMLType 列の例は、16-65 ページの「XMLType 列の例 :」を参照してください。XMLSchema の作成例は、E-8 ページの「SQL 文での XML の使用方法」を参照してください。
- XMLType 列と表および XMLSchema の作成の詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

modify_column_clauses

modify_column_clauses を使用すると、既存の列のプロパティや既存のオブジェクト型の列の代替性を変更できます。

参照 : 「表の列の変更例 :」(12-78 ページ)

modify_col_properties

この句を使用すると、列のプロパティを変更できます。この句で省略した列定義のオプション部分（データ型、デフォルト値、制約）は、変更されません。

datatype 列のすべての行が NULL の場合、列のデータ型を変更できます。ただし、マテリアライズド・ビューのコンテナ表にある列のデータ型を変更した場合、それに対応するマテリアライズド・ビューが無効になります。

参照整合性制約の外部キーの一部として、文で列が指定されている場合のみ、データ型を省略できます。参照整合性制約の参照キーに対応する列のデータ型が、その列に自動的に割り当てられます。

すべての行に NULL 値が存在するどうかにかかわらず、文字型または RAW 型の列のサイズ、または数値型の列の精度は、いつでも大きくすることができます。また、変更を行ってもデータを変更しなくて済む場合、列のデータ型のサイズを小さくすることができます。既存のデータがスキャンされ、新しいサイズの制限を超えるデータが存在する場合はエラーが戻されます。

DATE 列を TIMESTAMP または TIMESTAMP WITH LOCAL TIME ZONE に変更できます。すべての TIMESTAMP WITH LOCAL TIME ZONE を DATE 列に変更できます。

注意： TIMESTAMP WITH LOCAL TIME ZONE 列を DATE 列に変更すると、秒の小数部およびタイムゾーンで調整されたデータが失われます。

- TIMESTAMP WITH LOCAL TIME ZONE データに秒の小数部がある場合、秒の小数部を丸めて列の行データが更新されます。
- TIMESTAMP WITH LOCAL TIME ZONE データに 60 以上の分フィールドがある場合（夏時間の規則が切り換えられた場合に境界で発生）、その分フィールドから 60 を引いて列の行データが更新されます。

表が空の場合、日時列または期間列の先行フィールドまたは秒の小数部を増やすことも減らすこともできます。表が空でない場合、日時列または期間列の先行フィールドまたは秒の小数部を増やすことのみできます。

LONG 列は CLOB または NCLOB 列に、LONG RAW 列は BLOB 列に変更できます。

- 変更された LOB 列は、元の LONG 列で定義されたすべての制約およびトリガーを継承します。いずれかの制約を変更する場合、後続の ALTER TABLE 文で変更する必要があります。
- ドメイン索引が LONG 列で定義されている場合、列を LOB に変更する前に削除する必要があります。
- 変更後、表のすべての列にある他のすべての索引を再構築する必要があります。

参照：

- LONG から LOB への移行の詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』を参照してください。
- 索引の削除および再構築の詳細は、10-64 ページの「ALTER INDEX」を参照してください。

CHAR および VARCHAR2 列の場合、CHAR（元々バイトで指定されていた列に対するキャラクタ・セマンティクス）または BYTE（元々文字で指定されていた列に対するバイト・セマンティクス）を指定すると、長さセマンティクスを変更できます。既存の列の長さを確認するには、ALL_TAB_COLUMNS、USER_TAB_COLUMNS または DBA_TAB_COLUMNS データ・ディクショナリ・ビューの CHAR_USED 列を問い合せます。

参照：

- バイト・セマンティクスおよびキャラクタ・セマンティクスの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。
- データ・ディクショナリ・ビューの詳細は、『Oracle Database リファレンス』を参照してください。

ENCRYPT *encryption_spec* | DECRYPT　この句は、暗号化された列の復号化、暗号化されていない列の暗号化、または暗号化された列の SALT オプションを変更するために使用します。

encryption_spec を指定して既存の列を暗号化する場合は、同じ表内で暗号化されている他の列の暗号化仕様と一致させることが必要です。*encryption_spec* の詳細や制限事項は、16-26 ページの「CREATE TABLE」の encryption_spec 句を参照してください。

新規または既存の列が LOB 列の場合は、列を SecureFile LOB として格納する必要があり、SALT オプションは指定できません。

参照： 「データの暗号化例 :」（12-78 ページ）

inline_constraint　この句を使用すると、変更する列に対する制約を追加できます。既存の列に設定されている既存の制約の状態を変更する場合は、*constraint_clauses* を使用します。

LOB_storage_clause *LOB_storage_clause* は、LONG 列を LOB 列に変換する場合のみ、*modify_col_properties* 内で使用できます。この場合のみ、*LOB_storage_clause* を使用して列に対する LOB 記憶域を指定できます。ただし、*LOB_item*。*LOB_storage_clause* で省略したすべての属性には、デフォルトの LOB 記憶域属性が適用されます。

alter_XMLSchemas_clause この句は、XMLType 表の *modify_col_properties* 内でのみ有効です。詳細は、12-43 ページの「alter_XMLSchemas_clause」を参照してください。

列のプロパティの変更の制限事項： 列のプロパティの変更には、次の制限事項があります。

- LOB 列のデータ型は変更できません。
- 列にドメイン索引が定義されている場合は、表の列を変更できません。最初にドメイン索引を削除してから列を変更する必要があります。
- 表または索引のパーティション化キーまたはサブパーティション化キーの一部である列の長さまたはデータ型は変更できません。
- CHAR 型の列を VARCHAR2（または VARCHAR）型に変更または VARCHAR2（または VARCHAR）型の列を CHAR 型に変更できるのは、BLANK_TRIMMING 初期化パラメータが TRUE に設定され、列のサイズが同じまたは増加する場合のみです。BLANK_TRIMMING 初期化パラメータが TRUE に設定されている場合、列のサイズを切捨て後のデータの最大値以上の値まで減らすこともできます。
- 表がクラスタの一部である場合は、LONG または LONG RAW 列を LOB に変更できません。LONG または LONG RAW 列を LOB に変更する場合は、同じ ALTER TABLE 文では、DEFAULT 句および *LOB_storage_clause* 以外は指定できません。
- LONG または LONG RAW 列を LOB に変更する場合のみ、*LOB_storage_clause* を *modify_col_properties* の一部として指定できます。
- 索引構成表に対して ROWID データ型の列は指定できませんが、UROWID 型の列は指定できます。
- 列のデータ型を REF に変更できません。

> **参照：** マテリアライズド・ビューの再検証の詳細は、11-2 ページの「ALTER MATERIALIZED VIEW」を参照してください。

modify_col_substitutable

この句を使用すると、既存のオブジェクト型列の代替性を設定または変更できます。

FORCE キーワードを指定すると、型 ID 情報またはサブタイプ属性に関するデータが含まれる非表示列が削除されます。オブジェクト型の列または属性が FINAL でない場合、FORCE を指定する必要があります。

列の代替性の変更の制限事項： 列の代替性の変更には、次の制限事項があります。

- ALTER TABLE 文には、この句を 1 度のみ指定できます。
- オブジェクト表自体の代替性が設定されている場合、オブジェクト表の列の代替性は変更できません。
- IS OF TYPE 構文を使用して列を作成または追加した場合は、この句を指定できません。この構文によって、オブジェクト列や属性で使用できるサブタイプの範囲が、特定のサブタイプに制限されます。IS OF TYPE 構文の詳細は、16-36 ページの「CREATE TABLE」の「substitutable_column_clause」を参照してください。
- 列の属性にネストしたオブジェクト型（FINAL 以外）が含まれる場合、FORCE を指定しても、VARRAY 列を NOT SUBSTITUTABLE に変更できません。

drop_column_clause

drop_column_clause を使用すると、不要になった列を削除したり、将来、システム・リソースへの要求が少なくなったときに削除するように列にマークを付けることによって、データベースの領域を解放できます。

- ネストした表の列を削除すると、その記憶表も削除されます。
- LOB 列を削除すると、LOB データおよび対応する LOB 索引セグメントも削除されます。
- BFILE 列を削除すると、その列に格納されたロケータのみ削除され、ロケータによって参照されるファイルは削除されません。
- INCLUDING 列として定義した列を削除（または未使用とマーク）すると、この列の直前に格納された列が新しい INCLUDING 列になります。

SET UNUSED 句

SET UNUSED を使用すると、1 つ以上の列が未使用としてマーク付けされます。内部ヒープ構成表の場合、この句を指定すると、対象の列は表の各行から実際には削除されません。これらの列が使用しているディスク領域はリストアされません。このため、応答時間は DROP 句を使用した場合よりも短縮されます。

外部表内の列に対してこの句を指定すると、句は透過的に ALTER TABLE ... DROP COLUMN 文に変換されます。その理由は、外部表に対する操作はメタデータのみの操作であるため、2 つのコマンドのパフォーマンスに違いがないためです。

UNUSED のマークが付いた列を持つすべての表は、データ・ディクショナリ・ビュー USER_UNUSED_COL_TABS、DBA_UNUSED_COL_TABS および ALL_UNUSED_COL_TABS で参照できます。

> **参照：** データ・ディクショナリ・ビューの詳細は、『Oracle Database リファレンス』を参照してください。

未使用列のデータは表の行に残っていますが、この列は削除されたものとして扱われます。UNUSED のマークが付けられた列にはアクセスできなくなります。SELECT * 問合せでも、未使用列からデータを取り出すことはできません。また、UNUSED のマークが付けられた列の名前および型は、DESCRIBE コマンドでは表示されず、未使用列と同じ名前の新しい列を表に追加できます。

注意： これらの列を実際に削除するまでは、表当たり 1000 列の制限に対して、これらの列もカウント対象になります。ただし、すべての DDL 文と同様に、この句の結果をロールバックすることはできません。SET UNUSED 列を取り出すために、対応する SET USED を発行することはできません。1000 列の制限の詳細は、16-6 ページの「CREATE TABLE」を参照してください。

また、LONG 列に UNUSED のマークを付けた場合、この未使用の LONG 列を実際に削除しないかぎり、その表には別の LONG 列を追加できません。

DROP 句

DROP を指定すると、表のそれぞれの行から、対象となる列に関連付けられた列記述子およびデータを削除できます。特定の列を明示的に削除した場合、対象の表で UNUSED のマークが付いている列もすべて同時に削除されます。

列データを削除した場合、次のものが削除されます。

- 対象の列に定義されているすべての索引。
- 対象の列を参照しているすべての制約。
- 対象の列に統計タイプが関連付けられている場合、FORCE オプションによって、関連付けは解除され、その統計タイプを使用して収集したすべての統計情報は削除されます。

注意： 対象の列が対象でない列の親キーである場合または CHECK 制約が対象である列と対象でない列の両方を参照している場合は、Oracle Database はエラーを戻し、CASCADE CONSTRAINTS 句を指定しないかぎり、列を削除しません。この句を指定した場合、対象である列を参照しているすべての制約が削除されます。

参照： 統計タイプの関連付けの解除方法の詳細は、17-30 ページの「DISASSOCIATE STATISTICS」を参照してください。

DROP UNUSED COLUMNS 句

DROP UNUSED COLUMNS を指定すると、未使用とマークされているすべての列を表から削除できます。表の未使用の列からディスク領域を回収する場合に、この文を使用します。表に未使用の列がない場合でも、エラーは戻されません。

column 未使用として設定または削除する 1 つ以上の列を指定します。列を 1 つのみ指定する場合にかぎり、COLUMN キーワードを使用します。列リストを指定する場合、リストには重複する列を指定できません。

CASCADE CONSTRAINTS CASCADE CONSTRAINTS を指定すると、削除する列に定義されている主キーおよび一意キーを参照する外部キー制約をすべて削除したり、削除する列に定義されているすべての複数列制約を削除することができます。他の表の列、または対象である表の他の列が参照している制約がある場合は、CASCADE CONSTRAINTS を指定する必要があります。CASCADE CONSTRAINTS を指定しない場合、その文は異常終了し、エラーが戻されます。

INVALIDATE INVALIDATE キーワードはオプションです。ビュー、トリガー、ストアド・プログラム・ユニットなどのすべての依存オブジェクトが自動的に無効になります。オブジェクトの無効化は再帰的プロセスです。したがって、すべての直接的な依存オブジェクトおよび間接的な依存オブジェクトが無効になります。ただし、データベースでは、リモート依存性をローカル依存性と別に管理しているため、ローカル依存性のみが無効になります。

この文によって無効となったオブジェクトは、次に参照された際に自動的に再検証されます。オブジェクトを参照する前に、オブジェクトに存在するエラーは、すべて修正しておく必要があります。

参照： 依存性の詳細は、『Oracle Database 概要』を参照してください。

CHECKPOINT CHECKPOINT を指定すると、*integer* 行を処理した後に DROP COLUMN 操作のチェックポイントが適用されます。*integer* はオプションであり、1 以上である必要があります。*integer* が表の行数より大きい場合、すべての行が処理された後にチェックポイントが適用されます。*integer* を指定しない場合、デフォルトの 512 が設定されます。チェックポイントは、DROP COLUMN 操作中に蓄積された UNDO ログの量を削減し、UNDO 領域の不足を回避します。ただし、チェックポイントが適用された後にこの文が中断された場合、表は使用禁止の状態のままになります。表が使用できない間、その表に対して実行可能な操作は、DROP TABLE、TRUNCATE TABLE および ALTER TABLE DROP ... COLUMNS CONTINUE（後述）のみです。

この句は列データを削除しないため、SET UNUSED と同時に使用できません。

DROP COLUMNS CONTINUE 句

DROP COLUMNS CONTINUE を指定すると、中断されたところから列削除操作を続行できます。表が無効な状態にあるときにこの文を発行すると、エラーになります。

列の削除の制限事項： 列の削除には、次の制限事項があります。

- この句の各部分は、文の中で 1 回のみ指定でき、他の ALTER TABLE 句と同時に使用することはできません。たとえば、次のような文は許可されません。

```
ALTER TABLE t1 DROP COLUMN f1 DROP (f2);
ALTER TABLE t1 DROP COLUMN f1 SET UNUSED (f2);
ALTER TABLE t1 DROP (f1) ADD (f2 NUMBER);
ALTER TABLE t1 SET UNUSED (f3)
   ADD (CONSTRAINT ck1 CHECK (f2 > 0));
```

- オブジェクト型の列は、エンティティとしてのみ削除できます。オブジェクト型の列から属性を削除するには、ALTER TYPE ... DROP ATTRIBUTE 文を CASCADE INCLUDING TABLE DATA 句とともに使用します。属性の削除は、すべての依存オブジェクトに影響することに注意してください。詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。
- 索引構成表からは、主キー列でない場合にかぎり、列を削除できます。索引構成表の主キー制約は削除できないため、CASCADE CONSTRAINTS を指定しても主キー列は削除できません。
- 削除した列または未使用の列を含む表をエクスポートできます。ただし、エクスポート・ファイルに指定されたすべての列が表に存在する（これらの列のいずれも削除または未使用のマークを付けられていない）場合のみ、その表をインポートできます。そうでない場合は、エラーが戻ります。
- ダイレクト・ロード操作のために圧縮された表の列を未使用として設定しますが、その列を削除することはできません。ただし、*drop_column_clause* のすべての句は、すべての操作のために圧縮された表に対して有効です。詳細は、16-31 ページの「table_compression」のセマンティクスを参照してください。
- ドメイン索引が構築されている列は削除できません。
- REF 列からは SCOPE 表制約および WITH ROWID 制約を削除できません。
- 次のものは、この句を使用して削除できません。
  - 疑似列、クラスタ列またはパーティション列。パーティションが作成されたすべての表領域がオンラインで読取り / 書込みモードである場合、パーティション表から非パーティション列を削除できます。
  - ネストした表の列、オブジェクト表の列または SYS が所有する表の列。

  参照： 「列の削除例 :」（12-74 ページ）

rename_column_clause

rename_column_clause を使用すると、*table* の列名を変更できます。新しい列には、*table* 内の他の列と同じ名前を指定しないでください。

列名を変更すると、依存オブジェクトは次のように処理されます。

- 名前が変更された列に依存するファンクション索引および CHECK 制約は、引き続き有効です。
- 依存ビュー、トリガー、ファンクション、プロシージャおよびパッケージは無効になります。これらのオブジェクトが次にアクセスされる際に、妥当性チェックが再度実行されます。妥当性チェックが正常に実行されない場合は、新しい列名を使用してこれらのオブジェクトを変更する必要があります。
- 名前を変更する列にドメイン索引を定義している場合、ODCIIndexAlter メソッドが RENAME オプション付きで起動されます。この操作によって、索引タイプ・メタデータと実表の間の対応が確立されます。

列名の変更の制限事項 : 列名の変更には、次の制限事項があります。

- 同じ文の中で、この句を他の *column_clauses* と同時に使用することはできません。
- 結合索引の定義に使用される列名は変更できません。列名を変更する場合、索引を削除し、列名を変更してから、索引を再作成する必要があります。

参照 : 「列名の変更例 :」(12-74 ページ)

modify_collection_retrieval

modify_collection_retrieval を使用すると、データベースからコレクション型の項目が取り出されたときの戻り値を変更できます。

collection_item ネストした表型または VARRAY 型の列修飾属性の名前を指定します。

RETURN AS 問合せの結果として何を戻り値とするかを指定します。

- LOCATOR は、ネストした表に対して一意のロケータを戻すことを指定します。
- VALUE は、ネストした表のコピーをそのまま戻すことを指定します。

参照 : 「コレクションの取出し例 :」(12-73 ページ)

modify_LOB_storage_clause

modify_LOB_storage_clause を使用すると、*LOB_item* の物理属性を変更できます。各 *modify_LOB_storage_clause* に対して、*LOB_item* を 1 つのみ指定できます。

以降の項では、modify_LOB_parameters に固有のパラメータのセマンティクスについて説明します。この項で特に指定がない場合は、残りの LOB パラメータのセマンティクスは、表の作成時と表の変更時で同じです。詳細は、この項の最後にある制限と、16-38 ページの「CREATE TABLE」の句「LOB_storage_parameters」を参照してください。

注意 : ALTER TABLE 文によって、または DBMS_REDEFINITION パッケージを使用したオンライン再定義によって、LOB 記憶域を変更できます。作成時に LOB 暗号化、圧縮または重複除外を有効にしなかった場合は、作成後にオンライン再定義を使用してこれらを有効にすることをお薦めします。この処理は、これら 3 つのパラメータの変更にとって、ディスク領域がより効率的であるためです。DBMS_REDEFINITION の詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

PCTVERSION *integer* この句の詳細は、16-38 ページの「CREATE TABLE」の句「PCTVERSION integer」を参照してください。

LOB_retention_clause データベースが自動 UNDO モードで稼働している場合、旧バージョンの LOB を保持するには、PCTVERSION ではなく RETENTION を指定します。この句によって、PCTVERSION のこれまでの設定が上書きされます。

FREEPOOLS *integer* データベースが自動 UNDO モードで稼働している場合、BasicFile LOB に対してこの句を使用すると LOB の空きリスト・グループ数を指定できます。この句によって、FREELIST GROUPS のこれまでの設定が上書きされます。このパラメータの詳細は、16-39 ページの「CREATE TABLE」の句「FREEPOOLS integer」を参照してください。SecureFile LOB の場合は、データベースはこのパラメータを無視します。

REBUILD FREEPOOLS この句は、BasicFile LOB にのみ適用されます。SecureFile LOB には適用されません。REBUILD FREEPOOLS 句を指定すると、LOB 列から古いバージョンのデータがすべて削除されます。この句は、LOB セグメント内に保持されている古いバージョンの領域をすべて削除し、その領域を新しい LOB データがすぐに使用できるように解放する場合に有効です。

LOB_deduplicate_clause この句は、SecureFile LOB に対してのみ有効です。KEEP_DUPLICATES は、LOB 重複除外を無効にします。DEDUPLICATE は、LOB 重複除外を有効にします。セグメント内のすべての LOB が読み取られ、一致する LOB がある場合は重複が除外されてから戻されます。

LOB_compression_clause この句は、SecureFile LOB に対してのみ有効です。COMPRESS は、セグメント内のすべての LOB を圧縮して戻します。NOCOMPRESS は、セグメント内のすべての LOB を圧縮解除して戻します。

ENCRYPT | DECRYPT LOB 暗号化は、一般に列の暗号化と同じセマンティクスを持ちます。詳細は、12-44 ページの「ENCRYPT encryption_spec | DECRYPT」を参照してください。

CACHE、NOCACHE、CACHE READS CACHE または NOCACHE から CACHE READS へ、または CACHE READS から CACHE または NOCACHE へ LOB 列を変更するときに、ロギング属性を変更できます。LOGGING または NOLOGGING を指定しない場合、LOB 列の現行ロギング属性がデフォルトになります。CACHE、NOCACHE または CACHE READS を指定しない場合、Oracle Database は、LOB 属性の既存の値を保持します。

LOB 記憶域の変更の制限事項： LOB 記憶域の変更には、次の制限事項があります。

- LOB 記憶域属性を変更する場合、*storage_clause* の INITIAL パラメータの値は変更できません。
- 同じ文で *allocate_extent_clause* と *deallocate_unused_clause* の両方を指定することはできません。
- PCTVERSION パラメータと RETENTION パラメータの両方を指定することはできません。
- *shrink_clause* は、SecureFile LOB に対して指定できません。

> **参照：** LOB パラメータの設定の詳細は、16-37 ページの「CREATE TABLE」の「LOB_storage_clause」および 12-79 ページの「LOB 列の例 :」を参照してください。

alter_varray_col_properties

alter_varray_col_properties を使用すると、VARRAY が格納されている既存の LOB の記憶特性を変更できます。

VARRAY 列のプロパティ変更の制限事項： *LOB_parameters* の TABLESPACE 句は、この句の一部として指定できません。VARRAY に対する LOB 表領域のデフォルトは、表を含む表領域になります。

REKEY *encryption_spec*

REKEY 句は、データベースで新しい暗号化キーを生成します。表内の暗号化されている列はすべて、新しいキーを使用して再度暗号化されます。その際 *encryption_spec* の USING 句を指定すると、新しい暗号化アルゴリズムが使用されます。この ALTER TABLE 文で、この句と別の句を組み合せて使用することはできません。

> **参照：** 列の透過的な暗号化の詳細は、『Oracle Database Advanced Security 管理者ガイド』を参照してください。

constraint_clauses

constraint_clauses を使用すると、アウトライン宣言を使用して新しい制約を追加、既存の制約の状態を変更、および制約を削除できます。アウトライン制約および *constraint_state* のすべてのキーワードとパラメータの詳細は、8-4 ページの「constraint」を参照してください。

制約の追加

ADD 句を使用すると、アウトライン制約またはアウトライン REF 制約を表に追加できます。

> **参照：** 12-74 ページの「CHECK 制約を使用禁止にする例 :」、12-77 ページの「オブジェクト識別子の指定例 :」および 12-81 ページの「REF 列の例 :」を参照してください。

制約の変更

MODIFY CONSTRAINT 句を使用すると、既存の制約の状態を変更できます。

制約変更の制限事項： 制約の変更には、次の制限事項があります。

- NOT DEFERRABLE 制約は、INITIALLY DEFERRED には変更できません。
- 索引構成表にこの句を指定した場合、同じ文では他の句を指定できません。
- 参照パーティション表の外部キー列の NOT NULL 制約は変更できません。また、参照パーティション表の参照制約のパーティション化状態は変更できません。

> **参照：** 「制約状態の変更例 :」（12-73 ページ）

制約名の変更

RENAME CONSTRAINT 句を使用すると、*table* の既存の制約名を変更できます。新しい制約名は、同一スキーマ内にあるオブジェクトの既存の制約と同じ名前にはできません。制約に依存するすべてのオブジェクトは、引き続き有効です。

> **参照：** 「制約名の変更例 :」（12-79 ページ）

drop_constraint_clause

drop_constraint_clause を使用すると、データベースの整合性制約を削除できます。制約の適用を中止し、データ・ディクショナリから制約が削除されます。
各 *drop_constraint_clause* には、制約を 1 つのみ指定できますが、1 つの文の中では、複数の *drop_constraint_clauses* を指定できます。

制約削除の制限事項： 参照パーティション表の外部キー列の NOT NULL 制約は削除できません。また、参照パーティション表の参照制約のパーティション化は削除できません。

PRIMARY KEY PRIMARY KEY を指定すると、*table* の主キー制約を削除できます。

UNIQUE UNIQUE を指定すると、指定した列の一意制約を削除できます。

ビットマップ結合索引が定義されている列から主キー制約または一意制約を削除すると、索引は無効になります。ビットマップ結合索引の詳細は、14-50 ページの「CREATE INDEX」を参照してください。

CONSTRAINT CONSTRAINT *constraint* を指定すると、主キー制約または一意制約以外の整合性制約を削除できます。

CASCADE CASCADE を指定すると、削除する整合性制約に依存するその他の整合性制約もすべて削除できます。

KEEP INDEX | DROP INDEX KEEP INDEX または DROP INDEX を指定すると、主キーまたは一意制約の適用に使用する索引を残すか削除するかを指定できます。

制約削除の制限事項： 制約の削除には、次の制限事項があります。

- 参照整合性制約の一部の主キー制約または一意キー制約は、外部キーを削除しないと削除できません。参照されたキーと外部キーをともに削除する場合は、CASCADE 句を使用してください。CASCADE を省略すると、外部キーによって参照される主キー制約および一意制約は削除されません。
- 主キーをオブジェクト識別子（OID）として使用している表では、主キー制約は（CASCADE 句を使用しても）削除できません。
- REF 列の参照整合性制約を削除した場合、REF 列の有効範囲には参照先の表が含まれたままになります。
- REF 列の有効範囲は削除できません。

参照： 「制約の削除例 :」（12-79 ページ）

alter_external_table

alter_external_table 句を使用すると、外部表の特性を変更できます。この句は、外部データには影響しません。*parallel_clause*、*enable_disable_clause*、*external_data_properties* および REJECT LIMIT 句の構文およびセマンティクスは、CREATE TABLE の内容と同じです。16-34 ページの「CREATE TABLE」の「external_table_clause」を参照してください。

PROJECT COLUMN 句 この句を使用すると、後続の問合せで、アクセス・ドライバが外部表に含まれる行を検証する方法を指定できます。デフォルトは PROJECT COLUMN ALL で、この場合、アクセス・ドライバは、選択されている列にかかわらず、すべての列の値を処理し、列のエントリが有効な行のみを検証します。いずれかの列の値でデータ型変換エラーなどのエラーが発生すると、その列が SELECT 構文のリスト内で参照されていない場合でも、行が拒否されます。PROJECT COLUMN REFERENCED を指定すると、アクセス・ドライバは SELECT 構文のリスト内の列のみを処理します。

ALL 設定の場合、一貫した結果セットが保証されます。REFERENCED 設定の場合、後続の問合せで参照される列の数に応じて、戻される行の数が異なる可能性がありますが、ALL 設定よりも高速です。後続の問合せで外部表のすべての列が選択されている場合、2 つの設定の動作は同じになります。

外部表の変更の制限事項： 外部表の変更には、次の制限事項があります。

- この句以外の句を使用して外部表を変更できません。
- LONG、VARRAY またはオブジェクト型の列は外部表に追加できません。また、外部表の列のデータ型を、これらの型に変更できません。
- 制約は外部表に追加できません。
- 外部表の記憶域パラメータは変更できません。

alter_table_partitioning

ここで説明する句は、パーティション表にのみ適用されます。1 つの ALTER TABLE 文の中では、パーティション操作を他のパーティション操作または実表での操作と組み合せて使用することはできません。

表パーティションの変更の制限事項： 表パーティションの変更には、次の注意事項があります。

- 1 つ以上のマテリアライズド・ビューのマスター表で、パーティションを削除、交換、切捨て、移動、変更または分割した場合、その表に関する大量の既存ロード情報が削除されます。したがって、前述の操作のいずれかを行う前に、必ず依存するマテリアライズド・ビューをリフレッシュしてください。
- *table* にビットマップ結合索引が定義されている場合、*table* のパーティションの変更操作を実行すると、索引に UNUSABLE のマークが付けられます。

- 参照パーティション表に指定できる *alter_table_partitioning* 句は、*modify_table_default_attrs*、*move_table_[sub]partition*、*truncate_partition_subpart* および *exchange_partition_subpart* のみです。これらの操作は、参照パーティション表の子表に対してカスケードしません。これ以外のパーティション・メンテナンス操作は、参照パーティション表に対しては有効ではありませんが、参照パーティション表の親表に対しては指定できます。操作は、子の参照パーティション表にカスケードします。
- パーティションまたはサブパーティションを追加する際、表ごとに指定できるパーティションとサブパーティションの合計は 1024K-1 です。
- 表のパーティションまたはサブパーティションを追加するときにパーティション名を省略すると、16-44 ページの「一般的なパーティション化の注意事項 :」で説明されているルールに従い、データベースによって名前が生成されます。

関連する CONTEXT ドメイン索引が含まれる表におけるパーティション操作の詳細は、『Oracle Text リファレンス』を参照してください。

ブロック・サイズが異なる表領域のパーティション化されたデータベース・エンティティの記憶域には、制限事項があります。これらの制限事項については、『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。

modify_table_default_attrs

modify_table_default_attrs を使用すると、*table* の属性に対する新しいデフォルト値を指定できます。文に指定した属性のみが影響を受けます。その後に作成するパーティションおよび LOB パーティションは、パーティションまたは LOB パーティションの作成時に明示的に上書きしないかぎり、この値を継承します。既存のパーティションおよび LOB パーティションは、この句の影響を受けません。

文の中で指定した属性のみが影響を受けます。指定されたデフォルト値は、個々のパーティションまたは LOB パーティションのレベルで指定された属性で上書きされます。

- FOR *partition_extended_name* は、コンポジット・パーティション表にのみ適用されます。この句は、*partition_extended_name* で指定されたパーティションの属性に新しいデフォルト値を指定します。その後に作成するパーティションのサブパーティションおよび LOB パーティションは、サブパーティションまたは LOB パーティションの作成時に明示的に上書きしないかぎり、この値を継承します。既存のサブパーティションは、この句の影響を受けません。
- PCTTHRESHOLD、*key_compression* および *alter_overflow_clause* は、パーティション化された索引構成表にのみ有効です。
- 表レベルでキー圧縮がすでに指定されている場合にかぎり、キー圧縮を指定できます。また、COMPRESS キーワードの後に int 型を指定することはできません。キー圧縮の長さは、表の作成時にのみ指定できます。
- 索引構成表の索引セグメントに対しては、*segment_attributes* 句で PCTUSED パラメータを指定できません。

alter_interval_partitioning

次の場合にこの句を使用します。

- 既存のレンジ・パーティション表を時間隔パーティションに変換する場合。データベースにより、最後のレンジ・パーティションの最大値を超えるデータの必要に応じて、指定した数値範囲または日時間隔のパーティションが自動的に作成されます。
- 既存の時間隔パーティション表の時間隔を変更する場合。データベースにより、時間隔パーティションがレンジ・パーティションに変換され、最後のレンジ・パーティションの最大値を超えるデータの必要に応じて、指定した数値範囲または日時間隔のパーティションが自動的に作成されます。
- 既存の時間隔パーティション表の表領域の記憶域を変更する場合。

- 時間隔パーティション表をレンジ・パーティション表に戻す場合。SET INTERVAL () を使用して、時間隔パーティションを無効にします。データベースは、作成されるレンジ・パーティションの上限として、作成済の時間隔パーティションの上限を使用し、既存の時間隔パーティションをレンジ・パーティションに変換します。

expr には、有効な数値または期間式を指定します。

> **参照：** 時間隔パーティションの詳細は、16-45 ページの「CREATE TABLE」の「INTERVAL 句」および『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。

set_subpartition_template

set_subpartition_template 句を使用すると、表パーティションに対するデフォルトのレンジ、リストまたはハッシュ・サブパーティション定義を作成したり、既存の定義を置き換えることができます。この句は、コンポジット・パーティション表に対してのみ有効です。この句によって、既存のサブパーティション・テンプレートは置き換えられます。既存のテンプレートがない場合は新規のテンプレートが作成されます。既存のサブパーティション、ローカル索引およびグローバル索引は影響を受けません。ただし、その後のパーティション化操作（追加操作やマージ操作）では、新規テンプレートが使用されます。

既存のサブパーティション・テンプレートを削除するには、ALTER TABLE *table* SET SUBPARTITION TEMPLATE () を指定します。

> **注意：** サブパーティションのテンプレートに対して表領域の記憶域を指定しても、*table* のパーティションに対して明示的に指定した表領域の記憶域は上書きされません。サブパーティションに対して表領域の記憶域を指定するには、次のいずれかの操作を実行します。
>
> - パーティション・レベルでの表領域の記憶域を省略し、サブパーティションのテンプレートに対して表領域の記憶域を指定します。
> - 固有の表領域の記憶域を持つサブパーティションを個別に定義します。

サブパーティションのテンプレートの制限事項： 16-51 ページの「CREATE TABLE」の「サブパーティションのテンプレートの制限事項：」を参照してください。

modify_table_partition

modify_table_partition 句を使用すると、レンジ・パーティション、ハッシュ・パーティション、リスト・パーティションまたはシステム・パーティションの実際の物理属性を変更できます。そのパーティションの 1 つ以上の LOB 項目の記憶域属性を任意に変更できます。物理属性（制限事項については後述）、ロギングおよび記憶域パラメータに対して、新しい値を指定できます。

すべてのタイプのパーティションについて、パーティションを変更した結果、使用不可能になったローカル索引の処理方法も指定できます。12-68 ページの「UNUSABLE LOCAL INDEXES 句」を参照してください。

パーティション化された索引構成表の場合、パーティションの変更時にマッピング表も同時に更新できます。12-39 ページの「alter_mapping_table_clauses」を参照してください。

表パーティションの変更の制限事項： レンジ、リストおよびハッシュ表パーティションの操作に、次の制限事項が適用されます。

- すべてのタイプの表パーティションについて、*partition_attributes* 句で *shrink_clause* を使用すると、各パーティションのセグメントを縮小できます。この句の詳細は、12-35 ページの「shrink_clause」を参照してください。

- システム・パーティションを変更する構文およびセマンティクスは、ハッシュ・パーティションを変更する場合と同じです。詳細は、12-56 ページの「modify_hash_partition」を参照してください。
- *table* がコンポジット・パーティション化されている場合 :
  - *allocate_extent_clause* を指定すると、*partition* のそれぞれのサブパーティションにエクステントが割り当てられます。
  - *deallocate_unused_clause* を指定すると、*partition* のそれぞれのサブパーティションから未使用の記憶域の割当てが解除されます。
  - この句で変更された他の属性は、*partition* のサブパーティションでも変更され、既存の値は上書きされます。既存のサブパーティションの属性が変更されないようにするには、*modify_table_default_attrs* の FOR PARTITION 句を使用します。
- 特に指定がないかぎり、*partition_attributes* の残りの句の動作は、パーティション表の作成時と同じです。詳細は、16-44 ページの「CREATE TABLE」の「table_partitioning_clauses」を参照してください。

  参照 : 「表のパーティションの変更例 :」(12-76 ページ)

modify_range_partition

この句を使用すると、レンジ・パーティションの特性を変更できます。

add_range_subpartition この句は、レンジ - レンジ・コンポジット・パーティションに対してのみ有効です。レンジ・サブパーティションを *partition* に追加できます。

add_hash_subpartition この句は、レンジ - ハッシュ・コンポジット・パーティションに対してのみ有効です。*add_hash_subpartition* 句を使用すると、ハッシュ・サブパーティションを *partition* に追加できます。Oracle Database は、ハッシュ・ファンクションによって *partition* の他のサブパーティションから再ハッシュされた行を、新しいサブパーティションに移入します。ロード・バランシングを最適化する場合、サブパーティションの合計数は 2 の累乗にする必要があります。

partitioning_storage_clause でサブパーティションに対して指定できる句は、TABLESPACE 句のみです。TABLESPACE を指定しなかった場合、新しいサブパーティションは *partition* のデフォルトの表領域に格納されます。

選択したパーティションに対応するローカル索引パーティションが追加されます。

追加したパーティションに対応するローカル索引パーティションに UNUSABLE のマークが付けられます。ヒープ構成表のすべての索引が無効になります。update_index_clauses を使用し、操作中にこれらの索引を更新できます。

add_list_subpartition この句は、レンジ - リストおよびリスト - リスト・コンポジット・パーティションに対してのみ有効です。リスト・サブパーティションを *partition* に追加できます。ただし、DEFAULT サブパーティションを作成していない場合にのみです。

- この操作には *list_values_clause* が必要です。また、*list_values_clause* には、*partition* のその他のサブパーティションには存在していない値を指定する必要があります。ただし、他のパーティションのサブパーティションで使用されている値は指定できます。
- *partitioning_storage_clause* でサブパーティションに対して指定できる句は、TABLESPACE 句および表の圧縮のみです。

表のすべてのローカル索引パーティションには、同じ値リストを持つサブパーティションも追加されます。表の既存のローカル索引パーティションおよびグローバル索引パーティションは影響を受けません。

リスト・サブパーティションの追加の制限事項： パーティションにデフォルト・サブパーティションがすでに作成されている場合は、この句を指定できません。その場合、*split_list_subpartition* 句を使用して、デフォルト・パーティションを分割する必要があります。

COALESCE SUBPARTITION COALESCE SUBPARTITION は、ハッシュ・サブパーティションにのみ適用されます。COALESCE SUBPARTITION を使用すると、最後のハッシュ・サブパーティションが選択され、その内容が 1 つ以上の残りのサブパーティション（ハッシュ・ファンクションが決定）に分散された後、選択されたサブパーティションが削除されます。

- 選択したパーティションに対応するローカル索引パーティションが削除されます。
- 1 つ以上の結合パーティションに対応するローカル索引パーティションに UNUSABLE のマークが付けられます。ヒープ構成表のすべてのグローバル索引が無効になります。update_index_clauses を使用し、操作中にこれらの索引を更新できます。

modify_hash_partition

partition_attributes でハッシュ・パーティションを変更する場合、*allocate_extent_clause* および *deallocate_unused_clause* のみを指定できます。パーティションのその他の属性は、表レベルのデフォルトから継承されます。ただし、TABLESPACE は、作成時と同じ状態です。

modify_list_partition

リスト・パーティションを変更するときに使用できる句のセマンティクスは、レンジ・パーティションを変更するときと同じです。リスト・パーティションを変更する場合、次の句も使用できます。

ADD | DROP VALUES 句 これらの句は、コンポジット・パーティションを変更する場合のみ有効です。これらの句によって、表のローカル索引とグローバル索引が影響を受けることはありません。

- ADD VALUES 句を使用すると、*partition* の *partition_value* リストが拡張され、追加した値が含まれます。追加するパーティションの値は、16-48 ページの「CREATE TABLE」の句 list_partitions に示すすべての規則および制限事項に準拠する必要があります。
- DROP VALUES 句を使用すると、1 つ以上の *partition_value* が削除され、*partition* の *partition_value* リストが縮小されます。この句を指定すると、Oracle Database はこの値の行が存在しないことを検証します。そのような行が存在する場合は、エラーが戻されます。

注意： 表にローカル同一キー索引が定義されている場合、デフォルトのリスト・パーティションのある表では ADD VALUES 操作および DROP VALUES 操作が改善されます。

リスト値の追加と削除の制限事項： リスト値の追加と削除には、次の制限事項があります。

- デフォルトのリスト・パーティションに対して値を追加または値を削除することはできません。
- *table* にデフォルト・パーティションが定義されている場合、デフォルト・パーティション以外に値を追加しようとすると、その値がデフォルト・パーティションに存在しているかどうかが確認されます。デフォルト・パーティションにその値が存在する場合、エラーが戻されます。

modify_table_subpartition

この句は、コンポジット・パーティション表にのみ適用されます。その副次句によって、個別のレンジ、リストまたはハッシュ・サブパーティションの特性を変更できます。

shrink_clause を使用すると、サブパーティションの各セグメントを縮小できます。この句の詳細は、12-35 ページの「shrink_clause」を参照してください。

また、パーティションを変更した結果、使用不可能になったローカル索引の処理方法も指定できます。12-68 ページの「UNUSABLE LOCAL INDEXES 句」を参照してください。

ハッシュ・サブパーティションの変更の制限事項： サブパーティションに指定できる *modify_LOB_parameters* は、*allocate_extent_clause* および *deallocate_unused_clause* のみです。

ADD | DROP VALUES 句 これらの句は、リスト・サブパーティションを変更する場合のみ有効です。これらの句によって、表のローカル索引とグローバル索引が影響を受けることはありません。

- ADD VALUES 句を使用すると、*subpartition* の *partition_value* リストが拡張され、追加した値が含まれます。追加するパーティションの値は、16-48 ページの「CREATE TABLE」の句 list_partitions に示すすべての規則および制限事項に準拠する必要があります。
- DROP VALUES 句を使用すると、1 つ以上の *partition_value* が削除され、*subpartition* の *partition_value* リストが縮小されます。この句を指定すると、Oracle Database はこの値の行が存在しないことを検証します。そのような行が存在する場合は、エラーが戻されます。

また、パーティションを変更した結果、使用不可能になったローカル索引の処理方法も指定できます。12-68 ページの「UNUSABLE LOCAL INDEXES 句」を参照してください。

リスト・サブパーティションの変更の制限事項： サブパーティションに指定できる *modify_LOB_parameters* は、*allocate_extent_clause* および *deallocate_unused_clause* のみです。

move_table_partition

move_table_partition 句を使用すると、*partition* を別のセグメントへ移動できます。パーティション・データの別の表領域への移動、断片化を削減するためのデータの再クラスタ化、および作成時の物理属性の変更ができます。

表に LOB 列が含まれている場合、*LOB_storage_clause* を使用して、このパーティションに関連付けられた LOB データおよび LOB 索引セグメントを移動できます。この場合、指定した LOB のみが影響を受けます。特定の LOB 列に *LOB_storage_clause* を指定しなかった場合、その列の LOB データおよび LOB 索引セグメントは移動されません。

選択したパーティションに対応するローカル索引パーティションが、Oracle Database によって移動されます。移動したパーティションが空でない場合、UNUSABLE のマークが付けられます。ヒープ構成表のグローバル索引が無効になります。update_index_clauses を使用し、操作中にこれらの索引を更新できます。

LOB データ・セグメントを移動する場合、古いデータ・セグメントおよび対応する索引セグメントが削除され、新しい表領域を指定しない場合でも、新しいセグメントが作成されます。

移動操作では、*parallel_clause*（指定されている場合）からパラレル属性が取得されます。parallel_clause が指定されていない場合は、表のデフォルトのパラレル属性があれば、これが使用されます。いずれも指定されていない場合は、シリアルに移動が行われます。

MOVE PARTITION で *parallel_clause* を指定した場合、*table* のデフォルトのパラレル属性は変更されません。

注意： 索引構成表の場合、主キーのアドレスおよびその値を使用して、論理 ROWID が構成されます。論理 ROWID は、表の 2 次索引に格納されます。索引構成表のパーティションを移動した場合、ROWID のアドレス部分が変更され、パフォーマンスの障害になる場合があります。最適なパフォーマンスを維持するには、移動したパーティションの 2 次索引を再構築し、ROWID を更新してください。

参照： 「表のパーティションの移動例：」（12-77 ページ）

MAPPING TABLE MAPPING TABLE 句は、マッピング表が定義されている索引構成表のみに有効です。マッピング表は、索引構成表のパーティションとともに移動されます。マッピング表のパーティションは、移動した索引構成表のパーティションの物理属性を継承します。これは、マッピング表のパーティションの属性を変更する唯一の方法です。この句を省略した場合、マッピング表のパーティションでは、元の属性が保持されます。

対応するすべてのビットマップ索引パーティションには UNUSABLE のマークが付けられます。

この句の詳細は、16-33 ページの「CREATE TABLE」の「mapping_table_clauses」を参照してください。

表パーティションの移動の制限事項： 表パーティションの移動には、次の制限事項があります。

- *partition* がハッシュ・パーティションである場合、この句に TABLESPACE の属性以外は指定できません。
- この句は、サブパーティションを含むパーティションに対して指定できません。ただし、*move_table_subpartition* 句を使用してサブパーティションを移動できます。

move_table_subpartition

move_table_subpartition 句を使用すると、*subpartition* を別のセグメントへ移動できます。TABLESPACE を指定しない場合、サブパーティションは同じ表領域に残ります。

サブパーティションが空でない場合、移動したサブパーティションに対応するすべてのローカル索引サブパーティションに UNUSABLE のマークが付けられます。update_index_clauses を使用し、操作中にヒープ構成表のすべての索引を更新できます。

表に LOB 列が含まれている場合、*LOB_storage_clause* を使用して、このサブパーティションに関連付けられた LOB データおよび LOB 索引セグメントを移動できます。この場合、指定した LOB のみが影響を受けます。特定の LOB 列に *LOB_storage_clause* を指定しなかった場合、その列の LOB データおよび LOB 索引セグメントは移動されません。

LOB データ・セグメントを移動する場合、古いデータ・セグメントおよび対応する索引セグメントが削除され、新しい表領域を指定しない場合でも、新しいセグメントが作成されます。

表サブパーティションの移動の制限事項： サブパーティションの定義で指定できる *partitioning_storage_clause* の句は、TABLESPACE および *table_compression* のみです。

add_table_partition

add_table_partition 句を使用すると、*table* にハッシュ・パーティション、レンジ・パーティション、リスト・パーティションまたはシステム・パーティションを追加できます。

table で定義されているローカル索引に、実表のパーティションと同じ名前で新しいパーティションが追加されます。索引に同じ名前のパーティションがすでに存在する場合、SYS_Pn という形式でパーティションの名前が生成されます。

table が索引構成されている場合、表で定義されたすべてのマッピング表およびオーバーフロー領域に、パーティションが追加されます。

table が参照パーティション表の親表の場合は、*dependent_tables_clause* を使用して、この文に指定するパーティション・メンテナンス操作を参照パーティション表のすべての子表に伝播できます。

コンポジット・パーティション表の場合、*table* に定義されたすべてのローカル索引に対して、同じサブパーティションを持つ新規の索引パーティションが追加されます。*table* のグローバル索引が影響を受けることはありません。

参照：「LOB を持つ表パーティションの追加例：」（12-75 ページ）

add_range_partition_clause

add_range_partition_clause を使用すると、新しいレンジ・パーティションをレンジ・パーティション表またはコンポジット・レンジ・パーティション表の一番上（最後の既存のパーティションの後）に追加できます。

ドメイン索引が *table* で定義されている場合、IN_PROGRESS または FAILED のマークが付いていると無効になります。

レンジ・パーティションの追加の制限事項： レンジ・パーティションの追加には、次の制限事項があります。

- 既存の上位パーティションにある各パーティション化キーのパーティションの上限が MAXVALUE の場合、表にパーティションを追加できません。そのかわり、*split_table_partition* 句を使用して、表の始めまたは中間にパーティションを追加します。
- *key_compression* および OVERFLOW は、パーティション化された索引構成表に対してのみ有効です。パーティション表にすでにオーバーフロー・セグメントが存在する場合にかぎり、OVERFLOW を指定できます。表レベルでキー圧縮が使用可能な場合にかぎり、キー圧縮を指定できます。
- PCTUSED パラメータは、索引構成表の索引セグメントに対して指定できません。

range_values_clause 新しいパーティションの上限を指定します。*value_list* は、パーティション・キー列に対応するリテラル値を順序どおりにカンマで区切ったリストです。*value_list* の値は、表内にある既存の最上位パーティションのパーティション境界より大きくする必要があります。

table_partition_description この句を使用すると、新しいパーティションに作成時の物理属性を指定できます。表に LOB 列が含まれている場合、1 つ以上の LOB 項目にパーティション・レベルの属性を指定することもできます。

サブパーティションの定義 これらの句は、コンポジット・パーティション表に対してのみ有効です。新しいパーティションのサブパーティションを指定する場合は、*range_subpartition_desc*、*list_subpartition_desc* または *hash_subpartition_desc* を適切に使用します。この句によって、表レベルで *subpartition_template* に定義された任意のサブパーティションの設定は上書きされます。

add_hash_partition_clause

add_hash_partition_clause を使用すると、ハッシュ・パーティション表の一番上に新しいハッシュ・パーティションを追加できます。Oracle Database は、ハッシュ・ファンクションによって *table* の他のパーティションから再ハッシュされた行を、新しいパーティションに移入します。ロード・バランシングを最適化する場合、パーティションの合計数は 2 の累乗にする必要があります。

パーティション名を指定します。また、パーティションが格納される表領域を指定することもできます。名前を指定しない場合、SYS_Pn という形式でパーティション名が割り当てられます。TABLESPACE を指定しなかった場合、新しいパーティションは表のデフォルトの表領域に格納されます。他の属性は、常に、表レベルのデフォルトから継承されます。

この操作によってパーティション間でデータが再ハッシュされると、対応するすべてのローカル索引パーティションに UNUSABLE のマークが付けられます。update_index_clauses を使用し、操作中にヒープ構成表のすべての索引を更新できます。

parallel_clause を使用すると、新しいパーティションの作成をパラレル化するかどうかを指定できます。

> **参照：** ハッシュ・パーティション化の詳細は、16-6 ページの「CREATE TABLE」および『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。

add_list_partition_clause

add_list_partition_clause を使用すると、パーティションの新しい一連の値を使用して、*table* に新しいパーティションを追加できます。新しいパーティションに作成時の物理属性を指定できます。表に LOB 列が含まれている場合、1 つ以上の LOB 項目にパーティション・レベルの属性を指定することもできます。

リスト・パーティションの追加の制限事項： 表のデフォルト・パーティションがすでに定義されている場合は、リスト・パーティションを追加できません。その場合、*split_table_partition* 句を使用して、デフォルト・パーティションを分割する必要があります。

> **参照：**
>
> - リスト・パーティションの詳細および制限事項については、16-48 ページの「CREATE TABLE」の「list_partitions」を参照してください。
> - 「デフォルト・リスト・パーティションの使用例：」（12-76 ページ）

add_system_partition_clause

この句を使用すると、パーティションをシステム・パーティション表に追加できます。表に定義されているすべてのローカル索引に、対応する索引パーティションが追加されます。

BEFORE 句によって、既存のパーティションとの関係で新しいパーティションが追加される場所を指定できます。システム・パーティションは分割できません。そのため、この句は、既存の 1 つのパーティションの内容を複数の新しいパーティションに分割する場合に有効です。この句を省略した場合、新しいパーティションは既存のパーティションの後に追加されます。

table_partition_description によって、新しいパーティションのパーティション・レベルの属性を指定できます。指定しない属性の値は、表レベルの値から継承されます。

システム・パーティションの追加の制限事項： システム・パーティションを追加する場合、OVERFLOW 句は指定できません。

> **参照：** システム・パーティションの詳細は、16-52 ページの「CREATE TABLE」の句「system_partitioning」を参照してください。

coalesce_table_partition

COALESCE は、ハッシュ・パーティションにのみ適用されます。*coalesce_table_partition* 句を使用すると、最後のハッシュ・パーティションが選択され、その内容がハッシュ・ファンクションによって決定される 1 つ以上の残りのパーティションに分散された後、選択されたパーティションが削除されます。

選択したパーティションに対応するローカル索引パーティションが削除されます。1 つ以上の吸収パーティションに対応するローカル索引パーティションに UNUSABLE のマークが付けられます。ヒープ構成表のすべての索引が無効になります。update_index_clauses を使用すると、操作中にすべての索引を更新できます。

表パーティションの結合の制限事項： *update_all_indexes_clause* を使用してグローバル索引を更新する場合、副次句ではなく UPDATE INDEXES キーワードのみを指定できます。

drop_table_partition

drop_table_partition 句を使用すると、パーティション表から、*partition_extended_name* で指定されるパーティションおよびそのパーティション内のデータを削除できます。データを表に残したままパーティションを削除する場合は、そのパーティションを隣接するパーティションにマージする必要があります。

参照：「merge_table_partitions」（12-65 ページ）

- *table* に LOB 列が存在する場合、*partition* に対応する LOB データ、LOB 索引パーティションおよび（存在する場合は）サブパーティションも削除されます。
- *table* が索引構成されており、定義されたマッピング表を持つ場合、対応するマッピング表のパーティションも同様に削除されます。
- ローカル索引パーティションおよび削除されるパーティションに対応するサブパーティションは、UNUSABLE のマークが付いている場合でも削除されます。

update_index_clauses を使用し、操作中に *table* のグローバル索引を更新できます。*update_index_clauses* とともに *parallel_clause* を指定すると、削除操作ではなく、索引の更新がパラレル化されます。

レンジ・パーティションを削除し、その後、削除したパーティションに属していた行を挿入した場合、1 つ上位のパーティションに行が格納されます。ただし、そのパーティションが最上位のパーティションである場合、削除したパーティションが表していた値の範囲が表に対して無効になるため、挿入は失敗します。

表パーティションの削除の制限事項： 表パーティションの削除には、次の制限事項があります。

- ハッシュ・パーティション表のパーティションは削除できません。かわりに、*coalesce_table_partition* 句を使用してください。
- *table* にパーティションが 1 つのみ存在する場合は、このパーティションを削除できません。その表を削除する必要があります。
- update_index_clauses を使用してグローバル索引を更新する場合、副次句ではなく UPDATE INDEXES キーワードのみを指定できます。

参照：「表のパーティションの削除例 :」（12-76 ページ）

drop_table_subpartition

この句を使用すると、レンジまたはリスト・コンポジット・パーティション表からレンジ、リストまたはハッシュ・サブパーティションを削除できます。削除対象のサブパーティション内の行はすべて削除されます。

ローカル索引の対応するサブパーティションは、自動的に削除されます。その他の索引サブパーティションには影響がありません。*update_global_index_clause* または *update_all_indexes_clause* を指定しないかぎり、グローバル索引には UNUSABLE のマークが付けられます。

表サブパーティションの削除の制限事項： 表サブパーティションの削除には、次の制限事項があります。

- ハッシュ・サブパーティションは削除できません。かわりに、MODIFY PARTITION ... COALESCE SUBPARTITION 構文を使用してください。
- パーティションの最後のサブパーティションは削除できません。かわりに、*drop_table_partition* 句を使用してください。
- グローバル索引を更新する場合、*update_all_indexes_clause* のオプションの副次句を指定することはできません。

rename_partition_subpart

rename_partition_subpart 句を使用すると、表パーティションまたは表サブパーティションの名前を *new_name* に変更できます。パーティションおよびサブパーティションのどちらの場合も、*new_name* は同じ表に存在するすべてのパーティションおよびサブパーティションと異なる値である必要があります。

table が索引構成されている場合、対応する主キー索引パーティションに、既存のオーバーフロー・パーティションおよびマッピング表パーティションと同じ名前が割り当てられます。

参照： 「表のパーティション名の変更例：」（12-77 ページ）

truncate_partition_subpart

TRUNCATE PARTITION を指定すると、*partition_extended_name* で指定されるパーティションからすべての行を削除できます。表がコンポジット・パーティション化されている場合は、そのパーティションのサブパーティションからすべての行が削除されます。TRUNCATE SUBPARTITION を指定すると、個別のサブパーティションからすべての行が削除されます。*table* が索引構成されている場合、対応するすべてのマッピング表のパーティションおよびオーバーフロー領域のパーティションが切り捨てられます。

- 切り捨てるパーティションまたはサブパーティションにデータが含まれている場合は、まず、その表の参照整合性制約を使用禁止にする必要があります。また、別の方法として、行を削除してからパーティションを切り捨てる方法もあります。
- *table* に LOB 列が存在する場合、このパーティションの LOB データおよび LOB 索引セグメントも切り捨てられます。*table* がコンポジット・パーティション化されている場合、このパーティションのサブパーティションの LOB データおよび LOB 索引セグメントは切り捨てられます。
- *table* でドメイン索引が定義されている場合、索引に IN_PROGRESS または FAILED のマークが付いていると無効になります。また、切り捨てられる表パーティションに対応する索引パーティションに IN_PROGRESS のマークが付いていると無効になります。

切り捨てられるそれぞれのパーティションまたはサブパーティションでは、対応するローカル索引パーティションおよびサブパーティションも切り捨てられます。これらの索引パーティションまたはサブパーティションに UNUSABLE のマークが付いている場合、これらは切り捨てられ、UNUSABLE のマークは VALID にリセットされます。

update_global_index_clause または update_all_indexes_clause を使用し、操作中に *table* のグローバル索引を更新できます。これらの句とともに *parallel_clause* を指定すると、切捨て操作ではなく、索引の更新がパラレル化されます。

DROP STORAGE DROP STORAGE を指定すると、削除した行が占有していた領域の割当てを解除できます。解放された領域は、表領域の他のスキーマ・オブジェクトが利用できます。

REUSE STORAGE REUSE STORAGE を指定すると、削除した行が占有していた領域をパーティションまたはサブパーティションに割り当てることができます。この領域は、そのパーティションまたはサブパーティションに対する後続の挿入および更新のためにのみ使用できます。

参照： 「表のパーティションの切捨て例：」（12-77 ページ）

表のパーティションおよびサブパーティションの切捨ての制限事項：
update_all_indexes_clause を使用してグローバル索引を更新する場合、副次句ではなく UPDATE INDEXES キーワードのみを指定できます。

split_table_partition

split_table_partition 句を使用すると、*partition_extended_name* によって指定されるパーティションから新しいセグメント、物理属性および初期エクステントをそれぞれ含む、2 つの新しいパーティションが作成されます。現行パーティションに対応付けられたセグメントは、廃棄されます。

新しいパーティションは、指定されていないすべての物理属性を現行パーティションから継承します。

> **注意：** 一定の条件が満たされると、SPLIT PARTITION および SPLIT SUBPARTITION を最適化して処理速度を上げることができます。これらの操作の最適化の詳細は、『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。

- デフォルト・リスト・パーティションを分割すると、作成される 1 つ目のパーティションには分割値が格納され、2 つ目のパーティションにはデフォルト値が格納されます。
- *table* が索引構成されている場合、対応するマッピング表のパーティションが分割され、親の索引構成表のパーティションと同じ表領域に配置されます。対応するオーバーフロー領域も分割されます。OVERFLOW 句を使用すると、新しいオーバーフロー領域にセグメント属性を指定できます。
- *table* に LOB 列がある場合、*LOB_storage_clause* を使用して、分割の結果生成された LOB データ・セグメントに対して個々の LOB 記憶域属性を指定できます。現行パーティションの LOB データおよび LOB 索引セグメントが削除された後、新しい表領域を指定しなくても、各パーティションの各 LOB 列に新しいセグメントが作成されます。

対応するローカル索引パーティションに UNUSABLE のマークが付いている場合でも、それらは分割されます。UNUSABLE のマークが付けられ、ユーザーは、分割パーティションに対応するローカル索引パーティションを再構築する必要があります。新しい索引パーティションの属性は、分割されたパーティションから継承されます。新しい索引パーティションは、分割された索引パーティションのデフォルト表領域に格納されます。索引パーティションにデフォルト表領域が定義されていない場合、基礎となる新しい表のパーティションの表領域が使用されます。

AT 句 AT 句は、レンジ・パーティションのみに適用されます。新しい 2 つのパーティションのうちの最初の方に、新しい上限（境界を含まない）を指定します。値リストは、現行パーティションの元のパーティション境界より小さく、その次に小さいパーティション（そのようなパーティションがある場合）のパーティション境界より大きい値にする必要があります。

VALUES 句 VALUES 句は、リスト・パーティションのみに適用されます。新しい 2 つのパーティションのうちの最初の方に含まれるパーティションの値を指定します。指定したパーティションの値リストに基づいて新しい 1 番目のパーティションが Oracle Database によって作成されます。また、現行パーティションのその他のパーティション値に基づいて新しい 2 番目のパーティションが作成されます。このため、値リストには現行パーティションのすべてのパーティション値を含めることはできません。また、現行パーティションに存在しないパーティション値も含めることはできません。

INTO 句 INTO 句を使用すると、分割の結果生成された 2 つのパーティションを定義できます。分割の結果生成される 2 つのパーティションにオプションの名前および物理属性を指定しない場合でも、*range_partition_desc* または *list_partition_desc* には PARTITION キーワードを適切に指定する必要があります。新しいパーティション名を指定しない場合、SYS_P*n* という形式の名前が割り当てられます。指定しないすべての属性は、現行パーティションから継承されます。

レンジ - ハッシュ・コンポジット・パーティション表の場合、新しいパーティションにサブパーティションを指定するとき、サブパーティションに対して TABLESPACE および表の圧縮のみを指定できます。他のすべての属性は、現行パーティションから継承されます。新しいパーティションにサブパーティション化を指定しない場合は、表領域も現行パーティションから継承されます。

レンジ - リストおよびリスト - リスト・コンポジット・パーティション表の場合、新しいパーティションにサブパーティションを指定できません。分割パーティションのリスト・サブパーティションでは、サブパーティションの数および値リストは現行パーティションから継承されます。

新しく作成したサブパーティションに対して名前を指定しなかったすべてのコンポジット・パーティション表では、次のように名前が親パーティションから継承されます。

- 親パーティション内のサブパーティションが *partition_name* アンダースコア（_）*subpartition_name* という形式の名前（たとえば、P1_SUBP1）を持つ場合、新しいパーティション名に基づいて、サブパーティションの名前が生成されます（たとえば、P1A_SUB1 や P1B_SUB1）。
- 親パーティション内のサブパーティションがその他の形式の名前を持つ場合、SYS_SUBP*n* という形式のサブパーティション名が生成されます。

索引に UNUSABLE のマークが付いている場合でも、*table* で定義されている各ローカル索引の対応するパーティションが分割されます。

ヒープ構成表のすべての索引が無効になります。update_index_clauses を使用し、操作中にこれらの索引を更新できます。

table が参照パーティション表の親表の場合は、*dependent_tables_clause* を使用して、この文に指定するパーティション・メンテナンス操作を参照パーティション表のすべての子表に伝播できます。

parallel_clause を使用すると、表のデフォルトのパラレル属性を変更せずに、分割操作をパラレル化できます。

表パーティションの分割の制限事項： この句は、ハッシュ・パーティションに対して指定できません。

split_table_subpartition

この句を使用すると、重複しない値リストを持つ 2 つのサブパーティションにリスト・サブパーティションを分割できます。

注意： 一定の条件が満たされると、SPLIT PARTITION および SPLIT SUBPARTITION を最適化して処理速度を上げることができます。これらの操作の最適化の詳細は、『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。

AT 句 AT 句は、レンジ・サブパーティションに対してのみ有効です。新しい 2 つのサブパーティションのうちの 1 つ目に、新しく上限（この値は含まない）を指定します。値リストは、*subpartition_extended_name* で指定されるサブパーティションの元のサブパーティション境界より小さく、その次に小さいパーティション（そのようなパーティションがある場合）のパーティション境界より大きい値にする必要があります。

VALUES 句 VALUES 句は、リスト・サブパーティションに対してのみ有効です。新しい 2 つのサブパーティションのうち、1 番目のサブパーティションに含める値を指定します。同じパーティション内の別のサブパーティションに対して NULL を指定していない場合は、NULL を指定できます。指定したサブパーティションの値リストに基づいて新しい 1 番目のサブパーティションが作成されます。また、現行のサブパーティションのその他のパーティション値に基づいて新しい 2 番目のサブパーティションが作成されます。このため、値リストには現行のサブパーティションのすべてのパーティション値を含めることはできません。また、現行のサブパーティションに存在しないパーティション値も含めることはできません。

INTO 句 レンジ・サブパーティションとリスト・サブパーティションのどちらに対しても、INTO 句を使用すると、分割の結果生成された 2 つのサブパーティションを定義できます。2 つのサブパーティションにオプションの名前および属性を指定しない場合でも、*range_subpartition_desc* または *list_subpartition_desc* では SUBPARTITION キーワードを適切に指定する必要があります。指定しないすべての属性は、現行のサブパーティションから継承されます。

対応するローカル索引サブパーティションに UNUSABLE のマークが付いている場合でも、それらは分割されます。新しい索引サブパーティションに名前があらかじめ指定されていないかぎり、新しい表のサブパーティションの名前が継承されます。この場合、SYS_SUBPn という形式の新しい索引サブパーティション名が割り当てられます。新しい索引サブパーティションの物理属性は、親サブパーティションから継承されます。親サブパーティションにデフォルトの TABLESPACE 属性が定義されていない場合、対応する新しい表のサブパーティションの表領域が継承されます。

ヒープ構成表の索引が無効になります。update_index_clauses を使用し、これらの索引を更新できます。

表サブパーティションの分割の制限事項： 表サブパーティションの分割には、次の制限事項があります。

- この句は、ハッシュ・サブパーティションに対して指定できません。
- サブパーティションの定義で指定できる *partitioning_storage_clause* の句は、TABLESPACE および表の圧縮のみです。

merge_table_partitions

merge_table_partitions 句を使用すると、*table* の 2 つのレンジ・パーティションまたは 2 つのリスト・パーティションの内容を 1 つの新しいパーティションにマージして、元の 2 つのパーティションを削除できます。この句はハッシュ・パーティションでは無効です。かわりに、*coalesce_table_partition* 句を使用してください。

- レンジ・パーティションの場合、マージされる 2 つのパーティションは、隣接している必要があります。リスト・パーティション**およびシステム・パーティション**をマージする場合は、隣接している必要はありません。
- 2 つのレンジ・パーティションをマージする場合、新しいパーティションは、元の 2 つのパーティションのうち、上位のパーティションのパーティション境界を継承します。
- 2 つのリスト・パーティションをマージする場合、結果のパーティションの値リストは、マージされる 2 つのパーティションの値リストの集合をあわせたものです。デフォルトのリスト・パーティションを別のリスト・パーティションとマージすると、結果のパーティションはデフォルト・パーティションになり、デフォルト値が含まれます。
- 2 つのコンポジット・レンジ・パーティションまたは 2 つのコンポジット・リスト・パーティション、レンジ - リストまたはリスト - リスト・コンポジット・パーティションをマージする場合、サブパーティションの定義は指定できません。サブパーティション化情報は、サブパーティション・テンプレートから取得できます。サブパーティションのテンプレートが指定されていない場合は、1 つの MAXVALUE サブパーティションがレンジ・サブパーティションから、または 1 つのデフォルト・サブパーティションがリスト・サブパーティションから作成されます。

segment_attributes_clause に指定されていない属性はすべて、表レベルのデフォルトから継承されます。

選択したパーティションに対応するローカル索引パーティションは削除され、マージされたパーティションに対応するローカル索引パーティションに UNUSABLE のマークが付けられます。ヒープ構成表のすべてのグローバル索引にも、UNUSABLE のマークが付けられます。update_index_clauses を使用し、操作中にこれらすべての索引を更新できます。

table が参照パーティション表の親表の場合は、*dependent_tables_clause* を使用して、この文に指定するパーティション・メンテナンス操作を参照パーティション表のすべての子表に伝播できます。

> **参照：** 12-76 ページの「2 つの表パーティションのマージ例 :」および 12-76 ページの「デフォルト・リスト・パーティションの使用例 :」を参照してください。

merge_table_subpartitions

merge_table_subpartitions 句を使用すると、*table* の 2 つのレンジ・サブパーティションまたはリスト・サブパーティションの内容を 1 つの新しいサブパーティションにマージして、元の 2 つのサブパーティションを削除できます。この句はハッシュ・サブパーティションでは無効です。かわりに、*coalesce_hash_subpartition* 句を使用してください。

マージされる 2 つのサブパーティションは、同じパーティションに属している必要があります。レンジ・サブパーティションである場合は、隣接している必要があります。リスト・サブパーティションである場合は、隣接している必要はありません。マージの結果生成されるサブパーティション内のデータは、マージされたサブパーティションのデータが結合されたものです。

INTO 句を指定する場合、range_subpartition_desc または list_subpartition_desc で、それぞれ *range_values_clause* または *list_values_clause* を指定することはできません。また、*partitioning_storage_clause* に指定できる句は、TABLESPACE および *table_compression* のみです。

新しいサブパーティションに対して明示的に指定しなかった属性は、パーティション・レベルの値から継承されます。ただし、新しいサブパーティションに対してサブパーティション名を再利用すると、パーティション・レベルのデフォルト値ではなく、名前が再利用されたサブパーティションの値が新しいサブパーティションに継承されます。

対応するローカル索引のサブパーティションがマージされ、結果として生成される索引サブパーティションには UNUSABLE のマークが付けられます。また、ヒープ構成表のパーティション化されたグローバル索引とパーティション化されていないグローバル索引の両方に対して、UNUSABLE のマークが付けられます。update_index_clauses を使用すると、操作中にすべての索引を更新できます。

exchange_partition_subpart

EXCHANGE PARTITION 句または EXCHANGE SUBPARTITION 句を使用すると、次のデータおよび索引セグメントを交換できます。

- 1 つの非パーティション表と、次のいずれかを交換できます。
  - 1 つのレンジ・パーティション、リスト・パーティションまたはハッシュ・パーティション
  - 1 つのレンジ・サブパーティション、リスト・サブパーティションまたはハッシュ・サブパーティション
- 1 つのレンジ・パーティション表とレンジ - レンジまたはリスト - レンジ・コンポジット・パーティション表パーティションのレンジ・サブパーティション
- 1 つのハッシュ・パーティション表とレンジ - ハッシュまたはリスト - ハッシュのコンポジット・パーティション表パーティションのハッシュ・サブパーティション
- 1 つのリスト・パーティション表とレンジ - リストまたはハッシュ - リストのコンポジット・パーティション表パーティションのリスト・サブパーティション

交換対象の表、パーティションおよびサブパーティションの構造は、パーティション・キーを含め常に同じである必要があります。リスト・パーティションとリスト・サブパーティションの場合、対応する値リストも一致している必要があります。

この句をトランスポータブル表領域とともに使用すると、高速データ・ロードが容易になります。

> **参照：** トランスポータブル表領域の詳細は、『Oracle Database 管理者ガイド』を参照してください。

table に LOB 列がある場合、各 LOB 列の LOB データ、および LOB 索引パーティション・セグメントまたは LOB 索引サブパーティション・セグメントは、*table* の対応する LOB データおよび LOB 索引セグメントと交換されます。

2 つのオブジェクトのすべてのセグメント属性（表領域およびロギングを含む）も交換されます。

表またはパーティションの統計およびヒストグラムは交換されません。DBMS_STATS パッケージを使用して、新しいパーティションを受け取る表について、統計を再集計するか、ヒストグラムを作成します。

交換されるオブジェクトのすべてのグローバル索引は、無効になります。update_global_index_clause または update_all_indexes_clause を使用し、パーティションが交換された表のグローバル索引を更新できます。*update_all_indexes_clause* には、副次句ではなく UPDATE INDEXES キーワードのみを指定できます。交換される表のグローバル索引は、無効のままになります。これらの句とともに *parallel_clause* を指定すると、交換操作ではなく、索引の更新がパラレル化されます。

> **参照 :** 「パーティションまたはサブパーティションの交換の注意事項 :」(12-67 ページ)

WITH TABLE *table* パーティションまたはサブパーティションを交換する表を指定します。

INCLUDING | EXCLUDING INDEXES INCLUDING INDEXES を指定すると、ローカル索引パーティションまたはサブパーティションを対応する表索引(非パーティション表の場合)またはローカル索引(ハッシュ・パーティション化表の場合)と交換できます。EXCLUDING INDEXES を指定すると、交換された表のすべての標準索引、索引パーティションおよびパーティションに対応するすべての索引パーティションまたはサブパーティションに UNUSABLE のマークを付けることができます。

WITH | WITHOUT VALIDATION WITH VALIDATION を指定すると、交換された表にあるいずれかの行が交換されたパーティションまたはサブパーティションにマップされない場合にエラーが戻されます。WITHOUT VALIDATION を指定すると、交換された表にある行が正しくマップされたかどうかがチェックされません。

exceptions_clause この句の詳細は、8-16 ページの「制約の例外の処理」を参照してください。パーティション交換のコンテキストでは、この句は、パーティション表が一意制約を使用して定義されている場合にのみ有効です。また、制約は DISABLE VALIDATE の状態である必要があります。この句は、サブパーティションではなくパーティション交換に対してのみ有効です。

> **参照 :**
>
> - SQL スクリプトの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の DBMS_IOT パッケージを参照してください。
> - 移行行および連鎖行の削除については、『Oracle Database 管理者ガイド』を参照してください。
> - 制約の確認の詳細は、8-4 ページの「constraint」および 12-74 ページの「索引構成表の例外表の作成例 :」を参照してください。

パーティションまたはサブパーティションの交換の注意事項 : パーティションとサブパーティションの交換時には、次の点に注意します。

- 交換される両方の表は同じ主キーを含む必要があり、参照表が空でないかぎり、どちらの表も有効な外部キーを参照できません。
- パーティション化された索引構成表の交換時には、次の点に注意します。
  - ソースおよびターゲットの表およびパーティションは、その主キーが同じ列に同じ順序で設定されている必要があります。
  - キー圧縮が使用可能な場合は、ソースおよびターゲットの両方で使用可能で、接頭辞の長さは同じである必要があります。
  - ソースおよびターゲットの両方は、索引構成されている必要があります。

- ソースおよびターゲットの両方に、オーバーフロー・セグメントが必要です。または、ソースおよびターゲットの両方がオーバーフロー・セグメントを持ってはいけません。また、ソースおよびターゲットの両方ともマッピング表を含むか、または両方とも含まない必要があります。
- ソースおよびターゲットの両方は、LOB 列に対して記憶域属性が同一である必要があります。

参照： 「表パーティションの交換例 :」（12-76 ページ）

dependent_tables_clause

この句は、参照パーティション表の親表を変更する場合にのみ有効です。この句では、親表の子の参照パーティション表の操作によって作成されるパーティションの属性を指定できます。

- 親表がコンポジット・パーティションではない場合、1 つ以上の子表を指定し、子表ごとに、親表内に作成された各パーティションの 1 つの *partition_spec* を指定します。
- 親表がコンポジットの場合、1 つ以上の子表を指定し、子表ごとに、親表内に作成された各サブパーティションの 1 つの *partition_spec* を指定します。

参照： 参照パーティション表の作成の詳細は、16-48 ページの「CREATE TABLE」の句「reference_partitioning」を参照してください。一般的な参照によるパーティション化の詳細は、『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。

UNUSABLE LOCAL INDEXES 句

この 2 つの句を使用すると、パーティションとサブパーティションのどちらを変更するかに応じて、*partition* に対応するローカル索引パーティションおよび索引サブパーティションの属性を変更できます。

- UNUSABLE LOCAL INDEXES を指定すると、*partition* に関連付けられたローカル索引パーティションまたは索引サブパーティションに、UNUSABLE のマークが付けられます。
- REBUILD UNUSABLE LOCAL INDEXES を指定すると、*partition* に関連付けられた UNUSABLE のローカル索引パーティションまたは索引サブパーティションが再構築されます。

UNUSABLE LOCAL INDEXES 句の制限事項： この句には、次の制限事項があります。

- この句を *modify_table_partition* の他の句と同時に指定することはできません。
- サブパーティションが含まれるパーティションに対しては、*modify_table_partition* でこの句を指定できません。ただし、*modify_range_subpartition*、*modify_hash_subpartition* または *modify_list_subpartition* 句では指定できます。

update_index_clauses

update_index_clauses を使用すると、表のパーティション化操作の一部として *table* の索引を更新できます。表パーティションで DDL 文を実行する場合、*table* に索引が定義されていると、DDL 文を実行中のパーティションだけでなく索引全体が無効になります。この句を使用すると、変更する索引パーティションを DDL 操作中に更新できるため、DDL 文の後で索引を再構築する必要がなくなります。

パーティション化された索引構成表では、*update_index_clauses* は不要であり、無効です。索引構成表は主キー・ベースであるため、値を変更せずにデータを移動する操作では、グローバル索引は USABLE のままです。

update_global_index_clause

この句を使用すると、*table* のグローバル索引を更新できます。

update_all_indexes_clause
この句を使用すると、*table* のすべての索引を更新できます。

update_index_partition この句は、表パーティションの操作に対してのみ有効で、ローカル索引にのみ影響を与えます。

- *index_partition_description* を使用すると、各ローカル索引のそれぞれのパーティションの物理属性、表領域の記憶域およびロギングを指定できます。PARTITION キーワードのみを指定すると、索引のパーティションは次のように更新されます。
  - 単一の表パーティションに対する操作（MOVE PARTITION、SPLIT PARTITION など）の場合、対応する索引パーティションは処理された索引表パーティションの属性を継承します。索引パーティションの名前は生成されないため、この操作によって作成された新しい索引パーティションは、対応する新しい表パーティションから名前を継承します。
  - MERGE PARTITION 操作の場合、この操作によって作成されたローカル索引パーティションは、作成された表パーティションの名前とローカル索引の属性を継承します。

  ドメイン索引の場合、PARAMETERS 句を使用すると、未解析のまま適切な ODCI 索引タイプ・ルーチンに渡すパラメータ文字列を指定できます。PARAMETERS 句は、ドメイン索引に対してのみ有効であり、ドメイン索引について指定できる *index_partition_description* の唯一の部分です。

  参照： ドメイン索引の詳細は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。

- コンポジット・パーティション索引の場合、*index_subpartition_clause* を使用すると、各サブパーティションに対して表領域の記憶域を指定できます。*update_index_partition* 句のこのコンポーネントの詳細は、14-65 ページの「CREATE INDEX」の「index_subpartition_clause」を参照してください。

update_index_subpartition この句は、コンポジット・パーティション表のサブパーティションの操作に対してのみ有効で、コンポジット・パーティション表のローカル索引にのみ影響を与えます。1 つ以上のサブパーティションに表領域の記憶域を指定できます。

すべての索引の更新の制限事項： この句は、索引構成表に対して指定できません。

update_global_index_clause
この句を指定すると、*table* のグローバル索引のみを更新できます。*table* のすべてのローカル索引には、UNUSABLE のマークが付けられます。

UPDATE GLOBAL INDEXES UPDATE GLOBAL INDEXES を指定すると、*table* で定義したグローバル索引を更新できます。

グローバル索引の更新の制限事項： グローバル索引が LOB 列のグローバル・ドメイン索引として定義されている場合、ドメイン索引は、更新されるのではなく UNUSABLE のマークが付けられます。

INVALIDATE GLOBAL INDEXES INVALIDATE GLOBAL INDEXES を指定すると、*table* で定義したグローバル索引を無効にできます。

どちらも指定しない場合、グローバル索引は無効になります。

グローバル索引の無効化の制限事項： この句はグローバル索引のみをサポートしています。索引構成表はサポートしていません。また、この句では USABLE および VALID の索引のみが更新されます。UNUSABLE の索引は使用禁止のままになり、INVALID のグローバル索引は無視されます。

参照： 12-77 ページの「グローバル索引の更新例 :」および 12-77 ページの「パーティション索引の更新例 :」を参照してください。

parallel_clause

parallel_clause を使用すると、表の問合せおよび DML に対するデフォルトの並列度を変更できます。

この句の詳細は、16-54 ページの「CREATE TABLE」の「parallel_clause」を参照してください。

表の並列化の変更の制限事項： 表の並列化の変更には、次の制限事項があります。

- *table* に LOB 型またはユーザー定義オブジェクト型の列が含まれている場合、この *table* での INSERT、UPDATE および DELETE は、通知なしに逐次実行されます。ただし、後続の問合せはパラレルで実行されます。
- *parallel_clause* を *move_table_clause* と組み合せて指定する場合、このパラレル化は移動のみに適用され、後続の表での DML 操作および問合せには適用されません。

参照： 「パラレル処理の指定例 :」（12-73 ページ）

move_table_clause

move_table_clause を使用すると、非パーティション表のデータまたはパーティション表のパーティションのデータを新しいセグメントに再配置できます。オプションとして、別の表領域への配置および記憶域属性の変更を行うこともできます。

LOB_storage_clause 句および *varray_col_properties* 句を使用して、表またはパーティションに関連付けられた LOB データ・セグメントを移動することもできます。この句で指定していない LOB 項目は移動できません。

表を別の表領域に移動する場合、COMPATIBLE パラメータが 10.0 以上に設定されていると、ネストした表の列に対する記憶表は、その表が作成された表領域に残ります。COMPATIBLE が 10.0 未満に設定されている場合、表と記憶表は新しい表領域に自動的に移動します。

ONLINE 句 この句は、トップレベルの索引構成表、および索引構成済のネストした表の記憶表に対してのみ有効です。ONLINE を指定すると、表の主キー索引の再構築中に、索引構成表に対する DML 操作を実行できます。

表のオンライン化の制限事項： 表のオンライン化には、次の制限事項があります。

- 同じ文でこの句と他の句は結合できません。
- この句は、パーティション化された索引構成表に対して指定できません。
- オンライン MOVE 中のパラレル DML はサポートされていません。ONLINE を指定し、パラレル DML 文を発行すると、Oracle Database はエラーを戻します。
- 索引構成表に LOB、VARRAY、Oracle が提供する型またはユーザー定義オブジェクト型の列が含まれている場合には、この句は指定できません。

index_org_table_clause

索引構成表の場合、*move_table_clause* の *index_org_table_clause* を使用すると、オーバーフロー・セグメント属性も指定できます。*move_table_clause* を使用すると、索引構成表の主キー索引が再構築されます。オーバーフロー・データ・セグメントは、キーワード OVERFLOW が明示的に指定されていないかぎり、再構築されません。ただし、次の場合は例外です。

- ALTER TABLE 文の一部として PCTTHRESHOLD の値または INCLUDING 列を変更する場合は、オーバーフロー・データ・セグメントが再構築されます。
- 索引構成表内のアウトラインの列（LOB 列、VARRAY 列、ネストした表の列）のいずれかを明示的に移動する場合は、オーバーフロー・データ・セグメントも再構築されます。

LOB 列の索引およびデータ・セグメントは、LOB 列を ALTER TABLE 文の一部として明示的に指定しないかぎり、再構築されません。

mapping_table_clauses MAPPING TABLE を指定すると、マッピング表が存在していない場合にマッピング表を作成できます。マッピング表がすでに存在する場合、マッピング表は索引構成表とともに移動され、すべてのビットマップ索引には UNUSABLE のマークが付けられます。新しいマッピング表は、親表と同じ表領域に作成されます。

NOMAPPING を指定すると、既存のマッピング表が削除されます。

この句の詳細は、16-33 ページの「CREATE TABLE」の「mapping_table_clauses」を参照してください。

表のマッピングの制限事項： *table* でビットマップ索引が定義されている場合は、NOMAPPING を指定できません。

key_compression *key_compression* 句を使用すると、索引構成表のキー圧縮を使用可能または使用禁止にできます。

- COMPRESS を指定すると、キー圧縮が使用可能になります。これによって、索引構成表の主キー列の値が重複しなくなります。*integer* を使用して、接頭辞の長さ（圧縮する接頭辞列数）を指定します。

  接頭辞の長さの有効範囲は、1 ～（主キー列数 -1）までです。デフォルトでは（主キー列数 -1）になります。

- NOCOMPRESS を指定すると、索引構成表でのキー圧縮が使用禁止になります。これはデフォルトです。

TABLESPACE *tablespace* 再構築した索引構成表を格納する表領域を指定します。

LOB_storage_clause この句を使用すると、LOB セグメントを別の表領域に移動できます。表に LONG 列が含まれている場合は、この句を使用して LOB セグメントを移動することはできません。かわりに、LONG 列を LOB に変換するか、または表をエクスポートし、LOB 列に必要な表領域の記憶域を指定する表を再作成して表データを再インポートする必要があります。

表の移動の制限事項： 表の移動には、次の制限事項があります。

- MOVE を指定する場合は、ALTER TABLE 文の最初の句にする必要があります。この句以外では、*physical_attributes_clause*、*parallel_clause* および *LOB_storage_clause* のみが指定できます。
- LONG または LONG RAW 列を含む表は、移動できません。
- パーティション表（ヒープ表または索引構成表）全体の移動はできません。個々のパーティションまたはサブパーティションを移動してください。

LOB についての注意： *move_table_clause* で指定するすべての LOB 列については、次のことに注意してください。

- 新しい表領域が指定されていない場合でも、古い LOB データ・セグメントとこれに対応する索引セグメントは削除され、新しいセグメントが作成されます。
- *table* の LOB 索引がその LOB データと異なる表領域にある場合、移動後の LOB 索引は、LOB データと同じ表領域にまとめて格納されます。

参照： 12-57 ページの「move_table_partition」および 12-58 ページの「move_table_subpartition」を参照してください。

enable_disable_clause

enable_disable_clause を使用すると、Oracle Database が整合性制約を適用するかどうか、およびその方法を指定できます。DROP および KEEP 句は、一意制約または主キー制約を使用禁止にする場合のみに有効です。

参照：

- この句の詳細、関連する注意および制限事項については、16-54 ページの「CREATE TABLE」の「enable_disable_clause」を参照してください。
- 索引の使用方法と制約の適用方法の詳細は、8-15 ページの「索引による制約の適用」を参照してください。

TABLE LOCK

DDL 操作中にロックされた表にのみ DDL 操作を実行できます。このような表ロックは、DML 操作中は必要ありません。

注意： 一時表に表ロックを適用することはできません。

ENABLE TABLE LOCK ENABLE TABLE LOCK を指定すると、表ロックが有効になり、表に対する DDL 操作が実行可能になります。現在実行中のすべてのトランザクションは、表ロックが有効になる前にコミットまたはロールバックする必要があります。

注意： Oracle Database は、表をロックする前に、データベース内のアクティブな DML トランザクションが完了するまで待機します。その結果、遅延が生じる可能性があることに注意してください。

DISABLE TABLE LOCK DISABLE TABLE LOCK を指定すると、表ロックが無効になり、表に対する DDL 操作が実行できなくなります。

ALL TRIGGERS

ALL TRIGGERS 句を使用すると、表に関連するすべてのトリガーを使用可能または使用禁止にできます。

ENABLE ALL TRIGGERS ENABLE ALL TRIGGERS を指定すると、表に関連するすべてのトリガーが使用可能になります。トリガー条件が満たされた場合に、トリガーが起動されます。

1 つのトリガーを使用可能にする場合は、ALTER TRIGGER の *enable_clause* を使用してください。

参照： 16-85 ページの「CREATE TRIGGER」、13-2 ページの「ALTER TRIGGER」および 12-74 ページの「トリガーを使用可能にする例 :」を参照してください。

DISABLE ALL TRIGGERS DISABLE ALL TRIGGERS を指定すると、表に関連するすべてのトリガーが使用禁止になります。トリガー条件が満たされた場合でも、使用禁止のトリガーは起動されません。

例

コレクションの取出し例： 次の文は、サンプル表 sh.print_media のネストした表の列 ad_textdocs_ntab を変更し、問合せ時にロケータのかわりに実値を戻します。

```
ALTER TABLE print_media MODIFY NESTED TABLE ad_textdocs_ntab
   RETURN AS VALUE;
```

パラレル処理の指定例： 次の文は、サンプル表 oe.customers への問合せに対してパラレル処理を指定します。

```
ALTER TABLE customers
   PARALLEL;
```

制約状態の変更例： 次の文は、employees 表の emp_manager_fk という名前の整合性制約を ENABLE VALIDATE 状態にします。

```
ALTER TABLE employees
   ENABLE VALIDATE CONSTRAINT emp_manager_fk
   EXCEPTIONS INTO exceptions;
```

Oracle Database が制約を使用可能にするためには、employees 表の各行がこの制約を満たしている必要があります。制約に違反する行があれば、制約は使用禁止のままになります。すべての例外は、exceptions 表に記述されます。次の文で、employees 表の例外を検出することもできます。

```
SELECT e.*
   FROM employees e, exceptions ex
   WHERE e.rowid = ex.row_id
      AND ex.table_name = 'EMPLOYEES'
      AND ex.constraint = 'EMP_MANAGER_FK';
```

次の文は、employees 表の 2 つの制約を ENABLE NOVALIDATE 状態にします。

```
ALTER TABLE employees
   ENABLE NOVALIDATE PRIMARY KEY
   ENABLE NOVALIDATE CONSTRAINT emp_last_name_nn;
```

この文には、次の 2 つの ENABLE 句が含まれています。

- 1 番目の ENABLE 句は、表の主キー制約を ENABLE NOVALIDATE 状態にします。
- 2 番目の ENABLE 句は、emp_last_name_nn という制約を ENABLE NOVALIDATE 状態にします。

この例では、表のそれぞれの行が 2 つの制約を満たす場合にかぎり、その制約が使用可能になります。どちらかの制約に違反する行があった場合、エラーが戻され、どちらの制約も使用禁止のままになります。

departments 表の location_id 列の外部キー制約について考えます。ここでは、locations 表の主キーを参照しています。次の文は、locations 表の主キーを使用禁止にします。

```
ALTER TABLE locations
   MODIFY PRIMARY KEY DISABLE CASCADE;
```

locations 表の一意キーは、departments 表の外部キーによって参照されるため、この主キーを使用禁止にする場合は、CASCADE 句を指定します。この句によって、外部キーも使用禁止になります。

索引構成表の例外表の作成例： 次の例は、主キー制約に違反する索引構成表 hr.countries からの行を保持する except_table 表を作成します。

```
EXECUTE DBMS_IOT.BUILD_EXCEPTIONS_TABLE ('hr', 'countries', 'except_table');
ALTER TABLE countries
   ENABLE PRIMARY KEY
   EXCEPTIONS INTO except_table;
```

例外表を指定する場合は、この表に行を挿入する権限が必要です。検出された例外を調べる場合、例外表を問い合せる権限が必要です。

> **参照：** 表に行を挿入するために必要な権限の詳細は、18-53 ページの「INSERT」および 19-4 ページの「SELECT」を参照してください。

CHECK 制約を使用禁止にする例： 次の文は、employees 表に CHECK 制約を定義し、その制約を使用禁止にします。

```
ALTER TABLE employees ADD CONSTRAINT check_comp
   CHECK (salary + (commission_pct*salary) <= 5000)
   DISABLE;
```

check_comp 制約は、給与総額が 5000 ドルを超える従業員がいないことを保証します。ただし、この制約が使用禁止になっているため、従業員の給与をこの制限以上に増やすことができます。

トリガーを使用可能にする例： 次の文は、employees 表に対応付けられているすべてのトリガーを使用可能にします。

```
ALTER TABLE employees
   ENABLE ALL TRIGGERS;
```

未使用領域の解放例： 次の文は、employees 表で再利用できるように最高水位標が MINEXTENTS を超えるすべての未使用領域を解放します。

```
ALTER TABLE employees
    DEALLOCATE UNUSED;
```

列名の変更例： 次の文は、サンプル表 oe.customers の列名を credit_limit から credit_amount に変更します。

```
ALTER TABLE customers
   RENAME COLUMN credit_limit TO credit_amount;
```

列の削除例： 次の文は、CASCADE CONSTRAINTS が指定されている *drop_column_clause* です。表 t1 が次のように作成されているとします。

```
CREATE TABLE t1 (
   pk NUMBER PRIMARY KEY,
   fk NUMBER,
   c1 NUMBER,
   c2 NUMBER,
   CONSTRAINT ri FOREIGN KEY (fk) REFERENCES t1,
   CONSTRAINT ck1 CHECK (pk > 0 and c1 > 0),
   CONSTRAINT ck2 CHECK (c2 > 0)
);
```

次の文に対してエラーが戻されます。

```
/* The next two statements return errors:
ALTER TABLE t1 DROP (pk);  -- pk is a parent key
ALTER TABLE t1 DROP (c1);  -- c1 is referenced by multicolumn
                           -- constraint ck1
```

次の文を発行すると、列 pk、主キー制約、外部キー制約 ri および CHECK 制約 ck1 が削除されます。

```
ALTER TABLE t1 DROP (pk) CASCADE CONSTRAINTS;
```

削除された列に定義した制約が参照する列もすべて削除される場合、CASCADE CONSTRAINTS は必要ありません。たとえば、他の表から列 pk を参照する他の参照制約が存在していないとします。この場合は、CASCADE CONSTRAINTS 句を指定しない次の文が有効になります。

```
ALTER TABLE t1 DROP (pk, fk, c1);
```

索引構成表の変更例： 次の文は、hr.countries に基づく索引構成表 countries_demo の索引セグメントの INITRANS パラメータを変更します。

```
ALTER TABLE countries_demo INITRANS 4;
```

次の文は、オーバーフロー・データ・セグメントを索引構成表 countries に追加します。

```
ALTER TABLE countries_demo ADD OVERFLOW;
```

次の文は、索引構成表 countries のオーバーフロー・データ・セグメントの INITRANS パラメータを変更します。

```
ALTER TABLE countries_demo OVERFLOW INITRANS 4;
```

表のパーティションの分割例： 次の文は、サンプル表 sh.sales の古いパーティション sales_q4_2000 を分割して 2 つの新しいパーティションを作成し、1 つには sales_q4_2000b という名前を付け、もう 1 つには旧パーティションの名前を再利用します。

```
ALTER TABLE sales SPLIT PARTITION SALES_Q4_2000
   AT (TO_DATE('15-NOV-2000','DD-MON-YYYY'))
   INTO (PARTITION SALES_Q4_2000, PARTITION SALES_Q4_2000b);
```

サンプル表 pm.print_media は、パーティション p1 とパーティション p2 にレンジ・パーティション化されたとします（表をパーティション化する前に、print_media の LONG 列を LOB に変換する必要があります）。次の文は、表のパーティション p2 をパーティション p2a と p2b に分割します。

```
ALTER TABLE print_media_part
   SPLIT PARTITION p2 AT (150) INTO
   (PARTITION p2a TABLESPACE omf_ts1
      LOB ad_photo, ad_composite) STORE AS (TABLESPACE omf_ts2),
   PARTITION p2b
      LOB (ad_photo, ad_composite) STORE AS (TABLESPACE omf_ts2));
```

p2a と p2b のパーティションでは、列 ad_photo と ad_composite に対する LOB セグメントが表領域 omb_ts2 内に作成されます。パーティション p2a のその他の列に対する LOB セグメントは、表領域 omf_ts1 に保存されます。パーティション p2b のその他の列に対する LOB セグメントは、この ALTER 文の実行前の表領域で保持されます。ただし、LOB データおよび LOB 索引セグメントが新しい表領域に移動されない場合でも、これらの新しいセグメントが作成されます。

LOB を持つ表パーティションの追加例： 次の文は、パーティション p3 を print_media_part 表に追加し（前述の例を参照）、表の BLOB と CLOB 列の記憶特性を指定します。

```
ALTER TABLE print_media_part ADD PARTITION p3 VALUES LESS THAN (MAXVALUE)
   LOB (ad_photo, ad_composite) STORE AS (TABLESPACE omf_ts2)
   LOB (ad_sourcetext, ad_finaltext) STORE AS (TABLESPACE omf_ts1);
```

パーティション p3 の列 ad_photo および ad_composite に対する LOB データと LOB 索引セグメントは、表領域 omf_ts2 に格納されます。LOB 列の他の属性は、まず表レベルのデフォルトから継承され、次に表領域のデフォルトから継承されます。

列 ad_source_text および ad_finaltext の LOB データ・セグメントは、omf_ts1 表領域に格納され、他のすべての属性は、表レベルのデフォルト値から継承され、次に表領域のデフォルト値から継承されます。

デフォルト・リスト・パーティションの使用例： 次の文は、リスト・パーティション表（16-67 ページの「リスト・パーティション化の例：」で作成）を使用します。最初の文は、既存のデフォルト・パーティションを新規の south パーティションとデフォルト・パーティションに分割します。

```
ALTER TABLE list_customers SPLIT PARTITION rest
   VALUES ('MEXICO', 'COLOMBIA')
   INTO (PARTITION south, PARTITION rest);
```

次の文は、結果のデフォルト・パーティションを asia パーティションとマージします。

```
ALTER TABLE list_customers
   MERGE PARTITIONS asia, rest INTO PARTITION rest;
```

次の文は、デフォルト・パーティションを分割して、asia パーティションを再作成します。

```
ALTER TABLE list_customers SPLIT PARTITION rest
   VALUES ('CHINA', 'THAILAND')
   INTO (PARTITION asia, partition rest);
```

2 つの表パーティションのマージ例： 次の文は、パーティション（12-75 ページの「表のパーティションの分割例：」で作成）をマージして、1 つのパーティションに戻します。

```
ALTER TABLE sales
   MERGE PARTITIONS sales_q4_2000, sales_q4_2000b
   INTO PARTITION sales_q4_2000;
```

表のパーティションの削除例： 次の文は、パーティション p3（12-75 ページの「LOB を持つ表パーティションの追加例：」で作成）を削除します。

```
ALTER TABLE print_media_part DROP PARTITION p3;
```

表パーティションの交換例： 次の例では、list_customers 表（16-67 ページの「リスト・パーティション化の例：」で作成）のパーティションと同じ構造を持つ exchange_table 表を作成します。次に、パーティション rest を表 exchange_table に置き換えます。ローカル索引パーティションと exchange_table に対応する索引との交換、および exchange_table 内のデータがパーティション rest の範囲内かどうかの検証は行われません。

```
CREATE TABLE exchange_table (
   customer_id     NUMBER(6),
   cust_first_name VARCHAR2(20),
   cust_last_name  VARCHAR2(20),
   cust_address    CUST_ADDRESS_TYP,
   nls_territory   VARCHAR2(30),
   cust_email      VARCHAR2(30));

ALTER TABLE list_customers
   EXCHANGE PARTITION feb97 WITH TABLE sales_feb97
   WITHOUT VALIDATION;
```

表のパーティションの変更例： 次の文は、list_customers 表のパーティション asia に対応するすべてのローカル索引パーティションに、UNUSABLE のマークを付けます。

```
ALTER TABLE list_customers MODIFY PARTITION asia
   UNUSABLE LOCAL INDEXES;
```

次の文は、UNUSABLE のマークが付けられたすべてのローカル索引パーティションを再構築します。

```
ALTER TABLE list_customers MODIFY PARTITION asia
   REBUILD UNUSABLE LOCAL INDEXES;
```

表のパーティションの移動例： 次の文は、パーティション p2b（12-75 ページの「表のパーティションの分割例：」で作成）を表領域 omf_ts1 に移動します。

```
ALTER TABLE print_media_part
   MOVE PARTITION p2b TABLESPACE omf_ts1;
```

表のパーティション名の変更例： 次の文は、sh.sales 表のパーティションの名前を変更します。

```
ALTER TABLE sales RENAME PARTITION sales_q4_2003 TO sales_currentq;
```

表のパーティションの切捨て例： 次の文は、print_media_demo 表（16-67 ページの「LOB 列のあるパーティション表の例：」で作成）を使用します。p1 パーティションのすべてのデータが削除され、解放された領域の割当てが解除されます。

```
ALTER TABLE print_media_demo
   TRUNCATE PARTITION p1 DROP STORAGE;
```

グローバル索引の更新例： 次の文は、サンプル表 sh.sales のパーティション sales_q1_2000 を分割し、定義されているグローバル索引を更新します。

```
ALTER TABLE sales SPLIT PARTITION sales_q1_2000
   AT (TO_DATE('16-FEB-2000','DD-MON-YYYY'))
   INTO (PARTITION q1a_2000, PARTITION q1b_2000)
   UPDATE GLOBAL INDEXES;
```

パーティション索引の更新例： 次の文は、サンプル表 sh.costs のパーティション costs_Q4_2003 を分割し、定義されているローカル索引を更新します。16-83 ページの「基本的な一時表領域の作成例：」で作成した表領域を使用します。

```
CREATE INDEX cost_ix ON costs(channel_id) LOCAL;

ALTER TABLE costs
  SPLIT PARTITION costs_q4_2003 at
    (TO_DATE('01-Nov-2003','dd-mon-yyyy'))
    INTO (PARTITION c_p1, PARTITION c_p2)
  UPDATE INDEXES (cost_ix (PARTITION c_p1 tablespace tbs_02,
                           PARTITION c_p2 tablespace tbs_03));
```

オブジェクト識別子の指定例： 次の文は、オブジェクト型、主キーに基づくオブジェクト識別子に対応するオブジェクト表、およびユーザー定義 REF 列を持つ表を作成します。

```
CREATE TYPE emp_t AS OBJECT (empno NUMBER, address CHAR(30));

CREATE TABLE emp OF emp_t (
   empno PRIMARY KEY)
   OBJECT IDENTIFIER IS PRIMARY KEY;

CREATE TABLE dept (dno NUMBER, mgr_ref REF emp_t SCOPE is emp);
```

次の文は、emp 表を参照する制約およびユーザー定義 REF 列を追加します。

```
ALTER TABLE dept ADD CONSTRAINT mgr_cons FOREIGN KEY (mgr_ref)
   REFERENCES emp;
ALTER TABLE dept ADD sr_mgr REF emp_t REFERENCES emp;
```

表の列の追加例： 次の文は、NUMBER データ型の列 duty_pct、サイズが 3 の VARCHAR2 データ型の列 visa_needed および CHECK 整合性制約を countries 表に追加します。

```
ALTER TABLE countries
   ADD (duty_pct     NUMBER(2,2)  CHECK (duty_pct < 10.5),
        visa_needed  VARCHAR2(3));
```

仮想表の列の追加例： 次の文は、列 income を hr.employees 表のコピーに追加します。この列は、給与と歩合の合計です。給与と歩合はどちらも NUMBER 列であるため、データ型が文に指定されていなくても、データベースによって仮想列は NUMBER 列として作成されます。

```
CREATE TABLE emp2 AS SELECT * FROM employees;

ALTER TABLE emp2 ADD (income AS (salary + (salary*commission_pct)));
```

表の列の変更例： 次の文は、duty_pct 列のサイズを増やします。

```
ALTER TABLE countries
   MODIFY (duty_pct NUMBER(3,2));
```

MODIFY 句には列の定義が 1 つのため、定義を囲むカッコは任意指定です。

次の文は、employees 表の PCTFREE パラメータと PCTUSED パラメータの値を、それぞれ 30 と 60 に変更します。

```
ALTER TABLE employees
   PCTFREE 30
   PCTUSED 60;
```

データの暗号化例： 次の文は、暗号化アルゴリズム 3DES168 を使用して、hr.employees 表の salary 列を暗号化します。前述の「セマンティクス」で説明したように、まず、データの透過的暗号化を有効にする必要があります。

```
ALTER TABLE employees
   MODIFY (salary ENCRYPT USING '3DES168');
```

次の文は、暗号化された新しい列 online_acct_pw を oe.customers 表に追加します。

```
ALTER TABLE customers
   ADD (online_acct_pw VARCHAR2(8) ENCRYPT);
```

次の例は、customer.online_acct_pw 列を復号化します。

```
ALTER TABLE customers
   MODIFY (online_acct_pw DECRYPT);
```

エクステントの割当て例： 次の文は、employees 表に 5KB のエクステントを割り当て、そのエクステントをインスタンス 4 が使用できるようにします。

```
ALTER TABLE employees
  ALLOCATE EXTENT (SIZE 5K INSTANCE 4);
```

この文には、DATAFILE パラメータが指定されていないため、エクステントは employees 表が入っている表領域に属するデータ・ファイルの 1 つに割り当てられます。

デフォルト列値の指定例： 次の文は、product_information 表の min_price 列のデフォルト値を 10 に変更します。

```
ALTER TABLE product_information
  MODIFY (min_price DEFAULT 10);
```

続いて min_price 列に値を指定せずに、product_information 表に新しい行を追加する場合、min_price 列の値は自動的に 0（ゼロ）になります。

```
INSERT INTO product_information (product_id, product_name,
   list_price)
   VALUES (300, 'left-handed mouse', 40.50);

SELECT product_id, product_name, list_price, min_price
    FROM product_information
    WHERE product_id = 300;

PRODUCT_ID PRODUCT_NAME         LIST_PRICE  MIN_PRICE
---------- -------------------- ---------- ----------
       300 left-handed mouse          40.5         10
```

以前に指定したデフォルト値を中止して、新しく追加する行にその値が自動的に挿入されないようにする場合、次の文に示すように、デフォルト値を NULL に置き換えます。

```
ALTER TABLE product_information
   MODIFY (min_price DEFAULT NULL);
```

MODIFY 句には、列の定義をすべて指定する必要はありません。列名および変更部分のみを指定してください。この文は、既存の行の既存の値には影響しません。

XMLType 表への制約の追加例： 次の例は、xwarehouses 表（16-64 ページの「XMLType の例」で作成）に主キー制約を追加します。

```
ALTER TABLE xwarehouses
   ADD (PRIMARY KEY(XMLDATA."WarehouseID"));
```

この疑似列の詳細は、3-10 ページの「XMLDATA 疑似列」を参照してください。

制約名の変更例： 次の文は、サンプル表 oe.customers の制約名を cust_fname_nn から cust_firstname_nn に変更します。

```
ALTER TABLE customers RENAME CONSTRAINT cust_fname_nn
   TO cust_firstname_nn;
```

制約の削除例： 次の文は、departments 表の主キーを削除します。

```
ALTER TABLE departments
    DROP PRIMARY KEY CASCADE;
```

主キー制約の名前が pk_dept であることがわかっている場合は、次のように指定しても削除できます。

```
ALTER TABLE departments
    DROP CONSTRAINT pk_dept CASCADE;
```

CASCADE 句によって、主キーを参照するすべての外部キーが削除されます。

次の文は、employees 表の email 列の一意キーを削除します。

```
ALTER TABLE employees
    DROP UNIQUE (email);
```

この文の DROP 句では CASCADE 句を省略します。CASCADE オプションを省略することによって、一意キーを参照する外部キーがある場合、その一意キーは削除されません。

LOB 列の例： 次の文は、CLOB 列の resume を employee 表に追加し、新しい列の LOB 記憶特性を指定します。

```
ALTER TABLE employees ADD (resume CLOB)
  LOB (resume) STORE AS resume_seg (TABLESPACE example);
```

次の文は、キャッシュを使用できるように LOB 列の resume を変更します。

```
ALTER TABLE employees MODIFY LOB (resume) (CACHE);
```

次の文は、SecureFile CLOB 列の resume を employee 表に追加し、新しい列の LOB 記憶特性を指定します。SecureFile LOB は、自動セグメント領域管理の表領域内に格納される必要があります。そのため、この例の LOB データは auto_seg_ts 表領域内に格納されます。この表領域は、16-84 ページの「表領域に対してセグメント領域管理を指定する場合の例 :」で作成されました。

```
ALTER TABLE employees ADD (resume CLOB)
LOB (resume) STORE AS SECUREFILE resume_seg (TABLESPACE auto_seg_ts);
```

次の文は、LOB 列の resume をキャッシュを使用しないように変更します。

```
ALTER TABLE employees MODIFY LOB (resume) (NOCACHE);
```

ネストした表の例 : 次の文は、ネストした表の列 skills を employees 表に追加します。

```
ALTER TABLE employees ADD (skills skill_table_type)
    NESTED TABLE skills STORE AS nested_skill_table;
```

また、ネストした表の記憶特性も変更できます。変更する場合、*nested_table_col_properties* に指定した記憶表の名前を使用してください。記憶表では、問合せまたは DML 文を実行することはできません。記憶表は、ネストした表の列の記憶特性を変更するためにのみ使用します。

次の文は、ネストした表の列 client と記憶表 client_tab を使用して、表 vet_service を作成します。ネストした表 client_tab を変更して制約を指定します。

```
CREATE TYPE pet_t AS OBJECT
   (pet_id NUMBER, pet_name VARCHAR2(10), pet_dob DATE);
/

CREATE TYPE pet AS TABLE OF pet_t;
/

CREATE TABLE vet_service (vet_name VARCHAR2(30),
                          client   pet)
  NESTED TABLE client STORE AS client_tab;

ALTER TABLE client_tab ADD UNIQUE (pet_id);
```

次の文は、REF 値のネストした表用の記憶表を変更して、REF の範囲が限定されることを指定します。

```
CREATE TYPE emp_t AS OBJECT (eno number, ename char(31));
CREATE TYPE emps_t AS TABLE OF REF emp_t;
CREATE TABLE emptab OF emp_t;
CREATE TABLE dept (dno NUMBER, employees emps_t)
   NESTED TABLE employees STORE AS deptemps;
ALTER TABLE deptemps ADD (SCOPE FOR (COLUMN_VALUE) IS emptab);
```

同様に、次の文は、REF を ROWID とともに格納することを指定します。

```
ALTER TABLE deptemps ADD (REF(column_value) WITH ROWID);
```

これらの ALTER TABLE 文を正確に実行するためには、記憶表 deptemps が空である必要があります。また、ネストした表は、スカラー値（REF 値）の表として定義されるため、Oracle Database は、暗黙的に列名 COLUMN_VALUE を記憶表に設定します。

参照 :

- ネストした表の記憶域の詳細は、16-6 ページの「CREATE TABLE」を参照してください。
- ネストした表の詳細は、『Oracle Database オブジェクト・リレーショナル開発者ガイド』を参照してください。

REF 列の例 : 次の文では、オブジェクト型 dept_t を作成し、その後に表 staff を作成します。

```
CREATE TYPE dept_t AS OBJECT
   (deptno NUMBER, dname VARCHAR2(20));
/

CREATE TABLE staff
   (name   VARCHAR(100),
    salary NUMBER,
    dept   REF dept_t);
```

オブジェクト表 offices を次のように作成します。

```
CREATE TABLE offices OF dept_t;
```

dept 列は、任意の表に格納された dept_t のオブジェクトに参照を格納できます。次のように dept 列に有効範囲制約を追加することによって、departments 表に格納されたオブジェクトのみが参照されるように制限できます。

```
ALTER TABLE staff
    ADD (SCOPE FOR (dept) IS offices);
```

前述の ALTER TABLE 文は、staff 表が空である場合のみ正常に実行されます。

次の文は、staff の dept 列に REF 値を格納する際、ROWID も同時に格納します。

```
ALTER TABLE staff
   ADD (REF(dept) WITH ROWID);
```

追加の例 : ALTER TABLE 文を使用した整合性制約の定義例は、8-4 ページの「constraint」を参照してください。

表の記憶域パラメータの変更例は、8-41 ページの「storage_clause」を参照してください。

ALTER TABLESPACE

用途

ALTER TABLESPACE を使用すると、既存の表領域、1 つ以上のデータ・ファイルまたは一時ファイルを変更できます。

この句では、ディクショナリ管理表領域をローカル管理表領域に変換することはできません。変換するには、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』に記載されている DBMS_SPACE_ADMIN パッケージを使用してください。

> **参照：** 表領域作成の詳細は、『Oracle Database 管理者ガイド』および16-71 ページの「CREATE TABLESPACE」を参照してください。

前提条件

SYSAUX 表領域を変更する場合は、SYSDBA システム権限が必要です。

ALTER TABLESPACE システム権限を持っている場合、すべての ALTER TABLESPACE 操作を実行できます。MANAGE TABLESPACE システム権限を持っている場合は、次の操作のみを実行できます。

- 表領域をオンラインまたはオフラインにする。
- バックアップを開始または終了する。
- 表領域を読取り専用または読み書き両用にする。

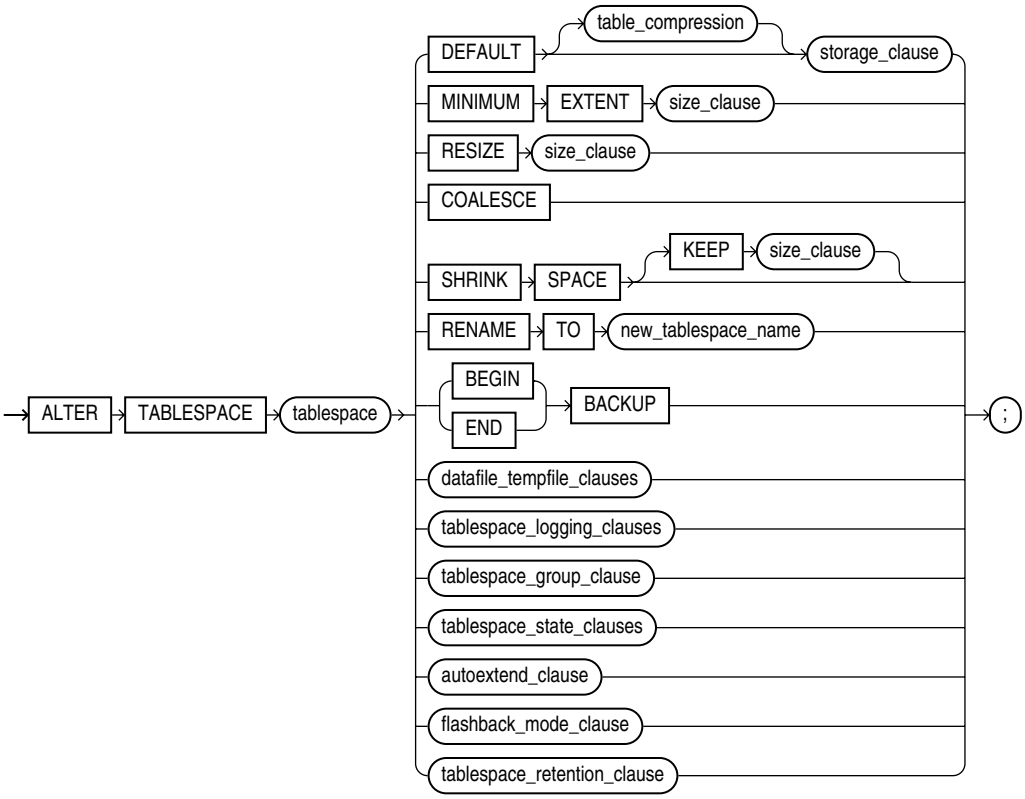
表領域を読取り専用にする場合、次の条件が満たされている必要があります。

- 表領域がオンラインになっている。
- 表領域にアクティブなロールバック・セグメントがない。SYSTEM 表領域には SYSTEM ロールバック・セグメントがあるため、読取り専用にはできません。また、読取り専用表領域のロールバック・セグメントにはアクセスできないため、ロールバック・セグメントを削除してから、表領域を読取り専用にすることをお薦めします。
- 表領域がオープン・バックアップに使用されていない。バックアップの終わりに表領域内のすべてのデータ・ファイルのヘッダー・ファイルが更新されるためです。

これらの条件を満たすために、制限モードでこの機能を実行すると有効です。制限モードでは、RESTRICTED SESSION システム権限を持つユーザーのみがログインできます。

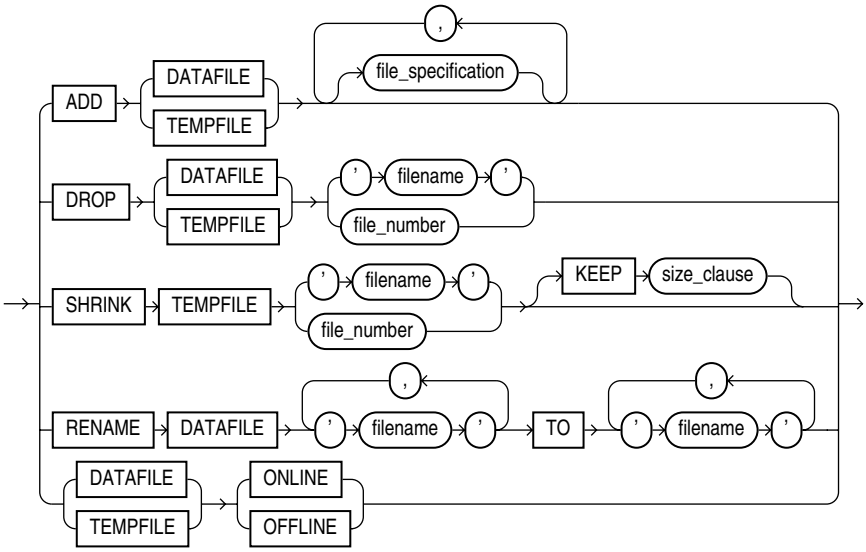
構文

***alter_tablespace*::=**



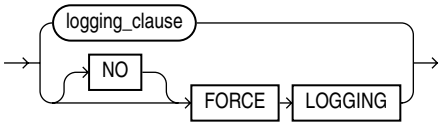
（「ALTER TABLE」の項にある 12-5 ページの table_compression::=、
8-44 ページの storage_clause::=、8-42 ページの size_clause::=、
12-83 ページの datafile_tempfile_clauses::=、12-84 ページの tablespace_logging_clauses::=、
12-84 ページの tablespace_group_clause::=、12-84 ページの tablespace_state_clauses::=、
12-84 ページの autoextend_clause::=、12-84 ページの flashback_mode_clause::=、
12-85 ページの tablespace_retention_clause::= を参照）

datafile_tempfile_clauses::=



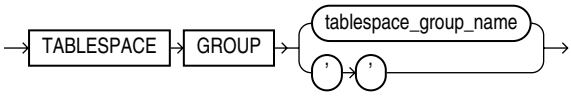
（8-26 ページの file_specification::= を参照）

tablespace_logging_clauses::=

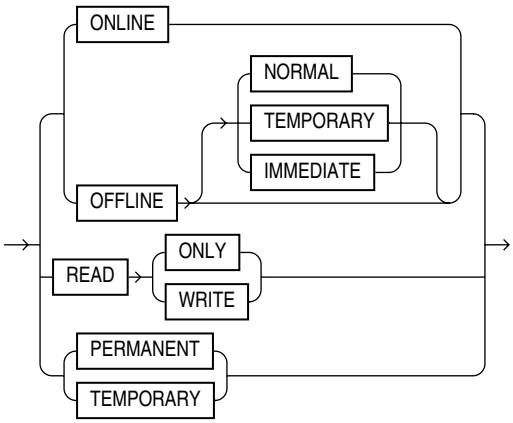


(8-34 ページの logging_clause::= を参照)

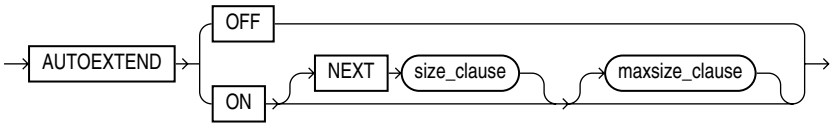
tablespace_group_clause::=



tablespace_state_clauses::=

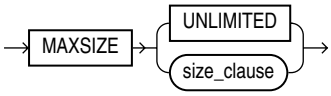


autoextend_clause::=



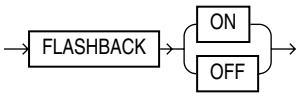
(8-42 ページの size_clause::= を参照)

***maxsize_clause*::=**

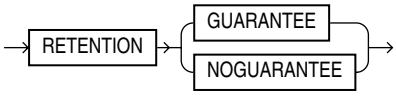


(8-42 ページの size_clause::= を参照)

flashback_mode_clause::=



***tablespace_retention_clause*::=**



セマンティクス

tablespace

変更する表領域の名前を指定します。

表領域の変更の制限事項： 表領域の変更には、次の制限事項があります。

- *tablespace* が UNDO 表領域の場合、この文では ADD DATAFILE、RENAME DATAFILE、RENAME TO（表領域の名前の変更）、DATAFILE ... ONLINE、DATAFILE ... OFFLINE、BEGIN BACKUP および END BACKUP のみが指定可能です。
- SYSTEM 表領域を、読取り専用または一時表領域にしたり、オフラインにすることはできません。
- ローカル管理の一時表領域に対してこの文で指定できるのは、ADD 句のみです。

> **参照：** 自動 UNDO 管理および UNDO 表領域の詳細は、『Oracle Database 管理者ガイド』を参照してください。

DEFAULT *storage_clause*

DEFAULT *storage_clause* を使用すると、表領域に作成される後続のオブジェクトに対して新しいデフォルトの記憶域パラメータを指定できます。ディクショナリ管理の一時表の場合は、*storage_clause* の NEXT パラメータのみが考慮されます。

詳細は、8-41 ページの「storage_clause」を参照してください。

表領域のデフォルト記憶域の制限事項： この句は、ローカル管理表領域に対して指定できません。

MINIMUM EXTENT

この句は、永続的なディクショナリ管理表領域に対してのみ有効です。MINIMUM EXTENT 句を指定すると、表領域内のすべての使用済エクステントまたは未使用エクステントの大きさが、*size_clause* で指定したサイズ以上であること、およびその倍数であることが保証され、表領域における空き領域の断片化を制御できます。

MINIMUM EXTENT の制限事項： この句は、ローカル管理の表領域またはディクショナリ管理の一時表領域に対して指定できません。

> **参照：** この句の詳細は、8-42 ページの「size_clause」を参照してください。MINIMUM EXTENT を使用した領域の断片化の制御の詳細は、『Oracle Database 管理者ガイド』を参照してください。

RESIZE 句

この句は、bigfile 表領域に対してのみ有効です。この句を使用すると、1 つのデータ・ファイルのサイズを絶対サイズの値まで増やしたり減らすことができます。K、M、G または T を使用して、それぞれ KB、MB、GB または TB 単位で指定することもできます。

smallfile 表領域に新しく追加されたデータ・ファイルまたは一時ファイルのサイズを変更するには、ALTER DATABASE ... *autoextend_clause* を使用します（10-25 ページの「database_file_clauses」を参照）。

> **参照：** bigfile 表領域については、16-74 ページの「BIGFILE | SMALLFILE」を参照してください。

COALESCE

この句を使用すると、表領域内の各データ・ファイルで、連続する未使用エクステントをすべて結合し、連続するより大きいエクステントを作成します。

SHRINK SPACE 句

この句は一時表領域に対してのみ有効です。表領域が使用する領域を削減できます。オプションの KEEP 句では、*size_clause* によって、表領域を縮小できる下限が定義されます。これは、自動拡張可能な表領域の MAXSIZE とは反対の機能です。KEEP 句を省略すると、表領域の他の記憶域属性を満たしている場合、データベースはできるだけ表領域を縮小しようとします。

RENAME 句

この句を指定すると、*tablespace* の名前を変更できます。この句は、*tablespace* およびそのすべてのデータ・ファイルがオンラインで、COMPATIBLE パラメータが 10.0.0 以上に設定されている場合にのみ有効です。名前の変更は、永続表領域および一時表領域の両方に対して実行できます。

tablespace が読取り専用の場合、データ・ファイルのヘッダーは更新されず、新しい名前は反映されません。アラート・ログに、データ・ファイルのヘッダーが更新されなかったことが記録されます。

注意： バックアップからリストアしたデータ・ファイルを使用して制御ファイルを再作成する場合、データ・ファイルのヘッダーに古い表領域の名前が反映されていると、再作成された制御ファイルにも古い表領域の名前が反映されます。ただし、データベースが完全にリカバリされた後は、制御ファイルに新しい名前が反映されます。

---

tablespace が、Oracle Real Application Clusters（RAC）環境のインスタンスに対する UNDO 表領域として指定されており、データベースの起動にサーバー・パラメータ・ファイルが使用されている場合、そのサーバー・パラメータ・ファイル（SPFILE）で、インスタンスに対する UNDO_TABLESPACE パラメータの値は、新しい表領域の名前を反映するように変更されます。単一インスタンス・データベースで、spfile のかわりにパラメータ・ファイル（pfile）が使用されている場合、データベース管理者に pfile 内の値を手動で変更することを推奨するメッセージがアラート・ログに書き込まれます。

表領域の名前の変更の制限事項： SYSTEM 表領域または SYSAUX 表領域の名前は変更できません。

BACKUP 句

この句を使用すると、表領域のすべてのデータ・ファイルをオンライン（ホット）・バックアップ・モードにしたり、このモードから戻すことができます。

参照：

- メディア・リカバリなしでデータベースを再起動する場合の詳細は、『Oracle Database 管理者ガイド』を参照してください。
- データベース内のすべてのデータ・ファイルに対するオンライン・バックアップ・モードの設定または解除の詳細は、10-24 ページの「ALTER DATABASE」の「BACKUP 句」を参照してください。
- 個々のデータ・ファイルをオンライン・バックアップ・モードから解除する場合の詳細は、10-26 ページの「ALTER DATABASE」の「alter_datafile_clause」を参照してください。

BEGIN BACKUP

BEGIN BACKUP を指定すると、表領域を構成するデータ・ファイルのオープン・バックアップを実行することを示すことができます。この句を指定することによって、ユーザーがこの表領域にアクセスできなくなることはありません。オープン・バックアップを開始する前に、この句を指定してください。

表領域のバックアップ開始の制限事項 : 表領域のバックアップ開始には、次の制限事項があります。

- この句は、読取り専用の表領域またはローカル管理の一時表領域に対して指定できません。
- バックアップ中は、表領域のオフラインへの正常な切替え、インスタンスの停止または表領域の別のバックアップ処理の開始は実行できません。

参照 : 「表領域のバックアップ例 :」(12-92 ページ)

END BACKUP

END BACKUP を指定すると、表領域のオンライン・バックアップが完了したことを示すことができます。オンライン・バックアップの完了後、できるだけ早くこの句を指定してください。インスタンスに障害または SHUTDOWN ABORT が発生した場合、次のインスタンス起動時にメディア・リカバリ（必要に応じて、アーカイブ REDO ログも）が必要であるとみなされます。

表領域のバックアップ終了の制限事項 : この句は、読取り専用表領域に対して使用できません。

datafile_tempfile_clauses

datafile_tempfile_clauses を使用すると、データ・ファイルまたは一時ファイルを追加および変更できます。

ADD 句

ADD を指定すると、*file_specification* によって指定されたデータ・ファイルまたは一時ファイルを表領域に追加できます。オペレーティング・システムのファイル・システム内の標準データファイルと一時ファイル、または自動ストレージ管理ディスク・グループのファイルを指定するには、*file_specification* の datafile_tempfile_spec 書式（8-26 ページの「file_specification」を参照）を使用します。

この句は、ローカル管理の一時表領域に対して、どんな場合でも指定できる唯一の句です。

file_specification を指定しないと、AUTOEXTEND が有効になった 100MB の Oracle 管理ファイルが作成されます。

データ・ファイルまたは一時ファイルを、オンラインのローカル管理表領域、またはオンラインまたはオフラインのディクショナリ管理表領域に追加できます。なお、そのデータ・ファイルが別のデータベースで使用中でないことを確認してください。

データ・ファイルおよび一時ファイルの追加の制限事項 : この句は、表領域に 1 つのデータ・ファイルまたは一時ファイルのみが含まれる bigfile（単一ファイル）表領域に対して指定できません。

注意： オペレーティング・システムによっては、一時ファイルのブロックが実際にアクセスされるまで、一時ファイル用の領域が割り当てられない場合があります。領域の割当ての遅延のため、一時ファイルの作成およびサイズ変更が速くなります。ただし、後で一時ファイルが使用されるときに、十分なディスク領域を使用可能にする必要があります。発生する可能性がある問題を回避するには、一時ファイルを作成またはサイズ変更する前に、ディスク領域が、新しく作成する一時ファイルまたはサイズ変更後の一時ファイルのサイズより大きいことを確認してください。ディスク領域に余裕を持たせることによって、関連のない操作が原因で増加が予想されるディスク使用量に対応できます。十分な領域があることを確認した後で、作成またはサイズ変更操作を実行してください。

参照： 8-26 ページの「file_specification」、12-92 ページの「データ・ファイルおよび一時ファイルの追加例 :」および 12-93 ページの「Oracle Managed Files のデータ・ファイルの追加例 :」を参照してください。

DROP 句

DROP を指定すると、*filename* や *file_number* によって指定された空のデータ・ファイルまたは一時ファイルを表領域から削除できます。この句は、データ・ファイルまたは一時ファイルをデータ・ディクショナリから削除し、オペレーティング・システムから削除します。この句を指定するときには、データベースがオープンしている必要があります。

ALTER TABLESPACE ... DROP TEMPFILE 文は、ALTER DATABASE TEMPFILE ... DROP INCLUDING DATAFILES を指定することと同じです。

ファイル削除の制限事項： データ・ファイルまたは一時ファイルを削除するには、これらのファイルが次の条件を満たしている必要があります。

- 空であること。
- 表領域内で最初に作成されたファイルではないこと。この場合、かわりに表領域が削除されます。
- 読取り専用の表領域内にないこと。

参照：

- 一時ファイルの削除の詳細は、10-27 ページの「ALTER DATABASE」の「alter_tempfile_clause」を参照してください。
- データ・ファイル番号の詳細と、データ・ファイルの管理ガイドラインの詳細は、『Oracle Database 管理者ガイド』を参照してください。
- 「データ・ファイルおよび一時ファイルの追加例 :」(12-92 ページ)

SHRINK TEMPFILE 句

この句は一時表領域を変更する場合にのみ有効です。指定した一時ファイルが使用する領域を削減できます。オプションの KEEP 句では、*size_clause* によって、一時ファイルを縮小できる下限が定義されます。これは、自動拡張可能な表領域の MAXSIZE とは反対の機能です。KEEP 句を省略すると、他の記憶域属性を満たしている場合、データベースはできるだけ一時ファイルを縮小しようとします。

RENAME DATAFILE 句

RENAME DATAFILE を指定すると、表領域の 1 つ以上のデータ・ファイルの名前を変更できます。データベースをオープンしておくこと、および名前の変更前に表領域をオフラインにすることが必要です。それぞれの *filename* には、ご使用のオペレーティング・システムのファイル名の表記規則に従って、データ・ファイル名を完全に指定してください。

この句では、表領域を古いファイルではなく新しいファイルに対応付けます。オペレーティング・システムのファイル名は実際には変更されません。このため、オペレーティング・システム上でこのファイル名を変更する必要があります。

> **参照：** 「表領域の移動および名前の変更例 :」（12-92 ページ）

ONLINE | OFFLINE 句

これらの句を使用すると、表領域のすべてのデータ・ファイルまたは一時ファイルを、オフラインまたはオンラインにできます。これらの句は、表領域の ONLINE または OFFLINE 状態には影響しません。

データベースは、マウントされている必要があります。*tablespace* が SYSTEM、UNDO 表領域、またはデフォルトの一時表領域の場合、データベースをオープンしないでおく必要があります。

tablespace_logging_clauses

この句を使用すると、表領域のロギング特性を設定または変更できます。

logging_clause

LOGGING を指定すると、表領域内のすべての表、索引およびパーティションのロギング属性を指定できます。表レベル、索引レベルおよびパーティション・レベルでのロギング指定によって、表領域レベルのロギング属性を上書きできます。

既存の表領域のロギング属性を ALTER TABLESPACE 文によって変更した場合、この文の実行後に作成されたすべての表、索引およびパーティションに、新しいデフォルトのロギング属性（これは後で上書きもできます）が適用されます。既存のオブジェクトのロギング属性は変更されません。

FORCE LOGGING モードの表領域がある場合、この文で NOLOGGING を指定すると、表領域のデフォルト・ロギング・モードを NOLOGGING に設定できます。ただし、この設定によって表領域の FORCE LOGGING モードは解除されません。

[NO] FORCE LOGGING

この句を使用すると、表領域で強制ロギング・モードを有効または無効にできます。データベースをオープンし、READ WRITE モードにしておく必要があります。この設定により、表領域のデフォルト LOGGING モードまたは NOLOGGING モードは変更されません。

強制ロギング・モードの制限事項： FORCE LOGGING は、UNDO 表領域および一時表領域に対して指定できません。

> **参照：** FORCE LOGGING モードの使用方法の詳細は、『Oracle Database 管理者ガイド』および 12-93 ページの「表領域のロギング属性の変更例 :」を参照してください。

tablespace_group_clause

この句は、ローカル管理の一時表領域に対してのみ有効です。この句を使用すると、*tablespace_group_name* 表領域グループに対して *tablespace* を追加または削除できます。

- グループ名を指定すると、*tablespace* がその表領域グループのメンバーであることを示すことができます。*tablespace_group_name* が存在しない場合、表領域を変更して表領域グループのメンバーにすると、その表領域グループが暗黙的に作成されます。
- 空の文字列（''）を指定すると、*tablespace_group_name* 表領域グループから *tablespace* を削除できます。

表領域グループの制限事項： 表領域グループは、永続表領域またはディクショナリ管理の一時表領域には指定できません。

> **参照：** 表領域グループの詳細は、『Oracle Database 管理者ガイド』を参照してください。13-10 ページの「表領域グループの割当て例：」も参照してください。

tablespace_state_clauses

この句を使用すると、表領域の状態を設定または変更できます。

ONLINE | OFFLINE

ONLINE を指定すると、表領域をオンラインにできます。OFFLINE を指定すると、表領域をオフラインにし、そのセグメントへの後続のアクセスを禁止できます。表領域をオフラインにすると、そのすべてのデータ・ファイルもオフラインになります。

提案： 表領域を長期間オフラインにする前に、デフォルト表領域または一時表領域としてその表領域が割り当てられているユーザーに対して、表領域の割当てを変更することを検討します。表領域をオフラインにしている間は、これらのユーザーは、その表領域内でオブジェクトに対して領域を割り当てたり、領域をソートすることはできません。ユーザーへの表領域の割当ての詳細は、13-5 ページの「ALTER USER」を参照してください。

表領域をオフラインにする場合の制限事項： 一時表領域はオフラインにできません。

OFFLINE NORMAL NORMAL を指定すると、システム・グローバル領域（SGA）以外にある表領域のすべてのデータ・ファイルにあるすべてのブロックをフラッシュできます。データ・ファイルをオンラインに戻す前に、表領域のメディア・リカバリを行う必要はありません。これはデフォルトです。

OFFLINE TEMPORARY TEMPORARY を指定すると、Oracle Database は表領域内のすべてのオンライン・データ・ファイルに対してチェックポイントを実行しますが、すべてのファイルに対して書込みを実行できるかどうかは保証しません。この文の発行時にオフラインであったファイルは、表領域をオンラインに戻す前に、メディア・リカバリを行う必要があります。

OFFLINE IMMEDIATE IMMEDIATE を指定すると、Oracle Database は表領域のファイルが使用可能であることを保証しません。また、チェックポイントも実行しません。表領域をオンラインに戻す前に、メディア・リカバリを行う必要があります。

注意： ALTER TABLESPACE ... OFFLINE に対する FOR RECOVER 設定は、非推奨になっています。この構文は、下位互換性を保つためにのみサポートされています。ただし、表領域のリカバリにはトランスポータブル表領域機能を使用することをお薦めします。

> **参照：** メディア・リカバリを実行するトランスポータブル表領域の使用の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

READ ONLY | READ WRITE

READ ONLY を指定すると、表領域を**読取り専用遷移モード**に設定できます。この状態では、既存のトランザクションは完了（コミットまたはロールバック）できますが、表領域内のブロックを変更した既存のトランザクションをロールバックすること以外は、その表領域に対してさらに DML 操作を行うことはできません。SYSAUX 表領域は、READ ONLY に設定できません。

表領域が読取り専用の場合は、そのファイルを読取り専用メディアにコピーできます。その場合、SQL 文の ALTER DATABASE ... RENAME を使用して、新しいファイル位置を示すように制御ファイル内のデータ・ファイルの名前を変更する必要があります。

参照 :

- 読取り専用の表領域の詳細は、『Oracle Database 概要』を参照してください。
- 「ALTER DATABASE」（10-9 ページ）

READ WRITE を指定すると、読取り専用に指定されている表領域に対して書込み操作を実行できるようになります。

PERMANENT | TEMPORARY

PERMANENT を指定すると、一時表領域を永続表領域に変換できます。永続表領域とは、永続的なデータベース・オブジェクトを格納できる場所です。表領域を作成するときのデフォルトです。

TEMPORARY を指定すると、永続表領域を一時表領域に変換できます。一時表領域とは、永続的なデータベース・オブジェクトを格納できない表領域です。一時表領域の中のオブジェクトはセッション中のみ保持されます。

一時表領域の制限事項 : 一時表領域には、次の制限事項があります。

- SYSAUX 表領域には、TEMPORARY を指定できません。
- *tablespace* を標準的なブロック・サイズで作成しなかった場合、永続表領域を一時表領域に変換できません。
- FORCE LOGGING モードでは、表領域に対して TEMPORARY を指定できません。

autoextend_clause

この句は、bigfile（単一ファイル）表領域に対してのみ有効です。この句を使用すると、表領域内の単一のデータ・ファイルに対して自動拡張を使用可能または使用禁止にできます。smallfile 表領域に新しく追加されたデータ・ファイルまたは一時ファイルの自動拡張を使用可能または使用禁止にするには、ALTER DATABASE 文で *autoextend_clause* を使用します（10-25 ページの「database_file_clauses」を参照）。

参照 :

- bigfile（単一ファイル）表領域については、『Oracle Database 管理者ガイド』を参照してください。
- *autoextend_clause* の詳細は、8-26 ページの「file_specification」を参照してください。

flashback_mode_clause

この句を使用すると、後続の FLASHBACK DATABASE 操作でこの表領域を使用するかどうかを指定できます。

- FLASHBACK モードをオンにするには、データベースはマウント済（オープン状態またはクローズ状態）である必要があります。
- FLASHBACK モードをオフにするには、データベースはマウント済で、クローズ状態である必要があります。

この句は一時表領域では無効です。

この句の詳細は、16-71 ページの「CREATE TABLESPACE」を参照してください。

参照： データベースのフラッシュバックの詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

tablespace_retention_clause

この句のセマンティクスは、CREATE TABLESPACE 文および ALTER TABLESPACE 文で同じです。16-81 ページの「CREATE TABLESPACE」の「tablespace_retention_clause」を参照してください。

例

表領域のバックアップ例： 次の文は、バックアップの開始をデータベースに通知します。

```
ALTER TABLESPACE tbs_01
    BEGIN BACKUP;
```

次の文は、バックアップが終了したことをデータベースに通知します。

```
ALTER TABLESPACE tbs_01
   END BACKUP;
```

表領域の移動および名前の変更例： 次の例は、tbs_02 表領域（16-83 ページの「表領域の自動拡張を使用可能にする場合の例：」で作成）に関連付けられたデータ・ファイルを、diskb:tbs_f5.dat から diska:tbs_f5.dat に移動して、名前を変更します。

1. OFFLINE 句を指定した ALTER TABLESPACE 文を使用して、この表領域をオフラインにします。

   ```
   ALTER TABLESPACE tbs_02 OFFLINE NORMAL;
   ```

2. オペレーティング・システムのコマンドを使用して、このファイルを diskb:tbs_f5.dat から diska:tbs_f5.dat にコピーします。

3. RENAME DATAFILE 句を指定した ALTER TABLESPACE 文を使用して、このデータ・ファイルの名前を変更します。

   ```
   ALTER TABLESPACE tbs_02
     RENAME DATAFILE 'diskb:tbs_f5.dat'
     TO              'diska:tbs_f5.dat';
   ```

4. ONLINE 句を指定した ALTER TABLESPACE 文を使用して、この表領域をオンラインに戻します。

   ```
   ALTER TABLESPACE tbs_02 ONLINE;
   ```

データ・ファイルおよび一時ファイルの追加例と削除例： 次の文は、表領域にデータ・ファイルを追加します。さらに多くの領域が必要な場合、10KB の新しいエクステントが最大 100KB まで追加されます。

```
ALTER TABLESPACE tbs_03
    ADD DATAFILE 'tbs_f04.dbf'
    SIZE 100K
    AUTOEXTEND ON
    NEXT 10K
    MAXSIZE 100K;
```

次の文は、空のデータ・ファイルを削除します。

```
ALTER TABLESPACE tbs_03
    DROP DATAFILE 'tbs_f04.dbf';
```

次の文は、16-82 ページの「一時表領域の作成例 :」で作成された一時表領域に一時ファイルを追加してから削除します。

```
ALTER TABLESPACE temp_demo ADD TEMPFILE 'temp05.dbf' SIZE 5 AUTOEXTEND ON;

ALTER TABLESPACE temp_demo DROP TEMPFILE 'temp05.dbf';
```

一時表領域の領域の管理例 : 次の文は、SHRINK SPACE 句を使用して、一時表領域（16-82 ページの「一時表領域の作成例 :」で作成）内の領域を管理します。KEEP 句を省略します。これにより、表領域の他の記憶域属性を満たしている場合、データベースはできるだけ表領域を縮小しようとします。

```
ALTER TABLESPACE temp_demo SHRINK SPACE;
```

Oracle Managed Files のデータ・ファイルの追加例 : 次の例は、Oracle Managed Files のデータ・ファイルを omf_ts1 表領域に追加します（この表領域の作成の詳細は、16-84 ページの「Oracle Managed Files の作成例 :」を参照してください）。新しいデータ・ファイルは 100MB で、最大サイズが制限なしで自動拡張されます。

```
ALTER TABLESPACE omf_ts1 ADD DATAFILE;
```

表領域のロギング属性の変更例 : 次の例は、表領域のデフォルトのロギング属性を NOLOGGING に変更します。

```
ALTER TABLESPACE tbs_03 NOLOGGING;
```

表領域のロギング属性を変更した場合でも、その表領域内の既存のスキーマ・オブジェクトのロギング属性には影響しません。表レベル、索引レベルおよびパーティション・レベルでのロギング指定によって、表領域レベルのロギング属性を上書きできます。

UNDO データの保持の変更例 : 次の文は、undots1 表領域の UNDO データの保持を、通常の UNDO データの動作に変更します。

```
ALTER TABLESPACE undots1
  RETENTION NOGUARANTEE;
```

次の文は、undots1 表領域の UNDO データの保持を、期限が切れていない UNDO データを保持する動作に変更します。

```
ALTER TABLESPACE undots1
  RETENTION GUARANTEE;
```

13

SQL 文 : ALTER TRIGGER ～ COMMIT

この章では、次の SQL 文について説明します。

- ALTER TRIGGER
- ALTER TYPE
- ALTER USER
- ALTER VIEW
- ANALYZE
- ASSOCIATE STATISTICS
- AUDIT
- CALL
- COMMENT
- COMMIT

ALTER TRIGGER

用途

トリガーは PL/SQL を使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

ALTER TRIGGER 文を使用すると、データベース・トリガーを使用可能化、使用禁止化またはコンパイルできます。

注意： この文を使用して既存のトリガーの宣言や定義は変更できません。トリガーを再宣言または再定義する場合は、OR REPLACE キーワードを指定した CREATE TRIGGER 文を使用します。

参照：

- トリガーの作成については、16-85 ページの「CREATE TRIGGER」を参照してください。
- トリガーの削除については、18-11 ページの「DROP TRIGGER」を参照してください。
- トリガーの概要は、『Oracle Database 概要』を参照してください。

前提条件

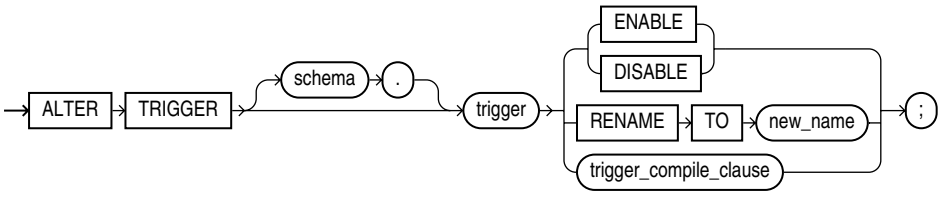
トリガーが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY TRIGGER システム権限が必要です。

DATABASE 上のトリガーを変更する場合は、ADMINISTER データベース・イベント・システム権限が必要です。

参照： DATABASE トリガーに基づいたトリガーの詳細は、16-85 ページの「CREATE TRIGGER」を参照してください。

構文

***alter_trigger*::=**



(*trigger_compile_clause*: この句の構文の詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。)

セマンティクス

schema

トリガーが含まれているスキーマを指定します。*schema* を指定しない場合、Oracle Database ではトリガーは自分のスキーマ内にあるとみなされます。

trigger

変更するトリガーの名前を指定します。

ENABLE | DISABLE

ENABLE を指定すると、トリガーを使用可能にできます。また、ALTER TABLE の ENABLE ALL TRIGGERS 句を使用することによって、表に対応付けられたすべてのトリガーを使用可能にできます。12-2 ページの「ALTER TABLE」を参照してください。

DISABLE を指定すると、トリガーを使用禁止にできます。また、ALTER TABLE の DISABLE ALL TRIGGERS 句を使用することによって、表に対応付けられたすべてのトリガーを使用禁止にできます。

RENAME 句

RENAME TO *new_name* を指定すると、トリガーの名前を変更できます。トリガーの名前は変更され、名前が変更される前と同じ状態になります。

トリガーの名前を変更すると、USER_SOURCE、ALL_SOURCE および DBA_SOURCE データ・ディクショナリ・ビューに記憶されているトリガーのソースが再構築されます。その結果、トリガー・ソースが変更されていなくても、これらのビューの TEXT 列のコメントおよび書式設定が変更される場合があります。

trigger_compile_clause

この句の構文とセマンティクスの詳細およびトリガーの作成とコンパイルの詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

ALTER TYPE

用途

オブジェクト型は PL/SQL を使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

ALTER TYPE 文を使用すると、メンバー属性またはメソッドを追加または削除できます。オブジェクト型の既存のプロパティ（FINAL または INSTANTIABLE）、または型のスカラー属性も変更できます。

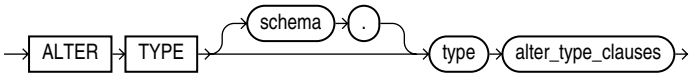
この文を使用すると、新しいオブジェクト・メンバーのサブプログラム仕様を追加することによって、型の仕様部または本体を再コンパイルしたり、オブジェクト型の仕様を変更することができます。

前提条件

オブジェクト型が自分のスキーマ内にあり、CREATE TYPE か CREATE ANY TYPE 権限を持っている必要があります。または、ALTER ANY TYPE システム権限が必要です。

構文

***alter_type*::=**



（*alter_type_clauses*: この句の構文の詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。）

セマンティクス

schema

型が含まれているスキーマを指定します。*schema* を指定しない場合、この型は現行のスキーマ内にあるとみなされます。

type

オブジェクト型、ネストした表型または VARRAY 型の名前を指定します。

alter_type_clauses

この句の構文とセマンティクスの詳細およびオブジェクト型の作成とコンパイルの詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

ALTER USER

用途

ALTER USER を使用すると、次の操作を実行できます。

- データベース・ユーザーの認証またはデータベース・リソースの特性を変更します。
- プロキシ・サーバーが認証なしでクライアントとして接続することを許可します。
- 自動ストレージ管理クラスタで、ユーザーのパスワードを現在のノードの自動ストレージ管理インスタンスに対してローカルなパスワード・ファイルで変更します。

> **参照：** ユーザーの認証方式の詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。

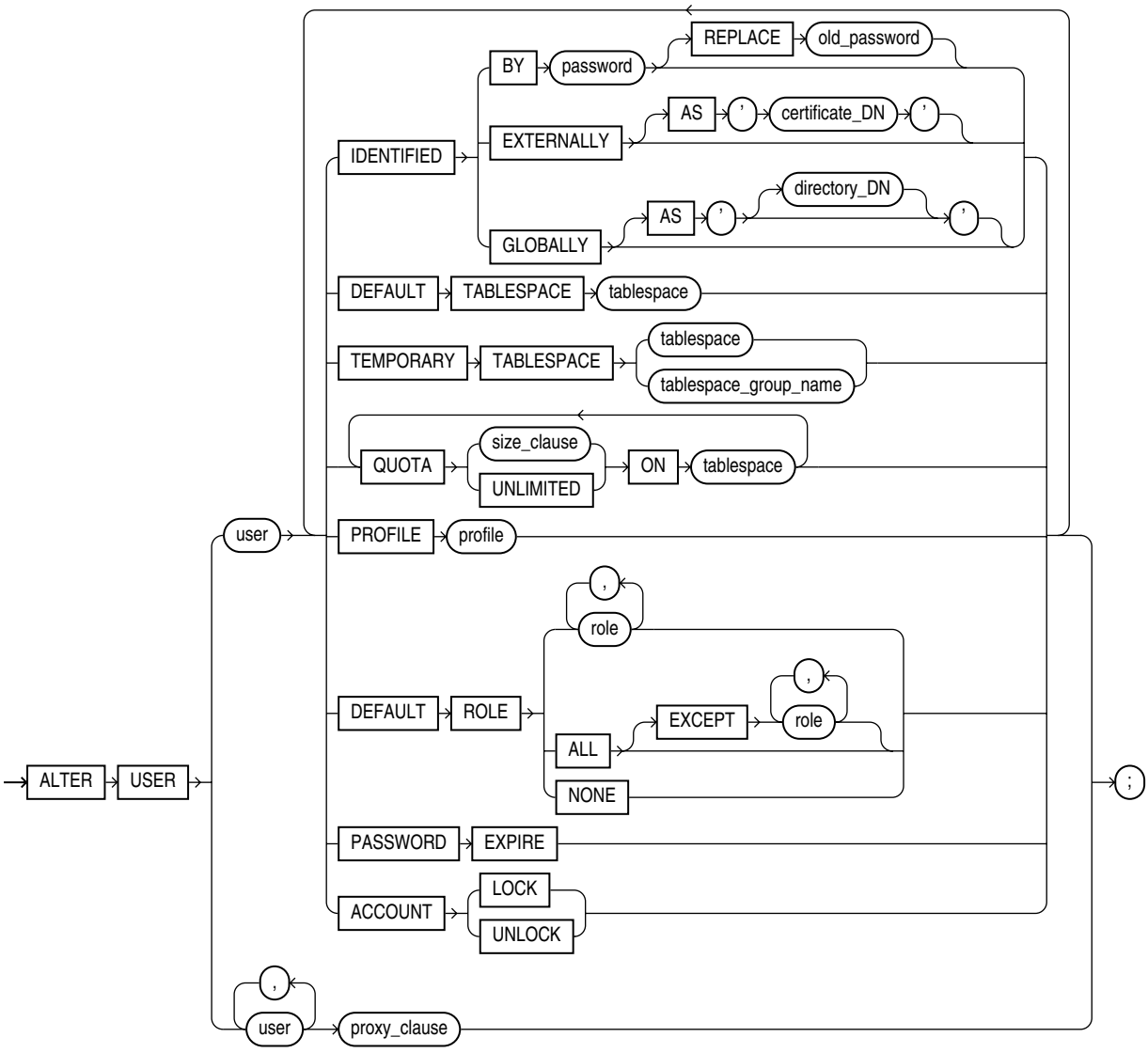
前提条件

ALTER USER システム権限が必要です。ただし、ユーザー自身のパスワードはこの権限がない場合でも変更できます。

自動ストレージ管理インスタンス・パスワード・ファイルで自分以外のユーザーのパスワードを変更するには、AS SYSASM として認証されている必要があります。

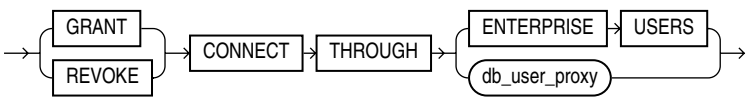
構文

***alter_user*::=**

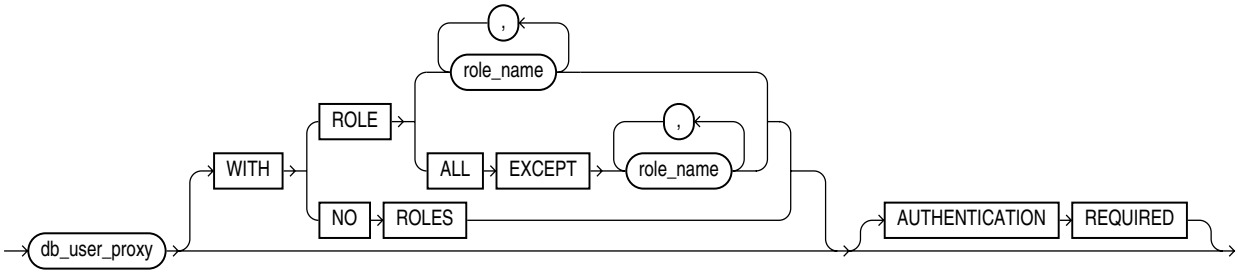


(8-42 ページの size_clause::= を参照)

proxy_clause::=



***db_user_proxy*::=**



セマンティクス

この項で説明するキーワード、パラメータおよび句は、ALTER USER 独自のものと、CREATE USER にもあるが、セマンティクスが異なるものがあります。その他のキーワード、パラメータおよび句には、CREATE USER 文と同じ意味があります。

注意： ユーザー名およびパスワードは、ご使用のプラットフォームに応じて、ASCII または EBCDIC 文字のみでエンコードすることをお薦めします。

参照： キーワードおよびパラメータについては、17-7 ページの「CREATE USER」を参照してください。ユーザーにデータベース・リソースへの制限を割り当てる方法については、15-47 ページの「CREATE PROFILE」を参照してください。

IDENTIFIED 句

BY *password* BY *password* を指定すると、ユーザーの新しいパスワードを指定できます。パスワードは、大 / 小文字が区別されます。この後に、ユーザーをデータベースに接続するために使用される CONNECT 文字列は、この ALTER USER 文で使用されているものと同じ文字（大文字、小文字または混在）を使用してパスワードを指定する必要があります。パスワードには、データベース・キャラクタ・セットから、シングルバイト文字、マルチバイト文字または両方を含めることができます。

注意： 異なるタイムスタンプで特定のパスワードを再設定する必要があります。1 秒以内に 1 つのパスワードを複数回再設定した場合（たとえば、スクリプトを使用して一連のパスワードの設定を繰り返した場合）、データベースはパスワードが再利用できないというエラー・メッセージを戻すことがあります。このため、パスワードの再設定には、スクリプトを使用しないことをお薦めします。

自分のパスワードを初めて設定する場合、または ALTER USER システム権限を持っていて、他のユーザーのパスワードを変更する場合は、REPLACE 句を省略できます。ただし、ALTER USER システム権限を持たないかぎり、複雑なパスワードの検証機能が使用可能な場合は、UTLPWDMG.SQL スクリプトを実行するか、またはユーザーに割り当てられたプロファイルの PASSWORD_VERIFY_FUNCTION パラメータに検証機能を指定して、常に REPLACE 句を指定する必要があります。

自動ストレージ管理クラスタで、この句を使用して、ユーザーのパスワードを、現行のノードの自動ストレージ管理インスタンスに対してローカルなパスワード・ファイルで変更できます。REPLACE *old_password* 句を使用せずに IDENTIFIED BY *password* を指定するには、AS SYSASM として認証されている必要があります。AS SYSASM として認証されていない場合は、REPLACE *old_password* を指定して自分のパスワードのみを変更できます。

以前のパスワードを REPLACE 句に指定した場合でも、自分以外の既存のパスワードを変更している場合は、以前のパスワードはチェックされません。

> **参照：** 複雑なパスワードの検証機能については、『Oracle Database セキュリティ・ガイド』を参照してください。

GLOBALLY この句の詳細は、17-7 ページの「CREATE USER」を参照してください。

ユーザー・アクセスの検証方法は、IDENTIFIED GLOBALLY から IDENTIFIED BY *password* または IDENTIFIED EXTERNALLY のいずれかに変更できます。ユーザーのアクセス検証方法を他のいずれかの検証方法から IDENTIFIED GLOBALLY に変更できるのは、ユーザーに明示的に付与されたすべての外部ロールが取り消された場合のみです。

EXTERNALLY この句の詳細は、17-7 ページの「CREATE USER」を参照してください。

> **参照：** グローバルまたは外部で識別されるユーザーの詳細は、『Oracle Database エンタープライズ・ユーザー・セキュリティ管理者ガイド』を参照してください。また、13-10 ページの「ユーザー識別の変更例 :」および 13-10 ページの「ユーザー認証の変更例 :」も参照してください。

DEFAULT TABLESPACE 句

この句を使用すると、ユーザーの永続セグメントの表領域の割当てまたは再割当てを行うことができます。この句は、データベース用に指定されているデフォルトの表領域を上書きします。

デフォルトの表領域の制限事項： ローカル管理の一時表領域（UNDO 表領域を含む）またはディクショナリ管理の一時表領域は、ユーザーのデフォルトの表領域として指定できません。

TEMPORARY TABLESPACE 句

この句を使用すると、ユーザーの一時セグメントの表領域または表領域グループの割当てまたは再割当てを行うことができます。

- *tablespace* に、ユーザーの一時セグメント表領域を指定します。
- *tablespace_group_name* に、ユーザーが一時セグメントを保存できる表領域の表領域グループを指定します。

ユーザーの一時表領域の制限事項： ユーザーの一時表領域として割り当てる表領域、または再度割り当てる表領域は、標準的なブロック・サイズの一時表領域である必要があります。

> **参照：** 「表領域グループの割当て例 :」（13-10 ページ）

DEFAULT ROLE 句

ログイン時にデフォルトによってユーザーに付与されるロールを指定します。この句では、GRANT 文を使用してユーザーに直接付与されているロールのみ指定できます。DEFAULT ROLE 句を使用して次のロールを使用可能にすることはできません。

- ユーザーに付与されていないロール
- 他のロールを介して付与されているロール
- 外部サービス（オペレーティング・システムなど）または Oracle Internet Directory によって管理されるロール

Oracle Database では、ユーザーがパスワードを指定したり、別の方法で認証を行わなくても、ログイン時にデフォルトのロールが使用可能になります。ユーザーにアプリケーション・ロールを付与している場合、DEFAULT ROLE ALL EXCEPT *role* 句を使用して、認可済パッケージを使用するアプリケーションを除いて、そのユーザーの後続のログインで使用可能にならないようにするロールを指定する必要があります。

> **参照：** 「CREATE ROLE」（15-56 ページ）

proxy_clause

proxy_clause を使用すると、エンタープライズ・ユーザー（データベースの外側のユーザー）またはデータベース・プロキシ（別のデータベース・ユーザー）が、変更対象のデータベース・ユーザーとして、どのように接続できるようにするかを制御できます。

- ENTERPRISE USER 句を使用すると、*user* を、エンタープライズ・ユーザーがプロキシ使用できるように公開できます。Oracle Internet Directory を担当する管理者は、*user* の代理として作業するエンタープライズ・ユーザーに権限を適切に付与する必要があります。
- *db_user_proxy* 句を使用すると、*user* を、データベース・ユーザー *db_user_proxy* がプロキシ使用できるように公開し、*user* のロールのすべてまたは一部をアクティブにし（またはすべてを非アクティブにし）、認証が必要かどうかを指定できます。アプリケーション・ユーザーのプロキシ認証の詳細は、『Oracle Database アドバンスト・アプリケーション開発者ガイド』を参照してください。

> **参照：** プロキシおよびデータベースの使用については、『Oracle Database セキュリティ・ガイド』および 13-10 ページの「プロキシ・ユーザー例 :」を参照してください。

GRANT | REVOKE

GRANT を指定すると、接続を許可できます。REVOKE を指定すると、接続を禁止できます。

CONNECT THROUGH 句

Oracle Database に接続するプロキシを識別します。AUTHENTICATED USING 句を指定しない場合は、Oracle Database は、ユーザーの認証にプロキシを想定します。

WITH ROLE WITH ROLE *role_name* を使用すると、プロキシは指定したユーザーとして接続でき、*role_name* で指定されたロールのみをアクティブにできます。

WITH ROLE ALL EXCEPT WITH ROLE ALL EXCEPT *role_name* を使用すると、プロキシは指定したユーザーとして接続でき、*role_name* で指定されたロール以外の、このユーザーに対応付けられたすべてのロールをアクティブにできます。

WITH NO ROLES WITH NO ROLES を使用すると、プロキシは指定したユーザーとして接続できますが、接続後にそのユーザーのロールを 1 つでもアクティブにすることは禁止されます。

WITH 句を指定しない場合、指定したユーザーに付与されているすべてのロールが自動的にアクティブになります。

AUTHENTICATION REQUIRED 句 AUTHENTICATION REQUIRED を指定すると、指定されたプロキシを介してユーザーが認証されるときに、ユーザーに認証の資格証明（パスワード）を要求できます。

AUTHENTICATED USING この句は、必要なくなりました。現在は非推奨になっています。コードで使用した場合は無視されます。*proxy_clause* は、AUTHENTICATION REQUIRED 句とともに指定するか、この句を含めずに指定してください。

> **参照：** データベース・セキュリティの概要および中間層システムおよびプロキシ認証の詳細は、『Oracle セキュリティ概要』を参照してください。

例

ユーザー識別の変更例： 次の文は、ユーザー sidney（17-11 ページの「データベース・ユーザーの作成例：」で作成）のパスワードを second_2nd_pwd に、デフォルト表領域を表領域 example に変更します。

```
ALTER USER sidney
    IDENTIFIED BY second_2nd_pwd
    DEFAULT TABLESPACE example;
```

次の文は、new_profile プロファイル（15-51 ページの「プロファイルの作成例：」で作成）をサンプル・ユーザー sh に割り当てます。

```
ALTER USER sh
    PROFILE new_profile;
```

後続のセッションでは、sh は new_profile プロファイルの制限に従います。

次の文は、sh に直接付与されているすべてのロール（dw_manager ロールを除く）をデフォルト・ロールに設定します。

```
ALTER USER sh
    DEFAULT ROLE ALL EXCEPT dw_manager;
```

sh の次のセッションの開始時には、dw_manager 以外で sh に直接付与されているすべてのロールが使用可能になります。

ユーザー認証の変更例： 次の文は、ユーザー app_user1（17-11 ページの「データベース・ユーザーの作成例：」で作成）の認証メカニズムを変更します。

```
ALTER USER app_user1 IDENTIFIED GLOBALLY AS 'CN=tom,O=oracle,C=US';
```

次の文は、ユーザー sidney のパスワードを期限切れにします。

```
ALTER USER sidney PASSWORD EXPIRE;
```

PASSWORD EXPIRE を使用してデータベース・ユーザーのパスワードを期限切れにした場合、そのユーザー（または DBA）は、期限切れの後でデータベースにログインする際、パスワードを変更する必要があります。ただし、SQL*Plus などのツール製品を使用した場合、期限切れの後の最初のログイン時に、パスワードを変更できます。

表領域グループの割当て例： 次の文は、tbs_grp_01（16-83 ページの「表領域グループへの一時表領域の追加例：」で作成）を表領域グループとしてユーザー sh に割り当てます。

```
ALTER USER sh
  TEMPORARY TABLESPACE tbs_grp_01;
```

プロキシ・ユーザー例： 次の文は、ユーザー app_user1 を変更します。例では、app_user1 がプロキシ・ユーザー sh を使用して接続できます。また、プロキシ sh を使用して接続したときに、app_user1 が warehouse_user ロール（15-58 ページの「ロールの作成例：」で作成）を使用可能にできます。

```
ALTER USER app_user1
   GRANT CONNECT THROUGH sh
   WITH ROLE warehouse_user;
```

基本的な構文を示すため、サンプル・データベース Sales History のユーザー（sh）をプロキシとして使用します。通常、プロキシ・ユーザーは、アプリケーション・サーバーまたは中間層のエンティティに存在します。アプリケーション・サーバーを経由して、アプリケーション・ユーザーとデータベースの間のインタフェースを作成する場合の詳細は、『Oracle Call Interface プログラマーズ・ガイド』を参照してください。

参照 :

- app_user ユーザーの作成方法については、17-11 ページの「外部データベース・ユーザーの作成例 :」を参照してください。
- dw_user ロールの作成方法については、15-58 ページの「ロールの作成例 :」を参照してください。

次の文は、ユーザー app_user1 がプロキシ・ユーザー sh を使用して接続する権限を取り消します。

```
ALTER USER app_user1 REVOKE CONNECT THROUGH sh;
```

次の仮想例は、プロキシ認証の他の方法を示します。

```
ALTER USER sully GRANT CONNECT THROUGH OAS1
   AUTHENTICATED USING PASSWORD;
```

次の例では、ユーザー app_user1 をエンタープライズ・ユーザーがプロキシ使用できるように公開します。エンタープライズ・ユーザーは、Oracle Internet Directory 管理者が適切な権限を付与するまでは、app_user1 の代理として作業することがきません。

```
ALTER USER app_user1
   GRANT CONNECT THROUGH ENTERPRISE USERS;
```

ALTER VIEW

用途

ALTER VIEW を使用すると、無効なビューを明示的に再コンパイルしたり、ビューの制約を変更することができます。明示的に再コンパイルすると、実行前にコンパイル・エラーを検査できます。再コンパイルは、ビューの実表を変更した後で、その変更がビューまたはそのビューに依存するオブジェクトに影響していないかどうかを確認するときに便利です。

ALTER VIEW を使用して、制約のビューを定義、変更または削除することもできます。

この文を使用して既存のビュー定義を変更することはできません。また、ビューの実表に対する DDL 変更によってビューが無効になる場合、この文を使用して無効なビューをコンパイルすることはできません。このような場合は、CREATE VIEW に OR REPLACE キーワードを指定して使用し、ビューを再定義する必要があります。

ALTER VIEW 文を発行した場合、指定したビューは有効か無効かにかかわらず再コンパイルされます。また、そのビューに依存するすべてのローカル・オブジェクトが無効になります。

1 つ以上のマテリアライズド・ビューが参照しているビューを変更した場合、これらのマテリアライズド・ビューは無効になります。無効なマテリアライズド・ビューは、クエリー・リライトによって使用できません。また、リフレッシュすることもできません。

参照 :

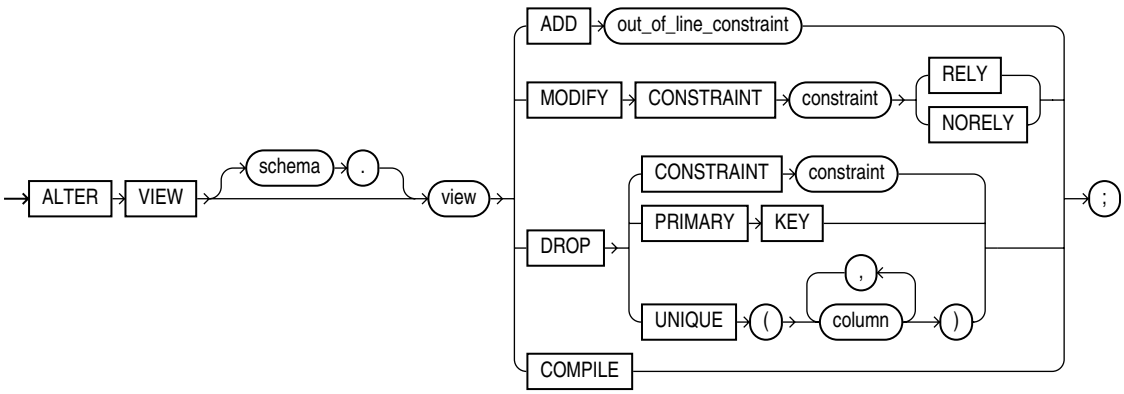
- ビューの再定義の詳細は、17-13 ページの「CREATE VIEW」を参照してください。無効なマテリアライズド・ビューの再検証の詳細は、11-2 ページの「ALTER MATERIALIZED VIEW」を参照してください。
- データ・ウェアハウスの概要は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。
- スキーマ・オブジェクト間の依存性については、『Oracle Database 概要』を参照してください。

前提条件

ビューが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY TABLE システム権限が必要です。

構文

***alter_view*::=**



(constraint::= 構文の項にある 8-5 ページの out_of_line_constraint::= を参照)

セマンティクス

schema

ビューが含まれているスキーマを指定します。*schema* を指定しない場合、Oracle Database ではビューは自分のスキーマ内にあるとみなされます。

view

再コンパイルするビューの名前を指定します。

ADD 句

ADD 句を使用すると、*view* に制約を追加できます。制約のビューおよび制限事項については、8-4 ページの「constraint」を参照してください。

MODIFY CONSTRAINT 句

MODIFY CONSTRAINT 句を使用すると、既存のビュー制約の RELY または NORELY 設定を変更できます。この設定の使用方法の詳細については、8-15 ページの「RELY 句」を、ビュー制約の概要については、8-17 ページの「ビュー制約の注意事項 :」を参照してください。

制約の変更の制限事項 : 一意制約または主キー制約が参照整合性制約の一部である場合、外部キーの削除または *view* の設定にあわせた変更をせずに、設定を変更することはできません。

DROP 句

DROP を使用すると、既存のビューの制約を削除できます。

制約の削除の制限事項 : 一意制約または主キー制約がビューの参照整合性制約の一部である場合は削除できません。

COMPILE

COMPILE キーワードを指定すると、ビューを再コンパイルできます。

例

ビューの変更例 : 次の文は、ビュー customer_ro（17-21 ページの「読取り専用ビューの作成例 :」で作成）を再コンパイルします。

```
ALTER VIEW customer_ro
    COMPILE;
```

customer_ro の再コンパイル時にエラーが発生しなければ、customer_ro は有効になります。再コンパイル・エラーが発生した場合はエラーが戻り、customer_ro は無効のままとなります。

依存するオブジェクトもすべて無効になります。依存オブジェクトとは、customer_ro を参照する、プロシージャ、ファンクション、パッケージ本体、ビューなどです。その後、明示的に再コンパイルせずに、これらのオブジェクトを参照した場合、データベースは実行時にそれらを暗黙的に再コンパイルします。

ANALYZE

用途

ANALYZE 文を使用すると、統計情報を収集して、次のような操作を実行できます。

- 索引または索引パーティション、表または表パーティション、索引構成表、クラスタまたはスカラー・オブジェクト属性の統計情報を収集または削除します。
- 索引または索引パーティション、表または表パーティション、索引構成表、クラスタまたはオブジェクト参照（REF）の構造を検証します。
- 表またはクラスタの移行行と連鎖行を識別します。

注意： ほとんどの統計の収集には、DBMS_STATS パッケージを使用してください。このパッケージを使用すると、パラレルでの統計収集、パーティション化オブジェクトに対するグローバル統計収集、および他の方法での統計収集の詳細なチューニングを行うことができます。DBMS_STATS パッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

次のようなコスト・ベースのオプティマイザとは関係のない統計情報の収集には、（DBMS_STATS ではなく）ANALYZE 文を使用します。

- VALIDATE 句、LIST CHAINED ROWS 句の使用
- 空きリストのブロックに関する情報の収集

前提条件

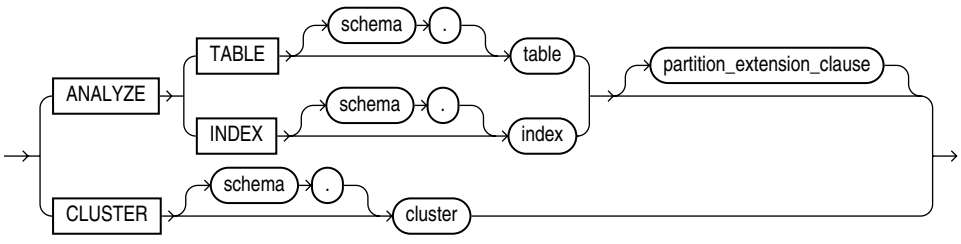
分析するスキーマ・オブジェクトがローカルである必要があります。自分のスキーマ内にない場合は、ANALYZE ANY システム権限が必要です。

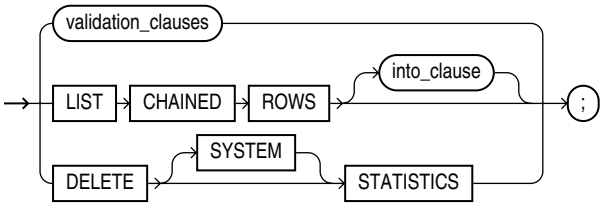
表またはクラスタの連鎖行をリスト表へ入れる場合、このリスト表が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、そのリスト表の INSERT 権限または INSERT ANY TABLE システム権限が必要です。

パーティション表の妥当性チェックを行う場合は、分析した ROWID を入れる表に対する INSERT オブジェクト権限または INSERT ANY TABLE システム権限が必要です。

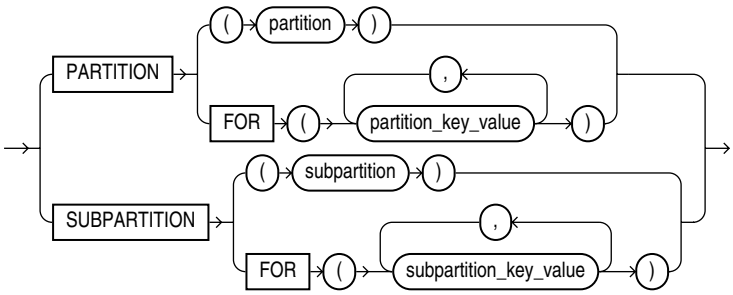
構文

***analyze*::=**

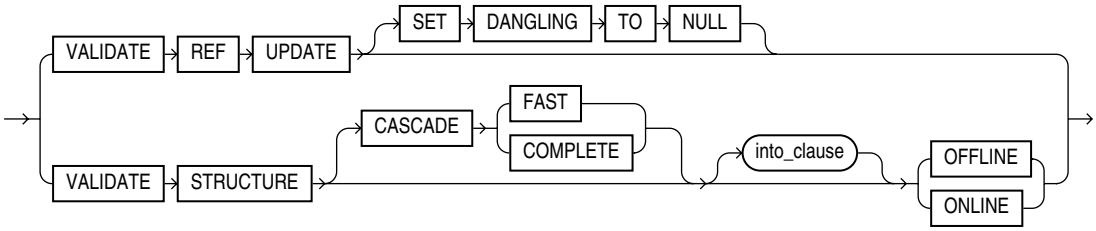




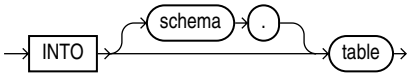
***partition_extension_clause*::=**



validation_clauses::=



***into_clause*::=**



セマンティクス

schema

表、索引またはクラスタが含まれているスキーマを指定します。*schema* を指定しない場合、表、索引またはクラスタは自分のスキーマ内にあるとみなされます。

TABLE *table*

分析する表を指定します。表を分析すると、すべてのファンクション索引に発生する式について統計情報が収集されます。したがって、表を分析する前に、必ずファンクション索引を作成してください。ファンクション索引の詳細は、14-50 ページの「CREATE INDEX」を参照してください。

表を分析すると、LOADING または FAILED のマークが付いたドメイン索引はすべてスキップされます。

索引構成表の場合、マッピング表が分析され、その PCT_ACCESSS_DIRECT 統計情報も計算されます。これらの統計情報は、マッピング表のローカル ROWID の一部として格納されたと推測されるデータ・ブロック・アドレスの精度を評価します。

表については、次の統計情報が収集されます。アスタリスクが付いた統計情報は、常に厳密に計算されます。表の統計情報（ドメイン索引の状態を含む）は、データ・ディクショナリ・ビュー USER_TABLES、ALL_TABLES および DBA_TABLES のカッコで示す列に表示されます。

- 行数（NUM_ROWS）
- 最高水位標を下回るデータ・ブロックの数（現在データを含むか含まないかにかかわらず、データを格納するようにフォーマットされているデータ・ブロックの数）（BLOCKS）*
- 未使用の表に対して割り当てられているデータ・ブロックの数（EMPTY_BLOCKS）*

- 各データ・ブロックにおける使用可能な空き領域サイズの平均値（バイト単位）（AVG_SPACE）
- 連鎖行の数（CHAIN_COUNT）
- 行のオーバーヘッドを含む、バイト単位での行の平均の長さ（AVG_ROW_LEN）

表の分析の制限事項： 表の分析には、次の制限事項があります。

- ANALYZE を使用して、データ・ディクショナリ表の統計情報を収集しないでください。
- ANALYZE を使用して、外部表の統計情報を収集しないでください。かわりに、DBMS_STATS パッケージを使用する必要があります。
- ANALYZE を使用して、一時表のデフォルト統計情報を収集しないでください。ただし、すでに一時表の 1 つ以上の列とユーザー定義統計タイプを対応付けている場合、ANALYZE を使用して一時表のユーザー定義統計情報を収集できます
- REF 列型、VARRAY、ネストした表、LOB 列型（LOB 列型は分析されずにスキップされる）、LONG 列型、オブジェクト型などの列型の統計情報は計算または推定できません。ただし、このような列に統計タイプが対応付けられている場合は、ユーザー定義統計情報が収集されます。

参照：

- 「ASSOCIATE STATISTICS」（13-21 ページ）
- データ・ディクショナリ・ビューの詳細は、『Oracle Database リファレンス』を参照してください。

partition_extension_clause

統計情報を収集するパーティションまたはサブパーティション、あるいはパーティション値またはサブパーティション値を指定します。クラスタの分析時にこの句は使用できません。

table がコンポジット・パーティションのときに PARTITION を指定した場合、指定したパーティション内ですべてのサブパーティションが分析されます。

INDEX *index*

分析する索引を指定します。

索引については、次の統計情報が収集されます。アスタリスクが付いた統計情報は、常に厳密に計算されます。従来索引について統計情報を計算または推定する場合、統計情報は、データ・ディクショナリ・ビュー USER_INDEXES、ALL_INDEXES および DBA_INDEXES のカッコで示す列に表示されます。

- ルート・ブロックからリーフ・ブロックまでの索引の深さ（BLEVEL）*
- リーフ・ブロックの数（LEAF_BLOCKS）
- 個別索引値の数（DISTINCT_KEYS）
- 索引の値ごとのリーフ・ブロックの平均数（AVG_LEAF_BLOCKS_PER_KEY）
- （表に対する索引の）索引の値ごとのデータ・ブロックの平均数（AVG_DATA_BLOCKS_PER_KEY）
- クラスタ係数（索引付きの値についての行が、どれだけ効率的に順序付けられているか）（CLUSTERING_FACTOR）

ドメイン索引の場合、索引に関連付けられた統計タイプで指定したユーザー定義統計収集ファンクションが、この文によって起動されます（13-21 ページの「ASSOCIATE STATISTICS」を参照）。ドメイン索引に関連付けられた統計タイプがない場合、その索引タイプに関連付けられた統計タイプが使用されます。索引またはその索引タイプの統計タイプがない場合、ユーザー定義統計情報は収集されません。ユーザー定義索引統計情報は、データ・ディクショナリ・ビュー USER_USTATS、ALL_USTATS および DBA_USTATS の STATISTICS 列に表示されます。

注意： 多数の行が削除されている索引を分析する場合、統計情報操作として ESTIMATE を要求しても、COMPUTE 統計情報操作が実行され、全表スキャンが行われることがあります。このような操作は、非常に時間がかかる場合があります。

索引の分析の制限事項： IN_PROGRESS または FAILED のマークが付いたドメイン索引は分析できません。

参照：

- ドメイン索引の詳細は、14-50 ページの「CREATE INDEX」を参照してください。
- データ・ディクショナリ・ビューの詳細は、『Oracle Database リファレンス』を参照してください。
- 「索引の分析例 :」（13-19 ページ）

CLUSTER *cluster*

分析するクラスタを指定します。クラスタの統計情報を収集すると、すべてのクラスタ表、およびクラスタ索引を含むすべての索引の統計情報も自動的に収集されます。

索引クラスタとハッシュ・クラスタには、単一クラスタ・キー（AVG_BLOCKS_PER_KEY）が使用するデータ・ブロックの平均数が収集されます。これらの統計情報は、データ・ディクショナリ・ビュー ALL_CLUSTERS、USER_CLUSTERS および DBA_CLUSTERS に表示されます。

参照： データ・ディクショナリ・ビューの詳細は、『Oracle Database リファレンス』および 13-20 ページの「クラスタの分析例 :」を参照してください。

VALIDATE REF UPDATE 句

VALIDATE REF UPDATE を指定すると、指定された表の REF 値の妥当性チェックを行い、各 REF 内の ROWID 部分をチェックして、それを真の ROWID と比較できます。誤りがある場合はそれを修正します。この句は、表を分析する場合にのみ使用できます。

表の所有者が参照先オブジェクトに対する SELECT オブジェクト権限を持っていない場合、このオブジェクトは無効とみなされ、NULL に設定されます。その結果、オブジェクトに対して適切な権限を持つユーザーによって発行された問合せであっても、問合せでこれらの REF 値を使用することはできません。

SET DANGLING TO NULL SET DANGLING TO NULL を指定すると、指定した表内の REF 値が（範囲が限定されるかどうかにかかわらず）無効なオブジェクトまたは存在しないオブジェクトを指している場合に、REF 値が NULL に設定されます。

VALIDATE STRUCTURE

VALIDATE STRUCTURE を指定すると、分析対象オブジェクトの構造を検証できます。この句で収集された統計情報は Oracle Database オプティマイザでは使用されません。

参照： 「表の検証例 :」（13-20 ページ）

- 表に対して、それぞれのデータ・ブロックと行の整合性が検証されます。索引構成表の場合、表の主キー索引に対する圧縮統計情報（最適なプレフィックス圧縮件数）も生成されます。
- クラスタに対して、自動的にクラスタ表の構造が検証されます。

- パーティションに対して、行が適切なパーティションに属するかどうかが検証されます。行が正しく照合されなかった場合は、ROWID が INVALID_ROWS 表に挿入されます。
- 一時表に対して、現行のセッション中に表の構造および索引が検証されます。
- 索引に対して、索引のそれぞれのデータ・ブロックの整合性が検証され、ブロックの破損がチェックされます。この句は、表のそれぞれの行が索引エントリを持っていること、またはそれぞれの索引エントリが表の行を指していることを確認するわけではありません。これらを確認する場合は、CASCADE 句を使用して表の構造を検証します。

  通常のすべての索引用の圧縮統計情報（最適なプレフィックス圧縮件数）も計算されます。

  索引の統計情報は、データ・ディクショナリ・ビュー INDEX_STATS および INDEX_HISTOGRAM に格納されます。

  > **参照：** これらのビューの詳細は、『Oracle Database リファレンス』を参照してください。

オブジェクトの構造に障害がある場合は、エラー・メッセージが戻ります。この場合、オブジェクトを削除して作成しなおす必要があります。

INTO VALIDATE STRUCTURE の INTO 句は、パーティション表のみに有効です。正しく照合されなかった行を持つパーティションの ROWID を格納するリスト表を指定します。*schema* を指定しない場合、リストは自分のスキーマ内にあるとみなされます。この句自体を指定しない場合、表の名前は INVALID_ROWS になります。この表を作成するために使用する SQL スクリプトは UTLVALID.SQL です。

CASCADE CASCADE を指定すると、表またはクラスタに関連付けられた索引の構造を検証できます。表を検証するときにこの句を指定すると、その表に定義された索引も検証されます。クラスタを検証するときにこの句を指定した場合、クラスタ表のすべての索引（クラスタ索引を含む）が検証されます。

この句を使用して使用可能な（以前は使用禁止であった）ファンクション索引を検証すると、検証エラーになる場合があります。この場合は、索引を再構築する必要があります。

ONLINE | OFFLINE ONLINE を指定すると、Oracle Database がオブジェクトの DML 操作中に検証を実行できるようになります。データベースは、並行して操作が行える程度に、実行する検証の量を減らします。

> **注意：** OFFLINE 指定でオブジェクトの構造を検証する場合と同様に、ONLINE 指定でオブジェクトの構造を検証する場合、Oracle Database は統計情報を収集しません。

OFFLINE を指定すると、実行する検証の量が最大になります。この設定は、検証中に INSERT、UPDATE および DELETE 文がオブジェクトに平行してアクセスすることを防ぎますが、問合せは許可されます。これはデフォルトです。

ONLINE の制限事項： ONLINE は、クラスタを分析する場合には指定できません。

LIST CHAINED ROWS

LIST CHAINED ROWS を指定すると、分析した表またはクラスタの移行行および連鎖行を識別できます。索引の分析時にこのオプションは使用できません。

INTO 句には、移行行および連鎖行をリストする表を指定します。*schema* を指定しない場合、連鎖行表は自分のスキーマ内にあるとみなされます。この句自体を指定しない場合、表の名前は CHAINED_ROWS になります。連鎖行表は、ローカル・データベース内にある必要があります。

次のいずれかのスクリプトを使用して、CHAINED_ROWS 表を作成できます。

- UTLCHAIN.SQL: 物理 ROWID を使用します。そのため、行は、索引構成表からではなく従来表から収集されます（次の注意を参照）。
- UTLCHN1.SQL: ユニバーサル ROWID を使用します。そのため、行は、従来表および索引構成表の両方から収集されます。

独自の連鎖行表を作成する場合、この 2 つのスクリプトのいずれかで規定されるフォーマットに従う必要があります。

ユニバーサル ROWID ではなく、主キーに基づく索引構成表を分析する場合、索引構成表ごとに別の連鎖行表を作成し、主キー記憶域を確保する必要があります。まず、SQL スクリプトの DBMSIOTC.SQL および PRVTIOTC.PLB を使用して、BUILD_CHAIN_ROWS_TABLE プロシージャを定義します。次に、このプロシージャを実行して、索引構成表の IOT_CHAINED_ROWS 表を作成します。

参照 :

- これらのスクリプトおよび連鎖行の排除の詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。
- パッケージ化された SQL スクリプトの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の DBMS_IOT パッケージを参照してください。
- 「連鎖行のリスト例 :」（13-20 ページ）

DELETE STATISTICS

DELETE STATISTICS を指定すると、現在データ・ディクショナリに格納されている、分析したオブジェクトの統計情報を削除できます。Oracle Database に統計情報を使用させないようにする場合、この文を使用します。

表を指定してこの句を使用した場合、指定した表に定義されたすべての索引の統計情報も自動的に削除されます。クラスタを指定してこの句を使用した場合、指定したクラスタのすべての表、およびこれらの表のすべての索引（クラスタ索引を含む）の統計情報も自動的に削除されます。

ユーザー定義統計情報ではなく、システム統計情報のみを削除する場合は、SYSTEM を指定します。SYSTEM を指定しないと、オブジェクトのユーザー定義列または索引統計情報が収集された場合、データベースは、統計情報を収集するために使用された情報タイプに指定されている統計削除ファンクションを起動して、ユーザー定義統計情報も削除します。

参照 : 「統計情報の削除例 :」（13-19 ページ）

validation_clauses

validation_clauses を使用すると、REF 値および分析したオブジェクトの構造を検証できます。

参照 : 表、索引、クラスタおよびマテリアライズド・ビューの検証の詳細は、『Oracle Database 管理者ガイド』を参照してください。

例

統計情報の削除例 : 次の文は、サンプル表 oe.orders とデータ・ディクショナリにあるそのすべての索引の統計情報を削除します。

```
ANALYZE TABLE orders DELETE STATISTICS;
```

索引の分析例 : 次の文は、サンプル索引 oe.inv_product_ix の構造を検証します。

```
ANALYZE INDEX inv_product_ix VALIDATE STRUCTURE;
```

表の検証例： 次の文は、サンプル表 hr.employees およびそのすべての索引を分析します。

```
ANALYZE TABLE employees VALIDATE STRUCTURE CASCADE;
```

表に対する VALIDATE REF UPDATE 句は、指定した表の REF 値を検証します。また、それぞれの REF の ROWID 部分をチェックし、それを真の ROWID と比較します。その結果、ROWID が誤っていると判断されると、ROWID 部分が正しくなるように REF が更新されます。

次の文は、サンプル表 oe.customers の REF 値を検証します。

```
ANALYZE TABLE customers VALIDATE REF UPDATE;
```

次の文は、DML 操作を同時に実行することを許可して、サンプル表 oe.customers の構造を検証します。

```
ANALYZE TABLE customers VALIDATE STRUCTURE ONLINE;
```

クラスタの分析例： 次の文は、personnel クラスタ（14-7 ページの「クラスタの作成例：」で作成）、そのすべての表、クラスタ索引を含むすべての索引を分析します。

```
ANALYZE CLUSTER personnel
    VALIDATE STRUCTURE CASCADE;
```

連鎖行のリスト例： 次の文は、orders 表のすべての連鎖行についての情報を収集します。

```
ANALYZE TABLE orders
   LIST CHAINED ROWS INTO chained_rows;
```

前述の文では、情報は表 chained_rows に格納されます。次の問合せでその行を検証できます（表に連鎖行が含まれない場合は、行は戻されません）。

```
SELECT owner_name, table_name, head_rowid, analyze_timestamp
    FROM chained_rows
    ORDER BY owner_name, table_name, head_rowid, analyze_timestamp;

OWNER_NAME  TABLE_NAME  HEAD_ROWID         ANALYZE_TIMESTAMP
----------  ----------  ------------------ -----------------
OE          ORDERS      AAAAZzAABAAABrXAAA 25-SEP-2000
```

ASSOCIATE STATISTICS

用途

ASSOCIATE STATISTICS 文を使用すると、統計収集、選択性またはコストに関するファンクションが含まれた統計タイプ（またはデフォルトの統計）を、1 つ以上の列、スタンドアロン・ファンクション、パッケージ、型、ドメイン索引または索引タイプに関連付けることができます。

現在のすべての統計タイプの関連付けのリストについては、USER_ASSOCIATIONS データ・ディクショナリ・ビューを問い合せます。統計情報に関連付けるオブジェクトを分析する場合、USER_USTATS ビューでその関連性を問い合せることができます。

> **参照：** ANALYZE が関連性で使用する優先順位については、13-14 ページの「ANALYZE」を参照してください。

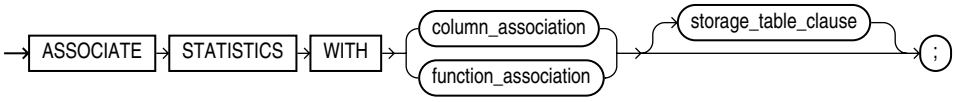
前提条件

この文を発行する場合は、ベース・オブジェクト（表、ファンクション、パッケージ、型、ドメイン索引または索引タイプ）を変更する適切な権限が必要です。さらに、デフォルト統計情報のみを関連付けていないかぎり、統計タイプに対する実行権限が必要です。統計タイプは、すでに定義されている必要があります。

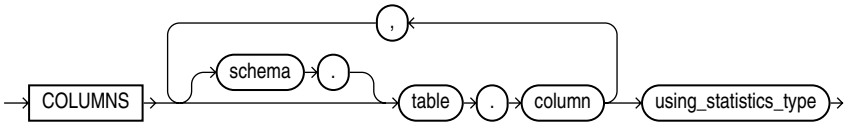
> **参照：** 型の定義の詳細は、17-3 ページの「CREATE TYPE」を参照してください。

構文

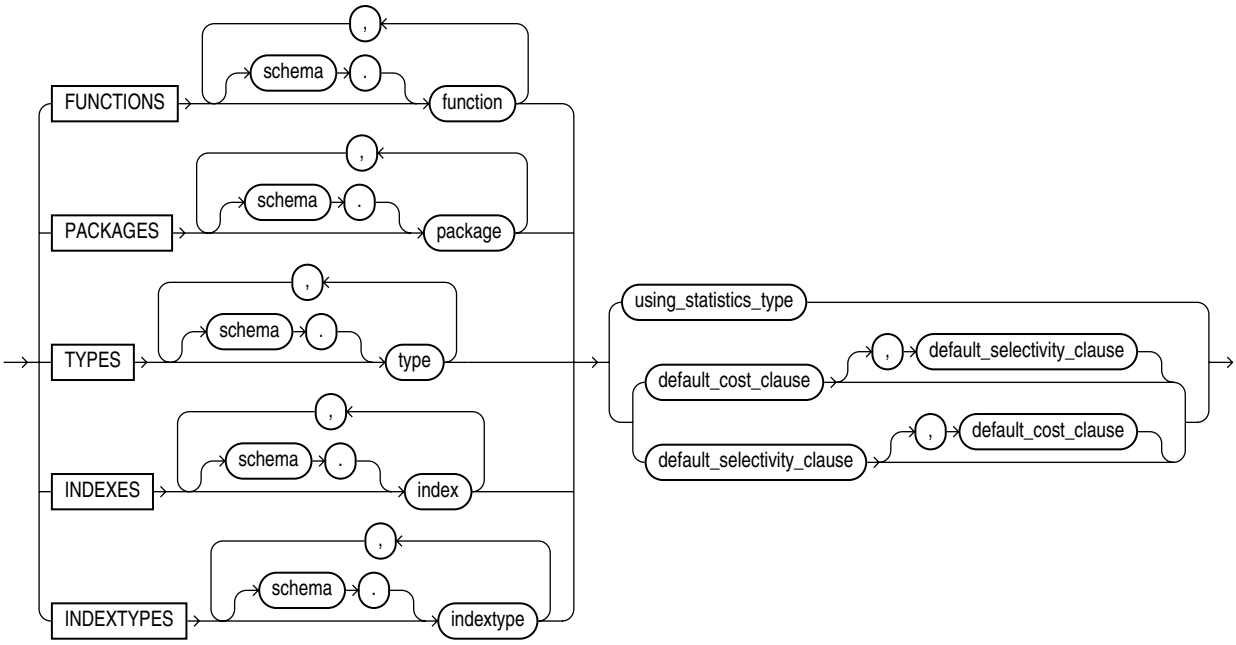
***associate_statistics*::=**



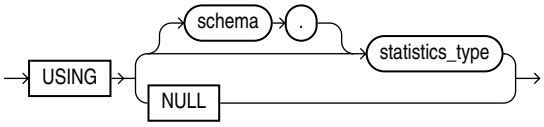
column_association::=



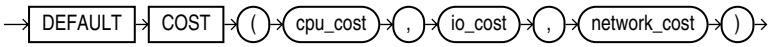
function_association::=



using_statistics_type::=



default_cost_clause::=



default_selectivity_clause::=

storage_table_clause::=

セマンティクス

column_association

1 つ以上の表の列を指定します。*schema* を指定しない場合、表は自分のスキーマ内にあるとみなされます。

function_association

1 つ以上のスタンドアロン・ファンクション、パッケージ、ユーザー定義データ型、ドメイン索引または索引タイプを指定します。*schema* を指定しない場合、オブジェクトは自分のスキーマ内にあるとみなされます。

- FUNCTIONS は、スタンドアロン・ファンクションのみを参照し、メソッド型または組込みファンクションは参照しません。
- TYPES はユーザー定義型のみを参照し、組込み SQL データ型は参照しません。

***function_association* の制限事項：** すでに関連性を定義してあるオブジェクトには指定できません。まず、このオブジェクトと統計情報の関連性を取り消す必要があります。

> **参照：** 「DISASSOCIATE STATISTICS」（17-30 ページ）および「統計情報の関連付け例 :」（13-24 ページ）

using_statistics_type

列、ファンクション、パッケージ、型、ドメイン索引または索引タイプと関連付ける統計タイプ（または型のシノニム）を指定します。*statistics_type* は作成済である必要があります。

NULL キーワードは、統計情報を列または索引に関連付ける場合のみに有効です。統計タイプをオブジェクト型に関連付ける場合は、そのオブジェクト型の列は、統計タイプを継承します。同様に、統計タイプを索引タイプに関連付ける場合は、索引タイプの索引インスタンスは統計タイプを継承します。列または索引に別の統計タイプを関連付けると、この継承を上書きできます。または、列または索引に統計タイプを関連付けない場合は、*using_statistics_type* 句で NULL を指定します。

統計タイプの指定の制限事項： ファンクション、パッケージ、型または索引タイプには NULL を指定できません。

> **参照：** 統計収集ファンクションの作成の詳細は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。

default_cost_clause

スタンドアロン・ファンクション、パッケージ、型、ドメイン索引または索引タイプのデフォルトのコストを指定します。この句を指定する場合、CPU コスト、I/O コスト、ネットワーク・コストの順でそれぞれに対して 1 つの数を指定する必要があります。それぞれのコストは、ファンクションまたはメソッドを 1 度実行した場合、またはドメイン索引へ 1 度アクセスした場合の値です。指定できる値は 0（ゼロ）以上の整数です。

default_selectivity_clause

スタンドアロン・ファンクション、型、パッケージまたはユーザー定義演算子が指定された述語に対するデフォルトの選択性をパーセントで指定します。*default_selectivity_clause* は、0 ～ 100 の数値である必要があります。範囲外のものは無視されます。

***default_selectivity_clause* 句の制限事項：** DEFAULT SELECTIVITY は、ドメイン索引または索引タイプに指定できません。

> **参照：** 「デフォルト・コストの指定例 :」（13-24 ページ）

storage_table_clause

この句は、INDEXTYPE の統計に対してのみ有効です。

- WITH SYSTEM MANAGED STORAGE TABLES を指定すると、統計データの格納がシステムで管理されます。*statistics_type* に指定するタイプによって、システムで保持される表に統計関連の情報が格納されます。また、指定する索引タイプはすでに登録済であるか、または WITH SYSTEM MANAGED STORAGE TABLES 句をサポートするように変更されている必要があります。
- WITH USER MANAGED STORAGE TABLES を指定すると、ユーザー定義の統計情報を格納する表は、ユーザーによって管理されます。これはデフォルトの動作です。

例

統計情報の関連付け例： 次の文は、スタンドアロン・パッケージ emp_mgmt への関連性を作成します。このパッケージを作成する例については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

```
ASSOCIATE STATISTICS WITH PACKAGES emp_mgmt DEFAULT SELECTIVITY 10;
```

デフォルト・コストの指定例： 次の文は、ドメイン索引 salary_index（E-2 ページの「拡張索引作成機能の使用方法」で作成）を使用して任意の述語を実装した場合、常に CPU コストは 100、I/O コストは 5、およびネットワーク・コストは 0 になるように指定します。

```
ASSOCIATE STATISTICS WITH INDEXES salary_index DEFAULT COST (100,5,0);
```

オプティマイザは、コスト・ファンクションをコールするかわりに、これらのデフォルト・コストを使用します。

AUDIT

用途

AUDIT 文を使用すると、次の操作を実行できます。

- 後続のユーザー・セッションでの SQL 文の発行の監査。特定の SQL 文、または特定のシステム権限によって許可されたすべての SQL 文の発行を監査できます。SQL 文操作の監査は、後続セッションにのみ適用され、現行のセッションには適用されません。
- 特定のスキーマ・オブジェクトに対する操作の監査。スキーマ・オブジェクト操作の監査は、後続のセッションと同様に、現行のセッションにも適用されます。

参照 :

- 値に基づいた監査方針を作成および管理する DBMS_FGA パッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。
- 監査を無効にする方法の詳細は、18-76 ページの「NOAUDIT」を参照してください。

前提条件

SQL 文の発行を監査するには、AUDIT SYSTEM システム権限が必要です。

監査結果を収集するには、初期化パラメータ AUDIT_TRAIL をデフォルトの設定である NONE 以外の値に設定して、監査を有効にする必要があります。監査オプションは、監査が使用可能であるかどうかにかかわらず指定できます。ただし、監査を有効にしないと、監査レコードは作成されません。

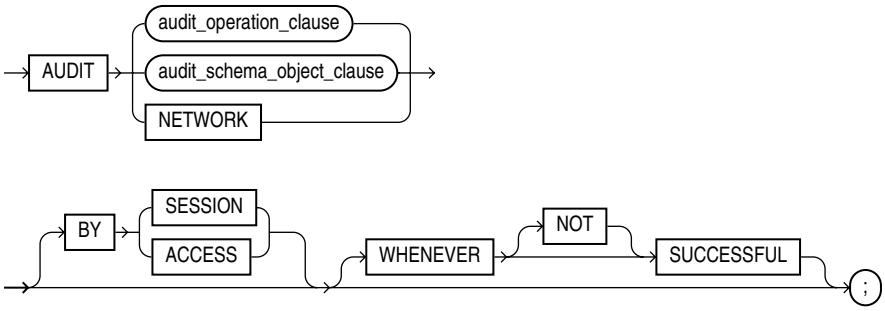
スキーマ・オブジェクト操作を監査するためには、監査対象のオブジェクトが自分のスキーマにある必要があります。自分のスキーマにない場合は、AUDIT ANY システム権限が必要です。また、監査の対象とするオブジェクトがディレクトリ・オブジェクトの場合は、それが自分で作成したものであっても、AUDIT ANY システム権限が必要です。

注意 : AUDIT ANY システム権限によって、権限受領者は、SYS スキーマを除く任意のスキーマ内の任意のオブジェクトを監査できます。O7_DICTIONARY_ACCESSIBILITY 初期化パラメータを TRUE に設定することによって、そのような権限受領者が SYS スキーマ内のオブジェクトを監査できるようにすることができます。セキュリティ上の理由のため、この設定の使用には注意が必要です。

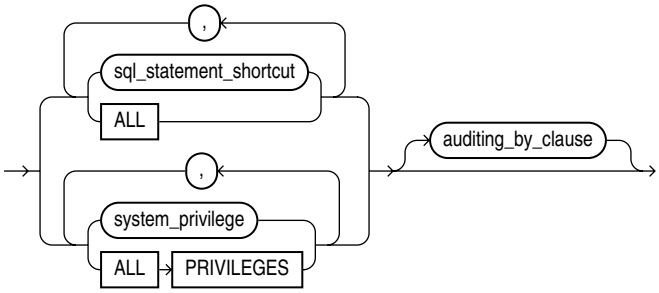
参照 : AUDIT_TRAIL パラメータの詳細は、『Oracle Database リファレンス』を参照してください。

構文

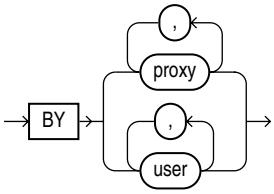
***audit*::=**



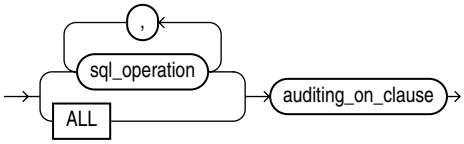
audit_operation_clause::=



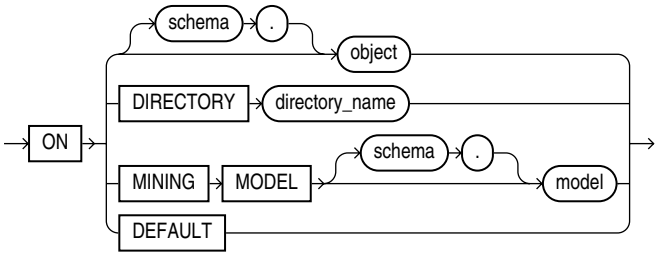
auditing_by_clause::=



audit_schema_object_clause::=



auditing_on_clause::=



セマンティクス

audit_operation_clause

audit_operation_clause を使用すると、操作の影響を受けるスキーマ・オブジェクトに関係なく、指定された操作を監査できます。

sql_statement_shortcut

ショートカットを指定すると、特定の SQL 文の使用を監査できます。13-30 ページの表 13-1 および 13-32 ページの表 13-2 に、ショートカットおよびそれらが監査する SQL 文を示します。

注意： SQL 文ショートカットとシステム権限を混同しないでください。次に例を示します。

- AUDIT USER 文は、SQL の CREATE USER、ALTER USER および DROP USER 文のすべてを監査する USER ショートカットを指定します。この場合の監査には、ユーザーが ALTER USER 文で自分のパスワードを変更する操作が含まれています。
- AUDIT ALTER USER 文は、そのシステム権限を使用するすべての操作を監査する ALTER USER システム権限を指定します。この場合の監査には、ユーザーが自分のパスワードを変更する操作は含まれません。これは、その操作には、ALTER USER システム権限は必要ないためです。

監査されるたびに、次の情報を持つ監査レコードが生成されます。

- 操作を行ったユーザー
- 操作の種類
- 操作に関連するオブジェクト
- 操作の日付と時刻

監査レコードは、監査証跡に書き込まれます。監査証跡とは、監査レコードが入っているデータベースの表です。データ・ディクショナリ・ビューを問い合せて監査証跡を調べることによって、データベース・アクティビティを再検討できます。

参照：

- 監査証跡データ・ディクショナリ・ビューのリストの詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。
- データ・ディクショナリ・ビューの詳細は、『Oracle Database リファレンス』を参照してください。
- 「ロールに関連する SQL 文の監査例 :」（13-34 ページ）

system_privilege

システム権限を指定すると、特定のシステム権限を与えられた SQL 文およびその他の操作を監査できます。

注意： ANY キーワードを含むシステム権限の使用の監査は、ANY キーワードを含まない同じ権限の使用の監査より限定的です。次に例を示します。

- AUDIT CREATE PROCEDURE は、CREATE PROCEDURE または CREATE ANY PROCEDURE 権限を使用して発行された文を監査します。
- AUDIT CREATE ANY PROCEDURE は、CREATE ANY PROCEDURE 権限を使用して発行された文のみを監査します。

多くの個々のシステム権限を指定するのではなく、ロール CONNECT、RESOURCE および DBA を指定できます。これは、そのロールに付与されているすべてのシステム権限を監査することと同じです。

Oracle Database には、システム権限と文オプションをまとめて指定するための、次の 2 つのショートカットが用意されています。

ALL ALL を指定すると、表 13-1 のすべての文オプションを監査できます。ただし、表 13-2 の追加文オプションは監査しません。

ALL PRIVILEGES ALL PRIVILEGES を指定すると、システム権限を監査できます。

注意： ロールまたはショートカットでなく、監査に個々のシステム権限および文オプションを指定することをお薦めします。ロールおよびショートカットに含まれるシステム権限および文オプションは、リリースごとに異なり、Oracle Database の今後のバージョンでサポートされない場合があります。

参照：

- すべてのシステム権限とそれによって許可される SQL 文のリストは、18-37 ページの表 18-1「システム権限」を参照してください。
- CONNECT、RESOURCE および DBA ロールの詳細は、18-31 ページの「GRANT」を参照してください。
- 13-35 ページの「問合せおよび更新を行う SQL 文の監査例 :」、13-35 ページの「削除の監査例 :」および 13-35 ページの「ディレクトリに関連する文の監査例 :」を参照してください。

auditing_by_clause

auditing_by_clause を指定すると、特定のユーザーが発行する SQL 文のみを監査できます。この句を指定しない場合、すべてのユーザー文が監査されます。

BY user この句を使用すると、特定のユーザーによって発行された SQL 文のみを監査するように制限できます。

BY proxy この句を使用すると、特定のプロキシによって発行された SQL 文のみを監査するように制限できます。

参照： プロキシおよびデータベースの使用については、『Oracle Database セキュリティ・ガイド』を参照してください。

audit_schema_object_clause

audit_schema_object_clause を使用すると、特定のスキーマ・オブジェクトの操作を監査できます。

sql_operation

監査する SQL 操作を指定します。13-33 ページの表 13-3 に、監査できるオブジェクトのタイプ、およびオブジェクトごとに監査できる SQL 文を示します。たとえば、ALTER 操作を指定して表の監査を選択した場合、その表に対して発行される ALTER TABLE 文がすべて監査されます。また、SELECT 操作を指定して順序の監査を選択した場合、その順序の値を使用するすべての文が監査されます。

ALL

ALL をショートカットに指定することは、オブジェクト・タイプに適用できる SQL 操作をすべて指定することと同じです。

auditing_on_clause

auditing_on_clause を使用すると、監査する特定のスキーマ・オブジェクトを指定できます。

> **参照：** 13-35 ページの「表の問合せの監査例 :」、13-35 ページの「表の挿入および更新の監査例 :」および 13-35 ページの「順序の操作の監査例 :」を参照してください。

schema 監査の対象として選択されたオブジェクトが定義されているスキーマを指定します。*schema* を指定しない場合、オブジェクトは自分のスキーマ内に定義されているものとみなされます。

object 監査するオブジェクトの名前を指定します。オブジェクトは、表、ビュー、順序、ストアド・プロシージャ、ストアド・ファンクション、ストアド・パッケージ、マテリアライズド・ビュー、マイニング・モデルまたはライブラリのいずれかである必要があります。

表、ビュー、順序、プロシージャ、ストアド・ファンクション、パッケージ、マテリアライズド・ビューまたはユーザー定義型については、それぞれシノニムも指定できます。

ON DEFAULT ON DEFAULT を指定すると、それ以降に作成したオブジェクトのデフォルト・オブジェクト・オプションとして使用する、指定のオブジェクト・オプションを作成できます。デフォルトの監査オプションが確立されると、その後作成されるオブジェクトに対して、これらのオプションが自動的に適用され、監査が行われます。ビューに対するデフォルト監査オプションは、常に、そのビューの実表に対する監査オプションの論理和となります。ALL_DEF_AUDIT_OPTS データ・ディクショナリ・ビューを問い合せることによって、現在のデフォルト監査オプションを表示できます。

デフォルト監査オプションを変更した場合でも、以前作成したオブジェクトの監査オプションはそのまま残ります。AUDIT 文の ON 句にオブジェクトを指定した場合のみ、既存のオブジェクトの監査オプションを変更できます。

> **参照：** 「デフォルト監査オプションの設定例 :」（13-36 ページ）

ON DIRECTORY ON DIRECTORY 句を使用すると、監査するディレクトリ名を指定できます。

ON MINING MODEL ON MINING MODEL 句を使用すると、監査するマイニング・モデル名を指定できます。

NETWORK

この句を使用すると、ネットワーク・レイヤーの内部的な障害を検出できます。

> **参照：** ネットワーク監査の詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。

BY SESSION

以前のリリースでは、BY SESSION を指定すると、データベースによって、同一セッションの同一スキーマ・オブジェクトで実行された同じ種類のすべての SQL 文または操作に対して 1 つのレコードが書き込まれました。このリリースの Oracle Database からは、BY SESSION および BY ACCESS を指定すると、監査された文および操作ごとに 1 つの監査レコードが書き込まれます。BY ACCESS と比較すると、BY SESSION では引き続き異なる値が監査証跡に移入されます。いずれの句も指定しない場合は、BY SESSION がデフォルトとなります。

注意： この変更は、データ定義言語（DDL）文ではなく SQL 文を監査する文オプションおよびシステム権限にのみ適用されます。DDL 文を監査するすべての SQL 文およびシステム権限は、BY ACCESS で常に監査されています。

BY ACCESS

BY ACCESS を指定すると、監査された各文および操作について、1 つのレコードを書き込むことができます。

> **注意：** データ定義言語（DDL）文を監査する SQL 文ショートカットまたはシステム権限を指定した場合は、常にアクセスごとに監査されます。それ以外のすべての場合、データベースは BY SESSION または BY ACCESS 指定に従います。

DDL 文以外の SQL 文を監査する文オプションとシステム権限には、BY SESSION と BY ACCESS のどちらでも指定できます。デフォルトは BY SESSION です。

WHENEVER [NOT] SUCCESSFUL

WHENEVER SUCCESSFUL を指定すると、正常に実行された SQL 文および操作のみを監査できます。

失敗またはエラーが発生した SQL 文および操作のみを監査する場合は、WHENEVER NOT SUCCESSFUL を指定します。

この句を指定しない場合、処理結果にかかわらず監査が行われます。

監査オプションの表

表 13-1 監査の SQL 文ショートカット

SQL 文ショートカット	監査済の SQL 文と操作
ALTER SYSTEM	ALTER SYSTEM
CLUSTER	CREATE CLUSTER
	ALTER CLUSTER
	DROP CLUSTER
	TRUNCATE CLUSTER
CONTEXT	CREATE CONTEXT
	DROP CONTEXT
DATABASE LINK	CREATE DATABASE LINK
	DROP DATABASE LINK
DIMENSION	CREATE DIMENSION
	ALTER DIMENSION
	DROP DIMENSION
DIRECTORY	CREATE DIRECTORY
	DROP DIRECTORY
INDEX	CREATE INDEX
	ALTER INDEX
	ANALYZE INDEX
	DROP INDEX
MATERIALIZED VIEW	CREATE MATERIALIZED VIEW
	ALTER MATERIALIZED VIEW
	DROP MATERIALIZED VIEW
NOT EXISTS	指定したオブジェクトが存在しない場合に失敗するすべての SQL 文

表 13-1 監査の SQL 文ショートカット（続き）

SQL 文ショートカット	監査済の SQL 文と操作
OUTLINE	CREATE OUTLINE
	ALTER OUTLINE
	DROP OUTLINE
PROCEDURE（表の後の「注意」を参照）	CREATE FUNCTION
	CREATE LIBRARY
	CREATE PACKAGE
	CREATE PACKAGE BODY
	CREATE PROCEDURE
	DROP FUNCTION
	DROP LIBRARY
	DROP PACKAGE
	DROP PROCEDURE
PROFILE	CREATE PROFILE
	ALTER PROFILE
	DROP PROFILE
PUBLIC DATABASE LINK	CREATE PUBLIC DATABASE LINK
	DROP PUBLIC DATABASE LINK
PUBLIC SYNONYM	CREATE PUBLIC SYNONYM
	DROP PUBLIC SYNONYM
ROLE	CREATE ROLE
	ALTER ROLE
	DROP ROLE
	SET ROLE
ROLLBACK SEGMENT	CREATE ROLLBACK SEGMENT
	ALTER ROLLBACK SEGMENT
	DROP ROLLBACK SEGMENT
SEQUENCE	CREATE SEQUENCE
	DROP SEQUENCE
SESSION	Logons
SYNONYM	CREATE SYNONYM
	DROP SYNONYM
SYSTEM AUDIT	AUDIT *sql_statements*
	NOAUDIT *sql_statements*
SYSTEM GRANT	GRANT *system_privileges_and_roles*
	REVOKE *system_privileges_and_roles*
TABLE	CREATE TABLE
	DROP TABLE
	TRUNCATE TABLE
TABLESPACE	CREATE TABLESPACE
	ALTER TABLESPACE
	DROP TABLESPACE

表 13-1 監査の SQL 文ショートカット（続き）

SQL 文ショートカット	監査済の SQL 文と操作
TRIGGER	CREATE TRIGGER ALTER TRIGGER ENABLE および DISABLE 句付き DROP TRIGGER ALTER TABLE ENABLE ALL TRIGGERS 句付き DISABLE ALL TRIGGERS 句付き
TYPE	CREATE TYPE CREATE TYPE BODY ALTER TYPE DROP TYPE DROP TYPE BODY
USER	CREATE USER ALTER USER DROP USER **注意：** ■ AUDIT USER は、これら 3 つの SQL 文を監査します。AUDIT ALTER USER を使用すると、ALTER USER システム権限を必要とする文を監査できます。 ■ AUDIT ALTER USER 文は、自分のパスワードを変更するユーザーを監査しません。これは、このアクティビティには ALTER USER システム権限は必要ではないためです。
VIEW	CREATE VIEW DROP VIEW

注意： Java スキーマ・オブジェクト（ソース、クラスおよびリソース）は、SQL 文の監査ではプロシージャと同じであるとみなされます。

表 13-2 監査のその他の SQL 文ショートカット

SQL 文ショートカット	監査済の SQL 文と操作
ALTER SEQUENCE	ALTER SEQUENCE
ALTER TABLE	ALTER TABLE
COMMENT TABLE	COMMENT ON TABLE *table*, *view*, *materialized view* COMMENT ON COLUMN *table.column*, *view.column*, *materialized view.column*
DELETE TABLE	DELETE FROM *table*, *view*
EXECUTE PROCEDURE	CALL プロシージャやファンクションの実行、またはパッケージ内の変数、ライブラリまたはカーソルへのアクセス。
GRANT DIRECTORY	GRANT *privilege* ON *directory* REVOKE *privilege* ON *directory*

表 13-2 監査のその他の SQL 文ショートカット（続き）

SQL 文ショートカット	監査済の SQL 文と操作
GRANT PROCEDURE	GRANT *privilege* ON *procedure*, *function*, package REVOKE *privilege* ON *procedure*, *function*, package
GRANT SEQUENCE	GRANT *privilege* ON *sequence* REVOKE *privilege* ON *sequence*
GRANT TABLE	GRANT *privilege* ON *table*, *view*, *materialized view* REVOKE *privilege* ON *table*, *view*, *materialized view*
GRANT TYPE	GRANT *privilege* ON TYPE REVOKE *privilege* ON TYPE
INSERT TABLE	INSERT INTO *table*, *view*
LOCK TABLE	LOCK TABLE *table*, *view*
SELECT SEQUENCE	*sequence*.CURRVAL または *sequence*.NEXTVAL を含む文
SELECT TABLE	SELECT FROM *table*, *view*, *materialized view*
UPDATE TABLE	UPDATE *table*, *view*

表 13-3 スキーマ・オブジェクト監査オプション

オブジェクト	SQL 操作
表	ALTER AUDIT COMMENT DELETE FLASHBACK（注意 3） GRANT INDEX INSERT LOCK RENAME SELECT UPDATE
ビュー	AUDIT COMMENT DELETE FLASHBACK（注意 3） GRANT INSERT LOCK RENAME SELECT UPDATE
順序	ALTER AUDIT GRANT SELECT
プロシージャ、ファンクション、パッケージ（注意 1）	AUDIT EXECUTE GRANT

表 13-3 スキーマ・オブジェクト監査オプション（続き）

オブジェクト	SQL 操作
マテリアライズド・ビュー（注意 2）	ALTER AUDIT COMMENT DELETE INDEX INSERT LOCK SELECT UPDATE
マイニング・モデル	AUDIT COMMENT GRANT RENAME SELECT
ディレクトリ	AUDIT GRANT READ
ライブラリ	EXECUTE GRANT
オブジェクト型	ALTER AUDIT GRANT
コンテキスト	AUDIT GRANT

注意 1: Java スキーマ・オブジェクト（ソース、クラスおよびリソース）は、監査オプションではプロシージャ、ファンクションおよびパッケージと同じであるとみなされます。

注意 2: INSERT、UPDATE および DELETE 操作の監査は、更新可能なマテリアライズド・ビューに対してのみ可能です。

注意 3: FLASHBACK 監査オブジェクト・オプションは、フラッシュバック問合せに対してのみ適用されます。

例

ロールに関連する SQL 文の監査例 : 次の文は、ロールの作成、変更、削除または設定を行う各 SQL 文が正常に終了したかどうかにかかわらず、それらの文について監査を行います。

```
AUDIT ROLE;
```

次の文は、ロールの作成、変更、削除または設定を行う、正常に終了した各 SQL 文ごとに監査を行います。

```
AUDIT ROLE
    WHENEVER SUCCESSFUL;
```

次の文は、Oracle Database エラーが発生した CREATE ROLE 文、ALTER ROLE 文、DROP ROLE 文または SET ROLE 文について監査を行います。

```
AUDIT ROLE
    WHENEVER NOT SUCCESSFUL;
```

問合せおよび更新を行う SQL 文の監査例： 次の文は、表の問合せまたは更新を実行する文について監査を行います。

```
AUDIT SELECT TABLE, UPDATE TABLE;
```

次の文は、ユーザー hr および oe が発行する、表やビューの問合せまたは更新を実行する文について監査を行います。

```
AUDIT SELECT TABLE, UPDATE TABLE
    BY hr, oe;
```

削除の監査例： 次の文は、DELETE ANY TABLE システム権限で発行された文について監査を行います。

```
AUDIT DELETE ANY TABLE;
```

ディレクトリに関連する文の監査例： 次の文は、CREATE ANY DIRECTORY システム権限で発行された文について監査を行います。

```
AUDIT CREATE ANY DIRECTORY;
```

CREATE ANY DIRECTORY システム権限を使用しない CREATE DIRECTORY（および DROP DIRECTORY）文を監査する場合は、次の文を発行します。

```
AUDIT DIRECTORY;
```

次の文は、bfile_dir ディレクトリからファイルを読み込む各文について監査を行います。

```
AUDIT READ ON DIRECTORY bfile_dir;
```

表の問合せの監査例： 次の文は、スキーマ hr 内の employees 表を問い合せる各 SQL 文について監査を行います。

```
AUDIT SELECT
    ON hr.employees;
```

次の文は、スキーマ hr 内の employees 表を問い合せて正常に終了した各文について監査を行います。

```
AUDIT SELECT
    ON hr.employees
    WHENEVER SUCCESSFUL;
```

次の文は、スキーマ hr 内の employees 表を問い合せて Oracle Database エラーが発生した SQL 文について監査を行います。

```
AUDIT SELECT
    ON hr.employees
    WHENEVER NOT SUCCESSFUL;
```

表の挿入および更新の監査例： 次の文は、スキーマ oe 内の customers 表に対して行を挿入または更新するそれぞれの文について監査を行います。

```
AUDIT INSERT, UPDATE
    ON oe.customers;
```

順序の操作の監査例： 次の文は、スキーマ hr 内の employees_seq 順序に対する操作を行うすべての文について監査を行います。

```
AUDIT ALL
    ON hr.employees_seq;
```

この文では、順序に対して操作を行う次の文について監査を行うため、ALL ショートカットを使用しています。

- ALTER SEQUENCE
- AUDIT
- GRANT
- 疑似列 CURRVAL または NEXTVAL を使用して、順序の値にアクセスするすべての文

デフォルト監査オプションの設定例： 次の文は、以降に作成されるオブジェクトのデフォルト監査オプションを指定します。

```
AUDIT ALTER, GRANT, INSERT, UPDATE, DELETE
    ON DEFAULT;
```

以降に作成されるオブジェクトについては、監査機能が使用可能な場合、指定したオプションによって自動的に次の監査が行われます。

- 表を作成した場合、その表に対して発行される ALTER 文、GRANT 文、INSERT 文、UPDATE 文または DELETE 文が自動的に監査されます。
- ビューを作成した場合、そのビューに対して発行される GRANT 文、INSERT 文、UPDATE 文または DELETE 文が自動的に監査されます。
- 順序を作成した場合、その順序に対して発行される ALTER 文または GRANT 文が自動的に監査されます。
- プロシージャ、パッケージまたはファンクションを作成した場合、それらに対して発行される ALTER 文または GRANT 文が自動的に監査されます。

CALL

用途

CALL 文を使用すると、SQL 内から**ルーチン**（スタンドアロン・プロシージャ、スタンドアロン・ファンクション、あるいは型またはパッケージ内で定義されたプロシージャまたはファンクション）を実行できます。

注意： 6-10 ページの「ファンクション式」に指定されているユーザー定義ファンクション式の制限は、CALL 文にも適用されます。

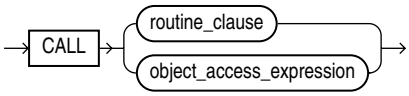
参照： これらのルーチンの作成の詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

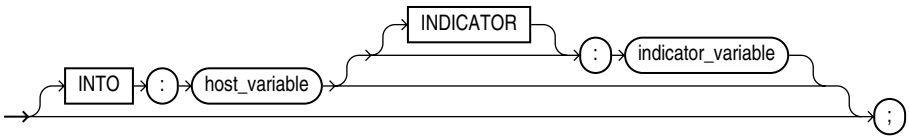
前提条件

スタンドアロン・ルーチン、またはルーチンが定義されている型またはパッケージに対する EXECUTE 権限が必要です。

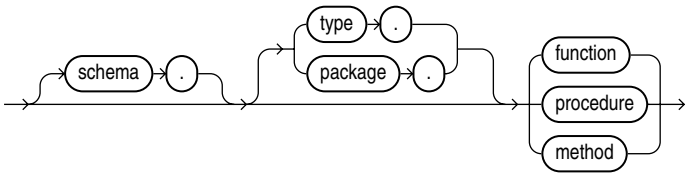
構文

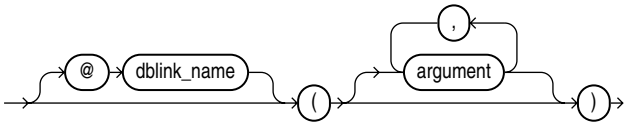
***call*::=**



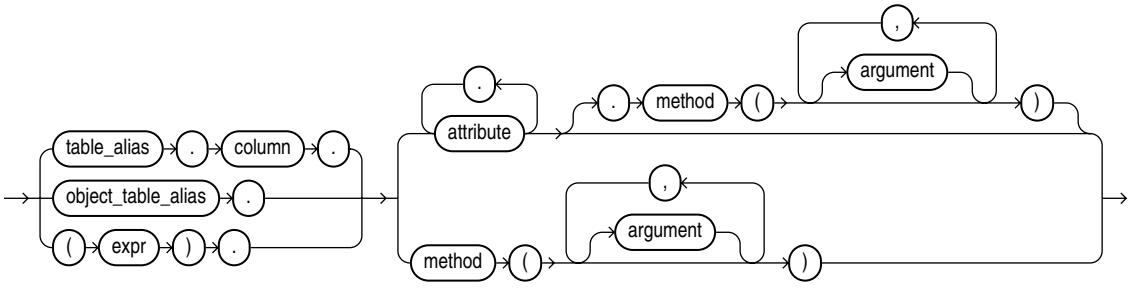


***routine_clause*::=**





***object_access_expression*::=**



セマンティクス

ルーチンは 2 つの方法で実行できます。*routine_clause* を使用してルーチン自体を名前でコールするか、または *object_access_expression* を使用して式の型の中でルーチンをコールします。

schema

スタンドアロン・ルーチン（または、ルーチンが含まれている型またはパッケージ）が存在するスキーマを指定します。*schema* を指定しない場合、ルーチンは自分のスキーマ内にあるとみなされます。

type* または *package

ルーチンが定義されている型またはパッケージを指定します。

routine_clause

コールするファンクション名またはプロシージャ名、またはファンクションまたはプロシージャに変換されるシノニムを指定します。

型のメンバー・ファンクションまたはメンバー・プロシージャをコールする場合、最初の引数（SELF）が NULL の IN OUT の場合は、エラーが戻されます。SELF が NULL の IN の場合には、NULL が戻されます。どちらの場合も、ファンクションまたはプロシージャはコールされません。

ファンクションの制限事項： ルーチンがファンクションの場合、INTO 句は必須です。

@dblink

分散データベース・システムで、スタンドアロン・ルーチンが含まれているデータベース、またはルーチンが含まれているパッケージまたはファンクションの名前を指定します。*dblink* を指定しない場合、ローカル・データベースを指定したとみなされます。

> **参照：** 直接ルーチンをコールする例については、13-39 ページの「プロシージャのコール例：」を参照してください。

object_access_expression

型コンストラクタ、バインド変数などのオブジェクト型の式を使用している場合は、この形式の式を使用して、型内で定義されたルーチンをコールできます。このコンテキストでは、*object_access_expression* はメソッドのコールに制限されます。

> **参照：** この形式の式の構文およびセマンティクスについては、6-13 ページの「オブジェクト・アクセス式」を参照してください。オブジェクト型の式を使用してルーチンをコールする例については、13-39 ページの「オブジェクト型の式を使用したプロシージャのコール例：」を参照してください。

argument

ルーチンに引数が必要な場合に、ルーチンの 1 つ以上の引数を指定します。*argument* を、位置表記法、名前表記法および混合表記法で表記できます。たとえば、次の表記はすべて正しい表記です。

```
CALL my_procedure(arg1 => 3, arg2 => 4)

CALL my_procedure(3, 4)

CALL my_procedure(3, arg2 => 4)
```

ルーチンへの引数の適用の制限事項： *argument* には、次の制限事項があります。

- CALL 文によって渡されるパラメータのデータ型は、SQL のデータ型であることが必要です。BOOLEAN など、PL/SQL 専用のデータ型は使用できません。
- 引数には、疑似列、オブジェクト参照ファンクション VALUE および REF は指定できません。
- ルーチンの IN OUT 引数または OUT 引数であるすべての引数は、ホスト変数の式に対応している必要があります。
- すべての戻り引数を含む引数の数は、1000 に制限されています。
- 4KB を超える文字データ型または RAW データ型（CHAR、VARCHAR2、NCHAR、NVARCHAR2、RAW、LONG RAW）の引数はバインドできません。

INTO :*host_variable*

INTO 句は、ファンクションのコールにのみ適用されます。ファンクションの戻り値を格納するホスト変数を指定します。

:*indicator_variable*

ホスト変数の値または状態を指定します。

> **参照：** ホスト変数および標識変数の詳細は、『Pro*C/C++ プログラマーズ・ガイド』を参照してください。

例

プロシージャのコール例： 次の文は、remove_dept プロシージャを使用して、娯楽部門（18-65 ページの「順序値の挿入例 :」で作成）を削除します。このプロシージャを作成する例については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

```
CALL emp_mgmt.remove_dept(162);
```

オブジェクト型の式を使用したプロシージャのコール例： 次の例では、CALL 文のオブジェクト型の式を使用したプロシージャのコール方法を示します。この例では、サンプルの注文入力スキーマ OE で warehouse_typ オブジェクト型を使用しています。

```
ALTER TYPE warehouse_typ
      ADD MEMBER FUNCTION ret_name
      RETURN VARCHAR2
      CASCADE;

CREATE OR REPLACE TYPE BODY warehouse_typ
      AS MEMBER FUNCTION ret_name
      RETURN VARCHAR2
      IS
         BEGIN
            RETURN self.warehouse_name;
         END;
      END;
```

```
/
VARIABLE x VARCHAR2(25);

CALL warehouse_typ(456, 'Warehouse 456', 2236).ret_name()
   INTO :x;

PRINT x;
X
--------------------------------
Warehouse 456
```

次の例では、外部ファンクションを使用したプロシージャのコール方法を示します。

```
CREATE OR REPLACE FUNCTION ret_warehouse_typ(x warehouse_typ)
  RETURN warehouse_typ
  IS
    BEGIN
      RETURN x;
    END;
/
CALL ret_warehouse_typ(warehouse_typ(234, 'Warehouse 234',
   2235)).ret_name()
   INTO :x;

PRINT x;

X
--------------------------------
Warehouse 234
```

COMMENT

用途

COMMENT 文を使用すると、表または表の列、ビュー、マテリアライズド・ビュー、演算子、索引タイプまたはマイニング・モデルに関するコメントを、データ・ディクショナリに追加できます。

データベースからコメントを削除する場合、空の文字列 '' を設定します。

参照 :

- SQL 文およびスキーマ・オブジェクトへのコメントの関連付けの詳細は、2-69 ページの「コメント」を参照してください。
- コメントを表示するデータ・ディクショナリ・ビューの詳細は、『Oracle Database リファレンス』を参照してください。

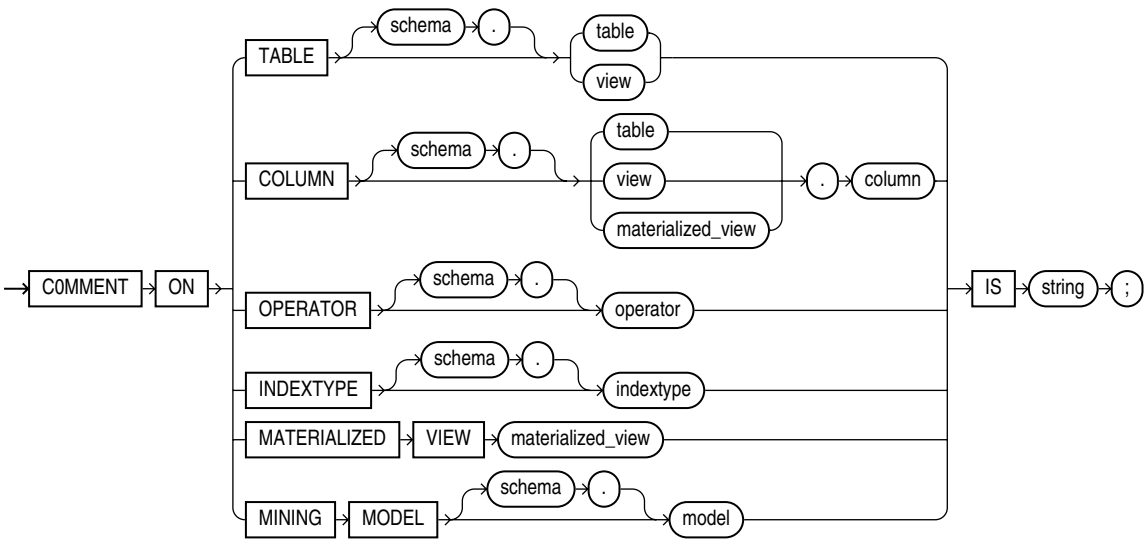
前提条件

コメントを追加するオブジェクトが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、次の条件を満たす必要があります。

- 表、ビューまたはマテリアライズド・ビューにコメントを追加する場合は、COMMENT ANY TABLE システム権限を持っている。
- 索引タイプにコメントを追加する場合は、CREATE ANY INDEXTYPE システム権限を持っている。
- 演算子にコメントを追加する場合は、CREATE ANY OPERATOR システム権限を持っている。

構文

***comment*::=**



セマンティクス

TABLE 句

コメントする表またはマテリアライズド・ビューの名前とスキーマを指定します。*schema* を指定しない場合、この表およびマテリアライズド・ビューは自分のスキーマ内にあるとみなされます。

注意： 以前のリリースでは、この句を使用してマテリアライズド・ビューにコメントを作成できました。今回のリリースでは、マテリアライズド・ビューに、COMMENT ON MATERIALIZED VIEW 句を使用する必要があります。

COLUMN 句

コメントする表、ビューまたはマテリアライズド・ビューの列の名前を指定します。*schema* を指定しない場合、この表、ビューおよびマテリアライズド・ビューは自分のスキーマ内にあるとみなされます。

データ・ディクショナリ・ビュー USER_TAB_COMMENTS、DBA_TAB_COMMENTS、ALL_TAB_COMMENTS、USER_COL_COMMENTS、DBA_COL_COMMENTS または ALL_COL_COMMENTS を問い合せることによって、特定の表または列に関するコメントを表示できます。

OPERATOR 句

コメントする演算子の名前を指定します。*schema* を指定しない場合、その演算子は自分のスキーマ内にあるとみなされます。

データ・ディクショナリ・ビュー USER_OPERATOR_COMMENTS、DBA_OPERATOR_COMMENTS または ALL_OPERATOR_COMMENTS を問い合せることによって、特定の演算子に関するコメントを表示できます。

INDEXTYPE 句

コメントする索引タイプの名前を指定します。*schema* を指定しない場合、この索引タイプは自分のスキーマ内にあるとみなされます。

データ・ディクショナリ・ビュー USER_INDEXTYPE_COMMENTS、DBA_INDEXTYPE_COMMENTS または ALL_INDEXTYPE_COMMENTS を問い合せることによって、特定の索引タイプに関するコメントを表示できます。

MATERIALIZED VIEW 句

コメントするマテリアライズド・ビューの名前を指定します。*schema* を指定しない場合、マテリアライズド・ビューは自分のスキーマ内にあるとみなされます。

データ・ディクショナリ・ビュー USER_MVIEW_COMMENTS、DBA_MVIEW_COMMENTS または ALL_MVIEW_COMMENTS を問い合せることによって、特定のマテリアライズド・ビューに関するコメントを表示できます。

MINING MODEL

コメントするマイニング・モデルの名前を指定します。この句を指定するには、COMMENT ANY MINING MODEL システム権限が必要です。

IS '*string*'

コメントのテキストを指定します。*'string'* の構文の詳細は、2-44 ページの「テキスト・リテラル」を参照してください。

例

コメントの作成例： 次の文は、employees 表の job_id 列にコメントを挿入します。

```
COMMENT ON COLUMN employees.job_id
   IS 'abbreviated job title';
```

次の文は、データベースからこのコメントを削除します。

```
COMMENT ON COLUMN employees.job_id IS ' ';
```

COMMIT

用途

COMMIT 文を使用すると、現行のトランザクションを終了し、トランザクションで実行したすべての変更を確定できます。**トランザクション**とは、Oracle Database が 1 つの単位として扱う一連の SQL 文です。また、この文によって、トランザクション内のセーブポイントがすべて消去され、トランザクション・ロックが解除されます。

トランザクションをコミットするまでは、次の操作が可能です。

- 変更された表を問い合せることで、トランザクション中に加えた変更内容を確認する。ただし、他のユーザーは変更内容を参照できません。トランザクションのコミットが終了すると、コミット後に実行される他のユーザーの文で変更内容を参照できます。
- トランザクション中に行った変更を、ROLLBACK 文でロールバックする（元に戻す）。18-92 ページの「ROLLBACK」を参照してください。

Oracle Database では、データ定義言語（DDL）文の前後で暗黙的に COMMIT が発行されます。

この文を使用して、次の操作を実行することもできます。

- インダウト分散トランザクションを手動でコミットします。
- SET TRANSACTION 文で開始した読取り専用トランザクションを終了します。

Oracle Database との接続を切断する前に、最新のトランザクションを含むアプリケーション・プログラムのすべてのトランザクションを、COMMIT 文または ROLLBACK 文を使用して明示的に終了することをお薦めします。トランザクションを明示的にコミットしなかった場合にプログラムが異常終了すると、コミットされていない最後のトランザクションは、自動的にロールバックされます。

Oracle ユーティリティおよび Oracle のツール製品が正常に終了すると、現行のトランザクションがコミットされます。Oracle プリコンパイラ・プログラムが正常に終了した場合は、トランザクションはコミットされず、現行のトランザクションが Oracle Database によってロールバックされます。

参照：

- トランザクションの詳細は、『Oracle Database 概要』を参照してください。
- トランザクションの特性の指定の詳細は、19-53 ページの「SET TRANSACTION」を参照してください。

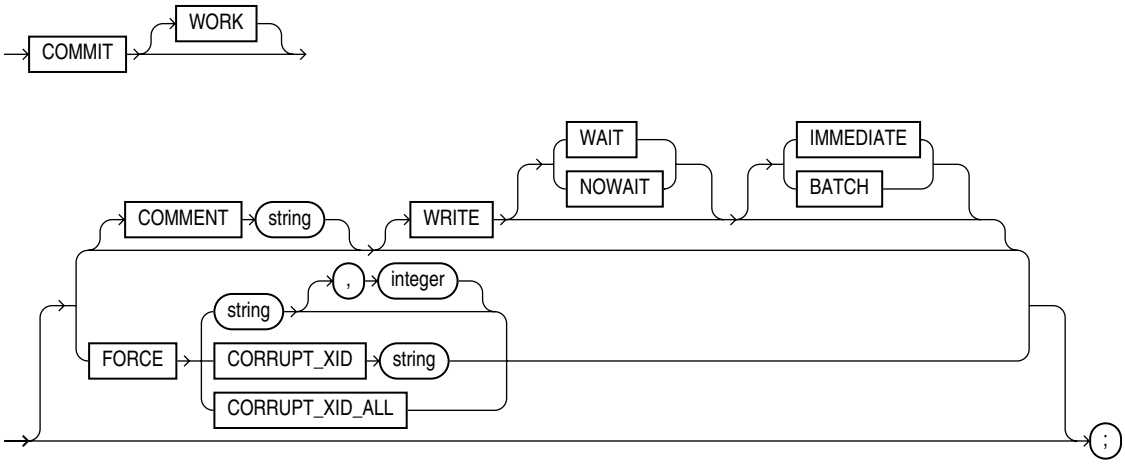
前提条件

現行のトランザクションをコミットするために、必要な権限は特にありません。

自分がコミットしたインダウト分散トランザクションを手動でコミットする場合は、FORCE TRANSACTION システム権限が必要です。別のユーザーがコミットしたインダウト分散トランザクションを手動でコミットする場合は、FORCE ANY TRANSACTION システム権限が必要です。

構文

***commit*::=**



セマンティクス

COMMIT

COMMIT キーワードに続く句は、すべてオプションです。COMMIT のみを指定した場合、デフォルトは COMMIT WORK WRITE IMMEDIATE WAIT です。

WORK

標準 SQL に準拠するために、WORK キーワードがサポートされています。COMMIT 文と COMMIT WORK 文は同じです。

COMMENT 句

この句は、下位互換性を保つためにのみサポートされています。コミット・コメントのかわりに名前付きトランザクションを使用することをお薦めします。

> **参照：** 名前付きトランザクションの詳細は、19-53 ページの「SET TRANSACTION」および『Oracle Database 概要』を参照してください。

現行のトランザクションに関するコメントを指定します。*'text'* は引用符で囲まれた最大 255 バイトのリテラルで、分散トランザクションの状態が不明（インダウト）になった場合に、そのトランザクション ID とともに、データ・ディクショナリ・ビュー DBA_2PC_PENDING に格納されます。このコメントは、分散トランザクションの障害を診断するときに役立ちます。

> **参照：** SQL 文へのコメントの追加の詳細は、13-41 ページの「COMMENT」を参照してください。

WRITE 句

この句を使用すると、コミット操作で生成される REDO 情報を REDO ログに書き込む優先度を指定できます。この句によって、待機時間を減らし REDO ログへの I/O を待機しないようにすることで、パフォーマンスを向上させることができます。この句は、レスポンス時間に対する要件が厳しい次のような環境下でのレスポンス時間を改善するために使用します。

- 更新トランザクションの量が多く、REDO ログを頻繁にディスクに書き込む必要がある。
- アプリケーションが、非同期でコミットされるトランザクションの消失を許容できる。
- REDO ログの書込みの発生を待つ待機時間が、全体のレスポンス時間に大きく影響する。

WAIT | NOWAIT および IMMEDIATE | BATCH 句を任意の順序で指定できます。

> **注意：** この句を省略したときのコミット操作は COMMIT_WRITE 初期化パラメータで制御されます（パラメータが設定されている場合）。パラメータのデフォルト値は、この句のデフォルトと同じです。したがって、パラメータが設定されていないときにこの句を省略すると、コミット・レコードがディスクに書き込まれてから制御がユーザーに戻ります。

WAIT | NOWAIT　この句を使用すると、制御をいつユーザーに戻すかを指定できます。

- WAIT パラメータを指定すると、対応する REDO がオンライン REDO ログで永続的になった後にのみコミットが戻ります。BATCH モードでも IMMEDIATE モードでも、この COMMIT 文から正常にクライアントに戻ったときは、トランザクションは永続メディアにコミットされています。ログへの正常な書込みの後で障害が発生した場合、成功のメッセージがクライアントに戻らない場合があります。この場合には、トランザクションがコミットされたかどうかはクライアントにはわかりません。
- NOWAIT パラメータを指定すると、REDO ログへの書込みが完了したかどうかに関係なく、コミットはクライアントに戻ります。この動作によって、トランザクションのスループットが向上します。WAIT パラメータを指定すると、コミット・メッセージを受け取った場合にデータの損失がないことがわかります。

> **注意：** NOWAIT を指定すると、コミット・メッセージを受け取った後で、REDO ログ・レコードが書き込まれる前に障害が発生した場合、その変更が永続的であることをトランザクションに誤って示す場合があります。

この句を指定しない場合、トランザクションは WAIT の動作でコミットされます。

IMMEDIATE | BATCH　この句を使用すると、REDO をいつログに書き込むかを指定できます。

- IMMEDIATE パラメータを指定すると、ログ・ライター・プロセス（LGWR）によって、トランザクションの REDO 情報がログに書き込まれます。この操作オプションはディスク I/O を強制するため、トランザクションのスループットは低下します。
- BATCH パラメータを指定すると、同時に実行されている他のトランザクションとともに、REDO が REDO ログにバッファされます。十分な REDO 情報が収集されると、REDO ログのディスク書込みが開始されます。この動作はグループ・コミットと呼ばれます。複数のトランザクションの REDO が一度の I/O 操作でログに書き込まれるためです。

この句を指定しない場合、トランザクションは IMMEDIATE の動作でコミットされます。

> **参照：** 非同期のコミットの詳細は、『Oracle Database アドバンスト・アプリケーション開発者ガイド』を参照してください。

FORCE 句

この句を使用すると、インダウト分散トランザクションまたは破損トランザクションを手動でコミットできます。

- 分散データベース・システムでは、FORCE *string* [, *integer*] 句によって、手動でインダウト分散トランザクションをコミットできます。このトランザクションは、ローカル・トランザクション ID またはグローバル・トランザクション ID を含む '*string*' で識別されます。このトランザクションの ID を確認する場合、データ・ディクショナリ・ビュー DBA_2PC_PENDING を問い合せます。また、*integer* を指定することによって、このトランザクションにシステム変更番号（SCN）を具体的に割り当てることもできます。*integer* を指定しない場合、このトランザクションは現行の SCN を使用してコミットされます。

- FORCE CORRUPT_XID '*string*' 句によって、1 つの破損トランザクションを手動でコミットできます。*string* は、破損トランザクションの ID です。V$CORRUPT_XID_LIST データ・ディクショナリ・ビューを問い合せて、破損トランザクションのトランザクション ID を検索します。V$CORRUPT_XID_LIST を表示し、この句を指定するには、DBA 権限が必要です。
- FORCE CORRUPT_XID_ALL を指定すると、すべての破損トランザクションを手動でコミットできます。この句を指定するには、DBA 権限が必要です。

注意： FORCE 句を指定して COMMIT 文を発行した場合、指定したトランザクションのみがコミットされます。この文は、現行のトランザクションには影響しません。

参照： 前述の項目の詳細は、『Oracle Database 管理者ガイド』を参照してください。

例

挿入のコミット例： 次の文は、hr.regions 表に行を挿入してこの変更をコミットします。

```
INSERT INTO regions VALUES (5, 'Antarctica');

COMMIT WORK;
```

同じ挿入操作をコミットし、データベースに対して、ディスク I/O を開始せずに変更を REDO ログにバッファするよう指示するには、次の COMMIT 文を使用します。

```
COMMIT WRITE BATCH;
```

COMMIT のコメント例： 次の文は、現行のトランザクションをコミットして、コメントを関連付けます。

```
COMMIT
    COMMENT 'In-doubt transaction Code 36, Call (415) 555-2637';
```

ネットワーク障害またはマシン障害によって分散トランザクションを適切にコミットできない場合、トランザクション ID とともにデータ・ディクショナリにコメントが格納されます。そのコメントには、障害が発生したアプリケーション部分が示されており、トランザクションがコミットされたデータベースの管理者に連絡する情報が提供されています。

強制インダウト・トランザクションの例： 次の文は、不確定なインダウト分散トランザクションを手動でコミットします。V$CORRUPT_XID_LIST データ・ディクショナリ・ビューを問い合せて、破損トランザクションのトランザクション ID を検索します。V$CORRUPT_XID_LIST を表示し、この文を発行するには、DBA 権限が必要です。

```
COMMIT FORCE '22.57.53';
```

14

SQL 文 : CREATE CLUSTER ～ CREATE JAVA

この章では、次の SQL 文について説明します。

- CREATE CLUSTER
- CREATE CONTEXT
- CREATE CONTROLFILE
- CREATE DATABASE
- CREATE DATABASE LINK
- CREATE DIMENSION
- CREATE DIRECTORY
- CREATE DISKGROUP
- CREATE FLASHBACK ARCHIVE
- CREATE FUNCTION
- CREATE INDEX
- CREATE INDEXTYPE
- CREATE JAVA

CREATE CLUSTER

用途

CREATE CLUSTER 文を使用すると、クラスタを作成できます。**クラスタ**とは、1 つ以上の表のデータが含まれているスキーマ・オブジェクトのことです。

- **索引クラスタ**には複数のクラスタが含まれている必要があり、クラスタ内のすべての表には 1 つ以上の共通の列があります。Oracle Database では、同一のクラスタ・キーを共有するすべての表からすべての行がまとめて格納されます。
- **ハッシュ・クラスタ**（1 つ以上の表を含むことが可能）には、同一のハッシュ・キー値を持つ行がまとめて格納されます。

既存のクラスタの情報を取得するには、データ・ディクショナリ・ビュー USER_CLUSTERS、ALL_CLUSTERS および DBA_CLUSTERS を問い合せます。

参照：

- クラスタの概要は、『Oracle Database 概要』を参照してください。
- クラスタを使用するタイミングついては、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。
- データ・ディクショナリ・ビューの詳細は、『Oracle Database リファレンス』を参照してください。

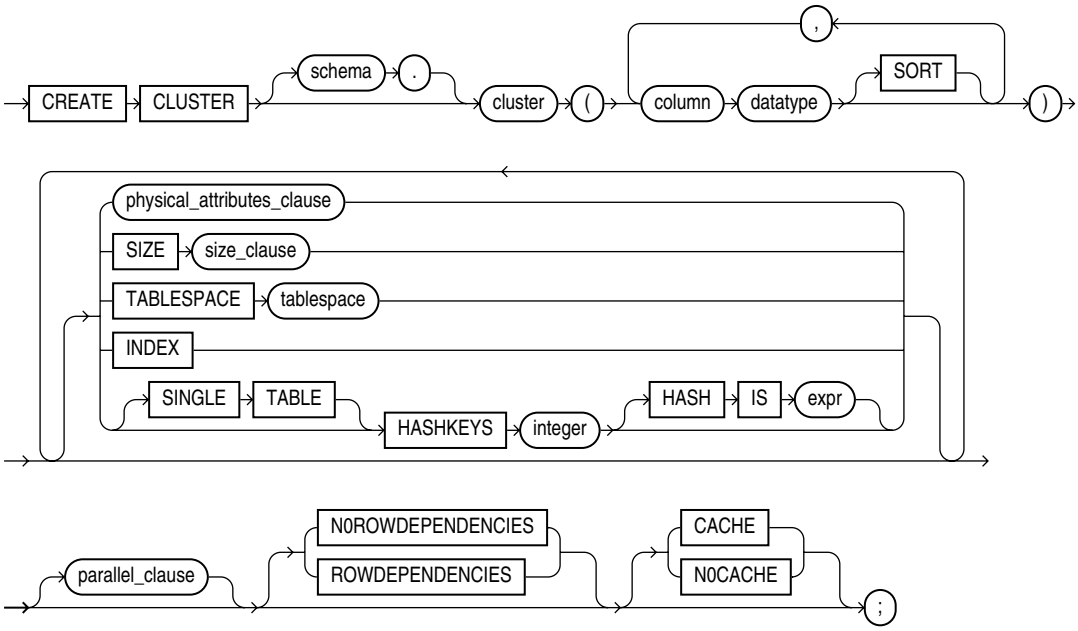
前提条件

自分のスキーマにクラスタを作成する場合は、CREATE CLUSTER システム権限が必要です。他のユーザーのスキーマ内にクラスタを作成する場合は、CREATE ANY CLUSTER システム権限が必要です。また、クラスタを設定するスキーマの所有者は、クラスタが定義されている表領域に対する領域の割当て制限、または UNLIMITED TABLESPACE システム権限のいずれかが必要です。

クラスタを最初に作成する場合は、Oracle Database はクラスタに対する索引を自動的に作成しません。このため、CREATE INDEX 文でクラスタ索引を作成するまでは、索引クラスタのクラスタ表に対して、データ操作言語（DML）文を発行できません。

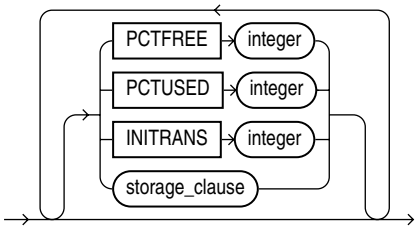
構文

***create_cluster*::=**



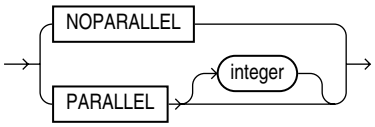
（14-3 ページの physical_attributes_clause::=、8-42 ページの size_clause::= を参照）

physical_attributes_clause::=



（8-44 ページの storage_clause::= を参照）

parallel_clause::=



セマンティクス

schema

作成するクラスタが定義されるスキーマを指定します。*schema* を指定しない場合、現行のスキーマにクラスタが作成されます。

cluster

作成するクラスタの名前を指定します。

クラスタを作成した後、そのクラスタに表を追加します。クラスタには、最大 32 個の表を指定できます。クラスタを作成し、そのクラスタに表を追加しても、クラスタを意識する必要はありません。クラスタ化されていない表と同様に、SQL 文を使用してクラスタ化表にアクセスできます。

> **参照：** クラスタへの表の追加については、16-6 ページの「CREATE TABLE」、14-7 ページの「クラスタの作成例 :」および 14-7 ページの「クラスタへの表の追加例 :」を参照してください。

column

クラスタ・キーに 1 つ以上の列名を指定します。最大 16 個のクラスタ・キー列を指定できます。これらの列は、データ型およびサイズについて、クラスタ化表の列と対応している必要があります。名前は対応している必要はありません。

クラスタ・キー列の定義の一部として整合性制約は指定できません。そのかわり、クラスタに属している表に整合性制約を対応付けることができます。

> **参照：** 「クラスタ・キーの例 :」（14-7 ページ）

datatype

各クラスタ・キー列のデータ型を指定します。

クラスタのデータ型の制限事項： クラスタのデータ型には、次の制限事項があります。

- データ型が `LONG`、`LONG RAW`、`REF`、ネストした表、VARRAY、`BLOB`、`CLOB`、`BFILE`、Oracle が提供する型 `Any*` またはユーザー定義オブジェクト型であるクラスタ・キー列は指定できません。
- `ROWID` 型の列を指定することはできますが、それらの列の値が有効な行 ID であることは保証されません。

> **参照：** データ型の詳細は、2-2 ページの「データ型」を参照してください。

SORT

`SORT` キーワードは、ハッシュ・クラスタを作成する場合にのみ有効です。この句を指定すると、Oracle Database に対して、ハッシュ・ファンクションを適用する前にこの列でクラスタの行をソートするように指示できます。これによって、クラスタ化データでの後続の操作時に、応答時間が短縮される場合があります。ハッシュ・クラスタの作成については、14-5 ページの「HASHKEYS 句」を参照してください。

physical_attributes_clause

`physical_attributes_clause` を使用すると、クラスタの記憶特性を指定できます。クラスタ内の各表もこれらの記憶特性を使用します。これらのパラメータの値を指定しない場合、次のデフォルトが使用されます。

- `PCTFREE`: 10
- `PCTUSED`: 40
- `INITRANS`: 2、またはクラスタを含む表領域のデフォルト値のいずれか大きい方

> **参照：** これらの句の詳細は、8-39 ページの「physical_attributes_clause」および 8-41 ページの「storage_clause」を参照してください。

SIZE

同一クラスタ・キー値または同一ハッシュ値を持つすべての行を格納するために確保する領域を、バイト単位で指定します。次に、この領域によって、データ・ブロックごとに格納されるクラスタやハッシュ値の最大値が決まります。SIZE の値がデータ・ブロック・サイズの約数でない場合、Oracle Database は、次に大きい約数を使用します。SIZE がデータ・ブロック・サイズより大きい場合、データベースは、クラスタまたはハッシュ値ごとに、1 つ以上のデータ・ブロックを確保し、オペレーティング・システムのブロック・サイズを採用します。

データベースは、クラスタ・キー値を持つ行に対して確保する必要がある領域を決定する際に、クラスタ・キーの長さを考慮します。クラスタ・キーが大きければ、それに必要なサイズも大きくなります。実際のサイズを参照するには、USER_CLUSTERS データ・ディクショナリ・ビューの KEY_SIZE 列を問い合せます（ハッシュ値は実際にはクラスタ内に格納されていないため、この値はハッシュ・クラスタには適用されません）。

このパラメータを指定しない場合、各クラスタ・キー値またはハッシュ値ごとにデータ・ブロックが 1 つ確保されます。

TABLESPACE

クラスタを作成する表領域を指定します。

INDEX 句

INDEX を指定すると、**索引クラスタ**を作成できます。索引クラスタには、同一のクラスタ・キー値が指定されている行がまとめて格納されます。それぞれのクラスタ・キー値は、そのキーを持つ表および行の数に関係なく、各データ・ブロックに 1 回のみ格納されます。INDEX も HASHKEYS も指定しない場合は、デフォルトで索引クラスタが作成されます。

索引クラスタの作成後、クラスタ内の表に対してデータ操作言語（DML）文を発行する前に、そのクラスタ・キーに索引を作成する必要があります。この索引を**クラスタ索引**と呼びます。

ハッシュ・クラスタに対してクラスタ索引は作成できないため、ハッシュ・クラスタ・キーで索引を作成する必要はありません。

> **参照：** クラスタ索引の作成方法の詳細は 14-50 ページの「CREATE INDEX」、索引クラスタの概要は『Oracle Database 概要』を参照してください。

HASHKEYS 句

HASHKEYS を指定すると、**ハッシュ・クラスタ**を作成し、ハッシュ・クラスタのハッシュ値の数を指定できます。ハッシュ・クラスタには、同一のハッシュ・キー値を持つ行がまとめて格納されます。それぞれの行のハッシュ値は、そのクラスタのハッシュ・ファンクションが戻す値です。

Oracle Database は、ハッシュ値の実際の数を決めるため、HASHKEYS 値を一番近い次の素数に切り上げます。このパラメータの最小値は 2 です。INDEX 句と HASHKEYS パラメータの両方を指定しないとき、デフォルトで索引クラスタが作成されます。

ハッシュ・クラスタの作成時に、データベースは、SIZE パラメータおよび HASHKEYS パラメータの値に基づいて、クラスタに領域を割り当てます。

> **参照：** Oracle Database がクラスタに領域を割り当てる方法の詳細は、『Oracle Database 概要』および 14-7 ページの「ハッシュ・クラスタの例 :」を参照してください。

SINGLE TABLE SINGLE TABLE を指定すると、表を 1 つのみ持つタイプのハッシュ・クラスタを作成できます。この句によって、表がクラスタの一部でない場合より行へのアクセスが高速になります。

単一表クラスタの制限事項： 同時にクラスタに存在できる表は 1 つのみです。ただし、表を削除して、同一のクラスタに別の表を作成することはできます。

> **参照：** 「単一表ハッシュ・クラスタの例：」（14-7 ページ）

HASH IS *expr* ハッシュ・クラスタに対するハッシュ・ファンクションとして使用する式を指定します。式には、次の制限事項があります。

- 正の値に評価される必要があります。
- 式全体の値が位取り 0（ゼロ）の数になる場合は、任意のデータ型の列が参照される 1 つ以上の列を持つ必要があります。たとえば、*number_column* * LENGTH(*varchar2_column*) です。
- ユーザー定義の PL/SQL ファンクションを参照できません。
- 疑似列 LEVEL または ROWNUM を参照できません。
- ユーザー関連ファンクション（USERENV、UID および USER）または日時ファンクション（CURRENT_DATE、CURRENT_TIMESTAMP、DBTIMEZONE、EXTRACT（日時）、FROM_TZ、LOCALTIMESTAMP、NUMTODSINTERVAL、NUMTOYMINTERVAL、SESSIONTIMEZONE、SYSDATE、SYSTIMESTAMP、TO_DSINTERVAL、TO_TIMESTAMP、TO_DATE、TO_TIMESTAMP_TZ、TO_YMINTERVAL および TZ_OFFSET）を参照できません。
- 定数に評価されることはありません。
- スカラー副問合せ式には指定できません。
- （クラスタ名ではなく）スキーマ名またはオブジェクト名で修飾された列を持つことができません。

HASH IS 句を指定しない場合、Oracle Database はハッシュ・クラスタに対して内部ハッシュ・ファンクションを使用します。

既存のクラスタの詳細は、データ・ディクショナリ表 USER_CLUSTER_HASH_EXPRESSIONS、ALL_CLUSTER_HASH_EXPRESSIONS および DBA_CLUSTER_HASH_EXPRESSIONS に問い合せます。

ハッシュ列のクラスタ・キーは、任意のデータ型で構成される 1 つ以上の列を持つことができます。複合クラスタ・キー、または整数以外の列で構成されるクラスタ・キーを持つハッシュ・クラスタに対しては、内部ハッシュ・ファンクションを使用する必要があります。

> **参照：** データ・ディクショナリ・ビューの詳細は、『Oracle Database リファレンス』を参照してください。

parallel_clause

parallel_clause を使用すると、クラスタの作成をパラレル化できます。

この句の詳細は、16-54 ページの「CREATE TABLE」の「parallel_clause」を参照してください。

NOROWDEPENDENCIES | ROWDEPENDENCIES

この句のクラスタに対する動作は、表に対する動作と同じです。詳細は、16-54 ページの「CREATE TABLE」の「NOROWDEPENDENCIES | ROWDEPENDENCIES」を参照してください。

CACHE | NOCACHE

CACHE CACHE を指定すると、全表スキャンの実行時に、このクラスタに対して取り出されたブロックを、バッファ・キャッシュ内の最高使用頻度（LRU）リストの最高使用頻度側に入れることができます。この句は、小規模な参照表で有効です。

NOCACHE NOCACHE を指定すると、全表スキャンの実行時に、このクラスタに対して取り出されたブロックを、バッファ・キャッシュ内の LRU リストの最低使用頻度側に入れることができます。これはデフォルトの動作です。

NOCACHE は、*storage_clause* に KEEP を指定したクラスタには影響しません。

例

クラスタの作成例： 次の文は、クラスタ・キー列 department、クラスタ・サイズ 512 バイトおよび記憶域パラメータ値を指定した索引クラスタ personnel を作成します。

```
CREATE CLUSTER personnel
   (department NUMBER(4))
SIZE 512
STORAGE (initial 100K next 50K);
```

クラスタ・キーの例： 次の文は、personnel のクラスタ・キーにクラスタ索引を作成します。

```
CREATE INDEX idx_personnel ON CLUSTER personnel;
```

クラスタ索引の作成後、索引に表を追加し、その表に対して DML 操作を行うことができます。

クラスタへの表の追加例： 次の文は、サンプル表 hr.employees から部門表を作成し、前述の例で作成した personnel クラスタに追加します。

```
CREATE TABLE dept_10
   CLUSTER personnel (department_id)
   AS SELECT * FROM employees WHERE department_id = 10;

CREATE TABLE dept_20
   CLUSTER personnel (department_id)
   AS SELECT * FROM employees WHERE department_id = 20;
```

ハッシュ・クラスタの例： 次の文は、クラスタ・キー列 cust_language、最大ハッシュ・キー値 10（各サイズ 512 バイト）および記憶域パラメータ値を指定したハッシュ・クラスタ language を作成します。

```
CREATE CLUSTER language (cust_language VARCHAR2(3))
   SIZE 512 HASHKEYS 10
   STORAGE (INITIAL 100k next 50k);
```

この文では、HASH IS 句を指定していないため、Oracle Database は、そのクラスタに対して内部ハッシュ・ファンクションを採用します。

次の文は、postal_code と country_id 列で構成されるクラスタ・キーを持つ address という名前のハッシュ・クラスタを作成し、これらの列を含む SQL 式をハッシュ・ファンクションに使用します。

```
CREATE CLUSTER address
   (postal_code NUMBER, country_id CHAR(2))
   HASHKEYS 20
   HASH IS MOD(postal_code + country_id, 101);
```

単一表ハッシュ・クラスタの例： 次の文は、クラスタ・キー customer_id および最大ハッシュ・キー値 100（各サイズ 512 バイト）を指定した単一表ハッシュ・クラスタ cust_orders を作成します。

```
CREATE CLUSTER cust_orders (customer_id NUMBER(6))
   SIZE 512 SINGLE TABLE HASHKEYS 100;
```

CREATE CONTEXT

用途

CREATE CONTEXT 文を使用すると、次の操作を実行できます。

- **コンテキスト**（アプリケーションを検証および保護するアプリケーション定義の一連の属性）のネームスペースを作成します。
- ネームスペースを、コンテキストを設定する外部作成パッケージと関連付けます。

専用パッケージの DBMS_SESSION.SET_CONTEXT プロシージャを使用して、コンテキスト属性を設定または再設定できます。

参照：

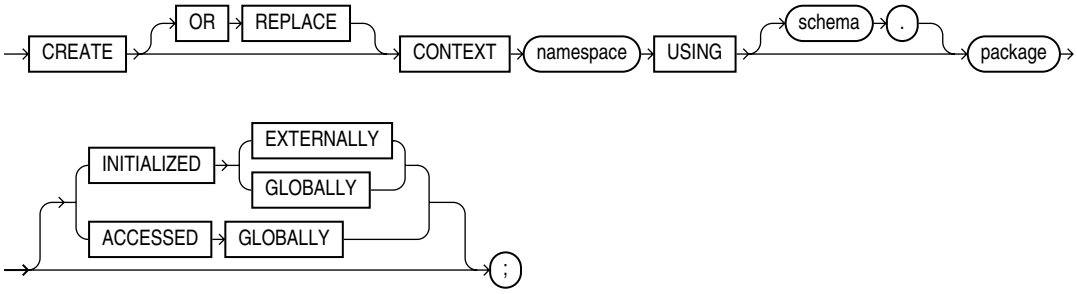
- コンテキストについては、『Oracle Database セキュリティ・ガイド』を参照してください。
- DBMS_SESSION.SET_CONTEXT プロシージャについては、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

前提条件

コンテキスト・ネームスペースを作成する場合、CREATE ANY CONTEXT システム権限が必要です。

構文

***create_context*::=**



セマンティクス

OR REPLACE

OR REPLACE を指定すると、異なるパッケージを使用して既存のコンテキスト・ネームスペースを再定義できます。

namespace

作成または変更するコンテキスト・ネームスペース名を指定します。コンテキスト・ネームスペースは、常にスキーマ SYS に格納されます。

参照： コンテキスト・ネームスペースのネーミングに関するガイドラインは、2-98 ページの「スキーマ・オブジェクトのネーミング規則」を参照してください。

schema

package を所有するスキーマを指定します。*schema* を指定しない場合、Oracle Database は現行のスキーマを使用します。

package

ユーザー・セッションのネームスペースに基づくコンテキスト属性を設定または再設定する PL/SQL パッケージを指定します。

柔軟に設計できるように、Oracle Database は、コンテキスト作成時のスキーマの存在またはパッケージの妥当性を検証しません。

INITIALIZED 句

INITIALIZED 句を使用すると、コンテキスト・ネームスペースを初期化する Oracle Database 以外のエンティティを指定できます。

EXTERNALLY EXTERNALLY を使用すると、セッション確立時に OCI インタフェースを使用してネームスペースを初期化できます。

> **参照：** OCI を使用したセッション確立の詳細は、『Oracle Call Interface プログラマーズ・ガイド』を参照してください。

GLOBALLY GLOBALLY を指定すると、グローバル・ユーザーがデータベースへ接続するときに、LDAP ディレクトリによってネームスペースを初期化できます。

セッション確立後、設定した PL/SQL パッケージのみがネームスペースのすべての属性に書込みを行うコマンドを発行することができます。

> **参照：**
>
> - グローバルに初期化されるコンテキストの確立の詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。
> - LDAP ディレクトリを介したデータベースへの接続の詳細は、『Oracle Internet Directory 管理者ガイド』を参照してください。

ACCESSED GLOBALLY

この句を使用すると、*namespace* に設定したすべてのアプリケーション・コンテキストによるインスタンス全体へのアクセスを許可できます。この設定によって、複数のセッションがアプリケーション属性を共有できます。

例

アプリケーション・コンテキストの作成例： この例では、hr アプリケーションを検証および保護する PL/SQL パッケージ empno_ctx を使用します。このアプリケーション・コンテキストを作成する例については、『Oracle Database セキュリティ・ガイド』を参照してください。次の文は、コンテキスト・ネームスペース hr_context を作成し、empno_ctx パッケージに関連付けます。

```
CREATE CONTEXT hr_context USING empno_ctx;
```

SYS_CONTEXT ファンクションを使用して、このコンテキストに基づいて、データ・アクセスを制御できます。たとえば、empno_ctx パッケージで、特定の従業員識別子として、属性 employee_id が定義されているとします。次のコマンドを実行し、employee_id の値に基づいてアクセスを制限するビューを作成して、employees 実表を保護できます。

```
CREATE VIEW hr_org_secure_view AS
   SELECT * FROM employees
   WHERE employee_id = SYS_CONTEXT('empno_ctx', 'employee_id');
```

> **参照：** 「SYS_CONTEXT」（5-184 ページ）

CREATE CONTROLFILE

注意： この文を使用する前に、データベース内のすべてのファイルの全体バックアップを実行することをお薦めします。詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

用途

CREATE CONTROLFILE 文は、ごく限られた状況でのみ使用します。データベースによって使用されているすべての制御ファイルが失われ、かつバックアップ制御ファイルが存在しない場合に、この文を使用して制御ファイルを再作成します。この文を使用して、REDO ログ・ファイル・グループ、REDO ログ・ファイル・メンバー、アーカイブ REDO ログ・ファイル、データ・ファイル、またはデータベースを同時にマウントおよびオープンするインスタンスの最大数を変更することもできます。

データベースの名前を変更するには、CREATE CONTROLFILE 文ではなく、DBNEWID ユーティリティを使用することをお薦めします。データベース名の変更後に OPEN RESETLOGS 操作が必要ないため、DBNEWID の方が適しています。

参照：

- DBNEWID ユーティリティの詳細は、『Oracle Database ユーティリティ』を参照してください。
- 既存のデータベース制御ファイルに基づくスクリプトの作成の詳細は、10-32 ページの「ALTER DATABASE」の「BACKUP CONTROLFILE 句」を参照してください。

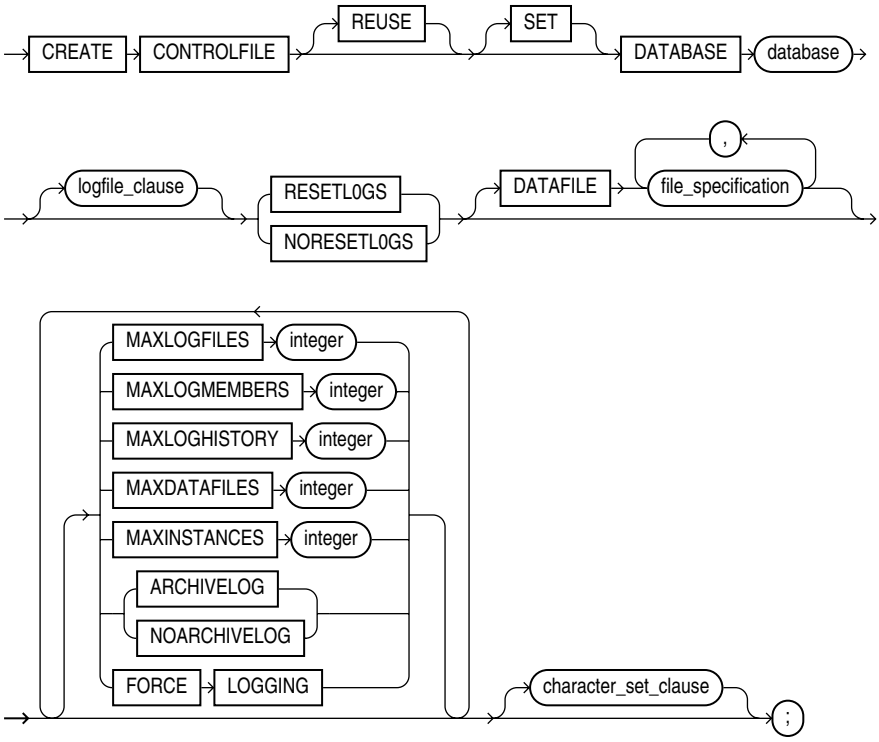
前提条件

制御ファイルを作成する場合は、SYSDBA システム権限が必要です。

データベースをマウントしているインスタンスがあってはいけません。制御ファイルが正常に作成された後、CLUSTER_DATABASE パラメータで指定したモードでデータベースがマウントされます。続いて、DBA は、データベースをオープンする前にメディア・リカバリを行う必要があります。データベースを Oracle Real Application Clusters（RAC）と併用している場合、次のインスタンスが起動する前に、データベースを停止して SHARED モード（CLUSTER_DATABASE 初期化パラメータの値に TRUE を設定）で再マウントする必要があります。

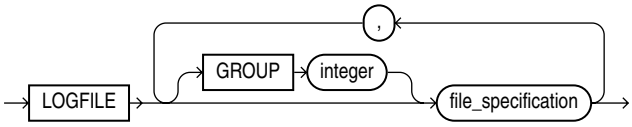
構文

***create_controlfile*::=**



（8-44 ページの storage_clause::= を参照）

logfile_clause::=



（8-26 ページの file_specification::= を参照）

character_set_clause::=

セマンティクス

CREATE CONTROLFILE 文を発行すると、この文に指定した情報に基づいて、新しい制御ファイルが作成されます。制御ファイルは、CONTROL_FILES 初期化パラメータで指定した場所に格納されます。このパラメータに値を指定していない場合、Oracle Managed Files の制御ファイルは、次のいずれか（優先度の高い順に示します）のとおり、制御ファイルのデフォルトの格納先に作成されます。

1. 1 つ以上の制御ファイルが DB_CREATE_ONLINE_LOG_DEST_*n* 初期化パラメータに指定されている場所に作成されます。最初のディレクトリのファイルは、プライマリ制御ファイルです。DB_CREATE_ONLINE_LOG_DEST_*n* を指定している場合、DB_CREATE_FILE_DEST または DB_RECOVERY_FILE_DEST（フラッシュ・リカバリ領域）には制御ファイルが作成されません。

2. DB_CREATE_ONLINE_LOG_DEST_*n* に値を指定しておらず、DB_CREATE_FILE_DEST および DB_RECOVERY_FILE_DEST の両方に値を指定している場合、それぞれの場所に 1 つの制御ファイルが作成されます。DB_CREATE_FILE_DEST に指定している場所は、プライマリ制御ファイルです。
3. DB_CREATE_FILE_DEST にのみ値を指定している場合、その場所に 1 つの制御ファイルが作成されます。
4. DB_RECOVERY_FILE_DEST にのみ値を指定している場合、その場所に 1 つの制御ファイルが作成されます。

これらのパラメータのいずれにも値を設定していない場合、データベースが稼働しているオペレーティング・システムのデフォルトの場所に制御ファイルが作成されます。この制御ファイルは Oracle Managed Files ではありません。

句を指定しない場合、Oracle Database は、以前の制御ファイルに対する値ではなく、デフォルト値を使用します。制御ファイルが正常に作成された後、Oracle Database は、初期化パラメータ CLUSTER_DATABASE で指定したモードでデータベースをマウントします。パラメータが設定されていないときのデフォルト値は FALSE で、その場合は EXCLUSIVE モードでデータベースがマウントされます。その後、インスタンスを停止し、データベースのすべてのファイルの全体バックアップを取ることをお薦めします。

> **参照：** 『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』

REUSE

REUSE を指定すると、初期化パラメータ CONTROL_FILES によって特定される既存の制御ファイルを再利用可能にできます。このとき、この初期化パラメータに現在格納されている情報は上書きされます。この句を指定しないと、既存の制御ファイルがある場合に、エラーが戻ります。

DATABASE 句

データベース名を指定します。このパラメータの値は、事前に CREATE DATABASE 文または CREATE CONTROLFILE 文で設定した既存のデータベース名である必要があります。

SET DATABASE 句

SET DATABASE を使用すると、データベース名を変更できます。データベース名には最大 8 バイトの名前を指定できます。

この句を指定する場合は、RESETLOGS も指定する必要があります。データベースの名前を変更し、既存のログ・ファイルを保持する場合は、この CREATE CONTROLFILE 文を発行した後、ALTER DATABASE RECOVER USING BACKUP CONTROLFILE 文を使用して、全データベースのリカバリを完了する必要があります。

logfile_clause

logfile_clause を使用すると、データベースの REDO ログ・ファイルを指定できます。すべての REDO ログ・ファイル・グループのすべてのメンバーを指定する必要があります。

オペレーティング・システムのファイル・システム内の標準 REDO ログ・ファイル、または自動ストレージ管理ディスク・グループの REDO ログ・ファイルを指定するには、*file_specification* の redo_log_file_spec 書式（8-26 ページの「file_specification」を参照）を使用します。*ASM_filename* の書式を使用する場合、*redo_log_file_spec* の *autoextend_clause* は指定できません。

この句に RESETLOGS を指定する場合、*ASM_filename* のいずれかのファイル作成形式を使用する必要があります。NORESETLOGS を指定する場合、*ASM_filename* のいずれかの参照書式を指定する必要があります。

> **参照：** 構文の様々な書式は、8-28 ページの「ASM_filename」を参照してください。自動ストレージ管理の使用方法の概要は、『Oracle Database ストレージ管理者ガイド』を参照してください。

GROUP *integer* ログ・ファイル・グループ番号を指定します。GROUP 値を指定した場合、データベースが前回オープンされたときの GROUP 値を基にして、この値が検証されます。

この句を指定しない場合、システムのデフォルト値を使用してログ・ファイルが作成されます。また、DB_CREATE_ONLINE_LOG_DEST_*n* 初期化パラメータまたは DB_CREATE_FILE_DEST 初期化パラメータのいずれかを設定した場合、および RESETLOGS を指定した場合は、DB_CREATE_ONLINE_LOG_DEST_*n* パラメータで指定したログ・ファイルのデフォルトの格納先に 2 つのログ・ファイルが作成されます。また、DB_CREATE_ONLINE_LOG_DEST_n を設定していない場合は、DB_CREATE_FILE_DEST パラメータで指定した格納先に作成されます。

> **参照：** この句の詳細は、8-26 ページの「file_specification」を参照してください。

RESETLOGS RESETLOGS を指定すると、Oracle Database に LOGFILE 句にリストされたファイルの内容を無視させることができます。LOGFILE 句に指定したファイルは、存在していなくてもかまいません。SET DATABASE 句を指定した場合、この句を指定する必要があります。

LOGFILE 句の各 *redo_log_file_spec* で、SIZE パラメータを指定する必要があります。データベースでは、スレッド 1 にすべてのオンライン REDO ログ・ファイル・グループを割り当てることによって、このスレッドを任意のインスタンスで共通に使用できるようにします。この句を使用した後は、RESETLOGS 句を指定した ALTER DATABASE 文を使用してデータベースをオープンする必要があります。

NORESETLOGS NORESETLOGS を指定すると、Oracle Database に、LOGFILE 句に指定したすべてのファイルを、前回データベースをオープンしたときの状態で使用させることができます。LOGFILE 句に指定したファイルは、存在する必要があります。また、バックアップからのリストアではなく、現行のオンライン REDO ログ・ファイルである必要があります。前回割り当てたスレッドに、REDO ログ・ファイル・グループが再度割り当てられ、そのスレッドが前回と同じく再度使用可能になります。

SET DATABASE 句を指定してデータベースの名前を変更した場合は、RESETLOGS を指定することはできません。詳細は、14-12 ページの「SET DATABASE 句」を参照してください。

DATAFILE 句

データベースのデータ・ファイルを指定します。すべてのデータ・ファイルを指定する必要があります。これらのファイルは既存のファイルである必要がありますが、メディア・リカバリを必要とするリストア・バックアップでもかまいません。

DATAFILE 句には、読取り専用の表領域のデータ・ファイルを含めないでください。これらのタイプのファイルは、後でデータベースに追加できます。また、この句には、一時データ・ファイル（一時ファイル）も含めないでください。

オペレーティング・システムのファイル・システム内の標準データ・ファイルと一時ファイル、または自動ストレージ管理ディスク・グループのファイルを指定するには、*file_specification* の datafile_tempfile_spec 書式（8-26 ページの「file_specification」を参照）を使用します。*ASM_filename* の書式を使用する場合、*ASM_filename* のいずれかの参照書式を使用する必要があります。構文の様々な書式は、8-28 ページの「ASM_filename」を参照してください。

> **参照：** 自動ストレージ管理の使用方法の概要は、『Oracle Database ストレージ管理者ガイド』を参照してください。

DATAFILE の制限事項： この DATAFILE 句では、*file_specification* の *autoextend_clause* は指定できません。

MAXLOGFILES 句

データベースに対して作成可能なオンライン REDO ログ・ファイル・グループの最大数を指定します。Oracle Database は、この値を基にして、制御ファイル内で REDO ログ・ファイル名に割り当てる領域の量を決定します。デフォルト値や最大値は、使用するオペレーティング・システムによって異なります。指定する値は、すべての REDO ログ・ファイル・グループの GROUP の最大値以上である必要があります。

MAXLOGMEMBERS 句

REDO ログ・ファイル・グループのメンバー（同一コピー）の最大数を指定します。Oracle Database は、この値を基にして、制御ファイル内で REDO ログ・ファイル名に割り当てる領域の量を決定します。最小値は 1 です。最大値およびデフォルト値は、使用するオペレーティング・システムによって異なります。

MAXLOGHISTORY 句

このパラメータは、Oracle Database を ARCHIVELOG モードで使用している場合にのみ有効です。データベースの自動メディア・リカバリに必要なアーカイブ REDO ログ・ファイル・グループの現在見積もられている最大数を指定します。この値を基にして、制御ファイル内でアーカイブ REDO ログ・ファイル名に割り当てられる領域が決定されます。

最小値は 0（ゼロ）です。デフォルト値は MAXINSTANCES 値の倍数で、使用するオペレーティング・システムによって異なります。最大値は、制御ファイルの最大サイズの制限のみを受けます。必要に応じてデータベースによって制御ファイルの該当セクションに引き続き領域が追加されるため、元の構成で十分でなくなった場合でも制御ファイルを再作成する必要はありません。その結果、このパラメータの実際の値は、指定した値を超える場合があります。

MAXDATAFILES 句

CREATE DATABASE または CREATE CONTROLFILE 実行時の、制御ファイルのデータ・ファイル・セクションの初期サイズを指定します。値が MAXDATAFILES より大きく、DB_FILES 以下のファイルを追加した場合、データ・ファイル・セクションにさらに多くのファイルを格納できるように、Oracle の制御ファイルが自動的に拡張されます。

インスタンスでアクセスできるデータ・ファイルの数は、初期化パラメータ DB_FILES の制限を受けます。

MAXINSTANCES 句

データベースを同時にマウントおよびオープンできるインスタンスの最大数を指定します。この値は、初期化パラメータ INSTANCES の値より優先されます。最小値は 1 です。最大値およびデフォルト値は、使用するオペレーティング・システムによって異なります。

ARCHIVELOG | NOARCHIVELOG

ARCHIVELOG を指定すると、REDO ログ・ファイルを再利用する前に、ファイルの内容をアーカイブできます。この句を指定した場合、インスタンスまたはシステム障害リカバリのみでなく、メディア・リカバリも実行できるようになります。

ARCHIVELOG 句および NOARCHIVELOG 句を指定しない場合、デフォルトで NOARCHIVELOG モードが選択されます。制御ファイルの作成後に、ALTER DATABASE 文を使用して、ARCHIVELOG モードと NOARCHIVELOG モードを切り替えることができます。

FORCE LOGGING

この句を使用すると、制御ファイルの作成後にデータベースを FORCE LOGGING モードにできます。データベースがこのモードで実行されている場合、一時表領域および一時セグメントへの変更以外のすべてのデータベース内の変更が記録されます。この設定は、各表領域で指定する NOLOGGING または FORCE LOGGING 設定、および各データベース・オブジェクトで指定する NOLOGGING 設定より優先され、これらの設定には影響されません。この句を指定しない場合、制御ファイルの作成後にデータベースは FORCE LOGGING モードになりません。

注意： FORCE LOGGING モードは、パフォーマンスに影響する場合があります。この設定の使用方法の詳細は、『Oracle Database 管理者ガイド』を参照してください。

---

character_set_clause

キャラクタ・セットを指定した場合、制御ファイルのキャラクタ・セット情報が再構成されます。データベースのメディア・リカバリが後で必要となる場合、データベースがオープンする前にこの情報が使用可能になり、リカバリ時に表領域名が正しく解析されます。この句は、デフォルト以外のキャラクタ・セットを使用している場合にのみ必要です。デフォルトはオペレーティング・システムによって異なります。現行のデータベース・キャラクタ・セットは、起動時に $ORACLE_HOME/log のアラート・ログに出力されます。

表領域のリカバリに Recovery Manager を使用して制御ファイルを再作成する場合、およびデータ・ディクショナリに格納されているキャラクタ・セットとは異なるキャラクタ・セットを指定した場合、表領域のリカバリは失敗します。ただし、データベースのオープン時、制御ファイルのキャラクタ・セットは、データ・ディクショナリの正しいキャラクタ・セットに更新されます。

この句ではデータベース・キャラクタ・セットを変更できません。

> **参照：** 表領域のリカバリの詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

例

制御ファイルの作成例： この文は、制御ファイルを再作成します。この文のデータベース demo は、WE8DEC キャラクタ・セットで作成されています。この例の *path* には、ご使用のシステムでの適切な Oracle Database ディレクトリへのパスを挿入してください。

```
STARTUP NOMOUNT

CREATE CONTROLFILE REUSE DATABASE "demo" NORESETLOGS NOARCHIVELOG
    MAXLOGFILES 32
    MAXLOGMEMBERS 2
    MAXDATAFILES 32
    MAXINSTANCES 1
    MAXLOGHISTORY 449
LOGFILE
  GROUP 1 '/path/oracle/dbs/t_log1.f'  SIZE 500K,
  GROUP 2 '/path/oracle/dbs/t_log2.f'  SIZE 500K
# STANDBY LOGFILE
DATAFILE
  '/path/oracle/dbs/t_db1.f',
  '/path/oracle/dbs/dbu19i.dbf',
  '/path/oracle/dbs/tbs_11.f',
  '/path/oracle/dbs/smundo.dbf',
  '/path/oracle/dbs/demo.dbf'
CHARACTER SET WE8DEC
;
```

CREATE DATABASE

注意： この文を実行すると、データベースの初期設定が行われ、指定ファイル内の現行のデータは消去されます。そのことを十分理解したうえで、この文を使用してください。

セキュリティの強化に関する注意： Oracle Database の今回のリリース以降では、デフォルトのデータベース・ユーザー・アカウントのセキュリティ機能が強化されています。今回のリリースのセキュリティ・チェックリストは、『Oracle Database セキュリティ・ガイド』を参照してください。このチェックリストに従ってデータベースを構成することをお薦めします。

用途

CREATE DATABASE 文を使用すると、汎用的なデータベースを作成できます。

この文を実行すると、データベースの初期使用に備えて、指定した既存のデータ・ファイル上のデータがすべて消去されます。したがって、既存のデータベースに対してこの文を実行した場合、データ・ファイル上のすべてのデータが失われます。

この文を指定した場合、データベースの作成後、データベースは排他モードまたはパラレル・モード（初期化パラメータ CLUSTER_DATABASE の値に依存する）でマウントおよびオープンされ、通常の用途に使用可能になります。その後、データベースの表領域を作成できます。

参照：

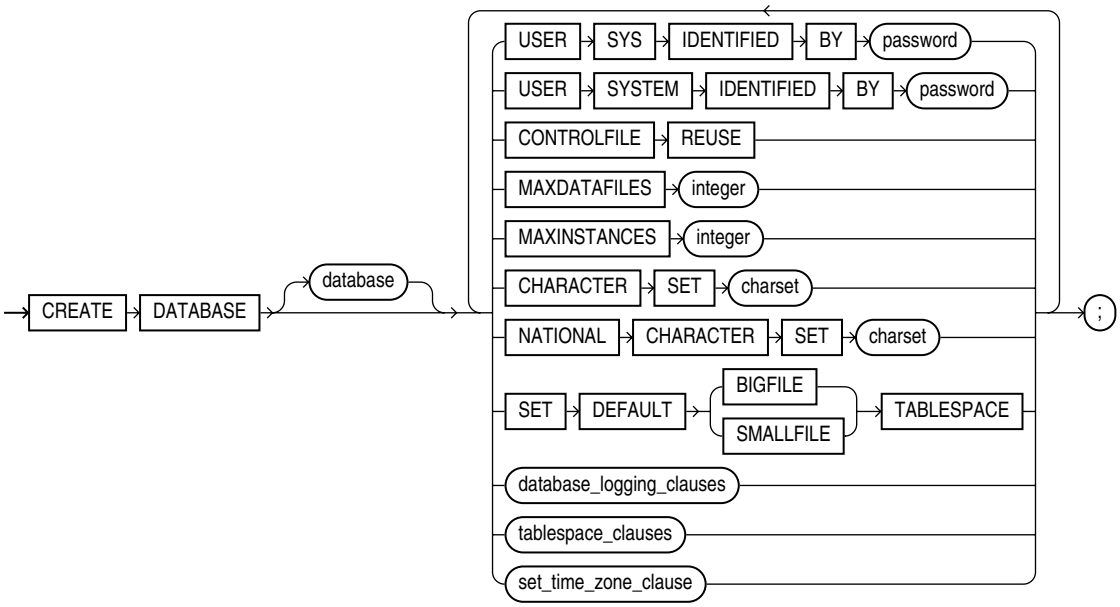
- データベースの変更の詳細は、10-9 ページの「ALTER DATABASE」を参照してください。
- Oracle Java Virtual Machine の作成の詳細は、『Oracle Database Java 開発者ガイド』を参照してください。
- 表領域の作成については、16-71 ページの「CREATE TABLESPACE」を参照してください。

前提条件

データベースを作成する場合は、SYSDBA システム権限が必要です。作成するデータベースの名前を持つ初期化パラメータ・ファイルが使用可能である必要があります。また、STARTUP NOMOUNT モードになっている必要があります。

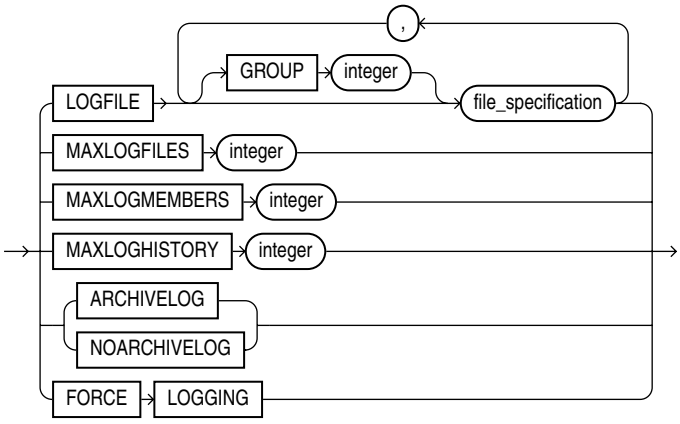
構文

***create_database*::=**



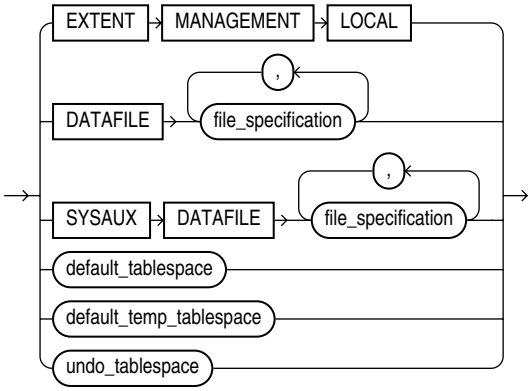
（14-17 ページの database_logging_clauses::=、14-18 ページの tablespace_clauses::=、14-19 ページの set_time_zone_clause::= を参照）

database_logging_clauses::=



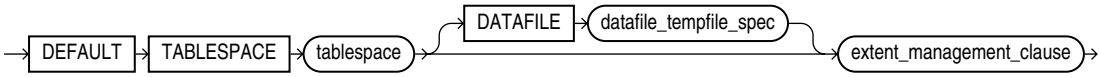
（8-26 ページの file_specification::= を参照）

tablespace_clauses::=

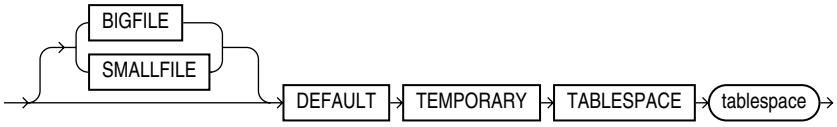


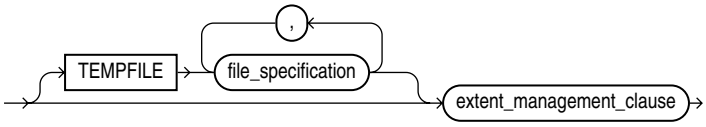
(8-26 ページの file_specification::=、14-18 ページの default_tablespace::=、
14-18 ページの default_temp_tablespace::=、14-18 ページの undo_tablespace::= を参照)

default_tablespace::=



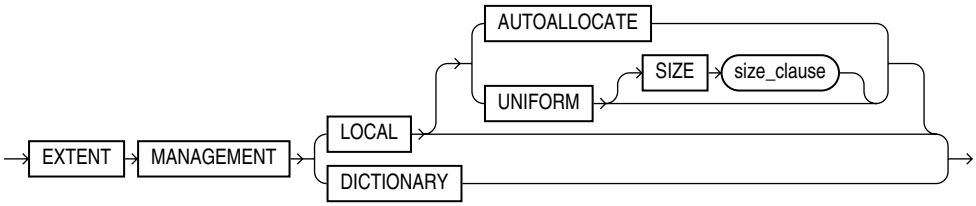
default_temp_tablespace::=





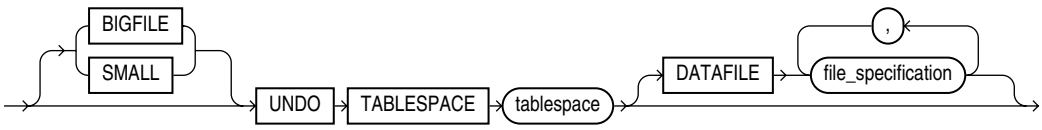
(8-26 ページの file_specification::= を参照)

extent_management_clause::=



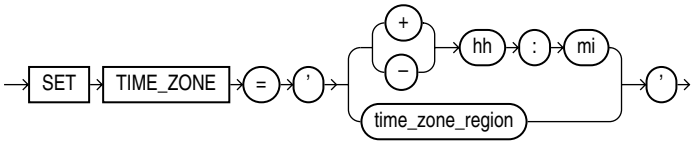
(8-42 ページの size_clause::= を参照)

undo_tablespace::=



(8-26 ページの file_specification::= を参照)

set_time_zone_clause::=



セマンティクス

database

作成するデータベースの名前を指定します。名前は、DB_NAME 初期化パラメータの値と一致する必要があります。名前の長さは最大 8 バイトです。データベース名には、ASCII 文字のみを指定できます。Oracle Database は、この名前を制御ファイルに書き込みます。後で、データベース名を明示的に指定する ALTER DATABASE 文を発行すると、制御ファイル内の名前に基づいて、そのデータベース名が検証されます。

データベース名は、大文字と小文字が区別されず、大文字の ASCII 文字で保存されます。データベース名を引用識別子として指定した場合、引用符は特に警告もなく無視されます。

注意： データベース名には、ヨーロッパやアジアのキャラクタ・セットの中の特殊文字は使用できません。たとえば、ウムラウト付きの文字は使用できません。

CREATE DATABASE 文でデータベース名を指定しない場合、DB_NAME 初期化パラメータで指定した名前が採用されます。DB_NAME 初期化パラメータは、データベースの初期化パラメータ・ファイルに設定する必要があります。そのパラメータの値とは異なる名前を指定した場合、データベースはエラーを戻します。データベース名のその他の規則については、2-102 ページの「スキーマ・オブジェクトのネーミングのガイドライン」を参照してください。

USER SYS ...、USER SYSTEM ...

これらの句を使用すると、SYS および SYSTEM ユーザーのパスワードを設定できます。これらの句は今回のリリースでは必須ではありません。ただし、いずれか一方の句を指定した場合は、もう一方の句も指定する必要があります。

これらの句を指定しない場合、Oracle Database は、デフォルト・パスワードとして、ユーザー SYS には change_on_install を、ユーザー SYSTEM には manager を作成します。これらのパスワードは、後で ALTER USER 文を使用して変更できます。また、データベース作成後に、ALTER USER を使用してパスワード管理属性を追加することもできます。

参照： 「ALTER USER」（13-5 ページ）

CONTROLFILE REUSE 句

CONTROLFILE REUSE を指定すると、初期化パラメータ CONTROL_FILES で特定される既存の制御ファイルを再利用可能にできます。このとき、この初期化パラメータに現在格納されている情報は上書きされます。通常、この句は、初めてデータベースを作成する際ではなく、データベースを再作成する際に使用します。データベースを初めて作成する場合、制御ファイルはデフォルトの格納先に作成されます。この場所は、いくつかの初期化パラメータの値によって決まります （14-11 ページの「CREATE CONTROLFILE」の「セマンティクス」を参照）。

制御ファイルを既存のファイルより大きくするためのパラメータ値もあわせて指定する場合、この句は使用できません。MAXLOGFILES、MAXLOGMEMBERS、MAXLOGHISTORY、MAXDATAFILES および MAXINSTANCES がこのようなパラメータです。

この句を指定しないと、CONTROL_FILES で指定した制御ファイルのいずれかがすでに存在する場合、エラーが戻ります。

MAXDATAFILES 句

CREATE DATABASE または CREATE CONTROLFILE 実行時の、制御ファイルのデータ・ファイル・セクションの初期サイズを指定します。値が MAXDATAFILES より大きく、DB_FILES 以下のファイルを追加した場合、データ・ファイル・セクションにさらに多くのファイルを格納できるように、Oracle Database の制御ファイルが自動的に拡張されます。

インスタンスでアクセスできるデータ・ファイルの数は、初期化パラメータ DB_FILES の制限を受けます。

MAXINSTANCES 句

データベースを同時にマウントおよびオープンできるインスタンスの最大数を指定します。この値は、初期化パラメータ INSTANCES の値より優先されます。最小値は 1 です。最大値は 1055 です。デフォルトは、使用するオペレーティング・システムによって異なります。

CHARACTER SET 句

データベースにデータを格納するときのキャラクタ・セットを指定します。サポートされているキャラクタ・セットおよびこのパラメータのデフォルト値は、使用するオペレーティング・システムによって異なります。

CHARACTER SET の制限事項： AL16UTF16 キャラクタ・セットは、データベース・キャラクタ・セットとして指定できません。

> **参照：** キャラクタ・セットの選択の詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

NATIONAL CHARACTER SET 句

データ型が NCHAR、NCLOB または NVARCHAR2 として定義された列にデータを格納する際に使用する各国語キャラクタ・セットを指定します。有効な値は、AL16UTF16 および UTF8 です。デフォルトは AL16UTF16 です。

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

database_logging_clauses

database_logging_clauses を使用すると、データベースの REDO ログ・ファイルの処理方法を指定できます。

LOGFILE 句

REDO ログ・ファイルとして使用する 1 つ以上のファイルを指定します。オペレーティング・システムのファイル・システム内の標準 REDO ログ・ファイル、または自動ストレージ管理ディスク・グループの REDO ログ・ファイルを作成するには、*file_specification* の *redo_log_file_spec* 書式を使用します。*ASM_filename* の書式を使用する場合、*redo_log_file_spec* の *autoextend_clause* は指定できません。

redo_log_file_spec 句には、1 つ以上の REDO ログ・ファイルのメンバー（コピー）を含む REDO ログ・ファイル・グループを指定します。CREATE DATABASE 文に指定したすべての REDO ログ・ファイルは、REDO ログのスレッド番号 1 に追加されます。

> **参照：** この句の詳細は、8-26 ページの「file_specification」を参照してください。

LOGFILE 句を指定しない場合、Oracle Managed Files のログ・ファイル・メンバーは、次のいずれか（優先度の高い順に示します）のとおり、デフォルトの格納先に作成されます。

- DB_CREATE_ONLINE_LOG_DEST_*n* を設定している場合、指定した各ディレクトリに、MAXLOGMEMBERS 初期化パラメータの値までログ・ファイル・メンバーが作成されます。

- DB_CREATE_ONLINE_LOG_DEST_*n* パラメータを設定しておらず、DB_CREATE_FILE_DEST および DB_RECOVERY_FILE_DEST 初期化パラメータの両方を設定している場合、それぞれの場所に 1 つの Oracle Managed Files のログ・ファイル・メンバーが作成されます。DB_CREATE_FILE_DEST に指定している場所のログ・ファイルが最初のメンバーです。
- DB_CREATE_FILE_DEST 初期化パラメータのみを指定している場合、その場所にログ・ファイル・メンバーが作成されます。
- DB_RECOVERY_FILE_DEST 初期化パラメータのみを指定している場合、その場所にログ・ファイル・メンバーが作成されます。

いずれの場合でも、パラメータ設定では、オペレーティング・システムのファイル名または作成書式自動ストレージ管理のファイル名を正確に指定する必要があります。

これらのパラメータのいずれにも値を設定していない場合、データベースが稼働しているオペレーティング・システムのデフォルトの場所にログ・ファイルが作成されます。このログ・ファイルは Oracle Managed Files ではありません。

GROUP *integer* REDO ログ・ファイル・グループの識別番号を指定します。*integer* の値は 1 ～ MAXLOGFILES パラメータの値の範囲です。データベースには、2 つ以上の REDO ログ・ファイル・グループが必要です。同一の GROUP 値を持つ REDO ログ・ファイル・グループは複数指定できません。このパラメータを指定しない場合、値が自動的に生成されます。REDO ログ・ファイル・グループの GROUP 値は、動的パフォーマンス・ビュー V$LOG で確認できます。

MAXLOGFILES 句

データベースに対して作成可能な REDO ログ・ファイル・グループの最大数を指定します。Oracle Database は、この値を基にして、制御ファイル内で REDO ログ・ファイル名に割り当てる領域の量を決定します。デフォルト値、最小値および最大値は、使用するオペレーティング・システムによって異なります。

MAXLOGMEMBERS 句

REDO ログ・ファイル・グループのメンバー（コピー）の最大数を指定します。Oracle Database は、この値を基にして、制御ファイル内で REDO ログ・ファイル名に割り当てる領域を決定します。最小値は 1 です。最大値およびデフォルト値は、使用するオペレーティング・システムによって異なります。

MAXLOGHISTORY 句

このパラメータが有用なのは、Oracle Real Application Clusters（RAC）で Oracle Database を ARCHIVELOG モードで使用している場合のみです。Oracle RAC の自動メディア・リカバリに使用するアーカイブ REDO ログ・ファイルの最大数を指定します。この値を基にして、制御ファイル内でアーカイブ REDO ログ・ファイル名に割り当てられる領域が決定されます。最小値は 0（ゼロ）です。デフォルト値は MAXINSTANCES 値の倍数で、使用するオペレーティング・システムによって異なります。最大値は、制御ファイルの最大サイズの制限のみを受けます。

ARCHIVELOG

ARCHIVELOG を指定すると、REDO ログ・ファイル・グループを再利用する前に、グループの内容をアーカイブできます。この句を指定すると、メディア・リカバリを実行できるようになります。

NOARCHIVELOG

NOARCHIVELOG を指定すると、REDO ログ・ファイル・グループを再利用する前に、グループの内容がアーカイブされません。この句を指定した場合、メディア・リカバリは実行できません。

デフォルトは NOARCHIVELOG モードです。データベースの作成後に、ALTER DATABASE 文を使用して、ARCHIVELOG モードと NOARCHIVELOG モードを切り替えることができます。

FORCE LOGGING

この句を使用すると、データベースを FORCE LOGGING モードにできます。一時表領域および一時セグメントへの変更以外のすべてのデータベース内の変更が記録されます。この設定は、各表領域で指定する NOLOGGING または FORCE LOGGING 設定、および各データベース・オブジェクトで指定する NOLOGGING 設定より優先され、これらの設定には影響されません。

FORCE LOGGING モードは、データベースのインスタンスで永続的です。データベースを停止し、再起動しても、データベースは FORCE LOGGING モードのままです。ただし、制御ファイルを再作成した場合は、CREATE CONTROLFILE 文で FORCE LOGGING を指定しないかぎり、データベースは FORCE LOGGING モードではなくなります。

注意： FORCE LOGGING モードは、パフォーマンスに影響する場合があります。この設定の使用方法の詳細は、『Oracle Database 管理者ガイド』を参照してください。

参照： 「CREATE CONTROLFILE」（14-10 ページ）

tablespace_clauses

tablespace_clauses を使用すると、SYSTEM および SYSAUX 表領域を構成したり、デフォルトの一時表領域と UNDO 表領域を指定することができます。

extent_management_clause

この句を使用すると、ローカル管理 SYSTEM 表領域を作成できます。この句を指定しない場合、SYSTEM 表領域はディクショナリ管理となります。

注意： ローカル管理 SYSTEM 表領域を作成すると、この表領域をディクショナリ管理に変更することはできません。このデータベース内に別のディクショナリ管理表領域を作成することもできません。

この句を指定した場合、ローカル管理の SYSTEM 表領域には一時セグメントを格納できないため、データベースにデフォルトの一時表領域が必要になります。

- EXTENT MANAGEMENT LOCAL を指定して、DATAFILE 句を指定しない場合は、*default_temp_tablespace* 句を省略できます。Oracle Database は、データ・ファイル・サイズが 10MB で、自動拡張を使用禁止にした状態で、TEMP という一時表領域を作成します。
- EXTENT MANAGEMENT LOCAL および DATAFILE の両方の句を指定する場合は、*default_temp_tablespace* を指定し、その表領域のデータ・ファイルを明示的に指定する必要があります。

インスタンスを自動 UNDO モードでオープンしている場合も、データベース UNDO 表領域には同様の要件があります。

- EXTENT MANAGEMENT LOCAL を指定して、DATAFILE 句を指定しない場合は、*undo_tablespace* 句を省略できます。Oracle Database は、SYS_UNDOTBS という UNDO 表領域を作成します。
- EXTENT MANAGEMENT LOCAL および DATAFILE の両方の句を指定する場合は、*undo_tablespace* を指定し、その表領域のデータ・ファイルを明示的に指定する必要があります。

参照： ローカル管理表領域およびディクショナリ管理表領域の詳細は、『Oracle Database 管理者ガイド』を参照してください。

SET DEFAULT TABLESPACE 句

この句を使用すると、以降に作成される表領域、および SYSTEM と SYSAUX 表領域のデフォルトの型を指定できます。以降に作成される表領域のデフォルトの型を bigfile または smallfile に設定するには、BIGFILE または SMALLFILE を指定します。

- **bigfile 表領域**には、1 つのデータ・ファイルまたは一時ファイルのみが含まれます。ファイルは、最大で約 40 億（2^{32}）のブロックを格納できます。1 つのデータ・ファイルまたは一時ファイルの最大サイズは、32K ブロックの表領域で 128TB、8K ブロックの表領域で 32TB です。
- **smallfile 表領域**は、Oracle の従来の表領域であり、1022 のデータ・ファイルまたは一時ファイルを含めることができます。それぞれのファイルは、最大で約 400 万（2^{22}）のブロックを格納できます。

この句を指定しない場合、デフォルトで smallfile 表領域が作成されます。

参照 :

- bigfile 表領域の詳細は、『Oracle Database 管理者ガイド』を参照してください。
- この構文の使用例は、10-41 ページの「表領域のデフォルト・タイプの設定例 :」を参照してください。

SYSAUX 句

Oracle Database は、すべてのデータベースに、SYSTEM と SYSAUX の両方の表領域を作成します。Oracle Managed Files を使用しておらず、SYSAUX 表領域の 1 つ以上のデータ・ファイルを指定する場合、この句を使用します。

DATAFILE 句を使用して SYSTEM 表領域の 1 つ以上のデータベースを指定した場合は、この句を指定する必要があります。Oracle Managed Files を使用している場合、この句を指定しないと、SYSAUX データ・ファイルが Oracle Managed Files に設定されたデフォルトの場所に作成されます。

Oracle Managed Files を使用可能にした場合に SYSAUX 句を省略すると、SYSAUX 表領域が、オンライン表領域、永続表領域およびローカル管理表領域として作成されます。この表領域は、100MB の 1 つのデータ・ファイルを持ち、ロギングおよび自動セグメント領域管理が有効になっています。

SYSAUX 表領域のデータ・ファイルを指定する構文は、ファイルを自動ストレージ管理を使用して格納するか、ファイル・システムまたは RAW デバイスに格納するかにかかわらず、CREATE TABLESPACE 文を使用して表領域の作成時にデータ・ファイルを指定する構文と同じです。

参照 :

- データベースのアップグレード時の SYSAUX 表領域の作成方法、および表領域のデータ・ファイルの指定方法は、16-71 ページの「CREATE TABLESPACE」を参照してください。
- SYSAUX 表領域の作成方法は、『Oracle Database 管理者ガイド』を参照してください。

default_tablespace

この句を指定すると、データベースのデフォルトの永続表領域を作成できます。Oracle Database は、smallfile 表領域を作成し、その後、この表領域に対して、別の永続表領域を指定していない、SYSTEM 以外のユーザーを割り当てます。この句を指定しない場合、SYSTEM 表領域が、SYSTEM 以外のユーザーのデフォルトの永続表領域になります。

DATAFILE 句および *extent_management_clause* は、CREATE TABLESPACE 文で同じセマンティクスを持ちます。これらの句の詳細は、16-75 ページの「DATAFILE | TEMPFILE 句」および 16-78 ページの「extent_management_clause」を参照してください。

default_temp_tablespace

この句を指定すると、データベースのデフォルトの一時表領域を作成できます。Oracle Database は、この一時表領域に対して、別の一時表領域を指定していないユーザーを割り当てます。この句を指定しないと、データベースがローカル管理の SYSTEM 表領域の作成時に、デフォルトの一時表領域を自動的に作成しない場合、SYSTEM 表領域がデフォルトの一時表領域になります。

デフォルトの一時表領域が bigfile か smallfile かを指定するには、BIGFILE または SMALLFILE を指定します。これらの句は、14-23 ページの「SET DEFAULT TABLESPACE 句」と同じセマンティクスを持ちます。

DB_CREATE_FILE_DEST 初期化パラメータを設定して Oracle Managed Files を使用可能にした場合、この句の TEMPFILE 句の部分はオプションです。このパラメータの値を指定していない場合は、TEMPFILE 句を指定する必要があります。BIGFILE を指定した場合、この句で指定できるのは 1 つの一時ファイルのみです。

デフォルトの一時表領域の一時ファイルを指定する構文は、ファイルを自動ストレージ管理を使用して格納するか、ファイル・システムまたは RAW デバイスに格納するかにかかわらず、CREATE TABLESPACE 文を使用して一時表領域の作成時に一時ファイルを指定する構文と同じです。

> **参照：** 一時ファイルの指定の詳細は、16-71 ページの「CREATE TABLESPACE」を参照してください。

注意： オペレーティング・システムによっては、一時ファイルのブロックが実際にアクセスされるまで、一時ファイル用の領域が割り当てられない場合があります。領域の割当ての遅延のため、一時ファイルの作成およびサイズ変更が速くなります。ただし、後で一時ファイルが使用されるときに、十分なディスク領域を使用可能にする必要があります。発生する可能性がある問題を回避するには、一時ファイルを作成またはサイズ変更する前に、ディスク領域が、新しく作成する一時ファイルまたはサイズ変更後の一時ファイルのサイズより大きいことを確認してください。ディスク領域に余裕を持たせることによって、関連のない操作が原因で増加が予想されるディスク使用量に対応できます。十分な領域があることを確認した後で、作成またはサイズ変更操作を実行してください。

デフォルトの一時表領域の制限事項： デフォルトの一時表領域には、次の制限事項があります。

- この句では、SYSTEM 表領域を指定できません。
- デフォルトの一時表領域は、標準的なブロック・サイズである必要があります。

extent_management_clause 句のセマンティクスは、CREATE DATABASE および CREATE TABLESPACE 文で同じです。この句の詳細は、16-78 ページの「CREATE TABLESPACE ...」の「extent_management_clause」を参照してください。

undo_tablespace

インスタンスを自動 UNDO モードでオープンした場合（UNDO_MANAGEMENT 初期化パラメータをデフォルトの AUTO に設定した場合）、*undo_tablespace* を指定して、UNDO データで使用する表領域を作成できます。自動 UNDO モードを使用することをお薦めします。ただし、UNDO 領域管理をロールバック・セグメントによって処理する場合、この句は指定しないでください。UNDO_TABLESPACE 初期化パラメータの値を設定した場合も、この句を省略できます。パラメータが設定されており、この句を指定した場合、*tablespace* はそのパラメータ値と同じである必要があります。

- UNDO 表領域を bigfile 表領域にする場合は、BIGFILE を指定します。**bigfile 表領域**には、1 つのデータ・ファイルのみが含まれます。このファイルの最大サイズは 8EB（8,000,000TB）です。

- UNDO 表領域を smallfile 表領域にする場合は、SMALLFILE を指定します。**smallfile 表領域**は、Oracle Database の従来の表領域です。この表領域には、最大で約 400 万（2^{22}）ブロックを含むことができます。
- DB_CREATE_FILE_DEST 初期化パラメータを設定して Oracle Managed Files を使用可能にした場合、この句の DATAFILE 句の部分はオプションです。このパラメータの値を指定していない場合は、DATAFILE 句を指定する必要があります。BIGFILE を指定した場合、この句で指定できるのは 1 つのデータ・ファイルのみです。

UNDO 表領域のデータ・ファイルを指定する構文は、ファイルを自動ストレージ管理を使用して格納するか、ファイル・システムまたは RAW デバイスに格納するかにかかわらず、CREATE TABLESPACE 文を使用して表領域の作成時にデータ・ファイルを指定する構文と同じです。

> **参照：** データファイルの指定の詳細は、16-71 ページの「CREATE TABLESPACE」を参照してください。

この句を指定すると、*tablespace* という名前の UNDO 表領域、および UNDO 表領域の一部として指定したデータ・ファイルが作成され、インスタンスの UNDO 表領域としてこの表領域が割り当てられます。Oracle Database は、この UNDO 表領域を使用して UNDO データを管理します。この句の DATAFILE 句は、14-25 ページの「DATAFILE 句」と同様に動作します。

初期化パラメータ・ファイル内の UNDO_TABLESPACE 初期化パラメータの値を指定している場合は、データベースをマウントする前に、この句で同じ名前を指定する必要があります。この名前が異なると、データベースのオープン時にエラーが戻されます。

この句を指定しないと、SYS_UNDOTBS という名前のデフォルトの smallfile の UNDO 表領域を持つデフォルトのデータベースが作成され、インスタンスの UNDO 表領域としてこのデフォルトの表領域が割り当てられます。この UNDO 表領域は、CREATE DATABASE 文で使用するデフォルトのファイルから、初期エクステントが 10MB のディスク領域を割り当てます。Oracle Database は、システムが生成したデータ・ファイルを処理します（14-25 ページの「DATAFILE 句」を参照）。Oracle Database が UNDO 表領域を作成できない場合、CREATE DATABASE 操作全体が失敗します。

> **参照：**
>
> - 自動 UNDO 管理および UNDO 表領域の詳細は、『Oracle Database 管理者ガイド』を参照してください。
> - データベースの作成後に UNDO 表領域を作成する方法は、16-71 ページの「CREATE TABLESPACE」を参照してください。

DATAFILE 句

データ・ファイルとして使用する 1 つ以上のファイルを指定します。ファイルは、すべて SYSTEM 表領域の一部となります。オペレーティング・システムのファイル・システム内の標準データ・ファイルと一時ファイル、または自動ストレージ管理ディスク・グループのファイルを作成するには、*file_specification* の datafile_tempfile_spec 書式を使用します。

注意： *undo_tablespace* 句の DATAFILE 句と同様に、この句はオプションです。したがって、あいまいさを回避するために、この句で SYSTEM 表領域のデータ・ファイルを指定する場合は、オプションの DATAFILE 句を含まない *undo_tablespace* 句の直後にこの句を指定しないでください。指定した場合、Oracle Database は DATAFILE 句を *undo_tablespace* 句の一部と認識します。

SYSTEM 表領域のデータ・ファイルを指定する構文は、ファイルを自動ストレージ管理を使用して格納するか、ファイル・システムまたは RAW デバイスに格納するかにかかわらず、CREATE TABLESPACE 文を使用して表領域の作成時にデータ・ファイルを指定する構文と同じです。

> **参照：** データ・ファイルの指定の詳細は、16-71 ページの「CREATE TABLESPACE」を参照してください。

自動 UNDO モードでデータベースを実行し、SYSTEM 表領域のデータ・ファイル名を指定した場合、すべての表領域に対してデータ・ファイルを生成するものとみなされます。Oracle Managed Files を使用する場合（DB_CREATE_FILE_DEST 初期化パラメータに値を設定している場合）、データ・ファイルは自動的に生成されます。ただし、Oracle Managed Files を使用しないでこの句を指定する場合は、*undo_tablespace* 句および *default_temp_tablespace* 句も指定する必要があります。

この句を指定しない場合、次のようになります。

- DB_CREATE_FILE_DEST 初期化パラメータを設定している場合、このパラメータで指定したデフォルトのファイル格納先に、サイズが 100MB でシステムが生成する名前を持つ、Oracle Managed Files のデータ・ファイルが作成されます。
- DB_CREATE_FILE_DEST 初期化パラメータを設定していない場合、1 つのデータ・ファイルが作成されます。そのファイル名およびサイズは、使用するオペレーティング・システムによって異なります。

> **参照：** 構文の詳細は、8-26 ページの「file_specification」を参照してください。

set_time_zone_clause

SET TIME_ZONE 句を使用すると、データベースのタイムゾーンを設定できます。次の 2 つ方法でタイムゾーンを設定します。

- UTC（協定世界時、以前のグリニッジ標準時）からの時差を指定。*hh:mm* の有効範囲は、-12:00 ～ +14:00 です。
- タイムゾーン地域を指定。有効な地域名を表示するには、V$TIMEZONE_NAMES 動的パフォーマンス・ビューの TZNAME 列を問い合せます。

注意： データベースのタイムゾーンを UTC（0:00）に設定することをお薦めします。これによって、タイムゾーンの変換が不要になるため、特にデータベース間のパフォーマンスを向上できます。

> **参照：** 動的パフォーマンス・ビューの詳細は、『Oracle Database リファレンス』を参照してください。

すべての TIMESTAMP WITH LOCAL TIME ZONE データは、ディスクに格納されるときにデータベースのタイムゾーンに正規化されます。SET TIME_ZONE 句を指定しない場合、サーバーのオペレーティング・システムのタイムゾーンが使用されます。オペレーティング・システムのタイムゾーンが Oracle Database で有効でない場合、データベースのタイムゾーンは、デフォルトで UTC になります。

例

データベースの作成例： 次の文は、すべての引数を指定してデータベースを作成します。

```
CREATE DATABASE sample
   CONTROLFILE REUSE
   LOGFILE
      GROUP 1 ('diskx:log1.log', 'disky:log1.log') SIZE 50K,
      GROUP 2 ('diskx:log2.log', 'disky:log2.log') SIZE 50K
   MAXLOGFILES 5
   MAXLOGHISTORY 100
   MAXDATAFILES 10
```

```
   MAXINSTANCES 2
   ARCHIVELOG
   CHARACTER SET AL32UTF8
   NATIONAL CHARACTER SET AL16UTF16
   DATAFILE
      'disk1:df1.dbf' AUTOEXTEND ON,
      'disk2:df2.dbf' AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED
   DEFAULT TEMPORARY TABLESPACE temp_ts
   UNDO TABLESPACE undo_ts
   SET TIME_ZONE = '+02:00';
```

この例では、初期化パラメータ・ファイルの DB_CREATE_FILE_DEST パラメータに値を設定して、Oracle Managed Files を使用可能にしている状態を想定しています。したがって、DEFAULT TEMPORARY TABLESPACE および UNDO TABLESPACE 句にファイル指定は必要ありません。

CREATE DATABASE LINK

用途

CREATE DATABASE LINK 文を使用すると、データベース・リンクを作成できます。**データベース・リンク**とは、他のデータベース上のオブジェクトにアクセスできる、データベース上のスキーマ・オブジェクトです。他のデータベースは、Oracle Database システムである必要はありません。ただし、Oracle 以外のシステムにアクセスする場合は、Oracle 異機種間サービスを使用する必要があります。

データベース・リンクを作成した後で、表名またはビュー名に *@dblink* を追加してそのリンクを SQL 文で利用して、他のデータベース上の表およびビューを参照できます。SELECT 文を使用して、他のデータベース上の表またはビューを問い合せることができます。INSERT 文、UPDATE 文、DELETE 文または LOCK TABLE 文を使用してもリモート表およびビューにアクセスできます。

参照 :

- PL/SQL ファンクション、プロシージャ、パッケージおよびデータ型を使用してリモート表またはビューへアクセスする方法については、『Oracle Database アドバンスト・アプリケーション開発者ガイド』を参照してください。
- 分散データベース・システムについては、『Oracle Database 管理者ガイド』を参照してください。
- ALL_DB_LINKS、DBA_DB_LINKS および USER_DB_LINKS データ・ディクショナリ・ビューの既存のデータベース・リンクの詳細、および V$DBLINK 動的パフォーマンス・ビューを使用して既存のリンクのパフォーマンスを監視する方法については、『Oracle Database リファレンス』を参照してください。
- 既存のデータベース・リンクを削除する方法については、17-36 ページの「DROP DATABASE LINK」を参照してください。
- DML 操作でリンクを使用する方法については、18-53 ページの「INSERT」、19-62 ページの「UPDATE」、17-23 ページの「DELETE」および 18-69 ページの「LOCK TABLE」を参照してください。

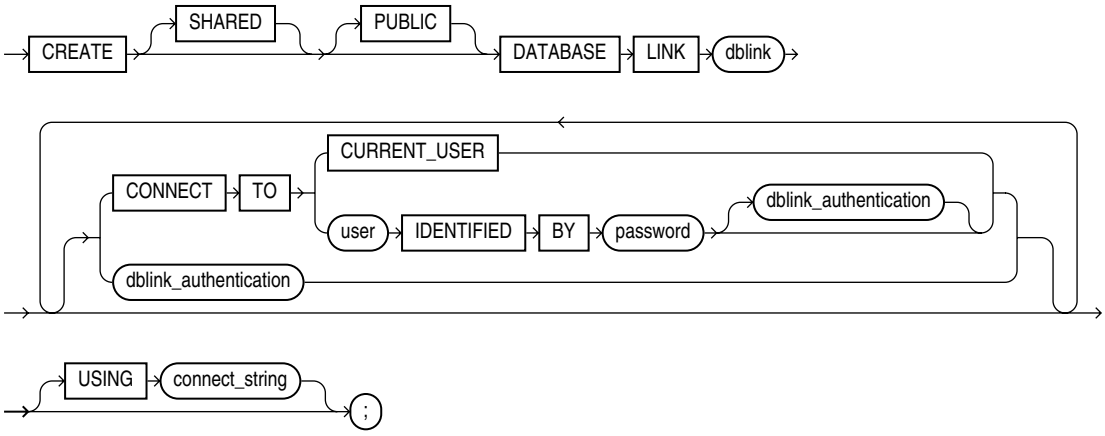
前提条件

プライベート・データベース・リンクを作成する場合、CREATE DATABASE LINK システム権限が必要です。パブリック・データベース・リンクを作成する場合、CREATE PUBLIC DATABASE LINK システム権限が必要です。また、リモートの Oracle Database に対する CREATE SESSION 権限が必要です。

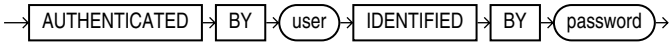
なお、ローカルとリモートの両方の Oracle Database に、Oracle Net をインストールしておく必要があります。

構文

***create_database_link*::=**



dblink_authentication::=



セマンティクス

PUBLIC

PUBLIC を指定すると、すべてのユーザーが参照可能なパブリック・データベース・リンクを作成できます。この句を指定しない場合、データベース・リンクはプライベートとなり、作成したユーザー専用になります。

リモート・データベース上のアクセス可能なデータは、リモート・データベースへの接続時にデータベース・リンクで使用される識別によって異なります。

- CONNECT TO *user* IDENTIFIED BY *password* を指定した場合、データベース・リンクは、指定されたユーザーとパスワードを使用して接続します。
- CONNECT TO CURRENT_USER を指定した場合、データベース・リンクは、そのリンクが使用される有効範囲に基づいて有効なユーザーを使用して接続します。
- いずれの句も指定しない場合、データベース・リンクは、ローカル接続されたユーザーとしてリモート・データベースに接続します。

参照 : 「パブリック・データベース・リンクの定義例 :」(14-31 ページ)

SHARED

SHARED を指定すると、ソース・データベースからターゲット・データベースへの 1 つのネットワーク接続を使用する複数のセッションで共有可能な 1 つのデータベース・リンクを作成できます。共有サーバー構成では、共有データベース・リンクによって、リモート・データベースへの接続数が多くなりすぎるのを防ぐことができます。通常、共有リンクはパブリック・データベース・リンクでもあります。ただし、多くのクライアントが同じローカル・スキーマにアクセスして同じプライベート・データベース・リンクを使用する場合は、共有プライベート・データベース・リンクが役立つ場合もあります。

共有データベース・リンクでは、ソース・データベースの複数のセッションでターゲット・データベースへの同じ接続が共有されます。ターゲット・データベースでセッションが確立されると、ソース・データベースの別のセッションで接続を使用できるように、そのセッションは接続から関連付けを解除されます。認可されていないセッションがデータベース・リンクを使用して接続できないように、SHARED を指定するときは、データベース・リンクの使用を認可されたユーザーに対して *dblink_authentication* 句も指定する必要があります。

参照： 共有データベース・リンクの詳細は、『Oracle Database 管理者ガイド』を参照してください。

dblink

データベース・リンクの完全な名前または名前の一部を指定します。データベース名のみを指定した場合、ローカル・データベースのデータベース・ドメインが暗黙的に追加されます。

dblink には、ASCII 文字のみを使用してください。マルチバイト文字はサポートされていません。データベース・リンク名は、大文字と小文字が区別されず、大文字の ASCII 文字で保存されます。データベース名を引用識別子として指定した場合、引用符は特に警告もなく無視されます。

GLOBAL_NAMES 初期化パラメータの値が TRUE の場合、データベース・リンクは、接続先のデータベースと同じ名前を持つ必要があります。GLOBAL_NAMES の値が FALSE で、データベースのグローバル名を変更した場合、グローバル名を指定できます。

Oracle RAC 構成の 1 つのセッションまたは 1 つのインスタンスでオープンできるデータベース・リンクの最大数は、OPEN_LINKS および OPEN_LINKS_PER_INSTANCE 初期化パラメータの値で指定します。

データベース・リンク作成の制限事項： 他のユーザーのスキーマ内にはデータベース・リンクを作成できません。また、*dblink* はスキーマ名を付けて指定できません（データベース・リンク名にはピリオドを指定できるため、ralph.linktosales のような名前を付けた場合、スキーマ ralph の linktosales という名前のデータベース・リンクと解析されるのではなく、名前全体が自分のスキーマにあるデータベース・リンク名と解析されます）。

参照：

- データベース・リンクのネーミングのガイドラインについては、2-104 ページの「リモート・データベース内のオブジェクトの参照」を参照してください。
- GLOBAL_NAMES、OPEN_LINKS および OPEN_LINKS_PER_INSTANCE 初期化パラメータについては、『Oracle Database リファレンス』を参照してください。
- データベースのグローバル名の変更については、10-38 ページの「ALTER DATABASE」の「RENAME GLOBAL_NAME 句」を参照してください。

CONNECT TO 句

CONNECT TO 句を使用すると、リモート・データベースへの接続に使用するユーザーおよび資格証明がある場合は、それらを指定できます。

CURRENT_USER 句

CURRENT_USER を指定すると、**現行のユーザーのデータベース・リンク**を作成できます。現行のユーザーは、リモート・データベースに有効なアカウントを持つグローバル・ユーザーである必要があります。

データベース・リンクがストアド・オブジェクト内からではなく直接使用される場合、現行のユーザーは接続ユーザーと同じです。

データベース・リンクを開始するストアド・オブジェクト（プロシージャ、ビュー、トリガーなど）を実行する場合、CURRENT_USER は、ストアド・オブジェクトを所有するユーザーの名前であり、オブジェクトをコールしたユーザーの名前ではありません。たとえば、データベース・リンクが（scott によって作成された）プロシージャ scott.p 内にあり、ユーザー jane がプロシージャ scott.p をコールした場合、現行のユーザーは scott になります。

ただし、ストアド・オブジェクトが実行者権限ファンクション、プロシージャまたはパッケージである場合、実行者認可 ID はリモート・ユーザーとしての接続に使用されます。たとえば、権限を持つデータベース・リンクがプロシージャ scott.p（scott によって作成された実行者権限プロシージャ）内にあり、ユーザー Jane がプロシージャ scott.p をコールした場合、CURRENT_USER は jane であり、プロシージャは、Jane の権限で実行されます。

参照：

- 実行者権限ファンクションの詳細は、14-48 ページの「CREATE FUNCTION」を参照してください。
- 「CURRENT_USER データベース・リンクの定義例 :」（14-32 ページ）

user IDENTIFIED BY *password*

固定ユーザー・データベース・リンクを使用して、リモート・データベースに接続するためのユーザー名およびパスワードを指定できます。この句を指定しない場合、データベース・リンクでは、データベースに接続している各ユーザーのユーザー名およびパスワードが使用されます。これを**接続ユーザー・データベース・リンク**といいます。

参照： 「固定ユーザー・データベース・リンクの定義例 :」（14-32 ページ）

dblink_authentication

共有データベース・リンクを作成する場合（SHARED 句を指定した場合）にのみ、この句を指定できます。ターゲット・インスタンスのユーザー名およびパスワードを指定します。この句は、リモート・サーバーに対してユーザーを認証するもので、セキュリティ上必要です。指定するユーザー名およびパスワードは、リモート・インスタンスで有効なユーザー名およびパスワードである必要があります。ユーザー名およびパスワードは、認証用としてのみ使用されます。このユーザーを対象とした認証以外の操作はありません。

USING '*connect string*'

リモート・データベースのサービス名を指定します。データベース名のみを指定した場合、接続文字列にデータベース・ドメインが暗黙的に追加され、完全なサービス名が作成されます。したがって、リモート・データベースのデータベース・ドメインが現行のデータベース・ドメインと異なる場合は、完全なサービス名を指定する必要があります。

参照： リモート・データベースの指定の詳細は、『Oracle Database 管理者ガイド』を参照してください。

例

次の例では、local という名前のデータベースと remote という名前の 2 つのデータベースを使用することを想定しています。この例では、Oracle Database ドメインを使用します。ご使用のドメインとは異なります。

パブリック・データベース・リンクの定義例： 次の例では、remote という共有パブリック・データベース・リンクを定義します。このデータベース・リンクは、サービス名 remote で指定されたデータベースと対応しています。

```
CREATE PUBLIC DATABASE LINK remote
   USING 'remote';
```

このデータベース・リンクによって、local データベース上のユーザー hr がリモート・データベース上の表を更新できます（hr に適切な権限があることを想定しています）。

```
UPDATE employees@remote
   SET salary=salary*1.1
   WHERE last_name = 'Baer';
```

固定ユーザー・データベース・リンクの定義例： 次の例では、リモート・データベース上のユーザー hr が、local データベース上の hr スキーマに、local という名前の固定ユーザー・データベース・リンクを定義します。

```
CREATE DATABASE LINK local
   CONNECT TO hr IDENTIFIED BY password
   USING 'local';
```

このデータベース・リンクが作成されると、hr は、次のように local データベース上のスキーマ hr の表を問い合せることができます。

```
SELECT * FROM employees@local;
```

ユーザー hr は、次の DML 文を使用して、local データベース上のデータを変更することもできます。

```
INSERT INTO employees@local
   (employee_id, last_name, email, hire_date, job_id)
   VALUES (999, 'Claus', 'sclaus@example.com', SYSDATE, 'SH_CLERK');

UPDATE jobs@local SET min_salary = 3000
   WHERE job_id = 'SH_CLERK';

DELETE FROM employees@local
   WHERE employee_id = 999;
```

この固定データベース・リンクを使用すると、remote データベース上のユーザー hr は、同じデータベース上の他のユーザーが所有する表にもアクセスできます。この文は、ユーザー hr が oe.customers 表に対する SELECT 権限を持っていることを想定しています。この文では、local データベースのユーザー hr に接続した後で、次のように oe.customers 表への問合せが行われます。

```
SELECT * FROM oe.customers@local;
```

CURRENT_USER データベース・リンクの定義例： 次の文は、リンク名としてサービス名全体を使用して、remote データベースに対する現行のユーザーのデータベース・リンクを定義します。

```
CREATE DATABASE LINK remote.us.example.com
   CONNECT TO CURRENT_USER
   USING 'remote';
```

この文を発行するユーザーは、LDAP ディレクトリ・サービスに登録されたグローバル・ユーザーである必要があります。

特定の表が remote データベース上にあることを示さないように、シノニムを作成できます。次の文によって、これ以降に emp_table を参照すると、リモート・データベース上の hr が所有する employees 表にアクセスします。

```
CREATE SYNONYM emp_table
   FOR oe.employees@remote.us.example.com;
```

CREATE DIMENSION

用途

CREATE DIMENSION 文を使用すると、**ディメンション**を作成できます。ディメンションは、列集合の組の親子関係を定義します。列集合の列は、すべて同じ表から得られたものである必要があります。ただし、1 つの列集合（**レベル**）の列は、別の集合の列とは異なる表から得ることができます。オプティマイザは、マテリアライズド・ビューとの関係を使用して**クエリー・リライト**を行います。SQL アクセス・アドバイザは、特定のマテリアライズド・ビューの作成の推奨にこの関係を使用します。

注意： Oracle Database は、ディメンションの作成中に宣言する関係の妥当性チェックを自動的には行いません。*hierarchy_clause* および CREATE DIMENSION の *dimension_join_clause* で指定する関係の妥当性チェックを行うには、DBMS_OLAP.VALIDATE_DIMENSION プロシージャを実行する必要があります。

参照：

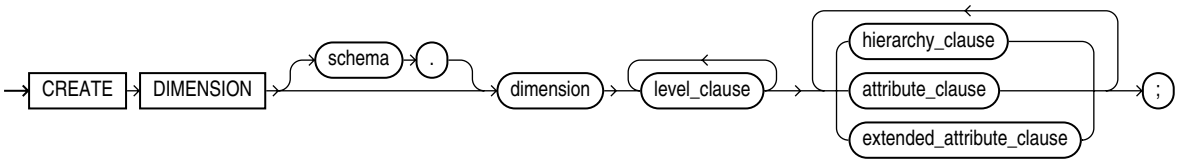
- マテリアライズド・ビューの詳細は、15-4 ページの「CREATE MATERIALIZED VIEW」を参照してください。
- クエリー・リライト、オプティマイザおよび SQL アクセス・アドバイザの詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。

前提条件

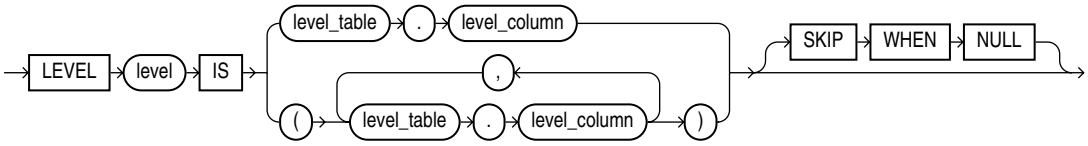
自分のスキーマ内にディメンションを作成する場合は、CREATE DIMENSION システム権限が必要です。他のユーザーのスキーマ内にディメンションを作成する場合は、CREATE ANY DIMENSION システム権限が必要です。どちらの場合も、ディメンションで参照されるオブジェクトに対して、SELECT オブジェクト権限が必要です。

構文

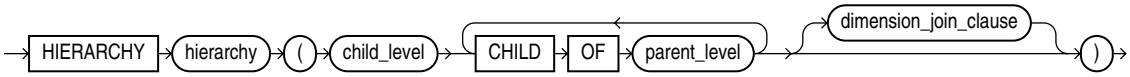
***create_dimension*::=**



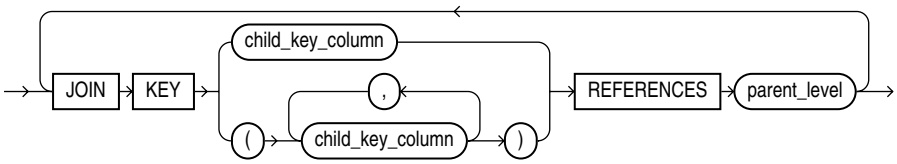
level_clause::=



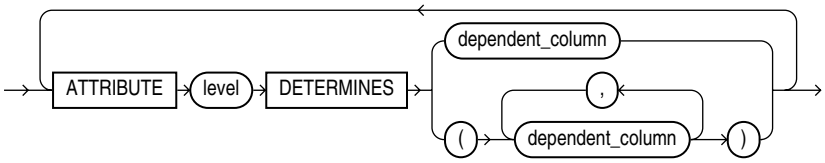
hierarchy_clause::=



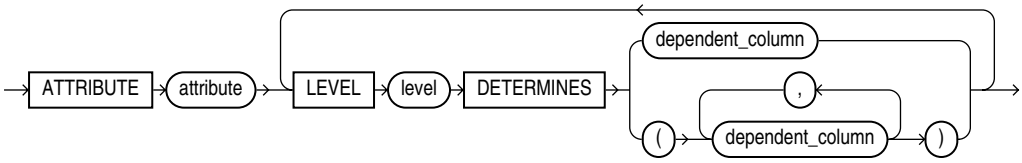
dimension_join_clause::=



attribute_clause::=



extended_attribute_clause::=



セマンティクス

schema

ディメンションを作成するスキーマを指定します。*schema* を指定しない場合、自分のスキーマ内にディメンションが作成されます。

dimension

ディメンション名を指定します。名前は、スキーマ内で一意である必要があります。

level_clause

ディメンションのレベルを指定します。レベルは、ディメンション階層および属性を定義します。

level レベル名を指定します。

level_table . level_column レベル内の列を指定します。最大 32 列を指定できます。この句で指定する表は、すでに存在している必要があります。

SKIP WHEN NULL この句を指定すると、指定したレベルが NULL の場合、そのレベルはスキップされます。この句を使用すると、指定したレベルをスキップする代替パスによって、親子関係の階層のつながりを維持できます。14-35 ページの「hierarchy_clause」を参照してください。

ディメンション・レベル列の制限事項： ディメンション・レベル列には、次の制限事項があります。

- レベルの列は、すべて同じ表から得られたものである必要があります。
- 異なるレベルの列が異なる表から得られる場合、*dimension_join_clause* を指定する必要があります。
- 指定する列の集合は、このレベルに一意である必要があります。
- 指定する列は、他のディメンションでは指定できません。
- レベルが SKIP WHEN NULL で指定されていないかぎり、各 level_column は、NULL 以外である必要があります。NULL 以外の列に NOT NULL 制約を指定する必要はありません。SKIP WHEN NULL を指定する列に、NOT NULL 制約を指定することはできません。

hierarchy_clause

ディメンションのレベルの線形階層を定義します。各階層が、ディメンションのレベル間で親子関係の連鎖を形成します。ディメンションの階層は、互いに依存していません。階層は、共通の列を持つことができます。

ディメンションの各レベルは、句の中で最高 1 回指定され、*level_clause* で名前を付けておく必要があります。

hierarchy 階層名を指定します。この名前は、ディメンションで一意である必要があります。

child_level 親レベルと n:1 の関係を持つレベルの名前を指定します。*child_level* の *level_columns* は NULL 以外である必要があります。各 *child_level* 値は、*parent_level* という名前の次の値を一意に定義します。

子 *level_table* が親 *level_table* と異なる場合、*dimension_join_clause* でそれらの結合関係を指定する必要があります。

parent_level レベル名を指定します。

dimension_join_clause

dimension_join_clause を使用すると、複数の表に列が含まれるディメンションに内部等価結合関係を指定できます。この句は、階層で指定されたすべての列が同じ表にあるとはかぎらない場合にのみ指定する必要があり、このときのみ指定できます。

child_key_column

親レベルの列と結合互換性のある 1 つ以上の列を指定します。

スキーマおよび各 *child_column* の表を指定しない場合、*hierarchy_clause* の CHILD OF 関係からスキーマおよび表が判断されます。*child_key_column* のスキーマおよび列を指定する場合は、*hierarchy_clause* の *parent_level* の子のスキーマおよび列の表と一致している必要があります。

parent_level

レベル名を指定します。

ディメンションの結合の制限事項： ディメンションの結合には、次の制限事項があります。

- 同じ階層の既存のレベルの組に対して、1 つの *dimension_join_clause* のみを指定できます。
- *child_key_columns* は NULL 以外であり、親キーが一意で NULL 以外である必要があります。条件を適用するために制約を定義する必要はありません。ただし、条件を満たさない場合、問合せが不適切な結果を戻すことがあります。

- 各子キーは、*parent_level* 表のキーと結合する必要があります。
- 自己結合は使用できません。*child_key_columns* を、*parent_level* と同じ表に置くことはできません。
- 子キー列は、すべて同じ表から得られたものである必要があります。
- 子キー列数は、*parent_level* の列数と一致し、列は結合可能である必要があります。
- 親レベルが複数の列で構成されている場合のみ、子キー列を指定します。

attribute_clause

attribute_clause を使用すると、階層レベルによって一意に定義されている列を指定できます。*level* の列は、*dependent_columns* と同じ表からすべて得る必要があります。*dependent_columns* は、*level_clause* で指定されている必要はありません。

たとえば、階層レベルが市、都道府県名、および国の場合、市は市長、都道府県名は知事、国は首相を決定します。

extended_attribute_clause

この句を使用すると、1 つ以上のレベルと列の関係に属性名を指定できます。この句で作成する属性の種類は、*attribute_clause* を使用して作成される属性の種類と同じです。唯一の違いは、属性にレベル名とは異なる名前を割り当てることができることです。

例

ディメンションの作成例： 次の文は、サンプル・スキーマ sh に customers_dim ディメンションを作成します。

```
CREATE DIMENSION customers_dim
   LEVEL customer   IS (customers.cust_id)
   LEVEL city       IS (customers.cust_city)
   LEVEL state      IS (customers.cust_state_province)
   LEVEL country    IS (countries.country_id)
   LEVEL subregion  IS (countries.country_subregion)
   LEVEL region     IS (countries.country_region)
   HIERARCHY geog_rollup (
      customer      CHILD OF
      city          CHILD OF
      state         CHILD OF
      country       CHILD OF
      subregion     CHILD OF
      region
   JOIN KEY (customers.country_id) REFERENCES country
   )
   ATTRIBUTE customer DETERMINES
   (cust_first_name, cust_last_name, cust_gender,
    cust_marital_status, cust_year_of_birth,
    cust_income_level, cust_credit_limit)
   ATTRIBUTE country DETERMINES (countries.country_name)
;
```

拡張属性を持つディメンションの作成例： また、次の例のように、*attribute_clause* のかわりに *extended_attribute_clause* を使用することもできます。

```
CREATE DIMENSION customers_dim
   LEVEL customer   IS (customers.cust_id)
   LEVEL city       IS (customers.cust_city)
   LEVEL state      IS (customers.cust_state_province)
   LEVEL country    IS (countries.country_id)
   LEVEL subregion  IS (countries.country_subregion)
```

```
   LEVEL region     IS (countries.country_region)
   HIERARCHY geog_rollup (
      customer      CHILD OF
      city          CHILD OF
      state         CHILD OF
      country       CHILD OF
      subregion     CHILD OF
      region
   JOIN KEY (customers.country_id) REFERENCES country
   )
   ATTRIBUTE customer_info LEVEL customer DETERMINES
   (cust_first_name, cust_last_name, cust_gender,
    cust_marital_status, cust_year_of_birth,
    cust_income_level, cust_credit_limit)
   ATTRIBUTE country DETERMINES (countries.country_name)
;
```

NULL 列値を持つディメンションの作成例： 次の例は、レベル列のいずれかが NULL の場合に、階層のつながりを維持する際のディメンションの作成方法を示しています。この例では、簡単に説明するために cust_marital_status 列を使用しています。この列は NOT NULL 列ではありません。この制約がある場合は、SKIP WHEN NULL 句を使用する前にこの制約を使用禁止にする必要があります。

```
CREATE DIMENSION customers_dim
   LEVEL customer IS (customers.cust_id)
   LEVEL status IS (customers.cust_marital_status) SKIP WHEN NULL
   LEVEL city IS (customers.cust_city)
   LEVEL state IS (customers.cust_state_province)
   LEVEL country IS (countries.country_id)
   LEVEL subregion IS (countries.country_subregion) SKIP WHEN NULL
   LEVEL region IS (countries.country_region)
   HIERARCHY geog_rollup (
      customer CHILD OF
      city CHILD OF
      state CHILD OF
      country CHILD OF
      subregion CHILD OF
      region
   JOIN KEY (customers.country_id) REFERENCES country
   )
   ATTRIBUTE customer DETERMINES
   (cust_first_name, cust_last_name, cust_gender,
    cust_marital_status, cust_year_of_birth,
    cust_income_level, cust_credit_limit)
   ATTRIBUTE country DETERMINES (countries.country_name)
;
```

CREATE DIRECTORY

用途

CREATE DIRECTORY 文を使用すると、ディレクトリ・オブジェクトを作成できます。ディレクトリ・オブジェクトは、外部バイナリ・ファイル LOB（BFILE）および外部表データが存在するサーバー・ファイル・システム上のディレクトリの別名を示します。PL/SQL コードおよび OCI コールで BFILE を参照する際、管理の汎用性のために、オペレーティング・システムのパス名をハード・エンコードせずにディレクトリ名を使用できます。

すべてのディレクトリは 1 つのネームスペースに作成されるため、個々のユーザーのスキーマで所有されるわけではありません。ディレクトリに対するオブジェクト権限を特定ユーザーに付与することによって、そのディレクトリ構造内に格納されている BFILE へのアクセスを制限できます。

参照：

- BFILE オブジェクトの詳細は、2-24 ページの「ラージ・オブジェクト（LOB）・データ型」を参照してください。
- オブジェクト権限の付与の詳細は、18-31 ページの「GRANT」を参照してください。
- 16-16 ページの「CREATE TABLE」の「external_table_clause::=」を参照してください。

前提条件

ディレクトリを作成する場合は、CREATE ANY DIRECTORY システム権限が必要です。

ディレクトリを作成する場合、そのディレクトリに対する READ オブジェクト権限および WRITE オブジェクト権限が自動的に付与され、他のユーザーおよびロールにこれらの権限を付与できます。DBA も、これらの権限を他のユーザーおよびロールに付与できます。

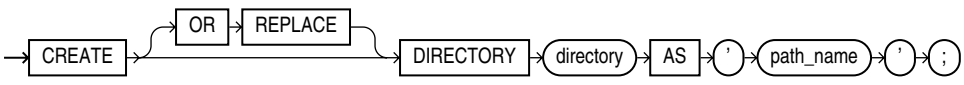
ディレクトリに対する WRITE 権限は、外部表との接続に便利です。これによって、権限受領者は、外部表のエージェントがディレクトリに書き込めるのがログ・ファイルなのか不良ファイルなのかを判断できます。

ファイルの記憶域用に、それに応じたオペレーティング・システムのディレクトリ、ASM ディスク・グループ、または ASM ディスク・グループ内のディレクトリを作成する必要もあります。システム管理者およびデータベース管理者は、このオペレーティング・システムのディレクトリに、Oracle Database プロセスに対する読取り権限および書込み権限が正しく設定されていることを確認する必要があります。

ディレクトリに対して付与される権限は、オペレーティング・システムのディレクトリ用に定義されたアクセス権限とは無関係に作成されるため、これらの権限は完全に対応しない場合があります。たとえば、サンプル・ユーザー hr に、ディレクトリ・オブジェクトに対する READ 権限が付与されていても、それに対応するオペレーティング・システムのディレクトリに Oracle Database プロセスに対する READ 権限が付与されていない場合は、エラーが発生します。

構文

***create_directory*::=**



セマンティクス

OR REPLACE

OR REPLACE を指定すると、既存のディレクトリ・データベース・オブジェクトを再作成できます。この句を指定した場合、既存のディレクトリに付与されているデータベース・オブジェクト権限を削除、再作成および再付与しなくても、そのディレクトリの定義を変更できます。

再定義したディレクトリに対する権限が付与されていたユーザーは、権限が再付与されなくてもそのディレクトリにアクセスできます。

> **参照：** データベースからのディレクトリの削除については、17-38 ページの「DROP DIRECTORY」を参照してください。

directory

作成するディレクトリ・オブジェクトの名前を指定します。*directory* の最大長は 30 バイトです。ディレクトリ・オブジェクトは、スキーマ名で修飾できません。

指定したディレクトリが実際に存在するかどうかは検証されません。このため、オペレーティング・システムに存在するディレクトリを指定してください。また、オペレーティング・システムでパス名の大 / 小文字が区別される場合は、正しい形式でディレクトリ名を指定する必要があります。パス名の終わりにスラッシュを指定する必要はありません。

ディレクトリ名で親ディレクトリを参照しないでください。たとえば、次の構文は有効です。

```
CREATE DIRECTORY mydir AS '/scratch/file_data';
```

ただし、次の構文は無効です。

```
CREATE DIRECTORY mydir AS '/scratch/data/../file_data';
```

path_name

ファイルが格納されているサーバー上のオペレーティング・システムのディレクトリのフルパス名を指定します。指定するフルパス名は、一重引用符で囲む必要があります。また、パス名の大 / 小文字は区別されます。

例

ディレクトリの作成例： 次の文は、サーバーのディレクトリを示す、ディレクトリのデータベース・オブジェクトを作成します。

```
CREATE DIRECTORY admin AS 'oracle/admin';
```

次の文は、オペレーティング・システムのディレクトリ /usr/bin に格納されている BFILE にアクセスできるように、ディレクトリのデータベース・オブジェクト bfile_dir を再定義します。

```
CREATE OR REPLACE DIRECTORY bfile_dir AS '/usr/bin/bfile_dir';
```

CREATE DISKGROUP

注意： この SQL 文は、自動ストレージ管理を使用しており、自動ストレージ管理インスタンスを起動している場合にのみ有効です。この文の発行は、通常のデータベース・インスタンスからではなく、自動ストレージ管理インスタンスから行う必要があります。自動ストレージ管理インスタンスの起動の詳細は、『Oracle Database ストレージ管理者ガイド』を参照してください。

用途

CREATE DISKGROUP 句を使用すると、ディスクのグループに名前を付け、そのグループを Oracle Database が管理するように指定できます。Oracle Database は、ディスク・グループを論理単位として管理し、I/O の均衡を保つために各ファイルを均等に分散します。また、ディスク・グループで使用可能なすべてのディスクにデータベース・ファイルを自動的に分散し、記憶域構成が変更されるたびに、自動的にストレージの均衡を再調整します。

この文を使用すると、ディスク・グループが作成され、1 つ以上のディスクがディスク・グループに割り当てられ、ディスク・グループの初めてのマウントが実行されます。後続のインスタンスで、自動ストレージ管理によって自動的にディスク・グループをマウントさせる場合は、初期化パラメータ・ファイルの ASM_DISKGROUPS 初期化パラメータの値に、そのディスク・グループ名を追加する必要があります。SPFILE を使用している場合、そのディスク・グループは自動的に初期化パラメータに追加されます。

参照：

- ディスク・グループの変更については、10-46 ページの「ALTER DISKGROUP」を参照してください。
- 自動ストレージ管理およびディスク・グループを使用してデータベース管理を簡略化する方法については、『Oracle Database ストレージ管理者ガイド』を参照してください。
- 初期化パラメータ・ファイルへのディスク・グループ名の追加の詳細は、ASM_DISKGROUPS を参照してください。
- 自動ストレージ管理操作を監視する方法については、V$ASM_OPERATION を参照してください。
- ディスク・グループの削除については、17-39 ページの「DROP DISKGROUP」を参照してください。

前提条件

この文を発行するには、SYSDBA システム権限が必要です。

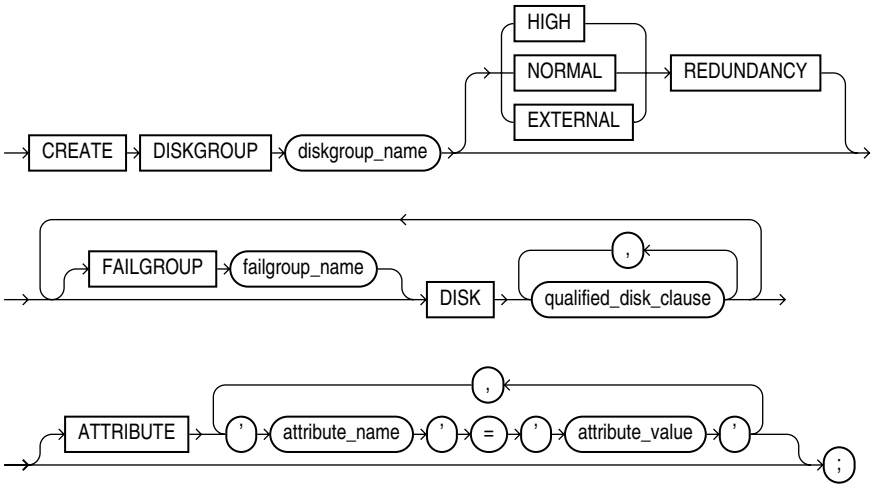
この文を発行する前に、オペレーティング・システムのフォーマット・ユーティリティを使用して、ディスクをフォーマットしておく必要があります。また、Oracle Database ユーザーが読取り / 書込み権限を持ち、ASM_DISKSTRING を使用してディスクが検出可能であることを確認します。

ファイル・システムまたは RAW デバイスではなく、自動ストレージ管理ディスク・グループにデータベース・ファイルを格納している場合、自動ストレージ管理インスタンスを構成および起動してディスク・グループを管理しないと、データベース・インスタンスがディスク・グループのファイルにアクセスできません。

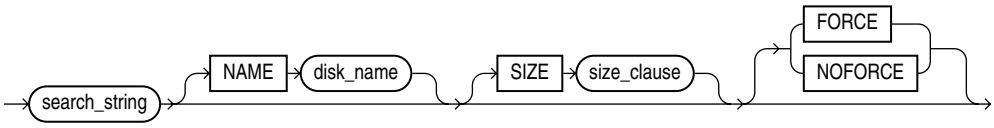
各データベース・インスタンスは、データベースと同じノード上にある 1 つの自動ストレージ管理インスタンスと通信します。同じノード上にある複数のデータベース・インスタンスが、1 つの自動ストレージ管理インスタンスと通信できます。

構文

***create_diskgroup*::=**



***qualified_disk_clause*::=**



（8-42 ページの size_clause::= を参照）

セマンティクス

diskgroup_name

ディスク・グループ名を指定します。ディスク・グループには、データベース・スキーマ・オブジェクトと同じネーミング規則および制限が適用されます。データベース・オブジェクト名の詳細は、2-98 ページの「スキーマ・オブジェクトのネーミング規則」を参照してください。

REDUNDANCY 句

REDUNDANCY 句を使用すると、ディスク・グループの冗長レベルを指定できます。

- NORMAL REDUNDANCY の場合、2 つ以上の障害グループが存在する必要があります（次の「FAILGROUP 句」を参照）。自動ストレージ管理では、ディスク・グループ・テンプレートで指定された属性に従って、ディスク・グループのすべてのファイルの冗長性が提供されます。NORMAL REDUNDANCY ディスク・グループでは、1 つのグループが消失してもリカバリできます。ディスク・グループ・テンプレートの詳細は、10-54 ページの「ALTER DISKGROUP ...」の「diskgroup_template_clauses」を参照してください。
- HIGH REDUNDANCY の場合、少なくとも 3 つの障害グループが存在する必要があります。自動ストレージ管理では、エクステントごとにミラー化された 2 つのコピーが存在する 3 方向ミラー化でミラーリングが行われます。HIGH REDUNDANCY ディスクグループでは、2 つの障害グループが消失してもリカバリできます。
- EXTERNAL REDUNDANCY は、自動ストレージ管理でディスク・グループの冗長性が提供されないことを示します。ディスク・グループ内のディスクは、（たとえばストレージ・アレイを使用して）冗長性を提供する必要があります。または、ディスク障害が発生した場合にディスク・グループが消失しても、（テスト環境などを使用することで）それを許容できる必要があります。EXTERNAL REDUNDANCY を指定する場合、FAILGROUP 句は指定できません。

FAILGROUP 句

この句を使用すると、1 つ以上の障害グループの名前を指定できます。NORMAL REDUNDANCY または HIGH REDUNDANCY を指定している場合にこの句を省略すると、Oracle Database は、自動的にディスク・グループの各ディスクを障害グループに追加します。障害グループの暗黙的な名前は、オペレーティング・システムに依存しないディスク名と同じです（14-42 ページの「NAME 句」を参照）。

EXTERNAL REDUNDANCY ディスク・グループを作成している場合、この句は指定できません。

qualified_disk_clause

DISK *qualified_disk_clause* を指定すると、ディスク・グループにディスクを追加できます。

search_string ディスク・グループに追加するディスクごとに、オペレーティング・システム依存の検索文字列を指定します。自動ストレージ管理では、この文字列を使用してディスクが検索されます。*search_string* は、ASM_DISKSTRING 初期化パラメータの文字列を使用した検索で戻される、ディスクのサブセットを指している必要があります。*search_string* が Oracle Database ユーザーが読取り / 書込み権限を持っているディスクを指していない場合、自動ストレージ管理はエラーを戻します。**search_string** が異なるディスク・グループに割り当てられている 1 つ以上のディスクを指している場合、FORCE を指定していなければ、エラーが戻されます。

自動ストレージ管理では、有効な追加候補のディスクごとにディスク・ヘッダーがフォーマットされ、このディスクが新規のディスク・グループのメンバーであることが示されます。

> **参照：** 検索文字列の指定の詳細は、「ASM_DISKSTRING 初期化パラメータ」を参照してください。

NAME 句 NAME 句は、*search_string* が 1 つのディスクを指す場合にのみ有効です。この句を使用すると、オペレーティング・システムに依存しないディスクの名前を指定できます。名前は、最大 30 文字の英数字で指定できます。最初の文字は、英字にする必要があります。ASMLIB でディスクにラベルを割り当てている場合にこの句を省略すると、ディスク名としてそのラベルが使用されます。ASMLIB でラベルを割り当てていない場合にこの句を省略すると、自動ストレージ管理は、「*diskgroup_name*_####」（#### はディスク番号）という書式のデフォルト名を作成します。この名前を使用して、後続の自動ストレージ管理操作でディスクを参照できます。

SIZE 句 この句を使用すると、ディスクのサイズをバイト単位で指定できます。ディスク容量を超えるサイズを指定すると、自動ストレージ管理はエラーを戻します。ディスク容量よりも小さいサイズを指定した場合、自動ストレージ管理で使用されるディスク領域が制限されます。この句を指定しない場合、自動ストレージ管理はプログラム的にディスクのサイズを決定します。

FORCE FORCE を指定すると、自動ストレージ管理で、別のディスク・グループのメンバーであるディスクをディスク・グループに追加できます。

> **注意：** この方法で FORCE を使用すると、既存のディスク・グループが破棄される可能性があります。

この句を有効にするには、ディスクはディスク・グループのメンバーである必要があり、ディスクがマウントされたディスク・グループの一部であってはなりません。

NOFORCE NOFORCE を指定すると、自動ストレージ管理で、ディスクが別のディスク・グループのメンバーである場合にエラーを戻すことができます。デフォルトは NOFORCE です。

ATTRIBUTE 句

この句を使用すると、ディスク・グループの属性値を設定できます。V$ASM_ATTRIBUTE ビューを問い合せることによって、現在の属性値を確認できます。表 14-1 に、この句で設定できる属性を示します。属性値はすべて文字列です。

表 14-1 ディスク・グループの属性

属性	有効な値	説明
AU_SIZE	サイズ（バイト単位）。有効な値は、1M ～ 64M の 2 の累乗です。例：4M、4194304	割当て単位サイズを指定します。この属性は、ディスク・グループの作成時にのみ設定でき、ALTER DISKGROUP 文で変更することはできません。
COMPATIBLE.RDBMS	有効な Oracle Database バージョン番号（注意 1）	この設定は元に戻すことができません。自動ストレージ管理インスタンスとデータベース・インスタンスの間で交換されるメッセージの形式を指定します。このパラメータには、異なる互換性設定で実行されている異なるデータベース・クライアントに対して、異なる値を設定できます。ただし、すべてのディスク・グループの互換性設定は、ディスク・グループを使用するデータベースの COMPATIBLE 初期化パラメータの値以下である必要があります。
COMPATIBLE.ASM	有効な Oracle Database バージョン番号（注意 1）	この設定は元に戻すことができません。ディスク上の ASM メタデータのデータ構造の形式を制御します。COMPATIBLE.ASM は、常に同じディスク・グループの COMPATIBLE.RDBMS 以上である必要があります。たとえば、ディスク・グループの COMPATIBLE.ASM を 11.0 に設定し、ディスク・グループの COMPATIBLE.RDBMS を 10.1 に設定できます。この場合、バージョンが 11.0 以上の自動ストレージ管理ソフトウェアによってのみディスク・グループは管理可能です。一方、バージョンが 10.1 以上のデータベース・クライアントは、このディスク・グループを使用できます。
DISK_REPAIR_TIME	0 ～ 136 年	ディスクは、オフラインに切り替えられると、デフォルトの時間が経過した後に ASM によって削除されます。compatible.rdbms と compatible.asm の両方の属性が 11.1 以上に設定されている場合、ディスクを修復してオンラインに戻すことができるように、ALTER DISKGROUP ... SET ATTRIBUTE 文で disk_repair_time 属性を使用して、そのデフォルトの時間を変更できます。この属性は、ディスク・グループの作成時には設定できません。 時間は分単位（M）または時間単位（H）で指定できます。指定した経過時間は、ディスク・グループがマウントされているときにのみ加算されます。単位を省略した場合、デフォルトは H になります。この属性を省略し、compatible.rdbms と compatible.asm の両方が 11.1 以上に設定されている場合、デフォルトは 3.6H になります。それ以外の場合、ディスクは即時に削除されます。この属性は、ALTER DISKGROUP ... DISK OFFLINE 文および DROP AFTER 句によって上書きできます。 **注意**：disk_repair_time の現行の値を使用してディスクがオフラインに切り替えられ、その後この属性の値が変更された場合、変更された値が ASM によってディスク・オフライン・ロジックで使用されます。 **参照**：詳細は、10-52 ページの「ALTER DISKGROUP」の「disk_offline_clause」および『Oracle Database ストレージ管理者ガイド』を参照してください。

注意 1: 有効な Oracle Database リリース番号の最初の 2 桁以上を指定してください。有効なバージョン番号の指定の詳細は、『Oracle Database 管理者ガイド』を参照してください。たとえば、*互換性*を 10.2 または 11.1 として指定できます。

参照： これらの属性設定の管理の詳細は、『Oracle Database ストレージ管理者ガイド』を参照してください。

例

次の例では、ASM_DISKSTRING パラメータは $ORACLE_HOME/disks/c* のスーパーセットであり、$ORACLE_HOME/disks/c* が自動ストレージ管理ディスクとして使用される 1 つ以上のデバイスを指し、Oracle Database ユーザーはディスクに対する読取り / 書込み権限を持っていることを想定しています。

参照： 自動ストレージ管理およびディスク・グループを使用してデータベース管理を簡略化する方法については、『Oracle Database ストレージ管理者ガイド』を参照してください。

ディスク・グループの作成例： 次の文は、自動ストレージ管理ディスク・グループ dgroup_01 を作成します。このディスク・グループには、自動ストレージ管理によって冗長性が提供されておらず、*search_string* と一致するすべてのディスクが含まれています。

```
CREATE DISKGROUP dgroup_01
  EXTERNAL REDUNDANCY
  DISK '$ORACLE_HOME/disks/c*';
```

CREATE FLASHBACK ARCHIVE

用途

CREATE FLASHBACK ARCHIVE 文を使用すると、フラッシュバック・データ・アーカイブを作成できます。フラッシュバック・データ・アーカイブによって、指定したデータベース・オブジェクトへのトランザクションでのデータ変更を自動的に追跡およびアーカイブできます。フラッシュバック・データ・アーカイブは複数の表領域で構成され、追跡対象の表に対するすべてのトランザクションの履歴データが格納されます。

フラッシュバック・データ・アーカイブには、RETENTION パラメータで指定した期間の履歴データが格納されます。履歴データは、フラッシュバック問合せの AS OF 句を使用して問い合せることができます。指定した保存期間を超えた古いアーカイブ済履歴データは、自動的に消去されます。

フラッシュバック・データ・アーカイブには、データベースに対するデータ定義言語（DDL）の変更があっても、DDL の変更が表の構造に影響しないかぎり、履歴データが保持されます。この規則の例外の 1 つとして、表に列が追加された場合にはフラッシュバック・データ・アーカイブに履歴データが保持されます。

参照：

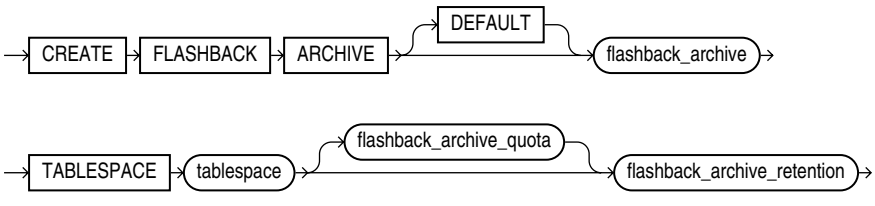
- フラッシュバック・データ・アーカイブの使用方法の概要は、『Oracle Database アドバンスト・アプリケーション開発者ガイド』を参照してください。
- 表を追跡対象の表として指定する方法の詳細は、16-56 ページの「CREATE TABLE」の「flashback_archive_clause」を参照してください。
- フラッシュバック・データ・アーカイブの割当て制限属性と保存属性の変更およびフラッシュバック・データ・アーカイブにおける表領域の記憶域の追加と変更の詳細は、10-60 ページの「ALTER FLASHBACK ARCHIVE」を参照してください。

前提条件

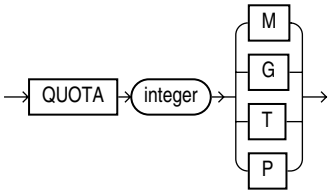
フラッシュバック・データ・アーカイブを作成するには、FLASHBACK ARCHIVE ADMINISTER システム権限が必要です。また、フラッシュバック・データ・アーカイブを作成するには CREATE TABLESPACE システム権限が必要であり、履歴情報が格納される表領域に十分な割当て制限も必要です。フラッシュバック・データ・アーカイブをシステムのデフォルトのフラッシュバック・データ・アーカイブとして指定するには、SYSDBA としてログインしている必要があります。

構文

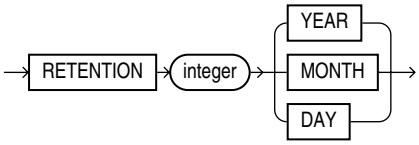
***create_flashback_archive*::=**



flashback_archive_quota::=



flashback_archive_retention::=



セマンティクス

DEFAULT

DEFAULT を指定するには、SYSDBA としてログインしている必要があります。この句を使用すると、このフラッシュバック・データ・アーカイブをデータベースのデフォルトのフラッシュバック・データ・アーカイブとして指定できます。CREATE TABLE 文または ALTER TABLE 文に、フラッシュバック・データ・アーカイブ名を指定しないで *flashback_archive_clause* が指定されている場合は、データベースはデフォルトのフラッシュバック・データ・アーカイブを使用して、該当する表のデータを格納します。

デフォルトのフラッシュバック・データ・アーカイブがすでに存在する場合は、この句を指定することはできません。ただし、ALTER FLASHBACK ARCHIVE ... SET DEFAULT 句を使用して、既存のデフォルトのフラッシュバック・データ・アーカイブを置き換えることはできます。

> **参照：** 詳細は、16-56 ページの「CREATE TABLE」の「flashback_archive_clause」を参照してください。

flashback_archive

フラッシュバック・データ・アーカイブの名前を指定します。名前は、2-98 ページの「スキーマ・オブジェクトのネーミング規則」に指定されている要件を満たしている必要があります。

TABLESPACE 句

このフラッシュバック・データ・アーカイブのアーカイブ・データが格納される表領域を指定します。この句では表領域を 1 つのみ指定できます。ただし、ALTER FLASHBACK ARCHIVE 文を使用して、後で表領域をフラッシュバック・データ・アーカイブに追加できます。

flashback_archive_quota

アーカイブ・データ用に確保する初期表領域の領域を指定します。フラッシュバック・データ・アーカイブ内のアーカイブ用の領域が一杯になると、このフラッシュバック・データ・アーカイブを使用する追跡対象の表での DML 操作は失敗します。フラッシュバック・データ・アーカイブの内容が指定された割当て制限の 90% になると、領域不足のアラートがデータベースから発行され、古いデータを消去するか、割当て制限を追加する時間が与えられます。この句を省略すると、指定された表領域でフラッシュバック・データ・アーカイブの割当て制限が無制限になります。

flashback_archive_retention

アーカイブされたデータがフラッシュバック・データ・アーカイブに保持される時間の長さを月、日または年単位で指定します。指定した時間でフラッシュバック・データ・アーカイブが一杯になると、データベースは 14-46 ページの「flashback_archive_quota」で説明されているように応答します。

例

次の文では、2 つのフラッシュバック・データ・アーカイブがテストのために作成されます。最初のフラッシュバック・データ・アーカイブは、データベースのデフォルトとして指定されます。どちらも、領域割当て制限は 1MB であり、アーカイブの保存は 1 日です。

```
CREATE FLASHBACK ARCHIVE DEFAULT test_archive1
   TABLESPACE example
   QUOTA 1 M
   RETENTION 1 DAY;

CREATE FLASHBACK ARCHIVE test_archive2
   TABLESPACE example
   QUOTA 1 M
   RETENTION 1 DAY;
```

次の文では、デフォルトのフラッシュバック・データ・アーカイブを変更して、保存期間を 1 か月に延長します。

```
ALTER FLASHBACK ARCHIVE test_archive1
   MODIFY RETENTION 1 MONTH;
```

次の文では、oe.customers 表の追跡を指定します。フラッシュバック・データ・アーカイブが指定されていないため、データはデフォルトのフラッシュバック・データ・アーカイブ test_archive1 にアーカイブされます。

```
ALTER TABLE oe.customers
   FLASHBACK ARCHIVE;
```

次の文では、oe.orders 表の追跡を指定します。この場合、データは指定したフラッシュバック・データ・アーカイブ test_archive2 にアーカイブされます。

```
ALTER TABLE oe.orders
   FLASHBACK ARCHIVE test_archive2;
```

次の文では、test_archive2 フラッシュバック・データ・アーカイブを削除します。

```
DROP FLASHBACK ARCHIVE test_archive2;
```

CREATE FUNCTION

用途

ファンクションは PL/SQL を使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

CREATE FUNCTION 文を使用すると、スタンドアロン・ストアド・ファンクションまたはコール仕様を作成できます。

- **ストアド・ファンクション**（**ユーザー・ファンクション**または**ユーザー定義ファンクション**）は、名前でコールできる PL/SQL 文の集合です。ストアド・ファンクションは、プロシージャとよく似ていますが、ファンクションは、コールした環境に値を戻す点で異なります。ストアド・ファンクションは、SQL 式の一部として使用できます。
- **コール仕様**は、PL/SQL からコールできるように、Java メソッドまたは第三世代言語（3GL）ルーチンを宣言します。CALL SQL 文を使用して、そのようなメソッドまたはルーチンをコールすることもできます。コール仕様は、コールされたときに起動する Java メソッドまたは共有ライブラリの名前付きファンクションを Oracle Database に指示します。また、引数および戻り値に対して実行する型変換もデータベースに指示します。

注意： CREATE PACKAGE 文を使用して、パッケージの一部としてファンクションを作成することもできます。

参照：

- プロシージャおよびファンクションの概要は 15-45 ページの「CREATE PROCEDURE」を参照してください。パッケージの作成については、15-39 ページの「CREATE PACKAGE」を参照してください。ファンクションの変更および削除については、10-63 ページの「ALTER FUNCTION」および 17-42 ページの「DROP FUNCTION」を参照してください。
- 共有ライブラリについては、15-2 ページの「CREATE LIBRARY」を参照してください。
- 外部ファンクションの登録の詳細は、『Oracle Database アドバンスト・アプリケーション開発者ガイド』を参照してください。

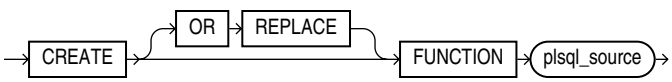
前提条件

自分のスキーマ内にファンクションを作成または再作成する場合は、CREATE PROCEDURE システム権限が必要です。他のユーザーのスキーマ内にファンクションを作成または再作成する場合は、CREATE ANY PROCEDURE システム権限が必要です。

構文

ファンクションは PL/SQL を使用して定義されます。このため、このマニュアルの構文図では SQL キーワードのみを示します。PL/SQL の構文、セマンティクスおよび例については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

***create_function*::=**



（*plsql_source* については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。）

セマンティクス

OR REPLACE

OR REPLACE を指定すると、既存のファンクションを再作成できます。この句を指定した場合、既存のファンクションに付与されているオブジェクト権限を削除、再作成および再付与しなくても、そのファンクションの定義を変更できます。ファンクションを再定義した場合、そのファンクションは再コンパイルされます。

再定義したファンクションに対して権限が付与されていたユーザーは、権限を再付与されなくても、そのファンクションにアクセスできます。

ファンクション索引がファンクションに依存している場合、索引に DISABLED のマークが付きます。

> **参照：** SQL を使用したファンクションの再コンパイルについては、「ALTER FUNCTION」を参照してください。

plsql_source

plsql_source の構文、セマンティクスおよび例については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

CREATE INDEX

用途

CREATE INDEX 文を使用すると、次のものに索引を作成できます。

- 表の 1 つ以上の列、パーティション表、索引構成表またはクラスタ
- 表またはクラスタの 1 つ以上のスカラー型オブジェクト属性
- ネストした表の列の索引を作成するためのネストした表の記憶表

索引は、スキーマ・オブジェクトの 1 つで、索引には、表またはクラスタの索引付き列の中に表示される各値のエントリが入ります。索引を使用した場合、行に直接、かつ高速にアクセスできます。Oracle Database は、次の索引をサポートしています。

- 通常の索引。(デフォルトでは B ツリー索引が作成されます。)
- **ビットマップ索引**。キー値に関連付けられた ROWID をビットマップとして格納します。
- **パーティション索引**。表の索引付き列に表示される各値のエントリを含むパーティションで構成されます。
- **ファンクション索引**。式をベースとしています。式によって戻される値を評価する問合せを組み立てることができます。その式には組込みファンクションまたはユーザー定義ファンクションが含まれることがあります。
- **ドメイン索引**。アプリケーション固有の *indextype* 索引タイプのインスタンスです。

参照:

- 索引については、『Oracle Database 概要』を参照してください。
- 10-64 ページの「ALTER INDEX」および 17-44 ページの「DROP INDEX」を参照してください。

前提条件

自分のスキーマ内に索引を作成する場合は、次のいずれかの条件が満たされている必要があります。

- 索引を作成する表またはクラスタが自分のスキーマ内に定義されている。
- 索引を作成する表に対する INDEX オブジェクト権限を持っている。
- CREATE ANY INDEX システム権限を持っている。

他のユーザーのスキーマ内に索引を作成する場合は、CREATE ANY INDEX システム権限が必要です。また、索引が定義されているスキーマの所有者には、UNLIMITED TABLESPACE システム権限、あるいはその索引または索引パーティションを格納するための表領域の割当て制限のいずれかが必要です。

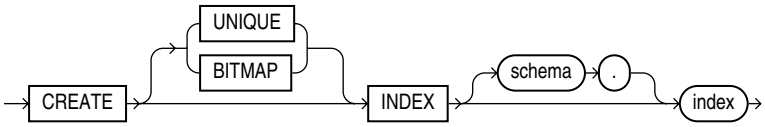
自分のスキーマにドメイン索引を作成する場合、従来索引の作成の前提条件の他に、索引タイプに対する EXECUTE オブジェクト権限が必要です。他のユーザーのスキーマにドメイン索引を作成する場合、索引の所有者にも索引タイプおよびその基礎となる実装タイプに対する EXECUTE オブジェクト権限が必要です。ドメイン索引を作成する前に、索引タイプを定義する必要があります。

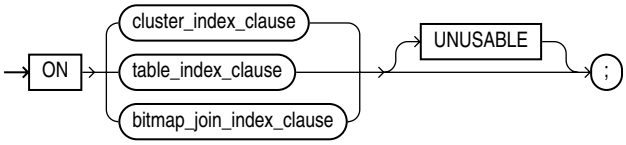
ファンクション索引を作成する場合は、従来索引の作成の前提条件の他に、索引がユーザー定義ファンクションに基づいている場合は、ファンクションに DETERMINISTIC のマークを付ける必要があります。また、これらのファンクションが他のユーザーに所有されている場合、ファンクション索引で使用されるユーザー定義ファンクションに対する、EXECUTE オブジェクト権限を持っている必要があります。

参照: 「CREATE INDEXTYPE」(14-73 ページ)

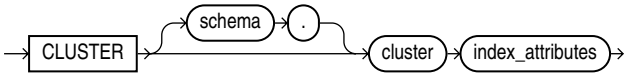
構文

***create_index*::=**



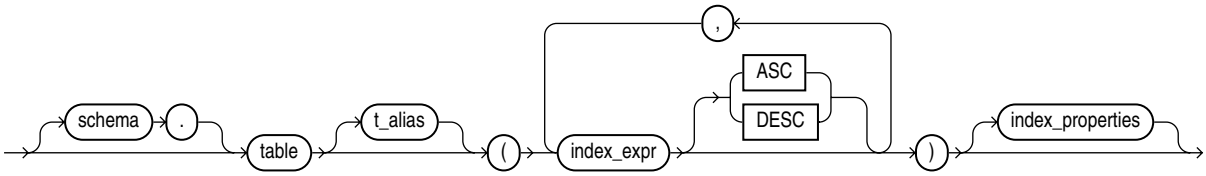


cluster_index_clause::=



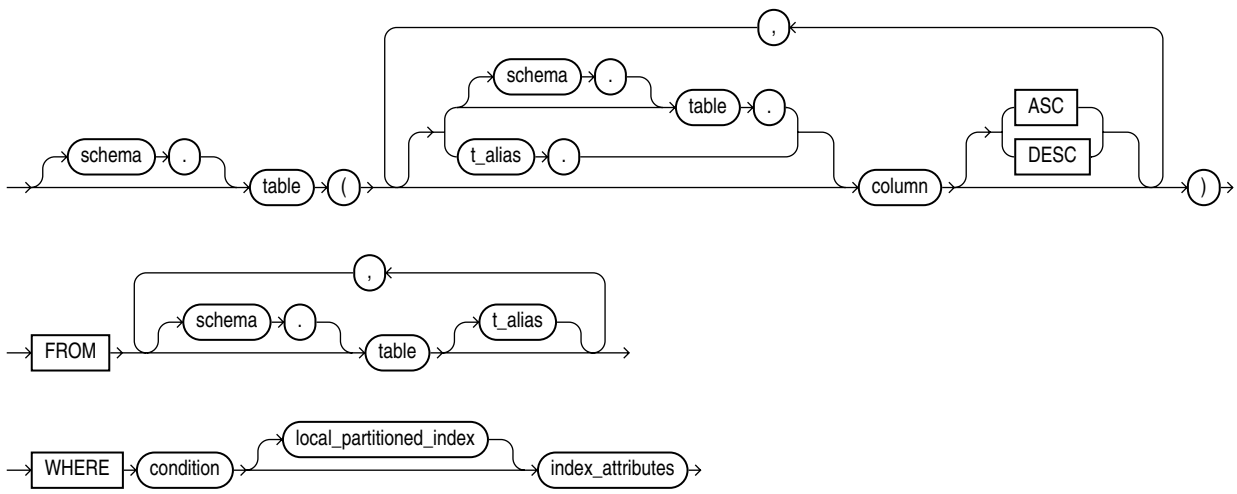
（14-52 ページの index_attributes::= を参照）

table_index_clause::=



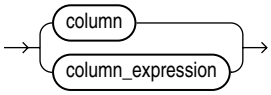
（14-52 ページの index_properties::= を参照）

bitmap_join_index_clause::=

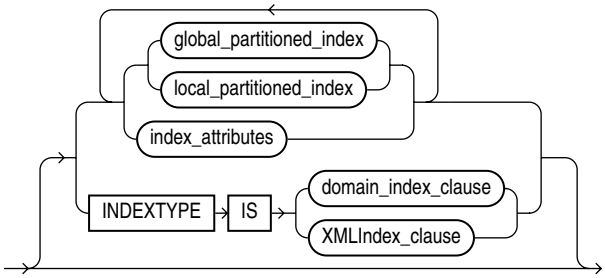


（14-55 ページの local_partitioned_index::=、14-52 ページの index_attributes::= を参照）

index_expr::=

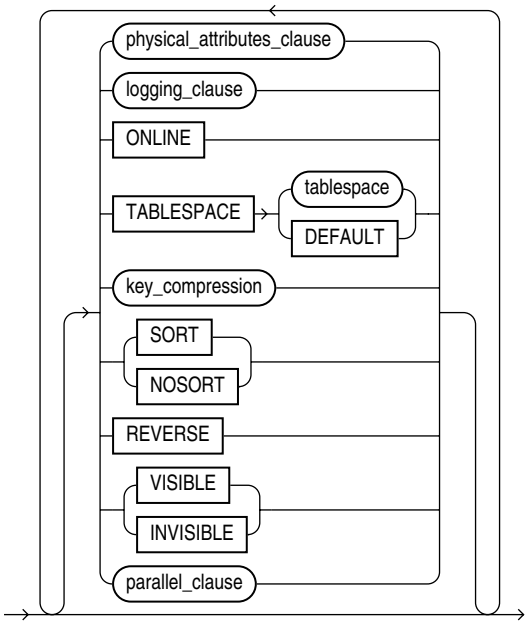


***index_properties*::=**



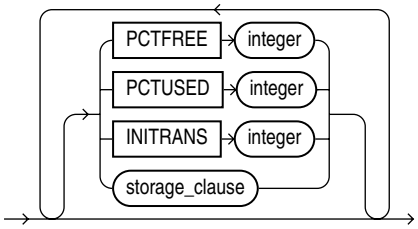
（14-53 ページの global_partitioned_index::=、14-55 ページの local_partitioned_index::=、14-52 ページの index_attributes::=、14-53 ページの domain_index_clause::=、14-53 ページの XMLIndex_clause::= を参照）

index_attributes::=



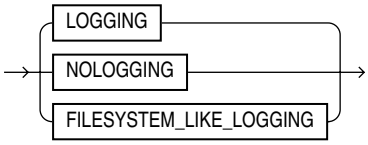
（14-52 ページの physical_attributes_clause::=、14-53 ページの logging_clause::=、14-53 ページの key_compression::=、14-56 ページの parallel_clause::= を参照）

physical_attributes_clause::=

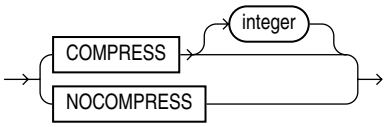


（8-44 ページの storage_clause::= を参照）

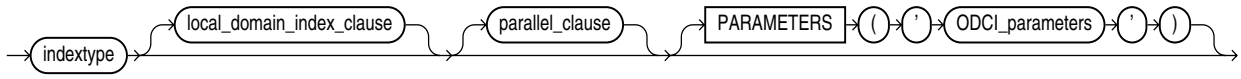
logging_clause::=



key_compression::=

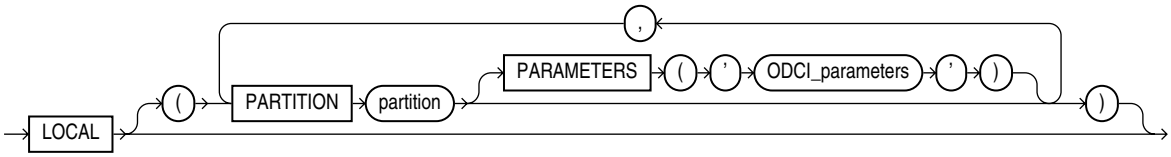


domain_index_clause::=

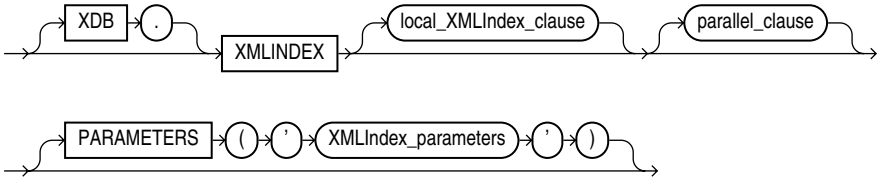


（14-56 ページの parallel_clause::= を参照）

local_domain_index_clause::=

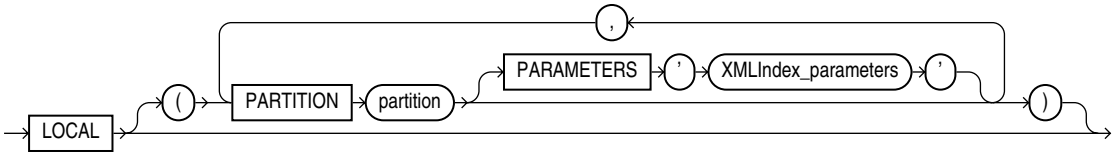


XMLIndex_clause::=

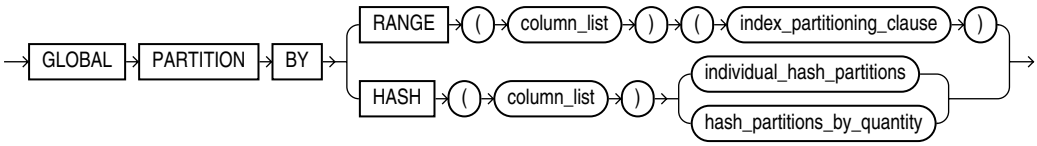


（*XMLIndex_parameters* については、『Oracle XML DB 開発者ガイド』を参照してください。）

***local_XMLIndex_clause*::=**

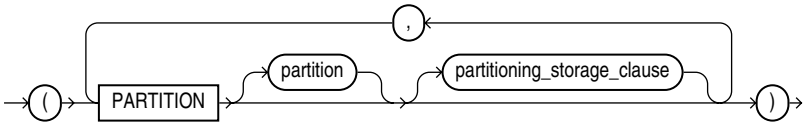


global_partitioned_index::=



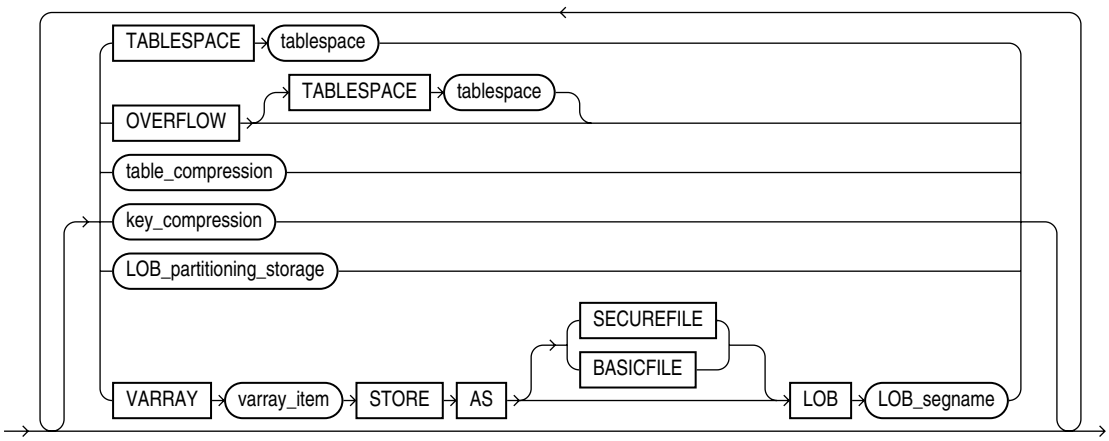
（14-54 ページの index_partitioning_clause::=、14-54 ページの individual_hash_partitions::=、14-54 ページの hash_partitions_by_quantity::= を参照）

***individual_hash_partitions*::=**

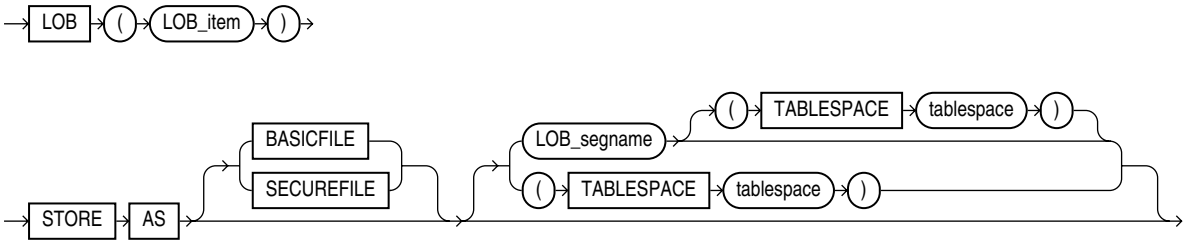


（14-54 ページの partitioning_storage_clause::= を参照）

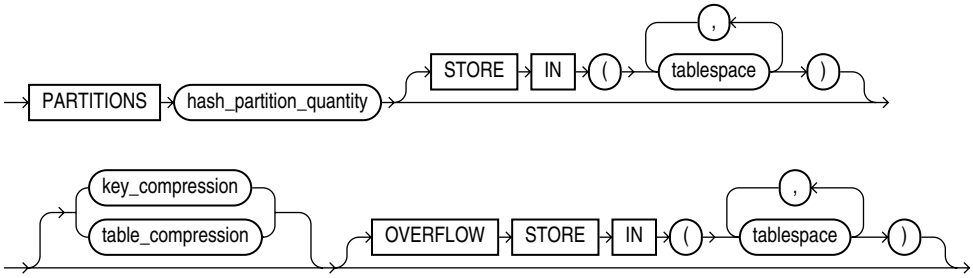
***partitioning_storage_clause*::=**



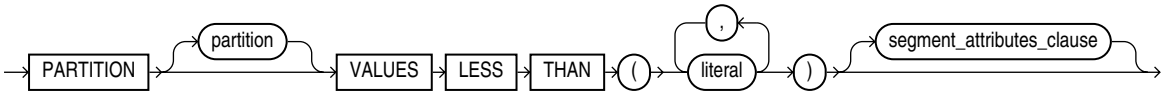
***LOB_partitioning_storage*::=**



***hash_partitions_by_quantity*::=**

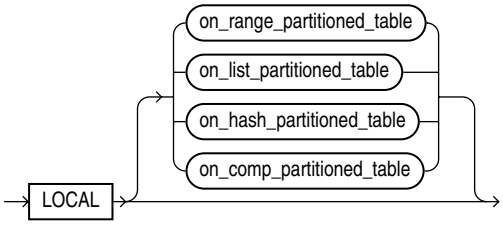


***index_partitioning_clause*::=**



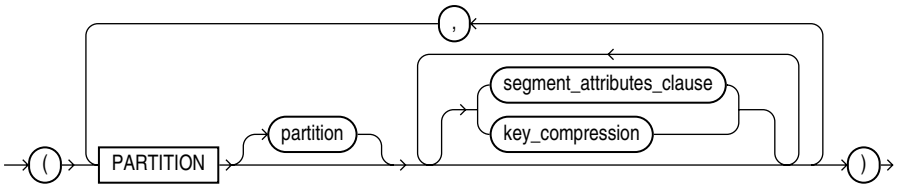
（14-55 ページの segment_attributes_clause::= を参照）

local_partitioned_index::=



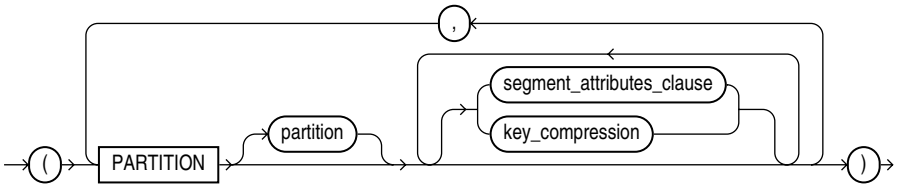
（14-55 ページの on_range_partitioned_table::=、14-55 ページの on_list_partitioned_table::=、14-55 ページの on_hash_partitioned_table::=、14-56 ページの on_comp_partitioned_table::= を参照）

on_range_partitioned_table::=



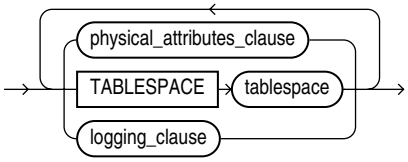
（14-55 ページの segment_attributes_clause::= を参照）

***on_list_partitioned_table*::=**



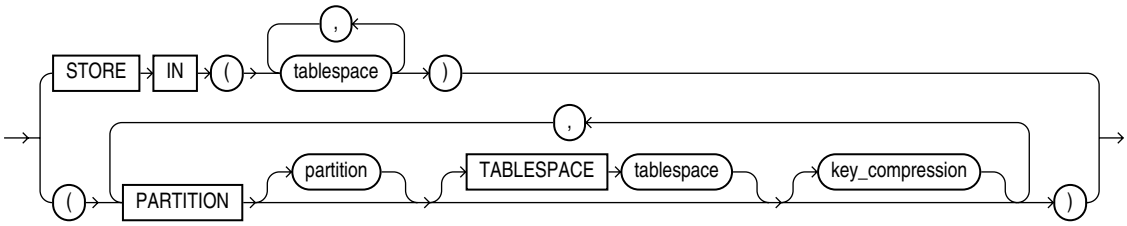
（14-55 ページの segment_attributes_clause::= を参照）

***segment_attributes_clause*::=**

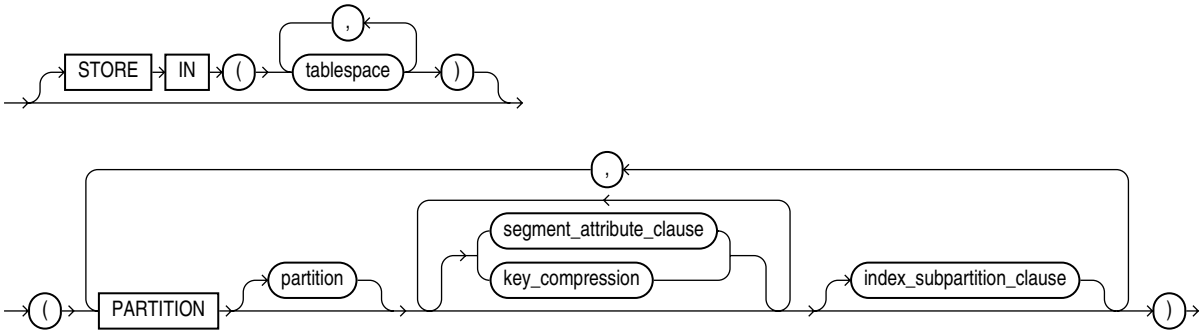


（14-52 ページの physical_attributes_clause::=、14-53 ページの logging_clause::= を参照）

on_hash_partitioned_table::=

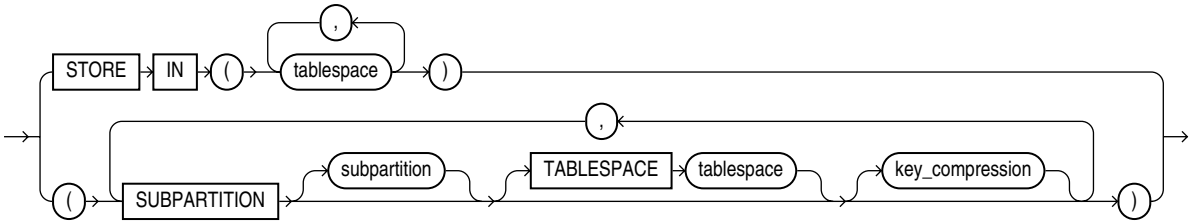


on_comp_partitioned_table::=

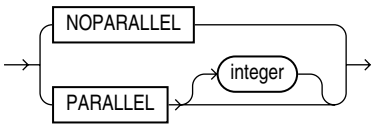


(14-55 ページの segment_attributes_clause::=、14-56 ページの index_subpartition_clause::= を参照)

***index_subpartition_clause*::=**



***parallel_clause*::=**



(8-44 ページの storage_clause::= を参照)

セマンティクス

UNIQUE

UNIQUE を指定すると、索引のベースとなっている列の値が一意である必要があることを指定できます。

一意索引の制限事項： 一意索引には、次の制限事項があります。

- UNIQUE および BITMAP は同時に指定できません。
- ドメイン索引には UNIQUE を指定できません。

> **参照：** 一意制約を満たす条件については、8-8 ページの「一意制約」を参照してください。

BITMAP

BITMAP を指定すると、*index* が各行を分割した索引付けではなく、各個別キーのビットマップで作成されることを指定できます。ビットマップ索引では、キー値にビットマップとして関連付けられた ROWID が格納されます。ビットマップ内の各ビットは、使用可能な ROWID に対応しています。ビットが設定されていれば、それに対応する ROWID を持つ行に、キー値が設定されていることになります。ビットマップの内部表現は、データ・ウェアハウスなど、低レベルの同時実行トランザクションが実行されるアプリケーションに最適です。

注意： Oracle では、すべてのキー列が NULL の表の行には索引を付けません（ただし、ビットマップ索引の場合を除きます）。したがって、表のすべての行に索引を作成する場合、索引のキー列に NOT NULL 制約を指定するか、またはビットマップ索引を作成する必要があります。

ビットマップ索引の制限事項： ビットマップ索引には、次の制限事項があります。

- グローバル・パーティション索引の作成には BITMAP を指定できません。
- 索引構成表がビットマップ化した 2 次索引に対応するマッピング表を持たない場合、索引構成表にビットマップ化した 2 次索引を作成できません。
- UNIQUE および BITMAP は同時に指定できません。
- ドメイン索引には BITMAP を指定できません。

参照：

- ビットマップ索引の使用の詳細は、『Oracle Database 概要』および『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。
- マッピング表の詳細は、16-6 ページの「CREATE TABLE」を参照してください。
- 「ビットマップ索引の例」（14-71 ページ）

schema

索引を作成するスキーマを指定します。*schema* を指定しない場合、自分のスキーマ内に索引が作成されます。

index

作成する索引の名前を指定します。

参照： 14-68 ページの「索引の作成例 :」および 14-68 ページの「XMLType 表の索引の作成例 :」を参照してください。

cluster_index_clause

cluster_index_clause を使用すると、クラスタ索引を作成するクラスタを指定できます。cluster を *schema* で修飾しない場合、そのクラスタは自分のスキーマ内にあるとみなされます。ハッシュ・クラスタにはクラスタ索引を作成できません。

参照： 14-2 ページの「CREATE CLUSTER」および 14-68 ページの「クラスタ索引の作成例 :」を参照してください。

table_index_clause

索引を定義する表を指定します。*table* を *schema* で修飾しない場合、その表は自分のスキーマにあるとみなされます。

ネストした表の記憶表に索引を作成することによって、ネストした表の列に索引を作成します。記憶表の NESTED_TABLE_ID 疑似列を組み込んだ一意索引を作成することは、ネストした表の値を持つ行がそれぞれ確実に異なるようにする有効な手段です。

参照： 「ネストした表の索引の例 :」（14-72 ページ）

セッションがバインドされていない場合にのみ、一時表で DDL 操作（ALTER TABLE、DROP TABLE、CREATE INDEX など）を実行できます。セッションを一時表にバインドするには、一時表で INSERT 操作を実行します。セッションを一時表からアンバインドするには、

TRUNCATE 文を発行するか、セッションを終了します。また、トランザクション固有の一時表からアンバインドするには、COMMIT または ROLLBACK 文を発行します。

***table_index_clause* の制限事項：** この句には、次の制限事項があります。

- ローカル・パーティション索引の場合は、*table* をパーティション化する必要があります。
- 索引構成表の場合、この文は 2 次索引を作成します。この索引には、索引構成表の索引キーおよび論理 ROWID が含まれます。論理 ROWID は、索引キーに含まれる列を除外します。この 2 次索引には、REVERSE を指定できません。索引キーおよび論理 ROWID の結合サイズは、ブロック・サイズの半分未満にする必要があります。
- *table* が一時表の場合、索引も *table* と同様の有効範囲（セッションまたはトランザクション）を持つ一時的なものとなります。一時表の索引には、次の制限があります。
  - *index_properties* で指定できるのは、*index_attributes* の部分のみです。
  - *index_attributes* では、*physical_attributes_clause*、*parallel_clause*、*logging_clause* または TABLESPACE を指定できません。
  - 一時表には、ドメイン索引は作成できません。

  参照： 一時表の詳細は、16-6 ページの「CREATE TABLE」および『Oracle Database 概要』を参照してください。

t_alias

索引を作成する表に対して相関名（別名）を指定します。

注意： *index_expr* がオブジェクト型属性またはオブジェクト型メソッドを参照する場合、この別名が必要になります。14-69 ページの「型メソッドのファンクション索引の作成例：」および 14-72 ページの「置換可能な列の索引の作成例：」を参照してください。

---

index_expr

索引のベースとなる列または列の式を指定します。

column 表内の 1 つ以上の列の名前を指定します。ビットマップ索引には最大 30 列を指定できます。他の索引には最大 32 列を指定できます。これらの列は**主キー**を定義します。

索引がローカル非同一キー索引（14-89 ページの「local_partitioned_index」を参照）の場合、索引キーはパーティション・キーを含んでいる必要があります。

スカラー・オブジェクト属性列またはネストした表の記憶表のシステム定義の NESTED_TABLE_ID 列には索引を作成できます。オブジェクト属性列を指定する場合、列名を表名で修飾する必要があります。ネストした表の列属性を指定する場合は、この属性は、一番外側の表の名前、ネストした表が定義されている列の名前、およびそのネストした表の列属性となるすべての中間属性の名前で修飾する必要があります。

索引列の制限事項： Oracle Database でサポートされている、SCOPE 句で定義した REF 型の列または属性の索引を除き、ユーザー定義型、LONG、LONG RAW、LOB または REF 型の列または属性に索引を作成できません。

column_expression *table* の列、定数、SQL ファンクションおよびユーザー定義ファンクションから作成された式を指定します。*column_expression* を指定した場合、**ファンクション索引**が作成されます。

参照： 6-6 ページの「列式」、14-59 ページの「ファンクション索引の注意事項：」、14-59 ページの「ファンクション索引の制限事項：」および 14-69 ページの「ファンクション索引の例」を参照してください。

ファンクションの名前解決は、索引作成者のスキーマに基づきます。*column_expression* で使用されるユーザー定義ファンクションは、CREATE INDEX 操作中に完全に名前解決されます。

ファンクション索引の作成後、DBMS_STATS パッケージを使用して、索引と実表の両方に関する統計情報を収集します。これらの統計情報によって、Oracle Database は索引を使用するタイミングを正しく判断できます。

ファンクションの一意索引は、列または列の組合せに対して条件付きの一意制約を定義する場合に便利なことがあります。例については、14-70 ページの「ファンクション索引の使用による条件付き一意性の定義例 :」を参照してください。

> **参照 :** DBMS_STATS パッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

ファンクション索引の注意事項 : ファンクション索引には、次の注意事項があります。

- ファンクション索引を使用する表を後で問い合せる場合、問合せで NULL が除外されないかぎり索引は使用されません。ただし、WHERE 句に指定した列の順序が、ファンクション索引を定義した *column_expression* での順序と異なる場合でも、問合せにファンクション索引が使用されます。

  > **参照 :** 「ファンクション索引の例」(14-69 ページ)

- 索引のベースになるファンクションが無効であるか、または削除された場合は、索引に DISABLED のマークが付けられます。オプティマイザが索引の使用を選択した場合、DISABLED 索引の問合せは失敗します。索引に UNUSABLE のマークも付けて、パラメータ SKIP_UNUSABLE_INDEXES を true に設定しないかぎり、DISABLED 索引の DML 操作は失敗します。このパラメータの詳細は、11-41 ページの「ALTER SESSION」を参照してください。
- ファンクション、パッケージまたは型のパブリック・シノニムが *column_expression* で使用され、後で同じ名前の実際のオブジェクトが表の所有者のスキーマに作成された場合、ファンクション索引は使用禁止になります。その後、ALTER INDEX ... ENABLE または ALTER INDEX ... REBUILD を使用してファンクション索引を使用可能にした場合、*column_expression* で使用されているファンクション、パッケージまたは型は、パブリック・シノニムが最初に指していたファンクション、パッケージまたは型に継続して変換されます。新しいファンクション、パッケージまたは型への変換は行われません。
- ファンクション索引の定義によって文字データに内部変換が生成される場合に、NLS パラメータの設定を変更するときは注意が必要です。ファンクション索引は、NLS パラメータの現行のデータベース設定を使用します。セッション・レベルでパラメータを再設定した場合、ファンクション索引を使用して問合せを行うと、無効な結果が戻る場合があります。2 つの照合パラメータ (NLS_SORT および NLS_COMP) は例外です。これらがセッション・レベルで再設定された場合でも、Oracle Database は正常に変換を処理します。
- 変換が明示的に要求された場合でも、Oracle Database はすべての場合にデータを変換できるわけではありません。たとえば、TO_NUMBER ファンクションを使用して文字列 '105 lbs' を VARCHAR2 から NUMBER に変換しようとすると、エラーが発生して失敗します。したがって、*column_expression* に TO_NUMBER や TO_DATE などのデータ変換ファンクションが含まれており、後続の INSERT または UPDATE 文に変換ファンクションで変換できないデータが含まれている場合、INSERT または UPDATE 文は索引が原因で失敗します。

ファンクション索引の制限事項 : ファンクション索引には、次の制限事項があります。

- *column_expression* で参照されるファンクションが戻す値には、B ツリー索引の索引列と同じ制限があります。詳細は、14-58 ページの「索引列の制限事項 :」を参照してください。
- *column_expression* で参照されるユーザー定義ファンクションは、DETERMINISTIC として宣言されている必要があります。

- グローバル・パーティション・ファンクション索引では、*column_expression* がパーティション・キーであってはいけません。
- *column_expression* には、6-6 ページの「列式」で説明されているすべての形式の式を指定できます。
- パラメータがない場合も、すべてのファンクションをカッコで指定する必要があります。カッコで指定していない場合は、列名として解析されます。
- *column_expression* で指定するファンクションは、リピータブル値を戻す必要があります。たとえば、SYSDATE や USER ファンクションまたは ROWNUM 疑似列は指定できません。

> **参照：** 14-48 ページの「CREATE FUNCTION」および『Oracle Database PL/SQL 言語リファレンス』を参照してください。

ASC | DESC

ASC または DESC を使用すると、索引を昇順で作成するか降順で作成するかを指定できます。文字データの索引は、データベース・キャラクタ・セットの文字値の昇順または降順で作成されます。

Oracle Database は、降順索引をファンクション索引として扱います。他のファンクション索引のように、最初に索引および索引が定義されている表を分析するまで、降順索引は使用しません。14-83 ページの「column_expression」を参照してください。

昇順の一意索引は、複数の NULL 値を含むことができます。ただし、降順の一意索引の場合は、複数の NULL 値は重複した値として扱われるため、許可されていません。

昇順索引および降順索引の制限事項： これらの句は、ドメイン索引に対して指定できません。逆索引には DESC を指定できません。*index* をビットマップ化したり、COMPATIBLE 初期化パラメータに 8.1.0 未満の値を設定すると、DESC は無視されます。

index_attributes

オプションの索引属性を指定します。

physical_attributes_clause *physical_attributes_clause* を使用すると、索引の物理特性および記憶特性の値を設定できます。

この句を指定しない場合、PCTFREE が 10、INITRANS が 2 に設定されます。

索引の物理属性の制限事項： PCTUSED パラメータは、索引に対して指定できません。

> **参照：** これらの句の詳細は、8-39 ページの「physical_attributes_clause」および 8-41 ページの「storage_clause」を参照してください。

TABLESPACE 索引、索引パーティションまたは索引サブパーティションを格納する表領域の名前を指定します。この句を指定しない場合、その索引を定義しているスキーマの所有者のデフォルトの表領域内に索引が作成されます。

ローカル索引の場合、*tablespace* のかわりにキーワード DEFAULT を指定できます。ローカル索引に追加される新規パーティションまたはサブパーティションは、基礎となる表の対応するパーティションまたはサブパーティションと同じ表領域内に作成されます。

key_compression COMPRESS を指定すると、キー圧縮を使用可能にできます。これによって、キー列値の繰返しがなくなり、記憶域を大幅に削減できます。*integer* を使用して、接頭辞の長さ（圧縮する接頭辞列数）を指定します。

Oracle Database では、索引（一意でない索引または 2 列以上の一意索引）が圧縮されます。パーティション索引の圧縮を使用する場合は、索引レベルで圧縮を有効にして索引を作成する必要があります。そのようなパーティション索引の個々のパーティションの圧縮設定は、後で有効および無効にすることができます。個々のパーティションを再作成するときに圧縮を有効および無効にできます。索引の再作成時にのみ、既存の非パーティション索引を変更して圧縮を有効または無効にできます。

- 一意索引の場合、接頭辞の長さの有効範囲は、1 ～（キー列の数 - 1）です。デフォルトの接頭辞の長さは、（キー列の数 - 1）です。
- 一意でない索引の場合、接頭辞の長さの有効範囲は、1 ～キー列の数です。デフォルトの接頭辞の長さはキー列数です。

キー圧縮の制限事項： COMPRESS は、ビットマップ索引に対して指定できません。

参照： 「索引の圧縮例 :」（14-68 ページ）

NOCOMPRESS NOCOMPRESS を指定すると、キー圧縮を使用禁止にできます。これはデフォルトです。

SORT | NOSORT 索引の作成時に、Oracle Database はデフォルトで索引を昇順ソートします。NOSORT を指定すると、索引の作成時に、データベース内ですでに昇順で格納されている行のソートを行わないようにできます。索引列の行または列が昇順に格納されていない場合、データベースはエラーを戻します。ソート時間および領域を削減するため、列を表へ初期ロードした直後にこの句を使用します。これらのキーワードのいずれも指定しない場合、デフォルトで SORT が使用されます。

NOSORT の制限事項： このパラメータには、次の制限事項があります。

- この句は、REVERSE と同時には指定できません。
- この句を使用して、クラスタ索引、パーティション索引またはビットマップ索引を作成することはできません。
- 索引構成表の 2 次索引には、この句を指定できません。

REVERSE REVERSE を指定すると、ROWID 以外の索引ブロックのバイトを逆順に格納できます。

逆索引の制限事項： 逆索引には、次の制限事項があります。

- この句は、NOSORT と同時には指定できません。
- ビットマップ索引または索引構成表の索引は逆順にできません。

VISIBLE | INVISIBLE この句を使用すると、オプティマイザで索引を参照可能にするかどうかを指定できます。参照不可の索引は DML 操作によってメンテナンスされますが、パラメータ OPTIMIZER_USE_INVISIBLE_INDEXES をセッションまたはシステム・レベルで明示的に TRUE に設定しないと、問合せ時にオプティマイザによって使用されません。

既存の索引がオプティマイザによって参照可能か参照不可かを確認するには、USER_、DBA_、ALL_INDEXES データ・ディクショナリ・ビューの VISIBILITY 列を問い合せます。

参照： この機能の詳細は、『Oracle Database 管理者ガイド』を参照してください。

logging_clause 索引作成を、REDO ログ・ファイル内に記録する（LOGGING）か記録しない（NOLOGGING）かを指定します。この設定によって、索引に対する後続のダイレクト・ローダー（SQL*Loader）およびダイレクト・パス・インサート操作を記録するか記録しないかも決定されます。デフォルトは LOGGING です。

非パーティション索引の場合、この句は索引のロギング属性を指定します。

パーティション索引の場合、この句は次の値を決定します。

- CREATE 文で指定されたすべてのパーティションのデフォルト値（PARTITION 記述句で *logging_clause* を指定している場合を除く）
- 索引パーティションに関連付けられたセグメントに対するデフォルト値
- 後続の ALTER TABLE ... ADD PARTITION 操作中に暗黙的に追加されたローカル索引パーティションまたはサブパーティションに対するデフォルト値

索引のロギング属性は、その実表の属性に依存しません。

この句を指定しない場合、ロギング属性は表が存在する表領域の属性になります。

参照：

- この句の詳細は、8-34 ページの「logging_clause」を参照してください。
- ロギングおよびパラレル DML の詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。
- 「NOLOGGING モードでの索引の作成例 :」（14-68 ページ）

ONLINE ONLINE を指定すると、索引作成中の表での DML 操作を許可できます。

オンライン索引の作成の制限事項： オンライン索引の作成には、次の制限事項があります。

- オンライン索引の作成中は、パラレル DML はサポートされません。ONLINE を指定し、パラレル DML 文を発行すると、Oracle Database はエラーを戻します。
- ビットマップ索引またはクラスタ索引には、ONLINE を指定できません。
- UROWID 列の従来索引には、ONLINE を指定できません。
- 索引構成表の一意でない 2 次索引の場合、索引構成表内の索引キー列の数と論理 ROWID の主キー列の数の合計は、32 以下にする必要があります。論理 ROWID は、索引キーに含まれる列を除外します。

参照： オンライン索引の作成および再作成については、『Oracle Database 概要』を参照してください。

parallel_clause

parallel_clause を指定すると、索引の作成をパラレル化できます。

この句の詳細は、16-54 ページの「CREATE TABLE」の「parallel_clause」を参照してください。

Index Partitioning 句

global_partitioned_index 句および *local_partitioned_index* 句を使用すると、*index* をパーティション化できます。

ブロック・サイズが異なる表領域のパーティション化されたデータベース・エンティティの記憶域には、制限事項があります。これらの制限事項については、『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。

参照： 「パーティション索引の例」（14-70 ページ）

global_partitioned_index

global_partitioned_index を使用すると、索引のパーティション化がユーザー定義であり、基礎となる表と同一レベルでパーティション化されないことを指定できます。デフォルトでは、非パーティション索引はグローバル索引です。

グローバル索引には、レンジ・パーティション化またはハッシュ・パーティション化を実行できます。どちらの場合でも、パーティション・キー列に最大 32 列を指定できます。列リストをパーティション化する場合、索引の列リストの左の接頭辞を指定する必要があります。索引が列 a、b および c に定義されている場合は、列に (a, b, c)、(a, b) または (a, c) は指定できますが、(b, c)、(c) または (b, a) は指定できません。パーティション名を指定しない場合、SYS_Pn の形式でパーティション名が割り当てられます。

GLOBAL PARTITION BY RANGE この句を使用すると、レンジ・パーティション・グローバル索引を作成できます。列リストに指定した表の列の値の範囲に基づいて、グローバル索引がパーティション化されます。

> **参照 :** 「レンジ・パーティション・グローバル索引の作成例 :」(14-70 ページ)

GLOBAL PARTITION BY HASH この句を使用すると、ハッシュ・パーティション・グローバル索引を作成できます。パーティション・キー列の値にハッシュ・ファンクションを使用して、行がパーティションに割り当てられます。

> **参照 :** ハッシュ・パーティションの 2 つの方法については、16-47 ページの「CREATE TABLE」句の「hash_partitions」および 14-70 ページの「ハッシュ・パーティション・グローバル索引の作成例 :」を参照してください。

グローバル・パーティション索引の制限事項 : グローバル・パーティション索引には、次の制限事項があります。

- パーティション・キー列リストには、ROWID 類似列または ROWID 型の列は指定できません。
- ハッシュ・パーティションに指定できるプロパティは、表領域の記憶域のみです。そのため、*individual_hash_partitions* の *partitioning_storage_clause* に LOB または VARRAY の記憶域句を指定できません。
- *hash_partitions_by_quantity* の OVERFLOW 句は、索引構成表のパーティションに対してのみ指定できます。
- *partitioning_storage_clause* では、*table_compression* は指定できませんが、*key_compression* は指定できます。

> **注意 :** 異なるキャラクタ・セットを使用してデータベースを使用しているか、使用する予定がある場合は、キャラクタ列を分割する際に注意してください。文字のソート順序は、すべてのキャラクタ・セットで同一ではありません。

> **参照 :** キャラクタ・セットのサポートの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

index_partitioning_clause この句を使用すると、個々の索引パーティションを記述できます。この句が繰り返される数によってパーティションの数が決まります。*partition* を指定しない場合、名前は SYS_Pn の形式で生成されます。

VALUES LESS THAN(*value_list*) には、グローバル索引の現在のパーティションの境界は含まない上限を指定します。値のリストは、*global_partitioned_index* 句の列リストに対応するリテラル値を含む、カンマで区切られた順序リストです。最後のパーティションの値としては、必ず MAXVALUE を指定します。

注意： 索引が DATE 列でパーティション化されている場合、および日付書式で年の最初の 2 桁の数字が指定されていない場合、年の 4 文字書式マスクで TO_DATE ファンクションを使用する必要があります。日付書式は、NLS_TERRITORY によって暗黙的に決定され、NLS_DATE_FORMAT によって明示的に決定されます。これらの初期化パラメータの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

参照： 「レンジ・パーティション化の例 :」（16-65 ページ）

local_partitioned_index

local_partitioned_index 句を使用すると、*table* と同じパーティション数および同じパーティション境界を使用し、同じ列で索引をパーティション化できます。基礎となる表が再パーティション化された場合、ローカル索引のパーティションは自動的に保持されます。

on_range_partitioned_table この句を使用すると、レンジ・パーティション表の索引パーティションの名前および属性を指定できます。この句を指定する場合、PARTITION 句は、表パーティションと同一の数と順序である必要があります。*partition* を指定しない場合、対応する表のパーティションと一貫した名前が生成されます。その名前が既存の索引パーティション名と競合する場合は、SYS_Pn の形式が使用されます。

索引にキー圧縮が指定されていない場合、索引パーティションにキー圧縮を指定することはできません。

on_list_partitioned_table *on_list_partitioned_table* 句は、14-64 ページの「on_range_partitioned_table」と同一です。

on_hash_partitioned_table この句を使用すると、ハッシュ・パーティション表の索引パーティションの名前および表領域の記憶域を指定できます。

PARTITION 句を指定する場合、これらの句の数は表パーティションの数と同一である必要があります。*partition* を指定しない場合、対応する表のパーティションと一貫した名前が生成されます。その名前が既存の索引パーティション名と競合する場合は、SYS_Pn の形式が使用されます。オプションで、1 つ以上の個々のパーティションに表領域の記憶域を指定できます。この句または STORE IN 句に表領域の記憶域を指定しない場合、各索引パーティションが、対応する表パーティションと同じ表領域に格納されます。

STORE IN 句を使用して、すべての索引ハッシュ・パーティションを分散させる 1 つ以上の表領域を指定できます。表領域の数は、索引パーティションの数と等しくなる必要はありません。索引パーティションの数が表領域の数より多い場合は、表領域名が繰り返し使用されます。

on_comp_partitioned_table この句を使用すると、コンポジット・パーティション表の索引パーティションの名前および属性を指定できます。

STORE IN 句は、レンジ - ハッシュまたはリスト - ハッシュ・コンポジット・パーティション表に対してのみ有効です。この句を使用して、すべてのパーティションのすべての索引ハッシュ・サブパーティションを分散させる 1 つ以上のデフォルト表領域を指定できます。*index_subpartition_clause* の第 2 STORE IN 句で個々のパーティションのサブパーティションに対して異なるデフォルトの表領域の記憶域を指定すると、この記憶域を上書きできます。

レンジ - レンジ、レンジ - リストおよびリスト - リスト・コンポジット・パーティション表に対しては、PARTITION 句に指定したレンジまたはリスト・サブパーティションのデフォルトの属性を指定できます。*index_subpartition_clause* の SUBPARTITION 句の個々のパーティションのレンジまたはリスト・サブパーティションに異なる属性を指定すると、この記憶域を上書きできます。

索引にキー圧縮が指定されていない場合、索引パーティションにキー圧縮を指定することはできません。

index_subpartition_clause この句を使用すると、コンポジット・パーティション表の索引サブパーティションに名前および表領域の記憶域を指定できます。

STORE IN 句は、レンジ - ハッシュおよびリスト - ハッシュ・コンポジット・パーティション表のハッシュ・サブパーティションに対してのみ有効です。この句を使用して、すべての索引ハッシュ・サブパーティションを分散させる 1 つ以上の表領域を指定できます。
SUBPARTITION 句は、すべてのサブパーティション・タイプに対して有効です。

SUBPARTITION 句を指定する場合、これらの句の数は表サブパーティションの数と同一である必要があります。*subpartition* を指定しない場合、対応する表のサブパーティションと一貫した名前が生成されます。その名前が既存の索引サブパーティション名と競合する場合は、SYS_SUBP*n* の形式が使用されます。

表領域の数は、索引サブパーティションの数と等しくなる必要はありません。索引サブパーティションの数が表領域の数より多い場合は、表領域名が繰り返し使用されます。

on_comp_partitioned_table 句または *index_subpartition_clause* にサブパーティションの表領域の記憶域を指定しない場合、*index* に指定された表領域が使用されます。*index* に表領域の記憶域を指定しない場合、サブパーティションが、対応する表サブパーティションと同じ表領域に格納されます。

domain_index_clause

index が、アプリケーション固有の *indextype* 索引タイプのインスタンスであるドメイン索引であることを指定します。

ドメイン索引を作成する前に、複数の操作を行う必要があります。まず、索引タイプの実装タイプを作成します。また、ファンクション実装を作成し、そのファンクションを使用する演算子も作成します。次に、演算子と実装タイプを関連付ける索引タイプを作成します。最後にこの句を使用してドメイン索引を作成します。付録 E「詳細な例」に、これらのすべての操作を含んだ単純なドメイン索引の作成例を記載しています。

index_expr *index_expr*（*table_index_clause* 内）に、索引が定義されている表の列またはオブジェクトの属性を指定します。基礎となる索引タイプが異なり、その索引タイプがユーザー定義操作の分割セットをサポートする場合のみ、1 つの列に複数のドメイン索引を定義できます。

ドメイン索引の制限事項： ドメイン索引には、次の制限事項があります。

- *index_expr*（*table_index_clause* 内）には 1 つの列のみ指定でき、データ型が REF、VARRAY、ネストした表、LONG または LONG RAW の列は指定できません。
- ビットマップ索引または一意ドメイン索引は作成できません。
- 一時表には、ドメイン索引は作成できません。

indextype 索引タイプの名前を指定します。名前は、作成済の有効なスキーマ・オブジェクトである必要があります。

Oracle Text をインストールしている場合、様々な組込み索引タイプを使用して、Oracle Text ドメイン索引を作成できます。Oracle Text および Oracle Text が使用する索引の詳細は、『Oracle Text リファレンス』を参照してください。

参照： 「CREATE INDEXTYPE」（14-73 ページ）

local_domain_index_clause この句を使用すると、パーティション表のローカル索引を索引にするように指定できます。

- PARTITIONS 句を使用すると、索引パーティションの名前を指定できます。指定するパーティション数は、実表内のパーティション数と一致する必要があります。この句を省略すると、SYS_Pn という形式のシステム生成の名前でパーティションが作成されます。
- PARAMETERS 句を使用すると、個々のパーティション固有のパラメータ文字列を指定できます。この句を省略した場合、索引に関連付けられたパラメータ文字列はパーティションにも関連付けられます。

parallel_clause *parallel_clause* を使用すると、ドメイン索引の作成をパラレル化できます。非パーティション・ドメイン索引の場合、Oracle Database は明示的またはデフォルトの並列度を ODCIIndexCreate カートリッジ・ルーチンに渡し、ODCIIndexCreate カートリッジ・ルーチンが索引の並列性を確立します。ローカル・ドメイン索引の場合、この句によって索引パーティションはパラレルに作成されます。

> **参照：** Oracle Data Cartridge Interface（ODCI）ルーチンの詳細は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。

PARAMETERS 未解析のまま適切な ODCI 索引タイプ・ルーチンに渡されたパラメータ文字列を指定します。パラメータ文字列の最大長は 1,000 文字です。

構文の最上位でこの句を指定した場合、パラメータは索引パーティションのデフォルトのパラメータになります。*local_domain_index_clause* の一部としてこの句を指定すると、個々のパーティションのパラメータでデフォルトのパラメータを上書きできます。

ドメイン索引が作成されると、適切な ODCI ルーチンがコールされます。ルーチンが正常に戻らない場合、ドメイン索引は FAILED のマークが付けられます。失敗したドメイン索引でサポートされる操作は、DROP INDEX および REBUILD INDEX（非ローカル索引用）のみです。

> **参照：** Oracle Data Cartridge Interface（ODCI）ルーチンの詳細は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。

XMLIndex_clause

XMLIndex_clause を使用すると、一般には XML データが含まれる列に、XMLIndex 索引を定義できます。XMLIndex 索引は、特に XML データのドメイン用に設計されたドメイン索引の一種です。

PARAMETERS PARAMETERS 句を使用すると、パス表に関する情報および XMLIndex のコンポーネントに対応する 2 次索引に関する情報を指定できます。パラメータ文字列の最大長は 1,000 文字です。

構文の最上位でこの句を指定した場合、パラメータは索引のパラメータおよび索引パーティションのデフォルトのパラメータになります。*local_xmlindex_clause* 句の一部としてこの句を指定すると、個々のパーティションのパラメータでデフォルトのパラメータを上書きできます。

> **参照：** PARAMETERS 句の構文およびセマンティクスの詳細、および XMLIndex の使用の詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

bitmap_join_index_clause

bitmap_join_index_clause を使用すると、**ビットマップ結合索引**を定義できます。ビットマップ結合索引は、単一の表に定義します。ディメンション表の列で構成される索引キーには、そのキーに対応するファクト表の ROWID が格納されます。データ・ウェアハウス環境では、一般的に、索引を定義する表を**ファクト表**といい、ファクト表と結合した表を**ディメンション表**といいます。ただし、結合索引の作成にはスター・スキーマは必須ではありません。

ON ON 句には、まずファクト表を指定し、次に索引を定義するディメンション表の列をカッコ内に指定します。

FROM FROM 句には、結合した表を指定します。

WHERE WHERE 句には、結合条件を指定します。

基礎となるファクト表がパーティション化されている場合、*local_partitioned_index* 句（14-64 ページの「local_partitioned_index」を参照）のいずれかを指定する必要があります。

ビットマップ結合索引の制限事項： 一般的なビットマップ索引の制限事項（14-56 ページの「BITMAP」を参照）に加え、ビットマップ結合索引には次の制限事項があります。

- 一時表にはビットマップ結合索引は作成できません。
- FROM 句で表を 2 回指定できません。
- ファンクション結合索引は作成できません。
- ディメンション表の列は、主キー列であるか、または一意制約を含む必要があります。
- ディメンション表が複合主キーを含む場合、主キーの各列は結合の一部である必要があります。
- ファクト表がパーティション化されていない場合は、*local_index_clauses* を指定できません。

> **参照：** ファクト表とディメンション表、およびデータ・ウェアハウス環境でのビットマップ索引の使用については、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

UNUSABLE

UNUSABLE を指定すると、UNUSABLE 状態で索引を作成できます。使用禁止の索引を使用可能にする場合、再構築するか、または削除して再作成する必要があります。

索引がパーティション化されている場合は、すべての索引パーティションに UNUSABLE のマークが付けられます。後で一部の索引パーティションのみを再作成して USABLE にすることができます。これは、一部の索引パーティションのみで索引を保持する場合に有効です。たとえば、新しいパーティションでは索引アクセスを有効にするものの、古いパーティションでは有効にしない場合などに有効です。

索引または索引の一部のパーティションまたはサブパーティションに UNUSABLE のマークが付けられている場合は、次の状況でのみ、索引はオプティマイザによってアクセス・パスとみなされます。オプティマイザはどのパーティションがアクセスされるかをコンパイル時に認識している必要があり、アクセスされるすべてのパーティションに USABLE のマークが付けられている必要があります。そのため、問合せにバインド変数を含めることはできません。

索引への使用禁止のマーク付けの制限事項： この句は、一時表の索引に対して指定できません。

例

一般的な索引の例

索引の作成例： 次の文は、サンプル表 oe.orders の customer_id 列にサンプル索引 ord_customer_ix を作成します。

```
CREATE INDEX ord_customer_ix
   ON orders (customer_id);
```

索引の圧縮例： 次の文は、COMPRESS 句を使用して索引 ord_customer_ix_demo を作成します。

```
CREATE INDEX ord_customer_ix_demo
   ON orders (customer_id, sales_rep_id)
   COMPRESS 1;
```

索引は、customer_id 列値の繰返し項目を圧縮します。

NOLOGGING モードでの索引の作成例： サンプル表 orders が高速パラレル・ロードで作成された場合（すべての行がソート済である場合）、次の文は、迅速に索引を作成します。

```
/* Unless you first sort the table oe.orders, this example fails
   because you cannot specify NOSORT unless the base table is
   already sorted.
*/
CREATE INDEX ord_customer_ix_demo
   ON orders (order_mode)
   NOSORT
   NOLOGGING;
```

クラスタ索引の作成例： 次の文は、personnel クラスタ（14-7 ページの「クラスタの作成例：」で作成）に対して索引を作成します。

```
CREATE INDEX idx_personnel ON CLUSTER personnel;
```

クラスタ・キーのすべての列にクラスタ索引が自動的に作成されるため、索引列は指定しません。クラスタ索引の場合は、すべての行に索引が付きます。

XMLType 表の索引の作成例： 次の文は、xwarehouses 表（16-64 ページの「XMLType 表の例：」で作成）の領域要素に索引を作成します。

```
CREATE INDEX area_index ON xwarehouses e
   (EXTRACTVALUE(VALUE(e),'/Warehouse/Area'));
```

この索引によって、たとえば、次の文にあるような倉庫の面積（平方フィート）に基づいた表から選択する問合せのパフォーマンスが大幅に向上します。

```
SELECT e.getClobVal() AS warehouse
   FROM xwarehouses e
   WHERE EXISTSNODE(VALUE(e),'/Warehouse[Area>50000]') = 1;
```

> **参照：** 5-61 ページの「EXISTSNODE」および 5-223 ページの「VALUE」を参照してください。

ファンクション索引の例

次の例では、ファンクション索引を作成および使用する方法を示します。

ファンクション索引の作成例： 次の文は、last_name 列の大文字評価に基づく employees 表にファンクション索引を作成します。

```
CREATE INDEX upper_ix ON employees (UPPER(last_name));
```

ファンクション索引の作成に必要な権限およびパラメータ設定の詳細は、14-50 ページの「前提条件」を参照してください。

Oracle Database が全表スキャンを実行するのではなく、索引を使用するようにするには、ファンクションが戻す値を後続の問合せで NULL 以外にします。たとえば、次の文では、必ず索引が使用されます。

```
SELECT first_name, last_name
   FROM employees WHERE UPPER(last_name) IS NOT NULL
   ORDER BY UPPER(last_name);
```

WHERE 句を指定しないと、全表スキャンが実行される場合があります。

索引の作成および後続の問合せを示す次の文では、問合せで列の順序が逆であっても、Oracle Database は income_ix を使用します。

```
CREATE INDEX income_ix
   ON employees(salary + (salary*commission_pct));

SELECT first_name||' '||last_name "Name"
   FROM employees
   WHERE (salary*commission_pct) + salary > 15000
   ORDER BY employee_id;
```

LOB 列のファンクション索引の作成例： 次の文は、text_length ファンクションを使用し、サンプル・スキーマ pm の LOB 列にファンクション索引を作成します。このファンクションを作成する例については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。例では、CLOB 列が 1000 字未満のサンプル表 print_media から行を検索します。

```
CREATE INDEX src_idx ON print_media(text_length(ad_sourcetext));

SELECT product_id FROM print_media
   WHERE text_length(ad_sourcetext) < 1000
   ORDER BY product_id;

PRODUCT_ID
----------
      2056
      2268
      3060
      3106
```

型メソッドのファンクション索引の作成例： この例には、2 つの数値属性（length および width）を含むオブジェクト型 rectangle が必要です。area() メソッドは、四角形の面積を計算します。

```
CREATE TYPE rectangle AS OBJECT
( length   NUMBER,
  width    NUMBER,
  MEMBER FUNCTION area RETURN NUMBER DETERMINISTIC
);

CREATE OR REPLACE TYPE BODY rectangle AS
  MEMBER FUNCTION area RETURN NUMBER IS
  BEGIN
```

```
   RETURN (length*width);
  END;
END;
```

rectangle 型の表 rect_tab を作成する場合、次のように area() メソッドにファンクション索引を作成できます。

```
CREATE TABLE rect_tab OF rectangle;
CREATE INDEX area_idx ON rect_tab x (x.area());
```

この索引を効率的に使用して、次の形式の問合せを評価できます。

```
SELECT * FROM rect_tab x WHERE x.area() > 100;
```

ファンクション索引の使用による条件付き一意性の定義例： 次の文は、oe.orders 表に一意なファンクション索引を作成して、プロモーション ID 2（大規模なセール）を顧客が 2 回以上利用できないようにします。

```
CREATE UNIQUE INDEX promo_ix ON orders
   (CASE WHEN promotion_id =2 THEN customer_id ELSE NULL END,
    CASE WHEN promotion_id = 2 THEN promotion_id ELSE NULL END);

INSERT INTO orders (order_id, order_date, customer_id, order_total, promotion_id)
   VALUES (2459, systimestamp, 106, 251, 2);
1 row created.

INSERT INTO orders (order_id, order_date, customer_id, order_total, promotion_id)
   VALUES (2460, systimestamp+1, 106, 110, 2);
insert into orders (order_id, order_date, customer_id, order_total, promotion_id)
*
ERROR at line 1:
ORA-00001: unique constraint (OE.PROMO_IX) violated
```

promotion_id が 2 以外の行を索引から削除するようにします。Oracle Database では、すべてのキーが NULL の行は索引内に格納されません。したがって、この例では、promotion_id が 2 である場合以外は、customer_id と promotion_id の両方が NULL にマップされます。その結果、customer_id 値が同じ 2 つの行について promotion_id が 2 の場合にのみ、索引の制約に違反することになります。

パーティション索引の例

レンジ・パーティション・グローバル索引の作成例： 次の文は、コストの範囲を 3 つのグループに分割した 3 つのパーティションを含むサンプル表 sh.sales にグローバル同一キー索引 cost_ix を作成します。

```
CREATE INDEX cost_ix ON sales (amount_sold)
   GLOBAL PARTITION BY RANGE (amount_sold)
      (PARTITION p1 VALUES LESS THAN (1000),
       PARTITION p2 VALUES LESS THAN (2500),
       PARTITION p3 VALUES LESS THAN (MAXVALUE));
```

ハッシュ・パーティション・グローバル索引の作成例： 次の文は、4 つのパーティションを持つサンプル表 sh.customers に、ハッシュ・パーティション・グローバル索引 cust_last_name_ix を作成します。

```
CREATE INDEX cust_last_name_ix ON customers (cust_last_name)
  GLOBAL PARTITION BY HASH (cust_last_name)
  PARTITIONS 4;
```

ハッシュ・パーティション表の索引の作成例： 次の文は、hash_products パーティション表（16-68 ページの「ハッシュ・パーティション化の例：」で作成）の product_id 列にローカル索引を作成します。LOCAL の直後に記述された STORE IN 句は、hash_products がハッシュ・パーティション化されていることを示します。Oracle Database は、tbs1 表領域と tbs2 表領域にハッシュ・パーティションを分散させます。

```
CREATE INDEX prod_idx ON hash_products(category_id) LOCAL
   STORE IN (tbs_01, tbs_02);
```

索引を作成するには、指定する表領域に対して割当て制限を持つ必要があります。表領域 tbs_1 および tbs_2 の作成例については、16-71 ページの「CREATE TABLESPACE」を参照してください。

コンポジット・パーティション表の索引の作成例： 次の文は、composite_sales 表（16-68 ページの「コンポジット・パーティション表の例：」で作成）にローカル索引を作成します。STORAGE 句では、索引のデフォルトの記憶域属性を指定します。ただし、別の TABLESPACE 記憶域が指定されているため、このデフォルトは、パーティション q3_2000 および q4_2000 の 5 つのサブパーティションに上書きされます。

索引を作成するには、指定する表領域に対して割当て制限を持つ必要があります。表領域 tbs_1 および tbs_2 の作成例については、16-71 ページの「CREATE TABLESPACE」を参照してください。

```
CREATE INDEX sales_ix ON composite_sales(time_id, prod_id)
   STORAGE (INITIAL 1M MAXEXTENTS UNLIMITED)
   LOCAL
   (PARTITION q1_1998,
    PARTITION q2_1998,
    PARTITION q3_1998,
    PARTITION q4_1998,
    PARTITION q1_1999,
    PARTITION q2_1999,
    PARTITION q3_1999,
    PARTITION q4_1999,
    PARTITION q1_2000,
    PARTITION q2_2000
      (SUBPARTITION pq2001, SUBPARTITION pq2002,
       SUBPARTITION pq2003, SUBPARTITION pq2004,
       SUBPARTITION pq2005, SUBPARTITION pq2006,
       SUBPARTITION pq2007, SUBPARTITION pq2008),
    PARTITION q3_2000
      (SUBPARTITION c1 TABLESPACE tbs_02,
       SUBPARTITION c2 TABLESPACE tbs_02,
       SUBPARTITION c3 TABLESPACE tbs_02,
       SUBPARTITION c4 TABLESPACE tbs_02,
       SUBPARTITION c5 TABLESPACE tbs_02),
    PARTITION q4_2000
      (SUBPARTITION pq4001 TABLESPACE tbs_03,
       SUBPARTITION pq4002 TABLESPACE tbs_03,
       SUBPARTITION pq4003 TABLESPACE tbs_03,
       SUBPARTITION pq4004 TABLESPACE tbs_03)
);
```

ビットマップ索引の例

次の文は、表 oe.hash_products（16-68 ページの「ハッシュ・パーティション化の例：」で作成）にビットマップ結合索引を作成します。

```
CREATE BITMAP INDEX product_bm_ix
   ON hash_products(list_price)
   TABLESPACE tbs_1
   LOCAL(PARTITION ix_p1 TABLESPACE tbs_02,
         PARTITION ix_p2,
```

```
  PARTITION ix_p3 TABLESPACE tbs_03,
  PARTITION ix_p4,
  PARTITION ix_p5 TABLESPACE tbs_04 );
```

hash_products はパーティション表であるため、ビットマップ結合索引はローカル・パーティションである必要があります。この例では、指定する表領域に対して割当て制限を持つ必要があります。表領域 tbs_2、tbs_3 および tbs_4 を作成する例は、16-71 ページの「CREATE TABLESPACE」を参照してください。

ネストした表の索引の例 :

サンプル表 pm.print_media には、記憶表 textdocs_nestedtab に格納されたネストした表の列 ad_textdocs_ntab が含まれます。次の文は、記憶表 textdocs_nestedtab に一意索引を作成します。

```
CREATE UNIQUE INDEX nested_tab_ix
      ON textdocs_nestedtab(NESTED_TABLE_ID, document_typ);
```

疑似列 NESTED_TABLE_ID を組み込むことによって、ネストした表の列 ad_textdocs_ntab 内に固有の行が確保されます。

置換可能な列の索引の作成例 :

置換可能な列を宣言した型の属性に、索引を作成できます。また、適切な TREAT ファンクションの使用によって、サブタイプの属性を参照できます。次の例では、表 books（16-61 ページの「置換可能な表および列のサンプル :」で作成）を使用します。この文は、books 表の、すべての employee_t 型の author の salary 属性に索引を作成します。

```
CREATE INDEX salary_i
   ON books (TREAT(author AS employee_t).salary);
```

TREAT ファンクションの引数のターゲットとなる型は、参照する属性を追加した型である必要があります。例では、TREAT のターゲットは、employee_t で、salary 属性を追加した型です。

この条件を満たさない場合、TREAT ファンクションがファンクションの定義式として解析され、ファンクション索引が作成されます。たとえば、次の文は、パートタイム従業員の salary 属性にファンクション索引を作成し、型の階層に含まれる他のすべての型のインスタンスに NULL を割り当てます。

```
CREATE INDEX salary_func_i ON persons p
   (TREAT(VALUE(p) AS part_time_emp_t).salary);
```

SYS_TYPEID ファンクションの使用によって、置換可能な列を基礎とする型判別式の列に索引を作成できます。

注意 : Oracle Database は型判別式の列を使用し、IS OF *type* 条件を含む問合せを評価します。型 ID 列のカーディナリティが通常低い場合、ビットマップ索引を作成することをお薦めします。

次の文は、books 表の author 列の型 ID にビットマップ索引を作成します。

```
CREATE BITMAP INDEX typeid_i ON books (SYS_TYPEID(author));
```

参照 :

- books 表を基礎とする型の階層の作成については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。
- 5-215 ページの「TREAT」、5-191 ページの「SYS_TYPEID」および 7-23 ページの「IS OF type 条件」を参照してください。

CREATE INDEXTYPE

用途

CREATE INDEXTYPE 文を使用すると、（アプリケーション固有の）ドメイン索引を管理するルーチンを指定するオブジェクトである**索引タイプ**を作成できます。索引タイプは、表、ビューおよび他のスキーマ・オブジェクトと同じネームスペースにあります。この文は、索引タイプ名を実装タイプに結合し、順番に索引タイプを実装するユーザー定義索引ファンクションおよびプロシージャを指定し、参照します。

> **参照：** 索引タイプの実装の詳細は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。

前提条件

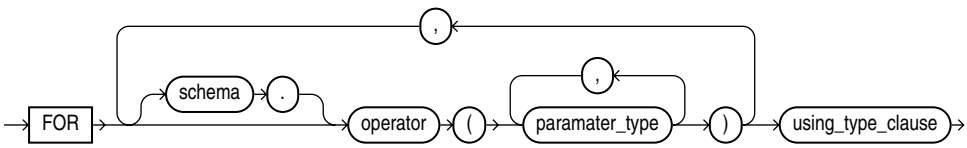
自分のスキーマに索引タイプを作成する場合は、CREATE INDEXTYPE システム権限が必要です。他のユーザーのスキーマ内に索引タイプを作成する場合は、CREATE ANY INDEXTYPE システム権限が必要です。どちらの場合も、実装タイプおよびサポートしている演算子に対する EXECUTE オブジェクト権限が必要です。

索引タイプは、1 つ以上の演算子をサポートしているため、索引タイプを作成する前に、サポートする演算子を設計し、これらの演算子に機能的な実装を提供する必要があります。

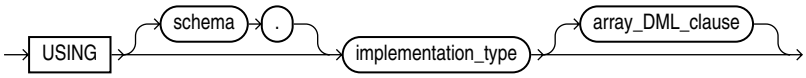
> **参照：** 「CREATE OPERATOR」（15-32 ページ）

構文

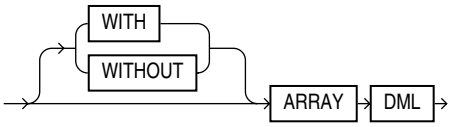
***create_indextype*::=**

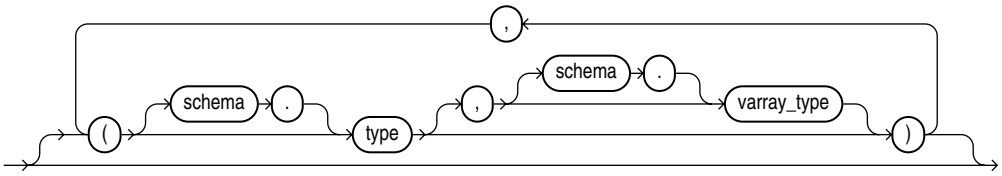


using_type_clause::=



array_DML_clause::=





storage_table_clause::=

セマンティクス

schema

索引タイプが存在するスキーマ名を指定します。*schema* を指定しない場合、自分のスキーマ内に索引タイプが作成されます。

indextype

作成する索引タイプの名前を指定します。

FOR 句

FOR 句を使用すると、索引タイプでサポートされる演算子のリストを指定できます。

- *schema* には、演算子が含まれているスキーマを指定します。*schema* を指定しない場合、その演算子は自分のスキーマにあるとみなされます。
- *operator* には、索引タイプによってサポートされる演算子の名前を指定します。

  この句に指定するすべての演算子は有効な演算子である必要があります。
- *parameter_type* には、演算子へのパラメータ・タイプを指定します。

using_type_clause

USING 句を使用すると、新しい索引タイプを実装するタイプを指定できます。

implementation_type には、適切な Oracle Data Cartridge Interface（ODCI）を実装するタイプ名を指定します。

- ODCI でルーチンを実装する有効なタイプを指定する必要があります。
- 実装タイプは、索引タイプと同じスキーマに存在する必要があります。

> **参照：** このインタフェースの詳細は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。

WITH LOCAL RANGE PARTITION

この句を使用すると、その索引タイプを使用してレンジ・パーティション表にローカル・ドメイン索引を作成するように指定できます。この句を省略すると、後でこの索引タイプを使用してパーティション表にローカル・ドメイン索引を作成することはできません。

storage_table_clause

この句を使用すると、この索引タイプに基づいて作成された索引の記憶表およびパーティション・メンテナンス操作の管理方法を指定できます。

- WITH SYSTEM MANAGED STORAGE TABLES を指定すると、統計データの格納がシステムで管理されます。*statistics_type* に指定するタイプによって、システムで保持される表に統計関連の情報が格納されます。また、指定する索引タイプはすでに登録済であるか、または WITH SYSTEM MANAGED STORAGE TABLES 句をサポートするように変更されている必要があります。
- WITH USER MANAGED STORAGE TABLES を指定すると、ユーザー定義の統計情報を格納する表は、ユーザーによって管理されます。これはデフォルトの動作です。

> **参照：** ドメイン索引の記憶表の詳細は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。

array_DML_clause

この句を使用すると、索引タイプで ODCIIndexInsert メソッドの配列インタフェースをサポートできるようになります。

type* および *varray_type 索引付けする列のデータ型がユーザー定義のオブジェクト型である場合、この句を指定して、Oracle が *type* の列値を保持するために使用する VARRAY の *varray_type* を識別する必要があります。索引タイプで型のリストがサポートされている場合、対応する VARRAY 型のリストを指定できます。*type* または *varray_type* で *schema* を省略した場合、型が自分のスキーマ内に定義されているとみなされます。

索引付けする列のデータ型が組込みシステム型である場合、その索引タイプに指定された VARRAY 型は、システムで定義された ODCI 型よりも優先されます。

> **参照：** ODCI 配列インタフェースの詳細は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。

例

索引タイプの作成例： 次の文は、position_indextype という名前の索引タイプを作成し、その索引タイプでサポートされている position_between 演算子、および索引インタフェースを実装する position_im タイプを指定します。この索引タイプを使用する拡張索引作成機能の使用例については、E-2 ページの「拡張索引作成機能の使用方法」を参照してください。

```
CREATE INDEXTYPE position_indextype
   FOR position_between(NUMBER, NUMBER, NUMBER)
   USING position_im;
```

CREATE JAVA

用途

CREATE JAVA を使用すると、Java ソース、クラスまたはリソースを含むスキーマ・オブジェクトを作成できます。

参照 :

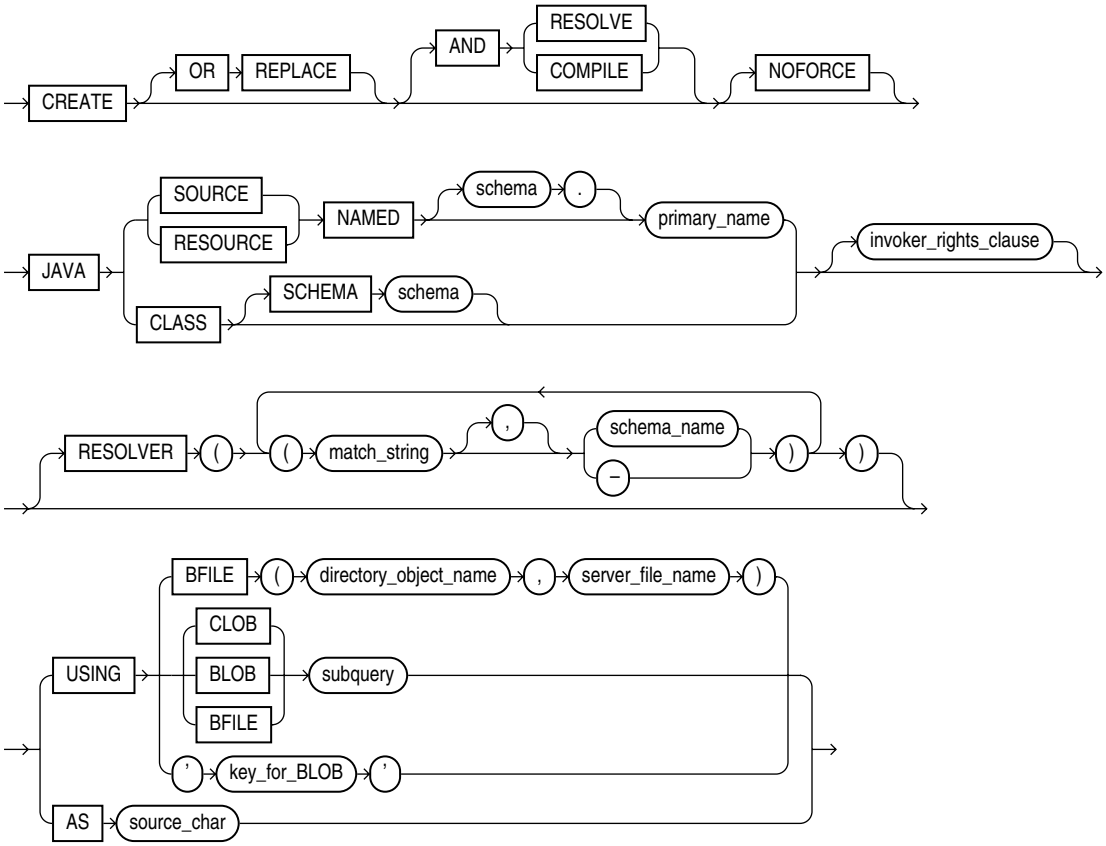
- Java の概要および Java ストアド・プロシージャの詳細は、『Oracle Database Java 開発者ガイド』を参照してください。
- JDBC については、『Oracle Database JDBC 開発者ガイドおよびリファレンス』を参照してください。

前提条件

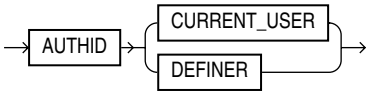
自分のスキーマに Java ソース、クラスまたはリソースを含むスキーマ・オブジェクトを作成または再作成する場合は、CREATE PROCEDURE システム権限が必要です。他のユーザーのスキーマ内にスキーマ・オブジェクトを作成または再作成する場合は、CREATE ANY PROCEDURE システム権限が必要です。

構文

***create_java*::=**



invoker_rights_clause::=



セマンティクス

OR REPLACE

OR REPLACE を指定すると、既存の Java クラス、ソースまたはリソースを含むスキーマ・オブジェクトを再作成できます。この句を指定した場合、付与されているオブジェクト権限を削除、再作成および再付与しなくても、既存のオブジェクトの定義を変更できます。

Java スキーマ・オブジェクトを再定義し、RESOLVE または COMPILE を指定した場合、そのオブジェクトが再コンパイルまたは変換されます。正常に変換またはコンパイルされたかどうかにかかわらず、Java スキーマ・オブジェクトを参照するクラスは有効になります。

再定義したファンクションに対して権限が付与されていたユーザーは、権限を再付与されなくても、そのファンクションにアクセスできます。

> **参照：** その他の情報は、10-85 ページの「ALTER JAVA」を参照してください。

RESOLVE | COMPILE

RESOLVE および COMPILE は、同義のキーワードです。この文が正常に実行された場合に作成される Java スキーマ・オブジェクトを変換することを指定します。

- クラスに適用された場合、他のクラス・スキーマ・オブジェクトに対する参照名に変換されます。
- ソースに適用された場合、ソースがコンパイルされます。

RESOLVE および COMPILE の制限事項： Java リソースには、これらのキーワードを指定できません。

NOFORCE

RESOLVE または COMPILE を指定しても、正常に変換またはコンパイルできない場合は、NOFORCE を指定すると CREATE コマンドの結果をロールバックできます。このオプションを指定しないと、正常に変換またはコンパイルできない場合でも何も処理が行われず、作成されたスキーマ・オブジェクトはそのままです。

JAVA SOURCE 句

JAVA SOURCE を指定すると、Java ソース・ファイルをロードできます。

JAVA CLASS 句

JAVA CLASS を指定すると、Java クラス・ファイルをロードできます。

JAVA RESOURCE 句

JAVA RESOURCE を指定すると、Java リソース・ファイルをロードできます。

NAMED 句

NAMED 句は、Java ソースまたはリソースの場合に指定します。*primary_name* は、二重引用符で囲む必要があります。

- Java ソースの場合、この句にはソース・コードが保持されているスキーマ・オブジェクト名を指定します。正常な CREATE JAVA SOURCE 文は、ソースによって定義された Java クラスをそれぞれ保持するために、追加スキーマ・オブジェクトも作成します。
- Java リソースの場合、この句には Java リソースを保持するスキーマ・オブジェクト名を指定します。

primary_name の小文字、または大文字と小文字の組合せを保持するには、二重引用符を使用します。

schema を指定しない場合、自分のスキーマ内にオブジェクトが作成されます。

NAMED Java クラスの制限事項 : NAMED 句には、次の制限事項があります。

- Java クラスには、NAMED を指定できません。
- *primary_name* は、データベース・リンクを含むことはできません。

SCHEMA 句

SCHEMA 句は、Java クラスにのみ適用されます。このオプションは、Java ファイルを含むオブジェクトが存在するスキーマを指定します。この句を指定しない場合、自分のスキーマ内にオブジェクトが作成されます。

invoker_rights_clause

invoker_rights_clause を使用すると、クラスを所有するユーザーのスキーマ内で、そのユーザーの権限を使用してクラスのメソッドが実行されるか、または、CURRENT_USER のスキーマ内で、そのユーザーの権限を使用してクラスのメソッドを実行するかを指定できます。

また、この句は、問合せ、DML 操作、およびその型のメンバー・ファンクションおよびプロシージャ内の動的 SQL 文の外部名の変換方法も定義します。

AUTHID CURRENT_USER

CURRENT_USER を使用すると、クラスのメソッドが CURRENT_USER 権限で実行されることを指定できます。この句はデフォルトで、**実行者権限クラス**を作成します。

また、この句は、問合せ、DML 操作、および動的 SQL 文の外部名を CURRENT_USER のスキーマで変換することも指定します。他のすべての文における外部名は、メソッドを含むスキーマで変換します。

AUTHID DEFINER

DEFINER を使用すると、クラスが存在するスキーマの所有者権限でクラスのメソッドを実行すること、およびクラスが存在するスキーマで外部名を変換することを指定できます。この句によって**定義者権限クラス**が作成されます。

参照 :

- 『Oracle Database Java 開発者ガイド』
- CURRENT_USER の判断方法については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

RESOLVER 句

RESOLVER 句を使用すると、Java スキーマ・オブジェクトに対する完全修飾 Java 名のマッピングを指定できます。ここでは次のとおりです。

- *match_string* は、完全修飾 Java 名、そのような Java 名と一致するワイルド・カードまたは任意の名前と一致するワイルド・カードを指定します。
- *schema_name* は、対応する Java スキーマ・オブジェクトを検索するスキーマを指定します。
- *schema_name* の代替としてのダッシュ（ - ）は、*match_string* が有効な Java 名と一致した場合、名前が変換されないことを示します。名前の変換は成功しますが、実行時にクラスがその名前を使用することはできません。

このマッピングは、後の変換で（暗黙的に、または ALTER JAVA ... RESOLVE 文で明示的に）使用されるコマンドで作成されるスキーマ・オブジェクトの定義とともに格納されます。

USING 句

USING は、Java クラスまたはリソースに対する文字データ（CLOB または BFILE）またはバイナリ・データ（BLOB または BFILE）の順序を決定します。文字の順序を使用して、1 つのファイルが Java クラスまたはリソースに、または 1 つのソース・ファイルおよび 1 つ以上の導出クラスが Java ソースに定義されます。

BFILE 句

順序を含む、オペレーティング・システム（*directory_object_name*）およびサーバー・ファイル（*server_file_name*）であらかじめ作成されているファイルのディレクトリおよびファイル名を指定します。BFILE は、通常、CREATE JAVA SOURCE によって文字順序として、CREATE JAVA CLASS または CREATE JAVA RESOURCE によってバイナリ順序として解析されます。

CLOB | BLOB | BFILE *subquery*

指定した型（CLOB、BLOB または BFILE）の行と列を選択する副問合せを指定します。列の値は文字列を構成します

> **注意：** 以前のリリースでは、USING 句は暗黙的にキーワード SELECT を提供しました。今回のリリースでは提供されません。ただし、キーワード SELECT なしの副問合せは、（下位互換性のために）サポートされています。

key_for_BLOB

key_for_BLOB 句は、次の暗黙的な問合せを提供します。

```
SELECT LOB FROM CREATE$JAVA$LOB$TABLE
   WHERE NAME = 'key_for_BLOB';
```

***key_for_BLOB* 句の制限事項：** このパラメータを使用する場合、表 CREATE$JAVA$LOB$TABLE が現行のスキーマ内にあり、BLOB 型の LOB 列および VARCHAR2 型の NAME 列が存在する必要があります。

AS *source_char*

Java ソースの文字列を指定します。

例

Java クラス・オブジェクトの作成例 : 次の文は、Java バイナリ・ファイルにある名前を使用して、Java クラスを含むスキーマ・オブジェクトを作成します。

```
CREATE JAVA CLASS USING BFILE (java_dir, 'Agent.class')
/
```

この例では、Java クラス Agent.class を含むオペレーティング・システム・ディレクトリを指すディレクトリ・オブジェクト java_dir がすでに存在していると想定しています。この例では、クラス名が Java クラス・スキーマ・オブジェクトの名前を決定します。

Java ソース・オブジェクトの作成例 : 次の文は、Java ソース・スキーマ・オブジェクトを作成します。

```
CREATE JAVA SOURCE NAMED "Welcome" AS
   public class Welcome {
      public static String welcome() {
         return "Welcome World";   } }
/
```

Java リソース・オブジェクトの作成例 : 次の文は、bfile から apptext という名前の Java リソース・スキーマ・オブジェクトを作成します。

```
CREATE JAVA RESOURCE NAMED "appText"
   USING BFILE (java_dir, 'textBundle.dat')
/
```

15

SQL 文 : CREATE LIBRARY ～ CREATE SPFILE

この章では、次の SQL 文について説明します。

- CREATE LIBRARY
- CREATE MATERIALIZED VIEW
- CREATE MATERIALIZED VIEW LOG
- CREATE OPERATOR
- CREATE OUTLINE
- CREATE PACKAGE
- CREATE PACKAGE BODY
- CREATE PFILE
- CREATE PROCEDURE
- CREATE PROFILE
- CREATE RESTORE POINT
- CREATE ROLE
- CREATE ROLLBACK SEGMENT
- CREATE SCHEMA
- CREATE SEQUENCE
- CREATE SPFILE

CREATE LIBRARY

用途

CREATE LIBRARY 文を使用すると、オペレーティング・システム共有ライブラリに関連するスキーマ・オブジェクトを作成できます。このスキーマ・オブジェクトの名前は、CREATE FUNCTION または CREATE PROCEDURE 文の *call_spec* で使用できます。また、パッケージまたはタイプにおけるファンクションまたはプロシージャを宣言する際にも使用できます。これによって、SQL および PL/SQL は、3GL ファンクションおよびプロシージャに対してコールできます。

> **参照：** ファンクションおよびプロシージャの詳細は、14-48 ページの「CREATE FUNCTION」および『Oracle Database PL/SQL 言語リファレンス』を参照してください。

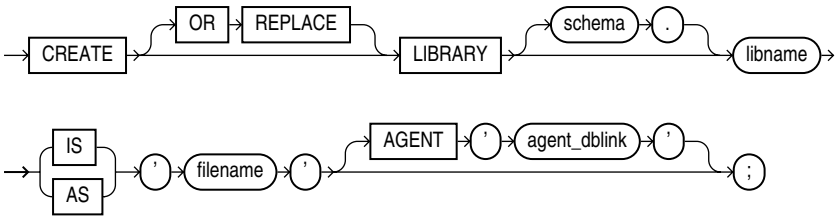
前提条件

自分のスキーマ内にライブラリを作成する場合は、CREATE LIBRARY システム権限が必要です。他のユーザーのスキーマ内にライブラリを作成する場合は、CREATE ANY LIBRARY システム権限が必要です。ライブラリに格納されているプロシージャおよびファンクションを使用する場合は、そのライブラリに対する EXECUTE オブジェクト権限が必要です。

CREATE LIBRARY 文は、共有ライブラリおよび動的リンクをサポートするプラットフォーム上でのみ有効です。

構文

***create_library*::=**



セマンティクス

OR REPLACE

OR REPLACE を指定すると、既存のライブラリを再作成できます。この句を指定した場合、既存のライブラリに付与されているスキーマ・オブジェクト権限を削除、再作成および再付与しなくても、ライブラリの定義を変更できます。

再定義したライブラリに対して権限を付与されていたユーザーは、権限を再付与されなくてもライブラリにアクセスできます。

libname

作成するライブラリの名前を指定します。*call_spec* でファンクションまたはプロシージャを宣言するときは、この名前を指定します。

filename

一重引用符で囲まれた文字列リテラルを指定します。文字列には、オペレーティング・システムの共有ライブラリの名前となるパスまたはファイル名を指定します。

filename は、CREATE LIBRARY 文の実行中は解析されません。ライブラリ・ファイルの存在は、そのファイルからルーチンが実行されるまでチェックされません。

AGENT 句

AGENT 句を指定すると、サーバーではなく、データベース・リンクから外部プロシージャを実行できます。外部プロシージャの実行には、*agent_dblink* に指定したデータベース・リンクが使用されます。この句を指定しない場合、サーバーのデフォルト・エージェント（extproc）が、外部プロシージャを実行します。

例

ライブラリの作成例： 次の文は、ライブラリ ext_lib を作成します。

```
CREATE LIBRARY ext_lib AS '/OR/lib/ext_lib.so';
/
```

次の文は、ライブラリ ext_lib を再作成します。

```
CREATE OR REPLACE LIBRARY ext_lib IS '/OR/newlib/ext_lib.so';
/
```

外部プロシージャ・エージェントの指定例： 次の例は、ライブラリ app_lib を作成し、パブリック・データベース sales.hq.acme.com から外部プロシージャを起動することを指定します。

```
CREATE LIBRARY app_lib as '${ORACLE_HOME}/lib/app_lib.so'
   AGENT 'sales.hq.acme.example.com';
/
```

> **参照：** データベース・リンクの作成方法については、14-31 ページの「パブリック・データベース・リンクの定義例 :」を参照してください。

CREATE MATERIALIZED VIEW

用途

CREATE MATERIALIZED VIEW 文を使用すると、**マテリアライズド・ビュー**を作成できます。マテリアライズド・ビューは、問合せ結果を含むデータベース・オブジェクトです。問合せの FROM 句には、表、ビューおよびその他のマテリアライズド・ビューを指定できます。これらのオブジェクトをあわせて、**マスター表**（レプリケーション用語）または**ディテール表**（データ・ウェアハウス用語）といいます。このマニュアルでは、「マスター表」を使用します。マスター表が格納されているデータベースを**マスター・データベース**といいます。

注意： 下位互換性を保つために、MATERIALIZED VIEW のかわりにキーワード SNAPSHOT もサポートされています。

レプリケーションでは、マテリアライズド・ビューを使用すると、ローカル・ノード上にあるリモート・データのコピーのメンテナンスができます。コピーは、アドバンスト・レプリケーション機能によって更新可能となりますが、この機能がない場合は読取り専用です。マテリアライズド・ビューのデータを、表またはビューと同じように選択することができます。レプリケーション環境では、通常作成されるマテリアライズド・ビューは、**主キー**、**ROWID**、**オブジェクト**および**副問合せ**のマテリアライズド・ビューです。

> **参照：** レプリケーションのサポートに使用するマテリアライズド・ビューの詳細は、『Oracle Database アドバンスト・レプリケーション』を参照してください。

データ・ウェアハウスでは、通常作成されるマテリアライズド・ビューは、**マテリアライズド集計ビュー**、**単一表マテリアライズド集計ビュー**および**マテリアライズド結合ビュー**です。3つのマテリアライズド・ビューは、クエリー・リライトで使用できます。クエリー・リライトとは、マスター表に関して記述したユーザー要求を、1つ以上のマテリアライズド・ビューを含む同等の要求に変換するための最適化手法です。

> **参照：**
>
> - 「ALTER MATERIALIZED VIEW」（11-2 ページ）
> - データ・ウェアハウスのサポートに使用するマテリアライズド・ビューの詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

前提条件

マテリアライズド・ビューの作成に必要な権限は、ロールを介してではなく、直接付与する必要があります。

自分のスキーマ内にマテリアライズド・ビューを作成する場合は、次の条件に従う必要があります。

- CREATE MATERIALIZED VIEW システム権限と、CREATE TABLE または CREATE ANY TABLE のいずれかのシステム権限が必要です。
- 所有していないマテリアライズド・ビューのマスター表にアクセスする場合は、各表に対する SELECT オブジェクト権限または SELECT ANY TABLE システム権限が必要です。

他のユーザーのスキーマ内にマテリアライズド・ビューを作成する場合、次の条件に従う必要があります。

- CREATE ANY MATERIALIZED VIEW システム権限が必要です。
- マテリアライズド・ビューの所有者には、CREATE TABLE システム権限が必要です。スキーマ所有者が所有していないマテリアライズド・ビューの任意のマスター表（リモート・データベースに存在するマスター表など）、およびそのマスター表に定義された任意のマテリアライズド・ビュー・ログにアクセスするには、各表に対する SELECT オブジェクト権限、または SELECT ANY TABLE システム権限が必要です。

REFRESH ON COMMIT モードのマテリアライズド・ビューを作成する（ON COMMIT REFRESH 句を使用）場合は、前述の権限の他に、所有しないマスター表に対する ON COMMIT REFRESH オブジェクト権限、または ON COMMIT REFRESH システム権限が必要です。

前述の権限の他にも、**クエリー・リライトが使用可能**なマテリアライズド・ビューを作成する場合は、次の条件に従う必要があります。

- スキーマ所有者がマスター表を所有していない場合は、そのスキーマ所有者には GLOBAL QUERY REWRITE 権限、または自分のスキーマ以外の各表に対する QUERY REWRITE オブジェクト権限が必要です。
- 事前作成コンテナにマテリアライズド・ビューを定義する（ON PREBUILT TABLE 句を使用）場合は、コンテナ表に対する WITH GRANT OPTION 付きの SELECT 権限が必要です。

マテリアライズド・ビューを含むスキーマのユーザーには、マテリアライズド・ビューのマスター表および索引を格納するターゲット表領域への十分な割当て制限または UNLIMITED TABLESPACE システム権限が必要です。

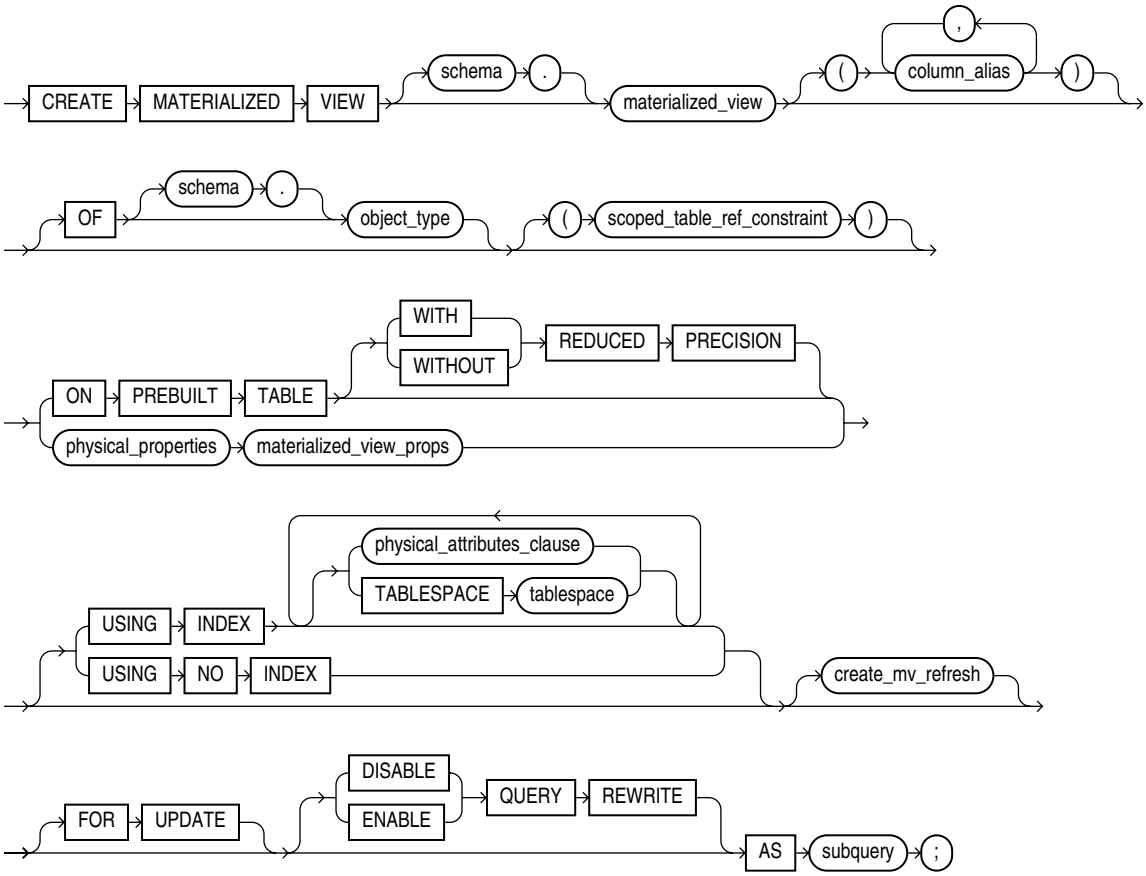
マテリアライズド・ビューを作成する場合、Oracle Database はマテリアライズド・ビューのスキーマ内に、1 つの内部表および 1 つ以上の索引を作成します。また、1 つのビューを作成することもあります。これらのオブジェクトは、マテリアライズド・ビューのデータをメンテナンスするために使用されます。ユーザーには、これらのオブジェクトを作成するための権限が必要です。

参照：

- これらの権限については、16-6 ページの「CREATE TABLE」、17-13 ページの「CREATE VIEW」および 14-50 ページの「CREATE INDEX」を参照してください。
- レプリケーションのためのマテリアライズド・ビューの作成についての前提条件は、『Oracle Database アドバンスト・レプリケーション』を参照してください。
- データ・ウェアハウスのためのマテリアライズド・ビューの作成についての前提条件は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

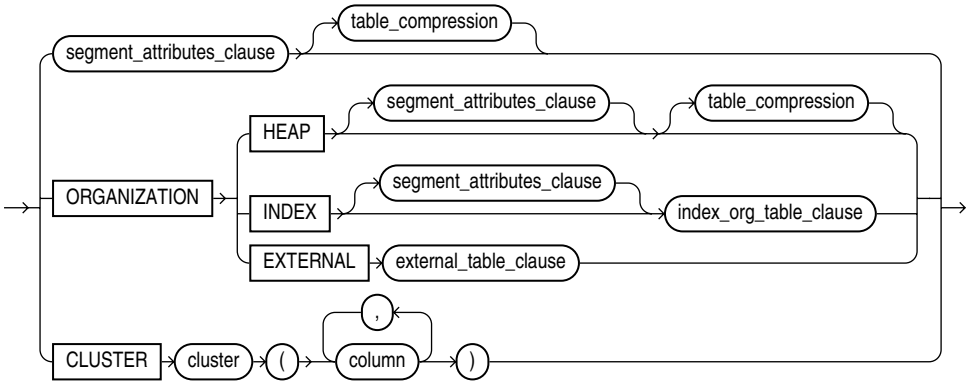
構文

***create_materialized_view*::=**



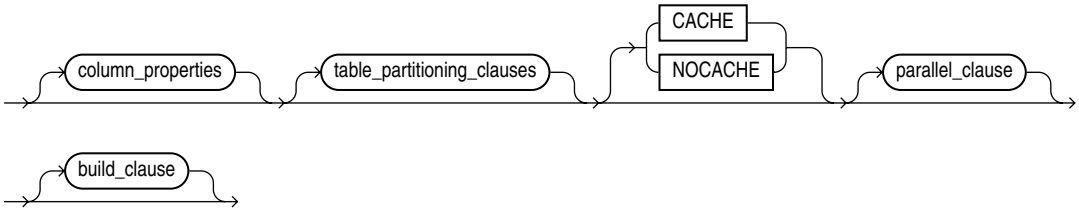
(15-6 ページの physical_properties::=、15-7 ページの scoped_table_ref_constraint::=、15-7 ページの materialized_view_props::=、15-8 ページの physical_attributes_clause::=、15-8 ページの create_mv_refresh::=、19-5 ページの subquery::= を参照)

***physical_properties*::=**



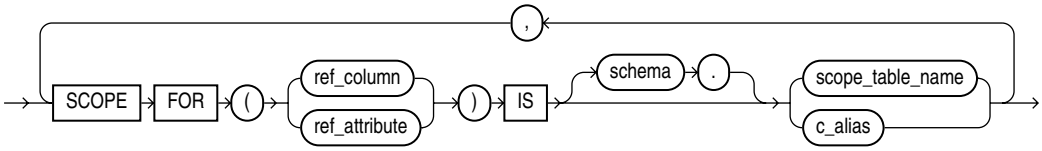
(15-8 ページの segment_attributes_clause::=、15-9 ページの table_compression::=、15-7 ページの index_org_table_clause::= を参照)

materialized_view_props::=

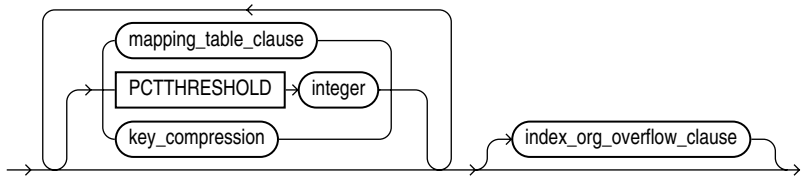


（15-9 ページの column_properties::=、16-17 ページの「CREATE TABLE」構文の項にある table_partitioning_clauses::=、15-11 ページの parallel_clause::=、15-12 ページの build_clause::= を参照）

scoped_table_ref_constraint::=

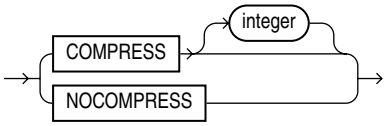


index_org_table_clause::=

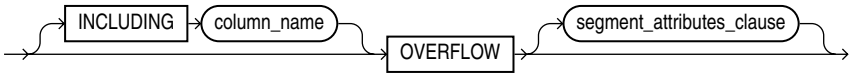


（*mapping_table_clause* は、マテリアライズド・ビューではサポートされていません。15-7 ページの key_compression::=、15-7 ページの index_org_overflow_clause::= を参照）

***key_compression*::=**

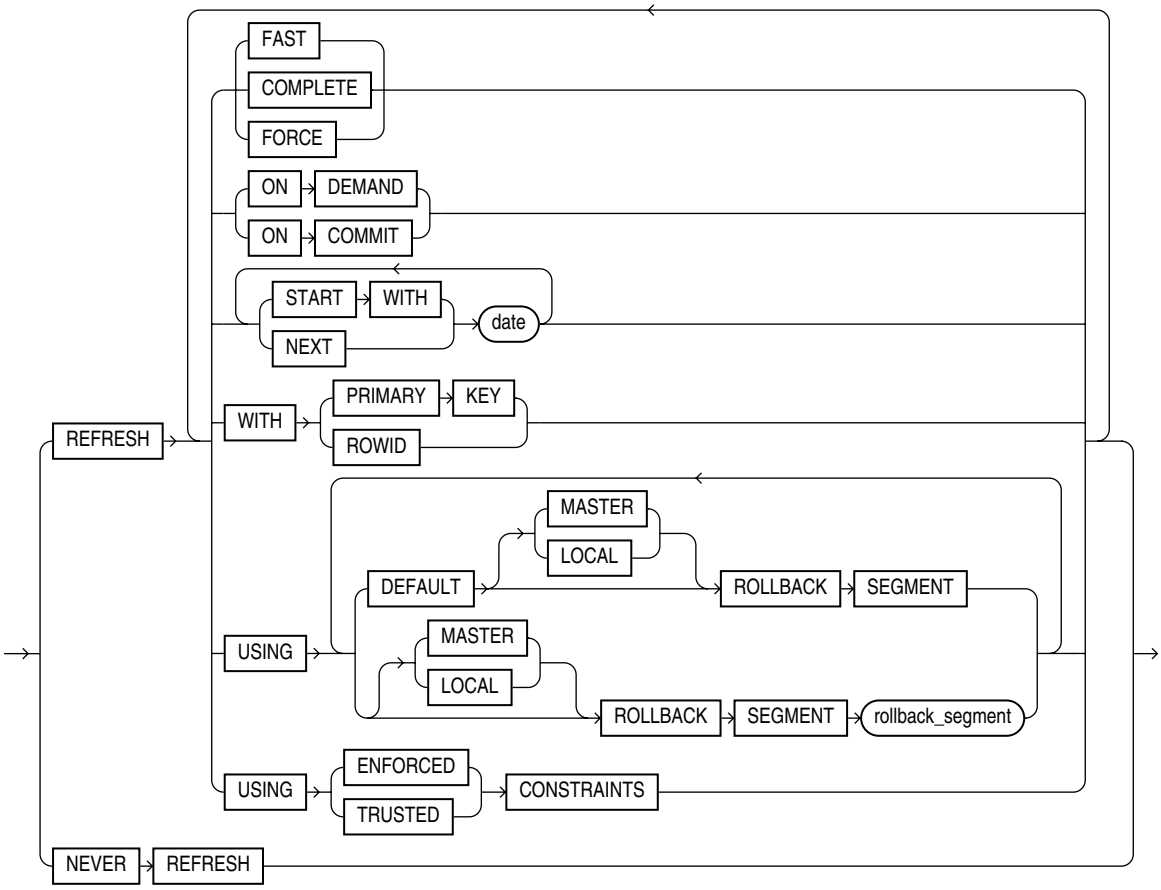


***index_org_overflow_clause*::=**

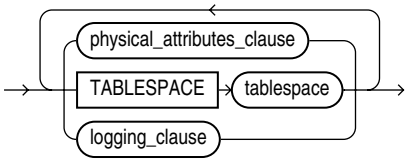


（15-8 ページの segment_attributes_clause::= を参照）

create_mv_refresh::=

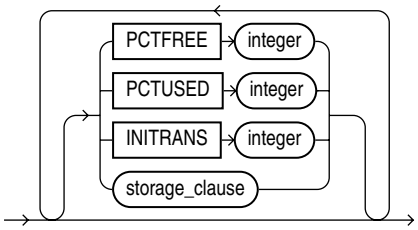


segment_attributes_clause::=



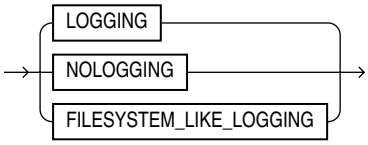
(15-8 ページの physical_attributes_clause::=、15-9 ページの logging_clause::= を参照)

***physical_attributes_clause*::=**

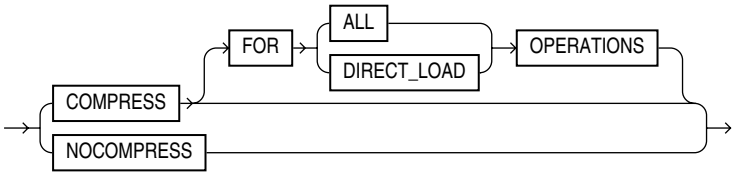


(8-34 ページの logging_clause::= を参照)

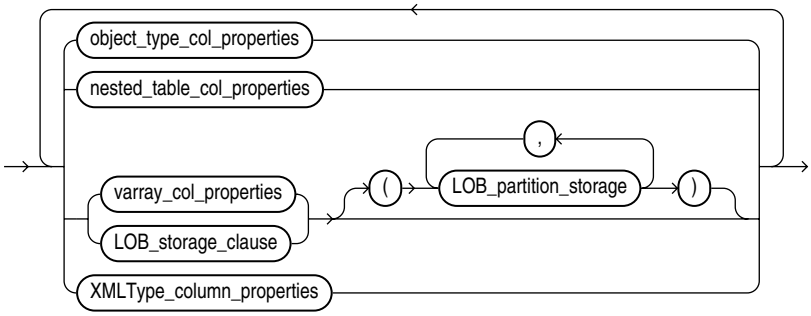
***logging_clause*::=**



table_compression::=

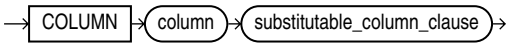


column_properties::=



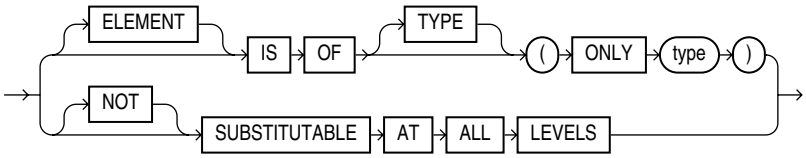
（15-9 ページの object_type_col_properties::=、15-10 ページの nested_table_col_properties::=、15-10 ページの varray_col_properties::=、15-11 ページの LOB_partition_storage::=、15-10 ページの LOB_storage_clause::= を参照。*XMLType_column_properties* は、マテリアライズド・ビューではサポートされていません。）

***object_type_col_properties*::=**

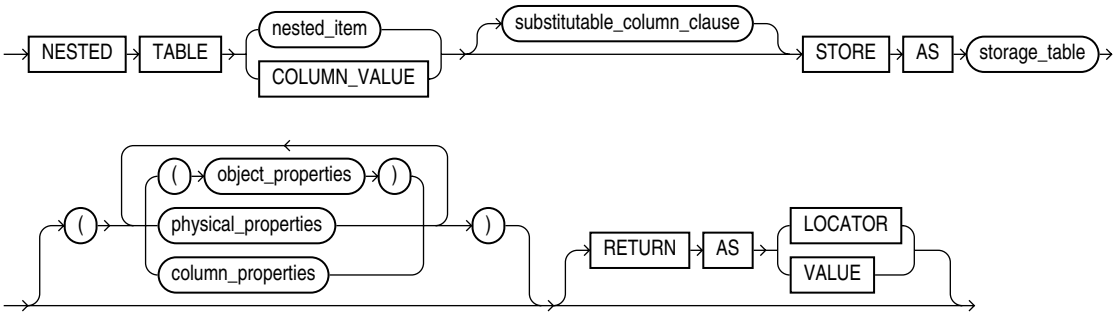


（15-9 ページの substitutable_column_clause::= を参照）

***substitutable_column_clause*::=**

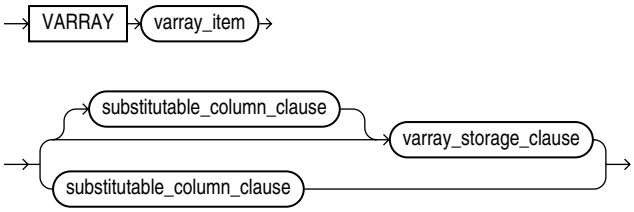


***nested_table_col_properties*::=**



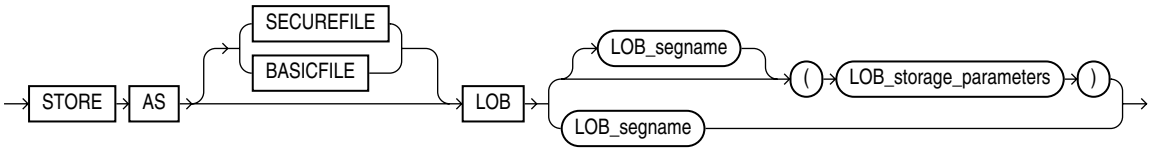
（15-9 ページの substitutable_column_clause::=、16-9 ページの object_properties::=、16-9 ページの「CREATE TABLE」構文の項にある physical_properties::=、15-9 ページの column_properties::= を参照）

***varray_col_properties*::=**



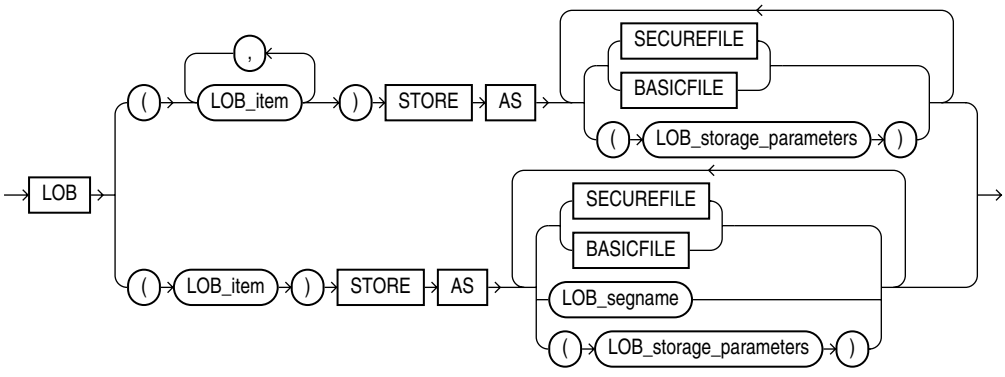
（15-9 ページの substitutable_column_clause::=、15-10 ページの varray_storage_clause::= を参照）

***varray_storage_clause*::=**



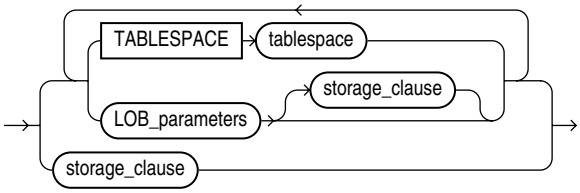
（15-11 ページの LOB_parameters::= を参照）

***LOB_storage_clause*::=**



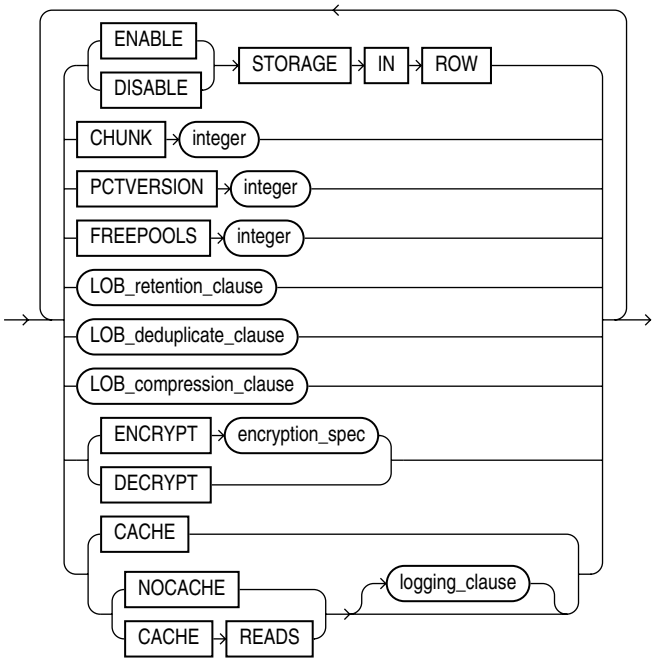
（15-11 ページの LOB_storage_parameters::= を参照）

***LOB_storage_parameters*::=**



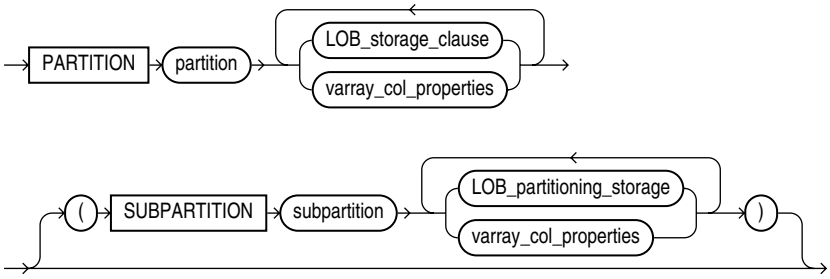
（15-11 ページの LOB_parameters::=、8-44 ページの storage_clause::= を参照）

***LOB_parameters*::=**



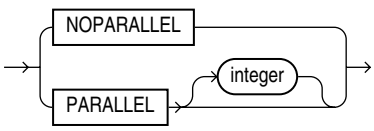
（8-44 ページの storage_clause::=、15-9 ページの logging_clause::= を参照）

***LOB_partition_storage*::=**

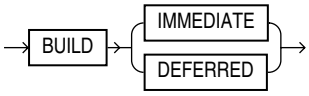


（15-10 ページの LOB_storage_clause::=、15-10 ページの varray_col_properties::= を参照）

parallel_clause::=



build_clause::=



セマンティクス

schema

マテリアライズド・ビューを含めるスキーマを指定します。*schema* を指定しないと、自分のスキーマ内にマテリアライズド・ビューが作成されます。

materialized_view

作成するマテリアライズド・ビューの名前を指定します。Oracle Database は、マテリアライズド・ビュー名に接頭辞または接尾辞を追加して、マテリアライズド・ビューをメンテナンスするための表および索引の名前を生成します。

column_alias

マテリアライズド・ビューの各列に対して別名を指定できます。列の別名のリストによって、競合する列名が明示的に解決されます。したがって、マテリアライズド・ビューの SELECT 句で別名を指定する必要がなくなります。この句で列の別名を指定する場合は、SELECT 句内で参照される各データ・ソースの別名を指定する必要があります。

OF *object_type*

OF *object_type* 句を指定すると、*object_type* 型の**オブジェクト・マテリアライズド・ビュー**を明示的に作成できます。

> **参照：** OF *type_name* 句の詳細は、16-7 ページの「CREATE TABLE」の「object_table」を参照してください。

scoped_table_ref_constraint

SCOPE FOR 句を使用すると、参照の有効範囲を 1 つのオブジェクト表に制限できます。*scope_table_name* を持つ表名または列の別名を参照できます。REF 列または属性の値は *scope_table_name* または *c_alias* 内のオブジェクトを指し、その場所に REF 列と同じ型のオブジェクト・インスタンスが格納されます。別名を指定する場合は、その別名が、マテリアライズド・ビューを定義する問合せの SELECT 構文のリストの列と 1 対 1 で対応する必要があります。

> **参照：** 詳細は、8-12 ページの「REF 列の有効範囲制約」を参照してください。

ON PREBUILT TABLE 句

ON PREBUILT TABLE 句を指定すると、既存の表を再初期化したマテリアライズド・ビューとして登録できます。この句は、データ・ウェアハウス環境において、大きいマテリアライズド・ビューを登録する場合に有効です。その表は、結果マテリアライズド・ビューと同じ名前で、同じスキーマにある必要があります。

マテリアライズド・ビューが削除されると、その既存の表は、1 つの表としての元の形に戻ります。

注意： この句は、表オブジェクトが副問合せの具体化を反映することを前提としています。マテリアライズド・ビューがそのマスター表のデータを正しく反映することを保証するために、この前提が満たされていることを確認することをお薦めします。

WITH REDUCED PRECISION WITH REDUCED PRECISION を指定すると、表またはマテリアライズド・ビュー列の精度が、副問合せで戻される精度と一致しない場合に、精度の低下を許可できます。

WITHOUT REDUCED PRECISION WITHOUT REDUCED PRECISION を指定すると、表またはマテリアライズド・ビュー列の精度が副問合せによって戻された精度と一致することを要求できます。一致しない場合、作成操作は中断します。これはデフォルトです。

事前作成表の使用の制限事項： 事前作成表には、次の制限事項があります。

- *subquery* の各列の別名は、事前作成表の列に対応し、対応する列のデータ型が一致している必要があります。
- この句を指定する場合、*subquery* で参照されない列にデフォルト値も指定しないかぎり、その列に NOT NULL 制約は指定できません。

参照： 「事前作成したマテリアライズド・ビューの作成例 :」(15-23 ページ)

physical_properties_clause

physical_properties_clause の構成要素は、マテリアライズド・ビューと表に対して同一のセマンティクスを持ちます。ただし、次の項で説明する例外および追加事項があります。

***physical_properties_clause* の制限事項：** マテリアライズド・ビューに対しては、ORGANIZATION EXTERNAL を指定できません。

segment_attributes_clause

segment_attributes_clause を使用すると、PCTFREE、PCTUSED、INITRANS パラメータの値、およびマテリアライズド・ビューの記憶特性の設定、表領域の割当て、ロギングが実行されるかどうかを指定できます。USING INDEX 句では、PCTFREE または PCTUSED は指定できません。

TABLESPACE 句 マテリアライズド・ビューを作成する表領域を指定します。この句を指定しないと、マテリアライズド・ビューを含むスキーマのデフォルト表領域内にマテリアライズド・ビューが作成されます。

参照： デフォルト値を含むこれらの句の詳細は、8-39 ページの「physical_attributes_clause」および 8-41 ページの「storage_clause」を参照してください。

logging_clause LOGGING または NOLOGGING を指定すると、マテリアライズド・ビュー・ログのロギング特性を設定できます。ロギング特性は、マテリアライズド・ビューの作成および DBMS_REFRESH パッケージによって開始される非アトミック・リフレッシュに影響します。デフォルトは、マテリアライズド・ビューが存在する表領域のロギング特性です。

参照： この句の詳細は、8-34 ページの「logging_clause」を参照してください。アトミック・リフレッシュおよび非アトミック・リフレッシュの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

table_compression

table_compression 句を使用すると、ディスクおよびメモリーの使用量を削減するために、データ・セグメントを圧縮するかどうかを指定できます。

参照： この句のセマンティクスの詳細は、16-31 ページの「CREATE TABLE」の「table_compression」を参照してください。

index_org_table_clause

ORGANIZATION INDEX を指定すると、索引構成マテリアライズド・ビューを作成できます。このマテリアライズド・ビューでは、データ行は、マテリアライズド・ビューの主キーに定義した索引に格納されます。次のようなマテリアライズド・ビューの索引構成を指定できます。

- 読取り専用および更新可能なオブジェクト・マテリアライズド・ビュー。マスター表に主キーが含まれている必要があります。
- 読取り専用および更新可能な主キー・マテリアライズド・ビュー。
- 読取り専用の ROWID マテリアライズド・ビュー。

index_org_table_clause のキーワードおよびパラメータは、CREATE TABLE と同じです。また、次の制限事項があります。

> **参照：** 詳細は、16-33 ページの「CREATE TABLE」の「index_org_table_clause」を参照してください。

索引構成マテリアライズド・ビューの制限事項： 索引構成マテリアライズド・ビューには、次の制限事項があります。

- CREATE MATERIALIZED VIEW の CACHE、NOCACHE、CLUSTER または ON PREBUILT TABLE 句は指定できません。
- *index_org_table_clause* には次の制限事項があります。
  - *mapping_table_clause* を指定できません。
  - 複合主キーに基づくマテリアライズド・ビューのみに COMPRESS を指定できます。単一主キーと複合主キーのいずれかに基づくマテリアライズド・ビューに NOCOMPRESS を指定できます。

CLUSTER 句

CLUSTER 句を指定すると、指定したクラスタの一部としてマテリアライズド・ビューを作成できます。クラスタ化マテリアライズド・ビューは、クラスタの領域割当てを使用します。したがって、CLUSTER 句で物理属性または TABLESPACE 句を指定しないでください。

クラスタ化マテリアライズド・ビューの制限事項： CLUSTER を指定する場合、*materialized_view_props* 内の *table_partitioning_clauses* は指定できません。

materialized_view_props

これらのプロパティ句を使用すると、既存の表に基づかないマテリアライズド・ビューを定義できます。既存の表に基づくマテリアライズド・ビューを作成するには、ON PREBUILT TABLE 句を使用します。

column_properties

column_properties 句を使用すると、LOB、ネストした表、VARRAY または XMLType の列の記憶特性を指定できます。*object_type_col_properties* は、マテリアライズド・ビューには関連しません。

> **参照：** この句のパラメータの指定の詳細は、16-6 ページの「CREATE TABLE」を参照してください。

table_partitioning_clauses

table_partitioning_clauses を使用すると、マテリアライズド・ビューを、指定した範囲の値またはハッシュ・ファンクションでパーティション化できます。マテリアライズド・ビューのパーティション化は、表のパーティション化と同じです

> **参照：** 16-44 ページの「CREATE TABLE」の「table_partitioning_clauses」を参照してください。

CACHE | NOCACHE

アクセス頻度の高いデータについて、CACHE は、全表スキャンの実行時にこの表に対して取り出された各ブロックを、バッファ・キャッシュの最低使用頻度（LRU）リストの最高使用頻度側に入れることを指定します。この属性は、小規模な参照表で有効です。NOCACHE は、ブロックを LRU リストの最低使用頻度側に入れることを指定します。

注意： NOCACHE は、*storage_clause* に KEEP を指定したマテリアライズド・ビューには、影響しません。

参照： CACHE または NOCACHE の指定の詳細は、16-6 ページの「CREATE TABLE」を参照してください。

parallel_clause

parallel_clause を使用すると、マテリアライズド・ビューへのパラレル操作をサポートするかどうかを指定できます。作成後にマテリアライズド・ビューに対する問合せおよび DML のデフォルトの並列度を設定します。

この句の詳細は、16-54 ページの「CREATE TABLE」の「parallel_clause」を参照してください。

build_clause

build_clause を指定すると、マテリアライズド・ビューをいつ移入するかを指定できます。

IMMEDIATE IMMEDIATE を指定すると、マテリアライズド・ビューにすぐに移入できます。これはデフォルトです。

DEFERRED DEFERRED を指定すると、次の REFRESH 操作でマテリアライズド・ビューに移入できます。最初の（遅延）リフレッシュは、常に、完全リフレッシュである必要があります。それ以前のマテリアライズド・ビューの値は UNUSABLE であるため、クエリー・リライトには使用できません。

USING INDEX 句

USING INDEX 句を使用すると、マテリアライズド・ビューのデータをメンテナンスするために使用されるデフォルトの索引の INITRANS パラメータおよび STORAGE パラメータの値を変更できます。USING INDEX が設定されていない場合、この索引にはデフォルト値が使用されます。デフォルトの索引は、マテリアライズド・ビューの増分リフレッシュ（高速リフレッシュ）を高速に処理するために使用されます。

USING INDEX 句の制限事項： この句では、PCTUSED パラメータは指定できません。

USING NO INDEX 句

USING NO INDEX 句を指定すると、デフォルトの索引の作成を抑制できます。CREATE INDEX 文を使用することによって、代替する索引を明示的に作成できます。USING NO INDEX を指定し、増分リフレッシュ（REFRESH FAST）でマテリアライズド・ビューを作成する場合は、このような索引を作成する必要があります。

create_mv_refresh

create_mv_refresh 句を使用すると、マテリアライズド・ビューのデフォルトのリフレッシュ方法、リフレッシュ・モードおよびリフレッシュ時刻を指定できます。マテリアライズド・ビューのマスター表が変更された場合、マテリアライズド・ビューのデータを更新し、その時点でマスター表にあるデータを正確に反映させる必要があります。この句によって、自動的にマテリアライズド・ビューをリフレッシュする日時をスケジューリングし、リフレッシュの方法およびモードを指定できます。

注意： この句では、デフォルトのリフレッシュ・オプションのみを設定します。リフレッシュを実際に実装する手順は、『Oracle Database アドバンスト・レプリケーション』および『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

参照：

- 15-23 ページの「マテリアライズド・ビューの定期的リフレッシュ例：」および 15-24 ページの「マテリアライズド・ビューの自動リフレッシュ時刻の例：」を参照してください。
- リフレッシュ方法の詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

FAST 句

FAST を指定すると、増分リフレッシュ方法を指定できます。これはマスター表に対して行った変更に従ってリフレッシュを行います。従来型 DML 変更の場合、変更は、マスター表に関連付けられたマテリアライズド・ビュー・ログに格納されます。ダイレクト・パス・インサート操作の場合、ダイレクト・ローダー・ログに格納されます。

REFRESH FAST を指定すると、マテリアライズド・ビューのマスター表のマテリアライズド・ビュー・ログが存在していない場合に、CREATE 文は正常に実行されません。ダイレクト・パス・インサートが行われると、ダイレクト・ローダー・ログが自動的に作成されます。手動での操作は必要ありません。

従来型 DML の変更の場合も、ダイレクト・パス・インサート操作の場合も、他の条件によって、高速リフレッシュへのマテリアライズド・ビューの適応性が制限されることがあります。

定義する問合せに分析ファンクションが含まれている場合、マテリアライズド・ビューは高速リフレッシュに適応しません。

参照：

- レプリケーション環境における高速リフレッシュの制限については、『Oracle Database アドバンスト・レプリケーション』を参照してください。
- データ・ウェアハウス環境における高速リフレッシュの制限については、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。
- 高速リフレッシュに関する問題の診断については、DBMS_MVIEW パッケージの EXPLAIN_MVIEW プロシージャを参照してください。高速リフレッシュの問題の修正については、DBMS_MVIEW パッケージの TUNE_MVIEW プロシージャを参照してください。
- 「分析ファンクション」（5-10 ページ）
- 「高速リフレッシュ可能なマテリアライズド・ビューの作成例：」（15-24 ページ）

COMPLETE 句

COMPLETE を指定すると、完全リフレッシュ方法を指定できます。これは、マテリアライズド・ビューを定義する問合せを実行することによって実装されます。完全リフレッシュを要求すると、高速リフレッシュが実行可能であっても、完全リフレッシュが実行されます。

FORCE 句

FORCE を指定すると、リフレッシュ時に、高速リフレッシュが可能な場合は高速リフレッシュを実行し、可能でない場合は完全リフレッシュを実行するように指定できます。リフレッシュ方法（FAST、COMPLETE または FORCE）を指定しないと、デフォルトで FORCE が指定されます。

ON COMMIT 句

ON COMMIT を指定すると、マテリアライズド・ビューのマスター表に対するトランザクションをコミットするときに、必ず高速リフレッシュが実行されるように指定できます。この句を指定すると、リフレッシュ操作がコミット処理の一部として行われるため、コミットの完了に時間がかかります。

ON COMMIT および ON DEMAND の両方を指定することはできません。ON COMMIT を指定した場合は、START WITH または NEXT を同時に指定できません。

ON COMMIT のリフレッシュの制限事項 :

- この句は、オブジェクト型または Oracle が提供する型を含むマテリアライズド・ビューについてはサポートしていません。
- この句を指定すると、それ以降は、このマテリアライズド・ビューのすべてのマスター表で分散トランザクションが実行できなくなります。たとえば、リモート表から選択してマスターに挿入することはできません。ON DEMAND 句の場合は、それ以降のマスター表での分散トランザクションに、このような制限はありません。

ON DEMAND 句

ON DEMAND を指定すると、3 つの DBMS_MVIEW リフレッシュ・プロシージャのうちのいずれかを使用してユーザーが手動でリフレッシュしないかぎりデータベースによってマテリアライズド・ビューがリフレッシュされないように指定できます。

ON COMMIT および ON DEMAND のどちらも指定しなかった場合、ON DEMAND がデフォルトになります。このデフォルト設定は、同じ CREATE MATERIALIZED VIEW 文または後続の ALTER MATERIALIZED VIEW 文のいずれかに START WITH 句または NEXT 句を指定することで上書きできます。

ON COMMIT および ON DEMAND の両方を指定することはできません。START WITH および NEXT は、ON DEMAND より優先されます。このため、START WITH または NEXT を指定したときは、ほとんどの場合、ON DEMAND を指定しても意味がありません。

参照 :

- これらのプロシージャの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。
- REFRESH ON DEMAND を指定することによって作成できるマテリアライズド・ビューのタイプについては、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

START WITH 句

最初の自動リフレッシュ時刻を表す日時式を指定します。

NEXT 句

自動リフレッシュの間隔を計算するための日時式を指定します。

START WITH 値および NEXT 値は、将来の時刻に評価される値です。START WITH 値を省略した場合、Oracle Database はマテリアライズド・ビューの作成時刻に対して NEXT 式を評価することによって、最初の自動リフレッシュ時刻を判断します。START WITH 値を指定し、NEXT 値を指定しない場合、Oracle Database は 1 回のみマテリアライズド・ビューをリフレッシュします。START WITH 値および NEXT 値のどちらも指定しない場合、または *create_mv_refresh* を指定しない場合は、マテリアライズド・ビューは自動リフレッシュされません。

WITH PRIMARY KEY 句

WITH PRIMARY KEY を指定すると、主キー・マテリアライズド・ビューを作成できます。これはデフォルトであり、WITH ROWID の項で説明する場合を除き、すべての場合に使用する必要があります。主キー・マテリアライズド・ビューを使用すると、高速リフレッシュに対するマテリアライズド・ビューの適応性に影響せずに、マテリアライズド・ビューのマスター表を再編成できます。マスター表には、使用可能な主キー制約が定義されている必要があり、マテリアライズド・ビューを定義する問合せでは、すべての主キー列が直接指定される必要があります。定義する問合せでは、UPPER などのファンクションへの引数として主キー列を指定できません。

主キー・マテリアライズド・ビューの制限事項： この句は、オブジェクト・マテリアライズド・ビューには指定できません。WITH OBJECT ID を指定してマテリアライズドされたオブジェクトは暗黙的にリフレッシュされます。

> **参照：** 主キー・マテリアライズド・ビューの詳細は、『Oracle Database アドバンスト・レプリケーション』および 15-23 ページの「主キー・マテリアライズド・ビューの作成例 :」を参照してください。

WITH ROWID 句

WITH ROWID を指定すると、ROWID マテリアライズド・ビューを作成できます。マテリアライズド・ビューがマスター表の主キー列をすべて含まない場合に、ROWID マテリアライズド・ビューは有効です。ROWID マテリアライズド・ビューは、単一表を基にしている必要があり、次のいずれも含むことができません。

- 固有ファンクションまたは集計ファンクション
- GROUP BY 句または CONNECT BY 句
- 副問合せ
- 結合
- 集合演算

完全リフレッシュが行われるまでは、マスター表の再編成後に、ROWID マテリアライズド・ビューは高速リフレッシュされません。

ROWID マテリアライズド・ビューの制限事項： この句は、オブジェクト・マテリアライズド・ビューには指定できません。WITH OBJECT ID を指定してマテリアライズドされたオブジェクトは暗黙的にリフレッシュされます。

> **参照：** 15-22 ページの「マテリアライズド集計ビューの作成例 :」および 15-23 ページの「ROWID マテリアライズド・ビューの作成例 :」を参照してください。

USING ROLLBACK SEGMENT 句

自動 UNDO モードではロールバック・セグメントではなく UNDO 表領域が使用されるため、データベースが自動 UNDO モードの場合、この句は無効です。自動 UNDO モードを使用することをお薦めします。この句は、ロールバック・セグメントが使用される以前のバージョンの Oracle Database が含まれるレプリケーション環境との下位互換性のためにサポートされています。

rollback_segment に、マテリアライズド・ビューのリフレッシュ中に使用するリモート・ロールバック・セグメントを指定します。

DEFAULT DEFAULT を使用すると、使用するロールバック・セグメントを自動的に選択できます。DEFAULT を指定した場合、*rollback_segment* は指定できません。DEFAULT は、マテリアライズド・ビューを（作成ではなく）変更する場合に有効です。

> **参照：** 「ALTER MATERIALIZED VIEW」（11-2 ページ）

MASTER MASTER を使用すると、個々のマテリアライズド・ビュー用のリモート・マスター・サイトで使用されるリモート・ロールバック・セグメントを指定できます。

LOCAL LOCAL を使用すると、マテリアライズド・ビューが含まれているローカル・リフレッシュ・グループで使用されるリモート・ロールバック・セグメントを指定できます。これはデフォルトです。

> **参照：** DBMS_REFRESH パッケージを使用するローカル・マテリアライズド・ビューのロールバック・セグメントの指定については、『Oracle Database アドバンスト・レプリケーション』を参照してください。

rollback_segment を指定しない場合、使用するロールバック・セグメントが自動的に選択されます。各マテリアライズド・ビューに対して 1 つのマスター・ロールバック・セグメントが格納され、マテリアライズド・ビューの作成およびリフレッシュ時に検証されます。複合マテリアライズド・ビューの場合、マスター・ロールバック・セグメントの指定は無視されます。

USING ...CONSTRAINTS 句

USING ... CONSTRAINTS 句を使用すると、リフレッシュ操作中に Oracle Database でより多くのリライト・オプションを選択でき、リフレッシュをより効果的に実行できます。リフレッシュ操作中に、適用される制約のみに依存するのではなく、適用されていない制約（RELY 状態のディメンションの関係や制約など）を使用できます。

> **注意：** USING TRUSTED CONSTRAINTS 句を指定すると、データベース管理者が信頼できると宣言したが、データベースで検証されていないディメンションおよび制約の情報を、Oracle Database で使用できます。ディメンションおよび制約の情報が有効であると、パフォーマンスを向上できる場合があります。ただし、この情報が無効であると、正常な状態が戻されても、リフレッシュ・プロシージャによってマテリアライズド・ビューが破損する場合があります。

この句を指定しない場合、USING ENFORCED CONSTRAINTS がデフォルトになります。

NEVER REFRESH 句

NEVER REFRESH を指定すると、Oracle Database のリフレッシュ・メカニズムまたはパッケージ・プロシージャを使用したマテリアライズド・ビューのリフレッシュを回避できます。このようなプロシージャから発行された、マテリアライズド・ビューに対する REFRESH 文は、無視されます。この句を無効にするには、ALTER MATERIALIZED VIEW ... REFRESH 文を発行する必要があります。

FOR UPDATE 句

FOR UPDATE を指定すると、副問合せ、主キー、オブジェクトまたは ROWID マテリアライズド・ビューを更新できます。アドバンスト・レプリケーションと組み合せて使用した場合、更新はマスターにも影響します。

QUERY REWRITE 句

QUERY REWRITE 句を使用すると、マテリアライズド・ビューがクエリー・リライトで使用できるかどうかを指定できます。

ENABLE 句 ENABLE を指定すると、クエリー・リライトでマテリアライズド・ビューを使用可能にできます。

クエリー・リライトを使用可能にする場合の制限事項： クエリー・リライトを使用可能にする処理には、次の制限事項があります。

- マテリアライズド・ビューのすべてのユーザー定義ファンクションが DETERMINISTIC である場合のみ、クエリー・リライトを使用可能にできます。
- 文内の式が反復可能な場合のみ、クエリー・リライトを使用可能にできます。たとえば、CURRENT_TIME、USER、順序値（たとえば、CURRVAL または NEXTVAL 疑似列）、または SAMPLE 句（マテリアライズド・ビューの変更内容と異なる行をサンプルとして抽出します）を含めることはできません。

注意：

- クエリー・リライトでのマテリアライズド・ビューの使用は、デフォルトでは無効です。この句を指定して、マテリアライズド・ビューをクエリー・リライトに適用できるようにする必要があります。
- マテリアライズド・ビューを作成した後は、DBMS_STATS パッケージを使用して、その統計情報を収集する必要があります。クエリー・リライトを最適化するために、このパッケージで生成した統計情報が必要です。

参照：

- クエリー・リライトの詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。
- DBMS_STATS パッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。
- クエリー・リライトに関する問題の診断については、DBMS_MVIEW パッケージの EXPLAIN_MVIEW プロシージャを参照してください。クエリー・リライトの問題の修正については、DBMS_MVIEW パッケージの TUNE_MVIEW プロシージャを参照してください。
- 「CREATE FUNCTION」（14-48 ページ）

DISABLE 句 DISABLE を指定すると、マテリアライズド・ビューをクエリー・リライトで使用禁止にできます。使用禁止にしたマテリアライズド・ビューはリフレッシュ可能です。

AS *subquery*

マテリアライズド・ビューを定義する問合せを指定します。マテリアライズド・ビューの作成時に、この副問合せが実行され、実行結果がマテリアライズド・ビューに格納されます。この副問合せは、有効な SQL 副問合せである必要があります。ただし、すべての副問合せに高速リフレッシュを適用できるわけではなく、すべての副問合せがクエリー・リライトに使用できるわけでもありません。

マテリアライズド・ビューを定義する問合せに関する注意事項： マテリアライズド・ビューには、次の注意事項があります。

- BUILD DEFERRED を指定する場合、定義する問合せはすぐには実行されません。
- マテリアライズド・ビューを定義する問合せの FROM 句に指定する各表およびビューを、それを含むスキーマで修飾することをお薦めします。
- 表内で暗号化されている SELECT 構文のリストの列は、マテリアライズド・ビューでは暗号化されません。

参照： その他の注意事項については、16-56 ページの「CREATE TABLE」の「AS subquery」を参照してください。

マテリアライズド・ビューを定義する問合せに関する制限事項 : マテリアライズド・ビュー問合せには、次の制限事項があります。

- マテリアライズド・ビューを定義する問合せでは、ユーザー SYS が所有する表、ビューまたはマテリアライズド・ビューから選択できますが、このようなマテリアライズド・ビューに対して QUERY REWRITE は使用できません。
- マテリアライズド・ビューを定義するとき、定義する問合せの SELECT 構文のリストに副問合せを使用することはできません。ただし、WHERE 句内など、定義する問合せの他の箇所には、副問合せを含めることができます。
- GROUP BY 句を含むマテリアライズド結合ビューおよびマテリアライズド集計ビューは、索引構成表からは選択できません。
- マテリアライズド・ビューに LONG データ型の列を含めることはできません。
- 一時表に対しては、マテリアライズド・ビュー・ログを作成できません。そのため、定義する問合せが一時表を参照する場合、マテリアライズド・ビューは高速リフレッシュに適応せず、この文で QUERY REWRITE を指定することもできません。
- 定義する問合せの FROM 句が他のマテリアライズド・ビューを参照する場合、この文で作成するマテリアライズド・ビューのリフレッシュの前に、定義する問合せで参照されているマテリアライズド・ビューを常にリフレッシュしておく必要があります。
- 定義する問合せ内に結合式を持つマテリアライズド・ビューは、XML データ型の列を持つことができません。XML データ型には、XMLType および URI データ型の列が含まれます。

クエリー・リライトが可能なマテリアライズド・ビューを作成する場合、次の制限があります。

- 定義する問合せには、ROWNUM、USER、SYSDATE、リモート表、順序、または（データベースやパッケージ状態の書込みまたは読取りを行う）PL/SQL ファンクションに対する直接またはビューを介した参照を含めることはできません。
- マテリアライズド・ビューおよびマテリアライズド・ビューのマスター表はリモートにできません。

マテリアライズド・ビュー・ログを使用した高速リフレッシュを実行する場合は、いくつかの制限事項があります。

参照 :

- データ・ウェアハウスの制限事項は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。
- レプリケーションの制限事項は、『Oracle Database アドバンスト・レプリケーション』を参照してください。
- 15-22 ページの「マテリアライズド結合ビューの作成例 :」、15-22 ページの「副問合せマテリアライズド・ビューの作成例 :」および 15-25 ページの「ネステッド・マテリアライズド・ビューの作成例 :」を参照してください。

例

次の例では、15-26 ページの「CREATE MATERIALIZED VIEW LOG」の「例」で作成したマテリアライズド・ログが必要です。

単純なマテリアライズド・ビューの作成例 : 次の文は、hr スキーマ内の employees 表に基づいた非常に単純なマテリアライズド・ビューを作成します。

```
CREATE MATERIALIZED VIEW mv1 AS SELECT * FROM hr.employees;
```

デフォルトでは、REFRESH ON DEMAND のみが指定されている主キー・マテリアライズド・ビューが作成されます。マテリアライズド・ビュー・ログが employees に存在する場合は、高速リフレッシュできるように mv1 を変更できます。このようなログが存在しない場合は、mv1 の完全リフレッシュのみが可能です。Oracle Database では、mv1 に対してデフォルトの記憶域プロパティが使用されます。この操作に必要な権限は、CREATE MATERIALIZED VIEW システム権限および hr.employees に対する SELECT オブジェクト権限のみです。

副問合せマテリアライズド・ビューの作成例： 次の文は、リモート・データベースの sh スキーマの countries および customers 表を基にした副問合せマテリアライズド・ビューを作成します。

```
CREATE MATERIALIZED VIEW foreign_customers FOR UPDATE
   AS SELECT * FROM sh.customers@remote cu
   WHERE EXISTS
     (SELECT * FROM sh.countries@remote co
      WHERE co.country_id = cu.country_id);
```

マテリアライズド集計ビューの作成例： 次の文は、サンプル表 sh.sales にマテリアライズド集計ビューを作成して移入し、デフォルトのリフレッシュ方法、モードおよび時刻を指定します。この文は、ここに示されている 2 つのログおよび 15-31 ページの「マテリアライズド・ビュー・ログの作成例 :」で作成されたマテリアライズド・ビュー・ログを使用します。

```
CREATE MATERIALIZED VIEW LOG ON times
   WITH ROWID, SEQUENCE (time_id, calendar_year)
   INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW LOG ON products
   WITH ROWID, SEQUENCE (prod_id)
   INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW sales_mv
   BUILD IMMEDIATE
   REFRESH FAST ON COMMIT
   AS SELECT t.calendar_year, p.prod_id,
      SUM(s.amount_sold) AS sum_sales
      FROM times t, products p, sales s
      WHERE t.time_id = s.time_id AND p.prod_id = s.prod_id
      GROUP BY t.calendar_year, p.prod_id;
```

マテリアライズド結合ビューの作成例： 次の文は、サンプル・スキーマ sh の表を使用して、マテリアライズド集計ビュー sales_by_month_by_state を作成して移入します。マテリアライズド・ビューは、この文が正しく実行されるとすぐに移入されます。デフォルトで、次のマテリアライズド・ビューを定義する問合せを再実行することによって、後続のリフレッシュが行われます。

```
CREATE MATERIALIZED VIEW sales_by_month_by_state
     TABLESPACE example
     PARALLEL 4
     BUILD IMMEDIATE
     REFRESH COMPLETE
     ENABLE QUERY REWRITE
     AS SELECT t.calendar_month_desc, c.cust_state_province,
        SUM(s.amount_sold) AS sum_sales
        FROM times t, sales s, customers c
        WHERE s.time_id = t.time_id AND s.cust_id = c.cust_id
        GROUP BY t.calendar_month_desc, c.cust_state_province;
```

事前作成したマテリアライズド・ビューの作成例： 次の文は、既存の集計表 sales_sum_table に対するマテリアライズド集計ビューを作成します。

```
CREATE TABLE sales_sum_table
   (month VARCHAR2(8), state VARCHAR2(40), sales NUMBER(10,2));

CREATE MATERIALIZED VIEW sales_sum_table
   ON PREBUILT TABLE WITH REDUCED PRECISION
   ENABLE QUERY REWRITE
   AS SELECT t.calendar_month_desc AS month,
             c.cust_state_province AS state,
             SUM(s.amount_sold) AS sales
      FROM times t, customers c, sales s
      WHERE s.time_id = t.time_id AND s.cust_id = c.cust_id
      GROUP BY t.calendar_month_desc, c.cust_state_province;
```

前述の例では、マテリアライズド・ビューは事前作成表と同じ名前を持ち、かつ事前作成表と同じデータ型で同じ数の列を持ちます。WITH REDUCED PRECISION 句によって、マテリアライズド・ビュー列の精度と副問合せによって戻された値の精度の違いは許可されます。

主キー・マテリアライズド・ビューの作成例： 次の文は、サンプル表 oe.product_information に主キー・マテリアライズド・ビュー catalog を作成します。

```
CREATE MATERIALIZED VIEW catalog
   REFRESH FAST START WITH SYSDATE NEXT  SYSDATE + 1/4096
   WITH PRIMARY KEY
   AS SELECT * FROM product_information;
```

ROWID マテリアライズド・ビューの作成例： 次の文は、サンプル表 oe.orders に ROWID マテリアライズド・ビューを作成します。

```
CREATE MATERIALIZED VIEW order_data REFRESH WITH ROWID
   AS SELECT * FROM orders;
```

マテリアライズド・ビューの定期的リフレッシュ例： 次の文は、主キー・マテリアライズド・ビュー emp_data を作成し、サンプル表 hr.employees のデータを移入します。

```
CREATE MATERIALIZED VIEW LOG ON employees
   WITH PRIMARY KEY
   INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW emp_data
   PCTFREE 5 PCTUSED 60
   TABLESPACE example
   STORAGE (INITIAL 50K NEXT 50K)
   REFRESH FAST NEXT sysdate + 7
   AS SELECT * FROM employees;
```

前述の文には START WITH パラメータが指定されていないため、Oracle Database では、現行の SYSDATE を使用して NEXT 値が評価され、最初の自動リフレッシュ時刻が判断されます。従業員表のマテリアライズド・ビュー・ログが作成されたため、マテリアライズド・ビュー作成の 7 日後から、7 日ごとにマテリアライズド・ビューの高速リフレッシュが実行されます。

マテリアライズド・ビューが高速リフレッシュを行うための条件と一致するため、高速リフレッシュが行われます。また、前述の文では、マテリアライズド・ビューをメンテナンスするためにデータベースで使用される表の記憶特性も設定しています。

マテリアライズド・ビューの自動リフレッシュ時刻の例： 次の文は、remote および local データベースの従業員表を問い合せる複合マテリアライズド・ビュー all_customers を作成します。

```
CREATE MATERIALIZED VIEW all_customers
   PCTFREE 5 PCTUSED 60
   TABLESPACE example
   STORAGE (INITIAL 50K NEXT 50K)
   USING INDEX STORAGE (INITIAL 25K NEXT 25K)
   REFRESH START WITH ROUND(SYSDATE + 1) + 11/24
   NEXT NEXT_DAY(TRUNC(SYSDATE), 'MONDAY') + 15/24
   AS SELECT * FROM sh.customers@remote
         UNION
      SELECT * FROM sh.customers@local;
```

このマテリアライズド・ビューは、翌日の午前 11:00 に自動的にリフレッシュされ、その後は、毎週月曜日の午後 3:00 にリフレッシュされます。デフォルトのリフレッシュ方法は、FORCE です。定義する問合せには UNION 演算子が含まれますが、UNION 演算子は高速リフレッシュでサポートされていないため、自動的に完全リフレッシュが実行されます。

前述の文では、マテリアライズド・ビューおよびこのマテリアライズド・ビューをメンテナンスするために使用される索引の記憶特性を次のように設定しています。

- 最初の STORAGE 句で、マテリアライズド・ビューの 1 番目と 2 番目のエクステントのサイズをそれぞれ 50KB に設定します。
- 次の USING INDEX 句付きの STORAGE 句では、索引の 1 番目と 2 番目のエクステントのサイズをそれぞれ 25KB に変更します。

高速リフレッシュ可能なマテリアライズド・ビューの作成例： 次の文は、product_information および inventories 表から WHERE 条件で戻される行を制限する UNION 集合演算子を使用して、サンプル・スキーマ oe の表 order_items から列を選択する高速リフレッシュが可能なマテリアライズド・ビューを作成します。order_items および product_information のマテリアライズド・ビュー・ログは、15-31 ページの CREATE MATERIALIZED VIEW LOG の「例」で作成されたものです。また、次の例では、oe.inventories のマテリアライズド・ビュー・ログが必要です。

```
CREATE MATERIALIZED VIEW LOG ON inventories
   WITH (quantity_on_hand);

CREATE MATERIALIZED VIEW warranty_orders REFRESH FAST AS
  SELECT order_id, line_item_id, product_id FROM order_items o
    WHERE EXISTS
    (SELECT * FROM inventories i WHERE o.product_id = i.product_id
      AND i.quantity_on_hand IS NOT NULL)
  UNION
    SELECT order_id, line_item_id, product_id FROM order_items
    WHERE quantity > 5;
```

マテリアライズド・ビュー warranty_orders には、order_items（product_id を結合列とする）および inventories（quantity_on_hand をフィルタ列とする）で定義されたマテリアライズド・ビュー・ログが必要です。15-31 ページの「マテリアライズド・ビュー・ログにフィルタ列を指定する例：」および「マテリアライズド・ビュー・ログに結合列を指定する例：」を参照してください。

ネステッド・マテリアライズド・ビューの作成例 : 次の例は、前述の例のマテリアライズド・ビューをマスター表として使用し、サンプル・スキーマ oe の特定の販売員で構成されるマテリアライズド・ビューを作成します。

```
CREATE MATERIALIZED VIEW my_warranty_orders
   AS SELECT w.order_id, w.line_item_id, o.order_date
   FROM warranty_orders w, orders o
   WHERE o.order_id = o.order_id
   AND o.sales_rep_id = 165;
```

CREATE MATERIALIZED VIEW LOG

用途

CREATE MATERIALIZED VIEW LOG 文を使用すると、マテリアライズド・ビューのマスター表に関連する表**マテリアライズド・ビュー・ログ**を作成できます。

注意： 下位互換性を保つために、MATERIALIZED VIEW のかわりにキーワード SNAPSHOT もサポートされています。

マスター表のデータが DML によって変更された場合、Oracle Database は変更を記述する行をマテリアライズド・ビュー・ログに格納し、そのマテリアライズド・ビュー・ログを使用して、マスター表を基にしたマテリアライズド・ビューをリフレッシュします。このプロセスを、増分リフレッシュまたは**高速リフレッシュ**といいます。マテリアライズド・ビュー・ログがない場合、マテリアライズド・ビュー問合せが再実行され、マテリアライズド・ビューがリフレッシュされます。このプロセスが**完全リフレッシュ**です。通常、高速リフレッシュは、完全リフレッシュよりも高速に処理されます。

マテリアライズド・ビュー・ログは、マスター表と同一のスキーマ内のマスター・データベースにあります。マスター表に定義できるマテリアライズド・ビュー・ログは 1 つのみです。Oracle Database では、このマテリアライズド・ビュー・ログを使用して、マスター表に基づく高速リフレッシュが可能なすべてのマテリアライズド・ビューに対して、高速リフレッシュを実行します。

マテリアライズド結合ビューを高速リフレッシュする場合、マテリアライズド・ビューが参照するそれぞれのマスター表に対するマテリアライズド・ビュー・ログを作成する必要があります。

参照：

- マテリアライズド・ビューの概要は、15-4 ページの「CREATE MATERIALIZED VIEW」、11-2 ページの「ALTER MATERIALIZED VIEW」、『Oracle Database 概要』、『Oracle Database データ・ウェアハウス・ガイド』および『Oracle Database アドバンスト・レプリケーション』を参照してください。
- マテリアライズド・ビュー・ログの変更については、11-16 ページの「ALTER MATERIALIZED VIEW LOG」を参照してください。
- マテリアライズド・ビュー・ログの削除については、17-51 ページの「DROP MATERIALIZED VIEW LOG」を参照してください。
- ダイレクト・ローダー・ログの使用については、『Oracle Database ユーティリティ』を参照してください。

前提条件

マテリアライズド・ビュー・ログを作成するために必要な権限は、マテリアライズド・ビュー・ログに関連付けられた基礎となるオブジェクトの作成に必要な権限と直接関連しています。

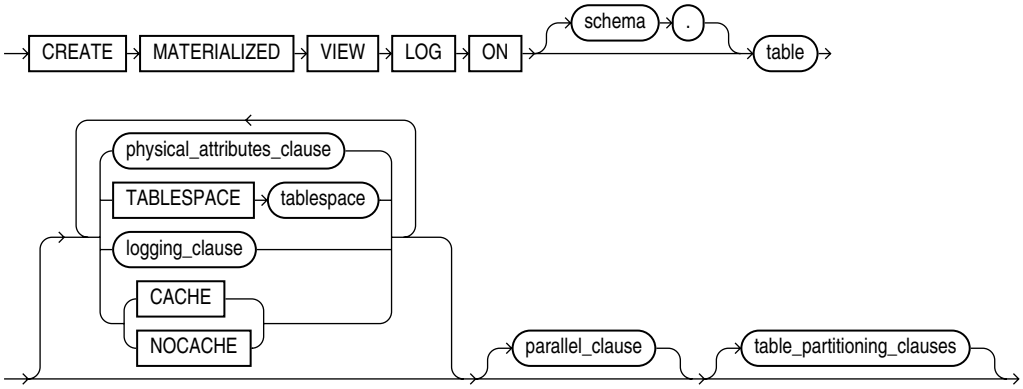
- マスター表を所有しているユーザーに CREATE TABLE 権限がある場合、関連するマテリアライズド・ビュー・ログを作成できます。
- 他のユーザーのスキーマ内に表のマテリアライズド・ビュー・ログを作成する場合は、CREATE ANY TABLE システム権限、COMMENT ANY TABLE システム権限、およびマスター表に対する SELECT オブジェクト権限または SELECT ANY TABLE システム権限が必要です。

どちらの場合も、マテリアライズド・ビュー・ログの所有者には、マテリアライズド・ビュー・ログを格納するための表領域への十分な割当て制限または UNLIMITED TABLESPACE システム権限が必要です。

> **参照：** マテリアライズド・ビュー・ログを作成する場合の前提条件については、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

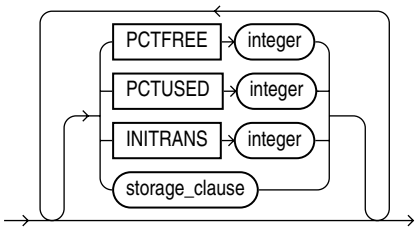
構文

***create_materialized_vw_log*::=**



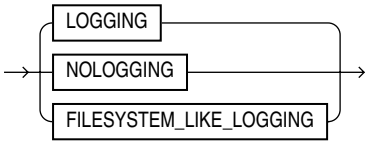
（15-8 ページの physical_attributes_clause::=、15-28 ページの logging_clause::=、15-28 ページの parallel_clause::=、16-17 ページの「CREATE TABLE」の table_partitioning_clauses::=、15-28 ページの new_values_clause::= を参照）

physical_attributes_clause::=

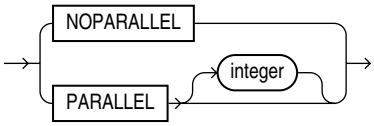


（8-44 ページの storage_clause::= を参照）

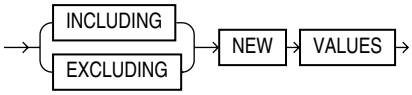
***logging_clause*::=**



parallel_clause::=



***new_values_clause*::=**



セマンティクス

schema

マテリアライズド・ビュー・ログのマスター表が含まれているスキーマを指定します。*schema* を省略した場合、マスター表は自分のスキーマ内に含まれているとみなされます。マテリアライズド・ビュー・ログは、マスター表のスキーマ内に作成されます。なお、ユーザー SYS のスキーマ内の表に対しては、マテリアライズド・ビュー・ログを作成できません。

table

マテリアライズド・ビュー・ログを作成するマスター表の名前を指定します。

マテリアライズド・ビュー・ログのマスター表の制限事項 : 一時表またはビューに対しては、マテリアライズド・ビュー・ログを作成できません。

参照 : 「マテリアライズド・ビュー・ログの作成例 :」(15-31 ページ)

physical_attributes_clause

physical_attributes_clause を使用すると、マテリアライズド・ビュー・ログにおける物理特性および記憶特性を定義できます。

参照 : デフォルト値を含むこれらの句の詳細は、8-39 ページの「physical_attributes_clause」および 8-41 ページの「storage_clause」を参照してください。

TABLESPACE 句

マテリアライズド・ビュー・ログを作成する表領域を指定します。この句を省略した場合、マテリアライズド・ビュー・ログのスキーマのデフォルト表領域内にマテリアライズド・ビュー・ログが作成されます。

logging_clause

LOGGING または NOLOGGING を指定すると、マテリアライズド・ビュー・ログのロギング特性を設定できます。デフォルトは、マテリアライズド・ビュー・ログが存在する表領域のロギング特性です。

参照： この句の詳細は、8-34 ページの「logging_clause」を参照してください。

CACHE | NOCACHE

アクセス頻度の高いデータについて、CACHE は、全表スキャンの実行時にこのログ用に取り出された各ブロックを、バッファ・キャッシュの最低使用頻度（LRU）リストの最高使用頻度側に入れることを指定します。この属性は、小規模な参照表で有効です。

NOCACHE は、ブロックを LRU リストの最低使用頻度側に入れることを指定します。デフォルトは NOCACHE です。

注意： NOCACHE は、*storage_clause* に KEEP を指定したマテリアライズド・ビュー・ログには、影響しません。

参照： CACHE または NOCACHE の指定の詳細は、16-6 ページの「CREATE TABLE」を参照してください。

parallel_clause

parallel_clause を使用すると、パラレル操作でマテリアライズド・ビュー・ログをサポートするかどうかを指定できます。

この句の詳細は、16-54 ページの「CREATE TABLE」の「parallel_clause」を参照してください。

table_partitioning_clauses

table_partitioning_clauses を使用すると、マテリアライズド・ビュー・ログが、指定された範囲の値またはハッシュ・ファンクションでパーティション化されることを指定できます。マテリアライズド・ビュー・ログのパーティション化は、表のパーティション化と同じです。

参照： 16-44 ページの「CREATE TABLE」の「table_partitioning_clauses」を参照してください。

WITH 句

WITH 句を使用すると、マスター内の行の更新時に、マテリアライズド・ビュー・ログに主キー、ROWID、オブジェクト ID またはこれらの行識別子の組合せを記録するかどうかを指定できます。また、マテリアライズド・ビュー・ログに順序を追加し、レコードに対する追加の順序情報を提供することもできます。

この句を指定すると、マテリアライズド・ビュー・ログが**フィルタ列**（副問合せマテリアライズド・ビューが参照する主キー以外の列）または**結合列**（副問合せの WHERE 句で結合を定義する主キー以外の列）として参照する追加の列を記録するかどうかを指定できます。

この句を指定しない場合、または PRIMARY KEY、ROWID または OBJECT ID なしで句を指定する場合は、デフォルトで主キー値が格納されます。ただし、作成時に OBJECT ID または ROWID のみを指定した場合は、主キー値は暗黙的に格納されません。明示的に、またはデフォルトで作成された主キー・ログによって、主キー制約の追加の検証が実行されます。

OBJECT ID OBJECT ID を指定すると、更新されるすべての行に対するシステム生成またはユーザー定義のオブジェクト識別子をマテリアライズド・ビュー・ログに記録することを指定できます。

OBJECT ID の制限事項： OBJECT ID は、オブジェクト表のログの作成時のみに指定でき、記憶表用には指定できません。

PRIMARY KEY PRIMARY KEY を指定すると、更新されるすべての行の主キーをマテリアライズド・ビュー・ログに記録することを指定できます。

ROWID ROWID を指定すると、更新されるすべての行の ROWID をマテリアライズド・ビュー・ログに記録することを指定できます。

SEQUENCE SEQUENCE を指定すると、追加の順序情報を提供する順序値をマテリアライズド・ビュー・ログに記録できます。更新操作後の高速リフレッシュには、順序番号が必要です。

> **参照：** マテリアライズド・ビュー・ログの順序番号の使用およびこの句の使用例については、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

column 更新されるすべての行に対して、マテリアライズド・ビュー・ログに値を記録する列を指定します。通常、フィルタ列および結合列を指定します。

WITH 句の制限事項： この句には、次の制限事項があります。

- 各マテリアライズド・ビュー・ログに指定できるのは、PRIMARY KEY、ROWID、OBJECT ID、SEQUENCE、および列リストを 1 つずつです。
- 主キー列は、マテリアライズド・ビュー・ログに暗黙的に記録されます。そのため、*column* が主キー列の 1 つを含む場合は、次の結合はいずれも指定できません。

```
WITH ... PRIMARY KEY ... (column)
WITH ... (column) ... PRIMARY KEY
WITH (column)
```

> **参照：**
>
> - マテリアライズド・ビュー・ログに値を明示的および暗黙的に含める方法については、15-4 ページの「CREATE MATERIALIZED VIEW」を参照してください。
> - フィルタ列および結合列については、『Oracle Database アドバンスト・レプリケーション』を参照してください。
> - 15-31 ページの「マテリアライズド・ビュー・ログにフィルタ列を指定する例：」および 15-31 ページの「マテリアライズド・ビュー・ログに結合列を指定する例：」を参照してください。

NEW VALUES 句

NEW VALUES 句を使用すると、更新 DML 操作で、古い値と新しい値の両方をマテリアライズド・ビュー・ログに保存するかどうかを指定できます。

> **参照：** 「マテリアライズド・ビュー・ログに新しい値を追加する例：」(15-31 ページ)

INCLUDING INCLUDING を指定すると、古い値と新しい値の両方を保存できます。このログが単一表マテリアライズド集計ビューの表用で、マテリアライズド・ビューに高速リフレッシュを実行する場合、INCLUDING を指定してください。

EXCLUDING EXCLUDING を指定すると、ログに新しい値が記録されなくなります。これはデフォルトです。この句を使用すると、新しい値の記録によるオーバーヘッドを回避できます。マスター表に高速リフレッシュが可能な単一表マテリアライズド集計ビューが定義されている場合は、この句を使用しないでください。

例

マテリアライズド・ビュー・ログの作成例： 次の文は、物理特性および記憶特性を指定するoe.customers表にマテリアライズド・ビュー・ログを作成します。

```
CREATE MATERIALIZED VIEW LOG ON customers
   PCTFREE 5
   TABLESPACE example
   STORAGE (INITIAL 10K NEXT 10K);
```

customersのマテリアライズド・ビュー・ログは、主キー・マテリアライズド・ビューの高速リフレッシュのみサポートします。

次の文は、ROWID句を指定してマテリアライズド・ビュー・ログの別のバージョンを作成します。これによって、高速リフレッシュが使用可能になるマテリアライズド・ビューのタイプが追加されます。

```
CREATE MATERIALIZED VIEW LOG ON customers WITH PRIMARY KEY, ROWID;
```

このcustomersのマテリアライズド・ビュー・ログによって、ROWIDマテリアライズド・ビューおよびマテリアライズド結合ビューに対する高速リフレッシュが実行可能になります。マテリアライズド集計ビューの高速リフレッシュを指定するには、次の例に示すとおり、SEQUENCEおよびINCLUDING NEW VALUES句も指定する必要があります。

マテリアライズド・ビュー・ログにフィルタ列を指定する例： 次の文は、sh.sales表にマテリアライズド・ビュー・ログを作成します。この文は、15-22ページの「マテリアライズド集計ビューの作成例：」で使用します。この文は、このマテリアライズド・ビューで参照される表のすべての列を、フィルタ列として指定します。

```
CREATE MATERIALIZED VIEW LOG ON sales
   WITH ROWID, SEQUENCE(amount_sold, time_id, prod_id)
   INCLUDING NEW VALUES;
```

マテリアライズド・ビュー・ログに結合列を指定する例： 次の文は、サンプル・スキーマoeのorder_items表にマテリアライズド・ビュー・ログを作成します。ログは、主キーおよび15-24ページの「高速リフレッシュ可能なマテリアライズド・ビューの作成例：」で結合列として使用したproduct_idを記録します。

```
CREATE MATERIALIZED VIEW LOG ON order_items WITH (product_id);
```

マテリアライズド・ビュー・ログに新しい値を追加する例： 次の例は、INCLUDING NEW VALUESを指定するoe.product_information表にマテリアライズド・ビュー・ログを作成します。

```
CREATE MATERIALIZED VIEW LOG ON product_information
   WITH ROWID, SEQUENCE (list_price, min_price, category_id), PRIMARY KEY
   INCLUDING NEW VALUES;
```

次の文は、マテリアライズド集計ビューを作成し、product_informationログを使用します。

```
CREATE MATERIALIZED VIEW products_mv
   REFRESH FAST ON COMMIT
   AS SELECT SUM(list_price - min_price), category_id
         FROM product_information
         GROUP BY category_id;
```

マスター表に定義されたログに古い値と新しい値の両方が含まれるため、このマテリアライズド・ビューでは、高速リフレッシュを実行できます。

CREATE OPERATOR

用途

CREATE OPERATOR 文を使用すると、新しい演算子を作成し、バインディングを定義できます。

演算子は、索引タイプ、SQL 問合せおよび DML 文で参照できます。それに対して演算子は、ファンクション、パッケージ、型および他のユーザー定義オブジェクトを参照します。

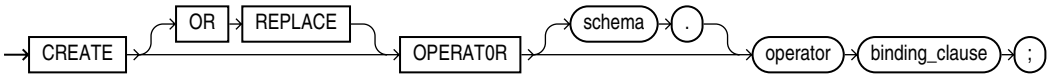
> **参照：** これらの依存性および一般的な演算子については、『Oracle Database データ・カートリッジ開発者ガイド』および『Oracle Database 概要』を参照してください。

前提条件

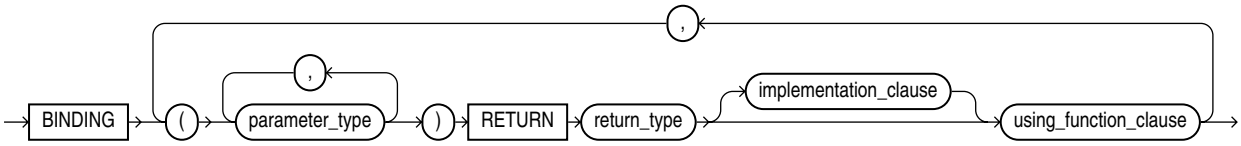
自分のスキーマ内に演算子を作成する場合は、CREATE OPERATOR システム権限が必要です。他のユーザーのスキーマ内に演算子を作成する場合は、CREATE ANY OPERATOR システム権限が必要です。どちらの場合も、参照するファンクションおよび演算子に対する EXECUTE オブジェクト権限が必要です。

構文

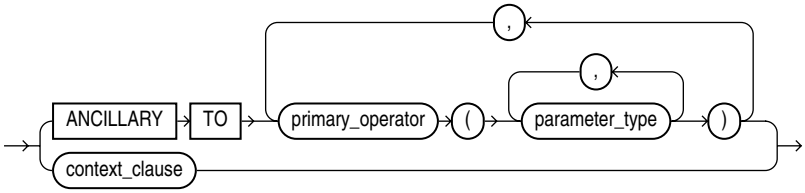
***create_operator*::=**



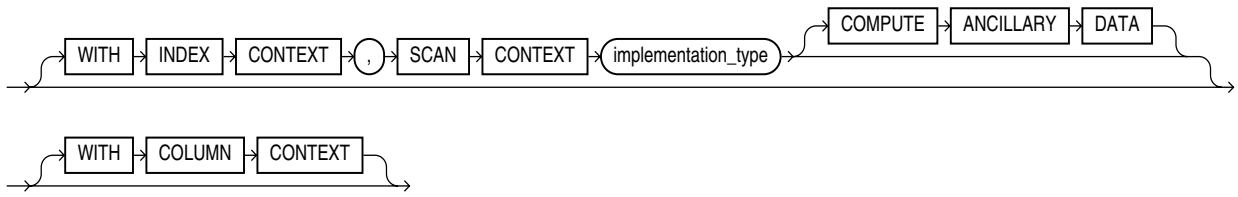
binding_clause::=



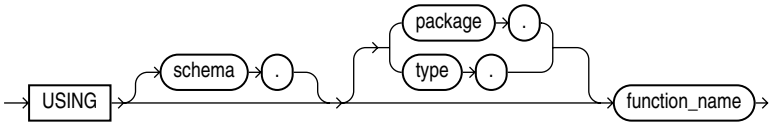
implementation_clause::=



context_clause::=



using_function_clause::=



セマンティクス

OR REPLACE

OR REPLACE を指定すると、演算子スキーマ・オブジェクトの定義を置換できます。

演算子の置換の制限事項： 演算子をサポートする索引タイプなどの依存オブジェクトが演算子にない場合のみ、定義を置換できます。

schema

演算子が含まれているスキーマを指定します。*schema* を省略した場合、自分のスキーマ内に演算子が作成されます。

operator

作成する演算子の名前を指定します。

binding_clause

binding_clause を使用すると、演算子をファンクションにバインドするために、1 つ以上のパラメータ・データ型（*parameter_type*）を指定できます。各バインドの署名（対応するファンクションに対する引数のデータ型のシーケンス）は、オーバーロードの規則に従って一意である必要があります。

parameter_type 自体をオブジェクト型にできます。この場合、任意にスキーマで修飾することもできます。

演算子のバインドの制限事項： REF、LONG または LONG RAW は *parameter_type* に指定できません。

> **参照：** オーバーロードの詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

RETURN 句

バインドに戻りデータ型を指定します。

return_type 自体をオブジェクト型にできます。この場合、任意にスキーマで修飾することもできます。

戻りデータ型のバインドの制限事項： REF、LONG または LONG RAW は *return_type* に対して指定できません。

implementation_clause

この句を使用すると、バインドの実装を指定できます。

ANCILLARY TO 句

ANCILLARY TO 句を使用すると、演算子バインドが、指定した主演算子バインド（*primary_operator*）を補助することを指定できます。この句を指定する場合は、1 つのみの数値パラメータで前のバインドを指定しないでください。

context_clause

context_clause を使用すると、主演算子バインドを補助しないバインドのファンクション実装を指定できます。

WITH INDEX CONTEXT、SCAN CONTEXT この句を使用すると、演算子のファンクション評価に、実装タイプで指定した索引およびスキャン・コンテキストが使用されるように指定できます。

COMPUTE ANCILLARY DATA COMPUTE ANCILLARY DATA を指定すると、演算子バインディングが補助データを計算するように指定できます。

WITH COLUMN CONTEXT WITH COLUMN CONTEXT を指定すると、列情報が演算子のファンクション実装に渡されるように指定できます。

この句を指定する場合は、実装するファンクションの署名に、余分な ODCIFuncCallInfo 構造を 1 つ含める必要があります。

> **参照：** ODCIFuncCallInfo ルーチンを使用する手順については、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。

using_function_clause

using_function_clause を使用すると、バインディングを実装するファンクションを指定できます。*function_name* には、スタンドアロン・ファンクション、パッケージ・ファンクション、型メソッドまたはこれらのシノニムを指定できます。

ファンクションが後で削除された場合は、演算子などのすべての依存オブジェクトに、INVALID のマークが付けられます。ただし、その後 ALTER OPERATOR ... DROP BINDING 文を発行してバインディングを削除した場合は、後続の問合せおよび DML で依存オブジェクトが再検証されます。

例

ユーザー定義演算子の作成例： 次の文は、非常に単純な等価性のファンクション実装を作成した後、このファンクションを使用する演算子を作成します。より詳細な例については、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。

```
CREATE FUNCTION eq_f(a VARCHAR2, b VARCHAR2) RETURN NUMBER AS
BEGIN
   IF a = b THEN RETURN 1;
   ELSE RETURN 0;
   END IF;
END;
/

CREATE OPERATOR eq_op
   BINDING (VARCHAR2, VARCHAR2)
   RETURN NUMBER
   USING eq_f;
```

CREATE OUTLINE

用途

注意： ストアド・アウトラインは、SQL 計画管理を優先して、今後のリリースではサポートされなくなります。Oracle Database 11*g* リリース 1（11.1）では、ストアド・アウトラインは過去のリリースと同様に機能します。ただし、新規のアプリケーションには SQL 計画管理を使用することをお薦めします。SQL 計画管理では、ストアド・アウトラインよりもすぐれた SQL パフォーマンスと安定性を実現する SQL 計画ベースラインが作成されます。

既存のストアド・アウトラインがある場合は、DBMS_SPM パッケージの LOAD_PLANS_FROM_CURSOR_CACHE プロシージャまたは LOAD_PLANS_FROM_SQLSET プロシージャを使用して、SQL 計画ベースラインに移行することを検討してください。移行が完了した場合、ストアド・アウトラインを無効にするか、または削除する必要があります。

参照 : SQL 計画管理の詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。DBMS_SPM パッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

CREATE OUTLINE 文を使用すると、**ストアド・アウトライン**を作成できます。ストアド・アウトラインは、実行計画を生成するためにオプティマイザが使用する属性集合です。最適化に影響する要因に変更があるかどうかにかかわらず、特定の SQL 文が発行された場合に、実行計画の生成に影響するアウトラインの集合を使用するように、オプティマイザに指示します。これらの要因における変更を考慮するように、アウトラインを変更することもできます。

注意： 影響を与える SQL 文には、アウトライン作成時に指定した文と一致する文字列を指定する必要があります。

参照 :

- 実行計画の詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。
- アウトラインの変更については、11-25 ページの「ALTER OUTLINE」を参照してください。
- USE_STORED_OUTLINES パラメータおよび USE_PRIVATE_OUTLINES パラメータの詳細は、11-41 ページの「ALTER SESSION」および 11-52 ページの「ALTER SYSTEM」を参照してください。

前提条件

パブリック・アウトラインまたはプライベート・アウトラインの作成には、CREATE ANY OUTLINE システム権限が必要です。

ソース・アウトラインからクローン・アウトラインを作成するには、SELECT_CATALOG_ROLE ロールも必要です。

個々のセッションまたはシステムに対して、ストアド・アウトラインを動的に使用可能または使用禁止にできます。

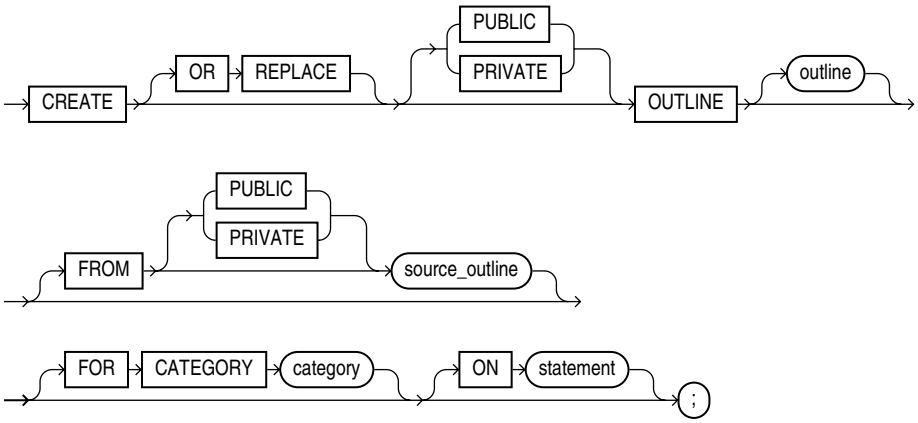
- USE_STORED_OUTLINES パラメータを使用可能にすると、パブリック・アウトラインが使用可能になります。
- USE_PRIVATE_OUTLINES パラメータを使用可能にすると、プライベート・ストアド・アウトラインが使用可能になります。

参照：

- アウトラインを使用したパフォーマンス・チューニングの詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。
- DBMS_OUTLN_EDIT パッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

構文

***create_outline*::=**



注意： *outline* の後に指定する必須の句はありません。ただし、*outline* の後には 1 つ以上の句を指定する必要があり、FROM 句または ON 句のどちらかを指定する必要があります。

セマンティクス

OR REPLACE

OR REPLACE を指定すると、既存のアウトラインを同じ名前の新しいアウトラインと置き換えることができます。

PUBLIC | PRIVATE

PUBLIC を指定すると、PUBLIC が使用するアウトラインを作成できます。これはデフォルトです。

PRIVATE を指定すると、現行のセッションのみがプライベートに使用するアウトラインを作成できます。このアウトラインのデータは、現行のスキーマに格納されます。

注意： プライベート・アウトラインを作成するには、DBMS_OUTLN_EDIT.CREATE_EDIT_TABLES プロシージャの実行によって自分のスキーマのアウトライン・データを保持するアウトライン編集表が必要です。このプロシージャの実行には、DBMS_OUTLN_EDIT パッケージに対する EXECUTE オブジェクト権限が必要です。

outline

ストアド・アウトラインに割り当てる一意の名前を指定します。*outline* を指定しない場合、データベースがアウトライン名を生成します。

参照： 「アウトラインの作成例 :」（15-38 ページ）

FROM *source_outline* 句

FROM 句を使用すると、既存のアウトラインのコピーによって新しいアウトラインを作成できます。デフォルトでは、パブリック領域の *source_category* が検索されます。PRIVATE を指定すると、現行のスキーマのアウトラインが検索されます。

アウトラインのコピーの制限事項： FROM 句を指定する場合、ON 句は指定できません。

参照： 15-38 ページの「プライベート・クローン・アウトラインの作成例 :」および 15-38 ページの「プライベート・アウトラインのパブリック領域への公開例 :」を参照してください。

FOR CATEGORY 句

ストアド・アウトラインをグループ化するために使用する任意の名前を指定します。たとえば、週末に使用するアウトラインの 1 つのカテゴリおよび四半期末に使用する別のカテゴリを指定できます。*category* を指定しない場合、アウトラインは DEFAULT カテゴリに格納されます。

ON 句

文をコンパイルする際にアウトラインが作成される SQL 文を指定します。この句は、FROM 句を使用して既存のアウトラインのコピーを作成する場合のみに指定するオプションです。

指定できる文は、SELECT、DELETE、UPDATE、INSERT ... SELECT または CREATE TABLE ... AS SELECT のいずれかです。

ON 句の制限事項： この句には、次の制限事項があります。

- ON 句を指定する場合、FROM 句は指定できません。
- マルチ表の INSERT 文でアウトラインを作成できません。
- ON 句内の SQL 文に、リモート・オブジェクトの DML 操作を含めることはできません。

注意： 後続の文で、同じ SQL 文に対して追加のアウトラインを指定できますが、同じ文の各アウトラインは CATEGORY 句で異なるカテゴリを指定する必要があります。

例

アウトラインの作成例： 次の文は、ON 文をコンパイルすることによってストアド・アウトラインを作成します。アウトラインは salaries という名前で、special カテゴリに格納されます。

```
CREATE OUTLINE salaries FOR CATEGORY special
   ON SELECT last_name, salary FROM employees;
```

USE_STORED_OUTLINES パラメータに special が設定されている場合、同じ SELECT 文が後でコンパイルされると、アウトライン salaries を作成する場合と同様に実行計画が生成されます。

プライベート・クローン・アウトラインの作成例： 次の文は、前述の例で作成したパブリック・カテゴリ salaries に基づいて、ストアド・プライベート・アウトライン my_salaries を作成します。プライベート・アウトラインを作成する場合、プライベート・アウトラインを作成するユーザーには、DBMS_OUTLN_EDIT パッケージに対する EXECUTE オブジェクト権限が必要です。また、このパッケージの CREATE_EDIT_TABLES プロシージャを実行する必要があります。

```
EXECUTE DBMS_OUTLN_EDIT.CREATE_EDIT_TABLES;

CREATE OR REPLACE PRIVATE OUTLINE my_salaries
   FROM salaries;
```

プライベート・アウトラインのパブリック領域への公開例： 次の文は、プライベート編集の後でプライベート・アウトラインをパブリック領域にコピー（公開）します。

```
CREATE OR REPLACE OUTLINE public_salaries
   FROM PRIVATE my_salaries;
```

CREATE PACKAGE

用途

パッケージは PL/SQL を使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は『Oracle Database PL/SQL 言語リファレンス』を参照してください。

CREATE PACKAGE 文を使用すると、ストアド・**パッケージ**の仕様部を作成できます。パッケージとは、関連するプロシージャ、ファンクション、およびデータベース上にまとめて格納されるその他のプログラム・オブジェクトの集合のことです。**パッケージ仕様部**では、これらのオブジェクトを宣言します。後で指定する**パッケージ本体**では、これらのオブジェクトを定義します。

参照 :

- パッケージ実装の指定については、15-41 ページの「CREATE PACKAGE BODY」を参照してください。
- スタンドアロン・ファンクションおよびプロシージャの作成については、14-48 ページの「CREATE FUNCTION」および 15-45 ページの「CREATE PROCEDURE」を参照してください。
- パッケージの変更および削除については、11-27 ページの「ALTER PACKAGE」および 17-55 ページの「DROP PACKAGE」を参照してください。
- パッケージの詳細および使用方法については、『Oracle Database アドバンスト・アプリケーション開発者ガイド』および『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

前提条件

自分のスキーマ内にパッケージを作成または再作成する場合は、CREATE PROCEDURE システム権限が必要です。他のユーザーのスキーマ内にパッケージを作成または再作成する場合は、CREATE ANY PROCEDURE システム権限が必要です。

Oracle Database のプリコンパイラ・プログラム内に CREATE PACKAGE 文を埋め込む場合、キーワード END-EXEC に続けて、各言語の埋込み SQL 文の終了記号を記述して文を終了する必要があります。

参照 : 詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

構文

パッケージは PL/SQL を使用して定義されます。このため、このマニュアルの構文図では SQL キーワードのみを示します。PL/SQL の構文、セマンティクスおよび例については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

***create_package*::=**

(*plsql_source* については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。)

セマンティクス

OR REPLACE

OR REPLACE を指定すると、既存のパッケージ仕様部を再作成できます。この句を指定した場合、パッケージに対して付与されていたオブジェクト権限を削除、再作成および再付与しなくても、既存のパッケージの仕様部を変更できます。パッケージ仕様部を変更した場合、その仕様部は自動的に再コンパイルされます。

再定義したパッケージに対して権限が付与されていたユーザーは、権限が再付与されなくてもそのパッケージにアクセスできます。

ファンクション索引がパッケージに依存している場合、索引に DISABLED のマークが付きます。

> **参照：** パッケージ仕様部の再コンパイルについては、「ALTER PACKAGE」を参照してください。

plsql_source

plsql_source の構文、セマンティクスおよび例については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

CREATE PACKAGE BODY

用途

パッケージ本体は PL/SQL を使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は『Oracle Database PL/SQL 言語リファレンス』を参照してください。

CREATE PACKAGE BODY 文を使用すると、ストアド・**パッケージ**の本体を作成できます。パッケージとは、関連するプロシージャ、ストアド・ファンクション、およびデータベース上にまとめて格納されるその他のプログラム・オブジェクトの集合のことです。**パッケージ本体**では、これらのオブジェクトを定義します。前述の CREATE PACKAGE 文で定義する**パッケージ仕様部**では、これらのオブジェクトを宣言します。

一連のプロシージャやファンクションをスタンドアロンのスキーマ・オブジェクトとして作成するかわりの方法としてパッケージを使用する方法があります。

参照 :

- スタンドアロン・ファンクションおよびプロシージャの作成については、14-48 ページの「CREATE FUNCTION」および 15-45 ページの「CREATE PROCEDURE」を参照してください。
- 作成方法を含むパッケージの詳細は、15-39 ページの「CREATE PACKAGE」を参照してください。
- パッケージの変更については、11-27 ページの「ALTER PACKAGE」を参照してください。
- データベースからのパッケージの削除については、17-55 ページの「DROP PACKAGE」を参照してください。

前提条件

自分のスキーマ内にパッケージを作成または再作成する場合は、CREATE PROCEDURE システム権限が必要です。他のユーザーのスキーマ内にパッケージを作成または再作成する場合は、CREATE ANY PROCEDURE システム権限が必要です。どちらの場合も、パッケージ本体はパッケージと同じスキーマ内に作成される必要があります。

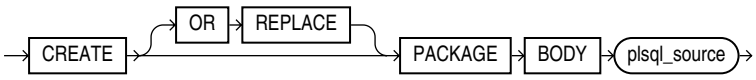
Oracle Database のプリコンパイラ・プログラム内に CREATE PACKAGE BODY 文を埋め込む場合、キーワード END-EXEC に続けて、各言語の埋込み SQL 文の終了記号を記述して文を終了する必要があります。

参照 : 『Oracle Database PL/SQL 言語リファレンス』

構文

パッケージ本体は PL/SQL を使用して定義されます。このため、このマニュアルの構文図では SQL キーワードのみを示します。PL/SQL の構文、セマンティクスおよび例については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

***create_package_body*::=**



(*plsql_source* については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。)

セマンティクス

OR REPLACE

OR REPLACE を指定すると、既存のパッケージ本体を再作成できます。この句を指定した場合、パッケージに対して付与されていたオブジェクト権限を削除、再作成および再付与しなくても、既存のパッケージの本体を変更できます。パッケージ本体を変更した場合、その本体は自動的に再コンパイルされます。

再定義したパッケージに対して権限が付与されていたユーザーは、権限が再付与されなくてもそのパッケージにアクセスできます。

> **参照：** パッケージ本体の再コンパイルについては、11-27 ページの「ALTER PACKAGE」を参照してください。

plsql_source

plsql_source の構文およびセマンティクスについては、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

CREATE PFILE

用途

CREATE PFILE 文を使用すると、バイナリのサーバー・パラメータ・ファイルまたは現行のメモリー内パラメータ設定をテキストの初期化パラメータ・ファイルにエクスポートできます。テキストのパラメータ・ファイルの作成は、データベースで使用している現行のパラメータ設定リストの取得に便利です。また、テキスト・エディタで簡単に編集でき、CREATE SPFILE 文を使用してサーバー・パラメータ・ファイルに変換して戻すこともできます。

この文が正常に実行されると、サーバーにテキストのパラメータ・ファイルが作成されます。Oracle Real Application Clusters 環境では、すべてのインスタンスのすべてのパラメータ設定が含まれます。サーバー・パラメータ・ファイルのパラメータ設定と同じ行のコメントも含まれます。

参照 :

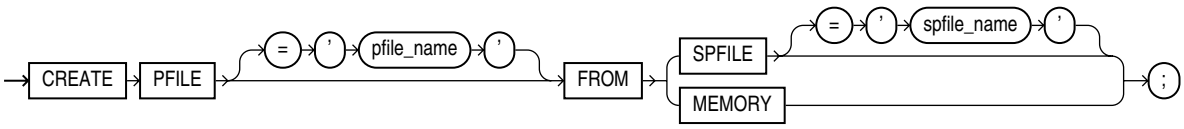
- サーバー・パラメータ・ファイルの詳細は、15-68 ページの「CREATE SPFILE」を参照してください。
- テキストの初期化パラメータ・ファイルおよびバイナリのサーバー・パラメータ・ファイルの詳細は、『Oracle Database 管理者ガイド』を参照してください。
- Oracle Real Application Clusters 環境でのサーバー・パラメータ・ファイルの使用については、『Oracle Real Application Clusters 管理およびデプロイメント・ガイド』を参照してください。

前提条件

この文を実行するには、SYSDBA ロールまたは SYSOPER ロールが必要です。この文は、インスタンスの起動前と起動後のいずれかで実行できます。

構文

***create_pfile*::=**



セマンティクス

pfile_name

作成するテキストのパラメータ・ファイル名を指定します。*pfile_name* を指定しないと、プラットフォーム固有のデフォルトの初期化パラメータ・ファイル名が使用されます。*pfile_name* には、パス接頭辞を含めることができます。パス接頭辞を指定しない場合は、データベースによってデフォルトの格納場所（プラットフォームによって異なる）のパス接頭辞が追加されます。

spfile_name

テキストのファイルを作成する元となるバイナリのサーバー・パラメータ・ファイル名を指定します。

- *spfile_name* を指定する場合、そのファイルがサーバーに存在する必要があります。ファイルがオペレーティング・システムのサーバー・パラメータ・ファイルのデフォルト・ディレクトリに存在しない場合、フルパス名を指定する必要があります。

- *spfile_name* を指定しない場合は、現在インスタンスに関連付けられている spfile がデータベースで使用されます。通常、これは起動時に使用された spfile です。インスタンスに spfile が関連付けられていない場合は、プラットフォーム固有のデフォルトのサーバー・パラメータ・ファイル名が検索されます。このファイルが存在しない場合は、エラーが戻されます。

  参照： デフォルトのパラメータ・ファイル名については、オペレーティング・システム固有のドキュメントを参照してください。

MEMORY

MEMORY を指定すると、現行のシステム全体のパラメータ設定を使用して pfile を作成できます。RAC 環境では、作成されたファイルには各インスタンスからのパラメータ設定が含まれます。

例

パラメータ・ファイルの作成例： 次の例は、バイナリのサーバー・パラメータ・ファイル s_params.ora からテキストのパラメータ・ファイル my_init.ora を作成します。

```
CREATE PFILE = 'my_init.ora' FROM SPFILE = 's_params.ora';
```

注意： 通常、オペレーティング・システムのパラメータ・ファイルには、フルパスのファイル名を指定する必要があります。パスについては、ご使用のオペレーティング・システムの Oracle マニュアルを参照してください。

CREATE PROCEDURE

用途

パッケージは PL/SQL を使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は『Oracle Database PL/SQL 言語リファレンス』を参照してください。

CREATE PROCEDURE 文を使用すると、スタンドアロンのストアド・プロシージャまたはコール仕様を作成できます。

プロシージャとは、名前によってコールできる PL/SQL 文の集合です。**コール仕様**は、SQL および PL/SQL からコールできるように、Java メソッドまたは第三世代言語（3GL）ルーチンを宣言します。コール仕様は、コールされたときに起動する Java メソッドを Oracle Database に指示します。また、引数および戻り値に対して実行する型変換もデータベースに指示します。

ストアド・プロシージャには、開発、整合性、セキュリティ、パフォーマンスおよびメモリー割当ての面でいくつかのメリットがあります。

参照：

- ストアド・プロシージャのコール方法など、ストアド・プロシージャの詳細および外部プロシージャの登録については、『Oracle Database アドバンスト・アプリケーション開発者ガイド』を参照してください。
- プロシージャと類似点が多いファンクションの詳細は、14-48 ページの「CREATE FUNCTION」を参照してください。
- パッケージの作成の詳細は、15-39 ページの「CREATE PACKAGE」を参照してください。CREATE PROCEDURE 文では、スタンドアロンのスキーマ・オブジェクトとしてプロシージャが作成されます。また、プロシージャをパッケージの一部としても作成できます。
- スタンドアロン・プロシージャの変更および削除については、11-28 ページの「ALTER PROCEDURE」および 17-57 ページの「DROP PROCEDURE」を参照してください。
- 共有ライブラリについては、15-2 ページの「CREATE LIBRARY」を参照してください。

前提条件

自分のスキーマ内にプロシージャを作成または再作成する場合は、CREATE PROCEDURE システム権限が必要です。他のユーザーのスキーマ内にプロシージャを作成または再作成する場合は、CREATE ANY PROCEDURE システム権限が必要です。

コール仕様を起動する場合、その他の権限が必要になることがあります。たとえば、C コール仕様には、C ライブラリに対する EXECUTE オブジェクト権限が必要です。

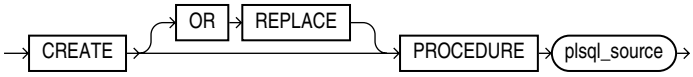
Oracle プリコンパイラ・プログラム内に CREATE PROCEDURE 文を埋め込む場合、キーワード END-EXEC に続けて、各言語の埋込み SQL 文の終了記号を記述して文を終了する必要があります。

参照： 詳細は、『Oracle Database PL/SQL 言語リファレンス』または『Oracle Database Java 開発者ガイド』を参照してください。

構文

パッケージ本体は PL/SQL を使用して定義されます。このため、このマニュアルの構文図では SQL キーワードのみを示します。PL/SQL の構文、セマンティクスおよび例については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

***create_procedure*::=**



（*plsql_source* については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。）

セマンティクス

OR REPLACE

OR REPLACE を指定すると、既存のプロシージャを再作成できます。この句を指定した場合、プロシージャに付与されているオブジェクト権限を削除、再作成および再付与しなくても、既存のプロシージャの定義を変更できます。プロシージャを再定義した場合、そのプロシージャは自動的に再コンパイルされます。

再定義したプロシージャに対して権限が付与されていたユーザーは、権限が再付与されなくてもそのプロシージャにアクセスできます。

ファンクション索引がパッケージに依存している場合、索引に DISABLED のマークが付きます。

> **参照：** プロシージャの再コンパイルについては、11-28 ページの「ALTER PROCEDURE」を参照してください。

plsql_source

plsql_source の構文およびセマンティクスについては、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

CREATE PROFILE

注意： リソース制限を設定する場合、この SQL 文ではなく、データベース・リソース・マネージャを使用することをお薦めします。データベース・リソース・マネージャを使用すると、リソース使用の管理および監査を柔軟に行うことができます。データベース・リソース・マネージャの詳細は、『Oracle Database 管理者ガイド』を参照してください。

用途

CREATE PROFILE 文を使用すると、**プロファイル**を作成できます。プロファイルとは、データベース・リソースの制限の設定です。あるユーザーに対してプロファイルを割り当てた場合、そのユーザーは、その割当て制限を超えることはできません。

参照： パスワード管理およびパスワード保護の詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。

前提条件

プロファイルを作成する場合、CREATE PROFILE システム権限が必要です。

次の方法でユーザーに対するリソース制限を指定します。

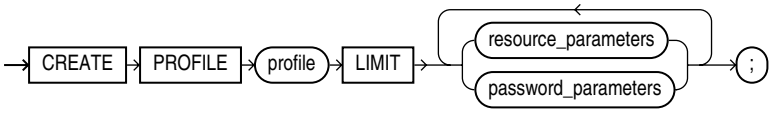
- ALTER SYSTEM 文または初期化パラメータ RESOURCE_LIMIT で動的にリソース制限を使用可能にします。このパラメータは、パスワード・リソースには適用されません。パスワード・リソースは、常に使用可能です。
- CREATE PROFILE 文を使用して、制限を定義するプロファイルを作成します。
- CREATE USER または ALTER USER 文を使用して、プロファイルを割り当てます。

参照：

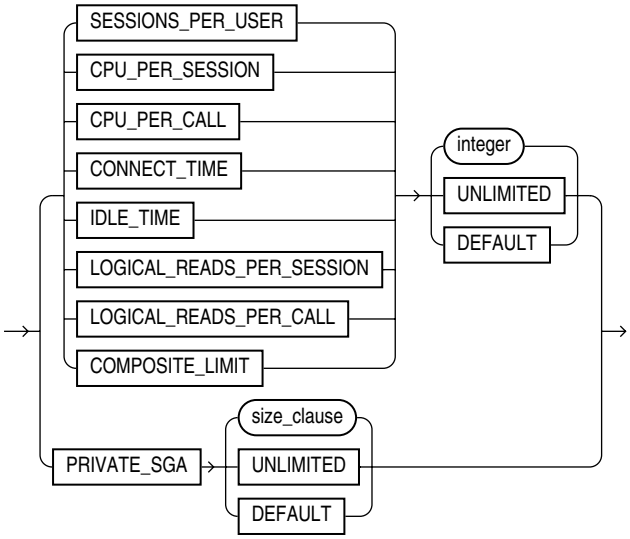
- 動的にリソース制限を使用可能にする場合の詳細は、11-52 ページの「ALTER SYSTEM」を参照してください。
- RESOURCE_LIMIT パラメータについては、『Oracle Database リファレンス』を参照してください。
- プロファイルについては、17-7 ページの「CREATE USER」および 13-5 ページの「ALTER USER」を参照してください。

構文

***create_profile*::=**

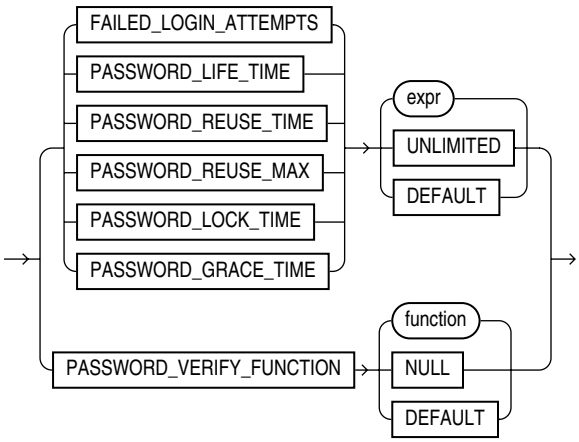


resource_parameters::=



（8-42 ページの size_clause::= を参照）

password_parameters::=



セマンティクス

profile

作成するプロファイルの名前を指定します。プロファイルを使用した場合、ユーザーが使用可能なデータベース・リソースを 1 つのコールまたは 1 つのセッションごとに制限できます。

Oracle Database では、次の方法でリソース制限を適用します。

- `CONNECT_TIME` または `IDLE_TIME` で指定したセッション・リソース制限を超えた場合、現行のトランザクションが自動的にロールバックされ、セッションが終了します。ユーザー・プロセスで次にコールを実行すると、エラーが戻ります。
- 他のセッション・リソースに対する制限を超える処理を実行しようとした場合、その処理が自動的に中断され、現行の文がロールバックされ、すぐにエラーが戻されます。この後、ユーザーは、現行のトランザクションをコミットまたはロールバックできます。そして、セッションを終了する必要があります。
- 1 つのコールに対する制限を超える処理を実行しようとした場合、その処理が自動的に中断され、現行の文がロールバックされ、エラーが戻されます。この場合、現行のトランザクションは有効です。

注意 :

- 日を単位として、時間を制限するすべてのパラメータに対して、分数で日数を指定できます。たとえば、1 時間は 1/24、1 分は 1/1440 になります。
- リソース制限を使用可能にしているかどうかにかかわらず、ユーザーに対してリソース制限を指定できます。ただし、Oracle Database では、実際にその制限が使用可能になってからリソース制限が適用されます。

参照 : 「プロファイルの作成例 :」(15-51 ページ)

UNLIMITED

リソース・パラメータで UNLIMITED を指定すると、このプロファイルを割り当てられたユーザーは無制限にリソースを使用できます。パスワード・パラメータで UNLIMITED を指定した場合は、パラメータに制限が設定されていないことを示します。

DEFAULT

DEFAULT を指定すると、このプロファイルでリソースの制限を省略できます。このプロファイルを割り当てられたユーザーは、DEFAULT プロファイルで指定した対象リソースに対する制限を受けます。DEFAULT プロファイルは、最初に無制限のリソースを定義します。ALTER PROFILE 文でこの制限を変更できます。

明示的にプロファイルが割り当てられていないユーザーは、DEFAULT プロファイルに定義されている制限を受けます。ユーザーに明示的に割り当てられているプロファイルでリソースに対する制限が省略されている場合、または制限に対して DEFAULT が指定されている場合、ユーザーは DEFAULT プロファイルで定義されているリソースに関する制限を受けます。

resource_parameters

SESSIONS_PER_USER ユーザーを制限する同時セッションの数を指定します。

CPU_PER_SESSION 1 セッション当たりの CPU 時間制限を指定します。この値は 100 分の 1 秒単位で指定します。

CPU_PER_CALL 1 コール (解析、実行またはフェッチ) 当たりの CPU 時間制限を指定します。この値は 100 分の 1 秒単位で指定します。

CONNECT_TIME 1 セッション当たりの合計経過時間制限を指定します。この値は分単位で指定します。

IDLE_TIME セッション中の連続的な非活動時間の長さを制限します。この値は分単位で指定します。長時間実行の問合せなどの処理は、この制限を受けません。

LOGICAL_READS_PER_SESSION メモリーおよびディスクから読み込まれるブロックなど、1 セッション中に読み込まれるデータ・ブロックの数の制限を指定します。

LOGICAL_READS_PER_CALL SQL 文 (解析、実行またはフェッチ) を処理する 1 つのコールで読み込まれるデータ・ブロックの数の制限を指定します。

PRIVATE_SGA 1 つのセッションでシステム・グローバル領域 (SGA) の共有プール内に割り当てることができるプライベート領域の大きさを指定します。この句の詳細は、8-42 ページの「size_clause」を参照してください。

注意： この制限は、共有サーバー・アーキテクチャを使用している場合のみに適用されます。SGA 内のセッション用のプライベート領域には、プライベート SQL および PL/SQL 領域が含まれますが、共有 SQL および PL/SQL 領域は含まれません。

COMPOSITE_LIMIT 1 セッション当たりのリソースの総コストを**サービス単位**で指定します。サービス単位の合計は、CPU_PER_SESSION、CONNECT_TIME、LOGICAL_READS_PER_SESSION および PRIVATE_SGA の重み付き合計として計算されます。

参照：

- セッション・リソースの重み付けを指定する方法については、11-32 ページの「ALTER RESOURCE COST」を参照してください。
- 「プロファイルのリソース制限の設定の例：」（15-51 ページ）

password_parameters

次の句を使用すると、パスワード・パラメータを設定できます。時間の長さを設定するパラメータは、日数で解析されます。テストのために、分（*n*/1440）または秒（*n*/86400）を指定できます。

FAILED_LOGIN_ATTEMPTS ユーザー・アカウントがロックされる前に、そのアカウントへのログインに失敗できる回数を指定します。この句を指定しない場合、10 日がデフォルトになります。

PASSWORD_LIFE_TIME 同じパスワードを認証に使用できる日数を制限します。PASSWORD_GRACE_TIME の値とともに設定した場合、猶予期間内にパスワードを変更しないと、そのパスワードは使用できなくなり、それ以降の接続は拒否されます。この句を指定しない場合、180 日がデフォルトになります。

PASSWORD_REUSE_TIME および PASSWORD_REUSE_MAX これら 2 つのパラメータは、両方を組み合せて設定する必要があります。PASSWORD_REUSE_TIME は、パスワードを再利用できない日数を指定します。PASSWORD_REUSE_MAX は、現行のパスワードを再利用する前に必要な、パスワードの変更回数を指定します。これらのパラメータを有効にするには、両方のパラメータに整数を指定する必要があります。

- 両方のパラメータに整数を指定すると、PASSWORD_REUSE_TIME に指定した日数の間は、PASSWORD_REUSE_MAX に指定した回数パスワードが変更されるまで、パスワードを再利用できません。

  たとえば、PASSWORD_REUSE_TIME に 30 を指定し、PASSWORD_REUSE_MAX に 10 を指定した場合、パスワードが 10 回変更されていると、30 日後にパスワードを再利用できます。
- これらのパラメータのいずれかに整数を指定し、他方に UNLIMITED を指定すると、パスワードを再利用できません。
- いずれかのパラメータに DEFAULT を指定すると、DEFAULT プロファイルに定義された値が使用されます。DEFAULT プロファイルでは、すべてのパラメータはデフォルトで UNLIMITED に設定されます。DEFAULT プロファイルのデフォルト設定の UNLIMITED を変更していない場合、パラメータの値は UNLIMITED として扱われます。
- 両方のパラメータを UNLIMITED に設定すると、これらのパラメータは無視されます。これは、両方のパラメータを指定しない場合のデフォルトです。

PASSWORD_LOCK_TIME ログインが指定された回数連続して失敗した場合、アカウントがロックされる日数を指定します。この句を指定しない場合、1 日がデフォルトになります。

PASSWORD_GRACE_TIME　警告が出され、ログインが許可される猶予期間の日数を指定します。この句を指定しない場合、7 日がデフォルトになります。

PASSWORD_VERIFY_FUNCTION　PASSWORD_VERIFY_FUNCTION 句を使用すると、PL/SQL の複雑なパスワード検証スクリプトを CREATE PROFILE 文の引数として渡すことができます。Oracle Database にはデフォルトのスクリプトがありますが、ユーザー固有のルーチンを作成することも、サード・パーティのソフトウェアを使用することもできます。

- *function* には、複雑なパスワード検証ルーチンの名前を指定します。
- NULL を指定すると、パスワードの検証が行われないことを示します。

パスワード・パラメータに *expr* を指定する場合、スカラー副問合せ式を除くすべての形式の式を指定できます。

参照：「プロファイルのパスワード制限の設定例 :」（15-52 ページ）

例

プロファイルの作成例：　次の文は、プロファイル new_profile を作成します。

```
CREATE PROFILE new_profile
  LIMIT PASSWORD_REUSE_MAX 10
        PASSWORD_REUSE_TIME 30;
```

プロファイルのリソース制限の設定の例：　次の文は、プロファイル app_user を作成します。

```
CREATE PROFILE app_user LIMIT
   SESSIONS_PER_USER          UNLIMITED
   CPU_PER_SESSION            UNLIMITED
   CPU_PER_CALL               3000
   CONNECT_TIME               45
   LOGICAL_READS_PER_SESSION  DEFAULT
   LOGICAL_READS_PER_CALL     1000
   PRIVATE_SGA                15K
   COMPOSITE_LIMIT            5000000;
```

ユーザーに app_user プロファイルを割り当てた場合、そのユーザーは、後続のセッションで次の制限を受けます。

- ユーザーは、無制限に同時セッションを使用できます。
- 1 つのセッションにおいて、ユーザーは CPU 時間を無制限に使用できます。
- ユーザーが作成した 1 つのコールで消費できる CPU 時間は 30 秒以下です。
- 1 つのセッションで継続できる時間は 45 分以下です。
- 1 つのセッションでメモリーおよびディスクから読み込むデータ・ブロック数は、DEFAULT プロファイルで指定した制限を受けます。
- ユーザーが実行した 1 つのコールで、メモリーまたはディスクから読み込むことができるデータ・ブロック数の合計は 1000 以下です。
- 1 つのセッションで割り当てることができる SGA 内のメモリーは、15KB 以下です。
- 1 つのセッションのリソースの総コストは、サービス単位で 500 万を超えることはできません。リソースの総コストの計算式は、ALTER RESOURCE COST 文で指定します。
- app_user プロファイルでは、IDLE_TIME およびパスワードに対する制限が指定されていないため、ユーザーは DEFAULT プロファイルに指定されているこれらのリソースの制限を受けます。

プロファイルのパスワード制限の設定例： 次の文は、app_user2 プロファイルに、パスワード制限値を設定して作成します。

```
CREATE PROFILE app_user2 LIMIT
   FAILED_LOGIN_ATTEMPTS 5
   PASSWORD_LIFE_TIME 60
   PASSWORD_REUSE_TIME 60
   PASSWORD_REUSE_MAX 5
   PASSWORD_VERIFY_FUNCTION verify_function
   PASSWORD_LOCK_TIME 1/24
   PASSWORD_GRACE_TIME 10;
```

この例では、Oracle Database のデフォルトのパスワード検証ファンクション verify_function を使用します。この検証ファンクションの使用方法または独自の検証ファンクションの設計方法については、『Oracle Database セキュリティ・ガイド』を参照してください。

CREATE RESTORE POINT

用途

CREATE RESTORE POINT 文を使用すると、**リストア・ポイント**（タイムスタンプまたはデータベースの SCN に関連付けられた名前）を作成できます。リストア・ポイントを使用すると、表またはデータベースをリストア・ポイントによって指定された時点にフラッシュバックできます。その際、SCN やタイムスタンプを指定する必要はありません。リストア・ポイントは、バックアップやデータベースの複製など、様々な Recovery Manager 操作でも有効です。Recovery Manager を使用して、アーカイブ・バックアップの実装プロセスにリストア・ポイントを作成できます。

参照 :

- リストア・ポイントおよび保証付きリストア・ポイントの作成と使用、データベースの複製およびアーカイブ・バックアップの詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- リストア・ポイントの使用方法および削除方法については、18-22 ページの「FLASHBACK DATABASE」、18-25 ページの「FLASHBACK TABLE」および 17-59 ページの「DROP RESTORE POINT」を参照してください。

前提条件

通常のリストア・ポイントを作成するには、SELECT ANY DICTIONARY 権限または FLASHBACK ANY TABLE 権限のいずれかが必要です。保証付きリストア・ポイントを作成するには、SYSDBA システム権限が必要です。

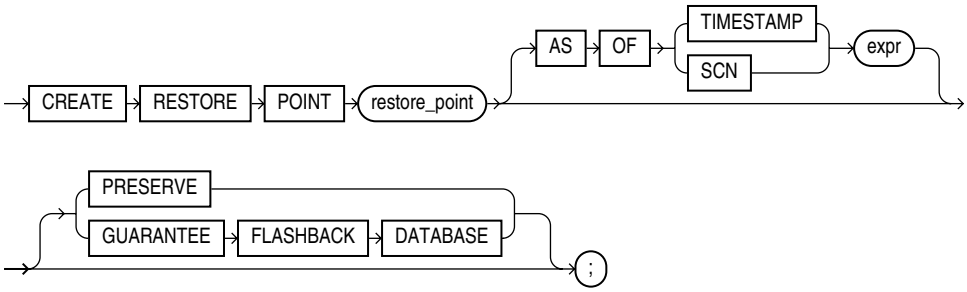
リストア・ポイントを表示または使用するには、SELECT ANY DICTIONARY または FLASHBACK ANY TABLE のいずれかのシステム権限か、SELECT_CATALOG_ROLE ロールが必要です。

リストア・ポイントは、プライマリ・データベースまたはスタンバイ・データベースに作成できます。データベースは、オープンされている状態でも、マウントされているがオープンされていない状態でもかまいません。データベースがマウントされている場合、それが物理スタンバイ・データベースでなければ、整合性を保持して停止されてからマウントされている必要があります。

保証付きリストア・ポイントを作成する場合は、その前に、フラッシュ・リカバリ領域を作成しておく必要があります。フラッシュバック・データベースを使用可能にしておかなくても、リストア・ポイントは作成できます。ただし、フラッシュバック・データベースが使用可能になっていない場合、このデータベースに作成する最初の保証付きリストア・ポイントは、データベースのマウント時に作成する必要があります。保証付きリストア・ポイントを作成する場合、データベースは ARCHIVELOG モードになっていることが必要です。

構文

***create_restore_point*::=**



セマンティクス

restore_point

リストア・ポイントの名前を指定します。名前は、最大 128 文字の文字値です。

データベースには、最大 2048 個のリストア・ポイントを保持できます。リストア・ポイントは、少なくとも初期化パラメータ CONTROL_FILE_RECORD_KEEP_TIME で指定された日数はデータベース内に保持されます。そのパラメータのデフォルト値は 7 日です。保証付きおよび保存済のリストア・ポイントは、ユーザーが明示的に削除するまでデータベース内に保持されます。

PRESERVE も GUARANTEE FLASHBACK DATABASE も指定しない場合、結果のリストア・ポイントによって、DB_FLASHBACK_RETENTION_TARGET 初期化パラメータで設定された期間内のリストア・ポイントにデータベースをフラッシュバックできます。そのようなリストア・ポイントは、データベースによって自動的に管理されます。リストア・ポイントの数が、前述の *restore_point* で定められた最大数に達すると、最も古いリストア・ポイントが自動的に削除されます。一定の状況では長期バックアップのリストアに使用するために、リストア・ポイントは Recovery Manager リカバリ・カタログに保持されます。DROP RESTORE POINT 文を使用して、リストア・ポイントを明示的に削除することもできます。

AS OF 句

この句を使用すると、過去の指定した日時または SCN でリストア・ポイントを作成できます。TIMESTAMP を指定する場合、*expr* は過去の日時となる有効な日時式である必要があります。SCN を指定する場合、*expr* は過去のデータベースの有効な SCN である必要があります。どちらの場合にも、*expr* はデータベースの現行のインカネーションの日時または SCN を参照する必要があります。

PRESERVE

PRESERVE を指定すると、リストア・ポイントを明示的に削除する必要があることを指定できます。このようなリストア・ポイントは、フラッシュバック履歴機能で使用するために作成すると有効です。

GUARANTEE FLASHBACK DATABASE

保証付きリストア・ポイントの場合、初期化パラメータ DB_FLASHBACK_RETENTION_TARGET の設定とは無関係に、データベースをリストア・ポイントに確定的にフラッシュバックできます。フラッシュバック機能が保証されるかどうかは、フラッシュ・リカバリ領域に十分な使用可能スペースがあるかどうかによって決まります。

保証付きリストア・ポイントで保証されるのは、データベースを保証付きリストア・ポイントにフラッシュバックするのに十分なフラッシュバック・ログが、データベースに保持されるということのみです。すべての表を同じリストア・ポイントにフラッシュバックするための UNDO がデータベースに保持されるかどうかは保証されません。

保証付きリストア・ポイントは、常に保存されます。DROP RESTORE POINT 文を使用してユーザーが明示的に削除する必要があります。このリストア・ポイントは、期限切れになることはありません。保証付きリストア・ポイントは、フラッシュ・リカバリ領域内の領域を大量に使用する場合があります。このため、保証付きリストア・ポイントは十分に検討したうえで作成することをお薦めします。

例

リストア・ポイントを作成して使用する例 : 次の例では、通常のリストア・ポイントを作成し、表を更新してから、変更した表をリストア・ポイントにフラッシュバックします。この例は、ユーザー hr がそれぞれの文を使用する適切なシステム権限を持っていることを前提としています。

```
CREATE RESTORE POINT good_data;

SELECT salary FROM employees WHERE employee_id = 108;

    SALARY
----------
     12000

UPDATE employees SET salary = salary*10
   WHERE employee_id = 108;

SELECT salary FROM employees
   WHERE employee_id = 108;

    SALARY
----------
    120000

COMMIT;

FLASHBACK TABLE employees TO RESTORE POINT good_data;

SELECT salary FROM employees
   WHERE employee_id = 108;

    SALARY
----------
     12000
```

CREATE ROLE

用途

CREATE ROLE 文を使用すると、**ロール**を作成できます。ロールは、ユーザーまたは他のロールに付与できる権限の集合です。ロールを使用してデータベース権限を管理できます。ロールに権限を追加したうえで、ユーザーにそのロールを付与できます。その後、ユーザーはロールを使用可能にし、そのロールによって付与された権限を使用できるようになります。

ロールには、そのロールに付与されたすべての権限、およびそのロールに付与された他のロールのすべての権限が含まれています。新しく作成されたロールには、ロールや権限は付与されていません。GRANT 文を使用して、ロールに様々な権限を追加します。

NOT IDENTIFIED、IDENTIFIED EXTERNALLY または BY *password* ロールを作成した場合、そのロールは ADMIN OPTION 付きで付与されます。ただし、ロール IDENTIFIED GLOBALLY を作成した場合、ロールは付与されません。

参照：

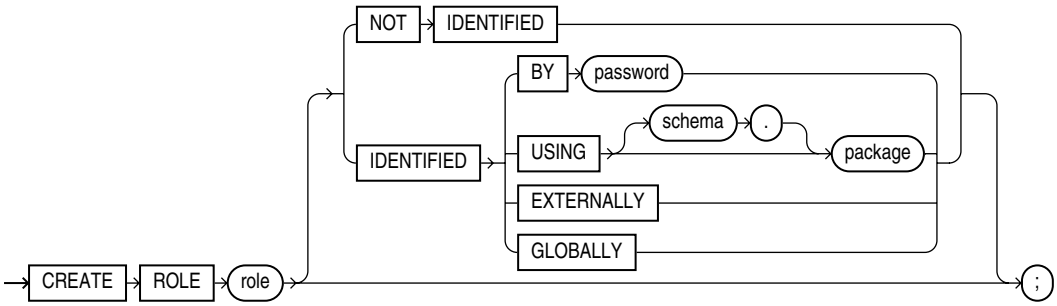
- ロールの付与については、18-31 ページの「GRANT」を参照してください。
- ロールを使用可能にする場合は、13-5 ページの「ALTER USER」を参照してください。
- データベースのロールの変更または削除については、11-34 ページの「ALTER ROLE」および 17-60 ページの「DROP ROLE」を参照してください。
- 現行のセッションに対するロールの使用可能化および使用禁止化については、19-51 ページの「SET ROLE」を参照してください。
- ロールの概要は、『Oracle Database セキュリティ・ガイド』を参照してください。

前提条件

CREATE ROLE システム権限が必要です。

構文

***create_role*::=**



セマンティクス

role

作成するロールの名前を指定します。データベース・キャラクタ・セットにマルチバイト文字がサポートされている場合でも、ロールにはシングルバイト文字を 1 つ以上使用することをお薦めします。各ユーザーに対し、一度に使用可能にできるユーザー定義のロールの最大数は 148 です。

配布メディアで提供されている SQL スクリプトには、いくつかのロールが定義されています。

> **参照：** 事前定義済のロールのリストについては、18-31 ページの「GRANT」を、ユーザーに対するロールの使用可能化および使用禁止化については、19-51 ページの「SET ROLE」を参照してください。

NOT IDENTIFIED 句

NOT IDENTIFIED を指定すると、指定するロールがデータベースによって認可され、パスワードを入力しなくてもこのロールを使用可能にできます。

IDENTIFIED 句

IDENTIFIED 句を使用すると、SET ROLE 文によってロールを使用可能にする前に、指定したメソッドによってユーザーが認可される必要があります。

BY *password* BY *password* 句を使用すると、**ローカル・ロール**を作成できます。また、ロールを使用可能にするときに、パスワードを指定する必要があることを指定できます。データベース・キャラクタ・セットにマルチバイト文字が含まれている場合でも、データベース・キャラクタ・セットのシングルバイト文字のみでパスワードを指定できます。

USING *package* USING *package* 句を使用すると、**保護アプリケーション・ロール**を作成できます。保護アプリケーション・ロールとは、認可済パッケージのみを使用するアプリケーションよって使用可能になるロールです。*schema* を指定しない場合、パッケージが自分のスキーマ内にあるとみなされます。

注意： ユーザーにロールを付与する場合、ロールはユーザーのデフォルトのロールとして付与されるため、ログインするとすぐに使用可能になります。アプリケーション・ロールのセキュリティを確保するには、ロールをデフォルトのロールにしないようにする必要があります。ユーザーにアプリケーション・ロールを付与した直後に、DEFAULT ROLE ALL EXCEPT *role* 句を指定した ALTER USER 文を発行し、アプリケーション・ロールを指定します。これによって、このユーザーによる以降のログイン時に、認可済パッケージを使用するアプリケーションのみがこのロールを使用可能にできるというルールが適用されます。

> **参照：** 保護アプリケーション・ロールの作成については、『Oracle Database セキュリティ・ガイド』を参照してください。

EXTERNALLY EXTERNALLY を指定すると、**外部ロール**を作成できます。外部ユーザーは、ロールを使用可能にする前に、オペレーティング・システムやサード・パーティ・サービスなどの外部サービスによって認可されている必要があります。

オペレーティング・システムによっては、ユーザーがオペレーティング・システムに対してパスワードを指定しないと、ロールが使用可能にできない場合もあります。

GLOBALLY GLOBALLY を指定すると、**グローバル・ロール**を作成できます。グローバル・ユーザーは、ログイン時にロールの使用が可能になる前に、エンタープライズ・ディレクトリ・サービスによってロールの使用を認可されている必要があります。

NOT IDENTIFIED 句および IDENTIFIED 句を両方とも省略した場合、ロールには NOT IDENTIFIED がデフォルト値として使用されます。

例

ロールの作成例： 次の文は、ロール dw_manager を作成します。

```
CREATE ROLE dw_manager;
```

この後に dw_manager ロールを付与されるユーザーは、このロールに付与されているすべての権限を継承します。

次の例のように、パスワードの指定によってロールにセキュリティのレイヤーを追加できます。

```
CREATE ROLE dw_manager
   IDENTIFIED BY warehouse;
```

この後、dw_manager ロールを付与されたユーザーは、パスワード warehouse を指定して、SET ROLE 文でロールを使用可能にする必要があります。

次の文は、グローバル・ロール warehouse_user を作成します。

```
CREATE ROLE warehouse_user IDENTIFIED GLOBALLY;
```

次の文は、同じロールを外部ロールとして作成します。

```
CREATE ROLE warehouse_user IDENTIFIED EXTERNALLY;
```

CREATE ROLLBACK SEGMENT

> **注意：** ロールバック・セグメントを使用せずに、自動 UNDO 管理モードでデータベースを実行することを強くお薦めします。ロールバック・セグメントは、以前のバージョンの Oracle Database との互換性に必要な場合以外は使用しないでください。自動 UNDO 管理の詳細は、『Oracle Database 管理者ガイド』を参照してください。

用途

CREATE ROLLBACK SEGMENT 文を使用すると、**ロールバック・セグメント**を作成できます。ロールバック・セグメントとは、トランザクションによる変更を元に戻す（取り消す）ために必要なデータを格納する際に Oracle Database が使用するオブジェクトです。

ここでは、データベースがロールバック UNDO モードで実行されている（初期化パラメータ UNDO_MANAGEMENT に MANUAL を設定、またはすべて設定しない）ことを前提としています。データベースが自動 UNDO モードで実行されている場合（初期化パラメータ UNDO_MANAGEMENT にデフォルト値の AUTO を設定）、ユーザー作成のロールバック・セグメントには関連がなくなります。

また、データベースにローカル管理 SYSTEM 表領域がある場合、ディクショナリ管理表領域にロールバック・セグメントは作成できません。かわりに、自動 UNDO 管理機能を使用するか、またはローカル管理表領域を作成して、ロールバック・セグメントを保持する必要があります。

> **注意：** 1 つの表領域に複数のロールバック・セグメントを作成できます。一般に、複数のロールバック・セグメントがあると、パフォーマンスが向上します。
>
> - 表領域にロールバック・セグメントを追加する場合、表領域は必ずオンラインである必要があります。
> - ロールバック・セグメントを作成した場合、最初はオフライン状態になります。そのロールバック・セグメントを Oracle Database インスタンスでトランザクション可能にする場合、ALTER ROLLBACK SEGMENT 文を使用してオンラインの状態にしてください。データベース起動時に自動的にオンライン状態にする場合、初期化パラメータ ROLLBACK_SEGMENT の値にそのセグメントの名前を追加してください。

SYSTEM 以外の表領域のオブジェクトを使用する場合は、次の注意事項があります。

- UNDO 用にロールバック・セグメントを使用するには、1 つ以上のロールバック・セグメント（SYSTEM ロールバック・セグメント以外）がオンラインである必要があります。
- 自動 UNDO モードでデータベースを実行するには、1 つ以上の UNDO 表領域がオンラインである必要があります。

参照：

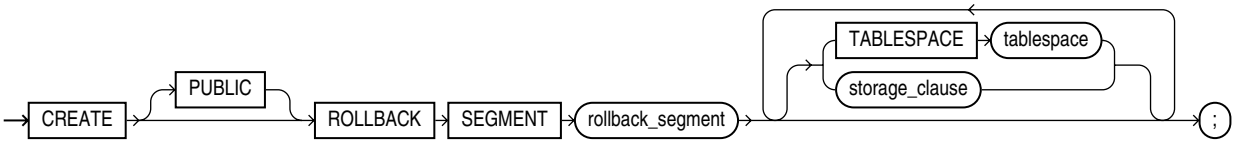
- ロールバック・セグメントの変更については、11-36 ページの「ALTER ROLLBACK SEGMENT」を参照してください。
- ロールバック・セグメントの削除については、17-61 ページの「DROP ROLLBACK SEGMENT」を参照してください。
- UNDO_MANAGEMENT パラメータの詳細は、『Oracle Database リファレンス』を参照してください。
- 自動 UNDO モードの詳細は、『Oracle Database 管理者ガイド』を参照してください。

前提条件

ロールバック・セグメントを作成するには、CREATE ROLLBACK SEGMENT システム権限が必要です。

構文

***create_rollback_segment*::=**



(8-43 ページの storage_clause を参照)

セマンティクス

PUBLIC

PUBLIC を指定すると、ロールバック・セグメントがパブリックとなり、すべてのインスタンスに対して使用可能にできます。この句を省略した場合、ロールバック・セグメントはプライベートになり、インスタンスの初期化パラメータ ROLLBACK_SEGMENTS で指定したインスタンスに対してのみ使用可能になります。

rollback_segment

作成するロールバック・セグメントの名前を指定します。

TABLESPACE

TABLESPACE 句を使用すると、ロールバック・セグメントが作成される表領域を指定できます。この句を省略した場合、ロールバック・セグメントは SYSTEM 表領域に作成されます。

注意： Oracle Database は、頻繁にロールバック・セグメントにアクセスする必要があります。そのため、明示的または（この句を省略して）暗黙的に、ロールバック・セグメントを SYSTEM 表領域に作成しないことをお薦めします。さらに、ロールバック・セグメントが含まれている表領域の競合を回避するために、この表領域には表、索引などの他のオブジェクトを含めないでください。また、エクステントの割当ておよび割当て解除は、最小限に抑える必要があります。

そのためには、自動割当てを使用禁止にしてローカル管理表領域にロールバック・セグメントを作成します。ローカル管理表領域とは、EXTENT MANAGEMENT LOCAL 句に UNIFORM を指定して作成された表領域です。AUTOALLOCATE 設定はサポートされません。

参照： 「CREATE TABLESPACE」(16-71 ページ)

storage_clause

storage_clause を使用すると、ロールバック・セグメントの記憶特性を指定できます。

- *storage_clause* の OPTIMAL パラメータは、ロールバック・セグメントにのみ適用されるので、特に重要です。
- *storage_clause* の PCTINCREASE パラメータは、CREATE ROLLBACK SEGMENT では指定できません。

参照： 「storage_clause」(8-43 ページ)

例

ロールバック・セグメントの作成例： 次の文は、適切に構成された表領域にデフォルトの記憶域値でロールバック・セグメントを作成します。

```
CREATE TABLESPACE rbs_ts
   DATAFILE 'rbs01.dbf' SIZE 10M
   EXTENT MANAGEMENT LOCAL UNIFORM SIZE 100K;

/* This example and the next will fail if your database is in
   automatic undo mode.
*/
CREATE ROLLBACK SEGMENT rbs_one
   TABLESPACE rbs_ts;
```

前述の文は、次の文と同じ結果になります。

```
CREATE ROLLBACK SEGMENT rbs_one
   TABLESPACE rbs_ts
   STORAGE
   ( INITIAL 10K
     NEXT 10K
     MAXEXTENTS UNLIMITED );
```

CREATE SCHEMA

用途

CREATE SCHEMA 文を使用すると、複数表およびビューを作成し、自分のスキーマ内に 1 つのトランザクションで複数の権限を付与できます。

CREATE SCHEMA 文を実行すると、挿入されている個々の文が実行されます。すべての文が正常に実行された場合、そのトランザクションがコミットされます。文の結果が 1 つでもエラーになった場合は、すべての文がロールバックされます。

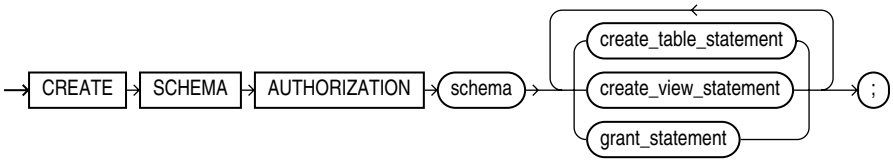
注意： この文では、実際にスキーマが作成されるわけではありません。ユーザーを作成すると、自動的にスキーマが作成されます（17-7 ページの「CREATE USER」を参照）。この文を実行すると、複数のトランザクションで複数の SQL 文を発行しなくても、スキーマに表およびビューが移入され、これらのオブジェクトに対する権限が付与されます。

前提条件

CREATE SCHEMA 文には、CREATE TABLE 文、CREATE VIEW 文および GRANT 文を含めることができます。CREATE SCHEMA 文を発行する場合は、挿入した文を発行するための権限が必要です。

構文

***create_schema*::=**



セマンティクス

schema

スキーマの名前を指定します。スキーマ名は、Oracle Database ユーザー名と一致している必要があります。

create_table_statement

この CREATE SCHEMA 文の一部として発行する CREATE TABLE 文を指定します。この文の終わりには、セミコロン（またはその他の終了文字）を付けないでください。

参照： 「CREATE TABLE」（16-6 ページ）

create_view_statement

この CREATE SCHEMA 文の一部として発行する CREATE VIEW 文を指定します。この文の終わりには、セミコロン（またはその他の終了文字）を付けないでください。

参照： 「CREATE VIEW」（17-13 ページ）

grant_statement

この CREATE SCHEMA 文の一部として発行する GRANT 文を指定します。この文の終わりには、セミコロン（またはその他の終了文字）を付けないでください。この句を使用すると、所有するオブジェクトに対するオブジェクト権限を、他のユーザーに付与できます。また、WITH ADMIN OPTION 付きでシステム権限を付与されている場合、それらの権限を他のユーザーに付与できます。

参照： 「GRANT」（18-31 ページ）

CREATE SCHEMA 文は、Oracle Database でサポートされている完全な構文ではなく、標準 SQL で定義されている構文のみをサポートします。

CREATE TABLE、CREATE VIEW および GRANT の各文を指定する順序は重要ではありません。CREATE SCHEMA 文の中の文では、既存のオブジェクトまたは同じ CREATE SCHEMA 文の他の文で作成したオブジェクトを参照できます。

スキーマに対する権限付与の制限事項： *parallel_clause* 構文は、CREATE SCHEMA の CREATE TABLE 文で使用できますが、オブジェクトの作成時に並列度は使用されません。

参照： 詳細は、16-54 ページの「CREATE TABLE」の「parallel_clause」を参照してください。

例

スキーマの作成例： 次の文は、サンプルの注文入力ユーザー oe 用の oe という名前のスキーマ、表 new_product、ビュー new_product_view を作成し、サンプルの人事情報のユーザー hr に new_product_view に対する SELECT オブジェクト権限を付与します。

```
CREATE SCHEMA AUTHORIZATION oe
   CREATE TABLE new_product
      (color VARCHAR2(10)  PRIMARY KEY, quantity NUMBER)
   CREATE VIEW new_product_view
      AS SELECT color, quantity FROM new_product WHERE color = 'RED'
   GRANT select ON new_product_view TO hr;
```

CREATE SEQUENCE

用途

CREATE SEQUENCE 文を使用すると、**順序**を作成できます。順序とは、データベース・オブジェクトの 1 つで、これを使用して複数のユーザーが一意の整数を生成することができます。順序を使用した場合、主キー値が自動的に生成されます。

順序番号が生成されると、順序はトランザクションのコミットやロールバックとは無関係に増加していきます。2 人のユーザーが、同時に同一の順序を増加させると、ユーザーがそれぞれ順序番号を生成しているため、取得する順序番号間に違いが発生することもあります。他のユーザーが生成した順序番号は取得できません。あるユーザーが順序値を生成すると、他のユーザーがその順序を増加させたかどうかに関係なく、順序を生成したユーザーは引き続きその値にアクセスできます。

順序番号は表から独立して生成されるため、1 つ以上の表に対して同一の順序を使用することができます。生成された順序番号が、最終的にロールバックされるトランザクションで使用されたため、個々の順序番号が連続していないように見える場合があります。また、他のユーザーが同一順序を使用していることを個々のユーザーが認識しない場合もあります。

順序が作成されると、SQL 文の中で CURRVAL 疑似列を使用してその値にアクセスできます（この場合、その順序の現在の値が戻ります）。また、NEXTVAL 疑似列を使用してもアクセスできます（この場合は、順序が増加され、新しい値が戻ります）。

参照：

- CURRVAL および NEXTVAL の使用方法については、第 3 章「疑似列」を参照してください。
- 順序の使用については、3-4 ページの「順序値の使用方法」を参照してください。
- 順序の変更または削除の詳細は、11-39 ページの「ALTER SEQUENCE」または 18-2 ページの「DROP SEQUENCE」を参照してください。

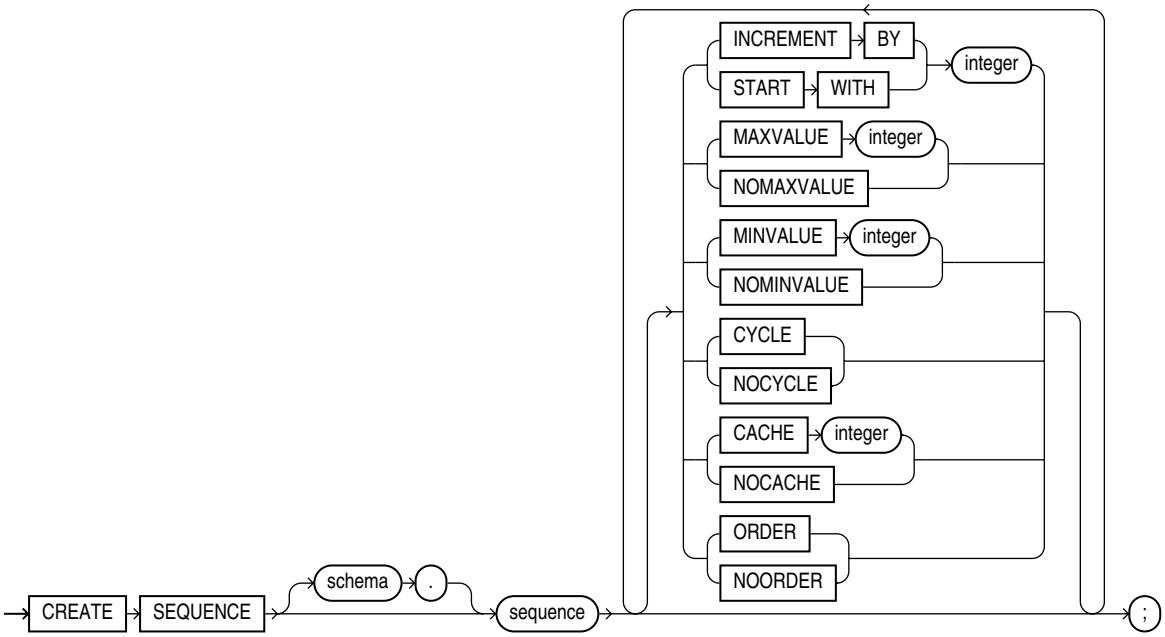
前提条件

自分のスキーマ内に順序を作成する場合は、CREATE SEQUENCE システム権限が必要です。

他のユーザーのスキーマ内に順序を作成する場合は、CREATE ANY SEQUENCE システム権限が必要です。

構文

***create_sequence*::=**



セマンティクス

schema

順序を含めるスキーマを指定します。*schema* を省略した場合、自分のスキーマ内に順序が作成されます。

sequence

作成する順序の名前を指定します。

次の句のうちどれも指定しない場合は、1 から始まる昇順の順序が作成され、上限なしで 1 ずつ増加していきます。INCREMENT BY に -1 のみを指定した場合は、初期値を -1 として、下限なしで 1 つずつ減少していきます。

- 昇順で、無制限に増加する順序を作成する場合は、MAXVALUE パラメータを省略するか、または NOMAXVALUE を指定します。降順の場合は、MINVALUE パラメータを省略するか、または NOMINVALUE を指定します。
- 昇順で、事前に定義した制限で停止する順序を作成する場合は、MAXVALUE パラメータに値を指定します。降順の場合は、MINVALUE パラメータの値を指定します。NOCYCLE も指定します。順序が制限に達したときに順序番号をさらに生成しようとした場合、エラーが発生します。
- 事前に定義した制限に達した後で初期値に戻る順序を作成する場合は、MAXVALUE パラメータと MINVALUE パラメータの両方に値を指定します。CYCLE も指定します。MINVALUE を指定しない場合、デフォルトで NOMINVALUE（値 1）が設定されます。

INCREMENT BY　順序の番号間の増分間隔を指定します。この値は、0（ゼロ）以外の正の整数または負の整数になります。この値には、28 桁以内の値を指定できます。この値の絶対値は、MAXVALUE と MINVALUE の差未満である必要があります。この値が負の場合、順序は降順になります。この値が正の場合、順序は昇順になります。この句を省略した場合、デフォルトで増分間隔は 1 に設定されます。

START WITH 生成する順序番号の初期値を指定します。この句を指定した場合、順序の最小値より大きい値を初期値として昇順を開始することも、最大値よりも小さい値を初期値として降順を開始することもできます。昇順の場合、デフォルト値は順序の最小値になります。降順の場合、デフォルト値は順序の最大値になります。28 桁以内の整数値を指定できます。

注意： この値は、必ずしも、順序の最大値または最小値に達した後に、昇順で循環する順序が戻るときの値ではありません。

MAXVALUE 順序の最大値を指定します。28 桁以内の整数値を指定できます。MAXVALUE 値は、START WITH 以上で、かつ MINVALUE を超える値である必要があります。

NOMAXVALUE NOMAXVALUE を指定すると、順序の最大値を、昇順の場合は 10^{27}、降順の場合は -1 に指定できます。これはデフォルトです。

MINVALUE 順序の最小値を指定します。28 桁以内の整数値を指定できます。MINVALUE 値は、START WITH 以下で、かつ MAXVALUE 未満である必要があります。

NOMINVALUE NOMINVALUE を指定すると、順序の最小値を、昇順の場合は 1、降順の場合は -10^{26} に指定できます。これはデフォルトです。

CYCLE CYCLE を指定すると、順序が最大値または最小値に達しても、引き続き値を生成できます。つまり、昇順の場合は、最大値に達すると最小値が生成されます。降順の場合は、最小値に達すると最大値が生成されます。

NOCYCLE NOCYCLE を指定すると、順序が最大値または最小値に達した場合は、それ以上値を生成しないように指定できます。これはデフォルトです。

CACHE より高速に順序番号にアクセスできるように、メモリー上に事前に割り当て、保持しておく順序番号の数を指定します。28 桁以内の整数値を指定できます。このパラメータの最小値は 2 です。循環する順序の場合、この値は、そのサイクル内で生成される値の数未満である必要があります。指定したサイクル内で生成される順序番号の数を超える値はキャッシュできません。したがって、CACHE に指定できる値の最大値は、次の式で求められる値未満である必要があります。

```
(CEIL (MAXVALUE - MINVALUE)) / ABS (INCREMENT)
```

システム障害が発生すると、キャッシュされた順序の値のうち、コミットされた DML 文で使用されていなかったものはすべて失われます。したがって、失われる可能性がある値の数は、CACHE パラメータの値と等しくなります。

注意： Oracle Real Application Clusters 環境で順序を使用する場合は、パフォーマンスを向上するために、CACHE の設定を使用することをお薦めします。

NOCACHE NOCACHE を指定すると、順序の値を事前に割り当てないように指定できます。CACHE および NOCACHE の両方を省略した場合、デフォルトで 20 の順序番号がキャッシュされます。

ORDER ORDER を指定すると、要求どおりの順序で順序番号を生成することを保証できます。順序番号をタイムスタンプとして使用する場合に、この句は有効です。通常、主キー生成用の順序については、順序どおりに生成するかどうかの保証は重要ではありません。

Oracle Real Application Clusters を使用する場合、ORDER は順序どおりの生成を確保する場合にのみ必要です。排他モードの場合、順序番号は必ず順序どおりに生成されます。

NOORDER NOORDER を指定すると、要求どおりの順で順序番号を生成することは保証されません。これはデフォルトです。

例

順序の作成例 : 次の文は、サンプル・スキーマ oe に順序 customers_seq を作成します。この順序は、customers 表に行が追加されるときに、顧客 ID 番号として使用できます。

```
CREATE SEQUENCE customers_seq
 START WITH     1000
 INCREMENT BY   1
 NOCACHE
 NOCYCLE;
```

最初に customers_seq.nextval を参照したときは、1000 が戻されます。次に参照したときは、1001 が戻されます。同様に、この後の各参照で、前回参照された値より 1 大きい値が戻されます。

CREATE SPFILE

用途

CREATE SPFILE を使用すると、従来のプレーンテキストの初期化パラメータ・ファイルまたは現行のシステム全体の設定から、サーバー・パラメータ・ファイルを作成できます。サーバー・パラメータ・ファイルは、サーバーのみに存在し、データベースを起動するためにクライアントからコールされるバイナリ・ファイルです。

サーバー・パラメータ・ファイルを指定すると、個々のパラメータを永続的に変更できます。サーバー・パラメータ・ファイルを使用する場合、新しいパラメータ値を永続化する ALTER SYSTEM SET *parameter* 文を指定できます。新しい値は、現行のインスタンスのみでなく、その後で起動するすべてのインスタンスにおいて適用されます。従来のプレーンテキストのパラメータ・ファイルでは、パラメータ値を永続的に変更できません。

サーバー・パラメータ・ファイルは、サーバー上に存在するため、Oracle Database による自動データベースのチューニングおよび Recovery Manager によるバックアップが可能です。

データベースの起動時にサーバー・パラメータ・ファイルを使用するには、CREATE SPFILE 文を使用してサーバー・パラメータ・ファイルを作成する必要があります。

Oracle Real Application Clusters 環境のすべてのインスタンスは、同一のサーバー・パラメータ・ファイルを使用する必要があります。ただし、個々のインスタンスで 1 つのファイル内の同じパラメータで異なる設定が可能な場合もあります。インスタンス固有のパラメータ定義は、*SID.parameter = value* で指定します。*SID* にはインスタンス識別子を指定します。

サーバー・パラメータ・ファイルを使用したデータベースの起動方法は、作成したサーバー・パラメータ・ファイルをデフォルトで作成したか、または非デフォルトで作成したかによって異なります。サーバー・パラメータ・ファイルの使用方法については、15-70 ページの「サーバー・パラメータ・ファイルの作成例 :」を参照してください。

参照 :

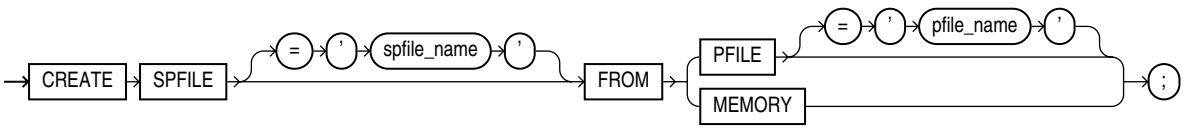
- バイナリのサーバー・パラメータ・ファイルから正規のテキストのパラメータ・ファイルを作成する場合の詳細は、15-43 ページの「CREATE PFILE」を参照してください。
- 従来のプレーンテキストの初期化パラメータ・ファイルおよびサーバー・パラメータ・ファイルの詳細は、『Oracle Database 管理者ガイド』を参照してください。
- Oracle Real Application Clusters 環境でのサーバー・パラメータ・ファイルの使用については、『Oracle Real Application Clusters 管理およびデプロイメント・ガイド』を参照してください。

前提条件

この文を実行するには、SYSDBA システム権限または SYSOPER システム権限が必要です。この文は、インスタンスの起動前と起動後のいずれかで実行できます。ただし、*spfile_name* を使用して、インスタンスがすでに起動済の場合は、この文で同じ *spfile_name* を指定できません。

構文

***create_spfile*::=**



セマンティクス

spfile_name

この句を指定すると、作成するサーバー・パラメータ・ファイルの名前を指定できます。

- *spfile_name* を指定しない場合、プラットフォーム固有のデフォルトのサーバー・パラメータ・ファイル名が使用されます。*spfile_name* がサーバーにすでに存在する場合、そのファイルは上書きされます。デフォルトのサーバー・パラメータ・ファイルを使用する場合、ファイル名を参照せずにデータベースが起動されます。
- *spfile_name* を指定する場合、デフォルト以外のサーバー・パラメータ・ファイルを作成します。この場合、データベースを起動するときに、サーバー・パラメータ・ファイルを指す単一行の以前のパラメータ・ファイルを作成し、STARTUP コマンドで単一行のファイルを指定する必要があります。

spfile_name には、パス接頭辞を含めることができます。パス接頭辞を指定しない場合は、データベースによってデフォルトの格納場所（プラットフォームによって異なる）のパス接頭辞が追加されます。

参照 :

- デフォルトおよびデフォルト以外のサーバー・パラメータ・ファイルを使用したデータベースの起動の詳細は、15-70 ページの「サーバー・パラメータ・ファイルの作成例 :」を参照してください。
- デフォルトのパラメータ・ファイル名については、オペレーティング・システム固有のドキュメントを参照してください。

pfile_name

サーバー・パラメータ・ファイルを作成する元になる従来のプレーンテキストの初期化パラメータ・ファイル名を指定します。従来のパラメータ・ファイルは、サーバー上にある必要があります。

- *pfile_name* を指定して、従来のパラメータ・ファイルがオペレーティング・システムのパラメータ・ファイルのデフォルト・ディレクトリに存在しない場合は、フルパス名を指定する必要があります。
- *pfile_name* を指定しない場合、オペレーティング・システムのパラメータ・ファイルのデフォルト・ディレクトリからデフォルトのパラメータ・ファイル名が検索され、使用されます。そのディレクトリにファイルが存在しない場合、エラーが戻されます。

注意 : Oracle Real Application Clusters 環境では、この文でパラメータ・ファイル名を指定してサーバー・パラメータ・ファイルを作成する前に、まず、すべてのインスタンスのパラメータ・ファイルを 1 つのファイルに結合する必要があります。この手順の実行の詳細は、『Oracle Real Application Clusters 管理およびデプロイメント・ガイド』を参照してください。

MEMORY

MEMORY を指定すると、現行のシステム全体のパラメータ設定を使用して spfile を作成できます。RAC 環境では、作成されたファイルには各インスタンスからのパラメータ設定が含まれます。

例

サーバー・パラメータ・ファイルの作成例： 次の例は、クライアントの初期化パラメータ・ファイル t_init1.ora からデフォルトのサーバー・パラメータ・ファイルを作成します。

```
CREATE SPFILE
   FROM PFILE = '$ORACLE_HOME/work/t_init1.ora';
```

> **注意：** 通常、オペレーティング・システムのパラメータ・ファイルには、フルパスのファイル名を指定する必要があります。

デフォルトのサーバー・パラメータ・ファイルを作成する場合、その後のデータベースの起動は、次のように PFILE パラメータなしで SQL*Plus のコマンド STARTUP を実行し、サーバー・パラメータ・ファイルを使用します。

```
STARTUP
```

次の例は、クライアントの初期化パラメータ・ファイル t_init1.ora からデフォルト以外のサーバー・パラメータ・ファイル s_params.ora を作成します。

```
CREATE SPFILE = 's_params.ora'
   FROM PFILE = '$ORACLE_HOME/work/t_init1.ora';
```

デフォルト以外のサーバー・パラメータ・ファイルを作成する場合、次の単一行を含む以前のパラメータ・ファイルを最初に作成し、その後でデータベースを起動します。

```
spfile = 's_params.ora'
```

このパラメータ・ファイル名は、ご使用のオペレーティング・システムのネーミング規則に従う必要があります。その後、STARTUP コマンドで単一行のパラメータ・ファイルを使用します。次の例では、単一行のパラメータ・ファイルの名前が new_param.ora であるとした場合のデータベースの起動方法を示します。

```
STARTUP PFILE=new_param.ora
```

16

SQL 文 : CREATE SYNONYM ～ CREATE TRIGGER

この章では、次の SQL 文について説明します。

- CREATE SYNONYM
- CREATE TABLE
- CREATE TABLESPACE
- CREATE TRIGGER

CREATE SYNONYM

用途

CREATE SYNONYM 文を使用すると、**シノニム**を作成できます。シノニムとは、表、ビュー、順序、演算子、プロシージャ、ストアド・ファンクション、パッケージ、マテリアライズド・ビュー、Java クラス・スキーマ・オブジェクト、ユーザー定義オブジェクト型および別のシノニムに付ける別名です。シノニムによってシノニムのターゲット・オブジェクトへの依存関係が設定され、ターゲット・オブジェクトが変更または削除されるとシノニムも無効になります。

シノニムによって、データの独立性および位置の透過性を実現できます。シノニムを使用した場合、どのユーザーが表やビューを所有しているか、どのデータベースに表やビューが格納されているかに関係なく、アプリケーションは機能します。ただし、シノニムはデータベース・オブジェクトに対する権限にかわるものではありません。シノニムを使用するユーザーに対して、適切な権限をあらかじめ付与しておく必要があります。

シノニムを参照できる DML 文は、SELECT、INSERT、UPDATE、DELETE、FLASHBACK TABLE、EXPLAIN PLAN および LOCK TABLE です。

シノニムを参照できる DDL 文は、AUDIT、NOAUDIT、GRANT、REVOKE および COMMENT です。

> **参照：** シノニムの概要については、『Oracle Database 概要』を参照してください。

前提条件

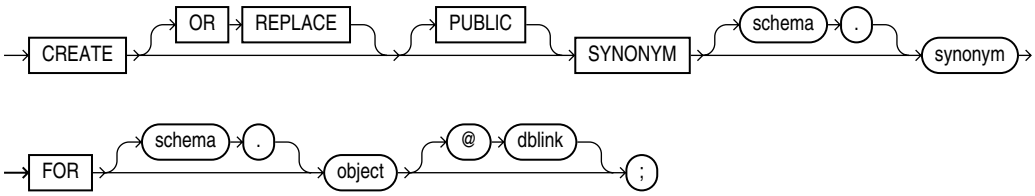
自分のスキーマ内にプライベート・シノニムを作成する場合は、CREATE SYNONYM システム権限が必要です。

他のユーザーのスキーマ内にプライベート・シノニムを作成する場合は、CREATE ANY SYNONYM システム権限が必要です。

パブリック・シノニムを作成する場合は、CREATE PUBLIC SYNONYM システム権限が必要です。

構文

***create_synonym*::=**



セマンティクス

OR REPLACE

OR REPLACE を指定すると、既存のシノニムを再作成できます。この句を使用すると、既存のシノニムの定義をはじめに削除しなくても、その定義を変更できます。

シノニムの置換の制限事項： OR REPLACE 句は、依存表または依存する有効なユーザー定義オブジェクト型を持つ型シノニムに対して使用できません。

PUBLIC

PUBLIC を指定すると、パブリック・シノニムを作成できます。パブリック・シノニムには、すべてのユーザーがアクセスできます。ただし、シノニムを使用するには、基礎となるオブジェクトに対する適切な権限が必要です。

オブジェクトの先頭にスキーマ名が指定されておらず、オブジェクトの後にデータベース・リンクが指定されていない場合のみ、オブジェクトへの参照を変換するときに、パブリック・シノニムが使用されます。

この句を指定しない場合、シノニムはプライベートです。プライベート・シノニム名は、スキーマ内で一意である必要があります。プライベート・シノニムに所有者以外のユーザーがアクセスできるのは、基礎となるデータベース・オブジェクトに対する適切な権限がユーザーにあり、シノニム名とともにスキーマを指定する場合のみです。

パブリック・シノニムの注意事項： パブリック・シノニムには、次の注意事項があります。

- パブリック・シノニムを作成した後、依存表または依存する有効なユーザー定義オブジェクト型が存在する場合、依存オブジェクトと同じスキーマ内には、そのシノニムと同じ名前で別のデータベース・オブジェクトは作成できません。
- 既存のスキーマと同じ名前のパブリック・シノニムを作成しないでください。同じ名前のパブリック・シノニムを作成すると、その名前が使用されるすべての PL/SQL ユニットが無効になります。

schema

シノニムを含めるスキーマを指定します。*schema* を省略した場合、自分のスキーマ内にシノニムが作成されます。PUBLIC を指定した場合、スキーマは指定できません。

synonym

作成するシノニムの名前を指定します。

注意： シノニム名は、30 バイトを超える場合も作成および削除できます。ただし、Java 名でない場合は他の SQL コマンドで機能しません。30 バイトを超える名前は、データ・ディクショナリに格納するために不確定で短い文字列に変換されます。

参照： 16-4 ページの「CREATE SYNONYM の例 :」および 16-4 ページの「Oracle Database によるシノニムの変換例 :」を参照してください。

FOR 句

シノニムを作成するオブジェクトを指定します。シノニムを作成するスキーマ・オブジェクトには、次のものがあります。

- 表またはオブジェクト表
- ビューまたはオブジェクト・ビュー
- 順序
- ストアド・プロシージャ、ファンクションまたはパッケージ
- マテリアライズド・ビュー
- Java クラス・スキーマ・オブジェクト
- ユーザー定義オブジェクト型
- シノニム

スキーマ・オブジェクトは、現在存在している必要はなく、スキーマ・オブジェクトへのアクセス権限も必要ありません。

FOR 句の制限事項： スキーマ・オブジェクトは、パッケージに含めることはできません。

schema オブジェクトが含まれているスキーマを指定します。オブジェクトに *schema* を指定しなかった場合、そのスキーマ・オブジェクトは自分のスキーマ内にあるとみなされます。

リモート・データベース上のプロシージャやファンクションに対するシノニムを作成する場合、この CREATE 文で *schema* を指定する必要があります。または、オブジェクトが存在するデータベースにローカル・パブリック・シノニムを作成することもできます。ただし、その後は、プロシージャやファンクションの後続のコールすべてにデータベース・リンクを組み込む必要があります。

dblink データベース・リンクを完全に指定するか、またはデータベース・リンクの一部を指定すると、スキーマ・オブジェクトが格納されているリモート・データベース上のオブジェクトのシノニムを作成できます。*dblink* を指定して、*schema* を省略した場合、シノニムは、データベース・リンクで指定されたスキーマ内のオブジェクトを参照します。リモート・データベースのオブジェクトが含まれているスキーマを指定することをお薦めします。

dblink を省略した場合、オブジェクトがローカル・データベース上にあるものとみなされます。

データベース・リンクの制限事項： *dblink* は、Java クラス・シノニムに対して指定できません。

参照：

- データベース・リンクの参照方法の詳細は、2-104 ページの「リモート・データベース内のオブジェクトの参照」を参照してください。
- データベース・リンクの作成方法の詳細は、14-28 ページの「CREATE DATABASE LINK」を参照してください。

例

CREATE SYNONYM の例： 次の文は、スキーマ hr 内の表 locations に対してシノニム offices を定義します。

```
CREATE SYNONYM offices
   FOR hr.locations;
```

次の文は、remote データベース上のスキーマ hr 内の employees 表に対してパブリック・シノニムを作成します。

```
CREATE PUBLIC SYNONYM emp_table
   FOR hr.employees@remote.us.example.com;
```

別のスキーマ内に基礎となるオブジェクトが含まれている場合は、基礎となるオブジェクトと同じ名前をシノニムに指定することもできます。

Oracle Database によるシノニムの変換例： Oracle Database は、オブジェクトの参照を、パブリック・シノニム・レベルで変換する前に、スキーマ・レベルで変換しようとします。たとえば、スキーマ oe と sh の両方に customers という名前の表が存在するとします。次の例では、ユーザー SYSTEM が、oe.customers に対して customers という名前のパブリック・シノニムを作成します。

```
CREATE PUBLIC SYNONYM customers FOR oe.customers;
```

ユーザー sh が次の文を発行すると、sh.customers から行数が戻されます。

```
SELECT COUNT(*) FROM customers;
```

oe.customers から行数を取得するには、ユーザー sh は、customers の前にスキーマ名を指定する必要があります（ユーザー sh は、oe.customers に対する SELECT 権限も必要です）。

```
SELECT COUNT(*) FROM oe.customers;
```

ユーザー hr のスキーマに customers という名前のオブジェクトは存在しないが、hr が oe.customers に対する SELECT 権限を持つ場合、hr は、パブリック・シノニム customers を使用して、oe のスキーマ内の customers 表にアクセスできます。

```
SELECT COUNT(*) FROM customers;
```

CREATE TABLE

用途

CREATE TABLE 文を使用すると、次の型の表を作成できます。

- **リレーショナル表**。ユーザー・データを格納する基本構造です。
- **オブジェクト表**。列の定義にオブジェクト型を使用する表です。特定の型のオブジェクト・インスタンスを格納するように明示的に定義されます。

オブジェクト型を作成しておき、リレーショナル表の作成時に列の中でそのオブジェクト型を使用することもできます。

副問合せを指定しない場合、データを含まない表が作成されます。INSERT 文を使用した場合、表に行を追加できます。表を作成した後、ALTER TABLE 文で ADD 句を指定すると、追加する列、パーティションおよび整合性制約を定義できます。ALTER TABLE 文で MODIFY 句を指定すると、既存の列またはパーティションの定義を変更できます。

参照：

- オブジェクトの作成の詳細は、『Oracle Database 管理者ガイド』および 17-3 ページの「CREATE TYPE」を参照してください。
- 表の変更および削除の詳細は、12-2 ページの「ALTER TABLE」および 18-5 ページの「DROP TABLE」を参照してください。

前提条件

自分のスキーマ内にリレーショナル表を作成する場合は、CREATE TABLE システム権限が必要です。他のユーザーのスキーマ内に表を作成する場合は、CREATE ANY TABLE システム権限が必要です。また、表が含まれるスキーマの所有者は、表を格納するため表領域への割当て制限または UNLIMITED TABLESPACE システム権限が必要です。

これらの表権限に加え、オブジェクト表またはオブジェクト型の列が存在するリレーショナル表を作成する場合は、表の所有者に、表が参照するすべての型にアクセスするための EXECUTE オブジェクト権限が付与されているかまたは EXECUTE ANY TYPE システム権限が付与されている必要があります。これらの権限は、ロールを介して取得するのではなく、明示的に付与される必要があります。

さらに、表の所有者が表へのアクセス権限を他のユーザーに付与する場合、所有者には、参照する型に対する WITH GRANT OPTION 付きの EXECUTE オブジェクト権限または WITH ADMIN OPTION 付きの EXECUTE ANY TYPE システム権限が必要です。これらの権限を持っていない場合、表の所有者は、表へのアクセス権限を他のユーザーに付与できません。

一意キー制約または主キー制約を有効にする場合は、表に索引を作成するための権限が必要です。Oracle Database では、表を含むスキーマにおいて、一意キーまたは主キーの列に索引を作成するため、この権限が必要になります。

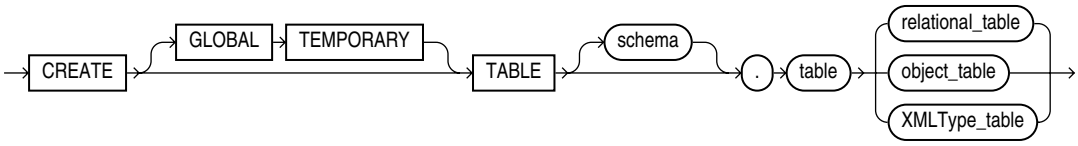
外部表を作成する場合は、適切なオペレーティング・システム・ディレクトリに対する、オペレーティング・システムの読取り権限および書込み権限が必要です。外部データが存在するオペレーティング・システム・ディレクトリに対応するデータベース・ディレクトリ・オブジェクトに対する READ オブジェクト権限が必要です。また、*opaque_format_spec* でログ・ファイルまたは不良ファイルを指定する場合、または AS *subquery* 句を指定してデータベース表から外部表にデータをアンロードする場合、ファイルが格納されるデータベース・ディレクトリに対する WRITE オブジェクト権限が必要です。

参照：

- 「CREATE INDEX」（14-50 ページ）
- 型を使用する表の作成に必要な権限については、『Oracle Database 管理者ガイド』を参照してください。

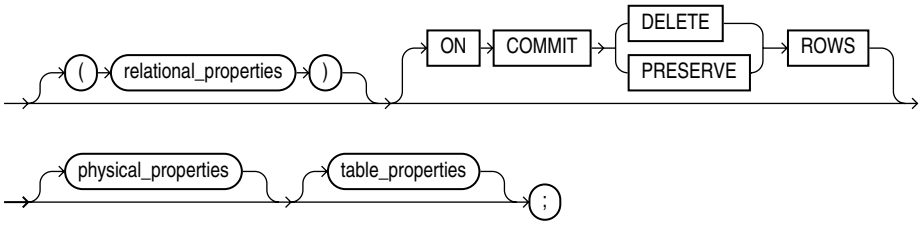
構文

***create_table*::=**



（16-7 ページの relational_table::=、16-7 ページの object_table::=、16-8 ページの XMLType_table::= を参照）

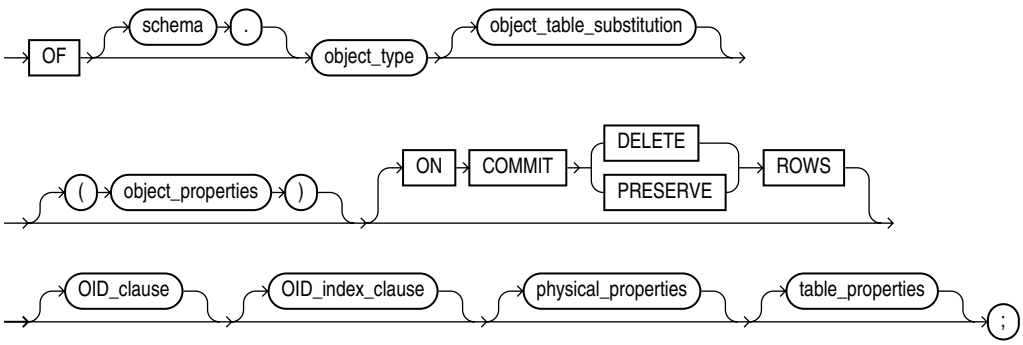
***relational_table*::=**



注意： 表名の後の各句は、任意のリレーショナル表に対して任意で指定します。ただし、すべての表に対して、*relational_properties* 句を使用して列名およびデータ型を指定するか、または *table_properties* 句を使用して AS *subquery* 句を指定する必要があります。

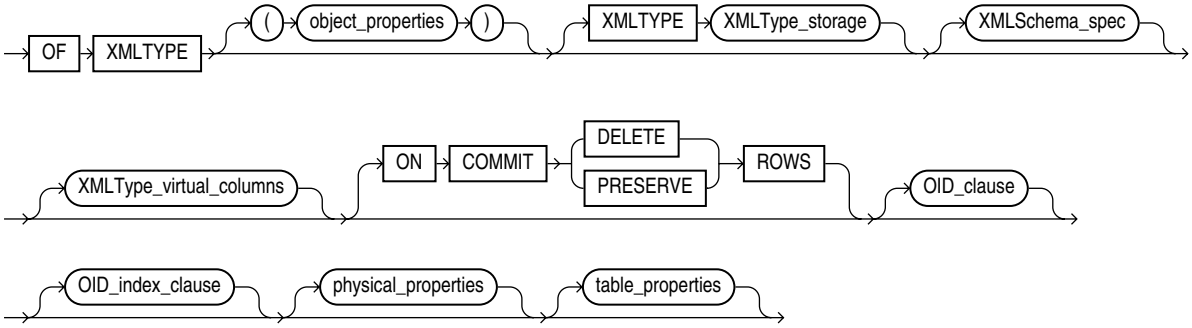
（16-8 ページの relational_properties::=、16-9 ページの physical_properties::=、16-10 ページの table_properties::= を参照）

object_table::=



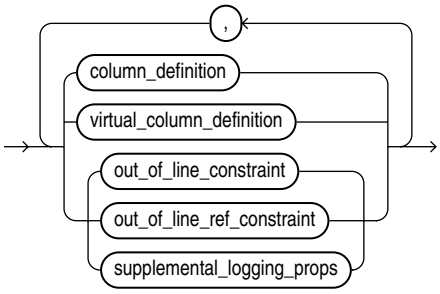
（16-9 ページの object_table_substitution::=、16-9 ページの object_properties::=、16-9 ページの oid_clause::=、16-9 ページの oid_index_clause::=、16-9 ページの physical_properties::=、16-10 ページの table_properties::= を参照）

XMLType_table::=



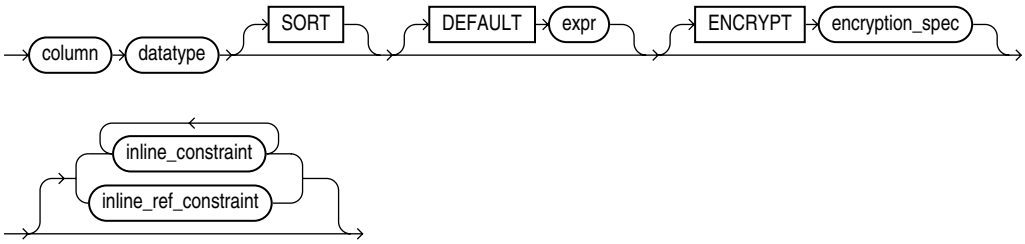
（16-14 ページの XMLType_storage::=、16-15 ページの XMLSchema_spec::=、16-15 ページの XMLType_virtual_columns::=、16-9 ページの oid_clause::=、16-9 ページの oid_index_clause::=、16-9 ページの physical_properties::=、16-10 ページの table_properties::= を参照）

relational_properties::=



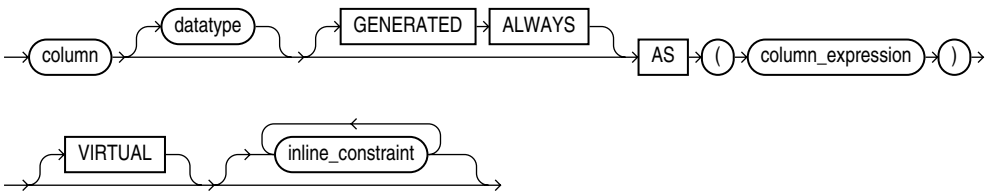
（16-8 ページの column_definition::=、16-8 ページの virtual_column_definition::=、8-5 ページの constraint::=、16-16 ページの supplemental_logging_props::= を参照）

column_definition::=



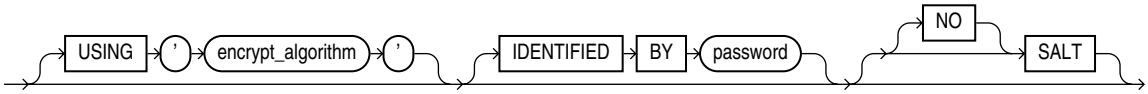
（16-9 ページの encryption_spec::=、8-5 ページの constraint::= を参照）

virtual_column_definition::=

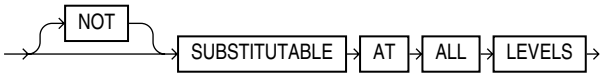


（8-5 ページの constraint::= を参照）

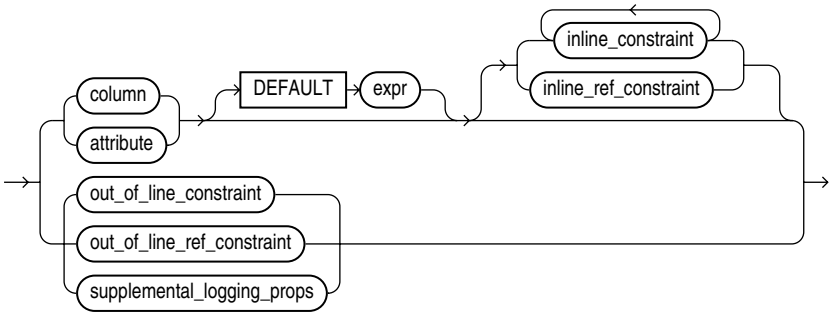
***encryption_spec*::=**



object_table_substitution::=

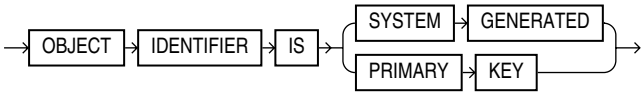


object_properties::=

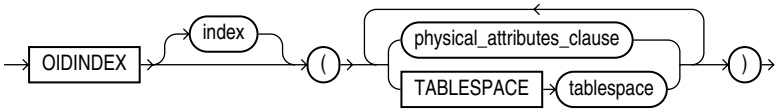


（8-5 ページの constraint::=、16-16 ページの supplemental_logging_props::= を参照）

oid_clause::=

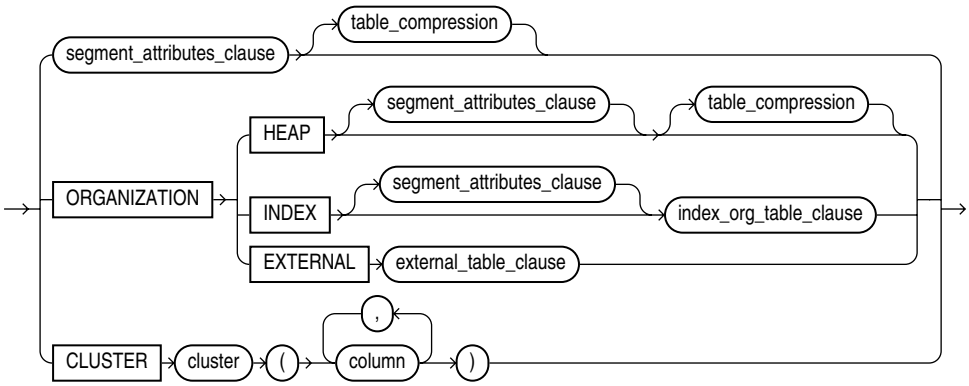


oid_index_clause::=



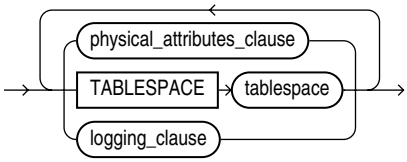
（16-10 ページの physical_attributes_clause::= を参照）

physical_properties::=



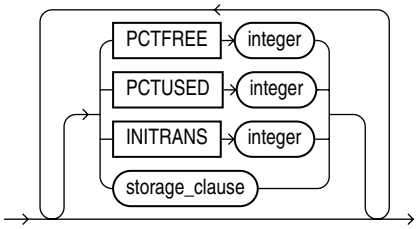
（16-10 ページの segment_attributes_clause::=、16-10 ページの table_compression::=、16-15 ページの index_org_table_clause::=、16-16 ページの external_table_clause::= を参照）

segment_attributes_clause::=



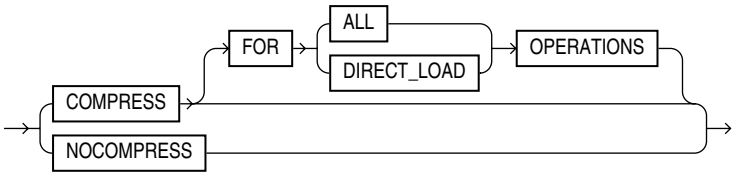
（16-10 ページの physical_attributes_clause::=、16-14 ページの logging_clause::= を参照）

physical_attributes_clause::=

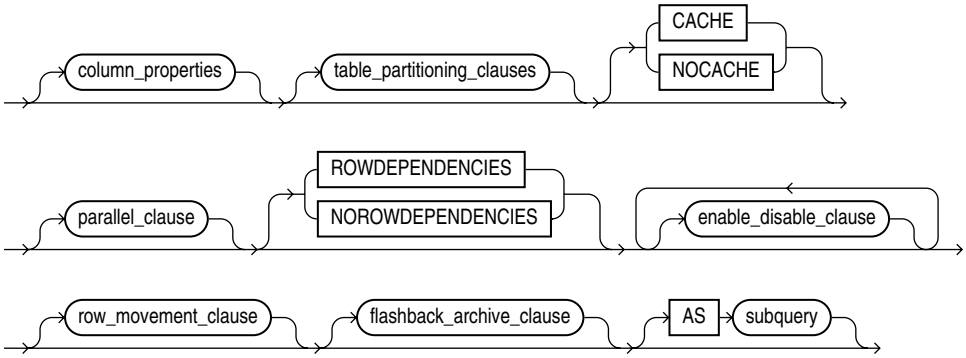


（8-44 ページの storage_clause::= を参照）

table_compression::=

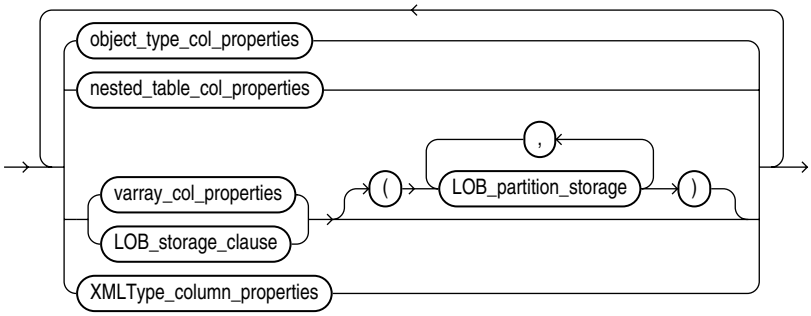


table_properties::=



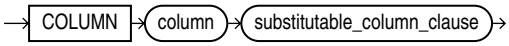
（16-11 ページの column_properties::=、16-17 ページの table_partitioning_clauses::=、
16-23 ページの parallel_clause::=、16-23 ページの enable_disable_clause::=、
16-15 ページの row_movement_clause::=、16-15 ページの flashback_archive_clause::=、
19-5 ページの subquery::= を参照）

column_properties::=

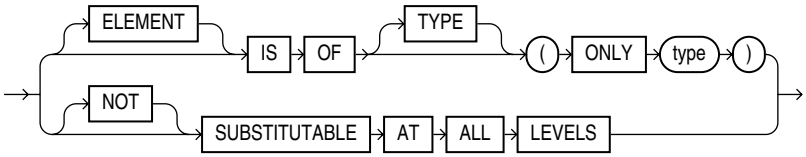


（16-11 ページの object_type_col_properties::=、16-11 ページの nested_table_col_properties::=、16-12 ページの varray_col_properties::=、16-12 ページの LOB_storage_clause::=、16-14 ページの LOB_partition_storage::=、16-14 ページの XMLType_column_properties::= を参照）

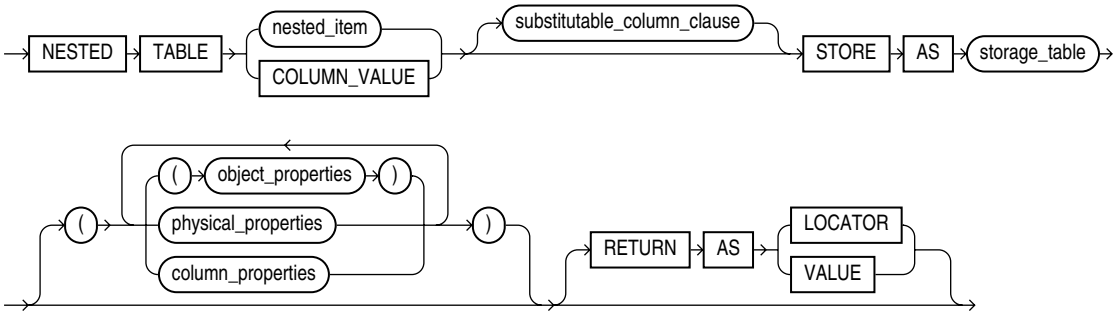
object_type_col_properties::=



***substitutable_column_clause*::=**

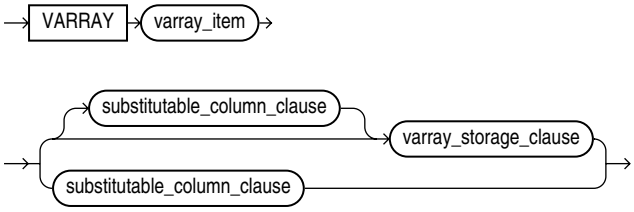


nested_table_col_properties::=



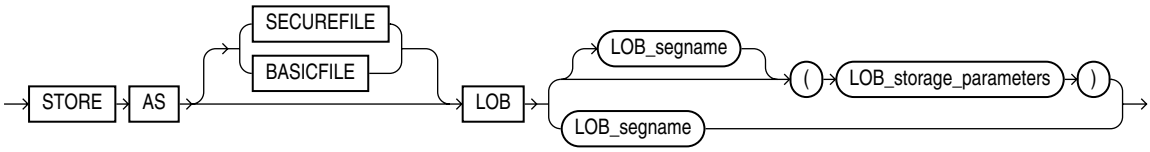
（16-11 ページの substitutable_column_clause::=、16-9 ページの object_properties::=、16-9 ページの physical_properties::=、16-11 ページの column_properties::= を参照）

varray_col_properties::=



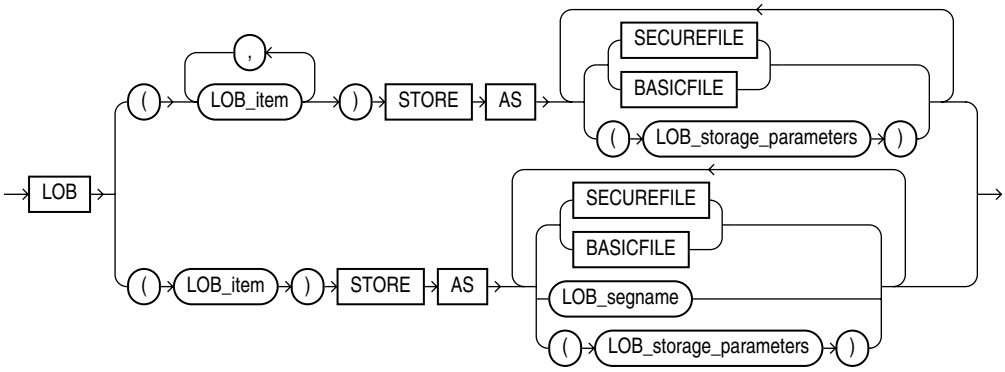
（16-11 ページの substitutable_column_clause::=、16-12 ページの varray_storage_clause::= を参照）

***varray_storage_clause*::=**



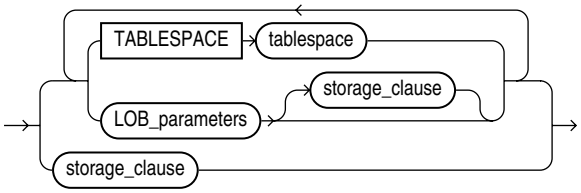
（16-13 ページの LOB_parameters::= を参照）

LOB_storage_clause::=



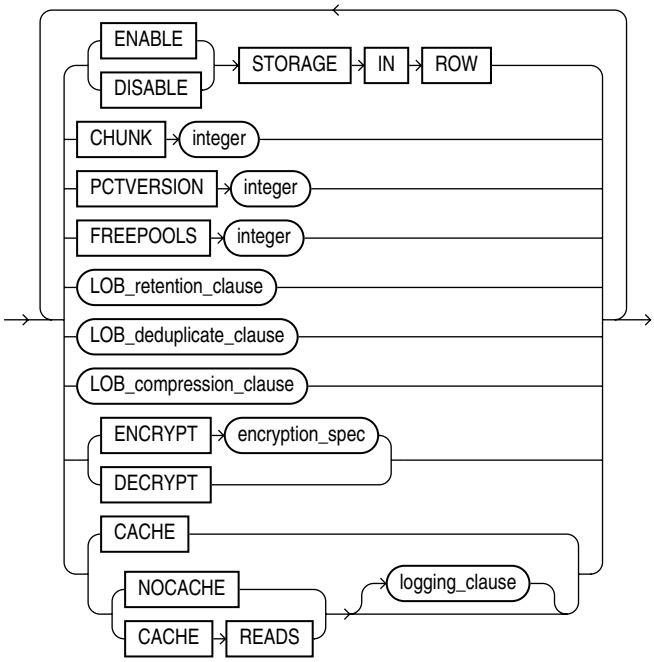
（16-12 ページの LOB_storage_parameters::= を参照）

***LOB_storage_parameters*::=**



（16-13 ページの LOB_parameters::=、8-44 ページの storage_clause::= を参照）

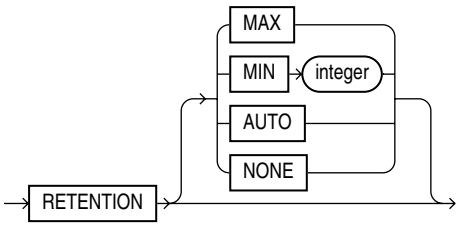
LOB_parameters::=



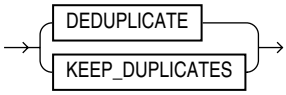
（16-13 ページの LOB_deduplicate_clause::=、16-13 ページの LOB_compression_clause::=、16-9 ページの encryption_spec::=、8-34 ページの logging_clause::= を参照）

> **注意：** LOB 記憶域に SecureFiles を使用する場合、いくつかの LOB パラメータは不要になります。詳細は、16-38 ページの「LOB_storage_parameters」を参照してください。

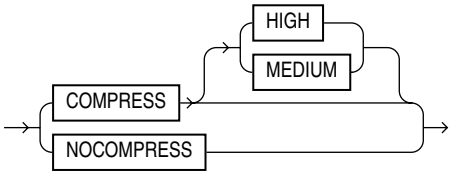
LOB_retention_clause::=



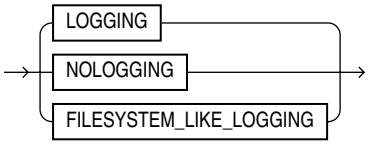
LOB_deduplicate_clause::=



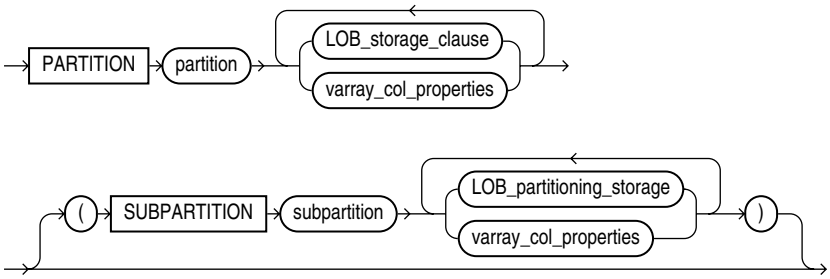
LOB_compression_clause::=



***logging_clause*::=**

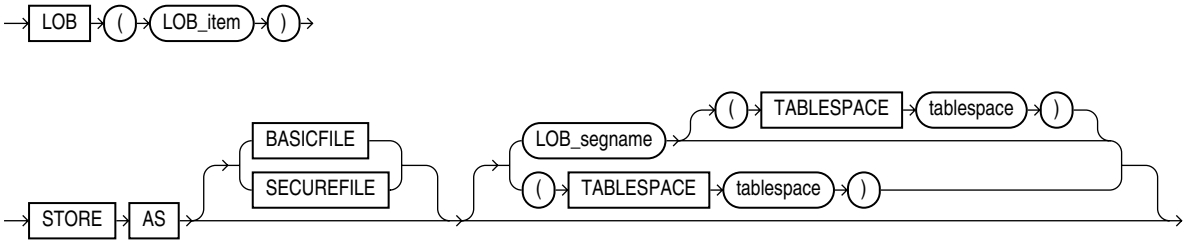


LOB_partition_storage::=

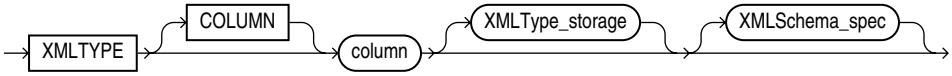


（16-12 ページの LOB_storage_clause::=、16-12 ページの varray_col_properties::=、16-14 ページの LOB_partitioning_storage::= を参照）

***LOB_partitioning_storage*::=**

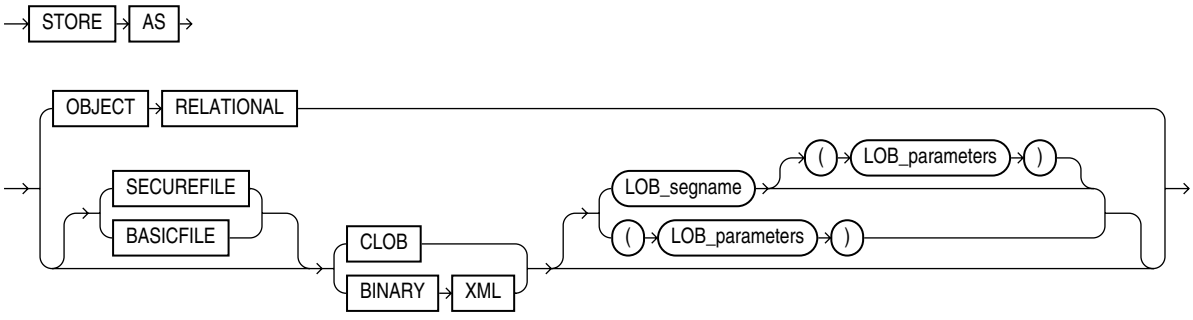


XMLType_column_properties::=



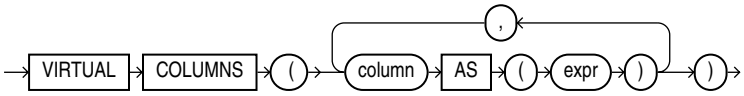
（16-14 ページの XMLType_storage::=、16-15 ページの XMLSchema_spec::= を参照）

***XMLType_storage*::=**

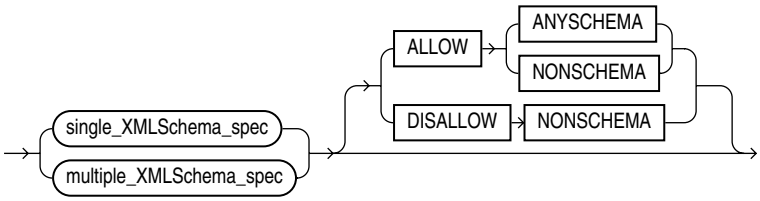


（16-13 ページの LOB_parameters::= を参照）

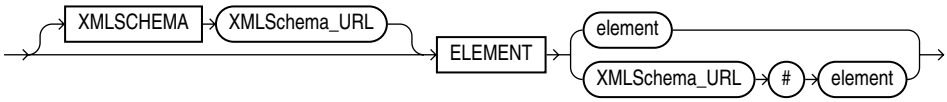
XMLType_virtual_columns::=



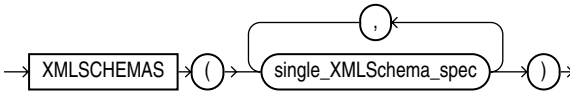
***XMLSchema_spec*::=**



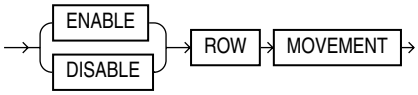
***single_XMLSchema_spec*::=**



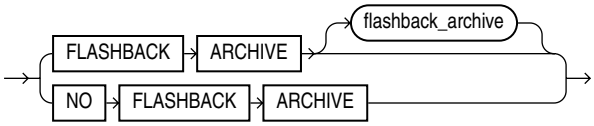
***multiple_XMLSchema_spec*::=**



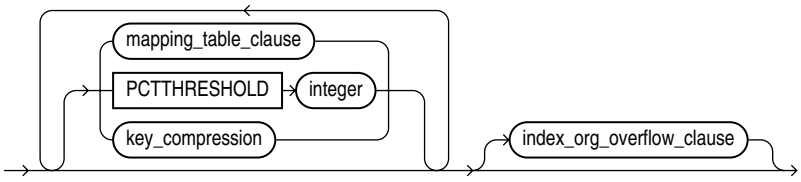
row_movement_clause::=



flashback_archive_clause::=

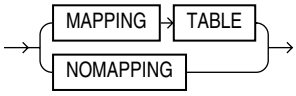


index_org_table_clause::=

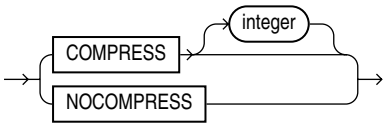


（16-16 ページの mapping_table_clauses::=、16-16 ページの key_compression::=、16-16 ページの index_org_overflow_clause::= を参照）

mapping_table_clauses::=



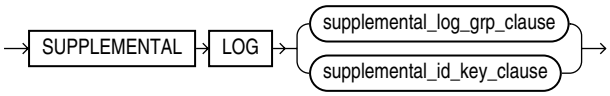
key_compression::=



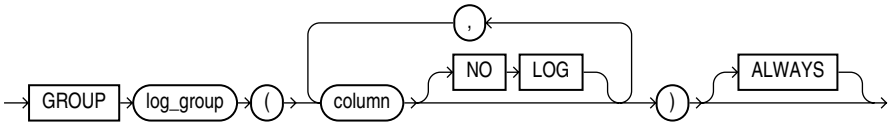
index_org_overflow_clause::=

(16-10 ページの segment_attributes_clause::= を参照)

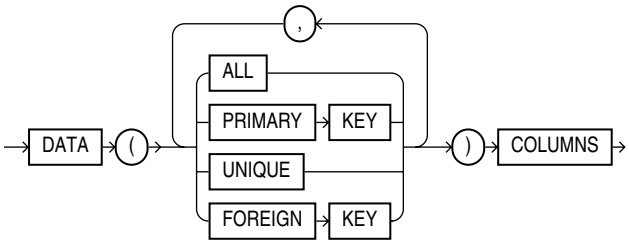
supplemental_logging_props::=



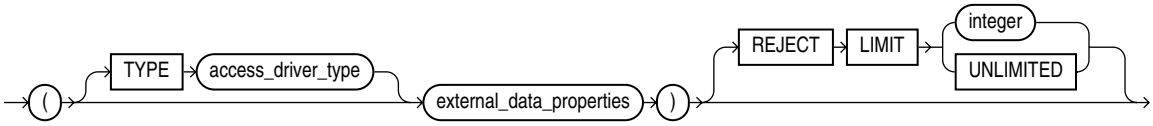
***supplemental_log_grp_clause*::=**



***supplemental_id_key_clause*::=**

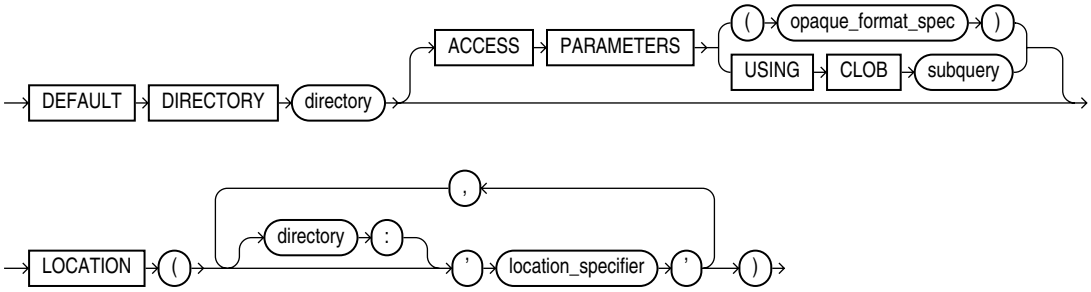


external_table_clause::=



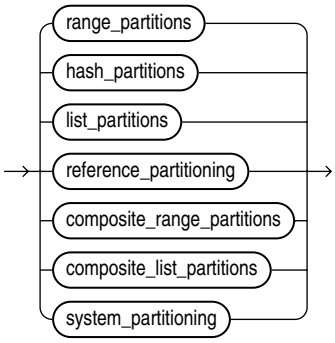
(16-17 ページの external_data_properties::= を参照)

***external_data_properties*::=**



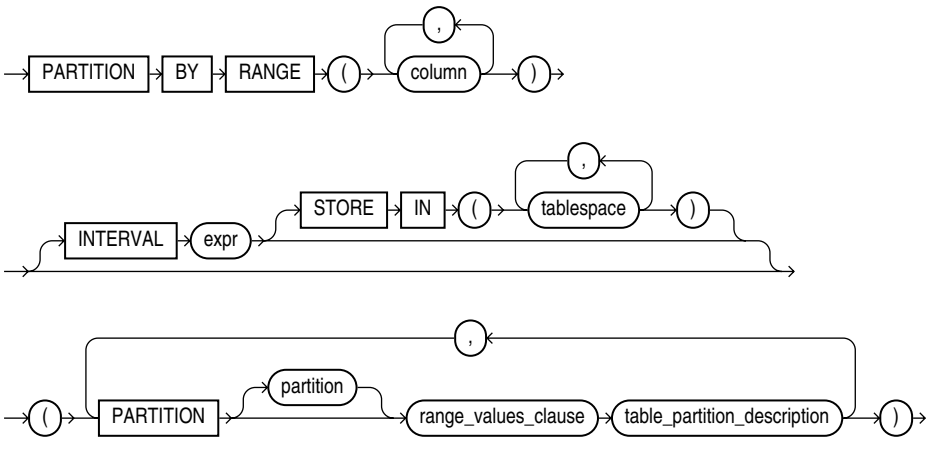
（*opaque_format_spec*: この句は、ORACLE_LOADER および ORACLE_DATAPUMP アクセス・ドライバのすべてのアクセス・パラメータを指定します。これらのパラメータの詳細は、『Oracle Database ユーティリティ』を参照してください。）

table_partitioning_clauses::=



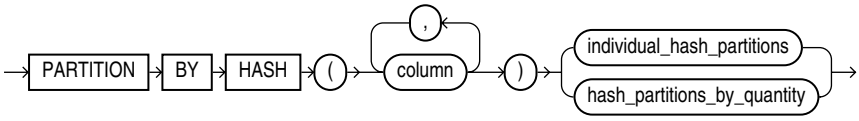
（16-17 ページの range_partitions::=、16-18 ページの hash_partitions::=、16-18 ページの list_partitions::=、16-18 ページの reference_partitioning::=、16-19 ページの composite_range_partitions::=、16-19 ページの composite_list_partitions::= および 16-19 ページの system_partitioning::= を参照）

range_partitions::=



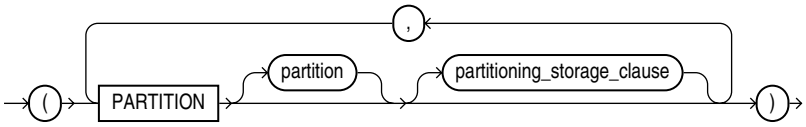
（16-22 ページの range_values_clause::=、16-22 ページの table_partition_description::= を参照）

hash_partitions::=



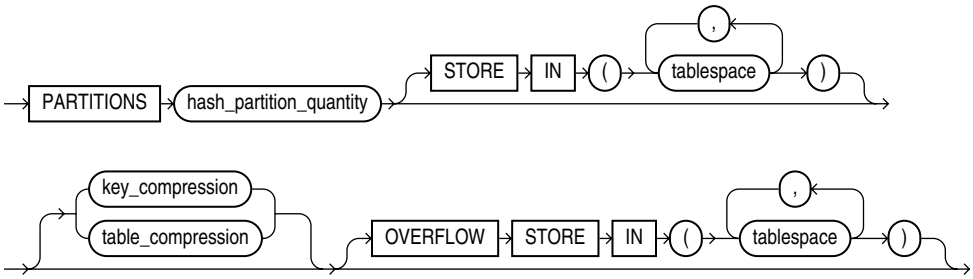
(16-18 ページの individual_hash_partitions::=、16-18 ページの hash_partitions_by_quantity::= を参照)

***individual_hash_partitions*::=**

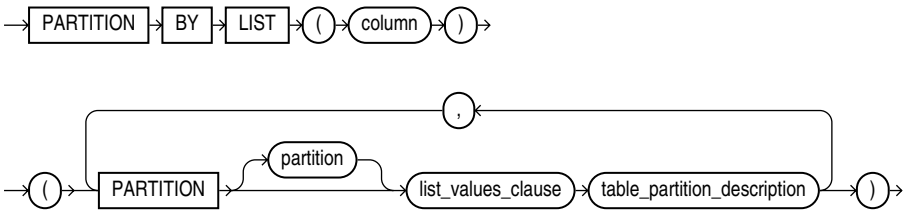


(16-22 ページの partitioning_storage_clause::= を参照)

***hash_partitions_by_quantity*::=**

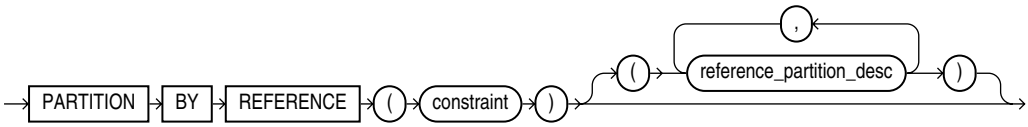


list_partitions::=



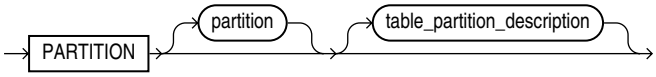
(16-22 ページの list_values_clause::=、16-22 ページの table_partition_description::= を参照)

reference_partitioning::=



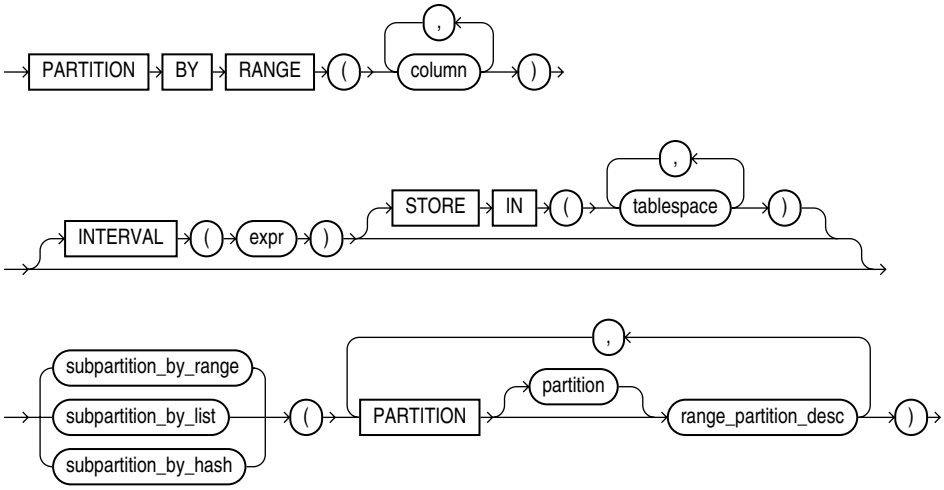
(16-19 ページの reference_partition_desc::= を参照)

***reference_partition_desc*::=**



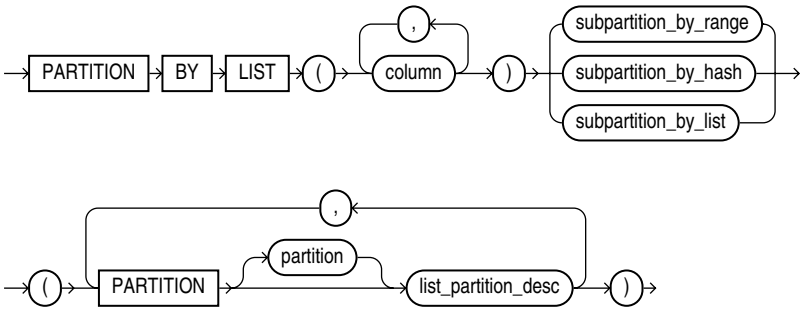
（16-22 ページの table_partition_description::= を参照）

composite_range_partitions::=



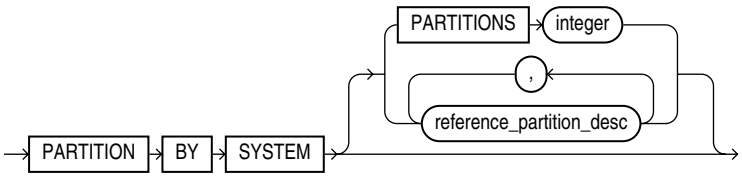
（16-21 ページの subpartition_by_range::=、16-21 ページの subpartition_by_list::=、
16-21 ページの subpartition_by_hash::=、16-20 ページの range_partition_desc::= を参照）

composite_list_partitions::=



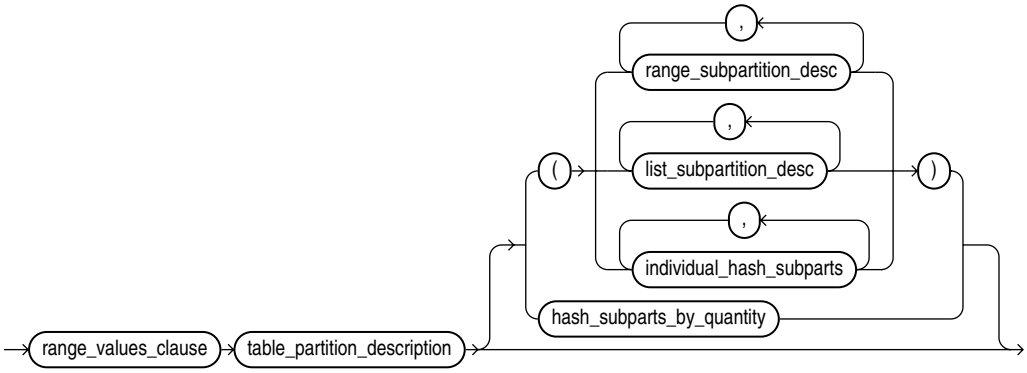
（16-21 ページの subpartition_by_range::=、16-21 ページの subpartition_by_list::=、
16-21 ページの subpartition_by_hash::=、16-20 ページの list_partition_desc::= を参照）

system_partitioning::=



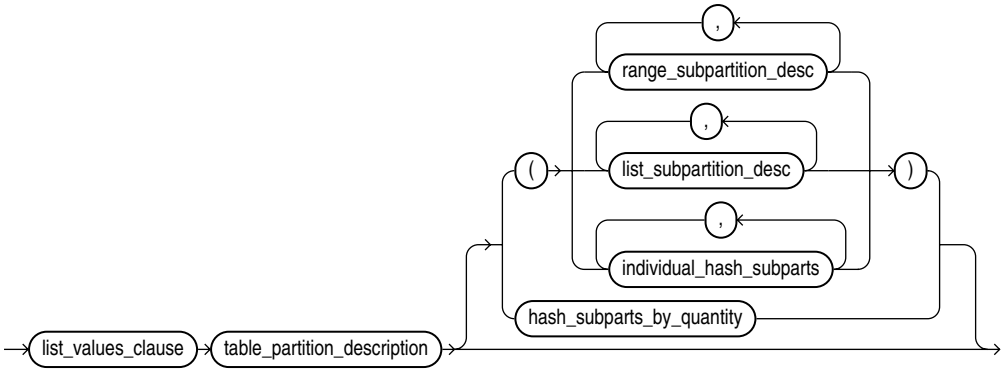
（16-19 ページの reference_partition_desc::= を参照）

***range_partition_desc*::=**



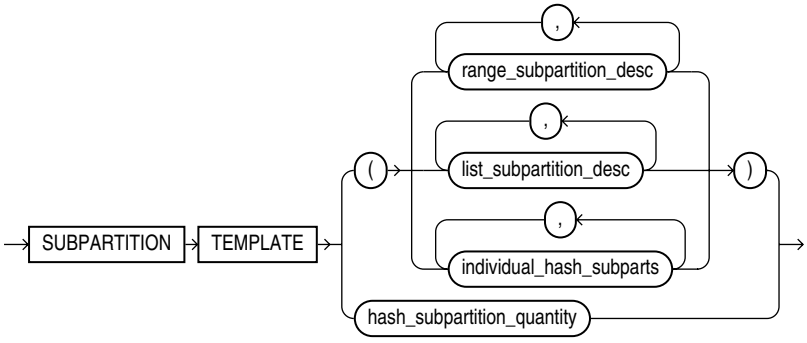
（16-22 ページの range_values_clause::=、16-22 ページの table_partition_description::=、16-21 ページの range_subpartition_desc::=、16-21 ページの list_subpartition_desc::=、16-21 ページの individual_hash_subparts::=、16-22 ページの hash_subparts_by_quantity::= を参照）

***list_partition_desc*::=**



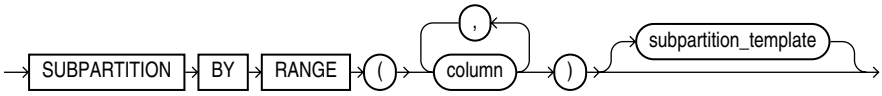
（16-22 ページの list_values_clause::=、16-22 ページの table_partition_description::=、16-21 ページの range_subpartition_desc::=、16-21 ページの list_subpartition_desc::=、16-21 ページの individual_hash_subparts::=、16-22 ページの hash_subparts_by_quantity::= を参照）

subpartition_template::=



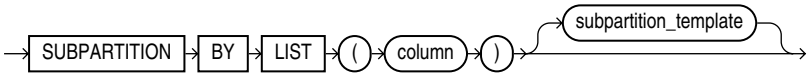
（16-21 ページの range_subpartition_desc::=、16-21 ページの list_subpartition_desc::=、16-21 ページの individual_hash_subparts::=、16-22 ページの hash_subparts_by_quantity::= を参照）

***subpartition_by_range*::=**



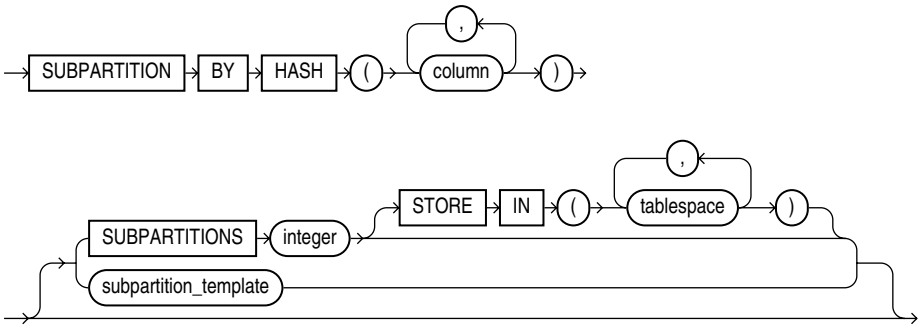
（16-20 ページの subpartition_template::= を参照）

***subpartition_by_list*::=**



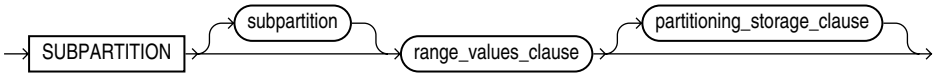
（16-20 ページの subpartition_template::= を参照）

subpartition_by_hash::=



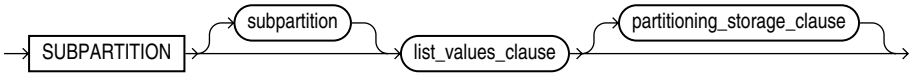
（16-20 ページの subpartition_template::= を参照）

***range_subpartition_desc*::=**



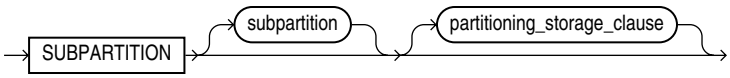
（16-22 ページの range_values_clause::=、16-22 ページの partitioning_storage_clause::= を参照）

***list_subpartition_desc*::=**



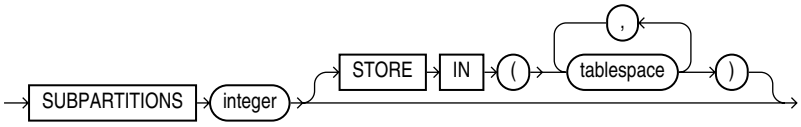
（16-22 ページの list_values_clause::=、16-22 ページの partitioning_storage_clause::= を参照）

***individual_hash_subparts*::=**

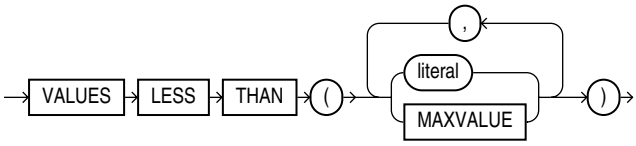


（16-22 ページの partitioning_storage_clause::= を参照）

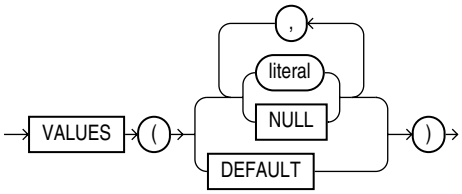
***hash_subparts_by_quantity*::=**



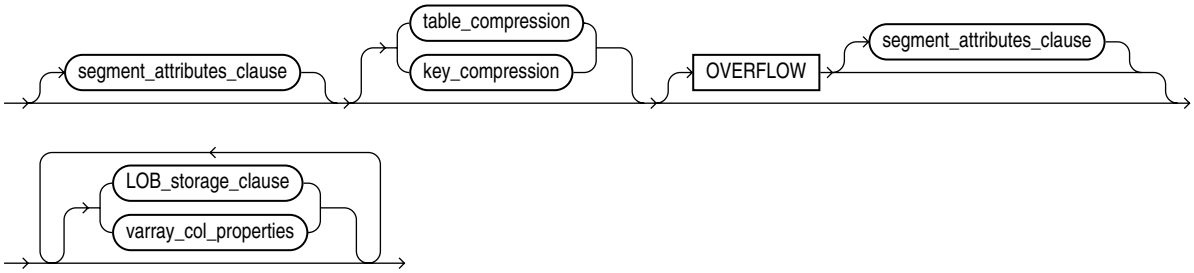
***range_values_clause*::=**



***list_values_clause*::=**

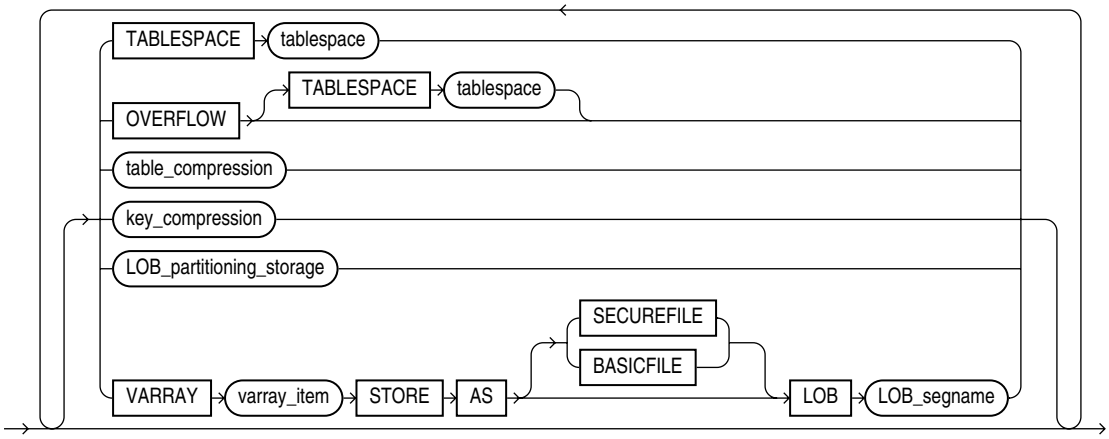


table_partition_description::=



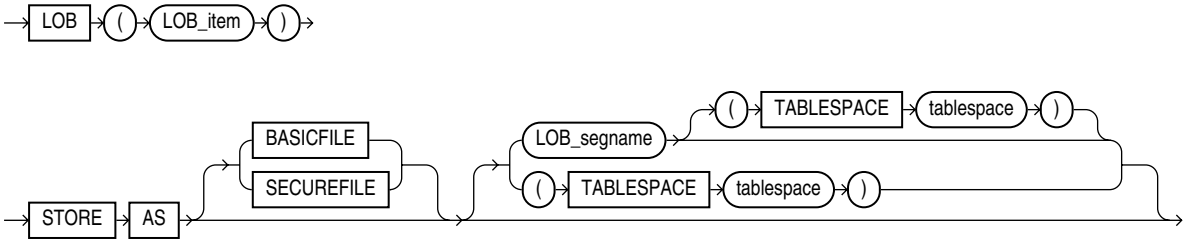
（16-10 ページ segment_attributes_clause::=、16-10 ページの table_compression::=、16-16 ページの key_compression::=、16-12 ページの LOB_storage_clause::=、16-12 ページの varray_col_properties::= を参照）

partitioning_storage_clause::=

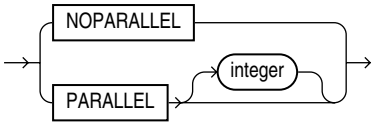


（16-10 ページの table_compression::= を参照）

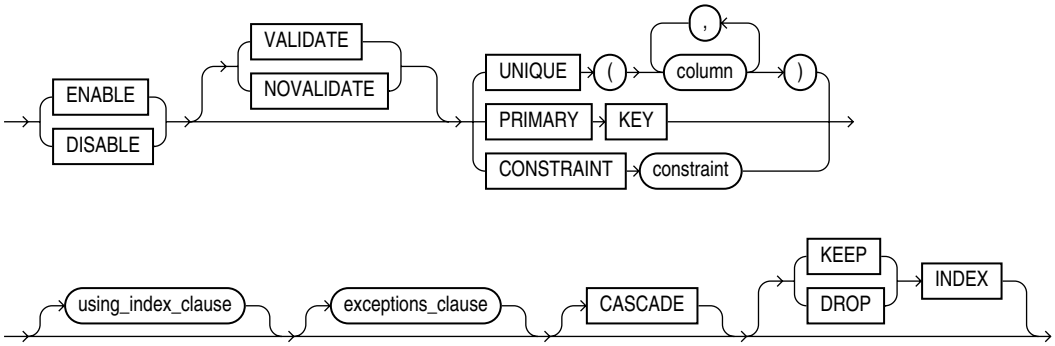
***LOB_partitioning_storage*::=**



parallel_clause::=

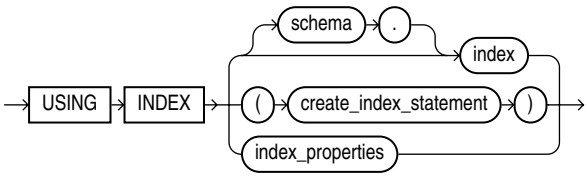


enable_disable_clause::=



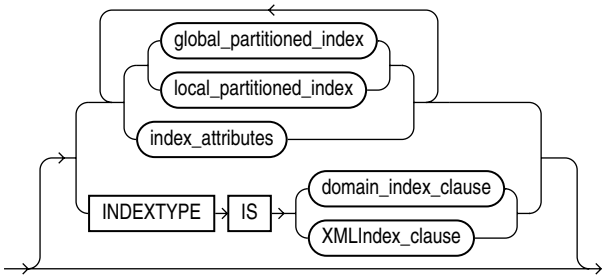
（16-23 ページの using_index_clause::= を参照。*exceptions_clause* は、CREATE TABLE 文ではサポートされていません。）

using_index_clause::=



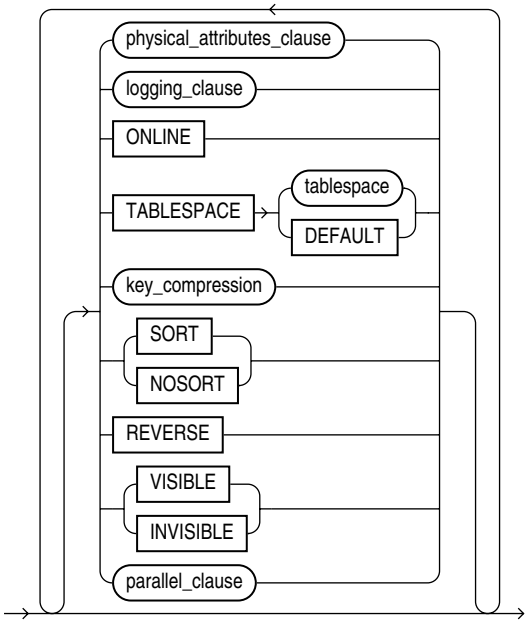
（14-51 ページの create_index::=、16-24 ページの index_properties::= を参照）

***index_properties*::=**



（「CREATE INDEX」の項にある 14-53 ページの global_partitioned_index::= と 14-55 ページの local_partitioned_index::=、および 14-52 ページの index_attributes::= を参照。*domain_index_clause* および *XMLIndex_clause* は、*using_index_clause* ではサポートされていません。）

***index_attributes*::=**



（16-10 ページの physical_attributes_clause::=、8-34 ページの logging_clause::=、16-16 ページの key_compression::= を参照。*parallel_clause* は、*using_index_clause* ではサポートされていません。）

セマンティクス

relational_table

GLOBAL TEMPORARY

GLOBAL TEMPORARY を指定すると、表が一時的で、適切な権限を持つすべてのセッションからその**定義**が参照できることを指定できます。一時表の**データ**は、データを表に挿入するセッションでのみ参照できます。

初めて一時表を作成した時点では、その表のメタデータはデータ・ディクショナリに格納されますが、表データの領域は割り当てられません。表セグメントの領域は、その表に初めて DML 操作を実行したときに割り当てられます。一時表の定義は、標準的な表の定義と同じように維持されますが、表に含まれる表セグメントとデータは、**セッション固有**または**トランザクション固有**のデータのいずれかになります。表セグメントとデータがセッション固有であるか、トランザクション固有であるかは、ON COMMIT キーワードで指定します。

セッションがバインドされていない場合にのみ、一時表で DDL 操作（ALTER TABLE、DROP TABLE、CREATE INDEX など）を実行できます。セッションを一時表にバインドするには、一時表で INSERT 操作を実行します。セッションを一時表からアンバインドするには、TRUNCATE 文を発行するか、セッションを終了します。また、トランザクション固有の一時表からアンバインドするには、COMMIT または ROLLBACK 文を発行します。

> **参照：** 一時表の詳細は、『Oracle Database 概要』および 16-61 ページの「一時表の例 :」を参照してください。

一時表の制限事項： 一時表には、次の制限事項があります。

- 一時表は、パーティション化、索引構成化またはクラスタ化できません。
- 一時表には、外部キー制約を指定できません。
- 一時表は、ネストした表の列を含むことはできません。
- *LOB_storage_clause* の TABLESPACE、*storage_clause* または *logging_clause* は指定できません。
- 一時表にパラレル UPDATE、DELETE および MERGE はサポートされていません。
- *nested_table_col_properties* または *parallel_clause* は指定できません。
- 一時表に対して指定できる *segment_attributes_clause* は、TABLESPACE のみです。
- 一時表での分散トランザクションはサポートされていません。

schema

表を含めるスキーマを指定します。*schema* を省略した場合、自分のスキーマ内に表が作成されます。

table

作成する表またはオブジェクト表の名前を指定します。

> **参照：** 「一般的な例」（16-60 ページ）

relational_properties

リレーショナル表のコンポーネントを指定します。

column_definition

列の性質を定義できます。

column

表の列の名前を指定します。

AS *subquery* を指定する場合、索引構成表を作成しないかぎり、*column* および *datatype* を省略できます。索引構成表の作成時に AS *subquery* を指定する場合は、*column* を指定し、*datatype* を省略する必要があります。

表の列の絶対最大数は 1000 です。オブジェクト表、またはオブジェクトの列、ネストした表、VARRAY または REF 型のリレーショナル表を作成する場合、制限の 1000 列までをカウントする有効な非表示列を作成して、ユーザー定義型の列をリレーショナル列にマップします。

datatype

列のデータ型を指定します。

表の列のデータ型の注意事項： 表の列のデータ型には、次の注意事項があります。

- AS *subquery* を指定する場合、*datatype* を省略できます。索引構成表を作成して AS *subquery* を指定する場合は、データ型を省略する必要があります。
- 参照整合性制約の外部キーの一部として、文で列が指定されている場合にも、*datatype* を省略できます。Oracle Database では、参照整合性制約の参照キーに対応する列のデータ型が列に自動的に割り当てられます。
- LONG 列を持つ表は作成しないでください。かわりに、LOB 列（CLOB、NCLOB または BLOB）を使用してください。LONG 列は、下位互換性のためにサポートされています。

表の列のデータ型の制限事項： ROWID 型の列を指定することはできますが、それらの列の値が有効な行 ID であることは保証されません。

> **参照：** LONG 列および Oracle が提供するデータ型については、2-2 ページの「データ型」を参照してください。

SORT

SORT キーワードは、この表をハッシュ・クラスタの一部として作成する場合、およびクラスタ列でもある列にのみ有効です。

この句を指定すると、データベースに対して、ハッシュ・ファンクションを適用する前にこの列でクラスタの行をソートするように指示できます。これによって、クラスタ化データでの後続の操作時に、応答時間が短縮される場合があります。

> **参照：** クラスタ表の作成の詳細は、16-36 ページの「CLUSTER 句」を参照してください。

DEFAULT

DEFAULT 句を指定すると、後続の INSERT 文が列の値を省略した場合に列に割り当てられる値を指定できます。式のデータ型は、列のデータ型と一致する必要があります。列には、この式が入る長さが必要です。

DEFAULT 式には、リテラル引数、列の参照またはネストしたファンクションの起動を戻さない、任意の SQL ファンクションを含めることができます。

デフォルトの列値の制限事項： DEFAULT 式に、PL/SQL ファンクション、他の列、疑似列 CURRVAL、NEXTVAL、LEVEL、PRIOR および ROWNUM への参照または完全に指定されていない日付定数は指定できません。

> **参照：** *expr* の構文については、6-2 ページの「SQL 式」を参照してください。

encryption_spec

ENCRYPT 句を指定すると、データの透過的暗号化機能を利用して、定義する列を暗号化できます。暗号化できる列の型は、CHAR、NCHAR、VARCHAR2、NVARCHAR2、NUMBER、DATE、LOB および RAW です。列を暗号化するユーザーなど、認可されたユーザーには、データは暗号化された形で表示されません。

注意： 列を暗号化するには、適切な権限を持つシステム管理者が、セキュリティ・モジュールを初期化し、ウォレットをオープンし、暗号化キーを設定しておく必要があります。暗号化の一般的な情報については、『Oracle Database Advanced Security 管理者ガイド』を参照してください。関連する ALTER SYSTEM 文については、11-61 ページの「alter_system_security_clauses」を参照してください。

USING '*encrypt_algorithm*'　この句では、使用するアルゴリズムの名前を指定できます。有効なアルゴリズムは、3DES168、AES128、AES192 および AES256 です。この句を省略すると、AES192 が使用されます。同じ表内の複数の列を暗号化するときに、ある 1 つの列に対して USING 句を指定した場合は、暗号化する他のすべての列についても同じアルゴリズムを指定する必要があります。

IDENTIFIED BY password　この句を指定すると、指定したパスワードから列のキーが導出されます。

SALT | NO SALT　デフォルトでは、列のクリア・テキストに SALT と呼ばれるランダムな文字列が追加されてから、そのテキストが暗号化されます。このデフォルトの動作により、暗号化された列にはいくつかの制限があります。

- 列を索引キーとして使用する場合は、NO SALT を指定する必要があります。このような場面で使用される SALT の詳細は、『Oracle Database Advanced Security 管理者ガイド』を参照してください。
- 列の暗号化中に SALT を指定した場合は、表に対して表の圧縮を指定した場合でも、暗号化される列内のデータは圧縮されません。ただし、SALT パラメータを指定しない場合は、暗号化されない列および暗号化される列のデータは圧縮されます。

LOB 暗号化に対しては、SALT または NO SALT を指定できません。

***encryption_clause* の制限事項 :**　列の暗号化には、次の制限事項があります。

- データの透過的暗号化は、従来のインポート / エクスポート・ユーティリティまたはトランスポータブル表領域ベースのエクスポートによってサポートされていません。かわりに、暗号化された列には、データ・ポンプ・インポート / エクスポート・ユーティリティを使用してください。
- 外部表の列を暗号化する場合、その表のアクセス・タイプとして ORACLE_DATAPUMP が使用されている必要があります。
- SYS が所有する表の列は暗号化できません。
- 外部キーの列は、暗号化できません。

> **参照 :**　データの透過的暗号化の詳細は、『Oracle Database Advanced Security 管理者ガイド』を参照してください。

virtual_column_definition

virtual_column_definition 句によって、仮想列を作成できます。仮想列はディスクには格納されません。仮想列の値は、一連の式またはファンクションを計算することによって必要に応じて導出されます。仮想列は、問合せ、DML および DDL 文で使用できます。索引付けが可能であり、統計を収集できます。したがって、他の列と同様に処理できます。例外と制限については、次の 16-28 ページの「仮想列の注意事項 :」および 16-28 ページの「仮想列の制限事項 :」で説明します。

- *column* には、仮想列の名前を指定します。
- オプションで、仮想列のデータ型を指定できます。datatype を省略すると、列のデータ型は基礎となる式のデータ型に基づいて決定されます。すべての Oracle スカラー・データ型および XMLType がサポートされています。
- キーワード GENERATED ALWAYS は、構文を明確にするために使用されます。列はディスクには格納されませんが、必要に応じて評価されることが示されます。
- AS *column_expr* 句によって、列の内容が決まります。*column_expr* の詳細は、6-6 ページの「列式」を参照してください。
- キーワード VIRTUAL はオプションであり、構文を明確にするためのものです。

仮想列の注意事項 :

- *column_expr* が列レベルのセキュリティが実装された列を参照する場合、仮想列は基本列のセキュリティ・ルールを継承しません。この場合は、仮想列に対して列レベルのセキュリティ・ポリシーを複製するか、またはデータを暗黙的にマスクするファンクションを適用して、仮想列のデータを保護する必要があります。たとえば、一般的にクレジット・カード番号は列レベルのセキュリティ・ポリシーで保護しますが、コール・センターの従業員に対しては確認目的でクレジット・カード番号の下 4 桁を参照できるようにします。このような場合、クレジット・カード番号の下 4 桁のサブストリングを取るように仮想列を定義できます。
- 仮想列に定義された表の索引は、表のファンクション索引と同じです。
- 仮想列を直接更新することはできません。したがって、UPDATE 文の SET 句に仮想列を指定することはできません。ただし、UPDATE 文の WHERE 句には仮想列を指定できます。同様に、DELETE 文の WHERE 句に仮想列を指定して、仮想列の導出値に基づいて表から行を削除できます。
- 仮想列を含む表を FROM 句に指定する問合せは、結果キャッシュに適応します。結果キャッシュの詳細は、2-93 ページの「RESULT_CACHE ヒント」を参照してください。
- 作成時にファンクションに DETERMINISTIC が明示的に指定されている場合、*column_expr* は、PL/SQL ファンクションを参照できます。ただし、後でファンクションが置き換えられた場合、仮想列に依存する定義は無効にされません。そのような場合、表にデータが含まれていると、仮想列が制約、索引またはマテリアライズド・ビューの定義あるいは結果キャッシュで使用された場合に、仮想列を参照する問合せで不適切な結果が戻される場合があります。そのため、仮想列の決定的な PL/SQL ファンクションを置き換えるために、次の手順を実行します。
  - 仮想列の制約を無効にして再度有効にします。
  - 仮想列の索引を再作成します。
  - 仮想列にアクセスするマテリアライズド・ビューを完全にリフレッシュします。
  - キャッシュされた問合せが仮想列にアクセスした場合、結果キャッシュをフラッシュします。
  - 表の統計情報を再収集します。

仮想列の制限事項 :

- 仮想列は、リレーショナル・ヒープ表にのみ作成できます。仮想列は、索引構成表、外部表、オブジェクト表、クラスタ化表または一時表ではサポートされません。
- AS 句の *column_expr* には、次の制限事項があります。
  - 別の仮想列を名前で参照できません。
  - *column_expr* で参照される列は、同じ表で定義されている必要があります。
  - 決定的なユーザー定義ファンクションを参照できますが、その場合、仮想列をパーティション化キー列として使用できません。
  - *column_expr* の出力は、スカラー値である必要があります。

    参照 : *column_expr* の詳細および制限事項は、6-6 ページの「列式」を参照してください。

- 仮想列は、Oracle が提供するデータ型、ユーザー定義型、または LOB または LONG RAW にすることはできません。

  参照 : 仮想列を持つ表の作成の例については、12-78 ページの「仮想表の列の追加例 :」および『Oracle Database 管理者ガイド』を参照してください。

制約句

制約句を使用すると、表の列に対する制約を作成できます。DEFERRABLE 以外の主キー制約を索引構成表に指定してください。これらの制約の構文、詳細および使用例は、8-4 ページの「constraint」を参照してください。

inline_ref_constraint* | *out_of_line_ref_constraint これらの句を使用すると、REF 型の列を指定できます。これらの句の唯一の違いは、表レベルで *out_of_line_ref_constraint* を指定することです。このため、定義する REF 型の列または属性を識別する必要があります。*inline_ref_constraint* は、REF 型の列または属性の定義の一部として指定してください。

> **参照：** 「REF 制約の例」（8-22 ページ）

inline_constraint *inline_constraint* を使用すると、整合性制約を列定義の一部として定義できます。

オブジェクト型の列のスカラー属性に、一意制約、主キー制約および参照制約を作成できます。また、オブジェクト型の列の NOT NULL 制約、オブジェクト型の列またはオブジェクト型の列の属性を参照する CHECK 制約も作成できます。

out_of_line_constraint *out_of_line_constraint* 構文を使用すると、整合性制約を表定義の一部として定義できます。

supplemental_logging_props

supplemental_logging_props 句を指定すると、追加のデータがログ・ストリームに入れられ、ログに基づくツール製品をサポートできます。

supplemental_log_grp_clause この句を使用すると、名前付きのログ・グループを作成できます。

- NO LOG 句を使用すると、REDO ログから 1 つ以上の列を省略できます。この句を指定しない場合、これらの列は名前付きのログ・グループの REDO に含まれます。名前付きのログ・グループに、1 つ以上の固定長列を NO LOG を使用せずに指定する必要があります。
- ALWAYS を指定すると、更新時にログ・グループのすべての列が REDO に含まれます。関連付けられた行が変更されるとログ・グループのすべての列に対してサプリメンタル・ロギングが行われるため、これは**無条件ログ・グループ**といいます（「常時ログ・グループ」ともいいます）。ALWAYS を指定しない場合、ログ・グループの任意の列が変更された場合のみ、ログ・グループのすべての列に対してサプリメンタル・ロギングが行われます。これは、**条件付きログ・グループ**といいます。

サプリメンタル・ロギングが指定されているかどうかを確認するには、適切な USER_、ALL_ または DBA_LOG_GROUP_COLUMNS データ・ディクショナリ・ビューを問い合せます。

supplemental_id_key_clause この句を使用すると、主キー列、一意キー列および外部キー列のすべて、またはこれらの列の組合せに対してサプリメンタル・ロギングを実行できます。Oracle Database は、**無条件ログ・グループ**または**条件付きログ・グループ**のいずれかを生成します。無条件ログ・グループでは、関連付けられた行が変更されると、ログ・グループのすべての列に対してサプリメンタル・ロギングが行われます。条件付きログ・グループでは、ログ・グループの任意の列が変更された場合のみ、ログ・グループのすべての列に対してサプリメンタル・ロギングが行われます。

- ALL COLUMNS を指定すると、この行の最大サイズが固定長のすべての列が REDO ログに含まれます。このような REDO ログは、システム生成無条件ログ・グループといいます。
- PRIMARY KEY COLUMNS を指定すると、主キーを持つすべての表において、更新が実行されるたびに、主キーのすべての列が REDO ログに置かれます。Oracle Database は、次のとおりサプリメンタル・ロギングを行う列を評価します。
  - まず、主キー制約が指定されている列が選択されます（制約が検証済か、または制約に RELY のマークが付いていて、DISABLED および INITIALLY DEFERRED のマークが付いていない場合）。

  - 主キー列が存在しない場合、1 つ以上の NOT NULL 列を持つ最小の UNIQUE 索引が検索され、この索引の列が使用されます。
  - このような索引が存在しない場合、表のすべてのスカラー列に対してサプリメンタル・ロギングが行われます。

- UNIQUE COLUMNS を指定すると、一意キーまたはビットマップ索引を持つすべての表において、一意キー列またはビットマップ索引列が変更された場合、一意キーまたはビットマップ索引に属するその他のすべての列も REDO ログに置かれます。このようなログ・グループは、システム生成条件付きログ・グループといいます。
- FOREIGN KEY COLUMNS を指定すると、外部キーを持つすべての表において、外部キー列が変更された場合、外部キーに属するその他のすべての列も REDO ログに置かれます。このような REDO ログは、システム生成条件付きログ・グループといいます。

この句を複数回指定すると、指定するたびに個別のログ・グループが作成されます。サプリメンタル・ロギング・データが指定されているかどうかを確認するには、適切な USER_、ALL_ または DBA_LOG_GROUPS データ・ディクショナリ・ビューを問い合せます。

ON COMMIT

ON COMMIT 句は、一時表を作成する場合のみに適用されます。この句を使用すると、一時表のデータがトランザクションまたはセッションの存続期間中保持されるかどうかを指定できます。

DELETE ROWS DELETE ROWS は、トランザクション固有の一時表に対して指定します。これはデフォルトです。各コミット後に表が切り捨てられます（すべての行が削除されます）。

PRESERVE ROWS PRESERVE ROWS は、セッション固有の一時表に対して指定します。セッション終了時に表が切り捨てられます（すべての行が削除されます）。

physical_properties

物理プロパティは、エクステントとセグメントの処理、および表の記憶特性に関係します。

segment_attributes_clause

segment_attributes_clause を指定すると、表の物理属性および表領域の記憶域を指定できます。

physical_attributes_clause *physical_attributes_clause* を指定すると、PCTFREE、PCTUSED、INITRANS パラメータの値、および表の記憶特性を指定できます。

- 非パーティション表の場合、指定した各パラメータおよび記憶特性は、表に関連付けられたセグメントの実際の物理属性となります。
- パーティション表の場合、パーティション作成文の PARTITION 句で明示的に値を上書きしないかぎり、指定したパラメータおよび記憶特性の値は、CREATE 文（および後続の ALTER TABLE ... ADD PARTITION 文）で指定するすべてのパーティションに関連付けられたセグメントのデフォルト物理属性になります。

この句を省略すると、PCTFREE は 10、PCTUSED は 40、INITRANS は 1 に設定されます。

参照：

- これらの句の詳細は、8-39 ページの「physical_attributes_clause」および 8-41 ページの「storage_clause」を参照してください。
- 「記憶域の例：」（16-60 ページ）

TABLESPACE Oracle Database が、表、オブジェクト表 OIDINDEX、パーティション、LOB のデータ・セグメント、LOB の索引セグメントまたは索引構成表のオーバーフロー・データ・セグメントを作成する表領域を指定します。TABLESPACE を省略した場合、その表を含むスキーマの所有者のデフォルトの表領域内に作成されます。

1 つ以上の LOB 列を持つヒープ構成表の場合、LOB 記憶域に対する TABLESPACE を省略すると、表を作成する表領域に LOB データおよび索引セグメントが作成されます。

1 つ以上の LOB 列を持つ索引構成表の場合、TABLESPACE を省略すると、索引構成表の主キー索引セグメントが作成された表領域に、LOB データおよび索引セグメントが作成されます。

非パーティション表の場合、TABLESPACE に指定する値は、表に関連付けられたセグメントの実際の物理属性となります。パーティション表の場合、TABLESPACE に指定する値は、PARTITION 記述で TABLESPACE を指定しないかぎり、この CREATE 文（および後続の ALTER TABLE ... ADD PARTITION 文）で指定されたすべてのパーティションに関連付けられたセグメントのデフォルト物理属性となります。

> **参照：** 表領域の詳細は、16-71 ページの「CREATE TABLESPACE」を参照してください。

logging_clause

表、および制約のために必要な索引、パーティションまたは LOB の記憶特性の作成を REDO ログ・ファイルに記録する（LOGGING）かしないか（NOLOGGING）を指定します。表のロギング属性は、その索引の属性に依存しません。

表、パーティションまたは LOB の記憶域に対して、後で実行されるダイレクト・ローダー（SQL*Loader）操作およびダイレクト・パス・インサート操作のログをとる（LOGGING）かとらない（NOLOGGING）かも指定します。

この句の詳細は、8-34 ページの「logging_clause」を参照してください。

table_compression

table_compression 句は、ヒープ構成表に対してのみ有効です。この句を使用すると、ディスク使用量を削減するためにデータ・セグメントを圧縮するかどうかを指定できます。この句は、挿入操作や更新操作が少ないデータ・ウェアハウスなどの環境や、OLTP 環境で特に有効です。COMPRESS キーワードを指定すると、表の圧縮が使用可能になります。NOCOMPRESS キーワードを指定すると、表の圧縮が使用禁止になります。デフォルトは NOCOMPRESS です。

- COMPRESS または COMPRESS FOR DIRECT_LOAD OPERATIONS を指定して表の圧縮を使用可能にすると、Oracle Database では、表の圧縮が効果的である場合に、ダイレクト・パス・インサート操作中に表の圧縮を開始します。元のインポート・ユーティリティ（imp）はダイレクト・パス・インサートをサポートしないため、圧縮フォーマットでデータをインポートすることはできません。
- COMPRESS FOR ALL OPERATIONS を指定して表の圧縮を使用可能にすると、Oracle Database は表でのすべての DML 操作中にデータの圧縮を開始します。

> **注意：** COMPRESS または COMPRESS FOR DIRECT_LOAD OPERATIONS が指定された表では、*physical_attributes_clause* で PCTFREE の値を明示的に設定しないかぎり、PCTFREE 値に 0 を使用して圧縮を最大限にします。COMPRESS FOR ALL OPERATIONS または NOCOMPRESS が指定された表では、PCTFREE のデフォルトを明示的に上書きしないかぎり、デフォルト値 10 を使用して、圧縮を最大限にしながらデータへの今後の DML 変更も考慮します。

表の圧縮は、ヒープ構成表の次の部分に対して指定できます。

- ヒープ構成表全体の場合は、*relational_table* または *object_table* の *physical_properties* 句で指定します。
- レンジ・パーティションの場合は、*range_partitions* 句の *table_partition_description* で指定します。
- コンポジット・レンジ・パーティションの場合は、*range_partition_desc* 句の *table_partition_description* で指定します。

- コンポジット・リスト・パーティションの場合は、*list_partition_desc* 句の *table_partition_description* で指定します。
- リスト・パーティションの場合は、*list_partitions* 句の *table_partition_description* で指定します。
- システム・パーティションまたは参照パーティションの場合は、*reference_partition_description* 句の *table_partition_description* で指定します。
- ネストした表の記憶表の場合は、*nested_table_col_properties* 句で指定します。

参照：

- ダイレクト・パス・インサート操作の制限などの詳細は、18-53 ページの「従来型 INSERT およびダイレクト・パス・インサート」を参照してください。
- 表の圧縮の使用例は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

表の圧縮の制限事項： 表の圧縮には、次の制限事項があります。

- 表の圧縮は、列数が 255 を超える表ではサポートされません。
- BasicFile LOB のデータ・セグメントは圧縮されません。SecureFile LOB の圧縮の詳細は、16-40 ページの「LOB_compression_clause」を参照してください。
- 索引構成表、オーバーフロー・セグメント、オーバーフロー・セグメントのパーティションまたは索引構成表のマッピング表セグメントには、表の圧縮を指定できません。
- 外部表またはクラスタの一部である表には、表の圧縮を指定できません。
- ダイレクト・ロード操作用に圧縮された表の列は削除できませんが、未使用に設定することはできます。すべての操作用に圧縮された表では、ALTER TABLE ... *drop_column_clause* のすべての操作が有効です。

RECOVERABLE | UNRECOVERABLE

これらのキーワードは以前のリリースで非推奨になったもので、それぞれ LOGGING および NOLOGGING に置き換えられています。RECOVERABLE および UNRECOVERABLE は、下位互換性のためにサポートされていますが、LOGGING および NOLOGGING キーワードを使用することをお薦めします。

[UN]RECOVERABLE の制限事項： この句には、次の制限事項があります。

- パーティション表または LOB 記憶特性に RECOVERABLE を指定できません。
- パーティション表または索引構成表に UNRECOVERABLE を指定できません。
- AS *subquery* でのみ UNRECOVERABLE を指定できます。

ORGANIZATION

ORGANIZATION 句を指定すると、表のデータ行が格納される順序を指定できます。

HEAP HEAP を使用すると、*table* のデータ行の格納順序を特定しないことを指定できます。これはデフォルトです。

INDEX INDEX を使用すると、*table* を索引構成表として作成することを指定できます。索引構成表では、表の主キーが定義された索引内にデータ行が格納されます。

EXTERNAL EXTERNAL を使用すると、表がデータベースの外部にある読取り専用表であることを指定できます。

参照 : 「外部表の例 :」(16-64 ページ)

index_org_table_clause

index_org_table_clause を使用すると、索引構成表を作成できます。表の行(主キー列の値と非キー列の値の両方)は、主キーに基づいて作成された索引に格納されます。このため、索引構成表は主キーベースのアクセスおよび操作に最適です。索引構成表は、次のいずれかの表のかわりです。

- CREATE INDEX 文を使用して主キーベースで索引付けされるクラスタ化されていない表。
- 索引クラスタに格納されるクラスタ表。索引クラスタは、表に対する主キーをクラスタ・キーにマップする CREATE CLUSTER 文を使用して作成されます。

主キーは行を一意に識別するため、索引構成表には主キーを指定してください。主キーには DEFERRABLE を指定できません。索引構成表の行に直接アクセスする場合は、ROWID のかわりに主キーを使用してください。

索引構成表がパーティション化され、LOB 列を含む場合、最初に *index_org_table_clause*、次に *LOB_storage_clause*、その後に適切な *table_partitioning_clauses* を指定する必要があります。

索引構成表を作成する場合は、CREATE TABLE ... AS SELECT 文の副問合せで、TO_LOB ファンクションを使用して LONG 列を LOB 列に変換することはできません。LONG 列を含まない索引構成表を作成し、INSERT ...AS SELECT 文で TO_LOB ファンクションを使用してください。

索引構成表の ROWID 疑似列は、物理 ROWID ではなく、論理 ROWID を戻します。データ型 ROWID として作成した列には、IOT の論理 ROWID を格納できません。データ型 ROWID の列に格納できるデータは、ヒープ構成表の ROWID のみです。IOT の論理 ROWID を格納する場合は、かわりに型 UROWID の列を作成します。データ型 UROWID の列には、物理 ROWID と論理 ROWID の両方を格納できます。

参照 : 「索引構成表の例 :」(16-63 ページ)

索引構成表の制限事項 : 索引構成表には、次の制限事項があります。

- 索引構成表の ROWID 疑似列は、物理 ROWID ではなく、論理 ROWID を戻します。データ型 ROWID として作成した列には、IOT の論理 ROWID を格納できません。データ型 ROWID の列に格納できるデータは、ヒープ構成表の ROWID のみです。IOT の論理 ROWID を格納する場合は、かわりに型 UROWID の列を作成します。データ型 UROWID の列には、物理 ROWID と論理 ROWID の両方を格納できます。
- 索引構成表には、仮想列は定義できません。
- 索引構成表には、*composite_partitioning_clause* は指定できません。

PCTTHRESHOLD *integer* 索引ブロック内で、索引構成表の行を格納するために確保されている領域の割合(パーセント)を指定します。PCTTHRESHOLD は、主キーを保持するために十分な大きさである必要があります。指定したしきい値を超える列から始まる行の後続列はすべて、オーバーフロー・セグメントに格納されます。PCTTHRESHOLD は 1 ～ 50 の値を取る必要があります。PCTTHRESHOLD を指定しない場合のデフォルト値は 50 です。

PCTTHRESHOLD の制限事項 : PCTTHRESHOLD は、索引構成表の個別パーティションに対して指定できません。

mapping_table_clauses MAPPING TABLE を指定すると、ローカルから物理 ROWID へのマッピングを作成してヒープ構成表に格納できます。このマッピングは、索引構成表のビットマップ索引の作成に必要です。索引構成表がパーティション化されている場合、マッピング表もパーティション化され、マッピング表のパーティションの名前および物理属性は実表のパーティションと同じになります。

マッピング表またはマッピング表のパーティションは、親である索引構成表またはパーティションと同じ表領域に作成されます。マッピング表またはそのパーティションの記憶特性に対して、問合せ、DML 操作または変更は実行できません。

key_compression *key_compression* を使用すると、索引構成表のキー圧縮を使用可能または使用禁止にできます。

- COMPRESS を指定すると、**キー圧縮**が使用可能になります。これによって、索引構成表の主キー列の値が重複しなくなります。*integer* を使用して、接頭辞の長さ（圧縮する接頭辞列数）を指定します。

  接頭辞の長さの有効範囲は、1 ～（主キー列数 -1）までです。デフォルトでは（主キー列数 -1）になります。

- NOCOMPRESS を指定すると、索引構成表でのキー圧縮が使用禁止になります。これはデフォルトです。

索引構成表のキー圧縮の制限事項： パーティション・レベルでは、COMPRESS を指定できますが、*integer* で接頭辞の長さを指定できません。

index_org_overflow_clause *index_org_overflow_clause* を指定すると、指定されたしきい値を超える索引構成表のデータ行を、この句で指定したデータ・セグメントに格納できます。

- 索引構成表を作成した場合、各列の最大サイズが評価され、行の最大値が計算されます。オーバーフロー・セグメントが必要で、OVERFLOW を指定していない場合は、エラーが発生し CREATE TABLE 文は実行されません。このチェック機能によって、索引構成表に対する後続の DML 操作が、オーバーフロー・セグメントがないために失敗することを回避できます。
- OVERFLOW キーワードの後の句に指定するすべての物理属性および記憶特性は、表のオーバーフロー・セグメントにのみ適用されます。索引構成表自体の物理属性と記憶特性、すべてのパーティションに対するデフォルト値、および各パーティションに対する値は、このキーワードの前に指定する必要があります。
- 索引構成表に 1 つ以上の LOB 列が含まれる場合は、LOB がインラインに格納できるほど小さい場合でも、OVERFLOW を指定しないと、アウトラインに格納されます。
- 表がパーティション化されている場合、オーバーフロー・データ・セグメントが主キー索引セグメントと同一レベルでパーティション化されます。

INCLUDING *column_name* 索引構成表の行を索引部分とオーバーフロー部分に分割する列を指定します。主キー列は常に索引に格納されます。*column_name* は、最後の主キー列でもその他の主キー以外の列でもかまいません。*column_name* に続くすべての主キー以外の列は、オーバーフロー・データ・セグメントに格納されます。

column_name で行を分割しようとした場合に、行の索引部分のサイズが、PCTTHRESHOLD の指定値またはデフォルト値を超えると、PCTTHRESHOLD の値に基づいて、行は切り離されます。

INCLUDING 句の制限事項： 索引構成表の個々のパーティションにこの句は指定できません。

external_table_clause

external_table_clause を使用すると、外部表を作成できます。外部表は読取り専用表で、そのメタデータはデータベースに格納されますが、データはデータベースの外部に格納されます。外部表では、データを最初はデータベースにロードせずに、データベースの外部でデータを問い合せることができます。

> **参照：** 外部表の使用方法の詳細は、『Oracle Database データ・ウェアハウス・ガイド』、『Oracle Database 管理者ガイド』および『Oracle Database ユーティリティ』を参照してください。

外部表の場合、データベースにデータが存在しないため、表の作成時に通常は使用可能な句の小規模のサブセットを定義します。

- *relational_properties* 句内では、*column* および *datatype* のみを指定できます。
- *physical_properties_clause* 内では、表の構成（ORGANIZATION EXTERNAL *external_table_clause*）のみを指定できます。
- *table_properties* 句内では、*parallel_clause* のみを指定できます。*parallel_clause* を使用すると、外部データに対する後続の問合せおよび外部表を移入する後続の操作をパラレル化できます。
- 外部表は、作成時に AS *subquery* 句を使用することによって移入できます。

同じ CREATE TABLE 文で他の句を指定することはできません。

参照：

- 「外部表の例：」（16-64 ページ）
- 列の投影のデフォルトのプロパティを変更することによる影響の詳細は、12-52 ページの「ALTER TABLE」の「PROJECT COLUMN 句」を参照してください。

外部表の制限事項： 外部表には、次の制限事項があります。

- 外部表を一時表にすることはできません。
- 外部表には制約を指定できません。
- 外部表に仮想列を含めることはできません。
- 外部表にオブジェクト型、VARRAY または LONG 列を含めることはできません。ただし、内部データベース表の VARRAY または LONG データを外部表の LOB 列に移入することはできます。

TYPE TYPE *access_driver_type* を指定すると、外部表の**アクセス・ドライバ**を指定できます。アクセス・ドライバは、データベースに対する外部データを解析する API です。Oracle Database では、ORACLE_LOADER および ORACLE_DATAPUMP の 2 つのアクセス・ドライバが提供されています。TYPE を指定しない場合、デフォルトのアクセス・ドライバ ORACLE_LOADER が使用されます。AS *subquery* 句を指定して 1 つの Oracle Database からデータをアンロードし、同じ、または異なる Oracle Database に再ロードする場合、ORACLE_DATAPUMP アクセス・ドライバを指定する必要があります。

参照： ORACLE_LOADER および ORACLE_DATAPUMP アクセス・ドライバの詳細は、『Oracle Database ユーティリティ』を参照してください。

DEFAULT DIRECTORY DEFAULT DIRECTORY を指定すると、外部データ・ソースが存在するファイル・システムのディレクトリに対応するデフォルト・ディレクトリ・オブジェクトを 1 つ指定できます。デフォルト・ディレクトリは、アクセス・ドライバから使用でき、エラー・ログなどの補助ファイルを格納できます。

ACCESS PARAMETERS オプションの ACCESS PARAMETERS 句を指定すると、その外部表用の特定のアクセス・ドライバのパラメータに値を割り当てることができます。

- *opaque_format_spec* 句は、ORACLE_LOADER および ORACLE_DATAPUMP アクセス・ドライバのすべてのアクセス・パラメータを指定します。これらのパラメータの詳細は、『Oracle Database ユーティリティ』を参照してください。

  opaque_format_spec で指定するフィールド名は、表定義の列と一致している必要があります。表定義の列と一致していない *opaque_format_spec* のフィールドは無視されます。

- USING CLOB *subquery* を指定すると、副問合せを使用して、パラメータおよびその値を導出できます。副問合せには、集合演算子または ORDER BY 句を含めません。1 つの CLOB データ型を含む単一行を戻します。

opaque_format_spec でパラメータを指定する場合、または副問合せを使用してそれらを導出する場合は、この句は解析されません。外部データのコンテキスト情報は、アクセス・ドライバが解析します。

LOCATION LOCATION 句を指定すると、1 つ以上の外部データ・ソースを指定できます。通常、*location_specifier* はファイルですが、ファイル以外も指定できます。Oracle Database はこの句を解析しません。外部データのコンテキスト情報は、アクセス・ドライバが解析します。*location_specifier* では、ワイルド・カードを使用した複数ファイルの指定はできません。

REJECT LIMIT REJECT LIMIT 句を指定すると、Oracle Database エラーが戻され、問合せが異常終了するまでに、外部データの問合せで許容される変換エラーの数を指定できます。デフォルト値は 0（ゼロ）です。

CLUSTER 句

CLUSTER 句は、表が *cluster* の一部であることを示します。この句で指定する各列は、クラスタの各列に対応する表の列となります。一般に、表のクラスタ列は、主キーまたは主キーの一部を構成する 1 つ以上の列です。詳細は、14-2 ページの「CREATE CLUSTER」を参照してください。

クラスタ・キー内の列ごとに表から 1 つの列を指定します。列は、名前ではなく位置で一致させます。

クラスタ表はクラスタの領域割当てを使用します。このため、PCTFREE、PCTUSED または INITRANS パラメータ、TABLESPACE 句または *storage_clause* を CLUSTER 句とともに使用しないでください。

クラスタ表の制限事項： クラスタ表には、次の制限事項があります。

- オブジェクト表、および LOB 列または Oracle が提供する Any* 型の列を含む表はクラスタの一部にはできません。
- クラスタの一部である表に *parallel_clause*、CACHE または NOCACHE は指定できません。
- クラスタが同じ ROWDEPENDENCIES または NOROWDEPENDENCIES 設定で作成されていないかぎり、CLUSTER を ROWDEPENDENCIES または NOROWDEPENDENCIES とともに指定することはできません。

table_properties

table_properties を使用すると、表の特性をさらに詳しく定義できます。

column_properties

column_properties 句を使用すると、列の記憶域属性を指定できます。

object_type_col_properties

object_type_col_properties を使用すると、オブジェクト列、属性、あるいは列または属性の集合要素の記憶特性を指定できます。

column *column* には、オブジェクト列または属性を指定します。

substitutable_column_clause *substitutable_column_clause* を使用すると、同じ階層のオブジェクト列または属性が互いに置換可能かどうかを指定できます。列が特定の型であるか、サブタイプのインスタンスを含むものであるか、またはその両方を指定できます。

- ELEMENT を指定すると、コレクション列または属性の要素型が宣言した型のサブタイプに制約されます。
- IS OF [TYPE] (ONLY *type*) 句を指定すると、オブジェクト列の型が宣言した型のサブタイプに制約されます。

- NOT SUBSTITUTABLE AT ALL LEVELS を指定すると、オブジェクト列がサブタイプに対応するインスタンスを持つことはできないことを指定できます。また、置換は、埋込みオブジェクト属性、埋込みのネストした表および VARRAY の要素には使用できません。デフォルトは、SUBSTITUTABLE AT ALL LEVELS です。

***substitutable_column_clause* の制限事項 :** この句には、次の制限事項があります。

- この句は、オブジェクト列の属性には指定できません。ただし、オブジェクト表自体の代替性が設定されていない場合、リレーショナル表におけるオブジェクト型の列およびオブジェクト表のオブジェクト列に対してこの句を指定できます。
- コレクション型の列の場合、この句で指定できる部分は [NOT] SUBSTITUTABLE AT ALL LEVELS のみです。

LOB_storage_clause

LOB_storage_clause を使用すると、LOB データ・セグメントの記憶域属性を指定できます。STORE AS キーワードの後に、1 つ以上の句を指定する必要があります。複数の句を指定する場合は、構文図で上から下に表示されている順に指定する必要があります。

非パーティション表の場合、この句は、表の LOB データ・セグメントの記憶域属性を指定します。

パーティション表の場合、この句は指定した位置に応じて実装されます。

- 表レベルで指定されたパーティション表の場合（パーティション句とともに *physical_properties* 句で指定した場合）、この句は、各パーティションまたはサブパーティションに関連付けられた LOB データ・セグメントに対するデフォルト記憶域属性を指定します。この記憶域属性は、パーティションまたはサブパーティション・レベルで *LOB_storage_clause* によって上書きされないかぎり、すべてのパーティションまたはサブパーティションに適用されます。
- パーティション表の各パーティションの場合（*table_partition_description* の一部として指定した場合）、この句は、そのパーティションのデータ・セグメントの記憶域属性、またはこのパーティションのサブパーティションのデフォルト記憶域属性を指定します。パーティション・レベルの *LOB_storage_clause* は、表レベルの *LOB_storage_clause* を上書きします。
- パーティション表の各サブパーティションの場合（*subpartition_by_hash* または *subpartition_by_list* の一部として指定した場合）、この句は、サブパーティションのデータ・セグメントの記憶域属性を指定します。サブパーティション・レベルの *LOB_storage_clause* は、パーティション・レベルおよび表レベルの *LOB_storage_clauses* を上書きします。

参照 :

- サイズが GB になる LOB を作成する場合のガイドラインを含む LOB の詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』を参照してください。
- 「LOB 列の例 :」（16-63 ページ）

LOB_item

表の表領域および記憶特性とは異なる表領域および記憶特性を明示的に定義する場合に、その LOB 列名または LOB オブジェクト属性を指定します。作成する各 *LOB_item* に、システム管理された索引が自動的に作成されます。

SECUREFILE | BASICFILE

この句を使用して、LOB 記憶域のタイプに、高パフォーマンスの LOB（SecureFile）または従来の LOB（BasicFile）を指定します。

参照 : SecureFile LOB の詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』を参照してください。

注意： LOB の記憶域の種類を変換することはできません。かわりに、オンライン再定義またはパーティション交換を使用して、SecureFile または BasicFile を移行する必要があります。

LOB_segname
LOB データ・セグメントの名前を指定します。*LOB_item* が複数指定されている場合は、*LOB_segname* を使用できません。

LOB_storage_parameters
LOB_storage_parameters 句を使用すると、様々な LOB 記憶域の要素を指定できます。

TABLESPACE 句 この句を使用すると、LOB データが格納される表領域を指定します。

storage_clause *storage_clause* を使用すると、LOB セグメント記憶域の様々な側面を指定できます。LOB 記憶域に関して特に重要なのは、*storage_clause* の MAXSIZE 句です。これは、*LOB_parameters* の *LOB_retention_clause* と組み合せて使用できます。詳細は、8-43 ページの「storage_clause」を参照してください。

LOB_parameters
LOB 記憶域に SecureFile を使用する場合、いくつかの *LOB_parameters* は不要になります。PCTVERSION および FREEPOOLS は、BasicFile LOB 記憶域を使用する場合にのみ有効かつ有用です。

ENABLE STORAGE IN ROW 行の記憶域を使用可能にした場合、LOB 値の長さが、約 4000 バイトからシステム制御情報分を引いた長さより小さければ、LOB 値がインラインに格納されます。これはデフォルトです。

行の記憶域を使用可能にする場合の制限事項： *index_org_table_clause* で OVERFLOW セグメントを指定しないかぎり、索引構成表に対して、このパラメータを指定できません。

DISABLE STORAGE IN ROW 行の記憶域を使用禁止にした場合、LOB 値の長さに関係なく、LOB 値はアウトライン（行の外側）に格納されます。

LOB 値が格納されている場所にかかわらず、LOB ロケータは、常にインラインに格納されます。STORAGE IN ROW の値は、一度設定すると、表を移動しないかぎり、変更できません。詳細は、12-70 ページの「ALTER TABLE」の「move_table_clause」を参照してください。

CHUNK *integer* LOB の操作用に割り当てるバイト数を指定します。*integer* にデータベースのブロック・サイズの倍数を指定しなかった場合、自動的に次に大きい倍数（バイト単位）に切り上げられます。たとえば、データベースのブロック・サイズが 2048 バイトのときに *integer* に 2050 を指定すると、4096 バイト（2 ブロック）が割り当てられます。最大値は 32768（32KB）で、これが Oracle Database のブロック・サイズとして使用できる最も大きな値です。デフォルトの CHUNK サイズは、Oracle での 1 データベース・ブロックです。

CHUNK の値は、NEXT の値（デフォルト値または *storage_clause* で指定された値）以下である必要があります。CHUNK の値が NEXT の値を超えると、エラーが戻ります。CHUNK の値は、一度設定すると変更できません。

PCTVERSION *integer* LOB の記憶域全体のうち、旧バージョンの LOB の保持に使用される割合（パーセント）の最大値を指定します。デフォルト値は 10 です。これは、LOB の記憶域全体の 10% が使用されるまで以前のバージョンの LOB データが上書きされないことを意味します。

データベースが手動 UNDO モードと自動 UNDO モードのどちらで稼働されていても、PCTVERSION パラメータを指定できます。PCTVERSION は、手動 UNDO モードのデフォルト値です。RETENTION は、自動 UNDO モードのデフォルト値です。PCTVERSION と RETENTION の両方は指定できません。

この句は、SECUREFILE を指定した場合は無効です。SECUREFILE と PCTVERSION の両方を指定した場合、PCTVERSION パラメータは特に警告もなく無視されます。

LOB_retention_clause この句を使用すると、LOB セグメントを保持する用途に、フラッシュバック、読取り一貫性、その両方、またはどちらでもないを指定できます。

データベースが自動 UNDO モードで稼働している場合にのみ、RETENTION パラメータを使用できます。データベースに保持されるコミット済の UNDO データの量は、UNDO_RETENTION 初期化パラメータの値を使用して決定されます。自動 UNDO モードでは、PCTVERSION を指定しないかぎり、RETENTION がデフォルト値となります。PCTVERSION と RETENTION の両方は指定できません。

SecureFile を使用している場合にのみ、RETENTION の後にオプションの設定を指定できます。*LOB_storage_clause* の SECUREFILE パラメータは、データベースが SecureFile を使用して記憶域を動的に管理することを示します。データベースの UNDO モードなどの要因が考慮されます。

- MAX: LOB セグメントが MAXSIZE に達するまで UNDO を保持するように指定します。MAX を指定する場合は、*storage_clause* で MAXSIZE 句も指定する必要があります。
- MIN: データベースがフラッシュバック・モードで、特定の LOB セグメントの UNDO 保存**期間**を *n* 秒に制限する場合に指定します。
- AUTO: 読取り一貫性に必要十分な UNDO を保持します。これはデフォルトです。
- NONE: 読取り一貫性またはフラッシュバックのどちらにも UNDO が必要ない場合に指定します。

参照 :

- SecureFile を使用して簡略化された LOB 記憶域の詳細は、16-38 ページの「CREATE TABLE」の LOB_storage_parameters 句を参照してください。
- SecureFile の使用の詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』を参照してください。
- データベースをフラッシュバック・モードにする方法については、10-39 ページの「ALTER DATABASE」の「flashback_mode_clause」を参照してください。
- 「UNDO 表領域の作成例 :」（16-82 ページ）

FREEPOOLS *integer* LOB セグメントに対する空きリストのグループ数を指定します。通常、*integer* は、Oracle Real Application Clusters 環境のインスタンス数です。シングル・インスタンス・データベースの場合、この値は 1 になります。

データベースが自動 UNDO モードで稼働している場合にのみ、このパラメータを指定できます。自動 UNDO モードでは、*storage_clause* で FREELIST GROUPS パラメータを指定しないかぎり、FREEPOOLS がデフォルト値になります。FREEPOOLS と FREELIST GROUPS のどちらも指定しない場合、データベースが自動 UNDO 管理モードで稼働している場合は FREEPOOLS 1 のデフォルト値が使用され、手動 UNDO 管理モードで稼働している場合は FREELIST GROUPS 1 のデフォルト値が使用されます。

この句は、SECUREFILE を指定した場合は無効です。SECUREFILE と FREEPOOLS の両方を指定した場合、FREEPOOLS パラメータは特に警告もなく無視されます。

FREEPOOLS の制限事項 : *storage_clause* では、FREEPOOLS および FREELIST GROUPS パラメータを指定できません。

LOB_deduplicate_clause この句は、SecureFile LOB に対してのみ有効です。*LOB_deduplicate_clause* を使用すると、重複する LOB データを除外する、LOB の重複の除外を有効または無効にできます。

DEDUPLICATE キーワードは、LOB の重複コピーを除外するようデータベースに対して指定します。セキュアなハッシュ索引を使用して重複を検出すると、同じ内容を持つ LOB は単一のコピーに結合され、消費される記憶域が削減されて記憶域の管理が簡素化されます。

この句を指定しない場合、デフォルトでは、LOB の重複除外は無効です。

この句は、LOB セグメント全体に対して、LOB の重複除外を実装します。個々の LOB に対して重複除外を有効または無効にするには、DBMS_LOB.SETOPTIONS プロシージャを使用します。

> **参照：** LOB の重複除外の詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』を参照してください。DBMS_LOB パッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

LOB_compression_clause この句は、BasicFile LOB ではなく、SecureFile LOB に対してのみ有効です。*LOB_compression_clause* を使用すると、サーバー側の LOB 圧縮を有効または無効にすることをデータベースに指定できます。サーバー側の圧縮された LOB セグメントで、ランダムな読取り / 書込みアクセスが可能です。LOB 圧縮は、表の圧縮または索引の圧縮からは独立しています。この句を指定しない場合、NOCOMPRESS がデフォルトになります。

MEDIUM または HIGH を指定して、圧縮の程度を変更できます。圧縮の程度を HIGH にすると、待機時間は MEDIUM よりも長くなりますが、圧縮率は高くなります。このオプションのパラメータを指定しない場合、MEDIUM がデフォルトになります。

この句は、LOB セグメント全体のサーバー側の LOB 圧縮を実装します。個々の LOB の圧縮を有効または無効にするには、DBMS_LOB.SETOPTIONS プロシージャを使用します。

> **参照：** サーバー側の LOB 記憶域の詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』を参照してください。提供されているパッケージ UTL_COMPRESS を使用したクライアント側の LOB 圧縮および DBMS_LOB パッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

ENCRYPT | DECRYPT これらの句は、LOB 記憶域に SecureFile を使用している LOB に対してのみ有効です。ENCRYPT を指定すると、列内のすべての LOB を暗号化できます。DECRYPT を指定すると、LOB をクリアテキストで保持できます。この句を指定しない場合、DECRYPT がデフォルトになります。

この句の概要は、16-26 ページの「encryption_spec」を参照してください。LOB 列に適用すると *encryption_spec* は個々の LOB 列固有になるため、他の LOB 列や他の非 LOB 列とは、暗号化アルゴリズムが異なる場合があります。*column_definition* の一部として *encryption_clause* を使用すると、LOB 列全体を暗号化できます。*table_partition_description* で *LOB_storage_clause* の一部として *encryption_clause* を使用すると、LOB パーティションを暗号化できます。

LOB の *encryption_spec* の制限事項： LOB 暗号化に対しては、*encryption_spec* の SALT または NO SALT 句を指定できません。

> **参照：** LOB 暗号化の詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』を参照してください。DBMS_LOB パッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

CACHE | NOCACHE | CACHE READS この句は、LOB 記憶域だけでなく、セグメント記憶域全般に関連します。この句の詳細は、16-53 ページの「CACHE | NOCACHE | CACHE READS」を参照してください。

LOB_partition_storage

LOB_partition_storage 句を使用すると、各パーティションに、別の *LOB_storage_clause* または *varray_col_properties* を指定できます。パーティションは、その位置の順に指定してください。パーティションの順番を確認するには、USER_IND_PARTITIONS ビューの PARTITION_NAME および PARTITION_POSITION 列を問い合せます。

特定のパーティションに、*LOB_storage_clause* または *varray_col_properties* 句を指定しなかった場合、表レベルで LOB 項目に指定された記憶特性が設定されます。表のレベルでも LOB 項目に記憶特性を指定しなかった場合、LOB データ・パーティションは、対応する表パーティションと同じ表領域に格納されます。

LOB_storage_clause の *LOB_parameters* では、*encryption_spec* を指定できません。パーティションおよびサブパーティションに対する暗号化アルゴリズムの指定は無効であるためです。

varray_col_properties

varray_col_properties を使用すると、VARRAY 型のデータが格納される LOB に対して、別々の記憶特性を指定できます。*varray_item* がマルチレベル・コレクションの場合、*varray_item* 内にネストされたすべてのコレクション項目は、常に *varray_item* と同じ LOB に格納されます。

- 非パーティション表の場合（パーティション句なしで *physical_properties* 句に指定した場合）、この句は、VARRAY の LOB データ・セグメントの記憶域属性を指定します。
- 表レベルで指定されたパーティション表の場合（パーティション句とともに *physical_properties* 句で指定した場合）、この句は、各パーティション（またはサブパーティション）に対応付けられた VARRAY の LOB データ・セグメントに対するデフォルト記憶域属性を指定します。
- パーティション表の各パーティションの場合（*table_partition_description* の一部として指定した場合）、この句は、そのパーティションの VARRAY の LOB データ・セグメントの記憶域属性、またはこのパーティションのサブパーティションにある VARRAY の LOB データ・セグメントのデフォルト記憶域属性を指定します。パーティション・レベルの *varray_col_properties* は、表レベルの *varray_col_properties* を上書きします。
- パーティション表の各サブパーティションの場合（*subpartition_by_hash* または *subpartition_by_list* の一部として指定した場合）、この句は、このサブパーティションの VARRAY データ・セグメントの記憶域属性を指定します。サブパーティション・レベルの *varray_col_properties* は、パーティション・レベルおよび表レベルの *varray_col_properties* を上書きします。

STORE AS [SECUREFILE | BASICFILE] LOB 句　STORE AS LOB を指定したときに実行される処理は、次のとおりです。

- VARRAY の最大サイズが約 4000 バイト未満で、行の記憶域を使用禁止にしていない場合、VARRAY はインライン LOB に格納されます。
- VARRAY の最大サイズが約 4000 バイトを超える場合または行の記憶域を使用禁止にしている場合、VARRAY はアウトライン LOB として格納されます。

STORE AS LOB を指定しなかった場合、記憶域は、VARRAY 列の実際のサイズではなく、VARRAY の最大サイズに基づいて決定されます。VARRAY の最大サイズは、要素数×要素サイズ + システム制御情報分の容量です。この句を指定しない場合、次のようになります。

- VARRAY の最大サイズが約 4000 バイト未満の場合、VARRAY は LOB としてではなくインライン・データとして格納されます。
- 最大サイズが約 4000 バイトを超える場合、VARRAY は常に LOB として格納されます。
  - 実際のサイズが約 4000 バイト未満の場合、VARRAY はインライン LOB として格納されます。

– 実際のサイズが約 4000 バイトを超える場合、VARRAY はアウトライン LOB として格納されます。これはその他の LOB 列でも同様です。

substitutable_column_clause *substitutable_column_clause* の動作は、16-36 ページの「object_type_col_properties」の場合と同じです。

> **参照：** 「置換可能な表および列のサンプル :」（16-61 ページ）

nested_table_col_properties

nested_table_col_properties を使用すると、ネストした表に対して別々の記憶特性を指定し、そのネストした表を索引構成表として定義できるようになります。この句で特に明示的に指定しないかぎり、記憶表は次のとおり作成されます。

- 非パーティション表の場合、記憶表は親表と同じスキーマおよび同じ表領域内に作成されます。
- パーティション表の場合、記憶表はスキーマのデフォルトの表領域内に作成されます。デフォルトでは、ネストした表はパーティション実表でパーティション化されます。
- どちらの場合も、記憶表ではデフォルトの記憶特性が使用され、この表の作成の基になった列のネストした表の値が格納されます。

ネストした表の型を持つ列または列属性付きで表を作成する場合は、この句を挿入する必要があります。*nested_table_col_properties* 句内で、親表に対する場合と同じ働きをする句については、ここでは説明しません。

nested_item 型がネストした表である列、またはその表のオブジェクト型の最上位の属性の名前を指定します。

COLUMN_VALUE ネストした表がマルチレベル・コレクションの場合、内部のネストした表または VARRAY には名前が割り当てられていない場合があります。この場合、*nested_item* 名のかわりに COLUMN_VALUE を指定します。

> **参照：** *nested_item* および COLUMN_VALUE の使用例は、16-62 ページの「マルチレベル・コレクションの例 :」を参照してください。

storage_table *nested_item* の行を含む表の名前を指定します。

storage_table に対して問合せや DML 文を直接実行することはできませんが、その記憶特性は、ALTER TABLE 文で名前を指定することによって変更できます。

記憶表の制限事項： ネストした表の記憶表はパーティション化できません。

> **参照：** ネストした表の列に対する記憶特性の変更方法については、12-2 ページの「ALTER TABLE」を参照してください。

RETURN AS 問合せの結果として何を戻り値とするかを指定します。

- VALUE は、ネストした表自体のコピーを戻します。
- LOCATOR は、ネストした表のコピーに対するコレクション・ロケータを戻します。

  ロケータの有効範囲は 1 つのセッションであり、複数のセッションにわたって使用できません。LOB ロケータとは異なり、コレクション・ロケータはコレクション・インスタンスの変更に使用できません。

segment_attributes_clause または *LOB_storage_clause* を指定しない場合、ネストした表はヒープ構成され、デフォルトの記憶特性で作成されます。

ネストした表の列のプロパティの制限事項： ネストした表の列のプロパティには、次の制限事項があります。

- この句は、一時表には指定できません。
- *oid_clause* は指定できません。
- 作成時、*object_properties* を使用して、*out_of_line_ref_constraint*、*inline_ref_constraint* またはネストした表の属性に対する外部キー制約を指定することはできません。ただし、ALTER TABLE を使用してネストした表を修正し、これらの制約を追加できます。

  参照：

  - ネストした表の列に対する記憶特性の変更方法については、12-2 ページの「ALTER TABLE」を参照してください。
  - 16-62 ページの「ネストした表の例 :」および 16-62 ページの「マルチレベル・コレクションの例 :」を参照してください。

XMLType_column_properties

XMLType_column_properties を指定すると、XMLTYPE 列に対する記憶域属性を指定できます。

XMLType_storage XMLType 列は、LOB 列、オブジェクト・リレーショナル列またはバイナリ XML 列に格納できます。

- OBJECT RELATIONAL を指定すると、オブジェクト・リレーショナル列に XMLType データを格納できます。データ・オブジェクトをリレーショナルに格納すると、リレーショナル列に索引を定義できるため、問合せのパフォーマンスが向上します。

  オブジェクト・リレーショナル形式での格納を指定する場合、*XMLSchema_spec* 句も指定する必要があります。

- CLOB を指定すると、CLOB 列に XMLType データを格納できます。CLOB 列にデータを格納すると、元の内容が保持されるため、検索時間が短縮されます。

  LOB 記憶域を定義する場合、LOB パラメータと *XMLSchema_spec* 句のいずれかを指定できますが、両方は指定できません。*XMLSchema_spec* 句を指定すると、特定のスキーマ・ベースの XML インスタンスに表や列を制限できます。

- BINARY XML を指定すると、縮小されたバイナリ XML 書式で XML データを格納できます。

  指定した LOB パラメータは、バイナリ XML エンコード値を格納するために作成された、基礎となる BLOB 列に適用されます。

CLOB とバイナリ XML 記憶域の両方に対して、データが SecureFile LOB で格納されるように指定できます。詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』を参照してください。

XMLSchema_spec この句を使用すると、単一の登録済 XMLSchema または複数のスキーマ（XMLSCHEMA 句で登録するか ELEMENT 句の一部として登録）の URL および XML 要素名を指定できます。BINARY XML 記憶域を指定した場合のみ複数のスキーマを使用できます。

XMLSchema の URL は省略可能ですが、要素は必ず指定します。XMLSchema の URL を指定する場合は、DBMS_XMLSCHEMA パッケージを使用して XMLSchema をあらかじめ登録しておく必要があります。

オプションの ALLOW | DISALLOW 句は、BINARY XML 記憶域を指定した場合にのみ有効です。

- ALLOW ANYSCHEMA を指定すると、スキーマ・ベースの任意のドキュメントを XMLType 列に格納できます。
- ALLOW NONSCHEMA を指定すると、スキーマ・ベース以外のドキュメントを XMLType 列に格納できます。

- DISALLOW NONSCHEMA を指定すると、スキーマ・ベース以外のドキュメントは XMLType 列に格納できません。

  参照：

  - *LOB_segname* および *LOB_parameters* 句の詳細は、12-42 ページの「LOB_storage_clause」を参照してください。
  - オブジェクト・リレーショナル表における XMLType 列の例は、16-65 ページの「XMLType 列の例 :」を参照してください。XMLSchema の作成例は、E-8 ページの「SQL 文での XML の使用方法」を参照してください。
  - XMLType 列と表および XMLSchema の作成の詳細は、『Oracle XML DB 開発者ガイド』を参照してください。
  - DBMS_XMLSCHEMA パッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

table_partitioning_clauses

table_partitioning_clauses を使用すると、パーティション表を作成できます。

一般的なパーティション化の注意事項： すべてのタイプのパーティション化には、次の注意事項があります。

- 指定できるパーティションとサブパーティションの合計は 1024K-1 です。
- パーティションが 1 つのみのパーティション表も作成できます。パーティションが 1 つの表は、非パーティション表とは異なります。たとえば、非パーティション表にはパーティションを追加できません。
- すべての表と LOB パーティションおよびすべての表と LOB サブパーティションに名前を指定できますが、必須ではありません。名前を省略すると、次のように名前が生成されます。
  - パーティション名を省略すると、SYS_Pn の形式で名前が生成されます。LOB データおよび LOB 索引パーティションに対するシステム生成名は、それぞれ SYS_LOB_Pn および SYS_IL_Pn の形式をとります。
  - *subpartition_template* でサブパーティション名を指定すると、そのテンプレートで作成される各サブパーティションに対して、パーティション名とテンプレートのサブパーティション名を連結して名前が生成されます。LOB サブパーティションの場合、生成される LOB サブパーティション名は、パーティション名とテンプレートの LOB セグメント名の連結です。どちらの場合も、連結の結果が 30 文字を超える場合は、エラーが戻されて文は失敗します。
  - 個々のサブパーティションを指定するときにサブパーティション名を指定せず、*subpartition_template* を指定していない場合、SYS_SUBPn という形式で名前が生成されます。LOB データおよび索引サブパーティションに対する、対応するシステム生成名は、それぞれ SYS_LOB_SUBPn および SYS_IL_SUBPn です。
- 表領域の記憶域は、CREATE TABLE 文で、表セグメントと LOB セグメントの両方に対して様々なレベルで指定できます。表領域数は、パーティション数またはサブパーティション数と同じである必要はありません。パーティション数またはサブパーティション数が表領域数より多い場合は、表領域名が繰り返し使用されます。

  データベースでは、表領域の記憶域を、次の順序で（優先度の高いものから順に）評価します。

  - 個々の表サブパーティションまたは LOB サブパーティション・レベルで指定された表領域の記憶域が、最も優先度が高くなります。次に、*subpartition_template* でパーティションまたは LOB に対して指定された記憶域です。
  - 個々の表パーティションまたは LOB パーティション・レベルで指定された表領域の記憶域。ここで指定された記憶域パラメータは、*subpartition_template* よりも優先します。

  - 表に対して指定された表領域の記憶域
  - ユーザーに対して指定されたデフォルトの表領域の記憶域
- デフォルトでは、ネストした表はパーティション実表でパーティション化されます。

一般的なパーティション化の制限事項： すべてのパーティション化には、次の制限事項があります。

- クラスタの一部である表は、パーティション化できません。
- LONG または LONG RAW 列を含む表は、パーティション化できません。

ブロック・サイズが異なる表領域のパーティション化されたデータベース・エンティティの記憶域には、制限事項があります。これらの制限事項については、『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。

参照： 「パーティション化の例」（16-65 ページ）

range_partitions

range_partitions 句を使用すると、列リストの値の範囲で表をパーティション化できます。索引構成表の場合、列リストは表の主キー列のサブセットである必要があります。

column

行がどのパーティションに属するかを判断するために使用される、列の順序リストを指定します。これらの列は、**パーティション化キー**です。仮想列をパーティション化キー列として指定できます。

パーティション化キー列の制限事項： 列リスト内の列には、ROWID、LONG、LOB、XMLType または TIMESTAMP WITH TIME ZONE 以外の組込みデータ型を指定できます。ただし、TIMESTAMP 型または TIMESTAMP WITH LOCAL TIME ZONE の列は、パーティション化キーで使用できます。

INTERVAL 句

この句を使用すると、表の**時間隔パーティション**を設定できます。時間隔パーティションは、数値範囲または日時期間に基づくパーティションです。表に挿入されたデータがすべてのレンジ・パーティションを超える場合に、指定されたレンジまたは期間のパーティションを自動的に作成することをデータベースに指定することによって、レンジ・パーティション化を拡張します。

- *expr* には、有効な数値または期間式を指定します。
- オプションの STORE IN 句を使用して、時間隔パーティション・データが格納される 1 つ以上の表領域を指定できます。
- また、*range_partitions* の PARTITION 句を使用して、1 つ以上のレンジ・パーティションを指定する必要があります。レンジ・パーティション・キー値によって、レンジ・パーティションの上限が決まります。これは**遷移ポイント**と呼ばれ、その遷移ポイントを超えるデータに対して時間隔パーティションが作成されます。

時間隔パーティション化の制限事項： INTERVAL 句には、次の制限事項があります。

- パーティション化キーは 1 つのみ指定でき、NUMBER 型または DATE 型である必要があります。
- この句は索引構成表ではサポートされていません。
- 時間隔パーティション表には、ドメイン索引は作成できません。
- 時間隔パーティションは、サブパーティション・レベルではサポートされていません。

- VALUES 句の場合 :
  - MAXVALUE は指定できません（無限）。指定すると、必要に応じてパーティションを自動追加するという目的に反するためです。
  - パーティション化キー列には NULL 値を指定できません。

参照 : 時間隔パーティションの詳細は、『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。

PARTITION *partition*

パーティション名を指定する際、*partition* の名前は、スキーマ・オブジェクトのネーミング規則および 2-98 ページの「スキーマ・オブジェクトのネーミング規則」にある該当部分の記述に従って指定する必要があります。*partition* を省略すると、16-44 ページの「一般的なパーティション化の注意事項 :」で説明されているように名前が生成されます。

range_values_clause

現行パーティションの上限（境界は含まない）を指定します。値リストは、*range_partitioning* 句の列リストに対応するリテラル値を含む順序リストです。値リスト内のリテラルのかわりに、キーワード MAXVALUE を使用できます。MAXVALUE には、常に他の値（NULL を含む）より高位にソートされる最大値を指定します。

パーティション境界の上限に MAXVALUE 以外の値を指定した場合、表に暗黙の整合性制約が課せられます。

注意 : 表が DATE 列でパーティション化されている場合および日付書式で年の最初の 2 桁の数字が指定されていない場合、年の「YYYY」4 文字書式マスクで TO_DATE ファンクションを使用する必要があります。この句では、「RRRR」書式マスクはサポートしていません。日付書式は、NLS_TERRITORY によって暗黙的に決定され、NLS_DATE_FORMAT によって明示的に決定されます。これらの初期化パラメータの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

参照 : パーティション・バウンドの詳細は、『Oracle Database 概要』を参照してください。また、16-65 ページの「レンジ・パーティション化の例 :」を参照してください。

table_partition_description

table_partition_description を使用すると、表の物理特性および記憶特性を定義できます。

segment_attributes_clause 句および *table_compression* 句の機能は、表全体の table_properties と同じです。

key_compression 句および OVERFLOW 句の機能は、index_org_table_clause と同じです。

LOB_storage_clause *LOB_storage_clause* を使用すると、このパーティションまたはこのパーティションの任意の**レンジまたは**リスト・サブパーティション内にある 1 つ以上の LOB 項目に対して LOB 記憶特性を指定できます。LOB 項目に *LOB_storage_clause* を指定しない場合、16-44 ページの「一般的なパーティション化の注意事項 :」で説明されているように、各 LOB データ・パーティションに対する名前が生成されます。

varray_col_properties *varray_col_properties* を使用すると、このパーティションまたはこのパーティションの任意の**レンジまたは**リスト・サブパーティション内にある 1 つ以上の VARRAY 項目に対して記憶特性を指定できます。

partitioning_storage_clause

partitioning_storage_clause を使用すると、ハッシュ・パーティションおよびレンジ、ハッシュおよびリスト・サブパーティションの記憶特性を指定できます。

***partitioning_storage_clause* の制限事項 :** この句には、次の制限事項があります。

- OVERFLOW 句は、索引構成パーティション表にのみ関連し、*individual_hash_partitions* 句でのみ有効です。レンジまたはハッシュ・パーティションまたは任意のタイプのサブパーティションには有効ではありません。
- *key_compression* は、索引構成表のパーティションにのみ指定できます。

hash_partitions

hash_partitions 句を使用すると、表がハッシュ方式でパーティション化されるように指定できます。列の値にパーティション化キーとして指定されたハッシュ・ファンクションを使用して、行がパーティションに割り当てられます。個々のハッシュ・パーティションを指定するか、または作成されるサブパーティションの数を指定できます。

column 行がどのパーティションに属するかを判断するために使用される、列の順序リストを指定します（パーティション化キー）。

individual_hash_partitions この句を使用すると、個々のパーティションを名前で指定できます。

個々のハッシュ・パーティションを指定する場合の制限事項 :
partitioning_storage_clause でサブパーティションに対して指定できる句は、TABLESPACE 句および表の圧縮のみです。

> **注意 :** 異なるキャラクタ・セットを使用してデータベースを使用しているか、使用する予定がある場合は、キャラクタ列を分割する際に注意してください。文字のソート順序は、すべてのキャラクタ・セットで同一ではありません。キャラクタ・セット・サポートの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

hash_partitions_by_quantity 個々のパーティションを定義するかわりに、ハッシュ・パーティションの数を指定します。この場合、SYS_Pn の形式でパーティション名が割り当てられます。STORE IN 句を使用すると、ハッシュ・パーティション・データが格納される 1 つ以上の表領域を指定できます。表領域の数とパーティションの数が同じである必要はありません。パーティション数が表領域数より多い場合は、表領域名が繰り返し使用されます。

ハッシュ・パーティション化の両方の方法でロード・バランシングを最適化するには、2 の累乗のパーティション数を指定します。個々のハッシュ・パーティションを指定する際は、*partitioning_storage_clause* に TABLESPACE と表の圧縮の両方を指定できます。ハッシュ・パーティションを数で指定する場合は、TABLESPACE のみを指定できます。ハッシュ・パーティションは、その他のすべての属性を表レベルのデフォルトから継承します。

table_compression 句の機能は、表の table_properties について説明されている機能と、ほぼ同じです。

key_compression 句および OVERFLOW 句の機能は、index_org_table_clause と同じです。

表レベルで指定された表領域の記憶域は、パーティション・レベルで指定された表領域の記憶域で上書きされ、パーティション・レベルで指定された表領域の記憶域は、サブパーティション・レベルで指定された表領域の記憶域で上書きされます。

individual_hash_partitions 句に含まれる *partitioning_storage_clause* の TABLESPACE 句は、作成される個々のパーティションのみについて、表領域の記憶域を決定します。*hash_partitions_by_quantity* 句では、STORE IN 句によって、表の作成時のパーティションの位置と、後から追加されるパーティションのデフォルトの格納場所が決定されます。

参照： ハッシュ・パーティション化については、『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。

ハッシュ・パーティション化の制限事項： ハッシュ・パーティション化には、次の制限事項があります。

- 指定できるパーティション化キー列は 16 以下です。
- 列リストには、ROWID または UROWID 疑似列を含めることはできません。
- 列リストには、ROWID、LONG または LOB 以外の組込みデータ型を指定できます。

list_partitions

list_partitions 句を使用すると、*column* のリテラル値のリストで表をパーティション化できます。リスト・パーティション化は、個々の行が固有のパーティションにマップする方法に関する制御に便利です。

list_values_clause 各パーティションの *list_values_clause* では、1 つ以上の値を割り当てる必要があります。複数のパーティションに同じ値（NULL を含む）を割り当てることはできません。リスト・パーティションは、順序付けされていません。

VALUES 句のパーティション値にリテラル NULL を指定した場合、後続の問合せで、そのパーティション内のデータにアクセスするには、WHERE 句で、比較条件ではなく IS NULL 条件を使用する必要があります。

DEFAULT キーワードを指定すると、行の挿入先となるパーティションが作成されます。この行は、別のパーティションにはマップされません。このため、DEFAULT を指定できるのは 1 つのパーティションのみです。そのパーティションに対して、その他の値を指定することはできません。また、デフォルト・パーティションは、パーティションの中で最後に定義する必要があります。DEFAULT は、レンジ・パーティションで MAXVALUE を使用する場合と同様に使用します。

各パーティションの値のリストを構成する文字列は、最大 4KB です。すべてのパーティションの値の総数を、64K-1 以下に指定します。

table_partition_description *table_partition_description* の副次句の動作は、16-48 ページの「table_partition_description」のレンジ・パーティションで説明した動作と同じです。

リスト・パーティション化の制限事項： リスト・パーティション化には、16-45 ページの「一般的なパーティション化の制限事項 :」に示されている制限事項があります。

reference_partitioning

この句を使用すると、参照によって表をパーティション化できます。参照によるパーティション化は、作成される表（**子表**）を既存のパーティション表（**親表**）への参照制約によって同一レベルでパーティション化する方法です。参照によって表をパーティション化すると、その後に親表で実行されるパーティションのメンテナンス操作は子表に自動的にカスケードします。そのため、パーティションのメンテナンス操作は、参照パーティション表で直接実行できません。

constraint パーティション化参照制約は、次の条件を満たしている必要があります。

- 作成される表で定義される参照整合性制約を指定する必要があります。これは、親表の主キーまたは一意制約を参照する必要があります。制約は、ENABLE VALIDATE NOT DEFERRABLE 状態である必要があります。これは、表の作成時に参照整合性制約を指定するときのデフォルトです。
- 制約で参照されるすべての外部キー列は、NOT NULL である必要があります。
- 制約を指定するときに、*references_clause* の ON DELETE SET NULL 句は指定できません。

- 制約で参照される親表は、既存のパーティション表である必要があります。時間隔パーティション以外の方法でパーティション化されている必要があります。
- 外部キーに仮想列を含めることはできません。
- 親表の参照される主キーまたは一意制約に仮想列を含めることはできません。

reference_partition_desc このオプションの句を使用すると、パーティション名を指定し、パーティションの物理特性および記憶特性を定義できます。
table_partition_description の副次句の動作は、16-48 ページの「table_partition_description」のレンジ・パーティションで説明した動作と同じです。

参照パーティション化の制限事項： 参照パーティション化には、次の制限事項があります。

- この句は、索引構成表、外部表またはドメイン索引記憶表に対して指定できません。
- 親表を参照でパーティション化することはできますが、制約を自己参照型にすることはできません。作成される表を自己参照に基づいてパーティション化することはできません。
- ROW MOVEMENT が親表に対して有効である場合、子表に対しても有効である必要があります。
- CREATE TABLE ...AS SELECT 文で参照パーティション化は指定できません。

> **参照：** 参照によるパーティション化の詳細は、『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。

composite_range_partitions

composite_range_partitions 句を使用すると、まず、表をレンジ・パーティション化し、次にそれらのパーティションをレンジ・サブパーティション化、ハッシュ・サブパーティション化またはリスト・サブパーティション化できます。

コンポジット・レンジ・パーティション化での INTERVAL 句のセマンティクスは、レンジ・パーティション化の場合と同じです。詳細は、16-45 ページの「INTERVAL 句」を参照してください。

subpartition_by_range、subpartition_by_hash または subpartition_by_list を指定すると、各コンポジット・レンジ・パーティションのサブパーティション化のタイプを指定できます。これらの句では、サブパーティション・テンプレートを指定できます。サブパーティション・テンプレートによって、この文の一部として作成されるサブパーティションまたは後で作成されるサブパーティションのデフォルトのサブパーティション特性が設定されます。

表のサブパーティション化のタイプ、およびオプションでサブパーティション・テンプレートを設定した後、1 つ以上のレンジ・パーティションを定義する必要があります。

- 非コンポジット・レンジ・パーティションと同じ要件を持つ range_values_clause を指定する必要があります。
- table_partition_description を使用すると、各パーティションの物理特性および記憶特性を定義できます。
- *range_partition_desc* で、*range_subpartition_desc*、*list_subpartition_desc* または *hash_subpartition_desc* を使用して、パーティションの個々のサブパーティションの特性を指定します。これらの句で指定する値は、これらのサブパーティションについて、*subpartition_template* で指定した値にかわるものです。
- ハッシュ・サブパーティション、リスト・サブパーティションまたは LOB サブパーティションに指定できる特性は、TABLESPACE および *table_compression* のみです。

コンポジット・レンジ・パーティション化の制限事項： サブパーティション化のタイプにかかわらず、コンポジット・レンジ・パーティションには次の制限事項があります。

- サブパーティション・レベルで指定できる物理属性は、TABLESPACE および表の圧縮のみです。

- コンポジット・パーティション化は、索引構成表に対して指定できません。そのため、コンポジット・パーティション表に対して、*table_partition_description* の OVERFLOW 句は無効です。

  参照： コンポジット・レンジ・パーティション化の例は、16-68 ページの「コンポジット・パーティション表の例：」を参照してください。コンポジット・リスト・パーティション化の例は、『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。

composite_list_partitions

composite_list_partitions 句を使用すると、まず、表をリスト・パーティション化し、次にそれらのパーティションをレンジ・サブパーティション化、ハッシュ・サブパーティション化またはリスト・サブパーティション化できます。

subpartition_by_range、subpartition_by_hash または subpartition_by_list を指定すると、各コンポジット・リスト・パーティションのサブパーティション化のタイプを指定できます。これらの句では、サブパーティション・テンプレートを指定できます。サブパーティション・テンプレートによって、この文の一部として作成されるサブパーティションおよび後で作成されるサブパーティションのデフォルトのサブパーティション特性が設定されます。

各コンポジット・パーティションのサブパーティション化のタイプを設定し、オプションでサブパーティション・テンプレートを定義した後、1 つ以上のリスト・パーティションを定義する必要があります。

- *list_partition_desc* で、非コンポジット・リスト・パーティションと同じ要件を持つ list_values_clause を指定する必要があります。
- table_partition_description を使用すると、各パーティションの物理特性および記憶特性を定義できます。
- *list_partition_desc* で、*range_subpartition_desc*、*list_subpartition_desc* または *hash_subpartition_desc* を使用して、パーティションの個々のサブパーティションの特性を指定します。これらの句で指定する値は、これらのサブパーティションについて、*subpartition_template* で指定した値にかわるものです。

コンポジット・リスト・パーティション化の制限事項： コンポジット・リスト・パーティション化には、16-49 ページの「コンポジット・レンジ・パーティション化の制限事項：」に示されているものと同じ制限事項があります。

subpartition_template *subpartition_template* は、レンジ・サブパーティション化、リスト・サブパーティション化およびハッシュ・サブパーティション化のオプション要素です。このテンプレートを使用することで、表の各パーティションにデフォルトのサブパーティションを定義できます。明示的にサブパーティションを定義していないパーティションには、このデフォルト・サブパーティション特性が作成されます。この句は、対称型パーティションの作成時に有効です。パーティション・レベルでサブパーティションを明示的に定義すると（*range_subpartition_desc*、*list_subpartition_desc* または *hash_subpartition_desc* で指定）、この句を上書きできます。

テンプレートを使用してサブパーティションを定義する場合、各サブパーティションの名前を指定する必要があります。また、サブパーティション・テンプレートの *partitioning_storage_clause* の *LOB_partitioning_clause* を指定する場合、*LOB_segname* を指定する必要があります。

注意： サブパーティションのテンプレートに対して表領域の記憶域を指定しても、*table* のパーティションに対して明示的に指定した表領域の記憶域は上書きされません。サブパーティションに対して表領域の記憶域を指定するには、次のいずれかの操作を実行します。

- パーティション・レベルでの表領域の記憶域を省略し、サブパーティションのテンプレートに対して表領域の記憶域を指定します。
- 固有の表領域の記憶域を持つサブパーティションを個別に定義します。

サブパーティションのテンプレートの制限事項： サブパーティションのテンプレートには、次の制限事項があります。

- 1 つの LOB サブパーティションに対して TABLESPACE を指定した場合、次にその LOB 列のすべての LOB サブパーティションに対して TABLESPACE を指定する必要があります。複数の LOB サブパーティションに対して同じ表領域を指定できます。
- *subpartition_template* 内で、または個々のサブパーティションを定義するときに、*partitioning_storage_clause* を使用してリスト・サブパーティションに対して異なる LOB 記憶域を指定した場合、LOB と VARRAY 列の両方に *LOB_segname* を指定する必要があります。

subpartition_by_range

subpartition_by_range 句を使用すると、表の各パーティションをレンジ・サブパーティション化できます。列リストのサブパーティション化はパーティション化キーには関連しませんが、同じ制限事項が適用されます（16-45 ページの「column」を参照）。

subpartition_template を使用して、デフォルトのサブパーティション特性値を指定できます。16-50 ページの「subpartition_template」を参照してください。データベースは、特性を明示的に指定していないこのパーティションの任意のサブパーティションに対して、これらの値を使用します。

range_partition_desc または *list_partition_desc* の *range_subpartition_desc* を使用して、各パーティションのレンジ・サブパーティションを個別に定義することもできます。*subpartition_template* と *range_subpartition_desc* の両方を指定しない場合、単一の MAXVALUE サブパーティションが作成されます。

subpartition_by_hash

subpartition_by_hash 句を使用すると、表の各パーティションをハッシュ・サブパーティション化できます。列リストのサブパーティション化はパーティション化キーには関連しませんが、同じ制限事項が適用されます（16-47 ページの「column」を参照）。

subpartition_template 句または SUBPARTITIONS *integer* 句を使用すると、サブパーティションを定義できます。16-50 ページの「subpartition_template」を参照してください。どちらの場合も、ロード・バランシングを最適化するには、2 の累乗のパーティション数を指定する必要があります。

SUBPARTITIONS *integer* を指定する場合、表の各パーティションにおけるデフォルトのサブパーティション数を設定します。また、サブパーティションが格納される 1 つ以上の表領域を指定することもできます。デフォルト値は 1 です。この句と *subpartition_template* の両方を指定しない場合、1 つのハッシュ・サブパーティションを持つパーティションが作成されます。

subpartition_by_list

subpartition_by_list 句を使用すると、表の各パーティションを列のリテラル値に基づいてサブパーティション化できます。リスト・サブパーティション化キー列を 1 つのみ指定できます。

subpartition_template を使用して、デフォルトのサブパーティション特性値を指定できます。16-50 ページの「subpartition_template」を参照してください。データベースは、特性を明示的に指定していないこのパーティションの任意のサブパーティションに対して、これらの値を使用します。

range_partition_desc または *list_partition_desc* の *list_subpartition_desc* を使用して、各パーティションのリスト・サブパーティションを個別に定義することもできます。*subpartition_template* と *list_subpartition_desc* の両方を指定しない場合、単一の DEFAULT サブパーティションが作成されます。

リスト・サブパーティション化の制限事項： リスト・サブパーティション化には、16-49 ページの「コンポジット・レンジ・パーティション化の制限事項：」に示されているものと同じ制限事項があります。

コンポジット・パーティションの注意事項： コンポジット・パーティションには、次の注意事項があります。

- すべてのサブパーティションについて、*range_subpartition_spec*、*list_subpartition_spec*、*individual_hash_subparts* または *hash_subparts_by_quantity* を使用して、個々のサブパーティション名およびオプションでその他の特性を指定できます。
- または、ハッシュ・サブパーティションおよびリスト・サブパーティションの場合：
  - サブパーティションの数を指定できます。また、サブパーティションが格納される 1 つ以上の表領域を指定することもできます。この場合、SYS_SUBPn という形式でサブパーティション名が割り当てられます。
  - サブパーティションの定義を省略すると、サブパーティションのテンプレートが作成されている場合は、そのテンプレートに基づいてサブパーティションが作成されます。サブパーティションのテンプレートが作成されていない場合、1 つのハッシュ・サブパーティションまたは 1 つのデフォルト・リスト・サブパーティションが作成されます。
- すべてのタイプのサブパーティションについて、サブパーティションの定義全体を指定しない場合、次のようにサブパーティション名が割り当てられます。
  - サブパーティションのテンプレートを指定し、またパーティション名を指定した場合、「*partition_name* アンダースコア（_）*subpartition_name*」（たとえば、P1_SUB1）という形式でサブパーティション名が生成されます。
  - サブパーティションのテンプレートを指定していない場合、またはサブパーティションのテンプレートは指定したがパーティション名を指定しなかった場合、SYS_SUBP*n* という形式でサブパーティション名が生成されます。

system_partitioning

この句を使用すると、システム・パーティションを作成できます。システム・パーティション化には、パーティション化キー列もは必要ありません。またシステム・パーティションには、レンジまたはリスト境界またはハッシュ・アルゴリズムはありません。システム・パーティションは、パーティション化された実表を持つネストした表やドメイン索引記憶表などの依存表を同一レベルでパーティション化する方法です。

- PARTITION BY SYSTEM のみを指定すると、SYS_P*n* という形式のシステム生成の名前で 1 つのパーティションが作成されます。
- PARTITION BY SYSTEM PARTITIONS *integer* を指定すると、*integer* で指定した数（1 から 1024K-1 の範囲）のパーティションが作成されます。
- パーティションの定義は参照パーティションと同じ構文であるため、*reference_partition_desc* を共有します。*reference_partition_desc* 構文で、その他のパーティション属性を指定できます。ただし、*table_partition_description* で、OVERFLOW 句は指定できません。

システム・パーティション化の制限事項： システム・パーティション化には、次の制限事項があります。

- 索引構成表またはクラスタの一部である表は、システム・パーティション化できません。
- コンポジット・パーティション化は、システム・パーティション化ではサポートされていません。
- システム・パーティションは分割できません。
- CREATE TABLE ...AS SELECT 文でシステム・パーティション化は指定できません。
- INSERT INTO ... AS *subquery* 文を使用してシステム・パーティション表にデータを挿入するには、拡張パーティション構文を使用して、副問合せによって戻された値が挿入されるパーティションを指定する必要があります。

> **参照：** システム・パーティション化の使用の詳細は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。また、2-106 ページの「パーティション表と索引の参照」を参照してください。

CACHE | NOCACHE | CACHE READS

CACHE 句を使用すると、バッファ・キャッシュ内でのブロックの格納方法を指定できます。次に、CACHE または NOCACHE も指定しない場合の例を示します。

- CREATE TABLE 文では、NOCACHE がデフォルトです。
- ALTER TABLE 文では、既存の値は変更されません。

CACHE この句は、アクセス頻度の高いデータについて、全表スキャンの実行時に、この表のために取り出されたブロックを、バッファ・キャッシュ内の最低使用頻度（LRU）リストの最高使用頻度側に置く場合に指定します。この属性は、小規模な参照表で有効です。

> **参照：** データベースによる最低使用頻度（LRU）リストの保持の詳細は、『Oracle Database 概要』を参照してください。

LOB_storage_clause のパラメータとして CACHE を使用すると、より高速なアクセスのために、LOB データ値がバッファ・キャッシュに配置されます。このパラメータは、*logging_clause* と組み合せて評価されます。この句を指定しない場合、BasicFile LOB と SecureFile LOB の両方のデフォルト値は、NOCACHE LOGGING です。

CACHE の制限事項： CACHE は、索引構成表に対して指定できません。ただし、索引構成表は暗黙的に CACHE 動作を提供します。

NOCACHE アクセス頻度の低いデータについて、NOCACHE を指定すると、全表スキャンの実行時にこの表用に取り出されたブロックは、バッファ・キャッシュ内の LRU リストの最低使用頻度側に置かれます。NOCACHE は、LOB 記憶域のデフォルトです。

LOB_storage_clause のパラメータとして NOCACHE を使用すると、LOB 値はバッファ・キャッシュに入りません NOCACHE は、LOB 記憶域のデフォルトです。

NOCACHE の制限事項： NOCACHE は、索引構成表に対して指定できません。

CACHE READS CACHE READS は LOB 記憶域にのみ適用されます。LOB 値が書込み操作中ではなく読取り操作中にバッファ・キャッシュに入れられることを指定します。

logging_clause この句を使用すると、データ・ブロックの記憶域を記録するかどうかを指定できます。

> **参照：** *logging_clause* を *LOB_parameters* の一部として指定した場合については、8-34 ページの「logging_clause」を参照してください。

parallel_clause

parallel_clause を使用すると、表をパラレル作成した後、問合せおよび DML の INSERT、UPDATE、DELETE および MERGE に対するデフォルトの並列度を設定できます。

注意： *parallel_clause* 構文は、以前のリリースの構文にかわるものです。以前のリリースの構文は下位互換用にサポートされていますが、動作がわずかに異なることがあります。

NOPARALLEL NOPARALLEL を指定すると、シリアル実行が行われます。これはデフォルトです。

PARALLEL PARALLEL を指定すると、並列度を選択できます。並列度は、すべての関係するインスタンスで使用可能な CPU の数に、初期化パラメータ PARALLEL_THREADS_PER_CPU の値を掛けたものです。

PARALLEL *integer* *integer* には、パラレル操作で使用するパラレル・スレッド数である**並列度**を指定します。各パラレル・スレッドは、1、2 個のパラレル実行サーバーを使用します。通常、最適な並列度が計算されるため、*integer* に値を指定する必要はありません。

> **参照：** この句の詳細は、8-37 ページの「parallel_clause」を参照してください。

NOROWDEPENDENCIES | ROWDEPENDENCIES

表が**行レベル依存の追跡**を使用するかどうかを指定できます。この機能を使用すると、表の各行は、行を変更した最後のトランザクションのコミット時刻以降を示すシステム変更番号（SCN）を持つことになります。この設定は、表の作成後に変更できません。

ROWDEPENDENCIES ROWDEPENDENCIES を指定すると、行レベル依存の追跡を有効にできます。この設定は、主にレプリケーション環境でパラレル伝播を許可する場合に便利です。各行につきサイズが 6 バイト増えます。

NOROWDEPENDENCIES NOROWDEPENDENCIES を指定すると、行レベル依存の追跡機能を無効にできます。これはデフォルトです。

> **参照：** レプリケーション環境での行レベル依存の追跡の使用については、『Oracle Database アドバンスト・レプリケーション』を参照してください。

enable_disable_clause

enable_disable_clause を使用すると、制約が適用されるかどうかを指定できます。デフォルトでは、制約は ENABLE VALIDATE 状態で作成されます。

制約を使用可能および使用禁止にする場合の制限事項： 制約を使用可能および使用禁止にする処理には、次の制限事項があります。

- 整合性制約を使用可能または使用禁止にする場合、この文または前の文に制約を定義する必要があります。
- 参照された一意キー制約または主キー制約が使用可能でない場合は、外部キー制約を使用可能にできません。
- *using_index_clause* の *index_properties* 句で、INDEXTYPE IS ... 句は制約の定義において有効ではありません。

> **参照：** 制約の詳細は、8-4 ページの「constraint」を参照してください。また、16-62 ページの「ENABLE VALIDATE の例 :」および 16-62 ページの「DISABLE の例 :」も参照してください。

ENABLE 句　この句を使用すると、表のデータに制約を適用できます。この句の詳細は、8-14 ページの制約に関する項目の「ENABLE 句」を参照してください。

DISABLE 句　この句を使用すると、整合性制約を使用禁止にできます。この句の詳細は、8-15 ページの制約に関する項目の「DISABLE 句」を参照してください。

UNIQUE　UNIQUE 句を使用すると、指定された列または列の組合せに定義された一意制約を使用可能または使用禁止にできます。

PRIMARY KEY　PRIMARY KEY 句を使用すると、表に対して定義された主キー制約を使用可能または使用禁止にできます。

CONSTRAINT　CONSTRAINT 句を使用すると、*constraint* に指定する整合性制約を使用可能または使用禁止にできます。

KEEP | DROP INDEX　この句を使用すると、一意キー制約または主キー制約を適用するために使用される索引を保持するかまたは削除するかを指定できます。

索引の保持と削除の制限事項：　一意キー制約または主キー制約が使用禁止の場合にのみ、この句を指定できます。

using_index_clause　*using_index_clause* を使用すると、一意キー制約または主キー制約を適用するために使用される索引を指定または作成することができます。この句の詳細は、8-15 ページの制約に関する項目の「using_index_clause」を参照してください。

参照：

- index_attributes、global_partitioned_index および local_partitioned_index 句の詳細、および索引に関連する NOSORT と *logging_clause* の詳細は、14-50 ページの「CREATE INDEX」を参照してください。
- *using_index_clause*、主キー制約および一意制約の詳細は、8-4 ページの「constraint」を参照してください。
- 索引を使用した制約の適用例は、8-23 ページの「明示的な索引の制御例」を参照してください。

CASCADE　CASCADE を指定すると、指定した整合性制約に依存する整合性制約を使用禁止にできます。参照整合性制約を構成する主キーまたは一意キーを使用禁止にする場合、この句を指定します。

CASCADE の制限事項：　DISABLE を指定した場合のみ、CASCADE を指定できます。

row_movement_clause

row_movement_clause を使用すると、表の行が移動されるかどうかを指定できます。表の圧縮時や、パーティション・データの更新操作時などに、行を移動できます。

注意：　データ・アクセスで静的な ROWID が必要な場合は、行移動を有効にしないでください。通常の表（ヒープ構成表）の場合は、行を移動すると、その行の ROWID が変更されます。索引構成表での行の移動の場合は、論理 ROWID の物理推測コンポーネントは不正確になりますが、論理 ROWID は有効のままです。

- ENABLE を指定すると、行の移動を許可できます。このとき、ROWID は変更されます。
- DISABLE を指定すると、行の移動を禁止できます。このとき、ROWID は変更されません。

この句を省略すると、行移動は使用禁止になります。

行の移動の制限事項： この句は、非パーティション索引構成表に対して指定できません。

flashback_archive_clause

この句を指定するには、指定されたフラッシュバック・データ・アーカイブに対する FLASHBACK ARCHIVE オブジェクト権限が必要です。この句を使用すると、表の履歴追跡を有効または無効にできます。

- 表の追跡を有効にするには、FLASHBACK ARCHIVE を指定します。*flashback_archive* を指定すると、この表に特定のフラッシュバック・データ・アーカイブを指定できます。指定するフラッシュバック・データ・アーカイブは、すでに存在している必要があります。

  flashback_archive を省略すると、システムに対して指定されているデフォルトのフラッシュバック・データ・アーカイブが使用されます。システムにデフォルトのフラッシュバック・データ・アーカイブが指定されていない場合は、*flashback_archive* を指定する必要があります。

- 表の追跡を無効にするには、NO FLASHBACK ARCHIVE を指定します。これはデフォルトです。

***flashback_archive_clause* の制限事項：** フラッシュバック・データ・アーカイブには、次の制限事項があります。

- この句は、ネストした表、クラスタ化表、一時表、リモート表または外部表に対しては指定できません。
- この句を指定する表には、LONG 列またはネストした表の列を含めることはできません。
- この句を指定した後でエクスポート・ユーティリティ、インポート・ユーティリティまたはトランスポータブル表領域機能を使用して表を別のデータベースにコピーすると、コピー先の表では追跡は有効にならず、元の表のアーカイブ・データは使用できません。

参照：

- フラッシュバック・データ・アーカイブの使用方法の概要は、『Oracle Database アドバンスト・アプリケーション開発者ガイド』を参照してください。
- フラッシュバック・データ・アーカイブの割当て制限属性と保存属性の変更およびフラッシュバック・データ・アーカイブにおける表領域の記憶域の追加と変更の詳細は、10-60 ページの「ALTER FLASHBACK ARCHIVE」を参照してください。

AS *subquery*

副問合せを指定して表の内容を定義します。表の作成時に、副問合せの結果として戻された行を表の中に挿入します。

オブジェクト表の場合、*subquery* には表の型に対応する 1 つの式、または表の型の最上位属性の数のどちらかを設定できます。詳細は、19-4 ページの「SELECT」を参照してください。

subquery が、既存のマテリアライズド・ビューと部分的または完全に同じビューを戻す場合、*subquery* に指定された 1 つ以上の表のかわりにマテリアライズド・ビューがクエリー・リライトに使用されることがあります。

参照： マテリアライズド・ビューおよびクエリー・リライトの詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

データ型およびデータ長は、副問合せから導出されます。整合性制約や、その他の列および表の属性には次の規則が適用されます。

- 副問合せで、列を含む式ではなく列を選択した場合、選択した表の列に NOT NULL 制約が明示的に作成されていると、新しい表の対応する列にも NOT NULL 制約が自動的に定義されます。制約に違反する行がある場合、表は作成されずエラーが戻されます。
- 選択した表の列に暗黙的に作成された NOT NULL 制約（たとえば、主キー）は、新しい表には引き継がれません。
- 主キー、一意キー、外部キー、CHECK 制約、パーティション化条件、索引および列のデフォルト値は、新しい表に引き継がれません。
- 選択した表がパーティション化されている場合、新しい表を同じ方法でパーティション化するか、別の方法でパーティション化するか、またはパーティション化しないかを選択できます。パーティション化は、新しい表に引き継がれません。AS *subquery* 句の前に、CREATE TABLE 文の一部として、必要なパーティション化を指定します。

subquery 内のすべての式が、式ではなく列の場合、表定義から列を完全に省略できます。この場合、表の列名は *subquery* の列の名前と同じになります。

TO_LOB ファンクションと組み合せて *subquery* を使用すると、別の表の LONG 列の値を、作成する表の列の LOB 値に変換できます。

参照：

- LONG データを LOB にコピーする理由およびタイミングについては、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』を参照してください。
- TO_LOB ファンクションの使用方法については、5-5 ページの「変換ファンクション」を参照してください。
- *order_by_clause* の詳細は、19-4 ページの「SELECT」を参照してください。

parallel_clause この文で *parallel_clause* を指定した場合、INITIAL 記憶域パラメータに対して指定する値は無視され、かわりに NEXT パラメータの値が使用されます。

参照： これらのパラメータの詳細は、8-41 ページの「storage_clause」を参照してください。

ORDER BY ORDER BY 句を使用すると、副問合せによって戻される行の順序付けを行うことができます。

この句を CREATE TABLE で指定した場合、この句が表全体にわたるデータを順序付けるとはかぎりません。たとえば、パーティション間での順序付けは行われません。同じキーの索引を ORDER BY キー列として作成する場合に、この句を指定します。Oracle Database は、ORDER BY キーのデータをクラスタ化し、索引キーに対応させます。

表を定義する問合せの制限事項： 表問合せには、次の制限事項があります。

- 表中の列数は、副問合せ中の式の数と同じである必要があります。
- 列定義では、列名、デフォルト値および整合性制約のみを指定できます。データ型は指定できません。
- AS *subquery* を含む CREATE TABLE 文には、外部キー制約を定義できません。そのかわりに、制約を指定せずに表を作成し、後で ALTER TABLE 文を使用してその制約を追加できます。

object_table

OF 句を使用すると、明示的に *object_type* 型の**オブジェクト表**を作成できます。オブジェクト表の各列は、*object_type* 型の最上位の属性に対応します。各行には、オブジェクト・インスタンスが入り、また各インスタンスには、行の挿入時に一意のシステム生成オブジェクト識別子が割り当てられます。*schema* を省略した場合、自分のスキーマ内にオブジェクト表が作成されます。

オブジェクト表、XMLType 表、オブジェクト・ビューおよび XMLType ビューには、列名は付けられません。そのため、システム生成疑似列 OBJECT_ID が定義されます。問合せでこの列名を使用し、WITH OBJECT IDENTIFIER 句を指定して、オブジェクト・ビューを作成できます。

> **参照：** 「オブジェクト列と表の例」（16-69 ページ）

object_table_substitution

object_table_substitution 句を使用すると、サブタイプに対応する行オブジェクトの、このオブジェクト表への挿入を許可するかどうかを指定できます。

NOT SUBSTITUTABLE AT ALL LEVELS　NOT SUBSTITUTABLE AT ALL LEVELS を指定すると、作成するオブジェクト表は置換できなくなります。また、置換は、すべての埋込みオブジェクト属性および埋込みのネストした表と配列の要素には使用禁止です。デフォルトは、SUBSTITUTABLE AT ALL LEVELS です。

> **参照：**
>
> - オブジェクト型の作成の詳細は、17-3 ページの「CREATE TYPE」を参照してください。
> - REF 型の使用の詳細は、2-29 ページの「ユーザー定義型」、5-249 ページの「ユーザー定義ファンクション」、6-2 ページの「SQL 式」、17-3 ページの「CREATE TYPE」および『Oracle Database オブジェクト・リレーショナル開発者ガイド』を参照してください。

object_properties

オブジェクト表のプロパティは、基本的にリレーショナル表と同じです。ただし、列を指定するかわりに、オブジェクトの属性を指定します。

attribute には、オブジェクト内の項目の修飾した列名を指定します。

oid_clause

oid_clause を使用すると、オブジェクト表のオブジェクト識別子がシステム生成されるか、表の主キーを基に作成されるかを指定できます。デフォルトは SYSTEM GENERATED です。

***oid_clause* の制限事項：**　この句には、次の制限事項があります。

- 主キー制約を表に指定していないと、OBJECT IDENTIFIER IS PRIMARY KEY は指定できません。
- この句は、ネストした表には指定できません。

注意：　主キー・オブジェクト識別子はローカルで一意ですが、グローバルで一意であるとはかぎりません。グローバルで一意の識別子が必要な場合は、主キーがグローバルで一意であることを確認してください。

---

oid_index_clause

この句は、*oid_clause* を SYSTEM GENERATED として指定している場合のみに適用されます。非表示のオブジェクト識別子列に索引を指定します。また、任意に記憶特性を指定します。

index には、非表示のシステム生成オブジェクト識別子列の索引の名前を指定します。*index* を省略すると、名前が生成されます。

physical_properties および *table_properties*

これらの句のセマンティクスについては、リレーショナル表の対応する項を参照してください。16-30 ページの「physical_properties」および 16-36 ページの「table_properties」を参照してください。

XMLType_table

XMLType_table 構文を使用すると、XMLType データ型の表を作成できます。XMLType 表の作成に使用されるほとんどの句のセマンティクスは、オブジェクト表のセマンティクスと同じです。この項では、XMLType 表固有の句について説明します。

オブジェクト表、XMLType 表、オブジェクト・ビューおよび XMLType ビューには、列名は付けられません。そのため、システム生成疑似列 OBJECT_ID が定義されます。問合せでこの列名を使用し、WITH OBJECT IDENTIFIER 句を指定して、オブジェクト・ビューを作成できます。

XMLType_virtual_columns

この句は、バイナリ XML 記憶域の XMLType 表に対してのみ有効です。この記憶域は、*XMLType_storage_clause* で指定します。VIRTUAL COLUMNS 句を指定して、仮想列を定義します。仮想列は、ファンクション索引または制約の定義で使用できます。そのような仮想列には表の作成時には制約を定義できませんが、その後、ALTER TABLE 文を使用して制約を列に追加できます。

> **参照：** XML 環境でこの句を使用する方法の例は、『Oracle XML DB 開発者ガイド』を参照してください。

XMLSchema_spec

この句を使用すると、単一の登録済 XMLSchema または複数のスキーマ（XMLSCHEMA 句で登録するか ELEMENT 句の一部として登録）の URL および XML 要素名を指定できます。BINARY XML 記憶域を指定した場合のみ複数のスキーマを使用できます。

XMLSchema の URL は省略可能ですが、要素は必ず指定します。XMLSchema の URL を指定する場合は、DBMS_XMLSCHEMA パッケージを使用して XMLSchema をあらかじめ登録しておく必要があります。

オプションの ALLOW | DISALLOW 句は、BINARY XML 記憶域を指定した場合にのみ有効です。

- ALLOW ANYSCHEMA を指定すると、スキーマ・ベースの任意のドキュメントを XMLType 列に格納できます。
- ALLOW NONSCHEMA を指定すると、スキーマ・ベース以外のドキュメントを XMLType 列に格納できます。
- DISALLOW NONSCHEMA を指定すると、スキーマ・ベース以外のドキュメントは XMLType 列に格納できません。

> **参照：**
>
> - DBMS_XMLSCHEMA パッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。
> - XML データの作成方法および使用方法の詳細は、『Oracle XML DB 開発者ガイド』を参照してください。
> - 「XMLType 表の例 :」（16-64 ページ）

例

一般的な例

次の文は、人事情報のサンプル・スキーマ（hr）が所有する employees 表を定義します。テスト・データベースでこの例を使用できるように、表および制約には不確定な名前が指定されています。

```
CREATE TABLE employees_demo
    ( employee_id    NUMBER(6)
    , first_name     VARCHAR2(20)
    , last_name      VARCHAR2(25)
        CONSTRAINT emp_last_name_nn_demo NOT NULL
    , email          VARCHAR2(25)
        CONSTRAINT emp_email_nn_demo     NOT NULL
    , phone_number   VARCHAR2(20)
    , hire_date      DATE  DEFAULT SYSDATE
        CONSTRAINT emp_hire_date_nn_demo  NOT NULL
    , job_id         VARCHAR2(10)
       CONSTRAINT     emp_job_nn_demo  NOT NULL
    , salary         NUMBER(8,2)
       CONSTRAINT     emp_salary_nn_demo  NOT NULL
    , commission_pct NUMBER(2,2)
    , manager_id     NUMBER(6)
    , department_id  NUMBER(4)
    , dn             VARCHAR2(300)
    , CONSTRAINT     emp_salary_min_demo
                     CHECK (salary > 0)
    , CONSTRAINT     emp_email_uk_demo
                     UNIQUE (email)
    ) ;
```

この表は 12 列で構成されます。employee_id 列は、NUMBER データ型です。hire_date 列は、データ型が DATE、デフォルト値が SYSDATE です。last_name 列は、VARCHAR2 型で、NOT NULL 制約があります。他の列については省略します。

記憶域の例： 次の文は、記憶域の容量が小さく、割当てに制限のある example 表領域の中にサンプル表 employees_demo を定義します。

```
CREATE TABLE employees_demo
    ( employee_id    NUMBER(6)
    , first_name     VARCHAR2(20)
    , last_name      VARCHAR2(25)
        CONSTRAINT emp_last_name_nn_demo NOT NULL
    , email          VARCHAR2(25)
        CONSTRAINT emp_email_nn_demo     NOT NULL
    , phone_number   VARCHAR2(20)
    , hire_date      DATE  DEFAULT SYSDATE
        CONSTRAINT emp_hire_date_nn_demo  NOT NULL
    , job_id         VARCHAR2(10)
       CONSTRAINT     emp_job_nn_demo  NOT NULL
    , salary         NUMBER(8,2)
       CONSTRAINT     emp_salary_nn_demo  NOT NULL
    , commission_pct NUMBER(2,2)
    , manager_id     NUMBER(6)
    , department_id  NUMBER(4)
    , dn             VARCHAR2(300)
    , CONSTRAINT     emp_salary_min_demo
                     CHECK (salary > 0)
    , CONSTRAINT     emp_email_uk_demo
                     UNIQUE (email)
    )
   TABLESPACE example
```

```
   STORAGE (INITIAL     6144
            NEXT        6144
            MINEXTENTS     1
            MAXEXTENTS     5 );
```

一時表の例： 次の文は、販売員が使用するサンプル・データベースの一時表 today_sales を作成します。各販売員のセッションは、自身のその日の販売データを表に格納します。一時的なデータは、セッションの終わりに削除されます。

```
CREATE GLOBAL TEMPORARY TABLE today_sales
   ON COMMIT PRESERVE ROWS
   AS SELECT * FROM orders WHERE order_date = SYSDATE;
```

置換可能な表および列のサンプル： 次の文は、置換可能な表を作成するために使用可能な型の階層を作成します。employee_t 型は、person_t 型から name および ssn 属性を継承し、department_id および salary 属性を追加しています。part_time_emp_t 型は、employee_t および（employee_t を介して）person_t からすべての属性を継承し、num_hrs 属性を追加しています。part_time_emp_t はデフォルトで最終型であるため、そのサブタイプを作成できません。

```
CREATE TYPE person_t AS OBJECT (name VARCHAR2(100), ssn NUMBER)
   NOT FINAL;
/

CREATE TYPE employee_t UNDER person_t
   (department_id NUMBER, salary NUMBER) NOT FINAL;
/

CREATE TYPE part_time_emp_t UNDER employee_t (num_hrs NUMBER);
/
```

次の文は、person_t 型から置換可能な表を作成します。

```
CREATE TABLE persons OF person_t;
```

次の文は、person_t 型の置換可能な列で表を作成します。

```
CREATE TABLE books (title VARCHAR2(100), author person_t);
```

persons または books に挿入するときに、person_t またはそのサブタイプの属性に対する値を指定できます。挿入文の例は、18-65 ページの「置換可能な表および列への挿入例 :」を参照してください。

組込みファンクションおよび条件を使用して、このような表からデータを抽出することができます。その例は、5-215 ページの「TREAT」、5-191 ページの「SYS_TYPEID」および 7-23 ページの「IS OF type 条件」を参照してください。

PARALLEL の例： 次の文は、最適な数のパラレル実行サーバーを使用する表を作成して、employees をスキャンし、dept_80 に移入します。

```
CREATE TABLE dept_80
   PARALLEL
   AS SELECT * FROM employees
   WHERE department_id = 80;
```

パラレル化を使用することによって、表を作成するためにパラレル実行サーバーが使用されるため、表作成が高速化されます。表が作成された後、表へのアクセスに表作成と同じ並列度が使用されるため、表の問合せも高速化します。

NOPARALLEL の例： 次の文は、同じ表をシリアルで作成します。後続の DML および表の問合せもシリアルで実行されます。

```
CREATE TABLE dept_80
   AS SELECT * FROM employees
   WHERE department_id = 80;
```

ENABLE VALIDATE の例： 次の文は、サンプル表 departments_demo を作成します。この例では、NOT NULL 制約が定義され、ENABLE VALIDATE 状態に置かれます。テスト・データベースでこの例を使用できるように、表には不確定な名前が指定されています。

```
CREATE TABLE departments_demo
    ( department_id    NUMBER(4)
    , department_name  VARCHAR2(30)
          CONSTRAINT  dept_name_nn  NOT NULL
    , manager_id       NUMBER(6)
    , location_id      NUMBER(4)
    , dn               VARCHAR2(300)
    ) ;
```

DISABLE の例： 次の文は、同じ departments_demo 表を作成しますが、使用禁止の主キー制約も定義します。

```
CREATE TABLE departments_demo
    ( department_id    NUMBER(4)   PRIMARY KEY DISABLE
    , department_name  VARCHAR2(30)
          CONSTRAINT  dept_name_nn  NOT NULL
    , manager_id       NUMBER(6)
    , location_id      NUMBER(4)
    , dn               VARCHAR2(300)
    ) ;
```

ネストした表の例： 次の文は、ネストした表の列 ad_textdocs_ntab を使用してサンプル表 pm.print_media を作成します。

```
CREATE TABLE print_media
    ( product_id        NUMBER(6)
    , ad_id             NUMBER(6)
    , ad_composite      BLOB
    , ad_sourcetext     CLOB
    , ad_finaltext      CLOB
    , ad_fltextn        NCLOB
    , ad_textdocs_ntab  textdoc_tab
    , ad_photo          BLOB
    , ad_graphic        BFILE
    , ad_header         adheader_typ
    ) NESTED TABLE ad_textdocs_ntab STORE AS textdocs_nestedtab;
```

マルチレベル・コレクションの例： 次の例では、アカウント・マネージャが 2 つのレベルのネストした表を使用して顧客の表を作成します。

```
CREATE TYPE phone AS OBJECT (telephone NUMBER);
/
CREATE TYPE phone_list AS TABLE OF phone;
/
CREATE TYPE my_customers AS OBJECT (
   cust_name VARCHAR2(25),
   phones phone_list);
/
CREATE TYPE customer_list AS TABLE OF my_customers;
/
CREATE TABLE business_contacts (
   company_name VARCHAR2(25),
```

```
   company_reps customer_list)
   NESTED TABLE company_reps STORE AS outer_ntab
   (NESTED TABLE phones STORE AS inner_ntab);
```

前述の例を次のように使用した場合、内部のネストした表に列または属性名がない場合に COLUMN_VALUE キーワードが使用されます。

```
CREATE TYPE phone AS TABLE OF NUMBER;
/
CREATE TYPE phone_list AS TABLE OF phone;
/
CREATE TABLE my_customers (
   name VARCHAR2(25),
   phone_numbers phone_list)
   NESTED TABLE phone_numbers STORE AS outer_ntab
   (NESTED TABLE COLUMN_VALUE STORE AS inner_ntab);
```

LOB 列の例 : 次の文は、LOB 記憶特性を追加して pm.print_media を作成します。

```
CREATE TABLE print_media_new
    ( product_id        NUMBER(6)
    , ad_id             NUMBER(6)
    , ad_composite      BLOB
    , ad_sourcetext     CLOB
    , ad_finaltext      CLOB
    , ad_fltextn        NCLOB
    , ad_textdocs_ntab  textdoc_tab
    , ad_photo          BLOB
    , ad_graphic        BFILE
    , ad_header         adheader_typ
    , press_release     LONG
    ) NESTED TABLE ad_textdocs_ntab STORE AS textdocs_nestedtab_new
    LOB (ad_sourcetext, ad_finaltext) STORE AS
      (TABLESPACE example
       STORAGE (INITIAL 6144 NEXT 6144)
       CHUNK 4000
       NOCACHE LOGGING);
```

この例では、CHUNK の値を 4096（2048 のブロック・サイズの近似倍数）まで切り上げます。

索引構成表の例 : 次の文は、索引を構成したサンプル表 hr.countries を作成します。

```
CREATE TABLE countries_demo
    ( country_id      CHAR(2)
      CONSTRAINT country_id_nn_demo NOT NULL
    , country_name    VARCHAR2(40)
    , currency_name   VARCHAR2(25)
    , currency_symbol VARCHAR2(3)
    , region          VARCHAR2(15)
    , CONSTRAINT    country_c_id_pk_demo
                    PRIMARY KEY (country_id ) )
    ORGANIZATION INDEX
    INCLUDING   country_name
    PCTTHRESHOLD 2
    STORAGE
     ( INITIAL  4K
      NEXT  2K
      PCTINCREASE 0
      MINEXTENTS 1
      MAXEXTENTS 1 )
   OVERFLOW
    STORAGE
      ( INITIAL  4K
```

```
        NEXT  2K
        PCTINCREASE 0
        MINEXTENTS 1
        MAXEXTENTS 1 );
```

外部表の例： 次の文は、サンプル表 hr.departments のサブセットを示す外部表を作成します。*opaque_format_spec* はイタリック体で示しています。ORACLE_LOADER アクセス・ドライバの詳細および *opaque_format_spec* への値の指定方法については、『Oracle Database ユーティリティ』を参照してください。

```
CREATE TABLE dept_external (
   deptno     NUMBER(6),
   dname      VARCHAR2(20),
   loc        VARCHAR2(25)
)
ORGANIZATION EXTERNAL
(TYPE oracle_loader
 DEFAULT DIRECTORY admin
 ACCESS PARAMETERS
 (
  RECORDS DELIMITED BY newline
  BADFILE 'ulcase1.bad'
  DISCARDFILE 'ulcase1.dis'
  LOGFILE 'ulcase1.log'
  SKIP 20
  FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
  (
   deptno     INTEGER EXTERNAL(6),
   dname      CHAR(20),
   loc        CHAR(25)
  )
 )
 LOCATION ('ulcase1.ctl')
)
REJECT LIMIT UNLIMITED;
```

> **参照：** admin ディレクトリの作成方法は、14-39 ページの「ディレクトリの作成例 :」を参照してください。

XMLType の例

この項では、XMLType 表または XMLType 列の作成例について説明します。この例の詳細は、E-8 ページの「SQL 文での XML の使用方法」を参照してください。

XMLType 表の例： 次の例は、暗黙的に CLOB 列を 1 列のみ持つ非常に単純な XMLType 表を作成します。

```
CREATE TABLE xwarehouses OF XMLTYPE;
```

データは暗黙的に CLOB 列に格納されるため、LOB 列に対するすべての制限事項が適用されます。これらの制限事項を回避するには、次の例で示すとおり XMLSchema ベースの表を作成します。XMLSchema は事前に作成しておく必要があります（詳細は、E-8 ページの「SQL 文での XML の使用方法」を参照してください）。

```
CREATE TABLE xwarehouses OF XMLTYPE
   XMLSCHEMA "http://www.example.com/xwarehouses.xsd"
   ELEMENT "Warehouse";
```

XMLSchema ベースの表に制約を定義したり、索引を作成できるため、後続の問合せのパフォーマンスが大幅に向上します。XMLType 表にオブジェクト・リレーショナル・ビューを作成することも、オブジェクト・リレーショナル表に XMLType ビューを作成することもできます。

参照 :

- 制約の追加例は、E-8 ページの「SQL 文での XML の使用方法」を参照してください。
- 索引の作成例は、14-68 ページの「XMLType 表の索引の作成例 :」を参照してください。
- XMLType ビューの作成例は、17-22 ページの「XMLType ビューの作成例 :」を参照してください。

XMLType 列の例 : 次の文は、XMLType 列を持つ表を作成し、CLOB として格納します。この表では XMLSchema が必要ないため、コンテンツ構造は事前に定義しません。

```
CREATE TABLE xwarehouses (
   warehouse_id        NUMBER,
   warehouse_spec      XMLTYPE)
   XMLTYPE warehouse_spec STORE AS CLOB
   (TABLESPACE example
    STORAGE (INITIAL 6144 NEXT 6144)
    CHUNK 4000
    NOCACHE LOGGING);
```

次の例は、同様の表を作成しますが、オブジェクト・リレーショナル XMLType 列に XMLType データを格納します。この列の構造は、指定したスキーマによって判断されます。

```
CREATE TABLE xwarehouses (
   warehouse_id    NUMBER,
   warehouse_spec  XMLTYPE)
   XMLTYPE warehouse_spec STORE AS OBJECT RELATIONAL
      XMLSCHEMA "http://www.example.com/xwarehouses.xsd"
      ELEMENT "Warehouse";
```

次の例は、SecureFile CLOB として格納された XMLType 列を持つ別の同様の表を作成します。この表では XMLSchema が必要ないため、コンテンツ構造は事前に定義しません。SecureFile LOB には自動セグメント領域管理が行われる表領域が必要であるため、この例は 16-84 ページの「表領域に対してセグメント領域管理を指定する場合の例 :」で作成される表領域を使用します。

```
CREATE TABLE xwarehouses (
  warehouse_id   NUMBER,
  warehouse_spec XMLTYPE)
  XMLTYPE        warehouse_spec STORE AS SECUREFILE CLOB
  (TABLESPACE auto_seg_ts
  STORAGE (INITIAL 6144 NEXT 6144)
  CACHE);
```

パーティション化の例

レンジ・パーティション化の例 : サンプル・スキーマ sh の sales 表は、レンジでパーティション化されています。次の例では、簡略化した sales 表を作成します。この例では、制約および記憶域要素は省略されています。

```
CREATE TABLE range_sales
    ( prod_id        NUMBER(6)
    , cust_id        NUMBER
    , time_id        DATE
    , channel_id     CHAR(1)
    , promo_id       NUMBER(6)
    , quantity_sold  NUMBER(3)
    , amount_sold         NUMBER(10,2)
    )
PARTITION BY RANGE (time_id)
  (PARTITION SALES_Q1_1998 VALUES LESS THAN (TO_DATE('01-APR-1998','DD-MON-YYYY')),
```

```
   PARTITION SALES_Q2_1998 VALUES LESS THAN (TO_DATE('01-JUL-1998','DD-MON-YYYY')),
   PARTITION SALES_Q3_1998 VALUES LESS THAN (TO_DATE('01-OCT-1998','DD-MON-YYYY')),
   PARTITION SALES_Q4_1998 VALUES LESS THAN (TO_DATE('01-JAN-1999','DD-MON-YYYY')),
   PARTITION SALES_Q1_1999 VALUES LESS THAN (TO_DATE('01-APR-1999','DD-MON-YYYY')),
   PARTITION SALES_Q2_1999 VALUES LESS THAN (TO_DATE('01-JUL-1999','DD-MON-YYYY')),
   PARTITION SALES_Q3_1999 VALUES LESS THAN (TO_DATE('01-OCT-1999','DD-MON-YYYY')),
   PARTITION SALES_Q4_1999 VALUES LESS THAN (TO_DATE('01-JAN-2000','DD-MON-YYYY')),
   PARTITION SALES_Q1_2000 VALUES LESS THAN (TO_DATE('01-APR-2000','DD-MON-YYYY')),
   PARTITION SALES_Q2_2000 VALUES LESS THAN (TO_DATE('01-JUL-2000','DD-MON-YYYY')),
   PARTITION SALES_Q3_2000 VALUES LESS THAN (TO_DATE('01-OCT-2000','DD-MON-YYYY')),
   PARTITION SALES_Q4_2000 VALUES LESS THAN (MAXVALUE))
;
```

パーティション表のメンテナンス操作については、『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。

時間隔パーティションの例： 次の例は、credit_limit 列の期間によってパーティション化される oe.customers 表を作成します。遷移ポイントを設定するために 1 つのレンジ・パーティションが作成されます。表内の元のデータはすべてレンジ・パーティションの境界内にあります。次に、レンジ・パーティションを超えるデータが追加され、新しい時間隔パーティションが作成されます。

```
CREATE TABLE customers_demo (
  customer_id number(6),
  cust_first_name varchar2(20),
  cust_last_name varchar2(20),
  credit_limit number(9,2))
PARTITION BY RANGE (credit_limit)
INTERVAL (1000)
(PARTITION p1 VALUES LESS THAN (5001));

INSERT INTO customers_demo
  (customer_id, cust_first_name, cust_last_name, credit_limit)
  (select customer_id, cust_first_name, cust_last_name, credit_limit
  from customers);
```

USER_TAB_PARTITIONS データ・ディクショナリ・ビューを問い合せてから、時間隔パーティションが作成されます。

```
SELECT partition_name, high_value FROM user_tab_partitions
  WHERE table_name = 'CUSTOMERS_DEMO';

PARTITION_NAME                 HIGH_VALUE
------------------------------ ---------------
P1                             5001
```

レンジ・パーティションの上限を超えるデータを表に挿入します。

```
INSERT INTO customers_demo
  VALUES (699, 'Fred', 'Flintstone', 5500);
```

挿入後に USER_TAB_PARTITIONS ビューを再度問い合せ、挿入されたデータを格納するために作成された時間隔パーティションのシステム生成名を確認します。（システム生成名はセッションごとに異なります。）

```
SELECT partition_name, high_value FROM user_tab_partitions
  WHERE table_name = 'CUSTOMERS_DEMO'
  ORDER BY partition_name;

PARTITION_NAME                 HIGH_VALUE
------------------------------ ---------------
P1                             5001
SYS_P44                        6001
```

リスト・パーティション化の例： 次の文は、サンプル表 oe.customers をリスト・パーティション表として作成します。この例では、サンプル表の一部の列およびすべての制約は省略されています。

```
CREATE TABLE list_customers
   ( customer_id             NUMBER(6)
   , cust_first_name         VARCHAR2(20)
   , cust_last_name          VARCHAR2(20)
   , cust_address            CUST_ADDRESS_TYP
   , nls_territory           VARCHAR2(30)
   , cust_email              VARCHAR2(30))
   PARTITION BY LIST (nls_territory) (
   PARTITION asia VALUES ('CHINA', 'THAILAND'),
   PARTITION europe VALUES ('GERMANY', 'ITALY', 'SWITZERLAND'),
   PARTITION west VALUES ('AMERICA'),
   PARTITION east VALUES ('INDIA'),
   PARTITION rest VALUES (DEFAULT));
```

LOB 列のあるパーティション表の例： 次の文は、2 つのパーティション p1 と p2、およびいくつかの LOB 列を持つパーティション表 print_media_demo を作成します。この文では、サンプル表 pm.print_media を使用しますが、LONG 列はパーティション化でサポートされないため、LONG 列の press_release は省略します。

```
CREATE TABLE print_media_demo
   ( product_id NUMBER(6)
   , ad_id NUMBER(6)
   , ad_composite BLOB
   , ad_sourcetext CLOB
   , ad_finaltext CLOB
   , ad_fltextn NCLOB
   , ad_textdocs_ntab textdoc_tab
   , ad_photo BLOB
   , ad_graphic BFILE
   , ad_header adheader_typ
   ) NESTED TABLE ad_textdocs_ntab STORE AS textdocs_nestedtab_demo
      LOB (ad_composite, ad_photo, ad_finaltext)
      STORE AS(STORAGE (NEXT 20M))
   PARTITION BY RANGE (product_id)
      (PARTITION p1 VALUES LESS THAN (3000) TABLESPACE tbs_01
         LOB (ad_composite, ad_photo)
         STORE AS (TABLESPACE tbs_02 STORAGE (INITIAL 10M)),
       PARTITION P2 VALUES LESS THAN (MAXVALUE)
         LOB (ad_composite, ad_finaltext)
         STORE AS SECUREFILE (TABLESPACE auto_seg_ts)
       )
   TABLESPACE tbs_04;
```

パーティション p1 は、表領域 tbs_1 にあります。ad_composite および ad_photo に対する LOB データのパーティションは、表領域 tbs_2 にあります。残りの LOB 列に対する LOB データのパーティションは、表領域 tbs_1 にあります。LOB 列 ad_composite および ad_photo に、記憶域属性 INITIAL を指定します。他の属性は、デフォルトの表レベルの仕様から継承されます。表レベルで指定されていないデフォルトの LOB 記憶域属性は、ad_composite 列および ad_photo 列については表領域 tbs_2 から継承されます。残りの LOB 列については、表領域 tbs_1 から継承されます。LOB 索引パーティションは、対応する LOB データ・パーティションと同じ表領域に存在します。他の記憶域属性は、LOB データ・パーティションの対応する属性値および索引パーティションがある表領域のデフォルト属性に基づきます。

パーティション p2 は、デフォルトの表領域 tbs_4 内にあります。ad_composite および ad_finaltext に対する LOB データは、表領域 auto_seg_ts 内に SecureFile LOB としてあります。残りの LOB 列に対する LOB データは、表領域 tbs_4 にあります。ad_composite 列および ad_finaltext 列に対する LOB 索引は、表領域 auto_seg_ts 内にあります。残りの LOB 列に対する LOB 索引は、表領域 tbs_4 にあります。

ハッシュ・パーティション化の例： サンプル表 oe.hash_product は、パーティション化されていません。次の例は、パフォーマンス上の理由で、このような大規模な表をハッシュでパーティション化します。この例では、表領域は仮想の名前です。

```
CREATE TABLE hash_products
    ( product_id          NUMBER(6)    PRIMARY KEY
    , product_name        VARCHAR2(50)
    , product_description VARCHAR2(2000)
    , category_id         NUMBER(2)
    , weight_class        NUMBER(1)
    , warranty_period     INTERVAL YEAR TO MONTH
    , supplier_id         NUMBER(6)
    , product_status      VARCHAR2(20)
    , list_price          NUMBER(8,2)
    , min_price           NUMBER(8,2)
    , catalog_url         VARCHAR2(50)
    , CONSTRAINT          product_status_lov_demo
                          CHECK (product_status in ('orderable'
                                                    ,'planned'
                                                    ,'under development'
                                                    ,'obsolete')
 ) )
 PARTITION BY HASH (product_id)
 PARTITIONS 4
 STORE IN (tbs_01, tbs_02, tbs_03, tbs_04);
```

参照パーティション化の例： 次の文は、前述の例で作成された hash_products パーティション表を使用します。hash_products の製品 ID に基づくハッシュ・パーティション化への参照によってパーティション化された oe.order_items 表を作成します。結果の子表は、5 つのパーティションで作成されます。子表 part_order_items の各行について、外部キー値（product_id）が評価され、参照されるキーが属する親表 hash_products のパーティション番号が確認されます。part_order_items の行は、対応するパーティションに配置されます。

```
CREATE TABLE part_order_items (
    order_id        NUMBER(12) PRIMARY KEY,
    line_item_id    NUMBER(3),
    product_id      NUMBER(6) NOT NULL,
    unit_price      NUMBER(8,2),
    quantity        NUMBER(8),
    CONSTRAINT product_id_fk
    FOREIGN KEY (product_id) REFERENCES hash_products(product_id))
 PARTITION BY REFERENCE (product_id_fk);
```

コンポジット・パーティション表の例： 次の例は、表（16-65 ページの「レンジ・パーティション化の例：」で作成）のデータを販売時刻で分割します。時刻と販売チャネルに従って最近のデータにアクセスする場合、コンポジット・パーティション化は、より適切です。次の例では、同じ range_sales 表のコピーを作成しますが、レンジ - ハッシュ・コンポジット・パーティション化を指定します。最新のデータがあるパーティションは、システム生成とユーザー定義の両方のサブパーティション名でサブパーティション化されます。この例では、制約および記憶域属性は省略されています。

```
CREATE TABLE composite_sales
    ( prod_id        NUMBER(6)
    , cust_id        NUMBER
    , time_id        DATE
    , channel_id     CHAR(1)
    , promo_id       NUMBER(6)
    , quantity_sold  NUMBER(3)
    , amount_sold         NUMBER(10,2)
    )
PARTITION BY RANGE (time_id)
```

```
SUBPARTITION BY HASH (channel_id)
  (PARTITION SALES_Q1_1998 VALUES LESS THAN (TO_DATE('01-APR-1998','DD-MON-YYYY')),
   PARTITION SALES_Q2_1998 VALUES LESS THAN (TO_DATE('01-JUL-1998','DD-MON-YYYY')),
   PARTITION SALES_Q3_1998 VALUES LESS THAN (TO_DATE('01-OCT-1998','DD-MON-YYYY')),
   PARTITION SALES_Q4_1998 VALUES LESS THAN (TO_DATE('01-JAN-1999','DD-MON-YYYY')),
   PARTITION SALES_Q1_1999 VALUES LESS THAN (TO_DATE('01-APR-1999','DD-MON-YYYY')),
   PARTITION SALES_Q2_1999 VALUES LESS THAN (TO_DATE('01-JUL-1999','DD-MON-YYYY')),
   PARTITION SALES_Q3_1999 VALUES LESS THAN (TO_DATE('01-OCT-1999','DD-MON-YYYY')),
   PARTITION SALES_Q4_1999 VALUES LESS THAN (TO_DATE('01-JAN-2000','DD-MON-YYYY')),
   PARTITION SALES_Q1_2000 VALUES LESS THAN (TO_DATE('01-APR-2000','DD-MON-YYYY')),
   PARTITION SALES_Q2_2000 VALUES LESS THAN (TO_DATE('01-JUL-2000','DD-MON-YYYY'))
      SUBPARTITIONS 8,
   PARTITION SALES_Q3_2000 VALUES LESS THAN (TO_DATE('01-OCT-2000','DD-MON-YYYY'))
     (SUBPARTITION ch_c,
      SUBPARTITION ch_i,
      SUBPARTITION ch_p,
      SUBPARTITION ch_s,
      SUBPARTITION ch_t),
   PARTITION SALES_Q4_2000 VALUES LESS THAN (MAXVALUE)
      SUBPARTITIONS 4)
;
```

次の例は、サンプル表 oe.customers に基づいて、顧客のパーティション表を作成します。この例では、表は credit_limit 列でパーティション化され、nls_territory 列でリスト・サブパーティション化されます。個々のサブパーティションを定義してテンプレートを上書きしないかぎり、サブパーティション・テンプレートによって、後から追加されたパーティションのサブパーティション化が決定されます。このコンポジット・パーティション化によって、指定した地域内の掛貸限度範囲に基づいて、表を問い合せることができます。

```
CREATE TABLE customers_part (
   customer_id        NUMBER(6),
   cust_first_name    VARCHAR2(20),
   cust_last_name     VARCHAR2(20),
   nls_territory      VARCHAR2(30),
   credit_limit       NUMBER(9,2))
   PARTITION BY RANGE (credit_limit)
   SUBPARTITION BY LIST (nls_territory)
      SUBPARTITION TEMPLATE
         (SUBPARTITION east  VALUES
            ('CHINA', 'JAPAN', 'INDIA', 'THAILAND'),
          SUBPARTITION west VALUES
             ('AMERICA', 'GERMANY', 'ITALY', 'SWITZERLAND'),
          SUBPARTITION other VALUES (DEFAULT))
      (PARTITION p1 VALUES LESS THAN (1000),
       PARTITION p2 VALUES LESS THAN (2500),
       PARTITION p3 VALUES LESS THAN (MAXVALUE));
```

オブジェクト列と表の例

オブジェクト表の作成例 : 次の文は、オブジェクト型 department_typ を指定します。

```
CREATE TYPE department_typ AS OBJECT
   ( d_name   VARCHAR2(100),
     d_address VARCHAR2(200) );
/
```

オブジェクト表 departments_obj_t は、department_typ 型の部門オブジェクトを保持します。

```
CREATE TABLE departments_obj_t OF department_typ;
```

次の文は、ユーザー定義オブジェクト型 salesrep_typ を持つオブジェクト表 salesreps を作成します。

```
CREATE OR REPLACE TYPE salesrep_typ AS OBJECT
  ( repId NUMBER,
    repName VARCHAR2(64));

CREATE TABLE salesreps OF salesrep_typ;
```

ユーザー定義のオブジェクト識別子を含む表の作成例： 次の文は、オブジェクト型およびオブジェクト識別子が主キー・ベースの対応するオブジェクト表を作成します。

```
CREATE TYPE employees_typ AS OBJECT
   (e_no NUMBER, e_address CHAR(30));
/

CREATE TABLE employees_obj_t OF employees_typ (e_no PRIMARY KEY)
   OBJECT IDENTIFIER IS PRIMARY KEY;
```

その後、*inline_ref_constraint* 構文または *out_of_line_ref_constraint* 構文のいずれかを使用して、employees_object_t オブジェクト表を参照できます。

```
CREATE TABLE departments_t
   (d_no    NUMBER,
    mgr_ref REF employees_typ SCOPE IS employees_obj_t);

CREATE TABLE departments_t (
    d_no NUMBER,
    mgr_ref REF employees_typ
       CONSTRAINT mgr_in_emp REFERENCES employees_obj_t);
```

タイプ列での制約の指定例： 次の例は、オブジェクト型の列の属性に制約を定義します。

```
CREATE TYPE address_t AS OBJECT
  ( hno    NUMBER,
    street VARCHAR2(40),
    city   VARCHAR2(20),
    zip    VARCHAR2(5),
    phone  VARCHAR2(10) );
/

CREATE TYPE person AS OBJECT
  ( name        VARCHAR2(40),
    dateofbirth DATE,
    homeaddress address_t,
    manager     REF person );
/

CREATE TABLE persons OF person
  ( homeaddress NOT NULL,
     UNIQUE (homeaddress.phone),
     CHECK (homeaddress.zip IS NOT NULL),
     CHECK (homeaddress.city <> 'San Francisco') );
```

CREATE TABLESPACE

用途

CREATE TABLESPACE 文を使用すると、**表領域**を作成できます。表領域とは、スキーマ・オブジェクトを格納する、データベース内に割り当てられた領域です。

- **永続表領域**には、永続スキーマ・オブジェクトが格納されます。永続表領域のオブジェクトは、**データ・ファイル**に格納されます。
- **UNDO 表領域**は、データベースを自動 UNDO 管理モードで実行している場合に Oracle Database が UNDO データを管理するために使用する永続表領域です。UNDO には、ロールバック・セグメントではなく、自動 UNDO 管理モードを使用することをお薦めします。
- **一時表領域**には、セッションの存続期間中のみ、スキーマ・オブジェクトが格納されます。一時表領域のオブジェクトは、**一時ファイル**に格納されます。

表領域は、最初は読み書き両用として作成されます。その後、ALTER TABLESPACE 文でその表領域をオフラインまたはオンラインに設定したり、表領域にデータ・ファイルや一時ファイルを追加したり、読取り専用に設定することができます。

DROP TABLESPACE 文を使用して、データベースから表領域を削除することもできます。

参照：

- 表領域については、『Oracle Database 概要』を参照してください。
- 表領域の変更および削除の詳細は、12-82 ページの「ALTER TABLESPACE」および 18-8 ページの「DROP TABLESPACE」を参照してください。

前提条件

CREATE TABLESPACE システム権限が必要です。SYSAUX 表領域を作成する場合は、SYSDBA システム権限が必要です。

表領域を作成する場合、表領域を格納するデータベースを作成する必要があります。また、そのデータベースをオープンしておく必要もあります。

参照：「CREATE DATABASE」（14-16 ページ）

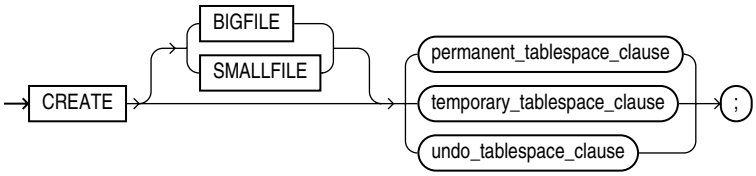
SYSTEM 以外の表領域のオブジェクトを使用する場合は、次の注意事項があります。

- 自動 UNDO 管理モードでデータベースを実行するには、1 つ以上の UNDO 表領域がオンラインである必要があります。
- 手動 UNDO 管理モードのデータベースを実行するには、SYSTEM ロールバック・セグメント以外の 1 つ以上のロールバック・セグメントがオンラインである必要があります。

注意： データベースを自動 UNDO 管理モードで実行することをお薦めします。詳細は、『Oracle Database 管理者ガイド』を参照してください。

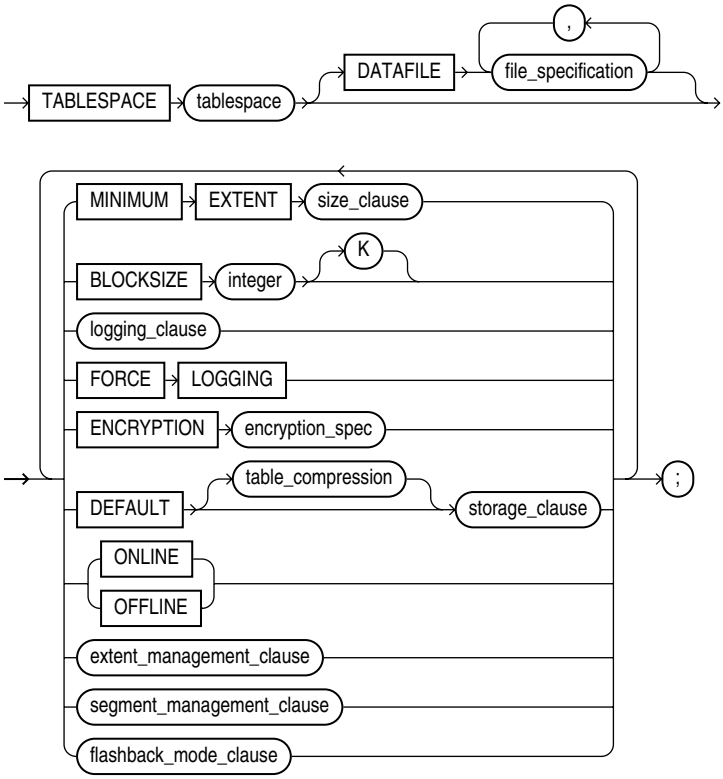
構文

***create_tablespace*::=**



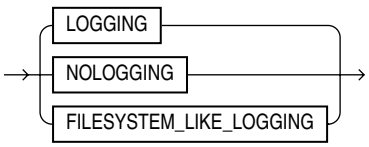
（16-74 ページの permanent_tablespace_clause、16-80 ページの temporary_tablespace_clause、16-81 ページの undo_tablespace_clause を参照）

permanent_tablespace_clause::=

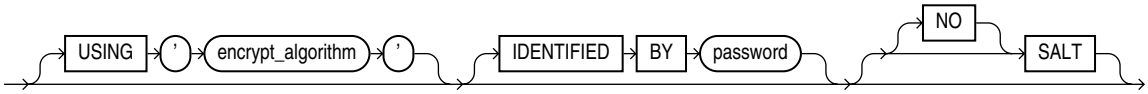


（8-26 ページの file_specification::=、8-42 ページの size_clause::=、16-72 ページの logging_clause::=、16-73 ページの encryption_spec::=、16-73 ページの table_compression::=、8-44 ページの storage_clause::=、16-73 ページの extent_management_clause::=、16-73 ページの segment_management_clause::=、16-73 ページの flashback_mode_clause::= を参照）

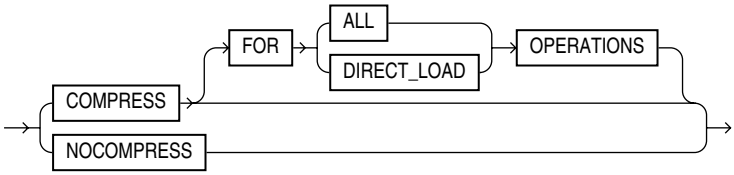
***logging_clause*::=**



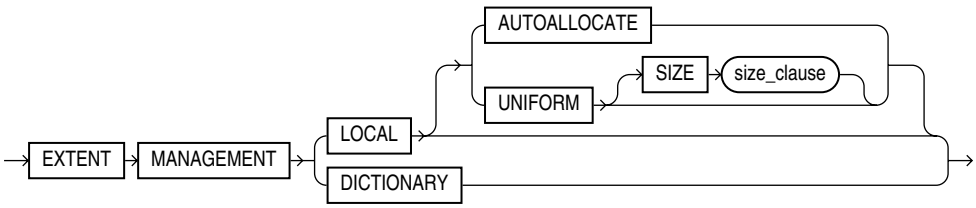
***encryption_spec*::=**



***table_compression*::=**



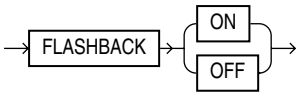
extent_management_clause::=



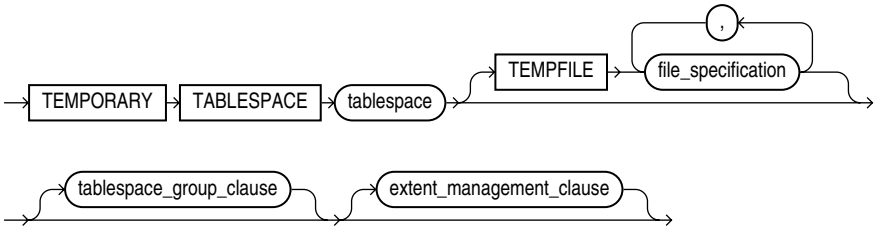
(8-42 ページの size_clause::= を参照)

segment_management_clause::=

flashback_mode_clause::=

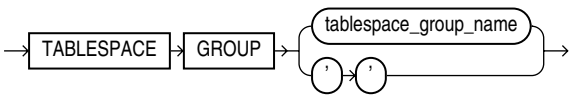


temporary_tablespace_clause::=

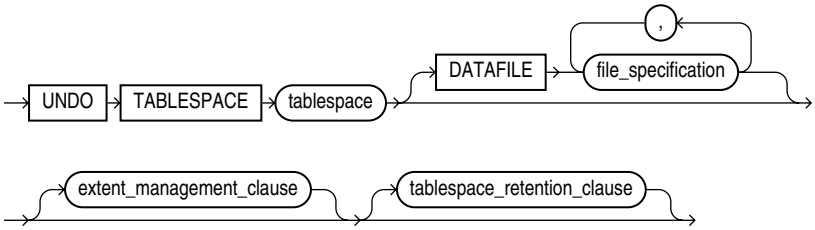


(8-26 ページの file_specification::=、16-81 ページの tablespace_group_clause、16-73 ページの extent_management_clause::= を参照)

tablespace_group_clause::=

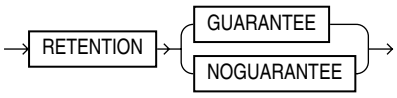


***undo_tablespace_clause*::=**



（8-26 ページの file_specification::=、16-73 ページの extent_management_clause::=、16-74 ページの tablespace_retention_clause::= を参照）

tablespace_retention_clause::=



セマンティクス

BIGFILE | SMALLFILE

この句を使用すると、表領域が bigfile か smallfile かを指定できます。この句は、データベースのデフォルトの表領域タイプの設定を上書きします。

- **bigfile 表領域**には、1 つのデータ・ファイルまたは一時ファイルのみが含まれます。ファイルは、最大で約 40 億（2^{32}）のブロックを格納できます。1 つのデータ・ファイルまたは一時ファイルの最大サイズは、32K ブロックの表領域で 128TB、8K ブロックの表領域で 32TB です。
- **smallfile 表領域**は、Oracle の従来の表領域であり、1022 のデータ・ファイルまたは一時ファイルを含めることができます。それぞれのファイルは、最大で約 400 万（2^{22}）のブロックを格納できます。

この句を省略した場合、データベースに設定された永続表領域または一時表領域の、現在のデフォルト表領域タイプが使用されます。永続表領域に `BIGFILE` を指定すると、デフォルトで、自動セグメント領域管理のローカル管理表領域が作成されます。

bigfile 表領域の制限事項： bigfile 表領域には、次の制限事項があります。

- `DATAFILE` 句に 1 つのみのデータ・ファイルを指定するか、または `TEMPFILE` 句に 1 つのみの一時ファイルを指定できます。
- `EXTENT MANAGEMENT DICTIONARY` を指定することはできません。

参照：

- bigfile 表領域の使用の詳細は、『Oracle Database 管理者ガイド』を参照してください。
- 「bigfile 表領域の作成例：」（16-82 ページ）

permanent_tablespace_clause

次の句を使用すると、永続表領域を作成できます。（これらの句の一部は、一時表領域または UNDO 表領域の作成にも使用されます。）

tablespace

作成する表領域の名前を指定します。

SYSAUX 表領域の注意事項： SYSAUX は、必須の補助システム表領域です。以前のリリースから Oracle Database 11*g* にアップグレードする場合、CREATE TABLESPACE 文を使用して SYSAUX 表領域を作成する必要があります。この句を指定するには、SYSDBA システム権限が必要です。また、データベースが MIGRATE モードでオープンされている必要があります。

SYSAUX 表領域には、EXTENT MANAGEMENT LOCAL および SEGMENT SPACE MANAGEMENT AUTO を指定する必要があります。DATAFILE 句は、Oracle Managed Files が使用可能になっている場合のみのオプションです。DATAFILE 句の動作については、16-75 ページの「DATAFILE | TEMPFILE 句」を参照してください。

SYSAUX 表領域には十分な領域を割り当ててください。この表領域の作成のガイドラインは、『Oracle Database アップグレード・ガイド』を参照してください。

SYSAUX 表領域の制限事項： SYSAUX 表領域に OFFLINE または TEMPORARY を指定することはできません。

DATAFILE | TEMPFILE 句

永続表領域を構成するデータ・ファイルを指定するか、または一時表領域を構成する一時ファイルを指定します。オペレーティング・システムのファイル・システム内の標準データ・ファイルと一時ファイル、または自動ストレージ管理ディスク・グループのファイルを作成するには、*file_specification* の datafile_tempfile_spec 書式を使用します。

DB_CREATE_FILE_DEST 初期化パラメータに値を設定して Oracle Managed Files を使用可能にしていないかぎり、DATAFILE または TEMPFILE 句を指定する必要があります。自動ストレージ管理ディスク・グループ・ファイルの場合、このパラメータを自動ストレージ管理ファイル名の複数ファイル作成形式に設定する必要があります。このパラメータが設定されている場合、パラメータで指定したデフォルトのファイルの出力先に、100MB のファイルがシステム名で作成されます。このファイルでは AUTOEXTEND が有効になっているため、最大サイズの制限はありません。

注意： メディア・リカバリは一時ファイルを認識しません。

参照：

- 自動ストレージ管理の使用については、『Oracle Database ストレージ管理者ガイド』を参照してください。
- AUTOEXTEND パラメータや自動ストレージ管理ファイル名の複数ファイル作成形式などの詳細は、8-26 ページの「file_specification」を参照してください。

データ・ファイルおよび一時ファイルの指定の注意事項：

- RAW デバイスをサポートするオペレーティング・システムで RAW デバイスをデータ・ファイルとして指定する場合、*datafile_tempfile_spec* に REUSE キーワードを指定しても意味がありません。このような CREATE TABLESPACE 文は、REUSE の指定の有無にかかわらず実行されます。
- *datafile_tempfile_spec* にディスク・グループ名のみを指定することによって、自動ストレージ管理ディスク・グループ内に表領域を作成できます。この場合、自動ストレージ管理は、システムが生成するファイル名で、指定したディスク・グループにデータ・ファイルを作成します。データ・ファイルは自動拡張可能であり、最大サイズの制限はありません。デフォルトのサイズは 100MB です。デフォルトのサイズは、*autoextend_clause* を使用して上書きできます。
- 既存のファイルを参照する *ASM_filename* の参照形式の 1 つを使用する場合、REUSE も指定する必要があります。

注意： オペレーティング・システムによっては、一時ファイルのブロックが実際にアクセスされるまで、一時ファイル用の領域が割り当てられない場合があります。領域の割当ての遅延のため、一時ファイルの作成およびサイズ変更が速くなります。ただし、後で一時ファイルが使用されるときに、十分なディスク領域を使用可能にする必要があります。発生する可能性がある問題を回避するには、一時ファイルを作成またはサイズ変更する前に、ディスク領域が、新しく作成する一時ファイルまたはサイズ変更後の一時ファイルのサイズより大きいことを確認してください。ディスク領域に余裕を持たせることによって、関連のない操作が原因で増加が予想されるディスク使用量に対応できます。十分な領域があることを確認した後で、作成またはサイズ変更操作を実行してください。

参照：

- AUTOEXTEND パラメータなどの詳細は、8-26 ページの「file_specification」を参照してください。
- 16-83 ページの「表領域の自動拡張を使用可能にする場合の例 :」および 16-84 ページの「Oracle Managed Files の作成例 :」を参照してください。

MINIMUM EXTENT 句

この句は、ディクショナリ管理表領域にのみ有効です。表領域で使用されるエクステントの最小サイズを指定します。この句を指定すると、表領域内のすべての使用済エクステントまたは未使用エクステントのサイズが、*size_clause* で指定された値以上であること、およびその倍数であることが保証され、表領域における空き領域の断片化を制御できます。

参照： この句の詳細は、8-42 ページの「size_clause」を参照してください。MINIMUM EXTENT を使用した断片化の制御については、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

BLOCKSIZE 句

BLOCKSIZE 句を使用すると、表領域に対して標準以外のブロック・サイズを指定できます。この句を指定するには、DB_CACHE_SIZE および 1 つ以上の DB_*n*K_CACHE_SIZE パラメータが設定されている必要があります。また、この句で指定する整数は、DB_*n*K_CACHE_SIZE パラメータの 1 つの設定と対応している必要があります。

BLOCKSIZE の制限事項： 一時表領域の場合、または表領域を一時表領域として任意のユーザーに割り当てる場合は、標準以外のブロック・サイズを指定できません。

参照： DB_*n*K_CACHE_SIZE パラメータの詳細は、『Oracle Database リファレンス』を参照してください。マルチ・ブロック・サイズの詳細は、『Oracle Database 概要』を参照してください。

logging_clause

表領域内のすべての表、索引、マテリアライズド・ビュー、マテリアライズド・ビュー・ログおよびパーティションのデフォルトのロギング属性を指定します。デフォルトは LOGGING です。この句は、一時表領域または UNDO 表領域では無効です。

表レベル、索引レベル、マテリアライズド・ビュー・レベル、マテリアライズド・ビュー・ログ・レベル、パーティション・レベルでのロギングを指定することで、表領域レベルのロギング属性を上書きできます。

参照： この句の詳細は、8-34 ページの「logging_clause」を参照してください。

FORCE LOGGING

この句を使用すると、表領域は FORCE LOGGING モードになります。一時セグメントへの変更を除き、表領域内のすべてのオブジェクトに対するすべての変更が記録され、個々のオブジェクトの NOLOGGING 設定が上書きされます。データベースをオープンし、READ WRITE モードにしておく必要があります。

この設定によって、NOLOGGING 属性が削除されることはありません。FORCE LOGGING と NOLOGGING の両方を指定できます。この場合、後から表領域に作成されるオブジェクトのデフォルト・ロギング・モードは NOLOGGING になりますが、表領域またはデータベースが FORCE LOGGING モードの場合、このデフォルト値は無視されます。この表領域で FORCE LOGGING モードを無効にすると、デフォルト値である NOLOGGING が再度適用されます。

注意： FORCE LOGGING モードは、パフォーマンスに影響する場合があります。この設定の使用方法の詳細は、『Oracle Database 管理者ガイド』を参照してください。

強制ロギングの制限事項： FORCE LOGGING は、UNDO 表領域および一時表領域に対して指定できません。

ENCRYPTION 句

この句を使用すると、表領域の暗号化プロパティを指定できます。実際は、この句によって表領域が暗号化されるわけではありません。表領域を暗号化するには、この文の DEFAULT *storage_clause* の一部として ENCRYPT キーワードも指定する必要があります。また、ALTER SYSTEM ... SET ENCRYPTION WALLET 句を使用して、サーバー・ウォレットからデータベース・アクセスのメモリーにすでに情報をロードしている必要があります。詳細は、11-61 ページの「SET ENCRYPTION WALLET 句」を参照してください。

注意： 暗号化を使用して作成された表領域は復号化できません。暗号化されていない表領域を作成し、暗号化されていない表領域にデータベース・オブジェクトを再作成する必要があります。

暗号化プロパティは、*encryption_spec* で指定されます。表領域の暗号化に関連する *encryption_spec* の句は、USING 句のみです。USING '*encrypt_algorithm*' を指定すると、使用するアルゴリズムの名前を指定できます。有効なアルゴリズムは、3DES168、AES128、AES192 および AES256 です。この句を省略すると、AES128 が使用されます。

参照： 「暗号化された表領域の作成例：」（16-84 ページ）

DEFAULT *storage_clause*

この句を使用すると、表領域内に作成されるすべてのオブジェクトに対するデフォルトの記憶域パラメータ、および表領域内に作成されるすべての表に対するデフォルトのデータ圧縮を指定できます。この句は、一時表領域では無効です。

ディクショナリ管理表領域の場合、この句で指定できる記憶域パラメータは COMPRESS のみです。

参照：

- 記憶域パラメータの詳細は、8-41 ページの「storage_clause」を参照してください。表の圧縮の詳細は、16-31 ページの「CREATE TABLE」の「table_compression」を参照してください。
- 「基本的な一時表領域の作成例：」16-83 ページ

ONLINE | OFFLINE 句

この句を使用すると、表領域がオンラインまたはオフラインのいずれであるかを決定できます。この句は、一時表領域では無効です。

ONLINE ONLINE を指定すると、表領域に対するアクセス権限を付与されているユーザーに対して、作成直後の表領域を使用可能にできます。これはデフォルトです。

OFFLINE OFFLINE を指定すると、作成直後の表領域を使用禁止にできます。

データ・ディクショナリ・ビュー DBA_TABLESPACES は、各表領域がオンラインまたはオフラインのいずれであるかを示します。

extent_management_clause

extent_management_clause を使用すると、表領域のエクステントの管理方法を指定できます。

> **注意：** この句でエクステントの管理を指定した後は、表領域を移行しないかぎり、エクステントの管理を変更できません。

- LOCAL を指定すると、表領域をローカルで管理できます。ローカル管理表領域では、ビットマップ用に表領域の一部を確保します。これは、永続表領域のデフォルトです。一時表領域は、自動的にローカル管理のエクステントで作成されます。
  - AUTOALLOCATE を指定すると、表領域がシステム管理されます。ユーザーはエクステント・サイズを指定できません。AUTOALLOCATE は、一時表領域に対して指定できません。
  - UNIFORM を指定すると、表領域を SIZE バイトの均一のエクステントで管理できます。SIZE のデフォルト値は 1MB です。一時表領域のすべてのエクステントはサイズが均一であるため、このキーワードは一時表領域ではオプションです。ただし、SIZE を指定する場合は、UNIFORM を指定する必要があります。UNDO 表領域に UNIFORM を指定することはできません。
- DICTIONARY を指定すると、ディクショナリ表を使用して表領域を管理できます。

ディクショナリ管理表領域の制限事項： データベースの SYSTEM 表領域がローカル管理される場合、または *temporary_tablespace_clause* が指定されている場合、DICTIONARY は指定できません。

> **注意：** ローカル管理表領域のみを作成することをお薦めします。ローカル管理表領域は、ディクショナリ管理表領域よりも効率的に管理できます。ディクショナリ管理表領域の新規作成は、サポートされなくなる予定です。

extent_management_clause を指定しない場合、MINIMUM EXTENT 句および DEFAULT *storage_clause* が解析され、エクステント管理が判断されます。

- DEFAULT *storage_clause* を指定しない場合、ローカル管理の自動割当て表領域が作成されます。
- DEFAULT *storage_clause* を指定する場合は、次のようになります。
  - MINIMUM EXTENT 句を指定した場合、MINIMUM EXTENT、INITIAL および NEXT の値が等しく、PCTINCREASE の値が 0（ゼロ）であるかどうかが評価されます。これらの値が同じである場合、エクステントのサイズが INITIAL で、ローカル管理される均一な表領域が作成されます。MINIMUM EXTENT、INITIAL および NEXT パラメータが等しくない場合または PCTINCREASE が 0（ゼロ）でない場合は、指定したエクステントの記憶域パラメータが無視され、ローカル管理の自動割当て表領域が作成されます。

– MINIMUM EXTENT 句を指定しなかった場合、INITIAL および NEXT の記憶域の値が等しく、PCTINCREASE が 0（ゼロ）であるかどうかが評価されます。これらの値が同じである場合、表領域はローカル管理され、エクステント・サイズは均一です。そうでない場合、表領域はローカル管理され、エクステント・サイズは自動割当てされます。

参照： ローカル管理表領域については、『Oracle Database 概要』を参照してください。

エクステント管理の制限事項： エクステント管理には、次の制限事項があります。

- 永続的なローカル管理表領域には、永続オブジェクトのみを格納できます。ローカル管理表領域に一時オブジェクトを格納する必要がある場合（たとえば、ユーザーの一時表領域に割り当てる場合）は、*temporary_tablespace_clause* を使用します。
- LOCAL を指定する場合、DEFAULT *storage_clause*、MINIMUM EXTENT または *temporary_tablespace_clause* は指定できません。

参照： 表領域の移行によってエクステントの管理を変更する場合は、『Oracle Database 管理者ガイド』および 16-83 ページの「ローカル管理表領域の作成例 :」を参照してください。

segment_management_clause

segment_management_clause は、永続的なローカル管理表領域に対してのみ有効です。Oracle Database が、空きリストまたはビットマップのどちらを使用して、表領域のセグメントにある使用済領域および空き領域を追跡するかを指定できます。この句は、一時表領域では無効です。

AUTO AUTO を指定すると、ビットマップを使用して表領域のセグメントにある空き領域を管理できます。AUTO を指定すると、この表領域のオブジェクトに対して後で指定する記憶域の PCTUSED、FREELIST および FREELIST GROUPS の値は無視されます。この設定を**自動セグメント領域管理**といい、これがデフォルトです。

MANUAL MANUAL を指定すると、空きリストを使用して表領域のセグメントにある空き領域を管理できます。この設定は使用しないようにして、自動セグメント領域の表領域を作成することを強くお薦めします。

既存の表領域のセグメント管理を確認するには、DBA_TABLESPACES または USER_TABLESPACES データ・ディクショナリ・ビューの SEGMENT_SPACE_MANAGEMENT 列を問い合せます。

注意： AUTO セグメント管理を指定する場合、次のことに注意します。

- エクステント管理を LOCAL UNIFORM に設定する場合は、各エクステントに 5 以上のデータベース・ブロックがあることを確認する必要があります。
- エクステント管理を LOCAL AUTOALLOCATE に設定する場合、およびデータベースのブロック・サイズが 16KB 以上の場合は、最小で 5 ブロック（64KB に切り上げられる）のエクステントが作成され、セグメント領域管理が管理されます。

自動セグメント領域管理の制限事項： この句には、次の制限事項があります。

- この句は、永続的なローカル管理表領域に対してのみ指定できます。
- この句は、SYSTEM 表領域に対して指定できません。

参照：

- 自動セグメント領域管理およびその使用については、『Oracle Database ストレージ管理者ガイド』を参照してください。
- データ・ディクショナリ・ビューの詳細は、『Oracle Database リファレンス』を参照してください。
- 「表領域に対してセグメント領域管理を指定する場合の例 :」（16-84 ページ）

flashback_mode_clause

この句を ALTER DATABASE FLASHBACK 句と組み合せて使用すると、FLASHBACK DATABASE 操作に表領域を使用できるかどうかを指定できます。この句は、データベースが FLASHBACK モードでオープンされているときに、この表領域のフラッシュバック・ログ・データを保持しない場合に便利です。

この句は、一時表領域または UNDO 表領域では無効です。

FLASHBACK ON FLASHBACK ON を指定すると、表領域で FLASHBACK モードを有効にできます。この表領域のフラッシュバック・ログ・データが保存され、FLASHBACK DATABASE 操作でこの表領域を使用できるようになります。*flashback_mode_clause* を指定しない場合、デフォルトで FLASHBACK ON が指定されます。

FLASHBACK OFF FLASHBACK OFF を指定すると、表領域で FLASHBACK モードを無効にできます。この表領域のフラッシュバック・ログ・データは保存されません。FLASHBACK DATABASE 操作の実行前に、この表領域のデータ・ファイルをオフラインにするか、または削除する必要があります。または、表領域全体をオフラインにすることもできます。どちらの場合も、既存のフラッシュバック・ログは削除されません。

注意： 表領域の FLASHBACK モードは、個々の表の FLASHBACK モードに依存しません。

参照：

- Oracle Flashback Database の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。
- データベース全体を FLASHBACK モードに設定し、データベースを以前のバージョンに戻す方法の詳細は、10-9 ページの「ALTER DATABASE」および 18-22 ページの「FLASHBACK DATABASE」を参照してください。
- 18-25 ページの「FLASHBACK TABLE」および 19-15 ページの「flashback_query_clause」を参照してください。

temporary_tablespace_clause

この句を使用すると、ローカル管理一時表領域を作成できます。一時表領域は、一時データを含むことができるデータベース内の領域の割当てで、この一時データはセッションの存続期間中のみ保持されます。プロセスまたはインスタンスに障害が発生した場合、この一時データをリカバリすることはできません。

一時データとは、一時表などのユーザー生成スキーマ・オブジェクト、またはハッシュ結合およびソート操作で使用される一時領域などのシステム生成データです。一時表領域、またはこの表領域が含まれる表領域グループを特定のユーザーに割り当てると、このユーザーによって開始されるトランザクションでのソート操作にこの表領域が使用されます。

TEMPFILE 句の詳細は、16-75 ページの「DATAFILE | TEMPFILE 句」を参照してください。*extent_management_clause* の詳細は、16-78 ページの「extent_management_clause」を参照してください。

参照： ユーザーに対する一時表領域の割当ての詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。

tablespace_group_clause

この句は、一時表領域に対してのみ有効です。この句を使用すると、*tablespace* が表領域グループに含まれるかどうかを決定できます。表領域グループを使用すると、複数の一時表領域を 1 人のユーザーに割り当て、一時表領域のアクセス性を向上できます。

- グループ名を指定すると、*tablespace* がその表領域グループのメンバーであることを示すことができます。グループ名に *tablespace* または他の既存の表領域と同じ名前を指定することはできません。表領域グループがすでに存在する場合、そのグループに新しい表領域が追加されます。表領域グループが存在しない場合、グループが作成され、そのグループに新しい表領域が追加されます。
- 空の文字列（''）を指定すると、*tablespace* がいずれの表領域グループにも属さないことを示すことができます。

参照：

- 表領域グループへの表領域の追加の詳細は、12-82 ページの「ALTER TABLESPACE」および 16-83 ページの「表領域グループへの一時表領域の追加例 :」を参照してください。
- ユーザーに対する一時表領域の割当てについては、17-7 ページの「CREATE USER」を参照してください。
- 表領域グループの詳細は、『Oracle Database 管理者ガイド』を参照してください。

一時表領域の制限事項： 一時表領域に格納されたデータは、セッションの存続期間中のみ保持されます。そのため、CREATE TABLESPACE 句のサブセットのみが一時表領域に対して有効です。一時表領域に対して指定できるのは、TEMPFILE 句、*tablespace_group_clause* および *extent_management_clause* のみです。

undo_tablespace_clause

UNDO を指定すると、UNDO 表領域を作成できます。自動 UNDO 管理モードでデータベースを実行する場合、Oracle Database は、ロールバック・セグメントのかわりに UNDO 表領域を使用して UNDO 領域を管理します。この句は、自動 UNDO 管理モードで作成しなかったデータベースを自動 UNDO 管理モードで実行している場合に便利です。

自動 UNDO 管理モードでデータベースを起動すると、UNDO 表領域が割り当てられます。UNDO 表領域がインスタンスに割り当てられない場合、SYSTEM ロールバック・セグメントが使用されます。UNDO 表領域の作成によってこれを回避することができ、他の UNDO 表領域がその時点で割り当てられていない場合、インスタンスに暗黙的に割り当てられます。

DATAFILE 句の詳細は、16-75 ページの「DATAFILE | TEMPFILE 句」を参照してください。*extent_management_clause* の詳細は、16-78 ページの「extent_management_clause」を参照してください。

tablespace_retention_clause

この句は、UNDO 表領域に対してのみ有効です。

- RETENTION GUARANTEE を指定すると、*tablespace* のすべての UNDO セグメントの期限切れ前の UNDO データが、これらのセグメントの UNDO 領域を必要とする実行中の操作が失敗する場合でも保持されます。この設定は、Oracle フラッシュバック問合せまたは Oracle フラッシュバック・トランザクション問合せを発行し、データの問題を診断および修正する必要がある場合に便利です。
- RETENTION NOGUARANTEE を指定すると、UNDO 動作を通常の動作に戻すことができます。実行中のトランザクションで必要な場合には、期限切れ前の UNDO データによって使用されている UNDO セグメントの領域を使用できます。これはデフォルトです。

UNDO 表領域の制限事項： UNDO 表領域には、次の制限事項があります。

- この表領域にはデータベース・オブジェクトを作成できません。システム管理の UNDO データ用に確保されています。
- ローカル・エクステント管理を指定するために UNDO 表領域に対して指定できるのは、DATAFILE 句および *extent_management_clause* 句のみです。*extent_management_clause* を使用して、ディクショナリ・エクステント管理を指定できません。すべての UNDO 表領域は、永続的、読取り / 書込み可能およびロギング・モードで作成されます。MINIMUM EXTENT および DEFAULT STORAGE に対する値は、システムで生成されます。

参照：

- 自動 UNDO 管理および UNDO 表領域の詳細は、『Oracle Database 管理者ガイド』を参照してください。UNDO_MANAGEMENT パラメータの詳細は、『Oracle Database リファレンス』を参照してください。
- データベースの作成時に UNDO 表領域を作成する方法は、14-16 ページの「CREATE DATABASE」を参照してください。また、12-82 ページの「ALTER TABLESPACE」および 18-8 ページの「DROP TABLESPACE」を参照してください。
- 「UNDO 表領域の作成例 :」（16-82 ページ）

例

これらの例では、8K ブロックを使用していると想定します。

bigfile 表領域の作成例： 次の例は、10MB のデータ・ファイル bigtbs_f1.dat を持つ bigfile 表領域 bigtbs_01 を作成します。

```
CREATE BIGFILE TABLESPACE bigtbs_01
  DATAFILE 'bigtbs_f1.dat'
  SIZE 20M AUTOEXTEND ON;
```

UNDO 表領域の作成例： 次の例は、10MB の UNDO 表領域 undots1 を作成します。

```
CREATE UNDO TABLESPACE undots1
   DATAFILE 'undotbs_1a.f'
   SIZE 10M AUTOEXTEND ON
   RETENTION GUARANTEE;
```

一時表領域の作成例： 次の文は、サンプル・データベースのデータベース・ユーザーに対するデフォルトの一時表領域を作成します。

```
CREATE TEMPORARY TABLESPACE temp_demo
   TEMPFILE 'temp01.dbf' SIZE 5M AUTOEXTEND ON;
```

デフォルトのデータベース・ブロック・サイズを 2KB とした場合、マップの各ビットは 1 つのエクステントを表し、各ビットは 8000 ブロックをマップします。

次の例は、データ・ファイルを作成するデフォルトの位置を設定し、デフォルトの位置に Oracle Managed Files の一時ファイルを持つ表領域を作成します。一時ファイルは 100MB で、最大サイズが制限なしで自動拡張されます。これらは、Oracle Managed Files のデフォルト値です。

```
ALTER SYSTEM SET DB_CREATE_FILE_DEST = '$ORACLE_HOME/rdbms/dbs';

CREATE TEMPORARY TABLESPACE tbs_05;
```

表領域グループへの一時表領域の追加例： 次の文は、tbs_grp_01 表領域グループに含まれる tbs_temp_02 一時表領域を作成します。この表領域グループが存在しない場合、文の実行中に作成されます。

```
CREATE TEMPORARY TABLESPACE tbs_temp_02
  TEMPFILE 'temp02.dbf' SIZE 5M AUTOEXTEND ON
  TABLESPACE GROUP tbs_grp_01;
```

基本的な一時表領域の作成例： 次の文は、1 つのデータ・ファイルを持つ表領域 tbs_01 を作成します。

```
CREATE TABLESPACE tbs_01
   DATAFILE 'tbs_f2.dat' SIZE 40M
   ONLINE;
```

次の文は、1 つのデータ・ファイルを持つ表領域 tbs_03 を作成し、すべてのエクステントを 500KB の倍数として割り当てます。

```
CREATE TABLESPACE tbs_03
   DATAFILE 'tbs_f03.dbf' SIZE 20M
   LOGGING;
```

表領域の自動拡張を使用可能にする場合の例： 次の文は、1 つのデータ・ファイルを持つ表領域 tbs_02 を作成します。さらに領域が必要な場合、500KB のエクステントが最大サイズ 100MB まで追加されます。

```
CREATE TABLESPACE tbs_02
   DATAFILE 'diskb:tbs_f5.dat' SIZE 500K REUSE
   AUTOEXTEND ON NEXT 500K MAXSIZE 100M;
```

ローカル管理表領域の作成例： 次の文では、データベース・ブロック・サイズが 2KB であると仮定します。

```
CREATE TABLESPACE tbs_04 DATAFILE 'file_1.f' SIZE 10M
   EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K;
```

この文で、すべてのエクステントが 128KB で、ビットマップの各ビットが 64 ブロックを示す、ローカル管理表領域を作成します。

次の文は、均一のエクステントを持つローカル管理表領域を作成して、その表領域に格納された表の例を示します。

```
CREATE TABLESPACE lmt1 DATAFILE 'lmt_file2.f' SIZE 100m REUSE
  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M;

CREATE TABLE lmt_table1 (col1 NUMBER, col2 VARCHAR2(20))
  TABLESPACE lmt1 STORAGE (INITIAL 2m NEXT 1m MINEXTENTS 5 MAXSIZE 100m);
```

セグメントの初期サイズは 5MB です。セグメントには、1MB のエクステントが 5 つ割り当てられています。セグメントの最大サイズは、100MB に制限されています。

次の例は、均一のエクステントを持たないローカル管理表領域を作成します。

```
CREATE TABLESPACE lmt2 DATAFILE 'lmt_file3.f' SIZE 100m REUSE
  EXTENT MANAGEMENT LOCAL;

CREATE TABLE lmt_table2 (col1 NUMBER, col2 VARCHAR2(20))
  TABLESPACE lmt2 STORAGE (INITIAL 2m NEXT 1m MINEXTENTS 5 MAXSIZE 100m);
```

表の初期セグメント・サイズは 5MB です。Oracle Database によって、初期セグメント・サイズを満たすように各エクステントのサイズおよび割り当てられるエクステントの合計数が決定されます。セグメントの最大サイズは、100MB に制限されています。

暗号化された表領域の作成例： 次の文は、暗号化された表領域を作成します。最初にウォレットを開いて、データベースに対して暗号化を有効にする必要があります。

```
ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY "welcome1";

System altered.

CREATE TABLESPACE encrypt_ts
  DATAFILE '$ORACLE_HOME/dbs/encrypt_df.dat' SIZE 1M
  ENCRYPTION USING '3DES168'
  DEFAULT STORAGE (ENCRYPT);

Tablespace created.
```

表領域に対してセグメント領域管理を指定する場合の例： 次の例では、自動セグメント領域管理の表領域を作成します。

```
CREATE TABLESPACE auto_seg_ts DATAFILE 'file_2.f' SIZE 1M
   EXTENT MANAGEMENT LOCAL
   SEGMENT SPACE MANAGEMENT AUTO;
```

Oracle Managed Files の作成例： 次の例は、データ・ファイルを作成するデフォルトの位置を設定し、デフォルトの位置にデータ・ファイルを持つ表領域を作成します。データ・ファイルは 100MB で自動拡張可能であり、最大サイズの制限がありません。

```
ALTER SYSTEM SET DB_CREATE_FILE_DEST = '$ORACLE_HOME/rdbms/dbs';

CREATE TABLESPACE omf_ts1;
```

次の例は、自動拡張されない 100MB の Oracle Managed Files のデータ・ファイルを持つ表領域を作成します。

```
CREATE TABLESPACE omf_ts2 DATAFILE AUTOEXTEND OFF;
```

CREATE TRIGGER

用途

トリガーは PL/SQL を使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は『Oracle Database PL/SQL 言語リファレンス』を参照してください。

CREATE TRIGGER 文を使用すると、**データベース・トリガー**を作成できます。データベース・トリガーとは、次のとおりです。

- 表、スキーマまたはデータベースに対応したストアド PL/SQL ブロック
- 無名 PL/SQL ブロック、あるいは PL/SQL または JAVA で実装されているプロシージャへのコール

指定された条件が発生した場合、トリガーは自動的に実行されます。

前提条件

自分のスキーマ内の表または自分のスキーマ（SCHEMA）に対するトリガーを自分のスキーマ内に作成する場合は、CREATE TRIGGER システム権限が必要です。

任意のスキーマ内の表または別のユーザーのスキーマ（*schema*.SCHEMA）に対するトリガーを任意のスキーマ内に作成する場合は、CREATE ANY TRIGGER システム権限が必要です。

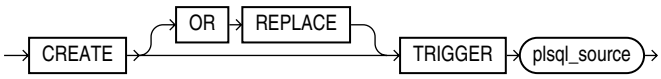
前述の権限の他にも、データベースに対するトリガーを作成する場合は、ADMINISTER DATABASE TRIGGER システム権限が必要です。

トリガーが SQL 文を発行、またはプロシージャやファンクションをコールする場合、そのトリガーの所有者には、これらの操作を行うための権限が必要です。これらの権限はロールを介して付与するのでなく、所有者に直接付与する必要があります。

構文

トリガーは PL/SQL を使用して定義されます。このため、このマニュアルの構文図では SQL キーワードのみを示します。PL/SQL の構文、セマンティクスおよび例については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

***create_trigger*::=**



（*plsql_source* については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。）

セマンティクス

OR REPLACE

OR REPLACE を指定すると、既存のトリガーを再作成できます。この句を指定すると、既存のトリガーの定義を削除しなくても変更できます。

plsql_source

plsql_source の構文およびセマンティクスについては、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

17

SQL 文 : CREATE TYPE ～ DROP ROLLBACK SEGMENT

この章では、次の SQL 文について説明します。

- CREATE TYPE
- CREATE TYPE BODY
- CREATE USER
- CREATE VIEW
- DELETE
- DISASSOCIATE STATISTICS
- DROP CLUSTER
- DROP CONTEXT
- DROP DATABASE
- DROP DATABASE LINK
- DROP DIMENSION
- DROP DIRECTORY
- DROP DISKGROUP
- DROP FLASHBACK ARCHIVE
- DROP FUNCTION
- DROP INDEX
- DROP INDEXTYPE
- DROP JAVA
- DROP LIBRARY
- DROP MATERIALIZED VIEW
- DROP MATERIALIZED VIEW LOG
- DROP OPERATOR
- DROP OUTLINE
- DROP PACKAGE

- DROP PROCEDURE
- DROP PROFILE
- DROP RESTORE POINT
- DROP ROLE
- DROP ROLLBACK SEGMENT

CREATE TYPE

用途

オブジェクト型は PL/SQL を使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

CREATE TYPE 文を使用すると、**オブジェクト型**、**SQLJ オブジェクト型**、名前付きの可変配列（**VARRAY**）、**ネストした表型**または**不完全なオブジェクト型**の仕様部を作成できます。CREATE TYPE 文および CREATE TYPE BODY 文を使用してオブジェクト型を作成します。CREATE TYPE 文では、オブジェクト型の名前、オブジェクトの属性、メソッドおよびその他のプロパティを指定します。CREATE TYPE BODY 文には、その型を実装するメソッドに対するコードが含まれます。

注意 :

- 型仕様で属性のみ宣言し、メソッドは宣言しないオブジェクト型を作成する場合は、型本体を指定する必要はありません。
- SQLJ オブジェクト型を作成する場合は、型本体を指定できません。型の実装は Java クラスとして指定されます。

不完全型とは、フォワード型定義によって作成される型です。このオブジェクト型には名前はありますが、属性およびメソッドがないため、不完全といわれます。他の型からの参照が可能なため、互いに参照する型の定義に使用できます。ただし、不完全オブジェクト型を使用して表やオブジェクト列またはネストした表型の列を作成する場合は、型を完全に指定しておく必要があります。

参照 :

- 型のメンバー・メソッドの作成については、17-5 ページの「CREATE TYPE BODY」を参照してください。
- オブジェクト、不完全型、VARRAY およびネストした表の詳細は、『Oracle Database PL/SQL 言語リファレンス』および『Oracle Database オブジェクト・リレーショナル開発者ガイド』を参照してください。

前提条件

自分のスキーマ内に型を作成する場合は、CREATE TYPE システム権限が必要です。他のユーザーのスキーマ内に型を作成する場合は、CREATE ANY TYPE システム権限が必要です。これらの権限は、明示的に取得することもロールを介して取得することもできます。

サブタイプを作成する場合は、UNDER ANY TYPE システム権限またはスーパータイプに対する UNDER オブジェクト権限が必要です。

型の所有者には、型定義内で参照する他のすべての型にアクセスするための EXECUTE オブジェクト権限が必要です。または、EXECUTE ANY TYPE システム権限が必要です。所有者は、これらの権限をロールを介して取得することはできません。

型の所有者が、型にアクセスする権限を他のユーザーに付与する場合、所有者には、参照型に対する Grant Option 付きの EXECUTE オブジェクト権限、または Admin Option 付きの EXECUTE ANY TYPE システム権限が必要です。これらの権限がないと、型の所有者は、型にアクセスする権限を他のユーザーに付与できません。

構文

型は PL/SQL を使用して定義されます。このため、このマニュアルの構文図では SQL キーワードのみを示します。PL/SQL の構文、セマンティクスおよび例については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

***create_type*::=**

（*plsql_source* については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。）

セマンティクス

OR REPLACE

OR REPLACE を指定すると、既存の型を再作成できます。この句を指定した場合、既存の型の定義をはじめに削除しなくても、その定義を変更できます。

再作成するオブジェクト型に対する権限があらかじめ付与されている場合は、再作成後にあらためて権限を付与されなくてもそのオブジェクト型を使用および参照できます。

ファンクション索引が型に依存している場合、索引に DISABLED のマークが付きます。

plsql_source

plsql_source の構文およびセマンティクスについては、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

CREATE TYPE BODY

用途

型本体は PL/SQL を使用して定義されます。このため、この項では一般的な情報について説明します。構文およびセマンティクスの詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

CREATE TYPE BODY を使用すると、オブジェクト型仕様部で定義されたメンバー・メソッドを定義または実装することができます。CREATE TYPE 文および CREATE TYPE BODY 文を使用してオブジェクト型を作成します。CREATE TYPE 文では、オブジェクト型の名前、オブジェクトの属性、メソッドおよびその他のプロパティを指定します。CREATE TYPE BODY 文には、その型を実装するメソッドに対するコードが含まれます。

call_spec を定義していないオブジェクト型仕様部に指定された各メソッドには、オブジェクト型本体の対応するメソッド本体を指定する必要があります。

注意： SQLJ オブジェクト型を作成する場合は、Java クラスとして指定されます。

参照：

- 型仕様部の作成の詳細は、17-3 ページの「CREATE TYPE」を参照してください。
- 型仕様部の変更の詳細は、13-4 ページの「ALTER TYPE」を参照してください。

前提条件

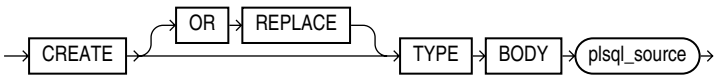
オブジェクト型用の CREATE TYPE 仕様部で行われるすべてのメンバー宣言には、それに対応する構造が CREATE TYPE 文または CREATE TYPE BODY 文内に存在する必要があります。

自分のスキーマ内で型本体を作成または再作成する場合は、CREATE TYPE システム権限または CREATE ANY TYPE システム権限が必要です。他のユーザーのスキーマ内でオブジェクト型を作成する場合は、CREATE ANY TYPE システム権限が必要です。他のユーザーのスキーマ内でオブジェクト型を置換する場合は、DROP ANY TYPE システム権限が必要です。

構文

型本体は PL/SQL を使用して定義されます。このため、このマニュアルの構文図では SQL キーワードのみを示します。PL/SQL の構文、セマンティクスおよび例については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

***create_type_body*::=**



（*plsql_source* については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。）

セマンティクス

OR REPLACE

OR REPLACE を指定すると、既存の型本体を再作成できます。この句を指定した場合、既存の型本体の定義をはじめに削除しなくても、その定義を変更できます。

再作成されたオブジェクト型本体に対する権限を付与されているユーザーは、権限を再付与されなくても、そのオブジェクト型本体を使用および参照できます。

この句を使用した場合、ALTER TYPE ... REPLACE 文によって追加された仕様部に、新規メンバー・サブプログラム定義を追加できます。

plsql_source

plsql_source の構文およびセマンティクスについては、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

CREATE USER

用途

CREATE USER 文を使用すると、データベース・**ユーザー**（データベースにログイン可能なアカウント）を作成および構成でき、Oracle Database がそのユーザーによるアクセスを許可する方法が確立されます。

この文を自動ストレージ管理クラスタで発行すると、現行のノードの ASM インスタンスに対してローカルなパスワード・ファイルに、ユーザーおよびパスワードの組合せを追加できます。各ノードの ASM インスタンスでは、この文を使用して個々のパスワード・ファイルを更新できます。このパスワード・ファイル自体は、ORAPWD ユーティリティで作成されている必要があります。

プロキシ・アプリケーションまたはアプリケーション・サーバーによって、データベースとユーザーを接続できます。構文および説明は、13-5 ページの「ALTER USER」を参照してください。

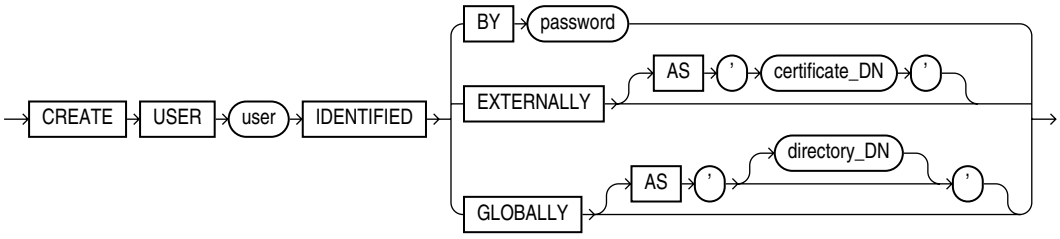
前提条件

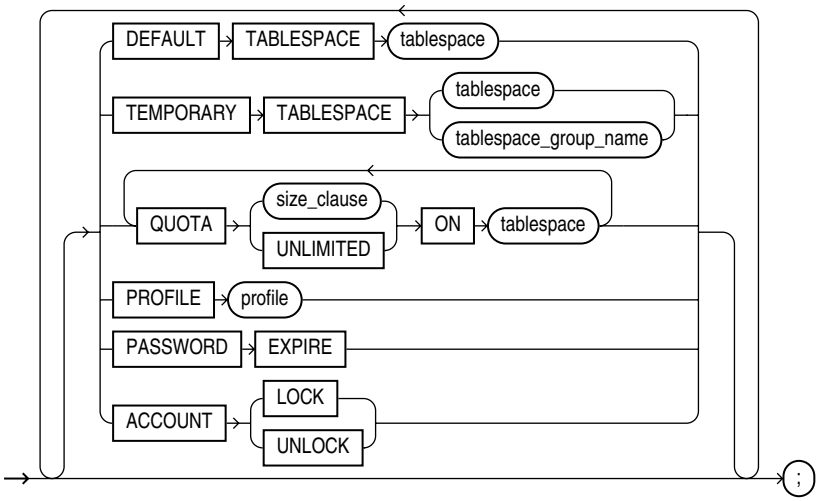
CREATE USER システム権限が必要です。CREATE USER 文を使用して作成したユーザーの権限ドメインは空（権限を付与されていない状態）になります。Oracle Database にログオンするユーザーには、CREATE SESSION システム権限が必要です。そのため、ユーザーを作成した場合、必ず CREATE SESSION システム権限をそのユーザーに付与してください。詳細は、18-31 ページの「GRANT」を参照してください。

AS SYSASM として認証されたユーザーのみが、このコマンドを発行して、自動ストレージ管理インスタンスのパスワード・ファイルを変更できます。

構文

***create_user*::=**





（8-42 ページの size_clause::= を参照）

セマンティクス

user

作成するユーザー名を指定します。この名前には、使用しているデータベース・キャラクタ・セットの文字のみを指定できます。また、2-98 ページの「スキーマ・オブジェクトのネーミング規則」で説明した規則に従う必要があります。データベース・キャラクタ・セットがマルチバイト文字を含んでいても、ユーザー名にはシングルバイト文字を 1 つ以上使用することをお薦めします。

注意： ユーザー名およびパスワードは、ご使用のプラットフォームに応じて、ASCII または EBCDIC 文字のみでエンコードすることをお薦めします。

参照： 「データベース・ユーザーの作成例 :」（17-11 ページ）

IDENTIFIED 句

IDENTIFIED 句を使用すると、Oracle Database によるユーザーの認証方法を指定できます。

BY password

BY *password* 句を使用すると、**ローカル・ユーザー**を作成し、データベースへのログイン時に、パスワードの指定が必要であることを指定できます。パスワードは、大 / 小文字が区別されます。この後に、ユーザーをデータベースに接続するために使用される CONNECT 文字列では、この CREATE USER 文または後述の ALTER USER 文で使用されているものと同じ文字（大文字、小文字または混在）を使用してパスワードを指定する必要があります。パスワードには、データベース・キャラクタ・セットのシングルバイト文字、マルチバイト文字、特殊文字またはこれらの組合せを含めることができます。

参照： パスワードでの大 / 小文字の区別、パスワードの複雑さ、その他パスワードに関するガイドラインについては、『Oracle Database セキュリティ・ガイド』を参照してください。

Oracle Database の複雑なパスワード検証ルーチンを使用していない場合、パスワードは、2-98 ページの「スキーマ・オブジェクトのネーミング規則」にある規則に従う必要があります。そのルーチンでは、通常のネーミング規則より複雑な文字の組合せが必要です。このルーチンは、UTLPWDMG.SQL スクリプトを使用して実装します。詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。

注意： ユーザー名およびパスワードは、ご使用のプラットフォームに応じて、ASCII または EBCDIC 文字のみでエンコードすることをお薦めします。

参照： パスワード管理およびパスワード保護の詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。

EXTERNALLY 句

EXTERNALLY を指定すると、**外部ユーザー**を作成できます。このユーザーは、外部サービス（オペレーティング・システムやサード・パーティのサービスなど）で認証する必要があります。その場合、オペレーティング・システムまたはサード・パーティ・サービスによる認証を使用して、特定の外部ユーザーが特定のデータベース・ユーザーへのアクセス権を所有することが可能になります。

AS '*certificate_DN*' この句は SSL 認証の外部ユーザーに必須で、そのユーザーに対してのみ使用されます。*certificate_DN* は、ユーザーのウォレット内のユーザーの PKI 証明書にある識別名です。

注意： ご使用のオペレーティング・システムにログインする際のセキュリティが十分でない場合は、IDENTIFIED EXTERNALLY を使用しないことをお薦めします。

参照：

- 外部で識別されるユーザーの詳細は、『Oracle Database エンタープライズ・ユーザー・セキュリティ管理者ガイド』を参照してください。
- 「外部データベース・ユーザーの作成例：」（17-11 ページ）

GLOBALLY 句

GLOBALLY 句を使用すると、**グローバル・ユーザー**を作成することができます。このユーザーは、エンタープライズ・ディレクトリ・サービス（Oracle Internet Directory）によって認可される必要があります。

directory_DN 文字列は、次のいずれかの形式になります。

- このユーザーを識別するエンタープライズ・ディレクトリ・サービスの X.509 の名前。文字列は、CN=*username,other_attributes* という形式である必要があります。*other_attributes* は、ディレクトリ内のユーザーの識別名（DN）以外の部分です。この形式では、**プライベート・グローバル・スキーマ**が作成されます。
- NULL 文字列（' '）。エンタープライズ・ディレクトリ・サービスが、認証されたグローバル・ユーザーを、このデータベース・スキーマに適切なロール付きでマップすることを示します。この形式は、GLOBALLY キーワードのみを指定するのと同じで、**共有グローバル・スキーマ**が作成されます。

特定のユーザーとして接続し、ALTER USER 文を使用してユーザーのロールをアクティブにするために、アプリケーション・サーバーの機能を制御できます。

参照：

- グローバル・ユーザーの詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。
- 「ALTER USER」（13-5 ページ）
- 「グローバル・データベース・ユーザーの作成例：」（17-12 ページ）

DEFAULT TABLESPACE 句

ユーザーが作成するオブジェクトを格納するデフォルトの表領域を指定します。この句を省略した場合、ユーザーのオブジェクトはデータベースのデフォルトの表領域に格納されます。データベースのデフォルトの表領域が指定されていない場合、ユーザーのオブジェクトは SYSTEM 表領域に格納されます。

デフォルトの表領域の制限事項： ローカル管理の一時表領域（UNDO 表領域を含む）またはディクショナリ管理の一時表領域は、ユーザーのデフォルトの表領域として指定できません。

参照：

- 表領域の概要および UNDO 表領域の詳細は、16-71 ページの「CREATE TABLESPACE」を参照してください。
- デフォルトの表領域をユーザーに割り当てる方法の詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。

TEMPORARY TABLESPACE 句

ユーザーの一時セグメントが確保される表領域または表領域グループを指定します。この句を省略した場合、ユーザーの一時セグメントはデータベースのデフォルトの一時表領域に格納されます。データベースのデフォルトの一時表領域が指定されていない場合は、SYSTEM 表領域に格納されます。

- *tablespace* に、ユーザーの一時セグメント表領域を指定します。
- *tablespace_group_name* に、ユーザーが一時セグメントを保存できる表領域の表領域グループを指定します。

一時表領域の制限事項 : この句には、次の制限事項があります。

- 表領域は一時表領域で、標準ブロック・サイズである必要があります。
- 表領域は、UNDO 表領域または自動セグメント領域管理の表領域にできません。

参照 :

- 表領域グループの詳細は、『Oracle Database 管理者ガイド』を参照してください。一時表領域をユーザーに割り当てる方法の詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。
- UNDO 表領域およびセグメント管理の詳細は、16-71 ページの「CREATE TABLESPACE」を参照してください。
- 「表領域グループの割当て例 :」（13-10 ページ）

QUOTA 句

QUOTA 句を使用すると、ユーザーが表領域内に割り当てることができる最大サイズを指定できます。

CREATE USER 文では、複数の表領域に対して複数の QUOTA 句を指定できます。

UNLIMITED を使用すると、ユーザーは、表領域の領域を無制限に割り当てることができます。

QUOTA 句の制限事項 : この句は、一時表領域には指定できません。

参照 : この句の詳細は、8-42 ページの「size_clause」を参照してください。表領域の割当て制限の指定については、『Oracle Database セキュリティ・ガイド』を参照してください。

PROFILE 句

ユーザーに割り当てるプロファイルを指定します。このプロファイルによって、ユーザーが使用できるデータベース・リソース容量が制限されます。この句を省略した場合、DEFAULT プロファイルがユーザーに割り当てられます。

注意 : データベース・リソース制限を設定する場合、この SQL プロファイルではなく、データベース・リソース・マネージャを使用することをお薦めします。データベース・リソース・マネージャを使用すると、リソース使用の管理および監査を柔軟に行うことができます。データベース・リソース・マネージャの詳細は、『Oracle Database 管理者ガイド』を参照してください。

参照 : 18-31 ページの「GRANT」および 15-47 ページの「CREATE PROFILE」を参照してください。

PASSWORD EXPIRE 句

PASSWORD EXPIRE を指定すると、ユーザーのパスワードを期限切れにできます。この設定によって、ユーザーがデータベースにログインする前に、ユーザーまたは DBA はパスワードの変更が必要となります。

ACCOUNT 句

ACCOUNT LOCK を指定すると、ユーザー・アカウントをロックし、アクセスを禁止できます。ACCOUNT UNLOCK を指定すると、ユーザー・アカウントのロックを解除し、アカウントへのアクセスを許可できます。

例

次のすべての例では、example 表領域を使用します。この表領域はシード・データベースに存在し、サンプル・スキーマにアクセス可能です。

データベース・ユーザーの作成例： PASSWORD EXPIRE を指定して新規ユーザーを作成する場合、データベースにログインする前に、そのユーザーのパスワードを変更する必要があります。次の文は、sidney ユーザーを作成します。

```
CREATE USER sidney
    IDENTIFIED BY out_standing1
    DEFAULT TABLESPACE example
    QUOTA 10M ON example
    TEMPORARY TABLESPACE temp
    QUOTA 5M ON system
    PROFILE app_user
    PASSWORD EXPIRE;
```

前述のユーザー sidney には次の特性があります。

- パスワード out_standing1
- 10MB の割当て制限のあるデフォルト表領域 example
- 一時表領域 temp
- 5MB の割当て制限のある表領域 SYSTEM へのアクセス
- プロファイル app_user（15-51 ページの「プロファイルの作成例 :」で作成）によって定義されているデータベース・リソースの制限
- sidney がデータベースにログインする前に変更する必要がある期限切れのパスワード

外部データベース・ユーザーの作成例： 次の例は、データベースにアクセスする前に外部ソースによって識別される必要のある、外部ユーザーを作成します。

```
CREATE USER app_user1
   IDENTIFIED EXTERNALLY
   DEFAULT TABLESPACE example
   QUOTA 5M ON example
   PROFILE app_user;
```

ユーザー app_user1 には次の追加の特性があります。

- デフォルトの表領域 example
- デフォルトの一時表領域 example
- 表領域 example に 5MB の領域、およびデータベースの一時表領域に無制限の割当て
- プロファイル app_user によって定義されているデータベース・リソースの制限

オペレーティング・システム・アカウントによってのみアクセス可能な別のユーザーを作成する場合、ユーザー名に初期化パラメータ OS_AUTHENT_PREFIX 値の接頭辞を付けます。たとえば、この値が「ops$」の場合、次の文を使用して、外部で識別されるユーザー external_user を作成できます。

```
CREATE USER ops$external_user
   IDENTIFIED EXTERNALLY
   DEFAULT TABLESPACE example
   QUOTA 5M ON example
   PROFILE app_user;
```

グローバル・データベース・ユーザーの作成例： 次の例は、グローバル・ユーザーを作成します。グローバル・ユーザーを作成する場合、エンタープライズ・ディレクトリ・サーバーでユーザーを識別する X.509 の名前を指定することができます。

```
CREATE USER global_user
   IDENTIFIED GLOBALLY AS 'CN=analyst, OU=division1, O=oracle, C=US'
   DEFAULT TABLESPACE example
   QUOTA 5M ON example;
```

CREATE VIEW

用途

CREATE VIEW 文を使用すると、**ビュー**を定義できます。ビューは論理表で、1 つ以上の表またはビューがベースとなります。ビューにデータそのものが格納されているわけではありません。ビューの基礎になる表を**実表**といいます。

LOB、オブジェクト型、REF データ型、ネストした表または VARRAY 型をサポートする**オブジェクト・ビュー**またはリレーショナル・ビューを、既存のビュー・メカニズムで作成することもできます。オブジェクト・ビューとは、ユーザー定義型のビューのことで、ビューの各行に、それぞれが一意のオブジェクト識別子を持つオブジェクトが含まれます。

XMLType ビューも作成できます。このビューは、オブジェクト・ビューと似ていますが、XMLType の XML Schema ベースの表のデータを表示します。

参照 :

- ビューの様々な種類およびその使用方法については、『Oracle Database 概要』、『Oracle Database アドバンスト・アプリケーション開発者ガイド』および『Oracle Database 管理者ガイド』を参照してください。
- XMLType ビューの詳細は、『Oracle XML DB 開発者ガイド』を参照してください。
- ビューの変更およびデータベースからのビューの削除については、13-12 ページの「ALTER VIEW」および 18-17 ページの「DROP VIEW」を参照してください。

前提条件

自分のスキーマ内にビューを作成する場合は、CREATE VIEW システム権限が必要です。他のユーザーのスキーマ内にビューを作成する場合は、CREATE ANY VIEW システム権限が必要です。

サブビューを作成する場合は、UNDER ANY VIEW システム権限またはスーパービューに対する UNDER オブジェクト権限が必要です。

ビューが含まれているスキーマの所有者は、そのビューの基礎となっているすべての表またはビューに対する行の選択、挿入、更新または削除の権限が必要です。また、所有者には、これらの権限がロールを介してではなく、直接付与されている必要があります。

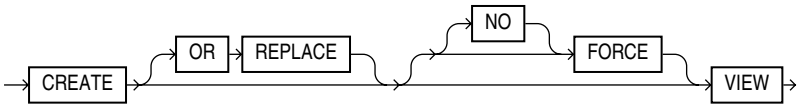
オブジェクト・ビューの作成時にオブジェクト型の基本コンストラクタ・メソッドを使用する場合、次のいずれかの条件を満たしている必要があります。

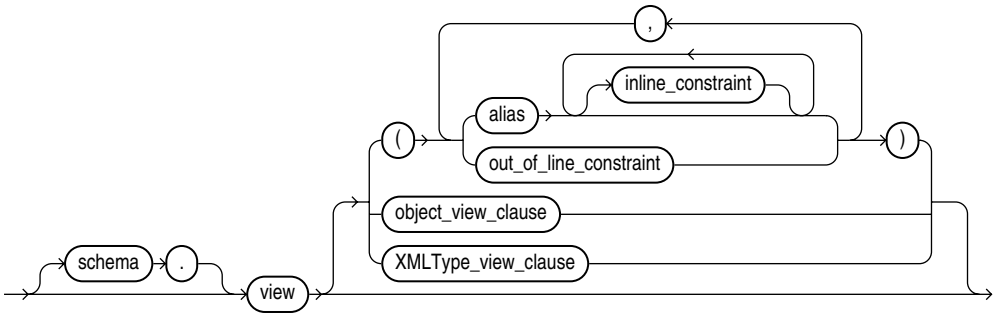
- オブジェクト型が作成対象のビューと同じスキーマに属している。
- EXECUTE ANY TYPE システム権限を持っている。
- そのオブジェクト型に対する EXECUTE オブジェクト権限を持っている。

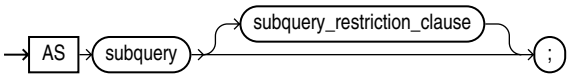
参照 : 作成するビューの基礎となる表またはビューに対して、ビューの所有者に必要な権限の詳細は、19-4 ページの「SELECT」、18-53 ページの「INSERT」、19-62 ページの「UPDATE」および 17-23 ページの「DELETE」を参照してください。

構文

***create_view*::=**

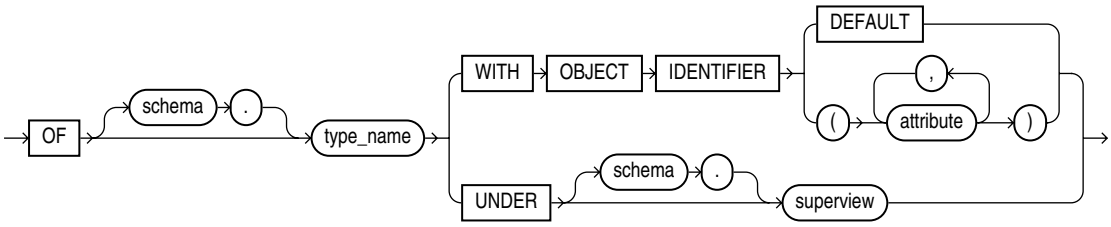


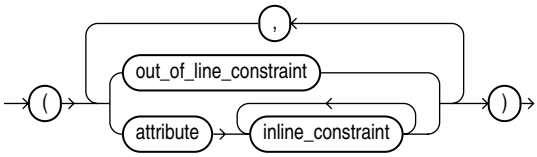




(8-5 ページの inline_constraint::=、8-5 ページの out_of_line_constraint::=、
17-14 ページの object_view_clause::=、17-14 ページの XMLType_view_clause::=、
19-5 ページの subquery::= (SELECT 構文の一部)、
17-15 ページの subquery_restriction_clause::= を参照)

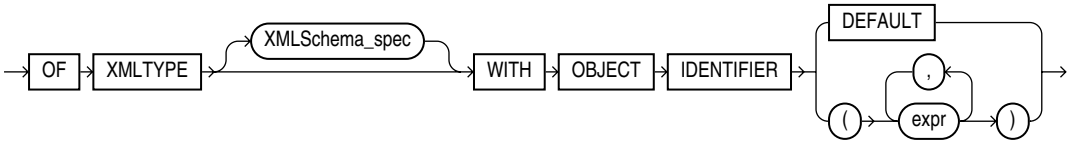
***object_view_clause*::=**



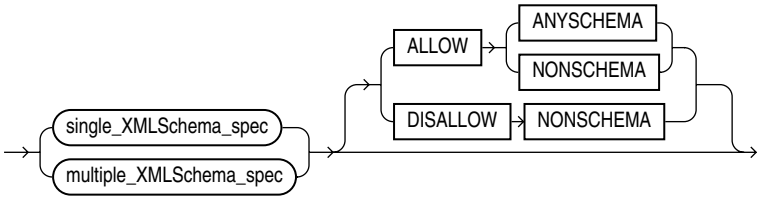


(8-5 ページの inline_constraint::= および 8-5 ページの out_of_line_constraint::= を参照)

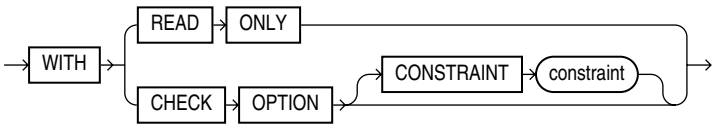
XMLType_view_clause::=



***XMLSchema_spec*::=**



subquery_restriction_clause::=



セマンティクス

OR REPLACE

OR REPLACE を指定すると、既存のビューを再作成できます。この句を使用した場合、以前に付与されたオブジェクト権限を削除、再作成、再付与しなくても、既存のビュー定義を変更できます。

ビューを再作成した場合、ビュー内で定義された INSTEAD OF トリガーが削除されます。

すべてのマテリアライズド・ビューが *view* に依存している場合、そのマテリアライズド・ビューには UNUSABLE というマークが付けられ、それらが再利用可能になるようにリストアする場合は、完全なリフレッシュが必要になります。無効なマテリアライズド・ビューはクエリー・リライトで使用されることはなく、また、再コンパイルされるまでリフレッシュされません。

参照 :

- 無効なマテリアライズド・ビューのリフレッシュについては、11-2 ページの「ALTER MATERIALIZED VIEW」を参照してください。
- マテリアライズド・ビューの概要は、『Oracle Database 概要』を参照してください。
- INSTEAD OF 句の詳細は、16-85 ページの「CREATE TRIGGER」を参照してください。

FORCE

FORCE を指定すると、ビューの実表または参照するオブジェクト型が存在しているか、またはそのビューを含むスキーマの所有者が、それらの表やオブジェクト型に対する権限を持っているかにかかわらず、ビューを作成できます。SELECT、INSERT、UPDATE または DELETE 文をビューに対して発行する場合、これらの条件を満たしている必要があります。

ビュー定義が制約を含む場合、実表または参照するオブジェクト型が存在しないと、CREATE VIEW ... FORCE は正常に実行されません。また、ビュー定義で存在しない制約が指定される場合も、CREATE VIEW ... FORCE は正常に実行されません。

NO FORCE

NOFORCE を指定すると、実表が存在し、ビューを含むスキーマの所有者がその実表に対する権限を持っている場合のみ、ビューを作成できます。これはデフォルトです。

schema

ビューを含めるスキーマを指定します。*schema* を指定しない場合、自分のスキーマにそのビューが作成されます。

view

ビューまたはオブジェクト・ビューの名前を指定します。

ビューの制限事項： あるビューに INSTEAD OF トリガーが含まれている場合、そのビューで作成されるビューが更新可能であっても、そのビューには INSTEAD OF トリガーが必要です。

参照： 「ビューの作成例：」（17-20 ページ）

alias

ビューを定義する問合せによって選択された式の名前を指定します。別名の数は、ビューによって選択された式の数と一致している必要があります。別名は、Oracle Database のスキーマ・オブジェクトのネーミング規則に従う必要があります。別名は、ビュー内で一意である必要があります。

別名を省略した場合、別名は問合せの列または列の別名から導出されます。このため、問合せに列の名前のみでなく式が含まれている場合は、別名を使用する必要があります。また、ビュー定義に制約が含まれている場合も、別名を指定する必要があります。

ビューの別名の制限事項： オブジェクト・ビューの作成時に別名は指定できません。

参照： 「スキーマ・オブジェクトの構文および SQL 文の構成要素」（2-102 ページ）

inline_constraint および *out_of_line_constraint*

ビューおよびオブジェクト・ビューには制約を指定できます。*out_of_line_constraint* 句を使用すると、ビュー・レベルで制約を定義できます。適切な別名の後で *inline_constraint* 句を使用すると、列定義または属性定義の一部として制約を定義できます。

Oracle Database では、ビュー制約を適用していません。制限事項を含むビュー制約の詳細は、8-17 ページの「ビュー制約」を参照してください。

参照： 「制約付きのビューの作成例：」（17-20 ページ）

object_view_clause

object_view_clause を使用すると、オブジェクト型にビューを定義できます。

参照： 「オブジェクト・ビューの作成例：」（17-21 ページ）

OF *type_name* 句

この句を使用すると、*type_name* 型の**オブジェクト・ビュー**を明示的に作成できます。オブジェクト・ビューの列は、*type_name* 型の最上位属性に対応しています。各行にはオブジェクト・インスタンスが含まれ、また、各インスタンスは WITH OBJECT IDENTIFIER 句で指定したオブジェクト識別子に関連付けられます。*schema* を指定しない場合、自分のスキーマ内にオブジェクト・ビューが作成されます。

オブジェクト表、XMLType 表、オブジェクト・ビューおよび XMLType ビューには、列名は付けられません。そのため、システム生成疑似列 OBJECT_ID が定義されます。問合せでこの列名を使用し、WITH OBJECT IDENTIFIER 句を指定して、オブジェクト・ビューを作成できます。

WITH OBJECT IDENTIFIER 句

WITH OBJECT IDENTIFIER 句を使用すると、最上位（ルート）のオブジェクト・ビューを指定できます。オブジェクト・ビュー内の各行を識別するためのキーとして使用されるオブジェクト型の属性を指定します。ほとんどの場合、各属性は実表の主キー列と対応します。属性リストが一意で、ビューの 1 行ずつを識別することを確認する必要があります。

オブジェクト・ビューの制限事項： オブジェクト・ビューには、次の制限事項があります。

- オブジェクト・ビュー内の複数のインスタンスに変換される主キー REF を参照解除または確保しようとした場合、データベースはエラーを戻します。
- サブビューはスーパービューのオブジェクト識別子を継承するため、サブビューを作成する場合は、この句を指定できません。

注意： Oracle8*i* リリース 8.0 の構文 WITH OBJECT OID は、この構文と置き換えられます。キーワード WITH OBJECT OID は下位互換用にサポートされていますが、新しい構文 WITH OBJECT IDENTIFIER を使用することをお薦めします。

オブジェクト・ビューがオブジェクト表またはオブジェクト・ビュー上で定義されている場合は、この句を省略するか、または DEFAULT を指定します。

DEFAULT DEFAULT を指定すると、基礎となるオブジェクト表またはオブジェクト・ビュー固有のオブジェクト識別子を使用して、各行を一意に識別できます。

attribute オブジェクト・ビューに対して作成されるオブジェクト識別子の基になるオブジェクト型の属性を指定します。

UNDER 句

UNDER 句を使用すると、オブジェクト・スーパービューに基づくサブビューを指定できます。

サブビューの制限事項： サブビューには、次の制限事項があります。

- サブビューは、スーパービューと同じスキーマに作成する必要があります。
- オブジェクト型 *type_name* は、*superview* 直属のサブタイプである必要があります。
- 同一のスーパービューには、特定の型のサブビューを 1 つのみ作成できます。

参照：

- オブジェクトの作成の詳細は、17-3 ページの「CREATE TYPE」を参照してください。
- データ・ディクショナリ・ビューの詳細は、『Oracle Database リファレンス』を参照してください。

AS *subquery*

ビューの基になっている表（実表）の列と行を識別する副問合せを指定します。副問合せの SELECT 構文のリストには 1000 以下の式を指定できます。

リモート表およびリモート・ビューを参照するビューを作成する場合、CREATE DATABASE LINK 文の CONNECT TO 句を使用して、指定するデータベース・リンクを作成しておく必要があります。また、ビュー副問合せのスキーマ名で修飾する必要があります。

ビューを定義する問合せで *flashback_query_clause* を使用してビューを作成した場合、AS OF 式は作成時には解析されず、ユーザーがビューを問い合せるたびに解析されます。

参照： Oracle フラッシュバック問合せの詳細は、17-20 ページの「結合ビューの作成例 :」および『Oracle Database アドバンスト・アプリケーション開発者ガイド』を参照してください。

ビューを定義する問合せの制限事項： ビュー問合せには、次の制限事項があります。

- 副問合せでは、CURRVAL および NEXTVAL 疑似列を選択できません。
- 副問合せで ROWID、ROWNUM または LEVEL の各疑似列を選択する場合、そのビュー副問合せには列の別名が必要です。
- 副問合せでアスタリスク（*）を使用して表のすべての列を選択し、後でその表に新しい列を追加する場合、CREATE OR REPLACE VIEW 文を発行してビューを再作成するまで、追加した列はそのビューに含まれません。
- オブジェクト・ビューの場合、副問合せの SELECT 構文のリスト内の要素数は、そのオブジェクト型の最上位属性数と同じである必要があります。それぞれの選択要素のデータ型は、対応する最上位属性と同じである必要があります。
- SAMPLE 句は指定できません。
- *subquery_restriction_clause* も指定する場合は、副問合せで ORDER BY 句を指定できません。

これらの制限は、マテリアライズド・ビューにも適用されます。

更新可能なビューの注意事項： 更新可能なビューには、次の注意事項があります。

更新可能なビューとは、実表の行を挿入、更新または削除できるビューです。更新可能なビューを作成するか、またはビューに INSTEAD OF トリガーを作成して更新可能にすることができます。

更新可能なビューの列を更新する方法を確認するには、USER_UPDATABLE_COLUMNS データ・ディクショナリ・ビューを問い合せます。このビューで表示される情報は、更新可能なビューのみに意味があります。ビューを更新可能にするには、次の条件が満たされている必要があります。

- ビューの各列は、単一表の列にマップする必要があります。たとえば、ビューの列が TABLE 句の出力にマップする場合（コレクション・ネスト解除）、ビューは更新可能ではありません。
- ビューには、次の構造体が含まれていない必要があります。

  集合演算子
  DISTINCT 演算子
  集計ファンクションまたは分析ファンクション
  GROUP BY、ORDER BY、MODEL、CONNECT BY または START WITH 句
  SELECT 構文のリストにあるコレクション式
  SELECT 構文のリストにある副問合せ
  WITH READ ONLY が指定された副問合せ
  結合（一部の例外を除く。詳細は、『Oracle Database 管理者ガイド』を参照してください。）
- 更新可能なビューが疑似列または式を含む場合は、これらの疑似列または式を参照する UPDATE 文を使用して、実表の行を更新できません。
- 結合ビューを更新可能にする場合、次のすべての条件が満たされている必要があります。
  - DML 文は、結合の基になる 1 つの表のみに影響する。
  - INSERT 文で、WITH CHECK OPTION を指定してビューを作成しておらず、値が挿入されたすべての列が**キー保存表**の列である。実表の各主キーまたは一意キーの値に対するキー保存表は、結合ビューで一意である。
  - UPDATE 文で、WITH CHECK OPTION を指定してビューを作成しておらず、更新されるすべての列はキー保存表から抽出されている。
- DELETE 文の場合、結合によって複数のキー保存表が作成されると、ビューが WITH CHECK OPTION を指定して作成されたかどうかにかかわらず、FROM 句に指定された最初の表から削除されます。

参照 :

- 更新可能なビューの詳細は、『Oracle Database 管理者ガイド』を参照してください。
- 更新可能な結合ビューおよびキー保存表の例については、17-20 ページの「更新可能なビューの作成例 :」および 17-20 ページの「結合ビューの作成例 :」を参照してください。更新できないビューの INSTEAD OF トリガーの例については、『Oracle Database PL/SQL 言語リファレンス』のを参照してください。

XMLType_view_clause

この句を使用すると、XMLType ビューを作成できます。このビューには、XMLType の XML Schema ベースの表のデータが表示されます。*XMLSchema_spec* は、XML データを対応するオブジェクト・リレーショナル・データにマップするために使用する XML Schema を示します。XML Schema は、XMLType ビューを作成する前に作成しておく必要があります。

オブジェクト表、XMLType 表、オブジェクト・ビューおよび XMLType ビューには、列名は付けられません。そのため、システム生成疑似列 OBJECT_ID が定義されます。問合せでこの列名を使用し、WITH OBJECT IDENTIFIER 句を指定して、オブジェクト・ビューを作成できます。

参照 :

- XMLType ビューおよび XMLSchema の詳細は、『Oracle XML DB 開発者ガイド』を参照してください。
- 17-22 ページの「XMLType ビューの作成例 :」および 17-22 ページの「XMLType 表のビューの作成例 :」を参照してください。

subquery_restriction_clause

subquery_restriction_clause を使用すると、次のいずれかの方法で、ビューを定義する問合せを制限できます。

WITH READ ONLY WITH READ ONLY を指定すると、表またはビューを更新禁止にできます。

WITH CHECK OPTION WITH CHECK OPTION を指定すると、副問合せに含まれない行を生成する表またはビューの変更を禁止できます。この句を DML 文の副問合せ内で使用する場合、FROM 句内の副問合せには指定できますが、WHERE 句内の副問合せには指定できません。

CONSTRAINT *constraint* CHECK OPTION 制約の名前を指定します。この識別子を省略した場合、その制約に SYS_C*n* という形式の名前が自動的に割り当てられます。この場合の n は、その制約名をデータベース内で一意の名前にする整数です。

注意 : 表に WITH CHECK OPTION を指定すると、挿入および更新処理の結果、表を定義する副問合せによって選択可能な表が戻されることが保証されます。ビューでは、次の場合、WITH CHECK OPTION を指定してもこれらの保証はされません。

- 対象のビュー、または対象のビューの基礎となるビューを定義する問合せ内に、副問合せが含まれている場合。
- INSERT、UPDATE または DELETE 操作が INSTEAD OF トリガーを使用して実行された場合。

***subquery_restriction_clause* の制限事項：** ORDER BY 句を指定した場合、この句は指定できません。

参照： 「読取り専用ビューの作成例 :」（17-21 ページ）

例

ビューの作成例： 次の文は、サンプル表 employees の emp_view という名前のビューを作成します。このビューには、部門 20 の従業員とその年間給与が表示されます。

```
CREATE VIEW emp_view AS
   SELECT last_name, salary*12 annual_salary
   FROM employees
   WHERE department_id = 20;
```

この場合、副問合せで式 salary*12 に列の別名（annual_salary）を使用しているため、この式に基づく列の名前をビュー宣言で定義する必要はありません。

制約付きのビューの作成例： 次の文は、サンプル表 hr.employees の制限付きのビューを作成し、ビューの列 email に一意制約を定義し、ビューの列 emp_id にビューに対する主キー制約を定義します。

```
CREATE VIEW emp_sal (emp_id, last_name,
      email UNIQUE RELY DISABLE NOVALIDATE,
   CONSTRAINT id_pk PRIMARY KEY (emp_id) RELY DISABLE NOVALIDATE)
   AS SELECT employee_id, last_name, email FROM employees;
```

更新可能なビューの作成例： 次の文は、employees 表内の事務員全員の更新可能なビュー clerk を作成します。従業員の ID、姓、部門番号および職種はビューで参照でき、事務員の行のこれらの列のみを更新できます。

```
CREATE VIEW clerk AS
   SELECT employee_id, last_name, department_id, job_id
   FROM employees
   WHERE job_id = 'PU_CLERK'
      or job_id = 'SH_CLERK'
      or job_id = 'ST_CLERK';
```

このビューでは、購買係の job_id を購買マネージャ（PU_MAN）に変更できます。

```
UPDATE clerk SET job_id = 'PU_MAN' WHERE employee_id = 118;
```

次の例は、同じビューを WITH CHECK OPTION 付きで作成します。新しい従業員が事務員ではない場合は、clerk に新しい行を挿入できません。従業員の job_id を他の事務タイプに更新できますが、事務員以外の job_id の場合はビューが employees にアクセスできないため、正常に更新できません。

```
CREATE VIEW clerk AS
   SELECT employee_id, last_name, department_id, job_id
   FROM employees
   WHERE job_id = 'PU_CLERK'
      or job_id = 'SH_CLERK'
      or job_id = 'ST_CLERK'
   WITH CHECK OPTION;
```

結合ビューの作成例： 結合ビューは、結合を含むビューの副問合せを持つビューです。副問合せの結合において、一意索引を持つ列が 1 列以上ある場合、結合ビューで 1 つの実表を変更できます。結合ビューの中の列が更新可能であるかどうかは、USER_UPDATABLE_COLUMNS を問い合せることでわかります。たとえば、次のようになります。

```
CREATE VIEW locations_view AS
   SELECT d.department_id, d.department_name, l.location_id, l.city
   FROM departments d, locations l
```

```
   WHERE d.location_id = l.location_id;

SELECT column_name, updatable
   FROM user_updatable_columns
   WHERE table_name = 'LOCATIONS_VIEW'
   ORDER BY column_name, updatable;

COLUMN_NAME                    UPD
------------------------------ ---
DEPARTMENT_ID                  YES
DEPARTMENT_NAME                YES
LOCATION_ID                    NO
CITY                           NO
```

前述の例では、locations 表の location_id 列に対する主キー索引は、locations_view ビュー内で一意ではありません。そのため、locations はキー保存表ではなく、その実表の列は更新できません。

```
INSERT INTO locations_view VALUES
   (999, 'Entertainment', 87, 'Roma');
INSERT INTO locations_view VALUES
*
ERROR at line 1:
ORA-01776: cannot modify more than one base table through a join view
```

departments 表にマップするビュー内のすべての列に更新可能のマークが付けられていて、departments の主キーがビューに保持されているため、departments 実表に対して、行の挿入、更新または削除を実行できます。

```
INSERT INTO locations_view (department_id, department_name)
   VALUES (999, 'Entertainment');

1 row created.
```

注意： ビューを使用して表に挿入するには、NOT NULL 列に対して DEFAULT 値を指定していないかぎり、結合内のすべての表のすべての NOT NULL 列がビューに含まれている必要があります。

参照： 結合ビューの更新の詳細は、『Oracle Database 管理者ガイド』を参照してください。

読取り専用ビューの作成例： 次の文は、oe.customers 表の customer_ro という名前の読取り専用ビューを作成します。顧客の姓、言語、掛貸限度額のみがこのビューで参照できます。

```
CREATE VIEW customer_ro (name, language, credit)
      AS SELECT cust_last_name, nls_language, credit_limit
      FROM customers
      WITH READ ONLY;
```

オブジェクト・ビューの作成例： 次の例では、oc スキーマ内での inventory_typ 型の作成方法、およびその型を基にした oc_inventories ビューの作成方法を示します。

```
CREATE TYPE inventory_typ
 OID '82A4AF6A4CD4656DE034080020E0EE3D'
 AS OBJECT
    ( product_id          NUMBER(6)
    , warehouse           warehouse_typ
    , quantity_on_hand    NUMBER(8)
    ) ;
/
CREATE OR REPLACE VIEW oc_inventories OF inventory_typ
```

```
 WITH OBJECT OID (product_id)
 AS SELECT i.product_id,
           warehouse_typ(w.warehouse_id, w.warehouse_name, w.location_id),
           i.quantity_on_hand
    FROM inventories i, warehouses w
    WHERE i.warehouse_id=w.warehouse_id;
```

XMLType 表のビューの作成例： 次の例は、XMLType 表 xwarehouses（16-60 ページの「例」で作成）の通常のビューを作成します。

```
CREATE VIEW warehouse_view AS
   SELECT VALUE(p) AS warehouse_xml
   FROM xwarehouses p;
```

このビューからは、次のように選択します。

```
SELECT e.warehouse_xml.getclobval()
   FROM warehouse_view e
   WHERE EXISTSNODE(warehouse_xml, '//Docks') =1;
```

XMLType ビューの作成例： オブジェクト・リレーショナル表に、XMLType ビューを作成する場合もあります。次の例は、サンプル表 oe.warehouses 内の XMLType 列 warehouse_spec と同様のオブジェクト・リレーショナル表を作成し、その表の XMLType ビューを作成します。

```
CREATE TABLE warehouse_table
(
  WarehouseID       NUMBER,
  Area              NUMBER,
  Docks             NUMBER,
  DockType          VARCHAR2(100),
  WaterAccess       VARCHAR2(10),
  RailAccess        VARCHAR2(10),
  Parking           VARCHAR2(20),
  VClearance        NUMBER
);

INSERT INTO warehouse_table
   VALUES(5, 103000,3,'Side Load','false','true','Lot',15);

CREATE VIEW warehouse_view OF XMLTYPE
 XMLSCHEMA "http://www.example.com/xwarehouses.xsd"
    ELEMENT "Warehouse"
    WITH OBJECT ID
    (extract(OBJECT_VALUE, '/Warehouse/Area/text()').getnumberval())
 AS SELECT XMLELEMENT("Warehouse",
            XMLFOREST(WarehouseID as "Building",
                      area as "Area",
                      docks as "Docks",
                      docktype as "DockType",
                      wateraccess as "WaterAccess",
                      railaccess as "RailAccess",
                      parking as "Parking",
                      VClearance as "VClearance"))
  FROM warehouse_table;
```

このビューは、次のように問い合せます。

```
SELECT VALUE(e) FROM warehouse_view e;
```

DELETE

用途

DELETE 文を使用すると、次の表から行を削除できます。

- 非パーティション表またはパーティション表
- ビューの非パーティション実表またはパーティション実表
- 書込み可能なマテリアライズド・ビューの非パーティション・コンテナ表またはパーティション・コンテナ表
- 更新可能なマテリアライズド・ビューの非パーティション・マスター表またはパーティション・マスター表

前提条件

表から行を削除する場合、その表が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、その表に対する DELETE オブジェクト権限が必要です。

更新可能なマテリアライズド・ビューから行を削除する場合、そのマテリアライズド・ビューが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、そのマテリアライズド・ビューに対する DELETE オブジェクト権限が必要です。

ビューの実表から行を削除する場合、そのビューが含まれるスキーマの所有者には、その実表に対する DELETE オブジェクト権限が必要です。また、他のスキーマ内にビューが存在している場合は、そのビューに対する DELETE オブジェクト権限が必要です。

DELETE ANY TABLE システム権限を持っている場合、任意の表、表パーティションまたは任意のビューの実表から行を削除できます。

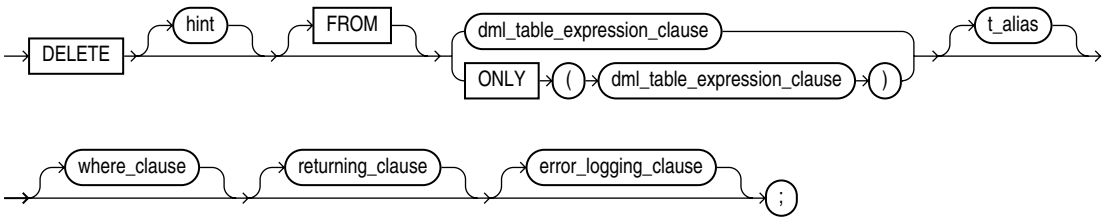
次の場合は、削除するオブジェクトに対する SELECT オブジェクト権限も必要です。

- オブジェクトがリモート・データベースに存在する。
- 初期化パラメータ SQL92_SECURITY が TRUE に設定され、DELETE 操作が表列（*where_clause* 内の列など）を参照する。

表のファンクション索引が無効な状態にあるとき、表から行を削除できません。まず、ファクション索引を検証する必要があります。

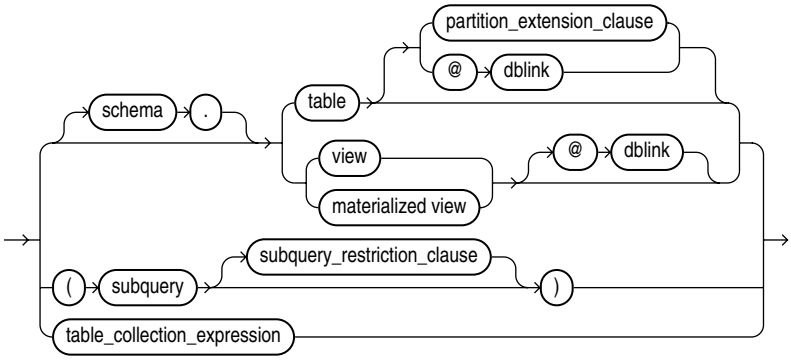
構文

***delete*::=**



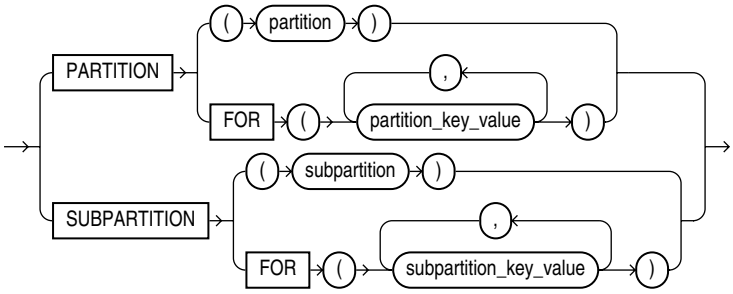
（17-24 ページの DML_table_expression_clause::=、17-24 ページの where_clause::=、17-24 ページの returning_clause::=、17-25 ページの error_logging_clause::= を参照）

DML_table_expression_clause::=

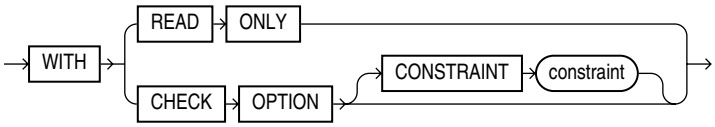


（17-24 ページの partition_extension_clause::=、19-5 ページの subquery::=、17-24 ページの subquery_restriction_clause::=、17-24 ページの table_collection_expression::= を参照）

***partition_extension_clause*::=**



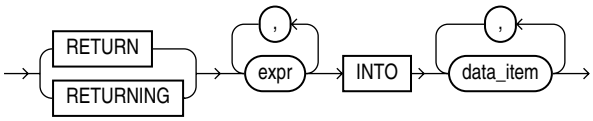
subquery_restriction_clause::=



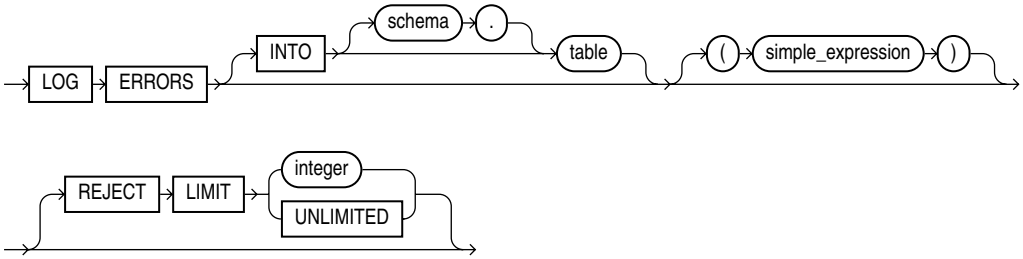
table_collection_expression::=

where_clause::=

returning_clause::=



error_logging_clause::=



セマンティクス

hint

文の実行計画を選択する場合に、オプティマイザに指示を与えるためのコメントを指定します。

> **参照：** ヒントの構文および説明については、2-70 ページの「ヒントの使用方法」を参照してください。

from_clause

FROM 句を使用すると、行を削除するデータベース・オブジェクトを指定できます。

ONLY 構文は、ビューのみに関連します。FROM 句のビューがビューの階層に属し、そのいずれのサブビューからも行を削除しない場合は、ONLY 句を使用します。

DML_table_expression_clause

この句を使用すると、データを削除するオブジェクトを指定できます。

schema

表またはビューが含まれているスキーマを指定します。*schema* を指定しない場合、表またはビューは自分のスキーマにあるとみなされます。

table | *view* | *materialized view* | *subquery*

行を削除する表、ビュー、マテリアライズド・ビュー、列または副問合せの結果の列の名前を指定します。

更新可能なビューから行を削除すると、実表から行が削除されます。

読取り専用マテリアライズド・ビューからは行を削除できません。書込み可能なマテリアライズド・ビューから行を削除する場合、基礎となるコンテナ表から行が削除されます。ただし、その削除は次のリフレッシュ操作で上書きされます。マテリアライズド・ビュー・グループ内の更新可能なマテリアライズド・ビューから行を削除する場合、マスター表の対応する行も削除されます。

table、*view* の実表または *materialized_view* のマスター表に、1 列以上のドメイン索引列が含まれる場合は、この文によって適切な索引タイプの削除ルーチンが実行されます。

> **参照：** これらのルーチンの詳細は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。

表に対して DELETE 文を実行すると、その表に定義されている DELETE トリガーが起動されます。

行の削除によって解放される表や索引のすべての領域は、表および索引によって引き続き保持されます。

partition_extension_clause

オブジェクト内にある削除対象のパーティションまたはサブパーティションの名前またはパーティション・キー値を指定します。

パーティション・オブジェクトから値を削除する場合、そのパーティション名を指定する必要はありません。ただし、パーティション名を指定した方が、複雑な *where_clause* より効率が上がることがあります。

> **参照：** 2-106 ページの「パーティション表と索引の参照」および 17-29 ページの「パーティションの行の削除例 :」を参照してください。

dblink

オブジェクトが格納されているリモート・データベースへのデータベース・リンクの完全名または部分名を指定します。Oracle Database の分散機能を使用している場合にのみ、リモート・オブジェクトから行を削除できます。

> **参照：** データベース・リンクの参照方法の詳細は、2-104 ページの「リモート・データベース内のオブジェクトの参照」および 17-29 ページの「リモート・データベースの行の削除例 :」を参照してください。

dblink を省略した場合、オブジェクトがローカル・データベース上にあるとみなされます。

subquery_restriction_clause

subquery_restriction_clause を使用すると、次のいずれかの方法で副問合せを制限できます。

WITH READ ONLY `WITH READ ONLY` を指定すると、表またはビューを更新禁止にできます。

WITH CHECK OPTION `WITH CHECK OPTION` を指定すると、副問合せに含まれない行を生成する表またはビューの変更を禁止できます。この句を DML 文の副問合せ内で使用する場合、`FROM` 句内の副問合せには指定できますが、`WHERE` 句内の副問合せには指定できません。

CONSTRAINT *constraint* `CHECK OPTION` 制約の名前を指定します。この識別子を省略した場合、その制約に `SYS_Cn` という形式の名前が自動的に割り当てられます。この場合の n は、その制約名をデータベース内で一意の名前にする整数です。

> **参照：** 「WITH CHECK OPTION 句の使用例 :」（19-39 ページ）

table_collection_expression

table_collection_expression を使用すると、問合せおよび DML 操作で、*collection_expression* 値を表として扱うことができます。*collection_expression* には、副問合せ、列、ファンクションまたはコレクション・コンストラクタのいずれかを指定できます。その形式にかかわらず、集合値（ネストした表型または VARRAY 型の値）を戻す必要があります。このようなコレクションの要素抽出プロセスを**コレクション・ネスト解除**といいます。

`TABLE` 式を親表と結合する場合は、オプションのプラス（+）には大きな意味があります。+ を指定すると、その 2 つの外部結合が作成され、コレクション式が NULL の場合でも、外部表の行が問合せで戻されるようになります。

注意： 以前のリリースの Oracle では、*collection_expression* が副問合せの場合、*table_collection_expression* を `THE` *subquery* と表現していました。現在、このような表現方法は非推奨になっています。

相関副問合せで *table_collection_expression* を使用すると、他の表に存在する値で行を削除できます。

参照：「表のコレクション例 :」（19-44 ページ）

collection_expression 削除するオブジェクトからネストした表の列を選択する副問合せを指定します。

***dml_table_expression_clause* 句の制限事項：** この句には、次の制限事項があります。

- *view* または *materialized_view* の *table*（実表またはマスター表）に IN_PROGRESS または FAILED のマークが付いたドメイン索引がある場合、この文は実行できません。
- 関係する索引パーティションが UNUSABLE とマークされている場合は、パーティションに挿入できません。
- *DML_table_expression_clause* の副問合せでは、ORDER BY 句を指定できません。
- ビューを定義する問合せに次のいずれかの構造体が含まれている場合は、INSTEAD OF トリガーを使用する場合を除いて、ビューから行を削除できません。

  集合演算子
  DISTINCT 演算子
  集計ファンクションまたは分析ファンクション
  GROUP BY、ORDER BY、MODEL、CONNECT BY または START WITH 句
  SELECT 構文のリストにあるコレクション式
  SELECT 構文のリストにある副問合せ
  WITH READ ONLY が指定された副問合せ
  結合（一部の例外を除く。詳細は、『Oracle Database 管理者ガイド』を参照してください。）

UNUSABLE のマークが付いている索引、索引パーティションまたは索引サブパーティションを指定する場合、SKIP_UNUSABLE_INDEXES 初期化パラメータが true に設定されていないかぎり、DELETE 文は正常に実行されません。

参照：「ALTER SESSION」（11-41 ページ）

where_clause

where_clause を使用すると、条件を満たす行のみを削除できます。この条件は、行を削除するオブジェクトを参照したり、副問合せを含むことができます。Oracle Database の分散機能を使用している場合にのみ、リモート・オブジェクトから行を削除できます。*condition* の構文は、第 7 章「条件」を参照してください。

この句がリモート・オブジェクトを参照する *subquery* を含む場合、参照がローカル・データベース上でオブジェクトにループバックしないかぎり、DELETE はパラレルで実行されます。ただし、*DML_table_expression_clause* の *subquery* がリモート・オブジェクトを参照する場合は、DELETE 操作はシリアルで実行されます。詳細は、16-54 ページの「CREATE TABLE」の「parallel_clause」を参照してください。

dblink を省略した場合、表またはビューがローカル・データベース上にあるとみなされます。

where_clause を省略した場合、オブジェクトのすべての行が削除されます。

t_alias 文中で参照する表、ビュー、マテリアライズド・ビュー、副問合せまたはコレクション値の**相関名**を指定します。*DML_table_expression_clause* がいずれかのオブジェクト型属性またはオブジェクト型メソッドを参照する場合、この別名が必要です。通常、別名は相関問合せを持つ DELETE 文で使用されます。

returning_clause

この句を使用すると、削除された列から値を戻すことができるため、DELETE 文の後で SELECT を実行する必要がなくなります。

この句を使用すると、DML 文に影響される行を取り出すことができます。この句は、表、マテリアライズド・ビュー、および単一の実表を持つビューに指定できます。

returning_clause を指定した DML 文を単一行に実行すると、影響された行、ROWID、および処理された行への REF を使用している列式が取り出され、ホスト変数または PL/SQL 変数に格納されます。

returning_clause を指定した DML 文を複数行に実行すると、式の値、ROWID および処理された行に関連する REF がバインド配列に格納されます。

expr *expr* リストの各項目は、適切な構文で表す必要があります。

INTO INTO 句を指定すると、変更された行の値を、*data_item* リストに指定する変数に格納できます。

data_item 取り出された *expr* 値を格納するホスト変数または PL/SQL 変数を指定します。

RETURNING リストの各式については、INTO リストに、対応する型に互換性がある PL/SQL 変数またはホスト変数を指定する必要があります。

RETURNING 句の制限事項 : RETURNING 句には、次の制限事項があります。

- *expr* に次の制限事項があります。
  - UPDATE 文および DELETE 文の場合、各 *expr* は、単純式または単一セットの集計ファンクション式である必要があります。1 つの *returning_clause* 内に単純式と単一セットの集計ファンクション式を混在させることはできません。INSERT 文の場合、各 *expr* は単純式である必要があります。INSERT 文の RETURNING 句では、集計ファンクションはサポートされていません。
  - 単一セットの集計ファンクション式を DISTINCT キーワードに含めることはできません。
- *expr* リストに主キー列またはその他の NOT NULL 列が含まれている場合、表に BEFORE UPDATE トリガーが定義されていると、UPDATE 文は正常に実行されません。
- マルチテーブル・インサートでは *returning_clause* を指定できません。
- パラレル DML またはリモート・オブジェクトにはこの句を使用できません。
- LONG 型を取り出すことはできません。
- INSTEAD OF トリガーが定義されたビューに対して指定することはできません。

参照 :

- BULK COLLECT 句を使用してコレクション変数に複数の値を戻す場合は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。
- 「RETURNING 句の使用例 :」(17-29 ページ)

error_logging_clause

error_logging_clause の DELETE 文での動作は、INSERT 文の場合と同じです。詳細は、18-63 ページの「INSERT」文の「error_logging_clause」を参照してください。

参照 : 「エラー・ロギングによる表への挿入例 :」(18-64 ページ)

例

行の削除例： 次の文は、language_id 列の値が AR の、サンプル表 oe.product_descriptions からすべての行を削除します。

```
DELETE FROM product_descriptions
   WHERE language_id = 'AR';
```

次の文は、サンプル表 hr.employees から歩合率が 10% 未満の購買係を削除します。

```
DELETE FROM employees
   WHERE job_id = 'SA_REP'
   AND commission_pct < .2;
```

次の文は前述の例と同じ結果を表しますが、副問合せを使用します。

```
DELETE FROM (SELECT * FROM employees)
   WHERE job_id = 'SA_REP'
   AND commission_pct < .2;
```

リモート・データベースの行の削除例： 次の文は、データベース・リンク remote によってアクセス可能なデータベースの、ユーザー hr が所有する locations 表から指定した行を削除します。

```
DELETE FROM hr.locations@remote
   WHERE location_id > 3000;
```

ネストした表の行の削除例： ネストした表の行の削除例は、19-44 ページの「表のコレクション例 :」を参照してください。

パーティションの行の削除例： 次の例は、sh.sales 表のパーティション sales_q1_1998 から行を削除します。

```
DELETE FROM sales PARTITION (sales_q1_1998)
   WHERE amount_sold > 1000;
```

RETURNING 句の使用例： 次の例は、削除された行から salary 列を戻し、その結果をバインド配列 :bnd1 に格納します。バインド配列は事前に宣言しておく必要があります。

```
DELETE FROM employees
   WHERE job_id = 'SA_REP'
   AND hire_date + TO_YMINTERVAL('01-00') < SYSDATE
   RETURNING salary INTO :bnd1;
```

DISASSOCIATE STATISTICS

用途

DISASSOCIATE STATISTICS 文を使用すると、デフォルトの統計情報または統計タイプと、列、スタンドアロン・ファンクション、パッケージ、型、ドメイン索引または索引タイプとの関連付けを解除できます。

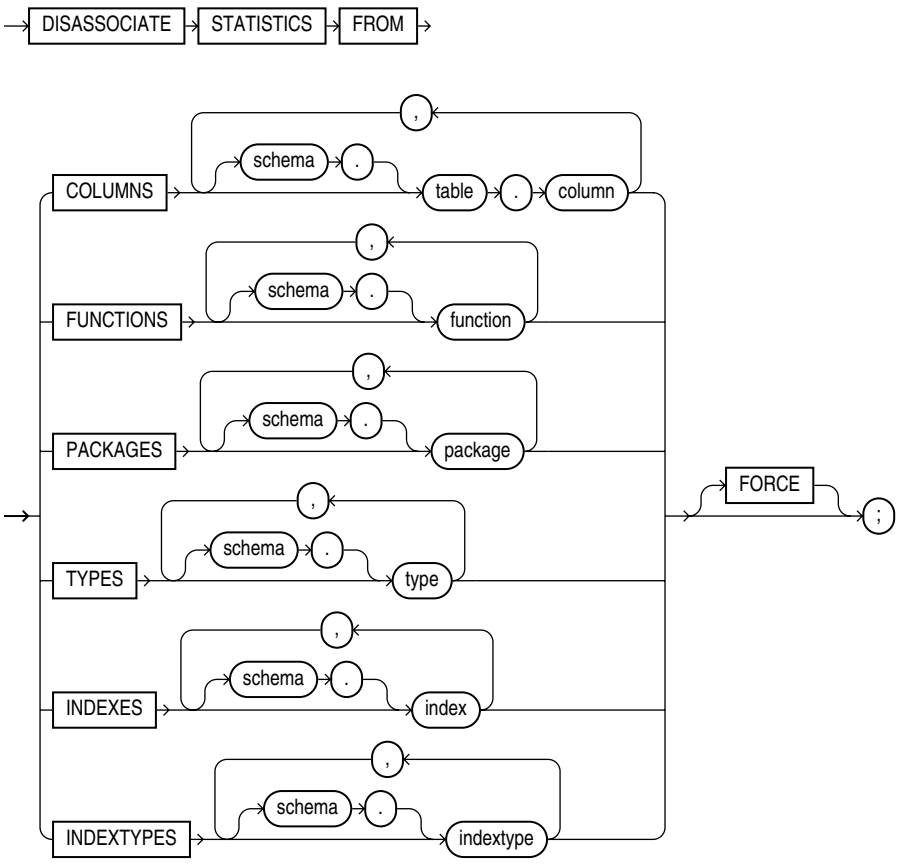
> **参照：** 統計タイプの関連付けの詳細は、13-21 ページの「ASSOCIATE STATISTICS」を参照してください。

前提条件

この文を発行する場合は、基礎となる表、ファンクション、パッケージ、型、ドメイン索引または索引タイプを変更する適切な権限が必要です。

構文

***disassociate_statistics*::=**



セマンティクス

FROM COLUMNS | FUNCTIONS | PACKAGES | TYPES | INDEXES | INDEXTYPES

統計との関連付けを解除する 1 つ以上の列、スタンドアロン・ファンクション、パッケージ、型、ドメイン索引または索引タイプを指定します。

schema を指定しない場合、オブジェクトは自分のスキーマ内にあるとみなされます。

オブジェクトのユーザー定義の統計情報を収集する場合、FORCE を指定しないかぎり文は実行されません。

FORCE

FORCE を指定すると、統計タイプを使用するオブジェクトの統計情報の有無にかかわらず、関連付けが解除されます。統計情報が存在する場合、関連付けが解除される前にその統計情報は削除されます。

> **注意：** 統計タイプが関連付けられているオブジェクトを削除する場合、FORCE オプションによって統計タイプの関連付けが自動的に解除され、統計タイプを使用して収集したすべての統計情報が削除されます。

例

統計情報の関連付けの解除例： 次の文は、emp_mgmt パッケージと統計情報の関連付けを解除します。hr スキーマにこのパッケージを作成する例については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

```
DISASSOCIATE STATISTICS FROM PACKAGES hr.emp_mgmt;
```

DROP CLUSTER

用途

DROP CLUSTER 文を使用すると、データベースからクラスタを削除できます。

> **注意：** クラスタを削除すると、ごみ箱内のそのクラスタの一部だった表はごみ箱から消去され、FLASHBACK TABLE 操作でもリカバリできなくなります。

個々の表は非クラスタ化できません。そのかわり、次の手順を実行します。

1. 既存の表と同じ構成および内容で、新しい表を CLUSTER 句なしで作成します。
2. 既存の表を削除します。
3. RENAME 文を使用して、新しい表に既存の表の名前を指定します。
4. 新しいクラスタ化表に対する権限を付与します。古いクラスタ化表に対する権限は使用できません。

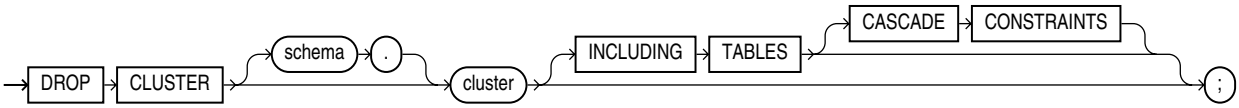
> **参照：** これらの手順の詳細は、16-6 ページの「CREATE TABLE」、18-5 ページの「DROP TABLE」、18-82 ページの「RENAME」および 18-31 ページの「GRANT」を参照してください。

前提条件

削除するクラスタが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY CLUSTER システム権限が必要です。

構文

***drop_cluster*::=**



セマンティクス

schema

クラスタが含まれているスキーマを指定します。*schema* を指定しない場合、クラスタは自分のスキーマ内にあるとみなされます。

cluster

削除するクラスタの名前を指定します。クラスタを削除するとクラスタの索引も削除され、その索引のデータ・ブロックを含むクラスタ領域が表領域に戻されます。

INCLUDING TABLES

INCLUDING TABLES を指定すると、クラスタに属するすべての表を削除できます。

CASCADE CONSTRAINTS

CASCADE CONSTRAINTS を指定すると、クラスタに含まれる表の主キーまたは一意キーを参照するクラスタ外の表から、すべての参照整合性制約を削除できます。このような参照整合性制約があるときにこの句の指定を省略した場合、エラーが戻され、クラスタは削除されません。

例

クラスタの削除例： 次の文は、クラスタ（14-7 ページの「CREATE CLUSTER」の「例」で作成）を削除します。

次の文は、language クラスタを削除します。

```
DROP CLUSTER language;
```

次の文は、dept_10、dept_20 およびこれらの表の主キーまたは一意キーを参照する参照整合性制約とともに personnel クラスタを削除します。

```
DROP CLUSTER personnel
   INCLUDING TABLES
   CASCADE CONSTRAINTS;
```

DROP CONTEXT

用途

DROP CONTEXT 文を使用すると、データベースからコンテキスト・ネームスペースを削除できます。

コンテキスト・ネームスペースを削除した場合でも、ユーザー・セッションに設定されたネームスペースのコンテキストは無効になりません。ただし、ユーザーがそのコンテキストを設定しようとした場合、そのコンテキストは無効になります。

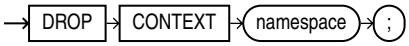
> **参照：** コンテキストの詳細は、14-8 ページの「CREATE CONTEXT」および『Oracle Database セキュリティ・ガイド』を参照してください。

前提条件

DROP ANY CONTEXT システム権限が必要です。

構文

***drop_context*::=**



セマンティクス

namespace

削除するコンテキスト・ネームスペースの名前を指定します。組込みネームスペース USERENV は削除できません。

> **参照：** USERENV ネームスペースの詳細は、5-184 ページの「SYS_CONTEXT」を参照してください。

例

アプリケーション・コンテキストの削除例： 次の文は、コンテキスト（14-8 ページの「CREATE CONTEXT」で作成）を削除します。

```
DROP CONTEXT hr_context;
```

DROP DATABASE

用途

注意： DROP DATABASE 文はロールバックできません。

DROP DATABASE 文を使用すると、データベースを削除できます。この文は、テスト・データベースを削除したり、新しいホストへの移行が完了した後に古いデータベースを削除する場合に便利です。

参照： データベースの削除の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

前提条件

この文を発行するには、SYSDBA システム権限が必要です。データベースは、排他モードおよび制限モードでマウントされている必要があります。また、データベースはクローズ状態である必要があります。

構文

***drop_database*::=**

セマンティクス

この文を発行すると、データベースを削除し、すべての制御ファイル、およびその制御ファイルに記述されているデータ・ファイルを削除できます。サーバー・パラメータ・ファイル（SPFILE）を使用している場合は、それも削除されます。

アーカイブ・ログおよびバックアップは削除されません。これらを削除するには、Recovery Manager を使用します。データベースが RAW ディスク上に存在する場合、この文では実際の RAW ディスクの特殊ファイルは削除されません。

DROP DATABASE LINK

用途

DROP DATABASE LINK 文を使用すると、データベースからデータベース・リンクを削除できます。

> **参照：** データベース・リンクの作成方法については、14-28 ページの「CREATE DATABASE LINK」を参照してください。

前提条件

プライベート・データベース・リンクは自分のスキーマにある必要があります。パブリック・データベース・リンクを削除する場合は、DROP PUBLIC DATABASE LINK システム権限が必要です。

構文

***drop_database_link*::=**

セマンティクス

PUBLIC

PUBLIC を指定すると、パブリック・データベース・リンクを削除できます。

dblink

削除するデータベース・リンクの名前を指定します。

データベース・リンクの削除の制限事項： 他のユーザーのスキーマ内のデータベース・リンクは削除できません。また、データベース・リンク名ではピリオドが許可されるため、スキーマ名で *dblink* を修飾することはできません。そのため、ralph.linktosales のような名前を付けた場合、スキーマ ralph の中の linktosales という名前のデータベース・リンクであるとはみなされず、名前全体が自分のスキーマにあるデータベース・リンク名であると解析されます。

例

データベース・リンクの削除例： 次の文は、パブリック・データベース・リンク remote（14-31 ページの「パブリック・データベース・リンクの定義例 :」で作成）を削除します。

```
DROP PUBLIC DATABASE LINK remote;
```

DROP DIMENSION

用途

DROP DIMENSION を使用すると、指定したディメンションを削除できます。

この文によって、ディメンションに指定された関連を使用するマテリアライズド・ビューが無効になるわけではありません。ただし、クエリー・リライトによってリライトされた要求が無効になり、そのようなビューに対する後続の操作の処理速度が遅くなることがあります。

参照 :

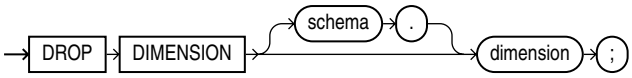
- ディメンションの作成および変更については、14-33 ページの「CREATE DIMENSION」および 10-43 ページの「ALTER DIMENSION」を参照してください。
- ディメンションの概要は、『Oracle Database 概要』を参照してください。

前提条件

この文を使用する場合、ディメンションが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY DIMENSION システム権限が必要です。

構文

***drop_dimension*::=**



セマンティクス

schema

ディメンションが格納されているスキーマの名前を指定します。*schema* を指定しない場合、ディメンションは自分のスキーマ内にあるとみなされます。

dimension

削除するディメンションの名前を指定します。そのディメンションはすでに存在している必要があります。

例

ディメンションの削除例 : この例は、sh.customers_dim を削除します。

```
DROP DIMENSION customers_dim;
```

参照 : ディメンションの作成および変更の例については、14-36 ページの「ディメンションの作成例 :」および 10-45 ページの「ディメンションの変更例 :」を参照してください。

DROP DIRECTORY

用途

DROP DIRECTORY 文を使用すると、データベースからディレクトリ・オブジェクトを削除できます。

> **参照：** ディレクトリの作成については、14-38 ページの「CREATE DIRECTORY」を参照してください。

前提条件

ディレクトリを削除する場合は、DROP ANY DIRECTORY システム権限が必要です。

> **注意：** PL/SQL または OCI プログラムによる関連ファイル・システム内のファイルへのアクセス中は、DROP によるディレクトリの削除はできません。

構文

***drop_directory*::=**

セマンティクス

directory_name

削除するディレクトリ・データベース・オブジェクトの名前を指定します。

Oracle Database はディレクトリ・オブジェクトを削除しますが、サーバーのファイル・システム上の関連するオペレーティング・システム・ディレクトリは削除しません。

例

ディレクトリの削除例： 次の文は、ディレクトリ・オブジェクト bfile_dir を削除します。

```
DROP DIRECTORY bfile_dir;
```

> **参照：** 「ディレクトリの作成例：」（14-39 ページ）

DROP DISKGROUP

> **注意：** この SQL 文は、自動ストレージ管理を使用しており、自動ストレージ管理インスタンスを起動している場合にのみ有効です。この文の発行は、通常のデータベース・インスタンスからではなく、自動ストレージ管理インスタンスから行う必要があります。自動ストレージ管理インスタンスの起動の詳細は、『Oracle Database ストレージ管理者ガイド』を参照してください。

用途

DROP DISKGROUP 文を使用すると、自動ストレージ管理ディスク・グループおよびそのディスク・グループのすべてのファイルを削除できます。自動ストレージ管理では、最初にディスク・グループのファイルが開かれていないことが確認されます。次に、ディスク・グループおよびそのメンバー・ディスクが削除され、ディスク・ヘッダーが消去されます。

参照：

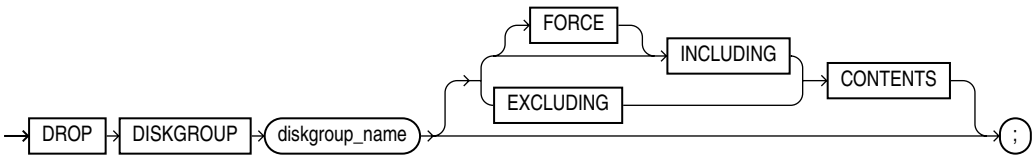
- ディスク・グループの作成および変更については、14-40 ページの「CREATE DISKGROUP」および 10-46 ページの「ALTER DISKGROUP」を参照してください。
- 自動ストレージ管理の詳細、およびディスク・グループを使用してデータベース管理を簡略化する方法については、『Oracle Database ストレージ管理者ガイド』を参照してください。

前提条件

SYSDBA システム権限が必要です。また、この文を発行する自動ストレージ管理インスタンスが起動されている必要があります。削除するディスク・グループはマウントされている必要があります。

構文

***drop_diskgroup*::=**



セマンティクス

diskgroup_name

削除するディスク・グループの名前を指定します。

INCLUDING CONTENTS

INCLUDING CONTENTS を指定すると、自動ストレージ管理によってディスク・グループのすべてのファイルが削除されます。ディスク・グループにファイルが含まれている場合、この句を指定する必要があります。

EXCLUDING CONTENTS

EXCLUDING CONTENTS を指定すると、自動ストレージ管理によって空のディスク・グループのみが削除されます。これはデフォルトです。ディスク・グループが空ではない場合、エラーが戻されます。

FORCE

この句では、ASM インスタンスではマウントできないディスク・グループに属するディスク上のヘッダーがクリアされます。ディスク・グループは、データベースのインスタンスからはマウントできません。

自動ストレージ管理インスタンスは、ディスク・グループが、同じストレージ・サブシステムを使用する他の ASM で使用されているかどうかをまず確認します。ディスク・グループが使用されており、このディスク・グループが同じクラスタまたは同じノードにあるときは、文の実行に失敗します。ディスク・グループが異なるクラスタにある場合は、さらに、そのディスク・グループが別のクラスタ内のインスタンスによってマウントされているかどうかがチェックされます。ディスク・グループがマウントされている場合、この文の実行は失敗します。ただし、後者のチェックは、同じクラスタのディスク・グループのチェックほど確実なものではありません。そのため、この句の使用には注意が必要です。

例

ディスク・グループの削除例： 次の文は、自動ストレージ管理ディスク・グループ dgroup_01（14-44 ページの「ディスク・グループの作成例：」で作成）およびそのディスク・グループ内のすべてのファイルを削除します。

```
DROP DISKGROUP dgroup_01 INCLUDING CONTENTS;
```

DROP FLASHBACK ARCHIVE

用途

DROP FLASHBACK ARCHIVE 句を使用すると、システムからフラッシュバック・データ・アーカイブを削除できます。この文では、フラッシュバック・データ・アーカイブとこの中にあるすべての履歴データが削除されますが、フラッシュバック・データ・アーカイブによって使用されている表領域は削除されません。

前提条件

フラッシュバック・データ・アーカイブを削除するには、FLASHBACK ARCHIVE ADMINISTER システム権限が必要です。

構文

***drop_flashback_archive*::=**

セマンティクス

flashback_archive

削除するフラッシュバック・データ・アーカイブの名前を指定します。

このフラッシュバック・データ・アーカイブによって履歴追跡が可能なすべての表について、フラッシュバック・データ・アーカイブが無効化されている場合を除き、デフォルトのフラッシュバック・データ・アーカイブを削除することはできません。

> **参照：** フラッシュバック・データ・アーカイブの作成の詳細およびフラッシュバック・データ・アーカイブの簡単な使用例は、14-45 ページの「CREATE FLASHBACK ARCHIVE」を参照してください。

DROP FUNCTION

用途

ファンクションは PL/SQL を使用して定義されます。ファンクションの作成、変更および削除の詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

DROP FUNCTION 文を使用すると、データベースからスタンドアロン・ストアド・ファンクションを削除できます。

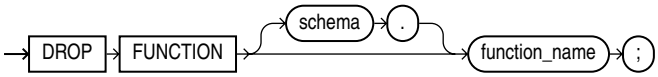
注意： この文を使用してパッケージの一部のファンクションを削除しないでください。かわりに、DROP PACKAGE 文を使用してパッケージ全体を削除するか、または OR REPLACE 句を指定した CREATE PACKAGE 文を使用して、そのファンクションを含めずにパッケージを再定義してください。

前提条件

削除するファンクションが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY PROCEDURE システム権限が必要です。

構文

***drop_function*::=**



セマンティクス

schema

ファンクションが含まれているスキーマを指定します。*schema* を指定しない場合、ファンクションは自分のスキーマ内にあるとみなされます。

function_name

削除するファンクションの名前を指定します。

Oracle Database は、削除したファンクションに依存するローカル・オブジェクト、または削除したファンクションをコールするローカル・オブジェクトを無効にします。後でこれらのオブジェクトを参照した場合、データベースは、そのオブジェクトを再コンパイルしようとします。削除したファンクションを再作成しないかぎり、エラーが戻されます。

任意の統計タイプがファンクションに関連付けられている場合、FORCE 句によって統計タイプの関連付けが解除され、統計タイプを使用して収集したユーザー定義のすべての統計情報が削除されます。

参照：

- リモート・オブジェクトを含むスキーマ・オブジェクト間の依存性を Oracle Database が管理する方法については、『Oracle Database 概要』を参照してください。
- 統計タイプの関連付けの詳細は、13-21 ページの「ASSOCIATE STATISTICS」および 17-30 ページの「DISASSOCIATE STATISTICS」を参照してください。

例

ファンクションの削除例： 次の文は、サンプル・スキーマ oe 内のファンクション SecondMax を削除します。この場合、SecondMax に依存するすべてのオブジェクトは無効になります。

```
DROP FUNCTION oe.SecondMax;
```

> **参照：** SecondMax ファンクションを作成する例については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

DROP INDEX

用途

DROP INDEX 文を使用すると、データベースから索引またはドメイン索引を削除できます。

索引を削除すると、ビュー、パッケージ、パッケージ本体、ファンクション、プロシージャなど、基礎となる表に依存するすべてのオブジェクトが無効になります。

グローバル・パーティション索引、レンジ・パーティション索引またはハッシュ・パーティション索引を削除すると、すべての索引パーティションも削除されます。コンポジット・パーティション索引を削除した場合、すべての索引パーティションおよびサブパーティションも削除されます。

また、ドメイン索引を削除すると、次のようになります。

- 適切なルーチンが起動されます。
- 任意の統計情報がドメイン索引に関連付けられている場合、FORCE 句によって統計タイプの関連付けが解除され、その統計タイプを使用して収集したユーザー定義の統計情報が削除されます。

参照 :

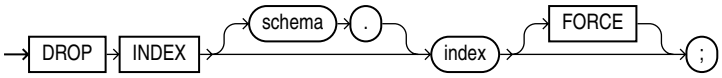
- ルーチンの詳細は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。
- 索引の作成および変更の詳細は、14-50 ページの「CREATE INDEX」および 10-64 ページの「ALTER INDEX」を参照してください。
- ドメイン索引の詳細は、14-50 ページの「CREATE INDEX」の「*domain_index_clause*」を参照してください。
- 統計タイプの関連付けの詳細は、13-21 ページの「ASSOCIATE STATISTICS」および 17-30 ページの「DISASSOCIATE STATISTICS」を参照してください。

前提条件

削除する索引が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY INDEX システム権限が必要です。

構文

***drop_index*::=**



セマンティクス

schema

索引が含まれているスキーマを指定します。*schema* を指定しない場合、索引は自分のスキーマ内にあるとみなされます。

index

削除する索引の名前を指定します。索引を削除すると、その索引に割り当てられていたすべてのデータ・ブロックが、その索引が含まれていた表領域に戻されます。

索引の削除の制限事項： 索引またはいずれかの索引パーティションに IN_PROGRESS のマークが付けられている場合は、ドメイン索引を削除できません。

FORCE

FORCE は、ドメイン索引のみに適用されます。この句を指定すると、索引タイプ・ルーチンの起動がエラーを戻した場合、または索引に IN PROGRESS のマークが付けられている場合でも、ドメイン索引を削除できます。FORCE がないと、その索引タイプ・ルーチンの起動がエラーを戻す場合、または索引に IN PROGRESS マークが付けられている場合にドメイン索引を削除できません。

例

索引の削除例： この文は、索引 ord_customer_ix_demo（14-68 ページの「索引の圧縮例 :」で作成）を削除します。

```
DROP INDEX ord_customer_ix_demo;
```

DROP INDEXTYPE

用途

DROP INDEXTYPE 文を使用すると、索引タイプを削除できます。同時に、統計タイプとの関連付けも解除されます。

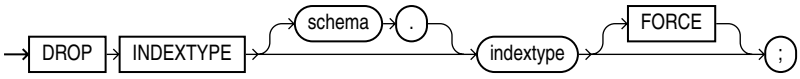
> **参照：** 索引タイプの詳細は、14-73 ページの「CREATE INDEXTYPE」を参照してください。

前提条件

削除する索引タイプが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY INDEXTYPE システム権限が必要です。

構文

***drop_indextype*::=**



セマンティクス

schema

索引タイプが含まれているスキーマを指定します。*schema* を指定しない場合、この索引タイプは自分のスキーマ内にあるとみなされます。

indextype

削除する索引タイプの名前を指定します。

任意の統計タイプが索引タイプに関連付けられている場合、統計タイプと索引タイプの関連付けが解除され、統計タイプを使用して収集したすべての統計情報が削除されます。

> **参照：** 統計情報の関連付けの詳細は、13-21 ページの「ASSOCIATE STATISTICS」および 17-30 ページの「DISASSOCIATE STATISTICS」を参照してください。

FORCE

FORCE を指定すると、1 つ以上のドメイン索引から現在参照されている状態でも、その索引タイプを削除できます。これらのドメイン索引には、INVALID マークが付けられます。FORCE を指定しない場合、任意のドメイン索引で参照されている索引タイプを削除できません。

例

索引タイプの削除例： 次の文は、索引タイプ position_indextype（E-2 ページの「拡張索引作成機能の使用方法」で作成）を削除し、この索引タイプで定義されているすべてのドメイン索引に INVALID のマークを付けます。

```
DROP INDEXTYPE position_indextype FORCE;
```

DROP JAVA

用途

DROP JAVA 文を使用すると、Java ソース、クラスまたはリソース・スキーマ・オブジェクトを削除できます。

参照 :

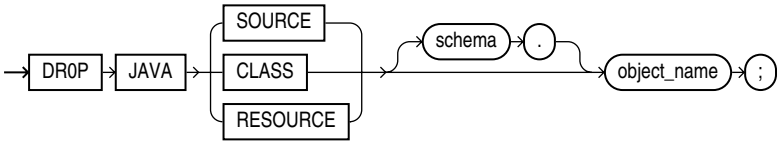
- Java オブジェクトの作成については、14-76 ページの「CREATE JAVA」を参照してください。
- Java ソース、クラスおよびリソースの変換の詳細は、『Oracle Database Java 開発者ガイド』を参照してください。

前提条件

削除する Java ソース、クラスまたはリソースが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY PROCEDURE システム権限が必要です。また、このコマンドを使用する場合は、Java クラスに対する EXECUTE オブジェクト権限も必要です。

構文

***drop_java*::=**



セマンティクス

JAVA SOURCE

SOURCE を指定すると、Java ソース・スキーマ・オブジェクトおよびそのオブジェクトから導出されたすべての Java クラス・スキーマ・オブジェクトを削除できます。

JAVA CLASS

CLASS を指定すると、Java クラス・スキーマ・オブジェクトを削除できます。

JAVA RESOURCE

RESOURCE を指定すると、Java リソース・スキーマ・オブジェクトを削除できます。

object_name

既存の Java クラス、ソースまたはリソース・スキーマ・オブジェクトの名前を指定します。*object_name* を二重引用符で囲むと、小文字の名前、または大文字と小文字を組み合せた名前を指定できます。

例

Java クラス・オブジェクトの削除例 : 次の文は、Java クラス Agent（14-80 ページの「Java クラス・オブジェクトの作成例 :」で作成）を削除します。

```
DROP JAVA CLASS "Agent";
```

DROP LIBRARY

用途

DROP LIBRARY 文を使用すると、データベースから外部プロシージャ・ライブラリを削除できます。

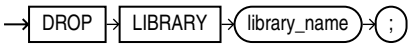
> **参照：** ライブラリの作成については、15-2 ページの「CREATE LIBRARY」を参照してください。

前提条件

DROP ANY LIBRARY システム権限が必要です。

構文

***drop_library*::=**



セマンティクス

library_name

削除する外部プロシージャ・ライブラリの名前を指定します。

例

ライブラリの削除例： 次の文は、ext_lib ライブラリ（15-3 ページの「ライブラリの作成例:」で作成）を削除します。

```
DROP LIBRARY ext_lib;
```

DROP MATERIALIZED VIEW

用途

DROP MATERIALIZED VIEW 文を使用すると、データベースから既存のマテリアライズド・ビューを削除できます。

削除したマテリアライズド・ビューは、ごみ箱内には移動しません。このため、削除したマテリアライズド・ビューを消去またはリカバリすることはできません。

注意： 下位互換性を保つために、MATERIALIZED VIEW のかわりにキーワード SNAPSHOT もサポートされています。

参照：

- 様々なタイプのマテリアライズド・ビューの詳細は、15-4 ページの「CREATE MATERIALIZED VIEW」を参照してください。
- マテリアライズド・ビューの変更については、11-2 ページの「ALTER MATERIALIZED VIEW」を参照してください。
- レプリケーション環境でのマテリアライズド・ビューの詳細は、『Oracle Database アドバンスト・レプリケーション』を参照してください。
- データ・ウェアハウス環境でのマテリアライズド・ビューの詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

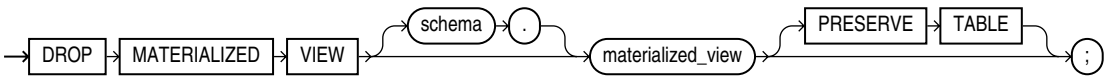
前提条件

削除するマテリアライズド・ビューが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY MATERIALIZED VIEW システム権限が必要です。また、データベースがマテリアライズド・ビューのデータを管理するために使用する内部表、ビュー、索引を削除する権限も必要です。

参照： マテリアライズド・ビューを管理するために使用するオブジェクトの削除に必要な権限については、18-5 ページの「DROP TABLE」、18-17 ページの「DROP VIEW」および 17-44 ページの「DROP INDEX」を参照してください。

構文

***drop_materialized_view*::=**



セマンティクス

schema

マテリアライズド・ビューが含まれているスキーマを指定します。*schema* を指定しない場合、このマテリアライズド・ビューは自分のスキーマ内にあるとみなされます。

materialized_view

削除する既存のマテリアライズド・ビューの名前を指定します。

- マスター表から、最後にリフレッシュされた単純マテリアライズド・ビューを削除した場合、マスター表のマテリアライズド・ビュー・ログのうち、削除されたマテリアライズド・ビューのリフレッシュに必要な行のみが自動的に削除されます。
- 事前作成表で作成されたマテリアライズド・ビューを削除した場合、そのマテリアライズド・ビューが削除され、事前作成表は 1 つの表としての元の形に戻ります。
- マスター表を削除した場合、その表に基づくマテリアライズド・ビューは自動的には削除されません。ただし、削除したマスター表に基づくマテリアライズド・ビューをリフレッシュしようとした場合、エラー・メッセージが戻されます。
- マテリアライズド・ビューを削除した場合、マテリアライズド・ビューを使用するためにリライトされたコンパイル済の要求が無効になり、自動的に再コンパイルされます。そのマテリアライズド・ビューが表に事前作成されていた場合、その表は削除されませんが、マテリアライズド・ビューのリフレッシュ・メカニズムで管理できなくなります。

PRESERVE TABLE 句

この句を使用すると、マテリアライズド・ビュー・オブジェクトを削除した後も、そのマテリアライズド・ビューのコンテナ表およびその内容を保持できます。結果の表の名前は、削除したマテリアライズド・ビューと同じになります。

マテリアライズド・ビューに関連付けられているすべてのメタデータは削除されます。ただし、そのマテリアライズド・ビューの作成時にコンテナ表で自動的に作成されたすべての索引は、保存されます。また、そのマテリアライズド・ビューにネストした表の列が含まれる場合、その列の記憶表がそのメタデータとともに保存されます。

PRESERVE TABLE 句の制限事項 : この句は、（マテリアライズド・ビューが「スナップショット」と呼ばれていた）Oracle9*i* より前のリリースからインポートしたマテリアライズド・ビューには無効です。

例

マテリアライズド・ビューの削除例 : 次の文は、サンプル・スキーマ hr のマテリアライズド・ビュー emp_data を削除します。

```
DROP MATERIALIZED VIEW emp_data;
```

次の文は、sales_by_month_by_state マテリアライズド・ビューおよびそのマテリアライズド・ビューの基礎となる表を削除します（基礎となる表が ON PREBUILT TABLE 句が指定された CREATE MATERIALIZED VIEW 文に登録されていない場合）。

```
DROP MATERIALIZED VIEW sales_by_month_by_state;
```

DROP MATERIALIZED VIEW LOG

用途

DROP MATERIALIZED VIEW LOG 文を使用すると、データベースからマテリアライズド・ビュー・ログを削除できます。

注意： 下位互換性を保つために、MATERIALIZED VIEW のかわりにキーワード SNAPSHOT もサポートされています。

参照：

- マテリアライズド・ビューの詳細は、15-4 ページの「CREATE MATERIALIZED VIEW」および 11-2 ページの「ALTER MATERIALIZED VIEW」を参照してください。
- マテリアライズド・ビュー・ログの詳細は、15-26 ページの「CREATE MATERIALIZED VIEW LOG」を参照してください。
- レプリケーション環境でのマテリアライズド・ビューの詳細は、『Oracle Database アドバンスト・レプリケーション』を参照してください。
- データ・ウェアハウス環境でのマテリアライズド・ビューの詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

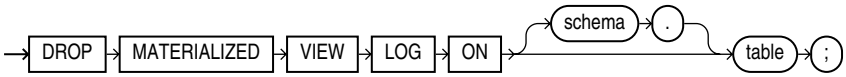
前提条件

マテリアライズド・ビュー・ログを削除する場合は、表を削除する権限が必要です。

参照： 「DROP TABLE」（18-5 ページ）

構文

***drop_materialized_view_log*::=**



セマンティクス

schema

マテリアライズド・ビュー・ログおよびそのマスター表が含まれているスキーマを指定します。*schema* を指定しない場合、マテリアライズド・ビュー・ログおよびマスター表は自分のスキーマ内にあるとみなされます。

table

削除するマテリアライズド・ビュー・ログに関連付けられたマスター表の名前を指定します。

マテリアライズド・ビュー・ログを削除した場合、マテリアライズド・ビュー・ログのマスター表に基づく一部のマテリアライズド・ビューが高速リフレッシュできなくなります。高速リフレッシュできなくなるマテリアライズド・ビューには、ROWID マテリアライズド・ビュー、主キー・マテリアライズド・ビューおよび副問合せマテリアライズド・ビューが含まれます。

参照： マテリアライズド・ビューの種類については、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

例

マテリアライズド・ビュー・ログの削除例： 次の文は、oe.customers マスター表のマテリアライズド・ビュー・ログを削除します。

```
DROP MATERIALIZED VIEW LOG ON customers;
```

DROP OPERATOR

用途

DROP OPERATOR 文を使用すると、ユーザー定義演算子を削除できます。

参照 :

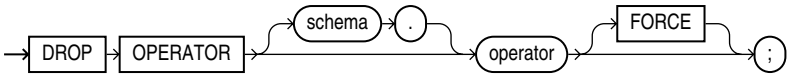
- 演算子の作成および変更の詳細は、15-32 ページの「CREATE OPERATOR」および 11-22 ページの「ALTER OPERATOR」を参照してください。
- 演算子の概要については、4-9 ページの「ユーザー定義演算子」および『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。
- ユーザー定義索引タイプの演算子の削除については、10-82 ページの「ALTER INDEXTYPE」を参照してください。

前提条件

削除する演算子が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY OPERATOR システム権限が必要です。

構文

***drop_operator*::=**



セマンティクス

schema

演算子が含まれているスキーマを指定します。*schema* を指定しない場合、その演算子は自分のスキーマ内にあるとみなされます。

operator

削除する演算子の名前を指定します。

FORCE

FORCE を指定すると、演算子が現在 1 つ以上のスキーマ・オブジェクト（索引タイプ、パッケージ、ファンクション、プロシージャなど）によって参照されている場合でも、その演算子を削除できます。演算子に依存するこのようなオブジェクトには、INVALID のマークが付けられます。FORCE を使用しないと、任意のスキーマ・オブジェクトに参照されている演算子を削除できません。

例

ユーザー定義演算子の削除例 : 次の文は、演算子 eq_op を削除します。

```
DROP OPERATOR eq_op;
```

FORCE 句が指定されていないため、この演算子のバインディングのいずれかが索引タイプによって参照されている場合、この操作は実行されません。

DROP OUTLINE

用途

注意： 新規のアプリケーションにはSQL 計画管理を使用することをお薦めします。SQL 計画管理では、ストアド・アウトラインよりも非常に安定したSQL パフォーマンスを実現するSQL 計画ベースラインが作成されます。

DBMS_SPM パッケージの LOAD_PLANS_FROM_CURSOR_CACHE プロシージャまたは LOAD_PLANS_FROM_SQLSET プロシージャを使用して、既存のストアド・アウトラインをSQL 計画ベースラインに移行できます。移行が完了したら、ストアド・アウトラインを無効にするか、または削除する必要があります。

参照：SQL 計画管理の詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。DBMS_SPM パッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

DROP OUTLINE 文を使用すると、ストアド・アウトラインを削除できます。

参照：

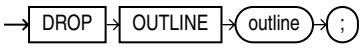
- アウトラインの作成については、15-35 ページの「CREATE OUTLINE」を参照してください。
- アウトラインについては、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。

前提条件

アウトラインを削除する場合は、DROP ANY OUTLINE システム権限が必要です。

構文

***drop_outline*::=**



セマンティクス

outline

削除するアウトラインの名前を指定します。

そのアウトラインが削除された後、ストアド・アウトラインが作成されたSQL 文がコンパイルされると、オプティマイザはアウトラインの影響なしに新しい実行計画を生成します。

例

アウトラインの削除例： 次の文は、ストアド・アウトライン salaries を削除します。

```
DROP OUTLINE salaries;
```

DROP PACKAGE

用途

パッケージは PL/SQL を使用して定義されます。パッケージの作成、変更および削除の詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

DROP PACKAGE 文を使用すると、データベースからストアド・パッケージを削除できます。この文は、パッケージの本体および仕様部を削除します。

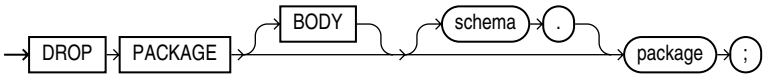
> **注意：** この文を使用してパッケージから単一のオブジェクトを削除しないでください。かわりに、CREATE PACKAGE 文および CREATE PACKAGE BODY 文に OR REPLACE 句を指定して、そのオブジェクトを含めずにパッケージを再作成してください。

前提条件

削除するパッケージが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY PROCEDURE システム権限が必要です。

構文

***drop_package*::=**



セマンティクス

BODY

BODY を指定すると、パッケージ本体のみを削除できます。この句を省略した場合、パッケージ本体と仕様部の両方が削除されます。

パッケージ本体のみを削除して仕様部は削除しない場合、依存するオブジェクトは無効になりません。ただし、パッケージ本体を再作成しないかぎり、パッケージ仕様部で宣言したプロシージャやストアド・ファンクションはコールできません。

schema

パッケージが含まれているスキーマを指定します。*schema* を指定しない場合、パッケージは自分のスキーマ内にあるとみなされます。

package

削除するパッケージの名前を指定します。

そのパッケージ仕様部に依存するすべてのローカル・オブジェクトが無効になります。後でこれらのオブジェクトを参照した場合、データベースは、そのオブジェクトを再コンパイルしようとします。削除したパッケージを再作成しないかぎり、エラーが戻されます。

任意の統計タイプがパッケージに関連付けられている場合、FORCE 句によって統計タイプの関連付けが解除され、統計タイプを使用して収集されたユーザー定義のすべての統計情報が削除されます。

> **参照：** 13-21 ページの「ASSOCIATE STATISTICS」および 17-30 ページの「DISASSOCIATE STATISTICS」を参照してください。

例

パッケージの削除例： 次の文は、emp_mgmt パッケージの仕様部および本体を削除し、その仕様部に依存するすべてのオブジェクトを無効にします。このパッケージを作成する例については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

```
DROP PACKAGE emp_mgmt;
```

DROP PROCEDURE

用途

プロシージャは PL/SQL を使用して定義されます。プロシージャの作成、変更および削除の詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

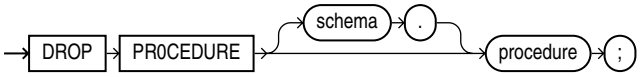
DROP PROCEDURE 文を使用すると、データベースからスタンドアロンのストアド・プロシージャを削除できます。この文を使用してパッケージの一部であるプロシージャを削除しないでください。かわりに、DROP PACKAGE 文を使用してパッケージ全体を削除するか、OR REPLACE 句を指定した CREATE PACKAGE 文を使用して、そのプロシージャを含めずにパッケージを再定義してください。

前提条件

削除するプロシージャが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY PROCEDURE システム権限が必要です。

構文

***drop_procedure*::=**



セマンティクス

schema

プロシージャが含まれているスキーマを指定します。*schema* を指定しない場合、プロシージャは自分のスキーマ内にあるとみなされます。

procedure

削除するプロシージャの名前を指定します。

プロシージャを削除すると、そのプロシージャに依存するすべてのローカル・オブジェクトは無効になります。後でこれらのオブジェクトを参照した場合、データベースは、そのオブジェクトを再コンパイルしようとします。削除したプロシージャを再作成しないかぎり、エラー・メッセージが戻されます。

例

プロシージャの削除例： 次の文は、ユーザー hr が所有するプロシージャ remove_emp を削除し、remove_emp に依存するすべてのオブジェクトを無効にします。

```
DROP PROCEDURE hr.remove_emp;
```

DROP PROFILE

用途

DROP PROFILE 文を使用すると、データベースからプロファイルを削除できます。DEFAULT プロファイル以外のプロファイルを削除できます。

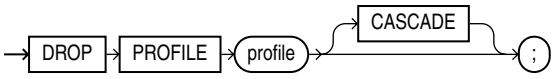
> **参照：** プロファイルの作成および変更の詳細は、15-47 ページの「CREATE PROFILE」および 11-29 ページの「ALTER PROFILE」を参照してください。

前提条件

DROP PROFILE システム権限が必要です。

構文

***drop_profile*::=**



セマンティクス

profile

削除するプロファイルの名前を指定します。

CASCADE

CASCADE を指定すると、プロファイルの割当て先のユーザーからプロファイルの割当てを解除できます。このようなユーザーには、自動的に DEFAULT プロファイルが割り当てられます。現在、複数のユーザーに割り当てられているプロファイルを削除する場合は、必ずこの句を指定します。

例

プロファイルの削除例： 次の文は、プロファイル app_user（15-51 ページの「プロファイルの作成例：」で作成）を削除します。プロファイル app_user は削除され、現在 app_user プロファイルが割り当てられているユーザーには DEFAULT プロファイルが割り当てられます。

```
DROP PROFILE app_user CASCADE;
```

DROP RESTORE POINT

用途

DROP RESTORE POINT 文を使用すると、通常のリストア・ポイントまたは保証付きリストア・ポイントをデータベースから削除できます。

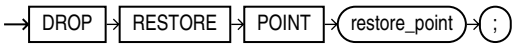
- 通常のリストア・ポイントは、ユーザーが削除する必要はありません。必要に応じて古いものから順に自動的に削除されます（15-54 ページの「restore_point」の説明を参照）。ただし、同じ名前を再利用する必要がある場合は、通常のリストア・ポイントをユーザーが削除することもできます。
- 保証付きリストア・ポイントは、自動的に削除されません。したがって、保証付きリストア・ポイントをデータベースから削除するには、明示的にこの文を使用する必要があります。

> **参照：** リストア・ポイントの作成方法および使用方法については、15-53 ページの「CREATE RESTORE POINT」、18-22 ページの「FLASHBACK DATABASE」および 18-25 ページの「FLASHBACK TABLE」を参照してください。

前提条件

通常のリストア・ポイントを削除するには、SELECT ANY DICTIONARY または FLASHBACK ANY TABLE のいずれかのシステム権限が必要です。保証付きリストア・ポイントを削除するには、SYSDBA システム権限が必要です。

構文



セマンティクス

restore_point 削除するリストア・ポイントの名前を指定します。

例

リストア・ポイントの削除例： 次の例は、good_data リストア・ポイント（15-55 ページの「リストア・ポイントを作成して使用する例：」で作成）を削除します。

```
DROP RESTORE POINT good_data;
```

DROP ROLE

用途

DROP ROLE 文を使用すると、データベースからロールを削除できます。ロールを削除した場合、そのロールが付与されていたすべてのユーザーおよびロールからそのロールが取り消され、データベースからも削除されます。すでに使用可能なロールのユーザー・セッションには影響しません。ただし、削除後は新しいユーザー・セッションでロールを使用可能にできません。

参照：

- ロールの作成、およびロールを使用可能にするために必要な認可の変更については、15-56 ページの「CREATE ROLE」および 11-34 ページの「ALTER ROLE」を参照してください。
- 現行のセッションに対するロールの使用禁止化については、19-51 ページの「SET ROLE」を参照してください。

前提条件

Admin Option 付きのロールが付与されているか、または DROP ANY ROLE システム権限を持っている必要があります。

構文

***drop_role*::=**

セマンティクス

role

削除するロールの名前を指定します。

例

ロールの削除例： 次の文は、ロール dw_manager（15-58 ページの「ロールの作成例：」で作成）を削除します。

```
DROP ROLE dw_manager;
```

DROP ROLLBACK SEGMENT

用途

DROP ROLLBACK SEGMENT 文を使用すると、データベースからロールバック・セグメントを削除できます。ロールバック・セグメントを削除した場合、ロールバック・セグメントに割り当てられていたすべての領域が表領域に戻されます。

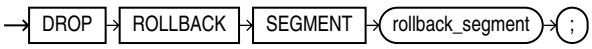
> **注意：** データベースが自動 UNDO モードで実行されている場合、ロールバック・セグメントに有効な操作はこの文のみです。このモードの場合、ロールバック・セグメントを作成または変更できません。

前提条件

DROP ROLLBACK SEGMENT システム権限が必要です。また、ロールバック・セグメントをオフラインにしておく必要があります。

構文

***drop_rollback_segment*::=**



セマンティクス

rollback_segment

削除するロールバック・セグメントの名前を指定します。

ロールバック・セグメントの削除の制限事項： この文には、次の制限事項があります。

- ロールバック・セグメントは、オフラインになっている場合にのみ削除できます。ロールバック・セグメントがオフラインであるかどうかを判断するには、DBA_ROLLBACK_SEGS データ・ディクショナリ・ビューを問い合せます。オフラインのロールバック・セグメントは、STATUS 列の値が AVAILABLE になっています。また、ALTER ROLLBACK SEGMENT 文に OFFLINE 句を指定して、ロールバック・セグメントをオフラインにすることもできます。
- SYSTEM ロールバック・セグメントは削除できません。

例

ロールバック・セグメントの削除例： 次の構文は、ロールバック・セグメント（15-61 ページの「ロールバック・セグメントの作成例 :」で作成）を削除します。

```
DROP ROLLBACK SEGMENT rbs_one;
```

18

SQL 文 : DROP SEQUENCE ～ ROLLBACK

この章では、次の SQL 文について説明します。

- DROP SEQUENCE
- DROP SYNONYM
- DROP TABLE
- DROP TABLESPACE
- DROP TRIGGER
- DROP TYPE
- DROP TYPE BODY
- DROP USER
- DROP VIEW
- EXPLAIN PLAN
- FLASHBACK DATABASE
- FLASHBACK TABLE
- GRANT
- INSERT
- LOCK TABLE
- MERGE
- NOAUDIT
- PURGE
- RENAME
- REVOKE
- ROLLBACK

DROP SEQUENCE

用途

DROP SEQUENCE 文を使用すると、データベースから順序を削除できます。

この文を使用すると、順序の削除および再作成を行って、順序を再開することもできます。たとえば、現在、値が 150 の順序を初期値 27 で再開する場合、順序を削除して同じ名前で再作成し、START WITH 値を 27 にします。

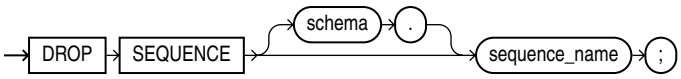
> **参照：** 順序の作成および変更の詳細は、15-64 ページの「CREATE SEQUENCE」および 11-39 ページの「ALTER SEQUENCE」を参照してください。

前提条件

削除する順序が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY SEQUENCE システム権限が必要です。

構文

***drop_sequence*::=**



セマンティクス

schema

順序が含まれているスキーマを指定します。*schema* を指定しない場合、順序は自分のスキーマ内にあるとみなされます。

sequence_name

削除する順序の名前を指定します。

例

順序の削除例： 次の文は、ユーザー oe が所有する順序 customers_seq（15-67 ページの「順序の作成例：」で作成）を削除します。この文を発行する場合は、ユーザー oe として接続するか、または DROP ANY SEQUENCE システム権限が必要です。

```
DROP SEQUENCE oe.customers_seq;
```

DROP SYNONYM

用途

DROP SYNONYM 文を使用すると、データベースからシノニムを削除できます。また、シノニムの削除および再作成を行って、シノニムの定義を変更することもできます。

> **参照：** シノニムの詳細は、16-2 ページの「CREATE SYNONYM」を参照してください。

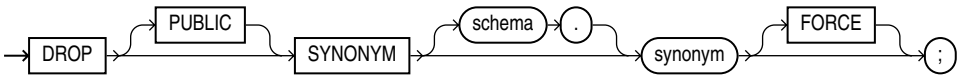
前提条件

プライベート・シノニムを削除する場合は、シノニムが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY SYNONYM システム権限が必要です。

パブリック・シノニムを削除する場合は、DROP PUBLIC SYNONYM システム権限が必要です。

構文

***drop_synonym*::=**



セマンティクス

PUBLIC

PUBLIC を指定すると、パブリック・シノニムを削除できます。PUBLIC を指定した場合、*schema* は指定できません。

schema

シノニムが含まれているスキーマを指定します。*schema* を指定しない場合、シノニムは自分のスキーマ内にあるとみなされます。

synonym

削除するシノニムの名前を指定します。

マテリアライズド・ビューを定義する問合せに実際の表名ではなくシノニムが指定されていた場合に、このマテリアライズド・ビューのマスター表のシノニムを削除すると、このマテリアライズド・ビューには使用禁止のマークが付けられます。

依存表またはユーザー定義型を持つオブジェクト型のシノニムを削除するには、FORCE も指定する必要があります。

FORCE

FORCE を指定すると、依存表またはユーザー定義型を持つシノニムでも強制的に削除できます。

> **注意：** FORCE を使用して、依存性のあるオブジェクト型のシノニムを削除することはお薦めしません。この操作によって、他のユーザー定義型が無効になるか、またはこのシノニムに依存する表列に UNUSED のマークが付く場合があります。型の依存性の詳細は、『Oracle Database オブジェクト・リレーショナル開発者ガイド』を参照してください。

例

シノニムの削除例： 次の文は、パブリック・シノニム customers（16-4 ページの「Oracle Database によるシノニムの変換例 :」で作成）を削除します。

```
DROP PUBLIC SYNONYM customers;
```

DROP TABLE

用途

DROP TABLE 文を使用すると、表またはオブジェクト表をごみ箱に移動したり、表およびそれに含まれるすべてのデータをデータベースから完全に削除することができます。

> **注意：** PURGE 句を指定しないかぎり、DROP TABLE を実行しても、他のオブジェクトが使用できるように領域が解放されて表領域に戻されることはなく、その領域はユーザーの領域割当てとして計算されたままになります。

外部表の場合、この文はデータベースにある表のメタデータのみを削除します。データベースの外部に存在する実際のデータには影響しません。

クラスタの一部である表を削除すると、その表はごみ箱に移動します。ただし、その後にそのクラスタを削除すると、その表はごみ箱から消去され、FLASHBACK TABLE 操作でもリカバリできなくなります。

表を削除すると、依存オブジェクトが無効になり、表のオブジェクト権限が取り消されます。表を再作成する際、表のオブジェクト権限を再度付与し、表の索引、整合性制約およびトリガーを再作成し、記憶域パラメータを再指定する必要があります。このような影響は、切捨ておよび置換ではありません。そのため、TRUNCATE 文による行の削除、または CREATE OR REPLACE TABLE 文による表の置換えは、表を削除して再作成するよりも効率的です。

参照：

- 表の作成および変更の詳細は、16-6 ページの「CREATE TABLE」および 12-2 ページの「ALTER TABLE」を参照してください。
- 表からデータを削除する方法については、19-58 ページの「TRUNCATE TABLE」および 17-23 ページの「DELETE」を参照してください。
- 削除した表をごみ箱から取り出す方法については、18-25 ページの「FLASHBACK TABLE」を参照してください。

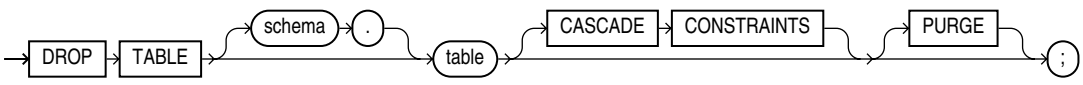
前提条件

削除する表が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY TABLE システム権限が必要です。

セッションがバインドされていない場合にのみ、一時表で DDL 操作（ALTER TABLE、DROP TABLE、CREATE INDEX など）を実行できます。セッションを一時表にバインドするには、一時表で INSERT 操作を実行します。セッションを一時表からアンバインドするには、TRUNCATE 文を発行するか、セッションを終了します。また、トランザクション固有の一時表からアンバインドするには、COMMIT または ROLLBACK 文を発行します。

構文

***drop_table*::=**



セマンティクス

schema

表が含まれているスキーマを指定します。*schema* を指定しない場合、表は自分のスキーマ内にあるとみなされます。

table

削除する表の名前を指定します。Oracle Database によって次の操作が自動的に実行されます。

- 表のすべての行が削除されます。
- 表のすべての索引およびドメイン索引のみでなく、その表に定義されているすべてのトリガーが、作成者やスキーマの所有者とは関係なく削除されます。*table* がパーティション化されている場合、対応するローカル索引パーティションも同様に削除されます。
- *table* のネストした表および LOB のすべての記憶表が削除されます。
- レンジ・パーティション表、ハッシュ・パーティション表またはリスト・パーティション表を削除すると、すべての表パーティションも削除されます。コンポジット・パーティション表を削除した場合、すべてのパーティションおよびサブパーティションが同様に削除されます。
- PURGE キーワードでパーティション表を削除する場合、その文は一連のサブトランザクションとして実行され、それぞれのサブトランザクションで、パーティションまたはサブパーティションのサブセットおよびそのメタデータが削除されます。削除操作がこのように複数のサブトランザクションに分割されていることで、特に大きなパーティション表を削除する場合に、内部システム・リソース（ライブラリ・キャッシュなど）の消費処理が最適化されます。最初のサブトランザクションがコミットされるとすぐに UNUSABLE のマークが表に付けられます。いずれかのサブトランザクションが失敗した場合、その表に対して可能な操作は、もう一度 DROP TABLE ... PURGE 文を実行することのみです。この場合、前回の DROP TABLE 文が失敗したところから、処理が継続されます。ただし、この処理は、前回の操作で発生したエラーが修正されていることを前提としています。

  このような削除操作で UNUSABLE のマークが付けられた表のリストは、該当する *_TABLES、*_PART_TABLES、*_ALL_TABLES または *_OBJECT_TABLES データ・ディクショナリ・ビューの status 列を問い合せて表示できます。

  > **参照：** パーティション表の削除の詳細は、『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。

- 索引構成表の場合、索引構成表に定義したマッピング表も削除されます。
- ドメイン索引の場合、適切な削除ルーチンが起動されます。これらのルーチンの詳細は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。
- 任意の統計タイプが表に関連付けられている場合、FORCE 句を使用して統計タイプの関連付けが解除され、統計タイプを使用して収集されたユーザー定義のすべての統計情報が削除されます。

  > **参照：** 統計タイプの関連付けの詳細は、13-21 ページの「ASSOCIATE STATISTICS」および 17-30 ページの「DISASSOCIATE STATISTICS」を参照してください。

- 表がクラスタの一部でない場合、表とその索引に割り当てられていたすべてのデータ・ブロックが、表とその索引が格納されていた表領域に戻されます。

  DROP CLUSTER 文に INCLUDING TABLES 句を指定した場合、クラスタおよびそのすべての表を削除できます。これによって、表を 1 つずつ削除する必要がなくなります。17-32 ページの「DROP CLUSTER」を参照してください。

- 表がビュー、マテリアライズド・ビューのコンテナ表またはマスター表の実表である場合、あるいは表がストアド・プロシージャ、ファンクションまたはパッケージで参照されている場合、依存するオブジェクトは無効になりますが、削除はされません。表を再作成するか、これらのオブジェクトを一度削除してその表に依存しない形で再作成しないかぎり、これらのオブジェクトは使用できません。

  表を再作成する場合、ストアド・プロシージャ、ファンクションまたはパッケージで参照する列、およびマテリアライズド・ビューの定義で使用されていた副問合せで選択されるすべての列が、その表に含まれている必要があります。ビュー、ストアド・プロシージャ、ファンクション、パッケージに対するオブジェクト権限が付与されていたユーザーに、これらの権限を再付与する必要はありません。

  表がマテリアライズド・ビューのマスター表の場合、マテリアライズド・ビューの問合せはできます。ただし、そのマテリアライズド・ビューを定義する問合せによって選択されるすべての列を含む表を再作成しないかぎり、リフレッシュはできません。

  表がマテリアライズド・ビュー・ログを含む場合、このログおよびその表に関連付けられているダイレクト・パス・インサートのその他のリフレッシュ情報は削除されます。

表の削除の制限事項 :

- ネストした表の記憶表は直接削除できません。かわりに、ALTER TABLE ... DROP COLUMN 句を使用して、ネストした表の列を削除する必要があります。
- 参照パーティション表の親表は削除できません。まず、参照パーティション表のすべての子表を削除する必要があります。

CASCADE CONSTRAINTS

CASCADE CONSTRAINTS を指定すると、削除される表の主キーまたは一意キーを参照するすべての参照整合性制約を削除できます。このような参照整合性制約があるときにこの句を省略した場合、エラーが戻され、表は削除されません。

PURGE

PURGE を指定すると、1 つの手順で、表を削除してその表に関連付けられた領域を解放できます。PURGE を指定すると、表およびその依存オブジェクトはごみ箱に移動しません。

注意 : PURGE 句を指定した DROP TABLE 文はロールバックできません。また、PURGE 句で削除した表はリカバリできません。

この句を使用することは、表を削除してからその表をごみ箱から消去することと同じです。この句を使用すると、それらの操作を 1 つの手順で実行できます。また、機密情報をごみ箱に表示しないようにできるため、セキュリティを強化できます。

参照 : ごみ箱の詳細、およびごみ箱内のオブジェクトのネーミング規則の詳細は、『Oracle Database 管理者ガイド』を参照してください。

例

表の削除例 : 次の文は、oe.list_customers 表（16-67 ページの「リスト・パーティション化の例 :」で作成）を削除します。

```
DROP TABLE list_customers PURGE;
```

DROP TABLESPACE

用途

DROP TABLESPACE 文を使用すると、データベースから表領域を削除できます。

削除した表領域は、ごみ箱内には移動しません。このため、削除した表領域を消去またはリカバリすることはできません。

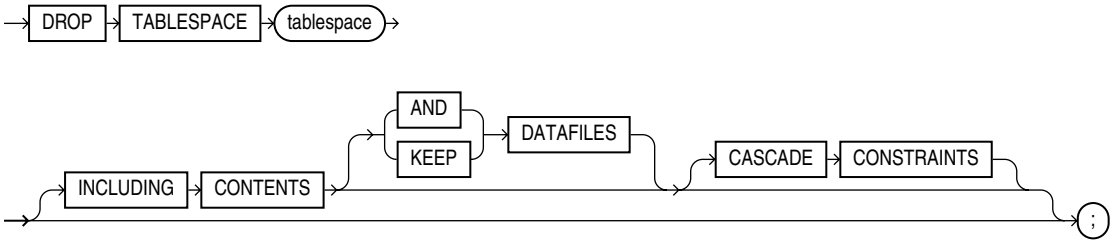
> **参照：** 表領域の作成および変更の詳細は、16-71 ページの「CREATE TABLESPACE」および 12-82 ページの「ALTER TABLESPACE」を参照してください。

前提条件

DROP TABLESPACE システム権限が必要です。アクティブ・トランザクションを保持するロールバック・セグメントを含む場合は、表領域を削除できません。

構文

***drop_tablespace*::=**



セマンティクス

tablespace

削除する表領域の名前を指定します。

表領域の状態がオンラインまたはオフラインのどちらであっても、その表領域を削除できます。実行中のトランザクション内の SQL 文で、表領域内のいずれかのオブジェクトにアクセスすることがないように、表領域はオフラインにしてから削除することをお薦めします。

SYSTEM 表領域は削除できません。SYSAUX 表領域は、SYSDBA システム権限を持ち、MIGRATE モードでデータベースを起動した場合にのみ削除できます。

削除する表領域がデフォルト表領域または一時表領域として割り当てられていたユーザーにアラートを出す必要がある場合があります。表領域が削除された後では、このようなユーザーはオブジェクトに領域を割り当てたり、表領域内で領域をソートすることはできません。ALTER USER 文を使用すると、ユーザーに新しいデフォルト表領域および一時表領域を割り当てることができます。

以前に表領域から削除し、ごみ箱に移動したオブジェクトがごみ箱から消去されます。表領域に関連するすべてのメタデータ、および表領域に含まれるすべてのデータ・ファイルと一時ファイルが、データ・ディクショナリから削除されます。また、表領域にある Oracle Managed Files のデータ・ファイルおよび一時ファイルが、オペレーティング・システムから自動的に削除されます。その他のデータ・ファイルおよび一時ファイルは、INCLUDING CONTENTS AND DATAFILES を指定しないかぎり、オペレーティング・システムから削除されません。

この文を使用して表領域グループを削除することはできません。ただし、*tablespace* が表領域グループ内で唯一の表領域である場合、その表領域グループもデータ・ディクショナリから削除されます。

表領域の削除の制限事項： 表領域の削除には、次の制限事項があります。

- ドメイン索引またはドメイン索引によって作成されたオブジェクトを格納している表領域は削除できません。
- いずれかのインスタンスによって使用されている場合、またはコミットされていないトランザクションのロールバックに必要な UNDO データを含む場合は、UNDO 表領域を削除できません。
- データベースのデフォルト表領域に指定されている表領域は削除できません。この表領域を削除するには、まず他の表領域をデフォルト表領域として再割当てする必要があります。
- データベースのデフォルトの一時表領域グループに属する一時表領域は削除できません。この表領域を削除するには、まずその表領域をデータベースのデフォルトの一時表領域グループから削除する必要があります。
- ある表領域を削除すると他の表領域の主キー制約または一意制約が無効になる場合、INCLUDING CONTENTS および CASCADE CONSTRAINTS 句を指定してもその表領域を削除できません。たとえば、削除する表領域に主キー列索引が含まれており、主キー列自体が別の表領域に存在する場合、その別の表領域内の主キー制約を手動で無効にしないかぎり、表領域を削除できません。

> **参照：** ドメイン索引の詳細は、『Oracle Database データ・カートリッジ開発者ガイド』および『Oracle Database 概要』を参照してください。

INCLUDING CONTENTS

INCLUDING CONTENTS を指定すると、表領域の中のすべてのデータベース・オブジェクトを削除できます。データベース・オブジェクトを格納している表領域を削除する場合は、必ずこの句を指定します。表領域が空でない場合にこの句を省略した場合、エラーが戻され、表領域は削除されません。

1 つの表のパーティションまたはサブパーティションが表領域に（すべてではなく）一部含まれていると、INCLUDING CONTENTS を指定しても DROP TABLESPACE コマンドが正常に実行されません。パーティション表のすべてのパーティションまたはサブパーティションが *tablespace* に存在する場合、DROP TABLESPACE ... INCLUDING CONTENTS は、*tablespace* を削除し、関連付けられた索引セグメント、LOB データ・セグメントおよび他の表領域にある表の LOB 索引セグメントも削除します。

パーティション化された索引構成表の場合、すべての主キー索引セグメントがこの表領域に存在する場合、この句は、他の表領域にあるオーバーフロー・セグメントも、他の表領域にある関連するマッピング表と同様に削除します。主キー索引セグメントのいくつかが存在しない場合、その文は実行されません。その場合、その表領域を削除する前に、ALTER TABLE ... MOVE PARTITION を使用して、それらの主キー索引セグメントをこの表領域に移動し、この表領域にオーバーフロー・データ・セグメントの存在しないパーティションを削除します。また、パーティション化された索引構成表も削除します。

表領域がマテリアライズド・ビューのマスター表を含む場合、マテリアライズド・ビューは無効になります。

表領域がマテリアライズド・ビュー・ログを含む場合、このログおよびその表に関連付けられているダイレクト・パス・インサートのその他のリフレッシュ情報は削除されます。

AND DATAFILES

INCLUDING CONTENTS を指定するときに AND DATAFILES 句を指定すると、関連するオペレーティング・システム・ファイルも削除できます。Oracle Database によって、アラート・ログに、削除された各オペレーティング・システム・ファイルに関するメッセージが書き込まれます。この句は、Oracle Managed Files については不要です。Oracle Managed Files は、AND DATAFILES を指定しなくてもシステムから削除されます。

KEEP DATAFILES

INCLUDING CONTENTS を指定するときに KEEP DATAFILES 句も指定すると、関連するオペレーティング・システム・ファイル（Oracle Managed Files も含む）を処理せずにそのままにしておくことができます。この句を指定する必要があるのは、Oracle Managed Files を使用しているときに、関連するオペレーティング・システム・ファイルを INCLUDING CONTENTS 句で削除しない場合です。

CASCADE CONSTRAINTS

CASCADE CONSTRAINTS を指定すると、*tablespace* に含まれる表の主キーまたは一意キーを参照する、*tablespace* の外の表からすべての参照整合性制約を削除できます。このような参照整合性制約があるときにこの句を省略した場合、エラーが戻され、表領域は削除されません。

例

表領域の削除例： 次の文は、tbs_01 表領域を削除し、tbs_01 に含まれる主キーおよび一意キーを参照するすべての参照整合性制約を削除します。

```
DROP TABLESPACE tbs_01
    INCLUDING CONTENTS
        CASCADE CONSTRAINTS;
```

オペレーティング・システム・ファイルの削除例： 次の例は、tbs_02 表領域およびそれに関連するすべてのオペレーティング・システムのデータ・ファイルを削除します。

```
DROP TABLESPACE tbs_02
   INCLUDING CONTENTS AND DATAFILES;
```

DROP TRIGGER

用途

トリガーは PL/SQL を使用して定義されます。トリガーの作成、変更および削除の詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

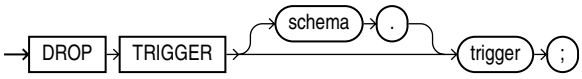
DROP TRIGGER 文を使用すると、データベースからデータベース・トリガーを削除できます。

前提条件

削除するトリガーが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY TRIGGER システム権限が必要です。他のユーザーのスキーマ内のデータベースにあるトリガーを削除する場合は、ADMINISTER DATABASE TRIGGER システム権限が必要です。

構文

***drop_trigger*::=**



セマンティクス

schema

トリガーが含まれているスキーマを指定します。*schema* を指定しない場合、トリガーは自分のスキーマ内にあるとみなされます。

trigger

削除するトリガーの名前を指定します。データベースからトリガーが削除され、再度、起動されることはありません。

例

トリガーの削除例： 次の文は、hr スキーマ内の salary_check トリガーを削除します。

```
DROP TRIGGER hr.salary_check;
```

DROP TYPE

用途

オブジェクト型は PL/SQL を使用して定義されます。オブジェクト型の作成、変更および削除の詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

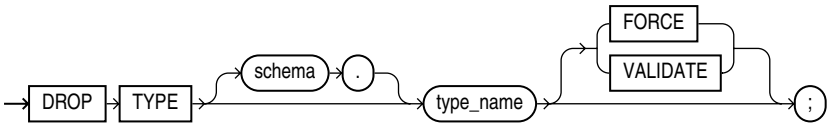
DROP TYPE 文を使用すると、オブジェクト型、VARRAY 型またはネストした表型の仕様部および本体を削除できます。

前提条件

削除するオブジェクト型、VARRAY 型またはネストした表型が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY TYPE システム権限が必要です。

構文

***drop_type*::=**



セマンティクス

schema

型が含まれているスキーマを指定します。*schema* を指定しない場合、型は自分のスキーマ内にあるとみなされます。

type_name

削除するオブジェクト型、VARRAY 型またはネストした表型の名前を指定します。型依存性または表依存性のない型のみ削除できます。

type_name がスーパータイプの場合、FORCE も指定しないと、この文は正常に実行されません。FORCE を指定すると、そのスーパータイプに依存するすべてのサブタイプが無効になります。

type_name が統計タイプの場合、FORCE も指定しないと、この文は正常に実行されません。FORCE を指定した場合、最初に *type_name* に関連付けられたすべてのオブジェクトの関連付けが解除され、*type_name* が削除されます。

> **参照：** 統計タイプの詳細は、13-21 ページの「ASSOCIATE STATISTICS」および 17-30 ページの「DISASSOCIATE STATISTICS」を参照してください。

type_name が統計タイプに関連付けられたオブジェクト型の場合、最初にこの統計タイプから *type_name* の関連付けが解除され、*type_name* が削除されます。ただし、統計タイプを使用して統計情報が収集された場合、この統計タイプから *type_name* の関連付けを解除することはできません。このため、この文は正常に実行されません。

type_name が索引タイプの実装タイプの場合、索引タイプに INVALID のマークが付けられます。

type_name にパブリック・シノニムが定義されている場合、このシノニムも削除されます。

FORCE オプションを指定していない場合に削除できるのは、依存性のないスタンドアロンのスキーマ・オブジェクトとして定義されているオブジェクト型またはネストした表型または VARRAY 型のみです。これはデフォルトの動作です。

参照：「CREATE INDEXTYPE」（14-73 ページ）

FORCE

FORCE を指定すると、依存するデータベース・オブジェクトを持つ型でも強制的に削除できます。削除する型に依存するすべての列に UNUSED のマークが付けられ、それらの列にアクセスできなくなります。

注意： FORCE を使用して、依存性のある型を削除することはお薦めしません。この操作はリカバリ不能であり、依存表または列のデータにアクセスできなくなる場合があります。

VALIDATE

型を削除するときに VALIDATE を指定すると、格納されているこの型のインスタンスがスーパータイプのいずれかの置換可能な列の範囲であることが検証されます。このようなインスタンスがない場合、削除操作が完了します。

この句はサブタイプのみに意味があります。明示的な型または表依存性のないサブタイプを安全に削除するために、このオプションの使用をお薦めします。

例

オブジェクト型の削除例：

次の文は、オブジェクト型 person_t を削除します。このオブジェクト型を作成する例については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。person_t に依存するすべての列は、UNUSED のマークが付けられ、アクセスできなくなります。

```
DROP TYPE person_t FORCE;
```

DROP TYPE BODY

用途

オブジェクト型は PL/SQL を使用して定義されます。オブジェクト型の作成、変更および削除の詳細は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

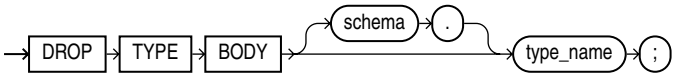
DROP TYPE BODY 文を使用すると、オブジェクト型、VARRAY 型またはネストした表型の本体を削除できます。型本体を削除しても、オブジェクト型の仕様部は残ります。また、削除した型本体は再作成できます。その本体を再作成する場合、型本体を削除したオブジェクト型は引き続き使用できますが、メンバー・ファンクションはコールできません。

前提条件

オブジェクト型の本体が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY TYPE システム権限が必要です。

構文

***drop_type_body*::=**



セマンティクス

schema

オブジェクト型が含まれているスキーマを指定します。*schema* を指定しない場合、オブジェクト型は自分のスキーマ内にあるとみなされます。

type_name

削除するオブジェクト型の本体の名前を指定します。

型本体の削除の制限事項： 型依存性および表依存性がない場合にのみ型の本体を削除できます。

例

オブジェクト型の本体の削除例： 次の文は、オブジェクト型本体 data_typ1 を削除します。このオブジェクト型を作成する例については、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

```
DROP TYPE BODY data_typ1;
```

DROP USER

用途

DROP USER 文を使用すると、データベース・ユーザーを削除できます。また、オプションでユーザーのオブジェクトを削除することもできます。

自動ストレージ管理クラスタでは、AS SYSASM と認証されたユーザーは、この句を使用して、現行のノードの自動ストレージ管理インスタンスに対してローカルなパスワード・ファイルでユーザーを削除できます。

ユーザーを削除すると、そのユーザーのすべてのスキーマ・オブジェクトもごみ箱から消去されます。

注意： SYS ユーザーまたは SYSTEM ユーザーを削除しないでください。これらのユーザーを削除すると、データベースが破損します。

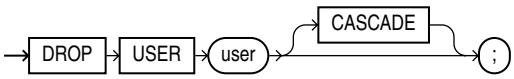
参照： ユーザーの作成および変更の詳細は、17-7 ページの「CREATE USER」および 13-5 ページの「ALTER USER」を参照してください。

前提条件

DROP USER システム権限が必要です。自動ストレージ管理クラスタでは、AS SYSASM と認証される必要があります。

構文

***drop_user*::=**



セマンティクス

user

削除するユーザーを指定します。CASCADE を指定しない場合、またはユーザーのオブジェクトを最初に明示的に削除しない場合、所有するスキーマにオブジェクトが含まれているユーザーは削除されません。

CASCADE

CASCADE を指定すると、ユーザーを削除する前に、そのユーザーのスキーマ内にあるすべてのオブジェクトを削除できます。所有するスキーマにオブジェクトが含まれているユーザーを削除する場合は、必ずこの句を指定します。

- ユーザーのスキーマに表がある場合、表が削除され、その表の主キーまたは一意キーを参照している他のユーザーのスキーマ内にある表の参照整合性制約も自動的に削除されます。
- この句で表が削除される場合、その表の列で作成されたすべてのドメイン索引も削除され、適切な削除ルーチンが起動されます。

  参照： これらのルーチンの詳細は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。

- 他のスキーマにある次のオブジェクトは削除されず、無効にされます。
  - 削除されたユーザーのスキーマ内のビューまたはシノニム
  - 削除されたユーザーのスキーマ内のオブジェクトを問い合せるストアド・プロシージャ、ファンクションまたはパッケージ
- 削除されたユーザーのスキーマ内にある表に基づく他のスキーマ内のマテリアライズド・ビューは削除されません。ただし、実表は存在しないため、他のスキーマ内のマテリアライズド・ビューはリフレッシュできなくなります。
- ユーザーのスキーマにあるすべてのトリガーは削除されます。
- ユーザーが作成したロールは削除されません。

注意： Oracle Database では、FORCE を使用するとユーザーが所有するすべての型が削除されます。詳細は、18-13 ページの「DROP TYPE」の FORCE キーワードを参照してください。

例

データベース・ユーザーの削除例： 次の文は、ユーザー sidney のスキーマ内にオブジェクトがない場合に、sidney を削除します。

```
DROP USER sidney;
```

sidney のスキーマ内にオブジェクトがある場合は、次の文のように CASCADE 句を指定して、sidney とそのオブジェクトを削除する必要があります。

```
DROP USER sidney CASCADE;
```

DROP VIEW

用途

DROP VIEW 文を使用すると、データベースからビューまたはオブジェクト・ビューを削除できます。ビューを削除して再作成すると、ビューの定義を変更できます。

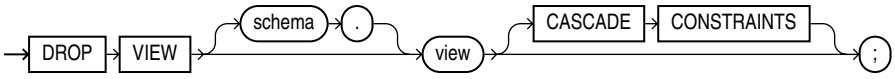
> **参照：** ビューの作成および変更の詳細は、17-13 ページの「CREATE VIEW」および 13-12 ページの「ALTER VIEW」を参照してください。

前提条件

削除するビューが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、DROP ANY VIEW システム権限が必要です。

構文

***drop_view*::=**



セマンティクス

schema

ビューが含まれているスキーマを指定します。*schema* を指定しない場合、ビューは自分のスキーマ内にあるとみなされます。

view

削除するビューの名前を指定します。

Oracle Database では、ビューに依存するビュー、マテリアライズド・ビューおよびシノニムは削除されませんが、INVALID のマークが付けられます。このようなビューおよびシノニムは削除または再定義するか、無効なビューやシノニムをもう一度有効にするビューを別に定義します。

指定するビューにサブビューが定義されている場合、サブビューも同様に無効になります。ビューがサブビューを持つかどうかを確認するには、USER_VIEWS、ALL_VIEWS または DBA_VIEWS データ・ディクショナリ・ビューの SUPERVIEW_NAME 列を問い合せます。

> **参照：**
>
> - 16-6 ページの「CREATE TABLE」および 16-2 ページの「CREATE SYNONYM」を参照してください。
> - 無効なマテリアライズド・ビューの再検証の詳細は、11-2 ページの「ALTER MATERIALIZED VIEW」を参照してください。

CASCADE CONSTRAINTS

CASCADE CONSTRAINTS を指定すると、削除するビューの主キーまたは一意キーを参照するすべての参照整合性制約を削除できます。このような制約が存在する状態でこの句を省略すると、DROP 文は正常に実行されません。

例

ビューの削除例： 次の文は、emp_view ビュー（17-20 ページの「ビューの作成例：」で作成）を削除します。

```
DROP VIEW emp_view;
```

EXPLAIN PLAN

用途

EXPLAIN PLAN 文を使用すると、指定した SQL 文を実行するために Oracle Database が使用する実行計画を決定できます。この文によって、実行計画の各手順を記述している行が、指定した表に挿入されます。SQL トレース機能の一部として EXPLAIN PLAN 文を発行することもできます。

この文によって、文を実行するコストも決まります。表にドメイン索引が定義されている場合、ユーザー定義の CPU および I/O コストが挿入されます。

配布メディアの SQL スクリプトの中に、サンプル出力表 PLAN_TABLE の定義があります。使用する出力表は、列の名前およびデータ型がこのサンプル表と同じである必要があります。このスクリプトの一般的な名前は、UTLXPLAN.SQL です。正確な名前および位置は、使用するオペレーティング・システムによって異なります。

Oracle Database キャッシュされたカーソルに関する情報は、次の動的パフォーマンス・ビューから得られます。

- SQL カーソルが使用する作業領域の詳細は、V$SQL_WORKAREA を問い合せます。
- キャッシュされたカーソルの実行計画の詳細は、V$SQL_PLAN を問い合せます。
- キャッシュされたカーソルの各ステップでの実行統計または実行計画の操作（生成された行数、読み取られたブロック数など）は、V$SQL_PLAN_STATISTICS を問い合せます。
- 前述の 3 つのビューの処理結果を 1 つにまとめた情報を取得する場合は、V$SQL_PLAN_STATISTICS_ALL を問い合せます。
- この文の実行が監視されている場合、キャッシュされたカーソルの実行計画の各ステップまたは各操作での実行統計は、V$SQL_PLAN_MONITOR に表示されます。MONITOR ヒントを使用すると、監視を強制的に適用できます。

参照：

- EXPLAIN PLAN の出力、SQL トレース機能の使用方法、および実行計画の生成と解析方法については、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。
- 動的パフォーマンス・ビューの詳細は、『Oracle Database リファレンス』を参照してください。

前提条件

EXPLAIN PLAN 文を実行する場合、実行計画を格納する既存の出力表に行を挿入するための権限が必要です。

出力する実行計画の適用対象である SQL 文を実行するための権限も必要です。この SQL 文でビューにアクセスする場合は、このビューの基礎になっているすべての表およびビューへのアクセス権限が必要です。このビューが別のビューに基づき、さらにこの別のビューが、ある表に基づいている場合、別のビューとそのビューの基礎になっている表へのアクセス権限が必要です。

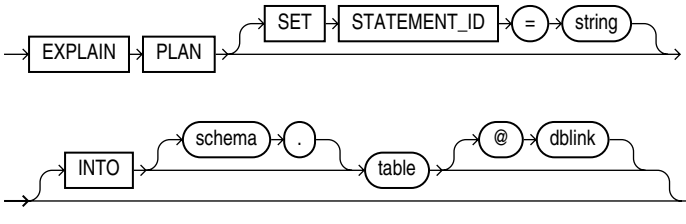
EXPLAIN PLAN で作成された実行計画を検証する場合は、出力表へ問い合せる権限が必要です。

EXPLAIN PLAN 文はデータ操作言語（DML）文であり、データ定義言語（DDL）文ではありません。そのため、EXPLAIN PLAN 文で加えられた変更内容は暗黙的にコミットされません。出力表の EXPLAIN PLAN 文で生成された行を保存する場合は、この文を指定したトランザクションをコミットする必要があります。

参照： 計画表の移入および問合せに必要な権限の詳細は、18-53 ページの「INSERT」および 19-4 ページの「SELECT」を参照してください。

構文

***explain_plan*::=**



セマンティクス

SET STATEMENT_ID 句

実行計画が出力される表の中で、この実行計画に該当する行にある STATEMENT_ID 列の値を指定します。この値によって、実行計画の行を出力表の中の他の行と区別できます。ユーザーの出力表に多数の実行計画の行が含まれている場合は、必ず、STATEMENT_ID の値を指定します。この句を省略した場合、デフォルトで STATEMENT_ID 値が NULL に設定されます。

INTO *table* 句

出力表の名前を指定し、オプションとしてそのスキーマおよびデータベースの名前も指定します。この表は、EXPLAIN PLAN 文を使用する前に作成しておく必要があります。

schema を指定しない場合、表は自分のスキーマ内にあるとみなされます。

dblink には、出力表が格納されているリモートの Oracle Database に対するデータベース・リンクの完全な名前または名前の一部を指定します。Oracle Database の分散機能を使用している場合にのみ、リモート出力表を指定できます。*dblink* を省略した場合、表がローカル・データベース上にあるとみなされます。データベース・リンクの参照方法の詳細は、2-104 ページの「リモート・データベース内のオブジェクトの参照」を参照してください。

INTO 句を省略した場合、出力表は、ローカル・データベース上の自分のスキーマ内にある PLAN_TABLE であるとみなされます。

FOR *statement* 句

実行計画生成の対象となる SELECT、INSERT、UPDATE、DELETE、CREATE TABLE、CREATE INDEX または ALTER INDEX ... REBUILD 文を指定します。

実行計画の注意事項： EXPLAIN PLAN には、次の注意事項があります。

- *statement* に *parallel_clause* を指定した場合、結果として生成される実行計画はパラレルで実行されます。ただし、EXPLAIN PLAN コマンドによって計画表に実際に文が挿入されるため、ユーザーが発行したパラレル DML 文は、トランザクションの最初の DML 文ではなくなります。これは、トランザクション 1 つにつきパラレル DML 文は 1 つという Oracle Database の制限に違反するため、この文はシリアルで実行されます。文をパラレルで実行するには、EXPLAIN PLAN 文をコミットまたはロールバックしてから、パラレル DML 文を発行する必要があります。
- 一時表上で操作するための実行計画を判断する場合、EXPLAIN PLAN は、同一セッションから実行する必要があります。これは、一時表内のデータがセッション固有であるためです。

例

実行計画の例： 次の文は、UPDATE 文の実行計画およびコストを決定し、実行計画を記述した行を STATEMENT_ID 値 'Raise in Tokyo' とともに、指定した plan_table 表に挿入します。

```
EXPLAIN PLAN
    SET STATEMENT_ID = 'Raise in Tokyo'
    INTO plan_table
    FOR UPDATE employees
        SET salary = salary * 1.10
        WHERE department_id =
           (SELECT department_id FROM departments
               WHERE location_id = 1200);
```

次の SELECT 文は、plan_table 表への問合せを実行し、実行計画およびコストを戻します。

```
SELECT LPAD(' ',2*(LEVEL-1))||operation operation, options,
  object_name, position
    FROM plan_table
    START WITH id = 0 AND statement_id = 'Raise in Tokyo'
    CONNECT BY PRIOR id = parent_id AND statement_id = 'Raise in Tokyo'
    ORDER BY operation, options, object_name, position;
```

問合せによって、次の実行計画が戻されます。

```
OPERATION            OPTIONS         OBJECT_NAME       POSITION
-------------------- --------------- --------------- ----------
UPDATE STATEMENT                                             2
 UPDATE                              EMPLOYEES               1
  TABLE ACCESS       FULL            EMPLOYEES               1
   VIEW                              index$_join$_00         1
                                     2
    HASH JOIN                                                1
     INDEX           RANGE SCAN      DEPT_LOCATION_I         1
                                     X
     INDEX           FAST FULL SCAN  DEPT_ID_PK              2
```

POSITION 列の 1 行目の値は、文のコストが 2 であることを示しています。

実行計画のパーティション化例： サンプル表 sh.sales は、time_id 列でパーティション化されています。パーティション sales_q3_2000 には、2000 年 10 月 1 日より小さい時刻の値があり、time_id 列にローカル索引 sales_time_bix があります。

次の問合せを考えてみます。

```
EXPLAIN PLAN FOR
   SELECT * FROM sales
      WHERE time_id BETWEEN :h AND '01-OCT-2000';
```

ここで、:h はすでに宣言されているバインド変数を指します。EXPLAIN PLAN 文は、PLAN_TABLE を出力表とする次の問合せを実行します。次の問合せによって、パーティション情報などの基本的な実行計画が得られます。

```
SELECT operation, options, partition_start, partition_stop,
   partition_id FROM plan_table;
```

FLASHBACK DATABASE

用途

FLASHBACK DATABASE 文を使用すると、データベースを過去の時点またはシステム変更番号（SCN）まで戻すことができます。この文を使用すると、データベースの不完全リカバリと同じ操作をより高速に実行できます。

FLASHBACK DATABASE 操作の後、フラッシュバックされたデータベースに書込みアクセスするためには、ALTER DATABASE OPEN RESETLOGS 文によってそのデータベースを再オープンする必要があります。

> **参照：** FLASHBACK DATABASE の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

前提条件

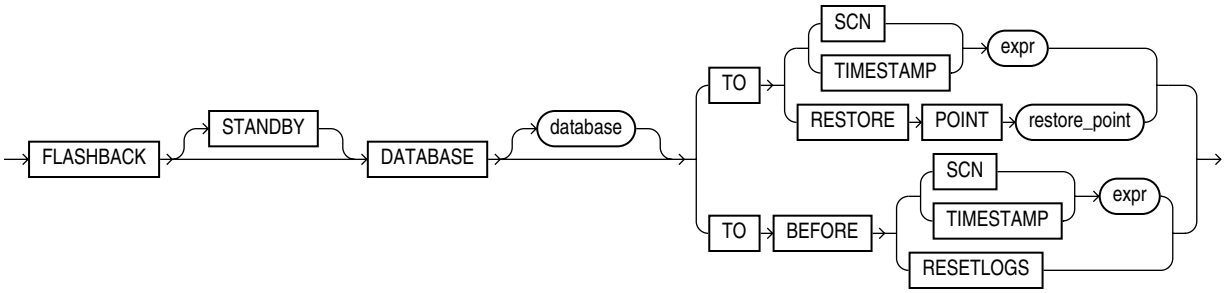
SYSDBA システム権限が必要です。フラッシュ・リカバリ領域をデータベースに設定しておく必要があります。また、データベースを保証付きリストア・ポイントにフラッシュバックするのでなければ、ALTER DATABASE FLASHBACK ON 文を使用して、データベースを FLASHBACK モードにしておく必要があります。データベースは、マウントされているがオープンされていない状態であることが必要です。また、次のことも必要です。

- データベースは、ARCHIVELOG モードで実行されている必要があります。
- データベースは、現行の制御ファイルとともにマウントされている必要がありますが、オープンされている必要はありません。制御ファイルは、バックアップおよび再作成することはできません。データベース制御ファイルがバックアップからリストアされるか、または再作成されると、既存のすべてのフラッシュバック・ログ情報は廃棄されます。
- データベースには、SQL 文 ALTER TABLESPACE ... FLASHBACK OFF を使用して、フラッシュバック機能が使用禁止にされたオンライン表領域は含めないでください。

> **参照：**
>
> - データベースを FLASHBACK モードにする方法については、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』および 10-39 ページの「ALTER DATABASE」の「flashback_mode_clause」を参照してください。
> - リストア・ポイントおよび保証付きリストア・ポイントの詳細は、15-53 ページの「CREATE RESTORE POINT」を参照してください。

構文

***flashback_database*::=**



セマンティクス

FLASHBACK DATABASE 文を発行すると、最初に、必要なすべてのアーカイブ REDO ログおよびオンライン REDO ログが使用可能であるかどうかが確認されます。これらのログが使用可能な場合、データベース内で現在オンラインになっているすべてのデータ・ファイルが、この文に指定した SCN または時刻まで戻されます。

- データベースに保持されるフラッシュバック・データの量は、DB_FLASHBACK_RETENTION_TARGET 初期化パラメータおよびフラッシュ・リカバリ領域のサイズによって制御されます。V$FLASHBACK_DATABASE_LOG ビューを問い合せることによって、どの時点までデータベースをフラッシュバックできるかを判断できます。
- フラッシュバックを実行するために十分なデータがデータベースに残っていない場合、標準のリカバリ手順によってデータベースを過去のある時点の状態にリカバリできます。
- 一連のデータ・ファイルに十分なデータが残っていない場合、エラーが戻されます。その場合、それらのデータ・ファイルをオフラインにし、この文を再実行してデータベースの残りのデータ・ファイルをリカバリできます。その後、標準のリカバリ手順を使用して、オフラインにしたデータ・ファイルをリカバリできます。

> **参照：** データ・ファイルのリカバリの詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

STANDBY

STANDBY を指定すると、スタンバイ・データベースを以前の SCN または時点まで戻すことができます。スタンバイ・データベースではない場合、エラーが戻されます。この句を省略すると、*database* には、プライマリ・データベースまたはスタンバイ・データベースのどちらでも指定できます。

> **参照：** スタンバイ・データベースで FLASHBACK DATABASE を使用して複数の遅延を実現する方法については、『Oracle Data Guard 概要および管理』を参照してください。

TO SCN 句

システム変更番号（SCN）を指定します。

- TO SCN では、指定した SCN の状態にデータベースが戻されます。
- TO BEFORE SCN では、指定した SCN の直前の SCN の状態にデータベースが戻されます。

現行の SCN を判断するには、V$DATABASE ビューの CURRENT_SCN 列を問い合せます。これによって、高リスクのバッチ・ジョブを実行する前などに、SCN をスプール・ファイルに保存することもできます。

TO TIMESTAMP 句

有効な日時式を指定します。

- TO TIMESTAMP では、指定したタイムスタンプ時点の状態にデータベースが戻されます。
- TO BEFORE TIMESTAMP では、指定したタイムスタンプの 1 秒前の状態にデータベースが戻されます。

タイムスタンプには、基準の値（SYSDATE など）からのオフセットか、またはシステム・タイムスタンプの絶対値を指定できます。

TO RESTORE POINT 句

この句を使用すると、指定したリストア・ポイントにデータベースをフラッシュバックできます。フラッシュバック・データベースが使用可能になっていない場合、この FLASHBACK DATABASE 文では、この句のみを指定できます。データベースのモードが FLASHBACK でない場合、前述の「前提条件」で示したように、これは、この文で指定可能な唯一の句です。

RESETLOGS

TO BEFORE RESETLOGS を指定すると、データベースを最後のリセットログ操作（ALTER DATABASE OPEN RESETLOGS）の直前の状態にフラッシュバックできます。

> **参照：** この句の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

例

データベースにフラッシュ・リカバリ領域が設定してあり、メディア・リカバリが使用可能になっていると想定します。次の文は、データベースの FLASHBACK モードを有効にして、データベースをオープンします。

```
STARTUP MOUNT
ALTER DATABASE FLASHBACK ON;
ALTER DATABASE OPEN;
```

次の文は、データベースを 1 日以上オープンしていた場合、データベースを 1 日フラッシュバックします。

```
SHUTDOWN DATABASE
STARTUP MOUNT
FLASHBACK DATABASE TO TIMESTAMP SYSDATE-1;
```

FLASHBACK TABLE

用途

FLASHBACK TABLE 文を使用すると、人為的エラーまたはアプリケーション・エラーが発生した場合に、表を以前の状態にリストアできます。表をフラッシュバックできる過去の時点は、システム内の UNDO データの量によって異なります。また、Oracle Database では、表の構造を変更する DDL 操作が行われた場合は、表を以前の状態にリストアできません。

注意： UNDO_MANAGEMENT 初期化パラメータをデフォルトの AUTO のままにして、データベースを自動 UNDO モードで実行することをお薦めします。また、UNDO_RETENTION 初期化パラメータを、必要となる最も古いデータを含むために十分長い期間に設定してください。詳細は、UNDO_MANAGEMENT 初期化パラメータおよび UNDO_RETENTION 初期化パラメータに関するドキュメントを参照してください。

FLASHBACK TABLE 文はロールバックできません。ただし、もう 1 つ FLASHBACK TABLE 文を発行し、現在の時間の直前の時間を指定することはできます。このため、FLASHBACK TABLE 句を発行する前に、現在の SCN を記録しておくことをお薦めします。

参照：

- データベース全体を以前の状態に戻す方法の詳細は、18-22 ページの「FLASHBACK DATABASE」を参照してください。
- 表から過去のデータを取り出す方法の詳細は、19-15 ページの「SELECT」の「flashback_query_clause」を参照してください。
- FLASHBACK TABLE 文を使用する方法の詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

前提条件

表を以前の SCN またはタイムスタンプまでフラッシュバックするには、その表に対する FLASHBACK オブジェクト権限か、FLASHBACK ANY TABLE システム権限が必要です。また、その表に対する SELECT、INSERT、DELETE および ALTER オブジェクト権限が必要です。

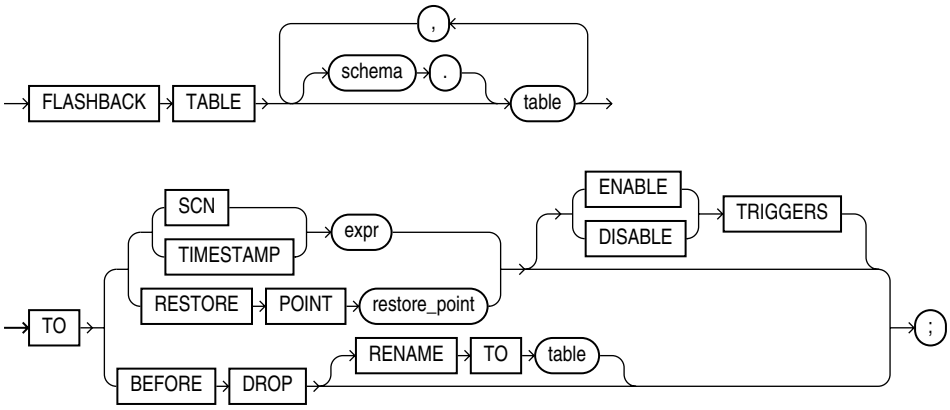
TO BEFORE DROP を使用せずに表をフラッシュバックする場合、フラッシュバック・リスト内のすべての表で、行の移動を有効にする必要があります。この操作は**フラッシュバック削除**といい、UNDO データではなく、ごみ箱に削除されたデータを使用します。行の移動の有効化の詳細は、10-39 ページの「row_movement_clause」を参照してください。

表をリストア・ポイントにフラッシュバックするには、SELECT ANY DICTIONARY または FLASHBACK ANY TABLE のいずれかのシステム権限か、SELECT_CATALOG_ROLE ロールが必要です。

DROP TABLE 操作の前まで表をフラッシュバックするために必要な権限は、その表の削除に必要な権限のみです。

構文

***flashback_table*::=**



セマンティクス

Oracle Flashback Table の操作中は、フラッシュバック・リストに指定されたすべての表が、排他 DML ロックによってロックされます。これらのロックによって、表を以前の状態に戻す操作の間に、それらの表に他の操作が行われなくなります。

表のフラッシュバック操作は、フラッシュバック・リストに指定された表の数に関係なく、1 回のトランザクションで実行されます。すべての表が以前の状態に戻されるか、何も戻されないかのいずれかです。いずれかの表のフラッシュバック操作が正常に実行されなかった場合、この文全体が正常に実行されません。

表のフラッシュバック操作が完了すると、表内のデータは、指定した過去の状態のものになっています。ただし、`FLASHBACK TABLE TO SCN` または `FLASHBACK TABLE TO TIMESTAMP` では行 ID が保持されず、`FLASHBACK TABLE TO BEFORE DROP` では参照制約がリカバリされません。

表に関連付けられた統計情報は、以前の形式には戻されません。表の現行の索引は戻され、フラッシュバック時点での表の状態が反映されます。現行の索引がフラッシュバック時点で存在していなかった場合、その索引は、フラッシュバック時点の表の状態を反映するように更新されます。ただし、フラッシュバック時点から現時点の間に削除された索引は、リストアされません。

schema

表が含まれているスキーマを指定します。*schema* を指定しない場合、表は自分のスキーマ内にあるとみなされます。

table

以前の状態に戻すデータを含む 1 つ以上の表の名前を指定します。

表のフラッシュバックの制限事項： この文には、次の制限事項があります。

- 表のフラッシュバック操作は、クラスタ内の表、マテリアライズド・ビュー、アドバンスト・キューイング（AQ）表、静的データ・ディクショナリ表、システム表、リモート表、オブジェクト表、ネストした表、表の個々のパーティションまたはサブパーティションには無効です。
- 表のアップグレード、移動、切捨て、表への制約の追加、クラスタへの表の追加、列の変更または削除、列の暗号化キーの変更、パーティションまたはサブパーティションの追加、削除、マージ、分割、結合、切捨て（レンジ・パーティションの追加を除く）の DDL 操作を実行すると表の構造が変更されるため、これらの操作後に `TO SCN` または `TO TIMESTAMP` 句を使用して表をその操作以前の時点にフラッシュバックすることはできません。

TO SCN 句

表を戻す時点に対応するシステム変更番号（SCN）を指定します。*expr* は、有効な SCN に評価される数値である必要があります。

TO TIMESTAMP 句

表を戻す時点に対応するタイムスタンプ値を指定します。*expr* は、過去の有効なタイムスタンプに評価される必要があります。表は、指定したタイムスタンプの約 3 秒以内の時点にフラッシュバックされます。

TO RESTORE POINT 句

表をフラッシュバックするリストア・ポイントを指定します。リストア・ポイントは作成済である必要があります。

> **参照：** リストア・ポイントの作成の詳細は、15-53 ページの「CREATE RESTORE POINT」を参照してください。

ENABLE | DISABLE TRIGGERS

デフォルトでは、表のフラッシュバック操作中は、*table* に定義したすべての有効なトリガーが無効にされ、表のフラッシュバック操作の完了後に再度有効にされます。このデフォルト動作を上書きして、フラッシュバック処理中もトリガーを有効にする必要がある場合、ENABLE TRIGGERS を指定します。

この句は、*table* に定義され、すでに有効にされているデータベース・トリガーのみに影響します。現在無効になっているトリガーを選択して有効にするには、ALTER TABLE ... *enable_disable_clause* を使用してから、FLASHBACK TABLE 文に ENABLE TRIGGERS 句を指定して発行します。

TO BEFORE DROP 句

この句を使用すると、削除された表およびすべての依存するオブジェクトをごみ箱から取り出すことができます。表は、SYSTEM 表領域以外のローカル管理表領域内に置いておく必要があります。

> **参照：**
>
> - ごみ箱の詳細、およびごみ箱内のオブジェクトのネーミング規則の詳細は、『Oracle Database 管理者ガイド』を参照してください。
> - ごみ箱からオブジェクトを完全に削除する方法の詳細は、18-80 ページの「PURGE」を参照してください。

ユーザーが指定した元の表名か、オブジェクトの削除時にそのオブジェクトに割り当てられたシステム生成名を指定できます。

- ごみ箱内のオブジェクトのシステム生成名は一意です。そのため、システム生成名を指定すると、その名前を持つ特定のオブジェクトが取り出されます。

  ごみ箱の内容を参照するには、USER_RECYCLEBIN データ・ディクショナリ・ビューを問い合せます。かわりに RECYCLEBIN シノニムを使用することもできます。次の 2 つの文は、同じ行を戻します。

  ```
  SELECT * FROM RECYCLEBIN;
  SELECT * FROM USER_RECYCLEBIN;
  ```

- ユーザーが指定した名前を指定し、その名前を持つオブジェクトがごみ箱内に 1 つ以上存在した場合、ごみ箱に移動した日時が一番近いオブジェクトが取り出されます。その表のより古い状態で取り出す場合、次のいずれかの操作を実行します。
  - 取り出す表の、ごみ箱内でのシステム生成名を指定します。
  - 必要な表が取り出されるまで FLASHBACK TABLE ... TO BEFORE DROP 文を繰り返し発行します。

Oracle Database は、元の表名を保持します。元の表が削除された後、同じスキーマ内に削除された表と同じ名前を持つ新しい表が作成されていた場合、RENAME TO 句も指定しないかぎりエラーが戻されます。

RENAME TO 句 この句を使用すると、ごみ箱から取り出される表に新しい名前を指定できます。

削除された表のフラッシュバックの注意事項： 削除された表のフラッシュバックには、次の注意事項があります。

- ごみ箱から取り出された表に定義されたすべての索引（ビットマップ結合索引以外）が取り出されます。（ビットマップ結合索引は DROP TABLE 操作中にごみ箱に移動しないため、取り出すことができません。）
- 表に定義されたすべてのトリガーおよび制約も取り出されます。ただし、他の表を参照する参照整合制約は取り出されません。

  取り出された索引、トリガーおよび制約には、ごみ箱での名前が付けられています。そのため、FLASHBACK TABLE ... TO BEFORE DROP 文を発行する前に、USER_RECYCLEBIN ビューを問い合せて、取り出されたトリガーおよび制約の名前を使用しやすい名前に変更することをお薦めします。
- 表を削除すると、その表に定義されたすべてのマテリアライズド・ビュー・ログも削除されますが、これらはごみ箱には移動しません。そのため、マテリアライズド・ビュー・ログを表とともにフラッシュバックすることはできません。
- 表を削除すると、その表の索引も削除され、表とともにごみ箱に移動します。その後領域が不足すると、領域を再利用するために、まずごみ箱内の索引が消去されます。この場合、表をフラッシュバックしても、その表に定義されていた索引の一部が戻されない場合があります。
- ユーザーによって、または領域の再利用操作の結果として消去された表は、フラッシュバックできません。

例

以前の状態への表のリストア例： 次の例では、新しい表 employees_test を行の移動を有効にして作成し、新しい表内の値を更新し、FLASHBACK TABLE 文を発行します。

hr サンプル・スキーマの employees 表から、employees_test 表を、行の移動を有効にして作成します。

```
CREATE TABLE employees_test
  AS SELECT * FROM employees;
```

指標として、2500 未満の給与をリストします。

```
SELECT salary
  FROM employees_test
  WHERE salary < 2500;

    SALARY
----------
      2400
      2200
      2100
      2400
      2200
```

注意： FLASHBACK TABLE 文によって使用されるマッピング表に SCN が伝播されるまで時間がかかるため、5 分以上待ってから次の文を発行してください。この例で、既存の表を使用した場合は、この待機時間は必要ありません。

表の行の移動を可能にします。

```
ALTER TABLE employees_test
   ENABLE ROW MOVEMENT;
```

給与が 2500 未満の従業員の給与を 10% 上げます。

```
UPDATE employees_test
  SET salary = salary * 1.1
  WHERE salary < 2500;

5 rows updated.
COMMIT;
```

2 つ目の指標として、10% の昇給後にも 2500 未満である給与をリストします。

```
SELECT salary
  FROM employees_test
  WHERE salary < 2500;

    SALARY
----------
      2420
      2310
      2420
```

employees_test 表を、現在のシステム時間より前の状態にリストアします。この例では、一連の例を迅速にテストできるように、1 分間（実際にはこのような短い期間は設定しない）を使用しています。通常の環境では、より長い期間が経過します。

```
FLASHBACK TABLE employees_test
  TO TIMESTAMP (SYSTIMESTAMP - INTERVAL '1' minute);
```

2500 未満の給与をリストします。前述の FLASHBACK TABLE 文を発行したため、このリストは 1 つ目の指標リストと一致しています。

```
SELECT salary
  FROM employees_test
  WHERE salary < 2500;

    SALARY
----------
      2400
      2200
      2100
      2400
      2200
```

削除された表の取出し例： 次の文は、誤って削除した pm.print_media 表を取り出します。

```
FLASHBACK TABLE print_media TO BEFORE DROP;
```

pm スキーマ内に別の print_media 表が作成されていた場合、RENAME TO 句を使用して、取り出された表の名前を変更します。

```
FLASHBACK TABLE print_media TO BEFORE DROP RENAME TO print_media_old;
```

複数回削除された従業員表を最も古い状態で取り出す必要がある場合、USER_RECYLEBIN 表を問い合せてシステム生成名を判断し、その名前を FLASHBACK TABLE 文で使用します。（ご使用のデータベースでのシステム生成名は、ここに示すものとは異なります。）

```
SELECT object_name, droptime FROM user_recyclebin
   WHERE original_name = 'PRINT_MEDIA';

OBJECT_NAME                    DROPTIME
------------------------------ -------------------
RB$$45703$TABLE$0              2003-06-03:15:26:39
RB$$45704$TABLE$0              2003-06-12:12:27:27
RB$$45705$TABLE$0              2003-07-08:09:28:01
```

GRANT

用途

GRANT 文を使用すると、次の権限またはロールを付与できます。

- ユーザーおよびロールに対するシステム権限。
- ユーザーおよびロールに対するロール。権限とロールは、ともにローカル、グローバルまたは外部になります。表 18-1 に、システム権限のリストを、操作対象のデータベース・オブジェクト別に示します。用意されているロールの詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。
- ユーザー、ロールおよび PUBLIC に対する、特定のオブジェクトに対するオブジェクト権限。表 18-2 に、オブジェクト権限とその権限によって許可される操作を示します。

データベース・ユーザーの認可の注意事項： データベース・ユーザーを認可する方法は、データベースや GRANT 文を介して行う以外にもあります。

- 多くの Oracle Database 権限は､提供される PL/SQL パッケージおよび Java パッケージを介して付与されます。これらの権限の詳細は、該当するパッケージのドキュメントを参照してください。
- オペレーティング・システムによっては、Oracle Database ユーザーに対して、初期化パラメータ OS_ROLES でロールを付与できます。オペレーティング・システム機能を使用してユーザーにロールを付与する場合、GRANT 文を使用してユーザーにシステム権限を付与したり、その他のロールにシステム権限およびロールを付与することはできますが、そのユーザーにロールを付与することはできません。

自動ストレージ管理の注意事項： AS SYSASM と認証されたユーザーは、この文を使用すると、システム権限 SYSASM、SYSOPER および SYSDBA を現行ノードの自動ストレージ管理パスワード・ファイルのユーザーに付与できます。

参照：

- ローカル、グローバルおよび外部権限の定義については、17-7 ページの「CREATE USER」および 15-56 ページの「CREATE ROLE」を参照してください。
- その他の認可方法および権限の詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。
- 権限付与の取消しについては、18-84 ページの「REVOKE」を参照してください。

前提条件

システム権限を付与にするには、次のいずれかの条件が満たされている必要があります。

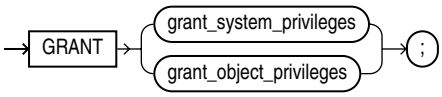
- GRANT ANY PRIVILEGE システム権限が付与されている必要があります。この場合、ロールがユーザーのセッションで有効になっていないかぎり、システム権限をロールに付与しても、ロールが付与されているユーザーに権限は付与されません。
- ADMIN OPTION 付きのシステム権限が付与されている必要があります。この場合、ロールがユーザーのセッションで有効になっているかどうかに関係なく、システム権限をロールに付与すると、ロールが付与されているユーザーに権限が付与されます。

ロールを付与する場合は、Admin Option 付きのロールが付与されているか、GRANT ANY ROLE システム権限が付与されているか、または付与するロールが自分で作成したロールである必要があります。

オブジェクト権限を付与する場合は、オブジェクトの所有者であるか、オブジェクトの所有者から Grant Option 付きのオブジェクト権限が付与されている必要があります。または、GRANT ANY OBJECT PRIVILEGE システム権限が付与されている必要があります。GRANT ANY OBJECT PRIVILEGE システム権限を持っている場合は、オブジェクトの所有者が付与可能なオブジェクト権限と同じオブジェクト権限のみを付与できます。その場合、DBA_TAB_PRIVS ビューの GRANTOR 列には、GRANT 文を発行したユーザーではなく、オブジェクトの所有者が表示されます。

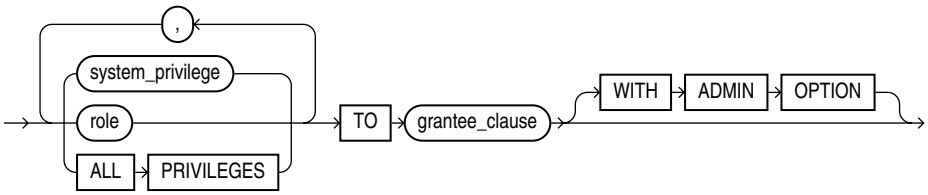
構文

***grant*::=**



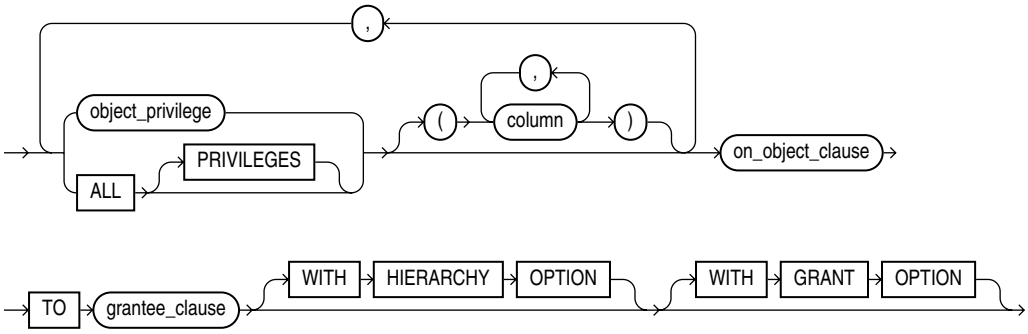
（18-32 ページの grant_system_privileges::=、18-32 ページの grant_object_privileges::= を参照）

grant_system_privileges::=



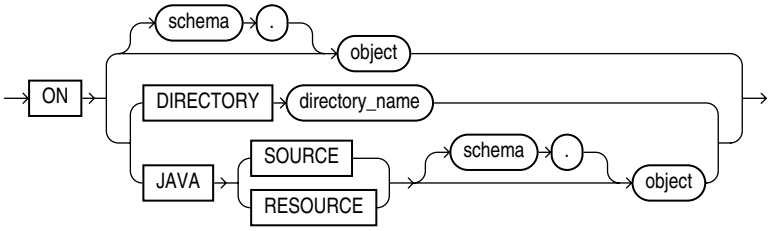
（18-33 ページの grantee_clause::= を参照）

grant_object_privileges::=

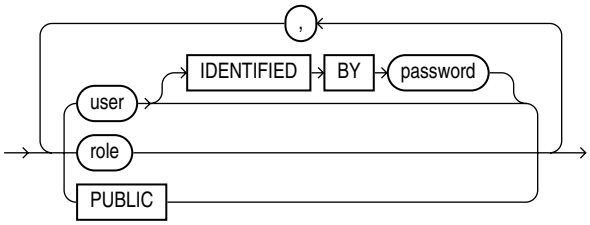


（18-33 ページの on_object_clause::=、18-33 ページの grantee_clause::= を参照）

on_object_clause::=



grantee_clause::=



セマンティクス

grant_system_privileges

システム権限を付与するには、次の句を使用します。

system_privilege

付与するシステム権限を指定します。表 18-1 に、システム権限のリストを、操作対象のデータベース・オブジェクト別に示します。

- **ユーザー**に権限を付与する場合、ユーザーの権限ドメインに権限が登録されます。このユーザーはその権限をすぐに使用できます。
- **ロール**に権限を付与する場合、ロールの権限ドメインに権限が登録されます。ロールが付与されているまたは使用可能となっているユーザーは、その権限をすぐに使用できます。なお、ロールが付与されている他のユーザーも、そのロールを使用可能にしてその権限を使用できます。

  参照： 18-50 ページの「ユーザーに対してシステム権限を付与する例 :」および 18-50 ページの「ロールに対してシステム権限を付与する例 :」を参照してください。

- **PUBLIC** に権限を付与する場合、各ユーザーの権限ドメインに権限が登録されます。すべてのユーザーは、その権限によって許可される操作をすぐに実行できます。

Oracle Database には、ALL PRIVILEGES というショートカットも用意されており、このショートカットを使用すると、SELECT ANY DICTIONARY 権限を除き、18-37 ページの表 18-1 に示すすべてのシステム権限を付与できます。

role

付与するロールを指定します。Oracle Database の事前定義ロールまたはユーザー定義ロールを付与できます。

- **ユーザー**にロールを付与する場合、そのユーザーが、付与されたロールを使用できます。ユーザーはそのロールをすぐに使用可能にし、ロールの権限ドメインに登録されている権限を使用できます。

注意： ユーザーにロールを付与する場合、ロールはユーザーのデフォルトのロールとして付与されるため、ログインするとすぐに使用可能になります。**保護アプリケーション・ロール**のセキュリティを確保するには、ロールをデフォルトのロールにしないようにする必要があります。ユーザーにアプリケーション・ロールを付与した直後に、DEFAULT ROLE ALL EXCEPT *role* 句を指定した ALTER USER 文を発行し、アプリケーション・ロールを指定します。これによって、このユーザーによる以降のログイン時に、認可済パッケージを使用するアプリケーションのみがこのロールを使用可能にできるというルールが適用されます。

また、ユーザーに保護アプリケーション・ロールを直接付与する必要はありません。ユーザーが適切なセキュリティ・ポリシーを渡すことを前提として、関連付けられた PL/SQL パッケージでこの処理を行うことができます。詳細は、15-57 ページの「USING package」の「CREATE ROLE セマンティクス」および『Oracle Database セキュリティ・ガイド』を参照してください。

- 他の**ロール**にロールを付与する場合、権限受領者のロールの権限ドメインに、権限付与者のロールの権限ドメインが登録されます。権限受領者のロールを付与されているユーザーは、それを使用可能にした場合、付与されたロールの権限ドメイン内の権限を使用できます。
- **PUBLIC** にロールを付与する場合、すべてのユーザーがロールを使用できます。すべてのユーザーはそのロールをすぐに使用可能にし、ロールの権限ドメインに登録されている権限を使用できます。

ユーザーに付与する各ロールは ALTER USER ... DEFAULT ROLE 文を発行するまでは、デフォルトのロールになります。この文によって、デフォルトのロールが指定されます。これ以外のすべてのロール（これまでに付与されたもの、今後付与するもの）は、もう一度 ALTER USER ... DEFAULT ROLE 文を使用してデフォルトに指定しないかぎり、デフォルトのロールにはなりません。

参照：

- Oracle の事前定義されているロールの詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。
- 「ロールに対してロールを付与する例：」（18-51 ページ）
- ユーザー定義ロールの作成については、15-56 ページの「CREATE ROLE」を参照してください。

IDENTIFIED BY 句

この句は、オブジェクト権限ではなくシステム権限を割り当てる場合にのみ有効です。IDENTIFIED BY 句を使用すると、パスワードによって既存ユーザーが明確に識別できるか、または存在しないユーザーを作成できます。この句は、権限受領者がロールまたは PUBLIC の場合は無効です。*grantee_clause* に指定したユーザーが存在しない場合、この句で指定したパスワードおよび権限とロール付きでユーザーが作成されます。

参照： ユーザー名およびパスワードの制限事項については、17-7 ページの「CREATE USER」を参照してください。

With Admin Option

With Admin Option を指定すると、権限受領者は次のことができます。

- ロールが GLOBAL ロールでない場合、権限またはロールを他のユーザーまたはロールに付与できます。
- 権限またはロールを他のユーザーまたは他のロールから取り消すことができます。

- 権限またはロールへのアクセスに必要な認可を変更するため、その権限またはロールを変更できます。
- 権限またはロールを削除します。

たとえば、With Admin Option を指定せずにユーザーにロールまたはシステム権限を付与し、後で With Admin Option を指定してその権限およびロールを付与した場合、そのユーザーはその権限またはロールに関して Admin Option を持つことになります。

システム権限またはロールの Admin Option をユーザーから取り消す場合、そのユーザーの権限またはロールを完全に取り消し、その次に Admin Option を指定せずに権限またはロールをユーザーに付与します。

参照： 「Admin Option 付きロールを付与する例 :」（18-50 ページ）

grantee_clause

TO *grantee_clause* を使用すると、システム権限、ロールまたはオブジェクト権限が付与されているユーザーまたはロールを識別できます。

権限受領者の制限事項： TO grantee_clause にユーザー、ロールおよび PUBLIC を指定できるのは 1 回のみです。

PUBLIC PUBLIC を指定すると、すべてのユーザーに権限を付与できます。

システム権限およびロールの付与の制限事項： 権限およびロールには、次の制限事項があります。

- 付与する権限およびロールのリストに、権限またはロールを指定できるのは 1 回のみです。
- ロールに対してそのロールと同じロールは付与できません。
- ロール IDENTIFIED GLOBALLY は、どのようなユーザーまたはロールに対しても付与できません。
- ロール IDENTIFIED EXTERNALLY は、グローバル・ユーザーまたはグローバル・ロールに対して付与できません。
- ロールは交互に付与できません。たとえば、ロール banker をロール teller に付与した場合、逆に teller を banker に付与することはできません。

grant_object_privileges

オブジェクト権限を付与するには、次の句を使用します。

object_privilege

付与するオブジェクト権限を指定します。表 18-2 に示すいずれかの値を指定します。

注意： ビューに対する SELECT を他のユーザーに付与するには、ビューの基礎となるすべてのオブジェクトを所有しているか、またはこれらすべての基礎となるオブジェクトに対する SELECT オブジェクト権限を WITH GRANT OPTION で付与されている必要があります。これは、これらの基礎となるオブジェクトに対する SELECT 権限を権限受領者がすでに持っている場合にも当てはまります。

オブジェクト権限の制限事項： 付与する権限のリストに権限を指定できるのは 1 回のみです。

ALL [PRIVILEGES]

ALL を指定すると、Grant Option 付きで付与されているオブジェクト権限に対するすべての権限を付与できます。オブジェクトが定義されているスキーマを所有しているユーザーは、自動的に Grant Option 付きのオブジェクトに対するすべての権限を持っています。キーワード PRIVILEGES はセマンティクスを明確にするためのものであり、指定は任意です。

column

権限を付与する表またはビューの列を指定します。INSERT、REFERENCES または UPDATE の各権限を付与する場合にのみ、列を指定できます。列を指定しない場合、権限受領者には表またはビューのすべての列に対して指定した権限が付与されます。

既存の列オブジェクトに対して付与された権限については、データ・ディクショナリ・ビュー USER_COL_PRIVS、ALL_COL_PRIVS または DBA_COL_PRIVS を問い合せます。

> **参照：** データ・ディクショナリ・ビューの詳細は、『Oracle Database リファレンス』および 18-51 ページの「別々の列に複数のオブジェクト権限を付与する例：」を参照してください。

on_object_clause

on_object_clause を指定すると、権限が付与されているオブジェクトを識別できます。ディレクトリ・スキーマ・オブジェクト、Java ソースおよびリソース・スキーマ・オブジェクトは、異なるネームスペースに格納されるため、別々に識別されます。

object の所有者でもなく、*object* WITH GRANT OPTION で *object_privilege* が付与されてもいないが、GRANT ANY OBJECT PRIVILEGE システム権限が付与されているためこの権限付与を行える場合、この権限付与を行うと、オブジェクト所有者の役割を果たすことになります。*_TAB_PRIVS データ・ディクショナリ・ビューには、この権限付与が *object* の所有者によって行われたことが反映されます。

> **参照：**
>
> - 「ロールに対してオブジェクト権限を付与する例：」（18-50 ページ）
> - GRANT ANY OBJECT PRIVILEGE システム権限による権限の取消し操作の詳細は、18-91 ページの「GRANT ANY OBJECT PRIVILEGE を使用する操作を取り消す例：」を参照してください。

With Grant Option

With Grant Option を指定すると、権限受領者による、他のユーザーまたはロールに対するオブジェクト権限の付与を許可できます。

With Grant Option の付与の制限事項： With Grant Option は、ユーザーまたは PUBLIC に権限を付与する場合にのみ指定できます。ロールに付与する場合は指定できません。

WITH HIERARCHY OPTION

With Hierarchy Option を指定すると、この文の後に作成されるサブオブジェクトも含め、*object* のすべてのサブオブジェクト（ビューを基に作成したサブビューなど）に対する指定したオブジェクト権限を付与できます。

この句は、SELECT オブジェクト権限とあわせて指定する場合のみ意味があります。

object 権限を付与するスキーマ・オブジェクトを指定します。*object* を *schema* で修飾しなかった場合、そのオブジェクトは自分のスキーマ内にあるとみなされます。オブジェクトには次のタイプがあります。

- 表、ビューまたはマテリアライズド・ビュー
- 順序
- プロシージャ、ファンクションまたはパッケージ
- ユーザー定義型
- これらの項目のシノニム
- ディレクトリ、ライブラリ、演算子または索引タイプ
- Java ソース、クラスまたはリソース

パーティション表の単一パーティションに直接権限を付与することはできません。

> **参照：** 18-51 ページの「ユーザーに対して表へのオブジェクト権限を付与する例 :」、18-51 ページの「ビューへのオブジェクト権限を付与する例 :」および 18-51 ページの「別のスキーマの順序に対してオブジェクト権限を付与する例 :」を参照してください。

DIRECTORY *directory_name* 権限を付与するディレクトリ・スキーマ・オブジェクトを指定します。*directory_name* はスキーマ名で修飾できません。

> **参照：** 14-38 ページの「CREATE DIRECTORY」および 18-51 ページの「ディレクトリへのオブジェクト権限を付与する例 :」を参照してください。

JAVA SOURCE | RESOURCE JAVA 句を使用すると、権限を付与する Java ソースまたはリソース・スキーマ・オブジェクトを指定できます。

> **参照：** 「CREATE JAVA」（14-76 ページ）

システム権限およびオブジェクト権限

> **注意：** オブジェクトに対して ANY 権限（CREATE ANY CLUSTER など）を付与する場合、結果は、O7_DICTIONARY_ACCESSIBILITY 初期化パラメータの値によって決まります。このパラメータはデフォルトでは FALSE に設定されているため、ANY 権限の受領者はすべてのスキーマ内の該当するタイプのオブジェクトにアクセスできますが、SYS スキーマにはアクセスできません。O7_DICTIONARY_ACCESSIBILITY を TRUE に設定した場合、ANY 権限の受領者は、SYS スキーマ内の Oracle スケジューラ・オブジェクト以外のすべてのオブジェクトにアクセスできます。セキュリティ上の理由のため、この設定の使用には注意が必要です。

表 18-1 システム権限

システム権限名	許可される操作
アドバイザ・フレームワーク権限（すべてのアドバイザ・フレームワーク権限は、DBA ロールに含まれています。）	—
ADVISOR	PL/SQL パッケージ（DBMS_ADVISOR や DBMS_SQLTUNE など）を介したアドバイザ・フレームワークへのアクセス。 これらのパッケージの詳細は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。
ADMINISTER SQL TUNING SET	DBMS_SQLTUNE パッケージを介した、権限受領者が所有する SQL チューニング・セットの作成、削除、選択（読取り）およびロード（書込み）
ADMINISTER ANY SQL TUNING SET	DBMS_SQLTUNE パッケージを介した、任意のユーザーが所有する SQL チューニング・セットの作成、削除、選択（読取り）およびロード（書込み）
CREATE ANY SQL PROFILE	Enterprise Manager または DBMS_SQLTUNE パッケージを介してアクセスする、SQL チューニング・アドバイザが推奨した SQL プロファイルの受入れ
DROP ANY SQL PROFILE	既存の SQL プロファイルの削除
ALTER ANY SQL PROFILE	既存の SQL プロファイルの属性の変更

表 18-1 システム権限（続き）

システム権限名	許可される操作
クラスタ	—
CREATE CLUSTER	権限を付与したスキーマ内でのクラスタの作成
CREATE ANY CLUSTER	任意のスキーマ内でのクラスタの作成。CREATE ANY TABLE と同様に動作します。
ALTER ANY CLUSTER	任意のスキーマ内でのクラスタの変更
DROP ANY CLUSTER	任意のスキーマ内でのクラスタの削除
コンテキスト	—
CREATE ANY CONTEXT	任意のコンテキスト・ネームスペースの作成
DROP ANY CONTEXT	任意のコンテキスト・ネームスペースの削除
コンテキスト	—
ALTER DATABASE	データベースの変更
ALTER SYSTEM	ALTER SYSTEM 文の発行
AUDIT SYSTEM	AUDIT 文の発行
データベース・リンク	—
CREATE DATABASE LINK	権限を付与したスキーマ内でのプライベート・データベース・リンクの作成
CREATE PUBLIC DATABASE LINK	パブリック・データベース・リンクの作成
DROP PUBLIC DATABASE LINK	パブリック・データベース・リンクの削除
デバッグ	—
DEBUG CONNECT SESSION	デバッガへの現行のセッションの接続
DEBUG ANY PROCEDURE	任意のデータベース・オブジェクトのすべての PL/SQL コードおよび Java コードのデバッグ、およびアプリケーションによって実行されたすべての SQL 文に関する情報の表示。 **注意 :** この権限を付与することは、データベース内の適用可能なすべてのオブジェクトについて DEBUG オブジェクト権限を付与することと同じです。
ディクショナリ	—
ANALYZE ANY DICTIONARY	データ・ディクショナリ・オブジェクトの分析
ディメンション	—
CREATE DIMENSION	権限を付与したスキーマ内でのディメンションの作成
CREATE ANY DIMENSION	任意のスキーマ内でのディメンションの作成
ALTER ANY DIMENSION	任意のスキーマ内でのディメンションの変更
DROP ANY DIMENSION	任意のスキーマ内でのディメンションの削除
ディレクトリ	—
CREATE ANY DIRECTORY	ディレクトリ・データベース・オブジェクトの作成
DROP ANY DIRECTORY	ディレクトリ・データベース・オブジェクトの削除
フラッシュバック・データ・アーカイブ	—
FLASHBACK ARCHIVE ADMINISTER	フラッシュバック・データ・アーカイブの作成、変更または削除

表 18-1 システム権限（続き）

システム権限名	許可される操作
索引タイプ	—
CREATE INDEXTYPE	権限を付与したスキーマ内での索引タイプの作成
CREATE ANY INDEXTYPE	任意のスキーマ内での索引タイプの作成、および任意のスキーマ内の索引タイプに対するコメントの作成
ALTER ANY INDEXTYPE	任意のスキーマ内での索引タイプの変更
DROP ANY INDEXTYPE	任意のスキーマ内での索引タイプの削除
EXECUTE ANY INDEXTYPE	任意のスキーマ内での索引タイプの参照
索引	—
CREATE ANY INDEX	任意のスキーマでの任意の表に対するドメイン索引または索引の作成
ALTER ANY INDEX	任意のスキーマでの索引の変更
DROP ANY INDEX	任意のスキーマでの索引の削除
ジョブ・スケジューラ・オブジェクト	次の権限は、DBMS_SCHEDULER パッケージのプロシージャを実行する場合に必要です。この権限は、データベース・オブジェクトではない軽量ジョブには適用されません。軽量ジョブの詳細は、『Oracle Database 管理者ガイド』を参照してください。
CREATE JOB	権限を付与したスキーマ内でのジョブ、スケジュールまたはプログラムの作成
CREATE ANY JOB	任意のスキーマ内でのジョブ、スケジュールまたはプログラムの作成、変更または削除 **注意 :** この強力な権限を使用すると、権限受領者は任意のユーザーとしてコードを実行できます。この権限の付与には注意が必要です。
CREATE EXTERNAL JOB	権限を付与したスキーマ内での、オペレーティング・システム上で実行可能なスケジューラ・ジョブの作成
EXECUTE ANY PROGRAM	権限を付与したスキーマ内でのジョブの任意のプログラムの使用
EXECUTE ANY CLASS	権限を付与したスキーマ内でのジョブのジョブ・クラスの指定
MANAGE SCHEDULER	任意のジョブ・クラス、ウィンドウまたはウィンドウ・グループの作成、変更または削除
ライブラリ	—
CREATE LIBRARY	権限を付与したスキーマ内での外部プロシージャまたはファンクション・ライブラリの作成
CREATE ANY LIBRARY	任意のスキーマ内での外部プロシージャまたはファンクション・ライブラリの作成
DROP ANY LIBRARY	任意のスキーマ内での外部プロシージャまたはファンクション・ライブラリの削除
マテリアライズド・ビュー	—
CREATE MATERIALIZED VIEW	権限を付与したスキーマ内でのマテリアライズド・ビューの作成
CREATE ANY MATERIALIZED VIEW	任意のスキーマでのマテリアライズド・ビューの作成
ALTER ANY MATERIALIZED VIEW	任意のスキーマでのマテリアライズド・ビューの変更
DROP ANY MATERIALIZED VIEW	任意のスキーマでのマテリアライズド・ビューの削除
QUERY REWRITE	この権限は現在非推奨になっています。ユーザー自身のスキーマ内の表またはビューを参照するマテリアライズド・ビューのリライトを可能にする場合、権限は不要です。

表 18-1 システム権限（続き）

システム権限名	許可される操作
GLOBAL QUERY REWRITE	マテリアライズド・ビューが任意のスキーマ内の表またはビューを参照している場合のそのマテリアライズド・ビューの使用
ON COMMIT REFRESH	データベースの任意の表に対する REFRESH ON COMMIT モードのマテリアライズド・ビューの作成 データベースの任意の表に対する REFRESH ON DEMAND モードのマテリアライズド・ビューの、REFRESH ON COMMIT モードのマテリアライズド・ビューへの変更
FLASHBACK ANY TABLE	任意のスキーマ内の任意の表、ビューまたはマテリアライズド・ビューでの SQL フラッシュバック問合せの発行。この権限は、DBMS_FLASHBACK プロシージャの実行には不要です。
マイニング・モデル	—
CREATE MINING MODEL	権限を付与したスキーマでの DBMS_DATA_MINING.CREATE_MODEL プロシージャによるマイニング・モデルの作成
CREATE ANY MINING MODEL	任意のスキーマでの DBMS_DATA_MINING.CREATE_MODEL プロシージャによるマイニング・モデルの作成
ALTER ANY MINING MODEL	任意のスキーマでの適切な DBMS_DATA_MINING プロシージャを使用した、マイニング・モデル名またはモデルの関連付けられたコスト・マトリックスの変更
DROP ANY MINING MODEL	任意のスキーマでの DBMS_DATA_MINING.DROP_MODEL プロシージャによる任意のマイニング・モデルの削除
SELECT ANY MINING MODEL	任意のスキーマでの任意のモデルのスコアリングまたは表示。スコアリングの実行には、SQL ファンクションの PREDICTION ファミリ、または DBMS_DATA_MINING.APPLY プロシージャを使用します。モデルの表示は、DBMS_DATA_MINING.GET_MODEL_DETAILS_* プロシージャで実行されます。
COMMENT ANY MINING MODEL	任意のスキーマの任意のモデルでの SQL COMMENT 文によるコメントの作成
OLAP キューブ	Oracle Database を OLAP オプションで使用している場合は、次の権限が有効です。
CREATE CUBE	権限を付与したスキーマでの OLAP キューブの作成
CREATE ANY CUBE	任意のスキーマでの OLAP キューブの作成
ALTER ANY CUBE	任意のスキーマでの OLAP キューブの変更
DROP ANY CUBE	任意のスキーマでの任意の OLAP キューブの削除
SELECT ANY CUBE	任意のスキーマでの任意の OLAP キューブの問合せまたは表示
UPDATE ANY CUBE	任意のスキーマでの任意のキューブの更新
OLAP キューブ・メジャー・フォルダ	Oracle Database を OLAP オプションで使用している場合は、次の権限が有効です。
CREATE MEASURE FOLDER	権限を付与したスキーマでの OLAP メジャー・フォルダの作成
CREATE ANY MEASURE FOLDER	任意のスキーマでの OLAP メジャー・フォルダの作成
DELETE ANY MEASURE FOLDER	任意のスキーマでの任意の OLAP メジャー・フォルダからの削除
DROP ANY MEASURE FOLDER	任意のスキーマでの任意のメジャー・フォルダの削除
INSERT ANY MEASURE FOLDER	任意のスキーマでの任意のメジャー・フォルダへのメジャーの挿入

表 18-1 システム権限（続き）

システム権限名	許可される操作
OLAP キューブ・ディメンション	Oracle Database を OLAP オプションで使用している場合は、次の権限が有効です。
CREATE CUBE DIMENSION	権限を付与したスキーマでの OLAP キューブ・ディメンションの作成
CREATE ANY CUBE DIMENSION	任意のスキーマでの OLAP キューブ・ディメンションの作成
ALTER ANY CUBE DIMENSION	任意のスキーマでの OLAP キューブ・ディメンションの変更
DELETE ANY CUBE DIMENSION	任意のスキーマでの OLAP キューブ・ディメンションからの削除
DROP ANY CUBE DIMENSION	任意のスキーマでの OLAP キューブ・ディメンションの削除
INSERT ANY CUBE DIMENSION	任意のスキーマでの OLAP キューブ・ディメンションへの挿入
SELECT ANY CUBE DIMENSION	任意のスキーマでの OLAP キューブ・ディメンションの表示または問合せ
UPDATE ANY CUBE DIMENSION	任意のスキーマでの OLAP キューブ・ディメンションの更新
OLAP キューブの作成プロセス	—
CREATE CUBE BUILD PROCESS	権限を付与したスキーマでの OLAP キューブの作成プロセスの作成
CREATE ANY CUBE BUILD PROCESS	任意のスキーマでの OLAP キューブの作成プロセスの作成
DROP ANY CUBE BUILD PROCESS	任意のスキーマでの OLAP キューブの作成プロセスの削除
UPDATE ANY CUBE BUILD PROCESS	任意のスキーマでの OLAP キューブの作成プロセスの更新
演算子	—
CREATE OPERATOR	権限を付与したスキーマ内での演算子およびバインディングの作成
CREATE ANY OPERATOR	任意のスキーマ内での演算子とそのバインディングの作成、および任意のスキーマ内の演算子に対するコメントの作成
ALTER ANY OPERATOR	任意のスキーマ内での演算子の変更
DROP ANY OPERATOR	任意のスキーマ内での演算子の削除
EXECUTE ANY OPERATOR	任意のスキーマ内での演算子の参照
アウトライン	—
CREATE ANY OUTLINE	アウトラインを使用する任意のスキーマ内で使用するパブリック・アウトラインの作成
ALTER ANY OUTLINE	アウトラインの変更
DROP ANY OUTLINE	アウトラインの削除
プロシージャ	—
CREATE PROCEDURE	権限受領者のスキーマ内でのストアド・プロシージャ、ストアド・ファンクションおよびストアド・パッケージの作成
CREATE ANY PROCEDURE	任意のスキーマ内でのストアド・プロシージャ、ストアド・ファンクションおよびストアド・パッケージの作成
ALTER ANY PROCEDURE	任意のスキーマ内でのストアド・プロシージャ、ストアド・ファンクションおよびストアド・パッケージの変更
DROP ANY PROCEDURE	任意のスキーマ内でのストアド・プロシージャ、ストアド・ファンクションおよびストアド・パッケージの削除

表 18-1 システム権限（続き）

システム権限名	許可される操作
EXECUTE ANY PROCEDURE	プロシージャまたはファンクションの実行（スタンドアロンまたはパッケージ） 任意のスキーマ内でのパブリック・パッケージ変数の参照
プロファイル	—
CREATE PROFILE	プロファイルの作成
ALTER PROFILE	プロファイルの変更
DROP PROFILE	プロファイルの削除
ロール	—
CREATE ROLE	ロールの作成
ALTER ANY ROLE	データベース内の任意のロールの変更
DROP ANY ROLE	ロールの削除
GRANT ANY ROLE	データベース内の任意のロールの付与
ロールバック・セグメント	—
CREATE ROLLBACK SEGMENT	ロールバック・セグメントの作成
ALTER ROLLBACK SEGMENT	ロールバック・セグメントの変更
DROP ROLLBACK SEGMENT	ロールバック・セグメントの削除
順序	—
CREATE SEQUENCE	権限を付与したスキーマ内での順序の作成
CREATE ANY SEQUENCE	任意のスキーマ内での順序の作成
ALTER ANY SEQUENCE	データベース内の任意の順序の変更
DROP ANY SEQUENCE	任意のスキーマ内での順序の削除
SELECT ANY SEQUENCE	任意のスキーマ内での順序の参照
セッション	—
CREATE SESSION	データベースへの接続
ALTER RESOURCE COST	セッション・リソースに対するコストの設定
ALTER SESSION	SQL トレース機能の有効と無効の切替え
RESTRICTED SESSION	SQL*Plus の STARTUP RESTRICT 文を使用した、インスタンス起動後のログイン
スナップショット	マテリアライズド・ビューを参照
シノニム	—
CREATE SYNONYM	権限を付与したスキーマ内でのシノニムの作成
CREATE ANY SYNONYM	任意のスキーマ内でのプライベート・シノニムの作成
CREATE PUBLIC SYNONYM	パブリック・シノニムの作成
DROP ANY SYNONYM	任意のスキーマ内でのプライベート・シノニムの削除
DROP PUBLIC SYNONYM	パブリック・シノニムの削除

表 18-1 システム権限（続き）

システム権限名	許可される操作
表	**注意 :** 外部表の場合、有効なシステム権限は、CREATE ANY TABLE、ALTER ANY TABLE、DROP ANY TABLE および SELECT ANY TABLE です。
CREATE TABLE	権限を付与したスキーマ内での表の作成
CREATE ANY TABLE	任意のスキーマ内での表の作成。なお、表が設定されるスキーマの所有者は、表領域内にその表を定義するための割当て制限が必要です。
ALTER ANY TABLE	スキーマ内の任意の表またはビューの変更
BACKUP ANY TABLE	エクスポート・ユーティリティを使用した他のユーザーのスキーマからのオブジェクトの増分エクスポート
DELETE ANY TABLE	任意のスキーマ内での表、表パーティションまたはビューからの行の削除
DROP ANY TABLE	任意のスキーマ内での表または表パーティションの削除または切捨て
INSERT ANY TABLE	任意のスキーマ内の表またはビューへの行の挿入
LOCK ANY TABLE	任意のスキーマ内の表またはビューのロック
SELECT ANY TABLE	任意のスキーマ内の表、ビューまたはマテリアライズド・ビューの問合せ
FLASHBACK ANY TABLE	任意のスキーマ内の任意の表、ビューまたはマテリアライズド・ビューでの SQL フラッシュバック問合せの発行。この権限は、DBMS_FLASHBACK プロシージャの実行には不要です。
UPDATE ANY TABLE	任意のスキーマ内の表またはビューの行の更新
表領域	—
CREATE TABLESPACE	表領域の作成
ALTER TABLESPACE	表領域の変更
DROP TABLESPACE	表領域の削除
MANAGE TABLESPACE	表領域のオンラインとオフラインの切替え、および表領域のバックアップの開始と終了の制御
UNLIMITED TABLESPACE	任意の表領域の無制限な使用。この権限は、設定されている任意の割当て制限を上書きします。ユーザーからこの権限を取り消した場合、ユーザーのスキーマ・オブジェクトはそのまま残りますが、表領域の割当て制限が許可されないかぎり、それ以上表領域を割り当てることはできません。このシステム権限をロールに付与することはできません。
トリガー	—
CREATE TRIGGER	権限を付与したスキーマ内でのデータベース・トリガーの作成
CREATE ANY TRIGGER	任意のスキーマ内でのデータベース・トリガーの作成
ALTER ANY TRIGGER	任意のスキーマ内でのデータベース・トリガーの使用可能化、使用禁止化またはコンパイル
DROP ANY TRIGGER	任意のスキーマ内でのデータベース・トリガーの削除
ADMINISTER DATABASE TRIGGER	データベース内でのトリガーの作成。CREATE TRIGGER または CREATE ANY TRIGGER システム権限も必要です。

表 18-1 システム権限（続き）

システム権限名	許可される操作
型	—
CREATE TYPE	権限を付与したスキーマ内でのオブジェクト型およびオブジェクト型本体の作成
CREATE ANY TYPE	任意のスキーマ内でのオブジェクト型およびオブジェクト型本体の作成
ALTER ANY TYPE	任意のスキーマ内でのオブジェクト型の変更
DROP ANY TYPE	任意のスキーマ内でのオブジェクト型およびオブジェクト型本体の削除
EXECUTE ANY TYPE	特定のユーザーに付与した場合、任意のスキーマ内でのオブジェクト型およびコレクション型を使用および参照した、任意のスキーマ内のオブジェクト型メソッドの起動。EXECUTE ANY TYPE をロールに付与した場合、使用可能なロールを保持したユーザーは、任意のスキーマ内のオブジェクト型メソッドを起動できません。
UNDER ANY TYPE	非最終オブジェクト型のサブタイプの作成
ユーザー	—
CREATE USER	ユーザーの作成。この権限によって、次の操作を実行できます。 ■ 任意の表領域に対する割当て制限の設定 ■ デフォルトの表領域および一時表領域の設定 ■ CREATE USER 文の一部としてのプロファイルの設定
ALTER USER	任意のユーザーの変更。この権限によって、次の操作を実行できます。 ■ 他のユーザーのパスワードまたは認証方法の変更 ■ 任意の表領域に対する割当て制限の設定 ■ デフォルトの表領域および一時表領域の設定 ■ プロファイルおよびデフォルト・ロールの設定
DROP USER	ユーザーの削除
ビュー	—
CREATE VIEW	権限を付与したスキーマ内でのビューの作成
CREATE ANY VIEW	任意のスキーマ内でのビューの作成
DROP ANY VIEW	任意のスキーマ内でのビューの削除
UNDER ANY VIEW	オブジェクト・ビューのサブビューの作成
FLASHBACK ANY TABLE	任意のスキーマ内の任意の表、ビューまたはマテリアライズド・ビューでの SQL フラッシュバック問合せの発行。この権限は、DBMS_FLASHBACK プロシージャの実行には不要です。
MERGE ANY VIEW	MERGE ANY VIEW 権限が付与されている場合、そのユーザーが発行するすべての問合せにおいて、オプティマイザはビューのマージを使用して問合せのパフォーマンスを向上することができます。このとき、ビューのマージがビュー作成者のセキュリティ意図に違反しないかどうかは確認されません。OPTIMIZER_SECURE_VIEW_MERGING パラメータの詳細は『Oracle Database リファレンス』を、ビューのマージの詳細は『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。

表 18-1 システム権限（続き）

システム権限名	許可される操作
その他	—
ANALYZE ANY	任意のスキーマ内の任意の表、クラスタまたは索引の分析
AUDIT ANY	AUDIT *schema_objects* 文を使用した、任意のスキーマ内の任意のオブジェクトの監査
BECOME USER	データ・ポンプ・インポート・ユーティリティ（impdp）および元のインポート・ユーティリティ（imp）のユーザーは、第三者が直接実行できない操作（オブジェクト権限を付与するようなオブジェクトのロード）を実行する場合には、別のユーザー・アイデンティティを引き受けることができます。 Streams 管理者は、取得ユーザーおよび適用ユーザーの作成または変更を 1 つの Streams 環境で行うことができます。この権限は、デフォルトでは DBA ロールに含まれています。Database Vault では、この権限は DBA ロールから除外されます。したがって、Streams でこの権限が必要とされるのは、Database Vault がインストールされている環境でのみです。
CHANGE NOTIFICATION	問合せの登録の作成と、登録された問合せに関連付けられたオブジェクトに対する DML または DDL 変更があったときのデータベース変更通知の受信。データベース変更通知の詳細は、『Oracle Database アドバンスト・アプリケーション開発者ガイド』を参照してください。
COMMENT ANY TABLE	任意のスキーマ内の任意の表、ビューまたは列についてのコメントの記述
EXEMPT ACCESS POLICY	ファイングレイン・アクセス・コントロールの回避 **注意 :** 強力なシステム権限で、権限受領者がアプリケーション駆動のセキュリティ・ポリシーを回避できます。データベース管理者がこの権限を付与する場合は、注意が必要です。
FORCE ANY TRANSACTION	ローカル・データベース内の、インダウト分散トランザクションのコミットまたはロールバック 分散トランザクション・エラーの意図的な発生
FORCE TRANSACTION	ローカル・データベース内の、インダウト分散トランザクションのコミットまたはロールバック
GRANT ANY OBJECT PRIVILEGE	オブジェクト所有者が付与を許されている任意のオブジェクト権限の付与 GRANT ANY OBJECT PRIVILEGE 権限を持つオブジェクト所有者または他のユーザーによって付与されたオブジェクト権限の取消し
GRANT ANY PRIVILEGE	任意のシステム権限の付与
RESUMABLE	再開可能な領域割当ての使用可能化
SELECT ANY DICTIONARY	SYS スキーマ内のデータ・ディクショナリ・オブジェクトへの問合せ。初期化パラメータ O7_DICTIONARY_ACCESSIBILITY のデフォルトの FALSE 設定を選択的に上書きします。
SELECT ANY TRANSACTION	FLASHBACK_TRANSACTION_QUERY ビューの内容の問合せ **注意 :** 強力なシステム権限で、権限受領者が、過去のデータも含めてデータベース内のすべてのデータを参照できます。この権限は、Oracle フラッシュバック・トランザクション問合せ機能を使用する必要があるユーザーのみに付与してください。

表 18-1 システム権限（続き）

システム権限名	許可される操作
SYSDBA	STARTUP および SHUTDOWN 操作の実行 ALTER DATABASE（オープン、マウント、バックアップまたはキャラクタ・セットの変更） CREATE DATABASE ARCHIVELOG および RECOVERY CREATE SPFILE RESTRICTED SESSION 権限を含みます。
SYSOPER	STARTUP および SHUTDOWN 操作の実行 ALTER DATABASE（オープン、マウントまたはバックアップ） ARCHIVELOG および RECOVERY CREATE SPFILE RESTRICTED SESSION 権限を含みます。
CONNECT、RESOURCE および DBA	以前のリリースとの互換性を確保するためのロールです。DBA_SYS_PRIVS データ・ディクショナリ・ビューを問い合せることによって、これらのロールにまとめられた権限を確認できます。 **注意 :** データベースのセキュリティを維持するために、独自のロールを設計することをお薦めします。これらのロールは、今後の Oracle Database のリリースでは自動的に作成されない可能性があります。 **参照 :** DBA_SYS_PRIVS ビューの詳細は、『Oracle Database リファレンス』を参照してください。
DELETE_CATALOG_ROLE EXECUTE_CATALOG_ROLE SELECT_CATALOG_ROLE	データ・ディクショナリ・ビューおよびパッケージへアクセスするためのロールです。 **参照 :** これらのロールの詳細は、『Oracle Database 管理者ガイド』を参照してください。
EXP_FULL_DATABASE IMP_FULL_DATABASE	インポート / エクスポート・ユーティリティを簡単に使用するためのロールです。 **参照 :** これらのロールの詳細は、『Oracle Database ユーティリティ』を参照してください。
AQ_USER_ROLE AQ_ADMINISTRATOR_ROLE	Oracle Advanced Queuing を使用する場合、これらのロールが必要です。 **参照 :** これらのロールの詳細は、『Oracle Streams アドバンスト・キューイング・ユーザーズ・ガイド』を参照してください。
SNMPAGENT	Oracle Enterprise Manager Management Agent で使用されるロールです。 **参照 :** これらのロールの詳細は、『Oracle Enterprise Manager Advanced Configuration』を参照してください。
RECOVERY_CATALOG_OWNER	リカバリ・カタログを所有するユーザーを作成する場合、このロールが必要です。 **参照 :** リカバリ・カタログの詳細は、『Oracle Database バックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

表 18-2 オブジェクト権限とその権限によって許可される操作

オブジェクト権限	許可される操作
ディレクトリ権限	次の**ディレクトリ権限**では、ディレクトリ・オブジェクトをポインタとして使用することにより、オペレーティング・システムのディレクトリに格納されている各ファイルにデータベースから安全にアクセスできるようになります。このディレクトリ・オブジェクトには、ファイルが格納されているオペレーティング・システムのディレクトリへのフルパス名が定義されています。これらのファイルは実際にはデータベース外に格納されているため、Oracle Database サーバーの各プロセスは、ファイル・システム・サーバーに対して適切なファイル・アクセス権も持っている必要があります。オペレーティング・システムに対するオブジェクト権限ではなく、ディレクトリ・データベース・オブジェクトに対するオブジェクト権限を個々のデータベース・ユーザーに付与することによって、データベースでファイル運用時のセキュリティが確保されます。
READ	ディレクトリ内のファイルの読取り。
WRITE	ディレクトリ内へのファイルの書込み。外部表に接続する場合のみに役立ちます。これによって、権限受領者は、外部表のエージェントがディレクトリに書き込めるのがログ・ファイルなのか不良ファイルなのかを判断できます。 **制限事項 :** この権限を持っていても、BFILE への書込みを行うことはできません。
索引タイプ権限	次の**索引タイプ権限**は、索引タイプの操作を許可します。
EXECUTE（注意 1）	索引タイプの参照。
フラッシュバック・データ・アーカイブ権限	次の**フラッシュバック・データ・アーカイブ権限**では、フラッシュバック・データ・アーカイブに対する操作を許可します。
FLASHBACK ARCHIVE	表の履歴追跡を使用可能または使用禁止にします。
ライブラリ権限	次の**ライブラリ権限**は、ライブラリの操作を許可します。
EXECUTE（注意 1）	特定のオブジェクトの使用と参照、およびそのメソッドの起動。
マテリアライズド・ビュー権限	次の**マテリアライズド・ビュー権限**は、マテリアライズド・ビューについての操作を許可します。DELETE、INSERT および UPDATE 権限は、更新可能なマテリアライズド・ビューにのみ付与できます。
ON COMMIT REFRESH	指定した表に対する REFRESH ON COMMIT モードのマテリアライズド・ビューの作成。
QUERY REWRITE	指定した表で使用するクエリー・リライトに対するマテリアライズド・ビューの作成。
SELECT	SELECT 文でのマテリアライズド・ビューの問合せ。
マイニング・モデル権限	次のマイニング・モデル権限では、マイニング・モデルに対する操作を許可します。これらの権限は、ユーザー自身のスキーマ内にあるモデルに対しては必要ありません。
ALTER	該当する DBMS_DATA_MINING プロシージャを使用して、マイニング・モデル名または関連のコスト・マトリックスを変更します。
SELECT	マイニング・モデルのスコアリングまたは表示。スコアリングの実行には、SQL ファンクションの PREDICTION ファミリ、または DBMS_DATA_MINING.APPLY プロシージャを使用します。モデルの表示は、DBMS_DATA_MINING.GET_MODEL_DETAILS_* プロシージャで実行されます。
オブジェクト型権限	次の**オブジェクト型権限**は、データベース・オブジェクト型の操作を許可します。
DEBUG	オブジェクト型に定義されたすべてのパブリック変数、非パブリック変数、メソッドおよび型へのデバッガを介したアクセス。 型本体内の行または指示境界へのブレークポイントの設定、またはこれらの場所のいずれかでの停止。
EXECUTE（注意 1）	特定のオブジェクトの使用と参照、およびそのメソッドの起動。 オブジェクト型に定義されたパブリック変数、型およびメソッドへのデバッガを介したアクセス。

表 18-2 オブジェクト権限とその権限によって許可される操作（続き）

オブジェクト権限	許可される操作
UNDER	型のサブタイプの作成。この型の直属のスーパータイプに With Grant Option 付きの UNDER ANY TYPE 権限を持つ場合のみ、このオブジェクト権限を付与できます。
OLAP 権限	Oracle Database を OLAP オプションで使用している場合は、次のオブジェクト権限が有効になります。
INSERT	メンバーを OLAP キューブ・ディメンションに挿入するか、またはメジャーをメジャー・フォルダに挿入します。
ALTER	OLAP キューブ・ディメンションまたはキューブの定義を変更します。
DELETE	メンバーを OLAP キューブ・ディメンションから削除するか、またはメジャーをメジャー・フォルダから削除します。
SELECT	OLAP キューブまたはキューブ・ディメンションを表示または問い合せます。
UPDATE	OLAP キューブのメジャー値またはキューブ・ディメンションの属性値を更新します。
演算子権限	次の**演算子権限**は、ユーザー定義演算子の操作を許可します。
EXECUTE（注意 1）	演算子の参照。
プロシージャ、ファンクション、パッケージ権限	次の**プロシージャ、ファンクションおよびパッケージ権限**は、プロシージャ、ファンクションおよびパッケージの操作を許可します。これらの権限は、**Java ソース、クラスおよびリソース**にも適用されます。Oracle Database では、これらはオブジェクト権限の付与のために生成されたプロシージャとして扱われます。
DEBUG	オブジェクトに定義されたすべてのパブリック変数、非パブリック変数、メソッドおよび型へのデバッガを介したアクセス。 プロシージャ、ファンクションまたはパッケージ内の行または指示境界へのブレークポイントの設定、またはこれらの場所のいずれかでの停止。この権限は、メソッドまたはパッケージの仕様部および本体の宣言にアクセスする権限を付与します。
EXECUTE（注意 1）	プロシージャかファンクションの直接実行、パッケージの仕様部に宣言された任意のプログラム・オブジェクトへのアクセス、または現在無効であるかコンパイルされていないファンクションかプロシージャへのコール中の暗黙的なオブジェクトのコンパイル。この権限を持っていても、ALTER PROCEDURE または ALTER FUNCTION を使用して明示的なコンパイルを実行することはできません。明示的なコンパイルを実行するには、適切な ALTER SYSTEM 権限が必要です。 プロシージャ、ファンクションまたはパッケージに定義されたパブリック変数、型およびメソッドへのデバッガを介したアクセス。この権限は、メソッドまたはパッケージの仕様部のみの宣言にアクセスする権限を付与します。 ジョブ・スケジューラ・オブジェクトは、DBMS_SCHEDULER パッケージを使用して作成します。作成したオブジェクトに、ジョブ・スケジューラのクラスおよびプログラムに対する EXECUTE オブジェクト権限を付与できます。ジョブ・スケジューラのジョブ、プログラムおよびスケジュールに対しても ALTER 権限を付与できます。 **注意 :** プロシージャ、ファンクションまたはパッケージを間接的に実行する場合、ユーザーはこの権限を持つ必要はありません。
SCHEDULER PRIVILEGES	ジョブ・スケジューラ・オブジェクトは、DBMS_SCHEDULER パッケージを使用して作成します。これらのオブジェクトを作成した後で、次の権限を付与できます。
EXECUTE	ジョブ・クラス、プログラム、チェーンおよび資格証明に対する操作。
ALTER	ジョブ、プログラム、チェーン、資格証明およびスケジュールに対する変更。

表 18-2 オブジェクト権限とその権限によって許可される操作（続き）

オブジェクト権限	許可される操作
順序権限	次の**順序権限**は、順序の操作を許可します。
ALTER	ALTER SEQUENCE 文での順序定義の変更。
SELECT	CURRVAL 疑似列および NEXTVAL 疑似列を使用した順序の値の検査および増分。
シノニム権限	**シノニム権限**は、対象となるオブジェクトに対して付与される権限と同じです。シノニムに権限を付与することは、基本オブジェクトに権限を付与することと同じです。同様に、基本オブジェクトに対して権限を付与することは、オブジェクトのすべてのシノニムに権限を付与することと同じです。あるユーザーにシノニムの権限を付与した場合、そのユーザーは、シノニム名または基本オブジェクト名を SQL 文に指定して、その権限を使用できます。
表権限	次の**表権限**は、表の操作を許可します。次のいずれかのオブジェクト権限を持っている場合は、LOCK TABLE 文を使用して任意のロック・モードで表をロックできます。 **注意：**外部表に有効な権限は、ALTER および SELECT のみです。
ALTER	ALTER TABLE 文での表定義の変更。
DELETE	DELETE 文での表の行の削除。 **注意：**表がリモート・データベースにある場合は、DELETE 権限とともに表に対する SELECT 権限を付与する必要があります。
DEBUG	デバッガを介した次のものへのアクセス。 ■ 表に定義されているトリガーの本体の PL/SQL コード ■ 表を直接参照する SQL 文に関する情報
INDEX	CREATE INDEX 文での表の索引の作成。
INSERT	INSERT 文での表への新しい行の追加。
REFERENCES	表参照制約の作成。この権限はロールには付与できません。
SELECT	SELECT 文での表の問合せ。
UPDATE	UPDATE 文での表のデータの変更。 **注意：**表がリモート・データベースにある場合は、UPDATE 権限とともに表に対する SELECT 権限を付与する必要があります。
ビュー権限	次の**ビュー権限**は、ビューの操作を許可します。次のいずれかのオブジェクト権限を持っている場合は、LOCK TABLE 文を使用して任意のロック・モードでビューをロックできます。 ビューの権限を付与する場合、そのビューのすべての実表に関して Grant Option 付きの権限が必要です。
DEBUG	デバッガを介した次のものへのアクセス。 ■ ビューに定義されているトリガーの本体の PL/SQL コード ■ ビューを直接参照する SQL 文に関する情報
DELETE	DELETE 文でのビューの行の削除。
INSERT	INSERT 文でのビューへの新しい行の追加。
MERGE	このオブジェクト権限の動作は、権限が ON 句で指定されたビューに制限される点を除き、18-44 ページの MERGE ANY VIEW システム権限の動作と同じです。指定されたビューに対して権限受領者が発行するすべての問合せに対して、オプティマイザはビューのマージを使用して問合せのパフォーマンスを向上することができます。このとき、ビューのマージがビュー作成者のセキュリティ意図に違反しないかどうかは確認されません。

表 18-2 オブジェクト権限とその権限によって許可される操作（続き）

オブジェクト権限	許可される操作
REFERENCES	ビューへの外部キー制約の定義。
SELECT	SELECT 文でのビューの問合せ。 **参照：**ビューに対するこのオブジェクト権限の付与の詳細は、18-35 ページの「object_privilege」を参照してください。
UNDER	ビューのサブビューの作成。このビューの直属のスーパービューに With Grant Option 付きの UNDER ANY VIEW 権限を持つ場合のみ、このオブジェクト権限を付与できます。
UPDATE	UPDATE 文でのビューのデータの変更。

例

ユーザーに対してシステム権限を付与する例： 次の文は、サンプル・ユーザー hr に CREATE SESSION システム権限を付与し、hr が Oracle Database にログインできるようにします。

```
GRANT CREATE SESSION
   TO hr;
```

ロールに対してシステム権限を付与する例： 次の文は、データ・ウェアハウス管理者ロール（15-58 ページの「ロールの作成例：」で作成）に、適切なシステム権限を付与します。

```
GRANT
     CREATE ANY MATERIALIZED VIEW
   , ALTER ANY MATERIALIZED VIEW
   , DROP ANY MATERIALIZED VIEW
   , QUERY REWRITE
   , GLOBAL QUERY REWRITE
   TO dw_manager
   WITH ADMIN OPTION;
```

dw_manager の権限ドメインには、マテリアライズド・ビューに関連するシステム権限が含まれます。

Admin Option 付きロールを付与する例： 次の文は、サンプル・ユーザー sh に Admin Option 付きの dw_manager ロールを付与します。

```
GRANT dw_manager
   TO sh
   WITH ADMIN OPTION;
```

dw_manager ロールによって、sh は次の操作を実行できます。

- ロールを使用可能にして、CREATE MATERIALIZED VIEW システム権限を含むそのロールの権限ドメインに登録されている権限の使用
- 他のユーザーに対するそのロールの付与および取消し
- ロールの削除

ロールに対してオブジェクト権限を付与する例： 次の文は、データ・ウェアハウス・ユーザー・ロール（15-58 ページの「ロールの作成例：」で作成）に、SELECT オブジェクト権限を付与します。

```
GRANT SELECT ON sh.sales TO warehouse_user;
```

ロールに対してロールを付与する例： 次の文は、dw_manager ロールに、warehouse_user ロールを付与します（いずれのロールも、15-58 ページの「ロールの作成例：」で作成）。

```
GRANT warehouse_user TO dw_manager;
```

dw_manager ロールには、warehouse_user ロールのドメインにあるすべての権限が含まれます。

ディレクトリへのオブジェクト権限を付与する例： 次の文は、ユーザー hr にディレクトリ bfile_dir に対する READ 権限を Grant Option 付きで付与します。

```
GRANT READ ON DIRECTORY bfile_dir TO hr
   WITH GRANT OPTION;
```

ユーザーに対して表へのオブジェクト権限を付与する例： 次の文は、ユーザー hr に対して、oe.bonuses 表（18-74 ページの「表へのマージ例：」で作成）についてのすべての権限を Grant Option 付きで付与します。

```
GRANT ALL ON bonuses TO hr
   WITH GRANT OPTION;
```

この結果、hr ユーザーは次の操作を実行できます。

- bonuses 表に対するすべての権限の使用
- 他のユーザーまたはロールに対する、bonuses 表についての権限の付与

ビューへのオブジェクト権限を付与する例： 次の文は、ビュー emp_view（17-20 ページの「ビューの作成例：」で作成）についての SELECT 権限および UPDATE 権限をすべてのユーザーに付与します。

```
GRANT SELECT, UPDATE
   ON emp_view TO PUBLIC;
```

この結果、すべてのユーザーが、従業員の詳細についてのビューを問合せおよび更新できるようになります。

別のスキーマの順序に対してオブジェクト権限を付与する例： 次の文は、ユーザー hr に対して、スキーマ oe 内の customers_seq 順序の SELECT 権限を付与します。

```
GRANT SELECT
   ON oe.customers_seq TO hr;
```

ユーザー hr は、次の文を指定して、順序の次の値を作成できるようになります。

```
SELECT oe.customers_seq.NEXTVAL
   FROM DUAL;
```

別々の列に複数のオブジェクト権限を付与する例： 次の文は、ユーザー oe に、スキーマ hr にある employees 表の employee_id 列に対する REFERENCES 権限、および employee_id、salary、commission_pct 列に対する UPDATE 権限を付与します。

```
GRANT REFERENCES (employee_id),
      UPDATE (employee_id, salary, commission_pct)
   ON hr.employees
   TO oe;
```

この結果、ユーザー oe は、employee_id 列、salary 列および commission_pct 列の値を更新できるようになります。ユーザー oe は、employee_id 列を参照する参照整合性制約を定義することもできます。ただし、GRANT 文にはこれらの列のみが指定されているため、ユーザー oe は employees 表の他の列を操作できません。

たとえば、oe は制約付きの表を作成できます。

```
CREATE TABLE dependent
   (dependno   NUMBER,
    dependname VARCHAR2(10),
    employee   NUMBER
   CONSTRAINT in_emp REFERENCES hr.employees(employee_id) );
```

スキーマ hr 内の employees 表の従業員に対応する dependent 表の依存性が、制約 in_emp によって保証されます。

INSERT

用途

INSERT 文を使用すると、表、ビューの実表、パーティション表のパーティション、コンポジット・パーティション表のサブパーティション、オブジェクト表またはオブジェクト・ビューの実表に、行を追加できます。

前提条件

表に行を挿入する場合は、その表が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、その表に対する INSERT 権限が必要です。

ビューの実表に行を挿入する場合、ビューが定義されているスキーマの所有者には、その実表に対する INSERT 権限が必要です。また、他のユーザーのスキーマ内のビューに行を挿入する場合は、そのビューに対する INSERT 権限が必要です。

INSERT ANY TABLE システム権限があれば、任意の表または任意のビューの実表に行を挿入できます。

従来型 INSERT およびダイレクト・パス・インサート

表、パーティションまたはビューにデータを挿入するために使用する INSERT 文には、従来型 INSERT およびダイレクト・パス・インサートの 2 種類があります。従来型 INSERT 文を発行すると、表の空き領域を再利用して挿入され、参照整合性制約が維持されます。ダイレクト・パス・インサートの場合、表の既存データの後に、挿入したデータが追加されます。データは、バッファ・キャッシュを回避してデータ・ファイルに直接書き込まれます。既存データの空き領域は再利用されません。これによって挿入操作中のパフォーマンスが向上します。また、これは Oracle のダイレクト・パスのローダー・ユーティリティ、SQL*Loader の機能に似ています。パラレル・モードで作成された表に挿入する場合は、ダイレクト・パス・インサートがデフォルトです。

データベースでの REDO データおよび UNDO データの生成方法は、従来型 INSERT とダイレクト・パス・インサートのいずれを使用しているかに一部関係しています。

- 従来型 INSERT では、表とアーカイブ・ログのロギング設定に関係なく、データとメタデータの両方に対する変更に対して常に最大の REDO および UNDO が生成され、データベースのロギング設定が強制的に使用されます。
- ダイレクト・パス・インサートでは、メタデータの変更に対して REDO と UNDO の両方が生成されます。これらは操作のリカバリに必要であるためです。データの変更に大しては UNDO と REDO が次のように生成されます。
  - ダイレクト・パス・インサートでは、データの変更に対する UNDO の生成が常に回避されます。
  - データベースが ARCHIVELOG または FORCE LOGGING モードでない場合は、表のロギング設定に関係なく、データの変更に対する REDO は生成されません。
  - データベースが ARCHIVELOG モード（ただし FORCE LOGGING モードではない）の場合、ダイレクト・パス・インサートでは、LOGGING 表のデータの REDO は生成されますが、NOLOGGING 表のデータの REDO は生成されません。
  - データベースが ARCHIVELOG モード**かつ** FORCE LOGGING モードの場合、ダイレクト・パス SQL では、LOGGING 表と NOLOGGING 表の両方のデータの REDO が生成されます。

ダイレクト・パス・インサートには、次の制限があります。これらの制限に違反する場合、他にエラーがないかぎりメッセージが戻されず、従来型 INSERT がシリアルで実行されます。

- DML 文の有無にかかわらず、1 つのトランザクションで複数のダイレクト・パス・インサート文を使用できます。ただし、ある DML 文が特定の表、パーティションまたは索引を変更した後は、トランザクションの他の DML 文は表、パーティションまたは索引にアクセスできません。

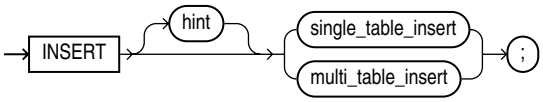
- 同じ表、パーティションまたは索引にアクセスする問合せは、ダイレクト・パス・インサート文の前であれば許可されますが、後は許可されません。
- シリアルまたはパラレル文が、同じトランザクションからダイレクト・パス・インサートによって変更された表にアクセスしようとする場合、エラーが戻され、文が拒否されます。
- 対象となる表は、クラスタの一部にはできません。
- 対象となる表は、オブジェクト型列を含むことはできません。
- ダイレクト・パス・インサートは、パーティション化されていない場合、マッピング表を含んでいる場合またはマテリアライズド・ビューによって参照されている場合は、索引構成表（IOT）をサポートしません。
- 索引構成表（IOT）の 1 つのパーティションまたは 1 つのパーティションのみ含むパーティション化された IOT へのダイレクト・パス・インサートは、IOT がパラレル・モードで作成されている場合または APPEND ヒントを指定した場合でも、シリアルで実行されます。ただし、パーティション化された IOT へのダイレクト・パス・インサート操作は、拡張パーティション名が使用されず、IOT に複数のパーティションが含まれているかぎり、パラレル・モードで実行されません。
- 対象となる表には、トリガーまたは参照整合性制約を定義できません。
- 対象となる表は、レプリケートできません。
- ダイレクト・パス・インサート文を含むトランザクションは、分散できません。

参照：

- ダイレクト・パス・インサートの詳細は、『Oracle Database 管理者ガイド』を参照してください。
- SQL*Loader の詳細は、『Oracle Database ユーティリティ』を参照してください。
- パラレル・ダイレクト・パス・インサートをチューニングする方法の詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。

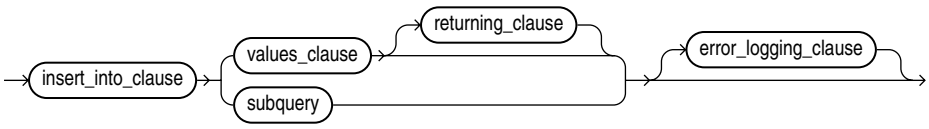
構文

***insert*::=**



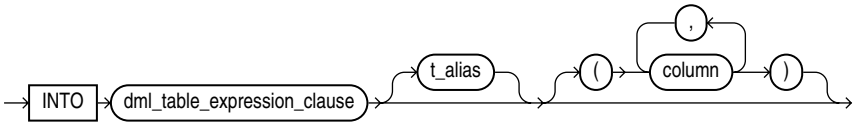
（18-54 ページの single_table_insert::=、18-55 ページの multi_table_insert::= を参照）

single_table_insert::=



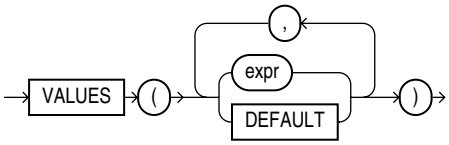
（18-55 ページの insert_into_clause::=、18-55 ページの values_clause::=、18-55 ページの returning_clause::=、19-5 ページの subquery::=、18-56 ページの error_logging_clause::= を参照）

insert_into_clause::=

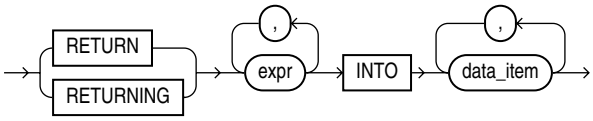


（18-56 ページの DML_table_expression_clause::= を参照）

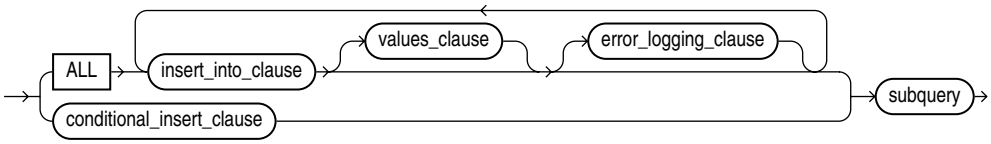
values_clause::=



returning_clause::=

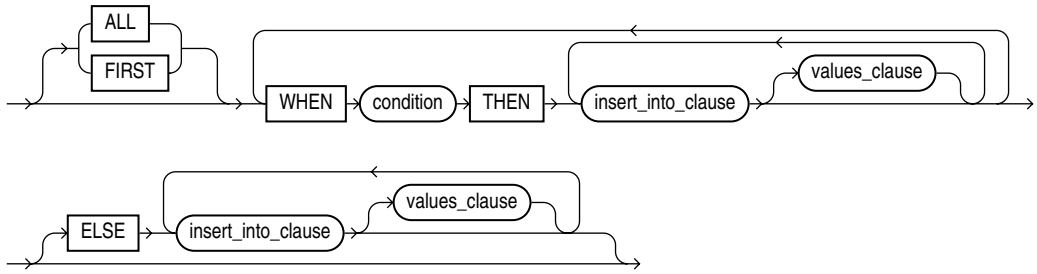


multi_table_insert::=



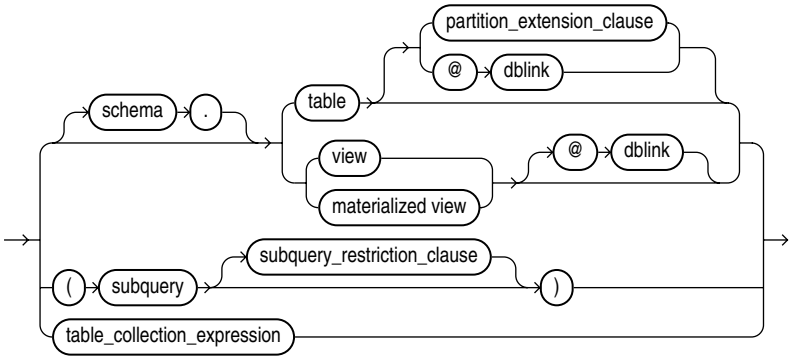
（18-55 ページの insert_into_clause::=、18-55 ページの values_clause::=、
18-55 ページの conditional_insert_clause::=、19-5 ページの subquery::=、
18-56 ページの error_logging_clause::= を参照）

conditional_insert_clause::=



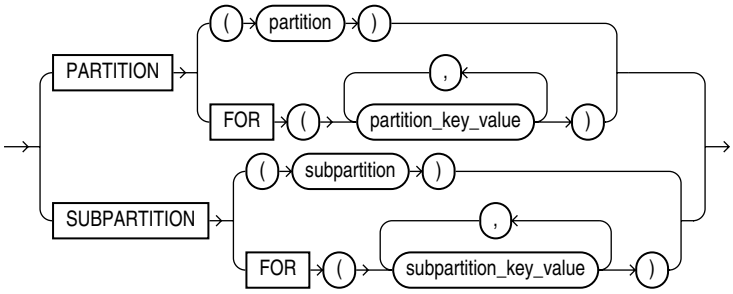
（18-55 ページの insert_into_clause::=、18-55 ページの values_clause::= を参照）

DML_table_expression_clause::=

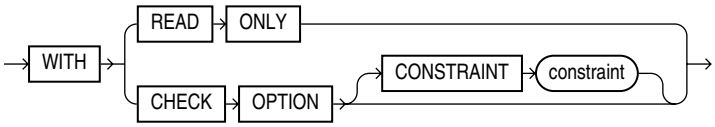


（18-70 ページの partition_extension_clause::=、SELECT 構文の項にある
19-5 ページの subquery::=、18-56 ページの subquery_restriction_clause::=、
18-56 ページの table_collection_expression::= を参照）

***partition_extension_clause*::=**

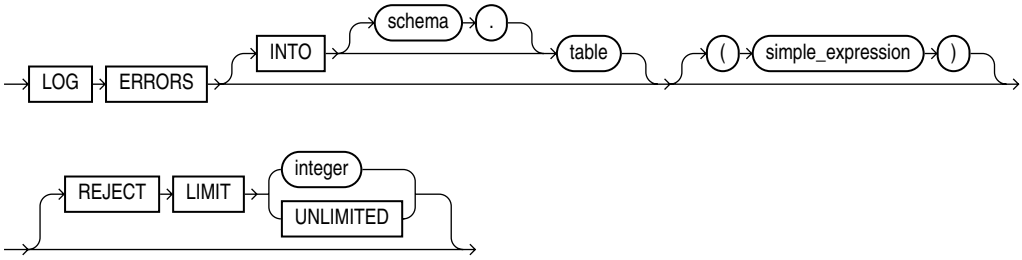


subquery_restriction_clause::=



table_collection_expression::=

error_logging_clause::=



セマンティクス

hint

文の実行計画を選択する場合に、オプティマイザに指示を与えるためのコメントを指定します。

マルチテーブル・インサートの場合、対象となる表に対して PARALLEL ヒントを指定すると、表が PARALLEL の指定付きで作成または変更されていなくても、マルチテーブル・インサート文全体がパラレル化されます。PARALLEL ヒントを指定しない場合、対象となるすべての表が PARALLEL の指定付きで作成または変更されていないかぎり、挿入操作はパラレル化されません。

参照 :

- ヒントの構文および説明については、2-70 ページの「ヒントの使用方法」を参照してください。
- 「マルチテーブル・インサートの制限事項 :」（18-62 ページ）

single_table_insert

シングルテーブル・インサートの場合、明示的に値を指定する、または副問合せで値を検索することによって、表、ビューまたはマテリアライズド・ビューの 1 行に値を挿入します。

副問合せで *flashback_query_clause* を使用すると、過去のデータを *table* に挿入できます。この句の詳細は、19-15 ページの「SELECT」の「flashback_query_clause」を参照してください。

シングルテーブル・インサートの制限事項 : 副問合せで値を検索する場合、副問合せの SELECT 構文のリストには INSERT 文の列リストと同じ数の列が含まれている必要があります。列リストを指定しない場合は、副問合せで表の各列の値を指定する必要があります。

参照 : 「表への値の挿入例 :」（18-64 ページ）

insert_into_clause

INSERT INTO 句を使用すると、データを挿入する対象となる表またはオブジェクトを指定できます。

DML_table_expression_clause

INTO *DML_table_expression_clause* を使用すると、データを挿入するオブジェクトを指定できます。

schema 表、ビューまたはマテリアライズド・ビューが含まれているスキーマを指定します。*schema* を指定しない場合、オブジェクトは自分のスキーマ内にあるとみなされます。

table* | *view* | *materialized_view* | *subquery 行を挿入する表またはオブジェクト表の名前、ビューまたはオブジェクト・ビューの名前、マテリアライズド・ビューの名前、あるいは副問合せから戻された列の名前を指定します。ビューまたはオブジェクト・ビューを指定した場合、そのビューの実表に行が挿入されます。

読取り専用マテリアライズド・ビューには行を挿入できません。書込み可能なマテリアライズド・ビューに行を挿入した場合、基礎となるコンテナ表にその行が挿入されます。ただし、その挿入操作は次のリフレッシュ操作によって上書きされます。マテリアライズド・ビュー・グループ内の更新可能なマテリアライズド・ビューに行を挿入した場合、対応する行もマスター表に挿入されます。

挿入される値がオブジェクト表に対する REF の場合、およびそのオブジェクト表に主キー・オブジェクト識別子がある場合、REF を挿入する列は、オブジェクト表に対する参照整合性制約または有効範囲制約を持つ REF 列である必要があります。

table または *view* の実表に、1 列以上のドメイン索引がある場合は、この文が適切な索引タイプの挿入ルーチンを実行します。

表に対して INSERT 文を発行した場合、その表に対して定義されている INSERT トリガーが起動します。

> **参照：** これらのルーチンの詳細は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。

***DML_table_expression_clause* の制限事項：** この句には、次の制限事項があります。

- *table* または *view* の実表に、IN_PROGRESS または FAILED とマークされたドメイン索引がある場合は、この文は実行できません。
- 関係する索引パーティションが UNUSABLE とマークされている場合は、パーティションに挿入できません。
- *DML_table_expression_clause* の *subquery* の ORDER BY 句に関して、順序付けは、挿入された行または表の各エクステント内のみに保証されています。既存の行に関連する新しい行の順序付けは保証されていません。
- WITH CHECK OPTION を指定してビューを作成した場合、ビューを定義する問合せを満たす行のみビューに挿入されます。
- 単一の実表を使用してビューを作成した場合、行をビューに挿入し、*returning_clause* を使用してその行の値を取り出せます。
- ビューを定義する問合せに次のいずれかの構造体が含まれる場合は、INSTEAD OF トリガーを使用する場合を除いて、そのビューに行を挿入できません。

  集合演算子
  DISTINCT 演算子
  集計ファンクションまたは分析ファンクション
  GROUP BY、ORDER BY、MODEL、CONNECT BY または START WITH 句
  SELECT 構文のリストにあるコレクション式
  SELECT 構文のリストにある副問合せ
  WITH READ ONLY が指定された副問合せ
  結合（一部の例外を除く。詳細は、『Oracle Database 管理者ガイド』を参照してください。）

- UNUSABLE のマークが付いている索引、索引パーティションまたは索引サブパーティションを指定する場合、SKIP_UNUSABLE_INDEXES セッション・パラメータが TRUE に設定されていないかぎり、INSERT 文は正常に実行されません。SKIP_UNUSABLE_INDEXES セッション・パラメータの詳細は、11-41 ページの「ALTER SESSION」を参照してください。

partition_extension_clause 挿入先の *table* または *view* の実表内にあるパーティションまたはサブパーティションの名前またはパーティション・キー値を指定します。

挿入する行が特定のパーティションまたはサブパーティションにマップされない場合、エラーが戻されます。

ターゲットのパーティションおよびサブパーティションの制限事項： この句は、オブジェクト表またはオブジェクト・ビューでは無効です。

> **参照：** 「パーティション表と索引の参照」（2-106 ページ）

dblink 表またはビューが格納されているリモート・データベースへのデータベース・リンクの完全名または部分名を指定します。Oracle Database の分散機能を使用している場合にのみ、リモート表またはリモート・ビューに行を挿入できます。

dblink を指定しない場合、その表またはビューはローカル・データベース内にあるとみなされます。

参照 :

- データベース・リンクの参照方法の詳細は、2-102 ページの「スキーマ・オブジェクトの構文および SQL 文の構成要素」および 2-104 ページの「リモート・データベース内のオブジェクトの参照」を参照してください。
- 「リモート・データベースへの挿入例 :」(18-65 ページ)

subquery_restriction_clause *subquery_restriction_clause* を使用すると、次のいずれかの方法で副問合せを制限できます。

WITH READ ONLY WITH READ ONLY を指定すると、表またはビューを更新禁止にできます。

WITH CHECK OPTION WITH CHECK OPTION を指定すると、副問合せに含まれない行を生成する表またはビューの変更を禁止できます。この句を DML 文の副問合せ内で使用する場合、FROM 句内の副問合せには指定できますが、WHERE 句内の副問合せには指定できません。

CONSTRAINT *constraint* CHECK OPTION 制約の名前を指定します。この識別子を省略した場合、その制約に SYS_C*n* という形式の名前が自動的に割り当てられます。この場合の n は、その制約名をデータベース内で一意の名前にする整数です。

参照 : 「WITH CHECK OPTION 句の使用例 :」(19-39 ページ)

table_collection_expression *table_collection_expression* を使用すると、問合せおよび DML 操作で、*collection_expression* 値を表として扱うことができます。*collection_expression* には、副問合せ、列、ファンクションまたはコレクション・コンストラクタのいずれかを指定できます。その形式にかかわらず、集合値(ネストした表型または VARRAY 型の値)を戻す必要があります。このようなコレクションの要素抽出プロセスを**コレクション・ネスト解除**といいます。

TABLE 式を親表と結合する場合は、オプションのプラス(+)には大きな意味があります。+ を指定すると、その 2 つの外部結合が作成され、コレクション式が NULL の場合でも、外部表の行が問合せで戻されるようになります。

注意 : 以前のリリースの Oracle では、*collection_expression* が副問合せの場合、*table_collection_expression* を THE *subquery* と表現していました。現在、このような表現方法は非推奨になっています。

参照 : 「表のコレクション例 :」(19-44 ページ)

t_alias

相関名を指定します。これは、文中で参照する表、ビュー、マテリアライズド・ビューまたは副問合せの別名です。

表の別名の制限事項 : *t_alias* は、マルチテーブル・インサートに指定できません。

column

表、ビューまたはマテリアライズド・ビューの列を指定します。挿入された行では、このリストにある各列に *values_clause* または副問合せの値が代入されます。

このリストに 1 つ以上の列を指定しない場合、挿入された行の、指定しなかった列の値には、表の作成時または最終更新時に指定した列のデフォルト値が使用されます。指定しなかった列のいずれかに NOT NULL 制約があり、デフォルト値がない場合、制約違反のエラーが発生して INSERT 文がロールバックされます。列のデフォルト値の詳細は、16-6 ページの「CREATE TABLE」を参照してください。

列リストを指定しない場合は、*values_clause* または問合せに、表の列をすべて指定する必要があります。

values_clause

シングルテーブル・インサート操作の場合、表またはビューに挿入する値の行を指定します。なお、値は、列リスト内の各列について *values_clause* に指定する必要があります。列リストを指定しない場合、*values_clause* または副問合せで、表の各列の値を指定する必要があります。

マルチテーブル・インサート操作の場合、*values_clause* の各式は、副問合せの SELECT 構文のリストによって戻される列を参照する必要があります。*values_clause* を省略すると、副問合せの SELECT 構文のリストによって挿入する値が決定されるため、*insert_into_clause* に対応する列リストと同じ列数を含んでいる必要があります。*insert_into_clause* で列リストを指定しない場合、対象となる表のすべての列に対する値を計算した行を指定する必要があります。

どちらの挿入操作の場合も、*insert_into_clause* で列リストを指定すると、リストの各列に値の句または副問合せからの対応する値が割り当てられます。*values_clause* の任意の値に対して DEFAULT を指定できます。表またはビューの対応する列に対してデフォルト値を指定すると、その値が挿入されます。対応する列に対してデフォルト値を指定しない場合、NULL が挿入されます。有効な式の構文の詳細は、6-2 ページの「SQL 式」および 19-4 ページの「SELECT」を参照してください。

注意： パラレル・ダイレクト・パス・インサートは、*values_clause* はサポートせず、INSERT 文の副問合せ構文のみサポートします。シリアル・ダイレクト・パス・インサートおよびパラレル・ダイレクト・パス・インサートの詳細は、『Oracle Database 管理者ガイド』を参照してください。

挿入値の制限事項： この値には、次の制限事項があります。

- BFILE ロケータを NULL に、またはディレクトリ名およびファイル名に初期化するまで、BFILE 値を挿入できません。

  参照：

  - BFILE 値の初期化の詳細および BFILE への挿入例については、5-21 ページの「BFILENAME」を参照してください。
  - BFILE ロケータの初期化の詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』を参照してください。

- リスト・パーティション表に挿入する場合、1 つのパーティションの *partition_value* リストに存在していないパーティション化キー列に値を挿入できません。
- ビューに挿入する場合は、DEFAULT を指定できません。
- RAW 列に文字列リテラルを挿入する場合、後続の問合せ中に RAW 列にある索引は使用されずに、全表スキャンが行われます。

  参照：

  - XMLType 表への値の挿入の詳細は、E-8 ページの「SQL 文での XML の使用方法」を参照してください。
  - 18-65 ページの「置換可能な表および列への挿入例 :」、18-65 ページの「TO_LOB ファンクションを使用した挿入例 :」、18-65 ページの「順序値の挿入例 :」および 18-65 ページの「バインド変数を使用した挿入例 :」を参照してください。

returning_clause

この句を使用すると、DML 文に影響される行を取り出すことができます。この句は、表、マテリアライズド・ビュー、および単一の実表を持つビューに指定できます。

returning_clause を指定した DML 文を単一行に実行すると、影響された行、ROWID、および処理された行への REF を使用している列式が取り出され、ホスト変数または PL/SQL 変数に格納されます。

returning_clause を指定した DML 文を複数行に実行すると、式の値、ROWID および処理された行に関連する REF がバインド配列に格納されます。

expr *expr* リストの各項目は、適切な構文で表す必要があります。

INTO INTO 句を指定すると、変更された行の値を、*data_item* リストに指定する変数に格納できます。

data_item 取り出された *expr* 値を格納するホスト変数または PL/SQL 変数を指定します。

RETURNING リストの各式については、INTO リストに、対応する型に互換性がある PL/SQL 変数またはホスト変数を指定する必要があります。

RETURNING 句の制限事項 : RETURNING 句には、次の制限事項があります。

- *expr* に次の制限事項があります。
  - UPDATE 文および DELETE 文の場合、各 *expr* は、単純式または単一セットの集計ファンクション式である必要があります。1 つの *returning_clause* 内に単純式と単一セットの集計ファンクション式を混在させることはできません。INSERT 文の場合、各 *expr* は単純式である必要があります。INSERT 文の RETURNING 句では、集計ファンクションはサポートされていません。
  - 単一セットの集計ファンクション式を DISTINCT キーワードに含めることはできません。
- *expr* リストに主キー列またはその他の NOT NULL 列が含まれている場合、表に BEFORE UPDATE トリガーが定義されていると、**UPDATE** 文は正常に実行されません。
- マルチテーブル・インサートでは *returning_clause* を指定できません。
- パラレル DML またはリモート・オブジェクトにはこの句を使用できません。
- LONG 型を取り出すことはできません。
- INSTEAD OF トリガーが定義されたビューに対して指定することはできません。

> **参照 :** BULK COLLECT 句を使用してコレクション変数に複数の値を戻す場合は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

multi_table_insert

マルチテーブル・インサートでは、副問合せの評価によって戻された行から導出され、計算された行が 1 つ以上の表に挿入されます。

副問合せの SELECT 構文のリストでは、表の別名は定義されていません。そのため、SELECT 構文のリストに依存する句では、表の別名は参照できません。たとえば、式内のオブジェクト列を参照しようとしても、表の別名は参照できません。表の別名とともに式を使用する場合は、列の別名を使用して式を SELECT 構文のリストに挿入してから、マルチテーブル・インサートの VALUES 句または WHEN 条件で列の別名を参照する必要があります。

ALL *into_clause*

ALL の後に複数の *insert_into_clauses* を指定すると、**無条件のマルチテーブル・インサート**を実行できます。副問合せによって戻される各行に対して、各 *insert_into_clause* が 1 回実行されます。

conditional_insert_clause

conditional_insert_clause を指定すると、**条件付きのマルチテーブル・インサート**を実行できます。各 *insert_into_clause* は、それに対応する WHEN 条件で実行されるかどうかが判断されます。WHEN 条件の各式は、副問合せの SELECT 構文のリストによって戻される列を参照する必要があります。1 つのマルチテーブル・インサートには、最大 127 個の WHEN 句を指定できます。

ALL ALL（デフォルト値）を指定すると、他の WHEN 句の評価結果にかかわらず、各 WHEN 句が評価されます。条件が真と評価された各 WHEN 句に対して、対応する INTO 句リストが実行されます。

FIRST FIRST を指定すると、文で指定されている順序で各 WHEN 句が評価されます。真と評価された最初の WHEN 句に対して、対応する INTO 句が実行され、指定された行に対する後続の WHEN 句はスキップされます。

ELSE 句 指定された行に対して、WHEN 句が真と評価されない場合、次のようになります。

- ELSE 句を指定する場合、ELSE 句に関連付けられた INTO 句リストが実行されます。
- ELSE 句を指定しない場合、その行に対して何も実行されません。

参照： 「マルチテーブル・インサートの例 :」（18-66 ページ）

マルチテーブル・インサートの制限事項： マルチテーブル・インサートには、次の制限事項があります。

- 表のみにマルチテーブル・インサートが実行でき、ビューまたはマテリアライズド・ビューには実行できません。
- リモート表には、マルチテーブル・インサートを実行できません。
- マルチテーブル・インサートの実行時、表のコレクション式は指定できません。
- マルチテーブル・インサートでは、すべての *insert_into_clause* で組み合せて指定できる列が最大 999 です。
- マルチテーブル・インサートは、対象となる表が索引構成されている場合、または対象となる表にビットマップ索引が定義されている場合は、パラレル化されません。
- プラン・スタビリティは、マルチテーブル・インサート文にはサポートされません。
- マルチテーブル・インサート文のどの部分にも順序を指定することはできません。マルチテーブル・インサートは、単一の SQL 文とみなされます。したがって、NEXTVAL を初めて参照するときに、その次の番号が生成された後、この番号と同じ番号が、この文の後続のすべての参照で戻されます。

subquery

表に挿入される行を戻す副問合せを指定します。副問合せによって、INSERT 文の対象となる表を含む任意の表、ビューおよびマテリアライズド・ビューを参照できます。副問合せで選択された行が 1 行もない場合、表に行は挿入されません。

subquery を TO_LOB ファンクションと組み合せて、LONG 列にある値を、同じ表の異なる列または別の表にある LOB 値に変換できます。ビュー内で LONG 値を別の列の LOB 値に移行する場合、実表内で移行を実行してからビューに LOB 列を追加する必要があります。

副問合せを使用する挿入の注意事項： 副問合せを使用した挿入には、次の注意事項があります。

- *subquery* が、既存のマテリアライズド・ビューと部分的または完全に同じビューを戻す場合、*subquery* に指定された 1 つ以上の表のかわりにマテリアライズド・ビューがクエリー・リライトに使用されることがあります。

参照： マテリアライズド・ビューおよびクエリー・リライトの詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

- *subquery* がリモート・オブジェクトを参照する場合、参照がローカル・データベース上でオブジェクトにループバックしないかぎり、INSERT はパラレルで実行されます。ただし、*DML_table_expression_clause* の *subquery* がリモート・オブジェクトを参照する場合は、INSERT 操作はシリアルで実行されます。詳細は、16-54 ページの「parallel_clause」を参照してください。

  参照：

  - 「副問合せを持つ値の挿入例 :」（18-64 ページ）
  - BFILE への挿入例については、5-21 ページの「BFILENAME」を参照してください。
  - BFILE の初期化の詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』を参照してください。
  - 有効な式の構文の詳細は、6-2 ページの「SQL 式」および 19-4 ページの「SELECT」を参照してください。

error_logging_clause

error_logging_clause を使用すると、DML エラーと、その影響を受ける行のログ列値を取得して、エラー・ロギング表に保存できます。

INTO *table* エラー・ロギング表の名前を指定します。この句を省略すると、DBMS_ERRLOG パッケージで生成されたデフォルトの名前が割り当てられます。エラー・ログ表のデフォルトの名前は、DML 操作の対象となっている表の名前の最初の 25 文字を、ERR$_ の後に加えたものです。

simple_expression 文のタグとして使用する値を指定します。エラー・ロギング表内のエラーは、この文で識別することになります。この式には、テキスト・リテラル、数値リテラル、一般の SQL 式（バインド変数など）のいずれかを指定できます。テキスト・リテラルに変換する場合は、たとえば TO_CHAR(SYSDATE) のようなファンクション式も使用できます。

REJECT LIMIT この句を使用すると、記録するエラーの数の上限値を整数で指定できます。エラーの数がこの値を超えると、文が終了し、その文によって変更された内容がロールバックされます。この拒否の制限のデフォルトは 0（ゼロ）です。パラレル DML 操作の場合、拒否の制限はパラレル・サーバーごとに適用されます。

DML エラー・ロギングの制限事項 :

- 次の状態では、文の実行が失敗しロールバックは実行されますが、エラー・ロギング機能は起動されません。
  - 遅延制約の違反
  - 一意制約違反または一意索引違反を発生する、ダイレクト・パスの INSERT または MERGE 操作
  - 一意制約違反または一意索引違反を発生する、更新操作の UPDATE または MERGE
- エラー・ロギング表では、LONG、LOB またはオブジェクト型の列のエラーは追跡できません。ただし、DML 操作を行う表に、これらの型の列を含めることができます。
  - 対応するエラー・ロギング表を、未サポートの型の列を含めるために作成または修正する場合、その列の名前が、ターゲットの DML 表の未サポートの列に対応していると、DML 文が解析時にエラーになります。
  - エラー・ロギング表に未サポートの列型が含まれない場合、エラー数が拒否の制限に達するまで、すべての DML エラーが記録されます。エラーが発生した行では、列の値とエラー・ロギング表の対応する列が、制御情報とともにログに記録されます。

参照：

- DBMS_ERRLOG パッケージの create_error_log プロシージャの使用方法は、『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。DML エラー・ロギングに関する一般情報は、『Oracle Database 管理者ガイド』を参照してください。
- 「エラー・ロギングによる表への挿入例：」（18-64 ページ）

例

表への値の挿入例： 次の文は、サンプル表 departments に行を挿入します。

```
INSERT INTO departments
   VALUES (280, 'Recreation', 121, 1700);
```

manager_id 列のデフォルト値が 121 として departments 表が作成された場合、次の文を発行できます。

```
INSERT INTO departments
   VALUES (280, 'Recreation', DEFAULT, 1700);
```

次の文は、employees 表に 6 つの列で構成される行を挿入します。NULL または科学表記の数値を設定されている列がそれぞれ 1 つ含まれています。

```
INSERT INTO employees (employee_id, last_name, email,
      hire_date, job_id, salary, commission_pct)
   VALUES (207, 'Gregory', 'pgregory@example.com',
      sysdate, 'PU_CLERK', 1.2E3, NULL);
```

次の文は、前述の例と同じ結果を表しますが、*DML_table_expression_clause* にある副問合せを使用します。

```
INSERT INTO
   (SELECT employee_id, last_name, email, hire_date, job_id,
      salary, commission_pct FROM employees)
   VALUES (207, 'Gregory', 'pgregory@example.com',
      sysdate, 'PU_CLERK', 1.2E3, NULL);
```

副問合せを持つ値の挿入例： 次の文は、歩合給が給与の 25% を超える従業員を bonuses 表（18-74 ページの「表へのマージ例：」で作成）にコピーします。

```
INSERT INTO bonuses
   SELECT employee_id, salary*1.1
   FROM employees
   WHERE commission_pct > 0.25;
```

エラー・ロギングによる表への挿入例： 次の文は、サンプル・スキーマ hr 内に raises 表を作成し、DBMS_ERRLOG パッケージを使用してエラー・ロギング表を作成して、employees 表のデータを raises 表に移入します。raises のチェック制約に違反する挿入操作があると、その行を errlog で参照できます。発生したエラーが 10 を超えると、その文は異常終了し、挿入された内容はロールバックされます。

```
CREATE TABLE raises (emp_id NUMBER, sal NUMBER
   CONSTRAINT check_sal CHECK(sal > 8000));

EXECUTE DBMS_ERRLOG.CREATE_ERROR_LOG('raises', 'errlog');

INSERT INTO raises
   SELECT employee_id, salary*1.1 FROM employees
   WHERE commission_pct > .2
   LOG ERRORS INTO errlog ('my_bad') REJECT LIMIT 10;
```

```
SELECT ORA_ERR_MESG$, ORA_ERR_TAG$, emp_id, sal FROM errlog;

ORA_ERR_MESG$               ORA_ERR_TAG$         EMP_ID SAL
--------------------------- -------------------- ------ -------
ORA-02290: check constraint my_bad               161    7700
 (HR.SYS_C004266) violated
```

リモート・データベースへの挿入例： 次の文は、データベース・リンク remote がアクセスできるデータベース上の、ユーザー hr が所有する employees 表に行を挿入します。

```
INSERT INTO employees@remote
   VALUES (8002, 'Juan', 'Fernandez', 'juanf@hr.example.com', NULL,
   TO_DATE('04-OCT-1992', 'DD-MON-YYYY'), 'SH_CLERK', 3000,
   NULL, 121, 20);
```

順序値の挿入例： 次の文は、departments_seq 順序の次の値を持つ行を、departments 表に挿入します。

```
INSERT INTO departments
   VALUES  (departments_seq.nextval, 'Entertainment', 162, 1400);
```

バインド変数を使用した挿入例： 次の文は、出力バインド変数 bnd1 および bnd2 に挿入された行の値を戻します。バインド変数は最初に宣言しておく必要があります。

```
INSERT INTO employees
      (employee_id, last_name, email, hire_date, job_id, salary)
   VALUES
   (employees_seq.nextval, 'Doe', 'john.doe@example.com',
       SYSDATE, 'SH_CLERK', 2400)
   RETURNING salary*12, job_id INTO :bnd1, :bnd2;
```

置換可能な表および列への挿入例： 次の例は、persons 表（16-61 ページの「置換可能な表および列のサンプル：」で作成）に挿入します。最初の文は、ルート型 person_t を使用します。2 番目の挿入は、person_t のサブタイプ employee_t を使用し、3 番目の挿入は employee_t のサブタイプ part_time_emp_t を使用します。

```
INSERT INTO persons VALUES (person_t('Bob', 1234));
INSERT INTO persons VALUES (employee_t('Joe', 32456, 12, 100000));
INSERT INTO persons VALUES (
   part_time_emp_t('Tim', 5678, 13, 1000, 20));
```

次の例は、books 表（16-61 ページの「置換可能な表および列のサンプル：」で作成）に挿入します。属性値の指定が、置換可能な表の例と同一であることに注意してください。

```
INSERT INTO books VALUES (
   'An Autobiography', person_t('Bob', 1234));
INSERT INTO books VALUES (
   'Business Rules', employee_t('Joe', 3456, 12, 10000));
INSERT INTO books VALUES (
   'Mixing School and Work',
   part_time_emp_t('Tim', 5678, 13, 1000, 20));
```

組込みファンクションおよび条件を使用して、置換可能な表および列からデータを抽出することができます。その例は、5-215 ページの「TREAT」、5-191 ページの「SYS_TYPEID」および 7-23 ページの「IS OF type 条件」を参照してください。

TO_LOB ファンクションを使用した挿入例： 次の例は、LONG データを次の long_tab 表にある LOB 列にコピーします。

```
CREATE TABLE long_tab (pic_id NUMBER, long_pics LONG RAW);
```

まず、LOB を持つ表を作成します。

```
CREATE TABLE lob_tab (pic_id NUMBER, lob_pics BLOB);
```

次に、INSERT ... SELECT を使用して、LONG 列のすべての行にあるデータを、新しく作成した LOB 列にコピーします。

```
INSERT INTO lob_tab
   SELECT pic_id, TO_LOB(long_pics) FROM long_tab;
```

移行が問題なく終了したことを確認したら、long_pics 表を削除できます。別の方法として、表が他の列を含む場合、次のように入力して表から LONG 列を削除できます。

```
ALTER TABLE long_tab DROP COLUMN long_pics;
```

マルチテーブル・インサートの例： 次の例はマルチテーブル・インサートの構文を使用して、サンプル表 sh.sales に、異なる構造の入力表からデータを挿入します。

> **注意：** 例を簡潔に示すために表の列が無視されているため、sales 表の NOT NULL 制約は使用禁止になっています。

入力表は次のように構成されています。

```
SELECT * FROM sales_input_table;

PRODUCT_ID CUSTOMER_ID WEEKLY_ST  SALES_SUN  SALES_MON  SALES_TUE  SALES_WED SALES_THU  SALES_FRI  SALES_SAT
---------- ----------- --------- ---------- ---------- ---------- ---------- ---------- ---------- ----------
       111         222 01-OCT-00        100        200        300        400        500        600        700
       222         333 08-OCT-00        200        300        400        500        600        700        800
       333         444 15-OCT-00        300        400        500        600        700        800        900
```

マルチテーブル・インサートの文を次に示します。

```
INSERT ALL
      INTO sales (prod_id, cust_id, time_id, amount)
      VALUES (product_id, customer_id, weekly_start_date, sales_sun)
      INTO sales (prod_id, cust_id, time_id, amount)
      VALUES (product_id, customer_id, weekly_start_date+1, sales_mon)
      INTO sales (prod_id, cust_id, time_id, amount)
      VALUES (product_id, customer_id, weekly_start_date+2, sales_tue)
      INTO sales (prod_id, cust_id, time_id, amount)
      VALUES (product_id, customer_id, weekly_start_date+3, sales_wed)
      INTO sales (prod_id, cust_id, time_id, amount)
      VALUES (product_id, customer_id, weekly_start_date+4, sales_thu)
      INTO sales (prod_id, cust_id, time_id, amount)
      VALUES (product_id, customer_id, weekly_start_date+5, sales_fri)
      INTO sales (prod_id, cust_id, time_id, amount)
      VALUES (product_id, customer_id, weekly_start_date+6, sales_sat)
   SELECT product_id, customer_id, weekly_start_date, sales_sun,
      sales_mon, sales_tue, sales_wed, sales_thu, sales_fri, sales_sat
      FROM sales_input_table;
```

sales 表には次の行のみが存在するものとします。内容は次のとおりです。

```
SELECT * FROM sales
   ORDER BY prod_id, cust_id, time_id;

   PROD_ID    CUST_ID TIME_ID   C   PROMO_ID QUANTITY_SOLD     AMOUNT       COST
---------- ---------- --------- - ---------- ------------- ---------- ----------
       111        222 01-OCT-00                                  100
       111        222 02-OCT-00                                  200
       111        222 03-OCT-00                                  300
```

```
       111        222 04-OCT-00                                 400
       111        222 05-OCT-00                                 500
       111        222 06-OCT-00                                 600
       111        222 07-OCT-00                                 700
       222        333 08-OCT-00                                 200
       222        333 09-OCT-00                                 300
       222        333 10-OCT-00                                 400
       222        333 11-OCT-00                                 500
       222        333 12-OCT-00                                 600
       222        333 13-OCT-00                                 700
       222        333 14-OCT-00                                 800
       333        444 15-OCT-00                                 300
       333        444 16-OCT-00                                 400
       333        444 17-OCT-00                                 500
       333        444 18-OCT-00                                 600
       333        444 19-OCT-00                                 700
       333        444 20-OCT-00                                 800
       333        444 21-OCT-00                                 900
```

次の例は、複数の表に挿入します。販売員に様々なサイズの注文に関する情報を提供するとします。小口、中口、大口および特別注文についての表を作成し、これらの表にサンプル表 oe.orders のデータを移入します。

```
CREATE TABLE small_orders
   (order_id       NUMBER(12)   NOT NULL,
    customer_id    NUMBER(6)    NOT NULL,
    order_total    NUMBER(8,2),
    sales_rep_id   NUMBER(6)
   );

CREATE TABLE medium_orders AS SELECT * FROM small_orders;

CREATE TABLE large_orders AS SELECT * FROM small_orders;

CREATE TABLE special_orders
   (order_id       NUMBER(12)    NOT NULL,
    customer_id    NUMBER(6)     NOT NULL,
    order_total    NUMBER(8,2),
    sales_rep_id   NUMBER(6),
    credit_limit   NUMBER(9,2),
    cust_email     VARCHAR2(30)
   );
```

最初のマルチテーブル・インサートでは、小口、中口および大口注文の表に対してのみ移入を行います。

```
INSERT ALL
   WHEN order_total < 1000000 THEN
      INTO small_orders
   WHEN order_total > 1000000 AND order_total < 2000000 THEN
      INTO medium_orders
   WHEN order_total > 2000000 THEN
      INTO large_orders
   SELECT order_id, order_total, sales_rep_id, customer_id
      FROM orders;
```

large_orders 表への挿入のかわりに、ELSE 句を使用しても、同じ結果が得られます。

```
INSERT ALL
   WHEN order_total < 100000 THEN
      INTO small_orders
   WHEN order_total > 100000 AND order_total < 200000 THEN
      INTO medium_orders
   ELSE
```

```
      INTO large_orders
   SELECT order_id, order_total, sales_rep_id, customer_id
      FROM orders;
```

次の例は、前述の例と同様、小口、中口および大口注文の表に挿入し、件数が 2,900,000 を超える注文を special_orders 表に挿入します。この表は、文を単純にするために列の別名を使用する方法も示しています。

```
INSERT ALL
   WHEN ottl < 100000 THEN
      INTO small_orders
         VALUES(oid, ottl, sid, cid)
   WHEN ottl > 100000 and ottl < 200000 THEN
      INTO medium_orders
         VALUES(oid, ottl, sid, cid)
   WHEN ottl > 200000 THEN
      into large_orders
         VALUES(oid, ottl, sid, cid)
   WHEN ottl > 290000 THEN
      INTO special_orders
   SELECT o.order_id oid, o.customer_id cid, o.order_total ottl,
      o.sales_rep_id sid, c.credit_limit cl, c.cust_email cem
      FROM orders o, customers c
      WHERE o.customer_id = c.customer_id;
```

最後の例は、FIRST 句を使用して、件数が 2,900,000 を超える注文を special_orders 表に挿入し、これらの注文を large_orders 表から削除します。

```
INSERT FIRST
   WHEN ottl < 100000 THEN
      INTO small_orders
         VALUES(oid, ottl, sid, cid)
   WHEN ottl > 100000 and ottl < 200000 THEN
      INTO medium_orders
         VALUES(oid, ottl, sid, cid)
   WHEN ottl > 290000 THEN
      INTO special_orders
   WHEN ottl > 200000 THEN
      INTO large_orders
         VALUES(oid, ottl, sid, cid)
   SELECT o.order_id oid, o.customer_id cid, o.order_total ottl,
      o.sales_rep_id sid, c.credit_limit cl, c.cust_email cem
      FROM orders o, customers c
      WHERE o.customer_id = c.customer_id;
```

LOCK TABLE

用途

LOCK TABLE 文を使用すると、1 つ以上の表、表パーティションまたは表サブパーティションを特定のモードでロックできます。操作中の表またはビューに対する他のユーザーによるアクセスを許可または制限するため、自動ロックを手動で無効にします。

ロックによっては、同じ表に同時に設定できる場合、または表ごとに 1 つのみ設定できる場合があります。

ロックされた表は、トランザクションをコミットするか、全体をロールバックするか、または表をロックする前のセーブポイントにロールバックするまでロックされています。

ロックした場合でも他のユーザーが表を問い合せることができます。問合せによって表がロックされることはありません。読取りプログラムは書込みプログラムをブロックすることはなく、書込みプログラムが読取りプログラムをブロックすることもありません。

参照 :

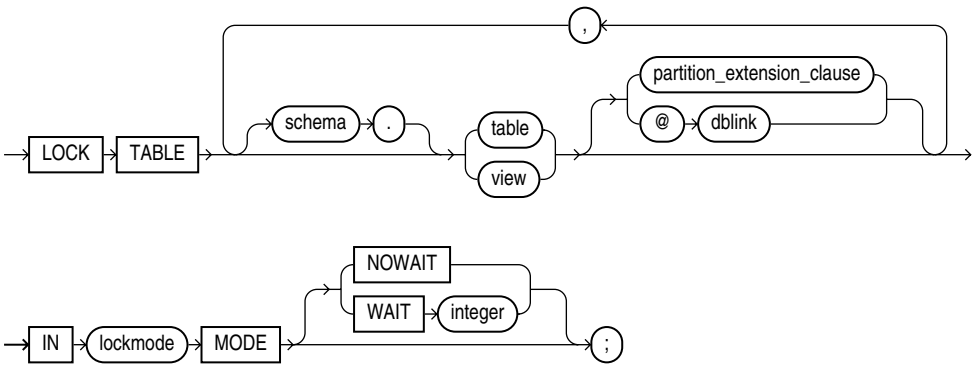
- ロック・モードの相互作用については、『Oracle Database 概要』を参照してください。
- 「COMMIT」（13-44 ページ）
- 「ROLLBACK」（18-92 ページ）
- 「SAVEPOINT」（19-2 ページ）

前提条件

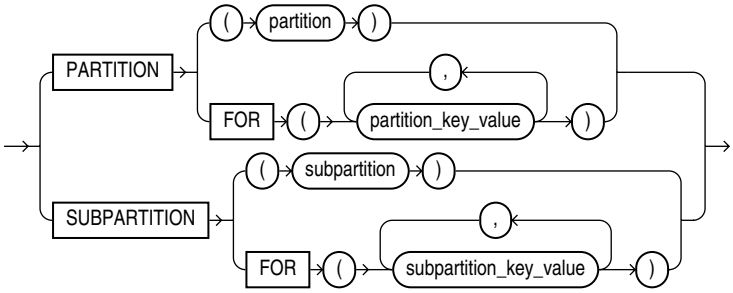
表またはビューが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、LOCK ANY TABLE システム権限、または表やビューに対するオブジェクト権限が必要です。

構文

***lock_table*::=**



***partition_extension_clause*::=**



セマンティクス

schema

表またはビューが含まれているスキーマを指定します。*schema* を指定しない場合、表またはビューは自分のスキーマにあるとみなされます。

table / *view*

ロックする表の名前を指定します。

view を指定した場合、ビューの実表がロックされます。

partition_extension_clause を指定した場合、Oracle Database では最初にその表が暗黙的にロックされます。表ロックは、partition または subpartition に指定したロックと同じです。ただし、次の 2 つの例外があります。

- SHARE ロックをサブパーティションに指定した場合、表が暗黙的に ROW SHARE ロックされます。
- EXCLUSIVE ロックをサブパーティションに指定した場合、表が暗黙的に ROW EXCLUSIVE ロックされます。

PARTITION を指定し、*table* がコンポジット・パーティション化されている場合、パーティションのすべてのサブパーティションがロックされます。

表のロックの制限事項： *view* が階層の一部である場合、階層のルートである必要があります。

dblink

表またはビューが格納されている、Oracle Database のリモート・データベースに対するデータベース・リンクを指定します。Oracle 分散機能を使用している場合のみ、リモート・データベースで表およびビューをロックできます。LOCK TABLE 文を使用してロックする表は、すべて同じデータベース上にある必要があります。

dblink を指定しない場合、その表またはビューはローカル・データベース内にあるとみなされます。

> **参照：** データベース・リンクの指定方法の詳細は、2-104 ページの「リモート・データベース内のオブジェクトの参照」を参照してください。

lockmode 句

次のいずれかのモードを指定します。

ROW SHARE ROW SHARE を指定すると、ロックされた表への同時アクセスは可能になりますが、排他アクセスのために表全体をロックできなくなります。ROW SHARE は、SHARE UPDATE と同じ意味で、以前のリリースの Oracle Database との互換性を保つために用意されています。

ROW EXCLUSIVE ROW EXCLUSIVE は、ROW SHARE と同じですが、SHARE モードでロックはできません。ROW EXCLUSIVE ロックは、更新、挿入、削除の実行時に自動的に適用されます。

SHARE UPDATE 18-70 ページの「ROW SHARE」を参照してください。

SHARE SHARE を指定すると、同時問合せは実行可能ですが、ロックされた表は更新できなくなります。

SHARE ROW EXCLUSIVE SHARE ROW EXCLUSIVE は、表全体を見る場合に使用します。これを使用すると他のユーザーがその表内の行を見ることはできますが、SHARE モードで表のロックまたは行の更新を行うことはできません。

EXCLUSIVE EXCLUSIVE を指定すると、ロックされた表上での問合せは実行可能ですが、他のアクティビティは実行できなくなります。

NOWAIT

NOWAIT を指定すると、指定した表、パーティションまたは表のサブパーティションが他のユーザーによってすでにロックされている場合に、制御をすぐに戻すことができます。この場合、表、パーティションまたはサブパーティションが他のユーザーによってロックされていることを示すエラー・メッセージが戻ります。

WAIT

WAIT 句を使用すると、LOCK TABLE 文では、DML ロックを取得するまでに指定した時間（秒数）待機するように指定できます。*integer* の値に制限はありません。

NOWAIT も WAIT も指定しない場合には、表が利用可能になり、ロックされ、制御が戻されるまで、データベースは無限に待機します。データベースで DML 文と同時に DDL 文が実行されている場合、タイムアウトまたはデッドロックが発生することがあります。このようなタイムアウトまたはデッドロックが検出されると、エラーが戻されます。

> **参照：** 表のロックの詳細は、『Oracle Database 管理者ガイド』を参照してください。

例

表のロック例： 次の文は、employees 表を排他モードでロックします。他のユーザーがすでに表をロックしている場合でも、待ち状態にはなりません。

```
LOCK TABLE employees
   IN EXCLUSIVE MODE
   NOWAIT;
```

次の文は、データベース・リンク remote を介してアクセスできるリモート表 employees をロックします。

```
LOCK TABLE employees@remote
   IN SHARE MODE;
```

MERGE

用途

MERGE 文を使用すると、1 つ以上のソースから行を選択し、表またはビューに対して更新および挿入できます。対象となる表またはビューに対して更新と挿入のどちらを実行するかを決定する条件を指定できます。

この文は、複数の操作を組み合せるときに便利です。DML 文 INSERT、UPDATE および DELETE を複数指定する必要がなくなります。

MERGE は、決定的な文です。対象となる表の同じ行を、同一の MERGE 文で何度も更新することはできません。

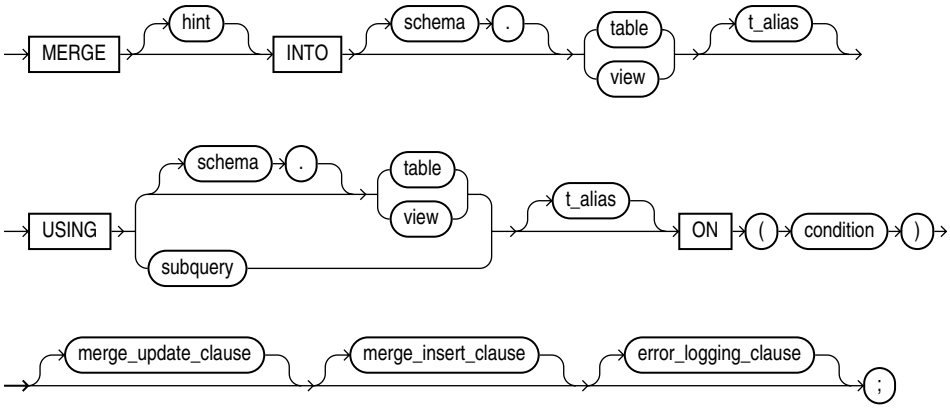
> **注意：** MERGE 文では、ファイングレイン・アクセス・コントロールが実行されません。操作対象の 1 つまたは複数の表でファイングレイン・アクセス・コントロール機能を使用する場合は、MERGE 文ではなく、それと等価な INSERT および UPDATE 文を使用して、エラー・メッセージを避け、適正なアクセス制御を行ってください。

前提条件

対象となる表に対する INSERT オブジェクト権限と UPDATE オブジェクト権限、およびソース表に対する SELECT オブジェクト権限が必要です。*merge_update_clause* の DELETE 句を指定するには、対象となる表に対する DELETE オブジェクト権限も必要です。

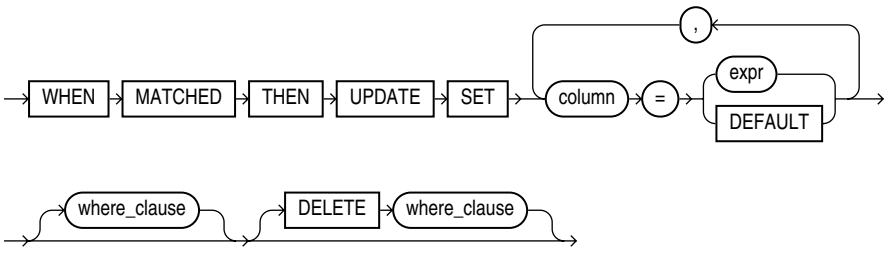
構文

***merge*::=**

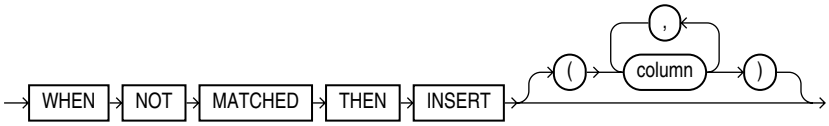


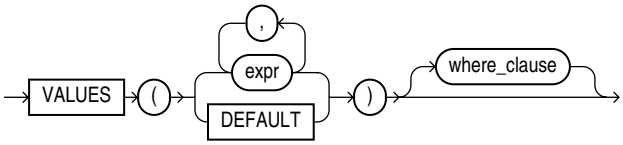
（18-72 ページの merge_update_clause::=、18-73 ページの merge_insert_clause::=、18-73 ページの error_logging_clause::= を参照）

merge_update_clause::=



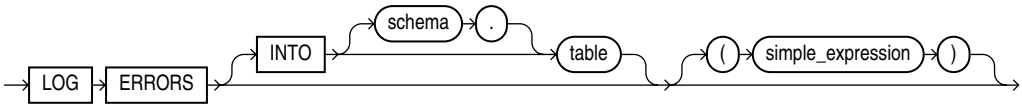
merge_insert_clause::=

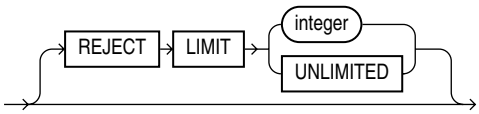




***where_clause*::=**

error_logging_clause::=





セマンティクス

INTO 句

INTO 句を使用すると、更新または挿入の対象となる表またはビューを指定できます。データをビューにマージする場合、そのビューは更新可能であることが必要です。詳細は、17-18 ページの「更新可能なビューの注意事項 :」を参照してください。

USING 句

USING 句を使用すると、更新または挿入の対象となるデータのソースを指定できます。ソースには、表、ビューまたは副問合せの結果を指定できます。

ON 句

ON 句を使用すると、MERGE の更新操作または挿入操作の条件を指定できます。対象となる表の中で検索条件が真となる各行は、ソース表の対応するデータに基づいて行が更新されます。どの行も条件が真とならない場合、ソース表の対応する行に基づいて対象となる表に行が挿入されます。

ON 句の制限事項 : MERGE 文では、ファイングレイン・アクセス・コントロールが実行されません。操作対象の 1 つまたは複数の表でファイングレイン・アクセス・コントロール機能を使用する場合は、MERGE 文ではなく、それと等価な INSERT および UPDATE 文を使用して、エラー・メッセージを避け、適正なアクセス制御を行ってください。

merge_update_clause

merge_update_clause を指定すると、対象となる表の新しい列値を指定できます。ON 句の条件が真となる場合、更新が実行されます。更新が実行されると、対象となる表に定義されているすべての更新トリガーがアクティブになります。

where_clause を指定すると、指定した条件が真の場合のみに更新操作が実行されるようにできます。条件には、データ・ソースまたは対象となる表を参照できます。条件が真ではない場合、行を表に挿入する際に更新操作がスキップされます。

DELETE *where_clause* を指定すると、表の移入中または更新中にその表内のデータをクリーンアップできます。この句によって処理される行は、マージ操作によって更新される対象の表内の行のみです。DELETE WHERE 条件は、更新後の値を評価し、UPDATE SET ... WHERE 条件によって評価された元の値は評価しません。更新先の表の行が DELETE 条件を満たし、ON 句によって定義された結合に含まれていない場合、その行は削除されます。更新先の表に定義されている削除トリガーが起動し、各行が削除されます。

この句は、単独で、または *merge_insert_clause* とともに指定できます。*merge_insert_clause* とともに指定する場合は、どちらを先に指定してもかまいません。

***merge_update_clause* の制限事項 :** この句には、次の制限事項があります。

- 参照する列は、ON *condition* 句で更新できません。
- ビューを更新する場合は、DEFAULT を指定できません。

merge_insert_clause

merge_insert_clause を指定すると、ON 句の条件が偽となる場合に対象となる表の列に挿入する値を指定できます。挿入が実行されると、対象となる表に定義されているすべての挿入トリガーがアクティブになります。INSERT キーワードの後に列リストを指定しない場合、対象となる表内の列数は、VALUES 句内の値の数と一致している必要があります。

すべてのソース行を表に挿入するには、ON 句の条件に**定数フィルタ条件**を使用します。定数フィルタ条件の一例は ON（0=1）です。Oracle Database はこのような条件を認識すると、すべてのソース行を無条件に表に挿入します。この方法は、*merge_update_clause* を省略することとは異なります。merge_update_clause を省略しても、結合は実行されます。定数フィルタ条件を設定すると、結合は実行されません。

where_clause を指定すると、指定した条件が真の場合のみに更新操作が実行されるようにできます。条件には、データ・ソース表のみを参照できます。条件が真ではないすべての行に対する挿入操作はスキップされます。

この句は、単独で、または *merge_update_clause* とともに指定できます。merge_insert_clause とともに指定する場合は、どちらを先に指定してもかまいません。

ビューへのマージの制限事項 : ビューを更新する場合は、DEFAULT を指定できません。

error_logging_clause

error_logging_clause の MERGE 文での動作は、INSERT 文の場合と同じです。詳細は、18-74 ページの INSERT 文の「error_logging_clause」を参照してください。

参照 : 「エラー・ロギングによる表への挿入例 :」（18-64 ページ）

例

表へのマージ例 : 次の例は、oe サンプル・スキーマ内の bonuses 表（ボーナスのデフォルトは 100）を使用します。次に、oe.orders 表の sales_rep_id 列に基づいて販売実績があったすべての従業員を、bonuses 表に挿入します。最終的に、人事部門マネージャが、給与が 8000 ドル以下の従業員にボーナスを支給することを決定します。販売実績がなかった従業員には、給与の 1% がボーナスとして支給されます。販売実績があった従業員には、給与の 1% がボーナスに加算されて支給されます。MERGE 文は、これらの変更を 1 行で実装します。

```
CREATE TABLE bonuses (employee_id NUMBER, bonus NUMBER DEFAULT 100);

INSERT INTO bonuses(employee_id)
   (SELECT e.employee_id FROM employees e, orders o
   WHERE e.employee_id = o.sales_rep_id
   GROUP BY e.employee_id);
```

```
SELECT * FROM bonuses ORDER BY employee_id;

EMPLOYEE_ID      BONUS
----------- ----------
        153        100
        154        100
        155        100
        156        100
        158        100
        159        100
        160        100
        161        100
        163        100

MERGE INTO bonuses D
   USING (SELECT employee_id, salary, department_id FROM employees
   WHERE department_id = 80) S
   ON (D.employee_id = S.employee_id)
   WHEN MATCHED THEN UPDATE SET D.bonus = D.bonus + S.salary*.01
     DELETE WHERE (S.salary > 8000)
   WHEN NOT MATCHED THEN INSERT (D.employee_id, D.bonus)
     VALUES (S.employee_id, S.salary*.01)
     WHERE (S.salary <= 8000);

SELECT * FROM bonuses ORDER BY employee_id;

EMPLOYEE_ID      BONUS
----------- ----------
        153        180
        154        175
        155        170
        159        180
        160        175
        161        170
        179        620
        173        610
        165        680
        166        640
        164        720
        172        730
        167        620
        171        740
```

NOAUDIT

用途

NOAUDIT 文を使用すると、AUDIT 文によって有効になった監査操作を停止できます。

NOAUDIT 文は先に発行した AUDIT 文と同じ構文である必要があります。また、NOAUDIT 文は、その特定の AUDIT 文のみを無効にします。たとえば、1 番目の AUDIT 文 A は特定のユーザーに対する監査を有効にするものとします。2 番目の文 B が、すべてのユーザーに対して監査を有効にします。すべてのユーザーに対して監査を無効にする NOAUDIT 文 C は、文 B を無効にします。ただし、文 A は無効にされず、文 A が指定したユーザーの監査は継続されます。

参照： 監査の詳細は、13-25 ページの「AUDIT」を参照してください。

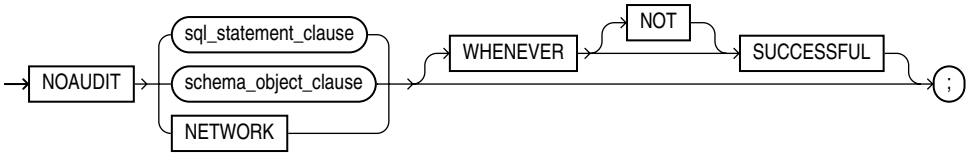
前提条件

SQL 文の監査を停止するには、AUDIT SYSTEM システム権限が必要です。

スキーマ・オブジェクトの監査を停止するには、監査を停止するオブジェクトが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、AUDIT ANY システム権限が必要です。監査の対象として選択するオブジェクトがディレクトリの場合、自分が作成したディレクトリであっても、AUDIT ANY システム権限が必要です。

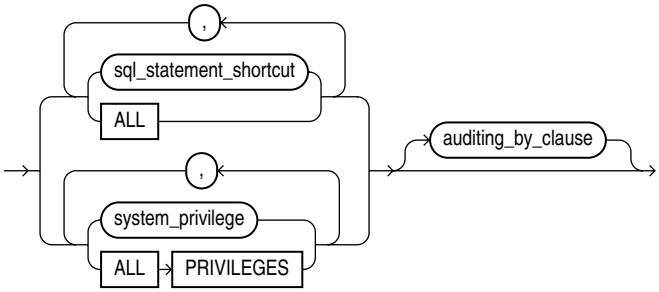
構文

***noaudit*::=**

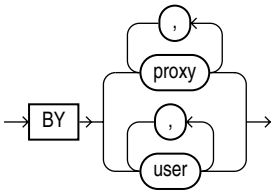


（18-76 ページの audit_operation_clause::=、18-77 ページの audit_schema_object_clause::= を参照）

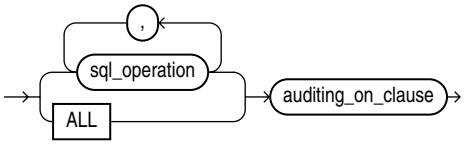
audit_operation_clause::=



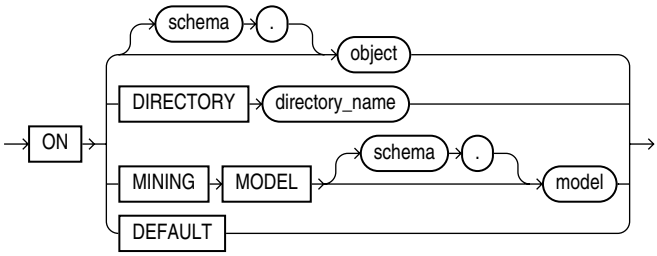
***auditing_by_clause*::=**



audit_schema_object_clause::=



***auditing_on_clause*::=**



セマンティクス

audit_operation_clause

audit_operation_clause を使用すると、特定の SQL 文の監査を停止できます。

statement_option *sql_statement_shortcut* では、監査を停止する SQL 文のショートカットを指定します。SQL 文のショートカットおよびショートカットによって監査される SQL 文の詳細は、13-30 ページの表 13-1 および 13-32 ページの表 13-2 を参照してください。

ALL ALL を指定すると、現在監査されているすべての文オプションの監査を停止できます。

system_privilege 監査を停止するシステム権限を指定します。システム権限および各システム権限によって許可される文については、18-37 ページの表 18-1 を参照してください。

ALL PRIVILEGES ALL PRIVILEGES を指定すると、現在監査されているすべてのシステム権限の監査を停止できます。

auditing_by_clause *auditing_by_clause* を使用すると、特定のユーザーが発行する SQL 文の監査のみを停止できます。この句を指定しない場合、すべてのユーザー文の監査が停止されます。

- 指定したユーザーのそれ以降のセッションで発行される SQL 文の監査のみを取り消す場合に、BY *user* を指定します。この句を指定しない場合、すべてのユーザー文の監査が停止されます。ただし、「WHENEVER SUCCESSFUL」で説明する状況は除きます。
- 特定のユーザーまたは任意のユーザーのかわりに指定されたプロキシによって発行された SQL 文の監査のみを停止する場合に、BY *proxy* を指定します。

audit_schema_object_clause

audit_schema_object_clause を使用すると、特定のデータベース・オブジェクトの監査を停止できます。

sql_operation *sql_operation* の場合、ON 句で指定したオブジェクトへの監査を停止する操作の種類を指定します。これらのオプションのリストは、13-33 ページの表 13-3 を参照してください。

ALL ALL をショートカットに指定することは、オブジェクト・タイプに適用できる SQL 操作をすべて指定することと同じです。

auditing_on_clause *auditing_on_clause* を使用すると、監査を停止する特定のスキーマ・オブジェクトを指定できます。

- オブジェクトには、表、ビュー、順序、ストアド・プロシージャ、ファンクション、パッケージ、マテリアライズド・ビューまたはライブラリのオブジェクト名を指定します。*object* を *schema* で修飾しなかった場合、そのオブジェクトは自分のスキーマ内にあるとみなされます。特定のスキーマ・オブジェクトの監査については、13-25 ページの「AUDIT」を参照してください。
- DIRECTORY では、監査を停止するディレクトリ名を指定できます。
- DEFAULT を指定して、オブジェクトを作成した後に、特定のオブジェクト・オプションをデフォルト・オブジェクト・オプションとして削除します。

NETWORK この句を使用すると、データベース・リンクの使用とログインの監査を停止できます。

WHENEVER [NOT] SUCCESSFUL WHENEVER SUCCESSFUL を指定すると、正常に実行されたスキーマ・オブジェクトに対する SQL 文および操作の監査のみを停止できます。

WHENEVER NOT SUCCESSFUL を指定すると、Oracle Database エラーとなった文および操作の監査のみが停止されます。

この句を指定しない場合、正常に実行されたかどうかにかかわらず、すべての文および操作の監査が停止されます。

例

ロールに関連する SQL 文の監査の停止例： 次の文は、ロールを作成または削除するすべての SQL 文の監査を停止します。

```
NOAUDIT ROLE;
```

特定ユーザーが所有するオブジェクトに対する更新または問合せの監視の停止例： 次の文は、ユーザー hr および oe によって発行された、表の問合せまたは更新を実行する文を監査している場合、hr の問合せの監査のみを停止します。

```
NOAUDIT SELECT TABLE BY hr;
```

この結果、ユーザー hr の問合せの監査のみが停止されます。oe の問合せと更新、および hr の更新の監査は継続されます。

特定のオブジェクト権限で許可された文の監査の停止例： 次の文は、DELETE ANY TABLE システム権限に許可されたすべての文の監査を停止します。

```
NOAUDIT DELETE ANY TABLE;
```

特定のオブジェクトに対する問合せの監査の停止例： 次の文は、スキーマ hr 内の employees 表に問い合せるすべての SQL 文の監査を選択していた場合に、この監査を停止します。

```
NOAUDIT SELECT
   ON hr.employees;
```

正常に実行される問合せの監査の停止例： 次の文は、正常に終了した問合せの監査を停止します。

```
NOAUDIT SELECT
   ON hr.employees
   WHENEVER SUCCESSFUL;
```

この文は、正常に終了した問合せの監査のみ停止します。Oracle Database エラーが発生した問合せの監査は継続されます。

PURGE

用途

PURGE 文を使用すると、ごみ箱内の表または索引を削除してそのオブジェクトに関連付けられていたすべての領域を解放するか、ごみ箱全体を空にするか、または削除された表領域の一部または全体をごみ箱から削除できます。

> **注意：** PURGE 文はロールバックできません。また、この文によって消去されたオブジェクトはリカバリできません。

ごみ箱の内容を参照するには、USER_RECYCLEBIN データ・ディクショナリ・ビューを問い合せます。かわりに RECYCLEBIN シノニムを使用することもできます。次の 2 つの文は、同じ行を戻します。

```
SELECT * FROM RECYCLEBIN;
SELECT * FROM USER_RECYCLEBIN;
```

参照：

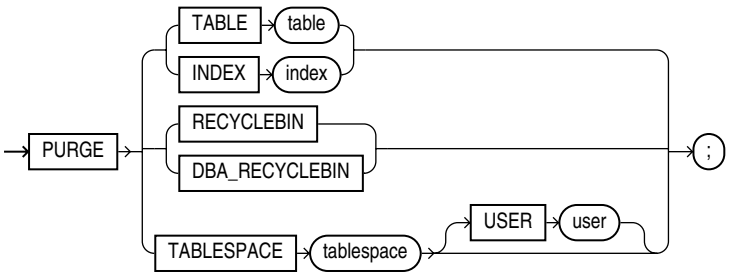
- ごみ箱の詳細、およびごみ箱内のオブジェクトのネーミング規則の詳細は、『Oracle Database 管理者ガイド』を参照してください。
- 削除した表をごみ箱から取り出す方法については、18-25 ページの「FLASHBACK TABLE」を参照してください。

前提条件

データベース・オブジェクトが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、消去するオブジェクトのタイプに対する DROP ANY... システム権限、または SYSDBA システム権限が必要です。

構文

***purge*::=**



セマンティクス

TABLE または INDEX

ごみ箱内の消去する表または索引の名前を指定します。ユーザーが指定した元の名前か、オブジェクトの削除時にそのオブジェクトに割り当てられたシステム生成名を指定できます。

- ユーザーが指定した名前を指定し、その名前を持つオブジェクトがごみ箱内に 1 つ以上存在した場合、ごみ箱内に最も長く存在しているオブジェクトが消去されます。
- ごみ箱内のオブジェクトのシステム生成名は一意です。そのため、システム生成名を指定すると、その名前を持つ特定のオブジェクトが消去されます。

表が消去されると、その表のすべての表パーティション、LOB と LOB パーティション、索引、およびその他の依存オブジェクトも消去されます。

RECYCLEBIN

この句を使用すると、現行のユーザーのごみ箱を空にできます。そのユーザーのごみ箱からすべてのオブジェクトが消去され、そのオブジェクトに関連付けられていたすべての領域が解放されます。

DBA_RECYCLEBIN

この句は、SYSDBA システム権限を持っている場合にのみ有効です。この句を使用すると、システム全体のごみ箱からすべてのオブジェクトを削除できます。これは、各ユーザーのごみ箱を空にすることと同じです。この操作は、以前のリリースへの移行などの場合に役立ちます。

TABLESPACE *tablespace*

この句を使用すると、ごみ箱から、指定した領域内に存在するすべてのオブジェクトを消去できます。

USER *user* この句を使用すると、指定したユーザーが、表領域内の領域を再利用できます。この操作は、特定のユーザーの、特定の表領域に対するディスク領域が割当て制限間近である場合に特に役立ちます。

例

ごみ箱からのファイルの削除例 : 次の文は、ごみ箱から test 表を削除します。複数のバージョンの test 表がごみ箱内に存在する場合、Oracle Database では、ごみ箱内に最も長く存在しているものが削除されます。

```
PURGE TABLE test;
```

ごみ箱から削除する表のシステム生成名を判断するには、ごみ箱に対する SELECT 文を発行します。そのオブジェクト名を使用して、次の文に類似した文を発行して表を削除できます。(システム生成名は、次の例に示すものとは異なります。)

```
PURGE TABLE RB$$33750$TABLE$0;
```

ごみ箱の内容の削除例 : 次の文は、ごみ箱のすべての内容を削除します。

```
PURGE RECYCLEBIN;
```

RENAME

用途

注意： RENAME 文はロールバックできません。

RENAME 文を使用すると、表、ビュー、順序またはプライベート・シノニムの名前を変更できます。

- 古いオブジェクトの整合性制約、索引および権限付与は、新しいオブジェクトに自動的に移行されます。
- 名前を変更した表を参照するビュー、シノニム、ストアド・プロシージャ、ストアド・ファンクションなど、名前を変更したオブジェクトに依存するオブジェクトはすべて無効になります。

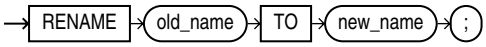
参照： 16-2 ページの「CREATE SYNONYM」および 18-3 ページの「DROP SYNONYM」を参照してください。

前提条件

オブジェクトが自分のスキーマ内にある必要があります。

構文

***rename*::=**



セマンティクス

old_name

既存の表、ビュー、順序またはプライベート・シノニムの名前を指定します。

new_name

既存のオブジェクトに指定する新しい名前を指定します。新しい名前は、同じネームスペース内の他のスキーマ・オブジェクトに使用されている名前以外を指定する必要があります。また、スキーマ・オブジェクトのネーミング規則に従って指定する必要があります。

オブジェクトの名前変更の制限事項： オブジェクト名の変更には、次の制限事項があります。

- パブリック・シノニムの名前は変更できません。そのかわり、該当するパブリック・シノニムを削除し、新しい名前で別のパブリック・シノニムを作成してください。
- 依存表または依存する有効なユーザー定義オブジェクト型を持つ型のシノニムの名前は変更できません。

参照： 「スキーマ・オブジェクトのネーミング規則」(2-98 ページ)

例

データベース・オブジェクトの名前の変更例： 次の例では、サンプル表 hr.departments のコピーを使用します。次の文は、表の名前を departments_new から emp_departments に変更します。

```
RENAME departments_new TO emp_departments;
```

この文では列名を直接変更できません。ただし、ALTER TABLE ... *rename_column_clause* を使用すると、列の名前を変更できます。

> **参照：** 「rename_column_clause」（12-48 ページ）

列名を変更するもう 1 つの方法は、AS 副問合せを指定した CREATE TABLE 文とともに、RENAME 文を使用する方法です。この方法は、単に列の名前を変更するのではなく、表の構造を変更する場合に有効です。次の文は、サンプル表 hr.job_history を再作成し、列の名前を department_id から dept_id に変更します。

```
CREATE TABLE temporary
   (employee_id, start_date, end_date, job_id, dept_id)
AS SELECT
     employee_id, start_date, end_date, job_id, department_id
FROM job_history;

DROP TABLE job_history;

RENAME temporary TO job_history;
```

前述の例の場合、job_history 表に定義されている整合性制約は失われます。これらの整合性制約は、ALTER TABLE 文を使用して、新しい job_history 表に再定義する必要があります。

REVOKE

用途

REVOKE 文を使用すると、次の操作を実行できます。

- ユーザーおよびロールからのシステム権限の取消し
- ユーザーおよびロールからのロールの取消し
- ユーザーまたはロールからの特定のオブジェクトに対するオブジェクト権限の取消し

自動ストレージ管理の注意事項： AS SYSASM と認証されたユーザーは、この文を使用すると、システム権限 SYSASM、SYSOPER および SYSDBA を、現行ノードの自動ストレージ管理パスワード・ファイルのユーザーから取り消すことができます。

参照：

- システム権限およびロールの付与については、18-31 ページの「GRANT」を参照してください。
- それぞれのオブジェクト型に対するオブジェクト権限の詳細は、18-47 ページの表 18-2 を参照してください。

前提条件

システム権限を取り消すには、Admin Option 付きの権限が必要です。

ロールを取り消すには、Admin Option 付きのロールが必要です。なお、GRANT ANY ROLE システム権限を持っている場合は、ロールを自由に取り消すことができます。

オブジェクト権限を取り消すには、以前にユーザーとロールにオブジェクト権限を付与している、または GRANT ANY OBJECT PRIVILEGE システム権限を持っている必要があります。後者の場合は、オブジェクト所有者によって付与されたかまたは所有者の役割を持つユーザー（GRANT ANY OBJECT PRIVILEGE を持つユーザー）によって付与されたオブジェクト権限を取り消すことができます。ただし、With Grant Option によって付与されたオブジェクト権限を取り消すことはできません。

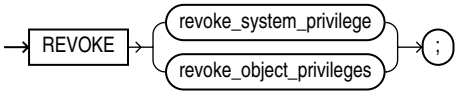
参照： 「GRANT ANY OBJECT PRIVILEGE を使用する操作を取り消す例:」（18-91 ページ）

REVOKE 文によって取り消すことができる権限およびロールは、GRANT 文によって直接付与されているものにかぎられます。この句では、次の権限を取り消すことはできません。

- 取消し側に付与されていない権限またはロール
- オペレーティング・システムを介して付与されているロールまたはオブジェクト権限
- ロールを介して取消し側に付与されている権限またはロール

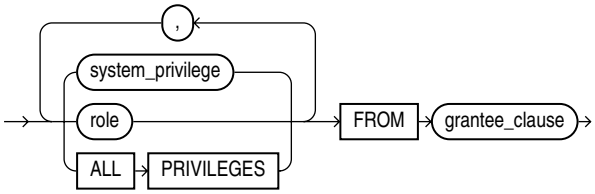
構文

***revoke*::=**



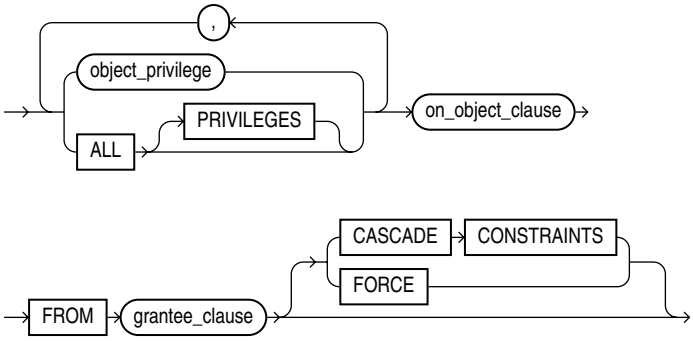
（18-85 ページの revoke_system_privileges::=、18-85 ページの revoke_object_privileges::= を参照）

revoke_system_privileges::=



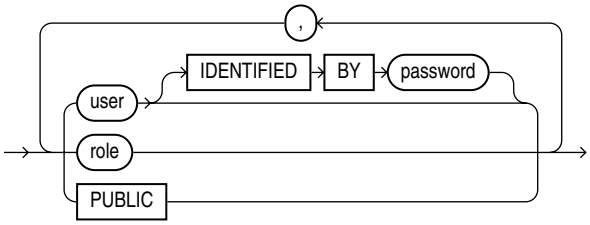
（18-85 ページの grantee_clause::= を参照）

revoke_object_privileges::=

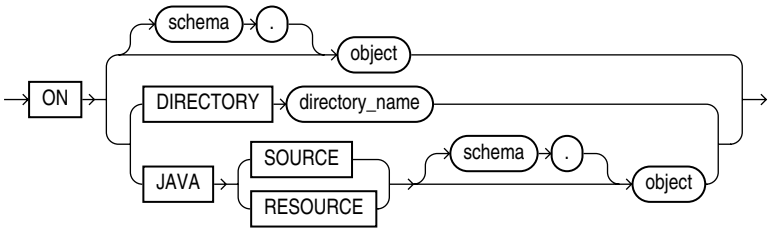


（18-85 ページの on_object_clause::=、18-85 ページの grantee_clause::= を参照）

grantee_clause::=



on_object_clause::=



セマンティクス

revoke_system_privileges

システム権限を取り消すには、次の句を使用します。

system_privilege

取り消すシステム権限を指定します。システム権限のリストは、18-37 ページの表 18-1 を参照してください。

ユーザーのシステム権限を取り消す場合、ユーザーの権限ドメインからその権限が取り消されます。この取消しはすぐに有効になるため、このユーザーはその権限を使用できなくなります。

ロールのシステム権限を取り消す場合、ロールの権限ドメインからその権限が取り消されます。この取消しはすぐに有効になるため、そのロールが使用可能となっている場合でも、この権限は使用できません。また、そのロールが付与されている他のユーザーは、ロールを使用可能にしても、その権限を使用できなくなります。

> **参照：** 18-89 ページの「ユーザーからシステム権限を取り消す例 :」および 18-89 ページの「ロールからシステム権限を取り消す例 :」を参照してください。

PUBLIC のシステム権限を取り消す場合、PUBLIC を介して権限を付与されている各ユーザーの権限ドメインからその権限が取り消されます。この取消しはすぐに有効になるため、ユーザーは権限を使用できなくなります。ただし、直接またはロールを介して権限が付与されているユーザーからは、権限を取り消すことはできません。

Oracle Database には、すべてのシステム権限を一度に指定できるショートカットがあります。ALL PRIVILEGES を指定すると、18-37 ページの表 18-1 に示すすべてのシステム権限を取り消すことができます。

システム権限の取消しの制限事項： 取り消す権限のリストに権限を指定できるのは 1 回のみです。

role

取り消すロールを指定します。

ユーザーからロールを取り消すと、ユーザーによるロールの使用が禁止されます。そのロールが使用可能となっている場合に、ロールの権限ドメインの権限が使用可能な場合はその権限を使用できます。ただし、ユーザーが後からロールを使用可能にできません。

他の**ロール**からロールを取り消すと、取消し側ロールの権限ドメインから、取り消されたロールの権限ドメインが削除されます。取り消されたロールの権限を付与され、そのロールの権限が使用可能になっているユーザーは、そのロールの権限が使用可能な間は、取り消されたロールの権限ドメインの権限を引き続き使用できます。ただし、取り消された権限を付与されていても、ロールの取消し操作の後でそれを使用可能にしたユーザーは、取り消されたロールの権限ドメインの権限を使用できません。

> **参照：** 18-89 ページの「ユーザーからロールを取り消す例 :」および 18-89 ページの「ロールからロールを取り消す例 :」を参照してください。

PUBLIC のロールを取り消すと、PUBLIC を介してロールが付与されているすべてのユーザーに対して、そのロールが使用禁止にされます。そのロールを使用可能としているユーザーは、権限ドメインの権限が使用可能であるかぎり、権限ドメインでその権限を引き続き使用できます。ただし、ユーザーが後からロールを使用可能にすることはできません。直接またはロールを介して権限が付与されているユーザーからは、ロールを取り消すことはできません。

システム・ロールの取消しの制限事項： 取り消すロールのリストにシステム・ロールを指定できるのは 1 回のみです。事前定義されているロールの詳細は、『Oracle Database セキュリティ・ガイド』を参照してください。

grantee_clause

FROM *grantee_clause* を指定すると、システム権限、ロールまたはオブジェクト権限が取り消されるユーザーまたはロールを識別できます。

PUBLIC　PUBLIC を指定すると、すべてのユーザーから権限を取り消すことができます。

revoke_object_privileges

オブジェクト権限を取り消すには、次の句を使用します。

object_privilege

取り消すオブジェクト権限を指定します。18-47 ページの表 18-2 に示すどのオブジェクト権限でも指定できます。

注意： 操作は権限によって許可されます。権限を取り消すと、取り消されたユーザーは、その操作を実行できなくなります。ただし、複数のユーザーが、同じ権限を同一ユーザー、ロールまたは PUBLIC に対して付与している場合があります。権限受領者の権限ドメインの権限を取り消す場合、すべての権限付与者が権限を取り消す必要があります。権限を取り消さない権限付与者が 1 人でもいれば、権限受領者は引き続きその権限を使用できます。

ユーザーのオブジェクト権限を取り消すと、ユーザーの権限ドメインからその権限が取り消されます。この取消しはすぐに有効になるため、このユーザーはその権限を使用できなくなります。

- ユーザーがその権限を他のユーザーまたはロールに付与している場合、他のユーザーまたはロールの権限も取り消されます。
- 権限を使用する SQL 文を記述したプロシージャ、ファンクションまたはパッケージが定義されているスキーマを持つユーザーのオブジェクト権限を取り消すと、それらのプロシージャ、ファンクションまたはパッケージは実行できなくなります。
- そのユーザーのスキーマにオブジェクトのビューが含まれる場合、ビューは無効になります。
- 参照整合性制約の定義権限を使用したユーザーの REFERENCES オブジェクト権限を取り消す場合、CASCADE CONSTRAINTS 句も指定する必要があります。

ロールのオブジェクト権限を取り消すと、ロールの権限ドメインからその権限が取り消されます。この取消しはすぐに有効になるため、そのロールが使用可能となっている場合でも、この権限は使用できません。ロールが付与されている他のユーザーは、ロールを使用可能にした場合でも、権限を使用できなくなります。

PUBLIC のオブジェクト権限を取り消すと、PUBLIC を介して権限を付与されている各ユーザーの権限ドメインからその権限が取り消されます。この取消しはすぐに有効になるため、それらのすべてのユーザーは、その権限を使用できなくなります。ただし、直接またはロールを介して権限が付与されているユーザーからは、権限を取り消すことはできません。

ALL [PRIVILEGES]

ALL を指定すると、ユーザーまたはロールに付与されているすべてのオブジェクト権限を取り消すことができます（キーワード PRIVILEGES はセマンティクスを明確にするためのものであり、指定は任意です。）

オブジェクトに権限が付与されていない場合、処理は行われず、エラーも戻りません。

オブジェクト権限の取消しの制限事項： 取り消す権限のリストに権限を指定できるのは 1 回のみです。FROM 句にユーザー、ロールまたは PUBLIC を指定できるのは 1 回のみです。

> **参照：** 18-89 ページの「ユーザーからオブジェクト権限を取り消す例 :」、18-89 ページの「PUBLIC からオブジェクト権限を取り消す例 :」および 18-89 ページの「ユーザーからすべてのオブジェクト権限を取り消す例 :」を参照してください。

CASCADE CONSTRAINTS

この句は、REFERENCES 権限または ALL [PRIVILEGES] を取り消すときにのみ適用されます。取消し側で REFERENCES 権限（ALL [PRIVILEGES] を付与して明示的または暗黙的に付与された権限）を使用して定義した参照整合性制約を削除します。

> **参照：** 「CASCADE CONSTRAINTS でオブジェクト権限を取り消す例 :」（18-90 ページ）

FORCE

FORCE を指定すると、表または型に依存するユーザー定義型オブジェクトで、EXECUTE オブジェクト権限を取り消すことができます。表に依存するユーザー定義型オブジェクトでは、FORCE を使用して EXECUTE オブジェクト権限を取り消します。

FORCE を指定した場合、すべての権限が取り消されますが、すべての依存するオブジェクトには INVALID のマークが付けられ、依存表のデータにはアクセスできなくなります。また、すべての依存するファンクション索引には、UNUSABLE のマークが付けられます。必要な型の権限を再付与した場合、表に対して再度妥当性チェックが行われます。

> **参照：** 型依存性およびユーザー定義オブジェクト権限の詳細は、『Oracle Database 概要』を参照してください。

on_object_clause

on_object_clause を指定すると、権限を取り消すオブジェクトを識別できます。

object オブジェクト権限を取り消すオブジェクトを指定します。取り消すことができるオブジェクトは次のとおりです。

- 表、ビュー、順序、プロシージャ、ストアド・ファンクション、パッケージまたはマテリアライズド・ビュー
- 表、ビュー、順序、プロシージャ、ストアド・ファンクション、パッケージ、マテリアライズド・ビューまたはユーザー定義型に対するシノニム
- ライブラリ、索引タイプまたはユーザー定義演算子

オブジェクトを *schema* で修飾しなかった場合、そのオブジェクトは自分のスキーマ内にあるとみなされます。

> **参照：** 「ユーザーから順序のオブジェクト権限を取り消す例 :」（18-90 ページ）

システム権限の Grant Option の有無にかかわらず、SELECT オブジェクト権限をマテリアライズド・ビューまたは表を含むマテリアライズド・ビューから取り消すと、そのマテリアライズド・ビューは無効になります。

システム権限の GRANT OPTION の有無にかかわらず、マテリアライズド・ビューのマスター表のいずれかにおける SELECT オブジェクト権限を取り消すと、そのマテリアライズド・ビューとそれに含まれる表の両方またはマテリアライズド・ビューが無効になります。

DIRECTORY *directory_name* 権限を取り消すディレクトリ・オブジェクトを指定します。*directory_name* は *schema* で修飾できません。このオブジェクトはディレクトリである必要があります。

> **参照：** 14-38 ページの「CREATE DIRECTORY」および 18-90 ページの「ユーザーからディレクトリへのオブジェクト権限を取り消す例 :」を参照してください。

JAVA SOURCE | RESOURCE JAVA 句を使用すると、権限を取り消す Java ソースまたはリソース・スキーマ・オブジェクトを指定できます。

例

ユーザーからシステム権限を取り消す例： 次の文は、ユーザー hr および oe の DROP ANY TABLE システム権限を取り消します。

```
REVOKE DROP ANY TABLE
    FROM hr, oe;
```

この結果、hr および oe は他のユーザーのスキーマに定義されている表を削除できなくなります。

ユーザーからロールを取り消す例： 次の文は、ユーザー sh のロール dw_manager を取り消します。

```
REVOKE dw_manager
    FROM sh;
```

この結果、sh ユーザーは dw_manager ロールを使用可能にできなくなります。

ロールからシステム権限を取り消す例： 次の文は、ロール dw_manager の CREATE TABLESPACE システム権限を取り消します。

```
REVOKE CREATE TABLESPACE
   FROM dw_manager;
```

ロール dw_manager を使用可能にしても、ユーザーは表領域を作成できません。

ロールからロールを取り消す例： 次の文は、ロール dw_manager からロール dw_user を取り消します。

```
REVOKE dw_user
  FROM dw_manager;
```

この結果、dw_user ロールの権限は dw_manager に付与されなくなります。

ユーザーからオブジェクト権限を取り消す例： 次の文は、orders 表に対する DELETE、INSERT、SELECT および UPDATE 権限をユーザー hr に付与します。

```
GRANT ALL
   ON orders TO hr;
```

次の文は、ユーザー hr から表 orders に対する DELETE 権限を取り消します。

```
REVOKE DELETE
   ON orders FROM hr;
```

ユーザーからすべてのオブジェクト権限を取り消す例： 次の文は、ユーザー hr の表 orders に対する残りのすべての権限を取り消します。

```
REVOKE ALL
   ON orders FROM hr;
```

PUBLIC からオブジェクト権限を取り消す例： 次の文は、ロール public に権限を付与することによって、すべてのユーザーにビュー emp_details_view に対する SELECT 権限および UPDATE 権限を付与します。

```
GRANT SELECT, UPDATE
    ON emp_details_view TO public;
```

次の文は、すべてのユーザーから emp_details_view に対する UPDATE 権限を取り消します。

```
REVOKE UPDATE
    ON emp_details_view FROM public;
```

ユーザーは、emp_details_view ビューへの問合せはできますが、更新はできなくなります。ただし、emp_details_view に対する UPDATE 権限も任意のユーザーに直接またはロールを介して付与している場合は、これらのユーザーはその権限を保持します。

ユーザーから順序のオブジェクト権限を取り消す例： 次の文は、ユーザー oe にスキーマ hr 内の departments_seq 順序に対する SELECT 権限を付与します。

```
GRANT SELECT
    ON hr.departments_seq TO oe;
```

次の文は、oe から departments_seq に対する SELECT 権限を取り消します。

```
REVOKE SELECT
    ON hr.departments_seq FROM oe;
```

ただし、ユーザー hr が departments に対する SELECT 権限を sh に付与している場合、sh は、hr からの権限付与によって departments を使用できます。

CASCADE CONSTRAINTS でオブジェクト権限を取り消す例： 次の文は、oe に、スキーマ hr 内の employees 表に対する REFERENCES 権限および UPDATE 権限を付与します。

```
GRANT REFERENCES, UPDATE
    ON hr.employees TO oe;
```

ユーザー oe は、REFERENCES 権限を使用して、hr スキーマ内の employees 表を参照する dependent 表の制約を定義できます。

```
CREATE TABLE dependent
(dependno   NUMBER,
 dependname VARCHAR2(10),
 employee   NUMBER
    CONSTRAINT in_emp REFERENCES hr.employees(employee_id) );
```

CASCADE CONSTRAINTS 句を指定した次の文を発行することによって、oe の hr.employees に対する REFERENCES 権限を取り消すことができます。

```
REVOKE REFERENCES
    ON hr.employees
    FROM oe
    CASCADE CONSTRAINTS;
```

oe の hr.employees に対する REFERENCES 権限を取り消した場合、oe は制約を定義する権限が必要になるため、in_emp 制約が自動的に削除されます。

ただし、oe が他のユーザーから hr.employees に対する REFERENCES 権限を付与されている場合は、その制約は削除されません。他のユーザーから権限付与されたため、oe は制約に対して必要な権限を保持しています。

ユーザーからディレクトリへのオブジェクト権限を取り消す例： 次の文は、hr の bfile_dir ディレクトリに対する READ オブジェクト権限を取り消します。

```
REVOKE READ ON DIRECTORY bfile_dir FROM hr;
```

GRANT ANY OBJECT PRIVILEGE を使用する操作を取り消す例 : データベース管理者によってユーザー sh に GRANT ANY OBJECT PRIVILEGE が付与されているとします。ユーザー hr が、employees 表に対する UPDATE 権限をユーザー oe に付与します。

```
CONNECT hr
GRANT UPDATE ON employees TO oe WITH GRANT OPTION;
```

これによって、オブジェクト権限を別のユーザーに付与する権限がユーザー oe に付与されます。

```
CONNECT oe
GRANT UPDATE ON hr.employees TO pm;
```

oe には hr によって権限が与えられたため、GRANT ANY OBJECT PRIVILEGE が付与されているユーザー sh は、ユーザー hr にかわってユーザー oe の UPDATE 権限を取り消すことができます。

```
CONNECT sh
REVOKE UPDATE ON hr.employees FROM oe;
```

pm に権限を付与したのがオブジェクトの所有者（hr）、ユーザー sh または GRANT ANY OBJECT PRIVILEGE を持つ他のユーザーではなく、ユーザー oe であったため、ユーザー sh は、ユーザー pm の UPDATE 権限を明示的に取り消すことはできません。ただし、前述の文は、連鎖的な取消しを行い、取り消された権限に依存するすべての権限を取り消します。そのため、pm のオブジェクト権限も暗黙的に取り消されます。

ROLLBACK

用途

ROLLBACK 文を使用すると、現行のトランザクションで実行された処理を取り消すことができます。また、インダウト分散トランザクションで実行された処理を手動で取り消すこともできます。

注意： アプリケーション・プログラムでは、COMMIT または ROLLBACK 文を使用してトランザクションを明示的に終了することをお薦めします。トランザクションを明示的にコミットせずにプログラムが異常終了した場合、コミットされていない最後のトランザクションがロールバックされます。

参照：

- トランザクションの詳細は、『Oracle Database 概要』を参照してください。
- 分散トランザクションの詳細は、『Oracle Database Heterogeneous Connectivity 管理者ガイド』を参照してください。
- 現行トランザクションの特性の設定については、19-53 ページの「SET TRANSACTION」を参照してください。
- 13-44 ページの「COMMIT」および 19-2 ページの「SAVEPOINT」を参照してください。

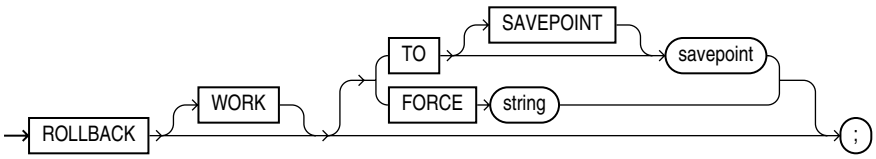
前提条件

現行のトランザクションをロールバックする場合、権限は不要です。

コミットしたインダウト分散トランザクションを手動でロールバックする場合は、FORCE TRANSACTION システム権限が必要です。他のユーザーがコミットしたインダウト分散トランザクションを手動でロールバックする場合は、FORCE ANY TRANSACTION システム権限が必要です。

構文

***rollback*::=**



セマンティクス

WORK

WORK キーワードの指定は任意です。このキーワードは、SQL 規格との互換性のために提供されています。

TO SAVEPOINT 句

現行のトランザクションをロールバックするセーブポイントを指定します。この句を指定しない場合、ROLLBACK 文によってトランザクション全体がロールバックされます。

TO SAVEPOINT 句を指定しないで ROLLBACK コマンドを実行すると、次の処理が行われます。

- トランザクションの終了
- 現行のトランザクションに対するすべての変更の取消し
- トランザクション中のセーブポイントの消去
- すべてのトランザクション・ロックの解除

参照： 「SAVEPOINT」（19-2 ページ）

TO SAVEPOINT 句を指定して ROLLBACK コマンドを実行すると、次の処理が行われます。

- 指定したセーブポイント後のトランザクションへのロールバック。トランザクションは終了しません。
- 指定したセーブポイントより後に作成されたセーブポイントの消去。指定したセーブポイントはそのまま残るため、そのセーブポイントまで繰り返しロールバックできます。指定したセーブポイントより前に作成されたセーブポイントも残ります。
- 指定したセーブポイント以降に設定された表と行のロックの解除。セーブポイント後にロックされた行へのアクセスを要求した他のトランザクションは、コミットまたはロールバックされるまで待つ必要があります。行を要求していない他のトランザクションは、すぐに行の要求およびアクセスができます。

インダウト・トランザクションの制限事項： インダウト・トランザクションは、セーブポイントまで手動でロールバックできません。

FORCE 句

FORCE を指定すると、インダウト分散トランザクションを手動でロールバックできます。このトランザクションは、ローカル・トランザクション ID またはグローバル・トランザクション ID を含む *string* で識別されます。このトランザクションの ID を確認する場合は、データ・ディクショナリ・ビュー DBA_2PC_PENDING を問い合せます。

FORCE 句を指定した ROLLBACK 文では、指定したトランザクションのみがロールバックされます。この文は、現行のトランザクションには影響しません。

参照： 分散トランザクションおよびインダウト・トランザクションのロールバックの詳細は、『Oracle Database 管理者ガイド』を参照してください。

例

ロールバック・トランザクションの例： 次の文は、現行のトランザクション全体をロールバックします。

```
ROLLBACK;
```

次の文は、現行のトランザクションをセーブポイント banda_sal にロールバックします。

```
ROLLBACK TO SAVEPOINT banda_sal;
```

前述の例の詳細は、19-2 ページの「セーブポイントの作成例：」を参照してください。

次の文は、インダウト分散トランザクションを手動でロールバックします。

```
ROLLBACK WORK
    FORCE '25.32.87';
```

19

SQL 文 : SAVEPOINT ～ UPDATE

この章では、次の SQL 文について説明します。

- SAVEPOINT
- SELECT
- SET CONSTRAINT[S]
- SET ROLE
- SET TRANSACTION
- TRUNCATE CLUSTER
- TRUNCATE TABLE
- UPDATE

SAVEPOINT

用途

SAVEPOINT 文を使用すると、後でロールバックできるシステム変更番号（SCN）の名前を作成できます。

参照：

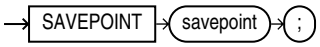
- セーブポイントの詳細は、『Oracle Database 概要』を参照してください。
- トランザクションのロールバックの詳細は、18-92 ページの「ROLLBACK」を参照してください。
- 現行トランザクションの特性の設定については、19-53 ページの「SET TRANSACTION」を参照してください。

前提条件

ありません。

構文

***savepoint*::=**



セマンティクス

savepoint

作成するセーブポイントの名前を指定します。

同一トランザクション内のセーブポイント名は、区別する必要があります。同じ識別子のセーブポイントを作成した場合、最初のセーブポイントは消去されます。セーブポイントを作成した後は、処理の継続、作業のコミット、トランザクション全体のロールバックまたはセーブポイントまでのロールバックを実行できます。

例

セーブポイントの作成例： 次の文は、サンプル表 hr.employees の Banda と Greene の給与を更新するために、部門の給与合計が 314,000 ドルを超えていないことを確認してから、Greene の給与を再入力します。

```
UPDATE employees
    SET salary = 7000
    WHERE last_name = 'Banda';
SAVEPOINT banda_sal;

UPDATE employees
    SET salary = 12000
    WHERE last_name = 'Greene';
SAVEPOINT greene_sal;

SELECT SUM(salary) FROM employees;

ROLLBACK TO SAVEPOINT banda_sal;
```

```
UPDATE employees
    SET salary = 11000
    WHERE last_name = 'Greene';

COMMIT;
```

SELECT

用途

SELECT 文または副問合せを使用すると、1 つ以上の表、オブジェクト表、ビュー、オブジェクト・ビューまたはマテリアライズド・ビューからデータを取り出すことができます。

SELECT 文の結果（またはその一部）が既存のマテリアライズド・ビューと同じ場合、そのマテリアライズド・ビューを SELECT 文で指定した 1 つ以上の表のかわりに使用できます。このような置換を**クエリー・リライト**といいます。これは、コストの最適化が有効で、QUERY_REWRITE_ENABLED パラメータが TRUE に設定されている場合にのみ行われます。クエリー・リライトが行われるかどうかを確認する場合は、EXPLAIN PLAN 文を使用してください。

参照：

- 問合せおよび副問合せの概要は、第 9 章「SQL 問合せおよび副問合せ」を参照してください。
- マテリアライズド・ビューおよびクエリー・リライトの詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。
- 「EXPLAIN PLAN」（18-19 ページ）

前提条件

表またはマテリアライズド・ビューからデータを選択する場合、表またはマテリアライズド・ビューが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、その表またはマテリアライズド・ビューに対する SELECT 権限が必要です。

ビューの実表から行を選択する場合、次の条件を 2 つとも満たしている必要があります。

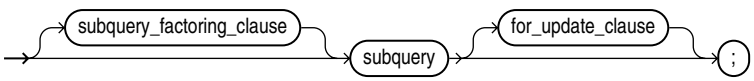
- ビューに対する SELECT 権限を持っている。
- ビューを含むスキーマの所有者が、実表に対する SELECT 権限を持っている。

SELECT ANY TABLE システム権限を持っている場合、任意の表、マテリアライズド・ビューまたはビューの実表からデータを選択できます。

flashback_query_clause を使用して Oracle フラッシュバック問合せを発行する場合は、SELECT 構文のリスト内のオブジェクトに対する SELECT 権限が必要です。さらに、SELECT 構文のリスト内のオブジェクトに対する FLASHBACK オブジェクト権限または FLASHBACK ANY TABLE システム権限のいずれかが必要です。

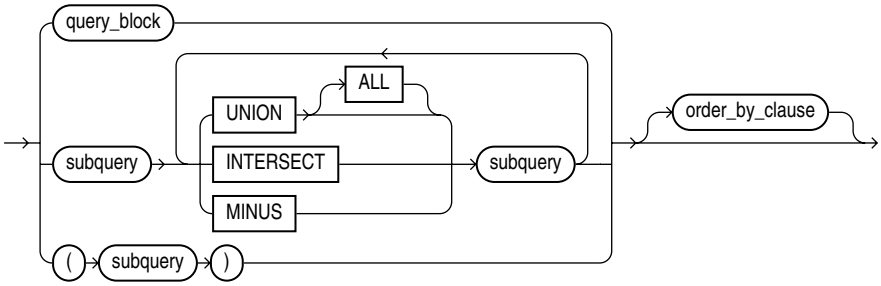
構文

***select*::=**



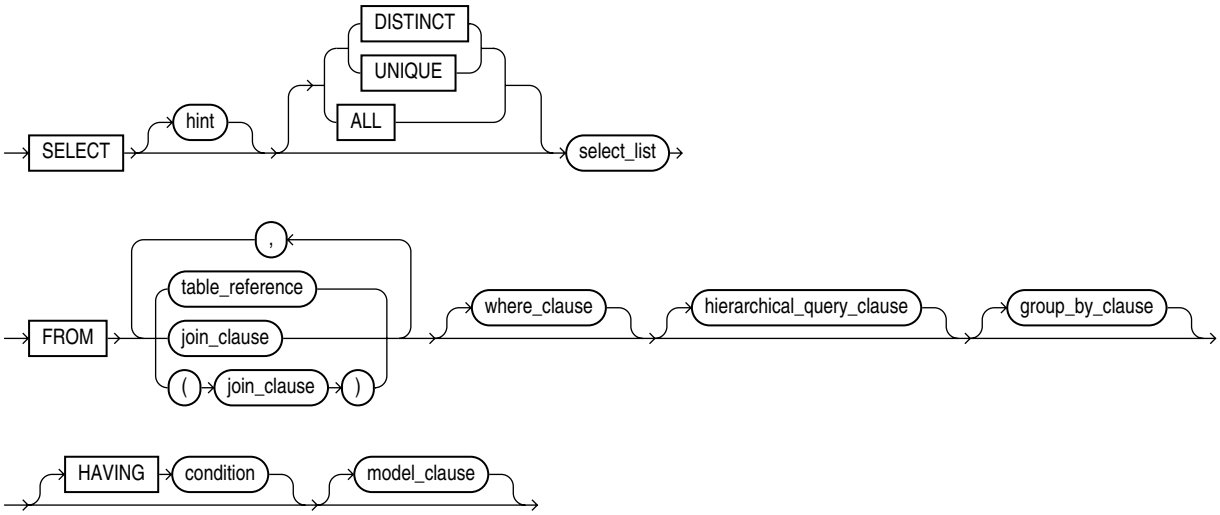
（19-5 ページの subquery_factoring_clause::=、19-12 ページの for_update_clause::= を参照）

***subquery*::=**



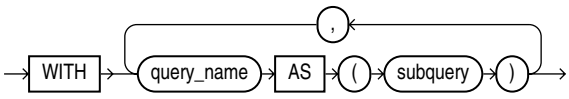
（19-5 ページの query_block::=、19-12 ページの order_by_clause::= を参照）

***query_block*::=**

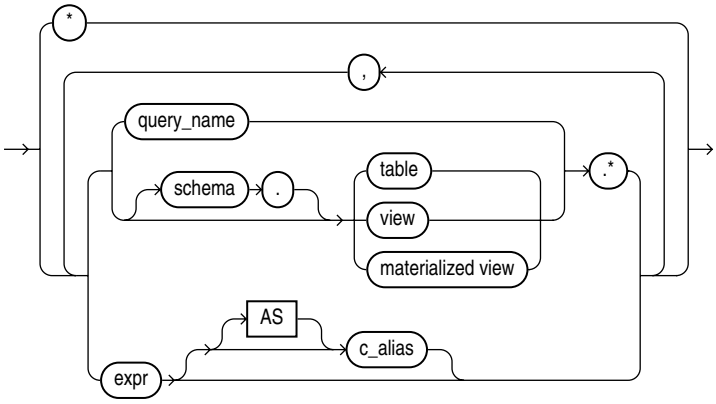


（19-6 ページの select_list::=、19-6 ページの table_reference::=、19-8 ページの join_clause::=、
19-9 ページの where_clause::=、19-9 ページの hierarchical_query_clause::=、
19-9 ページの group_by_clause::=、19-10 ページの model_clause::= を参照）

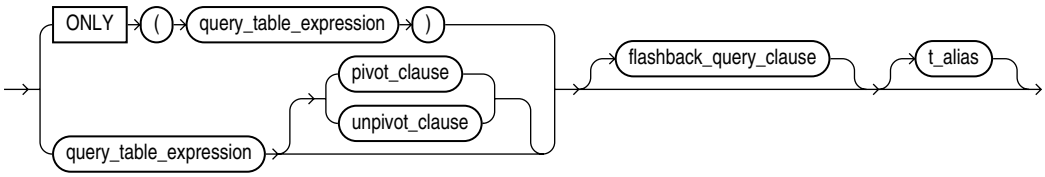
subquery_factoring_clause::=



select_list::=

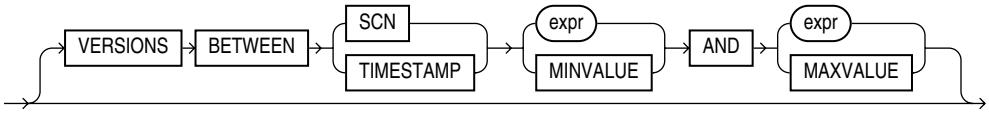


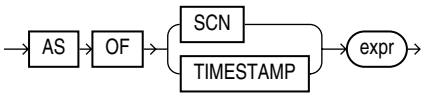
***table_reference*::=**



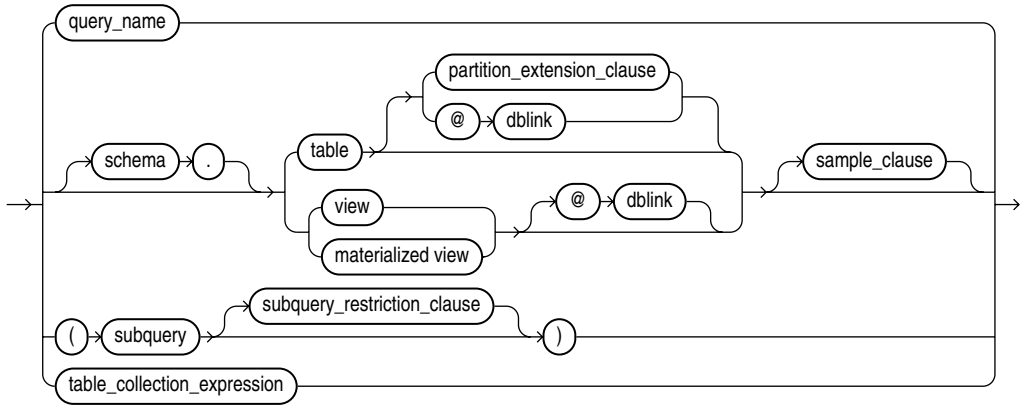
(19-6 ページの query_table_expression::=、19-6 ページの flashback_query_clause::=、19-7 ページの pivot_clause::=、19-7 ページの unpivot_clause::= を参照)

flashback_query_clause::=



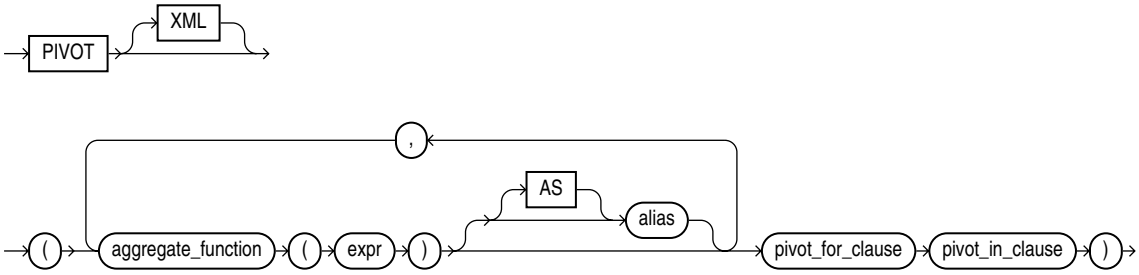


query_table_expression::=

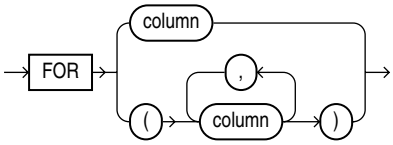


(19-8 ページの subquery_restriction_clause::=、19-8 ページの table_collection_expression::= を参照)

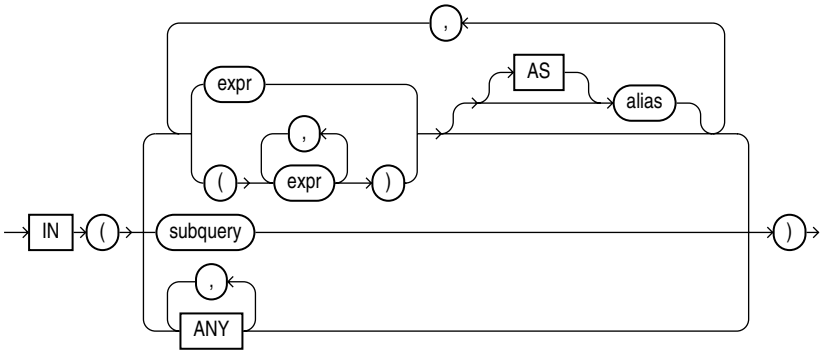
pivot_clause::=



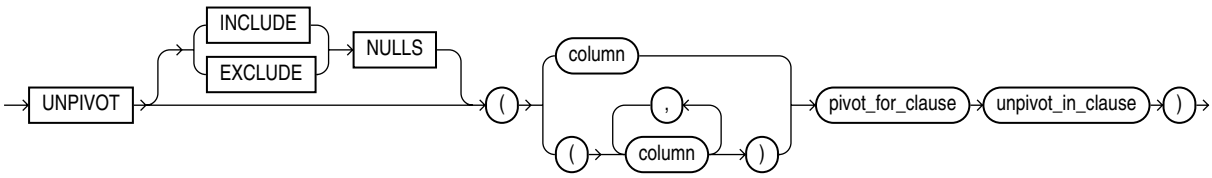
***pivot_for_clause*::=**



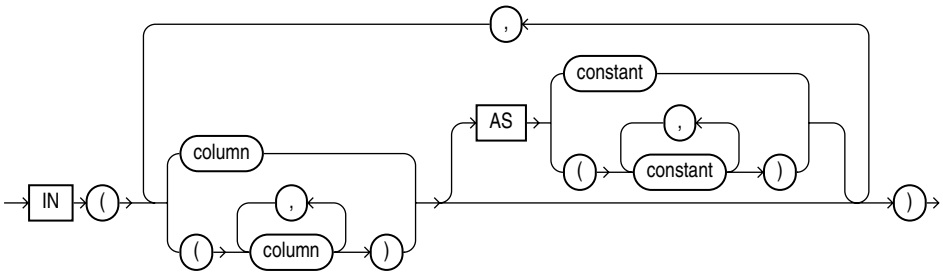
***pivot_in_clause*::=**



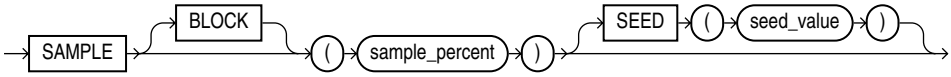
unpivot_clause::=



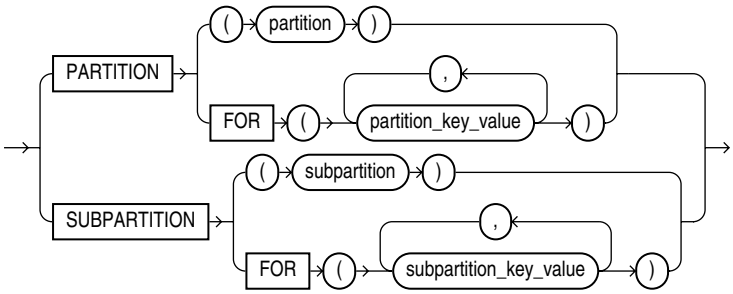
***unpivot_in_clause*::=**



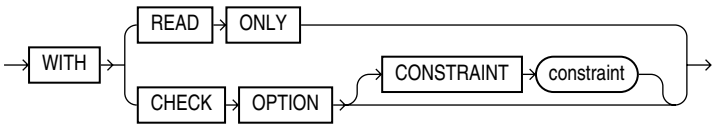
sample_clause::=



***partition_extension_clause*::=**

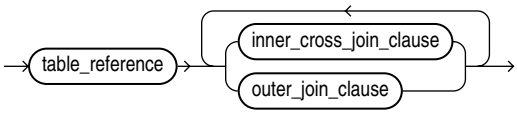


subquery_restriction_clause::=

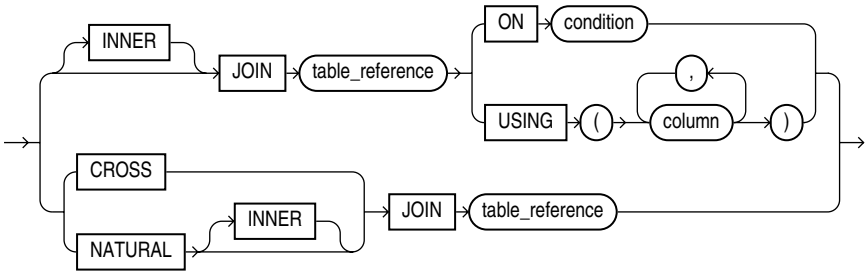


table_collection_expression::=

join_clause::=

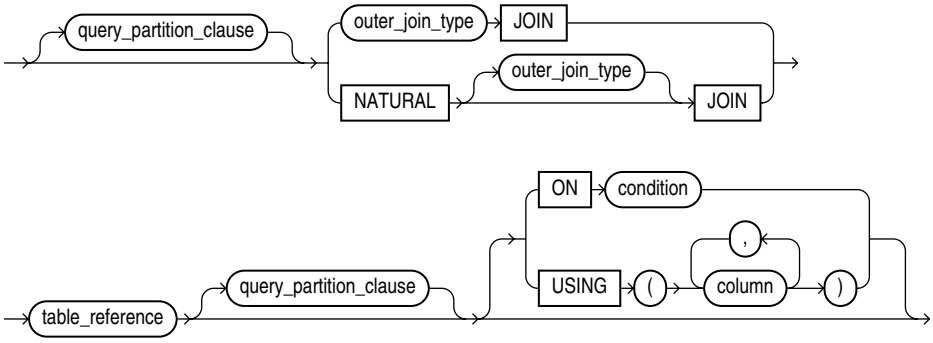


***inner_cross_join_clause*::=**



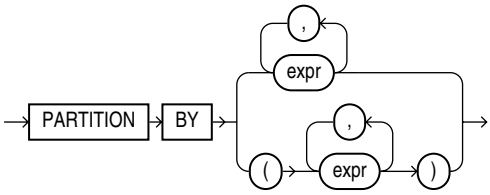
(19-6 ページの table_reference::= を参照)

***outer_join_clause*::=**

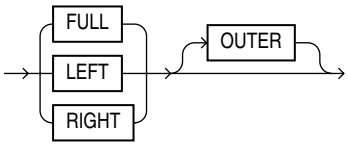


（19-6 ページの table_reference::=、19-9 ページの query_partition_clause::= を参照）

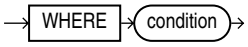
query_partition_clause::=



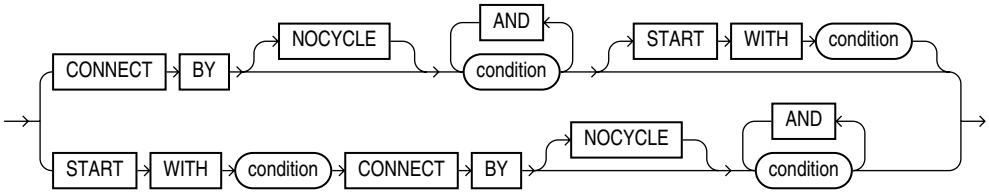
outer_join_type::=



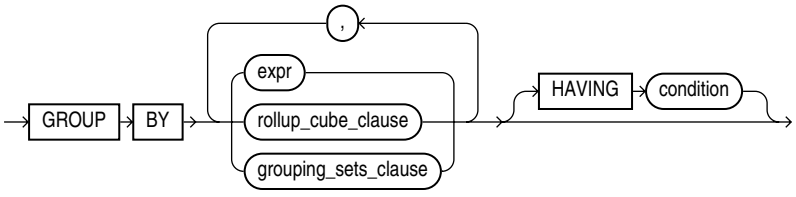
where_clause::=



hierarchical_query_clause::=

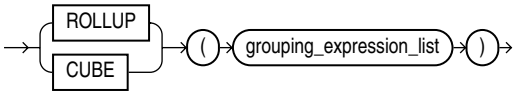


group_by_clause::=



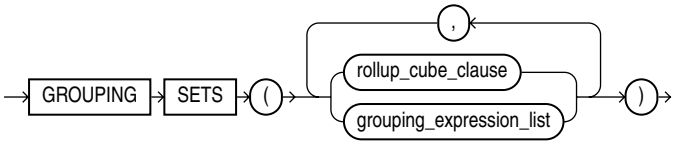
（19-10 ページの rollup_cube_clause::=、19-10 ページの grouping_sets_clause::= を参照）

***rollup_cube_clause*::=**



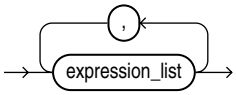
（19-10 ページの grouping_expression_list::= を参照）

***grouping_sets_clause*::=**

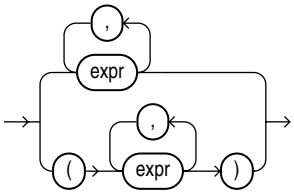


（19-10 ページの rollup_cube_clause::=、19-10 ページの grouping_expression_list::= を参照）

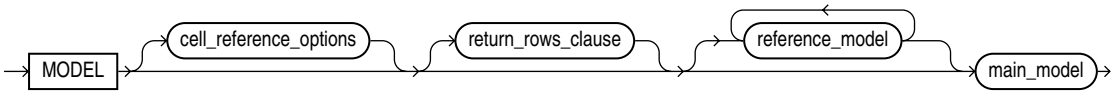
***grouping_expression_list*::=**



***expression_list*::=**

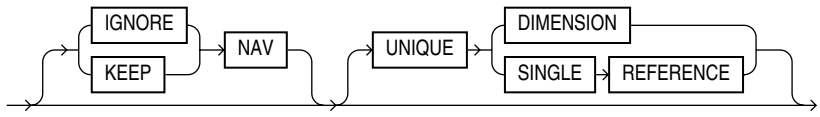


model_clause::=

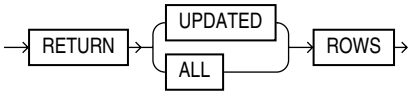


（19-10 ページの cell_reference_options::=、19-10 ページの return_rows_clause::=、
19-11 ページの reference_model::=、19-11 ページの main_model::= を参照）

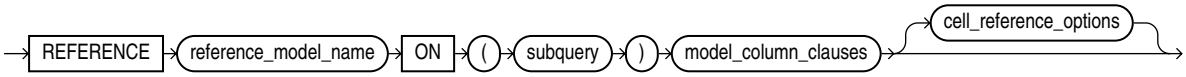
cell_reference_options::=



return_rows_clause::=

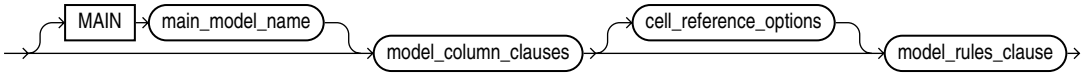


reference_model::=



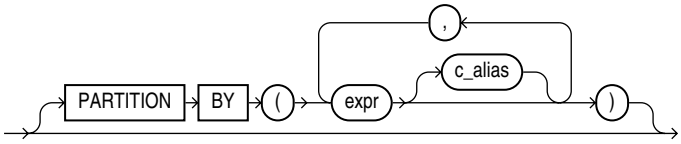
（19-11 ページの model_column_clauses::=、19-10 ページの cell_reference_options::= を参照）

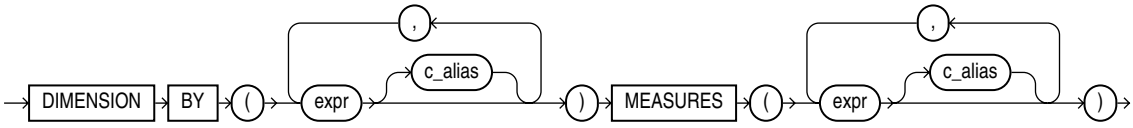
main_model::=



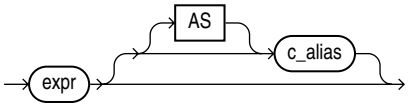
（19-11 ページの model_column_clauses::=、19-10 ページの cell_reference_options::=、19-11 ページの model_rules_clause::= を参照）

model_column_clauses::=

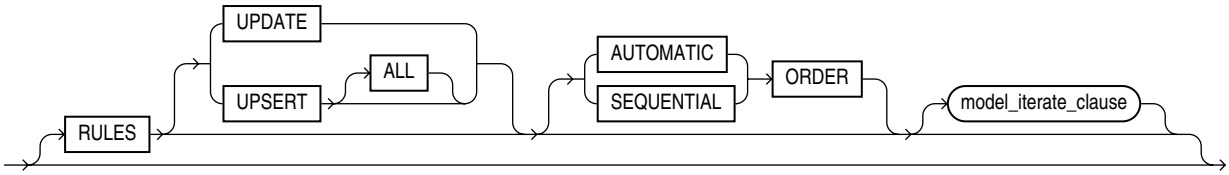


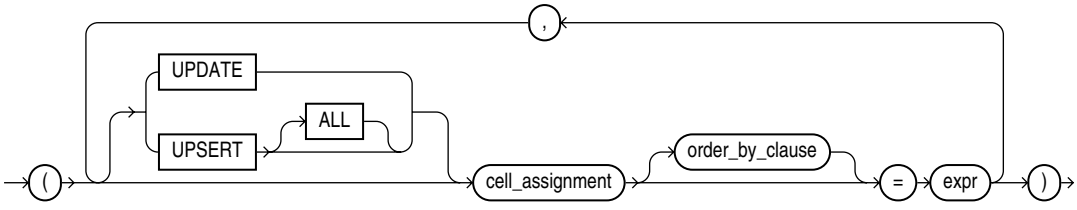


model_column::=



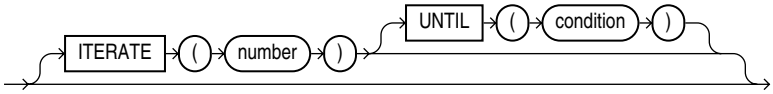
model_rules_clause::=



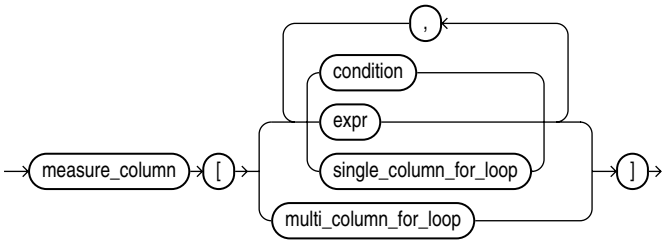


（19-12 ページの model_iterate_clause::=、19-12 ページの cell_assignment::=、19-12 ページの order_by_clause::= を参照）

***model_iterate_clause*::=**

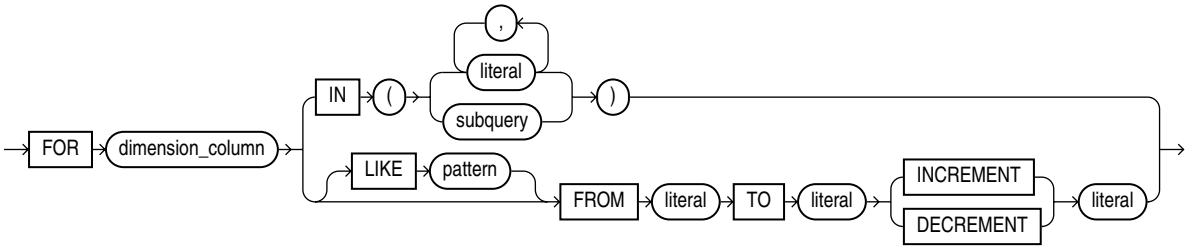


cell_assignment::=

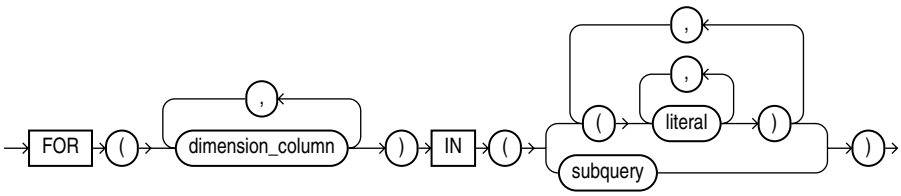


（19-12 ページの single_column_for_loop::=、19-12 ページの multi_column_for_loop::= を参照）

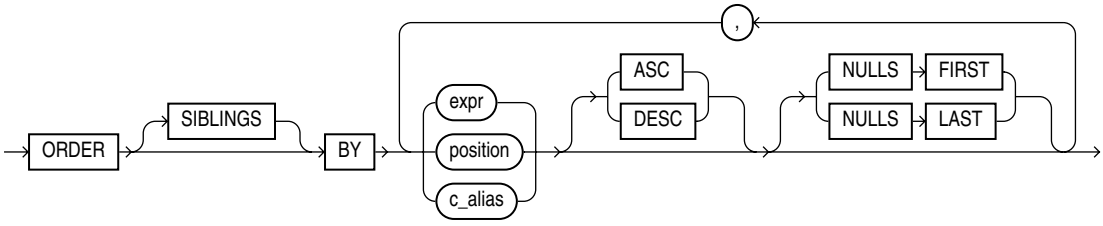
single_column_for_loop::=



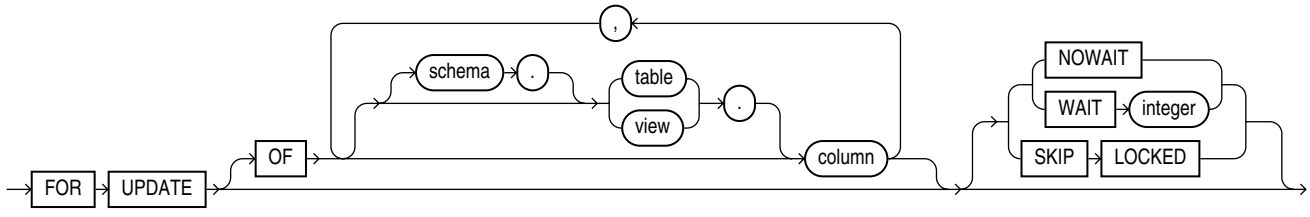
multi_column_for_loop::=



order_by_clause::=



for_update_clause::=



セマンティクス

subquery_factoring_clause

WITH *query_name* 句を使用すると、副問合せブロックに名前を割り当てることができます。問合せの名前を指定することによって、問合せに複数存在する副問合せブロックを参照することができます。問合せの名前をインライン・ビューまたは一時表として扱うことによって、問合せが最適化されます。

最上位の SELECT 文およびほとんどの副問合せでこの句を指定できます。問合せの名前は、主問合せおよび後続のすべての副問合せ（自身の問合せ名を定義する副問合せを除く）から参照できます。

副問合せのファクタリングの制限事項 : この句には、次の制限事項があります。

- *subquery_factoring_clause* は、1 つの SQL 文内に 1 つのみ指定できます。*query_name* は、それ自体の副問合せの中では指定できません。ただし、*subquery_factoring_clause* で定義した *query_name* は、その *subquery_factoring_clause* 内の後続の任意の名前付き問合せブロックで使用できます。
- 集合演算子を指定した複合問合せの場合、その問合せを構成する各問合せでは *query_name* を使用できませんが、各問合せの FROM 句では *query_name* を使用できます。

参照 :

- インライン・ビューの詳細は、『Oracle Database 概要』を参照してください。
- 19-31 ページの「副問合せのファクタリング例 :」

hint

文の実行計画を選択する場合に、オプティマイザに指示を与えるためのコメントを指定します。

参照 : ヒントの構文および説明については、2-70 ページの「ヒントの使用方法」を参照してください。

DISTINCT | UNIQUE

DISTINCT または UNIQUE を指定すると、選択された重複行の 1 行のみを戻すことができます。これらの 2 つのキーワードは同義です。重複行とは、SELECT 構文のリスト中のそれぞれの式で一致する値を持つ行のことです。

DISTINCT および UNIQUE 問合せの制限事項 : このタイプの問合せには、次の制限事項があります。

- DISTINCT または UNIQUE を指定する場合、SELECT 構文のリスト中の式すべての総バイト数は、データ・ブロックのサイズからオーバーヘッド分を引いたサイズに制限されます。このサイズは、初期化パラメータ DB_BLOCK_SIZE によって指定されます。
- *select_list* に LOB 列が含まれている場合、DISTINCT は指定できません。

ALL

ALL を指定すると、重複行を含め、選択されたすべての行を戻すことができます。デフォルトは ALL です。

*（全列ワイルド・カード）

全列ワイルド・カード（アスタリスク）を指定すると、疑似列を除いて、FROM 句に指定されているすべての表、ビューまたはマテリアライズド・ビューのすべての列を選択できます。列は、表、ビューまたはマテリアライズド・ビューの *_TAB_COLUMNS データ・ディクショナリ・ビューの COLUMN_ID によって指定されている順序で戻されます。

ビューやマテリアライズド・ビューではなく表から選択する場合、ALTER TABLE SET UNUSED 文によって UNUSED のマークが付けられた列は選択されません。

> **参照：** 12-2 ページの「ALTER TABLE」、19-31 ページの「単純問合せの例 :」および 19-48 ページの「DUAL 表からの選択例 :」を参照してください。

select_list

データベースから取り出す列を指定します。

query_name

subquery_factoring_clause ですでに指定されている名前を指定します。*select_list* で *query_name* を指定するには、*subquery_factoring_clause* を指定する必要があります。*select_list* で *query_name* を指定するには、*query_table_expression*（FROM 句）でも *query_name* を指定する必要があります。

table. | view.* | materialized view.**

オブジェクト名の後にピリオドおよびアスタリスクを指定すると、指定した表、ビューまたはマテリアライズド・ビューのすべての列を選択できます。オブジェクトの作成時に指定された順序で列の集合が戻されます。2 つ以上の表、ビューまたはマテリアライズド・ビューの行を選択する問合せを結合といいます。

他のユーザーのスキーマの表、ビューまたはマテリアライズド・ビューから選択する場合には、スキーマ修飾子を使用します。*schema* を指定しない場合、この表、ビューおよびマテリアライズド・ビューは自分のスキーマ内にあるとみなされます。

> **参照：** 「結合」（9-10 ページ）

expr

選択する情報を表す式を指定します。リスト中の列が含まれている表、ビューまたはマテリアライズド・ビューが FROM 句で *schema* 名で指定されている場合のみ、その列名を *schema* 名で指定できます。オブジェクト型のメンバー・メソッドを指定するときは、メソッドが引数をとらない場合でも、カッコを使用するメソッド名に従う必要があります。

> **参照：** 「順序値の選択例 :」（19-48 ページ）

c_alias 列式の別名を指定します。この別名は、結果セットの列のヘッダーで使用されます。AS キーワードはオプションです。別名によって、問合せ中に SELECT 構文のリストの項目名を効果的に変更できます。問合せにおいて、別名は *order_by_clause* で使用できますが、他の句では使用できません。

> **参照：**
>
> - 複数のマテリアライズド・ビューの問合せで、UNION ALL 演算子とともに *expr* AS *c_alias* 構文を使用する場合の詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。
> - *expr* の構文については、6-2 ページの「SQL 式」を参照してください。

SELECT 構文のリストの制限事項 : SELECT 構文のリストには、次の制限事項があります。

- この文に group_by_clause も指定している場合、この SELECT 構文のリストには次の式のみ指定できます。
  - 定数
  - 集計ファンクション、USER ファンクション、UID ファンクションおよび SYSDATE ファンクション
  - *group_by_clause* に指定されているものと同じ式。*group_by_clause* が副問合せの中にある場合、その副問合せの GROUP BY 列は、外部問合せの SELECT 構文のリストと対応する必要があります。副問合せの SELECT 構文のリストの列で、GROUP BY 操作に不要なものはすべて無視されます（エラーは発生しません）。
  - グループ内のすべての行が同じ値に評価される前述の式を伴っている式
- 結合内のキー保存表が 1 つのみの場合、結合ビューから ROWID を選択することができます。表の ROWID がビューの ROWID になります。

  参照 : キー保存表の詳細は、『Oracle Database 管理者ガイド』を参照してください。

- 複数の表に同じ名前の列があり、FROM 句で結合を指定した場合、表の名前または表の別名でその列名を修飾する必要があります。

FROM 句

FROM 句を指定すると、どのオブジェクトからデータを選択するかを指定できます。

query_table_expression

query_table_expression 句を使用すると、表、ビュー、マテリアライズド・ビュー、パーティションまたはサブパーティションの識別、またはオブジェクトを識別する副問合せの指定を行うことができます。

参照 : 「副問合せの使用例 :」（19-41 ページ）

ONLY ONLY 句は、ビューのみに適用されます。FROM 句のビューが階層に属し、サブビューの行を含めない場合は、ONLY 句を使用します。

flashback_query_clause

flashback_query_clause を使用すると、表、ビューまたはマテリアライズド・ビューの過去のデータを問い合せることができます。

この句は、SQL によるフラッシュバックを実装します。この句を使用すると、SELECT 構文のリスト内の各オブジェクトについて異なるシステム変更番号またはタイムスタンプを指定できます。DBMS_FLASHBACK パッケージを使用して、セッション・レベルのフラッシュバックも実装できます。

フラッシュバック問合せを使用すると、行に対して行った変更の履歴を取り出すことができます。VERSIONS_XID 疑似列を使用して、変更を行ったトランザクションの対応する識別子を取り出すことができます。また、Oracle Flashback Transaction Query を発行して、特定の行バージョンを生成したトランザクションの情報を取り出すこともできます。これを行うには、特定のトランザクション ID を FLASHBACK_TRANSACTION_QUERY データ・ディクショナリ・ビューで問い合せます。

AS OF AS OF を指定すると、特定のシステム変更番号（SCN）またはタイムスタンプでの問合せによって戻された行の単一のバージョンを取り出すことができます。SCN を指定する場合、*expr* は数値に評価される必要があります。TIMESTAMP を指定する場合、*expr* はタイムスタンプ値に評価される必要があります。指定されたシステム変更番号または時刻に存在した行が戻されます。

VERSIONS VERSIONS を指定すると、問合せによって戻された行の複数のバージョンを取り出すことができます。2 つの SCN または 2 つのタイムスタンプ値の間に存在する、行のすべてのコミット済バージョンが戻されます。戻された行には、削除後に再度挿入された行のバージョンが含まれます。

- BETWEEN SCN ... を指定すると、2 つの SCN の間に存在する行のバージョンを取り出すことができます。式は、両方とも数値に評価される必要があります。MINVALUE および MAXVALUE は、それぞれ使用可能な一番古いデータおよび最新のデータの SCN に解決されます。
- BETWEEN TIMESTAMP ... を指定すると、2 つのタイムスタンプの間に存在する行のバージョンを取り出すことができます。式は、両方ともタイムスタンプ値に評価される必要があります。MINVALUE および MAXVALUE は、それぞれ使用可能な一番古いデータおよび最新のデータのタイムスタンプに解決されます。

Oracle Database では、バージョン問合せ疑似列のグループを使用して、様々な行のバージョンに関する追加情報を取り出すことができます。詳細は、3-6 ページの「バージョン問合せ疑似列」を参照してください。

両方の句を同時に使用する場合、AS OF 句によって、SCN またはデータベースが問合せを発行した時点が判断されます。VERSIONS 句によって、AS OF で指定した時点を基準とした行のバージョンが判断されます。トランザクションが、BETWEEN の最初の値より前に開始したり、AS OF で指定した時点より後に終了した場合は、行のバージョンとして NULL が戻されます。

フラッシュバック問合せの制限事項 : この問合せには、次の制限事項があります。

- AS OF 句の式には、副問合せを指定できません。
- 一時表、外部表またはクラスタの一部である表に対するフラッシュバック問合せでは、VERSIONS 句を使用できません。
- ビューに対するフラッシュバック問合せでは VERSIONS 句を使用できません。ただし、ビューを定義する問合せには、VERSIONS 構文を使用できます。
- *query_table_expression* で *query_name* を指定した場合、この句は指定できません。

参照 :

- Oracle フラッシュバック問合せの詳細は、『Oracle Database アドバンスト・アプリケーション開発者ガイド』を参照してください。
- 「フラッシュバック問合せの使用例 :」(19-33 ページ)
- DBMS_FLASHBACK パッケージを使用したセッション・レベルのフラッシュバックの詳細は、『Oracle Database アドバンスト・アプリケーション開発者ガイド』および『Oracle Database PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。
- トランザクションの履歴の詳細は、『Oracle Database 管理者ガイド』、および『Oracle Database リファレンス』の FLASHBACK_TRANSACTION_QUERY に関する説明を参照してください。

partition_extension_clause データを取り出す表のパーティションまたはサブパーティションの名前を指定します。

レンジ・パーティション表およびリスト・パーティション・データでは、この句のかわりに、データの取出しを *table* の 1 つ以上のパーティションに制限する条件を WHERE 句に指定できます。Oracle Database はこの条件を認識し、これらのパーティションのみからデータをフェッチします。ハッシュ・パーティション・データには、この WHERE 条件を指定することはできません。

参照 : 2-106 ページの「パーティション表と索引の参照」および 19-32 ページの「パーティションからの行の選択例 :」を参照してください。

dblink 表、ビューまたはマテリアライズド・ビューが存在するリモート・データベースのデータベース・リンクの完全名または部分名を指定します。このデータベースは、Oracle Database である必要はありません。

参照 :

- データベース・リンクの参照方法の詳細は、2-104 ページの「リモート・データベース内のオブジェクトの参照」を参照してください。
- 分散問合せの詳細は、9-14 ページの「分散問合せ」および 19-47 ページの「分散問合せの使用例 :」を参照してください。

dblink を指定しない場合、その表、ビューまたはマテリアライズド・ビューは、ローカル・データベース内にあるものとみなされます。

データベース・リンクの制限事項 : データベース・リンクには、次の制限事項があります。

- リモート表のユーザー定義型またはオブジェクト REF を問い合せることはできません。
- リモート表の ANYTYPE 型、ANYDATA 型または ANYDATASET 型の列を問い合せることはできません。

table* | *view* | *materialized view データを選択する表、ビューまたはマテリアライズド・ビューの名前を指定します。

sample_clause

sample_clause を指定すると、表全体からではなく、表のランダムなサンプル・データから選択が行われます。

参照 : 「サンプルの選択例 :」(19-32 ページ)

BLOCK BLOCK を指定すると、ランダムな行サンプリングのかわりに、ランダムなブロック・サンプリングを実行できます。

ブロック・サンプリングは、全表スキャン中または高速全索引スキャン中にのみ使用可能です。より効率的な実行パスが存在する場合、ブロック・サンプリングは実行されません。特定の表または索引に対するブロック・サンプリングを確実に実行する場合は、FULL または INDEX_FFS のヒントを使用します。

sample_percent サンプルに含める行またはブロックの割合(パーセント)を指定します。0.000001 以上 100 未満の範囲の値を指定します。この割合は、各行(ブロック・サンプリングの場合は行の各クラスタ)が、サンプルの一部として選択される可能性を示します。*sample_percent* に指定した割合の行が表から正確に取り出されるわけではありません。

注意 : 統計的に適切でない想定値でこの機能を使用した場合、不正確な、または望ましくない結果になります。

SEED *seed_value* この句を指定すると、今回の実行と次の実行で同じサンプルが戻されます。*seed_value* には、0(ゼロ)～ 4294967295 の整数を指定します。この句を省略した場合、戻されるサンプルは実行ごとに異なります。

問合せ中のサンプリングの制限事項 : ビューからサンプリングを行う場合、ビューがキー保存されていることを確認する必要があります。これを確認する方法の 1 つは、CREATE TABLE ... AS *subquery* 文を使用して任意の問合せの結果を具体化し、結果として生成された問合せに対してサンプリングを実行することです。

***sample_clause* の制限事項 :** SAMPLE 句は、DML 文の副問合せで指定できません。

subquery_restriction_clause *subquery_restriction_clause* を使用すると、次のいずれかの方法で副問合せを制限できます。

WITH READ ONLY WITH READ ONLY を指定すると、表またはビューを更新禁止にできます。

WITH CHECK OPTION WITH CHECK OPTION を指定すると、副問合せに含まれない行を生成する表またはビューの変更を禁止できます。この句を DML 文の副問合せ内で使用する場合、FROM 句内の副問合せには指定できますが、WHERE 句内の副問合せには指定できません。

CONSTRAINT *constraint* CHECK OPTION 制約の名前を指定します。この識別子を省略した場合、その制約に SYS_Cn という形式の名前が自動的に割り当てられます。この場合の n は、その制約名をデータベース内で一意の名前にする整数です。

> **参照：** 「WITH CHECK OPTION 句の使用例 :」（19-39 ページ）

table_collection_expression

table_collection_expression を使用すると、問合せおよび DML 操作で、*collection_expression* 値を表として扱うことができます。*collection_expression* には、副問合せ、列、ファンクションまたはコレクション・コンストラクタのいずれかを指定できます。その形式にかかわらず、集合値（ネストした表型または VARRAY 型の値）を戻す必要があります。このようなコレクションの要素抽出プロセスを**コレクション・ネスト解除**といいます。

TABLE 式を親表と結合する場合は、オプションのプラス（+）には大きな意味があります。+ を指定すると、その 2 つの外部結合が作成され、コレクション式が NULL の場合でも、外部表の行が問合せで戻されるようになります。

> **注意：** 以前のリリースの Oracle では、*collection_expression* が副問合せの場合、*table_collection_expression* を THE *subquery* と表現していました。現在、このような表現方法は非推奨になっています。

collection_expression は、FROM 句で左側に定義された表の列を参照できます。これを**左相関**といいます。左相関は *table_collection_expression* のみで行われます。その他の副問合せは、その副問合せ以外で定義された列を参照することはできません。

オプションの (+) を使用すると、コレクションが NULL または空である場合、すべてのフィールドに NULL が設定された行を *table_collection_expression* が戻すように指定できます。この (+) は *collection_expression* が左相関を使用する場合にのみ有効です。結果は、外部結合の結果と似ています。

UPDATE または DELETE 操作で副問合せの WHERE 句に (+) 構文を使用する場合は、副問合せの FROM 句に 2 つの表を指定する必要があります。副問合せに結合が存在しないかぎり、外部結合構文は無視されます。

> **参照：**
>
> - 「外部結合」（9-11 ページ）
> - 19-44 ページの「表のコレクション例 :」および 19-45 ページの「コレクションのネスト解除例 :」を参照してください。

t_alias

相関名（表、ビュー、マテリアライズド・ビューまたは問合せを評価するための副問合せの別名）を指定します。SELECT 構文のリストがオブジェクト型属性またはオブジェクト型メソッドを参照する場合、この別名が必要になります。相関名は、相関問合せ内で最も頻繁に使用されます。表、ビューまたはマテリアライズド・ビューを参照する問合せでは、この別名を参照する必要があります。

参照 : 「相関副問合せの使用例 :」（19-47 ページ）

pivot_clause

pivot_clause を使用すると、行を列に変換し、変換処理中にデータを集計するクロス集計問合せを記述できます。ピボット演算の出力には、最初のデータセットよりも多くの列と少ない行が含まれています。*pivot_clause* では、次の手順が実行されます。

1. 句の先頭で指定されている集計ファンクションが計算されます。集計ファンクションは、複数の値を戻すように GROUP BY 句を指定する必要がありますが、*pivot_clause* には、明示的な GROUP BY 句が含まれていません。かわりに、暗黙的な GROUP BY が実行されます。暗黙的なグループ化は、*pivot_clause* で参照されていないすべての列、および *pivot_in_clause* で指定されている値セットに基づいています。
2. 列のグループ化および手順 1 で計算された集計値は、次のクロス集計出力を生成するように構成されています。
   a. 最初に、*pivot_clause* で参照されていないすべての暗黙的なグループ化列が出力されます。
   b. 次に、*pivot_in_clause* 内の値に対応している新しい列が出力されます。集計済の値はそれぞれ、クロス集計の該当する新しい列に書き換えられます。XML キーワードを指定した場合、結果は、XML 文字列としてデータを表記する単一の新しい列となります。

pivot_clause の副次句のセマンティクスは、次のとおりです。

XML オプションの XML キーワードは、問合せの XML 出力を生成します。XML キーワードを指定すると、*pivot_in_clause* には、副問合せまたはワイルド・カード・キーワード ANY を含めることができます。副問合せおよび ANY ワイルド・カードは、*pivot_in_clause* 値が事前にわかっていない場合に有効です。XML 出力では、ピボット列の値が実行時に評価されます。*pivot_in_clause* で式を使用して明示的なピボット値を指定する場合は、XML を指定することができません。

XML 出力が生成される際、集計ファンクションが各ピボット値に適用され、データベースによって、値とメジャーのすべてのペアの XML 文字列を含む XMLType の列が戻されます。

expr ピボット列の定数値への評価を行う式を指定します。オプションで、各ピボット列値の別名を指定できます。別名がない場合は、列ヘッダーが引用識別子となります。

subquery subquery は、XML キーワードとともにのみ使用されます。subquery を指定すると、subquery によって検出されたすべての値がピボットに使用されます。出力は、XML 以外のピボット問合せによって戻されるクロス集計書式とは異なります。*pivot_in_clause* で指定されている複数の列のかわりに、subquery では、XML 文字列の列が 1 つ生成されます。各行の XML 文字列は、その行の暗黙的な GROUP BY 値に対応する集計データを保持します。入力データに対応する行がない場合でも、各出力行の XML 文字列には、subquery によって検出されたすべてのピボット値が含まれています。

subquery は、ピボット問合せの実行時に、一意の値リストを戻します。subquery が一意の値を戻さない場合、Oracle Database によってランタイム・エラーが生成されます。問合せが一意の値を戻すかどうかがわからない場合は、subquery に DISTINCT キーワードを使用します。

ANY ANY キーワードは、XML キーワードとともにのみ使用されます。ANY キーワードは、ワイルド・カードとして機能し、*subquery* と同様に動作します。出力は、XML 以外のピボット問合せによって戻されるクロス集計書式とは異なります。*pivot_in_clause* で指定されている複数の列のかわりに、ANY キーワードでは、XML 文字列の列が 1 つ生成されます。各行の XML 文字列は、その行の暗黙的な GROUP BY 値に対応する集計データを保持します。ただし、*subquery* を指定した場合と比較すると、ANY ワイルド・カードでは、各出力行について、行に対応する入力データで検出されたピボット値のみを含む XML 文字列が生成されます。

参照 : PIVOT および UNPIVOT の詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。また、19-39 ページの「PIVOT および UNPIVOT の使用例 :」も参照してください。

unpivot_clause

unpivot_clause は、列を行に変換します。

- INCLUDE | EXCLUDE NULLS 句により、NULL 値の行を組み込むか、または除外するかを選択できます。INCLUDE NULLS は、アンピボット操作に NULL 値の行を組み込みます。EXCLUDE NULLS は、戻り値のセットから NULL 値の行を除外します。この句を省略すると、アンピボット操作で NULL が除外されます。
- *column* には、sales_quantity などのメジャー値を保持する各出力列の名前を指定します。
- *pivot_for_clause* には、四半期または製品などの記述子値を保持する各出力列の名前を指定します。
- *unpivot_in_clause* には、名前が *pivot_for_clause* の出力列の値となる入力データ列を指定します。これらの入力データ列には、Q1、Q2、Q3、Q4 など、カテゴリ値を指定する名前が含まれています。オプションの *alias* を指定すると、必要な値に列名をマップすることができます。

アンピボット操作は、複数の値列を単一の列に変更します。このため、値列のすべてのデータ型は、数値、文字などの同じデータ型グループに属している必要があります。

- すべての値列が CHAR の場合、アンピボットされる列は CHAR になります。値列が VARCHAR2 の場合、アンピボットされる列は VARCHAR2 になります。
- すべての値列が NUMBER の場合、アンピボットされる列は NUMBER になります。値列が BINARY_DOUBLE の場合、アンピボットされる列は BINARY_DOUBLE になります。BINARY_DOUBLE の値列はないが、いずれかの値列が BINARY_FLOAT の場合、アンピボットされる列は BINARY_FLOAT になります。

join_clause

適切な *join_clause* 構文を使用すると、データが選択され、結合の一部となる表を識別できます。*inner_cross_join_clause* を使用すると、内部結合またはクロス結合を指定できます。*outer_join_clause* を使用すると、外部結合を指定できます。

結合する行ソースが 3 つ以上ある場合は、カッコを使用してデフォルトの優先順位を無効にすることができます。たとえば、次のような構文があるとします。

```
SELECT ... FROM a JOIN (b JOIN c) ...
```

この場合、b と c が結合され、次にその結果と a が結合されます。

> **参照：** 結合の詳細は、9-10 ページの「結合」、19-40 ページの「結合問合せの使用例 :」、19-41 ページの「自己結合の使用例 :」および 19-41 ページの「外部結合の使用例 :」を参照してください。

内部結合

内部結合は、結合条件を満たす行のみを戻します。

INNER INNER を指定すると、内部結合を明示的に指定できます。

JOIN JOIN キーワードを使用すると、結合の実行を明示的に示すことができます。この構文を使用すると、WHERE 句の結合で使用されている、カンマで区切られた表の式を、FROM 句の結合構文に置き換えることができます。

ON *condition* ON 句を使用すると、結合条件を指定できます。これによって、WHERE 句の検索またはフィルタ条件とは分離して結合条件を指定できます。

USING (*column*) 両方の表で同じ名前の列同士を等価結合する場合、USING *column* 句に使用する列を指定します。両方の表で同じ名前の列同士を結合する場合のみ、この句を使用できます。この句の中では、列名を表の名前および別名で修飾しないでください。

クロス結合

CROSS キーワードを使用すると、クロス結合を実行できます。クロス結合は、2 つの関係のクロス積を生成します。カンマで区切られたの表記法と基本的に同じです。

外部結合

外部結合は、結合条件を満たすすべての行と、結合条件を満たす他方の表の行を除いた、一方の表のすべての行を戻します。指定可能な外部結合は、結合の両側に *table_reference* 構文を使用した従来の外部結合と、いずれかの側に *query_partition_clause* を使用したパーティション化された外部結合の 2 種類です。パーティション化された外部結合は、内部表の各パーティションと外部表の間で結合が行われるという点を除いて、従来の外部結合と同じです。この形式の結合では、対象のディメンションに沿って、選択的に疎データをより密にできます。このプロセスは**データの稠密化**といいます。

outer_join_type 実行する外部結合の種類を指定します。

- RIGHT を指定すると、右側外部結合が実行されます。
- LEFT を指定すると、左側外部結合が実行されます。
- FULL を指定すると、完全な外部結合または両側外部結合が実行されます。内部結合に加え、内部結合の結果に戻されない両方の表からの行は、保持され、NULL で拡張されます。
- RIGHT、LEFT または FULL の後にオプションの OUTER キーワードを指定し、外部結合の実行を明示的に示すことができます。

query_partition_clause *query_partition_clause* を使用すると、**パーティション化された外部結合**を定義できます。このような結合は、問合せによって戻されたパーティションに外部結合を適用し、従来の外部結合構文を拡張します。PARTITION BY 句で指定した各式に対する行のパーティションが作成されます。問合せの各パーティションの行は、PARTITION BY 式に対して同じ値を持ちます。

query_partition_clause は、外部結合のいずれかの側で使用できます。パーティション化された外部結合の結果は、パーティション化された結果セットの各パーティションの外部結合と結合の反対側の表との論理和になります。この形式の結果は、疎データの欠損の補完に役立つため、分析計算が簡単になります。

この句を省略した場合、表の式全体（*table_reference* に指定したすべてのもの）が単一のパーティションとして扱われるため、従来型の外部結合となります。

分析ファンクションで *query_partition_clause* を使用するには、構文の上位ブランチ（カッコなし）を使用します。この句をモデルの問合せ（*model_column_clauses* 内）またはパーティション化された外部結合（*outer_join_clause* 内）で使用するには、構文の下位ブランチ（カッコ付き）を使用します。

パーティション化された外部結合の制限事項： パーティション化された外部結合には、次の制限事項があります。

- *query_partition_clause* は、結合の右側または左側に指定できますが、両方に指定することはできません。
- パーティション化された完全外部結合（FULL）は指定できません。
- ON 句を使用して外部結合に *query_partition_clause* を指定した場合、ON 条件内には副問合せを指定できません。

参照：

- 外部結合に関するその他の規則および制限事項については、9-11 ページの「外部結合」を参照してください。
- パーティション化された外部結合およびデータの稠密化の詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。
- 「パーティション化された外部結合の使用例：」（19-43 ページ）

ON *condition* ON 句を使用すると、結合条件を指定できます。これによって、WHERE 句の検索またはフィルタ条件とは分離して結合条件を指定できます。

ON *condition* 句の制限事項 : NATURAL 外部結合を使用してこの句を指定することはできません。

USING *column* USING 句を含む外部結合の場合、問合せによって単一列が戻されます。この単一列は、結合内の一致する 2 つの列が結合したものです。この結合は、次のように機能します。

```
COALESCE (a, b) = a if a NOT NULL, else b.
```

したがって、次のことがいえます。

- 左側外部結合では、FROM 句内の左側の表から共通するすべての列値が戻されます。
- 右側外部結合では、FROM 句内の右側の表から共通するすべての列値が戻されます。
- 完全な外部結合では、結合された両方の表から共通するすべての列値が戻されます。

USING *column* 句の制限事項 :

- この句の中では、列名を表の名前および別名で修飾しないでください。
- LOB 列またはコレクション列は、USING *column* 句で指定できません。
- NATURAL 外部結合を使用してこの句を指定することはできません。

参照 : 「外部結合の使用例 :」(19-41 ページ)

NATURAL JOIN NATURAL キーワードを使用すると、自然結合を実行できます。自然結合は、2 つの表の間で同じ名前のすべての列に基づきます。2 つの表から関連する列の値が等しい行が選択されます。自然結合で使用する列を指定する場合は、表の名前または別名で列名を修飾しないでください。

自然結合またはクロス結合の表の組合せが不明瞭な場合があります。たとえば、次のような結合構文があるとします。

```
a NATURAL LEFT JOIN b LEFT JOIN c ON b.c1 = c.c1
```

この例は、次のどちらにも解釈できます。

```
a NATURAL LEFT JOIN (b LEFT JOIN c ON b.c1 = c.c1)
(a NATURAL LEFT JOIN b) LEFT JOIN c ON b.c1 = c.c1
```

このような不明瞭さをなくすため、カッコを使用して結合する表の組合せを明確にしてください。このようなカッコがないと、左から右へ表が組み合せられ、左の結合が使用されます。

自然結合の制限事項 : LOB 列、ANYTYPE 列、ANYDATA 列、ANYDATASET 列またはコレクション列は、自然結合の一部として指定できません。

where_clause

WHERE 条件を指定すると、選択する行を 1 つ以上の条件を満たす行のみに制限できます。*condition* には、有効な SQL 条件を指定します。

この句を省略した場合、FROM 句に指定されている表、ビューまたはマテリアライズド・ビューのすべての行が戻されます。

注意： この句がパーティション表またはパーティション索引の DATE 列を参照している場合、データベースは、次の条件でのみパーティション・プルーニングを実行します。

- TO_DATE ファンクションで 4 桁書式マスクを使用して年を完全に指定した表または索引パーティションを作成した場合
- TO_DATE ファンクションで 2 または 4 桁書式マスクを使用して問合せの *where_clause* に日付を指定した場合

参照：

- 条件の構文については、第 7 章「条件」を参照してください。
- 「パーティションからの行の選択例 :」（19-32 ページ）

hierarchical_query_clause

hierarchical_query_clause を使用すると、階層順序で行を選択できます。

階層問合せを含む SELECT 文では、SELECT 構文のリスト内の LEVEL 疑似列を使用できます。LEVEL は、ルート・ノードには 1 を、ルート・ノードの子であるノードには 2 を、孫であるノードには 3 を戻します（以下同様）。階層問合せによって戻されるレベルの数値は、使用可能なユーザー・メモリーによって制限されます。

Oracle は次のように階層問合せを処理します。

- 最初に、結合（指定されている場合）が、FROM 句で指定されているか、または WHERE 句述語で指定されているかが評価されます。
- CONNECT BY 条件が評価されます。
- 残りの WHERE 句述語が評価されます。

この句を指定する場合、ORDER BY および GROUP BY を指定すると、CONNECT BY 結果の階層順序が破棄されるため、これらの句のどちらも指定しないでください。同じ親の兄弟である行を順序付ける場合は、ORDER SIBLINGS BY 句を使用します。

参照： 階層問合せの詳細は、9-3 ページの「階層問合せ」および 19-45 ページの「LEVEL 疑似列の使用例 :」を参照してください。

START WITH 句

階層問合せのルートとして使用される行を識別する場合の条件を指定します。Oracle Database では、この条件を満たすすべての行がルートとして使用されます。この句を省略した場合、表内のすべての行がルート行として使用されます。

CONNECT BY 句

階層の親 / 子の行の関連を識別する条件を指定します。*connect_by_condition* には、第 7 章「条件」のどの条件でも含めることができます。ただし、親である行を参照するための PRIOR 演算子を使用する必要があります。

参照：

- LEVEL の詳細は、第 3 章「疑似列」を参照してください。
- 階層問合せの概要は、9-3 ページの「階層問合せ」を参照してください。
- 「階層問合せの例」（19-35 ページ）

group_by_clause

GROUP BY 句を指定すると、選択した行を各行の *expr* の値に基づいてグループ化し、各グループのサマリー情報を 1 行戻すことができます。この句に CUBE または ROLLUP 拡張要素を指定した場合、標準グループ化の他に超集合グループ化が生成されます。

GROUP BY 句の式には、SELECT 構文のリストに指定されている列であるかどうかにかかわらず、FROM 句の表、ビューおよびマテリアライズド・ビューの列を指定できます。

GROUP BY 句は行をグループ化しますが、結果セットの順序は保証しません。グループ化の順序付けを行うには、ORDER BY 句を使用します。

参照 :

- データを集計する SQL グループ化構文の詳細および例については、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。
- この句の例については、5-75 ページの「GROUP_ID」、「GROUPING」および「GROUPING_ID」を参照してください。
- 「GROUP BY 句の使用例 :」（19-34 ページ）

ROLLUP *simple_grouping_clause* の ROLLUP 操作を使用すると、選択した行を GROUP BY で指定した式 n、n-1、n-2、... 0 の最初の値に基づいてグループ化し、各グループのサマリー情報を 1 行戻すことができます。ROLLUP 操作を SUM ファンクションとともに使用すると、**小計値**を出力できます。ROLLUP を SUM とともに使用すると、最も詳細なレベルの小計から総計までが生成されます。COUNT などの集計ファンクションは、他の種類の超集合の出力に使用できます。

たとえば、*simple_grouping_clause* の ROLLUP 句に式を 3 つ指定した場合（n=3）、操作の結果は n+1=3+1=4 グループになります。

最初の *n* 式の値でグループ化した行を**標準行**、その他を**超集合行**といいます。

参照 : マテリアライズド・ビューで ROLLUP を使用する場合の詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

CUBE *simple_grouping_clause* の CUBE 操作を使用すると、選択した行を、指定した式のあらゆる組合せの値に基づいてグループ化できます。また、各グループのサマリー情報を 1 行戻すことができます。CUBE 操作を使用すると、**クロス集計値**を出力できます。

たとえば、*simple_grouping_clause* の CUBE 句に式を 3 つ指定した場合（n=3）、操作の結果は $2^n = 2^3 = 8$ グループになります。*n* 式の値でグループ化した行を**標準行**、その他を**超集合行**といいます。

参照 :

- マテリアライズド・ビューで CUBE を使用する場合の詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。
- 「GROUP BY CUBE 句の使用例 :」（19-34 ページ）

GROUPING SETS GROUPING SETS は、データを複数にグループ化する GROUP BY 句をさらに拡張したものです。これによって、不要な集計が排除され、効率的に集計できるようになります。必要なグループを指定すると、データベースが CUBE または ROLLUP によって生成された集計のすべてを実行する必要がなくなります。GROUPING SETS 句で指定したすべてのグループ化が計算され、UNION ALL 操作で個々のグループ化の結果が組み合されます。UNION ALL は、結果セットが重複行を含むことを許可します。

GROUP BY 句では、様々な方法で式を組み合せることができます。

- **複合列**を指定するには、カッコで列をグループ化します。データベースは、ROLLUP 操作または CUBE 操作の計算でこれらを 1 つの単位として処理します。

- **グルーピング・セットの連結**を指定するには、カンマで複数のグルーピング・セット、ROLLUP 操作および CUBE 操作を区切ります。データベースは、これらを 1 つの GROUP BY 句に結合します。結果は、各グルーピング・セットからのグループ化のクロス積です。

  参照 : 「GROUPING SETS 句の使用例 :」（19-34 ページ）

HAVING 句

HAVING 句を使用すると、戻す行のグループを、指定した条件が TRUE である行のグループのみに制限できます。この句を省略した場合、すべてのグループのサマリー行が戻されます。

where_clause および *hierarchical_query_clause* の後に、GROUP BY および HAVING を指定します。GROUP BY および HAVING 句を両方指定する場合は、どちらを先に指定してもかまいません。

参照 : 「HAVING 条件の使用例 :」（19-35 ページ）

GROUP BY 句の制限事項 : この句には、次の制限事項があります。

- LOB 列、ネストした表または VARRAY を *expr* の一部として指定できません。
- 式には、スカラー副問合せ式を除くすべての形式が可能です。
- *group_by_clause* がオブジェクト型列を参照する場合、問合せはパラレル化されません。

model_clause

model_clause では、選択した行を多次元配列として表示し、その配列内のセルにランダムにアクセスできます。*model_clause* を使用すると、一連のセルの割当てを指定できます。この割当ては**ルール**と呼ばれ、個々のセルおよびセルの範囲に対する計算を実行します。これらのルールは問合せの結果を操作しますが、データベース表は更新しません。

問合せで *model_clause* を使用する場合、SELECT 句および ORDER BY 句は、*model_column_clauses* で定義された列のみを参照する必要があります。

参照 :

- *expr* の構文については、6-2 ページの「SQL 式」を、*condition* の構文については、第 7 章「条件」を参照してください。
- 詳細および例は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。
- 「MODEL 句の例 :」（19-36 ページ）

main_model

main_model 句を使用すると、選択した行を多次元配列内で表示する方法および配列内の各セルに適用するルールを定義できます。

model_column_clauses

model_column_clauses を使用すると、問合せの列を、パーティション列、ディメンション列およびメジャー列の 3 つのグループに定義して分類できます。この 3 つの副次句内の *expr* がモデル列の場合、列の別名（*c_alias*）の指定は任意です。*expr* がモデル列でない場合、列の別名は必須です。

PARTITION BY PARTITION BY 句を使用すると、選択した行を列の値に基づいてパーティションに分割するために使用する列を指定できます。

DIMENSION BY DIMENSION BY 句を使用すると、パーティション内で行を識別する列を指定できます。ディメンション列およびパーティション列の値は、行のメジャー列に対する配列の索引として使用されます。

MEASURES MEASURES 句を使用すると、計算が実行可能な列を識別できます。個々の行のメジャー列は、パーティション列およびディメンション列の値を指定することによる参照および更新が可能なセルと同様に扱われます。

model_column *model_column* を使用すると、モデルの定義に使用する列を識別できます。*expr* が列名ではない場合、列の別名が必要です。モデル式の詳細は、6-11 ページの「モデル式」を参照してください。

cell_reference_options

cell_reference_options 句を使用すると、ルールで NULL または値なしを処理する方法および列の一意性を制約する方法を指定できます。

IGNORE NAV IGNORE NAV を指定すると、指定したデータ型の NULL または値なしに対して、次の値が戻されます。

- 数値データ型 : 0（ゼロ）
- 日時データ型 : 01-JAN-2000
- 文字データ型 : 空の文字列
- その他のデータ型 : NULL

KEEP NAV KEEP NAV を指定すると、NULL または値なしのセル値に対して NULL が戻されます。KEEP NAV はデフォルトです。

UNIQUE SINGLE REFERENCE UNIQUE SINGLE REFERENCE を指定すると、問合せの結果セット全体ではなく、ルールの右側の単一セルの参照のみが一意性をチェックされます。

UNIQUE DIMENSION UNIQUE DIMENSION を指定すると、PARTITION BY および DIMENSION BY で指定した列が、問合せに対する一意キーであるかどうかが確認されます。UNIQUE DIMENSION はデフォルトです。

model_rules_clause

model_rules_clause を使用すると、更新するセルおよびこれらのセルを更新するルールを指定できます。オプションで、ルールを適用および処理する方法も指定できます。

各ルールは割当てを表し、左側と右側にわかれています。ルールの左側は、ルールの右側によって更新されるセルを識別します。ルールの右側は、ルールの左側で指定されたセルに割り当てられる値を評価します。

UPSERT ALL UPSERT ALL を使用すると、ルールの左側に位置参照と記号参照の両方があるルールに対して UPSERT 動作が可能になります。UPSERT ALL ルールが評価されると、次の手順が実行され、アップサートするセル参照のリストが作成されます。

1. セル参照のすべてのシンボリック述語を満たす既存のセルを検索します。
2. 記号参照があるディメンションのみを使用して、これらのセルの異なるディメンション値の組合せを検索します。
3. 位置参照によって指定されたディメンション値を持つ、これらの値の組合せのクロス積が実行されます。

UPSERT ALL のセマンティクスの詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

UPSERT UPSERT を指定すると、ルールの左側で参照されるセルが多次元配列内に存在している場合、セルにルールが適用され、多次元配列内に存在しないセルに対しては新しい行が挿入されます。UPSERT 動作は、ルールの左側で位置参照が使用され、単一セルが参照されている場合にのみ適用されます。UPSERT はデフォルトです。位置参照および単一セル参照の詳細は、19-27 ページの「cell_assignment」を参照してください。

UPDATE および UPSERT は、個々のルールに同様に指定できます。特定のルールに UPDATE または UPSERT のいずれかを指定した場合、その指定は RULES 句に指定したその他のオプションより優先されます。

UPSERT [ALL] および UPDATE に関する注意： UPSERT ALL、UPSERT または UPDATE ルールに適切な条件が含まれていない場合は、別のタイプのルールに暗黙的に変換される場合があります。

- UPSERT ルールに存在述語が含まれている場合、そのルールは UPDATE ルールとして処理されます。
- UPSERT ALL ルールには、その左側に少なくとも 1 つの存在述語と 1 つの修飾述語が必要です。
  - 存在述語がない場合は、UPSERT ルールとして処理されます。
  - 修飾述語がない場合は、UPDATE ルールとして処理されます。

UPDATE UPDATE を指定するとルールの左側で参照されるセルが多次元配列内に存在している場合、そのセルにルールが適用されます。セルが存在しない場合、割当ては無視されます。

AUTOMATIC ORDER AUTOMATIC ORDER を指定すると、依存順序に基づいてルールが評価されます。この場合、セルには値が 1 回のみ割り当てられます。

SEQUENTIAL ORDER SEQUENTIAL ORDER を指定すると、表示されている順序でルールが評価されます。この場合、セルには値が複数回割り当てられます。SEQUENTIAL ORDER はデフォルトです。

ITERATE ... [UNTIL] ITERATE...[UNTIL] を使用すると、ルールを繰り返す回数を指定できます。また、早期終了条件も指定できます。UNTIL 条件を囲むカッコの使用は任意です。

ITERATE ... [UNTIL] を指定した場合、ルールは表示されている順序で評価されます。*model_rules_clause* で AUTOMATIC ORDER および ITERATE ... [UNTIL] の両方が指定されている場合、エラーが戻されます。

cell_assignment

cell_assignment 句は、ルールの左側に使用し、更新する 1 つ以上のセルを指定します。単一セルを参照する *cell_assignment* は、**単一セル参照**といいます。複数のセルが参照される場合は、**複数セル参照**といいます。

model_clause で定義したすべてのディメンション列は、*cell_assignment* 句で修飾する必要があります。ディメンションは、記号参照または位置参照を使用して修飾できます。

記号参照は、*dimension_column=constant* などのブール条件を使用して、単一のディメンション列を修飾します。**位置参照**では、DIMENSION BY 句でディメンション列の位置が示されます。記号参照と位置参照の唯一の相違点は、NULL の処理です。

a[x=null,y=2000] のような単一セルの記号参照を使用すると、x=null が FALSE と評価されるため、該当するセルは存在しません。ただし、a[null,2000] のような単一セルの位置参照を使用すると、null = null が TRUE と評価されるため、x が NULL、y が 2000 のセルが該当します。単一セルの位置参照を使用すると、ディメンション列が NULL のセルを参照、更新および挿入できます。

記号参照または位置参照を使用して、ディメンション列の値を表す条件または式を指定できます。*condition* に集計ファンクションまたは CV ファンクションを含めることはできません。また、*condition* は単一のディメンション列を参照する必要があります。*expr* に副問合せを含めることはできません。モデル式の詳細は、6-11 ページの「モデル式」を参照してください。

single_column_for_loop

single_column_for_loop 句を使用すると、更新するセルの範囲を単一のディメンション列内で指定できます。

IN 句を使用すると、ディメンション列の値を値のリストまたは副問合せとして指定できます。副問合せを使用する場合、次の制限事項があります。

- 相関問合せは使用できません。
- 10,000 を超える行を戻すことはできません。
- WITH 句で定義された問合せを使用できません。

FROM 句を使用すると、ディメンション列の値の範囲を指定できます。範囲内の増分は不連続でもかまいません。FROM 句は、加算および減算がサポートされているデータ型の列にのみ使用できます。INCREMENT および DECREMENT の値は、正の値である必要があります。

オプションで、FROM 句内で LIKE 句を指定することができます。LIKE 句の *pattern* は、単一のパターン一致文字 % を含む文字列です。この文字は、実行時に FROM 句の現在の増分値または減分値で置き換えられます。

FOR ループで使用されるディメンション以外のすべてのディメンションが単一セル参照に関係する場合は、式で新しい行を挿入できます。FOR ループによって生成されたディメンション値の組合せの数は、MODEL 句の行制限（10,000）の計算に含まれます。

multi_column_for_loop

multi_column_for_loop 句を使用すると、更新するセルの範囲を複数のディメンション列にまたがって指定できます。IN 句を使用すると、ディメンション列の値を複数の値のリストまたは副問合せとして指定できます。副問合せを使用する場合、次の制限事項があります。

- 相関問合せは使用できません。
- 10,000 を超える行を戻すことはできません。
- WITH 句で定義された問合せを使用できません。

FOR ループで使用されるディメンション以外のすべてのディメンションが単一セル参照に関係する場合は、式で新しい行を挿入できます。FOR ループによって生成されたディメンション値の組合せの数は、MODEL 句の行制限（10,000）の計算に含まれます。

> **参照：** MODEL 句で FOR ループを使用する方法の詳細は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

order_by_clause

ORDER BY 句を使用すると、ルールの左側のセルを評価する順序を指定できます。*expr* は、ディメンションまたはメジャー列に変換される必要があります。ORDER BY 句を指定しない場合、DIMENSION BY 句で指定した列の順序がデフォルトで使用されます。詳細は、19-29 ページの「order_by_clause」を参照してください。

***order_by_clause* 句の制限事項：** モデル・ルールで ORDER BY 句を使用する場合には、次の制限があります。

- *model_clause* 句の *order_by_clause* には、SIBLINGS、*position* または *c_alias* を指定できません。
- モデル・ルールの左側にこの句を指定し、右側に FOR ループも指定することはできません。

expr

ルールの右側に指定した単一または複数のセルの値を表す式を指定します。*expr* に副問合せを含めることはできません。モデル式の詳細は、6-11 ページの「モデル式」を参照してください。

return_rows_clause

return_rows_clause を使用すると、選択されたすべての行を戻すか、モデル・ルールによって更新された行のみを戻すかどうかを指定できます。ALL はデフォルトです。

reference_model

reference_model は、*model_clause* 内から複数の配列にアクセスする必要がある場合に使用します。この句は、問合せの結果に基づいて、読取り専用の多次元配列を定義します。

reference_model 句の副次句は、*main_model* 句と同じセマンティクスを持ちます。詳細は、19-26 ページの「cell_reference_options」および 19-25 ページの「model_column_clauses」を参照してください。

***reference_model* 句の制限事項：** この句には、次の制限事項があります。

- PARTITION BY 列を参照モデルに指定することはできません。
- 参照モデルの副問合せから外部副問合せの列を参照することはできません。

集合演算子 : UNION、UNION ALL、INTERSECT および MINUS

集合演算子は、2 つの SELECT 文によって戻された行を 1 つの結果に結合します。それぞれのコンポーネント問合せで選択される列の数とデータ型は同じである必要がありますが、列の長さは異なってもかまいません。結果セット内の列の名前は、集合演算子の前にある SELECT 構文のリスト内の式の名前です。

集合演算子で 3 つ以上の問合せを結合する場合、隣接する問合せが左から右へ評価されます。副問合せを囲むカッコは任意指定です。この評価順序を変更する場合、カッコを使用します。

これらの演算子の詳細および使用方法の制限事項については、9-7 ページの「UNION [ALL]、INTERSECT および MINUS 演算子」を参照してください。

order_by_clause

ORDER BY 句を使用すると、文によって戻された行を順序付けることができます。*order_by_clause* を指定しない場合、同じ問合せで取り出される行の順序が異なることがあります。

SIBLINGS SIBLINGS キーワードは、*hierarchical_query_clause*（CONNECT BY）を指定する場合のみに有効です。ORDER SIBLINGS BY は階層問合せ句で指定した任意の順序を保持し、兄弟関係にある階層に *order_by_clause* を適用します。

expr *expr* を使用すると、*expr* の値を基準にして行を順番付けることができます。式は、SELECT 構文のリストの列、あるいは FROM 句の表、ビューまたはマテリアライズド・ビューの列に基づきます。

position *position* を使用すると、SELECT 構文のリストの指定した位置にある式の値に基づいて行を順序付けることができます。*position* には整数を指定する必要があります。

order_by_clause には複数の式を指定できます。この場合、まず、最初の式の値に基づいて行がソートされます。次に、最初の式と同じ値を持つ行が 2 番目の式の値に基づいてソートされる、というように処理が行われます。NULL 値は昇順では最後に、降順では先頭にソートされます。問合せ結果の順位付けの詳細は、9-10 ページの「問合せ結果のソート」を参照してください。

ASC | DESC 昇順か降順かを指定します。デフォルトは ASC です。

NULLS FIRST | NULLS LAST NULL 値を含む戻された行が順序の最初にくるか、最後にくるかを指定します。

NULLS LAST は昇順のデフォルトで、NULLS FIRST は降順のデフォルトです。

ORDER BY 句の制限事項： ORDER BY 句には次の制限事項があります。

- この文中で DISTINCT 演算子を指定した場合、SELECT 構文のリストに指定された列でないかぎり、この句は列を参照することはできません。
- *order_by_clause* には最大 255 個の式を指定できます。
- LOB 列、LONG 列、LONG RAW 列、ネストした表または VARRAY を使用して順位付けすることはできません。
- 同じ文中で group_by_clause を指定する場合、この *order_by_clause* は次の式に制限されます。
  - 定数
  - 集計ファンクション
  - 分析ファンクション
  - USER ファンクション、UID ファンクションおよび SYSDATE ファンクション
  - *group_by_clause* に指定されているものと同じ式
  - グループ内のすべての行が同じ値に評価されるこれらの式を導出する式

参照： 「ORDER BY 句の使用例 :」（19-36 ページ）

for_update_clause

FOR UPDATE 句を使用すると、トランザクションが終了する前に、選択した行が別のユーザーによってロックまたは更新されることがないように、選択した行をロックできます。最上位の SELECT 文でのみ、この句を指定できます。副問合せでは指定できません。

注意： LOB 値を更新する場合、その LOB を含む行をロックしておく必要があります。行をロックする方法の 1 つに、埋込み SELECT ... FOR UPDATE 文があります。この場合、プログラム言語の 1 つまたは DBMS_LOB パッケージを使用します。LOB の書込み前に行う行のロックの詳細は、『Oracle Database SecureFiles およびラージ・オブジェクト開発者ガイド』を参照してください。

親表の行がロックされても、ネストした表の行はロックされません。ネストした表の行をロックする場合、ネストした表を明示的にロックする必要があります。

FOR UPDATE 句の制限事項： この句には、次の制限事項があります。

- この句を DISTINCT 演算子または CURSOR 式、集合演算子、*group_by_clause*、または集計ファンクションの構造体とともに指定することはできません。
- この句がロックした表は、同じ文で参照された LONG 列および順序と同じデータベース内にある必要があります。

参照： 「FOR UPDATE 句の使用例 :」（19-38 ページ）

OF ... *column*

OF ... *column* 句を使用すると、結合内の特定の表またはビューで選択された行のみをロックできます。OF 句の列は、どの表またはビューの行をロックするかを識別する場合にのみ使用します。指定する列は重要ではありません。ただし、列の別名ではなく、実際の列名を指定する必要があります。この句を省略した場合、問合せ内のすべての表の選択された行がロックされます。

NOWAIT | WAIT

NOWAIT および WAIT 句を使用すると、他のユーザーによってロックされている行を SELECT 文がロックしようとする場合に処理する方法を Oracle に指示できます。

- NOWAIT を指定すると、ロックされている場合に制御がすぐに戻ります。
- WAIT を指定すると、行が使用可能になるまで *integer* 秒待機した後で制御が戻されます。

WAIT および NOWAIT のどちらも指定しない場合、行が使用可能になるまで待機した後で SELECT 文の結果が戻されます。

SKIP LOCKED SKIP LOCKED は、競合するトランザクションを処理するもう 1 つの方法で、対象のいくつかの行をロックします。SKIP LOCKED を指定すると、WHERE 句で指定された行をロックし、ロック済であることが検出された行をスキップするようにデータベースに指示することができます。この機能は、問合せの目的が、行の実際の内容ではなく、その数を取得することである場合に有用です。

WAIT 句および SKIP LOCKED 句の注意事項

WAIT または SKIP LOCKED を指定したときに排他モードで表がロックされていると、表のロックが解除されるまでは SELECT 文の結果が戻りません。WAIT では、指定されている待機時間にかかわらず、SELECT FOR UPDATE 句がブロックされます。

例

副問合せのファクタリング例 : 次の文は、結合を含む初期問合せブロックに対する問合せの名前 dept_costs および avg_cost を作成し、主問合せの本体でその問合せの名前を使用します。

```
WITH
   dept_costs AS (
      SELECT department_name, SUM(salary) dept_total
         FROM employees e, departments d
         WHERE e.department_id = d.department_id
      GROUP BY department_name),
   avg_cost AS (
      SELECT SUM(dept_total)/COUNT(*) avg
      FROM dept_costs)
SELECT * FROM dept_costs
   WHERE dept_total >
      (SELECT avg FROM avg_cost)
      ORDER BY department_name;

DEPARTMENT_NAME                DEPT_TOTAL
------------------------------ ----------
Sales                              313800
Shipping                           156400
```

単純問合せの例 : 次の文は、部門番号 30 の従業員表 employees の行を選択します。

```
SELECT *
   FROM employees
   WHERE department_id = 30
   ORDER BY last_name;
```

次の文は、部門番号 30 の購買係を除くすべての従業員の名前、職種、給与および部門番号を選択します。

```
SELECT last_name, job_id, salary, department_id
   FROM employees
   WHERE NOT (job_id = 'PU_CLERK' AND department_id = 30)
   ORDER BY last_name;
```

次の文は、FROM 句の副問合せから、すべての従業員数と給与合計がすべての部門に対して占める割合を部門ごとに戻します。

```
SELECT a.department_id "Department",
   a.num_emp/b.total_count "%_Employees",
   a.sal_sum/b.total_sal "%_Salary"
FROM
(SELECT department_id, COUNT(*) num_emp, SUM(salary) sal_sum
   FROM employees
   GROUP BY department_id) a,
(SELECT COUNT(*) total_count, SUM(salary) total_sal
   FROM employees) b
ORDER BY a.department_id;
```

パーティションからの行の選択例： FROM 句にキーワード PARTITION を指定することによって、パーティション表の 1 つのパーティションから行を選択できます。次の SQL 文は、サンプル表 sh.sales の sales_q2_2000 パーティションへ別名を割り当て、行を取り出します。

```
SELECT * FROM sales PARTITION (sales_q2_2000) s
   WHERE s.amount_sold > 1500
   ORDER BY cust_id, time_id, channel_id;
```

次の文は、oe.orders 表から、指定した日付より早い注文が含まれる行を選択します。

```
SELECT * FROM orders
   WHERE order_date < TO_DATE('2000-06-15', 'YYYY-MM-DD');
```

サンプルの選択例： 次の問合せは、oe.orders 表の注文数を推定します。

```
SELECT COUNT(*) * 10 FROM orders SAMPLE (10);

COUNT(*)*10
-----------
         70
```

問合せでは推定値が戻されるため、実際の戻り値は問合せを行うたびに異なる場合があります。

```
SELECT COUNT(*) * 10 FROM orders SAMPLE (10);

COUNT(*)*10
-----------
         80
```

次の問合せでは、前述の問合せにシード値を追加します。同じシード値では、常に同じ推定値が戻されます。

```
SELECT COUNT(*) * 10 FROM orders SAMPLE(10) SEED (1);

COUNT(*)*10
-----------
        110

SELECT COUNT(*) * 10 FROM orders SAMPLE(10) SEED(4);

COUNT(*)*10
-----------
        120

SELECT COUNT(*) * 10 FROM orders SAMPLE(10) SEED (1);

COUNT(*)*10
-----------
        110
```

フラッシュバック問合せの使用例： 次の文は、サンプル表 hr.employees の現在の値を表示し、値を変更します。この例の目的はデモであるため、使用している時間隔が非常に短くなっています。実際の環境での時間隔は、これより長いのが一般的です。

```
SELECT salary FROM employees
   WHERE last_name = 'Chung';

    SALARY
----------
      3800

UPDATE employees SET salary = 4000
   WHERE last_name = 'Chung';
1 row updated.

SELECT salary FROM employees
   WHERE last_name = 'Chung';

    SALARY
----------
      4000
```

更新前の値を確認するには、次のフラッシュバック問合せを使用します。

```
SELECT salary FROM employees
   AS OF TIMESTAMP (SYSTIMESTAMP - INTERVAL '1' MINUTE)
   WHERE last_name = 'Chung';

    SALARY
----------
      3800
```

過去の特定の期間における値を確認するには、次のバージョン・フラッシュバック問合せを使用します。

```
SELECT salary FROM employees
  VERSIONS BETWEEN TIMESTAMP
    SYSTIMESTAMP - INTERVAL '10' MINUTE AND
    SYSTIMESTAMP - INTERVAL '1' MINUTE
  WHERE last_name = 'Chung';
```

元の値に戻すには、フラッシュバック問合せを別の UPDATE 文の副問合せとして使用します。

```
UPDATE employees SET salary =
   (SELECT salary FROM employees
   AS OF TIMESTAMP (SYSTIMESTAMP - INTERVAL '2' MINUTE)
   WHERE last_name = 'Chung')
   WHERE last_name = 'Chung';
1 row updated.

SELECT salary FROM employees
   WHERE last_name = 'Chung';

    SALARY
----------
      3800
```

GROUP BY 句の使用例 : 次の文は、employees 表の各部門について最低給与と最高給与を戻します。

```
SELECT department_id, MIN(salary), MAX (salary)
     FROM employees
     GROUP BY department_id
   ORDER BY department_id;
```

次の文は、各部門の事務員について最高給与と最低給与を戻します。

```
SELECT department_id, MIN(salary), MAX (salary)
     FROM employees
     WHERE job_id = 'PU_CLERK'
     GROUP BY department_id
   ORDER BY department_id;
```

GROUP BY CUBE 句の使用例 : 部門および職種のすべての組合せについて、従業員数と平均年収を戻すには、サンプル表 hr.employees および hr.departments に次の問合せを発行します。

```
SELECT DECODE(GROUPING(department_name), 1, 'All Departments',
      department_name) AS department_name,
   DECODE(GROUPING(job_id), 1, 'All Jobs', job_id) AS job_id,
   COUNT(*) "Total Empl", AVG(salary) * 12 "Average Sal"
   FROM employees e, departments d
   WHERE d.department_id = e.department_id
   GROUP BY CUBE (department_name, job_id)
   ORDER BY department_name, job_id;

DEPARTMENT_NAME                JOB_ID     Total Empl Average Sal
------------------------------ ---------- ---------- -----------
Accounting                     AC_ACCOUNT          1       99600
Accounting                     AC_MGR              1      144000
Accounting                     All Jobs            2      121800
Administration                 AD_ASST             1       52800
. . .
All Departments                ST_MAN              5       87360
All Departments                All Jobs          107  77798.1308
```

GROUPING SETS 句の使用例 : 次の例は、指定した 3 つのグループで集計した販売の合計を示します。

- (channel_desc, calendar_month_desc, country_id)
- (channel_desc, country_id)
- (calendar_month_desc, country_id)

GROUPING SETS 構文を指定しない場合、SQL はより複雑になり、問合せの効果は低くなります。たとえば、3 つの個別の問合せを実行し、UNION 演算を行うか、または CUBE(channel_desc, calendar_month_desc, country_id) 操作を指定した問合せを実行し、生成される 8 つのグループから 5 つを除去します。

```
SELECT channel_desc, calendar_month_desc, co.country_id,
      TO_CHAR(sum(amount_sold) , '9,999,999,999') SALES$
   FROM sales, customers, times, channels, countries co
   WHERE sales.time_id=times.time_id
      AND sales.cust_id=customers.cust_id
      AND sales.channel_id= channels.channel_id
      AND customers.country_id = co.country_id
      AND channels.channel_desc IN ('Direct Sales', 'Internet')
      AND times.calendar_month_desc IN ('2000-09', '2000-10')
      AND co.country_iso_code IN ('UK', 'US')
  GROUP BY GROUPING SETS(
      (channel_desc, calendar_month_desc, co.country_id),
```

```
      (channel_desc, co.country_id),
      (calendar_month_desc, co.country_id) );

CHANNEL_DESC         CALENDAR CO SALES$
-------------------- -------- -- --------------
Direct Sales         2000-09  UK      1,378,126
Direct Sales         2000-10  UK      1,388,051
Direct Sales         2000-09  US      2,835,557
Direct Sales         2000-10  US      2,908,706
Internet             2000-09  UK        911,739
Internet             2000-10  UK        876,571
Internet             2000-09  US      1,732,240
Internet             2000-10  US      1,893,753
Direct Sales                  UK      2,766,177
Direct Sales                  US      5,744,263
Internet                      UK      1,788,310
Internet                      US      3,625,993
                     2000-09  UK      2,289,865
                     2000-09  US      4,567,797
                     2000-10  UK      2,264,622
                     2000-10  US      4,802,459
```

> **参照：** これらのファンクションの詳細は、5-75 ページの「GROUP_ID」、「GROUPING」および「GROUPING_ID」を参照してください。

階層問合せの例 CONNECT BY 句を指定した次の問合せは、親である行の employee_id 値が子である行の manager_id 値と等しいという階層関係を定義します。

```
SELECT last_name, employee_id, manager_id FROM employees
   CONNECT BY employee_id = manager_id
   ORDER BY last_name;
```

次の CONNECT BY 句では、PRIOR 演算子が employee_id 値のみに適用されます。この条件を評価するために、データベースは親である行に対しては employee_id の値を評価し、子である行に対しては manager_id、salary および commission_pct のそれぞれの値を評価します。

```
SELECT last_name, employee_id, manager_id FROM employees
   CONNECT BY PRIOR employee_id = manager_id
   AND salary > commission_pct
   ORDER BY last_name;
```

子である行を限定する場合、manager_id の値と親である行の employee_id の値が等しく、salary の値が commission_pct の値より大きい必要があります。

HAVING 条件の使用例： 次の文は、従業員の最低給与が 5,000 ドル未満の部門についての最高給与と最低給与を戻します。

```
SELECT department_id, MIN(salary), MAX (salary)
   FROM employees
   GROUP BY department_id
   HAVING MIN(salary) < 5000
   ORDER BY department_id;

DEPARTMENT_ID MIN(SALARY) MAX(SALARY)
------------- ----------- -----------
           10        4400        4400
           30        2500       11000
           50        2100        8200
           60        4200        9000
```

次の例は、HAVING 句で相関副問合せを使用して、マネージャのいない部門および部門のないマネージャを結果セットから除外しています。

```
SELECT department_id, manager_id
   FROM employees
   GROUP BY department_id, manager_id HAVING (department_id, manager_id) IN
   (SELECT department_id, manager_id FROM employees x
      WHERE x.department_id = employees.department_id)
   ORDER BY department_id;
```

ORDER BY 句の使用例 : 次の文は、employees 表からすべての購買係のレコードを選択し、その給与によって降順にソートします。

```
SELECT *
   FROM employees
   WHERE job_id = 'PU_CLERK'
   ORDER BY salary DESC;
```

次の文は、employees 表から情報を選択し、最初に部門番号で昇順にソートした後、給与で降順にソートします。

```
SELECT last_name, department_id, salary
   FROM employees
   ORDER BY department_id ASC, salary DESC, last_name;
```

次の文は、前述の SELECT 文と同じ情報を選択し、ORDER BY 位置表記法を使用します。ここでは、まず department_id で昇順に、次に salary で降順に、最後に last_name でアルファベット順に順序付けします。

```
SELECT last_name, department_id, salary
   FROM employees
   ORDER BY 2 ASC, 3 DESC, 1;
```

MODEL 句の例 : ここで作成されたビューは、サンプル・スキーマ sh に基づいており、後述の例で使用されています。

```
CREATE OR REPLACE VIEW sales_view_ref AS
  SELECT country_name country,
         prod_name prod,
         calendar_year year,
         SUM(amount_sold) sale,
         COUNT(amount_sold) cnt
    FROM sales,times,customers,countries,products
    WHERE sales.time_id = times.time_id AND
          sales.prod_id = products.prod_id AND
          sales.cust_id = customers.cust_id AND
          customers.country_id = countries.country_id AND
          ( customers.country_id = 52779 OR
            customers.country_id = 52776 ) AND
          ( prod_name = 'Standard Mouse' OR
            prod_name = 'Mouse Pad' )
    GROUP BY country_name,prod_name,calendar_year;

SELECT country, prod, year, sale
  FROM sales_view_ref
  ORDER BY country, prod, year;

COUNTRY       PROD                                     YEAR        SALE
----------    ----------------------------------     --------   ---------
France        Mouse Pad                                1998     2509.42
France        Mouse Pad                                1999     3678.69
France        Mouse Pad                                2000     3000.72
France        Mouse Pad                                2001     3269.09
France        Standard Mouse                           1998     2390.83
```

```
France       Standard Mouse                          1999     2280.45
France       Standard Mouse                          2000     1274.31
France       Standard Mouse                          2001     2164.54
Germany      Mouse Pad                               1998     5827.87
Germany      Mouse Pad                               1999     8346.44
Germany      Mouse Pad                               2000     7375.46
Germany      Mouse Pad                               2001     9535.08
Germany      Standard Mouse                          1998     7116.11
Germany      Standard Mouse                          1999     6263.14
Germany      Standard Mouse                          2000     2637.31
Germany      Standard Mouse                          2001     6456.13

16 rows selected.
```

次の例では、国、製品、年および売上を含む列が存在する sales_view_ref から多次元配列を作成します。また、次の処理も実行されます。

- 2001 年のマウス・パッドの売上を含む行が存在する場合、1999 年および 2000 年のマウス・パッドの売上を 2001 年のマウス・パッドの売上に割り当てます。
- 2002 年のスタンダード・マウスの売上を含む行が存在しない場合、2001 年のスタンダード・マウスの売上を 2002 年のスタンダード・マウスの売上に割り当てます。

```
SELECT country,prod,year,s
  FROM sales_view_ref
  MODEL
    PARTITION BY (country)
    DIMENSION BY (prod, year)
    MEASURES (sale s)
    IGNORE NAV
    UNIQUE DIMENSION
    RULES UPSERT SEQUENTIAL ORDER
    (
      s[prod='Mouse Pad', year=2001] =
        s['Mouse Pad', 1999] + s['Mouse Pad', 2000],
      s['Standard Mouse', 2002] = s['Standard Mouse', 2001]
    )
  ORDER BY country, prod, year;

COUNTRY      PROD                                    YEAR        SALE
----------   ----------------------------------    --------   ---------
France       Mouse Pad                               1998     2509.42
France       Mouse Pad                               1999     3678.69
France       Mouse Pad                               2000     3000.72
France       Mouse Pad                               2001     6679.41
France       Standard Mouse                          1998     2390.83
France       Standard Mouse                          1999     2280.45
France       Standard Mouse                          2000     1274.31
France       Standard Mouse                          2001     2164.54
France       Standard Mouse                          2002     2164.54
Germany      Mouse Pad                               1998     5827.87
Germany      Mouse Pad                               1999     8346.44
Germany      Mouse Pad                               2000     7375.46
Germany      Mouse Pad                               2001     15721.9
Germany      Standard Mouse                          1998     7116.11
Germany      Standard Mouse                          1999     6263.14
Germany      Standard Mouse                          2000     2637.31
Germany      Standard Mouse                          2001     6456.13
Germany      Standard Mouse                          2002     6456.13

18 rows selected.
```

最初のルールでは、ルールの左側で記号参照が使用されているため、UPDATE 動作が使用されます。ルールの左側で表される行が存在するため、メジャー列が更新されます。行が存在しない場合、何も実行されません。

2 番目のルールでは、ルールの左側で位置参照が使用され、単一セルが参照されているため、UPSERT 動作が使用されます。行が存在しないため、新しい行が追加され、関連するメジャー列が更新されます。行が存在する場合、メジャー列は更新されません。

> **参照：** 詳細および例は、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

次の例は、同じ sales_view_ref ビューおよび分析ファンクション SUM を使用して、国および年ごとの累計（csum）を計算しています。

```
SELECT country, year, sale, csum
   FROM
   (SELECT country, year, SUM(sale) sale
    FROM sales_view_ref
    GROUP BY country, year
   )
   MODEL DIMENSION BY (country, year)
         MEASURES (sale, 0 csum)
         RULES (csum[any, any]=
                  SUM(sale) OVER (PARTITION BY country
                                  ORDER BY year
                                  ROWS UNBOUNDED PRECEDING)
                )
   ORDER BY country, year;

COUNTRY               YEAR       SALE       CSUM
--------------- ---------- ---------- ----------
France                1998    4900.25    4900.25
France                1999    5959.14   10859.39
France                2000    4275.03   15134.42
France                2001    5433.63   20568.05
Germany               1998   12943.98   12943.98
Germany               1999   14609.58   27553.56
Germany               2000   10012.77   37566.33
Germany               2001   15991.21   53557.54

8 rows selected.
```

FOR UPDATE 句の使用例： 次の文は、employees 表中のオックスフォード勤務（location_id は 2500）の購買係の行をロックし、departments 表中の購買係が存在するオックスフォードの部門の行をロックします。

```
SELECT e.employee_id, e.salary, e.commission_pct
   FROM employees e, departments d
   WHERE job_id = 'SA_REP'
   AND e.department_id = d.department_id
   AND location_id = 2500
   FOR UPDATE
   ORDER BY e.employee_id;
```

次の文は、employees 表中のオックスフォード勤務の購買係の行のみをロックします。departments 表中の行はロックされません。

```
SELECT e.employee_id, e.salary, e.commission_pct
   FROM employees e JOIN departments d
   USING (department_id)
   WHERE job_id = 'SA_REP'
   AND location_id = 2500
   FOR UPDATE OF e.salary
   ORDER BY e.employee_id;
```

WITH CHECK OPTION 句の使用例： 次の文は、3 番目の値が副問合せ *where_clause* の条件に違反していても有効です。

```
INSERT INTO (SELECT department_id, department_name, location_id
   FROM departments WHERE location_id < 2000)
   VALUES (9999, 'Entertainment', 2500);
```

ただし、次の文は WITH CHECK OPTION 句を含むため、無効になります。

```
INSERT INTO (SELECT department_id, department_name, location_id
   FROM departments WHERE location_id < 2000 WITH CHECK OPTION)
   VALUES (9999, 'Entertainment', 2500);
     *
ERROR at line 2:
ORA-01402: view WITH CHECK OPTION where-clause violation
```

PIVOT および UNPIVOT の使用例： oe.orders 表には、注文が発注された日時（order_date）、発注方法（order_mode）および注文の合計数（order_total）に関する情報とその他の情報が含まれています。次の例は、PIVOT 句を使用して、order_mode 値を列にピボットし、処理中に order_total データを集計して発注モードごとの年間合計を取得する方法を示します。

```
CREATE TABLE pivot_table AS
SELECT * FROM
(SELECT EXTRACT(YEAR FROM order_date) year, order_mode, order_total FROM orders)
PIVOT
(SUM(order_total) FOR order_mode IN ('direct' AS Store, 'online' AS Internet));

SELECT * FROM pivot_table ORDER BY year;

      YEAR      STORE   INTERNET
---------- ---------- ----------
      1990    61655.7
      1996     5546.6
      1997        310
      1998   309929.8   100056.6
      1999  1274078.8  1271019.5
      2000   252108.3   393349.4

6 rows selected.
```

UNPIVOT 句では、入力列ヘッダーが 1 つ以上の記述子列の値として出力され、入力列値が 1 つ以上のメジャー列の値として出力されるように、指定した列を変換します。次に示す最初の問合せでは、デフォルトで NULL が除外されています。2 番目の問合せは、INCLUDE NULLS 句を使用して NULL を組み込むことができることを示しています。

```
SELECT * FROM pivot_table
  UNPIVOT (yearly_total FOR order_mode IN (store AS 'direct', internet AS 'online'))
  ORDER BY year, order_mode;

      YEAR ORDER_ YEARLY_TOTAL
---------- ------ ------------
      1990 direct      61655.7
      1996 direct       5546.6
      1997 direct          310
      1998 direct     309929.8
      1998 online     100056.6
      1999 direct    1274078.8
      1999 online    1271019.5
      2000 direct     252108.3
      2000 online     393349.4
9 rows selected.
```

```
SELECT * FROM pivot_table
  UNPIVOT INCLUDE NULLS
    (yearly_total FOR order_mode IN (store AS 'direct', internet AS 'online'))
  ORDER BY year, order_mode;

      YEAR ORDER_ YEARLY_TOTAL
---------- ------ ------------
      1990 direct      61655.7
      1990 online
      1996 direct       5546.6
      1996 online
      1997 direct          310
      1997 online
      1998 direct     309929.8
      1998 online     100056.6
      1999 direct    1274078.8
      1999 online    1271019.5
      2000 direct     252108.3
      2000 online     393349.4

12 rows selected.
```

結合問合せの使用例： 次の例は、問合せにおける様々な表の結合方法を示します。最初の例では、等価結合は、それぞれの従業員の名前と職種、およびその従業員が属する部門の番号と名前を戻します。

```
SELECT last_name, job_id, departments.department_id, department_name
   FROM employees, departments
   WHERE employees.department_id = departments.department_id
   ORDER BY last_name, job_id;

LAST_NAME           JOB_ID     DEPARTMENT_ID DEPARTMENT_NAME
------------------- ---------- ------------- ----------------------
. . .
Sciarra             FI_ACCOUNT           100 Finance
Urman               FI_ACCOUNT           100 Finance
Popp                FI_ACCOUNT           100 Finance
. . .
```

従業員の名前および職種は部門名とは別の表に格納されているため、このデータを戻す場合は結合を使用する必要があります。次の結合条件に従って、2 つの表の行が結合されます。

```
employees.department_id = departments.department_id
```

次の等価結合は、すべての販売マネージャの名前、職種、部門番号および部門名を戻します。

```
SELECT last_name, job_id, departments.department_id, department_name
   FROM employees, departments
   WHERE employees.department_id = departments.department_id
   AND job_id = 'SA_MAN'
   ORDER BY last_name;

LAST_NAME           JOB_ID     DEPARTMENT_ID DEPARTMENT_NAME
------------------- ---------- ------------- -----------------------
Russell             SA_MAN               80 Sales
Partners            SA_MAN               80 Sales
Errazuriz           SA_MAN               80 Sales
Cambrault           SA_MAN               80 Sales
Zlotkey             SA_MAN               80 Sales
```

この問合せは、次の *where_clause* 条件を使用して 'SA_MAN' という job 値を持つ行のみを戻すこと以外は、前述の例と同じです。

副問合せの使用例： 次の文は、従業員 'Lorentz' と同じ部門で働く従業員を判断します。

```
SELECT last_name, department_id FROM employees
   WHERE department_id =
     (SELECT department_id FROM employees
      WHERE last_name = 'Lorentz')
   ORDER BY last_name, department_id;
```

次の文は、employees 表の職種を変更した（job_history 表に示される）すべての従業員の給与を 10% 上げます。

```
UPDATE employees
    SET salary = salary * 1.1
    WHERE employee_id IN (SELECT employee_id FROM job_history);
```

次の文は、departments 表から 3 つの列のみを伴って新しい表 new_departments を作成します。

```
CREATE TABLE new_departments
   (department_id, department_name, location_id)
   AS SELECT department_id, department_name, location_id
   FROM departments;
```

自己結合の使用例： 次の問合せは、自己結合を使用して、それぞれの従業員の名前およびその従業員の上司の名前を戻します。出力を短くするために WHERE 句が追加されています。

```
SELECT e1.last_name||' works for '||e2.last_name
   "Employees and Their Managers"
   FROM employees e1, employees e2
   WHERE e1.manager_id = e2.employee_id
      AND e1.last_name LIKE 'R%'
   ORDER BY e1.last_name;

Employees and Their Managers
------------------------------
Rajs works for Mourgos
Raphaely works for King
Rogers works for Kaufling
Russell works for King
```

この問合せの結合条件では、サンプル表 employees に対する別名 e1 および e2 を使用します。

```
e1.manager_id = e2.employee_id
```

外部結合の使用例： 次の例では、パーティション化された外部結合によって行のデータの欠損を補完し、分析ファンクションの指定および信頼性の高いレポートの書式設定を簡単にする方法を示します。この例では、結合で使用する小規模なデータ表を最初に作成します。

```
SELECT d.department_id, e.last_name
   FROM departments d LEFT OUTER JOIN employees e
   ON d.department_id = e.department_id
   ORDER BY d.department_id, e.last_name;
```

これは、次に示す以前の Oracle Database の外部結合の構文と同じ問合せです。

```
SELECT d.department_id, e.last_name
   FROM departments d, employees e
   WHERE d.department_id = e.department_id(+)
   ORDER BY d.department_id, e.last_name;
```

この構文ではなく、前述の例で示した、より柔軟性が高い FROM 句の結合構文を使用することをお薦めします。

左側外部結合では、従業員のいない部門を含むすべての部門が戻されます。右側外側結合が指定された同一文では、どの部門にも割り当てられていない従業員を含むすべての従業員が戻されます。

> **注意：** 前述の例の従業員表には、従業員 Zeuss が追加されていますが、この従業員はサンプル・データには含まれません。

```
SELECT d.department_id, e.last_name
   FROM departments d RIGHT OUTER JOIN employees e
   ON d.department_id = e.department_id
   ORDER BY d.department_id, e.last_name;

DEPARTMENT_ID LAST_NAME
------------- -------------------------
. . .
          110 Higgins
          110 Gietz
              Grant
              Zeuss
```

この結果からは、Grant と Zeuss という従業員の department_id が NULL かどうか、またはその department_id が departments 表に存在するかどうかは不明です。これを確認するには、完全な外部結合が必要です。

```
SELECT d.department_id as d_dept_id, e.department_id as e_dept_id,
      e.last_name
   FROM departments d FULL OUTER JOIN employees e
   ON d.department_id = e.department_id
   ORDER BY d.department_id, e.last_name;

 D_DEPT_ID  E_DEPT_ID LAST_NAME
---------- ---------- -------------------------
  . . .
       110        110 Gietz
       110        110 Higgins
  . . .
       260
       270
                  999 Zeuss
                      Grant
```

この例の列名は、結合状態にある両方の表で同じであるため、結合構文の USING 句を指定することで、共通列機能も使用できます。出力は、一致する 2 つの department_id 列が USING 句によって結合されることを除き、前述の例と同じです。

```
SELECT department_id AS d_e_dept_id, e.last_name
   FROM departments d FULL OUTER JOIN employees e
   USING (department_id)
   ORDER BY department_id, e.last_name;

D_E_DEPT_ID LAST_NAME
----------- -------------------------
  . . .
        110 Higgins
        110 Gietz
  . . .
        260
        270
        999 Zeuss
            Grant
```

パーティション化された外部結合の使用例 : 次の例では、パーティション化された外部結合によって行の欠損を補完し、分析計算の指定および信頼性の高いレポートの書式設定を簡単にする方法を示します。この例では、結合で使用する単純な表を最初に作成し、移入します。

```
CREATE TABLE inventory (time_id    DATE,
                        product    VARCHAR2(10),
                        quantity   NUMBER);

INSERT INTO inventory VALUES (TO_DATE('01/04/01', 'DD/MM/YY'), 'bottle', 10);
INSERT INTO inventory VALUES (TO_DATE('06/04/01', 'DD/MM/YY'), 'bottle', 10);
INSERT INTO inventory VALUES (TO_DATE('01/04/01', 'DD/MM/YY'), 'can', 10);
INSERT INTO inventory VALUES (TO_DATE('04/04/01', 'DD/MM/YY'), 'can', 10);

SELECT times.time_id, product, quantity FROM inventory
   PARTITION BY  (product)
   RIGHT OUTER JOIN times ON (times.time_id = inventory.time_id)
   WHERE times.time_id BETWEEN TO_DATE('01/04/01', 'DD/MM/YY')
      AND TO_DATE('06/04/01', 'DD/MM/YY')
   ORDER BY  2,1;

TIME_ID   PRODUCT      QUANTITY
--------- ---------- ----------
01-APR-01 bottle             10
02-APR-01 bottle
03-APR-01 bottle
04-APR-01 bottle
05-APR-01 bottle
06-APR-01 bottle             10
06-APR-01 bottle              8
01-APR-01 can                10
01-APR-01 can                15
02-APR-01 can
03-APR-01 can
04-APR-01 can                10
04-APR-01 can                11
05-APR-01 can
06-APR-01 can

15 rows selected.
```

これによって、製品ディメンションの各パーティションのデータの密度は、時間ディメンションに沿ってより密になります。ただし、各パーティションに新しく追加された行の quantity 列はそれぞれ NULL です。時間順に NULL 値をそれより前の NULL ではない値に置き換えて表示すると、よりわかりやすくなります。これを行うには、問合せ結果の最上位に次のように分析ファンクション LAST_VALUE を適用します。

```
SELECT time_id, product, LAST_VALUE(quantity IGNORE NULLS)
   OVER (PARTITION BY product ORDER BY time_id) quantity
   FROM ( SELECT times.time_id, product, quantity
             FROM inventory PARTITION BY  (product)
                RIGHT OUTER JOIN times ON (times.time_id = inventory.time_id)
   WHERE times.time_id BETWEEN TO_DATE('01/04/01', 'DD/MM/YY')
      AND TO_DATE('06/04/01', 'DD/MM/YY'))
   ORDER BY  2,1;

TIME_ID   PRODUCT      QUANTITY
--------- ---------- ----------
01-APR-01 bottle             10
02-APR-01 bottle             10
03-APR-01 bottle             10
04-APR-01 bottle             10
05-APR-01 bottle             10
06-APR-01 bottle              8
```

```
06-APR-01 bottle             8
01-APR-01 can               15
01-APR-01 can               15
02-APR-01 can               15
03-APR-01 can               15
04-APR-01 can               11
04-APR-01 can               11
05-APR-01 can               11
06-APR-01 can               11

15 rows selected.
```

> **参照：** 時系列計算における欠損補完の詳細および使用例については、『Oracle Database データ・ウェアハウス・ガイド』を参照してください。

アンチ結合の使用例： 次の例では、特定の部門の集合に所属していない従業員のリストを選択します。

```
SELECT * FROM employees
   WHERE department_id NOT IN
   (SELECT department_id FROM departments
       WHERE location_id = 1700)
   ORDER BY last_name;
```

セミ結合の使用例： 次の例では、副問合せに一致する行が employees 表に多数存在する場合でも、departments 表から 1 つの行のみが戻されます。employees の salary 列に索引が定義されていない場合、セミ結合を使用すると、問合せのパフォーマンスが向上します。

```
SELECT * FROM departments
   WHERE EXISTS
   (SELECT * FROM employees
       WHERE departments.department_id = employees.department_id
       AND employees.salary > 2500)
   ORDER BY department_name;
```

表のコレクション例： DML 操作は、表の列として定義された場合にのみ、ネストした表で実行できます。したがって、INSERT、DELETE または UPDATE 文の *query_table_expr_clause* が *table_collection_expression* の場合、コレクション式は、TABLE ファンクションを使用して表のネストした表の列を選択する副問合せである必要があります。次の例は、次の使用例に基づいています。

データベースに、department_id 列、location_id 列、manager_id 列を持つ hr_info 表と、マネージャごとのすべての従業員の last_name 列、department_id 列および salary 列を持つネストした表型 people の列が含まれていると仮定します。

```
CREATE TYPE people_typ AS OBJECT (
   last_name      VARCHAR2(25),
   department_id  NUMBER(4),
   salary         NUMBER(8,2));
/
CREATE TYPE people_tab_typ AS TABLE OF people_typ;
/
CREATE TABLE hr_info (
   department_id   NUMBER(4),
   location_id     NUMBER(4),
   manager_id      NUMBER(6),
   people          people_tab_typ)
   NESTED TABLE people STORE AS people_stor_tab;

INSERT INTO hr_info VALUES (280, 1800, 999, people_tab_typ());
```

次の例では、hr_info 表の部門番号 280 のネストした表 people の列に値を挿入します。

```
INSERT INTO TABLE(SELECT h.people FROM hr_info h
   WHERE h.department_id = 280)
   VALUES ('Smith', 280, 1750);
```

次の例では、部門番号 280 のネストした表 people を更新します。

```
UPDATE TABLE(SELECT h.people FROM hr_info h
   WHERE h.department_id = 280) p
   SET p.salary = p.salary + 100;
```

次の例では、部門番号 280 のネストした表 people を削除します。

```
DELETE TABLE(SELECT h.people FROM hr_info h
   WHERE h.department_id = 280) p
   WHERE p.salary > 1700;
```

コレクションのネスト解除例： ネストした表の列からデータを選択するには、TABLE ファンクションを使用して、ネストした表を表の列として処理します。このプロセスを**コレクション・ネスト解除**と呼びます。

次の文を使用すると、前述の例で作成した hr_info からすべての行を取得し、hr_info のネストした表 people の列からすべての行を取得できます。

```
SELECT t1.department_id, t2.* FROM hr_info t1, TABLE(t1.people) t2
   WHERE t2.department_id = t1.department_id;
```

people は、hr_info のネストした表の列ではなく、last_name、department_id、address、hiredate および salary 列と別の表であると仮定します。次の文を使用して、前述の例と同じ行を抽出できます。

```
SELECT t1.department_id, t2.*
   FROM hr_info t1, TABLE(CAST(MULTISET(
      SELECT t3.last_name, t3.department_id, t3.salary
         FROM people t3
      WHERE t3.department_id = t1.department_id)
      AS people_tab_typ)) t2;
```

最後に、people は hr_info 表のネストした表の列でも、表そのものでもないと仮定します。かわりに、すべての従業員の名前、部門および給与を様々な情報から抽出する people_func ファンクションを作成しておきます。次の問合せを使用して、前述の例と同様の情報を得ることができます。

```
SELECT t1.department_id, t2.* FROM hr_info t1, TABLE(CAST
   (people_func( ... ) AS people_tab_typ)) t2;
```

> **参照：** コレクション・ネスト解除の別の例は、『Oracle Database オブジェクト・リレーショナル開発者ガイド』を参照してください。

LEVEL 疑似列の使用例： 次の文は、すべての従業員を階層順序で戻します。職種が AD_VP である従業員がルート行となるように定義されています。また、親である行の従業員番号が上司の従業員番号となるように、親である行の子である行が定義されています。

```
SELECT LPAD(' ',2*(LEVEL-1)) || last_name org_chart,
        employee_id, manager_id, job_id
    FROM employees
    START WITH job_id = 'AD_VP'
    CONNECT BY PRIOR employee_id = manager_id;

ORG_CHART          EMPLOYEE_ID MANAGER_ID JOB_ID
------------------ ----------- ---------- ----------
Kochhar                    101        100 AD_VP
  Greenberg                108        101 FI_MGR
    Faviet                 109        108 FI_ACCOUNT
```

```
    Chen                  110        108 FI_ACCOUNT
    Sciarra               111        108 FI_ACCOUNT
    Urman                 112        108 FI_ACCOUNT
    Popp                  113        108 FI_ACCOUNT
  Whalen                  200        101 AD_ASST
  Mavris                  203        101 HR_REP
  Baer                    204        101 PR_REP
  Higgins                 205        101 AC_MGR
    Gietz                 206        205 AC_ACCOUNT
De Haan                   102        100 AD_VP
  Hunold                  103        102 IT_PROG
    Ernst                 104        103 IT_PROG
    Austin                105        103 IT_PROG
    Pataballa             106        103 IT_PROG
    Lorentz               107        103 IT_PROG
```

次の文は、前述の例とほぼ同じですが、職種が FI_MGR である従業員を選択しません。

```
SELECT LPAD(' ',2*(LEVEL-1)) || last_name org_chart,
       employee_id, manager_id, job_id
    FROM employees
    WHERE job_id != 'FI_MGR'
    START WITH job_id = 'AD_VP'
    CONNECT BY PRIOR employee_id = manager_id;

ORG_CHART          EMPLOYEE_ID MANAGER_ID JOB_ID
------------------ ----------- ---------- ----------
Kochhar                   101        100 AD_VP
    Faviet                109        108 FI_ACCOUNT
    Chen                  110        108 FI_ACCOUNT
    Sciarra               111        108 FI_ACCOUNT
    Urman                 112        108 FI_ACCOUNT
    Popp                  113        108 FI_ACCOUNT
  Whalen                  200        101 AD_ASST
  Mavris                  203        101 HR_REP
  Baer                    204        101 PR_REP
  Higgins                 205        101 AC_MGR
    Gietz                 206        205 AC_ACCOUNT
De Haan                   102        100 AD_VP
  Hunold                  103        102 IT_PROG
    Ernst                 104        103 IT_PROG
    Austin                105        103 IT_PROG
    Pataballa             106        103 IT_PROG
    Lorentz               107        103 IT_PROG
```

Greenberg が管理する従業員は戻されますが、マネージャ Greenberg は戻されません。

次の文も、前述の例と同じですが、LEVEL 疑似列を使用して管理階層の最初の 2 つのレベルのみを選択します。

```
SELECT LPAD(' ',2*(LEVEL-1)) || last_name org_chart,
employee_id, manager_id, job_id
    FROM employees
    START WITH job_id = 'AD_PRES'
    CONNECT BY PRIOR employee_id = manager_id AND LEVEL <= 2;

ORG_CHART          EMPLOYEE_ID MANAGER_ID JOB_ID
------------------ ----------- ---------- ----------
King                      100            AD_PRES
  Kochhar                 101        100 AD_VP
  De Haan                 102        100 AD_VP
  Raphaely                114        100 PU_MAN
  Weiss                   120        100 ST_MAN
```

```
Fripp                   121        100 ST_MAN
Kaufling                122        100 ST_MAN
Vollman                 123        100 ST_MAN
Mourgos                 124        100 ST_MAN
Russell                 145        100 SA_MAN
Partners                146        100 SA_MAN
Errazuriz               147        100 SA_MAN
Cambrault               148        100 SA_MAN
Zlotkey                 149        100 SA_MAN
Hartstein               201        100 MK_MAN
```

分散問合せの使用例： 次の文は、ローカル・データベース上の departments 表と、remote データベース上の employees 表を結合します。

```
SELECT last_name, department_name
   FROM employees@remote, departments
   WHERE employees.department_id = departments.department_id;
```

相関副問合せの使用例： 次に、相関副問合せの構文の一般的な例を示します。

```
SELECT select_list
    FROM table1 t_alias1
    WHERE expr operator
        (SELECT column_list
            FROM table2 t_alias2
            WHERE t_alias1.column
               operator t_alias2.column);

UPDATE table1 t_alias1
    SET column =
        (SELECT expr
            FROM table2 t_alias2
            WHERE t_alias1.column = t_alias2.column);

DELETE FROM table1 t_alias1
    WHERE column operator
        (SELECT expr
            FROM table2 t_alias2
            WHERE t_alias1.column = t_alias2.column);
```

次の文は、部門内の平均給与を超える給与を支給されている従業員の情報を戻します。給与情報が格納されている empoyees 表に別名を割り当て、相関副問合せではその別名を使用します。

```
SELECT department_id, last_name, salary
   FROM employees x
   WHERE salary > (SELECT AVG(salary)
      FROM employees
      WHERE x.department_id = department_id)
   ORDER BY department_id;
```

親問合せでは、相関副問合せを使用して同一部門の従業員の平均給与を、employees 表の行ごとに計算します。相関副問合せは、employees 表の各行について次の手順を実行します。

1. 行の department_id を判断します。
2. department_id に基づいて親問合せが評価されます。
3. 行の部門の平均給与より高い給与の行がある場合は、その行を戻します。

副問合せは、employees 表の各行につき 1 回ずつ評価されます。

DUAL 表からの選択例： 次の文は、現在の日付を戻します。

```
SELECT SYSDATE FROM DUAL;
```

employees 表から簡単に SYSDATE を選択できますが、このとき、employees 表のすべての行に対して 1 件ずつ 14 行の同じ SYSDATE が戻ります。このため、DUAL から選択する方が便利です。

順序値の選択例： 次の文は、employees_seq 順序を増分し、新しい値を戻します。

```
SELECT employees_seq.nextval
    FROM DUAL;
```

次の文は、employees_seq の現在値を選択します。

```
SELECT employees_seq.currval
    FROM DUAL;
```

SET CONSTRAINT[S]

用途

SET CONSTRAINTS 文を使用すると、遅延可能な制約の検証を、各 DML 文の実行後に行うか（IMMEDIATE）、トランザクションのコミット時に行うか（DEFERRED）をトランザクションごとに指定できます。この文を使用して、制約名のリストまたは ALL 制約のモードを設定できます。

SET CONSTRAINTS モードは、トランザクションの存続期間中、または別の SET CONSTRAINTS 文によってモードがリセットされるまで継続します。

> **注意：** ALTER SESSION 文を SET CONSTRAINTS 句とともに使用して、すべての遅延可能制約を設定することもできます。これは、現在のセッションの各トランザクションの開始時に SET CONSTRAINTS 文を発行することと同等です。

トリガー定義の内部でこの文を指定することはできません。

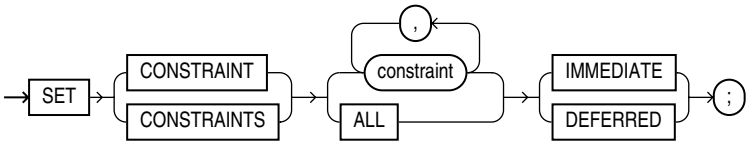
SET CONSTRAINTS は、分散型の文にすることができます。処理中のトランザクションを持つ既存のデータベース・リンクには SET CONSTRAINTS ALL 文の発行時にその発行が通知され、新しいリンクにはトランザクションの開始直後にその発行が通知されます。

前提条件

遅延可能な制約を検証する時期を指定する場合は、表が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、制約が適用される表に対する SELECT 権限が必要です。

構文

***set_constraints*::=**



セマンティクス

constraint

1 つ以上の整合性制約の名前を指定します。

ALL

ALL を指定すると、このトランザクションに対するすべての遅延可能な制約を設定できます。

IMMEDIATE

IMMEDIATE を指定すると、指定された制約が、各制約 DML 文の実行時に即時にチェックされます。Oracle Database では、チェック済のすべての制約に一貫性があり、他の SET CONSTRAINTS 文が発行されていない場合、トランザクションで以前に遅延された制約が最初にチェックされ、その後すぐにそのトランザクションの他の文の制約チェックが継続してチェックされます。制約のチェックに失敗した場合は、エラーが通知されます。その時点で、COMMIT 文を実行すると、トランザクション全体が元に戻されます。

COMMIT を正常に実行できるかどうかをチェックする方法としてトランザクションの終了直後に制約を行います。制約をトランザクション内の最後の文として IMMEDIATE に設定することで、予期しないロールバックを回避できます。いずれかの制約のチェックに失敗した場合は、トランザクションをコミットする前にエラーを解決できます。

DEFERRED

DEFERRED を指定すると、遅延可能な制約によって指定された条件が、トランザクションのコミット時に検証されます。

> **注意：** SET CONSTRAINTS ALL IMMEDIATE 文を発行することによって、遅延可能な制約をコミットする前に、それらの制約が完全に適用されたかどうかを検証できます。

例

制約の設定例： 次の文は、このトランザクション内のすべての遅延可能な制約が、各 DML 文の直後に検証されるように設定します。

```
SET CONSTRAINTS ALL IMMEDIATE;
```

次の文は、トランザクションのコミット時に 3 つの遅延制約を検証します。この例では、制約を NOT DEFERRABLE に設定すると失敗します。

```
SET CONSTRAINTS emp_job_nn, emp_salary_min ,
   hr.jhist_dept_fk@remote DEFERRED;
```

SET ROLE

用途

データベースでは、ユーザー・ログイン時に、ユーザーに明示的に付与されたすべての権限およびユーザーのすべてのデフォルトのロールが使用可能になります。セッション中、ユーザーまたはアプリケーションは、SET ROLE 文を使用して、そのセッションに対してロールを何度でも使用可能または使用禁止にできます。

148 を超えるユーザー定義のロールを一度に使用可能にできません。

注意： ほとんどのロールは、直接的に付与されているか、または他のロールを介して付与されていないかぎり、使用可能または使用禁止にできません。ただし、保護アプリケーション・ロールは、関連付けられている PL/SQL パッケージによって付与して使用可能にすることができます。保護アプリケーション・ロールについては、15-57 ページの「USING package」の CREATE ROLE セマンティクスおよび『Oracle Database セキュリティ・ガイド』を参照してください。

SESSION_ROLES データ・ディクショナリ・ビューを問い合せると、現在使用可能なロールを参照できます。

参照：

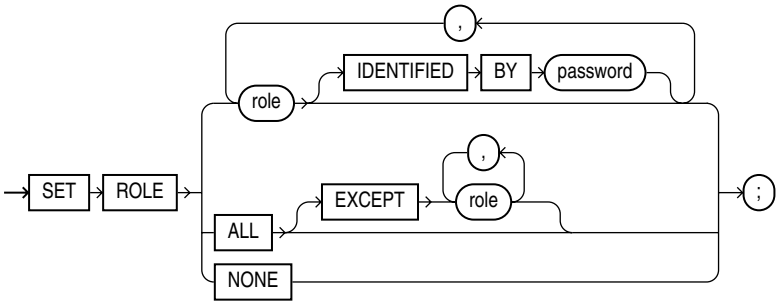
- ロールの作成の詳細は、15-56 ページの「CREATE ROLE」を参照してください。
- ユーザーのデフォルト・ロールの変更については、13-5 ページの「ALTER USER」を参照してください。
- SESSION_ROLES セッション・パラメータの詳細は、『Oracle Database リファレンス』を参照してください。

前提条件

SET ROLE 文に指定するロールが付与されている必要があります。

構文

***set_role*::=**



セマンティクス

role

現行のセッションで使用可能にするロールを指定します。リストされないロールおよび使用可能でないロールは、現行のセッションで使用禁止になります。

IDENTIFIED BY *password* 句では、ロールに対するパスワードを指定します。ロールにパスワードが設定されている場合は、指定する必要があります。

ロールの設定の制限事項： グローバルに識別されるロールは指定できません。グローバル・ロールは、ログイン時にデフォルトで使用可能になり、後で再度使用可能にすることはできません。

ALL 句

ALL を指定すると、現行のセッションに対して付与されているすべてのロールを使用可能にできます。ただし、EXCEPT 句に任意に指定されているロールは除きます。

EXCEPT 句に指定するロールは、ユーザーに直接付与されている必要があります。他のロールによってユーザーに付与されたものは無効です。

直接付与されているロール、および他のロールを介してユーザーに付与されているロールを EXCEPT 句に指定した場合、そのロールの付与先のロールにより、そのロールは使用可能のままになります。

ALL 句の制限事項： このオプションを使用して、ユーザーに直接付与されているパスワード付きのロールを使用可能にすることはできません。

NONE

NONE を指定すると、現行のセッションで、DEFAULT ロールを含むすべてのロールを使用禁止にできます。

例

ロールの設定例： 次の文は、現行のセッションのパスワード warehouse によって識別されるロール dw_manager を使用可能にします。

```
SET ROLE dw_manager IDENTIFIED BY warehouse;
```

次の文は、現行のセッションで付与されているロールをすべて使用可能にします。

```
SET ROLE ALL;
```

次の文は、dw_manager を除くロールをすべて使用可能にします。

```
SET ROLE ALL EXCEPT dw_manager;
```

次の文は、現行のセッションで付与されているすべてのロールを使用禁止にします。

```
SET ROLE NONE;
```

SET TRANSACTION

用途

SET TRANSACTION 文を使用すると、現行のトランザクションを読取り専用または読み書き両用として設定したり、分離レベルを設定したり、指定したロールバック・セグメントにトランザクションを割り当てることができます。

TX ロックを取得する操作によって、トランザクションが暗黙的に開始されます。

- データを変更する文が発行されたとき
- SELECT ... FOR UPDATE 文が発行されたとき
- SET TRANSACTION 文または DBMS_TRANSACTION パッケージによってトランザクションが明示的に開始されたとき

COMMIT 文または ROLLBACK 文を発行すると、現行のトランザクションが明示的に終了されます。

SET TRANSACTION 文によって実行される処理は、現行のトランザクションのみに影響します。他のユーザーまたは他のトランザクションには影響しません。COMMIT 文または ROLLBACK 文を発行すると、トランザクションは常に終了されます。現行のトランザクションはデータ定義言語（DDL）文の実行の前後で Oracle Database によって暗黙的にコミットされます。

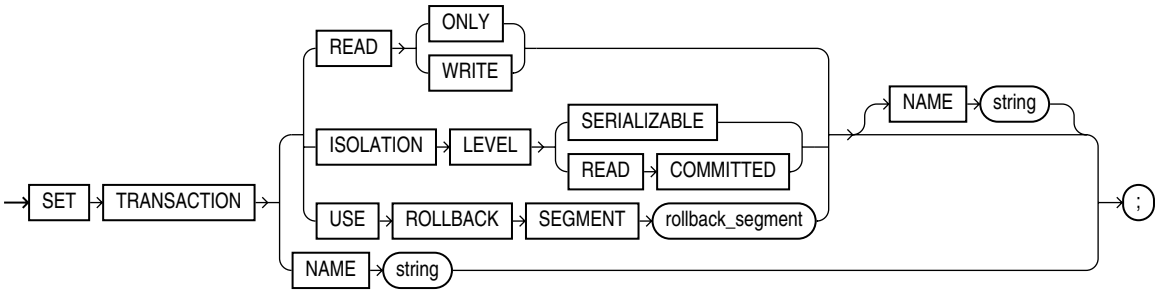
> **参照：** 13-44 ページの「COMMIT」および 18-92 ページの「ROLLBACK」を参照してください。

前提条件

SET TRANSACTION 文を使用する場合、トランザクションの先頭に記述する必要があります。ただし、SET TRANSACTION 文が必要ないトランザクションもあります。

構文

***set_transaction*::=**



セマンティクス

READ ONLY

READ ONLY 句を使用すると、現行のトランザクションを読取り専用に設定できます。この句では、**トランザクション・レベルの読取り一貫性**を設定します。

そのトランザクションの後続のすべての問合せでは、トランザクションの開始前にコミットされた変更のみが参照されます。読取り専用トランザクションは、他のユーザーが更新中の 1 つ以上の表に対して、複数の問合せを実行するレポートに役立ちます。

ユーザー SYS は、この句を使用できません。SYS による問合せでは、SYS がトランザクションを READ ONLY に設定した場合でも、トランザクション中の変更が戻されます。

読取り専用トランザクションの制限事項： 読取り専用トランザクションは、次の文のみを使用できます。

- 副問合せ（*for_update_clause* を指定しない SELECT 文）
- LOCK TABLE
- SET ROLE
- ALTER SESSION
- ALTER SYSTEM

READ WRITE

READ WRITE を指定すると、現行のトランザクションを読み書き両用に設定できます。この句では、**文レベルの読取り一貫性**を設定します。これはデフォルトです。

読み書き両用トランザクションの制限事項： 同一トランザクション内では、読取り一貫性のレベル（トランザクション・レベルおよび文レベル）を切り替えることができません。

ISOLATION LEVEL 句

ISOLATION LEVEL 句を使用すると、データベースを変更するトランザクションがどのように処理されるかを指定できます。

- SQL92 規格に定義されているシリアライズ可能トランザクション分離モードを設定する場合に、SERIALIZABLE を指定します。シリアライズ可能トランザクションにデータ操作言語（DML）が含まれている場合、その DML がシリアライズ可能トランザクションの開始時にコミットされなかったトランザクション内の更新済のリソースを更新しようとすると、その DML 文は正常に実行されません。
- Oracle Database のトランザクションでは、デフォルトで READ COMMITTED が設定されています。別のトランザクションで行ロックを保持しておく必要がある DML がトランザクションに指定されていると、DML 文は行ロックが解除されるまで待ち状態になります。

USE ROLLBACK SEGMENT 句

> **注意：** この句は、UNDO 用のロールバック・セグメントを使用している場合にのみ有効です。自動 UNDO 管理モードで UNDO 領域を管理することをお薦めします。データベースを自動 UNDO モードで実行すると、この句は無視されます。

USE ROLLBACK SEGMENT を指定すると、現行のトランザクションを、指定したロールバック・セグメントに割り当てることができます。この句によって、現行のトランザクションは暗黙的に読み書き両用トランザクションに設定されます。

パラレル DML では、複数のロールバック・セグメントが必要です。したがって、トランザクションにパラレル DML 操作が含まれている場合、この句は無視されます。

NAME 句

NAME 句を使用すると、現行のトランザクションに名前を割り当てることができます。この句は、分散データベース環境でインダウト・トランザクションを識別および変換する場合に便利です。*string* 値の最大長は 255 バイトです。

分散トランザクションに対して名前を指定する場合、トランザクションのコミット時に、名前はコミットのコメントとなり、COMMIT 文で明示的に指定した任意のコメントを上書きします。

> **参照：** トランザクションの名前指定の詳細は、『Oracle Database 概要』を参照してください。

例

トランザクションの設定例： 次の文は、サンプルの注文入力スキーマ（oe）のトロントの倉庫にある在庫の製品と量を計算するもので、毎月の最終日の真夜中に実行されます。このレポートは、別の倉庫の在庫を追加および削除する他のユーザーの影響は受けません。

```
COMMIT;

SET TRANSACTION READ ONLY NAME 'Toronto';

SELECT product_id, quantity_on_hand FROM inventories
   WHERE warehouse_id = 5
   ORDER BY product_id;

COMMIT;
```

最初の COMMIT 文によって、SET TRANSACTION がトランザクションの最初の文であることが保証されます。最後の COMMIT 文は、データベースに対する変更を保存するためではありません。単に、読取り専用トランザクションを終了するためのものです。

TRUNCATE CLUSTER

用途

注意： TRUNCATE CLUSTER 文はロールバックできません。

TRUNCATE CLUSTER 文を使用すると、クラスタからすべての行を削除できます。デフォルトでは、次の処理も実行されます。

- 削除された行が使用していたすべての領域（ただし、MINEXTENTS 記憶域パラメータで指定された領域は除く）の割当てが解除されます。
- NEXT 記憶域パラメータが、切捨て処理によってセグメントから最後に削除されたエクステントのサイズに設定されます。

クラスタを削除して再作成するより、TRUNCATE 文で行を削除する方が効果的です。クラスタを削除して再作成した場合、そのクラスタに依存するオブジェクトが無効になり、クラスタに対するオブジェクト権限を再度付与する必要があります。また、表の索引とクラスタを再作成し、その記憶域パラメータを再指定する必要があります。切捨ての場合、このような影響はありません。

TRUNCATE CLUSTER 文を使用すると、DELETE 文を使用してすべての行を削除するよりも迅速に削除できます。特に、クラスタに索引およびその他の依存オブジェクトが多数ある場合に有効です。

参照：

- クラスタからデータを削除する他の方法については、17-23 ページの「DELETE」および 17-32 ページの「DROP CLUSTER」を参照してください。
- 表の切捨ての詳細は、19-58 ページの「TRUNCATE TABLE」を参照してください。

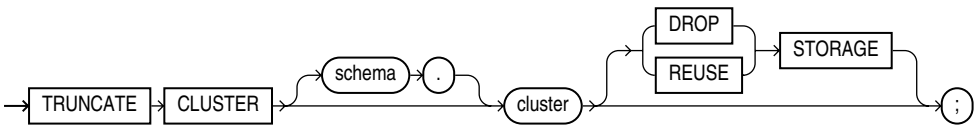
前提条件

クラスタを切り捨てるには、自分のスキーマ内にそのクラスタがあるか、または DROP ANY TABLE システム権限が必要です。

参照： 「表の切捨ての制限事項：」（19-59 ページ）

構文

***truncate_cluster*::=**



セマンティクス

CLUSTER 句

切り捨てるクラスタが設定されているスキーマと、そのクラスタの名前を指定します。なお、索引クラスタは切り捨てられますが、ハッシュ・クラスタは切り捨てられません。*schema* を指定しない場合、クラスタは自分のスキーマ内にあるとみなされます。

クラスタを切り捨てた場合、そのクラスタにある表のすべての索引データも自動的に削除されます。

STORAGE 句

STORAGE 句を使用すると、行の切捨てによって解放された領域をどのようにするかを指定できます。DROP STORAGE 句および REUSE STORAGE 句は、対応する索引から削除されたデータの空き領域にも適用されます。

DROP STORAGE DROP STORAGE を指定すると、クラスタの MINEXTENTS パラメータで割り当てられた領域を除き、クラスタから削除された行からすべての領域の割当てを解除できます。この領域は、後で表領域内の他のオブジェクトで使用できます。また、NEXT 記憶域パラメータが、切捨て処理によってセグメントから最後に削除されたエクステントのサイズに設定されます。これはデフォルトです。

REUSE STORAGE REUSE STORAGE を指定すると、クラスタに割り当てられた削除行の領域を確保できます。STORAGE の値は、表またはクラスタを作成したときの値にリセットされません。この領域は、挿入操作または更新操作によってそのクラスタ内に作成される新規データによってのみ使用されます。記憶域パラメータは現行の設定のまま残ります。

切り捨てるオブジェクトに対して複数の空きリストを指定している場合は、REUSE STORAGE 句によって、インスタンスへの空きリストのマッピングも削除され、最高水位標は第 1 エクステントの先頭までリセットされます。

例

クラスタの切捨て例 : 次の文は、personnel クラスタ内の表のすべての行を削除しますが、空き領域は表に割り当てられたままにしておきます。

```
TRUNCATE CLUSTER personnel REUSE STORAGE;
```

この文では、personnel クラスタにある表のすべての索引データも削除されます。

TRUNCATE TABLE

用途

注意： TRUNCATE TABLE 文をロールバックしたり、FLASHBACK TABLE 文を使用して、切り捨てられた表の内容を取得することはできません。

TRUNCATE TABLE を使用すると、表からすべての行を削除できます。デフォルトでは、次の処理も実行されます。

- 削除された行が使用していたすべての領域（ただし、MINEXTENTS 記憶域パラメータで指定された領域は除く）の割当てが解除されます。
- NEXT 記憶域パラメータが、切捨て処理によってセグメントから最後に削除されたエクステントのサイズに設定されます。

表を削除して再作成するより、TRUNCATE TABLE 文で行を削除する方が効果的です。表を削除して再作成した場合、その表に依存するオブジェクトが無効になり、表に対するオブジェクト権限を再度付与する必要があります。また、表の索引、整合性制約およびトリガーを再作成し、その記憶域パラメータを再指定する必要があります。切捨ての場合、このような影響はありません。

TRUNCATE TABLE 文を使用すると、DELETE 文を使用してすべての行を削除するよりも迅速に削除できます。特に、表にトリガー、索引およびその他の依存オブジェクトが多数ある場合に有効です。

参照：

- 表からデータを削除する他の方法については、17-23 ページの「DELETE」および 18-5 ページの「DROP TABLE」を参照してください。
- クラスタの切捨ての詳細は、19-56 ページの「TRUNCATE CLUSTER」を参照してください。

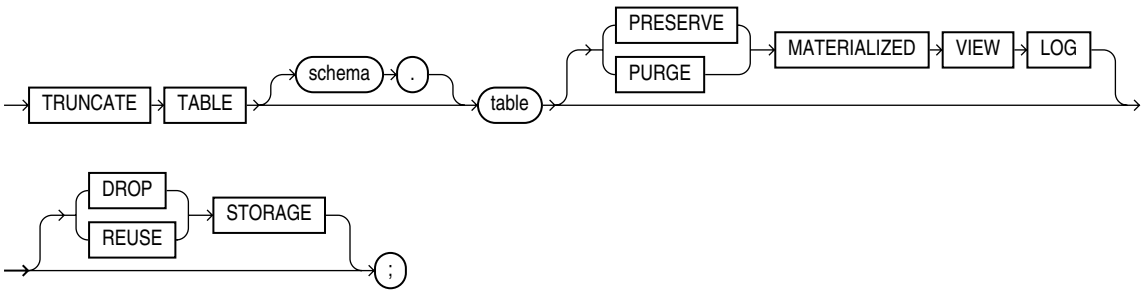
前提条件

表を切り捨てるには、自分のスキーマ内にその表があるか、または DROP ANY TABLE システム権限が必要です。

参照： 「表の切捨ての制限事項：」（19-59 ページ）

構文

***truncate_table*::=**



セマンティクス

TABLE 句

切り捨てる表が設定されているスキーマおよびその表の名前を指定します。クラスタを構成する表は、切り捨てることができません。*schema* を指定しない場合、表が自分のクラスタに定義されているとみなされます。

- 索引構成表や一時表も切り捨てることができます。一時表を切り捨てた場合、現行のセッションで作成された行のみが削除されます。
- *table* の記憶域パラメータ NEXT が、切捨て処理中にセグメントから最後に削除されたエクステントのサイズに変更されます。
- *table* に対する索引（ローカル索引のレンジ・パーティションとハッシュ・パーティション、およびローカル索引のサブパーティション）の UNUSABLE のインジケータも、自動的に切捨ておよびリセットされます。
- *table* が空でない場合は、表中の非パーティション索引およびグローバル・パーティション索引のすべてのパーティションに UNUSABLE のマークが付けられます。ただし、表が切り捨てられると索引も切り捨てられ、索引セグメントに対して新しい最高水位標が計算されます。この操作は、索引に対して新しいセグメントを作成することと同じです。このため、切捨て操作の最後に、索引が再度 USABLE になります。
- ドメイン索引の場合は、この文が、適切な TRUNCATE ルーチンを起動し、ドメイン索引のデータを切り捨てます。

  参照： ドメイン索引の詳細は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。

- 標準表および索引構成表が LOB 列を含む場合、すべての LOB データおよび LOB 索引セグメントは切り捨てられます。
- *table* がパーティション化されている場合、各パーティションまたはサブパーティションの LOB データ・セグメントおよび LOB 索引セグメントと同様に、パーティションおよびサブパーティションも切り捨てられます。

注意： 表を切り捨てた場合、表の索引データおよび表に対応付けられたマテリアライズド・ビューのダイレクト・パス・インサート情報はすべて、自動的に削除されます。この情報は、マテリアライズド・ビュー・ログのいずれにも依存していません。このダイレクト・パス・インサート情報を削除した場合、マテリアライズド・ビューの増分リフレッシュのデータが失われる場合があります。

表の切捨ての制限事項： この文には、次の制限事項があります。

- クラスタを構成する表は、個別に切り捨てることはできません。これを行うには、クラスタを切り捨てるか、表のすべての行を削除するか、または表を削除して再作成する必要があります。
- 使用可能になっている外部キー制約の親である表は、切り捨てることはできません。その表を切り捨てる場合、制約を無効にしておく必要があります。整合性制約が自己参照型の場合は、例外として表を切り捨てることができます。
- *table* でドメイン索引が定義されている場合、索引および索引パーティションに IN_PROGRESS のマークを付けることはできません。
- 参照パーティション表の親表は切り捨てることができません。まず、参照パーティション表の子表を削除する必要があります。

MATERIALIZED VIEW LOG 句

MATERIALIZED VIEW LOG 句を使用すると、表が切り捨てられた場合に、この表に定義されているマテリアライズド・ビュー・ログを保存するか、または削除するかを指定できます。この句を使用した場合、マテリアライズド・ビューのマスター表を、エクスポートまたはインポートによって再編成できます。この場合、マスター表で定義された主キー・マテリアライズド・ビューを高速リフレッシュする機能は影響を受けません。主キー・マテリアライズド・ビューの連続高速リフレッシュをサポートする場合、マテリアライズド・ビュー・ログに主キー情報を記録する必要があります。

注意： 下位互換性を保つために、MATERIALIZED VIEW のかわりにキーワード SNAPSHOT もサポートされています。

PRESERVE PRESERVE を指定すると、マスター表を切り捨てたときにマテリアライズド・ビュー・ログを保存できます。これはデフォルトです。

PURGE PURGE を指定すると、マスター表を切り捨てたときにマテリアライズド・ビュー・ログを削除できます。

> **参照：** マテリアライズド・ビュー・ログおよび TRUNCATE 文の詳細は、『Oracle Database アドバンスト・レプリケーション』を参照してください。

STORAGE 句

STORAGE 句を使用すると、行の切捨てによって解放された領域をどのようにするかを指定できます。DROP STORAGE 句および REUSE STORAGE 句は、対応する索引から削除されたデータの空き領域にも適用されます。

DROP STORAGE DROP STORAGE を指定すると、表またはクラスタの MINEXTENTS パラメータで割り当てられた領域を除き、表から削除された行からすべての領域の割当てを解除できます。この領域は、後で表領域内の他のオブジェクトで使用できます。また、NEXT 記憶域パラメータが、切捨て処理によってセグメントから最後に削除されたエクステントのサイズに設定されます。これはデフォルトです。

REUSE STORAGE REUSE STORAGE を指定すると、表に割り当てられた削除行の領域を確保できます。STORAGE の値は、表またはクラスタを作成したときの値にリセットされません。この領域は、挿入操作または更新操作によってその表またはクラスタ内に作成される新規データによってのみ使用されます。記憶域パラメータは現行の設定のまま残ります。

切り捨てるオブジェクトに対して複数の空きリストを指定している場合は、REUSE STORAGE 句によって、インスタンスへの空きリストのマッピングも削除され、最高水位標は第 1 エクステントの先頭までリセットされます。

例

表の切捨て例： 次の文は、サンプル表 hr.employees の仮想コピーのすべての行を削除して、解放された領域を employees 表が定義されている表領域に戻します。

```
TRUNCATE TABLE employees_demo;
```

ここでは、employees 表の索引データもすべて削除され、解放された領域は、それらの索引が定義されていた表領域に戻されます。

切捨て後のマテリアライズド・ビュー・ログの保存例 : 次の文は、マテリアライズド・ビュー・ログを保存する TRUNCATE 文の使用例です。

```
TRUNCATE TABLE sales_demo PRESERVE MATERIALIZED VIEW LOG;

TRUNCATE TABLE orders_demo;
```

UPDATE

用途

UPDATE 文を使用すると、表、ビューの実表またはマテリアライズド・ビューのマスター表の既存の値を変更できます。

前提条件

表の値を更新する場合は、表が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、その表に対する UPDATE オブジェクト権限が必要です。

ビューの実表の値を更新する場合は、次の条件を満たす必要があります。

- そのビューに対する UPDATE オブジェクト権限を持っている。
- そのビューが含まれているスキーマの所有者が、実表に対する UPDATE オブジェクト権限を持っている。

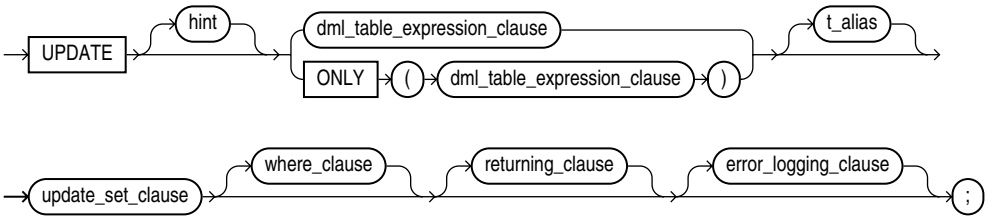
UPDATE ANY TABLE システム権限を持っている場合は、任意の表またはビューの実表の値を更新できます。

次の場合は、更新するオブジェクトに対する SELECT オブジェクト権限も必要です。

- オブジェクトがリモート・データベースに存在する。
- SQL92_SECURITY 初期化パラメータが TRUE に設定され、UPDATE 操作が表の列（*where_clause* 内の列など）を参照する。

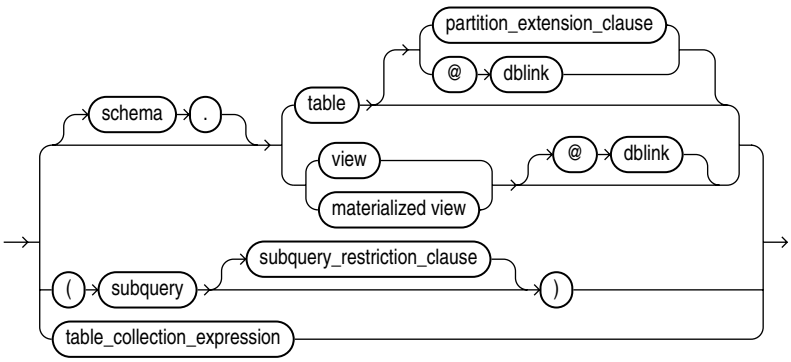
構文

***update*::=**



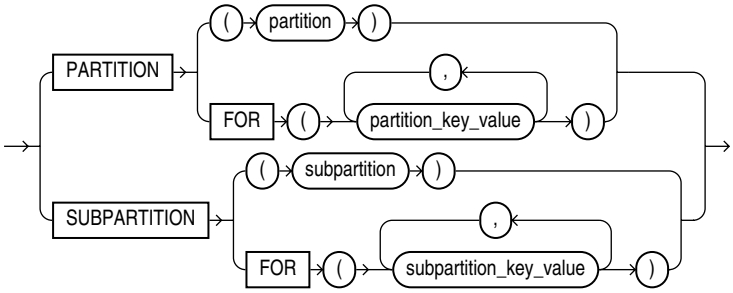
（19-62 ページの DML_table_expression_clause::=、19-63 ページの update_set_clause::=、19-63 ページの where_clause::=、19-63 ページの returning_clause::=、19-64 ページの error_logging_clause::= を参照）

DML_table_expression_clause::=

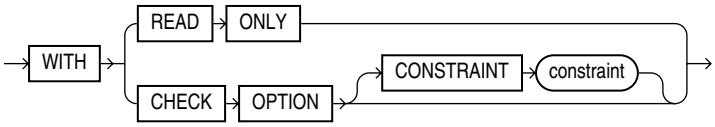


（19-63 ページの partition_extension_clause::=、19-5 ページの「SELECT」構文の項にある subquery::=、19-63 ページの subquery_restriction_clause::=、19-63 ページの table_collection_expression::= を参照）

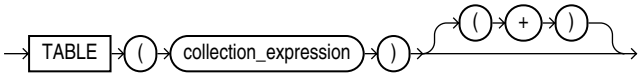
***partition_extension_clause*::=**



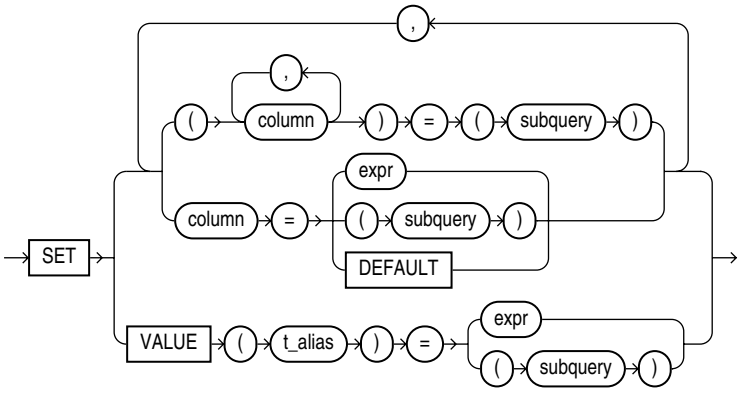
subquery_restriction_clause::=



table_collection_expression::=

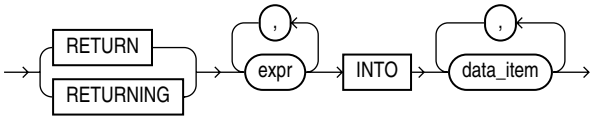


update_set_clause::=

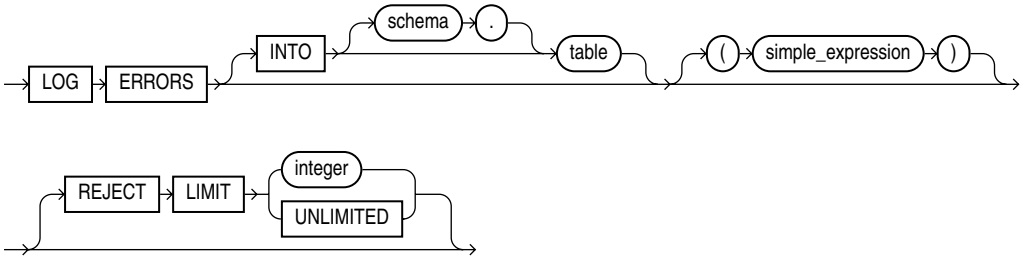


where_clause::=

returning_clause::=



error_logging_clause::=



セマンティクス

hint

文の実行計画を選択する場合に、オプティマイザに指示を与えるためのコメントを指定します。

UPDATE キーワードの直後にパラレル・ヒントを指定した場合、基礎となるスキャンおよび UPDATE 操作の両方をパラレル化できます。

参照 :

- ヒントの構文および説明については、2-70 ページの「ヒントの使用方法」を参照してください。
- パラレル実行の詳細は、『Oracle Database 概要』を参照してください。

DML_table_expression_clause

ONLY 句は、ビューのみに適用されます。UPDATE 句のビューが階層に属し、そのどのサブビューの行も変更しない場合は、ONLY 構文を指定します。

参照 : 19-66 ページの「DML_table_expression_clause の制限事項 :」および 19-69 ページの「表の更新例 :」を参照してください。

schema

更新するオブジェクトが含まれているスキーマを指定します。*schema* を指定しない場合、オブジェクトは自分のスキーマ内にあるとみなされます。

table | *view* | *materialized_view* | *subquery*

更新する対象となる、表、ビュー、マテリアライズド・ビュー、または副問合せから戻された列の名前を指定します。表に対して UPDATE 文を実行した場合、その表に対応付けられた UPDATE トリガーが起動します。

- *view* を指定した場合、ビューの実表が更新されます。ビューを定義する問合せに次のいずれかの要素が含まれる場合は、INSTEAD OF トリガーを除き、そのビューを更新することはできません。

  集合演算子
  DISTINCT 演算子
  集計ファンクションまたは分析ファンクション
  GROUP BY、ORDER BY、MODEL、CONNECT BY または START WITH 句
  SELECT 構文のリストにあるコレクション式
  SELECT 構文のリストにある副問合せ
  WITH READ ONLY が指定された副問合せ
  結合（一部の例外を除く。詳細は、『Oracle Database 管理者ガイド』を参照してください。）

- ビューから、複数の実表は更新できません。
- また、WITH CHECK OPTION を指定してビューを作成した場合、実行結果がビューを定義する問合せの条件を満たす場合にのみ、ビューを更新できます。
- *table* または *view* の実表に、1 つ以上のドメイン索引列がある場合は、この文によって適切な索引タイプの更新ルーチンが実行されます。
- 読取り専用のマテリアライズド・ビューの行は更新できません。書込み可能なマテリアライズド・ビューの行を更新すると、基礎となるコンテナ表の行も更新されます。ただし、その更新内容は次のリフレッシュ操作によって上書きされます。マテリアライズド・ビュー・グループ内の更新可能なマテリアライズド・ビューの行を更新すると、マスター表の対応する行も更新されます。

参照 :

- 索引タイプの更新ルーチンの詳細は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。
- 更新可能なマテリアライズド・ビューについては、15-4 ページの「CREATE MATERIALIZED VIEW」を参照してください。

partition_extension_clause

更新対象の *table* 内にあるパーティションまたはサブパーティションの名前またはパーティション・キー値を指定します。パーティション表内の値を更新する場合は、パーティション名を指定する必要はありません。ただし、パーティション名を指定した方が、複雑な *where_clause* を使用するよりも効果的な場合もあります。

参照 : 2-106 ページの「パーティション表と索引の参照」および 19-70 ページの「パーティションの更新例 :」を参照してください。

dblink

オブジェクトが格納されているリモート・データベースへのデータベース・リンクの完全名または部分名を指定します。Oracle Database の分散機能を使用している場合にかぎり、データベース・リンクを使用してリモート・オブジェクトを更新できます。

dblink を省略した場合、オブジェクトがローカル・データベース上にあるとみなされます。

参照 : データベース・リンクの参照方法の詳細は、2-104 ページの「リモート・データベース内のオブジェクトの参照」を参照してください。

subquery_restriction_clause

subquery_restriction_clause を使用すると、次のいずれかの方法で副問合せを制限できます。

WITH READ ONLY WITH READ ONLY を指定すると、表またはビューを更新禁止にできます。

WITH CHECK OPTION WITH CHECK OPTION を指定すると、副問合せに含まれない行を生成する表またはビューの変更を禁止できます。この句を DML 文の副問合せ内で使用する場合、FROM 句内の副問合せには指定できますが、WHERE 句内の副問合せには指定できません。

CONSTRAINT *constraint* CHECK OPTION 制約の名前を指定します。この識別子を省略した場合、その制約に SYS_C*n* という形式の名前が自動的に割り当てられます。この場合の n は、その制約名をデータベース内で一意の名前にする整数です。

参照 : 「WITH CHECK OPTION 句の使用例 :」(19-39 ページ)

table_collection_expression

table_collection_expression を使用すると、問合せおよび DML 操作で、*collection_expression* 値を表として扱うことができます。*collection_expression* には、副問合せ、列、ファンクションまたはコレクション・コンストラクタのいずれかを指定できます。その形式にかかわらず、集合値（ネストした表型または VARRAY 型の値）を戻す必要があります。このようなコレクションの要素抽出プロセスを**コレクション・ネスト解除**といいます。

TABLE 式を親表と結合する場合は、オプションのプラス（+）には大きな意味があります。+ を指定すると、その 2 つの外部結合が作成され、コレクション式が NULL の場合でも、外部表の行が問合せで戻されるようになります。

注意： 以前のリリースの Oracle では、*collection_expression* が副問合せの場合、*table_collection_expression* を THE *subquery* と表現していました。現在、このような表現方法は非推奨になっています。

---

table_collection_expression を使用して、ある表の行を別の表の行を基にして更新できます。たとえば、四半期ごとの売上表を、年度ごとの売上表にまとめることができます。

t_alias

文中で参照する表、ビューまたは副問合せの**相関名**（別名）を指定します。*DML_table_expression_clause* がいずれかのオブジェクト型属性またはオブジェクト型メソッドを参照する場合、この別名が必要です。

> **参照：** 「相関更新の例 :」（19-70 ページ）

***DML_table_expression_clause* の制限事項：** この句には、次の制限事項があります。

- *table* または *view* の実表に、IN_PROGRESS または FAILED とマークされたドメイン索引がある場合は、この文は実行できません。
- 関係する索引パーティションが UNUSABLE とマークされている場合は、パーティションに挿入できません。
- *DML_table_expression_clause* の副問合せには *order_by_clause* を指定できません。
- UNUSABLE のマークが付いている索引、索引パーティションまたは索引サブパーティションを指定する場合、SKIP_UNUSABLE_INDEXES セッション・パラメータが TRUE に設定されていないかぎり、UPDATE 文は正常に実行されません。

> **参照：** SKIP_UNUSABLE_INDEXES セッション・パラメータの詳細は、11-41 ページの「ALTER SESSION」を参照してください。

update_set_clause

update_set_clause を使用すると、列の値を設定できます。

column

更新するオブジェクトの列の名前を指定します。*update_set_clause* に表の列を指定しない場合、その列の値は変更されません。

column が LOB オブジェクト属性を参照している場合、まず空または NULL の値で初期化する必要があります。リテラルで更新はできません。また、UPDATE 以外の SQL 文を使用して LOB 値を更新する場合は、LOB を含む行を最初にロックしておく必要があります。詳細は、19-30 ページの「for_update_clause」を参照してください。

column が仮想列である場合、ここで指定することはできません。この場合、仮想列の導出元となっている値を更新する必要があります。

column がパーティション表のパーティション化キーに含まれる場合、別のパーティションまたはサブパーティションに行を移動する列の値を変更すると、行の移動を有効にしないかぎり、UPDATE は正常に実行されません。16-6 ページの「CREATE TABLE」の「*row_movement_clause*」または 12-2 ページの「ALTER TABLE」を参照してください。

また、*column* がリスト・パーティション表のパーティション化キーの一部である場合、パーティションの *partition_value* リストに存在していない列の値を指定すると、UPDATE は正常に実行されません。

subquery

更新される行ごとに 1 行ずつ戻す副問合せを指定します。

- *update_set_clause* で 1 列のみを指定した場合、副問合せは 1 つの値のみを戻します。
- *update_set_clause* で複数の列を指定した場合、副問合せは指定した列の数の値を戻します。
- 副問合せが行を戻さなかった場合は、列には NULL が割り当てられます。
- *subquery* がリモート・オブジェクトを参照する場合、参照がローカル・データベースのオブジェクトにループバックしないかぎり、UPDATE はパラレルで実行されます。ただし、*DML_table_expression_clause* の *subquery* がリモート・オブジェクトを参照する場合は、UPDATE はシリアルで実行されます。

副問合せ内で *flashback_query_clause* を使用すると、過去のデータで *table* を更新できます。この句の詳細は、19-15 ページの「SELECT」の「flashback_query_clause」を参照してください。

参照 :

- 19-4 ページの「SELECT」および 9-13 ページの「副問合せの使用方法」を参照してください。
- 16-57 ページの「CREATE TABLE」の「parallel_clause」を参照してください。

expr

対応する列に割り当てられた新しい値に変換する式を指定します。

参照 : *expr* の構文については、第 6 章「式」および 19-70 ページの「オブジェクト表の更新例 :」を参照してください。

DEFAULT DEFAULT を指定すると、以前に列のデフォルト値として指定した値を列に設定できます。対応する列に対してデフォルト値を指定していない場合、列に NULL が設定されます。

デフォルト値への更新の制限事項 : ビューを更新する場合は、DEFAULT を指定できません。

VALUE 句

VALUE 句を使用すると、オブジェクト表の行全体を指定できます。

VALUE 句の制限事項 : この句は、オブジェクト表に対してのみ指定できます。

注意 : RAW 列に文字列リテラルを挿入する場合、後続の問合せ中に RAW 列にある索引は使用されずに、全表スキャンが行われます。

参照 : 「オブジェクト表の更新例 :」(19-70 ページ)

where_clause

where_clause を使用すると、指定した条件が真の行のみが更新されるように制限できます。この句を指定しない場合、表またはビューのすべての行が更新されます。*condition* の構文は、第 7 章「条件」を参照してください。

where_clause は、値を更新する行を決定します。*where_clause* を指定しない場合、すべての行が更新されます。*where_clause* の条件を満たす行ごとに、*update_set_clause* の等号演算子（=）の左側にある列に、演算子の対応する右側の式の値が設定されます。式は行が更新される場合に評価されます。

returning_clause

この句を使用すると、DML 文に影響される行を取り出すことができます。この句は、表、マテリアライズド・ビュー、および単一の実表を持つビューに指定できます。

returning_clause を指定した DML 文を単一行に実行すると、影響された行、ROWID、および処理された行への REF を使用している列式が取り出され、ホスト変数または PL/SQL 変数に格納されます。

returning_clause を指定した DML 文を複数行に実行すると、式の値、ROWID および処理された行に関連する REF がバインド配列に格納されます。

expr *expr* リストの各項目は、適切な構文で表す必要があります。

INTO INTO 句を指定すると、変更された行の値を、*data_item* リストに指定する変数に格納できます。

data_item 取り出された *expr* 値を格納するホスト変数または PL/SQL 変数を指定します。

RETURNING リストの各式については、INTO リストに、対応する型に互換性がある PL/SQL 変数またはホスト変数を指定する必要があります。

RETURNING 句の制限事項： RETURNING 句には、次の制限事項があります。

- *expr* に次の制限事項があります。
  - UPDATE 文および DELETE 文の場合、各 *expr* は、単純式または単一セットの集計ファンクション式である必要があります。1 つの *returning_clause* 内に単純式と単一セットの集計ファンクション式を混在させることはできません。INSERT 文の場合、各 *expr* は単純式である必要があります。INSERT 文の RETURNING 句では、集計ファンクションはサポートされていません。
  - 単一セットの集計ファンクション式を DISTINCT キーワードに含めることはできません。
- *expr* リストに主キー列またはその他の NOT NULL 列が含まれている場合、表に BEFORE UPDATE トリガーが定義されていると、UPDATE 文は正常に実行されません。
- マルチテーブル・インサートでは *returning_clause* を指定できません。
- パラレル DML またはリモート・オブジェクトにはこの句を使用できません。
- LONG 型を取り出すことはできません。
- INSTEAD OF トリガーが定義されたビューに対して指定することはできません。

> **参照：** BULK COLLECT 句を使用してコレクション変数に複数の値を戻す場合は、『Oracle Database PL/SQL 言語リファレンス』を参照してください。

error_logging_clause

error_logging_clause の UPDATE 文での動作は、INSERT 文の場合と同じです。詳細は、18-63 ページの INSERT 文の「error_logging_clause」を参照してください。

参照：「エラー・ロギングによる表への挿入例 :」（18-64 ページ）

例

表の更新例： 次の文は、職種が SH_CLERK のすべての従業員の歩合に NULL 値を指定します。

```
UPDATE employees
   SET commission_pct = NULL
   WHERE job_id = 'SH_CLERK';
```

次の文は、Douglas Grant を部門 20 の管理者に昇格させ、給与を 1,000 ドル引き上げます。

```
UPDATE employees SET
    job_id = 'SA_MAN', salary = salary + 1000, department_id = 120
    WHERE first_name||' '||last_name = 'Douglas Grant';
```

次の文は、remote データベースの employees 表の従業員の給与を増加します。

```
UPDATE employees@remote
   SET salary = salary*1.1
   WHERE last_name = 'Baer';
```

次の例は、UPDATE 文の次の構文要素を示します。

- 単一文にまとめた *update_set_clause* の 2 つの形式
- 相関副問合せ
- 更新された行を制限する *where_clause*

```
UPDATE employees a
    SET department_id =
        (SELECT department_id
            FROM departments
            WHERE location_id = '2100'),
        (salary, commission_pct) =
        (SELECT 1.1*AVG(salary), 1.5*AVG(commission_pct)
          FROM employees b
          WHERE a.department_id = b.department_id)
    WHERE department_id IN
        (SELECT department_id
          FROM departments
          WHERE location_id = 2900
              OR location_id = 2700);
```

この UPDATE 文によって、次の処理が実行されます。

- ジュネーブまたはミュンヘン（location_id は 2900 または 2700）で働く従業員のみを更新します。
- これらの従業員の department_id をボンベイ（location_id は 2100）の対応する department_id に設定します。
- 各従業員の給与を、その部門の平均給与の 110% に上げます。
- 各従業員の歩合を、その部門の平均歩合の 150% に上げます。

パーティションの更新例： 次の文は、sales 表の 1 つのパーティションの値を更新します。

```
UPDATE sales PARTITION (sales_q1_1999) s
   SET s.promo_id = 494
   WHERE amount_sold > 1000;
```

オブジェクト表の更新例： 次の文は、2 つのオブジェクト表 people_demo1 および people_demo2 を作成します。ここで使用する people_typ オブジェクトは、19-44 ページの「表のコレクション例：」で作成したものです。この例では、people_demo2 から行を選択して people_demo1 の行を更新する方法を示します。

```
CREATE TABLE people_demo1 OF people_typ;

CREATE TABLE people_demo2 OF people_typ;

UPDATE people_demo1 p SET VALUE(p) =
   (SELECT VALUE(q) FROM people_demo2 q
    WHERE p.department_id = q.department_id)
   WHERE p.department_id = 10;
```

この例では、SET 句と副問合せの両方で、VALUE オブジェクト参照ファンクションを使用します。

相関更新の例： 相関副問合せを使用してネストした表の行を更新する例は、19-44 ページの「表のコレクション例：」を参照してください。

UPDATE 操作中に RETURNING 句を使用する例： 次の文は、更新された行の値を戻し、PL/SQL 変数 bnd1、bnd2、bnd3 に結果を格納します。

```
UPDATE employees
  SET job_id ='SA_MAN', salary = salary + 1000, department_id = 140
  WHERE last_name = 'Jones'
  RETURNING salary*0.25, last_name, department_id
    INTO :bnd1, :bnd2, :bnd3;
```

次の文は、RETURNING 句の式で単一セットの集計ファンクションを指定できることを示します。

```
UPDATE employees
   SET salary = salary * 1.1
   WHERE department_id = 100
   RETURNING SUM(salary) INTO :bnd1;
```

A

構文図の読み方

この付録では、構文図の読み方について説明します。

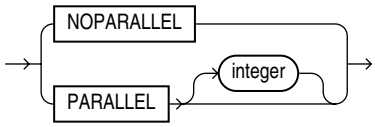
図形構文図

構文図とは、SQL の有効な構文を図で示したものです。構文図は、矢印が示す方向に左から右へ読んでください。

コマンドおよびキーワードは、四角形の中に大文字で書かれています。コマンドおよびキーワードは、四角形の中に書かれているとおりに指定してください。パラメータは、楕円形の中に小文字で書かれています。パラメータには変数を指定します。句読点、デリミタ、終了記号は円の中に書かれています。

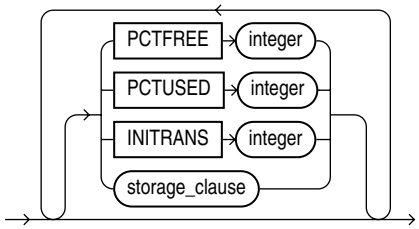
構文図の中にパスが複数ある場合は、ユーザーがパスを選択できます。次の例では、NOPARALLEL または PARALLEL のいずれかを指定できます。

***parallel_clause*::=**



キーワード、演算子、パラメータに複数の選択肢がある場合は、選択できるオプションが縦に並べて書かれています。次の構文図では、スタックにある 4 つのパラメータから 1 つ以上を指定することができます。

***physical_attributes_clause*::=**



構文図で使用されるパラメータと、各文でそのパラメータに代入する値の例を次の表に示します。

表 A-1 構文のパラメータ

パラメータ	説明	例
table	パラメータによって指定された型のオブジェクト名で置き換える必要があります。オブジェクト型の一覧は、2-97 ページの「スキーマ・オブジェクト」を参照してください。	employees
c	ご使用のデータベース・キャラクタ・セットの単一文字で置き換える必要があります。	T s
'text'	一重引用符で囲んだテキスト文字列で置き換える必要があります。*'text'* の構文は、2-44 ページの「テキスト・リテラル」を参照してください。	'Employee records'
char	CHAR または VARCHAR2 データ型の式か、一重引用符で囲んだ文字リテラルで置き換える必要があります。	last_name 'Smith'
condition	TRUE または FALSE に評価される条件で置き換える必要があります。*condition* の構文は、第 7 章「条件」を参照してください。	last_name >'A'

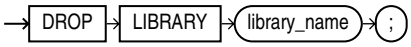
表 A-1 構文のパラメータ（続き）

パラメータ	説明	例
date *d*	日付定数または DATE データ型の式で置き換える必要があります。	TO_DATE('01-Jan-2002', 'DD-MON-YYYY')
expr	6-2 ページの「SQL 式」にある *expr* の構文の説明で定義されている任意のデータ型の式で置き換えることができます。	salary + 1000
integer	2-46 ページの「整数リテラル」にある integer の構文の説明で定義されている整数で置き換える必要があります。	72
number *m* *n*	NUMBER データ型の式、または 2-46 ページの「数値リテラル」にある *number* の構文の説明で定義されている数値定数で置き換える必要があります。	AVG(salary) 15 * 7
raw	RAW データ型の式で置き換える必要があります。	HEXTORAW('7D')
subquery	SELECT 文で置き換える必要があります。この SELECT 文は、別の SQL 文中で使用されます。19-4 ページの「SELECT」を参照してください。	SELECT last_name FROM employees
db_name	埋込み SQL プログラム内のデフォルト以外のデータベース名で置き換える必要があります。	sales_db
db_string	Oracle Net データベース接続のデータベース識別文字列で置き換える必要があります。詳細は、ご使用の Oracle Net プロトコルのユーザーズ・ガイドを参照してください。	—

必須キーワードとパラメータ

必須キーワードおよびパラメータは、単独で示されているか、または選択肢のリストとして縦に並べて書かれています。必須キーワードおよびパラメータが 1 つの場合は、メイン・パス、つまり現在選択しているパスの線上に書かれています。次の例では、*library_name* が必須パラメータです。

***drop_library*::=**

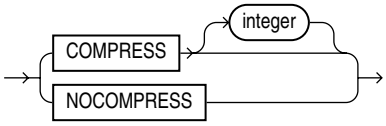


HQ_LIB という名前のライブラリがあるとすると、この図は、次の文を示しています。

```
DROP LIBRARY hq_lib;
```

メイン・パスに交わる縦に並んだリストの中に、複数のキーワードまたはパラメータがある場合、そのうちの 1 つが必須です。複数のキーワードまたはパラメータから、いずれか 1 つを選択する必要があります。ただし、メイン・パスに存在しなくてもかまいません。次の例では、いずれかの設定値を選択する必要があります。

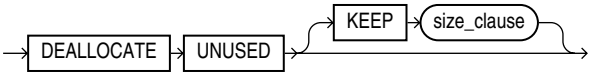
***key_compression*::=**



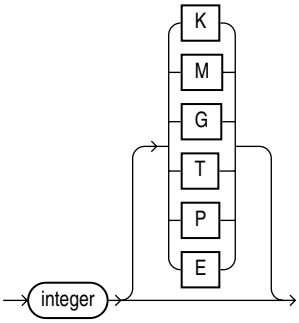
オプションのキーワードとパラメータ

キーワードおよびパラメータが、メイン・パスより上に縦に並べてリストされている場合は、それらのキーワードおよびパラメータがオプションであることを示しています。次の例では、上のパスへ進んでも、続けてメイン・パスに沿って進んでもかまいません。

***deallocate_unused_clause*::=**



***size_clause*::=**



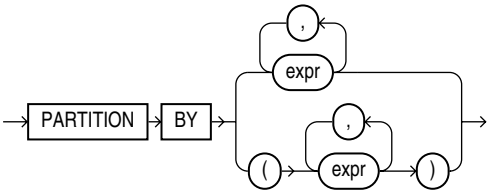
これらの図に従うと、次の文はすべて有効です。

```
DEALLOCATE UNUSED;
DEALLOCATE UNUSED KEEP 1000;
DEALLOCATE UNUSED KEEP 10G;
DEALLOCATE UNUSED 8T;
```

構文のループ

ループがある場合、ループ内の構文を何度でも繰り返して実行できます。次の例では、値の式を 1 つ選択した後、繰り返し別の式を選択できます。式と式の間はカンマで区切ります。

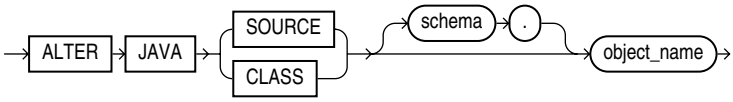
***query_partition_clause*::=**

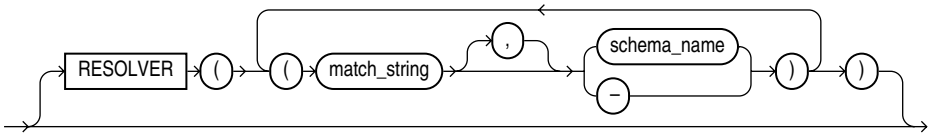


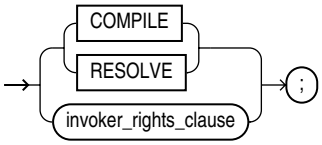
複数の部分に分割された構文図

複数の部分に分割された構文図は、すべてのメイン・パスの端と端が結合されているものとしてみてください。次の例は、3 つに分割された構文図です。

***alter_java*::=**







この図に従うと、次の文は有効です。

```
ALTER JAVA SOURCE jsource_1 COMPILE;
```

データベース・オブジェクト

表や列などの Oracle 識別子名は、30 文字以内に制限されています。先頭文字は英字である必要がありますが、それ以外は、英字、数字、ドル記号（$）、シャープ記号（#）およびアンダースコア（_）を任意に組み合せて指定できます。

ただし、Oracle の識別子を二重引用符（"）で囲むと、一重引用符以外の空白を含むすべての有効な文字を組み合せて指定できます。Oracle の識別子は、二重引用符で囲んだ場合を除いて、大 / 小文字が区別されません。

> **参照：** 詳細は、2-98 ページの「スキーマ・オブジェクトのネーミング規則」を参照してください。

B

Oracle と標準 SQL

この付録では、SQL:2003 規格への Oracle の規格準拠について説明します。SQL:2003 規格の必須部分は、Core SQL:2003 として知られ、SQL:2003 の Part 2「Foundation」および Part 11「Schemata」に記載されています。基礎的な機能は、Part 2 の Annex F の「Feature taxonomy and definition for mandatory features of SQL/Foundation」表で分析されています。スキーマ機能は、Part 11 の Annex E の「Feature taxonomy and definition for mandatory features of SQL/Schemata」表で分析されています。

この付録では、ANSI（米国規格協会）および ISO（国際標準化機構）によって確立された SQL 規格への Oracle の規格準拠について説明します（ANSI および ISO の SQL 規格は同一です）。

この付録では、次の内容を説明します。

- ANSI 規格
- ISO 規格
- Core SQL:2003 に対する Oracle の準拠
- SQL/Foundation:2003 のオプション機能に対する Oracle のサポート
- SQL/CLI:2003 に対する Oracle の準拠
- SQL/PSM:2003 に対する Oracle の準拠
- SQL/MED:2003 に対する Oracle の準拠
- SQL/OLB:2003 に対する Oracle の準拠
- SQL/XML:2006 に対する Oracle の準拠
- FIPS 127-2 に対する Oracle の準拠
- 標準 SQL に対する Oracle 拡張機能
- 以前の規格に対する Oracle の準拠
- キャラクタ・セットのサポート

ANSI 規格

SQL に関連する ANSI のドキュメントは、次のとおりです。

- ANSI/ISO/IEC 9075-1:2003, Information technology--Database languages--SQL--Part 1: Framework (SQL/Framework)
- ANSI/ISO/IEC 9075-2:2003, Information technology--Database languages--SQL--Part 2: Foundation (SQL/Foundation)
- ANSI/ISO/IEC 9075-3:2003, Information technology--Database languages--SQL--Part 3: Call-Level Interface (SQL/CLI)
- ANSI/ISO/IEC 9075-4:2003, Information technology--Database languages--SQL--Part 4: Persistent Stored Modules (SQL/PSM)
- ANSI/ISO/IEC 9075-9:2003, Information technology--Database languages--SQL--Part 9: Management of External Data (SQL/MED)
- ANSI/ISO/IEC 9075-10:2003, Information technology--Database languages--SQL--Part 10: Object Language Bindings (SQL/OLB)
- ANSI/ISO/IEC 9075-11:2003, Information technology--Database languages--SQL--Part 11: Information and Definition Schemas (SQL/Schemata)
- ANSI/ISO/IEC 9075-13:2003, Information technology--Database languages--SQL--Part 13: SQL Routines and Types using the Java Programming Language (SQL/JRT)
- ANSI/ISO/IEC 9075-14:2005, Information technology--Database languages--SQL--Part 14: XML-Related Specifications (SQL/XML)

これらの規格は、次の項に示す対応する ISO 規格と同一です。

ANSI 規格のコピーについては、次の宛先に申し込んでください。

American National Standards Institute
25 West 43rd Street
New York, NY 10036 USA
電話番号 : +1.212.642.4900
FAX 番号 : +1.212.398.0023

規格のコピーは、次の Web サイトからも入手できます。

```
http://webstore.ansi.org/ansidocstore/default.asp
```

SQL 規格を含む ANSI 規格のサブセットは、INCITS 規格です。これらは、INCITS（InterNational Committee for Information Technology Standards）から入手できます。URL は、次のとおりです。

```
http://www.incits.org/
```

ISO 規格

SQL に関連する ISO のドキュメントは、次のとおりです。

- ISO/IEC 9075-1:2003, Information technology--Database languages--SQL--Part 1: Framework (SQL/Framework)
- ISO/IEC 9075-2:2003, Information technology--Database languages--SQL--Part 2: Foundation (SQL/Foundation)
- ISO/IEC 9075-3:2003, Information technology--Database languages--SQL--Part 3: Call-Level Interface (SQL/CLI)

- ISO/IEC 9075-4:2003, Information technology--Database languages--SQL--Part 4: Persistent Stored Modules (SQL/PSM)
- ISO/IEC 9075-9:2003, Information technology--Database languages--SQL--Part 9: Management of External Data (SQL/MED)
- ISO/IEC 9075-10:2003, Information technology--Database languages--SQL--Part 10: Object Language Bindings (SQL/OLB)
- ISO/IEC 9075-11:2003, Information technology--Database languages--SQL--Part 11: Information and Definition Schemas (SQL/Schemata)
- ISO/IEC 9075-13:2003, Information technology--Database languages--SQL--Part 13: SQL Routines and Types using the Java Programming Language (SQL/JRT)
- ISO/IEC 9075-14:2005, Information technology--Database languages--SQL--Part 14: XML-Related Specifications (SQL/XML)

ISO 規格のコピーについては、次の宛先に申し込んでください。

International Organization for Standardization
1 Rue de Varembe
Case postale 56
CH-1211, Geneva 20, Switzerland
電話番号 : +41.22.749.0111
FAX 番号 : +41.22.733.3430
URL: http://www.iso.ch/

Web ストアからも入手できます。URL は、次のとおりです。

```
http://www.iso.ch/iso/en/prods-services/ISOstore/store.html
```

Core SQL:2003 に対する Oracle の準拠

ANSI および ISO の SQL 規格では、規格準拠性について明示する箇所に、準拠の種類および実装されている機能について記載することを義務付けています。規格準拠性についての最小限の明示を Core SQL:2003 といいます。これは、規格の Part 2「SQL/Foundation」および Part 11「SQL/Schemata」に定義されています。次の製品は、表 B-1 に示す Core SQL:2003 に完全（または部分的）に準拠しています。

- Oracle Database サーバー
- Pro*C/C++ リリース 9.2.0
- Pro*COBOL リリース 9.2.0
- Pro*FORTRAN リリース 1.8.77
- SQL Module for Ada （Mod*Ada）リリース 9.2.0
- Pro*COBOL 1.8 リリース 1.8.77
- Pro*PL/I リリース 1.6.28
- OTT（Oracle Type Translator）

表 B-1 に、Oracle が完全にサポートする Core SQL:2003 機能を示します。

表 B-1 完全にサポートする Core SQL:2003 機能

機能識別番号	機能
E011	数値データ型
E031	識別子
E061	基本的な述語および検索条件
E081	基本的な権限
E091	集合ファンクション
E101	基本的なデータ操作
E111	単一行の SELECT 文
E131	NULL 値のサポート（値のかわりの NULL）
E141	基本的な整合性制約
E151	トランザクションのサポート
E152	基本的な SET TRANSACTION 文
E153	副問合せを持つ更新可能な問合せ
E161	先頭に 2 つの負の符号（-）を付けた SQL 文のコメント
E171	SQLSTATE のサポート
F041	基本的な結合表
F051	基本的な日付および時刻
F081	ビューの UNION および EXCEPT
F131	グループ操作
F181	複数モジュールのサポート
F201	CAST ファンクション
F221	明示的なデフォルト
F261	CASE 式
F311	スキーマ定義文
F471	スカラー副問合せの値
F481	拡張 NULL 述語
T631	1 つのリスト要素を使用した IN 述語

表 B-2 に、Oracle が部分的にサポートする Core SQL:2003 機能を示します。

表 B-2 部分的にサポートする Core SQL:2003 機能

機能識別子、機能	部分的なサポート
E021、文字データ型	次の副機能を完全にサポートします。 ■ E021-01、CHARACTER データ型 ■ E021-07、文字の連結 ■ E021-08、UPPER および LOWER ファンクション ■ E021-09、TRIM ファンクション ■ E021-10、文字データ型間の暗黙的な加算 ■ E021-12、文字の比較 次の副機能を部分的にサポートします。 ■ E021-02、CHARACTER VARYING データ型（Oracle は、長さ 0 の VARCHAR 列と NULL を区別しません） ■ E021-03、文字リテラル（Oracle は、長さ 0 のリテラル '' を NULL とみなします） Oracle は、次の副機能と同等の機能を持ちます。 ■ E021-04、CHARACTER_LENGTH ファンクションのかわりに、LENGTH ファンクションを使用します。 ■ E021-05、OCTET_LENGTH ファンクションのかわりに、LENGTHB ファンクションを使用します。 ■ E021-06、SUBSTRING ファンクションのかわりに、SUBSTR ファンクションを使用します。 ■ E021-11、POSITION ファンクションのかわりに、INSTR ファンクションを使用します。
E051、基本的な問合せ指定	次の副機能を完全にサポートします。 ■ E051-01、SELECT DISTINCT ■ E051-02、GROUP BY 句 ■ E051-04、<SELECT 構文のリスト > にない列を GROUP BY に含めることができます。 ■ E051-05、SELECT 構文のリスト項目の名前の変更 ■ E051-06、HAVING 句 ■ E051-07、SELECT 構文のリストでの * の修飾 次の副機能を部分的にサポートします。 ■ E051-08、FROM 句の相関名（Oracle は、相関名をサポートしますが、オプションの AS キーワードはサポートしません） 次の副機能はサポートされません。 ■ E051-09、FROM 句での列の名前の変更

表 B-2 部分的にサポートする Core SQL:2003 機能（続き）

機能識別子、機能	部分的なサポート
E071、基本的な問合せ式	次の副機能を完全にサポートします。 ■ E071-01、UNION DISTINCT 表演算子 ■ E071-02、UNION ALL 表演算子 ■ E071-05、表演算子によって結合された列の型は同じでなくてもかまいません。 ■ E071-06、副問合せの表演算子 Oracle は、次の副機能と同等の機能を持ちます。 ■ E071-03、EXCEPT DISTINCT 表演算子（EXCEPT DISTINCT のかわりに MINUS を使用します）
E121、基本的なカーソルのサポート	次の副機能を完全にサポートします。 ■ E121-01、DECLARE CURSOR ■ E121-02、ORDER BY 列は、SELECT 構文のリストになくてもかまいません。 ■ E121-03、ORDER BY 句の値の式 ■ E121-04、OPEN 文 ■ E121-06、位置付けされた UPDATE 文 ■ E121-07、位置付けされた DELETE 文 ■ E121-08、CLOSE 文 ■ E121-10、FETCH 文および暗黙的な NEXT 次の副機能を部分的にサポートします。 ■ E121-17、WITH HOLD カーソル（規格では、カーソルは ROLLBACK 中は保持されませんが、Oracle では ROLLBACK 中も保持されます。）
F031、基本的なスキーマ操作	次の副機能を完全にサポートします。 ■ F031-01、実表を作成するための CREATE TABLE 文 ■ F031-02、CREATE VIEW 文 ■ F031-03、GRANT 文 次の副機能を部分的にサポートします。 ■ F031-04、ALTER TABLE 文の ADD COLUMN 句（Oracle は、この構文のオプションのキーワード COLUMN をサポートしません） 次の副機能はサポートしません（Oracle は、キーワード RESTRICT をサポートしません）。 ■ F031-13、DROP TABLE 文の RESTRICT 句 ■ F031-16、DROP VIEW 文の RESTRICT 句 ■ F031-19、REVOKE 文の RESTRICT 句

表 B-2 部分的にサポートする Core SQL:2003 機能（続き）

機能識別子、機能	部分的なサポート
F812、基本的なフラグ付け	Oracle は、フラガーを持っていますが、SQL:2003 よりも SQL-92 に準拠しています。
T321、基本的な SQL 起動ルーチン	次の副機能を完全にサポートします。 ■ T321-03、ファンクションの起動 ■ T321-04、CALL 文 Oracle は、次の副機能を異なる構文でサポートします。 ■ T321-01、オーバーロードしないユーザー定義ファンクション ■ T321-02、オーバーロードしないユーザー定義プロシージャ Oracle の構文 CREATE FUNCTION および CREATE PROCEDURE と規格の構文の違いは、次のとおりです。 ■ 規格の構文では、パラメータのモード（IN、OUT または INOUT）はパラメータ名の前ですが、Oracle 構文では、パラメータ名の後です。 ■ 規格の構文では、INOUT を使用しますが、Oracle の構文では、IN OUT を使用します。 ■ Oracle の構文では、復帰型の後およびルーチン本体の定義の前に IS または AS が必要ですが、規格の構文では、必要ありません。 ■ ルーチンの本体が C である場合、そのルーチンに名前を付けるとき、規格の構文では、キーワード LANGUAGE C EXTERNAL NAME を使用しますが、Oracle の構文では、LANGUAGE C NAME を使用します。 ■ ルーチンの本体が SQL である場合、Oracle では、プロシージャを独自に拡張した PL/SQL を使用します。 Oracle は、次の副機能を PL/SQL ではサポートしますが、Oracle SQL ではサポートしません。 ■ T321-05、RETURN 文 Oracle は、次の副機能と同等の機能を提供します。 ■ T321-06、ROUTINES ビュー : ALL PROCEDURES メタデータ・ビューを使用します。 ■ T321-07、PARAMETERS ビュー : ALL_ARGUMENTS メタデータ・ビューおよび ALL _METHOD_PARAMS メタデータ・ビューを使用します。

Oracle には、表 B-3 で示す機能と同等の機能があります。

表 B-3 Core SQL:2003 機能と同等の機能

機能識別子、機能	同等の機能
F021、基本的な情報スキーマ	Oracle は、この機能のビューを持ちませんが、同じ情報を別のメタデータ・ビューで使用可能にします。 ■ TABLES のかわりに、ALL_TABLES を使用します。 ■ COLUMNS のかわりに、ALL_TAB_COLUMNS を使用します。 ■ VIEWS のかわりに、ALL_VIEWS を使用します。 ただし、ユーザーのビューに WITH CHECK OPTION が定義されていたり、更新可能な場合は、Oracle の ALL_VIEWS は表示されません。ビューに WITH CHECK OPTION が定義されているかどうかを確認する場合は、ALL_CONSTRAINTS を使用し、TABLE_NAME をそのビュー名にして、CONSTRAINT_TYPE が 'V' であるビューを検索します。 ■ TABLE_CONSTRAINTS、REFERENTIAL_CONSTRAINTS および CHECK_CONSTRAINTS のかわりに、ALL_CONSTRAINTS を使用します。 ただし、Oracle の ALL_CONSTRAINTS は、制約が遅延可能か初期遅延かは表示しません。
S011、固有型	固有型は、強い型指定のスカラー型です。固有型は、1 つの属性のみを持つオブジェクト型を使用して、Oracle でエミュレートできます。USER_DEFINED_TYPES と呼ばれる規格の情報スキーマ・ビューは、Oracle のメタデータ・ビュー ALL_TYPES と同じです。
T695、変換のサポート	Oracle の CONVERT ファンクションでは、複数のキャラクタ・セット間での変換が可能です。Oracle には、キャラクタ・セットの変換を追加したり、削除する機能はありません。

表 B-4 に、Oracle がサポートしない Core SQL:2003 機能を示します。

表 B-4 サポートしない Core SQL:2003 機能

機能識別番号	機能
F501	機能および準拠するビュー

注意： Oracle は、「E182、モジュール言語」をサポートしません。この機能は、「SQL/Foundation」の表 35 に示されていますが、Core にはモジュール言語と埋込み言語の選択肢があることが示されているのみです。モジュール言語および埋込み言語は、機能は同じで、SQL 文がホスト・プログラミング言語と関連付けられている点のみが異なります。Oracle は、埋込み言語をサポートします。

SQL/Foundation:2003 のオプション機能に対する Oracle のサポート

Oracle は、表 B-5 に示す SQL/Foundation:2003 のオプション機能をサポートします。

表 B-5 完全にサポートする SQL/Foundation:2003 のオプション機能

機能識別番号	機能
B011	埋込み ADA
B012	埋込み C
B013	埋込み COBOL
B014	埋込み Fortran
B021	ダイレクト SQL （Oracle では SQL*Plus）
F281	LIKE 拡張
F411	タイムゾーンの指定
F421	各国語キャラクタ
F442	集合ファンクションでの複合列の参照
F491	制約管理
F555	拡張秒精度 （Oracle は、小数点以下 9 桁までをサポートします）
F561	完全評価式
F721	遅延可能制約
F731	列 INSERT 権限
F781	自己参照操作
F801	完全集合ファンクション
S151	型述語
S161	サブタイプ処理
T201	参照制約での比較可能データ型
T351	大カッコで囲まれたコメント
T431	拡張グループ化機能
T441	ABS および MOD ファンクション
T611	基本的な OLAP 演算子
T621	拡張数値ファンクション

表 B-6 に、Oracle が部分的にサポートする SQL/Foundation:2003 のオプション機能を示します。

表 B-6 部分的にサポートする SQL/Foundation:2003 のオプション機能

機能識別子、機能	部分的なサポート
B031、基本的な動的 SQL	Oracle はこの機能をサポートしますが、次の制限事項があります。 ■ Oracle は、記述子の項目のサブセットをサポートします。 ■ < 句による入力 > の場合、Oracle は < 入力記述子の使用 > のみをサポートします。 ■ < 句による出力 > の場合、Oracle は < 記述子への出力 > のみをサポートします。 ■ 動的パラメータは、コロンの後に疑問符ではなく識別子を指定する形式で示されます。
B032、拡張動的 SQL	Oracle は、この機能からグローバル文およびグローバル・カーソルを宣言する機能のみを実装します。その他の機能はサポートしません。
F034、拡張 REVOKE 文	Oracle は、この機能の次の部分をサポートします。 ■ F034-01、スキーマ・オブジェクトの所有者以外のユーザーによって実行される REVOKE 文 ■ F034-03、権限受領者が WITH GRANT OPTION を持つ権限を取り消す REVOKE 文 Oracle は、この機能のうち、次の部分と同等の機能を提供します。 ■ CASCADE: Oracle では、REVOKE によってすべての依存オブジェクトが無効になります。これにより、この後に CREATE コマンドと GRANT コマンドによって、無効になったオブジェクトが正常に再コンパイルできるようになり、メタデータが変更されるまでは、これらのオブジェクトは完全に使用禁止になります。
F052、期間および日時の算術	Oracle は、INTERVAL YEAR TO MONTH および INTERVAL DAY TO SECOND データ型のみをサポートします。
F111、SERIALIZABLE 以外の分離レベル	Oracle は、SERIALIZABLE に加えて、READ COMMITTED 分離レベルをサポートします。
F191、参照削除アクション	Oracle は、ON DELETE CASCADE および ON DELETE SET NULL をサポートします。
F302、INTERSECT 表演算子	Oracle は INTERSECT をサポートしますが、INTERSECT ALL はサポートしません。
F312、MERGE 文	Oracle の MERGE 文は規格とほとんど同じですが、次の制限事項があります。 ■ Oracle は、表の別名の前の AS キーワード（オプション）をサポートしません。 ■ Oracle は、表の別名の後にカッコで囲まれた列名のリストを使用して、USING 句で指定した表の列名を変更する機能をサポートしません。 ■ Oracle は、< 上書き句 > をサポートしません。
F391、長い識別子	Oracle は、最大 30 文字の識別子をサポートします。
F401、拡張結合表	Oracle は、FULL 外部結合をサポートします。
F403、パーティション結合表	Oracle は、FULL 外部結合を除くこの機能をサポートします。

表 B-6 部分的にサポートする SQL/Foundation:2003 のオプション機能（続き）

機能識別子、機能	部分的なサポート
F441、拡張集合ファンクション・サポート	Oracle は、この機能の次の部分をサポートします。 ■ ビューまたはインライン・ビューのいずれかで集計を使用して定義された列を参照する WHERE 句の機能 ■ 式の DISTINCT を使用しない COUNT ■ 集計問合せに対する外部参照となる列を参照する集計
F461、名前付きのキャラクタ・セット	Oracle は、Oracle 定義の名前が付いた複数のキャラクタ・セットをサポートします。Oracle は、この機能の他の部分をサポートしません。
F531、一時表	Oracle は、GLOBAL TEMPORARY 表をサポートします。
F591、導出表	Oracle は < 導出表 > をサポートしますが、次の制限事項があります。 ■ Oracle は、表の別名の前の AS キーワード（オプション）をサポートしません。 ■ Oracle は、< 導出列リスト > をサポートしません。
F831、完全カーソル更新	Oracle は、問合せでの FOR UPDATE 句と ORDER BY 句の組合せをサポートします。
S111、問合せ式での ONLY	Oracle は、ビューの階層に対する ONLY 句をサポートしますが、実表の階層はサポートしません。
S162、参照のサブタイプ処理	規格では参照する型の名前をカッコで囲む必要がありますが、Oracle はこの位置でのカッコの使用をサポートしません。
T041、基本的な LOB データ型サポート	Oracle は、この機能の次の部分をサポートします。 ■ キーワード BLOB、CLOB および NCLOB ■ 連結（CLOB での UPPER、LOWER および TRIM） Oracle は、この機能のうち、次の部分と同等のサポートを提供します。 ■ POSITION のかわりに INSTR の使用 ■ CHAR_LENGTH のかわりに LENGTH の使用 ■ SUBSTRING のかわりに SUBSTR の使用 Oracle は、この機能の次の部分をサポートしません。 ■ BLOB、CLOB および NCLOB のそれぞれのシノニムとしてのキーワード BINARY LARGE OBJECT、CHARACTER LARGE OBJECT または NATIONAL CHARACTER LARGE OBJECT ■ < バイナリ文字列リテラル > ■ LOB や CLOB の長さの上限を指定する機能 ■ BLOB の結合
T111、更新可能な結合、論理和および列	Oracle の更新可能な結合ビューは、規格の更新可能な結合機能のサブセットです。
T121、問合せ式での WITH（RECURSIVE を除く）	Oracle はこの機能をサポートします。ただし、< 問合せの名前 > の後に続く列名の変更機能はサポートしません。かわりに、< 問合せの名前 > の定義である問合せの <SELECT 構文のリスト > の列名を変更できます。
T122、副問合せでの WITH（RECURSIVE を除く）	機能 T121 と同じ制限事項があります。

表 B-6 部分的にサポートする SQL/Foundation:2003 のオプション機能（続き）

機能識別子、機能	部分的なサポート
T211、基本的なトリガー機能	Oracle のトリガーと規格のトリガーの違いは、次のとおりです。 ■ Oracle は、デフォルトではオプションの構文 FOR EACH STATEMENT（文トリガー）を提供しません。 ■ Oracle は、OLD TABLE および NEW TABLE をサポートしません。規格で指定されている遷移表（影響される行のイメージの前後の多重集合）は使用できません。 ■ トリガー本体は PL/SQL で記述されています。PL/SQL の機能は規格の手続き型言語 PSM と同等ですが、同じものではありません。 ■ トリガー本体では、古い遷移変数と新しい遷移変数が、コロンで始まる形式で参照されます。 ■ Oracle の行トリガーは、バッファリングおよびすべての行の処理後にまとめて実行されるのではなく、行の処理と同時に実行されます。規格のセマンティクスは決定的ですが、Oracle のリアルタイムで実行される行トリガーはより実用的です。 ■ Oracle の行および文の前のトリガーでは DML 文を実行できますが、規格では許可されていません。これに対し、Oracle の行の後の文では DML 文を実行できませんが、規格では許可されています。 ■ 複数のトリガーが適用される場合、規格ではそれらのトリガーは定義順に実行されます。Oracle では、実行順序は非決定的です。 ■ Oracle では、規格の（表権限の）TRIGGER 権限のかわりに、システム権限 CREATE TRIGGER および CREATE ANY TRIGGER を使用してトリガーの作成を規制します。
T271、セーブポイント	Oracle はこの機能をサポートしますが、次の制限事項があります。 ■ Oracle は、RELEASE SAVEPOINT をサポートしません。 ■ Oracle は、セーブポイント・レベルをサポートしません。
T331、基本的なロール	Oracle は、REVOKE ADMIN OPTION FOR の < ロール名 > を除いて、この機能をサポートします。
T432、ネストおよび連結した GROUPING SETS	Oracle は、連結した GROUPING SETS をサポートしますが、ネストした GROUPING SETS はサポートしません。
T591、NULL の可能性のある列に対する UNIQUE 制約	Oracle では、NULL であることが可能な 1 つ以上の列に対する UNIQUE 制約が可能です。UNIQUE 制約の対象が単一列の場合、セマンティクスは規格と同じになります（この制約には、指定した列の NULL である行の数に制限はありません）。UNIQUE 制約の対象が 2 つ以上の列の場合、セマンティクスは規格外になります。Oracle では、指定したすべての列の NULL である行の数に制限はありません。規格と異なり、指定した列の 1 つ以上の行が NULL 以外になる場合、制約対象の NULL 以外の列の値と同じ値を持つ別の行は、制約違反になり、許可されません。
T612、拡張 OLAP 操作	Oracle は、この機能の次の要素をサポートします。PERCENT_RANK、CUME_DIST、WIDTH_BUCKET、仮想の集合ファンクション、PERCENTILE_CONT および PERCENTILE_DISC です。 Oracle は、この機能の次の要素をサポートしません。 ■ ウィンドウ名 ■ ORDER BY 句のない ROW_NUMBER
T641、複数列の割当て	割当てソースが副問合せの場合、複数の列を割り当てる規格の構文がサポートされます。

Oracle には、表 B-7 で示す機能と同等の機能があります。

表 B-7 SQL/Foundation:2003 のオプション機能と同等の機能

機能識別子、機能	同等の機能
B031、基本的な動的 SQL	Oracle の埋込みプリプロセッサはこの機能を実装していますが、次の変更が加えられています。 ■ パラメータは、コロンの後に疑問符ではなく識別子を指定する形式で示されます。 ■ 規格の DESCRIBE OUTPUT は、Oracle の DESCRIBE SELECT LIST FOR 文に置き換えられます。 ■ Oracle では、動的 SQL 文を準備する PREPARE 文の前に、その動的 SQL 文を使用して物理的にカーソルを宣言する場合、DECLARE STATEMENT を使用できます。
B032、拡張動的 SQL	Oracle の DESCRIBE BIND VARIABLES は、規格の DESCRIBE INPUT と同等です。Oracle は、この機能のこれ以外のものを実装しません。
B122、ルーチン言語 C	Oracle は、C で記述された外部ルーチンをサポートしますが、このようなルーチンを作成する規格構文はサポートしません。
F032、CASCADE 削除動作	Oracle では、DROP コマンドによって削除されたオブジェクトのすべての依存オブジェクトが無効になります。無効になったオブジェクトを正常に再コンパイルできる方法を使用して、削除されたオブジェクトを再定義するまで、無効になったオブジェクトは完全に使用禁止になります。
F033、ALTER TABLE 文の DROP COLUMN 句	Oracle は DROP COLUMN 句を提供しますが、規格で使用される RESTRICT または CASCADE オプションは提供しません。
F121、基本的な診断管理	この機能の大部分の機能は、埋込み言語の SQLCA によって提供されます。
F231、権限表	Oracle は、この情報を次のメタデータ・ビューで使用可能にします。 ■ TABLE_PRIVILEGES のかわりに、ALL_TAB_PRIVS を使用します。 ■ COLUMN_PRIVILEGES のかわりに、ALL_COL_PRIVS を使用します。 ■ Oracle は USAGE 権限をサポートしないため、USAGE_PRIVILEGES と同等の権限は存在しません。
F341、使用状況表	Oracle は、この情報を ALL_DEPENDENCIES、DBA_DEPENDENCIES および USER_DEPENDENCIES ビューで使用可能にします。
F381、拡張スキーマ操作	Oracle は、この機能の次の要素を完全にサポートします。 ■ Oracle は、ALTER TABLE を使用して、表の制約を追加する規格の構文をサポートします。 Oracle は、この機能の次の要素を部分的にサポートします。 ■ Oracle は、表の制約を削除する規格の構文をサポートしますが、RESTRICT はサポートしません。 Oracle は、この機能のうち、次の要素と同等の機能を提供します。 ■ 列のデフォルト値を変更するには、ALTER TABLE の MODIFY オプションを使用します。 Oracle は、この機能の次の部分をサポートしません。 ■ DROP SCHEMA 文 ■ ALTER ROUTINE 文

表 B-7 SQL/Foundation:2003 のオプション機能と同等の機能（続き）

機能識別子、機能	同等の機能
F93、リテラルの Unicode エスケープ	Oracle の UNISTR 機能は、すべての Unicode 文字に対する数値のエスケープ・シーケンスをサポートします。
F402、LOB、配列および多重集合での名前列の結合	Oracle は、宣言した型がネストした表である列での名前付き列の結合をサポートします。Oracle は、LOB または配列での名前付き列の結合はサポートしません。
F571、真理値テスト	Oracle の LNNVL ファンクションは、規格の IS NOT TRUE と似ています。
F690、照合サポート	Oracle は、文字式の照合の変更に使用可能なファンクションを提供します。
F695、変換のサポート	Oracle の CONVERT ファンクションは、キャラクタ・セット間での変換に使用できます。
F771、結合管理	Oracle の CONNECT 文は規格の CONNECT 文と同じ機能を持ちますが、構文は異なります。規格の SET CONNECTION を使用するかわりに、Oracle では SQL 文を実行する必要のある接続を指定する AT 句を提供します。Pro*COBOL では、COMMIT または ROLLBACK の RELEASE オプションを使用して接続を切断できます。
S023、基本的な構造型	Oracle のオブジェクト型は、規格の構造型と同等です。
S025、最終構造型	Oracle の最終オブジェクト型は、規格の最終構造型と同等です。
S026、自己参照構造型	Oracle では、オブジェクト型 OT は、OT を参照する参照を持つことができます。
S041、基本的な参照型	Oracle の参照型は、規格の参照型と同等です。
S051、型の表の作成	Oracle のオブジェクト表は、規格の構造型の表と同等です。
S081、サブテーブル	Oracle はオブジェクト・ビューの階層をサポートしますが、オブジェクトの実表の階層はサポートしません。実表の階層をエミュレートするには、それらの実表にビューの階層を作成します。
S091、配列型	Oracle の VARRAY 型は、規格の配列型と同等です。ただし、Oracle は、LOB の配列の記憶域をサポートしません。添字を使用して配列の単一要素にアクセスするには、PL/SQL を使用する必要があります。Oracle は、規格以外の構文を使用してこの機能の次の部分をサポートします。 ■ 空の配列を含む VARRAY 型のインスタンスを構成するには、VARRAY 型コンストラクタを使用します。 ■ FROM 句で VARRAY をネスト解除するには、TABLE 演算子を使用します。
S092、ユーザー定義型の配列	Oracle は、オブジェクト型の VARRAY をサポートします。
S094、参照型の配列	Oracle は、参照の VARRAY をサポートします。
S095、問合せ別の配列コンストラクタ	Oracle は、CAST (MULTISET (SELECT ...) AS *varray_type*) を使用して配列コンストラクタをサポートします。ORDER BY を使用して配列の要素を順序付ける機能はサポートされません。
S097、配列要素割当て	PL/SQL では、規格（SQL/PSM）に似た構文を使用して配列要素を割り当てることができます。
S201、配列での SQL 起動ルーチン	PL/SQL は、配列をパラメータとして渡し、ファンクションの結果として配列を返す機能を提供します。
S202、多重集合での SQL 起動ルーチン	PL/SQL ルーチンは、パラメータとしてネストした表を持つことができます。 PL/SQL ルーチンは、ネストした表を戻すことができます。

表 B-7 SQL/Foundation:2003 のオプション機能と同等の機能（続き）

機能識別子、機能	同等の機能
S233、多重集合ロケータ	Oracle は、ネストした表のロケータをサポートします。
S241、変換ファンクション	Oracle Type Translator（OTT）は、変換と同じ機能を提供します。
S251、ユーザー定義の順序付け	Oracle のオブジェクト型の順序付け機能は、次のとおり規格の機能に対応します。 ■ Oracle の MAP 順序付けは、規格の ORDER FULL BY MAP 順序付けに対応します。 ■ Oracle の ORDER 順序付けは、規格の ORDER FULL BY RELATIVE 順序付けに対応します。 ■ Oracle のオブジェクト型で MAP または ORDER のいずれも宣言されていない場合、これは規格の EQUALS ONLY BY STATE に対応します。 ■ Oracle には、順序付けられていないオブジェクト型は存在しません。そのため、順序付けは変更できますが、削除することはできません。
S271、基本的な多重集合のサポート	Oracle では、ネストした表型として規格の多重集合がサポートされます。スカラー型（ST）に基づいた Oracle のネストした表のデータ型は、規格の用語では、ST 型の単一フィールドを持つ *column_value* という名前の行の多重集合と同じです。オブジェクト型に基づいた Oracle のネストした表型は、規格の構造型の多重集合と同等です。 Oracle は、ネストした表で使用するこの機能のうち、次の要素をサポートします。この場合、規格で多重集合に使用するのと同じ構文を使用します。 ■ CARDINALITY ファンクション ■ SET ファンクション ■ MEMBER 述語 ■ IS A SET 述語 ■ COLLECT 集計 次のように、この機能の他のすべての部分が規格以外の構文でサポートされます。 ■ 規格に MULTISET[] と表される、空の多重集合を作成するには、ネストした表型の空のコンストラクタを使用します。 ■ 規格では、ELEMENT（< 多重集合値の式 >）と表される、要素が 1 つ付いた多重集合の要素を 1 つのみ取得するには、スカラー副問合せを使用して、ネストした表から単一要素を選択します。 ■ 多重集合を列挙ごとに構成するには、ネストした表型のコンストラクタを使用します。 ■ 多重集合を問合せごとに構成するには、多重集合の引数を指定した CAST を使用して、ネストした表型にキャストします。 ■ 多重集合をネスト解除するには、FROM 句の TABLE 演算子を使用します。
S272、ユーザー定義型の多重集合	Oracle のネストした表型では、構造型の多重集合が可能です。Oracle には固有型がないため、固有型の多重集合はサポートされません。
S274、参照型の多重集合	ネストした表型は、参照型の列を 1 つ以上持つことができます。

表 B-7 SQL/Foundation:2003 のオプション機能と同等の機能（続き）

機能識別子、機能	同等の機能
S275、拡張多重集合のサポート	Oracle は、ネストした表で使用するこの機能のうち、次の要素をサポートします。この場合、規格で多重集合に使用するのと同じ構文を使用します。 ■ MULTISET UNION、MULTISET INTERSECTION および MULTISET EXCEPT 演算子 ■ SUBMULTISET 述語 ■ = および <> 述語 Oracle は、FUSION または INTERSECTION 集計をサポートしません。
S281、ネストしたコレクション型	Oracle では、コレクション型のネストが可能です（VARRAY およびネストした表）。
T042、拡張 LOB のサポート	Oracle は、この機能の次の要素を完全にサポートします。 ■ CLOB 引数の TRIM ファンクション Oracle は、この機能のうち、次の要素と同等の機能を提供します。 ■ BLOB および CLOB サブストリング（SUBSTR を使用してサポート） ■ SIMILAR 述語（REGEXPR_LIKE を使用し、Perl に似た構文とのパターン一致を実行してサポート） この機能の次の要素はサポートされません。 ■ BLOB または CLOB オペランドの付いた比較述語 ■ BLOB または CLOB オペランドの付いた CAST ■ OVERLAY（SUBSTR および文字列連結を使用してエミュレート可能） ■ BLOB または CLOB オペランドの付いた LIKE 述語
T051、行型	Oracle のオブジェクト型は、規格の行型のかわりに使用できます。
T061、UCS のサポート	Oracle は、この機能のうち、次の要素と同等の機能を提供します。 ■ Oracle は、文字データ型の宣言では、CHARACTERS と OCTETS のかわりに、それぞれ CHAR と BYTE のキーワードをサポートします。 ■ Oracle の COMPOSE ファンクションは、規格の NORMALIZE ファンクションと同等です。 Oracle は、IS NORMALIZED 述語をサポートしません。
T071、BIGINT データ型	多くの実装では、BIGINT は、19 桁（10 進）の大部分をサポートする 64 ビットの 2 進整数型を参照します。Oracle の NUMBER 型は、39 桁（10 進）をサポートします。
T131、再帰的問合せ	Oracle の START WITH および CONNECT BY 句を使用すると、多くの再帰的問合せを実行できます。
T132、副問合せでの再帰的問合せ	Oracle の START WITH および CONNECT BY 句を使用すると、多くの再帰的問合せを実行できます。
T141、SIMILAR 述語	Oracle は、Perl に似た構文とのパターン一致のための REGEXP_LIKE を提供します。
T172、表定義の AS 副問合せ句	Oracle の CREATE TABLE の AS 副問合せ機能は、規格とほとんど同じ機能を持ちますが、構文の一部が異なります。

表 B-7 SQL/Foundation:2003 のオプション機能と同等の機能（続き）

機能識別子、機能	同等の機能
T175、生成された列	生成された列は、他の列の式で計算された表の列です。Oracle は生成された列をサポートしませんが、ファンクション索引を使用すると、式の結果に索引を作成できます。
T176、シーケンス・ジェネレータのサポート	Oracle のシーケンスは規格のシーケンスと同じ機能を持ちますが、構文は異なります。
T322、SQL 起動ファンクションおよびプロシージャのオーバーロード	Oracle は、ファンクションおよびプロシージャのオーバーロードをサポートします。ただし、特定のデータ型の組合せを処理するルールは、規格と異なります。たとえば、規格では、引数の数値型のみが異なる同一名の 2 つのファンクションの共存が許可されますが、Oracle では許可されません。
T323、外部ルーチンの明示的なセキュリティ	外部ファンクション、プロシージャまたはパッケージの作成時に使用される Oracle の構文 AUTHID { CURRENT USER | DEFINER } は、規格の EXTERNAL SECURITY { DEFINER | INVOKER } と同等です。
T324、外部ルーチンの明示的なセキュリティ	PL/SQL ファンクション、プロシージャまたはパッケージの作成時に使用される Oracle の構文 AUTHID { CURRENT USER | DEFINER } は、規格の SQL SECURITY { DEFINER | INVOKER } と同等です。
T325、修飾された SQL パラメータ参照	PL/SQL は、ルーチン名を使用したパラメータ名の修飾をサポートします。
T326、表ファンクション	Oracle は、この機能のうち、次の要素と同等の機能を提供します。 ■ < 問合せ別の多重集合値コンストラクタ > は、CAST（MULTISET（< 問合せ式 >）AS < ネストした表型 >）を使用してサポートされます。 ■ < 表ファンクションの導出表 > は、VARRAY またはネストした表を引数として指定した FROM 句の TABLE 演算子を使用してサポートされます。 ■ < コレクション値の式 > は、VARRAY またはネストした表になる Oracle の式と同等です。 ■ < 戻り値の表型 > は、ネストした表を戻す PL/SQL ファンクションと同等です。
T433、複数引数ファンクション GROUPING	Oracle の GROUP_ID ファンクションを使用すると、グループ化した問合せによりグループを適切に区別できます。このファンクションは、規格の複数引数の GROUPING ファンクションの場合と同じ用途で使用できます。
T471、結果セットの戻り値	PL/SQL REF カーソルは、規格の結果セット・カーソルのすべての機能を提供します。
T491、LATERAL 導出表	Oracle の FROM 句の TABLE 演算子は、規格の LATERAL 演算子と同等です。
T571、配列を戻す外部 SQL 起動ファンクション	VARRAY を戻す Oracle の表ファンクションは、外部のプログラミング言語で定義できます。SQL でこのようなファクションを宣言する場合は、CREATE FUNCTION コマンドを PIPELINED USING 句とともに使用します。
T571、多重集合を戻す外部 SQL 起動ファンクション	ネストした表を戻す Oracle の表ファンクションは、外部のプログラミング言語で定義できます。SQL でこのようなファクションを宣言する場合は、CREATE FUNCTION コマンドを PIPELINED USING 句とともに使用します。
T581、正規表現のサブストリング・ファンクション	Oracle では、正規表現の一致を使用してサブストリング操作を実行する REGEXP_SUBSTR ファンクションを提供します。

表 B-7 SQL/Foundation:2003 のオプション機能と同等の機能（続き）

機能識別子、機能	同等の機能
T613、サンプリング	Oracle では、規格のキーワード TABLESAMPLE のかわりに、キーワード SAMPLE を使用します。Oracle では、規格のキーワード SYSTEM のかわりに、キーワード BLOCK を使用します。Oracle では、行のベルヌーイ・サンプリングの指定に、BLOCK キーワードを使用しません。規格では、キーワード BERNOULLI でこのサンプリングを指定します。
T652、SQL ルーチンでの SQL 動的文	PL/SQL は、動的 SQL をサポートします。
T654、外部ルーチンでの SQL 動的文	Oracle は、埋込み C での動的 SQL をサポートします。これは、外部ルーチンの作成に使用できます。
T655、循環的な依存ルーチン	PL/SQL は、再帰型をサポートします。

SQL/CLI:2003 に対する Oracle の準拠

Oracle の ODBC ドライバは、SQL/CLI:2003 に準拠しています。

SQL/PSM:2003 に対する Oracle の準拠

Oracle の PL/SQL は、キーワードの綴りや構成などの構文の小さな違いはありますが、SQL/PSM:2003 と同等の機能を提供します。

SQL/MED:2003 に対する Oracle の準拠

Oracle は SQL/MED:2003 に準拠しません。

SQL/OLB:2003 に対する Oracle の準拠

Oracle SQLJ は、SQL/OLB 99 に準拠していますが、SQL/OLB 2003 にはまだ準拠していません。

SQL/XML:2006 に対する Oracle の準拠

このドキュメントのリリース時点では、新版の SQL/XML（SQL/XML:2006）を想定していますが、最終形式ではまだ使用できません。この項には、素案および公認の変更提案に基づいた、最も適切と考えられる内容が反映されています。ただし、SQL/XML:2006 の最終形式には基づいていません。

規格の XML データ型は XML です。Oracle の同等のデータ型は、XMLType です。Oracle と規格の相違点がデータ型の名前の綴りのみである場合、規格の機能は Oracle で完全にサポートされているとみなします。

B-18 ページの表 B-8 に、XML スキーマ組込み型から XQuery の Oracle SQL データ型へのマッピングを示します。

表 B-8 XML スキーマ組込み型から XQuery の SQL データ型へのマッピング

XML スキーマ型	Oracle SQL 型
string	VARCHAR2(4000)
decimal	NUMBER
float	BINARY_FLOAT

表 B-8 XML スキーマ組込み型から XQuery の SQL データ型へのマッピング（続き）

XML スキーマ型	Oracle SQL 型
double	BINARY_DOUBLE
DateTime	TIMESTAMP WITH TIME ZONE
time	TIMESTAMP WITH TIME ZONE
date	TIMESTAMP WITH TIME ZONE
gDay	TIMESTAMP WITH TIME ZONE
gMonth	TIMESTAMP WITH TIME ZONE
gYear	TIMESTAMP WITH TIME ZONE
gYearMonth	TIMESTAMP WITH TIME ZONE
gMonthDay	TIMESTAMP WITH TIME ZONE
dayTimeDuration	INTERVAL DAY TO SECOND
yearMonthDuration	INTERVAL YEAR TO MONTH
normalizedString	VARCHAR2(4000)
untypedAtomic	VARCHAR2(4000)
integer	NUMBER
nonPositiveInteger	NUMBER
negativeInteger	NUMBER
long	NUMBER
int	NUMBER
short	NUMBER
byte	NUMBER
nonNegativeInteger	NUMBER
unsignedLong	NUMBER
unsignedInt	NUMBER
unsignedShort	NUMBER
unsignedByte	NUMBER
positiveInteger	NUMBER

表 B-9 に、Oracle が完全にサポートする規格の XML 機能を示します。

表 B-9 完全にサポートする SQL/XML:2005 の機能

機能識別番号	機能
X010	XML 型
X016	永続 XML 値
X020	XML の連結
X031	XMLElement
X032	XMLForest

表 B-9 完全にサポートする SQL/XML:2005 の機能（続き）

機能識別番号	機能
X034	XMLAgg
X035	XMLAgg の ORDER BY オプション
X036	XMLComment
X036	XMLPi
X041	表の基本マッピング（NULL なし）
X042	表の基本マッピング（NULL を NIL としてマッピング）
X043	表の基本マッピング（表をフォレストとしてマッピング）
X044	表の基本マッピング（表を要素としてマッピング）
X045	表の基本マッピング（ターゲット・ネームスペースを含む）
X046	表の基本マッピング（データのマッピング）
X047	表の基本マッピング（メタデータのマッピング）
X049	表の基本マッピング（16 進エンコーディング）
X060	XMLParse（文字列の入力および CONTENT オプション）
X061	XMLParse（文字列の入力および DOCUMENT オプション）
X070	XMLSerialize（文字列のシリアライズおよび CONTENT オプション）
X071	XMLSerialize（文字列のシリアライズおよび DOCUMENT オプション）
X072	XMLSerialize（文字列のシリアライズ）
X086	XMLTable での XML ネームスペースの宣言
X120	SQL ルーチンでの XML パラメータ
X121	外部ルーチンでの XML パラメータ
X201	XMLQuery（RETURNING CONTENT）
X203	XMLQuery（コンテキスト項目の受渡し）
X204	XMLQuery（XQuery 変数の初期化）
X251	XML（DOCUMENT（UNTYPED））型の永続 XML 値
X252	XML（DOCUMENT（ANY））型の永続値
X256	XML（DOCUMENT（XMLSCHEMA））型の永続値
X302	順序性列付きの XMLTable
X303	XMLTable（列のデフォルト・オプション）
X304	XMLTable（コンテキスト項目の受渡し）
X305	XMLTable（XQuery 変数の初期化）

表 B-9 の注意事項 : 機能 X041 ～ X047（表の基本マッピング）: Oracle の表のマッピングは、Java インタフェースおよびパッケージを介して実行できます。Oracle の表のマッピングは、表のみでなく問合せもマップするために汎用化されています。表のみをマップするには、SELECT * FROM *table_name* を指定します。

表 B-10 に、部分的にサポートする SQL/XML:2005 の機能を示します。

表 B-10 部分的にサポートする SQL/XML:2005 の機能

機能識別子、機能	部分的なサポート
X040、表の基本マッピング	Oracle は、この機能の次の要素をサポートします。 ■ X041、表の基本マッピング（NULL なし） ■ X042、表の基本マッピング（NULL を NIL としてマッピング） ■ X043、表の基本マッピング（表をフォレストとしてマッピング） ■ X044、表の基本マッピング（表を要素としてマッピング） ■ X045、表の基本マッピング（ターゲット・ネームスペースを含む） ■ X046、表の基本マッピング（データのマッピング） ■ X047、表の基本マッピング（メタデータのマッピング） ■ X049、表の基本マッピング（16 進エンコーディング） Oracle は、この機能の次の要素をサポートしません。 ■ X048、表の基本マッピング（BASE64 エンコーディング）
X060、XMLParse（文字列の入力および CONTENT オプション）	Oracle は、{PRESERVE | STRIP} WHITESPACE 構文をサポートしません。動作は常に STRIP WHITESPACE になります。
X200、XMLQuery	Oracle は、この機能の次の要素を完全にサポートします。 ■ X201、XMLQuery（RETURNING CONTENT） ■ X203、XMLQuery（コンテキスト項目の受渡し） ■ X204、XMLQuery（XQuery 変数の初期化） Oracle は、この機能の次の要素をサポートしません。 ■ X202、XMLQuery（RETURNING SEQUENCE） ■ { NULL | EMPTY } ON EMPTY 構文 ■ PASSING 句の必須の BY {REF | VALUE}。（Oracle は値セマンティクスのみをサポートします。）
X300、XMLTable	Oracle は、列パス式の逆軸をサポートしません。この制限事項を除き、Oracle は、この機能の次の要素を完全にサポートします。 ■ X086、XMLTable での XML ネームスペースの宣言 ■ X302、順序性列付きの XMLTable ■ X303、XMLTable（列のデフォルト・オプション） ■ X304、XMLTable（コンテキスト項目の受渡し） ■ X305、XMLTable（XQuery 変数の初期化） Oracle は、この機能の次の要素をサポートしません。 ■ X301、XMLTable（導出列リスト・オプション） ■ PASSING 句の必須の BY {REF | VALUE}。Oracle は、現時点で BY VALUE セマンティクスのみをサポートします。

表 B-11 に、Oracle が同等の機能を介してサポートする SQL/XML:2005 の機能を示します。

表 B-11 SQL/XML:2005 機能と同等の機能

機能識別子、機能	同等の機能
X011、XML 型の配列	Oracle では、配列に名前を付ける必要があります。規格では匿名です。
X012、XML 型の多重集合	Oracle で XML 型の多重集合に相当するものは、XML 型の単一列を持つネストした表です。
X013、固有の XML 型	固有型は、1 つの属性のみを持つオブジェクト型を使用してエミュレートできます。
X014、XML 型の属性	Oracle では、オブジェクト型の属性は XMLType 型にできますが、この構文はオブジェクト型の作成には標準ではありません。
X025、XMLCast	Oracle は、この機能のうち、次の要素と同等の機能を提供します。 ■ XML からスカラー型にキャストするには、EXTRACTVALUE を使用します。XML 値が型指定される場合、結果は XML 型に最も類似します。型指定されない場合、結果の型は VARCHAR（4000）になります。他のスカラー型に変換するには、CAST を使用します。 ■ スカラー型から XML にキャストするには、スカラー値を XMLQuery に渡し、ドキュメント・コンストラクタに挿入します。 Oracle は XML 型を 1 つしか持たないため、XML から XML にキャストする必要はありません。
X076、XMLSerialize の VERSION オプション	シリアライズの前に XML バージョンを設定するには、XMLRoot を使用します。
X080、XML 公開時のネームスペース	XMLElement の Oracle 実装では、XMLAttributes を使用してネームスペースが定義されます（XMLNamespaces は実装されません）。ただし、XMLAttributes は、XMLForest ではサポートされません。
X090、XML 文書の述語	Oracle では、ISFRAGMENT メソッドを使用して XML 値が文書かどうかをテストできます。
X096、XMLExists	XPath を評価するには、EXISTSNODE を使用します。ノードが検出された場合は 1 を戻し、検出されない場合は 0 を戻します。XPath 式以外の XQuery 式はサポートされません。また、Oracle は、XPath 1.0 式をサポートします（XQuery のサブセットである XPath 2.0 はサポートされません）。
X121、外部ルーチンでの XML パラメータ	Oracle は、規格以外のインタフェースを使用して外部ルーチンに渡される XML 値をサポートします。
X141、IS VALID 述語（データ駆動形式）	XMLISVALID メソッドは IS VALID 述語と同等で、データ駆動形式をサポートします。
X142、IS VALID 述語（ACCORDING TO 句）	XMLISVALID メソッドは IS VALID 述語と同等で、ACCORDING TO 句に相当するものを含みます。
X143、IS VALID 述語（ELEMENT 句）	XMLISVALID メソッドは IS VALID 述語と同等で、ELEMENT 句に相当するものを含みます。
X144、IS VALID 述語（スキーマの場所）	XMLISVALID メソッドは IS VALID 述語と同等で、登録済 XML スキーマのスキーマの場所の指定をサポートします。
X145、IS VALID 述語（CHECK 制約の外部）	XMLISVALID メソッドは IS VALID 述語と同等で、CHECK 制約の外部で使用できます。
X151、DOCUMENT オプション付きの IS VALID 述語	XMLISVALID メソッドは IS VALID 述語と同等で、DOCUMENT 句と同等の検証を実行します（XMLISVALID は「内容」の検証をサポートしません）。

表 B-11 SQL/XML:2005 機能と同等の機能（続き）

機能識別子、機能	同等の機能
X156、IS VALID 述語（ELEMENT 句の付いたオプションの NAMESPACE）	XMLISVALID メソッドは IS VALID 述語と同等で、ネームスペースの要素の検証に使用できます。
X157、IS VALID 述語（ELEMENT 句の付いた NO NAMESPACE）	XMLISVALID メソッドは IS VALID 述語と同等で、名前のないネームスペースの要素の検証に使用できます。
X160、登録済 XML スキーマの基本情報スキーマ	Oracle の静的データ・ディクショナリ・ビュー ALL_XML_SCHEMAS は、現行のユーザーがアクセスできる登録済の XML スキーマのリストを提供します。ALL_XML_SCHEMAS.SCHEMA_URL 列は、規格の XML_SCHEMAS.XML_SCHEMA_LOCATION 列に対応します。登録済 XML スキーマのターゲット・ネームスペースは、ALL_XML_SCHEMAS.SCHEMA を調べることによって確認できます。Oracle は、規格の XML_SCHEMAS の他の列に相当する列を持ちません。
X161、登録済 XML スキーマの拡張情報スキーマ	Oracle は、規格の XML_SCHEMA_NAMESPACES および XML_SCHEMA_ELEMENTS に対応する静的データ・ディクショナリ・ビューを持ちません。ただし、登録済 XML スキーマに関するすべての情報は、ALL_XML_SCHEMAS.SCHEMA 列にある実際の XML スキーマを調べることによって確認できます。これを調べると、登録済 XML スキーマが非決定的であるかどうかも確認できます。また、非決定的なネームスペースと要素も確認できます。
X191、XML（DOCUMENT (XMLSCHEMA)）型	Oracle は、この構文をサポートしません。ただし、表の列は登録済 XML スキーマで制約できます。この場合、列のすべての値は XML（DOCUMENT（XMLSCHEMA））型になります。
X221、XML 受渡しメカニズム BY VALUE	Oracle は値セマンティクスのみをサポートしますが、明示的な BY VALUE 句はサポートしません。
X232、XML（CONTENT（ANY））型	Oracle は、この構文を型の修飾子としてサポートしませんが、Oracle の XMLType は、一時値でこのデータ型をサポートします。永続値は、XML（DOCUMENT（ANY））型になります。この型は、XML（CONTENT（ANY））のサブセットです。
X241、XML 文書作成時の RETURNING CONTENT	Oracle は、この構文をサポートしません。Oracle では、文書を作成するファンクション（XMLAgg、XMLComment、XMLConcat、XMLElement、XMLForest および XMLPi）の動作は、常に RETURNING CONTENT になります。
X260、XML 型（ELEMENT 句）	Oracle は、この構文をサポートしません。ただし、表の列は登録済 XML スキーマの最上位の要素で制約できます。
X262、XML 型（ELEMENT 句の付いたオプションの NAMESPACE）	Oracle は、この構文をサポートしません。ただし、表の列は、登録済 XML スキーマのターゲット・ネームスペース以外のネームスペースにある最上位の要素で制約できます。
X263、XML 型（ELEMENT 句の付いた NO NAMESPACE）	Oracle は、この構文をサポートしません。ただし、表の列は登録済 XML スキーマの名前のないネームスペースにある最上位の要素で制約できます。
X264、XML 型（スキーマの場所）	Oracle は、この構文をサポートしません。ただし、表の列はスキーマの場所で識別される登録済 XML スキーマで制約できます。
X271、XMLValidate（データ駆動形式）	SCHEMAVALIDATE メソッドは XMLValidate と同等で、データ駆動形式をサポートします。
X272、XMLValidate（ACCORDING TO 句）	SCHEMAVALIDATE メソッドは XMLValidate と同等で、特定の登録済 XML スキーマの指定に使用できます。
X273、XMLValidate（ELEMENT 句）	SCHEMAVALIDATE メソッドは XMLValidate と同等で、特定の登録済 XML スキーマの特定要素を指定するために使用できます。

表 B-11 SQL/XML:2005 機能と同等の機能（続き）

機能識別子、機能	同等の機能
X274、XMLValidate（スキーマの場所）	SCHEMAVALIDATE メソッドは XMLValidate と同等で、スキーマの場所を示す URL で特定の登録済 XML スキーマを指定するために使用できます。
X281、DOCUMENT オプションの付いた XMLValidate	SCHEMAVALIDATE メソッドは XMLValidate と同等です。SCHEMAVALIDATE は XML 文書の検証のみを実行します（内容の検証は実行しません）。
X285、XMLValidate（ELEMENT 句の付いたオプションの NAMESPACE）	SCHEMAVALIDATE メソッドは XMLValidate と同等で、特定の登録済 XML スキーマのターゲット・ネームスペース以外のネームスペースにある特定要素を指定するために使用できます。
X286、XMLValidate（ELEMENT 句の付いた NO NAMESPACE）	SCHEMAVALIDATE メソッドは XMLValidate と同等で、特定の登録済 XML スキーマの名前のないネームスペースにある特定要素を指定するために使用できます。
Xnnn *、XML テキスト・ノード・コンストラクタ	Oracle の XMLCData ファンクションは、テキスト・ノードの作成に使用できます。

* 正確な機能識別番号は、このドキュメントの公開時点では不明です。

表 B-12 に、Oracle がサポートしない SQL/XML:2003 の機能を示します。

表 B-12 サポートしない SQL/XML:2003 機能

機能識別番号	機能
X015	XML 型のフィールド
X030	XMLDocument
X048	表の基本マッピング（BASE64 エンコーディング）
X050	表の拡張マッピング
X051	表の拡張マッピング（NULL なし）
X052	表の拡張マッピング（NULL を NIL としてマッピング）
X053	表の拡張マッピング（表をフォレストとしてマッピング）
X054	表の拡張マッピング（表を要素としてマッピング）
X055	表の拡張マッピング（ターゲット・ネームスペースを含む）
X056	表の拡張マッピング（データのマッピング）
X057	表の拡張マッピング（メタデータのマッピング）
X058	表の拡張マッピング（バイナリ文字列の BASE64 エンコーディング）
X059	表の拡張マッピング（バイナリ文字列の 16 進エンコーディング）
X065	XMLParse（BLOB 入力および CONTENT オプション）
X066	XMLParse（BLOB 入力および DOCUMENT オプション）
X073	XMLSerialize（BLOB シリアライズおよび CONTENT オプション）
X074	XMLSerialize（BLOB シリアライズおよび DOCUMENT オプション）
X075	XMLSerialize（BLOB シリアライズ）
X076	XMLSerialize（VERSION）

表 B-12 サポートしない SQL/XML:2003 機能（続き）

機能識別番号	機能
X077	XMLSerialize（明示的な ENCODING オプション）
X078	XMLSerialize（明示的な XML 宣言）
X081	問合せレベルのネームスペースの宣言
X082	DML での XML ネームスペースの宣言
X083	DDL での XML ネームスペースの宣言
X084	複合文での XML ネームスペースの宣言
X085	事前定義のネームスペース接頭辞
X091	XML コンテンツの述語
X100	XML に対するホスト言語のサポート（CONTENT オプション）
X101	XML に対するホスト言語のサポート（DOCUMENT オプション）
X110	XML に対するホスト言語のサポート（VARCHAR マッピング）
X111	XML に対するホスト言語のサポート（CLOB マッピング）
X131	問合せレベルの XMLBINARY 句
X132	DML の XMLBINARY 句
X133	DDL の XMLBINARY 句
X134	複合文の XMLBINARY 句
X135	副問合せの XMLBINARY 句
X152	CONTENT オプション付きの IS VALID 述語
X153	SEQUENCE オプション付きの IS VALID 述語
X155	IS VALID 述語（ELEMENT 句のない NAMESPACE）
X170	XML NULL 処理のオプション
X171	NIL ON NO CONTENT オプション
X181	XML（DOCUMENT（UNTYPED））型
X182	XML（DOCUMENT（ANY））型
X190	XML（SEQUENCE）型
X192	XML（CONTENT（XMLSCHEMA））型
X202	XMLQuery（RETURNING SEQUENCE）
X211	XML 1.1 のサポート
X222	XML 受渡しメカニズム BY REF
X231	XML（CONTENT（UNTYPED））型
X242	XML 文書作成時の RETURNING SEQUENCE
X253	XML（CONTENT（UNTYPED））型の永続 XML 値
X254	XML（CONTENT（ANY））型の永続 XML 値
X255	XML（SEQUENCE）型の永続値

表 B-12 サポートしない SQL/XML:2003 機能（続き）

機能識別番号	機能
X257	XML（CONTENT（XMLSCHEMA））型の永続値
X261	XML 型（ELEMENT 句のない NAMESPACE）
X282	XMLValidate（CONTENT オプション付き）
X283	XMLValidate（SEQUENCE オプション付き）
X284	XMLValidate（ELEMENT 句のない NAMESPACE）
X290	名前および識別子のマッピング
X301	XMLTable（導出列リスト・オプション）
Xnnn *	XML に対するホスト言語のサポート（BLOB マッピング）
Xnnn *	XML に対するホスト言語のサポート（STRIP WHITESPACE オプション）
Xnnn *	XML に対するホスト言語のサポート（PRESERVE WHITESPACE オプション）

* 正確な機能識別番号は、このドキュメントの公開時点では不明です。

FIPS 127-2 に対する Oracle の準拠

Oracle は、最新の FIPS（Federal Information Processing Standard）である FIPS PUB 127-2 に完全に準拠しています。現在、この規格は、公開されていません。ただし、FIPS 127-2 で定義されたデータベース要素のサイズに関する情報に依存するアプリケーションを使用するユーザーのために、準拠性についての詳細を表 B-13 に示します。

表 B-13 データベース要素のサイズ設定

データベース要素	FIPS	Oracle Database
識別子の長さ（バイト単位）	18	30
CHARACTER データ型の長さ（バイト単位）	240	2000
NUMERIC データ型の 10 進精度	15	38
DECIMAL データ型の 10 進精度	15	38
INTEGER データ型の 10 進精度	9	38
SMALLINT データ型の 10 進精度	4	38
FLOAT データ型の 2 進精度	20	126
REAL データ型の 2 進精度	20	63
DOUBLE PRECISION データ型の 2 進精度	30	126
表の中の列	100	1000
INSERT 文の中の値	100	1000
UPDATE 文内の SET 句（注意 1）	20	1000
行の長さ（注意 2、注意 3）	2,000	2,000,000
一意制約の中の列	6	32

表 B-13 データベース要素のサイズ設定（続き）

データベース要素	FIPS	Oracle Database
一意制約の長さ（注意 2）	120	（注意 4）
外部キー列リストの長さ（注意 2）	120	（注意 4）
GROUP BY 句の中の列	6	255（注意 5）
GROUP BY 列のリストの長さ	120	（注意 5）
ORDER BY 句の中のソート指定	6	255（注意 5）
ORDER BY 列のリストの長さ	120	（注意 5）
参照整合性制約内の列	6	32
SQL 文で参照される表	15	無制限
同時にオープンできるカーソル	10	（注意 6）
SELECT 構文のリストの項目	100	1000

注意 1: UPDATE 文の SET 句の数とは、SET キーワードの後に続くカンマで区切られる項目の数のことです。

注意 2: FIPS PUB では、列セットの長さを次の値の合計として規定しています。つまり、列の数を 2 倍した値、各文字列の長さ（バイト単位）、各真数値列の 10 進精度に 1 を加えた値、各概数値列の 2 進精度を 4 で割って 1 を加えた値の合計です。

注意 3: 行の最大長に対する Oracle の制限は、長さ 2GB の LONG 値とそれぞれの長さが 4000 バイトである 999 の VARCHAR2 値を含む行の最大長に基づいています。2(254) + 231 + (999(4000))

注意 4: 一意キー制約に対する Oracle の制限は、Oracle データ・ブロックのサイズ（初期化パラメータ DB_BLOCK_SIZE によって指定される）の半分からオーバーヘッドを引いたものになります。

注意 5: Oracle は、GROUP BY 句の列数や ORDER BY 句のソート指定の数に対して制限を設定しません。ただし、GROUP BY 句や ORDER BY 句のすべての式のサイズの合計は、Oracle データ・ブロックのサイズ（初期化パラメータ DB_BLOCK_SIZE によって指定される）からオーバーヘッドを引いたサイズに制限されています。

注意 6: 同時にオープンできるカーソルの数に対する Oracle の制限は、初期化パラメータ OPEN_CURSORS によって指定されます。このパラメータの最大値は、使用しているオペレーティング・システムで使用可能なメモリーによって異なりますが、どんな場合でも 100 を超えます。

標準 SQL に対する Oracle 拡張機能

Oracle では、標準 SQL 以外でも様々な機能をサポートしています。Oracle アプリケーションでは、Core SQL:2003 の使用と同様に、これらの拡張機能を使用できます。

他の SQL 処理系へのアプリケーションの移植性を考慮する場合、Oracle の FIPS フラガーを使用して、埋込み SQL プログラムでの Entry SQL92 に Oracle の拡張機能を位置付けてください。FIPS フラガーは、Oracle プリコンパイラと SQL*Module コンパイラの一部です。

> **参照：** FIPS フラガーの使用方法の詳細は、『Pro*COBOL プログラマーズ・ガイド』および『Pro*C/C++ プログラマーズ・ガイド』を参照してください。

以前の規格に対する Oracle の準拠

今回のリリースの Oracle Database は、最新版の SQL 規格である SQL:2003 に準拠しています。SQL-92（特に SQL-92 のエントリ・レベル）または SQL:1999 は、SQL:2003 に置き換えられているため、Oracle は、データベースの今回のリリースがこれらの以前の規格に準拠していることを正式に明示していません。SQL 規格の新旧版の間（SQL-92 と SQL:1999 の間および SQL:1999 と SQL:2003 の間の両方）に行われた一部の変更（大部分は小規模なもの）は、アプリケーションに影響を及ぼす可能性があります。影響のある非互換の詳細については、SQL 規格またはこの規格に関する説明資料を参照してください。重要な情報の 1 つは、SQL/Foundation:1999 および SQL/Foundation:2003 の Annex E です。

今回のリリースの Oracle Database では、以前の版の SQL 構造を引続き使用できる場合があります。このようなサポートの多くは、ベンダーによる妥当な拡張機能として容認されています。これは、データベースのバージョン間の非互換性を最小限にするための Oracle の一般的な方針です。この方針に基づき、以前の形式が適している場合は、引続き保持されます。いずれにしても、以前の SQL と SQL:2003 の間の差異（前述）はあまり重要ではありません。

キャラクタ・セットのサポート

Oracle は、ほとんどの各国語およびベンダー固有のエンコードされたキャラクタ・セットの規格をサポートしています。Oracle がサポートするキャラクタ・セットの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

Unicode は、エンコードされたユニバーサル・キャラクタ・セットで、単一キャラクタ・セットを使用したすべての言語の情報を格納できます。Unicode は、XML、Java、JavaScript、LDAP などの最新の規格で必要です。Unicode は、ISO/IEC 規格 10646 に準拠しています。ISO/IEC 規格 10646 のコピーについては、次の宛先に申し込んでください。

International Organization for Standardization
1 Rue de Varembé
Case postale 56
CH-1211, Geneva 20, Switzerland
電話番号 : +41.22.749.0111
FAX 番号 : +41.22.733.3430
URL: http://www.iso.ch/

Oracle Database は、Unicode 規格の最新バージョンである Unicode 4.0 に完全に準拠しています。この規格の最新情報については、次の Unicode Consortium の Web サイトを参照してください。

`http://www.unicode.org`

Oracle は、3 つのデータベース・キャラクタ・セット（ASCII ベースのプラットフォーム用の UTF8 と AL32UTF8、および EBCDIC プラットフォーム用の UTFE）によってエンコードされた UTF-8（8 ビット）を使用します。Unicode のサポートを段階的に実装する場合は、SQL の `NCHAR` データ型（`NCHAR`、`NVARCHAR2` および `NCLOB`）に対して、Unicode データを、各国語キャラクタ・セットで、UTF-16 または UTF-8 エンコード形式によって格納できます。

> **参照：** Oracle のキャラクタ・セットのサポートの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

C

Oracle の正規表現のサポート

Oracle による正規表現の実装は、IEEE の Portable Operating System Interface（POSIX）正規表現規格および Unicode Consortium の Unicode Regular Expression Guidelines に準拠します。

この付録では、次の内容を説明します。

- 多言語の正規表現の構文
- 正規表現の演算子の多言語拡張
- Oracle の正規表現における Perl による拡張機能

多言語の正規表現の構文

表 C-1 に、POSIX 規格の Extended Regular Expression（ERE）の構文に定義されているすべての演算子を示します。Oracle は、POSIX 規格に ASCII（英語）データの検索用に定義されているこれらの演算子の完全な構文および検索セマンティクスに従います。演算子とその使用例、およびそれらの演算子の Oracle 多言語拡張の詳細は、『Oracle Database アドバンスト・アプリケーション開発者ガイド』を参照してください。表の後に示す注意では、演算子とそれらの機能、およびそれらの演算子の Oracle 多言語拡張の詳細を示します。表 C-2 に、POSIX 演算子の Oracle のサポートおよび多言語拡張を示します。

表 C-1 正規表現の演算子およびメタ記号

演算子	説明
\	バックスラッシュは、そのコンテキストに応じて、次の 4 つの異なる意味を持つことができます。 ■ バックスラッシュ自体 ■ 次の文字の引用 ■ 演算子の導入 ■ 意味なし
*	0（ゼロ）回以上の繰返しに一致します。
+	1 回以上の繰返しに一致します。
?	0 回または 1 回の繰返しに一致します。
|	代替一致を指定する代替演算子です。
^	デフォルトでは文字列の先頭に一致します。複数行モードでは、ソース文字列内の任意の行の先頭に一致します。
$	デフォルトでは文字列の末尾に一致します。複数行モードでは、ソース文字列内の任意の行の末尾に一致します。
.	サポートされるキャラクタ・セットの任意の文字（NULL を除く）に一致します。
[ ]	大カッコで囲んだ一致リストのいずれかの正規表現に一致します。非一致リストの正規表現はキャレット（^）で始まり、リストに示されている正規表現以外の任意の文字に一致するリストを指定します。
( )	グループ化の正規表現で、1 つの部分正規表現として処理されます。
{m}	厳密に m 回の繰返しに一致します。
{m,}	m 回以上の繰返しに一致します。
{m,n}	m 回以上 n 回以下の繰返しに一致します。
\n	後方参照表現（n は 1 ～ 9 の数字）は、\n の前のカッコで囲まれた n 番目の部分正規表現に一致します。
[..]	1 つの照合要素を指定します。複数文字要素を指定できます（スペイン語の [.ch.] など）。
[: :]	文字クラスを指定します（[:alpha:] など）。文字クラス内の任意の文字に一致します。
[==]	等価クラスを指定します。たとえば、[=a=] は基本文字「a」のすべての文字に一致します。

正規表現の演算子の多言語拡張

Oracle による POSIX 演算子の実装は、多言語データに適用される場合、POSIX 規格に指定されている一致機能を拡張します。表 C-2 に、POSIX 規格のコンテキストの演算子の関係を示します。

- 1 列目に、サポートされる演算子を示します。
- 2 列目と 3 列目に、POSIX 規格（Basic Regular Expression（BRE）および Extended Regular Expression（ERE））でその演算子が定義されているかどうかを示します。
- 4 列目に、Oracle による実装で、多言語データを処理するためにその演算子のセマンティクスが拡張されているかどうかを示します。

Oracle では、マルチバイト文字を直接入力する手段があるか、またはファンクションを使用してマルチバイト文字を構成できる場合は、マルチバイト文字を直接入力できます。書式「\xxxx」の Unicode エンコーディング値の 16 進数値は使用できません。Oracle は、文字の図形的な表現ではなく、文字のエンコードに使用されるバイト値に基づいて文字を評価します。

表 C-2 POSIX と多言語の演算子の関係

演算子	POSIX の BRE 構文	POSIX の ERE 構文	多言語拡張
\	はい	はい	—
*	はい	はい	—
+	--	はい	—
?	—	はい	—
|	—	はい	—
^	はい	はい	はい
$	はい	はい	はい
.	はい	はい	はい
[ ]	はい	はい	はい
( )	はい	はい	—
{m}	はい	はい	—
{m,}	はい	はい	—
{m,n}	はい	はい	—
\n	はい	はい	はい
[..]	はい	はい	はい
[::]	はい	はい	はい
[==]	はい	はい	はい

Oracle の正規表現における Perl による拡張機能

Oracle Database の正規表現ファンクションおよびその条件では、一般的に使用される Perl による演算子の多くが使用可能です。ただし、POSIX 規格の一部は使用できません。表 C-3 に、使用可能な演算子を示します。詳細とその例については、『Oracle Database アドバンスト・アプリケーション開発者ガイド』を参照してください。

表 C-3 Oracle の正規表現における Perl による演算子

演算子	説明
\d	数字です。
\D	数字以外です。
\w	単語構成文字です。
\W	単語構成文字以外です。
\s	空白文字です。
\S	空白文字以外です。
\A	文字列の先頭、または文字列末尾の改行文字の直前のみに一致します。
\Z	文字列の末尾のみに一致します。
*?	先行のパターン要素に 0 回以上一致します（最短マッチ）。
+?	先行のパターン要素に 1 回以上一致します（最短マッチ）。
??	先行のパターン要素に 0 回または 1 回一致します（最短マッチ）。
{n}?	先行のパターン要素に *n* 回一致します（最短マッチ）。
{n,}?	先行のパターン要素に *n* 回以上一致します（最短マッチ）。
{n,m}?	先行のパターン要素に *n* 回以上 *m* 回以下一致します（最短マッチ）。

D

Oracle Database 予約語

この付録では、Oracle SQL の予約語を示します。アスタリスク（*）が付いた語は、ANSI の予約語でもあります。

> **注意：** Oracle では、次の表に示す予約語の他に、暗黙的に生成されるスキーマ・オブジェクトとサブオブジェクトには「SYS_」で始まるシステム生成名を使用します。名前解決での競合を避けるため、この接頭辞は明示的に指定するスキーマ・オブジェクト名やサブオブジェクト名では使用しないでください。

V$RESERVED_WORDS データ・ディクショナリ・ビューには、すべてのキーワードに関する追加情報が表示されます。この情報には、キーワードが常に予約されているか、特定の使用に対してのみ予約されているかの情報も含まれます。詳細は、『Oracle Database リファレンス』を参照してください。

```
ACCESS
ADD *
ALL *
ALTER *
AND *
ANY *
AS *
ASC *
AUDIT
BETWEEN *
BY  *
CHAR *
CHECK *
CLUSTER
COLUMN
COMMENT
COMPRESS
CONNECT *
CREATE *
CURRENT *
DATE *
DECIMAL *
DEFAULT *
DELETE *
DESC *
DISTINCT *
DROP *
ELSE *
EXCLUSIVE
EXISTS
```

```
FILE
FLOAT *
FOR *
FROM *
GRANT *
GROUP *
HAVING *
IDENTIFIED
IMMEDIATE *
IN  *
INCREMENT
INDEX
INITIAL
INSERT *
INTEGER *
INTERSECT *
INTO *
IS *
LEVEL *
LIKE *
LOCK
LONG
MAXEXTENTS
MINUS
MLSLABEL
MODE
MODIFY
NOAUDIT
NOCOMPRESS
NOT *
NOWAIT
NULL *
NUMBER
OF *
OFFLINE
ON *
ONLINE
OPTION *
OR *
ORDER *
PCTFREE
PRIOR *
PRIVILEGES *
PUBLIC *
RAW
RENAME
RESOURCE
REVOKE *
ROW
ROWID
ROWNUM
ROWS *
SELECT *
SESSION *
SET *
SHARE
SIZE *
SMALLINT *
START
SUCCESSFUL
```

```
SYNONYM
SYSDATE
TABLE *
THEN *
TO *
TRIGGER
UID
UNION *
UNIQUE *
UPDATE *
USER *
VALIDATE
VALUES *
VARCHAR *
VARCHAR2
VIEW *
WHENEVER *
WHERE
WITH *
```

E

詳細な例

このマニュアルの本文では、ほぼすべてのリファレンス項目に例を記載しています。この付録では、単一の SQL 文のコンテキストに適さない詳細な例を示します。これらの例は、特定の Oracle 機能を活用するために使用する一連の手順を示すためのものです。このリファレンス本文の個々の SQL 文の構文図およびセマンティクスにかわるものではありません。記載されている参照項目を使用して、必要な権限、制限事項、構文などの補足情報を参照してください。

この付録では、次の内容を説明します。

- 拡張索引作成機能の使用方法
- SQL 文での XML の使用方法

拡張索引作成機能の使用方法

この項では、拡張索引作成機能の単純で現実的な使用例で必要となる手順の例を示します。

HR.employees 表の給与をランク付けし、10 ～ 20 にランク付けされる給与を検索するとします。この場合、次のように DENSE_RANK ファンクションを使用できます。

```
SELECT last_name, salary FROM
   (SELECT last_name, DENSE_RANK() OVER
      (ORDER BY salary DESC) rank_val, salary FROM employees)
   WHERE rank_val BETWEEN 10 AND 20;
```

参照： 「DENSE_RANK」（5-56 ページ）

このネストした問合せは多少複雑で、employees 表のソートのみではなく全体スキャンも必要です。同じ結果が得られるもう 1 つの方法には、拡張索引作成機能を使用する方法があります。この方法による問合せは単純になります。この問合せには、ROWID による索引スキャンおよび表アクセスのみ必要です。そのため、問合せが効率的に実行されます。

最初の手順では、position_im 実装タイプ（索引の定義、メンテナンスおよび作成を行うためのメソッド・ヘッダーを含む）を作成します。型本体のほとんどには PL/SQL が使用されています。その部分はイタリック体で示しています。

ファンクション ODCIINDEXCREATE() 内に EXECUTE IMMEDIATE 文が含まれるため、型は、AUTHID CURRENT_USER 句を使用して作成する必要があります。デフォルトでは、このファンクションは定義者権限で実行されます。後続のドメイン索引の作成でこのファンクションがコールされる場合、実行者の権限は定義者権限とは異なります。

参照：

- 17-3 ページの「CREATE TYPE」および 17-5 ページの「CREATE TYPE BODY」を参照してください。
- この文の ODCI ルーチンの詳細は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。

```
CREATE OR REPLACE TYPE position_im AUTHID CURRENT_USER AS OBJECT
(
  curnum  NUMBER,
  howmany NUMBER,
  lower_bound NUMBER,
  upper_bound NUMBER,
/* lower_bound and upper_bound are used for the
index-based functional implementation */
  STATIC FUNCTION ODCIGETINTERFACES(ifclist OUT SYS.ODCIOBJECTLIST) RETURN NUMBER,
  STATIC FUNCTION ODCIINDEXCREATE
    (ia SYS.ODCIINDEXINFO, parms VARCHAR2, env SYS.ODCIEnv) RETURN NUMBER,
  STATIC FUNCTION ODCIINDEXTRUNCATE (ia SYS.ODCIINDEXINFO,
                                     env SYS.ODCIEnv) RETURN NUMBER,
  STATIC FUNCTION ODCIINDEXDROP(ia SYS.ODCIINDEXINFO,
                                env SYS.ODCIEnv) RETURN NUMBER,
  STATIC FUNCTION ODCIINDEXINSERT(ia SYS.ODCIINDEXINFO, rid ROWID,
                                  newval NUMBER, env SYS.ODCIEnv) RETURN NUMBER,
  STATIC FUNCTION ODCIINDEXDELETE(ia SYS.ODCIINDEXINFO, rid ROWID, oldval NUMBER,
                                  env SYS.ODCIEnv) RETURN NUMBER,
  STATIC FUNCTION ODCIINDEXUPDATE(ia SYS.ODCIINDEXINFO, rid ROWID, oldval NUMBER,
                                  newval NUMBER, env SYS.ODCIEnv) RETURN NUMBER,
  STATIC FUNCTION ODCIINDEXSTART(SCTX IN OUT position_im, ia SYS.ODCIINDEXINFO,
                                 op SYS.ODCIPREDINFO, qi SYS.ODCIQUERYINFO,
                                 strt NUMBER, stop NUMBER, lower_pos NUMBER,
                                 upper_pos NUMBER, env SYS.ODCIEnv) RETURN NUMBER,
  MEMBER FUNCTION ODCIINDEXFETCH(SELF IN OUT position_im, nrows NUMBER,
                                 rids OUT SYS.ODCIRIDLIST, env SYS.ODCIEnv)
                                 RETURN NUMBER,
```

```
  MEMBER FUNCTION ODCIINDEXCLOSE(env SYS.ODCIEnv) RETURN NUMBER
);
/

CREATE OR REPLACE TYPE BODY position_im
IS
   STATIC FUNCTION ODCIGETINTERFACES(ifclist OUT SYS.ODCIOBJECTLIST)
       RETURN NUMBER IS
   BEGIN
       ifclist := SYS.ODCIOBJECTLIST(SYS.ODCIOBJECT('SYS','ODCIINDEX2'));
       RETURN ODCICONST.SUCCESS;
   END ODCIGETINTERFACES;
 STATIC FUNCTION ODCIINDEXCREATE (ia SYS.ODCIINDEXINFO, parms VARCHAR2, env SYS.ODCIEnv) RETURN
 NUMBER
  IS
   stmt   VARCHAR2(2000);
  BEGIN
/* Construct the SQL statement */
   stmt := 'Create Table ' || ia.INDEXSCHEMA || '.' || ia.INDEXNAME ||
           '_STORAGE_TAB' || '(col_val, base_rowid, constraint pk PRIMARY KEY ' ||
           '(col_val, base_rowid)) ORGANIZATION INDEX AS SELECT ' ||
           ia.INDEXCOLS(1).COLNAME || ', ROWID FROM ' ||
           ia.INDEXCOLS(1).TABLESCHEMA || '.' || ia.INDEXCOLS(1).TABLENAME;
   EXECUTE IMMEDIATE stmt;
   RETURN ODCICONST.SUCCESS;
  END;
  STATIC FUNCTION ODCIINDEXDROP(ia SYS.ODCIINDEXINFO, env SYS.ODCIEnv) RETURN NUMBER IS
   stmt VARCHAR2(2000);
  BEGIN
/* Construct the SQL statement */
   stmt := 'DROP TABLE ' || ia.INDEXSCHEMA || '.' || ia.INDEXNAME ||
   '_STORAGE_TAB';
/* Execute the statement */
   EXECUTE IMMEDIATE stmt;
   RETURN ODCICONST.SUCCESS;
  END;
  STATIC FUNCTION ODCIINDEXTRUNCATE(ia SYS.ODCIINDEXINFO, env SYS.ODCIEnv) RETURN NUMBER IS
   stmt VARCHAR2(2000);
  BEGIN
/* Construct the SQL statement */
   stmt := 'TRUNCATE TABLE ' || ia.INDEXSCHEMA || '.' || ia.INDEXNAME || '_STORAGE_TAB';

   EXECUTE IMMEDIATE stmt;
   RETURN ODCICONST.SUCCESS;
  END;
  STATIC FUNCTION ODCIINDEXINSERT(ia SYS.ODCIINDEXINFO, rid ROWID,
                          newval NUMBER, env SYS.ODCIEnv) RETURN NUMBER IS
   stmt VARCHAR2(2000);
  BEGIN
/* Construct the SQL statement */
   stmt := 'INSERT INTO ' || ia.INDEXSCHEMA || '.' || ia.INDEXNAME ||
          '_STORAGE_TAB  VALUES (''' || newval || ''' , ''' || rid || ''' )';
/* Execute the SQL statement */
   EXECUTE IMMEDIATE stmt;
   RETURN ODCICONST.SUCCESS;
  END;

  STATIC FUNCTION ODCIINDEXDELETE(ia SYS.ODCIINDEXINFO, rid ROWID, oldval NUMBER,
                                  env SYS.ODCIEnv)
     RETURN NUMBER IS
   stmt VARCHAR2(2000);
  BEGIN
/* Construct the SQL statement */
```

```
   stmt := 'DELETE FROM ' || ia.INDEXSCHEMA || '.' || ia.INDEXNAME ||
          '_STORAGE_TAB WHERE col_val = ''' || oldval || ''' AND base_rowid = ''' || rid || '''';
/* Execute the statement */
   EXECUTE IMMEDIATE stmt;
   RETURN ODCICONST.SUCCESS;
  END;
  STATIC FUNCTION ODCIINDEXUPDATE(ia SYS.ODCIINDEXINFO, rid ROWID, oldval NUMBER,
                           newval NUMBER, env SYS.ODCIEnv) RETURN NUMBER IS
   stmt VARCHAR2(2000);
  BEGIN
/* Construct the SQL statement */
   stmt := 'UPDATE ' || ia.INDEXSCHEMA || '.' || ia.INDEXNAME ||
          '_STORAGE_TAB SET col_val = ''' || newval || ''' WHERE f2 = '''|| rid ||'''';
/* Execute the statement */
   EXECUTE IMMEDIATE stmt;
   RETURN ODCICONST.SUCCESS;
  END;
  STATIC FUNCTION ODCIINDEXSTART(SCTX IN OUT position_im, ia SYS.ODCIINDEXINFO,
                         op SYS.ODCIPREDINFO, qi SYS.ODCIQUERYINFO,
                         strt NUMBER, stop NUMBER, lower_pos NUMBER,
                         upper_pos NUMBER, env SYS.ODCIEnv) RETURN NUMBER IS
    rid              VARCHAR2(5072);
    storage_tab_name VARCHAR2(65);
    lower_bound_stmt VARCHAR2(2000);
    upper_bound_stmt VARCHAR2(2000);
    range_query_stmt VARCHAR2(2000);
    lower_bound      NUMBER;
    upper_bound      NUMBER;
    cnum             INTEGER;
    nrows            INTEGER;

  BEGIN
/* Take care of some error cases.
    The only predicates in which position operator can appear are
       op() = 1     OR
       op() = 0     OR
       op() between 0 and 1
*/
    IF (((strt != 1) AND (strt != 0)) OR
        ((stop != 1) AND (stop != 0)) OR
        ((strt = 1) AND (stop = 0))) THEN
      RAISE_APPLICATION_ERROR(-20101,
                              'incorrect predicate for position_between operator');
    END IF;
    IF (lower_pos > upper_pos) THEN
      RAISE_APPLICATION_ERROR(-20101, 'Upper Position must be greater than or
      equal to Lower Position');
    END IF;
    IF (lower_pos <= 0) THEN
      RAISE_APPLICATION_ERROR(-20101, 'Both Positions must be greater than zero');
    END IF;
    storage_tab_name := ia.INDEXSCHEMA || '.' || ia.INDEXNAME ||
                        '_STORAGE_TAB';
    upper_bound_stmt := 'Select MIN(col_val) FROM (Select /*+ INDEX_DESC(' ||
                        storage_tab_name || ') */ DISTINCT ' ||
                        'col_val FROM ' || storage_tab_name || ' ORDER BY ' ||
                        'col_val DESC) WHERE rownum <= ' || lower_pos;
    EXECUTE IMMEDIATE upper_bound_stmt INTO upper_bound;
    IF (lower_pos != upper_pos) THEN
      lower_bound_stmt := 'Select MIN(col_val) FROM (Select /*+ INDEX_DESC(' ||
                          storage_tab_name || ') */ DISTINCT ' ||
                          'col_val FROM ' || storage_tab_name ||
                          ' WHERE col_val < ' || upper_bound || ' ORDER BY ' ||
```

```
                        'col_val DESC) WHERE rownum <= ' ||
                        (upper_pos - lower_pos);
      EXECUTE IMMEDIATE lower_bound_stmt INTO lower_bound;
    ELSE
      lower_bound := upper_bound;
    END IF;
    IF (lower_bound IS NULL) THEN
      lower_bound := upper_bound;
    END IF;
    range_query_stmt := 'Select base_rowid FROM ' || storage_tab_name ||
                        ' WHERE col_val BETWEEN ' || lower_bound || ' AND ' ||
                       upper_bound;
    cnum := DBMS_SQL.OPEN_CURSOR;
    DBMS_SQL.PARSE(cnum, range_query_stmt, DBMS_SQL.NATIVE);
/* set context as the cursor number */
    SCTX := position_im(cnum, 0, 0, 0);
/* return success */
    RETURN ODCICONST.SUCCESS;
  END;
  MEMBER FUNCTION ODCIINDEXFETCH(SELF IN OUT position_im, nrows NUMBER,
                                 rids OUT SYS.ODCIRIDLIST, env SYS.ODCIEnv)
   RETURN NUMBER IS
    cnum    INTEGER;
    rid_tab DBMS_SQL.Varchar2_table;
    rlist   SYS.ODCIRIDLIST := SYS.ODCIRIDLIST();
    i       INTEGER;
    d       INTEGER;
  BEGIN
    cnum := SELF.curnum;
    IF self.howmany = 0 THEN
      dbms_sql.define_array(cnum, 1, rid_tab, nrows, 1);
      d := DBMS_SQL.EXECUTE(cnum);
    END IF;
    d := DBMS_SQL.FETCH_ROWS(cnum);
    IF d = nrows THEN
      rlist.extend(d);
    ELSE
      rlist.extend(d+1);
    END IF;
    DBMS_SQL.COLUMN_VALUE(cnum, 1, rid_tab);
    for i in 1..d loop
      rlist(i) := rid_tab(i+SELF.howmany);
    end loop;
    SELF.howmany := SELF.howmany + d;
    rids := rlist;
    RETURN ODCICONST.SUCCESS;
  END;
  MEMBER FUNCTION ODCIINDEXCLOSE(env SYS.ODCIEnv) RETURN NUMBER IS
    cnum INTEGER;
  BEGIN
    cnum := SELF.curnum;
    DBMS_SQL.CLOSE_CURSOR(cnum);
    RETURN ODCICONST.SUCCESS;
  END;
END;
/
```

次の手順では、索引タイプに関連付けられる演算子に必要な function_for_position_between ファンクション実装を作成します。(PL/SQL ブロックはカッコで囲んでいます。)

このファンクションは、索引ベースのファンクション評価で使用するためのものです。そのため、索引コンテキストおよびスキャン・コンテキストをパラメータとして指定します。

参照：

- 索引ベースのファンクション実装を作成する場合の詳細は、『Oracle Database データ・カートリッジ開発者ガイド』を参照してください。
- 14-48 ページの「CREATE FUNCTION」および『Oracle Database PL/SQL 言語リファレンス』を参照してください。

```
CREATE OR REPLACE FUNCTION function_for_position_between
                          (col NUMBER, lower_pos NUMBER, upper_pos NUMBER,
                           indexctx IN SYS.ODCIIndexCtx,
                           scanctx IN OUT position_im,
                           scanflg IN NUMBER)
RETURN NUMBER AS
  rid              ROWID;
  storage_tab_name VARCHAR2(65);
  lower_bound_stmt VARCHAR2(2000);
  upper_bound_stmt VARCHAR2(2000);
  col_val_stmt     VARCHAR2(2000);
  lower_bound      NUMBER;
  upper_bound      NUMBER;
  column_value     NUMBER;
BEGIN
  IF (indexctx.IndexInfo IS NOT NULL) THEN
    storage_tab_name := indexctx.IndexInfo.INDEXSCHEMA || '.' ||
                        indexctx.IndexInfo.INDEXNAME || '_STORAGE_TAB';
    IF (scanctx IS NULL) THEN
/* This is the first call. Open a cursor for future calls.
   First, do some error checking
*/
      IF (lower_pos > upper_pos) THEN
        RAISE_APPLICATION_ERROR(-20101,
          'Upper Position must be greater than or equal to Lower Position');
      END IF;
      IF (lower_pos <= 0) THEN
        RAISE_APPLICATION_ERROR(-20101,
          'Both Positions must be greater than zero');
      END IF;
/* Obtain the upper and lower value bounds for the range we're interested in.
*/
      upper_bound_stmt := 'Select MIN(col_val) FROM (Select /*+ INDEX_DESC(' ||
                          storage_tab_name || ') */ DISTINCT ' ||
                          'col_val FROM ' || storage_tab_name || ' ORDER BY ' ||
                          'col_val DESC) WHERE rownum <= ' || lower_pos;
      EXECUTE IMMEDIATE upper_bound_stmt INTO upper_bound;
      IF (lower_pos != upper_pos) THEN
        lower_bound_stmt := 'Select MIN(col_val) FROM (Select /*+ INDEX_DESC(' ||
                            storage_tab_name || ') */ DISTINCT ' ||
                            'col_val FROM ' || storage_tab_name ||
                            ' WHERE col_val < ' || upper_bound || ' ORDER BY ' ||
                            'col_val DESC) WHERE rownum <= ' ||
                            (upper_pos - lower_pos);
        EXECUTE IMMEDIATE lower_bound_stmt INTO lower_bound;
      ELSE
        lower_bound := upper_bound;
      END IF;
      IF (lower_bound IS NULL) THEN
        lower_bound := upper_bound;
      END IF;
/* Store the lower and upper bounds for future function invocations for
   the positions.
*/
      scanctx := position_im(0, 0, lower_bound, upper_bound);
```

```
    END IF;
/* Fetch the column value corresponding to the rowid, and see if it falls
   within the determined range.
*/
    col_val_stmt := 'Select col_val FROM ' || storage_tab_name ||
                    ' WHERE base_rowid = ''' || indexctx.Rid || '''';
    EXECUTE IMMEDIATE col_val_stmt INTO column_value;
    IF (column_value <= scanctx.upper_bound AND
        column_value >= scanctx.lower_bound AND
        scanflg = ODCICONST.RegularCall) THEN
      RETURN 1;
    ELSE
      RETURN 0;
    END IF;
  ELSE
    RAISE_APPLICATION_ERROR(-20101, 'A column that has a domain index of' ||
                            'Position indextype must be the first argument');
  END IF;
END;
/
```

次の手順は、function_for_position_between ファンクションを使用する position_between 演算子を作成します。この演算子には、索引列 NUMBER を最初の引数として指定し、その後に NUMBER の下限および上限を 2 番目および 3 番目の引数として指定します。

参照：「CREATE OPERATOR」（15-32 ページ）

```
CREATE OR REPLACE OPERATOR position_between
   BINDING (NUMBER, NUMBER, NUMBER) RETURN NUMBER
   WITH INDEX CONTEXT, SCAN CONTEXT position_im
   USING function_for_position_between;
```

索引コンテキストおよびスキャン・コンテキストが、索引ベースのファンクション評価に渡されるように、この CREATE OPERATOR 文には WITH INDEX CONTEXT, SCAN CONTEXT position_im 句が含まれています。

次の手順は、position_operator に必要な索引タイプ position_indextype を作成します。

参照：「CREATE INDEXTYPE」（14-73 ページ）

```
CREATE INDEXTYPE position_indextype
   FOR position_between(NUMBER, NUMBER, NUMBER)
   USING position_im;
```

演算子 position_between は、索引ベースのファンクション実装を使用します。そのため、索引情報がファンクション評価に渡されるように、参照列にドメイン索引を定義する必要があります。そのため、最後の手順では、索引タイプ position_indextype を使用してドメイン索引 salary_index を作成します。

参照：「CREATE INDEX」（14-50 ページ）

```
CREATE INDEX salary_index ON employees(salary)
   INDEXTYPE IS position_indextype;
```

これで、演算子 position_between を使用して、元の問合せを次のように書き換えることができます。

```
SELECT last_name, salary FROM employees
   WHERE position_between(salary, 10, 20)=1
   ORDER BY salary DESC, last_name;
```

```
LAST_NAME                     SALARY
------------------------- ----------
Tucker                         10000
King                           10000
Baer                           10000
Bloom                          10000
Fox                             9600
Bernstein                       9500
Sully                           9500
Greene                          9500
Hunold                          9000
Faviet                          9000
McEwen                          9000
Hall                            9000
Hutton                          8800
Taylor                          8600
Livingston                      8400
Gietz                           8300
Chen                            8200
Fripp                           8200
Weiss                           8000
Olsen                           8000
Smith                           8000
Kaufling                        7900
```

SQL 文での XML の使用方法

この項では、XMLType データをデータベースで使用する方法を説明します。

XMLType 表

サンプル・スキーマ oe には表 warehouses が含まれ、この表には XMLType 列 warehouse_spec が含まれます。warehouse_spec 情報を持つ別の表を作成するとします。次の例は、暗黙的に CLOB 列を 1 列のみ持つ非常に単純な XMLType 表を作成します。

```
CREATE TABLE xwarehouses OF XMLTYPE;
```

このような表には、次の文に示すように、XMLType 構文を使用してデータを挿入することができます。（この例で挿入されるデータは、サンプル表 oe.warehouses の warehouse_spec 列にあるデータに対応します。warehouse_id は 1 です。）

```
INSERT INTO xwarehouses VALUES
  (xmltype('<?xml version="1.0"?>
  <Warehouse>
    <WarehouseId>1</WarehouseId>
    <WarehouseName>Southlake, Texas</WarehouseName>
    <Building>Owned</Building>
    <Area>25000</Area>
    <Docks>2</Docks>
    <DockType>Rear load</DockType>
    <WaterAccess>true</WaterAccess>
    <RailAccess>N</RailAccess>
    <Parking>Street</Parking>
    <VClearance>10</VClearance>
  </Warehouse>'));
```

> **参照：** XMLType およびそのメンバー・メソッドの詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

この表の問合せには、次の文を使用します。

```
SELECT e.getClobVal() FROM xwarehouses e;
```

データは暗黙的に CLOB 列に格納されるため、LOB 列に対するすべての制限事項が適用されます。これらの制限事項を回避するには、XMLSchema ベースの表を作成します。XMLSchema は、XML 要素を対応するオブジェクト・リレーショナル・データにマップします。次の例は、XMLSchema をローカルに登録します。XMLSchema（xwarhouses.xsd）には、xwarehouses 表と同じ構造が反映されます。（XMLSchema の宣言では PL/SQL および DBMS_XMLSCHEMA パッケージが使用されています。例では、これらをイタリック体で示しています。）

> **参照：** XMLSchema の作成の詳細は、『Oracle XML DB 開発者ガイド』を参照してください。

```
begin
 dbms_xmlschema.registerSchema(
  'http://www.example.com/xwarehouses.xsd',
  '<schema xmlns="http://www.w3.org/2001/XMLSchema"
       targetNamespace="http://www.example.com/xwarehouses.xsd"
       xmlns:who="http://www.example.com/xwarehouses.xsd"
       version="1.0">

  <simpleType name="RentalType">
   <restriction base="string">
    <enumeration value="Rented"/>
    <enumeration value="Owned"/>
   </restriction>
  </simpleType>

  <simpleType name="ParkingType">
   <restriction base="string">
    <enumeration value="Street"/>
    <enumeration value="Lot"/>
   </restriction>
  </simpleType>

  <element name = "Warehouse">
    <complexType>
     <sequence>
      <element name = "WarehouseId"   type = "positiveInteger"/>
      <element name = "WarehouseName" type = "string"/>
      <element name = "Building"      type = "who:RentalType"/>
      <element name = "Area"          type = "positiveInteger"/>
      <element name = "Docks"         type = "positiveInteger"/>
      <element name = "DockType"      type = "string"/>
      <element name = "WaterAccess"   type = "boolean"/>
      <element name = "RailAccess"    type = "boolean"/>
      <element name = "Parking"       type = "who:ParkingType"/>
      <element name = "VClearance"    type = "positiveInteger"/>
     </sequence>
    </complexType>
  </element>
</schema>',
   TRUE, TRUE, FALSE, FALSE);
end;
/
```

これで、次の例に示すように、XMLSchema ベースの表を作成できます。

```
CREATE TABLE xwarehouses OF XMLTYPE
   XMLSCHEMA "http://www.example.com/xwarehouses.xsd"
   ELEMENT "Warehouse";
```

デフォルトでは、この表はオブジェクト・リレーショナル表として格納されます。そのため、次の例に示すように、この表にデータを挿入できます。(この例で挿入されるデータは、サンプル表 oe.warehouses の warehouse_spec 列にあるデータに対応します。warehouse_id は 1 です。)

```
INSERT INTO xwarehouses VALUES(
   xmltype.createxml('<?xml version="1.0"?>
   <who:Warehouse xmlns:who="http://www.example.com/xwarehouses.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.example.com/xwarehouses.xsd
   http://www.example.com/xwarehouses.xsd">
      <WarehouseId>1</WarehouseId>
      <WarehouseName>Southlake, Texas</WarehouseName>
      <Building>Owned</Building>
      <Area>25000</Area>
      <Docks>2</Docks>
      <DockType>Rear load</DockType>
      <WaterAccess>true</WaterAccess>
      <RailAccess>false</RailAccess>
      <Parking>Street</Parking>
      <VClearance>10</VClearance>
      </who:Warehouse>'));
...
```

XMLSchema ベースの表には制約を定義できます。その場合は、XML 要素 Warehouse 内の適切な属性を参照するように、XMLDATA 疑似列を使用します。

```
ALTER TABLE xwarehouses ADD (PRIMARY KEY(XMLDATA."WarehouseId"));
```

xwarehouses のデータはオブジェクト・リレーショナルに格納されるため、可能な場合に基礎となる記憶域を参照できるように、この XMLType 表への問合せが書き換えられます。そのため、次の問合せでは、前述の例の主キー制約によって作成された索引を使用します。

```
SELECT * FROM xwarehouses x
   WHERE EXISTSNODE(VALUE(x), '/Warehouse[WarehouseId="1"]') = 1,
   'xmlns:who="http://www.example.com/xwarehouses.xsd"') = 1;

SELECT * FROM xwarehouses x
   WHERE EXTRACTVALUE(VALUE(x), '/Warehouse/WarehouseId') = 1,
   'xmlns:who="http://www.example.com/xwarehouses.xsd"') = 1;
```

XMLSchema ベースの表に索引を明示的に作成すると、後続の問合せのパフォーマンスが大幅に向上します。XMLType 表にオブジェクト・リレーショナル・ビューを作成することも、オブジェクト・リレーショナル表に XMLType ビューを作成することもできます。

参照:

- XMLDATA 疑似列の詳細は、3-10 ページの「XMLDATA 疑似列」を参照してください。
- 「XMLType ビューの作成例:」(17-22 ページ)
- 「XMLType 表の索引の作成例:」(14-68 ページ)

XMLType 列

サンプル表 oe.warehouses は、XMLType 型の warehouse_spec 列を使用して作成されました。記憶域は指定していないため、XMLType 列は暗黙的に CLOB として格納されます。この項の例は、2 つのタイプの記憶域を使用して、簡略化した oe.warehouses 表を作成します。

最初の例は、CLOB として格納された XMLType 表を持つ表を作成します。この表では XMLSchema が必要ないため、コンテンツ構造は事前に定義しません。

```
CREATE TABLE xwarehouses (
   warehouse_id        NUMBER,
   warehouse_spec      XMLTYPE)
   XMLTYPE warehouse_spec STORE AS CLOB
   (TABLESPACE example
    STORAGE (INITIAL 6144 NEXT 6144)
    CHUNK 4000
    NOCACHE LOGGING);
```

次の例でも前述の例とほぼ同じ表を作成しますが、指定された XMLSchema によって構造が決められているオブジェクト・リレーショナル XMLType 列に、XMLType データが格納されます。

```
CREATE TABLE xwarehouses (
   warehouse_id    NUMBER,
   warehouse_spec  XMLTYPE)
   XMLTYPE warehouse_spec STORE AS OBJECT RELATIONAL
      XMLSCHEMA "http://www.example.com/xwarehouses.xsd"
      ELEMENT "Warehouse";
```

索引

記号

+（プラス記号）
　自動ストレージ管理ファイル名，8-28

数字

16 進数値
　戻す，2-56
20 世紀，2-62
21 世紀，2-62

A

ABORT LOGICAL STANDBY 句
　ALTER DATABASE，10-36
ABS ファンクション，5-15
ACCESSED GLOBALLY 句
　CREATE CONTEXT，14-9
ACCOUNT LOCK 句
　ALTER USER「CREATE USER」を参照，13-7
　CREATE USER，17-11
ACCOUNT UNLOCK 句
　ALTER USER「CREATE USER」を参照
　CREATE USER，17-11
ACOS ファンクション，5-16
ACTIVATE STANDBY DATABASE 句
　ALTER DATABASE，10-33
ADD DATAFILE 句
　ALTER TABLESPACE，12-87
ADD LOGFILE GROUP 句
　ALTER DATABASE，10-29
ADD LOGFILE MEMBER 句
　ALTER DATABASE，10-13，10-30
ADD LOGFILE THREAD 句
　ALTER DATABASE，10-29
ADD LOGFILE 句
　ALTER DATABASE，10-13
ADD OVERFLOW 句
　ALTER TABLE，12-38
ADD PARTITION 句
　ALTER TABLE，12-59，12-60
ADD PRIMARY KEY 句
　ALTER MATERIALIZED VIEW LOG，11-19
ADD ROWID 句
　ALTER MATERIALIZED VIEW，11-19
　ALTER MATERIALIZED VIEW LOG，11-19
ADD SUPPLEMENTAL LOG DATA 句
　ALTER DATABASE，10-31
ADD SUPPLEMENTAL LOG GROUP 句
　ALTER TABLE，12-34
ADD TEMPFILE 句
　ALTER TABLESPACE，12-87
ADD VALUES 句
　ALTER TABLEMODIFY PARTITION，12-56，12-57
ADD_MONTHS ファンクション，5-16
ADD 句
　ALTER DIMENSION，10-44
　ALTER INDEXTYPE，10-83
　ALTER TABLE，12-39
　ALTER VIEW，13-13
ADMINISTER ANY SQL TUNING SET システム権限，18-37
ADMINISTER DATABASE TRIGGER システム権限，18-43
ADMINISTER SQL TUNING SET システム権限，18-37
ADVISE 句
　ALTER SESSION，11-42
ADVISOR システム権限，18-37
A.D. 日時書式要素，2-61
AD 日時書式要素，2-61
AGENT 句
　CREATE LIBRARY，15-3
ALL EXCEPT 句
　SET ROLE，19-52
ALL PRIVILEGES 句
　GRANT，18-35
　REVOKE，18-87
ALL PRIVILEGES ショートカット
　AUDIT，13-28
ALL_COL_COMMENTS データ・ディクショナリ・ビュー，13-42
ALL_INDEXTYPE_COMMENTS データ・ディクショナリ・ビュー，13-42
ALL_MVIEW_COMMENTS データ・ディクショナリ・ビュー，13-42
ALL_OPERATOR_COMMENTS データ・ディクショナリ・ビュー，13-42
ALL_ROWS ヒント，2-74
ALL_TAB_COMMENTS データ・ディクショナリ・ビュー，13-42
ALLOCATE EXTENT 句
　ALTER CLUSTER，10-6，10-7
　ALTER INDEX，10-70
　ALTER MATERIALIZED VIEW，11-6

ALTER TABLE，12-34
ALLOW CORRUPTION 句
ALTER DATABASERECOVER，10-22
ALL 演算子，7-4
ALL 句
SELECT，19-13
SET CONSTRAINTS，19-49
SET ROLE，19-52
ALL ショートカット
AUDIT，13-28
ALTER ANY CLUSTER システム権限，18-38
ALTER ANY CUBE DIMENSION システム権限，18-41
ALTER ANY CUBE システム権限，18-40
ALTER ANY DIMENSION システム権限，18-38
ALTER ANY INDEXTYPE システム権限，18-39
ALTER ANY INDEX システム権限，18-39
ALTER ANY MATERIALIZED VIEW システム権限，18-39
ALTER ANY MINING MODEL システム権限，18-40
ALTER ANY OPERATOR システム権限，18-41
ALTER ANY OUTLINE システム権限，18-41
ALTER ANY PROCEDURE システム権限，18-41
ALTER ANY ROLE システム権限，18-42
ALTER ANY SEQUENCE システム権限，18-42
ALTER ANY SQL PROFILE システム権限，18-37
ALTER ANY TABLE システム権限，18-43
ALTER ANY TRIGGER システム権限，18-43
ALTER ANY TYPE システム権限，18-44
ALTER CLUSTER 文，10-5
ALTER DATABASE システム権限，18-38
ALTER DATABASE 文，10-9
ALTER DIMENSION 文，10-43
ALTER DISKGROUP 文，10-46
ALTER FLASHBACK ARCHIVE 文，10-60
ALTER FUNCTION 文，10-63
ALTER INDEXTYPE 文，10-82
ALTER INDEX 文，10-64
ALTER JAVA CLASS 文，10-85
ALTER JAVA SOURCE 文，10-85
ALTER MATERIALIZED VIEW LOG 文，11-16
ALTER MATERIALIZED VIEW 文，11-2
ALTER OPERATOR 文，11-22
ALTER OUTLINE 文，11-25
ALTER PACKAGE 文，11-27
ALTER PROCEDURE 文，11-28
ALTER PROFILE システム権限，18-42
ALTER PROFILE 文，11-29
ALTER RESOURCE COST システム権限，18-42
ALTER RESOURCE COST 文，11-32
ALTER ROLE 文，11-34
ALTER ROLLBACK SEGMENT システム権限，18-42
ALTER ROLLBACK SEGMENT 文，11-36
ALTER SEQUENCE 文，11-39
ALTER SESSION システム権限，18-42
ALTER SESSION 文，11-41
ALTER SNAPSHOT LOG「ALTER MATERIALIZED VIEW LOG」を参照
ALTER SNAPSHOT「ALTER MATERIALIZED VIEW」を参照
ALTER SYSTEM システム権限，18-38
ALTER SYSTEM 文，11-52
ALTER TABLESPACE システム権限，18-43
ALTER TABLESPACE 文，12-82
ALTER TABLE 文，12-2
ALTER TRIGGER 文，13-2
ALTER TYPE 文，13-4
ALTER USER システム権限，18-44
ALTER USER 文，13-5
ALTER VIEW 文，13-12
alter_external_table_clause
ALTER TABLE，12-17
ALTER オブジェクト権限
OLAP キューブ，18-48
OLAP キューブ・ディメンション，18-48
順序，18-49
表，18-49
マイニング・モデル，18-47
A.M. 日時書式要素，2-61
AM 日時書式要素，2-61
ANALYZE ANY システム権限，18-45
ANALYZE CLUSTER 文，13-14
ANALYZE INDEX 文，13-14
ANALYZE TABLE 文，13-14
ANCILLARY TO 句
CREATE OPERATOR，15-33
AND DATAFILES 句
DROP TABLESPACE，18-9，18-10
AND 条件，7-8
ANSI「ANSI（米国規格協会）」を参照
ANSI（米国規格協会），B-1
規格，1-2，B-2
サポートしているデータ型，2-5
データ型，2-27
Oracle データ型への変換，2-27
ANY 演算子，7-4
APPENDCHILDXML ファンクション，5-17
APPEND ヒント，2-74
AQ_ADMINISTRATOR_ROLE ロール，18-46
AQ_USER_ROLE ロール，18-46
ARCHIVE LOG 句
ALTER SYSTEM，11-55
ARCHIVELOG 句
ALTER DATABASE，10-13，10-27
CREATE CONTROLFILE，14-14
CREATE DATABASE，14-21
ASCII
キャラクタ・セット，2-37
ASCIISTR ファンクション，5-17
ASCII ファンクション，5-18
ASC 句
CREATE INDEX，14-60
ASIN ファンクション，5-19
ASM_DISKGROUPS 初期化パラメータ
ALTER SYSTEM での設定，10-57，14-40
ASM_DISKSTRING 初期化パラメータ
ALTER SYSTEM での設定，14-42
ASSOCIATE STATISTICS 文，13-21
AS 句
CREATE JAVA，14-79
AS 副問合せ句
CREATE MATERIALIZED VIEW，15-20
CREATE TABLE，16-56
CREATE VIEW，17-17
ATAN2 ファンクション，5-20
ATAN ファンクション，5-19

ATTRIBUTE 句
ALTER DIMENSION，10-44
CREATE DIMENSION，14-34，14-36
AUDIT ANY システム権限，18-45
AUDIT SYSTEM システム権限，18-38
AUTHENTICATED BY 句
CREATE DATABASE LINK，14-31
AUTHENTICATED 句
ALTER USER，13-9
AUTHENTICATION REQUIRED 句
ALTER USER，13-9
AUTHID CURRENT_USER 句
ALTER JAVA，10-86
CREATE JAVA，14-77，14-78
AUTHID DEFINER 句
ALTER JAVA，10-86
CREATE JAVA，14-77，14-78
AUTOALLOCATE 句
CREATE TABLESPACE，16-78
AUTOEXTEND 句
ALTER DATABASE，10-13
CREATE DATABASE，14-18
AVG ファンクション，5-20

B

BACKUP ANY TABLE システム権限，18-43
BACKUP CONTROLFILE 句
ALTER DATABASE，10-14，10-32
B.C. 日時書式要素，2-61
BC 日時書式要素，2-61
BECOME USER システム権限，18-45
BEGIN BACKUP 句
ALTER DATABASE，10-24
ALTER TABLESPACE，12-87
BETWEEN 条件，7-20
BFILE
データ型，2-25
ロケータ，2-25
BFILENAME ファンクション，5-21
BIN_TO_NUM ファンクション，5-22
BINDING 句
CREATE OPERATOR，15-32
BITAND ファンクション，5-23
BITMAP 句
CREATE INDEX，14-56
BLOB データ型，2-25
トランザクションのサポート，2-25
BLOCKSIZE 句
CREATE TABLESPACE，16-76
BUFFER_POOL パラメータ
STORAGE 句，8-48
BUILD DEFERRED 句
CREATE MATERIALIZED VIEW，15-15
BUILD IMMEDIATE 句
CREATE MATERIALIZED VIEW，15-15
BY proxy 句
AUDIT，13-28
BY SESSION 句
AUDIT，13-29
BY user 句
AUDIT，13-28
BYTE キャラクタ・セマンティクス，2-9，2-10
BYTE の長さセマンティクス，12-44

C

CACHE READS 句
ALTER TABLE，12-42
CREATE TABLE，16-53
CACHE 句
ALTER MATERIALIZED VIEW，11-10
ALTER MATERIALIZED VIEW LOG，11-19
ALTER TABLE，16-53
CREATE CLUSTER，14-6
CREATE MATERIALIZED VIEW，15-15
CREATE MATERIALIZED VIEW LOG，15-29
CACHE パラメータ
ALTER SEQUENCE「CREATE SEQUENCE」を参照，11-39
CREATE SEQUENCE，15-66
CACHE ヒント，2-75
CALL 文，13-37
CARDINALITY ファンクション，5-25
CASCADE CONSTRAINTS 句
DROP CLUSTER，17-33
DROP TABLE，18-7
DROP TABLESPACE，18-10
DROP VIEW，18-17
REVOKE，18-88
CASCADE 句
CREATE TABLE，16-55
DROP PROFILE，17-58
DROP USER，18-15
CASE 式，6-5
検索，6-5
単純，6-5
CAST ファンクション，5-25
MULTISET パラメータ，5-25
CATSEARCH 条件，4-2，7-2
CEIL ファンクション，5-28
CHANGE CATEGORY 句
ALTER OUTLINE，11-26
CHANGE NOTIFICATION システム権限，18-45
CHARACTER SET パラメータ
CREATE CONTROLFILE，14-15
CREATE DATABASE，14-20
CHARTOROWID ファンクション，5-28
CHAR キャラクタ・セマンティクス，2-9，2-10
CHAR データ型，2-9
VARCHAR2 への変換，2-54
CHAR の長さセマンティクス，12-44
CHECK DATAFILES 句
ALTER SYSTEM，11-57
CHECKPOINT 句
ALTER SYSTEM，11-57
CHECK 句
CREATE TABLE，16-29
制約，8-11
CHECK 制約，8-11
CHR ファンクション，5-29
CHUNK 句
ALTER TABLE，12-42
CREATE TABLE，16-38

CLEAR LOGFILE 句
ALTER DATABASE，10-13，10-28
CLOB データ型，2-26
トランザクションのサポート，2-26
CLOSE DATABASE LINK 句
ALTER SESSION，11-42
CLUSTER_ID ファンクション，5-30
CLUSTER_PROBABILITY ファンクション，5-31
CLUSTER_SET ファンクション，5-32
CLUSTER 句
ANALYZE，13-17
CREATE INDEX，14-57
CREATE TABLE，16-36
TRUNCATE，19-57
CLUSTER ヒント，2-75
COALESCE SUBPARTITION 句
ALTER TABLE，12-56
COALESCE 句
ALTER INDEX，10-76
ALTER TABLE，12-38，12-39，12-56
ALTER TABLESPACE，12-86
パーティション，12-60
COALESCE ファンクション，5-35
様々な CASE 式，5-35
COLLECT ファンクション，5-36
COLUMN_VALUE 疑似列，3-6
COLUMNS 句
ASSOCIATE STATISTICS，13-21，13-22
COMMENT ANY MINING MODEL システム権限，18-40
COMMENT ANY TABLE システム権限，18-45
COMMENT 句
COMMIT，13-45
COMMENT 文，13-41
COMMIT IN PROCEDURE 句
ALTER SESSION，11-42
COMMIT TO SWITCHOVER 句
ALTER DATABASE，10-35
COMMIT 文，13-44
COMPILE 句
ALTER DIMENSION，10-45
ALTER JAVA SOURCE，10-86
ALTER MATERIALIZED VIEW，11-13
ALTER VIEW，13-13
CREATE JAVA，14-77
COMPOSE ファンクション，5-36
COMPOSITE_LIMIT パラメータ
ALTER PROFILE，11-29
CREATE PROFILE，15-50
COMPRESS 句
ALTER INDEX ...REBUILD，10-74
CREATE TABLE，16-34
CONCAT ファンクション，5-37
CONNECT BY 句
SELECT，9-4，19-23
問合せおよび副問合せ，19-23
CONNECT THROUGH 句
ALTER USER，13-9
CONNECT TO 句
CREATE DATABASE LINK，14-30
CONNECT_BY_ISCYCLE 疑似列，3-2
CONNECT_BY_ISLEAF 疑似列，3-2
CONNECT_BY_ROOT 演算子，4-5
CONNECT_TIME パラメータ
ALTER PROFILE，11-29
ALTER RESOURCE COST，11-32
CONNECT 句
SELECT および副問合せ，19-9
CONNECT ロール，18-46
CONSIDER FRESH 句
ALTER MATERIALIZED VIEW，11-14
CONSTRAINT(S) セッション・パラメータ，11-46
CONTAINS 条件，4-2，7-2
CONTROLFILE REUSE 句
CREATE DATABASE，14-19
controlfile_clauses
ALTER DATABASE，10-14
CONVERT ファンクション，5-38
CORR_K ファンクション，5-42
CORR_S ファンクション，5-42
CORR ファンクション，5-39
COSH ファンクション，5-43
COS ファンクション，5-42
COUNT ファンクション，5-43
COVAR_POP ファンクション，5-45
COVAR_SAMP ファンクション，5-46
CPU_PER_CALL パラメータ
ALTER PROFILE，11-29
CREATE PROFILE，15-49
CPU_PER_SESSION パラメータ
ALTER PROFILE，11-29
ALTER RESOURCE COST，11-32
CREATE PROFILE，15-49
CREATE ANY CLUSTER システム権限，18-38
CREATE ANY CONTEXT システム権限，18-38
CREATE ANY CUBE BUILD PROCESS システム権限，18-41
CREATE ANY CUBE DIMENSION システム権限，18-41
CREATE ANY CUBE システム権限，18-40
CREATE ANY DIMENSION システム権限，18-38
CREATE ANY DIRECTORY システム権限，18-38
CREATE ANY INDEXTYPE システム権限，18-39
CREATE ANY INDEX システム権限，18-39
CREATE ANY JOB システム権限，18-39
CREATE ANY LIBRARY システム権限，18-39
CREATE ANY MATERIALIZED VIEW システム権限，18-39
CREATE ANY MEASURE FOLDER システム権限，18-40
CREATE ANY MINING MODEL システム権限，18-40
CREATE ANY OPERATOR システム権限，18-41
CREATE ANY OUTLINE システム権限，18-41
CREATE ANY PROCEDURE システム権限，18-41
CREATE ANY SEQUENCE システム権限，18-42
CREATE ANY SQL PROFILE システム権限，18-37
CREATE ANY SYNONYM システム権限，18-42
CREATE ANY TABLE システム権限，18-43
CREATE ANY TRIGGER システム権限，18-43
CREATE ANY TYPE システム権限，18-44
CREATE ANY VIEW システム権限，18-44
CREATE CLUSTER システム権限，18-38
CREATE CLUSTER 文，14-2
CREATE CONTEXT 文，14-8
CREATE CONTROLFILE 文，14-10
CREATE CUBE BUILD PROCESS システム権限，18-41
CREATE CUBE DIMENSION システム権限，18-41

CREATE CUBE システム権限，18-40
CREATE DATABASE LINK システム権限，18-38
CREATE DATABASE LINK 文，14-28
CREATE DATABASE 文，14-16
CREATE DATAFILE 句
　ALTER DATABASE，10-12，10-25
CREATE DIMENSION
　システム権限，18-38
CREATE DIMENSION 文，14-33
CREATE DIRECTORY 文，14-38
CREATE DISKGROUP 文，14-40
CREATE EXTERNAL JOB システム権限，18-39
CREATE FLASHBACK ARCHIVE 文，14-45
CREATE FUNCTION 文，14-48
CREATE INDEX
　文，14-50
CREATE INDEXTYPE
　文，14-73
CREATE INDEXTYPE システム権限，18-39
CREATE JAVA 文，14-76
CREATE JOB システム権限，18-39
CREATE LIBRARY システム権限，18-39
CREATE LIBRARY 文，15-2
CREATE MATERIALIZED VIEW LOG 文，15-26
CREATE MATERIALIZED VIEW システム権限，18-39
CREATE MATERIALIZED VIEW 文，15-4
CREATE MEASURE FOLDER システム権限，18-40
CREATE MINING MODEL システム権限，18-40
CREATE OPERATOR システム権限，18-41
CREATE OPERATOR 文，15-32
CREATE OUTLINE 文，15-35
CREATE PACKAGE BODY 文，15-41
CREATE PACKAGE 文，15-39
CREATE PFILE 文，15-43
CREATE PROCEDURE システム権限，18-41
CREATE PROCEDURE 文，15-45
CREATE PROFILE システム権限，18-42
CREATE PROFILE 文，15-47
CREATE PUBLIC DATABASE LINK システム権限，
　18-38
CREATE PUBLIC SYNONYM システム権限，18-42
CREATE RESTORE POINT 文，15-53
CREATE ROLE システム権限，18-42
CREATE ROLE 文，15-56
CREATE ROLLBACK SEGMENT システム権限，18-42
CREATE ROLLBACK SEGMENT 文，15-59
CREATE SCHEMA 文，15-62
CREATE SEQUENCE システム権限，18-42
CREATE SEQUENCE 文，15-64
CREATE SESSION システム権限，18-42
CREATE SPFILE 文，15-68
CREATE STANDBY CONTROLFILE 句
　ALTER DATABASE，10-14，10-32
CREATE SYNONYM システム権限，18-42
CREATE SYNONYM 文，16-2
CREATE TABLESPACE システム権限，18-43
CREATE TABLESPACE 文，16-71
CREATE TABLE システム権限，18-43
CREATE TABLE 文，16-6
CREATE TRIGGER システム権限，18-43
CREATE TRIGGER 文，16-85
CREATE TYPE BODY 文，17-5
CREATE TYPE システム権限，18-44
CREATE TYPE 文，17-3
CREATE USER システム権限，18-44
CREATE USER 文，17-7
CREATE VIEW システム権限，18-44
CREATE VIEW 文，17-13
CUBE_TABLE ファンクション，5-47
CUBE 句
　SELECT 文，19-24
CUME_DIST ファンクション，5-48
CURRENT_DATE ファンクション，5-50
CURRENT_SCHEMA セッション・パラメータ，11-46
CURRENT_TIMESTAMP ファンクション，5-50
CURRENT_USER 句
　CREATE DATABASE LINK，14-30
CURRVAL 疑似列，3-3，15-64
CURSOR_SHARING_EXACT ヒント，2-75
CURSOR 式，6-7
CV ファンクション，5-51
CYCLE パラメータ
　ALTER SEQUENCE「CREATE SEQUENCE」を参
　　照，11-39
　CREATE SEQUENCE，15-66

D

DATAFILE OFFLINE 句
　ALTER DATABASE，10-26
DATAFILE ONLINE 句
　ALTER DATABASE，10-26
DATAFILE RESIZE 句
　ALTER DATABASE，10-27
DATAFILE 句
　ALTER DATABASE，10-12，10-26
　CREATE DATABASE，14-25
DATAOBJ_TO_PARTITION ファンクション，5-52
DATE データ型，2-17
　ユリウス，2-17
DATE 列
　日時列への変換，12-43
DAY 日時書式要素，2-61
DB2 データ型，2-27
　制限事項，2-28
DBA_2PC_PENDING データ・ディクショナリ・
　　ビュー，11-42
DBA_COL_COMMENTS データ・ディクショナリ・
　　ビュー，13-42
DBA_INDEXTYPE_COMMENTS データ・ディクショナ
　　リ・ビュー，13-42
DBA_MVIEW_COMMENTS データ・ディクショナリ・
　　ビュー，13-42
DBA_OPERATOR_COMMENTS データ・ディクショナ
　　リ・ビュー，13-42
DBA_ROLLBACK_SEGS データ・ディクショナリ・
　　ビュー，17-61
DBA_TAB_COMMENTS データ・ディクショナリ・
　　ビュー，13-42
DBA ロール，18-46
DBMS_ROWID パッケージ
　拡張 ROWID，2-27
DBTIMEZONE ファンクション，5-53
DDL「データ定義言語（DDL)」を参照
DEALLOCATE UNUSED 句
　ALTER CLUSTER，10-6，10-7

ALTER INDEX，10-65
ALTER TABLE，12-34
DEBUG ANY PROCEDURE システム権限，18-38
DEBUG オブジェクト権限
オブジェクト型，18-47
ビュー，18-49
表，18-49
ファンクション、プロシージャまたはパッケージ，18-48
DECODE ファンクション，5-53
DECOMPOSE ファンクション，5-54
DEFAULT COST 句
ASSOCIATE STATISTICS，13-22，13-23
DEFAULT ROLE 句
ALTER USER，13-8
DEFAULT SELECTIVITY 句
ASSOCIATE STATISTICS，13-22，13-23
DEFAULT TABLESPACE 句
ALTER DATABASE，10-37
ALTER USER，13-8
ALTER USER「CREATE USER」を参照
CREATE USER，17-9
DEFAULT TEMPORARY TABLESPACE 句
ALTER DATABASE，10-37
CREATE DATABASE，14-18
DEFAULT 記憶域句
ALTER TABLESPACE，12-85
CREATE TABLESPACE，16-77
DEFAULT 句
ALTER TABLE，12-41
CREATE TABLE，16-26，16-29
DEFAULT プロファイル
ユーザーへの割当て，17-58
DEFERRABLE 句
制約，8-13
DEFERRED 句
SET CONSTRAINTS，19-50
DELETE ANY CUBE DIMENSION システム権限，18-41
DELETE ANY MEASURE FOLDER システム権限，18-40
DELETE ANY TABLE システム権限，18-43
DELETE STATISTICS 句
ANALYZE，13-19
DELETE_CATALOG_ROLE ロール，18-46
DELETEXML ファンクション，5-55
DELETE オブジェクト権限
OLAP キューブ・ディメンション，18-48
OLAP メジャー・フォルダ，18-48
ビュー，18-49
表，18-49
DELETE 文，17-23
エラー・ロギング，17-28
DENSE_RANK ファンクション，5-56
DEREF ファンクション，5-58
DESC 句
CREATE INDEX，14-60
DISABLE ALL TRIGGERS 句
ALTER TABLE，12-72
DISABLE DISTRIBUTED RECOVERY 句
ALTER SYSTEM，11-58
DISABLE NOVALIDATE 制約状態，8-15
DISABLE PARALLEL DML 句
ALTER SESSION，11-43
DISABLE QUERY REWRITE 句
ALTER MATERIALIZED VIEW，11-13
CREATE MATERIALIZED VIEW，15-19
DISABLE RESTRICTED SESSION 句
ALTER SYSTEM，11-61
DISABLE RESUMABLE 句
ALTER SESSION，11-44
DISABLE ROW MOVEMENT 句
ALTER TABLE，12-36
CREATE TABLE，16-15，16-55
DISABLE STORAGE IN ROW 句
ALTER TABLE，12-42
CREATE TABLE，16-38
DISABLE TABLE LOCK 句
ALTER TABLE，12-72
DISABLE VALIDATE 制約状態，8-15
DISABLE 句
ALTER INDEX，10-75
CREATE TABLE，16-54
DISASSOCIATE STATISTICS 文，17-30
DISCONNECT SESSION 句
ALTER SYSTEM，11-57
DISTINCT 句
SELECT，19-13
DML「データ操作言語（DML）」を参照
domain_index_clause
CREATE INDEX，14-53
DOWNGRADE 句
ALTER DATABASE，10-19
DRIVING_SITE ヒント，2-75
DROP ANY CLUSTER システム権限，18-38
DROP ANY CONTEXT システム権限，18-38
DROP ANY CUBE BUILD PROCESS システム権限，18-41
DROP ANY CUBE DIMENSION システム権限，18-41
DROP ANY CUBE システム権限，18-40
DROP ANY DIMENSION システム権限，18-38
DROP ANY DIRECTORY システム権限，18-38
DROP ANY INDEXTYPE システム権限，18-39
DROP ANY INDEX システム権限，18-39
DROP ANY LIBRARY システム権限，18-39
DROP ANY MATERIALIZED VIEW システム権限，18-39
DROP ANY MEASURE FOLDER システム権限，18-40
DROP ANY MINING MODEL システム権限，18-40
DROP ANY OPERATOR システム権限，18-41
DROP ANY OUTLINE システム権限，18-41
DROP ANY PROCEDURE システム権限，18-41
DROP ANY ROLE システム権限，18-42
DROP ANY SEQUENCE システム権限，18-42
DROP ANY SYNONYM システム権限，18-42
DROP ANY TABLE システム権限，18-43
DROP ANY TRIGGER システム権限，18-43
DROP ANY TYPE システム権限，18-44
DROP ANY VIEW システム権限，18-44
DROP CLUSTER 文，17-32
DROP COLUMN 句
ALTER TABLE，12-46
DROP CONSTRAINT 句
ALTER TABLE，12-51
DROP CONTEXT 文，17-34
DROP DATABASE LINK 文，17-36

DROP DATABASE 文，17-35
DROP DIMENSION 文，17-37
DROP DIRECTORY 文，17-38
DROP DISKGROUP 文，17-39
DROP FLASHBACK ARCHIVE 文，17-41
DROP FUNCTION 文，17-42
DROP INDEXTYPE 文，17-46
DROP INDEX 文，17-44
DROP JAVA 文，17-47
DROP LIBRARY 文，17-48
DROP LOGFILE MEMBER 句
  ALTER DATABASE，10-13，10-31
DROP LOGFILE 句
  ALTER DATABASE，10-13，10-30
DROP MATERIALIZED VIEW LOG 文，17-51
DROP MATERIALIZED VIEW 文，17-49
DROP OPERATOR 文，17-53
DROP OUTLINE 文，17-54
DROP PACKAGE BODY 文，17-55
DROP PACKAGE 文，17-55
DROP PARTITION 句
  ALTER INDEX，10-79
  ALTER TABLE，12-61
DROP PRIMARY 制約句
  ALTER TABLE，12-51
DROP PROCEDURE 文，17-57
DROP PROFILE システム権限，18-42
DROP PROFILE 文，17-58
DROP PUBLIC DATABASE LINK システム権限，18-38
DROP PUBLIC SYNONYM システム権限，18-42
DROP RESTORE POINT 文，17-59
DROP ROLE 文，17-60
DROP ROLLBACK SEGMENT システム権限，18-42
DROP ROLLBACK SEGMENT 文，17-61
DROP SEQUENCE 文，18-2
DROP SUPPLEMENTAL LOG DATA 句
  ALTER DATABASE，10-31
DROP SUPPLEMENTAL LOG GROUP 句
  ALTER TABLE，12-34
DROP SYNONYM 文，18-3
DROP TABLESPACE システム権限，18-43
DROP TABLESPACE 文，18-8
DROP TABLE 文，18-5
DROP TRIGGER 文，18-11
DROP TYPE BODY 文，18-14
DROP TYPE 文，18-12
DROP UNIQUE 制約句
  ALTER TABLE，12-51
DROP USER システム権限，18-44
DROP USER 文，18-15
DROP VALUES 句
  ALTER TABLEMODIFY PARTITION，12-56，12-57
DROP VIEW 文，18-17
DROP 句
  ALTER DIMENSION，10-44
  ALTER INDEXTYPE，10-83
DROP 制約句
  ALTER VIEW，13-13
DUAL ダミー表，2-99，9-14
DUMP ファンクション，5-59
DYNAMIC_SAMPLING ヒント，2-76
DY 日時書式要素，2-61

E

EBCDIC キャラクタ・セット，2-37
EMPTY_BLOB ファンクション，5-60
EMPTY_CLOB ファンクション，5-60
ENABLE ALL TRIGGERS 句
  ALTER TABLE，12-72
ENABLE DISTRIBUTED RECOVERY 句
  ALTER SYSTEM，11-58
ENABLE NOVALIDATE 制約状態，8-14
ENABLE PARALLEL DML 句
  ALTER SESSION，11-43
ENABLE QUERY REWRITE 句
  ALTER MATERIALIZED VIEW，11-13
  CREATE MATERIALIZED VIEW，15-19
ENABLE RESTRICTED SESSION 句
  ALTER SYSTEM，11-61
ENABLE RESUMABLE 句
  ALTER SESSION，11-44
ENABLE ROW MOVEMENT 句
  ALTER TABLE，12-36
  CREATE TABLE，16-15，16-55
ENABLE STORAGE IN ROW 句
  ALTER TABLE，12-42
  CREATE TABLE，16-38
ENABLE TABLE LOCK 句
  ALTER TABLE，12-72
ENABLE VALIDATE 制約状態，8-14
ENABLE 句
  ALTER INDEX，10-75
  ALTER TRIGGER，13-3
  CREATE TABLE，16-54
END BACKUP 句
  ALTER DATABASE，10-24
  ALTER DATABASEDATAFILE，10-27
  ALTER TABLESPACE，12-87
ERROR_ON_OVERLAP_TIME セッション・パラメータ，11-46
EXCEPTIONS INTO 句
  ALTER TABLE，12-67
EXCHANGE PARTITION 句
  ALTER TABLE，12-25，12-66
EXCHANGE SUBPARTITION 句
  ALTER TABLE，12-25，12-66
EXCLUDING NEW VALUES 句
  ALTER MATERIALIZED VIEW LOG，11-20
  CREATE MATERIALIZED VIEW LOG，15-30
EXCLUSIVE ロック・モード，18-71
EXECUTE ANY CLASS システム権限，18-39
EXECUTE ANY INDEXTYPE システム権限，18-39
EXECUTE ANY OPERATOR システム権限，18-41
EXECUTE ANY PROCEDURE システム権限，18-42
EXECUTE ANY PROGRAM システム権限，18-39
EXECUTE ANY TYPE システム権限，18-44
EXECUTE_CATALOG_ROLE ロール，18-46
EXECUTE オブジェクト権限
  演算子，18-48
  オブジェクト型，18-47
  ライブラリ，18-47
EXEMPT ACCESS POLICY システム権限，18-45
EXISTSNODE ファンクション，5-61
EXISTS 条件，7-19，7-21
EXP_FULL_DATABASE ロール，18-46

EXPLAIN PLAN 文，18-19
EXP ファンクション，5-62
EXTENT MANAGEMENT DICTIONARY 句
　CREATE TABLESPACE，16-78
EXTENT MANAGEMENT LOCAL 句
　CREATE DATABASE，14-22
　CREATE TABLESPACE，16-78
EXTENT MANAGEMENT 句
　CREATE DATABASE，14-18
　CREATE TABLESPACE，16-73，16-78
EXTRACTVALUE ファンクション，5-65
EXTRACT（XML）ファンクション，5-65
EXTRACT（日時）ファンクション，5-62

F

FACT ヒント，2-76
FAILED_LOGIN_ATTEMPTS パラメータ
　ALTER PROFILE，11-30
　CREATE PROFILE，15-50
FEATURE_ID ファンクション，5-66
FEATURE_SET ファンクション，5-67
FEATURE_VALUE ファンクション，5-69
FIPS
　準拠，B-26
　フラグ付け，11-47
FIRST_ROWS(n) ヒント，2-77
FIRST_VALUE ファンクション，5-72
FIRST ファンクション，5-71
FLAGGER セッション・パラメータ，11-47
FLASHBACK ANY TABLE システム権限，18-40，18-43，18-44
FLASHBACK ARCHIVE ADMINISTER システム権限，18-38
FLASHBACK ARCHIVE オブジェクト権限，18-47
FLASHBACK DATABASE 文，18-22
FLASHBACK TABLE 文，18-25
FLOOR ファンクション，5-74
FLUSH BUFFER_CACHE 句
　ALTER SYSTEM，11-59
FLUSH SHARED POOL 句
　ALTER SYSTEM，11-58
FM 書式モデル修飾子，2-63
FOR UPDATE 句
　CREATE MATERIALIZED VIEW，15-19
　SELECT，19-12，19-30
FORCE ANY TRANSACTION システム権限，18-45
FORCE LOGGING 句
　ALTER DATABASE，10-28
　ALTER TABLESPACE，12-89
　CREATE CONTROLFILE，14-14
　CREATE DATABASE，14-22
　CREATE TABLESPACE，16-77
FORCE PARALLEL DML 句
　ALTER SESSION，11-43
FORCE TRANSACTION システム権限，18-45
FORCE 句
　COMMIT，13-46
　CREATE VIEW，17-15
　DISASSOCIATE STATISTICS，17-31
　DROP INDEX，17-45
　DROP INDEXTYPE，17-46
　DROP OPERATOR，17-53
　DROP TYPE，18-13
　REVOKE，18-88
　ROLLBACK，18-25，18-93
FOR 句
　CREATE INDEXTYPE，14-74
　EXPLAIN PLAN，18-20，18-25
FREELIST GROUPS パラメータ
　STORAGE 句，8-47
FREELISTS パラメータ
　STORAGE 句，8-47
FREEPOOLS パラメータ
　LOB 記憶域，16-39
FROM COLUMNS 句
　DISASSOCIATE STATISTICS，17-31
FROM FUNCTIONS 句
　DISASSOCIATE STATISTICS，17-31
FROM INDEXES 句
　DISASSOCIATE STATISTICS，17-31
FROM INDEXTYPES 句
　DISASSOCIATE STATISTICS，17-31
FROM PACKAGES 句
　DISASSOCIATE STATISTICS，17-31
FROM TYPES 句
　DISASSOCIATE STATISTICS，17-31
FROM_TZ ファンクション，5-74
FROM 句
　問合せ，9-11
FULL ヒント，2-77
FX 書式モデル修飾子，2-64

G

general_recovery 句
　ALTER DATABASE，10-10，10-19
GLOBAL PARTITION BY HASH 句
　CREATE INDEX，14-62
GLOBAL PARTITION BY RANGE 句
　CREATE INDEX，14-53，14-62
GLOBAL QUERY REWRITE システム権限，18-40
GLOBAL TEMPORARY 句
　CREATE TABLE，16-24
GLOBAL_TOPIC_ENABLED システム・パラメータ，11-64
GRANT ANY OBJECT PRIVILEGE システム権限，18-45
GRANT ANY PRIVILEGE システム権限，18-45
GRANT ANY ROLE システム権限，18-42
GRANT CONNECT THROUGH 句
　ALTER USER，13-6，13-7，13-9
GRANT 句
　ALTER USER，13-9
GRAPHIC データ型
　DB2，2-28
　SQL/DS，2-28
GREATEST ファンクション，5-75
GROUP BY 句
　CUBE 拡張，19-24
　ROLLUP 拡張，19-24
　SELECT および副問合せ，19-9，19-24
　重複するグループ化の識別，5-75
GROUP_ID ファンクション，5-75
GROUPING SETS 句
　SELECT および副問合せ，19-24
GROUPING_ID ファンクション，5-77

GROUPING ファンクション，5-76
GUARD ALL 句
　ALTER DATABASE，10-40
GUARD NONE 句
　ALTER DATABASE，10-40
GUARD STANDBY 句
　ALTER DATABASE，10-40
GUARD 句
　ALTER DATABASE，10-40
　　上書き，11-42

H

HASH IS 句
　CREATE CLUSTER，14-6
HASHKEYS 句
　CREATE CLUSTER，14-5
HASH ヒント，2-77
HAVING 条件
　GROUP BY 句，19-25
HEXTORAW ファンクション，5-78
HIERARCHY 句
　CREATE DIMENSION，14-34，14-35

I

IDENTIFIED BY 句
　ALTER ROLE「CREATE ROLE」を参照，11-34
　CREATE DATABASE LINK，14-31
　SET ROLE，19-52
IDENTIFIED EXTERNALLY 句
　ALTER ROLE「CREATE ROLE」を参照
　ALTER USER「CREATE USER」を参照
　CREATE ROLE，15-57
　CREATE USER，17-8
IDENTIFIED GLOBALLY 句
　ALTER ROLE「CREATE ROLE」を参照，11-34
　CREATE ROLE，15-57
　CREATE USER，17-9
IDLE_TIME パラメータ
　ALTER PROFILE，11-29
IEEE754
　Oracle の規格準拠，2-14
　浮動小数点算術，2-14
IMMEDIATE 句
　SET CONSTRAINTS，19-49
IMP_FULL_DATABASE ロール，18-46
INCLUDING CONTENTS 句
　DROP TABLESPACE，18-9
INCLUDING DATAFILES 句
　ALTER DATABASE TEMPFILE DROP 句，10-27
INCLUDING NEW VALUES 句
　ALTER MATERIALIZED VIEW LOG，11-20
　CREATE MATERIALIZED VIEW LOG，15-30
INCLUDING TABLES 句
　DROP CLUSTER，17-32
INCREMENT BY 句
　ALTER SEQUENCE「CREATE SEQUENCE」を参照
INCREMENT BY パラメータ
　CREATE SEQUENCE，15-65
INDEX_ASC ヒント，2-78
INDEX_COMBINE ヒント，2-79
INDEX_DESC ヒント，2-79
INDEX_FFS ヒント，2-79
INDEX_JOIN ヒント，2-80
INDEX_SS_ASC ヒント，2-80
INDEX_SS_DESC ヒント，2-81
INDEX_SS ヒント，2-80
INDEXTYPE 句
　CREATE INDEX，14-53，14-65
INDEX オブジェクト権限
　表，18-49
INDEX 句
　ANALYZE，13-16
　CREATE CLUSTER，14-5
INDEX ヒント，2-78
INITCAP ファンクション，5-78
INITIALIZED EXTERNALLY 句
　CREATE CONTEXT，14-9
INITIALIZED GLOBALLY 句
　CREATE CONTEXT，14-9
INITIALLY DEFERRED 句
　制約，8-14
INITIALLY IMMEDIATE 句
　制約，8-14
INITIAL パラメータ
　STORAGE 句，8-45
INITRANS パラメータ
　ALTER CLUSTER，10-6
　ALTER INDEX，10-71
　ALTER MATERIALIZED VIEW LOG，11-17
　ALTER TABLE，12-33
　CREATE INDEX「CREATE TABLE」を参照，14-60
　CREATE MATERIALIZED VIEW LOG「CREATE TABLE」を参照
　CREATE MATERIALIZED VIEW「CREATE TABLE」を参照
　CREATE TABLE，8-41
INSERT
　ダイレクト・パスと従来型，18-53
INSERT ANY CUBE DIMENSION システム権限，18-41
INSERT ANY MEASURE FOLDER システム権限，18-40
INSERT ANY TABLE システム権限，18-43
INSERTCHILDXMLAFTER ファンクション，5-80
INSERTCHILDXMLBEFORE ファンクション，5-81
INSERTCHILDXML ファンクション，5-79
INSERTXMLAFTER ファンクション，5-82
INSERTXMLBEFORE ファンクション，5-82
INSERT オブジェクト権限
　OLAP キューブ・ディメンション，18-48
　OLAP メジャー・フォルダ，18-48
　ビュー，18-49
　表，18-49
INSERT 句
　MERGE，18-73
INSERT 文，18-53
　append，2-74
　エラー・ロギング，18-63
INSTANCE セッション・パラメータ，11-47
INSTR2 ファンクション，5-83
INSTR4 ファンクション，5-83
INSTRB ファンクション，5-83
INSTRC ファンクション，5-83
INSTR ファンクション，5-83
INTERSECT 集合演算子，4-6，19-29
INTERVAL DAY TO SECOND データ型，2-20

INTERVAL YEAR TO MONTH データ型，2-19
INTO 句
　EXPLAIN PLAN，18-20
　INSERT，18-57
INVALIDATE GLOBAL INDEXES 句
　ALTER TABLE，12-69
IN 条件，7-22
IS [NOT] EMPTY 条件，7-12
IS ANY 条件，7-9
IS NOT NULL 演算子，7-18
IS NULL 演算子，7-18
IS OF type 条件，7-23
IS PRESENT 条件，7-10
ISO「ISO（国際標準化機構）」を参照，1-2
ISOLATION_LEVEL セッション・パラメータ，11-47
ISO（国際標準化機構），B-1
　規格，1-2，B-2
ITERATION_NUMBER ファンクション，5-85

J

Java
　Java ソース・スキーマ・オブジェクト
　　作成，14-77
　クラス
　　削除，17-47
　　作成，14-76，14-77
　　変換，10-85，14-77
　スキーマ・オブジェクト
　　名前解決，14-79
　ソース
　　コンパイル，10-85，14-77
　　削除，17-47
　　作成，14-76
　リソース
　　削除，17-47
　　作成，14-76，14-77
JOIN KEY 句
　ALTER DIMENSION，10-44
　CREATE DIMENSION，14-35
JOIN 句
　CREATE DIMENSION，14-34

K

KEEP キーワード
　FIRST ファンクション，5-71
　LAST ファンクション，5-71
　集計ファンクションとの使用，5-9
KILL SESSION 句
　ALTER SYSTEM，11-58

L

LAG ファンクション，5-86
LAST_DAY ファンクション，5-87
LAST_VALUE ファンクション，5-88
LAST ファンクション，5-87
LEADING ヒント，2-81
LEAD ファンクション，5-90
LEAST ファンクション，5-91
LENGTH2 ファンクション，5-91
LENGTH4 ファンクション，5-91
LENGTHB ファンクション，5-91
LENGTHC ファンクション，5-91
LENGTH ファンクション，5-91
LEVEL 疑似列，3-3，19-23
　階層問合せ，3-3
LEVEL 句
　ALTER DIMENSION，10-43
　CREATE DIMENSION，14-33，14-34
LIKE 演算子で使用する %（パーセント），7-15
LIKE 条件，7-14
LIST CHAINED ROWS 句
　ANALYZE，13-18
LNNVL ファンクション，5-93
LN ファンクション，5-92
LOB
　外部，2-24
　記憶域
　　インライン，16-38
　　属性，16-37
　　特性，8-41
　キャッシュへの値の保存，12-42，16-53
　旧バージョンの保存，16-38，16-39
　処理されるバイト数，16-38
　属性の初期化，2-25
　ディレクトリの指定，14-38
　内部，2-24
　表領域
　　定義，16-30
　物理属性の変更，12-49
　列
　　LONG と LONG RAW の違い，2-24
　　移入，2-25
　ロギング属性，16-31
　ロケータ，2-24
LOB 記憶域句
　ALTER MATERIALIZED VIEW，11-4，11-9
　ALTER TABLE，12-13，12-42
　CREATE MATERIALIZED VIEW，15-9，15-10，
　　15-13，15-14
　CREATE TABLE，16-12，16-37
　パーティション，12-42
LOB データ型，2-24
LOB 列
　LONG 列からの作成，2-15，12-44
　圧縮，16-40
　暗号化，16-40
　記憶域の変更，12-42
　結合の制限，9-10
　制限事項，2-25
　重複除外，16-39
　追加，12-39
　プロパティの定義
　　マテリアライズド・ビュー，15-9
　変更，12-43
　マテリアライズド・ビューの記憶特性，11-9
LOCALTIMESTAMP ファンクション，5-94
LOCAL 句
　CREATE INDEX，14-55，14-64
LOCK ANY TABLE システム権限，18-43
LOCK TABLE 文，18-69
LOGFILE GROUP 句
　CREATE CONTROLFILE，14-13

logfile_clauses
ALTER DATABASE，10-13
LOGFILE 句
CREATE DATABASE，14-20
LOGGING 句
ALTER INDEX，10-71
ALTER MATERIALIZED VIEW，11-9
ALTER MATERIALIZED VIEW LOG，11-19
ALTER TABLE，12-33
ALTER TABLESPACE，12-89
CREATE MATERIALIZED VIEW，15-13
CREATE MATERIALIZED VIEW LOG，15-28
CREATE TABLE，16-31
CREATE TABLESPACE，16-76
LOGICAL_READS_PER_CALL パラメータ
ALTER PROFILE，11-29
LOGICAL_READS_PER_SESSION パラメータ
ALTER PROFILE，11-29
ALTER RESOURCE COST，11-32
LogMiner
サプリメンタル・ロギング，12-34，16-29
LOG ファンクション，5-95
LONG RAW データ型，2-23
CHAR データからの変換，2-24
LONG VARGRAPHIC データ型
DB2，2-28
SQL/DS，2-28
LONG データ型，2-15
トリガー内，2-16
LONG 列
LOB への変換，2-15，12-44
参照先，2-15
制限事項，2-15
テキスト文字列の格納，2-15
ドメイン索引，12-44
ビュー定義の格納，2-15
LOWER ファンクション，5-95
LPAD ファンクション，5-96
LTRIM ファンクション，5-96

M

MAKE_REF ファンクション，5-97
MANAGE SCHEDULER システム権限，18-39
MANAGE TABLESPACE システム権限，18-43
MANAGED STANDBY RECOVERY 句
ALTER DATABASE，10-22
MAPPING TABLE 句
ALTER TABLE，12-58，12-71
MATCHES 条件，4-2，7-2
MAXDATAFILES パラメータ
CREATE CONTROLFILE，14-14
CREATE DATABASE，14-20
MAXEXTENTS パラメータ
STORAGE 句，8-46
MAXINSTANCES パラメータ
CREATE CONTROLFILE，14-14
CREATE DATABASE，14-20
MAXLOGFILES パラメータ
CREATE CONTROLFILE，14-13
CREATE DATABASE，14-21
MAXLOGHISTORY パラメータ
CREATE CONTROLFILE，14-14
CREATE DATABASE，14-21
MAXLOGMEMBERS パラメータ
CREATE CONTROLFILE，14-14
CREATE DATABASE，14-21
MAXSIZE 句
ALTER DATABASE，10-13
MAXTRANS パラメータ
physical_attributes_clause，8-41
MAXVALUE パラメータ
ALTER SEQUENCE「CREATE SEQUENCE」を参照，11-39
CREATE SEQUENCE，15-66
MAX ファンクション，5-98
MEDIAN ファンクション，5-99
MEMBER 条件，7-12
MERGE ANY VIEW システム権限，18-44
MERGE PARTITIONS 句
ALTER TABLE，12-65
merge_insert_clause
MERGE，18-74
MERGE ヒント，2-82
MERGE 文，18-72，18-73
エラー・ロギング，18-74
更新，18-73
挿入，18-74
MINEXTENTS パラメータ
STORAGE 句，8-46
MINIMIZE RECORDS PER BLOCK 句
ALTER TABLE，12-36
MINIMUM EXTENT 句
ALTER TABLESPACE，12-85
CREATE TABLESPACE，16-76
MINUS 集合演算子，4-6，19-29
MINVALUE パラメータ
ALTER SEQUENCE「CREATE SEQUENCE」を参照，11-39
CREATE SEQUENCE，15-66
MIN ファンクション，5-101
MODEL_MIN_ANALYSIS ヒント，2-82
MODEL 句
SELECT，19-10，19-25
MODE 句
LOCK TABLE，18-70
MODIFY CONSTRAINT 句
ALTER TABLE，12-10，12-51
ALTER VIEW，13-13
MODIFY DEFAULT ATTRIBUTES 句
ALTER INDEX，10-68，10-77
ALTER TABLE，12-53
MODIFY LOB 記憶域句
ALTER MATERIALIZED VIEW，11-5，11-9
ALTER TABLE，12-49
MODIFY NESTED TABLE 句
ALTER TABLE，12-10，12-49
MODIFY PARTITION 句
ALTER INDEX，10-78
ALTER MATERIALIZED VIEW，11-9
ALTER TABLE，12-54
MODIFY scoped_table_ref_constraint 句
ALTER MATERIALIZED VIEW，11-10
MODIFY SUBPARTITION 句
ALTER INDEX，10-69，10-79

MODIFY VARRAY 句
ALTER TABLE，12-16，12-50
MODIFY 句
ALTER TABLE，12-43
MOD ファンクション，5-102
MONITORING USAGE 句
ALTER INDEX，10-76
MONITOR ヒント，2-82
MONTHS_BETWEEN ファンクション，5-103
MONTH 日時書式要素，2-61
MON 日時書式要素，2-61
MOUNT 句
ALTER DATABASE，10-18
MOVE ONLINE 句
ALTER TABLE，12-70
MOVE SUBPARTITION 句
ALTER TABLE，12-58
MOVE 句
ALTER TABLE，12-30，12-70
MTS.「共有サーバー」を参照
MULTISET EXCEPT 演算子，4-7
MULTISET INTERSECT 演算子，4-7
MULTISET UNION 演算子，4-8
MULTISET 演算子，4-6
MULTISET EXCEPT，4-7
MULTISET INTERSECT，4-7
MULTISET UNION，4-8
MULTISET パラメータ
CAST ファンクション，5-25

N

NAMED 句
CREATE JAVA，14-78
NAME 句
SET TRANSACTION，19-54
NANVL ファンクション，5-103
NATIONAL CHARACTER SET パラメータ
CREATE DATABASE，14-20
NCHAR データ型，2-10
NCHR ファンクション，5-104
NCLOB データ型，2-26
トランザクションのサポート，2-26
NESTED TABLE 句
ALTER TABLE，12-11，12-41
CREATE TABLE，16-11，16-42
NEW_TIME ファンクション，5-105
NEXT_DAY ファンクション，5-106
NEXTVAL 疑似列，3-3，15-64
NEXT 句
ALTER MATERIALIZED VIEW ...REFRESH，11-12
NEXT パラメータ
STORAGE 句，8-45
NLS_CHARSET_DECL_LEN ファンクション，5-106
NLS_CHARSET_ID ファンクション，5-107
NLS_CHARSET_NAME ファンクション，5-107
NLS_DATE_LANGUAGE 初期化パラメータ，2-61
NLS_INITCAP ファンクション，5-108
NLS_LANGUAGE 初期化パラメータ，2-61，9-10
NLS_LENGTH_SEMANTICS 初期化パラメータ
上書き，2-10
NLS_LOWER ファンクション，5-109
NLS_SORT 初期化パラメータ，9-10
NLS_TERRITORY 初期化パラメータ，2-61
NLS_UPPER ファンクション，5-111
NLSSORT ファンクション，5-109
NLS 文字ファンクション，5-4
NO FORCE LOGGING 句
ALTER DATABASE，10-28
ALTER TABLESPACE，12-89
NO_EXPAND ヒント，2-83
NO_FACT ヒント，2-84
NO_INDEX_FFS ヒント，2-84
NO_INDEX_SS ヒント，2-85
NO_INDEX ヒント，2-84
NO_MERGE ヒント，2-85
NO_MONITOR ヒント，2-85
NO_PARALLEL_INDEX，2-86
NO_PARALLEL ヒント，2-86
NO_PUSH_PRED ヒント，2-86
NO_PUSH_SUBQ ヒント，2-87
NO_PX_JOIN_FILTER ヒント，2-87
NO_QUERY_TRANSFORMATION ヒント，2-87
NO_RESULT_CACHE ヒント，2-87
NO_REWRITE ヒント，2-87
NO_STAR_TRANSFORMATION ヒント，2-88
NO_UNNEST ヒント，2-88
NO_USE_HASH ヒント，2-88
NO_USE_MERGE ヒント，2-88
NO_USE_NL ヒント，2-89
NO_XML_QUERY_REWRITE ヒント，2-89
NO_XMLINDEX_REWRITE ヒント，2-89
NOAPPEND ヒント，2-83
NOARCHIVELOG 句
ALTER DATABASE，10-13，10-28
CREATE CONTROLFILE，14-14
CREATE DATABASE，10-19，14-21
NOAUDIT 文，18-76
NOCACHE 句
ALTER CLUSTER，10-7
ALTER MATERIALIZED VIEW，11-10
ALTER MATERIALIZED VIEW LOG，11-19
ALTER SEQUENCE「CREATE SEQUENCE」を参照
ALTER TABLE，16-53
CREATE CLUSTER，14-6
CREATE MATERIALIZED VIEW，15-15
CREATE MATERIALIZED VIEW LOG，15-29
CREATE SEQUENCE，15-66
NOCACHE ヒント，2-83
NOCOMPRESS 句
ALTER INDEX ...REBUILD，10-74
CREATE INDEX，14-61
CREATE TABLE，16-34
NOCYCLE パラメータ
ALTER SEQUENCE「CREATE SEQUENCE」を参照，11-39
CREATE SEQUENCE，15-66
NOFORCE 句
CREATE JAVA，14-77
CREATE VIEW，17-15
NOLOGGING モード
強制ロギング・モード，8-35
パーティション・オブジェクト，8-35
非パーティション・オブジェクト，8-35
NOMAXVALUE パラメータ
ALTER SEQUENCE「CREATE SEQUENCE」を参照

CREATE SEQUENCE，15-66
NOMINIMIZE RECORDS PER BLOCK 句
ALTER TABLE，12-36
NOMINVALUE パラメータ
ALTER SEQUENCE「CREATE SEQUENCE」を参照，11-39
CREATE SEQUENCE，15-66
NOMONITORING USAGE 句
ALTER INDEX，10-76
NONE 句
SET ROLE，19-52
NOORDER パラメータ
ALTER SEQUENCE「CREATE SEQUENCE」を参照，11-39
CREATE SEQUENCE，15-67
NOPARALLEL_INDEX ヒント，2-86
NOPARALLEL 句
CREATE INDEX，8-38，16-54
NOPARALLEL ヒント，2-86
NORELY 句
制約，8-15
NORESETLOGS 句
CREATE CONTROLFILE，14-13
NOREVERSE パラメータ
ALTER INDEX ...REBUILD，10-73，10-74
NOREWRITE ヒント，2-87
NOROWDEPENDENCIES 句
CREATE CLUSTER，14-6
CREATE TABLE，16-54
NOSORT 句
ALTER INDEX，14-61
NOT DEFERRABLE 句
制約，8-13
NOT IDENTIFIED 句
ALTER ROLE「CREATE ROLE」を参照
CREATE ROLE，15-57
NOT IN 副問合せ
NOT EXISTS 副問合せへの変換，5-93
NOT NULL 句
CREATE TABLE，16-29
NOT 条件，7-8
NOWAIT 句
LOCK TABLE，18-71
NTILE ファンクション，5-111
NULL，2-67
0（ゼロ）との違い，2-67
条件，2-68
表，2-68
比較条件，2-68
ファンクション，2-68
NULLIF ファンクション，5-112
CASE 式の書式，5-112
NULL 関連ファンクション，5-8
NULL 条件，7-18
NUMBER データ型，2-11
VARCHAR2 への変換，2-54
位取り，2-11
精度，2-11
NUMTODSINTERVAL ファンクション，5-113
NUMTOYMINTERVAL ファンクション，5-114
NVARCHAR2 データ型，2-10
NVL2 ファンクション，5-115
NVL ファンクション，5-115

O

OBJECT IDENTIFIER 句
CREATE TABLE，16-58
OBJECT_ID 疑似列，3-7，16-58，16-59，17-16，17-19
OBJECT_VALUE 疑似列，3-7
ODCIIndexInsert メソッド
索引タイプのサポート，10-83，14-75
OFFLINE 句
ALTER TABLESPACE，12-90
CREATE TABLESPACE，16-78
OF 句
CREATE VIEW，17-16
OIDINDEX 句
CREATE TABLE，16-59
OID「オブジェクト識別子」を参照，2-30
OLAP ファンクション
CUBE_TABLE，5-47
ON COMMIT REFRESH オブジェクト権限
マテリアライズド・ビュー，18-47
ON COMMIT REFRESH システム権限，18-40
ON COMMIT 句
CREATE TABLE，16-30
ON DEFAULT 句
AUDIT，13-29
NOAUDIT，18-78
ON DELETE CASCADE 句
制約，8-10
ON DELETE SET NULL 句
制約，8-10
ON DIRECTORY 句
AUDIT，13-29
NOAUDIT，18-78
ON MINING MODEL 句
AUDIT，13-29
ON object 句
NOAUDIT，18-78
REVOKE，18-88
ON PREBUILT TABLE
CREATE MATERIALIZED VIEW，15-12
ONLINE 句
ALTER TABLESPACE，12-90
CREATE INDEX，14-62
CREATE TABLESPACE，16-78
ONLINE パラメータ
ALTER INDEX ...REBUILD，10-74
ON 句
CREATE OUTLINE，15-37
OPEN READ ONLY 句
ALTER DATABASE，10-19
OPEN READ WRITE 句
ALTER DATABASE，10-18
OPEN 句
ALTER DATABASE，10-18
OPT_PARAM ヒント，2-89
OPTIMAL パラメータ
STORAGE 句，8-48
OR REPLACE 句
CREATE CONTEXT，14-8
CREATE DIRECTORY，14-39
CREATE FUNCTION，14-49，14-77
CREATE LIBRARY，15-2
CREATE OUTLINE，15-36

CREATE PACKAGE，15-40
CREATE PACKAGE BODY，15-42
CREATE PROCEDURE，15-46
CREATE TRIGGER，16-85
CREATE TYPE，17-4
CREATE TYPE BODY，17-6
CREATE VIEW，17-15
ORA_HASH ファンクション，5-116
ORA_ROWSCN 疑似列，3-8
Oracle Call Interface，1-4
Oracle Expression Filter
演算子，4-2
条件，7-2
Oracle Text
CATSEARCH，4-2，7-2
CONTAINS，4-2，7-2
MATCHES，4-2，7-2
組込み条件，4-2，7-2
ドメイン索引の作成，14-65
Oracle のツール製品
SQL のサポート，1-4
Oracle の予約語，D-1
ORDAudio データ型，2-34
ORDDicom データ型，2-35
ORDDoc データ型，2-35
ORDER BY 句
SELECT，9-10，19-12，19-29
ROWNUM，3-9
問合せ，9-10
ORDER SIBLINGS BY 句
SELECT，19-29
ORDERED ヒント，2-90
ORDER 句
ALTER SEQUENCE「CREATE SEQUENCE」を参照，11-39
ORDER パラメータ
CREATE SEQUENCE，15-66
ORDImageSignature データ型，2-35
ORDImage データ型，2-34
ORDVideo データ型，2-35
ORGANIZATION EXTERNAL 句
CREATE TABLE，16-33，16-34
ORGANIZATION HEAP 句
CREATE TABLE，16-32
ORGANIZATION INDEX 句
CREATE TABLE，16-32
OR 条件，7-8，7-9
out-of-line 制約
CREATE TABLE，16-29
OVERFLOW 句
ALTER INDEX，10-69
ALTER TABLE，12-38
CREATE TABLE，16-34
OVER 句
分析ファンクション，5-10

P

PARALLEL_INDEX ヒント，2-91
PARALLEL 句
ALTER CLUSTER，10-6，10-7
ALTER INDEX，10-71
ALTER MATERIALIZED VIEW，11-6，11-9
ALTER MATERIALIZED VIEW LOG，11-18，11-19
ALTER TABLE，12-70
CREATE CLUSTER，14-6
CREATE INDEX，14-62
CREATE MATERIALIZED VIEW，15-11，15-15
CREATE MATERIALIZED VIEW LOG，15-28，15-29
CREATE TABLE，16-23，16-54
PARALLEL ヒント，2-90
PARAMETERS 句
CREATE INDEX，14-66
PARTITION BY HASH 句
CREATE TABLE，16-47
PARTITION BY LIST 句
CREATE TABLE，16-48
PARTITION BY RANGE 句
CREATE TABLE，16-17，16-45
PARTITION ...LOB 記憶域句
ALTER TABLE，12-42
partition_storage_clause
ALTER TABLE，12-14
PARTITION 句
ANALYZE，13-16
CREATE INDEX，14-63
CREATE TABLE，16-46
DELETE，17-26
INSERT，18-58
LOCK TABLE，18-70
UPDATE，19-65
PASSWORD EXPIRE 句
ALTER USER「CREATE USER」を参照
CREATE USER，17-11
PASSWORD_GRACE_TIME パラメータ
ALTER PROFILE，11-30
CREATE PROFILE，15-51
PASSWORD_LIFE_TIME パラメータ
ALTER PROFILE，11-30
CREATE PROFILE，15-50
PASSWORD_LOCK_TIME パラメータ
ALTER PROFILE，11-30
CREATE PROFILE，15-50
PASSWORD_REUSE_MAX パラメータ
ALTER PROFILE，11-30
CREATE PROFILE，15-50
PASSWORD_REUSE_TIME パラメータ
ALTER PROFILE，11-30
CREATE PROFILE，15-50
PASSWORD_VERIFY_FUNCTION パラメータ
ALTER PROFILE，11-30
CREATE PROFILE，15-51
PATH_VIEW，7-19
PCT_ACCESS_DIRECT 統計情報
索引構成表，13-15
PCTFREE パラメータ
ALTER CLUSTER，10-6
ALTER INDEX，10-71
ALTER MATERIALIZED VIEW LOG，11-17
ALTER TABLE，12-33
CREATE MATERIALIZED VIEW LOG「CREATE TABLE」を参照
CREATE MATERIALIZED VIEW「CREATE TABLE」を参照
CREATE TABLE，8-40

PCTINCREASE パラメータ
STORAGE 句，8-46
PCTTHRESHOLD パラメータ
CREATE TABLE，16-33
PCTUSED パラメータ
ALTER CLUSTER，10-6
ALTER INDEX，10-71
ALTER MATERIALIZED VIEW LOG，11-17
ALTER TABLE，12-33
CREATE INDEX「CREATE TABLE」を参照
CREATE MATERIALIZED VIEW LOG「CREATE TABLE」を参照，15-28
CREATE MATERIALIZED VIEW「CREATE TABLE」を参照，15-13
CREATE TABLE，8-40
PCTVERSION パラメータ
LOB 記憶域，16-38
LOB 記憶域句，12-49
PERCENT_RANK ファンクション，5-118
PERCENTILE_CONT ファンクション，5-119
PERCENTILE_DISC ファンクション，5-121
PERMANENT 句
ALTER TABLESPACE，12-91
physical_attributes_clause
ALTER CLUSTER，10-5
ALTER INDEX，10-71
ALTER MATERIALIZED VIEW LOG，11-17
ALTER TABLE，12-33
CREATE CLUSTER，14-3
CREATE MATERIALIZED VIEW，15-8
CREATE TABLE，16-15，16-30
PLAN_TABLE サンプル表，18-19
P.M. 日時書式要素，2-61
PM 日時書式要素，2-61
POSIX 正規表現規格，C-1
POWERMULTISET_BY_CARDINALITY ファンクション，5-124
POWERMULTISET ファンクション，5-123
POWER ファンクション，5-122
PQ_DISTRIBUTE ヒント，2-91
PREDICTION_BOUNDS ファンクション，5-127
PREDICTION_COST ファンクション，5-128
PREDICTION_DETAILS ファンクション，5-130
PREDICTION_PROBABILITY ファンクション，5-131
PREDICTION_SET ファンクション，5-133
PREDICTION ファンクション，5-125
PREPARE TO SWITCHOVER 句
ALTER DATABASE，10-35
PRESENTNNV ファンクション，5-135
PRESENTV ファンクション，5-136
PREVIOUS ファンクション，5-137
PRIMARY KEY 句
CREATE TABLE，16-29
制約，8-9
PRIOR 演算子，4-5
PRIOR 句
階層問合せ，9-3
PRIVATE_SGA パラメータ
ALTER PROFILE，11-29
ALTER RESOURCE COST，11-33
PRIVATE 句
CREATE OUTLINE，15-36
PROFILE 句
ALTER USER「CREATE USER」を参照，13-7
CREATE USER，17-10
PUBLIC 句
CREATE OUTLINE，15-36
CREATE SYNONYM，16-3
DROP DATABASE LINK，17-36
PURGE 文，18-80
PUSH_PRED ヒント，2-92
PUSH_SUBQ ヒント，2-93
PX_JOIN_FILTER ヒント，2-93

Q

QB_NAME ヒント，2-93
QUERY REWRITE オブジェクト権限
マテリアライズド・ビュー，18-47
QUERY REWRITE システム権限，18-39
QUIESCE RESTRICTED 句
ALTER SYSTEM，11-60
QUOTA 句
ALTER USER「CREATE USER」を参照
CREATE USER，17-10

R

RANK ファンクション，5-138
RATIO_TO_REPORT ファンクション，5-140
RAWTOHEX ファンクション，5-141
RAWTONHEX ファンクション，5-141
RAW データ型，2-23
CHAR データからの変換，2-24
READ ONLY 句
ALTER TABLESPACE，12-90
READ WRITE 句
ALTER TABLESPACE，12-90
READ オブジェクト権限
マテリアライズド・ディレクトリ，18-47
REBUILD PARTITION 句
ALTER INDEX，10-73
REBUILD SUBPARTITION 句
ALTER INDEX，10-73
REBUILD UNUSABLE LOCAL INDEXES 句
ALTER TABLE，12-68
REBUILD 句
ALTER INDEX，10-72
ALTER OUTLINE，11-26
RECOVER AUTOMATIC 句
ALTER DATABASE，10-20
RECOVER CANCEL 句
ALTER DATABASE，10-10，10-22
RECOVER CONTINUE 句
ALTER DATABASE，10-10，10-22
RECOVER DATABASE 句
ALTER DATABASE，10-10，10-20
RECOVER DATAFILE 句
ALTER DATABASE，10-10，10-21
RECOVER LOGFILE 句
ALTER DATABASE，10-10，10-21
RECOVER MANAGED STANDBY DATABASE 句
ALTER DATABASE，10-11
RECOVER STANDBY DATAFILE 句
ALTER DATABASE，10-21

RECOVER STANDBY TABLESPACE 句
ALTER DATABASE，10-21
RECOVER TABLESPACE 句
ALTER DATABASE，10-10，10-21
RECOVERABLE，10-72，16-32
「LOGGING 句」を参照
RECOVERY_CATALOG_OWNER ロール，18-46
recovery_clauses
ALTER DATABASE，10-10
RECOVER 句
ALTER DATABASE，10-19
REDO ログ，10-18
アーカイブ位置，11-56
アーカイブ・モードの指定，14-21
グループの切替え，11-59
サイズ，8-31
再利用，8-31
削除，10-27，10-30
指定，8-26，14-20
メディア・リカバリ用，10-21
自動アーカイブ，11-55
開始，11-56
停止，11-56
自動名前生成，10-19，10-20
手動アーカイブ，11-55
SCN，11-55
グループ番号，11-56
現行，11-55
順序番号，11-55
すべて，11-56
次，11-56
消去，10-27
スレッドの使用可能化および使用禁止化，10-27
追加，10-27，10-29
変更の削除，10-18
メンバー
既存のグループへの追加，10-30
削除，10-31
名前の変更，10-25
論理スタンバイ・データベースへの適用，10-36
REDO ログ・ファイル
指定，8-26
制御ファイルのための指定，14-11
REF，2-30，8-11
DANGLING，13-17
オブジェクト識別子のコンテナ，2-30
検証，13-17
更新，13-17
REFERENCES オブジェクト権限
ビュー，18-50
表，18-49
REFERENCES 句
CREATE TABLE，16-29
REFRESH COMPLETE 句
ALTER MATERIALIZED VIEW，11-11
CREATE MATERIALIZED VIEW，15-15
REFRESH FAST 句
ALTER MATERIALIZED VIEW，11-11
CREATE MATERIALIZED VIEW，15-15
REFRESH FORCE 句
ALTER MATERIALIZED VIEW，11-11
CREATE MATERIALIZED VIEW，15-15
REFRESH ON COMMIT 句
ALTER MATERIALIZED VIEW，11-11
CREATE MATERIALIZED VIEW，15-15
REFRESH ON DEMAND 句
ALTER MATERIALIZED VIEW，11-12
CREATE MATERIALIZED VIEW，15-15
REFRESH 句
ALTER MATERIALIZED VIEW，11-8，11-10
CREATE MATERIALIZED VIEW，15-8
REFTOHEX ファンクション，5-142
REF 制約
ALTER TABLE，12-41
マテリアライズド・ビューの有効範囲の定義，11-8
REF 表制約，8-11
CREATE TABLE，16-29
REF ファンクション，5-142
REF 列
指定，16-29
範囲の最限定，11-10
表や列からの指定，16-29
REGEXP_COUNT ファンクション，5-143
REGEXP_INSTR ファンクション，5-144
REGEXP_LIKE 条件，7-17
REGEXP_REPLACE ファンクション，5-147
REGEXP_SUBSTR ファンクション，5-149
REGISTER LOGFILE 句
ALTER DATABASE，10-34
REGISTER 句
ALTER SYSTEM，11-62
REGR_AVGX ファンクション，5-150
REGR_AVGY ファンクション，5-150
REGR_COUNT ファンクション，5-150
REGR_INTERCEPT ファンクション，5-150
REGR_R2 ファンクション，5-150
REGR_SLOPE ファンクション，5-150
REGR_SXX ファンクション，5-150
REGR_SXY ファンクション，5-150
REGR_SYY ファンクション，5-150
RELY 句
制約，8-15
REMAINDER ファンクション，5-155
REMOTE_LOGIN_PASSWORDFILE 初期化パラメータ
データベース，14-16
RENAME CONSTRAINT 句
ALTER TABLE，12-51
RENAME DATAFILE 句
ALTER TABLESPACE，12-87
RENAME FILE 句
ALTER DATABASE，10-9，10-25
RENAME GLOBAL_NAME 句
ALTER DATABASE，10-38
RENAME PARTITION 句
ALTER INDEX，10-78
ALTER TABLE，12-62
RENAME SUBPARTITION 句
ALTER INDEX，10-78
ALTER TABLE，12-62
RENAME 句
ALTER INDEX，10-76
ALTER OUTLINE，11-26
ALTER TABLE，12-37
ALTER TABLESPACE，12-86
ALTER TRIGGER，13-3

RENAME 文，18-82
REPLACE ファンクション，5-155
RESET COMPATIBILITY 句
　ALTER DATABASE，10-9
RESETLOGS パラメータ
　CREATE CONTROLFILE，14-13
RESOLVER 句
　ALTER JAVA CLASS，10-86
　ALTER JAVA SOURCE，10-86
　CREATE JAVA，14-79
RESOLVE 句
　ALTER JAVA CLASS，10-86
　CREATE JAVA，14-77
RESOURCE_VIEW，7-19
RESOURCE ロール，18-46
RESTRICTED SESSION システム権限，18-38，18-42
RESULT_CACHE ヒント，2-93
RESUMABLE システム権限，18-45
RESUME 句
　ALTER SYSTEM，11-59
RETENTION パラメータ
　LOB 記憶域，16-39
RETURNING 句
　DELETE，17-28
　INSERT，18-55
　UPDATE，19-63，19-64，19-68
RETURN 句
　CREATE OPERATOR，15-33
REUSE 句
　CREATE CONTROLFILE，14-12
　ファイル指定，8-31
REVERSE 句
　CREATE INDEX，14-61
REVERSE パラメータ
　ALTER INDEX ...REBUILD，10-73，10-74
REVOKE CONNECT THROUGH 句
　ALTER USER，13-6，13-7，13-9
REVOKE 句
　ALTER USER，13-9
REVOKE 文，18-84
REWRITE ヒント，2-94
ROLLBACK 文，18-92
ROLLUP 句
　SELECT 文，19-24
ROUND ファンクション
　書式モデル，5-248
　数値ファンクション，5-156
　日付ファンクション，5-157
ROW EXCLUSIVE ロック・モード，18-71
ROW SHARE ロック・モード，18-70
ROW_NUMBER ファンクション，5-158
ROWDEPENDENCIES 句
　CREATE CLUSTER，14-6
　CREATE TABLE，16-54
ROWID，2-26
　外部キー表，2-27
　拡張
　　基本，2-27
　　直接利用できない，2-27
　索引構成表，2-27
　使用方法，3-8
　説明，2-26
　非物理，2-27
ROWIDTOCHAR ファンクション，5-159
ROWIDTONCHAR ファンクション，5-160
ROWID 疑似列，2-26，2-27，3-8
ROWID データ型，2-26
ROWNUM 疑似列，3-9
　使用方法，3-10
RPAD ファンクション，5-160
RR 日時書式要素，2-62
RTRIM ファンクション，5-161

S

SAMPLE 句
　SELECT，19-17
　SELECT および副問合せ，19-8
SAVEPOINT 文，19-2
SCHEMA 句
　CREATE JAVA，14-78
SCOPE FOR 句
　ALTER MATERIALIZED VIEW，11-8
　CREATE MATERIALIZED VIEW，15-12
SDO_GEOMETRY データ型，2-33
SDO_GEORASTER データ型，2-33
SDO_TOPO_GEOMETRY データ型，2-34
SEGMENT MANAGEMENT 句
　CREATE TABLESPACE，16-79
SELECT ANY CUBE DIMENSION システム権限，18-41
SELECT ANY CUBE システム権限，18-40
SELECT ANY DICTIONARY システム権限，18-45
SELECT ANY MINING MODEL システム権限，18-40
SELECT ANY SEQUENCE システム権限，18-42
SELECT ANY TABLE システム権限，18-43
SELECT_CATALOG_ROLE ロール，18-46
SELECT オブジェクト権限
　OLAP キューブ，18-48
　OLAP キューブ・ディメンション，18-48
　順序，18-49
　ビュー，18-50
　ビューに対する付与，18-35
　表，18-49
　マイニング・モデル，18-47
　マテリアライズド・ビュー，18-47
SELECT 構文のリスト，9-2
　順序付け，9-10
SELECT 文，9-2，19-4
SESSION_ROLES ビュー，19-51
SESSIONS_PER_USER パラメータ
　ALTER PROFILE，11-29
SESSIONTIMEZONE ファンクション，5-163
SET CONSTRAINT(S) 文，19-49
SET DANGLING TO NULL 句
　ANALYZE，13-17
SET DATABASE 句
　CREATE CONTROLFILE，14-12
SET KEY 句
　ALTER SYSTEM，11-62
SET ROLE 文，19-51
SET STANDBY DATABASE 句
　ALTER DATABASE，10-33
SET STATEMENT_ID 句
　EXPLAIN PLAN，18-20
SET TIME_ZONE 句
　ALTER DATABASE，10-17，10-39

ALTER SESSION，11-47
CREATE DATABASE，14-19
SET TRANSACTION 文，19-53
SET UNUSED 句
ALTER TABLE，12-46
SET WALLET 句
ALTER SYSTEM，11-61
SET 句
ALTER SESSION，11-44
ALTER SYSTEM，11-63
SET 条件，7-11
SET ファンクション，5-163
SGA「システム・グローバル領域（SGA）」を参照
SHARE ROW EXCLUSIVE ロック・モード，18-71
SHARE UPDATE ロック・モード，18-71
SHARED 句
CREATE DATABASE LINK，14-29
SHRINK SPACE 句
ALTER INDEX，10-70
ALTER MATERIALIZED VIEW，11-10
ALTER MATERIALIZED VIEW LOG，11-19
ALTER TABLE，12-35
SHUTDOWN 句
ALTER SYSTEM，11-62
SI_AverageColor データ型，2-35
SI_ColorHistogram データ型，2-35
SI_Color データ型，2-35
SI_FeatureList データ型，2-35
SI_PositionalColorHistogram データ型，2-35
SI_StillImage データ型，2-35
SI_Texture データ型，2-35
SIGN ファンクション，5-164
SINGLE TABLE 句
CREATE CLUSTER，14-5
SINH ファンクション，5-165
SIN ファンクション，5-165
SIZE 句
ALTER CLUSTER，10-6
CREATE CLUSTER，14-5
ファイル指定，8-31
SNMPAGENT ロール，18-46
SOME 演算子，7-4
SOUNDEX ファンクション，5-166
SPLIT PARTITION 句
ALTER INDEX，10-79
ALTER TABLE，12-62
SPTH 日時書式要素の接尾辞，2-63
SP 日時書式要素の接尾辞，2-63
SQL Developer，1-4
SQL:99 規格，1-2
SQL/DS データ型，2-27
制限事項，2-28
SQL*Loader の挿入のログの記録，10-71
SQL コメント
XMLCOMMENT，5-232
SQL ファンクション
ABS，5-15
ACOS，5-16
ADD_MONTHS，5-16
APPENDCHILDXML，5-17
ASCII，5-18
ASCIISTR，5-17
ASIN，5-19
ATAN，5-19
ATAN2，5-20
AVG，5-20
BFILENAME，5-21
BIN_TO_NUM，5-22
BITAND，5-23
CARDINALITY，5-25
CAST，5-25
CEIL，5-28
CHARTOROWID，5-28
CHR，5-29
CLUSTER_ID，5-30
CLUSTER_PROBABILITY，5-31
CLUSTER_SET，5-32
COALESCE，5-35
COLLECT，5-36
COMPOSE，5-36
CONCAT，5-37
CONVERT，5-38
CORR，5-39
CORR_K，5-42
CORR_S，5-42
COS，5-42
COSH，5-43
COUNT，5-43
COVAR_POP，5-45
COVAR_SAMP，5-46
CUBE_TABLE，5-47
CUME_DIST，5-48
CURRRENT_DATE，5-50
CURRRENT_TIMESTAMP，5-50
CV，5-51
DATAOBJ_TO_PARTITION，5-52
DBTIMEZONE，5-53
DECOMPOSE，5-54
DELETEXML，5-55
DENSE_RANK，5-56
DEREF，5-58
DUMP，5-59
EMPTY_BLOB，5-60
EMPTY_CLOB，5-60
EXISTSNODE，5-61
EXP，5-62
EXTRACTXML，5-65
EXTRACT（XML），5-65
EXTRACT（日時），5-62
FEATURE_ID，5-66
FEATURE_SET，5-67
FEATURE_VALUE，5-69
FIRST，5-71
FIRST_VALUE，5-72
FLOOR，5-74
FROM_TZ，5-74
GREATEST，5-75
GROUP_ID，5-75
GROUPING，5-76
GROUPING_ID，5-77
HEXTORAW，5-78
INITCAP，5-78
INSERTCHILDXML，5-79
INSERTCHILDXMLAFTER，5-80
INSERTCHILDXMLBEFORE，5-81
INSERTXMLAFTER，5-82

INSERTXMLBEFORE，5-82
INSTR，5-83
INSTR2，5-83
INSTR4，5-83
INSTRB，5-83
INSTRC，5-83
ITERATION_NUMBER，5-85
LAG，5-86
LAST，5-87
LAST_DAY，5-87
LAST_VALUE，5-88
LEAD，5-90
LEAST，5-91
LENGTH，5-91
LENGTH2，5-91
LENGTH4，5-91
LENGTHB，5-91
LENGTHC，5-91
LN，5-92
LNNVL，5-93
LOB 列への適用，5-2
LOCALTIMESTAMP，5-94
LOG，5-95
LOWER，5-95
LPAD，5-96
LTRIM，5-96
MAKE_REF，5-97
MAX，5-98
MEDIAN，5-99
MIN，5-101
MOD，5-102
MONTHS_BETWEEN，5-103
NANVL，5-103
NCHR，5-104
NEW_TIME，5-105
NEXT_DAY，5-106
NLS_CHARSET_DECL_LEN，5-106
NLS_CHARSET_ID，5-107
NLS_CHARSET_NAME，5-107
NLS_INITCAP，5-108
NLS_LOWER，5-109
NLS_UPPER，5-111
NLSSORT，5-109
NLS 文字，5-4
NLV2，5-115
NTILE，5-111
NULLIF，5-112
NULL 関連，5-8
NUMTODSINTERVAL，5-113
NUMTOYMINTERVAL，5-114
NVL，5-115
ORA_HASH，5-116
PERCENT_RANK，5-118
PERCENTILE_CONT，5-119
PERCENTILE_DISC，5-121
POWER，5-122
POWERMULTISET，5-123
POWERMULTISET_BY_CARDINALITY，5-124
PREDICTION，5-125
PREDICTION_BOUNDS，5-127
PREDICTION_COST，5-128
PREDICTION_DETAILS，5-130
PREDICTION_PROBABILITY，5-131
PREDICTION_SET，5-133
PRESENTNNV，5-135
PRESENTV，5-136
PREVIOUS，5-137
RANK，5-138
RATIO_TO_REPORT，5-140
RAWTOHEX，5-141
RAWTONHEX，5-141
REF，5-142
REFTOHEX，5-142
REGEXP_COUNT，5-143
REGEXP_INSTR，5-144
REGEXP_REPLACE，5-147
REGEXP_SUBSTR，5-149
REGR_AVGX，5-150
REGR_AVGY，5-150
REGR_COUNT，5-150
REGR_INTERCEPT，5-150
REGR_R2，5-150
REGR_SLOPE，5-150
REGR_SXX，5-150
REGR_SXY，5-150
REGR_SYY，5-150
REMAINDER，5-155
REPLACE，5-155
ROUND（数値），5-156
ROUND（日付），5-157
ROW_NUMBER，5-158
ROWIDTOCHAR，5-159
ROWIDTONCHAR，5-160
RPAD，5-160
RTRIM，5-161
SESSIONTIMEZONE，5-163
SET，5-163
SIGN，5-164
SIN，5-165
SINH，5-165
SOUNDEX，5-166
SQRT，5-167
STATS_BINOMIAL_TEST，5-167
STATS_CROSSTAB，5-168
STATS_F_TEST，5-169
STATS_KS_TEST，5-170
STATS_MODE，5-171
STATS_MW_TEST，5-172
STATS_ONE_WAY_ANOVA，5-173
STATS_T_TEST_INDEP，5-174，5-176
STATS_T_TEST_INDEPU，5-174，5-176
STATS_T_TEST_ONE，5-174，5-175
STATS_T_TEST_PAIRED，5-174，5-176
STATS_WSR_TEST，5-177
STDDEV，5-178
STDDEV_POP，5-179
STDDEV_SAMP，5-180
SUBSTR，5-181
SUBSTR2，5-181
SUBSTR4，5-181
SUBSTRB，5-181
SUBSTRC，5-181
SUM，5-182
SYS_CONNECT_BY_PATH，5-184
SYS_CONTEXT，5-184
SYS_DBURIGEN，5-190

SYS_EXTRACT_UTC，5-190
SYS_GUID，5-191
SYS_TYPEID，5-191
SYS_XMLAGG，5-192
SYS_XMLGEN，5-193
SYSDATE，5-194
SYSTIMESTAMP，5-194
TAN，5-195
TANH，5-195
TIMESTAMP_TO_SCN，5-196
TO_BINARY_DOUBLE，5-196
TO_BINARY_FLOAT，5-198
TO_BLOB，5-198
TO_CHAR（数値），5-201
TO_CHAR（日時），5-200
TO_CHAR（文字），5-199
TO_CLOB，5-203
TO_DATE，5-203
TO_DSINTERVAL，5-205
TO_LOB，5-206
TO_MULTI_BYTE，5-206
TO_NCHAR（数値），5-208
TO_NCHAR（日時），5-207
TO_NCHAR（文字），5-207
TO_NCLOB，5-209
TO_NUMBER，5-209
TO_SINGLE_BYTE，5-210
TO_TIMESTAMP，5-210
TO_YMINTERVAL，5-212
TRANSLATE，5-213
TRANSLATE ... USING，5-214
TREAT，5-215
TRIM，5-216
TRUNC（数値），5-217
TRUNC（日付），5-218
TZ_OFFSET，5-218
t 検定，5-174
UID，5-219
UNISTR，5-219
UPDATEXML，5-220
UPPER，5-221
USER，5-221
USERENV，5-222
VALUE，5-223
VAR_POP，5-223
VAR_SAMP，5-224
VARIANCE，5-225
VSIZE，5-226
WIDTH_BUCKET，5-227
XMLAGG，5-228
XMLCAST，5-229
XMLCDATA，5-230
XMLCONCAT，5-232
XMLDiff，5-233
XMLExists，5-237
XMLPARSE，5-238
XMLPI，5-240
XMLQUERY，5-241
XMLROOT，5-242
XMLSERIALIZE，5-244
XMLTABLE，5-245
XML ファンクション，5-7
一般的な比較ファンクション，5-5
エンコーディングおよびデコーディング，5-8
オブジェクト参照，5-15
階層，5-6
環境および識別子，5-8
集計，5-8
収集，5-6
数値，5-3
線形回帰，5-150
単一行，5-3
データ・マイニング，5-7
日付，5-5
分析，5-10
変換，5-5
文字
  数値，5-4
  文字値，5-4
モデル，5-15
ラージ・オブジェクト，5-6
SQL 文
ALTER FLASHBACK ARCHIVE，10-60
CREATE FLASHBACK ARCHIVE，14-45
DDL，10-2
DML，10-3
DROP FLASHBACK ARCHIVE，17-41
監査
  正常終了，13-30
  セッション，13-29
  停止，18-76
  プロキシ，13-28
  ユーザー，13-28
構成，10-4
再開可能な領域割当て，11-44
システム制御，10-3
実行計画の判断，18-19
種類，10-2
セッション制御，10-3
セッション中の実行の監査，13-25
停止およびコンパイル，11-44
トランザクション制御，10-3
取消し，18-92
ロールバック，18-92
SQRT ファンクション，5-167
STAR_TRANSFORMATION_ENABLED 初期化パラメータ，2-95
STAR_TRANSFORMATION ヒント，2-94
START LOGICAL STANDBY APPLY 句
ALTER DATABASE，10-36
START WITH 句
ALTER MATERIALIZED VIEW ...REFRESH，11-12
SELECT および副問合せ，19-9
問合せおよび副問合せ，19-23
START WITH パラメータ
CREATE SEQUENCE，15-66
startup_clauses
ALTER DATABASE，10-10
STATS_BINOMIAL_TEST ファンクション，5-167
STATS_CROSSTAB ファンクション，5-168
STATS_F_TEST ファンクション，5-169
STATS_KS_TEST ファンクション，5-170
STATS_MODE ファンクション，5-171
STATS_MW_TEST ファンクション，5-172
STATS_ONE_WAY_ANOVA ファンクション，5-173
STATS_T_TEST_INDEPU ファンクション，5-174，5-176

STATS_T_TEST_INDEP ファンクション，5-174，5-176
STATS_T_TEST_ONE ファンクション，5-174，5-175
STATS_T_TEST_PAIRED ファンクション，5-174，5-176
STATS_WSR_TEST ファンクション，5-177
STDDEV_POP ファンクション，5-179
STDDEV_SAMP ファンクション，5-180
STDDEV ファンクション，5-178
STOP LOGICAL STANDBY 句
  ALTER DATABASE，10-36
STORAGE 句
  ALTER CLUSTER，10-6
  ALTER INDEX，10-71
  ALTER MATERIALIZED VIEW LOG，11-17
  CREATE MATERIALIZED VIEW LOG，15-27
  CREATE MATERIALIZED VIEW LOG「CREATE TABLE」を参照
  CREATE TABLE，8-41，16-10
STORE IN 句
  ALTER TABLE，12-38，16-47
Structured Query Language（SQL）
  Oracle のツール製品のサポート，1-4
  キーワード，A-3
  規格，1-2，B-1
  構文，10-4，A-2
  説明，1-2
  パラメータ，A-3
  ファンクション，5-2
  文
    コストの判断，18-19
SUBMULTISET 条件，7-13
SUBPARTITION BY HASH 句
  CREATE TABLE，16-21，16-51
SUBPARTITION BY LIST 句
  CREATE TABLE，16-51
SUBPARTITION 句
  ANALYZE，13-16
  DELETE，17-26
  INSERT，18-58
  LOCK TABLE，18-70
  UPDATE，19-65
SUBSTR2 ファンクション，5-181
SUBSTR4 ファンクション，5-181
SUBSTRB ファンクション，5-181
SUBSTRC ファンクション，5-181
SUBSTR ファンクション，5-181
SUM ファンクション，5-182
SUSPEND 句
  ALTER SYSTEM，11-59
SWITCH LOGFILE 句
  ALTER SYSTEM，11-59
SYS_CONNECT_BY_PATH ファンクション，5-184
SYS_CONTEXT ファンクション，5-184
SYS_DBURIGEN ファンクション，5-190
SYS_EXTRACT_UTC ファンクション，5-190
SYS_GUID ファンクション，5-191
SYS_NC_ROWINFO$ 列，16-59，17-19
SYS_TYPEID ファンクション，5-191
SYS_XMLAGG ファンクション，5-192
SYS_XMLGEN ファンクション，5-193
SYSAUX 句
  CREATE DATABASE，14-23
SYSAUX 表領域
  作成，14-23，16-75
SYSDATE ファンクション，5-194
SYSDBA システム権限，18-46
SYSOPER システム権限，18-46
SYSTEM 表領域
  ローカル管理，14-22
SYSTEM ユーザー
  パスワードの割当て，14-19
SYSTIMESTAMP ファンクション，5-194
SYS ユーザー
  パスワードの割当て，14-19

T

TABLESPACE 句
  ALTER INDEX ...REBUILD，10-73
  CREATE CLUSTER，14-5
  CREATE INDEX，14-60
  CREATE MATERIALIZED VIEW，15-13
  CREATE MATERIALIZED VIEW LOG，15-28
  CREATE TABLE，16-30
TABLE 句
  ANALYZE，13-15
  INSERT，18-59
  SELECT，19-18
  TRUNCATE，19-59
  UPDATE，19-64，19-66
TANH ファンクション，5-195
TAN ファンクション，5-195
TEMPFILE 句
  ALTER DATABASE，10-12，10-27
TEMPORARY TABLESPACE 句
  ALTER USER，13-8
  ALTER USER「CREATE USER」を参照，13-7
  CREATE USER，17-10
TEMPORARY 句
  ALTER TABLESPACE，12-91
  CREATE TABLESPACE，16-80
TEST 句
  ALTER DATABASERECOVER，10-22
THSP 日時書式要素の接尾辞，2-63
TH 日時書式要素の接尾辞，2-63
TIME_ZONE セッション・パラメータ，11-47
TIMESTAMP WITH LOCAL TIME ZONE データ型，2-19
TIMESTAMP WITH TIME ZONE データ型，2-18
TIMESTAMP_TO_SCN ファンクション，5-196
TIMESTAMP データ型，2-18
  DB2，2-28
  SQL/DS，2-28
TIME データ型
  DB2，2-28
  SQL/DS，2-28
TO SAVEPOINT 句
  ROLLBACK，18-93
TO_BINARY_DOUBLE ファンクション，5-196
TO_BINARY_FLOAT ファンクション，5-198
TO_BLOB ファンクション，5-198
TO_CHAR
  数値変換ファンクション，5-201
  日時変換ファンクション，5-200
TO_CHAR ファンクション，2-54，2-57，2-63
TO_CHAR（文字）ファンクション，5-199
TO_CLOB ファンクション，5-203

TO_DATE ファンクション，2-57，2-62，2-64，5-203
TO_DSINTERVAL ファンクション，5-205
TO_LOB ファンクション，5-206
TO_MULTI_BYTE ファンクション，5-206
TO_NCHAR（数値）ファンクション，5-208
TO_NCHAR（日時）ファンクション，5-207
TO_NCHAR（文字）ファンクション，5-207
TO_NCLOB ファンクション，5-209
TO_NUMBER ファンクション，2-54，5-209
TO_SINGLE_BYTE ファンクション，5-210
TO_TIMESTAMP_TZ ファンクション
  SQL ファンクション
    TO_TIMESTAMP_TZ，5-211
TO_TIMESTAMP ファンクション，5-210
TO_YMINTERVAL ファンクション，5-212
TRANSLATE ... USING ファンクション，5-214
TRANSLATE ファンクション，5-213
TREAT ファンクション，5-215
TRIM ファンクション，5-216
TRUNCATE PARTITION 句
  ALTER TABLE，12-62
TRUNCATE SUBPARTITION 句
  ALTER TABLE，12-62
TRUNCATE_CLUSTER 文，19-56
TRUNCATE_TABLE 文，19-58
TRUNC ファンクション
  書式モデル，5-248
  数値ファンクション，5-217
  日付ファンクション，5-218
TZ_OFFSET ファンクション，5-218

U

UID ファンクション，5-219
UNDER ANY TYPE システム権限，18-44
UNDER ANY VIEW システム権限，18-44
UNDER_PATH 条件，7-19
UNDER オブジェクト権限
  型，18-48
  ビュー，18-50
UNDER 句
  CREATE VIEW，17-17
UNDO
  システム管理，11-36，14-24
  ロールバック，11-36，14-24
UNDO_RETENTION 初期化パラメータ
  ALTER SYSTEM での設定，18-25
UNDO のロールバック，11-36，14-24
UNDO 表領域
  期限が切れていないデータの保持，12-92，16-81
  削除，18-9
  作成，14-24，16-81
  変更，12-85
UNDO 表領域句
  CREATE DATABASE，14-24
  CREATE TABLESPACE，16-81
UNIFORM 句
  CREATE TABLESPACE，16-78
UNION ALL 集合演算子，4-6，19-29
UNION 集合演算子，4-6，19-29
UNIQUE 句
  CREATE INDEX，14-56
  CREATE TABLE，16-29
  SELECT，19-13
UNISTR ファンクション，5-219
UNLIMITED TABLESPACE システム権限，18-43
UNNEST ヒント，2-95
UNQUIESCE 句
  ALTER SYSTEM，11-60
UNRECOVERABLE，10-72，16-32
  「NOLOGGING 句」を参照
UNUSABLE LOCAL INDEXES 句
  ALTER MATERIALIZED VIEW，11-9
  ALTER TABLE，12-68
UNUSABLE 句
  ALTER INDEX，10-75
UPDATE ANY CUBE BUILD PROCESS システム権限，18-41
UPDATE ANY CUBE DIMENSION システム権限，18-41
UPDATE ANY CUBE システム権限，18-40
UPDATE ANY TABLE システム権限，18-43
UPDATE BLOCK REFERENCES 句
  ALTER INDEX，10-76，10-78
UPDATE GLOBAL INDEXES 句
  ALTER TABLE，12-69
UPDATE SET 句
  MERGE，18-72
UPDATEXML ファンクション，5-220
UPDATE オブジェクト権限
  OLAP キューブ，18-48
  OLAP キューブ・ディメンション，18-48
  ビュー，18-50
  表，18-49
UPDATE 文，19-62
UPGRADE 句
  ALTER DATABASE，10-19
  ALTER TABLE，12-35
UPPER ファンクション，5-221
URL
  生成，5-190
UROWID
  外部キー表，2-27
  索引構成表，2-27
  説明，2-27
UROWID データ型，2-27
USE_CONCAT ヒント，2-95
USE_HASH ヒント，2-96
USE_MERGE ヒント，2-96
USE_NL_WITH_INDEX ヒント，2-97
USE_NL ヒント，2-96
USE_PRIVATE_OUTLINES セッション・パラメータ，11-48
USE_STORED_OUTLINES セッション・パラメータ，11-49，11-64
USER SYSTEM 句
  CREATE DATABASE，14-19
USER SYS 句
  CREATE DATABASE，14-19
USER_COL_COMMENTS データ・ディクショナリ・ビュー，13-42
USER_INDEXTYPE_COMMENTS データ・ディクショナリ・ビュー，13-42
USER_MVIEW_COMMENTS データ・ディクショナリ・ビュー，13-42

USER_OPERATOR_COMMENTS データ・ディクショナリ・ビュー，13-42
USER_TAB_COMMENTS データ・ディクショナリ・ビュー，13-42
USERENV ファンクション，5-222
USER ファンクション，5-221
USING BFILE 句
  CREATE JAVA，14-79
USING BLOB 句
  CREATE JAVA，14-79
USING CLOB 句
  CREATE JAVA，14-79
USING INDEX 句
  ALTER MATERIALIZED VIEW，11-10
  ALTER TABLE，12-31
  CREATE MATERIALIZED VIEW，15-15
  CREATE TABLE，16-55
  制約，8-15
USING NO INDEX 句
  CREATE MATERIALIZED VIEW，15-15
USING ROLLBACK SEGMENT 句
  ALTER MATERIALIZED VIEW ...REFRESH，11-12
  CREATE MATERIALIZED VIEW，15-18
USING 句
  ALTER INDEXTYPE，10-83
  ASSOCIATE STATISTICS，13-22，13-23
  CREATE DATABASE LINK，14-31
  CREATE INDEXTYPE，14-74
UTC
  日時値から抽出，5-190
UTC オフセット
  タイムゾーン地域に置換，2-50
UTLCHN.SQL スクリプト，13-18
UTLEXPT1.SQL スクリプト，12-67
UTLXPLAN.SQL スクリプト，18-19

V

VALIDATE REF UPDATE 句
  ANALYZE，13-17
VALIDATE STRUCTURE 句
  ANALYZE，13-17
VALIDATE 句
  DROP TYPE，18-13
VALUES LESS THAN 句
  CREATE TABLE，16-46
VALUES 句
  CREATE INDEX，14-63
  INSERT，18-60
VALUE ファンクション，5-223
VAR_POP ファンクション，5-223
VAR_SAMP ファンクション，5-224
VARCHAR2 データ型，2-10
  NUMBER への変換，2-54
VARCHAR データ型，2-11
VARGRAPHIC データ型
  DB2，2-28
  SQL/DS，2-28
VARIANCE ファンクション，5-225
VARRAY，2-30
  アウトラインでの格納，2-30
  記憶特性，12-42，12-50，16-41
  作成，17-3
  仕様部の削除，18-12
  ネストした表との比較，2-39
  比較規則，2-39
  本体の削除，18-14
  戻り値の変更，12-49
  列プロパティの変更，12-16
VARRAY 句
  ALTER TABLE，12-13
VARRAY 列プロパティ
  ALTER TABLE，12-13，12-42
  CREATE MATERIALIZED VIEW，15-10
  CREATE TABLE，16-12，16-41
VARYING 配列「VARRAY」を参照
VSIZE ファンクション，5-226

W

WHENEVER NOT SUCCESSFUL 句
  NOAUDIT，18-78
WHENEVER SUCCESSFUL 句
  AUDIT sql_statements，13-30
  NOAUDIT，18-78
WHERE 句
  DELETE，17-27
  SELECT，9-4
  UPDATE，19-68
  問合せおよび副問合せ，19-22
WIDTH_BUCKET ファンクション，5-227
With Admin Option 句
  GRANT，18-34
WITH CHECK OPTION 句
  CREATE VIEW，17-15，17-19
  DELETE，17-26
  INSERT，18-59
  SELECT，19-8
  UPDATE，19-65
With Grant Option 句
  GRANT，18-36
With Hierarchy Option
  GRANT，18-36
WITH INDEX CONTEXT 句
  CREATE OPERATOR，15-34
WITH OBJECT IDENTIFIER 句
  CREATE VIEW，17-17
WITH OBJECT ID 句
  CREATE MATERIALIZED VIEW LOG，15-29
WITH OBJECT OID「WITH OBJECT IDENTIFIER」を参照
WITH PRIMARY KEY 句
  ALTER MATERIALIZED VIEW，11-12
  CREATE MATERIALIZED VIEW LOG，15-29
  CREATE MATERIALIZED VIEW ...REFRESH，15-18
WITH *query_name* 句
  SELECT，19-13
WITH READ ONLY 句
  CREATE VIEW，17-15，17-19
  DELETE，17-26
  INSERT，18-59
  SELECT，19-8
  UPDATE，19-65
WITH ROWID 句
  CREATE MATERIALIZED VIEW LOG，15-29
  CREATE MATERIALIZED VIEW ...REFRESH，15-18

REF 制約，8-12
WITH SEQUENCE 句
CREATE MATERIALIZED VIEW LOG，15-29
WRITE オブジェクト権限
ディレクトリ，18-47
WRITE 句
COMMIT，13-45

X

XML
例，E-8
XMLAGG ファンクション，5-228
XMLCAST ファンクション，5-229
XMLCDATA ファンクション，5-230
XMLCOMMENT ファンクション，5-232
XMLCONCAT ファンクション，5-232
XMLDATA 疑似列，3-10
XMLDiff ファンクション，5-233
XMLExists ファンクション，5-237
XMLGenFormatType オブジェクト，2-66
XMLIndex
作成，14-66
変更，10-74
XMLPARSE ファンクション，5-238
XMLPI ファンクション，5-240
XMLQUERY ファンクション，5-241
XMLROOT ファンクション，5-242
XMLSchema
単一および複数，16-43
表からの削除，12-43
表への追加，12-43，16-43
XMLSERIALIZE ファンクション，5-244
XMLTABLE ファンクション，5-245
XMLType 記憶域句
CREATE TABLE，16-43
XMLType ビュー，17-19
問合せ，17-19
XMLType 表
索引の作成，14-68
作成，16-59，16-64
XMLType 列
記憶域，16-43
バイナリ XML 書式での保存，16-43
プロパティ，12-43，16-43
XML 出力の出力整形，5-244
XML 条件，7-19
XML 書式モデル，2-66
XML データ
記憶域，16-43
XML データベース・リポジトリ
SQL アクセス，7-19
XML 文書
XML のフラグメントから生成，5-192
データベースからの検索，5-190
XML のフラグメント，5-65
XML ファンクション，5-7

あ

アーカイブ REDO ログ
位置，10-20
アーカイブ・モード
指定，14-21
アウトライン
PUBLIC による使用，15-36
置換え，15-36
オプティマイザによる使用，11-64
オプティマイザによるプライベート・アウトラインの使用，11-48
グループの格納，15-37
現行のセッションによる使用，15-36
異なるカテゴリへの割当て，11-25，11-26，11-27
コピー，15-37
再構築，11-25，11-27
再コンパイル，11-26
作成，15-35
実行計画の生成での使用，11-49，15-35
データベースからの削除，17-54
動的な使用可能化または使用禁止化，15-36
名前の変更，11-25，11-26，11-27
付与
システム権限，18-41
文に対する作成，15-37
空きリスト
LOB に対する指定，16-39
表、パーティション、クラスタまたは索引の指定，8-47
アスタリスク
問合せの全列ワイルド・カード，19-14
圧縮
索引キー，10-67
アプリケーション
検証，14-8
保護，14-8
ユーザーとしての接続の許可，13-9
アプリケーション・サーバー
ユーザーとしての接続の許可，13-9
暗号化，16-26
表領域，8-49，16-77
暗号化キー
生成，11-62
アンチ結合，9-12
アンピボット操作，19-20
構文，19-7
例，19-39
暗黙的なデータ変換，2-39，2-41

い

移行行
クラスタ，13-17
リスト，13-18
以上 / 以下のテスト，7-4
一意索引，14-56
一意制約
索引，16-55
使用可能，16-55
条件，14-70
一意の問合せ，19-13
一意の要素，5-163
一時表
作成，16-6，16-24
セッション固有，16-24
トランザクション固有，16-24

一時表領域
作成，16-80
データベース作成時のエクステント管理の指定，14-18
デフォルト，10-37
ユーザーへの指定，13-8，17-10
一時表領域グループ
ユーザーへの再割当て，13-8
ユーザーへの指定，17-10
一時ファイル
オフライン化，10-27
オンライン化，10-27
サイズ，8-31
サイズ変更，10-27
再利用，8-31
削除，10-27，12-88
指定，8-26
自動拡張，8-31
自動拡張の使用可能化，8-31，10-27
自動拡張の使用禁止化，10-27
縮小，12-88
データベースの定義，14-18
名前の変更，10-25
表領域の定義，16-72，16-73，16-74
位置の透過性，16-2
インサート
MERGE の使用，18-73
従来型，18-53
条件，18-62
シングルテーブル，18-57
ダイレクト・パス，18-53
同時の更新，18-72
マルチテーブル，18-61，18-62
マルチテーブルの例，18-66
インスタンス
索引エクステントの使用可能化，10-70
パラメータの設定，11-63
インスタンス・リカバリ
中断後の継続，10-19
インダウト・トランザクション
強制コミット，13-46
強制実行，13-46
ロールバック，18-92
ロールバックの強制，18-25，18-93
インライン制約
ALTER TABLE，12-41
CREATE TABLE，16-29
インライン・ビュー，9-13

う

埋込み SQL，10-4
プリコンパイラのサポート，10-4

え

エクステント
インスタンスを使用したアクセス制限，10-70
オブジェクト作成時に割り当てられるエクステント数の指定，8-46
オブジェクトの最大エクステント数の指定，8-46
オブジェクトの第 1 エクステントの指定，8-45
オブジェクトの第 2 エクステントの指定，8-45
サイズ増加の割合の指定，8-46
サブパーティションへの割当て，12-34
パーティションへの割当て，12-34
表への割当て，12-34
エラー・ロギング
DELETE 操作，17-28
INSERT 操作，18-63
MERGE 操作，18-74
エンコーディング・ファンクション，5-8
演算
DATE 値，2-20
演算子，4-1
CONNECT_BY_ROOT，4-5
MULTISET EXCEPT，4-7
MULTISET INTERSECT，4-7
MULTISET UNION，4-8
PRIOR，4-5
コメント，13-42
索引タイプからの削除，10-83
索引タイプへの追加，10-83
算術，4-3
実装の指定，15-32
集合，4-6，19-29
単項，4-2
バイナリ，4-2
付与
システム権限，18-41
変更，11-22
ユーザー定義，4-9
削除，17-53
作成，15-32
実装タイプ，15-34
バインド実装方法，15-33
バインドの戻り型，15-33
ファンクションへのバインド，11-23，15-33
ユーザー定義型、コンパイル，11-22
優先順位，4-3
連結，4-4
エンタープライズ・ユーザー
データベース・ユーザーとしての接続の許可，13-9

お

応答時間
最適化，2-77
オブジェクト・アクセス式，6-13
オブジェクト・インスタンス
型，7-23
オブジェクト「オブジェクト型」または「データベース・オブジェクト」を参照，17-3
オブジェクト型，2-29
値，3-7
コンポーネント，2-29
作成，17-3
サブタイプの権限，18-36
参照「REF」を参照
識別子，3-7
システム権限の付与，18-44
仕様部の削除，18-12
属性，2-108
階層のメンバーシップ，12-41
型の階層，16-36
代替性，12-41

統計タイプ，13-21
統計タイプの関連付けの解除，18-12
比較規則，2-39
MAP ファンクション，2-39
ORDER ファンクション，2-39
不完全，17-3
本体
再作成，17-6
作成，17-5
本体の削除，18-14
メソッド，2-108
メンバー・メソッドの定義，17-5
オブジェクト型マテリアライズド・ビュー
作成，15-12
オブジェクト型列
階層のメンバーシップ，12-41
型の階層，16-36
代替性，12-41
プロパティの定義
マテリアライズド・ビュー，15-9
プロパティの変更
表，12-11，12-41
オブジェクト権限
FLASHBACK ARCHIVE，18-47
データベース・オブジェクト
取消し，18-88
取消し，18-85
PUBLIC，18-87
ユーザー，18-84，18-87
ロール，18-84，18-87
付与，15-56
特定の列，18-36
複数，15-62
オブジェクト参照ファンクション，5-15
オブジェクト識別子
REF に含まれる，2-30
オブジェクト・ビュー，17-17
索引の指定，16-59
システム生成，16-58
指定，16-58
主キー，16-58
オブジェクト・ビュー，17-16
作成，17-16
サブビューの作成，17-17
実表
行の追加，18-53
定義，17-13
問合せ，17-16
オブジェクト表
アップグレード，12-35
階層の一部，16-58
行の追加，18-53
最新バージョンへの更新，12-35
作成，16-7，16-58
システム生成の列名，16-58，16-59，17-16，17-19
問合せ，16-58
オペランド，4-1
オペレーティング・システム・ファイル
削除，10-27，18-9，18-10
オンライン REDO ログ
再初期化，10-28
オンライン索引，14-62
再構築，12-70
オンライン・バックアップ
表領域のオンライン・バックアップの終了，12-87

か

カーソル
キャッシュ，18-19
下位 N 番のレポート，5-57，5-138，5-158
階層
ディメンションからの削除，10-44
ディメンションの追加，10-44
ディメンションの定義，14-35
階層問合せ，3-3，9-3，19-23
CONNECT_BY_ISCYCLE 疑似列，3-2
CONNECT_BY_ISLEAF 疑似列，3-2
CONNECT_BY_ROOT 演算子，4-5
PRIOR 演算子，4-5
親である行，3-3，9-4
疑似列，3-2
子である行，3-3，9-4
順序付け，19-29
説明，3-3
リーフ行，3-3
ルートおよびノードの値の検出，5-184
階層問合せ句
SELECT および副問合せ，19-9
階層ファンクション，5-6
外部 LOB，2-24
外部キー制約，8-9
外部キー表
ROWID，2-27
外部結合，9-11
制限事項，9-12
外部表，16-33
ORACLE_DATAPUMP アクセス・ドライバ，16-35
ORACLE_LOADER アクセス・ドライバ，16-35
アクセス・ドライバ，16-35
作成，16-34
制限事項，16-35
変更，12-52
外部ファンクション，14-48，15-45
外部プロシージャ，15-45
リモート・データベースからの実行，15-3
外部ユーザー，15-57，17-8
科学表記法，2-55
拡張 ROWID
基本，2-27
直接利用できない，2-27
拡張索引作成機能
例，E-2
拡張サブパーティション表名
DML 文，2-106
構文，2-106
制限事項，2-107
拡張パーティション表名
DML 文，2-106
構文，2-106
制限事項，2-107
数
SQL 構文内，2-46
構文，2-46
精度，2-47
比較規則，2-36

浮動小数点，2-11，2-13
フルスペルでの表記，2-63
仮想列
作成，16-27
表への追加，12-40
変更，12-40
型「オブジェクト型」または「データ型」を参照
型コンストラクタ式，6-14
各国語キャラクタ・セット
可変長文字列，2-10
固定幅と可変幅，2-10
変更，10-37
マルチバイト・キャラクタ・セット，2-10
マルチバイト・キャラクタ・データ，2-26
空でないサブセット，5-123
環境ファンクション，5-8
監査
SQL 文，13-26
プロキシ，13-26
ユーザー，13-26
SQL 文の停止，18-76
オプション
SQL 文，13-32
システム権限，13-26
スキーマに対する SQL 文，13-26
ディレクトリに対する SQL 文，13-26
方針
値に基づいた方針，13-25
完全な外部結合，19-21
管理スタンバイ・リカバリ
既存の終了，10-23，10-24
制御の復帰，10-23，10-24
遅延の上書き，10-23
バックグラウンド・プロセス，10-23
物理スタンバイから論理スタンバイの作成，10-23
管理リカバリ
データベース，10-11

き

キー圧縮，16-34
索引
使用禁止，10-74
索引構成表，16-34
索引再構築，12-71
使用可能化，10-72
使用禁止，10-74，14-61
定義，10-74
キー保存表，17-18
キーワード，2-99
オブジェクト名，2-99
オプション，A-4
必須，A-3
記憶域パラメータ
デフォルトの変更，12-85
リセット，19-56，19-58
期間
演算，2-20
データ型，2-16
リテラル，2-51
期間式，6-10
期間条件，7-20
疑似列，3-1
COLUMN_VALUE，3-6
CONNECT_BY_ISCYCLE，3-2
CONNECT_BY_ISLEAF，3-2
CURRVAL，3-3
LEVEL，3-3
NEXTVAL，3-3
OBJECT_ID，3-7，16-58，16-59，17-16，17-19
OBJECT_VALUE，3-7
ORA_ROWSCN，3-8
ROWID，3-8
ROWNUM，3-9
使用方法，3-10
XMLDATA，3-10
階層問合せ，3-2
バージョン問合せ，3-6
逆索引，14-61
逆分散関数，5-119，5-121
キャッシュされたカーソル
実行計画，18-19
キャラクタ・セット
一般的，2-37
データベースのキャラクタ・セットの指定，14-20
データベースのデータ・ファイルの指定，14-20
変更，10-37
マルチバイト・キャラクタ，2-99
キャラクタ・ラージ・オブジェクト「CLOB」を参照
キューブ
データの抽出，5-47
行
階層順序の選択，9-3
削除
クラスタ，19-56，19-58
パーティションおよびサブパーティション，17-26
表，19-56，19-58
表およびビュー，17-23
制約違反時の格納，12-67
制約の指定，8-11
挿入
サブパーティション，18-58
パーティション，18-58
リモート・データベース，18-58
パーティション間の移動，16-15，16-55
表への追加，18-53
行コンストラクタ，6-15
兄弟関係
階層問合せでの順序付け，19-29
行値
列へのピボット，19-19
行値コンストラクタ，6-15
共有サーバー
システム・パラメータ，11-65
プロセス
終了，11-65
追加プロセスの作成，11-65
共有プール
フラッシュ，11-58
行レベル依存の追跡，14-6，16-54

く

空白埋め
書式モデルへの指定，2-63

抑制，2-63
クエリー・リライト
定義済，19-4
ディメンション，14-33
位取り
NUMBER データ型，2-11
精度より大きい，2-12
クラスタ
SQL 例，17-33
移行行および連鎖行，13-17，13-18
エクステントの割当て，10-6，10-7
キー値
領域の変更，10-6
領域の割当て，14-5
記憶域属性
変更，10-5
記憶特性の変更，10-6
クラスタ索引，14-57
構造の検証，13-17
索引，14-5
作成，14-2
作成表領域，14-5
システム権限の付与，18-38
データベースからの削除，17-32
統計情報の収集，13-17
取り出されたブロックのキャッシュ，14-6
ハッシュ，14-5
シングルテーブル，14-5
ソート，14-4，16-26
表の削除，17-32
表の割当て，16-36
物理属性
指定，14-4
変更，10-5
並列度
作成時，14-6
変更，10-6，10-7
変更，10-5
未使用エクステントの割当て解除，10-6
未使用領域の解放，10-7
グルーピング・セット，19-24
グループ化
重複の排除，5-75
グループ比較条件，7-6
グローバル索引「グローバル・パーティション索引」を参照
グローバル・パーティション索引，14-62，14-64
グローバル・ユーザー，15-57，17-9
クローン・データベース
マウント，10-18
クロス結合，19-21

け

桁区切り
指定，2-55
結合，9-10
アンチ結合，9-12
外部，9-11
グループ化された表，9-11
制限事項，9-12
データの稠密化，9-11
完全な外部結合，19-21
クロス，19-21
結合条件のない，9-11
自己，9-11
自然，19-22
条件
定義，9-10
セミ結合，9-13
等価結合，9-10
内部，9-11，19-21
パラレル，2-91
左側外部結合，19-21
右側外部結合，19-21
結合ビュー
更新可能化，17-18
変更，17-27，18-58，19-64
例，17-20
権限
オブジェクト型のサブタイプ，18-36
権限受領者からの取消し，18-85
「システム権限」または「オブジェクト権限」を参照
検証
オンラインのデータベース・オブジェクト，13-18
クラスタ，13-17
索引，13-17
データベース・オブジェクト
オフライン，13-18
表，13-17

こ

更新
MERGE の使用，18-72，18-73
同時の挿入，18-72
更新操作
サプリメンタル・ログ・データの収集，10-31
構文図，A-2
複数の部分に分割された図，A-5
ループ，A-4
コール
CPU 時間の制限，15-49
読み込まれたデータ・ブロックの制限，15-49
コール仕様
プロシージャ，15-45
コミット
自動，13-44
非同期，13-45
ごみ箱
オブジェクトの消去，18-80
コメント，2-69
SQL 文，2-69
演算子，13-42
オブジェクトからの削除，13-41
オブジェクトへの追加，13-41
索引タイプ，13-42
指定，2-69
スキーマ・オブジェクト，2-70
データ・ディクショナリからの削除，13-41
トランザクションとの関連付け，13-46
表，13-42
表示，13-42
表の列，13-42
固有の問合せ，19-13

コレクション
VARRAY，2-30
空のテスト，7-12
行の挿入，18-59
検索メソッドの変更，12-10
ネスト解除，19-18
例，19-45
ネストした表，2-30
表としての扱い，18-59，19-18，19-64，19-66
変更，12-49
コレクション型
マルチレベル，16-42
コレクション型値
データ型への変換，5-25
コレクション・ネスト解除，19-18
例，19-45
コンテキスト
システム権限の付与，18-38
ネームスペースの作成，14-8
コンテキスト・ネームスペース
LDAP ディレクトリを使用した初期化，14-9
OCI を使用した初期化，14-9
インスタンスへのアクセス性，14-9
データベースからの削除，17-34
パッケージへの関連付け，14-8
コンポジット・パーティション化
表の作成時，16-21，16-49
レンジ - リスト，12-55，16-51
コンポジット・レンジ・パーティション，16-49

さ

サーバー・ウォレット
キー，11-61
サーバー・パラメータ・ファイル
作成，15-68
メモリー，15-69
サービス名
リモート・データベース，14-31
再開可能な領域割当て，11-44
最高水位標
クラスタ，10-7
索引，10-70
表，12-34，13-15
索引，10-71
B ツリー，14-50
UNUSABLE のマーク付け，10-75
XML Type 表，14-68
アプリケーション固有，14-73
一意，14-56
移動，10-72
オプティマイザによる参照不可，10-76，14-61
オンライン，14-62
オンラインの場合の再構築，10-74
キー圧縮，10-74
キー圧縮の使用可能化，10-72
キーの反復の排除，10-72
逆，10-72，10-73，10-74，14-61
クラスタ，14-57
クラスタへの作成，14-51
グローバル・パーティション，14-62，14-64
更新，12-69
グローバル・パーティション索引の作成，14-53
降順，14-60
クエリー・リライト，14-60
ファンクション索引，14-60
構造の検証，13-17
高速全体スキャン，2-79
ごみ箱からの消去，18-80
コンポジット・パーティション表，14-64
コンポジット・パーティション表への作成，14-56
再構築，10-72
再構築操作のログの記録，10-72
再作成，10-72
索引構成表，14-57
索引に基づく，14-65
索引パーティションの削除，17-44
索引パーティション・ブロックの内容のマージ，10-78
索引ブロックの内容のマージ，10-76
作成，14-50
作成のパラレル化，14-62
サブパーティション
UNUSABLE のマーク付け，10-79
移動，10-72
エクステントの割当て，10-79
記憶特性の変更，10-77
再構築，10-72
再作成，10-72
使用禁止化，10-75
デフォルト属性の変更，10-77
名前の変更，10-78
表領域の指定，10-72，10-73
物理属性の変更，10-71
変更，10-72
未使用領域の解放，10-70，10-79
システム権限の付与，18-39
使用禁止化，10-75
使用禁止として作成，14-67
昇順，14-60
使用の統計情報，10-76
新規エクステントの割当て，10-70
スカラー型オブジェクト属性，14-57
制約の適用，12-51，16-55
属性の変更，10-71，10-72
ダイレクト・パス・インサートのログの記録，10-71
データベースからの削除，17-44
統計情報の収集，13-16
統計タイプの関連付けの解除，17-44
ドメイン，14-50，14-65，14-73
ドメイン例，E-2
名前の変更，10-72，10-76
ネストした表の記憶表，14-57
パーティション，2-106，14-50，14-62
UNUSABLE のマーク付け，10-79，12-68
記憶特性の変更，10-77
再構築，10-72
再作成，10-72
削除，10-77，10-79
実特性の変更，10-78
使用禁止化，10-75
使用禁止のパーティションの再構築，12-68
新規の追加，10-79
デフォルト属性の変更，10-77
名前の変更，10-78
ハッシュ・パーティションの結合，10-79

表領域の指定，10-72，10-73
物理属性の変更，10-71
分割，10-77，10-79
未使用領域の解放，10-70
ユーザー定義，14-62
パーティション化，14-62
パーティション表，14-57
ハッシュ・パーティションの追加，10-77
ハッシュ・パーティション表，14-64
作成，14-55
ビットマップ，14-56
ビットマップ結合，14-67
表，14-51
表の列，14-57
表領域，14-60
表領域の指定，10-72，10-73
ファンクション，14-50
作成，14-58
ブロック内容のマージ，10-72
並列性の変更，10-71
未使用領域の解放，10-70
未ソート，14-61
リスト・パーティション表
作成，14-55
例，14-68
レンジ・パーティション表，14-64
レンジ・パーティション表への作成，14-55
ローカル・ドメイン，14-66
ローカル・パーティション，14-64
索引キー
圧縮，10-67
索引クラスタ
作成，14-5
索引構成表
2 次索引の更新，10-76
PCT_ACCESS_DIRECT 統計情報，13-15
ROWID，2-27
移動，12-70
オーバーフロー・セグメント
記憶域の指定，12-38，16-47
再構築，12-70
索引ブロックの内容のマージ，12-39
作成，16-6
主キー索引
結合，12-38
パーティション化、2 次索引の更新，10-78
ビットマップ索引の作成，16-33
変更，12-37，12-39
マッピング表，12-71
移動，12-58
作成，16-33
索引サブパーティション，14-56
索引タイプ
インスタンス，14-50
演算子の追加，10-82
コメント，13-42
索引タイプに基づく索引，14-65
削除ルーチンの起動，17-44
作成，14-73
システム権限の付与，18-39
実装タイプの変更，10-82
データベースからの削除，17-46
統計情報の関連付け，13-22，13-23
統計タイプの関連性の削除，17-46
変更，10-82
索引パーティション
サブパーティションの作成，14-56
削除，18-73
サブタイプ
安全な削除，18-13
サブパーティション
値の変更，19-65
エクステントの割当て，12-34
行の削除，12-62，17-26
行の挿入，18-58
行の追加，18-53
結合，12-56
作成，16-21
指定，16-49
挿入操作のロギング，12-33
追加，12-55
テンプレートの置換え，12-54
テンプレートの削除，12-54
テンプレートの作成，12-54，16-50
名前の変更，12-62
ハッシュ，16-51
非パーティション表への変換，12-66
表との交換，12-25
物理属性
変更，12-33
別のセグメントへの移動，12-58
未使用領域の解放，12-34
リスト，16-51
リスト・サブパーティションの追加，12-55
ロック，18-69
サブパーティション・テンプレート
置換え，12-54
作成，12-54
サプリメンタル・ロギング
最小，10-31
識別キー（フル），10-31
算術
演算子，4-3
参照整合性制約，8-9
参照によるパーティション，16-48
参照パーティション化，16-48
参照パーティション表，12-52
メンテナンス操作，12-68

し

ジオイメージング，2-33
時間隔パーティション，12-53
時間隔の変更，12-53
パーティション化
期間，16-45
式
CASE，6-5
CURSOR，6-7
DUAL 表の計算，9-14
SQL 構文内，6-2
オブジェクト・アクセス，6-13
型コンストラクタ，6-14
期間，6-10
スカラー副問合せ，6-14
宣言型の変更，5-215

単純，6-3
日時，6-8
比較，5-53
複合，6-4
プレースホルダ，6-13
モデル，6-11
リスト，6-15
列，6-6
識別子ファンクション，5-8
時刻書式モデル
短い，2-57，2-60
自己結合，9-11
システム・グローバル領域
更新，11-57
フラッシュ，11-58，11-59
システム権限
ADMINISTER ANY SQL TUNING SET，18-37
ADMINISTER DATABASE TRIGGER，18-43
ADMINISTER SQL TUNING SET，18-37
ADVISOR，18-37
ALTER ANY CLUSTER，18-38
ALTER ANY CUBE，18-40
ALTER ANY CUBE DIMENSION，18-41
ALTER ANY DIMENSION，18-38
ALTER ANY INDEX，18-39
ALTER ANY INDEXTYPE，18-39
ALTER ANY MATERIALIZED VIEW，18-39
ALTER ANY MINING MODEL，18-40
ALTER ANY OPERATOR，18-41
ALTER ANY OUTLINE，18-41
ALTER ANY PROCEDURE，18-41
ALTER ANY ROLE，18-42
ALTER ANY SEQUENCE，18-42
ALTER ANY SQL PROFILE，18-37
ALTER ANY TABLE，18-43
ALTER ANY TRIGGER，18-43
ALTER ANY TYPE，18-44
ALTER DATABASE，18-38
ALTER PROFILE，18-42
ALTER RESOURCE COST，18-42
ALTER ROLLBACK SEGMENT，18-42
ALTER SESSION，18-42
ALTER SYSTEM，18-38
ALTER TABLESPACE，18-43
ALTER USER，18-44
ANALYZE ANY，18-45
AUDIT ANY，18-45
AUDIT SYSTEM，18-38
BACKUP ANY TABLE，18-43
BECOME USER，18-45
CHANGE NOTIFICATION，18-45
COMMENT ANY MINING MODEL，18-40
COMMENT ANY TABLE，18-45
CREATE ANY CLUSTER，18-38
CREATE ANY CONTEXT，18-38
CREATE ANY CUBE，18-40
CREATE ANY CUBE BUILD PROCESS，18-41
CREATE ANY CUBE DIMENSION，18-41
CREATE ANY DIMENSION，18-38
CREATE ANY DIRECTORY，18-38
CREATE ANY INDEX，18-39
CREATE ANY INDEXTYPE，18-39
CREATE ANY JOB，18-39
CREATE ANY LIBRARY，18-39
CREATE ANY MATERIALIZED VIEW，18-39
CREATE ANY MEASURE FOLDER，18-40
CREATE ANY MINING MODEL，18-40
CREATE ANY OPERATOR，18-41
CREATE ANY OUTLINE，18-41
CREATE ANY PROCEDURE，18-41
CREATE ANY SEQUENCE，18-42
CREATE ANY SQL PROFILE，18-37
CREATE ANY SYNONYM，18-42
CREATE ANY TABLE，18-43
CREATE ANY TRIGGER，18-43
CREATE ANY TYPE，18-44
CREATE ANY VIEW，18-44
CREATE CLUSTER，18-38
CREATE CUBE，18-40
CREATE CUBE BUILD PROCESS，18-41
CREATE CUBE DIMENSION，18-41
CREATE DATABASE LINK，18-38
CREATE DIMENSION，18-38
CREATE EXTERNAL JOB，18-39
CREATE INDEXTYPE，18-39
CREATE JOB，18-39
CREATE LIBRARY，18-39
CREATE MATERIALIZED VIEW，18-39
CREATE MEASURE FOLDER，18-40
CREATE MINING MODEL，18-40
CREATE OPERATOR，18-41
CREATE PROCEDURE，18-41
CREATE PROFILE，18-42
CREATE PUBLIC DATABASE LINK，18-38
CREATE PUBLIC SYNONYM，18-42
CREATE ROLE，18-42
CREATE ROLLBACK SEGMENT，18-42
CREATE SEQUENCE，18-42
CREATE SESSION，18-42
CREATE SYNONYM，18-42
CREATE TABLE，18-43
CREATE TABLESPACE，18-43
CREATE TRIGGER，18-43
CREATE TYPE，18-44
CREATE USER，18-44
CREATE VIEW，18-44
DEBUG ANY PROCEDURE，18-38
DELETE ANY CUBE DIMENSION，18-41
DELETE ANY MEASURE FOLDER，18-40
DELETE ANY TABLE，18-43
DROP ANY CLUSTER，18-38
DROP ANY CONTEXT，18-38
DROP ANY CUBE，18-40
DROP ANY CUBE BUILD PROCESS，18-41
DROP ANY CUBE DIMENSION，18-41
DROP ANY DIMENSION，18-38
DROP ANY DIRECTORY，18-38
DROP ANY INDEX，18-39
DROP ANY INDEXTYPE，18-39
DROP ANY LIBRARY，18-39
DROP ANY MATERIALIZED VIEW，18-39
DROP ANY MEASURE FOLDER，18-40
DROP ANY MINING MODEL，18-40
DROP ANY OPERATOR，18-41
DROP ANY OUTLINE，18-41
DROP ANY PROCEDURE，18-41

DROP ANY ROLE，18-42
DROP ANY SEQUENCE，18-42
DROP ANY SYNONYM，18-42
DROP ANY TABLE，18-43
DROP ANY TRIGGER，18-43
DROP ANY TYPE，18-44
DROP ANY VIEW，18-44
DROP PROFILE，18-42
DROP PUBLIC DATABASE LINK，18-38
DROP PUBLIC SYNONYM，18-42
DROP ROLLBACK SEGMENT，18-42
DROP TABLESPACE，18-43
DROP USER，18-44
EXECUTE ANY CLASS，18-39
EXECUTE ANY INDEXTYPE，18-39
EXECUTE ANY OPERATOR，18-41
EXECUTE ANY PROCEDURE，18-42
EXECUTE ANY PROGRAM，18-39
EXECUTE ANY TYPE，18-44
EXEMPT ACCESS POLICY，18-45
FLASHBACK ANY TABLE，18-40，18-43，18-44
FLASHBACK ARCHIVE ADMINISTER，18-38
FORCE ANY TRANSACTION，18-45
FORCE TRANSACTION，18-45
GLOBAL QUERY REWRITE，18-40
GRANT ANY OBJECT PRIVILEGE，18-45
GRANT ANY PRIVILEGE，18-45
GRANT ANY ROLE，18-42
INSERT ANY CUBE DIMENSION，18-41
INSERT ANY MEASURE FOLDER，18-40
INSERT ANY TABLE，18-43
LOCK ANY TABLE，18-43
MANAGE SCHEDULER，18-39
MANAGE TABLESPACE，18-43
MERGE ANY VIEW，18-44
ON COMMIT REFRESH，18-40
QUERY REWRITE，18-39
RESTRICTED SESSION，18-38，18-42
RESUMABLE，18-45
SELECT ANY CUBE，18-40
SELECT ANY CUBE DIMENSION，18-41
SELECT ANY DICTIONARY，18-45
SELECT ANY MINING MODEL，18-40
SELECT ANY SEQUENCE，18-42
SELECT ANY TABLE，18-43
SYSDBA，18-46
SYSOPER，18-46
UNDER ANY TYPE，18-44
UNDER ANY VIEW，18-44
UNLIMITED TABLESPACE，18-43
UPDATE ANY CUBE，18-40
UPDATE ANY CUBE BUILD PROCESS，18-41
UPDATE ANY CUBE DIMENSION，18-41
UPDATE ANY TABLE，18-43
アドバイザ・フレームワーク，18-37
ジョブ・スケジューラのタスク，18-39
取消し，18-84
  PUBLIC，18-86
  ユーザー，18-86
  ロール，18-86
付与，15-56，18-31
  PUBLIC，18-33
  ユーザー，18-33
  ロール，18-33
  リスト，13-25，18-37
システム制御文，10-3
  PL/SQL サポート，10-3
システム・パーティション化，16-52
システム・パラメータ
  GLOBAL_TOPIC_ENABLED，11-64
システム変更番号
  取得，3-8
自然結合，19-22
事前定義されているロール，18-33
実行計画
  アウトラインの削除，17-54
  判断，18-19
  保存，15-35
実行時のコンパイル
  回避，11-28，13-12
実行者権限
  Java クラスについての変更，10-86
  Java クラスの定義，14-77，14-78
自動 UNDO モード，11-36，14-24
自動ストレージ管理
  クラスタ内のノードの移行，11-59
自動セグメント領域管理，16-79
シノニム
  作成，16-2
  シノニム，16-2
  定義の変更，18-3
  データベースからの削除，18-3
  名前の変更，18-82
  パブリック，16-3
    削除，18-3
  付与
    システム権限，18-42
  プライベート・シノニムの削除，18-3
  リモート，16-4
  ローカル，16-4
集計ファンクション，5-8
集合演算子，4-6，19-29
  INTERSECT，4-6
  MINUS，4-6
  UNION，4-6
  UNION ALL，4-6
収集ファンクション，5-6
主キー
  値の生成，15-64
主キー制約，8-9
  索引，16-55
  使用可能，16-55
順序，3-3，15-64
  値の事前割当て，15-66
  値の順序付け，11-39
  値のリサイクル，11-39
  値へのアクセス，15-64
  キャッシュされた値の数の変更，11-39
  再開，18-2
    値，15-66
    事前に定義した制限，15-65
    異なる番号，11-39
  最小値
    設定，15-66
    設定または変更，11-39
    排除，11-39

最大値
設定，15-66
設定または変更，11-39
排除，11-39
再利用，15-64
作成，15-64
事前に定義した制限での停止，15-65
シノニム，16-2
使用場所，3-4
使用方法，3-4
初期値の設定，15-66
増分，15-64
増分値の設定，15-65
データベースからの削除，18-2
名前の変更，18-82
付与
システム権限，18-42
変更
増分値，11-39
無制限での作成，15-65
連続する値の保証，15-66
順序のリセット，10-18
上位 N 番のレポート，3-9，5-57，5-138，5-158
障害グループ
ディスク・グループ用の作成，10-50，14-42
小計値
導出，19-24
条件
BETWEEN，7-20
EXISTS，7-19，7-21
IN，7-22
IS [NOT] EMPTY，7-12
IS ANY，7-9
IS OF type，7-23
IS PRESENT，7-10
LIKE，7-14
MEMBER，7-12
NULL，7-18
REGEXP_LIKE，7-17
SET，7-11
SQL 構文内，7-1
SUBMULTISET，7-13
UNDER_PATH，7-19
XML，7-19
期間，7-20
グループ比較，7-6
多重集合，7-11
単純比較，7-5
パターン一致，7-14
範囲，7-20
比較，7-4
複合，7-20
浮動小数点，7-7
メンバーシップ，7-12，7-22
モデル，7-9
論理，7-8
小数点文字，2-47
指定，2-55
小のテスト，7-4
初期化パラメータ
セッション設定の変更，11-44
書式
データベースからの戻り値，2-54
データベースに格納された値，2-54
日付および数値「書式モデル」を参照
書式モデル，2-54
XML，2-66
指定，2-65
修飾子，2-63
数値，2-54
数値、要素，2-55
日付，2-57
最大長，2-57
書式要素，2-58
デフォルト書式，2-57
変更，2-57
戻り値の書式の変更，2-65
書式モデルの修飾子，2-63
序数
指定，2-63
フルスペルでの表記，2-63
ジョブ・スケジューラ・オブジェクト権限，18-39
シングルテーブル・インサート，18-57

す

数値書式モデル，2-54
数値の優先順位，2-15
数値ファンクション，5-3
スカラー副問合せ，6-14
スキーマ
作成，15-62
セッションの変更，11-46
定義，2-97
スキーマ・オブジェクト，2-97
位置の保護，16-2
オブジェクト型，2-29
構造の検証，13-17
再コンパイル，10-2
再認可，10-2
削除，18-15
参照，2-102，11-46
所有者の保護，16-2
代替名の提供，16-2
デフォルトのバッファ・プールの定義，8-48
名前解決，2-103
名前の指定
ガイドライン，2-102
規則，2-98
例，2-101
ネームスペース，2-100
パーティション索引，2-106
パーティション表，2-106
部分，2-98
他のスキーマ内，2-104
リスト，2-97
リモート・データベース，2-104
リモート・ビューへのアクセス，14-28
スタンドアロン・プロシージャ
削除，17-57
スタンバイ・データベース
Data Guard，10-36
アクティブ，10-33
使用の制御，10-40
物理スタンバイへの変換，10-36
プライマリ状態へのコミット，10-35

マウント，10-18
メディア・リカバリの設計，10-19
リカバリ，10-20，10-21
ストアド・ファンクション，14-48
スループット
最適化，2-74

せ

正規表現
Perl による演算子，C-4
Oracle のサポート，C-1
演算子、多言語拡張，C-3
多言語の構文，C-2
部分正規表現，5-145，5-149
制御ファイル
強制ロギング・モード，14-14
再作成，14-10
再利用の許可，14-12，14-19
スタンバイの作成，10-32
バックアップ，10-32
整合性制約「制約」を参照
整数
SQL 構文内，2-46
一意の値の生成，15-64
構文，2-46
精度，2-46
精度
NUMBER データ型，2-11
桁数，2-47
制約
CHECK，8-11
REF 表，8-11
REF 列，8-11
一意
索引の属性，8-15
使用可能，16-55
違反の行の格納，12-67
外部キー，8-9
既存の制約の変更，12-51
削除，12-10，12-51，18-10
参照整合性，8-9
主キー，8-9
索引の属性，8-15
使用可能，16-55
使用可能，16-54，16-55
使用禁止，16-54
CASCADE，16-55
制限事項，8-8
チェック
各 DML 文の終わり，8-13
トランザクションの終わり，8-13
トランザクションの始まり，8-14
遅延可能，8-13，19-49
規定，11-46
定義，8-4，16-6
表，16-29
列，16-29
トランザクション内の状態設定，19-49
名前の変更，12-51
パーティション化参照，12-51，16-48
ビュー
削除，13-13，18-17
表の作成後の無効化，12-72
表の作成後の有効化，12-72
表の作成時の無効化，16-23
表の作成時の有効化，16-23
表への追加，12-51
変更，12-10
制約の追加，12-51
セーブポイント
指定，19-2
消去，13-44
ロールバック，18-93
セキュリティ
規定，16-85
セキュリティ句
ALTER SYSTEM，11-61
セグメント
表
縮小化，10-70，11-10，11-19，12-35
領域管理
空きリストの使用，16-79
自動，16-79
手動，16-79
ビットマップの使用，16-79
セグメント属性句
CREATE TABLE，16-15
セッション
CPU 時間の制限，11-32
インスタンス全体で終了，11-58
権限を持つユーザーに制限，11-61
合計経過時間の制限，11-32
終了，11-58
制限，11-60
切断，11-57
タイムゾーンの設定，11-47
特性の変更，11-44
非アクティブ期間の制限，11-29
付与
システム権限，18-42
プライベート SGA 領域の制限，11-33
別のインスタンスへの切替え，11-47
読み取るデータ・ブロックの制限，11-32
リソース・コスト制限の計算，11-32
リソース・コスト制限の変更，11-32
リソース・コストの制限，11-32
リソースの合計の制限，11-29
セッション制御文，10-3
PL/SQL サポート，10-3
セッション・パラメータ
INSTANCE，11-47
設定の変更，11-46
セッション・ロック
解放，11-58
セットのテスト，7-11
セミ結合，9-13
線形回帰ファンクション，5-150
全列ワイルド・カード，19-14

そ

相関ファンクション
ケンドールのタウ b，5-41
スピアマンのロー，5-41
ピアソン，5-39

相関副問合せ，9-13
相関名
　DELETE，17-27
　SELECT，19-18
　索引の実表，14-58
属性
　オブジェクト型での最大数，16-25
　ディスク・グループ，10-57，14-43
　ディメンションからの削除，10-44
　ディメンションの追加，10-44
　ディメンションの定義，14-36

た

大のテスト，7-4
タイムスタンプ
　ローカル・タイムゾーンへの変換，6-8
タイムゾーン
　書式化，2-60
　セッションの確認，5-163
　データの変換，6-8
　データベースのタイムゾーンの指定，14-26
ダイレクト・パス・インサート，2-74，18-53
多重集合条件，7-11
単一行ファンクション，5-3
単項演算子，4-2
単純式，6-3
単純比較条件，7-5
短絡評価
　DECODE ファンクション，5-53

ち

チェックポイント
　強制実行，11-57
遅延可能制約，19-49
中央値の値，5-121
長期スタンバイ・リカバリ・モード，10-22

つ

追跡
　表に対する有効化，12-37，16-56
通貨
　桁区切り，2-55
通貨記号
　ISO，2-55
　連合，2-56
　ローカル，2-56

て

定数値「リテラル」を参照
ディスク
　オフライン化，10-52
　オンライン化，10-52
ディスク・グループ
　均衡の再調整，10-53
　削除，17-39
　作成，14-40
　障害グループの作成，10-50，14-42
　制御ファイルのファイルの指定，14-13
　属性の設定，10-57，14-43
　表領域の作成，16-75
　ファイルの作成，8-28
　ファイルの指定，8-28
　変更，10-46
ディスパッチャ・プロセス
　終了，11-65
　追加プロセスの作成，11-65
ディメンション
　階層
　　削除，10-44
　　追加，10-44
　　定義，14-35
　　変更，10-43
　作成，14-33
　システム権限の付与，18-38
　属性
　　削除，10-44
　　追加，10-44
　　定義，14-36
　　変更，10-43
　データの抽出，5-47
　データベースからの削除，17-37
　無効のディメンションのコンパイル，10-45
　例，14-36
　レベル
　　親子階層，14-34
　　削除，10-44
　　追加，10-44
　　定義，14-34
　レベルの定義，14-33
ディメンション・オブジェクト
　データの抽出，5-47
ディレクトリ・オブジェクト
　オペレーティング・システム・ディレクトリの別名，14-38
　監査，13-29
　再定義，14-39
　作成，14-38
　システム権限の付与，18-38
　データベースからの削除，17-38
ディレクトリ「ディレクトリ・オブジェクト」を参照
データ
　UNDO
　　保持，12-92，16-81
　一時データとしての指定，16-24
　検索，9-2
　サブセットの分析，5-116
　集計
　　GROUP BY のグルーピング・セットの連結，19-24
　　GROUP BY の複合列，19-24
　　グルーピング・セット，19-24
　独立性，16-2
　入力に対する整合性チェック，2-12
　ピボット，19-19
　ピボット解除，19-20
　頻繁に使用されるデータのキャッシュ，16-53
データ型，2-2
　ANSI のサポート，2-5
　BFILE，2-25
　BLOB，2-25
　CHAR，2-9
　CLOB，2-26

DATE，2-17
INTERVAL DAY TO SECOND，2-20
INTERVAL YEAR TO MONTH，2-19
LONG，2-15
LONG RAW，2-23
NCHAR，2-10
NCLOB，2-26
NUMBER，2-11
NVARCHAR2，2-10
Oracle が提供する型，2-31
RAW，2-23
ROWID，2-26
SDO_TOPO_GEOMETRY，2-34
TIMESTAMP，2-18
TIMESTAMP WITH LOCAL TIME ZONE，2-19
TIMESTAMP WITH TIME ZONE，2-18
UROWID，2-27
VARCHAR，2-11
VARCHAR2，2-10
XML 型，2-32
期間，2-16
空間型，2-33
コレクション型値への変換，5-25
統計情報の関連付け，13-22，13-23
長さセマンティクス，2-9，2-10
日時，2-16
任意型，2-31
比較規則，2-36
別のデータ型への変換，5-25
メディア型，2-34
文字，2-9
ユーザー定義，2-29
データ操作言語（DML）
影響される行の取出し，17-28，18-61，19-68
索引付け中の許可，10-72
操作
索引再構築中，12-70
索引作成中，14-62
操作の制限，11-60
パラレル化，16-54
文，10-3
PL/SQL サポート，10-3
データ定義言語（DDL）
排他的アクセスを必要とする文，10-2
文，10-2
PL/SQL サポート，10-2
暗黙的コミット，10-2
再コンパイルの原因，10-2
データ・ディクショナリ
コメントの追加，13-41
データの透過的暗号化，16-26
マスター鍵，11-62
データ・ファイル
一時
縮小，12-88
オフライン化，10-26
オンライン化，10-26
オンラインの情報更新，11-57
オンライン・バックアップ，12-87
オンライン・バックアップの終了，10-26，12-87
サイズ，8-31
サイズの変更，10-26
サイズ変更，10-27
再利用，8-31
削除，12-88，18-9，18-10
システム生成，10-25
指定，8-26
表領域，16-75
自動拡張，8-31
自動拡張の使用可能化，8-31
新規作成，10-25
損失または破損した場合の再作成，10-25
データベースの定義，14-18
データベースのデータ・ファイルの指定，14-25
名前の変更，10-25
破損のリカバリ，10-19
表領域の定義，16-72，16-73，16-74
メディア・リカバリの設計，10-19
リカバリ，10-21
データベース，10-24
FLASHBACK モード，10-39
FORCE LOGGING モード，10-28，14-14，14-22
REDO ログ生成の可能化，10-18
アーカイブ・モード
指定，14-21
アカウント
作成，17-7
アクティビティの再開，11-59
アクティビティの停止，11-59
アップグレード，10-9
以前のバージョンのリストア，10-39，12-91，16-80
一時ファイル
変更，10-25
インスタンス，14-20
オープン，10-18，14-16
オンライン
ログ・ファイルの追加，10-29
過去の時点に戻す，18-22
各国語キャラクタ・セット
指定，14-20
管理リカバリ，10-11
キャラクタ・セット
指定，14-20
キャラクタ・セットの指定，14-20
グローバル名の変更，10-38
再作成の準備，10-32
削除，17-35
作成，14-16
時間ベースのリカバリ，10-21
システム権限の付与，18-38
システム・ユーザー・パスワード，14-19
使用の制御，10-40
スクリプトの作成，10-32
スタンバイ
ログ・ファイルの追加，10-29
スタンバイ状態へのコミット，10-35
すべてのデータの消去，14-16
制御，10-40
制御ファイルの再作成，14-10
制御ファイルの再利用の許可，14-19
静止状態，11-60
接続文字列，2-105
タイムゾーン
判断，5-53
有効な値の設定，10-39，14-26
データ非消失モード，10-33

データ・ファイル
指定，14-25
変更，10-25
特性の変更，14-10
取消しベースのリカバリ，10-21
終了，10-22
名前の指定，10-18
名前の変更，14-10，14-12
ネームスペース，2-100
破損したデータベースの再構成，10-19
バックアップの終了，10-24
フラッシュバック，18-22
ブロック
サイズの指定，16-76
別のデータベースへのサブセットの移動，12-66
変更，10-9
変更禁止，10-40
変更の許可，11-42
変更ベースのリカバリ，10-21
保護モード，10-33
マウント，10-18，14-16
メディア・リカバリの設計，10-19
ユーザーに対する無制限のリソース，15-49
ユーザーに対するリソース制限，15-47
ユーザーへの読取り専用トランザクションの制限，10-19
読み書き両用，10-18
読取り専用，10-18
リカバリ，10-19，10-20
テスト，10-22
破損ブロックの許容，10-22
バックアップ制御ファイルの使用，10-21
リセット
以前のバージョン，10-9
リモート
アクセス，9-14
サービス名，14-31
接続，14-30
挿入，18-58
表ロック，18-70
ユーザーの認証，14-31
ログ・ファイル
指定，14-20
変更，10-27
データベース・オブジェクト
削除，18-15
スキーマ，2-97
非スキーマ，2-98
データベース・トリガー「トリガー」を参照
データベース・リンク，9-14
共有，14-29
クローズ，11-42
現行のユーザー，14-30
構文，2-104
作成，2-104，14-28
参照，2-105
システム権限の付与，18-38
シノニムの作成，16-4
データベースからの削除，17-36
名前の指定，2-104
パブリック，14-29
削除，17-36
ユーザー名およびパスワード，2-105
データ変換，2-39
暗黙的
デメリット，2-39
暗黙的と明示的，2-39
暗黙的に行う場合，2-39，2-41
キャラクタ・データ型，2-41
明示的に行う場合，2-42
データ・マイニング・ファンクション，5-7
デカルト積，9-11
テキスト
CHAR および VARCHAR2 データ型のプロパティ，2-45
構文，2-44
日付および数値書式，2-54
リテラル
SQL 構文内，2-44
テキスト・リテラル
データベースのキャラクタ・セットへの変換，2-45
デコーディング・ファンクション，5-8
デバッグ
システム権限の付与，18-38
デフォルトの索引の抑制，15-15
デフォルトの表領域，14-23
ユーザーへの指定，13-8

と

問合せ，9-2，19-4
SELECT 構文のリスト，9-2
値で戻された行のグループ化，19-24
階層「階層問合せ」を参照
階層問合せでの順序付け，19-29
外部結合，19-18
過去のデータ，19-15
行の複数のバージョン，19-15
行ロック，19-30
結果のソート，9-10
結合，9-10，19-20
構文，9-2
コメント，9-3
上位 N 番，3-9
すべての列の選択，19-14
相関
左相関，19-18
定義済，9-2
トップレベル，9-2
任意の行からの選択，19-17
ヒント，9-3
複合，9-10
複数表の参照，9-10
分散，9-14
戻された行の順序付け，19-29
等価結合，9-10
ディメンションの定義，14-35
等価性のテスト，7-4，7-22
統計情報
関連付けの強制解除，17-31
索引再構築中の収集，10-72
索引の使用，10-76
スカラー・オブジェクト属性
収集，13-14
スキーマ・オブジェクト
削除，13-14

収集，13-14
データ・ディクショナリからの削除，13-19
ユーザー定義
削除，17-44，17-46，17-55，18-6，18-12
統計タイプ
関連付け
データ型，13-22
パッケージ，13-22
ファンクション，13-22
列，13-22
関連付けの解除
型，17-30
索引タイプ，17-30
ドメイン索引，17-30
パッケージ，17-30
ファンクション，17-30
列，17-30
索引タイプとの関連付け，13-22，13-23
データ型との関連付け，13-23
ドメイン索引との関連付け，13-22，13-23
パッケージへの関連付け，13-23
ファンクションとの関連付け，13-23
特殊文字
パスワード，15-51
ドメイン索引，14-50，14-65，14-73
LONG 列，12-44
再構築，10-72
削除ルーチンの起動，18-6
作成の前提条件，14-65
作成のパラレル化，14-66
システム管理，14-75
統計情報の関連付け，13-24
データベースからの削除，17-44
統計情報の関連付け，13-22，13-23
変更，10-74
ユーザー定義の CPU および I/O コストの判断，18-19
例，E-2
ローカル・パーティション，14-66
トランザクション
暗黙的コミット，10-2，10-3
インダウト
強制実行，13-46
コミット，13-44
変換，19-54
完了，11-57
コメント，13-45
自動コミット，13-44
終了，13-44
セーブポイント，19-2
名前の指定，19-54
分散トランザクションの強制実行，11-42
分離レベル，19-53
変更の確定，13-44
読み書き両用，19-53
読取り専用，19-53
ロールバック，11-58，18-92
セーブポイント，18-93
ロックの解除，13-44
割当て
ロールバック・セグメント，19-53
トランザクション制御文，10-3
PL/SQL サポート，10-3
トリガー
INSTEAD OF
削除，17-15
コンパイル，13-2
再作成，16-85
作成，16-85
使用可能，16-85
使用可能化，12-72，13-2，13-3
使用禁止，12-72，13-2
データベース
削除，18-11，18-15
変更，13-3
データベースからの削除，18-11
名前の変更，13-3
付与
システム権限，18-43

な

内部 LOB，2-24
内部 N 番のレポート，5-158
内部結合，9-11，19-21
夏時間，2-22
開始または終了，2-22
境界ケース，2-22

に

日時式，6-8
日時書式要素，2-58
ISO 規格，2-62
RR，2-62
大文字化，2-58
グローバリゼーション・サポート，2-61
接尾辞，2-63
日時データ型，2-16
夏時間，2-22
日時の演算，2-20
境界ケース，11-46
夏時間の演算，2-22
日時ファンクション，5-5
日時フィールド
日時または期間値からの抽出，5-62
日時リテラル，2-48
日時列
DATE 列からの作成，12-43

ね

ネームスペース
オブジェクトのネーミング規則，2-100
スキーマ・オブジェクト，2-100
非スキーマ・オブジェクト，2-100
ネームスペース、データベース，2-100
ネストした表，2-30，5-123，5-163，7-11
VARRAY との比較，2-39
階層の判断，7-13
記憶特性，12-41，16-42
既存の列からの作成，5-36
組合せ，4-6
索引構成表としての定義，12-41
索引列，14-58
作成，17-3

仕様部の削除，18-12
比較規則，2-39
変更，12-49
本体の削除，18-14
マテリアライズド・ビュー，15-9，15-10
マルチレベル，16-42
戻り値の変更，12-49
列プロパティの変更，12-11
ネストした副問合せ，9-13

は

パーティション
LOB 記憶特性，12-42
値の変更，19-65
エクステント
索引への割当て，10-70
エクステントの割当て，12-34
記憶特性，8-41
行の削除，12-62，17-26
行の挿入，18-58
行の追加，18-53
コンポジット
指定，16-49
索引，14-62
削除，12-61
挿入操作のロギング，12-33
追加，12-52
名前の変更，12-62
ハッシュ
結合，12-60
指定，16-47
追加，12-59
非パーティション表への変換，12-66
表との交換，12-25
表領域
定義，16-30
物理属性
変更，12-33
分割，12-62
変更，12-52，12-54
マージ，12-65
未使用領域の解放，12-34
リスト・パーティションの追加，12-60
リテラル値に基づくパーティション，16-48
レンジ
指定，16-45
追加，12-59
ロギング属性，16-31
ロック，18-69
パーティション化
句
ALTER INDEX，10-67
ALTER TABLE，12-52
時間隔パーティションを持つレンジ，16-45
システム，16-52
マテリアライズド・ビュー，11-9，15-7，15-14
マテリアライズド・ビュー・ログ，11-19，15-29
レンジ，16-17
パーティション化された索引構成表
2 次索引の更新，10-78
パーティション化参照制約，12-51，16-48
パーティション交換
制限事項，12-67
パーティション索引，2-106，14-50，14-64
ユーザー定義，14-62
ローカルのパーティション索引の作成，14-55
パーティション表，2-106
バイナリ XML 記憶域，16-43
バイナリ XML 書式，16-43
バイナリ演算子，4-2
バイナリ・ラージ・オブジェクト「BLOB」を参照
バインド
演算子からの削除，11-24
演算子への追加，11-23
パスワード
期限切れ，17-11
使用および再利用の制限，15-50
使用禁止化，15-50
特殊文字，17-8
パラメータ
CREATE PROFILE，15-48
複雑さに対する保証，15-50
猶予期間，15-50
ロック，15-50
パターン一致条件，7-14
バックアップ
増分
ブロック・チェンジ・トラッキング，10-38
バックアップの開始，10-24
パッケージ
再定義，15-40
作成，15-39
シノニム，16-2
データベースからの削除，17-55
統計情報の関連付け，13-22，13-23
統計タイプの関連付けの解除，17-55
パッケージ・プロシージャ
削除，17-57
パッケージ本体
再作成，15-42
作成，15-41
データベースからの削除，17-55
ハッシュ・クラスタ
作成，14-5
単一表の作成，14-5
ハッシュ・ファンクションの指定，14-6
ハッシュ・パーティション
結合，12-56
追加，12-59
ハッシュ・パーティション化句
CREATE TABLE，16-23，16-47
バッファ・キャッシュ
フラッシュ，11-59
パブリック・シノニム，16-3
削除，18-3
パブリック・データベース・リンク
削除，17-36
パラメータ
構文
オプション，A-4
必須，A-3
パラメータ・ファイル
作成，15-43
メモリー，15-44

パラレル実行
DDL 文，11-43
DML 文，11-43
ヒント，2-90
範囲条件，7-20

ひ

ヒープ構成表
作成，16-6
比較条件，7-4
比較セマンティクス
文字列，2-36
引数
演算子，4-1
非スキーマ・オブジェクト
ネームスペース，2-100
リスト，2-98
ヒストグラム
等幅の作成，5-227
左側外部結合，19-21
日付
演算，2-20
比較規則，2-36
日付書式モデル，2-57，2-59
句読点，2-58
テキスト，2-58
長い，2-59
短い，2-59
日付ファンクション，5-5
ビット・ベクトル
数値への変換，5-22
ビットマップ索引，14-56
結合索引の作成，14-51
非同期のコミット，13-45
ピボット操作，19-19
構文，19-7
例，19-39
ビュー
XMLType，17-19
XMLType の作成，17-22
XMLType ビューの問合せ，17-19
オブジェクト・サブビューの作成，17-17
オブジェクト・ビューの作成，17-16
結合
キー保存表，17-18
結合を使用した更新可能化，17-18
更新可能，17-18
再コンパイル，13-12
再作成，17-15
削除
実表の行，17-23
データベース，18-17
作成
コメント，13-41
実表の前，17-15
複数，15-62
実表
行の追加，18-53
シノニム，16-2
制約の削除，13-13
制約の変更，13-13
定義，17-13
データの取出し，19-4
名前の変更，18-82
副問合せ，17-17
制限，17-19
付与
システム権限，18-44
変更
実表の値，19-62
定義，18-17
リモート・ビューへのアクセス，14-28
ビュー制約，8-17，17-16
削除，18-17
変更，13-13
マテリアライズド・ビュー，8-15
ビューに対する MERGE オブジェクト権限，18-49
表
LOB 記憶特性，8-41
SQL 例，16-60
XMLType の作成，16-59
XMLType ビューの問合せ，16-59
新しいセグメントへの移動，12-70
圧縮，12-34，16-31
移行行および連鎖行，13-18
以前のバージョンへのフラッシュバック，18-25
一時
セッション固有，16-24
データの存続期間，16-30
トランザクション固有，16-24
移動，12-30
エクステントの割当て，12-34
オブジェクト
作成，16-7
問合せ，16-58
外部，16-33
作成，16-34
制限事項，16-35
外部構成，16-33
記憶域属性
定義，16-6
記憶域プロパティ，16-30，16-36
記憶特性
定義，8-41
既存値の変更，19-62
キャッシュへのブロックの保存，16-53
行の削除，17-23
行の追加，18-53
クラスタへの割当て，16-36
構成の定義，16-32
構造の検証，13-17
ごみ箱からの消去，18-80
コメント，13-42
コメントの作成，13-41
索引構成，16-32
オーバーフロー・セグメント，16-34
索引ブロックの領域，16-33
索引構成表の移動，12-70
削除
クラスタ，17-32
索引，18-6
所有者，18-15
パーティション，18-6
作成，16-6
複数，15-62

サブパーティション属性，12-53
参照パーティション，12-52，12-68，16-48
シノニム，16-2
制限
ブロックのレコード，12-36
ダイレクト・パスを使用した挿入，18-53
追跡の有効化，16-56
データの取出し，19-4
データベースからの削除，18-5
データベースの外部へのデータの格納，16-34
デフォルト物理属性
変更，12-33
問合せ内での結合，19-20
統計情報の収集，13-15
統計タイプの関連付けの解除，18-6
名前の変更，12-37，18-82
ネスト
記憶特性，16-42
パーティション化，2-106，16-6，16-44
行をパーティション間で移動可能にする，12-36
デフォルト属性，12-53
パーティション属性，12-53
パラレル化
デフォルト値の設定，16-54
パラレル作成，16-54
ヒープ構成，16-32
非クラスタ化，17-32
ビューを介した更新，17-18
表領域
定義，16-6，16-30
副問合せを使用した行の挿入，16-56
物理属性
変更，12-33
付与
システム権限，18-43
並列度
指定，16-6
並列度の変更，12-70
別名，2-108
CREATE INDEX，14-58
DELETE，17-27
未使用領域の解放，12-34
読取り / 書込みモード，12-37
読取り専用モード，12-37
リモート・ビューへのアクセス，14-28
リレーショナル
作成，16-7
ロギング
挿入操作，12-33
表作成，16-31
ロック，18-69
標準 SQL，B-1
Oracle 拡張機能，B-27
表の圧縮，11-9，12-34，15-13，16-31
表パーティション
圧縮，12-34，16-31
表領域，12-85
bigfile，16-74
UNDO，14-24
サイズ変更，12-85
データベースのデフォルト表領域，14-23
デフォルトの一時表領域，14-24
FLASHBACK モード，12-91，16-80
FORCE LOGGING モード，12-89，16-77
smallfile，16-74
UNDO，14-24
データベースのデフォルト表領域，14-23
デフォルトの一時表領域，14-24
UNDO
削除，18-9
作成，14-24，16-81
変更，12-85
暗号化，8-49，16-77
一時
縮小，12-86
ユーザーへの指定，13-8，17-10
一時表領域の作成，16-80
一時ファイル
追加，12-87
エクステント管理，16-78
エクステント・サイズ，16-76
オフライン化，12-90，16-78
オンライン化，12-90，16-78
オンライン・バックアップの終了，12-87
書込み操作の許可，12-90
コンテンツの削除，18-9
作成，16-71
システム権限の付与，18-43
指定
索引再構築，12-71
データ・ファイル，16-75
表，16-30
ユーザー，17-9
自動セグメント領域管理，16-79
損失または破損した場合の再構成，10-19，10-25
データ・ファイル
追加，12-87
名前の変更，12-87
データ・ファイルのバックアップ，12-87
データベースからの削除，18-8
データベースの一時表領域の定義，14-18
デフォルト，10-37
ユーザーへの指定，13-8
デフォルトの一時表領域，10-37
名前の確認，10-37
デフォルトの永続表領域，14-23
名前の変更，12-86
変換
一時表領域から永続表領域，12-91
永続表領域から一時表領域，12-91
未使用エクステントの結合，12-86
未使用のエクステント・サイズ，12-85
メディア・リカバリの設計，10-19
ユーザーの領域の割当て，17-10
読取り専用，12-90
リカバリ，10-19，10-21
ローカル管理，8-45
変更，12-85
ロギング属性，12-89，16-76
表ロック
EXCLUSIVE，18-70，18-71
ROW EXCLUSIVE，18-70，18-71
ROW SHARE，18-70
SHARE，18-70
SHARE ROW EXCLUSIVE，18-71
SHARE UPDATE，18-71

継続期間，18-69
サブパーティション，18-70
使用可能化，12-72
使用禁止，12-72
問合せ，18-69
パーティション，18-70
モード，18-70
リモート・データベース，18-70
ヒント，9-3
ALL_ROWS，2-74
APPEND，2-74
CACHE，2-75
CLUSTER，2-75
CURSOR_SHARING_EXACT，2-75
DRIVING_SITE，2-75
DYNAMIC_SAMPLING，2-76
FACT，2-76
FIRST_ROWS(n)，2-77
FULL，2-77
HASH，2-77
INDEX，2-78
INDEX_ASC，2-78
INDEX_COMBINE，2-79
INDEX_DESC，2-79
INDEX_FFS，2-79
INDEX_JOIN，2-80
INDEX_SS，2-80
INDEX_SS_ASC，2-80
INDEX_SS_DESC，2-81
LEADING，2-81
MERGE，2-82
MODEL_MIN_ANALYSIS，2-82
MONITOR，2-82
NO_EXPAND，2-83
NO_FACT，2-84
NO_INDEX，2-84
NO_INDEX_FFS，2-84
NO_INDEX_SS，2-85
NO_MERGE，2-85
NO_MONITOR，2-85
NO_PARALLEL，2-86
NO_PARALLEL_INDEX，2-86
NO_PUSH_PRED，2-86
NO_PUSH_SUBQ，2-87
NO_PX_JOIN_FILTER，2-87
NO_QUERY_TRANSFORMATION，2-87
NO_REWRITE，2-87
NO_STAR_TRANSFORMATION，2-88
NO_UNNEST，2-88
NO_USE_HASH，2-88
NO_USE_MERGE，2-88
NO_USE_NL，2-89
NO_XML_QUERY_REWRITE，2-89
NO_XMLINDEX_REWRITE，2-89
NOAPPEND，2-83
NOCACHE，2-83
NOPARALLEL，2-86
NOPARALLEL_INDEX，2-86
NOREWRITE，2-87
OPT_PARAM，2-89
ORDERED，2-90
PARALLEL，2-90
PARALLEL_INDEX，2-91
PQ_DISTRIBUTE，2-91
PUSH_PRED，2-92
PUSH_SUBQ，2-93
PX_JOIN_FILTER，2-93
QB_NAME，2-93
REWRITE，2-94
SQL 文，2-70
STAR_TRANSFORMATION，2-94
UNNEST，2-95
USE_CONCAT，2-95
USE_HASH，2-96
USE_MERGE，2-96
USE_NL，2-96
USE_NL_WITH_INDEX，2-97
位置構文，2-71
オプティマイザの引渡し，19-62
構文，2-72
問合せブロックの指定，2-71

ふ

ファイル
REDO ログ・ファイル・グループとしての指定，8-26
一時ファイルとしての指定，8-26
データ・ファイルとしての指定，8-26
ファンクション
「SQL ファンクション」を参照
3GL、コール，15-2
COMMIT または ROLLBACK 文の発行，11-42
DECODE，5-53
一般的な比較，5-5
外部，14-48，15-45
逆分散，5-119，5-121
組込み
式，6-10
コール，13-37
再作成，14-49，14-77
索引の定義，14-58
実行，13-37
実行時のコンパイルの防止，10-63
シノニム，16-2
ストアド，14-48
線形回帰，5-150
宣言の変更，14-49
定義の変更，14-49
データベースからの削除，17-42
統計情報、デフォルト・コストの割当て，13-22
統計情報、デフォルト選択性の定義，13-22
統計情報の関連付け，13-22，13-23
日時，5-5
ネーミング規則，2-101
無効なファンクションの再コンパイル，10-63
戻り値の格納，13-39
ユーザー定義，5-249
式，6-10
ファンクション索引，14-50
作成，14-58
使用可能化，10-72，10-75
使用可能化および使用禁止化，10-72
リフレッシュ，10-36
ファンクション式
組込み，6-10
ユーザー定義，6-10

不完全なオブジェクト型，17-3
作成，17-3
複合外部キー，8-9
複合式，6-4
複合主キー，8-9
複合条件，7-20
副問合せ，9-2，9-13，19-4
インライン・ビュー，9-13
拡張された副問合せのネスト解除，9-14
過去のデータ，19-15
式のかわりに使用，6-14
スカラー，6-14
相関，9-13
定義済，9-2
名前の割当て，19-13
ネスト，9-13
ネスト解除，9-14
表データの挿入，16-56
ファクタリング，19-13
副問合せを含む，9-13
副問合せのネスト解除，9-14
物理スタンバイ・データベース
アクティブ，10-33
スナップショット・スタンバイ・データベースへの変換，10-36
不等価性のテスト，7-22
浮動小数点条件，7-7
浮動小数点数，2-13
NaN の処理，5-103
変換，5-196，5-198
不等性のテスト，7-4
部分正規表現
正規表現，5-145，5-149
プライベート・アウトライン
オプティマイザによる使用，11-48
プライマリ・データベース
物理スタンバイ・データベースへの変換，10-36
フラッシュバック・データ・アーカイブ
権限，18-38
削除，17-41
作成，14-45
表のための指定，12-37，16-56
変更，10-60
フラッシュバック問合せ，19-15
疑似列，3-6
挿入を指定した使用，18-57，19-67
プラン・スタビリティ，15-35
プリコンパイラ，1-4
フルスペルで表した数
指定，2-63
プレースホルダ式，6-13
プロキシ句
ALTER USER，13-6，13-7，13-9
プロシージャ
3GL、コール，15-2
COMMIT または ROLLBACK 文の発行，11-42
依存するローカル・オブジェクトの無効化，17-57
外部，15-45
リモート・データベースからの実行，15-3
コール，13-37
再コンパイル，11-28
再作成，15-46
作成，15-45
実行，13-37
シノニム，16-2
データベースからの削除，17-57
ネーミング規則，2-101
付与
システム権限，18-41
プロファイル
作成，15-47
例，15-51
データベースからの削除，17-58
付与
システム権限，18-42
変更の例，11-30
ユーザーへの割当て，17-10
ユーザーへの割当ての削除，17-58
リソース制限の削除，11-29
リソース制限の追加，11-29
リソース制限の変更，11-29
分散
ヒント，2-91
分散問合せ，9-14
制限事項，9-15
分析ファンクション，5-10
AVG，5-20
CORR，5-39
COUNT，5-43
COVAR_POP，5-45
COVAR_SAMP，5-46
CUME_DIST，5-48
DENSE_RANK，5-56
FIRST，5-71
FIRST_VALUE，5-72
LAG，5-86
LAST，5-87
LAST_VALUE，5-88
LEAD，5-90
MAX，5-98
MIN，5-101
NTILE，5-111
OVER 句，5-10
PERCENT_CONT，5-119
PERCENT_DISC，5-121
PERCENT_RANK，5-118
RANK，5-138
RATIO_TO_REPORT，5-140
ROW_NUMBER，5-158
STDDEV，5-178
STDDEV_POP，5-179
STDDEV_SAMP，5-180
SUM，5-182
VAR_POP，5-223
VAR_SAMP，5-224
VARIANCE，5-225
構文，5-10

へ

別名
問合せおよび副問合せでの指定，19-18
ビュー問合せの式，17-16
列，9-3

変換
ファンクション，5-5
文字列から日付への規則，2-66

ま

マイニング・モデル
監査，13-29
コメント，13-42
マスター・データベース，15-4
マスター表，15-4
マッピング表
索引構成表，12-71，16-33
変更，12-39
マテリアライズド結合ビュー，15-26
マテリアライズド・ビュー，15-15
LOB 記憶域属性，11-9
ROWID，15-18
ROWID の値
マスター表への記録，11-19
ROWID ベースから主キー・ベースへの変更，11-12
圧縮，11-9，15-13
移入，15-15
オブジェクト型マテリアライズド・ビューの作成，15-12
完全リフレッシュ，11-11，15-16
記憶域属性，15-13
変更，11-8
キャッシュへのブロックの保存，11-10
強制リフレッシュ，11-11
クエリー・リライト
資格，8-15
使用可能化および使用禁止化，11-13
クエリー・リライトの使用可能化と使用禁止化，15-19
結合，15-26
更新可能化，15-19
高速リフレッシュ，11-11，15-16
コメントの作成，13-41
再検証，11-13
索引特性
変更，11-9
作成，15-4
システム権限の付与，18-39
シノニム，16-2
主キー，15-18
マスター表の値の記録，11-19
主キー・ベースへの変更，11-20
制約，8-15
データ・ウェアハウス，15-4
データの取出し，19-4
データベースからの削除，17-49
デフォルトの索引の作成の抑制，15-15
パーティション，11-9
圧縮，11-9，15-13
副問合せ，15-20
物理属性，15-13
変更，11-8
並列度，11-9，11-19
作成時，15-15
マスター表の削除，17-50
メンテナンスする索引，15-15
有効範囲の制限，15-12
リフレッシュ，10-36
信頼できる制約の使用，15-19
次の COMMIT での，11-11，15-17
マスター表の DML 後，11-12，15-17
モードの変更，11-10
リフレッシュ日時の変更，11-10
リフレッシュ中の再作成，11-11
例，15-21，15-31
レプリケーション，15-4
ロギングの変更，11-9
マテリアライズド・ビュー・ログ，15-26
ROWID に基づく，11-20
新しい値の除外，11-20
新しい値の保存，11-20
オブジェクト ID に基づく，11-20
記憶域属性
指定，15-28
高速リフレッシュに必要，15-26
作成，15-26
作成のパラレル化，15-29
データベースからの削除，17-51
パーティション，15-29
パーティション属性の変更，11-19
物理属性
指定，15-28
変更，11-18
古い値の保存，15-30
列の追加，11-19
ロギングの変更，11-19
マルチスレッド・サーバー「共有サーバー」を参照
マルチテーブル・インサート，18-61
条件，18-61
無条件，18-61
例，18-66
マルチレベル・コレクション，16-42

み

右側外部結合，19-21
未ソート索引，14-61

め

明示的なデータ変換，2-39，2-42
メディア・リカバリ
起動時の回避，10-26
指定の REDO ログ，10-19
準備，10-27
使用禁止，10-24
使用中の実行，10-22
スタンバイ・データベース，10-19
制限事項，10-20
設計，10-19
長期スタンバイ・リカバリ，10-22
データ・ファイル，10-19
データベース，10-19
表領域，10-19
メンバーシップ条件，7-12，7-22

も

文字長セマンティクス，12-44
文字ファンクション，5-4

文字リテラル「テキスト」を参照
文字列
  ASCII 値への変換，5-17
  Unicode への変換，5-36
  各国語キャラクタ・セット，2-10
  可変長，2-10，2-15
  完全一致，2-64
  固定長，2-9
  長さが 0（ゼロ），2-9
  比較規則，2-36
文字列「テキスト・リテラル」を参照，2-44
文字列リテラル「テキスト・リテラル」を参照
モデル式，6-11
モデル条件，7-9
  IS ANY，7-9
  IS PRESENT，7-10
モデル・ファンクション，5-15
  CV，5-51
  ITERATION_NUMBER，5-85
  PRESENTNNV，5-135
  PRESENTV，5-136
  PREVIOUS，5-137

ゆ

ユーザー
  SQL 例，17-11
  アカウントのロック，17-11
  一時表領域，13-8，17-10
  外部，15-57，17-8
  グローバル，15-57，17-9
  作成，17-7
  データベースからの削除，18-15
  データベース・リンク，14-30
  デフォルトの表領域，13-8，17-9
  認証，13-9
  認証の変更，13-9
  パスワードの期限切れ，17-11
  表およびビューへのアクセスの制限，18-69
  付与
    システム権限，18-44
  リモート・サーバーの認証，14-31
  領域の割当て，17-10
  ローカル，15-57，17-8
  割当て
    デフォルト・ロール，13-8
    プロファイル，17-10
ユーザー定義演算子，4-9
ユーザー定義型，2-29
ユーザー定義の統計情報
  削除，17-44，17-46，17-55，18-6，18-12
ユーザー定義ファンクション，5-249
  名前の優先順位，5-250
  ネーミング規則，5-250
優先順位
  演算子，4-3
  条件，7-3
  数値，2-15
ユニバーサル ROWID「UROWID」を参照
ユリウス日，2-17

よ

予約語，2-99，D-1

ら

ラージ・オブジェクト「LOB データ型」を参照
ラージ・オブジェクト・ファンクション，5-6
ライブラリ
  再作成，15-2
  作成，15-2
  システム権限の付与，18-39
  データベースからの削除，17-48
ライブラリ・ユニット「Java スキーマ・オブジェクト」を参照

り

リカバリ
  中断後のインスタンスの継続，10-19
  データの廃棄，10-18
  データベース，10-10
  分散リカバリの使用可能化，11-58
  メディア、使用中の実行，10-22
  メディア・リカバリの設計，10-19
リストア・ポイント
  使用
    データベースのフラッシュバック，18-23
    表のフラッシュバック，18-27
  保証付き，15-54
  保存済，15-54
リスト・サブパーティション
  追加，12-55
リスト・パーティション化
  値の削除，12-56，12-57
  値の追加，12-56，12-57
  デフォルト・パーティションとデフォルト以外のパーティションのマージ，12-65
  デフォルト・パーティションの作成，16-48
  デフォルト・パーティションの追加，12-60
  デフォルト・パーティションの分割，12-62
  パーティションの作成，16-48
  パーティションの追加，12-60
リスナー
  登録，11-62
リソース・パラメータ
  CREATE PROFILE，15-48
リソース・マネージャ，11-60
リテラル
  SQL 構文内，2-44
  SQL 文およびファンクション，2-44
  期間，2-51
  日時，2-48
リレーショナル表
  作成，16-7，16-24

る

累積分布，5-48
ルーチン
  コール，13-37
  実行，13-37

れ

列
  LOB
    記憶域属性，12-42
  REF
    記述，8-11
  値の制限，8-4
  親子関係，14-33
  仮想
    表への追加，12-40
    変更，12-40
  仮想列の作成，16-27
  記憶域の変更，12-41
  記憶域プロパティ，16-36
  既存の列の変更，12-43
  コメント，13-42
  コメントの作成，13-41
  最大数，16-25
  索引，14-58
  修飾名，9-2
  主キーとしての指定，8-9
  制約の指定，16-29
  置換可能な識別型，5-191
  追加，12-39
  定義，16-6
  デフォルト値の指定，16-26
  統計情報の関連付け，13-22
  名前の変更，12-48
  表からの削除，12-46
  プロパティの変更，12-11，12-41
  別名，9-3
列 REF 制約，8-11
  CREATE TABLE，16-29
列式，6-6
列値
  行へのピボット解除，19-20
レプリケーション
  行レベル依存の追跡，14-6，16-54
レベル
  ディメンションからの削除，10-44
  ディメンションの追加，10-44
  ディメンションの定義，14-34
レベル列
  デフォルト値の指定，16-29
連結演算子，4-4
連鎖行
  クラスタ，13-17
  リスト，13-18
レンジ・パーティション
  値，16-46
  作成，16-45
  追加，12-59
  時間隔パーティションへの変換，12-53

ろ

ローカル管理表領域
  記憶域属性，8-45
  変更，12-85
ローカル・パーティション索引，14-64
ローカル・ユーザー，15-57，17-8
ロール，18-33
  AQ_ADMINISTRATOR_ROLE，18-46
  AQ_USER_ROLE，18-46
  CONNECT，18-46
  DBA，18-46
  DELETE_CATALOG_ROLE，18-46
  EXECUTE_CATALOG_ROLE，18-46
  EXP_FULL_DATABASE，18-46
  IMP_FULL_DATABASE，18-46
  RECOVERY_CATALOG_OWNER，18-46
  RESOURCE，18-46
  DELETE_CATALOG_ROLE，18-46
  SNMPAGENT，18-46
  エンタープライズ・ディレクトリ・サービスを使用した識別，15-57
  外部からの識別，15-57
  作成，15-56
  使用可能
    現行セッション，19-51，19-52
  使用禁止
    現行セッション，19-51，19-52
  データベースからの削除，17-60
  取消し，18-84
    PUBLIC，18-86
    別のロール，17-60，18-86
    ユーザー，17-60，18-86
  認証
    エンタープライズ・ディレクトリ・サービス，15-57
    外部サービス，15-57
    データベース，15-57
    パスワード，15-57
    変更，11-34
  パスワードによる識別，15-57
  パッケージを使用した識別，15-57
  付与，18-31，18-33
    PUBLIC，18-34
    システム権限，18-42
    別のロール，18-34
    ユーザー，18-33
ロールバック・セグメント
  最適なサイズの指定，8-48
  データベースからの削除，17-61
  付与
    システム権限，18-42
ロギング
  REDO ログ・サイズ，8-35
  最小限度の指定，8-35
  サプリメンタル
    削除，10-31
  補助ログ・グループの削除，12-34
  補助ログ・グループの追加，12-34
ログ・グループ
  削除，12-34
  追加，12-34
ログ・データ
  更新操作中の収集，10-31
ログ・ファイル
  削除，10-27
  追加，10-27
  データベースのログ・ファイルの指定，14-20
  登録，10-34

名前の変更，10-25
変更，10-27
ロケール非依存，2-59
ロック
自動
上書き，18-69
ロック「表ロック」を参照
論理条件，7-8
論理スタンバイ・データベース
アクティブ，10-33
異常終了，10-36
停止，10-36