Oracle8*i*

SQL リファレンス Vol.2

リリース 8.1

2000 年 11 月

部品番号 : J02327-01

ORACLE®

Oracle8*i* SQL リファレンス Vol.2, リリース 8.1

部品番号 : J02327-01

原本名：SQL Reference, Volume 2 Release 3 (8.1.7)

原本部品番号：A86013-01

原本著者：Diana Lorentz

原本協力者：Dave Alpern、Vikas Arora、Lance Ashdown、Hermann Baer、Vladimir Barriere、Lucy Burgess、Souripriya Das、Carolyn Gray、John Haydu、Thuvan Hoang、Wei Hu、Namit Jain、Hakan Jakobsson、Bob Jenkins、Mark Johnson、Jonathan Klein、Susan Kotsovolos、Vishu Krishnamurthy、Muralidhar Krishnaprasad、Paul Lane、Geoff Lee、Nina Lewis、Bryn Llewellyn、Phil Locke、David McElhoes、Jack Melnick、Ari Mozes、Subramanian Muralidhar、Ravi Murthy、Sujatha Muthulingam、Bruce Olsen、Alla S Pfauntsch、Tom Portfolio、Kevin Quinn、Ananth Raghavan、Den Raphaely、John Russel、Anant Singh、Rajesh Sivaramasubramaniom、Roger Snowden、Jags Srinivisan、Sankar Subramanian、Murali Thiyagarajah、Michael Tobie、AhnTuan Tran、Randy Urbano、Andy Witkowski、Daniel Wong、Aravind Yalamanchi、Qin Yu、Fred Zemke、Mohamed Ziauddin

目次

10 SQL 文 : CREATE SYNONYM ～ DROP ROLLBACK SEGMENT

11 SQL 文 : DROP SEQUENCE ～ UPDATE

A 構文図

B Oracle と標準 SQL

C Oracle の予約語

索引

はじめに

このマニュアルでは、Oracle のデータベース内の情報を管理するために使用される構造化問合せ言語（Structured Query Language: SQL）について説明します。Oracle SQL は、米国規格協会（American National Standards Institute: ANSI）と国際標準化機構（ISO）の SQL92 規格に基本レベルで準拠していますが、さらに多くの内容を盛り込んでいます。

参照：

- SQL に対する Oracle プロシージャ型言語拡張機能である PL/SQL の詳細は、『Oracle8*i* PL/SQL 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。
- Oracle 埋込み SQL の詳細は、『Oracle8*i* Pro*C/C++ プリコンパイラ・プログラマーズ・ガイド』、『Programmer's Guide to SQL*Module for Ada』および『Oracle8*i* Pro*COBOL プリコンパイラ・プログラマーズ・ガイド』を参照してください。

特徴および機能

『Oracle8*i* SQL リファレンス』には、Oracle8*i* および Oracle8*i* Enterprise Edition 製品の特徴および機能に関する情報が含まれています。Oracle8*i* および Oracle8*i* Enterprise Edition の基本的な機能は同じです。ただし、Enterprise Edition のみで使用できる拡張機能がいくつかあります。また、オプションとして使用できる機能もあります。

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。

Oracle8*i* の新機能

このマニュアルでは、Oracle8*i* の次の新機能を記載しています。

リリース 8.1.7

このリリースでは、次の SQL 関数が追加されています。

- BITAND（4-20 ページ）
- NVL2（4-72 ページ）

リリース 8.1.6

このリリースでは、次の SQL 関数が追加されています。

- CORR（4-25 ページ）
- COVAR_POP（4-29 ページ）
- COVAR_SAMP（4-31 ページ）
- CUME_DIST（4-33 ページ）
- DENSE_RANK（4-34 ページ）
- FIRST_VALUE（4-38 ページ）
- LAG（4-45 ページ）
- LAST_VALUE（4-47 ページ）
- LEAD（4-49 ページ）
- NTILE（4-67 ページ）
- NUMTOYMINTERVAL（4-70 ページ）

- NUMTODSINTERVAL（4-69 ページ）
- PERCENT_RANK（4-73 ページ）
- RATIO_TO_REPORT（4-75 ページ）
- REGR_（線形リグレッション）関数（4-78 ページ）
- STDDEV_POP（4-95 ページ）
- STDDEV_SAMP（4-96 ページ）
- VAR_POP（4-122 ページ）
- VAR_SAMP（4-123 ページ）

さらに、次の機能が拡張されています。

- 集計関数は機能性が拡張されました。4-6 ページの「集計関数」を参照してください。
- LOB 記憶パラメータを指定するとき、読取り専用で LOB キャッシュを指定できます。10-7 ページの「CREATE TABLE」を参照してください。
- 式に関する項に、新しい式が追加されました。5-14 ページの「CASE 式」を参照してください。
- 副問合せのネスト解除が可能になりました。5-27 ページの「ネストした副問合せのネスト解除」を参照してください。

リリース 8.1.5

リリース 8.1.5 では、次の SQL 文が追加されています。

- ALTER DIMENSION（7-33 ページ）
- ALTER JAVA（7-56 ページ）
- ALTER OUTLINE（7-79 ページ）
- ASSOCIATE STATISTICS（8-108 ページ）
- CALL（8-126 ページ）
- CREATE CONTEXT（9-13 ページ）
- CREATE DIMENSION（9-34 ページ）
- CREATE INDEXTYPE（9-75 ページ）
- CREATE JAVA（9-78 ページ）
- CREATE OPERATOR（9-112 ページ）
- CREATE OUTLINE（9-116 ページ）
- CREATE TEMPORARY TABLESPACE（10-63 ページ）

構成

このマニュアルの構成は次のとおりです。

第 1 章「概要」

SQL についての定義と、その歴史およびリレーショナル・データベースへの SQL を使用したアクセスのメリットについて説明します。

第 2 章「Oracle SQL の基本要素」

Oracle データベースと Oracle SQL の基本的な構成ブロックについて説明します。

第 3 章「演算子」

SQL の演算子を、式や条件の中で組み合せて使用する方法について説明します。

第 4 章「関数」

SQL の関数を、式や条件の中で組み合せて使用する方法について説明します。

第 5 章「式、条件および問合せ」

SQL 式、条件および問合せを使用してデータベースから情報を取り出す様々な方法について説明します。

第 6 章「SQL 文について」

SQL 文の様々な型を示し、データベース・タスクに適切な SQL 文を見つけるための表を示します。

第 7 章「SQL 文 : ALTER CLUSTER ～ ALTER SYSTEM」
第 8 章「SQL 文 : ALTER TABLE ～ constraint_clause」
第 9 章「SQL 文 : CREATE CLUSTER ～ CREATE SEQUENCE」
第 10 章「SQL 文 : CREATE SYNONYM ～ DROP ROLLBACK SEGMENT」
第 11 章「SQL 文 : DROP SEQUENCE ～ UPDATE」

すべての SQL 文をアルファベット順に説明します。

付録 A「構文図」

このマニュアルでの構文図の読み方を説明します。

付録 B「Oracle と標準 SQL」

ANSI および ISO 規格に対する Oracle の準拠性について説明します。

付録 C「Oracle の予約語」

Oracle 内部で使用する予約語を示します。

リリース 8.1.7 のマニュアルでの構成変更

すべての SQL 文を含む章（以前の第 7 章）は、5 つの章に分割されました。

リリース 8.1.7 では、次の SQL 文が変更されました。

- SQL 文 `GRANT` *`object_privileges`* および `GRANT` *`system_privileges_and_roles`* は、`GRANT` 文に結合されました。11-31 ページの「GRANT」を参照してください。
- SQL 文 `REVOKE` *`schema_object_privileges`* および `REVOKE` *`system_privileges_and_roles`* は、`REVOKE` 文に結合されました。11-73 ページの「REVOKE」を参照してください。
- SQL 文 `AUDIT` *`sql_statements`* および `AUDIT` *`schema_objects`* は、`AUDIT` 文に結合されました。8-112 ページの「AUDIT」を参照してください。
- SQL 文 `NOAUDIT` *`sql_statements`* および `NOAUDIT` *`schema_objects`* は、`NOAUDIT` 文に結合されました。11-66 ページの「NOAUDIT」を参照してください。

リリース 8.1.5 のマニュアルでの構成変更

リリース 8.0 のマニュアルを使用していたユーザーは、次の項が移動または改名されているため注意してください。

- 「書式モデル」の項は、第 2 章（2-39 ページ）に移動しました。
- 第 3 章は、次の章に分割されました。
  - 第 3 章「演算子」
  - 第 4 章「関数」
  - 第 5 章「式、条件および問合せ」

    5-20 ページの「問合せおよび副問合せ」では、11-88 ページの「SELECT および副問合せ」の構文および比較情報の背景について説明します。
- 新規の第 6 章「SQL 文について」は、特定のタスクに対する適切な SQL 文を見つけやすくするために追加されました。
- 「*archive_log_clause*」は個別の項ではなく、7-123 ページの「ALTER SYSTEM」に含められています。
- 「*deallocate_unused_clause*」は個別の項ではなく、8-2 ページの「ALTER TABLE」、7-3 ページの「ALTER CLUSTER」および 7-38 ページの「ALTER INDEX」に含められています。
- 「*disable_clause*」は個別の項ではなく、10-7 ページの「CREATE TABLE」および 8-2 ページの「ALTER TABLE」に含められています。
- 「*drop_clause*」は個別の項ではなく、「ALTER TABLE 文」の「*drop_constraint_clause*」になりました（ALTER TABLE 文の新しい *drop_column_clause* と区別するため）。8-2 ページの「ALTER TABLE」を参照してください。
- 「*enable_clause*」は個別の項ではなく、10-7 ページの「CREATE TABLE」および 8-2 ページの「ALTER TABLE」に含められています。
- 「*parallel_clause*」は個別の項ではなく、関連のある様々な文に含まれています。
- 「*recover_clause*」は個別の項ではなく、「ALTER DATABASE 文」に含まれています。これは、*recover_clause* のリカバリ機能が向上し、ALTER DATABASE 文を介して常に実行されるためです。7-7 ページの「ALTER DATABASE」を参照してください。

- 「**スナップショット**」および「**スナップショット・ログ**」の項は、移動および改名されています。スナップショット機能が大幅に向上し、これらのオブジェクトを**マテリアライズド・ビュー**と呼びます。9-86 ページの「CREATE MATERIALIZED VIEW」、7-59 ページの「ALTER MATERIALIZED VIEW」、10-140 ページの「DROP MATERIALIZED VIEW」、9-104 ページの「CREATE MATERIALIZED VIEW LOG」、7-73 ページの「ALTER MATERIALIZED VIEW LOG」、および 10-142 ページの「DROP MATERIALIZED VIEW LOG」を参照してください。
- 「**副問合せ**」の項は、「SELECT」の項と結合されました。11-88 ページの「SELECT および副問合せ」を参照してください。

このマニュアルで使用する表記規則

この項では、このマニュアルで使用する表記上の規則について説明します。

- 本文
- 構文図および表記法
- コード例
- サンプル・データ

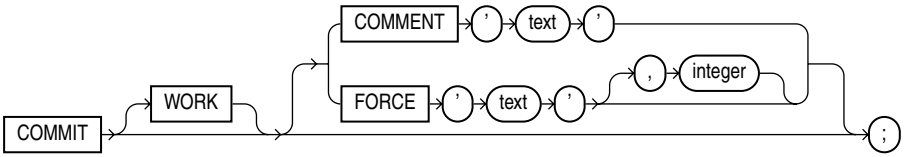
本文

このマニュアルの本文は、次の規則に従って記述されています。

大文字	アルファベットの大文字は、SQL キーワード、ファイル名および初期化パラメータを示します。
イタリック	イタリック体の文字は、SQL 文のパラメータを示します。
太字	太字は、重要な意味を持つ用語を示します。

構文図および表記法

構文図 このマニュアルでは、第 7 章～第 11 章における SQL 文の説明や、第 2 章「Oracle SQL の基本要素」、第 3 章「演算子」、第 4 章「関数」および第 5 章「式、条件および問合せ」におけるその他の SQL 言語の要素の説明に、構文図を使用しています。これらの構文図では、次のとおり、線と矢印を使用して構文構造を示しています。



このような構文図に慣れていない場合、付録 A「構文図」を参照して、読み方を理解してください。そこでは、構文図の構成要素と、SQL 文の書き方の例を説明しています。構文図は、次の項目で構成されます。

キーワード キーワードは、SQL 言語の中では特別な意味を持っています。このマニュアルの構文図では、キーワードは大文字で記述されています。キーワードは、構文図に表記されているとおりに使用する必要がありますが、SQL 文内では大文字でも小文字でも使用できます。たとえば、CREATE TABLE 文では、CREATE TABLE 構文図に表記されているとおり、CREATE キーワードを使用して文を開始する必要があります。

パラメータ パラメータは、構文図の中でプレースホルダの役割を果たします。パラメータは、小文字で記述されます。パラメータは、通常はデータベース・オブジェクト名、Oracle データ型名または式です。構文図にパラメータがある場合、実際の SQL 文では、そのパラメータを適切な型のオブジェクトまたは式に置き換えます。たとえば、CREATE TABLE 文を記述する場合、構文図の *table* パラメータのかわりに、作成する表の名前（たとえば emp）を使用します。パラメータ名は、本文中ではイタリック体で記述されています。

コード例

このマニュアルでは、多数の SQL 文の例を示しています。これらの例は、SQL 文の要素の使用方法を示しています。CREATE TABLE 文の例を次に示します。

```
CREATE TABLE accounts
    ( accno     NUMBER,
      owner     VARCHAR2(10),
      balance   NUMBER(7,2) );
```

例は、本文と異なるフォントで表記されています。

次の規則に従って、例では大文字と小文字を区別して使用しています。

- CREATE や NUMBER などのキーワードは、大文字で表記しています。
- accounts や accno などのデータベースのオブジェクトやその一部の名前は、小文字で表記しています。ただし、本文では大文字で表記しています。
- PL/SQL ブロックは、イタリック体で表記しています。このブロックのキーワードとパラメータが、SQL のキーワードとパラメータでない場合、このマニュアルには記載されていません。詳細は、『Oracle8*i* PL/SQL 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。

多くの例では、その例で作成しないオブジェクトが存在するように記述されています。このようなオブジェクトを作成してからでないと、その例は想定されたように機能しません。

SQL では、（引用符で囲まれた識別子を除いて）大文字と小文字は区別されないため、実際の SQL 文を作成するときに、これらの規則に従う必要はありません。ただし、このように区別しておくことで文が読みやすくなります。

Oracle Tools によっては、SQL 文を特殊文字で終了させる必要がある場合があります。たとえば、このマニュアルで示すコード例は SQL*Plus で記述されているため、セミコロン（;）で終了しています。これらの例文を Oracle に対して発行する場合は、ご使用の Oracle Tool ごとに決められた特殊文字を使用して文を終了させてください。

サンプル・データ

このマニュアルに記載している例の多くでは、サンプル表に対して操作を行います。これらの表の一部は、配布メディアに収録されている SQL スクリプトで定義されています。多くのオペレーティング・システムでは、このスクリプトの名前が `UTLSAMPL.SQL` になっていますが、正確な名前と位置はオペレーティング・システムによって異なります。このスクリプトでは、サンプル・ユーザーを作成し、ユーザー `scott`（パスワード：`tiger`）のスキーマに次のサンプル表を作成します。

```
CREATE TABLE dept
    (deptno    NUMBER(2)      CONSTRAINT pk_dept PRIMARY KEY,
     dname     VARCHAR2(14),
     loc       VARCHAR2(13) );
CREATE TABLE emp
    (empno     NUMBER(4)      CONSTRAINT pk_emp PRIMARY KEY,
     ename     VARCHAR2(10),
     job       VARCHAR2(9),
     mgr       NUMBER(4),
     hiredate  DATE,
     sal       NUMBER(7,2),
     comm      NUMBER(7,2),
     deptno    NUMBER(2)    CONSTRAINT fk_deptno REFERENCES dept );
CREATE TABLE bonus
    (ename     VARCHAR2(10),
     job       VARCHAR2(9),
     sal       NUMBER,
     comm      NUMBER );
CREATE TABLE salgrade
    (grade     NUMBER,
     losal     NUMBER,
     hisal     NUMBER );
```

このスクリプトは、サンプル表に次のデータを挿入します。

```
SELECT * FROM dept;

DEPTNO  DNAME      LOC
------- ---------- ---------
10      ACCOUNTING NEW YORK
20      RESEARCH   DALLAS
30      SALES      CHICAGO
40      OPERATIONS BOSTON

SELECT * FROM emp;

EMPNO  ENAME   JOB          MGR  HIREDATE    SAL   COMM  DEPTNO
-----  ------- --------- ------ --------- ------ ------ -------
 7369  SMITH   CLERK       7902 17-DEC-80    800             20
 7499  ALLEN   SALESMAN    7698 20-FEB-81   1600    300      30
 7521  WARD    SALESMAN    7698 22-FEB-81   1250    500      30
 7566  JONES   MANAGER     7839 02-APR-81   2975             20
 7654  MARTIN  SALESMAN    7698 28-SEP-81   1250   1400      30
 7698  BLAKE   MANAGER     7839 01-MAY-81   2850             30
 7782  CLARK   MANAGER     7839 09-JUN-81   2450             10
 7788  SCOTT   ANALYST     7566 19-APR-87   3000             20
 7839  KING    PRESIDENT        17-NOV-81   5000             10
 7844  TURNER  SALESMAN    7698 08-SEP-81   1500             30
 7876  ADAMS   CLERK       7788 23-MAY-87   1100             20
 7900  JAMES   CLERK       7698 03-DEC-81    950             30
 7902  FORD    ANALYST     7566 03-DEC-81   3000             20
 7934  MILLER  CLERK       7782 23-JAN-82   1300             10

SELECT * FROM salgrade;

GRADE LOSAL HISAL
----- ----- -----
1     700   1200
2     1201  1400
3     1401  2000
4     2001  3000
5     3001  9999
```

bonus 表には、データが含まれません。

スクリプトのすべての操作を実行するには、ユーザー SYSTEM として Oracle にログインしているときにスクリプトを実行してください。

8

SQL 文 : ALTER TABLE ～ *constraint_clause*

この章では、次の SQL 文について説明します。

- ALTER TABLE
- ALTER TABLESPACE
- ALTER TRIGGER
- ALTER TYPE
- ALTER USER
- ALTER VIEW
- ANALYZE
- ASSOCIATE STATISTICS
- AUDIT
- CALL
- COMMENT
- COMMIT
- constraint_clause

ALTER TABLE

用途

ALTER TABLE は、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する場合に使用します。

前提条件

表が自スキーマ内にある必要があります。自スキーマ内にない場合は、その表に対する ALTER 権限または ALTER ANY TABLE システム権限が必要です。また、CREATE ANY INDEX 権限が必要な操作もあります。

パーティション化操作におけるその他の前提条件　また、そのテーブルの所有者でない場合、*drop_partition_clause* または *truncate_partition_clause* を使用するには、DROP ANY TABLE 権限が必要です。

add_partition_clause、*modify_partition_clause*、*move_partition_clause* および *split_partition_clause* を使用する場合、領域を確保する表領域に割当て制限が必要です。

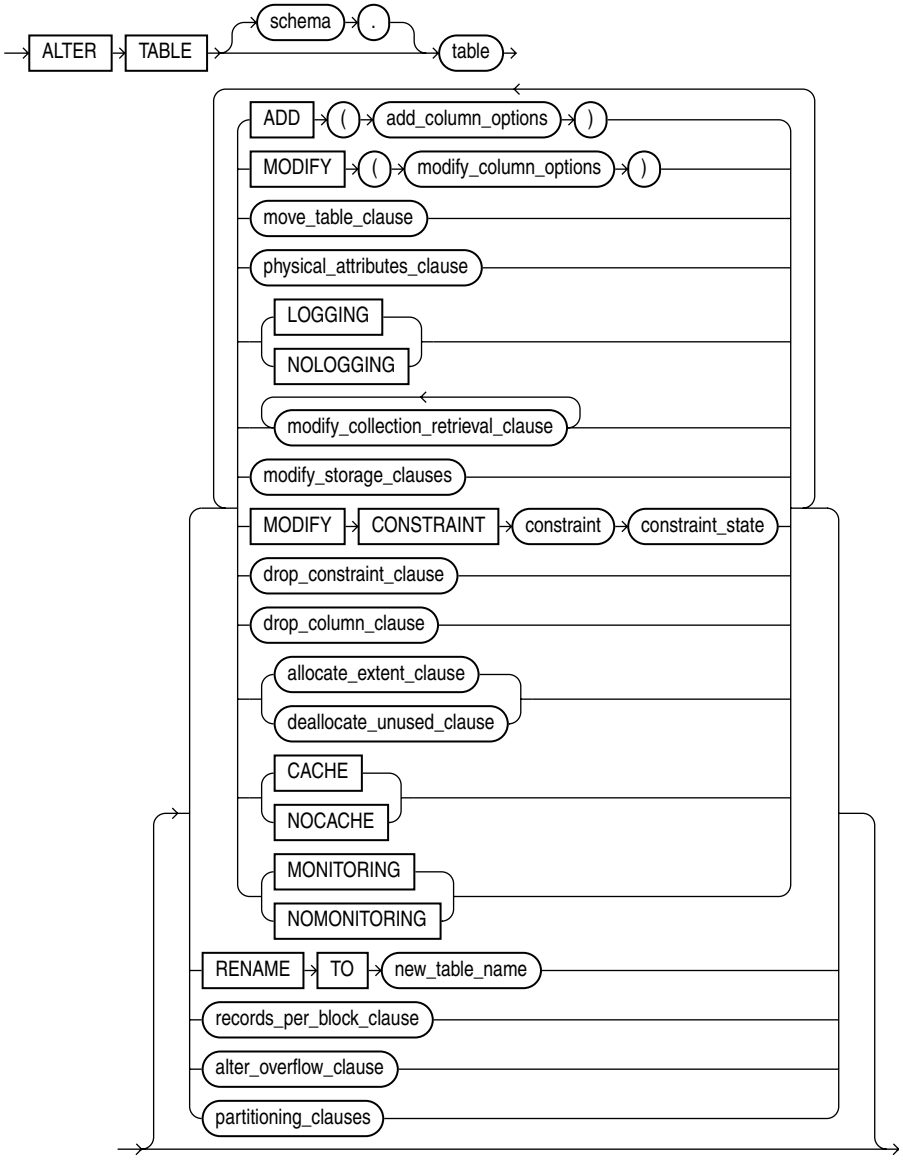
制約およびトリガーにおけるその他の前提条件　一意または主キー制約を有効にする場合は、表に索引を作成するための権限が必要です。Oracle は、その表を含むスキーマにある一意または主キー列に索引を作成するため、この権限が必要になります。

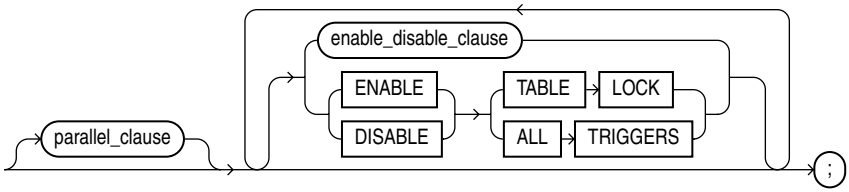
トリガーを使用可能または使用禁止にする場合、トリガーが自スキーマ内にある必要があります。自スキーマにない場合は、ALTER ANY TRIGGER システム権限が必要です。

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。

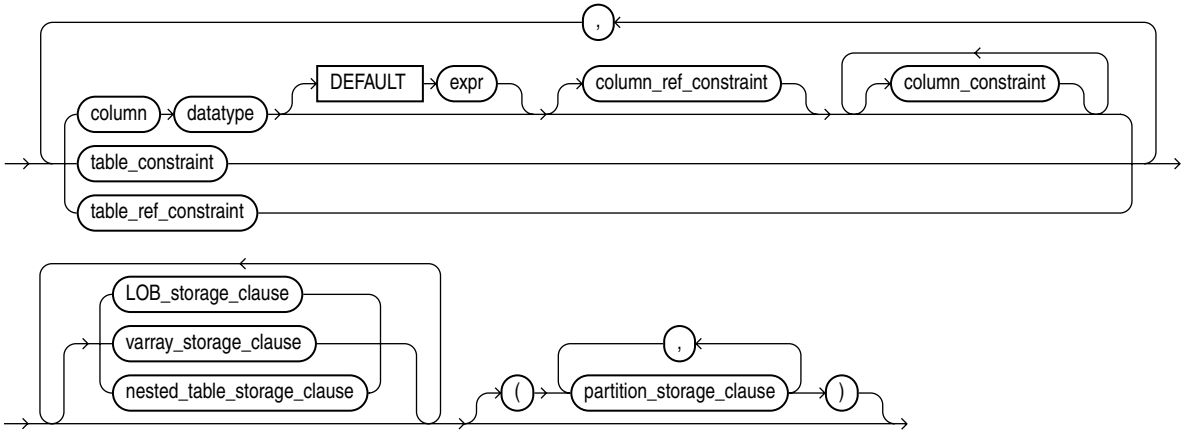
> **参照：** 索引を作成する場合に必要な権限については、9-51 ページの「CREATE INDEX」を参照してください。

構文


ALTER
TABLE
schema
.
table
ADD
(
add_column_options
)
MODIFY
(
modify_column_options
)
move_table_clause
physical_attributes_clause
LOGGING
NOLOGGING
modify_collection_retrieval_clause
modify_storage_clauses
MODIFY
CONSTRAINT
constraint
constraint_state
drop_constraint_clause
drop_column_clause
allocate_extent_clause
deallocate_unused_clause
CACHE
NOCACHE
MONITORING
NOMONITORING
RENAME
TO
new_table_name
records_per_block_clause
alter_overflow_clause
partitioning_clauses

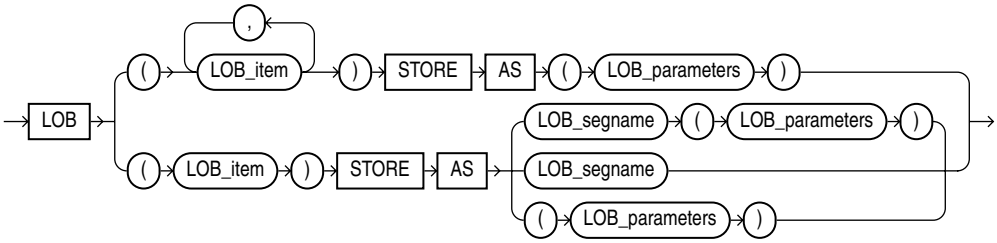


add_column_options::=

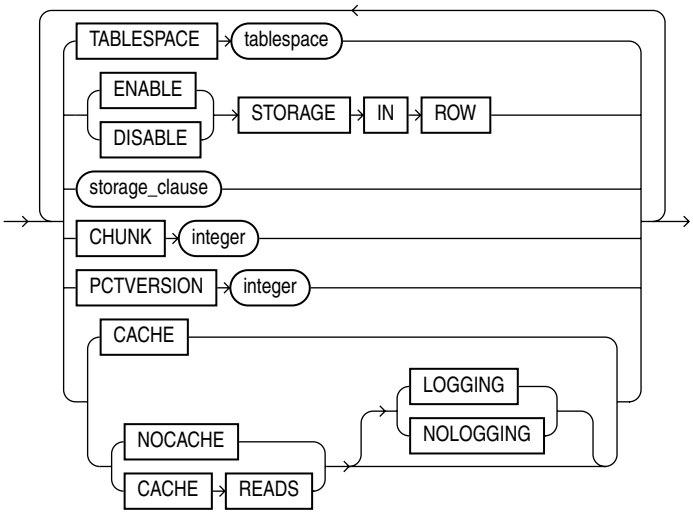


column_constraint, **table_constraint**, **column_ref_constraint**, **table_ref_constraint**, **constraint_state**: 8-134 ページの「constraint_clause」を参照してください。

LOB_storage_clause::=

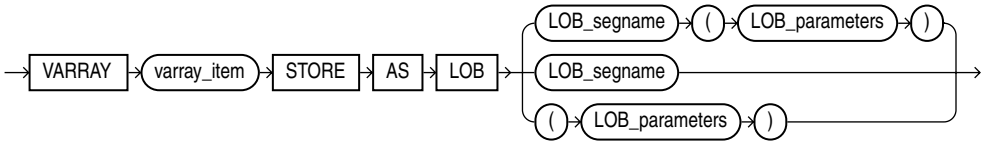


LOB_parameters::=

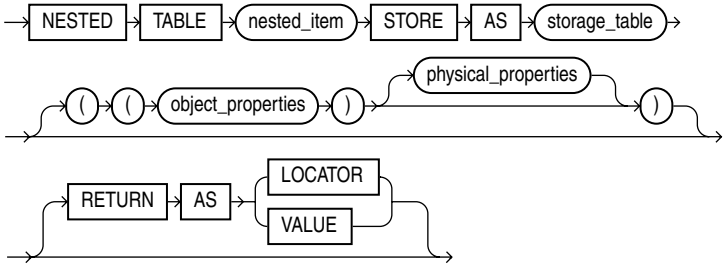


storage_clause: 11-129 ページの「storage_clause」を参照してください。

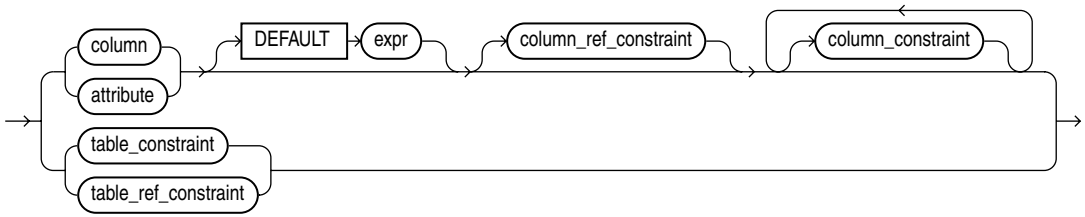
varray_storage_clause::=



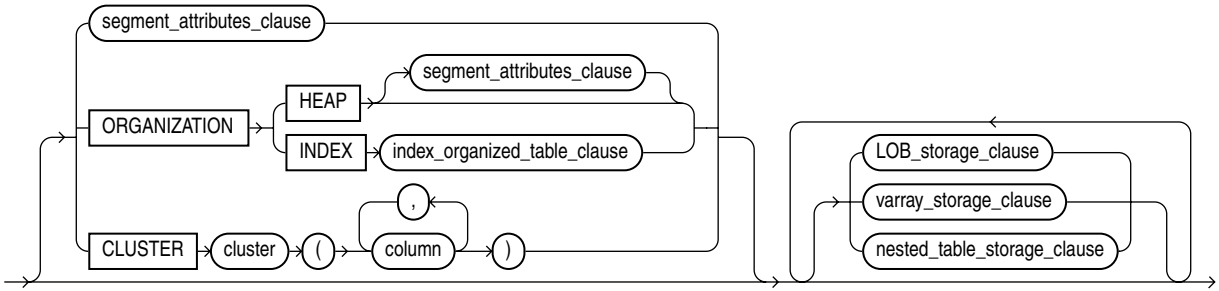
nested_table_storage_clause::=



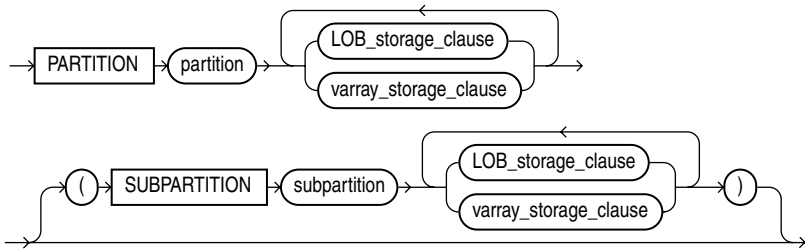
object_properties::=



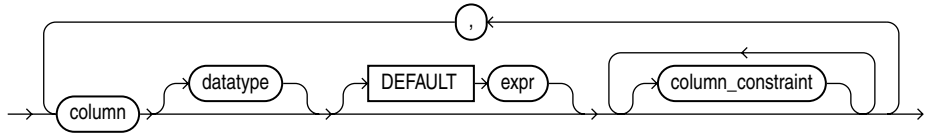
physical_properties::=



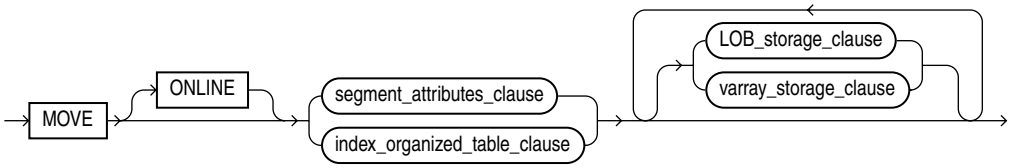
partition_storage_clause::=



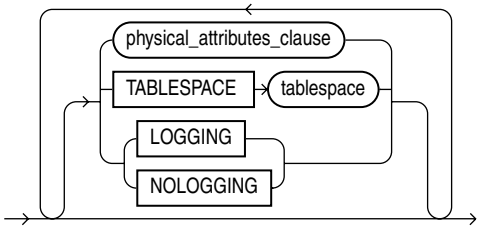
modify_column_options::=



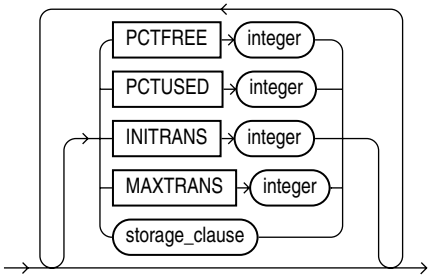
move_table_clause::=



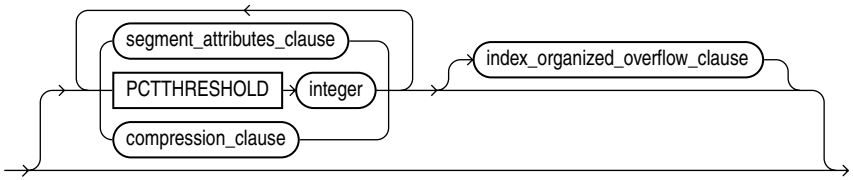
segment_attributes_clause::=



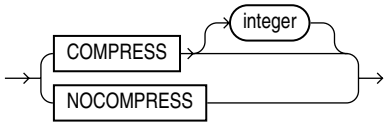
physical_attributes_clause::=



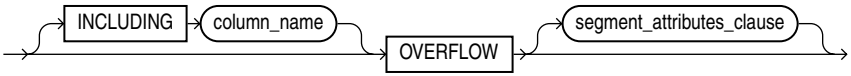
index_organized_table_clause::=



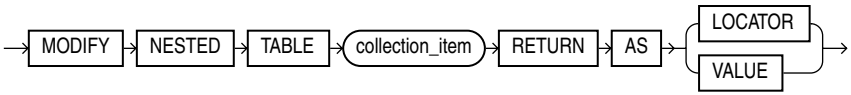
compression_clause::=



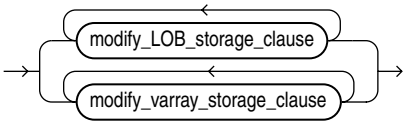
index_organized_overflow_clause::=



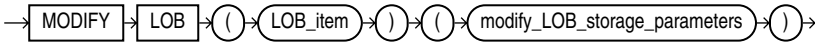
modify_collection_retrieval_clause::=



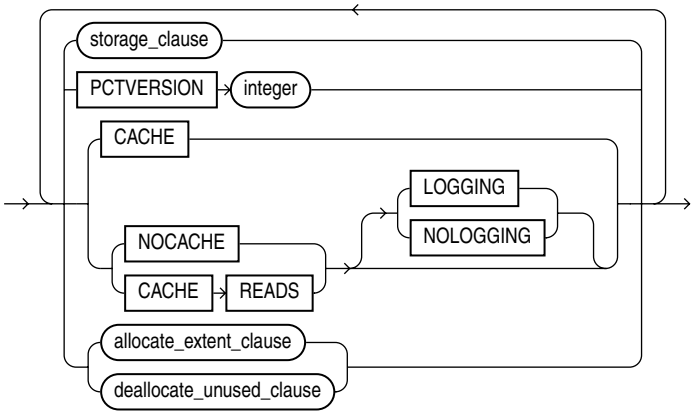
modify_storage_clauses::=



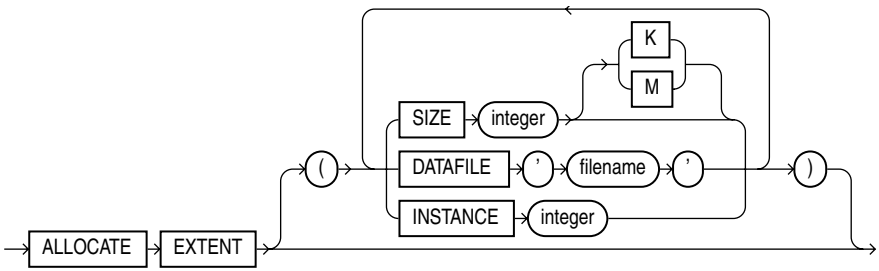
modify_LOB_storage_clause::=



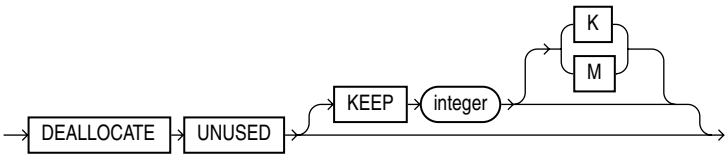
modify_LOB_storage_parameters::=



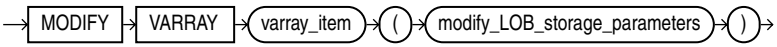
allocate_extent_clause::=



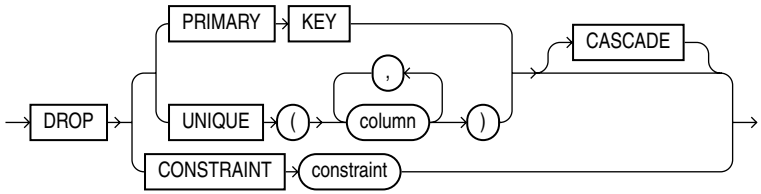
deallocate_unused_clause::=



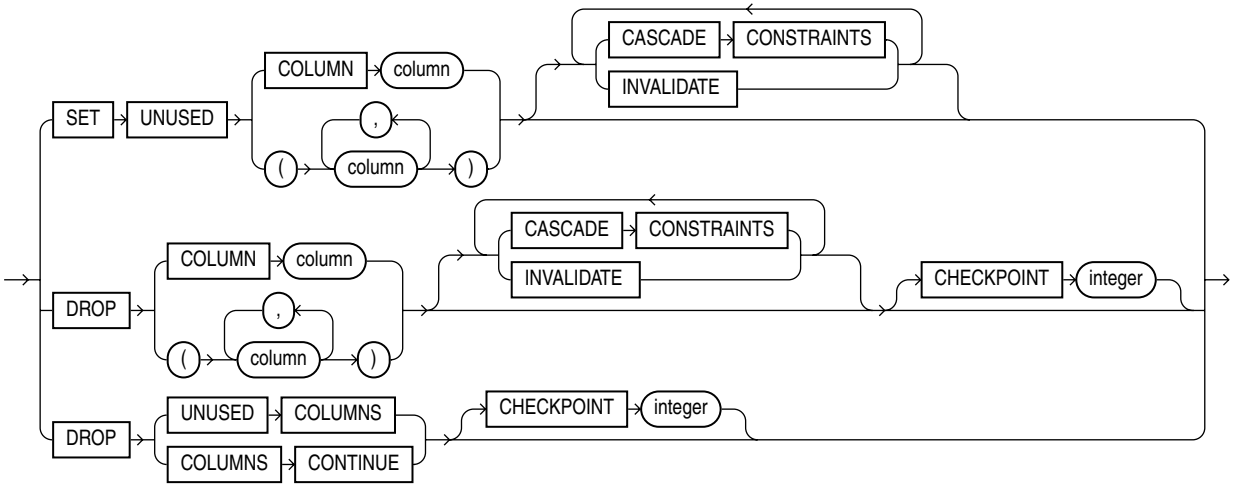
modify_varray_storage_clause::=



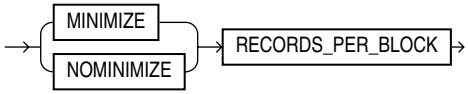
drop_constraint_clause::=



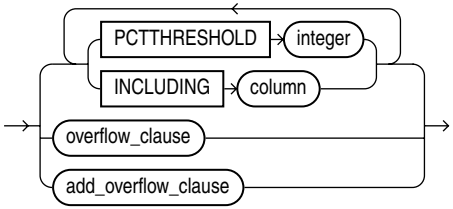
drop_column_clause::=



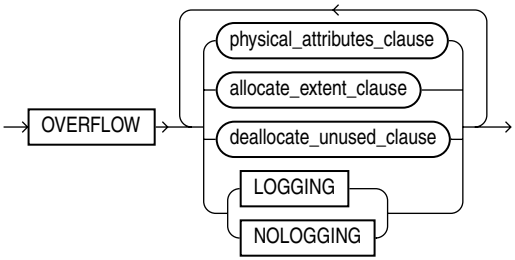
records_per_block_clause::=



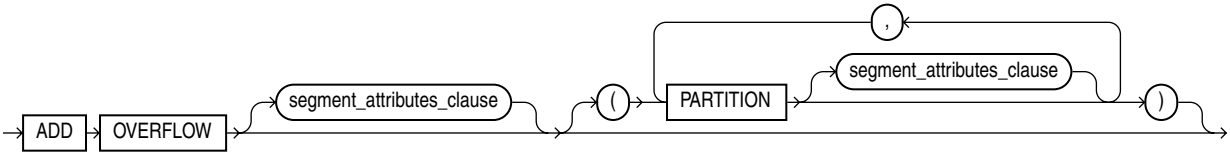
alter_overflow_clause::=



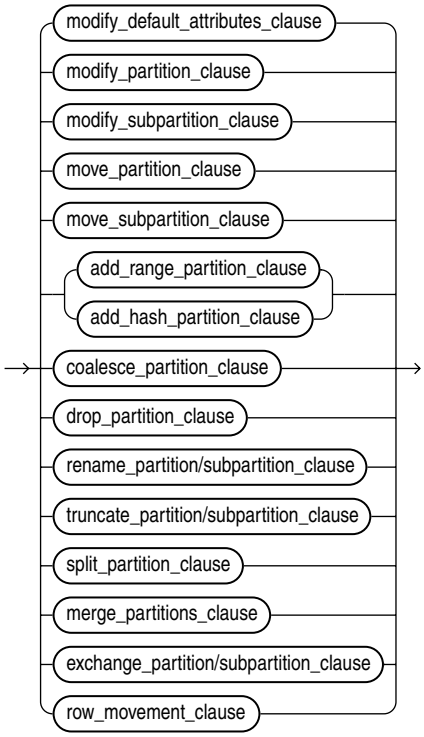
overflow_clause::=



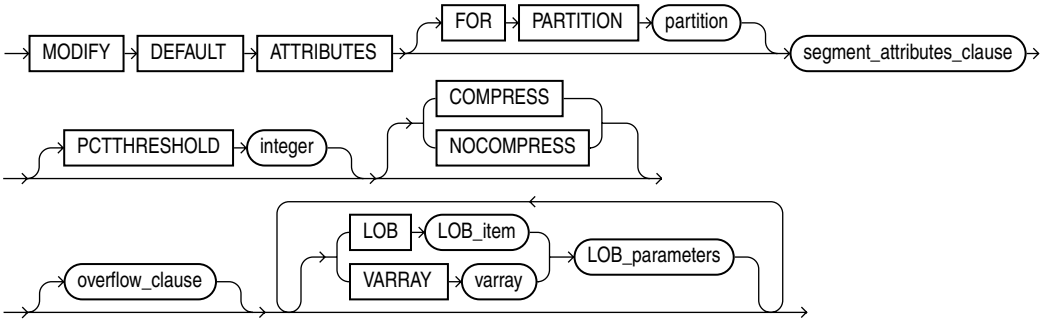
add_overflow_clause::=



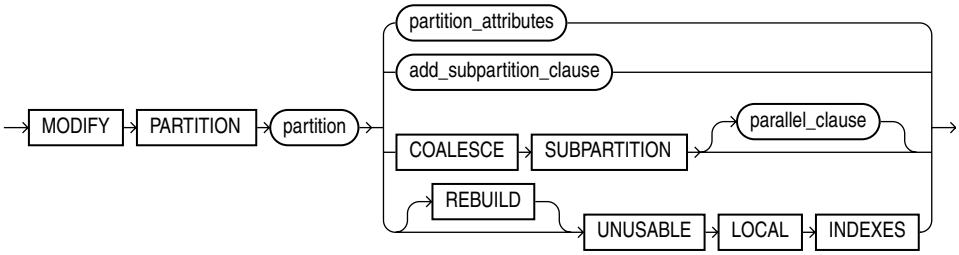
partitioning_clauses::=



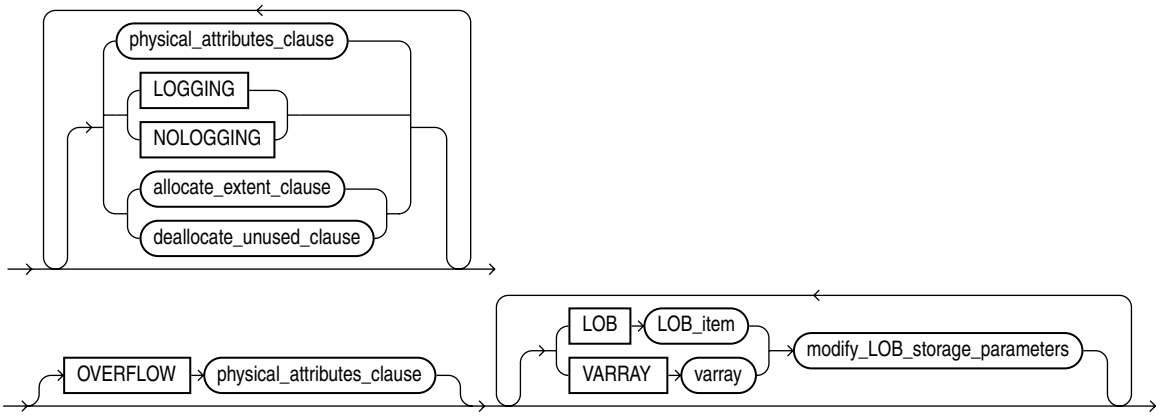
modify_default_attributes_clause::=



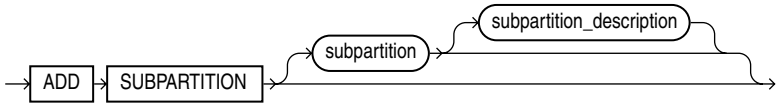
modify_partition_clause::=



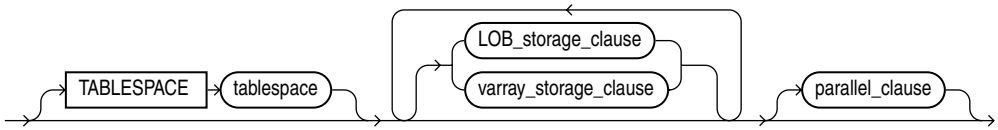
partition_attributes::=



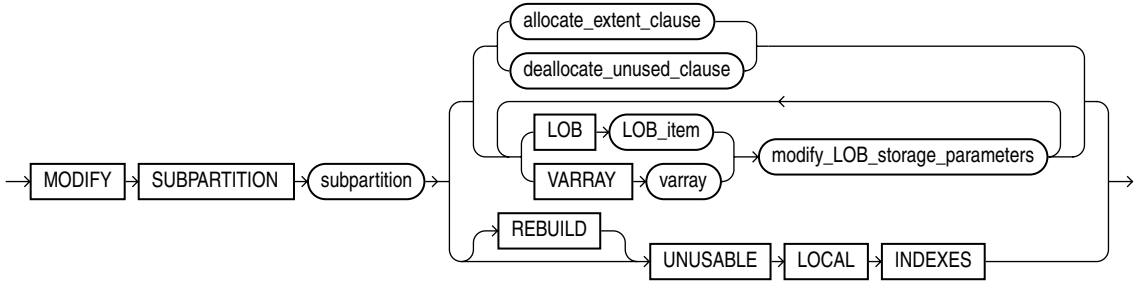
add_subpartition_clause::=



subpartition_description::=

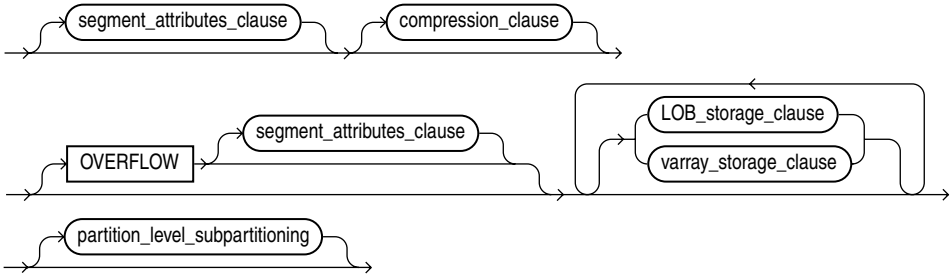


modify_subpartition_clause::=

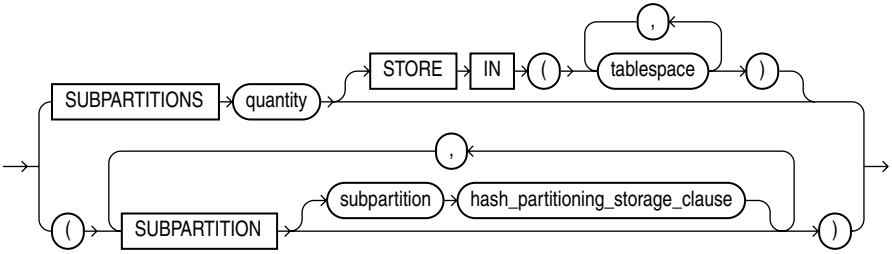


move_partition_clause::=

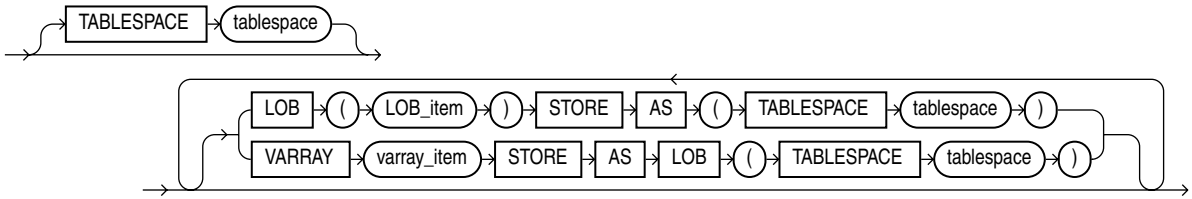
table_partition_description::=



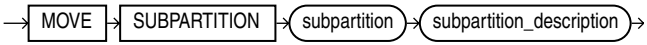
partition_level_subpartitioning::=



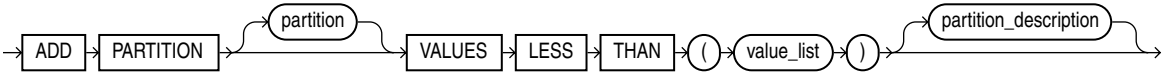
hash_partitioning_storage_clause::=



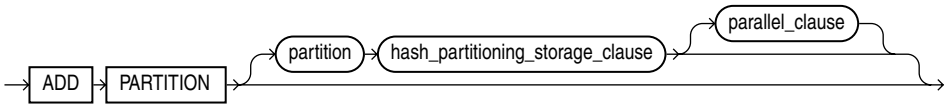
move_subpartition_clause::=



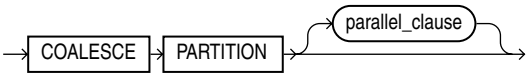
add_range_partition_clause::=



add_hash_partition_clause::=

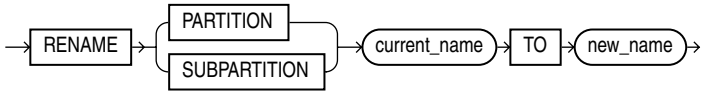


coalesce_partition_clause::=

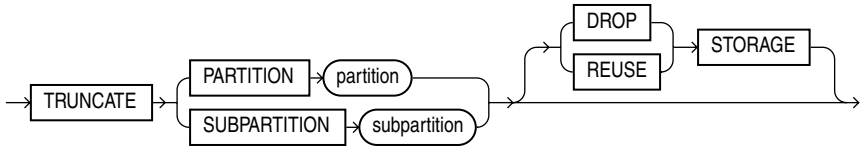


drop_partition_clause::=

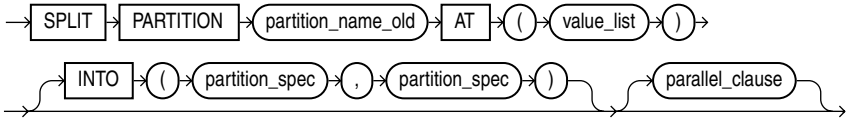
rename_partition/subpartition_clause::=



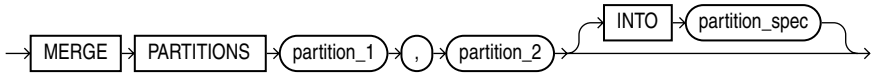
truncate_partition_clause および truncate_subpartition_clause::=



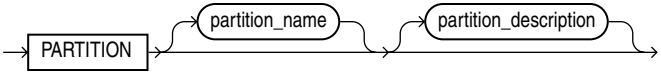
split_partition_clause::=



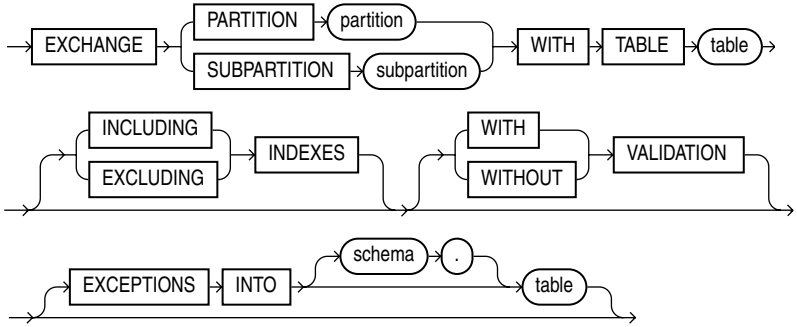
merge_partitions_clause::=



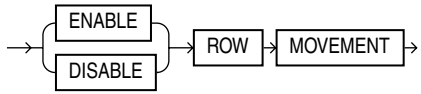
partition_spec::=



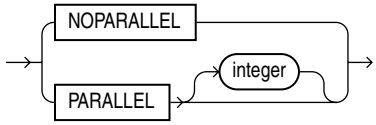
exchange_partition_clause および exchange_subpartition_clause::=



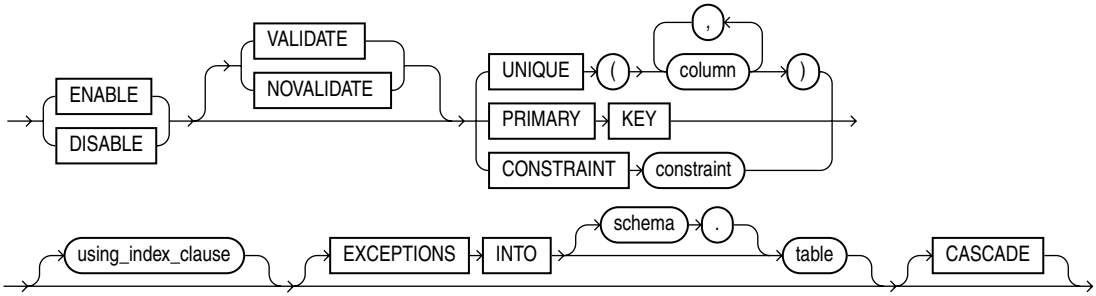
row_movement_clause::=



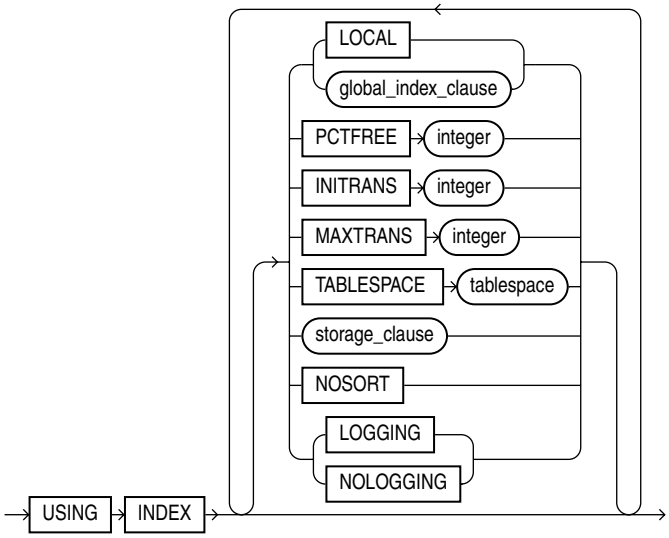
parallel_clause::=



enable_disable_clause::=



using_index_clause::=



キーワードとパラメータ

後述の句は、ALTER TABLE 文で特別な意味を持ちます。その他のキーワードの説明は、10-7 ページの「CREATE TABLE」を参照してください。

注意： ALTER TABLE 文を実行して表を変更すると、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、『Oracle8*i* 概要』を参照してください。

---

schema

変更する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、この表が自スキーマに存在するとみなされます。

table

変更する表の名前を指定します。

表の変更、表からの列を削除または一時表を改名できます。ただし、一時表に対して次のことはできません。

- ネストした表または VARRAY 型の列の追加。その他の型の列の追加。
- 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 *LOB_storage_clause* の TABLESPACE、*storage_clause*、LOGGING または NOLOGGING、*LOB_index_clause* 句の指定。
- *physical_attribute_clause*、*nested_table_storage_clause*、*parallel_clause*、*allocate_extent_clause*、*deallocate_unused_clause* または索引構成表句の指定。
- パーティションと一時表間でのパーティションの交換。
- LOGGING または NOLOGGING の指定。
- MOVE の指定。

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、問合せのリライトでは使用できません。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、7-59 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、『Oracle8*i* データ・ウェアハウス』を参照してください。

add_column_options

ADD *add_column_options* は、列または整合性制約を追加する場合に使用します。

参照： この句のキーワードおよびパラメータの詳細は、10-7 ページの「CREATE TABLE」を参照してください。

列を追加した場合、DEFAULT 句を指定しない限り、新しい列の各行には初期値として NULL が設定されます。この場合、新しい列の各行は、DEFAULT で指定した値で更新されます。この更新操作によって、表に定義された AFTER UPDATE トリガーが起動します。

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、その表に列を追加しても、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。

参照： 10-105 ページの「CREATE VIEW」を参照してください。

制限事項：

- パーティション化された索引構成表には、LOB 列を追加できません（この制限事項はパーティション化されていない索引構成表には適用されません）。
- 表に行がある場合は、DEFAULT 句を指定しない限り、NOT NULL 制約のある列は追加できません。
- 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

新しい列にデフォルト値を指定する場合、または既存の列に新しいデフォルト値を指定する場合は、DEFAULT 句を使用します。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に、デフォルト値を指定した場合、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長さが必要です。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。

table_ref_constraint

および

column_ref_constraint

REF 型の列について詳細に指定します。これらの句の違いは、表レベルから *table_ref* を指定することのみです。したがって、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。REF 型の列または属性を識別した後、*column_ref* を指定してください。

参照： これらの制約の構文および説明は、8-134 ページの「constraint_clause」を参照してください。

column_constraint
既存の列に対して NOT NULL 制約を追加する場合、または既存の列から NOT NULL 制約を削除する場合は、*column_constraint* を使用します。ALTER TABLE にこの句を使用して、他の型の制約を変更することはできません。

参照 : 8-134 ページの「constraint_clause」を参照してください。

table_constraint
表に対しての整合性制約を追加または変更する場合は、*table_constraint* を使用します。

参照 : 8-134 ページの「constraint_clause」を参照してください。

LOB_storage_clause

新しく追加した LOB 列の LOB 記憶特性を指定する場合は、*LOB_storage_clause* を使用します。この句では、既存の LOB 列は変更できません。その場合は、*modify_LOB_storage_clause* を使用してください。

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。
- *table* がパーティション化されている場合、*LOB_index_clause* を指定できません。

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 オブジェクトの属性を指定します。

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。

ENABLE | DISABLE STORAGE IN ROW
LOB 値をインラインまたはアウトラインのどちらに格納するかを指定します（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）。

- ENABLE は、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い場合、LOB 値をインラインに格納することを指定します。これはデフォルト値です。
- DISABLE は、LOB 値の長さにかかわらず、LOB 値を行外に格納することを指定します。

制限事項 : 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_column_options* 句の一部には指定できません。ただし、新しい列 *add_column_options* を追加するとき、または表 *move_table_clause* を移動するときに、この設定を変更することはできます。

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・ブロック・サイズが 2048 の場合、*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 で許容されるブロック・サイズの最大値です。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、一度設定すると変更できません。

注意： CHUNK の値は、NEXT の値（デフォルト値または *storage_clause* で指定した値）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。

PCTVERSION *integer* LOB の記憶領域全体のうち、新規バージョンの LOB の作成に使用される最大記憶領域の割合（パーセント）を指定します。デフォルト値は 10 です。これは、LOB の記憶領域全体の 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。

LOB_index_clause この句は、Oracle8*i* 以降は使用されません。Oracle は各 LOB 列に対して索引を生成します。LOB 索引は、システムによって名前が付けられ管理されており、LOB データ・セグメントと同じ表領域に格納されます。

この句の指定が可能な場合もありますが、指定しないことをお薦めします。どんな場合でも、LOB 索引を LOB データと異なる表領域に置かないでください。

参照： 以前のバージョンから移行された表の LOB 索引の管理方法については、『Oracle8*i* 移行ガイド』を参照してください。

varray_storage_clause

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する場合は、*varray_storage_clause* を使用します。また、この句を指定した場合、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。

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 表領域は、デフォルトではそれを格納する表の表領域になります。

nested_table_storage_clause

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する場合は、*nested_table_storage_clause* を使用します。ネストした表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）。

制限事項 :

- *parallel_clause* は指定できません。
- （*segment_attributes_clause* の一部として）TABLESPACE をネストした表に指定することはできません。表領域は常に親表の表領域です。

nested_item ネストした表型の列（または、その表のオブジェクト型の最上位属性）の名前を指定します。

storage_table *nested_item* の列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。

partition_storage_clause

個々のパーティションに、別々の *LOB_storage_clause* または *varray_storage_clause* を指定する場合は、*partition_storage_clause* を設定します。パーティションは、その位置の順に指定してください。

特定のパーティションに、*LOB_storage_clause* または *varray_storage_clause* を指定しなかった場合、その記憶特性は、表レベルで LOB 項目に指定された記憶特性になります。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。

制限事項 : ALTER TABLE 文ごとに *partition_storage_clauses* のリストを 1 つのみ指定できます。*LOB_storage_clauses* および *varray_storage_clauses* はすべて、*partition_storage_clauses* のリストの前に配置する必要があります。

modify_column_options

既存の列定義を変更する場合は、MODIFY *modify_column_options* を使用します。列定義のオプション部分（データ型、デフォルト値または列制約）を省略した場合、その部分は変更されません。

- 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、列のすべての行が NULL 値である場合、または列のサイズを変更しない場合のみです。
- 列のデータ型を変更、またはサイズを小さくできるのは、列のすべての行が NULL 値の場合のみです。
- すべての列が NULL 値であるかどうかにかかわらず、文字列と未処理列のサイズまたは数値列の精度は、いつでも大きくできます。

制限事項：

- 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。
- ドメイン・インデックスが構築されている列の定義は変更できません。
- 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。

column

追加または変更する列名を指定します。

列の制約構文と MODIFY 句を使用して、既存の列に追加できる整合性制約は NOT NULL 制約のみで、その列に NULL 値がない場合に限ります。既存の列にこれ以外の整合性制約（一意、主キー、参照整合性および CHECK 制約）を定義する場合は、ADD 句と表の制約構文を使用します。

datatype

既存の列に新しいデータ型を指定します。

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。

マテリアライズド・ビューの記憶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：**マテリアライズド・ビュー・ログの再検証の詳細は、7-59 ページの「ALTER MATERIALIZED VIEW」を参照してください。

制限事項 :

- 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。
- 列のデータ型は、LOB データ型や REF データ型には変更できません。

MODIFY CONSTRAINT *constraint*

MODIFY CONSTRAINT を使用して、既存の制約 *constraint* の状態を変更することができます。

参照 : *constraint_state* のすべてのキーワードおよびパラメータの詳細は、8-134 ページの「constraint_clause」を参照してください。

move_table_clause

ヒープ構成表の場合は、構文の *segment_attributes_clause* を使用してください。*move_table_clause* によって、非パーティション表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 を使用して、その表に関連付けられた 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）。

索引構成表の場合は、構文の *index_organized_table_clause* を使用してください。*move_table_clause* は、索引構成表の主キー索引 B* ツリーを再構築します。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限り、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）のいずれかが明示的に移動される場合は、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 の一部として明示的に指定しない限り、再構築されません。

ONLINE

その表の主キー索引 B* ツリーの再構築中に、索引構成表に対する DML 操作を可能にする場合は、ONLINE を指定します。

制限事項：

- この句は、パーティション化されていない索引構成表に対してのみ指定できます。
- 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 はエラーを戻します。

compression_clause

索引構成表に対するキー圧縮を有効化または無効化する場合は、*compression_clause* を使用します。

- COMPRESS は、キー圧縮を使用可能にします。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。

  制限事項：

  - この句は、索引構成表に対してのみ指定できます。
  - 圧縮を表のレベルで指定した場合のみ、索引構成表のパーティションに圧縮を指定できます。

- NOCOMPRESS は、索引構成表でのキー圧縮を使用禁止にします。これはデフォルト値です。

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。

***move_table_clause* の制限事項：**

- MOVE を指定する場合は、最初の句である必要があります。索引構成表では、この句以外に使用できる句は、*physical_attribute_clause* および *parallel_clause* のみです。ヒープ構成表では、前述の 2 つの句に加えて *LOB_storage_clauses* も指定できます。
- 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。

参照： 8-44 ページの「move_partition_clause」および 8-45 ページの「move_subpartition_clause」を参照してください。

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。
- 表の LOB 索引がその LOB データと異なる表領域にある場合、移動の後、LOB 索引は LOB データと一緒に LOB データの表領域にまとめて格納されます。

---

physical_attributes_clause

physical_attributes_clause は、PCTFREE、PCTUSED、INITRANS および MAXTRANS パラメータの値および記憶特性を変更します。

制限事項： 索引構成表の索引セグメントに対して、PCTUSED パラメータを指定することはできません。

参照： 10-7 ページの「CREATE TABLE」および 11-129 ページの「storage_clause」パラメータ PCTFREE、PCTUSED、INITRANS および MAXTRANS を参照してください。

注意：

- 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オーバーライドされます。
- レンジ・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オーバーライドされます。既存パーティションの値を変えずにデフォルトの表属性を変更する場合は、*modify_default_attributes_clause* 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オーバーライドされます。既存のサブパーティションの値を変え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_default_attributes_clause* を使用してください。

modify_collection_retrieval_clause

データベースから集合項目が取り出されたときの戻り値を変更する場合は、*modify_collection_retrieval_clause* を使用します。

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。

modify_storage_clauses

modify_LOB_storage_clause

LOB *lob_item* の物理属性を変更する場合は、*modify_LOB_storage_clause* を使用します。
各 *modify_LOB_storage_clause* には、*lob_item* を 1 つのみ指定できます。

制限事項 :

- LOB 記憶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。

modify_varray_storage_clause

VARRAY が格納される既存の LOB の記憶特性を変更する場合は、*modify_varray_storage_clause* を使用します。

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 表領域は、デフォルトではそれを格納する表の表領域になります。

drop_constraint_clause

データベースの整合性制約を削除する場合は、*drop_constraint_clause* を使用します。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。

PRIMARY KEY

表の主キー制約を削除する場合は、PRIMARY KEY を指定します。

UNIQUE

指定した列の一意制約を削除する場合は、UNIQUE を指定します。

CONSTRAINT *constraint*

削除する整合性制約を指定します。

CASCADE

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する場合は、CASCADE を指定します。

***drop_constraint_clause* の制限事項：**

- 参照整合性制約の一部である一意または主キー制約は、外部キーも削除しない限り削除できません。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 を省略した場合、いずれかの外部キーが主キーまたは一意制約を参照していると、それらは削除されません。
- 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。
- 列の有効範囲は削除できません。

drop_column_clause

drop_column_clause は、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します。

- ネストした表の列を削除した場合、その記憶表も削除されます。
- LOB 列を削除した場合、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 列を削除した場合、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）する場合は、この列の直前に保存された列が新しい INCLUDING 列になります。

SET UNUSED

未使用として 1 つ以上の列をマークする場合は、SET UNUSED を使用します。この句を指定した場合でも、実際に目的の列が表の各行から削除されるわけではありません（これらの列が使用しているディスク領域がリストアされるわけではありません）。このため、応答時間は DROP 句を使用した場合よりも速くなります。

未使用のマークが付いた列を持つ表はすべて、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 で表示できます。

参照：これらのビューの詳細は、『Oracle8*i* リファレンス・マニュアル』を参照してください。

未使用列は削除されたように扱われますが、その列データは表の行に残っています。未使用のマークを付けた列にはアクセスできなくなります。SELECT * 問合せでも、未使用列からはデータを取り出すことができません。また、未使用のマークを付けた列の名前および型は、DESCRIBE 中には表示されず、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、表あたり 1000 列の絶対限界まで、これらの列もカウント対象になります。ただし、DLL 文でこの句の結果をロールバックすることはできません。つまり、SET USED に相当するものを発行しても、SET UNUSED 列を取り出すことはできません。

また、LONG 型の列に未使用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限り、その表には別の LONG 列を追加できません。

参照 : 1000 列の制限に関する詳細は、10-7 ページの「CREATE TABLE」を参照してください。

DROP

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する場合は、DROP を指定します。特定の列を明示的に削除した場合、対象の表にある未使用のマークが付けられた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、その統計タイプを使用して収集したすべての統計情報。FORCE オプションによって、関連付けは解除されます。

参照 : 統計タイプの関連付けの解除については、10-121 ページの「DISASSOCIATE STATISTICS」を参照してください。

注意：対象の列が他の列の親キーである場合またはチェック制約が対象列とその他の列を参照している場合は、Oracle はエラーを返し、CASCADE CONSTRAINTS 句を指定しない限り、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。

DROP UNUSED COLUMNS

表から未使用にマークされている列をすべて削除する場合は、DROP UNUSED COLUMNS を指定します。表の未使用の列からディスク領域を再利用する場合に、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限り、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

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参照整合性制約をすべて削除する場合、および削除する列に定義されている複数列制約をすべて削除する場合は、CASCADE CONSTRAINTS を指定します。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 を指定します。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

注意：現在、キーワード INVALIDATE の指定にかかわらず、Oracle はこの句を実行します。

ビュー、トリガー、ストアド・プログラム・ユニットなどの依存オブジェクトをすべて無効にします。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、Oracle 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、無効になるのはローカル依存性のみです。

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有効になります。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、『Oracle8*i* 概要』を参照してください。

CHECKPOINT　整数列を処理した後に列削除操作のチェックポイントを Oracle に適用させる場合は、CHECKPOINT を使用します。このとき整数は任意の値であり、0 以上である必要があります。整数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。整数を指定しなかった場合は、512（デフォルト）が設定されます。

チェックポイントの実行によって、列の削除操作中に蓄積される取消しログの量が減り、ロールバック・セグメント領域を使い切らないようにできます。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　中断されたところから列削除操作を続ける場合は、DROP COLUMNS CONTINUE を指定します。表が有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。

***drop_column_clause* の制限事項：**

- この句の各部分は、文の中で 1 回のみ指定でき、他の ALTER TABLE 句と一緒に使用できません。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。オブジェクト型列の属性を削除することはできません。
- 索引構成表からは、主キー列でない場合に限り、列を削除できます。索引構成表の主キー制約は絶対に削除できないため、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で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。

- ドメイン・インデックスが構築されている列は削除できません。
- 次のものは、この句を使用して削除できません。
  - 擬似列、クラスタ化列またはパーティション列
    (ただし、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、オブジェクト表または SYS が所有する表の列。

allocate_extent_clause

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場合は、*allocate_extent_clause* を使用します。

制限事項 : レンジ・パーティションまたはコンポジット・パーティション表にエクステントを割り当てることはできません。

注意 : この句を使用して明示的にエクステントを割り当てた場合、次に割り当てられるエクステント・サイズには、NEXT および PCTINCREASE の記憶領域パラメータで指定したサイズが反映されます。

SIZE *integer*
: エクステント・サイズをバイト単位で指定します。K または M を使用して、KB または MB 単位でエクステント・サイズを指定することもできます。このパラメータを省略した場合、表のオーバーフロー・データ・セグメントまたは LOB 索引の STORAGE パラメータに基づいてサイズが決定されます。

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、表の表領域、オーバーフロー・データ・セグメント、LOB データ表領域または LOB 索引の中から 1 つ指定します。このパラメータを省略した場合、Oracle によって選択されます。

INSTANCE *integer*
: 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 INSTANCE *integer* を指定します。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、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。この場合には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 : パラレル・モードの Parallel Server で Oracle を使用している場合に限り、このパラメータを使用できます。

参照 : 『Oracle8*i* 概要』を参照してください。

deallocate_unused_clause

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し、解放された領域を表領域の他のセグメントから利用できるようにする場合は、*deallocate_unused_clause* を使用します。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。

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。

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。

解放される領域の実際の量は、INITIAL、MINEXTENTS および NEXT パラメータによって異なります。

参照 : これらのパラメータについては、11-129 ページの「storage_clause」を参照してください。

KEEP *integer*　割当てを解除した後に表、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 索引に残す、最高水位標を超えるバイト数を指定します。

- KEEP を省略した場合、最高水位標が INITIAL および MINEXTENTS のサイズを超えるときは、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。
- どちらの場合も、NEXT は、割当てを解除した最後のエクステント・サイズに設定されます。

CACHE | NOCACHE

CACHE

この句は、アクセス頻度の高いデータについて、フル・テーブル・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 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。

LOB_storage_clause のパラメータとして CACHE を使用する場合は、より高速にアクセスできるように、バッファ・キャッシュに LOB データ値が事前に配置されるように指定します。

制限事項 : 索引構成表に CACHE を指定することはできません。ただし、索引構成表は暗黙的に CACHE 動作を指定します。

NOCACHE

この句は、アクセス頻度の低いデータについて、フル・テーブル・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 LRU リストの最低使用頻度側に置く場合に指定します。

LOB_storage_clause のパラメータとして、NOCACHE は、LOB 値がバッファ・キャッシュに入れられないか、またはバッファ・キャッシュに入れられ、LRU リストの最低使用頻度側に置かれるかのいずれかを指定します（デフォルトでは後者です）。NOCACHE は、LOB 記憶域のデフォルトです。

制限事項 : 索引構成表に NOCACHE は指定できません。

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込み操作中にバッファ・キャッシュに入れられることを指定します。

- 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。
- 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。

既存の LOB に対し、CACHE、NOCACHE または CACHE READS を指定しない場合、Oracle は、LOB 属性の既存値を保持します。

MONITORING | NOMONITORING

MONITORING　表の変更統計を収集するように指定する場合は、MONITORING を指定します。これらの統計表は、一定の期間に DML 文の影響を受ける行数の推定値です。これらは、オプティマイザが使用またはユーザーが分析する場合に使用できます。

参照 : この句の使用方法の詳細は、『Oracle8*i* パフォーマンスのための設計およびチューニング』を参照してください。

NOMONITORING　表の変更統計を収集しないように指定する場合は、NOMONITORING を指定してください。

制限事項 : 一時表に対して MONITORING または NOMONITORING は指定できません。

LOGGING | NOLOGGING

LOGGING | NOLOGGING　非パーティション表、表パーティション、パーティション表のすべてのパーティションまたは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後続のダイレクト・ローダー（SQL*Loader）およびダイレクト・ロードの INSERT 操作について、REDO ログ・ファイルにログを記録するか（LOGGING）、記録しないか（NOLOGGING）を指定します。

modify_default_attributes_clause 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。

LOGGING|NOLOGGING は、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。

表または表パーティションに対して、LOGGING|NOLOGGING を省略した場合、表が存在する表領域のロギング属性が、その表または表パーティションのデフォルトのロギング属性として使用されます。

LOB に対して LOGGING|NOLOGGING を省略した場合、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、行データとともにインラインに格納された LOB に適用されません。つまり、LOB に対する NOLOGGING を 400 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。

NOLOGGING モードでは、データの変更時に、（新しいエクステントを無効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表の場合は、NOLOGGING 操作の後にバックアップを取る必要があります。

データベースが ARCHIVELOG モードで実行されている場合、LOGGING モードでの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、表がリストアされます。ただし、NOLOGGING モードでの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、表はリストアされません。

ベース表のロギング属性は、その索引のロギング属性に依存しません。

> **参照：** *logging_clause* およびパラレル DML の詳細は、『Oracle8*i* Parallel Server 概要』を参照してください。

RENAME TO

RENAME TO　　表を *new_table_name* に改名する場合は、RENAME 句を使用します。

注意 : 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。

参照 : マテリアライズド・ビューの詳細は、9-86 ページの「CREATE MATERIALIZED VIEW」および『Oracle8*i* データ・ウェアハウス』を参照してください。

records_per_block_clause

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する場合は、*records_per_block_clause* を使用します。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小さくなり（圧縮され）ます。

制限事項 :

- 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。
- 索引構成表およびネストした表にはこの句を指定できません。

MINIMIZE　　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するために、MINIMIZE を指定します。

制限事項 : 空の表に MINIMIZE は指定できません。

NOMINIMIZE　　MINIMIZE 機能を無効にするために、NOMINIMIZE を指定します。これはデフォルト値です。

alter_overflow_clause

alter_overflow_clause を使用して、索引構成表の定義を変更します。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。

注意 :　索引構成表を変更した場合、各列の最大サイズが評価され、許容される行サイズ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はなくなります。

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。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。

制限事項：

- 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。
- 索引構成表の個別パーティションに対して PCTTHRESHOLD は指定できません。

参照：*index_organized_table_clause* の INCLUDING 句を参照してください。

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非主キー列でもかまいません。*column_name* に続くすべての非主キー列は、オーバーフロー・データ・セグメントに保存されます。

制限事項：索引構成表の個別パーティションにこの句は指定できません。

注意：*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 の値（指定した値またはデフォルト）を超えると、Oracle は PCTTHRESHOLD の値に基づいて、行を切り離します。

overflow_clause

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属性およびロギング属性を指定する場合に、*overflow_clause* を使用します。この句に指定するパラメータは、オーバーフロー・データ・セグメントにのみ適用できます。

制限事項：パーティション化された索引構成表のパーティションには、その表にすでにオーバーフロー・セグメントがない限り、OVERFLOW を指定することはできません。

参照：10-7 ページの「CREATE TABLE」を参照してください。

add_overflow_clause

指定した索引構成表にオーバーフロー・データ・セグメントを追加する場合に、*add_overflow_clause* を使用します。

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、表の中のパーティションそれぞれに個別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のではなく、パーティションが作成された順番で属性を指定してください。

パーティションの順番は、USER_IND_PARTITIONS ビューの PARTITION_NAME および PARTITION_POSITION 列の問合せから得ることができます。

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。表のレベルで 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。

partitioning_clauses

次の句は、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、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表に対する操作と組み合せて行うことはできません。

注意： 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。

---

modify_default_attributes_clause

modify_default_attributes_clause では、表の属性に対する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 パーティション作成時に明示的にオーバーライドしない限り、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、指定されたデフォルト値は、個々のパーティション・レベルで指定された属性でオーバーライドされます。

FOR PARTITION　FOR PARTITION は、コンポジット・パーティション表にのみ適用されます。*partition* 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 *partition* 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オーバーライドしない限り、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。

制限事項 :

- PCTTHRESHOLD、COMPRESS、*physical_attributes_clause* および *overflow_clause* 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。
- 索引構成表の索引セグメントに対しては、PCTUSED パラメータを指定できません。
- 圧縮がすでに表レベルで指定されている場合にのみ、COMPRESS を指定できます。

modify_partition_clause

modify_partition_clause によって、表パーティション *partition* の実際の物理属性を変更します。そのパーティションの 1 つ以上の LOB 項目の記憶属性を任意に変更できます。パーティションに対して、物理属性であるロギング属性 PCTFREE、PCTUSED、INITRANS および MAXTRANS パラメータまたは記憶領域パラメータに新しい値を指定できます。

表がコンポジット・パーティション表の場合は、次の点に注意してください。

- *allocate_extent_clause* を指定した場合、*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* を指定した場合、*partition* のそれぞれのサブパーティションから未使用のエクステントを解放します。
- 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、既存の値はオーバーライドされます。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_default_attributes_clause* に FOR PARTITION 句を指定します。

制限事項 : 表がハッシュ・パーティション化されている場合、*allocate_extent* および *deallocate_unused* 句のみを指定できます。パーティションのその他の属性は、TABLESPACE 以外の表レベルのデフォルトによって継承されます。TABLESPACE は作成時と同じ状態です。

add_subpartition_clause

add_subpartition_clause によって、*partition* にハッシュ・パーティションを追加できます。Oracle は、ハッシュ関数によって *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入れます。

追加および再ハッシュされたサブパーティションに対応するローカル索引サブパーティションに、UNUSABLE のマークが付けられます。このサブパーティションは、再構築する必要があります。

subpartition を指定しなかった場合、SYS_SUBP*nnn* という形式の名前が割り当てられます。

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。

COALESCE SUBPARTITION

ハッシュ・サブパーティションが選択され、その内容が 1 つ以上の他のサブパーティション（ハッシュ関数が決定する）に分割された後、選択されたサブパーティションが削除されるようにする場合は、COALESCE PARTITION を指定します。

選択されたサブパーティションに対応するローカル索引サブパーティションも削除されます。1 つ以上の吸収サブパーティションに対応する索引サブパーティションに、UNUSABLE のマークが付けられます。この索引サブパーティションは、再構築する必要があります。

UNUSABLE LOCAL INDEXES

次の 2 つの句は、*partition* に対応するローカル**索引パーティション**の属性を変更します。

- UNUSABLE LOCAL INDEXES によって、*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 のマークが付けられます。
- REBUILD UNUSABLE LOCAL INDEXES によって、*partition* に関連付けられたすべての使用禁止のローカル索引パーティションが再構築されます。

制限事項 :

- この句を *modify_partition_clause* の他の句とともに指定することはできません。
- この句をサブパーティション化されているパーティションに対して指定することはできません。

modify_subpartition_clause

modify_subpartition_clause によって、表の個々のサブパーティションに対して、記憶域の割当てまたは解放を行います。

制限事項 : サブパーティションに指定できる *modify_LOB_storage_parameters* は、*allocate_extent_clause* および *deallocate_unused_clause* のみです。

- UNUSABLE LOCAL INDEXES によって、*subpartition* に関連付けられた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。
- REBUILD UNUSABLE LOCAL INDEXES によって、*subpartition* に関連付けられたすべての使用禁止のローカル索引サブパーティションが再構築されます。

rename_partition/subpartition_clause

表パーティションまたは表サブパーティションを *current_name* から *new_name* に改名する場合は、*rename_partition_clause* または *rename_subpartition_clause* を使用します。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。

move_partition_clause

表パーティション *partition* を別のセグメントへ移動する場合は、*move_partition_clause* を使用します。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。

partition が空でない場合は、MOVE PARTITION によって、すべての対応するローカル索引パーティション、すべてのグローバルな非パーティション索引、およびグローバルな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。

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。どちらも指定されていない場合は、パラレル化を使用せずに移動が行われます。

MOVE PARTITION の *parallel_clause* は、*table* のデフォルトのパラレル属性を変更しません。

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 が組み立てられます。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： 論理 ROWID の詳細は、『Oracle8*i* 概要』を参照してください。

制限事項：

- *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。
- コンポジット・パーティション表のパーティションは移動できません。それぞれのサブパーティションは、*move_subpartition_clause* とは別に移動する必要があります。
- 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_subpartition_clause* を使用してサブパーティションを移動できます。

move_subpartition_clause

表サブパーティション *subpartition* を別のセグメントに移動する場合は、*move_subpartition_clause* を使用します。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限り、移動される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。また、グローバルな非パーティション索引、およびグローバルな索引パーティションにも、UNUSABLE のマークが付けられます。

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。

add_range_partition_clause

新しいレンジ・パーティション *partition* を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）に追加する場合は、*add_range_partition_clause* を使用します。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。

64KB - 1 パーティションまで指定できます。

> **参照：** この数より小さい値に制限される場合の原因については、『Oracle8*i* 管理者ガイド』を参照してください。

制限事項：

- 上位のパーティションのパーティション境界の第 1 要素が MAXVALUE である場合、パーティションは表に追加できません。そのかわり、*split_partition_clause* を使用してパーティションを表の始めまたは中間に追加してください。
- *compression_clause*、*physical_attributes_clause* 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。
- 索引構成表の索引セグメントに対しては、PCTUSED パラメータを指定できません。
- 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限り、OVERFLOW を指定できます。
- 表レベルで圧縮が使用可能な場合に限り、圧縮を指定できます。

VALUES LESS THAN (*value_list*)	新しいパーティションの上限を指定します。*value_list* は、*column_list* 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。
partition_level_subpartitioning	*partition_level_subpartitioning* 句は、コンポジット・パーティション句でのみ使用できます。この句を使用して、*partition* に対して特定のハッシュ・サブパーティションを指定できます。コンポジット・パーティションは、次のいずれかの方法で指定できます。

- 個々のパーティションを名前で指定できます。また、各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。
- サブパーティションの数を指定できます（または、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_SUBP*nnn* という形式のパーティション名を割り当てられます。表領域の数は、サブパーティションの数と等しくなくてもかまいません。サブパーティションの数が表領域の数より多い場合、表領域名は繰り返されます。

サブパーティションは、*new_partition* に指定したサブパーティション・レベルで指定できる TABLESPACE 以外のすべての属性を継承します。サブパーティションまたはパーティション・レベルで指定されなかった属性は、表レベルのデフォルト値から継承されます。

この句によって、表レベルで指定されたすべてのサブパーティションはオーバーライドされます。

この句を指定せずに表レベルでデフォルトのサブパーティションを指定した場合、*new_partition_name* は、表レベルのデフォルトのサブパーティションを継承します。

参照 : 10-7 ページの「CREATE TABLE」を参照してください。

add_hash_partition_clause

パーティション表の 1 番上に新しいハッシュ・パーティションを追加する場合は、*add_hash_partition_clause* を使用します。Oracle は、ハッシュ関数によって *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サブパーティションに入れます。

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。*new_partition_name* を指定しなかった場合、SYS_P*nnn* 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。

参照 : ハッシュ・パーティション化については、10-7 ページの「CREATE TABLE」および『Oracle8*i* 概要』を参照してください。

parallel_clause 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。

coalesce_partition_clause

COALESCE は、ハッシュ・パーティション表にのみ適用されます。この句は、ハッシュ・パーティションが選択され、その内容が 1 つ以上の残りのパーティション（ハッシュ関数が決定する）に分配された後、選択されたパーティションが削除されるように指定します。選択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も削除されます。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。この索引パーティションは、再構築する必要があります。

drop_partition_clause

drop_partition_clause は、レンジ・メソッドまたはコンポジット・メソッドを使用してパーティション化された表にのみ適用されます。この句は、パーティション表から、パーティション *partition* およびそのパーティション内のデータを削除します。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：** 8-50 ページの「merge_partitions_clause」を参照してください。

表に LOB 列がある場合、*partition* に対応する LOB データ、および LOB 索引パーティション（および存在する場合にはサブパーティション）が削除されます。

- *partition* に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は、UNUSABLE のマークが付けられている場合でも削除されます。
- パーティションが削除されていないか、またはすべてのサブパーティションが空でない限り、表に定義されたすべてのグローバルな非パーティション索引およびグローバルなパーティション索引のパーティションに、UNUSABLE のマークが付けられます。
- 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、行は 1 つ後のパーティションに格納されます。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。

制限事項： *table* にパーティションが 1 つのみの場合は、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。

truncate_partition_clause および *truncate_subpartition_clause*

TRUNCATE PARTITION は、*partition* からすべての行を削除します。表がコンポジット・パーティションの場合は、*partition* のサブパーティションからすべての行を削除します。TRUNCATE SUBPARTITION は、*subpartition* からすべての行を削除します。

表に LOB 列がある場合、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。表がコンポジット・パーティションである場合、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。

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。

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。さらに、切り捨てられるパーティション、サブパーティション、または切り捨てられるパーティションのすべてのサブパーティションが空でない場合、表に定義されたすべてのグローバルな非パーティション索引およびグローバル索引のパーティションに、UNUSABLE のマークが付けられます。

DROP STORAGE　削除した行が占有していた領域の割当てを解除し、表領域内の別のスキーマ・オブジェクトがその領域を使用できるように、DROP STORAGE を指定します。

REUSE STORAGE　パーティションまたはサブパーティションに割り当てられた、削除された行から領域を確保します。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。

split_partition_clause

split_partition_clause によって、元のパーティション *partition_name_old* から 2 つの新しいパーティションを作成し、それぞれのパーティションのセグメント属性、物理属性および初期エクステントを新しく指定します。*partition_name_old* に対応付けられていたセグメントは破棄されます。

制限事項 : この句をハッシュ・パーティション表に指定することはできません。

AT (*value_list*)　*split_partition_1* の上限（境界は含まない）を指定します。*value_list* の値は、*partition_name_old* の元のパーティション境界より小さく、1 つ前のパーティション（ただし、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くする必要があります。

INTO *partition_spec*、*partition_spec*　INTO 句によって、分割の結果生成された 2 つのパーティションを記述します。キーワード PARTITION が必要です。分割の結果生成された 2 つのパーティションの任意の名前と物理属性を指定します。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、*partition_name_old* から継承されます。

制限事項：

- パーティション化された索引構成表に対してのみ、*compression_clause*、*physical_attributes_clause* および OVERFLOW を指定できます。
- 索引構成表の索引セグメントに対しては、PCTUSED パラメータを指定できません。

parallel_clause 分割操作をパラレル化しても、表のデフォルトのパラレル属性を変更しない場合は、*parallel_clause* を使用します。

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、サブパーティションには TABLESPACE のみ指定できます。他のすべての属性は、このサブパーティションが含まれている新しいパーティションから継承されます。

partition_name_old がサブパーティション化されており、新しいパーティションにサブパーティションを指定していない場合、新しいパーティションは、*partition_name_old* のサブパーティションの数および表領域を継承します。

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。その結果得られるローカル索引パーティションは、分割されるローカル索引パーティションからパーティション・レベルのすべてのデフォルト属性を継承します。

partition_name_old が空でない場合、表にあるすべてのグローバルな非パーティション索引およびグローバル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（グローバル索引に対する処理は、索引構成表に適用されません）。さらに、分割の結果生成されたすべてのパーティションまたはサブパーティションが空でない場合、対応するすべての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に、UNUSABLE のマークが付けられます。

表に 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 記憶属性を指定できます。*partition_name_old* の LOB データおよび LOB 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、それぞれの LOB 列およびパーティションに対して新しいセグメントが作成されます。

merge_partitions_clause

表の隣接する 2 つの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た後、元の 2 つのパーティションを削除する場合は、*merge_partitions_clause* を使用します。

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。

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。

新しい *partition_name* を指定しなかった場合、SYS_P*nnn* という形式でパーティション名が割り当てられます。新しいパーティションにサブパーティションがある場合、サブパーティションには、SYS_SUBP*nnn* という形式で名前が割り当てられます。

元のパーティションのいずれか、または両方が空でない場合、表にあるすべてのグローバルな非パーティション索引およびグローバル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。さらに、結合の結果生成されたパーティションまたはサブパーティションが空でない場合、すべての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に UNUSABLE のマークが付けられます。

制限事項 : この句は、索引構成表、またはハッシュ・メソッドを使用してパーティション化された表には指定できません。

partition_level_subpartitioning
: 新しいパーティションのハッシュ・サブパーティション属性を指定します。この句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。

  この句を指定しない場合、マージされた新しいパーティションは、表レベルのデフォルトからサブパーティション属性を継承します。

parallel_clause
: マージ操作をパラレル化します。

exchange_partition_clause および *exchange_subpartition_clause*

EXCHANGE PARTITION 句または EXCHANGE SUBPARTITION 句を使用して、次のデータおよび索引セグメントを交換します。

- ハッシュ・パーティションまたはレンジ・パーティション（またはサブパーティション）を持つ非パーティション表
- コンポジット・パーティション表のレンジ・パーティションのハッシュ・サブパーティションを持つハッシュ・パーティション表

2 つのオブジェクト（表領域を含む）のすべてのセグメント属性も交換されます。

デフォルトの動作は、EXCLUDING INDEXES WITH VALIDATION です。この操作を実行する場合、両方の表に対する ALTER TABLE システム権限が必要です。

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、『Oracle8*i* 管理者ガイド』を参照してください。

表に LOB 列がある場合、Oracle は、各 LOB 列に対して、LOB データ、および LOB 索引パーティションまたはサブパーティション・セグメントを、*table* の対応する LOB データおよび LOB 索引セグメントと交換します。

表およびパーティションのすべての統計情報（表、列、索引統計およびヒストグラムを含む）が交換されます。新しいパーティション表を受け取る表の集計統計は再計算されます。

表とパーティションのロギング属性も交換されます。

制限事項 : 交換される両方の表は同じ主キーを持つ必要があり、参照表が空でない限り、どちらの表もどのような有効な外部キーからも参照できません。

WITH TABLE *table*
: パーティションを交換する表を指定します。

INCLUDING INDEXES
: 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（非パーティション表の）対応する表索引または（ハッシュ・パーティション化表の）対応するローカル索引と交換する場合は、INCLUDING INDEXES を指定します。

EXCLUDING INDEXES
: 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 のマークを付ける場合に、EXCLUDING INDEXES を指定します。

WITH VALIDATION
: WITH VALIDATION を指定すると、交換された表にあるすべて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
: 交換された表にある行が正しくマップされたかどうかのチェックが必要ない場合は、WITHOUT VALIDATION を指定します。

EXCEPTIONS INTO
: 制約に違反するすべての列の ROWID を格納する表を指定します。*schema* を指定しない場合、自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。例外表は、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、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。

注意 : 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々の例外表を作成し、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。

参照 :

- SQL スクリプトの詳細は、『Oracle8*i* PL/SQL パッケージ・プロシージャ 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、『Oracle8*i* パフォーマンスのための設計およびチューニング』を参照してください。
- これらのスクリプトの使用に関する互換性については、『Oracle8*i* 移行ガイド』を参照してください。

EXCEPTIONS INTO 句の制限事項 :

- この句はサブパーティションでは無効です。
- パーティション表は一意制約で定義する必要があり、その制約は DISABLE VALIDATE の状態である必要があります。

これらの条件が当てはまらない場合、この句は無視されます。

参照 : 制約チェックの詳細は、8-134 ページの「constraint_clause」を参照してください。

パーティション交換の制限事項 :

ハッシュ・パーティション表とコンポジット・パーティション表のレンジ・パーティションの交換時には、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化キーは、コンポジット・パーティション表のサブパーティション化キーと同じである必要があります。
- ハッシュ・パーティション表のパーティション数は、コンポジット・パーティション表のレンジ・パーティションのサブパーティション数と同じである必要があります。
- 両方の表にあるすべてのグローバル索引に、UNUSABLE のマークが付けられます。

パーティション化された索引構成表に対して、次の追加の制限事項があります。

- ソースおよびターゲットの表 / 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。
- 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。
- 索引構成表パーティションは標準表と交換できません。また、その逆もできません。
- 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。

row_movement_clause

row_movement_clause によって、1 つ以上のキー値の変更によって、行が別のパーティションまたはサブパーティションに移動できるかどうかを指定します。

制限事項 : パーティション表に対してのみこの句を指定できます。

ENABLE　ENABLE を指定すると、パーティション化またはサブパーティション化キーを更新した結果、行が異なるパーティションやサブパーティションに移動されます。

制限事項 : ドメイン・インデックスが表のいずれかの列に作成されている場合、この句は指定できません。

注意 : UPDATE 操作中に行を移動すると、その行の ROWID が変更されます。

DISABLE　DISABLE を指定すると、パーティション化またはサブパーティション化キーを更新した結果、異なるパーティションまたはサブパーティションに行が移動され、エラーが戻ります。これはデフォルト値です。

parallel_clause

parallel_clause によって、表の問合せおよび DML に対してデフォルトの並列度を変更します。

注意： *parallel_clause* 構文は、以前のリリースの構文に代わるものです。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	シリアル実行を行う場合は、NOPARALLEL を指定します。これはデフォルト値です。
PARALLEL	すべてのパーティション化インスタンスで使用可能な CPU の数に、PARALLEL_THREADS_PER_CPU 初期化パラメータの値を掛けた並列度を選択させる場合は、PARALLEL を指定します。
PARALLEL *integer*	パラレル操作で使用されるパラレル・スレッド数である**並列度**を指定する場合は、*integer* 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。

制限事項：

- 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、この表での 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： CREATE TABLE については、10-41 ページの「parallel_clause に関する注意事項」を参照してください。

enable_disable_clause

enable_disable_clause によって、整合性制約が適用されるかどうかを指定できます。

参照： この句の詳細（この文に関連する注意および制限事項を含む）は、10-41 ページの「CREATE TABLE」にある *enable_disable_clause* 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 を指定して表ロックを有効にすることによって、表に対する DDL 操作が可能になります。
DISABLE TABLE LOCK	DISABLE TABLE LOCK を指定して表ロックを無効にすることによって、表に対する DML 操作が不可能になります。

ALL TRIGGERS

ENABLE ALL TRIGGERS	ENABLE ALL TRIGGERS を指定して、表に関連するすべてのトリガーを使用可能にします。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。10-66 ページの「CREATE TRIGGER」を参照してください。 単一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：** 8-76 ページの「ALTER TRIGGER」を参照してください。
DISABLE ALL TRIGGERS	DISABLE ALL TRIGGERS を指定して、表に関連するすべてのトリガーを無効にします。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

ネストした表の例　次の文は、emp 表に問い合せたときに、ロケータのかわりに実値を戻すように、emp 表のネストした表の列 projects の記憶特性を変更します。

```
ALTER TABLE emp MODIFY NESTED TABLE projects RETURN AS VALUE;
```

PARALLEL の例　次の文は、emp 表への問合せに対してパラレル処理を指定します。

```
ALTER TABLE emp
   PARALLEL;
```

ENABLE VALIDATE の例　次の文は、emp 表の fk_deptno という名前の整合性制約を ENABLE VALIDATE 状態にします。

```
ALTER TABLE emp
   ENABLE VALIDATE CONSTRAINT fk_deptno
   EXCEPTIONS INTO except_table;
```

Oracle が制約を使用可能にするためには、emp 表のそれぞれの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、except_table 表の中に表示されます。次の文によって、emp 表の例外を検出することもできます。

```
SELECT emp.*
   FROM emp e, except_table ex
   WHERE e.row_id = ex.row_id
      AND ex.table_name = 'EMP'
      AND ex.constraint = 'FK_DEPTNO';
```

ENABLE NOVALIDATE の例　次の文は、emp 表の 2 つの制約を ENABLE NOVALIDATE 状態にします。

```
ALTER TABLE emp
   ENABLE NOVALIDATE UNIQUE (ename)
   ENABLE NOVALIDATE CONSTRAINT nn_ename;
```

この文には、次の 2 つの ENABLE 句が含まれています。

- 最初の ENABLE 句は、ename 列の一意制約を ENABLE NOVALIDATE 状態にします。
- 2 番目の ENABLE 句は、nn_ename 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限り、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。

制約を使用禁止にする例　phone_calls 表の areaco 列および phoneno 列の組合せに対する外部キーを使用した参照整合性制約があるとします。外部キーは customers 表の areaco 列および phoneno 列にある一意キーを参照します。次の文は、customers 表の areaco 列および phoneno 列にある一意キーを使用禁止にします。

```
ALTER TABLE customers
   DISABLE UNIQUE (areaco, phoneno) CASCADE;
```

customers 表の一意キーは、phone_calls 表の外部キーによって参照されるため、この一意キーを使用禁止にする場合は、CASCADE 句を使用します。この句によって、外部キーも使用禁止になります。

CHECK 制約の例　次の文は、emp 表に CHECK 制約を定義し、その制約を使用禁止にします。

```
ALTER TABLE emp
   ADD (CONSTRAINT check_comp CHECK (sal + comm <= 5000) )
   DISABLE CONSTRAINT check_comp;
```

check_comp 制約は、給与総額が $5000 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、emp 表に関連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 emp
   ENABLE ALL TRIGGERS;
```

DEALLOCATE UNUSED の例　次の文は、emp 表で再使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 emp
    DEALLOCATE UNUSED;
```

DROP COLUMN の例　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。

```
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 およびチェック制約 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);
```

索引構成表の例　次の文は、索引構成表 docindex の索引セグメントの INITRANS パラメータを変更します。

```
ALTER TABLE docindex INITRANS 4;
```

次の文では、オーバーフロー・データ・セグメントを索引構成表 docindex に追加します。

```
ALTER TABLE docindex ADD OVERFLOW;
```

次の文は、索引構成表 docindex 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 docindex OVERFLOW INITRANS 4;
```

ADD PARTITION の例　次の文は、パーティション p3 を追加し、表の 3 つの LOB 列（b、c および d）の記憶特性を指定します。

```
ALTER TABLE pt ADD PARTITION p3 VALUES LESS THAN (30)
   LOB (b, d) STORE AS (TABLESPACE tsz)
   LOB (c) STORE AS mylobseg;
```

パーティション P3 の列 b および列 d の LOB データおよび LOB 索引セグメントは、表領域 tsz 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 c の LOB データ・セグメントは mylobseg セグメントに格納され、他のすべての属性は、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。

SPLIT PARTITION の例　次の文は、パーティション p3 をパーティション p3_1 および p3_2 に分割します。

```
ALTER TABLE pt SPLIT PARTITION p3 AT (25)
   INTO (PARTITION p3_1 TABLESPACE ts4
            LOB (b,d) STORE AS (TABLESPACE tsz),
         PARTITION p3_2 (TABLESPACE ts5)
            LOB (c) STORE AS (TABLESPACE ts5);
```

パーティション p3_1 では、列 b と列 d の LOB セグメントが表領域 tsz 内に作成されます。パーティション p3_2 では、列 c の LOB セグメントが表領域 ts5 内に作成されます。パーティション p3_2 内の列 b と列 d の LOB セグメント、およびパーティション p3_1 内の列 c の LOB セグメントは、元のパーティション p3 の元の表領域内に残ります。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。

ユーザー定義オブジェクト識別子の例　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;
```

列の追加例　次の文は、最大 7 桁、小数点以下 2 桁の NUMBER データ型の thriftplan という名前の列と、サイズが 1 で NOT NULL 整合性制約がある CHAR データ型の loancode という名前の列を追加します。

```
ALTER TABLE emp
   ADD (thriftplan NUMBER(7,2),
        loancode CHAR(1) NOT NULL);
```

列の変更例　次の文は、thriftplan 列のサイズを 9 桁に変更します。

```
ALTER TABLE emp
   MODIFY (thriftplan NUMBER(9,2));
```

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、emp 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 emp
    PCTFREE 30
    PCTUSED 60;
```

ALLOCATE EXTENT の例　次の文は、emp 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 emp
  ALLOCATE EXTENT (SIZE 5K INSTANCE 4);
```

この文は、DATAFILE パラメータを指定していないため、エクステントは emp 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。

デフォルト列値の例　次の文は、accounts 表の bal 列のデフォルト値が 0 になるように変更します。

```
ALTER TABLE accounts
  MODIFY (bal DEFAULT 0);
```

この後で、accounts 表に新しい行を追加した場合、bal 列に値を指定しなければ、bal 列の値は自動的に 0（ゼロ）になります。

```
INSERT INTO accounts(accno, accname)
  VALUES (accseq.nextval, 'LEWIS');

SELECT *
    FROM accounts
    WHERE accname = 'LEWIS';

ACCNO  ACCNAME BAL
------ ------- ---
815234  LEWIS     0
```

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、次の文に示すとおり、デフォルト値を NULL に置き換えます。

```
ALTER TABLE accounts
   MODIFY (bal 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。

制約の削除例　次の文は、dept 表の主キーを削除します。

```
ALTER TABLE dept
    DROP PRIMARY KEY CASCADE;
```

PRIMARY KEY 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 dept
    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、dept 表の dname 列の一意キーを削除します。

```
ALTER TABLE dept
    DROP UNIQUE (dname);
```

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 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 employee ADD (resume CLOB)
  LOB (resume) STORE AS resume_seg (TABLESPACE resume_ts);
```

次の文を入力して、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 employee MODIFY LOB (resume) (CACHE);
```

ネストした表の例　次の文は、ネストした表の列 skills を employee 表に追加します。

```
ALTER TABLE employee 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_storage_clause* に指定した記憶域表の名前を使用してください。記憶域表では、DML 文の問合せまたは実行はできません。記憶域表は、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 と記憶域表 client_tab を使用して、表 vetservice を作成します。ネストした表 vetservice を変更して制約を指定します。

```
CREATE TYPE pet_table AS OBJECT
   (pet_name VARCHAR2(10), pet_dob DATE);

CREATE TABLE vetservice (vet_name VARCHAR2(30),
                         client   pet_table)
  NESTED TABLE client STORE AS client_tab;

ALTER TABLE client_tab ADD UNIQUE (ssn);
```

次の文は、ネストした表 nested_skill_table に、一意制約を追加します。

```
ALTER TABLE nested_skill_table ADD UNIQUE (a);
```

次の文は、REF 値のネストした表用の記憶域表を変更して、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、記憶域表 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（REF）表として定義されるため、Oracle は、暗黙的に列名 COLUMN_VALUE を記憶域表に設定します。

参照：

- ネストした表の記憶領域の詳細は、10-7 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、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

REF の例　次の文は、オブジェクト型 dept_t があらかじめ定義されています。次のように emp 表を作成します。

```
CREATE TABLE emp
   (name VARCHAR(100),
    salary NUMBER,
    dept REF dept_t);
```

オブジェクト表 DEPARTMENTS を次のように作成します。

```
CREATE TABLE departments 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 emp
    ADD (SCOPE FOR (dept) IS departments);
```

前述の ALTER TABLE 文は、emp 表が空である場合のみ正常に実行されます。

emp の dept 列に REF 値を格納する場合に ROWID も一緒に格納する場合は、次の文を発行します。

```
ALTER TABLE emp
   ADD (REF(dept) WITH ROWID);
```

パーティションの追加例　次の文は、パーティション jan99 を表領域 tsx に追加します。

```
ALTER TABLE sales
  ADD PARTITION jan99 VALUES LESS THAN( '970201' )
  TABLESPACE tsx;
```

パーティションの削除例　次の文は、パーティション dec98 を削除します。

```
ALTER TABLE sales DROP PARTITION dec98;
```

パーティションの交換例　次の文は、パーティション feb97 を表 sales_feb97 に変換します。ローカル索引パーティションと sales_feb97 に対応する索引との交換、および sales_feb97 内のデータがパーティション feb97 の範囲内かどうかの検証は行われません。

```
ALTER TABLE sales
   EXCHANGE PARTITION feb97 WITH TABLE sales_feb97
   WITHOUT VALIDATION;
```

パーティションの変更例　次の文は、sales 表のパーティション nov96 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 sales MODIFY PARTITION nov96
   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 sales MODIFY PARTITION jan97
   REBUILD UNUSABLE LOCAL INDEXES;
```

次の文は、パーティション branch_ny の MAXEXTENTS およびロギング属性を変更します。

```
ALTER TABLE branch MODIFY PARTITION branch_ny
   STORAGE (MAXEXTENTS 75) LOGGING;
```

パーティションの移動例　次の文は、パーティション depot2 を表領域 ts094 に移動します。

```
ALTER TABLE parts
  MOVE PARTITION depot2 TABLESPACE ts094 NOLOGGING;
```

パーティションの改名例　次の文は、表を改名します。

```
ALTER TABLE emp RENAME TO employee;
```

次の文では、パーティション emp3 が改名されます。

```
ALTER TABLE employee RENAME PARTITION emp3 TO employee3;
```

パーティションの分割例　次の文は、古いパーティション depot4 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 depot9 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使用します。

```
ALTER TABLE parts
   SPLIT PARTITION depot4 AT ( '40-001' )
   INTO ( PARTITION depot4 TABLESPACE ts009 STORAGE (MINEXTENTS 2),
         PARTITION depot9 TABLESPACE ts010 )
   PARALLEL (10);
```

パーティションの切捨て例　次の文は、パーティション sys_p017 内のすべてのデータを削除し、解放された領域の割当てを解除します。

```
ALTER TABLE deliveries
   TRUNCATE PARTITION sys_p017 DROP STORAGE;
```

追加の例　ALTER TABLE 文を使用した整合性制約の定義の例は、8-134 ページの「constraint_clause」を参照してください。

表の記憶領域パラメータの値を変更する例は、11-129 ページの「storage_clause」を参照してください。

ALTER TABLESPACE

用途

ALTER TABLESPACE は、既存の表領域、1 つ以上のデータ・ファイルまたはテンポラリ・ファイルを変更する場合に使用します。

> **参照：** 表領域の作成については、10-56 ページの「CREATE TABLESPACE」を参照してください。

前提条件

ALTER TABLESPACE システム権限を持っている場合、この文のすべての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。

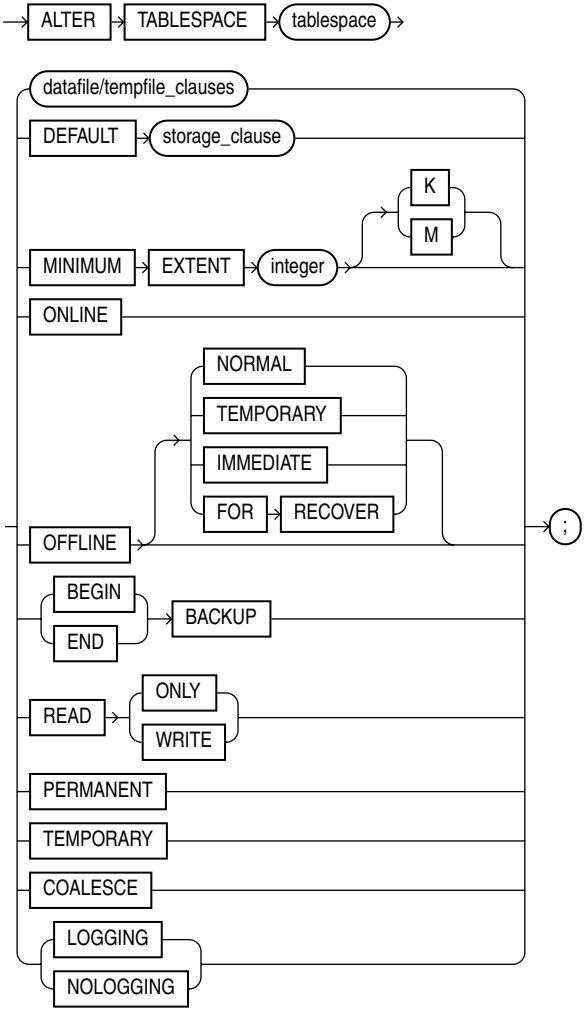
- 表領域をオンラインまたはオフラインにする。
- バックアップを開始または終了する。
- 表領域を読取り専用または読み書き両用にする。

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。

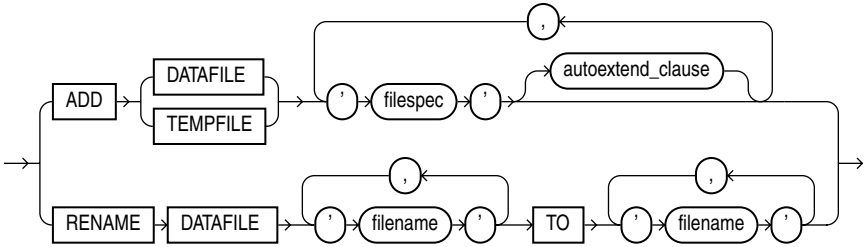
- 表領域がオンラインになっている。
- 表領域にアクティブなロールバック・セグメントがない。SYSTEM 表領域には SYSTEM ロールバック・セグメントがあるため、読取り専用にはできません。また、読取り専用表領域のロールバック・セグメントにはアクセスできないため、ロールバック・セグメントを削除してから、表領域を読取り専用にすることをお薦めします。
- 表領域がオープン・バックアップに使用されていない。バックアップの終わりに表領域内のすべてのデータ・ファイルのヘッダー・ファイルが更新されるためです。

これらの条件を満たす場合、制限モードでこの機能を実行すると便利です。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

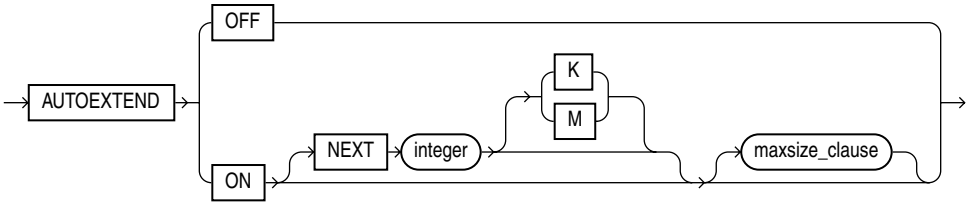

ALTER
TABLESPACE
tablespace
datafile/tempfile_clauses
DEFAULT
storage_clause
MINIMUM
EXTENT
integer
K
M
ONLINE
OFFLINE
NORMAL
TEMPORARY
IMMEDIATE
FOR
RECOVER
;
BEGIN
END
BACKUP
READ
ONLY
WRITE
PERMANENT
TEMPORARY
COALESCE
LOGGING
NOLOGGING

datafile / tempfile_clauses::=

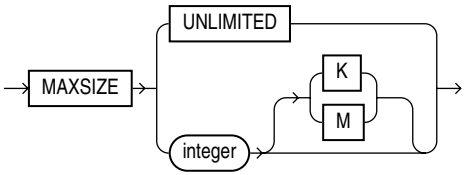


filespec: 11-27 ページの「filespec」を参照してください。

autoextend_clause::=



maxsize_clause::=



storage_clause: 11-129 ページの「storage_clause」を参照してください。

キーワードとパラメータ

tablespace

変更する表領域の名前を指定します。

注意： この句は、ローカル管理の一時表領域に対して、ADD 句のこの文で指定できる唯一の句です。

---

datafile / tempfile_clauses

datafile 句および *tempfile* 句によって、データ・ファイルまたはテンポラリ・ファイルの追加および削除をします。

ADD DATAFILE | TEMPFILE

filespec によって指定されたデータ・ファイルまたはテンポラリ・ファイルを追加する場合は、ADD を指定します。

データ・ファイルまたはテンポラリ・ファイルを、オンラインのローカル管理表領域、あるい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。

参照： 11-27 ページの「filespec」を参照してください。

注意： 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。

RENAME DATAFILE

1 つ以上の表領域のデータ・ファイルを改名する場合は、RENAME DATAFILE で指定します。表領域をオフラインにしてからデータ・ファイルを改名します。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。

autoextend_clause

autoextend_clause によって、表領域にあるデータ・ファイルのサイズの自動拡張を可能または使用禁止にします。

OFF	OFF を指定すると、自動拡張を使用禁止にします。NEXT および MAXSIZE は 0（ゼロ）に設定されます。NEXT および MAXSIZE の値は、後続の ALTER TABLESPACE AUTOEXTEND 文で再指定する必要があります。
ON	ON を指定すると、自動拡張を使用可能にします。
NEXT *integer*	追加のエクステントが必要になった場合、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のサイズ（バイト単位）を指定します。K または M を使用すると、KB または MB 単位で指定することもできます。デフォルト値は 1 データ・ブロックです。
maxsize_clause	*maxsize_clause* には、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します。 UNLIMITED　データ・ファイルへのディスク領域割当てを無制限にする場合は、UNLIMITED を指定します。

DEFAULT *storage_clause*

DEFAULT *storage_clause* には、表領域に作成される後続のオブジェクトに対する新しいデフォルトの記憶領域パラメータを指定します。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。

制限事項 : この句は、ローカル管理表領域に指定できません。

参照 : 11-129 ページの「storage_clause」を参照してください。

MINIMUM EXTENT

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（あるいはその両方）の大きさが、*integer* で指定したサイズ以上であり、かつその倍数になるように MINIMUM EXTENT を指定することによって、表領域における空き領域の断片化を制御します。この句は、ディクショナリ管理された一時表領域には関連がありません。

制限事項 : この句は、ローカル管理表領域に指定できません。

参照 : MINIMUM EXTENT を使用した断片化の制御については、『Oracle8*i* 管理者ガイド』を参照してください。

ONLINE | OFFLINE

ONLINE を指定して、表領域をオンラインにします。

OFFLINE を指定して、表領域をオフラインにし、そのセグメントにアクセスできないようにします。

提案： 表領域を長期間オフラインにするには、デフォルト表領域または一時表領域としてその表領域を割り当てられているユーザーの表領域割当てを変更した方がよい場合があります。表領域をオフラインにした場合、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。

参照： 8-87 ページの「ALTER USER」を参照してください。

BEGIN BACKUP

BEGIN BACKUP は、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します。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。

制限事項： 読取り専用の表領域またはテンポラリ・ローカル管理表領域に対して、この句を指定することはできません。

注意： バックアップ中は、表領域の標準オフラインへの切替え、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。

END BACKUP

END BACKUP は、表領域のオープン・バックアップが完了したことを示します。オープン・バックアップの終了後、できるだけ早くこの句を指定してください。この句は、読取り専用表領域に対しては使用できません。

オンラインの表領域バックアップの終了を指定せずに、インスタンス障害または SHUTDOWN ABORT が発生した場合は、インスタンスを次に開始するときにメディア・リカバリ（アーカイブ REDO ログ・ファイルを使用する場合もあります）が必要です。

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、『Oracle8*i* 管理者ガイド』を参照してください。

READ ONLY | READ WRITE

READ ONLY によって、表領域を**読取専用推移モード**に設定します。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、以前の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のロールバック以外は、その表領域に対してさらに書込み操作（DML）を行うことはできません。

表領域を読取り専用にした場合、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 ... 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のデータ・ファイルを改名する必要があります。

参照：

- 読取り専用の表領域の詳細は、『Oracle8*i* 概要』を参照してください。
- 7-7 ページの「ALTER DATABASE」を参照してください。

読取専用に指定されている表領域に対して書き込み操作を可能にする場合は、READ WRITE を指定します。

PERMANENT | TEMPORARY

一時表領域を永続表領域に変換する場合は、PERMANENT を指定します。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値です。

永続表領域を一時表領域に変換する場合は、TEMPORARY を指定します。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。

COALESCE

この句は、表領域内の各データ・ファイルについて、連続する未使用エクステントをすべて結合して大きい連続エクステントにします。

LOGGING | NOLOGGING

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する場合、LOGGING を指定します。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オーバーライドできます。

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（これは後でオーバーライドもできます）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。

次の操作のみが、NOLOGGING モードをサポートします。

- DML: ダイレクト・ロード INSERT（シリアルまたはパラレル）、ダイレクト・ローダー（SQL*Loader）
- DDL: CREATE TABLE ... AS SELECT、CREATE INDEX、ALTER INDEX ... REBUILD、ALTER INDEX ... REBUILD PARTITION、ALTER INDEX ... SPLIT PARTITION、ALTER TABLE ... SPLIT PARTITION および ALTER TABLE ... MOVE PARTITION

NOLOGGING モードでは、データの変更時に、（新しいエクステントに INVALID のマークを設定し、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。

例

バックアップの例　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 accounting
    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 accounting
    END BACKUP;
```

移動および改名例　次の例は、accounting 表領域に対応付けられたデータ・ファイル 'diska:pay1:dat' を移動して 'diskb:receive1:dat' に改名します。

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 accounting 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 'diska:pay1.dat' から 'diskb:receive1.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・ファイルを改名します。

   ```
   ALTER TABLESPACE accounting
     RENAME DATAFILE 'diska:pay1.dbf'
     TO     'diskb:receive1.dbf';
   ```

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 accounting ONLINE;
```

データ・ファイルの追加例　次の文は、データ・ファイルを表領域に追加し、デフォルトのロギング属性を NOLOGGING に変更します。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 accounting NOLOGGING
    ADD DATAFILE 'disk3:pay3.dbf'
    SIZE 50K
    AUTOEXTEND ON
    NEXT 10K
    MAXSIZE 100K;
```

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オーバーライドできます。

エクステントの割当ての変更例　次の文は、tabspace_st の各エクステントの割当てを 128KB の倍数に変更します。

```
ALTER TABLESPACE tabspace_st MINIMUM EXTENT 128K;
```

ALTER TRIGGER

用途

ALTER TRIGGER は、データベース・トリガーを使用可能、使用禁止またはコンパイルする場合に使用します。

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 を指定した CREATE TRIGGER 文を使用してください。

参照：

- トリガーの作成については、10-66 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、11-13 ページの「DROP TRIGGER」を参照してください。

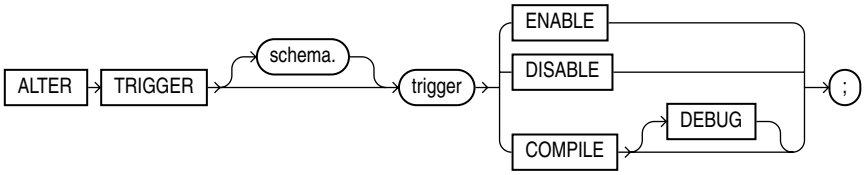
前提条件

トリガーが自スキーマ内にある必要があります。自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。

さらに、DATABASE 上のトリガーを変更する場合は、ADMINISTER DATABASE TRIGGER システム権限が必要です。

参照： DATABASE に基づいたトリガーの詳細は、10-66 ページの「CREATE TRIGGER」を参照してください。

構文



キーワードとパラメータ

schema

削除する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、トリガーは自スキーマ内に定義されているとみなされます。

trigger

変更するトリガーの名前を指定します。

ENABLE

ENABLE を指定して、トリガーを使用可能にします。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。

参照： 8-2 ページの「ALTER TABLE」を参照してください。

DISABLE

DISABLE を指定して、トリガーを使用禁止にします。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。

参照： 8-2 ページの「ALTER TABLE」を参照してください。

COMPILE

トリガーが使用可能または使用禁止であるかにかかわらず、トリガーを明示的にコンパイルするには、COMPILE を指定します。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。

トリガーが依存するオブジェクトに無効なオブジェクトがある場合は、最初にそのオブジェクトが再コンパイルされます。トリガーの再コンパイルが正常に終了した場合、このトリガーは使用可能になります。

トリガーの再コンパイル時にエラーが発生した場合、エラーが戻り、そのトリガーは使用禁止のままです。SQL*Plus コマンド SHOW ERRORS を使用して、関連するコンパイラ・エラー・メッセージを表示できます。

DEBUG　　DEBUG を指定して、PL/SQL コンパイラに対して、PL/SQL デバッガ用のコードを生成および保存するように指示します。この句は、標準トリガーおよび代替トリガーに使用できます。

参照 :

- これらのプロシージャについては、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。
- リモート・オブジェクトなどのスキーマ・オブジェクト間の依存性を Oracle が管理する方法の詳細は、『Oracle8*i* 概要』を参照してください。

例

トリガーを使用禁止にする例　inventory 表に作成された reorder という名前のトリガーがあるとします。UPDATE 文によって、ある部品の在庫数がその再発注点を下回るたびにこのトリガーが起動されます。このトリガーによって部品番号、再発注の数量および現在の日付を含む行が保留中の発注表に挿入されます。

このトリガーは、作成時に自動的に使用可能になります。その後、次の文を指定して使用禁止にできます。

```
ALTER TRIGGER reorder DISABLE;
```

このトリガーを使用禁止にすると、UPDATE 文によって部品の在庫数がその再発注点を下回ってもトリガーは起動されません。

トリガー使用可能の例　トリガーを使用禁止にした後、次の文を使用してそのトリガーを再び使用可能にできます。

```
ALTER TRIGGER reorder ENABLE;
```

再びトリガーを使用可能にした場合、UPDATE 文によって部品の在庫数がその再発注点を下回るたびにトリガーが起動されます。トリガーが使用禁止になっている間に、部品の在庫数がその再発注点を下回っている可能性があります。この場合、再びトリガーを使用可能にしても、別のトランザクションによって在庫数が減らない限り、その部品に対して、このトリガーが自動的に起動されることはありません。

ALTER TYPE

用途

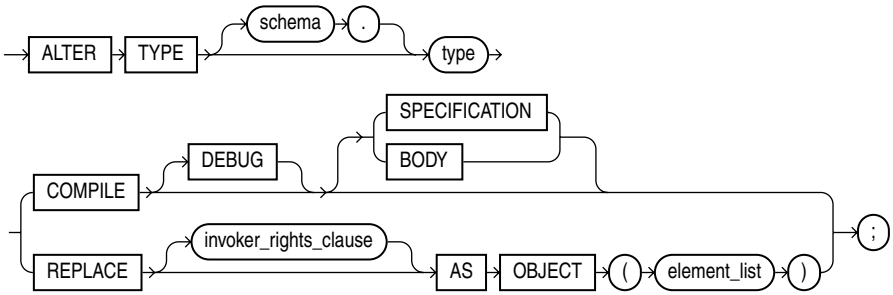
ALTER TYPE は、新しいオブジェクト・メンバーのサブプログラム仕様を追加することによって、オブジェクト型の仕様部または本体（あるいはその両方）を再コンパイルまたはオブジェクトの仕様を変更する場合に使用します。

オブジェクト型の既存のプロパティ（属性、メンバー・サブプログラム、MAP ファンクションまたは順序関数）は変更できませんが、新しいメンバー・サブプログラム仕様は追加できます。

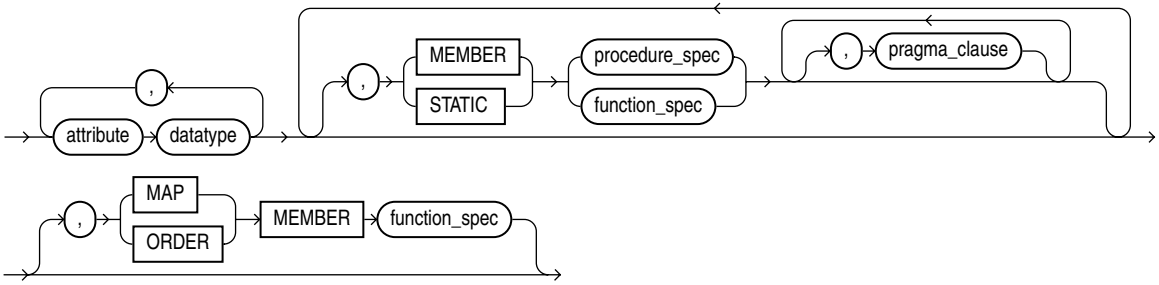
前提条件

オブジェクト型が自スキーマ内にあり、CREATE TYPE または CREATE ANY TYPE システム権限が必要です。または、ALTER ANY TYPE システム権限が必要です。

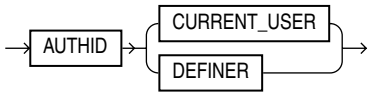
構文



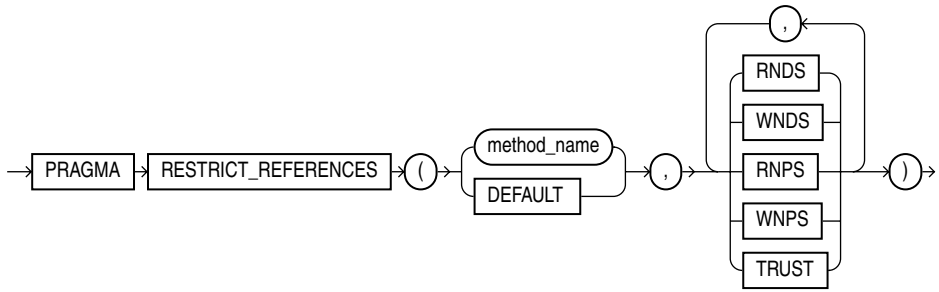
element_list::=



invoker_rights_clause::=



pragma_clause::=



キーワードとパラメータ

schema

型を定義するスキーマを指定します。*schema* を指定しない場合、この型が現行のスキーマに存在するものとみなされます。

type

オブジェクト型、ネストした表型または ROWID 型の名前を指定します。

COMPILE

COMPILE を指定して、オブジェクト型の仕様部および本体をコンパイルします。SPECIFICATION または BODY のいずれも指定していない場合、これはデフォルトです。

型の再コンパイル時にエラーが発生した場合、エラーが戻り、その型は無効のままです。SQL*Plus コマンド SHOW ERRORS を使用して、関連するコンパイラ・エラー・メッセージを表示できます。

DEBUG	DEBUG を指定して、PL/SQL コンパイラに対して、PL/SQL デバッガ用のコードを生成および保存するように指示します。
SPECIFICATION	SPECIFICATION を指定して、オブジェクト型の仕様部のみをコンパイルします。
BODY	BODY を指定して、オブジェクト型の本体のみをコンパイルします。

REPLACE AS OBJECT

REPLACE AS OBJECT 句によって、新しいメンバー・サブプログラム仕様を追加します。この句はオブジェクト型のみに有効で、ネストした表型または VARRAY 型には無効です。

element_list

オブジェクトの要素を指定します。

attribute オブジェクトの属性名を指定します。属性とは、名前と型指定子を持つオブジェクト構造を形成するデータ項目です。

MEMBER | STATIC この句によって、属性として参照されるオブジェクト型に関連付けられたファンクションまたはプロシージャ・サブプログラムを指定します。

それぞれのプロシージャまたはファンクションの仕様部について、オブジェクト型本体に対応するメソッド本体を指定する必要があります。

参照：

- メンバー・メソッドとスタティック・メソッドの違い、およびこれらの例については、10-80 ページの「CREATE TYPE」を参照してください。
- パッケージ内のサブプログラム名のオーバーロードの詳細は、『Oracle8*i* PL/SQL 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。
- 10-93 ページの「CREATE TYPE BODY」を参照してください。

procedure_spec プロシージャ・サブプログラムの仕様部を指定します。

function_spec ファンクション・サブプログラムの仕様部を指定します。

pragma_clause *pragma_clause* は、データベースの表またはパッケージ変数（あるいはその両方）に対するメンバー・ファンクションの読み書きアクセスを拒否し、副作用の発生を防止するコンパイラ・ディレクティブです。

参照：『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

method	プラグマが適用されている MEMBER ファンクションまたはプロシージャの名前を指定します。
DEFAULT	DEFAULT を指定すると、プラグマが明示的に指定されていない型のすべてのメソッドにプラグマが適用されます。
WNDS	WNDS を指定すると、データベースの**書込み禁止状態**制約（データベース表を変更しない）が適用されます。
WNPS	WNPS を指定すると、パッケージの**書込み禁止状態**制約（パッケージ変数を変更しない）が適用されます。
RNDS	RNDS を指定すると、データベースの**読取り禁止状態**制約（データベース表を問い合せない）が適用されます。
RNPS	RNPS を指定すると、パッケージの**読取り禁止状態**制約（パッケージ変数を参照しない）が適用されます。
TRUST	TRUST を指定すると、プラグマに指定されている制限が、実際に適用されているのではなく、単に真であることを指定します。

MAP | ORDER MEMBER *function_spec*

MAP メソッドまたは ORDER メソッドのいずれかを宣言できますが、両方は宣言できません。いずれかのメソッドを宣言すると、SQL 内でオブジェクト・インスタンスを比較できます。

どちらのメソッドも宣言しない場合、比較できるのはオブジェクト・インスタンスの等価性と非等価のみです。同じ型定義のインスタンスは、それぞれの対応する属性の各組が等しい場合にのみ等しくなります。2 つのオブジェクト型の等価性を判断するために比較方法を指定する必要はありません。

参照：オブジェクト値の比較の詳細は、2-28 ページの「オブジェクト値」を参照してください。

MAP

オブジェクトの順序付けられたすべてのインスタンスの中から、指定したインスタンスの相対的な位置を戻すメンバー・ファンクション（MAP メソッド）を指定します。MAP メソッドは暗黙的にコールされ、オブジェクト・インスタンスを事前定義済のスカラー型の値にマップすることによって、それらのオブジェクト・インスタンスに順序を設定します。比較演算子および ORDER BY 句の順序が使用されます。

MAP メソッドの引数が NULL の場合、MAP メソッドは NULL に戻り、メソッドは起動されません。

オブジェクトの仕様部には、1 つの MAP メソッドのみを指定することができます。この MAP メソッドは、ファンクションである必要があります。結果として生成される型は、事前定義済の SQL スカラー型である必要があります。また、MAP ファンクションに指定できる引数は、暗黙的な SELF 引数のみです。

> **注意：**（ORDER BY 句、GROUP BY 句、DISTINCT 句または UNION 句を使用する）ソートまたは結合を指定した問合せが *type_name* を参照し、これらの問合せをパラレル化する場合は、MAP メンバー・ファンクションを指定する必要があります。

ORDER

オブジェクトのインスタンスを明示的な引数および暗黙的な SELF 引数として取るメンバー・ファンクション（ORDER メソッド）を指定し、負の整数、0（ゼロ）または正の整数のいずれかを戻します。負の整数は、暗黙的な SELF 引数が明示的な引数より小さいことを示し、0（ゼロ）は、両方が等しいことを示します。また、正の整数は、暗黙的な SELF 引数が明示的な引数より大きいことを示します。

ORDER メソッドの引数が NULL の場合、ORDER メソッドは NULL を戻し、メソッドは起動されません。

同じオブジェクト型定義の各インスタンスを ORDER BY 句の中で比較した場合、ORDER メソッドの関数が呼び出されます。

オブジェクトの仕様部には、1 つの ORDER メソッドのみ指定でき、その ORDER メソッドは戻り型が NUMBER のファンクションである必要があります。

invoker_rights_clause

invoker_rights_clause によって、オブジェクト型のメンバー・ファンクションおよびプロシージャが、そのオブジェクト型を所有するユーザーのスキーマで、特権付きで実行されるか、または CURRENT_USER のスキーマで、特権付きで実行されるかを指定します。この仕様は、対応する型本体にも適用されます。

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。

制限事項 : この句はオブジェクト型のみに指定でき、ネストした表型および VARRAY 型には指定できません。

AUTHID CURRENT_USER

CURRENT_USER を指定して、オブジェクト型のメンバー・ファンクションおよびプロシージャを CURRENT_USER の権限で実行します。この句によって**実行者権限型**が作成されます。

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。その他すべての文の外部名は、型が存在するスキーマ内で変換します。

> **注意 :** 実行者権限を実行者権限状態で作成した場合、この型への実行者権限を維持するためにこの句を指定してください。指定しないと、この状態は定義者権限に戻ります。

AUTHID DEFINER

DEFINER によって、ファンクションおよびプロシージャが存在するスキーマの所有者権限で、オブジェクト型のメンバー・ファンクションおよびプロシージャを実行し、メンバー・ファンクションおよびプロシージャが存在するスキーマで外部名を変換することを指定します。これはデフォルト値です。

参照 :

- CURRENT_USER を判断する方法については、『Oracle8*i* 概要』および『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。
- 『Oracle8*i* PL/SQL 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。

例

メンバー関数の追加　次の文は、メンバー関数 qtr を data_t の型定義に追加します。

```
CREATE TYPE data_t AS OBJECT
   ( year NUMBER,
     MEMBER FUNCTION prod(invent NUMBER) RETURN NUMBER
   );

   CREATE TYPE BODY data_t IS
      MEMBER FUNCTION prod (invent NUMBER) RETURN NUMBER IS
         BEGIN
             RETURN (year + invent);
         END;
      END;

   ALTER TYPE data_t REPLACE AS OBJECT
   ( year NUMBER,
     MEMBER FUNCTION  prod(invent NUMBER) RETURN NUMBER,
     MEMBER FUNCTION  qtr(der_qtr DATE) RETURN CHAR
   );

   CREATE OR REPLACE TYPE BODY data_t IS
      MEMBER FUNCTION prod (invent NUMBER) RETURN NUMBER IS
      MEMBER FUNCTION qtr(der_qtr DATE) RETURN CHAR IS
         BEGIN
            RETURN (year + invent);
         END;
         BEGIN
            RETURN 'FIRST';
         END;
      END;
```

型の再コンパイル　次の文は、型 loan_t を作成し、再コンパイルします。

```
CREATE TYPE loan_t AS OBJECT
  ( loan_num        NUMBER,
    interest_rate   FLOAT,
    amount          FLOAT,
    start_date      DATE,
    end_date        DATE );

ALTER TYPE loan_t COMPILE;
```

型本体の再コンパイル　次の文は、link2 の型本体をコンパイルします。

```
CREATE TYPE link1 AS OBJECT
  (a NUMBER);

CREATE TYPE link2 AS OBJECT
  (a NUMBER,
   b link1,
   MEMBER FUNCTION p(c1 NUMBER) RETURN NUMBER);

CREATE TYPE BODY link2 AS
   MEMBER FUNCTION p(c1 NUMBER) RETURN NUMBER IS t13 link1;
      BEGIN t13 := link1(13);
         dbms_output.put_line(t13.a);
         RETURN 5;
      END;
   END;

CREATE TYPE link3 AS OBJECT (a link2);
CREATE TYPE link4 AS OBJECT (a link3);
CREATE TYPE link5 AS OBJECT (a link4);
ALTER TYPE link2 COMPILE BODY;
```

型仕様部の再コンパイル　次の文は、link2 の型本体をコンパイルします。

```
CREATE TYPE link1 AS OBJECT
  (a NUMBER);

CREATE TYPE link2 AS OBJECT
  (a NUMBER,
   b link1,
   MEMBER FUNCTION p(c1 NUMBER) RETURN NUMBER);

CREATE TYPE BODY link2 AS
    MEMBER FUNCTION p(c1 NUMBER) RETURN NUMBER IS t14 link1;
      BEGIN t14 := link1(14);
         dbms_output.put_line(t14.a);
         RETURN 5;
      END;
    END;

CREATE TYPE link3 AS OBJECT (a link2);
CREATE TYPE link4 AS OBJECT (a link3);
CREATE TYPE link5 AS OBJECT (a link4);
ALTER TYPE link2 COMPILE SPECIFICATION;
```

ALTER USER

用途

ALTER USER は、データベース・ユーザーのユーザー認証またはデータベース・リソース特性を変更する場合に使用します。

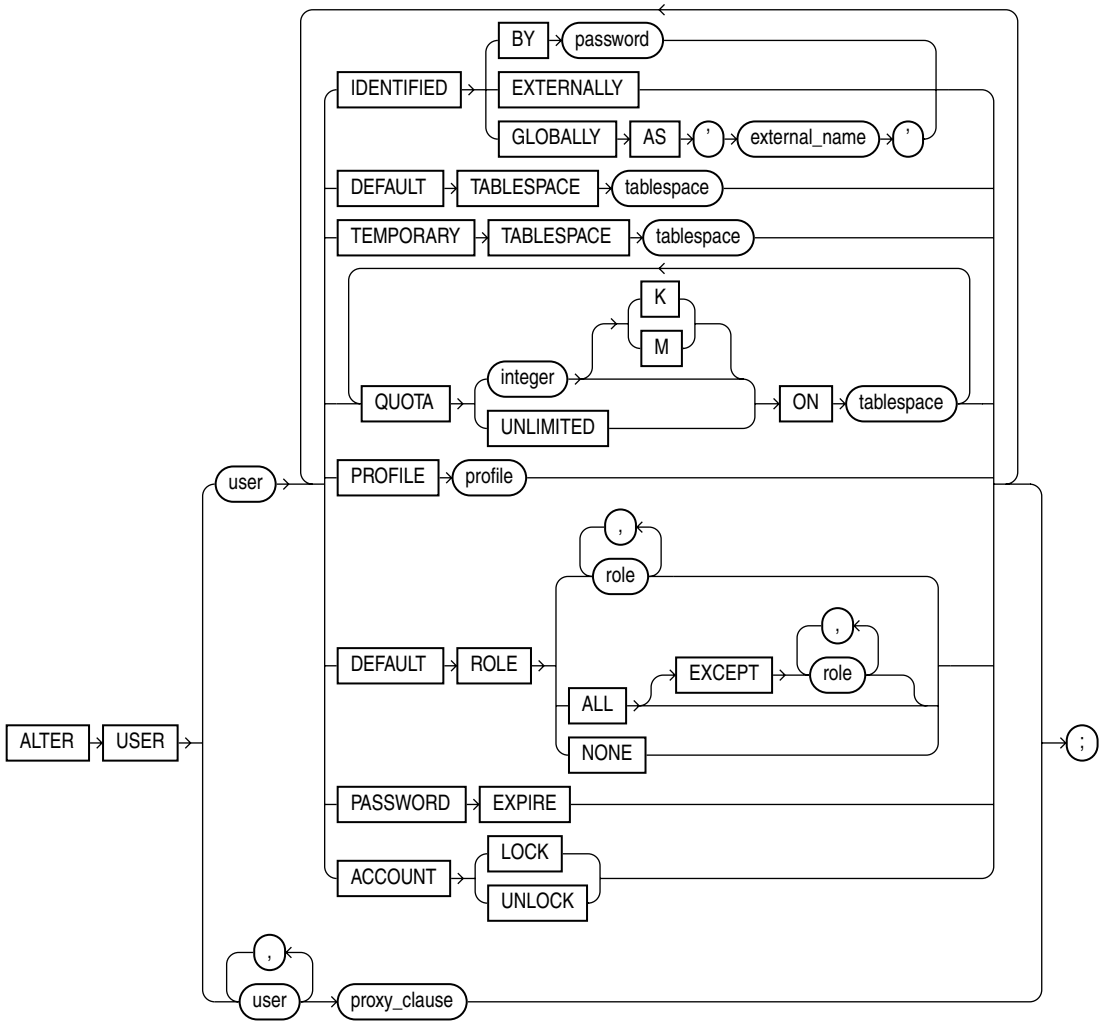
プロキシ・サーバーが認証なしでクライアントとして接続することを許可します。

> **注意：** ALTER USER 構文は、古いパスワードを受け入れません。したがって、この構文では古いパスワードを使用して認証することも、新しいパスワードを設定する前に古いパスワードと新しいパスワードを照合することもできません。古いパスワードをチェックする必要がある場合、ALTER USER のかわりに OCIPasswordChange() コールを使用してください。詳細は、『Oracle8*i* コール・インタフェース・プログラマーズ・ガイド』を参照してください。

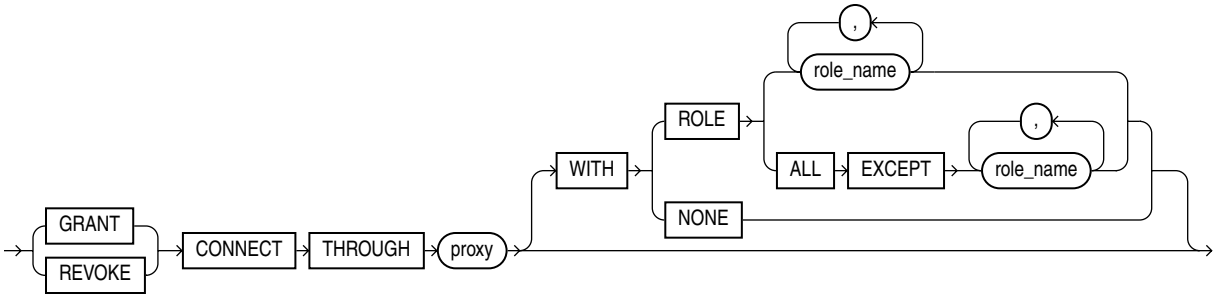
前提条件

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。

構文


ALTER
USER
user
IDENTIFIED
BY
password
EXTERNALLY
GLOBALLY
AS
'
external_name
'
DEFAULT
TABLESPACE
tablespace
TEMPORARY
TABLESPACE
tablespace
QUOTA
integer
K
M
UNLIMITED
ON
tablespace
PROFILE
profile
DEFAULT
ROLE
,
role
ALL
EXCEPT
,
role
NONE
PASSWORD
EXPIRE
ACCOUNT
LOCK
UNLOCK
,
user
proxy_clause
;

proxy_clause::=



キーワードとパラメータ

後述のキーワードおよびパラメータは、ALTER USER に対して一意であるか、または CREATE USER のキーワードおよびパラメータとは異なる機能があります。ALTER USER 文のその他のすべてのキーワードおよびパラメータは、CREATE USER 文のキーワードとパラメータと同じです。

参照：

- キーワードおよびパラメータについては、10-99 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、9-134 ページの「CREATE PROFILE」を参照してください。

IDENTIFIED

BY *password*　　ユーザーのためのパスワードを指定します。

注意： Oracle は、特定のパスワードを再設定するたびに、異なるタイムスタンプを想定します。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）、Oracle はパスワードが再使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、オラクル社は、パスワードの再設定にスクリプトを使用しないことをお薦めします。

GLOBALLY AS　'*external_name*' を指定して、LDAP V3 に準拠するディレクトリ・サービス（Oracle Internet Directory など）を使用して、ユーザーを認証する必要があることを示します

ユーザーに直接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、ユーザー・アクセス検証方法を IDENTIFIED GLOBALLY AS '*external_name*' に変更できます。

IDENTIFIED GLOBALLY AS '*external_name*' として作成されたユーザーを、IDENTIFIED BY *password* または IDENTIFIED EXTERNALLY に変更できます。

参照 : 10-99 ページの「CREATE USER」を参照してください。

DEFAULT ROLE

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 では、ユーザーがパスワードを指定しなかった場合でも、ログイン時にデフォルトのロールは使用可能になります。

参照 : 9-141 ページの「CREATE ROLE」を参照してください。

proxy_clause

proxy_clause によって、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）の機能を制御し、指定したユーザーで接続し、すべてまたはいくつかのユーザー・ロールをアクティブにします。

参照 : プロキシおよびそれらのデータベースの使用方法については、『Oracle8*i* 概要』を参照してください。

GRANT　接続を許可するには、GRANT を指定します。

REVOKE　接続を禁止するには、REVOKE を指定します。

proxy　Oracle に接続するプロキシを識別します。

WITH 句　アプリケーションがユーザーとして接続した後、アクティブにできるロールを指定します。この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。

- ROLE *role_name* は、指定したユーザーとしてプロキシが接続することを許可し、*role_name* で指定されたロールのみをアクティブにします。
- ROLE ALL EXCEPT *role_name* は、指定したユーザーとしてプロキシが接続することを許可し、*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します。
- NONE は、指定したユーザーとしてプロキシが接続することを許可しますが、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を禁止します。

例

ALTER USER の例　次の文は、ユーザー scott のパスワードを lion に変更し、デフォルト表領域を tstest に変更します。

```
ALTER USER scott
    IDENTIFIED BY lion
    DEFAULT TABLESPACE tstest;
```

次の文は、clerk プロファイルを scott に割り当てます。

```
ALTER USER scott
    PROFILE clerk;
```

後続のセッションでは、scott は clerk プロファイル内の制限に従います。

次の文は、scott に直接付与されているすべてのロール（agent ロールを除く）をデフォルト・ロールに設定します。

```
ALTER USER scott
    DEFAULT ROLE ALL EXCEPT agent;
```

scott が次のセッションを開始するときには、agent 以外の、scott に付与されているすべてのロールが使用可能になります。

ユーザー認証の例　次の文は、ユーザー tom の認証を変更します。

```
ALTER USER tom IDENTIFIED GLOBALLY AS 'CN=tom,O=oracle,C=US';
```

次の文は、ユーザー fred のパスワードを期限切れにします。

```
ALTER USER fred 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 Tools 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。

プロキシ・ユーザーの例　次の文は、プロキシ・ユーザー APPSERVER1 をユーザー JANE として接続します。また、この文は、APPSERVER1 がロール INVENTORY をアクティブにすることを許可します。

```
ALTER USER jane GRANT CONNECT THROUGH appserver1 WITH ROLE inventory;
```

次の文は、プロキシ・ユーザー appserver1 からユーザー jane として接続する権限を取り消します。

```
ALTER USER jane REVOKE CONNECT THROUGH appserver1;
```

ALTER VIEW

用途

ALTER VIEW は、無効なビューを明示的に再コンパイルする場合に使用します。明示的に再コンパイルした場合、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、ビューのベース表を変更した後で、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 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。

注意：

- 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。ビューを再定義する場合、OR REPLACE を指定した CREATE VIEW を使用してください。
- 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、問合せの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。

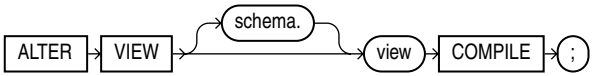
参照：

- ビューの再定義の詳細は、10-105 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・ログの再検証の詳細は、7-59 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、『Oracle8*i* データ・ウェアハウス』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、『Oracle8*i* 概要』を参照してください。

前提条件

ビューが自スキーマ内にある必要があります。自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



キーワードとパラメータ

schema

削除する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。

COMPILE

COMPILE キーワードは必須です。指定したビューが再コンパイルされます。

例

ALTER VIEW の例　次の文は、ビュー customer_view を再コンパイルします。

```
ALTER VIEW customer_view
    COMPILE;
```

customer_view の再コンパイル時にエラーが発生しなければ、customer_view は有効になります。再コンパイル・エラーが発生した場合、エラー・メッセージが戻り、customer_view は無効のままです。

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_view 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、Oracle 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 は、次の処理を行う場合に使用します。

- 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、クラスタあるいはスカラー・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、クラスタあるい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。

統計収集には、DBMS_STATS パッケージを使用することをお薦めします。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することができます。

この項で説明されている目的でこの文を使用することは可能ですが、次の目的に対しては、（DBMS_STATS ではなく）この文を使用する必要があります。

- VALIDATE 句、LIST CHAINED ROWS 句の使用
- （パーセントではなく）行数の抽出
- オプティマイザが使用していない統計（たとえば、空きリストのブロックに関する情報）の収集

> **参照：** このパッケージの詳細は、『Oracle8*i* PL/SQL パッケージ・プロシージャ リファレンス』を参照してください。

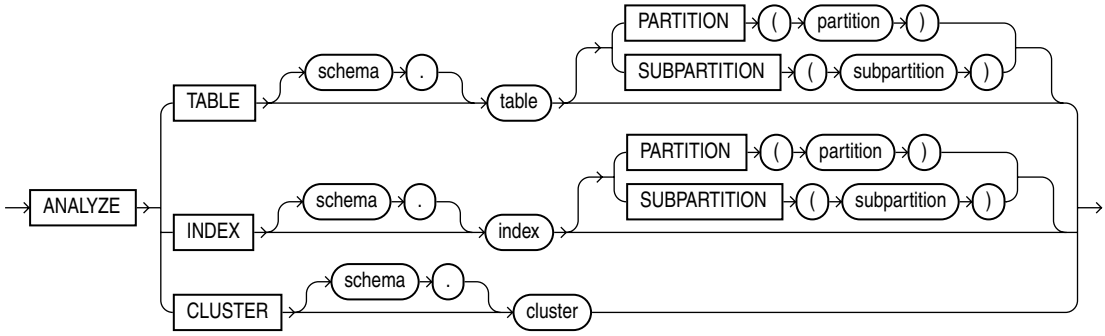
前提条件

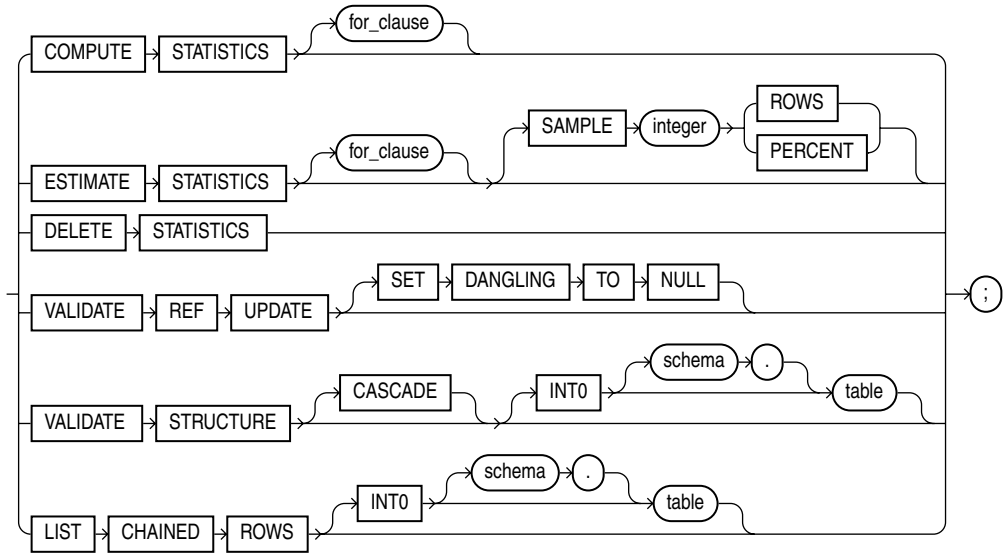
分析するスキーマ・オブジェクトがローカルである必要があります。自スキーマ内にない場合は、ANALYZE ANY システム権限が必要です。

表またはクラスタの連鎖行をリスト表へ入れる場合、このリスト表が自スキーマ内にある必要があります。自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。

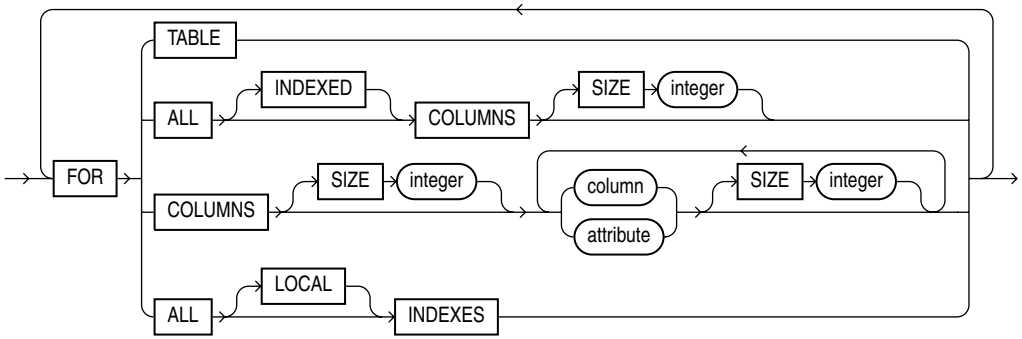
パーティション表の妥当性チェックを行う場合は、分析した ROWID を入れる表に対する INSERT 権限または INSERT ANY TABLE システム権限が必要です。

構文


ANALYZE
TABLE
schema
.
table
PARTITION
(
partition
)
SUBPARTITION
(
subpartition
)
INDEX
schema
.
index
PARTITION
(
partition
)
SUBPARTITION
(
subpartition
)
CLUSTER
schema
.
cluster


COMPUTE
STATISTICS
for_clause
ESTIMATE
STATISTICS
for_clause
SAMPLE
integer
ROWS
PERCENT
DELETE
STATISTICS
VALIDATE
REF
UPDATE
SET
DANGLING
TO
NULL
VALIDATE
STRUCTURE
CASCADE
INT0
schema
.
table
LIST
CHAINED
ROWS
INT0
schema
.
table
;

for_clause::=



キーワードとパラメータ

schema

索引、表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、索引、表またはクラスタは自スキーマ内にあるとみなされます。

INDEX *index*

分析する索引を指定します（*for_clause* を使用しない場合）。

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は、常に厳密に計算されます。**従来索引**についての統計情報は、USER_INDEXES、ALL_INDEXES および DBA_INDEXES のそれぞれのデータ・ディクショナリ・ビューに表示されます。

- ルート・ブロックからリーフ・ブロック（BLEVEL）までの索引の深さ *
- 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・インデックスの場合、索引に関連付けられた統計タイプに指定したユーザー定義統計収集関数が、この文によって起動されます（8-108 ページの「ASSOCIATE STATISTICS」を参照）。ドメイン・インデックス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 で STATISTICS 列に表示されます。

制限事項：LOADING または FAILED のマークが付いたドメイン・インデックスは分析できません。

参照：

- ドメイン・インデックスの詳細は、9-51 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、『Oracle8*i* リファレンス・マニュアル』を参照してください。

TABLE *table*

分析する表を指定します。*for_clauses* を使用しない場合、表の統計情報を収集すると各表の索引およびドメイン・インデックスの統計情報も自動的に収集されます。

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。

参照： ファンクション索引の詳細は、9-51 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・インデックスはすべてスキップされます。

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は、常に厳密に計算されます。表の統計情報（ドメイン・インデックスの状態を含む）は、カッコで示した列のデータ・ディクショナリ・ビュー USER_TABLES、ALL_TABLES および DBA_TABLES 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）

制限事項 :

- 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、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、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（ANALIZE 使用前に対応付けが行われている必要があります）。
- 次の列型に対しての統計は計算および推定できません。
  - REF
  - VARRAY
  - ネストした表
  - LOB（LOB は分析されず、スキップされる）
  - LONG またはオブジェクト型

  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 :

  - 8-108 ページの「ASSOCIATE STATISTICS」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、『Oracle8*i* リファレンス・マニュアル』を参照してください。

PARTITION | SUBPARTITION

統計を収集するパーティションまたはサブパーティションを指定します。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。

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した場合、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、『Oracle8*i* リファレンス・マニュアル』を参照してください。

COMPUTE STATISTICS

COMPUTE STATISTICS は、分析対象のオブジェクトの正確な統計情報を計算してデータ・ディクショナリに格納します。表を分析した場合、表および列の統計情報が収集されます。

計算された統計情報および推定された統計情報は、分析したオブジェクトにアクセスする SQL 文の実行計画を選択するために、Oracle オプティマイザによって使用されます。また、これらの統計情報は SQL 文を記述するアプリケーション開発者にも役立ちます。

> **参照：** これらの統計情報の使用方法については、『Oracle8*i* パフォーマンスのための設計およびチューニング』を参照してください。

for_clause

for_clause は、表または索引全体を分析するか、特定の列のみを分析するかを指定します。次に示す句は、この文の ANALYZE TABLE にのみ使用できます。

FOR TABLE	FOR TABLE を指定して、表および列の情報ではなく、表のみの統計情報が収集されるように制限します。
FOR COLUMNS	FOR COLUMNS を指定して、すべての列または属性の列統計情報ではなく、指定した列およびスカラー・オブジェクト属性のみの列統計情報が収集されるように制限します。属性は、オブジェクト内の項目の修飾列名を指定します。
FOR ALL COLUMNS	FOR ALL COLUMNS を指定して、すべての列およびスカラー・オブジェクト属性の列統計情報を収集します。

FOR ALL INDEXED COLUMNS

FOR ALL INDEXED COLUMNS を指定して、表にあるすべての索引付きの列の列統計情報を収集します。

列の統計情報を収集する場合、列全体に基づいた情報を収集するか、SIZE integer を指定してヒストグラムを使用します（後述を参照）。

Oracle は、次の列統計情報を収集します。

- 列全体として、重複していない値の数
- 各区間の最大値と最小値

参照 : ヒストグラムの詳細については、『Oracle8*i* パフォーマンスのための設計およびチューニング』および 8-106 ページの「ヒストグラムの例」を参照してください。

列の統計情報は、データ・ディクショナリ・ビュー USER_TAB_COLUMNS、ALL_TAB_COLUMNS および DBA_TAB_COLUMNS に表示されます。ヒストグラムは、データ・ディクショナリ・ビュー USER_TAB_HISTOGRAMS、DBA_TAB_HISTOGRAMS、ALL_TAB_HISTOGRAMS; USER_PART_HISTOGRAMS、DBA_PART_HISTOGRAMS、ALL_PART_HISTOGRAMS; USER_SUBPART_HISTOGRAMS、DBA_SUBPART_HISTOGRAMS および ALL_SUBPART_HISTOGRAMS に表示されます。

注意 : USER_、DBA_ および ALL_TAB_COLUMNS の MAXVALUE 列および MINVALUE 列は、長さが 32 バイトです。32 バイトより長い列を分析する場合、および列に先行空白が埋め込まれている場合、Oracle は、先行空白のみを考慮し、予期しない統計情報を戻します。

ユーザー定義型が列に関連付けられている場合、*for_clause* は、その統計タイプを使用してユーザー定義統計情報を収集します。列に関連付けられた統計タイプがない場合、列の型に関連付けられた統計タイプがあるかがチェックされ、ある場合は、その統計タイプが使用されます。列またはそのユーザー定義型に関連付けられた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。ユーザー定義列統計情報は、データ・ディクショナリ・ビュー USER_USTATS、ALL_USTATS および DBA_USTATS で STATISTICS 列に表示されます。

表全体および 1 つ以上の列の両方の統計情報を収集する場合、まず、表の統計情報を作成してから、次に、列の統計情報を作成してください。このようにしないと、表のみの ANALYZE が列 ANALYZE で作成されたヒストグラムを上書きしてしまいます。次の文はこの手順の例です。

```
ANALYZE TABLE emp ESTIMATE STATISTICS;
ANALYZE TABLE emp ESTIMATE STATISTICS
   FOR ALL COLUMNS;
```

FOR ALL INDEXES	FOR ALL INDEXES を指定して、表に関連付けられたすべての索引を分析します。
FOR ALL LOCAL INDEXES	FOR ALL LOCAL INDEXES を指定して、すべてのローカル索引パーティションを分析します。PARTITION 句および INDEX が指定されている場合、キーワード LOCAL を指定する必要があります。
SIZE *integer*	ヒストグラムのバケットの最大数を指定します。デフォルト値は 75 で、最小値は 1、最大値は 254 です。

注意：サンプルの行数以上のバケットを持つヒストグラムは作成しません。また、サンプルに繰返しが非常に多い値が含まれる場合、指定された数のバケットは作成されますが、ALL_、DBA_ および USER_TAB_COLUMNS ビューの NUM_BUCKETS 列で指定した値は、内部圧縮アルゴリズムのために小くなる場合があります。

ESTIMATE STATISTICS

ESTIMATE STATISTICS は、分析対象オブジェクトの統計情報を概算してデータ・ディクショナリに格納します。

計算された統計情報および推定された統計情報は、分析したオブジェクトにアクセスする SQL 文の実行計画を選択するために、Oracle オプティマイザによって使用されます。また、これらの統計情報は SQL 文を記述するアプリケーション開発者にも役立ちます。

参照： これらの統計情報の使用方法については、『Oracle8*i* パフォーマンスのための設計およびチューニング』を参照してください。

for_clause	8-100 ページの「COMPUTE STATISTICS」を参照してください。
SAMPLE *integer*	統計情報を推定するために、分析対象のオブジェクトからサンプルとして抽出されるデータ量を指定します。このパラメータを指定しない場合、サンプルとして 1064 行が抽出されます。

サンプルのデフォルト値は、数千行までが表に対して適切な値です。表が大きい場合、より大きい値を SAMPLE に指定します。データの半分以上を指定した場合、すべてのデータが読み込まれ、統計が計算されます。

- ROWS は、表またはクラスタの *integer* 行、または索引の *integer* エントリ数をサンプルとして抽出します。*integer* は 1 以上である必要があります。
- PERCENT は、表またはクラスタの *integer* パーセント、または索引エントリの *integer* パーセントをサンプルとして抽出します。*integer* は 1 ～ 99 の範囲で指定します。

DELETE STATISTICS

DELETE STATISTICS は、現在データ・ディクショナリに格納されている、分析対象のオブジェクトの統計情報を削除します。Oracle 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、指定した表の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）の統計情報が自動的に削除されます。

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、Oracle 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関数を起動して、ユーザー定義統計情報も削除します。

VALIDATE REF UPDATE

VALIDATE REF UPDATE を指定して、指定された表の REF の妥当性チェックを行い、各 REF 内の ROWID 部分をチェックし、それを真の ROWID と比較します。間違っている場合は修正します。この句は、表を分析する場合にのみ使用できます。

SET DANGLING TO NULL

SET DANGLING TO NULL は、指定した表内の REF 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は、REF を NULL に設定します。

注意 : 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。この結果、オブジェクトに対して適切な権限がある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 を使用することはできません。

VALIDATE STRUCTURE

VALIDATE STRUCTURE を指定して、分析対象オブジェクトの構造を検証します。COMPUTE STATISTICS 句および ESTIMATE STATISTICS 句で収集された統計情報は、Oracle オプティマイザで使用されますが、この句で収集された統計情報は、Oracle オプティマイザでは使用されません。

- 表に対して、表のそれぞれのデータ・ブロックと行の整合性を検証します。
- クラスタに対して、自動的にクラスタ表の構造を検証します。
- 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を検証します。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、現行セッション中に表の構造および索引を検証します。
- 索引に対して、索引のそれぞれのデータ・ブロックの整合性を検証し、ブロックの破損をチェックします。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。

データ・ディクショナリ・ビュー INDEX_STATS および INDEX_HISTOGRAM 内にある索引についての統計情報が格納されます。

> **参照：** これらのビューの詳細は、『Oracle8*i* リファレンス・マニュアル』を参照してください。

オブジェクトの構造の妥当性チェックを行った場合、そのオブジェクトに対して SELECT 文、INSERT 文、UPDATE 文および DELETE 文を同時に実行できなくなります。このため、データベース・アクティビティが高い期間中は、稼働アプリケーションの表、クラスタおよび索引に対してこの句を使用しないでください。

オブジェクトの構造に障害がある場合には、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直す必要があります。

INTO *table*　　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、このリスト表は自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　表またはクラスタに関連付けられた索引の構造を検証する場合は、CASCADE を指定します。表を検証するときにこの句を指定すると、その表の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、クラスタ化表のすべての索引（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。

LIST CHAINED ROWS

LIST CHAINED ROWS によって、分析対象の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 *table*　移行行および連鎖行のリスト表を指定します。*schema* を指定しない場合、この表が自スキーマにあるものとみなされます。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。このリスト表はローカル・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 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）。
- UTLCHN1.SQL は、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形式に従う必要があります。

参照 : これらのスクリプトを使用する場合の互換性については、『Oracle8*i* 移行ガイド』を参照してください。

注意 : 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：

- SQL スクリプトの詳細は、『Oracle8*i* PL/SQL パッケージ・プロシージャ 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、『Oracle8*i* パフォーマンスのための設計およびチューニング』を参照してください。

例

クラスタの分析例　次の文は、cust_history 表とそのすべての索引の統計情報を推定します。

```
ANALYZE TABLE cust_history
   ESTIMATE STATISTICS;
```

統計情報の削除例　次の文は、データ・ディクショナリから cust_history 表とこの表のすべての索引に関する統計情報を削除します。

```
ANALYZE TABLE cust_history
   DELETE STATISTICS;
```

ヒストグラムの例　次の文は、EMP 表の SAL 列について 10 区間のヒストグラムを作成します。

```
ANALYZE TABLE emp
   COMPUTE STATISTICS FOR COLUMNS sal SIZE 10;
```

USER_TAB_COLUMNS データ・ディクショナリ・ビューに問い合せて、統計情報を取り出すことができます。

```
SELECT NUM_DISTINCT, NUM_BUCKETS, SAMPLE_SIZE
   FROM USER_TAB_COLUMNS
   WHERE TABLE_NAME = 'EMP' AND COLUMN_NAME = 'SAL';

NUM_DISTINCT NUM_BUCKETS SAMPLE_SIZE
------------ ----------- -----------
          12           7          14
```

ANALYZE 文で 10 のバケットが指定されていますが、この例では 7 のみ作成されます。詳細は、8-102 ページの「SIZE integer」を参照してください。

また、表の単一のパーティションのヒストグラムも収集できます。次の文は、emp 表のパーティション p1 を分析します。

```
ANALYZE TABLE emp PARTITION (p1) COMPUTE STATISTICS;
```

索引の分析例　次の文は、索引 parts_index の構造を検証します。

```
ANALYZE INDEX parts_index VALIDATE STRUCTURE;
```

表の分析例　次の文は、emp 表とそのすべての索引を分析します。

```
ANALYZE TABLE emp VALIDATE STRUCTURE CASCADE;
```

表に対する VALIDATE REF UPDATE 句は、指定した表の REF 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、emp 表内の REF の妥当性チェックを行います。

```
ANALYZE TABLE emp
   VALIDATE REF UPDATE;
```

クラスタの分析例　次の文は、order_custs クラスタ、この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）を分析します。

```
ANALYZE CLUSTER order_custs
    VALIDATE STRUCTURE CASCADE;
```

連鎖行のリスティング例　次の文は、order_hist 表のすべての連鎖行についての情報を収集します。

```
ANALYZE TABLE order_hist
    LIST CHAINED ROWS INTO cr;
```

この文では、情報は表 cr に格納されます。次の問合せで、その行を検証できます。

```
SELECT *
    FROM cr;

OWNER_NAME  TABLE_NAME  CLUSTER_NAME  HEAD_ROWID         TIMESTAMP
----------  ----------  ------------  ------------------ ---------
SCOTT       ORDER_HIST                AAAAZzAABAAABrXAAA 15-MAR-96
```

COMPUTE STATISTICS の例　次の文は、スカラー・オブジェクト属性の統計情報を計算します。

```
ANALYZE TABLE emp COMPUTE STATISTICS FOR COLUMNS addr.street;
```

ASSOCIATE STATISTICS

用途

ASSOCIATE STATISTICS 文は、統計収集、選択性またはコストに関する関数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・インデックスまたは索引タイプに関連付ける場合に使用します。

現在のすべての統計タイプの関連付けの一覧については、USER_ASSOCIATIONS 表を参照してください。統計情報と関連付けられたオブジェクトを分析する場合、USER_USTATS 表でその関連性を表示することができます。

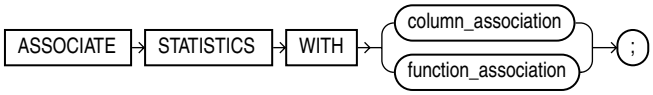
> **参照：** ANALYZE が使用される関連性の優先順位については、8-95 ページの「ANALYZE」を参照してください。

前提条件

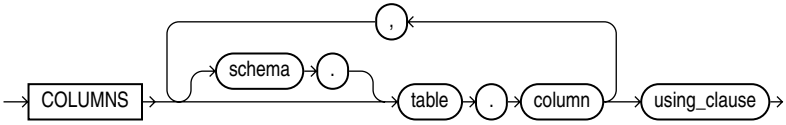
この文を発行する場合は、ベース・オブジェクト（表、ファンクション、パッケージ、型、ドメイン・インデックスまたは索引タイプ）を変更する適切な権限が必要です。さらに、デフォルト統計情報のみを関連付けていない限り、統計タイプに対する実行権限が必要です。統計タイプは、すでに定義されている必要があります。

> **参照：** 型の定義の詳細は、10-80 ページの「CREATE TYPE」を参照してください。

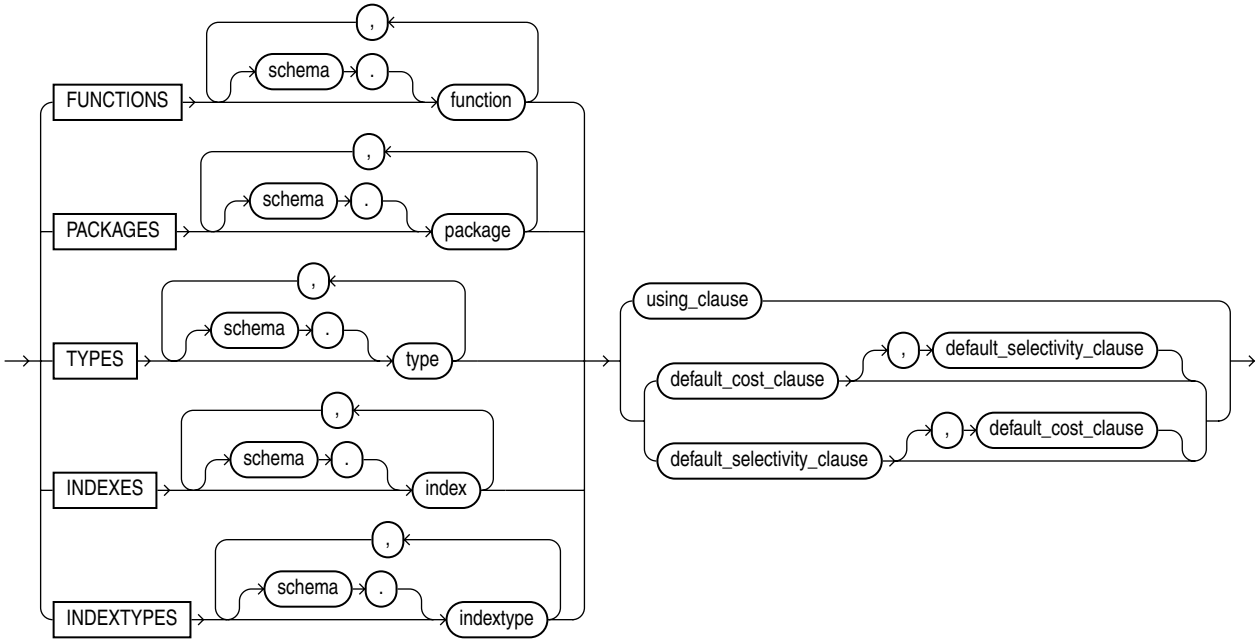
構文



column_association::=

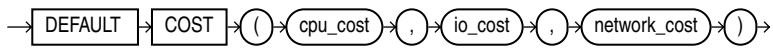


function_association::=



using_clause::=

default_cost_clause::=



default_selectivity_clause::=

キーワードとパラメータ

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、表が自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・インデックスまたは索引タイプを指定します。*schema* で指定しない場合、オブジェクトは自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、内部 SQL データ型は参照しません。

制限事項 : すでに関連性を定義したオブジェクトは指定できません。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 :** 10-121 ページの「DISASSOCIATE STATISTICS」を参照してください。

using_clause

列、ファンクション、パッケージ、型、ドメイン・インデックスまたは索引タイプと関連付けられている統計タイプを指定します。*statistics_type* は作成済である必要があります。

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・インデックス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、関数またはメソッドを一度実行した場合、またはドメイン・インデックスへ一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* は、0 ～ 100 の整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。

制限事項 : ドメイン・インデックスまたは索引タイプに、DEFAULT SELECTIVITY は指定できません。

例

スタンドアロン・ファンクションの例　次の文は、スタンドアロン・ファンクション FN の関連性を作成し、オプティマイザが統計タイプ stat_fn にある適切なコスト・ファンクション（存在する場合）をコールします。

```
ASSOCIATE STATISTICS WITH FUNCTIONS fn USING stat_fn;
```

デフォルト・コストの例　次の文は、ドメイン・インデックス t_a 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 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 t_a 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・コストをそのまま使用します。

AUDIT

用途

AUDIT 文は、次の処理を行う場合に使用します。

- 後続のユーザー・セッションでの SQL 文の発生の監査。

  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 文の発生を監査します。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、現行セッションにも適用されます。

> **参照：** SQL 文の監査を使用禁止にする方法については、11-66 ページの「NOAUDIT」を参照してください。

前提条件

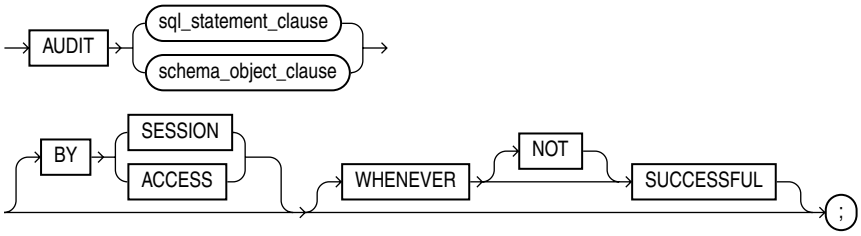
SQL 文の発生を監査するには、AUDIT SYSTEM システム権限が必要です。

スキーマ・オブジェクト操作を監査するためには、監査対象のオブジェクトが自スキーマにあるか、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。

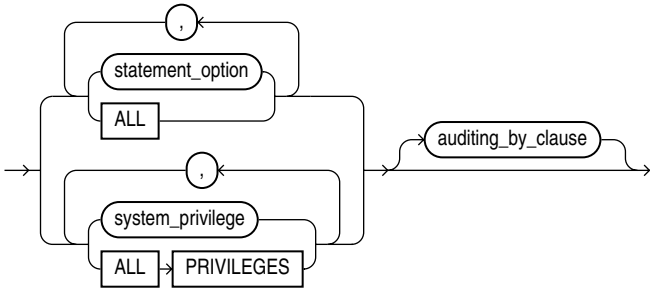
監査結果を収集するには、初期化パラメータ AUDIT_TRAIL を DB に設定する必要があります。監査オプションは、監査が使用可能であるかどうかにかかわらず指定できます。ただし、監査を使用可能にしなければ、監査レコードは作成されません。

> **参照：** AUDIT_TRAIL パラメータについての詳細は、『Oracle8*i* リファレンス・マニュアル』を参照してください。

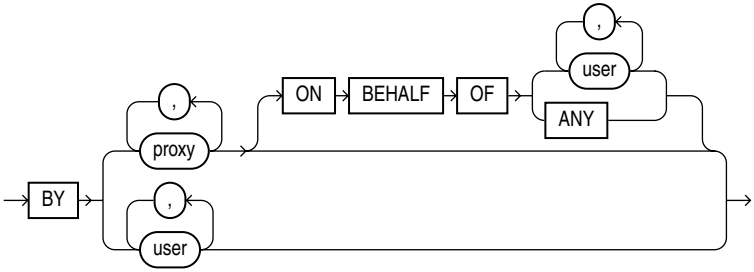
構文



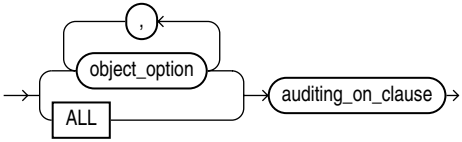
sql_statement_clause::=



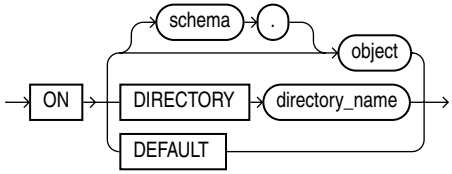
auditing_by_clause::=



schema_object_clause::=



auditing_on_clause::=



キーワードとパラメータ

sql_statement_clause

statement_option

特定の SQL 文を監査するには、文オプションを指定します。

参照：これらの文オプションとそれによって監査される SQL 文のリストは、8-118 ページの表 8-1 および 8-121 ページの表 8-2 を参照してください。

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：これらのビューの詳細は、『Oracle8*i* リファレンス・マニュアル』を参照してください。

system_privilege

特定のシステム権限を与えられた SQL 文を監査する場合は、システム権限を指定します。

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、表 11-1 を参照してください。

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、すべてのシステム権限がそのロールに付与されていることを監査することと同じです。

参照 : これらのロールの詳細は、11-31 ページの「GRANT」を参照してください。

Oracle 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 は、表 8-1 のすべての文オプションを監査しますが、表 8-2 の追加文オプションを監査しません。
ALL PRIVILEGES	システム権限を監査するためには、ALL PRIVILEGES を指定します。

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 の将来のバージョンでサポートされない場合があります。

auditing_by_clause

auditing_by_clause は、特定のユーザーが発行する SQL 文のみを監査します。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*	この句は、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します。
BY *proxy*	この句は、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します。 **参照 :** プロキシおよびそれらのデータベースについては、『Oracle8*i* 概要』を参照してください。
ON BEHALF OF	■ *user* は、特定のユーザーのプロキシとして実行された文を監査することを指定します。 ■ ANY は、すべてのユーザーのプロキシとして実行された文を監査することを指定します。

schema_object_clause

object_option 監査の対象とする操作を指定します。8-122 ページの表 8-3 に、各オブジェクト・オプションとそれが適用されるオプションのタイプを示します。それぞれのオブジェクト・オプションには、監査の対象となる SQL 文を指定します。たとえば、ALTER オプション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 オプション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オプションをすべて指定することと同じです。

auditing_on_clause *auditing_on_clause* は、特定のスキーマ・オブジェクトを監査できます。

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、パッケージまたはマテリアライズド・ビューについては、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・オプションとして構築します。デフォルト監査オプションを指定した場合、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、そのビューのベース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。

ON DIRECTORY *directory_name*

ON DIRECTORY 句によって、監査対象のディレクトリ名を指定することができます。

BY SESSION

同一セッションの同一スキーマ・オブジェクトで発行された同じ種類の SQL 文すべて、および実行された同じ種類の操作について、1 つのレコードが書き込まれるようにする場合は、BY SESSION を指定します。

BY ACCESS

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場合は、BY ACCESS を指定します。

DDL 文を監査する文オプションまたはシステム権限を指定した場合、BY SESSION 句と BY ACCESS 句のどちらを指定しても、Oracle は、自動的に監査を行います。

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

SQL 文および操作が正常に実行された場合のみに監査する場合は、WHENEVER SUCCESSFUL を指定します。

SQL 文および操作が失敗またはエラーが発生した場合のみに監査する場合は、WHENEVER NOT SUCCESSFUL を指定します。

この句を省略すると、処理結果にかかわらず監査を実行します。

監査オプションの表

表 8-1 データベース・オブジェクトの文監査オプション

文オプション	SQL 文と操作
CLUSTER	CREATE CLUSTER
	AUDIT 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
	DROP INDEX
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
PROCEDURE[a]	CREATE FUNCTION
	CREATE LIBRARY
	CREATE PACKAGE
	CREATE PACKAGE BODY
	CREATE PROCEDURE
	DROP FUNCTION
	DROP LIBRARY
	DROP PACKAGE
	DROP PROCEDURE

表 8-1 データベース・オブジェクトの文監査オプション（続き）

文オプション	SQL 文と操作
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 STATEMENT	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 8-1 データベース・オブジェクトの文監査オプション（続き）

文オプション	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
	ENABLE 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
VIEW	CREATE VIEW
	DROP VIEW

[a] 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 8-2 SQL 文のその他の文監査オプション

文オプション	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
	プロシージャまたはファンクションの実行。または、変数、ライブラリ、パッケージ内のまたはカーソルへのアクセス。
GRANT DIRECTORY	GRANT privilege ON directory
	REVOKE privilege ON directory
GRANT PROCEDURE	GRANT privilege ON procedure, function, package
	GRANT 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 8-3 オブジェクト監査オプション

オブジェクト・オプション	表	ビュー	順序	プロシージャ、ファンクション、パッケージ[a]	マテリアライズド・ビュー/スナップショット	ディレクトリ	ライブラリ	オブジェクト・タイプ	コンテキスト
ALTER	X		X		X			X	
AUDIT	X	X	X	X	X	X		X	X
COMMENT	X	X			X				
DELETE	X	X			X				
EXECUTE				X			X		
GRANT	X	X	X	X		X	X	X	X
INDEX	X				X				
INSERT	X	X			X				
LOCK	X	X			X				
READ						X			
RENAME	X	X		X	X				
SELECT	X	X	X		X				
UPDATE	X	X			X				

[a] 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。

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 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 scott および blake が発行した文のうち、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 scott, blake;
```

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;
```

表の問合せの監査例　次の文は、スキーマ scott 内の emp 表を問い合せる各 SQL 文について監査を行います。

```
AUDIT SELECT
    ON scott.emp;
```

次の文は、スキーマ scott 内の emp 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 scott.emp
    WHENEVER SUCCESSFUL;
```

次の文は、スキーマ scott 内の emp 表を問い合せて、エラーが発生した SQL 文について監査を行います。

```
AUDIT SELECT
    ON scott.emp
    WHENEVER NOT SUCCESSFUL;
```

表の挿入および更新の監査例　次の文は、スキーマ blake 内の dept 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 blake.dept;
```

順序に対するすべての操作の監査例　次の文は、スキーマ adams 内の order 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 adams.order;
```

この文は、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

ディレクトリに対する読込み操作の監査例　次の文は、bfile_dir1 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_dir1;
```

デフォルト監査操作の設定例　次の文は、その後作成されるオブジェクトについて、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;
```

その後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

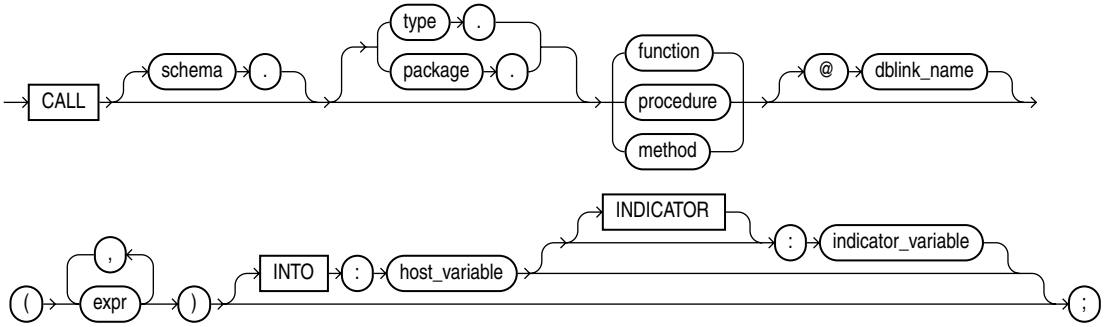
CALL 文は、SQL 内から**ルーチン**（スタンドアロン・プロシージャ、スタンドアロン・ファンクション、あるいは型またはパッケージ内で定義されたプロシージャまたはファンクション）を実行する場合に使用します。

> **参照：** このようなルーチンの作成の詳細は、『Oracle8*i* PL/SQL 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。

前提条件

スタンドアロン・ルーチン、あるい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



キーワードとパラメータ

schema

スタンドアロン・ルーチン（あるいは、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、ルーチンが自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。

function | *procedure* | *method*

コールするファンクション名またはプロシージャ名、あるい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。

型のメンバーであるファンクションまたは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 であれば、エラーが戻されます。SELF が NULL の IN 引数の場合、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起動されません。

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（あるいは、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* を指定しなかった場合、ローカル・データベースを指定したとみなされます。

expr

ルーチンに 1 つ以上の引数を指定します

制限事項 :

- *expr* には、疑似列、オブジェクト参照関数 VALUE または相関変数 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 *expr* は、ホスト変数の式に対応している必要があります。

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、『Oracle8*i* Pro*C/C++ プリコンパイラ・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、プロシージャ updateSalary を作成してからそのプロシージャをコールし、指定された従業員 ID を新しい給与で更新します。

```
CREATE OR REPLACE PROCEDURE updateSalary
   (id NUMBER, newsalary NUMBER) IS
   BEGIN
      UPDATE emp SET sal=newsalary WHERE empno=id;
   END;

CALL updateSalary(1404, 50000);
```

COMMENT

用途

COMMENT は、表、ビュー、スナップショットまたは列に関するコメントを、データ・ディクショナリに追加する場合に使用します。

データ・ディクショナリ・ビューの 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。

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。

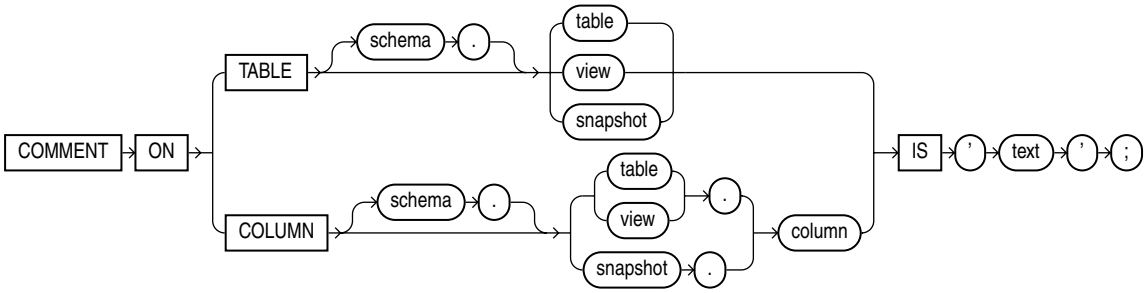
参照 :

- 8-129 ページの「COMMENT」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、『Oracle8*i* リファレンス・マニュアル』を参照してください。

前提条件

表、ビューまたはスナップショットが自スキーマ内にある必要があります。自スキーマ内にない場合は、COMMENT ANY TABLE システム権限が必要です。

構文



キーワードとパラメータ

TABLE

コメントする表、ビューまたはマテリアライズド・ビューの名前とスキーマです。*schema* を指定しない場合、この表、ビューおよびスナップショットは自スキーマ内にあるとみなされます。

COLUMN

コメントする表、ビューまたはスナップショットの列の名前を指定します。*schema* を指定しない場合、この表、ビューおよびスナップショットは自スキーマ内にあるとみなされます。

IS '*text*'

コメントのテキストを指定します。

> **参照：** '*text*' の構文については、2-32 ページの「Text（テキスト）」を参照してください。

例

COMMENT の例　次の文は、shipping 表の notes 列にコメントを挿入します。

```
COMMENT ON COLUMN shipping.notes
    IS 'Special packing or shipping instructions';
```

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 shipping.notes IS ' ';
```

COMMIT

用途

COMMIT 文は、現行トランザクションを終了し、トランザクションで実行された変更をすべて確定する場合に使用します。トランザクションとは、Oracle 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、トランザクションのロックが解除されます。

注意： Oracle 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、次のことができます。

- 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 文で始まる読取り専用トランザクションを終了します。

Oracle 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してください。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 Tools が正常に終了すると、現行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、現行トランザクションは Oracle によってロールバックされます。

参照：

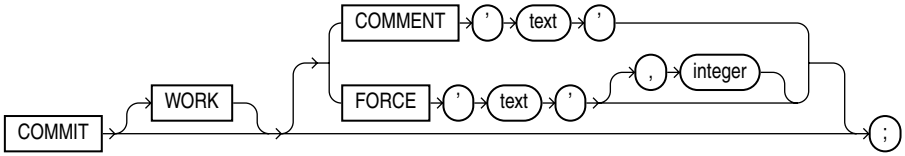
- トランザクションの詳細は、『Oracle8*i* 概要』を参照してください。
- トランザクションの特性の指定については、11-125 ページの「SET TRANSACTION」を参照してください。

前提条件

現行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



キーワードとパラメータ

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 '*text*'

現行トランザクションに関するコメントを指定します。'*text*' は、データ・ディクショナリ・ビュー DBA_2PC_PENDING に格納される、引用符で囲まれた最大 50 文字のリテラルです。トランザクションの状態が不明（インダウト）になった場合は、そのトランザクション ID とともに格納されます。

> **参照：** SQL 文へのコメントの追加については、8-129 ページの「COMMENT」を参照してください。

FORCE '*text*'

分散データベースシステムでは、FORCE 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 '*text*' 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。

> **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るため注意が必要です。この文は、現行トランザクションには影響しません。

制限事項 : FORCE 句を使用した COMMIT 文は、PL/SQL ではサポートされていません。

> **参照 :** これらの項目の詳細は、『Oracle8*i* 分散システム』を参照してください。

例

挿入のコミット例 次の文は、dept 表に行を挿入し、この変更をコミットします。

```
INSERT INTO dept VALUES (50, 'MARKETING', 'TAMPA');
COMMIT WORK;
```

COMMIT および COMMENT の例 次の文は、現行トランザクションをコミットして、コメントを対応付けます。

```
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、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。

```
COMMIT FORCE '22.57.53';
```

constraint_clause

用途

CREATE TABLE の *constraint_clause* または ALTER TABLE 文は、整合性制約を定義する場合に使用します。整合性制約とは、表または索引構成表の 1 つ以上の列に対する値を制限する規則です。

注意： 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の制約はサポートされていません。唯一の例外として、オブジェクト、VARRAY、REF または LOB の型の列または属性に対して NOT NULL 制約のみがサポートされています。

前提条件

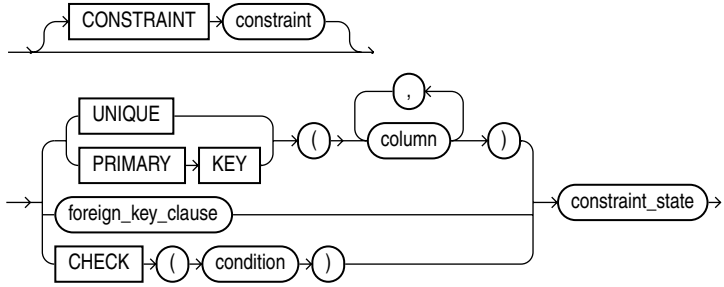
制約句は、CREATE TABLE 文または ALTER TABLE 文のいずれかで指定できます。整合性制約を定義する場合、これらの文のどちらかを発行する権限が必要です。

参照整合性制約を作成する場合、親表が自スキーマ内に設定されている必要があります。または、親表の参照キー列に対する REFERENCES 権限が必要です。

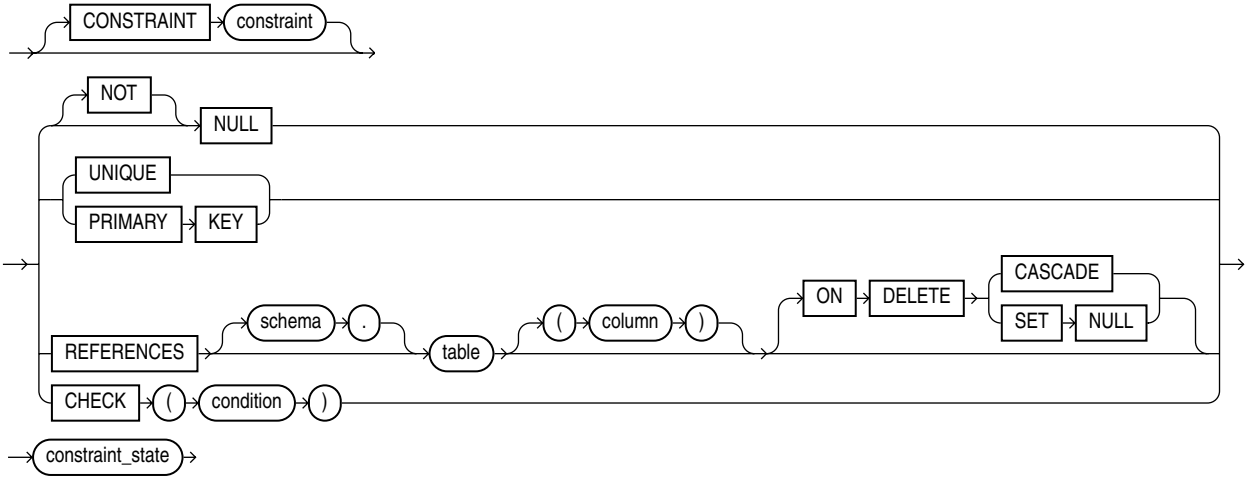
参照： 10-7 ページの「CREATE TABLE」および 8-2 ページの「ALTER TABLE」を参照してください。

構文

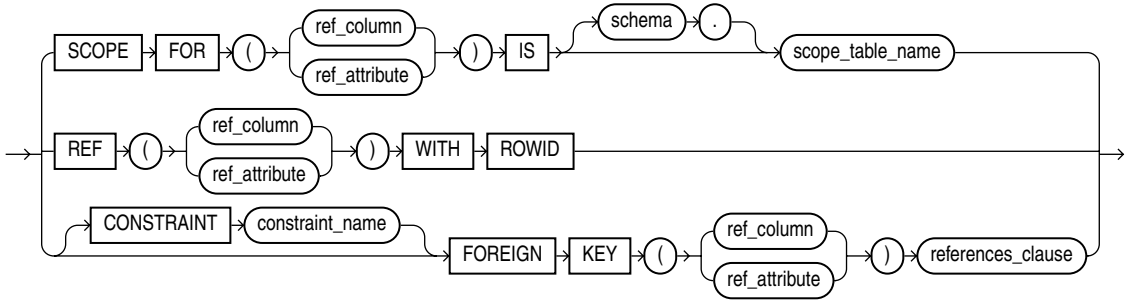
table_constraint::=



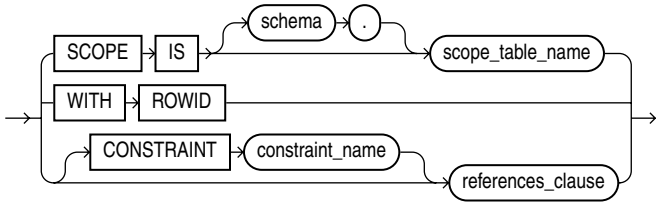
column_constraint::=



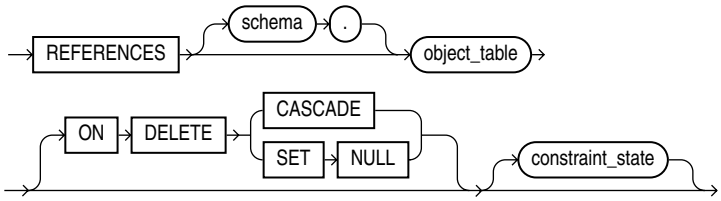
table_ref_constraint::=



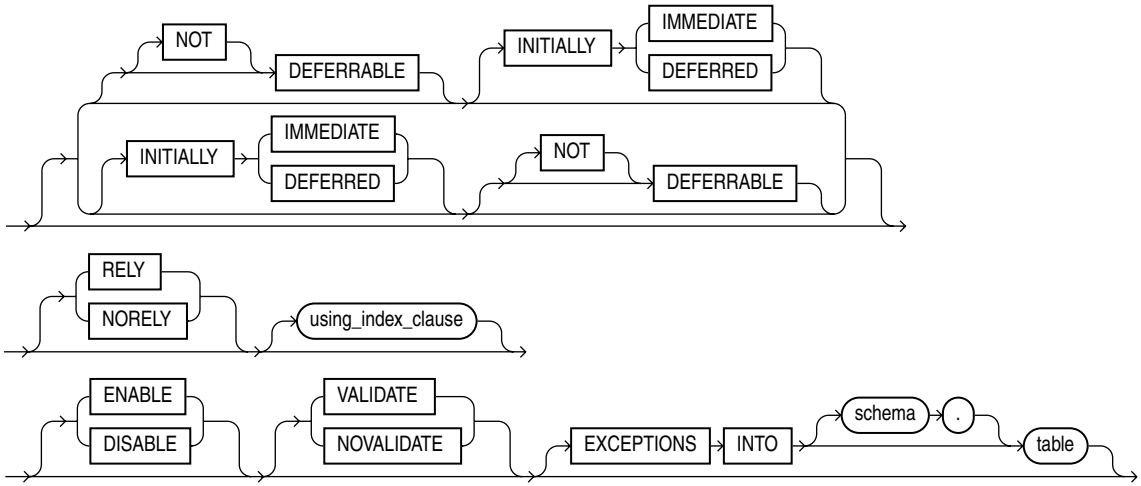
column_ref_constraint::=



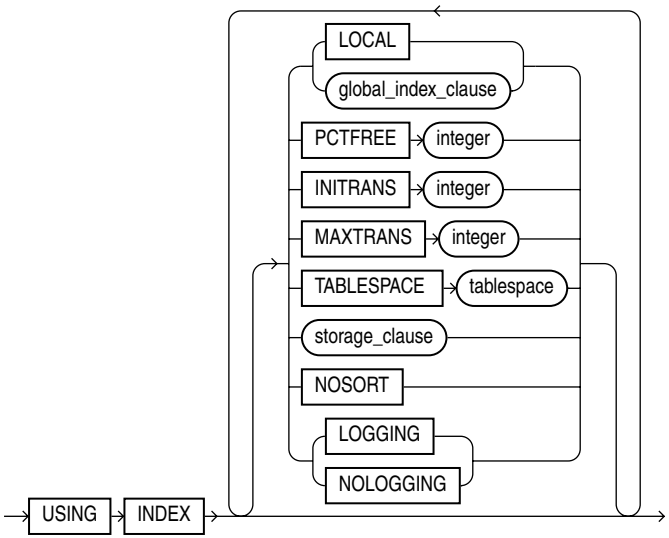
references_clause::=



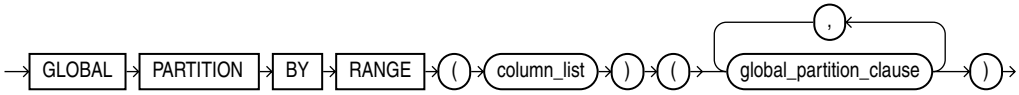
constraint_state::=



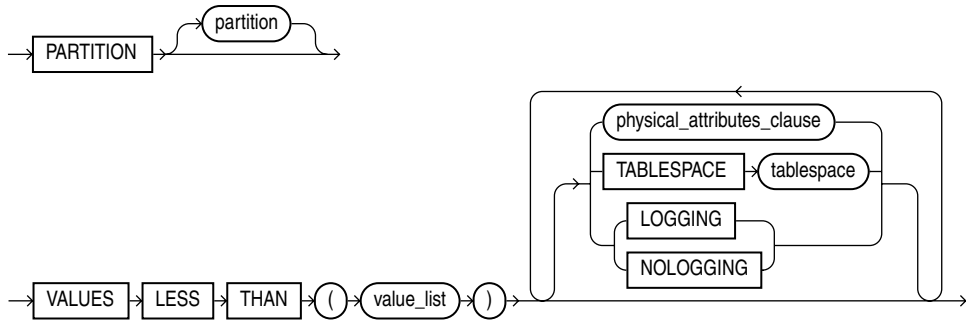
using_index_clause::=



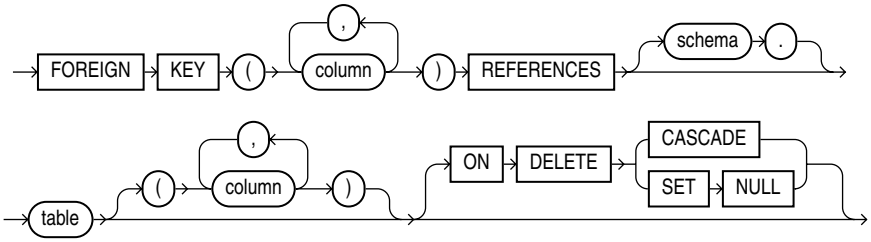
global_index_clause::=



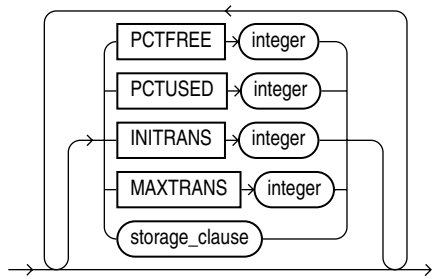
global_partition_clause::=



foreign_key_clause::=



physical_attributes_clause::=



storage_clause: 11-129 ページの「storage_clause」を参照してください。

キーワードとパラメータ

table_constraint

table_constraint 構文は表定義の一部です。この構文によって定義した整合性制約は、表のすべての列に規則を適用します。

table_constraint 構文は、CREATE TABLE 文または ALTER TABLE 文で指定できます。NOT NULL 制約以外の整合性制約を定義できます。

column_constraint

column_constraint 構文は列定義の一部です。一般的に、この構文によって定義した整合性制約は、定義された列に対してのみ規則を適用します。

- CREATE TABLE 文または ALTER TABLE ADD 文に指定する *column_constraint* 構文は、すべての整合性制約を定義できます。
- ALTER TABLE MODIFY *column_options* 文に指定する *column_constraint* 構文は、NOT NULL 制約の定義または削除のみできます。

制限事項 : VARRAY 列に許可された列制約は、NOT NULL のみです。ただし、ネストした表列のスカラー属性にある列制約の型はすべて指定できます。

CONSTRAINT

制約名を指定します。Oracle は、整合性制約の定義とこの制約名をデータ・ディクショナリに格納します。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。

列定義に NULL も NOT NULL も指定しない場合、デフォルト値は NULL になります。

制限事項 : 次の場合を除いて、ユーザー定義オブジェクト型、LOB 型または REF 型を持つ列または属性に対して、制約を作成することはできません。

- ユーザー定義オブジェクト型、VARRAY および LOB の列または属性に対して NOT NULL 制約を指定できます。
- REF 型の列に NOT NULL および参照整合性制約を指定できます。

UNIQUE

UNIQUE を指定して、列または列の組合せを一意キーとして指定します。一意制約を満たす場合、表の中の 2 つの行が一意キーに対して同じ値を持つことはできません。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。

複合一意キーは、列の組合せからなる一意キーです。複合一意キーを定義する場合、*column_constraint* 構文ではなく *table_constraint* 構文を使用してください。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。

制限事項：

- 複合一意キーを指定する制約を満たす場合、表の中の 2 つの行が複合キー列に対して同じ組合せの値を持つことはできません。
- 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。キーの全体サイズ（バイト単位）は、すべての索引列の幅と索引列の数を足した値を超えてはいけません。
- 一意キーの列のデータ型は、LONG または LONG RAW であってはいけません。
- 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。

PRIMARY KEY

PRIMARY KEY を指定して、列または列の組合せを表の主キーとして指定します。**複合主キー**は、列の組合せからなる主キーです。複合主キーを定義する場合、*column_constraint* 構文ではなく、*table_constraint* 構文を使用してください。

制限事項：

- 表には、主キーを 1 つのみ指定できます。
- 主キー列は、LONG、LONG RAW、VARRAY、ネストした表、OBJECT、LOB、BFILE または REF データ型を持つことができません。
- 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。
- 索引構成表の主キーのサイズは、データベース・ブロック・サイズの半分または 3800 バイトのいずれか小さい方のサイズを超えてはいけません（索引構成表には、主キーが必要です）。

- 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。キーの全体サイズ（バイト単位）は、すべての索引列の幅と索引列の数を足した値を超えてはいけません。
- 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。

NULL | NOT NULL

列に NULL が存在するかどうかを示します。*table_constraint* 構文ではなく、*column_constraint* 構文を使用して、NULL および NOT NULL を指定する必要があります。

NULL	列に NULL 値を入れることができる場合は、NULL を指定します。NULL キーワードは、実際には、整合性制約を定義しません。NOT NULL または NULL のどちらも指定しない場合、デフォルトでは列に NULL を入れることができます。
NOT NULL	列に NULL 値を入れることができない場合は、NOT NULL を指定します。この制約を満たす場合、表の中のすべての行がその列に対して値を持つ必要があります。

制限事項 : 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 : 8-153 ページの「属性レベル制約の例」を参照してください。

参照整合性制約

参照整合性制約は、外部キーとして列または列の組合せを指定し、その外部キーと、**参照キー**といわれる指定した主キーまたは一意キーの間の関連を設定します。外部キーを持つ表を**子表**といい、参照キーを持つ表を**親表**といいます。外部キーと参照キーを、同一の表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。

- 表レベルからは、*table_constraint* 構文の *foreign_key_clause* を使用して、参照整合性を指定します。この構文で、列の組合せからなる**複合外部キー**を指定できます。
- 列レベルからは *column_constraint* の REFERENCES 句を使用して、1 つの列からなる外部キーの参照整合性制約を指定します。

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 つの表の中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。

参照整合性制約の制限事項 :

- 外部キーは、LONG 型または LONG RAW 型にはできません。
- 親表上で参照される一意制約または主キー制約は、あらかじめ定義しておく必要があります。
- 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。
- AS *subquery* 句を含む CREATE TABLE 文には、参照整合性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 :** 『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

foreign_key_clause

foreign_key_clause によって、表レベルから列または列の組合せを外部キーとして指定できます。この構文を使用して、複合外部キーを定義する必要があります。

複合キーを含む参照整合性制約を満たすためには、外部キー列の値が親表の行の参照キー列の値と一致するか、または、少なくとも 1 つ以上の外部キー列の値が NULL である必要があります。

制限事項 :

- 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。キーの全体サイズ（バイト単位）は、すべての索引列の幅と索引列の数を足した値を超えてはいけません。
- 複合外部キーは、複合一意キーまたは複合主キーを参照する必要があります。

REFERENCES　REFERENCES 句によって、現在の列または属性を外部キーとして指定し、親表および参照キーを構成する列または列の組合せを指定できます。親表のみを指定して、列名を指定しない場合、外部キーは自動的に親表の主キーを参照します。参照キー列は、外部キー列と同じ数とデータ型で構成されている必要があります。

ON DELETE　ON DELETE 句によって、参照主キーまたは一意キーを削除した場合、参照整合性をメンテナンスする方法が自動的に判断できます。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。

CHECK 制約

CHECK 句によって、表にある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、条件にある列名によって、その行の列値が参照されます。

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して設計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。Oracle は、CHECK 条件が相互に排他的かどうかについては検証しません。

> **参照：** その他の詳細および構文については、5-15 ページの「条件」を参照してください。

制限事項：

- 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 制約条件は、次の構造を持つことができません。
  - 別の行の値を参照する問合せ
  - 関数 SYSDATE、UID、USER または USERENV へのコール
  - 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

table_ref_constraint および *column_ref_constraint*

table_ref 制約および *column_ref* 制約によって、REF 型の列について詳細に指定できます。これらの句の違いは、表レベルから *table_ref_constraint* を指定することのみです。したがって、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。REF 列または属性を識別した後、*column_ref_constraint* を指定します。どちらの制約でも、SCOPE 制約、WITH ROWID 制約または参照整合性制約を指定できます。

標準表制約および列制約については、表レベルの参照整合性制約の FOREIGN KEY 構文および列レベルの参照整合性制約の REFERENCES 構文を使用します。

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 および 8-141 ページの「参照整合性制約」の詳細は、『Oracle8*i* 概要』を参照してください。

ref_column 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 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。

SCOPE 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_name* に制限します。REF 列または属性の値は *scope_table_name* 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 列ごとに有効範囲表を 1 つのみ指定できます。

制限事項 :

- 表が空白の場合にのみ、SCOPE 制約を既存の列に追加できます。
- VARRAY 列の REF 要素に対して SCOPE は指定できません。
- AS *subquery* を指定し、この副問合せがユーザー定義 REF を戻す場合、この句を指定する必要があります。
- *scope_table_name* が自スキーマ内にあるか、または *scope_table_name* 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 列からは SCOPE 表制約を削除できません。

WITH ROWID

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、さらに多くの領域が必要になります。デフォルトの REF 値の記憶域では、ROWID は格納されません。

制限事項 :

- VARRAY 列の REF 要素に対して、WITH ROWID 制約は指定できません。
- REF 列から WITH ROWID 表制約は削除できません。
- 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。

references_clause

references_clause によって、REF 列の参照整合性制約を指定できます。また、この句によって参照表の REF 列または属性の範囲が暗黙的に制限されます。

CONSTRAINT を指定しない場合、Oracle によって制約に対するシステム名が生成されます。

制限事項：

- 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。
- 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、システムが SCOPE 制約を追加します。したがって、SCOPE 制約に適用されるすべての制限はこの場合にも適用されます。
- 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。

DEFERRABLE | NOT DEFERRABLE

DEFERRABLE を指定すると、SET CONSTRAINT(S) 文を使用して、制約のチェックをトランザクションの終わりまで遅らせることができます。

参照：

- 各 DML 文の後の制約チェックについては、11-120 ページの「SET CONSTRAINT[S]」を参照してください。
- 遅延制約の詳細は、『Oracle8*i* 管理者ガイド』および『Oracle8*i* 概要』を参照してください。

NOT DEFERRABLE を指定すると、この制約は各 DML 文の終わりでチェックされます。DEFERRABLE と NOT DEFERRABLE のどちらも指定しない場合、デフォルト値は NOT DEFERRABLE です。

INITIALLY IMMEDIATE	INITIALLY IMMEDIATE を指定すると、各トランザクションの開始時に、デフォルトでは、各 DML 文の終わりにこの制約がチェックされます。INITIALLY を指定しない場合、デフォルト値は INITIALLY IMMEDIATE になります。
INITIALLY DEFERRED	INITIALLY DEFERRED を指定すると、この制約が DEFERRABLE であり、デフォルトでは各トランザクションの終了時にのみ制約がチェックされます。

制限事項 :

- SET CONSTRAINT(S) 文でも、NOT DEFERRABLE 制約は遅延できません。
- 既存の制約を直接変更する（ALTER TABLE ... MODIFY 制約文を指定する）場合は、DEFERRABLE または NOT DEFERRABLE のどちらも指定できません。
- 制約の遅延可能状態は変更できません。制約を削除してからそれを再作成する必要があります。

RELY | NORELY

RELY および NORELY パラメータは、問合せのリライトを行うアカウントに NOVALIDATE モードの制約を適用するかどうかを指定します。RELY を指定すると、適用されていない問合せのリライトに対する 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・ビューおよび問合せのリライトにのみ有効です。QUERY_REWRITE_INTEGRITY モード（7-101 ページの「ALTER SESSION」を参照）に従って、問合せのリライトは、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。

> **参照 :** マテリアライズド・ビューおよび問合せリライトの詳細は、『Oracle8*i* データ・ウェアハウス』を参照してください。

制限事項 :

- RELY および NORELY は、既存の制約を変更する（ALTER TABLE ... MODIFY 制約文が発行された）場合にのみ適用されます。
- RELY には、NOT NULL 制約を設定できません。

using_index_clause

using_index_clause によって、一意制約または主キー制約を使用可能にするために使用する索引のパラメータを指定できます。索引名は制約名と同じです。

索引には、INITRANS、MAXTRANS、TABLESPACE、STORAGE および PCTFREE の各パラメータ値を選択できます。

表がパーティション化されている場合、一意キー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。

制限事項 : 一意制約と主キー制約を使用可能にする場合は、この句のみを使用します。

参照：

- これらのパラメータの詳細は、10-7 ページの「CREATE TABLE」を参照してください。
- LOCAL および *global_index_clause* の詳細、索引に関する NOSORT および LOGGING|NOLOGGING の詳細は、9-51 ページの「CREATE INDEX」を参照してください。

NOSORT

NOSORT によって、データベースに行が昇順で格納されているため、索引を作成する場合に行をソートする必要がないことを指定します。

ENABLE

制約を表のすべての新しいデータに適用させる場合は、ENABLE を指定します。なお、参照整合性制約を使用可能にする前に、その参照先の制約を使用可能にしておく必要があります。

- ENABLE VALIDATE によって、すべての旧データも制約に従うことを指定します。制約が使用可能で、妥当性チェック済であれば、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。

  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、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 マークを付けます（最適な結果を得るためには、列にデータを挿入する前に行ってください）。

- ENABLE NOVALIDATE によって、制約データのすべての新規 DML 操作が制約に従うことは保証されますが、表の既存データが制約に従うかどうかは保証されません。

  主キー制約または一意キー制約を使用可能にした場合、制約を適用する一意索引が自動的に作成されます。その後で制約が使用禁止になった場合、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。この動作を回避するには、主キーと一意キー制約を最初は使用禁止にして作成してください。その後、一意でない索引を作成するか、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。

参照： その他の注意および制限事項については、10-41 ページの「CREATE TABLE」の *enable_disable_clause* を参照してください。

DISABLE

整合性制約を使用禁止にする場合は、DISABLE を指定します。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制約の索引を削除しますが、制約は使用可能のままです。この機能はたいへん便利です。データ・ウェアハウスでは、一意キーに重複しない範囲の値を持つ一定量のデータを、レンジ・パーティション表にロードする必要があります。このような場合、制約が使用禁止で、妥当性チェック済であれば、索引を持たないことによって領域を節約できます。ALTER TABLE 文の *exchange_partition_clause* 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。

  一意キーがパーティション表のパーティション・キーと一致する場合、制約を使用禁止にすることによって、オーバーヘッドを減らすことができ、有害な影響もなくなります。一意キーがパーティション・キーと一致しない場合は、制約の妥当性チェックを行うための変換中に自動テーブル・スキャンが実行され、これによって索引なしでロードするメリットがオフセットされてしまいます。

- 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：** この設定の使用については、『Oracle8*i* パフォーマンスのための設計およびチューニング』を参照してください。

- VALIDATE も NOVALIDATE も指定しない場合、NOVALIDATE がデフォルト値になります。

- 一意索引を使用している一意キー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。

EXCEPTIONS INTO

EXCEPTIONS INTO 句によって、制約の整合性が違反するすべての行の ROWID を格納した表を指定できます。*schema* を指定しない場合、自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。例外表は、ローカル・データベース内にある必要があります。

EXCEPTIONS INTO 句は、制約の妥当性チェックを行うときにのみ有効です。

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。

制限事項 : 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。

注意 : 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。

参照 :

- これらのスクリプトを使用する場合の互換性については、『Oracle8*i* 移行ガイド』を参照してください。
- SQL スクリプトの詳細は、『Oracle8*i* PL/SQL パッケージ・プロシージャ 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、『Oracle8*i* パフォーマンスのための設計およびチューニング』を参照してください。

例

一意キーの例　次の文は、dept 表を作成し、dname 列に対して一意キーを定義して使用可能にします。

```
CREATE TABLE dept
    (deptno  NUMBER(2),
     dname   VARCHAR2(9)  CONSTRAINT unq_dname UNIQUE,
      loc     VARCHAR2(10) );
```

unq_dname 制約は、dname 列を一意キーとして識別します。この制約によって、表中では部門名に同じ名前を付けることはできません。ただし、名前のない部門があってもかまいません。

また、*table_constraint* 構文を使用して、制約を定義し使用可能にできます。

```
CREATE TABLE dept
    (deptno  NUMBER(2),
     dname   VARCHAR2(9),
     loc     VARCHAR2(10),
        CONSTRAINT unq_dname
        UNIQUE (dname)
    USING INDEX PCTFREE 20
        TABLESPACE user_x
        STORAGE (INITIAL 8K  NEXT 6K) );
```

前述の文では、USING INDEX 句を使用して、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も指定しています。

複合一意キーの例　次の文は、census 表の city 列と state 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 census
    ADD CONSTRAINT unq_city_state
    UNIQUE (city, state)
    USING INDEX PCTFREE 5
        TABLESPACE user_y
    EXCEPTIONS INTO bad_keys_in_ship_cont;
```

unq_city_state 制約によって、city と state の値として、同じ組合せ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 句を指定した場合、制約に違反している census 表の行に関する情報は、bad_keys_in_ship_cont 表に書き込まれます。

主キーの例　次の文は、dept 表を作成し、deptno 列に対して主キーを定義して、使用可能にします。

```
CREATE TABLE dept
    (deptno  NUMBER(2) CONSTRAINT pk_dept PRIMARY KEY,
     dname   VARCHAR2(9),
     loc     VARCHAR2(10) );
```

pk_dept 制約は、deptno 列を dept 表の主キーとして指定します。この制約によって、表の中の複数の部門が同一の部門番号を持つことはなく、かつ部門番号が NULL にならないことが保証されます。

次の *table_constraint* 構文を使用しても、この制約を定義し、使用可能にできます。

```
CREATE TABLE dept
    (deptno  NUMBER(2),
    dname   VARCHAR2(9),
    loc   VARCHAR2(10),
    CONSTRAINT pk_dept PRIMARY KEY (deptno) );
```

複合主キーの例　次の文は、ship_cont 表の ship_no 列と container_no 列を組み合せて複合主キーを定義します。

```
ALTER TABLE ship_cont
    ADD PRIMARY KEY (ship_no, container_no) DISABLE;
```

この制約は、ship_no 列と container_no 列の組合せを ship_cont 表の主キーとして指定します。この制約によって、表の中の 2 つの行が ship_no 列と container_no 列の両方に対して同じ値を持たないことが保証されます。

この CONSTRAINT 句では、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。

NOT NULL の例　次の文は、emp 表を変更し、SAL 列に NOT NULL 制約を定義して、使用可能にします。

```
ALTER TABLE emp
   MODIFY (sal  NUMBER  CONSTRAINT nn_sal NOT NULL);
```

nn_sal を指定した場合、表の中の従業員の給与が NULL 値になることはありません。

属性レベル制約の例　次の文では、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));
```

参照整合性制約の例　次の文では、emp 表を作成し、dept 表の deptno 列の主キーを参照する deptno 列に外部キーを定義し、使用可能にします。

```
CREATE TABLE emp
   (empno      NUMBER(4),
    ename      VARCHAR2(10),
    job        VARCHAR2(9),
    mgr        NUMBER(4),
    hiredate   DATE,
    sal        NUMBER(7,2),
    comm       NUMBER(7,2),
    deptno     CONSTRAINT fk_deptno REFERENCES dept(deptno) );
```

この fk_deptno 制約によって、emp 表の従業員が属しているすべての部門が dept 表にあることになります。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。すべての従業員が部門に割り当てられたことを保証するには、emp 表の deptno 列に対し NOT NULL 制約を作成し、さらに REFERENCES 制約も作成します。

この制約を定義して使用可能にする前に、dept 表の deptno 列を主キーとして指定する制約を定義し、使用可能にしておく必要があります。

なお、参照整合性制約の定義では、外部キーを構成する列を指定するために FOREIGN KEY キーワードは使用しません。この制約は、deptno 列の *column_constraint* 句によって定義されるため、外部キーは自動的に deptno 列に設定されます。

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。

前述の文では、deptno 列のデータ型は指定されていません。この列は外部キーであるため、外部キーとして参照する dept.deptno 列のデータ型が自動的に割り当てられます。

また、*table_constraint* 構文を使用しても、参照整合性制約を定義できます。

```
CREATE TABLE emp
  (empno     NUMBER(4),
   ename     VARCHAR2(10),
   job       VARCHAR2(9),
   mgr       NUMBER(4),
   hiredate  DATE,
   sal       NUMBER(7,2),
   comm      NUMBER(7,2),
   deptno,
   CONSTRAINT fk_deptno
      FOREIGN  KEY (deptno)
      REFERENCES  dept(deptno) );
```

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、emp 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 emp
  (empno    NUMBER(4) PRIMARY KEY,
   ename    VARCHAR2(10),
   job      VARCHAR2(9),
   mgr      NUMBER(4) CONSTRAINT fk_mgr
            REFERENCES emp ON DELETE SET NULL,
   hiredate DATE,
   sal      NUMBER(7,2),
   comm     NUMBER(7,2),
   deptno   NUMBER(2)   CONSTRAINT fk_deptno
            REFERENCES dept(deptno)
            ON DELETE CASCADE );
```

最初の ON DELETE 句によって、上司の従業員番号 2332 が emp 表から削除された場合、上司の従業員番号が 2332 である、emp 表のすべての従業員の mgr 列に NULL 値が設定されます。

次の ON DELETE 句によって、dept 表の deptno 値が削除されると、これに依存する emp 表の行の deptno 値も同時に削除されます。たとえば、部門番号 20 が dept 表から削除されると、Oracle はその部門の従業員を emp 表から削除します。

複合参照整合性制約の例　次の文は、phone_calls 表の areaco 列と phoneno 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 phone_calls
    ADD CONSTRAINT fk_areaco_phoneno
        FOREIGN KEY (areaco, phoneno)
        REFERENCES customers(areaco, phoneno)
        EXCEPTIONS INTO wrong_numbers;
```

fk_areaco_phoneno 制約によって、phone_calls 表のすべての電話番号は、必ず customers 表内の電話番号で構成されます。この制約を定義して使用可能にする前に、主キーまたは一意キーとして、customers 表の areaco 列と phoneno 列の組合せを指定する制約をあらかじめ定義し、使用可能にしておく必要があります。

EXCEPTIONS INTO 句によって、phone_calls 表の制約に違反している行に関する情報が、wrong_numbers 表に書き込まれます。

CHECK 制約の例　次の文は、dept 表を作成し、その表の各列に check 制約を定義します。

```
CREATE TABLE dept
   (deptno NUMBER  CONSTRAINT check_deptno
         CHECK (deptno BETWEEN 10 AND 99)
         DISABLE,
    dname VARCHAR2(9)  CONSTRAINT check_dname
         CHECK (dname = UPPER(dname))
         DISABLE,
    loc VARCHAR2(10)  CONSTRAINT check_loc
         CHECK (loc IN ('DALLAS','BOSTON',
         'NEW YORK','CHICAGO'))
         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_deptno によって、部門番号は必ず 10 ～ 99 の範囲内になります。
- check_dname によって、部門名はすべて大文字になります。
- check_loc によって、部門の所在地は Dallas、Boston、New York または Chicago 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、Oracle は、これらの制約を定義するのみで使用可能にはしません。

次の文は、emp 表を作成し、*table_constraint_clause* を使用して、CHECK 制約を定義し、使用可能にします。

```
CREATE TABLE emp
    (empno          NUMBER(4),
     ename          VARCHAR2(10),
     job            VARCHAR2(9),
     mgr            NUMBER(4),
     hiredate       DATE,
     sal            NUMBER(7,2),
     comm           NUMBER(7,2),
     deptno         NUMBER(2),
     CHECK (sal + comm <= 5000) );
```

この制約は、不等式の条件を使用して、従業員の給与の合計（給与と歩合の合計金額）を $5000 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、これらの値の合計金額が $5000 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 CONSTRAINT 句には制約名が指定されていないため、Oracle が制約に対して名前を生成します。

次の文は、主キー制約、2 つの参照整合性制約、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 scott.order (order_id),
   part_no            NUMBER
      CONSTRAINT fk_pno REFERENCES scott.part (part_no),
   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、scott のスキーマ内の order 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、scott.order.order_id 列にあらかじめ存在する必要があります。
- fk_pno は、scott のスキーマ内の part 表にある part_no 列を参照する外部キーとして、part_no 列を指定します。order_detail.part_no 列に追加されるすべての新しい値は、scott.part.part_no 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、CONSTRAINT 句と列定義について、次の点も具体的に示しています。

- *Table_constraint* 構文と列定義は、任意の順序で指定できます。この例では、pk_od 制約を定義する *table_constraint* 構文が列定義より先に指定されています。
- 列定義には、*column_constraint* 構文を複数回使用できます。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、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。

DEFERRABLE 制約の例　次の文は、scores 列に対して NOT DEFERRABLE INITIALLY IMMEDIATE のチェック制約を指定し、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 orders
  (ord_num NUMBER CONSTRAINT unq_num UNIQUE (ord_num)
   INITIALLY DEFERRED DEFERRABLE);
```

9

SQL 文 : CREATE CLUSTER ～ CREATE SEQUENCE

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 FUNCTION
- CREATE INDEX
- CREATE INDEXTYPE
- CREATE JAVA
- CREATE LIBRARY
- CREATE MATERIALIZED VIEW
- CREATE MATERIALIZED VIEW LOG
- CREATE OPERATOR
- CREATE OUTLINE
- CREATE PACKAGE
- CREATE PACKAGE BODY

- CREATE PROCEDURE
- CREATE PROFILE
- CREATE ROLE
- CREATE ROLLBACK SEGMENT
- CREATE SCHEMA
- CREATE SEQUENCE

CREATE CLUSTER

用途

CREATE CLUSTER 文は、クラスタを作成する場合に使用します。**クラスタ**とは、1 つ以上の列を共有する、1 つ以上の表のデータで構成されるスキーマ・オブジェクトです。同一のクラスタ・キーを共有する（すべての表の）すべての行がまとめて格納されます。

既存のクラスタの詳細は、データ・ディクショナリ USER_CLUSTERS、ALL_CLUSTERS および DBA_CLUSTERS に問い合せます。

参照：

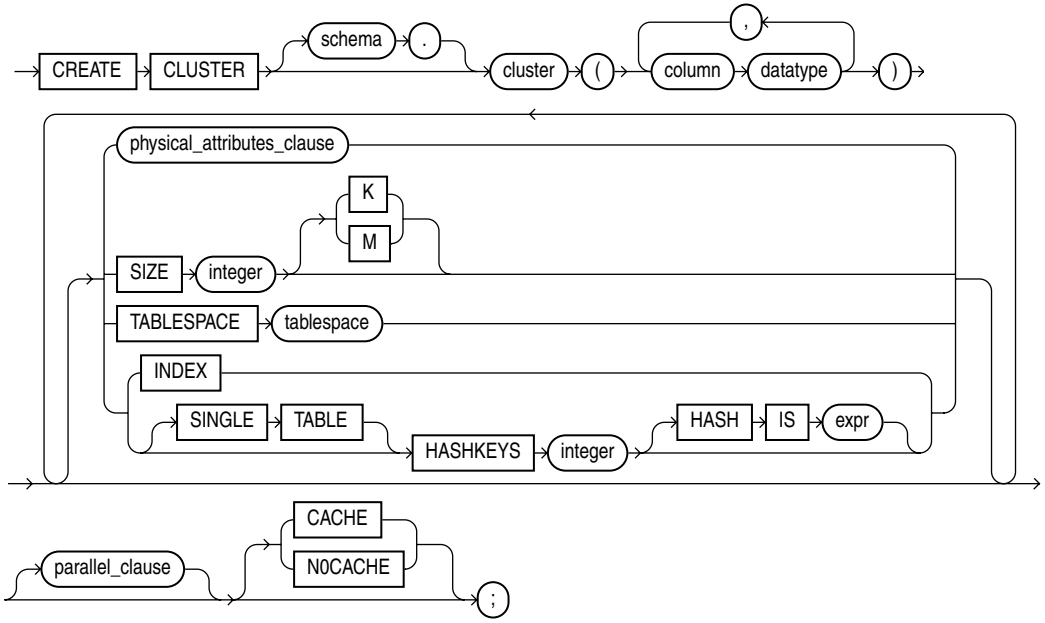
- クラスタの概要については、『Oracle8*i* 概要』を参照してください。
- クラスタのパフォーマンスについては、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。
- クラスタの使用方法については、『Oracle8*i* パフォーマンスのための設計およびチューニング』を参照してください。
- データ・ディクショナリ・ビューの詳細は、『Oracle8*i* リファレンス・マニュアル』を参照してください。

前提条件

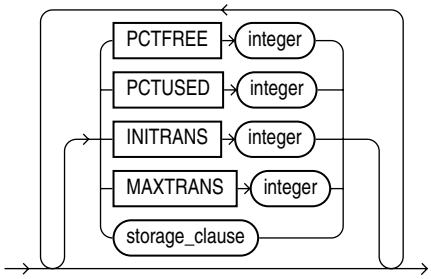
自スキーマにクラスタを作成する場合は、CREATE CLUSTER システム権限が必要です。他のユーザーのスキーマ内にクラスタを作成する場合は、CREATE ANY CLUSTER システム権限が必要です。また、クラスタを設定するスキーマの所有者は、クラスタが定義されている表領域に対する領域の割当て制限、または UNLIMITED TABLESPACE システム権限のいずれかが必要です。

Oracle では、クラスタを最初に作成する場合は、クラスタに対する索引は自動的に作成されません。このため、クラスタ索引を作成するまでは、クラスタ化表に対して、データ操作言語（DML）文を発行できません。

構文

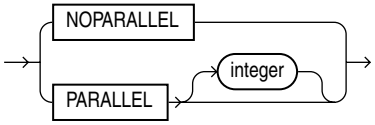


physical_attributes_clause::=



storage_clause: 11-129 ページの「storage_clause」を参照してください。

parallel_clause::=



キーワードとパラメータ

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じように、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、10-7 ページの「CREATE TABLE」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 個までクラスタ・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。

datatype

各クラスタ・キー列のデータ型を指定します。

制限事項：

- データ型が LONG、LONG RAW、REF、ネストした表、VARRAY、BLOB、CLOB、BFILE 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。
- 列のデータ型が、位取りが 0 である INTEGER 型や NUMBER 型でない場合、HASH IS 句は使用できません。
- 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列 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。

physical_attributes_clause

physical_attributes_clause によって、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。

PCTUSED　クラスタのデータ・ブロックに対して行を追加できる時期を決定するために、Oracle が使用するしきい値を指定します。このパラメータの値は、整数値で表現され、パーセントとして解析されます。

PCTFREE　将来的な拡張に備えて、クラスタ内の各データ・ブロックに確保される空き領域を指定します。このパラメータの値は、整数値で表現され、パーセントとして解析されます。

INITRANS　クラスタに属しているデータ・ブロックに対して割り当てられた、同時実行更新トランザクションの初期値を指定します。クラスタに対するこのパラメータに、2 より小さい値や MAXTRANS パラメータの値より大きい値は指定できません。デフォルト値は、クラスタの表領域に対する INITRANS の値と 2 のどちらか大きい方の値になります。

MAXTRANS　クラスタに属している任意のデータ・ブロックに対して、同時実行更新トランザクションの最大数を指定します。このパラメータに、INITRANS パラメータの値より小さい値は指定できません。最大値は 255 です。デフォルト値は、クラスタが含まれる表領域の MAXTRANS 値です。

参照： PCTUSED、PCTFREE、INITRANS および MAXTRANS の各パラメータの詳細は、10-7 ページの「CREATE TABLE」を参照してください。

storage_clause　*storage_clause* によって、データ・ブロックをどのようにクラスタに割り当てるかを指定できます。

参照： 11-129 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の領域を、バイト単位で指定します。K または M を使用すると、この領域を KB または MB 単位で指定できます。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 は、次に大きな約数を使用します。SIZE がデータ・ブロック・サイズより大きい場合、Oracle 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。

Oracle 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（なお、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、この方法はハッシュ・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 | HASH

INDEX

INDEX を指定して、索引クラスタを作成します。**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。

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・キーに索引を付ける必要があります。この索引を**クラスタ索引**と呼びます。

注意 : 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。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。

参照 : クラスタ索引の作成方法については 9-51 ページの「CREATE INDEX」、索引クラスタの概要については『Oracle8*i* 概要』を参照してください。

HASHKEYS

HASHKEYS を指定して、**ハッシュ・クラスタ**の作成およびハッシュ・クラスタのハッシュ値の数を指定します。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関数が戻す値です。

Oracle は、ハッシュ値の実際の数を決めるため、HASHKEYS 値を 1 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 句も HASHKEYS パラメータも指定しない場合、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、Oracle 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 がクラスタに領域を割り当てる方法については、『Oracle8*i* 概要』を参照してください。

SINGLE TABLE

SINGLE TABLE は、クラスタを、表を 1 つのみ持つタイプのハッシュ・クラスタに指定します。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。

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。

HASH IS *expr*

ハッシュ・クラスタに対するハッシュ関数として使用する式を指定します。式は、次の条件を満たす必要があります。

- 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）の数になる場合には、任意のデータ型の列が参照される 1 つ以上の列を持ちます。次に例を示します。

  ```
  NUM_COLUMN * length(VARCHAR2_COLUMN)
  ```

- ユーザー定義の PL/SQL ファンクションを参照できません。
- SYSDATE、USERENV、TO_DATE、UID、USER、LEVEL または ROWNUM を参照できません。
- 定数に評価されることはありません。
- 副問合せは指定できません。
- （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はできません。

HASH IS 句を指定しない場合、ハッシュ・クラスタに対して内部ハッシュ関数が使用されます。

既存のクラスタの詳細は、データ・ディクショナリ表 USER_、ALL_ および DBA_CLUSTER_HASH_EXPRESSIONS に問い合せます。

> **参照 :** データ・ディクショナリ・ビューの詳細は、『Oracle8*i* リファレンス・マニュアル』を参照してください。

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。コンポジット・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、内部ハッシュ関数を使用する必要があります。

parallel_clause

parallel_clause によって、クラスタ作成をパラレル化します。

注意： *parallel_clause* 構文は、以前のリリースの構文に代わるものです。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　シリアル実行を行う場合は、NOPARALLEL を指定します。これはデフォルト値です。

PARALLEL　すべてのパーティション化インスタンスで使用可能な CPU の数に、PARALLEL_THREADS_PER_CPU 初期化パラメータの値を掛けた並列度を選択させる場合は、PARALLEL を指定します。

PARALLEL *integer*　パラレル操作で使用されるパラレル・スレッド数である**並列度**を指定する場合は、*integer* 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。

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、クラスタでその後に行う 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： CREATE TABLE については、10-41 ページの「parallel_clause に関する注意事項」を参照してください。

CACHE | NOCACHE

CACHE　フル・テーブル・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場合は、CACHE を指定します。この句は、小規模な参照表で有効です。

NOCACHE　フル・テーブル・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場合は、NOCACHE を指定します。これはデフォルトの動作です。

注意： 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_number、クラスタ・サイズ 512 バイトおよび記憶領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  NUMBER(2) )
    SIZE 512
    STORAGE (INITIAL 100K NEXT 50K);
```

クラスタへの表の追加例　次の文は、このクラスタに emp 表と dept 表を追加します。

```
CREATE TABLE emp
    (empno     NUMBER        PRIMARY KEY,
     ename     VARCHAR2(10)  NOT NULL
                             CHECK (ename = UPPER(ename)),
     job       VARCHAR2(9),
     mgr       NUMBER        REFERENCES scott.emp(empno),
     hiredate  DATE
        CHECK (hiredate < TO_DATE ('08-14-1998', 'MM-DD-YYYY')),
     sal       NUMBER(10,2)  CHECK (sal > 500),
     comm      NUMBER(9,0)   DEFAULT NULL,
     deptno    NUMBER(2)      NOT NULL )
     CLUSTER personnel (deptno);

CREATE TABLE dept
    (deptno  NUMBER(2),
     dname   VARCHAR2(9),
     loc     VARCHAR2(9))
     CLUSTER personnel (deptno);
```

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は、emp 表と dept 表のどちらにも行を挿入できます。

ハッシュ・クラスタの例　次の文は、クラスタ・キー列 department_number、最大ハッシュ・キー値 500（各サイズ 512 バイト）および記憶領域パラメータ値を指定したハッシュ・クラスタ personnel を作成します。

```
CREATE CLUSTER personnel
( department_number  NUMBER )
    SIZE 512  HASHKEYS 500
    STORAGE (INITIAL 100K  NEXT 50K);
```

この文では、HASH IS 句を指定していないため、Oracle は、そのクラスタに対して内部ハッシュ関数を採用します。

次の文は、home_area_code 列と home_prefix 列で構成されるクラスタ・キーを持つ personnel 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 式をハッシュ関数に使用します。

```
CREATE CLUSTER personnel
    ( home_area_code  NUMBER,
      home_prefix     NUMBER )
    HASHKEYS 20
    HASH IS MOD(home_area_code + home_prefix, 101);
```

単一表ハッシュ・クラスタの例　次の文は、クラスタ・キー deptno および最大ハッシュ・キー値 500（各サイズ 512 バイト）を指定した単一表ハッシュ・クラスタ personnel を作成します。

```
CREATE CLUSTER personnel
   (deptno NUMBER)
   SIZE 512 SINGLE TABLE HASHKEYS 500;
```

CREATE CONTEXT

用途

CREATE CONTEXT は、**コンテキスト**（アプリケーションを検証し、保護するアプリケーション定義属性の集合）にネームスペースを作成し、コンテキストを設定する外部作成パッケージとネームスペースを関連付ける場合に使用します。指定されたパッケージの 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。

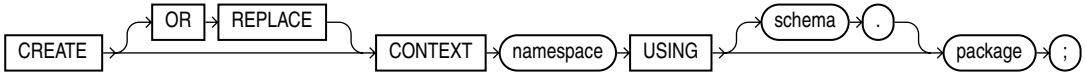
参照 :

- コンテキストの定義および説明については、『Oracle8*i* 概要』を参照してください。
- DBMS_SESSION.set_context プロシージャについては、『Oracle8*i* PL/SQL パッケージ・プロシージャ 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



キーワードとパラメータ

OR REPLACE

OR REPLACE を指定して、異なるパッケージを使用して既存コンテキストを再定義します。

namespace

作成または変更するためのコンテキスト・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、スキーマ SYS に格納されます。

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 は現在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。

注意： 柔軟に設計するために、Oracle 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。

参照： パッケージの設定の詳細は、『Oracle8*i* PL/SQL パッケージ・プロシージャ リファレンス』を参照してください。

例

CREATE CONTEXT の例 人事情報アプリケーション（hr）、およびそれを検証し、保護する PL/SQL パッケージ（hr_secure_context）があるとします。次の文は、コンテキスト・ネームスペース hr_context を作成し、hr_secure_context パッケージに関連付けます。

```
CREATE CONTEXT hr_context USING hr_secure_context;
```

SYS_CONTEXT 関数を使用する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、hr_secure_context パッケージが、特定の編成識別子として属性 org_id を定義していると仮定します。org_id の値に基づき、アクセスを制限するビューを作成して hr_org_unit ベース表を次のように保護できます。

```
CREATE VIEW hr_org_secure_view AS
     SELECT * FROM hr_org_unit
     WHERE organization_id = SYS_CONTEXT('hr_context', 'org_id');
```

参照： SYS_CONTEXT 関数については、4-101 ページの「SYS_CONTEXT」を参照してください。

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取ることをお薦めします。詳細は、『Oracle8*i* バックアップおよびリカバリ・ガイド』を参照してください。

用途

CREATE CONTROLFILE 文は、次の場合に制御ファイルを再作成します。

- 既存の制御ファイルのすべてのコピーが、メディア障害により消失してしまった場合。
- データベース名を変更する場合。
- 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場合。

CREATE CONTROLFILE 文を発行した場合、この文にユーザーが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。句を指定しない場合、Oracle 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、初期化パラメータ PARALLEL_SERVER で指定したモードでデータベースがマウントされます。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。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： 『Oracle8*i* バックアップおよびリカバリ・ガイド』を参照してください。

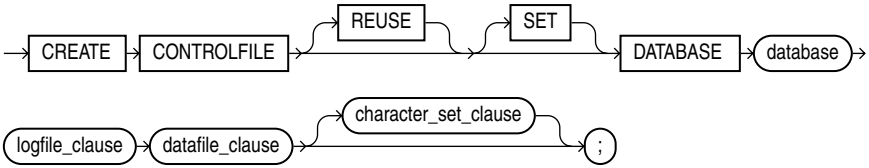
前提条件

OSDBA ロールを使用可能にする必要があります。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。

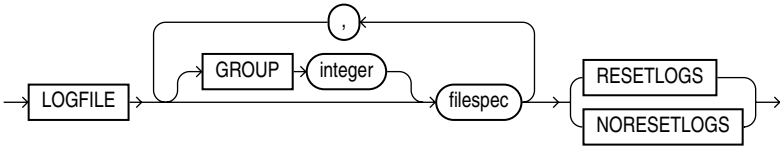
初期化パラメータ REMOTE_LOGIN_PASSWORDFILE が exclusive に設定されている場合、Oracle は、制御ファイルを再作成しようとした際にエラーを戻します。このメッセージを回避するには、制御ファイルを再作成する前に、パラメータに shared を設定するか、またはパスワード・ファイルを再作成します。

参照： REMOTE_LOGIN_PASSWORDFILE パラメータについては、『Oracle8*i* リファレンス・マニュアル』を参照してください。

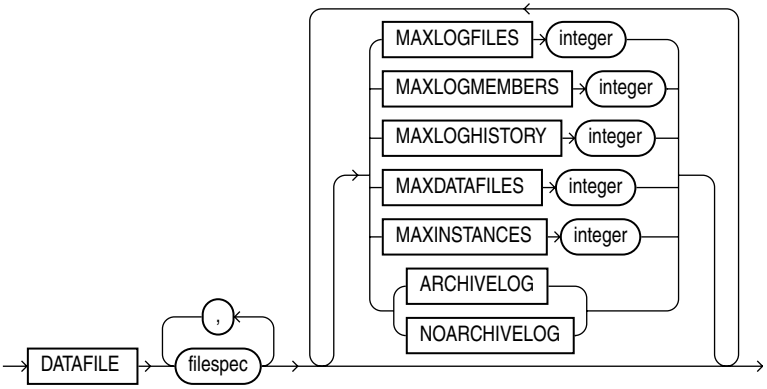
構文



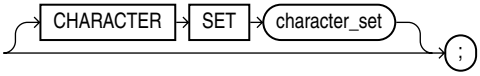
logfile_clause::=



datafile_clause::=



character_set_clause::=



filespec: 11-27 ページの「filespec」を参照してください。

キーワードとパラメータ

REUSE

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使用することを指定します。このとき、現在保存されている情報は無視され、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 *database*

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 *database*

SET DATABASE を使用して、データベース名を変更します。データベース名は最大 8 バイトまで指定可能です。

logfile_clause

LOGFILE *filespec*

データベースに対する REDO ログ・ファイルを指定します。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。

参照 : *filespec* の構文については、11-27 ページの「filespec」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。

RESETLOGS

LOGFILE 句にリストされたファイルのコンテキストを無視する場合は、RESETLOGS を指定します。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。LOGFILE 句の各 *filespec* で、SIZE パラメータを指定する必要があります。Oracle 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 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　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する場合は、NORESETLOGS を指定します。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。

datafile_clause

DATAFILE *filespec*　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。*filespec* の構文については、11-27 ページの「filespec」を参照してください。

MAXLOGFILES *integer*　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。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られる領域が決定されます。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 *integer*　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。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られる領域が決定されます。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。

MAXLOGHISTORY *integer*　Oracle Parallel Server の自動メディア・リカバリでアーカイブ REDO ログ・ファイル・グループ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。このパラメータは、Parallel Server を使用する Oracle をパラレル・モードとアーカイブ・ログ・モードの両方で使用している場合のみに有効です。

MAXDATAFILES *integer*
: 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設定を指定します。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 *integer*
: 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
: ARCHIVELOG を指定して、REDO ログ・ファイルを再使用する前に、ファイルの内容をアーカイブします。この句を指定した場合、インスタンス・リカバリのみでなくメディア・リカバリもできるようになります。

NOARCHIVELOG
: ARCHIVELOG 句および NOARCHIVELOG 句を省略した場合、デフォルトでは NOARCHIVELOG モードが選択されます。制御ファイルの作成後に、ALTER DATABASE 文を使用した場合、ARCHIVELOG モードと NOARCHIVELOG モードを切り替えることができます。

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・セット情報を再構成します。データベースのケース・メディア・リカバリが必要となる場合、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、デフォルトの US7ASCII 以外のキャラクタ・セットを使用している場合にのみ有効です。

制御ファイルを再作成し、表領域のリカバリに Recovery Manager を使用している場合、およびデータ・ディクショナリに格納されている異なるキャラクタ・セットを指定する場合、表領域はリカバリされません（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）。

注意： データベース・キャラクタ・セットは、この句では変更できません。

参照： 表領域のリカバリについては、『Oracle8*i* Recovery Manager ユーザーズ・ガイド』を参照してください。

例

CREATE CONTROLFILE の例　この文は、制御ファイルを再作成します。この文で、データベース orders_2 は F7DEC キャラクタ・セットで作成されます。

```
CREATE CONTROLFILE REUSE
   DATABASE orders_2
   LOGFILE GROUP 1 ('diskb:log1.log', 'diskc:log1.log') SIZE 50K,
           GROUP 2 ('diskb:log2.log', 'diskc:log2.log') SIZE 50K
   NORESETLOGS
   DATAFILE 'diska:dbone.dat' SIZE 2M
                              MAXLOGFILES 5
                              MAXLOGHISTORY 100
                              MAXDATAFILES 10
                              MAXINSTANCES 2
                              ARCHIVELOG
   CHARACTER SET F7DEC;
```

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。

用途

CREATE DATABASE 文は、汎用的なデータベースを作成する場合に使用します。

この文は、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 PARALLEL_SERVER 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。同時に、データベースの表領域およびロールバック・セグメントを作成できます。

参照：

- データベースの変更の詳細は、7-7 ページの「ALTER DATABASE」を参照してください。
- Oracle8*i* Java 仮想マシンの作成の詳細は、『Oracle8*i* Java 開発者ガイド』を参照してください。
- ロールバック・セグメントおよび表領域の作成の詳細は、9-144 ページの「CREATE ROLLBACK SEGMENT」および 10-56 ページの「CREATE TABLESPACE」を参照してください。

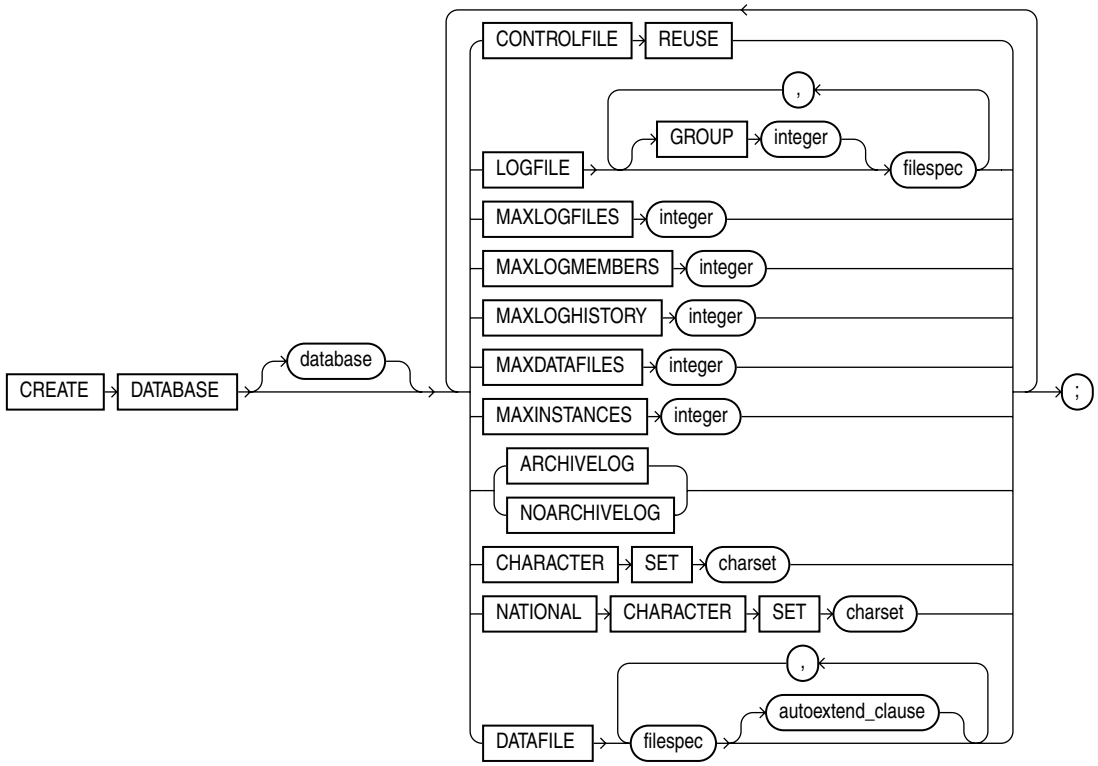
前提条件

OSDBA ロールを使用可能にする必要があります。

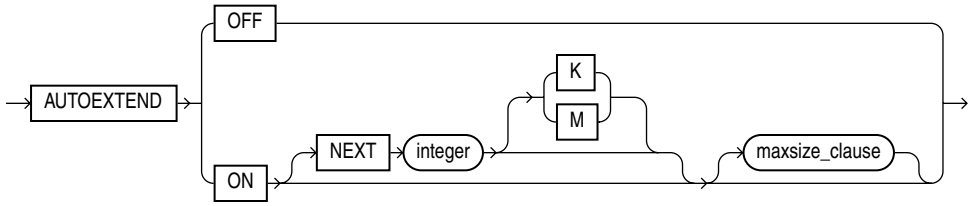
初期化パラメータ REMOTE_LOGIN_PASSWORDFILE が exclusive に設定されている場合、Oracle は、データベースを再作成しようとした際にエラーを戻します。エラーを発生させないためには、データベースを再作成する前にパラメータを shared に設定するか、またはパスワード・ファイルを再作成します。

参照： REMOTE_LOGIN_PASSWORDFILE パラメータの詳細は、『Oracle8*i* リファレンス・マニュアル』を参照してください。

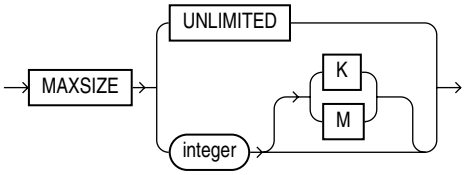
構文



autoextend_clause::=



maxsize_clause::=



filespec: 11-27 ページの「filespec」を参照してください。

キーワードとパラメータ

database

作成するデータベースの名前を、最大 8 バイトで指定します。データベース名には ASCII 文字のみを使用できます。指定したデータベース名は、Oracle によって制御ファイルに記録されます。後で、ALTER DATABASE 文を発行したときにデータベース名を明示的に指定すると、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。

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、初期化パラメータ DB_NAME で指定した名前が採用されます。初期化パラメータ DB_NAME が設定されており、そのパラメータの値とは異なる名前を指定した場合、Oracle はエラーを戻します。

参照： その他の規則については、2-85 ページの「スキーマ・オブジェクト名のネーミング計画」を参照してください。

CONTROLFILE REUSE

CONTROLFILE REUSE を指定して、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使用することを指定します。これによって、現在保存されている情報は無視され、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。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、Oracle はエラー・メッセージを戻します。

LOGFILE *filespec*

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。各 *filespec* 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：** *filespec* の構文については、11-27 ページの「filespec」を参照してください。

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表 V$LOG で調べることができます。

LOGFILE 句を指定しない場合、デフォルトでは 2 つの REDO ログ・ファイル・グループが作成されます。各デフォルト・ファイルの名前およびサイズは、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。

MAXLOGFILES *integer*

データベース用に作成可能な REDO ログ・ファイル・グループの最大数を指定します。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られる領域が決定されます。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 *integer*

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。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られる領域が決定されます。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。

MAXLOGHISTORY *integer*

Oracle Parallel Server での自動メディア・リカバリ用のアーカイブ 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。

注意： このパラメータは、Parallel Server を使用する Oracle をパラレル・モードとアーカイブ・ログ・モードの両方で使用している場合のみに有効です。

MAXDATAFILES *integer*

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設定を指定します。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 *integer*

作成した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
: REDO ログ・ファイル・グループを再使用する前に、グループの内容をアーカイブする場合は、ARCHIVELOG を指定します。この句を指定した場合、メディア・リカバリができるようになります。

NOARCHIVELOG
: REDO ログ・ファイル・グループを再使用する前に、グループの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。この句を指定した場合、メディア・リカバリはできません。

  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 文を使用した場合、ARCHIVELOG モードと NOARCHIVELOG モードを切り替えることができます。

CHARACTER SET *character_set*

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。

制限事項 : データベース・キャラクタ・セットとして、固定幅マルチバイト・キャラクタ・セットは指定できません。

> **参照 :** キャラクタ・セットの詳細は、『Oracle8*i* NLS ガイド』を参照してください。

NATIONAL CHARACTER SET *character_set*

データ型が NCHAR、NCLOB または NVARCHAR2 で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。指定しない場合は、デフォルトとしてデータベース・キャラクタ・セットが各国語キャラクタ・セットとして設定されます。

> **参照 :** 有効なキャラクタ・セット名の詳細は、『Oracle8*i* NLS ガイド』を参照してください。

DATAFILE *filespec*

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。この句を省略した場合、デフォルトで、1 つのデータ・ファイルが作成されます。デフォルトのファイルの名前およびサイズは、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。

注意 : SYSTEM 表領域に割り当てる初期領域は、全体で 5MB 以上になるようにしてください。

> **参照 :** 構文の詳細は、11-27 ページの「filespec」を参照してください。

autoextend_clause

autoextend_clause によって、データ・ファイルの自動拡張機能を使用可能または使用禁止にできます。この句を指定しないと、データ・ファイルは自動的に拡張されません。

OFF　OFF を指定すると、自動拡張が使用禁止になります。NEXT および MAXSIZE は 0（ゼロ）に設定されます。NEXT および MAXSIZE の値は、ALTER DATABASE AUTOEXTEND 文または ALTER TABLESPACE AUTOEXTEND 文で再指定する必要があります。

ON　ON を指定すると、自動拡張が使用可能になります。

NEXT *integer*　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（バイト単位）を指定します。K または M を使用すると、KB または MB 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　データ・ファイルの自動拡張で使用される最大ディスク領域のサイズを指定します。

- *integer* は、最大ディスク領域をバイトで指定します。K または M を使用すると、KB または MB 単位で指定できます。
- UNLIMITED は、データ・ファイルへのディスク領域割当てを無制限にします。

例

CREATE DATABASE の例　次の文は、すべての引数にデフォルト値を使用して小規模なデータベースを作成します。

```
CREATE DATABASE;
```

次の文は、すべての各引数を完全に指定してデータベースを作成します。

```
CREATE DATABASE newtest
    CONTROLFILE REUSE
    LOGFILE
       GROUP 1 ('diskb:log1.log', 'diskc:log1.log') SIZE 50K,
       GROUP 2 ('diskb:log2.log', 'diskc:log2.log') SIZE 50K
    MAXLOGFILES 5
    MAXLOGHISTORY 100
    DATAFILE 'diska:dbone.dat' SIZE 2M
    MAXDATAFILES 10
    MAXINSTANCES 2
    ARCHIVELOG
    CHARACTER SET US7ASCII
    NATIONAL CHARACTER SET JA16SJISFIXED
    DATAFILE
    'disk1:df1.dbf' AUTOEXTEND ON
    'disk2:df2.dbf' AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED;
```

CREATE DATABASE LINK

用途

CREATE DATABASE LINK 文は、データベース・リンクを作成する場合に使用します。**データベース・リンク**とは、リモート・データベース上のオブジェクトにアクセスできるローカル・データベース上のスキーマ・オブジェクトです。リモート・データベースは、Oracle データベースである必要はありません。

データベース・リンクを作成した場合、そのリンクを利用して、リモート・データベース上の表およびビューを参照できます。また、表名またはビュー名に *@dblink* を付けることによって、SQL 文の中でリモート表またはビューを参照できます。SELECT 文を使用すると、リモート表またはビューの問合せができます。Oracle 分散オプション付きで使用する場合は、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、『Oracle8*i* アプリケーション開発者ガイド 基礎編』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、『Oracle8*i* 分散システム』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、『Oracle8*i* リファレンス・マニュアル』を参照してください。
- 既存のデータベース・リンクの削除については、10-127 ページの「DROP DATABASE LINK」を参照してください。
- DML 操作でのリンクの使用については、11-52 ページの「INSERT」、11-141 ページの「UPDATE」、10-114 ページの「DELETE」および 11-62 ページの「LOCK TABLE」を参照してください。

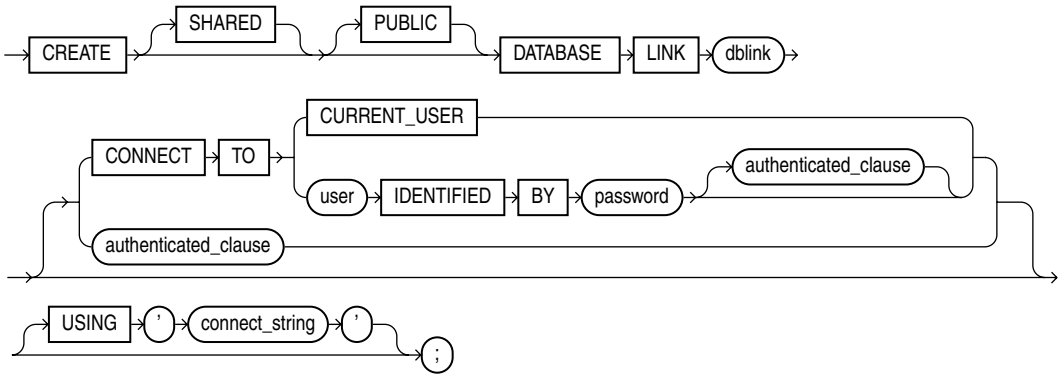
前提条件

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は、CREATE PUBLIC DATABASE LINK システム権限が必要です。また、リモートの Oracle データベースに対する CREATE SESSION 権限が必要です。

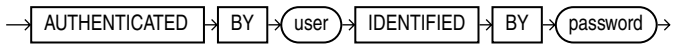
なお、ローカルとリモートの両方の Oracle データベースに、Net8 をインストールしておく必要があります。

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。

構文



authenticated_clause::=



キーワードとパラメータ

SHARED

SHARED を指定して、1 つのネットワーク接続を使用することによって、複数ユーザーで共有できるパブリック・データベース・リンクを作成します。この句は、マルチスレッド・サーバー構成でのみ使用できます。

> **参照：** 共有データベース・リンクの詳細は、『Oracle8*i* 分散システム』を参照してください。

PUBLIC

PUBLIC を指定して、すべてのユーザーが使用可能なパブリック・データベース・リンクを作成します。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。

参照： 9-33 ページの「PUBLIC データベース・リンクの例」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。GLOBAL_NAMES 初期パラメータの値は、データベース・リンクが接続するデータベースと同じ名前を持つ必要があるかどうかを指定します。

Oracle Parallel Server 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 初期パラメータの値で決まります。

制限事項：他のユーザーのスキーマでデータベース・リンクを作成することはできません。また、スキーマ名によって *dblink* を修飾することはできません（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、2-88 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 初期パラメータについては、『Oracle8*i* リファレンス・マニュアル』を参照してください。

CONNECT TO

CONNECT TO によって、リモート・データベースへの接続が可能になります。

CURRENT_USER　CURRENT_USER を指定して、**カレント・ユーザー・データベース・リンク**を作成します。正常にリンクを作成する場合、カレント・ユーザーが、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・リンクが直接使用される場合（ストアド・オブジェクト内から使用されていない場合）、カレント・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、ビューまたはトリガーなど）を実行する場合、CURRENT_USER とは、ストアド・オブジェクトを所有するユーザー名であり、オブジェクトをコールしたユーザー名ではありません。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、カレント・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については、9-42 ページの「CREATE FUNCTION」を参照してください。
- 9-32 ページの「CURRENT_USER の例」も参照してください。

user IDENTIFIED BY *password*

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します（固定ユーザー・データベース・リンク）。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（接続ユーザー・データベース・リンク）。

参照 : 9-32 ページの「固定ユーザーの例」を参照してください。

authenticated_clause

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。

SHARED 句を使用している場合は、必ずこの句を指定してください。

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。

参照 : リモート・データベースの指定については、『Oracle8*i* Net8 管理者ガイド』を参照してください。

例

CURRENT_USER の例　次の文は、カレント・ユーザーのデータベース・リンクを定義します。

```
CREATE DATABASE LINK sales.hq.acme.com
   CONNECT TO CURRENT_USER
   USING 'sales';
```

固定ユーザーの例　次の文では、sales.hq.acme.com という名前の固定ユーザーのデータベース・リンクを定義します。

```
CREATE DATABASE LINK sales.hq.acme.com
   CONNECT TO scott IDENTIFIED BY tiger
   USING 'sales';
```

データベース・リンクが作成された後は、次の方法でリモート・データベース上のスキーマ scott 内の表に問い合せることができます。

```
SELECT *
   FROM emp@sales.hq.acme.com;
```

次のように DML 文を使用して、リモート・データベース上のデータを変更できます。

```
INSERT INTO accounts@sales.hq.acme.com(acc_no, acc_name, balance)
   VALUES (5001, 'BOWER', 2000);

UPDATE accounts@sales.hq.acme.com
   SET balance = balance + 500;

DELETE FROM accounts@sales.hq.acme.com
   WHERE acc_name = 'BOWER';
```

また、同一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。次の文は、scott が adam の dept 表へのアクセス権限を持っていることを前提としています。

```
SELECT *
   FROM adams.dept@sales.hq.acme.com;
```

この文では、リモート・データベースのユーザー scott に接続してから、adam の dept 表への問合せが行われます。

scott の emp 表がリモート・データベース上にあることを隠すために、シノニムを作成できます。次の文は、emp を参照すると、必ず、scott が所有するリモート emp 表にアクセスするようになります。

```
CREATE SYNONYM emp
   FOR scott.emp@sales.hq.acme.com;
```

PUBLIC データベース・リンクの例　次の文では、sales.hq.acme.com という名前の共有パブリック固定ユーザーのデータベース・リンクを定義します。このデータベース・リンクは、文字列サービス名 'sales' で指定したデータベースに対して、ユーザー scott（パスワード tiger）と対応しています。

```
CREATE SHARED PUBLIC DATABASE LINK sales.hq.acme.com
   CONNECT TO scott IDENTIFIED BY tiger
   AUTHENTICATED BY anupam IDENTIFIED BY bhide
   USING 'sales';
```

CREATE DIMENSION

用途

CREATE DIMENSION 文は、**ディメンション**を作成する場合に使用します。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（またはレベル）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問合せのリライトを行います。サマリー・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 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 *join_clause* で指定され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。このプロシージャの詳細は、『Oracle8*i* PL/SQL パッケージ・プロシージャ リファレンス』を参照してください。

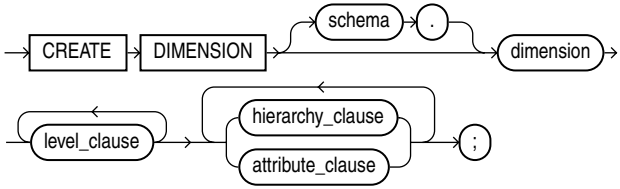
参照：

- マテリアライズド・ビューの詳細は、9-86 ページの「CREATE MATERIALIZED VIEW」を参照してください。
- 問合せのリライト、オプティマイザおよびサマリー・アドバイザの詳細は、『Oracle8*i* データ・ウェアハウス』を参照してください。

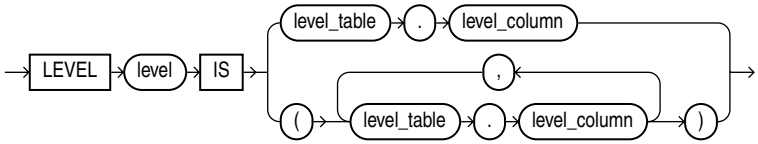
前提条件

自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。

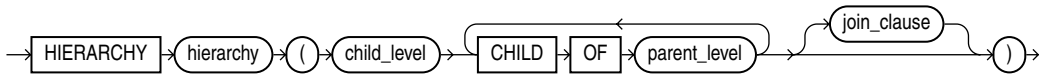
構文



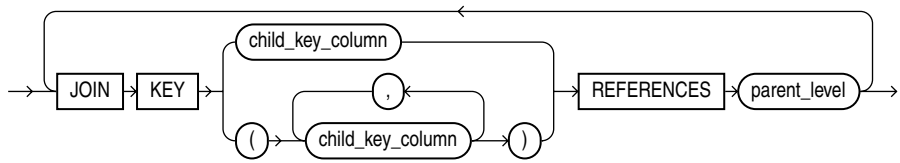
level_clause::=



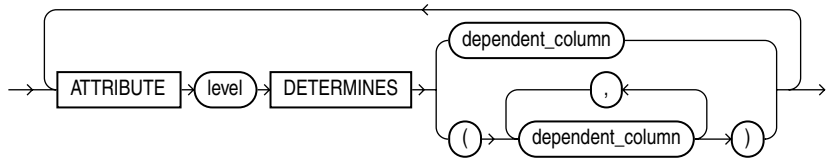
hierarchy_clause::=



join_clause::=



attribute_clause::=



キーワードとパラメータ

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。

制限事項：

- 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、*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。
- 各 *level_column* は、NULL 以外である必要があります（ただし、この列は NOT NULL 制約を受ける必要はありません）。

hierarchy_clause

ディメンションの線形階層レベルを定義します。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名を指定します。*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、*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。

join_clause

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限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。

制限事項 :

- *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。
- 内部結合はできません（*child_key_columns* は、*parent_level* として同じ表に置くことはできません）。

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* の子を含むスキーマおよび表と一致している必要があります。

制限事項 :

- 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。
- 同じ階層の既存のレベルの組に対して、1 つの *join_clause* のみを指定できます。

parent_level レベル名を指定します。

attribute_clause

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* として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。

例

CREATE DIMENSION の例　次の文は、time_tab 表における time ディメンションと city 表、state 表および country 表における geog ディメンションを作成します。

```
CREATE DIMENSION time
   LEVEL curDate         IS time_tab.curDate
   LEVEL month           IS time_tab.month
   LEVEL qtr             IS time_tab.qtr
   LEVEL year            IS time_tab.year
   LEVEL fiscal_week     IS time_tab.fiscal_week
   LEVEL fiscal_qtr      IS time_tab.fiscal_qtr
   LEVEL fiscal_year     IS time_tab.fiscal_year
   HIERARCHY month_rollup (
      curDate            CHILD OF
      month              CHILD OF
      qtr                CHILD OF
      year)
   HIERARCHY fiscal_year_rollup (
      curDate            CHILD OF
      fiscal_week        CHILD OF
      fiscal_qtr         CHILD OF
      fiscal_year )
   ATTRIBUTE curDate     DETERMINES (holiday, dayOfWeek)
   ATTRIBUTE month       DETERMINES (yr_ago_month, qtr_ago_month)
   ATTRIBUTE fiscal_qtr  DETERMINES yr_ago_qtr
   ATTRIBUTE year        DETERMINES yr_ago ;

CREATE DIMENSION geog
   LEVEL cityID         IS (city.city, city.state)
   LEVEL stateID        IS state.state
   LEVEL countryID      IS country.country
   HIERARCHY political_rollup (
      cityID            CHILD OF
      stateID           CHILD OF
      countryID
         JOIN KEY city.state REFERENCES stateID
         JOIN KEY state.country REFERENCES countryID);
```

CREATE DIRECTORY

用途

CREATE DIRECTORY 文は、ディレクトリ・オブジェクトを作成する場合に使用します。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）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、オペレーティング・システムのパス名をハードコード化せずにディレクトリ名を使用できます。このため、ファイル管理の汎用性が向上します。

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、2-15 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、11-31 ページの「GRANT」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した場合、READ 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 READ 権限を付与できます。DBA も、この権限を他のユーザーおよびロールに付与できます。

また、ファイル保存用として、対応するオペレーティング・システムのディレクトリも作成する必要があります。各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 プロセスに対する読取り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ます。このため、これら 2 つは正確に対応しない場合があります。たとえば、ユーザー scott に、ディレクトリ・スキーマ・オブジェクトに対する読取り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・システム上のディレクトリに Oracle プロセスに対する読取り権限が付与されていない場合には、エラーが発生します。

構文

キーワードとパラメータ

OR REPLACE

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する場合は、OR REPLACE を指定します。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、10-130 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。

> **注意：** 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・システムで使用するパス名が大文字と小文字を区別する場合は、必ず、正しい形式でディレクトリ名を指定してください（ただし、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）。

'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、パス名の大文字と小文字は区別されます。

例

CREATE DIRECTORY の例　次の文は、オペレーティング・システムのディレクトリ /private1/lob/files 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 '/private1/LOB/files';
```

CREATE FUNCTION

用途

CREATE FUNCTION 文は、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する場合に使用します（また、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）。

ストアド・ファンクション（**ユーザー・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、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 3 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 メソッドまたは共有ライブラリの名前付きファンクションを問い合せます。引数および戻り値に対する型変換も問い合せます。

参照：

- プロシージャおよびファンクションの概要は、9-127 ページの「CREATE PROCEDURE」を参照してください。
- ファンクション作成の例については、9-49 ページの「例」を参照してください。
- パッケージの作成については、9-118 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については、7-36 ページの「ALTER FUNCTION」を参照してください。
- 共有ライブラリについては、9-84 ページの「CREATE LIBRARY」を参照してください。
- スタンドアロン・ファンクションの削除については、10-131 ページの「DROP FUNCTION」を参照してください。
- 外部ファンクションの登録については、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

前提条件

ストアド・ファンクションを作成する前に、ユーザー SYS は SQL スクリプト DBMSSTDX.SQL を実行する必要があります。このスクリプトの正確な名前および格納位置は、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。

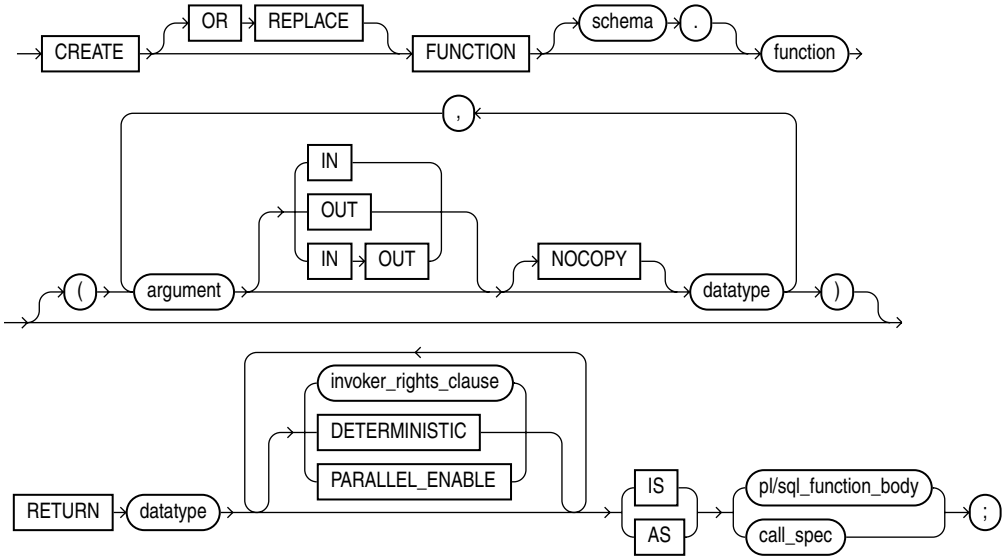
自スキーマ内にディメンションを作成する場合は、CREATE PROCEDURE システム権限が必要です。他のユーザーのスキーマ内にファンクションを作成する場合は、CREATE ANY PROCEDURE システム権限が必要です。他のユーザーのスキーマ内のファンクションを置換する場合は、ALTER ANY PROCEDURE システム権限が必要です。

コール仕様を起動する場合、追加権限が必要となることがあります（たとえば、C コール仕様の C ライブラリに対する EXECUTE 権限）。

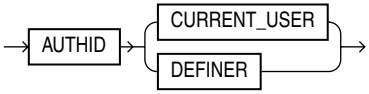
Oracle プリコンパイラ・プログラム内に CREATE FUNCTION 文を埋め込む場合、キーワード END-EXEC に続けて、各言語の埋込み SQL 文の終了記号を記述して文を終了する必要があります。

> **参照：** このような前提条件の詳細は、『Oracle8*i* PL/SQL ユーザーズ・ガイドおよびリファレンス』または『Oracle8*i* Java ストアド・プロシージャ開発者ガイド』を参照してください。

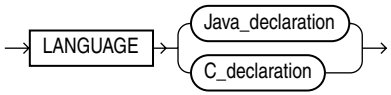
構文



invoker_rights_clause::=

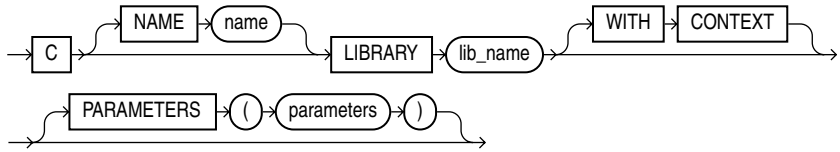


call_spec::=



Java_declaration::=

C_declaration::=



キーワードとパラメータ

OR REPLACE

既存のファンクションを再作成する場合は、OR REPLACE を指定します。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は、索引に DISABLED のマークを付けます。

> **参照：** ファンクションの再コンパイルについては、7-36 ページの「ALTER FUNCTION」を参照してください。

schema

ファンクションを定義するスキーマを指定します。*schema* を指定しない場合、現行スキーマにファンクションが作成されます。

function

作成するファンクションの名前を指定します。コンパイル・エラーでファンクション結果を作成した場合、Oracle はエラーを戻します。SHOW ERRORS コマンドを使用すると、関連するコンパイラ・エラー・メッセージを表示できます。

ユーザー定義ファンクションの制限事項

ユーザー定義ファンクションは、未変更定義を必要とする状況では使用できません。このため、ユーザー定義ファンクションは、次の場所では使用できません。

- CREATE TABLE 文または ALTER TABLE 文の CHECK 制約句
- CREATE TABLE 文または ALTER TABLE 文の DEFAULT 句

また、ファンクションが問合せまたは DML 文内からコールされる場合、そのファンクションには次の制限があります。

- OUT パラメータまたは IN OUT パラメータは指定できません。
- 現行のトランザクションのコミットやロールバック、セーブポイントの作成やロールバックまたはセッションやシステムの変更はできません。DDL 文が明示的に現行のトランザクションをコミットするため、ユーザー定義ファンクションは DDL 文を実行できません。
- ファンクションが SELECT 文からコールされる場合のデータベースへの書込みはできません。ただし、DML 文で副問合せからコールされるファンクションは、データベースへの書込みができます。
- ファンクションが DML 文からコールされる場合、ファンクションがコールされる文で修正される同じ表への書込みはできません。

OUT パラメータおよび IN OUT パラメータの制限を除き、Oracle は、SQL 文から直接コールされるファンクションのみでなく、そのファンクションがコールするすべてのファンクションや、そのファンクションまたはファンクションがコールすることによって実行される SQL 文からコールされるファンクションに対しても、これらの制限を適用します。

argument

ファンクションへの引数の名前を指定します。ファンクションが引数を受け入れない場合は、ファンクション名の後のカッコを省略できます。

IN　　IN は、ファンクションをコールするときに、引数に値を指定する必要があることを示します。これはデフォルト値です。

OUT　　OUT は、ファンクションによって引数の値が設定されることを示します。

IN OUT　　IN OUT は、引数の値を、ユーザーが指定することも、ファンクションで設定することも可能であることを示します。

NOCOPY　　NOCOPY は、できるだけ速く引数を渡すように指示します。この句は、OUT パラメータや IN OUT パラメータに対して、レコード、索引付き表、VARRAY などの大きい値を渡す際のパフォーマンスの向上に有効です（IN パラメータ値には、常に NOCOPY が渡されます）。

- NOCOPY パラメータを指定すると、このパラメータに対応する実際の割当てとしてパッケージ変数が渡された場合に、パッケージ変数に対して行われた割当ては、すぐにこのパラメータに表示されます（または、このパラメータに対して行われた割当ては、すぐにパッケージ変数に表示されます）。
- このパラメータまたは別のパラメータに対して行われた変更は、同じ変数が両方に渡された場合、両方の名前を介してすぐに参照できます。
- ファンクションが未処理例外で終了した場合、このパラメータに対するすべての割当てはコール元の変数で参照できます。

このような効果がないコールもあります。この効果に問題がない場合にのみ NOCOPY を使用してください。

datatype　　引数のデータ型を指定します。引数には、PL/SQL でサポートされるデータ型を指定できます。

データ型には、データ長、精度および位取りを指定できません。Oracle では、そのファンクションがコールされた環境から引数のデータ長、精度および位取りを導出します。

RETURN

datatype　　ファンクションの戻り値のデータ型を指定します。すべてのファンクションが必ず値を戻すため、この句の指定は必須です。戻り値は、PL/SQL でサポートされているデータ型を持つことができます。

データ型には、データ長、精度および位取りを指定できません。Oracle では、そのファンクションがコールされた環境から戻り値のデータ長、精度および位取りを導出します。

参照 : PL/SQL のデータ型については、『Oracle8*i* PL/SQL 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。

invoker_rights_clause

invoker_rights_clause によって、ファンクションを、スキーマを所有するユーザーの権限でそのスキーマ内で実行するか、または CURRENT_USER の権限でそのスキーマ内で実行するかを指定できます。

この句は、問合せ、DML 操作およびファンクションにおける動的 SQL 文の外部名の変換方法も定義します。

AUTHID CURRENT_USER	ファンクションを CURRENT_USER 権限で実行する場合は、CURRENT_USER を指定します。この句は、実行者権限ファンクションを作成します。 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。他のすべての文の外部名は、ファンクションを含むスキーマで変換します。
AUTHID DEFINER	ファンクションを含むスキーマの所有者権限でファンクションを実行する場合、およびファンクションを含むスキーマで外部名を変換する場合は、DEFINER を指定します。これはデフォルト値です。

参照 :

- CURRENT_USER の判断方法については、『Oracle8*i* 概要』および『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。
- 『Oracle8*i* PL/SQL ユーザーズ・ガイドおよびリファレンス』も参照してください。

DETERMINISTIC

DETERMINISTIC は、ファンクションが戻した結果が保存されたコピー（このようなコピーが使用可能な場合）をシステムが使用できるようにするための最適化のヒントです。保存されたコピーは、マテリアライズド・ビュー、ファンクション索引、または同じ SQL 文内の同じファンクションに対する別の呼出しから得ることができます。問合せオプティマイザは、保存されたコピーを使用するか、またはファンクションを再コールするかを選択できます。

ファンクションは、同じ値の引数でコールされると、必ず同じ結果を示します。したがって、システムがファンクションをコールしないことを選択した場合、結果が得られなくなります。このため、ファンクションが戻した結果に影響するような方法で、パッケージ変数の使用や、データベースへアクセスを行うファンクションは定義しないでください。

ファンクションは、ファンクション索引の式、またはマテリアライズド・ビューが REFRESH FAST または ENABLE QUERY REWRITE でマークされている場合のビューの問合せからコールされるように、DETERMINISTIC を宣言する必要があります。

参照：

- マテリアライズド・ビューについては、『Oracle8*i* データ・ウェアハウス』を参照してください。
- ファンクション索引については、9-51 ページの「CREATE INDEX」を参照してください。

PARALLEL_ENABLE

PARALLEL_ENABLE は、パラレル問合せ操作のパラレル実行サーバーからファンクションを実行するための最適化ヒントを指定します。パッケージ変数などの変数は、パラレル実行サーバー内で共有されないことがあるため、ファンクションはそのようなセッション状態を使用してはいけません。

参照：『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

IS | AS

pl/sql_subprogram_body

PL/SQL サブプログラム本体のファンクションを宣言します。

参照：PL/SQL サブプログラムの詳細は、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

call_spec　*call_spec* によって、Java または C メソッド名、パラメータ・タイプおよび戻り型を SQL で相当するものにマップできます。
Java_declaration では、'*string*' が JAVA 実装メソッドを定義します。

参照：

- 『Oracle8*i* Java ストアド・プロシージャ開発者ガイド』を参照してください。
- パラメータおよび *C_declaration* のセマンティックについては、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

AS EXTERNAL　AS EXTERNAL は、C メソッドを宣言するもう 1 つの方法です。この句は以前のリリースのもので、下位互換用にのみサポートされています。AS LANGUAGE C 構文を使用することをお薦めします。

例

CREATE FUNCTION の例　次の文は、ファンクション get_bal を作成します。

```
CREATE FUNCTION get_bal(acc_no IN NUMBER)
   RETURN NUMBER
   IS acc_bal NUMBER(11,2);
   BEGIN
      SELECT balance
      INTO acc_bal
      FROM accounts
      WHERE account_id = acc_no;
      RETURN(acc_bal);
    END;
```

get_bal ファンクションを実行した場合、指定された預金口座の残高が戻ります。

このファンクションをコールする際は、残高を確認する口座番号 acc_no を引数として指定する必要があります。acc_no のデータ型は NUMBER です。

このファンクションでは、口座の残高が戻ります。CREATE FUNCTION 文の RETURN 句は、戻り値のデータ型が NUMBER であることを示しています。

このファンクションでは、SELECT 文によって、accounts 表の引数 acc_no で特定される行から balance 列が選択されます。また、RETURN 文によって、ファンクションがコールされる環境にこの値が戻ります。

この例で作成されたファンクションは、SQL 文の中で使用できます。たとえば、次のように割り当てることができます。

```
SELECT get_bal(100) FROM DUAL;
```

次の文では、C ルーチン c_get_val を外部ファンクションとして登録する PL/SQL スタンドアロン・ファンクション get_val が作成されます（この例では、パラメータは省略されています）。

```
CREATE FUNCTION get_val
    ( x_val IN NUMBER,
    y_val IN NUMBER,
    image IN LONG RAW )
    RETURN BINARY_INTEGER AS LANGUAGE C
       NAME "c_get_val"
       LIBRARY c_utils
       PARAMETERS (...);
```

CREATE INDEX

用途

CREATE INDEX 文は、次の索引を作成する場合に使用します。

- 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 は、次の索引をサポートしています。

- 従来（B* ツリー）索引
- **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が入る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。その式にはファンクション（組込みまたはユーザー定義）が含まれることがあります。
- **ドメイン・インデックス**は、アプリケーション固有の索引タイプのインスタンスです。

参照 :

- 索引については、『Oracle8*i* 概要』を参照してください。
- 索引の変更については、7-38 ページの「ALTER INDEX」を参照してください。
- 索引の削除については、10-133 ページの「DROP INDEX」を参照してください。

前提条件

自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 権限がある。
- CREATE ANY INDEX システム権限がある。

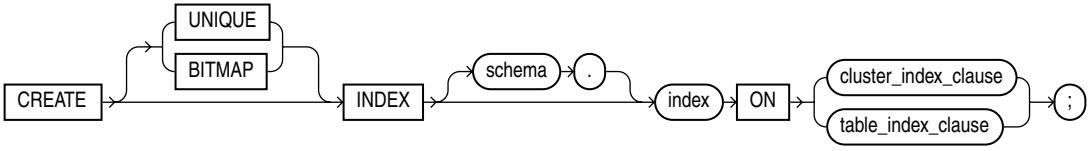
他のユーザーのスキーマ内に索引を作成する場合は、CREATE ANY INDEX システム権限が必要です。また、索引が定義されているスキーマの所有者には、その索引または索引パーティションを格納するための表領域の割当て制限または UNLIMITED TABLESPACE システム権限のいずれかが必要です。

自スキーマにドメイン・インデックスを作成する場合、従来索引の作成の前提条件の他に、索引タイプについての EXECUTE 権限が必要です。他のユーザーのスキーマにドメイン・インデックスを作成する場合、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ついての EXECUTE 権限が必要です。ドメイン・インデックスを作成する前に、索引タイプを定義する必要があります。

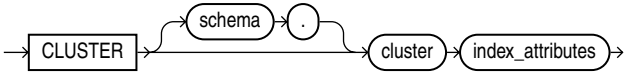
自表の自スキーマにファンクション索引を作成する場合、従来索引の作成の前提条件の他に、QUERY REWRITE システム権限が必要です。別のスキーマまたは別のスキーマの表に索引を作成する場合は、GLOBAL QUERY REWRITE 権限が必要です。どちらの場合も、表の所有者は、ファンクション索引で使用されるファンクションについての EXECUTE オブジェクト権限が必要です。また、問合せでファンクション索引を使用する場合は、QUERY_REWRITE_ENABLED パラメータを true に、QUERY_REWRITE_INTEGRITY パラメータを trusted に設定する必要があります。

参照： 9-75 ページの「CREATE INDEXTYPE」を参照してください。

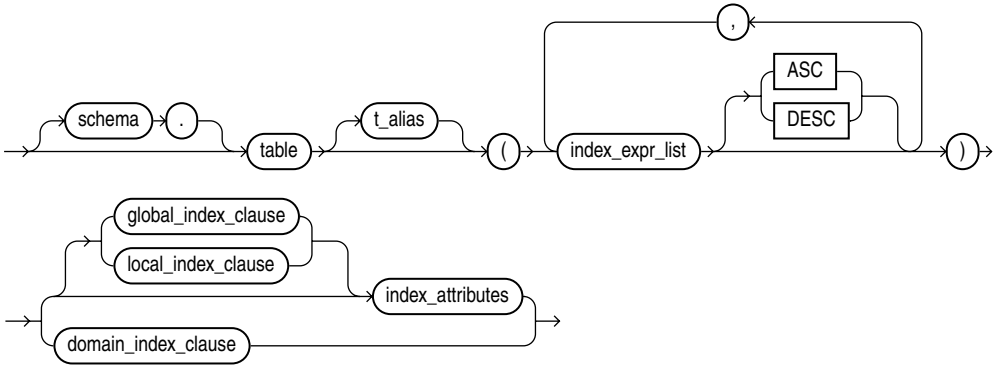
構文



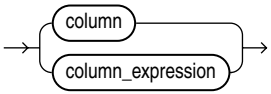
cluster_index_clause::=



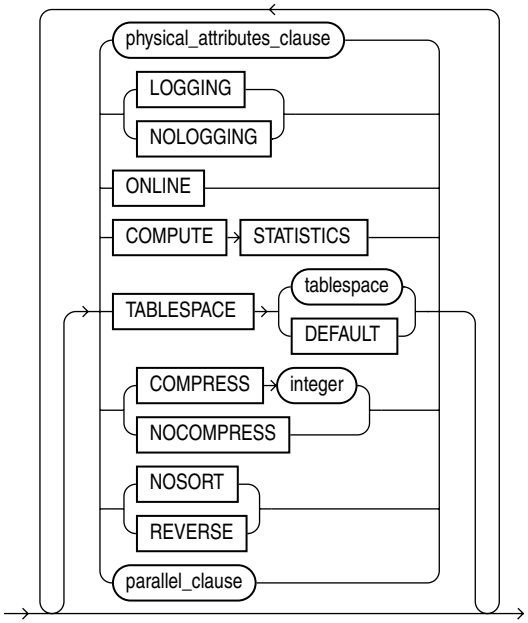
table_index_clause::=



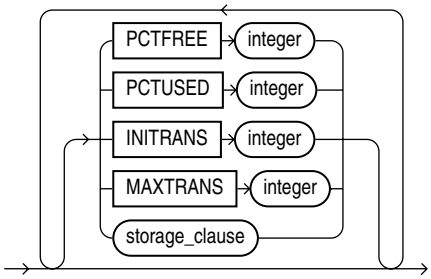
index_expr_list::=



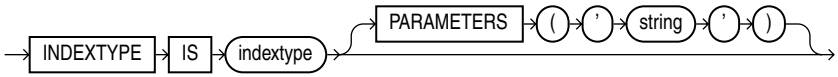
index_attributes::=



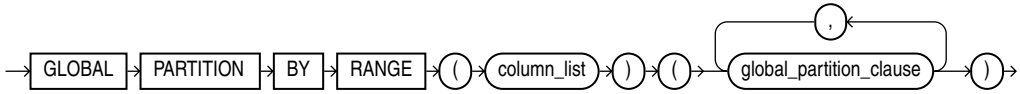
physical_attributes_clause::=



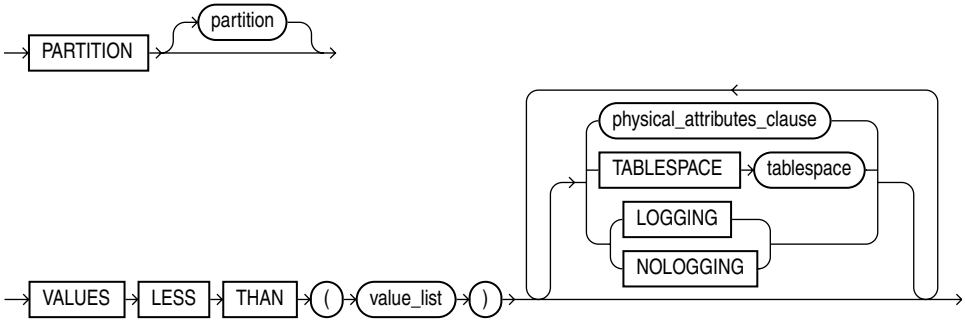
domain_index_clause::=



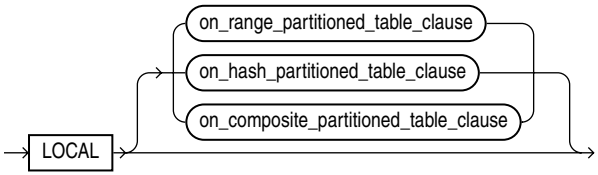
global_index_clause::=



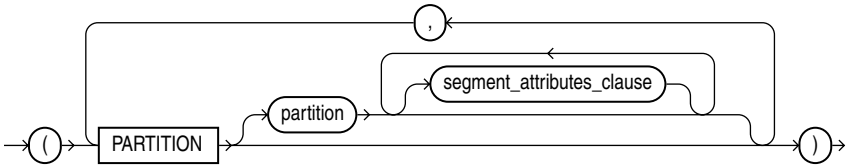
global_partition_clause::=



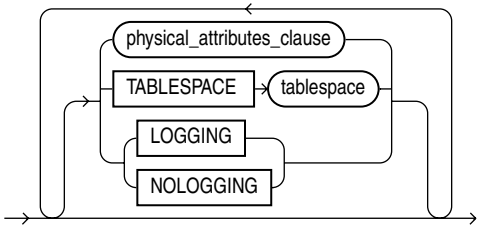
local_index_clauses::=



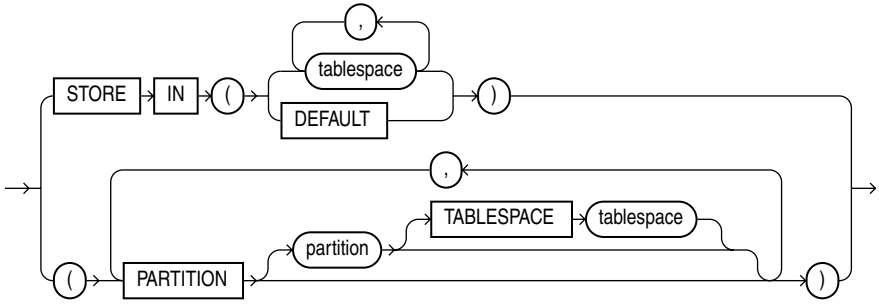
on_range_partitioned_table_clause::=



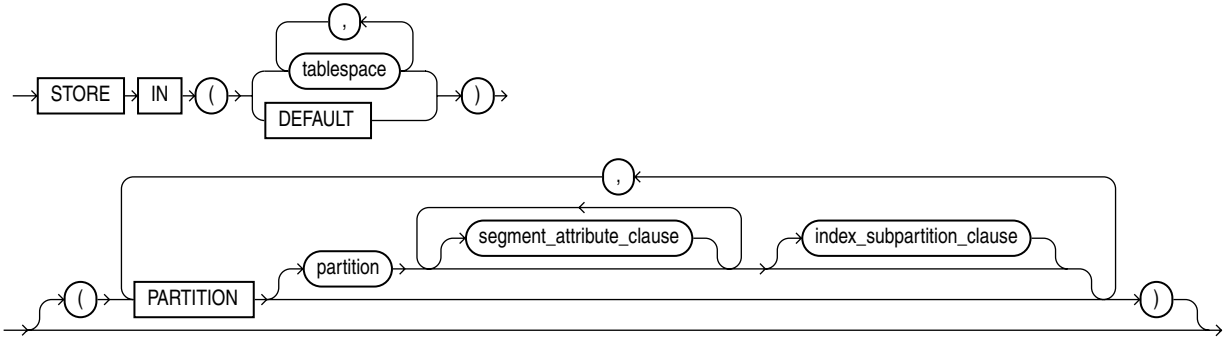
segment_attributes_clause::=



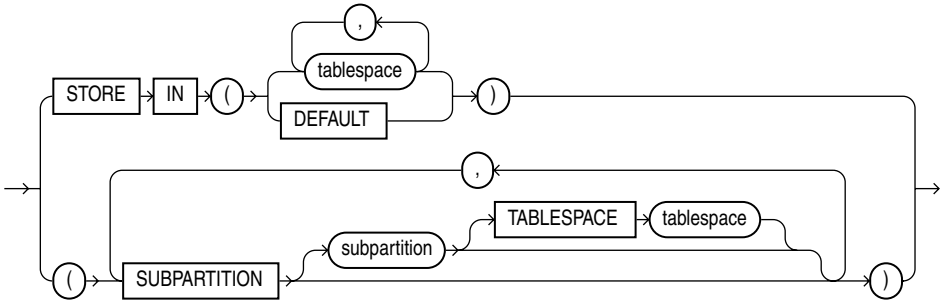
on_hash_partitioned_table_clause::=



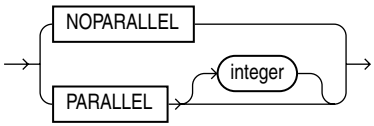
on_composite_partitioned_table_clause::=



index_subpartition_clause::=



parallel_clause::=



storage_clause: 11-129 ページの「storage_clause」を参照してください。

キーワードとパラメータ

UNIQUE

UNIQUE によって、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します。索引がローカル非同一キー索引（次の説明を参照）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。

表には、明示的に UNIQUE 索引を定義しないでください。一意性は非常に論理的な概念であり、表の *definition* と関連付ける必要があります。このため、目的の列に一意整合性制約を定義してください。

制限事項 :

- UNIQUE および BITMAP を同時に指定することはできません。
- ドメイン・インデックスには UNIQUE を指定できません。

> **参照 :** 整合性制約については、8-134 ページの「constraint_clause」を参照してください。

BITMAP

BITMAP によって、*index* を、B ツリーではなくビットマップとして作成することを指定します。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 に対応しているため、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。

制限事項 :

- グローバル・パーティション索引または索引構成表の作成時は、BITMAP を指定できません。
- UNIQUE および BITMAP を同時に指定することはできません。
- ドメイン・インデックスには 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、『Oracle8*i* 概要』および『Oracle8*i* パフォーマンスのための設計およびチューニング』を参照してください。

schema

作成する索引が定義されるスキーマを指定します。*schema* を指定しないと、自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。*index* は、複数のパーティションを持つことができます。

cluster_index_clause

cluster_index_clause を使用して、クラスタ索引が作成されるクラスタを識別します。*cluster* を *schema* で修飾しない場合、そのクラスタが自スキーマ内に定義されているとみなされます。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 :** 9-3 ページの「CREATE CLUSTER」を参照してください。

table_index_clause

索引を定義している表（およびその属性）を指定します。指定する *table* を *schema* で修飾しない場合は、その表が自スキーマに定義されているとみなされます。

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 UNIQUE 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。

制限事項 :

- 索引が LOCAL の場合には、指定する *table* をパーティションで分割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 2 次索引には、BITMAP または 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・サイズの半分です。
- *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。
  - 索引は、パーティション索引またはドメイン・インデックスであってはいけません。
  - *physical_attributes_clause* または *parallel_clause* は指定できません。
  - LOGGING、NOLOGGING または TABLESPACE は指定できません。

    参照 : 一時表の詳細は、10-7 ページの「CREATE TABLE」および『Oracle8*i* 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_expression_list* 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。詳細は、9-72 ページの「Type Method のファンクション索引の例」を参照してください。

index_expr_list

index_expr_list によって、索引に基づく列または列の式を指定できます。

column

表内の列の名前を指定します。ビットマップ索引には最大 30 列まで指定できます。他の索引には最大 32 列まで指定できます。

制限事項 : SCOPE 句で定義されている REF 型列または属性の索引が Oracle でサポートされている場合を除き、ユーザー定義型、LONG 型、LONG RAW 型、LOB 型または REF 型の列または属性の索引は作成できません。

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、1 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。

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の列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。

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、ANALYZE 文を使用して索引とその基になる表の両方に関する統計項目を収集します。この統計項目が生成されるまで、ファンクション索引は使用できません。

参照 : 8-95 ページの「ANALYZE」を参照してください。

ファンクション索引における注意事項 :

- ファンクション索引を使用する表を問い合せる場合、その問合せで *column_expression* が NULL でないことを確認する必要があります。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 : 9-71 ページの「ファンクション索引の例」を参照してください。

- 索引の基になるファンクションが無効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 のマークが付けられ、パラメータ SKIP_UNUSABLE_INDEXES が true に設定された場合を除き、DISABLED 索引の DML 操作は失敗します。

  参照 : このパラメータについては、7-101 ページの「ALTER SESSION」を参照してください。

- ファンクション索引の使用も、QUERY_REWRITE_ENABLED セッション・パラメータの設定による影響を受けます。

  参照 : 7-101 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定されたファンクション、パッケージまたは型への変換を続けます。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 は正常に変換を処理します。

ファンクション索引に関する制限事項 :

- *column_expression* で参照されるユーザー定義ファンクションは、DETERMINISTIC である必要があります。
- 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。

- 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。
- LOB、REF、ネストした表または VARRAY 列にファンクション索引は作成できません。さらに、*column_expression* のファンクションは、LOB、REF、ネストした表または VARRAY 型の属性のオブジェクトを引数とすることはできません。
- *column_expression* に集計関数を含めることはできません。

参照 : 9-42 ページの「CREATE FUNCTION」および『Oracle8*i* PL/SQL 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。

ASC | DESC

ASC または DESC を使用して、索引を昇順で作成するか降順で作成するかを指定します。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 は、降順索引をファンクション索引として扱います。索引作成に QUERY REWRITE または GLOBAL QUERY REWRITE は必要ありません。ただし、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。この文の *column_expression* 句を参照してください。

制限事項 : ドメイン・インデックスにはこれらの句を指定できません。逆順索引には DESC を指定できません。索引がビットマップ化されたり、COMPATIBLE 初期化パラメータが 8.1.0 未満の値に設定されると、DESC は無視されます。

index_attributes

physical_attributes_clause

physical_attributes_clause を使用して、索引における物理特性および記憶特性の値の設定をします。10-7 ページの「CREATE TABLE」を参照してください。

制限事項 : 索引に PCTUSED パラメータは指定できません。

PCTFREE

索引の各データ・ブロック内で、更新および挿入に備えて確保しておく領域の割合（パーセント）を指定します。

storage_clause

storage_clause を使用して、索引における記憶特性を指定します。

参照 : 11-129 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。

COMPRESS

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、キー列値の繰返しがなくなり、記憶域を削減できます。*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。

- 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。

非パーティション索引（一意でない索引または 2 列以上の一意索引）のみを圧縮します。

制限事項 : ビットマップ索引には、COMPRESS を指定できません。

NOCOMPRESS

NOCOMPRESS を指定すると、キー圧縮が使用禁止になります。これはデフォルト値です。

NOSORT

NOSORT を指定すると、行がデータベース内に昇順で格納されます。そのため、Oracle は索引の作成時に行のソートを行う必要がありません。索引列の行または列が昇順に格納されていない場合、Oracle 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。

制限事項：

- 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が逆順に格納されます。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は逆順には格納できません。

LOGGING | NOLOGGING

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。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・ロードの INSERT 操作を記録するか記録しないかも指定します。デフォルト値は LOGGING です。

非パーティション索引の場合、この指定は索引のロギング属性になります。

パーティション索引の場合、指定されたロギング属性は次のようになります。

- 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 または NOLOGGING を指定する場合を除く）
- 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 操作中に明示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

NOLOGGING モードでは、データの変更時に、（新しいエクステントを無効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 データのログへの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、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索引がある場合は、NOLOGGING 操作の後にバックアップを取ってください。

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、索引が再作成されます。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、索引は再作成されません。

索引のロギング属性は、そのベース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 : ロギングおよびパラレル DML の詳細は、『Oracle8*i* 概要』および『Oracle8*i* Parallel Server 概要』を参照してください。

ONLINE

ONLINE によって、索引作成中に表での DML 操作ができることを指定します。

制限事項 : オンラインで索引を作成している間、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 はエラーを戻します。

参照 : オンラインでの索引作成および再作成については、『Oracle8*i* 概要』を参照してください。

COMPUTE STATISTICS

COMPUTE STATISTICS によって、索引作成中の統計情報収集を指定します。これらの統計情報は、SQL 文の実行計画を選択する際に、オプティマイザによって使用中のデータ・ディクショナリに格納されます。

収集された統計情報のタイプは、作成する索引のタイプによって異なります。

注意 : （表のかわりに）別の索引を使用して索引を作成する場合、元の索引は適切な統計情報を提供しない場合があります。このため、一般的に基となる表を使用して統計を計算します。その結果、統計は改善されますが、パフォーマンスが低下することがあります。

PL/SQL パッケージおよびプロシージャでは、その他の方法で統計を収集できます。

参照：『Oracle8*i* PL/SQL パッケージ・プロシージャ リファレンス』を参照してください。

parallel_clause　索引の作成をパラレル化する場合は、*parallel_clause* を指定します。

注意： *parallel_clause* 構文は、以前のリリースの構文に代わるものです。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　シリアル実行を行う場合は、NOPARALLEL を指定します。これはデフォルト値です。

PARALLEL　すべてのパーティション化インスタンスで使用可能な CPU の数に、PARALLEL_THREADS_PER_CPU 初期化パラメータの値を掛けた並列度を選択させる場合は、PARALLEL を指定します。

PARALLEL *integer*　パラレル操作で使用されるパラレル・スレッド数である**並列度**を指定する場合は、*integer* 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。

global_index_clause

global_index_clause によって、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。

PARTITION BY RANGE　PARTITION BY RANGE によって、*column_list* で指定した列の値の範囲でグローバル索引がパーティション化されるように指定します。ローカル索引にこの句は指定できません。

(column_list)　索引のパーティション化を行う表の列名を指定します。*column_list* では、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。

column_list には、最大 32 列まで指定できます。なお、ROWID 疑似列および ROWID 型の列は指定できません。

注意 : 別のキャラクタ・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・セットのサポートについては、『Oracle8*i* NLS ガイド』を参照してください。

PARTITION *partition*

PARTITION 句によって、個々のパーティションを記述できます。句の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*)

グローバル索引の現在のパーティション・バウンドの上限（上限値は含まない）を指定します。*value_list* には、*partition_by_range_clause* の *column_list* と対応するリテラル値を順番にカンマで区切って指定します。最後のパーティションの *value_list* は、必ず MAXVALUE を指定してください。

制限事項 : ローカル索引には、この句を指定できません。

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 NLS 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。NLS 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。

参照 :

これらの初期化パラメータの詳細は、『Oracle8*i* NLS ガイド』を参照してください。

- 10-51 ページの「パーティション表の例」も参照してください。

local_index_clauses

local_index_clauses によって、表と同じ列上で、同じ数のパーティション、同じパーティション・バウンドで、索引をパーティションに分割することを指定します。Oracle は、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を自動的にメンテナンスします。

on_range_partitioned_table_clause

レンジ・パーティション表の索引の名前および属性を指定します。

PARTITION *partition*

個々のパーティションの名前を指定します。句の数によってパーティションの数が決まります。ローカル索引では、索引のパーティション数は表のパーティション数と同じで、表のパーティションと同じ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整合性のある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、SYS_P*n* の形式が使用されます。

on_hash_partitioned_table_clause

ハッシュ・パーティション表の索引の名前および属性を指定します。*partition* を指定しない場合、明示的に指定された別の索引パーティションの名前と競合しない限り、対応する基本の表パーティション名を使用します。この場合、SYS_P*nnn* の形式の名前を生成します。

索引パーティションまたは 1 つ以上の個々のパーティションに対して、任意に TABLESPACE を指定できます。索引またはパーティション・レベルで TABLESPACE を指定しない場合、同じ表領域の各索引パーティションを対応する表のパーティションとして格納します。

on_composite_partitioned_table_clause

コンポジット・パーティション表の索引の名前および属性を指定します。最初の STORE IN 句には、索引のサブパーティションのデフォルトの表領域を指定します。*index_subpartitioning_clause* に異なる表領域を指定して、この記憶域をオーバーライドできます。

この句または *index_subpartitioning_clause* のサブパーティションに TABLESPACE を指定しない場合、*index* に指定された表領域を使用します。*index* に TABLESPACE を指定しない場合、同じ表領域のサブパーティションを、対応する表のサブパーティションとして格納します。

STORE IN

STORE IN 句によって、索引のハッシュ・パーティション（ハッシュ・パーティション索引用）または索引のサブパーティション（コンポジット・パーティション索引用）が複数の表領域に分散される方法を指定できます。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、表領域名を介して循環します。

DEFAULT

DEFAULT 句は、ハッシュ・パーティション表またはコンポジット・パーティション表のローカル索引にのみ有効です。この句は、パーティションまたはサブパーティションの索引レベルで指定された表領域をオーバーライドし、同じパーティションの索引パーティションまたはサブパーティションを、対応する表パーティションまたはサブパーティションとして格納します。

index_subpartition_clause

index_subpartition_clause によって、パーティションにおけるすべてのサブパーティションを格納する 1 つ以上の表領域、またはパーティションにおける 1 つ以上の個々のサブパーティションを指定します。サブパーティションは、パーティションからの他の属性をすべて継承します。パーティションに指定されていない属性は、索引から継承されます。

domain_index_clause

domain_index_clause を使用して、索引がドメイン・インデックスであることを指定します。

制限事項 :

- *index_expr_list* は、単一列のみ指定できます。
- 1 つの列には、1 つのドメイン・インデックスのみを指定できます。
- ビットマップ、一意またはファンクション・ベースのドメイン・インデックスは指定できません。
- パーティション表にはローカル・ドメイン・インデックスを作成できません。
- 列の移動が可能な状態では、パーティション表にドメイン・インデックスを作成できません。

column

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。各 *column* は、列に定義されたドメイン・インデックスを 1 つのみ指定できます。

制限事項 :

- REF データ型、VARRAY、ネストした表、LONG または LONG RAW の列にはドメイン・インデックスを作成できません。
- ユーザー定義型の列ではドメイン・インデックスを作成できますが、属性自体がユーザー定義型の場合、ユーザー定義型の列の属性にはドメイン・インデックスを作成できません。

indextype

索引タイプの名前を指定します。名前は、すでに定義された有効なスキーマ・オブジェクトです。

参照 : 9-75 ページの「CREATE INDEXTYPE」を参照してください。

PARAMETERS '*string*'

未解析のまま適切な索引タイプ・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。

ドメイン・インデックスが作成されると、このルーチンが呼び出されます。ルーチンが正常に戻らない場合、ドメイン・インデックスには FAILED というマークが付けられます。失敗したドメイン・インデックスで可能な操作は、DROP INDEX のみです。

参照 : これらのルーチンについては、『Oracle8*i* 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

PARALLEL の例　次の文は、10 のパラレル実行サーバーを使用して索引を作成します。10 のうち 5 つは scott.emp のスキャン用で、残りの 5 つは索引 emp_idx への移入用です。

```
CREATE INDEX emp_idx
   ON scott.emp (ename)
   PARALLEL 5;
```

COMPRESS の例　次の文は、COMPRESS 句に対して索引を作成します。

```
CREATE INDEX emp_idx2 ON emp(job, ename) COMPRESS 1;
```

索引は、job 列値の繰返し項目を圧縮します。

NOLOGGING の例　次の文は、高速パラレル・ロードによって作成された表（この場合、すべての行はソート済）に、パラレルで索引を作成します。（適切な並列度が選択されます）。

```
CREATE INDEX i_loc
   ON big_table (akey)
   NOSORT
   NOLOGGING
   PARALLEL;
```

クラスタ索引の例　次の文は、employee クラスタに対して索引を作成します。

```
CREATE INDEX ic_emp ON CLUSTER employee;
```

クラスタ・キーのすべての列に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。

NULL の例　次の文を指定するとします。

```
SELECT ename FROM emp WHERE comm IS NULL;
```

この問合せでは、ビットマップ索引でない限り、comm 列に作成された索引は使用されません。

ファンクション索引の例　次の文は、ename 列の大文字評価に基づく emp 表にファンクション索引を作成します。

```
CREATE INDEX emp_i ON emp (UPPER(ename));
```

フル・テーブル・スキャンの実行ではなく、索引を使用するように、ファンクション値を後続の問合せで NULL 以外にします。次に例を示します。

```
SELECT * FROM emp WHERE UPPER(ename) IS NOT NULL
   ORDER BY UPPER(ename);
```

前述の文では、Oracle は索引を使用しますが、WHERE を省略すると、フル・テーブル・スキャンを実行する場合もあります。

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 は emp_fi を使用します。

```
CREATE INDEX emp_fi ON emp(cola + colb);

SELECT * FROM emp WHERE colb + cola > 500;
```

Type Method のファンクション索引の例　次の文には、2 つの数値属性（長さおよび幅）を含むオブジェクト型が必要です。area() メソッドは、四角形の領域を計算します。

```
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 recttab 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 recttab OF rectangle;
CREATE INDEX area_idx ON recttab x (x.area());
```

この索引を使用して、効率的に次の形式の問合せを評価できます。

```
SELECT * FROM recttab x WHERE x.area() > 100;
```

統計情報の計算例　次の文は、非パーティション索引の emp_indx 統計情報を収集します。

```
CREATE INDEX emp_indx ON emp(empno) COMPUTE STATISTICS;
```

収集された統計のタイプは、作成する索引のタイプによって異なります。

参照：『Oracle8*i* 概要』を参照してください。

パーティション索引の例　次の文は、stock_xactions 表にグローバル同一キー索引 stock_ix を、2 つのパーティションに分割して作成します。各パーティションがアルファベットを半分ずつ受け持ちます。索引パーティション名はシステムが生成します。

```
CREATE INDEX stock_ix ON stock_xactions
  (stock_symbol, stock_series)
   GLOBAL PARTITION BY RANGE (stock_symbol)
     (PARTITION VALUES LESS THAN ('N') TABLESPACE ts3,
      PARTITION VALUES LESS THAN (MAXVALUE) TABLESPACE ts4);
```

ハッシュ・パーティション表の索引の例　次の文は、sales 表の item 列にローカル索引を作成します。STORE IN 句は、sales がハッシュ・パーティション化されていることを示す LOCAL の直後に記述します。Oracle は、tbs1 表領域および tbs2 表領域の間にハッシュ・パーティションを分散します。

```
CREATE INDEX sales_idx ON sales(item) LOCAL
   STORE IN (tbs1, tbs2);
```

コンポジット・パーティション表の索引の例　次の文は、コンポジット・パーティション化されている sales 表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。STORE IN 句では、索引サブパーティションに 1 つ以上のデフォルト表領域を指定します。ただし、別の TABLESPACE 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_1997 の 4 つのサブパーティションにオーバーライドされます。

```
CREATE INDEX sales_idx ON sales(sale_date, item)
   STORAGE (INITIAL 1M, MAXEXTENTS UNLIMITED)
   LOCAL
   STORE IN (tbs1, tbs2, tbs3, tbs4, tbs5)
   (PARTITION q1_1997, PARTITION q2_1997,
    PARTITION q3_1997
      (SUBPARTITION q3_1997_s1 TABLESPACE ts2,
       SUBPARTITION q3_1997_s2 TABLESPACE ts4,
       SUBPARTITION q3_1997_s3 TABLESPACE ts6,
       SUBPARTITION q3_1997_s4 TABLESPACE ts8),
    PARTITION q4_1997,
    PARTITION q1_1998);
```

ビットマップ索引の例　次の文は、4 つのパーティションを持つ表にビットマップ・パーティション索引を作成します。

```
CREATE BITMAP INDEX partno_ix
    ON lineitem(partno)
    TABLESPACE ts1
    LOCAL (PARTITION quarter1 TABLESPACE ts2,
           PARTITION quarter2 STORAGE (INITIAL 10K NEXT 2K),
           PARTITION quarter3 TABLESPACE ts2,
           PARTITION quarter4);
```

ネストした表の索引の例　次の文は、一意索引 uniq_proj_indx を記憶域表 nested_project_table に作成します。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 projs_managed 内に固有の行が確保されます。

```
CREATE TYPE proj_type AS OBJECT
   (proj_num NUMBER, proj_name VARCHAR2(20));
CREATE TYPE proj_table_type AS TABLE OF proj_type;
CREATE TABLE employee ( emp_num NUMBER, emp_name CHAR(31),
   projs_managed proj_table_type )
   NESTED TABLE projs_managed STORE AS nested_project_table;
CREATE UNIQUE INDEX uniq_proj_indx
   ON nested_project_table ( NESTED_TABLE_ID, proj_num);
```

CREATE INDEXTYPE

用途

CREATE INDEXTYPE 文は、（アプリケーション固有の）ドメイン・インデックス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**を作成する場合に使用します。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、『Oracle8*i* データ・カートリッジ開発者ガイド』および『Oracle8*i* 概要』を参照してください。

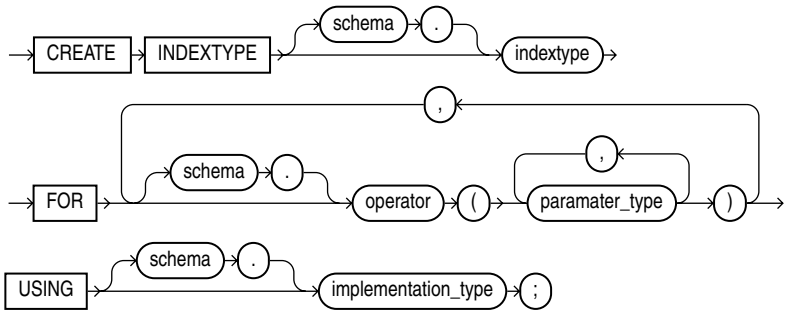
前提条件

自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、その演算子または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指定します。

> **参照：** 9-112 ページの「CREATE OPERATOR」を参照してください。

構文



キーワードとパラメータ

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

FOR を使用して、索引タイプにサポートされる演算子のリストを指定します。

schema 演算子が含まれているスキーマを指定します。*schema* を指定しない場合、その演算子が自スキーマにあるとみなされます。

operator 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。

この句に指定され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 演算子に対するパラメータ・タイプを指定します。

USING

USING 句によって、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 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、『Oracle8*i* 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

CREATE INDEXTYPE の例　次の文は、TextIndexType 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 contains 演算子および索引インタフェースを実装する TextIndexMethods タイプを指定します。

```
CREATE INDEXTYPE TextIndexType
   FOR contains (VARCHAR2, VARCHAR2)
   USING TextIndexMethods;
```

CREATE JAVA

用途

CREATE JAVA は、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する場合に使用します。

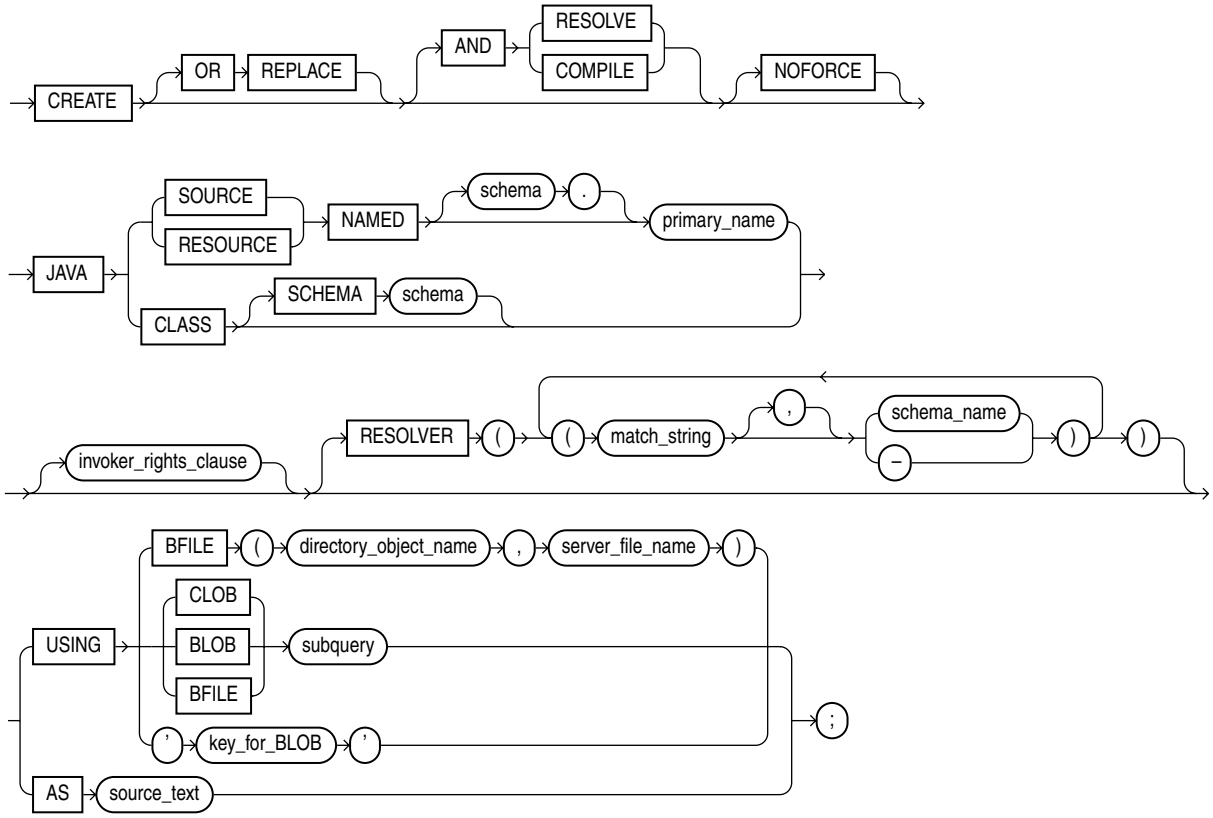
参照 :

- Java の概念については、『Oracle8*i* Java 開発者ガイド』を参照してください。
- Java ストアド・プロシージャについては、『Oracle8*i* Java ストアド・プロシージャ開発者ガイド』を参照してください。
- SQLJ については、『Oracle8*i* SQLJ 開発者ガイドおよびリファレンス』を参照してください。
- JDBC については、『Oracle8*i* 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。
- CORBA および EJB については、『Oracle8*i* CORBA 開発者ガイド』および『Oracle8*i* Enterprise JavaBeans 開発者ガイドおよびリファレンス』を参照してください。

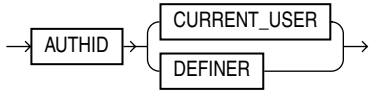
前提条件

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する場合は、CREATE ANY 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・オブジェクトを再作成する場合は、ALTER ANY PROCEDURE システム権限が必要です。

構文



invoker_rights_clause::=



キーワードとパラメータ

OR REPLACE

OR REPLACE を指定して、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します。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 を指定する場合、Oracle は、オブジェクトを再コンパイルまたは変換します。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： 詳細は、7-56 ページの「ALTER JAVA」を参照してください。

RESOLVE | COMPILE

RESOLVE および COMPILE は、同義キーワードです。この文が正常に実行されると作成される 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。

制限事項： Java リソースには、この句を指定できません。

NOFORCE

NOFORCE を指定すると、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に、この CREATE コマンドの結果をロールバックします。このオプションを省略した場合、正常に変換またはコンパイルできない場合でも処理は行われません（作成されたスキーマ・オブジェクトが残ります）。

JAVA SOURCE

JAVA SOURCE を指定すると、Java ソース・ファイルがロードされます。

JAVA CLASS

JAVA CLASS を指定すると、Java クラス・ファイルがロードされます。

JAVA RESOURCE

JAVA RESOURCE を指定すると、Java リソース・ファイルがロードされます。

NAMED

NAMED 句は、Java ソースまたは Java リソースに対して指定します。*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、この句にはスキーマ・オブジェクト名を指定し、Java リソースを保持します。

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 を省略した場合、自スキーマ内にオブジェクトが作成されます。

制限事項 :

- 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 *schema*

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。この句を省略した場合、自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 を使用して、クラスのメソッドが、権限を持つそのクラスを所有するユーザーのスキーマ内で実行されるか、または権限を持つ CURRENT_USER のスキーマ内で実行されるかを指定します。

また、この句は、問合せ、DML 操作、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

クラスのメソッドを CURRENT_USER 権限で実行する場合は、CURRENT_USER を指定します。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　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場合、およびクラスが存在するスキーマで外部名を変換する場合は、DEFINER を指定します。

参照：

- 『Oracle8*i* Java ストアド・プロシージャ開発者ガイド』を参照してください。
- CURRENT_USER の判断方法については、『Oracle8*i* 概要』および『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

RESOLVER

RESOLVER 句によって、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します。

- *match_string* は、完全な Java 名、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 ... 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

USING 句によって、Java クラスまたはリソースに対する文字（CLOB または BFILE）またはバイナリ（BLOB または BFILE）データの順序を指定します。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　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問合せを指定します。列の値は文字の順序を構成します

注意 : USING 句は、暗黙的にキーワード SELECT を使用します。そのため、副問合せではこのキーワードを指定しないでください。

key_for_BLOB　*key_for_BLOB* 句は、次の暗黙的な問合せを実行します。

```
SELECT LOB FROM CREATE$JAVA$LOB$TABLE
   WHERE NAME = 'key_for_BLOB';
```

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、表の列に 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_text*

Java または SQLJ ソースの文字列を指定します。

例

Java クラスの例　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 (bfile_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・ディレクトリに指定されるディレクトリ・オブジェクト bfile_dir が、すでに存在していることを前提としています。この例では、クラス名は Java クラス・スキーマ・オブジェクトの名前になります。

Java ソースの例　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 "Hello" AS
   public class Hello {
      public static String hello() {
         return "Hello World";   } };
```

Java リソースの例　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 (bfile_dir, 'textBundle.dat');
```

CREATE LIBRARY

用途

CREATE LIBRARY 文は、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する場合に使用します。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、第 3 世代言語（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。

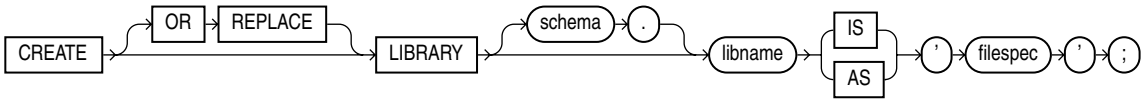
> **参照：** ファンクションおよびプロシージャの詳細は、9-42 ページの「CREATE FUNCTION」および『Oracle8*i* PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。

前提条件

自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



filespec: 11-27 ページの「filespec」を参照してください。

キーワードとパラメータ

OR REPLACE

既存のライブラリを再作成する場合は、OR REPLACE を指定します。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

call_spec でファンクションまたはプロシージャを宣言する場合、ライブラリを表すために作成する名前を指定します。

'*filespec*'

引用符で囲まれた文字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。

'*filespec*' は、CREATE LIBRARY 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。

例

CREATE LIBRARY の例　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';
```

CREATE MATERIALIZED VIEW

用途

CREATE MATERIALIZED VIEW 文は、**マテリアライズド・ビュー**を作成する場合に使用します。**スナップショット**と**マテリアライズド・ビュー**は、Oracle マニュアルでは同義語です。このマニュアルでは、「マテリアライズド・ビュー」を使用しています。どちらの用語も、1 つ以上の表の問合せ結果を含むデータベース・オブジェクトという意味です。

問合せの表を、**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。

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、拡張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** および**副問合せ**のマテリアライズド・ビューです。

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、問合せのリライトで使用できます。問合せのリライトとは、マスター表に関して記述されたユーザー要求を、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。データ・ウェアハウス環境では、すべてのマスター表はローカルである必要があります。

参照 :

- レプリケーションをサポートするマテリアライズド・ビューの詳細は、『Oracle8*i* レプリケーション・ガイド』を参照してください。
- データ・ウェアハウスをサポートするマテリアライズド・ビューの詳細は、『Oracle8*i* データ・ウェアハウス』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を直接付与する必要があります。

自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 または CREATE SNAPSHOT システム権限、および CREATE TABLE または CREATE ANY TABLE システム権限を付与する必要があります。
- 各表に対する SELECT オブジェクト権限または SELECT ANY TABLE システム権限を使用して、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。

- 各表に対する SELECT オブジェクト権限または SELECT ANY TABLE システム権限を使用して、CREATE ANY MATERIALIZED VIEW システム権限または CREATE ANY SNAPSHOT システム権限、および所有しないマテリアライズド・ビューのすべてのマスター表にアクセスする権限が必要です。
- 所有者またはマテリアライズド・ビューには、CREATE TABLE システム権限が必要です。また、この所有者は、各表に対する SELECT オブジェクト権限または SELECT ANY TABLE システム権限を使用して、スキーマ所有者が所有していないマテリアライズド・ビューのすべてのマスター表にアクセスする権限、およびこれらのマスター表に定義されたマテリアライズド・ビュー・ログにアクセスする権限が必要です。

前述の権限の他にも、問合せ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。

- マスター表の所有者には、QUERY REWRITE システム権限が必要です。
- マスター表の所有者でない場合は、GLOBAL QUERY REWRITE システム権限が必要です。
- スキーマ所有者がマスター表を所有していない場合、そのスキーマ所有者には GLOBAL QUERY REWRITE 権限が必要です。

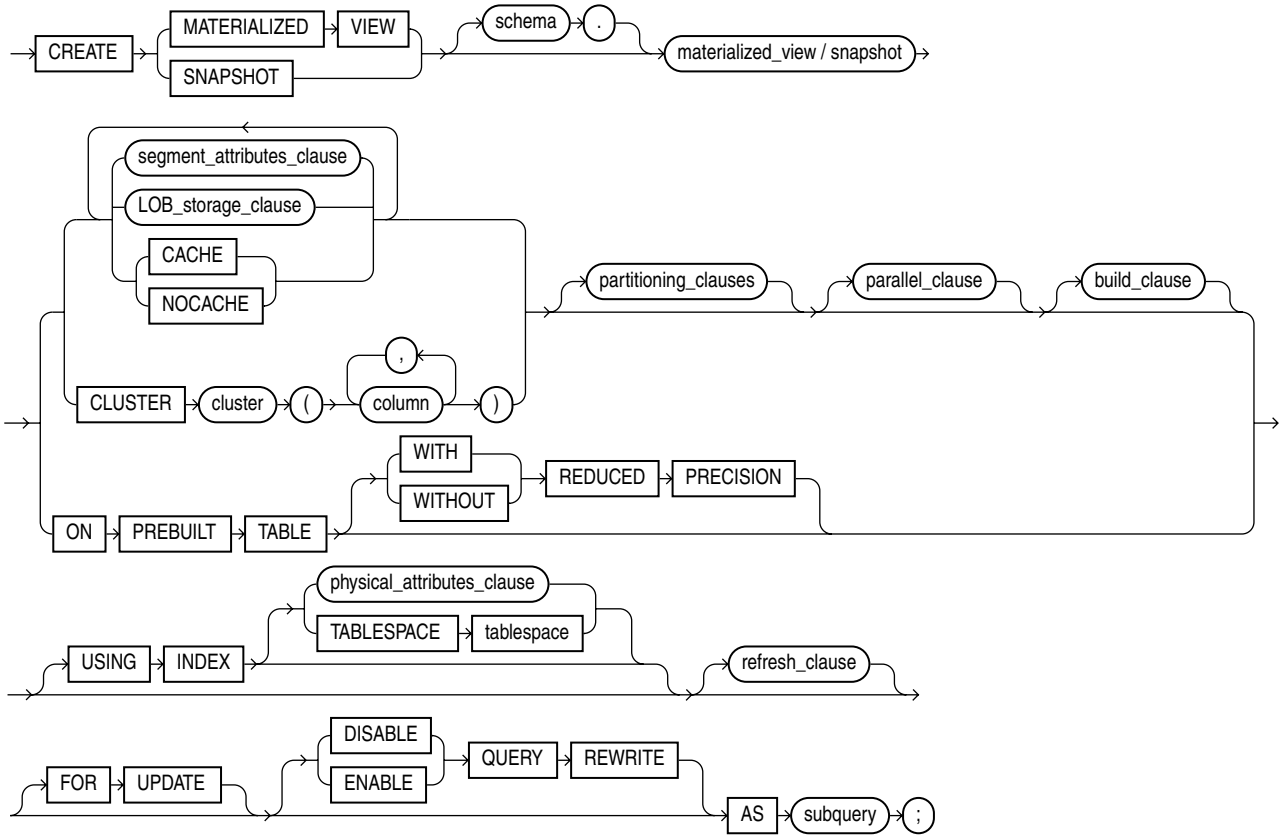
マテリアライズド・ビューを含むスキーマのユーザーは、マテリアライズド・ビューのマスター表および索引を格納するターゲット表領域に十分な割当て制限を持つか、または UNLIMITED TABLESPACE システム権限を持つ必要があります。

マテリアライズド・ビューを作成する場合、Oracle はマテリアライズド・ビューのスキーマ内すべてに、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します。ユーザーは、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。

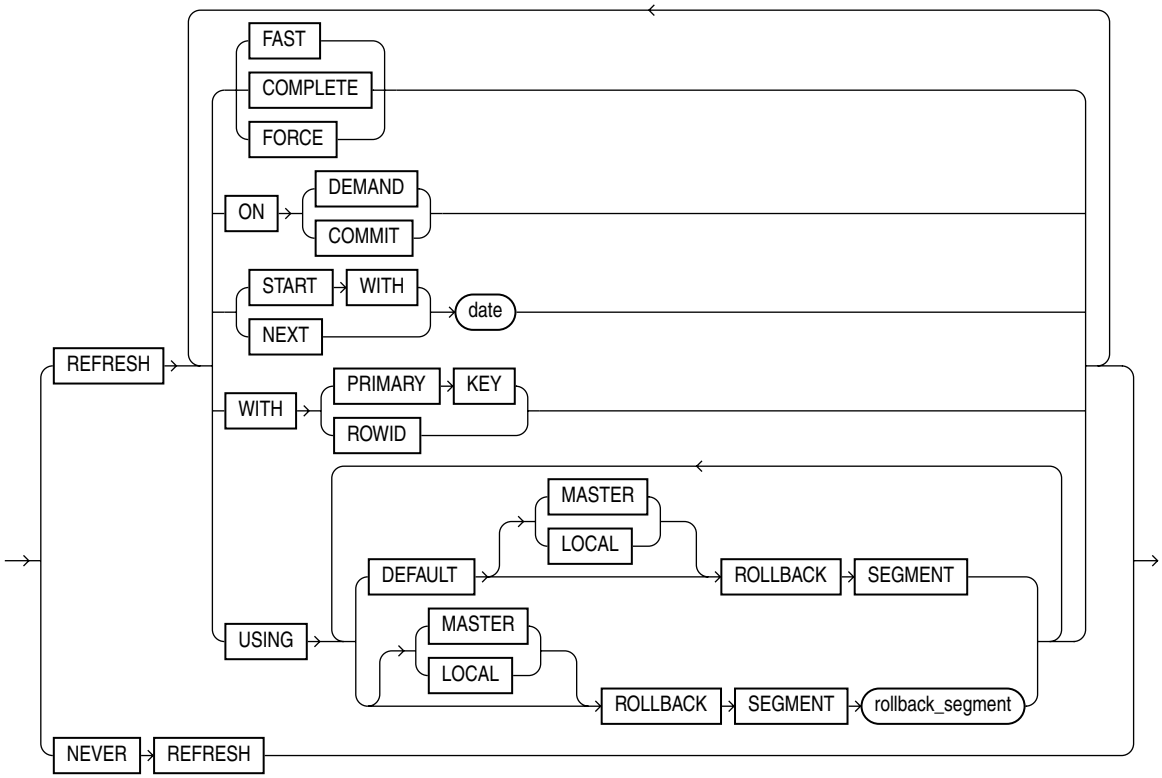
参照：

- これらの権限については、10-7 ページの「CREATE TABLE」、10-105 ページの「CREATE VIEW」および 9-51 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、『Oracle8*i* レプリケーション・ガイド』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、『Oracle8*i* データ・ウェアハウス』を参照してください。

構文

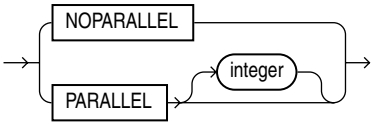


refresh_clause::=

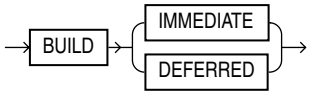


segment_attributes_clause: 10-7 ページの「CREATE TABLE」を参照してください。

parallel_clause::=



build_clause::=



subquery: 11-88 ページの「SELECT および副問合せ」を参照してください。

LOB_storage_clause: 10-7 ページの「CREATE TABLE」を参照してください。

partitioning_clauses: 10-7 ページの「CREATE TABLE」を参照してください。

キーワードとパラメータ

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* を指定しないと、自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 は、マテリアライズド・ビューを格納するための表、ビューおよび索引の名前を生成する際、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ます。

segment_attributes_clause

segment_attributes_clause は、PCTFREE、PCTUSED、INITRANS および MAXTRANS パラメータの値（INITRANS および MAXTRANS パラメータの場合は、USING INDEX 句でいつ使用するか）およびマテリアライズド・ビューの記憶特性を設定し、表領域を割り当て、ロギングが発生するかどうかを指定する場合に使用します。

参照：

- PCTFREE、PCTUSED、INITRANS および MAXTRANS、TABLESPACE および LOGGING | NOLOGGING パラメータについては、10-7 ページの「CREATE TABLE」を参照してください。
- 記憶特性については、11-129 ページの「storage_clause」を参照してください。

TABLESPACE

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・ビューを定義しているスキーマの所有者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。

LOB_storage_clause

LOB_storage_clause によって、LOB 記憶特性を指定できます。

> **参照：** この句のパラメータの指定の詳細は、10-7 ページの「CREATE TABLE」を参照してください。

LOGGING | NOLOGGING

LOGGING または NOLOGGING を指定して、マテリアライズド・ビューのロギング特性を設定します。

> **参照：** ロギング特性については、10-7 ページの「CREATE TABLE」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、フル・テーブル・スキャンの実行時にこの表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。

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 の指定については、10-7 ページの「CREATE TABLE」を参照してください。

CLUSTER

CLUSTER 句を使用して、指定したクラスタの一部としてマテリアライズド・ビューを作成します。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 句を指定した *physical_attributes_clause* または TABLESPACE を使用しないでください。

partitioning_clauses

partitioning_clauses によって、マテリアライズド・ビューが、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： 10-7 ページの「CREATE TABLE」を参照してください。

parallel_clause

parallel_clause によって、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。

注意： *parallel_clause* 構文は、以前のリリースの構文に代わるものです。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	シリアル実行を行う場合は、NOPARALLEL を指定します。これはデフォルト値です。
PARALLEL	すべてのパーティション化インスタンスで使用可能な CPU の数に、PARALLEL_THREADS_PER_CPU 初期化パラメータの値を掛けた並列度を選択させる場合は、PARALLEL を指定します。
PARALLEL *integer*	パラレル操作で使用されるパラレル・スレッド数である**並列度**を指定する場合は、*integer* 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： CREATE TABLE については、10-41 ページの「parallel_clause に関する注意事項」を参照してください。

build_clause

build_clause によって、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・ビューはすぐに移入されます。これはデフォルト値です。

DEFERRED　DEFERRED を指定すると、マテリアライズド・ビューは次の REFRESH 操作で移入されます。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。リフレッシュ前のマテリアライズド・ビューの値は UNUSABLE であるため、問合せのリライトには使用できません。

ON PREBUILT TABLE

ON PREBUILT TABLE 句によって、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。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。

制限事項：

- *subquery* の各列の別名は、*table_name* 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、非管理列（*subquery* で参照されない列）にデフォルト値も指定しない限り、その列に NOT NULL 制約は指定できません。

WITH REDUCED PRECISION	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の精度の低下を認める場合は、WITH REDUCED PRECISION を指定します。
WITHOUT REDUCED PRECISION	表またはマテリアライズド・ビュー列の精度を副問合せによって戻された精度と一致させるかまたは操作を中断するには、WITHOUT REDUCED PRECISION を指定します。これはデフォルト値です。

USING INDEX

USING INDEX 句によって、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、MAXTRANS 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。

制限事項 : この句では、PCTUSED または PCTFREE パラメータを指定できません。

refresh_clause

refresh_clause を使用して、Oracle がマテリアライズド・ビューを自動的にリフレッシュするデフォルトの方法、モードおよび時間を指定します。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は、現在マスター表にあるデータを実際に反映していることを確認するために、マテリアライズド・ビューのデータを更新します。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時間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、『Oracle8*i* レプリケーション・ガイド』および『Oracle8*i* データ・ウェアハウス』を参照してください。

FAST

FAST によって、増分リフレッシュ方法を指定します。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・ロードの INSERT 操作の場合）に格納されます。

基礎となるマスター表にマテリアライズド・ビューを作成していなくても、マテリアライズド集計ビューを作成できます。ただし、他の型のマテリアライズド・ビューを作成する場合、マテリアライズド・ビュー・ログがすでに存在していない限り、CREATE 文は失敗します。（ダイレクト・ロードの INSERT が行われると、Oracle は、ダイレクト・ローダー・ログを自動的に作成します。ユーザーによる手動操作は必要ありません。）

作成後、適切なマテリアライズド・ビュー・ログが存在する場合のみ、Oracle は、従来型 DML に対して高速リフレッシュを行います。

従来型 DML の変更の場合も、ダイレクト・パス・ロードの場合も、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関数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、『Oracle8*i* レプリケーション・ガイド』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、『Oracle8*i* データ・ウェアハウス』を参照してください。
- 4-8 ページの「分析関数」を参照してください。

`COMPLETE`

`COMPLETE` によって、完全リフレッシュ方法を指定します。これは、マテリアライズド・ビューの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`

`FORCE` によって、リフレッシュ時、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、そうでない場合には完全リフレッシュを実行することを指定します。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`

`ON COMMIT` によって、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は、必ず高速リフレッシュが実行されるように指定します。

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 : 『Oracle8*i* レプリケーション・ガイド』および『Oracle8*i* データ・ウェアハウス』を参照してください。

ON DEMAND　ON DEMAND によって、マテリアライズド・ビューが、3 つの DBMS_MVIEW 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ることを指定します。ON COMMIT および ON DEMAND の両方を指定しなかった場合、ON DEMAND がデフォルト値になります。

参照：

- これらのプロシージャについては、『Oracle8*i* PL/SQL パッケージ・プロシージャ 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、『Oracle8*i* データ・ウェアハウス』を参照してください。

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。

START WITH　最初の自動リフレッシュ時間を表す日付式を指定します。

NEXT　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します。

START WITH 値および NEXT 値は、将来の時間に評価される値です。START WITH 値を省略した場合、Oracle はマテリアライズド・ビューの作成時に 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間を判断します。START WITH 値を指定し、NEXT 値を指定しない場合、Oracle 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 値を両方とも指定しない場合、または *refresh_clause* 自体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　WITH PRIMARY KEY によって、作成される主キー・マテリアライズド・ビューを指定します。これはデフォルトであり、WITH ROWID に記述される値を除き、すべての場合に使用される必要があります。主キー・マテリアライズド・ビューを使用すると、高速リフレッシュを継続できるマテリアライズド・ビューの機能に影響を及ぼさずに、マテリアライズド・ビュー・マスター表を再編成できます。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ます。

参照：主キー・マテリアライズド・ビューの詳細は、『Oracle8*i* レプリケーション・ガイド』を参照してください。

WITH ROWID　WITH ROWID によって、作成される ROWID マテリアライズド・ビューを指定します。ROWID マテリアライズド・ビューは、リリース 8.0 以前のマスター表と互換性があります。

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、ROWID マテリアライズド・ビューを使用することができます。ROWID マテリアライズド・ビューは、単一リモート表をもと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。

- 固有または集計関数
- GROUP BY 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・ビューを高速リフレッシュすることはできません。

USING ROLLBACK SEGMENT *rollback_segment*

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。使用するロールバック・セグメント名を *rollback_segment* に指定します。

- DEFAULT は、使用するロールバック・セグメントが自動的に選択されることを指定します。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・ビューを変更する場合に有効です。

  参照 : 7-59 ページの「ALTER MATERIALIZED VIEW」を参照してください。

- MASTER は、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・セグメントを指定します。
- LOCAL は、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・セグメントを指定します。

  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・ロールバック・セグメントの指定については、『Oracle8*i* レプリケーション・ガイド』を参照してください。

MASTER または LOCAL を指定しない場合、デフォルトで LOCAL が使用されます。*rollback_segment* を指定しない場合、Oracle では、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。

マスター・ロールバック・セグメントは、マテリアライズド・ビューごとに格納され、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スキャンされます。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。

NEVER REFRESH　NEVER REFRESH によって、Oracle のリフレッシュ・メカニズムまたはプロシージャによるマテリアライズド・ビューのリフレッシュを行わないようにします。マテリアライズド・ビューで REFRESH 文を発行すると、エラーが戻ります。

FOR UPDATE

FOR UPDATE によって、副問合せ、主キーまたは 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。

参照 : 『Oracle8*i* レプリケーション・ガイド』を参照してください。

QUERY REWRITE

QUERY REWRITE 句によって、マテリアライズド・ビューが問合せのリライトで使用できるかどうかを指定します。

ENABLE　ENABLE を指定すると、問合せのリライトでマテリアライズド・ビューが使用可能になります。

参照 : 問合せのリライトの詳細は、『Oracle8*i* データ・ウェアハウス』を参照してください。

注意 :

- 問合せの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、問合せのリライトで使用できるマテリアライズド・ビューを作成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、必ず分析してください。問合せのリライトを最適化するために、ANALYZE 操作で生成した統計が必要です。

制限事項 :

- 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、問合せの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、問合せの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。

  参照 : 9-42 ページの「CREATE FUNCTION」および『Oracle8*i* データ・ウェアハウス』を参照してください。

DISABLE　DISABLE を指定すると、マテリアライズド・ビューが問合せのリライトで使用禁止になります。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。

AS *subquery*

マテリアライズド・ビュー定義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。この問合せは、有効な SQL 問合せです。ただし、すべての問合せが高速リフレッシュされるわけではなく、問合せの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・ビュー副問合せに関する注意事項

- BUILD DEFERRED を指定する場合、すぐには問合せは実行されません。
- マテリアライズド・ビュー問合せの FROM 句に指定する各表およびビューを、それを定義しているスキーマで修飾することをお薦めします。

  参照 : その他の通告については、10-7 ページの「CREATE TABLE」の AS 副問合せ句を参照してください。

マテリアライズド・ビュー副問合せに関する制限事項

- ユーザー SYS が所有する表またはビューから、マテリアライズド・ビュー問合せを選択できますが、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・ビュー問合せではユーザー定義型を参照できません。

- 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。
- 副問合せが一時表を参照する場合、このマテリアライズド・ビューに対するログは作成できません。また、CREATE MATERIALIZED VIEW または ALTER MATERIALIZED VIEW の QUERY REWRITE 句も指定できません。
- マテリアライズド・ビューの FROM リストが別のマテリアライズド・ビューを参照する場合、手動でマテリアライズド・ビューのリフレッシュ順序を制御する必要があります。整合性をメンテナンスするには、まず、他のマテリアライズド・ビューを参照するマテリアライズド・ビューをリフレッシュし、次に、参照されるマテリアライズド・ビューをリフレッシュする必要があります。

問合せの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。

- 副問合せには、ROWNUM、USER、SYSDATE、リモート表、順序、あるいはデータベースまたはパッケージ状態の書込み、読取りを行う PL/SQL ファンクションへの（直接的またはビューを介しての）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、ローカルである必要があります。

マテリアライズド・ビューの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・ウェアハウスに関する制限事項は、『Oracle8*i* データ・ウェアハウス』を参照してください。
- レプリケーションに関する制限事項は、『Oracle8*i* レプリケーション・ガイド』を参照してください。

例

マテリアライズド集計ビューの例　次の文は、マテリアライズド・ビューを作成して移入し、リフレッシュ方法、モードおよび時間を指定します。

```
CREATE MATERIALIZED VIEW mv1 REFRESH FAST ON COMMIT
   BUILD IMMEDIATE
   AS SELECT t.month, p.prod_name, SUM(f.sales) AS sum_sales
      FROM time t, product p, fact f
      WHERE f.curDate = t.curDate AND f.item = p.item
      GROUP BY t.month, p.prod_name;
```

次の文は、sales_by_month_by_state マテリアライズド・ビューを作成し、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・ビューの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 my_ts PARALLEL (10)
   ENABLE QUERY REWRITE
   BUILD IMMEDIATE
   REFRESH COMPLETE
   AS SELECT t.month, g.state, SUM(f.sales) AS sum_sales
      FROM fact f, time t, geog g
      WHERE f.cur_date = t.cur_date AND f.city_id = g.city_id
      GROUP BY month, state;
```

事前作成したマテリアライズド・ビューの例　次の文は、既存の集計表 sales_sum_table に対して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 DATE, state VARCHAR2(25), sales NUMBER);

CREATE MATERIALIZED VIEW sales_sum_table
   ON PREBUILT TABLE
   ENABLE QUERY REWRITE
   AS SELECT t.month, g.state, SUM(f.sales) AS sum_sales
      FROM fact f, time t, geog g
      WHERE f.cur_date = t.cur_date AND f.city_id = g.city_id
      GROUP BY month, state;
```

この例では、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。

マテリアライズド結合ビューの例　次の文は、マテリアライズド結合ビュー mjv を作成します。

```
CREATE MATERIALIZED VIEW mjv
   REFRESH FAST
      AS SELECT l.rowid as l_rid, l.pk, l.ofk, l.c1, l.c2,
         o.rowid as o_rid, o.pk, o.cfk, o.c1, o.c2,
         c.rowid as c_rid, c.pd, c.c1, c.c2
      FROM l, o, c
      WHERE l.ofk = o.pk(+) AND o.ofk = c.pk(+);
```

副問合せマテリアライズド・ビューの例　次の文は、リモート・データベースで sales スキーマの orders 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 sales.orders FOR UPDATE
   AS SELECT * FROM sales.orders@dbs1.acme.com o
   WHERE EXISTS
     (SELECT * FROM sales.customers@dbs1.acme.com c
      WHERE o.c_id = c.c_id);
```

主キーの例　次の文は、主キー・マテリアライズド・ビュー human_genome を作成します。

```
CREATE MATERIALIZED VIEW human_genome
   REFRESH FAST START WITH SYSDATE NEXT  SYSDATE + 1/4096
   WITH PRIMARY KEY
   AS SELECT * FROM genome_catalog;
```

ROWID の例　次の文は、ROWID マテリアライズド・ビューを作成します。

```
CREATE MATERIALIZED VIEW emp_data REFRESH WITH ROWID
    AS SELECT * FROM emp_table73;
```

定期的リフレッシュの例　次の文は、ニューヨークの scott が所有する従業員表のデータを含むマテリアライズド・ビュー emp_sf を作成します。

```
CREATE MATERIALIZED VIEW emp_sf
   PCTFREE 5 PCTUSED 60
   TABLESPACE users
   STORAGE (INITIAL 50K NEXT 50K)
   REFRESH FAST NEXT sysdate + 7
   AS SELECT * FROM scott.emp@ny;
```

この文では、START WITH パラメータが指定されていないため、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間が判断されます。現在、ニューヨークの従業員表にマテリアライズド・ビューのログが存在する場合、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、Oracle がマテリアライズド・ビューをメンテナンスするために使用する表の記憶特性も設定しています。

自動リフレッシュ時間の例　次の文は、ダラスおよびボルティモア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_emps を作成します。

```
CREATE MATERIALIZED VIEW all_emps
   PCTFREE 5 PCTUSED 60
   TABLESPACE users
   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 fran.emp@dallas
         UNION
      SELECT * FROM marco.emp@balt;
```

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。all_emps には UNION が含まれますが、UNION は高速リフレッシュをサポートしていないため、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_clause* 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・サイズをそれぞれ 50KB とします。
- 2 番目の *storage_clause*（USING INDEX 句付きで指定）では、索引の 1 番目と 2 番目のエクステント・サイズをそれぞれ 25KB とします。

ロールバック・セグメントの例　次の文では、リモート・マスターにあるロールバック・セグメント master_seg、およびマテリアライズド・ビューを含むローカル・リフレッシュ・グループに対するロールバック・セグメント snap_seg を使用して、マテリアライズド・ビュー sales_emp を作成します。

```
CREATE MATERIALIZED VIEW sales_emp
   REFRESH FAST START WITH SYSDATE NEXT SYSDATE + 7
   USING MASTER ROLLBACK SEGMENT master_seg
   LOCAL ROLLBACK SEGMENT snap_seg
   AS SELECT * FROM bar;
```

次の文には誤りがあり、DEFAULT ロールバック・セグメントを使用してセグメント名を指定しているため、エラーが生成されます。

```
CREATE MATERIALIZED VIEW bogus
   REFRESH FAST START WITH SYSDATE NEXT SYSDATE + 7
   USING DEFAULT ROLLBACK SEGMENT snap_seg
   AS SELECT * FROM faux;
```

CREATE MATERIALIZED VIEW LOG

用途

CREATE MATERIALIZED VIEW LOG 文は、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**を作成する場合に使用します。**スナップショット**と**マテリアライズド・ビュー**は同義語です。どちらの用語も、1 つ以上の表の問合せ結果を含む表という意味です。それぞれ同じデータベースまたはリモート・データベースに存在することがあります。

マスター表のデータで DML が変更された場合、Oracle 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。このプロセスが高速リフレッシュです。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。マスター表には、単一マテリアライズド・ビュー・ログのみ必要です。Oracle 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（結合を含むマテリアライズド・ビュー）を高速リフレッシュする場合、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、9-86 ページの「CREATE MATERIALIZED VIEW」、『Oracle8*i* 概要』、『Oracle8*i* データ・ウェアハウス』および『Oracle8*i* レプリケーション・ガイド』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、7-73 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、10-142 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・ログについては、『Oracle8*i* 概要』を参照してください。

前提条件

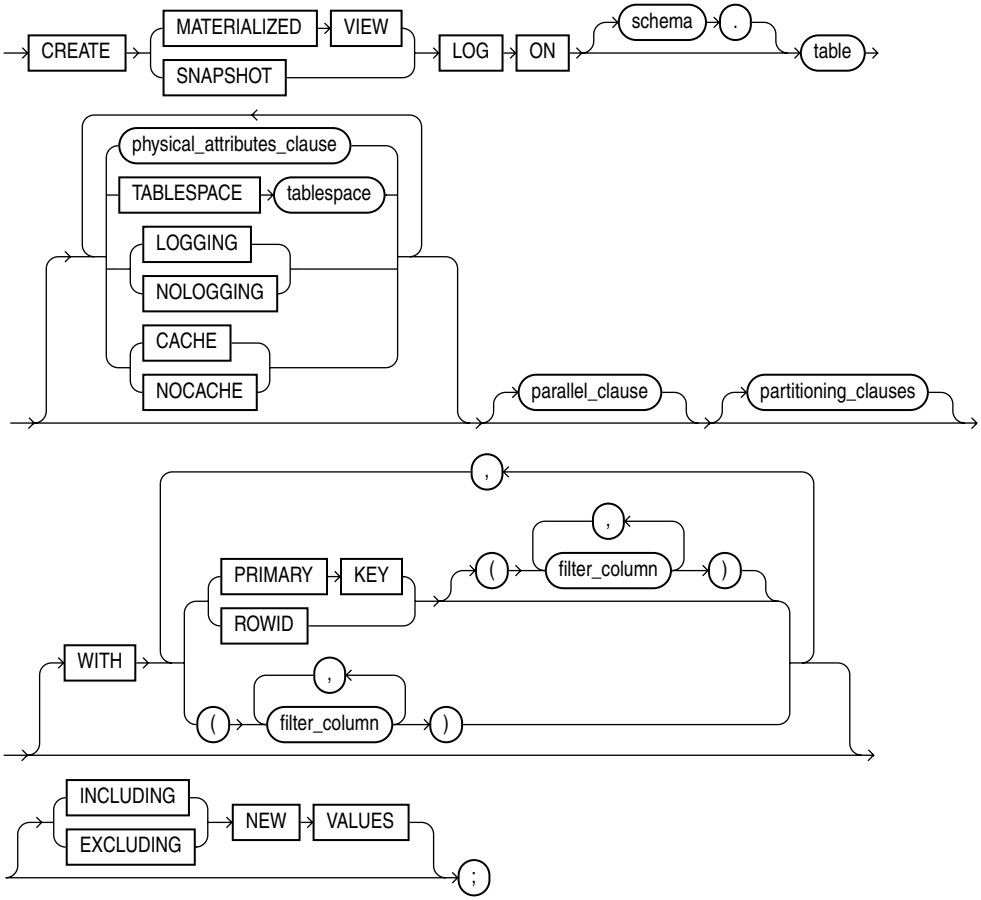
マテリアライズド・ビュー・ログを作成するために必要な権限は、マテリアライズド・ビュー・ログに関連付けられた基礎となるオブジェクトの作成に必要な権限と直接関連しています。

- 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 権限、COMMENT ANY TABLE 権限、およびマスター表に対する SELECT 権限または SELECT ANY TABLE 権限が必要です。

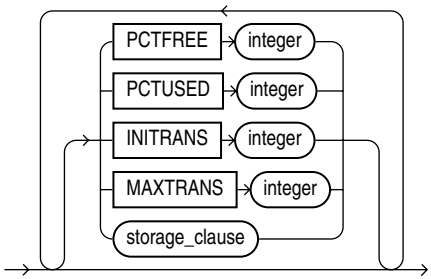
どちらの場合も、マテリアライズド・ビュー・ログの所有者には、マテリアライズド・ビュー・ログを格納するための表領域に十分な割当て制限または UNLIMITED TABLESPACE システム権限が必要です。

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、『Oracle8*i* データ・ウェアハウス』を参照してください。

構文

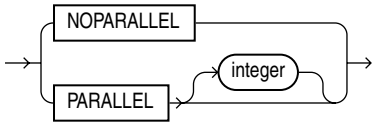

CREATE
MATERIALIZED
VIEW
SNAPSHOT
LOG
ON
schema
.
table
physical_attributes_clause
TABLESPACE
tablespace
LOGGING
NOLOGGING
CACHE
NOCACHE
parallel_clause
partitioning_clauses
WITH
,
PRIMARY
KEY
ROWID
(
filter_column
)
INCLUDING
EXCLUDING
NEW
VALUES
;

physical_attributes_clause::=



storage_clause: 11-129 ページの「storage_clause」を参照してください。

parallel_clause::=



partitioning_clauses: 10-34 ページの「CREATE TABLE」の「table_properties」を参照してください。

キーワードとパラメータ

schema

マテリアライズド・ビュー・ログのマスター表が定義されているスキーマを指定します。*schema* を省略した場合、マスター表は自スキーマ内に定義されているものとみなされます。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。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。

physical_attributes_clause

physical_attributes_clause を使用して、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の値の設定をします。

参照： 10-7 ページの「CREATE TABLE」および 11-129 ページの「storage_clause」を参照してください。

TABLESPACE

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・ログを定義しているスキーマの所有者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 | NOLOGGING

LOGGING または NOLOGGING を指定して、マテリアライズド・ビュー・ログのロギング特性を設定します。

参照： ロギング特性については、10-7 ページの「CREATE TABLE」を参照してください。

CACHE | NOCACHE

アクセス頻度が高いデータについて、CACHE は、フル・テーブル・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。

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 の指定については、8-2 ページの「ALTER TABLE」を参照してください。

parallel_clause

parallel_clause によって、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。

注意： *parallel_clause* 構文は、以前のリリースの構文に代わるものです。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　シリアル実行を行う場合は、NOPARALLEL を指定します。これはデフォルト値です。

PARALLEL　すべてのパーティション化インスタンスで使用可能な CPU の数に、PARALLEL_THREADS_PER_CPU 初期化パラメータの値を掛けた並列度を選択させる場合は、PARALLEL を指定します。

PARALLEL *integer*　パラレル操作で使用されるパラレル・スレッド数である**並列度**を指定する場合は、*integer* 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： CREATE TABLE については、10-41 ページの「parallel_clause に関する注意事項」を参照してください。

partitioning_clauses

partitioning_clauses を使用して、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。詳細は、10-7 ページの「CREATE TABLE」を参照してください。

WITH

WITH 句を使用して、マスター内の行の更新時に、マテリアライズド・ビュー・ログに主キーまたは ROWID のいずれか、あるいは主キーおよび ROWID の両方を記録するかどうかを指定します。

この句では、マテリアライズド・ビュー・ログに、副問合せのマテリアライズド・ビューによって参照される非主キー列であるフィルタ列を記録するかどうかを指定します。

この句を指定しなかった場合、主キーの値がデフォルトで格納されます。フィルタ列リストを指定した場合は、主キーの値は、暗黙的にフィルタ列リストに格納されます。ただし、作成時に ROWID または ROWID（filter_column）のみを指定した場合は、主キーの値が暗黙的に格納されることはありません。

PRIMARY KEY　PRIMARY KEY によって、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します。マスター表で更新された行の主キーは、マテリアライズド・ビュー・ログに記録される必要があります。

ROWID　ROWID によって、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します。ROWID は、マテリアライズド・ビュー・ログに記録される必要があります。

filter_column　マテリアライズド・ビュー・ログに記録されるフィルタ列を、カンマで区切られたリストで指定します。副問合せで定義された高速リフレッシュ可能な主キーのマテリアライズド・ビューの場合、定義している副問合せによって参照されるすべてのフィルタ列が、マテリアライズド・ビュー・ログに記録される必要があります。

制限事項：

- PRIMARY KEY、ROWID およびフィルター列リストの指定は、各マテリアライズド・ビュー・ログで、それぞれ 1 回のみ可能です。

  これは、PRIMARY KEY が暗黙的に *filter_column* に含まれるためです。次の組合せは指定できません。

  ```
  ADD PRIMARY KEY,(filter_column)
  ADD (filter_column), PRIMARY KEY
  ```

NEW VALUES

NEW VALUES 句によって、マテリアライズド・ビュー・ログの古い値と新しい値の両方を保存するかどうかを指定できます。

INCLUDING　INCLUDING を指定すると、古い値と新しい値の両方が保存されます。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、ログに対する新しい値の記録が使用禁止にされます。これはデフォルト値です。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。ただし、この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

主キーの例　次の文は、主キー値のみを記録する従業員表のマテリアライズド・ビュー・ログが作成します。

```
CREATE MATERIALIZED VIEW LOG ON emp WITH PRIMARY KEY;
```

Oracle では、このマテリアライズド・ビューを使用して、emp 表に後で作成される単純主キー・マテリアライズド・ビューに対しても高速リフレッシュを実行できます。

次の文は、更新された行の主キーのみを記録するマテリアライズド・ビュー・ログも作成します。

```
CREATE MATERIALIZED VIEW LOG ON emp
   PCTFREE 5
   TABLESPACE users
   STORAGE (INITIAL 10K NEXT 10K);
```

ROWID の例　次の文は、更新された行の主キーおよび ROWID を記録するマテリアライズド・ビュー・ログを作成します。

```
CREATE MATERIALIZED VIEW LOG ON sales WITH ROWID, PRIMARY KEY;
```

フィルタ列の例　次の文は、主キーおよびフィルタ列 zip への更新内容を記録するマテリアライズド・ビュー・ログを作成します。

```
CREATE MATERIALIZED VIEW LOG ON address WITH (zip);
```

NEW VALUES の例　次の文は、マスター表を作成し、次に INCLUDING NEW VALUES を指定するマテリアライズド・ビュー・ログを作成します。

```
CREATE TABLE agg
   (u NUMBER, a NUMBER, b NUMBER, c NUMBER, d NUMBER);

CREATE MATERIALIZED VIEW LOG ON agg
   WITH ROWID (u,a,b,c,d)
   INCLUDING NEW VALUES;
```

次の文は、agg ログを使用するためのマテリアライズド集計ビューを作成します。

```
CREATE MATERIALIZED VIEW sn0
   REFRESH FAST ON COMMIT
   AS SELECT SUM(b+c), COUNT(*), a, d, COUNT(b+c)
         FROM agg
         GROUP BY a,d;
```

使用する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 文は、新しい演算子を作成し、バインディングを定義する場合に使用します。

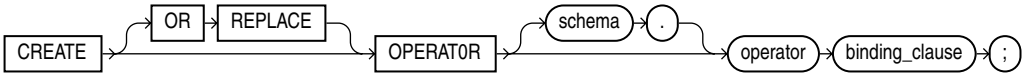
演算子は、索引タイプ、DML および問合せ SQL 文によって参照できます。演算子は、ファンクション、パッケージ、型および他のユーザー定義オブジェクトを順番に参照します。

> **参照：** これらの依存性および一般的な演算子については、『Oracle8*i* データ・カートリッジ開発者ガイド』および『Oracle8*i* 概要』を参照してください。

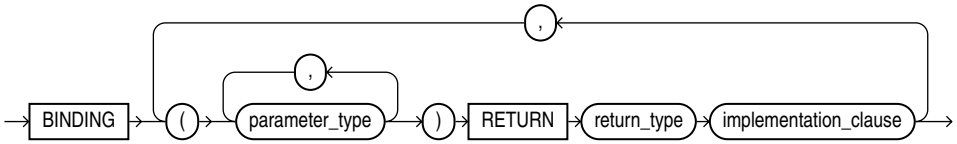
前提条件

自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 権限が必要です。

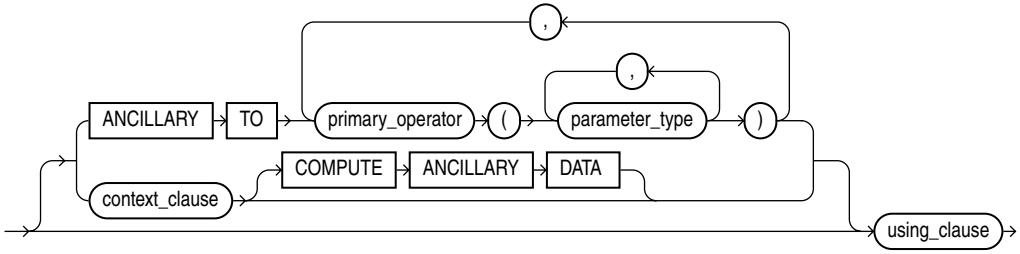
構文



binding_clause::=

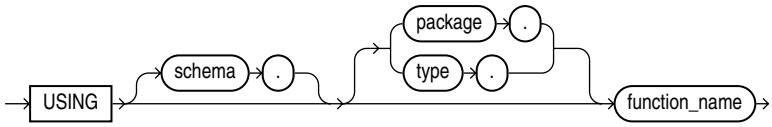


implementation_clause::=



context_clause::=

using_clause::=



キーワードとパラメータ

OR REPLACE

OR REPLACE を指定して、演算子スキーマ・オブジェクトの定義を置き換えます。

制限事項 : 演算子に依存オブジェクト（たとえば、演算子をサポートする索引タイプ）がない場合のみ、定義を再作成できます。

schema

演算子が含まれているスキーマを指定します。*schema* を指定しない場合、その演算子が自スキーマにあるとみなされます。

operator

作成する演算子の名前を指定します。

binding_clause

binding_clause を使用して、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*）を指定します。各バインドの署名（対応するファンクションの引数のデータ型順序）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。

制限事項：REF、LONG または LONG RAW の *parameter_type* は指定できません。

> **参照：** オーバーロードの詳細は、『Oracle8*i* PL/SQL 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。

RETURN *return_type*	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。 **制限事項：**REF、LONG または LONG RAW の *return_type* は指定できません。	
implementation_clause	ANCILLARY TO *primary_operator*	ANCILLARY TO 句によって、演算子バインディングが、指定した主演算子バインディング（*primary_operator*）を補助することを指定します。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	演算子の機能的な実装によって、スキャン・コンテキストとして使用される実装タイプ名を指定します。
	COMPUTE ANCILLARY DATA	COMPUTE ANCILLARY DATA によって、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します。
using_clause	*using_clause* によって、バインドを実装するファンクションを指定できます。	
	function_name	ファンクションの名前を指定します。ファンクションは、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、型メソッドまたはそのいずれかに対するシノニムになります。

例

CREATE OPERATOR の例　次の文は、2 つのバインドがある scott スキーマに MERGE という演算子を作成します。1 番目のバインドは、2 つの VARCHAR2 値をマージし、VARCHAR2 結果を戻します。2 番目のバインドは、2 つのジオメトリを単一のジオメトリにマージします。バインド用に対応するファンクション実装も指定されます。

```
CREATE OPERATOR scott.merge
BINDING (varchar2, varchar2) RETURN varchar2
         USING text.merge,
   (spatial.geo, spatial.geo) RETURN spatial.geo
         USING spatial.merge;
```

CREATE OUTLINE

用途

CREATE OUTLINE 文は、**ストアド・アウトライン**を作成する場合に使用します。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にもかかわらず、特定の SQL 文が発行された場合、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。これらの要因の変更が考慮されるように、アウトラインを変更することもできます。

個々のセッションまたはシステムに対して、ストアド・アウトラインを使用可能または使用禁止にします。

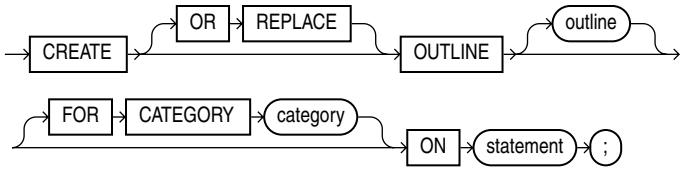
参照：

- 『Oracle8*i* パフォーマンスのための設計およびチューニング』を参照してください。
- アウトラインの変更については、7-79 ページの「ALTER OUTLINE」を参照してください。
- 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する場合は、7-101 ページの「ALTER SESSION」および 7-123 ページの「ALTER SYSTEM」を参照してください。

前提条件

アウトラインを作成する場合は、CREATE ANY OUTLINE システム権限が必要です。

構文



キーワードとパラメータ

OR REPLACE

OR REPLACE を指定して、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ます。

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、システムがアウトライン名を生成します。

FOR CATEGORY カテゴリ

グループ・ストアド・アウトラインに使用され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 *statement*

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。次のいずれかを指定できます。

- SELECT
- DELETE
- UPDATE
- INSERT ... SELECT
- CREATE TABLE ... AS SELECT

注意： 単一文に対して複数アウトラインを指定できますが、同じ文に対するアウトラインは別のカテゴリにある必要があります。

例

CREATE OUTLINE の例 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 ename, sal FROM emp;
```

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。

CREATE PACKAGE

用途

CREATE PACKAGE 文は、ストアド・パッケージを指定する場合に使用します。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。**仕様部**では、これらのオブジェクトを宣言します。

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、9-42 ページの「CREATE FUNCTION」および 9-127 ページの「CREATE PROCEDURE」を参照してください。
- パッケージの変更については、7-81 ページの「ALTER PACKAGE」を参照してください。
- パッケージの削除については、10-147 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、『Oracle8*i* アプリケーション開発者ガイド 基礎編』および『Oracle8*i* PL/SQL パッケージ・プロシージャ リファレンス』を参照してください。

前提条件

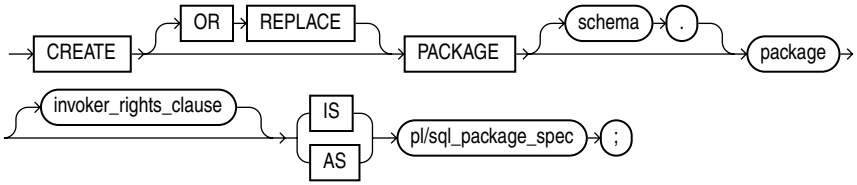
パッケージを作成する前に、ユーザー SYS は SQL スクリプト DBMSSTDX.SQL を実行する必要があります。このスクリプトの正確な名前および格納位置は、使用するオペレーティング・システムによって異なります。

自パッケージ内にパッケージを作成する場合は、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する場合は、CREATE ANY PROCEDURE システム権限が必要です。

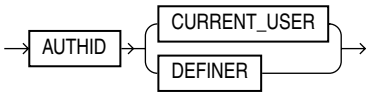
Oracle プリコンパイラ・プログラム内に CREATE PACKAGE 文を埋め込む場合、キーワード END-EXEC とその後に各言語の埋込み SQL 文の終了記号を記述して文を終了する必要があります。

参照：『Oracle8*i* PL/SQL 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。

構文



invoker_rights_clause::=



キーワードとパラメータ

OR REPLACE

既存のパッケージ仕様部を再作成する場合は、OR REPLACE を指定します。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、7-81 ページの「ALTER PACKAGE」を参照してください。

schema

パッケージを含むスキーマを指定します。*schema* を指定しない場合、自スキーマ内に表が作成されます。

package

作成するパッケージの名前を指定します。

パッケージの作成がコンパイル・エラーになった場合、Oracle はエラーを戻します。SHOW ERRORS コマンドを使用すると、関連するコンパイラ・エラー・メッセージを表示できます。

invoker_rights_clause

invoker_rights_clause によって、パッケージ内のファンクションおよびプロシージャが、権限を持つユーザーのスキーマ内で実行されるか、または、権限を持つ CURRENT_USER のスキーマ内で実行されるかを指定できます。この仕様部は対応するパッケージ本体にも適用されます。

この句は、問合せ、DML 操作およびパッケージにおける動的 SQL 文の外部名の変換方法も定義します。

AUTHID CURRENT_USER	パッケージを CURRENT_USER 権限で実行する場合は、CURRENT_USER を指定します。この句は、実行者権限パッケージを作成します。 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、パッケージを含むスキーマで変換します。
AUTHID DEFINER	パッケージを含むスキーマの所有者権限でパッケージを実行する場合、およびパッケージを含むスキーマ内で外部名を変換する場合は、DEFINER を指定します。これはデフォルト値です。

参照：

- 『Oracle8*i* PL/SQL 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。
- CURRENT_USER の判断方法については、『Oracle8*i* 概要』および『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

pl/sql_package_spec

型定義、カーソル宣言、変数宣言、定数宣言、例外宣言、PL/SQL サブプログラム仕様およびコール仕様（PL/SQL 内で記述された C または Java ルーチンの宣言）を含むことができるパッケージ仕様部を指定します。

参照：

- PL/SQL サブプログラムの詳細は、『Oracle8*i* PL/SQL 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。
- Oracle が提供するパッケージについては、『Oracle8*i* PL/SQL パッケージ・プロシージャ リファレンス』を参照してください。
- パッケージにおけるユーザー定義ファンクションの制限事項については、9-45 ページの「ユーザー定義ファンクションの制限事項」を参照してください。

例

CREATE PACKAGE の例　次の文は、emp_mgmt パッケージ仕様部を作成します。

```
CREATE PACKAGE emp_mgmt AS
   FUNCTION hire(ename VARCHAR2, job VARCHAR2, mgr NUMBER,
                 sal NUMBER, comm NUMBER, deptno NUMBER)
      RETURN NUMBER;
   FUNCTION create_dept(dname VARCHAR2, loc VARCHAR2)
      RETURN NUMBER;
   PROCEDURE remove_emp(empno NUMBER);
   PROCEDURE remove_dept(deptno NUMBER);
   PROCEDURE increase_sal(empno NUMBER, sal_incr NUMBER);
   PROCEDURE increase_comm(empno NUMBER, comm_incr NUMBER);
      no_comm EXCEPTION;
      no_sal EXCEPTION;
END emp_mgmt;
```

emp_mgmt パッケージ仕様部では、次のパブリック・プログラム・オブジェクトが宣言されます。

- hire および create_dept の各ファンクション
- remove_emp、remove_dept、increase_sal および increase_comm の各プロシージャ
- no_comm および no_sal の各例外

これらのすべてのオブジェクトは、このパッケージに対するアクセス権限があるユーザーが使用できます。パッケージの作成後は、パッケージのパブリック・プロシージャまたはファンクションをコールしたり、パッケージのパブリック例外を発生させるアプリケーションを開発できます。

このパッケージのプロシージャおよびファンクションをコールする前に、パッケージ本体でこれらのプロシージャおよびファンクションを定義しておく必要があります。emp_mgmt パッケージ本体を作成する CREATE PACKAGE BODY 文の例については、9-122 ページの「CREATE PACKAGE BODY」を参照してください。

CREATE PACKAGE BODY

用途

CREATE PACKAGE BODY 文は、ストアド・パッケージを作成する場合に使用します。ストアド・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。**本体**では、これらのオブジェクトを定義します。

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、9-42 ページの「CREATE FUNCTION」および 9-127 ページの「CREATE PROCEDURE」を参照してください。
- パッケージの作成方法などの説明は、9-118 ページの「CREATE PACKAGE」を参照してください。
- 使用例については、9-124 ページの「例」を参照してください。
- パッケージの変更については、7-81 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、10-147 ページの「DROP PACKAGE」を参照してください。

前提条件

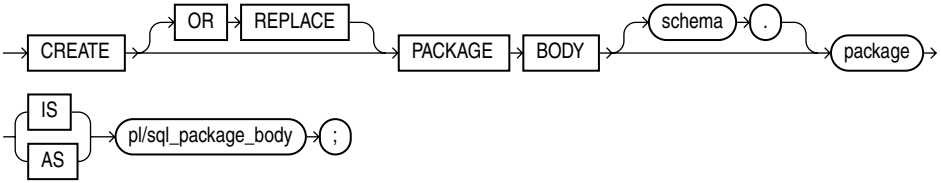
パッケージを作成する前に、ユーザー SYS は SQL スクリプト DBMSSTDX.SQL を実行する必要があります。このスクリプトの正確な名前および格納位置は、使用するオペレーティング・システムによって異なります。

自パッケージ内にパッケージを作成する場合は、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する場合は、CREATE ANY PROCEDURE システム権限が必要です。

Oracle 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 と、その後に各言語用の埋込み 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：『Oracle8*i* PL/SQL 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。

構文



キーワードとパラメータ

OR REPLACE

既存のパッケージ本体を再作成する場合は、OR REPLACE を指定します。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、7-81 ページの「ALTER PACKAGE」を参照してください。

schema

パッケージを含むスキーマを指定します。*schema* を指定しない場合、現行のスキーマ内にパッケージが作成されます。

package

作成するパッケージの名前を指定します。

pl/sql_package_body

PL/SQL サブプログラム本体またはコール仕様（PL/SQL に記述された C または Java ルーチンの宣言）を含むことができるパッケージ本体を指定します。

参照：

- PL/SQL または C パッケージ・プログラム・ユニットの書込みの詳細は、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。
- JAVA パッケージ・プログラム・ユニットについては、『Oracle8*i* Java ストアド・プロシージャ開発者ガイド』を参照してください。
- パッケージにおけるユーザー定義ファンクションの制限事項については、9-45 ページの「ユーザー定義ファンクションの制限事項」を参照してください。

例

CREATE PACKAGE BODY の例　次の文は、emp_mgmt パッケージ本体を作成します。

```
CREATE PACKAGE BODY emp_mgmt AS
   tot_emps  NUMBER;
   tot_depts NUMBER;

FUNCTION hire
   (ename VARCHAR2,
    job VARCHAR2,
    mgr NUMBER,
    sal NUMBER,
    comm NUMBER,
    deptno NUMBER)

RETURN NUMBER IS
   new_empno NUMBER(4);
BEGIN
   SELECT empseq.NEXTVAL
      INTO new_empno
      FROM DUAL;
   INSERT INTO emp
      VALUES (new_empno, ename, job, mgr, sal, comm, deptno,
         tot_emps := tot_emps + 1;
   RETURN(new_empno);
END;

FUNCTION create_dept(dname VARCHAR2, loc VARCHAR2)
   RETURN NUMBER IS
      new_deptno NUMBER(4);
   BEGIN
      SELECT deptseq.NEXTVAL
         INTO new_deptno
```

```
         FROM dual;
      INSERT INTO dept
         VALUES (new_deptno, dname, loc);
            tot_depts := tot_depts + 1;
      RETURN(new_deptno);
   END;

PROCEDURE remove_emp(empno NUMBER) IS
   BEGIN
      DELETE FROM emp
      WHERE emp.empno = remove_emp.empno;
         tot_emps := tot_emps - 1;
   END;

PROCEDURE remove_dept(deptno NUMBER) IS
   BEGIN
      DELETE FROM dept
      WHERE dept.deptno = remove_dept.deptno;
         tot_depts := tot_depts - 1;
      SELECT COUNT(*)
         INTO tot_emps
         FROM emp;
         /* In case Oracle deleted employees from the EMP table
         to enforce referential integrity constraints, reset
         the value of the variable TOT_EMPS to the total
         number of employees in the EMP table. */
   END;

PROCEDURE increase_sal(empno NUMBER, sal_incr NUMBER) IS
   curr_sal NUMBER(7,2);
   BEGIN
      SELECT sal
      INTO curr_sal
      FROM emp
      WHERE emp.empno = increase_sal.empno;
      IF curr_sal IS NULL
         THEN RAISE no_sal;
      ELSE
         UPDATE emp
         SET sal = sal + sal_incr
         WHERE empno = empno;
      END IF;
   END;

PROCEDURE increase_comm(empno NUMBER, comm_incr NUMBER) IS
      curr_comm NUMBER(7,2);
   BEGIN
```

```
      SELECT comm
      INTO curr_comm
      FROM emp
      WHERE emp.empno = increase_comm.empno
      IF curr_comm IS NULL
         THEN RAISE no_comm;
      ELSE
         UPDATE emp
         SET comm = comm + comm_incr;
      END IF;
   END;

END emp_mgmt;
```

このパッケージ本体は、この章ですでに説明した CREATE PACKAGE 文の作成例のパッケージ仕様部と対応しています。パッケージ本体では、パッケージ仕様部で宣言した次のパブリック・プログラム・オブジェクトを定義しています。

- hire および create_dept の各ファンクション
- remove_emp、remove_dept、increase_sal および increase_comm の各プロシージャ

これらのオブジェクトは、パッケージ仕様部で宣言されているため、パッケージ外のアプリケーション・プログラム、プロシージャおよびファンクションからコールできます。たとえば、パッケージに対するアクセス権限がある場合には、increase_comm プロシージャをコールする emp_mgmt パッケージとは別に、increase_all_comms プロシージャを作成できます。

これらのオブジェクトはパッケージ本体で定義されているため、その定義を変更した場合でも、Oracle が依存スキーマ・オブジェクトを無効にすることはありません。たとえば、後で hire の定義を変更した場合に、increase_all_comms は実行するまで再コンパイルされません。

また、この例のパッケージ本体では、プライベート・プログラム・オブジェクトである変数 tot_emps および tot_depts が宣言されています。これらのオブジェクトは、パッケージ仕様部ではなくパッケージ本体で宣言されているため、パッケージ内の他のオブジェクトからアクセスできますが、パッケージ外からはアクセスできません。たとえば、変数 tot_depts の値を明示的に変更するアプリケーションは開発できません。ただし、ファンクション create_dept はパッケージの一部であるため、create_dept で tot_depts の値を変更できます。

CREATE PROCEDURE

用途

CREATE PROCEDURE 文は、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する場合に使用します。

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**（call spec）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 3 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 メソッドを問い合せます。引数および戻り値に対する型変換も問い合せます。

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：

- コール方法などのストアド・プロシージャの詳細は、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。
- 類似点が多いファンクションの詳細は、9-42 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、9-118 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、7-84 ページの「ALTER PROCEDURE」および 10-149 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、9-84 ページの「CREATE LIBRARY」を参照してください。
- 外部プロシージャの登録については、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

前提条件

プロシージャを作成する前に、ユーザー SYS は SQL スクリプト DBMSSTDX.SQL を実行する必要があります。このスクリプトの正確な名前と格納位置は、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。

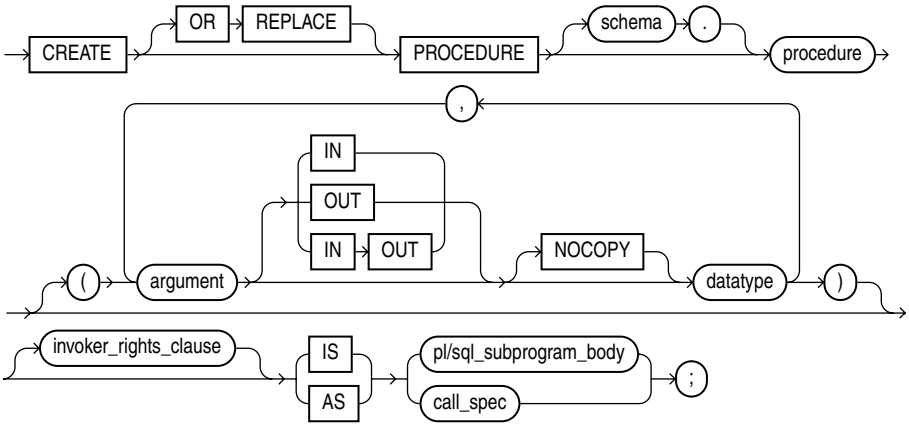
自スキーマ内にプロシージャを作成する場合は、CREATE PROCEDURE システム権限が必要です。他のユーザーのスキーマ内にプロシージャを作成する場合は、CREATE ANY PROCEDURE システム権限が必要です。他のユーザーのスキーマ内にプロシージャを再配置する場合は、ALTER ANY PROCEDURE システム権限が必要です。

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 コール仕様の C ライブラリには EXECUTE 権限が必要です）。

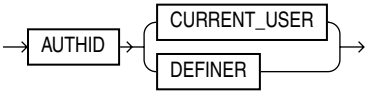
Oracle プリコンパイラ・プログラム内に CREATE PROCEDURE 文を埋め込む場合、キーワード END-EXEC と、その後に各言語用の埋込み SQL 文の終了記号を記述して文を終了する必要があります。

> **参照：** このような前提条件の詳細は、『Oracle8*i* PL/SQL ユーザーズ・ガイドおよびリファレンス』または『Oracle8*i* Java ストアド・プロシージャ開発者ガイド』を参照してください。

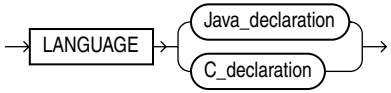
構文



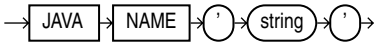
invoker_rights_clause::=



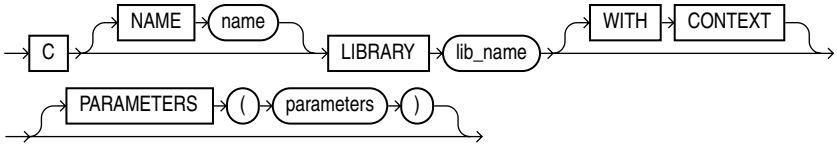
call_spec::=



Java_declaration::=



C_declaration::=



キーワードとパラメータ

OR REPLACE

既存のプロシージャを再作成する場合は、OR REPLACE を指定します。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、7-84 ページの「ALTER PROCEDURE」を参照してください。

schema

プロシージャを定義するスキーマを指定します。*schema* を省略した場合、現行のスキーマ内にプロシージャが作成されます。

procedure

作成するプロシージャの名前を指定します。

プロシージャの作成でコンパイル・エラーがある場合、Oracle はエラーを戻します。SQL*Plus コマンド SHOW ERRORS を使用して、関連するコンパイラ・エラー・メッセージを表示できます。

argument

argument
: プロシージャへの引数の名前を指定します。プロシージャが引数を受け入れない場合は、プロシージャ名の後のカッコを省略できます。

IN
: IN は、プロシージャをコールするときに、引数に値を指定する必要があることを示します。

OUT
: OUT は、プロシージャが、実行後にコールの環境に対して、指定した引数の値を戻すことを示します。

IN OUT
: IN OUT は、プロシージャのコール時に、引数に値を指定し、プロシージャの実行後にコール先の環境に値を渡す必要があることを示します。

  IN、OUT および IN OUT のいずれの引数も指定しない場合、デフォルトでは IN が設定されます。

NOCOPY
: NOCOPY は、できるだけ速く引数を渡すように指示します。この句は、OUT パラメータや IN OUT パラメータに対して、レコード、索引付き表、VARRAY などの大きい値を渡す際のパフォーマンスの向上に有効です（IN パラメータ値には、常に NOCOPY が渡されます）。

  - NOCOPY パラメータを指定すると、このパラメータに対応する実際の割当てとしてパッケージ変数が渡された場合に、パッケージ変数に対して行われた割当ては、すぐにこのパラメータに表示されます（または、このパラメータに対して行われた割当ては、すぐにパッケージ変数に表示されます）。
  - このパラメータまたは別のパラメータに対して行われた変更は、同じ変数が両方に渡された場合、両方の名前を介してすぐに参照できます。
  - プロシージャが未処理例外で終了した場合、このパラメータに対する割当ては、コール元の変数で参照できます。

  このような効果がないコールもあります。この効果に問題がない場合にのみ NOCOPY を使用してください。

datatype　引数のデータ型を指定します。引数には、PL/SQL でサポートされるデータ型を指定できます。

データ型には、データ長、精度または位取りは指定できません。たとえば、VARCHAR2(10) は無効ですが、VARCHAR2 は有効です。Oracle では、そのプロシージャがコールされた環境から引数のデータ長、精度および位取りを導出します。

invoker_rights_clause

invoker_rights_clause によって、プロシージャを、スキーマを所有するユーザーの権限でそのスキーマ内で実行するか、または CURRENT_USER の権限でそのスキーマ内で実行するかを指定できます。

この句は、問合せ、DML 操作およびプロシージャにおける動的 SQL 文の外部名の変換方法も指定します。

AUTHID CURRENT_USER　プロシージャを CURRENT_USER 権限で実行する場合は、CURRENT_USER を指定します。この句は、実行者権限プロシージャを作成します。

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、プロシージャを含むスキーマで変換します。

AUTHID DEFINER　プロシージャを含むスキーマの所有者権限でプロシージャを実行する場合、およびプロシージャを含むスキーマ内で外部名を変換する場合は、DEFINER を指定します。これはデフォルト値です。

参照 :

- 『Oracle8*i* PL/SQL 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。
- CURRENT_USER の判断方法については、『Oracle8*i* 概要』および『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

IS | AS

pl/sql_subprogram_body

PL/SQL サブプログラムでは、PL/SQL サブプログラム本体にあるプロシージャを宣言します。

参照：PL/SQL サブプログラムの詳細は、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

call_spec

call_spec を使用して、Java または C メソッド名、パラメータ・タイプおよび戻り型を SQL で相当するものにマップします。

Java_declaration では、'*string*' が JAVA 実装メソッドを定義します。

参照：

- 『Oracle8*i* Java ストアド・プロシージャ開発者ガイド』を参照してください。
- パラメータおよび *C_declaration* のセマンティクスの詳細は、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

AS EXTERNAL

AS EXTERNAL 句は、C メソッドを宣言するもう 1 つの方法です。この句は以前のリリースのもので、下位互換用にのみサポートされています。AS LANGUAGE C 構文を使用することをお薦めします。

例

CREATE PROCEDURE の例　次の文は、スキーマ sam 内にプロシージャ credit を作成します。

```
CREATE PROCEDURE sam.credit (acc_no IN NUMBER, amount IN NUMBER) AS
   BEGIN
      UPDATE accounts
      SET balance = balance + amount
      WHERE account_id = acc_no;
   END;
```

プロシージャ credit を実行すると、指定した金額が指定の預金口座に記入されます。このプロシージャをコールする場合、次の引数を指定する必要があります。

ACC_NO

この引数には預金口座の番号を指定します。この引数のデータ型は NUMBER です。

AMOUNT

この引数は預金の金額です。この引数のデータ型は NUMBER です。

このプロシージャでは、UPDATE 文を使用して引数 acc_no によって特定される口座に対して、accounts 表の balance 列の値を引数 amount の値分増加させます。

次の文では、外部プロシージャ c_find_root によって、ポインタがパラメータとみなされます。プロシージャ find_root は、BY REF 句を使用した参照によって、そのパラメータを渡します。

```
CREATE PROCEDURE find_root
   ( x IN REAL )
   IS LANGUAGE C
      NAME "c_find_root"
      LIBRARY c_utils
      PARAMETERS ( x BY REF );
```

CREATE PROFILE

用途

CREATE PROFILE 文は、プロファイルを作成する場合に使用します。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、『Oracle8*i* 管理者ガイド』を参照してください。

前提条件

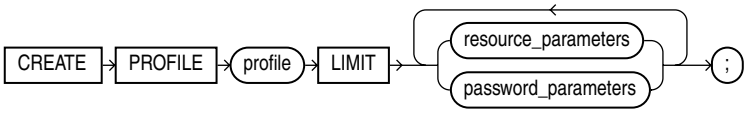
CREATE PROFILE システム権限が必要です。

ユーザーに対するリソース制限を指定する場合、次の条件が必要です。

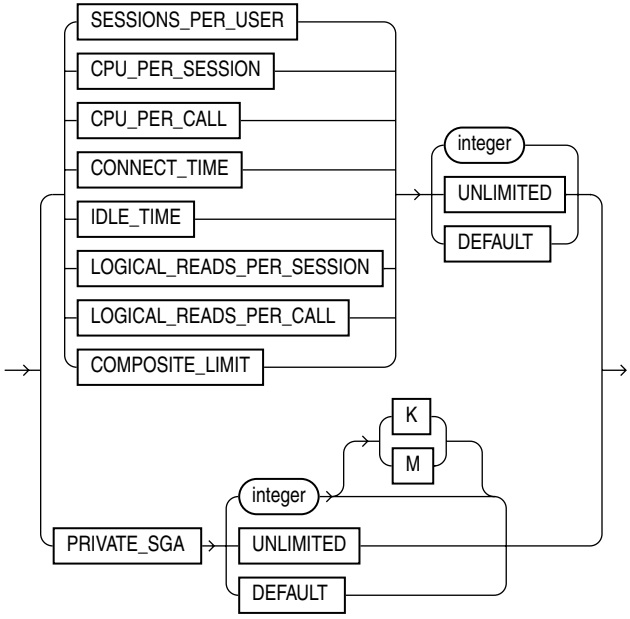
- ALTER SYSTEM 文または初期化パラメータ RESOURCE_LIMIT で動的にリソース制限を使用可能にします（このパラメータは、パスワード・リソースには適用されません。パスワード・リソースは、常に使用可能です）。
- CREATE PROFILE 文を使用して、制限を定義するプロファイルを作成します。
- CREATE USER または ALTER USER 文を使用して、プロファイルを割り当てます。

> **参照：**
>
> - 動的にリソース制限を使用可能にするには、7-123 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、『Oracle8*i* リファレンス・マニュアル』を参照してください。
> - プロファイルについては、10-99 ページの「CREATE USER」および8-87 ページの「ALTER USER」を参照してください。

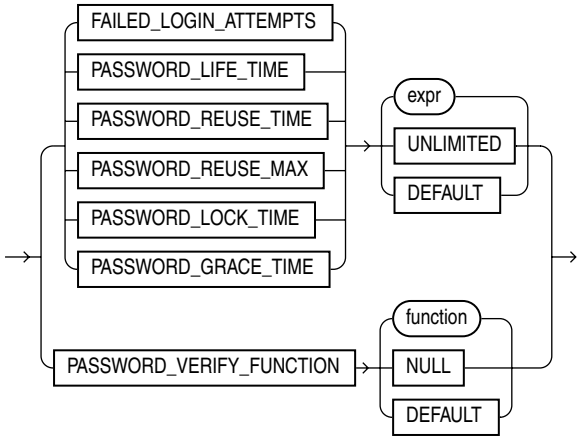
構文



resource_parameters::=



password_parameters::=



キーワードとパラメータ

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 では、次の方法でリソース制限を割り当てます。

- 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判断され、現行文がロールバックされ、エラーが戻されます。この後、ユーザーは、現行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 では、実際にその制限が使用可能になってから、リソース制限が割り当てられます。

UNLIMITED

リソース・パラメータで指定した場合、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ることを指定します。パスワード・パラメータで指定した場合は、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

このプロファイルにあるリソースの制限を指定しない場合は、DEFAULT を指定します。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 セッションあたりの 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）あたりの 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 セッションあ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同期を制限します。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 文を処理するコール（解析、実行またはフェッチ）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 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をバイト単位で指定します。K または M を使用すると、この制限を KB または MB 単位で指定できます。 **注意 :** この制限は、マルチスレッド・サーバー・アーキテクチャを使用している場合にのみ有効です。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 セッションあ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については、7-91 ページの「ALTER RESOURCE COST」を参照してください。

password_parameters

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。

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。この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。

PASSWORD_REUSE_TIME　パスワードが再使用できなくなるまでの日数を指定します。PASSWORD_REUSE_TIME を整数値に設定する場合は、PASSWORD_REUSE_MAX を UNLIMITED に設定する必要があります。

PASSWORD_REUSE_MAX　現行のパスワードを再使用する前に必要な、パスワードの変更回数を指定します。PASSWORD_REUSE_MAX を整数値に設定する場合は、PASSWORD_REUSE_TIME を UNLIMITED に設定する必要があります。

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。

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。猶予期間中にパスワードが変更されない場合、そのパスワードは使用できなくなります。

PASSWORD_VERIFY_FUNCTION　PASSWORD_VERIFY_FUNCTION 句によって、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を可能にします。Oracle 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*　複雑なパスワード検証ルーチンの名前を指定します。
- NULL　パスワードの検証が行われないことを指定します。

パスワード・パラメータに関する制限事項

- PASSWORD_REUSE_TIME を整数値に設定する場合、PASSWORD_REUSE_MAX を UNLIMITED に設定する必要があります。PASSWORD_REUSE_MAX を整数値に設定する場合、PASSWORD_REUSE_TIME を UNLIMITED に設定する必要があります。
- PASSWORD_REUSE_TIME および PASSWORD_REUSE_MAX の両方を UNLIMITED に設定した場合、どちらのパスワードのリソースも使用されません。
- PASSWORD_REUSE_MAX を DEFAULT に設定し、PASSWORD_REUSE_TIME を UNLIMITED に設定した場合、DEFAULT プロファイルに定義された PASSWORD_REUSE_MAX 値が使用されます。
- PASSWORD_REUSE_TIME を DEFAULT に設定し、PASSWORD_REUSE_MAX を UNLIMITED に設定した場合、DEFAULT プロファイルに定義された PASSWORD_REUSE_TIME 値が使用されます。
- PASSWORD_REUSE_TIME および PASSWORD_REUSE_MAX の両方を DEFAULT に設定した場合、DEFAULT プロファイルに定義された値はどちらでも使用されます。

例

CREATE PROFILE の例　次の文は、プロファイル prof を作成します。

```
CREATE PROFILE prof
  LIMIT PASSWORD_REUSE_MAX DEFAULT
        PASSWORD_REUSE_TIME UNLIMITED;
```

リソース制限の設定例　次の文は、プロファイル system_manager を作成します。

```
CREATE PROFILE system_manager
   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 system_manager 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 セッションにおいて、ユーザーは CPU 時間を無制限に消費できます。
- ユーザーが作成した 1 コールで消費できる CPU 時間は 30 秒以下です。
- 1 セッションで継続できる時間は 45 分以下です。
- 1 セッションのメモリーおよびディスクのデータ・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。
- ユーザーが作成した 1 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 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 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。
- system_manager プロファイルでは、IDLE_TIME 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対象リソースの制限を受けます。

パスワード制限の設定例　次の文は、パスワード・プロファイルの制限が設定されたプロファイル myprofile を作成します。

```
CREATE PROFILE myprofile LIMIT
   FAILED_LOGIN_ATTEMPTS 5
   PASSWORD_LIFE_TIME 60
   PASSWORD_REUSE_TIME 60
   PASSWORD_REUSE_MAX UNLIMITED
   PASSWORD_VERIFY_FUNCTION verify_function
   PASSWORD_LOCK_TIME 1/24
   PASSWORD_GRACE_TIME 10;
```

CREATE ROLE

用途

CREATE ROLE 文は、**ロール**を作成する場合に使用します。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。その結果、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。

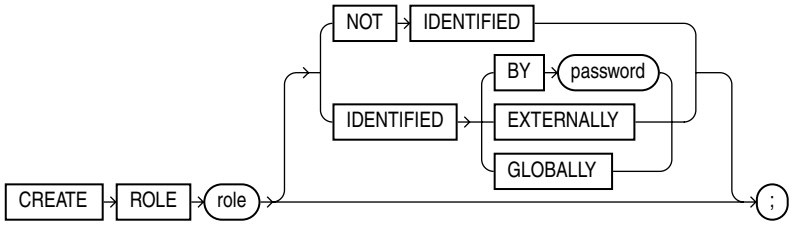
参照：

- ロールの付与については、11-31 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、8-87 ページの「ALTER USER」を参照してください。
- ロールの変更については、7-94 ページの「ALTER ROLE」を参照してください。
- データベースからのロールの削除については、10-153 ページの「DROP ROLE」を参照してください。
- カレント・セッションに対するロールの使用可能および使用禁止については、11-122 ページの「SET ROLE」を参照してください。
- グローバル・ロールの使用方法の詳細は、『Oracle8*i* 分散システム』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



キーワードとパラメータ

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。

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は、11-31 ページの「GRANT」を参照してください。

NOT IDENTIFIED

NOT IDENTIFIED を指定すると、この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

IDENTIFIED 句を指定すると、SET ROLE 文によってロールを使用可能にする前に、ユーザーが、指定したメソッドによって認可されている必要があります。

BY *password*
: BY *password* 句によって、**ローカル・ユーザー**を作成します。また、ロールを使用可能にするときに、ユーザーがパスワードを指定する必要があることを示します。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することもできます。

EXTERNALLY
: EXTERNALLY によって**外部ユーザー**を作成します。また、ロールを使用可能にする前に、ユーザーが（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）外部サービスで認可されている必要があることを示します。

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、ロールを使用可能にできない場合もあります。

GLOBALLY　GLOBALLY によって、**グローバル・ユーザー**を作成します。また、SET ROLE 文を使用して、またはログイン時にロールを使用可能にする前に、ユーザーがエンタープライズ・ディレクトリ・サービスによってロールの使用を許可されている必要があることを示します。

NOT IDENTIFIED 句および IDENTIFIED 句の両方を省略した場合、ロールには NOT IDENTIFIED がデフォルト値として使用されます。

例

CREATE ROLE の例　次の文は、グローバル・ロール vendor を作成します。

```
CREATE ROLE vendor IDENTIFIED GLOBALLY;
```

次の文は、teller ロールを作成します。

```
CREATE ROLE teller
   IDENTIFIED BY cashflow;
```

この後、teller ロールを付与されたユーザーは、パスワード cashflow 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。

CREATE ROLLBACK SEGMENT

用途

CREATE ROLLBACK SEGMENT 文は、ロールバック・セグメントを作成する場合に使用します。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 が使用するオブジェクトです。

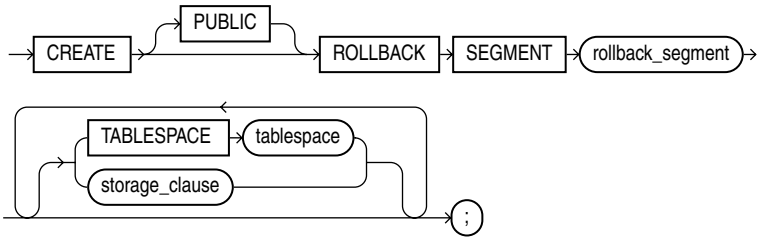
参照：

- ロールバック・セグメントの変更については、7-96 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、10-154 ページの「DROP ROLLBACK SEGMENT」を参照してください。

前提条件

CREATE ROLLBACK SEGMENT システム権限が必要です。

構文



storage_clause: 11-129 ページの「storage_clause」を参照してください。

キーワードとパラメータ

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で、すべてのインスタンスに対して使用可能になります。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 句を使用して、ロールバック・セグメントが作成される表領域を識別します。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。

制限事項 : システム管理されている表領域にロールバック・セグメントは作成できません（EXTENT MANAGEMENT LOCAL AUTOALLOCATE を指定した作成時）。

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 インスタンスによって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、ROLLBACK_SEGMENTS 初期化パラメータの値にそのセグメントの名前を追加してください。

参照 :

- 10-56 ページの「CREATE TABLESPACE」を参照してください。
- ロールバック・セグメントを作成および使用可能にする方法については、『Oracle8*i* 管理者ガイド』を参照してください。

storage_clause

storage_clause によって、ロールバック・セグメントの特性を指定できます。

注意：

- *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ため、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： 11-129 ページの「storage_clause」を参照してください。

例

CREATE ROLLBACK SEGMENT の例 次の文は、システム表領域内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 ROLLBACK SEGMENT rbs_2
    TABLESPACE system;
```

この文は、次の文と同じ結果になります。

```
CREATE ROLLBACK SEGMENT rbs_2
   TABLESPACE system
   STORAGE
   ( INITIAL 10K
     NEXT 10K
     MAXEXTENTS UNLIMITED);
```

CREATE SCHEMA

用途

CREATE SCHEMA は、複数の表およびビューを作成し、1 つのトランザクションで複数の権限の付与を行う場合に使用します。

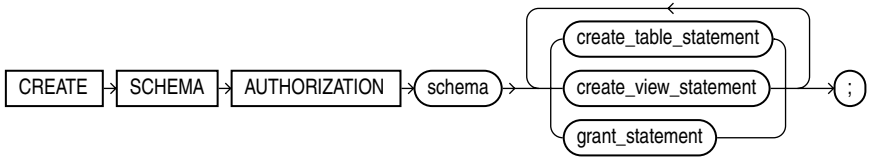
CREATE SCHEMA 文を実行すると、挿入されている個々の文が実行されます。すべての文が正常に実行された場合、そのトランザクションがコミットされます。文の結果が 1 つでもエラーになった場合は、すべての文がロールバックされます。

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（10-99 ページの「CREATE USER」を参照してください）。この文によって、表およびビューとともにスキーマが作成され、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、これらのオブジェクトに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



キーワードとパラメータ

schema

スキーマの名前を指定します。スキーマ名は、Oracle 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：** 10-7 ページの「CREATE TABLE」を参照してください。

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：** 10-105 ページの「CREATE VIEW」を参照してください。

grant_statement

この CREATE SCHEMA 文の一部として発行する GRANT *object_privileges* 文を指定します。この文の終りには、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：** 11-31 ページの「GRANT」を参照してください。

CREATE SCHEMA 文は、Oracle 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。

制限事項：*parallel_clause* の構文は、CREATE SCHEMA の CREATE TABLE 文に適用されますが、オブジェクトの作成時に並列度は使用されません。

> **参照：** 10-40 ページの「CREATE TABLE」の parallel_clause を参照してください。

例

CREATE SCHEMA の例　次の文は、ユーザー Blair に対して blair という名前のスキーマを作成します。その後、SOX 表を作成し、red_sox ビューを作成し、ユーザー waites に対して red_sox ビューについての SELECT 権限を付与します。

```
CREATE SCHEMA AUTHORIZATION blair
   CREATE TABLE sox
      (color VARCHAR2(10)  PRIMARY KEY, quantity NUMBER)
   CREATE VIEW red_sox
      AS SELECT color, quantity FROM sox WHERE color = 'RED'
   GRANT select ON red_sox TO waites;
```

CREATE SEQUENCE

用途

CREATE SEQUENCE 文は、**順序**を作成する場合に使用します。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一度生成すると、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続きその値にアクセスすることができます。

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：

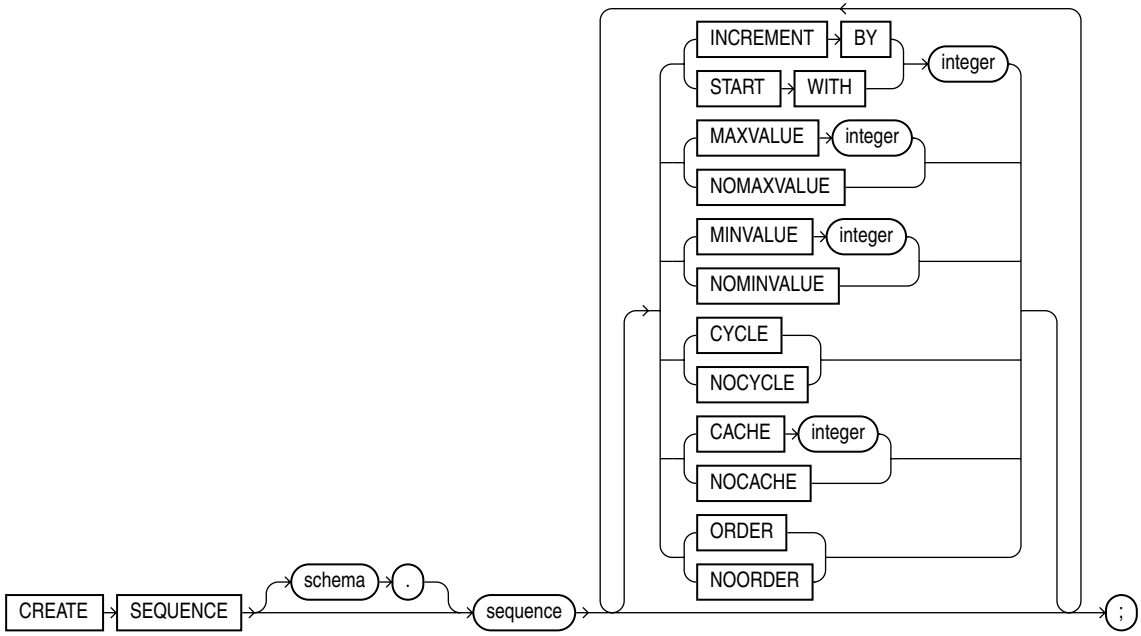
- CURRVAL および NEXTVAL の使用方法については、2-57 ページの「疑似列」を参照してください。
- 順序の使用方法については、2-59 ページの「順序値の使用方法」を参照してください。
- 順序の変更または削除については、7-99 ページの「ALTER SEQUENCE」または 11-3 ページの「DROP SEQUENCE」を参照してください。

前提条件

自スキーマ内に順序を作成する場合は、CREATE SEQUENCE 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 権限が必要です。

構文



キーワードとパラメータ

schema

順序を含むスキーマを指定します。*schema* を省略した場合、自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には、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。

順序パラメータ

INCREMENT BY *integer*
: 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。この増分値が正の場合、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 *integer*
: 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 *integer*
: 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 は、順序の最大値を、昇順の場合は 10^{27}、降順の場合は -1 に指定します。これはデフォルト値です。

MINVALUE *integer*
: 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 は、順序の最小値を、昇順の場合は 1、降順の場合は -（10^{26}）に指定します。これはデフォルト値です。

CYCLE　CYCLE は、順序が最大値または最小値に達しても、引き続き値を生成することを示します。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 は、順序が最大値または最小値に達した場合は、それ以上の値を生成できないことを示します。これはデフォルト値です。

CACHE *integer*　より高速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値を指定します。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。

NOCACHE　NOCACHE は、順序の値が事前に割り当てられていないことを示します。

CACHE パラメータおよび NOCACHE オプション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 は、要求どおりの順で順序番号を生成することを保証する場合に指定します。順序番号をタイムスタンプとして使用する場合に、この句を使用できます。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。

ORDER オプションは、Parallel Server を使用する Oracle をパラレル・モードで使用する場合、生成順序を保証するためにのみ指定してください。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 は、要求どおりの順で順序番号を生成することを保証しない場合に指定します。これはデフォルト値です。

例

CREATE SEQUENCE の例　次の文は、順序 eseq を作成します。

```
CREATE SEQUENCE eseq
  INCREMENT BY 10;
```

最初に eseq.nextval を参照した場合、1 が戻されます。次に参照した場合、11 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 10 大きい値が戻されます。

10

SQL 文 : CREATE SYNONYM ～ DROP ROLLBACK SEGMENT

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 TEMPORARY TABLESPACE
- CREATE TRIGGER
- CREATE TYPE
- CREATE TYPE BODY
- CREATE USER
- CREATE VIEW
- DELETE
- DISASSOCIATE STATISTICS
- DROP CLUSTER
- DROP CONTEXT
- DROP DATABASE LINK
- DROP DIMENSION
- DROP DIRECTORY
- 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 ROLE
- DROP ROLLBACK SEGMENT

CREATE SYNONYM

用途

CREATE SYNONYM 文は、シノニムを作成する場合に使用します。シノニムとは、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、Java クラス・スキーマ・オブジェクト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、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。

表 10-1 に、シノニムを参照できる SQL 文を示します。

表 10-1 シノニムの使用方法

DML 文	DDL 文
SELECT	AUDIT
INSERT	NOAUDIT
UPDATE	GRANT
DELETE	REVOKE
EXPLAIN PLAN	COMMENT
LOCK TABLE	

参照： シノニムの概要については、『Oracle8*i* 概要』を参照してください。

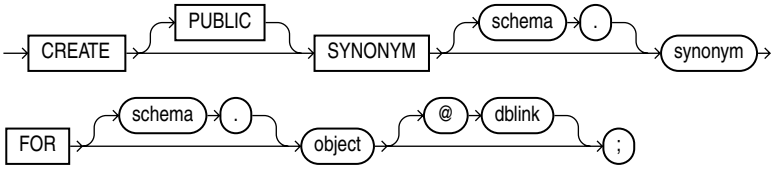
前提条件

自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。

PUBLIC 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



キーワードとパラメータ

PUBLIC

PUBLIC を指定して、パブリック・シノニムを作成します。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。

オブジェクトの先頭にスキーマ名が指定されていない場合、および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は、オブジェクトの参照を変換するときのみ、パブリック・シノニムが使用されます。

この句を省略した場合、シノニムはプライベートとなり、定義しているスキーマ内に限りアクセスできます。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。

schema

シノニムを含むスキーマを指定します。*schema* を省略した場合、自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： シノニム名の最大長は 32 バイトです。30 バイトを超える名前は、Java の機能にのみ使用できます。30 バイトを超える名前を指定した場合、名前は暗号化され、暗号化された表現でデータ・ディクショナリに格納されます。実際の暗号にはアクセスできません。また、元の指定とデータ・ディクショナリの表現のどちらも、シノニム名としては使用できません。

FOR *object*

シノニムを作成するオブジェクトを指定します。オブジェクトに *schema* を指定しなかった場合、そのスキーマ・オブジェクトは自スキーマ内にあるとみなされます。スキーマ・オブジェクトには、次のものを指定できます。

- 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
- 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。

制限事項 :

- スキーマ・オブジェクトは、パッケージに入れることはできません。
- オブジェクト型にはシノニムを作成できません。

dblink

dblink を完全に指定するか、または *dblink* の一部を指定した場合、スキーマ・オブジェクトが格納されているリモート・データベース上でそのスキーマ・オブジェクトのシノニムを作成することができます。*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・リンクで指定したスキーマ内のオブジェクトを参照します。リモート・データベースでは、オブジェクトを定義し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。

制限事項 : Java クラス・シノニムには *dblink* を指定できません。

参照 :

- データベース・リンクの参照方法の詳細は、2-88 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、9-28 ページの「CREATE DATABASE LINK」を参照してください。

例

CREATE SYNONYM の例　スキーマ scott 内の market_research 表に対してシノニム market を定義する場合は、次の文を発行します。

```
CREATE SYNONYM market
   FOR scott.market_research;
```

リモート・データベース SALES 上のスキーマ scott 内の emp 表に対して PUBLIC シノニムを作成する場合は、次の文を発行します。

```
CREATE PUBLIC SYNONYM emp
   FOR scott.emp@sales;
```

別のスキーマ内にベース表が定義されている場合は、ベース表と同じ名前をシノニムに指定することもできます。

シノニムの変換例　Oracle は、オブジェクトの参照を、PUBLIC 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 scott およびスキーマ blake にそれぞれ dept という名前の表があり、ユーザー SYSTEM が blake.dept に対して dept という名前の PUBLIC シノニムを作成するとします。この場合、ユーザー scott が次の文を発行すると、scott.dept の行が戻されます。

```
SELECT * FROM dept;
```

blake.dept の行を検索する場合、ユーザー scott は、次のようにスキーマ名の後に dept を指定する必要があります。

```
SELECT * FROM blake.dept;
```

ユーザー adam のスキーマに dept という名前のオブジェクトが定義されていない場合、adam は、パブリック・シノニム dept を使用して、blake のスキーマ内の dept 表にアクセスできます。

```
SELECT * FROM dept;
```

CREATE TABLE

用途

CREATE TABLE 文は、次の型の表を作成する場合に使用します。

- **リレーショナル表**は、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**は、列の定義にオブジェクト型を使用する表です。オブジェクト表とは、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た表です。

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。

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：** オブジェクト作成の詳細は、『Oracle8*i* アプリケーション開発者ガイド 基礎編』および 10-80 ページの「CREATE TYPE」を参照してください。

前提条件

自スキーマ内に**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、表を定義してい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を持つ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 GRANT OPTION 付きの EXECUTE 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。

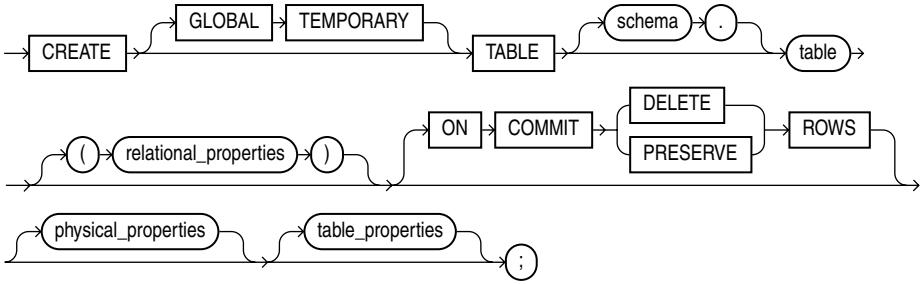
一意または主キー制約を有効にする場合は、表に索引を作成するための権限が必要です。Oracle は、その表を含むスキーマにある一意または主キー列に索引を作成するため、この権限が必要になります。

参照：

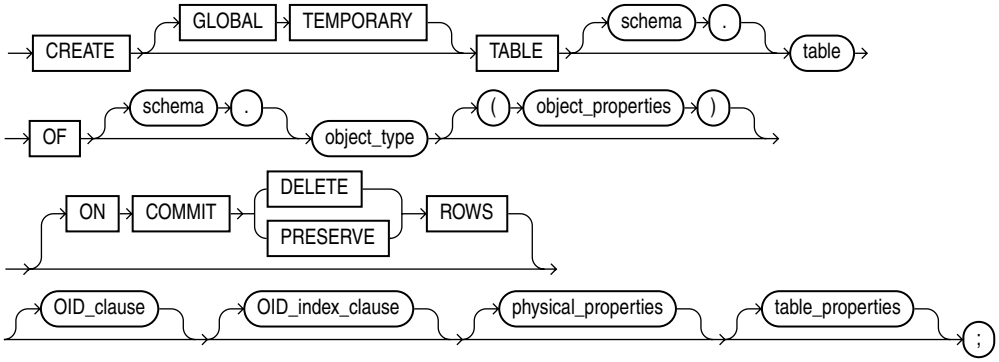
- 9-51 ページの「CREATE INDEX」を参照してください。
- 型を使用する表の作成に必要な権限については、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

構文

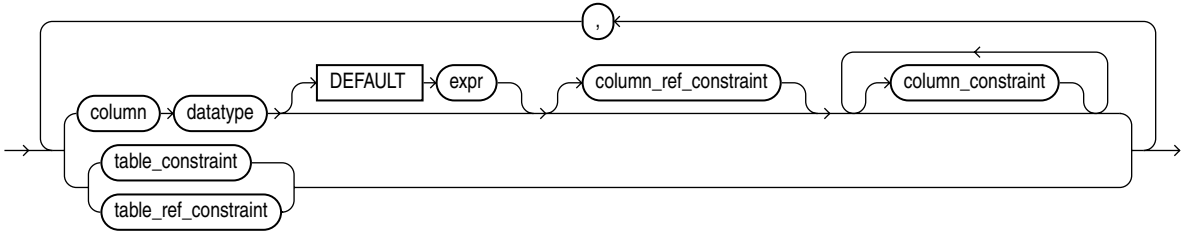
relational_table::=



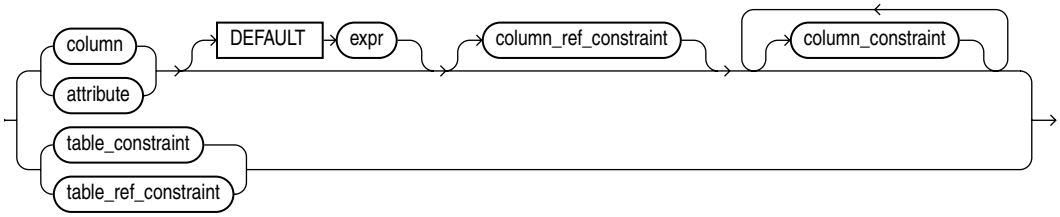
object_table::=



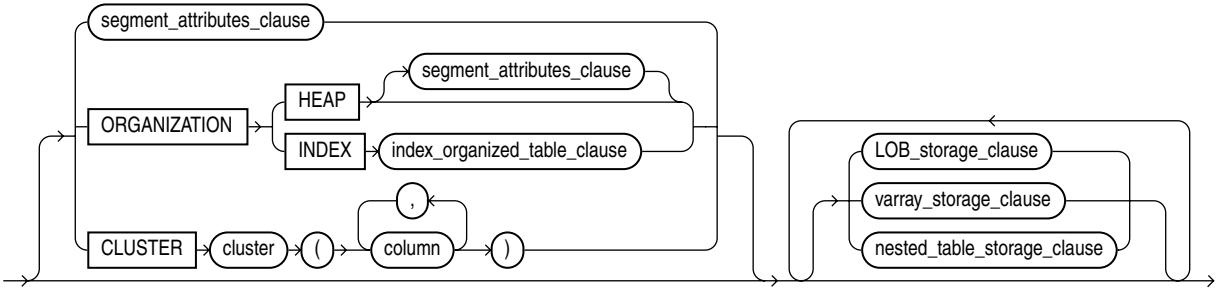
relational_properties::=



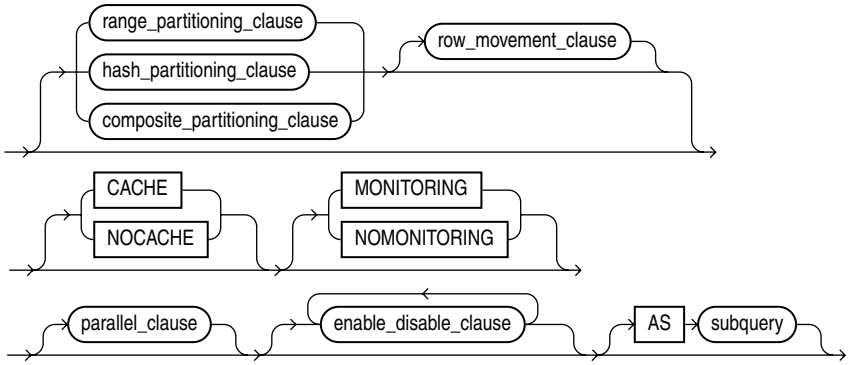
object_properties::=



physical_properties::=



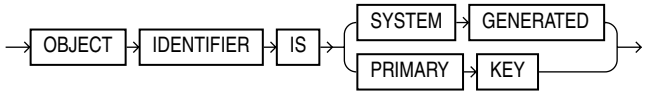
table_properties::=



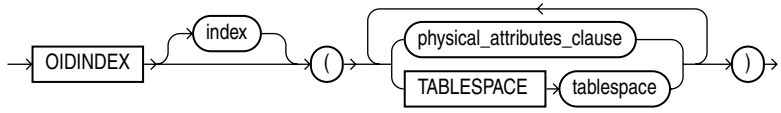
subquery::= 11-88 ページの「SELECT および副問合せ」を参照してください。

table_constraint, **column_constraint**, **table_ref_constraint**, **column_ref_constraint**, **constraint_state**: 8-134 ページの「constraint_clause」を参照してください。

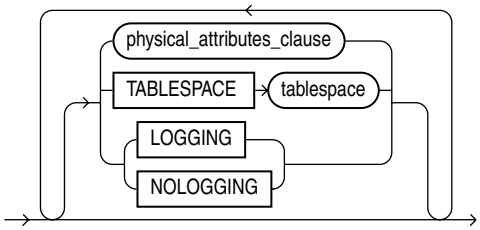
OID_clause::=



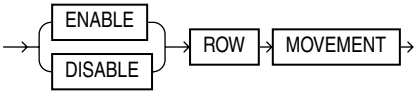
OID_index_clause::=



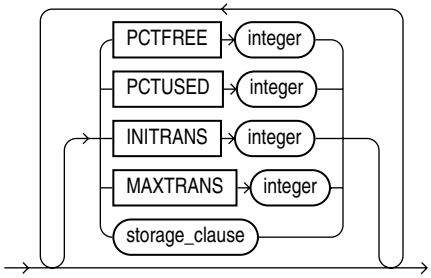
segment_attributes_clause:=



row_movement_clause::=

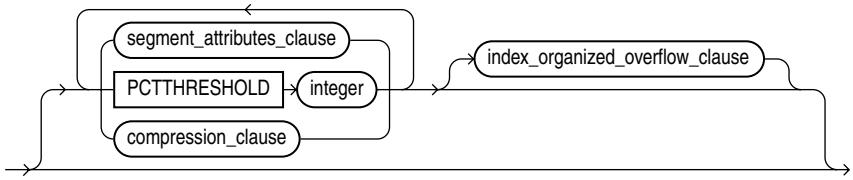


physical_attributes_clause::=

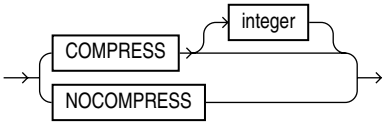


storage_clause: 11-129 ページの「storage_clause」を参照してください。

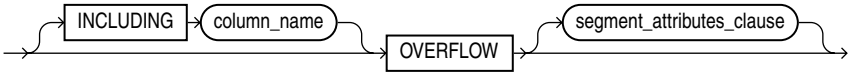
index_organized_table_clause::=



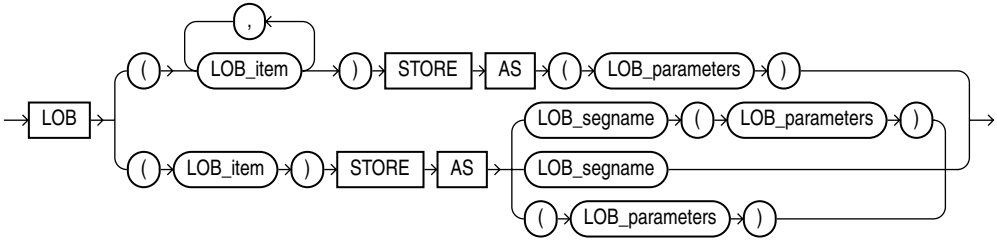
compression_clause::=



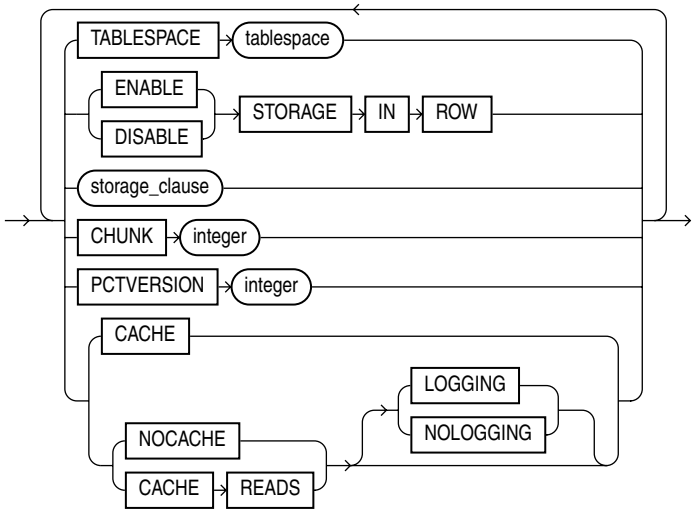
index_organized_overflow_clause::=



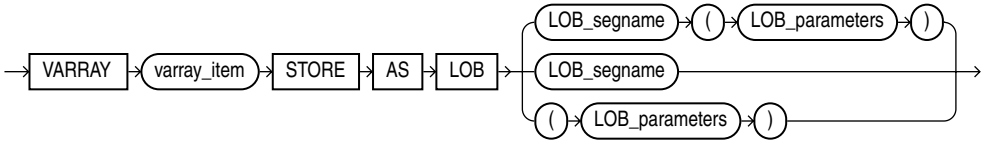
LOB_storage_clause::=



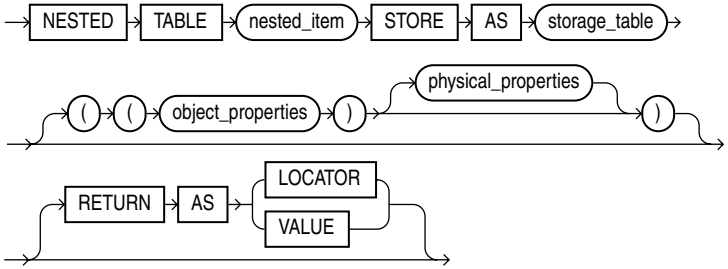
LOB_parameters::=



varray_storage_clause::=

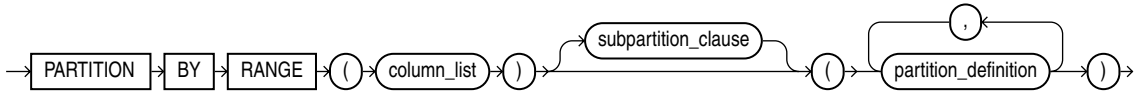


nested_table_storage_clause::=

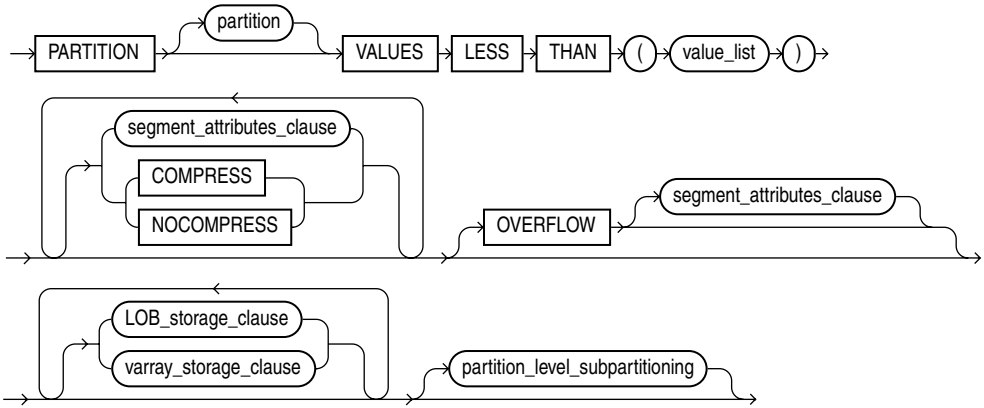


range_partitioning_clause::=

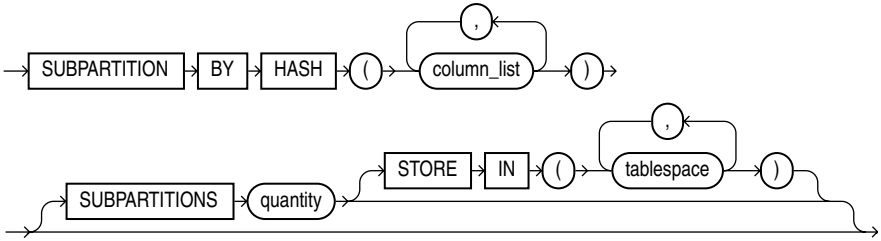
composite_partitioning_clause::=



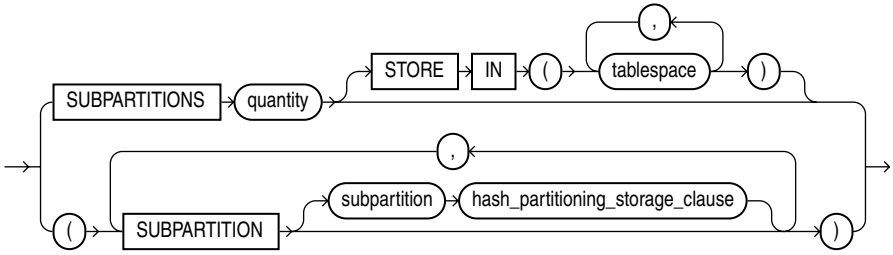
partition_definition::=



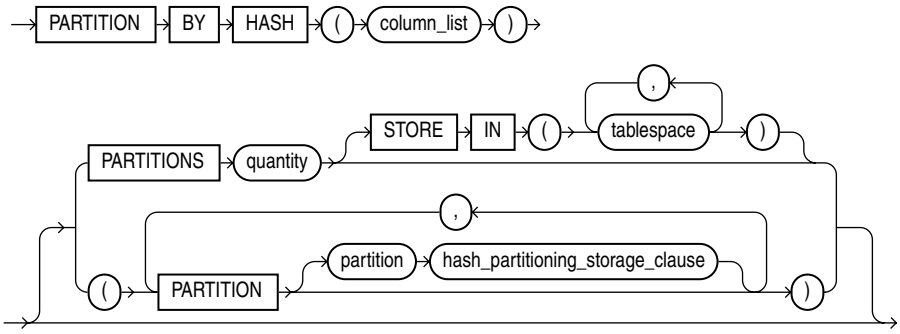
subpartition_clause::=



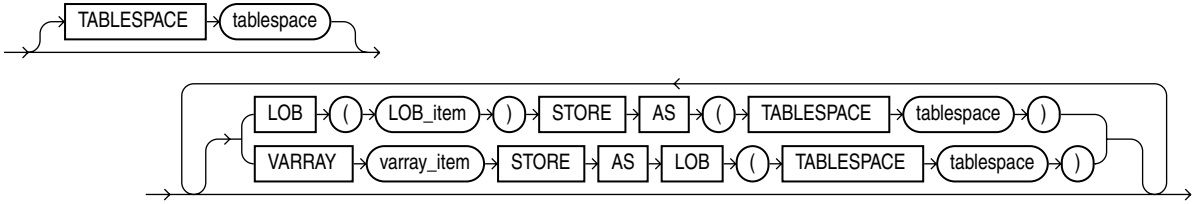
partition_level_subpartitioning::=



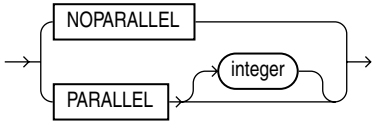
hash_partitioning_clause::=



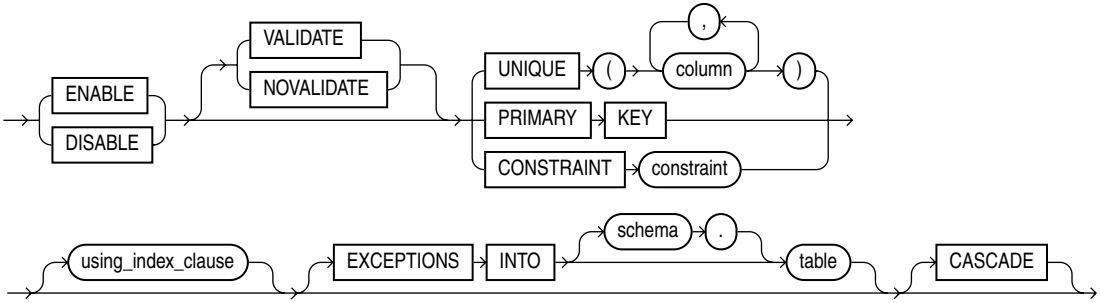
hash_partitioning_storage_clause::=



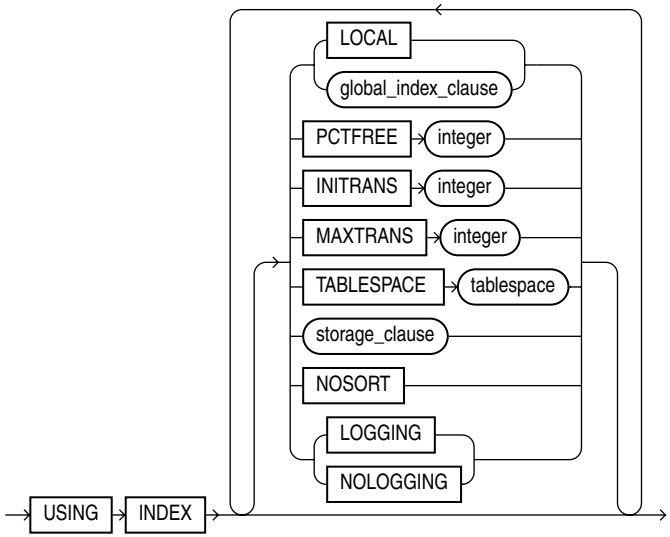
parallel_clause::=



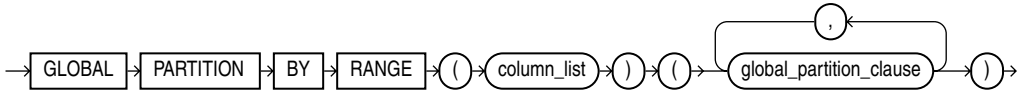
enable_disable_clause::=



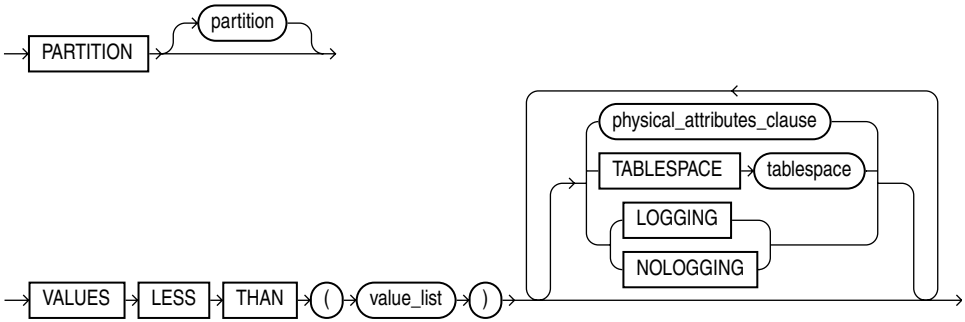
using_index_clause::=



global_index_clause::=



global_partition_clause::=



キーワードとパラメータ

GLOBAL TEMPORARY

GLOBAL TEMPORARY を指定して、表が一時的で、その定義がすべてのセッションで参照できることを示します。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。

一時表は、標準表の定義に準拠する定義を持ちますが、**セッション固有**または**トランザクション固有**データのいずれかを含みます。データがセッション固有かトランザクション固有かは、ON COMMIT キーワードで指定します。

参照 : 一時表の詳細は、『Oracle8*i* 概要』を参照してください。

制限事項 :

- 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、参照整合性（外部キー）制約を指定できません。
- 一時表は、ネストした表または VARRAY 型の列を含むことはできません。

- *LOB_storage_clause* の TABLESPACE、*storage_clause*、LOGGING または NOLOGGING、MONITORING、NOMONITORING または *LOB_index_clause* 句は、指定できません。
- 一時表にパラレル DML およびパラレル問合せはサポートされていません（パラレル・ヒントは無視されます。*parallel_clause* を指定すると、エラーが戻されます）。
- *segment_attributes_clause*、*nested_table_storage_clause* または *parallel_clause* は指定できません。
- 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

表を含むスキーマを指定します。*schema* を省略した場合、自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。

OF *object_type*

OF 句によって、暗黙的に *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（OID）が割り当てられます。*schema* を省略した場合、オブジェクト表が自スキーマ内に作成されます。

オブジェクト表に常駐するオブジェクトは参照可能です。

参照：

- オブジェクト作成の詳細は、10-80 ページの「CREATE TYPE」を参照してください。
- REF 型の使用方法の詳細は、2-23 ページの「ユーザー定義型のカテゴリ」、4-128 ページの「ユーザー定義ファンクション」、5-2 ページの「式」、10-80 ページの「CREATE TYPE」および『Oracle8*i* 管理者ガイド』を参照してください。

relational_properties

column

表の列の名前を指定します。

AS *subquery* を指定すると、索引構成表（IOT）を作成しない限り、*column* および *datatype* を省略できます。IOT 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。ただし、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。このような表における総列数の計算方法については、『Oracle8*i* 管理者ガイド』を参照してください。

datatype

列のデータ型を指定します。

参照 : 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。

制限事項 :

- パーティション化された索引構成表（IOT）に、LOB 列または VARRAY 型の列を指定することはできません。パーティション化されていない IOT のデータ型は、制限されません。
- 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列 ID であることは保証されません。

注意 : 次の場合は、*datatype* は省略できます。

- AS *subquery* を指定する場合（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。）
- 文が、参照整合性制約で外部キーとして列を指定する場合（Oracle 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。）

DEFAULT

DEFAULT 句によって、後続の INSERT 文が列の値を省略した場合に値が列に割り当てられるように指定できます。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。

制限事項 : DEFAULT 式には、他の列（疑似列 CURRVAL、NEXTVAL、LEVEL、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は指定できません。

参照 : *expr* の構文については、5-2 ページの「式」を参照してください。

table_ref_constraint

および

column_ref_constraint

REF 型の列について詳細に指定します。これらの句の違いは、表レベルから *table_ref* を指定することのみです。したがって、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。REF 型の列または属性を識別した後、*column_ref* を指定してください。

参照 : これらの制約の構文および詳細は、8-134 ページの「constraint_clause」を参照してください。

column_constraint

column_constraint を使用して、整合性制約を列定義の一部として定義します。

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。

参照 : 8-134 ページの「constraint_clause」の *column_constraint* の構文を参照してください。

table_constraint

table_constraint を使用して、整合性制約を表定義の一部として定義します。

参照 : 8-134 ページの「constraint_clause」の *table_constraint* の構文の説明を参照してください。

注意 : DEFERRABLE 以外の主キー制約を索引構成表に指定してください。

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 オブジェクト内の項目の修飾した列名を指定します。

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。この句は、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します。

DELETE ROWS トランザクション固有の一時表に DELETE ROWS を指定します（デフォルト）。各コミット後に表を切り捨てます（すべての行を削除します）。

PRESERVE ROWS セッション固有の一時表に対して、PRESERVE ROWS を指定します。セッション終了時に表を切り捨てます（すべての行を削除します）。

OID_clause

OID_clause によって、オブジェクト表のオブジェクト識別子（OID）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。

制限事項 :

- 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 OID はローカルで（グローバルである必要はありません）一意です。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 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。指定しないと、名前は自動生成されます。

physical_properties

segment_attributes_clause

physical_attributes_clause

physical_attributes_clause は、PCTFREE、PCTUSED、INITRANS および MAXTRANS パラメータの値、および表の記憶特性を指定します。

- 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、指定した各パラメータおよび記憶特性は、パーティションを作成する文の PARTITION 句で明示的にオーバーライドしない限り、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の物理属性となります。

PCTFREE *integer*

表、オブジェクト表の OID 索引、パーティションそれぞれのデータ・ブロックの中で、表の行に対して今後行われる更新用に確保する領域が占める割合（パーセント）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。

PARTITION 記述句の中での PCTFREE の機能と、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・ログの作成および変更を行う文の中での PCTFREE の機能は同じです。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。

PCTUSED *integer*

使用領域のうち、表、オブジェクト表 OID 索引、索引構成表のオーバーフロー・データ・セグメントのデータ・ブロックごとに、Oracle によって確保される最小限の使用領域の割合（パーセント）を指定します。ブロックは、使用領域が PCTUSED の値を下回ると、行挿入の対象となります。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。

PARTITION 記述句の中での PCTUSED の機能と、クラスタ、マテリアライズド・ビュー、マテリアライズド・ビュー・ログの作成および変更を行う文の中での PCTUSED の機能は同じです。

PCTUSED は、索引構成表（ORGANIZATION INDEX）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、表内の領域を効果的に利用できます。

> **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、『Oracle8*i* パフォーマンスのための設計およびチューニング』を参照してください。

INITRANS *integer*

表、オブジェクト表 OID 索引、パーティション、LOB の索引セグメントまたはオーバーフロー・データ・セグメントに割り当てられた各データ・ブロック内の初期トランザクション・エントリ数を指定します。この値の範囲は 1 ～ 255 までで、デフォルト値は 1 です。通常、このデフォルトの INITRANS 値を変更する必要はありません。

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。トランザクション・エントリのサイズは、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。

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、PARTITION 記述、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・ログの中では、表の場合と同じ働きをします。クラスタまたは索引の場合、INITRANS の最小値およびデフォルト値は 1 ではなく 2 です。

MAXTRANS *integer*

表、オブジェクト表 OID 索引、パーティション、LOB の索引セグメント、索引構成オーバーフロー・データ・セグメントに割り当てられたデータ・ブロックを更新できる同時実行トランザクションの最大数を指定します。この制限は問合せには適用されません。この値の範囲は 1 ～ 255 までで、デフォルト値はデータ・ブロック・サイズの関数となります。MAXTRANS 値は、変更せずにデフォルトのまま使用してください。

ブロックを同時に更新するトランザクション数が INITRANS 値を超えると、MAXTRANS 値を超えるまで、またはブロックの空き領域がなくなるまで、そのブロックにトランザクション・エントリが動的に割り当てられます。

MAXTRANS パラメータは、PARTITION 記述、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・ログの中では、表の場合と同じ働きをします。

storage_clause

storage_clause によって、表、オブジェクト表 OID 索引、パーティション、LOB の記憶域、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。

参照：11-129 ページの「storage_clause」を参照してください。

TABLESPACE

Oracle が、表、オブジェクト表 OID 索引、パーティション、LOB の記憶域、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項目が作成されます。

1 つ以上の LOB 列を持つヒープ構成表の場合、LOB 記憶特性に対する 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を作成します。

ただし、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 を指定しない限り、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、10-56 ページの「CREATE TABLESPACE」を参照してください。

LOGGING | NOLOGGING

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、次に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・ロードの INSERT 操作のログを取る（LOGGING）か取らない（NOLOGGING）かも指定します。

表または表パーティションに対して、この句を省略した場合、表が存在する表領域のロギング属性が、その表または表パーティションのデフォルトのロギング属性として使用されます。

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、行データとともにインラインに格納された LOB に適用されません。つまり、LOB に対する NOLOGGING を 400 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。

非パーティション表の場合、LOGGING 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、ロギング属性に指定する値は、PARTITION 記述に LOGGING|NOLOGGING を指定しない限り、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。

NOLOGGING モードでは、データの変更時に、（新しいエクステントに INVALID のマークを設定し、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 データは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表の場合は、NOLOGGING 操作の後にバックアップを取る必要があります。

NOLOGGING モードでの操作用に生成される REDO ログのサイズは、LOGGING 属性の設定時に生成されるログより非常に小さくなります。

データベースを ARCHIVELOG モードで実行する場合、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、表がリストアされます。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、表はリストアされません。

参照 : ロギングおよびパラレル DML の詳細は、『Oracle8*i* 概要』および『Oracle8*i* 管理者ガイド』を参照してください。

RECOVERABLE | UNRECOVERABLE

これらのキーワードは以前のバージョンのもので、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、下位互換用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。

制限事項 :

- 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 句によって、表のデータ行が格納される順序を指定できます。

HEAP

HEAP は、構成する *table* のデータ行の格納順序を特定しないことを示します。これはデフォルト値です。

INDEX

INDEX は、*table* を索引構成表として作成することを示します。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。

index_organized_table_clause

index_organized_table_clause を使用して、表の行（主キー列値および非キー列値）が、主キーに基づいて構築された B* ツリー索引によって管理されるように指定します。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、次のいずれかの表です。

- CREATE INDEX 文を使用して主キーに基づいて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化表。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。

制限事項 :

- ROWID 型の列は、索引構成表に指定できません。
- パーティション化された索引構成表には LOB または VARRAY 型の列を含めることはできません（この制限事項はパーティション化されていない索引構成表には適用されません）。

注意 : 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。

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。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。

制限事項 :

- 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。
- 索引構成表の個別パーティションに対して PCTTHRESHOLD は指定できません。

参照 : *index_organized_table_clause* の INCLUDING 句を参照してください。

compression_clause

compression_clause によって、キー圧縮を使用可能または使用禁止にできます。

COMPRESS

COMPRESS を指定して、キー圧縮を使用可能にします。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。

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。

NOCOMPRESS

NOCOMPRESS を指定して、索引構成表でのキー圧縮を使用禁止にします。これはデフォルト値です。

index_organized_overflow_clause

index_organized_overflow_clause によって、指定されたしきい値を超える索引構成表のデータ列を、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、各列の最大サイズを評価し、最大限の列を概算します。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り、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防げます。
- 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され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。

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非主キー列でもかまいません。*column_name* に続くすべての非主キー列は、オーバーフロー・データ・セグメントに保存されます。

制限事項 : 索引構成表の個別パーティションにこの句は指定できません。

注意 : *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 の値（指定した値またはデフォルト）を超えると、Oracle は PCTTHRESHOLD の値に基づいて、行を切り離します。

CLUSTER

CLUSTER 句は、表がクラスタの一部であることを指定します。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。

参照 : 9-3 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。

クラスタ化表はクラスタの領域割当てを使用します。このため、PCTFREE、PCTUSED、INITRANS または MAXTRANS 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。

制限事項 : オブジェクト表および LOB 列を含む表はクラスタの一部にはできません。

LOB_storage_clause

LOB_storage_clause によって、LOB データ・セグメントの記憶属性を指定できます。

- 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、LOB データ・セグメントの表の記憶属性を指定します。

- 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 LOB データに対するデフォルト記憶属性を指定します。この記憶属性は、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* によってオーバーライドされない限り、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（*partition_definition* 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・セグメントの記憶特性、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特性を指定します。パーティション・レベルの *LOB_storage_clause* は、表レベルの *LOB_storage_clause* をオーバーライドします。
- パーティション表のそれぞれのサブパーティションの場合（*subpartition_clause* の一部として指定した場合）、この句は、このパーティションのデータ・セグメントの記憶属性を指定します。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* をオーバーライドします。

制限事項 : 表がパーティション化されている場合は、*LOB_index_clause* を指定できません。

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など、LOB の詳細は、『Oracle8*i* アプリケーション開発者ガイド ラージ・オブジェクト』を参照してください。
- 10-51 ページの「LOB 列の例」を参照してください。

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 オブジェクトの属性を指定します。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。

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_parameters
: *LOB_parameters* 句によって、様々な LOB 記憶域の要素を指定できます。

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 値をインラインに格納します。これはデフォルト値です。

制限事項 : *index_organized_table_clause* に OVERFLOW セグメントを指定しない限り、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は、LOB 値の長さにかかわらず、LOB 値をアウトラインに格納します。

LOB 値が格納されている場所にかかわらず、LOB ロケータは常にアウトラインに格納されます。STORAGE IN ROW の値は、一度設定すると、表を移動しない限り変更できません。

参照 : 8-25 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 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、一度設定すると変更できません。

注意 : CHUNK 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。

PCTVERSION *integer*	LOB の記憶領域全体のうち、新規バージョンの LOB の作成に使用される最大記憶領域の割合（パーセント）を指定します。デフォルト値は 10 です。これは、LOB の記憶領域全体の 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。
LOB_index_clause	この句は、Oracle8*i* 以降は使用されません。Oracle は各 LOB 列に対して索引を生成します。LOB 索引は、内部的に名前を付けられ、管理されます。 この句を指定することはできますが、指定しないようにしてください。どんな場合でも、LOB 索引を LOB データと異なる表領域に置かないでください。 **参照 :** 以前のバージョンから移行された表の LOB 索引の管理方法については、『Oracle8*i* 移行ガイド』を参照してください。

varray_storage_clause

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する場合は、*varray_storage_clause* を使用します。また、この句を指定した場合、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。

- 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・セグメントの表の記憶特性を指定します。
- 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特性を指定します。
- パーティション表の各パーティションの場合（*partition_definition* 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・セグメントの記憶特性、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特性を指定します。パーティション・レベルの *varray_storage_clause* は、表レベルの *varray_storage_clause* をオーバーライドします。

- パーティション表の各サブパーティションの場合（*subpartition_clause* の一部として指定した場合）、この句は、このパーティションの VARRAY のデータ・セグメントの記憶特性を指定します。サブパーティション・レベルの *varray_storage_clause* は、パーティション・レベルおよび表レベルの *varray_storage_clauses* をオーバーライドします。

制限事項 : *LOB_parameters* の TABLESPACE パラメータは、この句の一部として指定できません。VARRAY に対する LOB 表領域は、デフォルトではそれを格納する表の表領域になります。

nested_table_storage_clause

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する場合は、*nested_table_storage_clause* を使用します。この記憶表は、（デフォルトの記憶特性によって）親表と同じ表領域内に作成されます。この記憶表には、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（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）。

制限事項 :

- 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。
- （*segment_attributes_clause* の一部として）TABLESPACE をネストした表に指定することはできません。表領域は常に親表の表領域です。
- 作成時、*table_ref_constraint*、*column_ref_constraint* またはネストした表の属性に対する参照制約を（*object_properties* の一部として）指定することはできません。ただし、ネストした表を修正して ALTER TABLE を使用する制約を追加できます。
- 記憶表に対して問合せまたは DML 文を直接実行できませんが、ALTER TABLE 文に記憶表の名前を指定することで、ネストした表の列の記憶特性を変更できます。

> **参照 :** ネストした表の列の記憶特性の変更方法については、8-2 ページの「ALTER TABLE」を参照してください。

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。

storage_table *nested_item* の列を含む表の名前を指定します。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。

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。

storage_table で DML 文を直接問い合せたり、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。

参照：ネストした表の列の記憶特性の変更方法については、8-2 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、コレクション・ロケータをネストした表のコピーに戻します。

  注意：ロケータのセッションには有効範囲が決められており、分散したセッションに使用できません。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。

table_properties

range_partitioning_clause

range_partitioning_clause を使用して、*column_list* の範囲の値で表をパーティション化します。索引構成表に対して、*column_list* は表の主キー列のサブセットである必要があります。

column_list 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**）。

制限事項：*column_list* の列には、ROWID、LONG または LOB 以外の組込みデータ型を指定できます。

hash_partitioning_clause

hash_partitioning_clause を使用して、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します。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関数を使用して、列をパーティションに割り当てます。

> **参照：** ハッシュ・パーティションについては、『Oracle8*i* 概要』を参照してください。

column_list 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**）。

制限事項 :

- *column_list* に指定できるのは 16 列以下です。
- *column_list* には、ROWID または UROWID 疑似列を含めることはできません。
- *column_list* の列には、ROWID、LONG または LOB 以外の組込みデータ型を指定できます。

> **注意：** 別のキャラクタ・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・セットのサポートについては、『Oracle8*i* NLS ガイド』を参照してください。

次のいずれかの方法で、ハッシュ・パーティション化できます。

- パーティション数を指定します。この場合、SYS_P*nnn* 形式のパーティション名を割り当てます。STORE IN 句は、ハッシュ・パーティションが格納されている 1 つ以上の表領域を指定します。表領域数はパーティション数と同じである必要はありません。パーティション数が表領域数より多い場合、表領域名を循環させます。
- 名前によってそれぞれのパーティションを指定します。TABLESPACE 句は、パーティションが格納される場所を指定します。

注意： ハッシュ・パーティション（またはサブパーティション）に指定できる唯一の属性は TABLESPACE です。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。ハッシュ・パーティションは、パーティション・レベルで指定された属性と表レベルのデフォルトからのその他すべての属性を継承します。

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オーバーライドされ、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オーバーライドされます。

---

composite_partitioning_clause

composite_partitioning_clause を使用して、表がまずレンジ・パーティション化され、次にそれらのパーティションがハッシュ・サブパーティション化します。レンジ・パーティションとハッシュ・サブ・パーティションの組合せを、**コンポジット・パーティション**といいます。

subpartition_clause
: *subpartition_clause* を使用して、表の各パーティションをハッシュすることによって、サブパーティション化します。*column_list* 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。

SUBPARTITIONS *quantity*
: 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と、格納されている 1 つ以上の表領域を任意で指定します。

  デフォルト値は 1 です。ここで *subpartition_clause* を指定しないと、後で *partition_level_hash_subpartitioning* 句を指定しない限り、Oracle は、各パーティションを 1 つのハッシュ・パーティションで作成します。

partition_definition

PARTITION *partition*
: 物理パーティション属性を指定します。*partition* を省略した場合、名前はパーティションに対して SYS_P*n* の形式で生成されます。*partition* は、スキーマ・オブジェクトのネーミング規則に従っている必要があります。また、各部分は、2-81 ページの「スキーマ・オブジェクトのネーミング規則」の説明に従って指定する必要があります。

注意 :

- 64KB - 1 以内のパーティションと 64KB - 1 以内のサブパーティションを指定できます。この数字以下の制限を適用する関数については、『Oracle8*i* 管理者ガイド』を参照してください。
- 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。ただし、パーティションが 1 つのみのパーティション表と、非パーティション表は違うことに注意してください。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。

VALUES LESS THAN *value_list*

現在のパーティションの上限（境界は含まない）を指定します。*value_list* は、*partition_by_range_clause* の *column_list* に対応するリテラル値の順序リストです。*value_list* のリテラルには、キーワード MAXVALUE を指定できます。MAXVALUE は、常に他の値より高位にソートされる最大値（NULL を含む）を指定します。

パーティション・バウンドの上限に 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。

参照 : パーティション・バウンドの詳細は、『Oracle8*i* 概要』を参照してください。

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 NLS 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 YYYY4 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。（RRRR 書式マスクは、サポートしていません。）NLS 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。

参照 :

- これらの初期化パラメータの詳細は、『Oracle8*i* NLS ガイド』を参照してください。
- 10-51 ページの「パーティション表の例」を参照してください。

LOB_storage_clause

LOB_storage_clause によって、このパーティションの 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、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。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_P*n* および SYS_IL_P*n* の形式を取ります。ここで、P はパーティション、*n* はシステム生成番号です。

varray_storage_clause

varray_storage_clause によって、このパーティションの 1 つ以上の VARRAY 項目に対して記憶特性が指定できます。

partition_level_subpartitioning

partition_level_subpartitioning 句によって、パーティションのハッシュ・サブパーティションを指定します。この句は、*subpartition_clause* でのデフォルト設定をオーバーライドします。

制限事項 : コンポジット・パーティション表に対してのみこの句を指定できます。

- 個々のパーティションを名前で指定できます。また、各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。
- サブパーティションの数を指定できます（または、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_SUBP*nnn* という形式のサブパーティション名を割り当てます。表領域の数は、サブパーティションの数と等しくなくてもかまいません。パーティション数が表領域数より多い場合、表領域名は繰り返されます。

row_movement_clause

row_movement_clause によって、更新操作中に 1 つ以上のキー値の変更によって、行が別のパーティションまたはサブパーティションに移動できるかどうかを指定します。

制限事項 : パーティション表に対してのみこの句を指定できます。

ENABLE　ENABLE を指定すると、パーティション化またはサブパーティション化キーを更新した結果、異なるパーティションやサブパーティションに行が移動されます。

注意 : UPDATE 操作中に行を移動すると、その行の ROWID が変更されます。

DISABLE　DISABLE を指定すると、パーティション化またはサブパーティション化キーを更新した結果、異なるパーティションまたはサブパーティションに行が移動され、エラーが戻ります。これはデフォルト値です。

CACHE | NOCACHE | CACHE READS

CACHE　アクセス頻度の高いデータについて、CACHE は、フル・テーブル・スキャンの実行時にこの表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置くことを指定します。この句は、小規模な参照表で有効です。

LOB_storage_clause のパラメータとして CACHE を使用する場合は、バッファ・キャッシュに LOB 値を配置することを指定します。

制限事項 : 索引構成表に CACHE を指定することはできません。ただし、索引構成表は暗黙的に CACHE 動作を指定します。

NOCACHE　アクセス頻度の低いデータについて、NOCACHE は、フル・テーブル・スキャンの実行時にこの表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 リストの最低使用頻度側に置くことを指定します。これはデフォルト値です。

LOB_storage_clause のパラメータに NOCACHE を使用する場合は、LOB 値をバッファ・キャッシュには入れない場合と、バッファ・キャッシュに入れて LRU リストの最低使用頻度側に置く場合のどちらかを指定します（デフォルトでは後者です）。

制限事項 : 索引構成表に NOCACHE を指定することはできません。

注意 : NOCACHE は、*storage_clause* に KEEP を指定した表には、影響しません。

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込み操作中にバッファ・キャッシュに入れられることを指定します。

MONITORING | NOMONITORING

MONITORING　表の変更統計を収集する場合は、MONITORING を指定します。これらの統計表は、一定の期間に DML 文の影響を受ける行数の推定値です。これらは、オプティマイザが使用またはユーザーが分析する場合に使用できます。

制限事項 : 一時表に MONITORING を指定することはできません。

NOMONITORING　表の変更統計を収集しない場合は、NOMONITORING を指定します。これはデフォルト値です。

制限事項 : 一時表に NOMONITORING を指定することはできません。

parallel_clause

parallel_clause によって、表のパラレル作成および作成後に問合せおよび DML のパラレル化のデフォルト値の設定ができます。

注意 : *parallel_clause* 構文は、以前のリリースの構文に代わるものです。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　シリアル実行を行う場合は、NOPARALLEL を指定します。これはデフォルト値です。

PARALLEL　すべてのパーティション化インスタンスで使用可能な CPU の数に、PARALLEL_THREADS_PER_CPU 初期化パラメータの値を掛けた並列度を選択させる場合は、PARALLEL を指定します。

PARALLEL *integer*　パラレル操作で使用されるパラレル・スレッド数である**並列度**を指定する場合は、*integer* 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 に関する注意事項

- 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る場合、表に対する後続の INSERT 操作、UPDATE 操作、DELETE 操作およびこの文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。
- 索引構成表のパーティションでは、CREATE TABLE ... AS SELECT はシリアルで実行され、DML 操作の後に行われます。ただし、後続の問合せはパラレルで実行されます。
- *parallel_clause* の結果はパラレル・ヒントによってオーバーライドされます。
- DML 文および CREATE TABLE ... AS SELECT 文はパラレルで実行されるリモート・オブジェクトを参照します。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）。

> **参照：** パラレル化操作の詳細は、『Oracle8*i* パフォーマンスのための設計およびチューニング』、『Oracle8*i* 概要』および『Oracle8*i* Parallel Server 概要』を参照してください。

enable_disable_clause

enable_disable_clause によって、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 文に作成されます。

制限事項：

- 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または主キー制約がすでに使用可能でない場合は、参照整合性制約を使用可能にできません。

> **参照：** 制約の詳細は、8-134 ページの「constraint_clause」を参照してください。

ENABLE

制約を表のすべての新しいデータに適用させる場合は、ENABLE を指定します。

- VALIDATE は、すべての旧データも制約に従うことを指定します。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、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 マークを付けます（最適な結果を得るためには、列にデータを入力してから行ってください）。

- NOVALIDATE は、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。この句は、表の既存データが制約に従っていることを保証しないため、表レベルロックは必要ありません。

- VALIDATE も NOVALIDATE も指定しない場合、VALIDATE がデフォルト値になります。

- 一意キー制約または主キー制約を使用可能にした場合で、キーに索引が存在しない場合、Oracle は一意の索引を作成します。その後で制約が使用禁止になった場合、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ために、最初に使用禁止にした主キー制約および一意キー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 NOVALIDATE から ENABLE VALIDATE までの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、読取り、書込みまたはその他の DDL 操作が中断されません。

  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

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。この機能は大変有効です。データ・ウェアハウスでは、一意キーに重複しない範囲の値を持つ一定量のデータをレンジ・パーティション表にロードする必要があります。このような場合、制約が使用禁止で、妥当性チェック済の場合は、索引を持たないことによって領域を節約できます。ALTER TABLE 文の *exchange_partition_clause* 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。

  一意キーがパーティション表のパーティション・キーと一致する場合、制約を使用禁止にすることによって、オーバーヘッドを減らすことができ、有害な影響もなくなります。一意キーがパーティション・キーと一致しない場合は、制約の妥当性チェックを行うための変換中に自動テーブル・スキャンが実行され、これによって索引なしでロードすることのメリットがオフセットされてしまいます。

- 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。

  参照：この設定の使用については、『Oracle8*i* パフォーマンスのための設計およびチューニング』を参照してください。

  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。

- VALIDATE も NOVALIDATE も指定しない場合、NOVALIDATE がデフォルト値になります。
- 一意索引を使用している一意キー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。

UNIQUE　UNIQUE 句によって、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 句によって、表の主キー制約を使用可能または使用禁止にできます。

CONSTRAINT　CONSTRAINT 句によって、*constraint* という整合性制約を使用可能または使用禁止にできます。

using_index_clause

using_index_clause によって、Oracle が作成する索引に対してパラメータを指定し、一意キー制約または主キー制約を適用します。索引には制約と同じ名前が割り当てられます。

索引には、INITRANS、MAXTRANS、TABLESPACE、STORAGE および PCTFREE の各パラメータ値を選択できます。これらのパラメータは、この文ですでに説明されています。表がパーティション化されている場合、一意キー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。

> **参照：** LOCAL および *global_index_clause* の詳細、索引に関する NOSORT および LOGGING|NOLOGGING の詳細は、9-51 ページの「CREATE INDEX」を参照してください。

制限事項：一意キー制約および主キー制約を使用可能にする場合のみ、これらのパラメータを使用します。

EXCEPTIONS INTO

制約に違反するすべての列の ROWID を格納する表を指定します。スキーマを指定しない場合、自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。例外表は、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、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。

参照： これらのスクリプトの使用に関する互換性については、『Oracle8*i* 移行ガイド』を参照してください。

注意： 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々の例外表を作成し、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。

参照：

- SQL スクリプトの詳細は、『Oracle8*i* PL/SQL パッケージ・プロシージャ 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、『Oracle8*i* パフォーマンスのための設計およびチューニング』を参照してください。

CASCADE

CASCADE を指定して、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します。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。

制限事項： DISABLE を指定した場合のみ、CASCADE を指定できます。

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。

> **参照：** 11-88 ページの「SELECT および副問合せ」を参照してください。

制限事項：

- 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、参照整合性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。

この文で *parallel_clause* を指定した場合、INITIAL 記憶領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、11-129 ページの「storage_clause」を参照してください。

データ型およびデータ長は、副問合せの結果から導出されます。Oracle では、整合性制約については次の規則も適用されます。

- 副問合せで列を含む式ではなく、列を選択した場合、選択されている表の列に NOT NULL 制約があると、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。
- CREATE TABLE 文に AS *subquery* と、CONSTRAINT 句または EXCEPTIONS INTO を付けた ENABLE 句の両方を指定した場合、AS *subquery* は無視されます。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。

subquery 中のすべての式が、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 関数と組み合せて *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 を LOB にコピーする理由および時期については、『Oracle8*i* 移行ガイド』を参照してください。
- TO_LOB 関数の使用方法については、4-5 ページの「変換関数」を参照してください。

注意 : *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、Oracle は、*subquery* で指定した 1 つ以上の表のかわりに（問合せのリライト用の）マテリアライズド・ビューを使用することがあります。

参照 : マテリアライズド・ビューおよび問合せのリライトの詳細は、『Oracle8*i* データ・ウェアハウス』を参照してください。

---

order_by_clause

ORDER BY 句は、文によって戻される行を順序付けます。

参照 : *order_by_clause* の詳細は、11-88 ページの「SELECT および副問合せ」を参照してください。

注意 : 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限りません（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 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。

例

一般的な例　scott が所有している emp 表を定義する場合、次の文を発行します。

```
CREATE TABLE scott.emp
   (empno     NUMBER        CONSTRAINT pk_emp PRIMARY KEY,
    ename     VARCHAR2(10)  CONSTRAINT nn_ename NOT NULL
                            CONSTRAINT upper_ename
   CHECK (ename = UPPER(ename)),
       job        VARCHAR2(9),
       mgr        NUMBER       CONSTRAINT fk_mgr
                               REFERENCES scott.emp(empno),
       hiredate  DATE          DEFAULT SYSDATE,
       sal       NUMBER(10,2)  CONSTRAINT ck_sal
   CHECK (sal > 500),
          comm      NUMBER(9,0)   DEFAULT NULL,
          deptno    NUMBER(2)     CONSTRAINT nn_deptno NOT NULL
                                  CONSTRAINT fk_deptno
                                  REFERENCES scott.dept(deptno) )
   PCTFREE 5 PCTUSED 75;
```

この表は 8 列で構成されます。empno 列は、データ型が NUMBER、関連付けられている整合性制約の名前が pk_emp です。hiredate 列は、データ型が DATE、デフォルト値が SYSDATE です。

この表定義では、PCTFREE を 5、PCTUSED を 75 に指定しています。この定義は、比較的変更の少ない表に適しています。また、この定義では、emp 表の列のいくつかに整合性制約も定義します。

一時表の例　次の文は、航空券自動予約スケジュール・システムで使用する一時表 flight_schedule を作成します。各クライアントは、それぞれのセッションを持ち、一時的なスケジュールを格納できます。一時的なスケジュールは、セッションの終わりに削除されます。

```
CREATE GLOBAL TEMPORARY TABLE flight_schedule (
   startdate DATE,
   enddate DATE,
   cost NUMBER)
   ON COMMIT PRESERVE ROWS;
```

記憶域の例　記憶域の容量が小さく、割当てに制限のある human_resource 表領域の中にサンプル表 salgrade を定義する場合は、次の文を発行します。

```
CREATE TABLE salgrade
   ( grade  NUMBER  CONSTRAINT pk_salgrade
                    PRIMARY KEY
                    USING INDEX TABLESPACE users_a,
     losal  NUMBER,
     hisal  NUMBER )
   TABLESPACE human_resource
   STORAGE (INITIAL     6144
            NEXT        6144
            MINEXTENTS     1
            MAXEXTENTS     5 );
```

この文では grade 列に主キー制約を定義し、この制約を適用するために自動的に作成される索引を、users_a 表領域に作成するように指定しています。

> **参照：** 整合性制約の定義の例は、8-134 ページの「constraint_clause」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、scott.emp をスキャンし、emp_dept を移入します。

```
CREATE TABLE emp_dept
   PARALLEL
   AS SELECT * FROM scott.emp
   WHERE deptno = 10;
```

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、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 emp_dept
   AS SELECT * FROM scott.emp
   WHERE deptno = 10;
```

ENABLE VALIDATE の例　次の文は、dept 表を作成し、主キー制約を定義して、その制約を ENABLE VALIDATE 状態にします。

```
CREATE TABLE dept
   (deptno  NUMBER (2) PRIMARY KEY,
    dname   VARCHAR2(10),
    loc     VARCHAR2(9) )
   TABLESPACE user_a;
```

DISABLE の例　次の文は、dept 表を作成し、使用禁止の主キー制約を定義します。

```
CREATE TABLE dept
   (deptno   NUMBER (2)   PRIMARY KEY DISABLE,
    dname    VARCHAR2(10),
    loc      VARCHAR2(9) );
```

EXCEPTIONS INTO の例　次の文は、整合性制約 check_orders に違反する索引構成表 orders の行を格納するための order_exceptions 表を作成します。

```
CREATE TABLE orders
   (ord_num  NUMBER PRIMARY KEY,
    ord_quantity NUMBER)
   ORGANIZATION INDEX;

EXECUTE DBMS_IOT.BUILD_EXCEPTIONS_TABLE
   ('SCOTT', 'ORDERS', 'ORDER_EXCEPTIONS');

ALTER TABLE orders
   ADD CONSTRAINT CHECK_ORDERS CHECK (ord_quantity > 0)
   EXCEPTIONS INTO ORDER_EXCEPTIONS;
```

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：

- 11-52 ページの「INSERT」を参照してください。
- 表に行を挿入するために必要な権限の詳細は、11-88 ページの「SELECT および副問合せ」を参照してください。

ネストした表の例　次の文は、ネストした表の列 projects を持つリレーショナル表 employee を作成します。

```
CREATE TABLE employee
   (empno NUMBER, name CHAR(31), projects PROJ_TABLE_TYPE)
   NESTED TABLE projects STORE AS nested_proj_table(
      (PRIMARY KEY (nested_table_id, pno)) ORGANIZATION INDEX)
   RETURN AS LOCATOR;
```

LOB 列の例　次の文は、2 つの LOB 列がある表 lob_tab を作成し、LOB 記憶特性を指定します。

```
CREATE TABLE lob_tab (col1 BLOB, col2 CLOB)
   STORAGE (INITIAL 256 NEXT 256)
   LOB (col1, col2) STORE AS
      (TABLESPACE lob_seg_ts
       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、索引構成表を作成します。

```
CREATE TABLE docindex
  ( token                  CHAR(20),
    doc_oid                INTEGER,
    token_frequency        SMALLINT,
    token_occurrence_data  VARCHAR2(512),
       CONSTRAINT pk_docindex PRIMARY KEY (token, doc_oid) )
  ORGANIZATION INDEX TABLESPACE text_collection
  PCTTHRESHOLD 20 INCLUDING token_frequency
  OVERFLOW TABLESPACE text_collection_overflow;
```

パーティション表の例　次の文は、3 つのパーティションを持つ表を作成します。

```
CREATE TABLE stock_xactions
      (stock_symbol CHAR(5),
       stock_series CHAR(1),
       num_shares NUMBER(10),
       price NUMBER(5,2),
       trade_date DATE)
   STORAGE (INITIAL 100K NEXT 50K) LOGGING
   PARTITION BY RANGE (trade_date)
     (PARTITION sx1992 VALUES LESS THAN (TO_DATE('01-JAN-1993','DD-MON-YYYY'))
       TABLESPACE ts0 NOLOGGING,
      PARTITION sx1993 VALUES LESS THAN (TO_DATE('01-JAN-1994','DD-MON-YYYY'))
       TABLESPACE ts1,
      PARTITION sx1994 VALUES LESS THAN (TO_DATE('01-JAN-1995','DD-MON-YYYY'))
       TABLESPACE ts2);
```

参照：　パーティション表のメンテナンス操作については、『Oracle8*i* 管理者ガイド』を参照してください。

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 3 つの LOB 列 b、c、d を持つパーティション表 pt を作成します。

```
CREATE TABLE PT (A NUMBER, B BLOB, C CLOB, D CLOB)
   LOB (B,C,D) STORE AS (STORAGE (NEXT 20M))
   PARTITION BY RANGE (A)
   (PARTITION P1 VALUES LESS THAN (10) TABLESPACE TS1
      LOB (B,D) STORE AS (TABLESPACE TSA STORAGE (INITIAL 20M)),
   PARTITION P2 VALUES LESS THAN (20)
      LOB (B,C) STORE AS (TABLESPACE TSB)
   TABLESPACE TSX;
```

パーティション p1 は、表領域 ts1 にあります。b および d に対する LOB データ・パーティションは、表領域 tsa にあります。c に対する LOB データ・パーティションは、表領域 ts1 にあります。記憶域属性 INITIAL は、LOB 列 b および d に指定され、その他の属性はデフォルトの表レベルの指定から継承します。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、b および d 列に対しては表領域 tsa から、c 列に対しては表領域 ts1 から継承されます。LOB 索引のパーティションは、対応する LOB データ・パーティションと同じ表領域にあります。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 tsx にあります。b および c に対する LOB データは、表領域 tsb にあります。d に対する LOB データは、表領域 tsx にあります。b 列および c 列に対する LOB 索引は、表領域 tsb にあります。d 列に対する LOB 索引は、表領域 tsx にあります。

ハッシュ・パーティション表の例　次の文は、化学に関するデータを含む列のハッシュによってパーティション化された表を作成します。ハッシュ・パーティションは、表領域 tbs1、tbs2、tbs3 および tbs4 に格納されます。

```
CREATE TABLE exp_data (
      d DATE, temperature NUMBER, Fe2O3_concentration NUMBER,
      HCl_concentration NUMBER, Au_concentration NUMBER,
      amps NUMBER, observation VARCHAR(4000))
   PARTITION BY HASH (HCl_concentration, Au_concentration)
   PARTITIONS 32 STORE IN (tbs1, tbs2, tbs3, tbs4);
```

コンポジット・パーティション表の例　次の文は、コンポジット・パーティション表を作成します。レンジ・パーティション化によって、販売日ごとのデータおよびパーティションのプルーニングが容易になります。ハッシュ・サブパーティションは、特定の項目番号で、問合せに対するサブパーティションの排除を可能にします。パーティションのほとんどは、8 つのサブパーティションで構成されています。ただし、閑散四半期を処理するパーティションは、4 つのサブパーティションで構成され、繁忙四半期を処理するパーティションは、16 のサブパーティションで構成されています。

```
CREATE TABLE sales (item INTEGER, qty INTEGER,
                    store VARCHAR(30),
                    dept NUMBER, sale_date DATE)
   PARTITION BY RANGE (sale_date)
   SUBPARTITION BY HASH(item)
   SUBPARTITIONS 8
   STORE IN (tbs1, tbs2, tbs3, tbs4, tbs5, tbs6, tbs7, tbs8)
   (PARTITION q1_1997
      VALUES LESS THAN (TO_DATE('01-apr-1997', 'dd-mon-yyyy')),
    PARTITION q2_1997
      VALUES LESS THAN (TO_DATE('01-jul-1997', 'dd-mon-yyyy')),
    PARTITION q3_1997
      VALUES LESS THAN (TO_DATE('01-oct-1997', 'dd-mon-yyyy'))
      (SUBPARTITION q3_1997_s1 TABLESPACE ts1,
       SUBPARTITION q3_1997_s2 TABLESPACE ts3,
       SUBPARTITION q3_1997_s3 TABLESPACE ts5,
       SUBPARTITION q3_1997_s4 TABLESPACE ts7),
    PARTITION q4_1997
      VALUES LESS THAN (TO_DATE('01-jan-1998', 'dd-mon-yyyy'))
      SUBPARTITIONS 16
      STORE IN (tbs1, tbs3, tbs5, tbs7, tbs8, tbs9, tbs10,
                tbs11),
    PARTITION q1_1998
      VALUES LESS THAN (TO_DATE('01-apr-1998', 'dd-mon-yyyy')));
```

オブジェクト表の例　オブジェクト型 dept_t を指定するとします。

```
CREATE TYPE dept_t AS OBJECT
   ( dname VARCHAR2(100),
     address VARCHAR2(200) );
```

オブジェクト表 dept は、dept_t 型の部門オブジェクトを持ちます。

```
CREATE TABLE dept OF dept_t;
```

次の文は、ユーザー定義オブジェクト型 salesrep_t を持つオブジェクト表 salesreps を作成します。

```
CREATE OR REPLACE TYPE salesrep_t AS OBJECT
  ( repId NUMBER,
    repName VARCHAR2(64));
CREATE TABLE salesreps OF salesrep_t;
```

ネストした表の例　次の文は、ネストした表の列 projects を持つリレーショナル表 employee を作成します。

```
CREATE TABLE employee (empno NUMBER, name CHAR(31),
   projects PROJ_TABLE_TYPE)
   NESTED TABLE projects STORE AS nested_proj_table;
```

REF の例　次の文は、オブジェクト型 dept_t およびオブジェクト表 dept を作成し、すべての部署のインスタンスを格納します。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 TYPE dept_t AS OBJECT
   ( dname   VARCHAR2(100),
     address VARCHAR2(200) );

CREATE TABLE dept OF dept_t;

CREATE TABLE emp
   ( ename   VARCHAR2(100),
     enumber NUMBER,
     edept   REF dept_t SCOPE IS dept );
```

次の文は、参照制約が定義されている REF 列を含む表を作成します。

```
CREATE TABLE emp
   ( ename   VARCHAR2(100),
     enumber NUMBER,
     edept   REF dept_t REFERENCES dept);
```

ユーザー定義 OID の例　次の文は、オブジェクト型および OID 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 emp_t AS OBJECT (empno NUMBER, address CHAR(30));
CREATE TABLE emp OF emp_t (empno PRIMARY KEY)
   OBJECT IDENTIFIER IS PRIMARY KEY;
```

この後は、次の 2 つのいずれかの方法で emp オブジェクト表を参照できます。

```
CREATE TABLE dept (dno NUMBER
                   mgr_ref REF emp_t SCOPE IS emp);
CREATE TABLE dept (
    dno NUMBER,
    mgr_ref REF emp_t CONSTRAINT mgr_in_emp REFERENCES emp);
```

タイプ列の制約の例

```
CREATE TYPE address 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,
    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') );
```

PARALLEL の例　次の文は、10 のパラレル実行サーバーを使用して表を作成します。10 のうち 5 つは `scott.emp` のスキャン用で、残りの 5 つは `emp_dept` に値を入れるためのものです。

```
CREATE TABLE emp_dept
   PARALLEL (5)
   AS SELECT * FROM scott.emp
   WHERE deptno = 10;
```

CREATE TABLESPACE

用途

CREATE TABLESPACE 文は、**表領域**を作成する場合に使用します。表領域とは、データベース内に割り当てられた領域で永続スキーマ・オブジェクトを格納します。

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。

CREATE TEMPORARY TABLESPACE 文を使用して、セッションの存続期間中のみスキーマ・オブジェクトを格納する表領域を作成できます。

参照：

- 表領域については、『Oracle8*i* 概要』を参照してください。
- 表領域の変更については、8-67 ページの「ALTER TABLESPACE」を参照してください。
- 表領域の削除については、11-10 ページの「DROP TABLESPACE」を参照してください。
- 10-63 ページの「CREATE TEMPORARY TABLESPACE」を参照してください。

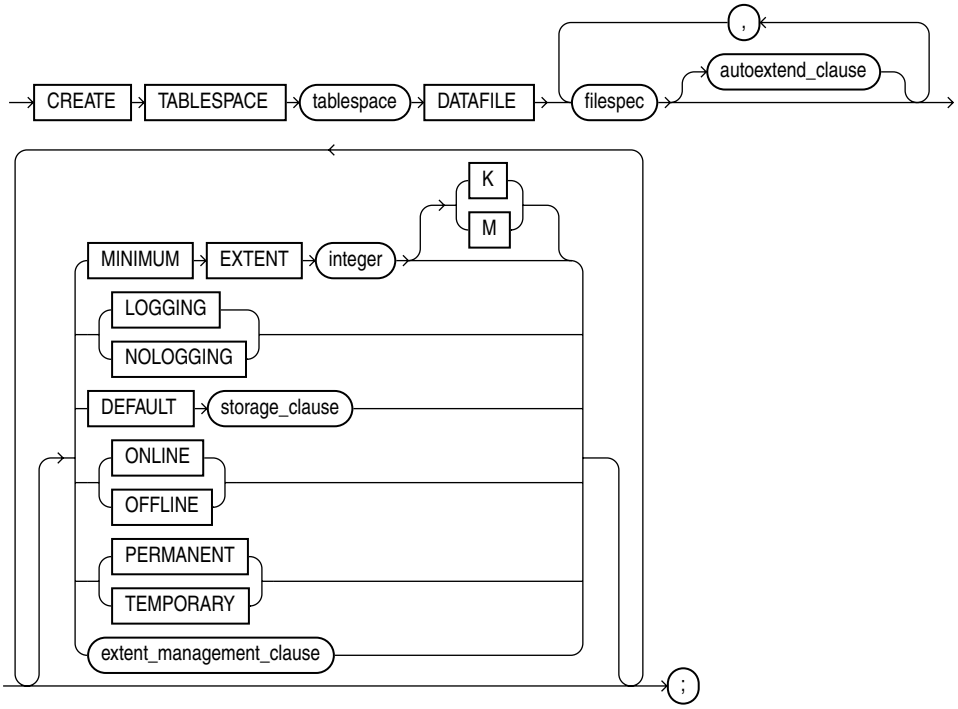
前提条件

CREATE TABLESPACE システム権限が必要です。また、SYSTEM 表領域内に、SYSTEM ロールバック・セグメントを含めた、2 つ以上のロールバック・セグメントが必要です。

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。

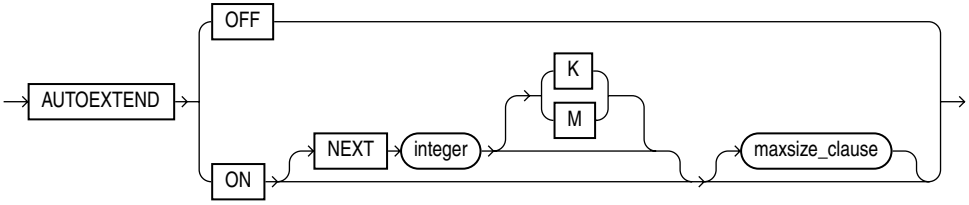
参照： 9-21 ページの「CREATE DATABASE」を参照してください。

構文

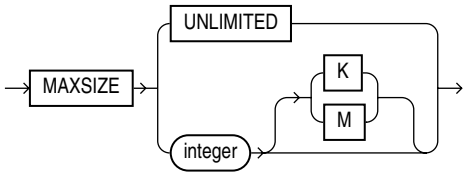


filespec: 11-27 ページの「filespec」を参照してください。

autoextend_clause::=

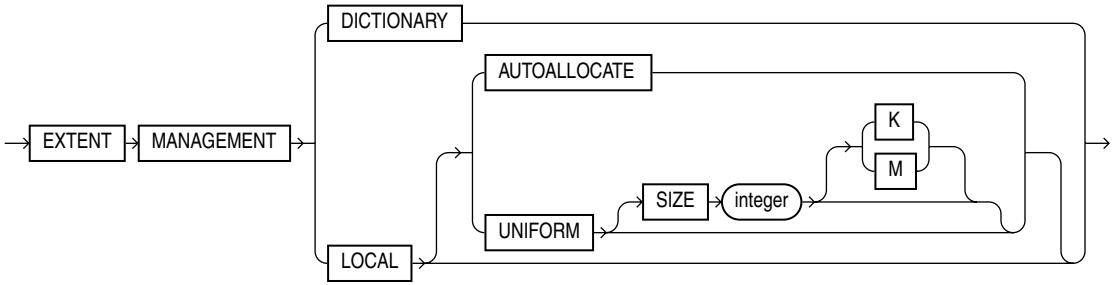


maxsize_clause::=



storage_clause: 11-129 ページの「storage_clause」を参照してください。

extent_management_clause::=



キーワードとパラメータ

tablespace

作成する表領域の名前を指定します。

DATAFILE *filespec*

表領域を構成する 1 つ以上のデータ・ファイルを指定します。

注意： ロー・デバイスをサポートするオペレーティング・システムの場合、*filespec* の REUSE キーワードには、ロー・デバイスをデータ・ファイルとして指定する際の意味はありません。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。

参照： 11-27 ページの「filespec」を参照してください。

autoextend_clause

autoextend_clause を使用して、データ・ファイルの自動拡張機能を使用可能または使用禁止にします。

OFF	OFF を指定すると、自動拡張を使用禁止にします。NEXT および MAXSIZE は 0（ゼロ）に設定されます。NEXT および MAXSIZE の値は、後続の ALTER TABLESPACE AUTOEXTEND 文で再指定する必要があります。
ON	ON を指定すると、自動拡張を使用可能にします。
NEXT *integer*	エクステントがさらに必要な場合に、データ・ファイルに割り当てるディスク領域を指定します。
maxsize_clause	*maxsize_clause* によって、データ・ファイルに割り当てることができるディスク領域の最大サイズを指定します。 ■ *integer*: テンポラリ・ファイルに割り当てることができるディスク領域の最大サイズをバイトで指定します。K または M を使用すると、この領域を KB または MB 単位で指定できます。 ■ UNLIMITED: データ・ファイルへのディスク領域割当てを無制限にする場合は、UNLIMITED を指定します。

MINIMUM EXTENT *integer*

表領域で使用されるエクステント最小サイズを指定します。この句によって、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、*integer* で指定したサイズ以上であり、その倍数になるように指定することによって、表領域における空き領域の断片化を制御します。

注意： この句は、ディクショナリ管理された一時表領域には適用されません。

参照： MINIMUM EXTENT を使用した断片化制御については、『Oracle8*i* 概要』を参照してください。

LOGGING | NOLOGGING

表領域内のすべての表、索引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。デフォルト値は LOGGING です。

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オーバーライドできます。

次の操作でのみ、NOLOGGING モードがサポートされます。

- **DML**: ダイレクト・ロードの INSERT（シリアルまたはパラレル）、ダイレクト・ローダー（SQL*Loader）
- **DDL**: CREATE TABLE ... AS SELECT、CREATE INDEX、ALTER INDEX ... REBUILD、ALTER INDEX ... REBUILD PARTITION、ALTER INDEX ... SPLIT PARTITION、ALTER TABLE ... SPLIT PARTITION および ALTER TABLE ... MOVE PARTITION

NOLOGGING モードでは、データの変更時に、（新しいエクステントに INVALID のマークを設定し、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。

DEFAULT *storage_clause*

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領域パラメータを指定します。ディクショナリ管理された一時表領域の場合、*storage_clause* の NEXT パラメータのみを考慮します。

> **参照：** 記憶領域パラメータについては、11-129 ページの「storage_clause」を参照してください。

ONLINE | OFFLINE

ONLINE	ONLINE を指定して、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します。これはデフォルト値です。
OFFLINE	OFFLINE を指定して、作成直後の表領域を使用禁止にします。 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。

PERMANENT | TEMPORARY

PERMANENT　表領域を永続オブジェクトの格納に使用する場合は、PERMANENT を指定します。これはデフォルト値です。

TEMPORARY　表領域を一時オブジェクトの格納にのみ使用する場合は、TEMPORARY を指定します。たとえば、ORDER BY 句の処理での暗黙的なソートに使用されるセグメントの格納などです。

制限事項 : TEMPORARY を指定する場合、EXTENT MANAGEMENT LOCAL は指定できません。

extent_management_clause

extent_management_clause によって、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場合は、表領域を移行しない限り、エクステントの管理を変更できません。

DICTIONARY　ディクショナリ表を使用して表領域を管理する場合は、DICTIONARY を指定します。これはデフォルト値です。

LOCAL　表領域をローカルで管理する場合は、LOCAL を指定します。ローカルで管理される表領域には、ビットマップ用に取っておいた表領域が一部あります。

- AUTOALLOCATE は、表領域がシステム管理されることを指定します。ユーザーはエクステント・サイズを指定できません。
- UNIFORM は、表領域が SIZE バイトの均一のエクステントで管理されることを指定します。K または M を使用して、KB または MB 単位でエクステント・サイズを指定することもできます。SIZE のデフォルト値は 1MB です。

  参照 : ローカルで管理される表領域については、『Oracle8*i* 概要』を参照してください。

AUTOALLOCATE と UNIFORM のどちらも指定しない場合、AUTOALLOCATE がデフォルト値です。

制限事項 : LOCAL を指定する場合、DEFAULT *storage_clause*、MINIMUM EXTENT または TEMPORARY 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、『Oracle8*i* 移行ガイド』を参照してください。

例

DEFAULT 記憶域の例　次の文は、1 つのデータ・ファイルを持つ tabspace_2 という名前の表領域を作成します。

```
CREATE TABLESPACE tabspace_2
   DATAFILE 'diska:tabspace_file2.dat' SIZE 20M
   DEFAULT STORAGE (INITIAL 10K NEXT 50K
                    MINEXTENTS 1 MAXEXTENTS 999)
   ONLINE;
```

AUTOEXTEND の例　次の文は、1 つのデータ・ファイルを持つ tabspace_3 という名前の表領域を作成します。さらに領域が必要な場合、50KB のエクステントが最大サイズ 10MB まで追加されます。

```
CREATE TABLESPACE tabspace_3
   DATAFILE 'diskb:tabspace_file3.dat' SIZE 500K REUSE
   AUTOEXTEND ON NEXT 500K MAXSIZE 10M;
```

MINIMUM EXTENT の例　次の文は、1 つのデータ・ファイルを持つ tabspace_5 という名前の表領域を作成し、すべてのエクステントを 64KB の倍数として割り当てます。

```
CREATE TABLESPACE tabspace_5
   DATAFILE 'tabspace_file5.dbf' SIZE 2M
   MINIMUM EXTENT 64K
   DEFAULT STORAGE (INITIAL 128K NEXT 128K)
   LOGGING;
```

ローカルで管理される例　次の文は、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_1 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、ローカルで管理される表領域を作成します。

CREATE TEMPORARY TABLESPACE

用途

CREATE TEMPORARY TABLESPACE 文は、**一時表領域**を作成する場合に使用します。一時表領域は、セッションの存続期間中に、スキーマ・オブジェクトを含むことができるデータベース中の領域の割当てです。

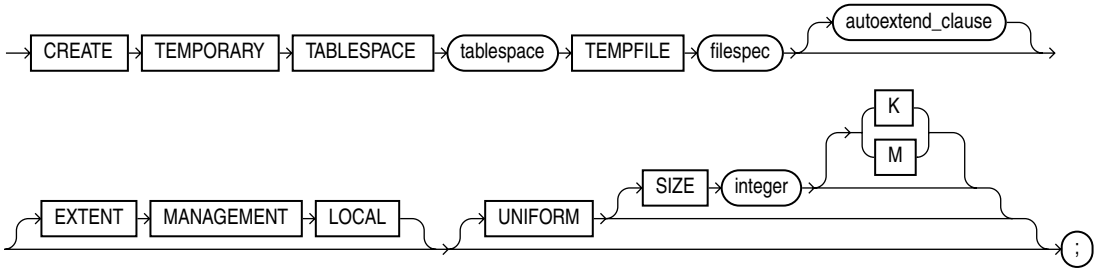
CREATE TABLESPACE 文を使用して、永続スキーマ・オブジェクトを含む表領域を作成します。

参照： 10-56 ページの「CREATE TABLESPACE」を参照してください。

前提条件

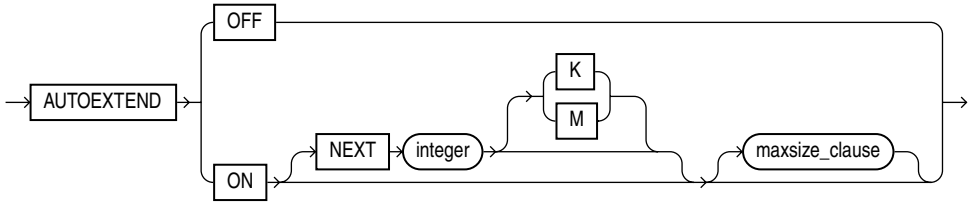
CREATE TABLESPACE システム権限が必要です。

構文

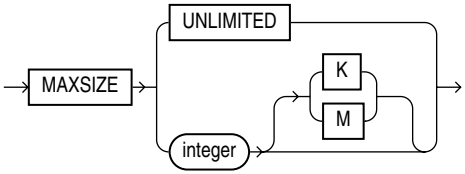


filespec: 11-27 ページの「filespec」を参照してください。

autoextend_clause::=



maxsize_clause::=



キーワードとパラメータ

tablespace

一時表領域の名前を指定します。

TEMPFILE *filespec*

表領域を構成するテンポラリ・ファイルを指定します。

注意： メディア・リカバリはテンポラリ・ファイルを認識しません。

参照： 11-27 ページの「filespec」を参照してください。

autoextend_clause

autoextend_clause によって、テンポラリ・ファイルの自動拡張機能を使用可能または使用禁止にできます。

OFF	OFF を指定すると、自動拡張を使用禁止にします。NEXT および MAXSIZE は 0（ゼロ）に設定されます。NEXT および MAXSIZE の値は、後続の ALTER TABLESPACE AUTOEXTEND 文で再指定する必要があります。
ON	ON を指定すると、自動拡張を使用可能にします。
NEXT *integer*	エクステントがさらに必要な場合に、テンポラリ・ファイルに割り当てるディスク領域を指定します。
maxsize_clause	*maxsize_clause* によって、テンポラリ・ファイルの割当てで使用されるディスク領域の最大サイズを指定します。

- *integer*: テンポラリ・ファイルに割当てできるディスク領域の最大サイズをバイトで指定します。このとき、K または M を使用して、KB または MB 単位で指定することもできます。
- UNLIMITED: テンポラリ・ファイルへのディスク領域割当てを無制限にする場合は、UNLIMITED を指定します。

EXTENT MANAGEMENT LOCAL

EXTENT MANAGEMENT 句によって、表領域がローカルで管理されるように指定できます。つまり、表領域の一部をビットマップ用に確保しておきます。

UNIFORM *integer*
: 一時表領域のエクステント・サイズをバイトで指定します。表領域のエクステントはすべて同じサイズ（均一）です。この句を指定しない場合は、均一のエクステント 1MB を使用します。

SIZE *integer*
: 表領域のエクステントのサイズをバイト単位で指定します。K または M を使用して、KB または MB 単位で指定することもできます。

  SIZE を指定しない場合は、デフォルトのエクステント・サイズ 1MB を使用します。

参照： ローカルで管理される表領域については、『Oracle8*i* 概要』を参照してください。

例

一時表領域の例 次の文は、各エクステントが 16MB の一時表領域を作成します。

```
CREATE TEMPORARY TABLESPACE tbs_1 TEMPFILE 'file_1.f'
   EXTENT MANAGEMENT LOCAL UNIFORM SIZE 16M;
```

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。

CREATE TRIGGER

用途

CREATE TRIGGER 文は、次の**データベース・トリガー**を作成および使用可能にする場合に使用します。

- 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
- 無名 PL/SQL ブロック、または 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。

トリガーを作成した場合、そのトリガーは自動的に使用可能になります。この後、DISABLE 句および ENABLE 句を指定した ALTER TRIGGER 文または ALTER TABLE 文を使用して、トリガーを使用禁止または使用可能にすることができます。

参照：

- トリガーの様々な型については、『Oracle8*i* 概要』を参照してください。
- 前述の用途のためにトリガーを設計する方法については、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。
- トリガーの使用可能化、使用禁止またはコンパイルの詳細は、8-76 ページの「ALTER TRIGGER」および 8-2 ページの「ALTER TABLE」を参照してください。
- トリガーの削除については、11-13 ページの「DROP TRIGGER」を参照してください。

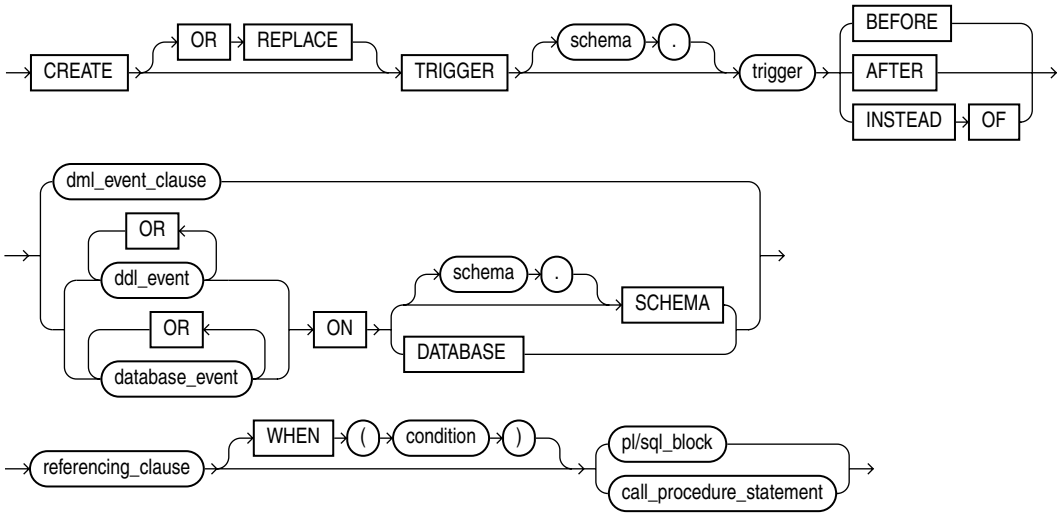
前提条件

トリガーを作成する前に、ユーザー SYS は SQL スクリプト DBMSSTDX.SQL を実行する必要があります。このスクリプトの正確な名前および格納位置は、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。

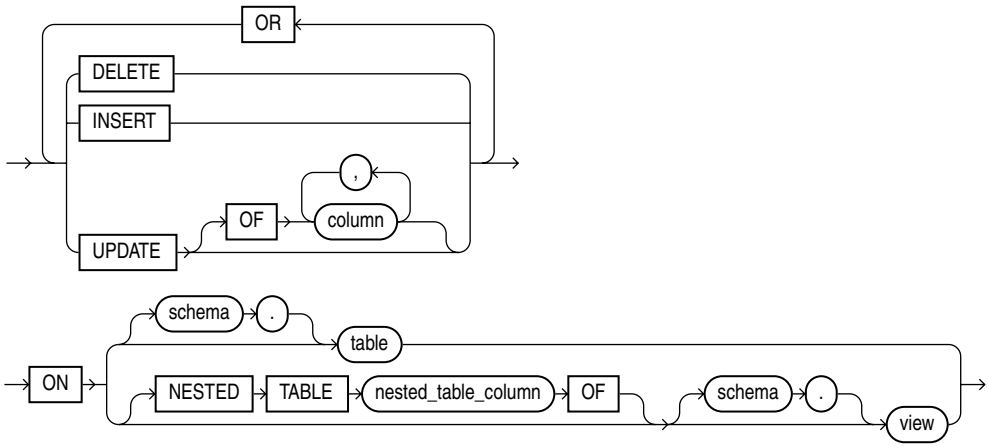
- 自スキーマ内の表または自スキーマ（SCHEMA）に対するトリガーを自スキーマ内に作成する場合は、CREATE TRIGGER 権限が必要です。
- 任意のスキーマ内の表または別のユーザーのスキーマ（*schema*.SCHEMA）に対するトリガーを任意のスキーマ内に作成する場合は、CREATE ANY TRIGGER 権限が必要です。
- 前述の権限の他にも、DATABASE に対するトリガーを作成する場合は、ADMINISTER DATABASE TRIGGER システム権限が必要です。

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。

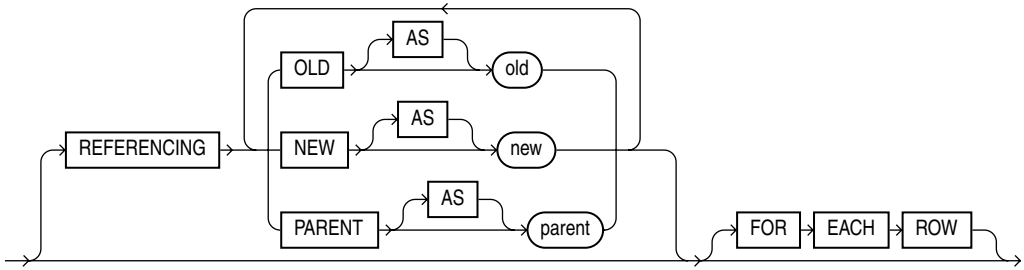
構文



dml_event_clause::=



referencing_clause::=



キーワードとパラメータ

OR REPLACE

既存のトリガーを再作成する場合は、OR REPLACE を指定します。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。

schema

作成するトリガーが含まれるスキーマを指定します。*schema* を指定しない場合、自スキーマにトリガーが作成されます。

trigger

作成するトリガーの名前を指定します。

トリガーのコンパイル・エラーが発生した場合、このトリガーは作成されますが実行時に失敗します。SQL*Plus コマンド SHOW ERRORS を使用して、関連するコンパイラ・エラー・メッセージを表示できます。この場合、コンパイル・エラーになったトリガーが使用禁止にされるか、コンパイル・エラーのないバージョンに置き換えられるか、または削除されるまでは、トリガーを起動するすべての DML 文は実行できません。

注意： マテリアライズド・ビューのベース表にトリガーを作成する場合は、トリガーがマテリアライズド・ビューのリフレッシュ中に起動しないようにする必要があります（リフレッシュ中、DBMS_SNAPSHOT プロシージャ I_AM_A_REFRESH が TRUE を戻します）。

BEFORE

BEFORE を指定すると、トリガーを起動するイベントが実行される前に、トリガーが起動されます。行レベル・トリガーの場合、対象となる各行が変更される前に個別に起動されます。

制限事項 :

- ビューおよびオブジェクト・ビューに対しては BEFORE トリガーを指定できません。
- LOB 列に BEFORE トリガーを定義した場合、:OLD 値の読込みはできますが、:NEW 値の読込みはできません。:OLD 値も :NEW 値も書込みはできません。

AFTER

AFTER を指定すると、トリガーを起動するイベント実行後に、トリガーが起動されます。行レベル・トリガーの場合、対象となる各行が変更された後に個別に起動されます。

制限事項 :

- ビューおよびオブジェクト・ビューに対しては AFTER トリガーを指定できません。
- LOB 列に AFTER トリガーを定義した場合、:OLD 値の読込みはできますが、:NEW 値の読込みはできません。:OLD 値も :NEW 値も書込みはできません。

注意 : 表に対するマテリアライズド・ビュー・ログを作成した場合、その表に AFTER ROW トリガーが暗黙的に作成されます。このトリガーは、INSERT 文、UPDATE 文または DELETE 文で表のデータが変更された場合、マテリアライズド・ビュー・ログに 1 行を挿入します。複数の行レベル・トリガーを起動する順序は制御できません。マテリアライズド・ビューの内容に影響するトリガーは記述しないでください。

参照 : マテリアライズド・ビュー・ログの詳細は、9-104 ページの「CREATE MATERIALIZED VIEW LOG」を参照してください。

INSTEAD OF

INSTEAD OF を指定すると、トリガーを起動するイベントのかわりに、トリガーが起動されます。デフォルトでは、INSTEAD OF トリガーは、各行に対してアクティブになります。

ビューが更新可能であり、そのビューに INSTEAD OF トリガーがある場合、トリガーが優先されます。つまり、ビューで DML を実行するかわりに、Oracle はトリガーを起動します。

制限事項 :

- INSTEAD OF は、ビューにのみ使用できる句です。INSTEAD OF トリガーは表には指定できません。

- あるビューに INSTEAD OF トリガーがある場合、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 トリガーがある必要があります。
- LOB 列に INSTEAD OF トリガーを定義した場合、:OLD 値および :NEW 値は、どちらも読取りはできますが、書込みはできません。

注意： 同一の表に対する同一の文について、同じ型（BEFORE、AFTER または INSTEAD OF）のトリガーを複数作成できます。これらのトリガーが起動される順番は不確定です。アプリケーションが同じ文を対象とした同じ種類のトリガーを 2 つ続けて起動する必要がある場合、これら 2 つのトリガーを 1 つのトリガーに結合し、適切な順序でもとのトリガーのトリガー・アクションを実行するようにしてください。

---

dml_event_clause

dml_event_clause によって、トリガーを起動する 3 つの DML 文のうちの 1 つを指定できます。Oracle は、既存のユーザー・トランザクションでトリガーを起動します。

DELETE　DELETE 文が行を表またはネストした表の要素から削除するときにトリガーを起動する場合には、DELETE を指定します。

INSERT　INSERT 文が表またはネストした表の要素へ行を挿入するときにトリガーを起動する場合には、INSERT を指定します。

UPDATE　OF 句の後に指定した列のいずれかの値を UPDATE 文で変更したときに、トリガーを起動する場合には、UPDATE を指定します。OF 句を省略した場合、UPDATE 文で表またはネストした表の任意の列の値を変更するたびにトリガーが起動されます。

UPDATE トリガーでは、オブジェクト型、VARRAY 型および REF 型の列を OF 句の後に指定した場合、UPDATE 文によってこれらの列のいずれかの値が変更される場合にトリガーが起動されます。ただし、トリガー自体の列の値は変更できません。

注意 : OCI 関数または DBMS_LOB パッケージを使用してオブジェクト列の LOB 値または LOB 属性を更新した場合、更新した列または属性が入っている表に定義されているトリガーは起動されません。

制限事項 :

- OF 句は、INSTEAD OF トリガーの UPDATE とともには指定できません。UPDATE 文でビューの任意の列の値を変更した場合、INSTEAD OF トリガーが起動されます。
- ネストした表または LOB 列は OF 句とともには指定できません。

参照 : ビューに対する挿入、更新または削除を禁止する構造体については、10-105 ページの「CREATE VIEW」の AS *subquery* を参照してください。

ネストした表の列に DML 操作を直接実行した場合、そのネストした表の列が入っている表に定義されているトリガーは起動されません。

ddl_event

トリガーを起動する 1 つ以上の DDL 文を指定します。特に指定がない限り、DATABASE または SCHEMA のイベントのトリガーを作成できます。これらのイベント用の BEFORE トリガーおよび AFTER トリガーを作成できます。Oracle は、既存のユーザー・トランザクションでトリガーを起動します。次の値が有効です。

ALTER
: ALTER を指定すると、ALTER 文でデータ・ディクショナリのデータベース・オブジェクトを変更した場合、トリガーが起動されます。

  制限事項 : トリガーは、ALTER DATABASE 文では起動されません。

ANALYZE
: ANALYZE を指定すると、統計情報が収集または削除された場合、またはデータベース・オブジェクトの構造が検証された場合、トリガーが起動されます。

ASSOCIATE STATISTICS
: ASSOCIATE STATISTICS を指定すると、統計タイプがデータベース・オブジェクトと関連付けられた場合、トリガーが起動されます。

AUDIT
: AUDIT を指定すると、SQL 文のオカレンスまたはスキーマ・オブジェクトに対する操作が追跡された場合、トリガーが起動されます。

COMMENT
: COMMENT を指定すると、データベース・オブジェクトに対するコメントがデータ・ディクショナリに追加された場合、トリガーが起動されます。

CREATE	CREATE を指定すると、CREATE 文でデータ・ディクショナリに新しいデータベース・オブジェクトを追加した場合、トリガーが起動されます。 **制限事項 :** トリガーは、CREATE DATABASE 文または CREATE CONTROLFILE 文では起動されません。
DISASSOCIATE STATISTICS	DISASSOCIATE STATISTICS を指定すると、統計タイプがデータベース・オブジェクトとの関連付けを解除した場合、トリガーが起動されます。
DROP	DROP を指定すると、DROP 文でデータ・ディクショナリのデータベース・オブジェクトを削除した場合、トリガーが起動されます。
GRANT	GRANT を指定すると、ユーザーが、別のユーザーまたはロールにシステム権限、ロールまたはオブジェクト権限を付与した場合、トリガーが起動されます。
NOAUDIT	NOAUDIT を指定すると、NOAUDIT 文で SQL 文またはスキーマ・オブジェクトの操作の追跡を停止させた場合、トリガーが起動されます。
RENAME	RENAME を指定すると、RENAME 文でデータベース・オブジェクトの名前を変更した場合、トリガーが起動されます。
REVOKE	REVOKE を指定すると、REVOKE 文でユーザーまたはロールからシステム権限、ロールまたはオブジェクト権限を取り消した場合、トリガーが起動されます。
TRUNCATE	TRUNCATE を指定すると、TRUNCATE 文で表またはクラスタから行を削除し、記憶特性を再設定した場合、トリガーが起動されます。
DDL	DDL を指定すると、前述の DDL 文のいずれかを発行した場合、トリガーが起動されます。

制限事項 : PL/SQL プロシージャを介して実行された DDL 操作は、トリガー・イベントとして指定することはできません。

database_event

トリガーを起動する 1 つ以上のデータベースの状態を指定します。特に指定がない限り、DATABASE または SCHEMA のイベントのトリガーを作成できます。これらのトリガー・イベントについて、Oracle が自律型トランザクションの有効範囲をオープンし、トリガーを起動して、（既存のユーザー・トランザクションには関係なく）別のトランザクションをコミットします。

> **参照 :** 自律型トランザクションの有効範囲の詳細は、『Oracle8*i* PL/SQL 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。

SERVERERROR　SERVERERROR を指定すると、サーバー・エラー・メッセージがログされた場合、トリガーが起動されます。

次のエラーが発生しても、SERVERERROR トリガーは起動されません（*string* には文字列が入ります）。

- ORA-01403: データが見つかりません。
- ORA-01422: 要求よりもより多くの行が取り出されました。
- ORA-01423: 行の取出しの検査中にエラーが発生しました。
- ORA-01034: Oracle は使用できません。
- ORA-04030: *string* バイト (*string*,*string*) を割り当てようとしてプロセス・メモリーが不足しました。

LOGON　LOGON を指定すると、クライアント・アプリケーションがデータベースにログオンした場合、トリガーが起動されます。

LOGOFF　LOGOFF を指定すると、クライアント・アプリケーションがデータベースにログオフした場合、トリガーが起動されます。

STARTUP　STARTUP を指定すると、データベースがオープンされた場合、トリガーが起動されます。

SHUTDOWN　SHUTDOWN を指定すると、データベースのインスタンスが停止された場合、トリガーが起動されます。

注意 :

- LOGON、STARTUP および SERVERERROR に適用されるのは AFTER トリガーのみです。
- LOGOFF および SHUTDOWN に適用されるのは BEFORE トリガーのみです。
- DATABASE にのみ適用されるのは AFTER STARTUP トリガーおよび BEFORE SHUTDOWN トリガーです。

ON *table* | *view*

ON 句によって、トリガーが作成されるデータベース・オブジェクトを決定できます。

[schema.] table | *view*

トリガーを作成する、次のいずれかのスキーマ、表またはビューの名前を指定します。

- 表またはビュー
- オブジェクト表またはオブジェクト・ビュー
- ネストした表型の列

schema を指定しない場合、この表が自スキーマに存在するとみなされます。トリガーは、索引構成表に作成できます。

制限事項 : トリガーは、SYS スキーマ内の表には作成できません。

NESTED TABLE

ビューの列 *nested_table_column* に定義するトリガーを指定します。そのようなトリガーは、DML の操作対象がネストした表の要素である場合にのみ起動されます。

制限事項 : NESTED TABLE は、INSTEAD OF トリガーにのみ指定できます。

DATABASE

DATABASE を指定すると、データベース全体にトリガーを定義します。

SCHEMA

SCHEMA を指定すると、現行のデータベースにトリガーを定義します。

referencing_clause

referencing_clause によって、相関名を指定できます。相関名は、特にカレント行における新旧の値を参照する場合に、PL/SQL ブロックおよび行レベル・トリガーの WHEN 句で使用します。デフォルトの相関名は OLD および NEW です。行レベル・トリガーを OLD または NEW という表に対応付ける場合は、この句を使用して異なる相関名を指定します。これにより、表名と相関名との混乱を避けることができます。

- トリガーが**ネストした表**に定義される場合、OLD および NEW はネストした表の行を参照し、PARENT は親表のカレント行を参照します。
- オブジェクト表またはビューにトリガーが定義されている場合、OLD および NEW はオブジェクト・インスタンスを参照します。

制限事項 : この句は、（DDL およびデータベース・イベント・トリガーではなく）DML イベント・トリガーにのみ有効です。

FOR EACH ROW

FOR EACH ROW を指定すると、トリガーを行トリガーとして設定します。行レベル・トリガーは、トリガーを起動する文の対象になり、かつ WHEN 条件で定義したオプションのトリガー制約を満たす行ごとに 1 回ずつ起動されます。

注意 : この句は DML イベントのみに適用され、DDL またはデータベースには適用されません。

INSTEAD OF トリガー以外のトリガーを指定する際に、この句を省略した場合、そのトリガーは文レベル・トリガーになります。文レベル・トリガーは、トリガーを起動する文が発行されたときにオプションのトリガー制約が満たされていると、1 回のみ起動されます。

INSTEAD OF トリガー文は、各行について暗黙的にアクティブになります。

WHEN

トリガー制約（Oracle がトリガーを起動するために必要な SQL 条件）を指定します。条件の構文については、5-15 ページの「条件」を参照してください。この条件には相関名を指定する必要があります。問合せは指定できません。

制限事項 :

- 行レベル・トリガーの場合のみ、トリガー制約を指定できます。トリガーを起動する文の対象となる行ごとに、この条件が評価されます。
- INSTEAD OF トリガー文にトリガー制約は指定できません。
- オブジェクト列、オブジェクト列の属性、VARRAY、ネストした表、LOB 列を参照できます。トリガー制約内では PL/SQL ファンクションおよびメソッドは起動できません。

pl/sql_block

Oracle がトリガーを起動するために実行する PL/SQL ブロックを指定します。

データベース・トリガーの PL/SQL ブロックは、システム・イベント属性の抽出用にのみ設計された SYS スキーマ内の一連の組込みファンクションの 1 つを指定できます。これらのファンクションは、データベース・トリガーの PL/SQL ブロック内でのみ使用できます。

制限事項：

- トリガーの PL/SQL ブロックは、ブロックが同一のトランザクションで実行される場合、トランザクション制御 SQL 文（COMMIT、ROLLBACK、SAVEPOINT および SET CONSTRAINT）を指定できません。
- PL/SQL ブロック内部のトリガー・アクションでは LOB 列を参照および使用できますが、トリガー・アクション内部でそれらの値を変更することはできません。

参照：

- PL/SQL ブロックの書込み方法など、PL/SQL の詳細は、『Oracle8*i* PL/SQL 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。
- これらのファンクションの詳細は、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。
- 『Oracle8*i* アプリケーション開発者ガイド 基礎編』も参照してください。

call_procedure_statement

call_procedure_statement 句によって、インライン・トリガー・コードを PL/SQL ブロックとして指定するかわりに、ストアド・プロシージャをコールできるようにします。この文の構文は、次の例外を除いて、8-126 ページの「CALL」の文と同じです。

- ファンクションにのみ適用されるため、CALL の INTO 句は指定できません。
- *expr* 内のバインド変数は指定できません。
- 定義されているトリガーの表の列を参照する場合、:NEW および :OLD を指定する必要があります。

参照： 10-77 ページの「トリガー本体のプロシージャのコール例」を参照してください。

例

DML トリガーの例　次の文は、スキーマ scott に emp_permit_changes という名前の BEFORE 文トリガーを作成します。そのトリガーを書き込み、この表に発行された DML 文に制限を配置します（たとえば、その文が発行できる場合）。

```
CREATE TRIGGER scott.emp_permit_changes
    BEFORE
    DELETE OR INSERT OR UPDATE
    ON scott.emp
       pl/sql block
```

DELETE、INSERT または UPDATE の各文がスキーマ scott 内の emp 表に対して実行されるたびに、このトリガーが起動されます。トリガー emp_permit_changes は BEFORE 文トリガーであるため、emp_permit_changes を起動する文が実行される前に 1 回起動されます。

制限付き DML トリガーの例　次の文は、スキーマ scott 内に salary_check という名前の BEFORE 行レベル・トリガーを作成します。たとえば、PL/SQL ブロックによって、従業員の給料が従業員の職種に対して設定された給料の範囲内にあるよう指定されたとします。

```
CREATE TRIGGER scott.salary_check
    BEFORE
    INSERT OR UPDATE OF sal, job ON scott.emp
    FOR EACH ROW
    WHEN (new.job <> 'PRESIDENT')
       pl/sql_block
```

このトリガーは、次の文のいずれかが発行されるたびに起動されます。

- emp 表に対して行を追加する INSERT 文
- emp 表の sal 列値または job 列値を変更する UPDATE 文

salary_check は BEFORE 行レベル・トリガーであるため、UPDATE 文で更新される各行を変更または INSERT 文で挿入される各行を追加する前に、このトリガーを起動します。

salary_check には、社長の給料をチェックできないトリガー制約が設定されています。

トリガー本体のプロシージャのコール例　PL/SQL ブロック内のトリガー本体を設定するかわりに、コール側プロシージャによって前述の例で記述された salary_check トリガーを作成することができます。従業員の給料が適切な範囲内にあることを確認する scott.salary_check プロシージャを定義したと仮定し、次のようにトリガー salary_check を作成できます。

```
CREATE TRIGGER scott.salary_check
   BEFORE INSERT OR UPDATE OF sal, job ON scott.emp
   FOR EACH ROW
   WHEN (new.job <> 'PRESIDENT')
   CALL check_sal(:new.job, :new.sal, :new.ename);
```

check_sal プロシージャは、PL/SQL、C または Java で実装されます。また、:NEW 値のかわりに CALL 句の :OLD を指定できます。

データベース・イベント・トリガーの例　次の文は、すべてのエラーをログするトリガーを作成します。PL/SQL ブロックは、特定のエラー（無効ログイン、エラー番号 1017）に対する特別な処理を行います。このトリガーは AFTER 文トリガーであるため、（ログインの失敗のように）文の実行が失敗した後で起動されます。

```
CREATE TRIGGER log_errors AFTER SERVERERROR ON DATABASE
   BEGIN
      IF (IS_SERVERERROR (1017)) THEN
         <special processing of logon error>
      ELSE
         <log error number>
      END IF;
   END;
```

DDL トリガーの例　次の文は、任意の DDL 文 CREATE の AFTER 文トリガーを作成します。このトリガーを使用して、自スキーマにある新規データ・ディクショナリ・オブジェクトの作成を監査します。

```
CREATE TRIOGGER audit_db_object AFTER CREATE
   ON SCHEMA
      pl/sql_block
```

INSTEAD OF トリガーの例　次の文は、顧客データを 2 つの表に格納します。all_customers オブジェクト・ビューは 2 つの表 customers_sj および customers_pa の UNION として作成されています。INSTEAD OF トリガーを使用して値を挿入します。

```
CREATE TABLE customers_sj
  ( cust    NUMBER(6),
    address VARCHAR2(50),
    credit   NUMBER(9,2)  );

CREATE TABLE customers_pa
  ( cust    NUMBER(6),
    address VARCHAR2(50),
    credit   NUMBER(9,2) );

CREATE TYPE customer_t AS OBJECT
  ( cust    NUMBER(6),
    address   VARCHAR2(50),
    credit    NUMBER(9,2),
    location   VARCHAR2(20)  );

CREATE VIEW all_customers (cust)
    AS SELECT customer_t (cust, address, credit, 'SAN_JOSE')
    FROM   customers_sj
  UNION ALL
    SELECT customer_t (cust, address, credit, 'PALO_ALTO')
```

```
    FROM   customers_pa;

CREATE TRIGGER instrig INSTEAD OF INSERT ON all_customers
   FOR EACH ROW
     BEGIN
      IF (:new.cust.location = 'SAN_JOSE') THEN
        INSERT INTO customers_sj
        VALUES (:new.cust.cust, :new.cust.address,:new.cust.credit);
      ELSE
       INSERT INTO customers_pa
       VALUES (:new.cust.cust, :new.cust.address, :new.cust.credit);
      END IF;
    END;
```

CREATE TYPE

用途

CREATE TYPE 文は、**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**を指定する場合に使用します。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、その型でのメソッドに対するコードが含まれます。

注意： メソッドではなく、属性のみを宣言する型仕様を持つオブジェクト型を作成した場合は、オブジェクト型本体を指定する必要はありません。

ユーザー定義型を作成した場合、Oracle は、暗黙的に、そのコンストラクタ・メソッドを定義します。**コンストラクタ**は、システム提供のプロシージャで、SQL 文または PL/SQL コード内で、その型の値のインスタンスを構成するために使用します。コンストラクタ・メソッドの名前は、ユーザー定義型の名前と同じです。

オブジェクト型のコンストラクタ・メソッドのパラメータは、オブジェクト型のデータ属性です。また、そのオブジェクト型用の属性定義と同じ順序で定義されます。ネストした表または VARRAY コンストラクタのパラメータは、ネストした表または VARRAY の要素です。

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、10-93 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、『Oracle8*i* PL/SQL ユーザーズ・ガイドおよびリファレンス』、『Oracle8*i* アプリケーション開発者ガイド 基礎編』および『Oracle8*i* 概要』を参照してください。

前提条件

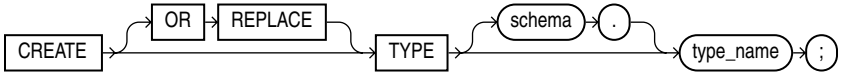
自スキーマ内に型を作成する場合は、CREATE TYPE システム権限が必要です。他のユーザーのスキーマ内に型を作成する場合は、CREATE ANY TYPE システム権限が必要です。これらの権限は、明示的に取得することもロールを介して受け取ることもできます。

型の所有者には、表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。

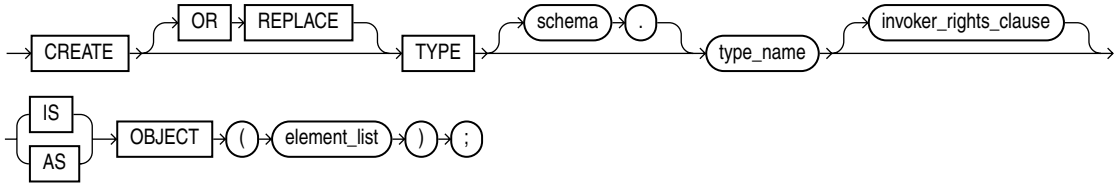
型の所有者が、型にアクセスする権限を他のユーザーに付与する場合、所有者には、参照型に対する GRANT OPTION 付きの EXECUTE オブジェクト権限、または ADMIN OPTION 付きの EXECUTE ANY TYPE システム権限が必要です。これらの権限がないと、型の所有者は、型にアクセスする権限を他のユーザーに付与できません。

構文

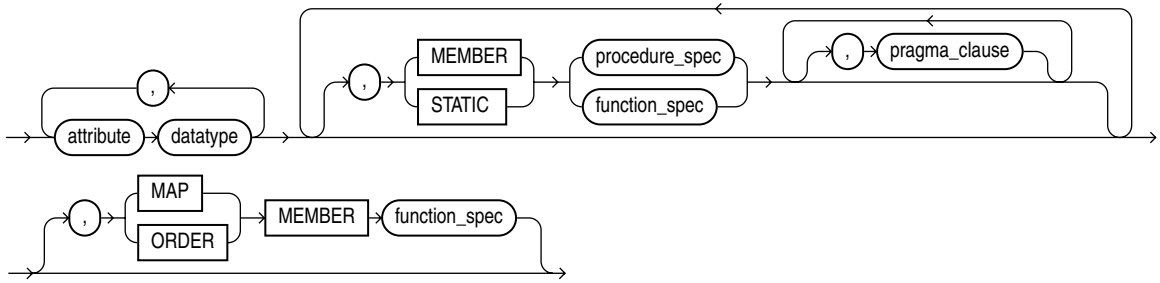
create_incomplete_type::=



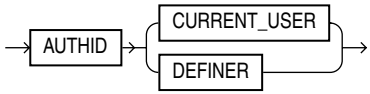
create_object_type::=



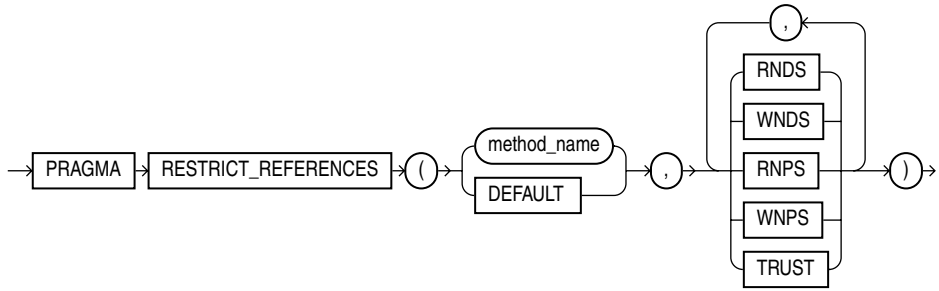
element_list::=



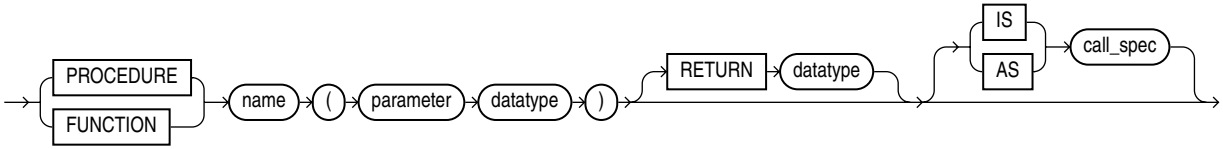
invoker_rights_clause::=



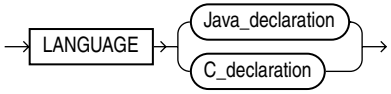
pragma_clause::=



procedure_spec または function_spec::=

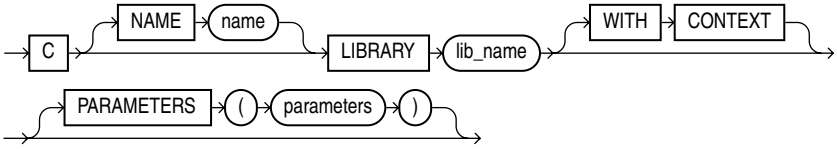


call_spec::=

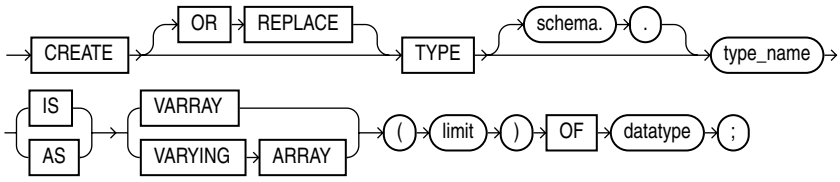


Java_declaration::=

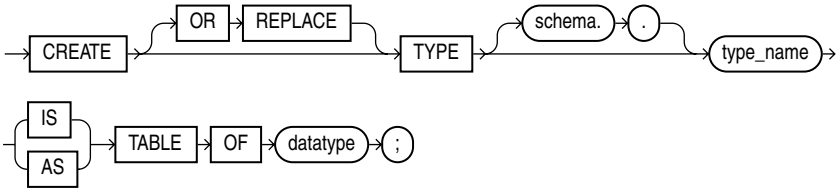
C_declaration::=



create_varray_type::=



create_nested_table_type::=



キーワードとパラメータ

OR REPLACE

既存の型を再作成する場合は、OR REPLACE を指定します。この句を指定した場合、既存のトリガー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よって異なる場合、その索引には DISABLED のマークが付きます。

schema

作成する型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、現行スキーマにその型が作成されます。

type_name

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。

型の作成時にコンパイル・エラーが発生した場合、エラーを戻します。SQL*Plus コマンド SHOW ERRORS を使用して、関連するコンパイラ・エラー・メッセージを表示できます。

create_object_type

create_object_type 句を使用すると、（不完全型ではなく）ユーザー定義型オブジェクト型を作成できます。データ構造を形成する変数を**属性**といいます。オブジェクトの動作を定義するメンバー・サブプログラムを**メソッド**といいます。AS OBJECT は、オブジェクト型の作成時に必要です。

invoker_rights_clause

invoker_rights_clause によって、オブジェクト型のメンバー・ファンクションおよびプロシージャが、そのオブジェクト型を所有するユーザーのスキーマで、特権付きで実行されるか、または、CURRENT_USER のスキーマで、特権付きで実行されるかを指定します。この仕様は、対応する型本体にも適用されます。

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。

制限事項 : この句はオブジェクト型のみに指定でき、ネストした表型および VARRAY 型には指定できません。

AUTHID CURRENT_USER	CURRENT_USER を指定して、オブジェクト型のメンバー・ファンクションおよびプロシージャを CURRENT_USER の権限で実行します。この句によって**実行者権限型**が作成されます。 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。その他すべての文の外部名は、型が存在するスキーマ内で変換します。
AUTHID DEFINER	DEFINER によって、ファンクションおよびプロシージャが存在するスキーマの所有者権限で、オブジェクト型のメンバー・ファンクションおよびプロシージャを実行し、メンバー・ファンクションおよびプロシージャが存在するスキーマで外部名を変換することを指定します。これはデフォルト値です。

参照 :

- CURRENT_USER を判断する方法については、『Oracle8*i* 概要』および『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。
- 『Oracle8*i* PL/SQL ユーザーズ・ガイドおよびリファレンス』も参照してください。

element_list

datatype

属性の Oracle 組込みデータ型またはユーザー定義型の名前を指定します。

制限事項：

- ROWID 型、LONG 型または LONG ROW 型の属性は指定できません。
- NCLOB、NCHAR または NVARCHAR2 の属性のオブジェクトを作成することはできませんが、メソッドにこれらデータ型のパラメータを指定することはできます。
- ユーザー定義オブジェクト型に対して UROWID のデータ型は指定できません。
- REF 型のオブジェクトを指定する場合、ターゲット・オブジェクトにはオブジェクト識別子が必要です。

  参照：指定可能な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。

attribute

オブジェクト型に、オブジェクト属性の名前を指定します。属性とは、名前と型指定子を持つオブジェクト構造を形成するデータ項目です。各オブジェクト型には、1 つ以上の属性を指定する必要があります。

MEMBER *procedure_spec* または *function_spec*

属性として参照されるオブジェクト型に関連付けられたファンクションまたはプロシージャ・サブプログラムを指定します。通常、object_expression.method() のように、「自己参照的」形式でメンバー・メソッドを起動します。このクラスのメソッドには、メソッド本体で SELF として参照される暗黙的な最初の引数があります。この引数は、メソッドが起動されるオブジェクトを表します。

STATIC *procedure_spec* または *function_spec*

オブジェクト型に関連付けられたファンクションまたはプロシージャ・サブプログラムを指定します。メンバー・メソッドとは異なり、スタティック・メソッドには暗黙的なパラメータがありません（SELF はそれ自体では参照できません）。通常、type_name.method() として起動されます。

メンバー・メソッドとスタティック・メソッドの両方に、各プロシージャまたはファンクションの仕様部について、オブジェクト型本体に対応するメソッド本体を指定する必要があります。

RETURN 句はファンクションにのみ有効です。構文は省略形です。

このサブプログラムは、プロシージャまたはファンクションの宣言を含みませんが、対応する CREATE TYPE BODY 文は発行する必要があります。

参照：

- メソッド起動およびメソッドの詳細は、『Oracle8*i* PL/SQL 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。
- 使用可能なすべての句を含む完全な構文については、9-127 ページの「CREATE PROCEDURE」および 9-42 ページの「CREATE FUNCTION」を参照してください。
- 10-93 ページの「CREATE TYPE BODY」も参照してください。
- ユーザー定義ファンクションの制限事項については、9-45 ページの「ユーザー定義ファンクションの制限事項」を参照してください。

call_spec

JAVA または C メソッドの名前、パラメータ型および戻り型を SQL にマップするコール仕様を指定します。型のメンバー・メソッドすべてがこの句で定義された場合、対応する CREATE TYPE BODY 文を発行する必要はありません。

Java_declaration では、'*string*' が JAVA 実装メソッドを定義します。

参照：

- 『Oracle8*i* Java ストアド・プロシージャ開発者ガイド』を参照してください。
- パラメータおよび *C_declaration* のセマンティックについては、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

pragma_clause

pragma_clause によって、コンパイラ・ディレクティブを指定できます。

PRAGMA RESTRICT_REFERENCES

PRAGMA RESTRICT_REFERENCES コンパイラ・ディレクティブは、データベースの表またはパッケージ変数（あるいはその両方）に対するメンバー・ファンクションの読み書きアクセスを拒否し、副作用の発生を防止します。

参照：『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

method_name

プラグマが適用されている MEMBER ファンクションまたはプロシージャの名前を指定します。

DEFAULT	DEFAULT を指定すると、プラグマが明示的に指定されていない型のすべてのメソッドにプラグマが適用されます。
WNDS	WNDS を指定すると、データベースに**書込み禁止状態**制約（データベース表を変更しない）が適用されます。
WNPS	WNPS を指定すると、パッケージに**書込み禁止状態**制約（パッケージ変数を変更しない）が適用されます。
RNDS	RNDS を指定すると、データベースに**読込み禁止状態**制約（データベース表を問い合せない）が適用されます。
RNPS	RNPS を指定すると、パッケージに**読込み禁止状態**制約（パッケージ変数を参照しない）が適用されます。
TRUST	TRUST を指定すると、プラグマに指定されている制限が、実際に適用されているのではなく、単に真であることを指定します。

MAP MEMBER *function_spec*

この句は、オブジェクトの順序付けられたすべてのインスタンスの中から、指定したインスタンスの相対的な位置を戻すメンバー・ファンクション（MAP メソッド）を指定します。MAP メソッドは暗黙的にコールされ、オブジェクト・インスタンスを事前定義済のスカラー型の値にマップすることによって、それらのオブジェクト・インスタンスに順序を設定します。PL/SQL は、この順序を使用してブール式の計算と比較を行います。

MAP メソッドの引数が NULL の場合、MAP メソッドは NULL に戻り、メソッドは起動されません。

オブジェクトの仕様部には、1 つの MAP メソッドのみを指定することができます。この MAP メソッドは、ファンクションである必要があります。結果として生成される型は、事前定義済の SQL スカラー型である必要があります。また、MAP ファンクションに指定できる引数は、暗黙の SELF 引数のみです。

> **注意：**（ORDER BY 句、GROUP BY 句、DISTINCT 句または UNION 句を使用する）ソートまたは結合を指定した問合せが *type_name* を参照し、これらの問合せをパラレル化する場合は、MAP メンバー・ファンクションを指定する必要があります。

ORDER MEMBER *function_spec*

この句によって、オブジェクトのインスタンスを明示的な引数および暗黙的な SELF 引数として取るメンバー・ファンクション（ORDER メソッド）を指定し、負の整数、0（ゼロ）または正の整数のいずれかを戻します。負、正または 0（ゼロ）は、それぞれ、暗黙的な SELF 引数が明示的な引数より小さい、等しいまたは大きいことを示します。

ORDER メソッドの引数が NULL の場合、ORDER メソッドは NULL を戻し、メソッドは起動されません。

同じオブジェクト型定義の各インスタンスを ORDER BY 句の中で比較した場合、ORDER メソッド *function_spec* が呼び出されます。

オブジェクトの仕様部には、1 つの ORDER メソッドのみ指定でき、その ORDER メソッドは戻り型が NUMBER のファンクションである必要があります。

型の指定では、MAP メソッドまたは ORDER メソッドのいずれかを定義できますが、両方を定義することはできません。いずれかのメソッドを宣言すると、SQL 内でオブジェクト・インスタンスを比較できます。

MAP メソッドも ORDER メソッドも定義されない場合、比較できるのは等価性または非等価性のみです。したがって、オブジェクト・インスタンスに順序を付けることはできません。同じ型定義のインスタンスは、それぞれの対応する属性の各組が等しい場合にのみ等しくなります。2 つのオブジェクト型の等価性を判断するために比較方法を指定する必要はありません。

オブジェクト・インスタンスで大規模のソートまたはハッシュ結合処理を実行する場合に、MAP を使用してください。MAP を使用してオブジェクトをスカラー値にマップすると、そのスカラーは、ソート中およびマージ中に使用されます。MAP メソッドは、各オブジェクトを比較するメソッドを起動する必要がある ORDER メソッドより効率的です。ハッシュ結合には MAP を使用する必要があります。ハッシュ・メカニズムはオブジェクト値でハッシュするため、ORDER メソッドを使用することはできません。

参照 : オブジェクト値の比較の詳細は、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

create_varray_type

create_varray_type は、それぞれが同じデータ型を持つ要素の順序付け集合としての型を作成します。名前および 0（ゼロ）以上の最大限度を指定する必要があります。配列限度は、整数リテラルである必要があります。Oracle では、無名の VARRAY はサポートされません。

VARRAY 内に含まれるオブジェクトの型名は、次のいずれかである必要があります。

- 組込みデータ型
- REF
- オブジェクト型

制限事項：

- 1 つのコレクション型には、直接的か間接的かに関係なく、他のコレクション型を入れることはできません。VARRAY 型に、VARRAY およびネストした表の要素を含めることはできません。
- LOB データ型の VARRAY 型は作成できません。

create_nested_table_type

create_nested_table_type によって、データ型で名前付きのネストした表を作成します。

- *datatype* がオブジェクト型である場合、ネストした表型は、オブジェクト型の名前および属性と一致する列を持つ表を記述します。
- データ型がスカラー型である場合、ネストした表の型は、「column_value」という 1 つのスカラー型列を持つ表を記述します。

制限事項：

- 1 つのコレクション型には、直接的か間接的かに関係なく、他のコレクション型を入れることはできません。ネストした表型に、VARRAY およびネストした表の要素を含めることはできません。
- *datatype* に NCLOB は指定できません。ただし、CLOB または BLOB は指定できます。

例

オブジェクト型の例　次の文は、LOB 属性を持つオブジェクト型 person_t を作成します。

```
CREATE TYPE person_t AS OBJECT
  (name    CHAR(20),
   resume  CLOB,
   picture BLOB);
```

VARRAY 型の例　次の文は、members_type を、100 の要素を持つ VARRAY 型として作成します。

```
CREATE TYPE members_type AS VARRAY(100) OF CHAR(5);
```

ネストした表型の例　次の文は、オブジェクト型 project_t の名前付き表型 project_table を作成します。

```
CREATE TYPE project_t AS OBJECT
  (pno CHAR(5),
   pname CHAR(20),
   budgets DEC(7,2));

CREATE TYPE project_table AS TABLE OF project_t;
```

コンストラクタの例　次の文は、メソッド・コンストラクタ col.getbar() を起動します。

```
CREATE TYPE foo AS OBJECT (a1 NUMBER,
                  MEMBER FUNCTION getbar RETURN NUMBER);
CREATE TABLE footab(col foo);

SELECT col.getbar() FROM footab;
```

ファンクションとは異なり、メソッドを起動する場合は、メソッドが他に引数を取らない場合でもカッコが必要です。

次の文は、システム定義のコンストラクタを起動して、foo_t オブジェクトを構成し、構成したオブジェクトを TAB 表 foo_tab に入れます。

```
CREATE TYPE foo_t AS OBJECT (a1 NUMBER, a2 NUMBER);
CREATE TABLE foo_tab (b1 NUMBER, b2 foo_t);
INSERT INTO foo_tab VALUES (1, foo_t(2,3));
```

> **参照：**　コンストラクタの詳細は、『Oracle8*i* アプリケーション開発者ガイド 基礎編』および『Oracle8*i* PL/SQL 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。

スタティック・メソッドの例　次の文は、employee_t 型の定義を変更し、変更した定義を construct_emp ファンクションと関連付けます。

```
CREATE OR REPLACE TYPE employee_t AS OBJECT(
   empid RAW(16),
   ename CHAR(31),
   dept   REF department_t,
   STATIC function construct_emp
      (name VARCHAR2, dept REF department_t)
   RETURN employee_t
  );
```

この文は、次の型本体の文（PL/SQL 文はイタリックの部分）が必要です。

```
CREATE OR REPLACE TYPE BODY employee_t IS
   STATIC FUNCTION construct_emp
      (name varchar2, dept REF department_t)
   RETURN employee_t IS
      BEGIN
         return employee_t(SYS_GUID(),name,dept);
      END;
   END;
```

この型および型本体の定義で次の操作を実行できます。

```
INSERT INTO emptab
   VALUES (employee_t.construct_emp('John Smith', NULL));
```

CREATE TYPE BODY

用途

CREATE TYPE BODY は、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場合に使用します。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、その型での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。

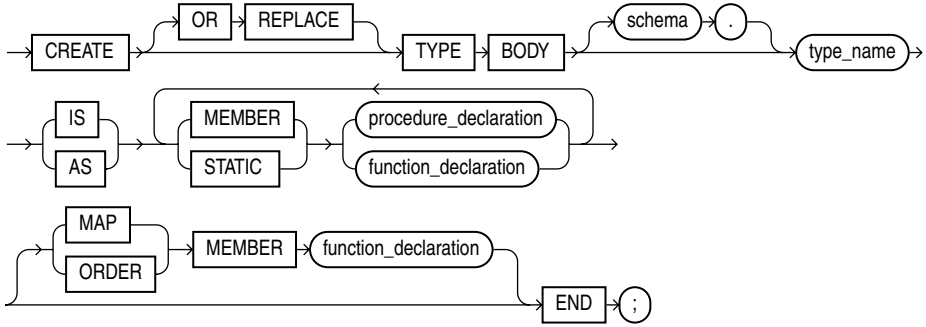
> **参照：** 型仕様部の作成および変更については、10-80 ページの「CREATE TYPE」および 8-79 ページの「ALTER TYPE」を参照してください。

前提条件

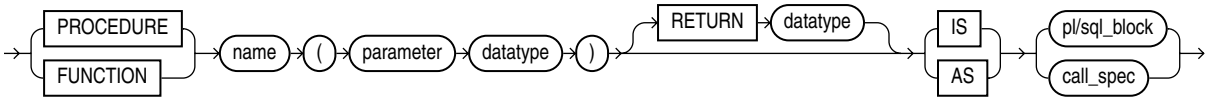
オブジェクト型用の CREATE TYPE 仕様部で行われるすべてのメンバー宣言には、それに対応する構造が CREATE TYPE 文または CREATE TYPE BODY 文内に存在する必要があります。

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。

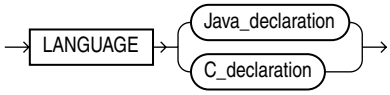
構文



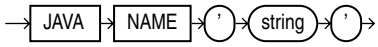
procedure_declaration | function_declaration::=



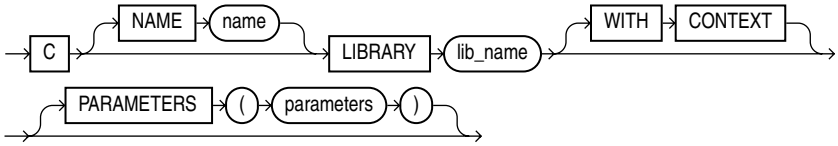
call_spec::=



Java_declaration::=



C_declaration::=



キーワードとパラメータ

OR REPLACE

既存の型本体を再作成する場合は、OR REPLACE を指定します。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。

schema

作成する型本体が定義されるスキーマを指定します。*schema* を省略した場合、Oracle では現行の自スキーマ内に型本体が作成されます。

type_name

オブジェクト型の名前を指定します。

IS | AS

MEMBER | STATIC

オブジェクト型仕様部に関連付けられたメソッド・ファンクションまたはプロシージャ・サブプログラムの型を指定します。

各プロシージャまたはファンクション宣言には、対応するメソッド名およびオプション・パラメータ・リスト、また、ファンクションの場合は、オブジェクト型仕様部の戻り型を定義する必要があります。

procedure_declaration

プロシージャ・サブプログラムを宣言します。

function_declaration

ファンクション・サブプログラムを宣言します。

参照：

- ユーザー定義ファンクションの制限事項については、10-80 ページの「CREATE TYPE」を参照してください。
- パッケージ内のサブプログラム名のオーバーロードについては、『Oracle8*i* PL/SQL 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。
- 9-127 ページの「CREATE PROCEDURE」、9-42 ページの「CREATE FUNCTION」および『Oracle8*i* アプリケーション開発者ガイド 基礎編』も参照してください。

MAP | ORDER

MAP MEMBER

MAP MEMBER を指定すると、メンバー・ファンクション（MAP メソッド）が定義または実装されます。メンバー・ファンクションは、オブジェクトのすべてのインスタンスの順序付けにおいて、指定したインスタンスの相対的な位置を戻します。MAP メソッドは暗黙的にコールされ、オブジェクト・インスタンスを事前定義済のスカラー型の値にマップすることにより、それらのオブジェクト・インスタンスに順番を指定します。PL/SQL は、この順序を使用してブール式の計算と比較を行います。

MAP メソッドの引数が NULL の場合、MAP メソッドは NULL に戻り、メソッドは起動されません。

オブジェクト型本体には、1 つの MAP メソッドのみを指定することができます。この MAP メソッドは、ファンクションである必要があります。この MAP ファンクションに指定できる引数は、暗黙的な SELF 引数のみです。

ORDER MEMBER

ORDER MEMBER を指定すると、オブジェクトのインスタンスを明示的な引数および暗黙的な SELF 引数として取るメンバー・ファンクション（ORDER メソッド）を指定し、負の整数、0（ゼロ）または正の整数のいずれかを戻します。負、正または 0（ゼロ）は、それぞれ、暗黙的な SELF 引数が明示的な引数より小さい、等しいまたは大きいことを示します。

ORDER メソッドの引数が NULL の場合、ORDER メソッドは NULL を戻し、メソッドは起動されません。

同じオブジェクト型定義の各インスタンスを ORDER BY 句の中で比較した場合、ORDER メソッドの *function_spec* が起動されます。

オブジェクトの仕様部には、1 つの ORDER メソッドのみ指定でき、その ORDER メソッドは戻り型が NUMBER のファンクションである必要があります。

MAP メソッドまたは ORDER メソッドのいずれかを宣言できますが、両方は宣言できません。いずれかのメソッドを宣言すると、SQL 内でオブジェクト・インスタンスを比較できます。

どちらのメソッドも宣言しない場合、比較できるのはオブジェクト・インスタンスの等価性と非等価のみです。同じ型定義のインスタンスは、それぞれの対応する属性の各組が等しい場合にのみ等しくなります。

procedure_declaration | *function_declaration*

プロシージャまたはファンクション・サブプログラムを宣言します。RETURN 句はファンクションにのみ有効です。構文は省略形です。

参照 : 使用可能なすべての句を含む完全な構文については、9-127 ページの「CREATE PROCEDURE」および 9-42 ページの「CREATE FUNCTION」を参照してください。

pl/sql_block

プロシージャまたはファンクションを宣言します。

参照 : 『Oracle8*i* PL/SQL 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。

call_spec

JAVA または C メソッド名、パラメータ型および戻り型を SQL にマップするコール仕様を指定します。

Java_declaration では、'*string*' が JAVA 実装メソッドを定義します。

参照 :

- 『Oracle8*i* Java ストアド・プロシージャ開発者ガイド』を参照してください。
- パラメータおよび *C_declaration* のセマンティックについては、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

AS EXTERNAL

AS EXTERNAL は、C メソッドを宣言するもう 1 つの方法です。この句は以前のリリースのもので、下位互換用にのみサポートされています。*call_spec* 構文は、*C_declaration* とともに使用することをお薦めします。

例

型本体の作成例　次のオブジェクト型本体は、リレーショナル用のメンバー・サブプログラムを実装します（PL/SQL 文はイタリック体の部分です）。

```
CREATE TYPE BODY rational
  IS
   MAP MEMBER FUNCTION rat_to_real RETURN REAL IS
      BEGIN
         RETURN numerator/denominator;
      END;

   MEMBER PROCEDURE normalize IS
      gcd NUMBER := integer_operations.greatest_common_divisor
                    (numerator, denominator);
      BEGIN
         numerator := numerator/gcd;
         denominator := denominator/gcd;
      END;

   MEMBER FUNCTION plus(x rational) RETURN rational IS
      r rational := rational_operations.make_rational
                     (numerator*x.denominator +
                      x.numerator*denominator,
                      denominator*x.denominator);
      BEGIN
         RETURN r;
      END;

   END;
```

CREATE USER

用途

CREATE USER 文は、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）を作成する場合に使用します。その結果、Oracle 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。

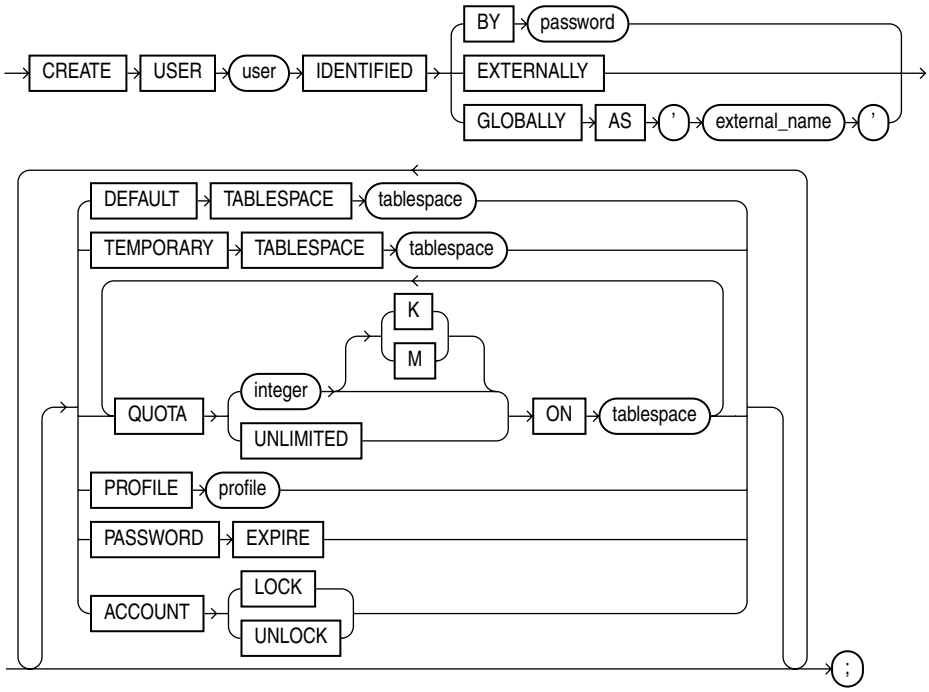
注意： プロキシ（アプリケーションまたはアプリケーション・サーバー）を経由して、ユーザーが Oracle に接続できるようになります。構文および説明は、8-87 ページの「ALTER USER」を参照してください。

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 にログイ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際、少なくとも CREATE SESSION 権限をそのユーザーに付与してください。

参照： 11-31 ページの「GRANT」を参照してください。

構文



キーワードとパラメータ

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、2-81 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。

IDENTIFIED

IDENTIFIED 句によって、ユーザー認証をどのように行うか示します。

> **参照：** 詳細は、『Oracle8*i* アプリケーション開発者ガイド 基礎編』およびご使用のオペレーティング・システム用のドキュメントを参照してください。

BY *password*

ローカル・ユーザーを作成し、ログインするときに、パスワードを指定する必要があることを示します。データベース・キャラクタ・セットにマルチバイト文字が含まれている場合でも、パスワードを指定できるのは、データベース・キャラクタ・セットのシングルバイト文字のみです。

Oracle 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、2-81 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。このルーチンを UTLPWDMG.SQL スクリプトで実装します。詳細は、『Oracle8*i* 管理者ガイド』を参照してください。

参照 : パスワード管理およびパスワード保護の詳細は、『Oracle8*i* 管理者ガイド』を参照してください。

EXTERNALLY

外部ユーザーを作成し、外部サービス（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）によってユーザーが認証される必要があることを指定します。その結果、オペレーティング・システムのログイン認証によって、特定のオペレーティング・システム・ユーザーから特定のデータベース・ユーザーへのアクセスが可能になります。

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。詳細は、『Oracle8*i* 管理者ガイド』を参照してください。

GLOBALLY AS '*external_name*'

グローバル・ユーザーを作成し、ユーザーをエンタープライズ・ディレクトリ・サービスによって認証する必要があることを指定します。'*external_name*' 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・ディクショナリ・サービスの X.509 という名前。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。
- NULL 文字列（''）は、エンタープライズ・ディクショナリ・サービスが、認証されたグローバル・ユーザーを、適切なデータベース・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。

注意：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 Advanced Security 管理者ガイド』を参照してください。
- 8-87 ページの「ALTER USER」も参照してください。

DEFAULT TABLESPACE

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、オブジェクトはデフォルトで SYSTEM 表領域に格納されます。

参照： 表領域の詳細は、10-56 ページの「CREATE TABLESPACE」を参照してください。

TEMPORARY TABLESPACE

ユーザーの一時セグメントが確保される表領域を指定します。この句を省略した場合、一時セグメントはデフォルトで SYSTEM 表領域に確保されます。

QUOTA

QUOTA 句を使用すると、ユーザーは表領域中で領域を割り当てることが許可されます。任意の *integer* バイトで割当て制限を設定できます。K または M を使用して、KB または MB 単位で指定することもできます。この割当て制限は、ユーザーが割当て可能な表領域の最大領域です。

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、表領域への領域を無制限に割り当てることができます。

PROFILE

プロファイルを指定すると、ユーザーへの割当てを削除できます。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。

参照： 11-31 ページの「GRANT」および 9-134 ページの「CREATE PROFILE」を参照してください。

PASSWORD EXPIRE

ユーザーのパスワードを期限切れにする場合は、PASSWORD EXPIRE を指定します。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）にパスワードを変更させます。

ACCOUNT

ACCOUNT LOCK	ACCOUNT LOCK を指定すると、ユーザー・アカウントをロックし、アクセスを禁止にします。
ACCOUNT UNLOCK	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、アクセスを可能にします。

例

ユーザーの作成例　PASSWORD EXPIRE 指定して新規ユーザーを作成する場合、その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。次の文を発行することによって、sidney ユーザーを作成できます。

```
CREATE USER sidney
    IDENTIFIED BY welcome
    DEFAULT TABLESPACE cases_ts
    QUOTA 10M ON cases_ts
    TEMPORARY TABLESPACE temp_ts
    QUOTA 5M ON system
    PROFILE engineer
    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 welcome
- 10MB の割当て制限のあるデフォルト表領域 cases_ts
- 一時表領域 temp_ts
- 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 engineer 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

オペレーティング・システム・アカウント george によってのみアクセス可能なユーザーを作成する場合、george 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$」の場合、次の文でユーザー ops$george を作成できます。

```
CREATE USER ops$george
    IDENTIFIED EXTERNALLY
    DEFAULT TABLESPACE accs_ts
    TEMPORARY TABLESPACE temp_ts
    QUOTA UNLIMITED ON accs_ts;
```

ユーザー ops$george には、さらに次の特性があります。

- デフォルト表領域 accs_ts
- デフォルト一時表領域 temp_ts
- 表領域 accs_ts および temp_ts 上に無制限の領域
- DEFAULT プロファイルによって定義されるデータベース・リソースの制限

次の文は、ユーザー cindy をグローバル・ユーザーとして作成します。

```
CREATE USER cindy
   IDENTIFIED GLOBALLY AS 'CN=cindy,OU=division1,O=oracle,C=US'
   DEFAULT TABLESPACE legal_ts
   QUOTA 20M ON legal_ts
   PROFILE lawyer;
```

CREATE VIEW

用途

CREATE VIEW 文は、**ビュー**を定義する場合に使用します。ビューとは、1 つ以上の表またはビューに基づく論理表です。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**ベース表**といいます。

LOB およびオブジェクト・データ型（オブジェクト型、REF、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・メカニズムで作成します。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。

参照：

- ビューの様々な型およびその使用方法については、『Oracle8*i* 概要』、『Oracle8*i* アプリケーション開発者ガイド 基礎編』および『Oracle8*i* 管理者ガイド』を参照してください。
- ビューの変更については、8-93 ページの「ALTER VIEW」を参照してください。
- データベースからのビューの削除については、11-21 ページの「DROP VIEW」を参照してください。

前提条件

自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。

ビューを含む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付与されるのではなく、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、次のいずれかが真である必要があります。

- 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 システム権限がある。
- 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： ビューの所有者に必要な権限および作成するビューのベース表またはビューの詳細は、11-88 ページの「SELECT および副問合せ」、11-52 ページの「INSERT」、11-141 ページの「UPDATE」および 10-114 ページの「DELETE」を参照してください。

パーティション・ビュー

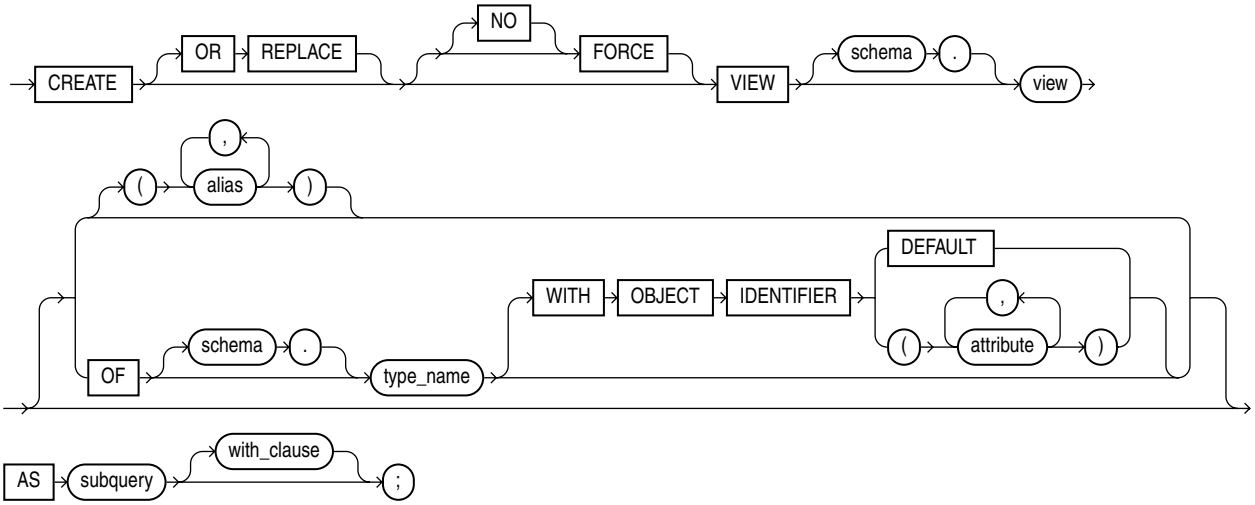
パーティション・ビューは、パーティション化機能を必要とするアプリケーション向けに、リリース 7.3 で導入されたものです。Oracle8*i* では、パーティション・ビューがサポートされているため、何も変更しなくてもリリース 7.3 からのアプリケーションのアップグレードができます。通常、Oracle8*i* に移行した後で、パーティション・ビューをパーティションに移行します。

Oracle8*i* では、`CREATE TABLE` 文を使用して、パーティション表を簡単に作成できます。パーティション表には、パーティション・ビューと同じメリットがあるのみでなく、パーティション・ビューのデメリットも補います。通常の動作環境では、パーティション・ビューではなく、パーティション表を使用することをお薦めします。

参照：

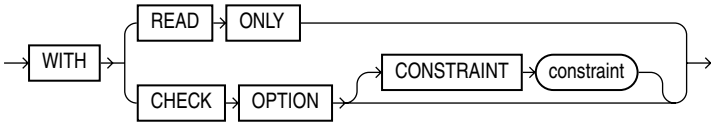
- パーティション・ビューのデメリットについては、『Oracle8*i* 概要』を参照してください。
- パーティション・ビューのパーティションへの移行については、『Oracle8*i* 管理者ガイド』を参照してください。
- パーティション表の詳細は、10-7 ページの「CREATE TABLE」を参照してください。

構文



subquery: 11-88 ページの「SELECT および副問合せ」を参照してください。

with_clause::=



キーワードとパラメータ

OR REPLACE

既存のビューを再作成する場合は、OR REPLACE を指定します。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、それらが再使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・ビューは問合せの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・ログのリフレッシュについては、7-59 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、『Oracle8*i* 概要』を参照してください。
- INSTEAD OF 句については、10-66 ページの「CREATE TRIGGER」を参照してください。

FORCE

ビューのベース表または参照先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んでいるスキーマの所有者が、それらの表やオブジェクトに対する権限を持っているかにかかわらず、強制的にビューを作成する場合は、FORCE を指定します。SELECT、INSERT、UPDATE または DELETE 文をビューに対して発行する場合、前述の条件が真である必要があります。

NO FORCE

ベース表が存在し、ビューを含むスキーマの所有者がその権限を持っている場合のみビューを作成する場合は、NOFORCE を指定します。これはデフォルト値です。

schema

ビューが定義されるスキーマを指定します。*schema* を省略した場合、自スキーマに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。

制限事項：あるビューが INSTEAD OF トリガーを持っている場合、そのビューで作成されるビューが更新可能であっても、INSTEAD OF トリガーが必要です。

alias

ビューの問合せによって選択された式に対して名前を指定します。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、次の Oracle スキーマ・オブジェクトのネーミングの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、Oracle はビューの問合せの列または列の別名から別名を導出します。このため、ビューの問合せが列の名前のみでなく式を含んでいる場合は、別名を使用する必要があります。

制限事項 : オブジェクト・ビュー作成時に別名は指定できません。

> **参照 :** 2-86 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」を参照してください。

OF *type_name*

type_name 型のオブジェクト・ビューを明示的に作成します。オブジェクト・ビューの列が、*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（OID）に関連付けられます。*schema* を省略した場合、自スキーマ内にオブジェクト・ビューが作成されます。

WITH OBJECT IDENTIFIER

WITH OBJECT IDENTIFIER によって、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できます。ほぼすべての場合、各属性はベース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、Oracle はエラーを戻します。

> **注意 :** 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

各行を一意に識別するために、基に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する場合は、DEFAULT を指定します。

attribute

作成対象のオブジェクト・ビューの基になるオブジェクト型の属性を指定します。

> **参照 :** オブジェクト作成の詳細は、10-80 ページの「CREATE TYPE」を参照してください。

AS *subquery*

ビューの基になっている表（ベース表）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 構文のリストとして 1000 以内の式を指定できます。

リモート表およびビューを参照するビューを作成する場合、指定するデータベース・リンクを CREATE DATABASE LINK 文の CONNECT TO 句を使用して作成する必要があります。また、ビュー問合せのスキーマ名で指定する必要があります。

ビュー問合せの制限事項：

- ビューの問合せは、CURRVAL および NEXTVAL 疑似列を選択できません。
- ビューの問合せが ROWID、ROWNUM または LEVEL の各疑似列を選択する場合、そのビューの問合せには列の別名が必要です。
- ビューの問合せがアスタリスク（*）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、そのビューにそれらの列は含まれません。
- オブジェクト・ビューの場合、ビュー副問合せの 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。

前述の制限は、マテリアライズド・ビューにも適用されます。

- ビューを固有の特性として更新可能にする場合、次の構造体を指定しないでください。
  - 集合演算子
  - DISTINCT 演算子
  - 集計グループ関数
  - GROUP BY、ORDER BY、CONNECT BY または START WITH 句
  - SELECT リストのコレクション式
  - SELECT リストの副問合せ
  - 結合（一部の例外を除く）

    詳細は、『Oracle8*i* 管理者ガイド』を参照してください。
- 固有の特性として更新可能なビューに疑似列または式が含まれる場合、UPDATE 文はこれらの疑似文および式を参照できません。

- 結合ビューを変更する場合、次の条件が真である必要があります。
  - DML 文は、結合の基になる 1 つの表のみに影響する。
  - UPDATE 文の場合、更新されるすべての列がキー保存表から抽出される。ビューが CHECK OPTION 付きの場合、結合列、およびビュー内で 2 回以上参照される表の列は UPDATE できない。
  - DELETE 文の場合、結合内のキー保存表は 1 つのみである。この表は、ビューに CHECK OPTION を指定しないと、結合内に 2 回以上参照される。
  - INSERT 文の場合、値を挿入するすべての列はキー保存表の列であり、ビューには CHECK OPTION を指定できない。

    参照：

    - 更新可能なビューの詳細は、『Oracle8*i* 管理者ガイド』を参照してください。
    - オブジェクト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・ビューの更新の詳細は、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

with_clause

副問合せを次のように制限する場合に、*with_clause* を使用します。

WITH READ ONLY
: ビューを介して削除、挿入または更新を行わない場合は、WITH READ ONLY を指定します。

WITH CHECK OPTION
: WITH CHECK OPTION を指定すると、ビューを介して実行される挿入および更新処理の結果が、ビュー問合せによって選択できる行であることを保証します。次の場合、CHECK OPTION ではこの条件を保証できません。

  - ビューの問合せまたはビューの基になるビューの問合せ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。

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。

例

基本ビューの例　次の文は、emp 表から dept20 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 dept20
    AS SELECT ename, sal*12 annual_salary
        FROM emp
        WHERE deptno = 20;
```

この場合、副問合せで式 sal*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。

変更可能なビューの例　次の文は、emp 表内の事務員全員の更新可能なビュー clerk を作成します。このビューの中では、これらの従業員の ID、名前および部門番号のみが表示されます。また、これらの列は、事務員として識別される行でのみ更新できます。

```
CREATE VIEW clerk (id_number, person, department, position)
    AS SELECT empno, ename, deptno, job
       FROM emp
       WHERE job = 'CLERK'
    WITH CHECK OPTION CONSTRAINT wco;
```

CHECK OPTION により、新しい従業員が事務員でない場合には clerk に新しい行を挿入できません。

結合ビューの例　結合ビューは、結合を含むビューの副問合せです。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 つのベース表を変更できます。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、次のように割り当てることができます。

```
CREATE VIEW ed AS
   SELECT e.empno, e.ename, d.deptno, d.loc
      FROM emp e, dept d
   WHERE e.deptno = d.deptno

View created.

SELECT column_name, updatable
  FROM user_updatable_columns
  WHERE table_name = 'ED';

COLUMN_NAME     UPDATABLE
--------------- ---
ENAME           YES
DEPTNO          NO
EMPNO           YES
```

```
LOC             NO

INSERT INTO ed (ENAME, EMPNO) values ('BROWN', 1234);
```

この例では、dept 表の deptno 列に一意索引があります。emp 表にマップするビュー内のすべての列が更新可能とマークされていて、emp の主キーがビューに含まれているため、emp ベース表に対して、行の挿入、更新または削除ができます。

注意： NOT NULL 列に DEFAULT 値を指定していない場合は、ビューに、結合内のすべての表のすべての NOT NULL 列が含まれない限り、ビューを使用して表に挿入することはできません。

参照： 結合ビューの更新の詳細は、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

読取り専用ビューの例　次の文は、従業員表 emp 内の事務員全員の読取り専用のビュー clerk を作成します。このビューには、従業員の ID、名前、部門番号および職種のみが表示されます。

```
CREATE VIEW clerk (id_number, person, department, position)
     AS SELECT empno, ename, deptno, job
       FROM emp
       WHERE job = 'CLERK'
    WITH READ ONLY;
```

オブジェクト・ビューの例　次の文は、employee_type のオブジェクト・ビュー emp_object_view を作成します。

```
CREATE TYPE employee_type AS OBJECT
  ( empno       NUMBER(4),
    ename       VARCHAR2(20),
    job         VARCHAR2(9),
    mgr         NUMBER(4),
    hiredate    DATE,
    sal         NUMBER(7,2),
    comm        NUMBER(7,2)  );

CREATE OR REPLACE VIEW emp_object_view OF employee_type
  WITH OBJECT IDENTIFIER (empno)
  AS SELECT empno, ename, job, mgr, hiredate, sal, comm
     FROM emp;
```

DELETE

用途

DELETE は、表、パーティション表、ビューのベース表、またはビューのベース表パーティションから行を削除する場合に使用します。

前提条件

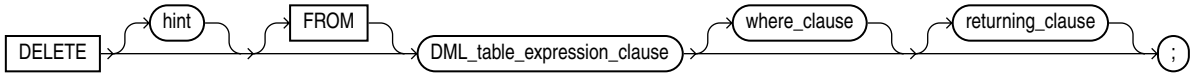
表から行を削除する場合、その表が自スキーマ内に存在している必要があります。存在していない場合は、その表の DELETE 権限が必要です。

ビューのベース表から行を削除する場合、そのビューを含むスキーマの所有者には、そのベース表の DELETE 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、そのビューの DELETE 権限が必要です。

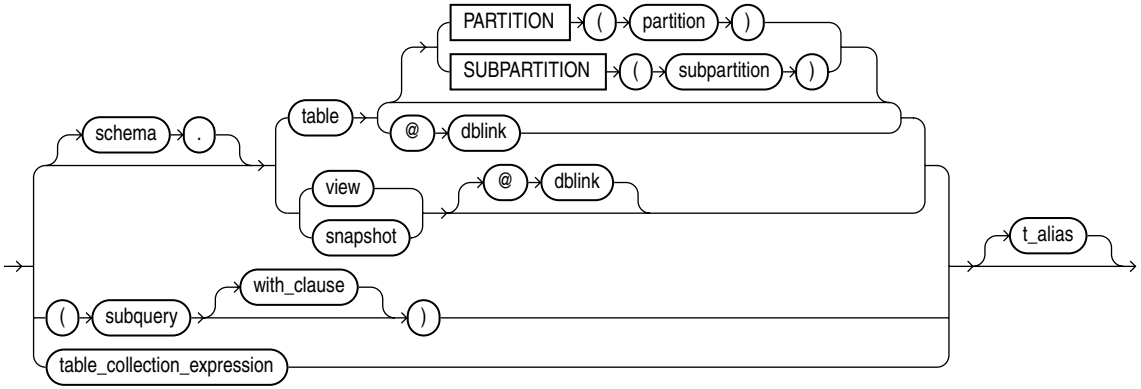
DELETE ANY TABLE システム権限があれば、任意の表、表パーティションまたはビューのベース表から行を削除できます。

SQL92_SECURITY 初期化パラメータが true に設定されている場合、表の列（*where_clause* の列など）を参照する DELETE を実行するには、その表の SELECT 権限が必要です。

構文

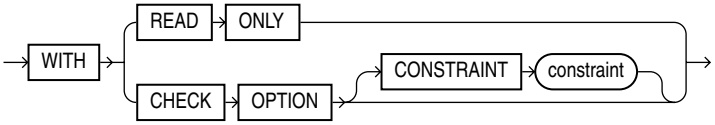


DML_table_expression_clause::=

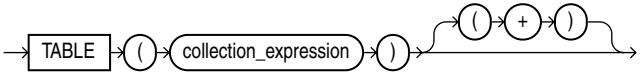


subquery: 11-88 ページの「SELECT および副問合せ」を参照してください。

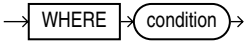
with_clause::=



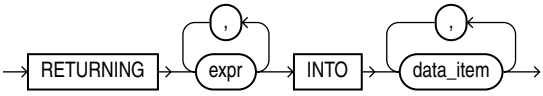
table_collection_expression::=



where_clause::=



returning_clause::=



キーワードとパラメータ

hint

文に対して実行計画を選択する際の、オプティマイザへ指示を渡す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、2-65 ページの「ヒント」および『Oracle8*i* パフォーマンスのための設計およびチューニング』を参照してください。

DML_table_expression_clause

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、表またはビューが自スキーマにあるとみなされます。

table | *view* | *snapshot* | *subquery*

行が削除される表、ビュー、列または副問合せから生成される列の名前を指定します。*view* を指定した場合、ビューのベース表から行が削除されます。

表（またはビューのベース表）が 1 つ以上のドメイン・インデックス列を含む場合、この文は、適切な索引タイプ削除ルーチンを実行します。

参照： これらのルーチンの詳細は、『Oracle8*i* データ・カートリッジ開発者ガイド』を参照してください。

PARTITION (*partition_name*) および SUBPARTITION (*subpartition_name*)

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。

更新対象の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を指定します。

パーティション表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。

dblink

表またはビューが存在しているリモート・データベースとのデータベース・リンクの完全名または部分名を指定します。Oracle 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、2-88 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。

dblink を省略した場合、その表またはビューはローカル・データベースにあるとみなされます。

with_clause

副問合せを次のように制限する場合に、*with_clause* を使用します。

- WITH READ ONLY によって、副問合せを更新禁止にすることを指定します。
- WITH CHECK OPTION によって、副問合せに存在しない行を生成する表変更の禁止を指定します。

  > **参照 :** 11-108 ページの「WITH CHECK OPTION の例」を参照してください。

table_collection_expression

table_collection_expression によって、コレクション値の式を表として扱う必要がある場合に Oracle に通知します。*table_collection_expression* を使用して他の表にもある行のみを削除します。

collection_expression には、ネストした表の列を表またはビューから選択する副問合せを指定します。

> **注意 :** 以前のリリースの Oracle では、*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されていません。

***DML_table_expression_clause* の制限事項 :**

- *table*（または *view* のベース表）に、LOADING または FAILED とマークされたドメイン・インデックスがある場合は、この文を実行できません。
- *DML_query_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、ビューからの削除はできません。
  - 集合演算子
  - DISTINCT 演算子
  - 集計グループ関数
  - GROUP BY、ORDER BY、CONNECT BY または START WITH 句
  - SELECT リストのコレクション式
  - SELECT リストの副問合せ
  - 結合（一部の例外を除く）

    詳細は、『Oracle8*i* 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 パラメータが true に設定されていない限り、DELETE 文は正常に実行されません。

  参照： 7-101 ページの「ALTER SESSION」を参照してください。

where_clause

where_clause を使用すると、条件を満たす行のみが削除されます。この条件は、表を参照したり、副問合せを含むことができます。Oracle 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・ビューから行を削除できます。

参照： *condition* の構文については、5-15 ページの「条件」を参照してください。

注意： 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限り、DELETE はパラレルで実行されます。ただし、*DML_query_expression_clause* の副問合せがリモート・オブジェクトを参照する場合、UPDATE 操作はシリアルで実行されます。

参照： 詳細は、10-40 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、その表またはビューはローカル・データベースにあるとみなされます。

where_clause を省略した場合、表またはビュー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、副問合せまたはコレクション値の**別名**です。通常、別名は相関問合せを持つ DELETE 文で使用されます。

> **注意：** *DML_query_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。

returning_clause

DML（INSERT、UPDATE または DELETE）文に影響される行を取り出します。この句は、表、スナップショット、および単一のベース表を持つビューに指定できます。

- 単一行で処理する場合は、*returning_clause* 付きの DML 文で、処理された行 ROWID および REF を使用して列式を取り出し、ホスト変数または PL/SQL 変数に格納します。
- 複数行で処理する場合は、*returning_clause* 付きの DML 文で、式の値、ROWID および処理された行に関連する REF をバインド配列に格納します。

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 句は、変更された行の値を、*data_item* リストに指定した変数に格納することを示します。

data_item 各 *data_item* は、取り出された *expr* 値を格納するホスト変数または PL/SQL 変数です。

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。

制限事項：

- パラレル DML またはリモート・オブジェクトでは、この句を使用できません。
- この句で 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は、この句を指定できません。

> **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、『Oracle8*i* PL/SQL 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。

注意： 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。

例

基本的な例　次の文は、temp_assign 表からすべての行を削除します。

```
DELETE FROM temp_assign;
```

次の文は、先月の歩合が 100 ドル未満のすべての営業担当者を emp 表から削除します。

```
DELETE FROM emp
    WHERE JOB = 'SALESMAN'
    AND COMM < 100;
```

次の文は、前述の文と同じ結果にはなりますが、副問合せを使用します。

```
DELETE FROM (select * from emp)
    WHERE JOB = 'SALESMAN'
    AND COMM < 100;
```

リモート・データベースの例　次の文は、データベース・リンク dallas によってアクセス可能なデータベースの、ユーザー blake が所有する accounts 表からすべての行を削除します。

```
DELETE FROM blake.accounts@dallas;
```

ネストした表の例　次の文は、ネストした表 projs の行のうち、部門番号が 123 か 456 のいずれかであるもの、または部門の予算が 456.78 を超えるものを削除します。

```
DELETE THE(SELECT projs
           FROM dept d WHERE d.dno = 123) AS p
   WHERE p.pno IN (123, 456) OR p.budgets > 456.78;
```

パーティションの例　次の文は、sales 表のパーティション nov98 から行を削除します。

```
DELETE FROM sales PARTITION (nov98)
   WHERE amount_of_sale != 0;
```

RETURNING 句の例　次の例は、削除された行から sal 列を戻し、その結果をバインド配列 :1 に格納します。

```
DELETE FROM emp
   WHERE job = 'SALESMAN' AND COMM < 100
   RETURNING sal INTO :1;
```

DISASSOCIATE STATISTICS

用途

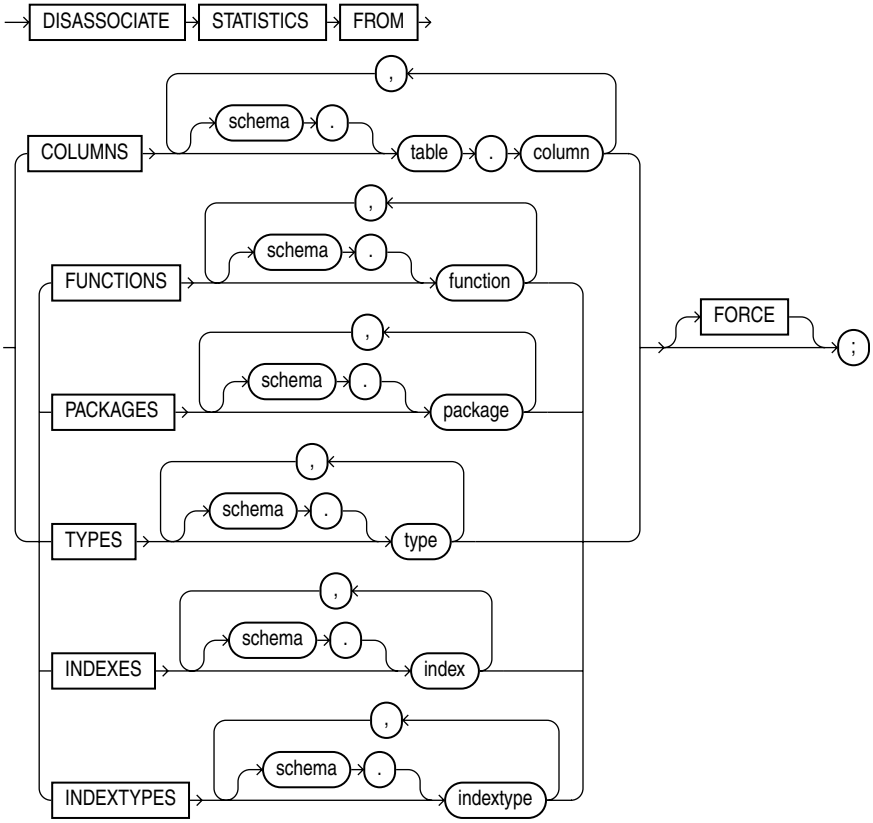
DISASSOCIATE STATISTICS 文は、統計タイプ（またはデフォルト統計）と、列、スタンドアロン・ファンクション、パッケージ、型、ドメイン・インデックスまたは索引タイプとの関連付けを解除する場合に使用します。

> **参照：** 統計タイプの関連付けの詳細は、8-108 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、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・インデックスまたは索引タイプ）を変更する適切な権限が必要です。

構文



キーワードとパラメータ

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 で指定しない場合、オブジェクトは自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を収集する場合、FORCE を指定しない限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での統計の有無にかかわらず、関連付けを解除します。統計が存在する場合、関連付けが解除される前に統計は削除されます。

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 オプションが指定されている統計タイプの関連付けを自動的に解除し、統計タイプで収集したすべての統計を削除します。

例

統計との関連付けの解除例 次の文は、hr スキーマの pack パッケージと統計との関連付けを解除します。

```
DISASSOCIATE STATISTICS FROM PACKAGES hr.pack;
```

DROP CLUSTER

用途

DROP CLUSTER 句は、データベースのクラスタを削除する場合に指定します。

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. 既存のクラスタ化表に対して付与された権限は適用されないため、新しく作成した非クラスタ化表に権限を付与します。

> **参照：** この手順については、10-7 ページの「CREATE TABLE」、11-7 ページの「DROP TABLE」、11-71 ページの「RENAME」および 11-31 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スキーマ内にあるか、または DROP ANY CLUSTER システム権限が必要です。

構文

キーワードとパラメータ

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、そのクラスタは自スキーマ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が削除されます。

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にあるすべての参照整合性制約が削除されます。このような参照整合性制約がある場合にこの句の指定を省略した場合、エラー・メッセージが表示され、クラスタは削除されません。

例

DROP CLUSTER の例　次の文は、クラスタ geography、その表すべて、およびその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します。

```
DROP CLUSTER geography
    INCLUDING TABLES
    CASCADE CONSTRAINTS;
```

DROP CONTEXT

用途

DROP CONTEXT 文は、データベースからコンテキスト・ネームスペースを削除する場合に使用します。

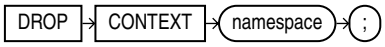
注意： コンテキスト・ネームスペースを削除した場合でも、ユーザー・セッションに設定されたネームスペースのコンテキストは無効になりません。ただし、次にユーザーがそのコンテキストを設定しようとした場合、そのコンテキストは無効になります。

参照： コンテキストの詳細は、9-13 ページの「CREATE CONTEXT」および『Oracle8*i* 概要』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



キーワードとパラメータ

namespace

削除するコンテキスト・ネームスペースを指定します。組込みネームスペース USERENV は削除できません。

例

DROP CONTEXT の例　次の文は、9-13 ページの「CREATE CONTEXT」で作成したコンテキストを削除します。

```
DROP CONTEXT hr_context;
```

DROP DATABASE LINK

用途

DROP DATABASE LINK 文は、データベースからデータベース・リンクを削除する場合に使用します。

> **参照：** データベース・リンクの作成方法については、9-28 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・リンクを削除する場合は、データベース・リンクが自スキーマ内にある必要があります。PUBLIC 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

キーワードとパラメータ

PUBLIC

PUBLIC データベース・リンクを削除する場合は、PUBLIC を指定する必要があります。

dblink

削除するデータベース・リンクの名前を指定します。

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、スキーマ名で *dblink* を修飾することはできません。これは、データベース・リンクの名前で、ピリオドが解析されるためです。そのため、たとえば、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・リンクであると解析されるため、名前全体が自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、boston というプライベート・データベース・リンクを削除します。

```
DROP DATABASE LINK boston;
```

DROP DIMENSION

用途

DROP DIMENSION は、名前付けしたディメンションを削除する場合に使用します。

注意： 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、問合せのリライトによって再書込み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。

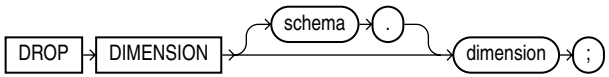
参照：

- ディメンションの作成については、9-34 ページの「CREATE DIMENSION」を参照してください。
- ディメンションの変更については、7-33 ページの「ALTER DIMENSION」を参照してください。
- 『Oracle8*i* 概要』も参照してください。

前提条件

この文を使用する場合、ディメンションが自スキーマ内に設定されているか、または DROP ANY DIMENSION システム権限が必要です。

構文



キーワードとパラメータ

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、そのディメンションは自スキーマ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

DROP DIMENSION の例　次の文は、time ディメンションを削除します。

```
DROP DIMENSION time;
```

DROP DIRECTORY

用途

DROP DIRECTORY 文は、データベースからディレクトリ・オブジェクトを削除する場合に使用します。

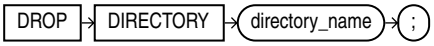
> **参照：** ディレクトリの作成については、9-39 ページの「CREATE DIRECTORY」を参照してください。

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



キーワードとパラメータ

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 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

DROP DIRECTORY の例 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;
```

DROP FUNCTION

用途

DROP FUNCTION 文は、データベースからスタンドアロンのストアド・ファンクションを削除する場合に使用します。

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。DROP PACKAGE 文を使用して完全なパッケージを削除するか、OR REPLACE 句付きの CREATE PACKAGE 文を使用するファンクションを含まないパッケージを再定義します。

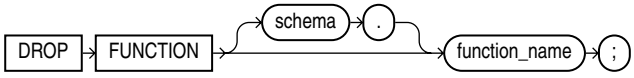
参照：

- ファンクションの作成については、9-42 ページの「CREATE FUNCTION」を参照してください。
- ファンクションの変更については、7-36 ページの「ALTER FUNCTION」を参照してください。

前提条件

削除するファンクションが自スキーマ内にあるか、または DROP ANY PROCEDURE システム権限が必要です。

構文



キーワードとパラメータ

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、ファンクションが自スキーマ内に定義されているものとみなされます。

function_name

削除するファンクションの名前を指定します。

Oracle 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。後でこのようなオブジェクトを参照した場合、Oracle 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限り、エラーが戻されます。

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が削除されます。

参照：

- リモート・オブジェクトなどのスキーマ・オブジェクト間の依存性を Oracle が管理する方法の詳細は、『Oracle8*i* 概要』を参照してください。
- 統計タイプの関連の詳細は、8-108 ページの「ASSOCIATE STATISTICS」および 10-121 ページの「DISASSOCIATE STATISTICS」を参照してください。

例

DROP FUNCTION の例　次の文は、スキーマ riddley 内のファンクション new_acct を削除します。この場合、new_acct 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 riddley.new_acct;
```

DROP INDEX

用途

DROP INDEX 文は、データベースから索引またはドメイン・インデックスを削除する場合に使用します。

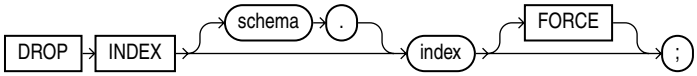
参照 :

- 索引の作成については、9-51 ページの「CREATE INDEX」を参照してください。
- 索引の変更については、7-38 ページの「ALTER INDEX」を参照してください。
- ドメイン・インデックスの詳細は、9-51 ページの「CREATE INDEX」の *domain_index_clause* を参照してください。

前提条件

索引が自スキーマ内にあるか、または DROP ANY INDEX システム権限が必要です。

構文



キーワードとパラメータ

schema

削除する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、索引が自スキーマ内に定義されているものとみなされます。

index

削除する索引の名前を指定します。索引を削除した場合、その索引に割り当てられていたすべてのデータ・ブロックが索引の表領域に戻されます。

ドメイン・インデックスを削除した場合、Oracle によって次の処理が実行されます。

- 適切な索引タイプの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、『Oracle8*i* データ・カートリッジ開発者ガイド』を参照してください。
- 統計タイプがドメイン・インデックス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が削除されます。

> **参照：** 統計タイプの関連の詳細は、8-108 ページの「ASSOCIATE STATISTICS」および 10-121 ページの「DISASSOCIATE STATISTICS」を参照してください。

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した場合、すべての索引パーティションが同様に削除されます。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が同様に削除されます。

FORCE

FORCE は、ドメイン・インデックスのみに適用されます。この句は、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 LOADING のマークが付けられている場合でも、ドメイン・インデックスを削除します。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 LOADING マークが付けられている場合にドメイン・インデックスを削除できません。

例

DROP INDEX の例　次の文は、monolith という名前の索引を削除します。

```
DROP INDEX monolith;
```

DROP INDEXTYPE

用途

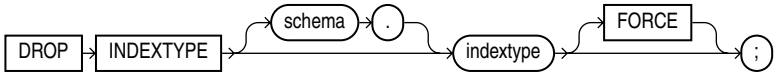
DROP INDEXTYPE 文は、索引タイプを削除する場合に使用します。同時に、統計タイプとの関連付けも解除します。

> **参照：** 索引タイプの詳細は、9-75 ページの「CREATE INDEXTYPE」を参照してください。

前提条件

索引タイプが自スキーマ内にあるか、または DROP ANY INDEXTYPE システム権限が必要です。

構文



キーワードとパラメータ

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、索引タイプが自スキーマ内に定義されているものとみなされます。

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されたすべての統計が削除されます。

> **参照：** 統計情報の関連付けの詳細は、8-108 ページの「ASSOCIATE STATISTICS」および 10-121 ページの「DISASSOCIATE STATISTICS」を参照してください。

FORCE

FORCE を指定すると、索引タイプが 1 つ以上のドメイン・インデックスから現在参照されている状態でも、索引タイプを削除します。また、それらのドメイン・インデックスに INVALID のマークを付けます。FORCE を使用しない場合、任意のドメイン・インデックスが索引タイプを参照していると索引タイプを削除できません。

例

DROP INDEXTYPE の例　次の文は、索引タイプ textindextype 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・インデックスに INVALID のマークを付けます。

```
DROP INDEXTYPE textindextype FORCE;
```

DROP JAVA

用途

DROP JAVA 文は、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する場合に使用します。

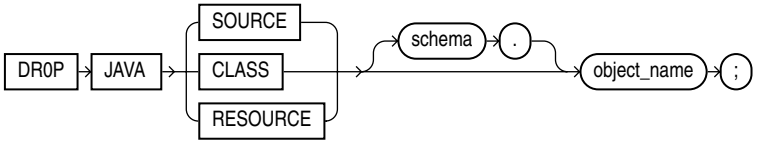
参照 :

- Java オブジェクトの作成については、9-78 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、『Oracle8*i* Java ストアド・プロシージャ開発者ガイド』を参照してください。

前提条件

Java ソース、クラスまたはリソースが自スキーマ内にあるか、または 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 オブジェクト権限が必要です。

構文



キーワードとパラメータ

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・オブジェクトが削除されます。

JAVA CLASS

CLASS を指定すると、Java クラス・スキーマ・オブジェクトが削除されます。

JAVA RESOURCE

RESOURCE を指定すると、Java リソース・スキーマ・オブジェクトが削除されます。

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

DROP JAVA CLASS の例　次の文は、Java クラス `MyClass` を削除します。

```
DROP JAVA CLASS "MyClass";
```

DROP LIBRARY

用途

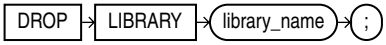
DROP LIBRARY 文は、データベースから外部プロシージャ・ライブラリを削除する場合に使用します。

> **参照：** ライブラリの作成については、9-84 ページの「CREATE LIBRARY」を参照してください。

前提条件

DROP LIBRARY システム権限が必要です。

構文



キーワードとパラメータ

library_name
削除対象の外部プロシージャ・ライブラリの名前を指定します。

例

DROP LIBRARY の例　次の文は、ext_procs ライブラリを削除します。

```
DROP LIBRARY ext_procs;
```

DROP MATERIALIZED VIEW

用途

DROP MATERIALIZED VIEW 文は、データベースから既存のマテリアライズド・ビューを削除する場合に使用します。

「スナップショット」と「マテリアライズド・ビュー」は同義語です。

参照：

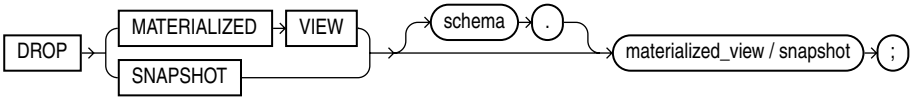
- マテリアライズド・ビューの種類については、9-86 ページの「CREATE MATERIALIZED VIEW」を参照してください。
- マテリアライズド・ビューの変更については、7-59 ページの「ALTER MATERIALIZED VIEW」を参照してください。
- レプリケーション環境でのマテリアライズド・ビューについては、『Oracle8*i* レプリケーション・ガイド』を参照してください。
- データ・ウェアハウス環境でのマテリアライズド・ビューについては、『Oracle8*i* データ・ウェアハウス』を参照してください。

前提条件

マテリアライズド・ビューが自スキーマ内にあるか、または DROP ANY MATERIALIZED VIEW（または DROP ANY SNAPSHOT）システム権限が必要です。また、Oracle 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、11-7 ページの「DROP TABLE」、11-21 ページの「DROP VIEW」および 10-133 ページの「DROP INDEX」を参照してください。

構文



キーワードとパラメータ

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、ディテール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。
- ディテール表を削除した場合、Oracle は、その表に基づくマテリアライズド・ビューを自動的には削除しません。ただし、削除したディテール表を基礎とするマテリアライズド・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再書込み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。事前にマテリアライズド・ビューが作成されている場合、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・メカニズムによる管理ができなくなります。

例

DROP MATERIALIZED VIEW の例　次の文は、ユーザー hq が所有するマテリアライズド・ビュー parts を削除します。

```
DROP SNAPSHOT hq.parts;
```

次の文は、マテリアライズド・ビュー sales_by_month およびマテリアライズド・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 句で CREATE MATERIALIZED VIEW 文に登録されていない場合）。

```
DROP MATERIALIZED VIEW sales_by_month;
```

DROP MATERIALIZED VIEW LOG

用途

DROP MATERIALIZED VIEW LOG 文は、データベースからマテリアライズド・ビュー・ログを削除する場合に使用します。

「スナップショット」と「マテリアライズド・ビュー」は同義語です。

参照：

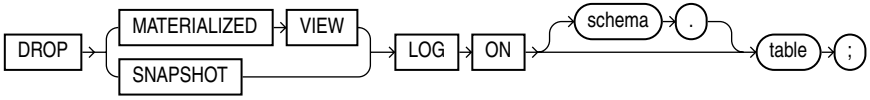
- マテリアライズド・ビューについては、9-86 ページの「CREATE MATERIALIZED VIEW」および7-59 ページの「ALTER MATERIALIZED VIEW」を参照してください。
- マテリアライズド・ビュー・ログについては、9-104 ページの「CREATE MATERIALIZED VIEW LOG」を参照してください。
- レプリケーション環境でのマテリアライズド・ビューについては、『Oracle8*i* レプリケーション・ガイド』を参照してください。
- データ・ウェアハウス環境でのマテリアライズド・ビューについては、『Oracle8*i* データ・ウェアハウス』を参照してください。

前提条件

マテリアライズド・ビュー・ログは、表とトリガーで構成されます。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： 11-7 ページの「DROP TABLE」を参照してください。

構文



キーワードとパラメータ

schema

削除するマテリアライズド・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ディテール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・ログのディテール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。これらのマテリアライズド・ビューは、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ビューを含みます。

> **参照：** マテリアライズド・ビューの種類については、『Oracle8*i* データ・ウェアハウス』を参照してください。

例

DROP MATERIALIZED VIEW LOG の例　次の文は、parts 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 parts;
```

DROP OPERATOR

用途

DROP OPERATOR 文は、ユーザー定義演算子を削除する場合に使用します。

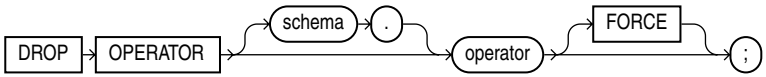
参照：

- 演算子の作成については、9-112 ページの「CREATE OPERATOR」を参照してください。
- 演算子の詳細は、3-16 ページの「ユーザー定義演算子」および『Oracle8*i* 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

演算子が自スキーマ内にあるか、または DROP ANY OPERATOR システム権限が必要です。

構文



キーワードとパラメータ

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、その演算子が自スキーマ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、その演算子を削除し、演算子に依存するオブジェクトに INVALID のマークが付きます。FORCE を使用しないと、任意のスキーマ・オブジェクトが演算子を参照しているときに演算子を削除できません。

例

DROP OPERATOR の例　次の文は、演算子 merge を削除します。

```
DROP OPERATOR ordsys.merge;
```

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

DROP OUTLINE 文は、ストアド・アウトラインを削除する場合に使用します。

参照：

- アウトラインの作成については、9-116 ページの「CREATE OUTLINE」を参照してください。
- アウトラインの詳細は、『Oracle8*i* パフォーマンスのための設計およびチューニング』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

キーワードとパラメータ

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

DROP OUTLINE の例　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

DROP PACKAGE 文は、データベースからストアド・パッケージを削除する場合に使用します。この文は、パッケージ本体および仕様部を削除します。

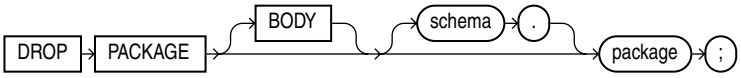
注意： この文を使用してパッケージからシングル・オブジェクトを削除しないでください。かわりに、CREATE PACKAGE 文および CREATE PACKAGE BODY 文に OR REPLACE 句を指定して、そのオブジェクトがないパッケージを再作成してください。

参照： 9-118 ページの「CREATE PACKAGE」を参照してください。

前提条件

削除するパッケージが自スキーマ内にあるか、または DROP ANY PROCEDURE システム権限が必要です。

構文



キーワードとパラメータ

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限り、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、パッケージが自スキーマ内に定義されているとみなされます。

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。後でこのようなオブジェクトのいずれかを参照した場合、Oracle が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限り、エラーが戻されます。

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が削除されます。

参照：

- リモート・オブジェクトなどのスキーマ・オブジェクト間の依存性を Oracle が管理する方法については、『Oracle8*i* 概要』を参照してください。
- 8-108 ページの「ASSOCIATE STATISTICS」および 10-121 ページの「DISASSOCIATE STATISTICS」も参照してください。

例

DROP PACKAGE の例　次の文は、banking 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。

```
DROP PACKAGE banking;
```

DROP PROCEDURE

用途

DROP PROCEDURE 文は、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する場合に使用します。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。DROP PACKAGE 文を使用して完全にパッケージを削除するか、OR REPLACE 句付きの CREATE PACKAGE 文を使用して、プロシージャを含まないパッケージを再定義します。

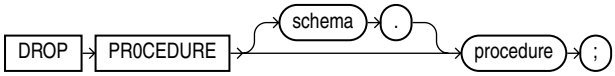
参照：

- プロシージャの作成については、9-127 ページの「CREATE PROCEDURE」を参照してください。
- プロシージャの変更については、7-84 ページの「ALTER PROCEDURE」を参照してください。

前提条件

削除するプロシージャが自スキーマ内にあるか、または DROP ANY PROCEDURE システム権限が必要です。

構文



キーワードとパラメータ

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、プロシージャが自スキーマ内に定義されているとみなされます。

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。後でこのようなオブジェクトのいずれかを参照した場合、Oracle が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限り、エラー・メッセージが戻されます。

> **参照：** リモート・オブジェクトなどのスキーマ・オブジェクト間の依存性を Oracle が管理する方法については、『Oracle8*i* 概要』を参照してください。

例

DROP PROCEDURE の例　次の文は、ユーザー `kerner` が所有するプロシージャ `transfer` を削除し、`transfer` に依存するすべてのオブジェクトを無効にします。

```
DROP PROCEDURE kerner.transfer
```

DROP PROFILE

用途

DROP PROFILE 文は、データベースからプロファイルを削除する場合に使用します。

参照 :

- プロファイルの作成については、9-134 ページの「CREATE PROFILE」を参照してください。
- プロファイルの変更については、7-87 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

キーワードとパラメータ

profile

削除するプロファイルの名前を指定します。

制限事項 : DEFAULT プロファイルは削除できません。

CASCADE

CASCADE を指定すると、削除するプロファイルが割り当てられているすべてのユーザーから、そのプロファイルの割当てが解除されます。このようなユーザーに対しては、DEFAULT プロファイルが自動的に割り当てられます。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

DROP PROFILE の例　次の文は、engineer プロファイルを削除します。

```
DROP PROFILE engineer CASCADE;
```

現在、engineer プロファイルが割り当てられているユーザーについては、engineer プロファイルが削除され、DEFAULT プロファイルが割り当てられます。

DROP ROLE

用途

DROP ROLE 文は、データベースからロールを削除する場合に使用します。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。

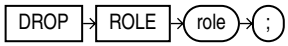
参照 :

- ロールの作成については、9-141 ページの「CREATE ROLE」参照してください。
- ロールを使用可能にするときに必要な認証の変更については、7-94 ページの「ALTER ROLE」を参照してください。
- 現行セッションに対するロールの使用禁止については、11-122 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 システム権限が必要です。

構文



キーワードとパラメータ

role

削除するロールの名前を指定します。

例

DROP ROLE の例　次の文は、ロール florist を削除します。

```
DROP ROLE florist;
```

DROP ROLLBACK SEGMENT

用途

DROP ROLLBACK SEGMENT 文は、データベースからロールバック・セグメントを削除する場合に使用します。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。

参照：

- ロールバック・セグメントの作成については、9-144 ページの「CREATE ROLLBACK SEGMENT」を参照してください。
- ロールバック・セグメントの変更については、7-96 ページの「ALTER ROLLBACK SEGMENT」を参照してください。
- 10-56 ページの「CREATE TABLESPACE」も参照してください。

前提条件

DROP ROLLBACK SEGMENT システム権限が必要です。

構文

キーワードとパラメータ

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。

制限事項：

- ロールバック・セグメントが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、データ・ディクショナリ・ビュー DBA_ROLLBACK_SEGS に問い合せます。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

DROP ROLLBACK SEGMENT の例　次の文は、ロールバック・セグメント accounting を削除します。

```
DROP ROLLBACK SEGMENT accounting;
```

11

SQL 文 : DROP SEQUENCE ～ UPDATE

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
- filespec
- GRANT
- INSERT
- LOCK TABLE
- NOAUDIT
- RENAME
- REVOKE
- ROLLBACK

- SAVEPOINT
- SELECT および副問合せ
- SET CONSTRAINT[S]
- SET ROLE
- SET TRANSACTION
- storage_clause
- TRUNCATE
- UPDATE

DROP SEQUENCE

用途

DROP SEQUENCE は、データベースの順序を削除する場合に使用します。

この文を使用した場合、順序の削除および再作成を行って、順序を再開できます。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。

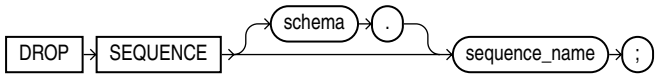
参照：

- 順序の作成の詳細は、9-149 ページの「CREATE SEQUENCE」を参照してください。
- 順序の変更の詳細は、7-99 ページの「ALTER SEQUENCE」を参照してください。

前提条件

削除する順序が自スキーマ内にあるか、または DROP ANY SEQUENCE システム権限が必要です。

構文



キーワードとパラメータ

schema

順序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、順序が自スキーマ内に定義されているとみなされます。

sequence_name

削除する順序の名前を指定します。

例

DROP SEQUENCE の例　次の文は、ユーザー elly が所有する順序 ESEQ を削除します。この文を発行する場合は、ユーザー elly 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 elly.eseq;
```

DROP SYNONYM

用途

DROP SYNONYM 文は、データベースからシノニムを削除する場合に使用します。または、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します。

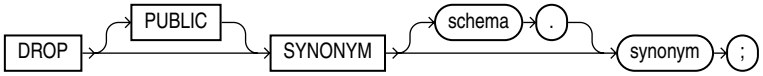
> **参照：** シノニムの詳細は、10-3 ページの「CREATE SYNONYM」を参照してください。

前提条件

プライベート・シノニムを削除する場合は、そのシノニムが自スキーマ内にあるか、または DROP ANY SYNONYM システム権限が必要です。

PUBLIC 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



キーワードとパラメータ

PUBLIC

パブリック・シノニムを削除する場合は、PUBLIC を指定する必要があります。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* の指定を省略した場合、シノニムが自スキーマ内に定義されているとみなされます。

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・ビューのシノニム、マテリアライズド・ビューを含む表またはスナップショット、あるいはマテリアライズド・ビューに依存する表が削除される場合、マテリアライズド・ビューは無効になります。

例

DROP SYNONYM の例　次の文は、シノニム market を削除します。

```
DROP SYNONYM market;
```

DROP TABLE

用途

DROP TABLE 文は、データベースの表またはオブジェクト表、およびそれに含まれるすべてのデータを削除する場合に使用します。

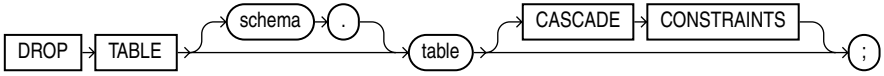
参照：

- 表の作成の詳細は、10-7 ページの「CREATE TABLE」を参照してください。
- 表の変更の詳細は、8-2 ページの「ALTER TABLE」を参照してください。

前提条件

削除する表が自スキーマ内にあるか、または DROP ANY TABLE システム権限が必要です。

構文



キーワードとパラメータ

schema

変更する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、この表が自スキーマに存在するとみなされます。

table

削除する表、オブジェクト表または索引構成表の名前を指定します。Oracle によって次の操作が自動的に実行されます。

- 表上のすべての行が削除されます（行を明示的に削除した場合と同じです）。
- 表のすべての索引およびドメイン・インデックスが、作成者やスキーマの所有者とは関係なく削除されます。

- **レンジ・パーティション表**または**ハッシュ・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。
- **ドメイン・インデックス**の場合は、この文によって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、『Oracle8*i* 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が削除されます。

  参照： 統計タイプの関連の詳細は、8-108 ページの「ASSOCIATE STATISTICS」および 10-121 ページの「DISASSOCIATE STATISTICS」を参照してください。

- 表が**クラスタ**の一部でない場合は、表とその索引に割り当てられていたすべてのデータ・ブロックを、表とその索引が格納されていた表領域に戻します。

  注意： 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。10-124 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・ビューのビューのコンテナ表またはマスター表のベース表である場合、あるいは表がストアド・プロシージャ、ストアド・ファンクションまたはストアド・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。ユーザーが表を再作成するか、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限り、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、パッケージで参照する列と、ビューの定義で使用されていた問合せで選択したすべての列が、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を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・ビューのディテール表**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・ビューの問合せによって選択されるすべての列が含まれる表を再作成しない限り、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**を持つ場合、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・ロード INSERT リフレッシュ情報は削除されます。

CASCADE CONSTRAINTS

CASCADE CONSTRAINTS は、削除した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します。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、表は削除されません。

例

DROP TABLE の例　次の文は、test_data 表を削除します。

```
DROP TABLE test_data;
```

DROP TABLESPACE

用途

DROP TABLESPACE は、データベースの表領域を削除する場合に使用します。

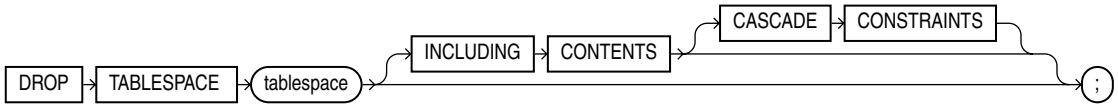
参照：

- 表領域の作成の詳細は、10-56 ページの「CREATE TABLESPACE」を参照してください。
- 表領域の変更の詳細は、8-67 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



キーワードとパラメータ

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。

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を変更することもできます。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。

制限事項 :

- SYSTEM 表領域は削除できません。
- ドメイン・インデックス、またはドメイン・インデックス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を削除できません。

> **参照 :** ドメイン・インデックスの詳細は、『Oracle8*i* データ・カートリッジ開発者ガイド』および『Oracle8*i* 概要』を参照してください。

INCLUDING CONTENTS

INCLUDING CONTENTS は、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します。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。

パーティション表の場合、次の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。

- レンジ・パーティション表またはハッシュ・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

> **注意 :** パーティション表のすべてのパーティションが *tablespace* に常駐する場合、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 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で、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・セグメントも削除します。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・ビューのコンテナ表またはディテール表を含む場合、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 / スナップショット・ログを持つ場合、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・ロード INSERT リフレッシュ情報は削除されます。

CASCADE CONSTRAINTS

CASCADE CONSTRAINTS は、*tablespace* に含まれる表の主キーまたは一意キーを参照する、*tablespace* の外の表にあるすべての参照整合性制約を削除します。このような参照整合性制約がある際にこの句の指定を省略した場合、エラーが戻され、表領域は削除されません。

例

DROP TABLESPACE の例　次の文は、mfrg 表領域とその内容を削除します。

```
DROP TABLESPACE mfrg
    INCLUDING CONTENTS
        CASCADE CONSTRAINTS;
```

DROP TRIGGER

用途

DROP TRIGGER は、データベースからデータベース・トリガーを削除する場合に使用します。

参照：

- トリガーの作成の詳細は、10-66 ページの「CREATE TRIGGER」を参照してください。
- トリガーの使用可能化、使用禁止またはコンパイルの詳細は、8-76 ページの「ALTER TRIGGER」を参照してください。

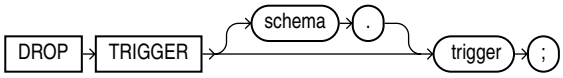
前提条件

削除するトリガーが自スキーマ内にあるか、または 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に DATABASE を作成する場合は、ADMINISTER DATABASE TRIGGER システム権限が必要です。

参照： これらの権限の詳細は、10-66 ページの「CREATE TRIGGER」を参照してください。

構文



キーワードとパラメータ

schema

削除する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、トリガーは自スキーマ内に定義されているとみなされます。

trigger

削除するトリガーの名前を指定します。データベースの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

DROP TRIGGER の例　次の文は、スキーマ ruth 内のトリガー reorder を削除します。

```
DROP TRIGGER ruth.reorder;
```

DROP TYPE

用途

DROP TYPE 文は、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する場合に使用します。

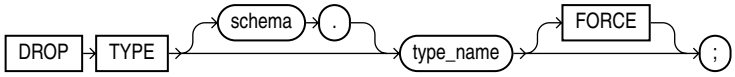
参照：

- オブジェクト型の本体を削除する場合は、11-17 ページの「DROP TYPE BODY」を参照してください。
- 型の作成の詳細は、10-80 ページの「CREATE TYPE」を参照してください。

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スキーマ内にあるか、または DROP ANY TYPE システム権限が必要です。

構文



キーワードとパラメータ

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、その型が自スキーマに存在するとみなされます。

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、8-108 ページの「ASSOCIATE STATISTICS」および 10-121 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。

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：** 9-75 ページの「CREATE INDEXTYPE」を参照してください。

FORCE

FORCE は、依存性オブジェクトがある型でも強制的に削除します。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、リカバリが不可能であり、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。型の依存性の詳細は、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

例

DROP TYPE の例 次の文は、オブジェクト型 person_t を削除します。

```
DROP TYPE person_t;
```

DROP TYPE BODY

用途

DROP TYPE BODY 文は、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する場合に使用します。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。

参照 :

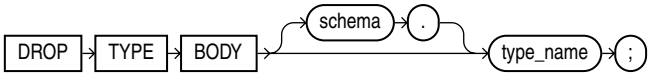
- オブジェクトの本体および仕様部を削除する場合は、11-15 ページの「DROP TYPE」を参照してください。
- 型本体の詳細は、10-93 ページの「CREATE TYPE BODY」参照してください。

前提条件

オブジェクト型の本体が自スキーマ内にある必要があります。また、次の権限が必要です。

- CREATE TYPE または CREATE ANY TYPE システム権限
- DROP ANY TYPE システム権限

構文



キーワードとパラメータ

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、このオブジェクト型が自スキーマに存在するとみなされます。

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。

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

DROP TYPE BODY の例　次の文は、オブジェクト型の本体 rational を削除します。

```
DROP TYPE BODY rational;
```

DROP USER

用途

DROP USER 文は、データベース・ユーザーを削除する場合、およびオプションでユーザーのオブジェクトを削除する場合に使用します。

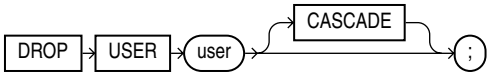
参照 :

- ユーザーの作成の詳細は、10-99 ページの「CREATE USER」を参照してください。
- ユーザー定義の変更の詳細は、8-87 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。

構文



キーワードとパラメータ

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 は、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します。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・インデックスも削除され、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、『Oracle8*i* データ・カートリッジ開発者ガイド』を参照してください。

- Oracle は、**他の**スキーマにある、削除された自スキーマ内のオブジェクトのビューまたはシノニム、および削除された自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、パッケージのオブジェクトを削除せず、無効にします。
- 削除されたユーザー・スキーマ内の表またはビューに対するマテリアライズド・ビューは削除されません。ただし、CASCADE を指定した場合、このようなマテリアライズド・ビューはリフレッシュできなくなります。
- 自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 では、ユーザーが所有するすべての型も FORCE で削除します。詳細は、11-16 ページの「DROP TYPE」にある FORCE キーワードを参照してください。

例

DROP USER の例　ユーザー bradley のスキーマ内にオブジェクトがない場合、次の文を発行すると、bradley を削除できます。

```
DROP USER bradley;
```

bradley 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、bradley とその中のオブジェクトを削除する必要があります。

```
DROP USER bradley CASCADE;
```

DROP VIEW

用途

DROP VIEW 文は、データベースからビューまたはオブジェクト・ビューを削除する場合に使用します。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。

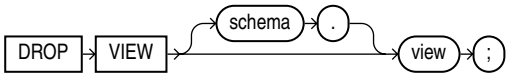
参照：

- ビューの作成の詳細は、10-105 ページの「CREATE VIEW」を参照してください。
- ビューの変更の詳細は、8-93 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スキーマ内にあるか、または DROP ANY VIEW システム権限が必要です。

構文



キーワードとパラメータ

schema

削除する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。

ビューを参照するビュー、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、無効になります。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。

参照：

- 10-7 ページの「CREATE TABLE」および 10-3 ページの「CREATE SYNONYM」を参照してください。
- 無効のマテリアライズド・ビュー・ログの再検証の詳細は、7-59 ページの「ALTER MATERIALIZED VIEW」を参照してください。

例

DROP VIEW の例　次の文は、`view_data` ビューを削除します。

```
DROP VIEW view_data;
```

EXPLAIN PLAN

用途

EXPLAIN PLAN 文は、指定した SQL 文を実行するために Oracle が使用する実行計画を決定する場合に使用します。この文によって、実行計画の各ステップを記述している行が、指定した表に挿入されます。コストベースの最適化を使用している場合、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・インデックス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 ですが、正確な名前および格納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。

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。

参照 :

- EXPLAIN PLAN の出力の詳細は、『Oracle8*i* パフォーマンスのための設計およびチューニング』を参照してください。
- 実行計画の生成、解析方法および SQL トレース機能の使用方法についても、『Oracle8*i* パフォーマンスのための設計およびチューニング』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。

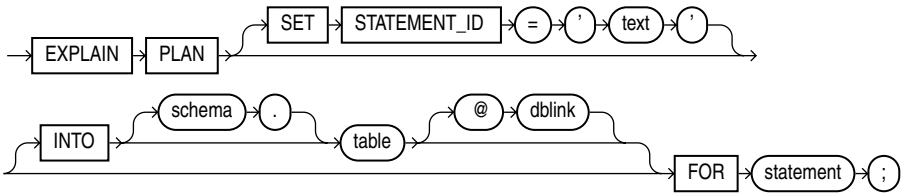
出力する実行計画の適用対象である SQL 文を実行するための権限も必要です。この SQL 文でビューにアクセスする場合は、このビューの基礎になっているすべての表およびビューへのアクセス権限が必要です。このビューが別のビューに基づき、さらにこの別のビューが、ある表に基づいている場合、別のビューとそのビューの基礎になっている表へのアクセス権限が必要です。

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、11-52 ページの「INSERT」および 11-88 ページの「SELECT および副問合せ」を参照してください。

構文



キーワードとパラメータ

SET STATEMENT_ID = '*text*'

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には、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*

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、この表が自スキーマに存在するとみなされます。

dblink には、出力表が格納されているリモートの Oracle データベース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 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* の指定を省略した場合、表がローカル・データベース上にあるとみなされます。

> **参照：** データベース・リンクの参照方法の詳細は、2-88 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スキーマ内にある PLAN_TABLE であるとみなされます。

FOR *statement*

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。

注意：

- *statement* が *parallel_clause* を持つ場合、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 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

EXPLAIN PLAN 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 Chicago' とともに指定した output 表に挿入します。

```
EXPLAIN PLAN
    SET STATEMENT_ID = 'Raise in Chicago'
    INTO output
    FOR UPDATE emp
        SET sal = sal * 1.10
        WHERE deptno =  (SELECT deptno
                  FROM dept
                  WHERE loc = 'CHICAGO');
```

次の SELECT 文は、output 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 output
    START WITH id = 0 AND statement_id = 'Raise in Chicago'
    CONNECT BY PRIOR id = parent_id AND
    statement_id = 'Raise in Chicago';
```

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                                         1
    FILTER                                               0
      TABLE ACCESS    FULL        EMP                    1
      TABLE ACCESS    FULL        DEPT                   2
```

POSITION 列の 1 行目の値は、文のコストが 1 であることを示しています。

EXPLAIN PLAN: パーティション例　stock_num 列に従って 8 つにパーティション化されている stocks 表があり、stock_num 列上にローカルの同一キー索引 stock_ix が作成されているとします。パーティション HIGHVALUES の値は、1000、2000、3000、4000、5000、6000、7000 および 8000 です。

次の問合せを考えてみます。

```
SELECT * FROM stocks WHERE stock_num BETWEEN 3800 AND :h;
```

（ここで、h はバインド変数を指します。）EXPLAIN PLAN は、PLAN_TABLE を出力表とする次の問合せを実行します。この問合せによって、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 id, operation, options, object_name,
  partition_start, partition_stop, partition_id FROM plan_table;
```

filespec

用途

filespec 構文は、ファイルをデータ・ファイルまたはテンポラリ・ファイルとして指定する場合、または 1 つ以上のファイルのグループを REDO ログ・ファイル・グループとして指定する場合に使用します。

前提条件

filespec は、次の文で使用できます。

- CREATE DATABASE
- ALTER DATABASE
- CREATE TABLESPACE
- ALTER TABLESPACE
- CREATE CONTROLFILE
- CREATE LIBRARY
- CREATE TEMPORARY TABLESPACE

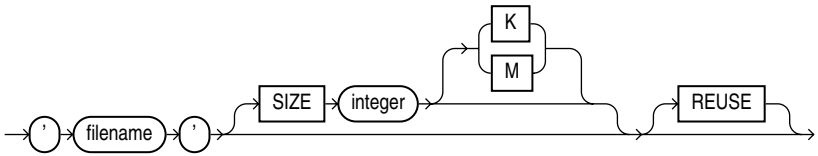
これらの文の発行には、その文の実行権限が必要です。

参照： 詳細は、次の項を参照してください。

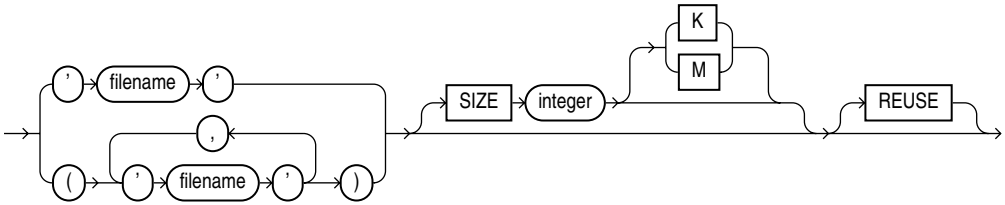
- 9-21 ページの「CREATE DATABASE」
- 7-7 ページの「ALTER DATABASE」
- 10-56 ページの「CREATE TABLESPACE」
- 8-67 ページの「ALTER TABLESPACE」
- 9-15 ページの「CREATE CONTROLFILE」
- 9-84 ページの「CREATE LIBRARY」
- 10-63 ページの「CREATE TEMPORARY TABLESPACE」

構文

filespec_datafiles & filespec_tempfiles::=



filespec_redo_log_file_groups::=



キーワードとパラメータ

'filename'

データ・ファイル、テンポラリ・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。

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。

SIZE *integer*

ファイルのサイズをバイト単位で指定します。K または M を使用して、KB または MB 単位で指定することもできます。

- このパラメータは、ファイルがすでにある場合にのみ省略できます。
- 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。

REUSE

REUSE は、既存ファイルの再使用を可能にします。

- ファイルが存在する場合、そのサイズが SIZE パラメータの値と一致するかどうかが検証されます（SIZE を指定した場合）。
- 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。
- ファイルがすでに作成されていない場合のみ、この句を省略できます。この句を省略した場合、ファイルが作成されます。

注意： 既存のファイルが使用される場合、そのファイルに入っていた内容は失われます。

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 *filespec* は、REDO ログ・ファイル・グループに GROUP1 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 *filespec* は、REDO ログ・ファイル・グループに GROUP2 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 *filespec* 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 *filespec* 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 *filespec* は、REDO ログ・ファイル・グループに GROUP3 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 *filespec* 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。また、そのサイズは 50KB である必要があります。指定したファイルがない場合は、Oracle によって 50KB のファイルが作成されます。

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 'diskc:stock1.dat',
            'diskc:stock2.dat',
            'diskc:stock3.dat';
```

データ・ファイルの *filespecs* 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。各 *filespec* の指定に SIZE パラメータが指定されていないため、各ファイルはすでに存在している必要があります。

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 'diskc:stock4.dat' REUSE;
```

filespec は、'diskc:stock4.dat' という名前のデータ・ファイルを指定します。*filespec* の指定に SIZE パラメータが指定されていないため、このデータ・ファイルはすでに存在している必要があります。また、REUSE 句は無効です。

GRANT

用途

GRANT 文は、次の権限を付与する場合に使用します。

- 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 11-1 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。表 11-2 に、Oracle で事前定義されているロールを示します。
- 特定のオブジェクトに対する**オブジェクト権限**を、ユーザー、ロールおよび PUBLIC に付与します。表 11-3 に、各オブジェクトに付与できるオブジェクト権限を示します。表 11-4 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。これらのシステム権限は、GRANT コマンドを使用して付与します。

注意： データベース・ユーザーにロールの使用を許可する方法は、データベースや GRANT 文を介して行う以外にもあります。たとえば、オペレーティング・システムによっては、Oracle ユーザーに対して、初期化パラメータ OS_ROLES でロールを付与する機能を備えている場合もあります。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。

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、10-99 ページの「CREATE USER」および 9-141 ページの「CREATE ROLE」を参照してください。
- その他の認可方法の詳細は、『Oracle8*i* 管理者ガイド』を参照してください。
- 付与の取消しについては、11-73 ページの「REVOKE」を参照してください。

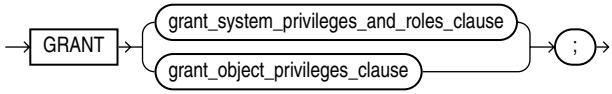
前提条件

システム権限を付与する場合は、ADMIN OPTION 付きのシステム権限または GRANT ANY PRIVILEGE システム権限が付与されている必要があります。

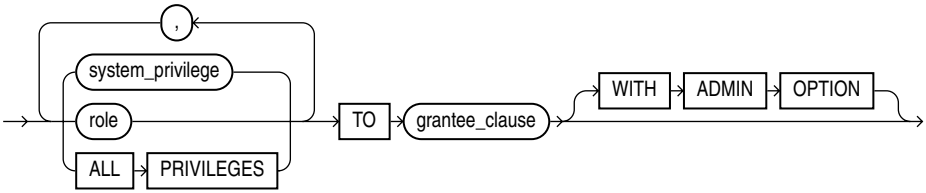
ロールを付与する場合は、ADMIN OPTION 付きのロールまたは GRANT ANY ROLE システム権限を付与されているか、あるいは付与するロールが自分で作成したロールである必要があります。

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必要があります。そうでない場合、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。この規則は、DBA ロールを持つユーザーに適用されます。

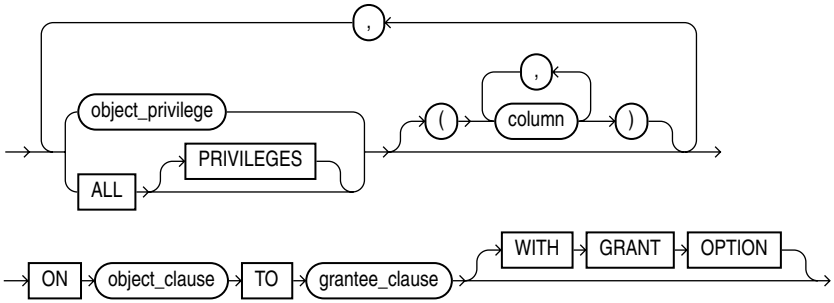
構文



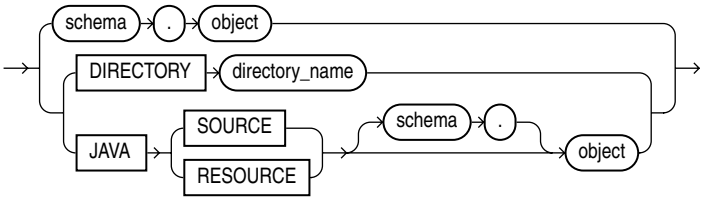
grant_system_privileges_and_roles_clause::=



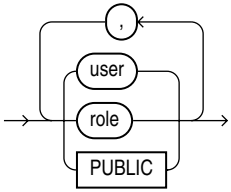
grant_object_privileges_clause::=



object_clause::=



grantee_clause::=



キーワードとパラメータ

grant_system_privileges_and_roles_clause

system_privileges

付与するシステム権限を指定します。表 11-1 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。
- **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。

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。

- 11-37 ページの表 11-1「システム権限」に示すすべてのシステム権限を付与する場合は、**ALL PRIVILEGES** を指定します。

role

付与するロールを指定します。事前に定義されたロールまたはユーザーが定義したロールを付与できます。表 11-2 は、事前に定義されたロールのリストです。

- **ユーザー**にロールを付与する場合、ユーザーに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。
- 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。ユーザーはそのロールを使用可能にし、ロールの権限ドメインに登録されている権限をすぐに使用できます。

  参照：ユーザー定義ロールの作成については、9-141 ページの「CREATE ROLE」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、そのロールを他のユーザーまたはロールに付与できます。
- 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。
- ロールへのアクセスに必要な認可を変更するため、ロールを変更できます。
- ロールを削除できます。

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。

ADMIN OPTION は取り消されません。ADMIN OPTION を取り消す場合、そのユーザーのロールおよびシステム権限も取り消し、ADMIN OPTION を指定せずにロールまたは権限を付与します。

grantee_clause

TO *grantee_clause* は、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します。

制限事項 : TO *grantee_clause* 句にユーザー、ロールおよび PUBLIC を指定できるのは 1 回のみです。

PUBLIC

すべてのユーザーに権限を付与する場合は、PUBLIC を指定します。

システム権限およびロールの付与に関する制限事項

- 権限またはロールの付与は 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_clause

object_privileges

付与するオブジェクト権限を指定します。表 11-3 に示すいずれかの値を指定します。表 11-4 も参照してください。

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]

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する場合に、ALL を指定します。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関するすべての権限を持っています（キーワード PRIVILEGES の指定は任意です）。

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関する権限の付与については、データ・ディクショナリ・ビュー USER_、ALL_ および DBA_CLUSTER_HASH_EXPRESSIONS に問い合せます。

> **参照：** データ・ディクショナリ・ビューの詳細は、『Oracle8*i* リファレンス・マニュアル』を参照してください。

WITH GRANT OPTION

WITH GRANT OPTION は、権限受領者が、他のユーザーまたはロールに対してオブジェクト権限を付与する場合に指定します。

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。

object_clause

ON *object_clause* は、権限が付与されているオブジェクトを識別します。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

権限を付与するスキーマ・オブジェクトを指定します。*object* に *schema* を指定しない場合、自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型があります。

- 表、ビュー、マテリアライズド・ビューまたはスナップショット
- 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

注意 : 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。単一パーティションに間接的に権限を付与する方法の詳細は、『Oracle8*i* 概要』を参照してください。

DIRECTORY *directory_name*
: 権限を付与するスキーマ・オブジェクトを指定します。*directory_name* にはスキーマ名を指定できません。

参照 : 9-39 ページの「CREATE DIRECTORY」を参照してください。

JAVA SOURCE | RESOURCE
: JAVA 句によって、権限が付与された Java ソースまたはリソース・スキーマ・オブジェクトを指定します。

参照 : 9-78 ページの「CREATE JAVA」を参照してください。

表 11-1 システム権限

システム権限名	許可される操作
クラスタ	
CREATE CLUSTER	自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	コンテキスト・ネームスペースの作成
DROP ANY CONTEXT	コンテキスト・ネームスペースの削除
データベース	
ALTER DATABASE	データベースの変更
ALTER SYSTEM	ALTER SYSTEM 文の発行
AUDIT SYSTEM	AUDIT *sql_statements* 文の発行

注意 : ANY オブジェクトの権限（CREATE ANY CLUSTER など）を付与した場合、SYS スキーマを含めたすべてのスキーマ内で、そのタイプのオブジェクトにアクセスできるようになります。SYS スキーマ内のオブジェクトへのアクセスを禁止するには、初期化パラメータ O7_DICTIONARY_ACCESSIBILITY を FALSE に設定します。ANY オブジェクトに付与された権限によって、SYS 以外のスキーマにアクセスできるようになります。

表 11-1 システム権限（続き）

システム権限名	許可される操作
データベース・リンク	
CREATE DATABASE LINK	自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
ディメンション	
CREATE DIMENSION	自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
索引タイプ	
CREATE INDEXTYPE	自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
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へのドメイン・インデックスまたは索引の作成
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
QUERY REWRITE	マテリアライズド・ビューまたは索引が自スキーマ内の表およびビューを参照している場合の次の操作、そのマテリアライズド・ビューを使用したリライト、ファンクション索引の作成

注意 : ANY オブジェクトの権限（CREATE ANY CLUSTER など）を付与した場合、SYS スキーマを含めたすべてのスキーマ内で、そのタイプのオブジェクトにアクセスできるようになります。SYS スキーマ内のオブジェクトへのアクセスを禁止するには、初期化パラメータ O7_DICTIONARY_ACCESSIBILITY を FALSE に設定します。ANY オブジェクトに付与された権限によって、SYS 以外のスキーマにアクセスできるようになります。

表 11-1 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・ビューまたは索引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次の操作、そのマテリアライズド・ビューを使用したリライト、ファンクション索引の作成
ライブラリ	
CREATE LIBRARY	自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 LIBRARY	自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
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（スナップショット）	
CREATE MATERIALIZED VIEW	自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	マテリアライズド・ビューまたは索引が自スキーマ内の表およびビューを参照している場合の次の操作、そのマテリアライズド・ビューを使用したリライト、ファンクション索引の作成
GLOBAL QUERY REWRITE	マテリアライズド・ビューまたは索引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次の操作、そのマテリアライズド・ビューを使用したリライト、ファンクション索引の作成
演算子	
CREATE OPERATOR	自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およびバインディングの作成
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参照の削除

注意 : ANY オブジェクトの権限（CREATE ANY CLUSTER など）を付与した場合、SYS スキーマを含めたすべてのスキーマ内で、そのタイプのオブジェクトにアクセスできるようになります。SYS スキーマ内のオブジェクトへのアクセスを禁止するには、初期化パラメータ O7_DICTIONARY_ACCESSIBILITY を FALSE に設定します。ANY オブジェクトに付与された権限によって、SYS 以外のスキーマにアクセスできるようになります。

表 11-1 システム権限（続き）

システム権限名	許可される操作
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	自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、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、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、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、ストアド・パッケージの削除
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

注意 : ANY オブジェクトの権限（CREATE ANY CLUSTER など）を付与した場合、SYS スキーマを含めたすべてのスキーマ内で、そのタイプのオブジェクトにアクセスできるようになります。SYS スキーマ内のオブジェクトへのアクセスを禁止するには、初期化パラメータ O7_DICTIONARY_ACCESSIBILITY を FALSE に設定します。ANY オブジェクトに付与された権限によって、SYS 以外のスキーマにアクセスできるようになります。

表 11-1 システム権限（続き）

システム権限名	許可される操作
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	自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ディメンションの削除
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	ALTER SESSION 文の発行
RESTRICTED SESSION	SQL*Plus の STARTUP RESTRICT 文によるインスタンスの起動後のログイン
スナップショット（マテリアライズド・ビュー）	
CREATE SNAPSHOT	自スキーマ内でのスナップショットの作成
CREATE ANY SNAPSHOT	任意のスキーマ内でのスナップショットの作成
ALTER ANY SNAPSHOT	データベース内の任意のスナップショットの変更
DROP ANY SNAPSHOT	任意のスキーマ内でのスナップショットの削除
GLOBAL QUERY REWRITE	スナップショットまたは索引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次の操作、そのスナップショットを使用したリライト、ファンクション索引の作成
QUERY REWRITE	スナップショットまたは索引が自スキーマ内の表およびビューを参照している場合の次の操作、そのスナップショットを使用したリライト、ファンクション索引の作成
シノニム	
CREATE SYNONYM	自スキーマ内でのシノニムの作成

注意 : ANY オブジェクトの権限（CREATE ANY CLUSTER など）を付与した場合、SYS スキーマを含めたすべてのスキーマ内で、そのタイプのオブジェクトにアクセスできるようになります。SYS スキーマ内のオブジェクトへのアクセスを禁止するには、初期化パラメータ O7_DICTIONARY_ACCESSIBILITY を FALSE に設定します。ANY オブジェクトに付与された権限によって、SYS 以外のスキーマにアクセスできるようになります。

表 11-1 システム権限（続き）

システム権限名	許可される操作
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	
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
UPDATE ANY TABLE	任意のスキーマ内の表またはビューの行の更新
SELECT ANY TABLE	任意のスキーマ内の表、ビュー、スナップショットの問合せ
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、表領域のバックアップの開始および終了の制御

注意 : ANY オブジェクトの権限（CREATE ANY CLUSTER など）を付与した場合、SYS スキーマを含めたすべてのスキーマ内で、そのタイプのオブジェクトにアクセスできるようになります。SYS スキーマ内のオブジェクトへのアクセスを禁止するには、初期化パラメータ O7_DICTIONARY_ACCESSIBILITY を FALSE に設定します。ANY オブジェクトに付与された権限によって、SYS 以外のスキーマにアクセスできるようになります。

表 11-1 システム権限（続き）

システム権限名	許可される操作
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オーバーライドします。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限り、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	自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	DATABASE 内でのトリガーの作成（CREATE TRIGGER または CREATE ANY TRIGGER 権限が必要）
型	
CREATE TYPE	自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。

注意 : ANY オブジェクトの権限（CREATE ANY CLUSTER など）を付与した場合、SYS スキーマを含めたすべてのスキーマ内で、そのタイプのオブジェクトにアクセスできるようになります。SYS スキーマ内のオブジェクトへのアクセスを禁止するには、初期化パラメータ O7_DICTIONARY_ACCESSIBILITY を FALSE に設定します。ANY オブジェクトに付与された権限によって、SYS 以外のスキーマにアクセスできるようになります。

表 11-1 システム権限（続き）

システム権限名	許可される操作
ユーザー	
CREATE USER	ユーザーの作成 この権限により、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 この権限により、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
BECOME USER	別のユーザーになること（全データベース・インポートを実行するユーザーに必要）
DROP USER	ユーザーの削除
ビュー	
CREATE VIEW	自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
その他	
ANALYZE ANY	任意のスキーマ内の任意の表、クラスタ、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
COMMENT ANY TABLE	任意のスキーマ内の任意の表、ビュー、列についてのコメントの記述
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・エラーを意図的に発生させます。

注意 : ANY オブジェクトの権限（CREATE ANY CLUSTER など）を付与した場合、SYS スキーマを含めたすべてのスキーマ内で、そのタイプのオブジェクトにアクセスできるようになります。SYS スキーマ内のオブジェクトへのアクセスを禁止するには、初期化パラメータ O7_DICTIONARY_ACCESSIBILITY を FALSE に設定します。ANY オブジェクトに付与された権限によって、SYS 以外のスキーマにアクセスできるようになります。

表 11-1 システム権限（続き）

システム権限名	許可される操作
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 PRIVILEGE	任意のシステム権限の付与
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 RESTRICTED SESSION 権限を含みます。
SYSOPER	STARTUP および SHUTDOWN 操作の実行 ALTER DATABASE OPEN/MOUNT/BACKUP ARCHIVELOG および RECOVERY RESTRICTED SESSION 権限を含みます。

注意 : ANY オブジェクトの権限（CREATE ANY CLUSTER など）を付与した場合、SYS スキーマを含めたすべてのスキーマ内で、そのタイプのオブジェクトにアクセスできるようになります。SYS スキーマ内のオブジェクトへのアクセスを禁止するには、初期化パラメータ O7_DICTIONARY_ACCESSIBILITY を FALSE に設定します。ANY オブジェクトに付与された権限によって、SYS 以外のスキーマにアクセスできるようになります。

表 11-2 Oracle によって定義されたロール

定義されたロール	用途
CONNECT、RESOURCE および DBA	前回のバージョンとの互換性を確保します。DBA_SYS_PRIVILEGES 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。 **参照 :** このビューの詳細は、『Oracle8*i* リファレンス・マニュアル』を参照してください。 **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、将来の Oracle バージョンでは自動的に作成されない可能性があります。
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にアクセスします。 **参照 :** これらのロールの詳細は、『Oracle8*i* 管理者ガイド』を参照してください。

表 11-2 Oracle によって定義されたロール（続き）

定義されたロール	用途
EXP_FULL_DATABASE IMP_FULL_DATABASE	EXP_FULL_DATABASE ロールおよび IMP_FULL_DATABASE ロールは、インポート / エクスポート・ユーティリティに有効です。 **参照 :** これらのロールの詳細は、『Oracle8*i* ユーティリティ・ガイド』を参照してください。
AQ_USER_ROLE AQ_ADMINISTRATOR_ROLE	Oracle のアドバンスト・キューイング機能を使用する場合、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、『Oracle8*i* アプリケーション開発者ガイド アドバンスト・キューイング』を参照してください。
SNMPAGENT	このロールは、Enterprise Manager/Intelligent Agent で使用します。 **参照 :** 『Oracle Enterprise Manager 開発者ガイド』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、『Oracle8*i* バックアップおよびリカバリ・ガイド』を参照してください。
HS_ADMIN_ROLE	Oracle の異種機間サービス機能を使用する DBA がデータ・ディクショナリにある適切な表にアクセスし、その表を DBMS_HS パッケージと一緒に操作する場合、このロールが必要です。 **参照 :** 詳細は、『Oracle8*i* 分散システム』および『Oracle8*i* PL/SQL パッケージ・プロシージャ リファレンス』を参照してください。

Oracle では、データベース管理を許可するロールも作成されます。多くのオペレーティング・システムでは、このようなロールには OSOPER および OSDBA という名前が付いています。ただし、実際の名前は、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。

表 11-3 特定のオブジェクトで使用可能なオブジェクト権限

オブジェクト権限	表	ビュー	順序	プロシージャ、ファンクション、パッケージ[a]	マテリアライズド・ビュー	ディレクトリ	ライブラリ	ユーザー定義型	演算子	索引タイプ
ALTER	X		X							
DELETE	X	X			X[b]					
EXECUTE				X			X	X	X	X
INDEX	X									
INSERT	X	X			X[b]					
READ						X				
REFERENCES	X									
SELECT	X	X	X		X					
UPDATE	X	X			X[b]					

[a]Oracle では、Java クラス、ソースまたはリソースを、オブジェクト権限の付与のためのプロシージャのように扱います。
[b]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。

表 11-4 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
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がある場合には、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。	
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 :** 表に対する DELETE 権限とともに SELECT 権限を付与する必要があります。
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。

表 11-4 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
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 :** 表に対する UPDATE 権限とともに SELECT 権限を付与する必要があります。
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がある場合は、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ベース表に関して GRANT OPTION 付きの権限が必要です。	
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
SELECT	SELECT 文でのビューの問合せ
UPDATE	UPDATE 文でのビューのデータの変更
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
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、ファンクションまたはパッケージの操作を許可します。この権限は、**Java ソース、クラスおよびリソース**にも適用されます。Oracle 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のように扱われます。	
EXECUTE	プロシージャまたはファンクションのコンパイルまたは直接実行、またはパッケージの仕様部に宣言されたプログラム・オブジェクトへのアクセス **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。 **参照 :** 『Oracle8*i* 概要』および『Oracle8*i* アプリケーション開発者ガイド基礎編』を参照してください。
次の**スナップショット権限**は、スナップショットについての操作を許可します。	
SELECT	SELECT 文でのスナップショットの問合せ

表 11-4 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
シノニム権限は、基本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。	
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 Server 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・アクセス権限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、Oracle はファイル運用時のセキュリティを実現できます。	
READ	ディレクトリ内のファイルの読取り
次の**オブジェクト型権限**は、オブジェクト型の操作を許可します。	
EXECUTE	特定のオブジェクトの使用および参照、また、そのメソッドの呼出し
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	索引タイプの参照
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	演算子の参照

例

ユーザーに対してシステム権限を付与する例　richard に CREATE SESSION システム権限を付与し、richard が Oracle にログインできるようにするには、次の文を発行します。

```
GRANT CREATE SESSION
    TO richard;
```

ロールに対してシステム権限を付与する例　travel_agent ロールに CREATE TABLE システム権限を付与するには、次の文を発行します。

```
GRANT CREATE TABLE
    TO travel_agent;
```

この結果、travel_agent の権限ドメインに CREATE TABLE システム権限が登録されます。

ロールに対してロールを付与する例　次の文は、EXECUTIVE ロールに travel_agent ロールを付与します。

```
GRANT travel_agent
    TO executive;
```

この結果、executive に travel_agent が付与され、executive の権限ドメインに CREATE TABLE システム権限が登録されます。

ADMIN OPTION 付きロールを付与する例　THOMAS に ADMIN OPTION 付きの executive ロールを付与するには、次の文を発行します。

```
GRANT executive
    TO thomas
    WITH ADMIN OPTION;
```

executive ロールによって、thomas は次の操作を実行できます。

- ロールを使用可能にして、CREATE TABLE 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

ディレクトリへのオブジェクト権限を付与する例　ユーザー scott にディレクトリ bfile_dir1 に対する READ を GRANT OPTION 付きで付与するには、次の文を発行します。

```
GRANT READ ON DIRECTORY bfile_dir1 TO scott
    WITH GRANT OPTION;
```

ユーザーに対して表へのシステム権限を付与する例　ユーザー jones に対して bonus 表についてのすべての権限を GRANT OPTION 付きで付与するには、次の文を発行します。

```
GRANT ALL ON bonus TO jones
    WITH GRANT OPTION;
```

この結果、jones は次の操作が実行できます。

- bonus 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、bonus 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　ビュー golf_handicap についての SELECT および UPDATE 権限をすべてのユーザーに付与するには、次の文を発行します。

```
GRANT SELECT, UPDATE
    ON golf_handicap TO PUBLIC;
```

この結果、すべてのユーザーが、ゴルフのハンディについてのビューを問合せおよび更新できます。

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　ユーザー blake に対して、スキーマ elly 内の eseq 順序の SELECT 権限を付与するには、次の文を発行します。

```
GRANT SELECT
    ON elly.eseq TO blake;
```

ユーザー blake は、次の文を指定して、順序の次の値を作成できます。

```
SELECT elly.eseq.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　ユーザー blake に対して、スキーマ SCOTT 内の EMP 表の empno 列についての REFERENCES 権限、および empno、sal、comm の各列についての UPDATE 権限を付与するには、次の文を発行します。

```
GRANT REFERENCES (empno), UPDATE (empno, sal, comm)
    ON scott.emp
    TO blake;
```

この結果、blake は empno、sal および comm の各列の値を更新できます。また、empno 列を参照する参照整合性制約を定義できます。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 blake は emp 表の他の列は操作できません。

たとえば、blake 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 scott.emp(empno) );
```

スキーマ scott 内の emp 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 文は、表、ビューのベース表、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・ビューのベース表に、行を追加する場合に使用します。

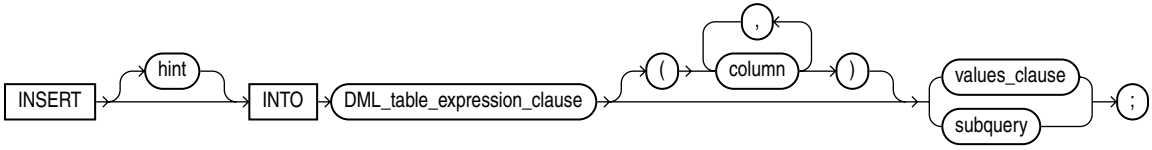
前提条件

表に行を挿入する場合は、その表が自スキーマ内にある必要があります。自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。

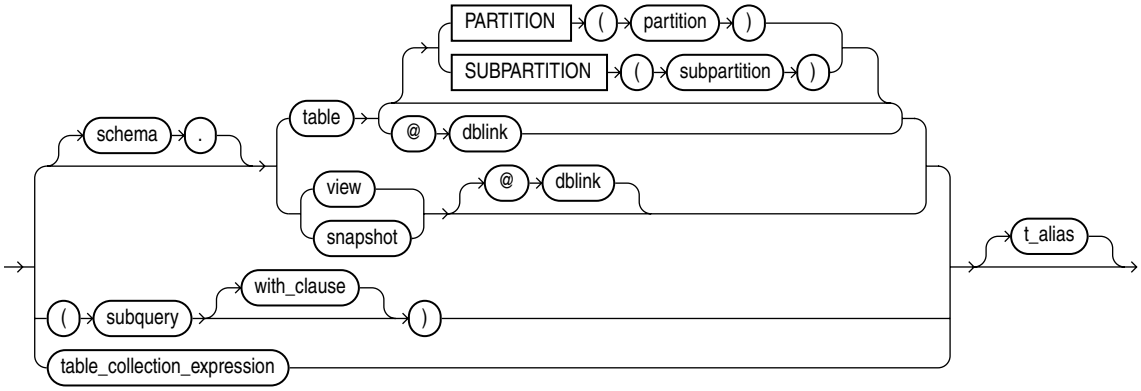
ビューのベース表に行を挿入する場合、ビューが定義されているスキーマの所有者には、そのベース表に対する INSERT 権限が必要です。また、他のユーザーのスキーマ内のビューに行を挿入する場合は、そのビューに対する INSERT 権限が必要です。

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ベース表に行を挿入できます。

構文

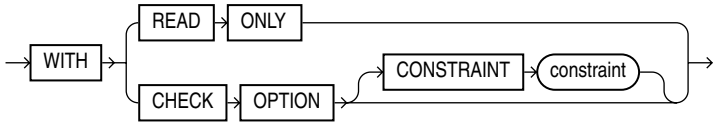


DML_table_expression_clause::=

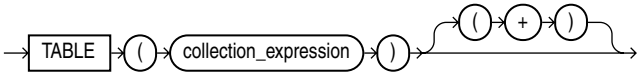


subquery: 11-88 ページの「SELECT および副問合せ」を参照してください。

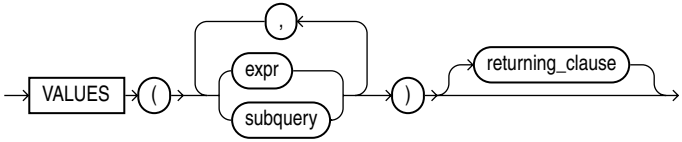
with_clause::=



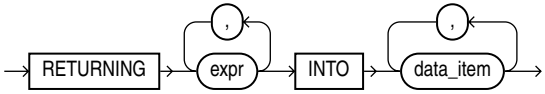
table_collection_expression::=



values_clause::=



returning_clause::=



キーワードとパラメータ

hint

文に対して実行計画を選択する際の、オプティマイザへ指示を引渡す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、2-65 ページの「ヒント」および『Oracle8*i* パフォーマンスのための設計およびチューニング』を参照してください。

DML_table_expression_clause

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、表またはビューが自スキーマにあるとみなされます。

table | *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、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、Oracle はそのビューのベース表に行を挿入します。

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 SCOPE 制約を持つ REF 列である必要があります。

table（または *view* のベース表）に、1 列以上のドメイン・インデックスがある場合は、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、『Oracle8*i* データ・カートリッジ開発者ガイド』を参照してください。

PARTITION (*partition_name*) | SUBPARTITION (*subpartition_name*)

挿入先の *table*（または *view* のベース表）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、Oracle はエラーを戻します。

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。

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 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、Oracle は、その表またはビューがローカル・データベース内にあるとみなします。

> **参照 :** データベース・リンクの参照方法の詳細は、2-86 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」を参照してください。

***DML_table_expression_clause* に関する制限事項 :**

- *table*（または *view* のベース表）に、LOADING または FAILED とマークされたドメイン・インデックスがある場合は、この文を実行できません。
- *DML_query_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、ビューに定義されている問合せを満たす行のみビューに挿入されます。
- 単一ベース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、INSTEAD OF トリガー以外の次のいずれかの構造体が含まれる場合は、そのビューは挿入できません。
  - 集合演算子
  - DISTINCT 演算子
  - 集計グループ関数
  - GROUP BY、ORDER BY、CONNECT BY または START WITH 句
  - SELECT リストのコレクション式
  - SELECT リストの副問合せ

- 結合（一部の例外を除く）

  詳細は、『Oracle8*i* 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 パラメータが TRUE に設定されていない限り、INSERT 文は正常に実行されません。

参照： 7-101 ページの「ALTER SESSION」を参照してください。

with_clause

副問合せを次のように制限する場合に、*with_clause* を使用します。

- WITH READ ONLY で副問合せを更新禁止にすることを指定します。
- WITH CHECK OPTION で、副問合せに存在しない行を生成する表変更の禁止を指定します。

参照： 11-108 ページの「WITH CHECK OPTION の例」を参照してください。

table_collection_expression

table_collection_expression は、コレクション値の式を表として扱う必要がある場合に Oracle に通知します。

参照： 11-115 ページの「表コレクションの例」を参照してください。

collection_expression　ネストした表の列を *table* または *view* から選択する副問合せを指定します。

注意：以前のリリースの Oracle では、*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されていません。

t_alias

文内で参照する表、ビューまたは副問合せの**相関名**（別名）を指定します。

column

表または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。

表のいずれかの列も指定しない場合、挿入された行の列の値には、表の作成時に指定したデフォルト値が使用されます。列のいずれかに NOT NULL 制約がある場合、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。

> **参照：** 列のデフォルト値の詳細は、10-7 ページの「CREATE TABLE」を参照してください。

values_clause

表またはビューに挿入する行の値を指定します。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。

制限事項：

- オブジェクトにある内部 LOB 属性を空または NULL 以外の値で初期化することはできません。つまり、リテラルを使用することはできません。
- BFILE ロケータを NULL に、またはディレクトリ別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：

- 11-61 ページの「BFILE への挿入例」を参照してください。
- BFILE の初期化の詳細は、『Oracle8*i* コール・インタフェース・プログラマーズ・ガイド』および『Oracle8*i* アプリケーション開発者ガイド基礎編』を参照してください。
- 有効な式の詳細は、5-2 ページの「式」および11-88 ページの「SELECT および副問合せ」を参照してください。

注意： 後続の問合せ中に文字リテラルを RAW 列に挿入する場合、RAW 列にある索引は使用せずに、フル・テーブル・スキャンを行います。

---

returning_clause

DML（INSERT、UPDATE または DELETE）文に影響される行を取り出します。この句は、表、スナップショット、および単一のベース表を持つビューに指定できます。

- 単一行で処理する場合は、*returning_clause* 付きの DML 文で、処理された行 ROWID および REF を使用して列式を取り出し、ホスト変数または PL/SQL 変数に格納します。
- 複数行で処理する場合は、*returning_clause* 付きの DML 文で、式の値、ROWID および処理された行に関連する REF をバインド配列に格納します。

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 句は、変更された行の値を、*data_item* リストに指定した変数に格納することを示します。
data_item	各 *data_item* は、取り出された *expr* 値を格納するホスト変数または PL/SQL 変数です。

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。

制限事項：

- パラレル DML またはリモート・オブジェクトでは、この句を使用できません。
- この句で 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は、この句を指定できません。

> **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、『Oracle8*i* PL/SQL 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。

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。

- VALUES なしで指定すると、副問合せは 0（ゼロ）または複数行を戻し、その行が挿入されます。
- VALUES とともに指定する場合、副問合せは**スカラー副問合せ**である必要があります。1 つの値を持つ 1 行を戻す必要があります。

副問合せによって、INSERT 文の対象となる表を含む任意の表、ビューおよびスナップショットを参照できます。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指定さ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。

subquery を TO_LOB 関数と組み合せて、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 を LOB に移行する場合、ベース表内で移行を実行してからビューに LOB を追加する必要があります。

参照 :

- 4-5 ページの「変換関数」を参照してください。
- LONG を LOB にコピーする理由および時期については、『Oracle8*i* 移行ガイド』を参照してください。
- TO_LOB 関数の使用方法については、11-61 ページの「TO_LOB を使用した挿入例」を参照してください。
- 11-88 ページの「SELECT および副問合せ」を参照してください。

注意 :

- *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、Oracle は、*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（問合せのリライト用）を使用することがあります。

  参照 : マテリアライズド・ビューおよび問合せのリライトの詳細は、『Oracle8*i* データ・ウェアハウス』を参照してください。

- この *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限り、INSERT はパラレルで実行されます ただし、*DML_query_expression_clause* の *subquery* がリモート・オブジェクトを参照する場合は、INSERT はシリアルで実行されます。

  参照 : 10-40 ページの「CREATE TABLE」の parallel_clause を参照してください。

例

値の挿入例　次の文は、dept 表に行を挿入します。

```
INSERT INTO dept
   VALUES (50, 'PRODUCTION', 'SAN FRANCISCO');
```

次の文は、emp 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 emp (empno, ename, job, sal, comm, deptno)
   VALUES (7890, 'JINKS', 'CLERK', 1.2E3, NULL, 40);
```

次の文は、前述の例と同じ結果を表しますが、*DML_query_expression_clause* にある副問合せを使用します。

```
INSERT INTO (SELECT empno, ename, job, sal, comm, deptno FROM emp)
   VALUES (7890, 'JINKS', 'CLERK', 1.2E3, NULL, 40);
```

副問合せを持つ値の挿入例　次の文は、管理職、社長、または歩合給が給与の 25% 以上の従業員を bonus 表にコピーします。

```
INSERT INTO bonus
   SELECT ename, job, sal, comm
   FROM emp
   WHERE comm > 0.25 * sal
   OR job IN ('PRESIDENT', 'MANAGER');
```

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 sales がアクセスできるデータベース上の、ユーザー scott が所有する accounts 表に行を挿入します。

```
INSERT INTO scott.accounts@sales (acc_no, acc_name)
   VALUES (5001, 'BOWER');
```

accounts 表に balance 列がある場合、INSERT 文に balance の値が指定されていないため、新しく挿入された行にはこの列のデフォルト値が割り当てられます。

順序値の挿入例　次の文は、従業員順序の次の値を持つ行を、emp 表に挿入します。

```
INSERT INTO emp
   VALUES  (empseq.nextval, 'LEWIS', 'CLERK',
            7902, SYSDATE, 1200, NULL, 20);
```

パーティションへの挿入例　次の文は、latest_data の行を sales 表のパーティション oct98 に挿入します。

```
INSERT INTO sales PARTITION (oct98)
   SELECT * FROM latest_data;
```

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。

```
INSERT INTO emp VALUES (empseq.nextval, 'LEWIS', 'CLARK',
                        7902, SYSDATE, 1200, NULL, 20)
   RETURNING sal*12, job INTO :bnd1, :bnd2;
```

バインド配列への戻り値の例　次の文は、バインド配列 :1 に挿入された行の参照値を戻します。

```
INSERT INTO employee
   VALUES ('Kitty Mine', 'Peaches Fuzz', 'Meena Katz')
   RETURNING REF(employee) INTO :1;
```

TO_LOB を使用した挿入例　次の文は、LONG データを次の既存の表にある LOB 列にコピーします。

```
CREATE TABLE long_tab (long_pics LONG RAW);
```

まず、LOB を持つ表を作成します。

```
CREATE TABLE lob_tab (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 (lob_pics)
   SELECT 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後、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;
```

BFILE への挿入例　BFILE を INSERT または UPDATE する場合、次の例に示すとおり NULL に、またはディレクトリ別名およびファイル名に初期化する必要があります。emp 表の BFILE 列の次に number 列がある場合、次のように入力します。

```
INSERT INTO emp
   VALUES (1, BFILENAME ('a_dir_alias', 'a_filename'));
```

LOCK TABLE

用途

LOCK TABLE 文は、1 つ以上の表、表パーティションまたはサブパーティションを特定のモードでロックする場合に使用します。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。

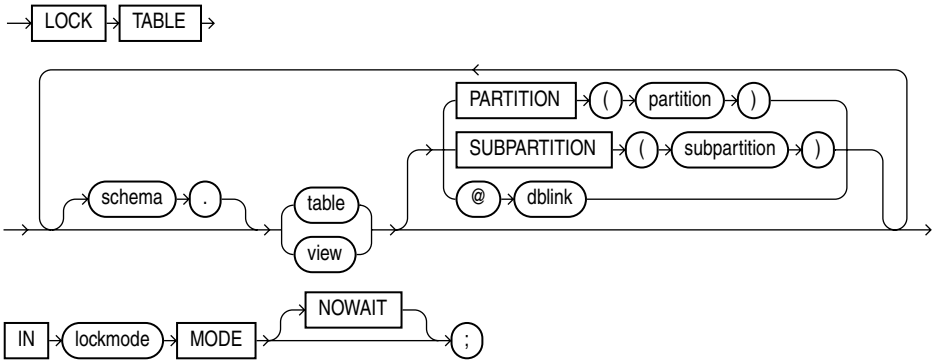
参照：

- ロック・モードの相互作用については、『Oracle8*i* 概要』を参照してください。
- 8-131 ページの「COMMIT」を参照してください。
- 11-83 ページの「ROLLBACK」を参照してください。
- 11-86 ページの「SAVEPOINT」を参照してください。

前提条件

表またはビューが自スキーマ内にある必要があります。自スキーマ内にない場合は、LOCK ANY TABLE システム権限が付与されているか、表またはビューに対するオブジェクト権限が必要です。

構文



キーワードとパラメータ

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、表またはビューが自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、ビューのベース表がロックされます。

PARTITION（パーティション）または SUBPARTITION（サブパーティション）を指定する場合、最初にその表を暗黙的にロックします。表ロックは、パーティションまたはサブパーティション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する場合、Oracle は、表を暗黙的に ROW SHARE ロックします。
- EXCLUSIVE ロックをサブパーティションに指定する場合、Oracle は、表を暗黙的に ROW EXCLUSIVE ロックします。

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、Oracle は、パーティションのすべてのサブパーティションをロックします。

dblink

表またはビューが格納されている、Oracle 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、2-88 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

次のいずれかのモードを指定します。

- ROW SHARE は、ロックされた表への同時アクセスを可能にしますが、排他アクセスのために表全体をロックすることはできなくなります。ROW SHARE は、SHARE UPDATE と同じ意味で、以前のバージョンの Oracle との互換性を保つために用意されています。
- 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。行の排他ロックは、更新、挿入、削除の実行時に自動的に適用されます。
- SHARE UPDATE については、ROW SHARE を参照してください。
- SHARE は、同時問合せは実行できますが、ロックされた表は更新できません。
- 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 は、ロックされた表上で問合せを実行できますが、他のアクティビティは実行できません。

NOWAIT

指定した表（あるいは指定した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すでにロックされている場合、制御をすぐに戻すには NOWAIT を指定します。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。

この句を指定しない場合、Oracle は、表が使用可能になるまで待ち状態になり、表をロックした後に、発行元に制御を戻します。

例

LOCK TABLE の例　次の文は、emp 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 emp
    IN EXCLUSIVE MODE
    NOWAIT;
```

次の文は、データベース・リンク boston を介してアクセスできるリモート accounts 表をロックします。

```
LOCK TABLE accounts@boston
    IN SHARE MODE;
```

NOAUDIT

用途

NOAUDIT は、AUDIT 文によって有効になった監査を停止する場合に使用します。

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 つの AUDIT 文（A）が、特定のユーザーに対して監査を有効にするとします。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。

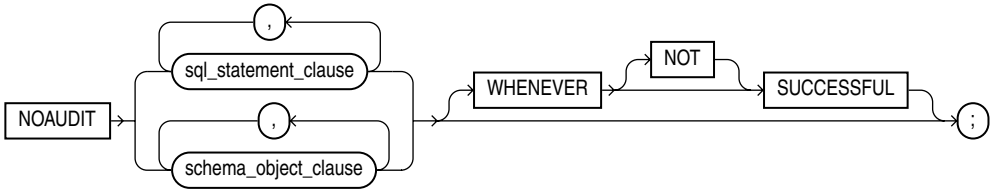
参照： 監査の詳細は、8-112 ページの「AUDIT」を参照してください。

前提条件

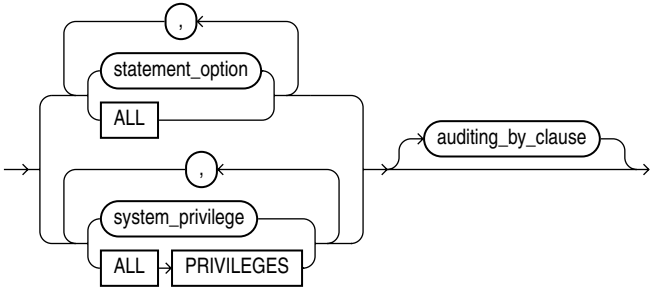
SQL 文の監査を停止するには、AUDIT SYSTEM システム権限が必要です。

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スキーマ内にある必要があります。自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。

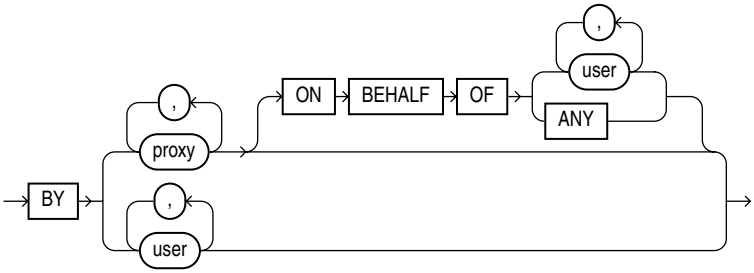
構文



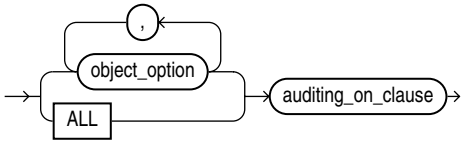
sql_statement_clause::=



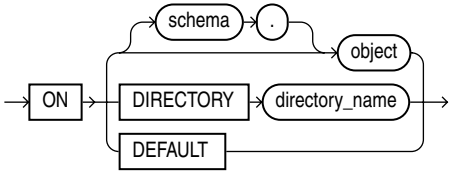
auditing_by_clause::=



schema_object_clause::=



auditing_on_clause::=



キーワードとパラメータ

sql_statement_clause

statement_option
監査を停止する文のオプションを指定します。

参照 : 文のオプションおよびそれによって監査される SQL 文のリストは、8-118 ページの表 8-1 および 8-121 ページの表 8-2 を参照してください。

ALL
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する場合に、ALL を指定します。

system_privilege
監査を停止するシステム権限を指定します。

参照 : 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、11-37 ページの表 11-1 を参照してください。

ALL PRIVILEGES
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する場合に、ALL PRIVILEGES を指定します。

auditing_by_clause
auditing_by_clause は、特定のユーザーが発行する SQL 文のみを監査します。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。

BY *user*
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を停止します。ただし、WHENEVER SUCCESSFUL で説明する状況は除きます。

BY *proxy*
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。

schema_object_clause

object_option
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。

参照 : これらのオプションのリストは、8-122 ページの表 8-3 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オプションをすべて指定することと同じです。

auditing_on_clause

auditing_on_clause では、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。

object

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、スナップショットまたはライブラリのオブジェクト名を指定します。*object* に *schema* を指定しない場合、自スキーマ内にあるとみなされます。

参照 : 特定のスキーマ・オブジェクトの監査については、8-112 ページの「AUDIT」を参照してください。

DIRECTORY *directory_name*

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

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。

WHENEVER [NOT] SUCCESSFUL

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する場合は、WHENEVER SUCCESSFUL を指定します。

NOT は、Oracle エラーとなった文および操作の監査のみを停止します。

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 scott および blake 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、scott の問合せの監査のみを停止します。

```
NOAUDIT SELECT TABLE BY scott;
```

この結果、ユーザー scott の問合せの監査のみが停止されます。blake の問合せと更新、および scott 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　スキーマ scott 内の emp 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、次の文を発行すると、この監査が停止されます。

```
NOAUDIT SELECT
   ON scott.emp;
```

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 scott.emp
   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 は、Oracle エラーの結果発生した問合せの監査を継続します。

RENAME

用途

RENAME 文は、表、ビュー、順序または（表、ビューまたは順序の）プライベート・シノニムを改名する場合に使用します。

- 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。
- 改名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、改名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。

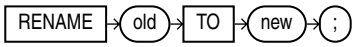
この文を使用してパブリック・シノニムを改名しないでください。そのかわり、該当するパブリック・シノニムを削除し、新しい名前で別のパブリック・シノニムを作成してください。

参照： 10-3 ページの「CREATE SYNONYM」および 11-5 ページの「DROP SYNONYM」を参照してください。

前提条件

オブジェクトが自スキーマ内にある必要があります。

構文



キーワードとパラメータ

old

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

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。

参照： 2-81 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。

例

データベース・オブジェクトの改名例　表の名前を dept から emp_dept に変更する場合、次の文を発行します。

```
RENAME dept TO emp_dept;
```

この文では列を直接改名できません。ただし、AS 副問合せを指定した CREATE TABLE 文とともにこの文を使用すると、列を改名できます。次の文は、static 表を再作成し、oldname から newname の列を改名します。

```
CREATE TABLE temporary (newname, col2, col3)
    AS SELECT oldname, col2, col3 FROM static;

DROP TABLE static;

RENAME temporary TO static;
```

REVOKE

用途

REVOKE 文は、次の処理を行う場合に使用します。

- 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

参照：

- システム権限およびロールの付与については、11-31 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概要については、11-47 ページの表 11-3 を参照してください。

前提条件

システム権限または**ロール**を取り消すには、ADMIN OPTION 付きの権限が必要です。

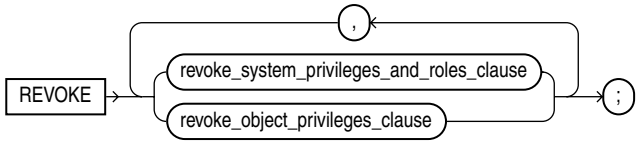
ロールを取り消すには、ADMIN OPTION 付きのロールが必要です。なお、GRANT ANY ROLE システム権限がある場合は、ロールを自由に取り消すことができます。

オブジェクト権限を取り消すには、各ユーザーおよびロールに、オブジェクト権限が事前に付与されている必要があります。

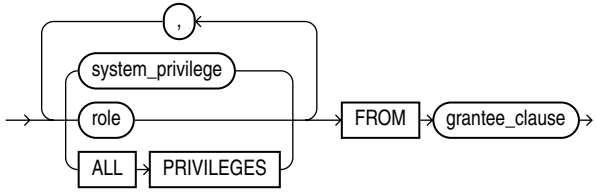
REVOKE 文によって取り消すことができる権限およびロールは、GRANT 文によって直接付与されているものに限られます。この句では、次の権限を取り消すことはできません。

- 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

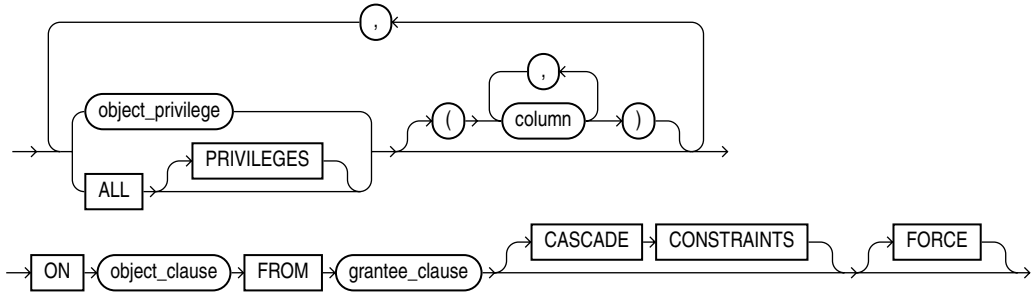
構文



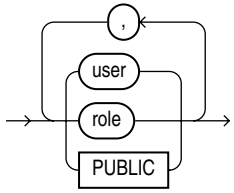
revoke_system_privileges_and_roles_clause::=



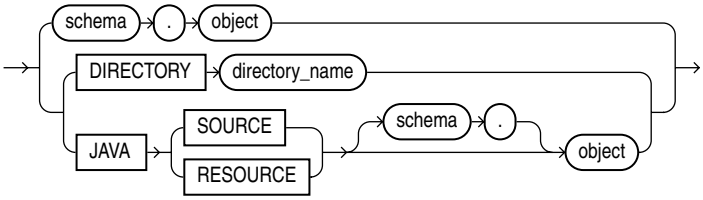
revoke_object_privileges_clause::=



grantee_clause::=



object_clause::=



キーワードとパラメータ

revoke_system_privileges_and_roles_clause

system_privilege

取り消すシステム権限を指定します。

参照 : システム権限のリストは、11-37 ページの表 11-1 を参照してください。

- **ユーザーの権限**を取り消す場合は、ユーザーの権限ドメインの権限を取り消します。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。
- **ロールの権限**を取り消す場合は、ロールの権限ドメインの権限を取り消します。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、そのロールが権限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。
- **PUBLIC の権限**を取り消す場合は、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の権限を取り消します。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。

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。

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。

- 11-37 ページの表 11-1 に示すすべてのシステム権限を取り消す場合は、ALL PRIVILEGES を指定します。

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場合は、ユーザーによるロールの使用を禁止します。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場合、Oracle は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を削除します。被取消し側ロールの権限が付与されており、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続き使用できます。ただし、被取消し側の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。
- **PUBLIC** のロールを取り消す場合、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、そのロールを使用禁止にします。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限り、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。

制限事項 : 取り消すロールのリストにロールを指定できるのは 1 回のみです。

参照 : 事前定義されたロールのリストは、11-45 ページの表 11-2 を参照してください。

grantee_clause

FROM *grantee_clause* は、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します。

PUBLIC　すべてのユーザーから権限を取り消す場合は、PUBLIC を指定します。

revoke_object_privileges_clause

object_privilege

取り消すオブジェクト権限を指定します。次のいずれかの値を指定できます。ALTER、DELETE、EXECUTE、INDEX、INSERT、READ、REFERENCES、SELECT または UPDATE

注意 : 操作は権限によって許可されます。権限を取り消すことによって、取り消されたユーザーは、その操作ができなくなります。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 に対して付与できます。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、権限受領者は引続きその権限を使用できます。

- **ユーザーの権限**を取り消す場合は、ユーザーの権限ドメインの権限を取り消します。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 権限を取り消す場合、CASCADE CONSTRAINTS 句も指定する必要があります。

- **ロールの権限**を取り消す場合は、ロールの権限ドメインの権限を取り消します。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。ロールが権限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 の権限**を取り消す場合は、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の権限を取り消します。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。

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。

ALL [PRIVILEGES]

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場合に指定します（キーワード PRIVILEGES の指定は任意です）。

注意 : 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。

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。

FORCE

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場合に、FORCE を指定します。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、依存する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、『Oracle8*i* 概要』を参照してください。

object_clause

ON *object_clause* は、権限を取り消すオブジェクトを識別します。

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 / スナップショット
- 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 / スナップショットに対するシノニム
- ライブラリ、索引タイプまたはユーザー定義演算子

schema 名を指定しなかった場合、自スキーマ内にあるとみなされます。

（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・ビューが含まれる表またはマテリアライズド・ビューのスナップショットから取り消すと、マテリアライズド・ビューは無効になります。

（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、マテリアライズド・ビューまたはビューおよびこれが含まれる表は無効になります。

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* にはスキーマを指定できません。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 : 9-39 ページの「CREATE DIRECTORY」を参照してください。

JAVA SOURCE | RESOURCE

JAVA 句によって、権限が取り消された Java ソースまたはリソース・スキーマ・オブジェクトを指定します。

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 bill および mary の DROP ANY TABLE システム権限を取り消します。

```
REVOKE DROP ANY TABLE
    FROM bill, mary;
```

この結果、bill と mary 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 hanson のロール controller を取り消します。

```
REVOKE controller
    FROM hanson;
```

この結果、hanson はロール controller 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 controller の CREATE TABLESPACE システム権限を取り消します。

```
REVOKE CREATE TABLESPACE
   FROM controller;
```

ロール controller 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 ceo のロール vp を取り消します。

```
REVOKE vp
  FROM ceo;
```

この結果、ceo は vp を使用できなくなります。

ユーザーからオブジェクト権限を取り消す例　次の文は、bonus 表に対する DELETE、INSERT、SELECT および UPDATE 権限をユーザー pedro に付与します。

```
GRANT ALL
    ON bonus TO pedro;
```

次の文は、ユーザー pedro から表 bonus に対する DELETE 権限を取り消します。

```
REVOKE DELETE
    ON bonus FROM pedro;
```

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 pedro の表 bonus に対する残りのすべての権限を取り消します。

```
REVOKE ALL
    ON bonus FROM pedro;
```

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 reports に対する SELECT 権限および UPDATE 権限を付与します。

```
GRANT SELECT, UPDATE
    ON reports TO public;
```

次の文は、すべてのユーザーから reports に対する UPDATE 権限を取り消します。

```
REVOKE UPDATE
    ON reports FROM public;
```

ユーザーは、reports 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、reports 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には、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 blake にスキーマ elly 内の eseq 順序に対する SELECT 権限を付与します。

```
GRANT SELECT
    ON elly.eseq TO blake;
```

blake から eseq に対する SELECT 権限を取り消す場合、次の文を発行します。

```
REVOKE SELECT
    ON elly.eseq FROM blake;
```

ただし、ユーザー elly が eseq に対する SELECT 権限を blake に付与している場合、blake は、elly からの権限付与によって eseq 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、blake に、スキーマ scott 内の emp 表に対する REFERENCES 権限および UPDATE 権限を付与します。

```
GRANT REFERENCES, UPDATE
    ON scott.emp TO blake;
```

blake は、REFERENCES 権限を使用して、スキーマ scott 内の emp 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 scott.emp(ename) );
```

CASCADE CONSTRAINTS 句を指定した次の文は、blake の scott.emp に対する REFERENCES 権限を取り消します。

```
REVOKE REFERENCES
    ON scott.emp
    FROM blake
    CASCADE CONSTRAINTS;
```

blake の scott.emp に対する REFERENCES 権限を取り消した場合、blake 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、blake が他のユーザーから scott.emp に対する REFERENCES 権限を付与されている場合は、Oracle はその制約を削除しません。他のユーザーから権限付与されたため、blake 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のオブジェクト権限を取り消す例　次の文は、sue のディレクトリ bfile_dir1 に対する READ 権限を取り消します。

```
REVOKE READ ON DIRECTORY bfile_dir1 FROM sue;
```

ROLLBACK

用途

ROLLBACK 文は、現行のトランザクションで実行された処理を取り消す場合、または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場合に使用します。

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：

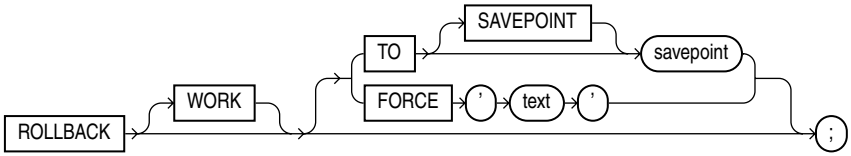
- トランザクションの詳細は、『Oracle8*i* 概要』を参照してください。
- 分散トランザクションの詳細は、『Oracle8*i* 分散システム』を参照してください。
- 現行トランザクションの特性の設定については、11-125 ページの「SET TRANSACTION」を参照してください。
- 8-131 ページの「COMMIT」を参照してください。
- 11-86 ページの「SAVEPOINT」を参照してください。

前提条件

現行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



キーワードとパラメータ

WORK

キーワード WORK の指定は任意です。ANSI 互換性のために提供されています。

TO SAVEPOINT *savepoint*

現行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。

ROLLBACK に TO SAVEPOINT 句を指定しないと、次の処理が行われます。

- トランザクションの終了
- 現行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
- トランザクションのロックの解除

参照： 11-86 ページの「SAVEPOINT」を参照してください。

ROLLBACK に TO SAVEPOINT 句を指定すると、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にロールバックします。
- 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はすべて消去されます。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を解除します。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。

制限事項：インダウト・トランザクションをセーブポイントまで手動でロールバックすることはできません。

FORCE

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に、FORCE を指定します。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 'text' 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。

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をロールバックするため注意してください。この文は、現行トランザクションには影響しません。

制限事項 : PL/SQL では、FORCE 句を指定する ROLLBACK 文はサポートされていません。

> **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については、『Oracle8*i* 分散システム』を参照してください。

例

次の文は、現行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、現行トランザクションをセーブポイント sp5 にロールバックします。

```
ROLLBACK TO SAVEPOINT sp5;
```

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';
```

SAVEPOINT

用途

SAVEPOINT 文は、トランザクション内でロールバックされる位置を指定する場合に使用します。

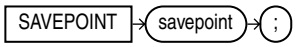
参照：

- セーブポイントの詳細は、『Oracle8*i* 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、11-83 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、11-125 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



キーワードとパラメータ

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指定した場合、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ができます。

例

blake と clark の給与を更新するために、会社の給与合計が $27,000 を超えていないことを確認してから、次のように clark の給与を再入力します。

```
UPDATE emp
    SET sal = 2000
    WHERE ename = 'BLAKE';
SAVEPOINT blake_sal;

UPDATE emp
    SET sal = 1500
    WHERE ename = 'CLARK';
SAVEPOINT clark_sal;

SELECT SUM(sal) FROM emp;

ROLLBACK TO SAVEPOINT blake_sal;

UPDATE emp
    SET sal = 1200
    WHERE ename = 'CLARK';

COMMIT;
```

SELECT および副問合せ

用途

SELECT 文または副問合せは、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場合に使用します。

注意： 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 文で指定した表のかわりに使用できます。このような代用を**問合せのリライト**といいます。これは、コストの最適化が使用可能で、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。問合せの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：

- 5-20 ページの「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・ビューおよび問合せのリライトの詳細は、『Oracle8*i* データ・ウェアハウス』を参照してください。
- 11-23 ページの「EXPLAIN PLAN」を参照してください。

前提条件

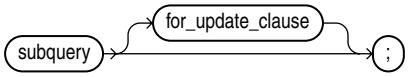
表またはマテリアライズド・ビューからデータを選択する場合、表またはマテリアライズド・ビューが自スキーマ内にあるか、またはその表またはマテリアライズド・ビューに対する SELECT 権限が必要です。

ビューのベース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。

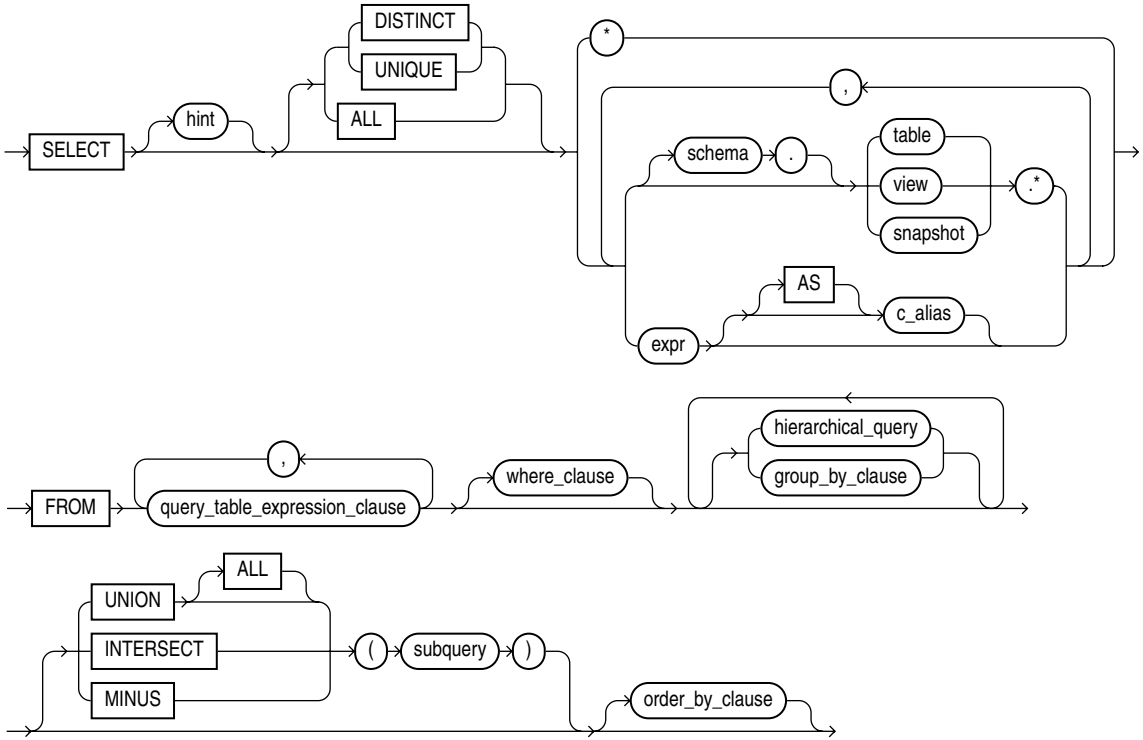
- ビューに対する SELECT 権限を持っている。
- ビューを含むスキーマの所有者が、ベース表に対する SELECT 権限を持っている。

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ベース表からデータを選択できます。

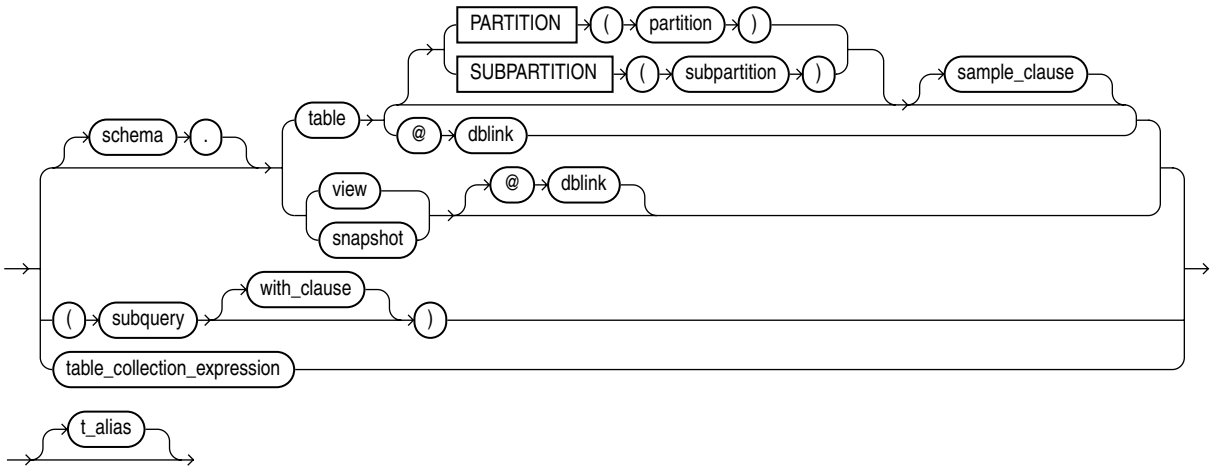
構文


subquery
for_update_clause
;

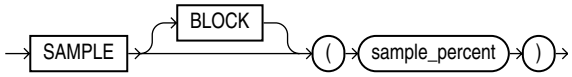
subquery::=


SELECT
hint
DISTINCT
UNIQUE
ALL
*
,
schema
.
table
view
snapshot
.*
expr
AS
c_alias
FROM
,
query_table_expression_clause
where_clause
hierarchical_query
group_by_clause
UNION
ALL
INTERSECT
MINUS
(
subquery
)
order_by_clause

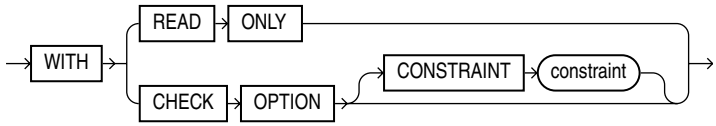
query_table_expression_clause::=



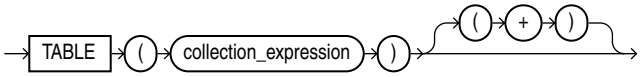
sample_clause::=



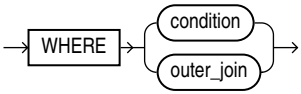
with_clause::=



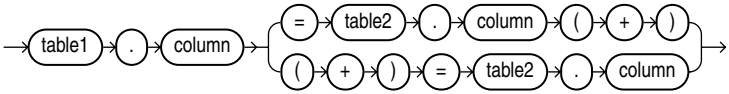
table_collection_expression::=



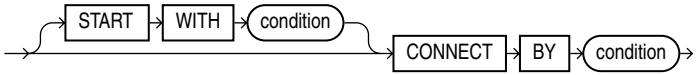
where_clause::=



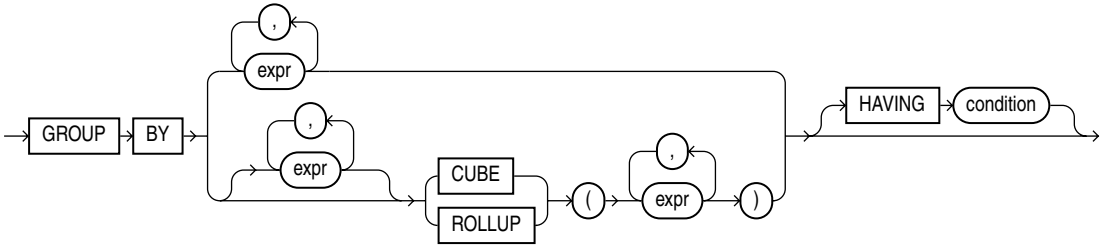
outer_join::=



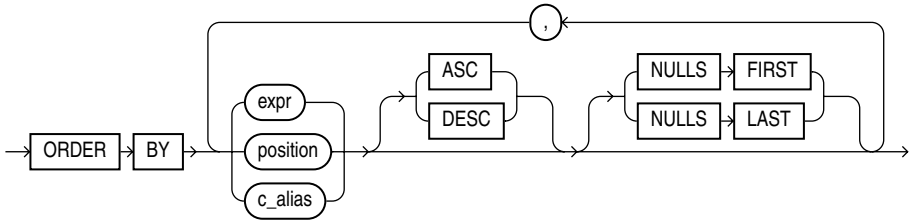
hierarchical_query_clause::=



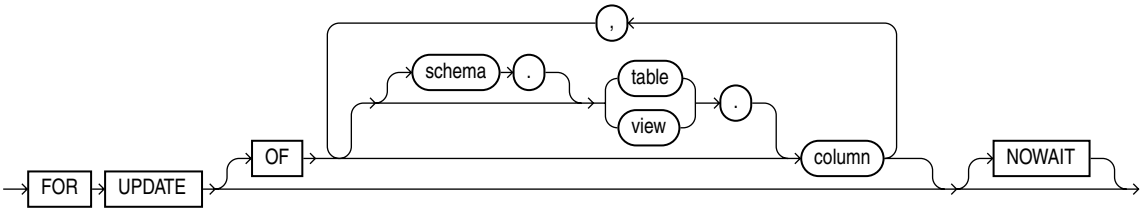
group_by_clause::=



order_by_clause::=



for_update_clause::=



キーワードとパラメータ

hint

文に対して実行計画を選択する際の、オプティマイザへ指示を渡す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、2-65 ページの「ヒント」および『Oracle8*i* パフォーマンスのための設計およびチューニング』を参照してください。

DISTINCT | UNIQUE

選択された行に重複行の 1 行のみを戻す場合に、DISTINCT または UNIQUE（これらの 2 つのキーワードが同義）を指定します。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。

制限事項：

- 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。
- FROM 句に LOB 列が含まれている場合、DISTINCT は指定できません。

ALL

重複行を含め、選択されたすべての行を戻す場合に、ALL を指定します。デフォルトは ALL です。

*

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する場合に、アスタリスクを指定します。

注意： 表から選択する（FROM 句に、ビューやマテリアライズド・ビューではなく表を指定する）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： 8-2 ページの「ALTER TABLE」を参照してください。

schema

選択した表、ビューまたはマテリアライズド・ビューが含まれるスキーマを指定します。*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・ビューは自スキーマ内にあるとみなされます。

table. | view.* | snapshot.**

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する場合に、ピリオドおよびアスタリスクの後にオブジェクト名を指定します。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。結合とは、2 つ以上の表、ビューまたはマテリアライズド・ビューの行を選択する問合せです。

参照： 5-23 ページの「結合」を参照してください。

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。

参照： *expr* の構文については、5-2 ページの「式」を参照してください。

制限事項：

- この文に *group_by_clause* も指定されている場合、この SELECT 構文のリストには次の式のみ指定できます。
  - 定数
  - 集計関数、USER 関数、UID 関数および SYSDATE 関数
  - *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。キー保存表の詳細は、『Oracle8*i* 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る場合、表の名前でその列名を修飾する必要があります。

c_alias

列式に他の名前（別名）を指定します。この別名は、列のヘッダーで使用します。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を効果的に改名できます。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。

FROM

query_table_expression_clause

FROM 句には、データを選択する表、ビュー、マテリアライズド・ビューまたはパーティション、あるいはデータを選択するオブジェクトを指定する副問合せを指定します。

PARTITION (*partition*)

SUBPARTITION (*subpartition*)

データを取り出すパーティションまたはサブパーティションを指定します。*partition* パラメータには、データ検索対象の *table* の中のパーティションの名前を指定するか、または検索を表の 1 つのパーティションのみに限定するより複雑な述語を指定できます。

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 のデータベース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、2-88 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、5-28 ページの「分散問合せ」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・データベースに存在するものとみなされます。

制限事項 : 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。

table, view, snapshot 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。「マテリアライズド・ビュー」と「スナップショット」は同義語です。

sample_clause

sample_clause では、表全体の行ではなく、表からサンプル行をランダムに選択します。

BLOCK BLOCK を指定した場合、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・サンプリングが行われます。

参照 : 相違点については、『Oracle8*i* 概要』を参照してください。

sample_percent *sample_percent* は、サンプルに含まれているとみなされる行またはブロックの割合（パーセント）を指定する数字です。値は .000001 ～ 99 の範囲内である必要があります。

sample_clause の制限事項

- 1 つの表から選択する問合せでのみ SAMPLE を指定できます。結合はサポートされません。ただし、CREATE TABLE ...AS SELECT を使用して、基礎となる表のサンプルをマテリアライズし、新しく作成されたサンプルを参照するように元の問合せを書き直しても、同じ結果になります。他の表に対してサンプルをマテリアライズするために、追加問合せを書き込むことができます。

  参照 : 11-105 ページの「SAMPLE の例」を参照してください。

- SAMPLE を指定した場合、自動的にコストベースのオプティマイザが使用されます。ルールベースのオプティマイザは、この句ではサポートされません。

注意 : 統計的にみて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。*sample_clause* の使用方法の詳細は、『Oracle8*i* 概要』を参照してください。

---

with_clause

副問合せを次のように制限する場合に、*with_clause* を使用します。

WITH READ ONLY	副問合せが更新禁止であることを示す場合に、WITH READ ONLY を指定します。
WITH CHECK OPTION	WITH CHECK OPTION は、INSERT、UPDATE または DELETE 文で、表のかわりに副問合せが使用された場合に、副問合せに含めることができない行を生成する表変更を禁止します。

参照 : 11-108 ページの「WITH CHECK OPTION の例」を参照してください。

table_collection_expression

問合せおよび DML 操作で、コレクション値表現を表として扱う場合に、*table_collection_expression* を指定します。*collection_expression* は副問合せ、列、CAST 式、DECODE 式、関数またはコレクション・コンストラクタにすることができます。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。

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 (+) を使用した場合、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。

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_expr* を「THE *subquery*」と表現していました。現在、このような表現方法はされていません。

参照 :

- 5-24 ページの「外部結合」を参照してください。
- 11-116 ページの「コレクション・ネスト解除の例」を参照してください。

t_alias

表、ビュー、または問合せを評価するための副問合せの**相関名**（別名）を指定します。相関名は、相関問合せ内で頻繁に使用する必要があります。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。

注意： *query_table_expression_clause* がオブジェクト型の属性またはオブジェクト型のメソッドを参照する場合、この別名が必要です。

---

where_clause

where_clause は、条件を満たすように行を制限します。

- *condition* には、有効な SQL 条件を指定します。

  参照： 条件の構文については、5-2 ページの「式」を参照してください。

- *outer_join* は、*query_table_expression_clause* が 1 つ以上の表を参照する場合にのみ適用されます。この特殊な形式の条件では、行が条件を満たさない表のすべての行、および条件をみたすすべての行を戻す必要があります。

  query_table_expression_clause の要素がネストした表または他の形式のコレクションの場合、*where_clause* ではなく *table_collection_expression* の外部結合の構文を指定します。

  参照： 外部結合に適用される規制、制限などの詳細は、5-24 ページの「外部結合」を参照してください。

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索引の DATE 列を参照する場合、Oracle は次の 2 つの条件を満たす場合にのみパーティション・プルーニングを行います。

(1) 4 桁書式マスクの TO_DATE 関数を使用して年を完全に指定し、表または索引パーティションを作成する。

(2) 2 または 4 桁書式マスクの TO_DATE 関数を使用して、問合せの *where_clause* に日付を指定する。

参照： 11-104 ページの「PARTITION の例」を参照してください。

hierarchical_query_clause

hierarchical_query_clause では、階層順序で行を選択できます。階層問合せの詳細は、5-21 ページの「階層問合せ」を参照してください。

前述の *where_clause*（指定されている場合）は、この階層の他の行に影響を与えずに問合せによって戻される行を制限します。

階層問合せを含む SELECT 文では、LEVEL 疑似列を使用できます。LEVEL では、ルート・ノードには 1、ルート・ノードの子ノードには 2、孫ノードには 3 というような値が戻ります。階層問合せによって戻されるレベルの数は、使用可能なユーザー・メモリーによって制限されます。

参照：

- LEVEL の詳細は、2-57 ページの「疑似列」を参照してください。
- 階層問合せについては、5-21 ページの「階層問合せ」を参照してください。

制限事項：階層問合せを指定する場合、次の制限事項があります。

- 同じ文は、結合を実行できません。
- 同じ文は、問合せで結合を実行するビューからはデータを選択できません。
- *order_by_clause* を指定した場合、階層問合せによって指定された順序より優先されます。

START WITH *condition*

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に条件を指定します。Oracle では、この条件を満たすすべての行が root で使用されます。この句を省略した場合、表内のすべての行が root 行として使用されます。START WITH *condition* には、副問合せを含めることができます。

CONNECT BY *condition*

階層の行の親子関係を識別する場合の条件を指定します。*condition* は、5-15 ページの「条件」に定義されているいずれの条件でもかまいません。ただし、この条件のどこかで、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。PRIOR 演算子を指定した条件は、次のいずれかの形式である必要があります。

- PRIOR expr comparison_operator expr
- expr comparison_operator PRIOR expr

制限事項 : CONNECT BY 条件に副問合せは記述できません。

group_by_clause

それぞれの行の *expr* の値に基づいて選択した行をグループ化し、各グループのサマリー情報を 1 行戻す場合に、*group_by_clause* を使用します。この句に CUBE または ROLLUP 拡張要素が指定された場合、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_clause 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。

制限事項 :

- *group_by_clause* には最大 255 個の式を指定できます。
- 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。
- *group_by_clause* に指定したすべての式の総バイト数は、データ・ブロックのサイズ（初期化パラメータ DB_BLOCK_SIZE によって指定されたもの）からオーバーヘッドを引いた値になります。
- *group_by_clause* がオブジェクト列を参照する場合、問合せはパラレル化されません。

ROLLUP

ROLLUP は、*group_by_clause* に対する拡張要素です。それぞれの行の式 n、n-1、n-2、... 0 の最初の値に基づいて選択した行をグループ化し、それぞれのグループのサマリー情報を 1 行戻します。ROLLUP 操作を使用して、**小計値**を出力できます。

たとえば、*group_by_clause* の ROLLUP 句に式を 3 つ指定した場合、操作の結果は n+1=3+1=4 グループになります。

最初の 'n' 式の値に基づいた行を**標準行**、その他を**超集合行**といいます。

参照 :

- 例については、4-41 ページの「GROUPING」を参照してください。
- 『Oracle8*i* データ・ウェアハウス』を参照してください。

CUBE

CUBE は、*group_by_clause* に対する拡張要素です。それぞれの行の式のあらゆる組合せでの値に基づいて選択した行をグループ化し、それぞれのグループのサマリー情報を 1 行戻します。CUBE 操作を使用して、**クロス集計値**を出力できます。

たとえば、*group_by_clause* の CUBE 句に式を 3 つ指定した場合、操作の結果は $2^n = 2^3=8$ グループになります。'n' 式の値に基づいた行を標準行、その他を**超集合行**といいます。

参照 :

- 4-41 ページの「GROUPING」を参照してください。
- 例については、11-106 ページの「CUBE の例」を参照してください。
- 『Oracle8*i* データ・ウェアハウス』を参照してください。

HAVING

指定した *condition* が TRUE である行のグループのみを戻す場合に、HAVING 句を使用します。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 CONNECT BY 句の後に、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 : *expr* の構文の詳細は、5-2 ページの「式」を、*condition* の構文の詳細は、5-15 ページの「条件」を参照してください。

集合演算子

UNION | UNION ALL | INTERSECT | MINUS

これらの集合演算子は、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。

集合演算子で 2 つ以上の問合せを結合する場合、隣接する問合せを左から右へと評価します。この評価順序を変更する場合、カッコを使用します。

参照 : これらの演算子の詳細は、3-13 ページの「集合演算子 : UNION [ALL]、INTERSECT、MINUS」を参照してください。

制限事項 :

- これらの集合演算子は、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。
- 列を参照する場合は、別名を使用して列に名前を付ける必要があります。
- *for_update_clause* はこれらの集合演算子とともには指定できません。
- 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。
- コンポーネント問合せのすべての SELECT 構文のリスト式の総バイト数は、データ・ブロックのサイズ（初期化パラメータ DB_BLOCK_SIZE で指定したサイズ）からオーバーヘッドを引いた値になります。

注意 : SQL 標準に準拠するために、Oracle の将来のリリースでは、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。

---

order_by_clause

文によって戻される行を順序付ける場合に、*order_by_clause* を使用します。*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。

- *expr* は、*expr* の値を基に行を順序付けます。式は、SELECT 構文のリストの列、FROM あるいは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* は、SELECT 構文のリストのその位置にある式の値に基づいて行を順序付けます。*position* は、整数である必要があります。

> **参照：** 問合せ結果の順位付けの詳細は、5-22 ページの「問合せ結果のソート」を参照してください。

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、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。

ASC | DESC	昇順か降順かを指定します。ASC がデフォルトです。
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。

制限事項：

- 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限り、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、ネストした表または VARRAY を使用しての順位付けはできません。

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
- 集計関数
- 分析関数

- USER 関数、UID 関数および SYSDATE 関数
- *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前述の式を伴っている式

for_update_clause

for_update_clause によって、トランザクションが終了する前に、別のユーザーによって選択した行が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します。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。

- 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、SELECT ... FOR UPDATE 文があります。

  参照： 11-108 ページの「LOB ロックの例」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。

OF
: 結合内の特定の表の選択された行のみをロックする場合に、OF 句を使用します。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
: SELECT 文で、他のユーザーによってロックされている行をロックしたときに制御を戻す場合、NOWAIT を指定します。この句を省略した場合、行が使用可能になるまで待ってから SELECT 文の結果が戻されます。

制限事項：

- この句を DISTINCT または CURSOR 演算子、集合演算子、*group_by_clause*、または集計関数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。

例

単純問合せの例　次の文は、部門番号 30 の従業員表 emp の行を選択します。

```
SELECT *
    FROM emp
    WHERE deptno = 30;
```

次の文は、部門番号 30 の営業担当員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 ename, job, sal, deptno
    FROM emp
    WHERE NOT (job = 'SALESMAN' AND deptno = 30);
```

次の文は、FROM 句の副問合せから、部門のすべての従業員数と給与合計がすべての部門に占める割合を算出します。

```
SELECT a.deptno "Department",
       a.num_emp/b.total_count "%Employees",
       a.sal_sum/b.total_sal "%Salary"
  FROM
 (SELECT deptno, COUNT(*) num_emp, SUM(SAL) sal_sum
    FROM scott.emp
    GROUP BY deptno) a,
 (SELECT COUNT(*) total_count, SUM(sal) total_sal
    FROM scott.emp) b ;
```

PARTITION の例　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、sales 表の nov98 パーティションへの別名の割当ておよび行の取出しを行います。

```
SELECT * FROM sales PARTITION (nov98) s
   WHERE s.amount_of_sale > 1000;
```

次の文は、sales 表から行を選択します。この場合、指定したデータより売上を先に検索します。

```
SELECT * FROM sales
   WHERE sale_date < TO_DATE('1998-06-15', 'YYYY-MM-DD');
```

SAMPLE の例　次の文は、emp 表の従業員数を推定します。

```
SELECT COUNT(*) * 100 FROM emp SAMPLE BLOCK (1);
```

次の文は、emp 表のサンプル・サブセットを生成し、結果のサンプル表と dept で結合します。この操作によって、結合問合せで *sample_clause* を指定できないという制限を回避できます。

```
CREATE TABLE sample_emp AS SELECT empno, deptno FROM emp SAMPLE(10);
SELECT e.empno FROM sample_emp e, dept d
   WHERE e.deptno = d.deptno AND d.name = 'DEV';
```

GROUP BY の例　次の文は、従業員表のそれぞれの部門について最高給与と最低給与を戻します。

```
SELECT deptno, MIN(sal), MAX (sal)
     FROM emp
     GROUP BY deptno;

DEPTNO     MIN(SAL)   MAX(SAL)
---------- ---------- ----------
        10       1300       5000
        20        800       3000
        30        950       2850
```

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 deptno, MIN(sal), MAX (sal)
     FROM emp
     WHERE job = 'CLERK'
     GROUP BY deptno;

DEPTNO     MIN(SAL)   MAX(SAL)
---------- ---------- ----------
        10       1300      1300
        20        800      1100
        30        950       950
```

CUBE の例　次の文は、部門およびジョブ・カテゴリのすべての組合せについて、従業員数と平均年収を戻します。

```
SELECT DECODE(GROUPING(dname), 1, 'All Departments',
      dname) AS dname,
   DECODE(GROUPING(job), 1, 'All Jobs', job) AS job,
   COUNT(*) "Total Empl", AVG(sal) * 12 "Average Sal"
   FROM emp, dept
   WHERE dept.deptno = emp.deptno
   GROUP BY CUBE (dname, job);

DNAME           JOB       Total Empl Average Sa
--------------- --------- ---------- ----------
ACCOUNTING      CLERK              1      15600
ACCOUNTING      MANAGER            1      29400
ACCOUNTING      PRESIDENT          1      60000
ACCOUNTING      All Jobs           3      35000
RESEARCH        ANALYST            2      36000
RESEARCH        CLERK              2      11400
RESEARCH        MANAGER            1      35700
RESEARCH        All Jobs           5      26100
SALES           CLERK              1      11400
SALES           MANAGER            1      34200
SALES           SALESMAN           4      16800
SALES           All Jobs           6      18800
All Departments ANALYST            2      36000
All Departments CLERK              4      12450
All Departments MANAGER            3      33100
All Departments PRESIDENT          1      60000
All Departments SALESMAN           4      16800
All Departments All Jobs          14 24878.5714
```

階層問合せの例　次の CONNECT BY 句は、親である行の empno 値が子である行の mgr 値と等しいという階層関係を定義します。

```
CONNECT BY PRIOR empno = mgr;
```

次の CONNECT BY 句では、PRIOR 演算子が empno 値にのみ適用されます。この条件を評価するために、Oracle は親である行に対しては empno の値を評価し、子である行に対しては mgr、sal および comm のそれぞれの値を評価します。

```
CONNECT BY PRIOR empno = mgr AND sal > comm;
```

子である行を限定する場合、mgr の値と親である行の empno の値が等しく、sal の値が comm の値より大きい必要があります。

HAVING の例　次の文は、事務員の最低給与が $1,000 以下の部門についての最高給与と最低給与を戻します。

```
SELECT deptno, MIN(sal), MAX (sal)
     FROM emp
     WHERE job = 'CLERK'
     GROUP BY deptno
     HAVING MIN(sal) < 1000;

DEPTNO     MIN(SAL)   MAX(SAL)
---------- ---------- ----------
        20        800       1100
        30        950        950
```

ORDER BY の例　次の文は、emp 表から営業担当員のすべてのレコードを選択し、その歩合によって降順にソートします。

```
SELECT *
    FROM emp
    WHERE job = 'SALESMAN'
    ORDER BY comm DESC;
```

次の文は、emp 表から従業員を選択し、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 ename, deptno, sal
    FROM emp
    ORDER BY deptno ASC, sal DESC;
```

次の文は、先の SELECT 文と同じ情報を選択し、位置に基づく ORDER BY 句の指定を使用します。

```
SELECT ename, deptno, sal
    FROM emp
    ORDER BY 2 ASC, 3 DESC;
```

FOR UPDATE の例　次の文は、ニューヨーク勤務の事務員の emp 表中の行をロックし、事務員がいるニューヨークの部門の dept 表中の行をロックします。

```
SELECT empno, sal, comm
    FROM emp, dept
    WHERE job = 'CLERK'
        AND emp.deptno = dept.deptno
        AND loc = 'NEW YORK'
    FOR UPDATE;
```

次の文は、ニューヨーク勤務の事務員の emp 表のみをロックします。dept 表では、行のロックはできません。

```
SELECT empno, sal, comm
    FROM emp, dept
    WHERE job = 'CLERK'
        AND emp.deptno = dept.deptno
        AND loc = 'NEW YORK'
    FOR UPDATE OF emp.sal;
```

LOB ロックの例　次の文は、SELECT ... FOR UPDATE 文を使用して、LOB 値を更新する前にその LOB が含まれている行をロックします。

```
INSERT INTO t_table VALUES (1, 'abcd');

COMMIT;
   DECLARE
     num_var      NUMBER;
     clob_var     CLOB;
     clob_locked  CLOB;
     write_amount NUMBER;
     write_offset NUMBER;
     buffer       VARCHAR2(20) := 'efg';

   BEGIN
     SELECT clob_col INTO clob_locked FROM t_table
     WHERE num_col = 1 FOR UPDATE;

  write_amount := 3;
  dbms_lob.write(clob_locked, write_amount, write_offset, buffer);
END;
```

WITH CHECK OPTION の例　次の文は、2 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 ename, deptno FROM emp WHERE deptno < 10)
    VALUES ('Taylor', 20);
```

ただし、次の文は WITH CHECK OPTION 句によって無効になります。

```
INSERT INTO
   (SELECT ename, deptno FROM emp
       WHERE deptno < 10
       WITH CHECK OPTION)
    VALUES ('Taylor', 20);
```

等価結合の例　次の等価結合は、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 ename, job, dept.deptno, dname
     FROM emp, dept
     WHERE emp.deptno = dept.deptno;

ENAME      JOB       DEPTNO     DNAME
---------- --------- ---------- --------------
CLARK      MANAGER           10 ACCOUNTING
KING       PRESIDENT         10 ACCOUNTING
MILLER     CLERK             10 ACCOUNTING
SMITH      CLERK             20 RESEARCH
ADAMS      CLERK             20 RESEARCH
FORD       ANALYST           20 RESEARCH
SCOTT      ANALYST           20 RESEARCH
JONES      MANAGER           20 RESEARCH
ALLEN      SALESMAN          30 SALES
BLAKE      MANAGER           30 SALES
MARTIN     SALESMAN          30 SALES
JAMES      CLERK             30 SALES
TURNER     SALESMAN          30 SALES
WARD       SALESMAN          30 SALES
```

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。

```
emp.deptno = dept.deptno
```

次の等価結合は、すべての事務員の名前、職種、部門番号および部門名を戻します。

```
SELECT ename, job, dept.deptno, dname
     FROM emp, dept
     WHERE emp.deptno = dept.deptno
     AND job = 'CLERK';

ENAME      JOB       DEPTNO     DNAME
---------- --------- ---------- --------------
MILLER     CLERK             10 ACCOUNTING
SMITH      CLERK             20 RESEARCH
ADAMS      CLERK             20 RESEARCH
JAMES      CLERK             30 SALES
```

この問合せは、次の *where_clause* を使用して 'CLERK' 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。

副問合せの例　'TAYLOR' 部門で働く人を判断する場合、次の文を発行します。

```
SELECT ename, deptno
    FROM emp
    WHERE deptno =
        (SELECT deptno
            FROM emp
            WHERE ename = 'TAYLOR');
```

ボーナスがまだ支給されていない（bonus 表に存在しない）emp 表の従業員の給料を 10% アップする場合、次の文を発行します。

```
UPDATE emp
    SET sal = sal * 1.1
    WHERE empno NOT IN (SELECT empno FROM bonus);
```

newdept という名前の dept 表の複製を作成する場合、次の文を発行します。

```
CREATE TABLE newdept (deptno, dname, loc)
    AS SELECT deptno, dname, loc FROM dept;
```

内部結合の例　次の問合せは、内部結合を使用して、それぞれの従業員の名前と一緒にその従業員の上司の名前を戻します。

```
SELECT e1.ename||' works for '||e2.ename
"Employees and their Managers"
    FROM emp e1, emp e2   WHERE e1.mgr = e2.empno;

Employees and their Managers
------------------------------
BLAKE works for KING
CLARK works for KING
JONES works for KING
FORD works for JONES
SMITH works for FORD
ALLEN works for BLAKE
WARD works for BLAKE
MARTIN works for BLAKE
SCOTT works for JONES
TURNER works for BLAKE
ADAMS works for SCOTT
JAMES works for BLAKE
MILLER works for CLARK
```

この問合せの結合条件では、emp 表に対する別名 e1 および e2 を使用します。

```
e1.mgr = e2.empno
```

外部結合の例　次の問合せは、外部結合を使用して前述の等価結合の例の結果を拡張します。

```
SELECT ename, job, dept.deptno, dname
    FROM emp, dept
    WHERE emp.deptno (+) = dept.deptno;

ENAME      JOB       DEPTNO     DNAME
---------- --------- ---------- --------------
CLARK      MANAGER           10 ACCOUNTING
KING       PRESIDENT         10 ACCOUNTING
MILLER     CLERK             10 ACCOUNTING
SMITH      CLERK             20 RESEARCH
ADAMS      CLERK             20 RESEARCH
FORD       ANALYST           20 RESEARCH
SCOTT      ANALYST           20 RESEARCH
JONES      MANAGER           20 RESEARCH
ALLEN      SALESMAN          30 SALES
BLAKE      MANAGER           30 SALES
MARTIN     SALESMAN          30 SALES
JAMES      CLERK             30 SALES
TURNER     SALESMAN          30 SALES
WARD       SALESMAN          30 SALES
                             40 OPERATIONS
```

この外部結合では、operations 部門を含む行を、従業員がこの部門で働いていない場合でも戻します。この行の ename 列および job 列には NULL が戻されます。この例の結合問合せでは、従業員のいる部門のみを選択します。

次の問合せは、外部結合を使用して前述の結果を拡張します。

```
SELECT ename, job, dept.deptno, dname
    FROM emp, dept
    WHERE emp.deptno (+) = dept.deptno
        AND job (+) = 'CLERK';

ENAME      JOB       DEPTNO     DNAME
---------- --------- ---------- --------------
MILLER     CLERK             10 ACCOUNTING
SMITH      CLERK             20 RESEARCH
ADAMS      CLERK             20 RESEARCH
JAMES      CLERK             30 SALES
                             40 OPERATIONS
```

この外部結合では、operations 部門を含む行を、事務員がこの部門で働いていない場合でも戻します。job 列の (+) 演算子によって、job 列が NULL である行も戻されます。この (+) が省略された場合、operations 部門を含む行は、job 値が 'CLERK' ではないため戻されません。

次に、customers、orders、lineitems および parts の 4 つの表の外部結合問合せを示します。これらの表は、それぞれ次のようになります。

```
SELECT custno, custname
    FROM customers;

CUSTNO     CUSTNAME
---------- --------------------
         1 Angelic Co.
         2 Believable Co.
         3 Cables R Us

SELECT orderno, custno,
    TO_CHAR(orderdate, 'MON-DD-YYYY') "ORDERDATE"
    FROM orders;

ORDERNO    CUSTNO     ORDERDATE
---------- ---------- -----------
      9001          1 OCT-13-1998
      9002          2 OCT-13-1998
      9003          1 OCT-20-1998
      9004          1 OCT-27-1998
      9005          2 OCT-31-1998

SELECT orderno, lineno, partno, quantity
    FROM lineitems;

ORDERNO    LINENO     PARTNO     QUANTITY
---------- ---------- ---------- ----------
      9001          1        101         15
      9001          2        102         10
      9002          1        101         25
      9002          2        103         50
      9003          1        101         15
      9004          1        102         10
      9004          2        103         20
```

```
SELECT partno, partname
    FROM parts;

PARTNO PARTNAME
------ --------
   101 X-Ray Screen
   102 Yellow Bag
   103 Zoot Suit
```

顧客 Cables R Us は何も注文していません。また、注文番号 9005 の明細項目はありません。

次の外部結合は、すべての顧客名と顧客が注文した日付を戻します。(+) 演算子を使用した場合、注文をしていない顧客も戻されます。

```
SELECT custname, TO_CHAR(orderdate, 'MON-DD-YYYY') "ORDERDATE"
    FROM customers, orders
    WHERE customers.custno = orders.custno (+);

CUSTNAME             ORDERDATE
-------------------- --------------
Angelic Co.          OCT-13-1993
Angelic Co.          OCT-20-1993
Angelic Co.          OCT-27-1993
Believable Co.       OCT-13-1993
Believable Co.       OCT-31-1993
Cables R Us
```

次の外部結合は、`lineitems` 表を `FROM` 句に、この表の列を SELECT 構文のリストに、この表と `orders` 表を結合する結合条件を *`where_clause`* に追加し、前述の外部結合の結果に別の情報を追加します。この問合せは、前述の問合せ結果を `lineitems` 表と結合し、すべての顧客名、顧客が注文した日付、および注文した各部品の部品番号と数量を戻します。最初の (+) 演算子は、前述の問合せと同じ用途で使用します。2 番目の (+) 演算子によって、明細項目がない注文も確実に戻されます。

```
SELECT custname,
   TO_CHAR(orderdate, 'MON-DD-YYYY') "ORDERDATE",
   partno,
   quantity
      FROM customers, orders, lineitems
      WHERE customers.custno = orders.custno (+)
      AND orders.orderno = lineitems.orderno (+);
```

```
CUSTNAME             ORDERDATE      PARTNO     QUANTITY
-------------------- -------------- ---------- ----------
Angelic Co.          OCT-13-1993           101         15
Angelic Co.          OCT-13-1993           102         10
Angelic Co.          OCT-20-1993           101         15
Angelic Co.          OCT-27-1993           102         10
Angelic Co.          OCT-27-1993           103         20
Believable Co.       OCT-13-1993           101         25
Believable Co.       OCT-13-1993           103         50
Believable Co.       OCT-31-1993
Cables R Us
```

次の外部結合は、parts 表を FROM 句に、この表の partname 列を SELECT 構文のリストに、この表と lineitems 表を結合する結合条件を *where_clause* に追加することで、前述の外部結合の結果に別の情報を追加します。この問合せは、前述の問合せ結果を parts 表と結合し、すべての顧客名、顧客が注文した日付および注文した各部品の数量と部品名を戻します。最初の 2 つの (+) 演算子は、前述の問合せと同じ用途で使用します。3 番目の (+) 演算子によって、部品番号が NULL の行も戻されます。

```
SELECT custname, TO_CHAR(orderdate, 'MON-DD-YYYY') "ORDERDATE",
    quantity, partname
    FROM customers, orders, lineitems, parts
    WHERE customers.custno = orders.custno (+)
    AND orders.orderno = lineitems.orderno (+)
    AND lineitems.partno = parts.partno (+);

CUSTNAME             ORDERDATE        QUANTITY PARTNAME
-------------------- -------------- ---------- ------------
Angelic Co.          OCT-13-1993            15 X-Ray Screen
Angelic Co.          OCT-13-1993            10 Yellow Bag
Angelic Co.          OCT-20-1993            15 X-Ray Screen
Angelic Co.          OCT-27-1993            10 Yellow Bag
Angelic Co.          OCT-27-1993            20 Zoot Suit
Believable Co.       OCT-13-1993            25 X-Ray Screen
Believable Co.       OCT-13-1993            50 Zoot Suit
Believable Co.       OCT-31-1993
Cables R Us
```

表コレクションの例　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_expression_clause* が *table_collection_expression* の場合、コレクション式は、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、この使用例に基づいています。

```
CREATE TYPE ProjectType AS OBJECT(
    pno    NUMBER,
    pname  CHAR(31),
    budget NUMBER);
CREATE TYPE ProjectSet AS TABLE OF ProjectType;

CREATE TABLE Dept (dno NUMBER, dname CHAR(31), projs ProjectSet)
    NESTED TABLE projs STORE AS
        ProjectSetTable ((Primary Key(Nested_Table_Id, pno)) ORGANIZATION
INDEX COMPRESS 1);

INSERT INTO Dept VALUES (1, 'Engineering', ProjectSet());
```

次の例では、Engineering 部門のネストした表 projs に挿入します。

```
INSERT INTO TABLE(SELECT d.projs
                  FROM   Dept d
                  WHERE  d.dno = 1)
  VALUES (1, 'Collection Enhancements', 10000);
```

次の例では、Engineering 部門のネストした表 projs を更新します。

```
UPDATE TABLE(SELECT d.projs
             FROM   Dept d
             WHERE  d.dno = 1) p
  SET  p.budget = p.budget + 1000;
```

次の例では、Engineering 部門のネストした表 projs から削除します。

```
DELETE TABLE(SELECT d.projs
             FROM   Dept d
             WHERE  d.dno = 1) p
 WHERE p.budget > 100000;
```

コレクション・ネスト解除の例　データベースに、dept 列、location 列、mgr 列を持つ hr_info 表と、name 列、dept 列および sal 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。次の文を使用して、hr_info および people のすべての列を選択できます。

```
SELECT t1.dept, t2.* FROM hr_info t1, TABLE(t1.people) t2
   WHERE t2.dept = t1.dept;
```

people は、hr_info のネストした表の列ではなく、name、dept、address、hiredate および sal 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, t2.*
   FROM hr_info t1, TABLE(CAST(MULTISET(
      SELECT t3.name, t3.dept, t3.sal FROM people t3
      WHERE t3.dept = t1.dept)
      AS NESTED_PEOPLE)) 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、部門および給料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.dept, t2.* FROM HY_INFO t1, TABLE(CAST
   (people_func( ... ) AS NESTED_PEOPLE)) t2;
```

> **参照：**　コレクション・ネスト解除の詳細は、『Oracle8*i* アプリケーション開発者ガイド 基礎編』を参照してください。

LEVEL の例　次の文を実行すると、すべての従業員が階層順序で戻されます。職種が 'PRESIDENT' である従業員が root 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)) || ename org_chart,
        empno, mgr, job
    FROM emp
    START WITH job = 'PRESIDENT'
    CONNECT BY PRIOR empno = mgr;

ORG_CHART    EMPNO      MGR        JOB
------------ ---------- ---------- ---------
KING               7839            PRESIDENT
  JONES            7566       7839 MANAGER
    SCOTT          7788       7566 ANALYST
      ADAMS        7876       7788 CLERK
    FORD           7902       7566 ANALYST
```

```
      SMITH        7369       7902 CLERK
  BLAKE            7698       7839 MANAGER
    ALLEN          7499       7698 SALESMAN
    WARD           7521       7698 SALESMAN
    MARTIN         7654       7698 SALESMAN
    TURNER         7844       7698 SALESMAN
      JAMES        7900       7698 CLERK
 CLARK             7782       7839 MANAGER
   MILLER          7934       7782 CLERK
```

次の文は、前述の例とほぼ同じですが、職種が 'ANALYST' である従業員は選択されません。

```
SELECT LPAD(' ', 2*(LEVEL-1)) || ename org_chart,
     empno, mgr, job
     FROM emp
     WHERE job != 'ANALYST'
     START WITH job = 'PRESIDENT'
     CONNECT BY PRIOR empno = mgr;

ORG_CHART            EMPNO      MGR        JOB
-------------------- ---------- ---------- ---------
KING                       7839            PRESIDENT
  JONES                    7566       7839 MANAGER
      ADAMS                7876       7788 CLERK
      SMITH                7369       7902 CLERK
  BLAKE                    7698       7839 MANAGER
    ALLEN                  7499       7698 SALESMAN
    WARD                   7521       7698 SALESMAN
    MARTIN                 7654       7698 SALESMAN
    TURNER                 7844       7698 SALESMAN
    JAMES                  7900       7698 CLERK
  CLARK                    7782       7839 MANAGER
    MILLER                 7934       7782 CLERK
```

アナリスト scott と ford のデータは戻されませんが、この 2 人の部下である従業員のデータは戻されます。

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 つのレベルのみが選択されます。

```
SELECT LPAD(' ',2*(LEVEL-1)) || ename org_chart,
empno, mgr, job
    FROM emp
    START WITH job = 'PRESIDENT'
    CONNECT BY PRIOR empno = mgr AND LEVEL <= 2;
```

```
ORG_CHART    EMPNO      MGR        JOB
------------ ---------- ---------- ---------
KING               7839            PRESIDENT
  JONES            7566       7839 MANAGER
  BLAKE            7698       7839 MANAGER
  CLARK            7782       7839 MANAGER
```

分散問合せの例　次の文は、ローカル・データベース上の dept 表と、houston データベース上の emp 表を結合します。

```
SELECT ename, dname
    FROM emp@houston, dept
    WHERE emp.deptno = dept.deptno;
```

相関副問合せの例　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 emp 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 deptno, ename, sal
    FROM emp x
    WHERE sal > (SELECT AVG(sal)
          FROM emp
          WHERE x.deptno = deptno)
    ORDER BY deptno;
```

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、emp 表の行ごとに計算します。相関副問合せは、emp 表の各行について次のステップを実行します。

1. 行の deptno を判断します。
2. deptno 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、emp 表の各行につき 1 回ずつ評価されます。

DUAL 表の例　次の文は、現在の日付を戻します。

```
SELECT SYSDATE FROM DUAL;
```

emp 表から簡単に SYSDATE を選択できますが、このとき、emp 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。

順序の例　次の文は、zseq 順序を増分し、新しい値を戻します。

```
SELECT zseq.nextval
    FROM dual;
```

次の文は、zseq の現在値を選択します。

```
SELECT zseq.currval
    FROM dual;
```

SET CONSTRAINT[S]

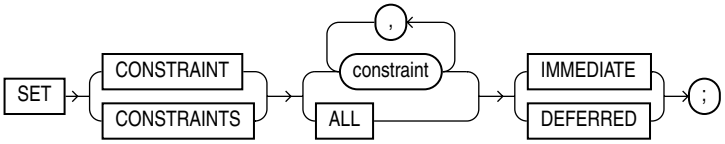
用途

SET CONSTRAINTS 文は、遅延可能な制約のチェックを、各 DML 文の実行後に行うか、トランザクションのコミット時に行うかをトランザクションごとに指定する場合に使用します。

前提条件

遅延可能な制約を確認する時期を指定する場合は、制約が適用される表に対する SELECT 権限を持っているか、またはその表が自スキーマ内にある必要があります。

構文



キーワードとパラメータ

constraint

1 つ以上の整合性制約の名前を指定します。

ALL

このトランザクション内のすべての遅延可能な制約を設定する場合に、ALL を指定します。

IMMEDIATE

遅延可能な制約によって指定された条件が、各 DML 文の直後にチェックされようにする場合に、IMMEDIATE を指定します。

DEFERRED

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チェックされるようにする場合に、DEFERRED を指定します。

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 文の直後にチェック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 つの遅延制約をチェックします。

```
SET CONSTRAINTS unq_name, scott.nn_sal,
  adams.pk_dept@dblink DEFERRED;
```

SET ROLE

用途

SET ROLE 文は、現行セッションのロールを有効化、無効化またはコンパイルする場合に使用します。

ユーザー・ログイン時に、Oracle は、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を使用可能にします。セッション中、ユーザーまたはアプリケーションはそのセッションに対して使用可能になっているロールを変更するために何度でも SET ROLE 文を使用できます。ただし、同時に使用可能にできるロールの数は、初期化パラメータ MAX_ENABLED_ROLES の値によって制限されます。

SESSION_ROLES データ・ディクショナリ・ビューを検索することにより、現在使用可能なロールを参照できます。

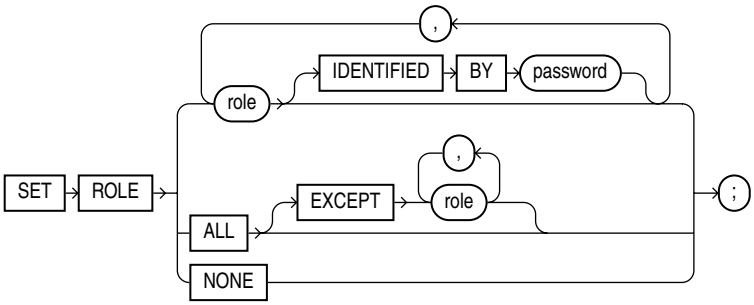
参照：

- ロールの作成の詳細は、9-141 ページの「CREATE ROLE」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、8-87 ページの「ALTER USER」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



キーワードとパラメータ

role

現行セッションで使用可能にするロールを指定します。指定されないロールは、現行セッションで使用禁止になります。

制限事項 : ロールは、直接的または他のロールを介して付与されていない限り、指定できません。

IDENTIFIED BY *password*	ロールの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。

ALL

現行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する場合に、ALL を指定します。ただし、EXCEPT 句に任意に指定されているロールは除きます。

制限事項 : また、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。

EXCEPT	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であってはいけません。 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。

NONE

現行セッションで、DEFAULT ロールを含むすべてのロールを使用禁止にする場合に、NONE を指定します。

例

ロールの設定例　次の文は、現行セッションのパスワード marigolds によって識別されるロール gardener を使用可能にします。

```
SET ROLE gardener IDENTIFIED BY marigolds;
```

次の文は、現行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、banker を除くロールをすべて使用可能にします。

```
SET ROLE ALL EXCEPT banker;
```

次の文は、現行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 文は、現行トランザクションを読取り専用または読み書きの両方を設定するときに、分離レベルを設定するか、または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場合に使用します。

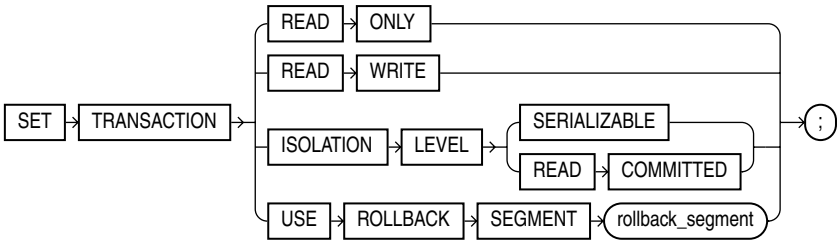
SET TRANSACTION 文によって実行される処理は、現行トランザクションのみに影響します。他のユーザーまたは他のトランザクションには影響しません。COMMIT 文または ROLLBACK 文を発行すると、トランザクションは終了します。なお、現行トランザクションは、データ定義文が実行される前後に暗黙的にコミットされます。

> **参照：** 8-131 ページの「COMMIT」および 11-83 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



キーワードとパラメータ

READ ONLY

READ ONLY 句は、現行トランザクションを読取り専用に設定します。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、そのトランザクション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便利です。

注意 : ユーザー SYS では、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。

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

参照 : 詳細は、『Oracle8*i* 概要』を参照してください。

READ WRITE

現行トランザクションを読み書き両用に設定する場合に、READ WRITE を指定します。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。

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

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する場合に、ISOLATION LEVEL 句を指定します。

SERIALIZABLE　SQL92 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、SERIALIAZBLE 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ていないトランザクションですでに更新されているリソースを更新すると、その DML 文は正常に実行されません。

注意 : SERIALIZABLE モードで運用する場合は、初期化パラメータ COMPATIBLE を 7.3.0 以上に設定してください。

READ COMMITTED　Oracle 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

現行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場合に、USE ROLLBACK SEGMENT を指定します。この句によって、現行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。

この句では、トランザクションのタイプごとに、異なるサイズのロールバック・セグメントを割り当てることができます。たとえば、次のように割り当てることができます。

- 実行時間が長い複数の問合せが、同時に同じ表を読み取っていない場合は、小さいトランザクションを、小さいロールバック・セグメントに割り当てることができます。ロールバック・セグメントが小さいと、メモリー内に保持される可能性が高くなります。
- 実行時間が長い複数の問合せによって同時に読み取られる表を変更するトランザクションには、大きいロールバック・セグメントを割り当てることができます。これは、読取り一貫性問合せのために必要なロールバック情報が、上書きされないようにするためです。
- 大量のデータを挿入、更新または削除するトランザクションは、そのトランザクションのロールバック情報を保持するための十分な大きさを持つロールバック・セグメントに割り当てることができます。

1 つの SET TRANSACTION 文または同じトランザクション内の異なる文に、READ ONLY 句および USE ROLLBACK SEGMENT 句は指定できません。読取り専用トランザクションはロールバック情報を生成しないため、ロールバック・セグメントは割り当てられません。

例

次の文は、ある企業が所有する船舶とコンテナの数を月末の夜中にカウントします。このレポートは、船舶やコンテナを追加または削除する他のユーザーの影響は受けません。

```
COMMIT;
SET TRANSACTION READ ONLY;
SELECT COUNT(*) FROM ship;
SELECT COUNT(*) FROM container;
COMMIT;
```

最初の COMMIT 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。

次の文は、ユーザーの現行トランザクションを、ロールバック・セグメント oltp_5 に割り当てます。

```
SET TRANSACTION USE ROLLBACK SEGMENT oltp_5;
```

storage_clause

用途

storage_clause は、次のスキーマ・オブジェクトの記憶特性を指定する場合に使用します。

- クラスタ
- 索引
- ロールバック・セグメント
- マテリアライズド・ビュー
- マテリアライズド・ビュー・ログ
- 表
- 表領域
- パーティション

記憶領域パラメータは、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。これらのパラメータの影響の詳細は、『Oracle8*i* パフォーマンスのための設計およびチューニング』を参照してください。

表領域の作成時に、記憶領域パラメータの値を指定できます。指定した値は、表領域に割り当てられたセグメントのデフォルト値になります。

表領域を変更する場合、記憶領域パラメータの値を変更できます。指定した値は、それ以降に割り当てられるセグメント（または、それ以降に作成されるオブジェクト）のデフォルト値になります。

注意： ローカル管理表領域では、*storage_clause* は異なって解析されます。作成時には、MAXEXTENTS は無視され、他のパラメータ値を使用してセグメントの初期サイズが計算されます。詳細は、10-56 ページの「CREATE TABLESPACE」を参照してください。

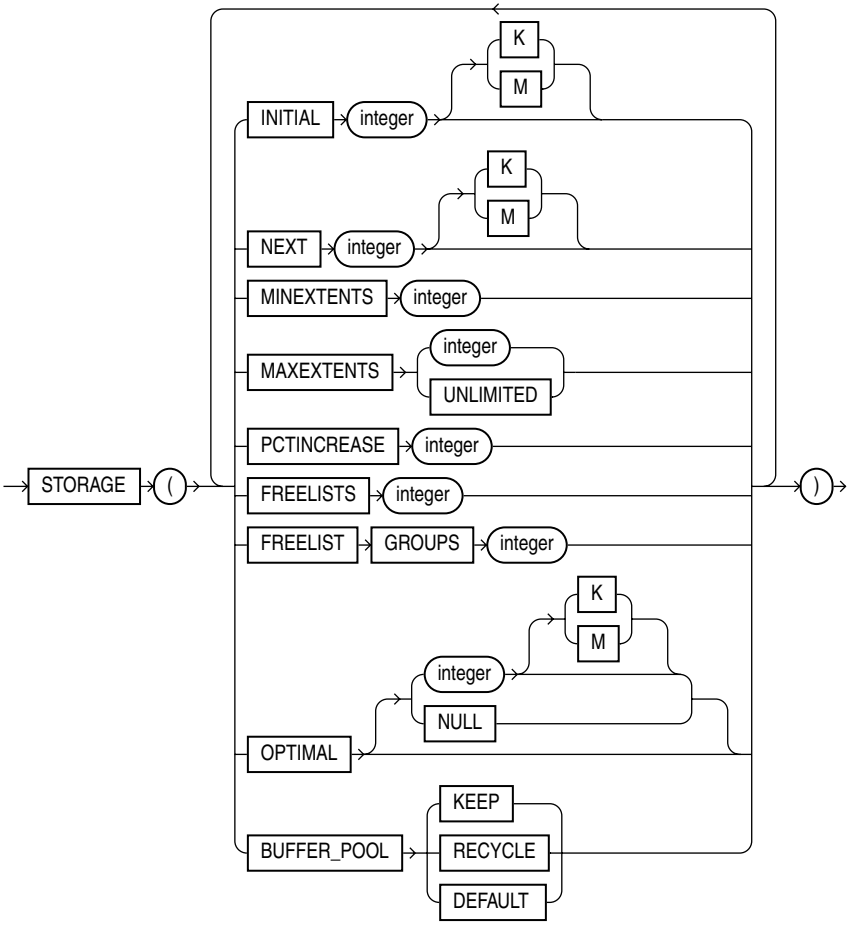
クラスタ、索引、ロールバック・セグメント、スナップショット、スナップショット・ログ、表またはパーティションを作成する場合、これらのオブジェクトに割り当てられるセグメントに、記憶領域パラメータの値を指定できます。記憶領域パラメータを指定しない場合、表領域に指定されている記憶領域パラメータの値が使用されます。

クラスタ、索引、ロールバック・セグメント、スナップショット、スナップショット・ログ、表またはパーティションを変更する場合、記憶領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。

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



キーワードとパラメータ

INITIAL

オブジェクトの第 1 エクステントのサイズをバイト単位で指定します。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。K または M を使用すると、KB または MB 単位で指定できます。

デフォルトのサイズは 5 データ・ブロックです。最小値は、ビットマップ化されていないセグメントの 2 データ・ブロック・サイズ、またはビットマップ化されたセグメントの 3 データ・ブロック・サイズに、指定した空きリスト・グループの各 1 データ・ブロック・サイズを加えた値になります。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。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 1 番近い倍数に丸められます。5 データ・ブロックより大きい値については、5 データ・ブロックの次の倍数まで切り上げられます。

制限事項 : ALTER 文では、INITIAL を指定できません。

> **参照 :** 空きリスト・グループの詳細は、11-133 ページの「FREELIST GROUPS」を参照してください。

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。K または M を使用して、KB または MB 単位で指定することもできます。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 1 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。

NEXT パラメータの値を変更した場合（ALTER 文で指定した場合）、次に割り当てられるエクステントのサイズは、直前に割り当てられたエクステントのサイズおよび PCTINCREASE パラメータの値とは関係なく、指定したサイズになります。

> **参照 :** 断片化の最小化の詳細は、『Oracle8*i* 概要』を参照してください。

PCTINCREASE

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 です。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 1 番近い倍数に切り上げられます。

また、PCTINCREASE パラメータの値を変更した場合（ALTER 文で指定した場合）、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 を 0（ゼロ）に設定して、SMON 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が極端に拡大することを防ぐことができます。

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

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。このパラメータを使用した場合、使用可能な領域が連続していない場合でも、オブジェクト作成時にたくさんの領域を割り当てることができます。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。

MINEXTENTS の値が 1 より大きい場合、INITIAL、NEXT および PCTINCREASE 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。

制限事項 :ALTER 文で MINEXTENTS は指定できません。

MAXEXTENTS

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。

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。

ただし、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。長時間にわたって挿入、更新または削除を続ける特殊なトランザクションでは、ディスクが一杯になるまで新規エクステントの作成を続けます。

注意： *storage_clause* を指定せずにロールバック・セグメントを作成した場合、そのロールバック・セグメントが作成された表領域と同じ記憶領域パラメータになります。そのため、MAXEXTENTS UNLIMITED で表領域を作成した場合、ロールバック・セグメントも同じデフォルト値になります。

FREELIST GROUPS

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 Parallel Server 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。
- 均一の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。

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。

参照： 『Oracle8*i* Parallel Server 概要』を参照してください。

FREELISTS

表領域以外のオブジェクトについて、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。

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 を指定することができます。

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。K または M を使用すると、KB または MB 単位で指定できます。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。

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 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。

OPTIMAL には、MINEXTENTS、INITIAL、NEXT および PCTINCREASE の各パラメータにによってロールバック・セグメントの最初に割り当てた領域より小さい値は指定できません。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 1 番近い倍数に丸められます。

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、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限り、表定義または索引定義のバッファ・プールを継承します。

注意 :　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場合、BUFFER_POOL は有効な句ではありません。

KEEP	I/O 操作を避けるために、メモリーのスキーマ・オブジェクトを保持する場合に、KEEP を指定します。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	不要なブロックをすぐにメモリーから排除する場合に、RECYCLE を指定します。これによって、オブジェクトが不要なキャッシュ領域を占領しなくなります。
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、『Oracle8*i* パフォーマンスのための設計およびチューニング』を参照してください。

例

記憶域属性を使用した表の作成 次の文は、表の作成時に記憶領域パラメータの値を指定します。

```
CREATE TABLE dept
    (deptno     NUMBER(2),
     dname      VARCHAR2(14),
     loc        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。

記憶域属性を使用したロールバック・セグメントの作成　次の文は、表の作成時に記憶領域パラメータの値を指定します。

```
CREATE ROLLBACK SEGMENT rsone
    STORAGE ( INITIAL  10K  NEXT 10K
              MINEXTENTS 2  MAXEXTENTS 25
               OPTIMAL 50K );
```

STORAGE パラメータの値に基づいて、ロールバック・セグメントに次のように領域が割り当てられます。

- MINEXTENTS の値は 2 のため、ロールバック・セグメント作成時にエクステントが 2 つ割り当てらます。
- INITIAL の値は 10KB のため、第 1 エクステントのサイズは 10KB となります。
- NEXT の値は 10KB のため、第 2 エクステントのサイズは 10KB となります。
- ロールバック・データが 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。ロールバック・セグメントの PCTINCREASE の値は常に 0（ゼロ）のため、第 3 エクステントのサイズは第 2 エクステントのサイズと同じ 10KB です。
- MAXEXTENTS の値は 25 のため、ロールバック・セグメントに割り当てられるエクステントの最大数は 25 になります。
- OPTIMAL の値は 50KB のため、ロールバック・セグメントが 50KB を超えた場合、エクステントの割当てが解除されます。割当てが解除されるエクステントは、アクティブでなくなったトランザクションのデータが入っているエクステントのみです。

TRUNCATE

注意： TRUNCATE 文はロールバックできません。

用途

TRUNCATE 文は、表またはクラスタのすべての行を削除し、STORAGE パラメータを表またはクラスタが作成されたときの値にリセットする場合に使用します。

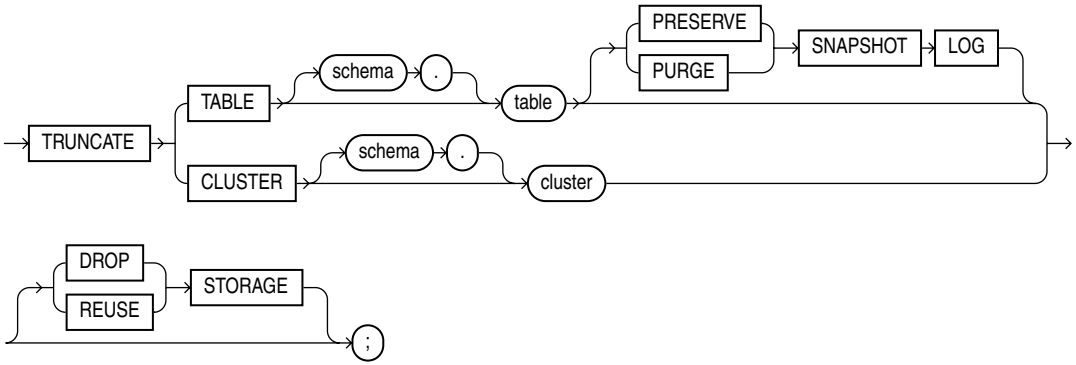
表を削除して再作成するより、TRUNCATE 文で行を削除する方が効果的です。表を削除して再作成した場合、その表に依存するオブジェクトが無効になり、表に対するオブジェクト権限を再度付与する必要があります。また、表の索引、整合性制約およびトリガーを再作成し、その記憶領域パラメータを再指定する必要があります。TRUNCATE の場合、このような影響はありません。

参照： 10-114 ページの「DELETE」、10-124 ページの「DROP CLUSTER」および 11-7 ページの「DROP TABLE」を参照してください。

前提条件

表またはクラスタを切り捨てるには、表またはクラスタが自スキーマ内にあるか、または DROP ANY TABLE システム権限が必要です。

構文



キーワードとパラメータ

TABLE

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、この表が自クラスタ内に存在するとみなされます。

- 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、現行セッションで作成された行のみが切り捨てられます。
- 表を切り捨てると、記憶領域パラメータ NEXT が、切捨てプロセス中にセグメントから最後に切り捨てら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 標識も、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。
- ドメイン・インデックスの場合は、この文が、適切な TRUNCATE ルーチンを起動し、ドメイン・インデックス・データを切り捨てます。

  参照：『Oracle8*i* データ・カートリッジ開発者ガイド』を参照してください。

- *table*（索引構成表および標準表）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、表中の索引データおよび表に対応付けされたマテリアライズド・ビュー・ダイレクト・ロードの INSERT 情報もすべて、自動的に削除されます（この情報は、マテリアライズド・ビューおよびスナップショット・ログのいずれにも依存していません）。このダイレクト・ロードの INSERT 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。

制限事項：

- 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参照整合性制約の親表は、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）。
- 表中の列に定義されたドメイン・インデックスに LOADING または FAILED のマークが付けられている場合は、その表を切り捨てることはできません。

SNAPSHOT LOG

SNAPSHOT LOG 句では、表が切り捨てられた場合に、この表に定義されているスナップショット・ログを保存するか、または削除するかを指定します。この句を使用した場合、スナップショット・マスター表を、エクスポート / インポートにより再編成できます。この場合、マスター表で定義された主キー・スナップショット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・スナップショットの連続高速リフレッシュをサポートする場合、スナップショット・ログに主キー情報を記録する必要があります。

PRESERVE　マスター表を切り捨てたときにスナップショット・ログを保存する場合は、PRESERVE を指定します。これはデフォルト値です。

PURGE　マスター表を切り捨てたときにスナップショット・ログを削除する場合は、PURGE を指定します。

参照： スナップショット・ログおよび TRUNCATE 文の詳細は、『Oracle8*i* レプリケーション・ガイド』を参照してください。

CLUSTER

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、この表が自クラスタ内に存在するとみなされます。

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

DROP STORAGE　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、表または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する場合に、DROP STORAGE を指定します。解放された領域は、表領域の他のオブジェクトに使用されます。これはデフォルト値です。

REUSE STORAGE　表またはクラスタに割り当てられた削除行から領域を確保する場合に、REUSE STORAGE を指定します。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、挿入または更新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。

注意：切り捨てるオブジェクトに対して、2 つ以上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 エクステントの始まりにリセットされます。

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。

例

簡単な TRUNCATE の例　次の文は、emp 表のすべての行を削除して、解放された領域を emp 表が定義されている表領域に戻します。

```
TRUNCATE TABLE emp;
```

ここでは、emp 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。

切捨て後の空き領域の保持　次の文は、cust クラスタ内の表のすべての行を削除しますが、表に割り当てられている領域はそのままにしておきます。

```
TRUNCATE CLUSTER cust REUSE STORAGE
```

この文では、cust の表にあるすべての索引データも削除されます。

切捨て後のマテリアライズド・ビューの保存　次の文は、スナップショット・ログを保存する TRUNCATE 文の使用例です。

```
TRUNCATE TABLE emp PRESERVE SNAPSHOT LOG;
TRUNCATE TABLE stock;
```

UPDATE

用途

UPDATE は、表またはビューのベース表の既存の値を変更する場合に使用します。

前提条件

表の値を更新する場合は、表が自スキーマ内にあるか、またはその表に対する UPDATE 権限が必要です。

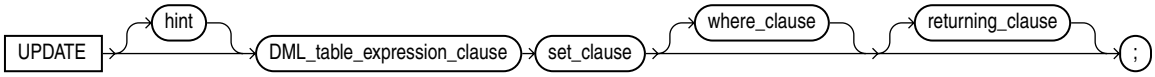
ビューのベース表の値を更新する場合は、次のことが必要です。

- そのビューに対する UPDATE 権限を持っている。
- そのビューが設定されているスキーマの所有者が、ベース表に対する UPDATE 権限を持っている。

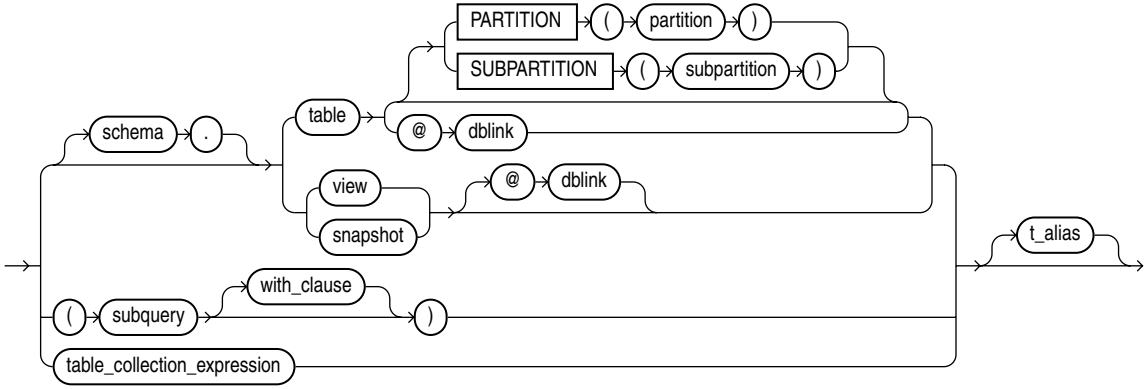
SQL92_SECURITY 初期化パラメータが TRUE に設定されている場合、UPDATE を実行するには、（*where_clause* の列などの）列値を参照している表に対する SELECT 権限が必要です。

UPDATE ANY TABLE システム権限がある場合は、任意の表または任意のベース表の値を更新できます。

構文

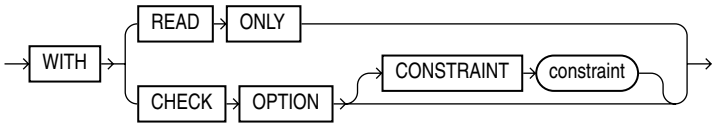


DML_table_expression_clause::=

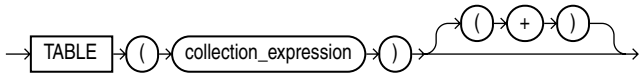


subquery: 11-88 ページの「SELECT および副問合せ」を参照してください。

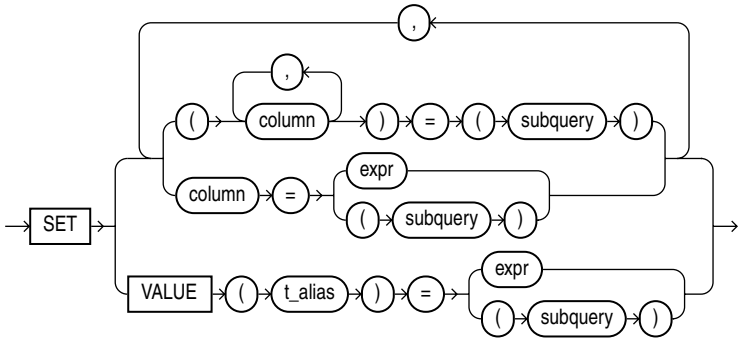
with_clause::=



table_collection_expression::=

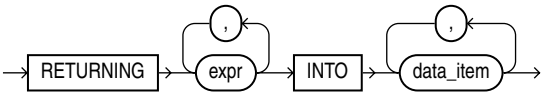


set_clause::=



where_clause::=

returning_clause::=



キーワードとパラメータ

hint

文に対して実行計画を選択する際の、オプティマイザへ指示を渡すコメントを指定します。

UPDATE キーワードに続けてパラレル・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、『Oracle8*i* パフォーマンスのための設計およびチューニング』および 2-65 ページの「ヒント」を参照してください。
- パラレル DML の詳細は、『Oracle8*i* パフォーマンスのための設計およびチューニング』、『Oracle8*i* Parallel Server 概要』および『Oracle8*i* 概要』を参照してください。

DML_table_expression_clause

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、表またはビューが自スキーマにあるとみなされます。

table | *view* | *subquery*
表またはビュー、または副問合せから戻された列の名前を指定します。指定した値は、更新する必要があります。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、Oracle はそのビューのベース表を更新します。

table（または *view* のベース表）に、1 列以上のドメイン・インデックスがある場合は、この文が適切な索引タイプの更新ルーチンを実行します。

参照 : これらのルーチンの詳細は、『Oracle8*i* データ・カートリッジ開発者ガイド』を参照してください。

PARTITION (*partition*) | SUBPARTITION (*subpartition*)
更新対象の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。

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 の分散機能を使用している場合に限り、データベース・リンクを使用して、リモート表またはリモート・ビューを更新できます。

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、2-88 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。

with_clause
副問合せを次のように制限する場合に、*with_clause* を使用します。

- WITH READ ONLY で、副問合せを更新禁止にすることを指定します。
- WITH CHECK OPTION で、副問合せに存在しない行を生成する表変更の禁止を指定します。

  参照 : 11-108 ページの「WITH CHECK OPTION の例」を参照してください。

***DML_table_expression_clause* に関する制限事項 :**

- *table*（または *view* のベース表）に、LOADING または FAILED とマークされたドメイン・インデックスがある場合は、この文は実行できません。
- *DML_query_expression_clause* の副問合せには *order_by_clause* を指定できません。
- ビューを定義する問合せに、INSTEAD OF トリガー以外の次のいずれかの要素が含まれる場合は、そのビューは更新できません。
  - 集合演算子
  - DISTINCT 演算子
  - 集計グループ関数
  - GROUP BY、ORDER BY、CONNECT BY または START WITH 句
  - SELECT リストのコレクション式
  - SELECT リストの副問合せ
  - 結合（一部の例外を除く）

    詳細は、『Oracle8*i* 管理者ガイド』を参照してください。
- 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。
- UNUSABLE のマークが付けられた索引、索引パーティションまたは索引サブパーティションを指定した場合、、SKIP_UNUSABLE_INDEXES パラメータが TRUE に設定されていない限り、UPDATE 文は正常に実行されません。

参照 : 7-101 ページの「ALTER SESSION」を参照してください。

table_collection_expression

table_collection_expression は、コレクション値の式を表として扱う必要がある場合に Oracle に通知します。*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。

collection_expression ネストした表の列を *table* または *view* から選択する副問合せを指定します。

注意 : 以前のリリースの Oracle では、*table_collection_expr* を「THE *subquery*」と表現していました。現在、このような表現方法はされていません。

---

t_alias

文内で参照する表、ビューまたは副問合せの**相関名**（別名）を指定します。

注意： *DML_query_expression_clause* がオブジェクト型の属性またはオブジェクト型のメソッドを参照する場合、この別名が必要です。

---

set_clause

set_clause によって、列の値を設定します。

column

更新する表またはビューの列の名前を指定します。*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。

制限事項：

- *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 を含む最初の行をロックしておく必要があります。

  参照：11-108 ページの「LOB ロックの例」を参照してください。

- *column* がパーティション表のパーティション・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、行の移動を可能にしない限り、UPDATE は正確に実行されません。

  参照：詳細は、10-7 ページの「CREATE TABLE」または 8-2 ページの「ALTER TABLE」にある *row_movement_clause* を参照してください。

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。

- *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。
- *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。

参照 : 11-88 ページの「SELECT および副問合せ」および 5-25 ページの「副問合せの使用」を参照してください。

注意 : この副問合せがリモート・オブジェクトを参照する場合、参照がローカル・データベース上のオブジェクトへループバックしない限り、UPDATE 操作をパラレルで実行できます。ただし、*DML_query_expression_clause* の副問合せがリモート・オブジェクトを参照する場合、UPDATE 操作はシリアルで実行されます。

参照 : 10-40 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値を変換する式を指定します。この式には、ホスト変数やオプションの標識変数を指定できます。

参照 : 構文については、5-2 ページの「式」を参照してください。

VALUE

VALUE 句によって、オブジェクト表の行全体を指定できます。

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。

参照 : 11-150 ページの「SET VALUE の例」を参照してください。

注意 : 後続の問合せ中に文字リテラルを RAW 列に挿入する場合、RAW 列にある索引は使用せずに、フル・テーブル・スキャンを行います。

where_clause

where_clause によって、指定した条件が TRUE 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。

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* の条件を満たす各行ごとに、*set_clause* の等号（=）の左側にある列に、その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。式は行が更新される場合に評価されます。

> **参照：** 条件の構文については、5-15 ページの「条件」を参照してください。

returning_clause

DML（INSERT、UPDATE または DELETE）文に影響される行を取り出します。この句は、表、スナップショット、および単一のベース表を持つビューに指定できます。

- 単一行で処理する場合は、*returning_clause* 付きの DML 文で、処理された行 ROWID および REF を使用して列式を取り出し、ホスト変数または PL/SQL 変数に格納します。
- 複数行で処理する場合は、*returning_clause* 付きの DML 文で、式の値、ROWID および処理された行に関連する REF をバインド配列に格納します。

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 句は、変更された行の値を、*data_item* リストに指定した変数に格納することを示します。
data_item	各 *data_item* は、取り出された *expr* 値を格納するホスト変数または PL/SQL 変数です。

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。

制限事項：

- パラレル DML またはリモート・オブジェクトでは、この句を使用できません。
- この句で 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は、この句を指定できません。

> **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、『Oracle8*i* PL/SQL 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。

例

簡単な例　次の文は、職種が trainee の従業員の賞与に NULL 値を指定します。

```
UPDATE emp
    SET comm = NULL
    WHERE job = 'TRAINEE';
```

次の文は、jones 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（jones は 1 人のみとします）。

```
UPDATE emp
    SET job = 'MANAGER', sal = sal + 1000, deptno = 20
    WHERE ename = 'JONES';
```

次の文は、データベース・リンク accounts を介してアクセスできるリモート・データベースの boston 表の銀行口座番号 5001 の残高を増額します。

```
UPDATE accounts@boston
    SET balance = balance + 500
    WHERE acc_no = 5001;
```

PARTITION 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 (feb96) s
   SET s.account_name = UPPER(s.account_name);
```

複雑な例　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 *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 emp a
    SET deptno =
        (SELECT deptno
            FROM dept
            WHERE loc = 'BOSTON'),
        (sal, comm) =
        (SELECT 1.1*AVG(sal), 1.5*AVG(comm)
          FROM emp b
          WHERE a.deptno = b.deptno)
    WHERE deptno IN
        (SELECT deptno
          FROM dept
          WHERE loc = 'DALLAS'
              OR loc = 'DETROIT');
```

この UPDATE 文によって、次の処理が実行されます。

- ダラスまたはデトロイトで働く従業員のみを更新します。
- これらの従業員の deptno に、ボストンの deptno を設定します。
- 各従業員の給与を、その部門の平均給与の 10% 引き上げます。
- 各従業員の賞与を、その部門の平均賞与の 50% 引き上げます。

SET VALUE の例　次の文は、異なるオブジェクト表 table2 の行を選択して、オブジェクト表 table1 の行を更新します。

```
UPDATE table1 p SET VALUE(p) =
   (SELECT VALUE(q) FROM table2 q WHERE p.id = q.id)
   WHERE p.id = 10;
```

この副問合せでは、式に value オブジェクト参照関数を使用します。

相関更新の例　次の文は、部門番号 123 の部門に関連するネストした表 projs の特定の行を更新します。

```
UPDATE TABLE(SELECT projs
           FROM dept d WHERE d.dno = 123)  p
  SET p.budgets = p.budgets + 1
  WHERE p.pno IN (123, 456);
```

RETURNING 句の例　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 emp
  SET job ='MANAGER', sal = sal + 1000, deptno = 20
  WHERE ename = 'JONES'
  RETURNING sal*0.25, ename, deptno INTO bnd1, bnd2, bnd3;
```

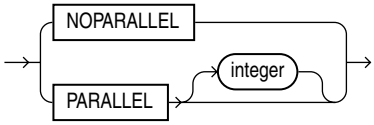
A

構文図

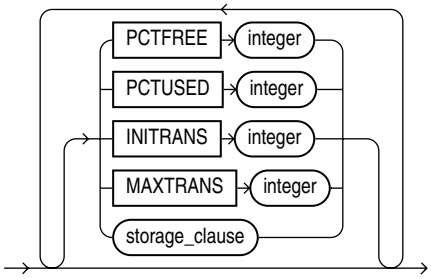
構文図とは、SQL の有効な構文を図で示したものです。構文図は、矢印が示す方向に左から右へ読んでください。

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、ユーザーが通る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。



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。次の例では、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。



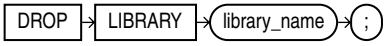
構文図で使用されるパラメータと、各文でそのパラメータに代入する値の例を次の表に示します。

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、2-77 ページの「スキーマ・オブジェクト」を参照してください。	emp
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、2-32 ページの「Text（テキスト）」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。	ename '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、5-15 ページの「条件」を参照してください。	ename >'A'
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-1994', 'DD-MON-YYYY')
expr	5-2 ページの「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。	sal + 1000
integer	2-33 ページの「Integer（整数）」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 2-33 ページの「Number（数）」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(sal) 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。11-88 ページの「SELECT および副問合せ」を参照してください。	SELECT ename FROM emp

パラメータ	説明	例
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	Net8 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 Net8 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

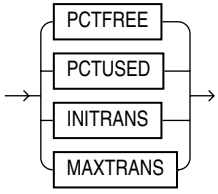
必須キーワードおよびパラメータは、単独で示されているか、または選択肢のリストとして縦に並べて書かれています。必須キーワードおよびパラメータが 1 つの場合は、メイン・パス、つまり現在選択しているパスの線上に書かれています。次の例では、*library_name* が必須パラメータです。



`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。

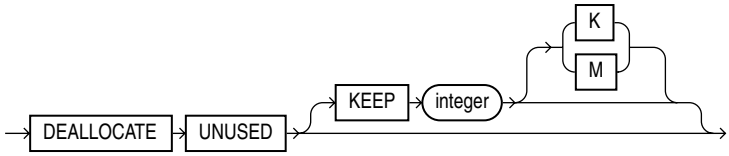
```
DROP LIBRARY hq_lib;
```

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。つまり、複数のキーワードまたはパラメータのうち、いずれか 1 つを選択する必要があります。ただし、メイン・パス上にあるものでなくてもかまいません。次の例では、4 つの設定値のうちの 1 つを選択する必要があります。



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、上のパスの方へ進んでも、続けてメイン・パスに沿って進んでもかまいません。

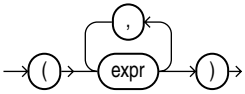


この図に従うと、次の文はすべて有効です。

```
DEALLOCATE UNUSED;
DEALLOCATE UNUSED KEEP 1000;
DEALLOCATE UNUSED KEEP 10M;
```

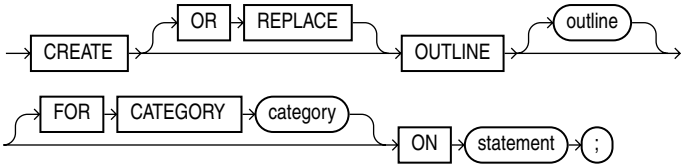
構文のループ

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、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。



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、2 つの部分に分割された構文図です。



この図に従うと、次の文は有効です。

```
CREATE OUTLINE ON UPDATE;
```

データベース・オブジェクト

表や列などの Oracle の識別子の名前は、長さが 30 文字以内に制限されています。先頭の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（#）、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、有効な文字であればどんな文字でも（空白を含む）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とき以外は、大文字、小文字を区別しません。

詳細は、2-81 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、業界標準規格を管理する団体によって確立される SQL-92 規格への Oracle の規格準拠について説明します。新しい SQL-99 規格を考慮して、この付録に変更が必要になる事項を評価しています。また、FIPS フラガーによる標準 SQL への拡張機能の位置付けについても説明します。

標準 SQL への規格準拠

この項では、次の組織によって確立される SQL 規格に対する Oracle の準拠性について説明します。

- 米国規格協会（ANSI）
- 国際標準化機構（ISO）
- アメリカ合衆国連邦政府

ANSI と ISO の規格準拠

Oracle8*i* は、ANSI 文書『X3.135-1992「Database Language SQL」』に定義されている基本レベルに準拠しています。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
1430 Broadway
New York, NY 10018 USA

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。Oracle Server、Oracle プリコンパイラ（C/C++ 用）リリース 8.1、Oracle プリコンパイラ（COBOL 用）リリース 8.1 および SOL*Module（ADA 用）リリース 8.0.4 は、次の ANSI X3.135-1992/ISO 9075-1992 規格に準拠しています。

- 基本レベルでの規格準拠（SQL-DDL および SQL-DML を含む）
- ADA 用モジュール言語
- SQL 埋込み C
- SQL 埋込み COBOL

Oracle は、基本レベルでは完全に規格に準拠しており、表 B-1 に示すように、変換レベル、中間レベル、完全レベルでも一部準拠しています（SQL-DDL と SQL-DML を含む）。

表 B-1 変換レベル、中間レベル、完全レベルでの規格準拠

レベル	SQL92 の機能（番号および名称）
変換	7. TRIM 関数
	8. ビューの UNION
	9. 暗黙の数値加算
	10. 暗黙の文字加算
	13. グループ操作
	14. SELECT 構文のリストでの修飾 *
	15. 小文字の識別子
	16. PRIMARY KEY の拡張
	18. 複数モジュールのサポート
	21. INSERT 式
中間	31. スキーマ定義文
	42. 各国文字
	48. 拡張 NULL 述語
完全	60. 後続アンダースコア
	62. 参照名の順番

FIPS の規格準拠

Oracle は、Entry レベルの SQL の FIPS PUB 127-2 に完全に準拠しています。また、「Section 16"Special Procurement Considerations"」に対して、次の情報が提供されています。

Section 16.2 Programming Language Interfaces（プログラミング言語インタフェース）

Oracle プリコンパイラでは、C および COBOL での埋込み SQL の使用がサポートされています。SQL*Module では、ADA でのモジュール言語の使用がサポートされています。

Section 16.3 Style of Language Interface（言語インタフェースのスタイル）

SQL*Module が付いた Oracle では、ADA 用のモジュール言語をサポートしています。
Oracle プリコンパイラが付いた Oracle では、C および COBOL をサポートしています。サポートされる言語は、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。

Section 16.5 Interactive Direct SQL（対話型直接 SQL）

Oracle8*i* では、SQL*Plus バージョン 3.1（および他の Oracle Tools）によって、次の SQL 文を直接起動でき、FIPS PUB 127-2 の要件を満たしています。

- `CREATE TABLE` 文
- `CREATE VIEW` 文
- `GRANT` 文
- `INSERT` 文
- `SELECT` 文（`ORDER BY` 句は付くが、`INTO` 句は付かない）
- `UPDATE` 文 : 検索
- `DELETE` 文 : 検索
- `COMMIT WORK` 文
- `ROLLBACK WORK` 文

このマニュアルで説明されている他のほとんどの SQL 文も、サポートされています。

Section 16.6 Sizing for Database Constructs（データベース要素のサイズ設定）

表 B-2 には、FIPS PUB 127-1 の中で規定される要件、およびこれらの要件に対する Oracle8*i* での設定値を示します。

表 B-2 データベース要素のサイズ設定

データベース要素	FIPS	Oracle8*i*
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 (a)	20	1000
行の長さ（b,c）	2,000	2,000,000
UNIQUE 制約の中の列	6	32
UNIQUE 制約の長さ (b)	120	（d）
外部キー列リストの長さ（b）	120	（d）
GROUP BY 句の中の列	6	255（e）
GROUP BY 列のリストの長さ	120	（e）
ORDER BY 句の中のソート指定	6	255（e）
ORDER BY 列のリストの長さ	120	（e）
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

表 B-2 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	Oracle8*i*
同時にオープンできるカーソル	10	(f)
SELECT 構文のリストの項目	100	1000

(a) 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。

(b) 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。

(c) 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))

(d) UNIQUE 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。

(e) Oracle は、GROUP BY 句内の列の数や ORDER BY 句内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 句内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。

(f) 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・システム上で利用可能なメモリーによって異なりますが、すべての場合において 100 を超えます。

Section 16.7 Character Set Support（キャラクタ・セットのサポート）

Oracle では、ほとんどのコンピュータ上で ASCII キャラクタ・セット（FIPS PUB 1-2）を、また、IBM メインフレーム・コンピュータ上で EBCDIC キャラクタ・セットをサポートしています。Oracle 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。

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、Entry SQL92 を使用するのと同じように、これらの拡張機能を使用できます。

他の SQL 処理系に対する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 プログラムの中で 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、『Oracle8*i* Pro*COBOL プリコンパイラ・プログラマーズ・ガイド』および『Oracle8*i* Pro*C/C++ プリコンパイラ・プログラマーズ・ガイド』を参照してください。

C

Oracle の予約語

この付録では、Oracle 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。

表 C-1 Oracle の予約語

ACCESS	CHAR *	DEFAULT *
ADD *	CHECK *	DELETE *
ALL *	CLUSTER	DESC *
ALTER *	COLUMN	DISTINCT *
AND *	COMMENT	DROP *
ANY *	COMPRESS	ELSE *
AS *	CONNECT *	EXCLUSIVE
ASC *	CREATE *	EXISTS
AUDIT	CURRENT *	FILE
BETWEEN *	DATE *	FLOAT *
BY *	DECIMAL *	FOR *
FROM *	NOT *	SHARE
GRANT *	NOWAIT	SIZE *
GROUP *	NULL *	SMALLINT *

表 C-1 Oracle の予約語（続き）

HAVING *	NUMBER	START
IDENTIFIED	OF *	SUCCESSFUL
IMMEDIATE *	OFFLINE	SYNONYM
IN *	ON *	SYSDATE
INCREMENT	ONLINE	TABLE *
INDEX	OPTION *	THEN *
INITIAL	OR *	TO *
INSERT *	ORDER *	TRIGGER
INTEGER *	PCTFREE	UID
INTERSECT *	PRIOR *	UNION *
INTO *	PRIVILEGES *	UNIQUE *
IS *	PUBLIC *	UPDATE *
LEVEL *	RAW	USER *
LIKE *	RENAME	VALIDATE
LOCK	RESOURCE	VALUES *
LONG	REVOKE *	VARCHAR *
MAXEXTENTS	ROW	VARCHAR2
MINUS	ROWID	VIEW *
MISLABEL	ROWNUM	WHENEVER *
MODE	ROWS *	WHERE
MODIFY	SELECT *	WITH *
NOAUDIT	SESSION *	
NOCOMPRESS	SET *	

索引

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

ち

て

と

ね

は

ひ

ふ

へ

ほ

ま

み

ゆ

よ

ら

り

る

れ

ろ