

## Oracle8 Server

SQL リファレンス : Vol.2

リリース 8.0

1998 年 2 月

部品番号 A56824-1

**ORACLE®**

---

The Database for Network Computing™

---

Oracle8 Server SQL リファレンス Vol.2、リリース 8.0

部品番号：A56824-1

第1版：1998年2月

原本名：Oracle8 SQL Reference, Release 8.0

原本部品番号：A58239-01

原著者：Diana Lorentz

協力者：Steve Bobrowski, Robert Jenkins, Susan Kotsovolos, Andre Kruglikov, Vishu Krishna, Muralidhar Krishnaprasa, Michael Kung, Paul Lane, Nina Lewis, Lefty Leverenz, Phil Locke, William Maimone, Mohammad Monajjemi, Rita Moran, Thomas Portfolio, Valarie Moore, Denis Raphaely, Richard Sarwal, Rick Wong

Copyright© 1997, 1998, Oracle Corporation. All rights reserved.

Printed In Japan

制限付権利の説明

プログラムの使用、複製、または開示は、オラクル社との契約に記された制約条件に従うものとします。

本書の情報は、予告なしに変更されることがあります。本書に問題を見つけたら、当社にコメントをお送りください。オラクル社は本書の無謬性を保証しません。

危険な用途への使用について

当社製品は、原子力、航空産業、大量輸送、又は医療の分野など、本質的に危険が伴うアプリケーションを用途として特に開発されていません。当社製品を上述のようなアプリケーションに使用することについての安全確保は顧客各位の責任と費用により行っていただきたく、万一かかる用途での使用によりクレームや損害が発生いたしましても、当社および開発元である米国 Oracle Corporation（その関連会社も含まれます。）は一切責任を負いかねます。

ORACLE は、Oracle Corporation の登録商標です。

本文中の他社の商品名は、それぞれ各社の商標または登録商標です。

---

---

# 目次

はじめに .....	XV
------------	----

## 第 1 章 概要

SQL の歴史 .....	1-1
SQL の規格 .....	1-1
埋込み SQL .....	1-3
字句規則 .....	1-4
サポートするツール .....	1-4

## 第 2 章 Oracle8 SQL の要素

リテラル .....	2-1
Text( テキスト ) .....	2-2
Integer( 整数 ) .....	2-3
Number ( 数 ) .....	2-3
データ型 .....	2-5
NULL .....	2-28
疑似列 .....	2-30
コメント .....	2-35
データベース・オブジェクト .....	2-40
スキーマ・オブジェクト名および修飾子 .....	2-43
スキーマ・オブジェクトと部分を参照する .....	2-47

## 第3章 演算子、関数、式、条件

演算子 .....	3-1
SQL 関数 .....	3-15
ユーザー関数 .....	3-57
書式モデル .....	3-60
式 .....	3-73
条件 .....	3-84

## 第4章 コマンド

SQL コマンドの概要 .....	4-2
データ定義言語 (DDL) コマンド .....	4-2
データ操作言語 (DML) コマンド .....	4-7
トランザクション制御コマンド .....	4-8
セッション制御コマンド .....	4-8
システム制御コマンド .....	4-9
埋込み SQL コマンド .....	4-10
ALTER CLUSTER .....	4-11
使用上の注意 .....	4-13
ALTER DATABASE .....	4-15
例 .....	4-23
ALTER FUNCTION .....	4-26
使用上の注意 .....	4-26
ALTER INDEX .....	4-28
例 .....	4-35
ALTER PACKAGE .....	4-38
使用上の注意 .....	4-38
ALTER PROCEDURE .....	4-41
使用上の注意 .....	4-41
ALTER PROFILE .....	4-43
パスワード履歴の使用 .....	4-45
例 .....	4-45
ALTER RESOURCE COST .....	4-48
使用上の注意 .....	4-48
ALTER ROLE .....	4-51
使用上の注意 .....	4-51



ALTER ROLLBACK SEGMENT .....	4-53
使用上の注意 .....	4-54
ALTER SEQUENCE .....	4-56
例 .....	4-57
ALTER SESSION .....	4-58
SQL トレース機能の使用可能と使用禁止の切替え .....	4-67
NLS パラメータの使用 .....	4-67
最適化の方法とモードの変更 .....	4-70
FIPS のフラグ使用 .....	4-71
セッション・カーソルのキャッシュ .....	4-71
パラレル・サーバーの別のインスタンスに接続している場合と同様にデータベースに アクセス .....	4-71
データベース・リンクのクローズ .....	4-72
インダウト分散トランザクションの強制処理 .....	4-72
プロシージャとストアード・ファンクションのトランザクション制御 .....	4-73
パラレル DML .....	4-74
ALTER SNAPSHOT .....	4-76
例 .....	4-81
ロールバック・セグメントの指定 .....	4-82
主キー・スナップショット .....	4-82
パーティション・スナップショット .....	4-83
ALTER SNAPSHOT LOG .....	4-84
物理属性の変更 .....	4-86
主キーおよび ROWID、フィルタ列の追加 .....	4-86
パーティション・スナップショット・ログ .....	4-87
ALTER SYSTEM .....	4-88
ログインの制限 .....	4-97
共有プールの消去 .....	4-97
チェックポイントの実行 .....	4-98
データ・ファイルのチェック .....	4-98
リソース制限の使用 .....	4-98
グローバル・ネーム変換 .....	4-99
マルチスレッド・サーバーのプロセスの管理 .....	4-99
ライセンス制限の使用 .....	4-100
REDO ログ・ファイル・グループの切替え .....	4-102
分散回復の使用可能と使用禁止の切替え .....	4-102
セッションの終了 .....	4-103

セッションの切断 .....	4-104
ALTER TABLE .....	4-105
列の追加 .....	4-121
列定義の変更 .....	4-121
索引構成表 .....	4-123
LOB 列 .....	4-124
ネストした表の列 .....	4-124
REF .....	4-125
表パーティションの変更 .....	4-126
ALTER TABLESPACE .....	4-131
使用上の注意 .....	4-137
ALTER TRIGGER .....	4-139
無効トリガー .....	4-139
トリガーの使用可能と使用禁止の切替え .....	4-140
ALTER TYPE .....	4-142
使用上の注意 .....	4-145
ALTER USER .....	4-148
デフォルト・ロールの設定 .....	4-150
認証方式の変更 .....	4-150
ALTER VIEW .....	4-152
使用上の注意 .....	4-152
ANALYZE .....	4-154
使用上の注意 .....	4-158
統計情報の収集 .....	4-158
クラスタ .....	4-160
統計情報の削除 .....	4-161
構造の検証 .....	4-161
連鎖行のリスト .....	4-163
ARCHIVE LOG 句 .....	4-164
使用上の注意 .....	4-166
AUDIT(SQL 文) .....	4-167
監査 .....	4-168
データベース・オブジェクトの文オプション .....	4-169
コマンドの文オプション .....	4-171
システム権限と文オプションのショートカット .....	4-173
AUDIT(スキーマ・オブジェクト) .....	4-175
オブジェクト・オプション .....	4-177

デフォルト監査 .....	4-178
COMMENT .....	4-180
使用上の注意 .....	4-180
COMMIT .....	4-182
使用上の注意 .....	4-183
トランザクションの終わり .....	4-184
CONSTRAINT 句 .....	4-185
整合性制約の定義 .....	4-189
NOT NULL 制約 .....	4-190
UNIQUE 制約 .....	4-190
PRIMARY KEY 制約 .....	4-192
参照整合性制約 .....	4-193
CHECK 制約 .....	4-197
DEFERRABLE 制約 .....	4-200
制約の使用可能と使用禁止の切替え .....	4-200
CREATE CLUSTER .....	4-202
使用上の注意 .....	4-205
クラスタ・キー .....	4-206
クラスタの種類 .....	4-206
クラスタ・サイズ .....	4-207
クラスタへの表の追加 .....	4-208
CREATE CONTROLFILE .....	4-210
使用上の注意 .....	4-213
CREATE DATABASE .....	4-214
例 .....	4-218
CREATE DATABASE LINK .....	4-220
使用上の注意 .....	4-222
現行ユーザーのデータベース・リンク .....	4-223
例 .....	4-223
CREATE DIRECTORY .....	4-226
ディレクトリ・オブジェクト .....	4-227
CREATE FUNCTION .....	4-228
例 .....	4-231
CREATE INDEX .....	4-233
使用上の注意 .....	4-238
索引の列 .....	4-239
単一の表に対する複数の索引 .....	4-239

NOSORT オプション .....	4-240
NOLOGGING .....	4-240
NULL .....	4-241
クラスタ索引の作成 .....	4-241
パーティション索引の作成 .....	4-241
ビットマップ索引の作成 .....	4-242
ネストした表の列に対する索引の作成 .....	4-242
CREATE LIBRARY .....	4-244
例 .....	4-245
CREATE PACKAGE .....	4-246
使用上の注意 .....	4-247
CREATE PACKAGE BODY .....	4-250
例 .....	4-251
CREATE PROCEDURE .....	4-255
使用上の注意 .....	4-258
CREATE PROFILE .....	4-261
プロファイルの使用方法 .....	4-264
分数での日数指定 .....	4-264
DEFAULT プロファイル .....	4-265
CREATE ROLE .....	4-268
ロールの使用方法 .....	4-269
Oracle によって定義されているロール .....	4-269
CREATE ROLLBACK SEGMENT .....	4-272
使用上の注意 .....	4-273
CREATE SCHEMA .....	4-275
使用上の注意 .....	4-275
CREATE SEQUENCE .....	4-278
順序の使用方法 .....	4-280
順序のデフォルト .....	4-281
順序値の増加 .....	4-281
順序番号のキャッシュ .....	4-281
順序値へのアクセス .....	4-282
CREATE SNAPSHOT .....	4-283
スナップショットについて .....	4-288
スナップショットの種類 .....	4-288
スナップショットのリフレッシュ .....	4-289
ロールバック・セグメントの指定 .....	4-291

主キーまたは ROWID スナップショットの指定 .....	4-292
パーティション・スナップショット .....	4-293
CREATE SNAPSHOT LOG .....	4-294
スナップショット・ログの使用方法 .....	4-296
主キーおよび ROWID、フィルタ列の記録 .....	4-297
CREATE SYNONYM .....	4-299
シノニムの使用方法 .....	4-300
シノニムの有効範囲 .....	4-301
CREATE TABLE .....	4-303
例 .....	4-318
LOB 列の例 .....	4-320
索引構成表 .....	4-320
パーティション表 .....	4-321
オブジェクト表 .....	4-321
ネストした表の格納 .....	4-322
REF .....	4-322
CREATE TABLESPACE .....	4-325
使用上の注意 .....	4-327
CREATE TRIGGER .....	4-330
トリガーの使用方法 .....	4-333
条件述語 .....	4-334
トリガーの構成要素 .....	4-335
トリガー・タイプ .....	4-336
スナップショット・ログ・トリガー .....	4-337
INSTEAD OF トリガー .....	4-339
ユーザー定義の型および LOB、REF 列 .....	4-340
CREATE TYPE .....	4-342
不完全オブジェクト型 .....	4-348
コンストラクタ .....	4-349
CREATE TYPE BODY .....	4-350
使用上の注意 .....	4-352
CREATE USER .....	4-353
オペレーティング・システムを介したユーザーの検証 .....	4-356
ネットワークを介したユーザーの検証 .....	4-356
ユーザーに対する表領域の割当て制限の設定 .....	4-356
ユーザーに対する権限付与 .....	4-356

CREATE VIEW .....	4-359
ビューの使用方法 .....	4-362
ビューの問合せ .....	4-362
結合ビュー .....	4-363
パーティション・ビュー .....	4-365
例 .....	4-365
オブジェクト・ビュー .....	4-366
DEALLOCATE UNUSED 句 .....	4-368
使用上の注意 .....	4-368
DELETE .....	4-370
DELETE の使用方法 .....	4-372
1 つのパーティションからの削除 .....	4-374
RETURNING 句 .....	4-374
DISABLE 句 .....	4-375
使用上の注意 .....	4-376
DROP 句 .....	4-379
使用上の注意 .....	4-379
DROP CLUSTER .....	4-381
使用上の注意 .....	4-381
DROP DATABASE LINK .....	4-383
使用上の注意 .....	4-383
例 .....	4-383
DROP DIRECTORY .....	4-384
使用上の注意 .....	4-384
DROP FUNCTION .....	4-385
使用上の注意 .....	4-385
DROP INDEX .....	4-387
使用上の注意 .....	4-387
DROP LIBRARY .....	4-388
例 .....	4-388
DROP PACKAGE .....	4-389
使用上の注意 .....	4-389
DROP PROCEDURE .....	4-391
使用上の注意 .....	4-391
DROP PROFILE .....	4-393
使用上の注意 .....	4-393

DROP ROLE .....	4-394
使用上の注意 .....	4-394
DROP ROLLBACK SEGMENT .....	4-395
使用上の注意 .....	4-395
DROP SEQUENCE .....	4-397
使用上の注意 .....	4-397
DROP SNAPSHOT .....	4-399
使用上の注意 .....	4-399
DROP SNAPSHOT LOG .....	4-400
使用上の注意 .....	4-400
DROP SYNONYM .....	4-401
使用上の注意 .....	4-401
DROP TABLE .....	4-402
使用上の注意 .....	4-402
DROP TABLESPACE .....	4-404
使用上の注意 .....	4-404
DROP TRIGGER .....	4-406
使用上の注意 .....	4-406
DROP TYPE .....	4-407
使用上の注意 .....	4-407
DROP TYPE BODY .....	4-409
使用上の注意 .....	4-409
DROP USER .....	4-410
使用上の注意 .....	4-410
DROP VIEW .....	4-412
使用上の注意 .....	4-412
ENABLE 句 .....	4-414
制約の有効化および無効化 .....	4-416
Oracle による整合性制約の検証方法 .....	4-417
例外の検出方法 .....	4-418
トリガーの使用可能化 .....	4-420
EXPLAIN PLAN .....	4-422
EXPLAIN PLAN の使用方法 .....	4-423
EXPLAIN PLAN およびパーティション表 .....	4-425
EXPLAIN PLAN およびパラレル DML .....	4-427
Filespec .....	4-428
例 .....	4-430

GRANT( システム権限とロール ) .....	4-432
使用上の注意 .....	4-433
ADMIN OPTION の付与 .....	4-439
その他の認可メソッド .....	4-439
例 .....	4-440
GRANT( オブジェクト権限 ) .....	4-442
使用上の注意 .....	4-444
シノニム権限 .....	4-447
ディレクトリ権限 .....	4-447
例 .....	4-447
INSERT .....	4-449
VALUES 句および副問合せ .....	4-452
パラレル DML .....	4-452
ビューへの挿入 .....	4-452
RETURNING 句 .....	4-453
例 .....	4-453
LOCK TABLE .....	4-455
使用上の注意 .....	4-456
NOAUDIT(SQL 文) .....	4-458
使用上の注意 .....	4-459
NOAUDIT( スキーマ・オブジェクト ) .....	4-460
例 .....	4-461
PARALLEL 句 .....	4-462
使用上の注意 .....	4-463
非パーティション表および非パーティション索引 .....	4-463
パーティション表およびパーティション索引 .....	4-464
例 .....	4-464
RECOVER 句 .....	4-466
例 .....	4-468
RENAME .....	4-470
オブジェクトの改名 .....	4-470
使用上の注意 .....	4-470
REVOKE( システム権限とロール ) .....	4-472
権限の取消し .....	4-472
ロールの取消し .....	4-473
使用上の注意 .....	4-473
例 .....	4-473



REVOKE( スキーマ・オブジェクト権限 ) .....	4-475
使用上の注意 .....	4-477
FORCE の使用 .....	4-477
複数の同じ権限の取消し .....	4-477
取消しのカスケード .....	4-477
例 .....	4-478
ROLLBACK .....	4-481
使用上の注意 .....	4-481
分散トランザクション .....	4-482
SAVEPOINT .....	4-484
使用上の注意 .....	4-484
SELECT .....	4-486
単純な問合せの作成 .....	4-491
階層問合せ .....	4-492
GROUP BY 句 .....	4-496
HAVING 句 .....	4-497
UNION および UNION ALL、INTERSECT、MINUS .....	4-498
ORDER BY 句 .....	4-498
FOR UPDATE 句 .....	4-499
結合 .....	4-501
SET CONSTRAINT(S) .....	4-509
例 .....	4-509
SET ROLE .....	4-511
権限ドメイン .....	4-512
例 .....	4-513
SET TRANSACTION .....	4-514
読取り専用トランザクションの設定 .....	4-515
ロールバック・セグメントへのトランザクションの割当て .....	4-516
STORAGE 句 .....	4-518
使用上の注意 .....	4-521
ロールバック・セグメントと MAXEXTENTS UNLIMITED .....	4-522
例 .....	4-522
副問合せ .....	4-525
使用上の注意 .....	4-527
フラット化した副問合せの使用 .....	4-528
相関副問合せ .....	4-528
DUAL 表からの選択 .....	4-530

順序の使用 .....	4-530
分散問合せ .....	4-530
TRUNCATE .....	4-532
使用上の注意 .....	4-533
制限事項 .....	4-534
例 .....	4-534
UPDATE .....	4-536
ビューの更新 .....	4-539
パーティション表の更新 .....	4-540
相関更新 .....	4-541
RETURNING 句 .....	4-542

## 付録 A 構文図

## 付録 B Oracle と標準 SQL

## 付録 C Oracle の予約語とキーワード

## 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 (SQL) について詳しく説明します。

Oracle SQL は、米国規格協会 (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。

SQL を拡張したオラクル社のプロシージャ型言語である PL/SQL については、『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 の埋込み SQL については、『Pro\*C/C++ プリコンパイラ・プログラマーズ・ガイド』および『SQL\*Module for Ada Programmer's Guide』、『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。

## 特徴と機能

『Oracle8 Server SQL リファレンス』には、Oracle8 および Oracle8 Enterprise Edition 製品の特徴と機能の説明があります。Oracle8 および Oracle8 Enterprise Edition の基本的な機能は同じです。ただし、Enterprise Edition だけで利用可能な拡張機能がいくつかあります。また、さらにオプションが必要な場合もあります。たとえば、CREATE TYPE コマンドを使用する場合には、Enterprise Edition と Object Option を持っていなければなりません。

Oracle8 と Oracle8 Enterprise Edition の違いや、ご使用の Oracle で利用できる機能とオプションについては、『Oracle8 と Oracle8 Enterprise Edition の解説』を参照してください。

## 対象読者

このマニュアルは、Oracle SQL のすべてのユーザーを対象とします。

---

## このマニュアルの構成

このマニュアルの構成は次のとおりです。

### 第 1 部

#### 第 1 章「概要」

SQL について定義し、その歴史と SQL を使用したリレーショナル・データベースへのアクセスの利点について説明します。

#### 第 2 章「Oracle8 SQL の要素」

Oracle データベースと Oracle SQL の要素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、関数、式、条件」

SQL の演算子と関数を、式や条件の中で組み合わせて使用方法について説明します。

### 第 2 部

#### 第 4 章「コマンド」

すべての SQL コマンドを、アルファベット順に説明します。

#### 付録 A「構文図」

このマニュアルで使用する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 の規格に対する Oracle の準拠性について説明し、これらの標準規格を拡張した部分を示します。

#### 付録 C「Oracle の予約語とキーワード」

Oracle の予約語とキーワードを示します。

## このマニュアルで使用する表記上の規則

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。項目は次のとおりです。

- 本文
- 構文図と表記法
- コード例
- 例題のデータ

### 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

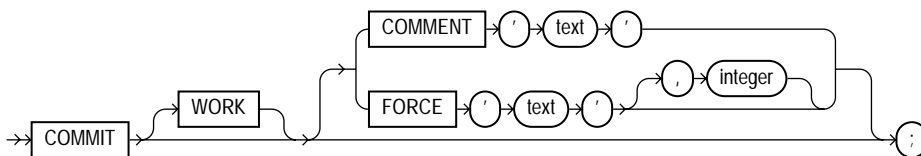
アルファベットの大文字は、SQL コマンド、キーワード、ファイル名、初期化パラメータを示しています。

イタリック

イタリック体の文字は、用語の定義と SQL コマンドのパラメータを示しています。

### 構文図と表記法

**構文図** . このマニュアルでは、第 4 章「コマンド」における SQL コマンドの説明や、第 2 章「Oracle8 SQL の要素」および第 3 章「演算子、関数、式、条件」におけるその他の SQL 言語の要素の説明に、構文図を使用します。これらの構文図では、次のように、線と矢印を使用して構文構造を示しています。



このタイプの構文図になじみが薄い場合は、付録 A「構文図」を参照して、読み方を理解してください。この項では、構文図のコンポーネントと、SQL 文の書き方の例を示します。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。大文字小文字の違いを除いて、キーワードは構文図に表記されているとおりに SQL 文で使用しなければなりません。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目を果たします。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・オブジェクト名または Oracle データ型名、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で置き換えます。たとえば、CREATE TABLE 文を作成する場合、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。

このマニュアルの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に示します。

パラメータ	説明	例
<i>table</i>	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なければなりません。オブジェクト型の一覧は「スキーマ・オブジェクト」(2-40 ページ) を参照してください。	emp
<i>c</i>	使用しているデータベースのキャラクタ・セットの単一文字で置き換えなければなりません。	T S
<i>'text'</i>	一重引用符で囲んだテキスト文字列で置き換えなければなりません。text' の構文については、「Text(テキスト)」(2-2 ページ) を参照してください。	'Employee records'
<i>char</i>	データ型 CHAR または VARCHAR2 の式か、一重引用符で囲んだ文字リテラルで置き換えなければなりません。	ename 'Smith'
<i>condition</i>	TRUE または FALSE に評価される条件で置き換えなければなりません。condition の構文については、「条件」(3-84 ページ) を参照してください。	ename > 'A'
<i>date</i> <i>d</i>	日付定数または DATE データ型の式で置き換えなければなりません。	TO_DATE( '01-Jan-1994', 'DD-MON-YYYY')
<i>expr</i>	「式」(3-73 ページ) 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
<i>integer</i>	「Integer(整数)」(2-3 ページ) 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なければなりません。	72

パラメータ	説明	例
<i>label</i>	MLSLABEL データ型の式で置き換えなければなりません。このような式については、使用している Trusted Oracle のマニュアルを参照してください。	TO_LABEL( 'SENSITIVE:ALPHA')
<i>number</i>	NUMBER データ型の式、または	AVG(sal)
<i>m</i>	「Number ( 数 )」(2-3 ページ)にある <b>number</b>	15 * 7
<i>n</i>	の構文の説明で定義されている数値定数で置き換えなければなりません。	
<i>raw</i>	RAW データ型の式で置き換えなければなりません。	HEXTORAW('7D')
<i>rowid</i>	ROWID データ型の式で置き換えなければなりません。	AAAAZzAABAAABrXAAA
<i>subquery</i>	他の SQL 文の中で使われる SELECT 文で置き換えなければなりません。「副問合せ」(4-525 ページ)を参照してください。	SELECT ename FROM emp
<i>:host_variable</i>	埋込み SQL プログラムで宣言されている変数の名前で置き換えなければなりません。このマニュアルでは、特定のデータ型を表すために、:host_integer と :d も使用しています。	:employee_number
<i>cursor</i>	埋込み SQL プログラム内のカーソルの名前で置き換えなければなりません。	curs1
<i>db_name</i>	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の名前で置き換えなければなりません。	sales_db
<i>db_string</i>	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なければなりません。詳細は、使用している Net8 プロトコルのユーザーズ・ガイドを参照してください。	
<i>statement_name</i>	SQL 文または PL/SQL ブロックに対する識別子で置き換えなければなりません。	s1
<i>block_name</i>		lab1

## コード例

このマニュアルには、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する。
- 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する。ただし、本文では大文字で表記されています。

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が、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 で書かれているので、セミコロン (;) で終了しています。しかし、これらの例文を使う場合には、使用している Oracle Tool ごとに規定された特殊文字を使って文を終了させなければなりません。

## 例題のデータ

このマニュアルの例題の多くでは、サンプルの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、このスクリプトの名前は UTLSAMPLE.SQL となっていますが、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、サンプルのユーザーを作成し、ユーザー SCOTT 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

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。

---

構造化問合せ言語 (SQL) は、Oracle データベース内のデータにアクセスするために、すべてのプログラムとユーザーが使用しなければならない一連のコマンドです。アプリケーション・プログラムや Oracle のツールを使用すれば、SQL を直接使用しなくてもデータベースにアクセスできることが少なくありません。しかし、アプリケーションがユーザーの要求を実行するときには必ず SQL を使用します。この章では、多くのリレーショナル・データベース・システムに採用されている SQL の背景について説明します。説明する内容は次のとおりです。

- SQL の歴史
- SQL の規格
- 埋込み SQL
- 字句規則
- サポートするツール

## SQL の歴史

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

## SQL の規格

Oracle SQL は、業界標準に準拠しています。オラクル社は、SQL 標準化委員会の主要メンバーとして積極的に標準化の活動に関わっていくことで、発展し続ける SQL 規格に今後も対応していきます。業界で認知されている委員会には、米国規格協会 (ANSI) およびは国際

電気標準会議 (IEC) が電気・電子部門を担当している国際標準化機構 (ISO) があります。ANSI と ISO/IEC はともに、SQL をリレーショナル・データベースの標準言語として認めています。新しい SQL 規格がこの両機関から同時に発表された場合、その規格の名前は、各機関の規則に従って命名されますが、技術的な詳細はまったく同じです。

ANSI および ISO が発表した最新の SQL 規格は、SQL-92( または SQL2) と呼ばれます。新規格の正式名称は、次のとおりです。

- ANSI X3.135-1992、「Database Language SQL」
- ISO/IEC 9075:1992、「Database Language SQL」

SQL-92 では、規格への準拠に基本、変換、中間、完全の 4 レベルを定義しています。SQL 規格に準拠する処理系は、少なくとも基本レベルの SQL に準拠していなければなりません。Oracle8 リリース 8.0 は、基本レベルの SQL に完全に準拠していますが、変換レベルまたは中間レベル、完全レベルに準拠する多くの機能もあります。

Oracle8 が基本レベルの SQL-92 に準拠していることは、米国連邦情報・技術局 (NIST) が連邦情報処理標準 (FIPS) の FIPS PUB 127-2 を用いて試験済みです。

**追加情報：** Oracle および SQL の詳細は、付録 B 「Oracle と標準 SQL」を参照してください。

## SQL の特長

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

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

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

SQL コマンドを使って、次の処理を行うことができます。

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

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

## すべてのリレーショナル・データベースに共通の言語

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

## 埋込み SQL

埋込み SQL とは、プロシージャ型プログラミング言語に埋め込んで使用する標準の SQL コマンドのことです。埋込み SQL コマンドは、Oracle プリコンパイラ関連のマニュアル、『SQL\*Module for Ada Programmer's Guide』、『Pro\*C/C++ プリコンパイラ・プログラマーズ・ガイド』および『Pro\*COBOL プリコンパイラ・プログラマーズ・ガイド』に記載されています。

埋込み SQL は以下のコマンドの集まりです。

- 対話形式のツールを使って SQL で使用できる全 SQL コマンド (SELECT、INSERT など)。
- 動的 SQL 実行コマンド (PREPARE、OPEN など)。プロシージャ型プログラミング言語に標準 SQL コマンドを統合します。

埋込み SQL には標準 SQL コマンドの拡張機能も含まれています。埋込み SQL は Oracle プリコンパイラによってサポートされます。Oracle プリコンパイラによって、埋込み SQL 文が解釈され、プロシージャ型言語のコンパイラが処理できる文に変換されます。

次の各 Oracle プリコンパイラによって、埋込み SQL のプログラムは異なるプロシージャ型言語に変換されます。

- Pro\*C/C++ プリコンパイラ
- Pro\*COBOL プリコンパイラ
- Pro\*FORTRAN プリコンパイラ
- SQL\*Module、ADA

**追加情報：** Oracle プリコンパイラおよび埋込み SQL コマンドの定義については、『SQL\*Module for Ada Programmer's Guide』、『Pro\*C/C++ プリコンパイラ・プログラマーズ・ガイド』および『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。

## 字句規則

SQL 文の記述に関する以下の字句規則は、Oracle の SQL インプリメンテーションに対してだけ適用されますが、他の SQL インプリメンテーションにも一般的に適用されます。

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

```
SELECT ENAME,SAL*12,MONTHS_BETWEEN(HIREDATE,SYSDATE) FROM EMP;
```

```
SELECT ENAME,
       SAL * 12,
       MONTHS_BETWEEN( HIREDATE, SYSDATE )
FROM EMP;
```

予約語およびキーワード、識別子、パラメータは大文字と小文字を区別せずに記述できます。テキスト・リテラルと引用符で囲んだ名前では大文字と小文字は区別されます。構文の説明は、「Text( テキスト)」（2-2 ページ）を参照してください。

## サポートするツール

Oracle の提供するツールの大部分（すべてではありません）が、Oracle の SQL の機能をすべてサポートしています。このマニュアルでは、SQL の全機能について記述します。オラクル提供のツールでサポートしていない機能がある場合、『PL/SQL ユーザーズ・ガイドおよびリファレンス』などのツールについて記述したマニュアルにその制限が示されていますので、参照してください。

---

# Oracle8 SQL の要素

この章には、Oracle SQL の基本要素に関する参照情報が記載されています。

---

**注意：** **OBJ** が前についているコマンドと説明は、Oracle Object Option がデータベース・サーバーにインストールされている場合だけ有効です。

---

第4章「コマンド」で説明するコマンドを使用する前に、この章で説明する次の概念を理解しておく必要があります。

- リテラル
- Text( テキスト )
- Integer( 整数 )
- Number ( 数 )
- データ型
- NULL
- 疑似列
- コメント
- データベース・オブジェクト
- スキーマ・オブジェクト名および修飾子
- スキーマ・オブジェクトと部分を参照する

## リテラル

リテラルと定数値という用語は同義であり、固定データ値のことを表しています。たとえば、'JACK' および 'BLUE ISLAND'、'101' はすべて文字リテラルです。なお、文字リテラル

は、単一引用符で囲みます。単一引用符を付けると、**Oracle** は文字リテラルとスキーマ・オブジェクト名を区別します。

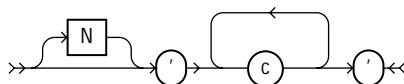
多くの **SQL** 文と関数では、文字リテラルと数値リテラルを指定する必要があります。式と条件の一部として、リテラルを指定できます。文字リテラルは *'text'* の表記法を、各国文字リテラルは *N'text* の表記法を、数値リテラルはリテラルのコンテキストにより *integer* または *number* の表記法を使用して指定できます。これらの表記法の構文については、次の項で説明します。

## Text( テキスト )

テキスト・リテラルまたは文字リテラルを指定します。このマニュアルの他の箇所でも、式および条件、**SQL** 関数、**SQL** コマンドに示されている *'text'* (テキスト) や *char* (文字) に値を指定するときには、必ずこの表記法を使用してください。

*text* (テキスト) の構文は次の通りです。

**text::=**



ここで、それぞれの記号の意味は次のとおりです。

- |    |   |
|----|---|
| N  | 各国キャラクタ・セットを使ってリテラルを指定します。この表記法を使って入力したテキストは、使用時に <b>Oracle</b> によって各国キャラクタ・セットに変換されます。 |
| c  | 単一引用符 (') を除く、データベースのキャラクタ・セットの任意の要素です。   |
| '' | テキスト・リテラルの始まりと終わりを示す 2 つの単一引用符です。リテラル内で単一引用符を表すには、単一引用符を 2 つ使用します。                      |

テキスト・リテラルは、単一引用符で囲んでください。このマニュアルでは、テキスト・リテラルと文字リテラルは同じ用語として使用しています。

テキスト・リテラルは、次のように **CHAR** データ型と **VARCHAR2** データ型の両方の特性を持ちます。

- 式と条件の中のテキスト・リテラルは、**Oracle** によりデータ型 **CHAR** として扱われ、空白を埋めて長さを揃えた後で比較される。
- テキスト・リテラルの最大長は 4000 バイト。

有効なテキスト・リテラルの例は次のとおりです。

```
'Hello'
'ORACLE.dbs'
```



```
'Jackie''s raincoat'
'09-MAR-92'
N'nchar literal'
```

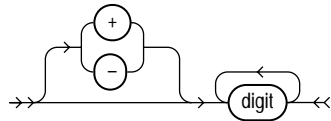
詳細は、「式」(3-73 ページ)にある *expr* の構文の説明を参照してください。

## Integer( 整数 )

このマニュアルの他の箇所で、式および条件、SQL 関数、SQL コマンドに示されている *integer*( 整数 ) に値を指定するときには、必ずこの表記法を使用してください。

*integer*( 整数 ) の構文は次のとおりです。

**integer::=**



ここで、それぞれの記号の意味は次のとおりです。

*digit*                    0、1、2、3、4、5、6、7、8、9 の 1 つです。

整数は最大 38 桁の精度を記憶できます。

有効な *integers*( 整数 ) の例は次のとおりです。

```
7
+255
```

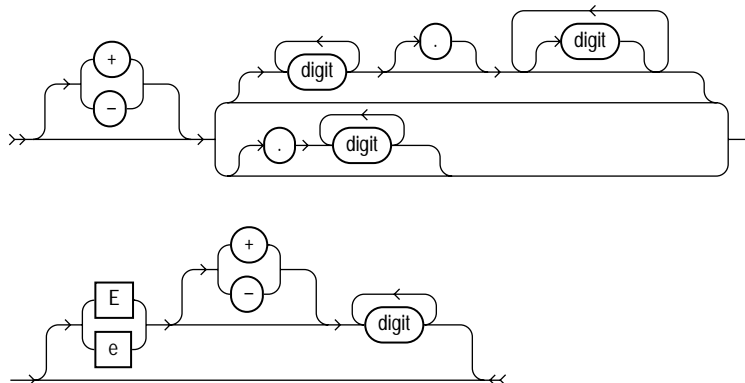
詳細は、「式」(3-73 ページ)にある *expr* の構文の説明を参照してください。

## Number ( 数 )

このマニュアルの他の箇所で、式および条件、SQL 関数、SQL コマンドに示されている *number*( 数 ) に値を指定するときには、必ずこの表記法を使用してください。

*number*( 数 ) の構文は次のとおりです。

**number::=**



ここで、それぞれの記号の意味は次のとおりです。

- +, -                    正の値または負の値を示します。符号を指定しない場合、正の値がデフォルトです。
- digit*                0、1、2、3、4、5、6、7、8、9の中の1つです。
- e, E                    数が科学表記法で指定されることを示します。Eの後の数字が指数を指定します。指数は-130から125までの範囲で指定します。

*number*(数)は最大38桁の精度を記憶できます。

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

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

```
'5,123'
```

このパラメータの詳細は、『Oracle8 Server リファレンス』を参照してください。

有効な *number*(数)の例は、次のとおりです。

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

詳細は、「式」(3-73 ページ)にある *expr* の構文の説明を参照してください。

# データ型

Oracle によって操作されるリテラル値や列値は、それぞれデータ型を持っています。値のデータ型は、値に対して固定されたある特性を持つ集合を対応付けます。これらの特性に応じて、Oracle はあるデータ型の値に対して別のデータ型の値とは異なる扱いをします。たとえば、NUMBER データ型の値は加算できますが、RAW データ型の値は加算できません。

表やクラスタを作成するとき、その列のそれぞれについて内部データ型を指定しなければなりません。プロシージャやストアド・ファンクションを作成するとき、その各引数にデータ型を指定しなければなりません。データ型により各列が含むことのできる値のドメインまたは各引数が持つことのできる値のドメインが規定されます。たとえば、DATE 列は、2 月 29 日（うるう年を除く）または 2、'SHOE' という値を受け入れることはできません。列に入れられる各値は列のデータ型を受け継ぎます。たとえば、DATE 列に '01-JAN-92' を挿入すると、Oracle はそれが有効な日付に解釈されることを確認した後で、文字列 '01-JAN-92' を DATE 値として扱います。

表 2-1 に、Oracle の内部データ型をまとめます。これらのデータ型について次に詳しく説明します。

**注意：** Oracle プリコンパイラにより埋込み SQL プログラムで他のデータ型が区別されます。このようなデータ型は、外部データ型と呼ばれ、ホスト変数に対応付けられています。内部データ型を外部データ型と混同しないでください。Oracle による内部データ型と外部データ型の変換を含めて、外部データ型の説明は、『Pro\*COBOL プリコンパイラ・プログラマーズ・ガイド』および『Pro\*C/C++ プリコンパイラ・プログラマーズ・ガイド』、『SQL\*Module for Ada Programmer's Guide』を参照してください。

表 2-1 内部データ型のまとめ

コード	内部のデータ型	説明
1	VARCHAR2( <i>size</i> )	最大長が <i>size</i> バイトの可変長文字列。 <i>size</i> の最大は 4000、最小は 1。VARCHAR2 では <i>size</i> を必ず指定しなければならない。
	NVARCHAR2( <i>size</i> )	最大長が <i>size</i> 文字またはバイト（選択された各国キャラクタ・セットによる）の可変長文字列。最大サイズは、各文字を保存するのに必要なバイト数によって決まるが、最大で 4000 バイト。NVARCHAR2 では <i>size</i> を必ず指定しなければならない。
2	NUMBER( <i>p</i> , <i>s</i> )	精度 <i>p</i> 、位取り <i>s</i> を持つ数。精度 <i>p</i> には 1 から 38 までの値を指定できる。位取り <i>s</i> には -84 から 127 までの値を指定できる。

表 2-1 内部データ型のまとめ

コード	内部のデータ型	説明
8	LONG	最大 2 ギガバイト (2 の 31 乗から 1 を引いたバイト数) の可変長の文字データ。
12	DATE	紀元前 4712 年 1 月 1 日から紀元 4712 年 12 月 31 日までの日付を指定できる。
23	RAW( <i>size</i> )	長さが <i>size</i> バイトのバイナリ・データ。 <i>size</i> の最大は 2000 バイト。 RAW 値では <i>size</i> を必ず指定しなければならない。
24	LONG RAW	最大 2GB の可変長のバイナリ・データ。
69	ROWID	表の中の行のアドレスを一意に表す 16 進数ストリング。主に、 ROWID 疑似列によって戻される値のためのデータ型。
96	CHAR( <i>size</i> )	長さ <i>size</i> バイトの固定長文字データ。 <i>size</i> の最大は 2000 バイト。 デフォルトと最小 <i>size</i> は 1。
	NCHAR( <i>size</i> )	長さ <i>size</i> 文字またはバイト（各国キャラクタ・セットの選択内容に応じる）の固定長文字データ。 <i>size</i> の最大は、各文字を保存するのに必要なバイト数によって決まるが、最大で 2000 バイト。 デフォルトと最小 <i>size</i> は 1 文字または 1 バイト（キャラクタ・セットに応じる）。
106	MLSLABEL	バイナリ形式のオペレーティング・システム・ラベル。このデータ型は、 Trusted Oracle との下位互換性に使います。
112	CLOB	シングルバイト・キャラクタを含む文字ラージ・オブジェクト。可変幅のキャラクタ・セットはサポートされません。最大サイズは 4GB。
	NCLOB	固定幅のマルチバイト・キャラクタを含む文字ラージ・オブジェクト。可変幅のキャラクタ・セットはサポートされません。最大サイズは 4GB。各国キャラクタ・セットのデータを保存します。
113	BLOB	バイナリ・ラージ・オブジェクト。最大サイズは 4GB。
114	BFILE	データベース外に保存された大きなバイナリ・ファイルへの参照が格納されます。データベース・サーバー機上に存在する外部 LOB へのバイト・ストリーム I/O アクセスを可能にします。最大サイズは 4GB。

上記のデータ型のコードは Oracle によって内部的に使用されます。 DUMP 関数を使用すると、列またはオブジェクト属性のデータ型コードが戻されます。

## 文字データ型

文字データ型を使用すると、単語や自由形式のテキストなど、データベースのキャラクタ・セットまたは各国キャラクタ・セット内の文字 (英数字) データを格納できます。他のデータ型よりも制限が少なく、その結果、特性も少なくなっています。たとえば、文字データ型の列はすべての英数字の値を格納できますが、NUMBER 型の列は数値しか格納できません。

文字データは、7 ビット ASCII や EBCDIC コード・ページ 500 など、データベース作成時に指定されたキャラクタ・セットの 1 つに対応しているバイト値で文字列に格納されます。Oracle は、シングルの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ています。

以下のデータ型が文字データに対して使用されます。

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

### CHAR データ型

CHAR データ型は固定長の文字列を指定します。CHAR 列で表を作成する場合、列の長さをバイト単位で指定します。Oracle では、その列の中に格納される値がすべてこの長さを持つように調整します。列の長さよりも短い値が挿入されると、その値の後に空白を埋め込んで列の長さに合わせます。列に対して長すぎる値を挿入しようとする、Oracle はエラーを戻します。

CHAR 列のデフォルトの長さは 1 文字で、この許容最大値は 2000 文字です。長さがゼロの文字列を CHAR 列に挿入できますが、この CHAR 列が比較されるときには、空白が 1 文字埋め込まれます。比較方法の説明は、「データ型の比較規則」(2-22 ページ) を参照してください。

### NCHAR データ型

NCHAR データ型は固定長の各国キャラクタ・セット文字列を指定します。NCHAR 列で表を作成する場合、列の長さを文字単位またはバイト単位のいずれかで定義します。使用する各国キャラクタ・セットは、データベースを作成するときに指定します。

指定した各国キャラクタ・セットが固定幅である場合は (たとえば、JA16EUCFIXED など)、NCHAR の列サイズを文字列の長さの文字数で宣言します。各国キャラクタ・セットが可変幅 (たとえば、JA16SJIS など) である場合は、列サイズをバイト単位で宣言します。次の文では、各国キャラクタ・セットとして JA16EUCFIXED を使用すると、長さが 30 文字までの文字列を保存できる 1 つの NCHAR 列が含まれた表が作成されます。

```
CREATE TABLE tab1 (col1 NCHAR(30));
```

列の最大長は、各国キャラクタ・セットの定義によって決まります。文字データ型 **NCHAR** の幅指定では、各国キャラクタ・セットが固定幅である場合には文字数が参照され、各国キャラクタ・セットが可変幅である場合にはバイト数が参照されます。許容最大列サイズは 2000 バイトです。固定幅のマルチバイト・キャラクタ・セットの場合は、列の許容最大長が、2000 バイト以下の文字数になります。

列の長さよりも短い値が挿入されると、その値の後に空白を埋め込んで列の長さに合わせます。**CHAR** 値を **NCHAR** 列に挿入することや、**NCHAR** 値を **CHAR** 列に挿入することはできません。

次の例では、`tab1` の `col1` 列と各国キャラクタ・セットの文字列 **NCHAR** リテラルを比較します。

```
SELECT * FROM tab1 WHERE col1 = N'NCHAR literal';
```

❗ **NCHAR** 属性を持つオブジェクトは作成できませんが、メソッドでは **NCHAR** パラメータを指定できます。

### NVARCHAR2 データ型

**NVARCHAR2** データ型は可変長の各国キャラクタ・セット文字列を指定します。**NVARCHAR2** 列で表を作成する場合、保持できる最大の文字数またはバイト数を指定します。Oracle では、列の最大長を超えていない限り、各値を指定されたとおりに正確に列に保存します。

列の最大長は、各国キャラクタ・セットの定義によって決まります。文字データ型 **NVARCHAR2** の幅指定では、各国キャラクタ・セットが固定幅である場合には文字数が参照され、各国キャラクタ・セットが可変幅である場合にはバイト数が参照されます。許容最大列サイズは 4000 バイトです。固定幅のマルチバイト・キャラクタ・セットの場合は、列の許容最大長が、4000 バイト以下の文字数になります。

次の文では、各国キャラクタ・セットとして **JA16EUCFIXED** を使用すると、長さが 2000 文字（各文字が 2 バイトなので 4000 バイトとして保存される）の 1 つの **NVARCHAR2** 列が含まれた表が作成されます。

```
CREATE TABLE tab1 (col1 NVARCHAR2(2000));
```

❗ **NVARCHAR2** 属性を持つオブジェクトは作成できませんが、メソッドでは **NVARCHAR2** パラメータを指定できます。

### VARCHAR2 データ型

**VARCHAR2** データ型は可変長の文字列を指定します。**VARCHAR2** 列を作成する場合、保持できる最大のデータのバイト数を指定できます。Oracle では、列の最大長を超えていない限り、各値を指定されたとおりに正確に列に保存します。保存される文字列の実際の長さは 0 (ゼロ) でも構いませんが、最大長は 1 バイト以上でなければなりません。最大長を超える値を挿入しようとすると、Oracle はエラーを戻します。

VARCHAR2 列では最大長を指定しなければなりません。VARCHAR2 データの最大長は 4000 バイトです。Oracle は、非空白埋め比較を使用して VARCHAR2 値を比較します。比較方法については、「データ型の比較規則」(2-22 ページ)を参照してください。

## VARCHAR データ型

現在、VARCHAR データ型は VARCHAR2 データ型と同義です。VARCHAR よりも VARCHAR2 を使用することをお勧めします。将来、VARCHAR データ型が変更され、異なる比較方法で比較される別の可変長文字列の型になる可能性もあります。

## NUMBER データ型

NUMBER データ型は、38 桁の精度を持ち、ゼロおよび絶対値が、 $1.0 \times 10^{-130}$  から  $9.9...9 \times 10^{125}$  (38 個の 9 の後に 0 が 88 個続く) までの範囲にある正と負の固定小数点数および浮動小数点数を格納するために使用されます。 $1.0 \times 10^{126}$  以上の値を持つ算術式を指定した場合、Oracle はエラーを戻します。

次の書式で固定小数点数を指定できます。

NUMBER(*p*,*s*)

ここで、それぞれの意味は次のとおりです。

- |          |   |
|----------|---|
| <i>p</i> | 精度 ( <i>precision</i> )、すなわち全体の桁数です。Oracle は 38 桁までの精度で数の移植性を保証します。 |
| <i>s</i> | 位取り ( <i>scale</i> )、すなわち小数点の右側にある桁数です。位取りの有効範囲は -84 から 127 までです。   |

次の書式で整数を指定できます。

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

次の書式で浮動小数点を指定できます。

NUMBER 精度が 38 桁の浮動小数点数です。なお、浮動小数点数では、位取りは指定できません。(詳細は、「浮動小数点数」(2-10 ページ)を参照してください。)

## 位取りと精度

入力に対する特別の整合性検査として、固定小数点数列の位取りと精度を指定してください。位取りと精度を指定しても、すべての値が固定長に強制されるわけではありません。値が精度の有効範囲を超えると、Oracle はエラーを戻します。値が位取りの有効範囲を超えると、Oracle はその値を丸めます。

次に、いろいろな精度と位取りを使用して Oracle がデータを格納する例を示します。

実際のデータ	指定する精度と位取り	格納されるデータ
7456123.89	NUMBER	7456123.89
7456123.89	NUMBER(9)	7456124
7456123.89	NUMBER(9,2)	7456123.89
7456123.89	NUMBER(9,1)	7456123.9
7456123.89	NUMBER(6)	精度を超える
7456123.89	NUMBER(7,-2)	7456100
7456123.89	NUMBER(7,2)	精度を超える

負の位取り

位取りが負の場合には、実際のデータは整数部分の右から指定された桁数だけ丸められます。たとえば、(10,-2) と指定すると 100 の位まで丸められます。

精度より大きな位取り

通常はありえないことですが、精度よりも大きな位取りを指定できます。この場合、精度は小数点の右側にある最大有効桁数を示します。すべての NUMBER データ型と同じように、値が精度を超えると Oracle はエラー・メッセージを戻します。値が位取りの有効範囲を超えると、Oracle はその値を丸めます。たとえば、NUMBER(4,5) として定義された列は、小数点の後の最初の桁が 0(ゼロ) でなければならず、小数点以下 5 桁を超える値はすべて丸められます。次に、精度より大きい位取りを指定した場合の例を示します。

実際のデータ	指定する精度と位取り	格納されるデータ
.01234	NUMBER(4,5)	.01234
.00012	NUMBER(4,5)	.00012
.000127	NUMBER(4,5)	.00013
.0000012	NUMBER(2,7)	.0000012
.00000123	NUMBER(2,7)	.0000012

浮動小数点数

浮動小数点数は、最初の桁から最後の桁までの任意の位置に小数点を置くことも、小数点を省略することもできます。小数点以降の桁数に制限はないため、浮動小数点数に対して位取りは指定できません。



「NUMBER データ型」(2-9 ページ) で説明している形式で浮動小数点数を指定できます。また、Oracle は ANSI データ型 FLOAT もサポートします。次の構文書式のいずれかを使用してこのデータ型を指定します。

FLOAT	38 桁の 10 進精度または 126 桁の 2 進精度で浮動小数点数を指定します。
FLOAT( <i>b</i> )	2 進精度 <i>b</i> で浮動小数点数を指定します。精度 <i>b</i> は 1 から 126 までの範囲です。2 進精度から 10 進精度に変換するには、 <i>b</i> に 0.30103 を乗算します。10 進精度から 2 進精度に変換するには、10 進精度に 3.32193 を乗算します。2 進数精度の 126 桁は 10 進精度の 38 桁におよそ等しくなります。

## LONG データ型

LONG 列には、最大 2GB(2 の 31 乗から 1 を引いたバイト数) の可変長の文字列を格納できます。LONG 列には多くの点で VARCHAR2 列と同じ特性があります。LONG 列を使用すると、長いテキスト列を格納できます。Oracle は、ビュー定義のテキストを格納するために、データ・ディクショナリ内で LONG 列を使用しています。LONG 値の長さは、使用しているコンピュータで利用できるメモリーによって制限される可能性もあります。

SQL 文の中の次の場所で LONG 列を参照できます。

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

LONG 値を使用する場合には次の制限があります。

- 表には複数の LONG 列を含めることはできない。
- LONG 列は整合性制約に使用できない。ただし、NULL 制約と NOT NULL 制約を除く。
- LONG 列に索引を付けることはできない。
- ストアド・ファンクションは LONG 値を戻すことはできない。
- 単一の SQL 文に指定する、すべての LONG 列、更新対象の表、ロック対象の表は、同一データベース上に位置していなければならない。

また、LONG 列は SQL 文の次のような部分では使用できません。

- SELECT 文の WHERE 句、GROUP BY 句、ORDER BY 句、CONNECT BY 句または DISTINCT 演算子
- SELECT 文の UNIQUE 句
- CREATE CLUSTER 文の列リスト
- CREATE SNAPSHOT 文の CLUSTER 句

- SQL 関数 (SUBSTR や INSTR など)
- 式または条件
- GROUP BY 句を含む問合せの SELECT リスト
- 集合演算子によって結合されている副問合せや問合せの SELECT リスト
- CREATE TABLE ... AS SELECT 文の SELECT リスト
- INSERT 文の副問合せの SELECT リスト

トリガーでは、LONG データ型は次のように使用されます。

- トリガー内の SQL 文で、データを LONG 列に挿入できる。
- LONG 列のデータを CHAR や VARCHAR2 などの制約があるデータ型に変換できる場合は、トリガー内の SQL 文で LONG 列を参照できる。これらのデータ型の最大長は 32KB です。
- トリガー内の変数は、LONG データ型を使用して宣言できない。
- :NEW と :OLD は LONG 列で使えない。

Oracle コール・インタフェースを使用して、データベースから LONG 値の一部を検索できます。『Oracle 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。

## DATE データ型

DATE データ型は日付と時間の情報を格納するために使用されます。日付と時間の情報は CHAR データ型と NUMBER データ型で表現できますが、DATE データ型には特別に対応付けられている特性があります。各 DATE 値には、世紀、年、月、日、時、分、秒の情報が格納されます。

日付値を指定するには、文字値や数値を TO\_DATE 関数によって日付値に変換しなければなりません。デフォルト日付書式の文字値が日付式で使用されると、Oracle は自動的にそれらを日付値に変換します。デフォルトの日付書式は、初期化パラメータ NLS\_DATE\_FORMAT によって指定され、たとえば 'DD-MON-YY' のような文字列になります。'DD-MON-YY' は、日付としての 2 桁の数、月の名前の省略形、年の下 2 桁を含む日付書式です。

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

日付関数 SYSDATE は現在の日付と時間を戻します。SYSDATE 関数と TO\_DATE 関数およびデフォルト日付書式の説明は、第 3 章「演算子、関数、式、条件」を参照してください。

### 日付算術

日付に対しては、日付の加算や減算だけでなく、数定数の加算や減算ができます。Oracle は算術日付式の数定数を日付の数として解釈します。たとえば、SYSDATE + 1 は明日です。SYSDATE - 7 は 1 週間前です。SYSDATE + (10/1440) は 10 分後です。SYSDATE から EMP

表の **HIREDATE** 列を引くと、各従業員が雇用されてから経過した日数が戻ります。**DATE** 値の乗算や除算はできません。

Oracle は一般的な日付操作のために関数を用意しています。たとえば、**ADD\_MONTHS** 関数は日付に月を加算したり、減算したりできます。**MONTHS\_BETWEEN** 関数は2つの日付の間の月数を戻します。結果の小数部は月(1ヶ月は31日)を単位として表されています。日付関数の詳細は、「日付関数」(3-34 ページ)を参照してください。

各日付には時間コンポーネントが含まれるため、日付操作のほとんどの結果には小数部が含まれます。この小数部は日を単位として表されています。たとえば、1.5 日は36時間です。

## ユリウス暦の使用法

ユリウス暦は紀元前4712年1月1日から経過した日数です。ユリウス暦によって共通の基準で日付を算定できます。日付関数 **TO\_DATE** と **TO\_CHAR** で日付書式モデル「J」を使用して、Oracle の **DATE** 値とユリウス暦の間で変換を行うことができます。

例 次の文では1997年1月1日と等価なユリウス暦が戻されます。

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

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

**DUAL** 表については、「**DUAL** 表からの選択」(4-530 ページ)を参照してください。

## RAW データ型と LONG RAW データ型

**RAW** データ型と **LONG RAW** データ型のデータは、Oracle によって解釈されません(異なるシステム間でデータを移動するときに変換されない)。これらのデータ型は、バイナリ・データまたはバイト列に使用されます。たとえば、**LONG RAW** は、図形または音声、文書、バイナリ・データの配列の格納に使用できますが、解釈方法は用途によって異なります。

**RAW** は、**VARCHAR2** 文字データ型と同様の可変長データ型です。ただし、**Net8**(ユーザー・セッションとインスタンスを接続する)と **Import** および **Export** ユーティリティは、**RAW** または **LONG RAW** データの転送時には文字変換を行いません。これに対し、**Net8** と **Import/Export** は、データベースのキャラクタ・セットとユーザーのセッションのキャラクタ・セット (**ALTER SESSION** コマンドの **NLS\_LANGUAGE** パラメータによって設定) が異なる場合に、**CHAR** および **VARCHAR2**、**LONG** データをこれら2つのキャラクタ・セット間で自動的に変換します。

**RAW** データまたは **LONG RAW** データと **CHAR** データ間の自動変換時に、バイナリ・データは16進数で表現されます。1つの16進文字で4ビットの **RAW** データを表します。たと

例えば、ビット列が 11001011 で表示される 1 バイトの RAW データは、'CB' として表示あるいは入力されます。

RAW データには索引を付けられますが、LONG RAW データには索引を付けられません。

## ラージ・オブジェクト (LOB) データ型

内部 LOB データ型の BLOB、CLOB、NCLOB、および外部データ型の BFILE には、構造化されていない大きいデータ（テキスト、イメージ、ビデオ、スペシャル・データなど）を最大 4GB まで格納できます。

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

内部 LOB 列には、ライン外 LOB 値またはインライン LOB 値を参照できる LOB ロケータが含まれています。表から LOB を選択すると、実際には LOB のロケータが戻され、LOB 値全体は戻されません。LOB に対する DBMS\_LOB パッケージと OCI の操作は、これらのロケータを介して実行されます。これらのインタフェースおよび LOB の詳細は、『Oracle8 Server アプリケーション開発者ガイド』および『Oracle8 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。

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

- LOB は、ユーザー定義のデータ型（オブジェクト）の属性に指定できる。
- LOB ロケータは、表の列に格納される。実際の LOB 値は表の列に格納される場合とされない場合がある。格納されない場合、BLOB および NCLOB、CLOB の値は、別々の表領域に格納される。また、BFILE データは、サーバー上の外部ファイルに格納される。
- LOB 列にアクセスしたときに戻されるのはロケータである。
- LOB のサイズは最大 4GB である。BFILE の最大サイズはオペレーティング・システムによって異なるが、4GB を超えることはない。
- LOB では、データの効果的かつランダムな断片単位のアクセスおよび操作が可能である。
- 1 つの表内に 2 つ以上の LOB 列を定義できる。
- NCLOB の例外を除いては、1 つのオブジェクトに 1 つ以上の LOB 属性を定義できる。
- LOB バインド変数を宣言できる。
- LOB 列と LOB 属性を選択できる。
- 1 つ以上の LOB 列または 1 つ以上の LOB 属性を持つ、オブジェクトが含まれている新しい行を挿入したり、既存の行を更新したりできる。（内部 LOB 値を NULL つまり空に設定したり、LOB 全体をデータに置き換えたりできる。BFILE は、NULL に設定、または別のファイルを指すように設定できる。）

- LOB 行と列の交差部またはLOB 属性を、別の LOB 行と列の交差部またはLOB 属性を使って更新できる。
- LOB 列またはLOB 属性が含まれている行を削除できる（これにより LOB 値も削除される）。BFILE の場合、実際のオペレーティング・システム・ファイルは削除されない。

詳細は、『Oracle8 Server アプリケーション開発者ガイド』にある LOB の制限に関する説明を参照してください。

内部 LOB 列の行にアクセスし、行を挿入するには、INSERT 文を使用して最初に内部 LOB 値を空に初期化します。行を挿入したら、空の LOB を選択し、DBMS\_LOB パッケージまたは OCI を使ってそれを移入できます。

次の例では、LOB 列を持つ表が作成されます。

```
CREATE TABLE person_table (name CHAR(40),
                             resume CLOB,
                             picture BLOB)
LOB (resume) STORE AS
( TABLESPACE resumes
  STORAGE (INITIAL 5M NEXT 5M) );
```

LOB 値の大容量のデータの読み書きにはLOBを使用します。

## BFILE データ型

BFILE データ型を使用すると、Oracle データベース外のファイル・システムに格納されているバイナリ・ファイル LOB にアクセスできます。BFILE 列または属性には、サーバーのファイル・システム上のバイナリ・ファイルに対するポインタとして機能する、BFILE ロケータが格納されます。ロケータには、ディレクトリ別名とファイル名が保持されます。「CREATE DIRECTORY コマンド」(4-226 ページ)を参照してください。

バイナリ・ファイル LOB は、トランザクションにはかかりません。ファイルの統合性と耐久性を提供しているのは基本にあるオペレーティング・システムです。サポートされるファイルの最大サイズは 4GB です。

データベース管理者は、ファイルが存在すること、および Oracle のプロセスがファイルに対するオペレーティング・システムの読み込み許可があることを確認する必要があります。

BFILE データ型を使用すると、大きいバイナリ・ファイルの読み込み専用サポートが可能になります。この場合、ファイルは修正できません。Oracle では、ファイル・データにアクセスするために API が提供されています。ファイル・データにアクセスするために使用するインタフェースは、DBMS\_LOB パッケージと OCI です。LOB の詳細は、『Oracle8 Server アプリケーション開発者ガイド』および『Oracle8 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。

### BLOB データ型

BLOB データ型は、構造化されていないバイナリラージ・オブジェクトを格納するために使用します。BLOB は、キャラクタ・セットの意味を持たないビットストリームとして考えることができます。BLOB には、4GB までのバイナリ・データを格納できます。

BLOB では、トランザクションが完全にサポートされます。SQL または DBMS\_LOB パッケージ、OCI を介して行った変更は、すべてトランザクションに反映されます。BLOB 値の操作は、コミットまたはロール・バックできます。1 つのトランザクション内の PL/SQL または OCI 変数を BLOB ロケータに保存し、そのロケータを別のトランザクションまたはセッションで使用することはできません。

### CLOB データ型

CLOB データ型は、シングルスバイト文字のラージ・オブジェクト・データを格納するために使用します。可変幅のキャラクタ・セットはサポートされません。CLOB には、4GB までの文字データを格納できます。

CLOB では、トランザクションが完全にサポートされます。SQL または OCI、DBMS\_LOB パッケージを介して行った変更はすべて、トランザクションに反映されます。CLOB 値の操作は、コミットまたはロール・バックできます。1 つのトランザクション内の PL/SQL または OCI 変数を CLOB ロケータに保存し、そのロケータを別のトランザクションまたはセッションで使用することはできません。

### NCLOB データ型

NCLOB データ型には、固定幅のマルチバイト各国キャラクタ・セット文字 (NCHAR) データを保存するために使用します。可変幅のキャラクタ・セットはサポートされません。NCLOB には、4GB までの文字テキスト・データを保存できます。

NCLOB では、トランザクションが完全にサポートされます。SQL または DBMS\_LOB パッケージ、OCI を介して行った変更は、すべてトランザクションに反映されます。NCLOB 値の操作は、コミットまたはロール・バックできます。1 つのトランザクション内の PL/SQL または OCI 変数を NCLOB ロケータに保存し、そのロケータを別のトランザクションまたはセッションで使用することはできません。

**【注意】** NCLOB 属性を持つオブジェクトは作成できませんが、メソッドで NCLOB パラメータを指定することはできます。

## ROWID データ型

データベース内の各行にはアドレスがあります。疑似列 ROWID を問い合わせることによって行のアドレスを調べることができます。この疑似列の値は、各行のアドレスを表す 16 進数ストリングです。16 進数ストリングのデータ型は ROWID です。ROWID 疑似列の詳細は、「疑似列」(2-30 ページ) を参照してください。また、ROWID データ型を持つ実際の列を含む表やクラスタを作成することもできます。Oracle では、そのような列の値が有効な ROWID であることは保証されません。

### 制限 ROWID

Oracle8 Server では、パーティション表と索引および表領域関連のデータ・ブロック・アドレス (DBA) を明確かつ効果的にサポートするために、ROWID の拡張形式が採用されています。

Oracle7 以前のリリースでは、ROWID は次のような文字値で表現されます。

`block.row.file`

ここで、それぞれの意味は次のとおりです。

- block*      行を含むデータ・ファイルのデータ・ブロックを識別する 16 進数ストリングです。このストリングの長さはオペレーティング・システムによって異なることがあります。
- row*        データ・ブロック内の行を識別する 4 桁の 16 進数ストリングです。ブロック内の最初の行は 0 になります。
- file*        行を含むデータ・ファイルを識別する 16 進数ストリングです。最初のデータ・ファイルは 1 になります。このストリングの長さはオペレーティング・システムによって異なることがあります。

Oracle8 では、この種の ROWID は制限 ROWID と呼ばれます。

### 拡張 ROWID

ユーザー列に格納される Oracle8 の拡張 ROWID データ型には、制限 ROWID のデータに加え、データ・オブジェクト番号が含まれます。データ・オブジェクト番号は、すべてのデータベース・セグメントに割り当てられる識別番号です。データ・オブジェクト番号は、データ・ディクショナリ・ビューの `USER_OBJECTS` および `DBA_OBJECTS`、`ALL_OBJECTS` から取り出すことができます。同じセグメントを共有するオブジェクト（たとえば、同じクラスタ内のクラスタ化された表など）には、同じオブジェクト番号が付けられます。

拡張 ROWID は直接利用できません。拡張 ROWID の内容を解釈するには、提供されているパッケージ `DBMS_ROWID` を使用します。パッケージ関数を使用すると、制限の ROWID から直接入手された情報が取り出され、提供されます。`DBMS_ROWID` パッケージの関数は、ビルトインの SQL 関数と同様に使用できます。表 2-2 は、`DBMS_ROWID` パッケージの関数とプロシージャのリストです。`DBMS_ROWID` の詳細は、『Oracle8 Server アプリケーション開発者ガイド』を参照してください。

表 2-2 DBMS\_ROWID 関数

関数名	説明
<code>ROWID_CREATE</code>	検査目的だけの ROWID を作成する。
<code>ROWID_TYPE</code>	ROWID のタイプを戻す。0 が制限、1 が拡張。

表 2-2 DBMS\_ROWID 関数

関数名	説明
ROWID_OBJECT	拡張 ROWID のオブジェクト番号を戻す。
ROWID_RELATIVE_FNO	ROWID のファイル番号を戻す。
ROWID_BLOCK_NUMBER	ROWID のブロック番号を戻す。
ROWID_ROW_NUMBER	行番号を戻す。
ROWID_TO_ABSOLUTE_FNO	特定の表内の行の ROWID に対応付けられている絶対ファイル番号を戻す。
ROWID_TO_EXTENDED	ROWID を制限形式から拡張形式に変換する。
ROWID_TO_RESTRICTED	拡張 ROWID を制限形式に変換する。
ROWID_VERIFY	ROWID が ROWID_TO_EXTENDED 関数によって正しく拡張されるかどうかをチェックする。

互換性と移行

制限形式の ROWID は、旧リリースとの互換性を持たせるために Oracle8 でもサポートされていますが、すべての表で拡張形式の ROWID が返されます。互換性と移行の問題については、『Oracle8 Server 移行ガイド』を参照してください。

MLSLABEL データ型

MLSLABEL データ型は、安全性の高いオペレーティング・システムで使用されるバイナリ形式のラベルを格納するために使用します。このデータ型は、Oracle8 では Trusted Oracle を使用した以前のバージョンの Oracle サーバーとの下位互換性のためにサポートされています。

ラベルは、情報へのアクセスを取り次ぐために Trusted Oracle によって使用されます。標準の Oracle Server を使用している場合も、これらのデータ型で列を定義できます。このデータ型とラベルを含めた Trusted Oracle の詳細は、使用している Trusted Oracle のマニュアルを参照してください。

ANSI、DB2、SQL/DS のデータ型

表とクラスタを作成する SQL コマンドは、ANSI データ型、および IBM 社の製品 SQL/DS と DB2 のデータ型も受け入れます。Oracle では ANSI または IBM のデータ型の名前を認識し、列のデータ型の名前として記録します。次に、表 2-3 および表 2-4 で規定される変換に基づいて Oracle のデータ型で列データを格納します。



表 2-3 Oracle データ型に変換される ANSI データ型

ANSI SQL データ型	Oracle データ型
CHARACTER( <i>n</i> )	CHAR( <i>n</i> )
CHAR( <i>n</i> )	
CHARACTER VARYING( <i>n</i> )	VARCHAR( <i>n</i> )
CHAR VARYING( <i>n</i> )	
NATIONAL CHARACTER( <i>n</i> )	NCHAR( <i>n</i> )
NATIONAL CHAR( <i>n</i> )	
NCHAR( <i>n</i> )	
NATIONAL CHARACTER VARYING( <i>n</i> )	NVARCHAR2( <i>n</i> )
NATIONAL CHAR VARYING( <i>n</i> )	
NCHAR VARYING( <i>n</i> )	
NUMERIC( <i>p,s</i> )	NUMBER( <i>p,s</i> )
DECIMAL( <i>p,s</i> ) <sup>a</sup>	
INTEGER	NUMBER(38)
INT	
SMALLINT	
FLOAT( <i>b</i> ) <sup>b</sup>	NUMBER
DOUBLE PRECISION <sup>c</sup>	
REAL <sup>d</sup>	

<sup>a</sup>NUMERIC データ型および DECIMAL データ型では、固定小数点数だけを指定できます。これらのデータ型では、s のデフォルトは 0 です。

<sup>b</sup>FLOAT データ型は、2 進精度 b を持つ浮動小数点数です。このデータ型のデフォルト精度は、126 桁の 2 進精度または 38 桁の 10 進精度です。

<sup>c</sup>DOUBLE PRECISION データ型は 126 桁の 2 進精度を持つ浮動小数点数です。

<sup>d</sup>REAL データ型は 63 桁の 2 進精度または 18 桁の 10 進精度を持つ浮動小数点数です。

表 2-4 Oracle データ型に変換される SQL/DS と DB2 のデータ型

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

<sup>a</sup>DECIMAL データ型では固定小数点数だけを指定できます。このデータ型では、s のデフォルトは 0 です。

<sup>b</sup>FLOAT データ型は、2 進精度 b を持つ浮動小数点数です。このデータ型のデフォルト精度は、126 桁の 2 進精度または 38 桁の 10 進精度です。

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

- GRAPHIC
- LONG VARGRAPHIC
- VARGRAPHIC
- TIME
- TIMESTAMP

TIME と TIMESTAMP のデータは、Oracle の DATE データとして表現できることに注意してください。

**OBJ ユーザー定義型**

ユーザー定義のデータ型には、Oracle ビルトイン・データ型と他のユーザー定義のデータ型が、アプリケーション内のデータの構造と動作をモデル化する型の構築ブロックとして使用されます。Oracle ビルトイン・データ型については、『Oracle8 概要』を参照してください。ユーザー定義型の作成方法については、「CREATE TYPE コマンド」(4-342 ページ) および「CREATE TYPE BODY コマンド」(4-350 ページ) を参照してください。ユーザー定義型の使用方法については、『Oracle8 Server アプリケーション開発者ガイド』を参照してください。

## オブジェクト型

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

- 名前とは、スキーマ内でオブジェクト型を一意に識別するためのものです。
- 属性とは、ビルトイン型または他のユーザー定義型です。属性は、実社会エンティティの構造をモデル化します。
- メソッドとは、PL/SQL で記述され、データベースに格納される関数またはプロシージャ、もしくは、C などの言語で記述され、外部に格納される関数またはプロシージャのことです。メソッドは、アプリケーションが実社会エンティティに対して実行できる操作をインプリメントします。

## REF

オブジェクト識別子 (OID) を使用することにより、オブジェクトを一意に識別し、他のオブジェクトまたは関係表からそのオブジェクトを参照できます。REF と呼ばれるビルトイン・データ型が、そのような参照を表します。REF は、オブジェクト識別子のコンテナです。REF は、オブジェクトへのポインタとなります。

REF 値が、存在しないオブジェクトを指している場合、その REF は Dangling(参照先がない)状態であるといえます。Dangling は、NULL の状態とは異なります。REF が Dangling 状態であるかないかを確認するには、IS [NOT] Dangling という述語を使います。たとえば、列型が EMP\_T 型を指している REF である MGR 列があります。この MGR 列を含む表 DEPT は、次のように指定します。

```
SELECT t.mgr.name
FROM dept t
WHERE t.mgr IS NOT Dangling;
```

## VARRAY

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

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

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

- 関係表の列のデータ型
- オブジェクト型属性
- PL/SQL の変数またはパラメータ、関数の戻り型

---

---

**注意：** 索引構成表の列のデータ型には **VARRAY** は指定できません。

---

---

1 つの配列オブジェクトはインライン形式でその行の他のデータと同じ表領域に格納されます。

### ネストした表

ネストした表の型は順序付けされていない要素のセットを表現するのに使われます。その要素は、ビルトイン型またはユーザー定義型です。ネストした表は、単一系列の表として表示できます。ネストした表がオブジェクト型の場合、オブジェクト型のそれぞれの属性を表す複数列の表としても表示できます。

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

- 関係表の列
- オブジェクト型属性
- PL/SQL の変数およびパラメータ、関数の戻り値

ネストした表が、関係表内の列型として現れた場合、またはオブジェクト表の基礎を形成するオブジェクト型の属性として現れた場合、Oracle は、ネストした表のすべてのデータを単一表に格納し、その単一表を、ネストした表を囲む関係表またはオブジェクト表に対応付けます。

## データ型の比較規則

ここでは Oracle が各データ型の値を比較する方法について記述します。

### 数値

大きな値は小さな値よりも大きいとみなされます。すべての負の数は、ゼロとすべての正の数より小さいとみなされます。したがって、-1 は 100 より小さく、-100 は -1 より小さいとみなされます。

### 日付値

後の日付は前の日付よりも大きいとみなされます。たとえば、'29-MAR-1991'(1991 年 3 月 29 日) に等しい日付は '05-JAN-1992'(1992 年 1 月 5 日) に等しい日付よりも小さく、'05-JAN-1992 1:35pm'(1992 年 1 月 5 日午後 1 時 35 分) に等しい日付は '05-JAN-1992 10:09am'(1992 年 1 月 5 日午前 10 時 9 分) に等しい日付よりも大きいとみなされます。

### 文字列値

文字値は、次のどちらかの比較方法を使用して比較されます。

- 空白埋め比較
- 非空白埋め比較

ここでは、これらの比較方法について説明します。これらの異なる比較方法を使用して2つの文字値を比較した場合、その結果が異なることもあります。表 2-5 に、それぞれの比較方法を使用して5組の文字を比較した結果を示します。通常、空白埋め比較と非空白埋め比較の結果は同じです。表に示されている最後の比較では、空白埋め比較と非空白埋め比較の違いが明確になっています。

表 2-5 空白埋め比較と非空白埋め比較による比較結果

空白埋め比較	非空白埋め比較
'ab' > 'aa'	'ab' > 'aa'
'ab' > 'a '	'ab' > 'a '
'ab' > 'a'	'ab' > 'a'
'ab' = 'ab'	'ab' = 'ab'
'a ' = 'a'	'a ' > 'a'

**空白埋め比較** 2つの値の長さが異なる場合、Oracle はまず短い方の値の最後に空白を追加して、2つの値が同じ長さになるようにします。次に Oracle は、その2つの値を、最初に異なる文字まで1文字ずつ比較します。最初に異なる文字位置で、より大きい文字をもつ値の方が大きいとみなされます。2つの値に異なる文字がない場合、その2つの値は等価とみなされます。この規則は、2つの値の後続する空白数だけが異なる場合、その2つの値は等価であるということの意味します。Oracle では、比較する両方の値が、CHAR データ型、NCHAR データ型、テキスト・リテラルのいずれかの式の場合、または USER 関数の戻り値の場合だけ空白埋め比較方法を使います。

**非空白埋め比較** Oracle は、2つの値を、最初に異なる文字まで1文字ずつ比較します。最初に異なる文字位置で、より大きい文字をもつ値の方が大きいとみなされます。長さの異なる2つの値を短い方の値の最後まで比較して、すべて同じ文字だった場合、長い方の値が大きいとみなされます。同じ長さの2つの値に異なる文字がない場合、その2つの値は等価とみなされます。Oracle では、比較する片方または両方の値が VARCHAR2 データ型か NVARCHAR2 データ型の場合、非空白埋め比較方法を使います。

単一文字

Oracle は、データベースのキャラクタ・セットで与えられた数値に従って各文字を比較します。第1の文字の数値が第2の文字の数値よりも大きい場合、第1の文字は第2の文字よりも大きいとみなされます。Oracle は、空白はどの文字よりも小さいとみなします。これは、ほとんどのキャラクタ・セットで言えることです。

次に、一般的なキャラクタ・セットを列挙します。

- 7ビット ASCII( 情報交換用米国標準コード)
- EBCDIC コード (拡張 2 進化 10 進コード) ページ 500
- ISO 8859/1( 国際標準化機構)
- JEUC 日本語拡張 UNIX

ASCII と EBCDIC のキャラクタ・セットの一部を表 2-6 と表 2-7 に示します。なお、大文字と小文字は同じではありません。また、キャラクタ・セットの照合順番は、特定の言語に対する言語順序と一致しない場合があります。

表 2-6 ASCII キャラクタ・セット

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

表 2-7 EBCDIC キャラクタ・セット

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

### ● オブジェクト値

オブジェクト値は、MAP と ORDER の 2 つの比較方法のいずれかを使って比較されます。どちらの関数でもオブジェクト型インスタンスは比較されますが、両者はきわめて異なります。これらの関数は、オブジェクト型の一部として指定されなければなりません。

MAP 関数は単一のオブジェクトを引数として取り、スカラー値を戻します。2 つのオブジェクトを比較するために、MAP メソッドはそれぞれのオブジェクトに個々に適用されます。その上で結果が比較されます。MAP は、オブジェクト型をスカラーにマップするハッシュ関数と似ています。

ORDER 関数は、2 つのオブジェクト (たとえば、object1 と object2) を引数にとり、単純に 1 つのオブジェクトを他方のオブジェクトと比較します。ORDER 関数は、object1 が object2 より大きい場合は +1、両者が等しい場合は 0、object1 が object2 より小さい場合は -1 を返します。ORDER メソッドによって NULL 値が戻ることはありません。

オブジェクト・インスタンスに対して大量のソートやハッシュ結合の操作を実行する場合には、MAP を使用します。MAP は、オブジェクトをスカラー値にマップするために 1 度だけ

適用され、その後のソートやマージではスカラーが使用されます。MAP メソッドは、オブジェクトの比較が生じるたびに呼び出さなければならない ORDER メソッドよりも効率的です。ハッシュ結合の場合には必ず MAP メソッドを使用します。ハッシュ結合ではオブジェクト値に関してハッシングを行うので、ORDER メソッドは使用できません。

オブジェクト指定には、1つの比較方法だけを含めることができます。その比較方法は関数でなければなりません。型指定に MAP メソッドまたは ORDER メソッドのいずれか（両方ではない）を定義できます。

2つのオブジェクト型が等しいかどうかを判断するために、比較方法を指定する必要はありません。

詳細は、「CREATE TYPE コマンド」(4-342 ページ) および『Oracle8 Server アプリケーション開発者ガイド』を参照してください。

### VARRAY とネストした表

Oracle8 では、VARRAY とネストした表は比較できません。

## データ変換

一般に、式には異なるデータ型の値を含めることができません。たとえば、式では 10 を 5 に乗じた結果に 'JAMES' を加えることはできません。ただし、Oracle では値のあるデータ型から別のデータ型へ変換する場合に、暗黙の変換と明示的な変換がサポートされています。

### 暗黙のデータ変換

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

- INSERT 文または UPDATE 文で、あるデータ型の値を別のデータ型の列に割り当てる場合。Oracle はその値を列のデータ型に変換します。
- SQL 関数または演算子に不当なデータ型の引数を指定して使用する場合。Oracle はその引数を正当なデータ型に変換します。
- 異なるデータ型の値に比較演算子を使用する場合。Oracle は式の一方を他方のデータ型に変換します。

**例 1** テキスト・リテラル '10' は CHAR データ型です。次の文のように数式で使用すると暗黙的に NUMBER データ型に変換されます。

```
SELECT sal + '10'  
FROM emp;
```

**例 2** 条件で文字値と NUMBER 型の値を比較する場合、NUMBER 型の値は文字値に変換されず、文字値が暗黙的に NUMBER 型の値に変換されます。次の文では、'7936' が暗黙的に 7936 に変換されます。



```
SELECT ename
FROM emp
WHERE empno = '7936';
```

例 3 次の文では、Oracle がデフォルトの日付書式 'DD-MON-YYYY' を使用して、'12-MAR-1993' を DATE 値に暗黙的に変換します。

```
SELECT ename
FROM emp
WHERE hiredate = '12-MAR-1993';
```

例 4 次の文では、Oracle がテキスト・リテラル 'AAAAZ8AABAAABvIAAA' を ROWID 値に暗黙的に変換します。

```
SELECT ename
FROM emp
WHERE ROWID = 'AAAAZ8AABAAABvIAAA';
```

明示的なデータ変換

SQL 変換関数を使用して、データ型の変換を明示的に指定できます。表 2-8 に、あるデータ型から別のデータ型に明示的に値を変換する SQL 関数を示します。

表 2-8 データ型変換の SQL 関数

変換後	CHAR	NUMBER	DATE	RAW	ROWID
変換前					
CHAR	—	TO_NUMBER	TO_DATE	HEXTORAW	CHARTOROWID
NUMBER	TO_CHAR	—	TO_DATE (number, 'J')		
DATE	TO_CHAR	TO_CHAR (date, 'J')	—		
RAW	RAWTOHEX			—	
ROWID	ROWIDTOCHAR				—

これらの関数の詳細は、「変換関数」(3-39 ページ) を参照してください。

---

**注意：** 表 2-8 には LONG 型と LONG RAW 型の値からの変換が示されていません。LONG と LONG RAW の値を指定すると、Oracle で暗黙のデータ型変換を行うことができないためです。たとえば、関数や演算子を含む式では LONG と LONG RAW の値を使用できません。LONG データ型および LONG RAW データ型の制限については、「LONG データ型」(2-11 ページ)を参照してください。

---

## 暗黙のデータ変換と明示的なデータ変換

次の理由で、暗黙の変換または自動変換に頼るのではなく、明示的な変換を指定することをお勧めします。

- 明示的なデータ型変換関数を使用すると、SQL 文がわかりやすくなる。
- 自動的なデータ型変換（特に列値のデータ型が定数に変換される場合）は、パフォーマンスに悪影響を及ぼす可能性がある。
- 暗黙の変換はその変換が起こるコンテキストに依存し、どんな場合でも同じように機能するとは限らない。
- 暗黙の変換のアルゴリズムは、ソフトウェア・リリースや Oracle 製品の変更によって変更されることがある。明示的な変換を指定しておくと、その動作は将来的にも確実になる。

## NULL

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

*NULL* は値ゼロと等価ではないので、ゼロを表すために *NULL* 値を使用しないでください。(現在 Oracle は長さがゼロの文字値を *NULL* として処理します。ただし、この処理は Oracle の将来のバージョンでも継続されるとは限りませんので、空の文字列を *NULL* として処理しないことをお勧めします。) *NULL* を含む算術式は必ず *NULL* に評価されます。たとえば、*NULL* に 10 を加算しても結果は *NULL* です。実際、オペランドに *NULL* を指定すると、(連結演算子を除く) すべての演算子は *NULL* を戻します。

## SQL 関数での NULL

引数として *NULL* を指定すると、(*NVL* と *TRANSLATE* を除く) すべてのスカラー関数では *NULL* が戻されます。*NVL* 関数を使用すると、*NULL* が発生したときに値を戻すことができます。たとえば、式 *NVL*(*COMM*,0) は、*COMM* が *NULL* であれば 0 を戻し、*COMM* が *NULL* でなければ *COMM* の値を戻します。

ほとんどのグループ関数では NULL は無視されます。たとえば、1000 および NULL、NULL、NULL、2000 という 5 つの値の平均を得る問合せを考えます。そのような問合せでは NULL は無視され、平均は  $(1000+2000)/2=1500$  となります。

## 比較演算子での NULL

NULL を検査するには、比較演算子 IS NULL および IS NOT NULL だけを使用します。NULL を他の演算子で使用する、その結果が NULL の値に依存する場合、結果は UNKNOWN になります。NULL はデータの欠落を表すため、任意の値や別の NULL との関係で等号や不等号は成り立ちません。ただし、Oracle は DECODE 式を評価するときに 2 つの NULL を等しい値とみなすので注意してください。DECODE の構文については、「式」(3-73 ページ) を参照してください。

複合キーの場合、2 つの NULL は等しいと判断されます。言い換えると、NULL を含んだ 2 つの複合キーは、そのキーの NULL でないコンポーネントのすべてが等しい場合、同一であると判断されます。

## 条件での NULL

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

表 2-9 は、条件に NULL を含んだ評価の例です。SELECT 文の WHERE 句で UNKNOWN と評価される条件が使用された場合、その問合せに対して行は戻されません。

表 2-9 NULL を含む条件

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

NULL を含む論理式の結果を示した真理値表は、表 3-6（3-11 ページ）および表 3-7 と表 3-8 を参照してください。

## 疑似列

疑似列は表の列のように使用できますが、実際に表に格納されているわけではありません。疑似列から値を選択できますが、疑似列に対して値の挿入、更新、削除はできません。この項では次の疑似列について説明します。

- CURRVAL と NEXTVAL
- LEVEL
- ROWID
- ROWNUM

## CURRVAL と NEXTVAL

「順序」は一意の連続値を生成できるスキーマ・オブジェクトです。これらの値は主キーや一意のキーによく使用されます。次の疑似列を使用した SQL 文で、順序値を参照できます。

- CURRVAL            順序の現在の値を戻します。
- NEXTVAL           順序を増加させて次の値を戻します。

CURRVAL と NEXTVAL は、順序の名前で修飾しなければなりません。

```
sequence.CURRVAL  
sequence.NEXTVAL
```

別のユーザーのスキーマ内での順序の現在値または次にくる値を参照するには、その順序に対する **SELECT** オブジェクト権限または **SELECT ANYSEQUENCE** システム権限のどちらかが必要です。さらに、その順序は次に示すように順序を含むスキーマで修飾しなければなりません。

```
schema.sequence.CURRVAL  
schema.sequence.NEXTVAL
```

リモート・データベース上の順序の値を参照するには、次のように、データベース・リンクの完全な名前または部分的な名前でも順序を修飾しなければなりません。

```
schema.sequence.CURRVAL@dblink  
schema.sequence.NEXTVAL@dblink
```

データベース・リンクの参照については、「リモート・データベース内のオブジェクトを参照」(2-50 ページ) を参照してください。

**Trusted Oracle** を **DBMS MAC** モードで使用している場合、順序を参照するには、**DBMS** ラベルが順序の作成ラベルより上位であるか、次のいずれかの基準を満たしていなければなりません。

- 順序の作成ラベルが **DBMS** ラベルよりも高い場合、**READUP** システム権限と **WRITEUP** システム権限を持つ。
- 順序の作成ラベルと **DBMS** ラベルが一致していない場合、**READUP**、**WRITEUP**、**WRITEDOWN** の各システム権限を持つ。

**Trusted Oracle** を **OS MAC** モードで使用している場合、**DBMS** ラベルよりも低い作成ラベルを持つ順序は参照できません。

## 順序値の使用場所

次の場所で **CURRVAL** と **NEXTVAL** を使用できます。

- 副問合せまたはスナップショット、ビューに含まれていない **SELECT** 文の **SELECT** リスト
- **INSERT** 文内の副問合せの **SELECT** リスト
- **INSERT** 文の **VALUES** 句
- **UPDATE** 文の **SET** 句

次の場所では **CURRVAL** と **NEXTVAL** を使用できません。

- **DELETE** 文または **SELECT** 文、**UPDATE** 文内の副問合せ
- ビューの問合せまたはスナップショットの問合せ

- DISTINCT 演算子を持つ SELECT 文
- GROUP BY 句または ORDER BY 句を持つ SELECT 文
- 集合演算子 UNION または INTERSECT、MINUS によって別の SELECT 文と結合されている SELECT 文
- SELECT 文の WHERE 句
- CREATE TABLE 文または ALTER TABLE 文の列の DEFAULT 値
- CHECK 制約の条件

また、CURVAL または NEXTVAL を使用する単一の SQL 文では、参照対象の LONG 列、更新対象の表、ロック対象の表がすべて同じデータベース上になければなりません。

### 順序値の使用方法

順序を作成するときに、初期値と増分値を定義できます。NEXTVAL の最初の参照によって、順序の初期値が戻されます。その後の参照によって、定義された増分値で順序が増加され、その新しい値が戻されます。CURRVAL を参照すると、NEXTVAL への最後の参照で戻された値である、順序の現在の値が常に戻されます。なお、セッションの順序に対して CURRVAL を使用する前に、まず NEXTVAL で順序を初期化してください。

単一の SQL 文の中で、Oracle は一度だけ順序を増加させます。1 つの文の中で順序に対して複数回 NEXTVAL を参照している場合、Oracle は一度だけ順序を増加させ、NEXTVAL が現れる度にすべて同じ値を戻します。1 つの文の中で CURRVAL と NEXTVAL の両方を参照している場合、Oracle は順序を増加させ、それらが文の中で現れる順序にかかわらず、CURRVAL と NEXTVAL の両方に対して同じ値を戻します。

順序には、待機またはロックすることなく数多くのユーザーが同時にアクセスできます。順序については、「CREATE SEQUENCE コマンド」(4-278 ページ)を参照してください。

**例 1** この例は従業員順序の現在の値を選択します。

```
SELECT empseq.currval
FROM DUAL;
```

**例 2** この例は従業員順序を増加させ、従業員表に挿入される新しい従業員のためにその値を使用します。

```
INSERT INTO emp
VALUES (empseq.nextval, 'LEWIS', 'CLERK',
       7902, SYSDATE, 1200, NULL, 20);
```

**例 3** この例は次の注文番号を使用して新しい注文をマスター注文表に追加します。次にこの番号を使用して関連する注文をディテール注文表に追加します。

```
INSERT INTO master_order(orderno, customer, orderdate)
VALUES (orderseq.nextval, 'Al's Auto Shop', SYSDATE);
```

```

INSERT INTO detail_order (orderno, part, quantity)
VALUES (orderseq.currval, 'SPARKPLUG', 4);

INSERT INTO detail_order (orderno, part, quantity)
VALUES (orderseq.currval, 'FUEL PUMP', 1);

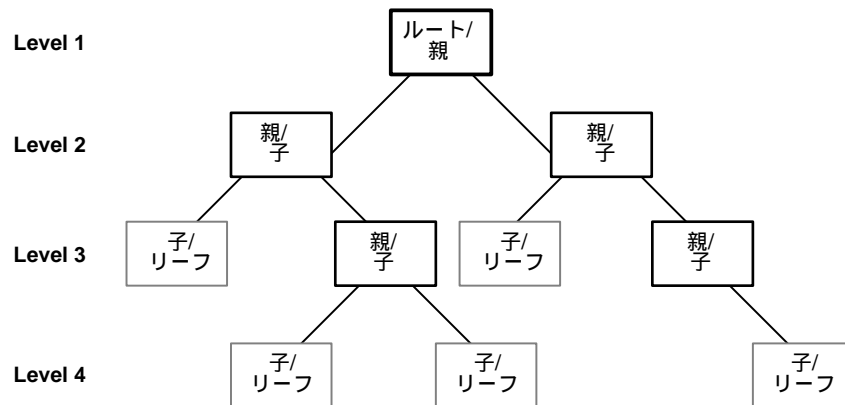
INSERT INTO detail_order (orderno, part, quantity)
VALUES (orderseq.currval, 'TAILPIPE', 2);

```

## LEVEL

階層問合せによって戻される各行について、LEVEL 疑似列は、ルート・ノードに 1 を戻し、ルートの子には 2 を戻します（以下同様です）。ルート・ノードは逆ツリー構造の最上位ノードです。子ノードはルートではない任意のノードです。親ノードは子を持つ任意のノードです。リーフ・ノードは子を持たない任意のノードです。図 2-1 に逆ツリーのノードとそれらの LEVEL 値を示します。

図 2-1 階層ツリー



問合せで階層型の関連を規定するためには、START WITH 句と CONNECT BY 句を使用しなければなりません。LEVEL 疑似列の使用については、「SELECT コマンド」（4-486 ページ）を参照してください。

## ROWID

データベース内の各行について、ROWID 疑似列により行のアドレスが戻されます。Oracle8 の ROWID 値には、行を捜し出すために必要な次の情報が含まれています。

- オブジェクトのデータ・オブジェクト番号

- データ・ファイル内のデータ・ブロック
- データ・ブロック内の行 (最初の行は 0)
- データ・ファイル (最初のファイルは 1)。ファイル番号は表領域に対して相対的です。

ほとんどの場合、**ROWID** 値ではデータベース内の行は一意に識別されます。ただし、同じクラスタに格納されている異なる表の行には、同じ **ROWID** を持つことができます。

**ROWID** 疑似列の値はデータ型 **ROWID** を持ちます。**ROWID** データ型については、「**ROWID** データ型」(2-16 ページ)を参照してください。

**ROWID** 値には、次の重要な用途があります。

- 単一の行をアクセスする最も速い方法。
- 表の行が格納されている様子を示すことができる。
- 表の行に対する一意の識別子。

表の主キーとして **ROWID** を使用しないでください。たとえば、**Import** ユーティリティと **Export** ユーティリティで行を削除してから再挿入する場合、**ROWID** は変わる場合があります。行を削除した場合、**Oracle** は、後から新たに挿入される行にその **ROWID** を再度割り当てる可能性があります。

問合せの **SELECT** 句と **WHERE** 句で **ROWID** 疑似列を使用できますが、これらの疑似列の値が実際にデータベースに格納されるわけではありません。**ROWID** 疑似列の値に対して挿入および更新、削除はできません。

**例** 次の文は、部門 20 の従業員のデータを含むすべての行のアドレスを選択します。

```
SELECT ROWID, ename
      FROM emp
     WHERE deptno = 20;
```

ROWID	ENAME
-----	-----
AAAAfSAABAAAClaAAA	SMITH
AAAAfSAABAAAClaAAD	JONES
AAAAfSAABAAAClaAAH	SCOTT
AAAAfSAABAAAClaAAK	ADAMS
AAAAfSAABAAAClaAAM	FORD

## ROWNUM

問合せによって戻される各行について、**ROWNUM** 疑似列により表や結合処理された行の集合から **Oracle** が行を選択する順序を示す番号が戻されます。つまり、選択される最初の行の **ROWNUM** は 1、2 番目の行の **ROWNUM** は 2 です (以下同様です)。

次の例のように、**ROWNUM** を使用して問合せによって戻される行数を制限できます。



```
SELECT *  
  FROM emp  
 WHERE ROWNUM < 10;
```

比較条件「**ROWNUM 値**＞正の整数」は常に偽となるので注意してください。たとえば、次の問合せでは行は戻されません。

```
SELECT * FROM emp  
 WHERE ROWNUM > 1;
```

最初にフェッチされる行の **ROWNUM** には1が割り当てられるため、条件は偽と判定されます。さらに、2番目にフェッチされる予定であった行は最初の行になるため、この **ROWNUM** にも1が割り当てられ、条件も偽と判定されます。このように、後続するすべての行が条件を満たさないため、行はまったく戻されません。

また、次の例のように、**ROWNUM** を使用して表の各行に一意の値を割り当てることもできます。

```
UPDATE tabx  
  SET coll = ROWNUM;
```

Oracle は検索した各行に **ROWNUM** 値を割り当ててから、**ORDER BY** 句に従って行をソートします。そのため、**ORDER BY** 句が各行の **ROWNUM** に影響を及ぼすことはありません。ただし、**ORDER BY** 句の指定によって Oracle が索引を使用してデータにアクセスする場合、索引なしの場合とは異なる順序で行が取り出されることがあります。そのため、**ROWNUM** は **ORDER BY** 句なしの場合とは異なる値になることもあります。

---

---

**注意：** 問合せで **ROWNUM** を使用すると、ビューの最適化に影響を及ぼすことがあります。詳細は、『Oracle8 概要』を参照してください。

---

---

## コメント

SQL 文とスキーマ・オブジェクトに対してコメントを付けることができます。

### SQL 文中のコメント

SQL 文中のコメントは、文の実行には影響を及ぼしませんが、アプリケーションを読みやすく、メンテナンスしやすくなります。文にはアプリケーションでのその文の目的を記述したコメントを含めることができます。

コメントは、文中のキーワードまたはパラメータ、句読点の間に入れることができます。次のどちらかの方法を使用します。

- /\* を使用してコメントを開始する。コメントのテキストを続けます。このテキストは複数行に及んでもかまいません。\*/ でコメントを終了します。開始文字と終了文字は、空白や改行によってテキストから切り離す必要はありません。

- --(ハイフン2個)を使用してコメントを開始する。コメントのテキストを続けます。このテキストは複数行にまたがることはできません。改行によってコメントを終了します。

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

---

---

**注意：** SQL スクリプト内の SQL 文では、このようなスタイルのコメントを使用できません。この場合、Server Manager または SQL\*Plus の REMARK コマンドを使用してください。これらのコマンドの説明は、『Oracle Server Manager User's Guide』または『SQL\*Plus ユーザーズ・ガイドおよびリファレンス』を参照してください。

---

---

**例** 次の文には多くのコメントが含まれています。

```
SELECT ename, sal + NVL(comm, 0), job, loc
/* Select all employees whose compensation is
greater than that of Jones.*/
FROM emp, dept
    /*The DEPT table is used to get the department name.*/
WHERE emp.deptno = dept.deptno
    AND sal + NVL(comm,0) >    /* Subquery: */
    (SELECT sal + NLV(comm,0)
                                /* total compensation is sal + comm */
FROM emp
WHERE ename = 'JONES')

SELECT ename, -- select the name
    sal + NVL(comm, 0),          -- total compensation
    job,                        -- job
    loc -- and city containing the office
FROM emp, -- of all employees
    dept
WHERE emp.deptno = dept.deptno
    AND sal + NVL(comm, 0) >    -- whose compensation
                                -- is greater than
    (SELECT sal + NVL(comm,0)    -- the compensation
FROM emp
WHERE ename = 'JONES')        -- of Jones.
```

## スキーマ・オブジェクトに関するコメント

このマニュアルの第4章「コマンド」に記述されている COMMENT コマンドを使用して、表またはビュー、スナップショット、列にコメントを付けることができます。スキーマ・オブジェクトに付けたコメントは、データ・ディクショナリに格納されます。

## ヒント

Oracle オプティマイザに指示（すなわちヒント）を引き渡すために、SQL 文中でコメントを使用できます。オプティマイザは、これらのヒントを使用して文の実行計画を選択します。

文ブロックには、ヒントを含んだコメントは1つしか指定できません。このコメントは、SELECT または UPDATE、INSERT、DELETE のいずれかのキーワードの後でだけ指定できます。次の構文は、Oracle が文ブロック内でサポートする両方のスタイルのコメントに含まれるヒントの構文です。

```
{DELETE|INSERT|SELECT|UPDATE} /*+ hint [text] [hint[text]]... */
```

または

```
{DELETE|INSERT|SELECT|UPDATE} ---+ hint [text] [hint[text]]...
```

ここで、それぞれの意味は次のとおりです。

DELETE	文ブロックを始める DELETE または INSERT、SELECT、
INSERT	UPDATE のいずれかのキーワードです。ヒントを含むコメントは、これらのキーワードの後でだけ指定できます。
SELECT	
UPDATE	
+	Oracle にコメントをヒントのリストとして解釈させるようにするためのプラス記号です。プラス記号は、コメント・デリミタの直後（空白は認められない）になければなりません。
<i>hint</i>	この項と『Oracle8 Server チューニング』で説明するヒントの1つです。プラス記号とヒントの間の空白は入れても入れなくてもかまいません。コメントに複数のヒントが含まれている場合は、ヒントの対ごとに1つ以上の空白で区切る必要があります。
<i>text</i>	ヒントに含めることができるその他のコメント・テキストです。

表 2-10 に、ヒントの構文と説明を示します。ヒントの詳細は、『Oracle8 Server チューニング』および『Oracle8 Parallel Server 概要および管理』、『Oracle8 Server 概要』を参照してください。

表 2-10 ヒントの構文と説明

ヒントの構文	説明
最適化方法と目標	
<code>/*+ ALL_ROWS */</code>	最高のスループットを実現する（つまり、リソース使用の合計を最小限にする）という目標で文ブロックを最適化するために、コストベース方法を明示的に選択する。
<code>/*+ CHOOSE */</code>	各 SQL 文について、ルールベース方法とコストベース方法のいずれかをオプティマイザが選択する。どちらを選択するかは、この文による表アクセスの統計情報を利用する。
<code>/*+ FIRST_ROWS */</code>	最良の応答時間を実現する（つまり、最初の行を戻すために使用するリソースを最小限にする）という目標で文ブロックを最適化するために、コストベース方法を明示的に選択する。
<code>/*+ RULE */</code>	文ブロックのためにルールベース最適化を明示的に選択する。
アクセス方法のヒント	
<code>/*+ AND_EQUAL(table index) */</code>	複数の単一列索引の走査結果をマージするアクセス・パスを使う実行計画を明示的に選択する。
<code>/*+ CLUSTER(table) */</code>	指定された表にアクセスするためにクラスタ走査を明示的に選択する。
<code>/*+ FULL(table) */</code>	指定された表に対する全表走査を明示的に選択する。
<code>/*+ HASH(table) */</code>	指定された表にアクセスするためにハッシュ走査を明示的に選択する。
<code>/*+ HASH_AJ(table) */</code>	指定された表にアクセスするために、NOT IN 副問合せをハッシュ非結合に変換する。
<code>/*+ HASH_SJ (table) */</code>	指定された表にアクセスするために、NOT IN 副問合せをハッシュ非結合に変換する。
<code>/*+ INDEX(table index) */</code>	指定された表に対する索引走査を明示的に選択する。
<code>/*+ INDEX_ASC(table index) */</code>	指定された表に対する昇順範囲索引走査を明示的に選択する。
<code>/*+ INDEX_COMBINE(table index) */</code>	索引が INDEX_COMBINE ヒントの引数として指定されていない場合は、推定コストが最良であるビットマップ索引の組合せをオプティマイザが使用する。特定の索引が引数として指定されている場合は、オプティマイザは、指定されたビットマップ索引の組合せを次々試す。
<code>/*+ INDEX_DESC(table index) */</code>	指定された表に対する降順範囲索引走査を明示的に選択する。
<code>/*+ INDEX_FFS(table index) */</code>	全表走査ではなく、高速全索引走査を実行する。

表 2-10 ヒントの構文と説明

ヒントの構文	説明
<code>/*+ MERGE_AJ (table) */</code>	指定された表にアクセスするために、NOT IN 副問合せをマージ非結合に変換する。
<code>/*+ MERGE_SJ (table) */</code>	指定された表にアクセスするために、相関 EXISTS 副問合せをマージ半結合に変換する。
<code>/*+ ROWID(table) */</code>	指定された表に対する ROWID での表走査を明示的に選択する。
<code>/*+ USE_CONCAT */</code>	問合せの WHERE 句内の結合 OR 条件を、UNION ALL 集合演算子を使って複合問合せに変換する。
<b>結合順序のヒント</b>	
<code>/*+ ORDERED */</code>	表が FROM 句に指定されている順序で結合されるようにする。
<code>/*+ STAR */</code>	索引のネストされたループの結合を使うことにより、大きな表が最後に結合されるようにする。
<b>結合操作のヒント</b>	
<code>/*+ DRIVING_SITE (table) */</code>	問合せ処理が Oracle により選択されたサイトとは異なるサイトで実行されるようにする。
<code>/*+ USE_HASH (table) */</code>	ハッシュ結合を使って、指定された各表を別の行ソースに結合させる。
<code>/*+ USE_MERGE (table) */</code>	ソート・マージ結合を使って、指定された各表を別の行ソースに結合させる。
<code>/*+ USE_NL (table) */</code>	ネストされたループの結合を使って、指定された各表を別の行ソースに結合させる。この場合、指定された表を内部表として使用する。
<b>パラレル実行のヒント</b>	
<code>/*+ APPEND */</code>	データが単に表に追加される（または追加されない）ように指定する。これにより、既存の空き領域は使用されない。これらのヒントは INSERT キーワードの後でだけ使用する。
<code>/*+ NOAPPEND */</code>	
<code>/*+ NOPARALLEL(table) */</code>	表が PARALLEL 句を使って作成された場合でも、表のパラレル走査を使用できないようにする
<code>/*+ PARALLEL(table, instances) */</code>	操作に使用できる同時発生のスレーブ・プロセスの数を希望の数に指定できるようにする。  DELETE および INSERT、UPDATE の各操作は、セッションが PARALLEL DML 使用可能モードであるときにだけ、パラレル化とみなされる。（このモードに入るには、ALTER SESSION PARALLEL DML を使用する。）

表 2-10 ヒントの構文と説明

ヒントの構文	説明
<code>/*+ PARALLEL_INDEX */</code>	PARALLEL 属性を持つ分割された索引および分割されていない索引に対して高速全索引操作をパラレル化できるようにする。
<code>/*+ NOPARALLEL_INDEX */</code>	索引に対する PARALLEL 属性の設定を上書きする。
その他のヒント	
<code>/*+ CACHE */</code>	全表操作が実行されると、ヒント内の表のために取り出されたブロックは、バッファ・キャッシュ内の LRU リストの最大使用頻度の場所に置かれる。
<code>/*+ NOCACHE */</code>	全表操作が実行されると、この表のために取り出されたブロックは、バッファ・キャッシュ内の LRU リストの最低使用頻度の場所に置かれる。
<code>/*+ MERGE (table) */</code>	周囲の問合せの前に複雑なビューまたは副問合せを評価するようにする。
<code>/*+ NO_MERGE (table) */</code>	マージ可能なビューをマージしないようにする。
<code>/*+ PUSH_JOIN_PRED (table) */</code>	オプティマイザが、個々の結合述語をビューにプッシュするかどうかを、コスト・ベースで評価するようにする。
<code>/*+ NO_PUSH_JOIN_PRED (table) */</code>	結合述語のビューへのプッシュをしないようにする。
<code>/*+ PUSH_SUBQ */</code>	マージされていない副問合せを、実行計画の中で可能な限り最も早く評価する。
<code>/*+ STAR_TRANSFORMATION */</code>	型変換が適用された最適な計画がオプティマイザにより使用されるようにする。

## データベース・オブジェクト

### スキーマ・オブジェクト

スキーマは、論理的なデータの構造またはスキーマ・オブジェクトの集まりです。スキーマは、データベース・ユーザーによって所有され、そのユーザーと同じ名前を持ちます。各ユーザーは、1つのスキーマを所有します。スキーマ・オブジェクトは、SQLを使って作成したり操作したりできます。スキーマ・オブジェクトには次のタイプのオブジェクトがあります。

- クラスタ
- データベース・リンク
- データベース・トリガー
- 外部プロシージャ・ライブラリ

- 索引構成表
- 索引
- パッケージ
- 順序
- ストアド・ファンクション
- ストアド・プロシージャ
- シノニム
- 表
- ビュー

さらに、Oracle の分散オプションを使っている場合には、次のスキーマ・オブジェクトが利用できます。

- スナップショット
- スナップショット・ログ

さらに、データベース・サーバーにオブジェクト・オプションがインストールされている場合には、次のスキーマ・オブジェクトが利用できます。

- オブジェクト表
- オブジェクト型
- オブジェクト・ビュー

## 非スキーマ・オブジェクト

また、次のタイプのオブジェクトもデータベースに格納され、SQL で作成、操作されますが、スキーマには含まれません。

- ディレクトリ
- プロファイル
- ロール
- ロールバック・セグメント
- 表領域
- ユーザー

各タイプのオブジェクトは、このマニュアルの第4章「コマンド」にある、データベース・オブジェクトを作成するコマンドの項で簡単に定義されています。これらのコマンドは、キーワード CREATE で始まります。たとえば、クラスタの定義については、「CREATE

CLUSTER コマンド」(4-202 ページ)を参照してください。データベース・オブジェクトの概要については、『Oracle8 概要』を参照してください。

大部分のスキーマ・オブジェクトでは、作成時に名前を指定する必要があります。名前は、以降の項に示す規則に従って付けてください。

## スキーマ・オブジェクトの部分

スキーマ・オブジェクトの中には、次に示すように名前を付ける必要のある部分から構成されるものもあります。

- 表やビューの中の列
- 索引と表のパーティション
- 表に対する整合性制約
- パッケージ・プロシージャおよびパッケージ・ストアド・ファンクション、パッケージ内に格納されるその他のオブジェクト

### パーティション表と索引

表と索引はパーティション化できます。パーティション化されたスキーマ・オブジェクトは、パーティションと呼ばれる多数の部分で構成され、各パーティションの論理属性はすべて同じです。たとえば、表のパーティションはすべて同じ列定義と制約定義を共有し、索引のパーティションはすべて同じ索引列を共有します。

### 拡張パーティション表名

パーティションは表として使用できます。このため、一部のパーティション・レベルの操作を簡単に実行できます。他の方法では WHERE 句の述語が必要となります。

パーティションを表として使用するには、単一のパーティションからデータを選択してビューを作成し、そのビューを表として使用します。この方法の利点は、これらのビューに対する権限を他のユーザーやロールに付与する（または取り消す）ことによって、パーティション・レベルのアクセス制御機構を構築できる点です。

パーティション・レベルの大量データの操作（たとえば、パーティションからすべての行を削除する操作など）は、1つのパーティションに対する操作だけに限られます。このようなタイプの操作は、表名の構文で簡単に表されます。同じ操作を WHERE 句の述語で表そうとすると、特にレンジ・パーティション・キーで複数の列を使用しているときに、とても複雑になる可能性があります。

次の DML 文では、表指定の構文が拡張され、リモートでないパーティション表に対するオプションのパーティション指定ができるようになりました。

- DELETE
- INSERT
- LOCK TABLE



- SELECT
- UPDATE

---

**注意：** アプリケーションの移植性と ANSI 構文準拠を考慮し、この Oracle 特有の拡張部分からアプリケーションを切り離すためにはビューを使用することをお勧めします。

---

現在、拡張パーティション表名を使用するときには、次の制限があります。

- リモート表なし：拡張パーティション表名には、データベース・リンク (dblink)、または dblink を使って表に変換するシノニムを含めることはできません。リモート・パーティションを使用するには、拡張パーティション表名の構文を使ってリモート・サイトにビューを作成し、そのリモート・ビューを参照します。
- ダイレクト PL/SQL サポートなし：拡張パーティション表名の構文を使用する SQL 文は、DBMS\_SQL パッケージを使って動的 SQL を介して使用できますが、PL/SQL ブロック内では使用できません。PL/SQL ブロック内のパーティションを参照するには、拡張パーティション表名の構文を使用するビューを使用します。
- シノニムなし：パーティション拡張は、実表を使って指定する必要があります。シノニムまたはビュー、その他のオブジェクトは使用できません。

**構文** 拡張パーティション表名を使用する基本構文は次のとおりです。

```
[schema.]{table | view} [@dblink | PARTITION (partition_name)]
```

**例** 次の文では、SALES がパーティション JAN97 を持つパーティション表です。シングル・パーティション JAN97 のビューを作成でき、それをあたかも表のように使用できます。この例では、パーティションから行が削除されます。

```
CREATE VIEW sales_jan97 AS
  SELECT * FROM sales PARTITION (jan97);
DELETE FROM sales_jan97 WHERE amount < 0;
```

## スキーマ・オブジェクト名および修飾子

この項では、次の内容について説明します。

- スキーマ・オブジェクトとスキーマ・オブジェクト位置修飾子の命名規則
- スキーマ・オブジェクトと修飾子の命名ガイドライン

### スキーマ・オブジェクトの命名規則

スキーマ・オブジェクトの命名には次の規則が適用されます。

1. 名前は 1 文字から 30 文字までの長さでなければなりません。ただし、次の 2 つは例外です。
  - データベースの名前は、8 文字までに制限されています。
  - データベース・リンクの名前は、128 文字まで指定できます。
2. 名前には引用符を含むことができません。
3. 名前は大文字と小文字を区別しません。
4. 名前は二重引用符で囲まれていない限り、使用しているデータベースのキャラクタ・セットのアルファベット文字で開始しなければなりません。
5. 名前は使用しているデータベースのキャラクタ・セットの英数字の文字と記号 (、\$、#) だけを含むことができます。データベース・リンクの名前は、ピリオド (.) とアットマーク (@) を含むこともできます。\$ と # はできるだけ使用しないでください。

データベースのキャラクタ・セットがマルチバイト文字を含む場合、ユーザーまたはロールの名前にはシングルバイト文字を最低 1 つ含めることをお勧めします。

---

**注意：** データベース名またはグローバル・データベース名、データベース・リンク名にはヨーロッパまたはアジアのキャラクタ・セットの中の特  
殊文字は使用できません。たとえば、ウムラートを使用することはできま  
せん。

---

6. 名前には、Oracle の予約語は使用できません。Oracle の予約語の全リストは、付録 C「Oracle の予約語とキーワード」を参照してください。  
  
名前は、データベース・オブジェクトにアクセスするために使用する Oracle 製品固有のその他の予約語によって、さらに制限されることもあります。製品の予約語のリストについては、『PL/SQL ユーザーズ・ガイドおよびリファレンス』などの各製品のマニュアルを参照してください。
7. DUAL という語をオブジェクトまたはオブジェクトの部分の名前として使用しないでください。DUAL は、ダミー表の名前です。
8. Oracle の SQL 言語には、特別な意味を持つキーワードが含まれています。これらのキーワードは予約語ではないため、オブジェクトおよびオブジェクトの部分の名前として使用できます。しかし、キーワードを名前として使用すると、SQL 文が読みにくいものになります。  
  
Oracle のキーワードのリストは、付録 C「Oracle の予約語とキーワード」を参照してください。
9. 名前領域内では、2 つのオブジェクトに同じ名前を付けることはできません。

図 2-2 は、スキーマ・オブジェクトの名前領域を示します。表とビューは同じ名前領域に存在するため、同じスキーマ内の表とビューには同じ名前を付けることはできませ

ん。しかし、表と索引は異なる名前領域に存在するため、同じスキーマ内の表と索引には同じ名前を付けることができます。

データベース内の各スキーマには、その中のオブジェクトのために固有の名前領域があります。たとえば、異なるスキーマ内の2つの表は異なる名前領域に存在し、同じ名前を付けることができます。

図 2-2 スキーマ・オブジェクトの名前領域

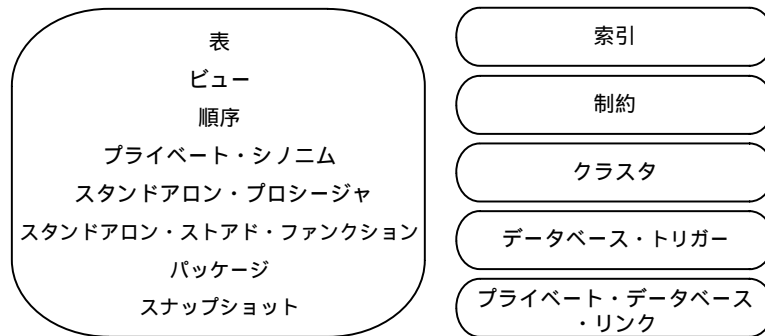
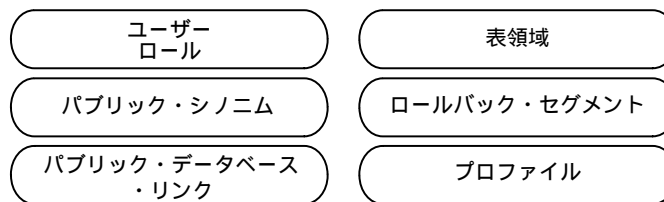


図 2-3 は、非スキーマ・オブジェクトの名前領域を示します。これらの名前領域内のオブジェクトはスキーマに含まれないため、これらの名前領域はデータベース全体で使用されます。

図 2-3 非スキーマ・オブジェクトの名前領域



10. 同じ表やビューでは、複数の列に同じ名前を付けることはできません。しかし、異なる表やビューでは、複数の列に同じ名前を付けることができます。
11. 同じパッケージに含まれるプロシージャや関数には、引数の数やデータ型を変えることによって、同じ名前を付けることができます。異なる引数を持つ、同じ名前のプロシージャや関数を同じパッケージ内に複数作成することを、オーバーロードと言います。
12. 名前は、二重引用符で囲むことができます。その場合、このリストの規則3から規則7にとらわれずに、名前には空白も含めた任意の文字の組合せを使用できます。移植性を

持たせるためにこのような例外が認められていますが、規則 3 から規則 7 に違反しないことをお勧めします。

二重引用符で囲んだスキーマ・オブジェクト名を使用した場合、そのオブジェクトを参照するときには常に二重引用符を使用しなければなりません。

次に示すような場合には、名前を二重引用符で囲みます。

- 空白を含める。
- 大文字と小文字を区別する。
- 数字のようなアルファベット以外の文字で名前を開始する。
- 英数字および `_`、`$`、`#` 以外の文字を名前に含める。
- 予約語を名前として使用する。

名前を二重引用符で囲むことによって、同じ名前領域内の異なるオブジェクトに対して次の名前を指定できます。

```
emp
"emp"
"Emp"
"EMP "
```

しかし、Oracle は次の名前を同じ名前として解釈するため、同じ名前領域内の異なるオブジェクトには次の名前を使用できません。

```
emp
EMP
"EMP"
```

ユーザーまたはパスワードに引用符で囲んだ名前を付ける場合、その名前に小文字を含めることはできません。

データベース・リンク名を引用符で囲むことはできません。

## スキーマ・オブジェクトの命名例

次に、有効なスキーマ・オブジェクト名の例を示します。

```
ename
horse
scott.hiredate
"EVEN THIS & THAT!"
a_very_long_and_valid_name
```

列別名および表別名、ユーザー名、パスワードはオブジェクトやオブジェクトの部分ではありませんが、同様にこれらの命名規則に従わなければなりません。ただし、次のような例外があります。

- 列別名と表別名は、単一の SQL 文の実行時に存在するだけで、データベースには格納されないため、規則 12 は適用されません。
- パスワードには名前領域がないので、規則 9 は適用されません。
- ユーザー名とパスワードで大/小文字を区別するために引用符を使用しないでください。ユーザー名とパスワードの命名規則の詳細は、「CREATE USER コマンド」(4-353 ページ)を参照してください。

## スキーマ・オブジェクト名の命名ガイドライン

オブジェクトとその部分に名前を付ける場合に役立つガイドラインを次に示します。

- わかりやすい名前（またはよく知られている省略形）を使用する。
- 一貫した命名規則を使用する。
- 複数の表にまたがる同一のエンティティや属性を記述するためには、同一の名前を使用する。

オブジェクトに名前を付けるときには、短く使いやすい名前とわかりやすい名前のバランスを考えてください。迷ったときには、わかりやすい名前を選択してください。データベース内のオブジェクトは、多くの人々が長期間にわたって使用する可能性があるからです。PAYMENT\_DUE\_DATE のかわりに PMDD という名前を使用すると、10 年後の担当者はデータベースを理解するのに苦労することになるでしょう。

一貫した命名規則を使用すると、アプリケーション上の各表の働きが理解しやすくなります。そのような規則の 1 つの例として、FINANCE アプリケーションに属している表の名前をすべて FIN\_ で始めるような場合が考えられます。

同一のエンティティや属性に対しては、複数の表にまたがっていても同じ名前を使用してください。たとえば、EMP 表と DEPT 表の部門番号列には、どちらにも DEPTNO という名前を付けます。

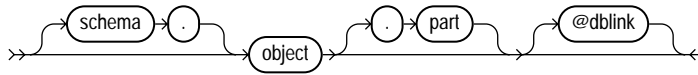
## スキーマ・オブジェクトと部分を参照する

SQL 文のコンテキストでスキーマ・オブジェクトとそれらの部分を参照する方法について説明します。次の項目について説明します。

- オブジェクトを参照するための一般的な構文
- Oracle がオブジェクトへの参照を解決する方法
- 自分のスキーマ以外のスキーマ内のオブジェクトを参照する方法
- リモート・データベース内のオブジェクトを参照する方法

次の構文図は、オブジェクトや部分を参照するための一般的な構文を示しています。

**schema\_object ::=**



ここで、それぞれの意味は次のとおりです。

**object** オブジェクトの名前です。

**schema** オブジェクトを含むスキーマです。この修飾子を指定することによって、自分のスキーマ以外のスキーマ内のオブジェクトを参照できます。その場合には、自分のスキーマ以外のスキーマ内のオブジェクトを参照するための権限が付与されていなければなりません。この修飾子を指定しないと、自分自身のスキーマ内のオブジェクトを参照するものとみなされます。

スキーマ・オブジェクトだけが *schema* で修飾できます。スキーマ・オブジェクトを図 2-2 (2-45 ページ) に示します。図 2-3 (2-45 ページ) に示す非スキーマ・オブジェクトはスキーマ・オブジェクトではないため、*schema* では修飾できません。ただし、パブリック・シノニムは例外で、"PUBLIC"(引用符が必要)で修飾できます。

**part** オブジェクトの部分です。この識別子によって、スキーマ・オブジェクトの部分（たとえば、表の列またはパーティション）を参照できます。なお、すべてのタイプのオブジェクトが部分を持っているわけではありません。

**dblink** Oracle の分散オプションを使用している場合にだけ適用されます。オブジェクトを含んでいるデータベースの名前です。この修飾子 *dblink* を指定することによって、ローカル・データベース以外のデータベース内のオブジェクトを参照できます。この *dblink* を指定しないと、自分自身のローカル・データベース内のオブジェクトを参照するものとみなされます。なお、すべての SQL 文でリモート・データベースのオブジェクトにアクセスできるとは限りません。

オブジェクトを参照するコンポーネントを区切っているピリオドの前後には、空白を入れることができます。しかし、通常は入れません。

## Oracle がスキーマ・オブジェクト参照を解決する方法

SQL 文内のオブジェクトが参照される場合、Oracle はその SQL 文のコンテキストを検討して、該当する名前領域内でそのオブジェクトを突き止めます。そのオブジェクトを突き止め

てから、そのオブジェクトに対して文の操作を実行します。指定した名前のオブジェクトが適切な名前領域内に存在しない場合、Oracle はエラー・メッセージを戻します。

次の例で、Oracle が SQL 文内のオブジェクト参照を解決する方法について説明します。名前 DEPT で識別される表にデータ行を追加する次の文を考えます。

```
INSERT INTO dept
VALUES (50, 'SUPPORT', 'PARIS');
```

文のコンテキストに基づいて、Oracle は DEPT が次のようなオブジェクトであると判断します。

- 自分のスキーマ内の表
- 自分のスキーマ内のビュー
- 表またはビューに対するプライベート・シノニム
- パブリック・シノニム

Oracle は、文を発行したユーザーのスキーマ外の名前領域を考慮する前に、そのユーザーのスキーマ内の名前領域からオブジェクト参照を解決しようとします。この例では、Oracle は次の方法で名前 DEPT を解決しようとします。

1. Oracle では最初に、表およびビュー、プライベート・シノニムを含む、文を発行したユーザーのスキーマ内の名前領域でオブジェクトを突き止めようとします。オブジェクトがプライベート・シノニムの場合、Oracle はそのシノニムが表すオブジェクトを突き止めます。このオブジェクトは、ユーザー自身のスキーマまたは他のスキーマ、他のデータベースにあることもあります。このオブジェクトが別のシノニムであることもあります。その場合、Oracle はそのシノニムが表すオブジェクトを突き止めます。
2. オブジェクトが名前領域に存在する場合、Oracle はそのオブジェクトに対して文を実行しようとします。この場合、Oracle はデータの行を DEPT に追加しようとします。オブジェクトがその処理にとって正しい型でない場合、Oracle はエラー・メッセージを戻します。この場合、DEPT は、表またはビュー、あるいは表またはビューとなるプライベート・シノニムでなければなりません。DEPT が順序であると、Oracle はエラー・メッセージを戻します。
3. 上記の処理で検索された名前領域にオブジェクトが存在しない場合、Oracle はパブリック・シノニム (図 2-3 (2-45 ページ) を参照) を含む名前領域を検索します。オブジェクトがその名前領域に存在すると、Oracle はそのオブジェクトに対して文を実行しようとします。オブジェクトがその処理にとって正しい型でない場合、Oracle はエラー・メッセージを戻します。この例では、DEPT が順序のパブリック・シノニムである場合、Oracle はエラー・メッセージを戻します。

## 他のスキーマ内のオブジェクトを参照する

自分が所有するスキーマ以外のスキーマ内のオブジェクトを参照するには、次のように、オブジェクト名の前にスキーマ名を付けます。

`schema.object`

たとえば、次の文は、スキーマ **SCOTT** 内の **EMP** 表を削除します。

```
DROP TABLE scott.emp
```

## リモート・データベース内のオブジェクトを参照

ローカル・データベース以外のデータベース内のオブジェクトを参照するには、オブジェクト名にそのデータベースへのデータベース・リンクの名前を続けます。データベース・リンクはスキーマ・オブジェクトであり、これによって Oracle がリモート・データベースに接続され、そこのオブジェクトにアクセスします。この項では次の項目について説明します。

- データベース・リンクを作成する方法
- SQL 文でデータベース・リンクを使用する方法

### データベース・リンクを作成する

第4章「コマンド」で説明する **CREATE DATABASE LINK** コマンドを使用して、データベース・リンクを作成できます。このコマンドでは、データベース・リンクに関する次の情報を指定できます。

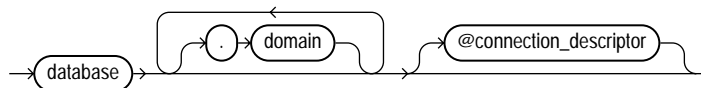
- データベース・リンクの名前
- リモート・データベースにアクセスするためのデータベース接続文字列
- リモート・データベースへ接続するためのユーザー名とパスワード

これらの情報はデータ・ディクショナリに格納されます。

**データベース・リンク名** データベース・リンクを作成するとき、データベース・リンク名を指定する必要があります。データベース・リンク名は 128 バイト以内の長さで指定し、ピリオド (.) とアットマーク (@) を使用できます。このように、データベース・リンク名は、他のオブジェクト型の名前とは異なります。

データベース・リンクに付ける名前には、データベース・リンクが参照するデータベースの名前、およびデータベース名の階層内でそのデータベースの位置に一致しなければなりません。データベース・リンク名の書式を次の構文図に示します。

**dblink::=**





ここで、それぞれの意味は次のとおりです。

<i>database</i>	データベース・リンクが接続するリモート・データベースの名前を指定します。リモート・データベースの名前は、初期化パラメータ <code>DB_NAME</code> によって指定されます。
<i>domain</i>	データベース・リンクが接続するリモート・データベースのドメインを指定します。データベース・リンクの名前にドメインを指定しないと、Oracle は現在データ・ディクショナリに存在しているローカル・データベースのドメイン名をデータベースに付加し、そのリンク名をデータ・ディクショナリに格納します。
<i>connect_descriptor</i>	データベース・リンクをさらに修飾できます。接続修飾子を使用すると、同じデータベースに複数のデータベース・リンクを作成できます。たとえば、接続修飾子を使用して、同じデータベースにアクセスする Oracle パラレル・サーバーの異なるインスタンスに複数のデータベース・リンクを作成できます。

*database.domain* の組合せは、「サービス名」と呼ばれることもあります。詳細は、『Net8 管理者ガイド』を参照してください。

**ユーザー名とパスワード** リモート・データベースへ接続するためにユーザー名とパスワードを使用します。データベース・リンクでは、ユーザー名とパスワードはオプションです。

**データベース接続文字列** データベース接続文字列は、Net8 によりリモート・データベースにアクセスするために使用される仕様です。データベース接続文字列の記述方法については、使用しているネットワーク・プロトコル用の Net8 のマニュアルを参照してください。データベース・リンク用のデータベース文字列はオプションです。

## データベース・リンクを参照する

データベース・リンクは、分散オプションを指定して Oracle を使用している場合しか利用できません。データベース・リンクを含む SQL 文の発行時に、次のいずれかの書式でデータベース・リンク名を指定します。

完全指定	データ・ディクショナリ内に格納される、データベースおよびドメイン、オプションの接続修飾子を含む完全なデータベース・リンク名を指定します。
部分指定	データベースとオプションの接続修飾子コンポーネントを含むが、ドメイン・コンポーネントを含まないように指定します。

Oracle は、リモート・データベースに接続する前に次のタスクを実行します。

1. 文中に指定されているデータベース・リンク名が部分指定の場合、Oracle は、データ・ディクショナリ内に格納されているグローバル・データベース名に見られるように、そのリンク名にローカル・データベースのドメイン名を付加します。現在のグローバル・データベース名は、GLOBAL\_NAME データ・ディクショナリ・ビューで見ることができます。
2. Oracle は最初に、文を発行したユーザーのスキーマ内で、文の中のデータベース・リンクと同じ名前を持つプライベート・データベース・リンクを検索し、必要であれば同じ名前を持つパブリック・データベース・リンクを検索します。
  - Oracle は必ず、最初に一致したデータベース・リンク（プライベートまたはパブリック）のユーザー名とパスワードを採用します。最初に一致したデータベース・リンクに対応付けられているユーザー名およびパスワードがあると、Oracle はそれを使用します。対応付けられているユーザー名およびパスワードがない場合、Oracle は現在のユーザー名とパスワードを使用します。
  - 最初に一致したデータベース・リンクに対応付けられたデータベース文字列が存在する場合、Oracle はそのデータベース文字列を使用します。データベース文字列がない場合、Oracle は一致する次の（パブリック）データベース・リンクを検索します。一致するデータベース・リンクが存在しない場合、または一致するリンクに対応付けられているデータベース文字列が存在しない場合、Oracle はエラー・メッセージを戻します。
3. Oracle は、リモート・データベースにアクセスするためにデータベース文字列を使用します。リモート・データベースにアクセスした後で、Oracle は次の条件が両方とも満たされているか確認します。
  - リモート・データベースの名前（初期化パラメータ DB\_NAME によって指定される）がデータベース・リンク名のデータベース・コンポーネントと一致する。
  - リモート・データベースのドメイン（初期化パラメータ DB\_DOMAIN によって指定される）が、データベース・リンク名のドメイン・コンポーネントと一致する。これらの条件が両方とも満たされている場合、Oracle はステップ 2 で選択したユーザー名とパスワードを使用して接続を続行します。それ以外の場合は、Oracle はエラー・メッセージを戻します。
4. データベース文字列およびユーザー名、パスワードを使用した接続が成功すると、Oracle はリモート・データベース上の指定されたオブジェクトにアクセスしようとします。このとき、この項の前半で説明した、オブジェクト参照を解決するための規則および他のスキーマ内のオブジェクトを参照するための規則が使用されます。

リモート・オブジェクトに対する Oracle の名前の解決は、初期化パラメータ GLOBAL\_NAMES、および ALTER SYSTEM コマンドと ALTER SESSION コマンドの GLOBAL\_NAMES パラメータを使用して、使用可能または使用禁止にできます。

リモートでの名前の解決の詳細は、『Oracle8 Server 分散システム』を参照してください。

## オブジェクト型の属性とメソッドの参照

SQL 文でオブジェクト型の属性とメソッドを参照するには、参照を表別名で完全に修飾しなければなりません。次に例を示します。

```
CREATE TYPE person AS OBJECT
  (ssno VARCHAR(20),
   name VARCHAR (10));

CREATE TABLE emptab (pinfo person);
```

次に示すように、SQL 文では、SSNO 属性への参照は表別名を使用して完全に修飾しなければなりません。

```
SELECT e.pinfo.ssno FROM emptab e;

UPDATE emptab e SET e.pinfo.ssno = '510129980'
  WHERE e.pinfo.name = 'Mike';
```

引数を取らないオブジェクト型のメンバー・メソッドを参照する場合は、空のカッコを付けなければなりません。たとえば、AGE が個人型の中のメソッドで、引数を取らないとします。SQL 文でこのメソッドを呼び出すには、次の例のように、空のカッコを付けなければなりません。

```
SELECT e.pinfo.age() FROM emptab e
  WHERE e.pinfo.name = 'Mike';
```

詳細は、『Oracle8 概要』のユーザー定義データ型の項を参照してください。



---

## 演算子、関数、式、条件

個々のデータ項目を操作する方法について説明します。加算や減算などの標準的な算術演算子に加え、絶対値や文字列の長さを戻すようなあまり一般的でない関数についても説明します。説明する内容は次のとおりです。

- 演算子
- SQL 関数
- ユーザー関数
- 書式モデル
- 式
- 条件

---

**注意：** **OBJ**が前についている関数および式、説明は、Oracle Object Option がデータベース・サーバーにインストールされている場合にだけ有効です。

---

### 演算子

演算子は、個々のデータ項目を操作し、結果を戻すために使用されます。これらのデータ項目をオペランドまたは引数と呼びます。演算子は特殊な文字またはキーワードで表します。たとえば、乗算演算子はアスタリスク (\*) で表し、NULL を検査する演算子はキーワード IS NULL で表します。以降の表に SQL 演算子を記載します。

## 単項演算子と 2 項演算子

演算子には一般的に次の 2 つのクラスがあります。

単項                    単項演算子はただ 1 つのオペランドについて操作します。単項演算子は次の書式で表されます。

```
operator operand
```

2 項                    2 項演算子は 2 つのオペランドについて操作します。2 項演算子は次の書式で表されます。

```
operand1 operator operand2
```

このほか、特別な書式を持ち、3 つ以上のオペランドを認める演算子もあります。演算子のオペランドに NULL が指定された場合、結果は常に NULL となります。この規則に従わない唯一の演算子が連結演算子 (||) です。

## 優先順位

優先順位とは、同じ式の中の異なる演算子を Oracle が評価する順序のことです。複数の演算子を含む式を評価するとき、Oracle は優先順位の高い演算子を評価した後で優先順位の低い演算子を評価します。優先順位の等しい演算子は、式の中で左から右に評価されます。

表 3-1 に、SQL 演算子を優先順位のレベルの高い方から順に示します。同じ行にリストされている演算子には同じ優先順位が付けられています。

表 3-1    SQL 演算子の優先順位

演算子	演算
+, -	同一、否定
*, /	乗算、除算
+, -,	加算、減算、連結
=, !=, <, >, <=, >=, IS NULL, LIKE, BETWEEN, IN	比較
NOT	指数、論理否定
AND	論理積
OR	論理和

例 次の式では、乗算は加算よりも優先順位が高いため、2 と 3 を掛けた結果に 1 が加算されます。

```
1+2*3
```

式の中でカッコを使用して演算子の優先順位を置き換えることができます。Oracle はカッコの内側の式を評価した後で外側の式を評価します。

SQL では集合演算子 (UNION および UNION ALL、INTERSECT、MINUS) もサポートされます。集合演算子により結合されるのは、問合せによって戻される行のセットであり、個々のデータ項目ではありません。集合演算子の優先順位はすべて同じです。

## 算術演算子

算術演算子を式の中で使用することにより、数値を否定（正負を反転）および加算、減算、乗算、除算できます。演算の結果も数値になります。これらの演算子の中には日付算術で使われるものもあります。表 3-2 は算術演算子のリストです。

表 3-2 算術演算子

演算子	用途	例
+-	式の正負を示す。これらは単項演算子である。	<pre>SELECT * FROM orders WHERE qtysold = -1; SELECT * FROM emp WHERE -sal &lt; 0;</pre>
*/	乗算、除算する。これらは2項演算子である。	<pre>UPDATE emp SET sal = sal * 1.1;</pre>
+-	加算、減算する。これらは2項演算子である。	<pre>SELECT sal + comm FROM emp WHERE SYSDATE - hiredate &gt; 365;</pre>

二重否定や負の数の減算を表現する場合に、算術式の中で、セパレータのない連続した負の符号 (--) は使用しないでください。文字 -- は、SQL 文ではコメントの開始を示すのに使用されるからです。連続した負の符号は空白やカッコで区切ってください。SQL 文の中のコメントについては、「コメント」(2-35 ページ) を参照してください。

## 連結演算子

連結演算子は文字列を操作するのに使用します。表 3-3 に連結演算子の説明を示します。

表 3-3 連結演算子

演算子	用途	例
	文字列を連結する。	<pre>SELECT 'Name is '    ename FROM emp;</pre>

2 つの文字列を連結した結果は別の文字列になります。両方の文字列が CHAR データ型である場合、結果は CHAR データ型の文字列となり、その最大文字数は 2000 です。どちらかの

文字列が `VARCHAR2` データ型である場合、結果は `VARCHAR2` データ型の文字列となり、その最大文字数は 4000 です。文字列のデータ型にかかわらず、文字列に後続する空白は連結後も残ります。`CHAR` データ型と `VARCHAR2` データ型の違いについては、「文字データ型」(2-7 ページ) を参照してください。

多くのプラットフォームで、連結演算子には表 3-3 に示すように 2 本の実線垂直バーで表されます。ただし、IBM 社のプラットフォームの中には、この演算子として破線垂直バーを使用するものもあります。異なるキャラクタ・セットを持つシステム間（たとえば ASCII と EBCDIC 間）で SQL スクリプト・ファイルを移動する場合、垂直バーがターゲットの Oracle 環境で必要な垂直バーに変換されない場合があります。オペレーティング・システムやネットワーク・ユーティリティによる変換を制御するのが困難であったり不可能であったりする場合に備えて、Oracle では垂直バー演算子にかわるものとして `CONCAT` 文字関数が提供されています。異なるキャラクタ・セットを持つ環境にアプリケーションを移動する場合は、アプリケーションでこの文字関数を使用することをお勧めします。

Oracle は長さがゼロの文字列を `NULL` として処理しますが、長さがゼロの文字列を別のオペランドと連結すると、その結果は常にもう一方のオペランドになります。結果が `NULL` になるのは、2 つの `NULL` 文字列を連結したときだけです。ただし、この処理は Oracle の将来のバージョンでも継続されるとは限りません。`NULL` となる可能性がある式を連結する場合には、`NVL` 関数を使用して明示的に式を長さがゼロの文字列に変換してください。

**例** 次の例では、`CHAR` 列と `VARCHAR2` 列を持つ表を作成し、後続する空白のある値とない値を挿入してから、これらの値を連結して選択しています。なお、`CHAR` 列と `VARCHAR2` 列では、ともに後続する空白が保存されます。

```
CREATE TABLE tab1 (col1 VARCHAR2(6), col2 CHAR(6),
                    col3 VARCHAR2(6), col4 CHAR(6) );
```

Table created.

```
INSERT INTO tab1 (col1, col2, col3, col4)
VALUES ('abc', 'def ', 'ghi ', 'jkl');
```

1 row created.

```
SELECT col1||col2||col3||col4 "Concatenation"
FROM tab1;
```

```
Concatenation
-----
abcdef ghi jkl
```



## 比較演算子

比較演算子は、2つの式を比較します。比較の結果は、真(TRUE)または偽(FALSE)、不定(UNKNOWN)になります。条件については、「条件」(3-84 ページ)を参照してください。表 3-4 は比較演算子のリストです。

表 3-4 比較演算子

演算子	用途	例
=	等価性の検査。	<pre>SELECT *   FROM emp  WHERE sal = 1500;</pre>
!=	不等性の検査。プラットフォームによっては、一部の不等号演算子の書式を使用できない場合もある。	<pre>SELECT *   FROM emp  WHERE sal != 1500;</pre>
>	大 / 小の検査。	<pre>SELECT * FROM emp  WHERE sal &gt; 1500;</pre>
<		<pre>SELECT * FROM emp  WHERE sal &lt; 1500;</pre>
>=	以上 / 以下の検査。	<pre>SELECT * FROM emp  WHERE sal &gt;= 1500;</pre>
<=		<pre>SELECT * FROM emp  WHERE sal &lt;= 1500;</pre>
IN	メンバーとの等価性の検査。"= ANY" と等価。	<pre>SELECT * FROM emp  WHERE job IN    ('CLERK', 'ANALYST'); SELECT * FROM emp  WHERE sal IN    (SELECT sal FROM emp     WHERE deptno = 30);</pre>
NOT IN	"!=ALL" と等価。メンバーのいずれかが NULL である場合には、FALSE と評価される。	<pre>SELECT * FROM emp  WHERE sal NOT IN    (SELECT sal FROM emp     WHERE deptno = 30); SELECT * FROM emp  WHERE job NOT IN    ('CLERK', 'ANALYST');</pre>

表 3-4 比較演算子

演算子	用途	例
ANY SOME	リスト内の各値または問合せによって戻される各値と値を比較する。=、!=、>、<、<=、>= のいずれかを先に指定しなければならない。  問合せにより行が戻されない場合には FALSE と評価される。	<pre>SELECT * FROM emp WHERE sal = ANY (SELECT sal FROM emp WHERE deptno = 30);</pre>
ALL	リスト内のすべての値または問合せによって戻されるすべての値と値を比較する。=、!=、>、<、<=、>= のいずれかを先に指定しなければならない。  問合せにより行が戻されない場合には TRUE と評価される。	<pre>SELECT * FROM emp WHERE sal &gt;= ALL ( 1400, 3000);</pre>
[NOT] BETWEEN x AND y	x 以上 y 以下の範囲にある [ ない ] ことを検査する。	<pre>SELECT * FROM emp WHERE sal BETWEEN 2000 AND 3000;</pre>
EXISTS	副問合せにより行が 1 行以上戻される場合に TRUE と評価される。	<pre>SELECT ename, deptno FROM dept WHERE EXISTS (SELECT * FROM emp WHERE dept.deptno = emp.deptno);</pre>
x [NOT] LIKE y	x がパターン y と一致する [ しない ] 場合に TRUE と評価される。y 中の "%" 文字は NULL を除く 0 文字以上の任意の文字列に一致する。"_" 文字は任意の 1 文字に一致する。	「LIKE 演算子」(3-7 ページ) を参照。  <pre>SELECT * FROM tabl WHERE coll LIKE 'A_C/%E%' ESCAPE '/';</pre>
[ESCAPE 'z']	パーセント (%) とアンダースコア ( ) を除く任意の文字を ESCAPE の後に指定できる。ワイルドカード文字は、エスケープ文字に指定された文字が前についている場合はリテラルとして扱われる。	
IS [NOT] NULL	NULL を検査する。NULL を検査するために使用すべき唯一の演算子。「NULL」(2-28 ページ) を参照。	<pre>SELECT ename, deptno FROM emp WHERE comm IS NULL;</pre>

NOT IN 演算子および LIKE 演算子については、次の項で詳しく説明します。

## NOT IN 演算子

NOT IN 演算子に続くリストの中のいずれかの項目が NULL である場合、すべての行は (UNKNOWN) 不定と評価されます（行はまったく戻されない）。たとえば、次の文ではそれぞれの行に対して文字列 'TRUE' が戻されます。

```
SELECT 'TRUE'
FROM emp
WHERE deptno NOT IN (5,15);
```

ただし、次の文では行は戻りません。

```
SELECT 'TRUE'
FROM emp
WHERE deptno NOT IN (5,15,null);
```

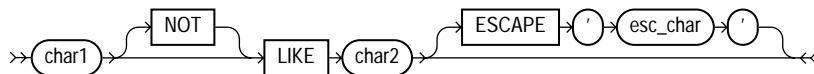
この例で行が戻らないのは、WHERE 句の条件が次のように評価されるからです。

```
deptno != 5 AND deptno != 15 AND deptno != null
```

NULL を比較する条件はすべて NULL になるため、式全体の結果は NULL になります。特に NOT IN 演算子が副問合せを参照するときに、このような作用を見逃してしまう可能性があることに注意してください。

## LIKE 演算子

LIKE 演算子は、パターン・マッチングによる文字列の比較で使用されます。次の図は LIKE 演算子を使用する条件の構文を示しています。



ここで、それぞれの意味は次のとおりです。

**char1** パターンと比較される値です。この値は CHAR データ型または VARCHAR2 データ型を持ちます。

**NOT** 条件の結果を論理的に反転させます。つまり、条件が TRUE に評価された場合には FALSE を返し、FALSE に評価された場合には TRUE を返します。

**char2** char1 の比較対象となるパターンです。パターンは CHAR データ型または VARCHAR2 データ型の値であり、特殊なパターン・マッチング文字 % と \_ を含むことができます。

**ESCAPE** エスケープ文字として単一の文字を識別します。エスケープ文字を使用して、パターン中で % や \_ を特殊文字としてではなくリテラルとして解釈させることができます。

エスケープ文字を含む文字列を検索する場合、そのエスケープ文字を 2 回指定する必要があります。たとえば、エスケープ文字が '/' で、文字列 'client/server' を検索する場合、'client//server' と指定してください。

等号 (=) は、ある文字値全体を別の文字値と適合しますが、**LIKE** 演算子は、ある文字値の一部を別の文字値と適合します（ある値が指定したパターンの検索を、もう一方の値に対して行う）。**LIKE** 比較では空白埋めが使用されないので注意してください。

**LIKE** 演算子では、値を定数ではなくパターンと比較できます。必ず **LIKE** キーワードの直後に、パターンを指定してください。たとえば、次の問合せを発行することにより、名前が 'SM' で始まる従業員すべての給与を検索できます。

```
SELECT sal
  FROM emp
 WHERE ename LIKE 'SM%';
```

次の問合せは **LIKE** 演算子のかわりに等号を使用しているため、名前が 'SM%' の従業員すべての給与を検索することになってしまいます。

```
SELECT sal
  FROM emp
 WHERE ename = 'SM%';
```

次の問合せでは名前が 'SM%' の従業員すべての給与が検索されます。この場合、'SM%' が **LIKE** 演算子の前にあるため、Oracle は 'SM%' をパターンとしてではなく、テキスト・リテラルとして解釈します。

```
SELECT sal
  FROM emp
 WHERE 'SM%' LIKE ename;
```

パターンでは、値の中の異なる文字の代用となる特殊文字がよく使用されます。

- パターン中のアンダースコア ( ) は、値の中の 1 文字（マルチバイトのキャラクタ・セットでの 1 バイトとは異なる）の代用となります。
- パターン中のパーセント記号 (%) は、値の中のゼロを含む任意の数の文字（マルチバイトのキャラクタ・セットでの 1 バイトとは異なる）の代用となります。しかし、パターン '%' は NULL と一致しないので注意してください。

**大文字/小文字の区別とパターン・マッチング** **LIKE** 演算子と等号 (=) 演算子を含む文字式を比較するすべての条件において、大文字と小文字は区別されます。次の例のように、**UPPER()** 関数を使用すると、大文字と小文字を区別しないでマッチングできます。

```
UPPER(ename) LIKE 'SM%'
```

**索引付きの列でのパターン・マッチング** LIKE を使用して索引付きの列をパターン検索する場合、パターンの先頭文字が "%" または "\_" でなければ、Oracle では索引を利用して文のパフォーマンスを向上できます。この場合、Oracle はこの先頭文字によって索引を走査できます。パターンの先頭文字が "%" または "\_" であると、Oracle は索引を走査できないので、問合せのパフォーマンスは向上しません。

**例 1** 次の条件は "MA" で始まるすべての ENAME 値について真となります。

```
ename LIKE 'MA%'
```

次の ENAME 値のすべてが条件を満たすと TRUE(真) となります。

```
MARTIN, MA, MARK, MARY
```

大文字と小文字は区別されるので、"Ma" および "ma"、"mA" で始まる ENAME 値では条件が FALSE(偽) になります。

**例 2** 次の条件を考えます。

```
ename LIKE 'SMITH_'
```

この条件は以下の ENAME 値について真となります。

```
SMITHE, SMITHY, SMITHS
```

特殊文字「\_」は ENAME 値の 1 つの文字の代用であるため、この条件は 'SMITH' について偽となります。

**ESCAPE オプション** ESCAPE オプションを使用すると、パターン中に "%" や "\_" を実際の文字として含めることができます。ESCAPE オプションはエスケープ文字を識別します。エスケープ文字がパターンの中で文字 "%" や "\_" の前に指定されている場合、Oracle はこの文字を特殊なパターン・マッチング文字としてではなく実際の文字として解釈します。

**例：** 次の文は、名前の中に文字列 'A\_B' を持つ従業員を検索します。

```
SELECT ename
FROM emp
WHERE ename LIKE '%A¥B%' ESCAPE '¥';
```

ESCAPE オプションはエスケープ文字として円記号 (¥) を識別します。パターンの中でエスケープ文字はアンダースコア (\_) に先行します。これによって、Oracle はアンダースコアを特殊なパターン・マッチング文字としてではなくリテラルとして解釈します。

**% なしのパターン** パターンに文字 "%" が含まれていない場合、両方のオペランドの長さが同じ場合にだけ、条件が TRUE(真) になることが可能です。

例：この表の定義と挿入される値について考えてみます。

```
CREATE TABLE freds (f CHAR(6), v VARCHAR2(6));
INSERT INTO freds VALUES ('FRED', 'FRED');
```

Oracle は CHAR 値に空白埋めを行うので、F の値は空白埋めによって 6 バイトになります。  
V の値は空白埋めされず、4 文字長のままです。

論理演算子

論理演算子は、2 つのコンポーネントの条件の結果を結合して、両者に基づいた単一の結果を生成するため、または単一の条件の結果を反転させるために使用します。表 3-5 は論理演算子のリストです。

表 3-5 論理演算子

演算子	機能	例
NOT	後続する条件が FALSE の場合に TRUE を返す。TRUE の場合には FALSE を返す。UNKNOWN の場合には UNKNOWN を返す。	SELECT * FROM emp WHERE NOT (job IS NULL);  SELECT * FROM emp WHERE NOT (sal BETWEEN 1000 AND 2000);
AND	コンポーネントの条件が両方とも TRUE である場合に TRUE を返す。どちらかが FALSE の場合には FALSE を返す。それ以外の場合は UNKNOWN を返す。	SELECT * FROM emp WHERE job = 'CLERK' AND deptno = 10;
OR	コンポーネントの条件のどちらかが TRUE である場合に TRUE を返す。両方とも FALSE の場合には FALSE を返す。それ以外の場合は UNKNOWN を返す。	SELECT * FROM emp WHERE job = 'CLERK' OR deptno = 10;

たとえば、次の SELECT 文の WHERE 句は、AND 論理演算子を使用して、1984 年より前に入社し、さらに月給が 1,000 ドルを超える従業員だけが選択されることを指定しています。

```
SELECT *  
FROM emp  
WHERE hiredate < TO_DATE('01-JAN-1984', 'DD-MON-YYYY')  
AND sal > 1000;
```

### NOT 演算子

表 3-6 は、条件に NOT 演算子を適用した結果を示しています。

表 3-6 NOT の真理値表

NOT	TRUE	FALSE	UNKNOWN
	FALSE	TRUE	UNKNOWN

### AND 演算子

表 3-7 は、AND 演算子で 2 つの式を結合した結果を示しています。

表 3-7 AND の真理値表

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

### OR 演算子

表 3-8 は、OR 演算子で 2 つの式を結合した結果を示しています。

表 3-8 OR の真理値表

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

# 集合演算子

集合演算子は 2 つのコンポーネントの問合せ結果を 1 つの結果にまとめます。集合演算子を含む問合せを複問合せと呼びます。表 3-9 は、SQL の集合演算子を示しています。

表 3-9 集合演算子

演算子	戻る結果
UNION	各問合せによって戻るすべての行（重複行は含まない）
UNION ALL	各問合せによって戻るすべての行（重複行も含む）
INTERSECT	両方の問合せによって戻るすべての行（重複行は含まない）
MINUS	最初の問合せによって戻る行で、2 番目の問合せでは戻されない行（重複行は含まない）

集合演算子の優先順位はすべて同じです。SQL 文に複数の集合演算子がある場合、カッコによって明示的に別の順序が指定されていない限り、Oracle は左から右の順に評価します。今後の SQL の規格に準拠するために、Oracle の将来のバージョンでは、INTERSECT 演算子には他の集合演算子よりも高い優先順位を与える予定です。そのため、INTERSECT 演算子以外の集合演算子とともに使用する問合せでは、カッコを使用して評価の順序を明示的に指定することをお勧めします。

複問合せを構成する各問合せと、それに対応する選択リスト内の各式は、数値とデータ型が一致している必要があります。集合演算子によって結合された 2 つの問合せが文字データを選択する場合、戻される値のデータ型は次のようにして決定されます。

- 両方の問合せが CHAR データ型の値を選択する場合、戻される値のデータ型は CHAR となる。
- 問合せのどちらか一方または両方が、VARCHAR2 データ型の値を選択する場合、戻される値のデータ型は VARCHAR2 となる。

例 次の 2 つの問合せとその結果を想定します。

```
SELECT part
  FROM orders_list1;

PART
-----
SPARKPLUG
FUEL PUMP
FUEL PUMP
TAILPIPE
```



```
SELECT part
      FROM orders_list2;
```

```
PART
-----
CRANKSHAFT
TAILPIPE
TAILPIPE
```

次に、各集合演算子でこの2つの問合せの結果を結合する例を示します。

**UNION の例** 次の文は、UNION 演算子によって2つの結果を結合しています。結果に重複行は含まれません。次の文は、他方の表に存在していない列がある場合に、データ型を一致させる方法を示しています。

```
SELECT part, partnum, to_date(null) date_in
      FROM orders_list1
UNION
SELECT part, to_null(null), date_in
      FROM orders_list2;
```

PART	PARTNUM	DATE_IN
-----	-----	-----
SPARKPLUG	3323165	
SPARKPLUG		10/24/98
FUEL PUMP	3323162	
FUEL PUMP		12/24/99
TAILPIPE	1332999	
TAILPIPE		01/01/01
CRANKSHAFT	9394991	
CRANKSHAFT		09/12/02

```
SELECT part
      FROM orders_list1
UNION
SELECT part
      FROM orders_list2;
```

```
PART
-----
SPARKPLUG
FUEL PUMP
TAILPIPE
CRANKSHAFT
```

**UNION ALL の例** 次の文は、UNION ALL 演算子を使用して2つの結果を結合しています。結果には重複行が含まれることもあります。

```
SELECT part
      FROM orders_list1
UNION ALL
SELECT part
      FROM orders_list2;
```

```
PART
-----
SPARKPLUG
FUEL PUMP
FUEL PUMP
TAILPIPE
CRANKSHAFT
TAILPIPE
TAILPIPE
```

UNION ALL 演算子がすべての行を返すのに対して、UNION 演算子は重複しない行だけを返すことに注目してください。問合せで複数回戻される PART 値（「FUEL PUMP」など）は、UNION 演算子では一度だけ戻されますが、UNION ALL 演算子では複数回戻されています。

**INTERSECT の例** 次の文は、INTERSECT 演算子によって2つの結果を結合しています。この場合、両方の問合せによって共通に戻される行だけが戻されます。

```
SELECT part
      FROM orders_list1
INTERSECT
SELECT part
      FROM orders_list2;
```

```
PART
-----
TAILPIPE
```

**MINUS の例** 次の文は、MINUS 演算子を使用して2つの結果を結合します。この場合、最初の問合せでは戻されるが2番目の問合せでは戻されない行だけが戻されます。

```
SELECT part
      FROM orders_list1
MINUS
SELECT part
      FROM orders_list2;
```

```
PART
-----
SPARKPLUG
FUEL PUMP
```

## その他の演算子

表 3-10 に、その他の SQL 演算子を示します。

表 3-10 その他の SQL 演算子

演算子	用途	例
(+)	(+) の直前の列が結合で外部結合列であることを示す。「外部結合」(4-504 ページ) を参照。	<pre>SELECT ename, dname FROM emp, dept WHERE dept.deptno = emp.deptno(+);</pre>
PRIOR	階層型（ツリー構造）の問合せで、PRIOR の次の式を現行の行の親行に対して評価する。このような問合せでは、行の親子関連を定義するために CONNECT BY 句でこの演算子を使用しなければならない。また、階層問合せを実行する SELECT 文の他の部分でこの演算子を使用できる。PRIOR 演算子は単項演算子であり、単項算術演算子の + や - と同じ優先順位を持っている。「階層問合せ」(4-492 ページ) を参照。	<pre>SELECT empno, ename, mgr FROM emp CONNECT BY PRIOR empno = mgr;</pre>

## SQL 関数

SQL 関数は、データ項目を操作し結果を戻すという点で演算子と似ています。SQL 関数と演算子は引数を指定する書式が異なります。次の書式によって、関数ではゼロ以上の引数を操作できます。

```
function(argument, argument, ...)
```

SQL 関数が必要とするデータ型以外のデータ型の引数で SQL 関数を呼び出すと、Oracle は SQL 関数を実行する前に、その引数を必要なデータ型に暗黙的に変換します。「データ変換」(2-26 ページ) を参照してください。

NULL を引数として SQL 関数を呼び出すと、SQL 関数により自動的に NULL が戻されます。この規則に従わない SQL 関数は、CONCAT および DECODE、DUMP、NVL、REPLACE の 5 つだけです。

SQL 関数と、PL/SQL で書かれたユーザー関数を混同しないでください。ユーザー関数については、「ユーザー関数」(3-57 ページ) を参照してください。

SQL 関数の構文図では、このマニュアルの「はじめに」にある「構文図と表記法」で説明した規則に従って、データ型とともに引数が示されています。

一般に、SQL 関数には次のタイプがあります。

- 単一行（またはスカラー）関数
- グループ（または集合体）関数

これら 2 つの SQL 関数は、操作の対象とする行数が異なります。単一行関数は、問合せ対象の表やビューの各行に対して 1 つの結果行を戻します。グループ関数は、問合せ対象の行グループに対して 1 つの結果行を戻します。

単一行関数は、(GROUP BY 句を含まない SELECT 文の) 選択リスト、WHERE 句、START WITH 句、CONNECT BY 句に指定できます。

グループ関数は、選択リストと HAVING 句に指定できます。SELECT 文で GROUPBY 句が使用されている場合、Oracle は問合せ対象の表やビューの行をグループに分けます。GROUP BY 句を含む問合せでは、選択リストのすべての要素は、GROUP BY 句からの式またはグループ関数を含む式、定数のいずれかでなければなりません。Oracle は、選択リスト内のグループ関数を行の各グループに適用し、各グループに単一の結果行を戻します。

GROUP BY 句を指定しないと、Oracle は選択リスト内のグループ関数を問合せ対象の表またはビューのすべての行に適用します。HAVING 句でグループ関数を指定して、グループを出力しないこともできます。このとき、出力は、問合せ対象の表またはビューの各行の値ではなく、グループ関数の結果に基づきます。GROUP BY 句と HAVING 句の詳細は、「GROUP BY 句」(4-496 ページ) および「HAVING 句」(4-497 ページ) を参照してください。

以降の項では、関数は引数のデータ型と戻り値によってグループ化されています。

数値関数

数値関数は入力として数値を受け取り、結果として数値を戻します。この項では、SQL の数値関数を説明します。これらの関数の大部分は 38 桁 (10 進) の値を戻します。超越関数の COS、COSH、EXP、LN、LOG、SIN、SINH、SQRT、TAN、TANH は 36 桁 (10 進) の値を戻します。超越関数の ACOS、ASIN、ATAN、ATAN2 は 30 桁 (10 進) の値を戻します。

ABS

構文	ABS( <i>n</i> )
用途	<i>n</i> の絶対値を戻します。
例	<pre>SELECT ABS(-15) "Absolute" FROM DUAL;</pre> <div><div>Absolute</div><div>-----</div><div>15</div></div>

## ACOS

構文	ACOS( <i>n</i> )
用途	<i>n</i> の逆コサインを戻します。入力 は -1 ～ 1 の範囲で、出力は 0 ～ $\pi$ (ラジアン) の範囲です。
例	<pre>SELECT ACOS(.3) "Arc_Cosine" FROM DUAL;  Arc_Cosine ----- 1.26610367</pre>

## ASIN

構文	ASIN( <i>n</i> )
用途	<i>n</i> の逆サインを戻します。入力 は -1 ～ 1 の範囲で、出力は $-\pi/2 \sim \pi/2$ (ラジアン) の範囲です。
例	<pre>SELECT ASIN(.3) "Arc_Sine" FROM DUAL;  Arc_Sine ----- .304692654</pre>

## ATAN

構文	ATAN( <i>n</i> )
用途	<i>n</i> の逆タンジェントを戻します。入力の範囲に制限はなく、出力は $-\pi/2 \sim \pi/2$ (ラジアン) の範囲です。
例	<pre>SELECT ATAN(.3) "Arc_Tangent" FROM DUAL;  Arc_Tangent ----- .291456794</pre>

ATAN2

構文	ATAN2( <i>n</i> , <i>m</i> )
用途	<i>n</i> と <i>m</i> の逆タンジェントを戻します。入力の範囲に制限はなく、出力は <i>n</i> と <i>m</i> の符号により - <i>p</i> ～ <i>p</i> ( ラジアン ) の範囲です。ATAN2( <i>n</i> , <i>m</i> ) は ATAN2( <i>n</i> / <i>m</i> ) と同じです。
例	<pre>SELECT ATAN2(.3, .2) "Arc_Tangent2" FROM DUAL;  Arc_Tangent2 ----- .982793723</pre>

CEIL

構文	CEIL( <i>n</i> )
用途	<i>n</i> 以上の最も小さい整数を戻します。
例	<pre>SELECT CEIL(15.7) "Ceiling" FROM DUAL;        Ceiling -----           16</pre>

COS

構文	COS( <i>n</i> )
用途	<i>n</i> ( ラジアンで表された角度 ) のコサインを戻します。
例	<pre>SELECT COS(180 * 3.14159265359/180) "Cosine of 180 degrees" FROM DUAL;  Cosine of 180 degrees ----- -1</pre>

## COSH

構文	COSH( <i>n</i> )
用途	<i>n</i> の双曲線コサインを戻します。
例	<pre>SELECT COSH(0) "Hyperbolic cosine of 0" FROM DUAL;  Hyperbolic cosine of 0 ----- 1</pre>

## EXP

構文	EXP( <i>n</i> )
用途	<i>e</i> を <i>n</i> 乗した値を戻します ( <i>e</i> = 2.71828183 ... )。
例	<pre>SELECT EXP(4) "e to the 4th power" FROM DUAL;  e to the 4th power ----- 54.59815</pre>

## FLOOR

構文	FLOOR( <i>n</i> )
用途	<i>n</i> 以下の最も大きい整数を戻します。
例	<pre>SELECT FLOOR(15.7) "Floor" FROM DUAL;  Floor ----- 15</pre>

## LN

構文	LN( <i>n</i> )
用途	<i>n</i> の自然対数を戻します ( <i>n</i> は正の数)。
例	<pre>SELECT LN(95) "Natural log of 95" FROM DUAL;  Natural log of 95 ----- 4.55387689</pre>

LOG

構文

LOG(m,n)

用途

$m$  を底とする  $n$  の対数を戻します。底  $m$  は 0 および 1 以外の任意の正の数、 $n$  は任意の正の数です。

例

```
SELECT LOG(10,100) "Log base 10 of 100" FROM DUAL;
```

```
Log base 10 of 100
-----
                      2
```

MOD

構文

MOD(m,n)

用途

$m$  を  $n$  で除した余りを戻します。 $n$  が 0 の場合は、 $m$  を戻します。

例

```
SELECT MOD(11,4) "Modulus" FROM DUAL;
```

```
Modulus
-----
        3
```

この関数は、 $m$  が負の場合には古典数学のモジュール関数とは異なる動作をします。古典数学のモジュール関数は、次の公式を用いた MOD 関数で表すことができます。

$$m - n * \text{FLOOR}(m/n)$$

次の文は、MOD 関数と古典数学のモジュール関数の相違を示します。

```
SELECT m, n, MOD(m, n),
m - n * FLOOR(m/n) "Classical Modulus"
FROM test_mod_table;
```

M	N	MOD (M,N)	Classical Modulus
11	4	3	3
11	-4	3	-1
-11	4	-3	1
-11	-4	-3	-3



## POWER

構文	POWER( <i>m</i> , <i>n</i> )
用途	<i>m</i> を <i>n</i> 乗した値を返します。底 <i>m</i> と指数 <i>n</i> は任意の数です。ただし、 <i>m</i> が負の場合、 <i>n</i> は整数でなければなりません。
例	<pre>SELECT POWER(3,2) "Raised" FROM DUAL;</pre> <div>Raised ----- 9</div>

## ROUND

構文	ROUND( <i>n</i> [, <i>m</i> ])
用途	<i>n</i> を小数点以下 <i>m</i> 桁に丸めた値を返します。 <i>m</i> が省略されると小数点以下が丸められます。 <i>m</i> が負の場合は小数点の左 <i>m</i> 桁が丸められます。 <i>m</i> は整数でなければなりません。
例 1	<pre>SELECT ROUND(15.193,1) "Round" FROM DUAL;</pre> <div>Round ----- 15.2</div>
例 2	<pre>SELECT ROUND(15.193,-1) "Round" FROM DUAL;</pre> <div>Round ----- 20</div>

## SIGN

構文	SIGN( <i>n</i> )
用途	<i>n</i> が 0 より小さい場合は -1、 <i>n</i> が 0 の場合は 0、 <i>n</i> が 0 より大きい場合は 1 を返します。
例	<pre>SELECT SIGN(-15) "Sign" FROM DUAL;</pre> <div>Sign ----- -1</div>

構文	SIN(n)
用途	n(ラジアンで表された角度)のサインを戻します。
例	<pre>SELECT SIN(30 * 3.14159265359/180)        "Sine of 30 degrees" FROM DUAL;</pre> <p>Sine of 30 degrees</p> <p>-----</p> <p>.5</p>

SINH

構文	<code>SINH(n)</code>
用途	$n$ の双曲線サインを戻します。
例	<pre>SELECT SINH(1) "Hyperbolic sine of 1" FROM DUAL;  Hyperbolic sine of 1 ----- 1.17520119</pre>

SQRT

構文	SQRT( <i>n</i> )
用途	<i>n</i> の平方根を戻します。 <i>n</i> には負の値は指定できません。SQRT は "実数" を戻します。
例	<pre>SELECT SQRT(26) "Square root" FROM DUAL;</pre> <pre>Square root</pre> <pre>-----</pre> <pre>5.09901951</pre>

## TAN

構文	TAN
用途	$n$ (ラジアンで表された角度) のタンジェントを戻します。
例	<pre>SELECT TAN(135 * 3.14159265359/180)       "Tangent of 135 degrees" FROM DUAL;  Tangent of 135 degrees -----                 - 1</pre>

## TANH

構文	TANH( $n$ )
用途	$n$ の双曲線タンジェントを戻します。
例	<pre>SELECT TANH(.5) "Hyperbolic tangent of .5"       FROM DUAL;  Hyperbolic tangent of .5 -----                 .462117157</pre>

## TRUNC

構文	TRUNC( $n$ [, $m$ ])
用途	$n$ を小数点以下 $m$ 桁に切り捨てた値を戻します。 $m$ が省略されたときは、小数点以下が切り捨てられます。 $m$ が負の場合は、小数点の左 $m$ 桁がゼロになります。
例	<pre>SELECT TRUNC(15.79,1) "Truncate" FROM DUAL;  Truncate -----       15.7  SELECT TRUNC(15.79,-1) "Truncate" FROM DUAL;  Truncate -----       10</pre>

## 文字関数

単一行文字関数は、入力として文字を受け取り、文字または数値のどちらかを返します。

### 文字値を返す文字関数

この項では文字値を返す文字関数を説明します。特に断りのない限り、これらの関数はすべて VARCHAR2 データ型を持つ値を返し、長さは 4000 バイトに制限されます。データ型が CHAR の値を返す関数は、長さは 2000 バイトに制限されます。戻り値の長さが制限を超えた場合、Oracle は戻り値から制限を超えた部分を切り捨てて、エラー・メッセージを出力せずにその結果を返します。

#### CHR

構文	CHR( <i>n</i> [USING NCHAR_CS])
用途	<p>データベース・キャラクタ・セットまたは各国キャラクタ・セットの中の <i>n</i> に等しい 2 進数を持つ文字を返します。</p> <p>USING NCHAR_CS 句が指定されていない場合は、この関数はデータベース・キャラクタ・セットの中の <i>n</i> に等しい 2 進数を持つ文字を VARCHAR2 値として返します。</p> <p>USING NCHAR_CS 句が指定されている場合は、この関数は各国キャラクタ・セットの中の <i>n</i> に等しい 2 進数を持つ文字を NVARCHAR2 値として返します。</p>
例 1	<pre>SELECT CHR(67)    CHR(65)    CHR(84) "Dog"        FROM DUAL; Dog ---</pre>
例 2	<pre>SELECT CHR(16705 USING NCHAR_CS) FROM DUAL;  C - A</pre>

#### CONCAT

構文	CONCAT( <i>char1</i> , <i>char2</i> )
用途	<p><i>char1</i> を <i>char2</i> と連結して返します。この関数は連結演算子 (  ) と等価です。この演算子の詳細は、「連結演算子」(3.3 ページ) を参照してください。</p>

例                    次の例は、ネストを使用して3つの文字列を連結しています。

```
SELECT CONCAT( CONCAT(ename, ' is a '), job) "Job"
FROM emp
WHERE empno = 7900;
```

Job  
-----  
JAMES is a CLERK

## INITCAP

構文                    `INITCAP(char)`

用途                    単語の最初の文字を大文字、残りの文字を小文字にして *char* を戻します。  
単語は空白または英数字以外の文字で区切ります。

例                    `SELECT INITCAP('the soap') "Capitals" FROM DUAL;`

Capitals  
-----  
The Soap

## LOWER

構文                    `LOWER(char)`

用途                    すべての文字を小文字にして *char* を戻します。戻り値は引数 *char* と同じ  
データ型 (CHAR または VARCHAR2) になります。

例                    `SELECT LOWER('MR. SCOTT MCMILLAN') "Lowercase"
FROM DUAL;`

Lowercase  
-----  
mr. scott mcmillan

LPAD

構文	LPAD(char1,n [,char2])
用途	<p><i>char1</i> の左に <i>char2</i> で指定した文字を連続的に埋め込んで <i>n</i> 桁にして戻します。<i>char2</i> のデフォルト値は単一の空白です。<i>char1</i> が <i>n</i> 文字より長い場合、この関数は <i>n</i> に収まる <i>char1</i> の一部を戻します。</p> <p>引数 <i>n</i> は端末画面に表示される場合の戻り値の全体の長さです。多くのキャラクタ・セットでは、これは戻り値の文字数でもあります。ただし、マルチバイトのキャラクタ・セットでは、文字列の表示長が文字列の文字数と異なる場合もあります。</p>
例	<pre>SELECT LPAD('Page 1',15,'*.*') "LPAD example" FROM DUAL;  LPAD example ----- *.*.*.*.*Page 1</pre>

LTRIM

構文	LTRIM(char [,set])
用途	<p><i>char</i> の左側にあって <i>set</i> に指定された文字を削除します。<i>set</i> のデフォルト値は単一の空白です。Oracle は <i>char</i> の先頭文字から走査し始め、<i>set</i> に指定された文字をすべて削除します。<i>set</i> に指定された以外の文字が見つかった時点で結果を戻します。</p>
例	<pre>SELECT LTRIM('xyzXxyLAST WORD','xy') "LTRIM example" FROM DUAL;  LTRIM exampl ----- XxyLAST WORD</pre>

## NLS\_INITCAP

構文	NLS_INITCAP(char [, 'nlsparams'] )
用途	<p>単語の最初の文字を大文字、残りの文字を小文字にして <i>char</i> を戻します。単語は空白または英数字以外の文字で区切ります。'nlsparams' の値は次の書式で指定します。</p> <p>'NLS_SORT = sort'</p> <p>sort は言語ソート順序または <b>BINARY</b> のどちらかです。言語のソート順序は、大文字と小文字の変換のために特別な言語要件を処理します。なお、これらの要件によって、<i>char</i> と異なる長さの値が戻される可能性があります。'nlsparams' を指定しないと、セッションのデフォルト・ソート順序が使用されます。ソート順序の説明は、『Oracle8 Server リファレンス』を参照してください。</p>
例	<pre>SELECT NLS_INITCAP       ('ijsland', 'NLS_SORT = XDutch') "Capitalized" FROM DUAL;</pre> <p>Capital ----- IJsland</p>

## NLS\_LOWER

構文	NLS_LOWER(char [, 'nlsparams'] )
用途	<p>すべての文字を小文字にして <i>char</i> を戻します。'nlsparams' の書式と用途は NLS_INITCAP 関数と同じです。</p>
例	<pre>SELECT NLS_LOWER       ('CITTA', 'NLS_SORT = XGerman') "Lowercase" FROM DUAL;</pre> <p>Lower ----- citta</p>

## NLS\_UPPER

構文	NLS_UPPER(char [, 'nlsparams'] )
用途	<p>すべての文字を大文字にして <i>char</i> を戻します。'nlsparams' の書式と用途は NLS_INITCAP 関数と同じです。</p>

例

```
SELECT NLS_UPPER
      ('große', 'NLS_SORT = XGerman') "Uppercase"
FROM DUAL;
```

Upper  
-----  
GROSS

REPLACE

構文

```
REPLACE(char,search_string[,replacement_string])
```

用途

*replacement\_string* で *search\_string* をすべて置換して *char* を戻します。  
*replacement\_string* を指定しない場合または NULL の場合、*search\_string* が  
すべて削除されます。*search\_string* が NULL の場合、*char* が戻されます。  
この関数は TRANSLATE 関数を拡張したものです。TRANSLATE 関数は、  
単一の文字を 1 対 1 で置き換えます。REPLACE 関数では、文字列の置換  
または削除を実行できます。

例

```
SELECT REPLACE('JACK and JUE','J','BL') "Changes"
FROM DUAL;
```

Changes  
-----  
BLACK and BLUE

RPAD

構文

```
RPAD(char1, n [,char2])
```

用途

*char1* の右に *char2* で指定した文字を連続的に埋め込んで *n* 桁にして戻し  
ます。*char2* のデフォルト値は単一の空白です。*char1* が *n* 文字より長い場  
合、この関数は *n* に収まる *char1* の一部を戻します。

引数 *n* は端末画面に表示される場合の戻り値の全体の長さです。多くの  
キャラクタ・セットでは、これは戻り値の文字数でもあります。ただし、  
マルチバイトのキャラクタ・セットでは、文字列の表示長が文字列の文字  
数と異なる場合もあります。

例

```
SELECT RPAD('MORRISON',12,'ab') "RPAD example"
FROM DUAL;
```

RPAD example  
-----  
MORRISONabab



## RTRIM

構文	<code>RTRIM(char [,set])</code>
用途	<i>char</i> の右側にあつて <i>set</i> に指定されたすべての文字を削除し、 <i>char</i> を戻します。 <i>set</i> のデフォルト値は単一の空白です。RTRIM は LTRIM と似た働きをします。
例	<pre>SELECT RTRIM('BROWNINGxXxy','xy') "RTRIM e.g."       FROM DUAL;</pre> <pre>RTRIM e.g ----- BROWNINGxX</pre>

## SOUNDEX

構文	<code>SOUNDEX(char)</code>
用途	<p><i>char</i> と同じ発音を持つ文字列を戻します。この関数によって、綴りは異なるが発音の似た単語（英語）を比較できます。</p> <p>音声表現については、『The Art of Computer Programming, Volume3: Sorting and Searching(Donald E.Knuth 著)』で規定されています。</p> <ul style="list-style-type: none"> <li>文字列の最初の文字を残し、以降 a、e、h、i、o、u、w、y が出てきた場合にはすべて削除します。</li> <li>残った文字の 2 文字目以降に対し、次のように数字を割り当てます。 <pre>b, f, p, v = 1 c, g, j, k, q, s, x, z = 2 d, t = 3 l = 4 m, n = 5 r = 6</pre> </li> <li>割り当てられた数字が同じ文字が 2 つ以上並んでいる場合は、最初の文字だけを残して削除します。</li> <li>残りを 0 で埋めて、最初の 4 バイトだけを戻します。</li> </ul>
例	<pre>SELECT ename       FROM emp      WHERE SOUNDEX(ename)            = SOUNDEX('SMYTHE');</pre> <pre>ENAME ----- SMITH</pre>

SUBSTR

構文	SUBSTR(char, m [,n])
用途	<p>char の m 番目の文字から n 文字分の文字列を抜き出して戻します。m が 0 である場合、1 として扱われます。m が正の数である場合、Oracle は char の始めから数えて最初の文字を見つけます。m が負の数である場合、Oracle は char の終わりから数えます。n を指定しないと、Oracle は char の終わりまでのすべての文字を戻します。n が 1 より小さい場合、NULL を戻します。</p> <p>引数として substr に渡された浮動小数点数は、自動的に整数に変換されます。</p>
例 1	<pre>SELECT SUBSTR('ABCDEFGF',3.1,4) "Subs" FROM DUAL;</pre> <p>Subs ---- CDEF</p>
例 2	<pre>SELECT SUBSTR('ABCDEFGF',-5,4) "Subs" FROM DUAL;</pre> <p>Subs ---- CDEF</p>

SUBSTRB

構文	SUBSTR(char, m [,n])
用途	<p>引数の m と n が文字ではなくバイトで表される以外、SUBSTR と同じです。データベースのキャラクタ・セットがシングルバイトの場合、SUBSTRB は SUBSTR と等価です。</p> <p>引数として substrb に渡された浮動小数点数は、自動的に整数に変換されます。</p>
例	<p>データベースのキャラクタ・セットがダブルバイトの場合を想定します。</p> <pre>SELECT SUBSTRB('ABCDEFGF',5,4.2)       "Substring with bytes" FROM DUAL;</pre> <p>Substring with bytes ----- CD</p>

## TRANSLATE

構文	<code>TRANSLATE(char, from, to)</code>
用途	<p><i>from</i> 内のすべての文字を <i>to</i> 内の対応する文字で置換して <i>char</i> を戻します。<i>from</i> 内に存在しない <i>char</i> 内の文字は置換されません。引数 <i>from</i> には <i>to</i> より多くの文字が含まれていてもかまいません。この場合、<i>from</i> の終わりのほうの余分な文字は <i>to</i> 内に対応する文字を持ちません。これらの余分な文字が <i>char</i> 内にあると、それらの文字は戻り値から取り除かれます。戻り値から <i>from</i> 内の文字をすべて削除するために <i>to</i> に空の文字列は使用できません。Oracle は空の文字列を NULL と解釈し、NULL の引数がある場合、この関数は NULL を戻します。</p>
例 1	<p>次の文は、ライセンス番号を置換します。文字 'ABC...Z' はすべて 'X' に置換され、数 '012...9' はすべて '9' に置換されます。</p> <pre>SELECT TRANSLATE('2KRW229', '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ', '9999999999XXXXXXXXXXXXXXXXXXXXXXXXXXXXX') "License" FROM DUAL;</pre> <p>License ----- 9XXX999</p>
例 2	<p>次の文は、文字を取り除いて数値だけになったライセンス番号を戻します。</p> <pre>SELECT TRANSLATE('2KRW229', '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ', '0123456789') "Translate example" FROM DUAL;</pre> <p>Translate example ----- 2229</p>

## UPPER

構文	<code>UPPER(char)</code>
用途	すべての文字を大文字にして <i>char</i> を戻します。戻り値は引数 <i>char</i> と同じデータ型を持ちます。

例	<pre>SELECT UPPER('Large') "Uppercase"       FROM DUAL;  Upper ----- LARGE</pre>
---	--

数値を戻す文字関数

この項では数値を戻す文字関数を説明します。

ASCII

構文	<code>ASCII(char)</code>
用途	<i>char</i> の最初の文字の、データベースのキャラクタ・セットでの 10 進表現を戻します。データベースのキャラクタ・セットが 7 ビットの ASCII である場合には ASCII 値を戻します。データベースのキャラクタ・セットが EBCDIC コード・ページ 500 である場合には EBCDIC 値を戻します。なお、この関数と同じような EBCDIC 文字関数は存在しません。
例	<pre>SELECT ASCII('Q')       FROM DUAL;  ASCII('Q') -----       81</pre>

INSTR

構文	<code>INSTR (char1,char2 [,n[,m]])</code>
用途	<i>char1</i> の <i>n</i> 番目の文字から <i>char2</i> の検索を開始し、 <i>char2</i> が <i>m</i> 番目に出現した位置を戻します ( <i>char2</i> が文字列であればその最初の文字の位置を戻します)。 <i>n</i> が負である場合、Oracle は <i>char1</i> の終わりに逆方向に検索します。 <i>m</i> の値は正でなければなりません。 <i>n</i> と <i>m</i> のデフォルト値は 1 です。この場合、Oracle は <i>char1</i> の最初の文字から検索を開始します。検索対象は <i>char2</i> の最初の出現です。戻り値は <i>n</i> の値にかかわらず、 <i>char1</i> の先頭から数えた文字位置です。検索が失敗した ( <i>char1</i> の <i>n</i> 番目の文字の後に <i>char2</i> が <i>m</i> 回出現しない) 場合、戻り値は 0 となります。

例 1            `SELECT INSTR('CORPORATE FLOOR','OR', 3, 2)`  
                  `"Instring" FROM DUAL;`

```
Instring
-----
      14
```

例 2            `SELECT INSTR('CORPORATE FLOOR','OR', -3, 2)`  
                  `"Reversed Instring"`  
                  `FROM DUAL;`

```
Reversed Instring
-----
              2
```

## INSTRB

構文            `INSTRB(char1,char2[,n[,m]])`

用途            `n` と戻り値が文字ではなくバイトで表されること以外は `INSTR` と同じです。データベースのキャラクタ・セットがシングルバイトの場合、`INSTRB` は `INSTR` と等価です。

例              この例では、データベースのキャラクタ・セットがダブルバイトの場合を想定しています。

```
SELECT INSTRB('CORPORATE FLOOR','OR',5,2)
" Instring in bytes"
FROM DUAL;
```

```
Instring in bytes
-----
              27
```

## LENGTH

構文            `LENGTH(char)`

用途            `char` の長さを文字単位で戻します。`char` のデータ型が `CHAR` の場合には、その長さには後続する空白がすべて含まれます。`char` が `NULL` である場合には、`NULL` が戻されます。

例              `SELECT LENGTH('CANDIDE') "Length in characters"`  
                  `FROM DUAL;`

```
Length in characters
-----
              7
```

LENGTHB

構文	LENGTHB(char)
用途	<i>char</i> の長さをバイト単位で戻します。 <i>char</i> が NULL である場合には、NULL が戻されます。データベースのキャラクタ・セットがシングルバイトの場合、LENGTHB は LENGTH と等価です。
例	<p>この例では、データベースのキャラクタ・セットがダブルバイトの場合を想定しています。</p> <pre>SELECT LENGTHB ('CANDIDE') "Length in bytes"       FROM DUAL;</pre> <pre>Length in bytes -----                 14</pre>

NLSSORT

構文	NLSSORT(char [, 'nlsparams'])
用途	<p><i>char</i> をソートするために使用される文字列のバイトを戻します。'nlsparams' の値は次の書式で指定します。</p> <pre>'NLS_SORT = sort'</pre> <p><i>sort</i> は言語ソート順序または BINARY のどちらかです。'nlsparams' を指定しないと、セッションのデフォルト・ソート順序が使用されます。BINARY を指定すると、この関数は <i>char</i> を戻します。ソート順序については、『Oracle8 Server リファレンス』の「各国語サポート」の章を参照してください。</p>
例	<p>この関数を使って、文字列の 2 進値を基にした比較ではなく、言語ソート順序を基にした比較を指定できます。</p> <pre>SELECT ename FROM emp       WHERE NLSSORT (ename, 'NLS_SORT = German')       &gt; NLSSORT ('S', 'NLS_SORT = German') ORDER BY ename;</pre> <pre>ENAME ----- SCOTT SMITH TURNER WARD</pre>

日付関数

日付関数は DATE データ型の値を操作します。数に戻す MONTHS\_BETWEEN 関数を除いた日付関数はすべて DATE データ型の値を戻します。

## ADD\_MONTHS

## 構文

```
ADD_MONTHS(d,n)
```

## 用途

日付  $d$  に  $n$  か月を加えて戻します。引数  $n$  には整数を指定します。 $d$  が月の最終日の場合、または結果の月の最終日が  $d$  の日付コンポーネントよりも小さい場合、戻される値は結果の月の最終日となります。それ以外の場合、結果は  $d$  と同じ日付コンポーネントを持ちます。

## 例

```
SELECT TO_CHAR(
    ADD_MONTHS(hiredate,1),
    'DD-MON-YYYY') "Next month"
FROM emp
WHERE ename = 'SMITH';
```

```
Next Month
-----
17-JAN-1981
```

## LAST\_DAY

## 構文

```
LAST_DAY(d)
```

## 用途

$d$  を含む月の最後の日付を戻します。その月の残りの日数を求めるためにこの関数を使用できます。

## 例 1

```
SELECT SYSDATE,
    LAST_DAY(SYSDATE) "Last",
    LAST_DAY(SYSDATE) - SYSDATE "Days Left"
FROM DUAL;
```

```
SYSDATE      Last          Days Left
-----
23-OCT-97  31-OCT-97          8
```

## 例 2

```
SELECT TO_CHAR(
    ADD_MONTHS(
        LAST_DAY(hiredate),5),
    'DD-MON-YYYY') "Five months"
FROM emp
WHERE ename = 'MARTIN';
```

```
Five months
-----
28-FEB-1982
```

MONTHS\_BETWEEN

構文	MONTHS_BETWEEN(d1, d2)
用途	日付 <i>d1</i> と <i>d2</i> の間の月数を返します。 <i>d1</i> が <i>d2</i> 以降の日付であれば結果は正の値になります。 <i>d1</i> が <i>d2</i> 以前の日付であれば結果は負の値になります。 <i>d1</i> と <i>d2</i> が月の同じ日または月の最終日の場合、結果は整数になります。それ以外の場合、Oracle は結果の小数部を 1 か月、31 日として計算します。また、 <i>d1</i> と <i>d2</i> の時間要素の相違も考慮されます。
例	<pre>SELECT MONTHS_BETWEEN       ( TO_DATE( '02-02-1995', 'MM-DD-YYYY' ),         TO_DATE( '01-01-1995', 'MM-DD-YYYY' ) ) "Months" FROM   DUAL;</pre> <pre>           Months ----- 1.03225806</pre>

NEW\_TIME

構文	NEW_TIME(d, z1, z2)
用途	時間帯 <i>z1</i> の日時が <i>d</i> である時点の時間帯 <i>z2</i> の日時を返します。引数 <i>z1</i> と <i>z2</i> には次のテキスト文字列の 1 つを指定します。  AST      大西洋地区の標準時間と夏時間  ADT  BST      ベーリング地区の標準時間と夏時間  BDT  CST      中央地区の標準時間と夏時間  CDT  EST      東地区の標準時間と夏時間  EDT  GMT      グリニッジ標準時間  HST      アラスカ、ハワイ地区の標準時間と夏時間  HDT  MST      山岳地区の標準時間と夏時間  MDT  NST      ニューファンドランド地区の標準時間



PST      太平洋地区の標準時間と夏時間  
PDT  
YST      ユーコン地区の標準時間と夏時間  
YDT

## NEXT\_DAY

**構文**                `NEXT_DAY(d, char)`

**用途**                *char* で指定した曜日で、日付 *d* 以降の最初の日付を戻します。引数 *char* は、セッションで使用している言語での曜日でなければなりません（フルネームでも省略形でも構いません）。必要最小限の文字数は、省略形の文字数です。有効な省略形の後に続けて文字が入力されていても、それらの文字は無視されます。戻り値は、引数 *d* と同じ時間、分、秒のコンポーネントを持っています。

**例**                    次の例では、1992 年 3 月 15 日以降の最初の火曜日の日付を戻します。

```
SELECT NEXT_DAY('15-MAR-92', 'TUESDAY') "NEXT DAY"
      FROM DUAL;
```

```

NEXT DAY
-----
17-MAR-92
```

## ROUND

**構文**                `ROUND(d[,fmt])`

**用途**                *d* を書式モデル *fmt* で指定した単位まで近似した結果を戻します。*fmt* を指定しないと、*d* は最も近い日が戻ります。*fmt* で使用できる書式モデルについては、「ROUND と TRUNC」（3-38 ページ）を参照してください。

**例**                    `SELECT ROUND (TO_DATE ('27-OCT-92'), 'YEAR')  
"New Year" FROM DUAL;`

```

New Year
-----
01-JAN-93
```

## SYSDATE

**構文**                `SYSDATE`

**用途**                現在の日時を戻します。引数は必要としません。分散 SQL 文では、ローカル・データベース上の日時が戻ります。CHECK 制約の条件でこの関数は使用できません。

例

```
SELECT TO_CHAR
      (SYSDATE, 'MM-DD-YYYY HH24:MI:SS') "NOW"
FROM DUAL;

NOW
-----
10-29-1993 20:27:11
```

TRUNC

構文

1TRUNC(*d*, [*fmt*])

用途

時刻部分を書式モデル *fmt* で指定された単位まで近似した *d* を戻します。 *fmt* を指定しないと、*d* は最も近い日が戻ります。 *fmt* で使用できる書式モデルについては、「ROUND と TRUNC」(3-38 ページ) を参照してください。

例

```
SELECT TRUNC(TO_DATE('27-OCT-92', 'DD-MON-YY'), 'YEAR')
      "New Year" FROM DUAL;

New Year
-----
01-JAN-92
```

ROUND と TRUNC

表 3-11 に、日付関数 **ROUND** および **TRUNC** で使用できる書式モデル、および日付の丸めと切捨ての単位を示します。デフォルト・モデル 'DD' では、午前 0 時（真夜中）を基準に近似を行い、日付を戻します。

表 3-11 日付関数 ROUND と TRUNC の日付書式モデル

書式モデル	丸めと切捨ての単位
CC SCC	4 桁の年の上 2 桁より 1 大きい数
SYYYYY YYYY YEAR SYEAR YYY YY Y	年 (7 月 1 日に切り上げ)
IYYY IY IY I	ISO 年
Q	四半期 (その四半期の 2 番目の月の 16 日に切上げ)

表 3-11 日付関数 ROUND と TRUNC の日付書式モデル

書式モデル	丸めと切捨ての単位
MONTH	月 (16 日に切上げ)
MON	
MM	
RM	
WW	年の最初の日と同じ曜日
IW	ISO 年の最初の日と同じ曜日
W	月の最初の日と同じ曜日
DDD	日
DD	
J	
DAY	週の開始日
DY	
D	
HH	時
HH12	
HH24	
MI	分

書式モデル DAY および DY、D によって使用される週の開始日は、初期化パラメータ NLS\_TERRITORY によって暗黙的に指定されています。このパラメータについては、『Oracle8 Server リファレンス』を参照してください。

変換関数

変換関数は、あるデータ型から他のデータ型に値を変換します。一般的に関数名はデータ型 TO データ型の形で与えられます。最初 (TO の前) のデータ型が入力データ型、後のデータ型が出力データ型です。この項では、SQL の変換関数を説明します。

CHARTOROWID

構文	CHARTOROWID ( char )
用途	CHAR データ型または VARCHAR2 データ型の値を ROWID データ型の値に変換します。

```
例
SELECT ename FROM emp
      WHERE ROWID = CHARTOROWID('AAAAfZAABAAACp8AAO');

ENAME
-----
LEWIS
```

CONVERT

```
構文
CONVERT(char, dest_char_set [,source_char_set] )
```

用途
あるキャラクタ・セットの文字列を別のキャラクタ・セットの文字列に変換します。

引数 char は変換する値です。  
引数 dest\_char\_set は char が変換されるキャラクタ・セットの名前です。  
引数 source\_char\_set は char をデータベースに格納しているキャラクタ・セットの名前です。デフォルト値はデータベースのキャラクタ・セットです。

変換先キャラクタ・セットと変換元キャラクタ・セットの引数として、リテラルまたはキャラクタ・セットの名前を含んでいる列を指定できます。  
完全に文字を変換するには、変換先キャラクタ・セットが変換元キャラクタ・セットで定義されているすべての文字を表現できなければなりません。文字が変換先キャラクタ・セットに存在しないと置換文字が使用されます。置換文字はキャラクタ・セット定義の一部として定義できます。

```
例
SELECT CONVERT('Gro&#223;', 'US7ASCII', 'WE8HP')
      "Conversion"
      FROM DUAL;
```

```
Conversion
-----
Gross
```

一般的なキャラクタ・セットを次に示します。

US7ASCII	US7 ビット ASCII キャラクタ・セット
WE8DEC	DEC 西ヨーロッパ 8 ビット・キャラクタ・セット
WE8HP	HP 西ヨーロッパ Laserjet 8 ビット・キャラクタ・セット
F7DEC	DEC フランス 7 ビット・キャラクタ・セット
WE8EBCDIC500	IBM 西ヨーロッパ EBCDIC コード・ページ 500
WE8PC850	IBM PC コード・ページ 850
WE8ISO8859P1	ISO 8859-1 西ヨーロッパ 8 ビット・キャラクタ・セット

## HEXTORAW

構文	HEXTORAW(char)
用途	16 進数を含む <i>char</i> を RAW 値に変換します。
例	<pre>INSERT INTO graphics (raw_column)   SELECT HEXTORAW('7D') FROM DUAL;</pre>

## RAWTOHEX

構文	RAWTOHEX(raw)
用途	<i>raw</i> を 16 進で表した文字の値に変換します。
例	<pre>SELECT RAWTOHEX(raw_column) "Graphics"   FROM graphics;</pre> <pre>Graphics ----- 7D</pre>

## ROWIDTOCHAR

構文	ROWIDTOCHAR(rowid)
用途	ROWID 値を VARCHAR2 データ型に変換します。この変換の結果は常に 18 文字です。
例	<pre>SELECT ROWID   FROM offices  WHERE   ROWIDTOCHAR(ROWID) LIKE 'Br1AAB%';</pre> <pre>ROWID ----- AAAAZ6AABAAABr1AAB</pre>

## TO\_CHAR( 日付変換 )

構文	TO_CHAR(d [, fmt [, 'nlsparams' ] ])
用途	DATE データ型の値 <i>d</i> を日付書式 <i>fmt</i> で指定した書式の VARCHAR2 データ型の値に変換します。 <i>fmt</i> を指定しないと、 <i>d</i> がデフォルトの日付書式の VARCHAR2 値に変換されます。日付書式については、「書式モデル」(3-60 ページ)を参照してください。

'nlsparams' には、月と日の名前や略称が戻される言語を指定します。この引数は次の書式で指定します。

```
'NLS_DATE_LANGUAGE = language'
```

nlsparams を指定しないと、この関数はセッションのデフォルト日付言語を使用します。

例

```
SELECT TO_CHAR(HIREDATE, 'Month DD, YYYY')
       "New date format" FROM emp
WHERE ename = 'BLAKE';
```

```
New date format
-----
May           01, 1981
```

## TO\_CHAR( 数値変換 )

構文

```
TO_CHAR(n [, fmt [, 'nlsparams' ] ])
```

用途

オプションの数値書式 *fmt* を使用して、NUMBER データ型の値 *n* を VARCHAR2 データ型の値に変換します。*fmt* を指定しないと、*n* の有効桁数を保持するために十分な長さの VARCHAR2 値に変換されます。数値書式については、「書式モデル」(3-60 ページ)を参照してください。

'nlsparams' は数の書式要素によって戻される次の文字を指定します。

- 小数点文字
- グループ・セパレータ
- 各国通貨記号
- 国際通貨記号

この引数は次の書式で指定します。

```
'NLS_NUMERIC_CHARACTERS = "dg"
NLS_CURRENCY = "text"
NLS_ISO_CURRENCY = territory '
```

文字 *d* と *g* はそれぞれ小数点文字とグループ・セパレータを表します。それらは異なるシングルスバイト文字でなければなりません。引用符で囲んだ文字列の中では、パラメータ値を囲む引用符のために単一引用符を2つ使用しなければならないことに注意してください。通貨記号には10文字使用できます。

'nlsparams' やパラメータのどれか1つを指定しないと、この関数はセッションのデフォルト・パラメータ値を使用します。

例 1                   この例では、出力で通貨記号の左に空白埋めが行われます。

```
SELECT TO_CHAR(-10000, 'L99G999D99MI') "Amount"
      FROM DUAL;
```

```
Amount
-----
$10,000.00-
```

例 2                   

```
SELECT TO_CHAR(-10000, 'L99G999D99MI',
      'NLS_NUMERIC_CHARACTERS = ",.'"
      NLS_CURRENCY = ''AusDollars'' ') "Amount"
      FROM DUAL;
```

```
Amount
-----
AusDollars10.000,00-
```

注意：

- オプションの数値書式 *fmt* では、**L** は各国通貨記号を、**MI** は後に付くマイナス記号 (-) を表します。数値書式要素の全リストは、表 3-13 (3-63 ページ) を参照してください。
- Oracle で数値を文字列に変換していて、Oracle の **NUMBER** データ型の範囲を超過または不足するような丸め処理が発生した場合、無限大を表す "∞" またはマイナス無限大を表す "-∞" が戻されることがあります。通常、このイベントは、**TO\_CHAR()** 関数を制限的な数値書式文字列で使用して、丸め処理が行われた場合に発生します。

## TO\_DATE

構文                   **TO\_DATE**(*char* [, *fmt* [, '*nlsparams*'] ])

用途                   **CHAR** データ型または **VARCHAR2** データ型の値 *char* を **DATE** データ型の値に変換します。*fmt* は *char* の書式を指定する日付書式です。*fmt* を指定しない場合、*char* はデフォルト日付書式でなければなりません。*fmt* が 'J' (ユリウス暦) である場合、*char* は整数でなければなりません。日付書式については、「書式モデル」(3-60 ページ) を参照してください。

'*nlsparams*' は日付変換の **TO\_CHAR** 関数の場合と同じ用途で使用されません。

引数 *char* として **DATE** 値を持つ **TO\_DATE** 関数を使用してはなりません。戻される **DATE** 値は、*fmt* またはデフォルト値に依存して、元の *char* とは異なる世紀の値を持つことがあります。

日付書式については、「日付書式モデル」(3-65 ページ) を参照してください。

```
例      INSERT INTO bonus (bonus_date)
        SELECT TO_DATE(
            'January 15, 1989, 11:00 A.M.',
            'Month dd, YYYY, HH:MI A.M.',
            'NLS_DATE_LANGUAGE = American')
        FROM DUAL;
```

TO\_MULTI\_BYTE

```
構文      TO_MULTI_BYTE(char)

用途      シングルバイト文字のすべてを、対応するマルチバイト文字に変換して
          char を戻します。char 内に等価なマルチバイト文字がないシングルバイト
          文字は、シングルバイト文字として出力されます。この関数は、使用して
          いるデータベースのキャラクタ・セットに、シングルバイト文字とマルチ
          バイト文字の両方が含まれている場合にだけ有効です。
```

TO\_NUMBER

```
構文      TO_NUMBER(char [,fmt [, 'nlsparams'] ])

用途      オプションの書式モデル fmt によって指定した書式の数を含む CHAR デー
          タ型または VARCHAR2 データ型の値 char を NUMBER データ型の値に変
          換します。

例 1      UPDATE emp SET sal = sal +
          TO_NUMBER('100.00', '9G999D99')
          WHERE ename = 'BLAKE';

          この関数内の nlsparams 文字列は、数値変換用の TO_CHAR 関数内で同じ
          用途に使用されます。

例 2      SELECT TO_NUMBER('-AusDollars100','L9G999D99',
          ' NLS_NUMERIC_CHARACTERS = ',.',
          NLS_CURRENCY = "          AusDollars"
          ') "Amount"
          FROM DUAL;

          Amount
          -----
          -100
```

TO\_SINGLE\_BYTE

```
構文      TO_SINGLE_BYTE(char)
```



**用途** マルチバイト文字のすべてを、対応するシングルバイト文字に変換して *char* を戻します。*char* 内に等価なシングルバイト文字がないマルチバイト文字は、マルチバイト文字として出力されます。この関数は、使用しているデータベースのキャラクタ・セットに、シングルバイト文字とマルチバイト文字の両方が含まれている場合にだけ有効です。

## TRANSLATE USING

**構文** `TRANSLATE(text USING {CHAR_CS | NCHAR_CS })`

**用途** *text* を、データベース・キャラクタ・セットと各国キャラクタ・セットの変換のために指定されたキャラクタ・セットに変換します。

引数 *text* は変換する式です。

USING CHAR\_CS 引数を指定すると、*text* がデータベース・キャラクタ・セットに変換されます。出力データ型は VARCHAR2 です。

USING NCHAR\_CS 引数を指定すると、*text* が各国キャラクタ・セットに変換されます。出力データ型は NVARCHAR2 です。

この関数は、Oracle の CONVERT 関数と似ていますが、入力または出力のデータ型に NCHAR または NVARCHAR2 を使用する場合は、CONVERT ではなくこの関数を使う必要があります。

**例 1**

```
CREATE TABLE t1 (char_col CHAR(20),
                  nchar_col nchar(20));

INSERT INTO t1
VALUES ('Hi', N'Bye');

SELECT * FROM t1;
```

```
CHAR_COL      NCHAR_COL
-----
Hi            Bye
```

**例 2**

```
UPDATE t1 SET
  nchar_col = TRANSLATE(char_col USING NCHAR_CS);
UPDATE t1 SET
  char_col = TRANSLATE(nchar_col USING CHAR_CS);
SELECT * FROM t1;
```

```
CHAR_COL      NCHAR_COL
-----
Hi            Hi
```

例 3

```
UPDATE t1 SET
  nchar_col = TRANSLATE('deo' USING NCHAR_CS);
UPDATE t1 SET
  char_col = TRANSLATE(N'deo' USING CHAR_CS);

CHAR_COL      NCHAR_COL
-----
deo           deo
```

その他の単一行関数

DUMP

構文

```
DUMP(expr[,return_format[,start_position[,length]] ] )
```

用途

*expr* のデータ型コード、および長さ（単位はバイト）、内部表現を含む VARCHAR2 値を戻します。戻される結果は常にデータベース・キャラクタ・セットの文字です。各コードに対応するデータ型については、表 2-1（2-5 ページ）を参照してください。

引数 *return\_format* には戻り値の書式として、次の値のいずれかを指定します。

デフォルトでは、戻り値にキャラクタ・セット情報が含まれません。*expr* のキャラクタ・セット名を取り出すには、下記のいずれかの書式値に 1000 を加えて指定します。たとえば、*return\_format* に 1008 を指定すると、8 進表記で結果が戻り、さらに *expr* のキャラクタ・セット名が得られます。

8    結果は 8 進表記で戻されます。

10   結果は 10 進表記で戻されます。

16   結果は 16 進表記で戻されます。

17   結果は単一文字として戻されます。

引数 *start\_position* と *length* を組み合わせて、戻される内部表現の部分を指定します。何も指定しないと 10 進表記で全体の内部表現が戻されます。

*expr* が NULL の場合、この関数は 'NULL' を戻します。

例 1

```
SELECT DUMP('abc', 1016)
FROM DUAL;

DUMP('ABC' ,1016)
-----
Typ=96 Len=3 CharacterSet=WE8DEC: 61,62,63
```

```

例 2      SELECT DUMP(ename, 8, 3, 2) "OCTAL"
           FROM emp
           WHERE ename = 'SCOTT';

           OCTAL
           -----
           Type=1 Len=5: 117,124

例 3      SELECT DUMP(ename, 10, 3, 2) "ASCII"
           FROM emp
           WHERE ename = 'SCOTT';

           ASCII
           -----
           Type=1 Len=5: 79,84

```

## EMPTY\_[B | C]LOB

構文            `EMPTY_[B|C]LOB()`

用途            LOB 変数を初期化するため、あるいは INSERT または UPDATE 文で LOB 列または属性を EMPTY に初期化するために使用できる空の LOB ロケータを戻します。EMPTY とは、LOB は初期化されているが、データが挿入されていない状態を言います。

この関数から戻されるロケータは、DBMS\_LOB パッケージまたは OCI へのパラメータとして使用できません。

```

例      INSERT INTO lob_tab1 VALUES (EMPTY_BLOB());
      UPDATE lob_tab1
      SET clob_col = EMPTY_BLOB();

```

## BFILENAME

構文            `BFILENAME ('directory', 'filename')`

用途            サーバーのファイル・システムの物理 LOB ファイルに対応付けられている BFILE ロケータを戻します。'directory' には、ファイルが実際に格納されているサーバー・ファイル・システム上での完全パス名の別名を指定します。'filename' には、サーバー・ファイル・システムでのファイル名を指定します。

BFILENAME を指定する時点では、'directory' および 'filename' はファイル・システムに存在しているオブジェクトを指している必要はありません。ただし、後続の SQL、PL/SQL、DBMS\_LOB パッケージ、または OCI の操作を実行する前には、BFILE 値を物理ファイルに関連付けておく必要があります。詳細は、「CREATE DIRECTORY コマンド」(4-226 ページ)を参照してください。

注意: この関数は、指定されたディレクトリやファイルが実際に存在しているかどうかを確認はしません。したがって、`BFILENAME` の後で `CREATE DIRECTORY` コマンドを呼び出すこともできます。ただし、`BFILE` ロケータを実際使用时には、そのオブジェクトが存在している必要があります (たとえば、`OCILobFileOpen()` や `DBMS_LOB.FILEOPEN()` のような `OCILob` または `DBMS_LOB` の操作の 1 つに対するパラメータとして)。

LOB の詳細は、『Oracle8 Server アプリケーション開発者ガイド』および『Oracle8 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。

例

```
INSERT INTO file_tbl
VALUES (BFILENAME ('lob_dir1', 'image1.gif'));
```

## GREATEST

構文

```
GREATEST(expr [,expr] ...)
```

用途

リストされた *expr* 内の最大値を返します。2 番目以降のすべての *expr* は、比較の前に最初の *expr* のデータ型に暗黙的に変換されます。Oracle は非空白埋め比較方法で *expr* を比較します。文字の比較はデータベースのキャラクタ・セットの文字値に基づいて行われます。文字値の大きい文字が、別の文字より大きい文字と見なされます。この関数によって戻される値が文字データである場合、そのデータ型は常に `VARCHAR2` となります。

例

```
SELECT GREATEST ('HARRY', 'HARRIOT', 'HAROLD')
       "Great" FROM DUAL;
```

```
Great
-----
HARRY
```

## LEAST

構文

```
LEAST(expr [,expr] ...)
```

用途

リストされた *expr* 内の最少値を返します。比較の前に、2 番目以降のすべての *expr* は最初の *expr* のデータ型に暗黙的に変換されます。Oracle は非空白埋め比較方法で *expr* を比較します。この関数によって戻される値が文字データである場合、そのデータ型は常に `VARCHAR2` となります。

例

```
SELECT LEAST('HARRY','HARRIOT','HAROLD') "LEAST"
FROM DUAL;
```

```
LEAST
-----
HAROLD
```

## NLS\_CHARSET\_DECL\_LEN

構文                   NLS\_CHARSET\_DECL\_LEN(*bytecnt*, *csid*)

用途                   NCHAR 列の宣言の長さ（文字数）を戻します。引数 *bytecnt* は、列の長さを指定します。引数 *csid* は、列のキャラクタ・セット ID を指定します。

例

```
SELECT NLS_CHARSET_DECL_LEN
(200, nls_charset_id('ja16eucfixed'))
FROM DUAL;
```

```
NLS_CHARSET_DECL_LEN(200,NLS_CHARSET_ID(' JA16EUCFIXED'))
-----
100
```

## NLS\_CHARSET\_ID

構文                   NLS\_CHARSET\_ID(*text*)

用途                   NLS キャラクタ・セット名 *text* に対応する NLS キャラクタ・セットの ID 番号を戻します。引数 *text* には、実行時の VARCHAR2 値を指定します。*text* の値を 'CHAR\_CS' に指定すると、サーバーのデータベース・キャラクタ・セットの ID 番号が戻されます。*text* の値を 'NCHAR\_CS' に指定すると、サーバーの各国キャラクタ・セットの ID 番号が戻されます。

無効なキャラクタ・セット名を指定すると、NULL が戻されます。

キャラクタ・セット名のリストについては、『Oracle8 Server リファレンス』を参照してください。

例 1

```
SELECT NLS_CHARSET_ID('ja16euc ')
FROM DUAL;
```

```
NLS_CHARSET_ID('JA16EUC ')
-----
830
```

例 2

```
SELECT NLS_CHARSET_ID('char_cs ')
      FROM DUAL;

NLS_CHARSET_ID('CHAR_CS ')
-----x-----
2
```

例 3

```
SELECT NLS_CHARSET_ID('nchar_cs ')
      FROM DUAL;

NLS_CHARSET_ID('NCHAR_CS ')
-----
2
```

NLS\_CHARSET\_NAME

構文

NLS\_CHARSET\_NAME(*n*)

用途

ID 番号 *n* に対応する NLS キャラクタ・セット名を戻します。キャラクタ・セット名は、データベース・キャラクタ・セットの VARCHAR2 値として戻されます。

*n* が有効なキャラクタ・セット ID として認識されない場合は、この関数によって NULL が戻されます。

キャラクタ・セット ID のリストについては、『Oracle8 Server リファレンス』を参照してください。

例

```
SELECT NLS_CHARSET_NAME(2)
      FROM DUAL;

NLS_CH
-----
WE8DEC
```

NVL

構文

NVL(*expr1*, *expr2*)

用途

*expr1* が NULL の場合には *expr2* を戻します。*expr1* が NULL でない場合には *expr1* を戻します。引数 *expr1* と *expr2* は任意のデータ型を持つことができます。2 つのデータ型が異なる場合、Oracle は *expr2* を *expr1* のデータ型に変換した後で比較を行います。戻り値のデータ型は、常に *expr1* のデータ型と同じになります。ただし、*expr1* が文字データの場合は、戻り値のデータ型は VARCHAR2 となります。

例

```
SELECT ename, NVL(TO_CHAR(COMM), 'NOT
APPLICABLE')
      "COMMISSION" FROM emp
WHERE deptno = 30;
```

ENAME	COMMISSION
-----	-----
ALLEN	300
WARD	500
MARTIN	1400
BLAKE	NOT APPLICABLE
TURNER	0
JAMES	NOT APPLICABLE

UID

構文

UID

用途

現行のユーザーを一意に識別する整数を戻します。

USER

構文

USER

用途

現行の Oracle ユーザーの名前をデータ型 VARCHAR2 で戻します。Oracle は、空白埋め比較方法でこの関数の値を比較します。  
  
分散 SQL 文では、UID 関数と USER 関数は、ローカル・データベース上のユーザーを識別します。CHECK 制約の条件でこれらの関数は使用できません。

例

```
SELECT USER, UID FROM DUAL;
```

USER	UID
-----	-----
SCOTT	19

USERENV

構文

USERENV(option)

用途

現行のセッションに関する VARCHAR2 データ型の情報を戻します。この情報は、アプリケーション固有の監査証跡表を書き込む場合や現行セッションで使用されている言語固有の文字を判定する場合に有効です。CHECK 制約の条件で USERENV は使用できません。引数 option には次の値のいずれかを指定できません。

'ISDBA'	現在使用可能な ISDBA ロールを持っている場合は 'TRUE' を、持っていない場合は 'FALSE' を戻します。
'LANGUAGE'	現行セッションで使用している言語 (language) と地域 (territory) をデータベースのキャラクタ・セット (character set) も含めて次の形で戻します。  language_territory.characterset
'TERMINAL'	現行セッションの端末に対するオペレーティング・システム識別子を戻します。分散 SQL 文では、このオプションはローカル・セッションの識別子を戻します。分散環境では、リモートの SELECT に対してだけこのオプションを使用でき、リモートの INSERT または UPDATE、DELETE には使用できません。
'SESSIONID'	監査セッション識別子を戻します。このオプションは分散 SQL 文で使用できません。USERENV でこのキーワードを使用するには、初期化パラメータ AUDIT_TRAIL を TRUE に設定しなければなりません。
'ENTRYID'	使用可能な監査エントリ識別子を戻します。このオプションは分散 SQL 文で使用できません。USERENV でこのキーワードを使用するには、初期化パラメータ AUDIT_TRAIL を TRUE に設定しなければなりません。
'LANG'	言語名の ISO 略称を戻します。これは、既存の 'LANGUAGE' パラメータを短縮したものです。
'INSTANCE'	現行のインスタンスのインスタンス ID 番号を戻します。

例

```
SELECT USERENV('LANGUAGE') "Language" FROM DUAL;

Language
-----
AMERICAN_AMERICA.WE8DEC
```

VSIZE

構文

```
VSIZE(expr)
```

用途

*expr* の内部表現でのバイト数を戻します。*expr* が NULL である場合には NULL を戻します。



例

```
SELECT ename, VSIZE (ename) "BYTES"
FROM emp
WHERE deptno = 10;
```

ENAME	BYTES
-----	-----
CLARK	5
KING	4
MILLER	6

## オブジェクト参照関数

オブジェクト参照関数は、REF( 指定されたオブジェクト型のオブジェクトへの参照 ) を操作します。REF の詳細は、『Oracle8 概要』および『Oracle8 Server アプリケーション開発者ガイド』を参照してください。

### DEREF

構文                   DEREF(*e*)

用途                   引数 *e* のオブジェクト参照を戻します。引数 *e* は、オブジェクトに対する REF を戻す式でなければなりません。

例                     CREATE TABLE tb1(c1 NUMBER, c2 REF t1);  
SELECT Deref(c2) FROM tb1;

### REFTOHEX

構文                   REFTOHEX(*r*)

用途                   引数 *r* を 16 進で表した文字の値に変換します。

例                     CREATE TABLE tb1(c1 NUMBER, c2 REF t1);  
SELECT REFTOHEX(c2) FROM tb1;

### MAKE\_REF

構文                   MAKE\_REF(*table*, *key* [,*key*...])

用途                   *key* を主キーとして使用して、オブジェクト・ビューの行に対する REF を作成します。オブジェクト・ビューの詳細は、『Oracle8 Server アプリケーション開発者ガイド』を参照してください。

```
例          CREATE TYPE t1 AS OBJECT(a NUMBER, b NUMBER);

          CREATE TABLE tb1
              (c1 NUMBER, c2 NUMBER, PRIMARY KEY(c1, c2));

          CREATE VIEW v1 OF t1 WITH OBJECT OID(a, b) AS
              SELECT * FROM tb1;

          SELECT MAKE_REF(v1, 1, 3) FROM DUAL;
```

## グループ関数

グループ関数は、単一行に基づく結果を戻すのではなく、行のグループに基づく結果を戻します。この点で、グループ関数は単一行関数と異なります。グループ関数と単一行関数の違いについては、「SQL 関数」(3-15 ページ)を参照してください。

多くのグループ関数は次のオプションを取ります。

**DISTINCT**      このオプションを指定すると、グループ関数は異なる値を持つ引数式だけに作用します。

**ALL**            このオプションを指定すると、グループ関数は重複を含めてすべての値に作用します。

たとえば、1、1、1、3 の平均値は **DISTINCT** では 2 となり、**ALL** では 1.5 となります。どちらのオプションも指定しないと **ALL** が使用されます。

**COUNT(\*)** 以外のすべてのグループ関数で **NULL** は無視されます。グループ関数に対する引数で **NVL** を使用して、**NULL** を値で置き換えることができます。

グループ関数を持つ問合せが、グループ関数への引数として行を戻さなかったり、**NULL** を持つ行だけを戻したりすると、グループ関数は **NULL** を戻します。

### AVG

**構文**            `AVG([DISTINCT|ALL] n)`

**用途**            `n` の平均値を戻します。

```
例          SELECT AVG(sal) "Average"
              FROM emp;

              Average
              -----
              2077.21429
```

## COUNT

構文	COUNT( <i>{*   [DISTINCT ALL] expr}</i> )
用途	<p>問合せ内の行数を返します。</p> <p><i>expr</i> を指定すると、この関数は <i>expr</i> が NULL でない行数を返します。 <i>expr</i> の行をすべて数えることもできますし、異なる値だけを数えることもできます。</p> <p>アスタリスク (*) を指定すると、この関数は重複値や NULL 値も含めたすべての行数を返します。</p>
例 1	<pre>SELECT COUNT(*) "Total"       FROM emp;        Total -----         18</pre>
例 2	<pre>SELECT COUNT(job) "Count"       FROM emp;        Count -----         14</pre>
例 3	<pre>SELECT COUNT(DISTINCT job) "Jobs"       FROM emp;        Jobs -----          5</pre>

## MAX

構文	MAX( <i>[DISTINCT ALL] expr</i> )
用途	<i>expr</i> の最大値を返します。
例	<pre>SELECT MAX(sal) "Maximum" FROM emp;        Maximum -----         5000</pre>

MIN

構文	MIN([DISTINCT ALL] expr)
用途	expr の最小値を返します。
例	<pre>SELECT MIN(hiredate) "Earliest" FROM emp;  Earliest ----- 17-DEC-80</pre>

STDDEV

構文	STDDEV([DISTINCT ALL] x)
用途	x の標準偏差を返します。Oracle は、VARIANCE グループ関数に対して定義された分散の平方根として標準偏差を計算します。
例	<pre>SELECT STDDEV(sal) "Deviation"       FROM emp;  Deviation ----- 1182.50322</pre>

SUM

構文	SUM([DISTINCT ALL] n)
用途	n の合計値を返します。
例	<pre>SELECT SUM(sal) "Total"       FROM emp;  Total ----- 29081</pre>

## VARIANCE

構文	VARIANCE([DISTINCT ALL]x)
用途	x の分散を戻します。Oracle は次の公式を使用して、x の分散を計算します。

$$\frac{\sum_{i=1}^n x_i^2 - \frac{1}{n} \left[ \sum_{i=1}^n x_i \right]^2}{n-1}$$

ここで、それぞれの意味は次のとおりです。

$x_i$  は、x の要素の 1 つです。

n は、セット x の要素の数です。n が 1 の場合、分散は 0 となります。

例

```
SELECT VARIANCE(sal) "Variance"
FROM emp;
```

```
Variance
-----
1389313.87
```

## ユーザー関数

PL/SQL で独自のユーザー関数を作成し、SQL や SQL 関数にはない機能を持たせることができます。ユーザー関数は、SQL 関数が使用可能なすべての SQL 文で使えます。つまり、式を置くことができる場所であればどこでも使用できます。

たとえば、次の箇所でユーザー関数を使用できます。

- SELECT コマンドの選択リスト
- WHERE 句の条件
- CONNECT BY 句、および START WITH 句、ORDER BY 句、GROUP BY 句
- INSERT コマンドの VALUES 句
- UPDATE コマンドの SET 句

ユーザー関数の作成と使用方法の詳細は、『Oracle8 Server アプリケーション開発者ガイド』を参照してください。

### 前提条件

ユーザー関数を作成する場合、トップ・レベルの PL/SQL 関数として作成するか、これらの関数を SQL 文で使用する前にパッケージ仕様部で宣言する必要があります。「CREATE FUNCTION コマンド」(4-228 ページ)で説明されている CREATEFUNCTION 文を使用すると、ユーザー関数はトップ・レベルの PL/SQL 関数として作成されます。パッケージ関数は、「CREATE PACKAGE コマンド」(4-246 ページ)で説明されている CREATE PACKAGE 文を含むパッケージで指定します。

パッケージ・ユーザー関数を呼び出すには、パッケージ仕様部で RESTRICT\_REFERENCES プラグマを宣言する必要があります。

### 必要な権限

SQL の式の中でユーザー関数を使用するには、ユーザーがユーザー関数の EXECUTE 権限の所有者であるか、その権限を与えられている必要があります。ユーザー関数で定義したビューを問い合わせるには、そのビューに対する SELECT 権限がなければなりません。ビューから選択するには、個別の EXECUTE 権限は必要ありません。

### ユーザー関数での制約

不変の定義が必要な場合には、ユーザー関数を使用できません。したがって、ユーザー関数には次の制約があります。

- CREATE TABLE コマンドまたは ALTER TABLE コマンドの CHECK 制約句では使用できない。
- CREATE TABLE コマンドまたは ALTER TABLE コマンドの DEFAULT 句では使用できない。
- OUT または IN OUT パラメータを含めることはできない。
- データベースは更新できない。
- 関数がリモート関数の場合は、パッケージ状態の読取りまたは書込みはできない。
- 関数がパッケージ状態を変更する場合は、関数内の SQL コマンドで parallelism\_clause は使用できない。
- 関数内で定義された変数は更新できない。ただし、関数がローカル関数であり、しかも SELECT リストまたは INSERT コマンドの VALUES 句、UPDATE コマンドの SET 句で使用されている場合は例外です。

### 名前の優先順位

PL/SQL では、データベースの列名は、パラメータなしの関数名よりも優先順位が高くなります。たとえば、ユーザー SCOTT が自分のスキーマ内で次の 2 つのオブジェクトを作成するとします。

```
CREATE TABLE emp(new_sal NUMBER, ...);
```

```
CREATE FUNCTION new_sal RETURN NUMBER IS BEGIN ... END;
```

その後、次の2つの文のように NEW\_SAL を参照すると列 EMP.NEW\_SAL を参照することになります。

```
SELECT new_sal FROM emp;
SELECT emp.new_sal FROM emp;
```

関数 NEW\_SAL にアクセスするには、次のように入力します。

```
SELECT scott.new_sal FROM emp;
```

SQL の式内で使用可能なユーザー関数の呼出し例を次に示します。

```
circle_area (radius)
payroll.tax_rate (empno)
scott.payroll.tax_rate (dependent, empno)@ny
```

**例** たとえば、スキーマ SCOTT から TAX\_RATE ユーザー関数を呼び出し、TAX\_TABLE 内の SS\_NO 列と SAL 列に対してこの関数を実行し、その結果を変数 INCOME\_TAX に入れるには、次のように指定します。

```
SELECT scott.tax_rate (ss_no, sal)
       INTO income_tax
       FROM tax_table
       WHERE ss_no = tax_id;
```

## 命名規則

オプションのスキーマまたはパッケージの名前の中から1つだけを指定すると、最初の識別子はスキーマ名またはパッケージ名のどちらかになります。たとえば、PAYROLL.TAX\_RATE という参照内の PAYROLL がスキーマ名かパッケージ名かを判定するには、Oracle は次の手順を実行します。

- 現行スキーマ内の PAYROLL パッケージをチェックする。
- PAYROLL パッケージが見つからない場合は、トップ・レベルに TAX\_RATE 関数を含むスキーマ名 PAYROLL を検索する。そのような関数が見つからない場合には、エラー・メッセージを戻します。
- 現行スキーマ内で PAYROLL パッケージが見つかると、その中で TAX\_RATE 関数を検索する。そのような関数が見つからない場合には、エラー・メッセージを戻します。

また、ユーザーが定義したシノニムを使用して、ストアド・トップ・レベル・ファンクションを参照できます。

# 書式モデル

書式モデルは、文字列に格納される DATE データや NUMBER データの書式を記述する文字リテラルです。書式モデルは、次の目的で、TO\_CHAR 関数や TO\_DATE 関数の引数として使用できます。

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

なお、書式モデルによってデータベース内に格納された値の内部表現は変更されません。

この項では次の使用方法について説明します。

- 数値書式モデル
- 日付書式モデル
- 書式モデルの修飾子

## 戻り値の書式の変更

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

**例 1** 次の文は、部門 30 の従業員のコミッション値を選択し、TO\_CHAR 関数を使用して、そのコミッション値を数値書式モデル '\$9,990.99' で指定した書式の文字値に変換します。

```
SELECT ename employee, TO_CHAR(comm, '$9,990.99') commission
FROM emp
WHERE deptno = 30;
```

EMPLOYEE	COMMISSION
-----	-----
ALLEN	\$300.00
WARD	\$500.00
MARTIN	\$1,400.00
BLAKE	
TURNER	\$0.00
JAMES	

Oracle はこの書式モデルによって、ドル記号を先頭に付け、3 桁ごとにカンマで区切り、小数点以下 2 桁を持つコミッションを戻します。COMM 列が NULL のすべての従業員に対して TO\_CHAR 関数が NULL を戻していることに注目してください。

**例 2** 次の文は、部門 20 の各従業員の入社した日付を選択し、TO\_CHAR 関数を使用して、その日付を日付書式モデル 'fmMonth DD, YYYY' で指定した書式の文字列に変換します。

```
SELECT ename, TO_CHAR(Hiredate, 'fmMonth DD, YYYY') hiredate
```



```
FROM emp
WHERE deptno = 20;
```

ENAME	HIREDATE
SMITH	December 17, 1980
JONES	April 2, 1981
SCOTT	April 19, 1987
ADAMS	May 23, 1987
FORD	December 3, 1981
LEWIS	October 23, 1997

Oracle はこの書式モデルによって、省略しない月の名前(「書式モデルの修飾子」(3-70 ページ)で説明する "fm" で指定)、2 桁の日、世紀も含めた 4 桁の年で示された入社日付を戻します。

## 正しい書式の付与

書式モデルを使用して、あるデータ型の値を列が必要とする別のデータ型の値に変換する書式を指定できます。列の値を挿入したり、更新したりするとき、指定する値のデータ型は列のデータ型に一致しなければなりません。たとえば、DATE 列に挿入する値は、DATE データ型の値か、デフォルト日付書式の文字列値でなければなりません (Oracle は暗黙的にデフォルト日付書式の文字列を DATE データ型に変換する)。値が別の書式で与えられる場合、TO\_DATE 関数を使用して値を DATE データ型に変換しなければなりません。また、文字列の書式を指定するためにも書式モデルを使用しなければなりません。

**例** 次の文は、TO\_DATE 関数を使用して BAKER の入社日を更新します。文字列 '1992 05 20' を DATE 値に変換するために、書式マスク 'YYYY MM DD' を指定します。

```
UPDATE emp
SET hiredate = TO_DATE('1992 05 20', 'YYYY MM DD')
WHERE ename = 'BLAKE';
```

## 数値書式モデル

次の場所で数値書式モデルを使用できます。

- NUMBER データ型の値を VARCHAR2 データ型の値に変換する TO\_CHAR 関数
- CHAR データ型または VARCHAR2 データ型の値を NUMBER データ型の値に変換する TO\_NUMBER 関数

すべての数値書式モデルでは、数値が指定された有効桁数に丸められます。小数点左の有効桁数が書式で指定された桁数よりも多い場合、ポンド記号 (#) が値のかわりに戻されます。正の値が非常に大きく、指定の書式で表せない場合、無限大記号 (~) が値のかわりに戻され

ます。同様に、負の値が非常に小さく、指定の書式で表せない場合、負の無限大記号 (-~) が値のかわりに戻されます。

数値書式の要素

数値書式モデルは、1 つ以上の数値書式の要素から成り立ちます。表 3-12 に数値書式モデルの要素を示します。例を表 3-13 に示します。

- 数値書式モデルに書式要素 MI または S、PR が指定されていない場合、負の戻り値には自動的に負の符号が先頭に付加され、正の戻り値には空白が先頭に付加されます。
- 数値書式モデルには単一の小数点文字 (D) またはピリオド (.) しか指定できませんが、複数のグループ・セパレータ (G) やカンマ (,) は指定できます。
- 数値書式モデルの先頭文字にカンマ (,) は指定できません。
- 数値書式モデルにおいて、グループ・セパレータ (G) やカンマは小数点文字やピリオドの右側に指定できません。

表 3-12 数値書式の要素

要素	例	説明
9	9999	正の値の場合、先頭に空白を埋め込んで指定の桁数にしてから値を戻す。  負の値の場合、先頭に負の符号を埋め込んで指定の桁数にしてから値を戻す。  固定小数点数の整数部分の場合、先行ゼロに対して空白を戻す。ただし、値ゼロに対してはゼロを戻す。
0	0999	先行ゼロを戻す。
	9990	後続ゼロを戻す。
\$	\$9999	値の前にドル記号を付けて戻す。
B	B9999	整数部がゼロの場合、書式モデル内の "0" にかかわらず、固定小数点数の整数部に対して空白を戻す。
MI	9999MI	負の値の後に負の符号 "-" を戻す。  正の値の後に空白を戻す。
S	S9999	負の値の前に負の符号 "-" を戻す。  正の値の前に正の符号 "+" を戻す。
	9999S	負の値の後に負の符号 "-" を戻す。  正の値の後に正の符号 "+" を戻す。
PR	9999PR	山カッコ <> の中に負の値を戻す。  正の値の前後に空白を付けて戻す。

表 3-12 数値書式の要素

要素	例	説明
D	99D99	指定した位置に小数点文字 (すなわちピリオド ".") を戻す。
G	9G999	指定した位置にグループ・セパレータを戻す。
C	C999	指定した位置に ISO 通貨記号を戻す。
L	L999	指定した位置に各国通貨記号を戻す。
.(カンマ)	9,999	指定した位置にカンマを戻す。
.(ピリオド)	99.99	指定した位置に小数点 (すなわちピリオド ".") を戻す。
V	999V99	値を 10 の $n$ 乗倍にして戻す (必要に応じて数値を丸める)。この例では $n$ は "V" の後の "9" の数。
EEEE	9.9EEEE	科学表記法で値を戻す。
RN	RN	大文字のローマ数字で値を戻す。
m		小文字のローマ数字で値を戻す。 値は 1 ～ 3999 の整数となる。
FM	FM90.9	前後に空白を付けずに値を戻す。

例 表 3-13 は、異なる number 値と 'fmt' に対する次の問合せの結果を示しています。

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

表 3-13 数値変換例の結果

number	'fmt'	結果
-1234567890	9999999999S	'1234567890-'
0	99.99	' .00'
+0.1	99.99	' 0.10'
-0.2	99.99	' -.20'
0	90.99	' 0.00'
+0.1	90.99	' 0.10'
-0.2	90.99	' -0.20'
0	9999	' 0'
1	9999	' 1'

表 3-13 数値変換例の結果

number	'fmt'	結果
0	B9999	' '
1	B9999	' 1 '
0	B90.99	' '
+123.456	999.999	' 123.456 '
-123.456	999.999	' -123.456 '
+123.456	FM999.009	' 123.456 '
+123.456	9.9EEEE	' 1.2E+02 '
+1E+123	9.9EEEE	' 1.0E+123 '
+123.456	FM9.9EEEE	' 1.23E+02 '
+123.45	FM999.009	' 123.45 '
+123.0	FM999.009	' 123.00 '
+123.45	L999.99	' \$123.45 '
+123.45	FML99.99	' \$123.45 '
+1234567890	9999999999S	' 1234567890+ '

書式要素 **MI** および **PR** は、数値書式モデルの最後の位置にだけ指定できます。書式要素 **S** は、数値書式モデルの最初か最後の位置にだけ指定できます。

これらの書式要素によって戻される文字には、初期化パラメータによって指定されるものもあります。表 3-14 に、これらの要素とパラメータを示します。

表 3-14 初期化パラメータによって決定される数値書式要素の値

要素	説明	初期化パラメータ
D	小数点文字	NLS_NUMERIC_CHARACTER
G	グループ・セパレータ	NLS_NUMERIC_CHARACTER
C	ISO 通貨記号	NLS_ISO_CURRENCY
L	各国通貨記号	NLS_CURRENCY

これらの書式要素によって戻される文字は、初期化パラメータ **NLS\_TERRITORY** を使用して暗黙的に指定することもできます。これらのパラメータの詳細は、『Oracle8 Server リファレンス』を参照してください。

これらの書式要素によって戻される文字をセッションごとに変更するには、**ALTER SESSION** コマンドを使用します。また、デフォルト日付書式をセッションごとに変更する場

合も、ALTER SESSION コマンドを使用します。詳細は、「ALTER SESSION コマンド」(4-58 ページ) を参照してください。

日付書式モデル

次の場所で日付書式モデルを使用できます。

- デフォルト日付書式以外の書式の DATE 値を変換する TO\_CHAR 関数
- デフォルト日付書式以外の書式の文字値を変換する TO\_DATE 関数

デフォルト日付書式

デフォルト日付書式は、初期化パラメータ NLS\_DATE\_FORMAT で明示的に指定すること  
も、初期化パラメータ NLS\_TERRITORY で暗黙的に指定することもできます。これらのパ  
ラメータの詳細は、『Oracle8 Server リファレンス』を参照してください。

デフォルト日付書式をセッションごとに変更するには、ALTER SESSION コマンドを使用し  
ます。詳細は、「ALTER SESSION コマンド」(4-58 ページ) を参照してください。

最大長

日付書式モデルの合計長は最大 22 文字です。

日付書式要素

日付書式モデルは、表 3-15 に示す、1 つ以上の日付書式要素から構成されます。入力書式モ  
デルの場合、同じ書式項目は 2 回指定できません。また、類似した情報を表す書式項目を組  
み合わせることもできません。たとえば、'SYYYY' と 'BC' は同一の書式文字列内で使用できま  
せん。表 3-15 に示すように、日付書式要素には、TO\_DATE 関数で使えるものと使えないも  
のがあります。

表 3-15 日付書式要素

要素	TO_DATE で 指定できるか	意味
- / , . ; : 'text'	はい	結果に取り込まれる句読点とテキスト。
AD A.D.	はい	ビリオド付き / なしで西暦を示す。

表 3-15 日付書式要素

要素	TO_DATE で 指定できるか	意味
AM A.M.	はい	ピリオド付き / なしで午前を示す。
BC B.C.	はい	ピリオド付き / なしで紀元前を示す。
CC SCC	いいえ	4 桁の年の最初の 2 桁より 1 大きい数。"S" を指定すると紀元前の日付の先頭に "-" が付けられる。たとえば、'1900' の場合は '20' となる。
D	はい	曜日 (1 ～ 7)。
DAY	はい	曜日。9 文字分の長さまで空白が詰められる。
DD	はい	月における日 (1 ～ 31)。
DDD	はい	年における日 (1 ～ 366)。
DY	はい	曜日の省略形。
E	いいえ	時代名の略称 (日本、韓国、タイ)。
EE	いいえ	時代名の完全名称 (日本、韓国、タイ)。
HH	はい	時間 (1 ～ 12)。
HH12	いいえ	時間 (1 ～ 12)。
HH24	はい	時間 (0 ～ 23)。
IW	いいえ	ISO 規格に基づく、年における週 (1 ～ 52 または 1 ～ 53)。
IYY IY I	いいえ	それぞれ ISO 年の下 3 桁、2 桁、1 桁。
IYYY	いいえ	ISO 規格に基づいた 4 桁の年。
J	はい	ユリウス暦。紀元前 4712 年 1 月 1 日から経過した日数。'J' を付けて指定する数値は、整数でなければならない。
MI	はい	分 (0 ～ 59)。
MM	はい	月 (01 ～ 12; 1 月 =01)。
MON	はい	月の名前の省略形。
MONTH	はい	月の名前。9 文字分の長さまで空白が詰められる。

表 3-15 日付書式要素

要素	TO_DATE で 指定できるか	意味
PM P.M.	いいえ	ピリオド付き / なしで午前を示す。
Q	いいえ	年の四半期 (1、2、3、4; 1 月 ~ 3 月 =1)。
RM	はい	ローマ数字で表した月 (I ~ XII; 1 月 =I)。
RR	はい	年を 2 桁に丸める。指定した年が <50 で現在の年の下 2 桁が >=50 の場合は、次の世紀の年を戻す。指定した年が >=50 で現在の年の下 2 桁が <50 の場合は、前の世紀の年を戻す。
RRRR	はい	年を丸める。4 桁または 2 桁で入力できる。2 桁の場合、RR の場合と同様の結果が戻る。年を 4 桁で入力すれば、この処理は行われない。
SS	はい	秒 (0 ~ 59)。
SSSSS	はい	午前 0 時から経過した秒 (0 ~ 86399)。
WW	いいえ	年における週 (1 ~ 53)。第 1 週はその年の 1 月 1 日で始まり、1 月 7 日で終了する。
W	いいえ	月における週 (1 ~ 5)。第 1 週はその月の 1 日で始まり、7 日で終了する。
Y,YYY	はい	指定した位置にカンマを付けた年。
YEAR SYEAR	いいえ	綴りで表した年。"S" を指定すると紀元前の日付の先頭に "-" が付けられる。
YYYY SYYYY	はい	4 桁の年。"S" を指定すると紀元前の日付の先頭に "-" が付けられる。
YYY YY Y	はい	それぞれ年の下 3 桁、2 桁、1 桁。

書式文字列では句読点文字がある場所に、日付文字列の中では英数字がある場合、Oracle はエラーを戻します。次に例を示します。

```
TO_CHAR (TO_DATE('0297','MM/YY'), 'MM/YY')
```

この場合、エラーが戻ります。

## 日付書式要素と各国語サポート

いくつかの日付書式要素の機能は、Oracle を使用している国および言語に依存します。たとえば、以下の日付書式要素は綴りで値が戻されます。

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

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

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

これらの初期化パラメータの詳細は、『Oracle8 Server リファレンス・マニュアル』を参照してください。

## ISO 標準日付書式要素

Oracle は、ISO 規格に従った日付書式要素 `IYYYY` および `IYY`、`IY`、`I`、`IW` によって戻される値を計算します。これらの値と日付書式要素 `YYYY` および `YYY`、`YY`、`Y`、`WW` によって戻される値の相違点については、『Oracle8 Server リファレンス』の「各国語サポート」の章を参照してください。

## RR 日付書式要素

日付書式要素 `RR` は日付書式要素 `YY` に類似していますが、20 世紀以外の日付値を格納する上でより優れた柔軟性をもたらします。日付書式要素 `RR` によって、現在が 20 世紀でも、年の下 2 桁を指定するだけで、21 世紀の日付を格納できます。また、必要であれば、同じ方法により、21 世紀の時点でも 20 世紀の日付を格納できます。



TO\_DATE 関数で日付書式要素 YY を使用すると、日付値は常にその時点の世紀で戻されます。そのかわりに日付書式要素 RR を使用すると、年に指定した 2 桁の数とその年の下 2 桁の数によって戻り値の世紀が変化します。表 3-16 に RR 日付書式要素の性質をまとめます。

表 3-16 RR 日付書式要素

		指定された 2 桁の年	
		0 ～ 49	50 ～ 99
現在の年の 下 2 桁	0 ～ 49	戻る日付は現在の世紀	戻る日付は前の世紀
	50-99	戻る日付は次の世紀	戻る日付は現在の世紀

次の例によって日付書式要素 RR の性質を具体的に説明します。

例 1 これらの問合せが、1950 年～ 1999 年の間に発行されると想定します。

```
SELECT TO_CHAR(TO_DATE(' 27-OCT-95', 'DD-MON-RR'), 'YYYY') "Year"
FROM DUAL;
```

```
Year
----
1995
```

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

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

例 2 これらの問合せが、2000 年～ 2049 年の間に発行されると想定します。

```
SELECT TO_CHAR(TO_DATE(' 27-OCT-95', 'DD-MON-RR'), 'YYYY') "Year";
FROM DUAL;
```

```
Year
----
1995
```

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

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

発行される年 (2000 年の前または後) にかかわらず、問合せが同じ値を戻していることに注目してください。日付書式要素 **RR** によって、世紀が変わっても同じ値を戻す **SQL** 文を記述できます。

日付書式要素の接尾辞

表 3-17 に日付書式要素に付加できる接尾辞を示します。

表 3-17 日付書式要素の接尾辞

接尾辞	意味	要素の例	値の例
TH	序数	DDTH	4TH
SP	綴りで表した数	DDSP	FOUR
SPTH または THSP	綴りで表した序数	DDSPTH	FOURTH

これらの接尾辞を 1 つでも日付書式要素に付加すると、戻り値は常に英語です。

注意： 日付の接尾辞は出力でだけ有効です。データベースに日付を挿入するためには使用できません。

日付書式要素での先頭文字の大文字化

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

日付書式モデルにおける句読点と文字リテラル

日付書式モデルでは次の文字も指定できます。

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

これらの文字は、戻り値の中で書式モデルに指定された同じ位置に現れます。

書式モデルの修飾子

**TO\_CHAR** 関数の書式モデルで修飾子 **FM** と **FX** を使用して、空白の埋め方および書式検査を制御できます。

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

**FM** "Fill mode( 埋込みモード )" です。この修飾子は **TO\_CHAR** 関数の戻り値における空白の埋込みを抑制します。

- **TO\_CHAR** 関数の日付書式要素では、この修飾子は後続の文字要素 (**MONTH** など) では空白を抑制し、日付書式モデルの後続の数要素 (**MI** など) では前後のゼロを抑制します。**FM** を指定していない場合、文字要素の結果は常に右に空白を埋め込んだ固定長となり、数要素に対しては常に先行ゼロが戻されます。**FM** を指定した場合、空白の埋込みがないために、戻り値の長さが異なることもあります。
- **TO\_CHAR** 関数の数値書式要素では、この修飾子によって数値の左に加えられた空白が抑制されます。これにより、その結果は出力バッファ中で左揃えになります。**FM** を指定していない場合、結果は常に右揃えとなり、数の左に空白が埋め込まれます。

**FX** "Format exact( 厳密な書式一致 )" です。この修飾子は **TO\_DATE** 関数の文字引数と日付書式モデルに対して、厳密な一致を指定します。

- 文字引数における句読点と引用符で囲まれたテキストは書式モデルの対応する部分と (大文字小文字の違いを除いて) 厳密に一致しなければなりません。
- 文字引数には余分な空白を含めることはできません。**FX** を指定していない場合、**Oracle** は余分な空白を無視します。
- 文字引数における数値データの桁数は、書式モデルの対応する要素と同じ桁数でなければなりません。**FX** を指定していない場合、文字引数における数値により先行ゼロが省略されます。

**FX** が使用可能になっているとき、**FM** 修飾子を指定して、この先行ゼロの検査を使用禁止にできます。

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

**例 1** 次の文は日付書式モデルを使用して文字式を戻します。

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

```
Ides
-----
3RD of April, 1995
```

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

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

```
Ides
-----
03RD of April , 1995
```

例 2 次の文は、2 つの連続した単一引用符を含む日付書式モデルを使用することによって、戻り値に単一引用符を含めます。

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

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

書式モデル内の文字リテラルにおいても、同じ目的で単一引用符を 2 つ連続して使用できます。

例 3 表 3-18 は、*char* 値と *'fmt'* の様々な組合せに対し、次の文が FX を使用したマッチング条件を満たしているかどうかを示しています。

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

表 3-18 FX 書式モデル修飾子によるマッチング文字データと書式モデル

char	'fmt'	一致とエラー
'15/ JAN /1993'	'DD-MON-YYYY'	一致
' 15! JAN % /1993'	'DD-MON-YYYY'	エラー
'15/JAN/1993'	'FXDD-MON-YYYY'	エラー
'15-JAN-1993'	'FXDD-MON-YYYY'	一致
'1-JAN-1993'	'FXDD-MON-YYYY'	エラー
'01-JAN-1993'	'FXDD-MON-YYYY'	一致
'1-JAN-1993'	'FXFMDD-MON-YYYY'	一致

### 文字列から日付への変換に関する規則

次の追加の形式化の規則は、文字列値を日付値に変換する場合に適用されます。

- 先行ゼロを含む数値書式要素の桁がすべて指定されている場合は、日付文字列から書式文字列に含まれる句読点を省略できます。つまり、MM、DD、YY などの 2 桁の書式要素については、2 のかわりに 02 を指定してください。
- 日付文字列から、書式文字列の最後にある時間フィールドを省略できます。

- 日付書式要素と日付文字列内の対応する文字のマッチングに失敗した場合、表 3-19 に示すように、元の書式要素のかわりに、別の書式要素の適用が試みられます。

表 3-19 Oracle 形式マッピング

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

式

式は、1 つ以上の値および演算子、値を評価する SQL 関数の組合せです。一般的に式のデータ型は、そのコンポーネントのデータ型となります。

次の単純式は、値が 4 になり、データ型は NUMBER データ型（コンポーネントと同じデータ型）になります。

2\*2

次の例は、関数と演算子を使用した複雑な式です。この式は今日の日付に 7 日を加算し、その合計から時間コンポーネントを削除し、結果を CHAR データ型に変換します。

TO\_CHAR ( TRUNC ( SYSDATE+7 ) )

次の場所で式を使用できます。

- SELECT コマンドの選択リスト
- WHERE 句と HAVING 句の条件
- CONNECT BY 句および START WITH 句、ORDER BY 句
- INSERT コマンドの VALUES 句
- UPDATE コマンドの SET 句

たとえば、次の UPDATE 文の SET 句で引用符で囲まれた文字列 'smith' のかわりに式を使用することもできます。

SET ename = 'smith';

この SET 句では、引用符で囲まれた文字列 'smith' のかわりに、LOWER(ename) を使用しています。

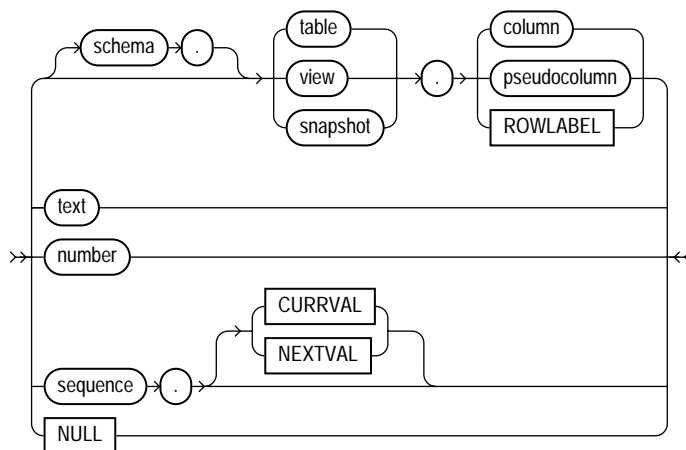
```
SET ename = LOWER(ename);
```

式にはいくつかの書式があります。Oracle は、SQL 全コマンドのすべての部分で、式の書式を全部受け入れるわけではありません。このマニュアルの他の箇所で、条件または SQL 関数、SQL コマンドに *expr* が示されている場合は、必ず適切な式の表記法を使用してください。このマニュアルの第 4 章「コマンド」にある各コマンドの項では、コマンドに指定する式の制限について説明してあります。次の項では、いくつかの例をあげて様々な式の形式を説明します。

## 書式 1

列または疑似列、定数、順序番号、NULL。

**expr\_form\_1 ::=**



スキーマは各自用の他に、"PUBLIC"(二重引用符が必要)にもなり得ます。その場合、スキーマは表またはビュー、スナップショットのパブリック・シノニムを修飾しなければなりません。"PUBLIC"でのパブリック・シノニムの修飾は、データ操作言語(DML)のコマンドでだけサポートされています。データ定義言語(DDL)のコマンドではサポートされていません。

疑似列は、LEVEL、ROWID、ROWNUM のいずれかです。疑似列は、表でだけ使用でき、ビューやスナップショットでは使用できません。NCHAR と NVARCHAR2 は、有効な疑似列または ROWLABEL データ型ではありません。疑似列については、「疑似列」(2-30 ページ)を参照してください。

Trusted Oracle を使用していない場合、式 ROWLABEL は常に NULL を返します。ラベルと ROWLABEL の使用方法については、使用している Trusted Oracle のマニュアルを参照してください。

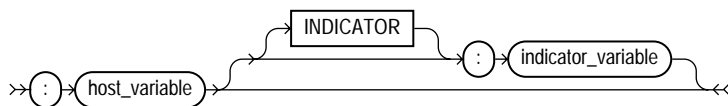
有効な書式 1 の式の例を次に示します。

```
emp.ename
'this is a text string'
10
N'this is an NCHAR string'
```

## 書式 2

オプションの標識変数を持つホスト変数。この書式の式は埋込み SQL 文または Oracle コール・インタフェース (OCI)・プログラムで処理される SQL 文に限り指定できます。

**expr\_form\_II ::=**



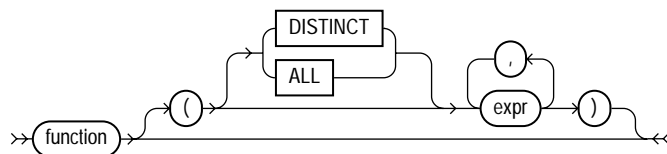
有効な書式 2 の式の例を次に示します。

```
:employee_name INDICATOR :employee_name_indicator_var
:department_location
```

## 書式 3

単一行を操作する SQL 関数の呼出し。

**expr\_form\_III ::=**



有効な書式 3 の式の例を次に示します。

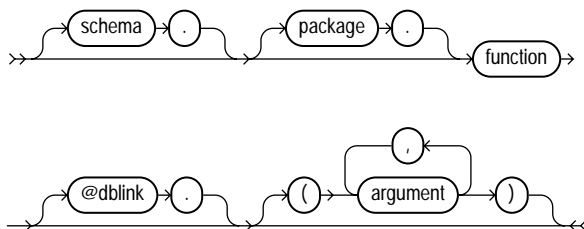
```
LENGTH('BLAKE')
ROUND(1234.567*43)
SYSDATE
```

SQL 関数については、「SQL 関数」(3-15 ページ)を参照してください。

## 書式 4

ユーザー関数の呼出し。

**expr\_form\_IV ::=**



有効な書式 4 の式の例を次に示します。

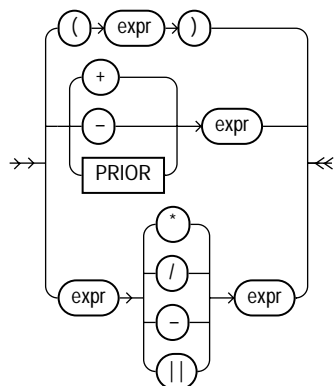
```
circle_area (radius)
payroll.tax_rate (empno)
scott.payroll.tax_rate(dependents, empno)@ny
```

ユーザー関数については、「ユーザー関数」(3-57 ページ) を参照してください。

## 書式 5

その他の式の組合せ。

**expr\_form\_V ::=**



関数の組合せによっては、適切でないものや拒否されるものもありますので、注意してください。たとえば、LENGTH 関数はグループ関数内では使用できません。



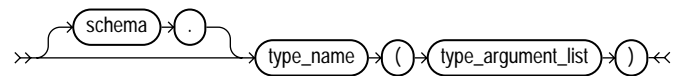
有効な書式 5 の式の例を示します。

```
( 'CLARK' || 'SMITH' )
LENGTH( 'MOOSE' ) * 57
SQRT(144) + 72
my_fun( TO_CHAR( sysdate, 'DD-MMM-YY' )
```

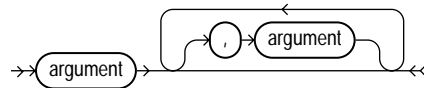
## 書式 6

型コンストラクタの呼出し。

**expr\_form\_VI ::=**



**type\_argument\_list ::=**



*type\_name* がオブジェクト型である場合、型引数のリストは、最初の引数の型がオブジェクト型の最初の属性に一致する値を取り、2 番目の引数の型がオブジェクト型の 2 番目の属性に一致するというように、順序付けられた引数のリストになっていなければなりません。コンストラクタの引数の合計は、オブジェクト型の属性の合計と一致しなければなりません。引数の最大数は 999 です。

*type\_name* が **VARRAY** またはネストされた表型である場合、引数のリストには 0 個以上の引数を含めることができます。0 の引数は、空集合の構造であることを示します。それ以外の場合は、各引数が、型が集合型の要素型である要素値に対応します。

*type\_name* がオブジェクト型、**VARRAY**、ネストされた表型のいずれであっても、引数の最大数は 999 です。

### 例

```
CREATE TYPE address_t AS OBJECT
  (no NUMBER, street CHAR(31), city CHAR(21), state CHAR(3), zip NUMBER);
CREATE TYPE address_book_t AS TABLE OF address_t;
DECLARE
  /* Object Type variable initialized via Object Type Constructor */
  myaddr address_t = address_t(500, 'Oracle Parkway', 'Redwood Shores',
    'CA', 94065);
  /* nested table variable initialized to an empty table via a
    constructor*/
```

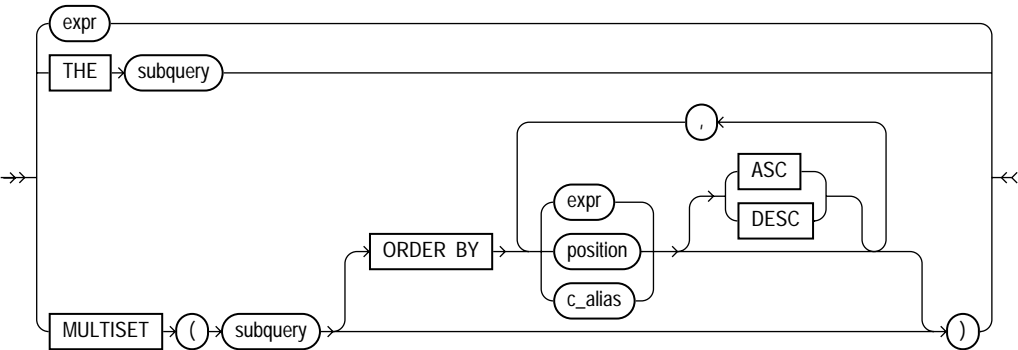
```
alladdr address_book_t = address_book_t();
BEGIN
  /* below is an example of a nested table constructor with two elements
     specified, where each element is specified as an object type
     constructor. */
  insert into employee values (666999, address_book_t(address_t(500,
    'Oracle Parkway', 'Redwood Shores', 'CA', 94065), address_t(400,
    'Mission Street', 'Fremont', 'CA', 94555)));
END;
```

**書式 7**

集合型の値を別の集合型の値に変換します。



operand ::=



CASTを使用すると、ある型の集合型値を別の集合型に変換できます。名前のない集合（副問合せの結果集合など）または名前付きの集合（VARRAY またはネストされた表など）を型互換の名前付き集合にキャストできます。*type\_name* は集合型の名前でなければならず、*operand* の値は集合値でなければなりません。

名前付き集合型を別の名前付き集合型にキャストするには、両方の集合の要素が同じ型でなければなりません。

副問合せの結果集合が複数行に評価される可能性がある場合は、MULTISET キーワードを指定する必要があります。副問合せの結果である行は、それらの行がキャストされた集合値の要素を形成します。MULTISET キーワードを指定しないと、副問合せは、CAST 式ではサポートされないスカラー副問合せとして扱われます。つまり、Oracle8 ではスカラー副問合せを CAST 演算子の引数にすることは無効です。

この項の CAST の例では、次のユーザー定義の型と表を使用します。

```
CREATE TYPE address_t AS OBJECT
    (no NUMBER, street CHAR(31), city CHAR(21), state CHAR(2));
CREATE TYPE address_book_t AS TABLE OF address_t;
CREATE TYPE address_array_t AS VARRAY(3) OF address_t;
CREATE TABLE emp_address (empno NUMBER, no NUMBER, street CHAR(31),
    city CHAR(21), state CHAR(2));
CREATE TABLE employees (empno NUMBER, name CHAR(31));
CREATE TABLE dept (dno NUMBER, addresses address_array_t);
```

### 例 1

副問合せを CAST します。

```
SELECT e.empno, e.name, CAST(MULTISET(SELECT ea.no, ea.street,
    ea.city, ea.state
    FROM emp_address ea,
    WHERE ea.empno = e.empno)
    AS address_book_t)
FROM employees e;
```

### 例 2

CAST では、VARRAY 型の列を、ネストされた表に変換します。表の値は、フラット化した副問合せによって生成されます。「フラット化した副問合せの使用」(4-528 ページ)を参照してください。

```
SELECT *
FROM THE(SELECT CAST(d.addresses AS address_book_t)
    FROM dept d
    WHERE d.dno = 111) a
WHERE a.city = 'Redwood Shores';
```

### 例 3

次の例は、ORDER BY 句で MULTISET 式を割り当てます。

```
CREATE TABLE projects (empid NUMBER, projname VARCHAR2(10));
CREATE TABLE employees (empid NUMBER, ename VARCHAR2(10));
CREATE TYPE projname_table_type AS TABLE OF VARCHAR2(10);
```

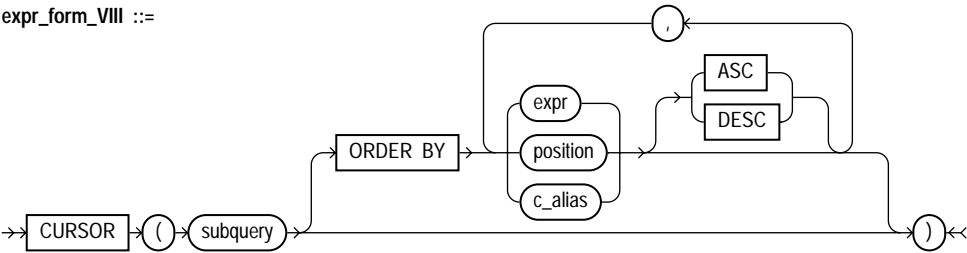
上記のスキーマでの MULTISET 式の例を次にあげます。

```
SELECT e.name, CAST(MULTISET(SELECT p.projname
    FROM projects p
    WHERE p.empid=e.empid
    ORDER BY p.projname)
    AS projname_table_type)
FROM employees e;
```

書式 8

ネストされた CURSOR(カーソル)を戻します。この書式の式は、PL/SQL REF カーソルと同様です。

expr\_form\_VIII ::=



ネストされたカーソルは、含まれている行が親カーソルからフェッチされると、暗黙的にオープンされます。ネストされたカーソルは、次の場合にはクローズしています。

- ユーザーによって明示的にクローズされたとき
- 親カーソルが再実行されたとき
- 親カーソルがクローズされたとき
- 親カーソルが取り消されたとき
- 親カーソルの 1 つでのフェッチ時にエラーが発生したとき（クリーン・アップの一部としてクローズされる）

CURSOR 式には次の制限が適用されます。

- ネストされたカーソルは、他の問合せ式の中でネストされていない SELECT 文の中でだけ適用できる。ただし、CURSOR 式自体の副問合せの場合を除きます。
- ネストされたカーソルは、問合せ指定の最も外側の SELECT リストだけで表示できる。
- ネストされたカーソルはビューに表示できない。
- ネストされたカーソルに対して BIND 操作と EXECUTE 操作は実行できない。

例

```
SELECT d.deptno, CURSOR(SELECT e.empno, CURSOR(SELECT p.projnum,
                                                p.projname
                                                FROM projects p
                                                WHERE p.empno = e.empno)
                        FROM TABLE(d.employees) e)
FROM dept d
WHERE d.dno = 605;
```

## 書式 9

オブジェクトに対する参照のコンストラクタ。

**expr\_form\_IX ::=**

→ REF → ( → correlation\_variable → ) ←

SQL 文では、REF の引数として、オブジェクト表またはオブジェクト・ビューの行に対応付けられている表別名がとられます。変数または行にバインドされたオブジェクト・インスタンスについての REF 値が戻ります。REF の詳細は、『Oracle8 概要』を参照してください。

### 例 1

```
SELECT REF(e)
FROM employee_t e
WHERE e.empno = 10000;
```

例 2 この例は REF を述語の中で使用します。

```
SELECT e.name
FROM employee_t
     e INTO :x
WHERE REF(e) = empref1;
```

## 書式 10

行オブジェクトを戻します。

**expr\_form\_X ::=**

→ VALUE → ( → correlation\_variable → ) ←

SQL 文では、VALUE の引数として、オブジェクト表の行に対応付けられている相関変数（表別名）がとられます。

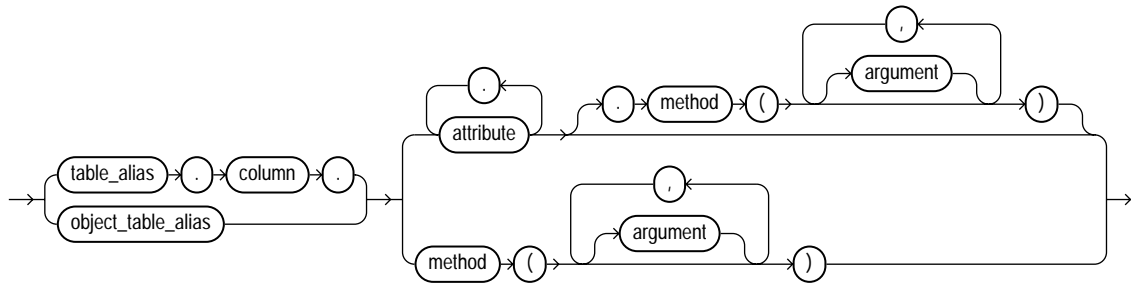
### 例

```
SELECT VALUE(e)
FROM employee e
WHERE e.name = 'John Smith';
```

## 書式 11

属性参照およびメソッドの起動。

**expr\_form\_XI::=**



**column** パラメータはオブジェクトまたは REF 列です。この項の例では、次のユーザー定義の型と表を使用します。

```
CREATE OR REPLACE TYPE employee_t AS OBJECT
(
    empid NUMBER,
    name CHAR(31),
    birthdate DATE,
    MEMBER FUNCTION age RETURN NUMBER,
    PRAGMA RESTRICT_REFERENCES(age, RNPS, WNPS, WNDS)
);

CREATE OR REPLACE TYPE BODY employee_t AS
MEMBER FUNCTION age RETURN NUMBER IS
    var NUMBER;
BEGIN
    var := months_between(ROUND(SYSDATE, 'YEAR'),
        ROUND(birthdate, 'YEAR'))/12;
    RETURN var ;
END;
/

CREATE TABLE department (dno NUMBER, manager EMPLOYEE_T);
```

**例** 次の例では、上記の例で定義したオブジェクト列とメソッドを更新し、そこから選択します。

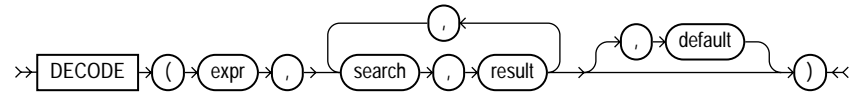
```
UPDATE department d
SET d.manager.empid = 100;

SELECT d.manager.name, d.manager.age()
FROM department d;
```

## デコード式

特別の DECODE 構文を使用する式。

`decode_expr ::=`



この式を評価するために、Oracle は各 *search* 値と *expr* を 1 つずつ比較します。*expr* が *search* に等しい場合に Oracle は対応する *result* を戻します。一致しない場合には *default* が戻されます。*default* が指定されていない場合は NULL が戻されます。*expr* と *search* が文字データを含む場合、Oracle は非空白埋め比較方法で比較します。これらの比較方法については、「データ型の比較規則」(2-22 ページ)を参照してください。

*search*、*result*、*default* の値を式から導出できます。Oracle は *expr* と比較する前に各 *search* 値を評価します。*expr* と比較する前にすべての *search* 値を評価するわけではありません。その結果、*expr* に等しい *search* が見つかり、Oracle はその後の *search* を評価しません。

比較する前に、Oracle は *expr* と各 *search* 値を最初の *search* 値のデータ型に自動的に変換します。Oracle は戻り値を最初の *result* と同じデータ型に自動的に変換します。最初の *result* のデータ型が CHAR である場合、あるいは最初の *result* が NULL である場合、Oracle は戻り値を VARCHAR2 データ型の値に変換します。データ型の変換については、「データ変換」(2-26 ページ)を参照してください。

DECODE 式では、Oracle は 2 つの NULL を等価と見なします。*expr* が NULL の場合、Oracle は最初の *search* 値の *result* も NULL として戻します。

DECODE 式のコンポーネントの最大数は、*expr*、*search*、*result*、*default* を含めて 255 です。

**例** 次の式は値 DEPTNO をデコードします。DEPTNO が 10 である場合、式は 'ACCOUNTING' を戻します。同様に、DEPTNO が 20 である場合、式は 'RESEARCH' を戻します。DEPTNO が 10 または 20、30、40 のどれでもない場合、式は 'NONE' を戻します。

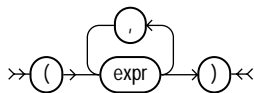
```

DECODE (deptno,10, 'ACCOUNTING',
        20, 'RESEARCH',
        30, 'SALES',
        40, 'OPERATION',
        'NONE')
  
```

## 式のリスト

カッコで囲まれ、カンマで区切られた一連の式。

`expr_list ::=`



選択リストとして 1000 以内の式を指定できます。有効な式のリストの例を次に示します。

```
10, 20, 40)
('SCOTT', 'BLAKE', 'TAYLOR')
(LENGTH('MOOSE') * 57, -SQRT(144) + 72, 69)
```

## 条件

条件は、1 つ以上の式と論理演算子の組合せで指定し、TRUE(真)またはFALSE(偽)、UNKNOWN(不定)のいずれかに評価されます。このマニュアルの第4章「コマンド」で説明する SQL コマンドの中に *condition* が示されている場合には、必ずこの構文を使用してください。

次の文の WHERE 句で条件を使用できます。

- DELETE
- SELECT
- UPDATE

SELECT コマンドの次の任意の句で、条件を使用できます。

- WHERE
- START WITH
- CONNECT BY
- HAVING

条件は「論理」データ型であるということもできます。ただし、Oracle で、正式にこのようなデータ型をサポートするわけではありません。

次のような単純な条件は、常に TRUE に評価されます。

```
1 = 1
```

次のやや複雑な条件は、SAL 値を COMM 値に加算し (NULL は 0 で置き換える)、その合計が定数 2500 よりも大きいかどうかを判定します。



```
NVL(sal, 0) + NVL(comm, 0) > 2500
```

論理演算子を使用すると、複数の条件を単一の条件に結合できます。たとえば、次のように AND 演算子を使用して 2 つの条件を結合できます。

```
(1 = 1) AND (5 < 7)
```

有効な条件の例を次に示します。

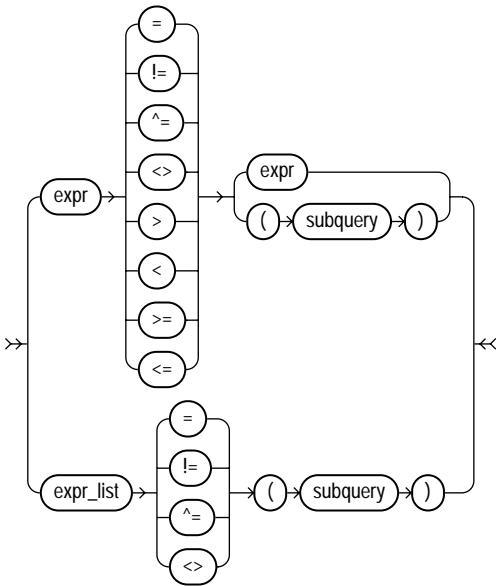
```
name = 'SMITH'  
emp.deptno = dept.deptno  
hiredate > '01-JAN-88'  
job IN ('PRESIDENT', 'CLERK', 'ANALYST')  
sal BETWEEN 500 AND 1000  
comm IS NULL AND sal = 2000
```

条件には次に示すような複数の書式があります。このマニュアルの第 4 章「コマンド」にある各コマンドの説明では、コマンドに指定する条件の制限について説明しています。以降の項で様々な形式の条件を説明します。

書式 1

式または副問合せの結果との比較。

`condition_form_1 ::=`

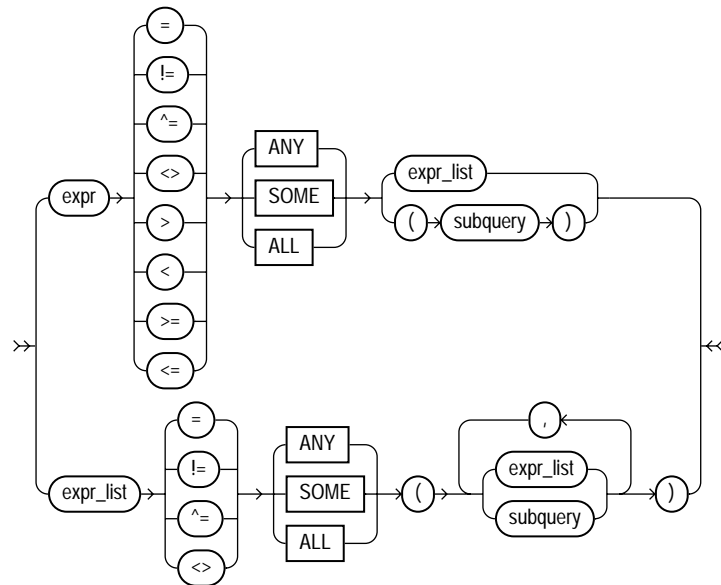


比較演算子については、「比較演算子」(3-5 ページ)を参照してください。

## 書式 2

リストまたは副問合せ内の任意またはすべてのメンバーとの比較。

**condition\_form\_II ::=**

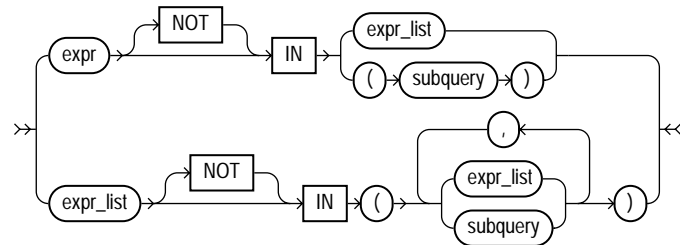


「副問合せ」（4-525 ページ）を参照してください。

## 書式 3

リストまたは副問合せ内のメンバーシップの検査。

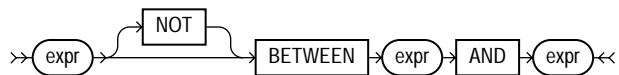
**condition\_form\_III ::=**



## 書式 4

範囲に含まれていることの検査。

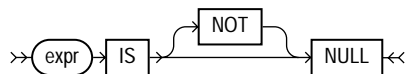
**condition\_form\_IV ::=**



## 書式 5

NULL の検査。

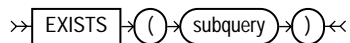
**condition\_form\_V ::=**



## 書式 6

副問合せにおける行の存在の検査。

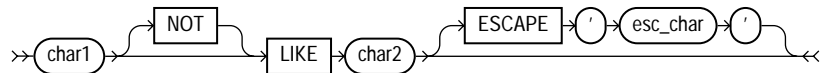
**condition\_form\_VI ::=**



## 書式 7

パターン・マッチングを含む検査。

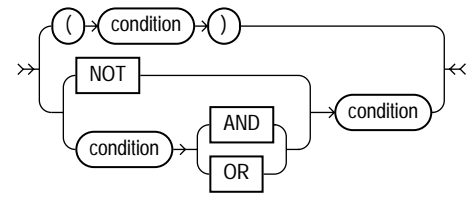
**condition\_form\_VII ::=**



書式 8

その他の条件の組合せ。

**condition\_form\_VIII ::=**





この章では、Oracle SQL のコマンドと句をアルファベット順に説明します。

---

---

**注意：** コマンドと句の説明で**OBJ**の印が前に付いている箇所は、Oracle Object Option がデータベース・サーバーにインストールされている場合にだけ読んでください。

---

---

各コマンドと句の説明は、次のような形式になっています。

用途	コマンドの基本的な使用方法を説明します。
前提条件	コマンドの実行に必要な権限と、コマンドを使用する前に実行すべき手順を示します。なお、特に断りのない限り、コマンドを使用するユーザーのインスタンスでデータベースがオープンされている必要があります。
構文	コマンドを構成するキーワードとパラメータを示します。
キーワードとパラメータ	キーワードとパラメータの用途を説明します。この章で使用するキーワードとパラメータの表記上の規則については、「はじめに」に説明があります。  使用上の注意：「キーワードとパラメータ」の後に、例を示す項が続く場合があります。この項では、コマンドをいつ、どのように使用するかを説明します。
関連項目	関連するコマンドおよび句、マニュアルの項を記載します。

## SQL コマンドの概要

次項以降の表は SQL コマンドの機能の概要を示しています。SQL コマンドは機能によって次のように分類されます。

- データ定義言語 (DDL) コマンド
- データ操作言語 (DML) コマンド
- トランザクション制御コマンド
- セッション制御コマンド
- システム制御コマンド

### データ定義言語 (DDL) コマンド

データ定義言語 (DDL) コマンドを使用することにより、次のタスクを実行できます。

- スキーマ・オブジェクトの作成および変更、削除
- 権限とロールの付与と取消し
- 表および索引、クラスタの情報の分析
- 監査オプションの設定
- データ・ディクショナリへのコメントの追加

CREATE および ALTER、DROP の各コマンドでは、処理されるオブジェクトに対して排他的アクセスが必要です。たとえば、ALTER TABLE コマンドで指定している表に別のユーザーがトランザクションをオープンしていると、コマンドは失敗します。

GRANT および REVOKE、ANALYZE、AUDIT、COMMENT の各コマンドでは、処理されるオブジェクトに対する排他的アクセスは必要ありません。たとえば、他のユーザーによって更新操作中の表を分析できます。

各 DDL 文の前後で、現行のトランザクションが暗黙的にコミットされます。

DDL 文によって、通常はスキーマ・オブジェクトの再コンパイルと再許可が行われます。Oracle によって行われるスキーマ・オブジェクトの再コンパイルと再許可の方法、および DDL 文によってこれらが引き起こされる環境に関する詳細は、『Oracle8 Server 概要』を参照してください。

DDL コマンドは、PL/SQL では直接サポートされていませんが、オラクル社が提供するパッケージ・プロシージャを使うと、使用できます。詳細は、『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。



表 4-1 に DDL コマンドを示します。

表 4-1 データ定義言語コマンド

コマンド	用途
ALTER CLUSTER	クラスタの記憶特性の変更 クラスタのエクステントの割当て
ALTER DATABASE	データベースのオープンとマウント Oracle8 移行時の Oracle7 データ・ディクショナリの変換 旧リリースの Oracle へのダウングレードのための準備 ARCHIVELOG/NOARCHIVELOG モードの選択 メディア回復の実行 REDO ログ・ファイルのグループ・メンバーの追加および削除、消去 データ・ファイルと REDO ログ・ファイル・メンバーの改名 現行の制御ファイルのバックアップの作成 トレース・ファイルへの SQL コマンドのバックアップの作成（データベースの再作成に使用） 新しいデータ・ファイルの作成 1 つ以上のデータ・ファイルのサイズ変更 古いデータ・ファイルにかわる回復用の新しいデータ・ファイルの作成 データ・ファイルのサイズの自動拡張の使用可能と使用禁止の切替え データ・ファイルのオンラインとオフラインの切替え REDO ログ・ファイル・グループのスレッドの使用可能と使用禁止の切替え データベースのグローバル名の変更
ALTER FUNCTION	ストアド・ファンクションの再コンパイル
ALTER INDEX	索引の記憶割当ての再定義
ALTER PACKAGE	ストアド・パッケージの再コンパイル
ALTER PROCEDURE	ストアド・プロシージャの再コンパイル
ALTER PROFILE	プロファイルに対するリソース制限の追加と削除
ALTER RESOURCE COST	セッションで使用されるリソースの合計を計算する式の指定
ALTER ROLE	ロールのアクセスに必要な許可の変更

表 4-1 データ定義言語コマンド

コマンド	用途
ALTER ROLLBACK SEGMENT	ロールバック・セグメントの記憶特性の変更 ロールバック・セグメントのオンラインとオフラインの切替え ロールバック・セグメントの最適サイズまたは指定サイズへの縮小
ALTER SEQUENCE	順序の値生成の再定義
ALTER SNAPSHOT	スナップショットの記憶特性、自動リフレッシュ時間、自動リフレッシュ・モードの変更
ALTER SHAPSHOT LOG	スナップショット・ログの記憶特性の変更
ALTER TABLE	表に対する列と整合性制約の追加 表の記憶特性を変更するための列の再定義 整合性制約の使用可能と使用禁止の切替えや削除 表に対する表ロックの使用可能と使用禁止の切替え 表に関するすべてのトリガーの使用可能と使用禁止の切替え 表に対するエクステンツの割当て 表への書込みの許可と禁止 表に対する並行度の変更
ALTER TABLESPACE	データ・ファイルの追加と改名 記憶特性の変更 表領域のオンラインとオフラインの切替え バックアップの開始と終了の設定 表領域への書込みの許可と禁止
ALTER TRIGGER	データベース・トリガーの使用可能と使用禁止の切替え
ALTER TYPE	ユーザー定義型の変更
ALTER USER	ユーザーのパスワード、デフォルトの表領域、一時表領域、表領域の割当て制限、プロファイル、デフォルト・ロールの変更
ALTER VIEW	ビューの再コンパイル
ANALYZE	表、クラスタ、索引に対する、パフォーマンス統計の収集、構造の妥当性検査、連鎖行の識別
AUDIT	指定した SQL コマンドまたはスキーマ・オブジェクトの操作に対する監査の選択
COMMENT	表、ビュー、スナップショット、列に関するコメントのデータ・ディクショナリへの追加

表 4-1 データ定義言語コマンド

コマンド	用途
CREATE CLUSTER	1 つ以上の表を含むクラスタの作成
CREATE CONTROLFILE	制御ファイルの再作成
CREATE DATABASE	データベースの作成
CREATE DATABASE LINK	リモート・データベースへのリンクの作成
CREATE DIRECTORY	データベース外に保存されている BFILE へのアクセスと BFILE の使用を管理するためのディレクトリ・データベース・オブジェクトの作成
CREATE FUNCTION	ストアド・ファンクションの作成
CREATE INDEX	表またはクラスタに対する索引の作成
CREATE LIBRARY	SQL と PL/SQL による第 3 世代言語 (3GL) の外部関数とプロシージャのコールを可能にするライブラリの作成
CREATE PACKAGE	ストアド・パッケージの仕様部の作成
CREATE PACKAGE BODY	ストアド・パッケージの本体の作成
CREATE PROCEDURE	ストアド・プロシージャの作成
CREATE PROFILE	プロファイルの作成とリソース制限の指定
CREATE ROLE	ロールの作成
CREATE ROLLBACK SEGMENT	ロールバック・セグメントの作成
CREATE SCHEMA	単一トランザクションでの複数の CREATE TABLE 文および CREATE VIEW 文、GRANT 文の発行
CREATE SEQUENCE	逐次値を生成するための順序の作成
CREATE SHAPSHOT	1 つ以上のリモート・マスター表から取得するデータのスナップショットの作成
CREATE SNAPSHOT LOG	スナップショットのマスター表に加えられた変更が記録されているスナップショット・ログの作成
CREATE SYNONYM	スキーマ・オブジェクトのシノニムの作成
CREATE TABLE	列、整合性制約、記憶域割当てを定義した表の作成
CREATE TABLESPACE	スキーマ・オブジェクト、ロールバック・セグメント、一時セグメントの記憶領域の作成、表領域を構成するデータ・ファイルの指定
CREATE TRIGGER	データベース・トリガーの作成
CREATE TYPE	オブジェクト型または名前付き可変配列 (VARRAY)、ネストした表型、不完全なオブジェクト型の作成
CREATE USER	データベース・ユーザーの作成

表 4-1 データ定義言語コマンド

コマンド	用途
CREATE VIEW	1 つ以上の表またはビューに対するビューの定義
DROP CLUSTER	データベースからのクラスタの削除
DROP DATABASE LINK	データベース・リンクの削除
DROP DIRECTORY	データベースからのディレクトリの削除
DROP FUNCTION	データベースからのストアド・ファンクションの削除
DROP INDEX	データベースからの索引の削除
DROP LIBRARY	データベースからのライブラリ・オブジェクトの削除
DROP PACKAGE	データベースからのストアド・パッケージの削除
DROP PROCEDURE	データベースからのストアド・プロシージャの削除
DROP PROFILE	データベースからのプロファイルの削除
DROP ROLE	データベースからのロールの削除
DROP ROLLBACK SEGMENT	データベースからのロールバック・セグメントの削除
DROP SEQUENCE	データベースからの順序の削除
DROP SNAPSHOT	データベースからのスナップショットの削除
DROP SNAPSHOT LOG	データベースからのスナップショット・ログの削除
DROP SYNONYM	データベースからのシノニムの削除
DROP TABLE	データベースからの表の削除
DROP TABLESPACE	データベースからの表領域の削除
DROP TRIGGER	データベースからのトリガーの削除
DROP TYPE	データベースからのユーザー定義型の削除
DROP USER	データベースからのユーザーおよびそのユーザーのスキーマ内のオブジェクトの削除
DROP VIEW	データベースからのビューの削除
GRANT	ユーザーとロールに対するシステム権限およびロール、オブジェクト権限の付与
NOAUDIT	前に発した AUDIT 文の効果の一部または全部を取り消すことによる監査の使用禁止

表 4-1 データ定義言語コマンド

コマンド	用途
RENAME	スキーマ・オブジェクトの名前の変更
REVOKE	ユーザーまたはロールからの、システム権限およびロール、オブジェクト権限の取消し
TRUNCATE	表またはクラスタの全行の削除と、それらの行で使用されていた領域の解放

データ操作言語 (DML) コマンド

データ操作言語 (DML) コマンドは、既存のスキーマ・オブジェクト内のデータの問合せと操作を行います。これらのコマンドは、現行トランザクションを暗黙的にコミットしません。

表 4-2 データ操作言語コマンド

コマンド	用途
DELETE	表からの行の削除
EXPLAIN PLAN	SQL 文の実行計画の取得
INSERT	表に対する新しい行の追加
LOCK TABLE	表またはビューをロックすることによる、他のユーザーからのアクセスを制限
SELECT	1 つ以上の表からの行と列のデータの選択
UPDATE	表内のデータの変更

EXPLAIN PLAN コマンド以外の DML コマンドはすべて、PL/SQL でサポートされています。

## トランザクション制御コマンド

トランザクション制御コマンドは、DML コマンドによる変更を管理します。

表 4-3 トランザクション制御コマンド

コマンド	用途
COMMIT	現行のトランザクション開始後に出された文によって行われた変更の確定
ROLLBACK	トランザクション開始後またはセーブポイント後に行われたすべての変更の取消し
SAVEPOINT	ロールバックポイントの設定
SET TRANSACTION	現行トランザクションの特性の設定

COMMIT コマンドと ROLLBACK コマンドの一部を除き、トランザクション制御コマンドはすべて、PL/SQL でサポートされています。制約事項の詳細は、COMMIT (4-182 ページ) および ROLLBACK (4-481 ページ) を参照してください。

## セッション制御コマンド

セッション制御コマンドは、ユーザー・セッションの特性を動的に管理します。これらのコマンドは、現行トランザクションを暗黙的にコミットしません。

セッション制御コマンドは、PL/SQL ではサポートされていません。

表 4-4 セッション制御コマンド

コマンド	用途
ALTER SESSION	SQL トレース機能の使用可能と使用禁止の切替え グローバル・ネーム変換の使用可能と使用禁止の切替え セッションの NLS パラメータ値の変更 パラレル・サーバーの場合に、別のインスタンスに接続されたセッションの場合と同様に、データベース・ファイルへのアクセスが必要であることを指示 データベース・リンクのクローズ インダウト分散トランザクションを強制処理するためのアドバイスのリモート・データベースへの送信 プロシージャやストアド・プロシージャでの COMMIT 文や ROLLBACK 文の発行の許可または禁止 コストベースの最適化方法の目標の変更
SET ROLE	現行セッションに対するロールの使用可能と使用禁止の切替え

システム制御コマンド

システム制御コマンドは Oracle インスタンスの特性を動的に管理します。このコマンドは、現行トランザクションを暗黙的にコミットしません。

ALTER SYSTEM コマンドは、PL/SQL ではサポートされていません。

表 4-5 システム制御コマンド

コマンド	用途
ALTER SYSTEM	特別な関数を実行することによる Oracle インスタンスの変更

### 埋込み SQL コマンド

埋込み SQL コマンドによって、DDL 文および DML 文、トランザクション制御文をプロシージャ型言語プログラム内で使用できます。埋込み SQL は、Oracle プリコンパイラによってサポートされます。埋込み SQL については、次のマニュアルを参照してください。

- 『Pro\*COBOL プリコンパイラ・プログラマーズ・ガイド』
- 『Pro\*C/C++ プリコンパイラ・プログラマーズ・ガイド』
- 『SQL\*Module for Ada Programmer's Guide』



# ALTER CLUSTER

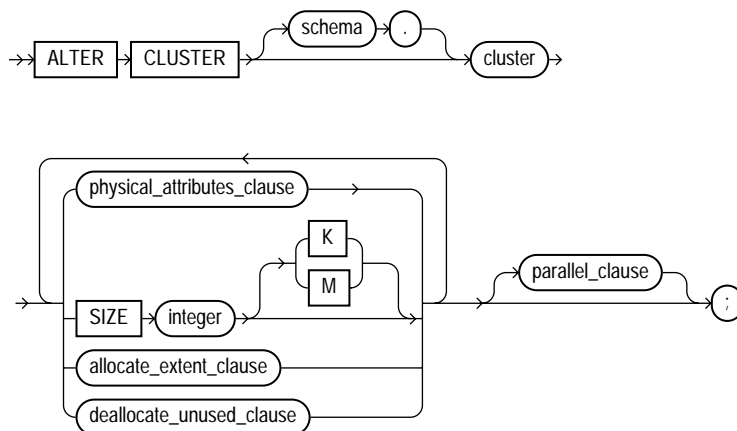
## 用途

クラスタの記憶特性と並列特性を再定義します。「使用上の注意」（4-13 ページ）も参照してください。

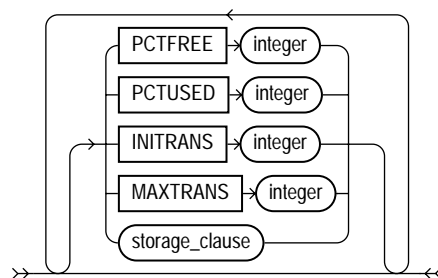
## 前提条件

クラスタが自スキーマ（このコマンドを出すユーザーのスキーマ）内にあることが必要です。自スキーマ内にはない場合は、ALTER ANY CLUSTER システム権限が必要です。

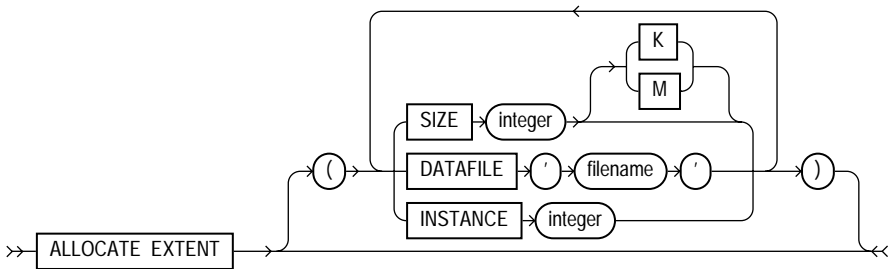
## 構文



physical\_attributes\_clause ::=



allocate\_extent\_clause ::=



deallocate\_unused\_clause: DEALLOCATE UNUSED 句（4-368 ページ）を参照。  
parallel\_clause: PARALLEL 句（4-462 ページ）を参照。

キーワードとパラメータ

<i>schema</i>	クラスタが定義されているスキーマです。 <i>schema</i> を指定しないと、Oracle はそのクラスタが自スキーマに存在するとみなします。
<i>cluster</i>	変更するクラスタの名前です。
<i>physical_attributes_clause</i>	クラスタの PCTUSED パラメータおよび PCTFREE パラメータ、INITTRANS パラメータ、MAXTRANS パラメータの値を変更します。CREATE CLUSTER（4-202 ページ）を参照してください。
<i>storage_clause</i>	クラスタの記憶特性を変更します。STORAGE 句（4-518 ページ）を参照してください。
SIZE	クラスタに割り当てられたデータ・ブロック中に格納されるクラスタ・キーの数を決定します。ハッシュ・クラスタではなく、索引クラスタの SIZE パラメータだけを変更できます。SIZE パラメータの説明は、CREATE CLUSTER（4-202 ページ）を参照してください。
<i>allocate_extent_clause</i>	クラスタの新しいエクステントを明示的に割り当てます。
SIZE	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 を使用してエクステントのサイズを KB または MB 単位で指定します。このパラメータの値を指定しないと、サイズはクラスタの STORAGE パラメータの値に基づいて決定されます。
DATAFILE	クラスタの表領域内で、新しいエクステントを割り当てるデータ・ファイルを 1 つ指定します。このパラメータを指定しないと、データ・ファイルは Oracle によって選択されます。

	<p><b>INSTANCE</b> 指定したインスタンスだけが新しいエクステントを使用できるようにします。インスタンスは初期化パラメータ <b>INSTANCE_NUMBER</b> の値で識別されます。このパラメータを指定しないと、すべてのインスタンスが新しいエクステントを使用できます。このパラメータは、<b>Parallel Server</b> オプションを指定して <b>Oracle</b> を使用している場合にだけ使用してください。</p> <p>この句でエクステントを明示的に割り当てると、そのクラスタの記憶領域パラメータは算出されず、次に割り当てられるエクステントの新たなサイズも設定されません。ハッシュ・クラスタではなく、索引クラスタだけに新しいエクステントを割り当てることができます。</p>
<i>deallocate_unused_clause</i>	<p>クラスタの終わりの未使用領域の割当てを明示的に解除し、解放された領域を他のセグメントで利用できるようにします。ただし解放されるのは、上限基準点を超える未使用領域だけです。<b>KEEP</b> を指定しないと、未使用領域がすべて解放されます。構文および詳細は、<b>DEALLOCATE UNUSED</b> 句（4-368 ページ）を参照してください。</p> <p><b>KEEP</b> 割当てを解除した後にクラスタに残す、上限基準点を超えるバイト数を指定します。残りのエクステント数が <b>MINEXTENTS</b> より小さいときは、<b>MINEXTENTS</b> は現行のエクステント数に設定されます。初期エクステントが <b>INITIAL</b> より小さくなると、<b>INITIAL</b> は初期エクステントの現行の値に設定されます。</p>
<i>parallel_clause</i>	<p>クラスタ作成時に並列度を指定します。また、作成されたクラスタについて問合せに関するデフォルトの並列度を指定します。構文および詳細は、<b>PARALLEL</b> 句（4-462 ページ）を参照してください。</p>

## 使用上の注意

**ALTER CLUSTER** コマンドを使用すると、次の処理を行うことができます。

- クラスタ内のデータ・ブロックの **MAXTRANS** パラメータの値を変更する。
- クラスタ内のデータ・ブロックの **SIZE** および **PCTUSED**、**PCTFREE**、**INITRANS** の各パラメータの値を変更する。
- **STORAGE** パラメータ **NEXT** および **PCTINCREASE**、**MAXEXTENTS** によって将来の記憶特性を変更する。
- エクステントを明示的に割り当てる。
- 未使用のエクステントから領域の割当てを明示的に解除する。

**ALTER CLUSTER** コマンドを使用して次の処理を行うことはできません。

- クラスタ・キー内の列の数と名前を変更する。
- **STORAGE** パラメータ **INITIAL** と **MINEXTENTS** の値を変更する。
- クラスタが格納されている表領域を変更する。

- クラスタから表を削除する。(DROP CLUSTER (4-381 ページ) および DROP TABLE (4-402 ページ) 参照)

**例 1** 次の文は、スキーマ SCOTT の CUSTOMER クラスタを変更します。

```
ALTER CLUSTER scott.customer
    SIZE 512
    STORAGE (MAXEXTENTS 25);
```

この結果、各クラスタ・キー値に 512 バイトが割り当てられます。データ・ブロックのサイズを 2KB と想定すると、このクラスタ内の将来のデータ・ブロックには、1 ブロックあたり 4 つのクラスタ・キー (2KB を 512 バイトで割った結果) が含まれます。

このクラスタは最大 25 のエクステンツを持つことができます。

**例 2** 次の文は、CUSTOMER クラスタから未使用領域の割当てを解除し、今後使用できるように 30KB の未使用領域を保持します。

```
ALTER CLUSTER scott.customer
    DEALLOCATE UNUSED KEEP 30 K;
```

## 関連項目

CREATE CLUSTER (4-202 ページ)

CREATE TABLE (4-303 ページ)

DEALLOCATE UNUSED 句 (4-368 ページ)

DROP CLUSTER (4-381 ページ)

DROP TABLE (4-402 ページ)

STORAGE 句 (4-518 ページ)

PARALLEL 句 (4-462 ページ)

## ALTER DATABASE

### 用途

次のいずれかの方法で既存のデータベースを変更します。

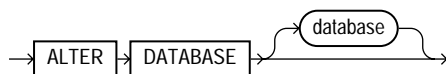
- データベース、データベースのコピー、またはスタンバイ・データベースをマウントする。
- Oracle8 への移行時に、Oracle7 のデータ・ディクショナリを変換する。
- データベースをオープンする。
- REDO ログ・ファイル・グループに対して ARCHIVELOG モードまたは NOARCHIVELOG モードを選択する。
- メディア回復を行う。
- REDO ログ・ファイル・グループまたは REDO ログ・ファイル・グループ内のメンバーを追加または削除する。
- オンライン REDO ログ・ファイルを消去および初期化する。
- REDO ログ・ファイル・メンバーまたはデータ・ファイルを改名する。
- 現行の制御ファイルのバックアップをとる。
- データベースのトレース・ファイルに SQL コマンドのバックアップをとる（データベースの再作成に使用）。
- データ・ファイルをオンラインまたはオフラインにする。
- REDO ログ・ファイル・グループのスレッドを使用可能または使用禁止にする。
- データベースのグローバル名を変更する。
- 旧リリースの Oracle へのダウングレードのための準備をする。
- 1 つ以上のデータ・ファイルをサイズ変更する。
- 回復のために、古いデータ・ファイルにかわる新しいデータ・ファイルを作成する。
- データ・ファイルのサイズの自動拡張の使用可能と使用禁止を切り替える。

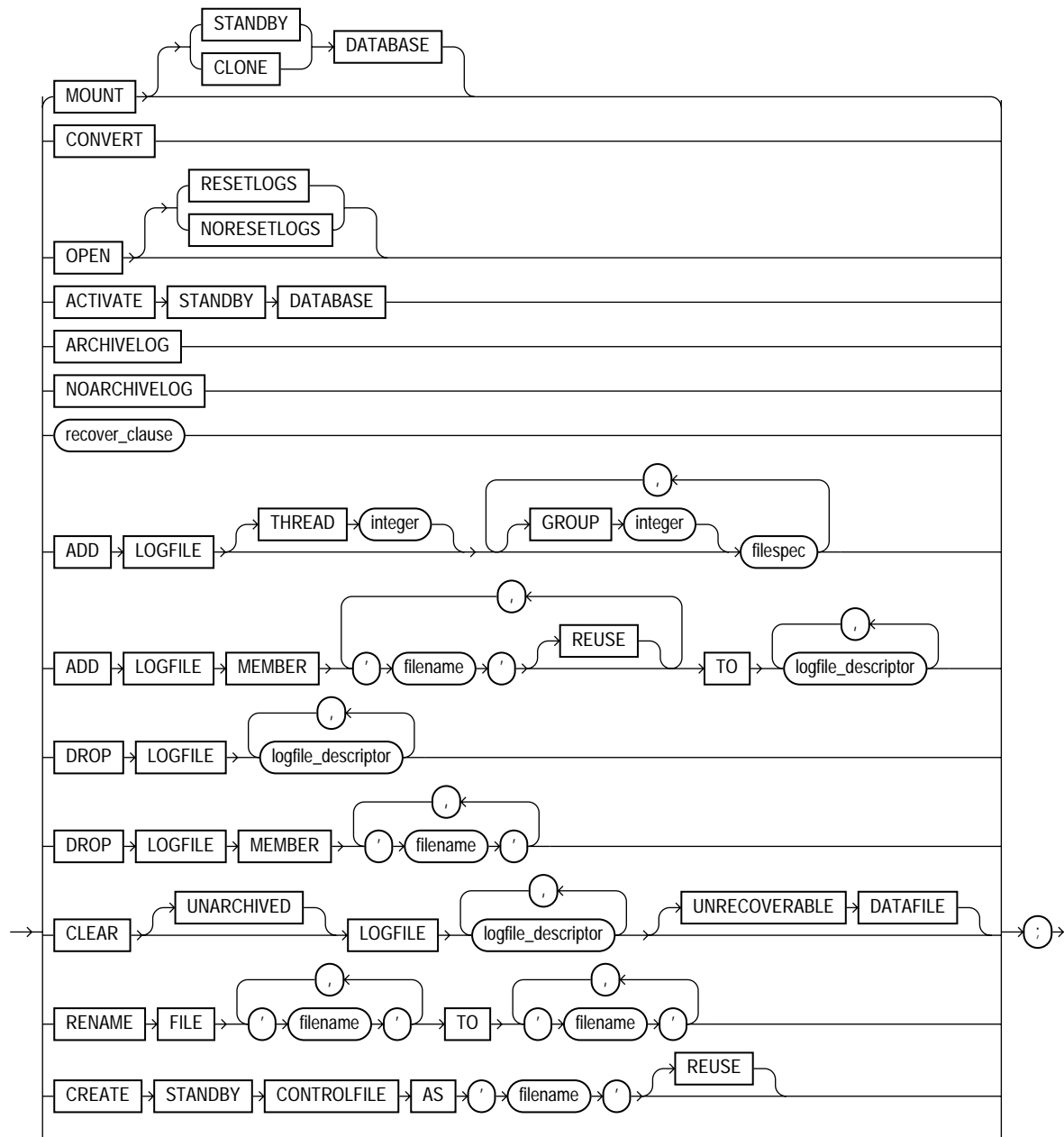
これらの用途の詳細は、「例」（4-45 ページ）も参照してください。

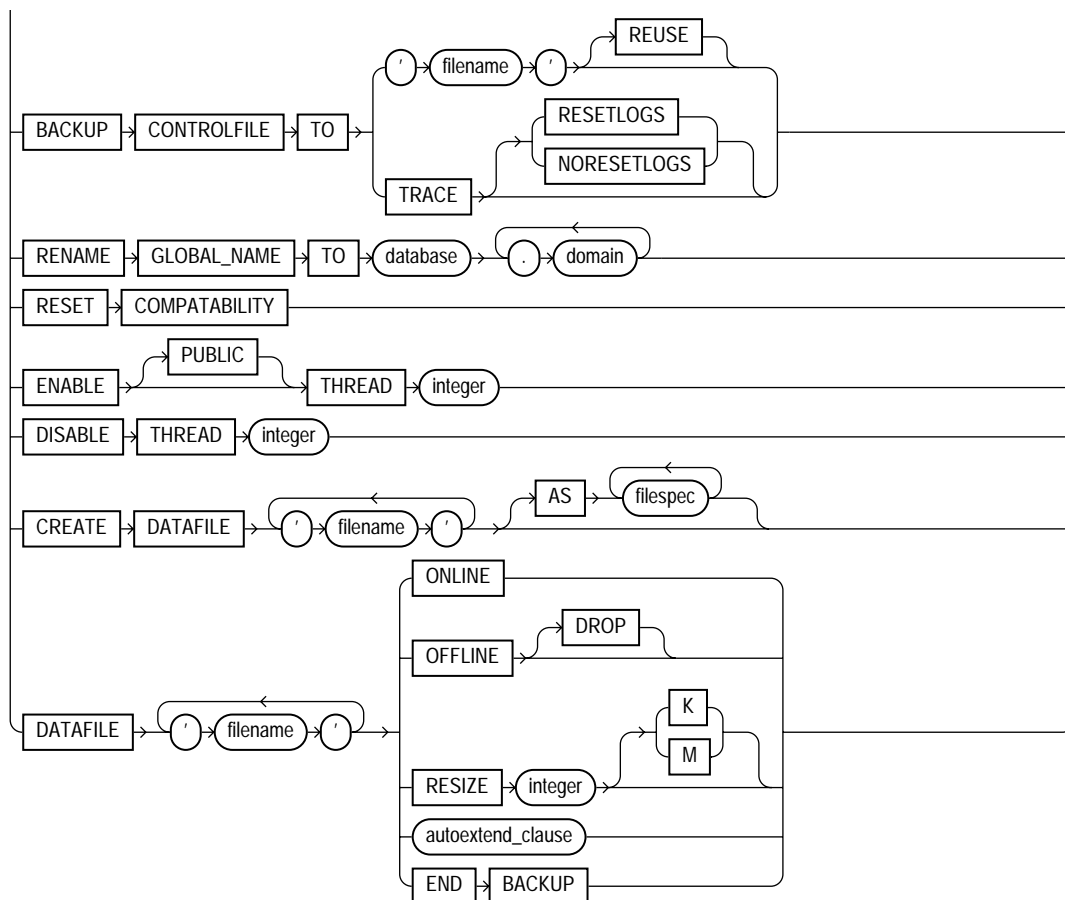
### 前提条件

ALTER DATABASE システム権限が必要です。

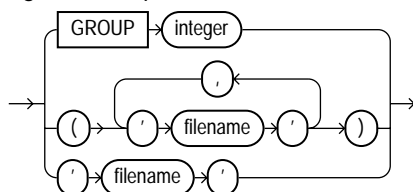
### 構文



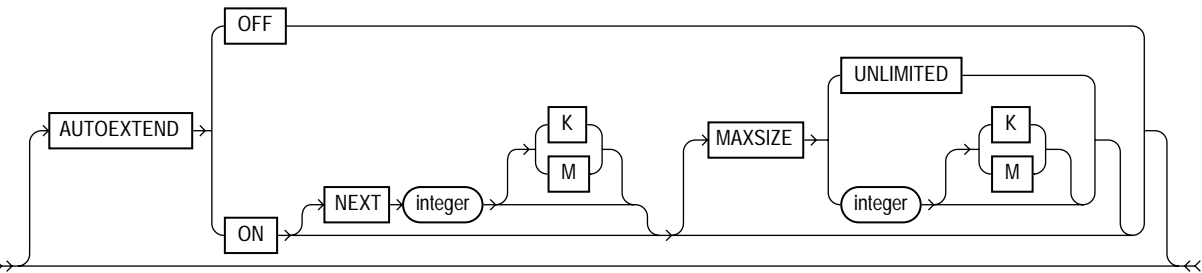




logfile\_descriptor::=



autoextend\_clause ::=



recover\_clause: RECOVER 句（4-466 ページ）を参照。

キーワードとパラメータ

**database** 変更するデータベースを指定します。データベース名には ASCII 文字だけが使用できます。データベースを指定しないと、初期化パラメータ DB\_NAME に指定されているデータベースが変更されます。なお、データベースの制御ファイルが初期化パラメータ CONTROL\_FILES に指定されている場合にだけ、そのデータベースを変更できます。データベース識別子は Net8 のデータベース指定とは関係ありません。

自インスタンスでデータベースがマウントされていない場合にだけ、次のオプションを使用できます。

- MOUNT** データベースをマウントします。
- STANDBY DATABASE** スタンバイ・データベースをマウントします。詳細は、『Oracle8 Server 管理者ガイド』を参照してください。
- CLONE DATABASE** データベースのコピーをマウントします。詳細は、『Oracle8 Server バックアップおよびリカバリ』を参照してください。
- CONVERT** Oracle7 のデータ・ディクショナリを変換します。このオプションを使用すると、Oracle7 のデータ・ディクショナリは Oracle データベース内に存在しなくなります。このため、このオプションは、Oracle8 への移行時にだけ使用してください。このオプションの使用の詳細は、『Oracle8 Server 移行ガイド』を参照してください。
- OPEN** 通常の用途のためにデータベースをオープンし、使用可能な状態にします。データベースをオープンするにはマウントしておく必要があります。アクティブになっていないスタンバイ・データベースはオープンできません。



	<p><b>RESETLOGS</b>      現行のログ順序番号を1にリセットし、回復時に適用されなかった REDO 情報が今後適用されないように破棄します。これにより、REDO ログに記録されていてもデータベースにはない変更はすべて廃棄されます。</p> <p><b>RECOVER UNTIL</b>(RECOVER 句 (4-466 ページ) 参照) による不完全回復やバックアップ用制御ファイルによるメディア回復を行った後は、このオプションを使用してデータベースをオープンしてください。このオプションを使用してデータベースをオープンした場合は、データベース全体のバックアップをとってください。</p>
	<p><b>NORESETLOGS</b>      ログ順序番号と REDO ログ・ファイルを現在の状態のままにします。</p>
	<p>これらのオプションは、バックアップ用制御ファイルによる完全メディア回復または不完全メディア回復を行った後でだけ指定できます。それ以外の場合は、NORESETLOGS が自動的に適用されます。</p>
<p><b>ACTIVATE STANDBY DATABASE</b></p>	<p>スタンバイ・データベースの状態をアクティブ・データベースに変更します。詳細は、『Oracle8 Server 管理者ガイド』を参照してください。</p>
	<p>次のオプションは、自インスタンスでデータベースがパラレル・サーバー使用禁止モードでマウントされていて、なおかつオープンされていない場合にだけ使用できます。</p>
<p><b>ARCHIVELOG</b></p>	<p>REDO ログ・ファイル・グループに対して ARCHIVELOG モードを設定します。このモードでは、REDO ログ・ファイル・グループは、内容がアーカイブされてから再使用されます。このオプションを指定するとメディア回復が可能となります。このオプションは、エラーなしの通常終了または即時終了でインスタンスを停止し再起動した後に、データベースをパラレル・サーバー使用禁止モードでマウントした場合にだけ使用できます。</p>
<p><b>NOARCHIVELOG</b></p>	<p>REDO ログ・ファイル・グループを NOARCHIVELOG モードに設定します。このモードでは、REDO ログ・ファイル・グループは、内容がアーカイブされずに再使用されます。このモードでは、メディア障害後の回復準備を行いません。</p>
	<p>自インスタンスでデータベースがマウント済み、オープン状態、クローズ状態のいずれの場合でも、関連ファイルが使用中でなければ、次のオプションはどれでも使用できます。</p>
<p><i>recover_clause</i></p>	<p>メディア回復を行います。構文および詳細は、RECOVER 句 (4-466 ページ) を参照してください。データベースがクローズされた状態である場合に限り、データベース全体を回復できます。回復の対象となる表領域やデータ・ファイルがオフラインであれば、データベースがオープン状態でもクローズ状態でも、表領域やデータ・ファイルを回復できます。ただし、マルチスレッド・サーバー・アーキテクチャで Oracle に接続している場合は、メディア回復を実行できません。また、Server Manager の回復ダイアログ・ボックスを使ってもメディア回復を実行できます。</p>
<p><b>ADD LOGFILE</b></p>	<p>指定したスレッドに1つ以上の REDO ログ・ファイル・グループを追加して、そのスレッドに割り当てられているインスタンスがそのグループを使用できるようにします。</p>
<p><b>THREAD</b></p>	<p>パラレル・モードの Parallel Server オプションで Oracle を使用している場合にだけ必要です。THREAD パラメータを指定しないと、REDO ログ・ファイル・グループは自インスタンスに割り当てられたスレッドに追加されます。</p>

	<p><b>GROUP</b></p> <p><i>filespec</i></p>	<p>全スレッドのすべてのグループの間で REDO ログ・ファイル・グループが一意に識別されます。この値は 1 から MAXLOGFILES 値までの範囲に設定できます。同じ GROUP 値を持つ REDO ログ・ファイル・グループは重複して追加できません。このパラメータを指定しないと、この値は自動的に生成されます。REDO ログ・ファイル・グループの GROUP 値は、動的パフォーマンスビュー V\$LOG で調べることができます。</p> <p><i>filespec</i> には、1 つ以上のメンバー、すなわちコピーからなる REDO ログ・ファイル・グループを指定します。<i>filespec</i> の構文の説明は、「Filespec」(4-428 ページ) を参照してください。</p>
<p><b>ADD LOGFILE MEMBER</b></p>		<p>新しいメンバーを既存の REDO ログ・ファイル・グループに追加します。新しいメンバーをそれぞれ 'filename' に指定します。すでにファイルが存在する場合、追加するメンバーはグループ内の他のメンバーと同じサイズである必要があり、REUSE オプションを指定しなければなりません。ファイルが存在していない場合は、適切なサイズのファイルが作成されます。メディア障害によってグループのすべてのメンバーを失った場合には、そのグループにはメンバーを追加できません。</p> <p>次のいずれかの方法で既存の REDO ログ・ファイル・グループを指定できます。</p> <p><b>GROUP</b> パラメータ REDO ログ・ファイル・グループを識別する GROUP パラメータの値を指定します。</p> <p>ファイル名のリスト REDO ログ・ファイル・グループのすべてのメンバーを指定します。使用しているオペレーティング・システムの表記法に従って、ファイル名を完全に修飾して指定する必要があります。</p>
<p><b>DROP LOGFILE</b></p>		<p>REDO ログ・ファイル・グループのメンバーをすべて削除します。REDO ログ・ファイル・グループは ADD LOGFILE MEMBER 句の場合と同じ方法で指定できます。アーカイブが必要な REDO ログ・ファイル・グループや、または現在アクティブな REDO ログ・ファイル・グループは、削除できません。また、REDO ログ・ファイル・グループを削除すると、その REDO スレッドの REDO ログ・ファイル・グループが 2 つ未満になる場合にも、削除はできません。</p>
<p><b>DROP LOGFILE MEMBER</b></p>		<p>1 つ以上の REDO ログ・ファイル・メンバーを削除します。各 'filename' には、使用しているオペレーティング・システムのファイル名の表記法に従って、メンバーを完全に修飾して指定する必要があります。</p> <p>この句では、有効なデータを含んでいる REDO ログ・ファイル・グループのすべてのメンバーは削除できません。このような操作には、DROP LOGFILE 句を使用してください。</p>
<p><b>CLEAR LOGFILE</b></p>		<p>オンライン REDO ログを初期化し直します。REDO ログをアーカイブしないオプションもあります。CLEAR LOGFILE は、REDO ログの追加および削除と似ていますが、スレッドに 2 つしかログがない場合や、クローズ状態のスレッドの現行の REDO ログ・ファイルに対しても発行できます。</p> <p><b>UNARCHIVED</b></p> <p>アーカイブされていない REDO ログを再利用する場合は、UNARCHIVED を指定する必要があります。</p> <p><b>警告：</b>UNARCHIVED を指定すると、回復のために REDO ログが必要な場合に、バックアップが使用できなくなります。</p>

CLEAR LOGFILE を使って、メディア回復に必要なログは消去できません。データベースのチェックポイント後の REDO を含むログを消去する必要がある場合は、不完全メディア回復を最初に行う必要があります。オープンしているスレッドの現行の REDO ログは消去できます。クローズしているスレッドの現行のログは、そのスレッド内でログを切り替えれば消去できます。

CLEAR LOGFILE コマンドが、システム障害またはインスタンス障害による割込みを受けると、データベースがハングする場合があります。ハングした場合は、データベースを再起動してからコマンドを再発行してください。ログ・グループのあるメンバーにアクセスしようとした時に I/O エラーによる障害が発生した場合は、そのメンバーを削除して他のメンバーを追加できます。

UNRECOVERABLE DATAFILE データベースに（削除用ではない）オフライン状態のデータ・ファイルがあり、そのデータ・ファイルをオンライン状態に戻すために、アーカイブされていない、消去対象のログが必要である場合は、UNRECOVERABLE DATAFILE を指定してください。この場合、CLEAR LOGFILE コマンドが終了してから、そのデータ・ファイルと表領域全体を削除する必要があります。

RENAME FILE データ・ファイルまたは REDO ログ・ファイル・メンバーを改名します。この句を指定すると、制御ファイルの中のファイル名だけが変更され、オペレーティング・システム上の実際のファイル名は変更されません。各ファイル名を指定するときは、オペレーティング・システムのファイル名の表記法に従ってください。

CREATE STANDBY CONTROLFILE スタンバイ・データベースを管理するための制御ファイルを作成します。詳細は、『Oracle8 Server 管理者ガイド』を参照してください。

BACKUP CONTROLFILE 現行の制御ファイルのバックアップをとります。

TO 'filename' 制御ファイルのバックアップ先ファイルを指定します。使用しているオペレーティング・システムの表記法に従って、ファイル名を完全に修飾して指定する必要があります。その名前のバックアップ・ファイルがすでに存在している場合は、REUSE オプションを指定してください。

TO TRACE 制御ファイルの物理バックアップをとるのではなく、データベースのトレース・ファイルに SQL 文を書き込みます。SQL コマンドを使うと、データベースの起動、制御ファイルの再作成、データベースの回復やオープンなどの操作を、作成した制御ファイルに基づいて正しく実行できます。

トレース・ファイルからスクリプト・ファイルにコマンドをコピーできます。必要に応じてコマンドを編集できます。そのスクリプトを利用して制御ファイルのコピーがすべて失われた場合にデータベースを回復したり、制御ファイルのサイズを変更したりします。

RESETLOGS データベースの起動用としてトレース・ファイルに書き込まれた SQL 文が ALTER DATABASE OPEN RESETLOGS であることを指定します。

NORESETLOGS	データベースの起動用としてトレース・ファイルに書き込まれた SQL 文が ALTER DATABASE OPEN NORESETLOGS であることを指定します。
RENAME GLOBAL_NAME	データベースのグローバル名を変更します。 <i>database</i> にはデータベースの新しい名前を 8 バイト以内の長さで指定します。オプションのドメイン <i>domain</i> には、ネットワーク階層におけるデータベースの有効な位置を指定します。  <b>注意：</b> データベース名を変更しても、リモート・データベース内の既存のデータベース・リンク、シノニム、ストアド・プロシージャ、ストアド・ファンクションからのユーザーのデータベースに対するグローバル参照は変更されません。これらの参照を変更するのは、リモート・データベース管理者の責任です。  グローバル名の詳細は、『Oracle8 Server 分散システム』を参照してください。
RESET COMPATIBILITY	次に再起動するときに、Oracle の旧バージョンにリセットする必要があるデータベースにマークを付けます。  <b>注意：</b> RESET COMPATIBILITY は、下位互換性に影響を及ぼす Oracle の機能を使用禁止にしている場合にだけ有効です。Oracle の旧バージョンへのダウングレードの詳細は、『Oracle8 Server 移行ガイド』を参照してください。
自インスタンスでデータベースがオープンしている場合にだけ、次のオプションを使用できます。	
ENABLE THREAD	パラレル・サーバーにおいて、REDO ログ・ファイル・グループの指定スレッドを使用可能にします。使用可能にできるスレッドは、2 つ以上の REDO ログ・ファイルを持っているスレッドだけです。  <b>PUBLIC</b> 初期化パラメータ THREAD によって特定のスレッドを明示的に要求していないインスタンスに対して、使用可能になったスレッドを使用できるようにします。 <b>PUBLIC</b> オプションを指定しないと、初期化パラメータ THREAD を使用して明示的に要求するインスタンスだけがこのスレッドを使用できます。
DISABLE THREAD	指定したスレッドをすべてのインスタンスで使用禁止にします。指定したスレッドを使用しているインスタンスがデータベースをマウント済みの場合には、そのスレッドは使用禁止にできません。
自インスタンスでデータベースがマウントまたはオープン、クローズされている場合に、関連ファイルが使用中でなければ、次のいずれのオプションも使用できます。	
CREATE DATAFILE	元のデータ・ファイルのかわりに新しい空のデータ・ファイルを作成します。このオプションを使用して、バックアップのない失われたデータ・ファイルを再作成できます。 <i>'filename'</i> にはデータベースの一部であるファイル、あるいは以前一部であったファイルを指定してください。 <i>filespec</i> には新しいデータ・ファイルの名前とサイズを指定します。 <b>AS</b> 句を指定しないと、Oracle によって、 <i>'filename'</i> に指定したファイルと同じ名前とサイズのファイルが新たに作成されます。  回復時には、元のデータ・ファイルの作成後に書き込まれたアーカイブ REDO ログを、失われたデータ・ファイルにかわる新しい空のデータ・ファイルに適用しなければなりません。  新しいファイルは元のファイルの作成時と同じ状態で作成されます。新しいファイルを元のファイルが失われた時点の状態に戻すには、メディア回復を行ってください。

---

	SYSTEM 表領域の最初のデータ・ファイルに基づく新しいファイルを作成できません。
DATAFILE	データベース・ファイルを次のように変更します。
ONLINE	データ・ファイルをオンラインにします。
OFFLINE	データ・ファイルをオフラインにします。データベースがオープンしている場合、データ・ファイルをオンラインに戻す前に、データ・ファイルのメディア回復を行う必要があります。これは、データ・ファイルがオフラインになる前にチェックポイントが実行されないためです。
DROP	データベースが NOARCHIVELOG モードの場合に、データ・ファイルをオフラインにします。
RESIZE	データ・ファイルのサイズを指定した絶対サイズ（単位はバイト）に変更します。K または M を使用して KB または MB 単位でもサイズを指定できます。デフォルト値はないので、必ずサイズを指定してください。
<i>autoextend_clause</i>	データ・ファイルの自動拡張の使用可能と使用禁止を切り替えます。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	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。ALTER DATABASE AUTOEXTEND コマンドを次に使用する場合は、NEXT および MAXSIZE に再度値を指定する必要があります。
ON	自動拡張を使用可能にします。
NEXT	エクステンツがさらに必要になったときに、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のサイズ（バイト単位）を指定します。K または M を使用して KB または MB 単位でもサイズを指定できます。デフォルト値は 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わ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します。
END BACKUP	オンライン表領域のバックアップ中に、システム障害またはインスタンス障害、SHUTDOWN ABORT による割込みが起きた後には、データベース起動時のメディア回復を回避します。
警告：影響を受けたファイルをバックアップから復元した場合は、ALTER TABLESPACE ... END BACK UP を使わないでください。メディア回復については、Oracle8 Server バックアップおよびリカバリおよび『Oracle8 Server 管理者ガイド』に詳しい説明があります。	

---

## 例

データベース管理のための ALTER DATABASE コマンドの使用の詳細は、『Oracle8 Server 管理者ガイド』を参照してください。

**例 1** 次の文は、2つのメンバーを持つ REDO ログ・ファイル・グループを追加し、GROUP パラメータの値に 3 を指定してこのグループを識別します。

```
ALTER DATABASE stocks
  ADD LOGFILE GROUP 3
    ('diska:log3.log' ,
     'diskb:log3.log') SIZE 50K;
```

**例 2** 次の文は、例 1 で追加した REDO ログ・ファイル・グループにメンバーを 1 つ追加します。

```
ALTER DATABASE stocks
  ADD LOGFILE MEMBER 'diskc:log3.log'
  TO GROUP 3;
```

**例 3** 次の文は、例 2 で追加した REDO ログ・ファイル・メンバーを削除します。

```
ALTER DATABASE stocks
  DROP LOGFILE MEMBER 'diskc:log3.log';
```

**例 4** 次の文は、REDO ログ・ファイル・メンバーを改名します。

```
ALTER DATABASE stocks
  RENAME FILE 'diskb:log3.log' TO 'diskd:log3.log';
```

上記の例では、REDO ログ・グループ・メンバーのファイルが別のファイル名に変更されただけです。つまり、ファイル名が実際に 'DISKB:LOG3.LOG' から 'DISKD:LOG3.LOG' へ変わるわけではありません。実際のファイル名を変更する場合には、オペレーティング・システムから操作する必要があります。

**例 5** 次の文は、REDO ログ・ファイル・グループ 3 のすべてのメンバーを削除します。

```
ALTER DATABASE stocks DROP LOGFILE GROUP 3;
```

**例 6** 次の文は、3つのメンバーを持つ REDO ログ・ファイル・グループをスレッド 5 に追加して、このグループに GROUP パラメータ値 4 を代入します。

```
ALTER DATABASE stocks
  ADD LOGFILE THREAD 5 GROUP 4
    ('diska:log4.log' ,
     'diskb:log4:log' ,
     'diskc:log4.log' );
```

**例 7** 次の文は、パラレル・サーバーの場合に、スレッド 5 を使用禁止にします。

```
ALTER DATABASE stocks
  DISABLE THREAD 5;
```

**例 8** 次の文は、パラレル・サーバーの場合に、スレッド 5 を使用可能にして、特定のスレッドを明示的に要求しない任意の Oracle インスタンスがこのスレッドを使用できるようにします。

```
ALTER DATABASE stocks
    ENABLE PUBLIC THREAD 5;
```

**例 9** 次の文は、データ・ファイル 'DISK2:DB1.DAT' に基づいてデータ・ファイル 'DISK1:DB1.DAT' を作成します。

```
ALTER DATABASE
    CREATE DATAFILE 'disk1:db1.dat' AS 'disk2:db1.dat';
```

**例 11** 次の文は、データベースのグローバル名を変更し、データベース名とドメインの両方を指定します。

```
ALTER DATABASE
    RENAME GLOBAL_NAME TO sales.australia.acme.com;
```

**例 12** 次の文は、データ・ファイル 'DISK1:DB1.DAT' のサイズを変更します。

```
ALTER DATABASE
    DATAFILE 'disk1:db1.dat' RESIZE 10 M;
```

メディア回復の例は、『Oracle8 Server 管理者ガイド』および『Oracle8 Server バックアップおよびリカバリ』を参照してください。

**例 13** 次の文は、ログ・ファイルを消去します。

```
ALTER DATABASE
    CLEAR LOGFILE 'disk3:log.dbf';
```

## 関連項目

CREATE DATABASE (4-214 ページ)

RECOVER 句 (4-466 ページ)

「Filespec」 (4-428 ページ)

# ALTER FUNCTION

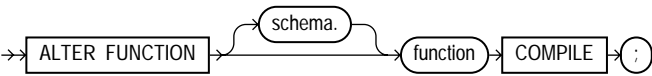
## 用途

スタンドアロンのストアド・ファンクションを再コンパイルします。「使用上の注意」(4-26 ページ) も参照してください。

## 前提条件

関数が自スキーマ内にあることが必要です。自スキーマ内にない場合は、ALTER ANY PROCEDURE システム権限が必要です。

## 構文



## キーワードとパラメータ

---

<i>schema</i>	再コンパイルする関数が設定されているスキーマです。 <i>schema</i> の指定を省略すると、Oracle は関数が自スキーマ内に定義されているものとみなします。
<i>function</i>	再コンパイルする関数名です。
COMPILE	指定した関数が再コンパイルされます。COMPILE キーワードは必須です。

---

## 使用上の注意

ALTER FUNCTION コマンドを使用すると無効な関数を明示的に再コンパイルできます。明示的に再コンパイルすることによって、実行時に暗黙的に再コンパイルされる必要がなくなり、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。

ALTER FUNCTION コマンドは、ALTER PROCEDURE (4-41 ページ) と似ています。関数とプロシージャの再コンパイルの詳細は、『Oracle8 Server 概要』を参照してください。

---

**注意：** このコマンドを使用して既存の関数の宣言や定義は変更できません。関数を再宣言または再定義するには、OR REPLACE オプションを指定して CREATE FUNCTION コマンドを使用します。CREATE FUNCTION (4-228 ページ) を参照してください。

---



**例** 次の文は、ユーザー MERRIWEATHER の関数 GET\_BAL を明示的に再コンパイルします。

```
ALTER FUNCTION merriweather.get_bal  
COMPILE;
```

GET\_BAL の再コンパイル時にエラーが発生しなければ、GET\_BAL は有効になります。その後、Oracle は実行時に GET\_BAL を再コンパイルせずに実行できます。GET\_BAL の再コンパイル時にコンパイル・エラーが発生した場合は、エラー・メッセージが戻り、GET\_BAL は無効のままです。

また、GET\_BAL に依存しているオブジェクトがすべて無効になります。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すると、Oracle によってそれらが実行時に暗黙的に再コンパイルされます。

## 関連項目

ALTER PROCEDURE (4-41 ページ)

CREATE FUNCTION (4-228 ページ)

## ALTER INDEX

---

### 用途

ALTER INDEX の用途は次のとおりです。

- 索引の記憶割当ての変更または索引の再構築、改名
- パーティション索引のパーティションの、物理属性またはロギング属性の変更、改名、分割、削除、使用不可のマーク付け、再構築
- 非パーティション索引の物理属性またはパラレル属性、ロギング属性の変更
- パーティション索引のデフォルトの物理属性またはパラレル属性、ロギング属性の変更
- 逆順で索引ブロックのバイトを格納するための索引の再構築
- **OBJ** ネストした表の索引の変更

これらの用途は、「例」（4-35 ページ）にいくつか例を示します。

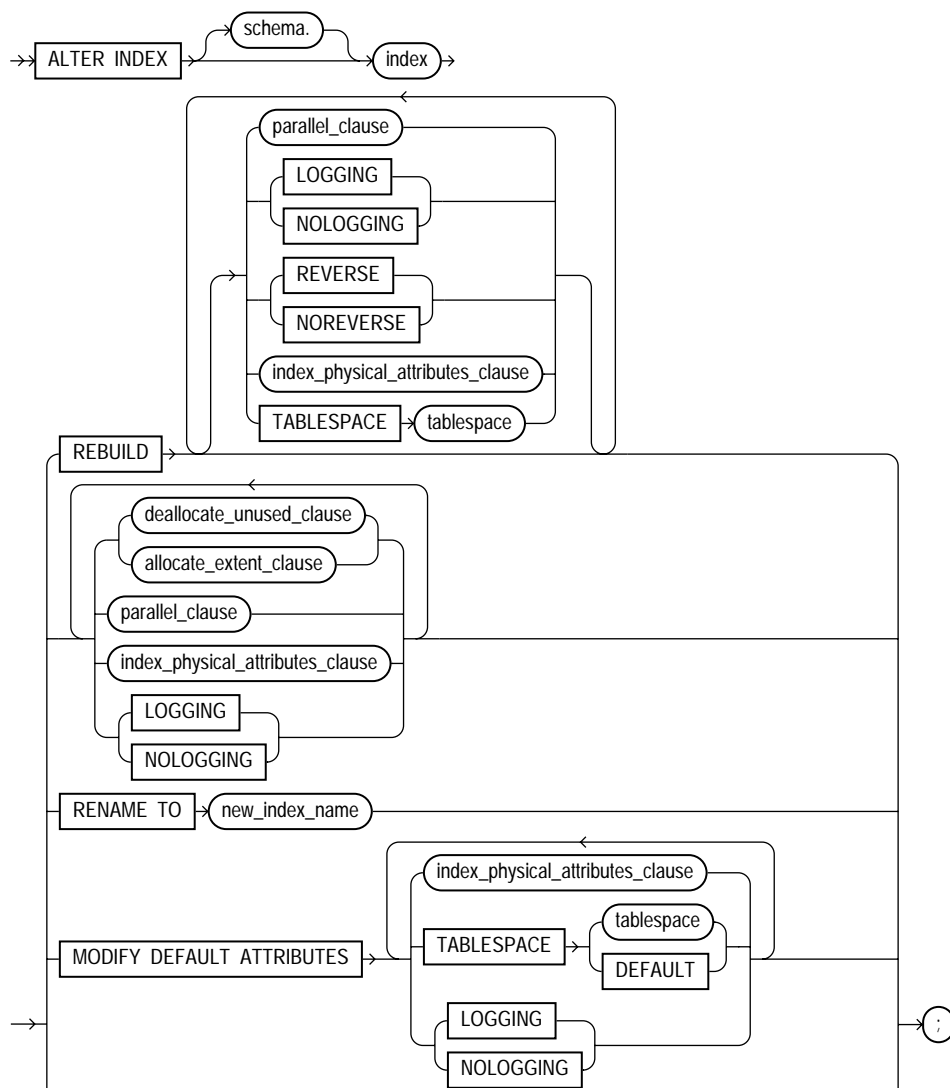
### 前提条件

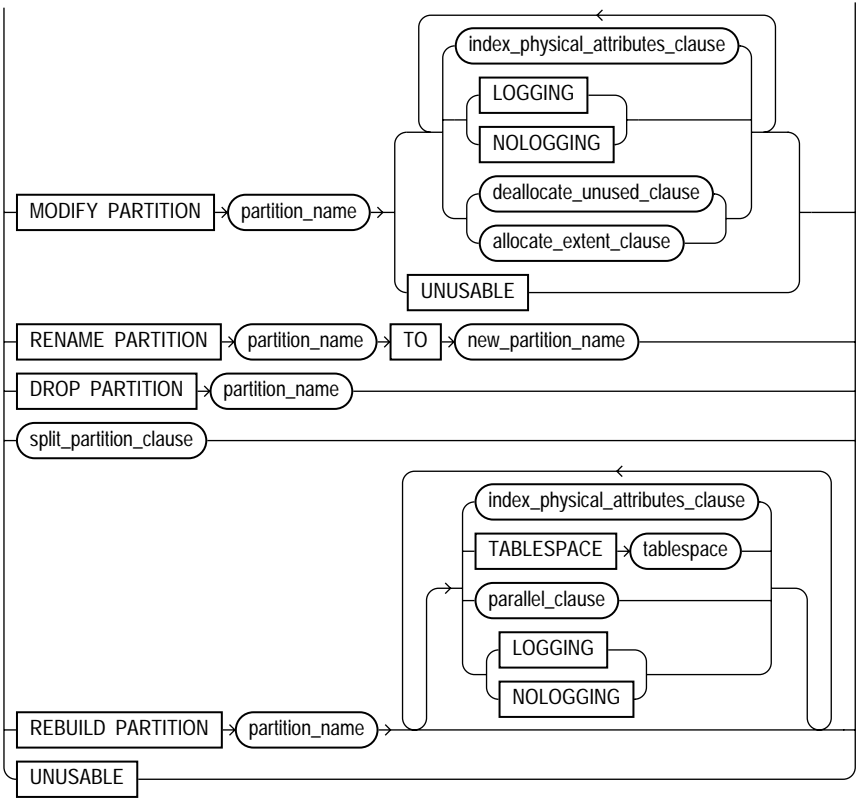
索引が自スキーマ内にあることが必要です。自スキーマ内にない場合は、ALTER ANY INDEX システム権限が必要です。

スキーマ・オブジェクト権限は、個々の索引パーティションではなく、親索引に付与されていなければなりません。次のような索引パーティション操作には、表領域割当て制限が必要です。

- 変更
- 再構築
- 分割

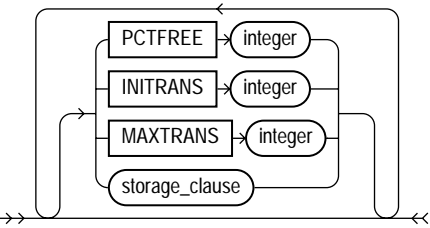
## 構文





`parallel_clause:PARALLEL` 句 (4-462 ページ) を参照。

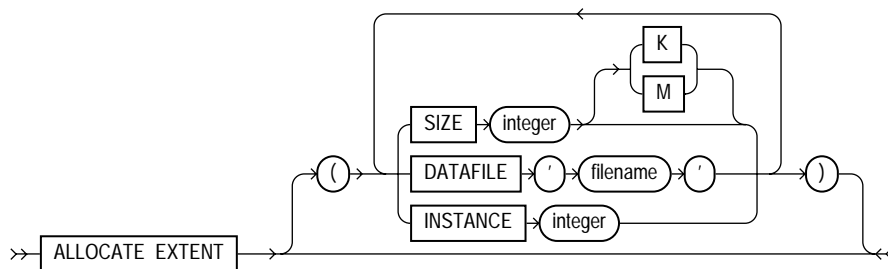
`index_physical_attributes_clause ::=`



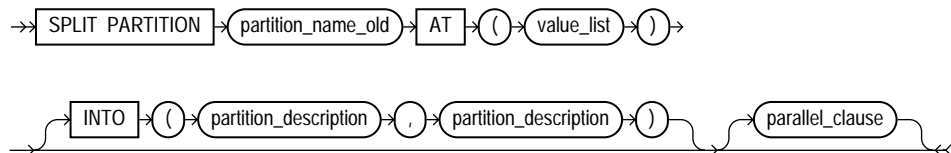
`storage_clause:STORAGE` 句 (4-518 ページ) を参照。

`deallocate_unused_clause:DEALLOCATE UNUSED` 句 (4-368 ページ) を参照。

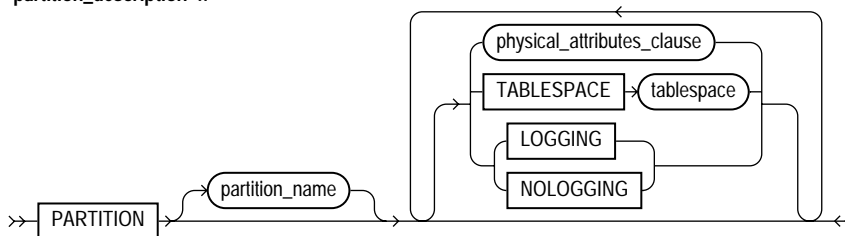
**allocate\_extent\_clause ::=**



**split\_partition\_clause ::=**



**partition\_description ::=**



## キーワードとパラメータ

<i>schema</i>	変更する索引が定義されているスキーマです。 <i>schema</i> を指定しないと、Oracle は索引が自スキーマ内に定義されているものとみなします。
<i>index</i>	変更する索引の名前です。

次の操作は、パーティション索引にしか実行できません。

- パーティションの削除
- パーティションの分割
- パーティションの改名
- パーティションの再構築
- パーティションの変更

上記のうち、パーティションの削除とパーティションの分割はグローバル索引にしか実行できません。

REBUILD

既存の索引を作成し直します。

<i>parallel_clause</i>	索引の再構築、あるいは索引や索引パーティションに対する一部の問合せを、シリアルまたはパラレルのどちらで実行するか指定します。このオプションとこの句の構文の説明は、PARALLEL 句（4-462 ページ）を参照してください。パラレル化操作の詳細は、『Oracle8 Parallel Server 概要および管理』を参照してください。
LOGGING/ NOLOGGING	ALTER INDEX...REBUILD( および ALTER INDEX...SPLIT) 操作のログを記録するかどうかを指定します。
REVERSE	索引ブロックのバイトを逆順で格納します（ただし、索引を再構築するときの ROWID は除きます）。
NOREVERSE	索引を再構築するときに順序を逆にせずに索引ブロックのバイトを格納します。NOREVERSE キーワードを指定せずに REVERSE 索引を再構築すると、再構築された索引は逆キーの索引となります。
<i>index_physical_attributes_clause</i>	<p>非パーティション索引または索引パーティション、パーティション索引のすべてのパーティションの PCTFREE および INITRANS、MAXTRANS パラメータの値、あるいはパーティション索引のパラメータのデフォルト値を変更します。それらパラメータについては、CREATE TABLE（4-303 ページ）を参照してください。</p> <p><b>注意：</b>索引全体について (ALTER INDEX) またはパーティションを変更するために (ALTER INDEX ... MODIFY PARTITION)、PCTFREE パラメータの値を変更することはできません。ALTER INDEX コマンドの他のすべての形式では、このパラメータの値を変更できます。</p>
<i>storage_clause</i>	非パーティション索引または索引パーティション、パーティション索引のすべてのパーティションの記憶領域パラメータ、あるいはパーティション索引のパラメータのデフォルト値を変更します。STORAGE 句（4-518 ページ）を参照してください。
TABLESPACE	再構築された索引または索引パーティションが格納される表領域を指定します。デフォルトの格納場所は、コマンドを発行するユーザーのデフォルトの表領域です。

<i>deallocate_unused_clause</i>	<p>索引の終わりの未使用領域の割当てを明示的に解除し、解放された領域を他のセグメントで利用できるようにします。ただし解放できるのは、上限基準点を超える未使用領域だけです。KEEP を指定しないと、未使用領域がすべて解放されます。DEALLOCATE UNUSED 句 (4-368 ページ) を参照してください。</p> <p>KEEP                      割当てを解除した後索引に残す、上限基準点を超えるバイト数を指定します。残りのエクステント数が MINEXTENTS より小さいときは、MINEXTENTS は現行のエクステント数に設定されます。初期エクステントが INITIAL より小さくなると、INITIAL は初期エクステントの現行の値に設定されます。</p>
<i>allocate_extent_clause</i>	<p>索引の新しいエクステントを明示的に割り当てます。</p> <p>SIZE                      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 を使用してエクステントのサイズを KB または MB 単位で指定してください。このパラメータを指定しないと、エクステントのサイズは索引の STORAGE パラメータの値に基づいて決定されます。</p> <p>DATAFILE                新しいエクステントを格納するデータ・ファイルを索引の表領域内で指定します。このパラメータを指定しないと、データ・ファイルは Oracle によって選択されます。</p> <p>INSTANCE                指定したインスタンスだけが新しいエクステントを使用できるようにします。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と、すべてのインスタンスが新しいエクステントを使用します。パラレル・モードの Parallel Server オプションで 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せん。</p> <p>この句を使用して明示的にエクステントを割り当てると、次に割り当てられるエクステントのサイズには、NEXT と 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。</p>
LOGGING/ NOLOGGING	<p>LOGGING/NOLOGGING は、非パーティション索引または索引パーティション、パーティション索引のすべての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 操作のログを、REDO ログ・ファイルに記録するか (LOGGING)、記録しないか (NOLOGGING) を指定します。</p> <p>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。メディア回復中に NOLOGGING が適用されると、REDO データはログに記録されないため、エクステント無効化レコードはブロック範囲を論理的に破壊されているものとしてマーク設定します。このため、消失してはならない索引の場合は、NOLOGGING モードでの操作後にバックアップをとる必要があります。</p> <p>データベースを ARCHIVELOG モードで運用する場合、LOGGING モードでの操作前に行ったバックアップからのメディア回復によって、索引が再作成されます。ただし、NOLOGGING モードでの操作前に行ったバックアップからのメディア回復では、索引は再作成されません。</p>

索引セグメントには、実表の属性と異なり、また同じ実表の他の索引セグメントとも異なるロギング属性を指定できます。

LOGGING オプションおよびパラレル DML の詳細は、『Oracle8 Server 概要』および『Oracle8 Parallel Server 概要および管理』を参照してください。

**注意：**LOGGING/NOLOGGING キーワードは、RECOVERABLE/UNRECOVERABLE オプションに代わるものです。このオプションは、非パーティション索引を変更または再構築するときには Oracle8 で有効なキーワードとして使用できますが、なるべく使用しないでください。

**RENAME TO** index を *new\_index\_name* 改名します。 *new\_index\_name* は単一の識別子で、スキーマ名は含まれません。

**MODIFY  
DEFAULT  
ATTRIBUTES** パーティション索引にだけ使用できるオプションです。このオプションを使って、パーティション索引のデフォルト属性に新しい値を指定できます。

**TABLESPACE** パーティション索引のデフォルトの表領域を格納する表領域を指定します。デフォルトの格納場所は、コマンドを発行したユーザーのデフォルトの表領域です。

**LOGGING/  
NOLOGGING** パーティション索引のデフォルトのロギング属性を指定します。

**注意：**実索引に対するいくつかの操作を 1 つの ALTER INDEX 文にまとめることはできますが (RENAME および REBUILD は除く)、パーティション操作を他のパーティション操作または実索引に対する操作と組み合わせることはできません。

**MODIFY  
PARTITION** 索引パーティション *partition\_name* の実物理属性またはロギング・オプション、記憶特性を変更します。 *partition\_name* は、変更する索引パーティションの名前です。これは *index* 内のパーティションでなければなりません。

**UNUSABLE** 索引または索引パーティションに使用不可のマークを付けます。使用不可の索引を使用可能にするには、再構築するか、または削除して再作成する必要があります。あるパーティションに使用不可のマークが付けられているときでも、その索引の他のパーティションは使用可能です。使用不可のパーティションにアクセスしない文であれば、その索引を必要とする文も実行できます。また、使用不可のパーティションは、分割または改名してから再構築できます。

**RENAME  
PARTITION** 索引の *partition\_name* を *new\_partition\_name* に改名します。

**DROP PARTITION** グローバル・パーティション索引からパーティションとその中のデータを削除します。グローバル索引のパーティションを削除すると、その索引の次のパーティションに使用不可のマークが付けられます。グローバル索引の最上位のパーティションは削除できません。

*split\_partition\_  
clause* グローバル・パーティション索引を 2 つのパーティションに分割し、新しいパーティションを索引に追加します。

使用不可のマークが付けられたパーティションを分割すると 2 つのパーティションが生成されますが、その両方に使用不可のマークが付けられます。このようなパーティションは、使用前に再構築する必要があります。



使用可能のパーティションを分割すると、索引データが入っている2つのパーティションが生成され、両方に使用可能のマークが付けられます。

**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* の分割前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** 分割の結果生成される2つのパーティションを記述します。

*partition\_description,*  
*partition\_description* 分割の結果生成される2つのパーティションの名前と物理属性を指定します。

## REBUILD PARTITION

索引のパーティションを1つ再構築します。このオプションは、索引パーティションを別の表領域に移動したり、作成時の物理属性を変更するために使用できます。パーティションのメンテナンス操作の説明は、『Oracle8 Server 管理者ガイド』を参照してください。

## 例

**例 1** 次の文は、データ・ブロックが、5つの初期トランザクション・エントリと100KBの増分エクステントを使用するように、SCOTTの索引CUSTOMERを変更します。

```
ALTER INDEX scott.customer
  INITRANS 5
  STORAGE (NEXT 100K);
```

**例 2** 次の例では、索引パーティションIX\_ANTARTICAを削除します。

```
ALTER INDEX sales_area_ix
  DROP PARTITION ix_antarctica;
```

**例 3** この文は、ローカル・パーティション索引SALES\_IX3のすべてのパーティションの実属性を変更します。将来追加される新しいパーティションでは、5つの初期トランザクション・エントリと100Kの増分エクステントが使用されます。

```
ALTER INDEX sales_ix3 INITRANS 5 STORAGE ( NEXT 100K );
```

**例 3(a)** この例では、ローカル・パーティション索引SALES\_IX3のデフォルトの属性を変更します。将来追加される新しいパーティションでは、5つの初期トランザクション・エントリと100Kの増分エクステントが使用されます。

```
ALTER INDEX sales_ix3
  MODIFY DEFAULT ATTRIBUTES INITRANS 5 STORAGE ( NEXT 100K );
```

**例 4** 次の例では、IDX\_ACCTNO索引にUNUSABLEのマークを付けます。

```
ALTER INDEX idx_acctno UNUSABLE;
```

**例 5** 次の例では、パーティション **BRIX\_NY** のエクステンツの最大数を変更します。

```
ALTER INDEX branch_ix MODIFY PARTITION brix_ny
    STORAGE( MAXEXTENTS 30 ) LOGGING;
```

**例 6** 次の例では、索引 **IDX\_ACCTNO** のパーティション **IDX\_FEB96** に **UNUSABLE** のマークを付けます。

```
ALTER INDEX idx_acctno MODIFY PARTITION idx_feb96 UNUSABLE;
```

**例 7** 次の例では、索引 **ARTIST\_IX** のパラレル属性を設定します。

```
ALTER INDEX artist_ix PARALLEL (DEGREE 4, INSTANCES 3);
```

**例 8** 次の例では、索引 **ARTIST\_IX** に対する走査がパラレル化されないように、索引のパラレル属性を設定します。

```
ALTER INDEX artist_ix NOPARALLEL;
```

**例 9** 次の例では、索引 **ARTIST\_IX** 内のパーティション **P063** を再構築します。索引パーティションの再構築は、ログに記録されません。

```
ALTER INDEX artist_ix
    REBUILD PARTITION p063 NOLOGGING;
```

**例 10** 次の例では、索引を改名します。

```
ALTER INDEX emp_ix1 RENAME TO employee_ix1;
```

**例 11** 次の例では、索引パーティションを改名します。

```
ALTER INDEX employee_ix2 RENAME PARTITION emp_ix2_p3
    TO employee_ix2_p3;
```

**例 12** 次の例では、パーティション索引 **PARTNUM\_IX** 内のパーティション **PARTNUM\_IX\_P6** を **PARTNUM\_IX\_P5** と **PARTNUM\_IX\_P6** に分割します。

```
ALTER INDEX partnum_ix
    SPLIT PARTITION partnum_ix_p6 AT ( 5001 )
    INTO ( PARTITION partnum_ix_p5 TABLESPACE ts017 LOGGING,
          PARTITION partnum_ix_p6 TABLESPACE ts004 );
```

2 番目のパーティションには元のパーティションの名前が付けられます。

**例 13** 次の例では、索引ブロックのバイトが **REVERSE** 順に格納されるように、索引 **EMP\_IX** を再構築します。

```
ALTER INDEX emp_ix REBUILD REVERSE;
```

## 関連項目

CREATE INDEX (4-233 ページ)

CREATE TABLE (4-303 ページ)

PARALLEL 句 (4-462 ページ)

STORAGE 句 (4-518 ページ)

DEALLOCATE UNUSED 句 (4-368 ページ)

# ALTER PACKAGE

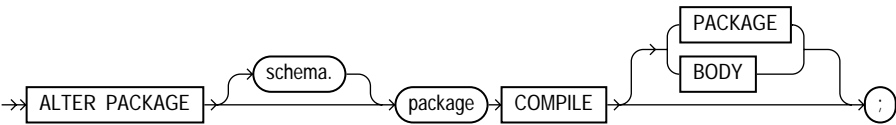
## 用途

ストアド・パッケージを再コンパイルします。「使用上の注意」（4-38 ページ）も参照してください。

## 前提条件

パッケージが自スキーマ内になければなりません。自スキーマ内にはない場合は ALTER 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

---

<i>schema</i>	パッケージが定義されているスキーマを指定します。 <i>schema</i> を指定しないと、Oracle はパッケージが自スキーマ内に定義されているものとみなします。
<i>package</i>	再コンパイルするパッケージの名前です。
COMPILE	パッケージ仕様部とパッケージ本体のいずれかを再コンパイルします。COMPILE キーワードは必須です。
PACKAGE	パッケージ本体とパッケージ仕様部の両方を再コンパイルします。
BODY	パッケージ本体だけを再コンパイルします。 デフォルトのオプションは PACKAGE です。

---

## 使用上の注意

ALTER PACKAGE コマンドを使用してパッケージの仕様部と本体の両方、またはパッケージ本体だけを明示的に再コンパイルできます。明示的に再コンパイルすることによって、実行時に暗黙的に再コンパイルする必要がなくなり、また、実行時のコンパイル・エラーとパフォーマンス上のオーバーヘッドもなくなります。

パッケージ中のオブジェクトはすべて 1 つの単位として格納されているため、ALTER PACKAGE コマンドによって、すべてのパッケージ・オブジェクトがまとめて再コンパイルされます。ALTER PROCEDURE コマンドまたは ALTER FUNCTION コマンドを使用して、

パッケージ中の一部のプロシージャまたは関数を個別に再コンパイルすることはできません。

---

**注意：** このコマンドを使用して既存のパッケージの宣言や定義は変更できません。パッケージを再宣言または再定義するには、**CREATE PACKAGE** コマンドまたは **OR REPLACE** オプションを指定して **CREATE PACKAGE BODY** コマンドを使う必要があります。

---

## パッケージ仕様部の再コンパイル

パッケージ仕様部を変更した場合、コンパイル・エラーの有無を検査するには再コンパイルします。**COMPILE PACKAGE** オプションを指定して **ALTER PACKAGE** 文を発行すると、その有効、無効にかかわらずパッケージ仕様部と本体が再コンパイルされます。その結果、そのパッケージ中のプロシージャまたは関数をコールするプロシージャなど、再コンパイルされたパッケージ仕様部に依存するローカル・オブジェクトはすべて無効になります。パッケージ本体もそのパッケージ仕様部に依存します。その後明示的な再コンパイルを行わずに、これらの依存オブジェクトを参照すると、Oracle は実行時にそれらを暗黙的に再コンパイルします。

## パッケージ本体の再コンパイル

パッケージ本体は、変更後に再コンパイルしてください。**COMPILE BODY** オプションを指定して **ALTER PACKAGE** 文を発行すると、その有効、無効にかかわらずパッケージと本体が再コンパイルされます。パッケージ本体を再コンパイルすると、そのパッケージ本体が依存するオブジェクトが無効の場合には、最初にそのオブジェクトが再コンパイルされます。パッケージ本体の再コンパイルが正常終了すれば、この本体は有効になります。パッケージ本体の再コンパイル時にコンパイル・エラーが発生した場合は、エラーが戻り、パッケージ本体は無効のままです。このとき、事前に定義されているパッケージ **DBMS\_OUTPUT** を使用して、そのパッケージ本体をデバッグできます。なお、パッケージ本体を再コンパイルしても、そのパッケージ仕様部に基づくオブジェクトは無効にはなりません。

パッケージのデバッグの詳細は、『Oracle8 Server アプリケーション開発者ガイド』を参照してください。リモート・オブジェクトを含むスキーマ・オブジェクトの依存性を Oracle がメンテナンスする方法については、『Oracle8 Server 概要』を参照してください。

**例 1** 次の文は、スキーマ **BLAIR** 内の **ACCOUNTING** パッケージの仕様部と本体を明示的に再コンパイルします。

```
ALTER PACKAGE blair.accounting  
COMPILE PACKAGE;
```

**ACCOUNTING** の仕様部と本体の再コンパイル時にコンパイル・エラーが発生しなければ、**ACCOUNTING** は有効になります。その後実行時に再コンパイルすることなく、**BLAIR** によって、**ACCOUNTING** の仕様部に宣言されたあらゆるパッケージ・オブジェクトをコール

または参照できます。ACCOUNTING の再コンパイル時にコンパイル・エラーが発生した場合は、エラー・メッセージが戻り、ACCOUNTING は無効のままです。

このとき ACCOUNTING に依存するオブジェクトもすべて無効となります。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すると、Oracle によってそれらが実行時に暗黙的に再コンパイルされます。

**例 2** 次の文は、スキーマ BLAIR 内の ACCOUNTING パッケージの本体を再コンパイルします。

```
ALTER PACKAGE blair.accounting  
COMPILE BODY;
```

パッケージ本体の再コンパイル時にコンパイル・エラーが発生しなければ、この本体は有効となります。その後実行時に再コンパイルすることなく、BLAIR によって、ACCOUNTING の仕様部に宣言されたあらゆるパッケージ・オブジェクトをコールまたは参照できます。本体の再コンパイル時にコンパイル・エラーが発生した場合は、エラー・メッセージが戻り、パッケージ本体は無効のままです。

この文は ACCOUNTING の仕様部ではなく本体を再コンパイルするため、依存するオブジェクトは無効にはなりません。

## 関連項目

CREATE PACKAGE (4-246 ページ)

CREATE PACKAGE BODY (4-250 ページ)

# ALTER PROCEDURE

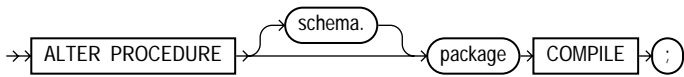
## 用途

スタンドアロンのストアド・プロシージャを再コンパイルします。「使用上の注意」(4-41 ページ)も参照してください。

## 前提条件

プロシージャが自スキーマ内にある必要があります。自スキーマ内にはない場合は、ALTER 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

<i>schema</i>	プロシージャが定義されているスキーマを指定します。 <i>schema</i> を指定しないと、Oracle はプロシージャが自スキーマ内に定義されているものとみなします。
<i>procedure</i>	再コンパイルするプロシージャ名です。
COMPILE	指定したプロシージャが再コンパイルされます。COMPILE キーワードは必須です。

## 使用上の注意

ALTER PROCEDURE コマンドは ALTER FUNCTION コマンドと似ています。次のプロシージャの明示的な再コンパイルの説明は、関数に対しても同じように適用されます。

ALTER PROCEDURE コマンドを使用して、無効なプロシージャを明示的に再コンパイルできます。明示的に再コンパイルすることによって、実行時に暗黙的に再コンパイルされる必要がなくなり、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。

ALTER PROCEDURE 文を発行すると、指定したプロシージャが有効でも無効でも関わらず再コンパイルされます。

ALTER PROCEDURE コマンドは、スタンドアロン・プロシージャを再コンパイルする場合にだけ使用できます。パッケージの一部であるプロシージャを再コンパイルするには、ALTER PACKAGE コマンドを使用してそのパッケージ全体を再コンパイルします。

プロシージャを再コンパイルすると、そのプロシージャ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。さらに、プロシージャに依存するあらゆるローカル・オブジェクト（たとえば、再コンパイルしたプロシージャをコールするプロシージャ、または再コンパイルしたプロシージャをコールするプロシージャを定義するパッケージ本体など）が無効になります。プロシージャの再コンパイルが正常終了すれば、このプロシージャは有効になります。プロシージャの再コンパイル時にエラーが発生した場合は、エラー・メッセージが戻り、プロシージャは無効のままです。このとき、事前に定義されているパッケージ DBMS\_OUTPUT を使用して、プロシージャをデバッグできます。プロシージャのデバッグの説明は、『Oracle8 Server アプリケーション開発者ガイド』を参照してください。リモート・オブジェクトを含むスキーマ・オブジェクトの依存性を Oracle がメンテナンスする方法については、『Oracle8 Server 概要』を参照してください。

---

---

**注意：** このコマンドは、既存のプロシージャの宣言や定義は変更しません。プロシージャを再宣言または再定義するには、**OR REPLACE** オプションを指定して **CREATE PROCEDURE** コマンドを使用します。

---

---

**例** 次の例では、ユーザー HENRY のプロシージャ CLOSE\_ACCT を明示的に再コンパイルします。

```
ALTER PROCEDURE henry.close_acct
  COMPILE;
```

CLOSE\_ACCT の再コンパイル時にエラーが発生しなければ、CLOSE\_ACCT は有効になります。その後、Oracle は実行時に CLOSE\_ACCT を再コンパイルせずに実行できます。CLOSE\_ACCT のコンパイル時にコンパイル・エラーが発生した場合は、エラー・メッセージが戻り、CLOSE\_ACCT 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、CLOSE\_ACCT をコールするプロシージャ、関数、パッケージ本体などです。その後明示的な再コンパイルを行わずに、これらのオブジェクトを参照すると、Oracle は実行時にそれらを暗黙的に再コンパイルします。

## 関連項目

ALTER FUNCTION (4-26 ページ)

ALTER PACKAGE (4-38 ページ)

CREATE PROCEDURE (4-255 ページ)



---

## ALTER PROFILE

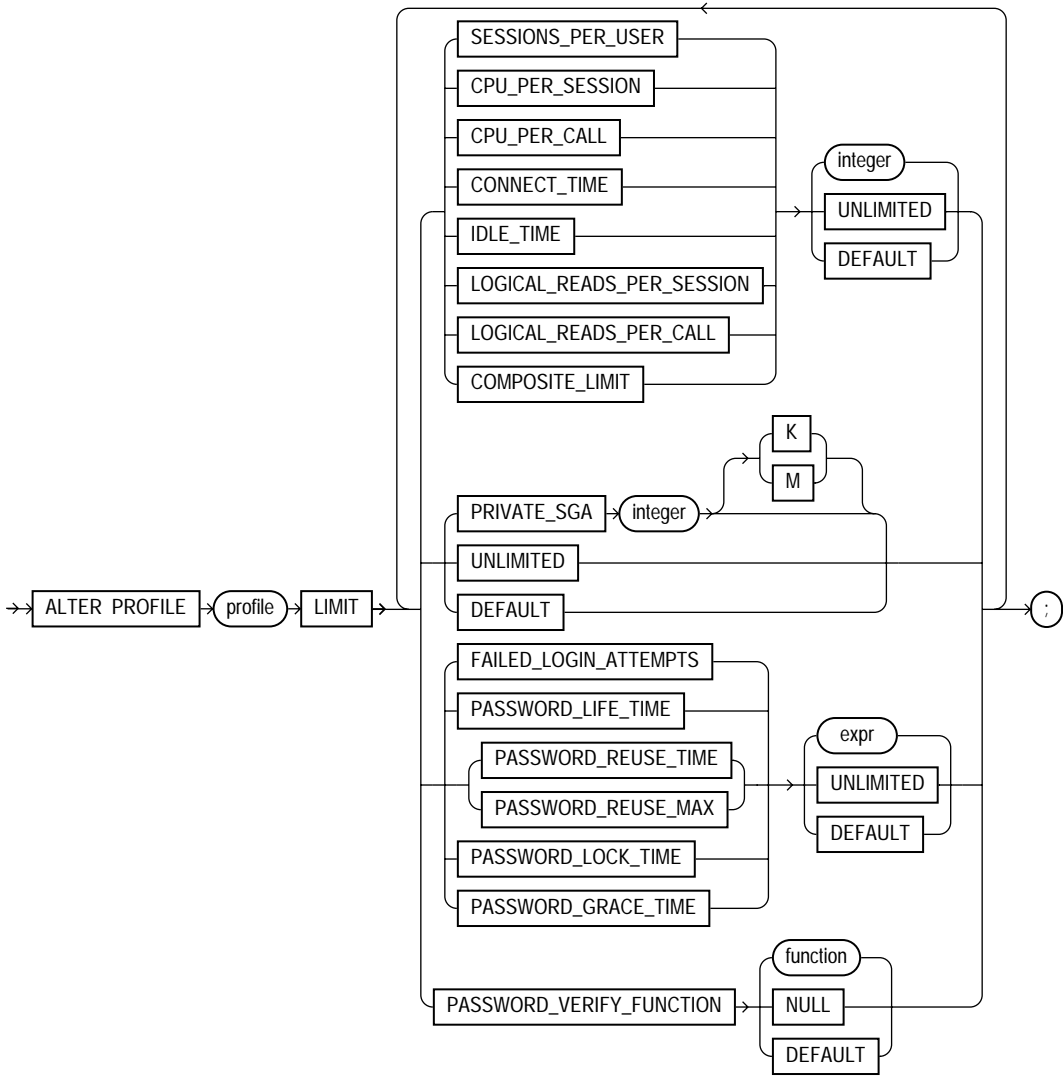
### 用途

プロファイルのリソース制限またはパスワード管理の追加または変更、削除を行います。「例」(4-45 ページ) も参照してください。

### 前提条件

プロファイルのリソース制限を変更するには、ALTER PROFILE システム権限が必要です。パスワード制限と保護を変更するには、ALTER PROFILE と ALTER USER システム権限が必要です。「パスワード履歴の使用」(4-45 ページ) を参照してください。

構文



キーワードとパラメータ

<i>profile</i>	変更するプロファイルの名前です。
<i>integer</i>	このプロファイルのリソースに対する新しい制限を定義します。

---

ALTER PROFILE パラメータのリソース制限については、CREATE PROFILE（4-261 ページ）を参照してください。

注意：

- DEFAULT プロファイルからは制限を削除できません。
- 日に関するすべてのパラメータには分数を単位として使用できます。分数は、 $x/y$  で表します。たとえば、1 時間は 1/24、1 分は 1/1440 になります。

---

## パスワード履歴の使用

ALTER PROFILE 文を使用してプロファイルに対して行った変更は、このプロファイルの現行セッションのユーザーには影響しません。後続のセッションのユーザーだけに影響します。

パスワード履歴パラメータを指定するときには、次の制限が適用されます。

- PASSWORD\_REUSE\_TIME を整数値に設定する場合、PASSWORD\_REUSE\_MAX は UNLIMITED に設定しなければならない。PASSWORD\_REUSE\_MAX を整数値に設定する場合、PASSWORD\_REUSE\_TIME は UNLIMITED に設定しなければならない。
- PASSWORD\_REUSE\_TIME と PASSWORD\_REUSE\_MAX の両方を UNLIMITED に設定すると、どちらのパスワードのリソースも使用されない。
- PASSWORD\_REUSE\_MAX を DEFAULT に設定し、PASSWORD\_REUSE\_TIME を UNLIMITED に設定すると、デフォルト・プロファイルに定義された PASSWORD\_REUSE\_MAX 値が使用される。
- PASSWORD\_REUSE\_TIME を DEFAULT に設定し、PASSWORD\_REUSE\_MAX を UNLIMITED に設定すると、デフォルト・プロファイルに定義された PASSWORD\_REUSE\_TIME 値が使用される。
- PASSWORD\_REUSE\_TIME と PASSWORD\_REUSE\_MAX の両方を DEFAULT に設定すると、デフォルト・プロファイルに定義された値はどちらでも使用される。

## 例

**例 1** 次の例では、パスワードを 90 日間再利用できないようにします。

```
ALTER PROFILE prof
LIMIT PASSWORD_REUSE_TIME 90
PASSWORD_REUSE_MAX UNLIMITED;
```

**例 2** 次の例では、PASSWORD\_REUSE\_TIME 値を DEFAULT プロファイルに定義された値にデフォルト設定します。

```
ALTER PROFILE prof
LIMIT PASSWORD_REUSE_TIME DEFAULT
PASSWORD_REUSE_MAX UNLIMITED;
```

**例 3** 次の例では、プロファイル PROF の FAILED\_LOGIN\_ATTEMPTS を 5 に設定し、PASSWORD\_LOCK\_TIME を 1 に設定します。

```
ALTER PROFILE prof LIMIT
FAILED_LOGIN_ATTEMPTS 5
PASSWORD_LOCK_TIME 1;
```

このコマンドを使用すると、PROF のアカウントはログインに 5 回失敗した場合に 1 日ロックされます。

**例 4** 次の例では、プロファイル PROF の PASSWORD\_LIFE\_TIME を 60 日に、PASSWORD\_GRACE\_TIME を 10 日に変更します。

```
ALTER PROFILE prof LIMIT
PASSWORD_LIFE_TIME 60
PASSWORD_GRACE_TIME 10;
```

**例 5** 次の例は、ENGINEER プロファイルに同時実行のセッションの新たな制限 (5) を指定します。

```
ALTER PROFILE engineer LIMIT SESSIONS_PER_USER 5;
```

ENGINEER プロファイルに SESSIONS\_PER\_USER の制限が現在定義されていない場合は、このプロファイルに制限 (5) が追加されます。プロファイルに制限が定義されている場合には、上の文によってその制限が 5 に再定義されます。ENGINEER プロファイルを割り当てられているすべてのユーザーは、同時実行のセッションが 5 件に制限されます。

**例 6** 次の文は、ENGINEER プロファイルに無制限のアイドル時間を定義します。

```
ALTER PROFILE engineer LIMIT IDLE_TIME UNLIMITED;
```

ENGINEER プロファイルを割り当てられているユーザーは誰でも次のセッションから無制限のアイドル時間が認められます。

**例 7** 次の文は、ENGINEER プロファイルの IDLE\_TIME の制限を削除します。

```
ALTER PROFILE engineer LIMIT IDLE_TIME DEFAULT;
```

ENGINEER プロファイルを割り当てられているユーザーは、後続のセッションからは DEFAULT プロファイルに定義された IDLE\_TIME の制限に従います。

**例 8** 次の文は、DEFAULT プロファイルにアイドル時間の制限 (2 分) を定義します。

```
ALTER PROFILE default LIMIT IDLE_TIME 2;
```

IDLE\_TIME の制限は次のユーザーに適用されます。

- プロファイルを明示的に割り当てられていないユーザー

- `IDLE_TIME` の制限が定義されていないプロファイルを明示的に割り当てられているユーザー

## 関連項目

[CREATE PROFILE \(4-261 ページ\)](#)

# ALTER RESOURCE COST

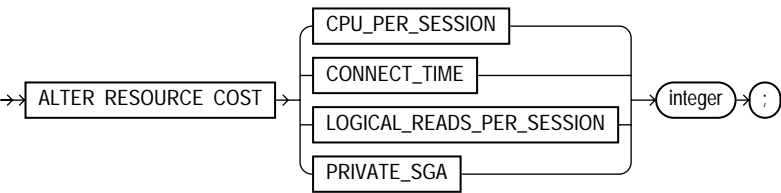
## 用途

セッションで使用されたリソース・コストの合計を算出する式を指定します。どのセッションについても、このコストはユーザーのプロファイル内の **COMPOSITE\_LIMIT** パラメータの値によって制限されます。「使用上の注意」（4-48 ページ）も参照してください。

## 前提条件

ALTER RESOURCE COST システム権限が必要です。

## 構文



## キーワードとパラメータ

CPU_PER_SESSION	セッションによって使用された CPU 時間で、単位は 100 分の 1 秒です。
CONNECT_TIME	セッションによって使用された CPU 時間で、単位は 100 分の 1 秒です。
CPU_PER_SESSION	セッションの経過時間で、単位は分です。
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、セッション中に読み込まれたデータ・ブロックの数です。
PRIVATE_SGA	セッションごとのシステム・グローバル領域 (SGA) 内のプライベート領域のバイト数です。マルチスレッド・サーバー・アーキテクチャを使用して、自セッション用として SGA 内でプライベート領域を割り当てている場合にだけ、この制限が適用されます。
<i>integer</i>	各ソースのウェイト（重み）です。

## 使用上の注意

ALTER RESOURCE COST コマンドは、セッションで使用するリソース・コストの合計を算出するための式を指定します。セッションで使用された各リソースの量にそのリソースのウェイトを乗算し、4 種類のリソースの乗算結果を加算することによってリソース・コストの

合計が算出されます。乗算結果と総コストは、ともにサービス単位と呼ばれる単位で表されます。

他のリソースの使用も監視されますが、セッションに対するリソース・コストの合計はこの4種類のリソースに基づいて算出されます。各リソースの詳細は、**CREATE PROFILE** (4-261 ページ) を参照してください。

各リソースに割り当てるウエイトによって、各リソースがリソース・コストの合計に影響する程度が決まります。ウエイトの小さいリソースを使用すると、ウエイトの大きいリソースを使用する場合と比べてコストに及ぼす影響は小さくなります。リソースにウエイトを割り当てない場合はデフォルト値の 0 が適用され、コストへの影響はありません。ウエイトを割り当てた場合は、データベースの次のセッション以降のすべてのセッションでそのウエイトが適用されます。

リソース・コストの合計を算出するための式を指定すると、**CREATE PROFILE** コマンドの **COMPOSITE\_LIMIT** パラメータを使用してセッションに対するコストを制限できます。セッションのコストが制限を超えると、セッションは異常終了し、エラーが戻ります。リソース制限の設定については、**CREATE PROFILE** (4-261 ページ) を参照してください。各リソースに割り当てたウエイトを変更するために **ALTER RESOURCE COST** コマンドを使用すると、現行のセッション以降のすべてのセッションで、その新たなウエイトを基にしてリソース・コストが計算されます。

**例 1** 次の文は、リソース **CPU\_PER\_SESSION** と **CONNECT\_TIME** にウエイトを割り当てます。

```
ALTER RESOURCE COST
CPU_PER_SESSION 100
CONNECT_TIME      1;
```

このウエイトによって、セッションに対して次のコスト計算式が設定されます。

$$T = (100 * CPU) + CON$$

ここで、それぞれの意味は次のとおりです。

<b>T</b>	サービス単位で表される、セッションに対するリソース・コストの合計です。
<b>CPU</b>	セッションで使用された CPU 時間で、単位は 100 分の 1 秒です。
<b>CON</b>	セッションの経過時間で、単位は分です。

ここではリソース **LOGICAL\_READS\_PER\_SESSION** と **PRIVATE\_SGA** にウエイトを割り当てていないため、これらのリソースは式に含まれません。

プロファイルで **COMPOSITE\_LIMIT** 値として 500 を割り当てた場合、**T** が 500 を超えるときに必ずセッションはこの制限を超えます。たとえば、CPU 時間 0.04 秒、経過時間 101 分を使用するセッションは、この制限を超えます。同様に、CPU 時間が 0.0301 秒、経過時間が 200 分のセッションもこの制限を超えてしまいます。

一度割り当てたウェイトは、次のように別の ALTER RESOURCE 文を発することにより変更できます。

```
ALTER RESOURCE COST
LOGICAL_READS_PER_SESSION 2
CONNECT_TIME 0;
```

新たに割り当てたウェイトによって、新たなコスト計算式が設定されます。

$$T = (100 * CPU) + (2 * LOG)$$

ここで、それぞれの意味は次のとおりです。

T CPU                      これらの意味は前述の式と同じです。

LOG                        セッション中に読み込まれるデータ・ブロックの数です。

この ALTER RESOURCE COST 文によって、式は次のように変更されます。

- CPU\_PER\_SESSION リソースのウェイトは指定しないが、このリソースにはすでにウェイトが割り当てられているため、式には先に指定したウェイトがそのまま使用されます。
- LOGICAL\_READS\_PER\_SESSION リソースにウェイトを割り当てたため、このリソースが式で使用されます。
- CONNECT\_TIME リソースに 0 を割り当てたため、このリソースは式に含まれていません。
- PRIVATE\_SGA リソースのウェイトを指定せず、かつ、このリソースにはウェイトが指定されていなかったため、このリソースは式で使用されません。

## 関連項目

CREATE PROFILE (4-261 ページ)



## ALTER ROLE

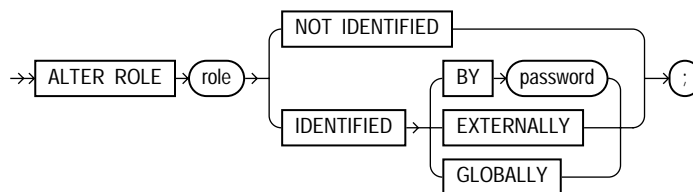
### 用途

ロールを使用可能にするために必要な許可を変更します。「使用上の注意」（4-51 ページ）も参照してください。

### 前提条件

ロールに ADMIN OPTION が付与されている必要があります。付与されていない場合は、ALTER ANY ROLE システム権限が必要です。

### 構文



### キーワードとパラメータ

ALTER ROLE コマンドのキーワードとパラメータの意味はすべて CREATE ROLE コマンドのキーワードとパラメータと同じです。CREATE ROLE（4-268 ページ）を参照してください。

### 使用上の注意

ロールを IDENTIFIED GLOBALLY に変更する前に次の作業が必要です。

- ロールに対して外部的に識別されたロール権限をすべて取り消す。
- すべてのユーザーおよびロール、PUBLIC のロール権限付与を取り消す。

この規則の唯一の例外は、現在ロールを変更しているユーザーからはそのロールを取り消さないことです。

ALTER ANY ROLE を持つユーザーが、IDENTIFIED GLOBALLY であるロールを次のいずれかに変更すると、Oracle によってそのロールに ADMIN OPTION が付与されます。

- IDENTIFIED BY *password*
- IDENTIFIED EXTERNALLY
- NOT IDENTIFIED

**例 1** 次の例では、ロール ANALYST を IDENTIFIED GLOBALLY に変更します。

```
ALTER ROLE analyst IDENTIFIED GLOBALLY;
```

**例 2** 次の文は、TELLER ロールのパスワードを LETTER に変更します。

```
ALTER ROLE teller  
IDENTIFIED BY letter;
```

パスワードの変更後、TELLER ロールが付与されているユーザーは、新しいパスワード "letter" を入力してこのロールを使用可能にしなければなりません。

## 関連項目

CREATE ROLE (4-268 ページ)

SET ROLE (4-511 ページ)

## ALTER ROLLBACK SEGMENT

### 用途

次のいずれかの方法で、ロールバック・セグメントを変更します。

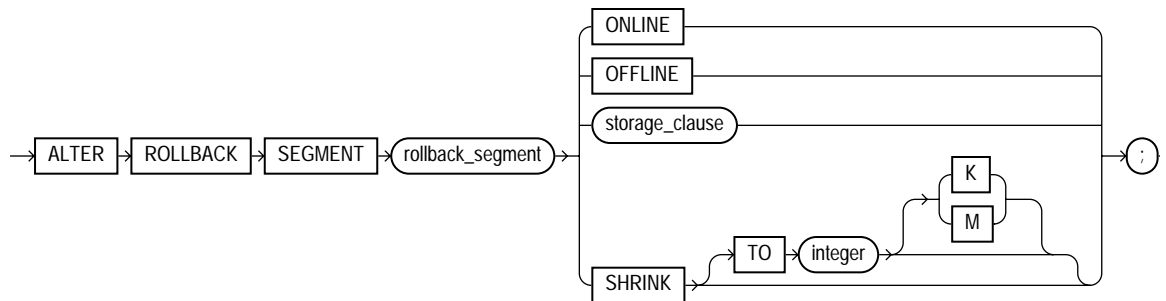
- オンラインにする
- オフラインにする
- 記憶特性を変更する
- 最適サイズまたは指定サイズに縮小する

詳細は、「使用上の注意」（4-54 ページ）を参照してください。

### 前提条件

ALTER ROLLBACK SEGMENT システム権限が必要です。

### 構文



`storage_clause:STORAGE` 句（4-518 ページ）を参照。

## キーワードとパラメータ

<i>rollback_segment</i>	既存のロールバック・セグメントの名前です。
ONLINE	ロールバック・セグメントをオンラインにします。
OFFLINE	ロールバック・セグメントをオフラインにします。
<i>storage_clause</i>	ロールバック・セグメントの記憶特性を変更します。構文および追加情報については、 <b>STORAGE</b> 句 (4-518 ページ) を参照してください。
SHRINK	ロールバック・セグメントを最適サイズまたは指定サイズに縮小します。

## 使用上の注意

ロールバック・セグメントを作成すると、最初はオフライン状態になります。オフラインのロールバック・セグメントはトランザクションに対して使用できません。

**ONLINE** オプションを指定すると、ロールバック・セグメントはオンラインとなり、インスタンスはトランザクションに対してそのロールバック・セグメントを使用できます。また、初期化パラメータ **ROLLBACK\_SEGMENT** を使用すると、インスタンスの起動時にロールバック・セグメントをオンラインにできます。

**OFFLINE** オプションはロールバック・セグメントをオフラインにします。ロールバック・セグメント内に、アクティブ・トランザクションのロールバックに必要な情報が含まれていない場合には、ただちにオフラインになります。ロールバック・セグメントにアクティブ・トランザクションについての情報が含まれている場合、このロールバック・セグメントをその後のトランザクションに対して使用できないようにし、そのアクティブ・トランザクションがすべてコミットまたはロールバックされた後で、ロールバック・セグメントはオフラインになります。一度オフラインにされたロールバック・セグメントは、どのインスタンスからもオンラインにできません。

**SYSTEM** ロールバック・セグメントはオフラインにできません。

ロールバック・セグメントがオンラインまたはオフラインのどちらであるかを確認するには、データ・ディクショナリ・ビュー **DBA\_ROLLBACK\_SEGS** を問い合わせます。オンライン・ロールバック・セグメントの場合は、**STATUS** の値が **IN\_USE** です。オフライン・ロールバック・セグメントの場合は、**STATUS** の値が **AVAILABLE** です。

ロールバック・セグメントを使用可能または使用禁止にする方法の詳細は、『Oracle8 Server 管理者ガイド』を参照してください。

**ALTER ROLLBACK SEGMENT** コマンドの **STORAGE** 句を指定すると、それ以降のロールバック・セグメント内の領域の割当てに影響を及ぼします。既存のロールバック・セグメントに対する **INITIAL** と **MINEXTENTS** の値は変更できません。

**ALTER ROLLBACK SEGMENT** コマンドで **SHRINK** 句を指定すると、指定したロールバック・セグメントの最適サイズへの縮小が開始されます。サイズを指定しない場合は、ロールバック・セグメントの作成に使用した **CREATE ROLLBACK SEGMENT** コマンドの **STORAGE** 句の **OPTIMAL** で指定した値がデフォルトのサイズになります。**OPTIMAL** 値を指定しないと、**CREATE ROLLBACK SEGMENT** コマンドの **STORAGE** 句の **MINEXTENTS**

で指定した値がデフォルトのサイズになります。コマンドの実行時には **SHRINK** 句で指定したサイズが有効です。その後、**OPTIMAL** は **CREATE ROLLBACK SEGMENT** コマンドの **OPTIMAL** の値に戻ります。サイズを指定してもしなくても、ロールバック・セグメントはエクステンツ数 2 未満には縮小できません。

ロールバック・セグメントの縮小を実行してから、**DBA\_ROLLBACK\_SEGS** ビューに対して問合せを行いロールバック・セグメントの実際のサイズを判別できます。

パラレル・サーバーでは、自インスタンスに対してオンライン状態のロールバック・セグメントだけを縮小できます。

**SHRINK** オプションは、ロールバック・セグメントのサイズ縮小を試行します。縮小されるかどうか、および縮小量は次の状況で変わります。

- ロールバック・セグメントの使用可能領域
- アクティブ・トランザクションのロールバック・セグメント内での領域保持状態

**例 1** 次の文は、ロールバック・セグメント **RSONE** をオンラインにします。

```
ALTER ROLLBACK SEGMENT rsone ONLINE;
```

**例 2** 次の文は、**RSONE** の **STORAGE** パラメータを変更します。

```
ALTER ROLLBACK SEGMENT rsone  
STORAGE (NEXT 1000 MAXEXTENTS 20);
```

**例 3** 次の文は、ロールバック・セグメントを最適サイズ 100MB に変更します。

```
ALTER ROLLBACK SEGMENT rsone  
SHRINK TO 100 M;
```

## 関連項目

**CREATE ROLLBACK SEGMENT** (4-272 ページ)

**CREATE TABLESPACE** (4-325 ページ)

**STORAGE** 句 (4-518 ページ)

## ALTER SEQUENCE

### 用途

次の方法で順序を変更します。

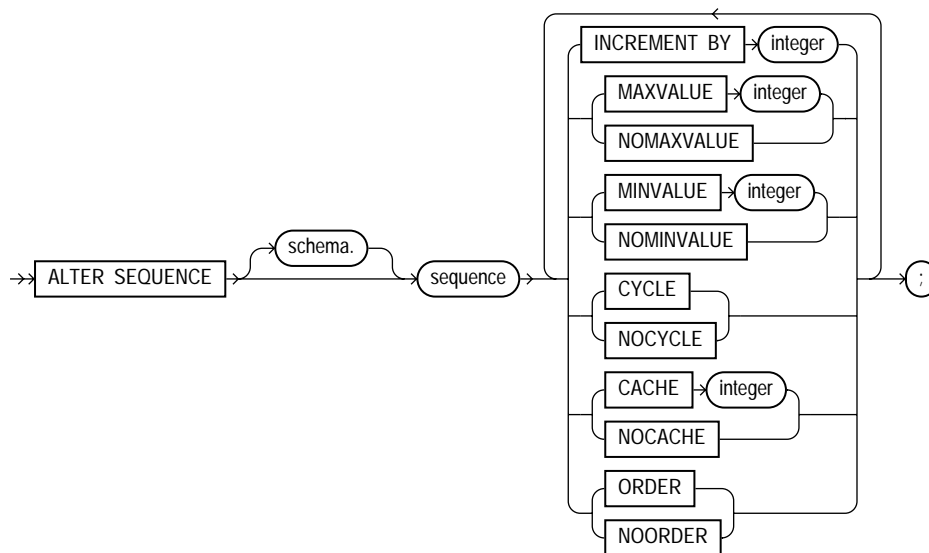
- 今後の順序番号間の増分値を変更する。
- 最小値または最大値を設定または削除する。
- キャッシュ順序番号を変更する。
- 順序が最大値または最小値に達した後も、引き続き番号を生成することを指定する。
- 順序番号を並べ変えるかどうかを指定する。

これらの用途については、「例」(4-57 ページ) にいくつか例を示します。

### 前提条件

順序が自スキーマ内にあることが必要です。自スキーマ内にはない場合は、順序に対する ALTER 権限、または ALTER 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

このコマンドのキーワードとパラメータは、CREATE SEQUENCE（4-278 ページ）と同じ用途で使われます。

### 注意：

- 異なる順序番号で再度開始するには、順序を削除して再作成する必要があります。なお、ALTER SEQUENCE コマンドの影響を受けるのは、コマンド実行後の順序番号に限られます。
- いくつかの妥当性検査が行われます。たとえば、MAXVALUE の値に現行の順序番号より小さい値は指定できません。

## 例

**例 1** 次の文は、ESEQ 順序に新たな最大値を設定します。

```
ALTER SEQUENCE eseq  
MAXVALUE 1500;
```

**例 2** 次の文は、ESEQ 順序に CYCLE オプションと CACHE オプションを指定します。

```
ALTER SEQUENCE eseq  
CYCLE  
CACHE 5;
```

## 関連項目

CREATE SEQUENCE（4-278 ページ）

DROP SEQUENCE（4-397 ページ）

## ALTER SESSION

---

### 用途

次のいずれかの方法で、現行セッションを変更します。

- SQL トレース機能を使用可能または使用禁止にする。
- グローバル・ネーム変換を使用可能または使用禁止にする。
- NLS パラメータの値を変更する。
- 使用頻度の高いカーソルを保持するためのキャッシュのサイズを指定する。
- キャッシュに入れたカーソルの COMMIT または ROLLBACK 実行時のクローズを、使用可能または使用禁止にする。
- パラレル・サーバーにおいて、別のインスタンスに接続される場合と同様に、セッションがデータベース・ファイルにアクセスする必要があることを指定する。
- ハッシュ結合処理の動作を使用可能にする、使用禁止にする、または変更する。
- リモート・プロシージャ・コールの依存性の処理方式を変更する。
- トランザクション・レベルの処理方式を変更する。
- データベース・リンクをクローズする。
- インダウト分散トランザクションを強制処理するためのアドバイスをリモート・データベースに送信する。
- ストアド・プロシージャやストアド・ファンクションでの、COMMIT 文と ROLLBACK 文の発行を許可または禁止する。
- コストベースの最適化方法の目標を変更する。
- パラレル・サーバーで、DML 文をパラレルで実行できるようにする。
- 使用禁止のマークが付いた索引や索引パーティションを持つ表に対し、挿入、更新、削除を行う。
- 遅延可能な制約のチェック、各 DML 文の実行直後またはトランザクションの終了時に実行できるようにする。

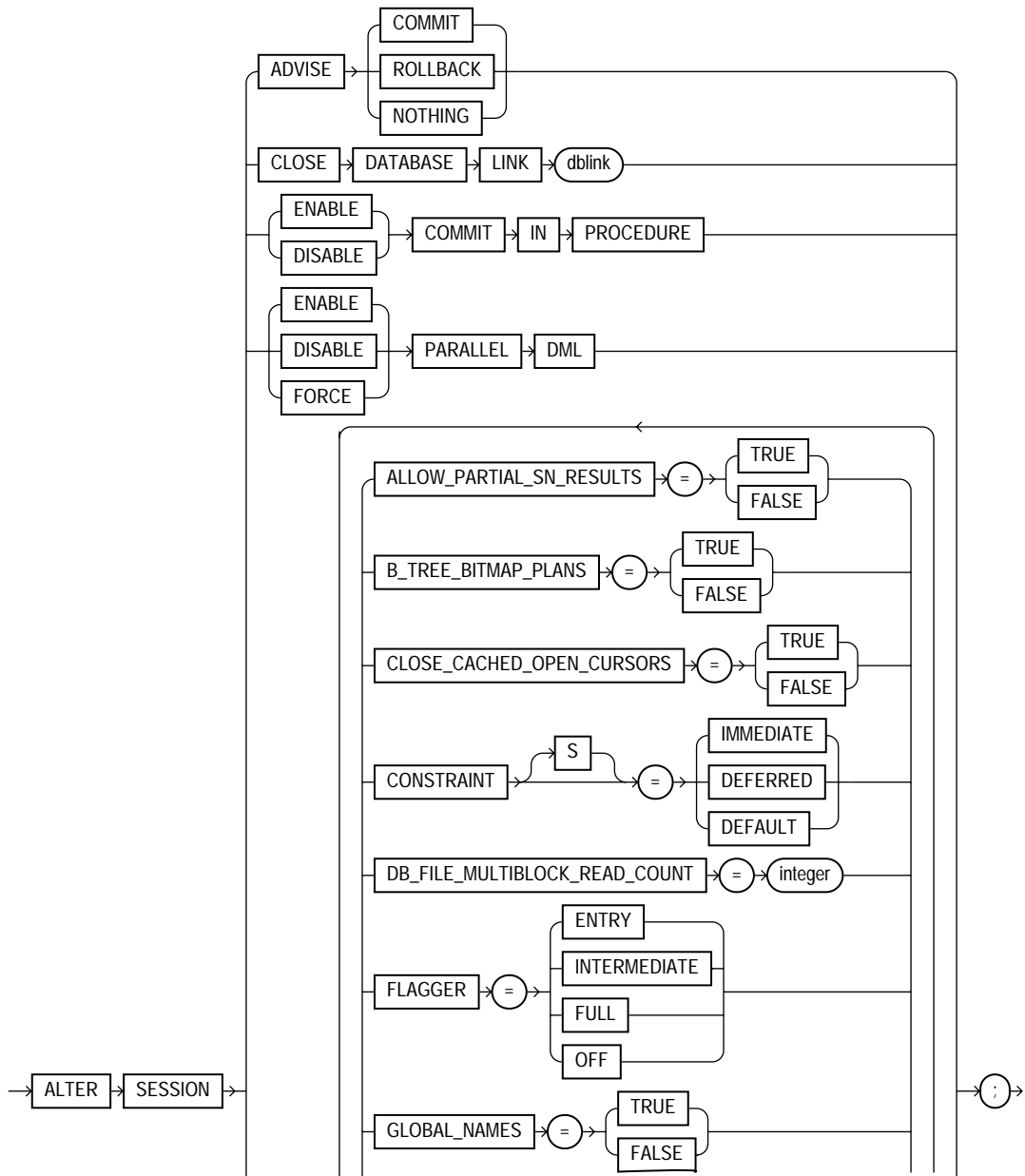
### 前提条件

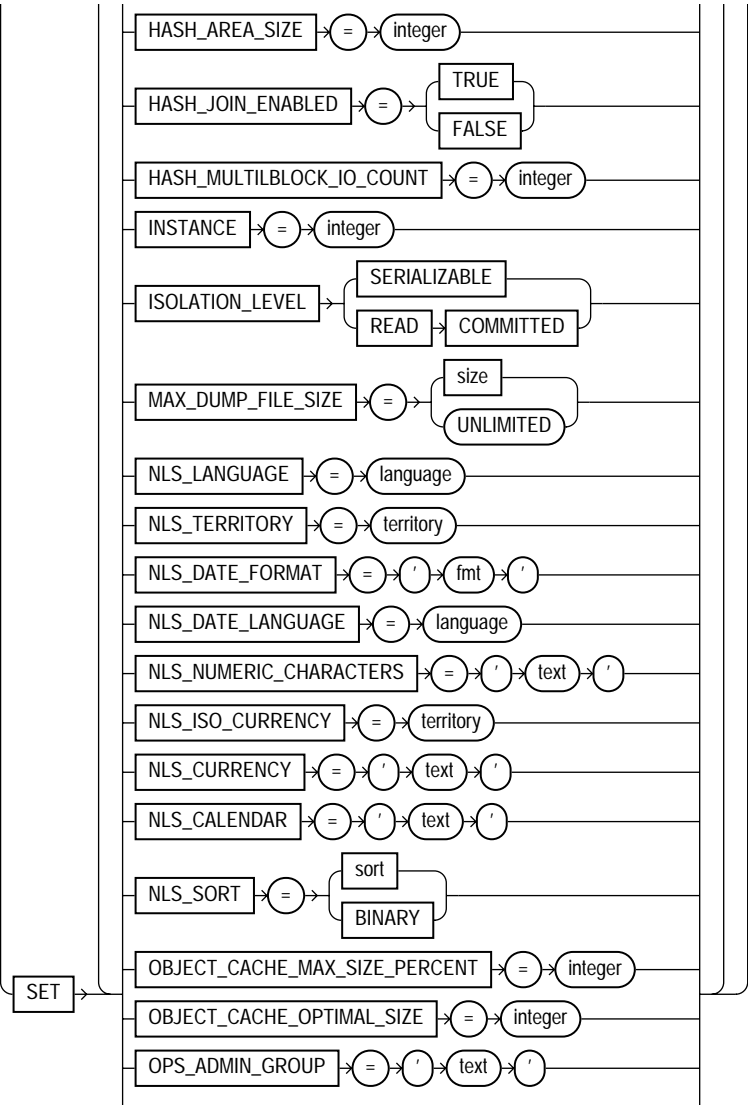
SQL トレース機能を使用可能または使用禁止にしたり、デフォルトのラベル書式を変更するには、ALTER SESSION システム権限が必要です。

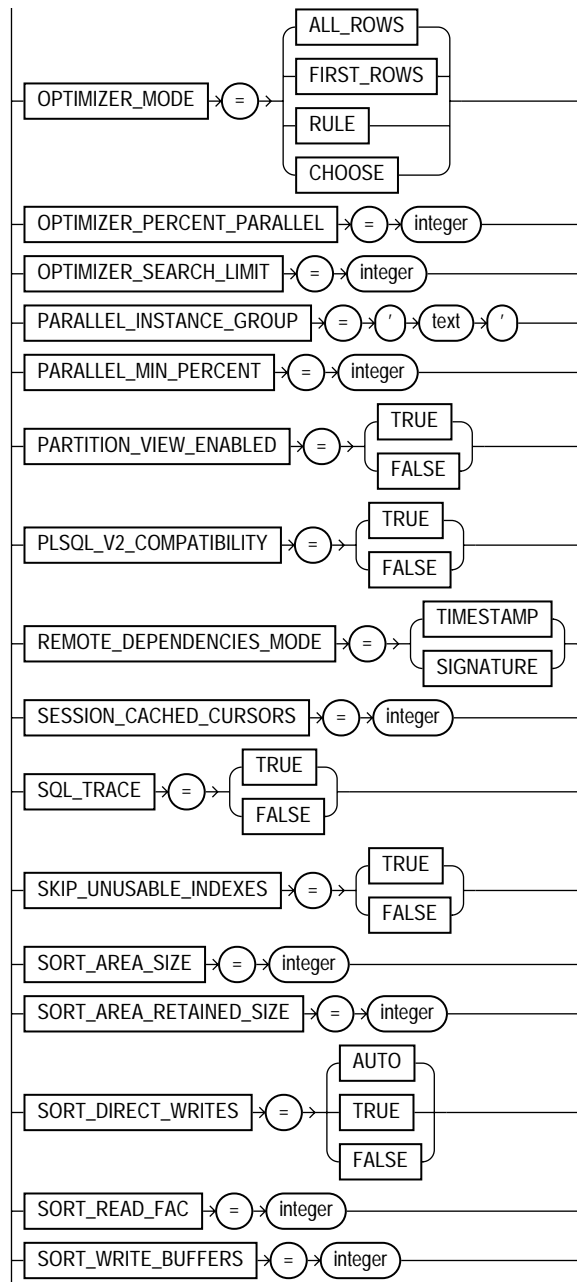
これ以外の操作について権限は不要です。

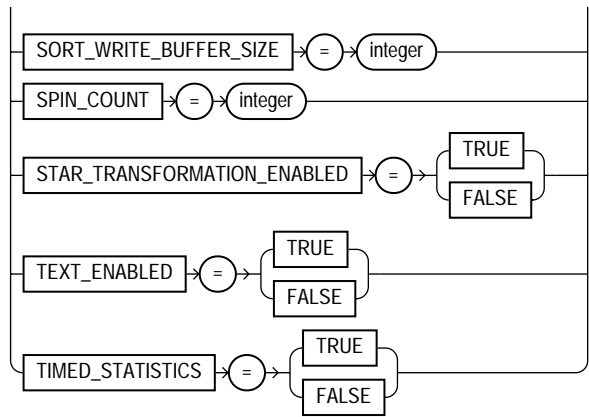


## 構文









キーワードとパラメータ

ADVISE	分散トランザクションを強制処理するためのアドバイスをリモート・データベースに送ります。分散トランザクションがインダウト状態になった場合、リモート・データベース内の DBA_2PC_PENDING データ・ディクショナリ・ビューの ADVICE 列にこのアドバイスが表示されます。（「インダウト分散トランザクションの強制処理」（4-72 ページ）も参照してください。）アドバイス・オプションは次のとおりです。	
	COMMIT	DBA_2PC_PENDING.ADVISE に値 'C' が表示されます。
	ROLLBACK	DBA_2PC_PENDING.ADVISE に値 'R' が表示されます。
	NOTHING	DBA_2PC_PENDING.ADVISE に値 '' が表示されます。
CLOSE	リモート・データベースに対する自セッションの接続を切り離して、データベース・リンク dblink をクローズします。データベース・リンクがアクティブ・トランザクションやオープン・カーソルで使用されているではありません。詳細は、「データベース・リンクのクローズ」（4-72 ページ）を参照してください。	
DATABASE LINK		
COMMIT IN PROCEDURE	ENABLE	プロシージャとストアド・ファンクションによるこれらの文の発行を許可します。
	DISABLE	プロシージャとストアド・ファンクションによるこれらの文の発行を禁止します。
	「プロシージャとストアド・ファンクションのトランザクション制御」（4-73 ページ）を参照してください。	
PARALLEL DML	そのセッションでの後続のすべての DML トランザクションをパラレル実行と見なすかどうかを指定します。（「パラレル DML」（4-74 ページ）を参照。）	

このオプションは、コミットされたトランザクション間だけで実行できます。このコマンドを実行するには、コミットされていないトランザクションをコミットまたはロール・バックする必要があります。

ENABLE	パラレル・ヒントまたはパラレル句が指定されている場合に、セッションの DML 文をパラレル・モードで実行します。
DISABLE	セッションの DML 文をシリアルで実行します。これはデフォルトのモードです。
FORCE	<p>パラレル DML 制限のどれにも違反していない場合に、セッション内の後続の DML 文を強制的にパラレル実行します。パラレル句もパラレル・ヒントも指定されていない場合は、デフォルトの並列性レベル（デフォルトの並列度およびインスタンス数の両方）が使用されます。</p> <p><b>注意：</b> FORCE を使用すると、自動的に、そのセッションで作成される表はすべてデフォルトの並列性レベルで作成されます。結果は、CREATE TABLE 文で（デフォルトの並列度とデフォルトのインスタンス数を使って）パラレル句を指定した場合と同じです。</p>

**SET**

次のセッション・パラメータを設定します。

**CLOSE\_OPEN\_  
CACHED\_  
CURSORS**

PL/SQL によってオープンされ、キャッシュ・メモリに入れられたカーソルを、COMMIT または ROLLBACK のたびに自動的にクローズするかどうかを制御します。

TRUE	COMMIT または ROLLBACK のたびにオープンしているカーソルがクローズされます。
FALSE	PL/SQL によってオープンされたカーソルがオープンされたままになります。これにより、次回以降の実行時にカーソルをオープンする必要がなくなります。

**CONSTRAINT[S]**

遅延可能制約によって指定された条件をいつ適用するかを指定します。

IMMEDIATE	遅延可能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ます。セッションの各トランザクションの始めに SET CONSTRAINTS ALL IMMEDIATE コマンドを発行するのと同じです。SET CONSTRAINT(S) (4-509 ページ) の IMMEDIATE パラメータを参照してください。
DEFERRED	遅延可能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ます。セッションの各トランザクションの始めに SET CONSTRAINTS ALL DEFERRED コマンドを発行するのと同じです。SET CONSTRAINT(S) (4-509 ページ) の DEFERRED パラメータを参照してください。
DEFAULT	各トランザクションの始めに、すべての制約をそれぞれの初期の状態 (DEFERRED または IMMEDIATE) に復元します。

**FLAGGER**

FIPS のフラグ使用を指定します。「FIPS のフラグ使用」(4-71 ページ) を参照してください。

ENTRY	SQL92 Entry レベルのフラグ。
-------	----------------------

	INTERMEDIATE	SQL92 Intermediate レベルのフラグ。
	FULL	SQL92 Full レベルのフラグ。
	OFF	フラグの使用を停止します。
GLOBAL_NAMES		グローバル・ネーム変換の適用を制御します。このパラメータを使用したグローバル・ネーム変換を使用可能にする方法および使用禁止にする方法については、ALTER SYSTEM（4-88 ページ）を参照してください。
	TRUE	グローバル・ネーム変換を使用可能にします。
	FALSE	グローバル・ネーム変換を使用禁止にします。
HASH_JOIN_ENABLED		問合せでのハッシュ結合処理を使用可能または使用禁止にします。デフォルト値は TRUE で、ハッシュ結合が使用可能です。
HASH_AREA_SIZE		ハッシュ結合処理用に使用するメモリー量をバイト単位で指定します。デフォルト値は、初期化パラメータ SORT_AREA_SIZE の値の 2 倍です。
HASH_MULTIBLOCK_IO_COUNT		ハッシュ結合処理時に読み取るまたは書き込むデータ・ブロックの数を指定します。このパラメータの値に初期化パラメータ DB_BLOCK_SIZE の値を掛けた積が 64K 以下になるようにしてください。このパラメータのデフォルト値は 1 です。マルチスレッド・サーバーを使用しているときは値は常に 1 になり、それ以外の値を指定しても無視されます。
INSTANCE		パラレル・サーバーで、セッションが integer に指定したインスタンスに接続する場合と同様に、データベース・ファイルにアクセスします。詳細は、「パラレル・サーバーの別のインスタンスに接続している場合と同様にデータベースにアクセス」（4-71 ページ）を参照してください。
ISOLATION_LEVEL		データベース変更を含むトランザクションがどのように処理されるかを指定します。
	SERIALIZABLE	セッション内のトランザクションは、SQL92 に規定されているように直列可能トランザクション分離モードを使用します。つまり、直列可能トランザクションが行を更新する DML 文を実行する場合、更新対象の行がその直列可能トランザクションの開始時にコミットされていない別のトランザクションによって更新されていたときは、その DML 文は失敗します。直列可能トランザクションは、同一トランザクション内で行った更新を確認できます。SERIALIZABLE モードで運用するには、COMPATIBLE 初期化パラメータを 7.3.0 以上に設定する必要があります。
	READ COMMITTED	セッション内のトランザクションは、Oracle トランザクションのデフォルトの動作を行います。このため、別のトランザクションが保持している行ロックを必要とする DML がトランザクションに含まれていると、DML 文は行ロックが解除されるまで待ち状態になります。
MAX_DUMP_FILE_SIZE		トレース・ダンプ・ファイルのサイズの上限を指定します。最大サイズには、ブロック数を表す負以外の整数、または 'UNLIMITED' を指定します。'UNLIMITED' を指定すると、上限は設定されません。

次の NLS パラメータの詳細は、「NLS パラメータの使用」（4-67 ページ）を参照してください。

NLS_LANGUAGE	<p>Oracle がエラー・メッセージとその他のメッセージを戻す場合に使用する言語を変更します。このパラメータによって次の値も暗黙的に変更されます。</p> <ul style="list-style-type: none"> <li>曜日および月の名前、その他の要素の略称と省略しない形に対する言語</li> <li>ソート基準</li> <li>B.C. および A.D. の指定</li> <li>A.M. および P.M. の指定</li> </ul>
NLS_TERRITORY	<p>次の新しい値が暗黙的に指定されます。</p> <ul style="list-style-type: none"> <li>デフォルトの日付書式</li> <li>小数点文字と桁グループ・セパレータ</li> <li>各国通貨記号</li> <li>ISO 通貨記号</li> <li>D 日付書式要素の週の第 1 日目</li> </ul>
NLS_DATE_FORMAT	<p>新しいデフォルト日付書式を明示的に指定します。'<i>fmt</i>' の値は、「日付書式モデル」(3-65 ページ) に示す日付書式モデルでなければなりません。</p>
NLS_DATE_LANGUAGE	<p>曜日、月の名前、その他の日付書式要素の省略形と省略しない形に対する言語を明示的に変更します。</p>
NLS_NUMERIC_CHARACTERS	<p>新しい小数点文字と桁グループ・セパレータを明示的に指定します。'<i>text</i>' の値は次の形で指定します。</p> <p style="text-align: center;">dg'</p> <p>ここで、<i>d</i> は新しい小数点文字を表し、<i>g</i> は新しい桁グループ・セパレータを表します。</p> <p>小数点文字と桁グループ・セパレータには、シングルスバイト文字を使用してそれぞれ異なる値を指定する必要があります。数値や次の記号は使用できません。"+" プラス記号、 "-" マイナス記号 (またはハイフン)、 "&lt;" 不等号 (より小)、または "&gt;" 不等号 (より大)。</p>
NLS_ISO_CURRENCY	<p>ISO 通貨記号を適用する範囲を明示的に指定します。</p>
NLS_CURRENCY	<p>ローカル通貨記号を明示的に指定します。記号は 10 文字以内です。</p>
NLS_SORT	<p>Oracle が文字値をソートするための言語ソート順序を変更します。</p> <p><i>sort</i>                      言語ソート順序の名前を指定します。</p> <p>BINARY                    バイナリ・ソートを指定します。</p> <p>すべてのキャラクタ・セットについて、デフォルトのソート方式はバイナリです。</p>
NLS_CALENDAR	<p>新しいカレンダーのタイプを明示的に指定します。</p>
OPTIMIZER_MODE	<p>セッションの最適化の方法とモードを指定します。オブティマイザ・モードの詳細は、「最適化の方法とモードの変更」(4-70 ページ) を参照してください。</p>

ALL_ROWS	コストベースの方法を指定し、最大のスループットを達成するため最適化します。
FIRST_ROWS	コストベースの方法を指定し、最も速い応答時間を達成するため最適化します。
RULE	ルールベースの方法を指定します。
CHOOSE	オプティマイザは、データ・ディクショナリの統計情報に基づいて最適化の方法を選択します。
PARTITION_ VIEW_ENABLED	このパラメータを TRUE に設定すると、オプティマイザはパーティション・ビュー内の不要な表のアクセスをスキップします。詳細は、『Oracle8 Server リファレンス・マニュアル』を参照してください。
PLSQL_V2_ COMPATABILITY	コンパイル時の PL/SQL プログラムの動作を変更して、Oracle7(PL/SQL V2)の言語要素では有効で、Oracle8(PL/SQL V3)では無効な言語要素を、使用できるようにします。このセッション・パラメータの詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』および『Oracle8 Server リファレンス・マニュアル』を参照してください。
TRUE	Oracle8 PL/SQL V3 のプログラムが、Oracle7 PL/SQL V2 の要素を実行できるようにします。
FALSE	Oracle7 PL/SQL V2 の無効な要素を使用禁止にします。これはデフォルト値です。
REMOTE_ DEPENDENCIES_ MODE	リモート・ストアド・プロシージャの依存性がこのセッションでどのように処理されるかを指定します。詳細は、『Oracle8 Server アプリケーション開発者ガイド』を参照してください。
SESSION_ CACHED_ CURSORS	頻繁に使われるカーソルを保持するためのセッション・キャッシュのサイズを指定します。 <i>integer</i> に、キャッシュに保持するカーソル数を指定します。このパラメータの詳細は、「セッション・カーソルのキャッシュ」(4-71 ページ)を参照してください。
SKIP_UNUSABLE_INDEXES	表に、使用不可能な索引または索引パーティションがある場合、その表の使用およびレポートを制御します。
TRUE	使用不可のマークが付けられた索引のエラー・レポートを使用禁止にします。使用不可の索引または索引パーティションを持つ表に対する挿入および削除、更新が認められます。
FALSE	使用不可のマークが付けられた索引のエラー・レポートを使用可能にします。使用不可の索引または索引パーティションを持つ表に対する挿入および削除、更新は認められません。これはデフォルト値です。
SQL_TRACE	セッションに対する SQL トレース機能を制御します。「SQL トレース機能の使用可能と使用禁止の切替え」(4-67 ページ)を参照してください。
TRUE	SQL トレース機能を使用可能にします。
FALSE	SQL トレース機能を使用禁止にします。



## SQL トレース機能の使用可能と使用禁止の切替え

SQL トレース機能は SQL 文の処理に関するパフォーマンス統計情報を生成します。初期化パラメータ `SQL_TRACE` を指定することによって、Oracle インスタンス上のすべてのセッションの SQL トレース機能を使用可能または使用禁止にできます。セッションを開始すると、このパラメータの値に基づいて SQL トレース機能が使用可能または使用禁止にされます。後から、`ALTER SESSION` コマンドの `SQL_TRACE` オプションを指定することにより、自セッションの SQL トレース機能を使用可能または使用禁止にできます。

出力の書式設定と解釈の方法など、SQL トレース機能の詳細は、『Oracle8 Server チューニング』を参照してください。

例 1. 次の文は、セッションに対して SQL トレース機能を使用可能にします。

```
ALTER SESSION
SET SQL_TRACE = TRUE;
```

## NLS パラメータの使用

Oracle は、さまざまな国のさまざまな言語での使用をサポートしています。インスタンスの起動時に、"NLS" で始まる初期化パラメータの値に基づいてサポートが提供されます。これらのパラメータについては、『Oracle8 Server リファレンス・マニュアル』を参照してください。ALTER SESSION コマンドの NLS 句を使用すると、自セッションで動的に NLS 属性を変更できます。動的性能表 `V$NLS_PARAMETERS` を問い合わせれば、セッションの現在の NLS 属性を参照できます。次に、それぞれの NLS パラメータの使用方法を説明します。

### エラー・メッセージの言語

`NLS_LANGUAGE` パラメータを使用してエラー・メッセージの言語を変更できます。ただし、このパラメータを使用すると、言語に関連する他の要素も暗黙的に変更されるので注意してください。Oracle は多数のプラットフォーム上での多様な言語のエラー・メッセージを用意しています。

例 2. 次の文は、エラー・メッセージの言語をフランス語に変更します。

```
ALTER SESSION
SET NLS_LANGUAGE = French
```

変更後はフランス語のエラー・メッセージが戻ります。

```
SELECT * FROM emp
ORA-00942: Table ou vue n'existe pas
```

---

---

**注意：** 選択する言語は、先にインストールしておく必要があります。それぞれのオペレーティング・システムでのインストールに関する説明を参照してください。

---

---

## デフォルトの日付書式

新しいデフォルトの日付書式は、NLS\_DATE\_FORMAT パラメータを使用して明示的に指定するか、NLS\_TERRITORY パラメータを使用して暗黙的に指定できます。デフォルトの日付書式モデルについては、「日付書式モデル」(3-65 ページ)を参照してください。

**例 3.** 次の文は、セッションのデフォルトの日付書式を 'YYYY MM DD-HH24:MI:SS' に動的に変更します。

```
ALTER SESSION
SET NLS_DATE_FORMAT = 'YYYY MM DD HH24:MI:SS'
```

変更後は新しい日付書式が次のように適用されます。

```
SELECT TO_CHAR(SYSDATE) Today
FROM DUAL
TODAY
-----
1997 08 12 14:25:56
```

## 月と曜日の言語

月、曜日の名前と省略形を記述する言語は、NLS\_DATE\_LANGUAGE パラメータを使用して明示的に変更するか、または NLS\_LANGUAGE パラメータを使用して暗黙的に変更できます。

**例 4.** 次の文は、日付書式要素の言語をフランス語に変更します。

```
ALTER SESSION
SET NLS_DATE_LANGUAGE = French

SELECT TO_CHAR(SYSDATE, 'Day DD Month YYYY') Today
FROM DUAL

TODAY
-----
Mardi 28 Février 1997
```

## 小数点文字と桁グループ・セパレータ

NLS\_NUMERIC\_CHARACTERS パラメータを使用して明示的に指定するか、または NLS\_TERRITORY パラメータを使用して暗黙的に指定することによって、次の数値書式要素の値を変更できます。

D( 小数点文字 )                      数値の整数部分と小数部分を区切る文字です。

G( 桁グループ・セパレータ    数値の整数部分を複数桁ごとに区切る文字です。  
)

数値書式モデルの使用方法的説明は、「数値書式モデル」(3-61 ページ)を参照してください。

小数点文字と桁グループ・セパレータは、シングलバイト文字でなければならず、同じ文字は使用できません。小数点文字がピリオド(.)でない場合は、SQL 文の式のすべての数値を一重引用符で囲む必要があります。小数点にピリオドを使用しないときには、必ず有効な数値が取り出されるように、常に TO\_NUMBER 関数を使用する必要があります。

**例 5.** 次の文は、小数点文字をカンマ(,)に、桁グループ・セパレータをピリオド(.)に動的に変更します。

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

これらの数値書式要素を使用すると、新しい文字が戻ります。

```
SELECT TO_CHAR( SUM(sal), 'L999G999D99') Total FROM emp ;
```

```
TOTAL
-----
FF29.025,00
```

## ISO 通貨記号

NLS\_ISO\_CURRENCY パラメータを使用して明示的に指定するか、または NLS\_TERRITORY パラメータを使用して暗黙的に指定することによって、C 数値書式要素 (ISO 通貨記号) の値を変更できます。これらのパラメータに地域 (国) を指定すると、その地域 (国) の ISO 通貨記号が C 数値書式要素の値となります。

**例 6.** 次の文では、ISO 通貨記号をアメリカ合衆国の ISO 通貨記号に動的に変更します。

```
ALTER SESSION
SET NLS_ISO_CURRENCY = America;
```

```
SELECT TO_CHAR( SUM(sal), 'L999G999D99') Total
FROM emp;
```

```
TOTAL
-----
USD29,025.00
```

## 各国通貨記号

NLS\_CURRENCY パラメータを使用して明示的に指定するか、または NLS\_TERRITORY パラメータを使用して暗黙的に指定することによって、L 数値書式要素 (各国通貨記号) の値を変更できます。

**例 7.** 次の文は、各国通貨記号を 'DM' に動的に変更します。

```
ALTER SESSION
SET NLS_CURRENCY = 'DM';

SELECT TO_CHAR( SUM(sal), 'L999G999D99') Total
FROM emp;

TOTAL
-----
DM29.025,00
```

### 言語ソート順序

NLS\_SORT パラメータを使用して明示的に指定するか、または NLS\_LANGUAGE パラメータを使用して暗黙的に指定することによって、言語ソート順序またはバイナリ・ソートを指定できます。

**例 8.** 次の文は、言語ソート順序をスペイン語に動的に変更します。

```
ALTER SESSION
SET NLS_SORT = XSpanish;
```

これによって、文字の値はスペイン語のソート順序に基づいてソートされます。

## 最適化の方法とモードの変更

Oracle オプティマイザは、次のどちらかの方法を使用して SQL 文を最適化できます。

- |        |  |
|--------|--|
| コストベース | オプティマイザは、ルールベースの方法で考慮した情報だけでなく、文がアクセスした表、索引、クラスタに関する統計情報を考慮して、SQL 文を最適化します。  |
| ルールベース | オプティマイザは、アクセスされた表に関連付けられた索引とクラスタ、文の構文要素、これらの要素の発見的ランク・リストに基づいて SQL 文を最適化します。 |

コストベースの方法では、オプティマイザは次のいずれかを目標に SQL 文を最適化します。

最大のスループット 文がアクセスしたすべての行を戻すために必要な時間の最小化

最も速い応答時間 文がアクセスしたすべての行を戻すために必要な時間の最小化

最適化の方法は、インスタンスの起動時に、初期化パラメータ OPTIMIZER\_MODE によって設定されます。このパラメータによってコストベースの方法が設定されている場合、デフォルトの目標は最大のスループットとなります。

アプリケーションの特性に基づいてコストベースの方法の目標を選択する方法については、『Oracle8 Server チューニング』を参照してください。

## FIPS のフラグ使用

FIPS のフラグを使用すると、ANSI SQL92 の拡張要素である SQL 文が発行されたときにエラー・メッセージが生成されます。Oracle では、Entry レベル、Intermediate レベル、Full レベルのそれぞれのフラグに違いはありません。セッションでフラグ使用が設定されると、これに続く ALTER SESSION SET FLAGGER コマンドは成功しますが、ORA-00097 のメッセージが生成されます。このため、セッションを切り離さずに FIPS フラグ使用を変更できます。

## セッション・カーソルのキャッシュ

アプリケーションが同一の SQL 文に対して解析を繰り返している場合は、セッション・カーソルの再オープンがパフォーマンスに影響を及ぼすことがあります。ALTER SESSION SET SESSION\_CACHED\_CURSORS コマンドを使用すると、頻繁に使われるセッション・カーソルはクローズされている場合にもセッション・キャッシュに格納できます。この機能は一部の Oracle Tools で特に便利です。たとえば、Oracle Forms のアプリケーションでは、フォームを切り換えるときに、そのフォームに関連しているセッション・カーソルをすべてクローズしますが、この場合、頻繁に使われるカーソルを再解析する必要はありません。

Oracle は共有 SQL 領域を使って、指定された文に対して解析要求が 4 回以上出されたかどうかを判断します。4 回以上出されている場合は、カーソルをセッション・カーソル・キャッシュに移します。同一セッション内で次にその SQL 文の解析が要求されたときには、セッション・カーソル・キャッシュにあるカーソルが使用されます。

初期化パラメータ SESSION\_CACHED\_CURSORS が正の値に設定されている場合、セッション・カーソルは自動的にキャッシュに入れられます。このパラメータは、キャッシュに保持するセッション・カーソルの最大数を指定します。新しいエントリが必要になると、最低使用頻度アルゴリズムによって、キャッシュ内の不要なエントリが選別されて新しいエントリのための空間が確保されます。ALTER SESSION SET SESSION\_CACHED\_CURSORS コマンドを使うと、セッション・カーソルのキャッシュを動的に使用可能にできます。

セッション・カーソル・キャッシュの詳細は、『Oracle8 Server チューニング』を参照してください。

## パラレル・サーバーの別のインスタンスに接続している場合と同様にデータベースにアクセス

パフォーマンスの最適化のために、パラレル・サーバーの各インスタンスはそれぞれ専用のロールバック・セグメントや空きリスト・グループなどを使っています。データベースは通常、1 つのパラレル・サーバー用に設計されています。パラレル・サーバーでは、ユーザーは 1 つのインスタンスに接続し、ユーザーがアクセスするデータは主としてそのユーザー向けにパーティションで分割されています。このインスタンスのユーザーが、別のインスタンスに接続する必要がある場合には、データのパーティション分割はなくなります。ALTER SESSION SET INSTANCE コマンドを使用すると、ユーザーは通常のインスタンスに接続している場合と同様に、別のインスタンスにアクセスできます。

## データベース・リンクのクローズ

データベース・リンクにより、DELETE 文および INSERT 文、LOCK TABLE 文、SELECT 文、UPDATE 文でリモート・データベースにアクセスできます。データベース・リンクを使用する SQL 文を発行すると、Oracle はこのデータベース・リンクを使用してリモート・データベース上にセッションを作成します。この接続は、セッションの終了またはデータベース・リンクの数が初期化パラメータ OPEN\_LINKS の値を超えるまでオープンされています。

セッションでデータベース・リンクを再度使用しない場合は、ALTER SESSION コマンドの CLOSE DATABASE LINK 句を使用してデータベース・リンクを明示的にクローズできます。たとえば、データベース・リンクをオープンしたままにしておくことによってネットワークのオーバーヘッドが生じ、コストが高くなる場合は明示的にクローズします。データベース・リンクをクローズするには、まずデータベース・リンクを使用するすべてのカーソルをクローズし、現行のトランザクションがリンクを使用している場合には、現行のトランザクションを終了する必要があります。

**例.** 次の文は、データベース・リンクを使用している SALES データベース上の employee 表を更新し、このトランザクションをコミットして、データベース・リンクを明示的にクローズします。

```
UPDATE emp@sales
SET sal = sal + 200
WHERE empno = 9001;
COMMIT;
ALTER SESSION
CLOSE DATABASE LINK sales;
```

## インダウト分散トランザクションの強制処理

分散トランザクションのコミット・プロセス中にネットワーク障害またはマシン障害が発生すると、トランザクションの状態が不明、つまりインダウトになる場合があります。この場合、COMMIT または ROLLBACK コマンドの FORCE 句を使用して、このトランザクションにかかわるデータベースに対して、トランザクションを手動でコミットまたはロールバックできます。

分散トランザクションの状態がインダウトになった場合、その分散トランザクションをコミットする前に、ALTER SESSION コマンドの ADVISE 句を使用してリモート・データベースにアドバイスを送信できます。トランザクションの状態がインダウトとなった場合、リモート・データベース内の DBA\_2PC\_PENDING ビューの ADVISE 列にアドバイスが表示されます。そのデータベースの管理者はこの情報によって、リモート・データベースに対するトランザクションをコミットするか、またはロールバックするかを判断できます。分散トランザクションの詳細と、インダウト分散トランザクションをコミットまたはロールバックするかを判断する方法については、『Oracle8 Server 分散システム』を参照してください。

単一のトランザクションにおいて、ADVISE 句を指定した ALTER SESSION 文を複数発行できます。ADVISE 句を指定した文はそれぞれ、ADVISE 句を指定した別の文が発行されるまで、トランザクション内の後続する文で参照されるデータベースに対してアドバイスを送ります。このため、データベースごとに異なるアドバイスを送ることができます。

例. 次の文は、データベース・リンク SITE1 によって識別されるデータベース上の EMP 表に従業員のレコードを挿入し、SITE2 によって識別されるデータベース上の EMP 表から従業員のレコードを削除します。

```
ALTER SESSION
ADVISE COMMIT
```

```
INSERT INTO emp@site1
VALUES (8002, 'FERNANDEZ', 'ANALYST', 7566,
TO_DATE('04-OCT-1992', 'DD-MON-YYYY'), 3000, NULL, 20)
```

```
ALTER SESSION
ADVISE ROLLBACK;
DELETE FROM emp@site2
WHERE empno = 8002;
COMMIT;
```

このトランザクションには ADVISE 句を指定した ALTER SESSION 文が2つあります。このトランザクションが状態不明（インダウト）になった場合、SITE1 には、最初に指定した ALTER SESSION 文によって 'COMMIT' が送信され、SITE2 には、2 番目の指定によって 'ROLLBACK' が送信されます。

## プロシージャとストアド・ファンクションのトランザクション制御

プロシージャとストアド・ファンクションは PL/SQL で記述されるため、COMMIT 文と ROLLBACK 文を発行できます。アプリケーションが行うレコード管理が、アプリケーション自体が直接出したのではない COMMIT 文や ROLLBACK 文によって中断されるような場合は、セッション中にコールされるプロシージャやストアド・ファンクションによってこれらの文が出されないように制御できます。その場合には、次の文を指定します。

```
ALTER SESSION DISABLE COMMIT IN PROCEDURE;
```

続いて、COMMIT または ROLLBACK 文を実行するプロシージャやストアド・ファンクションをコールすると、エラーが戻り、そのトランザクションはコミットもロールバックもされません。

後で、次の文を実行することにより、プロシージャとストアド・ファンクションが COMMIT 文や ROLLBACK 文を出せるようになります。

```
ALTER SESSION ENABLE COMMIT IN PROCEDURE;
```

このコマンドはデータベース・トリガーには適用されません。トリガーでは COMMIT 文も ROLLBACK 文も出せないためです。

---

---

**注意：** 一部のアプリケーション (SQL\*Forms など) は自動的にプロシージャとストアド・ファンクションでの COMMIT 文や ROLLBACK 文を禁止します。使用しているアプリケーションのマニュアルを参照してください。

---

---

## パラレル DML

そのセッションについてパラレル DML を使用可能にすると、発せられた文のすべての DML 部分がパラレル実行の対象とみなされます。しかし、パラレル DML を使用可能にしても、一部の DML 処理のパラレル化には制約があり、また、パラレル・ヒントとパラレル句が指定されなければ引き続きシリアルで実行される DML 処理もあります。パラレル DML の機能とヒントの詳細は、『Oracle8 Server チューニング』を参照してください。

パラレル DML 操作には次の制限が適用されます。

- クラスタ化表に対する DML 操作はパラレル化されない。
- データベースまたはパッケージ状態を読み書きする埋込み関数を使用した DML 操作は、パラレル化されない。
- 起動される可能性のあるトリガーを使用した表に対する DML 操作は、パラレル化されない。
- オブジェクト型 LONG または LOB データ型が含まれている表あるいはスキーマ・オブジェクトに対する DML 操作は、パラレル化されない

パラレル DML モードは、コミットされたトランザクション間だけで変更できます。このコマンドをコミットされていないトランザクションに続けて発すると、エラーが生成されます。コミットされていないトランザクションは、ALTER SESSION ENABLE | DISABLE PARALLEL DML コマンドを発する前に、コミットまたはロール・バックする必要があります。

**例 1.** 現行のセッションのパラレル DML モードを使用可能にするには、次の文を発します。

```
ALTER SESSION ENABLE PARALLEL DML;
```

**例 2.** 次の例では、各 DML 文の実行直後にすべての遅延可能制約をチェックするように、現行のセッションを変更します。

```
ALTER SESSION SET CONSTRAINTS IMMEDIATE;
```

**例 3.** 次の文では、使用不可のマークが付けられたローカル索引パーティションへの挿入を認めるように、現行のセッションを変更します。

```
ALTER SESSION SET SKIP_UNUSABLE_INDEXES=TRUE;
```



## 関連項目

ALTER SESSION (4-58 ページ)

SET CONSTRAINT(S) (4-509 ページ)

『PL/SQL ユーザーズ・ガイドおよびリファレンス』

『Oracle8 Server リファレンス・マニュアル』

『Oracle8 Server チューニング』

# ALTER SNAPSHOT

---

## 用途

次のいずれかの方法でスナップショットを変更します。

- 記憶特性を変更する。
- 自動リフレッシュのモードと時期を変更する。

これらの用途の一部を、「例」（4-81 ページ）に示します。

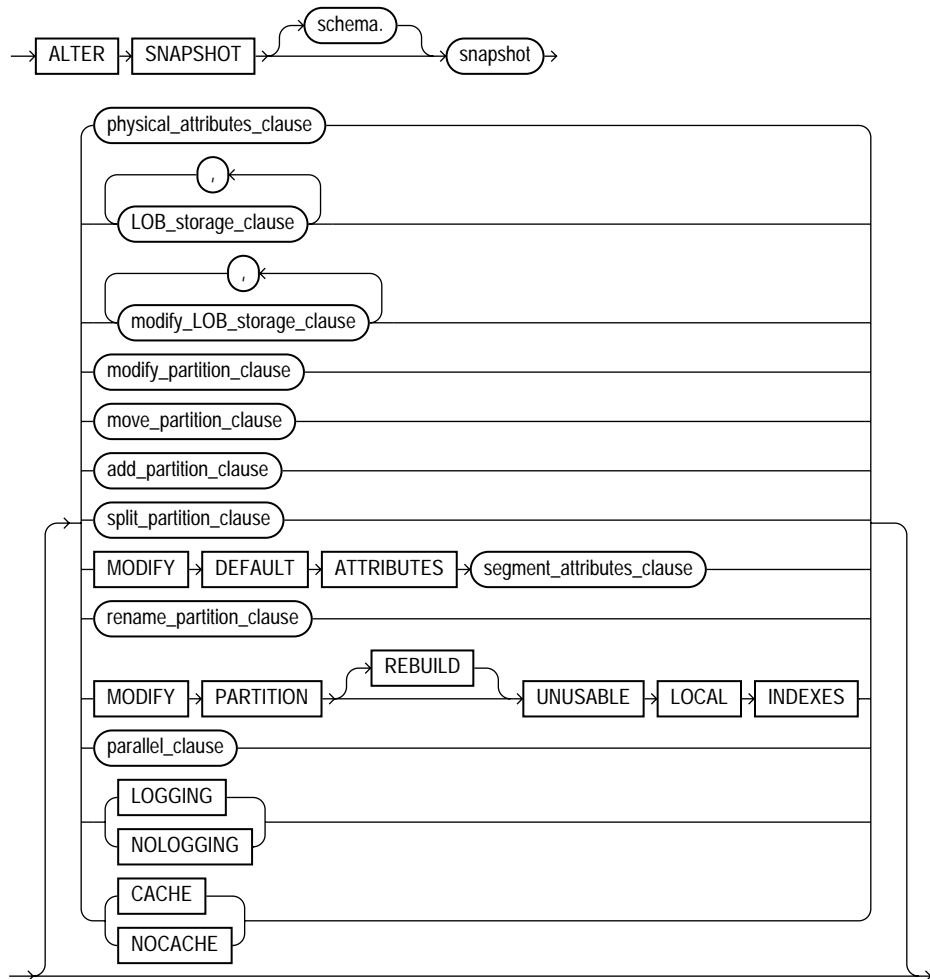
スナップショットのリフレッシュ方法など、スナップショットの詳細は、「CREATE SNAPSHOT」（4-283 ページ）を参照してください。

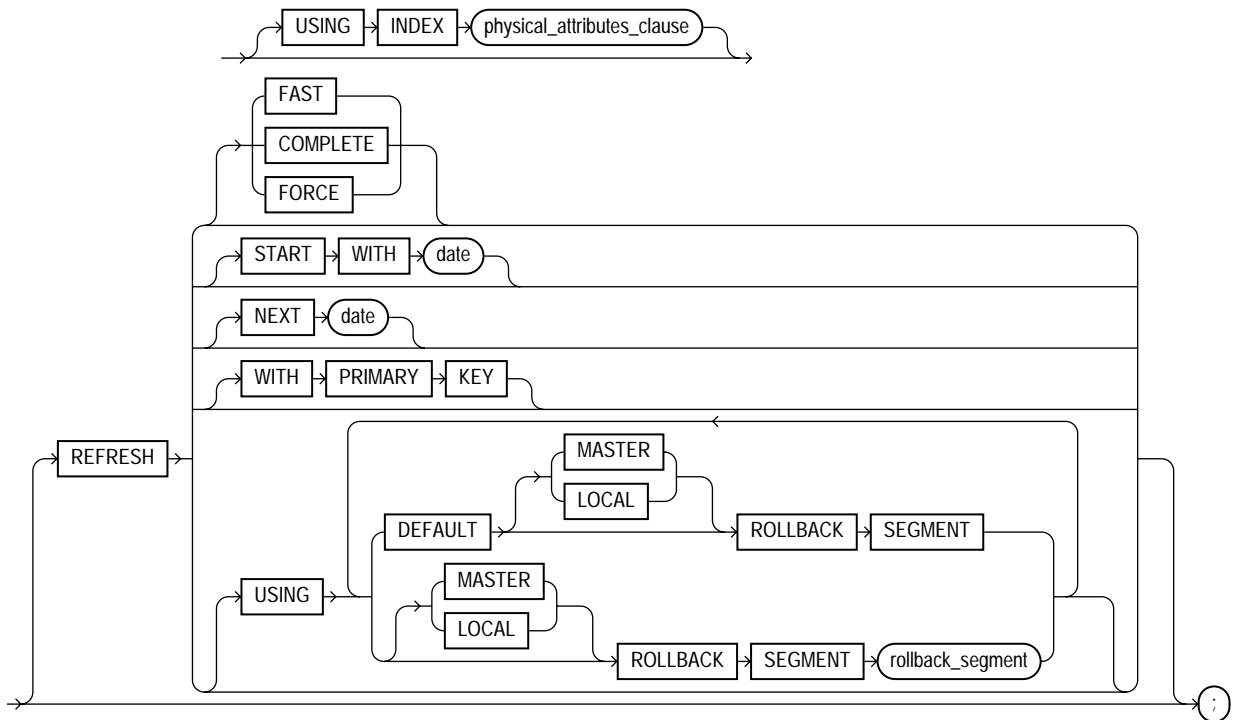
## 前提条件

スナップショットの記憶領域パラメータを変更するには、そのスナップショットが、自スキーマ内にあるか、ALTER ANY SNAPSHOT システム権限を持っている必要があります。

ALTER SNAPSHOT の前提条件の詳細は、『Oracle8 Server レプリケーション』を参照してください。

## 構文





`parallel_clause`:`PARALLEL` 句（4-462 ページ）を参照。

`storage_clause`:`STORAGE` 句（4-518 ページ）を参照。

次の句の構文については、`ALTER TABLE`（4-105 ページ）を参照してください。

- `physical_attributes_clause`
- `LOB_storage_clause`
- `modify_LOB_storage_clause`
- `modify_partition_clause`
- `move_partition_clause`
- `add_partition_clause`
- `split_partition_clause`
- `modify_default_attributes_clause`

- `rename_partition_clause`

## キーワードとパラメータ

<i>schema</i>	変更するスナップショットが定義されているスキーマです。スキーマを指定しないと、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。
<i>snapshot</i>	変更するスナップショットの名前です。
<i>modify_default_attributes</i>	パーティション表のデフォルト属性に新しい値を指定します。この句のパラメータの指定については、ALTER TABLE (4-105 ページ) を参照してください。
<i>physical_attributes_clause</i>	スナップショットのデータをメンテナンスするために使用される内部表の PCTFREE および PCTUSED、INITRANS、MAXTRANS の各パラメータの値および記憶特性を変更します。詳細は、CREATE TABLE (4-303 ページ) および STORAGE 句 (4-518 ページ) を参照してください。
LOGGING/ NOLOGGING	ロギング属性を指定します。このオプションの指定については、ALTER TABLE (4-105 ページ) 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、全表走査の実行時に、そ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入れるかどうかを指定します。このオプションは、小規模な参照表の場合に便利です。このオプションの指定については ALTER TABLE (4-105 ページ) を参照してください。
<i>LOB_storage_clause</i>	LOB 記憶特性を指定します。この句のパラメータの指定については、ALTER TABLE (4-105 ページ) を参照してください。
<i>modify_LOB_storage_clause</i>	LOB 属性 <i>lob_item</i> または LOB オブジェクト属性の物理属性を変更します。この句のパラメータの指定については、ALTER TABLE (4-105 ページ) を参照してください。
パーティション分割に関連した次の句の詳細は、「パーティション・スナップショット」(4-83 ページ) を参照してください。	
<i>modify_partition_clause</i>	表パーティションの実物理属性を変更します。この句のパラメータの指定については、ALTER TABLE (4-105 ページ) を参照してください。
<i>move_partition_clause</i>	表パーティション <i>partition_name</i> を別のセグメントに移動します。この句のパラメータの指定については、ALTER TABLE (4-105 ページ) を参照してください。
<i>add_partition_clause</i>	新しいパーティション <i>new_partition_name</i> をパーティション表の "一番上" に追加します。この句のパラメータの指定については、ALTER TABLE (4-105 ページ) を参照してください。
<i>split_partition_clause</i>	2つの新しいパーティションを作成し、セグメント属性および物理属性、初期エクステントを各パーティションに新たに指定します。この句のパラメータの指定については、ALTER TABLE (4-105 ページ) を参照してください。
<i>rename_partition_clause</i>	表パーティション <i>partition_name</i> を <i>new_partition_name</i> に改名します。この句のパラメータの指定については、ALTER TABLE (4-105 ページ) を参照してください。

---

*parallel\_clause* スナップショットの並列度を指定します。「PARALLEL 句」(4-462 ページ)を参照してください。この句がマスター表用に設定されると、スナップショットの作成およびリフレッシュのパフォーマンスが向上します(スナップショット定義問合せにより異なります)。

## MODIFY PARTITION UNUSABLE LOCAL INDEXES

*partition\_name* に対応付けられているすべてのローカル索引パーティションに使用不可のマークを付けます。

## MODIFY PARTITION REBUILD UNUSABLE LOCAL INDEXES

*partition\_name* に対応付けられている使用不可のローカル索引パーティションを再構築します。

**USING INDEX** スナップショットのデータをメンテナンスするために使用される索引の **INITRANS** パラメータおよび **MAXTRANS** パラメータ、**STORAGE** パラメータの値を変更します。**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われます。

**REFRESH** 自動リフレッシュのモードと日時を変更します。

**FAST** 高速リフレッシュ、つまりマスター表に対応付けられたスナップショット・ログを使用するリフレッシュを指定します。

**COMPLETE** 完全リフレッシュ、つまり各リフレッシュ時にスナップショットを再作成するリフレッシュを指定します。

**FORCE** 高速リフレッシュが可能な場合には高速リフレッシュを、そうでない場合には完全リフレッシュを指定します。高速リフレッシュが可能かどうかは、リフレッシュ時に Oracle によって決定されます。

**FAST** および **COMPLETE**、**FORCE** のいずれのオプションも指定しないと、デフォルトでは **FORCE** が採用されます。

**START WITH** 次の自動リフレッシュ日時のための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新たな日付式を指定します。

**START WITH** と **NEXT** には、未来の日付を指定しなければなりません。

**WITH PRIMARY KEY** **ROWID** スナップショットを主キー・スナップショットに変更します。主キーのスナップショットを使用すると、高速リフレッシュを継続するスナップショットの機能に影響を与えずに、スナップショット・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なければなりません。「主キー・スナップショット」(4-82 ページ)を参照してください。

**USING MASTER ROLLBACK SEGMENT** スナップショットのリフレッシュ時に使用するリモートのマスター・ロールバック・セグメントを変更します。*rollback\_segment* に、使用するロールバック・セグメントの名前を指定します。(ローカルのスナップショット・ロールバック・セグメントを変更するには、『Oracle8 Server レプリケーション』で説明している **DBMS\_REFRESH** パッケージを使用します。)  
「ロールバック・セグメントの指定」(4-82 ページ)も参照してください。

**DEFAULT** Oracle がどのロールバック・セグメントを選択して使用するかを指定します。**DEFAULT** を指定した場合、*rollback\_segment* は指定できません。

MASTER	リモート・マスターで個々のスナップショット用に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	スナップショットが属しているローカル・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。

## 例

**例 1.** 次の文は、HQ\_EMP スナップショットの自動リフレッシュ・モードを FAST に変更します。

```
ALTER SNAPSHOT hq_emp
REFRESH FAST;
```

これからのスナップショットの自動リフレッシュは高速リフレッシュとなります。これは単純なスナップショットであり、そのマスター表はスナップショットの作成前または最後のリフレッシュ前に作成されたスナップショット・ログを持っています。

REFRESH 句に START WITH と NEXT の値が指定されていないため、HQ\_EMP スナップショットの作成時またはその最後の変更時に REFRESH 句に設定された間隔がそのまま使用されます。

**例 2.** 次の文は、BRANCH\_EMP スナップショットの自動リフレッシュ間隔を変更します。

```
ALTER SNAPSHOT branch_emp
REFRESH NEXT SYSDATE+7;
```

REFRESH 句に START WITH の値が指定されていないため、BRANCH\_EMP スナップショットの作成時またはその最後の変更時に START WITH と NEXT の値によって設定された日時に次の自動リフレッシュが行われます。

このスナップショットは、次に自動リフレッシュが行われるときに、リフレッシュされます。NEXT に設定した式 SYSDATE+7 が計算されて次に自動リフレッシュが行われる日時が決定されます。以降は週に 1 度リフレッシュが自動的に行われます。

REFRESH 句に、リフレッシュ・モードが明示的に指定されていないため、直前の CREATE SNAPSHOT または ALTER SNAPSHOT 文の REFRESH 句で指定されたリフレッシュ・モードが引き続き使用されます。

**例 3.** 次の文は、SF\_EMP スナップショットの新しいリフレッシュ・モード、次のリフレッシュ日時と新しい自動リフレッシュ間隔を指定します。

```
ALTER SNAPSHOT sf_emp
REFRESH COMPLETE
START WITH TRUNC(SYSDATE+1) + 9/24
NEXT SYSDATE+7;
```

START WITH 句に指定した値によって、このスナップショットの次の自動リフレッシュは翌日の午前 9 時に発生するように設定されます。この日時にスナップショットの高速リフレッシュが実行され、NEXT に設定した式が計算されて、その後このスナップショットは毎週リフレッシュされます。

## ロールバック・セグメントの指定

リフレッシュ時にマスター・サイトとローカル・サイトの両方に使用するロールバック・セグメントを指定できます。マスター・ロールバック・セグメントは、スナップショットごとに保存され、スナップショットの作成およびリフレッシュ時に妥当性検査が行われます。スナップショットが複合の場合は、マスター・ロールバック・セグメントが無視されます（ただし、指定されている場合）。

ローカルのスナップショット・ロールバック・セグメントは、DBMS\_REFRESH パッケージを使って変更でき、リフレッシュ・グループ・レベルで保存されます。DBMS\_REFRESH パッケージについては、『Oracle8 Server レプリケーション』を参照してください。自動リフレッシュ・パラメータ (START WITH および NEXT) を指定すると、新しいリフレッシュ・グループが自動的に作成され、スナップショットがバックグラウンド・プロセスでリフレッシュされます。ローカルのロールバック・セグメントが指定されている場合は、この新しいリフレッシュ・グループと関連付けられます。自動リフレッシュ・パラメータを指定せずに、ローカルのロールバック・セグメントを指定すると、エラーが発生します。

---

---

**注意：** CREATE SNAPSHOT または ALTER SNAPSHOT を使用してロールバック・セグメントを指定した後に、このロールバック・セグメントが自動的に選択されるように設定するには、ALTER SNAPSHOT で DEFAULT オプションを指定します。

---

---

**例 1.** 次の例は、スナップショット・リフレッシュ時に使用されるリモートのマスター・ロールバック・セグメントを MASTER\_SEG に変更します。

```
ALTER SNAPSHOT inventory
  REFRESH USING MASTER ROLLBACK SEGMENT master_seg;
```

**例 2.** 次の例は、スナップショット・リフレッシュ時に使用されるリモートのマスター・ロールバック・セグメントを Oracle によって選択されたものに変更します。

```
ALTER SNAPSHOT sales REFRESH USING DEFAULT MASTER ROLLBACK SEGMENT;
```

## 主キー・スナップショット

ROWID スナップショットを主キー・スナップショットに変更するには、次の作業が必要です。

- 主キーのすべての列をスナップショット定義に加えます。
- 使用可能になった主キー制約をスナップショット・マスターで定義します。



- 更新可能なスナップショット・ログが、すべての更新可能なスナップショット用に空であることを確認します。

主キー・スナップショットを高速リフレッシュするには、まず **WITH PRIMARY KEY** を指定してスナップショット・マスター・ログを作成する必要があります。また、スナップショット・マスター・ログには **ROWID** を保存できます。スナップショットが高速リフレッシュ用にログを使用できるように、スナップショットを作成する前にスナップショット・マスター・ログを作成してください。

主キー・スナップショットの詳細は、『Oracle8 Server レプリケーション』を参照してください。

---

---

**注意：** 主キー・スナップショットは **ROWID** スナップショットに変更できません。この場合、主キー・スナップショットを削除し、それを **ROWID** スナップショットとして作成し直す必要があります。

---

---

**例 1.** 次の例は、**ROWID** を主キー・スナップショットに変更します。

```
ALTER SNAPSHOT emp_rs REFRESH WITH PRIMARY KEY;
```

## パーティション・スナップショット

スナップショットは基本的には表なので、パーティション・スナップショットはパーティション表と同じです。オプションの構文と方法は、**CREATE TABLE** および **ALTER TABLE** のパーティション表オプションと同じです。ただ 1 つ異なる点は、スナップショットおよびスナップショット・ログについては、次の操作を実行できないことです。

- **DROP PARTITION**
- **TRUNCATE PARTITION**
- **EXCHANGE PARTITION**

マスター表のパーティションを削除または切り捨てることによる大量削除は、実行できません。このため、パーティションを削除または切り捨てた後で、すべてのスナップショットを手動でリフレッシュしなければなりません。高速リフレッシュにより、正しくない結果が生成される可能性があります、Oracle はエラーを出しません。

## 関連項目

**CREATE SNAPSHOT** (4-283 ページ)

**DROP SNAPSHOT** (4-399 ページ)

**STORAGE 句** (4-518 ページ)

# ALTER SNAPSHOT LOG

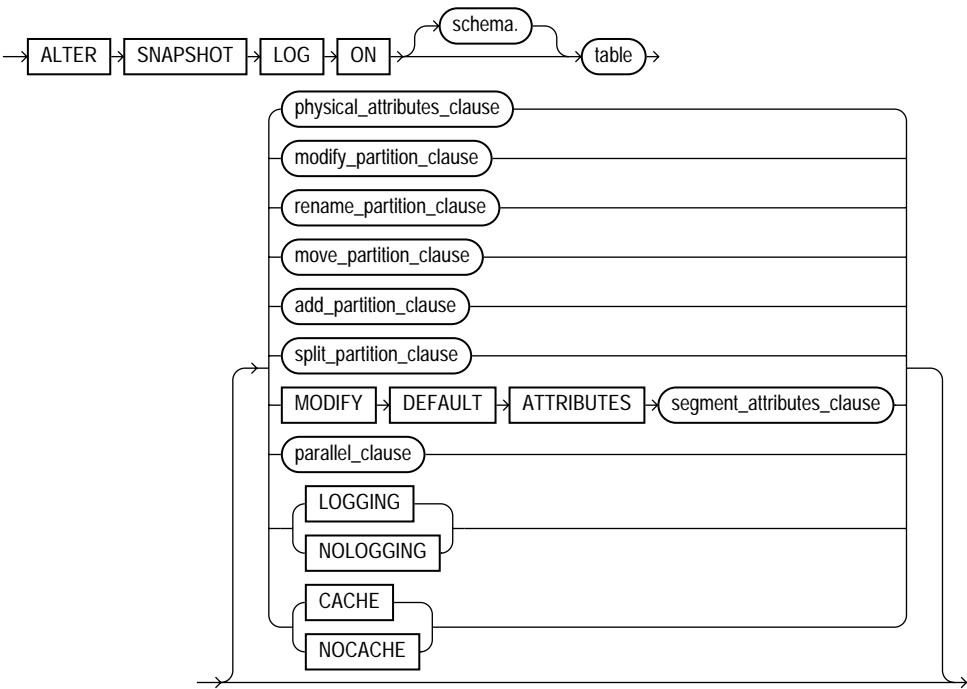
## 用途

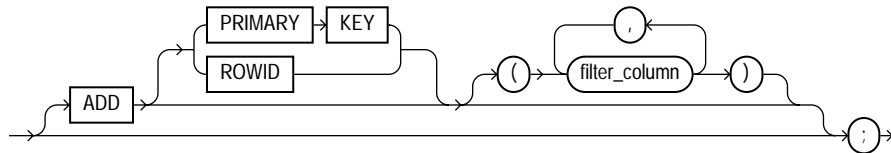
スナップショット・ログの記憶特性を変更します。スナップショット・ログの詳細は、CREATE SNAPSHOT（4-283 ページ）を参照してください。

## 前提条件

マスター表の所有者またはマスター表に対して SELECT 権限をもつユーザーだけが、スナップショット・ログを変更できます。ALTER SNAPSHOT LOG の前提条件の詳細は、『Oracle8 Server レプリケーション』を参照してください。

## 構文





次の句の構文については、ALTER TABLE（4-105 ページ）を参照してください。

- `physical_attributes_clause`
- `modify_partition_clause`
- `rename_partition_clause`
- `move_partition_clause`
- `add_partition_clause`
- `split_partition_clause`
- `modify_default_attributes`

## キーワードとパラメータ

<i>schema</i>	マスター表が定義されているスキーマです。 <i>schema</i> を指定しないと、スナップショット・ログが自スキーマ内にあるとみなされます。
<i>table</i>	変更するスナップショット・ログに関連付けられたマスター表の名前です。
<i>physical_attributes_clause</i>	表またはパーティション、オーバーフロー・データ・セグメントに対する PCTFREE および PCTUSED、INTRANS、MAXTRANS パラメータの値、あるいはパーティション表のデフォルト特性を変更します。CREATE TABLE（4-303 ページ）の PCTFREE および PCTUSED、INTRANS、MAXTRANS パラメータを参照してください。「物理属性の変更」（4-86 ページ）にある例を参照してください。
<i>rename_partition_clause</i>	表パーティション <i>partition_name</i> を <i>new_partition_name</i> に改名します。この句のパラメータの指定については、ALTER TABLE（4-105 ページ）を参照してください。
<i>modify_partition_clause</i>	表パーティションの実物理属性を変更します。この句のパラメータの指定については、ALTER TABLE（4-105 ページ）を参照してください。
<i>move_partition_clause</i>	表パーティション <i>partition_name</i> を別のセグメントに移動します。この句のパラメータの指定については、ALTER TABLE（4-105 ページ）を参照してください。
<i>add_partition_clause</i>	新しいパーティション <i>new_partition_name</i> をパーティション表の "一番上" に追加します。この句のパラメータの指定については、ALTER TABLE（4-105 ページ）を参照してください。

<i>split_partition_clause</i>	2つの新しいパーティションを作成し、セグメント属性および物理属性、初期エクステントを各パーティションに新たに指定します。この句のパラメータの指定については、ALTER TABLE（4-105 ページ）を参照してください。  詳細は、「パーティション・スナップショット・ログ」（4-87 ページ）を参照してください。
<i>modify_default_attributes_clause</i>	パーティション索引だけに使用できるオプションです。このオプションを使って、パーティション索引のデフォルト属性に新しい値を指定できます。
<i>parallel_clause</i>	スナップショットの並列度を指定します。PARALLEL 句（4-462 ページ）を参照してください。この句がマスター表用に設定されると、スナップショットの作成中およびリフレッシュ中のパフォーマンスが改善されます（スナップショット定義問合せにより異なります）。
LOGGING/ NOLOGGING	ロギング属性を指定します。このオプションの指定については ALTER TABLE（4-105 ページ）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、全表走査の実行時に、そ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入れるかどうかを指定します。このオプションは、小規模な参照表で有効です。このオプションの指定については ALTER TABLE（4-105 ページ）を参照してください。
ADD	スナップショット・マスター表内の行が更新されるときに、主キー値または ROWID 値も記録するようにスナップショット・ログを変更します。また、この句は、新たにフィルタ列を記録するためにも使用できます。  PRIMARY KEY      更新されるすべての行の主キーをスナップショット・ログに記録するように指定します。  ROWID              更新されるすべての行の ROWID 値をスナップショット・ログに記録するように指定します。  <i>filter_column(s)</i> スナップショットによって参照される主キー以外の列です。フィルタ列については、『Oracle8 Server レプリケーション』を参照してください。  詳細は、「主キーおよび ROWID、フィルタ列の追加」（4-86 ページ）を参照してください。

物理属性の変更

例 . 次の文は、スナップショット・ログの MAXEXTENTS の値を変更します。

```
ALTER SNAPSHOT LOG ON dept
STORAGE MAXEXTENTS 50;
```

主キーおよび ROWID、フィルタ列の追加

スナップショット・ログは、スナップショット・マスター表が更新されるときに、追加で主キーまたは ROWID、フィルタ列の情報が記録されるように変更できます。これらの情報の記録を停止するには、まずスナップショット・ログを削除してから、再作成する必要があります。

例 . 次の例は、主キー情報も記録するように既存の ROWID スナップショット・ログを変更します。

```
ALTER SNAPSHOT LOG ON sales ADD PRIMARY KEY;
```

## パーティション・スナップショット・ログ

スナップショット・ログは基本的には表なので、パーティション・スナップショット・ログはパーティション表と同じです。オプションの構文と方法は、**CREATE TABLE** および **ALTER TABLE** のパーティション表オプションと同じです。ただ 1 つ異なる点は、スナップショットおよびスナップショット・ログについては、次の操作を実行できないことです。

- **DROP PARTITION**
- **TRUNCATE PARTITION**
- **EXCHANGE PARTITION**

マスター表のパーティションの削除または切捨てによる大量削除は、実行できません。このため、パーティションを削除または切り捨てた後は、すべてのスナップショットを手動でリフレッシュしなければなりません。高速リフレッシュにより、正しくない結果が生成される可能性があります、Oracle はエラーを出しません。

## 関連項目

**ALTER TABLE** (4-105 ページ)

**CREATE SNAPSHOT** (4-283 ページ)

**CREATE SNAPSHOT LOG** (4-294 ページ)

**DROP SNAPSHOT LOG** (4-400 ページ)

## ALTER SYSTEM

---

### 用途

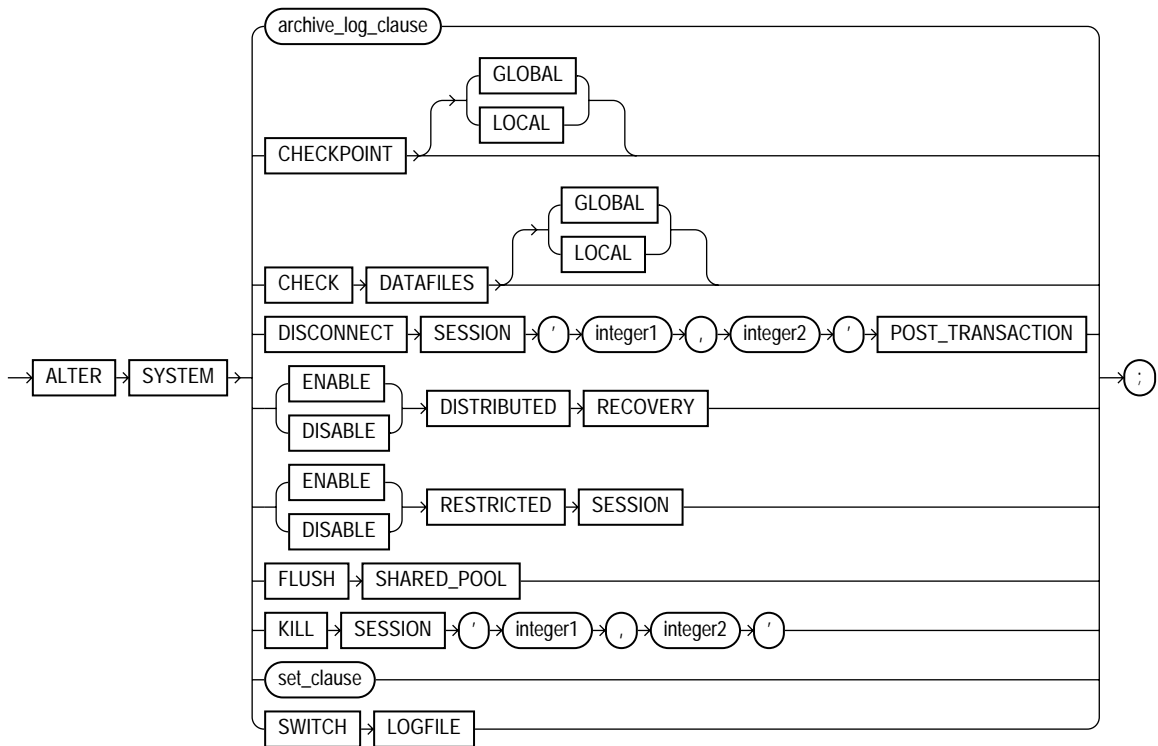
次のいずれかの方法で、ユーザーの Oracle インスタンスを動的に変更します。

- チェックポイントを明示的に実行する。
- データ・ファイルへのアクセスを検査する。
- セッションを終了する。
- リソース制限を使用可能または使用禁止にする。
- シングル・プロセス環境における分散回復を使用可能にする。
- 分散回復を使用禁止にする。
- Oracle へのログインを RESTRICTED SESSION システム権限が付与されているユーザーに制限する。
- システム・グローバル領域 (SGA) の共有プール上のデータをすべて消去する。
- グローバル・ネーム変換を使用可能または使用禁止にする。
- 同時使用ライセンスおよび名前付きユーザー・ライセンスに対する、制限やしきい値を動的に変更する、または使用禁止にする。
- REDO ログ・ファイル・グループを手動でアーカイブする、または自動アーカイブを使用可能にする、使用禁止にする。
- マルチスレッド・サーバー・アーキテクチャ用の共有サーバー・プロセスまたはディスパッチャ・プロセスを管理する。
- REDO ログ・ファイル・グループを明示的に切り替える。

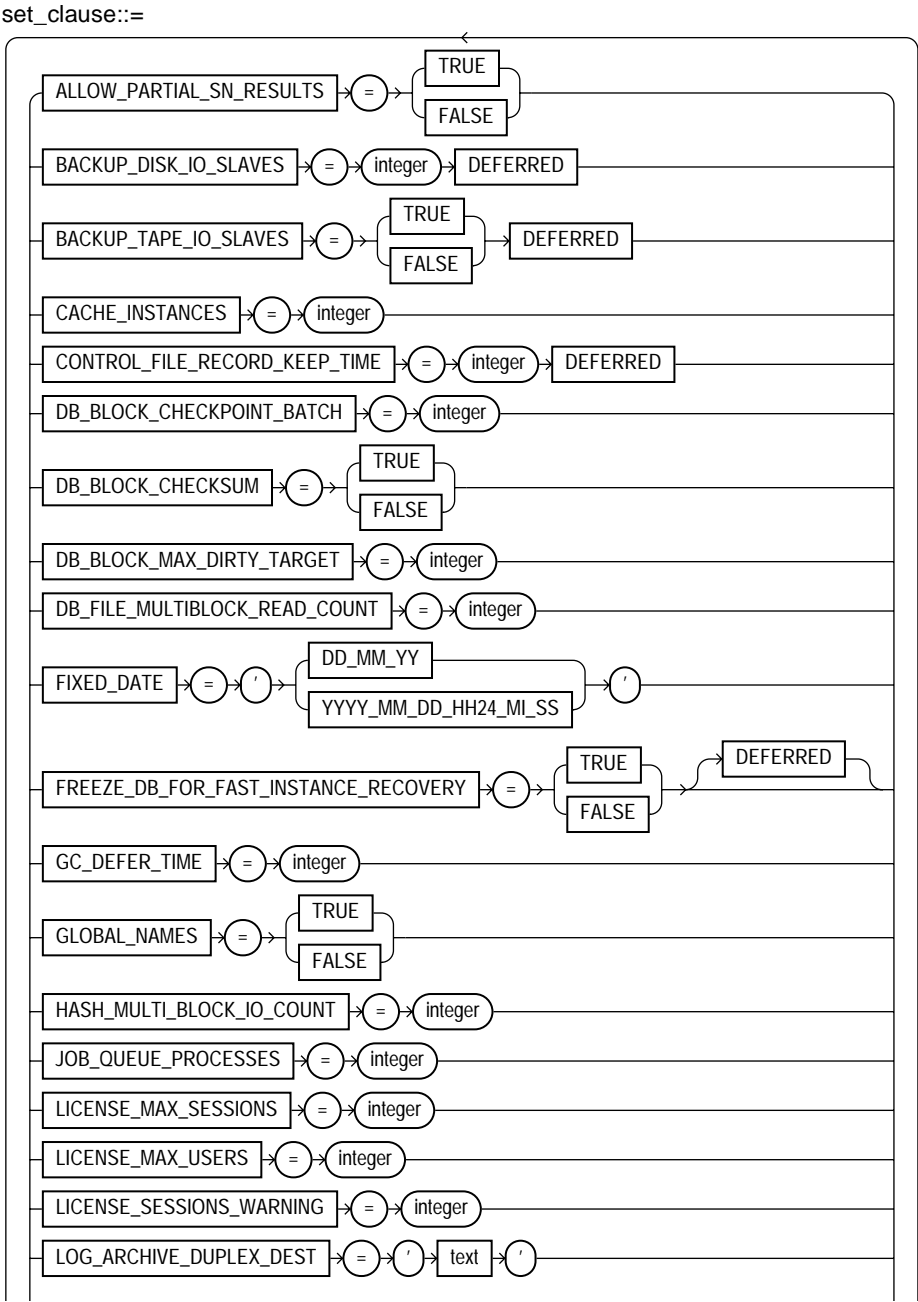
### 前提条件

ALTER SYSTEM 権限が必要です。

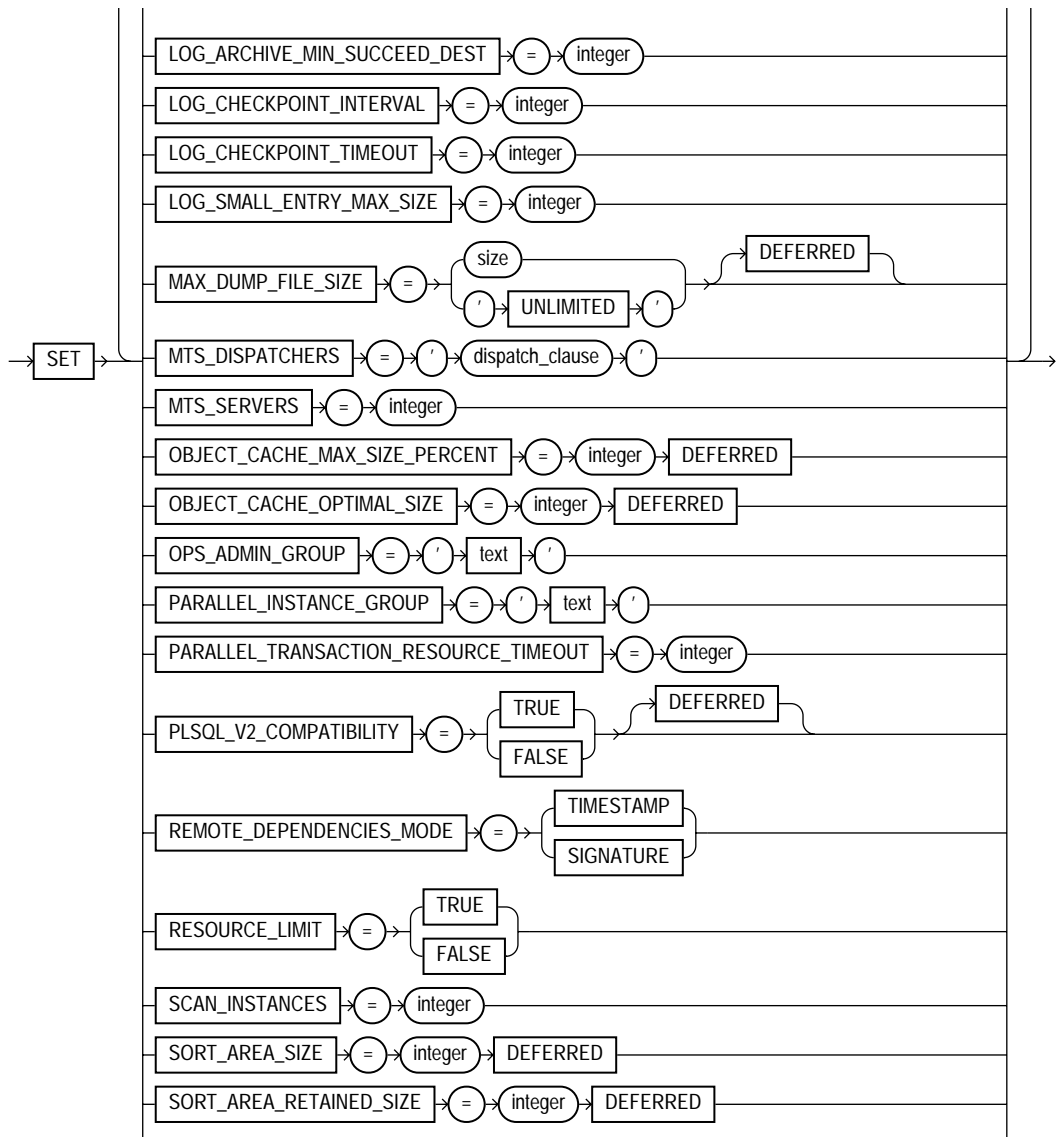
## 構文

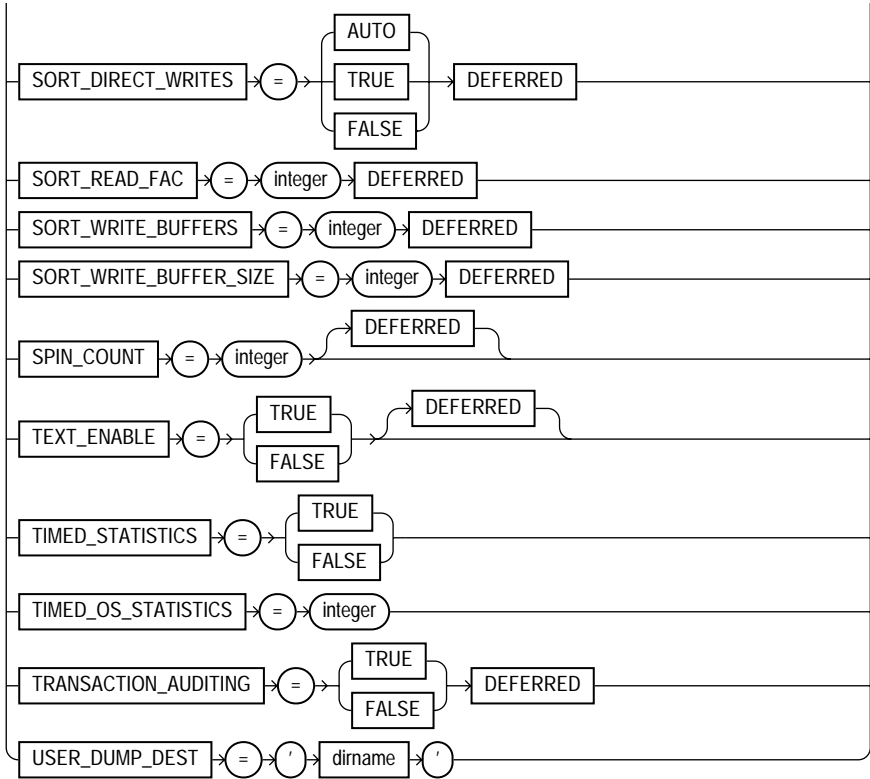


`archive_log_clause`: ARCHIVE LOG 句 (4-164 ページ) を参照。

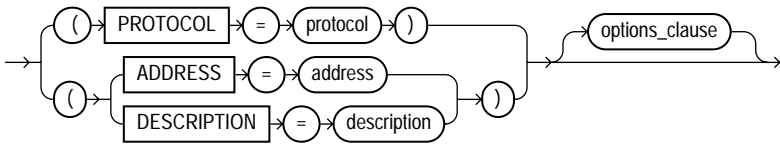




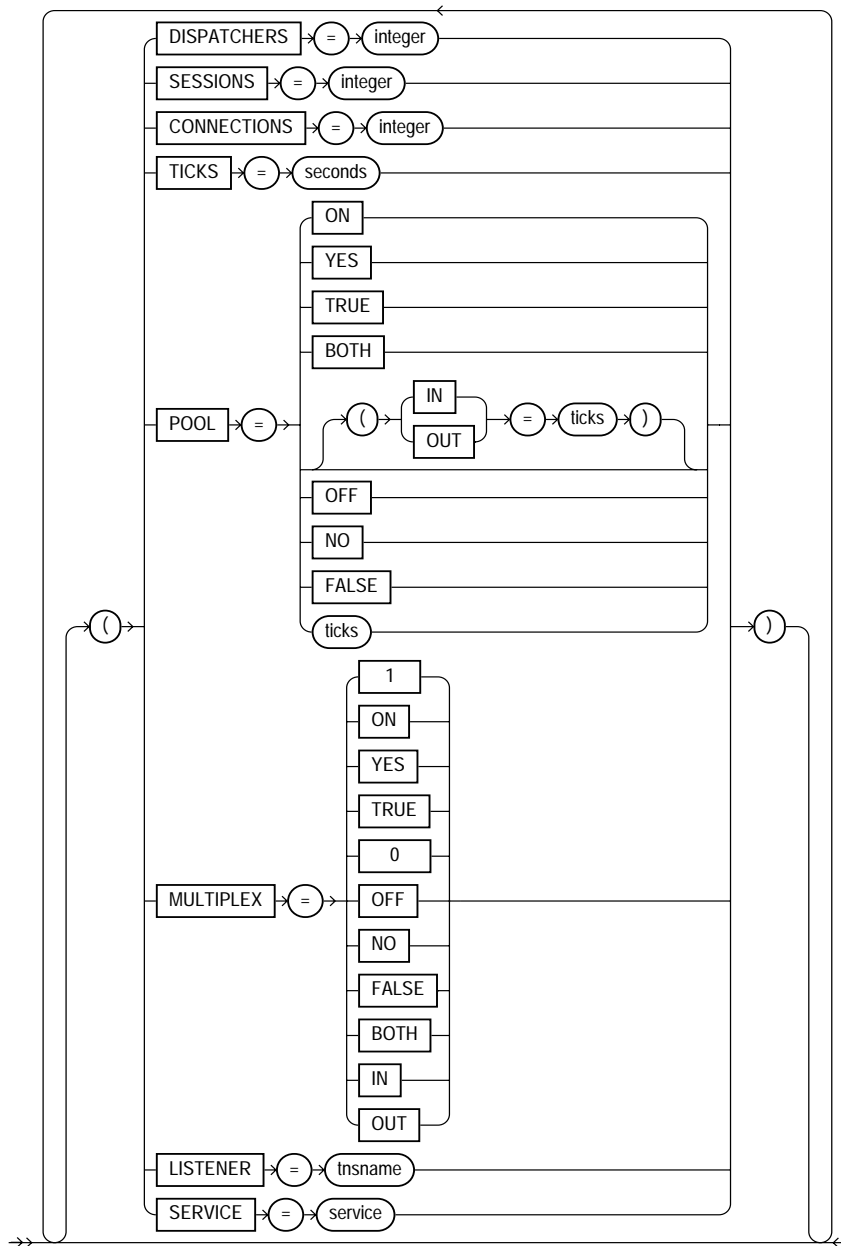




dispatch\_clause::=



options\_clause::=



キーワードとパラメータ

データベースがインスタンスによりマウントされていてもディスマウントされていても、あるいはオープン状態でもクローズ状態でも、次のオプションを使用できます。

RESTRICTED SESSION	Oracle へのログインを制限するかどうかを指定します。
ENABLE	RESTRICTED SESSION システム権限が付与されているユーザーだけが Oracle にログインできます。
DISABLE	ENABLE RESTRICTED SESSION オプションとは逆に機能します。つまり、CREATE SESSION システム権限が付与されている全ユーザーが Oracle にログインできます。
	詳細は、「ログインの制限」(4-97 ページ) を参照してください。
FLUSH SHARED_POOL	システム・グローバル領域 (SGA) の共有ブール上のデータをすべて消去します。詳細は、「共有ブールの消去」(4-97 ページ) を参照してください。
	インスタンスによってデータベースがマウントされていれば、オープンしていてもクローズしていても、次のオプションを使用できます。
CHECKPOINT	チェックポイントを実行します。
GLOBAL	データベースをオープンしているすべてのインスタンスに対してチェックポイントを実行します。
LOCAL	自インスタンスの REDO ログ・ファイル・グループのスレッドに対してだけチェックポイントを実行します。自インスタンスでデータベースをオープンしている場合にだけ、このオプションを使用できます。
	GLOBAL オプションと LOCAL オプションのいずれも指定しないと、グローバル・チェックポイントが自動的に実行されます。詳細は、「チェックポイントの実行」(4-98 ページ) を参照してください。
CHECK DATAFILES	GLOBAL データベースをオープンしているすべてのインスタンスが、すべてのオンラインのデータ・ファイルにアクセスできることを検査します。
	LOCAL 自インスタンスがすべてのオンライン・データ・ファイルにアクセスできることを検査します。
	GLOBAL オプションと LOCAL オプションのいずれも指定しないと、デフォルトでは GLOBAL が採用されます。詳細は、「データ・ファイルのチェック」(4-98 ページ) を参照してください。
	自インスタンスでデータベースがオープンしている場合にだけ、次のパラメータおよびオプションを使用できます。
RESOURCE_LIMIT	リソース制限を制御します。TRUE はリソース制限を使用可能にし、FALSE はリソース制限を使用禁止にします。「リソース制限の使用」(4-98 ページ) を参照してください。

GLOBAL_NAMES	グローバル・ネーム変換の適用を制御します。TRUE に設定すると、グローバル名の適用が使用可能になります。FALSE に設定すると、グローバル名の適用が使用禁止になります。詳細は、「グローバル・ネーム変換」(4-99 ページ)を参照してください。
SCAN_INSTANCES	パラレル・サーバーで、パラレル化操作を行うインスタンスの数を指定する。この構文は、次の主要リリースでは使用されなくなる予定です。
CACHE_INSTANCES	パラレル・サーバーで、表をキャッシュに入れるインスタンスの数を指定します。この構文は、次の主要リリースでは使用されなくなる予定です。

パラレル操作の詳細は、『Oracle8 Server チューニング』を参照してください。

次のマルチスレッド・サーバー・パラメータの詳細は、「マルチスレッド・サーバーのプロセスの管理」(4-99 ページ)を参照してください。

MTS_SERVERS	共有サーバー・プロセスの新たな最小数を指定します。
MTS_DISPATCHERS	ディスパッチャ・プロセスの新たな数値を指定します。
	<i>protocol</i> ディスパッチャ・プロセスのネットワーク・プロトコルです。
	<i>integer</i> 指定したプロトコルのディスパッチャ・プロセスの新しい数値です。
	なお、複数のネットワーク・プロトコルに対して、単一のコマンドで複数の MTS_DISPATCHERS パラメータを指定できます。

下記のライセンス関係パラメータの詳細は、「ライセンス制限の使用」(4-100 ページ)を参照してください。

JOB_QUEUE_PROCESSES	インスタンスごとのジョブ・キュー・プロセスの数を指定します (SNPn、n は 0～9 の後に A～Z を付けた値です)。スナップショットを自動的に更新する場合は、このパラメータを 1 以上に設定してください。同時に多数のスナップショットをリフレッシュする場合を除いて、通常はジョブ・キューは 1 つで十分です。
	Oracle では、DBMS_JOB パッケージによって作成された要求の処理にもジョブ・キュー・プロセスを使用します。表スナップショットの管理の詳細は、『Oracle8 Server レプリケーション』を参照してください。
LICENSE_MAX_SESSIONS	インスタンスの OS セッション数を制限します。値 0 は、この制限を使用禁止にします。
LICENSE_SESSIONS_WARNING	後続セッションのために ALERT ファイルに警告メッセージが書き込まれるセッションのしきい値を指定します。値 0 は、この警告しきい値を使用禁止にします。
LICENSE_MAX_USERS	データベースの同時ユーザー数を制限します。値 0 は、この制限を使用禁止にします。
REMOTE_DEPENDENCIES_MODE	リモートのストアド・プロシージャの依存性がサーバーによってどのように扱われるかを指定します。詳細は、『Oracle8 Server アプリケーション開発者ガイド』を参照してください。

SWITCH LOGFILE	REDO ログ・ファイル・グループを切り替えます。詳細は、「REDO ログ・ファイル・グループの切替え」(4-102 ページ)を参照してください。
DISTRIBUTED RECOVERY	分散回復を使用可能にするかどうかを指定します。  ENABLE                    分散回復を使用可能にします。シングル・プロセス環境では、分散回復を開始するには、このオプションを指定する必要があります。  DISABLE                   分散回復を使用禁止にします。  詳細は、「分散回復の使用可能と使用禁止の切替え」(4-102 ページ)を参照してください。
ARCHIVE LOG	REDO ログ・ファイルを手動でアーカイブ、または自動アーカイブを使用可能または使用禁止にします。ARCHIVE LOG 句 (4-164 ページ)を参照してください。
KILL SESSION	セッションおよび実行中のすべてのトランザクションを終了します。V\$SESSION ビューで次の値を両方とも指定してこのセッションを識別する必要があります。  integer1                   SID 列の値です。  integer2                   SERIAL# 列の値です。  詳細は、「セッションの終了」(4-103 ページ)を参照してください。
DISCONNECT SESSION	専用サーバー・プロセス(接続が MTS 経由の場合は、仮想回路)を破棄することによって、現行のセッションを切断します。アプリケーション・フェイルオーバーが設定されている場合は、それが有効になります。アプリケーション・フェイルオーバーの詳細は、『Oracle8 Server チューニング』および『Oracle8 Parallel Server 概要および管理』を参照してください。V\$SESSION ビューで次の値を両方とも指定してこのセッションを特定する必要があります。  integer1                   SID 列の値です。  integer2                   SERIAL# 列の値です。  POST_TRANSACTION        セッションが切断される前に、実行中のトランザクションを完了できるようにします。このキーワードは、DISCONNECT SESSION を指定する場合に必要です。詳細は、「セッションの切断」(4-104 ページ)を参照してください。
PLSQL_V2_COMPATIBILITY	コンパイル時の PL/SQL プログラムの動作を変更して、Oracle7(PL/SQL V2)の言語要素では有効で、Oracle8(PL/SQL V3)では無効な言語要素を、使用できるようにします。このシステム・パラメータの詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』および『Oracle8 Server リファレンス・マニュアル』を参照してください。  TRUE                      Oracle8 PL/SQL V3 のプログラムが Oracle7 PL/SQL V2 の要素を実行できるようにします。  FALSE                      無効な Oracle7 PL/SQL V2 の要素を使用禁止にします。これはデフォルト値です。
MAX_DUMP_FILE_SIZE	すべてのユーザー・セッションについて、トレース・ダンプ・ファイルのサイズの上限を指定します。最大サイズには、ブロック数を表す負以外の整数、または 'UNLIMITED' を指定します。'UNLIMITED' を指定すると、上限は設定されません。

---

DEFERRED	後続のユーザー・セッションについてだけ、トレース・ダンプ・ファイルのサイズの上限を変更します。
----------	---

---

## ログインの制限

デフォルトでは、CREATE SESSION システム権限を付与されたユーザーは、Oracle にログインできます。ALTER SYSTEM コマンドの ENABLE RESTRICTED SESSION オプションは、RESTRICTED SESSION システム権限を付与されていないユーザーのログインを禁止します。既存のセッションは終了しません。

たとえば、アプリケーションのメンテナンスを行っている最中には、RESTRICTED SESSION システム権限が付与されているアプリケーション開発者だけがログインできるように制限できます。このためには次の文を発行します。

```
ALTER SYSTEM
ENABLE RESTRICTED SESSION;
```

次に ALTER SYSTEM コマンドの KILL SESSION 句を使用して、既存のセッションをどれでも終了できます。

アプリケーションのメンテナンスが終了後に、次の文を発行することによって、CREATE SESSION システム権限が付与されているユーザーもログインできるようになります。

```
ALTER SYSTEM
DISABLE RESTRICTED SESSION;
```

## 共有プールの消去

ALTER SYSTEM コマンドの FLUSH SHARED\_POOL オプションは、システム・グローバル領域 (SGA) 内の共有プール上のすべての情報を消去します。共有プールには次の情報が格納されています。

- キャッシュに入れられたデータ・ディクショナリ情報
- SQL 文の共有 SQL 領域と共有 PL/SQL 領域およびストアド・プロシージャ、関数、パッケージ、トリガー

たとえば、共有プールを消去してから、パフォーマンス分析を開始します。共有プールを消去するには次の文を発行します。

```
ALTER SYSTEM
FLUSH SHARED_POOL;
```

なお、この文は現在実行中の SQL 文またはストアド・プロシージャ、関数、パッケージ、トリガー、あるいは行が完全にフェッチされていない SQL SELECT 文に対する共有 SQL と PL/SQL の各領域は消去しません。

## チェックポイントの実行

ALTER SYSTEM コマンドの CHECKPOINT 句は、チェックポイントの実行を明示的に強制します。チェックポイントを強制的に発生させることによって、コミット済みのトランザクションによる変更がすべてディスク上のデータ・ファイルに書き込まれます。チェックポイントの詳細は、『Oracle8 Server 概要』の「回復構造」を参照してください。

Parallel Server オプション付きの Oracle をパラレル・モードで使用している場合には、データベースをオープンしているすべてのインスタンスのチェックポイントを実行する GLOBAL オプション、または自インスタンスだけのチェックポイントを実行する LOCAL オプションのどちらも指定できます。

次の文は、チェックポイントを強制的に発生させます。

```
ALTER SYSTEM  
CHECKPOINT;
```

チェックポイントが完了するまで、ユーザーに制御は戻りません。

## データ・ファイルのチェック

ALTER SYSTEM コマンドの CHECK DATAFILES 句は、あらゆるオンライン・データ・ファイルへのアクセスを検査します。アクセスできるデータ・ファイルがないと、ALERT ファイルにメッセージが書き込まれます。この処理は、インスタンスによるデータ・ファイルへのアクセスを妨げていたハードウェア障害を解決した後などに実行します。この句の使用方法的詳細は、『Oracle8 Parallel Server 概要および管理』を参照してください。

次の文は、データベースをオープンしている全インスタンスが、すべてのオンライン・データ・ファイルにアクセスできることを検査します。

```
ALTER SYSTEM  
CHECK DATAFILES GLOBAL;
```

## リソース制限の使用

インスタンスを起動すると、初期化パラメータ RESOURCE\_LIMIT の値に基づいてリソース制限が使用可能または使用禁止になります。RESOURCE\_LIMIT オプションを指定した ALTER SYSTEM 文を発行することによって、後続するセッションのリソース制限を使用可能または使用禁止にできます。

リソース制限を使用可能にすると、ユーザーにこのリソース制限が強制的に割り当てられます。リソース制限の値を選択するには、プロファイルつまり制限のセットを作成し、ユーザーにこのプロファイルを割り当てます。このプロセスの詳細は、CREATE PROFILE（4-261 ページ）および CREATE USER（4-353 ページ）を参照してください。

次の ALTER SYSTEM 文は、リソース制限を動的に使用可能にします。

```
ALTER SYSTEM  
SET RESOURCE_LIMIT = TRUE;
```



## グローバル・ネーム変換

インスタンスを起動すると、初期化パラメータ `GLOBAL_NAMES` の値に基づいて、SQL 文でアクセスされるリモート・オブジェクトに対する、グローバル・ネーム変換の適用の可否が Oracle によって決定されます。その後、`ALTER SYSTEM` コマンドの `GLOBAL_NAMES` パラメータを使用したインスタンスの実行中に、グローバル・ネーム変換を使用可能または使用禁止にできます。また、「`ALTER SESSION` コマンド」で説明した `GLOBAL_NAMES` パラメータによって、自インスタンスに対してグローバル・ネーム変換を使用可能または使用禁止にできます。

分散処理を使用または計画している場合は、グローバル・ネーム変換を使用可能にしてください。グローバル・ネーム変換やその適用方法の詳細は、「リモート・データベース内のオブジェクトを参照」(2-50 ページ) および『Oracle8 Server 分散システム』を参照してください。

## マルチスレッド・サーバーのプロセスの管理

インスタンスを起動すると、次の初期化パラメータの値に基づいて、マルチスレッド・サーバー・アーキテクチャ用の共有サーバー・プロセスとディスパッチャ・プロセスが自動的に作成されます。

<code>MTS_SERVERS</code>	共有サーバー・プロセスの最小数の初期値を指定します。既存のプロセスの負荷が変わると、共有サーバー・プロセスの数が自動的に変更されます。インスタンス実行中の共有サーバー・プロセスの数は、初期化パラメータ <code>MTS_SERVERS</code> の値と <code>MTS_MAX_SERVERS</code> の値の範囲で変動します。
<code>MTS_DISPATCHERS</code>	ネットワーク・プロトコル（複数指定可）と、各プロトコルのディスパッチャ・プロセスの数を指定します。

マルチスレッド・サーバー・アーキテクチャの詳細は、『Oracle8 Server 概要』を参照してください。

`ALTER SYSTEM` コマンドの `MTS_SERVERS` パラメータおよび `MTS_DISPATCHERS` パラメータを指定すると、インスタンスの実行中に次の処理を実行できます。

- **共有サーバー・プロセスの追加作成：**共有サーバー・プロセスの最小値を大きくすることにより、共有サーバー・プロセスを追加作成できます。
- **既存の共有サーバー・プロセスの終了：**サーバー・プロセスの負荷が高すぎて残りのプロセスで管理できない場合を除き、共有サーバー・プロセスは、現行コールの処理がすべて終わってから終了します。
- **特定のプロトコルのためのディスパッチャ・プロセスの追加作成：**ディスパッチャ・プロセスは、初期化パラメータ `MTS_MAX_DISPATCHERS` に指定されている全プロトコルの最大数まで追加できます。

このコマンドでは、初期化パラメータ `MTS_DISPATCHERS` に指定されていないネットワーク・プロトコルのディスパッチャ・プロセスは作成できません。新しいプロトコル

のディスパッチャ・プロセスを作成するには、この初期化パラメータの値を変更する必要があります。

- **特定のプロトコルのための既存のディスパッチャ・プロセスの終了**：Oracle は、現在のユーザー・プロセスをそのインスタンスから切断した後に、これらのディスパッチャ・プロセスを終了します。

**例 1.** 次の文は、共有サーバー・プロセスの最小数を 25 に変更します。

```
ALTER SYSTEM
SET MTS_SERVERS = 25;
```

共有サーバー・プロセスの数が 25 より少ない場合には追加作成されます。共有サーバー・プロセスが 25 よりも多く、25 あれば負荷を管理できる場合は、現行のコールによる処理が終了した時点で、25 を超える分の共有サーバー・プロセスは終了します。

**例 2.** 次の文は、動的に、TCP/IP プロトコルのディスパッチャ・プロセス数を 5 に変更し、DECNET プロトコルのディスパッチャ・プロセス数を 10 に変更します。

```
ALTER SYSTEM
SET MTS_DISPATCHERS = 'TCP, 5'
MTS_DISPATCHERS = 'DECnet, 10';
```

TCP のディスパッチャ・プロセスの数が 5 より少ない場合、ディスパッチャ・プロセスが新たに作成されます。5 より多い場合は、接続されているユーザーが接続を切り離れた後に、5 を超える分のディスパッチャ・プロセスは終了します。

DECnet のディスパッチャ・プロセスの数が 10 より少ない場合にはディスパッチャ・プロセスが新たに作成されます。10 より多い場合は、接続されているユーザーが接続を切り離れた後に、10 を超える分のディスパッチャ・プロセスは終了します。

これ以外のプロトコル用として既存ディスパッチャがある場合に、この文はそのディスパッチャの数に影響を及ぼしません。

## ライセンス制限の使用

Oracle は、同時使用ライセンス制限と名前付きユーザー・ライセンス制限をユーザーの Oracle ライセンス契約に従って適用します。インスタンスを開始すると、Oracle は次の初期化パラメータの値に基づいてこれらの制限を設定します。

<b>LICENSE_MAX_SESSIONS</b>	同時使用ライセンス制限つまり同時セッションの最大数を設定します。この最大数に達すると、 <b>RESTRICTED SESSION</b> システム権限を持っているユーザーだけしか接続できません。
-----------------------------	--

<b>LICENSE_SESSIONS_WARNING</b>	同時使用に関する警告しきい値を設定します。このしきい値に達すると、後続の各セッションの ALERT データベース・ファイルに警告メッセージが書き込まれます。RESTRICTED SESSION システム権限を持っているユーザーも、後続セッションを開始するときに、警告メッセージを受け取ります。
<b>LICENSE_MAX_USERS</b>	データベースに接続できるユーザーの最大数を設定します。この最大数に達すると、これを超える数の接続はできません。

ALTER SYSTEM コマンドの LICENSE\_MAX\_SESSIONS、LICENSE\_SESSIONS\_WARNING、LICENSE\_MAX\_USERS の各パラメータを指定することで、インスタンスの実行中に制限やしきい値を動的に変更したり、使用禁止にしたりできます。Oracle ライセンスを適切にアップグレードしていない場合には、セッションやユーザーの最大数を変更も使用禁止にもしないでください。ライセンスのアップグレードが必要になる場合には、オラクル社技術サポートに連絡してください。

次の場合には、新たな制限は後続のセッションやユーザーだけに適用されます。

- 現行のセッション数を下回る数に最大数を減らしても、Oracle は新しい最大数を適用するために既存のセッションを終了しない。RESTRICTED SESSION システム権限のないユーザーは、セッション数が新しい最大数を超えない場合にだけ、新しいセッションを開始できます。
- セッションの警告しきい値を現行のセッション数より小さくすると、以降のすべてのセッションのたびにメッセージが ALERT ファイルに書き込まれる。
- ユーザー数の最大値はデータベースに登録されたユーザー数より小さくできない。

**例 1.** 次の文は、自インスタンスにおけるセッションの最大数を 64 に、セッションの警告しきい値を 54 に動的に変更します。

```
ALTER SYSTEM
SET LICENSE_MAX_SESSIONS = 64
LICENSE_SESSIONS_WARNING = 54;
```

セッション数が 54 に達すると、後続の各セッションの ALERT ファイルに警告メッセージが書き込まれます。RESTRICTED SESSION システム権限を持っているユーザーも、後続セッションを開始するときに、警告メッセージを受け取ります。

セッション数が 64 に達すると、セッション数が再び 64 を下回るまでは、RESTRICTED SESSION システム権限を持つユーザーしか新たなセッションを開始できません。

**例 2.** 次の文は、自インスタンスのセッションの最大数を動的に使用禁止にします。

```
ALTER SYSTEM
SET LICENSE_MAX_SESSIONS = 0;
```

この文の実行後は、インスタンスのセッション数に制限がなくなります。

例 3. 次の文は、データベースのユーザーの最大数を 200 に動的に変更します。

```
ALTER SYSTEM
SET LICENSE_MAX_USERS = 200;
```

この文の実行後は、データベース・ユーザー数が 200 を超えることはありません。

## REDO ログ・ファイル・グループの切替え

ALTER SYSTEM コマンドの SWITCH LOGFILE オプションは、現行の REDO ログ・ファイル・グループのファイルが満杯であるかどうかにかかわらず、新しい REDO ログ・ファイル・グループへの書き込みを強制的に開始します。書き込み中のファイルの削除および改名はできません。ただし、ログ・スイッチを強制的に発生させることによって、現行の REDO ログ・ファイル・グループまたはそのメンバーの 1 つは削除または改名できます。強制的に発生したログ・スイッチは自インスタンスの REDO ログ・スレッドだけに影響します。なお、ログ・スイッチを強制的に発生させると、Oracle によってチェックポイントが実行されます。チェックポイントの完了時ではなく、実行直後にユーザーに制御が戻ります。

次の文は、ログ・スイッチを強制的に発生させます。

```
ALTER SYSTEM
SWITCH LOGFILE;
```

## 分散回復の使用可能と使用禁止の切替え

Oracle では分散トランザクション（すなわち複数のデータベース上のデータを変更するトランザクション）を実行できます。分散トランザクションのコミット・プロセス中にネットワーク障害またはマシン障害が発生すると、トランザクションの状態が不明、つまりインダウトになる場合があります。これらの障害が解決され、ネットワークとそのノードが再びオンラインに戻されると、Oracle はトランザクションを回復します。

複数プロセス・モードで Oracle を使用している場合、分散回復は自動的に実行されます。MS-DOS のように、シングル・プロセス（シングル・ユーザー）・モードで Oracle を使用している場合は、次の文を発行して分散回復を明示的に開始する必要があります。

```
ALTER SYSTEM ENABLE DISTRIBUTED RECOVERY;
```

インダウト・トランザクションを回復する場合、特にそのトランザクションに関係があるリモート・ノードにアクセスできない場合は、この文を複数回発行しなければならないこともあります。インダウト・トランザクションはデータ・ディクショナリ・ビュー DBA\_2PC\_PENDING に表示されます。逆に、DBA\_2PC\_PENDING に表示されなくなれば、インダウト・トランザクションが回復されたことになります。分散トランザクションおよび分散回復の詳細は、『Oracle8 Server 分散システム』を参照してください。

次の文を使用して、シングル・プロセス・モードまたはマルチ・プロセス・モードのどちらの分散回復も使用禁止にできます。

```
ALTER SYSTEM DISABLE DISTRIBUTED RECOVERY;
```

デモンストレーションなどの目的で、分散回復を使用禁止にできます。ALTER SYSTEM 文に ENABLE DISTRIBUTED RECOVERY 句を指定して実行すれば、分散回復を再び使用可能にできます。

## セッションの終了

ALTER SYSTEM コマンドの KILL SESSION 句は、次のタスクをただちに実行してセッションを終了します。

- 現行トランザクションのロールバック
- すべてのロックの解除
- すべてのリソースの解放

あるユーザーのセッションで、他のユーザーが必要とするリソースを使用している場合、そのユーザーのセッションを中断させる必要があります。このユーザーは、セッションが中断されたことを示すエラー・メッセージを受け取り、新しいセッションを開始しない限りこのデータベースをコールできません。中断できるセッションは、自インスタンスの現行セッションと同じインスタンス上のセッションだけです。

リモート・データベースからの応答を待っていたりトランザクションのロールバック中などのような、最後まで完了する必要があるアクティビティを実行しているセッションを中断（キル）すると、このアクティビティが完了してからセッションが中断され、その後ユーザーに制御が戻ります。待ち時間が 1 分以上続く場合は、中断されるセッションがマーク付けされ、マーク付けされたセッションが中断されることを示すメッセージとともにユーザーに制御が戻ります。その後、このアクティビティが完了するとセッションが中断されます。

**例** . 次の V\$SESSION 動的性能表のデータについて考えます。

```
SELECT sid, serial, username
FROM v$session
```

SID	SERIAL	USERNAME
1	1	
2	1	
3	1	
4	1	
5	1	
7	1	
8	28	OPS\$BQUIGLEY
10	211	OPS\$SWIFT
11	39	OPS\$OBRIEN
12	13	SYSTEM
13	8	SCOTT

次の例では、V\$SESSION 上の SID 値と SERIAL# 値を使用してユーザー SCOTT のセッションを中断します。

```
ALTER SYSTEM  
KILL SESSION '13, 8';
```

### セッションの切断

DISCONNECT SESSION 句は、KILL SESSION 句に似ていますが、異なる点が2つあります。

1 つは、ALTER SYSTEM DISCONNECT SESSION 'X, Y' POST\_TRANSACTION コマンドは、セッションで処理中のすべての現行トランザクションが終了するまで、適用されないことです。

もう1点は、セッションが中断されるのではなく切断されることです。つまり、専用サーバー・プロセス（接続が MTS 経由の場合は、仮想回路）は、このコマンドによって破棄されるということです。セッションの接続が終了することによって、適切なシステム・パラメータが設定されている場合はアプリケーション・フェイルオーバーが適用されます。

セッションの切断によって、基本的にはアプリケーション・フェイルオーバーを手動で実行できるようになります。このコマンドをパラレル・サーバー環境で使用すると、オーバーロード状態のインスタンスのセッションを切断し、それらのセッションを別のインスタンスに移動できます。

POST\_TRANSACTION キーワードは必須です。

例 . 次の文は、V\$SESSION から SID と SERIAL# の値を使って、ユーザー SCOTT のセッションを切断します。

```
ALTER SYSTEM  
DISCONNECT SESSION '13, 8' POST_TRANSACTION;
```

アプリケーション・フェイルオーバーの詳細は、『Oracle8 Parallel Server 概要および管理』および『Oracle8 Server チューニング』を参照してください。

### 関連項目

ALTER SESSION (4-58 ページ)

ALTER SESSION (4-58 ページ)

CREATE USER (4-353 ページ)

ARCHIVE LOG 句 (4-164 ページ)

---

## ALTER TABLE

### 用途

次のいずれかの方法で表の定義を変更します。

- 列を追加する。
- 整合性制約を追加する。
- 列（データ型、サイズ、デフォルト値）を再定義する。
- 記憶特性またはその他のパラメータを変更する。
- 非パーティション表の実記憶属性またはパーティション表のデフォルト属性を変更する。
- 整合性制約またはトリガーを使用可能または使用禁止にする、あるいは削除する。
- エクステントを明示的に割り当てる。
- 表の未使用領域の割当てを明示的に解除する。
- 表への書込みを許可または禁止する。
- 表に対する並列度を変更する。
- 非パーティション表またはパーティション表、表パーティションのロギング属性を変更する。
- **CACHE/NOCACHE** 属性を変更する。
- 表パーティションを追加または変更、分割、移動、削除、切り捨てる。
- 表または表パーティションを改名する。
- 索引構成表の特性を追加または変更する。
- **LOB** 列を追加または変更する。
- **OBJ** オブジェクト型またはネストした表型、**VARRAY** 型の列を表に追加あるいは変更する。
- **OBJ** 整合性制約をオブジェクト型列に追加する。

---

**注意：** コマンドと句の説明で**OBJ**の印が前に付いている箇所は、Oracle Object Option がデータベース・サーバーにインストールされている場合にだけ有効です。

---

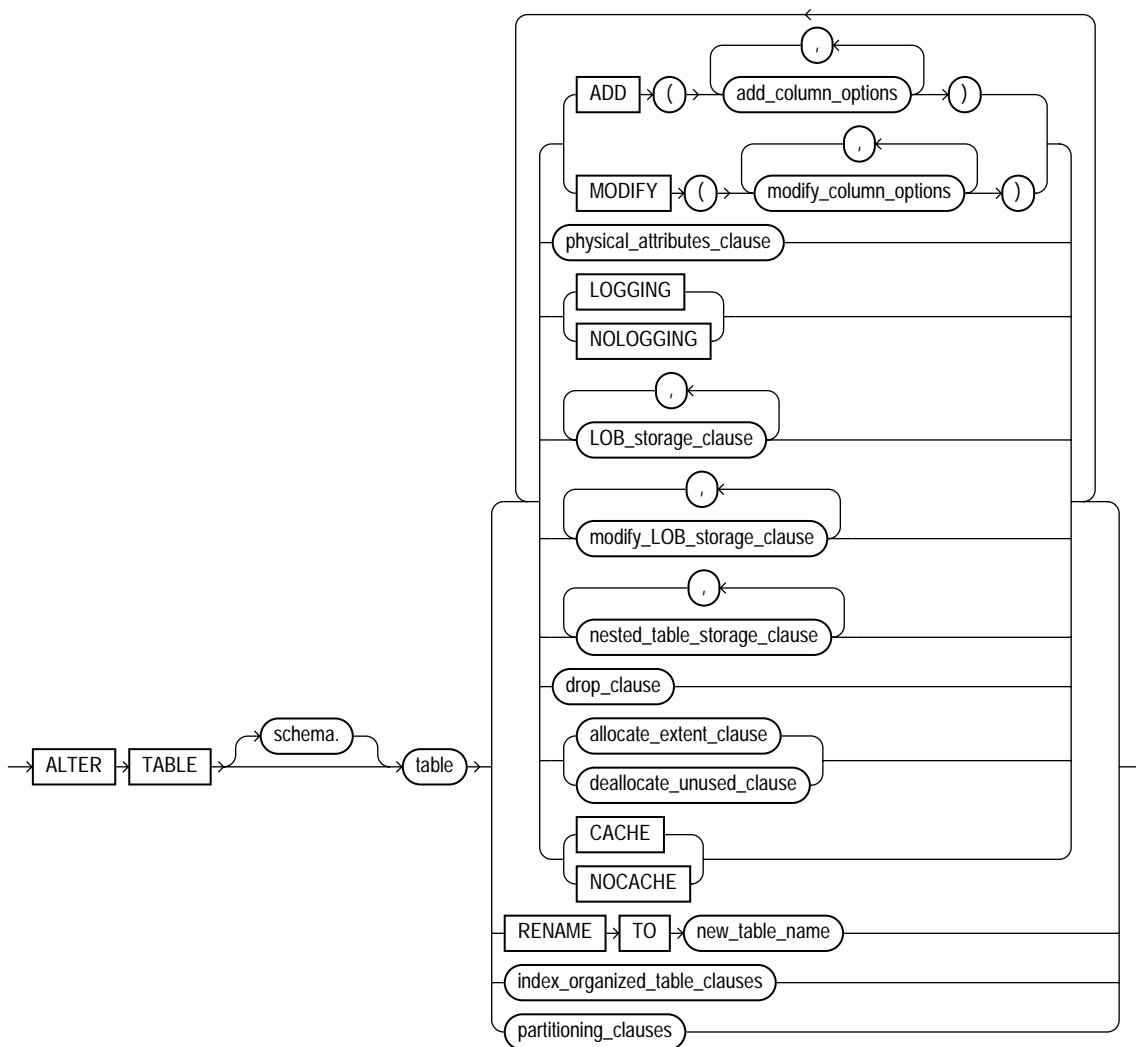
### 前提条件

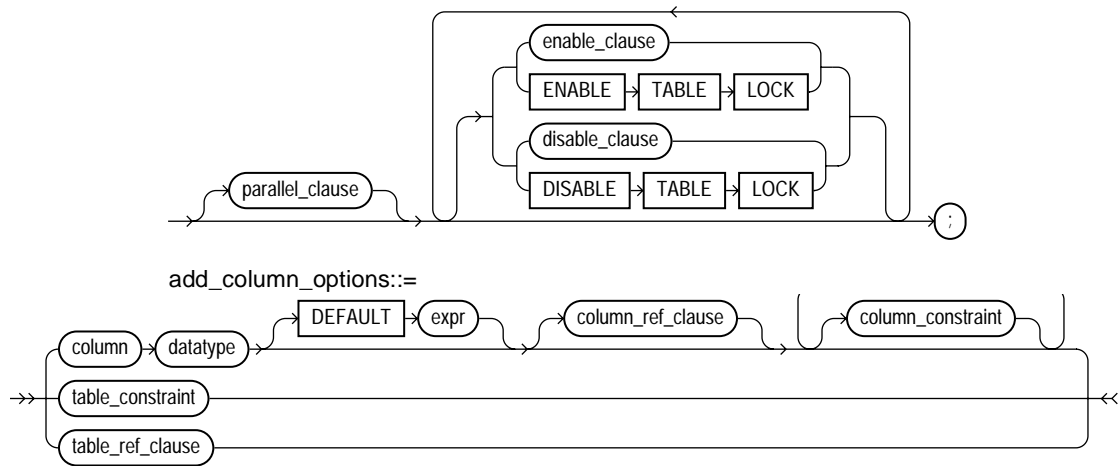
表が自スキーマ内にあることが必要です。自スキーマ内にはない場合は、その表に対する ALTER 権限または ALTER ANY TABLE システム権限が必要です。また、CREATE ANY INDEX 権限が必要な操作もあります。

**OBJ** 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には、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。



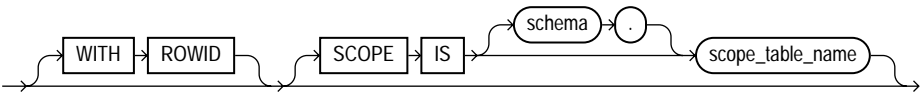
## 構文



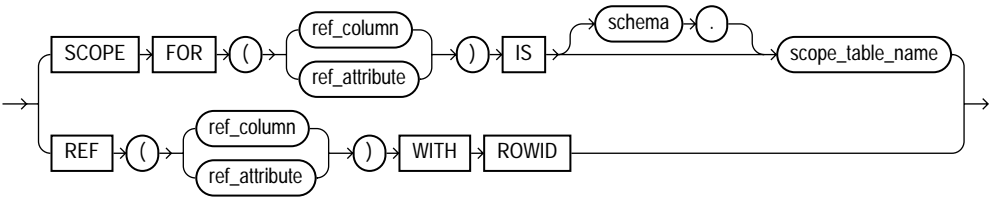


`column_constraint`、`table_constraint`:CONSTRAINT 句（4-185 ページ）を参照。

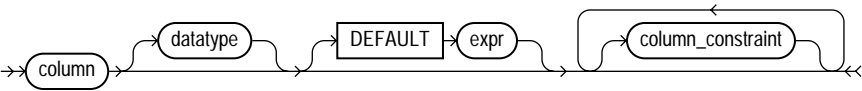
`column_ref_clause::=`



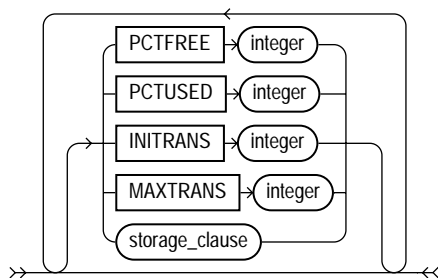
`table_ref_clause::=`



`modify_column_options::=`

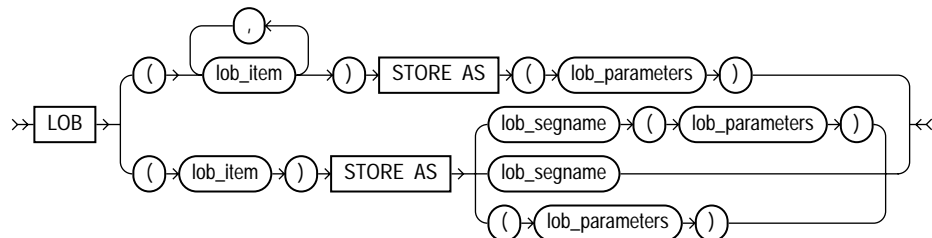


physical\_attributes\_clause::=

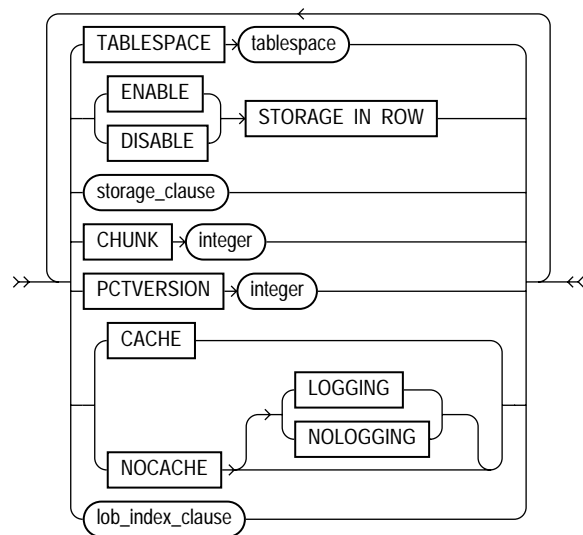


storage\_clause: STORAGE 句 (4-518 ページ) を参照。

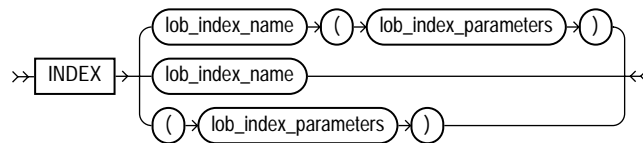
LOB\_storage\_clause::=



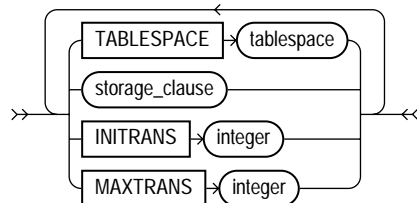
LOB\_parameters::=



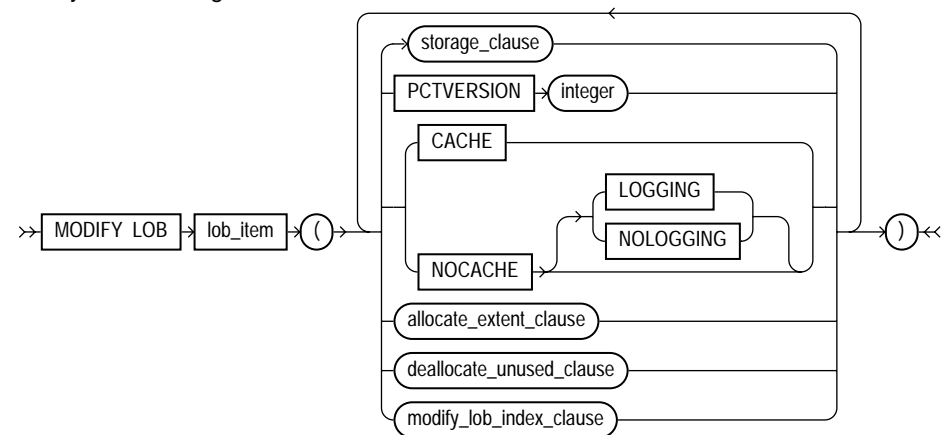
LOB\_index\_clause ::=



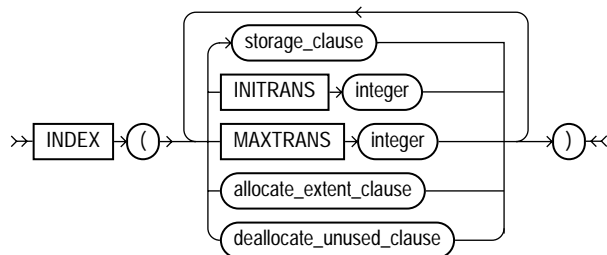
LOB\_index\_parameters::=



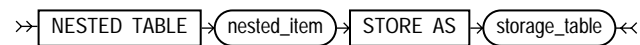
```
modify_LOB_storage_clause::=
```



modify\_LOB\_index\_clause::=

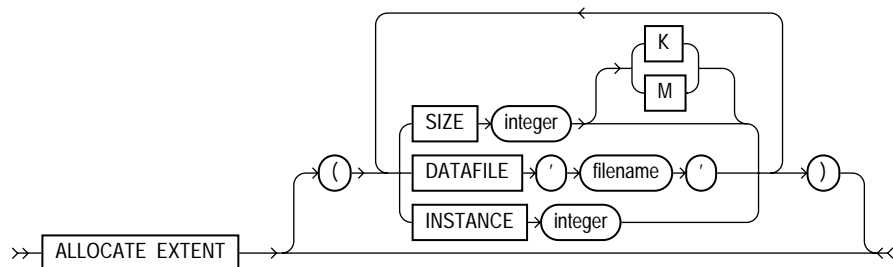


nested\_table\_storage\_clause::=



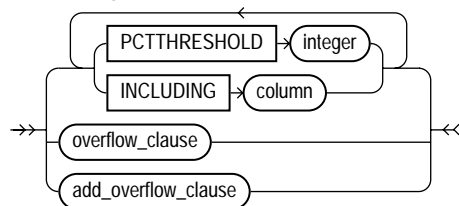
rop\_clause: DROP 句 (4-379 ページ) を参照。

allocate\_extent\_clause::=

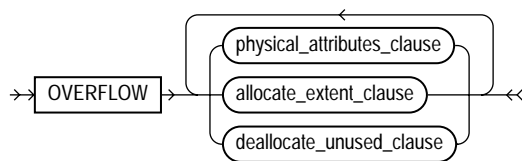


deallocate\_unused\_clause: DEALLOCATE UNUSED 句 (4-368 ページ) を参照。

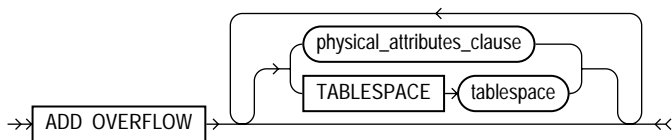
index\_organized\_table\_clauses::=



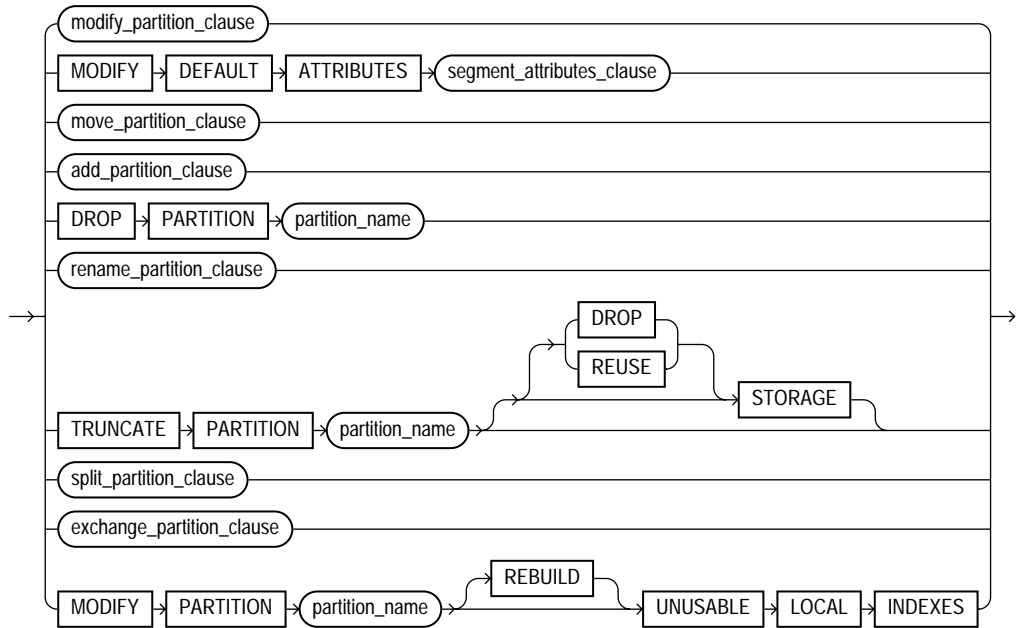
**overflow\_clause ::=**



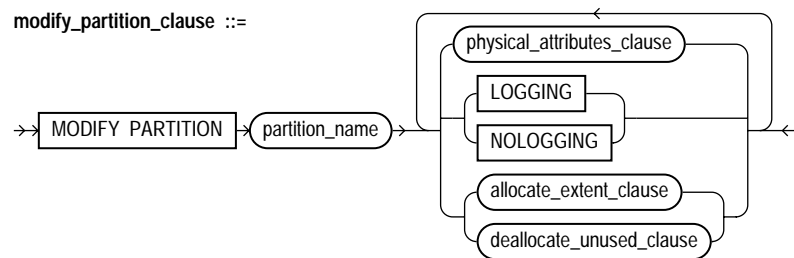
**add\_overflow\_clause ::=**



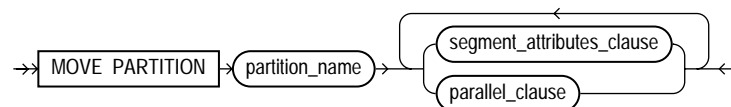
partitioning\_clauses::=



modify\_partition\_clause ::=



move\_partition\_clause ::=

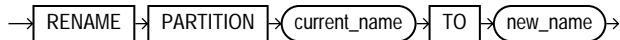


## ALTER TABLE

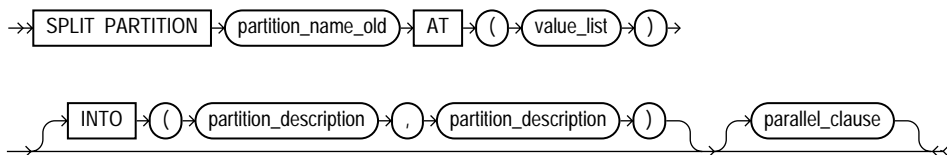
**add\_partition\_clause ::=**



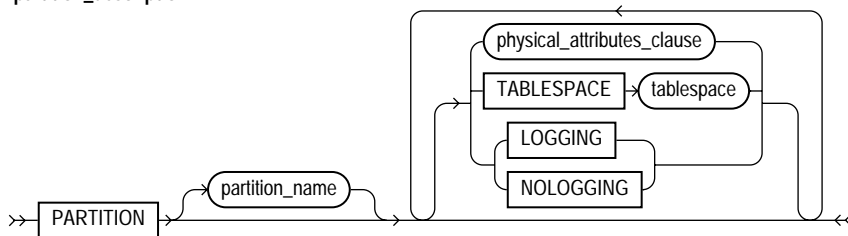
**rename\_partition\_clause ::=**



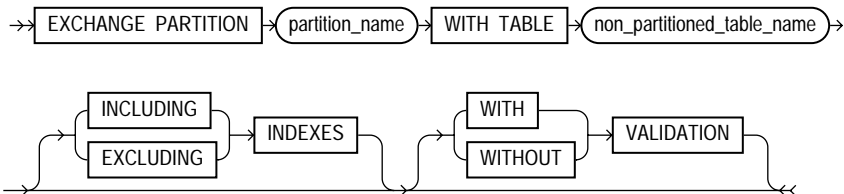
**split\_partition\_clause ::=**



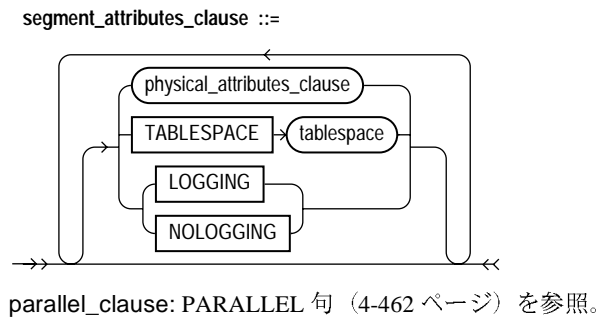
**partition\_description ::=**



**exchange\_partition\_clause ::=**







## キーワードとパラメータ

<i>schema</i>	変更する表が定義されているスキーマを指定します。 <i>schema</i> を指定しないと、この表は自スキーマ内にあるとみなされます。
<i>table</i>	変更する表の名前です。索引構成表の定義を変更できます。
ADD	列または整合性制約を追加します。索引構成表には列を ADD(追加) できません。「列の追加」(4-121 ページ) を参照してください。
MODIFY	既存の列の定義を変更します。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、この部分は変更されません。索引構成表の列定義は、MODIFY(変更) できません。「列定義の変更」(4-121 ページ) を参照してください。
<i>column</i>	追加または変更の対象となる列名です。
<i>datatype</i>	<p>新しい列に対するデータ型、または既存の列に対する新たなデータ型を指定します。</p> <p>参照整合性制約の外部キーの一部として、文で列が指定されている場合に限り、データ型を省略できます。Oracle は、参照整合性制約の参照キーの対応列のデータ型を、その列に自動的に割り当てます。</p>
DEFAULT	<p>新しい列に対するデフォルト値、または既存の列に対する新たなデフォルト値を指定します。後続の INSERT 文で列に値を指定しないと、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すると、その表のすべての行にデフォルトの列値が挿入されます。</p> <p>デフォルト値のデータ型は必ず列に対して指定したデータ型と一致していなければなりません。また、列にはこのデフォルト値を保持できる長さがが必要です。DEFAULT 式には他の列、つまり疑似列 CURRVAL および NEXTVAL、LEVEL、ROWNUM、あるい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。</p>
<i>column_constraint</i>	既存の列への NOT NULL 制約を追加または削除します。CONSTRAINT 句 (4-185 ページ) の <i>column_constraint</i> の構文を参照してください。

	<i>table_constraint</i>	表に対する整合性制約を追加します。CONSTRAINT 句（4-185 ページ）の <i>table_constraint</i> の構文を参照してください。  「REF」（4-125 ページ）を参照してください。
<i>modify_default_attributes_clause</i>		このオプションは、パーティション表だけに有効です。このオプションを使って、パーティション表のデフォルト属性に新しい値を指定できます。
<i>physical_attributes_clause</i>		表またはパーティション、オーバーフロー・データ・セグメントに対する PCTFREE および PCTUSED、INITRANS、MAXTRANS の各パラメータの値、あるいはパーティション表のデフォルト特性を変更します。CREATE TABLE（4-303 ページ）の PCTFREE および PCTUSED、INITRANS、MAXTRANS の各パラメータを参照してください。
	<i>storage_clause</i>	表またはパーティション、オーバーフロー・データ・セグメントの記憶特性、あるいはパーティション表のデフォルト特性を変更します。 STORAGE 句（4-518 ページ）を参照してください。
PCTTHRESHOLD		索引ブロック内で、索引構成表用に確保されている領域の割合（パーセント）を指定します。指定されたしきい値を超える行の部分は、オーバーフロー領域に保存されます。OVERFLOW が指定されていない場合は、THRESHOLD 限界を超える行は受け付けられません。 PCTTHRESHOLD は、0 ～ 50 までの値でなければなりません。
	<i>INCLUDING column_name</i>	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。 <i>column_name</i> に続く列はすべて、オーバーフロー・データ・セグメントに保存されます。 <i>column_name</i> は、最後の主キー列の名前でも主キー以外の列の名前でも構いません。
	OVERFLOW	索引構成表用と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を指定します。この句に指定するパラメータは、オーバーフロー・データ・セグメントだけに適用できます。CREATE TABLE（4-303 ページ）を参照してください。
	ADD OVERFLOW	指定された索引構成表にオーバーフロー・データ・セグメントを追加します。  「索引構成表」（4-123 ページ）も参照してください。
LOB		新しく追加した LOB 列の LOB 記憶特性を指定します。この句では、既存の LOB 列は変更できません。
	<i>lob_item</i>	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ときに、その LOB 列名つまり LOB オブジェクトの属性を指定します。
STORE AS		
	<i>lob_segname</i>	LOB データ・セグメントの名前を指定します。 <i>lob_item</i> が複数指定されている場合は、 <i>lob_segname</i> を使用できません。
	ENABLE STORAGE IN ROW	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 値を行（インライン）に格納することを指定します。これはデフォルトです。

DISABLE STORAGE IN ROW	LOB 値の長さに関係なく、LOB 値を行外に格納することを指定します。
	LOB の格納場所に関係なく、LOB ロケータは、常に行に格納されます。STORAGE IN ROW は、いったん設定すると変更できません。
CHUNK <i>integer</i>	LOB 操作に割り当てるバイト数を指定します。 <i>integer</i> がデータベース・ブロック・サイズの倍数でない場合は、次の倍数に (バイト単位で) 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、 <i>integer</i> 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 (32K) で、これが Oracle で許容されるブロック・サイズの最大値です。  注意: CHUNK の値は、INITIAL および NEXT の両方の値 (デフォルト値または記憶域句で指定した値) 以下でなければなりません。CHUNK が INITIAL または NEXT の値を超えていると、エラーになります。
PCTVERSION <i>integer</i>	LOB の新バージョンを作成するために使用する記憶領域の、LOB 記憶領域全体に対する割合の最大値です。デフォルト値は 10 です。すなわち LOB データの旧バージョンは LOB 記憶領域全体が 10% 使用されるまでは上書きされません。
INDEX <i>lob_index_name</i>	LOB 索引セグメントの名前です。 <i>lob_item</i> が複数指定されている場合は、 <i>lob_index_name</i> を使用できません。
MODIFY LOB ( <i>lob_item</i> )	LOB 属性 <i>lob_item</i> つまり LOB オブジェクトの属性の物理属性を変更します。各 MODIFY LOB 句に指定する LOB 列は 1 つだけです。  「LOB 列」(4-124 ページ) を参照してください。
◎ <b>NESTED TABLE</b> <i>nested_item</i> STORE AS <i>storage_table</i>	すべての <i>nested_item</i> 値を持つ行が入っている格納表の名前として <i>storage_table</i> を指定します。この句は、ネストした表型の表の列つまり列属性を変更するときに指定する必要があります。  <i>nested_item</i> には、ネストした表型の列つまり列修飾属性の名前を指定します。  <i>storage_table</i> は、記憶表の名前です。記憶表は、親表と同じスキーマおよび表領域内で変更されます。  「ネストした表の列」(4-124 ページ) を参照してください。
<i>drop_clause</i>	整合性制約を削除します。DROP 句 (4-379 ページ) を参照してください。
ALLOCATE EXTENT	表またはパーティション、オーバーフロー・データ・セグメント、LOB データ・セグメント、LOB 索引の新しいエクステントを明示的に割り当てます。
SIZE	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 単位でもサイズを指定できます。このパラメータを指定しないと、サイズは表またはオーバーフロー・データ・セグメント、LOB 索引の STORAGE パラメータに基づいて決定されます。

	<p><b>DATAFILE</b></p> <p>新しいエクステントを割り当てるデータ・ファイルを、表またはオーバーフロー・データ・セグメント、LOB データ表領域、LOB 索引の表領域の中から1つ指定します。このパラメータを指定しないと、データ・ファイルは Oracle によって選択されます。</p>
	<p><b>INSTANCE</b></p> <p>指定され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ると、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 <b>INSTANCE_NUMBER</b> の値で識別されます。このパラメータを指定しないと、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われ、必要に応じて領域が割り当てられます。詳細は、『Oracle8 Server 概要』を参照してください。パラレル・モードの <b>Parallel Server</b> オプションで 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。</p> <p>この句を使用して明示的にエクステントを割り当てると、<b>NEXT</b> と <b>PCTINCREASE</b> の記憶領域パラメータで指定した、次に割り当てられるエクステントのサイズに影響します。</p>
<p><b>DEALLOCATE UNUSED</b></p>	<p>表またはパーティション、オーバーフロー・データ・セグメント、LOB データ・セグメント、LOB 索引の最後における未使用領域の割当てを明示的に解除し、解放された領域を他のセグメントから利用できるようにします。ただし解放できるのは、上限基準点を超える未使用領域だけです。<b>KEEP</b> を指定しないと、未使用領域がすべて解放されます。詳細は、<b>DEALLOCATE UNUSED</b> 句（4-368 ページ）を参照してください。</p>
	<p><b>KEEP</b></p> <p>割当てを解除した後に表またはオーバーフロー・データ・セグメント、LOB データ・セグメント、LOB 索引に残す、上限基準点を超えるバイト数を指定します。残りのエクステント数が <b>MINEXTENTS</b> より少ないときは、<b>MINEXTENTS</b> は現行のエクステント数に設定されます。初期エクステントが <b>INITIAL</b> より小さくなると、<b>INITIAL</b> は初期エクステントの現行の値に設定されます。</p>
<p><i>enable_clause</i></p>	<p>表に対応付けられた単一の整合性制約またはすべてのトリガーを使用可能にします。<b>ENABLE</b> 句（4-414 ページ）を参照してください。</p>
<p><b>CACHE</b></p>	<p>アクセス頻度の高いデータについて、全表走査の実行時に、そ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 <b>LRU</b> リストの最高使用頻度側に入れるように指定します。このオプションは、小規模な参照表で有効です。</p> <p><b>CACHE</b> は、索引構成表に対しては無効です。</p>
<p><b>NOCACHE</b></p>	<p>アクセス頻度の低いデータについて、全表走査の実行時に、そ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 <b>LRU</b> リストの最低使用頻度側に入れるように指定します。</p> <p>LOB の場合、LOB 値は、バッファ・キャッシュに入れられないか、バッファ・キャッシュに入れられて <b>LRU</b> リストの最低使用頻度側に置かれるかのいずれかです。（デフォルトでは後者になります。）</p> <p><b>NOCACHE</b> は、索引構成表に対しては無効です。</p>

LOGGING/ NOLOGGING	<p>LOGGING/NOLOGGING は、非パーティション表、表パーティション、またはパーティション表の全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 操作を、REDO ログ・ファイルにログを記録するか (LOGGING)、しないか (NOLOGGING) を指定します。</p> <p><i>modify_default_attributes_clause</i> とともにこのオプションを使用すると、パーティション表のロギング属性が影響を受けます。</p> <p>LOGGING/NOLOGGING は、ALTER TABLE...MOVE 操作と ALTER TABLE...SPLIT 操作のログを記録するかどうかも指定します。</p> <p>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、レコード・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。メディア回復中に NOLOGGING が適用されると、REDO データはログに記録されないため、エクステント無効化レコードはブロック範囲を論理的に壊れているものとしてマーク設定します。このため、消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とってください。</p> <p>データベースが ARCHIVELOG モードで運用されている場合、LOGGING 操作の前にとったバックアップからのメディア回復によって、表が復元されます。ただし、NOLOGGING 操作の前にとったバックアップからのメディア回復では、表は復元されません。</p> <p>実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。</p> <p>LOGGING オプションおよびパラレル DML の詳細は、『Oracle8 Parallel Server 概要および管理』を参照してください。</p> <p>NOLOGGING は、索引構成表の変更に対しては有効なキーワードではありません。</p>
RENAME TO	<i>table</i> を <i>new_table_name</i> に改名します。
<i>partitioning_clauses</i>	「表パーティションの変更」(4-126 ページ) を参照してください。
MODIFY PARTITION[ 表 パーティション ]	表パーティションの実物理属性 <i>partition_name</i> を変更します。パーティションの新しい物理属性として、ロギング属性、PCTFREE または PCTUSED、INITRANS、MAXTRANS の各パラメータ、記憶領域パラメータを指定できます。
MODIFY PARTITION[ 索引 パーティション ]	索引パーティション <i>partition_name</i> の属性を変更します。次のオプションは、MODIFY PARTITION[ 表パーティション ] オプションの句では指定できません。
UNUSABLE LOCAL INDEXES	<i>partition_name</i> に対応付けられているすべてのローカル索引パーティションに使用不可のマークを付けます。
REBUILD UNUSABLE LOCAL INDEXES	<i>partition_name</i> に対応付けられている使用不可のローカル索引パーティションを再構築します。
RENAME PARTITION	表パーティションの名前を <i>current_name</i> から <i>new_name</i> に変更します。

MOVE PARTITION	表パーティション <i>partition_name</i> を別のセグメントに移動します。パーティション・データを別の表領域に移動したり、断片化を低減するためにデータを再クラスタ化したり、作成時の物理属性を変更したりできます。
ADD PARTITION	新しいパーティション <i>new_partition_name</i> をパーティション表の「一番上」に追加します。パーティションの新しい物理属性として、ロギング属性、PCTFREE または PCTUSED、INITRANS、MAXTRANS の各パラメータ、記憶領域パラメータを指定できます。  VALUES LESS THAN ( <i>value_list</i> )      新しいパーティションの上限を指定します。 <i>value_list</i> は、 <i>column_list</i> に対応するリテラル値を順序通りにカンマで区切ったリストです。 <i>value_list</i> 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しなければなりません。
DROP PARTITION	パーティション表から、パーティション <i>partition_name</i> およびそのパーティション内のデータを削除します。
TRUNCATE PARTITION	表内のパーティション <i>partition_name</i> から行をすべて削除します。  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します。  REUSE STORAGE      削除した行が占有していた領域を、そのまま元のパーティションに割り当てることを指定します。この領域は、その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だけ使用できます。
SPLIT PARTITION	元のパーティション <i>partition_name_old</i> から 2 つの新しいパーティションを作成し、各パーティションのセグメント属性および物理属性、初期エクステントを新たに指定します。 <i>partition_name_old</i> に対応付けられていたセグメントは破棄されます。  AT ( <i>value_list</i> ) <i>split_partition_1</i> の上限（境界は含まない）を指定します。 <i>value_list</i> の値は、 <i>partition_name_old</i> の分割前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      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。  <i>partition_description, partition_description</i> 分割の結果生成された 2 つのパーティションの適切な名前と物理属性を指定します。
EXCHANGE PARTITION	パーティション <i>partition_name</i> を非パーティション表に変換し、非パーティション表をパーティション表のパーティションに変換します。この変換は各データ（および索引）セグメントを交換することによって行います。デフォルトの動作は、EXCLUDING INDEXES WITH VALIDATION です。  WITH TABLE <i>table</i> 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に対応する標準索引と交換することを指定します。

	EXCLUDING INDEXES	パーティション表に対応するすべてのローカル索引パーティションと、交換された表のすべての標準索引に使用不可のマークを付けることを指定します。
	WITH VALIDATION	WITH VALIDATION を指定すると、交換後、表内の行の照合が行われ、一致しない行があるとエラーが戻ります。
	WITHOUT VALIDATION	WITHOUT VALIDATION を指定すると、交換後、表内の行の照合は行われません。
<i>parallel_clause</i>		表の並列度を指定します。PARALLEL は、索引構成表に対しては無効です。PARALLEL 句 (4-462 ページ) を参照してください。
	ENABLE TABLE LOCK	パラレル・サーバー環境で表の DML ロックおよび DDL ロックを使用可能にします。詳細は、『Oracle8 Parallel Server 概要および管理』を参照してください。
<i>disable_clause</i>		表に対応付けられた単一の整合性制約またはすべてのトリガーを使用禁止にします。DISABLE 句 (4-375 ページ) を参照してください。  DISABLE 句に指定する整合性制約は、ALTER TABLE 文または以前に出された文に定義されていなければなりません。整合性制約は、CONSTRAINT 句の ENABLE キーワードまたは DISABLE キーワードを使っても使用可能または使用禁止にできます。定義した整合性制約を、明示的に使用可能にも使用禁止にもしない場合は、デフォルトではその制約は使用可能になります。
	DISABLE TABLE LOCK	パラレル・サーバー環境でパフォーマンスを改善するために、表の DML ロックおよび DDL ロックを使用禁止にします。詳細は、『Oracle8 Parallel Server 概要および管理』を参照してください。

列の追加

ADD 句を使用して表に新しい列を追加すると、新しい列の各行には初期値として NULL が設定されます。NOT NULL 制約のある列は、行をまったく含まない表に対してだけ追加できます。

ビューを作成するときの問合せで選択リストにアスタリスク (\*) を指定して実表からすべての列を選択することを指定した後に、実表に列を追加した場合は、新しい列は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には、CREATE VIEW コマンドに OR REPLACE オプションを指定してビューを再作成してください。

ALTER TABLE コマンド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とストアド・ファンクションが無効になる場合があります。無効になる条件と具体的な状況については、『Oracle8 Server 概要』を参照してください。

列定義の変更

MODIFY 句を使用すると、列定義を部分的に変更できます。変更できるものは、データ型またはサイズ、デフォルト値、NOT NULL 列制約です。

**MODIFY** 句には、列の定義をすべて指定する必要はありません。列名と、変更部分だけを指定してください。

## データ型とサイズ

CHAR 型の列を VARCHAR2( または VARCHAR ) 型に変更したり、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 値である場合と列のサイズを変更しない場合だけです。列のデータ型を変更したり列のサイズを小さくしたりできるのは、列のすべての行が NULL 値の場合だけです。文字列と行、列のサイズ、数値列の精度は、いつでも大きくできます。

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。

## デフォルト値

列のデフォルト値を変更すると、変更後に表に挿入される行だけが影響を受けます。この変更によって、変更前に挿入されたデフォルト値は変わりません。

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には、次の例に示すように、デフォルト値を NULL に置き換えます。

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

この文は、既存の行の既存の値には影響しません。

## 整合性制約

列の制約構文と **MODIFY** 句を使用して、既存の列に追加できる唯一の整合性制約は **NOT NULL** 制約です。ただし **ADD** 句と表の制約構文を使用して、既存の列にこれ以外の種類の整合性制約 (**UNIQUE**、**PRIMARY KEY**、参照整合性、**CHECK** 制約) も定義できます。

既存の列に **NULL** がない場合にだけ、この列に **NOT NULL** 制約を定義できます。

**例 1.** 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);
```

**例 2.** 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 つしかないので、定義を囲むカッコは任意指定です。



**例 3.** 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;
```

**例 4.** 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);
```

このコマンドでは 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。

**例 5.** 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ても BAL 列に値を指定しなければ、BAL 列の値は自動的に 0 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
```

## 索引構成表

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特殊な表なので、主キーに基づくアクセスおよび操作には最適です。

列を索引構成表に ADD(追加) できませんが、索引構成表の定義を変更することはできます。

**例 1.** この例では、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;
```

**例 2.** 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;
```

**例 3.** この例では、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;
```

## LOB 列

LOB 列を表へ追加したり、LOB データ・セグメントまたは索引の記憶特性を変更したりできます。

**例 1.** 次の文では、CLOB 列の RESUME を EMPLOYEE 表に追加します。

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

**例 2.** 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);
```

## OBJ ネストした表の列

ネストした表型の列を表に追加できます。追加する各列ごとにネストした表の記憶句を指定してください。

**例 1.** 次の例では、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。変更するには、ネストした表の記憶句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 文を問い合わせたり実行したりできません。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だけに使用します。

**例 2.** **OBJ** 次の例では、ネストした表の列 CLIENT と記憶域表 CLIENT\_TAB を使って表 VETSERVICE を作成します。ネストした表 VETSERVICE は、ネストした記憶域表 CLIENT\_TAB の変更により、制約が指定され、列の長さを変更されます。

```
CREATE TABLE vetSERVICE (vet_name VARCHAR2(30),
  client pet_table)
  NESTED TABLE client STORE AS client_tab;
ALTER TABLE client_tab ADD UNIQUE (ssn);
ALTER TABLE client_tab MODIFY (pet_name VARCHAR2(35));
```

例 4. **OBJ** 次の文では、ネストした表 NESTED\_SKILL\_TABLE に UNIQUE 制約を追加します。

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

ネストした表の記憶領域の詳細は、CREATE TABLE (4-303 ページ) を参照してください。ネストした表については、『Oracle8 Server アプリケーション開発者ガイド』を参照してください。

例 5. 次の例では、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、REF を ROWID とともに格納することを指定するために、次のように入力します。

```
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 が空でなくてはなりません。また、ネストした表は、スカラー (REF) 表として定義されるので、Oracle は、暗黙的に列名 COLUMN\_VALUE を記憶域表に設定します。

## **OBJ** REF

REF 値は、オブジェクト表内の行に対する参照です。表は、最高レベルの REF 列、またはオブジェクト列に埋め込まれた REF 属性を持つことができます。一般的に、表が REF 列を持つ場合は、この列の各 REF 値で別のオブジェクト表内の行を参照できます。SCOPE 句では、1 つの表に対する参照の有効範囲を制限します。

新しい REF 列を追加したり、REF 句を既存の REF 列に追加したりするには、ALTER TABLE コマンドを使用します。名前付きのネストした内部表 (記憶域表) を含むすべての表を変更できます。REF 列を ROWID または有効範囲表付きで作成した場合は、これらのオプションの削除となる列変更はできません。しかし、REF 句なしで表を作成すると、ALTER TABLE 文を使って REF 句を後から追加できます。

**注意：** 有効範囲句は、表が空のときにだけ表の既存の REF 列に追加できます。有効範囲句の追加には *scope\_table\_name* が自スキーマ内にあるか、あるいはその表に対する SELECT 権限または SELECT ANY TABLE システム権限が必要です。この権限は、REF 列が含まれた表を変更するときだけに必要です。

例 1. 次の例では、オブジェクト型 DEPT\_T があらかじめ定義されています。次のように表 EMP 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;
```

DEPARTMENTS 表に考えられるすべての部門が含まれている場合、EMP 内の DEPT 列では、DEPARTMENTS 表の行しか参照できません。これは、次のように、DEPT 列の有効範囲句として表すことができます。

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

上記の ALTER TABLE 文は、EMP 表が空である場合にだけ正常に実行されます。

**例 2.** EMP の DEPT 列に REF 値を格納する場合に ROWID も一緒に格納したいときは、次の文を発行します。

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

## 表パーティションの変更

表または表パーティションの変更は、次のいずれかの方法で行うことができます。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や実表に対する操作と組み合わせて行うことはできません。

### ADD PARTITION

パーティションを表の一番上（既存の最後のパーティションの後）に追加するには、ALTER TABLE ADD PARTITION を使用します。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 である場合は、パーティションは表に追加できません。その場合、上位のパーティションを分割する必要があります。

1 つ以上の表索引または索引パーティションに UNUSABLE のマークが付けられている場合でも、パーティションを表に追加できます。

パーティションを表の始めまたは中間に追加するには、SPLIT PARTITION 句を使用する必要があります。

次の例では、パーティション JAN97 を表領域 TSX に追加します。

```
ALTER TABLE sales
  ADD PARTITION jan97 VALUES LESS THAN( '970201' )
  TABLESPACE tsx;
```

## DROP PARTITION

ALTER TABLE DROP PARTITION を使うと、パーティションとそのデータを削除できます。データを表に残したままパーティションを削除する場合は、そのパーティションを隣接するパーティションにマージする必要があります。2つの表パーティションのマージについては、『Oracle8 Server 管理者ガイド』を参照してください。

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すると、行は1つ後のパーティションに格納されます。しかし、最上位のパーティションを削除すると、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ので、挿入は失敗します。

また、この文は、*table* で定義された各ローカル索引内の対応するパーティションも削除します。索引パーティションは、使用不可のマークが付けられている場合にも削除されます。

*table* にグローバル索引が定義されていて、切り捨てるパーティションが空でない場合、パーティションを切り捨てると、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使用不可のマークが付けられます。

表に1つのパーティションしか含まれていないときには、そのパーティションは削除できません。その表を削除しなければなりません。

次の例では、パーティション DEC95 を削除します。

```
ALTER TABLE sales DROP PARTITION dec95;
```

## EXCHANGE PARTITION

この形式の ALTER TABLE では、それぞれのデータ・セグメントを交換することによって、パーティションを非パーティション表に変換し、NONPARTITIONED 表をパーティションに変換します。この操作を実行するには、両方の表に対する ALTER TABLE 権限が必要です。

表とパーティションの統計（表、列、索引の統計とヒストグラムを含む）が交換されます。パーティション表の集合体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が交換されます。

次の例では、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;
```

## MODIFY PARTITION

ALTER TABLE の MODIFY PARTITION オプションは、次の目的で使用します。

- 表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使用不可のマークを付ける。

- 表パーティションに対応するすべての使用不可のローカル索引パーティションを再構築する。
- 表パーティションの物理属性を変更する。

次の例では、sales 表の NOV96 パーティション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;
```

次の例では、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;
```

次の例では、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;
```

## MOVE PARTITION

ALTER TABLE のこのオプションを使うと、表パーティションが別のセグメントに移動します。MOVE PARTITION では、新しい表領域が指定されていない場合でも、常に、パーティションの古いセグメントを削除し、新しいセグメントを作成します。

*partition\_name* 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およびすべての非パーティション索引、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使用不可のマークが付けられます。

PARALLEL 句が指定されている場合は、ALTER TABLE MOVE PARTITION のパラレル属性は、PARALLEL 句から取得されます。指定されていない場合は、表のデフォルトの PARALLEL 属性があればこれが使用されます。PARALLEL 句が指定されず、デフォルトの PARALLEL 属性もない場合は、並列性を考慮せずに移動が行われます。

MOVE PARTITION の PARALLEL 句は、*table* のデフォルトの PARALLEL 属性を変更しません。

次の例では、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;
```

## RENAME

表を改名したり、パーティションを改名したりするには、ALTER TABLE の RENAME オプションを使用します。

次の例では、表を改名します。

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

次の例では、パーティション EMP3 が改名されます。

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

## SPLIT PARTITION

**SPLIT PARTITION** オプションを使うと、1つのパーティションを2つのパーティションに分割できます。分割の結果作成されたパーティションには、それぞれ新しいセグメントが割り当てられます。新しいパーティションの属性は、分割前のパーティションから継承されます (**SPLIT** 句で明示的に値を変更したものを除く)。古いパーティションに対応付けられていたセグメントは、破棄されます。

また、この文は、*table* で定義された各ローカル索引内の対応するパーティションに対して対応する分割を実行します。索引パーティションは、使用不可のマークが付けられている場合にも分割されます。

**TABLESPACE** 属性を除き、分割する **LOCAL** 索引パーティションの物理属性は、両方の新しい索引パーティションでも使用されます。親 **LOCAL** 索引にデフォルトの **TABLESPACE** 属性がない場合は、新しい **LOCAL** 索引パーティションの表領域は、基となる表から新たに作成された対応するパーティションと同じになります。

新しいパーティションの物理属性 (**PCTFREE**、**PCTUSED**、**INITRANS**、**MAXTRANS**、**STORAGE**) を指定しないと、分割するパーティションの現在の設定値が、両方のパーティションのデフォルト値として使用されます。

*partition\_name* が空でない場合は、**SPLIT PARTITION** によって、影響を受けるすべての索引パーティションに使用不可のマークが付けられます。これには、分割の結果生成されるローカル索引パーティションに加え、すべてのグローバル索引パーティションが含まれます。

**SPLIT PARTITION** の **PARALLEL** 句は、*table* のデフォルトの **PARALLEL** 属性を変更しません。

次の例では、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 (MINEXTENTS 2),
        PARTITION depot9 TABLESPACE ts010 )
  PARALLEL ( DEGREE 10 );
```

## TRUNCATE PARTITION

表内のパーティションからすべての行を削除するには、**TRUNCATE PARTITION** を使用します。解放された領域は、その句に **DROP STORAGE** と **REUSE STORAGE** のどちらが指定されているかによって、割当て解除、または再利用されます。

この文は、*table* に定義された各ローカル索引内の対応するパーティションを切り捨てます。ローカル索引パーティションは、使用不可のマークが付けられている場合にも切り捨てられます。使用不可のローカル索引パーティションは、**UNUSABLE** 標識がリセットされて、有効のマークが付けられます。

*table* にグローバル索引が定義されていて、切り捨てるパーティションが空でない場合、パーティションを切り捨てると、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使用不可のマークが付けられます。

データが入っているパーティションを切り捨てる場合は、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。

次の例では、**SYS\_P017** パーティション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;
```

**ALTER TABLE** コマンドを使った整合性制約の定義の例は、**CONSTRAINT** 句（4-185 ページ）を参照してください。

**ALTER TABLE** コマンドを使用した整合性制約およびトリガーの使用可能化、使用禁止化、削除の例は、**ENABLE** 句（4-414 ページ）および **DISABLE** 句（4-375 ページ）、**DROP** 句（4-379 ページ）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、**STORAGE** 句（4-518 ページ）を参照してください。

## 関連項目

**CREATE TABLE**（4-303 ページ）

**CONSTRAINT** 句（4-185 ページ）

**DISABLE** 句（4-375 ページ）

**DISABLE** 句（4-375 ページ）

**ENABLE** 句（4-414 ページ）

**STORAGE** 句（4-518 ページ）

**CREATE VIEW**（4-359 ページ）



---

## ALTER TABLESPACE

### 用途

次のいずれかの方法で既存の表領域を変更します。

- データ・ファイルを追加する。
- データ・ファイルを改名する。
- デフォルトの記憶領域パラメータを変更する。
- 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。
- 表領域への書込みを許可または禁止する。
- 表領域のデフォルトのロギング属性を変更する。
- 表領域エクステンツ長の最小値を変更する。

「使用上の注意」(4-137 ページ) も参照してください。

### 前提条件

ALTER TABLESPACE システム権限を持っている場合は、このコマンドの操作はどれでも実行できます。MANAGE TABLESPACE システム権限を持っている場合は、次の操作だけを実行できます。

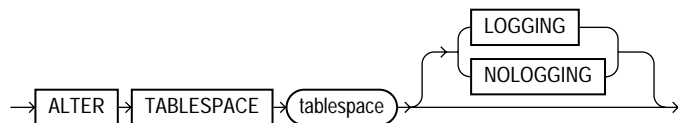
- 表領域をオンラインまたはオフラインにする。
- バックアップを開始または終了する。
- 表領域を読取り専用または読み書き両用にする。

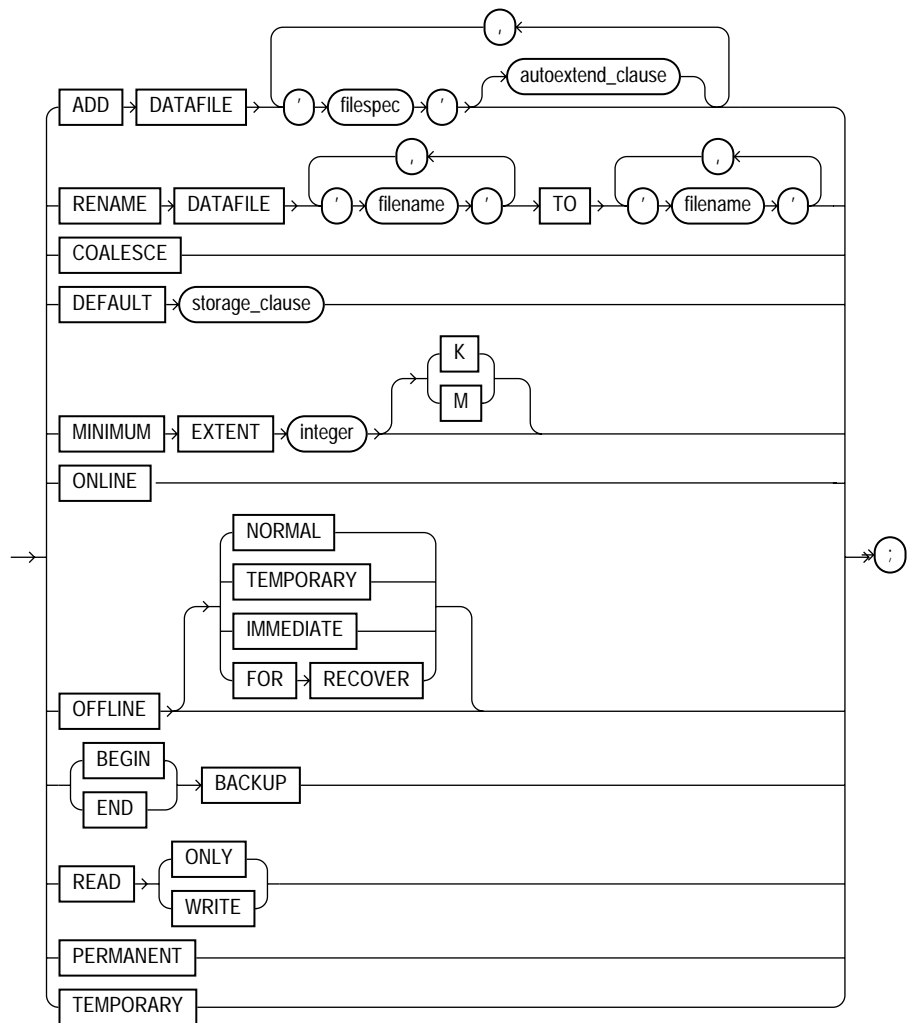
表領域を読取り専用にするには、次の条件が満たされている必要があります。これらの条件を満たすには、制限モードでこの機能を実行すると便利です。制限モードでは、RESTRICTED SESSION システム権限を持つユーザーだけがログインできます。

- 表領域がオンラインになっていること。
- データベースのどこにもアクティブ・トランザクションがないこと。これは、表領域に適用しなければならない取消し情報がないことを確認するために必要です。
- 表領域にアクティブなロールバック・セグメントがないこと。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ので、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をお勧めします。
- 表領域がオープン・バックアップに使用されていないこと。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からです。

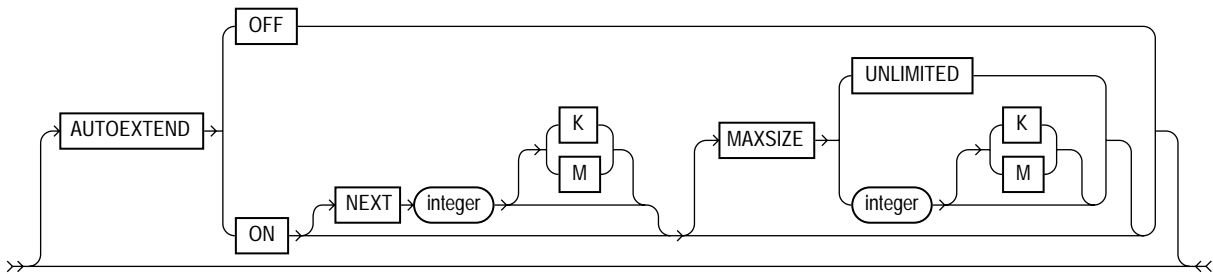
- COMPATIBLE 初期化パラメータが 7.1.0 以上に設定されていること。

### 構文





autoextend\_clause ::=



filespec: 「Filespec」 (4-428 ページ) を参照。  
storage\_clause: STORAGE 句 (4-518 ページ) を参照。

キーワードとパラメータ

<i>tablespace</i>	変更する表領域の名前です。
LOGGING/ NOLOGGING	<p>表領域内のすべての表および索引、パーティションのデフォルトのログギング属性を指定します。</p> <p>表レベルおよび索引レベル、パーティション・レベルでのログギング指定によって、表領域レベルのログギング属性を上書きできます。</p> <p>既存の表領域のログギング属性を ALTER TABLESPACE 文によって変更すると、この文の実行後に作成されたすべての表および索引、パーティションに新しいデフォルトのログギング属性 (これは後で上書きもできます) が適用されます。既存のオブジェクトのログギング属性は変更されません。</p> <p>以下の操作だけが、NOLOGGING モードをサポートします。</p> <ul style="list-style-type: none"><li>• 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)</li><li>• DDL: CREATE TABLE... AS SELECT、CREATE INDEX、ALTER INDEX...REBUILD、ALTER INDEX... REBUILD PARTITION、ALTER INDEX... SPLITPARTITION、ALTER TABLE... SPLIT PARTITION、ALTER TABLE... MOVE PARTITION。</li></ul> <p>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、レコード・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。メディア回復中に NOLOGGING が適用されると、REDO データはログに記録されないため、エクステント無効化レコードはブロック範囲を論理的に破壊されているものとしてマーク設定します。このため、消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とってください。</p>

ADD DATAFILE	<i>filespec</i> に指定されたデータ・ファイルを表領域に追加します。(Filespec (4-428 ページ) の構文の説明を参照。) 表領域がオンラインでもオフラインでもデータ・ファイルを追加できます。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。
AUTOEXTEND	表領域のデータ・ファイルのサイズの自動拡張を使用可能または使用禁止にします。
	<p>OFF                    この設定をすると、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。後で ALTER TABLESPACE AUTOEXTEND コマンドを使用するときに、NEXT および MAXSIZE に値を再指定する必要があります。</p> <p>ON                    自動拡張を使用可能にします。</p> <p>NEXT                  追加のエクステントが必要になったときに、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のサイズ (バイト単位) を指定します。K または M を使用して KB または MB 単位でもサイズを指定できます。デフォルトは 1 データ・ブロックです。</p> <p>MAXSIZE              データ・ファイルの自動拡張で使われるディスク領域の最大サイズを指定します。</p> <p>UNLIMITED          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。</p>
RENAME DATAFILE	<p>1 つまたは複数の表領域のデータ・ファイルを改名します。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 '<i>filename</i>' には、使用しているオペレーティング・システムのファイル名の表記法に従って、データ・ファイル名を完全に指定してください。</p> <p>この句は、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るだけです。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。</p>
COALESCE	<p>表領域内の各データ・ファイルについて、連続する使用可能エクステントをすべて結合して大きな連続エクステントにします。</p> <p>COALESCE は、他のコマンド・オプションと併用できません。</p>
DEFAULT <i>storage_</i> <i>clause</i>	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。STORAGE 句 (4-518 ページ) を参照してください。
MINIMUM EXTENT <i>integer</i>	表領域内のすべての使用済みエクステントまたは未使用エクステント (あるいは両方) の大きさが、 <i>integer</i> で指定したサイズ以上であり、なおかつその倍数になるように指定することにより、表領域における空き領域の断片化を制御します。MINIMUM EXTENT を使用した領域の断片化制御については、『Oracle8 Server 管理者ガイド』を参照してください。
ONLINE	表領域をオンラインにします。
OFFLINE	表領域をオフラインにして、そのセグメントにアクセスできないようにします。
	<p>NORMAL              表領域内のすべてのデータ・ファイルに対してチェックポイントを実行します。このデータ・ファイルはすべてオンラインでなければなりません。これはデフォルト値です。データ・ファイルをオンラインに戻す前に表領域のメディア回復を行う必要はありません。データベースが NOARCHIVELOG モードの場合は、このオプションを使用してください。</p>

TEMPORARY	表領域内のすべてのオンライン・データ・ファイルに対してチェックポイントを実行しますが、すべてのファイルに書き込むことができません。この表領域をオンラインに戻す前に、オフライン・ファイルのメディア回復を行う必要があります。
IMMEDIATE	表領域のファイルは使用可能とは限りません。また、チェックポイントも実行しません。表領域をオンラインに戻す前に、メディア回復を行わなければならないかもしれません。
FOR RECOVER	回復設定オフライン内に稼動データベースの表領域を確保します。このオプションは、表領域内の1つ以上のデータ・ファイルが使用不可能な場合に使います。
<p><b>提案：</b>長期にわたって表領域をオフラインにする前に、デフォルト表領域または一時表領域としてその表領域を割り当てられているユーザー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すると、これらのユーザーはその表領域内でオブジェクトに対する領域やソート領域を割り当ててすることはできません。<b>ALTER USER</b> コマンドを使用すると、これらの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再度割り当てることができます。</p>	
BEGIN BACKUP	<p>指定した表領域を構成するデータ・ファイルのオープン・バックアップの実行を示します。このオプション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オプションを指定してください。このオプションは、読取り専用表領域に対しては使用できません。</p> <p><b>注意：</b>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、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。</p>
END BACKUP	<p>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オプションを指定してください。このオプションは、読取り専用表領域に対しては使用できません。</p> <p>オンラインの表領域バックアップの終わりを指定し忘れて、インスタンス障害や <b>SHUTDOWN ABORT</b> 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回復（アーカイブ REDO ログ・ファイルを使う場合もあります）が必要になるとみなされます。メディア回復を行わずにデータベースを再起動する方法については、『Oracle8 Server 管理者ガイド』を参照してください。</p>
READ ONLY	<p>表領域に対して今後書き込み操作ができないことを示します。表領域は読取り専用になります。</p> <p>表領域を読取り専用にすると、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 コマンドの <b>ALTER DATABASE RENAME</b> を使って、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。</p>
READ WRITE	読取り専用だった表領域に対して書き込み操作をできるようにすることを示します。
PERMANENT	一時的な表領域を永続的な表領域に変換するように指定します。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。
TEMPORARY	永続的な表領域を一時的な表領域に変換するように指定します。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。

## 使用上の注意

次の例では、ALTER TABLESPACE コマンドの使用方法を示しています。

**例 1.** 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;
```

**例 2.** 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;
```

**例 3.** 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 オプションを指定した ALTER TABLESPACE 文を使って、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 コマンドを使って、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 オプションを指定した ALTER TABLESPACE 文を使って、この表領域をオンラインに戻します。

```
ALTER TABLESPACE accounting ONLINE;
```

**例 4.** 次の文では、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。追加の領域が必要なときは、新たなエクステントが 10KB ずつ、最大 100KB まで追加されます。

```
ALTER TABLESPACE accounting NOLOGGING
  ADD DATAFILE 'disk3:pay3.dbf'
  AUTOEXTEND ON
  NEXT 10 K
  MAXSIZE 100 K;
```

表領域のロギング属性を変更しても、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。表レベルおよび索引レベル、パーティション・レベルでのロギング方式を指定することによって、表領域レベルのロギング属性を変更できます。

**例 5.** 次の文は、TABLESPACE\_ST の各エクステンツの割当てを 128K の倍数に変更します。

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

## 関連項目

CREATE TABLESPACE (4-325 ページ)

CREATE DATABASE (4-214 ページ)

DROP TABLESPACE (4-404 ページ)

STORAGE 句 (4-518 ページ)

「Filespec」 (4-428 ページ)



# ALTER TRIGGER

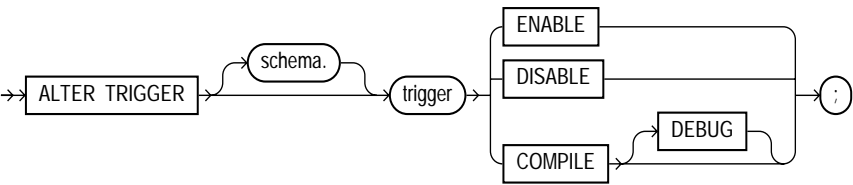
## 用途

データベース・トリガーを使用可能化または使用禁止化、コンパイルします。

## 前提条件

トリガーが自スキーマ内にあることが必要です。自スキーマ内にはない場合は、ALTER ANY TRIGGER システム権限が必要です。

## 構文



## キーワードとパラメータ

<i>schema</i>	変更するトリガーが定義されているスキーマです。 <i>schema</i> を指定しないと、トリガーは自スキーマ内に定義されているものとみなされます。	
<i>trigger</i>	変更するトリガーの名前です。「無効トリガー」(4-139 ページ) も参照してください。	
ENABLE	トリガーを使用可能にします。「トリガーの使用可能と使用禁止の切替え」(4-140 ページ) も参照してください。	
DISABLE	トリガーを使用禁止にします。「トリガーの使用可能と使用禁止の切替え」(4-140 ページ) も参照してください。	
COMPILE	トリガーをコンパイルします。	
DEBUG	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。このオプションは、標準トリガーおよび代替トリガーに使用できます。

## 無効トリガー

ALTER TRIGGER コマンドを使用すると無効なトリガーを明示的に再コンパイルできます。明示的に再コンパイルすることによって、実行時に暗黙的に再コンパイルされる必要がなく

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。

ALTER TRIGGER 文を発行すると、指定したトリガーは有効か無効にかかわらず再コンパイルされます。

トリガーを再コンパイルすると、その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終了すれば、このトリガーは有効になります。トリガーの再コンパイル時にエラーが発生すると、エラーが戻り、そのトリガーは無効なままです。この場合、事前定義済みパッケージ DBMS\_OUTPUT を使用してトリガーをデバッグできます。プロシージャのデバッグの説明は、『Oracle8 アプリケーション開発者ガイド』を参照してください。リモート・オブジェクトを含むスキーマ・オブジェクトの依存性を Oracle がメンテナンスする方法については、『Oracle8 Server 概要』を参照してください。

---

---

**注意：** このコマンド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には、OR REPLACE オプションを指定した CREATE TRIGGER コマンドを使用する必要があります。

---

---

## トリガーの使用可能と使用禁止の切替え

データベース・トリガーは必ず使用可能または使用禁止のどちらかになっています。トリガーが使用可能になっている場合は、トリガーを起動する文の発行時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場合は、トリガーを起動する文を発行してもトリガーは起動されません。

トリガーを作成すると、そのトリガーは自動的に使用可能になります。ALTER TRIGGER コマンドの ENABLE オプションまたは DISABLE オプションを使用すると、トリガーを使用可能または使用禁止にできます。

また、ALTER TABLE コマンドの ENABLE 句または DISABLE 句を使用することにより、表に対応付けられたすべてのトリガーを使用可能または使用禁止にできます。

---

---

**注意：** ALTER TRIGGER コマンドを使用して既存のトリガーの定義は変更できません。トリガーを再定義するには、必ず OR REPLACE オプションを指定した CREATE TRIGGER コマンドを使用します。

---

---

**例 .** INVENTORY 表に対して作成された REORDER という名前のトリガーを例にとります。UPDATE 文によってある部品の在庫数と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、および再発注の数量、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となります。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とその再発注点を下回ってもトリガーは起動されません。

トリガーを使用禁止にした後で、次の文を使用してこれを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すると、UPDATE 文によって部品の在庫数と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となっている間に部品の在庫数とその再発注点を下回っている可能性があるので注意してください。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されない限り、その部品についてはこのトリガーは自動的に起動されません。

## 関連項目

CREATE TRIGGER (4-330 ページ)

DROP TRIGGER (4-406 ページ)

DISABLE 句 (4-375 ページ)

ENABLE 句 (4-414 ページ)

ALTER TABLE (4-105 ページ)

OBJ ALTER TYPE

用途

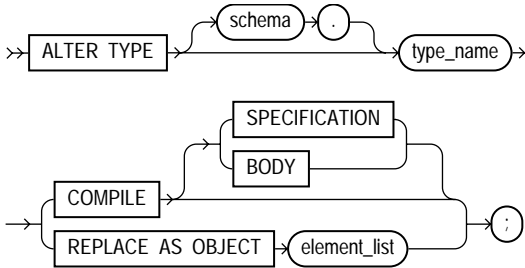
新しいオブジェクト・メンバーのサブプログラム仕様を追加することによって、オブジェクト型の仕様部と本体の両方またはどちらか一方を再コンパイル、あるいはオブジェクト型の仕様を変更します。

注意： このコマンドは、使用しているデータベース・サーバーに Oracle Object Option がインストールされている場合にだけ有効です。

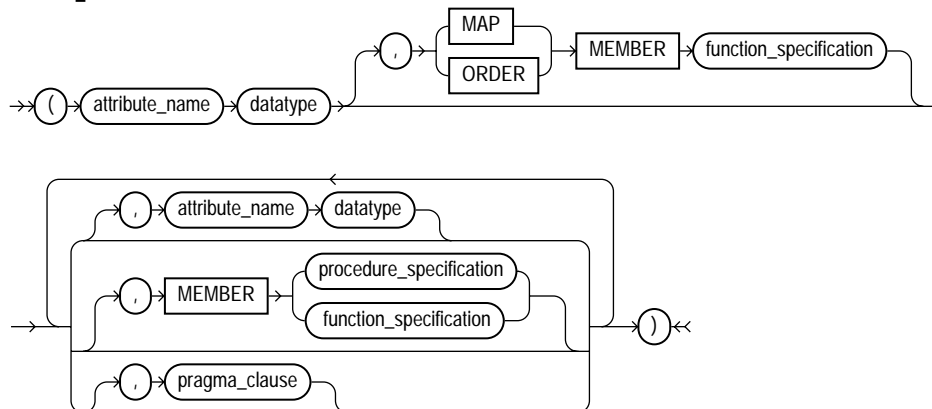
前提条件

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 のシステム権限を持っている必要があります。それ以外の場合は、ALTER ANY TYPE システム権限が必要です。

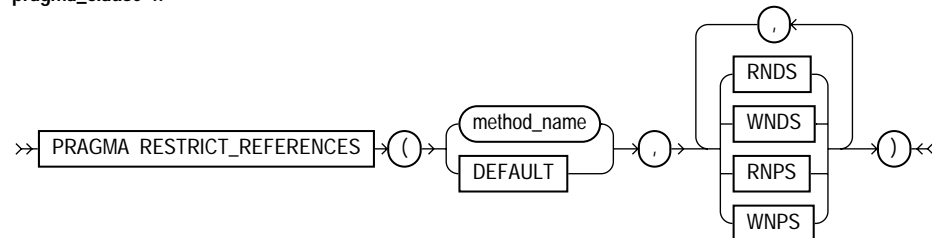
構文



element\_list ::=



pragma\_clause ::=



## キーワードとパラメータ

<i>schema</i>	型を定義するスキーマです。 <i>schema</i> を指定しないと、現行スキーマにその型が作成されます。
<i>type_name</i>	オブジェクト型またはネストした表型、 <b>VARRAY</b> 型の名前です。
COMPILE	オブジェクト型の仕様部と本体をコンパイルします。これは、オプションが指定されていないときのデフォルト値です。
SPECIFICATION	オブジェクト型の仕様部だけをコンパイルします。
BODY	オブジェクト型の本体だけをコンパイルします。
REPLACE AS OBJECT	新しいメンバーのサブプログラム仕様を追加します。このオプションは、オブジェクト型だけに有効です。
<i>attribute_name</i>	オブジェクトの属性名です。属性とは、名前と型指定子を持つオブジェクトの構造を形成するデータ項目です。

MAP/ORDER MEMBER *function\_specification*

## 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関数 (MAP メソッド) を指定します。MAP メソッドは暗黙にコールされ、オブジェクト・インスタンスを事前定義済みの *scalar* 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・インスタンスに順番を設定します。PL/SQL は、この順序を使ってブール式の計算と比較を行います。

*scalar* 値は、常に 1 つの単位として処理されます。スカラーは、基礎となるハードウェアに直接マップされます。たとえば、整数は、メモリー内またはディスク上の連続した 4 ～ 8 バイトの記憶域を占有します。

オブジェクトの仕様部には、1 つのマップ・メソッドだけ入れることができます。このマップ・メソッドは関数でなければなりません。結果の型は、事前定義済みの SQL スカラー型でなければならず、マップ関数には、暗黙的な SELF 引数以外の引数は指定できません。

## 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 引数としてとるメンバー関数 (ORDER メソッド) を指定し、負の整数、またはゼロ、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、ゼロは両方が等しいことを示し、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。

同じオブジェクト型定義の各インスタンスを ORDER BY 句の中で比較すると、順序メソッドの *function\_specification* が呼び出されます。

オブジェクトの仕様部には、1 つの ORDER メソッドしか指定できず、その ORDER メソッドは戻り型が INTEGER である関数でなければなりません。

MAP メソッドまたは ORDER メソッドはどちらか一方を宣言できますが、両方は宣言できません。どちらかのメソッドを宣言すると、SQL 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と、オブジェクト・インスタンスの同等性と不等性しか比較できません。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対が等しい場合にだけ同等になるので注意してください。2 つのオブジェクト型が等しいかどうかを判断するために、比較方法を指定する必要はありません。オブジェクト値の比較の説明は、「オブジェクト値」(2-25 ページ) を参照してください。

## MEMBER

属性として参照されるオブジェクト型に関連付ける関数またはプロシージャ・サブプログラムを指定します。パッケージ内のサブプログラム名のオーバーロードの説明は、『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。「使用上の注意」(4-145 ページ) も参照してください。

各プロシージャまたは関数の仕様部について、オブジェクト型本体の中に対応するメソッド本体を指定する必要があります。CREATE TYPE BODY (4-350 ページ) を参照してください。

*procedure\_specification*

プロシージャ・サブプログラムの仕様部です。

	<i>function_specification</i>	関数サブプログラムの仕様部です。
PRAGMA RESTRICT_REFERENCES		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関数の読み書きアクセスを拒否し、副次作用の発生を防止するコンパイラ指示です。詳細は、『PL/SQL ユーザー・ガイドおよびリファレンス』を参照してください。
	<i>method_name</i>	プラグマが適用されている MEMBER 関数またはプロシージャの名前です。
	WNDS	「データベース状態の書き込み禁止」制約（データベースの表を変更しない）を指定します。
	WNPS	「パッケージ状態の書き込み禁止」制約（パッケージ変数を変更しない）を指定します。
	RNDS	「データベース状態の読取り禁止」制約（データベースの表の問い合わせない）を指定します。
	RNPS	「パッケージ状態の読取り禁止」制約（パッケージ変数を参照しない）を指定します。

## 使用上の注意

オブジェクト型の既存のプロパティ（属性またはメンバー・サブプログラム、マップ関数、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。

例 1. 次の例では、メンバー関数 QTR が DATA\_T の型定義に追加されます。

```
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
```

```
BEGIN
    RETURN (year + invent);
END;
MEMBER FUNCTION qtr(der_qtr DATE) RETURN CHAR IS
BEGIN
    RETURN 'FIRST';
END;
END;
```

**例 2.** 次の例では、型 LOAN\_T を再コンパイルします。

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

```
ALTER TYPE loan_t COMPILE;
```

**例 3.** 次の例では、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;
```

**例 4.** 次の例では、LINK2 の型仕様部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;
```

## 関連項目

CREATE TYPE (4-342 ページ)

CREATE TYPE BODY (4-350 ページ)

『PL/SQL ユーザーズ・ガイドおよびリファレンス』

『Oracle8 Server アプリケーション開発者ガイド』

---

## ALTER USER

### 用途

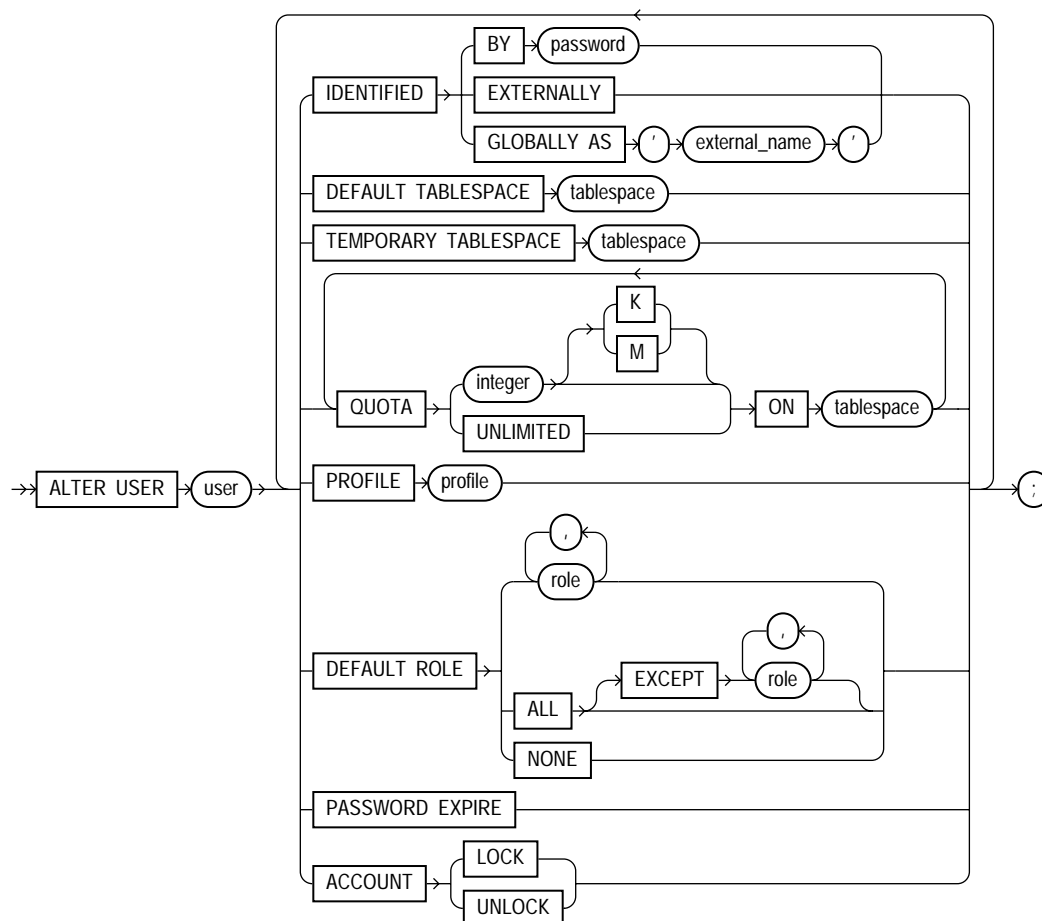
データベース・ユーザーの次の特性を変更します。

- ユーザー認証方式
- パスワード
- オブジェクト作成のためのデフォルト表領域
- ユーザーのために作成された一時セグメントの表領域
- 表領域のアクセスと表領域の割当て制限
- データベース・リソースに関する制限
- デフォルト・ロール

### 前提条件

ALTER USER 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くても変更できます。

## 構文



## キーワードとパラメータ

ALTER USER コマンドのキーワードとパラメータの意味はすべて CREATE USER コマンドのキーワードとパラメータと同じです。これらのキーワードとパラメータについては、CREATE USER（4-353 ページ）を参照してください。

デフォルト・ロールの詳細は、「デフォルト・ロールの設定」（4-150 ページ）を参照してください。セキュリティ・ドメインの詳細は、「認証方式の変更」（4-150 ページ）を参照してください。

## デフォルト・ロールの設定

DEFAULT ROLE 句には、GRANT 文を使用してユーザーに直接付与されているロールだけを指定できます。DEFAULT ROLE 句を使用して次のロールは使用可能にできません。

- 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 Security Service Certification Authority によって管理されるロール

なお、Oracle では、ユーザーがパスワードを指定しなくて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。

**例 1.** 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;
```

**例 2.** 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、SCOTT は CLERK プロファイル内の制限に従います。

**例 3.** 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 の次のセッションの開始時に、AGENT を除き、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。

## 認証方式の変更

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は、ユーザー・アクセス検証方法は IDENTIFIED GLOBALLY AS '*external\_name*' にしか変更できません。

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。

**例 1** 次の例では、ユーザー TOM の認証方式を変更します。

```
ALTER USER tom IDENTIFIED GLOBALLY AS 'CN=tom';
```

**例 2** 次の例では、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 を使ってデータベース・ユーザーのパスワードを期限切れにさせた場合は、そのユーザーは、後でデータベースにログインするにはパスワードを変更する必要があります。ただし、SQL\*Plus などのツールを使用すると、期限切れ後の最初のログイン時にパスワードを変更できます。

## 関連項目

CREATE PROFILE (4-261 ページ)

CREATE ROLE (4-268 ページ)

CREATE USER (4-353 ページ)

CREATE TABLESPACE (4-325 ページ)

# ALTER VIEW

## 用途

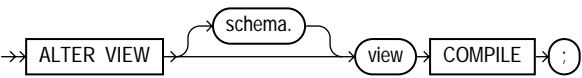
ビューまたはオブジェクト・ビューを再コンパイルします。「使用上の注意」(4-152 ページ)を参照してください。

注意： **OBJ** コマンドと句の説明での印が前に付いている箇所は、Oracle Object Option がデータベース・サーバーにインストールされている場合にだけ有効です。

## 前提条件

ビューが自スキーマ内にあることが必要です。自スキーマ内にはない場合は、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

<i>schema</i>	再コンパイルするビューが定義されているスキーマです。 <i>schema</i> を指定しないと、ビューは自スキーマ内にあるとみなされます。
<i>view</i>	再コンパイルするビューの名前です。
COMPILE	指定したビューが再コンパイルされます。COMPILE キーワードは必須です。

## 使用上の注意

ALTER VIEW コマンドを使用して無効なビューを明示的に再コンパイルできます。明示的に再コンパイルすると、実行前にコンパイル・エラーを検査できます。再コンパイルは、ビューの実表を変更した後で、その変更がビューまたはそのビューに依存するオブジェクトに影響していないかを確認するときに便利です。

ALTER VIEW 文を発行すると、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。スキーマ・オブジェクト間の依存性の詳細は、『Oracle8 Server 概要』を参照してください。

---

**注意：** このコマンドを使用して既存のビューの定義は変更できません。ビューを再定義するには、OR REPLACE オプションを指定した CREATE VIEW コマンドを使用してください。

---

**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、このビューは有効となります。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、このビューは無効のままです。

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、関数、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すると、Oracle は実行時にそれらを暗黙的に再コンパイルします。

## 関連項目

CREATE VIEW (4-359 ページ)

## ANALYZE

### 用途

索引または索引パーティション、あるいは表または表パーティション、索引構成表、クラスタに対して次の処理を実行します。

- オプティマイザが使用するスキーマ・オブジェクトの統計情報を収集してデータ・ディクショナリに格納する。
- データ・ディクショナリからスキーマ・オブジェクトに関する統計情報を削除する。
- スキーマ・オブジェクトの構造を検証する。
- 表やクラスタの移行された行と連鎖行を識別する。
- **OBJ** スカラー・オブジェクト属性の統計情報を収集する。
- **OBJ** オブジェクト参照 (REF) の妥当性検査および更新を行う。

---

---

**注意：** コマンドと句の説明で**OBJ**の印が前に付いている箇所は、Oracle Object Option がデータベース・サーバーにインストールされている場合にだけ有効です。

---

---

### 前提条件

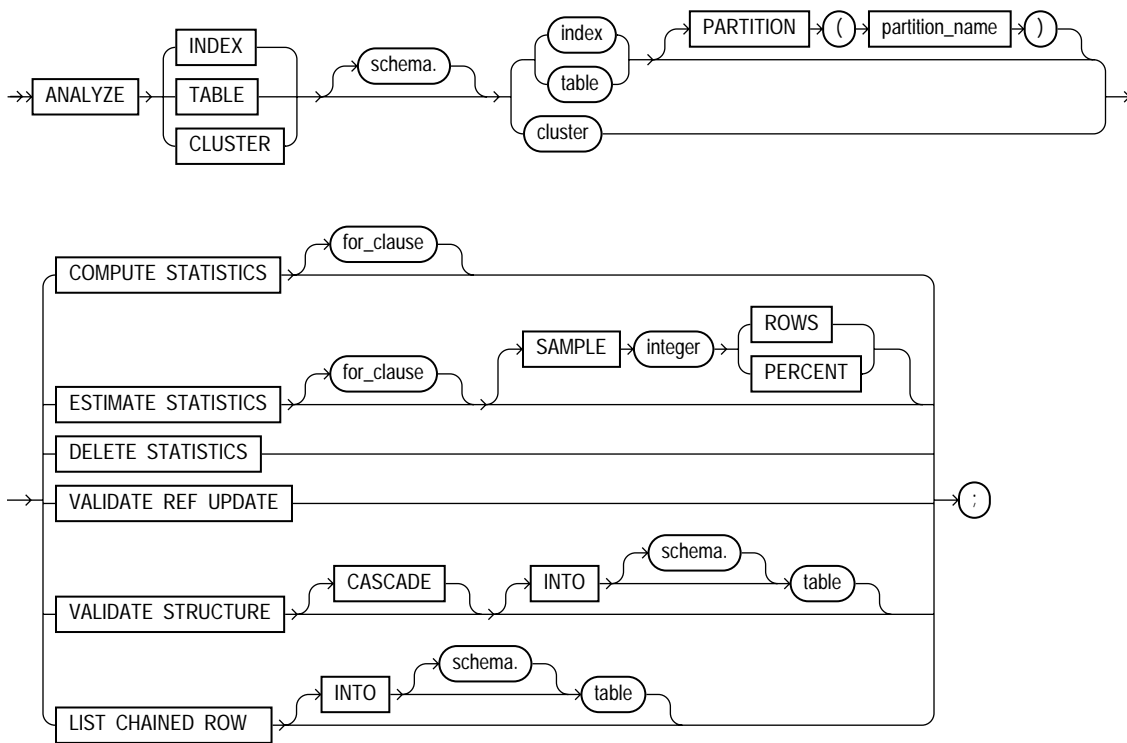
分析するスキーマ・オブジェクトが自スキーマ内にある必要があります。自スキーマ内にならない場合は、ANALYZE ANY システム権限が必要です。

表や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なければなりません。自スキーマ内にならない場合は、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検査を行う場合は、分析した ROWID を入れる表に対する INSERT 権限を持っているか、または INSERT ANY TABLE システム権限を持っている必要があります。

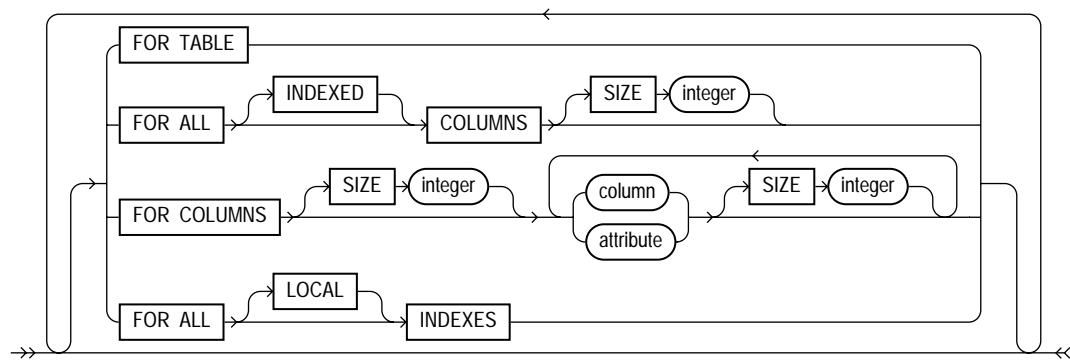
「使用上の注意」（4-158 ページ）も参照してください。



## 構文



**for\_clause ::=**



キーワードとパラメータ

<i>schema</i>	索引または表、クラスタが含まれているスキーマです。 <i>schema</i> を指定しないと、索引または表、クラスタは自スキーマ内にあるとみなされます。
<i>index</i>	分析する索引を指定します (FOR 句を使用しない場合)。
<i>table</i>	分析する表を指定します。FOR 句を使用しない場合は、表の統計情報を収集すると各表の索引の統計情報も自動的に収集されます。
PARTITION	( <i>partition_name</i> ) の統計情報収集を指定します。なお、クラスタの分析時にはこのオプションは使用できません。
<i>cluster</i>	分析するクラスタを指定します。クラスタの統計情報を収集すると、すべてのクラスタ表、およびクラスタ索引を含むすべての索引の統計も自動的に収集されます。「クラスタ」(4-160 ページ) も参照してください。
 VALIDATE REF UPDATE	指定された表の中の REF の妥当性検査を行い、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。まちがっている場合は訂正します。このオプションは、表を分析するときにはしか使用できません。
COMPUTE STATISTICS	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。「統計情報の収集」(4-158 ページ) を参照してください。
ESTIMATE STATISTICS	分析対象の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。
SAMPLE	統計情報を概算するために、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と、サンプルとして 1064 行が抽出されます。データの半分以上を指定すれば、すべてのデータが読み取られ統計が算出されます。
ROWS	Oracle は表またクラスタについて <i>integer</i> で指定する行数、または索引から <i>integer</i> で指定するエントリ数をサンプルとして抽出します。 <i>integer</i> は 1 以上でなければなりません。
PERCENT	Oracle は表またはクラスタの行または索引のエントリについて、 <i>integer</i> で指定するパーセントをサンプルとして抽出します。 <i>integer</i> は 1 ～ 99 の範囲で指定します。
<i>for_clause</i>	表または索引全体を分析するのか、特定の列だけを分析するのかを指定します。次に示す句は、このコマンドの ANALYZE TABLE にだけ使用できます。
FOR TABLE	表の統計情報を収集します。
FOR ALL COLUMNS	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。
	INDEX は表の列のうち、索引付きの列すべてについての列統計情報を収集します。
FOR COLUMNS	指定された列およびスカラー・オブジェクト属性の列統計情報を収集します。

	<b>OBJ</b> <i>attribute</i>	オブジェクト内の項目の修飾した列名を指定します。
	FOR ALL INDEXES	表に対応付けられている索引をすべて分析します。
	FOR ALL LOCAL INDEXES	すべてのローカル索引パーティションの分析を指定します。 <b>PARTITION</b> ( <i>partition_name</i> ) 句と索引を指定する場合は、キーワード <b>LOCAL</b> を指定する必要があります。
	SIZE	ヒストグラムのパーティションの最大数を指定します。デフォルト値は 75、最小値は 1、最大値は 254 です。
	ヒストグラムについては、『Oracle8 Server チューニング』を参照してください。「列」(4-159 ページ) も参照してください。	
DELETE STATISTICS	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。「統計情報の削除」(4-161 ページ) を参照してください。	
VALIDATE STRUCTURE	分析対象のオブジェクトの構造を検証します。クラスタの分析時にこのオプションを指定すると、そのクラスタの表の構造が自動的に検証されます。パーティション表を分析するときこのオプションを指定すると、該当するパーティションに属する行の検証も行われます。「構造の検証」(4-161 ページ) も参照してください。	
	INTO	正しく照合されなかった行を持つパーティションの <b>ROWID</b> を格納するリスト表を指定します。 <i>schema</i> を指定しないと、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と、表の名前は <b>INVALID_ROWS</b> となります。この表を作成するために使用する SQL スクリプトは <b>UTLVALID.SQL</b> です。
	CASCADE	表またはクラスタに関連する索引の構造を検証します。表の検証時にこのオプションを指定すると、その表の索引も検証されます。クラスタの検証時にこのオプションを指定すると、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。
LIST CHAINED ROWS	分析対象の表やクラスタの移行された行と連鎖行を識別します。なお、索引の分析時にこのオプションは使用できません。	
	INTO	移行された行と連鎖行のリスト表を指定します。 <i>schema</i> を指定しないと、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と、表の名前は <b>CHAINED_ROWS</b> になります。この表を作成するために使用するスクリプトは <b>UTLCHAIN.SQL</b> です。このリスト表はローカル・データベース内になければなりません。
	索引構成表を分析するには、その索引構成表の主キーの記憶領域を確保するために、索引構成表ごとに個別の連鎖行の表を作成する必要があります。まず <b>BUILD_CHAIN_ROWS_TABLE</b> パッケージを定義するために SQL スクリプトの <b>DBMSIOTC.SQL</b> と <b>PRVTIOTC.PLB</b> を使用します。次に索引構成表の <b>IOT_CHAINED_ROWS</b> 表を作成するためにこのプロシージャを実行します。	
	「連鎖行のリスト」(4-163 ページ) も参照してください。	

## 使用上の注意

ANALYZE を使って、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。

次の列型の統計値は計算および推定できません。

- REF
- VARRAY
- ネストした表
- LOB( 分析されずにスキップされます )
- LONG
- オブジェクト型

## 統計情報の収集

物理記憶特性および索引、表、列、クラスタのデータ分布に関する統計情報を収集して、データ・ディクショナリに格納できます。統計情報の計算と概算の違いを次に示します。

- 計算では正確な値を得ることができますが、概算より処理時間が余計にかかります。
- 概算は処理時間が短くて済み、ほぼ正確な値を得ることができます。

厳密な値が必要な場合以外は、計算ではなく概算を指定してください。計算と概算のどちらを指定した場合でも常に厳密に計算される統計項目もあります。一般に、概算を選択しても処理時間が短縮されない場合は、その統計項目については厳密な計算が行われています。

データ・ディクショナリに分析済みのオブジェクトの統計情報がすでに格納されている場合は、既存の統計値は新しい統計値に更新されます。

**例 1.**  次の文では、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;
```

この統計情報は、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。これらの統計情報の使用方法については、『Oracle8 Server チューニング』を参照してください。

次に、索引、表、列、クラスタについて収集される統計情報の一覧を記載します。アスタリスク (\*) が付いた統計は常に厳密に計算されます。

## 索引

索引については次の統計情報が収集されます。

- ルート・ブロックからリーフ・ブロックまでの索引の深さ \*
- リーフ・ブロックの数

- 索引の重複していない値の数
- 索引の値ごとのリーフ・ブロックの平均数
- (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
- 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)

索引についての統計情報は、`USER_INDEXES`、`ALL_INDEXES`、`DBA_INDEXES` の各データ・ディクショナリ・ビューに表示されます。

## 表

表については次の統計情報が収集されます。

- 行数
- 現在データが入っているデータ・ブロックの数 \*
- 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 \*
- 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)
- 連鎖行の数
- 行のオーバーヘッドを含む、行の平均の長さ (バイト単位)

表の統計情報は、データ・ディクショナリ・ビュー `USER_TABLES`、`ALL_TABLES`、`DBA_TABLES` に示されます。

## 列

列の統計情報を取得するには、列全体に基づいた情報を取得するか、ヒストグラムを使用します。ヒストグラムでは、列の値が一定の範囲ごとにいくつかの区間に分割されます。必要に応じて、各区間に分割された値の数を確認できます。Oracle ヒストグラムは、等幅タイプではなく等高タイプです。つまり、各区間に入る値の数がほぼ同じになるように列の値が分割されます。このタイプのヒストグラムでは、各区間の終端位置が有用なデータとなります。これに対して等幅タイプのヒストグラムでは、同じ区間幅で値が分割され、各区間に入る値の数が示されます。

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

## ヒストグラムが有効なとき

データの分布が均一な場合には、コストベースの方法を使用して必要な文を実行することでかなり正確な概算が得られます。しかし分布が均一でないときには、特定の列についての

データ分布が記述されたヒストグラムを格納しておくことができます。ヒストグラムはディクショナリに格納され、コストベースのオプティマイザから使用できます。

ただし、ヒストグラムは永続オブジェクトなので、メンテナンスと格納領域というコストがかかります。このためヒストグラムは、データの均一性がきわめて低い列についてだけ利用するようにしてください。また、オプティマイザのすべての統計情報だけでなく、ヒストグラムの情報も静的であることに注意してください。列のデータ分布が頻繁に変化するときには、ANALYZE コマンドを再発行して、その列のヒストグラムを再計算する必要があります。

次の特性を持つ列では、ヒストグラムはあまり有用ではありません。

- 列に対するすべての述語がバインド変数を使用する
- 列データが不均一に分散している
- 問合せの WHERE 句で列が使われていない
- 列が一意で、等価述語だけで使用されている

ヒストグラムは、問合せの WHERE 句の中で頻繁に使用されている列や、データ分布の偏りがきわめて高い列について作成してください。ヒストグラムを作成するには、このコマンドの ANALYZE TABLE を使用します。たとえば、EMP 表の SAL 列について 10 区間のヒストグラムを作成する場合、次の文を使用します。

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

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。次の文では、EMP 表のパーティション P1 を分析します。

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

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、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\_HISTOGRAMS、DBA\_HISTOGRAMS、ALL\_HISTOGRAMS に表示されます。

## クラスタ

索引クラスタには、単一クラスタ・キーの値とそのすべての行が使用するデータ・ブロックの平均数が収集されます。ハッシュ・クラスタには、単一ハッシュ・キーの値とそのすべての行が使用するデータ・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 SER\_CLUSTERS と DBA\_CLUSTERS に表示されます。

**例 2.** 次の文は、CUST\_HISTORY 表とそのすべての索引の統計情報を概算します。

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

## 統計情報の削除

ANALYZE コマンドの DELETE STATISTICS オプションを使用して、データ・ディクショナリからオブジェクトに関する既存の統計情報を削除できます。たとえば、Oracle オプティマイザが統計情報を使用しないようにする場合に、その統計情報を削除します。

表を指定して DELETE STATISTICS オプションを使用すると、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 DELETE STATISTICS オプションを使用すると、指定したクラスタの表およびクラスタ索引を含む、すべての索引の統計情報が自動的に削除されます。

**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;
```

## 構造の検証

ANALYZE コマンドの VALIDATE STRUCTURE オプションを使用して、索引および表、クラスタの各構造の整合性を検証できます。これらの構造の検証が正常終了すると、妥当性が検証されたことを示すメッセージが戻ります。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。

オブジェクトの構造の妥当性検査をすると、そのオブジェクトに対して SELECT 文および INSERT 文、UPDATE 文、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、稼動アプリケーションの表およびクラスタ、索引に対してこのオプションを使わないでください。

### 索引

索引に対する VALIDATE STRUCTURE オプションは、指定した索引の各データ・ブロックの整合性の検証とブロックの欠陥の検査を行います。なお、このオプションは、表の各行が索引エントリを持っていることや、各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 オプションを使用して表の構造を検証します。

索引に VALIDATE STRUCTURE オプションを使用すると、その索引に関する統計も収集され、データ・ディクショナリ・ビュー INDEX\_STATS に格納されます。以前に妥当性検査が行われた索引についての既存の統計情報がある場合は、すべて上書きされます。INDEX\_STATS には、常に 1 つの索引を記述する行が 1 行だけ設定されます。INDEX\_STATS ビューについては、『Oracle8 Server リファレンス・マニュアル』を参照してください。

このオプションで収集した統計は、Oracle オプティマイザでは使用されません。こうした統計情報と、COMPUTE STATISTICS オプションや ESTIMATE STATISTICS オプションで収集した統計情報とは異なります。混同しないように注意してください。

例 1. 次の文は、索引 PARTS\_INDEX の構造を検証します。

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

## 表

表に対する VALIDATE STRUCTURE オプションは、指定した表の各データ・ブロックと行の整合性を検証します。CASCADE オプションを使用すると、その表についてのすべての索引の構造を検証し、表と各索引間の相互参照を実行できます。各索引については、相互参照で次の妥当性検査が行われます。

- 表の索引付けされた列の各値は、索引エントリの索引付けされた列の値と一致しなければならない。一致する索引エントリは、正しい ROWID に基づいて表の行に対応していなければなりません。
- 索引の各エントリは表の行に対応する。索引エントリの索引付けされた列値は、対応する行と一致しなければなりません。

例 2. 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;
```

**OBJ** 表に対する VALIDATE REF UPDATE オプションは、指定した表の REF を検証します。また、各 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 が誤っていると判別されると、ROWID 部分が正しくなるように REF が更新されます。

例 3. **OBJ** 次の文では、EMP 表内の REF の妥当性検査を行います。

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

## クラスタ

クラスタに対する VALIDATE STRUCTURE オプションは、指定したクラスタの行の整合性を検証し、クラスタの各表の構造も自動的に検証します。CASCADE オプションを使用すると、クラスタ索引を含めて、クラスタの表に対するすべての索引の構造も検証できます。

例 4. 次の文は、ORDER\_CUSTS クラスタ、このクラスタのすべての表、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;
```

## パーティション表

パーティション表にはルールベースのオプティマイザは使用できないので、パーティション表と索引を定期的に分析することが重要です。



パーティション表に対する **VALIDATE STRUCTURE** オプションでは、パーティション内の各行が検証され、パーティション・キーの値が、そのパーティションのパーティション境界より小さく、直前のパーティション（最初のパーティションは除く）のパーティション境界より大きいかどうかをチェックします。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。

## 連鎖行のリスト

**ANALYZE** コマンドの **LIST** オプションを使用して、表やクラスタ内の移行された行と連鎖行の情報を収集できます。移行された行とは、あるデータ・ブロックから別のデータ・ブロックに移された行です。たとえば、クラスタ・キーの値が更新されるとそのクラスタ中の行は移動されます。連鎖行とは、複数のデータ・ブロックに含まれている行です。たとえば、表またはクラスタの行が長すぎて1つのデータ・ブロックに収まらない場合、この行は自動的に連鎖されます。移行された行と連鎖行では、通常より多くの I/O が必要になることがあります。このような行は識別して削除できます。移行された行および連鎖行の削除については、『Oracle8 Server チューニング』を参照してください。

**INTO** 句を使用して、統計情報を格納する出力表を指定できます。サンプル出力表 **CHAINED\_ROWS** の定義は、配布メディアの **SQL** スクリプトで提供されます。リスト表はこの **CHAINED\_ROWS** 表の列名および型、サイズと同じでなければなりません。通常、このスクリプト名は **UTLCHAIN.SQL** です。このスクリプトの実際の名前と格納場所はオペレーティング・システムによって異なります。

**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
```

## 関連項目

『Oracle8 Server チューニング』

# ARCHIVE LOG 句

## 用途

REDO ログ・ファイル・グループを手動でアーカイブするか、または自動アーカイブを使用可能または使用禁止にします。「使用上の注意」(4-166 ページ) も参照してください。

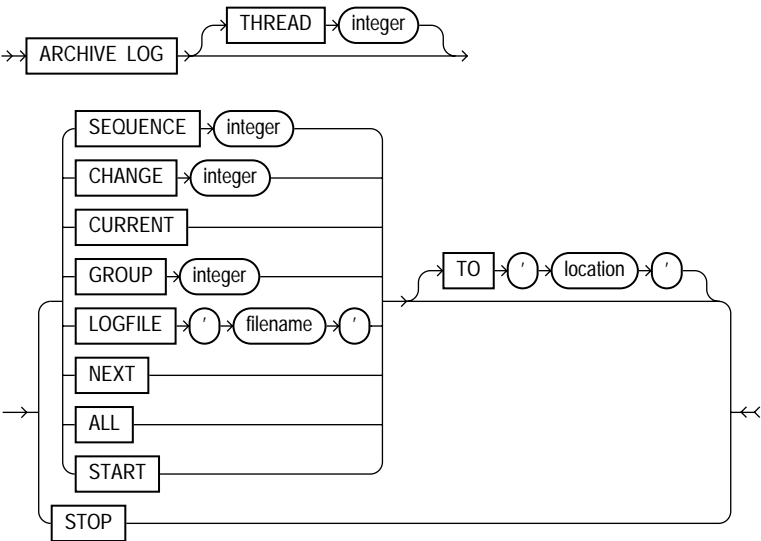
## 前提条件

ARCHIVE LOG 句は ALTER SYSTEM コマンドに指定されなければなりません。したがって、この文の実行権限が必要です。これらの権限の詳細は、ALTER SYSTEM (4-88 ページ) を参照してください。

OSDBA ロールまたは OSOPER ロールも使用可能でなければなりません。

自インスタンスでデータベースがマウントまたはオープン、クローズされているときには、この句のほとんどのオプションが使用できます。インスタンスでデータベースをオープンしておく必要があるオプションについては、別途明記します。

## 構文



## キーワードとパラメータ

THREAD	アーカイブする REDO ログ・ファイル・グループを含むスレッドです。Parallel Server オプション付き Oracle をパラレル・モードで使用している場合にだけ、指定してください。
SEQUENCE	指定したスレッド内のログ順序番号 <i>integer</i> によって識別されるオンライン REDO ログ・ファイル・グループを手動でアーカイブします。THREAD パラメータを指定しないと、自インスタンスに割り当てられているスレッドから、指定したグループがアーカイブされます。
CHANGE	指定したスレッド内の <i>integer</i> によって指定されたシステム変更番号 (SCN) を持つ REDO ログ・エントリを含むオンライン REDO ログ・ファイル・グループを手動でアーカイブします。この SCN が現行の REDO ログ・ファイル・グループ内にある場合、ログ・スイッチが実行されます。THREAD パラメータを指定しないと、使用可能な状態にあるすべてのスレッドからこの SCN を含むグループがアーカイブされます。自インスタンスでデータベースをオープンしている場合にだけ、このオプションを使用できます。
CURRENT	ログ・スイッチを強制的に発生させ、指定したスレッドの現行の REDO ログ・ファイル・グループを手動でアーカイブします。THREAD パラメータを指定しないと、すべての使用可能なスレッドから、現行のログ以前のログも含むすべての REDO ログ・ファイル・グループがアーカイブされます。自インスタンスでデータベースをオープンしている場合にだけ、このオプションを使用できます。
GROUP	<i>integer</i> で指定した GROUP 値を持つオンライン REDO ログ・ファイル・グループを手動でアーカイブします。この REDO ログ・ファイル・グループの GROUP 値は、データ・ディクショナリ・ビュー DBA_LOG_FILES を調べれば確認できます。THREAD パラメータと GROUP パラメータの両方を指定する場合は、指定する REDO ログ・ファイル・グループは指定するスレッド内に含まれていなければなりません。
LOGFILE	'filename' によって識別される、REDO ログ・ファイル・メンバーを含むオンライン REDO ログ・ファイル・グループを手動でアーカイブします。THREAD パラメータと LOGFILE パラメータの両方を指定する場合は、指定する REDO ログ・ファイル・グループが指定するスレッド内に含まれていなければなりません。
NEXT	満杯になったがまだアーカイブされていない次のオンライン REDO ログ・ファイルを、指定したスレッドから手動でアーカイブします。THREAD パラメータを指定しないと、使用可能な任意のスレッド上の、アーカイブされていない最初の REDO ログ・ファイル・グループがアーカイブされます。
ALL	満杯になったがまだアーカイブされていないすべてのオンライン REDO ログ・ファイルを、指定したスレッドから手動でアーカイブします。THREAD パラメータを指定しないと、使用可能な状態にあるすべてのスレッドから、満杯でアーカイブされていないすべての REDO ログ・ファイル・グループがアーカイブされます。
START	REDO ログ・ファイル・グループの自動アーカイブを使用可能にします。自インスタンスに割り当てられているスレッドについてだけ、自動アーカイブを使用可能にできます。
TO	アーカイブされる REDO ログ・ファイル・グループの格納場所を指定します。このパラメータの値は、オペレーティング・システムの規則に従って、ファイルの位置を完全に指定しなければなりません。このパラメータを指定しないと、REDO ログ・ファイル・グループは初期化パラメータ LOG_ARCHIVE_DEST に指定されている場所に保管されます。

---

STOP	REDO ログ・ファイル・グループの自動アーカイブを使用禁止にします。自インスタンスに割り当てられているスレッドについてだけ、自動アーカイブを使用禁止にできます。
------	---

---

## 使用上の注意

REDO ログ・ファイル・グループは、満杯になった順にアーカイブしなければなりません。LOGFILE パラメータを使用して REDO ログ・ファイル・グループのアーカイブを指定した場合、それ以前の REDO ログ・ファイル・グループがアーカイブされていないとエラー・メッセージが戻ります。CHANGE パラメータまたは CURRENT オプションを使用して REDO ログ・ファイル・グループのアーカイブを指定した場合、それ以前の REDO ログ・ファイル・グループがアーカイブされていないと、指定したグループとアーカイブされていないグループがすべてアーカイブされます。

ARCHIVE LOG という Server Manager のコマンドを使用すると、REDO ログ・ファイル・グループを手動でもアーカイブできます。このコマンドについては、『Oracle Server Manager ユーザーズ・ガイド』を参照してください。

REDO ログ・ファイル・グループの自動アーカイブも選択できます。自動アーカイブについては、『Oracle8 Server 管理者ガイド』を参照してください。なお、自動アーカイブが使用可能な状態にあるかどうかにかかわらず、常に REDO ログ・ファイル・グループを手動でアーカイブできます。

**例 1.** 次の文は、スレッド番号 3、ログ順序番号 4 の REDO ログ・ファイル・グループを手動でアーカイブします。

```
ALTER SYSTEM ARCHIVE LOG THREAD 3 SEQUENCE 4;
```

**例 2.** 次の文は、SCN 9356083 の REDO ログ・エントリを含む REDO ログ・ファイル・グループを手動でアーカイブします。

```
ALTER SYSTEM ARCHIVE LOG CHANGE 9356083;
```

**例 3.** 次の文は、メンバー 'DISKL:LOG6.LOG' を含む REDO ログ・ファイル・グループを、'DISKA:[ARCH\$]' という場所にあるアーカイブ REDO ログ・ファイルにアーカイブします。

```
ALTER SYSTEM ARCHIVE LOG
  LOGFILE 'disk1:log6.log'
  TO 'diska:[arch$]';
```

## 関連項目

ALTER SYSTEM (4-88 ページ)

## AUDIT(SQL 文)

### 用途

後続するユーザー・セッションを監査する SQL 文を選択します。監査の対象となる特定のスキーマ・オブジェクトを選択するには、AUDIT(スキーマ・オブジェクト) (4-175 ページ) を指定します。「監査」 (4-168 ページ) も参照してください。

---

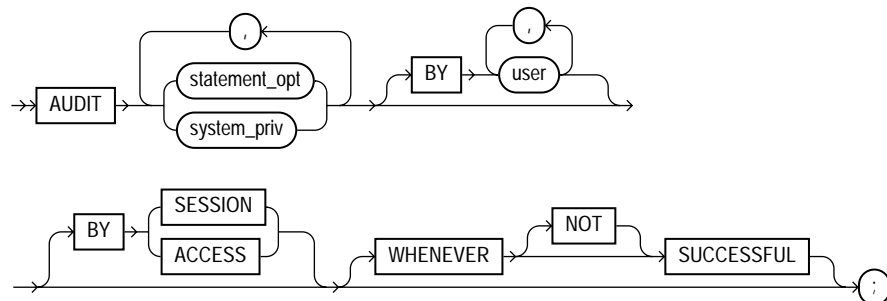
**注意：** コマンドと句の説明で **OBJ** の印が前に付いている箇所は、Oracle Object Option がデータベース・サーバーにインストールされている場合にだけ有効です。

---

### 前提条件

AUDIT SYSTEM システム権限が必要です。

### 構文



### キーワードとパラメータ

<i>statement_opt</i>	監査の対象となる特定の SQL 文を選択します。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、表 4-6 および表 4-7 を参照してください。「データベース・オブジェクトの文オプション」 (4-169 ページ) および「コマンドの文オプション」 (4-171 ページ) を参照してください。
<i>system_priv</i>	監査の対象となる、指定したシステム権限によって許可される SQL 文を選択します。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 4-19 を参照してください。「システム権限と文オプションのショートカット」 (4-173 ページ) を参照してください。

BY <i>user</i>	指定したユーザーによって発行された SQL 文だけを監査の対象として選択します。この句を指定しないと、すべてのユーザー文が監査されます。
BY SESSION	同一セッションで発行された同じ種類の SQL 文すべてについて、1 つのレコードが書き込まれます。
BY ACCESS	1 つの文が監査されるたびに、レコードが 1 件書き込まれます。  DDL 文を監査する文のオプションまたはシステム権限を指定すると、BY SESSION オプションと BY ACCESS オプションのどちらを指定しても、アクセスすることによって自動的に監査が行われます。  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 オプションと BY ACCESS オプションのどちらでも指定できます。デフォルトは BY SESSION です。
WHENEVER SUCCESSFUL	SQL 文が成功した場合にだけ監査が行われます。  NOT を指定すると、SQL 文が失敗またはエラーに終わった場合にだけ監査が行われます。  WHENEVER 句を指定しないと、SQL 文が成功しても失敗しても、監査が行われます。

監査

監査では、データベース・ユーザーが実行する操作を追跡し記録します。監査されるたびに、次の情報を持つ監査レコードが生成されます。

- 操作を行ったユーザー
- 操作の種類
- 操作に関連したオブジェクト
- 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。これらのビューの詳細は、『Oracle8 Server リファレンス・マニュアル』を参照してください。

監査レコードを作成するには次の手順に従います。

監査を使用可能にする

初期化パラメータ AUDIT\_TRAIL = DB を設定することによって、監査を使用可能にしてください。

監査オプションを指定する

監査オプションを指定するには、AUDIT コマンドを使う必要があります。監査オプションによって、監査対象の SQL コマンドおよび操作、データベース・オブジェクト、ユーザーを選択します。監査オプションを選択すると、それらのオプションがデータ・ディクシヨナ

りに表示されます。監査オプションが表示されるデータ・ディクショナリ・ビューの説明は、『Oracle8 Server リファレンス・マニュアル』を参照してください。

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いと監査レコードは作成されません。

AUDIT コマンド (SQL 文) を使用して指定した監査オプションは、現行セッションには適用されず、後続のセッションだけに適用されます。

データベース・オブジェクトの文オプション

表 4-6 に、データベース・オブジェクトに関連する文オプションと監査される文を示します。

表 4-6 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
DATABASE LINK	CREATE DATABASE LINK DROP DATABASE LINK
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ていない場合に失敗するすべての SQL 文。
PROCEDURE	CREATE FUNCTION CREATE LIBRARY CREATE PACKAGE CREATE PACKAGE BODY CREATE PROCEDURE DROP FUNCTION DROP LIBRARY DROP PACKAGE DROP PROCEDURE
PROFILE	CREATE PROFILE ALTER PROFILE DROP PROFILE

表 4-6 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
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	ログイン
SYNONYM	CREATE SYNONYM DROP SYNONYM
SYSTEM AUDIT	AUDIT(SQL 文) NOAUDIT(SQL 文)
SYSTEM GRANT	GRANT( システム権限とロール ) REVOKE ( システム権限とロール )
TABLE	CREATE TABLE DROP TABLE TRUNCATE TABLE
TABLESPACE	CREATE TABLESPACE ALTER TABLESPACE DROP TABLESPACE
TRIGGER	CREATE TRIGGER ALTER TRIGGER ( ENABLE オプションおよび DISABLE オプションを指定 ) DROP TRIGGER ALTER TABLE ( ENABLE ALL TRIGGERS 句 および DISABLE ALL TRIGGERS 句を指定 )



表 4-6 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
<b>OBJ</b> 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

## コマンドの文オプション

表 4-7 に、コマンドに関連するその他の文オプションおよび監査される SQL 文と操作を示します。

表 4-7 コマンド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 <i>table, view, snapshot</i> COMMENT ON COLUMN <i>table.column, view.column, snapshot.column</i>
DELETE TABLE	DELETE FROM <i>table, view</i>
EXECUTE PROCEDURE	プロシージャまたは関数の実行。あるいは、パッケージ内の変数またはライブラリ、カーソルへのアクセス。
GRANT DIRECTORY	GRANT <i>privilege</i> ON <i>directory</i> REVOKE <i>privilege</i> ON <i>directory</i>
GRANT PROCEDURE	GRANT <i>privilege</i> ON <i>procedure, function, package</i> REVOKE <i>privilege</i> ON <i>procedure, function, package</i>
GRANT SEQUENCE	GRANT <i>privilege</i> ON <i>sequence</i> REVOKE <i>privilege</i> ON <i>sequence</i>
GRANT TABLE	GRANT <i>privilege</i> ON <i>table, view, snapshot</i> . REVOKE <i>privilege</i> ON <i>table, view, snapshot</i>
<b>OBJ</b> GRANT TYPE	GRANT <i>privilege</i> ON TYPE REVOKE <i>privilege</i> ON TYPE

表 4-7 コマンドの文監査オプション

文オプション	SQL 文と操作
INSERT TABLE	INSERT INTO <i>table</i> , <i>view</i>
LOCK TABLE	LOCK TABLE <i>table</i> , <i>view</i>
SELECT SEQUENCE	<i>sequence</i> .CURRVAL または <i>sequence</i> .NEXTVAL を含む文
SELECT TABLE	SELECT FROM <i>table</i> , <i>view</i> , <i>snapshot</i>
UPDATE TABLE	UPDATE <i>table</i> , <i>view</i>

例 1. 次の文は、ロールの作成または変更、削除、設定を行う各 SQL 文が正常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、ロールの作成または変更、削除、設定を行う、正常終了した各 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 文または ALTER ROLE 文、DROP ROLE 文、SET ROLE 文について監査を行います。

```
AUDIT ROLE
  WHENEVER NOT SUCCESSFUL;
```

例 2. 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;
```

例 3. 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;
```

例 4. 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;
```

例 5. CREATE ANY DIRECTORY システム権限を使用しない CREATE DIRECTORY( および DROP DIRECTORY) 文を監査するには次の文を発行します。

```
AUDIT DIRECTORY;
```

## システム権限と文オプションのショートカット

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ショートカットが用意されています。ただし、Oracle の将来のバージョンではショートカットがサポートされない可能性があるので、監査のためのシステム権限と文オプションは個別に選択することをお勧めします。ショートカットには次のものがあります。

CONNECT	CREATE SESSION システム権限を指定するのと同じです。
RESOURCE	次のシステム権限を指定することと同じです。 <ul style="list-style-type: none"><li>• ALTER SESSION</li><li>• CREATE CLUSTER</li><li>• CREATE DATABASE LINK</li><li>• CREATE PROCEDURE</li><li>• CREATE ROLLBACK SEGMENT</li><li>• CREATE SEQUENCE</li><li>• CREATE SYNONYM</li><li>• CREATE TABLE</li><li>• CREATE TABLESPACE</li><li>• CREATE VIEW</li></ul>
DBA	SYSTEM GRANT 文オプションと次のシステム権限を指定することと同じです。 <ul style="list-style-type: none"><li>• AUDIT SYSTEM</li><li>• CREATE PUBLIC DATABASE LINK</li><li>• CREATE PUBLIC SYNONYM</li><li>• CREATE ROLE</li><li>• CREATE USER</li></ul>
ALL	表 4-7 の文オプションを除き、表 4-6 のすべての文オプションを指定することと同じです。
ALL PRIVILEGES	すべてのシステム権限を指定するのと同じです。

## 関連項目

[AUDIT\( スキーマ・オブジェクト \) \(4-175 ページ\)](#)

[NOAUDIT\( スキーマ・オブジェクト \) \(4-460 ページ\)](#)

---

## AUDIT( スキーマ・オブジェクト )

### 用途

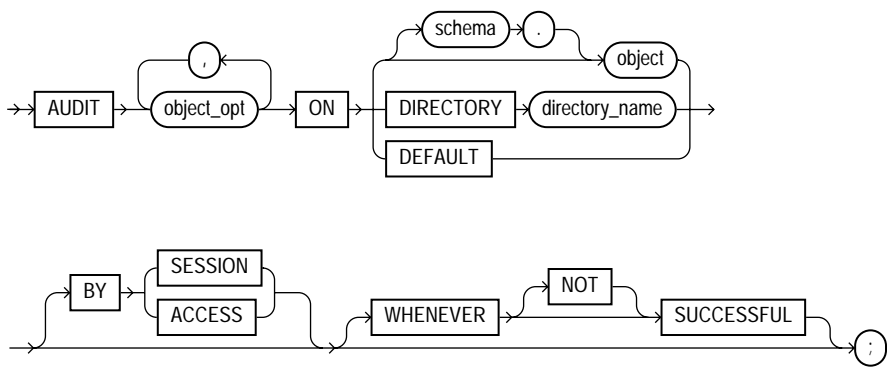
監査の対象とするスキーマ・オブジェクトを選択します。監査の対象として SQL コマンドを選択する場合は、AUDIT(SQL 文) (4-167 ページ) を参照してください。

監査では、データベース・ユーザーが実行する操作を追跡し記録します。監査を使用可能にする方法など監査の概要は、AUDIT(SQL 文) (4-167 ページ) を参照してください。なお、AUDIT コマンド (スキーマ・オブジェクト) を使用して設定した監査オプションは、後続のセッションだけでなく、現行セッションにも適用されるので注意してください。

### 前提条件

監査の対象とするオブジェクトが自スキーマ内にあることが必要です。自スキーマ内がない場合は、AUDIT ANY システム権限が必要です。また、監査の対象とな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。

構文



キーワードとパラメータ

<i>object_opt</i>	監査の対象とする操作を指定します。表 4-8 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。「オブジェクト・オプション」(4-177 ページ) も参照してください。
<i>schema</i>	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です。schema を指定しないと、このオブジェクトが自スキーマ内にあるとみなされます。
<i>object</i>	監査の対象として選択されたオブジェクトです。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、スナップショット、ライブラリのいずれかでなければなりません。  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、スナップショットについてはそれぞれシノニムも指定できます。
ON DEFAULT	指定したオブジェクト・オプションを、今後作成されるオブジェクトに対するデフォルトのオブジェクト・オプションとして設定します。「デフォルト監査」(4-178 ページ) も参照してください。
DIRECTORY <i>directory_name</i>	監査の対象とするディレクトリの名前を指定します。
BY SESSION	同一セッションで発行された同一オブジェクトに対する同じ種類の操作すべてについて、1 つのレコードが書き込まれます。
BY ACCESS	1 つの文が監査されるたびにレコードが 1 件書き込まれます。
前述のオプションのいずれも指定しないと、セッションごとに監査が行われます。	
WHENEVER SUCCESSFUL	正常に完了した SQL 文だけを対象とした監査を指定します。  NOT は SQL 文が失敗またはエラーになった場合にだけ監査します。

WHENEVER 句を指定しないと、SQL 文が成功しても失敗しても、監査がすべて行われます。

オブジェクト・オプション

表 4-8 に、各オブジェクトに対して選択できるオブジェクト・オプションを示します。

表 4-8 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、 関数、パッ ケージ	スナップ ショット	ライブラリ	ディレクトリ
ALTER	X		X		X		
AUDIT	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
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		

各オブジェクト・オプションには、監査の対象となるコマンドを指定します。たとえば、ALTER オプションを指定して表の監査を選択すると、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すると、その順序の値を使用する文がすべて監査されます。

オブジェクト・オプションのショートカット

Oracle には、オブジェクト監査オプションを指定するためのショートカットが用意されています。

ALL                      その種類のオブジェクトに適用可能なオプションをすべて指定した場合と同じです。オブジェクトの全オプションを明示的に指定するかわりに、このオプションを使用できます。

## デフォルト監査

AUDIT コマンドの DEFAULT オプションを使用して、まだ作成されていないオブジェクトに対する監査オプションを指定できます。デフォルト監査オプションを指定すると、これから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て監査が行われます。ビューに対するデフォルト監査オプションは、常にそのビューの実表に対する監査オプションの論理和となるので注意してください。

デフォルト監査オプションを変更しても、前回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 コマンドの ON 句にオブジェクトを指定した場合だけ、既存のオブジェクトの監査オプションを変更できます。

**例 1.** 次の文は、スキーマ SCOTT 内の EMP 表を問い合わせる各 SQL 文に対する監査を行います。

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

次の文は、スキーマ SCOTT 内の EMP 表を問い合わせて正常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 表を問い合わ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;
```

**例 2.** 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各文について監査を行います。

```
AUDIT INSERT, UPDATE
  ON blake.dept;
```

**例 3.** 次の文は、スキーマ ADAMS 内の ORDER 順序に対する操作を行う各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に、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

**例 4.** 次の文では、BFILE\_DIR1 ディレクトリからファイルを読み取る各文について監査を行います。

```
AUDIT READ ON DIRECTORY bfile_dir1;
```

**例 5.** 次の文は、今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;
```

今後作成されるオブジェクトは、監査機能が使用可能であれば、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。

- 表を作成すると、その表に対して発行される ALTER 文、INSERT 文、UPDATE 文、DELETE 文が自動的に監査される。
- ビューを作成すると、そのビューに対して発行される INSERT 文、UPDATE 文、DELETE 文が自動的に監査される。
- 順序を作成すると、その順序に対して発行される ALTER 文が自動的に監査される。
- プロシージャまたはパッケージ、関数を作成すると、それらに対して発行される ALTER 文が自動的に監査される。

## 関連項目

AUDIT(SQL 文) (4-167 ページ)

NOAUDIT( スキーマ・オブジェクト ) (4-460 ページ)

# COMMENT

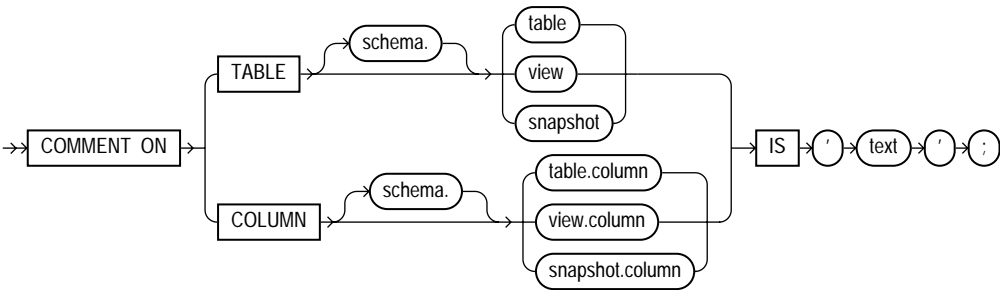
## 用途

表、ビュー、スナップショット、列などについてのコメントをデータ・ディクショナリに追加します。「使用上の注意」(4-180 ページ)を参照してください。

## 前提条件

表またはビュー、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。自スキーマ内がない場合は、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	スキーマと、コメントする表またはビュー、スナップショットの名前です。
COLUMN	コメントする表またはビュー、スナップショットの列の名前を指定します。 <i>schema</i> を指定しないと、Oracle は、この表およびビュー、スナップショットが自スキーマ内にあるとみなします。
IS 'text'	コメントのテキストです。'text' の構文の説明は、「Text( テキスト )」(2-2 ページ)を参照してください。

## 使用上の注意

空の文字列''を設定することによって、データベースから効果的にコメントを削除できます。コメントを含むデータ・ディクショナリ・ビューについては、『Oracle8 Server リファレンス・マニュアル』を参照してください。

例 . 次の文は、SHIPPING 表の NOTES 列の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 ' ';
```

## 関連項目

「コメント」(2-35 ページ)

# COMMIT

## 用途

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します。また、このコマンドによって、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すべてのロックが解除されます。「使用上の注意」(4-183 ページ) も参照してください。

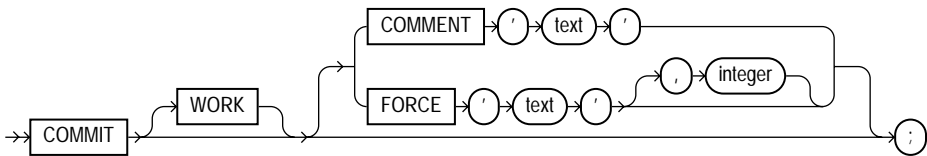
また、このコマンドを使用すると、インダウト分散トランザクションを手動でコミットできます。トランザクションの詳細は、「トランザクションの終わり」(4-184 ページ) を参照してください。

## 前提条件

現行トランザクションをコミットするために、特に必要な権限はありません。

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には、**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には、**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 に準拠するためにサポートされています。COMMIT 文と COMMIT WORK 文は同じです。
COMMENT	現行トランザクションに関するコメントです。'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、つまり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。

---

**FORCE**      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。このトランザクションはそのローカル・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 を確認するには、データ・ディクショナリ・ビュー DBA\_2PC\_PENDING を問い合わせます。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) を明示的に割り当てることもできます。integer を指定しないと、このトランザクションは現行の SCN を使用してコミットされます。

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

---

## 使用上の注意

トランザクション（論理的作業単位）とは、Oracle が 1 つの単位として扱う一連の SQL 文です。トランザクションは、COMMIT 文または ROLLBACK 文の後、またはデータベースに接続した後の最初の実行可能 SQL 文によって開始されます。トランザクションは、COMMIT 文または ROLLBACK 文、データベースとの接続の切離し（意図されたかどうかにかかわらず）によって終了します。なお、データ定義言語 (DDL) 文の前後に暗黙的な COMMIT が発行されます。

COMMIT 文または ROLLBACK 文を使用すると、SET TRANSACTION 文によって開始された読取り専用トランザクションも終了できます。

**例 1.** 次の文は、DEPT 表に行を挿入してこの変更をコミットします。

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

**例 2.** 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 WORK  
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この分散トランザクションが正常にコミットされない場合、コメントがトランザクション ID とともにデータ・ディクショナリに格納されます。このコメントは、アプリケーションの障害発生部分と、コミットされたトランザクションのデータベース管理者の連絡先を表示します。

## 分散トランザクション

分散オプション付きの Oracle では、分散トランザクション（つまり複数データベース内のデータを変更するトランザクション）を実行できます。分散トランザクションをコミットするには、他のトランザクションのコミットと同じように COMMIT 文を発行します。分散トランザクションの各コンポーネントは各データベース上でコミットされます。

分散トランザクションのコミット・プロセス中にネットワーク障害またはマシン障害が発生すると、トランザクションの状態が不明、つまりインダウトな場合があります。この場合、そのトランザクションにかかわる他のデータベースの管理者と相談し、ローカル・データベース上のトランザクションを手動でロールバックするか、コミットするかを判断してください。

さい。COMMIT コマンドの FORCE 句を使用することによって、ローカル・データベースのトランザクションを手動でコミットできます。詳細は、『Oracle8 Server 分散システム』を参照してください。

FORCE 句を指定して COMMIT 文を発行すると、指定したトランザクションだけがコミットされるので注意してください。この文は現行トランザクションには影響しません。

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';
```

## トランザクションの終わり

Oracle との接続を切り離す前に、最新のトランザクションを含む、アプリケーション・プログラムにおけるすべてのトランザクションを、COMMIT 文または ROLLBACK 文を使っ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 ユーティリティや Oracle Tools を正常に終了すると、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・プログラムを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。

## 関連項目

COMMENT (4-180 ページ)

COMMENT (4-180 ページ)

SET TRANSACTION (4-514 ページ)

## CONSTRAINT 句

---

### 用途

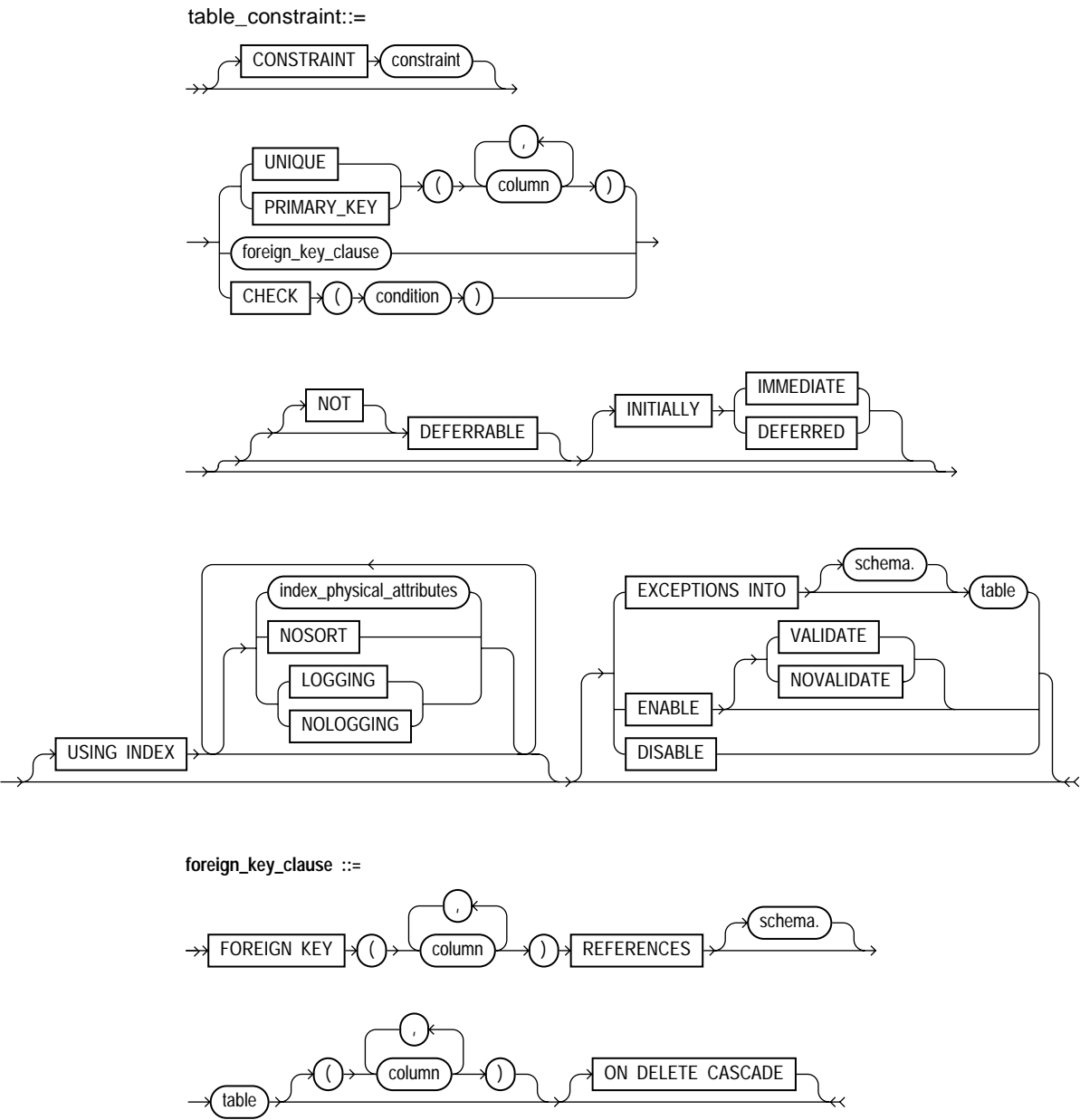
整合性制約を定義します。整合性制約とは、表または索引構成表の 1 つ以上の列に対する値を制限する規則です。

### 前提条件

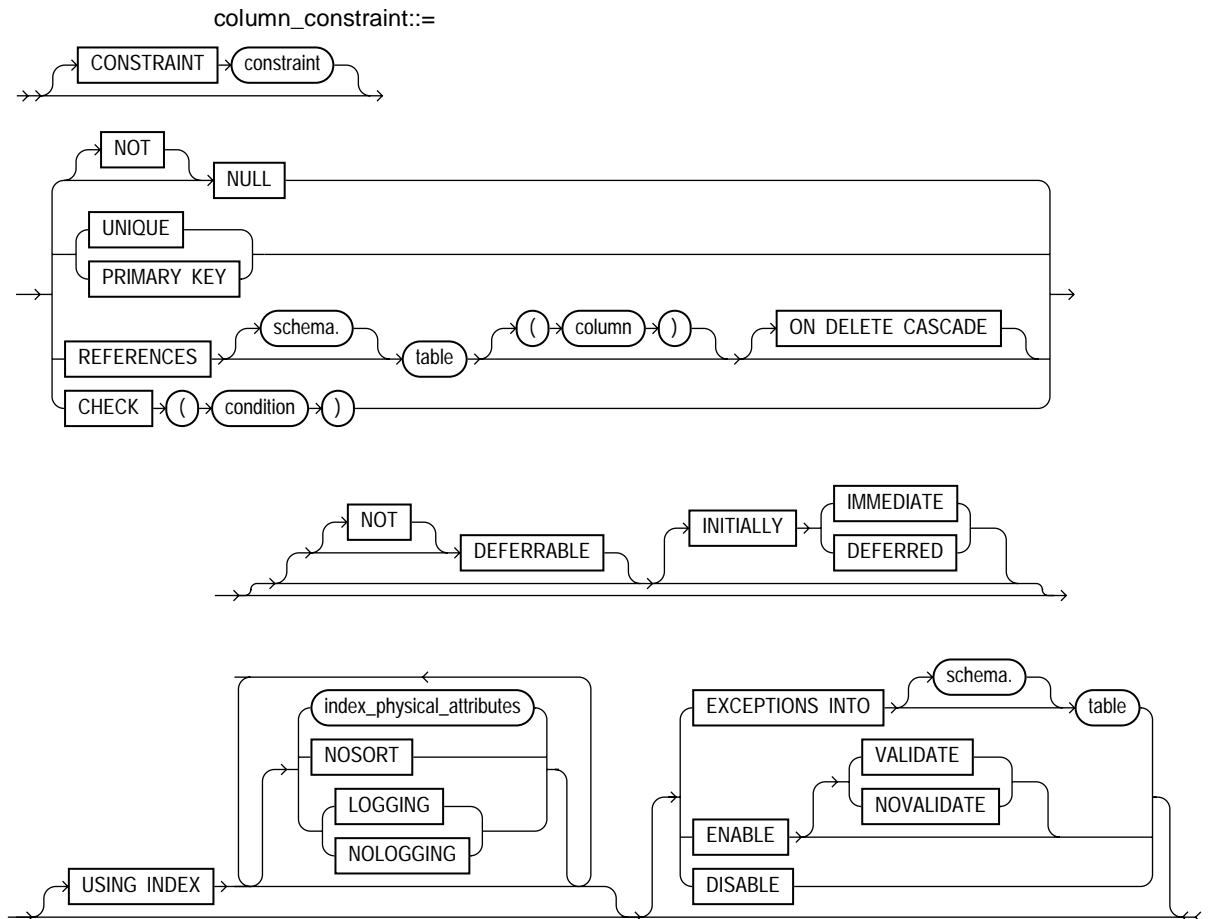
CONSTRAINT 句は、CREATE TABLE コマンドまたは ALTER TABLE コマンドのいずれかで指定されます。整合性制約を定義するには、これらのコマンドのどちらかを発行するために必要な権限を持っていなければなりません。CREATE TABLE (4-303 ページ) および ALTER TABLE (4-105 ページ) を参照してください。

また制約の定義は、制約の種類によって補足的な権限または前提条件を必要とします。これらの権限については、「整合性制約の定義」(4-189 ページ) の各種整合性制約の説明を参照してください。

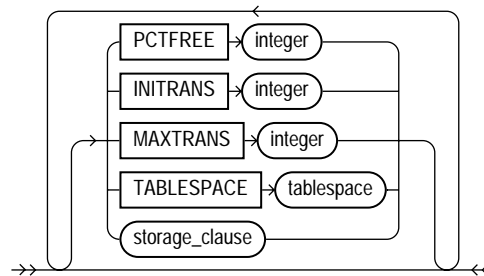
構文







**index\_physical\_attributes ::=**



storage\_clause: STORAGE 句 (4-518 ページ) を参照してください。

キーワードとパラメータ

CONSTRAINT	整合性制約の名前 <i>constraint</i> 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と、SYS_Cn の形で名前が生成されます。「整合性制約の定義」(4-189 ページ) も参照してください。  列定義において NULL も NOT NULL も指定しないと、デフォルト値は NULL になります。
UNIQUE	列または列の組合せを一意キーとして指定します。索引構成表には UNIQUE 制約は定義できません。「UNIQUE 制約」(4-190 ページ) も参照してください。
PRIMARY KEY	列または列の組合せを表の主キーとして指定します。「PRIMARY KEY 制約」(4-192 ページ) も参照してください。
FOREIGN KEY	列または列の組合せを参照整合性制約の外部キーとして指定します。
REFERENCES	参照整合性制約の外部キーによって参照される主キーまたは一意キーを識別します。「参照整合性制約」(4-193 ページ) も参照してください。
ON DELETE CASCADE	参照主キーまたは参照一意キーの値を削除する場合に、Oracle は対応する外部キーの値を自動的に削除することによって参照整合性をメンテナンスします。
NULL	列に NULL 値を入れることができます。
NOT NULL	列に NULL 値を入れることはできません。「NOT NULL 制約」(4-190 ページ) も参照してください。
CHECK	表の各行が満たさなければならない条件を指定します。「CHECK 制約」(4-197 ページ) も参照してください。
DEFERRABLE	SET CONSTRAINT(S) コマンドを使って、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。
NOT DEFERRABLE	この制約が各 DML 文の終わりでチェックされます。SET CONSTRAINT(S) コマンドでも、NOT DEFERRABLE 制約は延期できません。DEFERRABLE と NOT DEFERRABLE のどちらも指定しない場合は、NOT DEFERRABLE がデフォルト値です。「DEFERRABLE 制約」(4-200 ページ) も参照してください。
INITIALLY IMMEDIATE	各トランザクションの開始時に、デフォルトで、各 DML 文の終わりにこの制約がチェックされるようになります。INITIALLY 句が指定されていない場合は、INITIALLY IMMEDIATE がデフォルト値です。
INITIALLY DEFERRED	この制約について DEFERRABLE が暗黙に指定されます。すなわち、デフォルトでは各トランザクションの終了時にだけ制約がチェックされます。
USING INDEX	UNIQUE 制約または PRIMARY KEY 制約を使用可能にするために使用される索引のパラメータを指定します。索引名は制約名と同じです。索引には INITRANS、MAXTRANS、TABLESPACE、STORAGE、PCTFREE、LOGGING、NOLOGGING の各パラメータ値を指定できます。これらのパラメータについては、CREATE TABLE (4-303 ページ) を参照してください。  UNIQUE 制約と PRIMARY KEY 制約を使用可能にする場合は、この句だけを使用します。

NOSORT	データベースに行が昇順で格納されているので、索引を作成するときに行をソートする必要がないことを指定します。
EXCEPTIONS INTO	<p>制約に違反するすべての列の ROWID が入る表を指定します。</p> <p><b>注意：</b>制約を使用可能にする前に、ENABLE 句の EXCEPTIONS オプションからの情報を受け入れるための、適切な例外レポート表を作成する必要があります。EXCEPTIONS という名前の表を作成するスクリプト UTLEXCPT.SQL を発行することにより、例外表を作成できます。スクリプトを変更して再発行すると、別の名前の例外表を作成できます。</p> <p>EXCEPTIONS INTO 句は、制約の妥当性検査を行う場合（ENABLE 句（4-414 ページ）を参照）または ALTER TABLE コマンドを使用して制約を使用可能にする場合にだけ有効なオプションです。ALTER TABLE（4-105 ページ）を参照してください。</p>
ENABLE VALIDATE	制約データに対して新たに行うすべての挿入、削除、更新の操作が制約に従っているか確認します。また、すべての旧データが制約に従っているかもチェックします。制約を使用可能で妥当性検査される (ENABLE VALIDATE) 状態にすると、すべてのデータが現在有効であり、今後でも有効であることが保証されます。これはデフォルト値です。
ENABLE NOVALIDATE	制約データに対して新たに行うすべての挿入、更新、削除の操作が制約に従っているか確認します。すでに表にあるデータが制約に従っているかどうかは検証しません。
DISABLE	整合性制約を使用禁止にします。整合性制約が使用禁止 (DISABLED) に設定されている場合は、整合性制約は有効ではありません。このオプションを指定しないと、Oracle は自動的に整合性制約を使用可能にします。

また、CREATE TABLE コマンドと ALTER TABLE コマンドに ENABLE 句や DISABLE 句を指定すると、整合性制約を使用可能にしたり、使用禁止にしたりできます。ENABLE 句（4-414 ページ）および DISABLE 句（4-375 ページ）を参照してください。「制約の使用可能と使用禁止の切替え」（4-200 ページ）も参照してください。

使用禁止の制約は、ALTER TABLE（4-105 ページ）を使って使用可能にすることができます。

## 整合性制約の定義

整合性制約を定義するには、CREATE TABLE 文または ALTER TABLE 文に CONSTRAINT 句を指定します。CONSTRAINT 句には次の 2 種類の構文書式があります。

<i>table_constraint</i>	<p><i>table_constraint</i> 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。</p> <p><i>table_constraint</i> 構文は、CREATE TABLE 文または ALTER TABLE 文に指定でき、NOT NULL 制約以外の整合性制約を定義できます。</p>
<i>column_constraint</i>	<p><i>column_constraint</i> 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だけ規則を適用します。</p> <p>CREATE TABLE 文に指定する <i>column_constraint</i> 構文は、整合性制約をどれでも定義できます。ALTER TABLE 文に指定する <i>column_constraint</i> 構文は、NOT NULL 制約の定義または削除だけができます。</p>

*table\_constraint* 構文と *column\_constraint* 構文は、整合性制約定義の方法に構文上の違いがあるだけです。ただし、2 つ以上の列を参照する制約は表制約として定義する必要があります。*table\_constraint* 構文を使用して定義した整合性制約と *column\_constraint* 構文を使用して定義した整合性制約には、機能的な違いはまったくありません。

---

---

**注意：** ユーザー定義型または LOB 型、REF 型の列または属性に対しては、制約を作成できません。唯一の例外として、OBJECT 型または VARRAY 型、LOB、REF の列または属性に対する NOT NULL 制約の作成はサポートされます。

---

---

## NOT NULL 制約

NOT NULL 制約は、列が NULL を持つことができないことを指定します。この制約を満たすには、表の中のあらゆる行がその列に対して値を持っていないければなりません。

NULL キーワードは、列が NULL を持つことができることを示します。ただし、この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と、デフォルトでは列は NULL を持つことができます。

CREATE TABLE 文または ALTER TABLE 文の *column\_constraint* 構文では NOT NULL 句や NULL 句を指定できますが、*table\_constraint* 構文では指定できません。

**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 を指定すると、表の中の従業員の給与が NULL 値になることはありません。

## UNIQUE 制約

UNIQUE 制約は、列または列の組合せを一意キーとして指定します。UNIQUE 制約を満たすには、表の中の 2 つの行が一意キーに対して同じ値を持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。

一意キーの列には、データ型 LONG や LONG RAW を設定できません。同一の列または列の組合せを一意キーと主キーの両方、または一意キーとクラスターキーの両方には指定できません。ただし、同一の列または列の組合せを一意キーと外部キーの両方には指定できます。

### 一意キーの定義

*column\_constraint* 構文を使用して、単一の列に対して一意キーを定義できます。

**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構いません。

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) );
```

この文では、*USING INDEX* オプションを使用することで、制約を使用可能にするために Oracle が作成する索引の記憶特性も指定しています。

## 複合一意キーの定義

複合一意キーは、列の組合せからなる一意キーです。Oracle は一意キーとなる列に対して索引を作成するため、複合一意キーは最大 16 列で構成されます。複合一意キーを定義するには、*column\_constraint* 構文ではなく *table\_constraint* 構文を使用してください。

複合一意キーを指定する制約を満たすには、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。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。しかし、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、制約に反します。

**例** . 次の文は、CENSUS 表の CITY 列と STATE 列を組み合わせで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 の値と STATE の値として同一の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 CONSTRAINT 句には制約以外の特性も指定できます。

- USING INDEX オプションは、制約を使用可能にするために作成する索引の記憶特性を指定する。
- EXCEPTIONS オプションを指定すると、制約に反している CENSUS 表の行に関する情報は BAD\_KEYS\_IN\_SHIP\_CONT 表に書き込まれる。

## PRIMARY KEY 制約

PRIMARY KEY 制約は、列またはその組合せを表の主キーとして指定します。PRIMARY KEY 制約を満たすには、次の条件が両方とも真でなければなりません。

- 主キーの値が表の中の複数行に存在することがない。
- 主キーを構成する列が NULL を持たない。

表には主キーを 1 つだけ指定できます。

主キーの列は、データ型 LONG や LONG RAW にできません。同一の列または列の組合せは、主キーと一意キーの両方、または主キーとクラスタ・キーの両方としては、指定できません。ただし、主キーと外部キーの両方として、同一の列または列の組合せを指定できます。

## 主キーの定義

*column\_constraint* 構文を使用して、単一の列に対して主キーを定義できます。

**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。この制約により、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* 構文を使っても、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) );
```

## 複合主キーの定義

複合主キーは、列の組合せからなる主キーです。Oracle は主キーとなる列に対して索引を作成するため、複合主キーは最大 16 列で構成されます。複合主キーを定義するには、*column\_constraint* 構文ではなく *table\_constraint* 構文を使用してください。

例 . 次の文は、SHIP\_CONT 表の SHIP\_NO 列と CONTAINER\_NO 列を組み合わせで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、次の制約特性も指定しています。

- 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る。
- DISABLE オプションによって、制約が定義されるが使用可能にはならない。

## 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、その外部キーと指定した主キーまたは一意キー（参照キーと呼ばれる）の間の関連を設定します。この関連において、外部キーを持つ表を子表と呼び、参照キーを持つ表を親表と呼びます。次の制限に注意してください。

- 子表と親表は同一データベース上に設定されていなければならない。このため、分散データベースの異なるノード上には設定できません。Oracle では、データベース・トリガーによって分散データベースのノード間の参照整合性を維持できます。データベース・トリガーの使用方法については、『Oracle8 Server アプリケーション開発者ガイド』を参照してください。
- 外部キーと参照キーは、同一の表に設定できる。この場合、親表と子表は同一の表となります。

参照整合性制約を満たすには、子表の各行で次の条件の 1 つが成立する必要があります。

- 行の外部キーの値は、参照キーの値として親表の行の 1 つに存在しなければならない。子表の行は、親表の参照キーに依存しています。
- 外部キーを構成する列の 1 つの値は NULL でなければならない。

参照整合性制約は子表で定義します。参照整合性制約の定義で、次のキーワードのいずれかを指定できます。

FOREIGN KEY	外部キーを構成する子表の列または列の組合せを指定します。このキーワードは、表制約句を使用して外部キーを定義する場合にだけ使用してください。
REFERENCES	親表と参照キーを構成する列または列の組合せを指定します。  親表だけを指定して列名を指定しないと、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、外部キー列と同一の数とデータ型で構成されていなければなりません。
ON DELETE CASCADE	子表に依存行を持つ親表の中の参照キーの値を削除できるように、参照整合性を維持するために子表から依存行が自動的に削除されます。  このオプションを指定しないと、子表に依存行を持つ親表の中の参照キーの値は削除できません。

子表で参照整合性制約を定義する前に、親表で参照される **UNIQUE** 制約または **PRIMARY KEY** 制約を、あらかじめ定義しておかなければなりません。また、親表が自スキーマ内に設定されていること、または親表の参照キー列に対する **REFERENCES** 権限が必要です。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かなければなりません。

**AS** 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。

外部キー列は、データ型の **LONG** や **LONG RAW** にはできません。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。

## 参照整合性制約の定義

*column\_constraint* 構文を使用して、外部キーが 1 つの列から構成される場合の参照整合性制約を定義できます。

**例.** 次の文では、EMP 表を作成し、DEPT 表の主キー DEPTNO 列を参照する DEPTNO 列に対して外部キーを定義し、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、つまりどの部門にも属さない従業員がいても構いません。これを避けるときは、EMP 表の *deptno* 列に、**REFERENCES** 制約に加えて **NOT NULL** 制約を作成します。

この制約を定義し使用可能にする前に、DEPT 表の主キーとして DEPTNO 列を指定する制約を定義し使用可能におかなければなりません。この制約の定義は、「例」(4-192 ページ)を参照してください。

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* 構文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なので、参照キーの列名の指定は任意です。

上記の文で、DEPTNO 列のデータ型が指定されていないことに注意してください。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* 構文を使っても、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 CASCADE** オプション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 CASCADE による参照整合性のメンテナンス

**ON DELETE CASCADE** オプションを使用すると、親表の中の参照キーの値を削除できるようになり、参照整合性をメンテナンスするために子表の依存行が自動的に削除されます。

**例.** 次の例では、EMP 表を作成し、参照整合性制約の FK\_DEPTNO を定義し使用可能にして、ON DELETE CASCADE オプションを使用します。

```
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) CONSTRAINT fk_deptno
    REFERENCES dept(deptno)
    ON DELETE CASCADE );
```

ON DELETE CASCADE オプションが指定されているため、DEPT 表の DEPTNO 値が削除されると、EMP 表の依存行の 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。

### 複合キーによる参照整合性制約

複合外部キーは列の組合せを構成する外部キーです。複合外部キーは最大 16 列で構成されます。複合外部キーによって参照整合性制約を定義するには、*table\_constraint* 構文を使用してください。列制約は単一の列に対してだけ規則を適用するので、*column\_constraint* 構文は使用できません。複合外部キーは、複合一意キーまたは複合主キーを参照しなければなりません。

複合キーによる参照整合性制約を満たすには、子表の各行で次の条件の 1 つが成立する必要があります。

- 外部キー列の値は、親表の行の参照キー列の値と一致しなければならない。
- 少なくとも 1 つの外部キー列の値は NULL でなければならない。

**例.** 次の文は、PHONE\_CALLS 表の AREACO 列と PHONENO 列を組み合わせて、外部キーを定義し使用可能にします。

```
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 表内の電話番号から構成されます。この制約を定義し使用可能にする前に、あらかじめ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定義し、使用可能にしておかなければなりません。

EXCEPTIONS オプションの結果、制約に反している PHONE\_CALLS 表の行に関する情報が WRONG\_NUMBERS 表に書き込まれます。

## CHECK 制約

CHECK 制約は、明示的に条件を定義します。制約を満たすには、表の各行が TRUE または不明 (NULL のため) のいずれかの条件になっていなければなりません。条件については、「条件」(3-84 ページ) の *condition* の構文の説明を参照してください。CHECK 制約の条件では、その表の中のどの列でも参照できますが、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、USERENV のコール
- 疑似列 CURRVAL、NEXTVAL、LEVEL、ROWNUM
- 完全に指定されていない日付定数

Oracle が特定の列に対する CHECK 制約条件を評価するときには、必ずその条件で示されている列名がその行の列値を参照します。

1 つの列に対して複数の CHECK 制約を作成する場合には、制約の目的が矛盾しないように、設計時には十分注意してください。CHECK 条件が相互に排他的かどうかについては、Oracle は検証しません。

**例 1.** 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、列の値が制限されます。

CHECK\_DEPTNO 部門番号を必ず 10 ～ 99 の範囲内にします。

CHECK\_DNAME 部門名はすべて大文字にします。

**CHECK\_LOC**      部門の所在地を Dallas、または Boston、New York、Chicago のいずれかに制限します。

各 CONSTRAINT 句に **DISABLE** オプションが指定されているため、Oracle は、これらの制約を定義するだけで使用可能にはしません。

*column\_constraint* 構文によって定義した **CHECK** 制約は、他のタイプの制約とは異なり、定義される列に限らず、表の任意の列に対しても規則を適用します。

**例 2.** 次の文は、EMP 表を作成し、表制約句を使って **CHECK** 制約を定義して使用可能にします。

```
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、つまり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 を超えてはなりません。
- 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 によって制約に対して名前が生成されます。

**例 3.** 次の文は、**PRIMARY KEY** 制約および 2 つの参照整合性制約、**NOT NULL** 制約、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_low CHECK (quantity > 0),
 cost NUMBER
```

```
CONSTRAINT check_cost CHECK (cost > 0) );
```

この制約により、表のデータに対して次の規則を適用できます。

PK_OD	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、次の条件が真でなければなりません。 <ul style="list-style-type: none"> <li>表の ORDER_ID 列と PART_NO 列に同じ値の組合せを持つ行が 2 つない。</li> <li>表の ORDER_ID 列と PART_NO 列のいずれにも NULL を持つ行がない。</li> </ul>
FK_OID	SCOTT のスキーマ内の ORDER 表の 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 列に追加される新しい値はすべて、SCOTT.ORDER.ORDER_ID 列にあらかじめ設定されていなければなりません。
FK_PNO	SCOTT が所有する PART 表の 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 列に追加される新しい値はすべて、SCOTT.PART.PART_NO 列にあらかじめ設定されていなければなりません。
NN_QTY	QUANTITY 列に対して NULL を禁止します。
CHECK_QTY	QUANTITY 列の値を必ずゼロ (0) よりも大きくします。
CHECK_COST	COST 列の値を必ずゼロ (0) よりも大きくします。

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* 構文が列定義よりも先に位置することに注目してください。例 4 では、表の主キーを定義する *table\_constraint* 構文は列定義の後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 制約よりも、それぞれ 1 つのビジネス・ルールだけ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。制約が矛盾しているときには、その制約を示すエラー・メッセージが戻ります。エラーが検出された制約が 1 つのビジネス・ルールだけを有効にする場合、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・ルールをより正確に特定できます。

## DEFERRABLE 制約

表制約と列制約は、DEFERRABLE または NOT DEFERRABLE と指定できます。DEFERRABLE を指定すると、トランザクションがコミットされるまで制約がチェックされません。デフォルトは NOT DEFERRABLE です。

DEFERRABLE を指定すると、制約の初期状態も INITIALLY DEFERRED と指定できるので、トランザクションを DEFERRED モードでも開始できます。また、DEFERRABLE 制約の初期状態を INITIALLY IMMEDIATE に指定し、トランザクションを NOT DEFERRED モードでも開始できます。

**例 1.** 次の文では、SCORES 列に対して NOT DEFERRABLE INITIALLY IMMEDIATE のチェック制約を指定して、表 GAMES を作成します。

```
CREATE TABLE games (scores NUMBER CHECK (scores >= 0));
```

**例 3.** 次の文は、列に対する一意の制約を INITIALLY DEFERRED DEFERRABLE と定義します。

```
CREATE TABLE orders  
  (ord_num NUMBER CONSTRAINT unq_num UNIQUE (ord_num)  
   INITIALLY DEFERRED DEFERRABLE);
```

制約は NOT DEFERRABLE INITIALLY DEFERRED とは定義できません。

トランザクションごとに、各 DML 文の実行後またはトランザクションがコミットされた時に遅延可能制約をチェックするかどうかを指定するには、SET CONSTRAINT(S) (4-509 ページ) を使用します。制約の遅延可能状態は変更できないので、変更するときは制約を削除し、作成し直す必要があります。

遅延制約の詳細は、『Oracle8 Server 管理者ガイド』および『Oracle8 Server 概要』を参照してください。

## 制約の使用可能と使用禁止の切替え

制約には、DISABLE または ENABLE NOVALIDATE、ENABLE VALIDATE という 3 つの状態があります。

制約を使用禁止状態 (DISABLE) から使用可能で妥当性検査された状態 (ENABLE VALIDATE) に変えるには、表に排他ロックをかける必要があります。これは、すべての既存データの妥当性をチェックしている間、新規データを表に挿入できないようにするためです。このため、1 度に 1 つの制約だけしか使用可能にできません。新しい制約を使用可能にするには、そのつどシリアル走査によって既存のすべての行をチェックする必要があります。

表のロックを避けるには、ENABLE 句 (4-414 ページ) を使用して、制約を ENABLE NOVALIDATE 状態にしてください。この状態では、表の新しい DML 文はすべて妥当性検査されるので、表への同時アクセスを防ぐ必要がなくなります。

また、ENABLE NOVALIDATE によって、表のいくつかの制約を同時に ENABLE VALIDATE 状態にすることもできます。Oracle が、既存のデータの妥当性検査のために実行する各走査は、可能な場合はパラレルで実行することもできます。

複数の制約を同時に ENABLE VALIDATE 状態にするには、個々のセッションから複数の ALTER TABLE コマンドを発行する必要があります。

### 主キーおよび一意キー制約を使用可能にする

主キー制約または一意キー制約を使用可能にすると、制約を適用する一意索引が自動的に作成されます。この後に制約が使用禁止になると、この索引は削除されます。したがって、制約が使用可能になるたびに索引が再作成されます。

この動作への改善策として、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たに作成し、その後で一意でない索引を作成するか既存の一意でない索引を使用して、制約を適用します。制約が使用禁止の場合に一意でない索引は削除されないため、主キー制約や一意キー制約に対する ENABLE 操作は、（索引がすでにあるため）ほとんど瞬時に行われます。余分な索引も削除されます。

PRIMARY KEY 制約および UNIQUE の制約の詳細は、ENABLE 句（4-414 ページ）を参照してください。

## 関連項目

CREATE TABLE（4-303 ページ）

ALTER TABLE（4-105 ページ）

ENABLE 句（4-414 ページ）

DISABLE 句（4-375 ページ）

SET CONSTRAINT(S)（4-509 ページ）

ALTER SESSION（4-58 ページ）

---

# CREATE CLUSTER

## 用途

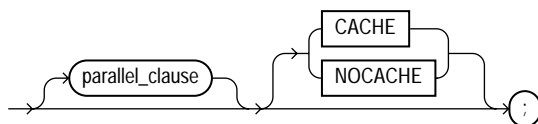
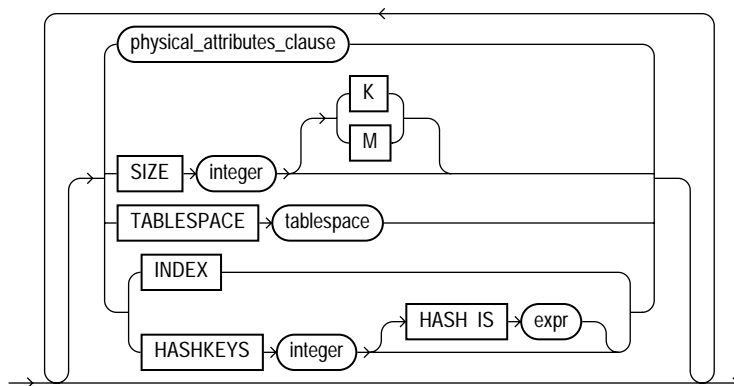
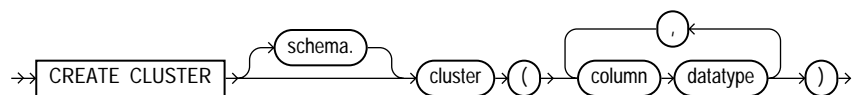
クラスタを作成します。クラスタとは、1 つ以上の列を共有する、1 つ以上の表からなるスキーマ・オブジェクトです。「使用上の注意」(4-205 ページ) および「クラスタへの表の追加」(4-208 ページ) を参照してください。

## 前提条件

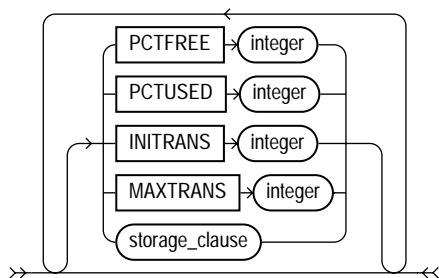
自スキーマにクラスタを作成するには、**CREATE CLUSTER** システム権限が必要です。別のユーザーのスキーマにクラスタを作成するには、**CREATE ANY CLUSTER** システム権限が必要です。また、クラスタを設定するスキーマの所有者は、クラスタが定義されている表領域に対する領域の割当て制限、または **UNLIMITED TABLESPACE** システム権限のいずれかを持っていなければなりません。



## 構文



**physical\_attributes\_clause ::=**



`storage_clause`: `STORAGE` 句 (4-518 ページ) を参照。

`parallel_clause`: `PARALLEL` 句 (4-462 ページ) を参照。

キーワードとパラメータ

<i>schema</i>	作成するクラスタが定義されるスキーマです。 <i>schema</i> を指定しないと、現行スキーマにクラスタが作成されます。
<i>cluster</i>	作成するクラスタの名前です。
<i>column</i>	クラスタ・キーの列の名前です。「クラスタ・キー」(4-206 ページ) も参照してください。
<i>datatype</i>	クラスタ・キー列のデータ型です。なお、クラスタ・キー列は <b>LONG</b> と <b>LONG RAW</b> 以外のデータ型を持つことができます。列のデータ型が位取り 0 の <b>INTEGER</b> 型や <b>NUMBER</b> 型でない場合は、 <b>HASH IS</b> 句は使用できません。データ型の説明は、「データ型」(2-5 ページ) の項を参照してください。
<i>physical_attributes_clause:</i>	
<b>PCTUSED</b>	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パーセントの整数値で表現されます。
<b>PCTFREE</b>	将来的な拡張に備えて、クラスタの各データ・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パーセントの整数値で表現されます。
<b>INITRANS</b>	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 <b>MAXTRANS</b> パラメータの値より大きな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 <b>INITRANS</b> の値と 2 のどちらか大きい方の値になります。
<b>MAXTRANS</b>	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、 <b>INITRANS</b> パラメータの値よりも小さい値は指定できません。このパラメータの最大値は 255 です。デフォルト値は、クラスタが定義される領域に対する <b>MAXTRANS</b> 値です。  <b>PCTUSED</b> および <b>PCTFREE</b> 、 <b>INITRANS</b> 、 <b>MAXTRANS</b> の各パラメータの詳細は、 <b>CREATE TABLE</b> (4-303 ページ) を参照してください。
<b>SIZE</b>	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。このとき、 <b>K</b> または <b>M</b> を使用して、 <b>KB</b> または <b>MB</b> 単位でも指定できます。このパラメータを指定しないと、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。「クラスタ・サイズ」(4-207 ページ) を参照してください。
<b>TABLESPACE</b>	クラスタを作成する表領域を指定します。
<i>storage_clause</i>	データ・ブロックをクラスタに割り当てる方法を指定します。 <b>STORAGE</b> 句 (4-518 ページ) を参照してください。
<b>INDEX</b>	索引クラスタを作成します。索引クラスタでは、行がクラスタ・キー値に基づき、まとめて格納されます。「クラスタの種類」(4-206 ページ) を参照してください。

HASHKEYS	ハッシュ・クラスタを作成し、ハッシュ・クラスタに対するハッシュ値の数を指定します。Oracle は、ハッシュ値の実際の数を決めるために、HASHKEYS 値を一番近い次の素数に切り上げます。このパラメータの最小値は 2 です。INDEX オプションも HASHKEYS パラメータも指定しないと、デフォルトでは索引クラスタが作成されます。「クラスタの種類」(4-206 ページ)を参照してください。
HASH IS	<p>ハッシュ・クラスタに対するハッシュ関数として使う式を指定します。式は次の条件を満たす必要があります。</p> <ul style="list-style-type: none"> <li>• 正の値をとる。</li> <li>• 式全体の値が位取り 0 の数になる場合には、1 つ以上の列を持ち、任意のデータ型の列が参照されている (たとえば、NUM_COLUMN * length(VARCHAR2_COLUMN))。</li> <li>• ユーザー定義の PL/SQL 関数を参照しない。</li> <li>• SYSDATE、USERENV、TO_DATE、UID、USER、LEVEL、ROWNUM を参照しない。</li> <li>• 定数をとらない。</li> <li>• 副問合せを持たない。</li> <li>• (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たない。</li> </ul> <p>HASH IS 句を指定しないと、ハッシュ・クラスタに対して内部ハッシュ関数が使われます。</p> <p>ハッシュ列のクラスタ・キーは、任意のデータ型からなる 1 つ以上の列を持つことができます。複合クラスタ・キーまたは整数でない列から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しなければなりません。</p>
<i>parallel_clause</i>	クラスタを作成するときは使用する並列度を指定し、作成後にクラスタに問合せをするときは使用するデフォルトの並列度を指定します。PARALLEL 句 (4-462 ページ) も参照してください。
CACHE	全表走査の実行時に、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入れます。このオプションは、小規模な参照表で有効です。
NOCACHE	全表走査の実行時に、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入れます。これはデフォルトの動作です。

## 使用上の注意

クラスタとは、1 つ以上の列を共有する、1 つ以上の表からなるスキーマ・オブジェクトです。これらの共通の列において同一の値を共有する表の行は、物理的にまとめてデータベースに格納されます。

クラスタ化によって、データベース内の行の物理記憶域の制御を強化できます。クラスタ化によって、クラスタ化表へのアクセス時間を短縮し、表を格納するために必要な領域を節約できます。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。

1 ハッシュ値に対するすべての列が 1 データ・ブロックに入りきらない場合は、ハッシュ・クラスタを使わないでください。ハッシュ・クラスタにデータ・ブロック・サイズを超える

行の挿入や更新を行うと、ブロックが一杯になり、残りの行を入れるために行の連鎖が発生するため、パフォーマンスが非常に低くなります。

一般的には、SQL 文で頻繁にクラスタ・キー列に結合される表だけをクラスタ化してください。複数の表をクラスタ化することによって結合のパフォーマンスは改善されますが、その反面、全表走査、INSERT 文、クラスタ・キー値を変更する UPDATE 文のパフォーマンスが低下する可能性が高くなります。クラスタ化する前に、データに対して実行する予定の操作に照らした利点と不利益な点について検討してください。クラスタ化によるパフォーマンスの説明は、『Oracle8 Server チューニング』を参照してください。

## クラスタ・キー

CREATE CLUSTER コマンドによって定義される列は、クラスタ・キーを構成します。これらのクラスタ列は、各クラスタ化表の列と名前で一致している必要はありませんが、データ型とサイズの両方で一致していなければなりません。

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。そのかわりに、整合性制約をクラスタに属している表に対応付けることができます。

## クラスタの種類

クラスタは、索引クラスタかハッシュ・クラスタのどちらかになります。

### 索引クラスタ

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。各クラスタ・キー値は、そのキーを持つ表や行の数には関係なく、各データ・ブロックに 1 度だけ格納されます。これによってディスク領域を節約でき、多くの操作のパフォーマンスが上がります。

索引クラスタは次のような場合に便利です。

- クラスタ・キー値の範囲を指定して行を検索する問合せを行う場合。
- クラスタ表がどれだけ大きくなるか予測できない場合。

索引クラスタの作成後、クラスタ内の表に対して 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。クラスタ索引の作成については、CREATE INDEX (4-233 ページ) を参照してください。索引の列の場合と同様に、クラスタ・キー内の列の順序もクラスタ索引の構造に影響します。

クラスタ索引により、クラスタ・キーに基づいてクラスタ内の行に迅速にアクセスできます。クラスタ内の行をクラスタ・キー値に基づいて検索する SQL 文を発行すると、Oracle がクラスタ索引からクラスタ・キー値を求め、ROWID に基づいてクラスタ内で行を探し出します。

## ハッシュ・クラスタ

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。各行のハッシュ値は、そのクラスタのハッシュ関数が戻す値です。ハッシュ・クラスタの作成時には、ハッシュ関数を指定しても、Oracle の内部ハッシュ関数を使用しても構いません。クラスタ・キー値はクラスタ内の各行に格納されますが、ハッシュ値は実際にクラスタ内に格納されるわけではありません。

ハッシュ・クラスタは次のような場合に便利です。

- すべてのクラスタ・キー列を対象とする等価条件に基づいて行を検索する問合せを行う場合。
- クラスタ内の表の大きさが変化しない場合、またはクラスタの作成時にそのクラスタに必要な最大行数と最大領域サイズを決定できる場合。

ハッシュ関数によって、クラスタ・キー値に基づいて表の行にアクセスできます。クラスタ・キー値に基づいてクラスタの行を検索する SQL 文を発行すると、Oracle は指定したクラスタ・キー値でハッシュ関数を実行し、結果として得られるハッシュ値を基に一致する行を検出します。複数のクラスタ・キー値が同一のハッシュ値に割り当てられることがあるため、Oracle はその行のクラスタ・キー値を検証しなければなりません。このプロセスは索引の探索を必要としないため、結果的には、索引クラスタに対するプロセスよりも I/O が少なくて済みます。

Oracle の内部ハッシュ関数は、HASHKEYS から 1 を引いた値から 0 までの範囲で戻り値を返します。HASH IS 句を使用して列を指定した場合には、列の値はこの範囲内に収まる必要はありません。Oracle は列値を HASHKEYS 値で除算し、その余りをハッシュ値として使用します。NULL に対するハッシュ値は、HASHKEYS-1 です。また、Oracle はハッシュ値の実際の数値を求めるために、近似値の素数まで HASHKEYS 値を切り上げます。この切り上げによって、ハッシュの衝突の可能性、つまり複数のクラスタ・キー値が同一のハッシュ値を持つ可能性が少なくなります。

ハッシュ・クラスタに対するクラスタ索引は作成できないため、ハッシュ・クラスタ・キーにおいて索引を作成する必要はありません。

## クラスタ・サイズ

Oracle は SIZE パラメータを基にして、クラスタ・キー値またはハッシュ値に対応する行を確保するための領域を決定します。次に、この領域によって、データ・ブロックあたり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には、Oracle は次に大きな約数を使用します。SIZE 値がデータ・ブロック・サイズよりも大きな場合には、Oracle はクラスタまたはハッシュ値ごとに少なくとも 1 データ・ブロックを確保して、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・キー値を有する行に対して確保する必要がある領域を決定するときに、クラスタ・キーの長さを考慮に入れます。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** 列を問い合わせます。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。

データ・ブロックあたりのクラスタ・キー値またはハッシュ・キー値の最大数はクラスタ単位で調整されますが、各クラスタ・キー値やハッシュ・キー値に対して確保される領域は等しくありません。クラスタ・キー値またはハッシュ・キー値あたりに格納されるデータが固定長になることはあまりないため、この領域を変更することによって、より効率的にデータを格納できます。

平均的なクラスタ・キー値またはハッシュ・キー値で必要となる領域より **SIZE** 値が少ない場合は、1つのクラスタ・キー値またはハッシュ・キー値に対するデータを、複数のデータ・ブロックにまたがって格納する必要があります。ただし反対に **SIZE** を大きくすると、その結果として領域が無駄になってしまいます。

ハッシュ・クラスタの作成時に、Oracle は **SIZE** パラメータと **HASHKEYS** パラメータの各値に基づいて、クラスタに対して領域を迅速に割り当てます。Oracle がクラスタに対して領域を割り当てる方法については、『Oracle8 Server 概要』を参照してください。

## クラスタへの表の追加

**CREATE TABLE** 文に **CLUSTER** 句を指定することによって、既存のクラスタ上に表を追加できます。1つのクラスタには最大 32 までの表を構成できますが、4～5 以上の表をクラスタ化すると、多くの場合クラスタ化によるパフォーマンス上の効果がなくなります。

クラスタ内の表はすべて、**PCTUSED**、**PCTFREE**、**INITRANS**、**MAXTRANS**、**TABLESPACE**、**STORAGE** の各パラメータで指定したクラスタの記憶特性を持っています。

**例 1.** 次の文は、クラスタ・キー列 **DEPARTMENT\_NUMBER**、クラスタ・サイズ 512 バイト、領域パラメータ値を指定した **PERSONNEL** という名前の索引クラスタを作成します。

```
CREATE CLUSTER personnel
  ( department_number NUMBER(2) )
  SIZE 512
  STORAGE ( INITIAL 100K NEXT 50K PCTINCREASE 10 );
```

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 >= SYSDATE),
    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);
```

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。

**例 2.** 次の文は、クラスタ・キー列 DEPARTMENT\_NUMBER、最大ハッシュ・キー値 503、各サイズが 512 バイト、領域パラメータ値を指定した PERSONNEL という名前のハッシュ・クラスタを作成します。

```
CREATE CLUSTER personnel
  ( department_number NUMBER )
  SIZE 512 HASHKEYS 500
  STORAGE (INITIAL 100K NEXT 50K PCTINCREASE 10);
```

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。

**例 3.** 次の文は、HOME\_AREA\_CODE 列と HOME\_PREFIX 列からなるクラスタ・キーを持つ PERSONNEL という名前のハッシュ・クラスタを作成し、これらの列を含んだ 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);
```

## 関連項目

CREATE INDEX (4-233 ページ)

CREATE TABLE (4-303 ページ)

「索引構成表」 (4-123 ページ)

STORAGE 句 (4-518 ページ)

---

# CREATE CONTROLFILE

## 用途

次のいずれかの状況の場合に、制御ファイルを再作成します。

- 既存の制御ファイルの全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。
- 次のものの最大数を変更する場合 (REDO ログ・ファイル・グループまたは REDO ログ・ファイル・メンバー、アーカイブ REDO ログ・ファイル、データ・ファイル、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)。

「使用上の注意」(4-213 ページ) を参照してください。

---

---

**警告：** このコマンドを使用する前に、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とっておくことをお勧めします。

---

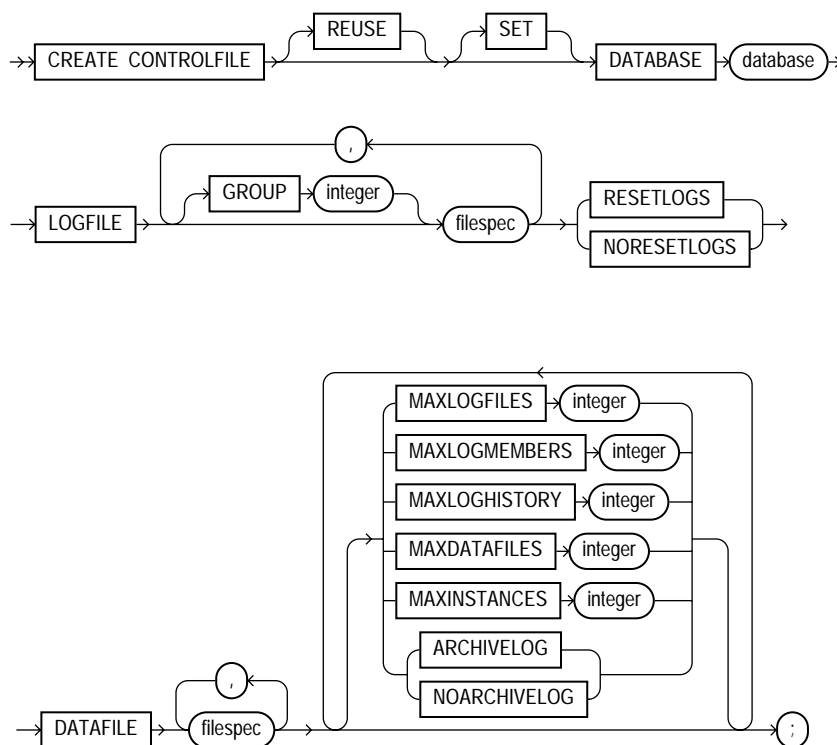
---

## 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なりません。



## 構文



`filespec`: 「Filespec」 (4-428 ページ) を参照。

## キーワードとパラメータ

REUSE	初期化パラメータ <code>CONTROL_FILES</code> 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。このため、現在保存されている情報は無視され、上書きされます。このオプションを指定しないと、既存の制御ファイルがある場合に、エラーが戻ります。
DATABASE	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 <code>CREATE DATABASE</code> 文または <code>CREATE CONTROLFILE</code> 文で設定した、既存のデータベース名でなければなりません。
SET DATABASE	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	データベースに対する REDO ログ・ファイル・グループを指定します。すべての REDO ログ・ファイル・グループの全メンバーを指定しなければなりません。「Filespec」 (4-428 ページ) にある <code>filespec</code> の構文の説明を参照してください。

RESETLOGS	LOGFILE 句に指定したファイルの内容を無視します。LOGFILE 句に指定したファイルは存在していなくても構いません。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 に全 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。このオプションを使用した後は、RESETLOGS オプションを指定した ALTERDATABASE コマンドを使ってデータベースをオープンする必要があります。
NORESETLOGS	LOGFILE 句に指定した全ファイルを、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ことを指定します。LOGFILE 句に指定したファイルは必ず存在しなければなりません。またバックアップからの復元ではなく、現行の REDO ログ・ファイルでなければなりません。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。GROUP 値を指定すると、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。
DATAFILE	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なければなりませんが、メディア回復を必要とする復元バックアップでも構いません。「Filespec」(4-428 ページ)にある filespec の構文の説明を参照してください。
MAXLOGFILES	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 の最大値以上でなければなりません。  インスタンスでアクセスできる REDO ログ・ファイル・グループの数は初期化パラメータ LOG_FILES の制限も受けます。
MAXLOGMEMBERS	REDO ログ・ファイル・グループのメンバーつまり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。最大値とデフォルト値は、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	Oracle8 Parallel Server の自動メディア回復でアーカイブ REDO ログ・ファイル・グループの最大数を指定します。Oracle はこの値を基にして、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る領域の大きさを決定します。最小値は 0 です。デフォルト値は MAXINSTANCE 値の倍数で、各オペレーティング・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だけを受けます。このパラメータは、Parallel Server オプション付きの 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だけに有効です。
MAXDATAFILES	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。番号が MAXDATAFILES より大きく、DB_FILES 以下のファイルを追加しようとする、データ・ファイル・セクションにもっと多くのファイルが入るように、Oracle8 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	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 の値よりも優先されます。最小値は 1 です。最大値とデフォルト値は、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	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するモードを設定します。このオプションを指定すると、インスタンス回復だけでなくメディア回復もできるようになります。
NOARCHIVELOG	ARCHIVELOG オプションも NOARCHIVELOG オプションも指定しないと、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 コマンドを使用すると、ARCHIVELOG モードと NOARCHIVELOG モードを切り替えることができます。

## 使用上の注意

CREATE CONTROLFILE 文を発行する前に、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とることをお勧めします。

CREATE CONTROLFILE 文を発行すると、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。この文のオプションを指定しないと、前回の制御ファイルのオプションではなく、デフォルトのオプションが採用されます。制御ファイルが正常に作成された後で、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。必ずメディア回復を実行してから、データベースをオープンしてください。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とることをお勧めします。

このコマンドの使用の詳細は、『Oracle8 Server 管理者ガイド』を参照してください。

**例** . 次の例では、制御ファイルが再作成されます。

```
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;
```

## 関連項目

CREATE DATABASE コマンド (4-214 ページ)

「Filespec」 (4-428 ページ)

## CREATE DATABASE

---

### 用途

汎用的なデータベースを作成します。オプションは次のとおりです。

- インスタンスまたはデータ・ファイル、REDO ログ・ファイル・グループ、REDO ログ・ファイル・メンバーの最大数の設定。
- データ・ファイルと REDO ログ・ファイルの名前とサイズの指定。
- REDO ログの使用モードの選択。
- 各国文字キャラクタ・セットおよびデータベース・キャラクタ・セットの指定。

これらの用途のうちの一部について、「例」（4-218 ページ）があります。

---

---

**警告：** このコマンドを実行すると、データベースの初期使用に備えて、指定ファイル内に現在あるデータは消去されます。そのことを十分理解した上で、このコマンドを使用してください。

---

---

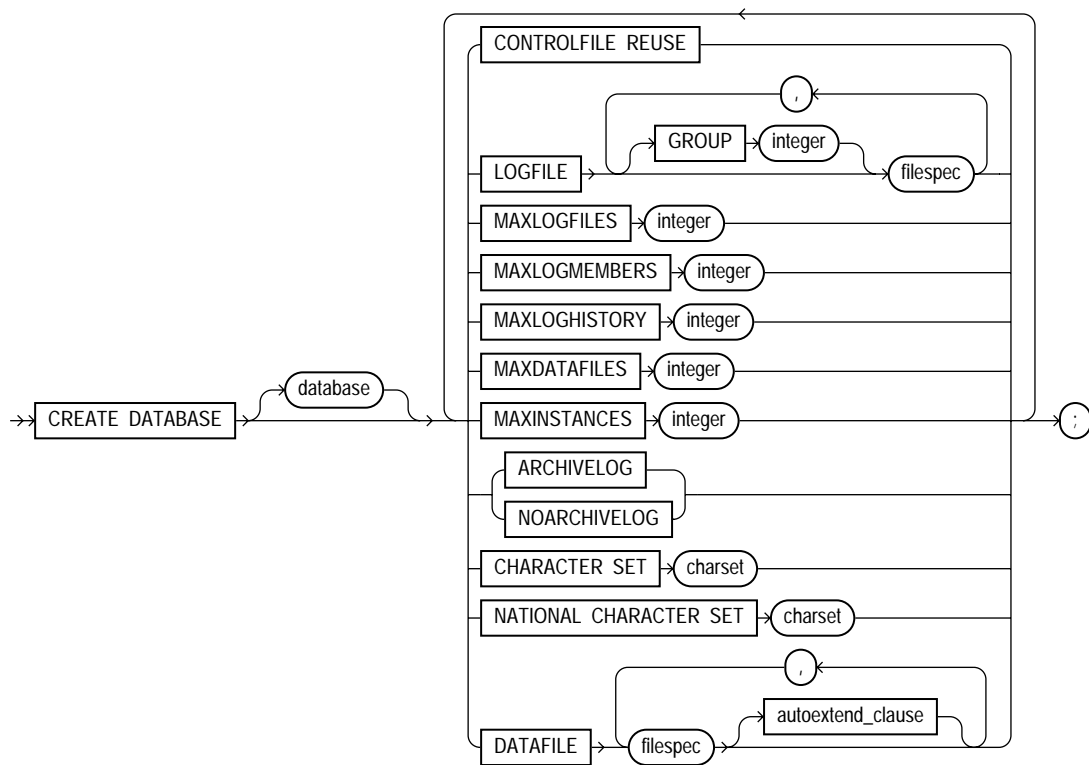
このコマンドは、データベースの初期使用に備えて、指定した既存のデータ・ファイル上のデータがすべて消去されます。したがって、既存のデータベースに対してこのコマンドを実行すると、データ・ファイル上の全データが失われます。

このコマンドを指定すると、データベースの作成後、データベースは初期化パラメータ **PARALLEL\_SERVER** で指定したモードでマウントおよびオープンされて、通常の用途に使用できるようになります。

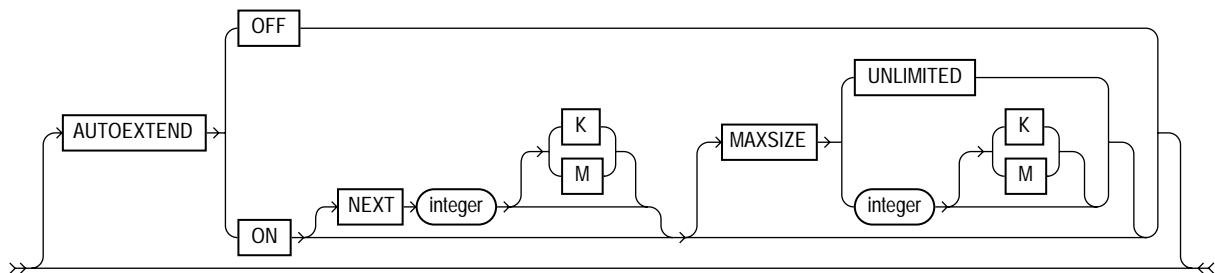
### 前提条件

OSDBA ロールを使用可能にする必要があります。

## 構文



`autoextend_clause ::=`



## キーワードとパラメータ

<i>database</i>	<p>作成するデータベースの名前を、最大 8 バイト長で指定します。データベース名には ASCII 文字だけを使用できます。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。データベース名は、「スキーマ・オブジェクトの命名規則」(2-43 ページ) で説明した規則に従って指定してください。</p> <p><b>注意：</b>データベース名には、ヨーロッパやアジアのキャラクタ・セットの中の特殊文字は使用できません。たとえば、ウムラウトは使用できません。</p> <p>CREATE DATABASE 文でデータベース名を指定しないと、初期化パラメータ DB_NAME で指定した名前が採用されます。</p>
CONTROLFILE REUSE	<p>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。このため、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、このオプションは、初めてデータベースを作成するときではなく、データベースを再作成するときに限って使用します。制御ファイルを既存のファイルよりも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には、このオプションは使用できません。</p> <p>MAXLOGFILES、MAXLOGMEMBERS、MAXLOGHISTORY、MAXDATAFILES、MAXINSTANCES がこのようなパラメータです。</p> <p>このオプションを指定しなかった場合に、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と、エラー・メッセージが戻ります。</p>
LOGFILE	<p>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 <i>filespec</i> は、1 つ以上の REDO ログ・ファイルのメンバー、すなわちコピーを含む REDO ログ・ファイル・グループを指定します。filespec の構文の説明は、「Filespec」(4-428 ページ) を参照してください。CREATE DATABASE 文に指定した全 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。</p>
GROUP	<p>REDO ログ・ファイル・グループを特定し、その範囲は 1 から MAXLOGFILES パラメータ値までの間となります。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と、この値は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、動的性能表の V\$LOG によって調べることができます。</p>
	<p>LOGFILE 句を指定しないと、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・ファイルの名前とサイズは、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。</p>
MAXLOGFILES	<p>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、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。</p> <p>インスタンスでアクセスできる REDO ログ・ファイル・グループの数は初期化パラメータ LOG_FILES の制限を受けます。</p>

MAXLOGMEMBERS	REDO ログ・ファイル・グループのメンバーつまり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。最大値とデフォルト値は各オペレーティング・システムによって異なります。
MAXLOGHISTORY	<p>Parallel Server オプション付きの Oracle の自動メディア回復用のアーカイブ 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 です。デフォルト値は MAXINSTANCES 値の倍数で、各オペレーティング・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だけを受けます。</p> <p>注意：このパラメータは、Parallel Server オプション付きの Oracle をパラレル・モードで、アーカイブ・ログ・モードを使用可能にして運用している場合にだけ有効です。</p>
MAXDATAFILES	<p>CREATE DATABASE または CREATE CONTROLFILE 実行時の、制御ファイルのデータ・ファイル・セクションの初期サイズを指定します。番号が MAXDATAFILES より大きく、DB_FILES 以下のファイルを追加しようとする、データ・ファイル・セクションにもっと多くのファイルが入るように、Oracle8 の制御ファイルが自動的に拡張されます。</p> <p>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。</p>
MAXINSTANCES	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 の値よりも優先されます。最小値は 1 です。最大値とデフォルト値は各オペレーティング・システムによって異なります。
ARCHIVELOG	REDO ログ・ファイル・グループに対して ARCHIVELOG モードを設定します。このモードでは、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あります。このオプションを指定するとメディア回復ができるようになります。
NOARCHIVELOG	<p>REDO ログ・ファイル・グループに対して NOARCHIVELOG モードを設定します。このモードでは、REDO ログ・ファイル・グループを再使用する前に、その内容をアーカイブする必要はありません。このオプションでは、メディア回復はできません。</p> <p>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 コマンドを使用すれば、ARCHIVELOG モードと NOARCHIVELOG モードを切り替えることができます。</p>
CHARACTER SET	データベースで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。データベースの作成後は、データベースのキャラクタ・セットを変更できません。サポートされているキャラクタ・セットと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。

	<p>次の固定幅マルチバイト・キャラクタ・セット（各国文字キャラクタ・セットとしてしか使用できない）を除き、サポートされているキャラクタ・セットであればどれも指定できます。</p> <p>JA16SJISFIXED</p> <p>JA16EUCFIXED</p> <p>JA16DBCSFIXED</p> <p>有効なキャラクタ・セットの詳細は、『Oracle8 Server リファレンス・マニュアル』を参照してください。</p>										
NATIONAL CHARACTER SET	<p>NCHAR または NCLOB、NVARCHAR2 と定義された列にデータを格納するときに使用する各国文字キャラクタ・セットを指定します。データベースの作成後は、各国文字キャラクタ・セットを変更できません。指定のない場合は、デフォルトとしてデータベース・キャラクタ・セットが各国文字キャラクタ・セットとして設定されます。有効なキャラクタ・セット名については、『Oracle8 Server リファレンス・マニュアル』を参照してください。</p>										
DATAFILE	<p>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。 <i>filespec</i> の構文の説明は、「Filespec」（4-428 ページ）を参照してください。ファイルはすべて SYSTEM 表領域の一部となります。この句を指定しないと、デフォルトでは 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とサイズは、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。</p> <p>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。</p>										
AUTOEXTEND	<p>データ・ファイルの自動拡張を使用可能または使用禁止にします。この句を指定しないと、データ・ファイルが自動的に拡張されません。</p> <table><tr><td>OFF</td><td>オンになっている場合に、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、ALTER DATABASE AUTOEXTEND コマンドまたは ALTER TABLESPACE AUTOEXTEND コマンドで再指定する必要があります。</td></tr><tr><td>ON</td><td>自動拡張を使用可能にします。</td></tr><tr><td>NEXT</td><td>エクステン트가さらに必要になったときに、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のサイズ（バイト単位）を指定します。K または M を使用して KB または MB 単位でもサイズを指定できます。デフォルト値は 1 データ・ブロックです。</td></tr><tr><td>MAXSIZE</td><td>データ・ファイルの自動拡張で使われる最大ディスク領域のサイズを指定します。</td></tr><tr><td>UNLIMITED</td><td>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。</td></tr></table>	OFF	オンになっている場合に、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、ALTER DATABASE AUTOEXTEND コマンドまたは ALTER TABLESPACE AUTOEXTEND コマンドで再指定する必要があります。	ON	自動拡張を使用可能にします。	NEXT	エクステン트가さらに必要になったときに、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のサイズ（バイト単位）を指定します。K または M を使用して KB または MB 単位でもサイズを指定できます。デフォルト値は 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われる最大ディスク領域の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します。
OFF	オンになっている場合に、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、ALTER DATABASE AUTOEXTEND コマンドまたは ALTER TABLESPACE AUTOEXTEND コマンドで再指定する必要があります。										
ON	自動拡張を使用可能にします。										
NEXT	エクステン트가さらに必要になったときに、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のサイズ（バイト単位）を指定します。K または M を使用して KB または MB 単位でもサイズを指定できます。デフォルト値は 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われる最大ディスク領域の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します。										

例

例 1. 次の文は、引数すべてにデフォルト値を使って小規模なデータベースを作成する例です。



```
CREATE DATABASE;
```

**例 2.** 次の文は、各引数すべてを完全に指定してデータベースを作成する例です。

```
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;
```

## 関連項目

ALTER DATABASE (4-15 ページ)

CREATE ROLLBACK SEGMENT (4-272 ページ)

CREATE TABLESPACE (4-325 ページ)

## CREATE DATABASE LINK

---

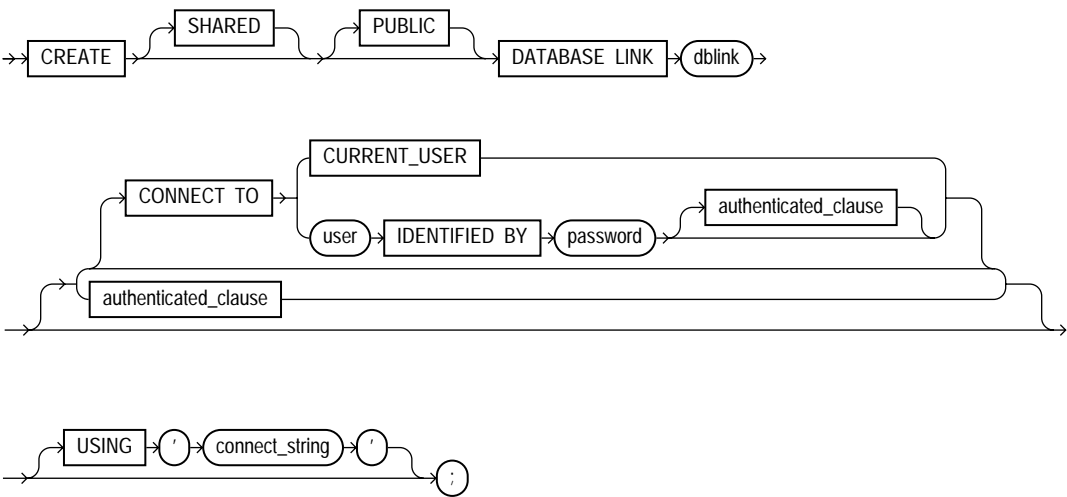
### 用途

データベース・リンクを作成します。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 データベースでも Oracle 以外のデータベースでも構いません。「使用上の注意」（4-222 ページ）も参照してください。

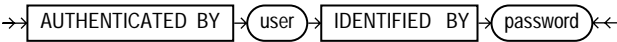
### 前提条件

プライベート・データベース・リンクを作成するには、CREATE DATABASE LINK の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 の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かなければなりません。Oracle 以外のシステムにアクセスする場合は、Oracle8 異機種間サービスを使用しなければなりません。

構文



authenticated\_clause ::=



キーワードとパラメータ

SHARED	1つのネットワーク接続を使用することによって複数のユーザー間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。このオプションは、マルチスレッド・サーバー構成でだけ使用できます。共有データベース・リンクの詳細は、『Oracle8 Server 分散システム』を参照してください。
PUBLIC	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。このオプションを指定しないと、データベース・リンクはプライベートとなり、作成ユーザー専用となります。
dblink	データベース・リンクの完全な名前または名前の一部を指定します。データベース・リンク命名のガイドラインについては、「リモート・データベース内のオブジェクトを参照」(2-50 ページ) および「現行ユーザーのデータベース・リンク」(4-223 ページ) を参照してください。
CONNECT TO	リモート・データベースに接続できるようにします。

CURRENT_USER	現行ユーザーのデータベース・リンクを作成します。現行のデータベース・リンクを使用するには、現行ユーザーが Oracle Security Server によってグローバル・ユーザーと認証されている必要があります。「現行ユーザーのデータベース・リンク」(4-223 ページ)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と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と、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と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)。
authenticated_clause	ターゲット・インスタンスのユーザー名と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とパスワードは、リモート・インスタンスで有効なユーザー名とパスワードでなければなりません。このユーザー名とパスワードは認証用としてだけ使用されます。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 オプションを使用している場合は、この句を必ず指定してください。
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。リモート・データベースの指定については、『Net8 管理者ガイド』を参照してください。

使用上の注意

他のユーザーのスキーマ内にはデータベース・リンクを作成できません。また、*dblink* にはスキーマ名を付けて指定できません。データベース・リンク名にはピリオドを指定できるため、たとえば RALPH.LINKTOSALES のような名前を付けると、スキーマ RALPH の中の LINKTOSALES という名前のデータベース・リンクと解釈されるのではなく、名前全体が自スキーマにあるデータベース・リンク名と解釈されます。

データベース・リンクを作成すると、そのリンクを利用して、リモート・データベース上の表やビューを参照できます。また、表名またはビュー名に *@dblink* を付加することによって、SQL 文の中でリモート表やビューを参照できます。SELECT コマンドを使うと、リモート表またはビューの問合せができます。Oracle を分散オプション付きで使用する場合には、次のいずれかのコマンドでもリモート表やビューにアクセスできます。

- DELETE (4-370 ページ)
- INSERT (4-449 ページ)
- LOCK TABLE (4-455 ページ)
- UPDATE (4-536 ページ)

PL/SQL 関数およびプロシージャ、パッケージ、データ型によるリモート表またはビューへのアクセスの詳細は、『Oracle8 Server アプリケーション開発者ガイド』を参照してください。

1 つの文で指定できる各種のデータベース・リンクの数は、初期化パラメータ OPEN\_LINKS の値に制限されます。

## 現行ユーザーのデータベース・リンク

現行ユーザーのデータベース・リンクは、ユーザー資格証明をもたないので、現行ユーザーとしてリモート・データベースに接続します。このリンクを使用する場合、現行ユーザーはローカル・データベースとリモート・データベースの両方でグローバル・アカウントをもつグローバル・ユーザーでなければなりません。どちらのデータベースも同じセキュリティ・ドメインに定義されている必要があります。

グローバル・ユーザーの作成については、**CREATE USER**（4-353 ページ）を参照してください。現行のデータベース・リンクの詳細は、『Oracle8 Server 分散システム』を参照してください。

### CURRENT\_USER

データベース・リンクを開始するストアド・オブジェクト（プロシージャ、ビュー、トリガーなど）を実行する場合に、**CURRENT\_USER** とはストアド・オブジェクトを作成したユーザー名であり、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 により作成された) プロシージャ **SCOTT.P** 内にあり、ユーザー **JANE** がプロシージャ **SCOTT.P** をコールした場合、現行ユーザーは **SCOTT** です。

データベース・リンクが直接使用される場合（つまり、ストアド・オブジェクト内から使用されているのではない場合）、現行ユーザーは接続ユーザーと同じです。

## 例

**例 1.** 次の例では、現行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';
```

**例 2.** 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** 内の表を問い合わせることができます。

```
SELECT *
FROM emp@sales.hq.acme.com
```

次のように **DML** コマンドを使用して、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。次の例では、SCOTT は ADAM の所有する DEPT 表に対するアクセス権を持っていることが前提です。

```
SELECT *
FROM adams.dept@sales.hq.acme.com
```

前の文では、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。次の文では、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;
```

**例 3.** 次の文では、SALES.HQ.ACME.COM という名前の共用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上で、パスワード TIGER を持つユーザー SCOTT 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';
```

**例 4.** 次の例では、現行ユーザーのデータベース・リンクを作成します。

```
CREATE DATABASE LINK sales.hq.acme.com
CONNECT TO CURRENT_USER
USING 'sales';
```

## 関連項目

CREATE SYNONYM (4-299 ページ)

CREATE USER (4-353 ページ)

DELETE (4-370 ページ)

INSERT (4-449 ページ)

LOCK TABLE (4-455 ページ)

SELECT (4-486 ページ)

UPDATE (4-536 ページ)

『PL/SQL ユーザーズ・ガイドおよびリファレンス』

『Oracle8 Server 分散システム』

# CREATE DIRECTORY

## 用途

CREATE DIRECTORY はディレクトリ・オブジェクトを作成するために使用します。このディレクトリ・オブジェクトは、データベースの外部に格納されている BFILE のアクセスと利用を管理するためのオペレーティング・システムのディレクトリを指します。ディレクトリとは、アクセスや利用の対象となるファイルが実際に格納されているサーバー・ファイル・システム上でのフルパス名の別名です。

## 前提条件

ディレクトリを作成するには、CREATE ANY DIRECTORY システム権限が必要です。

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しなければなりません。

## 構文



## キーワードとパラメータ

OR REPLACE	<p>ディレクトリのデータベース・オブジェクトがすでに定義されている場合に、これを再作成します。このオプションを指定すると、既存ディレクトリの定義の変更が、そのディレクトリに付与されているデータベース・オブジェクト権限の削除および再作成、再付与なしに実行できます。</p> <p>再定義したディレクトリに対する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そのディレクトリにアクセスできます。</p>
directory	<p>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。<i>directory</i> の最大長は 30 バイトです。ディレクトリ・オブジェクトをスキーマ名で修飾できません。「ディレクトリ・オブジェクト」(4-227 ページ) も参照してください。</p> <p><b>注意：</b> 指定したディレクトリが実際に存在するかどうかの検証は行われないため、オペレーティング・システム内の有効なディレクトリを指定するように注意してください。また、オペレーティング・システムで使用するパス名に大/小文字の区別がある場合は、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。)</p>



---

<code>'path_name'</code>	サーバー上の、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の完全パス名を指定します。一重引用符で囲む必要があります。したがって、パス名には大/小文字の区別があります。
--------------------------	---

---

## ディレクトリ・オブジェクト

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 を参照するときに、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。

Oracle の BFILE データ型により、外部ファイル・システムにアクセスできます。BFILE の列つまり属性には、ファイル自体ではなく、オペレーティング・システム上の外部ファイル・ロケータが入ります。このロケータによって、ディレクトリの別名とファイル名が保持されます。

すべてのディレクトリは 1 つの名前領域に作成されるので、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り、そのディレクトリ構造内に格納されている BFILE へのアクセスのセキュリティを守ることができます。ディレクトリを作成すると、READ オブジェクト権限が自動的に付与され、他のユーザーやロールに 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。

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・システム上でディレクトリ用に定義されたアクセス権限とは無関係に作成されるため、これら 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 READ 権限を付与されているが、それに対応するディレクトリへのアクセス権限がオペレーティング・システム上で Oracle プロセスに付与されていない場合には、エラーが発生します。

**例** 次の文では、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、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';
```

## 関連項目

DROP DIRECTORY (4-384 ページ)

GRANT( システム権限とロール ) (4-432 ページ)

「ラージ・オブジェクト (LOB) データ型」 (2-14 ページ)

## CREATE FUNCTION

---

### 用途

ストアド・ファンクションを作成します。また、外部関数を登録します。

ストアド・ファンクション（つまり、ユーザー関数）とは、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、関数は、コールした環境に値を戻す点で異なります。ユーザー関数は、SQL 式の一部として使用できます。プロシージャと関数の一般的な説明は、**CREATE PROCEDURE**（4-255 ページ）を参照してください。関数作成の例については、「例」（4-231 ページ）を参照してください。

外部関数とは、SQL または PL/SQL からコールできる共有ライブラリに格納されている第三世代言語 (3GL) ルーチンです。外部関数をコールするには、その外部関数のある場所、コールの仕方、関数に渡す値について、PL/SQL 関数の中に情報を指定しなければなりません。

**CREATE FUNCTION** コマンドでは、スタンドアロンのスキーマ・オブジェクトとして関数が作成されます。また、関数をパッケージの一部としても作成できます。パッケージの作成の詳細は、**CREATE PACKAGE**（4-246 ページ）を参照してください。

外部関数の登録の詳細は、『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

ストアド・ファンクションを作成するためには、ユーザー **SYS** は SQL スクリプト **DBMSSTD.SQL** を実行しなければなりません。このスクリプトの実際の名前と格納位置は、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。

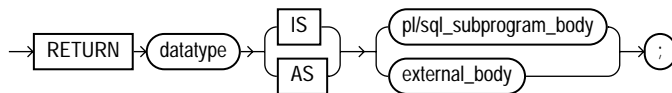
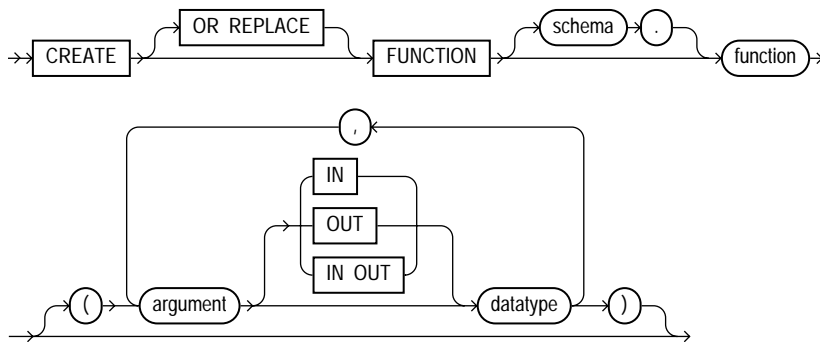
自スキーマに関数を作成する場合は、**CREATE PROCEDURE** システム権限が必要です。他のユーザーのスキーマ内で関数を作成するには、**CREATE ANY PROCEDURE** システム権限が必要です。

外部関数をコールする場合は、その関数が格納されているコールアウト・ライブラリに対する **EXECUTE** 権限が必要です。

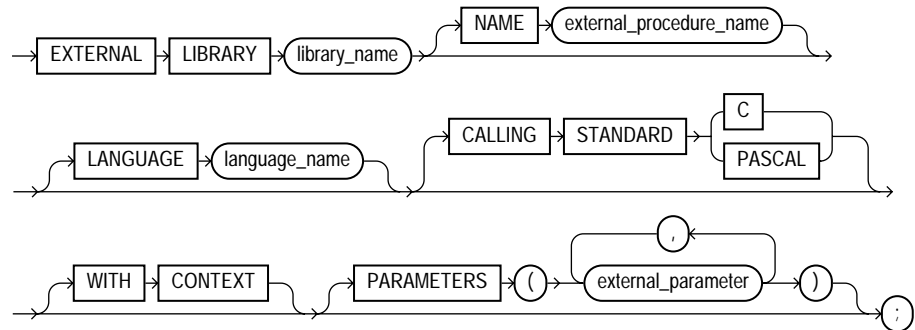
ストアド・ファンクションを作成するには、PL/SQL がインストールされている Oracle を使用しなければなりません。詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。

Oracle プリコンパイラ・プログラム内に **CREATE FUNCTION** 文を埋め込むには、キーワード **END-EXEC** に続けて、各言語の埋込み SQL 文終了記号で文を終了しなければなりません。

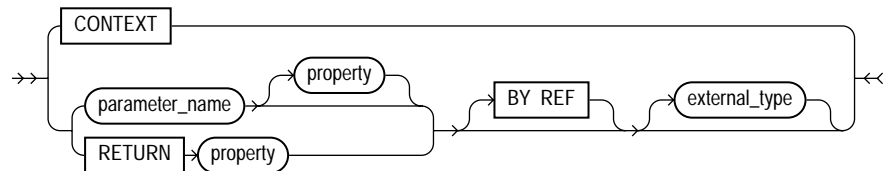
## 構文

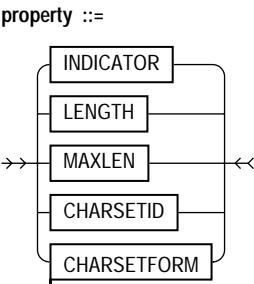


external\_body::=



external\_parameter ::=





キーワードとパラメータ

OR REPLACE	<p>関数がすでに定義されている場合には、関数を再作成します。このオプションを指定すると、既存の関数の定義の変更が、その関数に付与されているオブジェクト権限の削除、再作成、再付与なしに実行できます。関数を再定義すると、その関数は再コンパイルされます。関数の再コンパイルの詳細は、ALTER FUNCTION（4-26 ページ）を参照してください。</p> <p>再定義した関数に対する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その関数にアクセスできます。</p>
schema	<p>関数を定義するスキーマを指定します。schema を指定しないと、現行スキーマに関数が作成されます。</p>
function	<p>作成する関数の名前を指定します。</p>
argument	<p>関数への引数の名前を指定します。関数が引数を受け入れない場合は、関数名の後のカッコを省略できます。</p>
IN	<p>関数をコールするときに引数に値を指定しなければならないことを示します。これはデフォルト値です。</p>
OUT	<p>関数によって引数の値が設定されることを示します。</p>
IN OUT	<p>引数の値をユーザーが指定することも関数で設定することも可能であることを示します。</p>
datatype	<p>引数のデータ型を指定します。引数は、PL/SQL でサポートされるデータ型を持つことができます。</p> <p>データ型にはデータ長、精度、位取りは指定できません。Oracle では、その関数がコールされた環境から引数のデータ長、精度、位取りを導出します。</p>
RETURN datatype	<p>関数の戻り値のデータ型を指定します。すべての関数が必ず値を戻すので、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。</p> <p>データ型にはデータ長、精度、位取りは指定できません。Oracle では、その関数がコールされた環境から戻り値のデータ長、精度、位取りを導出します。PL/SQL データ型の詳細は、『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。</p>

<code>pl/sql_subprogram_body</code>	関数の定義を示します。関数の定義は、PL/SQL で作成します。PL/SQL の詳細は、『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。
<code>external_body_clause</code>	登録する外部関数を指定します。
<code>AS EXTERNAL</code>	共有可能なライブラリに格納されている外部 3GL 関数を指定します。この <code>AS EXTERNAL</code> 句は、PL/SQL と外部関数間のインタフェースとなります。
<code>LIBRARY</code>	外部関数が格納されている共有ライブラリを指定します。そのライブラリに対する EXECUTE 権限が必要です。構文については、 <code>CREATE LIBRARY</code> (4-244 ページ) を参照してください。
<code>library_name</code>	PL/SQL の識別子を指定します。 <code>library_name</code> を二重引用符で囲むと、大 / 小文字が区別されますが、引用符は必須ではありません。
<code>NAME external_function_name</code>	コールする外部関数の名前を指定します。 <code>external_function_name</code> を二重引用符で囲むと、大 / 小文字が区別されますが、引用符は必須ではありません。この名前を指定しないと、デフォルトでは PL/SQL サブプログラム (大文字) 名が設定されます。
<code>LANGUAGE</code>	外部関数を作成した 3GL を指定します。現在サポートされている言語名は C だけです。この名前を指定しないと、デフォルトで C が設定されます。
<code>CALLING STANDARD</code>	外部関数のコンパイルに使用したコール標準 (C または Pascal) を指定します。コール標準を指定しないと、デフォルトで C が設定されます。
<code>WITH CONTEXT</code>	外部関数に渡される最初のパラメータがコンテキスト・ポインタとなるように指定します。このコンテキストの意味は外部関数からは不透明ですが、外部関数によってコールされる各関数にアクセスするときに使用できます。 <code>WITH CONTEXT</code> 句の詳細は、『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。
<code>PARAMETERS</code>	外部関数に渡される各パラメータの位置とデータ型を指定します。また、現行のデータ長や最大長などのパラメータ・プロパティ、優先されるパラメータ受渡し方法 (値または参照による) も指定できます。パラメータ受渡しの詳細は、『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

例 1. 次の文は、関数 GET\_BAL を作成する例です。

```
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 関数を実行すると、指定された預金口座の残高が戻ります。

この関数をコールするときは、残高を確認する口座の番号 ACC\_NO を引数として指定しなければなりません。ACC\_NO のデータ型は NUMBER です。

この関数では、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。

この関数では 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、関数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関数は、SQL 文の中で使用できます。たとえば、次のように指定できます。

```
SELECT get_bal(100) FROM DUAL;
```

**例 2.** 次の文では、C ルーチン C\_GET\_VAL を外部関数として登録する PL/SQL スタンドアロン関数 GET\_VAL が作成されます。

```
CREATE FUNCTION get_val  
( x_val IN BINARY_INTEGER,  
  y_val IN BINARY_INTEGER,  
  image IN LONG RAW )  
RETURN BINARY_INTEGER AS EXTERNAL LIBRARY c_utils  
NAME "c_get_val"  
LANGUAGE C;
```

## 関連項目

ALTER FUNCTION (4-26 ページ)

CREATE LIBRARY (4-244 ページ)

CREATE PACKAGE (4-246 ページ)

CREATE PACKAGE BODY (4-250 ページ)

CREATE PROCEDURE (4-255 ページ)

DROP FUNCTION (4-385 ページ)

『PL/SQL ユーザーズ・ガイドおよびリファレンス』

---

## CREATE INDEX

### 用途

次のものについて索引を作成します。

- 表の 1 つ以上の列またはパーティション表、クラスタ
- **OBJ** 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
- **OBJ** ネストした表の列の索引を作成するためのネスト表の格納表

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。索引を使用すると、行に直接、高速にアクセスできます。パーティション索引は、表の索引付き列の中に表示される各値のエントリが入るパーティションから構成されます。「使用上の注意」（4-238 ページ）を参照してください。

---

**注意：** コマンドと句の説明で**OBJ**の印が前に付いている箇所は、Oracle Object Option がデータベース・サーバーにインストールされている場合にだけ有効です。

---

### 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当てはまらなければなりません。

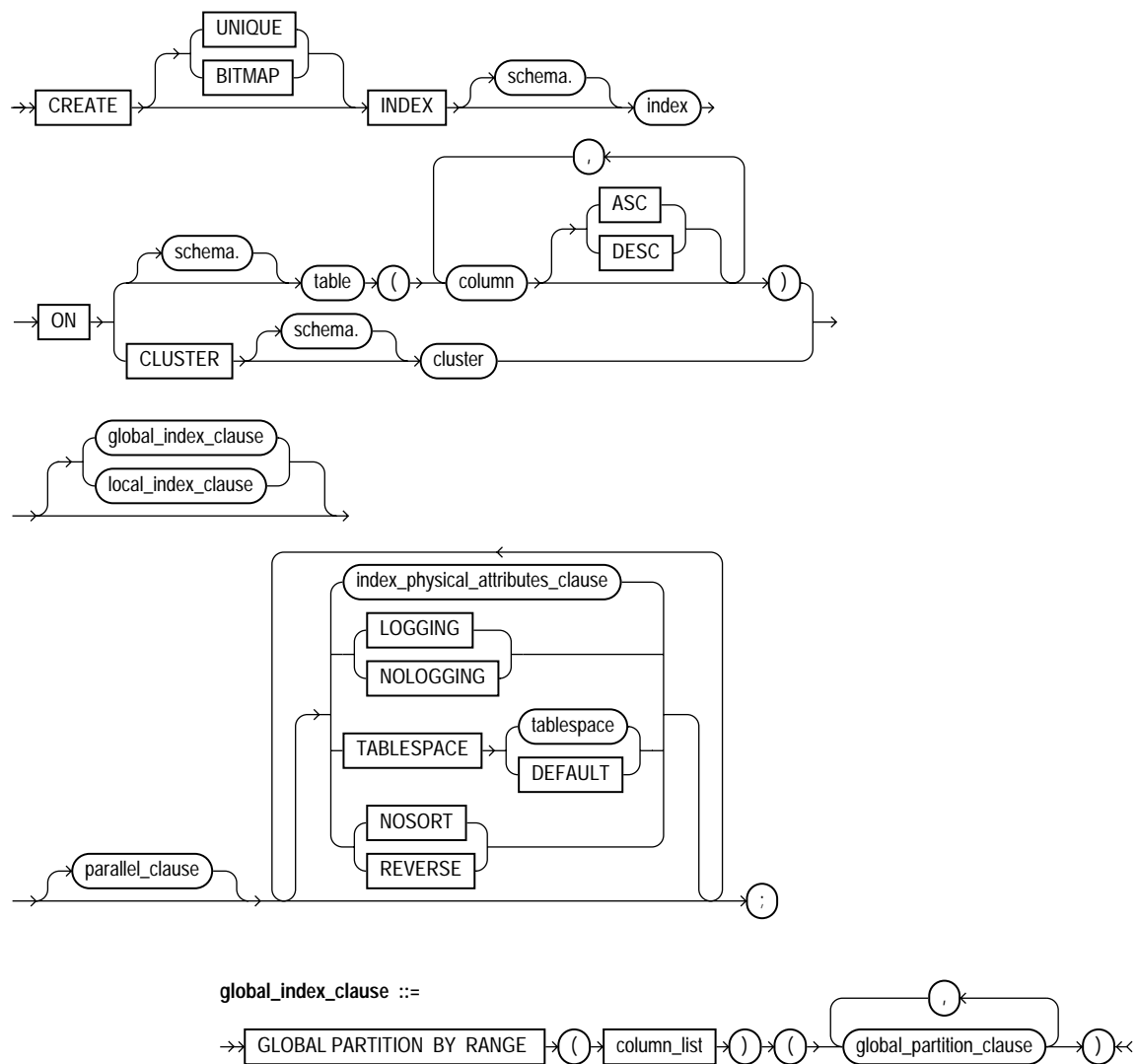
- 索引を作成する表またはクラスタが自スキーマ内に定義されている。
- 索引を作成する表に対する INDEX 権限がある。
- CREATE ANY INDEX システム権限がある。

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。

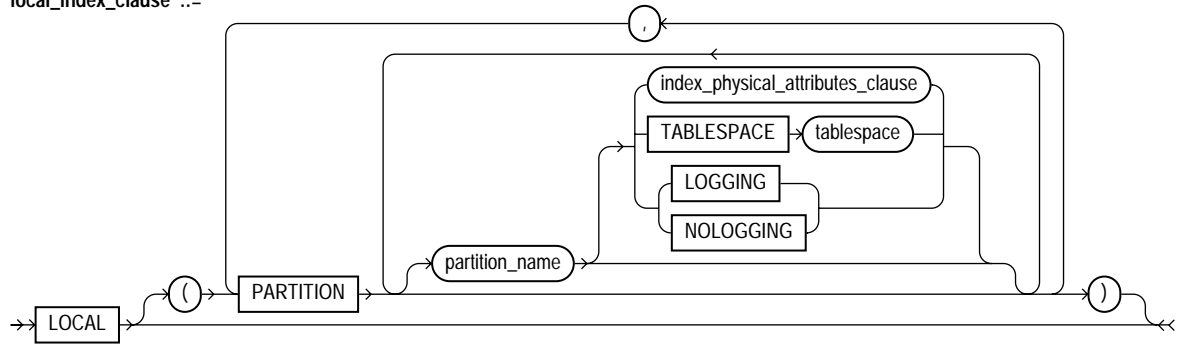
「索引の列」（4-239 ページ）も参照してください。

## 構文

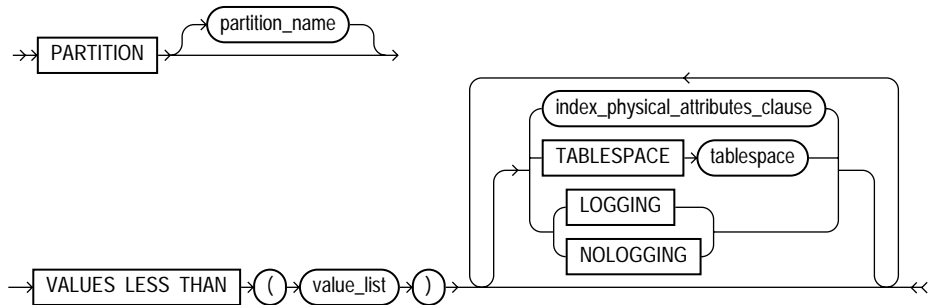




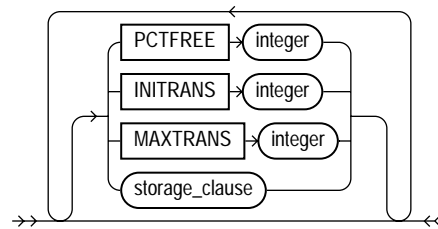
local\_index\_clause ::=



global\_partition\_clause ::=



index\_physical\_attributes\_clause ::=



parallel\_clause: `PARALLEL` 句 (4-462 ページ) を参照。

storage\_clause: `STORAGE` 句 (4-518 ページ) を参照。

## キーワードとパラメータ

UNIQUE	<p>索引を作成する表の列（または列の組合せ）の値が一意でなければならないことを指定します。</p> <p>索引がローカル非同一キー索引（後述の LOCAL 句の項を参照）である場合には、索引キーにパーティション化キーが定義されていなければなりません。</p>
BITMAP	<p>指定する <i>index</i> を、B* ツリーとして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。「ビットマップ索引の作成」（4-242 ページ）も参照してください。</p> <p><b>注意：</b> グローバル・パーティション索引を作成する場合には、このキーワードは使用できません。</p>
UNIQUE か BITMAP のいずれかを指定できますが、一意のビットマップ索引は作成できません。	
<i>schema</i>	<p>索引を定義するスキーマを指定します。<i>schema</i> を指定しないと、自スキーマ内に索引が作成されます。</p>
<i>index</i>	<p>作成する索引の名前を指定します。（「単一の表に対する複数の索引」（4-239 ページ）も参照してください。）<i>index</i> は、いくつかのパーティションを持つことができます。</p> <p>クラスタ索引またはクラスタ化表に定義された索引はレンジ・パーティションに分割できません。</p>
<i>table</i>	<p>索引を作成する名前を指定します。指定する <i>table</i> を <i>schema</i> 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。</p> <p>索引が LOCAL である場合には、指定する <i>table</i> をパーティションで分割する必要があります。</p> <p>索引構成表には索引を作成できません。</p> <p><b>OBJ</b> ネストした表の格納表には、索引を作成できます。</p>
<i>column</i>	<p>表内の列の名前を指定します。索引には最大 32 列まで指定できます。索引の列には、データ型として LONG や LONG RAW は指定できません。「索引の列」（4-239 ページ）も参照してください。</p> <p><b>OBJ</b> スカラー・オブジェクト属性列またはネストした表の格納表のシステム定義の NESTED_ TABLE_ID 列には索引を作成できます。オブジェクト属性列を指定する場合、列名を表名で修飾する必要があります。ネストした表の列属性を指定する場合は、この属性は、一番外側の表の名前およびネストした表が定義されている列の名前、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しなければなりません。「ネストした表の列に対する索引の作成」（4-242 ページ）も参照してください。</p> <p>「NULL」（4-241 ページ）も参照してください。</p>
ASC / DESC	<p>DB2 の構文との互換性のために使用できます。ただし、索引は常に昇順で作成されます。文字データの索引は、データベースのキャラクタ・セットの文字値の昇順で作成されます。</p>
CLUSTER	<p>クラスタ索引を作成するクラスタを指定します。<i>cluster</i> を <i>schema</i> で修飾しないと、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。なお、ハッシュ・クラスタにはクラスタ索引を作成できません。「クラスタ索引の作成」（4-241 ページ）も参照してください。</p>

<i>index_physical_attributes_clause</i>	索引に対して、INITRANS、MAXTRANS、PCTFREE の各パラメータ値と記憶特性値を設定します。CREATE TABLE (4-303 ページ) を参照してください。
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。
<i>storage_clause</i>	索引の記憶特性を設定します。STORAGE 句 (4-518 ページ) を参照してください。
TABLESPACE	索引または索引パーティションを格納する表領域の名前を指定します。このオプションを指定しないと、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。  パーティション索引の場合は、この指定は表領域名となります。  ローカル索引の場合、表領域名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は、基礎となる表の対応するパーティションと同じ表領域内に作成されます。
NOSORT	データベース内に行が昇順で格納されることを指定します。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。REVERSE をこのオプションとともに指定することはできません。「NOSORT オプション」(4-240 ページ) も参照してください。
REVERSE	ROWID 以外の索引ブロックのバイトを逆順に格納します。NOSORT をこのオプションとともに指定することはできません。  ビットマップ索引は逆順にできません。
LOGGING / NOLOGGING	索引の作成を、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は、この指定は索引のロギング属性となります。  パーティション索引の場合は、指定するロギング属性は、索引の各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になります。デフォルトのロギング値は、PARTITION 記述句に LOGGING/NOLOGGING を指定しないかぎり、CREATE 文（およびその後の各 ALTER TABLE ADD PARTITION 文）で指定したすべてのパーティションに適用されません。  NOLOGGING モードでのデータ変更では、最低限のログ（新しいエクステントに無効のマークを付けることとディクショナリの変更を記録すること）しか記録されません。メディア回復中に NOLOGGING が適用されると、REDO データがログに記録されないために、エクステント無効化レコードはブロック範囲を論理的に破壊されているものとしてマーク設定します。このため、消失が許されない索引の場合は、NOLOGGING 操作の後にバックアップをとってください。  データベースを ARCHIVELOG モードで運用する場合、LOGGING 操作の前に行ったバックアップからのメディア回復によって、索引が再作成されます。ただし、NOLOGGING 操作の前に行ったバックアップからのメディア回復では、索引は再作成されません。  索引のロギング属性は、その実表の属性に依存しません。

	<p>[NO]LOGGING 句を指定しないと、デフォルトでは索引が定義されている表領域のロギング属性が索引のロギング属性となります。</p> <p>LOGGING オプションおよびパラレル DML の詳細は、「NOLOGGING」(4-240 ページ) および『Oracle8 Server 概要』、『Oracle8 Parallel Server 概要および管理』を参照してください。</p>
GLOBAL	<p>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します。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。</p>
PARTITION BY RANGE	<p>グローバル索引のパーティション化を <i>column_list</i> で指定した列の値の範囲で行うことを指定します。この句は、LOCAL 索引には指定できません。</p>
( <i>column_list</i> )	<p>索引のパーティション化を行う表の列名を指定します。<i>column_list</i> では、索引の列リストの左の接頭辞を指定しなければなりません。</p> <p><i>column_list</i> に指定できるのは、最大 32 列までで、ROWID 疑似列や ROWID 型の列は指定できません。</p>
LOCAL	<p><i>table</i> 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、索引をレンジ・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ると、LOCAL 索引のパーティションを自動的にメンテナンスします。</p>
PARTITION <i>partition_name</i>	<p>個々のパーティションについて記述します。句の数によってパーティションの数が決まります。索引が LOCAL の場合、索引のパーティションの数は表のパーティション数と等しく、表のパーティションと同じ順序でなければなりません。</p> <p><i>partition_name</i> は、物理索引パーティションの名前です。<i>partition_name</i> を指定しないと、名前は SYS_Pn の形で生成されます。</p> <p>ローカル索引の場合は、<i>partition_name</i> を指定しないと、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は、SYS_Pn の形が使われます。</p> <p>「パーティション索引の作成」(4-241 ページ) も参照してください。</p>
VALUES LESS THAN ( <i>value_list</i> )	<p>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(上限値を含まない)を指定します。<i>value_list</i> は、PARTITION BY RANGE 句の <i>column_list</i> と対応するリテラル値をコンマで区切って順にならべたリストです。最後のパーティションの <i>value_list</i> には必ず MAXVALUE を指定してください。</p> <p>ローカル索引にはこの句を指定できません。</p>
<i>parallel_clause</i>	<p>索引作成時の並列度を指定します。PARALLEL 句(4-462 ページ)を参照してください。</p>

使用上の注意

索引とは、1 つ以上の列のグループに属する値すべてを任意の時点で順に並べたリストのことです。このリストを使用することにより、列の値を調べるような問合せを非常に効率的に実行できます。ただし、索引自体もデータ記憶領域を占有し、データが変更されれば索引の変更も必要になります。したがって、ケースごとにコストと効果の分析を行って、索引を使

用すべきかどうかと、その使用方法を判断する必要があります。次のような場合には、索引を使用することにより、パフォーマンスが上がります。

- 索引の列値を指定して、行を検索する場合
- 索引の列の順序で表にアクセスする場合

新しい表に最初から行を挿入するときは、一般に、表を作成し、行を挿入してから索引を作成するほうが、作業を速く行うことができます。これは、索引を作成してから行を挿入すると、行を挿入するたびに索引が更新されるからです。

表には明示的に UNIQUE 索引を定義しないようにしてください。一意性は非常に論理的な概念であり、表の定義と関連付けるべきものです。かわりに、目的の列に UNIQUE 整合性制約を定義してください。Oracle では、一意キーに自動的に一意の索引を定義すると、UNIQUE 整合性制約が強制的に定義されます。ここでお勧めする事柄の例外は、一般にパフォーマンスに関連するものです。たとえば、UNIQUE 制約を指定して CREATE TABLE ... AS SELECT を実行すると、制約なしで表を作成してから手動で UNIQUE 索引を作成する場合と比べて、非常に時間がかかります。

索引に NULL が含まれている場合は、その NULL は通常、固有の値とみなされます。ただし、例外が 1 つあります。索引内の複数の行にあるすべての非 NULL 値が同一である場合、それらの行は同一とみなされてしまいます。この問題の発生を防ぐには、UNIQUE 索引を使用します。ただし、非 NULL 値がない場合、つまり、行全体が NULL である場合には、NULL は固有の値とはみなされません。

---

---

**注意：** ユーザー定義型の列や属性、あるいは LOB 型または REF 型の列や属性には索引は作成できません。ただし、例外として、SCOPE 句で定義されている REF 型の列や属性に対する索引作成はサポートされます。

---

---

## 索引の列

索引には最大 32 列まで指定できます。索引エントリは、それぞれの列のデータ値がすべて連結したものです。索引の列は、任意の順序で指定できます。ただし、このとき指定した列の順序によって、索引がどのように使用されるかが決まります。

状況に応じて、索引全体または索引の先頭部分が使用されます。たとえば、IDX1 という名前の索引を TAB1 表の列 A、B、C (順序は A、B、C) に対して作成したとします。この場合 Oracle は、列 A、B、C に対する参照 (索引全体の参照)、列 A と B に対する参照、または列 A だけに対する参照というように索引を使用します。ただし、列 B と C に対する参照では IDX1 索引は使用しません。この場合、列 B と C に対して別の索引を作成できます。

## 単一の表に対する複数の索引

列の組合せが索引ごとに異なっている限り、1 つの表に対して作成できる索引の数に制限はありません。組合せを変えれば、同一の列を使って複数の索引を作成できます。たとえば、次の文は有効な組合せの指定です。

```
CREATE INDEX emp_idx1 ON emp (ename, job);
CREATE INDEX emp_idx2 ON emp (job, ename);
```

表の1つの列を参照する索引がすでに定義されている場合は、同じ索引を別に作成することはできません。

索引の付いたデータに対する更新処理中は、各索引によって表を管理するための処理時間が増大してしまいます。このため、5つの索引がある表を更新するよりも、1つしか索引がない表を更新するほうが短時間で済みます。

## NOSORT オプション

NOSORT オプションを使うと、索引作成に必要な時間を大幅に減らすことができます。通常の索引の作成の場合、最初に索引の列に基づいて表の行をソートした上で、索引を作成します。ソート操作は、関連する作業全体から見て大きな割合を占めることがあります。行が物理的に（索引の列値に基づいて）昇順で格納されている場合は、NOSORT オプションを指定すると、そのプロセスのソート・フェーズが省略されます。

クラスタ索引、パーティション索引、ビットマップ索引の作成では、このNOSORT オプションは使用できません。

NOSORT オプションを指定すると、索引を作成するために必要な領域を削減できます。Oracle はソート処理中には一時セグメントを使用します。したがって、ソート処理を実行しないため、その分だけ少ない一時領域で索引を作成できます。

NOSORT オプションを使用するには、行は物理的に昇順で格納されていなければなりません。しかし、NOSORT オプションを使用してもリスクはありません。行が昇順でなければ、エラーが戻ります。その場合には、NOSORT オプションを指定しないでCREATE INDEX コマンドを再度発行できます。リレーショナル・データベース管理システム（特に Oracle のような）が本来備えている物理的なデータ独立性のため、表に対して物理的な内部順序を強制する方法はありません。NOSORT オプションを指定したCREATE INDEX コマンドは、行を表に初期ロードした後すぐに使用するようにしてください。

## NOLOGGING

NOLOGGING オプションを使うと、大規模な索引の作成に必要な時間を大幅に短縮できる場合があります。この機能は、大規模な索引をパラレルで作成した後には特に有効です。バックアップと回復については、『Oracle8 Server バックアップおよびリカバリ』および『Oracle8 Server 管理者ガイド』を参照してください。

**例.** 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み）に、パラレルで索引をすばやく作成するには、次の文を発行します。

```
CREATE INDEX i_loc
ON big_table (akey)
NOSORT
NOLOGGING
PARALLEL (DEGREE 5);
```

## NULL

ビットマップ索引の場合を除いて、すべてのキー列が NULL となっている表の行には索引は付きません。

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。

## クラスタ索引の作成

Oracle では、クラスタを最初に作成するとき、クラスタに対して自動的に索引が作成されません。このため、クラスタ索引を作成するまでは、クラスタ表に対しては、データ操作言語文は発行できません。

例 . EMPLOYEE クラスタに対して索引を作成するには、次のコマンドを発行します。

```
CREATE INDEX ic_emp
ON CLUSTER employee
```

なお、クラスタ・はキーの列すべて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。

## パーティション索引の作成

索引は、ローカル同一キー索引（一意の場合も一意でない場合もある）、またはローカル非同一キー索引（パーティション化キーが索引キーのサブセットである場合にだけ一意。一意でない場合もある）、グローバル同一キー索引（一意の場合も一意でない場合もある）のいずれかです。グローバル非同一キー索引はサポートされません。ローカル索引は常にパーティション化されます。グローバル索引は、パーティション化することもしないこともできます。

索引のパーティションは、順序どおりに設定する必要があります。つまり、グローバル索引の場合では、リストの最初のパーティションのパーティション・バウンドが、リストの 2 番目のパーティションのパーティション・バウンドより小さく、リストの 2 番目のパーティションのパーティション・バウンドが、3 番目より小さいというように指定する必要があります。ローカル索引の場合のパーティションは、対応する基礎となる表のパーティションと同一順序で設定しなければなりません。

例 . 次の文では、表 STOCK\_XACTIONS にグローバル同一キー索引 STOCK\_IX が、2 つのパーティションに分割されて作成されます。各パーティションがアルファベットの半分ずつを受け持ちます。索引パーティション名はシステムによって生成されます。

```
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);
```

## ビットマップ索引の作成

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、データ格納など、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も適しています。ビットマップ索引の詳細は、『Oracle8 Server 概要』および『Oracle8 Server チューニング』を参照してください。

**例** . 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する場合、次の文を発行します。

```
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);
```

ビットマップ索引、一意ビットマップ索引、グローバル・パーティション索引は作成できません。

## ● ネストした表の列に対する索引の作成

ネストした表列を持つ表を作成すると、ネストした表の列ごとの格納表が暗黙に作成されます。この格納表には、ネストした表の値を持つ行、および各行に割り当てられたネストした表の識別子の値が格納されます。この識別子の値は、NESTED\_TABLE\_ID と呼ばれる格納表の疑似列に入ります。

ネストした表の格納表に索引を作成することにより、ネストした表の列に索引を作成します。NESTED\_TABLE\_ID 疑似列を組み込んだ UNIQUE 索引を作成するのは、ネストした表の値を持つ行がそれぞれ異なることを確実にする有効な手段です。

**例** . 次の例では、UNIQUE 索引 UNIQ\_PROJ\_INDX が、格納表 NESTED\_PROJECT\_TABLE について作成されます。NESTED\_TABLE\_ID を組み込むことにより、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_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);
```

## 関連項目

ALTER INDEX (4-28 ページ)

CREATE TABLE (4-303 ページ)

「索引構成表」 (4-123 ページ)

DROP INDEX (4-387 ページ)

CONSTRAINT 句 (4-185 ページ)

STORAGE 句 (4-518 ページ)

# CREATE LIBRARY

## 用途

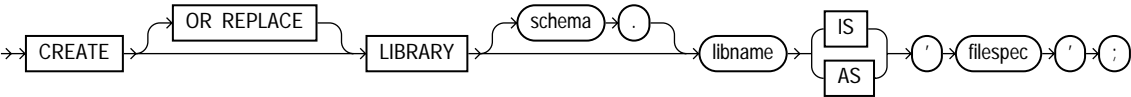
SQL および PL/SQL では、オペレーティング・システム共有ライブラリから外部第 3 世代言語 (3GL) 関数およびプロシージャをコールできますが、このオペレーティング・システム共有ライブラリを表すスキーマ・オブジェクト（ライブラリ）を作成する場合に **CREATE LIBRARY** を使用します。「例」（4-245 ページ）を参照してください。

## 前提条件

**CREATE ANY LIBRARY** システム権限が必要です。このライブラリに格納されているプロシージャおよび関数を使うには、このライブラリについての **EXECUTE** オブジェクト権限が必要です。

**CREATE LIBRARY** コマンドが有効なのは、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だけです。

## 構文



filespec: 「Filespec」（4-428 ページ）を参照。

## キーワードとパラメータ

OR REPLACE	既存のライブラリがある場合には、そのライブラリを再作成します。このオプションを指定すると、既存のライブラリの定義の変更を、そのライブラリに付与されているスキーマ・オブジェクト権限の削除、再作成、再付与なしに行うことができます。  再定義したライブラリに対する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そのライブラリにアクセスできます。
libname	SQL および PL/SQL で外部 3GL 関数およびプロシージャをコールするライブラリ（スキーマ・オブジェクト）の名前を指定します。
'filespec'	引用符 (') で囲んだ非ゼロ長文字列を指定します。'filespec' は、PL/SQL では解釈されません。  'filespec' 内で指定するディレクトリおよびファイル名は、PL/SQL では解釈されないため、その指定の存在はプロシージャの実行時までチェックされません。

## 例

例 1. 次の文は、ライブラリ EXT\_LIB を作成する例です。

```
CREATE LIBRARY ext_lib AS '/OR/lib/ext_lib.so';
```

例 2. 次の文は、ライブラリ EXT\_LIB を再作成する例です。

```
CREATE OR REPLACE ext_lib IS '/OR/newlib/ext_lib.so';
```

## 関連項目

CREATE FUNCTION (4-228 ページ)

CREATE PROCEDURE (4-255 ページ)

『PL/SQL ユーザーズ・ガイドおよびリファレンス』

# CREATE PACKAGE

## 用途

ストアド・パッケージの仕様を作成します。パッケージとは、関連のあるプロシージャや関数などのプログラム・オブジェクトの集合のことで、データベース上にまとめて格納されます。仕様部では、これらのオブジェクトを宣言します。

## 前提条件

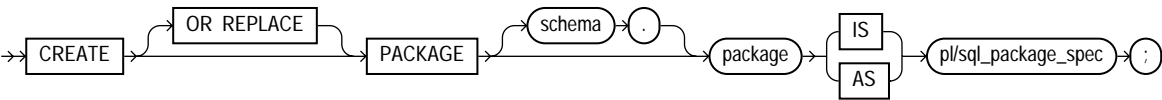
パッケージを作成するためには、ユーザー **SYS** は SQL スクリプト **DBMSSTD.SQL** を実行しなければなりません。このスクリプトの実際の名前と格納位置は、使用するオペレーティング・システムによって異なります。

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には、**CREATE ANY PROCEDURE** システム権限が必要です。

Oracle プリコンパイラ・プログラムの中に **CREATE PACKAGE** 文を埋め込むには、キーワード **END-EXEC** とその後に特定の言語用の埋込み SQL 文終了記号を記述して文を終了しなければなりません。

詳細は、『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

<b>OR REPLACE</b>	パッケージ仕様部がすでに定義されている場合に、パッケージ仕様部を再作成します。このオプションを指定すると、既存のパッケージの仕様部の変更を、そのパッケージに対して付与されていたオブジェクト権限の削除および再作成、再付与なしに行うことができます。パッケージ仕様部を変更すると、その仕様部は自動的に再コンパイルされます。パッケージ仕様部の再コンパイルの詳細は、 <b>ALTER PROCEDURE</b> (4-41 ページ) を参照してください。
	再定義したパッケージに対する権限を付与されていたユーザーは、その権限を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。
<i>schema</i>	パッケージを定義するスキーマを指定します。 <i>schema</i> を指定しないと、自スキーマ内に表が作成されます。

---

<i>package</i>	作成するパッケージの名前を指定します。「使用上の注意」(4-247 ページ)を参照してください。
<i>pl/sql_package_spec</i>	パッケージ仕様部を示します。パッケージ仕様部で、プログラム・オブジェクトを宣言します。パッケージ仕様部は PL/SQL で記述します。パッケージ仕様部の記述方法など、PL/SQL の詳細は、『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。

---

## 使用上の注意

パッケージとは、関連のあるプログラム・オブジェクトの集合で、データベース上にまとめて格納されます。パッケージを構成するプログラム・オブジェクトには、プロシージャ、関数、変数、定数、カーソル、例外があります。

一連のプロシージャや関数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、別の方法としてパッケージを使用します。パッケージを使用すると、スタンドアロンのプロシージャや関数を使用する場合よりも、次のような多くの利点があります。

- アプリケーション開発をより効率的に推進できる。
- 権限の付与が効率化する。
- パッケージ・オブジェクトの変更時に、依存オブジェクトの再コンパイルが必要ない。
- Oracle が一度に複数のパッケージ・オブジェクトをメモリーに読み取ることができる。
- すべてのプロシージャや関数から使用できるグローバル変数やカーソルを、パッケージの中に指定できる。
- プロシージャや関数を多重定義できる。プロシージャの多重定義とは、同一パッケージ内に、同一名であるが、引数の数値やデータ型が異なるプロシージャを複数作成することです。

パッケージの利点の詳細は、『Oracle8 Server アプリケーション開発者ガイド』の章を参照してください。

## パッケージの作成方法

パッケージを作成するには、次の 2 つの手順を実行してください。

1. **CREATE PACKAGE コマンドでパッケージ仕様部を作成します。**パッケージ仕様部にプログラム・オブジェクトを宣言できます。このようなオブジェクトをパブリック・オブジェクトと呼びます。パブリック・オブジェクトは、パッケージ中の他のオブジェクトからもパッケージの外からも参照できます。
2. **CREATE PACKAGE BODY コマンドで、パッケージ本体を作成します。**このパッケージ本体でプログラム・オブジェクトの宣言および定義ができます。
  - パッケージ仕様部で宣言したパブリック・オブジェクトを定義する必要がある。
  - 追加のパッケージ・オブジェクトを宣言および定義することもできる。このようなオブジェクトをプライベート・オブジェクトと呼びます。プライベート・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するため、パッケージ内の他のオブジェクトからしか参照できません。したがって、このようなオブジェクトはパッケージの外からは参照できません。

CREATE PACKAGE BODY (4-250 ページ) を参照してください。

## 仕様部と本体の分離

Oracle では、パッケージの仕様部と本体はデータベース上に別々に格納されます。パブリック・プログラム・オブジェクトをコールまたは参照する他のスキーマ・オブジェクトはパッケージ仕様部だけに依存し、パッケージ本体には依存しません。このように分離することによって、パッケージ本体にあるプログラム・オブジェクトの定義を変更したときに、Oracle ではプログラム・オブジェクトをコールしたり参照したりする他のスキーマ・オブジェクトが無効になりません。パッケージ仕様部のプログラム・オブジェクトの宣言を変更した場合にだけ、依存スキーマ・オブジェクトが無効になります。

例. 次の SQL 文は、EMP\_MGMT というパッケージの仕様部の作成例です。

```
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 パッケージの仕様部では、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 と CREATE\_DEPT の各関数
- REMOVE\_EMP、REMOVE\_DEPT、INCREASE\_SAL、INCREASE\_COMM の各プロシージャ
- NO\_COMM と NO\_SAL の各例外

上記のすべてのオブジェクトは、このパッケージに対してアクセス権の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・プロシージャや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や関数をコールするためには、パッケージ本体でこれらのプロシージャや関数を定義しておかなければなりません。EMP\_MGMT パッケージの本体を作

成する CREATE PACKAGE BODY 文の例は、CREATE PACKAGE BODY (4-250 ページ) を参照してください。

## 関連項目

ALTER PACKAGE (4-38 ページ)

CREATE FUNCTION (4-228 ページ)

CREATE PROCEDURE (4-255 ページ)

CREATE PACKAGE BODY (4-250 ページ)

DROP PACKAGE (4-389 ページ)

## CREATE PACKAGE BODY

---

### 用途

ストアド・パッケージの本体を作成します。パッケージとは、関連のあるプロシージャやストアド・ファンクションなどのオブジェクトの集合のことで、データベース上にまとめて格納されます。本体で、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関数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します。パッケージの作成方法などパッケージの説明は、**CREATE PACKAGE** (4-246 ページ) を参照してください。使用例は、「例」(4-251 ページ) を参照してください。

### 前提条件

パッケージを作成するためには、ユーザー **SYS** は SQL スクリプト **DBMSSTD.SQL** を実行しなければなりません。このスクリプトの実際の名前と格納位置は、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。

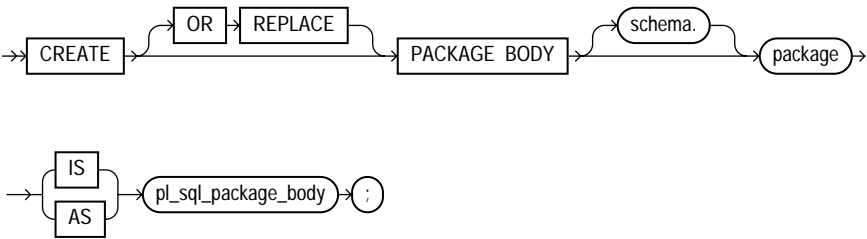
自スキーマ内にパッケージを作成する場合は、**CREATE PROCEDURE** システム権限が必要です。他のユーザーのスキーマ内にパッケージを作成するには、**CREATE ANY PROCEDURE** システム権限が必要です。

Oracle プリコンパイラ・プログラムの中に **CREATE PACKAGE BODY** 文を埋め込むには、キーワード **END-EXEC** とその後に特定の言語用の埋込み SQL 文終了記号を記述して文を終了しなければなりません。

詳細は、『**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	パッケージ本体がすでに定義されている場合に、それを再作成します。このオプションを指定すると、既存のパッケージの本体の変更を、そのパッケージに対して付与されていたオブジェクト権限の削除および再作成、再付与なしに実行できます。パッケージ本体を変更すると、その本体は自動的に再コンパイルされます。パッケージ本体の再コンパイルの詳細は、ALTER PACKAGE（4-38 ページ）を参照してください。  再定義したパッケージに対する権限を付与されていたユーザーは、その権限を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。
schema	パッケージを定義す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	パッケージ本体を示します。パッケージの本体で、プログラム・オブジェクトを宣言および定義します。パッケージ本体は PL/SQL で記述します。パッケージ本体の記述方法など、PL/SQL についての説明は、『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

例 . 次の SQL 文は、EMP\_MGMT というパッケージ本体の作成例です。

```
CREATE PACKAGE BODY emp_mgmt AS
    tot_ems  NUMBER;
    tot_depts NUMBER;

FUNCTION hire
    (ename VARCHAR2,
     job  VARCHAR2,
     mgr  NUMBER,
     sal  NUMBER,
     comm NUMBER,
     deptno NUMBER)
```

## CREATE PACKAGE BODY

---

```
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_ems := tot_ems + 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_ems := tot_ems - 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_ems
        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** コマンド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** と **CREATE\_DEPT** の各関数
- **REMOVE\_EMP**、**REMOVE\_DEPT**、**INCREASE\_SAL**、**INCREASE\_COMM** の各プロシージャ

上記のすべてのオブジェクトは、パッケージ仕様部で宣言しているため、パッケージ外のアプリケーション・プログラムおよびプロシージャ、関数からコールできます。たとえば、そのパッケージに対するアクセス権がある場合には、**INCREASE\_COMM** プロシージャをコールする **EMP\_MGMT** パッケージから分離させて、**INCREASE\_ALL\_COMMS** プロシージャを作成できます。

上記のオブジェクトはパッケージ本体で定義しているため、その定義を変更しても 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** と **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、関数 **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。

## 関連項目

**ALTER PACKAGE** (4-38 ページ)

**CREATE FUNCTION** (4-228 ページ)

**CREATE PROCEDURE** (4-255 ページ)

**CREATE PACKAGE** (4-246 ページ)

**DROP PACKAGE** (4-389 ページ)

## CREATE PROCEDURE

---

### 用途

スタンドアロンのストアド・プロシージャを作成します。また、外部プロシージャを登録します。プロシージャとは、名前でもコールできる PL/SQL 文の集合です。外部プロシージャとは、SQL または PL/SQL からコールできる共有ライブラリに格納されている第三世代言語 (3GL) ルーチンです。外部プロシージャをコールするには、その外部プロシージャの場所、およびそのコールの仕方、そのプロシージャに渡す値について、PL/SQL 関数に情報を指定しなければなりません。「使用上の注意」(4-258 ページ) を参照してください。

外部プロシージャの登録の詳細は、『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。

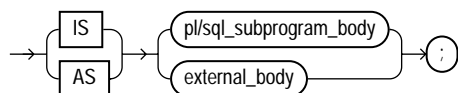
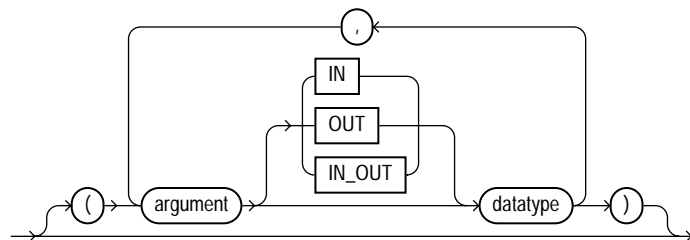
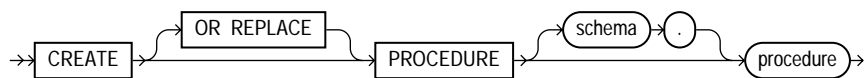
### 前提条件

プロシージャを作成するためには、ユーザー SYS が SQL スクリプト DBMSSTDY.SQL を実行しなければなりません。このスクリプトの実際の名前と格納位置は、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。

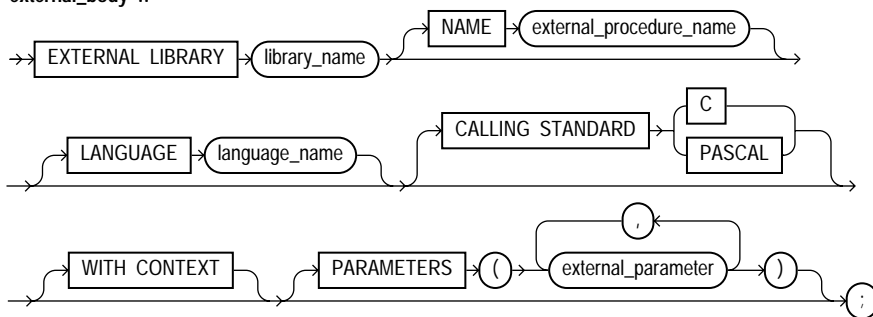
自スキーマ内にプロシージャを作成する場合は、CREATE PROCEDURE システム権限が必要です。他のユーザーのスキーマ内にプロシージャを作成する場合は、CREATE ANY PROCEDURE システム権限が必要です。他のスキーマ内のプロシージャを再作成する場合は、ALTER ANY PROCEDURE システム権限が必要です。

外部プロシージャをコールする場合は、そのプロシージャが定義されているコールアウト・ライブラリに対する EXECUTE 権限が必要です。

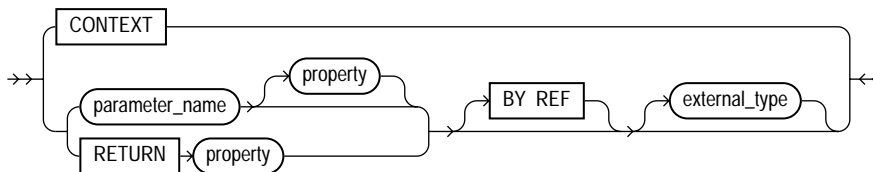
## 構文

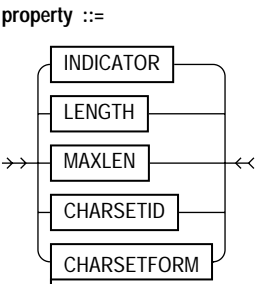


**external\_body ::=**



**external\_parameter ::=**





## キーワードとパラメータ

OR REPLACE	<p>プロシージャがすでに定義されている場合に、プロシージャを再作成します。このオプションを指定すると、既存のプロシージャの定義の変更を、その関数に付与されているオブジェクト権限の削除、再作成、再付与なしに実行できます。プロシージャを再定義すると、そのプロシージャは自動的に再コンパイルされます。プロシージャの再コンパイルの詳細は、ALTER PROCEDURE (4-41 ページ) を参照してください。</p> <p>再定義したプロシージャに対する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そのプロシージャにアクセスできます。</p>
schema	<p>プロシージャを定義するスキーマを指定します。<i>schema</i> を指定しないと、自スキーマ内にプロシージャが作成されます。</p>
procedure	<p>作成するプロシージャの名前を指定します。</p>
argument	<p>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には、プロシージャ名の後のカッコを省略できます。</p>
IN	<p>プロシージャをコールするときに引数に値を指定しなければならないことを示します。</p>
OUT	<p>プロシージャが実行後にコール側の環境に対して、指定した引数の値を戻すことを指定します。</p>
IN OUT	<p>プロシージャのコール時に引数の値を必ず指定することと、そのプロシージャが実行後にコール側環境に値を渡すことを指定します。</p>
datatype	<p>IN および OUT、IN OUT のいずれも指定しないと、デフォルトでは IN が設定されます。</p> <p>引数のデータ型を指定します。データ長の指定がない限り、引数は PL/SQL でサポートされるすべてのデータ型を持つことができます。PL/SQL のデータ型の詳細は、『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。</p> <p>データ型を指定するときは、データ長、精度、位取りは指定しません。たとえば、VARCHAR2(10) は無効ですが、VARCHAR2 は有効です。Oracle では、その関数のコール側の環境から引数のデータ長、精度、位取りを導出します。</p>

IS <i>pl/sql_subprogram_body</i>	プロシージャの定義を示します。プロシージャの定義は PL/SQL で作成します。PL/SQL のサブプログラム本体の記述方法など、PL/SQL の詳細は、『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。
AS <i>external_body</i>	共有可能なライブラリに格納されている外部 3GL プロシージャを指定します。この AS <i>external_body</i> 句は、PL/SQL と外部プロシージャ間のインタフェースとなります。
LIBRARY	外部プロシージャが格納されている共有ライブラリを指定します。そのライブラリに対する EXECUTE 権限が必要です。構文については、CREATE LIBRARY (4-244 ページ) を参照してください。
<i>library_name</i>	PL/SQL の識別子を指定します。 <i>library_name</i> を二重引用符で囲むと、大 / 小文字が区別されますが、引用符は必須ではありません。
NAME <i>external_procedure_name</i>	コール先外部プロシージャの名前を指定します。 <i>external_procedure_name</i> を二重引用符で囲むと、大 / 小文字が区別されますが、引用符は必須ではありません。この名前を省略すると、デフォルトでは PL/SQL サブプログラム (大文字) 名が設定されます。
LANGUAGE	外部プロシージャを作成した 3GL を指定します。現在サポートされている言語名は C だけです。この名前を指定しないと、デフォルトでは C が設定されます。
CALLING STANDARD	外部関数のコンパイルに使用したコール標準 (C または PASCAL) を指定します。コール標準を省略すると、デフォルトでは C が設定されます。
WITH CONTEXT	外部プロシージャに渡される最初のパラメータがコンテキスト・ポインタとなるように指定します。このコンテキストの意味は外部プロシージャからは不透明ですが、外部プロシージャによってコールされる各関数にアクセスするときに使用できます。WITH CONTEXT 句の詳細は、『PL/SQL ユーザー・ガイドおよびリファレンス』を参照してください。
PARAMETERS	その外部プロシージャに渡される各パラメータの位置とデータ型を指定します。また、現行のデータ長や最大長などのパラメータのプロパティや、優先されるパラメータ受渡し方法 (値によるまたは参照による) も指定できます。パラメータ受渡しの詳細は、『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 プリコンパイラ・プログラムの中に CREATE PROCEDURE 文を埋め込むには、キーワード END-EXEC とその後にそれぞれの言語用の埋込み SQL 文終了記号を記述して文を終了しなければなりません。

---

## 使用上の注意

プロシージャとは、名前によりコールできる PL/SQL 文の集合です。ストアード・プロシージャとストアード・ファンクションは、多くの点で似ています。ここで説明する事柄は、プロシージャだけでなく、関数にも当てはまります。関数に固有の情報は CREATE FUNCTION (4-228 ページ) を参照してください。

PL/SQL では、複数の SQL 文を、Ada や C などのプログラミング言語と似たプロシージャ型の PL/SQL 文にまとめることができます。CREATE PROCEDURE コマンドを使うと、プロシージャを作成し、そのプロシージャをデータベースに格納できます。SQL 文を発行できる環境であればどこからでも、ストアード・プロシージャを発行できます。

ストアード・プロシージャを使うと、開発、保全性、セキュリティ、パフォーマンス、メモリー割当ての面でいくつかの利点があります。ストアード・プロシージャのコール方法など、



ストアド・プロシージャの詳細は、『Oracle8 Server アプリケーション開発者ガイド』を参照してください。

CREATE PROCEDURE コマンドでは、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。パッケージの作成の詳細は、CREATE FUNCTION（4-228 ページ）を参照してください。

**例 1.** 次の文は、スキーマ SAM 内にプロシージャ CREDIT を作成する例です。

```
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の貸方に記入されます。このプロシージャをコールするときは、次の引数を指定しなければなりません。

**ACC\_NO**                      この引数は貸方の記入の対象となる預金口座の番号です。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 文を使って引数 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 の値の分だけ増加させます。

**例 2.** 次の例では、外部プロシージャ C\_FIND\_ROOT により、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 では、BY REF 句を使った参照によりそのパラメータが渡されます。

```
CREATE PROCEDURE
( x IN REAL ) AS
EXTERNAL
EXTERNAL
LIBRARY c_utils
NAME "c_find_root"
PARAMETERS
( x BY REF );
```

外部プロシージャの詳細は、『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。

## 関連項目

ALTER PROCEDURE（4-41 ページ）

CREATE FUNCTION (4-228 ページ)

CREATE LIBRARY (4-244 ページ)

CREATE PACKAGE (4-246 ページ)

DROP PROCEDURE (4-391 ページ)

『PL/SQL ユーザーズ・ガイドおよびリファレンス』

---

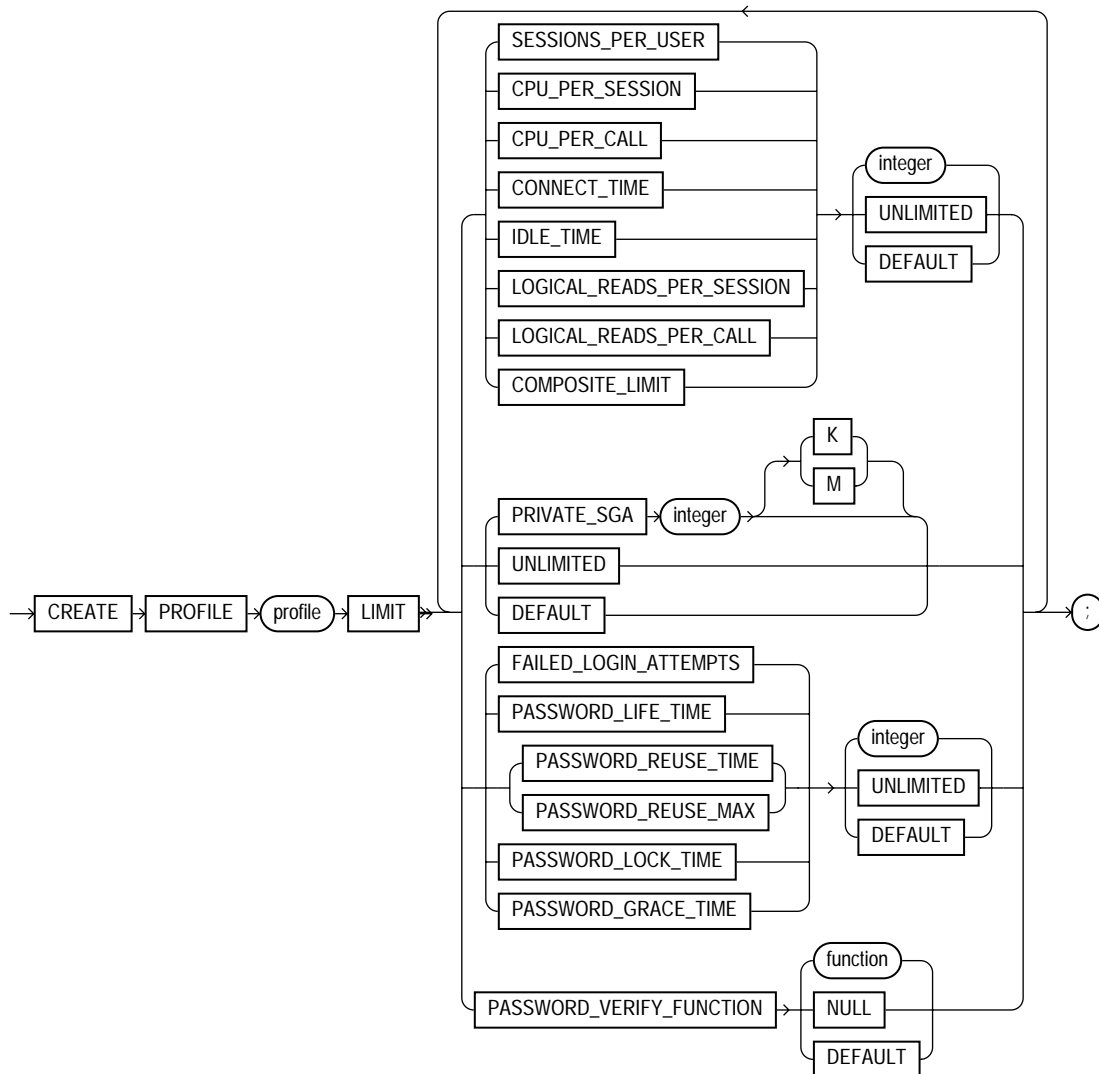
## CREATE PROFILE

### 用途

プロファイルを作成します。プロファイルとは、データベース・リソースに関する制限の集合です。あるユーザーに対してプロファイルを割り当てると、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。

### 前提条件

CREATE PROFILE システム権限が必要です。



## キーワードとパラメータ

作成するプロファイルの名前を指定します。「プロファイルの使用方法」(4-264 ページ)を参照してください。

SESSIONS_PER_USER	1 人のユーザーの同時セッション数を制限します。この値は整数で指定します。
CPU_PER_SESSION	1 セッション当たりの CPU 時間を制限します。この値は 100 分の 1 秒単位で指定します。
CPU_PER_CALL	1 コール（解析または実行、フェッチ）当たりの CPU 時間を制限します。この値は 100 分の 1 秒単位で指定します。
CONNECT_TIME	1 セッション当たりの合計経過時間を制限します。この値は分単位で指定します。
IDLE_TIME	セッション中で連続して非活動な時間を制限します。この値は分単位で指定します。実行時間の長い問合せなどの操作は、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取られるブロックなど、1 セッション中に読み取られるデータ・ブロックの数を指定します。
LOGICAL_READS_PER_CALL	SQL 文を処理するコール（解析または実行、フェッチ）で読み取られるデータ・ブロックの数を指定します。
PRIVATE_SGA	1 つのセッションでシステム・グローバル領域 (SGA) の共有プール内に割り当てることができるプライベート領域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 単位でもこの制限を指定できます。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限り適用されます。SGA 内のセッション用のプライベート領域にはプライベート SQL および PL/SQL 領域が含まれますが、共有 SQL および PL/SQL 領域は含まれません。
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場合、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。「分数での日数指定」（4-264 ページ）を参照してください。
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 に設定しなければなりません。
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 に設定しなければなりません。
PASSWORD_LOCK_TIME	ログインが指定された回数連続して失敗したときに、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と、そのパスワードは使用できなくなります。
PASSWORD_VERIFY_FUNCTION	PL/SQL の複雑なパスワード検証スクリプトを CREATE PROFILE コマンド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。
<i>function</i>	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実行されないことを示します。
	DEFAULT	このプロファイルの対象リソースに対する制限を省略します。このプロファイルを割り当てられたユーザーは、デフォルト・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。
COMPOSITE_LIMIT		1セッション当たりのリソースの総コストをサービス単位で指定します。Oracle では、サービス単位の合計は、CPU_PER_SESSION および CONNECT_TIME、LOGICAL_READS_PER_SESSION、PRIVATE_SGA の重みつき合計として計算されます。  セッションの各リソースの重みの指定方法は、ALTER RESOURCE COST（4-48 ページ）を参照してください。
UNLIMITED		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示します。
DEFAULT		このプロファイルの対象リソースに対する制限を省略します。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」（4-265 ページ）を参照してください。

プロファイルの使用方法

プロファイルとは、データベース・リソースに関する制限の集合です。プロファイルを使用すると、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に関する制限を適用します。

- CONNECT\_TIME または IDLE\_TIME のセッション・リソース制限を超えると、現行トランザクションが自動的にロールバックされ、セッションが終了する。ユーザー・プロセスで次に Oracle に対するコールを実行したときに、エラー・メッセージが戻ります。
- 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する、その処理が自動的に中止され、現行文がロールバックされて、エラーが戻る。ユーザーは、この後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。続いて、ユーザー・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する、その処理が自動的に中止され、現行文がロールバックされて、エラー・メッセージが戻される。このとき、現行トランザクションは有効です。

分数での日数指定

日数を単位と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。分数は、x/y. として表されます。たとえば、1 時間は 1/24、1 分は 1/1440 となります。

パスワード管理およびパスワード保護の詳細は、『Oracle8 Server 管理者ガイド』を参照してください。

ユーザーごとにリソース制限を指定する場合は、必ず次の2つの操作を行ってください。

### リソース制限を使用可能にする

リソース制限は、次のいずれかの方法で使用可能にできます。

- 初期化パラメータ `RESOURCE_LIMIT` を使用する。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。
- `ALTER SYSTEM` コマンドを使用して動的に使用可能にする。`ALTER SYSTEM` (4-88 ページ) を参照してください。

### リソース制限を指定する

ユーザーごとにリソース制限を指定するには、次の手順に従ってください。

1. `CREATE PROFILE` コマンドを使用して、制限を定義するプロファイルを作成します。
2. `CREATE USER` コマンドまたは `ALTER USER` コマンドを使用して、ユーザーにそのプロファイルを割り当てます。

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となって初めて、リソース制限が適用されます。

## DEFAULT プロファイル

Oracle では、`DEFAULT` という名前の付いたデフォルト・プロファイルが自動的に作成されます。最初にこのプロファイルは、無制限のリソースを定義します。`ALTER PROFILE` コマンドを使用すると、このプロファイルに定義されている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や制限に対して `DEFAULT` が指定されている場合は、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。

例 1. 次の文は、プロファイル `SYSTEM_MANAGER` の作成例です。

```
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;
```

次にあるユーザーに **SYSTEM\_MANAGER** プロファイルを割り当てると、そのユーザーは、後続のセッションで次の制限を適用されます。

- ユーザーは無制限に同時実行セッションを使用できる。
- 1つのセッションにおいて、ユーザーはCPU 時間を無制限に消費できる。
- ユーザーが作成した1つのコールで消費できるCPU 時間は30秒以下である。
- 1つのセッションで継続できる時間は45分以下である。
- 1つのセッションの、メモリーとディスクのデータ・ブロック数は、**DEFAULT** プロファイルで指定した制限を受ける。
- ユーザーが作成した1つのコールで、メモリーやディスクから読み取ることができるデータ・ブロック数の合計は1000以下である。
- 1つのセッションで割り当てることができるSGA 内のメモリーは、15KB 以下である。
- 1つのセッションのリソースの総コストは、サービス単位で500万を超えることはできない。リソースの総コストの計算式は、**ALTER RESOURCE COST** コマンドで指定します。
- **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る。

例 2. 次の文は、プロファイル **PROF** の作成例です。

```
CREATE PROFILE prof
  LIMIT PASSWORD_REUSE_MAX DEFAULT
        PASSWORD_REUSE_TIME UNLIMITED;
```

例 3. 次の例では、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;
```

## 関連項目

[ALTER PROFILE](#) (4-43 ページ)



ALTER RESOURCE COST (4-48 ページ)

ALTER SYSTEM (4-88 ページ)

ALTER USER (4-148 ページ)

DROP PROFILE (4-393 ページ)

# CREATE ROLE

## 用途

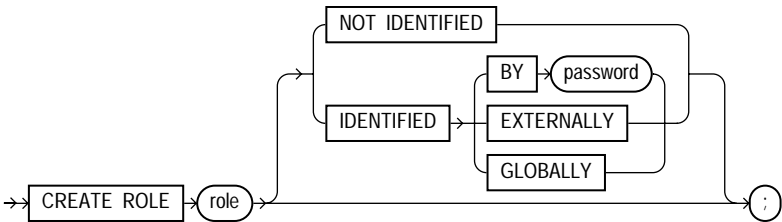
ロールを作成します。ロールとは、ユーザーまたは他のロールに付与することができる権限の集合です。「ロールの使用方法」(4-269 ページ) を参照してください。

グローバル・ロールの使用方法の詳細は、『Oracle8 Server 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

<i>role</i>	作成するロールの名前を指定します。データベースのキャラクタ・セットにマルチバイト文字がサポートされていても、ロールにはシングルバイト文字を最低1つ使用することをお薦めします。「Oracle によって定義されているロール」(4-269 ページ) を参照してください。
NOT IDENTIFIED	指定するロールがデータベースによって認可されているため、パスワードを入力しなくてもこのロールを有効にできることを示します。
IDENTIFIED	ロールを SET ROLE コマンドによって使用可能にするには、まずユーザーが指定メソッドによって認可されている必要があることを示します。
BY <i>password</i>	ロールを使用可能にするときに、ユーザーは Oracle に対してパスワードを指定しなければなりません。データベースのキャラクタ・セットにマルチバイト文字が含まれている場合でも、パスワードをデータベースのキャラクタ・セットのシングルバイト文字だけで指定することもできます。
EXTERNALLY	ロールを使用可能にするには、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、ロールが使用可能にならない場合もあります。サード・パーティ・サービスの詳細は、『Oracle Security Server ガイド』を参照してください。

#### GLOBALLY

ロールを SET ROLE コマンドで、またはログイン時に使用可能にするには、ユーザーが Oracle Security Service によりロールの使用を認可されていないことを示します。

NOT IDENTIFIED オプションと IDENTIFIED 句を両方とも省略すると、ロールには NOT IDENTIFIED がデフォルト値として使用されます。

---

## ロールの使用方法

ロールとは、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することができます。ロールに権限を追加した上で、ユーザーにそのロールを付与することができます。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。ロールを使用可能にする方法は、ALTER USER (4-148 ページ) を参照してください。

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。新しいロールには最初は何のロールも権限も付与されていません。GRANT コマンドを使ってロールに各種の権限を追加します。

ロールを作成すると、そのロールは ADMIN OPTION 付きで付与されます。ADMIN OPTION により、次のことが可能になります。

- ロールがグローバル・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する。
- ロールを他のユーザーまたは他のロールから取り消す。
- ロールへのアクセスに必要な認可を変更するため、ロールに変更を加える。
- ロールを削除する。

---

**注意：** IDENTIFIED GLOBALLY を指定してロールを作成すると、グローバルではないロールの場合とは異なり、作成者にそのロールは付与されません。

---

## Oracle によって定義されているロール

配布メディア上に提供されている 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。事前に定義されているロールは次のとおりです。

- CONNECT
- RESOURCE

- DBA
- EXP\_FULL\_DATABASE
- IMP\_FULL\_DATABASE
- DELETE\_CATALOG\_ROLE
- EXECUTE\_CATALOG\_ROLE
- SELECT\_CATALOG\_ROLE

CONNECT、RESOURCE、DBA の各ロールは、Oracle の以前のバージョンとの互換性を維持するために提供されています。ただし、これらのロールを使うのではなく、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勧めします。Oracle の今後のバージョンでは、これらのロールが自動的に作成されない可能性があるからです。

SELECT\_CATALOG\_ROLE ロールおよび EXECUTE\_CATALOG\_ROLE ロール、DELETE\_CATALOG\_ROLE ロールは、エクスポートされたデータ・ディクショナリのビューとパッケージにアクセスするために提供されています。これらのロールの詳細は、『Oracle8 Server アプリケーション開発者ガイド』を参照してください。

EXP\_FULL\_DATABASE ロールと IMP\_FULL\_DATABASE ロールは、インポート/エクスポート・ユーティリティに便利のように用意されています。

Oracle では、データベースを管理する許可をユーザーに与えるロールも他に作成されます。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る場合があります。

**例 1.** 次の文は、グローバル・ロール VENDOR の作成例です。

```
CREATE ROLE vendor IDENTIFIED GLOBALLY;
```

**例 2.** 次の文は、TELLER ロールの作成例です。

```
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 コマンドでロールを使用可能にする必要があります。

## 関連項目

ALTER ROLE (4-51 ページ)

DROP ROLE (4-394 ページ)

GRANT( システム権限とロール ) (4-432 ページ)

REVOKE( システム権限とロール ) (4-472 ページ)

SET ROLE (4-511 ページ)

# CREATE ROLLBACK SEGMENT

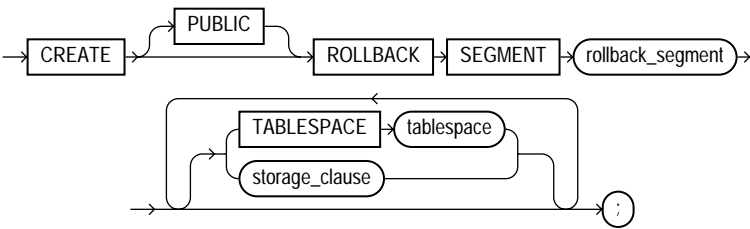
## 用途

ロールバック・セグメントを作成します。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ときに Oracle が使用するオブジェクトです。

## 前提条件

CREATE ROLLBACK SEGMENT システム権限が必要です。また、ロールバック・セグメントを収容する表領域上の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。

## 構文



storage\_clause: STORAGE 句（4-518 ページ）を参照。

## キーワードとパラメータ

PUBLIC	ロールバック・セグメントをパブリックと定義し、任意のインスタンスから使用できることを示します。このオプションを省略すると、ロールバック・セグメントはプライベートと定義され、インスタンスの初期化パラメータ ROLLBACK_SEGMENTS で指定したときにだけそのインスタンスで使うことができます。
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	ロールバック・セグメントを作成する表領域を指定します。このオプションを省略すると、SYSTEM 表領域にロールバック・セグメントが作成されます。「使用上の注意」（4-273 ページ）を参照してください。

<i>storage_clause</i>	<p>ロールバック・セグメントの特性を指定します。STORAGE 句（4-518 ページ）を参照してください。</p> <p><b>注意：</b> <i>storage_clause</i> の PCTINCREASE オプションは、CREATE ROLLBACK SEGMENT では使用できません。</p>
OPTIMAL	<p>STORAGE 句のこの部分では、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 を使用して KB 単位または MB 単位でもサイズを指定できます。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となった場合、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、できるだけ多数のエクステントの割当てを解除します。</p>
NULL	<p>ロールバック・セグメントに最適サイズを指定しません。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。</p> <p>このパラメータには MINEXTENTS および INITIAL、NEXT パラメータに指定したロールバック・セグメントの 1 次領域より小さい値は指定できません。最大値は使用しているオペレーティング・システムによって異なります。値はデータ・ブロック・サイズの倍数に切り上げられます。</p>

## 使用上の注意

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なければなりません。

ロールバック・セグメントを作成すると、最初はオフライン状態になります。Oracle インスタンスでそのロールバック・セグメントをトランザクションに使用できるようにするには、次のいずれかを使ってオンラインの状態にしなければなりません。

- ALTER ROLLBACK SEGMENT コマンド
- ROLLBACK\_SEGMENTS 初期化パラメータ

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の詳細は、『Oracle8 Server 管理者ガイド』を参照してください。

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。

**例：** 次の文は、システム表領域内にデフォルトの記憶域のサイズでロールバック・セグメントを作成する例です。

```
CREATE ROLLBACK SEGMENT rbs_2
TABLESPACE system;
```

上の文は、次の文と同じ結果になります。

```
CREATE ROLLBACK SEGMENT rbs_2
```

## CREATE ROLLBACK SEGMENT

---

```
TABLESPACE system
STORAGE
( INITIAL 10 K
  NEXT 10 K
  MAXEXTENTS UNLIMITED);
```

### 関連項目

CREATE TABLESPACE (4-325 ページ)

CREATE TABLESPACE (4-325 ページ)

ALTER ROLLBACK SEGMENT (4-53 ページ)

DROP ROLLBACK SEGMENT (4-395 ページ)

STORAGE 句 (4-518 ページ)



# CREATE SCHEMA

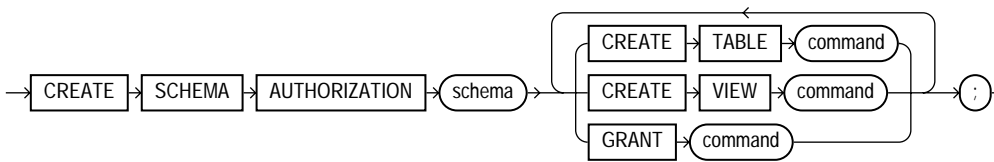
## 用途

複数の表およびビューを作成し、単一トランザクションで複数の権限を付与します。「使用上の注意」(4-275 ページ)を参照してください。

## 前提条件

CREATE SCHEMA 文には CREATE TABLE 文および CREATE VIEW 文、GRANT 文を挿入できます。CREATE SCHEMA 文を発行するには、挿入した文を発行するのに必要な権限を持っていなければなりません。

## 構文



## キーワードとパラメータ

<i>schema</i>	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なければなりません。
CREATE TABLE <i>command</i>	この CREATE SCHEMA 文の一部として発行する CREATE TABLE 文を指定します。CREATE TABLE (4-303 ページ)を参照してください。
CREATE VIEW <i>command</i>	この CREATE SCHEMA 文の一部として発行する CREATE VIEW 文を指定します。CREATE VIEW (4-359 ページ)を参照してください。
GRANT <i>command</i>	この CREATE SCHEMA 文の一部として発行する GRANT 文(オブジェクト権限)を指定します。GRANT(オブジェクト権限) (4-442 ページ)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 で定義されているコマンド構文だけをサポートします。	

## 使用上の注意

CREATE SCHEMA コマンドを使用すると、単一トランザクションで複数のデータ定義言語 (DDL) 文を発行することができます。CREATE SCHEMA 文を実行するために、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ると、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 つでもエラーとなった場合には、すべての文がロールバックされます。

CREATE SCHEMA 文では、他の SQL 文と同様、使用する Tool 固有の終了文字を使用して文を終了します。たとえば、SQL\*Plus または Server Manager で CREATE SCHEMA 文を発行する場合は、セミコロン (;) を使用して文を終了します。CREATE SCHEMA 文内では、個々の文を終了文字で区切らないでください。

CREATE TABLE、CREATE VIEW、GRANT の各文を指定する順序は次に示すように重要ではありません。

- CREATE VIEW 文では、後で実行する CREATE TABLE 文で作成される表に基づいたビューを作成できる。
- CREATE TABLE 文では、後で実行する CREATE TABLE 文で作成される表の主キーに依存する外部キーを持つ表を作成できる。
- GRANT 文では、後で実行する CREATE TABLE 文または CREATE VIEW 文で作成される表またはビューに対して権限を付与できる。

CREATE SCHEMA 文の中の文では、次のように既存のオブジェクトを参照することもできます。

- CREATE VIEW 文では、CREATE SCHEMA 文を実行する前に存在していた表に基づくビューを作成できる。
- GRANT 文では、すでに存在しているオブジェクトに権限を付与できる。

---

---

**注意：** CREATE TABLE、CREATE、INDEX、CREATE CLUSTER を CREATE SCHEMA で使用するとき、PARALLEL 句の構文を使用できません。ただし、オブジェクトを作成するとき並列性は使用できません。

---

---

**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、表 SOX を作成し、ビュー RED\_SOX を作成し、ユーザー WAITES に対して RED\_SOX ビューについての SELECT 権限を付与する例です。

```
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;
```

## 関連項目

CREATE TABLE (4-303 ページ)

CREATE VIEW (4-359 ページ)

GRANT( オブジェクト権限 ) (4-442 ページ)

# CREATE SEQUENCE

---

## 用途

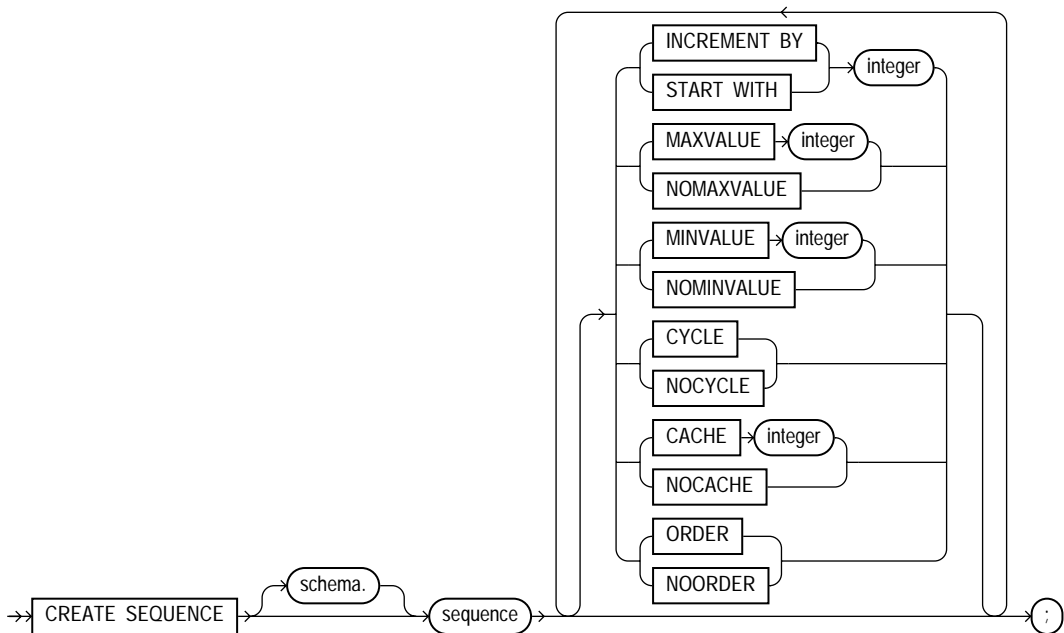
順序を作成します。順序とはデータベース・オブジェクトの1つで、これを使って複数のユーザーが一意の整数を生成することができます。順序を使用すると、主キー値を自動的に生成できます。「順序の使用方法」(4-280 ページ) および「順序のデフォルト」(4-281 ページ)、「順序値へのアクセス」(4-282 ページ)を参照してください。

## 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

<i>schema</i>	順序を定義するスキーマです。 <i>schema</i> を省略すると、自スキーマ内に順序が作成されます。
<i>sequence</i>	作成する順序の名前を指定します。
INCREMENT BY	順序の番号間の増分間隔を指定します。この値は、0 以外の正の整数または負の整数となります。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 の差未満でなければなりません。この値が負の場合、順序は降順となります。この増分値が正の場合は、順序は昇順となります。この句を省略すると、デフォルトで増分間隔は 1 に設定されます。「順序値の増加」(4-281 ページ) を参照してください。
START WITH	生成する順序番号の初期値を指定します。このオプションを指定すると、順序の最小値よりも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となります。降順の場合、デフォルト値は順序の最大値となります。28 桁以内の整数値を指定できます。
MAXVALUE	順序の最大値を指定します。28 桁以内の整数値を指定できます。MAXVALUE 値は、START WITH 以下で、かつ MINVALUE を超える値でなければなりません。
NOMAXVALUE	順序の最大値を、昇順の場合は 10 <sup>27</sup> 、降順の場合は -1 に指定します。これはデフォルトです。

MINVALUE	順序の最小値を指定します。28 桁以内の整数値を指定できます。MINVALUE 値は、START WITH 以下で、かつ MAXVALUE 未満でなければなりません。
NOMINVALUE	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。これはデフォルトです。
CYCLE	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指定します。つまり、昇順の場合は、最大値に達すると最小値が生成されます。降順の場合は、最小値に達すると最大値が生成されます。
NOCYCLE	順序が最大値または最小値に達すると、それ以上値を生成しないことを指定します。これはデフォルトです。
CACHE	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は、この値はそのサイクル内で生成される値の数未満でなければなりません。指定の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なければなりません。  (CEIL (MAXVALUE-MINVALUE)) / ABS(INCREMENT)  「順序番号のキャッシュ」(4-281 ページ)を参照してください。
NOCACHE	順序の値が事前に割り当てられていないことを指定します。
CACHE パラメータと NOCACHE オプションの両方を省略すると、デフォルトで 20 の順序番号がキャッシュされます。	
ORDER	要求どおりに順に順序番号を生成することを保証します。順序番号をタイムスタンプとして使用する場合に、このオプションを使用できます。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。
NOORDER	要求どおりの順序番号の生成を保証しません。
ORDER および NOORDER の両方のオプションを省略すると、NOORDER がデフォルト値として選択されます。ORDER オプションは、Oracle をパラレル・サーバー・オプション付きパラレル・モードで使用する場合、生成順序を保証するためにだけ指定してください。排他モードの場合は、順序番号は必ず順序どおりに生成されます。	

順序の使用方法

順序番号を使用すると、データに自動的に一意の主キー値を生成することができます。また、複数の行や表にわたってキーを連係させることもできます。

Oracle の特殊ルーチンでは、順序の値が自動的に生成されます。したがって、順序をアプリケーション・レベルで実現したときに発生するパフォーマンスのボトルネックを回避できます。たとえば、アプリケーション・レベルで実現された共通順序による処理が、各トランザクションによる順序番号表の強制ロック、次に順序番号の増加、最後に順序番号表の解放であった場合、1 回に生成できる順序番号は 1 つだけです。これに対して、Oracle の順序では、複数の順序番号を同時に生成できるだけでなく、すべての順序番号が確実に一意となります。

順序番号の生成時には、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していく場合、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生じることもあります。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したかどうかに関係なく、順序を生成したユーザーは引き続きその値にアクセス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、個々の順序番号が飛んでいる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。

## 順序のデフォルト

順序のデフォルトは、句を指定しない場合、1を初期値として、上限なしに1つずつ増加していく昇順の順序が作成されます。**INCREMENT BY**に-1だけを指定した場合には、初期値を-1として、下限なしに1つずつ減少していく降順が作成されます。

## 順序値の増加

値が次のいずれかの状態で増加する順序を作成することができます。

- 順序値が無制限に増加する。
- 順序値が事前に定義された制限まで増加し、その制限に達すると終了する。
- 順序値が事前に定義された制限まで増加し、その制限に達すると初期値に戻る。

無制限に増加する順序を作成する場合、昇順では、**MAXVALUE**パラメータの指定を省略するか、または**NOMAXVALUE**オプションを指定します。降順の場合は、**MINVALUE**パラメータの指定を省略するか、**NOMINVALUE**オプションを指定します。

順序を事前に定義した制限で終了させる場合は、昇順では**MAXVALUE**パラメータに値を指定します。降順の場合は、**MINVALUE**パラメータに値を指定します。さらに、**NOCYCLE**オプションも指定します。順序が制限に達したときに順序番号をさらに生成しようとする、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、**MAXVALUE**パラメータと**MINVALUE**パラメータの両方に値を指定します。さらに、**CYCLE**オプションも指定します。**MINVALUE**を指定しないと、デフォルトで**NOMINVALUE**(値 1)が設定されます。

**START WITH**パラメータの値は、順序が作成された後に生成される初期値を設定します。したがって、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、昇順の循環順序が循環するための値ではありません。

## 順序番号のキャッシュ

1つの順序でメモリ内にキャッシュされる値の数は、順序の**CACHE**パラメータの値で指定します。順序をキャッシュすることによって、順序番号をより速く生成することができます。

す。それぞれの順序に対するキャッシュは、その順序の番号に対する最初の要求で確保されます。キャッシュは、**CACHE** 要求ごとに再度確保されます。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、コミットされたデータ操作言語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、失われる可能性のある値の数は、**CACHE** パラメータの値と等しくなります。

順序番号の **CACHE** のデフォルト値は 20 です。

## 順序値へのアクセス

一度順序を作成しておくとし、**SQL** 文に次の疑似列を指定してその値にアクセスできます。

**CURRVAL**                      順序の現在の値を戻します。

**NEXTVAL**                      順序を増加させて、新しい値を戻します。

これらの疑似列の使用の詳細は、「疑似列」(2-30 ページ)を参照してください。

**例** . 次の文は、順序 **ESEQ** を作成する例です。

```
CREATE SEQUENCE eseq  
INCREMENT BY 10
```

最初に **ESEQ.NEXTVAL** 参照すると、1 が戻されます。次に参照すると、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。

## 関連項目

**ALTER SEQUENCE** (4-56 ページ)

**DROP SEQUENCE** (4-397 ページ)



## CREATE SNAPSHOT

### 用途

スナップショットを作成します。スナップショットとは1つ以上の表の問合せの結果を格納する表のことで、リモート・データベース上に存在することもあります。

### 前提条件

スナップショットの作成時の前提条件は次のとおりです。

- 自スキーマ内にスナップショットを作成する場合は、**CREATE SNAPSHOT** および **CREATE TABLE**、**CREATE VIEW** システム権限を持っていること。
- 別のユーザーのスキーマ内にスナップショットを作成するには、**CREATE ANY SNAPSHOT** システム権限を持っていること。
- スナップショットが登録されたスキーマに、スナップショットの実表と索引を格納するのに十分な割当て制限量がターゲット表領域が設定されていること、または **UNLIMITED TABLESPACE** システム権限があること。
- スナップショットを作成、リフレッシュするには、作成者とスナップショット所有者の両方がスナップショットを定義する問合せを発行できること。この可否は、スナップショットを定義する問合せで使用するデータベース・リンクに直接関係しています。

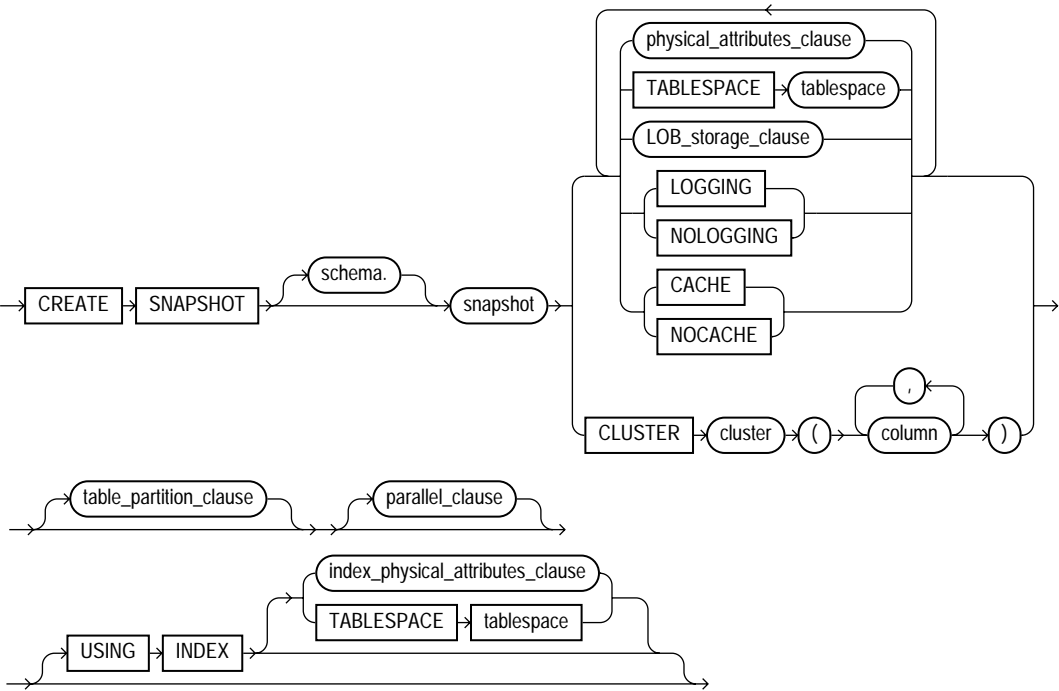
デフォルトでは、新しいスナップショットはすべて主キー・スナップショットとして作成されます。スナップショットの作成時には次のことを考慮してください。

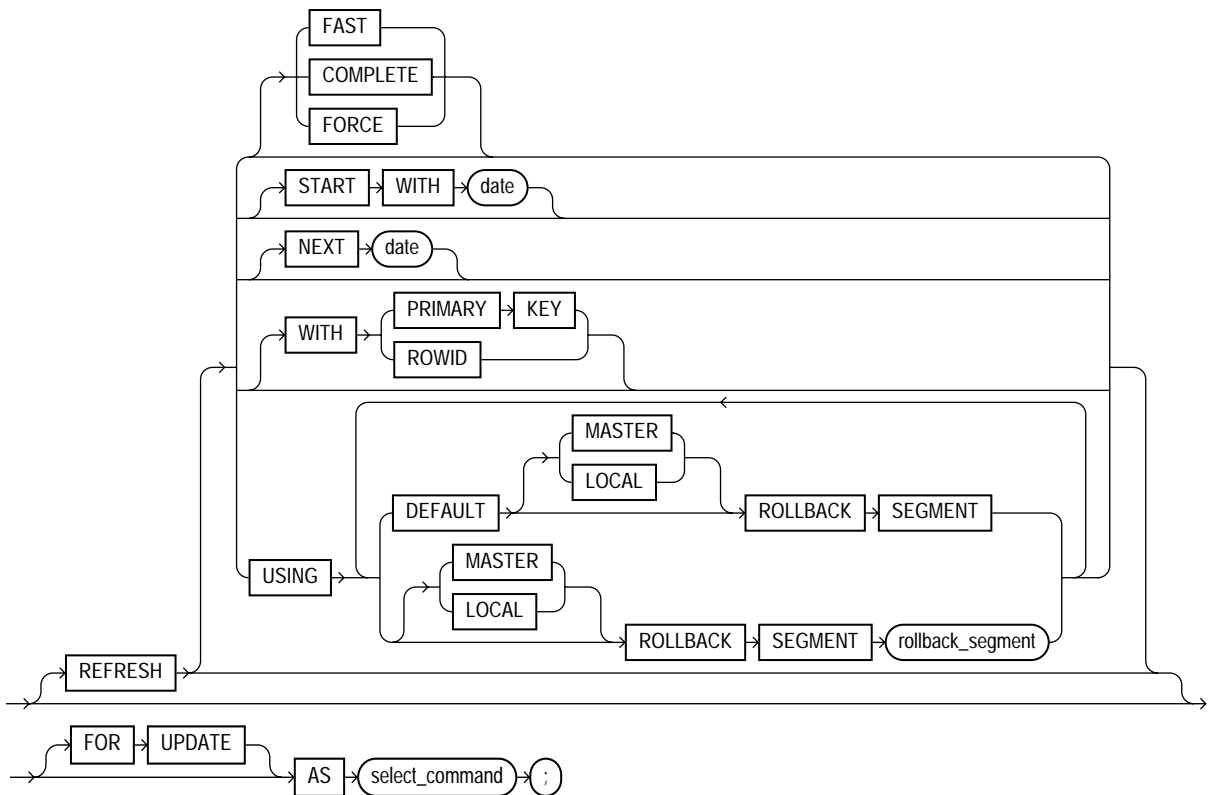
- 新しいスナップショットを作成するときに、スナップショットのマスター表に使用可能な **PRIMARY KEY** 制約が含まれていること。
- スナップショットの定義する問合せがマスター表の主キーですべての列を参照していること。

スナップショットを作成すると、スナップショットのスキーマ内に1つの表および1つのビュー、少なくとも1つの索引が自動的に作成されます。これらのオブジェクトは、スナップショットのデータを管理するために使われます。このとき、これらのオブジェクトを作成するのに必要な権限を持っていなければなりません。これらの権限の詳細は、**CREATE TABLE** (4-303 ページ) および **CREATE VIEW** (4-359 ページ)、**CREATE INDEX** (4-233 ページ) を参照してください。

スナップショットの作成に適用される前提条件については、『Oracle8 Server レプリケーション』を参照してください。

構文





physical\_attributes\_clause: ALTER TABLE (4-105 ページ) を参照。

parallel\_clause: PARALLEL 句 (4-462 ページ) を参照。

index\_physical\_attributes\_clause: ALTER INDEX (4-28 ページ) を参照。

select\_command: SELECT (4-486 ページ) を参照。

LOB\_storage\_clause: CREATE TABLE (4-303 ページ) を参照。

table\_partition\_clause: CREATE TABLE (4-303 ページ) を参照。

## キーワードとパラメータ

*schema*

スナップショットを定義するスキーマを指定します。*schema* を省略すると、自スキーマ内にスナップショットが作成されます。

<i>snapshot</i>	作成するスナップショットの名前を指定します。Oracle は、スナップショットを格納するための表、ビュー、索引の名前を生成するとき、スナップショット名に接頭辞または接尾辞を追加します。スナップショット名は 19 バイトまでに制限してください。これは、Oracle によって生成された名前を 30 バイト以下に制限し、さらにスナップショット名全体が入るようにするためです。「スナップショットについて」(4-288 ページ) を参照してください。
<i>physical_attributes_clause</i>	<p>PCTFREE、PCTUSED、INITRANS、MAXTRANS の各パラメータの値 (USING INDEX 句の中で使用する場合は INITRANS パラメータと MAXTRANS パラメータの値だけ) を設定します。また、Oracle がスナップショットのデータ・メンテナンス用に使用する内部表の記憶領域パラメータを設定します。</p> <p>PCTFREE および PCTUSED、INITRANS、MAXTRANS パラメータの詳細は、CREATE TABLE (4-303 ページ) を参照してください。STORAGE 句の詳細は、STORAGE 句 (4-518 ページ) を参照してください。</p>
TABLESPACE	スナップショットを作成する表領域を指定します。このオプションを省略すると、スナップショット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スナップショットが作成されます。
<i>LOB_storage_clause</i>	LOB 記憶特性を指定します。LOB 記憶域句でのパラメータ指定の詳細は、CREATE TABLE (4-303 ページ) を参照してください。
STORAGE	スナップショットのデータを格納するために使用される表の記憶特性を設定します。
CLUSTER	スナップショットを、指定するクラスタの一部として作成します。クラスタ化スナップショットではクラスタの領域割当てを使用するので、CLUSTER オプションと共に <i>physical_attributes_clause</i> オプションまたは TABLESPACE オプションを使用しないでください。
<i>table_partition_clause</i>	値の指定範囲で表をパーティション化することを指定します。表パーティション句でのパラメータ指定の詳細は、CREATE TABLE (4-303 ページ) を参照してください。「パーティション・スナップショット」(4-293 ページ) も参照してください。
USING INDEX	スナップショットの管理のために自動的に作成される索引の各パラメータを指定します。前述の「 <i>physical_attributes_clause</i> 」を参照してください。
USING ROLLBACK SEGMENT	スナップショットのリフレッシュ時に使用されるローカル・スナップショット・セグメントまたはリモート・マスター・ロールバック・セグメント、あるいはその両方を指定します。
<i>rollback_segment</i>	使用するロールバック・セグメントの名前を指定します。
DEFAULT	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。
MASTER	個々のスナップショットのリモート・マスターで使用するロールバック・セグメントを指定します。
LOCAL	スナップショットが属しているローカル・リフレッシュ・グループで使用するロールバック・セグメントを指定します。

MASTER と LOCAL をいずれも指定しない場合、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。DEFAULT を指定した場合、*rollback\_segment* は指定できません。「ロールバック・セグメントの指定」(4-291 ページ) を参照してください。

## REFRESH

Oracle がスナップショットを自動的にリフレッシュする方法と時期を次の値で指定します。

**FAST**                    高速リフレッシュ、つまりマスター表に関連するスナップショット・ログに格納されている更新データだけをもうリフレッシュを指定します。

**COMPLETE**            完全なリフレッシュ、つまりスナップショットの間合せを再実行するリフレッシュを指定します。

**FORCE**                高速リフレッシュが可能な場合には高速リフレッシュを、そうでない場合には完全なリフレッシュを指定します。リフレッシュ時に高速リフレッシュが可能かどうかは、Oracle によって判断されます。

FAST、COMPLETE、FORCE のオプションをいずれも省略すると、デフォルトで FORCE が使用されます。「スナップショットのリフレッシュ」(4-289 ページ) を参照してください。

**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 値と NEXT 値はどちらも、将来の時間に評価される値です。START WITH 値を省略すると、Oracle はスナップショットの作成時に 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求めます。START WITH 値を指定して NEXT 値を省略した場合、一度だけスナップショットがリフレッシュされます。START WITH 値と NEXT 値を両方とも省略した場合、または REFRESH 句自体を省略した場合、スナップショットは自動的にリフレッシュされません。

**WITH PRIMARY KEY**    主キーのスナップショットの作成を指定します。主キーのスナップショットを使用すると、高速リフレッシュを継続できるスナップショットの機能に影響を与えずに、スナップショット・マスター表を再編成できます。

主キーのスナップショットは、副間合せを持つ単純スナップショットとして定義することもできます。

**WITH ROWID**            ROWID スナップショットの作成を指定します。

ROWID スナップショットを使用すると、Oracle7 リリース 7.3 のマスター表との互換性を維持できます。

WITH PRIMARY KEY と WITH ROWID の両方を省略すると、デフォルトで主キーのスナップショットが自動的に作成されます。「主キーまたは ROWID スナップショットの指定」(4-292 ページ) を参照してください。

## FOR UPDATE

単純スナップショットを更新可能にします。レプリケーション・オプションと組み合わせて使用すると、更新はマスターにも伝播します。詳細は、『Oracle8 Server レプリケーション』を参照してください。

---

**AS *select\_command*** スナップショットの問合せを指定します。スナップショットの作成時に、この問合せが自動的に実行され、実行結果がスナップショットに格納されます。この問合せは有効な SQL 問合せですが、すべての問合せが高速リフレッシュされるわけではありません。「スナップショットの種類」(4-288 ページ) を参照してください。

---

## スナップショットについて

スナップショットとは 1 つ以上の表の問合せの結果を格納する表のことで、リモート・データベース上に存在することもあります。問合せに指定する表をマスター表と呼びます。マスター表が格納されているデータベースをマスター・データベースと呼びます。なお、スナップショットの問合せでは、ユーザー **SYS** が所有する表またはビューを **SELECT** 文の **FROM** 句に指定することはできません。

スナップショットは分散データベースで役に立ちます。スナップショットを使用すると、読取り専用のリモート・データのコピーをローカル・ノード上に格納することができます。このため、スナップショットのデータを、表またはビューと同じように選択することができます。

スナップショットの問合せの **FROM** 句では、表とビューそれぞれを、それを定義しているスキーマで修飾して指定してください。この他の注意については、「ビューの問合せ」(4-362 ページ) (**CREATE VIEW** コマンドのコンテキスト) の説明を参照してください。スナップショットを作成する場合にも同様に指定することをお勧めします。

スナップショットに **LONG** 型の列を入れることはできません。

スナップショットの詳細は、『Oracle8 Server レプリケーション』を参照してください。

## スナップショットの種類

ユーザーは、単純スナップショットと複合スナップショットの 2 種類のスナップショットを作成できます。

単純スナップショットとは、1 つのリモート表に基づくスナップショット、または限られたタイプの副問合せによって複数の表に対して定義されるスナップショットのことです。副問合せで使用する単純スナップショットの詳細は、『Oracle8 Server レプリケーション』を参照してください。

単純スナップショットでは、スナップショットの問合せ (*select\_command\_clause*) の中に次のものは含まれません。

- **GROUP BY** 句
- **CONNECT BY** 句
- 固有の関数または集約関数
- 結合 (使用可能なタイプの副問合せを除く)
- 集合演算子

複合スナップショットとは、スナップショットの間合せに、単純スナップショットの間合せでは指定できない構成体が1つ以上含まれているスナップショットのことです。複合スナップショットは、複数のマスター・データベース上の複数のマスター表を基にする場合があります。

## スナップショットのリフレッシュ

スナップショットのマスター表は変更できるため、スナップショットのデータを必要に応じて更新して、マスター表に現在格納されているデータが正確に反映されている状態にしておかなければなりません。この目的で行うスナップショットの更新プロセスを、スナップショットのリフレッシュと言います。`CREATE SNAPSHOT` コマンドに `REFRESH` 句を指定すると、自動的にスナップショットをリフレッシュする時間をスケジューリングし、そのモードを指定することができます。

スナップショットの作成後も、`ALTER SNAPSHOT` コマンドの `REFRESH` 句を使って、その自動リフレッシュのモードと時間を変更できます。また、`DBMS_SNAPSHOT.REFRESH()` プロシージャを使用すると、スナップショットを即時リフレッシュすることができます。

### リフレッシュ・モードの指定

`REFRESH` 句で `FAST` オプションまたは `COMPLETE` オプションを指定すると、リフレッシュ・モードを指定できます。

**高速** . 高速リフレッシュを実行する場合、スナップショット・ログに記録されたマスター表に対する変更内容に基づいて、スナップショットが更新されます。スナップショット・ログの詳細は、`CREATE SNAPSHOT LOG` (4-294 ページ) を参照してください。

次の条件すべてが当てはまる場合にだけ、高速リフレッシュが実行されます。

- スナップショットが単純スナップショットである場合
- スナップショットのマスター表がスナップショット・ログを持っている場合
- スナップショットが前回リフレッシュされる前、またはスナップショットの作成前にスナップショット・ログが作成されている場合

高速リフレッシュを指定していて、上記の条件がすべて当てはまる場合に、高速リフレッシュが自動的に実行されます。上記の条件のうちどれか1つでも当てはまらなければ、リフレッシュ時にエラーが戻され、スナップショットはリフレッシュされません。

**完全** . 完全リフレッシュを実行する場合、スナップショットの間合せが再実行され、その実行結果がスナップショットに格納されます。完全リフレッシュを指定すると、高速リフレッシュが実行可能かどうかに関係なく、完全なリフレッシュが実行されます。

高速リフレッシュは、完全なリフレッシュより高速に処理できる場合もあります。これは、高速リフレッシュの方が、ネットワークを介して行われるマスター・データベースからスナップショットのデータベースへのデータの送信量が少ないためです。高速リフレッシュで

は最後のリフレッシュ以降のマスター表のデータへの変更内容だけが送信されるのに対し、完全なリフレッシュでは、スナップショットの問合せの実行結果すべてが送信されます。

また、REFRESH 句に FORCE オプションを指定すると、スケジューリングしたリフレッシュ時間にスナップショットをリフレッシュする方法を自動的に決定させることができます。高速リフレッシュ条件を基にして、高速リフレッシュを実行できる場合には、自動的に高速リフレッシュが実行されます。高速リフレッシュを実行できない場合には、完全なリフレッシュが実行されます。

**例.** 次の文は、ニューヨークの SCOTT 所有の従業員表のデータが格納されている単純スナップショット EMP\_SF を作成する例です。

```
CREATE SNAPSHOT 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、現在の SYSDATE を基にして NEXT 値を評価することによって、最初の自動リフレッシュ時間が求められます。現在、ニューヨークの従業員表にスナップショット・ログが存在する場合、スナップショット作成の 7 日後から、7 日ごとにそのスナップショットの高速リフレッシュが実行されます。

また、この文では、Oracle がスナップショットをメンテナンスするための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指定

スナップショットを自動的にリフレッシュさせるように設定するためには、必ず次の操作を行ってください。

1. CREATE SNAPSHOT 文の REFRESH 句に START WITH と NEXT パラメータを指定します。これらのパラメータで、最初の自動リフレッシュ開始時間と自動リフレッシュ間の間隔を設定します。
2. 初期化パラメータ SNAPSHOT\_REFRESH\_PROCESSES、SNAPSHOT\_REFRESH\_INTERVAL、SNAPSHOT\_REFRESH\_KEEP\_CONNECTIONS を使用して、スナップショットの 1 つ以上のジョブ・キュー・プロセスを使用可能にします。この後、使用可能となったジョブ・キュー・プロセスによって、データベース内の各スナップショットの自動リフレッシュ時間が調べられます。現在の時刻またはその前にリフレッシュが行われるようスケジューリングされている各スナップショットについて、ジョブ・キュー・プロセスの 1 つで次の操作が実行されます。
  - 次の自動リフレッシュ時間を求めるために、スナップショットの NEXT 値を再度評価する。
  - スナップショットをリフレッシュする。



- 次の自動リフレッシュ時間をデータ・ディクショナリに格納する。

これらの初期化パラメータの詳細は、『Oracle8 Server リファレンス・マニュアル』を参照してください。

**例.** 次の文は、ダラスとボルティモアの従業員表を問い合わせる複合スナップショット ALL\_EMPS を作成する例です。

```
CREATE SNAPSHOT all_ems
  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;
```

このスナップショットは翌日の午前 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。このコマンドでは高速リフレッシュも完全リフレッシュも指定していないため、このスナップショットのリフレッシュ方法は自動的に決定されます。この場合、ALL\_EMPS が複合スナップショットであるため、完全なリフレッシュが実行されます。

この文では、スナップショットを格納するために使用される表と索引両方の記憶特性を次のように設定しています。

- 最初の STORAGE 句で、表の最初と 2 番目のエクステントのサイズをそれぞれ 50KB として設定する。
- 2 番目の STORAGE 句 (USING INDEX オプション付きで指定) では、索引の最初と 2 番目のエクステントのサイズをそれぞれ 25KB として設定する。

## ロールバック・セグメントの指定

マスター・サイトおよびローカル・サイトの両方でリフレッシュ中に使用されるロールバック・セグメントを指定できます。

ローカル・スナップショットのロールバック・セグメントは、リフレッシュ・グループ・レベルで格納されます。自動リフレッシュ・パラメータを指定すると、新しいリフレッシュ・グループが自動的に作成され、スナップショットがバックグラウンド・プロセスでリフレッシュされます。ローカルのロールバック・セグメントを指定すると、この新しいリフレッシュ・グループと関連付けられます。自動リフレッシュ・パラメータを指定せずに、ローカル・ロールバック・セグメントを指定すると、エラーが発生します。

マスター・ロールバック・セグメントは、スナップショットごとに格納されます。マスター・ロールバック・セグメントは、スナップショットの作成時およびリフレッシュ時に妥

当性検査を受けます。複合スナップショットの場合、マスター・ロールバック・セグメントは指定があっても無視されます。

---

**注意：** ALTER SNAPSHOT では、DEFAULT の指定が最も有効です。  
ALTER SNAPSHOT (4-76 ページ) を参照してください。

---

CREATE SNAPSHOT または ALTER SNAPSHOT でロールバック・セグメントを指定した後、このロールバック・セグメントが自動的に選択されるように設定するには、ALTER SNAPSHOT で DEFAULT を指定します。

**例.** 次の例では、リモート・マスターにはロールバック・セグメント MASTER\_SEG 付きで、スナップショット SALE\_EMP が属するローカル・リフレッシュ・グループにはロールバック・セグメント SNAP\_SEG 付きで、スナップショット SALE\_EMP が作成されます。

```
CREATE SNAPSHOT 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・セグメントを使っ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 SNAPSHOT bogus
REFRESH FAST START WITH SYSDATE NEXT SYSDATE + 7
USING DEFAULT ROLLBACK SEGMENT snap_seg
AS SELECT * FROM faux;
```

## 主キーまたは ROWID スナップショットの指定

主キーのスナップショットの作成に必要な定義を次に示します。

- 主キーのすべての列をスナップショットの定義に挿入する。
- 使用可能になっている主キーの制約をスナップショット・マスター上で定義する。

主キーのスナップショットを高速リフレッシュする場合は、まず **WITH PRIMARY KEY** を指定したスナップショット・マスター・ログを作成します。スナップショット・マスター・ログには、**ROWID** も記録できます。

**WITH** 句が指定されていない場合は、主キー・スナップショットがデフォルトです。

主キーのスナップショットを作成するには、上記の条件を満たす必要があります。

**例 1.** 次の文は主キーのスナップショット HUMAN\_GENOME を作成する例です。

```
CREATE SNAPSHOT human_genome
```

```
REFRESH FAST START WITH SYSDATE NEXT SYSDATE + 1/4096  
WITH PRIMARY KEY  
AS SELECT * FROM genome_catalog;
```

例 2. 次の文は、ROWID スナップショットを作成する例です。

```
CREATE SNAPSHOT emp_data WITH ROWID  
AS SELECT * FROM emp_table73;
```

## パーティション・スナップショット

スナップショットは基本的には表なので、パーティション・スナップショットはパーティション表と同じです。オプションは、CREATE TABLE と ALTER TABLE のパーティション表のオプションと同じ構文および意味です。スナップショットおよびスナップショット・ログでは、次の操作ができないという点だけが異なりなります。

- DROP PARTITION
- TRUNCATE PARTITION
- EXCHANGE PARTITION

マスター表のパーティションの削除 (DROP) や切捨て (TRUNCATE) によって、一括して削除することはできません。そのため、パーティションを削除したり、切り捨てたりした後で、すべてのスナップショットを手動でリフレッシュしなければなりません。高速リフレッシュが失敗したとしても、エラーにはなりません。

## 関連項目

ALTER SNAPSHOT (4-76 ページ)

CREATE SNAPSHOT LOG (4-294 ページ)

DROP SNAPSHOT (4-399 ページ)

SELECT (4-486 ページ)

## CREATE SNAPSHOT LOG

---

### 用途

スナップショット・ログを作成します。スナップショット・ログとは、スナップショットのマスター表に関連付けられる表です。Oracle は、スナップショット・ログ内のマスター表のデータに対する変更内容を格納してから、スナップショット・ログを使用して、マスター表のスナップショットをリフレッシュします。「スナップショット・ログの使用方法」(4-296 ページ) を参照してください。

### 前提条件

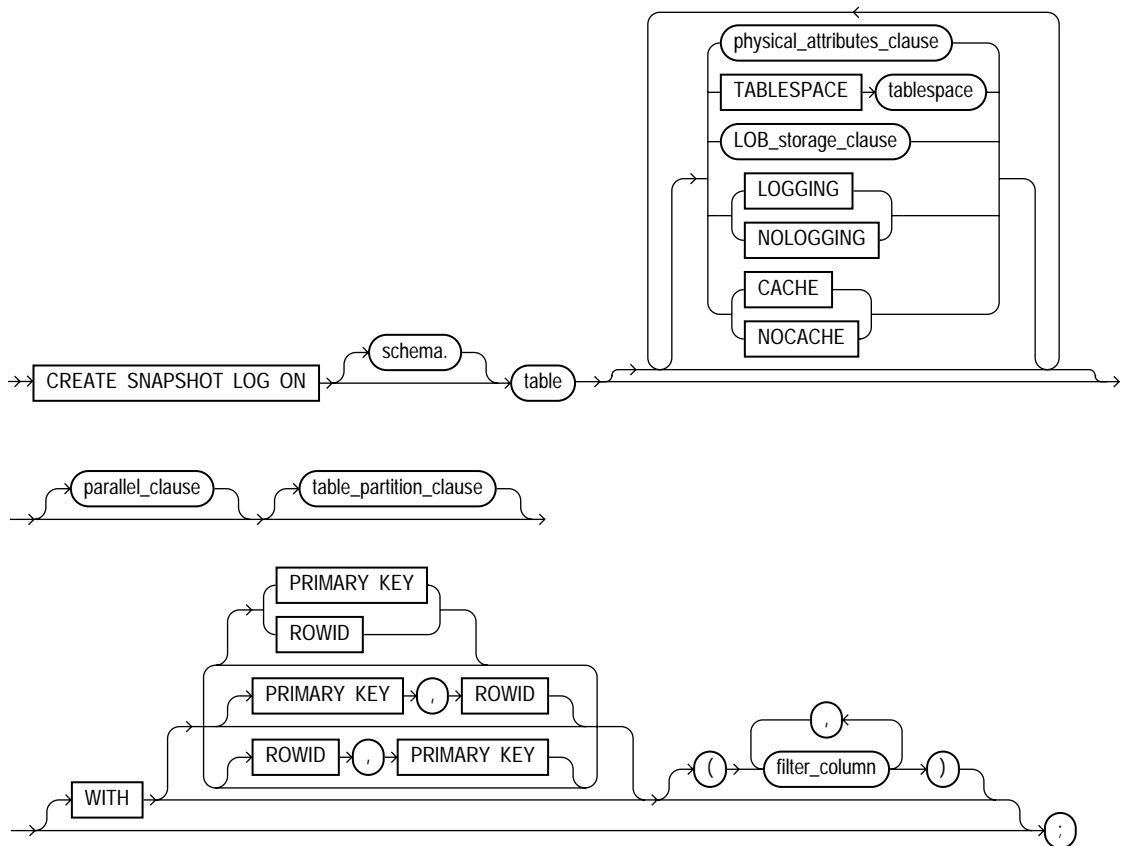
スナップショット・ログを作成するために必要な権限は、スナップショット・ログに関連付けられた、基礎となるオブジェクトの作成に必要な権限と直接関係があります。

- マスター表を所有しているユーザーに、CREATE TABLE 権限があれば、関連するスナップショット・ログを作成できる。
- 別のユーザーのスキーマ内に表のスナップショット・ログを作成する場合は、CREATE ANY TABLE 権限および COMMENT ANY TABLE 権限、マスター表に対する SELECT 権限または SELECT ANY TABLE 権限を持っている必要がある。

どちらの場合も、スナップショット・ログの所有者は、スナップショット・ログを格納するために、表領域に十分な割当て制限量を持っていることが必要です。

スナップショット・ログの作成に必要な前提条件については、『Oracle8 Server レプリケーション』を参照してください。

## 構文



`parallel_clause`:`PARALLEL` 句 (4-462 ページ) を参照。

`storage_clause`:`STORAGE` 句 (4-518 ページ) を参照。

`LOB_storage_clause`:`CREATE TABLE` (4-303 ページ) を参照。

`table_partition_clause`:`CREATE TABLE` (4-303 ページ) を参照。

`physical_attributes_clause`:`CREATE TABLE` (4-303 ページ) を参照。

キーワードとパラメータ

<i>schema</i>	スナップショット・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。 <i>schema</i> を省略すると、その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。スナップショット・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 <b>SYS</b> のスキーマ内の表に対してはスナップショット・ログを作成できません。
<i>table</i>	スナップショット・ログの作成対象のマスター表の名前を指定します。ビューに対しては、スナップショット・ログを作成できません。
<b>WITH</b>	マスター内の行の更新時に、スナップショット・ログに主キーか <b>ROWID</b> のいずれか、あるいは主キーと <b>ROWID</b> の両方を記録するかどうかを指定します。  この句では、スナップショット・ログでフィルタ列を記録するかどうかも指定します。フィルタ列とは、副問合せで単純スナップショットとして定義されているスナップショットによって参照される非主キー列のことです。「主キーおよび <b>ROWID</b> 、フィルタ列の記録」(4-297 ページ)を参照してください。  <b>PRIMARY KEY</b> 更新されるすべての行の主キーをスナップショット・ログに記録するように指定します。  <b>ROWID</b> 更新されるすべての行の <b>ROWID</b> をスナップショット・ログに記録するように指定します。  <i>filter_column</i> スナップショット・ログに記録されるフィルタ列のリストを指定する、カンマで区切られたリストです。  Oracle により、マスター内で更新されるすべての行の主キーがデフォルトで記録されます。
<i>physical_attributes_clause</i>	<b>PCTFREE</b> 、 <b>PCTUSED</b> 、 <b>INITRANS</b> 、 <b>MAXTRANS</b> の各パラメータの値およびスナップショット・ログの記憶特性値を設定します。これらのパラメータの説明は、 <b>CREATE TABLE</b> (4-303 ページ) および <b>STORAGE</b> 句 (4-518 ページ) を参照してください。
<b>TABLESPACE</b>	スナップショット・ログを作成する表領域を指定します。このオプションを省略すると、スナップショット・ログが定義されているスキーマの所有者のデフォルトの表領域内にスナップショット・ログが作成されます。
<b>STORAGE</b>	スナップショット・ログに対する記憶特性を設定します。 <b>STORAGE</b> 句 (4-518 ページ) を参照してください。
<i>LOB_storage_clause</i>	<b>LOB</b> 記憶特性を指定します。 <b>LOB</b> 記憶域句でのパラメータ指定の詳細は、 <b>STORAGE</b> 句 (4-518 ページ) を参照してください。
<i>table_partition_clause</i>	値の指定範囲で表をパーティション化することを指定します。表パーティション句でのパラメータ指定の詳細は、 <b>CREATE TABLE</b> (4-303 ページ) を参照してください。

スナップショット・ログの使用方法

スナップショット・ログとは、スナップショットのマスター表に関連付けられる表です。マスター表のデータに変更が加えられると、その変更を記述した行がスナップショット・ログに追加されます。この行は後で、マスター表を基にするスナップショットのリフレッシュ時に使われます。このプロセスを高速リフレッシュといいます。スナップショット・ログがな

い場合は、スナップショットをリフレッシュするためのスナップショットの間合せが再実行されます。このプロセスを完全なリフレッシュといいます。通常の場合、高速リフレッシュでは、完全なリフレッシュよりも高速に処理できます。

スナップショット・ログは、マスター表と同一のスキーマ内のマスター・データベースの中にあります。マスター表にはスナップショット・ログは1つしか必要ありません。Oracleでは、このスナップショット・ログを使用して、マスター表を基にするすべての単純スナップショットに対して高速リフレッシュが実行されます。Oracleによるスナップショットのリフレッシュ方法を含むスナップショットの詳細は、CREATE SNAPSHOT (4-283 ページ) および『Oracle8 Server レプリケーション』を参照してください。

**例.** 次の文は、主キー値だけを記録する従業員表のスナップショット・ログを作成する例です。

```
CREATE SNAPSHOT LOG ON emp
PCTFREE 5
TABLESPACE users
STORAGE (INITIAL 10K NEXT 10K PCTINCREASE 50);
```

Oracle では、このスナップショットを使って、EMP 表に対して後から作成されるどの単純主キー・スナップショットに対しても高速リフレッシュを実行できます。

## 主キーおよび ROWID、フィルタ列の記録

Oracle で主キーのスナップショットを自動作成するには、マスター表の中の更新済みの行の主キーをスナップショット・ログに記録する必要があります。ROWID スナップショットの場合も、同様に ROWID をスナップショット・ログに記録しなければなりません。主キーと ROWID の両方を記録すると、主キーおよび ROWID の両方のスナップショットが存在する構成をサポートできます。

副問合せで単純スナップショットとして定義された主キーのスナップショットの場合は、定義している副問合せが参照するすべてのフィルタ列がスナップショット・ログに記録されていなければなりません。

**例 1.** 次の例では、更新された行の主キーだけを記録するスナップショット・ログが作成されます。

```
CREATE SNAPSHOT LOG ON emp;
CREATE SNAPSHOT LOG ON emp WITH PRIMARY KEY;
```

**例 2.** 次の例では、更新された行の主キーと ROWID の両方を記録するスナップショット・ログが作成されます。

```
CREATE SNAPSHOT LOG ON sales WITH ROWID, PRIMARY KEY;
```

**例 3.** 次の例では、主キーとフィルタ列 ZIP への更新内容を記録するスナップショット・ログが作成されます。

```
CREATE SNAPSHOT LOG ON address WITH (zip);
```

### 関連項目

[ALTER SNAPSHOT LOG \(4-84 ページ\)](#)

[CREATE SNAPSHOT \(4-283 ページ\)](#)

[CREATE TABLE \(4-303 ページ\)](#)

[DROP SNAPSHOT \(4-399 ページ\)](#)



# CREATE SYNONYM

## 用途

シノニムを作成します。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、スナップショット、別のシノニムに付ける別名です。「シノニムの使用方法」(4-300 ページ)を参照してください。

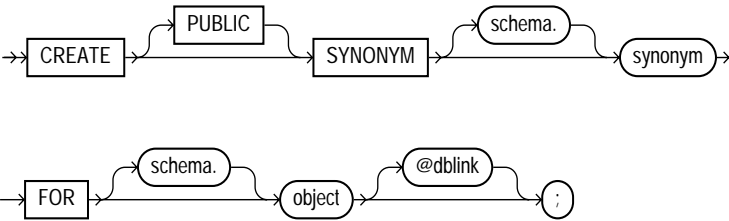
## 前提条件

自スキーマ内にプライベート・シノニムを作成する場合、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、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

<b>PUBLIC</b>	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。このオプションを省略すると、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することができます。
<i>schema</i>	シノニムを定義するスキーマを指定します。 <i>schema</i> を省略すると、自スキーマ内にシノニムが作成されます。 <b>PUBLIC</b> を指定した場合、 <i>schema</i> は指定できません。「シノニムの有効範囲」(4-301 ページ)を参照してください。
<i>synonym</i>	作成するシノニムの名前を指定します。

FOR	<p>シノニムを作成するオブジェクトを指定します。Object を <i>schema</i> で修飾しないと、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、次の型のものを指定できます。</p> <ul style="list-style-type: none"><li>表</li><li>オブジェクト表</li><li>ビュー</li><li>オブジェクト・ビュー</li><li>順序</li><li>ストアド・プロシージャまたは関数、パッケージ</li><li>スナップショット</li><li>シノニム</li></ul> <p><b>OBJ</b> オブジェクト表やオブジェクト・ビューにはシノニムを作成できますが、オブジェクト型には作成できません。</p> <p>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。</p> <p>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。</p>
<i>dblink</i>	<p><i>dblink</i> を完全に指定するか、<i>dblink</i> の一部を指定すると、スキーマ・オブジェクトが格納されているリモート・データベース上でそのスキーマ・オブジェクトのシノニムを作成することができます。データベース・リンクの参照方法の詳細は、「リモート・データベース内のオブジェクトを参照」(2-50 ページ) を参照してください。<i>dblink</i> を指定して、<i>schema</i> 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勧めします。</p> <p><i>dblink</i> を省略すると、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。</p>

シノニムの使用方法

シノニムは、次の文の中でシノニムの実オブジェクトの位置を表すかわりに使用できます。

DML 文	DDL 文
SELECT	AUDIT
INSERT	NOAUDIT
UPDATE	GRANT
DELETE	REVOKE
EXPLAIN PLAN	COMMENT
LOCK TABLE	

シノニムは、データベースのセキュリティを維持し、データを効率的に操作するために使用されます。オブジェクトに対してシノニムを作成すると、次のことが可能となります。

- オブジェクトを、所有者を指定せずに参照する。
- オブジェクトを、格納データベースを指定せずに参照する。
- オブジェクトに対してもう1つ名前を与える。

シノニムによりデータ独立性と位置の透過性の両方を実現できます。シノニムを使用すると、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変更なしに機能します。

## シノニムの有効範囲

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。Oracle は、オブジェクトの参照を、PUBLIC シノニム・レベルで解決する前に、スキーマ・レベルで解決しようとします。次の条件が両方とも当てはまる場合、オブジェクトの参照を解決するときだけ、パブリック・シノニムが使用されます。

- 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
- 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

たとえば、スキーマ SCOTT とスキーマ BLAKE 内にそれぞれ DEPT という名前の表が定義されていて、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。このとき、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 は次に示す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 表にアクセスすることができます。

```
SELECT *
  FROM dept;
```

**例 1.** スキーマ SCOTT 内の表 MARKET\_RESEARCH に対してシノニム MARKET を定義する場合、次の文を発行します。

```
CREATE SYNONYM market
  FOR scott.market_research;
```

**例 2.** 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、次の文を発行します。

```
CREATE PUBLIC SYNONYM emp
    FOR scott.emp@sales;
```

別のスキーマ内に実表が定義されている場合は、実表と同一の名前をシノニムに指定しても構いません。

## 関連項目

CREATE DATABASE LINK (4-220 ページ)

CREATE TABLE (4-303 ページ)

CREATE VIEW (4-359 ページ)

---

## CREATE TABLE

### 用途

表を作成します。表はユーザー・データを格納する基本構造であり、次の情報が指定されます。

- 列の定義
- 表編成の定義
- **OBJ** オブジェクトによる列の定義
- 整合性制約
- 表の表領域
- 記憶特性
- オプションのクラスタ
- 任意の問合せから得られるデータ
- 表作成に使用する並列度、および表に対する問合せのデフォルトの並列度
- パーティション化の定義
- 索引構成 / ヒープ構成

これらの用途の詳細は、「例」（4-318 ページ）を参照してください。

---

**注意：** コマンドや句の説明で**OBJ**の印が前に付いている箇所は、Oracle Object Option がデータベース・サーバーにインストールされている場合にだけ読んでください。

---

**OBJ** オブジェクト表、または列の定義にオブジェクト型を使う表を作成する場合に、**CREATE TABLE** を使用してください。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。

**OBJ** 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うこともできます。オブジェクト作成の詳細は、『Oracle8 Server アプリケーション開発者ガイド』および **CREATE TYPE**（4-342 ページ）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、**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、**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上に割当

て制限量を持っているか、または UNLIMITED TABLESPACE システム権限を持っていないければなりません。

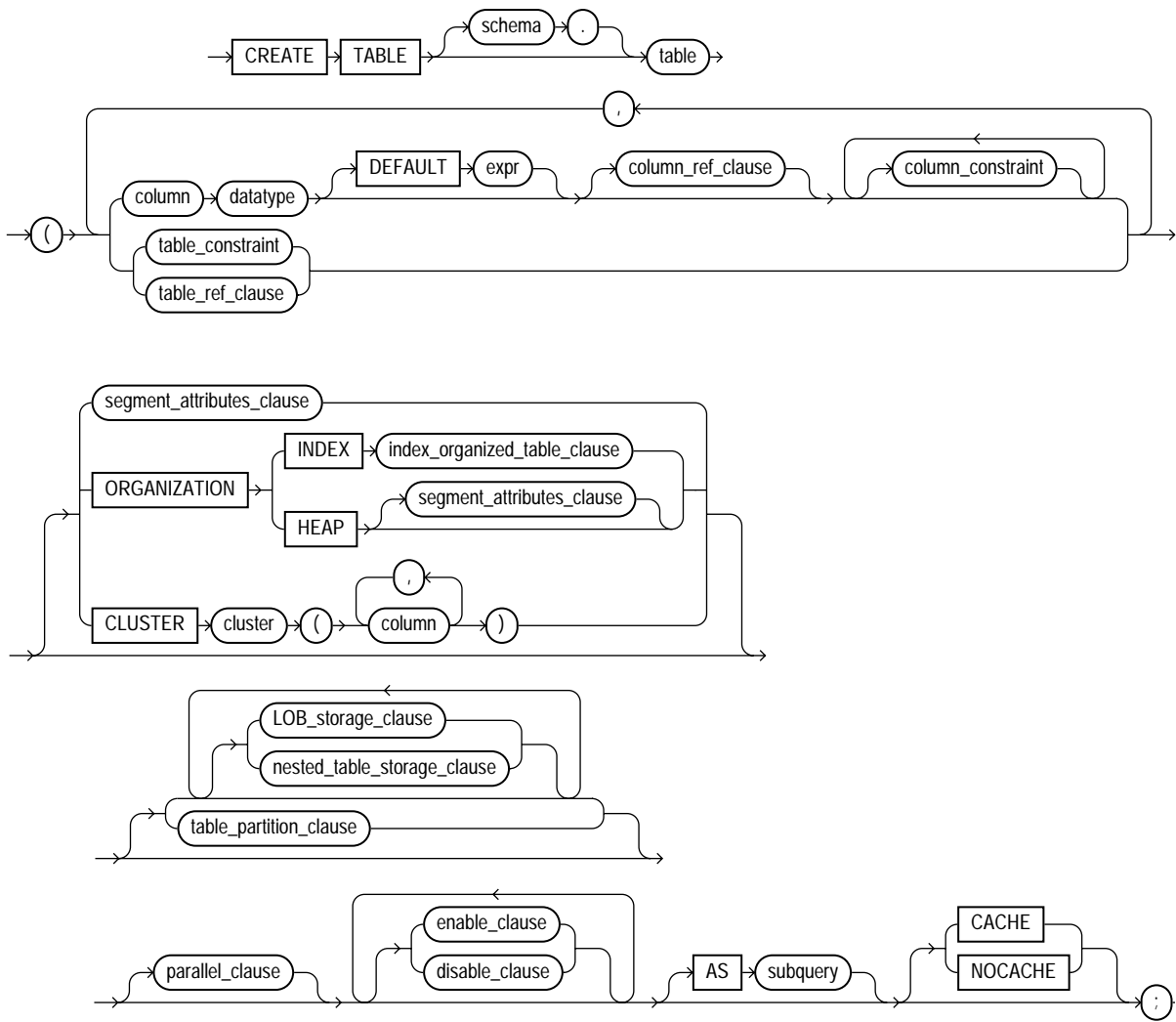
**●** 上記の表権限に加え、タイプを使用する表を作成する場合、表の所有者は、表が参照するすべてのタイプにアクセスするため明示的に EXECUTE オブジェクト権限を持っているか、EXECUTE ANY TYPE システム権限を持っている必要があります。これらの権限は、ロールを介して取得するのではなく明示的に付与されなければなりません。

さらに、表の所有者が表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、参照タイプに対する 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、表の所有者は表にアクセスする権限を他のユーザーに付与できません。

型を使用して表を作成する場合に必要な権限の詳細は、『Oracle8 Server アプリケーション開発者ガイド』を参照してください。

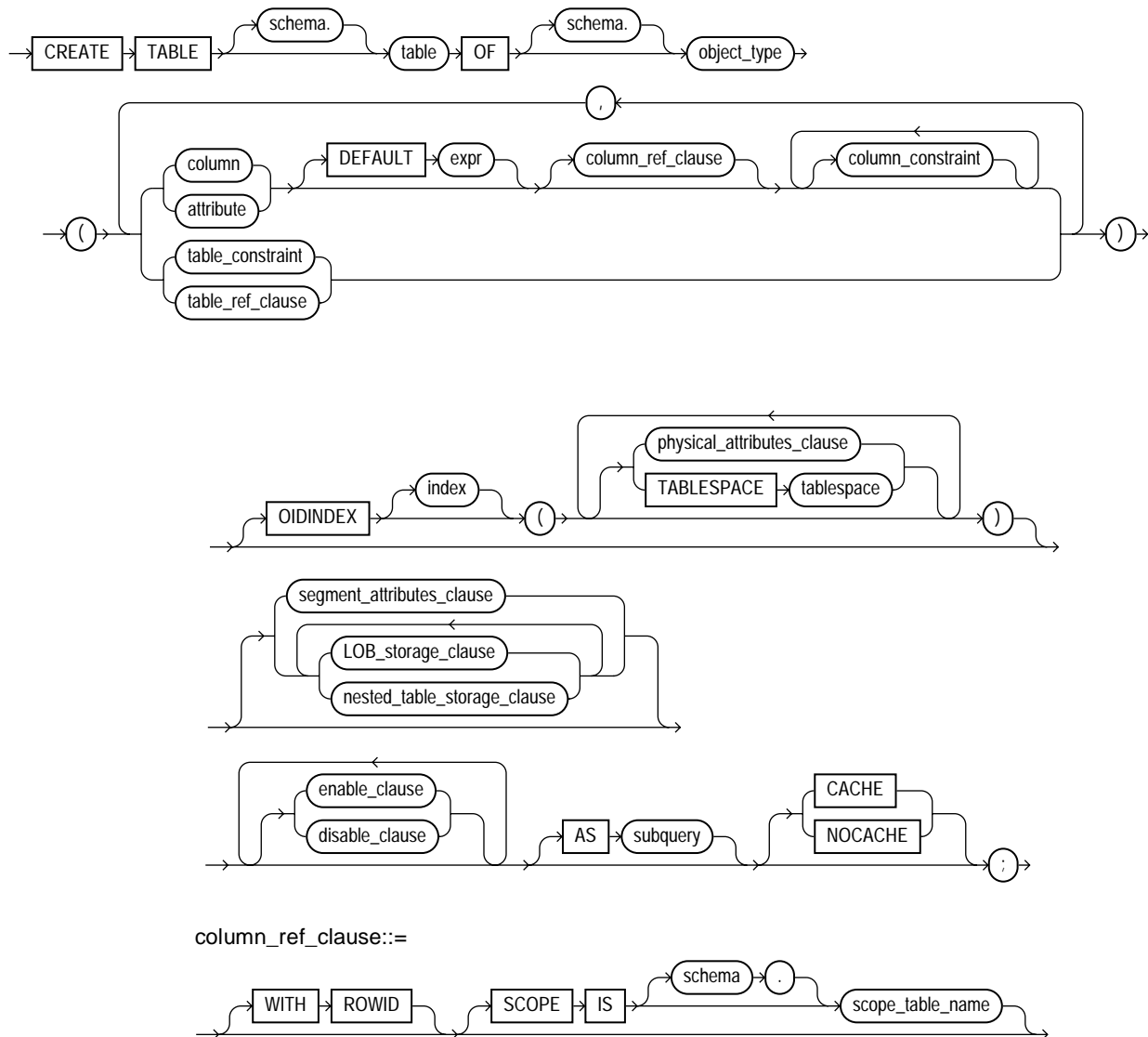
構文

リレーショナル表の定義 ::=



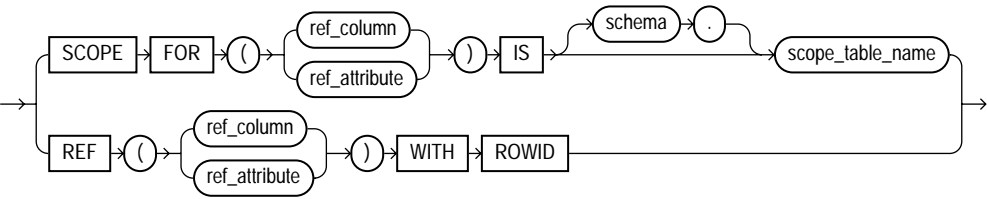
オブジェクト表の定義 ::=

## CREATE TABLE

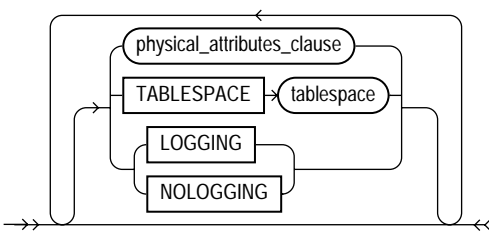




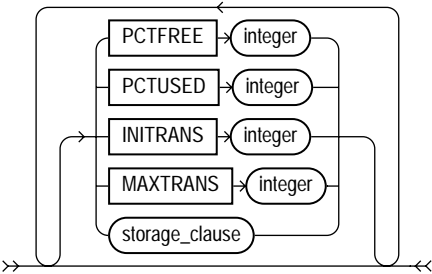
table\_ref\_clause::=



segment\_attributes\_clause::=

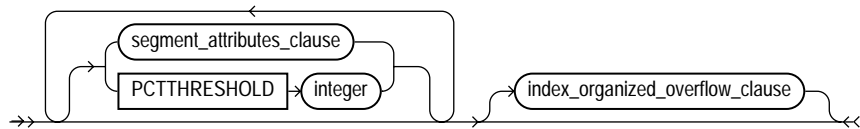


physical\_attributes\_clause::=



storage\_clause: STORAGE 句 (4-518 ページ) を参照。

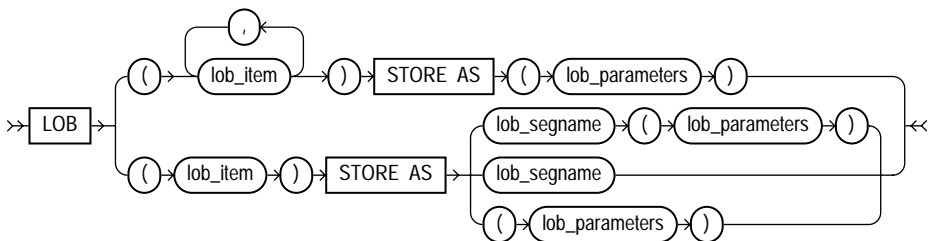
index\_organized\_table\_clause ::=

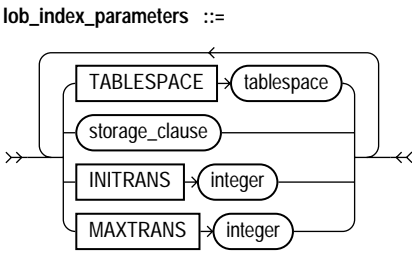
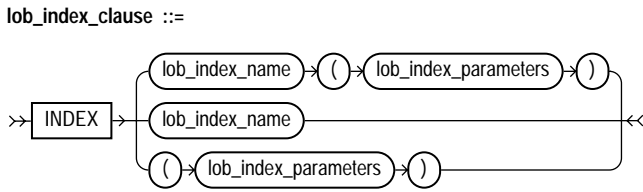
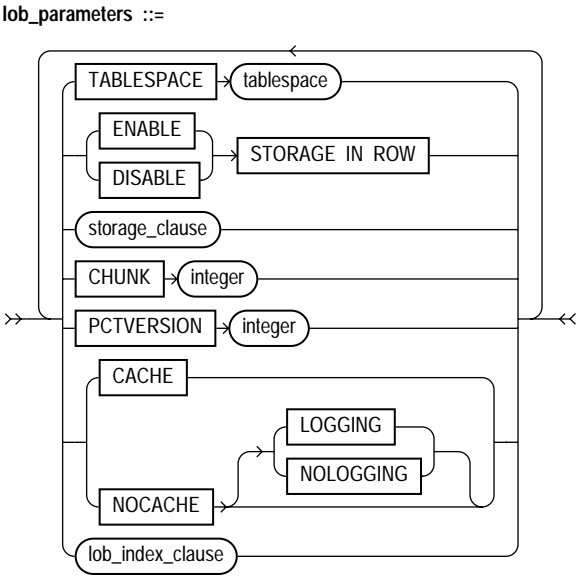


index\_organized\_overflow\_clause ::=

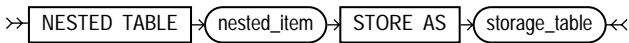


LOB\_storage\_clause ::=

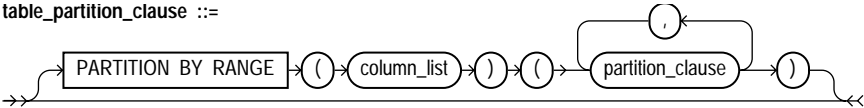




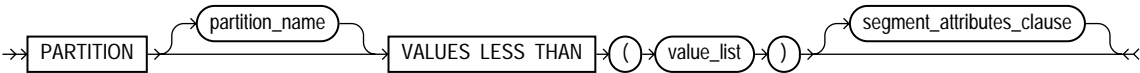
nested\_table\_storage\_clause ::=



table\_partition\_clause ::=



partition\_clause ::=



disable\_clause:DISABLE 句（4-375 ページ）を参照。

enable\_clause:ENABLE 句（4-414 ページ）を参照。

parallel\_clause:PARALLEL 句（4-462 ページ）を参照。

storage\_clause:STORAGE 句（4-518 ページ）を参照。

subquery:「副問合せ」（4-525 ページ）を参照。

キーワードとパラメータ

<i>schema</i>	表を定義するスキーマを指定します。 <i>schema</i> を省略すると、自スキーマ内に表が作成されます。
<i>table</i>	作成する表（またはオブジェクト表）の名前を指定します。クラスタ化表またはオブジェクト表をパーティション化される <i>table</i> に指定することはできません。
OBJ OF <i>object_type</i>	<i>object_type</i> 型のオブジェクト表を明示的に作成します。オブジェクト表の各列は <i>object_type</i> 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。 <i>schema</i> を省略すると、オブジェクト表が自スキーマ内に作成されます。オブジェクト作成の詳細は、CREATE TYPE（4-342 ページ）を参照してください。「オブジェクト表」（4-321 ページ）も参照してください。
<i>column</i>	表の列の名前を指定します。1 つの表に最高 1000 列まで指定できます。AS 副問合せ句を使用する場合に限り、列定義を省略できます。「LOB 列の例」（4-320 ページ）を参照してください。
OBJ attribute	オブジェクト内の項目の修飾した列名を指定します。

<i>datatype</i>	<p>列のデータ型を指定します。Oracle 提供のデータ型は、「データ型」(2-5 ページ) に定義されています。文に参照整合性制約の外部キーの一部として列を指定している場合に限って、列のデータ型を省略できます。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。</p> <p><b>OBJ</b> オブジェクト型および <i>REFobject_type</i>、<i>VARRAY</i>、ネストした表は、有効なデータ型です。「REF」(4-322 ページ) を参照してください。</p>
DEFAULT	<p>後で指定する INSERT 文で列の値を省略する場合に、その列に割り当てる値を指定します。式のデータ型は、列のデータ型と一致しなければなりません。列には、この式が入る長さが必要です。<i>expr</i> の構文については、「式」(3-73 ページ) を参照してください。DEFAULT 式には、他の列の参照、または CURRVAL、NEXTVAL、LEVEL、ROWNUM の各疑似列の参照、完全に指定されていない日付定数の参照を指定することはできません。</p>
<i>column_ref_clause</i>	<p>REF 型の列の詳細を指定します。</p> <p><b>OBJ</b> WITH ROWID <i>column</i> または <i>attribute</i> 内に ROWID と REF 値を格納します。ROWID とともに REF 値を格納すると、参照解除操作のパフォーマンスが向上しますが、より多くの領域が必要になります。REF 値のデフォルトでは、ROWID は格納されません。</p> <p><b>OBJ</b> SCOPE IS <i>scope_table_name</i> 列の REF 値の有効範囲を <i>scope_table_name</i> に制限します。列の REF 値は、句の中で指定したオブジェクト表から得られた REF 値でなければなりません。各 REF 列につき有効範囲表を 1 つだけ指定できます。</p> <p><i>scope_table_name</i> には、(REF 列と同じ型の) オブジェクト・インスタンスを格納するオブジェクト表の名前を指定します。REF 列の値は、有効範囲表内のオブジェクトを示します。</p> <p>作成する表に対する SELECT 権限または SELECT ANY TABLE システム権限が必要です。</p>
<i>column_constraint</i>	<p>整合性制約を列定義の一部として定義します。CONSTRAINT 句 (4-185 ページ) にある <i>column_constraint</i> の構文の説明を参照してください。</p>
<i>table_constraint</i>	<p>整合性制約を表定義の一部として定義します。CONSTRAINT 句 (4-185 ページ) にある <i>table_constraint</i> の構文の説明を参照してください。</p>
<i>table_ref_clause</i>	<p><b>OBJ</b> SCOPE FOR... IS... <i>ref_column</i> または <i>ref_attribute</i> 内の REF 値の有効範囲を <i>scope_table_name</i> に制限します。列または属性の REF 値は、句の中で指定したオブジェクト表から得られた REF 値でなければなりません。</p> <p><i>ref_column</i> または <i>ref_attribute</i> には、オブジェクト表の中の REF 列、またはレーショナル表のオブジェクト列内の埋込み REF 属性の名前を指定します。REF 列または属性の値は、有効範囲表内のオブジェクトを示します。</p> <p><b>OBJ</b> REF オブジェクト表の中の行に対する参照を示します。オブジェクト表またはレーショナル表 (<i>ref_column</i>) 内の REF 列名、あるいはオブジェクト列 (<i>ref_attribute</i>) 内の埋込み REF 属性を指定できます。</p>

<b>OBJ</b> <b>OIDINDEX</b>	非表示のオブジェクト識別子列の索引またはその索引の記憶域仕様を指定します。索引とその記憶域仕様の両方を指定する場合があります。 <i>index</i> 、 <i>storage_specification</i> のどちらかを必ず指定してください。
	<b>OBJ</b> <i>index</i> 非表示のオブジェクト識別子列の索引の名前を指定します。指定がない場合は、名前が自動生成されます。
<i>physical_attributes_clause</i>	PCTFREE、PCTUSED、INITRANS、MAXTRANS の各パラメータの値および表の記憶特性を指定します。  注意：非パーティション表の場合、指定した各パラメータと記憶特性は、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は、パラメータや記憶特性として指定した値は、このコマンドの PARTITION 句で同じ値を明示的に指定変更しないかぎり、CREATE 文（およびその後の各 ALTER TABLE ADD PARTITION 文）に指定したすべてのパーティションに関連付けられているセグメントのデフォルトの物理属性となります。
PCTFREE	表、オブジェクト表の <b>OIDINDEX</b> 、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に備えて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 から 99 までの整数でなければなりません。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 を指定すると、既存の行に加えられる更新に備えて各ブロックの 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することができます。  PCTFREE は、PARTITION 記述句の中での PCTFREE の機能と、クラスタおよび索引、スナップショット、スナップショット・ログの作成と変更を行うコマンドの中での 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	使用領域のうち、表、オブジェクト表 <b>OIDINDEX</b> 、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 1 ～ 99 までの正の整数で指定し、デフォルト値は 40 です。  PCTUSED は、PARTITION 記述句の中でも、クラスタおよびスナップショット、スナップショット・ログの作成と変更を行う各コマンドの中でも、同じ機能を果たします。  PCTUSED は、索引構成表 (ORGANIZATION INDEX) には無効な表記憶特性です。  PCTFREE と PCTUSED を合計した値を必ず 100 未満にします。PCTFREE と PCTUSED の両方を指定すると、表の中の領域をより効率的に利用できます。PCTUSED と PCTFREE の各値によるパフォーマンス上の効果については、『Oracle8 Server チューニング』を参照してください。
INITRANS	表、オブジェクト表 <b>OIDINDEX</b> 、パーティション、LOB の索引セグメント、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。この値は 1 ～ 255 までの範囲内であり、デフォルト値は 1 となります。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、トランザクション・エントリのサイズは、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。

	<p>このパラメータを指定すると、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にオーバーヘッドがかからないようになります。</p> <p>INITRANS パラメータの用途は、PARTITION 記述句の中でも、クラスタおよび索引、スナップショット、スナップショット・ログの中でも、表の場合と同じです。クラスタまたは索引の場合、INITRANS の最小値とデフォルト値は 1 ではなく 2 です。</p>
MAXTRANS	<p>表、オブジェクト表 OIDINDEX、パーティション、LOB の索引セグメント、索引構成表オーバーフロー・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、変更せずデフォルトのまま使用してください。</p> <p>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まったくなくなるまで、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。</p> <p>MAXTRANS パラメータの用途は、PARTITION 記述句、およびクラスタ、スナップショット、スナップショット・ログの中でも、表の場合と同じです。</p>
<i>storage_clause</i>	<p>表またはオブジェクト表 OIDINDEX、パーティション、LOB の記憶領域、LOB の索引セグメント、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します。この句は、大規模な表のパフォーマンスに影響を及ぼします。記憶領域は、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。STORAGE 句 (4-518 ページ) を参照してください。</p>
TABLESPACE	<p>Oracle が表またはオブジェクト表 OIDINDEX、パーティション、LOB の記憶領域、LOB の索引セグメント、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。このオプションを省略すると、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。</p> <p>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は、TABLESPACE に指定する値は、PARTITION 記述句に TABLESPACE を指定しないかぎり、CREATE 文 (およびその後の各 ALTER TABLE ADD PARTITION 文) で指定し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。</p>

LOGGING/  
NOLOGGING

表（および制約のために必要な索引）またはパーティション、LOB の記憶特性の作成を REDO ログ・ファイルに記録するかどうかを指定します。表またはパーティション、LOB の記憶域に対して次に実行される Direct Loader(SQL\*Loader) 操作とダイレクト・ロード INSERT 操作を記録する (LOGGING) か記録しない (NOLOGGING) かも指定します。

LOGGING/NOLOGGING 句を省略すると、表または表パーティション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 の場合は、LOGGING/NOLOGGING 句を省略すると、条件に応じて次の値が使用されます。

- CACHE を指定した場合は、LOGGING が使用される（CACHE NOLOGGING は指定できないためです）。
- これ以外の場合は、LOB が存在する表領域のロギング属性がデフォルトのロギング属性として使用されます。

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は、指定したロギング属性値は、PARTITION 記述句に LOGGING/NOLOGGING を指定しないかぎり、CREATE 文（およびその後の各 ALTER TABLE ADD PARTITION 文）で指定し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。

NOLOGGING モードでのデータ変更では、最低限のログ（新しいエクステントに無効のマークを付けることとディクショナリの変更を記録すること）しか記録されません。メディア回復中に NOLOGGING が適用されると、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します。このため、消失が許されない表の場合は、NOLOGGING 操作の後にバックアップをとる必要があります。

NOLOGGING モードでの操作作用に生成される REDO ログのサイズは、LOGGING オプションの設定時に生成されるログよりかなり小さくなります。

データベースを ARCHIVELOG モードで実行する場合、LOGGING 操作の前に行ったバックアップからのメディア回復によって、表は復元されます。ただし、NOLOGGING 操作の前に行われるバックアップからのメディア回復では、表は復元されません。

表のロギング属性は、その索引の属性から独立しています。

NOLOGGING は、索引構成表の作成時に無効なキーワードです。

LOGGING オプションおよびパラレル DML の詳細は、『Oracle8 Server 概要』および『Oracle8 Server 管理者ガイド』を参照してください。

**注意：**Oracle の今後のバージョンでは、RECOVERABLE オプションのかわりに LOGGING キーワードを使用します。RECOVERABLE は、Oracle の今後のバージョンでも非パーティション表作成時の有効キーワードとして使用できますが、使用はお勧めしません。

ORGANIZATION  
INDEX

指定の *table* を索引構成表として作成するように指定します。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。「索引構成表」（4-320 ページ）を参照してください。

ORGANIZATION  
HEAP

構成する *table* のデータ行を特定順序で格納しないことを指定します。これはデフォルトです。



<i>index_organized_ table_clause</i>	PCTTHRESHOLD <i>integer</i>	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。行の中で指定のしきい値を超える部分があれば、その領域に格納されます。PCTTHRESHOLD は、0 ～ 50 までの値でなければなりません。
	OVERFLOW	指定したしきい値を超える索引構成表データ行を、この句に指定するデータ・セグメントに格納するように指定します。OVERFLOW の指定がない場合、PCTTHRESHOLD 限界を超える行は受け付けられません。
	INCLUDING <i>column_name</i>	索引構成表の各行を索引部分とオーバフロー部分に分ける起点となる列を指定します。 <i>column_name</i> に続く列はすべて、オーバフロー・データ・セグメントに保存されます。 <i>column_name</i> は、最後の主キー列の名前か、または非主キー列の名前です。
RECOVERABLE	使用すべきでないオプション。RECOVERABLE は、パーティション表または LOB の記憶特性の作成時には無効なキーワードです。	
UNRECOVERABLE	<p>使用すべきでないオプション。表の作成（および制約のために必要な索引）を REDO ログ・ファイルに記録しないことを指定します。</p> <p>このキーワードの指定には、必ず AS 副問合せ句を使用します。UNRECOVERABLE は、パーティション表または索引構成表の作成時には無効なキーワードです。</p> <p>注意：Oracle の今後のバージョンでは、RECOVERABLE オプションのかわりに LOGGING キーワードを使用します。RECOVERABLE は、Oracle の今後のバージョンでも非パーティション表作成時の有効キーワードとして使用できますが、使用はお勧めしません。</p>	
LOB_storage_clause	LOB	LOB 記憶特性を指定します。LOB の詳細は、『Oracle8 Server アプリケーション開発者ガイド』を参照してください。
	<i>lob_item</i>	表とは異なる表領域および記憶特性を明示的に定義する LOB 列の名前または LOB のオブジェクト属性を指定します。
STORE AS	<i>lob_segname</i>	LOB のデータ・セグメントの名前を指定します。 <i>lob_item</i> を 2 つ以上指定している場合、 <i>lob_segname</i> は指定できません。
<i>lob_parameters</i>	ENABLE STORAGE IN ROW	LOB 値の長さが約 4000 バイトからシステム制御情報を引いた値に満たない場合、LOB を行内（インライン）に格納することを指定します。これはデフォルトです。
	DISABLE STORAGE IN ROW	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。
	<p>LOB 値が格納されている場所にかかわらず、LOB ロケータは常に行内に格納されます。STORAGE IN ROW の値は、いったん設定すると変更できません。</p>	

	CHUNK <i>integer</i>	<p>LOB の操作用に割り当てる記憶領域（バイト数）を指定します。<i>integer</i> 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 <i>integer</i> 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(32K) 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。</p> <p>注意：CHUNK の値は、INITIAL と NEXT（デフォルト値か記憶域句で指定した値のどちらか）の両方の値以下でなければなりません。CHUNK の値が INITIAL か NEXT のどちらかの値を超えると、エラーが戻されます。</p>
	PCTVERSION <i>integer</i>	<p>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限の記憶領域の割合（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% が使用されるまで旧バージョンの LOB データが上書きされないことを意味します。</p>
	INDEX <i>lob_index_name</i>	<p>LOB の索引セグメントの名前を指定します。<i>lob_item</i> を関連する <i>lob_item</i> リストに複数指定する場合、<i>lob_index_name</i> は指定できません。ALTER INDEX 文では LOB 索引を変更できません。ALTER TABLE 文を使用した場合にだけ、LOB 索引の仕様部を変更できます（ALTER TABLE（4-105 ページ）参照）。</p> <p>また、ユーザーは LOB 索引を削除できません。LOB 索引は、システムによって作成され、メンテナンスされるシステム索引です。</p>
● NESTED TABLE ... STORE AS ...		<p>すべての <i>nested_item</i> 値を持った行が存在する格納表の名前として <i>storage_table</i> を指定します。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。「ネストした表の格納」（4-322 ページ）を参照してください。</p> <ul style="list-style-type: none"><li>• <i>nested_item</i> には、ネストした表の型を持つ列または列修飾属性の名前を指定します。</li><li>• <i>storage_table</i> には、格納表の名前を指定します。格納表は、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。</li></ul>
CLUSTER		<p>表がクラスタの一部となるように指定します。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。</p> <p>クラスタ・キー内の各列ごとに表から 1 つの列を指定します。列の突合せは、名前ではなく位置で行われます。</p> <p>クラスタ化表では、クラスタの領域割当てが使われるため、CLUSTER オプションでは、PCTFREE、PCTUSED、INITRANS、MAXTRANS の各パラメータおよび TABLESPACE オプション、STORAGE 句は指定しないでください。</p> <p>● オブジェクト表は、クラスタの一部として指定することはできません。</p>
<i>table_partitioning_clause</i> :	PARTITION BY RANGE	<p><i>column_list</i> の値の範囲で表をパーティション化することを指定します。「パーティション表」（4-321 ページ）を参照してください。</p>

<i>column_list</i>	行がどのパーティションに属するかを判断するのに使われる、列の順序リストを指定します。 <i>column_list</i> に指定できるのは 16 列以下です。 <i>column_list</i> には、ROWID 疑似列、あるいはデータ型 ROWID および LONG の列は指定できません。
<b>PARTITION</b> <i>partition_name</i>	物理パーティション句を指定します。 <i>partition_name</i> を省略すると、名前はパーティションに対して <b>SYS_Pn</b> の形で生成されます。 <i>partition_name</i> は、スキーマ・オブジェクトの命名規則に従っていなければなりません。また、各部分は「スキーマ・オブジェクトの命名規則」（2-43 ページ）の説明どおりに指定しなければなりません。
<b>VALUES LESS THAN</b>	現在のパーティション・バウンド上限（その値を含めない）を指定します。
<i>value_list</i>	<p><b>PARTITION BY RANGE</b> 句内の <i>column_list</i> に対応するリテラル値の順序リストを指定します。<i>value_list</i> 内のリテラル値には、かわりにキーワード <b>MAXVALUE</b> を指定することができます。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。</p> <p>最高のパーティション・バウンドに <b>MAXVALUE</b> 以外の値を指定すると、表に暗黙の整合性制約が課せられます。パーティション・バウンドの詳細は、『Oracle8 Server 概要』を参照してください。</p>
<i>parallel_clause</i>	<p>表作成時の並列度を指定します。また、一度作成された表に対する問合せの並列度のデフォルトを指定します。</p> <p>索引構成表の作成時には無効なオプションです。詳細は、<b>PARALLEL</b> 句（4-462 ページ）を参照してください。</p>
<i>enable_clause</i>	整合性制約を使用可能にします。 <b>ENABLE</b> 句（4-414 ページ）を参照してください。
<i>disable_clause</i>	<p>整合性制約を使用禁止にします。<b>DISABLE</b> 句（4-375 ページ）を参照してください。</p> <p><b>CREATE TABLE</b> 文の <b>ENABLE</b> 句および <b>DISABLE</b> 句に指定する制約は、その文中で定義しなければなりません。<b>CONSTRAINT</b> 句の <b>ENABLE</b> キーワードと <b>DISABLE</b> キーワードを使用して、制約を使用可能または使用禁止にすることもできます。制約を定義しても明示的に使用可能または使用禁止にしていない場合、デフォルトで制約は使用可能になります。</p> <p><b>CREATE TABLE</b> 文の <b>ENABLE</b> 句と <b>DISABLE</b> 句を使用して、トリガーを使用可能および使用禁止にすることはできません。</p>
<i>AS subquery</i>	<p>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。「副問合せ」（4-525 ページ）を参照してください。</p> <p><b>注意：</b> この副問合せは、オーバーフローがある索引構成表ではサポートされません。</p> <p>表の中の列の数は、副問合せの中の式の数と等しくなければなりません。列定義では、列名、デフォルト値、整合性制約だけを指定できます。データ型は指定できません。データ型とデータ長は、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。</p>

- ・ 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る。
- ・ CREATE TABLE 文には、AS 句と参照整合性制約定義の両方は指定できない。
- ・ CREATE TABLE 文に AS 句と、EXCEPTIONS オプションを付けた CONSTRAINT 句または ENABLE 句の両方を指定すると、EXCEPTIONS オプションは無視される。制約に違反する行があると、表が作成されずエラー・メッセージが戻されます。

副問合せの中のすべての式が、式ではなく列である場合、表定義から列を完全に省略できます。この場合には、表の列名は副問合せの列の名前と同一になります。

**OBJ** オブジェクト表の場合、副問合せには表の型に対応する 1 つの式、表の型の最上位属性の数のどちらかを設定できます。

#### CACHE

データが頻繁にアクセスされる場合に、全表走査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。このオプションは、小さな参照表の場合に便利です。

LOB 記憶句内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、メモリーに LOB データ値が事前に割り当てられ、保持されるように指定します。

索引構成表の作成時には無効なキーワードです。

#### NOCACHE

データが頻繁にアクセスされない場合に、全表走査の実行時にこの表用に取り出される各ブロックを、バッファ・キャッシュ内の LRU リストの最低使用頻度側に入れることを指定します。LOB の場合、LOB 値はバッファ・キャッシュには入れられない場合と、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場合があります。

これは、索引構成表の作成時以外のデフォルト動作です。索引構成表の作成時には無効なキーワードです。

LOB 記憶句内のパラメータとして NOCACHE を使用する場合は、LOB 値が事前にメモリー内に割り当てられないことを指定します。これは LOB 記憶領域のデフォルトです。

---

## 例

問合せを指定しないと、データを含まない表が作成されます。INSERT コマンドを使用すると、表に行を追加できます。

表を作成した後、ALTER TABLE コマンドで ADD 句を指定すると、追加する列、パーティション、整合性制約を定義できます。ALTER TABLE コマンドで 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。整合性制約を修正する場合は、必ずその制約を削除し、再定義する必要があります。

**例 1.** 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。HIRDEDATE 列では、データ型が DATE、デフォルト値 SYSDATE を持っています。

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。

**例 2.** 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
        PCTINCREASE 5);

```

この文では GRADE 列に PRIMARY KEY 制約を定義し、この制約を施行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。

整合性制約の定義の例は、CONSTRAINT 句 (4-185 ページ) を参照してください。整合性制約を使用可能および使用禁止にする例は、ENABLE 句 (4-414 ページ) および DISABLE 句 (4-375 ページ) を参照してください。

**例 3.** パラレル問合せを使用する場合、部門 10 の従業員だけを対象として、EMP 表と同じ列を持つ表を最も速く作成するには、次のようなコマンドを発行します。

```

CREATE TABLE emp_tmp
NOLOGGING

```

```
PARALLEL (DEGREE 3)
AS SELECT * FROM emp WHERE deptno = 10;
```

並列性を使用すると、表を作成するために3つのプロセスが使用されるので、表作成が高速化します。表が作成されると、表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。

## LOB 列の例

次の文は、2つのLOB列がある表LOB\_TABを作成し、LOB記憶特性を指定する例です。

```
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 4
   NOCACHE LOGGING
   INDEX (TABLESPACE lob_index_ts
   STORAGE (INITIAL 256 NEXT 256)
   )
);
```

## 索引構成表

索引構成表は、主キーに基づいてソートされたデータを保持する特殊な表であり、主キーを基にしたアクセスおよび操作に最も適しています。

索引構成表は次のうちいずれかです。

- **CREATE INDEX** コマンド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、表用に主キーをクラスタ・キーにマップする **CREATE CLUSTER** コマンドを使用して作成されます

索引構成表が他の種類の表と異なる点は、表の行が主キーに基づいて構築されたB\*ツリー索引により管理されることです。ただし、索引行には、主キー列の値と、対応する行に関連付けられている非キー列の値の両方が入ります。

索引構成表には主キーを指定する必要があります。この主キーは行を一意に識別するからです。索引構成表の行に直接アクセスするには、**ROWID** のかわりに主キーを使います。

**例.** 次の文は、索引構成表を作成する例です。

```
CREATE TABLE docindex
( token CHAR(20),
  doc_oid INTEGER,
  token_frequency SMALLINT,
```

```

token_occurrence_data VARCHAR(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

パーティション表は、同じ論理属性を持ついくつかの部分から構成されています。たとえば、すべてのパーティションは同じ列定義および制約定義を共有します。

パーティションが1つしかない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つしかないパーティション表と、非パーティション表には違いがある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。

例. 次の文は、3つのパーティションを持つ表の作成例です。

```

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);

```

パーティション表のメンテナンス操作については、『Oracle8 Server 管理者ガイド』を参照してください。

## OBJ オブジェクト表

オブジェクトにオブジェクト識別子が自動的に割り当てられるようにするには、そのオブジェクトをオブジェクト表という特殊な表に常駐させる必要があります。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。REF の使用の詳細は、「ユーザー定義型」(2-20 ページ)、「ユーザー関数」(3-57 ページ)、「式」(3-73 ページ)、CREATE TYPE (4-342 ページ)、および『Oracle8 Server 管理者ガイド』を参照してください。

オブジェクト表の各列は、対応する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。

例 1. たとえば、オブジェクト型 DEPT\_T は次のようになります。

```
CREATE TYPE dept_t AS OBJECT
(  dname VARCHAR2(100),
   address VARCHAR2(200) );
```

オブジェクト表 DEPT は、型 DEPT\_T の部門オブジェクトを持ちます。

```
CREATE TABLE dept OF dept_t;
```

例 2. **OBJ** 次の文は、ユーザー定義のオブジェクト型 SALESREP\_T を持つオブジェクト表 SALESREPS を作成する例です。

```
CREATE OR REPLACE TYPE salesrep_t AS OBJECT
(  repId NUMBER,
   repName VARCHAR2(64));
CREATE TABLE salesreps OF salesrep_t;
```

## **OBJ** ネストした表の格納

TABLE 型の列を持つ表を作成すると、ネストした表の列ごとに格納表が作成されます。この格納表は、(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。この格納表には、この表の作成の基になった列のネスト表が格納されます。

格納表に対しては問合せや DML 文を直接実行できませんが、ALTER TABLE 文に格納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。ネストした表の列の記憶特性の変更方法については、ALTER TABLE (4-105 ページ) を参照してください。

例. 次の文は、ネストした表の列 PROJECTS を持つ関係表 EMPLOYEE の作成例です。

```
CREATE TABLE employee (empno NUMBER, name CHAR(31),
  projects PROJ_TABLE_TYPE)
  NESTED TABLE projects STORE AS nested_proj_table;
```

## **OBJ** REF

REF 値は、オブジェクト表内の行に対する参照です。表では、最上位の REF 列または REF 属性をオブジェクト型列の中に埋め込むことができます。一般に、表に REF 列がある場合、その列内の各 REF 値で他のオブジェクト表内の行を参照できます。SCOPE 句は、参照の有効範囲を 1 つの表に制限します。

たとえば、ある組織内のすべての部署を記憶するオブジェクト表 DEPT を作成すると、各従業員の勤務部署を指し示す REF 列 (E\_DEPT) が入った表 EMP を作成できます。どの従業員も DEPT 表に記憶されているどこかの部署で勤務しているため、EMP の E\_DEPT 列に有効範囲句を指定して、参照の有効範囲を DEPT 表に制限できます。

参照解除操作によって問合せのパフォーマンスを向上させ、さらに有効範囲句を使って REF 値に必要な記憶領域の量を減らすことができます。SCOPE 句は参照制約と同じ意味を持っていないことに注意してください。参照制約では、ダングリング参照はできません。また、



参照制約では、参照の有効範囲が必ずしも 1 つの表に制限されるわけではありません（同じ外部キーに対して複数の参照制約を指定し、各参照制約がそれぞれ異なる表を指すようにすることができます。）

REF 値は、ROWID を付けて格納することも ROWID なしで格納することもできます。ROWID を付けて REF 値を格納すると、参照解除のパフォーマンスが向上しますが、より多くの領域が必要になります。デフォルト動作は、ROWID なしで REF 値を格納することです。

CREATE TABLE 文を使用して、REF 句をネストした表の REF 列に指定することはできません。ネストした表の REF 列に REF 句を指定するには、ALTER TABLE を使って、ネストした表の格納表を変更します。

例 . 次の文は、オブジェクト型 DEPT\_T とオブジェクト表 DEPT を作成し、すべての部署のインスタンスを格納する例です。この文を実行すると、有効範囲付きの 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 );
```

## ● オブジェクト型列の制約

オブジェクト型列のスカラー属性に UNIQUE 制約および PRIMARY KEY 制約、REFERENCES 制約を作成できます。また、オブジェクト型列の NOT NULL 制約、オブジェクト型列またはオブジェクト型列の属性を参照する CHECK 制約も作成できます。

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

---

```
CREATE TABLE persons OF person
( homeaddress NOT NULL
  UNIQUE (homeaddress.phone),
  CHECK (homeaddress.zip IS NOT NULL),
  CHECK (homeaddress.city <> 'San Francisco') );
```

## 関連項目

CREATE TYPE (4-342 ページ)

CREATE CLUSTER (4-202 ページ)

CREATE CLUSTER (4-202 ページ)

CREATE TABLESPACE (4-325 ページ)

CREATE TYPE (4-342 ページ)

DROP TABLE (4-402 ページ)

CONSTRAINT 句 (4-185 ページ)

DISABLE 句 (4-375 ページ)

ENABLE 句 (4-414 ページ)

PARALLEL 句 (4-462 ページ)

STORAGE 句 (4-518 ページ)

# CREATE TABLESPACE

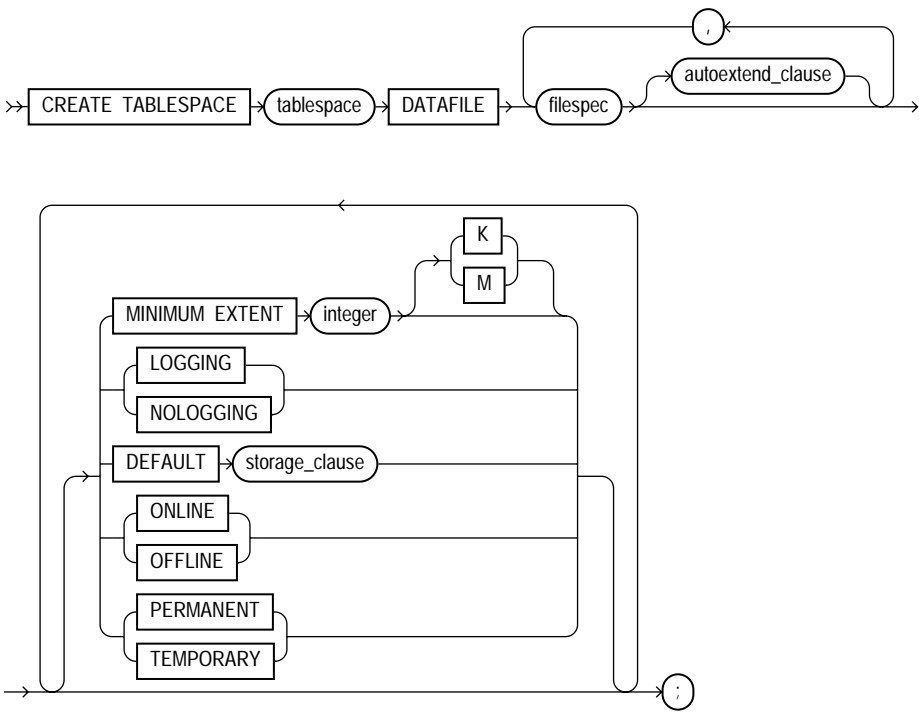
## 用途

表領域を作成します。表領域はデータベース内に割り当てられた領域で、この表領域内にスキーマ・オブジェクトが格納されます。「使用上の注意」(4-327 ページ)を参照してください。

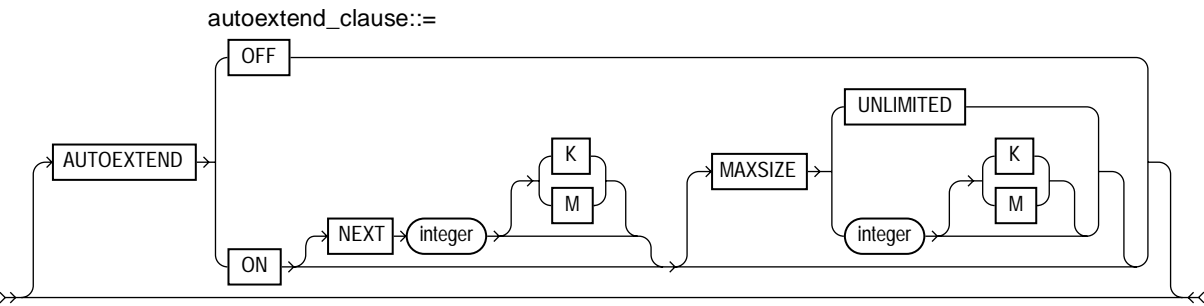
## 前提条件

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、少なくとも 2 つのロールバック・セグメントが必要です。

## 構文



`filespec`: 「Filespec」(4-428 ページ)を参照。



storage\_clause:STORAGE 句（4-518 ページ）を参照。

キーワードとパラメータ

<i>tablespace</i>	作成する表領域の名前を指定します。
DATAFILE <i>filespec</i>	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。「Filespec」（4-428 ページ）を参照してください。
<i>autoextend_clause</i>	データ・ファイルの自動拡張の使用可能と使用禁止の設定を切り替えます。 <div><div>OFF</div><div>オンになっている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。さらに ALTER TABLESPACE AUTOEXTEND コマンドを使用する場合は、NEXT および MAXSIZE に値を指定し直す必要があります。</div><div>ON</div><div>自動拡張を使用可能にします。</div><div>NEXT</div><div>エクステントがさらに必要になったとき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。</div><div>MAXSIZE</div><div>データ・ファイルに割当て可能なディスク領域の最大サイズを指定します。</div><div>UNLIMITED</div><div>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。</div></div>
MINIMUM EXTENT <i>integer</i>	表領域内のすべての使用済みエクステントおよび未使用エクステントの大きさを、 <i>integer</i> で指定したサイズの倍数 (1 以上) に統一することによって、表領域の空き領域断片化を制御します。MINIMUM EXTENT を使用した領域の断片化制御については、『Oracle8 Server 管理者ガイド』を参照してください。
LOGGING/ NOLOGGING	表領域内のすべての表および索引、パーティションのデフォルトのログギング属性を指定します。LOGGING がデフォルトです。

表および索引、パーティションのレベルでの指定のログを記録することによって、表領域レベルのロギング属性を上書きできます。

次に示す操作でだけ、NOLOGGING モードがサポートされます。

DML:

- 直接 INSERT( シリアルまたはパラレル ) 操作
- Direct Loader(SQL\*Loader) 操作

DDL:

- CREATE TABLE ... AS SELECT
- CREATE INDEX
- ALTER INDEX ... REBUILD
- ALTER INDEX ... REBUILD PARTITION
- ALTER INDEX ... SPLIT PARTITION
- ALTER TABLE ... SPLIT PARTITION
- ALTER TABLE ... MOVE PARTITION

NOLOGGING モードでのデータ変更では、最低限のログ（新しいエクステンツに無効のマークを付けることとディクショナリの変更を記録すること）しか記録されません。メディア回復中に NOLOGGING が適用されると、REDO データのログ記録が中断されるため、エクステンツ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、消失が許され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とる必要があります。

DEFAULT <i>storage_</i> <i>clause</i>	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。記憶領域パラメータについては、STORAGE 句（4-518 ページ）を参照してください。
ONLINE	表領域に対するアクセス権を付与されているユーザーに対して、作成直後に表領域を使用可能にします。
OFFLINE	作成直後に表領域を使用禁止にします。  ONLINE と OFFLINE の両オプションを省略すると、デフォルトにより表領域はオンラインで作成されます。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とオフラインのいずれであることを示します。
PERMANENT	表領域を永続オブジェクトの格納に使うように指定します。これはデフォルトです。
TEMPORARY	表領域を一時オブジェクトの格納にだけ使うように指定します。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。

## 使用上の注意

表領域はデータベース内に割り当てられた領域で、次のいずれかのセグメントが格納されます。

- データ・セグメント

- 索引セグメント
- ロールバック・セグメント
- 一時セグメント

すべてのデータベースは、最低 1 つの表領域 **SYSTEM** を持っています。この表領域は、ユーザーがデータベースを作成すると Oracle によって自動的に作成されます。

表領域は、最初は読み書き両用として作成されます。表領域を作成した後、続いて **ALTER TABLESPACE** コマンド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、読取り専用を設定したりできます。

多くのスキーマ・オブジェクトは、データベース内の領域を占有するセグメントに対応付けられています。これらのオブジェクトは表領域に格納されます。このようなオブジェクトを作成するユーザーは、オブジェクトを格納する表領域を任意に指定できます。オブジェクトを定義するスキーマの所有者は、必ずオブジェクトの表領域上に割当て制限を持っていなければなりません。ユーザーに対する表領域の割当て制限の割当てには、**CREATE USER** コマンドまたは **ALTER USER** コマンドの **QUOTA** 句を使います。

---

---

**警告：** ロー・デバイスをサポートするオペレーティング・システムでは、**CREATE TABLESPACE** コマンドでデータ・ファイルとしてロー・デバイスを指定すると、**STORAGE** 句の **REUSE** キーワードの意味がなくなることにご注意ください。このようなコマンドは **REUSE** を指定しなくても、必ず正常に実行されます。

---

---

**例 1.** 次のコマンドは、データ・ファイルが 1 つある **TABSPACE\_2** という名前の表領域の作成例です。

```
CREATE TABLESPACE tabspace_2
  DATAFILE 'diska:tabspace_file2.dat' SIZE 20M
  DEFAULT STORAGE (INITIAL 10K NEXT 50K
                   MINEXTENTS 1 MAXEXTENTS 999
                   PCTINCREASE 10)
  ONLINE;
```

**例 2.** 次のコマンドは、データ・ファイルが 1 つある **TABSPACE\_3** という名前の表領域の作成例です。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_5
  DATAFILE 'diskb:tabspace_file3.dat' SIZE 500K REUSE
  AUTOEXTEND ON NEXT 500K MAXSIZE 10M;
```

**例 3.** 次のコマンドは、データ・ファイルが 1 つある **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 64K の倍数として割り当てる例です。

```
CREATE TABLESPACE tabspace_3
  DATAFILE 'tabspace_file5.dbf' SIZE 2M
  MINIMUM EXTENT 64K
  DEFAULT STORAGE (INITIAL 128K NEXT 128K)
  LOGGING;
```

## 関連項目

[ALTER TABLESPACE \(4-131 ページ\)](#)

[DROP TABLESPACE \(4-404 ページ\)](#)

[「Filespec」 \(4-428 ページ\)](#)

# CREATE TRIGGER

## 用途

データベース・トリガーを作成し使用可能にします。データベース・トリガーは表に対応付けられる格納された PL/SQL ブロックです。指定した SQL 文を表に対して発行すると、自動的にトリガーが実行されます。「トリガーの使用方法」(4-333 ページ)を参照してください。

---

---

**注意：** コマンドと句の説明で **OBJ** の印が前に付いている箇所は、Oracle Object Option がデータベース・サーバーにインストールされている場合にだけ読んでください。

---

---

## 前提条件

トリガーを作成するには、SYS ユーザーが SQL スクリプトの DBMSSTD<sub>X</sub>.SQL を実行しなければなりません。このスクリプトの実際の名前と位置は、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。

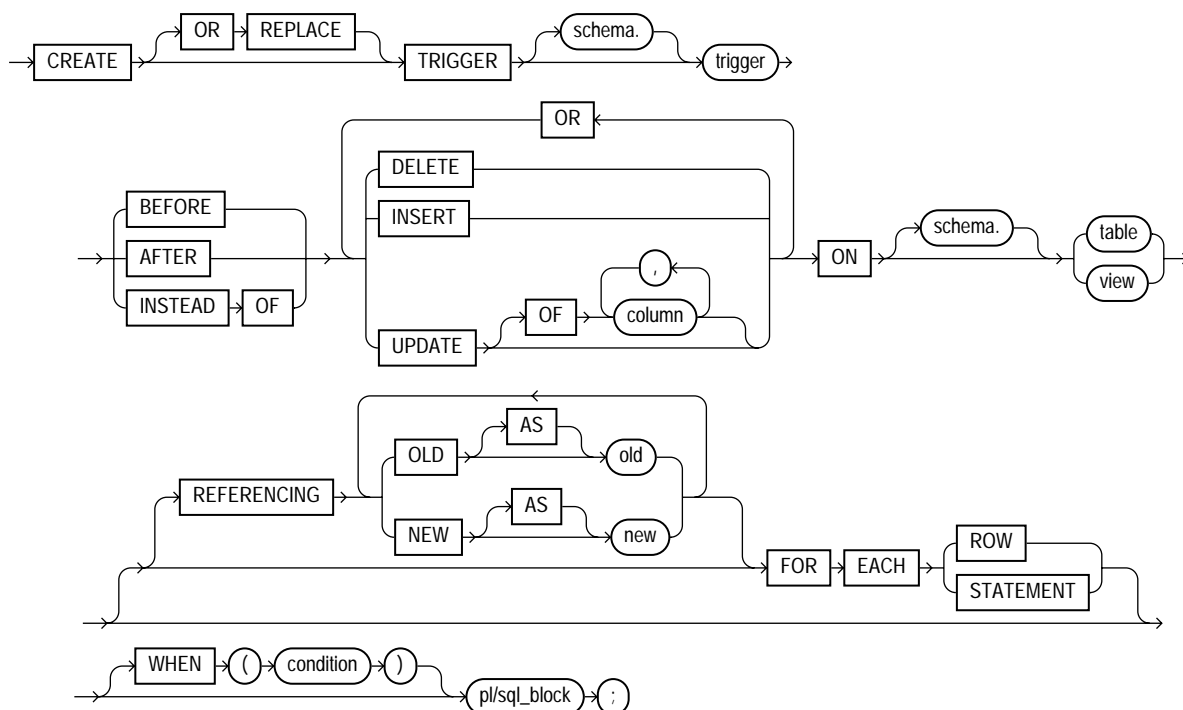
この文を発行するには、次のいずれかのシステム権限が必要です。

- |                    |   |
|--------------------|---|
| CREATE TRIGGER     | このシステム権限がある場合は、ユーザーが自スキーマ内の表に対するトリガーを自スキーマ内に作成できます。     |
| CREATE ANY TRIGGER | このシステム権限がある場合は、任意のスキーマ内の表に対するトリガーを任意のユーザーのスキーマ内に作成できます。 |

トリガーが SQL 文を発行したり、プロシージャや関数をコールしたりする場合、そのトリガーが持つことになるスキーマの所有者は、これらの操作を行うために必要な権限を持っていなければなりません。これらの権限はロールを介して付与するのではなく、所有者に直接付与しなければなりません。



## 構文



## キーワードとパラメータ

OR REPLACE	トリガーがすでに存在している場合に、そのトリガーを再作成します。このオプションを指定すると、既存のトリガーの定義を削除せずに変更できます。
<i>schema</i>	作成するトリガーを含むスキーマを指定します。 <i>schema</i> を指定しないと、自スキーマにトリガーが作成されます。
<i>trigger</i>	作成するトリガーの名前を指定します。「条件述語」(4-334 ページ) および「トリガーの構成要素」(4-335 ページ)、「トリガー・タイプ」(4-336 ページ)を参照してください。
BEFORE	トリガーを起動する文が実行される前に、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起動されます。  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。
AFTER	トリガーを起動する文が実行された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起動されます。「スナップショット・ログ・トリガー」(4-337 ページ)を参照してください。

	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。
INSTEAD OF	トリガーを起動する文が実行されず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、各行についてアクティブになります。「INSTEAD OF トリガー」(4-339 ページ)を参照してください。  INSTEAD OF は、ビューにだけ使用できるオプションです。INSTEAD OF トリガーは表には指定できません。
DELETE	DELETE 文で表から行を削除すると、トリガーが起動されます。
INSERT	INSERT 文で表に行を追加すると、トリガーが起動されます。
UPDATE	OF 句で指定した列のうちのいずれかの値を UPDATE 文で変更すると、トリガーが起動されます。OF 句を省略すると、UPDATE 文で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。  OF 句は INSTEAD OF トリガーとともに指定することはできません。UPDATE 文でビューの任意の列の値を変更すると、INSTEAD OF トリガーが起動されます。  OF 句にはネストした表および LOB 列は指定できません。「ユーザー定義の型および LOB、REF 列」(4-340 ページ)を参照してください。
ON	トリガー作成の対象のスキーマと、表またはビューの名前を指定します。トリガー作成の対象は、次のいずれかです。 <ul style="list-style-type: none"><li>• 表</li><li>• <b>OBJ</b> オブジェクト表</li><li>• ビュー</li><li>• <b>OBJ</b> オブジェクト・ビュー</li></ul> <p><i>schema</i> を指定しないと、Oracle は、この表が自スキーマに存在するとみなします。トリガーは索引構成表に作成できますが、SYS スキーマ内の表には作成できません。「ユーザー定義の型および LOB、REF 列」(4-340 ページ)を参照してください。</p>
table	表またはオブジェクト表の名前を指定します。
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。
REFERENCING	相関名を指定します。相関名は、特に現在の行における新旧の値を参照する場合に、PL/SQL ブロック内と行レベル・トリガーの WHEN 句内とで使用します。デフォルトの相関名は OLD と 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。  <b>OBJ</b> 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。

FOR EACH ROW	<p>行レベル・トリガーとなるトリガーを指定します。行レベル・トリガーは、トリガーを起動する文の対象となり、かつ WHEN 句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。</p> <p>INSTEAD OF トリガー以外のトリガーを指定するときに、この句を省略すると、そのトリガーは文レベル・トリガーとなります。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 回だけ起動されます。</p> <p>INSTEAD OF トリガー文は、各行について暗黙にアクティブになります。</p>
WHEN( <i>condition</i> )	<p>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。<i>condition</i> の構文の説明は、「条件」(3-73 ページ) を参照してください。この条件には相関名を指定しなければなりません。問合せは指定できません。</p> <p>行レベル・トリガーの場合だけ、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。</p> <p>INSTEAD OF トリガー文にはトリガー制約は指定できません。</p> <p><b>❗</b> 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 列を参照できません。トリガー制約内では PL/SQL 関数およびメソッドは起動できません。</p>
<i>pl/sql_block</i>	<p>Oracle がトリガーを起動するために実行する PL/SQL ブロックです。PL/SQL ブロックの記述方法を含む PL/SQL の詳細は、『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。</p> <p><b>注意:</b> トリガーの PL/SQL ブロックには、トランザクション制御 SQL 文 (COMMIT および ROLLBACK、SAVEPOINT、SET CONSTRAINT) は指定できません。</p>

## トリガーの使用方法

トリガーは、トリガーを起動する文が発行されると、自動的に起動つまり実行されます。トリガーの用途は次のとおりです。

- 高い水準の監査機能と透過性のあるイベント・ロギングを実現する。
- 関連する列の値を自動的に生成する。
- 複雑なセキュリティ認可や業務制限に対応する。
- 非同期複製表をメンテナンスする。

上記の用途でトリガーを設計する方法の詳細は、『Oracle8 Server アプリケーション開発者ガイド』を参照してください。

既存のトリガーは次のどちらかの状態になっています。

- トリガーが使用可能になっている場合、トリガーを起動する文が発行され、トリガー制約の条件が満たされるとトリガーが起動される。
- トリガーが使用禁止になっている場合、トリガーを起動する文が発行され、トリガー制約の条件が満たされてもトリガーは起動されない。

トリガーを作成すると、そのトリガーは自動的に使用可能になります。この後、**DISABLE** オプションまたは **ENABLE** オプションを指定した **ALTER TRIGGER** コマンドまたは **ALTER TABLE** コマンドを使用して、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。

トリガーを使用可能および使用禁止にする方法の詳細は、**ALTER TRIGGER** (4-139 ページ) および **ALTER TABLE** (4-105 ページ)、**ENABLE** 句 (4-414 ページ)、**DISABLE** 句 (4-375 ページ) を参照してください。

リリース 7.3 より前のリリースでは、トリガーが起動された時点でトリガーの解析とコンパイルが行われていました。リリース 7.3 以降では、コンパイル済みのトリガーをデータ・ディクショナリに格納しておき、トリガーが起動されると、データ・ディクショナリからコンパイル済みトリガーをコールするようになりました。これにより、多数のトリガーを使用するアプリケーションでは、パフォーマンスが大きく向上します。

トリガーのコンパイル・エラーが発生すると、このトリガーは作成されますが実行時に失敗します。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・エラーのないトリガーに置き換えられるか、削除されるまでの間、トリガーを起動するすべての **DML** 文の実行が完全にブロックされます。

Oracle プリコンパイラ・プログラムの中に **CREATE TRIGGER** 文を埋め込むには、キーワード **END-EXEC** とその後に特定の言語用の埋込み **SQL** 文終了記号を記述して文を終了する必要があります。

## 条件述語

複数の **DML** 操作用のトリガーを作成する場合、そのトリガーを起動する文の型によって、トリガー本体内で条件述語を使用して特定のコード・ブロックを実行できます。条件述語は次のように評価されます。

<b>INSERTING</b>	<b>INSERT</b> 文によってトリガーが起動された場合に真が戻ります。
<b>DELETING</b>	<b>DELETE</b> 文によってトリガーが起動された場合に真が戻ります。
<b>UPDATING</b>	<b>UPDATE</b> 文によってトリガーが起動された場合に真が戻ります。
<b>UPDATING</b> ( <i>column_name</i> )	<b>UPDATE</b> 文によってトリガーが起動され、 <i>column_name</i> が更新された場合に真が戻ります。
<b>注意：</b> オブジェクト属性は <i>column_name</i> として指定できません。	

トリガー本体での条件述語の作成および使用の詳細は、『Oracle8 Server アプリケーション開発者ガイド』を参照してください。

**例.** 次の例では、条件述語を使用して、**DML** 文がトリガー **AUDIT\_TRIGGER** を起動するための情報を提供します。

```
CREATE TRIGGER audit_trigger BEFORE INSERT OR DELETE OR UPDATE
ON classified_table FOR EACH ROW
BEGIN
```

```

IF INSERTING THEN
    INSERT INTO audit_table
        VALUES (USER || ' is inserting' ||
                ' new key: ' || :new.key);
ELSIF DELETING THEN
    INSERT INTO audit_table
        VALUES (USER || ' is deleting' ||
                ' old key: ' || :old.key);
ELSIF UPDATING('FORMULA') THEN
    INSERT INTO audit_table
        VALUES (USER || ' is updating' ||
                ' old formula: ' || :old.formula ||
                ' new formula: ' || :new.formula);
ELSIF UPDATING THEN
    INSERT INTO audit_table
        VALUES (USER || ' is updating' ||
                ' old key: ' || :old.key ||
                ' new key: ' || :new.key);
END IF;
END;

```

## トリガーの構成要素

CREATE TRIGGER 文の構文として、次のトリガーの要素を指定します。

### トリガーを起動する文

トリガーを起動させる SQL 文を指定します。

DELETE	トリガーを起動するこれらのコマンドのうち少なくとも1つを必ず指定しなければなりません。最大3つまでのコマンドを指定できます。
INSERT	
UPDATE	
ON	トリガーを対応付けた表も必ず指定しなければなりません。ここに指定された表を変更する文だけが、トリガーを起動する文になります。索引構成表にトリガーを定義することもできます。

### トリガー制限

行レベル・トリガーが起動される際に必要な追加条件を指定します。この条件は、WHEN 句で指定できます。この条件は PL/SQL 条件ではなく、SQL 条件でなければなりません。

### トリガー・アクション

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。

Oracle は、トリガーを起動する文が発行されると必ず、トリガー制約の条件を評価します。条件が満たされた場合、Oracle はトリガーを起動して、トリガー・アクションを実行します。

トリガー・タイプ

タイプを指定してトリガーを作成できます。トリガーのタイプによって、次に示す事項が決まります。

- トリガーが起動されるタイミング（トリガーを起動する文の実行を基準とする）
- トリガーが起動される回数

トリガーのタイプは、CREATE TRIGGER コマンドの BEFORE および AFTER、FOR EACH ROW オプションによって異なります。上記のオプションをトリガーのタイプとして組み合わせて使用すると、4 種類のトリガーを作成できます。表 4-9 に、各タイプのトリガーおよびその特性、トリガー作成のために使用するオプションを示します。

表 4-9 トリガーのタイプ

FOR EACH オプション		
	文（省略の場合もある）	行
BEFORE オプション	BEFORE 文レベル・トリガー： トリガーを起動する文が実行される前に 1 回トリガーを起動する。	BEFORE 行レベル・トリガー：トリガーを起動する文の対象となる各行が変更される前に、1 回ずつトリガーが起動される。
AFTER オプション	AFTER 文レベル・トリガー： トリガーを起動する文が実行された後に 1 回トリガーが起動される。	AFTER 行レベル・トリガー：トリガーを起動する文の対象となる各行が変更されるたびに、トリガーが起動される。

1 つの表の場合、次の各コマンドについて上記の各タイプのトリガーを作成できます。

- DELETE
- INSERT
- UPDATE

複数のコマンドによって起動されるトリガーも作成できます。

同一の表に対する同一のコマンドについて同じタイプのトリガーを複数作成した場合、これらのトリガーが起動される順番は不確定です。同じコマンドを対象とした同じ種類のトリガーを 2 つ続けて起動することがアプリケーションで必要な場合、これら 2 つのトリガーを 1 つのトリガーに結合し、2 つのトリガー動作が目的の順序で実行されるようにしてください。

## スナップショット・ログ・トリガー

表に対するスナップショットを作成すると、その表に AFTER 行トリガーが暗黙的に作成されます。このトリガーは、INSERT 文または UPDATE 文、DELETE 文で表のデータが変更されるとスナップショット・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ないので、スナップショットの内容に影響を与えるトリガーは記述しないでください。スナップショット・ログの詳細は、CREATE SNAPSHOT LOG (4-294 ページ) を参照してください。

**例 1.** この例では、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。このトリガーにより、平日の就業時間内に限り、従業員レコードを変更できるようになります。

```
CREATE TRIGGER scott.emp_permit_changes
  BEFORE
  DELETE OR INSERT OR UPDATE
  ON scott.emp
  DECLARE
    dummy INTEGER;
  BEGIN
    /* If today is a Saturday or Sunday,
       then return an error.*/
    IF (TO_CHAR(SYSDATE, 'DY') = 'SAT' OR
        TO_CHAR(SYSDATE, 'DY') = 'SUN')
      THEN raise_application_error( -20501,
        'May not change employee table during the weekend');
    END IF;
    /* Compare today's date with the dates of all
       company holidays. If today is a company holiday,
       then return an error.*/
    SELECT COUNT(*)
      INTO dummy
      FROM company_holidays
      WHERE day = TRUNC(SYSDATE);
    IF dummy > 0
      THEN raise_application_error( -20501,
        'May not change employee table during a holiday');
    END IF;
    /*If the current time is before 8:00AM or after
       6:00PM, then return an error.
    */
    IF (TO_CHAR(SYSDATE, 'HH24') < 8 OR
        TO_CHAR(SYSDATE, 'HH24') >= 18)
      THEN raise_application_error( -20502,
        'May only change employee table during working hours');
    END IF;
  END;
```

DELETE、INSERT、UPDATE の各文が SCOTT スキーマ内の 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。

このトリガーは次の処理を実行します。

1. 現在の曜日が土曜日または日曜日の場合、従業員表は週末に変更できないことを示すメッセージを表示し、アプリケーション・エラーを戻します。
2. 会社の全社的な休日の日程表と現在の日付を比較します。
3. 現在の日付が会社の全社的な休日である場合、従業員表は休日に変更できないことを示すメッセージを表示し、アプリケーション・エラーを戻します。
4. 現在の時間が 8:00AM から 6:00PM の間でない場合、従業員表は就業時間内に限り変更できることを示すメッセージを表示し、アプリケーション・エラーを戻します。

**例 2.** この例では、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。従業員表に対して新しい従業員を追加するときや、既存の従業員の給与や職種を変更するときには必ず、このトリガーによって、従業員の給与が従業員の職種に対して設定された給与の範囲内にあることが検査されます。

```
CREATE TRIGGER scott.salary_check
BEFORE
INSERT OR UPDATE OF sal, job ON scott.emp
FOR EACH ROW
WHEN (new.job <> 'PRESIDENT')
DECLARE
    minsal NUMBER;
    maxsal NUMBER;
BEGIN
    /* Get the minimum and maximum salaries for the
       employee's job from the SAL_GUIDE table. */
    SELECT minsal, maxsal
    INTO minsal, maxsal
    FROM sal_guide
    WHERE job = :new.job;
    /* If the employee's salary is below the minimum or */
    /* above the maximum for the job, then generate an */
    /* error.*/
    IF (:new.sal < minsal OR :new.sal > maxsal)
    THEN raise_application_error( -20601,
        'Salary ' || :new.sal || ' out of range for job '
        || :new.job || ' for employee ' || :new.ename );
    END IF;
END;
```



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する前に、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 には社長の給与を検査できないトリガー制約が設定されています。新規または変更された各従業員の行がこの条件に一致する場合、トリガーは次の手順を実行します。

1. 従業員の職種に対する最低および最高の給与金額を、給与規定表で問い合わせます。
2. 従業員の給与をその最小値や最大値と比較します。
3. 従業員の給与が許容範囲内に適合しない場合、従業員の給与が従業員の職種に対して設定された範囲内でないことを示すメッセージを表示し、アプリケーション・エラーを戻します。

## INSTEAD OF トリガー

本来変更できないビューに対して DELETE または UPDATE、INSERT 操作を行うには、INSTEAD OF トリガーを使用します。ビューに対する挿入または更新、削除を禁止する構成体については、「ビューの問合せ」(4-362 ページ)を参照してください。次の例では、顧客データが 2 つの表に格納されます。オブジェクト・ビュー ALL\_CUSTOMERS は、CUSTOMERS\_SJ および CUSTOMERS\_PA の 2 つの表の UNION として作成されます。INSTEAD OF トリガーを使用して、値を挿入する例を次に示します。

```
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.location = 'SAN_JOSE') THEN
        INSERT INTO customers_sj
            VALUES (:new.cust, :new.address, :new.credit);
    ELSE
        INSERT INTO customers_pa
            VALUES (:new.cust, :new.address, :new.credit);
    END IF;
END;
```

## ● ユーザー定義の型および LOB、REF 列

PL/SQL ブロック内部のトリガー・アクションではオブジェクトおよび VARRAY、ネストした表、LOB、REF 列を参照および使用できますが、トリガー・アクション内部ではそれらの値は変更できません。UPDATE トリガーでは、オブジェクト型および VARRAY 型、REF 型の列を OF 句に指定すると、UPDATE 文によってこれらの列のいずれかの値が変更されるときにトリガーが起動されます。

LOB 列に INSTEAD OF トリガーを定義すると、:OLD 値と :NEW 値はどちらも読取りが可能です。書込みはできません。LOB 列にこの他のトリガーを定義した場合、:OLD 値の読取りはできますが、:NEW 値は読み取れません。また、:OLD 値も :NEW 値も書込みはできません。

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。また、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。

## 関連項目

CREATE TRIGGER (4-330 ページ)

DROP TRIGGER (4-406 ページ)

ENABLE 句 (4-414 ページ)

DISABLE 句 (4-375 ページ)

CREATE VIEW (4-359 ページ)

ALTER VIEW (4-152 ページ)

**OBJ** CREATE TYPE

用途

オブジェクト型、または名前付きの可変配列 (VARRAY)、ネストされた表型、不完全オブジェクト型を作成します。

**注意：** このコマンドは、使用しているデータベース・サーバーに Oracle Object Option がインストールされている場合に限り使用可能です。

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。「不完全」と呼ばれる理由は、このオブジェクト型には名前があるものの、属性およびメソッドがないからです。ただし、他の型からの参照が可能であるため、互いに参照する型の定義に使用できます。「不完全オブジェクト型」(4-348 ページ) を参照してください。

オブジェクトおよび不完全型、VARRAY、ネストした表の詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』および『Oracle8 Server アプリケーション開発者ガイド』、『Oracle8 Server 概要』を参照してください。

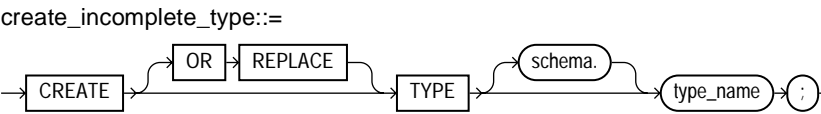
前提条件

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。

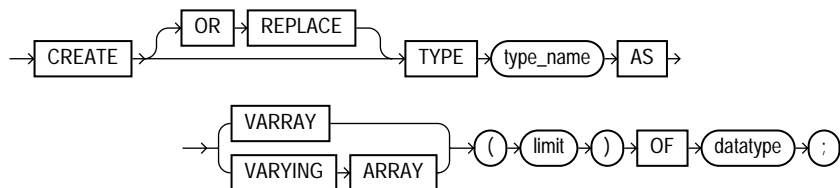
型の所有者は、表の定義内で参照される他のすべての型にアクセスするため明示的に EXECUTE オブジェクト権限が付与されているか、EXECUTE ANY TYPE システム権限が付与されている必要があります。所有者はこれらの権限をロールを介して取得することはできません。

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。これらの権限を持っていない場合、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。

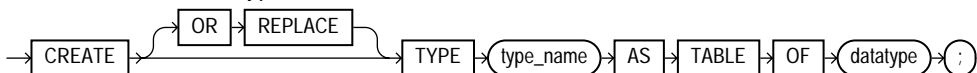
構文



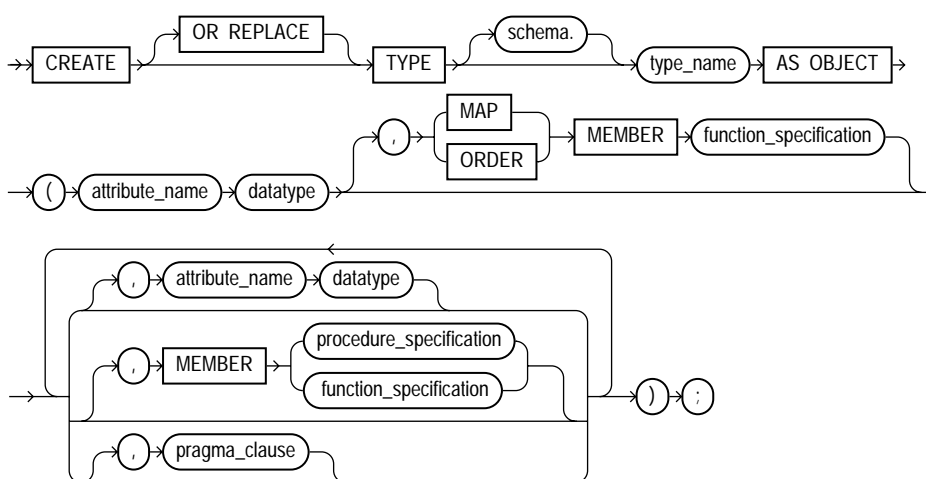
create\_varray\_type::=



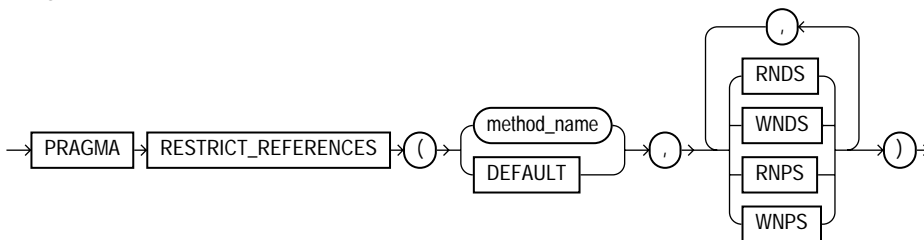
create\_nested\_table\_type::=

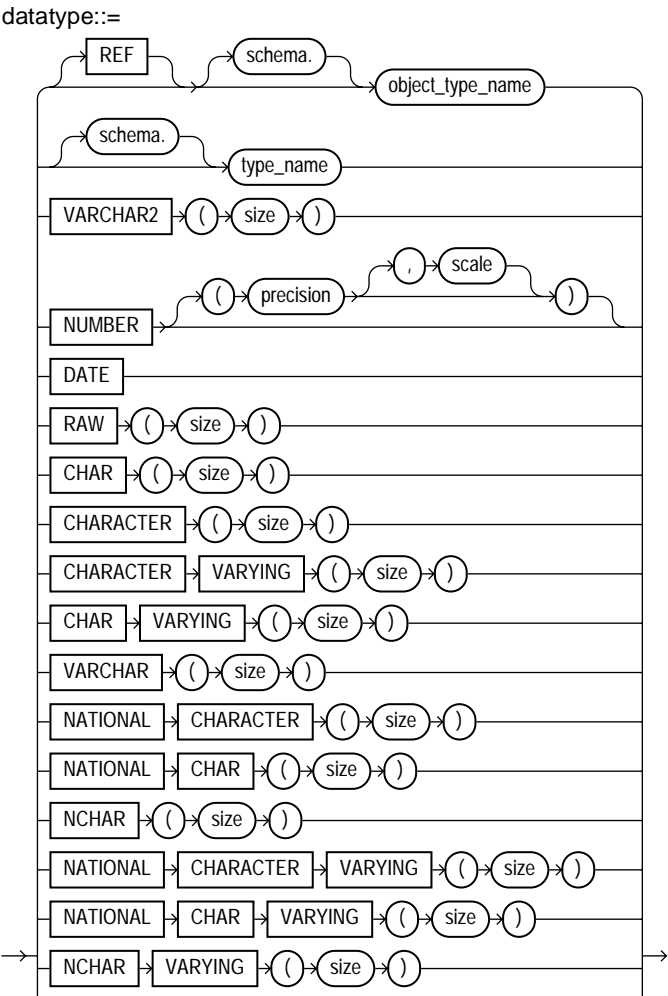


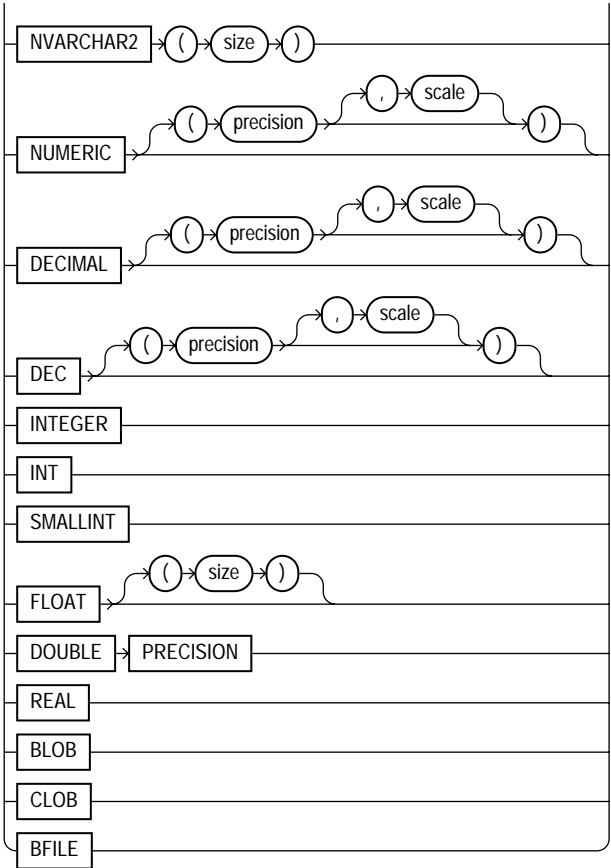
create\_object\_type::=



pragma\_clause::=







キーワードとパラメータ

OR REPLACE	型がすでに存在している場合に、それを再作成します。このオプションを使うと、既存の型の定義を、削除を伴わずに変更できます。  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、再作成後にあらためて権限を取得し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。
schema	作成する型を含むスキーマを指定します。schema を指定しないと、現行スキーマにその型が作成されます。
type_name	オブジェクト型またはネストした表の型、VARRAY 型の名前を指定します。

AS OBJECT	型をユーザー定義オブジェクト型として作成します。データ構造体を形成する変数は、属性と呼ばれます。オブジェクトの動作を定義するメンバー・サブプログラムは、メソッドと呼ばれます。AS OBJECT は、オブジェクト型の作成時に必要です。「コンストラクタ」(4-349 ページ)を参照してください。
AS TABLE OF	<p><i>datatype</i> 型で名前付きのネストした表を作成します。</p> <p><i>datatype</i> がオブジェクト型である場合、ネストした表の型は、オブジェクト型の名前および属性に一致する列をもつ表を記述します。</p> <p>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" と呼ばれる 1 つのスカラー型列をもつ表を記述します。</p> <p>1 つのコレクション型には、直接か間接かを問わず、他のコレクション型を入れることはできません。</p>
AS VARRAY( <i>limit</i> )	<p>それぞれが同じデータ型をも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 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なければなりません。可変長の配列だけがサポートされます。Oracle では、無名の VARRAY はサポートされません。</p> <p>VARRAY 内に含まれるオブジェクトの型名は、次のいずれかでなければなりません。</p> <ul style="list-style-type: none"> <li>スカラー・データ型 (以下の説明を参照)。使用可能なデータ型は、このコマンドの構文の中に示しています。</li> <li>REF 属性または</li> <li>VARRAY 属性を持つオブジェクトなどのオブジェクト型。</li> </ul> <p>VARRAY 内に含まれるオブジェクトの型名として、次のものは使えません。</p> <ul style="list-style-type: none"> <li>ネストした表を属性として持つオブジェクト型。</li> <li>VARRAY 型。</li> <li>TABLE 型。</li> </ul> <p>1 つのコレクション型には、直接か間接かを問わず、他のコレクション型を入れることはできません。</p>
OF <i>datatype</i>	Oracle のビルトイン・データ型またはライブラリ型の名前を指定します。ROWID および LONG、LONG RAW は無効なデータ型です。可能なデータ型のリストは、CREATE TYPE (4-342 ページ) の構文の定義を参照してください。
REF <i>object_type_name</i>	ソース・タイプのインスタンスをターゲット・オブジェクトのインスタンスに関連付けます。REF はターゲット・オブジェクトを論理的に識別および位置付けします。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。
<i>attribute_name</i>	オブジェクトの属性名を指定します。属性とは、名前および型指定子を持ち、オブジェクトの構造体を形成するデータ項目です。
MEMBER	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関数またはプロシージャ・サブプログラムを指定します。パッケージ内のサブプログラム名の多重定義の詳細は、『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。



	各プロシージャまたは関数の仕様部について、オブジェクト型本体に対応するメソッド本体を指定しなければなりません。 <b>CREATE TYPE BODY</b> (4-350 ページ) を参照してください。
<i>procedure_ specification</i>	プロシージャ・サブプログラムの仕様部です。
<i>function_ specification</i>	関数サブプログラムの仕様部です。
<b>MAP MEMBER function_ specification</b>	<p>オブジェクトのすべてのインスタンスにおける指定したインスタンスの順番を戻すメンバー関数 (MAP メソッド) を指定します。MAP メソッドは暗黙にコールされ、オブジェクト・インスタンスを事前定義済み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付けます。PL/SQL では、この順番を使用して、ブール式の評価および比較を行います。</p> <p>スカラー値は常に 1 つの単位として処理されます。スカラーは、基礎を形成するハードウェアに直接マップされます。たとえば、整数の場合、メモリー内またはディスク上に 4 バイトまたは 8 バイトの連続する記憶領域を占有します。</p> <p>オブジェクト仕様部に含められる MAP メソッドは、1 つだけです。この MAP メソッドは関数でなければなりません。結果として生成される型は、事前定義済みの SQL スカラー型でなければなりません。また、MAP 関数に指定できる引数は、暗黙の SELF 引数だけです。</p>
<b>ORDER MEMBER function_ specification</b>	<p>オブジェクトのインスタンスを明示の引数および暗黙の SELF 引数として取り、負の整数、またはゼロ、正の整数のいずれかを戻すメンバー関数 (ORDER メソッド) を指定します。負または正、ゼロは、それぞれ、暗黙の SELF 引数が明示の引数より小さい、または等しい、大きいことを指示します。</p> <p>同じオブジェクト型定義の各インスタンスを <b>ORDER BY</b> 句の中で比較すると、順序付けメソッド <i>function_specification</i> が起動されます。</p> <p>オブジェクト仕様部に含められる ORDER メソッドは 1 つだけです。この ORDER メソッドは戻り型 <b>INTEGER</b> をもつ関数でなければなりません。</p>
<p>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。</p>	
<p>MAP メソッドも ORDER メソッドも定義されない場合、実行できるのは等価性または非等価性に関する比較だけです。したがって、オブジェクト・インスタンスに順番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、それぞれに対応する属性が等しい場合に限り等価であるので注意してください。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。オブジェクト値の比較の詳細は、「オブジェクト値」(2-25 ページ) および『Oracle8 Server アプリケーション開発者ガイド』を参照してください。</p>	

<i>pragma_clause:</i>	
PRAGMA	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関数の読み書き
RESTRICT_	アクセスを否定することによって、副次的作用の発生を防止するコンパイラ・ディレクティブ
REFERENCES	です。詳細は、『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。
<i>method_name</i>	プラグマが適用されている MEMBER 関数またはプロシージャの名前です。
WNDS	データベースの書き込み禁止状態制約（データベースの表を変更しない）を指定します。
WNPS	パッケージの書き込み禁止状態制約（パッケージ変数を変更しない）を指定します。
RNDS	データベースの読取り禁止状態制約（データベースの表の問合せをしない）を指定します。
RNPS	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）を指定します。

不完全オブジェクト型

不完全オブジェクト型を使用して表やオブジェクト列、ネストされた表型の列を作成するには、不完全オブジェクト型を完全に指定しておく必要があります。

ネストした VARRAY およびネストした表の型は作成できません。つまり、VARRAY 型およびネストした表型に、VARRAY およびネストした表を要素として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。

例 1. 次の文は、LOB 属性を持つオブジェクト型 PERSON\_T を作成する例です。

```
CREATE TYPE person_t AS OBJECT
  (name CHAR(20),
   resume CLOB,
   picture BLOB);
```

例 2. 次の文は、MEMBER\_TYPE を、100 の要素を持つ VARRAY 型として作成する例です。

```
CREATE TYPE members_type AS VARRAY(100) OF CHAR(5);
```

例 3. 次の文は、オブジェクト型 PROJECT\_T の名前付き表型 PROJECT\_TABLE を作成する例です。

```
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;
```

例 4. 次の文は、メソッド・コンストラクタ COL.GETBAR() を起動する例です。

```
CREATE TYPE foo AS OBJECT (a1 NUMBER,  
                           MEMBER FUNCTION getbar RETURN NUMBER,  
                           pragma RESTRICT_REFERENCES(getbar, WNDS, WNPS));  
CREATE TABLE footab(col foo);  
  
SELECT col.getbar() FROM footab;
```

## コンストラクタ

ユーザー定義型を作成すると、Oracle は暗黙に、そのコンストラクタ・メソッドを定義します。コンストラクタとは、SQL 文または PL/SQL コード内で型の値のインスタンスを組み立てるために使用される、システム提供のプロシージャです。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出現します。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。

関数とは異なり、メソッドを起動する場合は、メソッドが他に引数をとらない場合でもカッコが必要です。

例 . 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 オブジェクトを組み立て、組み立てたオブジェクトを FOO\_TAB 表に入れる例です。

```
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));
```

コンストラクタの詳細は、『Oracle8 Server アプリケーション開発者ガイド』および『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。

## 関連項目

ALTER TYPE (4-142 ページ)

CREATE TYPE BODY (4-350 ページ)

『Oracle8 Server アプリケーション開発者ガイド』

『PL/SQL ユーザーズ・ガイドおよびリファレンス』

**OBJ** CREATE TYPE BODY

用途

オブジェクト型仕様部内に定義されているメンバー・メソッドを定義またはインプリメントします。「使用上の注意」（4-352 ページ）を参照してください。

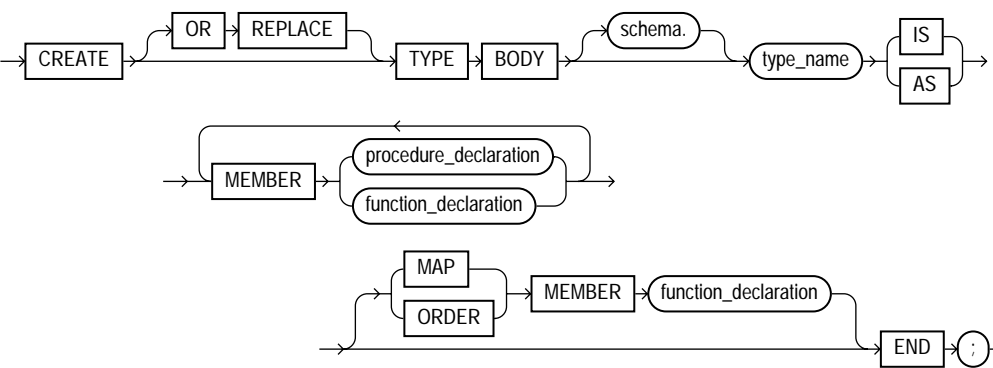
**注意：** このコマンドは、使用しているデータベース・サーバーに Oracle Object Option がインストールされている場合に限り使用可能です。

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 BODY 文内になければなりません。

自スキーマ内で型本体を作成または再作成するには、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には、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再作成するには、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

OR REPLACE	型本体がすでに存在している場合に、それを再作成します。このオプションを使うと、既存の型本体の定義を、削除を伴わずに変更できます。
------------	--

再作成されたオブジェクト型本体に対する権限を以前に付与されているユーザーは、権限の再付与が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。

このオプションを使うと、ALTER ... REPLACE コマンド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** 作成する本体を収録する型スキーマです。schema を省略すると、Oracle では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。

**MEMBER** オブジェクト型仕様部に関連付けられたメソッド関数またはメソッド・プロシージャ・サブプログラムを宣言またはインプリメントします。パッケージ内のサブプログラム名の多重定義の詳細は、『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。

各プロシージャまたは関数宣言には、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、関数の場合は、オブジェクト型仕様部内の戻り型を定義しなければなりません。CREATE TYPE BODY (4-350 ページ) を参照してください。

**procedure\_declaration** プロシージャ・サブプログラムの宣言です。型本体の書込みの詳細は、『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。

**function\_declaration** 関数サブプログラムの宣言です。型本体の書込みの詳細は、『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。

**MAP MEMBER function\_declaration** オブジェクトのすべてのインスタンスの中での指定インスタンスの順番を戻すメンバー関数 (MAP メソッド) を宣言またはインプリメントします。MAP メソッドは暗黙にコールされ、オブジェクト・インスタンスを事前定義済み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 では、この順番を使用して、ブール式の評価および比較を行います。

オブジェクト型本体に含まれる MAP メソッドは 1 つだけであり、これは関数でなければなりません。この MAP 関数に指定できる引数は、暗黙の SELF 引数だけです。

**ORDER MEMBER function\_specification** オブジェクトのインスタンスを明示の引数および暗黙の SELF 引数として取り、負の整数、またはゼロ、正の整数のいずれかを戻すメンバー関数 (ORDER メソッド) を指定します。負または正、ゼロは、それぞれ、暗黙の SELF 引数が明示の引数より小さい、または等しい、大きいことを指示します。

同じオブジェクト型定義の各インスタンスを ORDER BY 句の中で比較すると、順序付けメソッド *function\_specification* が起動されます。

オブジェクト仕様部に含まれる ORDER メソッドは 1 つだけです。この ORDER メソッドは戻り型 INTEGER をもつ関数でなければなりません。

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 内でオブジェクト型を比較できます。

どちらのメソッドも宣言しないと、オブジェクト・インスタンスの同等性と不等性しか比較できません。同じ型定義のインスタンスは、それぞれに対応する属性が等しい場合に限り等価であるので注意してください。

## 使用上の注意

CREATE TYPE コマンドおよび CREATE TYPE BODY コマンドを使ってオブジェクト型を作成します。CREATE TYPE コマンドでは、オブジェクト型の名前、オブジェクトの属性、メソッド、その他のプロパティを指定します。CREATE TYPE BODY コマンドは、型でのメソッドに対するコードを含みます。

オブジェクト型仕様部内で指定される各メソッドは、対応するメソッド本体がオブジェクト型本体内になければなりません。

**例.** 次のオブジェクト型本体は、RATIONAL 用のメンバー・サブプログラムをインプリメントする例です。

```
CREATE TYPE BODY rational
IS
  MAP MEMBER FUNCTION rat_to_real RETURN REAL IS
  BEGIN
    RETURN numerator/denominator;
  END;

  MEMBER PROCEDURE normalize IS
    gcd INTEGER := 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;
```

## 関連項目

CREATE TYPE (4-342 ページ)

ALTER TYPE (4-142 ページ)

『PL/SQL ユーザーズ・ガイドおよびリファレンス』

---

## CREATE USER

### 用途

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。ユーザーに対して任意に割り当てることができる特性は次のとおりです。

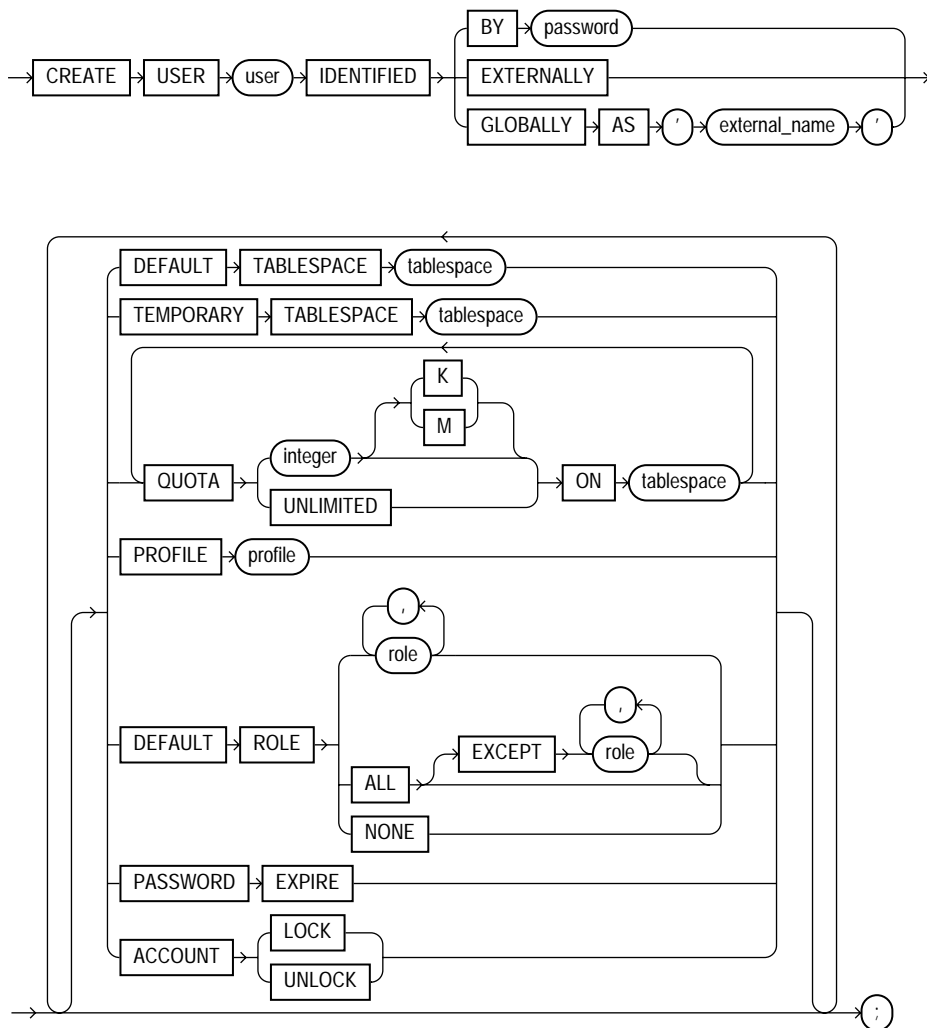
- デフォルトの表領域
- 一時表領域
- 表領域の割当て制限
- リソース制限を含むプロファイル

パスワード管理およびパスワード保護の詳細は、『Oracle8 Server 管理者ガイド』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。

## 構文





## キーワードとパラメータ

<i>user</i>	作成するユーザーの名前を指定します。この名前には、使用しているデータベースのキャラクタ・セットの文字だけを指定できます。また「スキーマ・オブジェクトの命名規則」(2-43 ページ)の項で説明した規則に従わなければなりません。データベースのキャラクタ・セットがマルチバイト文字をサポートしていても、ユーザー名にはシングルバイト文字を最低1つ使用することを推奨します。
IDENTIFIED	ユーザーを認証する方法を示します。詳細は、『Oracle8 Server アプリケーション開発者ガイド』および使用しているオペレーティング・システムに固有のマニュアルを参照してください。
BY <i>password</i>	ログオンするためにこのパスワードを指定するように求めます。パスワードは、「スキーマ・オブジェクトの命名規則」(2-43 ページ)で説明した規則に従っていなければなりません。また、データベースのキャラクタ・セットがマルチバイト文字をサポートしている場合でも、シングルバイト文字だけで指定してください。
EXTERNALLY	ユーザーを外部サービス（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）によって認証する必要があることを示します。「オペレーティング・システムを介したユーザーの検証」(4-356 ページ)を参照してください。
GLOBALLY AS ' <i>external_name</i> '	ユーザーを Oracle SecurityService によって認証する必要があることを示します。' <i>external_name</i> ' の文字列は、Oracle Security Service ではユーザーを識別する X.509 名となります。「ネットワークを介したユーザーの検証」(4-356 ページ)を参照してください。
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識別します。この句を省略すると、オブジェクトはデフォルトの SYSTEM 表領域に格納されます。
TEMPORARY TABLESPACE	ユーザーの一時セグメントが確保される表領域を識別します。この句を省略すると、一時セグメントはデフォルトの SYSTEM 表領域に確保されます。
QUOTA	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。任意の整数バイトで割当て制限を設定できます。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。K または M を使用して、KB 単位または MB 単位でも割当て制限を指定できます。「ユーザーに対する表領域の割当て制限の設定」(4-356 ページ)を参照してください。
	CREATE USER コマンドでは、複数の表領域に対して複数の QUOTA 句を指定できるので注意してください。
UNLIMITED	表領域への領域の割当てを無制限にします。
PROFILE	ユーザーに対して名前の付いたプロファイルを割り当てます。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。この句を省略すると、DEFAULT プロファイルがユーザーに割り当てられます。「ユーザーに対する権限付与」(4-356 ページ)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します。パスワードを変更してから、データベースにログインしてください。
ACCOUNT LOCK	ユーザーのアカウントをロックし、アクセスを禁止します。

ACCOUNT UNLOCK	ユーザーのアカウントのロックを解除し、そのアカウントにアクセスできるようにします。
-------------------	---

---

## オペレーティング・システムを介したユーザーの検証

CREATE USER ... IDENTIFIED EXTERNALLY を使用すると、データベース管理者は、特定のオペレーティング・システム・アカウントからだけアクセスできるデータベース・ユーザーを登録できます。実際には、オペレーティング・システムのログイン認証を運用することによって、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。したがって、データベース・アカウントのセキュリティの実効性は、このオペレーティング・システムに関連したセキュリティ・メカニズムに大きく依存しています。使用しているオペレーティング・システムでログイン時セキュリティに十分な配慮が払われていない場合、オラクル社では IDENTIFIED EXTERNALLY を使用しないようにお願いしています。詳細は、『Oracle8 Server 管理者ガイド』を参照してください。

## ネットワークを介したユーザーの検証

CREATE USER ... IDENTIFIED GLOBALLY を使用すると、データベース管理者は、外部認証サービス (Oracle Security Server(OSS) など)、または外部認証システムからだけ認証が得られるデータベース・ユーザーを作成できます。OSS の詳細は、『Oracle Security Server ガイド』および『Oracle8 Server 分散システム』を参照してください。

## ユーザーに対する表領域の割当て制限の設定

オブジェクトまたは一時セグメントを作成するには、ユーザーは必ず表領域に領域を割り当てなければなりません。ユーザーによる領域割当てを許可するには、QUOTA 句を使用します。CREATE USER 文には、異なる表領域ごとに複数の QUOTA 句を指定できます。他の句を指定できるのは、1 回だけです。

ある表領域に別のユーザーに対する割当て制限を設定する場合、その表領域に自分の割当て制限を設定する必要はありません。

## ユーザーに対する権限付与

ユーザーがデータベース処理を実行するには、ユーザーの権限ドメインに、その処理を許可する権限が含まれていなければなりません。ユーザーの権限ドメインには、そのユーザーに対して付与されている全権限と、そのユーザーが使用可能なロールの権限ドメインの全権限が含まれています。

注意：

- CREATE USER コマンド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）となる。
- Oracle にログオンするユーザーには、CREATE SESSION システム権限が必要。ユーザーを作成したら、この権限をそのユーザーに付与してください。

例 1. PASSWORD EXPIRE を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を試行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 carton
  DEFAULT TABLESPACE cases_ts
  QUOTA 10M ON cases_ts
  QUOTA 5M ON temp_ts
  QUOTA 5M ON system
  PROFILE engineer
  PASSWORD EXPIRE;
```

上記のユーザー SIDNEY には次の特性があります。

- パスワード CARTON
- 10MB の割当て制限のある、デフォルト表領域 CASES\_TS
- 5MB の割当て制限のある、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
- データベースへのログイン試行前に変更しなければならない期限切れのパスワード

例 2. オペレーティング・システム・アカウント GEORGE によってだけアクセス可能なユーザーを作成するには、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 "OPSS\$" である場合には、次の文でユーザー OPSS\$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
  QUOTA UNLIMITED ON temp_ts;
```

ユーザー OPSS\$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 上に無制限な領域
- 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

例 3. 次の文は、ユーザー CINDY をグローバル・ユーザーとして作成する例です。

## CREATE USER

---

```
CREATE USER cindy IDENTIFIED GLOBALLY AS 'CN=cindyuser'  
  DEFAULT TABLESPACE legal_ts  
  QUOTA 20M ON legal_ts  
  PROFILE lawyer;
```

## 関連項目

[ALTER USER](#) (4-148 ページ)

[CREATE PROFILE](#) (4-261 ページ)

[CREATE TABLESPACE](#) (4-325 ページ)

[GRANT\( システム権限とロール \)](#) (4-432 ページ)

---

## CREATE VIEW

### 用途

ビューを定義します。ビューとは、1 つ以上の表またはビューを基にして作成される論理的な表のことです。

---

**注意：** コマンドと句の説明で **OBJ** の印が前に付いている箇所は、Oracle Object Option がデータベース・サーバーにインストールされている場合にだけ読んでください。

---

**OBJ** CREATE VIEW は、LOB およびオブジェクト・データ型（オブジェクト型または REF、ネストした表、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・メカニズム上に作成します。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的オブジェクト識別子を持ったオブジェクトが含まれています。

オブジェクト・ビューの作成および使用の詳細は、「ビューの使用法」（4-362 ページ）および『Oracle8 Server アプリケーション開発者ガイド』を参照してください。ビューの作成例は、「例」（4-365 ページ）を参照してください。

### 前提条件

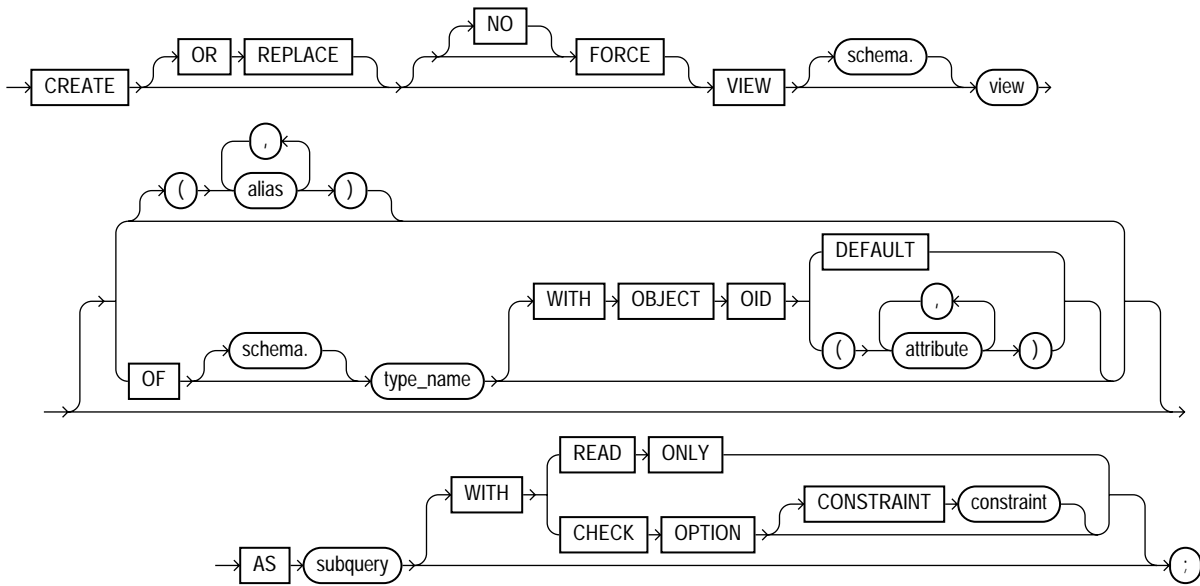
自スキーマ内にビューを作成するには、CREATE VIEW システム権限が必要です。別のユーザーのスキーマ内にビューを作成するには、CREATE ANY VIEW システム権限が必要です。

ビューを含んでいるスキーマの所有者は、そのビューの基礎となっているすべての表またはビューに対する行の選択または挿入、更新、削除の権限が必要です。これらの権限の詳細は、SELECT（4-486 ページ）および INSERT（4-449 ページ）、UPDATE（4-536 ページ）、DELETE（4-370 ページ）を参照してください。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、直接付与されていなければなりません。

**OBJ** 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には、次のいずれかが真でなければなりません。

- オブジェクト型が作成対象のビューと同じスキーマに属している。
- EXECUTE ANY TYPE システム権限を持つ。
- そのオブジェクト型に対する EXECUTE オブジェクト権限を持つ。

構文



subquery: 「副問合せ」(4-525 ページ) を参照。

キーワードとパラメータ

OR REPLACE	既存のビューがある場合には、そのビューを再作成します。このオプションを使用すると、以前に付与されたオブジェクト権限を削除、再作成、再付与することなく、既存のビュー定義を変更できます。
	ビューを再作成するとビュー内で定義された <b>INSTEAD OF</b> トリガーが削除されるので注意してください。 <b>INSTEAD OF</b> オプションの詳細は、 <b>CREATE TRIGGER</b> (4-330 ページ) を参照してください。
FORCE	ビューの実表または参照先オブジェクト型が存在しているかどうか、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どうかにかかわらず、強制的にビューを作成します。なお、 <b>SELECT</b> 、 <b>INSERT</b> 、 <b>UPDATE</b> 、 <b>DELETE</b> 文をビューに対して発行するには、上記の条件が真でなければなりません。
NO FORCE	実表が存在し、ビューを含むスキーマの所有者がそれらの表に対する権限を持っている場合だけ、ビューを作成します。これはデフォルトです。
schema	作成するビューが設定されているスキーマを指定します。schema を省略すると、自スキーマにビューが作成されます。

<b>view</b>	ビューまたはオブジェクト・ビューの名前を指定します。
<b>alias</b>	<p>ビューの間合せによって選択された式に対して名前を指定します。別名のは、ビューによって選択された式の数と一致していなければなりません。別名は、「スキーマ・オブジェクトと部分を参照する」(2-47 ページ) の項で説明しているスキーマ・オブジェクトの命名規則に従わなければなりません。別名はビュー内で一意でなければなりません。</p> <p>別名を省略すると、Oracle はビューの間合せの列または列の別名から別名を導き出します。このため、ビューの間合せが列の名前だけではなく式を含んでいる場合には、別名を使用する必要があります。</p> <p><b>OBJ</b> オブジェクト・ビュー作成時には別名は指定できません。</p>
<b>OBJ</b> OF <i>type_name</i>	<p><i>type_name</i> 型のオブジェクト表を明示的に作成します。オブジェクト・ビューの列が、<i>type_name</i> 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 OID 句で指定したオブジェクト識別子 (OID) に関連付けされます。「オブジェクト・ビュー」(4-366 ページ) を参照してください。</p> <p><i>schema</i> を省略すると、自スキーマ内にオブジェクト・ビューが作成されます。オブジェクト作成の詳細は、CREATE TYPE (4-342 ページ) を参照してください。</p>
<b>OBJ</b> [WITH OBJECT OID	<p>オブジェクト・ビュー内の各行を一意に識別するためのキーとして使用されるオブジェクト型の属性を指定します。多くの場合、各属性は実表の主キー列と対応します。</p> <p>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、DEFAULT を指定することもできます。</p>
<b>OBJ</b> DEFAULT	各行を一意に識別するために使用される基礎を形成するオブジェクト表またはオブジェクト・ビューの固有のオブジェクト識別子を指定します。
<b>OBJ</b> <i>attribute</i>	作成対象のオブジェクト・ビューの基となるオブジェクト型の属性です。
<b>AS subquery</b>	<p>ビューの基礎になっている表（実表）の列と行を識別します。ビューの間合せには、ORDER BY 句および FOR UPDATE 句を指定しない任意の SELECT 文を使用できます。選択リストとして 1000 以内の式を指定できます。「ビューの間合せ」(4-362 ページ) および「結合ビュー」(4-363 ページ)、「副問合せ」(4-525 ページ) を参照してください。</p> <p><b>OBJ</b> オブジェクト・ビューの場合、ビュー副問合せの選択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なければなりません。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なければなりません。</p> <p><b>OBJ</b> オブジェクト型、および REF<i>object_type</i>、LOB、VARRAY、ネストした表は、有効な列型です。</p>
WITH READ ONLY	ビューを介して削除および挿入、更新を実行できないことを指定します。

WITH CHECK OPTION	<p>ビューを介して実行される挿入や更新処理の結果が、ビュー問合せによって選択できる行でなければならないことを指定します。次のような場合、CHECKOPTION ではこの条件を保証できません。</p> <ul style="list-style-type: none"><li>• ビューの問合せまたはビューの基礎になるビューの問合せに副問合せが含まれている場合。</li><li>• INSERT または UPDATE、DELETE 操作が INSTEAD OF トリガーを使用して実行された場合。</li></ul>
CONSTRAINT <i>constraint</i>	<p>CHECK OPTION 制約の名前を設定します。この識別子を省略すると、その制約に SYS_Cn という形式の名前が自動的に割り当てられます。この場合の <i>n</i> 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。</p>

## ビューの使用方法

ビューは、別の表やビューからデータにアクセスできる論理表です。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実表と呼びます。

ビューは次の目的で使用できます。

- 事前に規定されている実表の行または列の集合、あるいはその両方に対してアクセスを制限することによる、表のセキュリティ・レベルの強化。
- データの複雑性を隠す。たとえば、結果を得るために実際にはいくつかの表が使われている場合に、ビューを 1 つの表として使用できます。
- 別の視点から見たデータの提示。たとえば、ビューを使用すると、実際には実表の定義を変更することなく列を改名できます。
- 同じ SQL 文で参照されている別のデータベースではなく、ビューを含んでいるデータベースでの、結合などの操作の実行。（「結合ビュー」（4-363 ページ）参照。）

これらの SQL 文中では、表を利用可能な箇所ならどこにでもビューを使用できます。

- COMMENT
- DELETE
- INSERT
- LOCK TABLE
- UPDATE
- SELECT

## ビューの問合せ

副問合せ内のビューの問合せの構文については、「副問合せ」（4-525 ページ）を参照してください。次のことに注意してください。

- ビューの問合せは CURRVAL および NEXTVAL 疑似列は選択できない。



- ビューの問合せが ROWID、ROWNUM、LEVEL の各疑似列を選択する場合、そのビューの問合せには別名がなければならない。
- 表の全列を選択するためにアスタリスク (\*) を使用する問合せでも、ビューを定義できる。

```
CREATE VIEW emp_vu
AS SELECT * FROM emp;
```

Oracle では、CREATE VIEW 文が発行される時に、アスタリスクが表の全列を示していると解釈します。後から表に新しい列を追加する場合には、別の CREATE VIEW 文に OR REPLACE オプションを指定してビューを再作成しない限り、ビューには追加した列は含まれません。ビューの問合せの選択リストの列をすべて指定する場合、アスタリスクは使わずに、すべての列を明示的に指定することをお勧めします。

- ビューの問合せにデータベース・リンクを使用することにより、リモート表およびビューを参照するビューを作成できる。ビューの問合せで参照されるリモート表やリモート・ビューは、そのビューが含まれるスキーマの名前を付けて修飾してください。また、ビューの問合せで使用されるデータベース・リンクは、CREATE DATABASE LINK コマンドの CONNECT TO 句を使用して定義してください。

上記の注意事項はスナップショットに対する問合せにも適用されます。

ビューは、INSTEAD OF トリガーを使わずに挿入または更新、削除できる場合、ならびに次に記載する制限に適合している場合には、固有の特性として更新可能です。ただし、ビューの問合せに、次のいずれかの構成体が含まれている場合には、固有の特性として更新不能であるため、そのビューに対して、INSTEAD OF トリガーを使う場合以外での挿入および更新、削除は実行できません。

- 集合演算子
- グループ関数
- GROUP BY、CONNECT BY、START WITH の各句
- DISTINCT 演算子
- 結合（結合ビューのサブセットは更新可能）

ビューに疑似列または式が含まれているときは、疑似列や式を参照していない UPDATE 文を使用する場合だけ、ビューを更新できます。

**●** オブジェクト・ビューまたはオブジェクト型をサポートする関連ビューの更新の詳細は、『Oracle8 Server アプリケーション開発者ガイド』を参照してください。

## 結合ビュー

結合ビューとは、結合を含む副問合せを使用するビューです。「ビューの問合せ」（4-362 ページ）で説明されている制限は、結合ビューにも適用されます。

副問合せの結合において、一意索引を持つ列が少なくとも1列ある場合、結合ビューで1つの実表を変更できます。結合ビューの中の列が更新可能であるかどうかは、`USER_UPDATABLE_COLUMNS` を問い合わせることでわかります。たとえば、次のように指定できます。

```
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 UPD
-----
ENAME YES
DEPTNO NO
EMPNO YES
LOC NO
```

この例では、DEPT 表の DEPTNO 列に一意の索引があります。EMP 表にマッピングするビュー内の列すべてが更新可能とマークされていて、しかも EMP の主キーがビューに含まれているため、EMP 実表に対し行の挿入または更新、削除ができます。結合ビューの更新の詳細は、『Oracle8 Server アプリケーション開発者ガイド』を参照してください。

---

---

**注意：** ビューの副問合せで指定されていない NOT NULL 列が EMP 実表にある場合、ビューを使用して表に挿入することはできません。

---

---

## ROWID の選択

結合内にキー保存表が1つだけある場合には、結合ビューから ROWID を選択できます。表の ROWID がビューの ROWID になります。

## 更新可能な結合ビュー

結合ビューは、結合を含むビューです。結合ビューは、この項で説明される条件の下で更新可能です。

キー保存表は、結合ビュー内の表であり、キー列がすべて結合ビュー内でキーとして存在している表です。すなわち、キーは、結合ビュー内になくてはならないだけでなく、結合ビュー内で一意かつ NOT NULL でなければなりません。つまり、キー保存表は一般に外部結合表にはできません。キー保存表を外部結合表として使用できるのは、外部結合で実際に

は NULL が生成されていない場合だけです。ただし、これはデータそのものの働きであるため、操作の基準としては使えません。

このため、次のすべてが真の場合にだけ、結合ビューに対して DML 文の INSERT または UPDATE、DELETE を実行できます。

- DML 文は、結合の基礎を形成する表の 1 つだけに影響を及ぼす。
- UPDATE 文である場合、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 を持っている場合、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 できません。
- DELETE 文の場合、結合内のキー保存表は 1 つだけ。この表は、ビューに CHECK OPTION が無い限り、結合内に 2 回以上提示できます。
- INSERT 文の場合は、値を挿入するすべての列がキー保存表の列であり、ビューに CHECK OPTION はない。

## 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 では、パーティション・ビューがサポートされているため、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は何の変更もなしに行えます。多くの場合は、Oracle8 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。(詳細は、『Oracle8 Server 移行ガイド』および『Oracle8 Server アプリケーション開発者ガイド』を参照してください)。

Oracle8 では、CREATE TABLE コマンドを使って、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・ビューと同じ利点がある上に、パーティション・ビューの欠点にも対処しています。ほとんどの動作環境では、パーティション・ビューではなく、パーティション表を使うことをお勧めします。パーティション表の詳細は、CREATE TABLE (4-303 ページ) を参照してください。

## 例

**例 1.** 次の文は、EMP 表から DEPT20 という名前のビューを作成する例です。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;
```

なお、副問合せで SAL\*12 式に対して列の別名 (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。

**例 2.** 次の文は、従業員表内の事務員全員の更新可能な表 **CLERKS** を作成する例です。このビューの中では、これらの従業員の **ID** および名前、部門番号だけが表示されます。また、これらの列は、事務員として識別される行だけで更新可能です。

```
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。

**例 3.** 次の文は、EMP 表の中の全事務員に関する **CLERKS** という読取り専用のビューを作成する例です。このビューには、従業員の **ID**、名前、部門番号だけが表示されます。

```
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

オブジェクト・ビューは、関係表またはオブジェクト表の間合せに基づいてオブジェクトを同期化します。ビュー副問合せの選択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なければなりません。各選択要素のデータ型は、対応する最上位属性と同じ（または変換可能）でなければなりません。

**WITH OBJECT OID** 句内の属性の集合によって、オブジェクト・ビューのオブジェクトに対する一意のキーが指定されなければなりません。オブジェクト・ビュー内の複数のインスタンスに解決される主キー **REF** を参照解除または確保しようとする、Oracle はエラーを戻します。

ビューが固有の特性として更新可能であり、**INSTEAD OF** トリガーを持っている場合、トリガーが優先します。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。

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は、その固有な特性として更新可能であっても、**INSTEAD OF** トリガーを持たなければなりません。

オブジェクト・ビューの詳細は、『Oracle8 Server 概要』および『Oracle8 Server アプリケーション開発者ガイド』を参照してください。

**例 .** 次は、EMPLOYEE\_TYPE のオブジェクト・ビュー EMP\_OBJECT\_VIEW を作成する例です。

```
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 OID (empno)
AS SELECT empno, ename, job, mgr, hiredate, sal, comm
FROM emp;
```

## 関連項目

CREATE TABLE (4-303 ページ)

CREATE SYNONYM (4-299 ページ)

CREATE TYPE (4-342 ページ)

DROP VIEW (4-412 ページ)

RENAME (4-470 ページ)

SELECT (4-486 ページ)

# DEALLOCATE UNUSED 句

## 用途

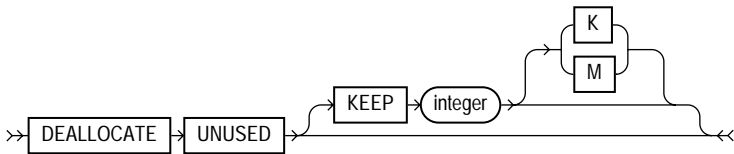
エクステントから解放する割当て済みの未使用領域の量を指定します。「使用上の注意」（4-368 ページ）を参照してください。

## 前提条件

この句は、次のコマンドでだけ使用できます。

- ALTER CLUSTER
- ALTER TABLE
- ALTER INDEX

## 構文



## キーワードとパラメータ

KEEP	保持する未使用領域の量を指定します。
<i>integer</i>	保持するバイト数を指定します。K または M を使用して KB 単位または MB 単位でもサイズを指定できます。

## 使用上の注意

スキーマ・オブジェクトの管理の詳細は、『Oracle8 Server 管理者ガイド』を参照してください。

`DEALLOCATE` 句は、クラスタまたは表、索引におけるエクステント内の未使用領域を解放して、表領域内の別のオブジェクトが再利用できるようにします。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だけ増加します。

未使用領域の割当て解除は、オブジェクトの末端から開始し、オブジェクトの先頭にある上限基準点に向かって進行します。割当て解除の範囲内にエクステントが完全に含まれるときは、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

ときは、上限基準点までのすでに使わ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 および MINEXTENTS、NEXT パラメータによって異なります（STORAGE 句（4-518 ページ）で説明します）。

- KEEP オプションを省略した場合、上限基準点が INITIAL と MINEXTENTS のサイズを超えるときは、上限基準点を超える未使用領域がすべて解放される。上限基準点が INITIAL または MINEXTENTS のサイズ未満のときは、MINEXTENTS より大きい未使用領域がすべて解放される。
- KEEP オプションを指定すると、指定した量の領域が保持され、残りの領域だけが解放される。このとき残りのエクステント数が MINEXTENTS より小さくなると、MINEXTENTS はそのエクステント数に変更される。また、初期エクステントが INITIAL より小さくなると、INITIAL がそのサイズに変更される。
- どちらの場合も NEXT は、割当てを解除された最後のエクステントのサイズに設定される。

**例** . 次のコマンドは、表 EMP で再利用できるように上限基準点が MINEXTENTS を超える未使用領域すべてを解放します。

```
ALTER TABLE emp
    DEALLOCATE UNUSED
```

## 関連項目

ALTER TABLE（4-105 ページ）

『Oracle8 Server 管理者ガイド』

『Oracle8 Server 概要』

---

## DELETE

### 用途

表またはパーティション表、ビューの実表、ビューの実パーティション表から行を削除します。「DELETE の使用方法」(4-372 ページ) を参照してください。

---

---

**注意：** コマンドと句の説明で **OBJ** の印が前に付いている箇所は、Oracle Object Option がデータベース・サーバーにインストールされている場合にだけ読んでください。

---

---

### 前提条件

表から行を削除するには、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。

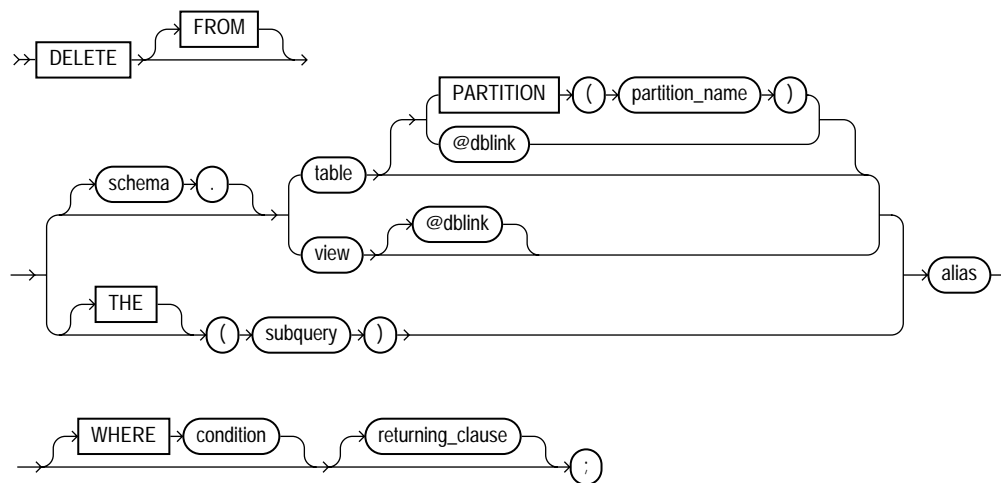
ビューの実表から行を削除するには、そのビューを含むスキーマの所有者は、その実表の DELETE 権限が必要です。また、他のスキーマ内にビューが存在している場合、そのビューの DELETE 権限が必要です。

DELETE ANY TABLE システム権限があれば、任意の表または任意のビューの実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、表列 (WHERE 句の列など) を参照する DELETE を実行するには、その表の SELECT 権限を持っていなければなりません。

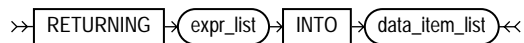


## 構文

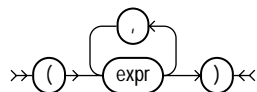


subquery: 「副問合せ」(4-525 ページ)を参照。

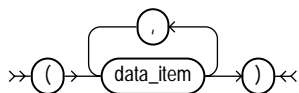
returning\_clause ::=



expr\_list ::=



data\_item\_list ::=



## キーワードとパラメータ

*schema*

更新する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* を指定しないと、Oracle は、この表が自スキーマに存在するとみなします。

<i>table</i> または <i>view</i>	行が削除される表またはビューの名前を指定します。 <i>view</i> を指定すると、ビューの実表から行が削除されます。
<i>dblink</i>	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。データベース・リンクの参照については、「リモート・データベース内のオブジェクトを参照」(2-50 ページ) を参照してください。Oracle の分散機能を使用している場合に限り、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。  <i>dblink</i> を省略すると、その表またはビューはローカル・データベースに存在するとみなされます。
PARTITION ( <i>partition_name</i> )	<i>table</i> のパーティション・レベルの行削除を指定します。 <i>partition_name</i> には、表内にある削除対象のパーティションの名前を指定します。「1つのパーティションからの削除」(4-374 ページ) を参照してください。
◯◯ THE	副問合せが戻す列値がスカラー値でなく、ネストした表であることを通知します。接頭辞 THE の付いた副問合せはフラット化した副問合せと呼ばれます。「フラット化した副問合せの使用」(4-528 ページ) を参照してください。
<i>subquery</i>	削除の対象として選択するデータを指定します。Oracle は副問合せを実行し、その結果の行を FROM 句内で表として使用します。副問合せでは、副問合せと同じ FROM 句に指定された表を問い合わせることはできません。「副問合せ」(4-525 ページ) を参照してください。
<i>alias</i>	表、ビュー、副問合せに対して割り当てられる別名です。通常、別名は相関問合せを持つ DELETE 文で使用されます。
WHERE <i>condition</i>	条件を満たす行だけを削除します。この条件は、表を参照したり、副問合せを指定したりします。構文の説明は、「条件」(3-84 ページ) を参照してください。Oracle の分散機能を使用している場合に限り、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。  <i>dblink</i> を省略すると、その表またはビューはローカル・データベースに存在するとみなされます。  WHERE 句を省略すると、表またはビューの行がすべて削除されます。
<i>returning_clause</i>	DELETE 文に影響される行を取り出します。スカラー、LOB 型、ROWID 型、REF 型だけが取り出されます。「RETURNING 句」(4-374 ページ) を参照してください。
<i>expr</i>	「式」(3-73 ページ) で説明した構文です。 <i>data_item_list</i> の各変数に対する列の式を指定してください。
INTO	変更された行の値を、 <i>data_item_list</i> に指定する変数に格納することを示します。
<i>data_item</i>	取り出された式の値を格納する PL/SQL 変数またはバインド変数です。
パラレル DML またはリモート・オブジェクトでは RETURNING 句を使用できません。	

DELETE の使用方法

Oracle オプティマイザに指示（すなわちヒント）を引き渡すために、DELETE 文中でコメントを使用できます。オプティマイザは、ヒントを使用して文の実行計画を選択します。ヒントの詳細は、『Oracle8 Server チューニング』を参照してください。

DELETE キーワードに続けてパラレル・ヒントを配置すれば、基礎となる走査および DELETE 操作の両方をパラレル化できます。パラレル DML の詳細は、『Oracle8 Server チューニング』および『Oracle8 Parallel Server 概要および管理』、『Oracle8 Server 概要』を参照してください。

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、表と索引によって引き続き保持されます。ビューを定義する問合せに次のいずれかの構成体が含まれている場合には、ビューからの削除はできません。

- 集合演算子
- GROUP BY 句
- グループ関数
- DISTINCT 演算子

表に対して DELETE 文を実行すると、表に対して定義されている DELETE トリガーが起動します。

**例 1.** 次の文は、表 TEMP\_ASSIGN からすべての行を削除する例です。

```
DELETE FROM temp_assign;
```

**例 2.** 次の文は、先月の歩合が 100 ドル未満の営業担当者をすべて従業員表から削除する例です。


```
DELETE FROM emp
WHERE JOB = 'SALESMAN'
AND COMM < 100;
```

**例 3.** 次の文は、例 2 と同じ効果があります。

```
DELETE FROM (select * from emp)
WHERE JOB = 'SALESMAN'
AND COMM < 100;
```

**例 4.** 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 が所有する銀行預金表からすべての行を削除する例です。

```
DELETE FROM blake.accounts@dallas;
```

**例 5.**  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する例です。

```
DELETE THE(SELECT proj
FROM dept d WHERE d.dno = 123) AS p
WHERE p.pno IN (123, 456) OR p.budgets > 456.78;
```

## 1つのパーティションからの削除

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が、パーティション名を指定した方が複雑な WHERE 句より効率が高まることがあります。パーティション表の1つのパーティションの値を変更するときは、PARTITION 句を指定します。この構文を使用すると、WHERE 句を使用するより処理が簡単になることがあります。

**例.** 次は、SALES 表のパーティション NOV96 から行を削除する例です。

```
DELETE FROM sales PARTITION (nov96)
WHERE amount_of_sale != 0;
```

## RETURNING 句

RETURNING 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ので、DELETE 文の後で SELECT を実行する必要がなくなります。

- 1つの行を削除する場合は、RETURNING 句付きの DELETE 文によって、削除された行 ROWID を使用している列式および削除された行への参照 REF を取り出し、PL/SQL 変数またはバインド変数に格納する。
- RETURNING 句を持つ DELETE 文を使用して複数の行を削除した場合、削除された行の式の値、ROWID、REF はバインド配列に格納されます。

RETURNING 句を持つ DELETE 文を使用すると、実表を1つずつ持つビューから行を削除できます。

ホスト・バインドでは、式のデータ型およびサイズはバインド変数と互換性がなければなりません。

**例.** 次の例では、削除された行から列 SAL 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;
```

## 関連項目

UPDATE (4-536 ページ)

---

## DISABLE 句

### 用途

整合性制約または表に関係するトリガーをすべて使用禁止にします。

- 整合性制約が使用禁止になっている場合、その制約は施行されない。ただし、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、使用可能な整合性制約とともに表示されます。
- トリガーが使用禁止になっている場合、トリガー条件が満たされても、トリガーは起動されない。

「使用上の注意」（4-352 ページ）を参照してください。

### 前提条件

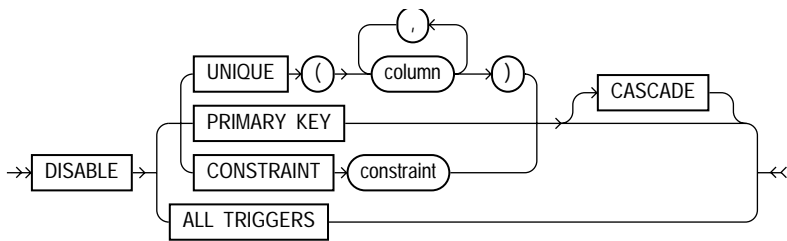
整合性制約を使用禁止にする **DISABLE** 句は、**CREATE TABLE** または **ALTER TABLE** コマンドに指定します。このため整合性制約を使用禁止にするには、どちらかのコマンドの実行権限が必要です。これらの権限の詳細は、**CREATE TABLE**（4-303 ページ）および **ALTER TABLE**（4-105 ページ）を参照してください。

**DISABLE** 句に指定する整合性制約について、次のいずれかの条件を満たす必要があります。

- 整合性制約が、その中の文に定義されている。
- 整合性制約がすでに定義済みで、かつ以前発行した文で使用可能にされている。

トリガーを使用禁止にする **DISABLE** 句は、**ALTER TABLE** 文でだけ指定できます。**DISABLE** 句でトリガーを使用禁止にする場合、この **ALTER TABLE** を発行するための各種権限が必要です。これらの権限の詳細は、**ALTER TABLE**（4-105 ページ）を参照してください。なお、トリガーが自スキーマ内になければなりません。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。

構文



キーワードとパラメータ

UNIQUE	指定した列または列の組合せに定義した UNIQUE 制約を使用禁止にします。
PRIMARY KEY	表の PRIMARY KEY 制約を使用禁止にします。
CONSTRAINT	constraint に指定した整合性制約を使用禁止にします。
CASCADE	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には、このオプションを指定します。
ALL TRIGGERS	表に関係するトリガーをすべて使用禁止にします。このオプションは、ALTER TABLE 文の DISABLE 句でだけ指定できます。CREATE TABLE 文では指定できません。

使用上の注意

- DISABLE 句を使用して使用禁止にできるのは次の 2 つです。
- 1 つの整合性制約
  - 表に対応付けられているすべてのトリガー
- 単一トリガーを使用禁止にするには、ALTER TRIGGER コマンドの DISABLE オプションを使用します。

整合性制約の使用禁止

整合性制約は、CREATE TABLE 文または ALTER TABLE 文の DISABLE 句に指定することによって、使用禁止にできます。CONSTRAINT 句を使用して定義した整合性制約を、同じ文に DISABLE 句を指定して使用禁止にできます。また、ある文に定義した整合性制約を、別の文で使用禁止にできます。

また、整合性制約を定義する CONSTRAINT 句に DISABLE キーワードを指定しても、整合性制約を使用禁止にできます。このキーワードの詳細は、CONSTRAINT 句 (4-185 ページ) を参照してください。

## Oracle が整合性制約を使用禁止にする方法

整合性制約を使用禁止にすると、その制約は施行されません。整合性制約を定義し、それを使用禁止にすると、Oracle は、それを使用可能な整合性制約とともにデータ・ディクショナリに格納しますが、それを表の既存行には適用しません。また、Oracle では、表データを変更したり、使用禁止になった整合性制約に違反するデータ操作言語 (DML) 文を実行できません。

すでに有効になっている UNIQUE 制約または PRIMARY KEY 制約を使用禁止にすると、その制約を実施する索引は削除されます。

使用禁止になっている整合性制約は、ENABLE 句を使用して使用可能にできます。

## 参照整合性制約の参照キーの使用禁止

参照整合性制約（外部キー）の参照キーを示す UNIQUE 制約または PRIMARY KEY 制約を使用禁止にするには、外部キーも使用禁止にする必要があります。これには、DISABLE 句の CASCADE オプションを使用します。

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は、使用可能にできません。

**例 1.** 次の文は、DEPT 表を作成し、使用禁止として PRIMARY KEY 制約を定義する例です。

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

主キーが使用禁止になっているため、主キーに違反する行を表に追加できます。たとえば、NULL の部門番号の部門を追加したり、同じ部門番号の部門を複数追加できます。

**例 2.** 次の文は、EMP 表に CHECK 制約を定義し、その制約を使用禁止にする例です。

```
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 を超える従業員はいないことを保証します。しかし、この制約が使用禁止になっているので、従業員の給与をこの制限以上に増やすことができます。

**例 3.** 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った参照整合性制約があるとしてします。この外部キーは、CUSTOMERS 表の AREACO 列および PHONENO 列を組み合わせた一意キーを参照します。次の文では、この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には、CASCADE オプションを使用します。このオプションによって外部キーも使用禁止となります。

### トリガーの使用禁止

ALTER TABLE 文の DISABLE 句内の ALL TRIGGERS オプションを使用して、表に関係するトリガーをすべて使用禁止にできます。トリガーを使用禁止にすると、トリガーを起動する文がトリガー制約の条件に適合しても、トリガーは実行されません。

**例** . 次の文は、EMP 表に関係するトリガーをすべて使用禁止にする例です。

```
ALTER TABLE emp
  DISABLE ALL TRIGGERS;
```

## 関連項目

ALTER TABLE (4-105 ページ)

ALTER TRIGGER (4-139 ページ)

CONSTRAINT 句 (4-185 ページ)

CREATE TABLE (4-303 ページ)

CREATE TRIGGER (4-330 ページ)

DISABLE 句 (4-375 ページ)



# DROP 句

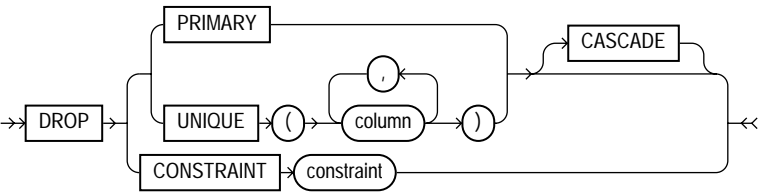
## 用途

データベースの整合性制約を削除します。「使用上の注意」(4-379 ページ)を参照してください。

## 前提条件

DROP 句は ALTER TABLE 文にだけ指定できます。整合性制約を削除するには、ALTER TABLE 文の実行権限が必要です。これらの権限の詳細は、ALTER TABLE (4-105 ページ)を参照してください。

## 構文



## キーワードとパラメータ

PRIMARY KEY	表の PRIMARY KEY 制約を削除します。
UNIQUE	指定した列の UNIQUE 制約を削除します。
CONSTRAINT	constraint に指定した整合性制約を削除します。
CASCADE	削除する整合性制約に依存する、他の整合性制約をすべて削除します。

## 使用上の注意

整合性制約は、ALTER TABLE 文の DROP 句に指定することにより削除できます。整合性制約を削除すると、その整合性制約は施行されなくなり、データ・ディクショナリから削除されます。

参照整合性制約を構成する一意キーまたは主キーは、外部キーを削除するまで削除できません。DROP 句の CASCADE オプションに参照キーを指定すると、参照キーと外部キーを同時に削除できます。CASCADE オプションを省略すると、一意キーまたは主キーを参照する外部キーがある場合、その一意キーまたは主キーは削除されません。

**例 1.** 次の文は、DEPT 表の主キーを削除する例です。

```
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 オプションによって、主キーを参照する外部キーがすべて削除されます。

**例 2.** 次の文は、DEPT 表の DNAME 列の一意キーを削除する例です。

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

この例の DROP 句には CASCADE オプションを指定していないことに注意してください。CASCADE オプションが指定されていないため、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。

## 関連項目

ALTER TABLE (4-105 ページ)

CONSTRAINT 句 (4-185 ページ)

# DROP CLUSTER

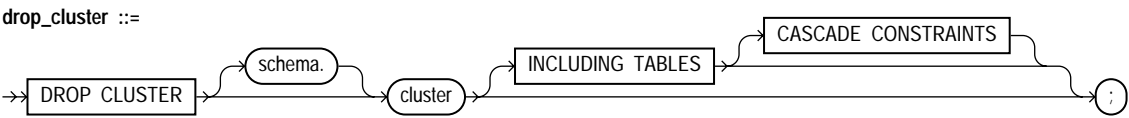
## 用途

データベースのクラスタを削除します。「使用上の注意」（4-381 ページ）を参照してください。

## 前提条件

クラスタが自スキーマ内にあるか、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

<i>schema</i>	クラスタが含まれているスキーマを指定します。 <i>schema</i> を指定しないと、そのクラスタは自分のスキーマにあるとみなされます。
<i>cluster</i>	削除するクラスタの名前を指定します。
INCLUDING TABLES	クラスタに属する表をすべて削除します。
CASCADE CONSTRAINTS	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、クラスタ外の表の参照整合性制約をすべて削除します。このような参照整合性制約があるときにこのオプションの指定を省略すると、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。

## 使用上の注意

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。

個々の表は非クラスタ化できません。既存のクラスタ化された表と同じ非クラスタ化された表を作成する手順は、次のとおりです。

1. 既存の表と同じ構成と内容の表を、CLUSTER オプションは指定せずに作成します。
2. 既存の表を削除します。
3. RENAME コマンドを使用して、新たな表に既存の表の名前を指定します。

既存のクラスタ化表に対して付与された権限は、新規作成の非クラスタ化表には適用されません。権限の再付与が必要です。

**例** . 次の文では GEOGRAPHY という名前のクラスタ、および関係する表、その表の主キーまたは一意キーを参照する参照整合性制約のすべてが削除されます。

```
DROP CLUSTER geography
INCLUDING TABLES
CASCADE CONSTRAINTS;
```

## 関連項目

[DROP TABLE \(4-402 ページ\)](#)

# DROP DATABASE LINK

## 用途

データベースのデータベース・リンクを削除します。

## 前提条件

プライベート・データベース・リンクを削除するには、データベース・リンクが自スキーマ内になければなりません。PUBLIC データベース・リンクを削除するには、DROP PUBLIC DATABASE LINK システム権限が必要です。次の「例」を参照してください。

## 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に必ず指定します。
<i>dblink</i>	削除するデータベース・リンクを指定します。

## 使用上の注意

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、スキーマ名により *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つけることができるため、たとえば RALPH.LINKTOSALES のような名前を付けると、スキーマ RALPH の中の LINKTOSALES という名前のデータベース・リンクと解釈されるのではなく、名前全体が自スキーマにあるデータベース・リンク名と解釈されます。

## 例

次の文は、BOSTON というプライベート・データベース・リンクを削除する例です。

```
DROP DATABASE LINK boston;
```

## 関連項目

CREATE DATABASE LINK (4-220 ページ)

---

# DROP DIRECTORY

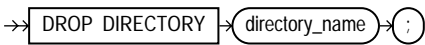
## 用途

DROP DIRECTORY は、データベース上のディレクトリ・オブジェクトを削除するときに使  
用します。次の「使用上の注意」を参照してください。

## 前提条件

ディレクトリを削除するには、DROP ANY DIRECTORY システム権限が必要です。

## 構文



## キーワードとパラメータ

---

<i>directory_name</i>	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。
-----------------------	------------------------------------

---

## 使用上の注意

ディレクトリを削除すると、データベース・オブジェクトが削除されます。ただし、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  
されません。

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。

例 . 次の文は、ディレクトリ・オブジェクト BFILE\_DIR を削除する例です。

```
DROP DIRECTORY bfile_dir;
```

## 関連項目

- CREATE DIRECTORY (4-226 ページ)
- GRANT( システム権限とロール ) (4-432 ページ)
- 「ラージ・オブジェクト (LOB) データ型」 (2-14 ページ)

## DROP FUNCTION

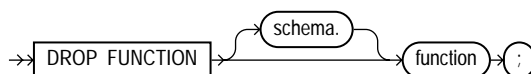
### 用途

データベースからスタンドアロン・ストアド・ファンクションを削除します。次の「使用上の注意」を参照してください。

### 前提条件

削除する関数が自スキーマ内にあるか、DROP ANY PROCEDURE システム権限を持っている必要があります。

### 構文



### キーワードとパラメータ

<i>schema</i>	関数が含まれているスキーマを指定します。 <i>schema</i> の指定を省略すると、関数は自スキーマに定義されているものとみなされます。
<i>function</i>	削除する関数の名前を指定します。

### 使用上の注意

関数を削除すると、その関数に依存するローカル・オブジェクト、つまりその関数をコールするローカル・オブジェクトはすべて無効となります。後でこのようなオブジェクトのいずれかを参照すると、Oracle がそのオブジェクトを再コンパイルしようとします。削除した関数を再作成しない限りエラー・メッセージが戻されます。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 Server 概要』を参照してください。

このコマンドは、スタンドアロン関数を削除する場合にだけ使用できます。パッケージを構成する関数の削除は、次のいずれかの方法で行います。

- DROP PACKAGE コマンドを使って、パッケージ全体を削除する。
- CREATE PACKAGE コマンドに OR REPLACE オプションを指定して、その関数を含まないパッケージを再定義する。

例 . 次の文は、スキーマ RIDDLEY 内の関数 NEW\_ACCT を削除する例です。

## DROP FUNCTION

---

```
DROP FUNCTION riddley.new_acct;
```

関数 NEW\_ACCT を削除すると、これに依存するオブジェクトはすべて無効となります。

## 関連項目

CREATE FUNCTION (4-228 ページ)



# DROP INDEX

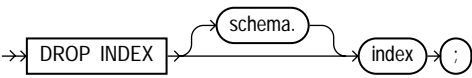
## 用途

データベースの索引を削除します。次の「使用上の注意」を参照してください。

## 前提条件

削除する索引が自スキーマ内にあるか、DROP ANY INDEX システム権限を持っている必要があります。

## 構文



## キーワードとパラメータ

<i>schema</i>	索引が含まれているスキーマを指定します。 <i>schema</i> の指定を省略すると、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。
<i>index</i>	削除する索引の名前を指定します。

## 使用上の注意

索引を削除すると、その索引に割り当てられていたすべてのデータ・ブロックが索引の表領域に戻されます。

例 . 次のコマンドは、MONOLITH という名前の索引を削除する例です。

```
DROP INDEX monolith;
```

## 関連項目

ALTER INDEX (4-28 ページ)

---

# DROP LIBRARY

## 用途

データベース上の外部プロシージャ・ライブラリを削除します。

## 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

<i>libname</i>	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

次の文は、EXT\_PROCS ライブラリを削除する例です。

```
DROP LIBRARY ext_procs;
```

## 関連項目

CREATE LIBRARY (4-244 ページ)

# DROP PACKAGE

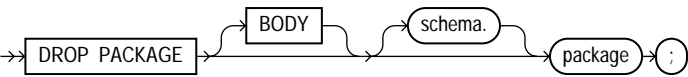
## 用途

データベース上のストアド・パッケージを削除します。次の「使用上の注意」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、DROP ANY PROCEDURE システム権限を持っている必要があります。

## 構文



## キーワードとパラメータ

<i>BODY</i>	パッケージの本体だけを削除します。このオプションを省略すると、パッケージの本体と仕様部の両方が削除されます。
<i>schema</i>	パッケージが含まれているスキーマを指定します。 <i>schema</i> の指定を省略すると、パッケージが自スキーマ内に定義されているものとみなされます。
<i>package</i>	削除するパッケージの名前を指定します。

## 使用上の注意

パッケージの本体と仕様部を削除すると、削除したパッケージ仕様部に依存するローカル・オブジェクトはすべて無効となります。後でこのようなオブジェクトのいずれかを参照すると、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。リモート・オブジェクトを含むスキーマ・オブジェクト間の依存性を Oracle が管理する方法の詳細は、『Oracle8 Server 概要』を参照してください。

パッケージの本体だけ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。

DROP PACKAGE コマンドを実行すると、パッケージとその中のオブジェクトが同時に削除されます。パッケージの中の 1 つのオブジェクトを削除する場合は、CREATE PACKAGE コ

マンドと `CREATE PACKAGE BODY` コマンドに `OR REPLACE` オプションを指定して、そのオブジェクトなしのパッケージを再作成します。

**例** . 次の文は、`BANKING` パッケージの仕様部と本体を削除し、その仕様部に依存するすべてのオブジェクトを無効にする例です。

```
DROP PACKAGE banking;
```

## 関連項目

`CREATE PACKAGE` (4-246 ページ)

# DROP PROCEDURE

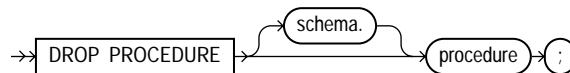
## 用途

データベース上のスタンドアロン・ストアド・プロシージャを削除します。次の「使用上の注意」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、DROP ANY PROCEDURE システム権限を持っている必要があります。

## 構文



## キーワードとパラメータ

<i>schema</i>	削除するプロシージャが含まれているスキーマを指定します。 <i>schema</i> の指定を省略すると、プロシージャが自スキーマ内に定義されているものとみなされます。
<i>procedure</i>	削除するプロシージャの名前を指定します。

## 使用上の注意

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ローカル・オブジェクトはすべて無効となります。後でこのようなオブジェクトのいずれかを参照すると、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。リモート・オブジェクトを含むスキーマ・オブジェクト間の依存性を Oracle が管理する方法の詳細は、『Oracle8 Server 概要』を参照してください。

このコマンドは、スタンドアロン・プロシージャを削除する場合にだけ使用できます。パッケージを構成するプロシージャを削除する場合は、次のいずれかの方法で行います。

- DROP PACKAGE コマンドを使って、パッケージ全体を削除する。
- CREATE PACKAGE コマンドに OR REPLACE オプションを指定して、そのプロシージャを含まないパッケージを再定義する。

**例** 次の文は、ユーザー KERNER が所有するプロシージャ TRANSFER を削除する例です。

```
DROP PROCEDURE kerner.transfer
```

プロシージャ TRANSFER を削除すると、これに依存するオブジェクトはすべて無効となります。

### 関連項目

[CREATE PROCEDURE](#) (4-255 ページ)

# DROP PROFILE

## 用途

データベースのプロファイルを削除します。次の「使用上の注意」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

<i>profile</i>	削除するプロファイルの名前を指定します。
CASCADE	削除するプロファイルを割り当てられているユーザーすべてから、そのプロファイルの割当てを取り消します。このようなユーザーに対しては、デフォルト・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オプションを指定します。

## 使用上の注意

DEFAULT プロファイルは削除できません。

例 . 次の文は、プロファイル ENGINEER を削除する例です。

```
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は、自動的にデフォルト・プロファイルが割り当てられます。

## 関連項目

CREATE PROFILE (4-261 ページ)

---

# DROP ROLE

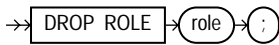
## 用途

データベースのロールを削除します。次の「使用上の注意」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、DROP ANY ROLE システム権限を持っている必要があります。

## 構文



## キーワードとパラメータ

---

<i>role</i>	削除するロールを指定します。
-------------	----------------

---

## 使用上の注意

ロールを削除すると、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。

**例** . 次の文は、ロール FLORIST を削除する例です。

```
DROP ROLE florist;
```

## 関連項目

CREATE ROLE (4-268 ページ)

SET ROLE (4-511 ページ)



---

## DROP ROLLBACK SEGMENT

### 用途

データベースのロールバック・セグメントを削除します。次の「使用上の注意」を参照してください。

### 前提条件

DROP ROLLBACK SEGMENT システム権限が必要です。

### 構文

```
→ DROP ROLLBACK SEGMENT rollback_segment → ;
```

### キーワードとパラメータ

---

<i>rollback_segment</i>	削除するロールバック・セグメントの名前を指定します。
-------------------------	----------------------------

---

### 使用上の注意

ロールバック・セグメントを削除すると、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。

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だけ削除できます。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` に問い合わせます。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` コマンドに `OFFLINE` オプションを指定して、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** . 次の文は、ロールバック・セグメント `ACCOUNTING` を削除する例です。

```
DROP ROLLBACK SEGMENT accounting;
```

### 関連項目

`ALTER ROLLBACK SEGMENT` (4-53 ページ)

`CREATE ROLLBACK SEGMENT` (4-272 ページ)

CREATE TABLESPACE (4-325 ページ)

---

## DROP SEQUENCE

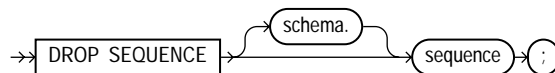
### 用途

データベースの順序を削除します。次の「使用上の注意」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、DROP ANY SEQUENCE システム権限を持っている必要があります。

### 構文



### キーワードとパラメータ

---

<i>schema</i>	削除する順序が含まれているスキーマを指定します。 <i>schema</i> の指定を省略すると、順序が自スキーマ内に定義されているものとみなされます。
<i>sequence</i>	削除する順序の名前を指定します。

---

### 使用上の注意

順序の初期値を再設定する方法の 1 つとして、順序を削除して再作成する方法があります。たとえば、現在値が 150 の順序があるときに、初期値 27 で順序を再スタートする方法は、次のとおりです。

1. 順序を削除します。
2. START WITH に値 27 を指定し、同じ名前で順序を作成します。

**例** . 次の文は、ユーザー ELLY が所有する順序 ESEQ を削除する例です。この文を発行するには、ユーザー ELLY として接続するか、DROP ANY SEQUENCE システム権限を持っている必要があります。

```
DROP SEQUENCE elly.eseq;
```

### 関連項目

ALTER SEQUENCE (4-56 ページ)

CREATE SEQUENCE (4-278 ページ)

# DROP SNAPSHOT

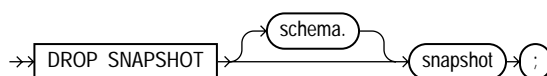
## 用途

データベースのスナップショットを削除します。次の「使用上の注意」を参照してください。

## 前提条件

削除するスナップショットが自スキーマ内にあるか、**DROP ANY SNAPSHOT** システム権限を持っている必要があります。また、**Oracle** がスナップショット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。これらの権限の詳細は、**DROP TABLE** (4-402 ページ) および **DROP VIEW** (4-412 ページ)、**DROP INDEX** (4-387 ページ) を参照してください。

## 構文



## キーワードとパラメータ

<i>schema</i>	削除するスナップショットが含まれているスキーマを指定します。 <i>schema</i> の指定を省略すると、スナップショットが自スキーマ内に定義されているものとみなされます。
<i>snapshot</i>	削除するスナップショットの名前を指定します。

## 使用上の注意

マスター表から、最後にリフレッシュされた単純スナップショットを削除すると、マスター表のスナップショット・ログのうち、削除されたスナップショットのリフレッシュに必要な行だけが自動的に削除されます。

マスター表を削除しても、その表を基礎とするスナップショットは自動的に削除されません。ただし、削除したマスター表を基礎とするスナップショットをリフレッシュしようとする、エラー・メッセージが戻されます。

次の文は、ユーザー **HQ** が所有するスナップショット **PARTS** を削除する例です。

```
DROP SNAPSHOT hq.parts;
```

## 関連項目

**CREATE SNAPSHOT** (4-283 ページ)

# DROP SNAPSHOT LOG

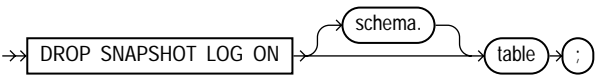
## 用途

データベースのスナップショット・ログを削除します。次の「使用上の注意」を参照してください。

## 前提条件

スナップショット・ログは表とトリガーで構成されます。スナップショット・ログを削除するには、**DROP TABLE**（4-402 ページ）に記載されている権限が必要です。また、スナップショット・ログのマスター表のトリガーを削除する権限も必要です。これらの権限の詳細は、**DROP TRIGGER**（4-406 ページ）を参照してください。

## 構文



## キーワードとパラメータ

---

<i>schema</i>	削除するスナップショット・ログとそのマスター表が含まれているスキーマを指定します。 <i>schema</i> の指定を省略すると、スナップショット・ログとマスター表は自スキーマ内にあるとみなされます。
<i>table</i>	削除するスナップショット・ログに関連付けられたマスター表の名前を指定します。

---

## 使用上の注意

スナップショット・ログを削除すると、そのスナップショット・ログのマスター表を基礎とするスナップショットの高速リフレッシュはできなくなります。完全リフレッシュを実行しなければなりません。スナップショットのリフレッシュ方法の詳細は、**CREATE SNAPSHOT**（4-283 ページ）を参照してください。

次の文は、PARTS マスター表のスナップショット・ログを削除する例です。

```
DROP SNAPSHOT LOG ON parts;
```

## 関連項目

**CREATE SNAPSHOT LOG**（4-294 ページ）

# DROP SYNONYM

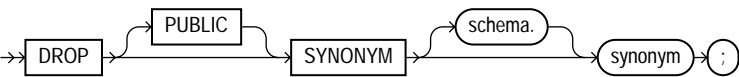
## 用途

データベースのシノニムを削除します。次の「使用上の注意」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、そのシノニムが自スキーマ内にあるか、**DROP ANY SYNONYM** システム権限を持っている必要があります。**PUBLIC** 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、**DROP ANY PUBLIC SYNONYM** システム権限を持っている必要があります。

## 構文



## キーワードとパラメータ

<b>PUBLIC</b>	パブリック・シノニムを削除する場合に指定します。 <b>PUBLIC</b> を指定すると、 <i>schema</i> は指定できません。
<i>schema</i>	削除するシノニムが含まれているスキーマを指定します。 <i>schema</i> の指定を省略すると、シノニムが自スキーマ内に定義されているものとみなされます。
<i>synonym</i>	削除するシノニムの名前を指定します。

## 使用上の注意

シノニムを削除して再作成することによって、シノニムの定義を変更できます。

次の文は、シノニム **MARKET** を削除する例です。

```
DROP SYNONYM market;
```

## 関連項目

[CREATE SYNONYM](#) (4-299 ページ)

# DROP TABLE

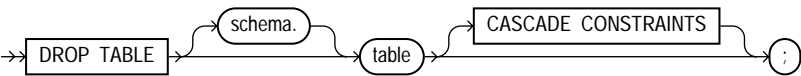
## 用途

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します。次の「使用上の注意」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、**DROP ANY TABLE** システム権限を持っている必要があります。

## 構文



## キーワードとパラメータ

<i>schema</i>	削除する表が含まれているスキーマを指定します。 <i>schema</i> を指定しないと、この表は自スキーマに存在するとみなされます。
<i>table</i>	削除する表またはオブジェクト表、索引構成表の名前を指定します。
CASCADE CONSTRAINTS	削除した表の主キーまたは一意キーを参照する参照整合性制約をすべて削除します。このような参照整合性制約があるときにこのオプションの指定を省略すると、エラー・メッセージが戻され表は削除されません。

## 使用上の注意

- 表を削除すると、Oracle によって次の操作が自動的に実行されます。
- 表上のすべての行を削除する（行を明示的に削除した場合と同じです）。
  - 表のすべての索引を、作成者やスキーマの所有者とは関係なく削除する。
  - 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データ・ブロックすべてを、表とその索引が格納されていた表領域に戻す。
  - 表がビューの実表の場合、またはストアド・プロシージャ、関数、パッケージで参照される場合は、これらのオブジェクトを削除するのではなく、無効にする。ユーザーが表を再作成するか、これらのオブジェクトを1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、関数、パッケージで参照する列とビューの定義にもともと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なければなりません。ビュー、ストアド・プロシージャ、関数、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は、これらの権限を再付与する必要はありません。

- 表がスナップショットのマスター表であれば、そのスナップショットは削除しない。このようなスナップショットの問合せはできますが、そのスナップショット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。

表を再作成する場合、最初にスナップショットを定義した問合せで選択した列をすべて含める必要があります。

- 表にスナップショット・ログがある場合、そのスナップショット・ログを削除する。

DROP CLUSTER コマンドに INCLUDING TABLES 句を指定すると、クラスタとその表すべてを削除できます。これによって表を 1 つずつ削除する手間が省けます。

例 . 次の文は、TEST\_DATA 表を削除する例です。

```
DROP TABLE test_data;
```

## 関連項目

DROP CLUSTER (4-381 ページ)

ALTER TABLE (4-105 ページ)

CREATE INDEX (4-233 ページ)

CREATE TABLE (4-303 ページ)

# DROP TABLESPACE

## 用途

データベースの表領域を削除します。次の「使用上の注意」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

<i>tablespace</i>	削除する表領域の名前を指定します。
INCLUDING CONTENTS	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には、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すると、エラー・メッセージが戻され、表領域は削除されません。  <b>注意：</b> 表領域に、パーティション表の全パーティションではなく一部のパーティションだけが含まれている場合、INCLUDING CONTENTS を指定しても、DROP TABLESPACE コマンドは正常に実行されません。
CASCADE CONSTRAINTS	この句の外で指定した表の参照整合性制約をすべて削除して、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します。このオプションを省略した場合に、参照整合性制約を指定していると、エラー・メッセージが戻され、表領域は削除されません。

## 使用上の注意

表領域の状態がオンラインと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を実行することをお勧めします。

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したりできません。ALTER USER コマンドを使用すると、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。

なお、SYSTEM 表領域は削除できません。

例 . 次の文は、MFRG 表領域とその内容を削除する例です。

```
DROP TABLESPACE mfrg
    INCLUDING CONTENTS
    CASCADE CONSTRAINTS;
```

## 関連項目

ALTER TABLESPACE (4-131 ページ)

CREATE DATABASE (4-214 ページ)

CREATE TABLESPACE (4-325 ページ)

# DROP TRIGGER

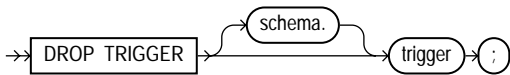
## 用途

データベースのデータベース・トリガーを削除します。次の「使用上の注意」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、**DROP ANY TRIGGER** システム権限を持っている必要があります。

## 構文



## キーワードとパラメータ

---

<i>schema</i>	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。 <i>schema</i> の指定を省略すると、トリガーは自スキーマ内に定義されているものとみなされます。
<i>trigger</i>	削除するトリガーの名前を指定します。

---

## 使用上の注意

データベース・トリガーを削除すると、**Oracle** によりデータベースからそのデータベース・トリガーが削除されてしまうため、トリガーを再起動できません。

次の文は、スキーマ **RUTH** 内のトリガー **REORDER** を削除する例です。

```
DROP TRIGGER ruth.reorder;
```

## 関連項目

[CREATE TRIGGER](#) (4-330 ページ)

# OBJ DROP TYPE

## 用途

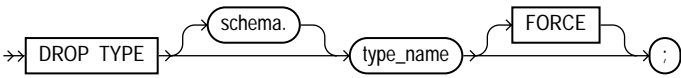
オブジェクト型、VARRAY 型、ネストした表型の仕様部と本体を削除します。オブジェクトの本体だけを削除する場合は、DROP TYPE BODY（4-409 ページ）で説明する DROP TYPE BODY コマンドを使用してください。次の「使用上の注意」も参照してください。

注意： このコマンドは、使用しているデータベース・サーバーに Oracle ObjectOption がインストールされている場合に限り有効です。

## 前提条件

削除するオブジェクト型または VARRAY 型、ネストした表型が自スキーマ内にあるか、DROP ANY TYPE システム権限を持っている必要があります。

## 構文



## キーワードとパラメータ

<i>schema</i>	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。 <i>schema</i> の指定を省略すると、トリガーは自スキーマ内に定義されているものとみなされます。
<i>type_name</i>	削除するオブジェクト型または VARRAY 型、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しか削除できません。
FORCE	表依存性または型依存性がある型でも強制的に削除します。

## 使用上の注意

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、VARRAY 型だけです。これはデフォルトの動作です。

---

---

**警告：** FORCE オプションを使用して、依存性のある型を削除することはお薦めしません。この操作は回復不可能であり、独立した表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 Server アプリケーション開発者ガイド』を参照してください。

---

---

**例 .** 次の文は、オブジェクト型 PERSON\_T を削除する例です。

```
DROP TYPE person_t;
```

## 関連項目

CREATE TYPE (4-342 ページ)

## OBJ DROP TYPE BODY

### 用途

オブジェクト型、VARRAY 型、ネストした表型の本体を削除します。次の「使用上の注意」を参照してください。

オブジェクト仕様部を削除する場合は、DROP TYPE（4-407 ページ）を参照してください。

---

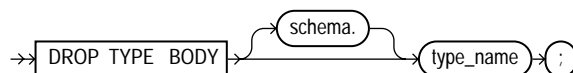
**注意：** このコマンドは、使用しているデータベース・サーバーに Oracle  
ObjectOption がインストールされている場合に限り有効です。

---

### 前提条件

削除するオブジェクト型本体が自スキーマ内にあり、かつ CREATE TYPE または CREATE ANY TYPE システム権限を持っている必要があります。あるいは、DROP ANY TYPE システム権限を持っている必要があります。

### 構文



### キーワードとパラメータ

<i>schema</i>	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。 <i>schema</i> の指定を省略すると、トリガーは自スキーマ内に定義されているものとみなされます。
<i>type_name</i>	削除するオブジェクト型の本体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本体しか削除できません。

### 使用上の注意

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。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、メンバー関数はコールできません。

次の文は、オブジェクト型の本体 RATIONAL を削除する例です。

```
DROP TYPE BODY rational;
```

### 関連項目

CREATE TYPE BODY（4-350 ページ）

# DROP USER

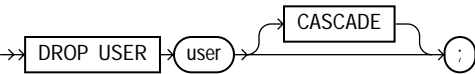
## 用途

データベース・ユーザーを削除します。また、オプションとしてそのユーザーのオブジェクトを削除します。次の「使用上の注意」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

<i>user</i>	削除するユーザーの名前を指定します。
CASCADE	ユーザーを削除する前に、そのユーザーのスキーマ内のオブジェクトをすべて削除します。所有するスキーマになんらかのオブジェクトが含まれているユーザーを削除する場合には、必ずこのオプションを指定します。

## 使用上の注意

Oracle では、所有するスキーマになんらかの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。このようなユーザーを削除する場合は、次のいずれかの操作を実行してください。

- ユーザーを削除する前に、そのユーザーのオブジェクトを明示的に削除する。
- CASCADE オプションを使用して、ユーザーと所有オブジェクトを同時に削除する。

CASCADE オプションを指定して自スキーマ内の表を削除した場合、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。CASCADE オプションを指定すると、他のスキーマ内にある次のようなオブジェクトは削除されず、無効となります。

- 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
- 削除された自スキーマ内のオブジェクトを問い合わせるストアド・プロシージャ、関数、パッケージ



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スナップショットは削除されませんが、**CASCADE** を指定すると、このようなスナップショットはリフレッシュできなくなります。

ユーザーが作成したロールは削除されません。

**例 1.** **BRADLEY** のスキーマ内にオブジェクトがない場合は、次の文を発行すれば **BRADLEY** を削除できます。

```
DROP USER bradley;
```

**例 2.** **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** オプションを指定してスキーマ **BRADLEY** とその中のオブジェクトを削除しなければなりません。

```
DROP USER bradley CASCADE;
```

## 関連項目

CREATE USER (4-353 ページ)

DROP TABLE (4-402 ページ)

CREATE TABLESPACE (4-325 ページ)

DROP TRIGGER (4-406 ページ)

DROP VIEW (4-412 ページ)

# DROP VIEW

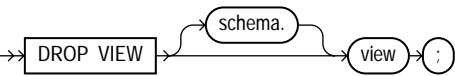
## 用途

データベース上のビューまたはオブジェクト・ビューを削除します。次の「使用上の注意」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、**DROP ANY VIEW** システム権限をもっている必要があります。

## 構文



## キーワードとパラメータ

<i>schema</i>	削除するビューが含まれているスキーマを指定します。 <b>schema</b> の指定を省略すると、ビューが自スキーマ内に定義されているものとみなされます。
<i>view</i>	削除するビューの名前を指定します。

## 使用上の注意

ビューを削除すると、削除したビューを参照するビューやシノニムが無効となりますが削除はされません。このようなビューや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 1 度有効にするビューを別に定義します。

ビューを削除して再作成することで、ビューの定義を変更することができます。

**例** . 次の文は、**VIEW\_DATA** ビューを削除する例です。

```
DROP VIEW view_data;
```

## 関連項目

- CREATE TABLE (4-303 ページ)
- CREATE VIEW (4-359 ページ)

---

CREATE SYNONYM (4-299 ページ)

## ENABLE 句

### 用途

表に関連付けられている整合性制約またはすべてのトリガーを使用可能にします。

- 制約を使用可能にすると、表内のすべてのデータにその制約が適用されます。表のデータはすべて、有効化された制約に合致しなければなりません。
- トリガーを使用可能にすると、その起動条件が満たされたときに必ずトリガーが起動されます。

単一トリガーを使用可能にするには、**ALTER TRIGGER** (4-139 ページ) の **ENABLE** オプションを使用してください。

「制約の有効化および無効化」(4-416 ページ) を参照してください。

### 前提条件

整合性制約を使用可能にする **ENABLE** 句は、**CREATE TABLE** 文または **ALTER TABLE** 文で指定できます。この方法で制約を使用可能にする場合、それぞれの文を発行するための各種権限が必要です。これらの権限の詳細は、**CREATE TABLE** (4-303 ページ) または **ALTER TABLE** (4-105 ページ) を参照してください。

**UNIQUE** 制約または **PRIMARY KEY** 制約を使用可能にすると、表が定義されているスキーマ内の一意キーまたは主キーの列に索引が作成されます。このような制約を使用可能にする場合、索引を作成するための各種権限が必要です。これらの権限の詳細は、**CREATE INDEX** (4-233 ページ) を参照してください。

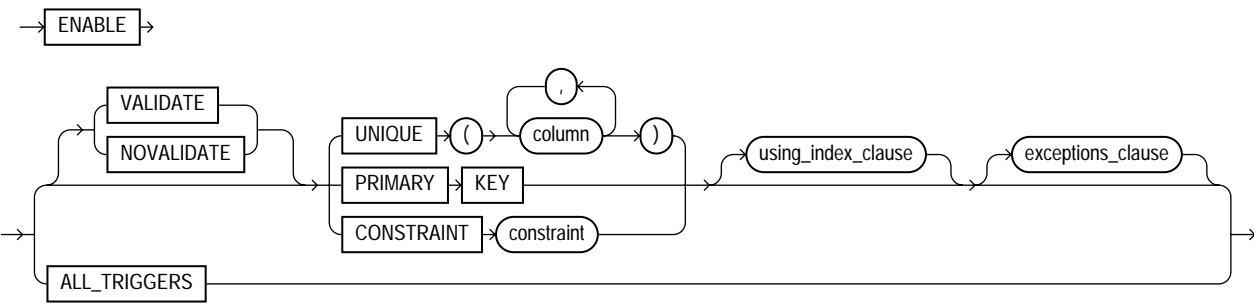
参照整合性制約を使用可能にする場合は、参照される **UNIQUE** 制約または **PRIMARY KEY** 制約が使用可能になっている必要があります。

**ENABLE** 句に指定する整合性制約については、次のいずれかの条件を満たす必要があります。

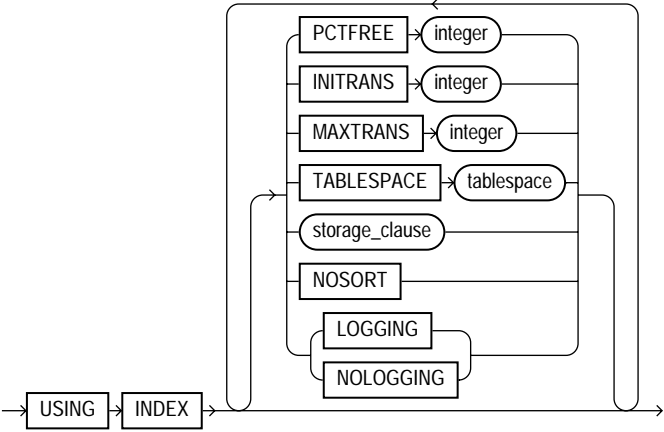
- 整合性制約が、その中の文に定義されている。
- 整合性制約がすでに定義済みで、かつ以前発行した文で使用禁止にされている。

トリガーを使用可能にする **ENABLE** 句は、**ALTER TABLE** 文で指定できます。**ENABLE** 句でトリガーを使用可能にする場合、この **ALTER TABLE** を発行するための各種権限が必要です。これらの権限の詳細は、**ALTER TABLE** (4-105 ページ) を参照してください。なお、トリガーが自スキーマ内になければなりません。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。

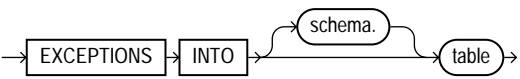
構文



using\_index\_clause::=



exceptions\_clause::=



storage\_clause: STORAGE 句（4-518 ページ）を参照。

キーワードとパラメータ

**VALIDATE** 制約データに対して新たに行うすべての挿入、削除、更新操作が制約に従っているか、および旧データもすべて制約に従っているかを確認します。制約が **ENABLE** かつ **VALIDATE** の状態にあると、現在およびその後のすべてのデータの有効性が保証されます。これはデフォルトです。「Oracle による整合性制約の検証方法」（4-417 ページ）も参照してください。

NOVALIDATE	制約データに対して新たに行うすべての挿入、更新、削除操作が制約に従っているかを確認します。すでに表にあるデータが制約に従っているかどうかは検証しません。
UNIQUE	指定した列または列の組合せに定義されている UNIQUE 制約を使用可能にします。
PRIMARY KEY	表の PRIMARY KEY 制約を使用可能にします。
CONSTRAINT	<i>constraint</i> に指定した整合性制約を使用可能にします。
<i>using_index_clause</i>	UNIQUE 制約または PRIMARY KEY 制約を適用するために自動生成される索引のパラメータを指定します。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、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の詳細は、CREATE TABLE（4-303 ページ）を参照してください。NOSORT および LOGGING/NOLOGGING の説明は、CREATE INDEX（4-233 ページ）を参照してください。  これらのパラメータは、UNIQUE 制約および PRIMARY KEY 制約が使用可能な場合にだけ使用してください。
<i>exceptions_clause:</i>	
EXCEPTIONS INTO	整合性制約に違反する行の情報を格納する表を指定します。このオプションを使用するには、指定する表が使用しているローカル・データベースに存在していなければなりません。 <i>schema</i> を指定しないと、自スキーマ内に例外の表があるとみなされます。「例外の検出方法」（4-418 ページ）を参照してください。
ALL TRIGGERS	表に関連付けられているトリガーをすべて使用可能にします。このオプションは、ALTER TABLE 文の ENABLE 句でしか指定できません。CREATE TABLE 文では指定できません。「トリガーの使用可能化」（4-420 ページ）を参照してください。

## 制約の有効化および無効化

制約には、次の 3 つの状態があります。

- DISABLE
- ENABLE NOVALIDATE
- ENABLE VALIDATE

制約を DISABLE から ENABLE VALIDATE の状態にするには、その表に対する排他ロックが必要です。既存のすべてのデータの妥当性が検査されるまでは、表に新しいデータを入れることができません。この動作のため、一度に 1 つの制約だけしか使用可能にできません。それぞれの新しい制約では、シリアル走査により既存のすべての行をチェックしなければなりません。（制約を同時に ENABLE VALIDATE 状態にするには、別々のセッションから複数の ALTER TABLE コマンドを発行する必要があります。）

表がロックされないようにするには、制約を ENABLE NOVALIDATE 状態にしてください。この状態では、表の新しい DML 文すべてが妥当性を検査されるので、表への同時アクセスを防ぐ必要がなくなります。

ENABLE NOVALIDATE は同時にその表のいくつかの制約を ENABLE VALIDATE 状態にすることもできます。Oracle による既存データの妥当性検査の各走査は、可能であればパラレルで行われます。

### 主キーと一意キーの制約を有効化する

主キー制約または一意キー制約を有効化すると、制約を施行する一意索引が自動的に作成されます。この索引は次に制約が使用禁止になるときに削除されるので、制約が使用可能になるたびに再作成されます。

この動作を避けるには、主キーと一意キーの制約を最初は使用禁止の状態で作成してください。その後、一意でない索引を作成するか、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。一意でない索引は制約が使用禁止のときも削除されないので、主キー制約や一意キー制約を使用可能にする操作はすべて、索引がすでに存在するためただちに行われます。この方法では、余分な索引も削除されます。

### 整合性制約を有効化する

制約を作成した場合、その制約を使用可能にできます（CREATE TABLE（4-303 ページ）および ALTER TABLE（4-105 ページ）参照）。または、ENABLE 句を使用して使用禁止の制約を使用可能にできます。並列性とパフォーマンスを最大限に維持するには、次の手順で制約を作成し、使用可能にしてください。

1. DISABLE の状態で制約を作成します。
2. 主キー制約および一意キー制約については、一意性を維持するため、一意でない索引を作成します。
3. 表に対するすべての制約を ENABLE NOVALIDATE 状態にします。これにより、表に新しく入力されるデータが制約に従っていることが保証されます。
4. 表に対するすべての制約を ENABLE VALIDATE 状態にします。

使用禁止状態の制約を使用可能にするには、ステップ 3 と 4 だけを実行します。

作成時の制約は、デフォルトにより ENABLE VALIDATE 状態になるので注意してください。上の手順を使用して、デフォルトの動作を避け、最大限のパフォーマンスが得られるようにしてください。

## Oracle による整合性制約の検証方法

整合性制約を ENABLE VALIDATE にすると、Oracle は表を走査し、表の中の既存の行に対して整合性制約を次のように適用します。

- 表のすべての行が整合性制約を満たす場合、整合性制約を ENABLE VALIDATE 状態にする。
- 表に整合性制約に違反する行がある場合、整合性制約は使用禁止のままにしておく。また、整合性制約が使用禁止のままであるというエラー・メッセージを戻す。

整合性制約が検証できるようになると、INSERT 文、UPDATE 文、DELETE 文で表のデータを変更しようとするたびに、Oracle はその整合性制約を次のように適用します。

- 新規データが整合性制約を満たす場合、文を実行する。
- 新規データが整合性制約に違反する場合、文は実行せず、整合性制約の違反を示すエラー・メッセージを生成する。

## 例外の検出方法

例外の検出方法例外とは、整合性制約に違反する表内の行です。整合性制約を ENABLE VALIDATE 状態にすると、Oracle に整合性制約に対する例外を検出させるよう要求できます。ENABLE 句に例外表を指定すると、それぞれの例外について、例外表に行が挿入されます。この例外表の行には、次の情報が含まれます。

- 例外の ROWID
- 整合性制約の名前
- スキーマと表の名前

配布媒体上の SQL スクリプトの中に、EXCEPTIONS という名前の例外表のサンプル定義があります。使用する例外表は、列のデータ型とデータ長がサンプルと同じでなければなりません。このスクリプトの一般的な名前は UTLEXCPT.SQL ですが、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。Oracle に対して、使用可能にした複数の整合性制約から 1 つの例外表へ例外を送信するように要求することができます。

索引構成表の場合、制約に違反する行は、ROWID ではなく主キーで識別されます。つまり、索引構成表用に作成される例外表は形式が異なることを意味します。整合性制約に違反する索引構成表の行を挿入するための EXCEPTIONS 表を作成する場合は、DBMS\_IOT パッケージ内の BUILD\_EXCEPTIONS\_TABLE プロシージャを使用します。

**例 1.** 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 表を作成する例です。

```
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;
```

ENABLE VALIDATE 句で例外表を指定する場合、この表への行の挿入に必要な権限を持っていなければなりません。これらの権限の詳細は、ALTER TABLE (4-105 ページ) を参照し



てください。検出された例外を調べるには、例外表の問合せに必要な権限を持っていないければなりません。これらの権限の詳細は、SELECT（4-486 ページ）を参照してください。

CREATE TABLE 文に、AS 句および EXCEPTIONS オプションを指定した ENABLE VALIDATE 句の両方の句がある場合、EXCEPTIONS オプションは無視されます。例外がある場合、表は作成されずエラー・メッセージが戻されます。

**例 2.** 次の文は、DEPT 表を作成し、PRIMARY KEY 制約を定義して、その制約を ENABLE VALIDATE 状態にする例です。

```
CREATE TABLE dept
  (deptno NUMBER(2) PRIMARY KEY,
   dname VARCHAR2(10),
   loc VARCHAR2(9) )
  TABLESPACE user_a
  ENABLE VALIDATE PRIMARY KEY;
```

**例 3.** 次の文は、EMP 表で FK\_DEPTNO という名前の整合性制約を ENABLE VALIDATE 状態にする例です。

```
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 表の各行がこの制約を満たしている必要があります。制約に違反する行があれば、制約は使用禁止のままになります。例外はすべて表 EXCEPT\_TABLE の中に表示されます。この表への問合せは次の文によって実行できます。

```
SELECT *
  FROM except_table;
```

この問合せの出力は、次のようになります。

ROW_ID	OWNER	TABLE_NAME	CONSTRAINT
AAAAZzAABAAABrXAAA	SCOTT	EMP	FK_DEPTNO

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, except_table
  WHERE emp.row_id except_table.row_id
        AND except_table.table_name = 'EMP'
        AND except_table.constraint = 'FK_DEPTNO';
```

FK\_DEPTNO 制約に例外があれば、この問合せの出力は次のようになります。

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
8001	JACK	CLERK	778	25-AUG-92	1100		70

例 4. 次の文は、EMP 表の 2 つの制約を妥当性検査を行わずに使用可能にする例です。

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

この文には、次の 2 つの ENABLE 句があります。

- 最初の ENABLE 句は、ENAME 列の UNIQUE 制約を ENABLE NOVALIDATE 状態にする。
- 2 番目の ENABLE 句は、NN\_ENAME という制約を ENABLE NOVALIDATE 状態にする。

この例では、表の中の各行が 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ると、エラー・メッセージが戻され、どちらの制約も使用禁止のままになります。

例 4 の制約を ENABLE VALIDATE 状態にするには、次の文を発行します。

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

この方法で制約を使用可能にすると、2 つの制約を同時に有効化できます。どちらの制約も ENABLE NOVALIDATE 状態にあるからです。また、この方法により各制約をパラレルで使用可能にできます。

トリガーの使用可能化

ALTER TABLE 文の ENABLE 句に ALL TRIGGER オプションを指定すると、表に関連付けられているトリガーすべてを使用可能にすることができます。トリガーを使用可能にすると、トリガー制限の条件を満たすトリガー文（トリガーを起動させる文）が発行されるたびにトリガーが実行されます。トリガーを作成すると、そのトリガーは自動的に使用可能になります。

例. 次の文は、EMP 表に関連付けられているトリガーすべてを使用可能にする例です。

```
ALTER TABLE emp
  ENABLE ALL TRIGGERS;
```

## 関連項目

ALTER TABLE (4-105 ページ)

ALTER TRIGGER (4-139 ページ)

CONSTRAINT 句 (4-185 ページ)

CREATE TABLE (4-303 ページ)

CREATE TRIGGER (4-330 ページ)

DISABLE 句 (4-375 ページ)

SET CONSTRAINT(S) (4-509 ページ)

STORAGE 句 (4-518 ページ)

## EXPLAIN PLAN

---

### 用途

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します。このコマンドによって、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、このコマンドによって文を実行するコストも決まります。「EXPLAIN PLAN の使用方法」(4-423 ページ) および「EXPLAIN PLAN およびパーティション表」(4-425 ページ)、「EXPLAIN PLAN およびパラレル DML」(4-427 ページ)を参照してください。

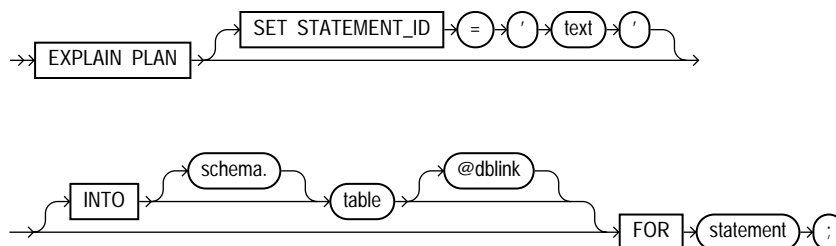
### 前提条件

EXPLAIN PLAN 文を実行するには、実行計画を格納するユーザー指定の既存の出力表へ行を挿入するための権限が必要です。これらの権限の詳細は、INSERT (4-449 ページ)を参照してください。

出力したい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は、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 文で作成された実行計画を検証する場合、出力表への問合せに必要な権限を持っていなければなりません。これらの権限の詳細は、SELECT (4-486 ページ)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	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。この句を省略すると、デフォルトで STATEMENT_ID 値が NULL に設定されます。
INTO	<p>出力表が定義されているスキーマ、出力表の名前、出力表があるデータベース名を指定します。この表は、EXPLAIN PLAN コマンドを使用する前に作成しておかなければなりません。<i>schema</i> を指定しないと、この表が自スキーマに存在するものとみなされます。</p> <p><i>dblink</i> 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。データベース・リンクの参照方法の詳細は、「リモート・データベース内のオブジェクトを参照」(2-50 ページ)を参照してください。Oracle の分散機能を使用している場合に限り、リモート出力表を指定できます。<i>dblink</i> の指定を省略すると、表がローカル・データベース上にあるものとみなされます。</p> <p>INTO 句自体を省略すると、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。</p>
FOR <i>statement</i>	実行計画生成の対象となる SELECT 文、INSERT 文、UPDATE 文、DELETE 文を指定します。

## EXPLAIN PLAN の使用方法

配布媒体上の 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、列の名前とデータ型がこのサンプル表と同じでなければなりません。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。

SET STATEMENT\_ID 句に指定した値は、実行計画の各行の STATEMENT\_ID 列に表示されます。この値を使用して、実行計画の行を出力表の中の他の行と区別することができます。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 の値を指定してください。

EXPLAIN PLAN コマンドはデータ操作言語 (DML) コマンドであり、データ定義言語 (DDL) コマンドではないため、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には、この文を指定したトランザクションをコミットしなければなりません。

EXPLAIN PLAN コマンドは、データ・ディクショナリ・ビューまたは動的性能表にアクセスする SQL 文の実行計画の決定には使用できません。

SQL トレース機能の一部として EXPLAIN PLAN コマンドを発行することもできます。SQL トレース機能の使用方法および実行計画の解釈方法の詳細は、『Oracle8 Server チューニング』を参照してください。

**例** . 次の EXPLAIN PLAN 文は、UPDATE 文の実行計画とコストを決定し、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 表に挿入する例です。

```
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');
```

次の文は、OUTPUT 表への問合せを実行して、実行計画とコストを戻す SELECT 文の例です。

```
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 およびパーティション表

パーティション化のための情報は、SQL 文の EXPLAIN PLAN コマンドによって作成される実行計画の各ステップ (EXPLAIN PLAN コマンドの出力表の各行) で提供されます。ここには次のような情報が含まれています。

- EXPLAIN PLAN の出力表の 3 つの列 : partition\_start、partition\_stop、partition\_id
- PARTITION というラベルが付いた EXPLAIN PLAN の出力表のステップ (行)
- TABLE ACCESS ステップおよび INDEX ステップでパーティション・オブジェクトが参照される場合、これらのステップを拡張したもの

### EXPLAIN PLAN の出力表のパーティション列

**partition\_start** 列および **partition\_stop** 列には、Oracle によって算定されたパーティションへのアクセス方法が表示され、アクセス可能なパーティションの範囲が示されます (範囲が認識される場合)。

**partition\_start** 列には、アクセスされるパーティションの範囲のうち、開始パーティションが示されます。次の値が表示されます。

- **NUMBER(*n*)**。開始パーティションが SQL コンパイラにより識別されており、そのパーティション番号が *n* であることを示します。
- **KEY**。開始パーティションがパーティション化キー値から実行時に識別されることを示します。
- **ROW LOCATION**。開始パーティション (終了パーティションと同じ) が、検索される各レコードの位置から実行時に計算されることを示します。レコードの位置は、ユーザーが取得するか、グローバル索引から取得されます。
- **INVALID**。アクセスされるパーティションの範囲が空であることを示します。

**partition\_stop** 列には、アクセスされるパーティションの範囲のうち、終了パーティションが示されます。次の値が表示されます。

- **NUMBER(*n*)**。終了パーティションが SQL コンパイラにより識別されており、そのパーティション番号が *n* であることを示します。
- **KEY**。開始パーティションがパーティション化キー値から実行時に識別されることを示します。
- **ROW LOCATION**。終了パーティション (開始パーティションと同じ) が、検索される各レコードの位置から実行時に計算されることを示します。レコードの位置は、ユーザーが取得するか、グローバル索引から取得されます。
- **INVALID**。アクセスされるパーティションの範囲が空であることを示します。

**partition\_id** 列には、対となる **partition\_start** 列と **partition\_stop** 列の値を計算したステップが示されます。

## EXPLAIN PLAN の出力表のパーティション・ステップ

PARTITION ステップでは、1つのパーティション・オブジェクト（表または索引）または1組の同一レベル・パーティション・オブジェクト（パーティション表およびそのローカル索引）に適用可能なパーティション境界を記述します。このパーティション境界は、PARTITION ステップの `partition_start` と `partition_stop` の値によって示されます。`partition_start` と `partition_stop` に表示される値は、NUMBER(n)、KEY、INVALID です。

PARTITION ステップの **options** 列には、次の値が表示されます。

- **CONCATENATED**。PARTITION ステップで、アクセスされたパーティションから戻された結果の集合を連結することを示します。
- **SINGLE**。アクセスされるパーティションの集合が、実行時に求められるパーティション1つだけで構成されることを示します。
- **EMPTY**。アクセスされるパーティションの集合が空であることを示します。

## EXPLAIN PLAN の出力表のステップ（行）に加えられた変更

パーティション表またはパーティション索引へのアクセスを記述する TABLE ACCESS ステップおよび INDEX ステップが、`partition_start` 列、`partition_stop` 列、`partition_id` 列にパーティション境界情報として提供されるように拡張されました。

パーティション境界が次に示すステップにより計算されている場合があります。

- 前回の PARTITION ステップ。この場合、`partition_start` 列値および `partition_stop` 列値には、PARTITION ステップの中にある値が複製されて入り、`partition_id` には、PARTITION ステップの ID が入ります。`partition_start` と `partition_stop` に表示される値は、NUMBER(n)、KEY、INVALID です。
- TABLE ACCESS ステップまたは INDEX ステップ自体。この場合、`partition_id` には、このステップの ID が入ります。`partition_start` および `partition_stop` に表示される値は、NUMBER(n)、KEY、ROW LOCATION(TABLE ACCESS の場合だけ)、INVALID です。

ROWID による表へのアクセスを示す TABLE ACCESS ステップの **options** 列には、次の値が表示されます。

- "BY USER ROWID"(ユーザー指定の ROWID を使って表中の行が検出される場合)
- "BY INDEX ROWID"(表がパーティション化されておらず、行が索引によって検出される場合)
- "BY GLOBAL INDEX ROWID"(表がパーティション化されており、行がグローバル索引だけで検出される場合)
- "BY LOCAL INDEX ROWID"(表がパーティション化されており、行が1つ以上のローカル索引あるいはいくつかのグローバル索引を使って検出される場合)

例 . STOCK\_NUM 列に従って8つにパーティション化されている表 STOCKS があり、STOCK\_NUM 列上にローカル同一キー索引 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、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;
```

EXPLAIN PLAN は、PLAN\_TABLE を出力表とするこの問合せを実行します。次の問合せによって、パーティション情報などの基本的な実行計画が得られます。

```
SELECT id, operation, options, object_name,  
       partition_start, partition_stop, partition_id FROM plan_table;
```

## EXPLAIN PLAN およびパラレル DML

EXPLAIN PLAN コマンドを使って、PARALLEL オプションが指定されている文の実行を決定した場合、結果として生成される実行計画はパラレルの実行となります。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、ユーザーが送ったパラレル 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。この状態は、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、この文はシリアルに実行されます。

文をパラレルに実行するためには、EXPLAIN PLAN コマンドをコミットするかロールバックしてから、パラレル DML 文を送る必要があります。

## 関連項目

『Oracle8 Server チューニング』

## Filespec

---

### 用途

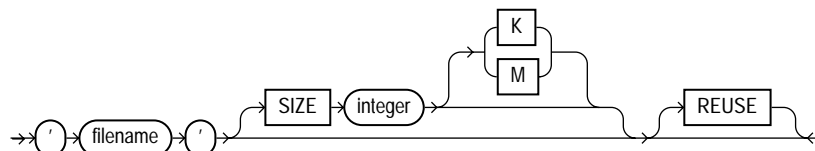
- ファイルをデータ・ファイルとして指定します。
  - 1 つ以上のファイルのグループを REDO ログ・ファイル・グループとして指定します。
- 使用例は、「例」(4-430 ページ) を参照してください。

### 前提条件

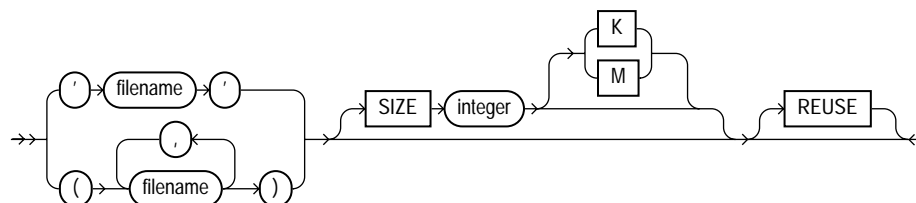
*filespec* は、CREATE DATABASE、ALTER DATABASE、CREATETABLESPACE、ALTER TABLESPACE の各コマンドに指定できます。そのコマンドの実行権限が必要です。これらの権限の詳細は、CREATE DATABASE (4-214 ページ) および ALTER DATABASE (4-15 ページ)、CREATE TABLESPACE (4-325 ページ)、ALTER TABLESPACE (4-131 ページ) を参照してください。

## 構文

filespec\_data\_files ::=



filespec\_redo\_log\_file\_groups ::=



## キーワードとパラメータ

<code>'filename'</code>	<p>データ・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・セットの文字など、シングルバイト文字だけを使用できます。マルチバイト文字は無効です。</p> <p>REDO ログ・ファイル・グループは、1 つ以上のメンバー、つまり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、完全な名前指定する必要があります。</p>
<code>SIZE integer</code>	<p>ファイルのサイズを指定します。このパラメータを指定しない場合には、ファイルがすでに定義されていることが必要です。表領域のサイズは、中に含まれるオブジェクトのサイズの合計より 1 ブロックだけ大きくなければなりません。K または M を使用して KB または MB 単位でもサイズを指定できます。</p>
<code>REUSE</code>	<p>既存のファイルを再利用することを指定します。指定したファイルがある場合は、そのサイズが SIZE パラメータの値と一致するか検査されます。指定したファイルがない場合は、そのファイルが作成されます。このオプションを指定しないと、既存のファイルはないとみなされ、新たなファイルが作成されます。</p> <p>REUSE オプションは、SIZE オプションとともに指定する場合だけに有効です。SIZE オプションを指定しないと、Oracle は既存のファイルがあるとみなします。</p> <p><b>注意：</b>既存のファイルが使用される場合には、そのファイルに入っていた内容は失われます。</p>

## 例

**例 1.** 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 *filespec* は、REDO ログ・ファイル・グループに GROUP 1 の値を指定します。このグループには 'DISKA:LOG1.LOG' および 'DISKB:LOG1.LOG' というメンバーがあり、サイズはそれぞれ 50KB です。

LOGFILE 句の2番目の *filespec* は、REDO ログ・ファイル・グループに GROUP 2 の値を指定します。このグループには 'DISKA:LOG2.LOG' および 'DISKB:LOG2.LOG' というメンバーがあり、サイズはそれぞれ 50KB です。

DATAFILE 句の *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。

これらの *filespecs* はすべて SIZE パラメータの値を指定し、REUSE オプション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 が新たにファイルを作成します。

**例 2.** 次の文は、2つのメンバーからなる新たな 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、新たな REDO ログ・ファイル・グループに GROUP 3 の値を指定します。この新たなグループには '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 オプションが指定されているため、各メンバーは既存のファイルとなります。また、そのサイズは 50KB でなければなりません。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。

**例 3.** 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、'DISK:STOCK1.DAT'、'DISK:STOCK2.DAT'、'DISK: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** パラメータは指定されていないため、それぞれのファイルは、既存のファイルでなければなりません。

**例 4.** 次の文は、**STOCKS** 表領域を変更し、新たなデータ・ファイルを追加します。

```
ALTER TABLESPACE stocks
    ADD DATAFILE 'disk:stock4.dat' REUSE;
```

*filespec* は 'DISK: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、すでに定義されていなければなりません。また、**REUSE** オプションは無効です。

## 関連項目

CREATE DATABASE (4-214 ページ)

ALTER DATABASE (4-15 ページ)

CREATE TABLESPACE (4-325 ページ)

ALTER TABLESPACE (4-131 ページ)

# GRANT( システム権限とロール)

## 用途

ユーザーやロールに対してシステム権限やロールを付与します。オブジェクト権限を付与するには、次の項で説明する **GRANT** コマンド（オブジェクト権限）を使用します。詳細は、「使用上の注意」（4-433 ページ）を参照してください。使用例は、「例」（4-440 ページ）を参照してください。

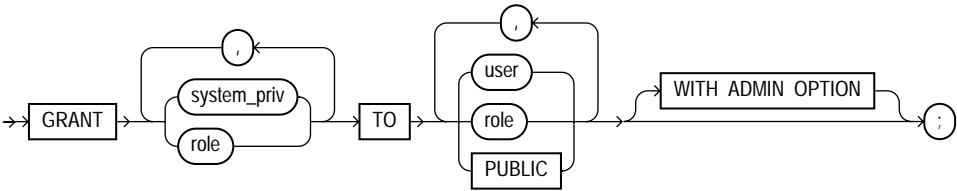
**注意：** コマンドと句の説明で**OBJ**の印が前に付いている箇所は、Oracle Object Option がデータベース・サーバーにインストールされている場合にだけ有効です。

## 前提条件

システム権限を付与するには、**ADMIN OPTION** 付きのシステム権限を付与されているか、**GRANT ANY PRIVILEGE** システム権限が付与されていない必要があります。

ロールを付与するには、**ADMIN OPTION** 付きのロールを付与されているか、**GRANT ANY ROLE** システム権限を付与されているか、付与するロールが自分で作成したロールであることが必要です。「その他の認可メソッド」（4-439 ページ）を参照してください。

## 構文



## キーワードとパラメータ

<i>system_priv</i>	付与するシステム権限です。
<i>role</i>	付与するロールです。
TO	システム権限やロールを付与する、ユーザーまたはロールを指定します。
PUBLIC	システム権限またはロールをすべてのユーザーに付与します。

WITH ADMIN OPTION	他のユーザーまたはロールに対して、システム権限またはロールを付与します。ADMIN OPTION を使用してロールを付与した場合は、ロールを付与されたユーザーまたはロールは、 そのロールを変更または削除できます。「ADMIN OPTION の付与」(4-439 ページ)を参照して ください。
----------------------	---

使用上の注意

この形式の GRANT コマンドを使うと、システム権限とロールの両方をユーザー、ロール、PUBLIC に付与できます。表 4-10 は、どのユーザーまたはロールにどの認可が与えられるかを示しています。

表 4-10 許可されるデータベース権限

ユーザー・タイプ→ _____ ロール・タイプ	パスワードで 識別される ユーザー	外部的に識別 されるユー ザー	グローバルに 識別される ユーザー	ローカル・ ロール	外部ロール	グローバル・ ロール
権限	可	可	可	可	可	可
ローカル・ロール	可	可	可	可	可	可
外部ロール	可	可	不可	可	可	不可
グローバル・ロール	不可	不可	不可	不可	不可	不可

**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ただちに行使できます。

**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、その権限をただちに行使できます。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行使できます。

**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ただちに実行できます。

**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ただちに使用可能にして、ロールの権限ドメインに登録されている権限を行使できます。

**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、それを使用可能にすると、付与されたロールの権限ドメイン内の権限を行使できます。

**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て、ロールの権限ドメインに登録されている権限をただちに行使できます。

さらに、次の制限が適用されます。

- 権限またはロールは 1 度しか付与できません。
- したがって、ユーザーおよびロール、PUBLIC は、TO 句に 1 度しか指定できません。
- ロールは交互に付与できません。たとえば、ロール BANKER にロール TELLER を付与した後で、逆に TELLER を BANKER に付与できません。
- また、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、どん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 を、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。

表 4-11 に、システム権限とその権限が許可する操作を示します。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。

表 4-11 システム権限

システム権限	許可される操作
ALTER ANY CLUSTER	任意のスキーマ内の任意のクラスタの変更
ALTER ANY INDEX	任意のスキーマ内の任意の索引の変更
ALTER ANY PROCEDURE	任意のスキーマ内の任意のストアド・プロシージャおよびストアド・ファンクション、ストアド・パッケージの変更
ALTER ANY ROLE	データベース内の任意のロールの変更
ALTER ANY SEQUENCE	データベース内の任意の順序の変更
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
ALTER ANY TABLE	スキーマ内の任意の表またはビューの変更
<b>OBJ</b> ALTER ANY TYPE	任意のスキーマ内の任意の型の変更
ALTER ANY TRIGGER	任意のスキーマ内のデータベース・トリガーの使用可能と使用禁止の切替え、コンパイル
ALTER DATABASE	データベースの変更
ALTER PROFILE	プロファイルの変更
ALTER RESOURCE COST	セッション・リソースに対するコストの設定
ALTER ROLLBACK SEGMENT	ロールバック・セグメントの変更
ALTER SESSION	ALTER SESSION 文の発行
ALTER SYSTEM	ALTER SYSTEM 文の発行



表 4-11 システム権限

システム権限	許可される操作
ALTER TABLESPACE	表領域の変更
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。 <ul style="list-style-type: none"> <li>• 他のユーザーのパスワードまたは認証方法の変更</li> <li>• <b>任意</b>の表領域に対する割当て制限の設定</li> <li>• デフォルトの表領域と一時表領域の設定</li> <li>• プロファイルとデフォルト・ロールの割当て</li> </ul>
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( スキーマ・オブジェクト ) 文による任意のスキーマ内の任意のオブジェクトの監査
AUDIT SYSTEM	AUDIT(SQL 文) 文の発行
BACKUP ANY TABLE	Export ユーティリティを使用して、他のユーザーのスキーマからオブジェクトの増分エクスポートが可能
BECOME USER	別のユーザーになる（全データベース・インポートを実行するユーザーに必要）。
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
CREATE ANY CLUSTER	任意のスキーマ内でのクラスタの作成。CREATE ANY TABLE と同じように機能する。
CREATE ANY DIRECTORY	任意のスキーマ内でのディレクトリ・データベース・オブジェクトの作成
CREATE ANY INDEX	任意のスキーマ内の任意の表に対する任意のスキーマ内での索引の作成
CREATE ANY LIBRARY	任意のスキーマ内での外部プロシージャまたは関数ライブラリの作成
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
CREATE ANY SEQUENCE	任意のスキーマ内での順序の作成
CREATE ANY SNAPSHOT	任意のスキーマ内でのスナップショットの作成

表 4-11 システム権限

システム権限	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 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。
CREATE ANY TRIGGER	任意のスキーマ内の表に対応付けられた任意のスキーマ内でのデータベース・トリガーの作成
<b>OBJ</b> CREATE ANY TYPE	任意のスキーマでの型および型本体の作成
CREATE ANY VIEW	任意のスキーマ内でのビューの作成
CREATE CLUSTER	自スキーマ内でのクラスタの作成
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 ANY LIBRARY	自スキーマ内での外部プロシージャまたは関数ライブラリの作成
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 PROFILE	プロファイルの作成
CREATE PUBLIC DATABASE LINK	パブリック・データベース・リンクの作成
CREATE PUBLIC SYNONYM	パブリック・シノニムの作成
CREATE ROLE	ロールの作成
CREATE ROLLBACK SEGMENT	ロールバック・セグメントの作成
CREATE SEQUENCE	自スキーマ内での順序の作成
CREATE SESSION	データベースへの接続
CREATE SNAPSHOT	自スキーマ内でのスナップショットの作成
CREATE SYNONYM	自スキーマ内でのシノニムの作成
CREATE TABLE	自スキーマ内での表の作成。なお、表を作成するには、その表を作成する表領域に対する割当て制限が必要。
CREATE TABLESPACE	表領域の作成
CREATE TRIGGER	自スキーマ内でのデータベース・トリガーの作成
<b>OBJ</b> CREATE TYPE	自スキーマ内での型および型本体の作成

表 4-11 システム権限

システム権限	許可される操作
CREATE USER	<p>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。</p> <ul style="list-style-type: none"> <li>• 任意の表領域に対する割当て制限の設定</li> <li>• デフォルトの表領域と一時表領域の設定</li> <li>• CREATE USER 文の一部としてのプロファイルの割当て</li> </ul>
CREATE VIEW	自スキーマ内でのビューの作成
DELETE ANY TABLE	<ul style="list-style-type: none"> <li>• 任意のスキーマ内の表またはビューからの行の削除</li> <li>• 任意のスキーマ内の表の切捨て</li> </ul>
DROP ANY CLUSTER	任意のスキーマ内のクラスタの削除
DROP ANY DIRECTORY	ディレクトリ・データベース・オブジェクトの削除
DROP ANY INDEX	任意のスキーマ内の索引の削除
DROP ANY LIBRARY	任意のスキーマ内の外部プロシージャまたは関数ライブラリの削除
DROP ANY PROCEDURE	任意のスキーマ内のストアド・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
DROP ANY ROLE	ロールの削除
DROP ANY SEQUENCE	任意のスキーマ内の順序の削除
DROP ANY SNAPSHOT	任意のスキーマ内のスナップショットの削除
DROP ANY SYNONYM	任意のスキーマ内のプライベート・シノニムの削除
DROP ANY TABLE	任意のスキーマ内の表の削除
DROP ANY TRIGGER	任意のスキーマ内のデータベース・トリガーの削除
<b>OBJ</b> DROP ANY TYPE	任意のスキーマ内のオブジェクト型およびオブジェクト型本体の削除
DROP ANY VIEW	任意のスキーマ内のビューの削除
DROP LIBRARY	外部プロシージャまたは関数ライブラリの削除
DROP PROFILE	プロファイルの削除

表 4-11 システム権限

システム権限	許可される操作
DROP PUBLIC DATABASE LINK	パブリック・データベース・リンクの削除
DROP PUBLIC SYNONYM	パブリック・シノニムの削除
DROP ROLLBACK SEGMENT	ロールバック・セグメントの削除
DROP TABLESPACE	表領域の削除
DROP USER	ユーザーの削除
EXECUTE ANY PROCEDURE	<ul style="list-style-type: none"> <li>・ プロシージャまたは関数の実行（スタンドアロンまたはパッケージ）</li> <li>・ 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</li> </ul>
<b>OBJ</b> EXECUTE ANY TYPE	オブジェクト型の使用および参照。また、任意のスキーマ内の任意の型のメソッドを呼出す。 EXECUTE ANY TYPE 権限は、特定のユーザーに付与しなければなりません。ロールには、EXECUTE ANY TYPE 権限は付与できません。
FORCE ANY TRANSACTION	<ul style="list-style-type: none"> <li>・ 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。</li> <li>・ 分散トランザクション・エラーを意図的に発生させる。</li> </ul>
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
GRANT ANY ROLE	データベース内の任意のロールの付与
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
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と終了の制御
RESTRICTED SESSION	Server Manager の STARTUP RESTRICT コマンドによるインスタンスの起動後のログオン
SELECT ANY SEQUENCE	任意のスキーマ内の順序の参照
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 4-11 システム権限

システム権限	許可される操作
SYSDBA	<ul style="list-style-type: none"><li>• Server Manager の STARTUP コマンドおよび SHUTDOWN コマンドの実行</li><li>• ALTER DATABASE OPEN/MOUNT/BACKUP および</li><li>• CREATE DATABASE、</li><li>• ARCHIVELOG、RECOVERY および</li><li>• RESTRICTED SESSION 権限の付与</li></ul>
SYSOPER	<ul style="list-style-type: none"><li>• Server Manager の STARTUP コマンドおよび SHUTDOWN コマンドの実行</li><li>• ALTER DATABASE OPEN/MOUNT/BACKUP および</li><li>• ARCHIVELOG、RECOVERY</li><li>• RESTRICTED SESSION 権限の付与</li></ul>
UNLIMITED TABLESPACE	任意の表領域の無制限な使用。この権限は、設定されている任意の割当て制限を上書きします。ユーザーからこの権限を取り消した場合、権限受領者のスキーマ・オブジェクトはそのまま残るが、表領域の割当て制限が許可されない限り、それ以上表領域の割当てを行うことはできない。このシステム権限をロールに付与することはできない。
UPDATE ANY TABLE	任意のスキーマ内の表またはビューの行の更新

ADMIN OPTION の付与

ADMIN OPTION を指定していない場合でも、改めて ADMIN OPTION を指定できます。たとえば、ADMIN OPTION を指定しないでロールまたはシステム権限を付与したユーザーに、後で ADMIN OPTION を指定してその権限とロールを付与すると、そのユーザーはその権限とロールに関して ADMIN OPTION を持つことになります。

しかし、すでに ADMIN OPTION を指定している場合、改めて ADMIN OPTION を指定しないでロールまたはシステム権限を付与しても、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と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。

その他の認可メソッド

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。オペレーティング・システムによっては、オペレーティング・システム・ユーザーに対してオペレーティング・システム権限を付与する機能を備えています。この機能を使用して Oracle ユーザーに対して初期化パラメータ OS\_ROLES でロールを

付与できます。オペレーティング・システム機能を使用してユーザーにロールを付与する場合は、**GRANT** コマンドを使用してもそのユーザーにロールは付与できませんが、そのユーザーにシステム権限を付与できます。また、別のロールに対してシステム権限やロールを付与できます。

その他の認可メソッドの詳細は、『Oracle8 Server 管理者ガイド』を参照してください。

### 例

**例 1.** 次の文は、**RICHARD** に **CREATE SESSION** システム権限を付与します。これにより、**RICHARD** は Oracle にログオンできます。

```
GRANT CREATE SESSION
TO richard;
```

**例 2.** 次の文は、ロール **TRAVEL\_AGENT** に **CREATE TABLE** システム権限を付与します。

```
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。

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** ロールに **TRAVEL\_AGENT** ロールが付与され、**EXECUTIVE** の権限ドメインに **CREATE TABLE** システム権限が登録されます。

次の文は、**THOMAS** に **ADMIN OPTION** 付きの **EXECUTIVE** ロールを付与します。

```
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行使。
- 他のユーザーに対するそのロールの付与と取消し。
- ロールの削除。

## 関連項目

ALTER USER (4-148 ページ)

CREATE USER (4-353 ページ)

GRANT( オブジェクト権限 ) (4-442 ページ)

REVOKE( システム権限とロール ) (4-472 ページ)

## GRANT( オブジェクト権限 )

### 用途

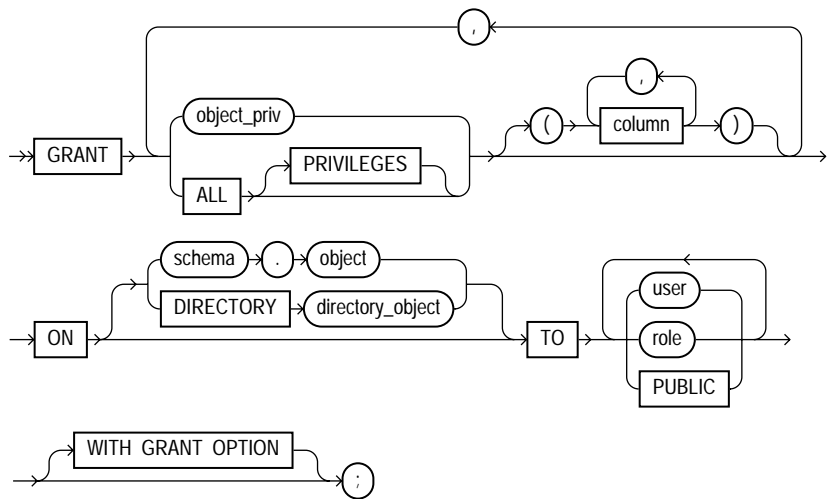
特定のオブジェクトの権限をユーザーまたはロールに付与します。システム権限またはロールを付与するには、前の項で説明した **GRANT** コマンド（システム権限とロール）を使用します。「使用上の注意」（4-444 ページ）も参照してください。使用例は、「例」（4-447 ページ）を参照してください。

### 前提条件

オブジェクトの所有者でなければなりません。そうでない場合は、オブジェクトの所有者から **GRANT OPTION** 付きでオブジェクト権限を付与されていなければなりません。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



キーワードとパラメータ

<i>object_priv</i>	付与するオブジェクト権限です。次のいずれかの値を指定します。 ALTER EXECUTE INDEX INSERT READ REFERENCES SELECT UPDATE
ALL [PRIVILEGES]	GRANT OPTION によって権限付与されているオブジェクトに対するすべての権限を付与します。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 によってそ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。)
<i>column</i>	権限を付与する表またはビューの列を指定します。INSERT、REFERENCES、UPDATE の各権限を付与する場合に限り、列を指定できます。列を指定しないと、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。
ON	権限を付与するオブジェクトを指定します。

	DIRECTORY	DBA により権限付与の対象となっている <i>directory_object</i> を示します。 <i>directory_object</i> をスキーマ名で修飾することはできません。「ディレクトリ権限」(4-447 ページ) を参照してください。  CREATE DIRECTORY (4-226 ページ) を参照してください。
	<i>object</i>	権限を付与するスキーマ・オブジェクトを指定します。 <i>object</i> を <i>schema</i> で修飾しないと、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。 <ul style="list-style-type: none"><li>表</li><li>ビュー</li><li>順序</li><li>プロシージャ、関数、パッケージ</li><li>スナップショット</li><li>表またはビュー、順序、スナップショット、プロシージャ、関数、パッケージのシノニム (「シノニム権限」(4-447 ページ) も参照)</li><li>ライブラリ</li><li>オブジェクト型</li></ul>
TO		オブジェクト権限を付与するユーザーまたはロールを指定します。
	PUBLIC	オブジェクト権限をすべてのユーザーに付与します。
WITH GRANT OPTION		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できることを指定します。WITH GRANT OPTION は、ロールではなくユーザーまたは PUBLIC に付与します。

使用上の注意

この GRANT 文を使用してユーザー、ロール、PUBLIC にオブジェクト権限を付与できます。各データベース・オブジェクト権限を管理者から付与されたユーザーは、権限の対象となっているオブジェクトに対するいくつかの操作を実行できます。表 4-12 は、各種オブジェクトについて付与できるオブジェクト権限をまとめたものです。

表 4-12 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、 関数、 パッケージ	スナップ ショット	ディレクトリ	ライブラリ
ALTER	X		X				
DELETE	X	X			X <sup>a</sup>		
EXECUTE				X			X
INDEX	X						
INSERT	X	X			X <sup>a</sup>		

表 4-12 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、 関数、 パッケージ	スナップ ショット	ディレクトリ	ライブラリ
READ						X	
REFERENCES	X						
SELECT	X	X	X		X		
UPDATE	X	X			X <sup>a</sup>		
DELETE および INSERT、UPDATE の各権限は更新可能なスナップショットについてだけ付与できます。							

**ユーザーに権限を付与した場合：**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ただちに行使できます。

**ロールに権限を付与した場合：**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、その権限をただちに行使できます。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行使できます。

**PUBLIC に権限を付与した場合：**各ユーザーの権限ドメインに権限が追加されます。すべてのユーザーがこの権限をただちに行使できます。

- 権限は、付与リストに 1 度しか指定できません。
- ユーザーおよびロールは、TO 句に 1 度しか指定できません。

表 4-13 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。

表 4-13 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
<b>表権限</b> は、表について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 コマンド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 コマンドを使用して表定義を変更できます。
DELETE	DELETE コマンドを使用して表の行を削除できます。  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 コマンドを使用して表の索引を作成できます。
INSERT	INSERT コマンドを使用して表に新たな行を追加できます。

表 4-13 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
REFERENCES	表を参照する制約を作成できます。この権限はロールには付与できません。
SELECT	SELECT コマンドを使用して表の間合せができます。
UPDATE	UPDATE コマンドを使用して表のデータを変更できます。 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。
<p><b>ビュー権限</b>は、ビューについての操作を許可します。上のいずれかのオブジェクト権限がある場合には、LOCK TABLE コマンド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。</p> <p>ビューの権限を付与するには、そのビューのすべての実表に関して GRANT OPTION 付きの権限が必要です。</p>	
DELETE	DELETE コマンドを使用してビューの行を削除できます。
INSERT	INSERT コマンドを使用してビューに新たな行を追加できます。
SELECT	SELECT コマンドを使用してビューの間合せができます。
UPDATE	UPDATE コマンドを使用してビューのデータを変更できます。
<p><b>順序権限</b>は、順序についての操作を許可します。</p>	
ALTER	ALTER SEQUENCE コマンドを使用して順序定義を変更できます。
SELECT	CURRVAL 疑似列と NEXTVAL 疑似列を使用して、順序の値を検査し、増分できます。
<p><b>プロシージャおよび関数、パッケージ権限</b>は、プロシージャまたは関数、パッケージについての操作を許可します。</p>	
EXECUTE	プロシージャおよび関数を実行、またはパッケージの仕様部に宣言されたプログラム・オブジェクトにアクセスできます。
<p><b>スナップショット権限</b>は、スナップショットについての操作を許可します。</p>	
SELECT	SELECT コマンドを使用してスナップショットの間合せができます。
<p><b>シノニム権限</b>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。次の「シノニム権限」を参照してください。</p>	
<p><b>ディレクトリ権限</b>があると、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ファイルに安全にアクセスできます。次の「ディレクトリ権限」を参照してください。</p>	
READ	ディレクトリ内のファイルを読み取ることができます。

## シノニム権限

シノニムに対して付与されるオブジェクト権限は、シノニムの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のと同じです。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、その権限を行使できます。

## ディレクトリ権限

ディレクトリに対するオブジェクト権限の場合は、ディレクトリ・オブジェクトをポインタとして使うことにより、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完全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、Oracle8 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・アクセス権も持っていなければなりません。

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、Oracle8 はファイル運用時のセキュリティを実現できます。

## 例

**例 1.** 次の文は、ユーザー SCOTT にディレクトリ BFILE\_DIR1 に対する READ を GRANT OPTION 付きで付与します。

```
GRANT READ ON DIRECTORY bfile_dir1 TO scott
WITH GRANT OPTION;
```

**例 2.** 次の文は、ユーザー JONES に対し、表 BONUS についてのすべての権限を GRANT OPTION 付きで付与します。

```
GRANT ALL ON bonus TO jones
WITH GRANT OPTION;
```

この結果、JONES は次の操作ができます。

- 表 BONUS に対するすべての権限の行使
- 他のユーザーまたはロールに対する、表 BONUS についての権限の付与

**例 3.** 次の文は、ビュー GOLF\_HANDICAP についての SELECT 権限および UPDATE 権限を全ユーザーに付与します。

```
GRANT SELECT, UPDATE
ON golf_handicap TO PUBLIC;
```

この結果、すべてのユーザーが、ゴルフのハンディのビューの問合せおよび更新をできます。

**例 4.** 次の文は、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;
```

**例 5.** 次の文は、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** 権限を付与します。

```
GRANT REFERENCES (empno), UPDATE (empno, sal, comm)
ON scott.emp
TO blake;
```

この結果、**BLAKE** は **EMPNO** および **SAL**、**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** 文にはこれらの列しか指定されていないため、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。

## 関連項目

[GRANT\(システム権限とロール\)](#) (4-432 ページ)

[REVOKE\(スキーマ・オブジェクト権限\)](#) (4-475 ページ)

---

# INSERT

## 用途

行を追加します。行を追加できるのは、次のものに対してです。

- 表
- ビューの実表
- パーティション表のパーティション
- **OBJ** オブジェクト表
- **OBJ** オブジェクト・ビューの実表

挿入の使用例は、「例」(4-453 ページ)を参照してください。

---

**注意：** コマンドと句の説明で**OBJ**の印が前に付いている箇所は、Oracle Object Option がデータベース・サーバーにインストールされている場合にだけ有効です。

---

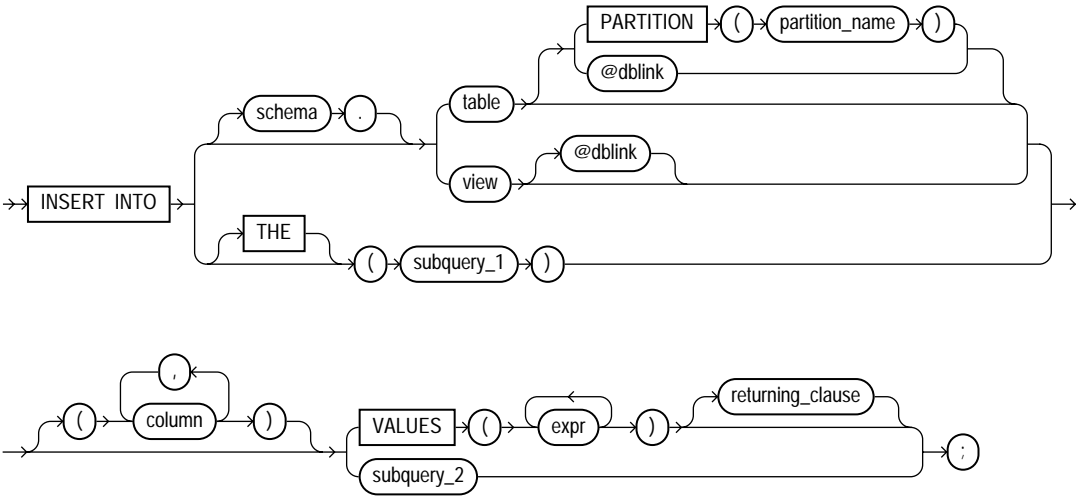
## 前提条件

表に行を挿入する場合は、その表が自スキーマ内にいることが必要です。自スキーマ内にならない場合は、その表の **INSERT** 権限が必要です。

ビューの実表に行を挿入する場合は、ビューが定義されているスキーマの所有者がその実表の **INSERT** 権限を持っていないけません。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、そのビューの **INSERT** 権限が必要です。

**INSERT ANY TABLE** システム権限があれば、任意の表または任意のビューの実表に行を挿入できます。

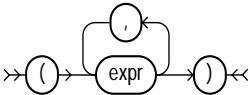
構文



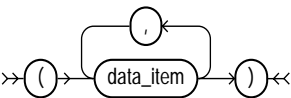
returning\_clause ::=



expr\_list ::=



data\_item\_list ::=





キーワードとパラメータ

*schema* 表またはビューが定義されているスキーマです。*schema* を指定しないと、表またはビューが自スキーマにあるとみなされます。



---

<i>table / view</i>	行を挿入する表またはオブジェクト表の名前です。ビューまたはオブジェクト・ビューを指定すると、Oracle はそのビューの実表に行を挿入します。「ビューへの挿入」(4-452 ページ) を参照してください。
<b>PARTITION</b> ( <i>partition_name</i> )	<i>table</i> にパーティション・レベルの行を挿入します。 <i>partition_name</i> は、挿入先の表内にあるパーティションの名前です。  <b>注意</b> :  このオプションは、オブジェクト表とオブジェクト・ビューに対しては無効です。
<i>dblink</i>	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です。データベース・リンクの参照については、「スキーマ・オブジェクトと部分を参照する」(2-47 ページ) を参照してください。Oracle の分散機能を使用している場合にだけ、リモート表またはリモート・ビューに行を挿入できます。  <i>dblink</i> を指定しないと、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。
 <b>THE</b>	副問合せが戻す列値がスカラー値でなく、ネストした表であることを示します。接頭辞 <b>THE</b> の付いた副問合せはフラット化した副問合せと呼ばれます。「フラット化した副問合せの使用」(4-528 ページ) を参照してください。
<i>subquery_1</i>	ビューと同様に扱われる副問合せです。「副問合せ」(4-525 ページ) を参照してください。
<i>column</i>	表またはビューの列です。挿入された行の各列には、 <b>VALUES</b> 句または副問合せの値が代入されます。  表のいずれかの列を指定しないと、挿入された行の列の値には、表作成時に指定したデフォルト値が使用されます。列リスト自体を指定しない場合は、 <b>VALUES</b> 句または問合せに、表の列をすべて指定しなければなりません。
<b>VALUES</b>	表またはビューに挿入する行の値を指定します。構文の説明については、「式」(3-73 ページ) を参照してください。なお、値は列リスト内の各列について <b>VALUES</b> 句に指定しなければなりません。「 <b>VALUES</b> 句および副問合せ」(4-452 ページ) を参照してください。
<i>subquery_2</i>	表に挿入される行を戻す副問合せです。副問合せの選択リストには、 <b>INSERT</b> 文の列リストと同数の列が指定されていなければなりません。「副問合せ」(4-525 ページ) を参照してください。
<b>RETURNING</b>	<b>INSERT</b> に影響される行を取り出します。スカラー、LOB 型、ROWID 型、REF 型だけが取り出されます。「 <b>RETURNING</b> 句」(4-453 ページ) も参照してください。
<i>expr</i>	「式」(3-73 ページ) で説明した構文です。 <i>data_item_list</i> の各変数について列の式を指定してください。
<b>INTO</b>	変更された行の値を、 <i>data_item_list</i> に指定する変数に格納することを示します。
<i>data_item</i>	取り出された <i>expr</i> 値を格納する PL/SQL 変数またはバインド変数です。

---

パラレル DML やリモート・オブジェクトでは **RETURNING** 句を使用できません。「パラレル DML」(4-452 ページ) も参照してください。

---

## VALUES 句および副問合せ

VALUES 句を指定した INSERT 文によって、VALUES 句に指定した値が入った単一行が表に挿入されます。

VALUES 句のかわりに副問合せを指定した INSERT 文では、副問合せが戻した行がすべて挿入されます。Oracle は副問合せを処理して戻された行を表に挿入します。副問合せで選択された行が 1 つもない場合は、表に行は挿入されません。副問合せによって、INSERT 文の対象となる表を含む任意の表およびビュー、スナップショットを参照できます。

INSERT 文の列リスト内に指定する列数は、VALUES 句内の値の数または副問合せにより選択された列数と同じでなければなりません。列リストを指定しない場合は、VALUES 句または副問合せで、表の各列の値を指定しなければなりません。

新しい行のフィールドの値は、表内の列の内部位置および VALUES 句または問合せの選択リストの値の順序に基づいて割り当てられます。表内の各列の位置は、データ・ディクショナリを検索して調べることができます。『Oracle8 Server リファレンス・マニュアル』を参照してください。

列リストに指定しなかった列には、表作成時に設定したデフォルト値が代入されます。デフォルトの列値の詳細は、CREATE TABLE（4-303 ページ）を参照してください。列のいずれかに NOT NULL 制約があると、制約違反のエラーが発生して INSERT 文はロールバックされます。

表に対して INSERT 文を実行すると、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。

## パラレル DML

INSERT キーワードの直後にパラレル・ヒントを配置すると、INSERT 操作をパラレル化できます。パラレル DML は、セッションに対しても使用可能にしなければなりません。パラレル DML を有効にする方法については、ALTER SESSION（4-58 ページ）を参照してください。パラレル DML の詳細は、『Oracle8 Server チューニング』および『Oracle8 Parallel Server 概要および管理』、『Oracle8 Server 概要』を参照してください。

## ビューへの挿入

WITH CHECK OPTION を指定してビューを作成した場合には、ビューに定義されている問合せを満たす行しかビューに挿入できません。

単一実表を使用してビューを作成した場合、行をビューに挿入し、RETURNING 句を使用してその行の値を取り出すことができます。

ビューの問合せに次のいずれかの構成要素が定義されている場合は、そのビューには行を挿入できません。

- 集合演算子
- GROUP BY 句

- グループ関数
- DISTINCT 演算子
- フラット化した副問合せ
- ネストした表の列
- CAST 式および MULTISSET 式

## RETURNING 句

RETURNING 句を含む INSERT 文は、挿入された行を取り出し、PL/SQL 変数またはバインド変数に格納します。VALUES 句を持つ INSERT 文で RETURNING 句を使用すると、列の式、ROWID、REF を戻すことができ、出力バインド変数に格納できます。単一実表のビューで RETURNING 句を持つ INSERT 文も使用できます。

PL/SQL では複数行の挿入はできません。1 つの行値だけを取り出して PL/SQL 変数に格納します。RETURNING 句の使用の詳細は、『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

例 1. 次の文は、DEPT 表に行を挿入します。

```
INSERT INTO dept
VALUES (50, 'PRODUCTION', 'SAN FRANCISCO');
```

例 2. 次の文は、EMP 表に 6 つの列からなる行を挿入します。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);
```

例 3. 次の文は、例 2 と同じ効果があります。

```
INSERT INTO (select empno, ename, job, sal, comm, deptno from emp)
VALUES (7890, 'JINKS', 'CLERK', 1.2E3, NULL, 40);
```

例 4. 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');
```

**例 5.** 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** 列がある場合は、**INSERT** 文に **BALANCE** の値が指定されていないため、新たに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。

**例 6.** 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);
```

**例 7.** 次の文は、**LATEST\_DATA** の行を **SALES** 表のパーティション **OCT96** に挿入します。

```
INSERT INTO sales PARTITION (oct96)
SELECT * FROM latest_data;
```

**例 8.** 次の例では、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;
```

**例 9.** 次の例では、バインド配列 **:l** 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 :l;
```

## 関連項目

[DELETE \(4-370 ページ\)](#)

[UPDATE \(4-536 ページ\)](#)

## LOCK TABLE

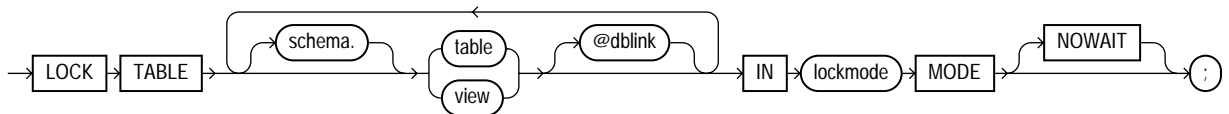
### 用途

指定したモードで 1 つ以上の表をロックします。操作中の表またはビューへの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。「使用上の注意」(4-456 ページ)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

<i>schema</i>	更新する表またはビューが定義されているスキーマです。 <i>schema</i> を指定しないと、この表が自スキーマ内にあるとみなされます。
<i>table / view</i>	ロックする表です。ビューを指定すると、ビューの実表がロックされます。
<i>dblink</i>	表またはビューが格納されている、リモート Oracle データベースに対するデータベース・リンクです。データベース・リンクの指定については、「リモート・データベース内のオブジェクトを参照」(2-50 ページ)を参照してください。Oracle 分散機能を使用している場合は、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なければなりません。  <i>dblink</i> を指定しないと、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。

lockmode	<p>次のいずれかを指定します。</p> <p><b>ROW SHARE</b> は、ロックされた表への同時アクセスを可能にしますが、ユーザーが排他アクセスの目的で表全体をロックすることはできなくなります。<b>ROW SHARE</b> は、<b>SHARE UPDATE</b> と同義で、旧バージョンの Oracle との互換性を保つために用意されています。</p> <p><b>ROW EXCLUSIVE</b> は、<b>ROW SHARE</b> と同じですが、<b>SHARE</b> 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。</p> <p><b>SHARE UPDATE ROW SHARE</b> を参照してください。</p> <p><b>SHARE</b> 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。</p> <p><b>SHARE ROW EXCLUSIVE</b> は、表全体を見る場合に使用します。これを使うと他のユーザーはその表内の行は見ることはできますが、<b>SHARE</b> モードで表をロックしたり、あるいは行を更新することはできません。</p> <p><b>EXCLUSIVE</b> は、ロックされた表上では、問合せは実行できますが、他のアクティビティは実行できません。</p>
NOWAIT	<p>指定した表が他のユーザーによってすでにロックされている場合に、制御をただちに <b>LOCK TABLE</b> 発行元に戻します。この場合、表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。</p> <p>この句を指定しないと、Oracle は表が使用可能になるまで待ち状態になり、表をロックした後で、発行元に制御を戻します。</p>

使用上の注意

ロックによっては、同じ表に同時に設定できるロックや表ごとに1つだけしか設定できないロックがあります。たとえば、複数ユーザーは同じ表に同時に複数の **SHARE** ロックを設定できますが、**EXCLUSIVE** ロックは同じ表に1度に1つしか設定できません。ロック・モードの相互作用についての説明は、『Oracle8 Server 概要』を参照してください。

表をロックする場合、他のユーザーによるアクセスを考慮します。ロックされた表は、トランザクションをコミット、または全体をロールバックするか、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。

ロックしても他のユーザーが表を問い合わ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はありません。

- 例 1. 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;
```
- 例 2. 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;
```

## 関連項目

DELETE (4-370 ページ)

INSERT (4-449 ページ)

UPDATE (4-536 ページ)

COMMIT (4-182 ページ)

ROLLBACK (4-481 ページ)

SAVEPOINT (4-484 ページ)

# NOAUDIT(SQL 文)

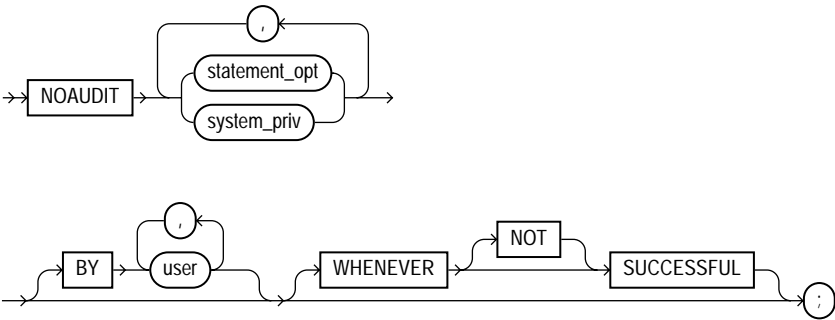
## 用途

AUDIT コマンド (SQL 文) で有効にした監査を停止します。AUDIT コマンド (スキーマ・オブジェクト) で有効にした監査の停止については、NOAUDIT(スキーマ・オブジェクト) (4-460 ページ) を参照してください。「使用上の注意」 (4-459 ページ) も参照してください。

## 前提条件

AUDIT SYSTEM システム権限が必要です。

## 構文



## キーワードとパラメータ

|                        |                                                                                                    |                                                   |
|------------------------|----------------------------------------------------------------------------------------------------|---------------------------------------------------|
| <i>statement_opt</i>   | 監査を停止する文のオプションです。文の各オプションと各オプションで監査する SQL 文については、表 4-6 (4-169 ページ) および表 4-7 (4-171 ページ) を参照してください。 |                                                   |
| <i>system_priv</i>     | 監査を停止するシステム権限です。システム権限と各システム権限によって許可される文については、表 4-6 (4-169 ページ) を参照してください。                         |                                                   |
| BY <i>user</i>         | 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 文の監査だけを取り消します。この句を指定しないと、すべてのユーザーの文の監査を停止します。ただし、この後の項で説明する状況は除きます。  |                                                   |
| WHENEVER<br>SUCCESSFUL | 正常終了した SQL 文の監査だけを停止します。                                                                           |                                                   |
|                        | NOT                                                                                                | エラーとなった文の監査だけを停止します。                              |
|                        |                                                                                                    | WHENEVER 句自体を指定しないと、文が成功しても失敗しても、すべての文の監査が停止されます。 |



## 使用上の注意

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 文だけを無効にします。したがって、1 つの AUDIT 文 (文 A) が特定ユーザーに対して監査を有効にし、2 番目の文 (文 B) がすべてのユーザーに対して監査を使用可能にしている場合、すべてのユーザーに対して監査を使用禁止にする NOAUDIT 文 (文 C) は、文 B を無効にします。しかし、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。特定の SQL コマンドの監査の詳細は、AUDIT(SQL 文) (4-167 ページ) を参照してください。

次の 3 つの例は、AUDIT(SQL 文) (4-167 ページ) の説明で示した最初の 3 つの例と対応しています。

**例 1.** 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;
```

**例 2.** 次の文は、ユーザー SCOTT と 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に、SCOTT の問合せの監査だけを停止します。

```
NOAUDIT SELECT TABLE  
  BY scott;
```

この結果、ユーザー SCOTT の問合せの監査だけが停止されます。BLAKE の問合せと更新、SCOTT の更新の監査は継続されます。

**例 3.** 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;
```

## 関連項目

NOAUDIT(スキーマ・オブジェクト) (4-460 ページ)

## NOAUDIT( スキーマ・オブジェクト )

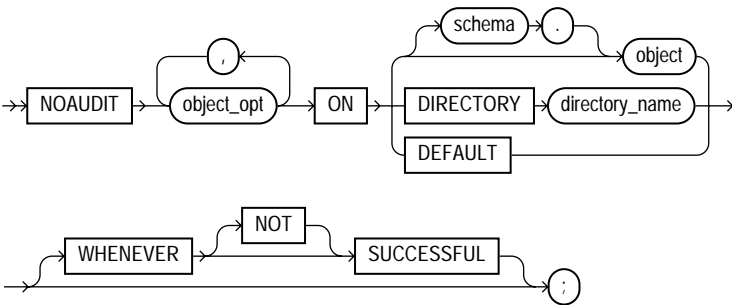
### 用途

AUDIT コマンド (スキーマ・オブジェクト) で有効にした監査を停止します。AUDIT コマンド (SQL 文) で有効にした監査の停止については、NOAUDIT(SQL 文) (4-458 ページ) を参照してください。使用例は、「例」 (4-453 ページ) を参照してください。

### 前提条件

監査を停止するオブジェクトが自スキーマ内にあることが必要です。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は、自分で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



### キーワードとパラメータ

|                   |                                                                                                                                                                                       |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>object_opt</i> | オブジェクト上の特定操作の監査を停止します。これらのオプションのリストは、表 4-8 (4-177 ページ) を参照してください。                                                                                                                     |
| ON                | 監査を停止するオブジェクトを指定します。 <i>object</i> を <i>schema</i> で修飾しないと、自スキーマ内にあるとみなされます。                                                                                                          |
| <i>object</i>     | 監査を停止するオブジェクトを指定します。指定するオブジェクトは、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、スナップショット、ライブラリのいずれかでなければなりません。<br><br>特定のスキーマ・オブジェクトの監査については、AUDIT( スキーマ・オブジェクト ) (4-175 ページ) を参照してください。 |

---

|                                    |                                                                    |
|------------------------------------|--------------------------------------------------------------------|
| DIRECTORY<br><i>directory_name</i> | 監査を停止するディレクトリの名前を指定します。                                            |
| DEFAULT                            | これ以降に作成されるオブジェクトに対するデフォルトのオブジェクト・オプションとなるよう指定したオブジェクト・オプションを取消します。 |
| WHENEVER<br>SUCCESSFUL             | 正常終了した SQL 文の監査だけを停止します。                                           |
| NOT                                | エラーとなった文の監査だけを停止します。                                               |
|                                    | WHENEVER 句自体を指定しないと、文が成功しても失敗しても、すべての文の監査が停止されます。                  |

---

## 例

スキーマ SCOTT 内の EMP 表に問合せを行うすべての SQL 文の監査を選択していた場合に、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;
```

次の文は、正常終了した問合せの監査を停止します。

```
NOAUDIT SELECT
  ON scott.emp
  WHENEVER SUCCESSFUL;
```

この結果、正常終了した問合せの監査だけが停止され、Oracle エラーとなった問合せの監査は継続します。

## 関連項目

AUDIT( スキーマ・オブジェクト ) (4-175 ページ)

NOAUDIT(SQL 文) (4-458 ページ)

# PARALLEL 句

## 用途

Oracle で 1 つの操作をシリアルに実行するかパラレルで実行するかを指定します。「使用上の注意」(4-463 ページ) も参照してください。使用例は、「例」(4-464 ページ) を参照してください。

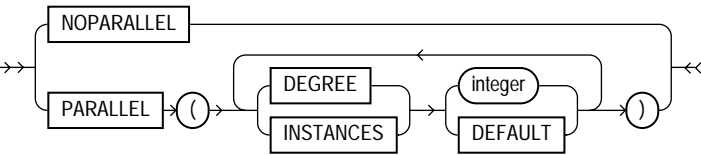
## 前提条件

この句は、次のコマンドでだけ使用できます。

- ALTER CLUSTER
- ALTER DATABASE ... RECOVER
- ALTER INDEX ... REBUILD
- ALTER TABLE
- CREATE CLUSTER
- CREATE INDEX
- CREATE TABLE

注意： PARALLEL 句の構文は、CREATE SCHEMA 文で表または索引、クラスタを作成するときに使用できます。ただしこの場合は、並列性は考慮されず、エラー・メッセージも発行されません。

## 構文



## キーワードとパラメータ

|            |                              |
|------------|------------------------------|
| NOPARALLEL | 操作のシリアル実行を指定します。これはデフォルト値です。 |
| PARALLEL   | 操作のパラレル実行を指定します。             |

---

|                |                                                                                                                                                                         |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DEGREE         | 単一のインスタンスを操作するための並列度を指定します。つまり、並列操作で使用する問合せサーバーの数を指定します。                                                                                                                |
| <i>integer</i> | <i>integer</i> に指定された数の問合せサーバーを使用します。                                                                                                                                   |
| DEFAULT        | 使用する問合せサーバーのデフォルトの数は、CPU の数、および並行走査の対象の表を格納している DEVICE の数から計算されます。                                                                                                      |
| INSTANCES      | 並列操作で使用するパラレル・サーバーのインスタンスの数を指定します。このキーワードは、パラレル・サーバーがない場合は無視されます。                                                                                                       |
| <i>integer</i> | <i>integer</i> で指定した数のインスタンスを使用します。                                                                                                                                     |
| DEFAULT        | 使用可能なすべてのインスタンスを使用します。                                                                                                                                                  |
|                | <p><b>注意：</b>INSTANCES は、Oracle Parallel Server を使用しているインスタンスにだけ適用されます。</p> <p>「非パーティション表および非パーティション索引」（4-463 ページ）および「パーティション表およびパーティション索引」（4-464 ページ）の説明を参照してください。</p> |

---

## 使用上の注意

CREATE TABLE コマンドおよび ALTER TABLE コマンドに表の並列性を指定するには PARALLEL 句を使用します。表定義の中でこの句を指定すると、この句を使用して DML 文および問合せの並列性が判別されます。ただし、明示的なパラレル・ヒントは、その表の PARALLEL 句を無効にします。

PARALLEL 句を指定しないと、使用される並列性の種類は、その表または索引のデフォルトの PARALLEL 属性によって決まります。

パラレル化操作の詳細は、『Oracle8 Server チューニング』および『Oracle8 Server 概要』、『Oracle8 Parallel Server 概要および管理』を参照してください。

## 非パーティション表および非パーティション索引

CREATE コマンドで PARALLEL 句を使用すると、スキーマ・オブジェクトの作成がパラレル化されます。CREATE コマンドが CREATE TABLE であるときは、作成後の表に対する問合せと DML にデフォルトの並列度が設定されます。

オブジェクトを変更するコマンドで PARALLEL 句を使用すると、そのオブジェクトに対する問合せと DML のデフォルトの並列度を変更できます。ALTER DATABASE RECOVER コマンドで、PARALLEL 句を使用すると、回復が並列化されます。

PARALLEL (DEGREE 1 INSTANCES 1) を指定するのは NOPARALLEL を指定するのと同じです。

問合せ内のヒントはデフォルトの NOPARALLEL を無効にします。同様に、問合せ内のヒントはデフォルトの PARALLEL を無効にします。

## パーティション表およびパーティション索引

CREATE TABLE ... AS SELECT および CREATE INDEX の INSTANCES パラメータは、CREATE 操作が使用するインスタンス数を決定します。インスタンスはパーティションの基礎を形成する（最初の）データ・ファイルとの物理的親和性を図るために選択されます。INSTANCES パラメータが、基礎となるデータ・ファイルと親和性のあるインスタンス数より大きい場合は、追加インスタンス（最大でパーティションの合計数）が任意に選択されます。データ・ディクショナリ内に格納されている DEGREE パラメータおよび INSTANCES パラメータが、後からスキーマ・オブジェクトのデフォルト PARALLEL 属性の計算に使用されます。

### 例

**例 1.** 次のコマンドは、10 の問合せサーバーを使用して表を作成する例です。10 のうち 5 つは SCOTT.EMP の走査用であり、残りの 5 つは EMP\_DEPT に値を入れるためです。

```
CREATE TABLE emp_dept
PARALLEL (DEGREE 5)
AS SELECT * FROM scott.emp
WHERE deptno = 10;
```

**例 2.** 次のコマンドは、10 の問合せサーバーを使用して索引を作成する例です。10 のうち 5 つは SCOTT.EMP の走査用であり、残りの 5 つは索引 EMP\_IDX に値を入れるためです。

```
CREATE INDEX emp_idx
ON scott.emp (ename)
PARALLEL 5;
```

**例 3.** 次のコマンドは、パラレル・サーバーにおける 5 つのインスタンスに対する 5 つの回復プロセスを使用して表領域の回復を行います。問合せサーバーの合計数は 25(5\*5) です。

```
ALTER DATABASE
RECOVER TABLESPACE binky
PARALLEL (DEGREE 5 INSTANCES 5);
```

**例 4.** 次のコマンドは、EMP 表を問い合わせるために使用する問合せサーバーのデフォルトの数を変更します。

```
ALTER TABLE emp
PARALLEL (DEGREE 9);
```

**例 5.** 次のコマンドは、6 つの問合せサーバーを使用して、既存の索引から索引を再構築します。6 つのサーバーのうち 3 つは既存の索引の走査用であり、残りは新しい索引の構築用です。

```
ALTER INDEX emp_idx
REBUILD
PARALLEL 3;
```

## 関連項目

ALTER CLUSTER (4-11 ページ)

ALTER DATABASE (4-15 ページ)

ALTER INDEX (4-28 ページ)

ALTER TABLE (4-105 ページ)

CREATE CLUSTER (4-202 ページ)

CREATE INDEX (4-233 ページ)

CREATE TABLE (4-303 ページ)

『Oracle8 Server チューニング』

『Oracle8 Parallel Server 概要および管理』

## RECOVER 句

---

### 用途

メディア回復を実行します。

SQL を使って独自のメディア回復アプリケーションを記述する場合は、**RECOVER** 句を指定した **ALTER DATABASE** コマンドを使用できます。これ以外の場合はメディア回復には、**RECOVER** 句を指定した **ALTER DATABASE** コマンドではなく、**Server Manager** の **RECOVER** コマンドを使ってください。

メディア回復の詳細は、『**Oracle8 Server** バックアップおよびリカバリ』および『**Oracle8 Server** 管理者ガイド』を参照してください。使用例は、「例」（4-468 ページ）を参照してください。

### 前提条件

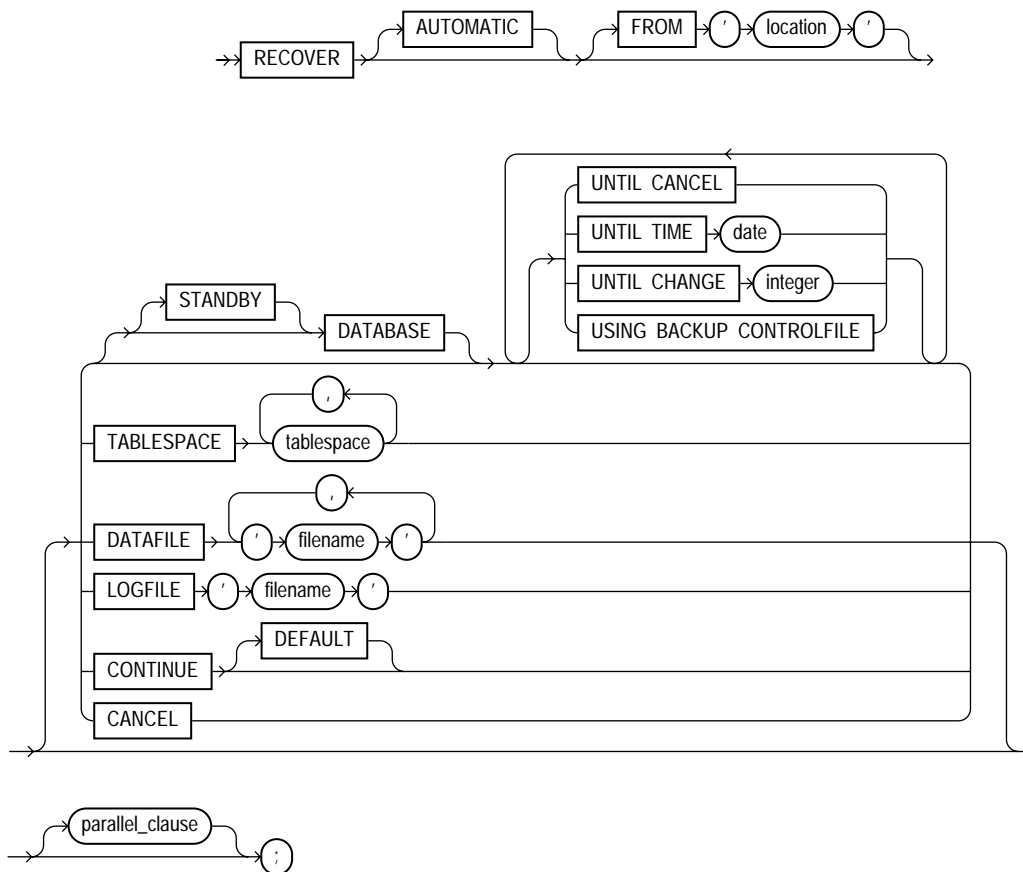
**RECOVER** 句は **ALTER DATABASE** 文に指定する必要があります。したがって、この文の実行権限が必要です。これらの権限の詳細は、**ALTER DATABASE**（4-15 ページ）を参照してください。

また、次の条件もあります。

- 有効な **OSDBA** ロールを持っている必要がある。
- マルチスレッド・サーバー・アーキテクチャを介しては **Oracle** に接続できない。
- インスタンスは排他モードでデータベースをマウントしなければならない。



## 構文



parallel\_clause: PARALLEL 句 (4-462 ページ) を参照。

## キーワードとパラメータ

**AUTOMATIC** メディア回復に適用される REDO ログ・ファイルの名前が自動的に生成されます。このオプションを指定しない場合は、ALTER DATABASE ...RECOVER コマンドの LOGFILE 句を使って REDO ログ・ファイル名を指定する必要があります。

|                          |                                                                                                                                                                                                     |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FROM                     | アーカイブ REDO ログ・ファイル・グループを読み取る位置を指定します。 <i>location</i> の値には、使用するオペレーティング・システムの表記規則に従って、ファイルの位置を完全に修飾して指定しなければなりません。このパラメータを指定しないと、そのアーカイブ REDO ログ・グループは初期化パラメータ LOG_ARCHIVE_DEST に指定された場合にあるとみなされます。 |
| STANDBY                  | ブラリマリ・データベースからコピーされたアーカイブ REDO ログ・ファイルと制御ファイルを使用してスタンバイ・データベースを回復します。詳細は、『Oracle8 Server 管理者ガイド』を参照してください。                                                                                          |
| DATABASE                 | データベース全体を回復します。これはデフォルトのオプションです。データベースがクローズされている場合にだけ、このオプションを使用できます。<br>注意：このオプションはオンライン・データ・ファイルだけを回復します。                                                                                         |
| UNTIL CANCEL             | 取消しベースの回復を実行します。CANCEL 句を指定した ALTER DATABASE RECOVER コマンドを発行するまで、回復が続行されます。                                                                                                                         |
| UNTIL TIME               | 時間ベースの回復を実行します。このパラメータは、日付に指定した時点までデータベースを回復します。日付は 'YYYY-MM-DD:HH24:MI:SS' の書式の文字列リテラルでなければなりません。                                                                                                  |
| UNTIL CHANGE             | 変更ベースの回復を実行します。 <i>integer</i> に指定したシステム変更番号 (SCN) の直前までデータベースのトランザクションの一貫性を回復します。                                                                                                                  |
| USING BACKUP CONTROLFILE | 現行の制御ファイルではなく、バックアップ制御ファイルを使うように指定します。                                                                                                                                                              |
| TABLESPACE               | 指定した表領域だけ回復します。回復の対象となる表領域がオフラインの場合、データベースがオープン状態でもクローズ状態でも、このオプションを使用できます。                                                                                                                         |
| DATAFILE                 | 指定したデータ・ファイルを回復します。回復の対象となるデータ・ファイルがオフラインの場合、データベースがオープン状態でもクローズ状態でも、このオプションを使用できます。                                                                                                                |
| LOGFILE                  | 指定した REDO ログ・ファイルを使って、メディア回復を続行します。                                                                                                                                                                 |
| CONTINUE                 | スレッドを使用禁止にするために中断されていたマルチ・インスタンス回復を再開します。                                                                                                                                                           |
| CONTINUE DEFAULT         | Oracle が自動生成した REDO ログ・ファイルを使って、回復を続行します。                                                                                                                                                           |
| CANCEL                   | 取消しベースの回復を終了します。                                                                                                                                                                                    |
| <i>parallel_clause</i>   | 回復時の並列度を指定します。PARALLEL 句（4-462 ページ）を参照してください。                                                                                                                                                       |

例

例 1. 次の文は、データベース全体の完全な回復を実行します。

```
ALTER DATABASE
  RECOVER AUTOMATIC DATABASE;
```

Oracle は、適用する REDO ログ・ファイルの名前を自動的に生成し、その名前をプロンプトに表示します。次の文ではこの提示されたファイルに回復を適用します。

```
ALTER DATABASE  
    RECOVER CONTINUE DEFAULT;
```

次の文は、Oracle が適用する REDO ログ・ファイルの名前を明示的に指定しています。

```
ALTER DATABASE  
    RECOVER LOGFILE 'diska:arch0006.arc';
```

例 2. 次の文は、時間ベースのデータベース回復を実行します。

```
ALTER DATABASE AUTOMATIC  
    RECOVER UNTIL TIME '1992-10-27:14:00:00';
```

データベースは、1992 年 10 月 27 日午後 2 時の状態に回復されます。

例 3. 次の文は、表領域 USER5 を回復します。

```
ALTER DATABASE  
    RECOVER TABLESPACE user5;
```

## 関連項目

ALTER DATABASE (4-15 ページ)

---

# RENAME

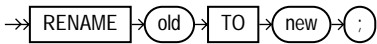
## 用途

表またはビュー、順序、( 表またはビュー、順序の ) プライベート・シノニムを改名します。「オブジェクトの改名」( 4-470 ページ ) 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ことが必要です。「使用上の注意」( 4-470 ページ ) も参照してください。

## 構文



## キーワードとパラメータ

---

|            |                                                                                                                                                 |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>old</i> | 既存の表またはビュー、順序、プライベート・シノニムの名前です。                                                                                                                 |
| <i>new</i> | 既存のオブジェクトに指定する新しい名前です。新しい名前は、同じ名前領域内の他のスキーマ・オブジェクトに使用されている名前であってはなりません。また、「スキーマ・オブジェクトの命名規則」( 2-43 ページ ) の項に定義されているスキーマ・オブジェクト命名規則に従わなければなりません。 |

---

## オブジェクトの改名

古いオブジェクトの整合性制約および索引、権限は、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およびシノニム、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。

表の名前を DEPT から EMP\_DEPT に変更するには、次の文を発行します。

```
RENAME dept TO emp_dept;
```

## 使用上の注意

このコマンドではパブリック・シノニムを改名できません。パブリック・シノニムを改名するには、DROP SYNONYM コマンドを使ってそのシノニムを削除してから、CREATE SYNONYM コマンドを使って新しい名前のパブリック・シノニムを作成します。

このコマンドでは列を改名できません。列を改名するには、AS 句を指定した CREATE TABLE コマンドの使用します。たとえば、次の文は、列 OLDNAME を NEWNAME に変更して、表 STATIC を再作成します。

```
CREATE TABLE temporary (newname, col2, col3)
      AS SELECT oldname, col2, col3 FROM static
DROP TABLE static
RENAME temporary TO static;
```

## 関連項目

CREATE SEQUENCE (4-278 ページ)

CREATE SYNONYM (4-299 ページ)

CREATE TABLE (4-303 ページ)

CREATE VIEW (4-359 ページ)

## REVOKE( システム権限とロール )

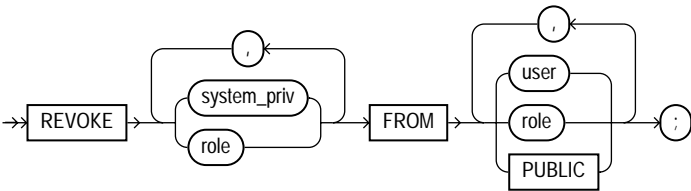
### 用途

ユーザーまたはロールのシステム権限やロールを取り消します。ユーザーおよびロールからオブジェクト権限を取り消すには、**REVOKE**( スキーマ・オブジェクト権限 ) (4-475 ページ) を参照してください。使用例は、「例」(4-473 ページ) を参照してください。

### 前提条件

システム権限または **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と、ロールを自由に取り消せます。「使用上の注意」(4-473 ページ) も参照してください。

### 構文



### キーワードとパラメータ

|                    |                                                                                                      |
|--------------------|------------------------------------------------------------------------------------------------------|
| <i>system_priv</i> | 取り消すシステム権限です。システム権限のリストは、表 4-19 (4-434 ページ) を参照してください。「権限の取消し」(4-472 ページ) も参照してください。                 |
| <i>role</i>        | 取り消すロールです。Oracle により事前定義済みのロールのリストは、『Oracle8 Server 管理者ガイド』を参照してください。「ロールの取消し」(4-473 ページ) も参照してください。 |
| FROM               | システム権限を取り消すユーザーまたはロールを指定します。                                                                         |
| PUBLIC             | すべてのユーザーのシステム権限またはロールを取り消します。                                                                        |

### 権限の取消し

**ユーザーの権限を取り消す場合** : Oracle は、ユーザーの権限ドメインの権限を取り消します。この取消しはただちに有効となるため、ユーザーはその権限を行使できなくなります。

**ロールの権限を取り消す場合** : ロールの権限ドメインの権限を取り消します。この取消しはただちに有効となるため、そのロールが使用可能となっても、この権限は行使できません。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、その権限を行使できなくなります。

**PUBLIC の権限を取り消す場合 :** PUBLIC を介して権限を付与されている各ユーザーの権限ドメインの権限を取り消します。これはただちに有効となるため、それらのユーザーは権限を行使できなくなります。ただし、直接またはロールを介してその権限を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。

## ロールの取消し

**ユーザーのロールを取り消す場合 :** 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であればその権限を行使できます。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。

**他のロールのロールを取り消す場合 :** Oracle は被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き続き行使できます。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行使できません。

**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、そのロールの使用禁止にします。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かぎり、権限ドメインでその権限を引き続き行使できます。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。直接または他のロールを介してロールを付与されているユーザーからは、ロールが取り消されません。

## 使用上の注意

REVOKE コマンドによって取り消すことができる権限やロールは、GRANT 文によって直接付与されているものに限られます。REVOKE コマンドでは、次の処理は実行できません。

- 被取消し側に付与されていない権限またはロールの取消し
- 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の取消し
- ロールを介して被取消し側に付与されている権限またはロールの取消し

システム権限またはロールは、取り消す権限またはロールのリストに 1 度しか指定できません。ユーザー、ロール、PUBLIC は、FROM 句に 1 度しか指定できません。

## 例

**例 1.** 次の文は、ユーザー BILL と 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 は別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。

**例 2.** 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。

**例 3.** 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。

**例 4.** 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。

**例 5.** ユーザー SCOTT の CREATE ANY DIRECTORY システム権限を取り消すには、次の文を発行します。

```
REVOKE CREATE ANY DIRECTORY FROM scott;
```

## 関連項目

GRANT( システム権限とロール ) (4-432 ページ)

REVOKE( スキーマ・オブジェクト権限 ) (4-475 ページ)



## REVOKE( スキーマ・オブジェクト権限 )

### 用途

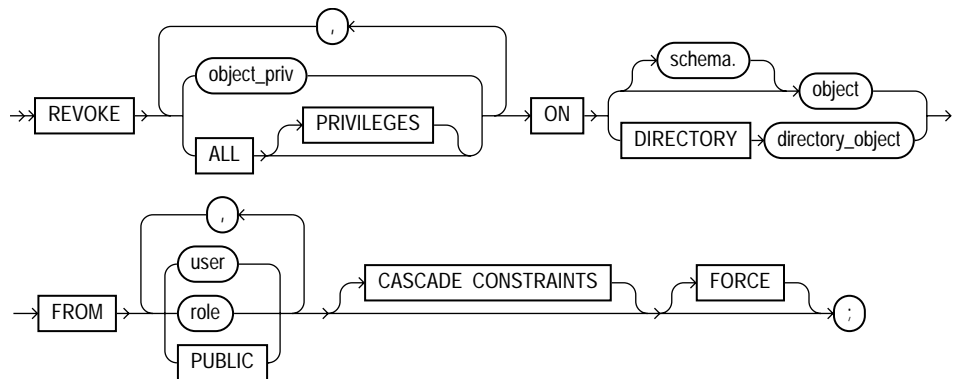
ユーザーまたはロールの特定のオブジェクト権限を取り消します。システム権限またはロールを取り消すには、REVOKE(システム権限とロール) (4-472 ページ) を参照してください。「使用上の注意」 (4-477 ページ) も参照してください。

それぞれのオブジェクト権限によって、オブジェクトに対するいくつかの操作を実行できます。オブジェクト権限を取り消すことによって、取り消されたユーザーまたはロールはその操作を禁止されます。それぞれの型のオブジェクトのオブジェクト権限の要約は、表 4-13 (4-445 ページ) を参照してください。使用例は、「例」 (4-478 ページ) を参照してください。

### 前提条件

それぞれのユーザーとロールに、オブジェクト権限が事前に付与されていることが前提です。「複数の同じ権限の取消し」 (4-477 ページ) も参照してください。

### 構文



## キーワードとパラメータ

---

|                                         |                                                                                                                                                                                                                   |
|-----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>object_priv</i>                      | 取り消すオブジェクト権限です。次のいずれかの値を指定します。<br><br>ALTER<br><br>DELETE<br><br>EXECUTE<br><br>INDEX<br><br>INSERT<br><br>READ<br><br>REFERENCES<br><br>SELECT<br><br>UPDATE                                                     |
| ALL PRIVILEGES                          | ユーザーまたはロールに付与されているオブジェクト権限をすべて取り消します。<br><br>注意： オブジェクトに権限がまったく付与されていない場合は、処理は行われず、エラー・メッセージも戻りません。                                                                                                               |
| ON DIRECTORY<br><i>directory_object</i> | 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 オプションを使用しているときは、 <i>directory_object</i> を <i>schema</i> で修飾できません。このオブジェクトはディレクトリでなければなりません。CREATE DIRECTORY（4-226 ページ）を参照してください。                                            |
| ON <i>object</i>                        | 権限を取り消すオブジェクトを指定します。オブジェクトとして指定できるのは、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、スナップショット、ライブラリのいずれか、または表、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、スナップショットのシノニムです。<br><br><i>object</i> を <i>schema</i> で修飾しないと、自スキーマ内にあるとみなされます。 |
| FROM                                    | オブジェクト権限を取り消すユーザーまたはロールを指定します。<br><br>PUBLIC      全ユーザーのオブジェクト権限を取り消します。                                                                                                                                          |
| CASCADE<br>CONSTRAINTS                  | 被取消し側で REFERENCES 権限または ALL PRIVILEGES オプションを使って定義した参照整合性制約を削除します。このオプションは、被取消し側で REFERENCES 権限により参照整合性制約が定義されている場合に有効です。「取消しのカスケード」（4-477 ページ）も参照してください。                                                         |
| FORCE                                   | 表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 オプションを使用して EXECUTE オブジェクト権限を取り消します。「FORCE の使用」（4-477 ページ）を参照してください。型依存性とユーザー定義オブジェクト権限の詳細は、『Oracle8 Server 概要』を参照してください。                 |

---

## 使用上の注意

**ユーザーの権限を取り消す場合：**Oracle は、ユーザーの権限ドメインの権限を取り消します。この取消しはただちに有効となるため、ユーザーはその権限を行使できなくなります。

**ロールの権限を取り消す場合：**ロールの権限ドメインの権限を取り消します。この取消しはただちに有効となるため、そのロールが使用可能となっても、この権限は行使できません。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ても、権限を行使できなくなります。

**PUBLIC の権限を取り消す場合：**PUBLIC を介して権限を付与されている各ユーザーの権限ドメインの権限を取り消します。この取消しはただちに有効となるため、それらすべてのユーザーはその権限を行使できなくなります。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。

REVOKE コマンドでは、被取消し側に直接付与しておいたオブジェクト権限だけを取り消します。REVOKE コマンドでは次の処理は実行できません。

- 被取消し側に付与していないオブジェクト権限の取消し
- オペレーティング・システムを介して付与されている権限の取消し
- 被取消し側に付与されているロールからの権限の取消し

権限は、取り消す権限のリストに一度しか指定できません。ユーザー、ロール、PUBLIC は、FROM 句に 1 度しか指定できません。

## FORCE の使用

表に依存するユーザー定義型オブジェクトでは、FORCE オプションを使用して EXECUTE オブジェクト権限を取り消します。FORCE オプションを指定すると、依存表のデータにアクセスできなくなります。必要な型の権限を再付与すると、表に対して再び妥当性検査が行われます。型依存性とユーザー定義オブジェクト権限の詳細は、『Oracle8 Server 概要』を参照してください。

## 複数の同じ権限の取消し

複数のユーザーが、同じオブジェクト権限を同一ユーザーおよびロール、PUBLIC に対して付与できます。権限受領者の権限定ドメインの権限を取り消すには、すべての権限付与者が権限を取り消す必要があります。権限を取り消さない権限付与者が 1 人でもいれば、権限受領者は引き続きその権限を行使できます。

## 取消しのカスケード

ユーザーが付与されているオブジェクト権限、またはオブジェクト参照整合性制約を定義するオブジェクト権限を取り消すと、次の影響があります。

- 他のユーザーまたはロールにオブジェクト権限を付与しているユーザーのオブジェクト権限を取り消すと、そのユーザーから権限を付与されているユーザーまたはロールの権限も自動的に取り消されます。

- 権限を行使する SQL 文を記述したプロシージャおよび関数、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および関数、パッケージは実行できなくなります。
- あるオブジェクトのオブジェクト権限を、そのオブジェクトのビューが設定されているスキーマを持つユーザーから取り消すと、Oracle はそのビューを無効にします。
- 参照整合性制約の定義権限を付与されているユーザーの REFERENCES 権限を取り消した場合、CASCADECONSTRAINTS オプションも指定しなければなりません。この指定により、権限が取り消され、制約が削除されます。

### 例

**例 1.** 次の文は、表 BONUS に対する DELETE、INSERT、SELECT、UPDATE の各権限をユーザー PEDRO に付与します。

```
GRANT ALL
  ON bonus TO pedro;
```

次の文は、ユーザー PEDRO の表 BONUS に対する DELETE 権限を取り消します。

```
REVOKE DELETE
  ON bonus FROM pedro;
```

次の文は、ユーザー PEDRO の表 BONUS に対する残りの権限をすべて取り消します。

```
REVOKE ALL
  ON bonus FROM pedro;
```

**例 2.** 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対してビュー REPORTS に対する SELECT 権限と UPDATE 権限を付与する例を次に示します。

```
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 ビューの問合せはできますが、その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。

**例 3.** 次の文は、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 権限を取り消すには、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 の権限付与により ESEQ を使用できます。

**例 4.** 次の文では、BLAKE にスキーマ SCOTT 内の EMP 表に対する REFERENCES 権限および UPDATE 権限を付与できます。

```
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、自分の 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 オプションを指定した次の文を発行することによって、BLAKE の SCOTT.EMP に対する REFERENCES 権限を取り消すことができます。

```
REVOKE REFERENCES
ON scott.emp
FROM blake
CASCADE CONSTRAINTS;
```

BLAKE の SCOTT.EMP に対する REFERENCES 権限を取り消すと、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 は制約のために必要な権限を保持しています。

**例 5.** 次の文を発行することにより、SUE のディレクトリ BFILE\_DIR1 に対する READ 権限を取り消すことができます。

```
REVOKE READ ON DIRECTORY bfile_dir1 FROM sue;
```

## 関連項目

[GRANT\( オブジェクト権限 \) \(4-442 ページ\)](#)

[REVOKE\( システム権限とロール \) \(4-472 ページ\)](#)

# ROLLBACK

## 用途

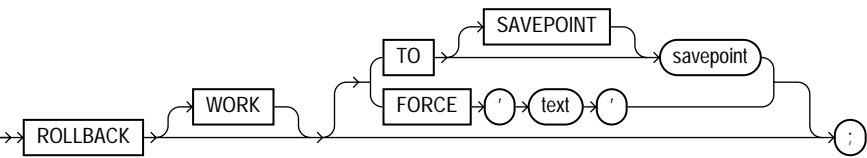
現行のトランザクションで実行された処理の取消しまたはインダウト分散トランザクションで実行された処理の手動での取消しを実行します。「使用上の注意」(4-481 ページ) も参照してください。

## 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は、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には、**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には、**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                                           | 指定は任意です。ANSI 互換性のために提供されています。                                                                                                                                                                         |
| TO                                             | 現行トランザクションを指定したセーブポイントにロールバックします。この句を指定しないと、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。                                                                                                                      |
| 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します。このトランザクションはそのローカル・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 を確認するには、データ・ディクショナリ・ビュー DBA_2PC_PENDING を問い合わせます。「分散トランザクション」(4-482 ページ) も参照してください。 |
| 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。 |                                                                                                                                                                                                       |

## 使用上の注意

トランザクション（論理的作業単位）とは、Oracle が 1 つの単位として扱う一連の SQL 文です。トランザクションは、COMMIT 文または ROLLBACK 文の後、またはデータベースに接

続した後の最初の実行可能 SQL 文によって開始され、COMMIT 文、ROLLBACK 文、またはデータベースとの接続の切離し（意図されたかどうかにかかわらず）によって終了します。

---

---

**注意：** データ定義言語 (DDL) 文の処理の前後には、暗黙的な COMMIT が発行されます。

---

---

TO SAVEPOINT 句を指定しない ROLLBACK コマンドによって、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

TO SAVEPOINT 句を指定した ROLLBACK コマンドによって、次の処理が行われます。

- 指定したセーブポイントの後のトランザクションにロールバックする。
- 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する。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する。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たなければなりません。行を要求していない他のトランザクションは、ただちに行の要求とアクセスできます。

アプリケーション・プログラムでは、COMMIT または ROLLBACK 文を使ってトランザクションを明示的に終了することをお勧めします。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すると、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。

**例 1.** 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;
```

**例 2.** 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;
```

## 分散トランザクション

Oracle の分散機能では分散トランザクション（すなわち複数のデータベース上のデータを変更するトランザクション）を実行できます。分散トランザクションをコミットまたはロール



バックするには、他のトランザクションと同様に、COMMIT 文または ROLLBACK 文を発行します。

分散トランザクションのコミット・プロセス中にネットワーク障害が発生すると、トランザクションの状態が不明、つまりインダウトになる場合があります。この場合、そのトランザクションにかかわる他のデータベースの管理者と相談し、自分のローカル・データベース上のトランザクションを手動でロールバックするか、コミットするかを判断してください。自分のローカル・データベースでトランザクションを手動でロールバックするには、ROLLBACK 文に FORCE 句を指定します。

インダウト・トランザクションをいつロールバックするかについては、『Oracle8 Server 分散システム』を参照してください。

インダウト・トランザクションは、セーブポイントに手動でロールバックできません。

FORCE 句を指定した ROLLBACK 文では、指定したトランザクションしかロールバックできないので注意してください。この文は現行トランザクションには影響しません。

例 . 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';
```

## 関連項目

COMMIT (4-182 ページ)

SAVEPOINT (4-484 ページ)

SET TRANSACTION (4-514 ページ)

# SAVEPOINT

## 用途

トランザクション内でロールバックされる位置を指定します。「使用上の注意」（4-484 ページ）も参照してください。

## 前提条件

なし。

## 構文



## キーワードとパラメータ

|                  |                   |
|------------------|-------------------|
| <i>savepoint</i> | 作成するセーブポイントの名前です。 |
|------------------|-------------------|

## 使用上の注意

セーブポイントは、ROLLBACK コマンドに指定します。これによって、現行トランザクションの一部をロールバックできます。詳細は、「使用上の注意」（4-481 ページ）を参照してください。

セーブポイントでは、プロシージャの中間ステップを作成し指定できるため、対話形式のプログラムに便利です。セーブポイントによって、より大規模で複雑なプログラムを制御できます。たとえば、一連の大規模な更新処理にセーブポイントを指定しておけば、エラーが発生した場合にすべての文を再実行する必要はありません。

セーブポイントは、アプリケーション・プログラムでも有効です。プログラムに複数のサブプログラムが記述されている場合は、各サブプログラムの開始点の前にセーブポイントを設定できます。このため、サブプログラムがエラーとなっても、データを容易にサブプログラム開始前の元の状態に戻すことができ、パラメータを訂正してサブプログラムを再実行したり、回復処理を実行したりできます。

同一トランザクション内のセーブポイント名は区別しなければなりません。同じ識別子のセーブポイントを指定すると、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、処理の継続または作業のコミットや、トランザクション全体のロールバック、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 \$20,000 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 = 1300
  WHERE ename = 'CLARK'
COMMIT;
```

## 関連項目

[COMMIT \(4-182 ページ\)](#)

[ROLLBACK \(4-481 ページ\)](#)

[SET TRANSACTION \(4-514 ページ\)](#)

---

## SELECT

### 用途

1 つ以上の表またはオブジェクト表、ビュー、オブジェクト・ビュー、スナップショットからデータを取り出します。

---

---

**注意：** コマンドと句の説明で**OBJ**の印が前に付いている箇所は、Oracle Object Option がデータベース・サーバーにインストールされている場合にだけ読んでください。

---

---

### 前提条件

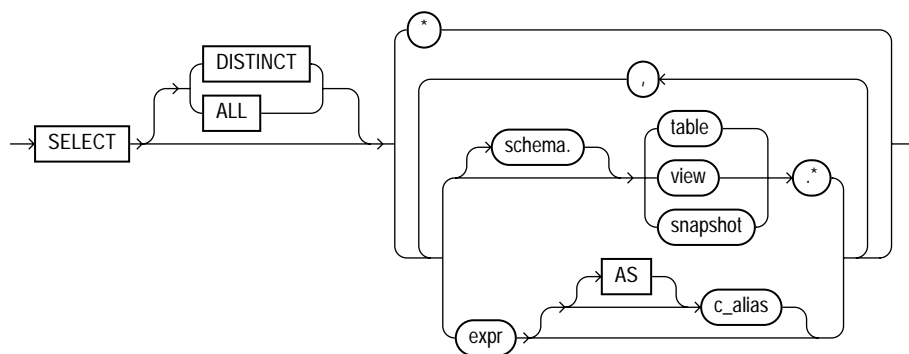
表またはスナップショットからデータを選択するには、表またはスナップショットが自スキーマ内にあるか、その表またはスナップショットの **SELECT** 権限をもっている必要があります。

ビューの実表から行を選択するには、つぎの条件を2つとも満たしていなければなりません。

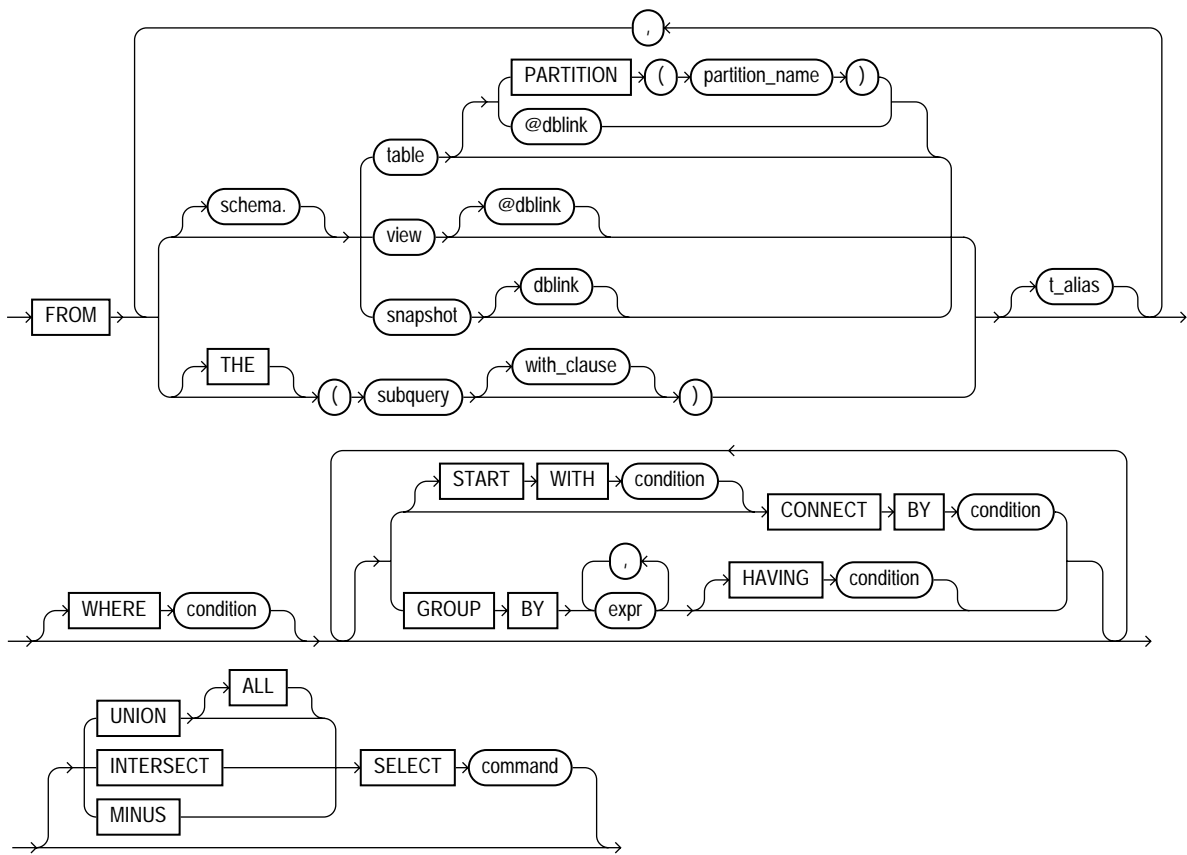
- そのビューに対する **SELECT** 権限を持っている。
- そのビューを含むスキーマの所有者が、実表の **SELECT** 権限を持っている。

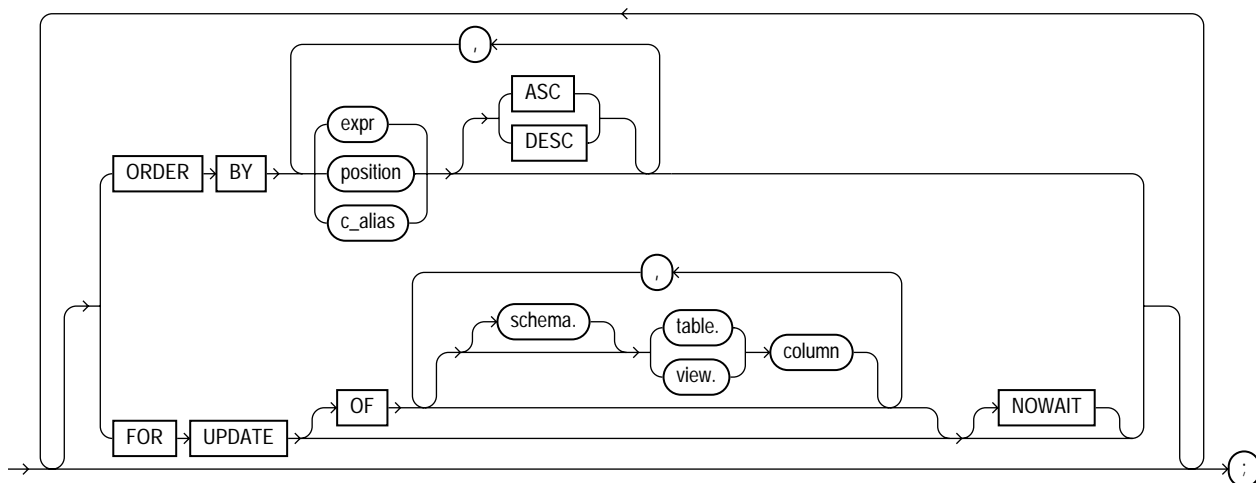
**SELECT ANY TABLE** システム権限を持っていると、任意の表またはスナップショット、任意のビューの実表からデータを選択できます。

## 構文

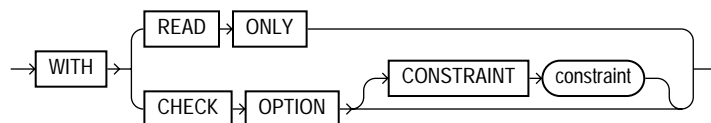


SELECT





WITH\_clause::=



## キーワードとパラメータ

|                                                      |                                                                                                                                                                                                 |
|------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>DISTINCT</b>                                      | 選択された行に重複行がある場合、そのうちの 1 行だけを戻します。重複行とは、選択リスト中の各式で一致する値を持った行のことです。                                                                                                                               |
| <b>ALL</b>                                           | 重複行を含め、選択された行をすべて戻します。デフォルトは <b>ALL</b> です。                                                                                                                                                     |
| <b>*</b>                                             | <b>FROM</b> 句に指定されているすべての表およびビュー、スナップショットの全列を選択します。                                                                                                                                             |
| <i>table.*</i><br><i>view.*</i><br><i>snapshot.*</i> | 表およびビュー、スナップショットの全列を選択します。自分以外のスキーマ内の表、ビュー、スナップショット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」(4-501 ページ) を参照してください。                                                                                          |
| <i>expr</i>                                          | 式を選択します。 <i>expr</i> の構文の説明は、「式」(3-73 ページ) を参照してください。「単純な問合せの作成」(4-491 ページ) も参照してください。この列を含む表およびビュー、スナップショットが <b>FROM</b> 句で <i>schema</i> によって修飾されている場合、このリストに指定する列名は <i>schema</i> だけで修飾できます。 |
| <i>c_alias</i>                                       | 列見出しとして表示される、列の式の別名を指定します。 <b>AS</b> キーワードはオプションです。別名によって、問合せ中に選択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 <b>ORDER BY</b> 句で使用できますが、他の句では使用できません。                                                          |

SELECT

|                                               |                                                                                                                                                                                                                                                                                                                                 |
|-----------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>PARTITION</b><br>( <i>partition_name</i> ) | パーティション・レベルのデータ検索を指定します。 <i>partition_name</i> パラメータには、データ検索対象の表の中のパーティションの名前を指定するか、検索を表の 1 つのパーティションだけに限定する、より複雑な述語を指定することもできます。                                                                                                                                                                                               |
| <i>schema</i>                                 | 選択した表またはビュー、スナップショットが含まれるスキーマです。 <i>schema</i> を指定しないと、この表またはビュー、スナップショットは自スキーマに存在するものとみなされます。                                                                                                                                                                                                                                  |
| <i>table, view, snapshot</i>                  | データを選択する表またはビュー、スナップショットの名前です。                                                                                                                                                                                                                                                                                                  |
| <i>dblink</i>                                 | <p>表またはビュー、スナップショット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です。データベース・リンクの参照方法の詳細は、「リモート・データベース内のオブジェクトを参照」(2-50 ページ) を参照してください。なお、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。</p> <p><i>dblink</i> を指定しないと、その表またはビュー、スナップショットはローカル・データベースに存在するものとみなされます。</p> <p>キーワード <b>THE</b> を適用すると、副問合せは 1 つの列値を戻します。これは、ネストした表、またはネストした表を作る式です。</p> |
| <b>OBJ</b> <b>THE</b>                         | <p>副問合せが戻す列値がスカラー値ではなく、ネストした表であることを通知します。接頭辞 <b>THE</b> の付いた副問合せはフラット化した副問合せと呼ばれます。「フラット化した副問合せの使用」(4-528 ページ) を参照してください。</p> <p><b>注意：</b>フラット化した副問合せでは集合演算子を使用できません。次の <b>set operators</b> を参照してください。</p>                                                                                                                    |
| <i>subquery</i>                               | ビューと同様に扱われる副問合せです。「副問合せ」(4-525 ページ) を参照してください。Oracle は副問合せを実行し、その結果の行を <b>FROM</b> 句内でビューとして使用します。                                                                                                                                                                                                                              |
| <i>t_alias</i>                                | 問合せを評価するための表またはビュー、スナップショット、副問合せの別名を指定します。通常は、相関問合せで使用します。表またはビュー、スナップショットを参照する問合せでは、この別名を参照しなければなりません。                                                                                                                                                                                                                         |
| <b>WHERE</b>                                  | <i>condition</i> に指定した条件が <b>TRUE</b> である行だけを選択します。この句を省略すると、 <b>FROM</b> 句に指定されている表またはビュー、スナップショットのすべての行が戻されます。 <i>condition</i> の構文の説明は、「条件」(3-84 ページ) を参照してください。                                                                                                                                                             |
| <b>START WITH ...</b><br><b>CONNECT BY</b>    | 階層順に行を戻します。「階層問合せ」(4-492 ページ) を参照してください。                                                                                                                                                                                                                                                                                        |
| <b>GROUP BY</b>                               | 選択された行を各行の <i>expr</i> の値に基づいてグループ化し、各グループのサマリー情報を 1 行戻します。「 <b>GROUP BY</b> 句」(4-496 ページ) を参照してください。                                                                                                                                                                                                                           |
| <b>HAVING</b>                                 | <p>指定した条件が <b>TRUE</b> である行のグループだけを戻します。この句を省略すると、すべてのグループの要約行が戻されます。「<b>HAVING</b> 句」(4-497 ページ) を参照してください。</p> <p><i>expr</i> の構文の説明は、「式」(3-73 ページ) を、<i>condition</i> の構文の説明は、「条件」(3-84 ページ) を参照してください。</p>                                                                                                                  |



---

|                                       |                                                                                                                                                                                                                                                                    |
|---------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>set operators:</i>                 | 2つの SELECT 文により戻された行を集合演算子を使って結合します。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 句をこれらの集合演算子と併せて指定することはできません。                                                                                                                                                       |
| UNION UNION<br>ALL INTERSECT<br>MINUS | <b>●</b> THE キーワードまたは MULTISET キーワードを使用した SELECT 文は、これらの集合演算子といっしょに使用することはできません。「UNION および UNION ALL、INTERSECT、MINUS」(4-498 ページ) を参照してください。                                                                                                                        |
| ORDER BY                              | 戻される行を順序付けます。<br><br><i>expr</i> は、 <i>expr</i> の値を基準にして行に順番を付けます。式は、選択リスト、または FROM 句の表およびビュー、スナップショットの列に基づきます。<br><br><i>position</i> は、選択リストのこの位置にある式の値に基づいて行に順番を付けます。<br><br>ASC と DESC は、昇順または降順を指定します。ASC がデフォルトです。<br><br>「ORDER BY 句」(4-498 ページ) を参照してください。 |
| FOR UPDATE                            | 選択された行をロックします。<br><br>OF                      結合内の特定の表の選択された行だけをロックします。<br><br>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ようとした場合、制御が戻されます。この句を省略すると、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。<br><br>「FOR UPDATE 句」(4-499 ページ) を参照してください。                  |

---

## 単純な問合せの作成

SELECT キーワードと FROM 句の間にある式のリストを選択リストと呼びます。各 *expr* は戻される行のセット内の 1 つの列の名前となり、各 *table.\** は列のセットとなります。表内の各列に対して 1 つの *expr* が存在し、その順序は表作成時に定義された列の順序と同じです。各式のデータ型およびデータ長は、その式の要素によって決まります。

複数の表に同じ名前の列がある場合、表の名前でその列名を修飾しなければなりません。列名の重複がない場合、列の名前は完全に修飾する必要はありませんが、表や列の参照には明示的な修飾名を使うことをお勧めします。Oracle では、完全修飾の表名や列名の方が、処理の負荷が少くなります。

FROM 句にキーワード PARTITION を指定することにより、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 NOV96 パーティションへの行の割当てと取出しを行います。

```
SELECT * FROM sales PARTITION (nov96) s
WHERE s.amount_of_sale > 1000;
```

列の別名 *s* を使用して選択リスト中の既存の式にラベルを付け、新たな見出しを表示できます。別名によって、問合せ中に選択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 句で使用できますが、他の句では使用できません。

重複行の 1 行だけが戻されるように **DISTINCT** オプションを使用する場合、全選択リスト式の総バイト数は、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。データ・ブロックのサイズは、初期化パラメータ **DB\_BLOCK\_SIZE** によって指定されます。

Oracle オプティマイザに指示、つまりヒントを渡すために、**SELECT** 文中でコメントを使用できます。オプティマイザは、ヒントを使用して文の実行計画を選択します。ヒントの詳細は、『Oracle8 Server チューニング』を参照してください。

**例 1.** 次の文は、部門番号 30 の従業員表 **EMP** の行を選択する例です。

```
SELECT *
  FROM emp
 WHERE deptno = 30;
```

**例 2.** 次の文は、部門番号 30 の営業担当員を除く全従業員の名前、職種、給与、部門番号を選択する例です。

```
SELECT ename, job, sal, deptno
  FROM emp
 WHERE NOT (job = 'SALESMAN' AND deptno = 30);
```

**例 3.** 次の文は、**FROM** 句の副問合せから選択し、部門の全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する例です。

```
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 ;
```

## 階層問合せ

表が階層データを含んでいる場合、次の句を使用して階層順に行を選択できます。

**START WITH**      この句を使用すると、階層のルート行を指定できます。

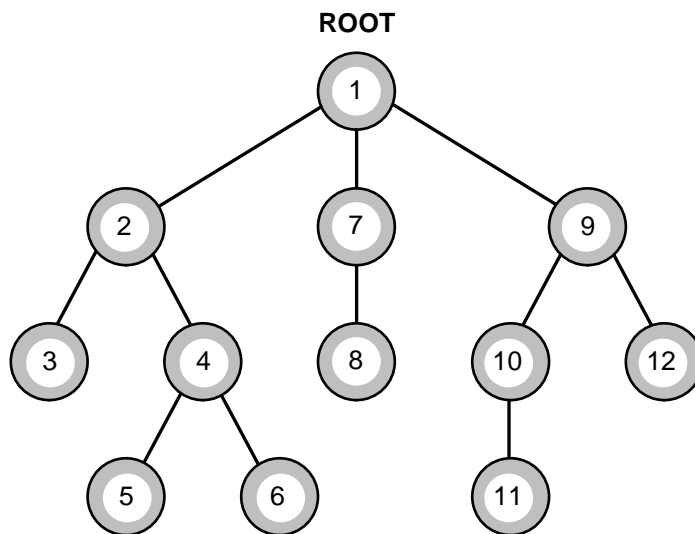
**CONNECT BY**      この句を使用すると、階層の行の親子関係を指定できます。

**WHERE**            この句を使用すると、階層の別の行に影響を及ぼすことなく、問合せで戻される行を制限できます。

Oracle は上記の句に指定された情報を使用して、次の手順に従い階層を構築します。

1. 階層のルート行を選択します。選択される行は、**START WITH** 句の条件を満たす行です。
2. 各ルート行の子行を選択します。各子行は、ルート行の1つに関して **CONNECT BY** 句の条件を満たしていなければなりません。
3. それぞれの子行の子孫となる行を選択していきます。まず、手順2で戻された行の子行が選択され、続いてさらにその孫といった順序で選択されます。Oracle は必ず現在の親行に関して **CONNECT BY** 条件を評価しながら子を選択します。
4. 問合せに **WHERE** 句が指定されている場合、Oracle は **WHERE** 句の条件を満たしていない行を階層からすべて削除します。Oracle は、条件を満たさない行の子をすべて削除するのではなく、各行ごとにこの条件を個別に評価します。
5. Oracle は、図 4-1 に示す順序で行を戻します。親の下にその子が位置付けられます。

図 4-1 階層問合せ



階層問合せを実施する **SELECT** 文には、次の制限があります。

- 階層問合せを実施する **SELECT** 文は、結合は実行できない。
- また、問合せで結合を実施するビューからはデータを選択できない。
- 階層問合せで **ORDER BY** 句を使用すると、Oracle は、図 4-1 に示した順序ではなく、**ORDER BY** 句に指定した順序で行を配置する。

次のセクションでは、**START WITH** 句と **CONNECT BY** 句を説明します。

## START WITH 句

**START WITH** 句には、階層問合せのルートとして使用される行を指定します。この句には、ルートが満たさなければならない条件を指定します。この句を省略すると、表内のすべての行がルート行として使用されます。なお、**START WITH** 句の条件には副問合せを記述できません。

## CONNECT BY 句

**CONNECT BY** 句には、階層問合せにおける行の親子関係を指定します。この句には、この関係を定義する条件を指定します。この条件は、「条件」(3-84 ページ)の構文の説明で定義されているいずれの条件でも構いません。ただし、この条件のどこかで、親行を参照するための **PRIOR** 演算子を使用しなければなりません。**PRIOR** 演算子を指定した条件は、次のいずれかの形式でなければなりません。

```
PRIOR expr comparison_operator expr  
expr comparison_operator PRIOR expr
```

親行の子を検出する場合、Oracle はその親行に対しては **PRIOR** 式を、表の各行に対してはもう一方の式を評価します。この条件が真である行は、この親の子です。**CONNECT BY** 句には、問合せで選択された行をさらに選別するために別の条件を記述できます。ただし、**CONNECT BY** 句には副問合せは記述できません。

**CONNECT BY** 句が階層の中でループすると、エラーが戻されます。ある行が親（または、親の親またはそれ以上の親元）であると同時に、別の行の子（または、孫または直系の子孫）であると、ループが発生します。

**例 1.** 次の **CONNECT BY** 句は、親行の **EMPNO** 値が子行の **MGR** 値と等しいという階層関係を定義します。

```
CONNECT BY PRIOR empno = mgr;
```

**例 2.** 次の **CONNECT BY** 句では、**PRIOR** 演算子が **EMPNO** 値にだけ適用されます。この条件を評価するために、Oracle は親行に対しては **EMPNO** の値を評価し、子行に対しては **MGR**、**SAL**、**COMM** の各値を評価します。

```
CONNECT BY PRIOR empno = mgr AND sal > comm;
```

子行を限定するには、**MGR** の値と親行の **EMPNO** の値が等しく、**SAL** の値が **COMM** の値よりも大きくなければなりません。

## LEVEL 疑似列

階層問合せを実行する **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 を、ルート・ノードの子ノードには 2 を、孫ノードには 3 をというように値が戻ります。**LEVEL** の詳細は、「疑似列」(2-30 ページ)を参照してください。

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。

**例 1.** 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' である従業員がルート行となるように定義されています。また、親行の従業員番号が子行の上司の従業員番号となるように子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     |

SELECT

|        |      |      |         |
|--------|------|------|---------|
| CLARK  | 7782 | 7839 | MANAGER |
| MILLER | 7934 | 7782 | CLERK   |

アナリストの '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 つのレベルだけが選択されます。

```
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   |

GROUP BY 句

GROUP BY 句を使うと、選択した行をグループ化して、情報を 1 行にまとめることができます。GROUP BY 句に指定されている式の値に基づいて、行の各グループが集められます。

SELECT 文に GROUP BY 句が指定してある場合、選択リストには次のような式だけを指定できます。

- 定数
- グループ関数
- 関数 USER、UID、SYSDATE
- GROUP BY 句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上記の式を伴っている式

GROUP BY 句の式には、選択リストに指定されている列に関係なく、FROM 句の表、ビュー、スナップショット内の任意の列を指定できます。

GROUP BY 句には最大 255 個の式を指定できます。GROUP BY 句に指定できる式の最大バイト数は、データ・ブロックのサイズからオーバーヘッドを引いた値となります。データ・ブロックのサイズは、初期化パラメータ DB\_BLOCK\_SIZE によって指定されます。

例 1. 次の文を発行すると、従業員表の各部門について最高給与と最低給与が戻ります。

```
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

```

例 2. 次の文を発行すると、各部門の事務員について最高給与と最低給与が戻ります。

```

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

```

## HAVING 句

HAVING 句を使うと、GROUP BY 句によって定義されたどのグループの行を問合せにより戻すかを制限できます。Oracle では、WHERE、GROUP BY、HAVING 句は次のように処理されます。

1. 問合せに WHERE 句が指定されている場合、Oracle は WHERE 句の条件を満たしていない行をすべて削除します。
2. GROUP BY 句に指定されているとおりに計算され、グループが形成されます。
3. HAVING 句を満たさないグループはすべて削除されます。

GROUP BY 句と HAVING 句は、WHERE 句と CONNECT BY 句よりも後に指定します。また、GROUP BY 句と HAVING 句は、どちらを先に指定しても構いません。

例 1. 次の文を発行すると、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が戻ります。

```

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

```

## UNION および UNION ALL、INTERSECT、MINUS

UNION、UNION ALL、INTERSECT、MINUS 演算子を使用すると、2つの問合せの結果を1つに結合できます。なお、各問合せで選択される列の数とデータ型は同じでなければなりません。列の長さは異なっても構いません。詳細は、「集合演算子」(3-12 ページ)を参照してください。

集合演算子によって3つ以上の問合せを結合する場合、隣接する問合せの組が左から右へ評価されます。この評価順序を変更するには、カッコを使用します。

コンポーネント問合せのすべての選択リスト式の最大バイト数は、データ・ブロックのサイズからオーバーヘッドを引いたサイズになります。データ・ブロックのサイズは初期化パラメータ DB\_BLOCK\_SIZE によって指定されます。

**OBJ** THE キーワードおよび MULTISSET キーワードを使用する問合せの結果は、これらの集合演算子では結合できません。

## ORDER BY 句

問合せによって選択される行に順番を付けるには、ORDER BY 句を使います。ORDER BY 句を指定しないと、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。この句で、文の選択リスト内の式、または位置、式の別名のいずれかを指定します。これらの式の値に基づいて行が戻ります。

ORDER BY 句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、次いで、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。

位置に基づくソートは次の場合に有効です。

- 長い選択リスト式によってソートする場合には、ORDER BY 句で式の位置を指定すれば、長い式を入力する必要がない。
- 複合問合せ（集合演算子 UNION、INTERSECT、MINUS、UNION ALL を含む）では、ORDER BY 句には式を明示的に指定せず、位置を指定しなければならない。なお、ORDER BY 句は、必ず最後の問合せに指定してください。ORDER BY 句により、複合問合せから戻った行の順序が決まります。

ORDER BY 句の値に対する Oracle のソート・メカニズムは、NLS\_SORT 初期化パラメータによって明示的に指定するか、NLS\_LANGUAGE 初期化パラメータによって暗黙的に指定します。これらのパラメータの詳細は、『Oracle8 Server リファレンス・マニュアル』を参照してください。また、ALTER SESSION コマンドを使用すると、ソート・メカニズムの言語ソート基準を動的に変更できます。ORDER BY 句に NLS\_SORT パラメータを指定して NLSSORT 関数を使用すると、1つの問合せに対して固有のソート順序を指定することもできます。

ORDER BY 句には、次の制限があります。



- SELECT 文に ORDER BY 句と DISTINCT 演算子が両方指定されている場合、ORDER BY 句は選択リストにある列しか参照できない。
- ORDER BY 句は、他の文中の副問合せには指定できない。
- ORDER BY 句に指定できる式の数は最大 255 個まで。

ORDER BY 句と GROUP BY 句を同じ文に指定する場合、ORDER BY 句に指定する式は、選択リスト内の式と同じ制限を受けます。これについては、「GROUP BY 句」(4-496 ページ)で説明しています。

階層問合せで ORDER BY 句を使用すると、その階層ではなく、ORDER BY 句によって行に順番が付けられます。

**例 1.** 次の文は、EMP 表から営業担当員のレコードをすべて選択し、その歩合によって降順にソートする例です。

```
SELECT *
  FROM emp
 WHERE job = 'SALESMAN'
 ORDER BY comm DESC;
```

**例 2.** 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てから、給与で降順にソートする例です。

```
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て、位置に基づく ORDER BY 句の指定を使用する例です。

```
SELECT ename, deptno, sal
  FROM emp
 ORDER BY 2 ASC, 3 DESC;
```

## FOR UPDATE 句

FOR UPDATE 句は、問合せで選択された行をロックします。更新のため行を選択すると、そのトランザクションを終了するまで、他のユーザーがその行をロックおよび更新することはできません。FOR UPDATE 句は、問合せで戻された行の挿入または更新、削除を宣言するものですが、これらの操作を必ずしも実行する必要はありません。FOR UPDATE 句を指定した SELECT 文には通常、WHERE 句を指定した UPDATE 文が 1 つ以上続きます。

FOR UPDATE 句は、次の構成体とともに使用することはできません。

- DISTINCT 演算子
- GROUP BY 句

- 集合演算子
- グループ関数
- **OBJ**CURSOR 演算子

FOR UPDATE 句でロックした表は、すべて同じデータベース内になければなりません。また、このようにロックされた表は、同じ文で参照された LONG 列や順序と同じデータベース内になければなりません。

更新のため選択した行が他のユーザーにロックされている場合、Oracle は、その行が使用可能となるまで待機します。ロックが解除されると、その行がロックされ制御がユーザーに戻ります。NOWAIT オプションを指定することによって、行がロックされている場合に待機せずにその文を終了させることができます。

**OBJ**副問合せから戻された行の列値がネストした表または VARRAY であり、スカラー値ではない場合、その行はロックされません。このような選択リストでは、最上位行だけがロックされます。

## LOB のロック

LOB 値を更新するには、その LOB を含む行をロックしておかなければなりません。行をロックする方法の 1 つに、SELECT... FOR UPDATE 文があります。

例 .

```
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;
```

## FOR UPDATE OF 句

OF 句の列は、どの表の行をロックするかを指定する場合にだけ使用します。指定した表の列すべてが重要であるとは限りません。OF 句を省略すると、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。

例 1. 次の文は、EMP 表中のニューヨーク勤務の事務員の行をロックし、DEPT 表中のニューヨークにあり事務員がいる部門の行をロックする例です。

```
SELECT empno, sal, comm
FROM emp, dept
WHERE job = 'CLERK'
      AND emp.deptno = dept.deptno
      AND loc = 'NEW YORK'
FOR UPDATE;
```

例 2. 次の文は、ニューヨーク勤務の事務員の EMP 表の行だけをロックする例です。DEPT 表の行はロックされません。

```
SELECT empno, sal, comm
FROM emp, dept
WHERE job = 'CLERK'
      AND emp.deptno = dept.deptno
      AND loc = 'NEW YORK'
FOR UPDATE OF emp.sal;
```

## 結合

結合とは、2 つ以上の表またはビュー、スナップショットの行を結合する問合せです。

Oracle では、問合せの FROM 句に複数の表が指定されていると、常に結合を実行します。問合せの選択リストを使って、複数の表から任意の表を選択し、そこから任意の列を選択できます。2 つの表に共通する列名がある場合は、混乱を避けるため、問合せでは列名は表名で修飾しなければなりません。

### 結合条件

ほとんどの結合問合せには、異なる表の 2 つの列を比較する WHERE 句条件が含まれます。そのような条件を結合条件と呼びます。結合を実行するときに、Oracle は結合条件が TRUE となる行のペアを 2 つの表から抽出し、抽出したペアを結合します。結合条件に指定された列が選択リストに入っている必要はありません。

3 つ以上の表の結合を実行するには、Oracle はまず 2 つの表の列を比較する結合条件に基づいて、それらの表を結合し、次に結合した表と新しい表の列を含む結合条件に基づいて、結合した表を新しい表に結合します。Oracle は、すべての表が結合されるまで処理を続けます。オプティマイザは、結合条件および表の索引、表の統計（コストベースの最適化の方法の場合）に基づいて表を結合する順序を決定します。

結合条件に加えて、結合問合せの WHERE 句にも 1 つの表の列だけを参照する他の条件を指定できます。これらの条件によって、結合問合せで戻される行をさらに制限できます。

### 等価結合

等価結合とは、結合条件に等価演算子を含む結合です。等価結合は、指定した列が等価の値を持つ行を結合します。結合を実行するためにオプティマイザが選択する内部アルゴリズム

によっては、1つの表の等価結合条件での列の全体のサイズが、データ・ブロックのサイズからオーバーヘッドを引いたサイズに制限される場合があります。データ・ブロックのサイズは初期化パラメータ `DB_BLOCK_SIZE` によって指定されます。

**例 1.** 次の等価結合は、各従業員の名前と職種、従業員が働いてい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      |

部門名とは別の表に従業員名とその職種が格納されているので、このデータを戻すには結合を使用する必要があります。Oracle は、次の結合条件に基づいて2つの表の行を結合します。

```
emp.deptno = dept.deptno
```

**例 2.** 次の等価結合は、すべての事務員の名前、職種、部門番号、部門名を戻す例です。

```
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** 句を使用して 'CLERK' という **JOB** 値を持つ行だけを戻すことを除けば例 12 と同じです。

```
job = 'CLERK'
```

## 内部結合

内部結合とは、同一の表の結合です。結合する表を **FROM** 句に 2 回指定し、その後にそれぞれ表の別名を指定して、結合条件で列名を修飾するために使用します。内部結合を実行すると、Oracle は結合条件を満たす表の行を結合して戻します。

**例.** この問合せでは、各従業員の名前と、従業員の上司の名前を戻す内部結合を使用します。

```
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 と E2 を使用します。

```
e1.mgr = e2.empno
```

## 直積演算

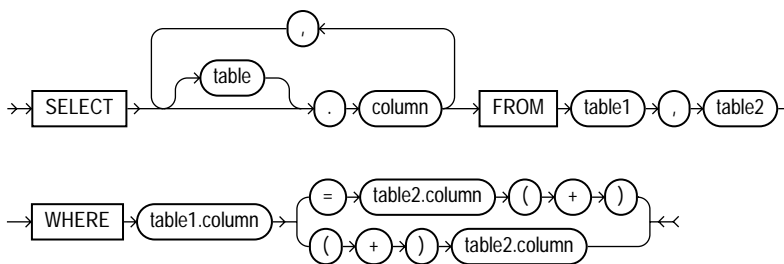
結合問合せをする 2 つの表に結合条件がない場合、Oracle は表の直積を戻します。すなわち、1 つの表の各行にもう 1 つの表の各行を結合します。直積では常に大量の行が生成されますが、役に立つことはほとんどありません。たとえば、それぞれが 100 行からなる 2 つの表の直積は 10000 行になります。特に直積が必要な場合を除いて、結合条件を指定してください 3 つ以上の表を結合する問合せを行い、特定の対を形成するための結合条件を指定しないと、オプティマイザにより中間的直積の生成を避けるための結合順序が選択される場合があります。

## 外部結合

外部結合は単純結合の結果を拡張します。外部結合は、結合条件を満たすすべての行と、1つの表の結合条件を満たしていない行を戻します。結合条件を満たしていない行は、単純結合では戻すことはできません。表 A と B の外部結合を実行し、A の行をすべて戻す問合せを作成するには、外部結合演算子 (+) を結合条件内の B のすべての列に適用します。B の行と一致しない A のすべての行について、B の列を含む選択リストの式に NULL 値が戻されます。

これは、2つの表の外部結合の基本構文です。

`outer_join ::=`



外部結合の問合せは、次の規則と制限に従います。

- (+) 演算子は選択リストではなく WHERE 句だけに指定でき、表またはビューの列だけに適用できる。
- A および B が複数の結合条件によって結合されている場合は、(+) 演算子をすべての条件で使用する。
- (+) 演算子は、任意の式ではなく、1つの列にだけ適用できる。ただし、(+) 演算子のマークが付いた列は任意の式に指定できる。
- (+) 演算子を含む条件を、OR 論理演算子を使用する他の条件と組み合わせることはできない。
- 条件では、IN 比較演算子を使用して、(+) 演算子のマークが付いた列と別の式を比較することはできない。
- 条件では、(+) 演算子のマークが付いた列と副問合せは比較できない。

WHERE 句に表 B の列と定数とを比較する条件が定義されている場合は、この列の NULL 値が生成されている表 A の行が戻されるように、(+) 演算子をこの列に適用しなければなりません。

2組以上の表の外部結合を実行する問合せでは、1つの表だけを他の表に対して NULL を生成する表にできます。このため、A と B の結合条件と、B と C の結合条件で B の列に (+) 演算子を適用することはできません。

例 1. この問合せでは、外部結合を使用して例 14 の結果を拡張しています。

```
SELECT ename, job, dept.deptno, dname
       FROM emp, dept
       WHERE emp.deptno (+) = dept.deptno;
```

| ENAME  | JOB       | DEPTN | 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。Oracle は、この列の ENAME 列と JOB 列に NULL 値を戻します。例 10 の結合問合せは、従業員のいる部門だけを選択します。

次の問合せは、外部結合を使用して例 15 の結果を拡張しています。

```
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 列に対して (+) 演算子を使用すると、JOB 列が NULL である行も戻されます。この (+) が省略された場合、JOB 列の値が 'CLERK' ではないので、OPERATIONS 部門を含む行は戻されません。

例 2. この例は、CUSTOMERS、ORDERS、LINEITEMS、PARTS の 4 つの表の外部結合問合せを示しています。これらの表を次に示します。

```
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-13-1993 |
| 9002    | 2      | OCT-13-1993 |
| 9003    | 1      | OCT-20-1993 |
| 9004    | 1      | OCT-27-1993 |
| 9005    | 2      | OCT-31-1993 |

```
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 は注文をしていないこと、注文番号 9005 の品目がないことに注意してください。

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。(+) 演算子を使用すると、注文をしていない顧客名も戻されます。

```
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、この表の列を選択リストに、この表を ORDERS 表と結合する結合条件を WHERE 句に追加して、前の外部結合の結果に別の情報を加えています。この問合せは、LINEITEMS 表への前の問合せの結果を結合し、すべての顧客名、および注文した日付、注文のあった各部品の部品番号と数量を戻します。最初の (+) 演算子は前の問合せと同じ働きをします。2 番目の (+) 演算子によって、品目がない注文も戻されるようになります。

```
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 列を選択リストに、この表を LINEITEMS 表に結合する結合条件を WHERE 句に追加することで、前の外部結合の結

果に別の情報を加えています。この問合せは、PARTS 表へ前の問合せの結果を結合し、すべての顧客名、および注文した日付、注文のあっ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、前の問合せと同じ働きをします。3 番目の (+) 演算子によって、部品番号が NULL の行も戻されるようになります。

```
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    |             |          |              |

関連項目

- DELETE (4-370 ページ)
- SET CONSTRAINT(S) (4-509 ページ)
- UPDATE (4-536 ページ)

## SET CONSTRAINT(S)

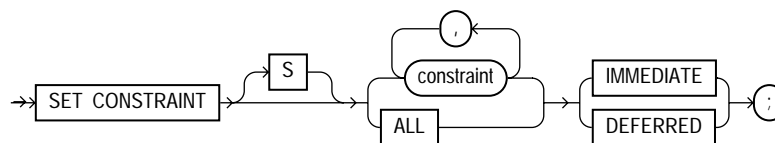
### 用途

遅延可能な制約のチェックを、各 DML 文の実行後に行うか (IMMEDIATE)、トランザクションのコミット時に行うか (DEFERRED) をトランザクションごとに指定します。使用例は、「例」(4-509 ページ) を参照してください。

### 前提条件

遅延可能な制約をチェックする時期を設定する場合は、制約が適用される表が自スキーマ内にあるか、その表に対する SELECT 権限を持っている必要があります。

### 構文



### キーワードとパラメータ

|            |                                             |
|------------|---------------------------------------------|
| constraint | 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が設定されます。             |
| IMMEDIATE  | 遅延可能な制約によって指定される条件が各 DML 文の直後にチェックされます。     |
| DEFERRED   | 遅延可能な制約によって指定される条件がトランザクションのコミット時にチェックされます。 |

SET CONSTRAINTS ALL IMMEDIATE 文を発行することにより、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

**例 1.** 次の例では、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;
```

例 2. 次の文では、トランザクションのコミット時に 3 つの遅延制約がチェックされます。

```
SET CONSTRAINTS unq_name, scott.nn_sal,  
                adams.pk_dept@dblink DEFERRED;
```

### 関連項目

[CREATE TABLE](#) (4-303 ページ)

[ALTER TABLE](#) (4-105 ページ)

[ENABLE 句](#) (4-414 ページ)

[DISABLE 句](#) (4-375 ページ)

[ALTER SESSION](#) (4-58 ページ)

# SET ROLE

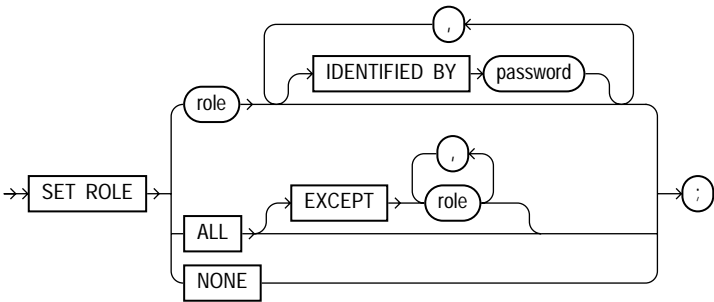
## 用途

ユーザーの現行セッションに対してロールを使用可能または使用禁止にします。使用例は、「例」(4-513 ページ)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なければなりません。「権限ドメイン」(4-512 ページ)を参照してください。

## 構文



## キーワードとパラメータ

|                 |                                                                                                                                         |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <i>role</i>     | 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となります。                                                                                  |
| <i>password</i> | 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、指定しなければなりません。                                                                                       |
| 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。                                                                     |
| 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ものでなければなりません。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なりません。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。 |

---

ユーザーに直接付与されており、同時に他のロールを通じて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の効力によって、そのロールは使用可能のままになります。

NONE

現行セッションですべてのロールは使用禁止となります。

---

## 権限ドメイン

ユーザーがログオンすると、Oracle は、デフォルト・ロールを使用可能にすることによってデフォルトの権限ドメインを設定します。デフォルトの権限ドメインには、そのユーザーに明示的に付与されているすべての権限、およびデフォルト・ロールの権限ドメイン内のすべての権限が登録されています。ログオンしたユーザーは、デフォルト権限ドメイン内の権限によって許可される操作を実行できます。

### 権限ドメインの変更

SET ROLE コマンドを使用すると、セッション中に権限ドメインを変更できます。このコマンドによって、現行セッションで使用可能になっているロールを変更できます。使用可能になっているロールは同一セッション中に何度でも変更できます。ただし、同時に使用可能にできるロールの数は、MAX\_ENABLED\_ROLES 初期化パラメータの値によって制限されます。

SET ROLE コマンドを使用して、次のロールを使用可能または使用禁止にできます。

- ユーザーに直接付与されているロール。
- 他のロールを通じてユーザーに付与されているロール。

直接にも他のロールによっても付与されていないロールは、SET ROLE コマンドを使って使用可能にすることはできません。

また、次の場合には現行の権限ドメインが変更されます。

- 権限が付与された場合。
- 権限が1つでも取り消された場合。
- 使用可能になっているロールが1つでも取り消された場合。
- 使用可能になっているロールの権限ドメインが1つでも変更された場合。

このようなことが発生しない場合、SET ROLE コマンドを発行するまでデフォルトの権限ドメインは、現行セッションの存続期間中は有効です。上記の最後の2つによる理由で、権限ドメインが変更されても、その変更は Oracle に再度ログオンするか、SET ROLE 文を発行するまで反映されません。

現行の権限ドメインに登録されているロールを確認するには、SESSION\_ROLES データ・ディクショナリ・ビューを調べます。

デフォルトのロールを変更するには、ALTER USER コマンドを使用します。

## 例

例 1. 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** を使用可能にする例です。

```
SET ROLE gardener IDENTIFIED BY marigolds;
```

例 2. 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する例です。

```
SET ROLE ALL;
```

例 3. 次の文は、**BANKER** を除くロールをすべて使用可能にする例です。

```
SET ROLE ALL EXCEPT banker IV;
```

例 4. 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禁止にする例です。

```
SET ROLE NONE;
```

## 関連項目

[ALTER USER](#) (4-148 ページ)

[CREATE ROLE](#) (4-268 ページ)

# SET TRANSACTION

## 用途

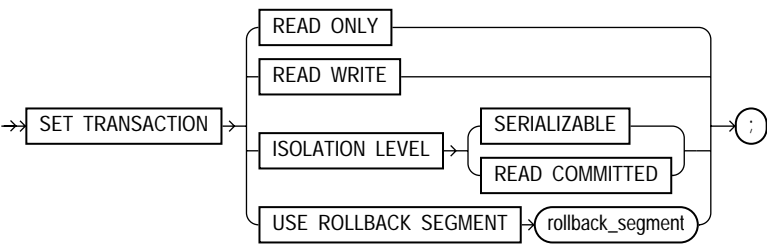
- 現行のトランザクションに対して次のことを行います。
- トランザクションを読取り専用、または読み書き両用トランザクションとして設定する。
  - 分離レベルを設定する。
  - 指定したロールバック・セグメントにトランザクションを割り当てる。

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だけに影響します。他のユーザーや他のトランザクションに影響はありません。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、データ定義言語文が実行される前後に暗黙的にコミットされます。

## 前提条件

SET TRANSACTION 文を使用するときは、トランザクションの先頭に記述しなければなりません。ただし、SET TRANSACTION 文の必要の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       | 現行トランザクションを読取り専用を設定します。「読取り専用トランザクションの設定」(4-515 ページ) 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します。                                        |
| 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します。                          |



|                                    |                                                                                                                                                                                                                                                                                                     |
|------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                    | <p><b>SERIALIZABLE</b> SQL92 に定義されている直列可能トランザクション分離モードを指定します。直列可能トランザクションに更新のデータ操作言語 (DML) が含まれている場合に、その更新が直列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更新されている可能性のあるリソースを対象とするものであれば、その DML 文は正常に実行されません。</p> <p><b>注意：</b>SERIALIZABLE モードで運用するには、COMPATIBLE 初期化パラメータを 7.3.0 以上に設定する必要があります。</p>           |
|                                    | <p><b>READ COMMITTED</b> デフォルトの Oracle トランザクションの動作です。別のトランザクションで行ロックを保持しておく必要のある 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。</p>                                                                                                                                                            |
| <p><b>USE ROLLBACK SEGMENT</b></p> | <p>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ます。このオプションの場合も、現行トランザクションは暗黙的に読み書き両用になります。</p> <p>1 つの SET TRANSACTION 文または 1 つのトランザクション内の異なる文に、READ ONLY オプションと USE ROLLBACK SEGMENT 句の両方は指定できません。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。「ロールバック・セグメントへのトランザクションの割当て」(4-516 ページ) を参照してください。</p> |

## 読取り専用トランザクションの設定

トランザクションのデフォルトでは、文レベルの読取り一貫性が維持されています。READ WRITE オプションを指定した SET TRANSACTION 文を実行すると、この状態を明示的に指定できます。

READ ONLY オプションを指定した SET TRANSACTION 文を発行することによって、トランザクション・レベルの読取り一貫性を設定できます。トランザクションを読取り専用にすると、そのトランザクションで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以前にコミットされた変更だけが参照されます。この読取り専用トランザクションは、他のユーザーが更新している 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非常に便利です。

読取り専用トランザクションは、次の文だけを使用できます。

- SELECT (FOR UPDATE 句を指定した場合を除く)
- LOCK TABLE
- SET ROLE
- ALTER SESSION
- ALTER SYSTEM

INSERT 文および UPDATE 文、DELETE 文、FOR UPDATE 句を指定した SELECT 文は使用できません。データ定義語文は、読取り専用トランザクションを暗黙的に終了します。

読取り専用トランザクションが備えている読取り一貫性は、文レベルの読取り一貫性と同じです。デフォルトでは、文を発行した時点の一貫性のあるデータのビューが使用されます。読取り専用トランザクションでは、**SET TRANSACTION READ ONLY** 文を発行した時点の一貫性のあるデータのビューが使用されます。読取り一貫性を実現する読取り専用トランザクションは、ローカル問合せおよび分散問合せによってアクセスされたすべてのノードが対象です。

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・レベルと文レベル）を、切り替えることができません。**SET TRANSACTION** 文は、トランザクションの最初の文としてだけ発行できます。

**例** .次に示すのは、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する例です。このレポート作成のプロセスは、船舶やコンテナを追加または削除している他のユーザーのプロセスには影響しません。

```
COMMIT
SET TRANSACTION READ ONLY
SELECT COUNT(*) FROM ship
SELECT COUNT(*) FROM container
COMMIT;
```

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のものではありません。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。

## ロールバック・セグメントへのトランザクションの割当て

トランザクションでデータ操作言語文を発行すると、そのトランザクションにロールバック・セグメントが割り当てられます。ロールバック・セグメントには、トランザクションによる変更を取り消すために必要な情報が保持されます。**SET TRANSACTION** 文に **USE ROLLBACK SEGMENT** 句を指定すると、ユーザーのトランザクションに特定のロールバック・セグメントを選択できます。ロールバック・セグメントを選択しないと、Oracle によってランダムに選択されたロールバック・セグメントがトランザクションに割り当てられます。

**SET TRANSACTION** 文によって、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。

- **OLTP** トランザクション、すなわち 2、3 行を修正するデータ操作言語文が少数あるだけの小さなトランザクションには、同時に同じ表を読み取る大規模な問合せがなければ、小さなロールバック・セグメントを割り当てます。小さなロールバック・セグメントは、メモリー内に保持される可能性が高くなります。
- 複数の大規模な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、大きなロールバック・セグメントを割り当てます。読取り一貫性を実現する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。

- 大量のデータ操作言語文、すなわち大量のデータを挿入、更新、削除する文のあるトランザクションには、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に十分な大きさのロールバック・セグメントを割り当てます。

**例.** 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;
```

## 関連項目

`COMMIT` (4-182 ページ)

`ROLLBACK` (4-481 ページ)

`SAVEPOINT` (4-484 ページ)

## STORAGE 句

---

### 用途

表および索引、クラスタ、ロールバック・セグメントの記憶特性、および表領域のデフォルトの記憶特性を指定します。「使用上の注意」(4-521 ページ) も参照してください。

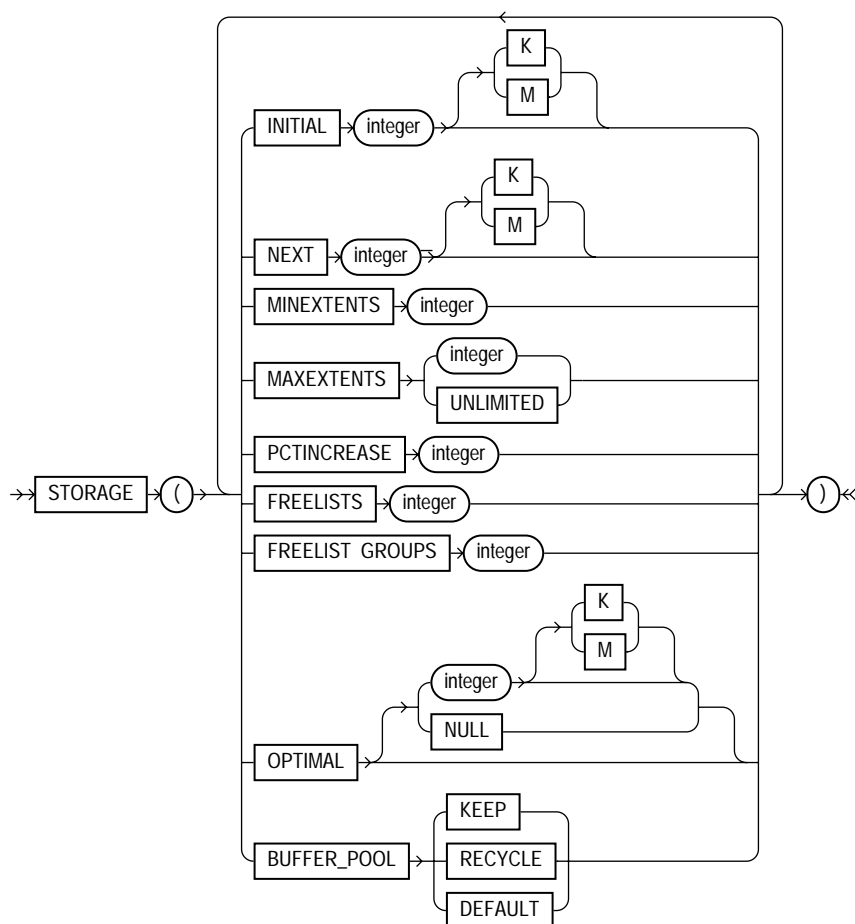
### 前提条件

STORAGE 句は、次のスキーマ・オブジェクトを作成または変更するコマンドに指定できます。

- 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
- 表領域
- パーティション

STORAGE パラメータの値を変更するには、適切な CREATE コマンドまたは ALTER コマンドを使用するために必要な権限を持っていなければなりません。

## 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、このエクステントが割り当てられます。K または M を使用して KB または MB 単位でもサイズを指定できます。デフォルトのサイズは 5 データ・ブロックです。最小のサイズは 2 データ・ブロックです。最大値は使用しているオペレーティング・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ると、データ・ブロック・サイズの一番近い倍数に丸められます。5 データ・ブロック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値が切り上げられます。

|                 |                                                                                                                                                                                                                                                                                                                                                  |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NEXT            | <p>オブジェクトに割り当てる次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 単位でもサイズを指定できます。デフォルトのサイズは 5 データ・ブロックです。最小のサイズは 1 データ・ブロックです。最大値は使用しているオペレーティング・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すると、データ・ブロック・サイズの一番近い倍数に丸められます。5 データ・ブロックを超える値の場合は、断片化を最小限にする値に切り上げられます。これについては、『Oracle8 Server 概要』を参照してください。</p>                                                |
| PCTINCREASE     | <p>3 番め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。最大値は使用しているオペレーティング・システムによって異なります。</p> <p>ロールバック・セグメントには 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 です。</p> <p>算出された各エクステントのサイズは、データ・ブロック・サイズの倍数に切り上げられます。</p> |
| MINEXTENTS      | <p>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うと、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 エクステントだけが割り当てられます。ただし、ロールバック・セグメントの場合は、最小値（デフォルト）は 2 です。最大値は使用しているオペレーティング・システムによって異なります。</p> <p>MINEXTENTS の値が 1 より大きい場合、INITIAL および NEXT、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算出されます。</p>                       |
| MAXEXTENTS      | <p>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）。デフォルト値や最大値はデータ・ブロックのサイズによって異なります。</p> <p>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指定します。ロールバック・セグメントにはこのオプションは使用しないでください。</p> <p>「ロールバック・セグメントと MAXEXTENTS UNLIMITED」（4-522 ページ）も参照してください。</p>                                               |
| FREELIST GROUPS | <p>表領域以外のスキーマ・オブジェクトの場合に、表またはパーティション、クラスタ、索引の空きリスト・グループ数を指定します。このパラメータのデフォルト値と最小値は 1 です。このパラメータは、Parallel Server オプション付きの Oracle をパラレル・モードで使用している場合にだけ指定してください。</p>                                                                                                                                                                              |
| FREELISTS       | <p>表領域以外のオブジェクトの場合に、表またはパーティション、クラスタ、索引の各空きリスト・グループについて空きリスト・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。この場合、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応じて異なります。FREELISTS に指定した値が大きすぎると、最大値を示すエラー・メッセージが戻ります。</p>                                                                                                                               |

---

|             |                                                                                                                                                                                                                                                                         |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | FREELISTS と FREELIST GROUPS の各パラメータは、CREATE TABLE 文、CREATE CLUSTER 文、CREATE INDEX 文にだけ指定できます。                                                                                                                                                                           |
| OPTIMAL     | このパラメータは、ロールバック・セグメントだけに指定します。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 を使用して KB または MB 単位でもサイズを指定できます。Oracle は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となった場合に、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、指定したロールバック・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、できるだけ多数のエクステントの割当てを解除します。 |
| NULL        | ロールバック・セグメントに最適サイズを指定しません。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。                                                                                                                                                                                       |
|             | このパラメータには、MINEXTENTS および INITIAL、NEXT、PCTINCREASE の各パラメータに指定したロールバック・セグメントの 1 次領域より小さい値は指定できません。最大値は使用しているオペレーティング・システムによって異なります。データ・ブロック・サイズの一番近い倍数に丸められます。                                                                                                            |
| BUFFER_POOL | 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には、パーティションは、パーティション・レベル定義で変更されないかぎり、表定義または索引定義のバッファ・プールを継承します。                                                                                      |
|             | 注意：BUFFER_POOL は、表領域やロールバック・セグメントを作成したり変更したりする場合は、有効なオプションではありません。複数のバッファ・プールの使用方法は、『Oracle8 Server チューニング』を参照してください。                                                                                                                                                   |
| KEEP        | I/O 操作を避けるため、メモリーのスキーマ・オブジェクトを保持します。                                                                                                                                                                                                                                    |
| RECYCLE     | 不要なブロックはただちにメモリーから排除されます。これにより、オブジェクトが不要なキャッシュ領域を占めなくなります。                                                                                                                                                                                                              |
| DEFAULT     | KEEP も RECYCLE も指定しないオブジェクト用として、デフォルトのバッファ・プールが定義されています。                                                                                                                                                                                                                |

---

## 使用上の注意

STORAGE パラメータは、データベースのデータへのアクセス時間とデータベース内での領域の利用の効率性の両方に影響します。これらのパラメータの影響についての説明は、『Oracle8 Server チューニング』を参照してください。

表領域の作成時に、STORAGE 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となります。

クラスタまたは索引、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、表を作成するときには、これらのオブジェクトに割り当てられるセグメントの STORAGE パラメータの値を指定できます。STORAGE パラメータを指定しないと、表領域

に指定されているパラメータの値が使用されます。ただし、ロールバック・セグメントを作成するときは、**PCTINCREASE** (常に 0) または **MINEXTENTS** (常に 2) は指定できません。

クラスタまたは索引、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、表を変更するときに、**STORAGE** パラメータの値を変更できます。この値を変更しても、それ以降のエクステントの割当てにしか影響しません。このため、**INITIAL** パラメータおよび **MINEXTENTS** パラメータの値は変更できません。**NEXT** パラメータの値を変更すると、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や **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。また、**PCTINCREASE** パラメータの値を変更すると、この変更した値と直前に割り当てられたエクステントのサイズに基づいて、次に割り当てるエクステントのサイズが算出されます。

表領域を変更するときに、**STORAGE** 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。

## ロールバック・セグメントと MAXEXTENTS UNLIMITED

ロールバック・セグメントを作成または変更するときには、**MAXEXTENTS UNLIMITED** を使用しないでください。長時間に渡って挿入または更新、削除を続ける特殊なトランザクションでは、ディスクがいっぱいになるまで新規エクステントの作成をいつまでも続けます。

**STORAGE** オプションを指定しないで作成されたロールバック・セグメントには、このロールバック・セグメントが作成された表領域の **STORAGE** オプションが適用されます。したがって、表領域が **MAXEXTENT UNLIMITED** で作成されていると、ロールバック・セグメントも同じデフォルト値を持つことになります。

## 例

例 1. 次の文は、表の作成時に **STORAGE** 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、次に示すように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 100K のため、第 1 エクステントのサイズは 100K になります。
- 表データが第 1 エクステントを超えると、第 2 エクステントが割り当てられます。第 2 エクステントのサイズは **NEXT** の値が 50K のため、50K になります。



- 表データが増えて最初の2つのエクステントを超えると、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 となります。このときデータ・ブロック・サイズが2KB であれば、値は丸められて52KB となります。

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 となります。

**例 2.** 次の文は、表の作成時に STORAGE 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 つ割り当てられます。
- INITIAL の値は10K のため、第1エクステントのサイズは10K となります。
- NEXT の値は10K のため、第2エクステントのサイズは10K 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ると、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 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50K のため、ロールバック・セグメントが50KB を超えると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だけです。

## 関連項目

CREATE CLUSTER (4-202 ページ)

CREATE INDEX (4-233 ページ)

CREATE ROLLBACK SEGMENT (4-272 ページ)

CREATE TABLE (4-303 ページ)

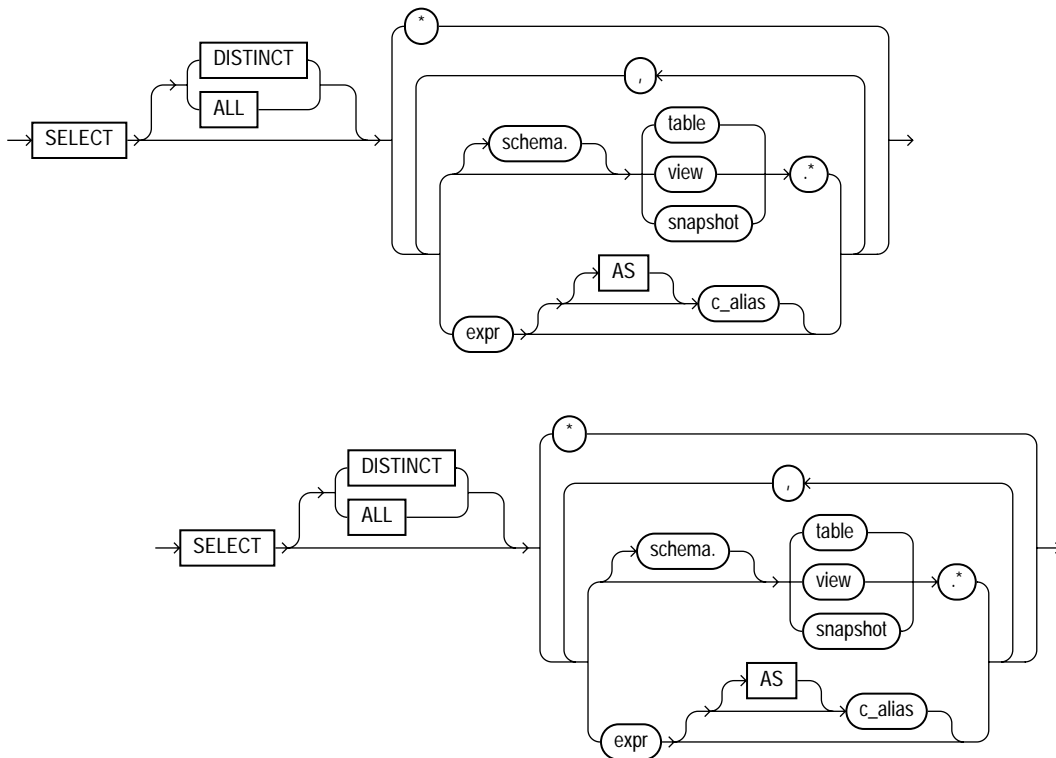
CREATE TABLESPACE (4-325 ページ)

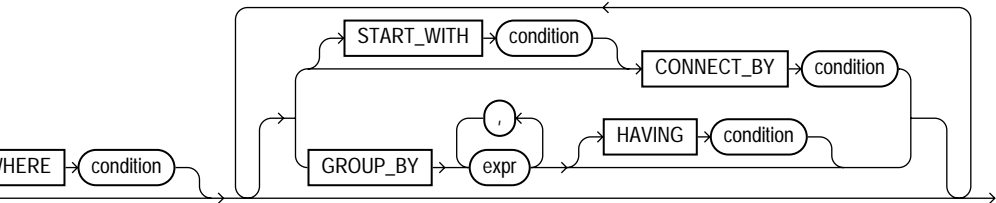
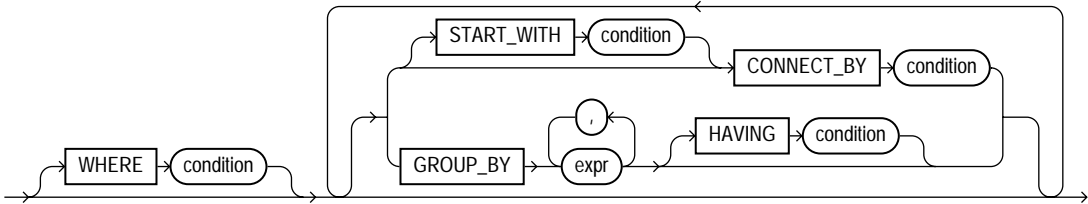
## 副問合せ

### 用途

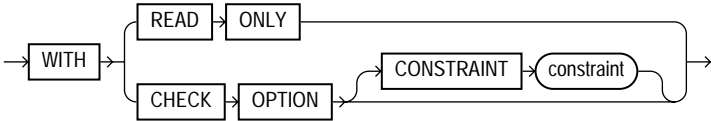
副問合せとは、他の SQL 文の中に指定されている **SELECT** コマンドの形式です。副問合せはネストされた問合せともいいます。副問合せが定義されている文を親文といいます。副問合せで戻る行は、親文で使用されます。「使用上の注意」(4-527 ページ) も参照してください。

### 構文





WITH\_clause::=



キーワードとパラメータ

|                   |                                                                                        |
|-------------------|----------------------------------------------------------------------------------------|
| WITH READ ONLY    | 副問合せを更新禁止にすることを指定します。                                                                  |
| WITH CHECK OPTION | INSERT、UPDATE、DELETE の各文で表のかわりに副問合せが使用された場合に、副問合せから除外されるような行を生成する表変更は禁止されます。例を以下に示します。 |

```
INSERT INTO (SELECT ename, deptno FROM emp
              WHERE deptno < 10)
VALUES ('Taylor', 20);
```

上の文は有効です。

```
INSERT INTO (SELECT ename, deptno FROM emp
              WHERE deptno < 10
              WITH CHECK OPTION)
VALUES ('Taylor', 20);
```

上の文は無効です。

---

**OBJ** THE 副問合せが戻す列値がスカラー値でなく、ネストした表であることを示します。接頭辞 THE の付いた副問合せはフラット化した副問合せと呼ばれます。「フラット化した副問合せの使用」(4-528 ページ) を参照してください。

**OBJ** TABLE(*nested \_table\_column*) 外部問合せと相関関係にあるネストした表の列を示します。

---

その他のキーワードとパラメータの機能の説明は、SELECT (4-486 ページ) を参照してください。詳細は、「相関副問合せ」(4-528 ページ) および「DUAL 表からの選択」(4-530 ページ)、「順序の使用」(4-530 ページ)、「分散問合せ」(4-530 ページ) を参照してください。

## 使用上の注意

副問合せは、次の目的のために使用できます。

- INSERT 文または CREATE TABLE 文の対象となる表に挿入する行のセットの定義。
- CREATE VIEW 文または CREATE SNAPSHOT 文において、ビュー、スナップショットに指定する行セットの定義。
- UPDATE 文において、既存の行に割り当てる 1 つ以上の値の定義。
- SELECT 文および UPDATE 文、DELETE 文の WHERE 句および HAVING 句、START WITH 句の条件の値の指定。
- 親文の問合せの対象となる表の定義。この表は親の問合せの FROM 句に、表名の場合と同様に、副問合せを指定することにより定義します。INSERT 文、UPDATE 文、DELETE 文でも、このようにして表のかわりに副問合せを使用します。このとき、副問合せで相関変数を指定できます。しかし、外部参照変数ではなく、その副問合せ自体の中に定義されるものだけです。

1 つの副問合せでマルチ・パート問合せを実行できます。たとえば、TAYLOR が働いている部門の従業員を調べるには、まず、TAYLOR が働いている部門を副問合せで問い合わせます。続いて、親の SELECT 文を使ってその部門の従業員を調べます。

相関副問合せは、親文が行を処理するたびに算出されるのに対して、副問合せは、親文全体に対して 1 度に算出されます。

副問合せ自体に副問合せを定義することもできます。問合せのネスト・レベルに制限はありません。

**例 1.** 次の文は、TAYLOR が働いている部門の従業員を示します。

```
SELECT ename, deptno
  FROM emp
 WHERE deptno =
        (SELECT deptno
         FROM emp
         WHERE ename = 'TAYLOR');
```

**例 2.** 次の文は、賞与が支給されていない (BONUS 表に存在しない) EMP 表の従業員の給与を 10% 昇給します。

```
UPDATE emp
  SET sal = sal * 1.1
  WHERE empno NOT IN (SELECT empno FROM bonus);
```

**例 3.** 次の文は、NEWDEPT という名前の DEPT 表の複製を作成します。

```
CREATE TABLE newdept (deptno, dname, loc)
  AS SELECT deptno, dname, loc FROM dept;
```

## フラット化した副問合せの使用

**【例】** データベースの列内に格納されているネストした表の個々の行を操作する場合は、キーワード **THE** を使います。1 つの列値、またはネストした表を生成する式を戻す副問合せの先頭には、**THE** を付けなければなりません。その副問合せが複数の列値を戻す場合は、ランタイム・エラーになります。戻り値は、スカラー値でなく、ネストした表であるため、**THE** を使ってこれを指定しなければなりません。

次の文は、列 **PROJECTS** に格納されている部門 40 のネストした表に新規行を追加します。

```
INSERT INTO
  THE(SELECT projects FROM dept WHERE deptno = 40)
VALUES(33, 'Install new email system', 14875);
```

次は、部門 70 に割り当てられた 2 つのプロジェクトの予算を引き上げる例です。

```
UPDATE
  THE(SELECT projects FROM dept WHERE deptno = 70)
  SET budget = budget + 1000
  WHERE projno IN (24, 25);
```

## 関連副問合せ

関連副問合せとは、親文が行を処理するたびに算出される副問合せです。親文は **SELECT** 文、**UPDATE** 文、**DELETE** 文です。次に、関連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);

```

ここでは SELECT 文の相関副問合せを取り上げますが、UPDATE 文、DELETE 文にも共通します。

1 つの相関副問合せを使用して、マルチ・パート問合せを処理できます。応答の内容は、親文が処理する各行の値によって異なります。たとえば、相関副問合せを使用して、部門の平均給与以上の給与を支給されている従業員を判別できます。この場合、相関副問合せは特に各部門の平均給与を算出します。

副問合せが親文の表の列を参照すると、相関副問合せが実行されます。

Oracle は副問合せ中の未修飾の列に出会うと、副問合せの表、親文の表、関係する次の親文の表という順で参照することによって列名を解決します。Oracle は、副問合せ中の未修飾の列をすべて同じ表の列であるとみなします。副問合せと親問合せ中の表に、同じ名前の列がある場合、および親問合せでその列を参照する場合、表名または表の別名を付けて修飾しなければなりません。文を読みやすくするため、相関副問合せ中の列を、表、ビュー、スナップショットの名前またはそれらの別名で修飾してください。

UPDATE 文の場合、相関副問合せを使用して他の表の行に基づく表の行を更新できます。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。

また、DELETE 文の場合、相関問合せを使って他の表にもある行だけを削除できます。

**例** . 次の文は、部門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す例です。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 表の各行ごとに算出します。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 を判別します。
2. DEPTNO に基づいて親問合せが算出されます。
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。

副問合せでは、EMP 表が 1 度に 1 行ずつ算出されます。

## DUAL 表からの選択

DUAL 表は、データ・ディクショナリとともに Oracle によって自動的に作成される表です。DUAL はユーザー SYS のスキーマ内にありますが、DUAL という名前ですべてのユーザーからアクセスできます。DUAL には VARCHAR2(1) として定義された DUMMY という列、および 'X' の値を持つ行が設定されています。DUAL 表からの選択は、SELECT コマンドを使用した定数の式の計算に便利です。DUAL には 1 行しかないため、その定数は 1 度だけ戻ります。別の方法として、任意の表から定数および疑似列、式を選択できます。

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 表の全行に対して 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。

## 順序の使用

順序疑似列 NEXTVAL および CURRVAL も、SELECT 文の選択リストの中で使用できます。順序の説明と使用方法については、CREATE SEQUENCE (4-278 ページ) および「疑似列」(2-30 ページ) を参照してください。

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;
```

## 分散問合せ

Oracle の分散データベース管理システム・アーキテクチャでは、Net8 と Oracle Server を使用してリモート・データベース上のデータにアクセスできます。リモートの表およびビュー、スナップショットは、末尾に @dblink を付けることによって識別されます。dblink は、リモートの表およびビュー、スナップショットが定義されている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でなければなりません。データベース・リンクの参照方法の詳細は、「リモート・データベース内のオブジェクトを参照」(2-50 ページ) を参照してください。

現在、分散問合せには次の制限があります。FOR UPDATE 句によりロックされるすべての表、および問合せにより選択される LONG 列のあるすべての表は、同じデータベース内に定義されていなければなりません。たとえば、次の文はエラーを発生させます。

```
SELECT emp_ny.*  
FROM emp_ny@ny, dept
```



```
WHERE emp_ny.deptno = dept.deptno
AND dept.dname = 'ACCOUNTING'
FOR UPDATE OF emp_ny.sal;
```

下の文は、NY データベースの EMP\_REVIEW 表から LONG\_COLUMN という LONG 値を選択して、ローカル・データベースの EMP 表をロックするため、失敗します

```
SELECT emp.empno, review.long_column, emp.sal
FROM emp, emp_review@ny review
WHERE emp.empno = emp_review.empno
FOR UPDATE OF emp.sal;
```

例 . 次は、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 表を結合する例です。

```
SELECT ename, dname
FROM emp@houston, dept
WHERE emp.deptno = dept.deptno;
```

## 関連項目

DELETE (4-370 ページ)

SET CONSTRAINT(S) (4-509 ページ)

UPDATE (4-536 ページ)

# TRUNCATE

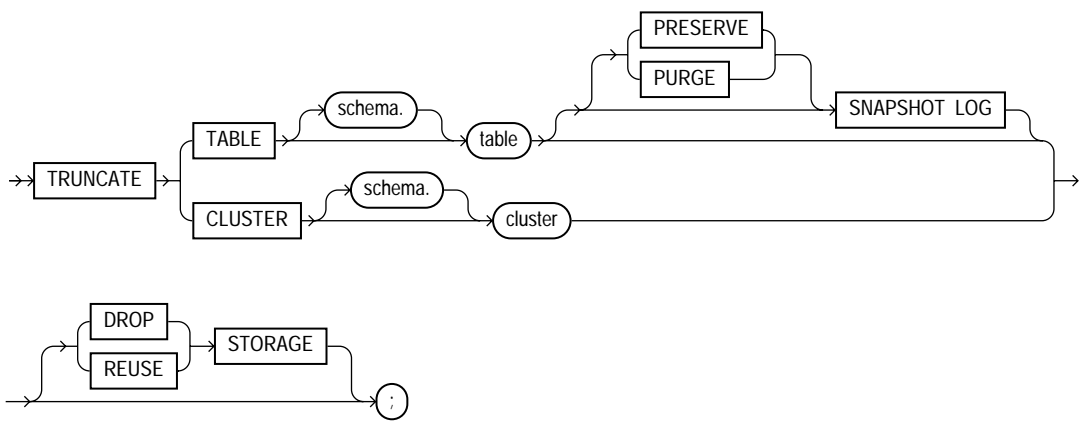
## 用途

表またはクラスタの行をすべて削除して、**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します。「使用上の注意」（4-533 ページ）も参照してください。使用例は、「例」（4-534 ページ）を参照してください。

## 前提条件

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。「制限事項」（4-534 ページ）も参照してください。

## 構文



## キーワードとパラメータ

|               |                                                                                                                    |
|---------------|--------------------------------------------------------------------------------------------------------------------|
| <i>schema</i> | 作成するトリガーを定義するスキーマです。 <i>schema</i> を指定しないと、自スキーマにトリガーが作成されます。                                                      |
| <b>TABLE</b>  | 切り捨てる表が設定されているスキーマとその表の名前を指定します。索引構成表も切り捨てることができます。クラスタを構成する表は切り捨てることができません。<br>表を切り捨てる時、表の索引の中のデータが自動的にすべて削除されます。 |

---

|               |                                                                                                                                                                                                                                                                                            |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SNAPSHOT LOG  | 表が切り捨てられたときに、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、削除するかを指定します。この句を使うと、スナップショット・マスター表を、エクスポートとインポートにより再編成できます。このとき、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には、スナップショット・ログに主キー情報を記録しなければなりません。スナップショット・ログと TRUNCATE コマンドの詳細は、『Oracle8 Server レプリケーション』を参照してください。 |
| PRESERVE      | マスター表を切り捨てた場合に、スナップショット・ログを保存することを指定します。これはデフォルト値です。                                                                                                                                                                                                                                       |
| PURGE         | マスター表を切り捨てた場合に、スナップショット・ログも削除することを指定します。                                                                                                                                                                                                                                                   |
| 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。<br><br>クラスタを切り捨てると、そのクラスタの表の索引データも自動的にすべて削除されます。                                                                                                                                                            |
| DROP STORAGE  | 削除した行に割り当てられていた領域を表またはクラスタから解放します。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 | 削除された行の領域を、その表またはクラスタに割り当てられたまま残します。STORAGE 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だけ使用されます。<br><br>DROP STORAGE オプションおよび REUSE STORAGE オプションは、対応付けられた索引から削除されたデータの領域にも適用されます。                                                                          |

---

## 使用上の注意

TRUNCATE コマンドを使用すると、表またはクラスタ内のすべての行を迅速に削除できます。TRUNCATE コマンドは、次の理由から DELETE コマンドより短時間に削除が実行されます。

- TRUNCATE コマンドはデータ定義言語のコマンドであり、ロールバック情報を生成しない。
- 表を切り捨て (TRUNCATE) ても、表の DELETE トリガーが起動されない。

TRUNCATE コマンドを指定すると、削除した行の領域の割当てを任意に解除できます。この場合、表の MINEXTENTS パラメータに指定した領域以外のすべての割当てが解除されます。

表の削除や再作成と比較した場合の、TRUNCATE コマンドによる行の削除の利点は次のとおりです。

- 表を削除または再作成すると、その表に依存するオブジェクトが無効になりますが、TRUNCATE では無効になりません。

- 表を削除または再作成すると、その表のオブジェクト権限を再度付与しなければなりません。TRUNCATE では不要です。
- 表を削除または再作成すると、その表の索引、整合性制約、トリガーを再作成した上で STORAGE パラメータを再指定しなければなりません。TRUNCATE では不要です。

---

**注意：** 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。

---

## 制限事項

表を切り捨てると、NEXT の値は、削除された最後のエクステントのサイズに自動的にリセットされます。

クラスタを構成する表は個別に切り捨てることはできません。これを行うには、クラスタを切り捨てるか、表の行をすべて削除するか、表を削除して再作成しなければなりません。

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には、制約を無効にしておかなければなりません。（整合性制約が自己参照型のときは、例外として表を切り捨てることができます。）

なお、TRUNCATE 文はロールバックできません。

## 例

**例 1.** 次の文は、EMP 表の行をすべて削除して、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 表の索引の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ります。

**例 2.** 次の文は、CUST クラスタ内の表の行をすべて削除しますが、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
```

上の文では、CUST の表内のすべての索引のデータも削除されます。

**例 3.** 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;
```

## 関連項目

DELETE (4-370 ページ)

DROP CLUSTER (4-381 ページ)

DROP TABLE (4-402 ページ)

## UPDATE

---

### 用途

表またはビューの実表の既存の値を変更します。

---

---

**注意：** コマンドと句の説明で**OBJ**の印が前に付いている箇所は、Oracle Object Option がデータベース・サーバーにインストールされている場合にだけ有効です。

---

---

Oracle オプティマイザに指示、つまりヒントを渡すために、UPDATE 文中でコメントを使用できます。オプティマイザは、ヒントを使用して文の実行計画を選択します。詳細は、『Oracle8 Server チューニング』を参照してください。

UPDATE キーワードに続けてパラレル・ヒントを指定すると、基礎となる走査および UPDATE 操作の両方をパラレル化できます。パラレル DML の詳細は、『Oracle8 Server チューニング』および『Oracle8 Parallel Server 概要および管理』、『Oracle8 Server 概要』を参照してください。

### 前提条件

表の値を更新するには、表が自スキーマ内にあるか、表の UPDATE 権限を持っていなければなりません。

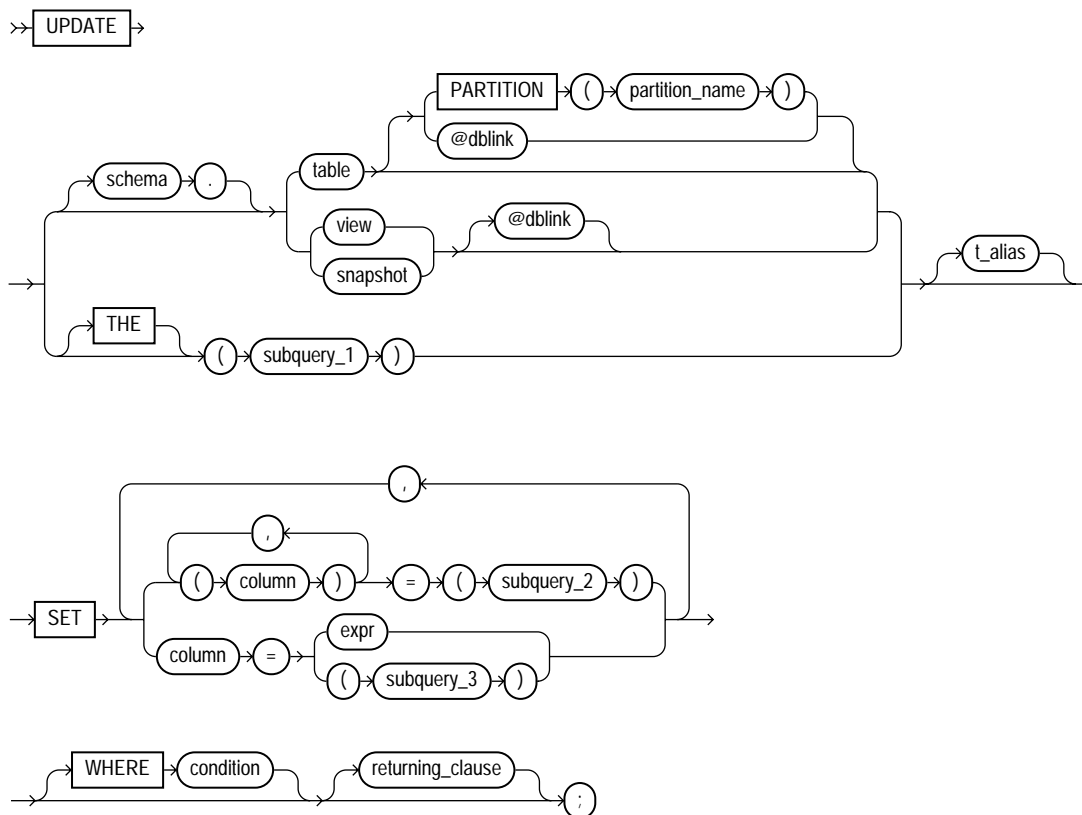
ビューの実表の値を更新するには、次のことが必要です。

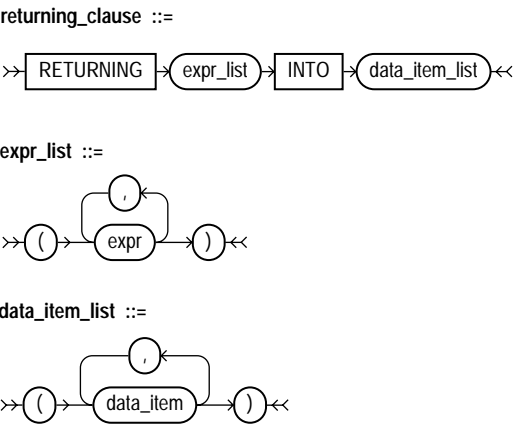
- そのビューに対する UPDATE 権限を持っている。
- そのビューが設定されているスキーマの所有者が、実表に対する UPDATE 権限を持っている。

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 句の列などの) 列値を参照している表に対して SELECT 権限を持っていなければなりません。

UPDATE ANY TABLE システム権限を持っていると、任意の表または任意の実表の値を更新できます。

## 構文





キーワードとパラメータ

|                                               |                                                                                                                                                                                                                                                         |
|-----------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>schema</i>                                 | 更新する表またはビューが定義されているスキーマです。 <i>schema</i> を指定しないと、この表が自スキーマ内にあるとみなされます。                                                                                                                                                                                  |
| <i>table / view</i>                           | 更新する表の名前です。表に対して UPDATE 文を実行すると、その表に対応付けられた UPDATE トリガーが起動します。 <i>view</i> を指定すると、Oracle はそのビューの実表を更新します。「ビューの更新」(4-539 ページ) も参照してください。                                                                                                                 |
| <b>PARTITION</b><br>( <i>partition_name</i> ) | <i>table</i> のパーティション・レベルの行更新を指定します。 <i>partition_name</i> パラメータには、更新対象の表の中のパーティションの名前または更新を 1 つのパーティションだけに限定するための、より複雑な述語を指定します。「パーティション表の更新」(4-540 ページ) も参照してください。                                                                                    |
| <i>dblink</i>                                 | 表またはビューが定義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です。データベース・リンクの参照方法の詳細は、「リモート・データベース内のオブジェクトを参照」(2-50 ページ) を参照してください。分散機能付きの Oracle 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。<br><br><i>dblink</i> を指定しないと、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。 |
| <b>◎</b> <b>THE</b>                           | 副問合せが戻す列値がスカラー値でなく、ネストした表であることを示します。接頭辞 <b>THE</b> の付いた副問合せはフラット化した副問合せと呼ばれます。「フラット化した副問合せの使用」(4-528 ページ) も参照してください。                                                                                                                                    |
| <i>subquery_1</i>                             | ビューと同様に扱われる副問合せです。「副問合せ」(4-525 ページ) も参照してください。                                                                                                                                                                                                          |
| <i>t_alias</i>                                | 文内で参照する表またはビュー、副問合せの別名です。                                                                                                                                                                                                                               |
| SET <i>clause</i>                             | SET 句には更新する列と、その列に格納する新たな値を指定します。                                                                                                                                                                                                                       |



---

|                   |                                                                                     |
|-------------------|-------------------------------------------------------------------------------------|
| <i>column</i>     | 更新する表またはビューの列の名前です。SET 句に表の列を指定しないと、その列の値は変更されません。                                  |
| <i>subquery_2</i> | 対応する複数の列に代入される、複数の新しい値を戻す副問合せです。「副問合せ」(4-525 ページ) も参照してください。                        |
| <i>expr</i>       | 対応する列に代入される新たな値です。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。構文の説明については、「式」(3-73 ページ) を参照してください。 |
| <i>subquery_3</i> | 対応する複数の列に代入される、複数の新しい値を戻す副問合せです。「副問合せ」(4-525 ページ) および「関連更新」(4-541 ページ) も参照してください。   |

SET 句に副問合せが定義されている場合に、副問合せは更新される行ごとに 1 行ずつ戻さなければなりません。副問合せで戻された各値は、カッコで囲まれたリスト内の列にそれぞれ代入されます。副問合せが行で戻さなかった場合は、列には NULL が割り当てられます。副問合せでは、更新中の表から値を選択することもあります。

SET 句では、式と副問合せの割当てを混合して指定できます。

|                         |                                                                                                                                                                                                                            |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| WHERE                   | 指定した条件が TRUE の行だけ更新します。この句を指定しないと、表またはビューの行はすべて更新されます。構文の説明は、「条件」(3-84 ページ) を参照してください。<br><br>WHERE 句には更新する行を指定します。WHERE 句を指定しないと、すべての行が更新されます。WHERE 句の条件を満たす各行ごとに、SET 句の等号(=)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ときに評価されます。 |
| <i>returning_clause</i> | UPDATE 文に影響される行を取り出します。スカラー、LOB 型、ROWID 型、REF 型だけが取り出されます。「RETURNING 句」(4-542 ページ) も参照してください。                                                                                                                              |
| <i>expr_list</i>        | 「式」(3-73 ページ) で説明した構文です。 <i>data_item_list</i> の各変数に対する列の式を <i>expr_list</i> に指定してください。                                                                                                                                    |
| INTO                    | 変更された行の値を、 <i>data_item_list</i> に指定された <i>data_item</i> 変数に格納することを示します。                                                                                                                                                   |
| <i>data_item</i>        | 取り出された <i>expr</i> の値を <i>expr_list</i> に格納する PL/SQL 変数またはバインド変数です。                                                                                                                                                        |

パラレル DML やリモート・オブジェクトでは *returning\_clause* を使用できません。

---

## ビューの更新

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だけ、ビューを更新できます。

ビューを定義する問合せに次のいずれかの要素が含まれる場合は、そのビューは更新できません。

- 集合演算子
- GROUP BY 句
- グループ関数
- DISTINCT 演算子
- フラット化した副問合せ

- ネストした表の列
- CAST 式および MULTISSET 式

## パーティション表の更新

パーティション表を作成するときは、列の順序付きリストを指定します。このリストは、行または索引エントリが入るパーティションの判別に使用されます。これらの列をパーティション列といいます。行のパーティション列の値は、その行のパーティション・キーです。

```
CREATE TABLE emp
(emp_no NUMBER(5),
 dept VARCHAR2(2),
 name VARCHAR2 (30))
STORAGE (INITIAL 100K NEXT 50K) LOGGING
PARTITION BY RANGE (emp_no)
( PARTITION acct VALUES LESS THAN (1000)
  TABLESPACE ts1,
  PARTITION sales VALUES LESS THAN (2000)
  TABLESPACE ts2
  PARTITION educ VALUES LESS THAN (3000) );
```

```
INSERT INTO EMP VALUES (1226, 'sa', 'smith');
```

```
INSERT INTO EMP VALUES (2100, 'ed', 'jones');
```

次の例では、EMP 表の従業員 SMITH の項目が更新されます。

```
UPDATE emp SET emp_no = 1356
WHERE name = 'SMITH';
```

行を更新すると、JONES が他のパーティションに移動するため、次の文は失敗します。

```
UPDATE emp SET emp_no = 1500
WHERE name = 'JONES';
```

パーティション・キーの一部となっている 1 つ以上の列の値を変更しようとする、更新した行が他のパーティションに移行してしまうため、エラーが発生します。

### 1 つのパーティションの更新

パーティション表の値を更新するときは、パーティション名を指定する必要はありませんが、パーティション名を指定した方が複雑な WHERE 句より効率が上がる場合があります。パーティション表の 1 つのパーティションの値を変更するときは、PARTITION 句を指定します。この構文を使用すると、WHERE 句を使用するより処理が簡単になることがあります。

**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);
```

## 相関更新

副問合せが更新された表の列を参照する場合、副問合せの評価は、更新処理全体について1度行われるのではなく、各行に1度ずつ行われます。このような更新を、相関更新といいます。更新された表の列の参照は、通常表の別名に基づいて行われます。

UPDATE 文が評価する各行は、相関副問合せで指定された異なる値で更新される可能性があります。ただし、通常の UPDATE 文は、同じ値で各行を更新します。

**例 1.** 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';
```

**例 2.** 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 人しかいないものとします)。

```
UPDATE emp
  SET job = 'MANAGER', sal = sal + 1000, deptno = 20
  WHERE ename = 'JONES';
```

**例 3.** 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・データベースの **ACCOUNTS** 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;
```

**例 4.** この例は、UPDATE コマンドの次の構文要素を示します。

- 単一文にまとめた SET 句の 2 つの形式
- 相関副問合せ
- 更新対象とする行を制限する WHERE 句

```
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だけを更新します。
- 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。

**例 5.** **OBJ** 次の例は、PROJS 表の特定の行を更新します。

```
UPDATE THE(SELECT projs
FROM dept d WHERE d.dno = 123) p
SET p.budgets = p.budgets + 1
WHERE p.pno IN (123, 456);
```

## RETURNING 句

RETURNING 句を使用して更新された列から値を戻すことができるので、UPDATE 文の後で SELECT を実行する必要がなくなります。

- 1 つの行を更新する RETURNING 句を持つ UPDATE 文は、行、ROWID、REF の更新された列を使用して、更新された行に列の式を取り出し、PL/SQL 変数またはバインド変数に格納します。
- RETURNING 句を持つ UPDATE 文を使用して複数の行を更新する場合、更新された行の式の値、ROWID、REF はバインド配列に格納されます。

1 つの実表を持つビューから更新する場合にも、RETURNING 句を持つ UPDATE 文を使用できます。

**例.** 次の例では、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;
```

## 関連項目

DELETE (4-370 ページ)

INSERT (4-449 ページ)

構文図とは、SQL の有効な構文を図示したものです。構文図は、矢印が示す方向に左から右へたどってください。

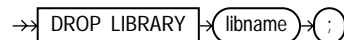
コマンドやキーワードは、四角形の中に大文字で書かれています。コマンドやキーワードは、四角形の中に書かれているとおりに指定してください。パラメータは、楕円形の中に小文字で書かれています。パラメータには変数を指定します。演算子、デリミタ、終了記号は円の中に書かれています。

構文図の中に経路が複数ある場合は、ユーザーがとる経路を選択できることを示します。

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記載されています。

## 必須キーワードとパラメータ

必須キーワードとパラメータは、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、主経路、つまり現在選択している経路の線上に書かれています。次の例では、*cursor* が必須パラメータです。



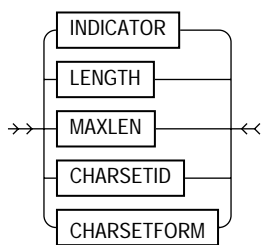
HQ\_LIB という名前のライブラリがあるとすると、この図は次の文を示しています。

```
DROP LIBRARY hq_lib;
```

主経路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

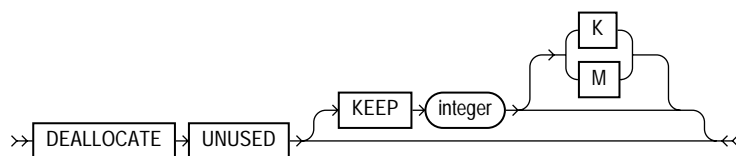
---

つを選択しなければなりません。ただし、主経路上にあるものでなくても構いません。次の例では、5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

キーワードやパラメータが主経路より上に縦に並べてリストされている場合は、オプションのキーワードやパラメータを示しています。次の例では、縦方向の経路の方へ行ってもよいし、続けて主経路を選択してもかまいません。

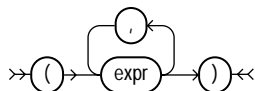


この図によると、次の文はすべて有効です。

```
DEALLOCATE UNUSED;  
DEALLOCATE UNUSED KEEP 1000;  
DEALLOCATE UNUSED KEEP 10 M;
```

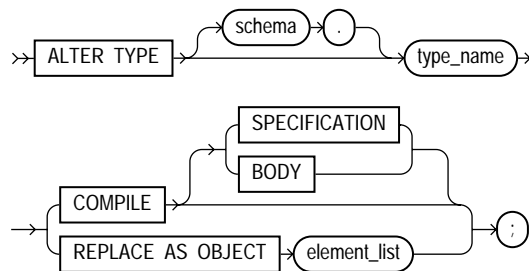
## 構文のループ

ループがある場合、ループ内の構文を何度でも繰り返し実行できます。次の例では、式を 1 つ選択した後、別の式の選択に戻ることを繰り返すことができます。式と式の間はコンマで区切ります。



## 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、すべての主経路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つの部分に分かれた構文図です。



この図によると、次の文は有効です。

```
ALTER TYPE type_name COMPILE BODY;
```

## 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 文字以内に限定されています。先頭の文字は英字でなければなりません、それ以外は、英字、数字、ドル記号 (\$)、ポンド記号 (#)、アンダースコア (\_) を任意に組み合わせて指定できます。

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) 組み合わせて指定できます。

Oracle の識別子には、二重引用符で囲んだとき以外は 大 / 小文字の区別がありません。





---

## Oracle と標準 SQL

この付録では、次の内容について説明します。

- 業界標準規格を管理する団体によって確立される SQL 規格への Oracle の規格準拠
- 標準 SQL への Oracle の拡張機能
- FIPS フラガーによる標準 SQL への拡張機能の位置付け

### 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 の規格準拠性を示します。

- 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

これらの規格との準拠性は、米国連邦情報・技術局 (NIST) の "SQL Test Suite" によって評価されます。NIST はアメリカ合衆国の政府機関です。

### ANSI と ISO の規格準拠

Oracle8 は、ANSI 文書 X3.135-1992 "Database Language SQL" で定義された規格準拠性に基本レベルで適合しています。ANSI 規格のコピーを入手するには、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 と ISO の SQL 規格では、規格準拠を明示した箇所において、準拠の種類とインプリメントされている機能について記載することを義務づけています。Oracle8 Server、および Oracle プリコンパイラ (Fortran 用) バージョン 1.8.25、Oracle プリコンパイラ (C/C++ 用) バージョン 8.0.4、Oracle プリコンパイラ (Cobol 用) バージョン 8.0.4、SQL\*Module(ADA 用) バージョン 8.0.4 は、次のような ANSI X3.135-1992/ISO 9075-1992 規格に準拠しています。

- エントリ・レベルでの規格準拠 (SQL-DDL と SQL-DML を含む)
- ADA 用モジュール言語
- SQL 埋込み C
- SQL 埋込み COBOL
- SQL 埋込み FORTRAN

## 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" のために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、Fortran 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せん。

### Section 16.3 Style of Language Interface( 言語インタフェースのスタイル )

SQL\*Module の付いた Oracle では、ADA 用のモジュール言語をサポートします。Oracle プリコンパイラの付いた Oracle では、C、COBOL、FORTRAN 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 は、SQL\*Plus バージョン 3.1( および他の Oracle Tools )によって、次の SQL コマンドの「直接的な呼出し」をサポートして、FIPS PUB 127-2 の要件を満たしています。

- CREATE TABLE コマンド
- CREATE VIEW コマンド
- GRANT コマンド
- INSERT コマンド
- SELECT コマンド (ORDER BY 句は付くが、INTO 句は付かない)
- UPDATE コマンド (検索)

- DELETE コマンド (検索)
- COMMIT WORK コマンド
- ROLLBACK WORK コマンド

このマニュアルで説明した他の SQL コマンドのほとんどは、対話型で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-1 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を Oracle8 ではどのように満たしているかを示します。

表 B-1 データベース構成体のサイズ設定

| データベース構成体                       | FIPS  | Oracle8            |
|---------------------------------|-------|--------------------|
| 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 <sup>(a)</sup> | 20    | 1000               |
| 行の長さ <sup>(b、c)</sup>           | 2,000 | 2,000,000          |
| UNIQUE 制約の中の列                   | 6     | 16                 |
| UNIQUE 制約の長さ <sup>(b)</sup>     | 120   | (d)                |
| 外部キーリストの長さ <sup>(b)</sup>       | 120   | (d)                |
| GROUP BY 句の中の列                  | 6     | 255 <sup>(e)</sup> |
| GROUP BY 列のリストの長さ               | 120   | (e)                |

表 B-1 データベース構成体のサイズ設定

|                    |     |                    |
|--------------------|-----|--------------------|
| ORDER BY 句の中のソート指定 | 6   | 255 <sup>(e)</sup> |
| ORDER BY 列のリストの長さ  | 120 | (e)                |
| 参照整合性制約内の列         | 6   | 16                 |
| SQL 文で参照される表       | 15  | 無制限                |
| 同時にオープン可能なカーソル     | 10  | (f)                |
| SELECT 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コ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に示す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 値を含んでいる行の最大長に基づいています。つまり、 $2(254) + 231 + (999(4000))$  です。

(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となります。

(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。しかし、GROUP BY 句や ORDER BY 句内のすべての式のサイズの和は、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) 同時にオープンされ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り異なりますが、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、そして 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 に対する拡張機能

この項では、標準 SQL「データベース言語 SQL」の範囲外で、Oracle がサポートする追加機能を示します。また、SQL 言語の次の部分に関する情報を提供します。

- ・ コマンド
- ・ 関数
- ・ 演算子
- ・ 疑似列

- データ型
- スキーマ・オブジェクトの名前
- 値

Oracle プリコンパイラでサポートされている標準埋込み SQL" データベース言語埋込み SQL" の拡張機能の詳細は、『Pro\*COBOL プリコンパイラ・プログラマーズ・ガイド』および『Pro\*C/C++ プリコンパイラ・プログラマーズ・ガイド』、『SQL\*Module for Ada Programmer's Guide』を参照してください。

## コマンド

この項では、これらの追加コマンド、および標準コマンドの追加の構文と機能について説明します。Oracle では、Entry SQL92 には含まれない次のコマンドがサポートされています。

|                         |                       |
|-------------------------|-----------------------|
| ALTER CLUSTER           | CREATE SEQUENCE       |
| ALTER DATABASE          | CREATE SNAPSHOT       |
| ALTER FUNCTION          | CREATE SNAPSHOT LOG   |
| ALTER INDEX             | CREATE SYNONYM        |
| ALTER PACKAGE           | CREATE TABLE          |
| ALTER PROCEDURE         | CREATE TABLESPACE     |
| ALTER PROFILE           | CREATE TRIGGER        |
| ALTER RESOURCE COST     | CREATE TYPE           |
| ALTER ROLLBACK SEGMENT  | CREATE TYPE BODY      |
| ALTER ROLE              | CREATE USER           |
| ALTER SEQUENCE          | CREATE VIEW           |
| ALTER SESSION           |                       |
| ALTER SNAPSHOT          | DROP CLUSTER          |
| ALTER SNAPSHOT LOG      | DROP DATABASE LINK    |
| ALTER SYSTEM            | DROP DIRECTORY        |
| ALTER TABLE             | DROP FUNCTION         |
| ALTER TABLESPACE        | DROP INDEX            |
| ALTER TRIGGER           | DROP LIBRARY          |
| ALTER TYPE              | DROP PACKAGE          |
| ALTER USER              | DROP PROCEDURE        |
| ALTER VIEW              | DROP PROFILE          |
| ANALYZE                 | DROP ROLLBACK SEGMENT |
| AUDIT                   | DROP ROLE             |
|                         | DROP SEQUENCE         |
| COMMENT                 | DROP SNAPSHOT         |
| CREATE CONTROLFILE      | DROP SNAPSHOT LOG     |
| CREATE CLUSTER          | DROP SYNONYM          |
| CREATE DATABASE         | DROP TABLE            |
| CREATE DATABASE LINK    | DROP TABLESPACE       |
| CREATE DIRECTORY        | DROP TYPE             |
| CREATE FUNCTION         | DROP TYPE BODY        |
| CREATE INDEX            |                       |
| CREATE LIBRARY          | EXPLAIN PLAN          |
| CREATE PACKAGE          |                       |
| CREATE PACKAGE BODY     | NOAUDIT               |
| CREATE PROCEDURE        |                       |
| CREATE PROFILE          | RENAME                |
| CREATE ROLLBACK SEGMENT | REVOKE                |
| CREATE ROLE             |                       |
|                         | SAVEPOINT             |
|                         | SET CONSTRAINT        |
|                         | SET TRANSACTION       |
|                         |                       |
|                         | TRUNCATE              |

---

## 標準コマンドの追加要素

Oracle では、Entry SQL92 の一部であるいくつかのコマンドに対して追加の構文がサポートされています。

## COMMIT

COMMIT コマンドでサポートされている追加の句は次のとおりです。

- COMMENT 句
- FORCE 句

また、Entry SQL92 では、COMMIT 文には **WORK** キーワードを指定する必要があります。Oracle では、COMMIT 文にこのキーワードを指定するかどうかは任意です。なお、このキーワードを指定しても、コマンドに新しい機能は追加されません。

## CREATE TABLE

CREATE TABLE コマンドでサポートされている追加のパラメータと句は次のとおりです。

- AS 句
- ENABLE 句
- DISABLE 句
- CLUSTER 句
- INITRANS パラメータ
- MAXTRANS パラメータ
- ORGANIZATION 句
- PCTFREE パラメータ
- PCTTHRESHOLD パラメータ
- PCTUSED パラメータ
- STORAGE 句
- TABLESPACE パラメータ

**CONSTRAINT 句** CREATE TABLE コマンドの CONSTRAINT 句でサポートされている追加のオプションと識別子は次のとおりです。

- ON DELETE CASCADE オプション
- ENABLE オプション
- DISABLE オプション
- CONSTRAINT 識別子

CREATE TABLE コマンドの**列定義**では、次の句もサポートされています。

- WITH ROWID
- SCOPE

さらに、列の定義には、Entry SQL92 データ型だけでなく Oracle の事前定義済みの型も使用できます。Oracle の拡張データ型は、次に示します。列のデータ型が BLOB または CLOB、NCLOB の場合、特別な LOB 記憶領域と索引に関する機能を CREATE TABLE コマンドで指定できます。

## CREATE VIEW

CREATE VIEW コマンドでサポートされている追加の構文は次のとおりです。

- OR REPLACE オプション
- FORCE オプションおよび NOFORCE オプション
- WITH CHECK OPTION 付きの CONSTRAINT 識別子

CREATE VIEW 文で列の名前を省略すると、ビューを定義している問合せに指定された列の別名が、ビューの列に使用されます。

## DELETE

DELETE コマンドでサポートされている追加の構文は次のとおりです。

- リモート・データベース上の表とビューから行を削除するためのデータベース・リンク
- 相関問合せで使用するための表の別名
- PARTITION 句
- RETURNING 句

また、Entry SQL92 では、DELETE 文には FROM キーワードを指定する必要があります。Oracle では、DELETE 文にこのキーワードを指定するかどうかは任意です。なお、このキーワードを指定しても、コマンドに新しい機能は追加されません。

変更可能な結合ビューに結合しているキー保存表がただ 1 つの場合、Oracle ではこのビューに DELETE コマンドを発行できます。SQL92 では結合ビューに対して DELETE コマンドは使用できません。

## GRANT

GRANT コマンド（システム権限とロール）は、標準 SQL の拡張機能です。

GRANT コマンド（オブジェクト権限）は、Entry SQL92 がサポートする表およびビューについての DELETE および INSERT、REFERENCES、SELECT、UPDATE の各権限に加えて、次に示す他のオブジェクトについての他の権限もサポートされています。



- ALTER
- EXECUTE
- INDEX
- READ

また、このコマンドではロールへのオブジェクト権限の付与もサポートされています。

## INSERT

INSERT コマンドでは、リモート・データベース上の表およびビューに行を挿入するために、データベース・リンクの使用がサポートされています。INSERT コマンドでサポートされている追加の構文は次のとおりです。

- PARTITION 句
- RETURNING 句

INSERT コマンドでは、ビューへの挿入と同様に、INTO 句内の副問合せの使用もサポートされています。

INSERT コマンドでは WITH CHECK OPTION が指定されていない変更可能な結合ビューへの挿入が可能です。ただし、挿入対象の列がすべて該当する結合の同一のキー保存表に存在していることが必要です。

## ROLLBACK

ROLLBACK コマンドでサポートされている追加の句は次のとおりです。

- TO 句
- FORCE 句

また、Entry SQL92 では、ROLLBACK 文には WORK キーワードを指定する必要があります。Oracle では、ROLLBACK 文にこのキーワードを指定するかどうかは任意です。なお、このキーワードを指定しても、コマンドに新しい機能は追加されません。

## SELECT

SELECT コマンドでサポートされている追加の句と構文は次のとおりです。

- START WITH 句
- CONNECT BY 句
- FOR UPDATE 句
- リモート・データベース上の表およびビュー、スナップショットを問い合わせるためのデータベース・リンク

- 外部結合を行うための外部結合演算子 (+)
- 選択のリスト内の NULL

**GROUP BY 句** SELECT コマンドの GROUP BY 句でサポートされている追加の構文および機能は次のとおりです。

- ビューを定義している問合せにグループ関数または GROUP BY 句が指定されている場合、そのビューから選択する SELECT 文には、グループ関数および GROUP BY 句、HAVING 句、WHERE 句を指定できる。
- ビューを定義している問合せに GROUP BY 句が指定されている場合、SELECT 文はこのビューに関連する結合を実行できる。

**ORDER BY 句** SELECT コマンドの ORDER BY 句でサポートされている追加の構文および機能は次のとおりです。

- 選択リストの名前、別名、列の位置番号だけでなく、FROM 句に指定されている表またはビューの任意の列に関係する式を指定できる。
- *table.column* または *view.column* という構文を使用して、表名またはビュー名で列名を修飾できる。

**副問合せ** 副問合せ（つまり、他の SQL 文中に指定する SELECT コマンドの形式）でサポートされている追加機能は次のとおりです。

- GROUP BY 句を指定できる。
- 定義の問合せに GROUP BY 句が含まれるビューからも、選択できる。

## UPDATE

UPDATE コマンドでサポートされている追加の構文は次のとおりです。

- リモート・データベース上の表およびビューのデータを更新するためのデータベース・リンク
- 相関副問合せで使用するための表の別名
- 単一の列だけでなく、SET 句の左側に指定するカッコで囲んだ列のリスト
- 式だけでなく、SET 句の右側に指定する副問合せ
- PARTITION 句
- RETURNING 句

また、UPDATE コマンドでサポートされている追加機能は次のとおりです。

- UPDATE 文では、変更可能な結合ビューを更新できる。ただし、更新対象の列がすべて、該当する結合の同一のキー保存表に存在している必要があります。ビューが WITH CHECK OPTION を指定した場合、結合列は修正できません。

- UPDATE 文の SET 句または WHERE 句内の副問合せは、更新している表またはビューで参照できる。
- 複合式 (FROM 句に表の列だけでなく、関数や演算子も含まれている) として定義される列を含むビューを更新できる。
- UPDATE コマンドでは、副問合せの更新がサポートされている。

## 関数

この項では、追加された関数、および標準関数に追加された機能について説明します。

### 追加された関数

Entry SQL92 の関数は、AVG および COUNT、MAX、MIN、SUM の 5 つだけです。Oracle では、Entry SQL92 には含まれない追加機能が多くサポートされています。「SQL 関数」(3-15 ページ) を参照してください。

### 標準関数に追加された機能

次の例に示されるように、SELECT 文の選択リストでグループ関数をネストできます。

```
SELECT MIN(MAX(sal))  
FROM emp  
GROUP BY deptno;
```

ネストの深さは、例で示されるよりも深くすることはできません。

また、ビューが定義されている問合せにグループ関数または GROUP BY 句が含まれている場合も、そのビューを問い合わせる SELECT 文でグループ関数を使用できます。

## 演算子

この項では、追加された演算子、および標準演算子に追加された機能について説明します。

### 追加された演算子

Oracle では、Entry SQL92 には含まれない次の演算子がサポートされています。

- || 文字演算子 (文字連結)
- != および <math>\neq</math>、<math>\neq</math> 比較演算子 (非等価性)
- MINUS 集合演算子
- INTERSECT 集合演算子
- (+) 演算子 (外部結合)
- PRIOR 演算子

## 標準演算子の追加機能

Oracle では、Entry SQL92 の演算子に対して次の追加機能がサポートされています。

- IN 演算子を含む式の左のメンバーは、単一の式だけではなく、カッコで囲まれた式のリストでもかまわない。
- 列だけでなく、どのような式でも比較演算子 IS NULL および IS NOT NULL で使用できる。
- LIKE 演算子で使用するパターンは、テキスト・リテラルだけでなく、データ型 CHAR または VARCHAR2 の式でもかまわない。

## 疑似列

疑似列は、表の列と同じような特性を持ちますが、実際には表に格納されていない値です。疑似列は、Oracle によってサポートされますが、Entry SQL92 の一部ではありません。疑似列のリストについては、「疑似列」(2-30 ページ)を参照してください。

## データ型

Oracle では、Entry SQL92 には含まれない次の追加のデータ型がサポートされています。

- DATE( 注意 : Oracle の DATE データ型は、Intermediate SQL92 の DATE データ型とは異なります。 )
- NUMBER
- VARCHAR2
- LONG
- RAW
- LONG RAW
- ROWID
- BLOB
- CLOB
- BFILE
- NCLOB

さらに、Oracle は次に示す Entry SQL92 には含まれないユーザー定義型もサポートします。

- オブジェクト型
- REF 型
- コレクション型 (VARRAY およびネストした表)

Oracle では、Entry SQL92 には含まれないデータ型間で値を相互に自動変換できます。

## スキーマ・オブジェクトの名前

Oracle では、スキーマ・オブジェクトの名前に対して、次の追加機能がサポートされています。

- 最大長が 18 バイトでなく 30 バイトの名前。および、特殊記号 # と \$、および繰返しアンダースコア (\_) を含む名前。

## 値

Oracle では、数値の指数表記に、大文字の "E" だけでなく小文字の "e" を使用しても構いません。

## FIPS フラガー

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、この付録の各項に記載した拡張部分を使用できます。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、埋込み SQL プログラムの中で Oracle の拡張部分を Entry SQL92 に位置付けるために、Oracle の FIPS フラガーを使用してください。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。FIPS フラガーの使用の詳細は、『Pro\*COBOL プリコンパイラ・プログラマーズ・ガイド』および『Pro\*C/C++ プリコンパイラ・プログラマーズ・ガイド』、『SQL\*Module for Ada Programmer's Guide』を参照してください。



# Oracle の予約語とキーワード

この付録では、Oracle の予約語とキーワードをまとめます。

表 C-1 の「予約語」は、Oracle の予約語をリストにしたものです。アスタリスク (\*) の付いた語は 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。ネーム変換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 予約語

|        |            |             |            |
|--------|------------|-------------|------------|
| ACCESS | AUDIT      | COMPRESS    | DESC*      |
| ADD*   | BETWEEN*   | CONNECT*    | DISTINCT*  |
| ALL*   | BY*        | CREATE*     | DROP*      |
| ALTER* | CHAR*      | CURRENT*    | ELSE*      |
| AND*   | CHECK*     | DATE*       | EXCLUSIVE  |
| ANY*   | CLUSTER    | DECIMAL*    | EXISTS     |
| AS*    | COLUMN     | DEFAULT*    | FILE       |
| ASC*   | COMMENT    | DELETE*     | FLOAT*     |
| FOR*   | LONG       | PCTFREE     | SUCCESSFUL |
| FROM*  | MAXEXTENTS | PRIOR*      | SYNONYM    |
| GRANT* | MINUS      | PRIVILEGES* | SYSDATE    |

表 C-1 予約語

|            |            |           |           |
|------------|------------|-----------|-----------|
| GROUP*     | MODE       | PUBLIC*   | TABLE*    |
| HAVING*    | MODIFY     | RAW       | THEN*     |
| IDENTIFIED | NETWORK    | RENAME    | TO*       |
| IMMEDIATE* | NOAUDIT    | RESOURCE  | TRIGGER   |
| IN*        | NOCOMPRESS | REVOKE*   | UID       |
| INCREMENT  | NOT*       | ROW       | UNION*    |
| INDEX      | NOWAIT     | ROWID     | UNIQUE*   |
| INITIAL    | NULL*      | ROWNUM    | UPDATE*   |
| INSERT*    | NUMBER     | ROWS*     | USER*     |
| INTEGER*   | OF*        | SELECT*   | VALIDATE  |
| INTERSECT* | OFFLINE    | SESSION*  | VALUES*   |
| INTO*      | ON*        | SET*      | VARCHAR*  |
| IS*        | ONLINE     | SHARE     | VARCHAR2  |
| LEVEL*     | OPTION*    | SIZE*     | VIEW*     |
| LIKE*      | OR*        | SMALLINT* | WHENEVER* |
| LOCK       | ORDER*     | START     | WHERE     |
|            |            |           | WITH*     |



表 C-2 の「キーワード」は、Oracle のキーワードをリストにしたものです。アスタリスク(\*)の付いたキーワードは ANSI の予約語でもあります。他の SQL 処理系への移植性を最大限にするために、次のワードはスキーマ・オブジェクト名として使用しないでください。

表 C-2 キーワード

|               |                     |                 |                  |
|---------------|---------------------|-----------------|------------------|
| ACCOUNT       | CACHE_INSTANCES     | CONSTRAINT*     | DEGREE           |
| ACTIVATE      | CANCEL              | CONSTRAINTS*    | DEREF            |
| ADMIN         | CASCADE*            | CONTENTS        | DIRECTORY        |
| AFTER         | CAST                | CONTINUE*       | DISABLE          |
| ALLOCATE*     | CFILE               | CONTROLFILE     | DISCONNECT       |
| ALL_ROWS      | CHAINED             | CONVERT*        | DISMOUNT         |
| ANALYZE       | CHANGE              | COST            | DISTRIBUTED      |
| ARCHIVE       | CHARACTER*          | COUNT*          | DML              |
| ARCHIVELOG    | CHAR_CS             | CPU_PER_CALL    | DOUBLE*          |
| ARRAY         | CHECKPOINT          | CPU_PER_SESSION | DUMP             |
| AT*           | CHOOSE              | CURRENT_SCHEMA  |                  |
| AUTHENTICATED | CHUNK               | CURRENT_USER*   | EACH             |
| AUTHORIZATION | CLEAR               | CURSOR*         | ENABLE           |
| AUTOEXTEND    | CLOB                | CYCLE           | END*             |
| AUTOMATIC     | CLONE               |                 | ENFORCE          |
|               | CLOSE*              | DANGLING        | ENTRY            |
| BACKUP        | CLOSED_CACHED_OPEN_ | DATABASE        | ESCAPE*          |
|               | CURSORS             |                 |                  |
| BECOME        | COALESCE*           | DATAFILE        | ESTIMATE         |
| BEFORE        | COLUMNS             | DATAFILES       | EVENTS           |
| BEGIN*        | COMMIT*             | DATAOBJNO       | EXEMPT*          |
| BFILE         | COMMITTED           | DBA             | EXCEPTIONS       |
| BITMAP        | COMPATIBILITY       | DEALLOCATE*     | EXCHANGE         |
| BLOB          | COMPILE             | DEBUG           | EXCLUDING        |
| BLOCK         | COMPLETE            | DEC*            | EXECUTE*         |
| BODY          | COMPOSITE_LIMIT     | DECLARE*        | EXPIRE           |
|               | COMPUTE             | DEFERRABLE      | EXPLAIN          |
| CACHE         | CONNECT_TIME        | DEFERRED        | EXTENT           |
| EXTENTS       | INDEXES             | MANAGE          | NLS_CALENDAR     |
| EXTERNALLY    | INDICATOR*          | MASTER          | NLS_CHARACTERSET |

表 C-2 キーワード

|               |                    |                 |                   |
|---------------|--------------------|-----------------|-------------------|
|               | IND_PARTITION      | MAX*            | NLS_ISO_CURRENCY  |
| FAILED_LOGIN_ | INITIALLY          | MAXARCHLOGS     | NLS_LANGUAGE      |
| ATTEMPTS      |                    |                 |                   |
| FALSE         | INITRANS           | MAXDATAFILES    | NLS_NUMERIC_      |
|               |                    |                 | CHARACTERS        |
| FAST          | INSTANCE           | MAXINSTANCES    | NLS_SORT          |
| FIRST_ROWS    | INSTANCES          | MAXLOGFILES     | NOS_SPECIAL_CHARS |
| FLAGGER       | INSTEAD            | MAXLOGHISTORY   | NLS_TERRITORY     |
| FLUSH         | INT*               | MAXLOGMEMBERS   | NOARCHIVELOG      |
| FORCE         | INTERMEDIATE       | MAXSIZE         | NOCACHE           |
| FOREIGN*      | ISOLATION*         | MAXTRANS        | NOCYCLE           |
| FREELIST      | ISOLATION_LEVEL    | MAXVALUE        | NOFORCE           |
| FREELISTS     |                    | MEMBER          | NOLOGGING         |
| FULL          | KEEP               | MIN*            | NOMAXVALUE        |
| FUNCTION      | KEY*               | MINEXTENTS      | NOMINVALUE        |
|               | KILL               | MINIMUM         | NONE              |
| GLOBAL*       |                    | MINVALUE        | NOORDER           |
| GLOBALLY      | LAYER              | MOUNT           | NOOVERRIDE        |
| GLOBAL_NAME   | LESS               | MOVE            | NOPARALLEL        |
| GROUPS        | LIBRARY            | MTS_DISPATCHERS | NORESETLOGS       |
| HASH          | LIMIT              | MULTISET        | NOREVERSE         |
| HASHKEYS      | LINK               |                 | NORMAL            |
| HEADER        | LIST               | NATIONAL*       | NOSORT            |
| INSTANCE      | LOB                | NCHAR*          | NOTHING           |
| HEAP          | LOCAL*             | NCHAR_CS        | NUMERIC*          |
|               | LOG                | NCLOB           | NVARCHAR2         |
| IDLE_TIME     | LOGFILE            | NEEDED          | OBJECT            |
| IF            | LOGGING            | NESTED          | OBJNO             |
| INCLUDING     | LOGICAL_READS_PER_ | NEW             | OBJNO_REUSE       |
|               | CALL               |                 |                   |
| INDEXED       | LOGICAL_READS_PER_ | NEXT*           | OFF               |
|               | SESSION            |                 |                   |
| OID           | PLSQL_DEBUG        | RESIZE          | SHARED_POOL       |

表 C-2 キーワード

|                              |                  |                            |                           |
|------------------------------|------------------|----------------------------|---------------------------|
| OIDINDEX                     | POST_TRANSACTION | RESTRICTED                 | SHRINK                    |
| OLD                          | PRECISION*       | RETURN                     | SKIM_UNUSABLE_<br>INDEXES |
| ONLY*                        | PRESERVE*        | RETURNING                  | SNAPSHOT                  |
| OPCODE                       | PRIMARY*         | REUSE                      | SOME*                     |
| OPEN*                        | PRIVATE          | REVERSE                    | SORT                      |
| OPTIMAL                      | PRIVATE_SGA      | ROLE                       | SPECIFICATION             |
| OPTIMIZER_GOAL               | PRIVILEGE        | ROLES                      | SPLIT                     |
| ORGANIZATION                 | PROCEDURE*       | ROLLBACK*                  | SQLCODE*                  |
| OVERFLOW                     | PROFILE          | ROWLABEL                   | SQLERROR*                 |
| OWN                          | PURGE            | RULE                       | SQL_TRACE                 |
|                              |                  |                            | STANDBY                   |
| PACKAGE                      | QUEUE            | SAMPLE                     | STATEMENT_ID              |
| PARALLEL                     | QUOTA            | SAVEPOINT                  | STATISTICS                |
| PARTITION                    |                  | SCAN_INSTANCES             | STOP                      |
| PASSWORD                     | RANGE            | SCHEMA*                    | STORAGE                   |
| PASSWORD_GRACE_<br>TIME      | RBA              | SCN                        | STORE                     |
| PASSWORD_LIFE_TIME           | READ*            | SCOPE                      | STRUCTURE                 |
| PASSWORD_LOCK_<br>TIME       | REAL*            | SD_ALL                     | SUM*                      |
| PASSWORD_REUSE_<br>MAX       | REBUILD          | SD_INHIBIT                 | SWITCH                    |
| PASSWORD_REUSE_<br>TIME      | RECOVER          | SD_SHOW                    | SYSDBA                    |
| PASSWORD_VERIFY_<br>FUNCTION | RECOVERABLE      | SEGMENT                    | SYSOPER                   |
| PCTINCREASE                  | RECOVERY         | SEG_BLOCK                  | SYSTEM                    |
| PCTTHRESHOLD                 | REF              | SEG_FILE                   |                           |
| PCTUSED                      | REFERENCES*      | SEQUENCE                   | TABLES                    |
| PCTVERSION                   | REFERENCING      | SERIALIZABLE               | TABLESPACE                |
| PERCENT                      | REFRESH          | SESSIONS_PER_USER          | TABLESPACE_NO             |
| PERMANENT                    | REPLACE          | SESSION_CACHED_<br>CURSORS | TABNO                     |

表 C-2 キーワード

|              |            |               |            |
|--------------|------------|---------------|------------|
| PLAN         | RESET      | SHARED        | TEMPORARY* |
|              | RESETLOGS  |               |            |
| THAN         | TRUE       | UNLOCK        | VALIDATION |
| THE          | TRUNCATE   | UNRECOVERABLE | VALUE      |
| THREAD       | TX         | UNTIL*        | VARRAY     |
| TIME         | TYPE       | UNUSABLE      | VARYING*   |
| TIMESTAMP    |            | UNUSED        | WHEN       |
| TOplevel     | UBA        | UPDATABLE     | WITHOUT    |
| TRACE        | UNARCHIVED | USAGE*        | WORK*      |
| TRACING      | UNDER      | USE           | WRITE*     |
| TRANSACTION* | UNDO       | USING*        |            |
| TRANSITIONAL | UNLIMITED  |               | XID        |
| TRIGGERS     |            |               |            |

## 記号

---

<= 比較演算子, 3-5  
< 比較演算子, 3-5  
!= 比較演算子, 3-5  
” 二重引用符  
    オブジェクト名で, 2-45  
\$ 書式要素, 3-62  
% 文字  
    パターン・マッチング, 3-8  
(+) 外部結合演算子, 3-15  
(. (ピリオド) 数値書式要素, 3-62  
(. (カンマ) 書式要素, 3-62  
= 比較演算子, 3-5  
>= 比較演算子, 3-5  
> 比較演算子, 3-5  
^= 比較演算子, 3-5  
\_ 文字  
    パターン・マッチング, 3-8

## 数字

---

0 書式要素, 3-62  
1 つのパーティションからの削除, 4-374  
9 書式要素, 3-62

## A

---

ABS 数値関数, 3-16  
AD/A.D. 書式要素, 3-68  
ADD DATAFILE 句  
    ALTER TABLESPACE コマンドの, 4-135  
ADD LOGFILE MEMBER 句  
    ALTER DATABASE コマンドの, 4-20  
ADD LOGFILE 句

    ALTER DATABASE コマンドの, 4-19  
ADD OVERFLOW オプション  
    ALTER TABLE コマンドの, 4-116  
ADD PARTITION オプション  
    ALTER SNAPSHOT コマンドの, 4-79, 4-85  
    ALTER TABLE コマンドの, 4-120  
add\_column\_options\_clause  
    ALTER TABLE の, 4-108  
ADD\_MONTHS 日付関数, 3-35  
add\_overflow\_clause  
    ALTER TABLE の, 4-112  
add\_partition\_clause  
    ALTER TABLE の, 4-114  
ADD 句  
    ALTER SNAPSHOT LOG コマンドの, 4-86  
    ALTER TABLE コマンドの, 4-115  
ADMIN OPTION  
    GRANT コマンドの, 4-439  
ADVISE 句  
    ALTER SESSION コマンドの, 4-62  
AFTER オプション  
    CREATE TRIGGER コマンドの, 4-331  
ALL PRIVILEGES オプション  
    GRANT コマンドの, 4-443  
    REVOKE コマンドの, 4-476  
ALL TRIGGERS オプション  
    DISABLE 句の, 4-376  
    ENABLE 句の, 4-416  
ALL\_INDEXES ビュー, 4-159  
ALL\_TAB\_COLUMNS ビュー, 4-160  
ALL\_TABLES ビュー, 4-159  
ALLOCATE EXTENT 句  
    ALTER CLUSTER コマンドの, 4-12  
    ALTER TABLE コマンドの, 4-117  
allocate\_extent\_clause

- ALTER CLUSTER, 4-12
- ALTER INDEX の, 4-31
- ALTER TABLE の, 4-111
- ALL オプション
  - ARCHIVE LOG 句の, 4-165
  - SELECT コマンドの, 4-489
  - SET ROLE コマンドの, 4-511
  - SQL グループ関数の, 3-54
- ALL 比較演算子, 3-5
- ALL 文監査のショートカット, 4-173
- ALTER CLUSTER
  - deallocate\_unused\_clause 「DEALLOCATE UNUSED 句」参照
  - physical\_attributes\_clause, 4-11
- ALTER CLUSTER コマンド, 4-11
  - allocate\_extent\_clause, 4-12
  - deallocate\_unused\_clause 「DEALLOCATE UNUSED 句」参照
- ALTER DATABASE コマンド, 4-15
  - autoextend\_clause, 4-18
  - recover\_clause 「RECOVER 句」参照
 例, 4-24, 4-25, 4-468
- ALTER FUNCTION コマンド, 4-26
  - 例, 4-27
- ALTER INDEX コマンド, 4-28
  - allocate\_extent\_clause, 4-31
  - deallocate\_unused\_clause 「DEALLOCATE UNUSED 句」参照
  - index\_physical\_attributes\_clause, 4-30
  - parallel\_clause 「PARALLEL 句」参照
  - partition\_description\_clause, 4-31
  - split\_partition\_clause, 4-31
  - storage\_clause 「STORAGE 句」参照
 例, 4-35
- ALTER PACKAGE コマンド
  - 例, 4-39, 4-40
- ALTER PROCEDURE コマンド, 4-41
  - 例, 4-42
- ALTER PROFILE コマンド, 4-43
  - 例, 4-46
- ALTER RESOURCE COST コマンド, 4-48
  - 例, 4-49
- ALTER ROLE コマンド, 4-51
  - 例, 4-52
- ALTER ROLLBACK SEGMENT コマンド, 4-53
  - storage\_clause 「STORAGE 句」参照
 例, 4-55
- ALTER SEQUENCE コマンド, 4-56
  - 例, 4-57
- ALTER SESSION コマンド, 4-58
  - 例, 4-67, 4-68, 4-69, 4-70, 4-72, 4-73
- ALTER SNAPSHOT LOG コマンド, 4-84
  - add\_partition\_clause 「ALTER TABLE コマンド」参照
  - modify\_default\_attributes\_clause 「ALTER TABLE コマンド」参照
  - modify\_partition\_clause 「ALTER TABLE コマンド」参照
  - move\_partition\_clause 「ALTER TABLE コマンド」参照
  - physical\_attributes\_clause 「ALTER TABLE コマンド」参照
  - rename\_partition\_clause 「ALTER TABLE コマンド」参照
  - split\_partition\_clause 「ALTER TABLE コマンド」参照
 例, 4-86
- ALTER SNAPSHOT コマンド, 4-76
  - add\_partition\_clause 「ALTER TABLE コマンド」参照
  - LOB\_storage\_clause 「ALTER TABLE コマンド」参照
  - modify\_default\_attributes\_clause 「ALTER TABLE コマンド」参照
  - modify\_LOB\_storage\_clause 「ALTER TABLE コマンド」参照
  - modify\_partition\_clause 「ALTER TABLE コマンド」参照
  - move\_partition\_clause 「ALTER TABLE コマンド」参照
  - parallel\_clause 「PARALLEL 句」参照
  - physical\_attributes\_clause 「ALTER TABLE コマンド」参照
  - rename\_partition\_clause 「ALTER TABLE コマンド」参照
  - split\_partition\_clause 「ALTER TABLE コマンド」参照
  - storage\_clause 「STORAGE 句」参照
 例, 4-81
- ALTER SYSTEM コマンド, 4-88
  - archive\_log\_clause 「ARCHIVE LOG 句」参照
  - dispatch\_clause, 4-92
  - opts\_clause, 4-93
  - set\_clause, 4-90
 例, 4-98, 4-100, 4-102, 4-103
- ALTER TABLESPACE コマンド, 4-131

- autoextend\_clause, 4-134
- filespec 「Filespec」 参照
- storage\_clause 「STORAGE 句」 参照
- 例, 4-137
- ALTER TABLE コマンド, 4-105
  - add\_column\_options\_clause, 4-108
  - add\_overflow\_clause, 4-112
  - add\_partition\_clause, 4-114
  - allocate\_extent\_clause, 4-111
  - column\_constraint, table\_constraint 「CONSTRAINT 句」 参照
  - column\_ref\_clause, 4-108
  - deallocate\_unused\_clause 「DEALLOCATE UNUSED 句」 参照
  - drop\_clause 「DROP 句」 参照
  - exchange\_partition\_clause, 4-114
  - index\_organized\_table\_clauses, 4-111
  - LOB\_index\_clause, 4-110
  - LOB\_parameters, 4-109
  - LOB\_storage\_clause, 4-109
  - modify\_column\_options\_clause, 4-108
  - modify\_LOB\_index\_clause, 4-111
  - modify\_LOB\_storage\_clause, 4-110
  - modify\_partition\_clause, 4-113
  - move\_partition\_clause, 4-113, 4-114
  - nested\_table\_storage\_clause, 4-111
  - overflow\_clause, 4-112
  - parallel\_clause 「PARALLEL 句」 参照
  - partitioning\_clauses, 4-113
  - physical\_attributes\_clause, 4-109
  - segment\_partition\_clause, 4-115
  - split\_partition\_clause, 4-114
  - storage\_clause 「STORAGE 句」 参照
  - table\_ref\_clause, 4-108
  - 例, 4-122
- ALTER TRIGGER コマンド, 4-139
  - 例, 4-140
- ALTER TYPE
  - 例, 4-145
- ALTER TYPE コマンド, 4-142
  - pragma\_clause, 4-143
- ALTER USER コマンド, 4-148
  - 例, 4-150
- ALTER VIEW コマンド, 4-152
  - 例, 4-153
- ALTER オブジェクト監査オプション, 4-171
- ALTER オブジェクト権限
  - 順序に対する, 4-446
  - 表に対する, 4-445
- AM/A.M. 書式要素, 3-68
- ANALYZE コマンド, 4-154
  - for\_clause, 4-155
  - 例, 4-160, 4-162
- AND 論理演算子, 3-10
  - 条件での, 3-85
  - 真理値表, 3-11
- ANSI
  - X3.135-1992, 1-2
  - データ型, 2-18
  - 米国規格協会, 1-1
- ANY 比較演算子, 3-5
- ARCHIVE LOG 句, 4-164
  - ALTER SYSTEM コマンドの, 4-96
  - 例, 4-166
- ARCHIVELOG オプション
  - ALTER DATABASE コマンドの, 4-19
  - CREATE CONTROLFILE コマンドの, 4-213
  - CREATE DATABASE コマンドの, 4-217
- AS EXTERNAL 句
  - CREATE FUNCTION コマンドの, 4-231
- AS OBJECT オプション
  - CREATE TYPE コマンドの, 4-346
- AS TABLE オプション
  - CREATE TYPE コマンドの, 4-346
- AS VARRAY オプション
  - CREATE TYPE コマンドの, 4-346
- ASCII
  - キャラクタ・セット, 2-24
  - と EBCDIC, 3-4
  - 文字関数, 3-32
- ASC オプション
  - CREATE INDEX コマンドの, 4-236
  - ORDER BY 句の, 4-491
- AS 句
  - CREATE SNAPSHOT コマンドの, 4-288
  - CREATE TABLE コマンドの, 4-317
  - CREATE VIEW コマンドの, 4-361
- attribute\_name パラメータ
  - ALTER TYPE コマンドの, 4-143
  - CREATE TYPE コマンドの, 4-346
- AUDIT\_TRAIL, 4-168
- AUDIT オブジェクト監査オプション, 4-171
- AUDIT コマンド, 4-167, 4-175
  - 例, 4-172, 4-178

- AUTHENTICATED BY 句
  - CREATE DATABASE LINK コマンドの, 4-222
- authenticated\_clause
  - CREATE DATABASE LINK の, 4-221
- AUTOEXTEND
  - 表領域内のデータ・ファイルのサイズ, 4-135, 4-326
- autoextend\_clause
  - ALTER DATABASE, 4-18
  - ALTER TABLESPACE の, 4-134
  - CREATE DATABASE の, 4-215
  - CREATE TABLESPACE, 4-326
- AUTOEXTEND 句
  - ALTER DATABASE コマンドの, 4-23
  - CREATE DATABASE コマンドの, 4-218
- AUTOMATIC オプション
  - RECOVER 句の, 4-467
- AVG グループ関数, 3-54

## B

---

- BACKUP CONTROLFILE 句
  - ALTER DATABASE コマンドの, 4-21
- BC/B.C. 書式要素, 3-68
- BEFORE オプション
  - CREATE TRIGGER コマンドの, 4-331
- BEGIN BACKUP オプション
  - ALTER TABLESPACE コマンドの, 4-136
- BETWEEN 比較演算子, 3-5
- BFILE
  - アクセス, 4-227
- BFILENAME 関数, 3-47
- BFILE データ型, 2-15
- BITMAP オプション
  - CREATE INDEX コマンドの, 4-236
- BLOB データ型, 2-16
- BODY オプション
  - ALTER PACKAGE コマンドの, 4-38
  - COMPILE 句の
    - ALTER TYPE コマンドの, 4-143
  - DROP PACKAGE コマンドの, 4-389
- BUFFER\_POOL オプション
  - STORAGE 句の, 4-521
- BY ACCESS オプション
  - AUDIT コマンドの, 4-168, 4-176
- BY SESSION オプション
  - AUDIT コマンドの, 4-168, 4-176
- BY 句

- AUDIT コマンドの, 4-168
- NOAUDIT コマンドの, 4-458
- B 書式要素, 3-62

## C

---

- CACHE\_INSTANCES パラメータ
  - ALTER SYSTEM コマンドの, 4-95
- CACHE オプション
  - CREATE TABLE の, 4-318
- CACHE パラメータ
  - CREATE SEQUENCE コマンドの, 4-280, 4-281
- CALLING STANDARD 句
  - CREATE FUNCTION コマンドの, 4-231
  - CREATE PROCEDURE コマンドの, 4-258
- CANCEL オプション
  - RECOVER 句の, 4-468
- CASCADE CONSTRAINT オプション
  - REVOKE(スキーマ・オブジェクト権限) コマンドの, 4-476
- CASCADE オプション
  - ANALYZE コマンドの, 4-157
  - DISABLE 句の, 4-376
  - DROP 句の, 4-379
- CAST 演算子
  - 式の構文, 3-78
- CAST 式, 3-78
- CEIL 数値関数, 3-18
- CHANGE パラメータ
  - ARCHIVE LOG 句の, 4-165
- CHARACTER SET パラメータ
  - CREATE DATABASE コマンドの, 4-217
- CHARTOROWID 変換関数, 3-39
- CHAR データ型, 2-7
  - 値の比較, 2-23
- CHECKPOINT 句
  - ALTER SYSTEM コマンドの, 4-94, 4-98
- CHECK 制約, 3-37, 4-197
- CHR 文字関数, 3-24
- CHUNK オプション
  - LOB 記憶域句の, 4-316
- CHUNK パラメータ
  - LOB 記憶域句の
    - ALTER TABLE コマンドの, 4-117
- CLEAR LOGFILE 句
  - ALTER DATABASE コマンドの, 4-20
- CLOB データ型, 2-16



- CLONE DATABASE オプション
  - ALTER DATABASE コマンドの, 4-18
- CLOSE DATABASE LINK オプション
  - ALTER SESSION コマンドの, 4-62
- CLOSE\_OPEN\_CACHED\_CURSORS パラメータ
  - ALTER SESSION コマンドの, 4-63
- CLUSTER オプション
  - ANALYZE コマンドの, 4-156
  - CREATE INDEX コマンドの, 4-236
  - TRUNCATE コマンドの, 4-533
- CLUSTER 句
  - CREATE SNAPSHOT コマンドの, 4-286
  - CREATE TABLE コマンドの, 4-316
- column\_ref\_clause
  - ALTER TABLE の, 4-108
  - CREATE TABLE の, 4-306
- COMMENT オブジェクト監査オプション, 4-171
- COMMENT 句
  - COMMIT コマンドの, 4-182
- COMMENT コマンド, 4-180
  - 例, 4-180
- COMMIT オプション
  - ADVISE 句の, 4-62
- COMMIT コマンド, 4-182
  - 概要, 4-8
  - トランザクションを終了する, 4-481, 4-516
  - 例, 4-183
- COMPILE オプション
  - ALTER FUNCTION コマンドの, 4-26, 4-38
  - ALTER PROCEDURE コマンドの, 4-41
  - ALTER TYPE コマンドの, 4-143
  - ALTER VIEW コマンドの, 4-152
- COMPLETE オプション
  - REFRESH 句の, 4-80, 4-287
- CONCAT 文字関数, 3-4
- CONNECT BY 句
  - SELECT コマンドの, 4-490, 4-494, 4-495
  - 例, 4-494
- CONNECT TO 句
  - CREATE DATABASE LINK コマンドの, 4-222
- CONNECT TO 句の CURRENT\_USER オプション
  - CREATE DATABASE LINK コマンドの, 4-222
- CONNECT\_TIME パラメータ
  - CREATE PROFILE コマンドの, 4-263
- CONNECT 文監査のショートカット, 4-173
- CONNECT ロール, 4-269
- CONSTRAINT(S) DEFAULT オプション
  - ALTER SESSION コマンドの, 4-63
- CONSTRAINT(S) DEFERRED オプション
  - ALTER SESSION コマンドの, 4-63
- CONSTRAINT オプション
  - DISABLE 句の, 4-376
  - DROP 句の, 4-379
- CONSTRAINT 句
  - foreign\_key\_clause, 4-186
  - index\_physical\_attributes\_clause, 4-187
  - storage\_clause 「STORAGE 句」 参照
  - 例, 4-194, 4-196
- CONSTRAINT 識別子
  - CONSTRAINT 句の, 4-188
  - WITH CHECK OPTION 句の, 4-362
- CONTINUE オプション
  - RECOVER 句の, 4-468
- CONVERT オプション
  - ALTER DATABASE コマンドの, 4-18
- CONVERT 変換関数, 3-40
- COSH 数値関数, 3-19
- COUNT グループ関数, 3-55
- CPU\_PER\_CALL パラメータ
  - CREATE PROFILE コマンドの, 4-263
- CREATE CLUSTER コマンド, 4-202
  - parallel\_clause, 4-203
  - parallel\_clause 「PARALLEL 句」 参照
  - physical\_attributes\_clause, 4-203
  - storage\_clause 「STORAGE 句」 参照
  - 例, 4-208
- CREATE CONTROLFILE コマンド, 4-210
  - filespec 「FILESPEC」 参照
  - 例, 4-213
- CREATE DATABASE LINK コマンド, 4-220
  - authenticated\_clause, 4-221
- CREATE DATABASE コマンド, 4-214
  - autoextend\_clause, 4-215
- CREATE DIRECTORY コマンド, 4-226
- CREATE FUNCTION コマンド, 4-228
- CREATE INDEX コマンド, 4-233
  - global\_index\_clause, 4-234
  - global\_partition\_clause, 4-235
  - index\_physical\_attributes\_clause, 4-235
  - local\_index\_clause, 4-235
  - parallel 句 「PARALLEL 句」 参照
  - storage\_clause 「STORAGE 句」 参照
- CREATE LIBRARY コマンド, 4-244
  - filespec 「FILESPEC」 参照

CREATE PACKAGE BODY コマンド, 4-250  
 例, 4-251

CREATE PACKAGE パッケージ, 4-246  
 例, 4-248

CREATE PROCEDURE コマンド, 4-255

CREATE PROFILE コマンド, 4-261  
 例, 4-265

CREATE ROLE コマンド, 4-268  
 例, 4-270

CREATE ROLLBACK SEGMENT コマンド, 4-272  
 storage\_clause 「STORAGE 句」参照  
 例, 4-273

CREATE SCHEMA コマンド, 4-275  
 例, 4-276

CREATE SEQUENCE コマンド, 4-278  
 例, 4-282

CREATE SNAPSHOT LOG コマンド, 4-294  
 LOB\_storage\_clause 「CREATE TABLE コマンド」参照  
 parallel\_clause 「PARALLEL 句」参照  
 physical\_attributes\_clause 「CREATE TABLE コマンド」参照  
 storage\_clause 「STORAGE 句」参照  
 table\_partition\_clause 「CREATE TABLE コマンド」参照  
 例, 4-297

CREATE SNAPSHOT コマンド, 4-283  
 index\_physical\_attributes\_clause 「ALTER TABLE コマンド」参照  
 LOB\_storage\_clause 「CREATE TABLE コマンド」参照  
 parallel\_clause 「PARALLEL 句」参照  
 physical\_attributes\_clause 「ALTER TABLE コマンド」参照  
 select\_command 「SELECT コマンド」参照  
 table\_partition\_clause 「CREATE TABLE コマンド」参照  
 例, 4-290

CREATE STANDBY CONTROLFILE オプション  
 ALTER DATABASE コマンドの, 4-21

CREATE SYNONYM コマンド, 4-299  
 例, 4-301

CREATE TABLESPACE コマンド, 4-325  
 autoextend\_clause, 4-326  
 filespec, 4-325  
 storage\_clause 「STORAGE 句」参照, 4-326  
 例, 4-328

CREATE TABLE コマンド, 4-303  
 column\_ref\_clause, 4-306  
 disable\_clause 「DISABLE 句」参照  
 enable\_clause 「ENABLE 句」参照  
 index\_organized\_table\_clause, 4-308  
 LOB\_index\_clause, 4-309  
 LOB\_storage\_clause, 4-308  
 nested\_table\_storage\_clause, 4-310  
 parallel\_clause 「PARALLEL 句」参照  
 physical\_attributes\_clause, 4-307  
 segment\_attributes\_clause, 4-307  
 storage\_clause 「STORAGE 句」参照  
 subquery 「副問合せ」参照  
 table\_partition\_clause, 4-310  
 table\_ref\_clause, 4-307  
 例, 4-318, 4-319

CREATE TRIGGER コマンド, 4-330  
 例, 4-337

CREATE TRIGGER コマンドの ORDER MEMBER 句, 4-347

CREATE TYPE BODY  
 例, 4-352

CREATE TYPE BODY コマンド, 4-350

CREATE TYPE コマンド, 4-342  
 pragma\_clause, 4-343  
 例, 4-348

CREATE TYPE コマンドの function\_specification パラメータ, 4-347

CREATE TYPE コマンドの MAP MEMBER 句, 4-347

CREATE TYPE コマンドの MEMBER 句, 4-346

CREATE TYPE コマンドの method\_name パラメータ, 4-348

CREATE TYPE コマンドの PRAGMA  
 RESTRICT\_REFERENCES 句, 4-348

CREATE TYPE コマンドの procedure\_specification パラメータ, 4-347

CREATE TYPE コマンドの RNDS パラメータ, 4-348

CREATE TYPE コマンドの RNPS パラメータ, 4-348

CREATE TYPE コマンドの WNDS パラメータ, 4-348

CREATE TYPE コマンドの WNPS パラメータ, 4-348

CREATE USER コマンド, 4-353  
 例, 4-357

CREATE VIEW コマンド, 4-359  
 副問合せ, 4-360  
 例, 4-365

CURRENT オプション  
 ARCHIVE LOG 句の, 4-165

CURRVAL 疑似列, 2-30

例, 2-32, 4-530

CYCLE オプション

CREATE SEQUENCE コマンドの, 4-280

C 書式要素, 3-62

## D

---

DANGLING REF, 2-21

DATABASE オプション

RECOVER 句の, 4-468

DATABASE パラメータ

CREATE CONTROLFILE コマンドの, 4-211

DATAFILE オプション

RECOVER 句の, 4-468

DATAFILE 句

ALTER DATABASE コマンドの, 4-23

CREATE CONTROLFILE 句の, 4-212

CREATE DATABASE コマンドの, 4-218

CREATE TABLESPACE コマンドの, 4-326

DATAFILE パラメータ

ALLOCATE EXTENT 句の, 4-12, 4-33, 4-118

DATE 書式要素, 3-62

DATE データ型, 2-12

値の比較, 2-22

ユリウス, 2-13

DAY 書式要素, 3-68

DB2

データ型, 2-18

DBA\_CLUSTERS ビュー, 4-160

DBA\_INDEXES ビュー, 4-159

DBA\_TAB\_COLUMNS ビュー, 4-160

DBA\_TABLES ビュー, 4-159

DBA 文監査のショートカット, 4-173

DBA ロール, 4-270

DBMS\_SNAPSHOT.REFRESH() プロシージャ, 4-289

DBMSSTD.SQL, 4-228, 4-246, 4-250, 4-255, 4-330

DDL(データ定義言語), 4-2

DEALLOCATE UNUSED 句, 4-368

ALTER TABLE コマンドの, 4-118

ECODE 式, 3-83

DEFAULT ROLE 句

ALTER USER コマンドの, 4-150

DEFAULT オプション

ALTER PROFILE コマンドの, 4-48

BUFFER\_POOL オプションの

STORAGE 句の, 4-521

CREATE PROFILE コマンドの, 4-264

CREATE TABLE コマンドの, 4-311

NOAUDIT コマンドの, 4-461

ROLLBACK SEGMENT 句の, 4-286

ALTER SNAPSHOT コマンドの, 4-80

DEFAULT プロファイル, 4-265, 4-355, 4-393

DEFERRABLE オプション

CONSTRAINT 句の, 4-188

DEFERRABLE 制約, 4-200

DELETE\_CATALOG\_ROLE ロール, 4-270

DELETE オブジェクト監査オプション, 4-171

DELETE オブジェクト権限

ビューに対する, 4-446

表に対する, 4-445

DELETE オプション

CREATE TRIGGER コマンドの, 4-332, 4-335

DELETE コマンド, 4-370

returning\_clause, 4-371

概要, 4-7

副問合せ, 4-371

DEREF 関数, 3-53

DESC オプション

CREATE INDEX コマンドの, 4-236

ORDER BY 句の, 4-491

DIRECTORY オプション

GRANT(オブジェクト権限)の, 4-444

DISABLE COMMIT IN PROCEDURE オプション

ALTER SESSION コマンドの, 4-62

DISABLE STORAGE IN ROW オプション

LOB 記憶域句の

ALTER TABLE コマンドの, 4-117

CREATE TABLE コマンドの, 4-315

DISABLE オプション

ALTER TRIGGER コマンドの, 4-139

CONSTRAINT 句の, 4-189

PARALLEL DML 句の

ALTER SESSION コマンドの, 4-63

DISABLE 句, 4-375

ALTER DATABASE コマンドの, 4-22

CREATE TABLE コマンドの, 4-317

例, 4-377

DISCONNECT SESSION 句

ALTER SYSTEM コマンドの, 4-96

dispatch\_clause

ALTER SYSTEM の, 4-92

DISTINCT オプション

SELECT コマンドの, 4-489

SQL グループ関数の , 3-54  
DISTINCT 句  
と ORDER BY 句 , 4-499  
DML( データ操作言語 ) , 4-7  
DROP CLUSTER コマンド , 4-381  
例 , 4-382  
DROP DATABASE LINK コマンド , 4-383  
例 , 4-383  
DROP DIRECTORY コマンド , 4-384  
例 , 4-384  
DROP FUNCTION コマンド , 4-385  
例 , 4-385  
DROP INDEX コマンド , 4-387  
例 , 4-387  
DROP LIBRARY コマンド , 4-388  
DROP LOGFILE MEMBER 句  
ALTER DATABASE コマンドの , 4-20  
DROP LOGFILE 句  
ALTER DATABASE コマンドの , 4-20  
DROP PACKAGE コマンド , 4-389  
DROP PARTITION オプション  
ALTER TABLE コマンドの , 4-120  
DROP PROCEDURE コマンド , 4-391  
例 , 4-390, 4-391  
DROP PROFILE コマンド , 4-393  
例 , 4-393  
DROP ROLE コマンド , 4-394  
例 , 4-394  
DROP ROLLBACK SEGMENT コマンド , 4-395  
例 , 4-395  
DROP SEQUENCE コマンド , 4-397  
例 , 4-397  
DROP SNAPSHOT LOG コマンド , 4-400  
例 , 4-400  
DROP SNAPSHOT コマンド , 4-399  
例 , 4-399  
DROP STORAGE オプション  
TRUNCATE コマンドの , 4-533  
DROP SYNONYM コマンド , 4-401  
例 , 4-401  
DROP TABLESPACE コマンド , 4-404  
例 , 4-405  
DROP TABLE コマンド , 4-402  
例 , 4-403  
DROP TRIGGER コマンド , 4-406  
例 , 4-406  
DROP TYPE BODY コマンド , 4-409

DROP TYPE コマンド , 4-407  
DROP USER コマンド , 4-410  
例 , 4-411  
DROP VIEW コマンド  
例 , 4-412  
DROP 句 , 4-379  
ALTER TABLE コマンドの , 4-117  
例 , 4-380  
DUAL データ・ディクショナリ表  
から選択する例 , 4-530  
定義 , 4-530  
DUMMY 列  
DUAL 表の , 4-530  
DUMP 関数 , 3-46  
DY 書式要素 , 3-68  
D 書式要素 , 3-62

## E

---

EBCDIC  
キャラクタ・セット , 2-24  
と ASCII , 3-4  
EEEE 書式要素 , 3-62  
EMPTY\_BLOB 関数 , 3-47  
EMPTY\_CLOB 関数 , 3-47  
ENABLE DISTRIBUTED RECOVERY オプション  
ALTER SYSTEM コマンドの , 4-102  
ENABLE NOVALIDATE , 4-417  
ENABLE NOVALIDATE オプション  
CONSTRAINT 句の , 4-189  
ENABLE NOVALIDATE 制約 , 4-189, 4-416  
ENABLE STORAGE IN ROW オプション  
LOB 記憶域句の  
ALTER TABLE コマンドの , 4-116  
CREATE TABLE コマンドの , 4-315  
ENABLE VALIDATE , 4-417  
ENABLE VALIDATE オプション  
CONSTRAINT 句の , 4-189  
ENABLE VALIDATE 制約 , 4-189, 4-415  
ENABLE オプション  
ALTER TRIGGER コマンドの , 4-139  
PARALLEL DML 句の  
ALTER SESSION コマンドの , 4-63  
ENABLE 句 , 4-414  
ALTER DATABASE コマンドの , 4-22  
CREATE TABLE コマンドの , 4-317  
except\_clause , 4-415

- exceptions\_clause, 4-415
- storage\_clause 「STORAGE 句」 参照
- using\_index\_clause, 4-415
- 例, 4-419
- END BACKUP
  - ALTER DATABASE コマンドの, 4-23
- END BACKUP オプション
  - ALTER TABLESPACE コマンドの, 4-136
- ENTRYID オプション
  - USERENV 関数の, 3-52
- ESCAPE 文字
  - LIKE 演算子の, 3-8
- except\_clause
  - ENABLE 句の, 4-415
- EXCEPTIONS INTO 句
  - CONSTRAINT 句の, 4-189
- exceptions\_clause
  - ENABLE, 4-415
- EXCEPT 句
  - SET ROLE コマンドの, 4-511
- EXCHANGE PARTITION オプション
  - ALTER TABLE コマンドの, 4-120
- exchange\_partition\_clause
  - ALTER TABLE の, 4-114
- EXECUTE\_CATALOG\_ROLE ロール, 4-270
- EXECUTE オブジェクト監査オプション, 4-171
- EXECUTE オブジェクト権限
  - プロシージャ、関数、パッケージに対する, 4-446
- EXISTS 比較演算子, 3-5
- EXPLAIN PLAN
  - パーティション表の分析, 4-425
- EXPLAIN PLAN コマンド, 4-422
  - 概要, 4-7
  - 例, 4-424
- EXP 数値関数, 3-19
- EXTERNALLY オプション
  - IDENTIFIED 句の, 4-268, 4-355, 4-356
- EXTERNAL 句
  - CREATE PROCEDURE コマンドの, 4-258

## F

---

- FAILED\_LOGIN\_ATTEMPTS オプション
  - CREATE PROFILE コマンドの, 4-263
- FALSE
  - 条件の結果, 3-84
- FAST オプション

- REFRESH 句の, 4-80, 4-287
- filespec, 4-428
  - CREATE TABLESPACE, 4-325
  - 例, 4-430
- FIPS, 4-63
  - PUB 127-2, 1-2
  - フラグを使用する, 4-71
  - 連邦情報処理標準, 1-2
- FIPS フラガー, B-13
- FLAGGER 句
  - ALTER SESSION コマンドの, 4-63
- FLOAT
  - ANSI データ型, 2-11
- FLOOR 数値関数, 3-19
- FM 書式モデル修飾子, 3-71
- FM 日付書式要素の接頭辞
  - 例, 3-71
- FOR EACH ROW オプション
  - CREATE TRIGGER コマンドの, 4-333
- FOR RECOVER オプション
  - ALTER TABLESPACE コマンドの, 4-136
- FOR UPDATE OF
  - 例, 4-501
- FOR UPDATE オプション
  - CREATE SNAPSHOT コマンドの, 4-287
- FOR UPDATE 句
  - SELECT コマンドの, 4-491, 4-499
- for\_clause
  - ANALYZE の, 4-155
- FORCE オプション
  - CREATE VIEW コマンドの, 4-360
  - PARALLEL DML 句の
    - ALTER SESSION コマンドの, 4-63
  - REFRESH 句の, 4-80, 4-287
  - REVOKE( スキーマ・オブジェクト権限 ) コマンドの, 4-476
- FORCE 句
  - COMMIT コマンドの, 4-183
  - ROLLBACK コマンドの, 4-481
- FOREIGN KEY オプション
  - CONSTRAINT 句の, 4-194
- foreign\_key\_clause
  - CONSTRAINT 句の, 4-186
- FOR 句
  - EXPLAIN PLAN コマンドの, 4-423
- FROM 句
  - REVOKE コマンドの, 4-472, 4-476

FROM パラメータ  
  RECOVER 句の, 4-468  
function\_specification パラメータ  
  CREATE TYPE コマンドの, 4-145  
FX 書式モデル修飾子, 3-71

## G

---

global\_index\_clause  
  CREATE INDEX の, 4-234  
GLOBAL\_NAMES, 4-99  
global\_partition\_clause  
  CREATE INDEX の, 4-235  
GLOBALLY AS オプション  
  IDENTIFIED 句の, 4-355  
GLOBALLY オプション  
  IDENTIFIED 句の, 4-269  
GLOBAL オプション  
  CHECK DATAFILES 句の, 4-94  
  CHECKPOINT 句の, 4-94  
GRANT OPTION  
  GRANT コマンドの, 4-444  
GRANT オブジェクト監査オプション, 4-171  
GRANT コマンド, 4-432, 4-442  
  CREATE SCHEMA コマンドの一部, 4-275  
  例, 4-440, 4-447  
GRAPHIC データ型, 2-20  
GREATEST 関数, 3-48  
GROUP BY 句  
  SELECT コマンドの, 4-490, 4-496  
  グループ SQL 関数と, 3-16  
GROUP パラメータ  
  ADD LOGFILE MEMBER 句の, 4-20  
  DROP LOGFILE 句の, 4-20  
G 書式要素, 3-62

## H

---

HASH パラメータ  
  CREATE CLUSTER コマンドの, 4-205  
HAVING 句  
  SELECT コマンドの, 4-490, 4-497  
HEXTORAW 変換関数, 3-41

## I

---

IDENTIFIED 句

  CREATE ROLE コマンドの, 4-268  
  CREATE USER コマンドの, 4-355  
IDLE\_TIME パラメータ  
  CREATE PROFILE コマンドの, 4-263  
IEC( 国際電気標準会議), 1-1  
IMMEDIATE オプション  
  ALTER TABLESPACE コマンドの, 4-136  
INCLUDING 句  
  ALTER TABLE コマンドの, 4-116  
  CREATE TABLE コマンドの, 4-315  
INCREMENT BY 句  
  CREATE SEQUENCE コマンドの, 4-279  
index\_organized\_table\_clause  
  CREATE TABLE の, 4-308  
index\_organized\_table\_clauses  
  ALTER TABLE の, 4-111  
index\_physical\_attributes\_clause  
  ALTER INDEX の, 4-30  
  CONSTRAINT 句の, 4-187  
  CREATE INDEX の, 4-235  
INDEX\_STATS ビュー, 4-161  
INDEX オブジェクト監査オプション, 4-171  
INDEX オブジェクト権限  
  表に対する, 4-445  
INDEX オプション  
  ANALYZE コマンドの, 4-156  
  CREATE CLUSTER コマンドの, 4-204  
INDEX パラメータ  
  LOB 記憶域句の, 4-316  
INITCAP 文字関数, 3-25  
INITIALLY DEFERRABLE オプション  
  CONSTRAINT 句の, 4-188  
INITIALLY IMMEDIATE オプション  
  CONSTRAINT 句の, 4-188  
INITIAL パラメータ  
  STORAGE 句の, 4-519, 4-522  
INTRANS パラメータ  
  ALTER CLUSTER コマンドの, 4-12  
  ALTER INDEX コマンドの, 4-32  
  ALTER SNAPSHOT コマンドの, 4-79  
  ALTER TABLE コマンドの, 4-85, 4-116  
  CREATE CLUSTER コマンドの, 4-204  
  CREATE INDEX コマンドの, 4-237  
  CREATE SNAPSHOT コマンドの, 4-286, 4-296  
  CREATE TABLE コマンドの, 4-312  
INSERT オブジェクト監査オプション, 4-171  
INSERT オブジェクト権限

- ビューに対する, 4-446
- 表に対する, 4-445
- INSERT オプション
  - CREATE TRIGGER コマンドの, 4-332
- INSERT コマンド, 2-11, 4-449
  - 概要, 4-7
  - 例, 2-32, 4-453
- INSTANCE 句
  - ALTER SESSION コマンドの, 4-64
- INSTANCE パラメータ
  - ALLOCATE EXTENT 句の, 4-13, 4-33, 4-118
- INSTEAD OF オプション
  - CREATE TRIGGER コマンドの, 4-332
- INSTEAD OF トリガー
  - 使用, 4-339
- INSTR 文字関数, 3-32
- INTEGER データ型, 2-3
- INTERSECT 集合演算子, 4-491
  - 例, 3-14
- INTO 句
  - ANALYZE コマンドの, 4-157
  - EXPLAIN PLAN コマンドの, 4-423
- IN 比較演算子, 3-5
  - 定義, 3-5
- IS DANGLING, 2-21
- IS NOT DANGLING, 2-21
- IS NOT NULL 比較演算子, 3-5
- IS NULL 比較演算子, 3-5
- ISDBA オプション
  - USERENV 関数の, 3-52
- ISO
  - ISO/IEC 9075
    - 1992, 1-2
  - 国際標準化機構, 1-1
- ISOLATION LEVEL 句
  - SET TRANSACTION コマンドの, 4-514
- ISOLATION\_LEVEL 句
  - ALTER SESSION コマンドの, 4-64
- IW 日付書式要素, 3-68
- IYYY 日付書式要素, 3-68
- IYY 日付書式要素, 3-68
- IY 日付書式要素, 3-68
- I 日付書式要素, 3-68

## K

---

- KEEP オプション

- BUFFER\_POOL オプションの
  - STORAGE 句の, 4-521
- KILL SESSION 句
  - ALTER SYSTEM コマンドの, 4-96

## L

---

- LANGUAGE オプション
  - USERENV 関数の, 3-52
- LANGUAGE 句
  - CREATE FUNCTION コマンドの, 4-231
  - CREATE PROCEDURE コマンドの, 4-258
- LAST\_DAY 日付関数, 3-35
- LEAST 関数, 3-48
- LENGTHB 文字関数, 3-34
- LENGTH 文字関数, 3-33
- LEVEL 疑似列, 2-33
- LIBRARY 句
  - CREATE FUNCTION コマンドの, 4-231
  - CREATE PROCEDURE コマンドの, 4-258
- LIKE 比較演算子, 3-5
  - 定義, 3-7
- LIST CHAINED ROWS 句
  - ANALYZE コマンドの, 4-157
- LN 数値関数, 3-19
- LOB\_index\_clause
  - ALTER TABLE の, 4-110
  - CREATE TABLE の, 4-309
- LOB\_parameters
  - ALTER TABLE の, 4-109
- LOB\_storage\_clause
  - ALTER TABLE の, 4-109
  - CREATE TABLE の, 4-308
- LOB 関数, 3-47
- LOB 記憶域
  - 行外, 4-117, 4-315
  - 行内, 4-315
  - 行内の, 4-116
- LOB 記憶域句
  - ALTER SNAPSHOT コマンドの, 4-79
  - CREATE TABLE コマンドの, 4-315
- LOB 記憶域句の変更
  - ALTER SNAPSHOT コマンドの, 4-79
- LOB 記憶領域パラメータ
  - ALTER TABLE コマンドでの指定, 4-117
- LOB データ型, 2-14
- LOB の格納, 4-116, 4-117, 4-315

## LOB 列

表に追加する, 4-124

### local\_index\_clause

CREATE INDEX の, 4-235

### LOCAL オプション

CHECK DATAFILES 句の, 4-94

CHECKPOINT 句の, 4-94

ROLLBACK SEGMENT 句の, 4-286

ALTER SNAPSHOT コマンドの, 4-81

### LOCAL パラメータ

ANALYZE コマンドの, 4-157

### LOCK TABLE コマンド, 4-455

概要, 4-7

例, 4-456

### LOCK オブジェクト監査オプション, 4-171

### LOG\_FILES, 4-216

### LOGFILE 句

CREATE CONTROLFILE コマンドの, 4-211

CREATE DATABASE コマンドの, 4-216

### LOGFILE パラメータ

ARCHIVE LOG 句の, 4-165

RECOVER 句の, 4-468

### LOGGING オプション

ALTER INDEX コマンドの, 4-33

ALTER TABLESPACE コマンドの, 4-134

ALTER TABLE コマンドの, 4-119

CREATE INDEX コマンドの, 4-237

CREATE TABLESPACE コマンドの, 4-326

CREATE TABLE コマンドの, 4-314

### LOG 数値関数, 3-20

### LONG RAW データ型, 2-13

LONG データ型との類似点, 2-13

索引付けの禁止, 2-14

### LONG VARGRAPHIC データ型, 2-20

### LONG データ型, 2-11

最大長, 2-11

制限, 2-11

### LOWER 文字関数, 3-25

### LPAD 文字関数, 3-26

### LTRIM 文字関数, 3-26

### L 書式要素, 3-62

## M

---

### MAKE\_REF 関数, 3-53

### MAP MEMBER 句

CREATE TYPE コマンドの, 4-144

### MAP メソッド

オブジェクト値の比較, 2-25

### MASTER オプション

ROLLBACK SEGMENT 句の, 4-286

ALTER SNAPSHOT コマンドの, 4-81

### MAX\_DUMP\_FILE\_SIZE オプション

ALTER SESSION コマンドの, 4-64

ALTER SYSTEM コマンドの, 4-96

### MAX\_ENABLE\_ROLES, 4-512

### MAXDATAFILES パラメータ

CREATE CONTROLFILE コマンドの, 4-212

CREATE DATABASE コマンドの, 4-217

### MAXEXTENTS パラメータ

STORAGE 句の, 4-520, 4-522

### MAXINSTANCES パラメータ

CREATE CONTROLFILE コマンドの, 4-213

CREATE DATABASE コマンドの, 4-217

### MAXLOGFILES パラメータ

CREATE CONTROLFILE コマンドの, 4-212

CREATE DATABASE コマンドの, 4-216

### MAXLOGHISTORY パラメータ

CREATE DATABASE コマンドの, 4-217

### MAXLOGMEMBERS パラメータ

CREATE DATABASE コマンドの, 4-217

### MAXSIZE 句

ALTER DATABASE コマンドの, 4-23

### MAXTRANS パラメータ

ALTER CLUSTER コマンドの, 4-12

ALTER INDEX コマンドの, 4-32

ALTER SNAPSHOT コマンドの, 4-79

ALTER TABLE コマンドの, 4-85, 4-116

CREATE CLUSTER コマンドの, 4-204

CREATE INDEX コマンドの, 4-237

CREATE SNAPSHOT コマンドの, 4-286, 4-296

CREATE TABLE コマンドの, 4-313

### MAXVALUE パラメータ

CREATE SEQUENCE コマンドの, 4-279

パーティション化句の, 4-317

### MAX グループ関数, 3-55

### MEMBER 句

CREATE TYPE コマンドの, 4-144

### method\_name パラメータ

CREATE TYPE コマンドの, 4-145

### MINEXTENTS パラメータ

STORAGE 句の, 4-520

### MINIMUM EXTENT パラメータ

ALTER TABLESPACE コマンドの, 4-135



- CREATE TABLESPACE コマンドの, 4-326
- MINUS 集合演算子, 4-491
  - 例, 3-14
- MINVALUE パラメータ
  - CREATE SEQUENCE コマンドの, 4-280
- MI 書式要素, 3-62
- MLSLABE データ型, 2-18
- MODIFY DEFAULT ATTRIBUTES
  - ALTER INDEX コマンドの, 4-34, 4-86
- MODIFY DEFAULT ATTRIBUTES オプション
  - ALTER SNAPSHOT コマンドの, 4-79
  - ALTER TABLE コマンドの, 4-116
- MODIFY PARTITION オプション
  - ALTER SNAPSHOT コマンドの, 4-79, 4-85
  - ALTER TABLE コマンドの, 4-119
- modify\_column\_options\_clause
  - ALTER TABLE の, 4-108
- modify\_LOB\_index\_clause
  - ALTER TABLE の, 4-111
- modify\_LOB\_storage\_clause
  - ALTER TABLE の, 4-110
- modify\_partition\_clause
  - ALTER TABLE の, 4-113
- MODIFY 句
  - ALTER TABLE コマンドの, 4-115
- MONTHS\_BETWEEN 日付関数, 3-36
- MONTH 書式要素, 3-68
- MON 書式要素, 3-68
- MOUNT オプション
  - ALTER DATABASE コマンドの, 4-18
- MOVE PARTITION オプション
  - ALTER SNAPSHOT コマンドの, 4-79, 4-85
  - ALTER TABLE コマンドの, 4-120
- move\_partition\_clause
  - ALTER TABLE の, 4-113, 4-114
- MTS\_DISPATCHERS パラメータ
  - ALTER SYSTEM コマンドの, 4-95, 4-99
- MTS\_MAX\_DISPATCHERS, 4-99
- MTS\_MAX\_SERVERS, 4-99
- MTS\_SERVERS, 4-99
- MTS\_SERVERS パラメータ
  - ALTER SYSTEM コマンドの, 4-95, 4-99

## N

- NAME 句
  - CREATE FUNCTION コマンドの, 4-231
- CREATE PROCEDURE コマンドの, 4-258
- NATIONAL CHARACTER SET パラメータ
  - CREATE DATABASE コマンドの, 4-218
- NCHAR データ型, 2-7
- NCLOB データ型, 2-16
- NESTED TABLE 記憶域句
  - ALTER TABLE コマンドの, 4-117
  - CREATE TABLE コマンドの, 4-316
- nested\_table\_storage\_clause
  - ALTER TABLE の, 4-111
  - CREATE TABLE の, 4-310
- NEW\_TIME 日付関数, 3-36
- NEXT\_DAY 日付関数, 3-37
- NEXTVAL 疑似列, 2-30
  - 例, 2-32, 4-454, 4-530
- NEXT オプション
  - ARCHIVE LOG 句の, 4-165
- NEXT 句
  - ALTER DATABASE コマンドの, 4-23
- NEXT パラメータ
  - REFRESH 句の, 4-80, 4-287
  - STORAGE 句の, 4-520
- NIST
  - 米国連邦情報・技術局, 1-2
- NLS\_CHARSET\_DECL\_LEN 関数, 3-49
- NLS\_CHARSET\_ID 関数, 3-49
- NLS\_CHARSET\_NAME 関数, 3-50
- NLS\_CURRENCY パラメータ
  - ALTER SESSION コマンドの, 4-65
- NLS\_DATE\_FORMAT パラメータ
  - ALTER SESSION コマンドの, 4-65
- NLS\_DATE\_LANGUAGE パラメータ
  - ALTER SESSION コマンドの, 4-65
- NLS\_INITCAP 文字関数, 3-27
- NLS\_ISO\_CURRENCY パラメータ
  - ALTER SESSION コマンドの, 4-65
- NLS\_LANGUAGE パラメータ
  - ALTER SESSION コマンドの, 4-65
- NLS\_LOWER 文字関数, 3-21, 3-24, 3-27
- NLS\_NUMERIC\_CHARACTERS パラメータ
  - ALTER SESSION コマンドの, 4-65
- NLS\_SORT パラメータ
  - ALTER SESSION コマンドの, 4-65
- NLS\_TERRITORY パラメータ
  - ALTER SESSION コマンドの, 4-65
- NLS\_UPPER 文字関数, 3-27
- NLSSORT 文字関数, 3-34

NOARCHIVELOG オプション  
    ALTER DATABASE コマンドの, 4-19  
    CREATE DATABASE コマンドの, 4-217  
NOAUDIT コマンド, 4-458, 4-460  
    例, 4-459, 4-461  
NOCACHE オプション  
    ALTER TABLE コマンドの, 4-118  
    CREATE SEQUENCE コマンドの, 4-280  
    CREATE TABLE の, 4-318  
NOCYCLE オプション  
    CREATE SEQUENCE コマンドの, 4-280  
NOFORCE オプション  
    CREATE VIEW コマンドの, 4-360  
NOLOGGING オプション  
    ALTER INDEX コマンドの, 4-33  
    ALTER TABLESPACE コマンドの, 4-134  
    ALTER TABLE コマンドの, 4-119  
    CREATE INDEX コマンドの, 4-240  
    CREATE TABLESPACE コマンドの, 4-326  
    CREATE TABLE コマンドの, 4-314  
NOMAXVALUE オプション  
    CREATE SEQUENCE コマンドの, 4-279  
NOMINVALUE オプション  
    CREATE SEQUENCE コマンドの, 4-280  
NONE オプション  
    SET ROLE コマンドの, 4-512  
NOORDER オプション  
    CREATE SEQUENCE コマンドの, 4-280  
NORESETLOGS オプション  
    CREATE CONTROLFILE コマンドの, 4-212  
NOREVERSE オプション  
    ALTER INDEX コマンドの, 4-32  
NORMAL オプション  
    ALTER TABLESPACE コマンドの, 4-135  
NOSORT オプション  
    CREATE INDEX コマンドの, 4-237, 4-240  
NOT DEFERRABLE オプション  
    CONSTRAINT 句の, 4-188  
NOT IN 比較演算子, 3-5  
    例, 3-7  
NOT LIKE 比較演算子, 3-5  
NOT NULL 句  
    ALTER TABLE コマンド, 4-122  
NOT NULL 制約, 4-190  
NOT オプション  
    WHENEVER 句の, 4-168, 4-176, 4-177, 4-458, 4-461  
NOT 論理演算子, 3-7

真理値表, 3-11  
NOVALIDATE オプション  
    ENABLE 句の, 4-416  
    CONSTRAINT 句の, 4-189  
NOWAIT オプション  
    FOR UPDATE 句の, 4-491  
    LOCK TABLE コマンドの, 4-456  
NULL, 2-28  
    索引, 4-241  
    条件での, 3-88  
    制約、例, 4-190  
    ビットマップ索引内の, 4-241  
NULL 値  
    OPTIMAL パラメータの, 4-273, 4-521  
NUMBER データ型, 2-9  
    値の比較, 2-22  
NVARCHAR2 データ型, 2-8  
NVL 関数, 2-28, 3-50

## O

---

OF datatype 句  
    CREATE TYPE コマンドの, 4-346  
OFFLINE オプション  
    ALTER ROLLBACK SEGMENT コマンドの, 4-54  
    ALTER TABLESPACE コマンドの, 4-135  
    CREATE TABLESPACE コマンドの, 4-327  
    DATAFILE 句の, 4-23  
OIDINDEX 句  
    CREATE TABLE コマンドの, 4-312  
ON DELETE CASCADE オプション  
    CONSTRAINT 句の, 4-194  
ONLINE オプション  
    ALTER ROLLBACK SEGMENT コマンドの, 4-54  
    ALTER TABLESPACE コマンドの, 4-135  
    CREATE TABLESPACE コマンドの, 4-327  
    DATAFILE 句の, 4-23  
ON 句  
    CREATE TRIGGER コマンドの, 4-332, 4-335  
    GRANT コマンドの, 4-443  
    NOAUDIT コマンドの, 4-460  
    REVOKE コマンドの, 4-476  
OPEN\_LINKS, 4-222  
OPEN オプション  
    ALTER DATABASE コマンドの, 4-18  
OPTIMIZER\_MODE, 4-70  
OPTIMIZER\_MODE パラメータ

- ALTER SESSION コマンドの, 4-65
- opts\_clause
  - ALTER SYSTEM の, 4-93
- OR REPLACE オプション
  - CREATE FUNCTION コマンドの, 4-230
  - CREATE PACKAGE BODY コマンドの, 4-251
  - CREATE PACKAGE コマンドの, 4-246
  - CREATE PROCEDURE コマンドの, 4-257
  - CREATE TRIGGER コマンドの, 4-331
  - CREATE TYPE コマンドの, 4-345
  - CREATE VIEW コマンドの, 4-360
- Oracle7 の正常終了, 4-184
- ORACLE 識別子
  - 構成, A-3
- ORDER BY 句
  - SELECT コマンドの, 4-491, 4-498
  - と ROWNUM 疑似列, 2-35
- ORDER オプション
  - CREATE SEQUENCE コマンドの, 4-280
- ORDER メソッド
  - オブジェクト値の比較, 2-25
- ORGANIZATION HEAP オプション
  - CREATE TABLE コマンドの, 4-314
- ORGANIZATION INDEX オプション
  - CREATE TABLE コマンドの, 4-314
- OR 論理演算子
  - 真理値表, 3-11
- OS\_AUTHENT\_PREFIX
  - 初期化パラメータ, 4-357
- OS\_ROLES, 4-439
- outer\_join
  - SELECT, 4-504
- overflow\_clause
  - ALTER TABLE の, 4-112
- OVERFLOW 句
  - ALTER TABLE コマンドの, 4-116
  - CREATE TABLE コマンドの, 4-315

## P

---

- PACKAGE オプション
  - ALTER PACKAGE コマンドの, 4-38
- PARALLEL DML オプション
  - ALTER SESSION コマンドの, 4-62
- parallel\_clause
  - CREATE CLUSTER の, 4-203
- PARALLEL\_DEFAULT\_SCANSIZE パラメータ, 4-463
- PARALLEL 句, 4-462
  - RECOVER 句の, 4-468
- PARAMETERS 句
  - CREATE FUNCTION コマンドの, 4-231
  - CREATE PROCEDURE コマンドの, 4-258
- PARTITION BY RANGE 句
  - CREATE TABLE コマンドの, 4-316
- partition\_description\_clause
  - ALTER INDEX の, 4-31
- partition\_start 列
  - EXPLAIN PLAN の, 4-425
- partition\_stop 列
  - EXPLAIN PLAN の, 4-425
- partitioning\_clauses
  - ALTER TABLE の, 4-113
- PARTITION オプション
  - ANALYZE コマンドの, 4-156
- PARTITION 句
  - UPDATE コマンドの, 4-538
- PASSWORD\_GRACE\_TIME オプション
  - CREATE PROFILE コマンドの, 4-263
- PASSWORD\_LIFE\_TIME オプション
  - CREATE PROFILE コマンドの, 4-263
- PASSWORD\_LOCK\_TIME オプション
  - CREATE PROFILE コマンドの, 4-263
- PASSWORD\_REUSE\_MAX オプション
  - CREATE PROFILE コマンドの, 4-263
- PASSWORD\_REUSE\_TIME オプション
  - CREATE PROFILE コマンドの, 4-263
- PASSWORD\_VERIFY\_FUNCTION オプション
  - CREATE PROFILE コマンドの, 4-263
- PCTFREE パラメータ
  - ALTER CLUSTER コマンドの, 4-12
  - ALTER SNAPSHOT コマンドの, 4-79
  - ALTER TABLE コマンドの, 4-85, 4-116
  - CREATE CLUSTER コマンドの, 4-204
  - CREATE INDEX コマンドの, 4-237
  - CREATE SNAPSHOT コマンドの, 4-286, 4-296
  - CREATE TABLE コマンドの, 4-312
- PCTINCREASE パラメータ
  - STORAGE 句の, 4-520
- PCTTHRESHOLD オプション
  - ALTER TABLE コマンドの, 4-116
  - CREATE TABLE コマンドの, 4-315
- PCTUSED パラメータ
  - ALTER CLUSTER コマンドの, 4-12
  - ALTER SNAPSHOT コマンドの, 4-79

ALTER TABLE コマンドの, 4-85, 4-116  
CREATE CLUSTER コマンドの, 4-204  
CREATE SNAPSHOT コマンドの, 4-286, 4-296  
CREATE TABLE コマンドの, 4-312  
PCTVERSION パラメータ  
LOB 記憶域句の, 4-316  
physical\_attributes\_clause  
ALTER CLUSTER, 4-11  
ALTER TABLE の, 4-109  
CREATE CLUSTER の, 4-203  
CREATE TABLE の, 4-307  
PL/SQL  
関数, 3-57  
PLSQL\_V2\_COMPATABILITY オプション  
ALTER SESSION コマンドの, 4-66  
ALTER SYSTEM コマンドの, 4-96  
PM/P.M. 書式要素, 3-68  
POST\_TRANSACTION オプション  
DISCONNECT SESSION 句の  
ALTER SYSTEM コマンドの, 4-96  
POWER 数値関数, 3-21  
PRAGMA RESTRICT\_REFERENCES 句  
CREATE TYPE コマンドの, 4-145  
pragma\_clause  
ALTER TYPE の, 4-143  
CREATE TYPE, 4-343  
PRIMARY KEY オプション  
ADD 句の  
ALTER SNAPSHOT LOG コマンドの, 4-86  
DISABLE 句の, 4-376  
DROP 句の, 4-379  
ENABLE 句の, 4-416  
REFRESH 句の, 4-80, 4-287  
PRIOR 演算子, 3-15  
PRIVATE\_SGA パラメータ  
ALTER RESOURCE COST コマンド, 4-48  
procedure\_specification パラメータ  
CREATE TYPE コマンドの, 4-144  
PROFILE 句  
CREATE USER コマンドの, 4-355  
PR 書式要素, 3-62  
PUBLIC\_DEFAULT プロファイル, 4-46, 4-48  
PUBLIC オプション  
CREATE DATABASE LINK コマンドの, 4-221  
CREATE ROLLBACK SEGMENT コマンドの, 4-272  
CREATE SYNONYM コマンドの, 4-299  
DROP DATABASE LINK コマンドの, 4-383

DROP SYNONYM コマンドの, 4-401  
ENABLE 句の, 4-22  
FROM 句の, 4-472, 4-476  
TO 句の, 4-432, 4-444

## Q

QUOTA 句  
CREATE USER コマンドの, 4-355, 4-356  
CREATE USER 文内の複数の, 4-356

## R

RAWTOHEX 変換関数, 3-41  
RAW データ型, 2-13  
RDBMS(リレーショナル・データベース管理システム), 1-1  
READ ONLY オプション  
SET TRANSACTION コマンドの, 4-514  
READ WRITE オプション  
SET TRANSACTION コマンドの, 4-514  
REBUILD UNUSABLE LOCAL INDEXES オプション  
ALTER SNAPSHOT コマンドの, 4-80  
ALTER TABLE コマンドの, 4-119  
RECOVER 句, 4-466  
ALTER DATABASE コマンドの, 4-19  
parallel\_clause 「PARALLEL 句」参照  
例, 4-468  
RECYCLE オプション  
BUFFER\_POOL オプションの  
STORAGE 句の, 4-521  
REDO ログ・スレッド  
REDO ログ・ファイル・グループを削除する, 4-20  
REDO ログ・ファイル・グループを追加する, 4-19  
REDO ログ・ファイル・グループを割り当てる,  
4-212  
使用可能にする, 4-22  
使用禁止にする, 4-22  
REDO ログ・ファイル  
アーカイブ, 4-19  
アーカイブする, 4-19, 4-214, 4-217  
切り替える, 4-88, 4-102  
最大数  
データベースの, 4-214, 4-216  
メンバーの, 4-214  
指定する, 4-428  
データベースに追加する, 4-214, 4-216

- REDO ログ・ファイルのアーカイブ
  - 使用可能にする, 4-19
  - 使用禁止にする, 4-19
- REDO ログ・ファイル・グループ
  - REDO ログ・スレッドに割り当てる, 4-212
  - 最大数
    - メンバーの, 4-217
  - 削除する, 4-20
  - スレッドに追加する, 4-19
  - メンバーを削除する, 4-20
  - メンバーを追加する, 4-20
- REDO ログ・ファイル・メンバー
  - REDO ログ・ファイル・グループから削除する, 4-20
  - REDO ログ・ファイル・グループに追加する, 4-20
  - 改名する, 4-21
  - 最大数
    - REDO ログ・ファイルの, 4-214
    - REDO ログ・ファイル・グループの, 4-217
  - 指定する, 4-428
- REF, 2-21
  - DANGLING, 2-21
  - 有効範囲付き, 4-322
- REFERENCES オブジェクト権限
  - 表に対する, 4-446
- REFERENCES オプション
  - CONSTRAINT 句の, 4-194
- REFERENCING 句
  - CREATE TRIGGER コマンドの, 4-332
- REFRESH 句
  - ALTER SNAPSHOT コマンドの, 4-80
  - CREATE SNAPSHOT コマンドの, 4-287
- REFTOHEX 関数, 3-53
- REF 句
  - CREATE TYPE コマンドの, 4-346
- REF コンストラクタ
  - 式の構文, 3-81
- REF 列
  - 表に作成, 4-322
  - 表に追加する, 4-125
- REF 列の作成
  - 表に, 4-322
- RENAME FILE 句
  - ALTER DATABASE コマンドの, 4-21
- RENAME PARTITION オプション
  - ALTER SNAPSHOT コマンドの, 4-79, 4-85
  - ALTER TABLE コマンドの, 4-119
- RENAME オブジェクト監査オプション, 4-171
- RENAME オプション
  - ALTER TABLE コマンドの, 4-119
- RENAME コマンド, 4-470
  - 例, 4-470
- REPLACE オプション
  - ALTER TYPE コマンドの, 4-143
- REPLACE 文字関数, 3-28
- RESETLOGS オプション
  - ALTER DATABASE コマンドの, 4-19
  - CREATE CONTROLFILE コマンドの, 4-212
- RESIZE 句
  - ALTER DATABASE コマンドの, 4-23
- RESOURCE\_LIMIT オプション
  - ALTER SYSTEM コマンドの, 4-98
- RESOURCE 文監査のショートカット, 4-173
- RESOURCE ロール, 4-269
- returning\_clause
  - DELETE, 4-371
  - INSERT
    - INSERT コマンド
      - returning\_clause, 4-450
    - UPDATE の, 4-538
- RETURNING 句
  - DELETE コマンドの, 4-372, 4-374, 4-539
  - INSERT コマンドの, 4-451
  - UPDATE コマンドの, 4-542
  - 更新された行の取出し, 4-542
  - 削除された行の取出し, 4-374
  - 挿入された行の取出し, 4-453
- RETURNING 句の data\_item パラメータ
  - DELETE コマンドの, 4-372, 4-539
  - INSERT コマンドの, 4-451
- REUSE STORAGE オプション
  - TRUNCATE コマンドの, 4-533
- REUSE オプション
  - BACKUP CONTROLFILE 句の, 4-21
  - CREATE CONTROLFILE コマンドの, 4-211
  - filespec の, 4-429
- REVERSE オプション
  - ALTER INDEX コマンドの, 4-32
  - CREATE INDEX コマンドの, 4-237
- REVOKE コマンド, 4-472, 4-475
  - 例, 4-473, 4-478
- RNDS パラメータ
  - CREATE TYPE コマンドの, 4-145
- RNPS パラメータ

- CREATE TYPE コマンドの, 4-145
- RN 書式要素, 3-62
- ROLLBACK SEGMENT オプション
  - USING INDEX 句の, 4-286
- ROLLBACK\_SEGMENTS, 4-54
- ROLLBACK オプション
  - ADVISE 句の, 4-62
- ROLLBACK コマンド, 4-481
  - 概要, 4-8
  - トランザクションを終了する, 4-481
  - 例, 4-482
- ROUND 数値関数, 3-21
- ROUND 日付関数, 3-37, 3-38
  - 書式モデル, 3-38
- ROWID
  - 疑似列, 2-33
  - 説明, 2-16
- ROWIDTOCHAR 変換関数, 3-41
- ROWID オプション
  - ADD 句の
    - ALTER SNAPSHOT LOG コマンドの, 4-86
- ROWNUM 疑似列, 2-34
  - と ORDER BY 句, 2-35
- RPAD 文字関数, 3-28
- RR 日付書式要素, 3-68
- RTRIM 文字関数, 3-29
- RX ロック, 4-456

## S

---

- SAMPLE パラメータ
  - ANALYZE コマンドの, 4-156
- SAVEPOINT コマンド, 4-484
  - 概要, 4-8
  - 例, 4-484
- SCAN\_INSTANCES パラメータ
  - ALTER SYSTEM コマンドの, 4-95
- SCOPE IS 句
  - CREATE TABLE コマンドの, 4-311
- SCOPE 句
  - 定義済み, 4-322
- segment\_attributes\_clause
  - CREATE TABLE の, 4-307
- segment\_partition\_clause
  - ALTER TABLE の, 4-115
- SELECT
  - outer\_join, 4-504
- SELECT\_CATALOG\_ROLE ロール, 4-270
- SELECT オブジェクト監査オプション, 4-171
- SELECT オブジェクト権限
  - 順序に対する, 4-446
  - ビューに対する, 4-446
  - 表に対する, 4-446
- SELECT 句
  - INSERT コマンド, 4-452
  - UPDATE コマンド, 4-539, 4-541
- SELECT コマンド, 4-486
  - WITH\_clause, 4-489
  - 概要, 4-7
  - 例, 2-32, 4-492, 4-495, 4-496, 4-499, 4-501, 4-502, 4-527, 4-531
- SEQUEL(構造化英語問合せ言語), 1-1
- SEQ パラメータ
  - ARCHIVE LOG 句の, 4-165
- SERIALIZABLE オプション
  - ISOLATION\_LEVEL パラメータの
    - ALTER SESSION コマンドの, 4-64
- SESSION\_CACHED\_CURSORS, 4-71
- SESSION\_CACHED\_CURSORS パラメータ
  - ALTER SESSION コマンドの, 4-66
- SESSIONID オプション
  - USERENV 関数の, 3-52
- SET CONSTRAINT(S) コマンド, 4-509
- SET DATABASE パラメータ
  - CREATE CONTROLFILE コマンドの, 4-211
- SET ROLE コマンド, 4-511
  - 例, 4-513
- SET TRANSACTION コマンド, 4-514
  - 概要, 4-8
  - 例, 4-516
- set\_clause
  - ALTER SYSTEM の, 4-90
- SET 句
  - EXPLAIN PLAN コマンドの, 4-423
  - UPDATE コマンドの, 4-539
- SHARED オプション
  - CREATE DATABASE LINK コマンドの, 4-221
- SHRINK 句
  - ALTER ROLLBACK SEGMENT コマンドの, 4-54
- SINH 数値関数, 3-22
- SIN 数値関数, 3-22
- SIZE パラメータ
  - ALLOCATE EXTENT 句の, 4-12, 4-33, 4-117
  - ALTER CLUSTER コマンドの, 4-12

- CREATE CLUSTER コマンドの, 4-204, 4-207
  - filespec の, 4-429
- SKIP\_UNUSABLE\_INDEXES オプション
  - ALTER SESSION コマンドの, 4-66
- SNAPSHOT\_REFRESH\_INTERVAL, 4-290
- SNAPSHOT\_REFRESH\_KEEP\_CONNECTIONS, 4-290
- SNAPSHOT\_REFRESH\_PROCESSES, 4-290
- SOME 比較演算子, 3-5
- SOUNDEX 文字関数, 3-29
- SPECIFICATION オプション
  - COMPILE 句の
    - ALTER TYPE コマンドの, 4-143
- SPLIT PARTITION オプション
  - ALTER SNAPSHOT コマンドの, 4-79, 4-86
  - ALTER TABLE コマンドの, 4-120
- split\_partition\_clause
  - ALTER INDEX の, 4-31
  - ALTER TABLE の, 4-114
- SQL
  - 埋込み, 1-3
  - オブティマイザ, 1-2
  - 関数, 3-15, 3-73
  - 規格, 1-1
  - コマンドの概要, 4-2
  - 字句規則, 1-4
  - 統一された言語, 1-3
  - 変換関数, 3-39
  - 歴史, 1-1
- SQL(構造化照会言語), 1-1
- SQL/DS
  - データ型, 2-18
- SQL\_TRACE, 4-67
- SQL2, 1-2
- SQL-92, 1-2
- SQL 関数
  - グループ, 3-54
  - 文字, 3-24
- SQL トレース機能
  - セッションに対して使用可能および使用禁止にする, 4-66, 4-67
- SRX ロック, 4-456
- START WITH 句
  - CREATE SEQUENCE コマンドの, 4-279
  - SELECT コマンドの, 4-490, 4-494, 4-495
- START WITH パラメータ
  - REFRESH 句の, 4-80, 4-287
- STDDEV グループ関数, 3-56

- STORAGE 句, 4-518
  - ALTER CLUSTER コマンドの, 4-12
  - ALTER INDEX コマンドの, 4-32
  - ALTER TABLE コマンドの, 4-116
  - CREATE CLUSTER コマンドの, 4-204
  - CREATE INDEX コマンドの, 4-237
  - CREATE ROLLBACK SEGMENT コマンドの, 4-273
  - CREATE SNAPSHOT コマンドの, 4-286, 4-296
  - CREATE TABLE コマンドの, 4-313
  - 例, 4-522
- SUBSTR 文字関数, 3-30
- SUM グループ関数, 3-56, 3-57
- SWITCH LOGFILE オプション
  - ALTER SYSTEM コマンドの, 4-102
- SYEAR 日付書式要素, 3-68
- SYSDATE 日付関数, 3-37
- SYSTEM 表領域, 4-325, 4-328
- SYSTEM ロールバック・セグメント, 4-325
- S 書式要素, 3-62

## T

---

- table\_partition\_clause
  - CREATE TABLE の, 4-310
- table\_ref\_clause
  - ALTER TABLE の, 4-108
  - CREATE TABLE の, 4-307
- TABSPACE オプション
  - CREATE CLUSTER コマンドの, 4-204
  - CREATE INDEX コマンドの, 4-237
  - CREATE ROLLBACK SEGMENT コマンドの, 4-272
  - CREATE SNAPSHOT コマンドの, 4-286, 4-296
  - CREATE TABLE コマンドの, 4-313
  - RECOVER 句の, 4-468
- TABLE オプション
  - ANALYZE コマンドの, 4-156
  - TRUNCATE コマンドの, 4-532
  - 副問合せの, 4-527
- TANH 数値関数, 3-23
- TAN 数値関数, 3-23
- TEMPORARY オプション
  - ALTER TABLESPACE コマンドの, 4-136
- TERMINAL オプション
  - USERENV 関数の, 3-52
- THE キーワード
  - SELECT コマンドの, 4-490
  - 副問合せの, 4-527

- フラット化した副問合せ, 4-528
- THREAD パラメータ
  - ADD LOGFILE 句の, 4-19
  - ARCHIVE LOG 句の, 4-165
- TO\_CHAR 変換関数, 3-41, 3-42
  - 例, 3-60, 3-71, 3-72
- TO\_DATE 変換関数, 3-43
- TO\_MULTI\_BYTE 変換関数, 3-44
- TO\_NUMBER 変換関数, 3-44
- TO\_SINGLE\_BYTE 変換関数, 3-44
- TO 句
  - GRANT コマンドの, 4-444
  - ROLLBACK コマンドの, 4-481
- TO パラメータ
  - ARCHIVE LOG 句の, 4-165
- TRANSLATE USING 変換関数, 3-45
- TRANSLATE 文字関数, 3-31
- TRUE
  - 条件の結果, 3-84
- TRUNCATE PARTITION オプション
  - ALTER TABLE コマンドの, 4-120
- TRUNCATE コマンド, 4-532
  - 例, 4-534
- TRUNC 数値関数, 3-23
- TRUNC 日付関数, 3-38
  - 書式モデル, 3-38

## U

---

- UID 関数, 3-51
- UNARCHIVED オプション
  - CLEAR LOGFILE 句の, 4-20
- UNION ALL 集合演算子, 4-491
  - 例, 3-14
- UNION 集合演算子, 4-491
  - 例, 3-13
- UNIQUE オプション
  - DISABLE 句の, 4-376
  - ENABLE 句の, 4-416
- UNLIMITED オプション
  - ALTER PROFILE コマンドの, 4-44
  - CREATE PROFILE コマンドの, 4-264
  - QUOTA 句の, 4-355
- UNLIMITED 句
  - ALTER DATABASE コマンドの, 4-23
- UNTIL CANCEL オプション
  - RECOVER 句の, 4-468
- UNTIL CHANGE パラメータ
  - RECOVER 句の, 4-468
- UNTIL TIME パラメータ
  - RECOVER 句の, 4-468
- UNUSABLE LOCAL INDEXES オプション
  - ALTER SNAPSHOT コマンドの, 4-80
  - ALTER TABLE コマンドの, 4-119
- UPDATE オブジェクト権限
  - ビューに対する, 4-446
  - 表に対する, 4-446
- UPDATE オプション
  - CREATE TRIGGER コマンドの, 4-332
- UPDATE コマンド, 4-536, 4-541
  - returning\_clause, 4-538
  - 副問合せ, 4-539
  - 例, 4-541
- UPPER 文字関数, 3-31
- USER\_CLUSTERS ビュー, 4-160
- USER\_INDEXES ビュー, 4-159
- USER\_TAB\_COLUMNS ビュー, 4-160
- USER\_TABLES ビュー, 4-159
- USERENV 関数, 3-51
- USER 関数, 3-51
- USING INDEX オプション
  - ALTER SNAPSHOT コマンドの, 4-80
  - CONSTRAINT 句の, 4-188
  - CREATE SNAPSHOT コマンドの, 4-286
  - ENABLE 句の, 4-416
- USING MASTER ROLLBACK SEGMENT 句
  - ALTER SNAPSHOT コマンドの, 4-80
- using\_index\_clause
  - ENABLE 句の, 4-415
  - ENABLE 句
    - using\_index\_clause, 4-415
- USING 句
  - CREATE DATABASE LINK コマンドの, 4-222
- UTLEXCPT.SQL, 4-418
- UTLSAMPL.SQL, xx
- UTLXPLAN.SQL, 4-423

## V

---

- V\$LOG 表, 4-20, 4-216
- V\$NLS\_PARAMETERS 表, 4-67
- VALIDATE REF UPDATE オプション
  - ANALYZE コマンドの, 4-156
- VALIDATE オプション



ENABLE 句の, 4-415  
    CONSTRAINT 句の, 4-189  
VALUES LESS THAN 句  
    CREATE TABLE コマンドの, 4-317  
VALUES 句  
    INSERT コマンドの, 4-451, 4-452  
VALUE 演算子  
    式の構文, 3-81  
VARCHAR2 データ型, 2-8  
    RAW データ型との類似点, 2-13  
VARCHAR データ型, 2-9  
VARGRAPHIC データ型, 2-20  
VARRAY, 2-21  
VSIZE 関数, 3-52  
V 書式要素, 3-62

## W

---

WHENEVER SUCCESSFUL 句  
    AUDIT (スキーマ・オブジェクト) コマンドの, 4-176  
WHEN 句  
    CREATE TRIGGER コマンドの, 4-333  
WHERE 句  
    SELECT コマンドの, 4-490  
    UPDATE コマンドの, 4-539  
WITH CONTEXT オプション  
    CREATE FUNCTION コマンドの, 4-231  
    CREATE PROCEDURE コマンドの, 4-258  
WITH GRANT OPTION, 4-444  
WITH\_clause  
    SELECT コマンドの, 4-489  
    副問合せ, 4-526  
WNDS パラメータ  
    CREATE TYPE コマンドの, 4-145  
WNPS パラメータ  
    CREATE TYPE コマンドの, 4-145  
WORK オプション  
    COMMIT コマンドの, 4-182  
    ROLLBACK コマンドの, 4-481

## Y

---

YEAR 日付書式要素, 3-68

## あ

---

値  
    式で使用される, 3-73  
値の変換, 2-26  
    暗黙的, 2-26, 2-28  
    明示的, 2-27, 2-28  
アプリケーション・フェイルオーバー  
    「ALTER SYSTEM コマンドの DISCONNECT SESSION 句」参照

## い

---

一意キー, 4-188, 4-190  
位置の透過性  
    シノニムによる, 4-301  
インスタンス  
    最大数  
        データベースの, 4-217  
引用符  
    テキスト・リテラルで使用する, 2-2  
引用符で囲まれた名前, 2-45  
インライン LOB 記憶域, 4-116, 4-315

## う

---

埋込み SQL, 1-3  
埋込みモード  
    後続する空白を切り捨てる, 3-71

## え

---

エクステンツ  
    INITIAL サイズ, 4-519  
    MAXEXTENTS 制限, 4-520  
    表に割り当てる, 4-105  
    領域の割当て解除, 4-368  
演算子  
    NOT IN, 3-7  
    算術, 3-3  
    式で使用される, 3-73  
    集合, 3-12  
    その他, 3-15  
    定義, 3-1  
    文字, 3-3  
    論理, 3-5, 3-10

## お

---

- オーバーロード
  - プロシージャとストアド・ファンクション, 2-45
- オープンする
  - データベース, 4-18
- 大文字
  - SQL 文での重要性, 1-4
- 大文字小文字の区別
  - SQL 文で, 1-4
  - パターン・マッチング, 3-8
- 大文字と小文字
  - パターン・マッチングでの重要性, 3-8
- 大文字にする
  - 日付書式要素, 3-70
- オブジェクト型権限の取消し, 4-476
- オブジェクト型, 2-21
  - 値の比較, 2-25
  - 権限の取消し, 4-476
  - 削除する, 4-407
  - 作成, 4-342
  - 属性とメソッド、参照, 2-53
- オブジェクト型の属性とメソッド
  - 参照, 2-53
- オブジェクト型本体
  - 削除する, 4-409
  - 作成, 4-350
- オブジェクト型列の制約, 4-323
- オブジェクト監査オプション, 4-171, 4-177
- オブジェクト監査のショートカット, 4-177
- オブジェクト権限, 4-444
  - シノニムに対する, 4-447
  - 表に対する, 4-445
  - ユーザーとロールから取り消す, 4-475
  - ユーザーとロールに付与する, 4-442
- オブジェクト参照
  - 関数, 3-53
- オブジェクト表, 4-321
  - オブジェクト識別子列の索引, 4-312
  - 作成, 4-310
- オブジェクト表の作成, 4-310
- オブジェクト・ビュー
  - 作成, 4-359, 4-366
  - 例, 4-366
- オブティマイザ
  - SQL, 1-2
  - ヒント, 2-37

## か

---

- カーソル
  - セッション・キャッシュに保持する, 4-71
- 外部関数
  - 作成する, 4-228
  - 定義済み, 4-228
- 外部キー制約, 4-193
- 回復
  - 分散トランザクションに対して使用可能にする, 4-88, 4-102
  - 分散トランザクションに対して使用禁止にする, 4-88, 4-102
- 回復可能
  - 表の, 4-314
- 外部結合, 3-15, 4-504
  - 例, 4-505
- 外部プロシージャ
  - 定義, 4-255
- 改名する
  - REDO ログ・ファイル・メンバー, 4-21
  - オブジェクト, 4-470
  - データ・ファイル, 4-21, 4-131
- 型
  - ユーザー定義, 2-20
- 型コンストラクタ
  - 式の構文, 3-77
- 型の仕様部
  - コンパイルする, 4-143
- 型の本体
  - コンパイルする, 4-143
- カッコ
  - 演算子の優先順位を置き換える, 3-3
  - 式を囲む, 3-84
- 各国語サポート (NLS)
  - セッションの設定, 4-65, 4-67
- 可変長
  - 日付書式モデル, 3-71
- 関数
  - PL/SQL, 3-57
  - SQL, 3-15
  - ストアド・ファンクション, 4-228
  - ユーザー, 3-57

## き

---

- 記憶特性

- クラスタの, 4-11, 4-204
- 索引の, 4-28, 4-237
- スナップショットの, 4-79, 4-286
- スナップショット・ログの, 4-296
- 表の, 4-312, 4-313
- ロールバック・セグメントの, 4-273
- 記憶領域
  - ALTER TABLESPACE, 4-131
- 疑似列, 2-30
- 起動
  - トリガー, 4-333
- キャラクタ・セット、ASCII と EBCDIC, 2-24
- 行
  - ROWID 値で識別する, 2-34
  - ROWID によってアクセスする, 2-34
  - 更新する, 4-536
  - 順序付け, 4-498
  - 表およびビューからの削除, 4-370
  - 表およびビューに挿入する, 4-449
  - 表から選択する, 4-486
- 行アドレス
  - ROWID, 2-16
- 行外 LOB 記憶域, 4-117, 4-315
- 行の共有ロック, 4-456
- 行の排他ロック, 4-456
- 共有 SQL 領域
  - セッション・カーソル, 4-71
- 共有行排他ロック, 4-456
- 共有更新ロック, 4-456
- 共有サーバー・プロセス
  - 作成と終了, 4-99
- 共有ブール
  - 消去する, 4-97
- 共有ロック, 4-456
- 切り替える
  - REDO ログ・ファイル, 4-88, 4-102
- 切り捨てる
  - クラスタ, 4-532
  - 表, 4-532

## く

---

- 空白埋めの抑制
  - 日付書式モデルでの, 3-71
- 空白埋め比較方法, 2-23
- 区切られた名前
  - 引用符で囲まれた名前, 2-45

- 句読点
  - 日付書式モデルでの, 3-71
- 位取り
  - NUMBER 列の, 2-9
  - 負, 2-10
- クラスタ, 4-202
  - 記憶特性, 4-11, 4-204
  - 切り捨てる, 4-532
  - クラスタ索引, 4-241
  - サイズ, 4-207
  - 索引クラスタ, 4-206, 4-207
  - 削除する, 4-381
  - 作成する, 4-202
  - スナップショットの追加, 4-286
  - 定義, 4-205
  - 表の追加, 4-316
  - 表の物理記憶域, 4-205
  - 表領域の指定, 4-204
  - 表を削除する, 4-381
  - 表を追加する, 4-208
  - 変更する, 4-11
  - 列の順序, 4-206
- クラスタ・キー, 4-206
  - ～の異なる値, 4-206
- グループ
  - SQL 関数, 3-54
- クローズする
  - データベース・リンク, 4-72

## け

---

- 結合
  - 単純, 4-501
  - とクラスタ, 4-206
  - 例, 4-502, 4-505
- 結合ビュー, 4-363
- 権限, 4-444
- 権限ドメイン
  - 変更する, 4-512
- 検索する
  - 索引で行を, 4-239

## こ

---

- 更新する
  - 表およびビューの行を, 4-536
- 国際電気標準会議 (IEC), 1-1

国際標準化機構 (ISO), 1-1

コスト

SQL 文を実行する, 4-424

この, 4-188

コミットする

トランザクション, 4-182

コメント

SQL 文で, 2-35

オブジェクトから削除する, 4-180

オブジェクトに追加する, 4-180

例, 2-36

小文字

SQL 文での重要性, 1-4

小文字と大文字

スキーマ・オブジェクト名, 2-44

パターン・マッチングでの重要性, 3-8

コンパイルする

型の仕様部, 4-143

型の本体, 4-143

ストアド・ファンクション, 4-26

プロシージャ, 4-41

## さ

---

再コンパイルする

ストアド・ファンクション, 4-26

プロシージャ, 4-41

再定義する

ストアド・ファンクション, 4-230

パッケージ, 4-246, 4-251

プロシージャ, 4-257

列, 4-115

最適なサイズ

ロールバック・セグメントの, 4-273, 4-521

索引

LONG RAW データ型では禁止, 2-14

記憶特性, 4-28, 4-237

クラスタ索引, 4-241

削除する, 4-387, 4-388, 4-402

作成, 4-233

定義, 4-233

と LIKE 演算子, 3-9

ネストした表の列, 4-242

パーティション索引, 4-241

ビットマップ索引, 4-242

表あたりの複数の, 4-239

変更する, 4-28

索引クラスタ, 4-206, 4-207

索引構成表, 4-320

作成, 4-314

例, 4-320

索引を作成する

表領域の指定, 4-237

削除

表およびビューからの行の, 4-370

表から行を, 4-402

削除する

REDO ログ・ファイル・グループ, 4-20

REDO ログ・ファイル・グループからメンバーを,  
4-20

オブジェクトからコメントを, 4-180

クラスタ, 4-381

索引, 4-387, 4-388

シノニム, 4-401

順序, 4-397

ストアド・ファンクション, 4-385

スナップショット, 4-399

スナップショット・ログ, 4-400

データベース・リンク, 4-383

パッケージ, 4-389

パッケージからストアド・ファンクションを, 4-385

パッケージからプロシージャを, 4-391

パッケージ本体, 4-389

ビュー, 4-412

表, 4-402

表から整合性制約を, 4-117, 4-379

表からトリガーを, 4-406

表領域, 4-404

プロシージャ, 4-244, 4-391

プロファイル, 4-393

プロファイルからリソース制限を, 4-43

ユーザー, 4-410

ユーザーが所有するオブジェクト, 4-410

列から整合性制約を, 4-115, 4-122

ロール, 4-394

作成

オブジェクト・ビュー, 4-359, 4-366

共有サーバー・プロセス, 4-99

索引, 4-233

シノニム, 4-299

順序, 4-278

スキーマ, 4-275

スナップショット, 4-283

スナップショット・ログ, 4-294

- ディスパッチャ・プロセス (DISP), 4-99
- トリガー, 4-330
- ビュー, 4-359
- 表, 4-303
- 表領域, 4-325
- ユーザー, 4-353
- ロール, 4-268
- ロールバック・セグメント, 4-272
- 作成する
  - 外部関数, 4-228
  - クラスタ, 4-202
  - ストアド・ファンクション, 4-228
  - セーブポイント, 4-484
  - ディレクトリ, 4-226
  - データベース, 4-214
  - データベース・リンク, 4-220
  - パッケージ, 4-246, 4-250
  - プロシージャ, 4-255
  - プロファイル, 4-261

## 算術

- DATE 値での, 2-12
- 算術演算子, 3-3
- 参照キー値の削除, 4-194
- 参照整合性制約, 4-188, 4-193
  - 削除する, 4-476
  - 定義, 4-194
  - メンテナンス, 4-195
- 参照整合性制約を削除する, 4-476

## し

---

### 式, 3-73

- 条件での使用, 3-84
- 例, 3-73

### 式の構文

- VALUE 演算子, 3-81
- 属性参照, 3-82
- メソッドの起動, 3-82

### 識別子

- 名前, 2-43

### 識別子、ORACLE

- 構成, A-3

### 字句規則

- SQL, 1-4

### システム権限

- 付与する, 4-433
- ユーザーとロールから取り消す, 4-472

- ユーザーとロールに付与する, 4-432
- システム制御コマンド, 4-9
- システム変更番号
  - 強制するトランザクションを指定する, 4-183
- 自然対数, 3-19
- 実行

- トリガー, 4-333

### シノニム

- オブジェクト権限を付与する, 4-447
- 改名する, 4-470
- 監査, 4-176
- 削除する, 4-401
- 作成, 4-299
- 定義, 4-299
- データベース・リンクと使用する, 4-224
- 有効範囲, 4-301

### 集合演算子, 4-498

### 終了する

- 共有シャドウ・プロセス, 4-99
- セッション, 4-96, 4-103
- ディスパッチャ・プロセス (DISP), 4-100
- トランザクション, 4-182

### 主キー, 4-188, 4-192

- スナップショット, 4-292

### 縮小

- ロールバック・セグメント, 4-273

### 縮小する

- ロールバック・セグメント, 4-521

### 順序, 4-278

- 値間の増分, 4-56, 4-279
- 値にアクセスする, 2-30, 4-530
- 値の循環, 4-281
- 値のスキップ, 4-281
- 値の制限, 4-281
- 値の損失, 4-281
- 値を増分する, 2-30, 4-530
- 改名する, 4-470
- 削除する, 4-397
- 作成, 4-278
- 初期値を再設定する, 4-397
- パフォーマンス上の利点, 4-280
- 変更する, 4-56

### 使用可能

- 整合性制約, 4-317, 4-414
- セッションのロール, 4-511

### 使用可能にする

- REDO ログ・スレッド, 4-22

- セッションに対して SQL トレース機能を, 4-66, 4-67
- トリガー, 4-140
- 分散回復, 4-88, 4-102
- リソース制限, 4-88, 4-98, 4-265
- 使用禁止にする
  - REDO ログ・スレッド, 4-22
  - 整合性制約, 4-317, 4-375
  - セッションに対して SQL トレース機能を, 4-66, 4-67
  - セッションのロール, 4-511
  - トリガー, 4-140
  - 分散回復, 4-102
  - リソース制限, 4-88, 4-98
- 使用禁止の制約
  - 使用可能, 4-417
- 条件
  - 構文, 3-84
  - 複数の, 3-85
  - 例, 3-84
- 小数点の位置
  - 負, 2-10
- 初期化パラメータ
  - AUDIT\_TRAIL, 4-168
  - GLOBAL\_NAMES, 4-99
  - LOG\_FILES, 4-216
  - MAX\_ENABLED\_ROLES, 4-512
  - MTS\_MAX\_DISPATCHERS, 4-99
  - MTS\_MAX\_SERVERS, 4-99
  - MTS\_SERVERS, 4-99
  - NLS\_DATE\_FORMAT, 3-65
  - NLS\_DATE\_LANGUAGE, 3-68
  - NLS\_LANGUAGE, 3-68
  - NLS\_TERRITORY, 3-64, 3-65, 3-68
  - OPEN\_LINKS, 4-72, 4-222
  - OPTIMIZER\_MODE, 4-70
  - OS\_AUTHENT\_PREFIX, 4-357
  - OS\_ROLES, 4-439
  - ROLLBACK\_SEGMENTS, 4-54
  - SNAPSHOT\_REFRESH\_INTERVAL, 4-290
  - SNAPSHOT\_REFRESH\_KEEP\_CONNECTIONS, 4-290
  - SNAPSHOT\_REFRESH\_PROCESSES, 4-290
  - SQL\_TRACE, 4-67
  - THREAD, 4-22
- 書式化する
  - 数値, 3-61

- 日付値, 3-65
- 書式モデル
  - 数値, 3-61
  - 定義, 3-60
  - 日付, 3-65
  - 例, 3-60, 3-71, 3-72
- ジョブ・キュー・プロセス, 4-290
- 真理値表, 3-11

## す

---

- 数値
  - 比較規則, 2-22
  - リテラル, 2-1
- 数値書式の要素, 3-62
- 数値書式モデル, 3-61
  - 例, 3-60
- 数値データの丸め, 2-10
  - 位取りを使用して, 2-10
- スカラー
  - 定義, 4-144, 4-347
- スキーマ
  - 作成, 4-275
- スキーマ・オブジェクト
  - 定義, 2-40
  - 名前領域, 2-44
  - 命名規則, 2-43
- スキーマ・オブジェクトの命名, 2-43
- スキーマ・オブジェクト名
  - 修飾子, 2-43
- スタンバイ・データベース
  - RECOVER 句, 4-468
- ストアド・ファンクション
  - PL/SQL, 3-57
  - オーバーロード, 2-45
  - 再コンパイルする, 4-26
  - 再定義する, 4-230
  - 削除する, 4-385
  - 作成する, 4-228
  - 多重定義する, 4-247
  - 定義, 4-228
  - パッケージから削除する, 4-385
  - パッケージに追加する, 4-247
- ストアド・プロシージャ
  - プロシージャ, 4-255
- スナップショット
  - rowid, 4-292

- 記憶特性, 4-79, 4-286
- クラスタへの追加, 4-286
- コメントを削除する, 4-180
- コメントを追加する, 4-180
- 削除する, 4-399
- 作成, 4-283
- 主キー, 4-292
- 種類, 4-288
- スナップショット・ログを使ったリフレッシュ, 4-296
- 単純, 4-288
- 定義, 4-283, 4-288
- パーティション, 4-83, 4-293
- 複合, 4-289
- 変更する, 4-76
- リフレッシュ, 4-287
- リフレッシュ時間, 4-290
- リフレッシュする, 4-80
- リフレッシュ・モード, 4-289
- ロールバック・セグメント, 4-291
- スナップショット・リフレッシュ、ジョブ・キュー・プロセス, 4-290
- スナップショット・ログ
  - 記憶特性, 4-296
  - 削除する, 4-400
  - 作成, 4-294
  - 定義, 4-294
  - 変更する, 4-84
- スレッド
  - REDO ログ・スレッド, 4-216

## せ

---

- 世紀
  - 格納する, 2-12
- 整合性制約
  - CHECK, 4-188, 4-197
  - NOT NULL, 4-190
  - PRIMARY KEY, 4-188, 4-192
  - UNIQUE, 4-188, 4-190
  - 参照, 4-188, 4-193
  - 使用可能, 4-317, 4-414
  - 使用禁止にする, 4-317, 4-375
  - 定義, 4-185
  - 定義する, 4-185
  - 表から削除する, 4-117
  - 表定義, 4-115, 4-185, 4-189, 4-311

- 表に追加する, 4-115, 4-122
- 表の一部として作成, 4-311
- 列から削除する, 4-115, 4-122
- 列定義, 4-185
- 列に追加する, 4-115, 4-122
- 精度
  - NUMBER 列の, 2-9
- 制約
  - ENABLE NOVALIDATE, 4-189, 4-416
  - ENABLE VALIDATE, 4-189, 4-415
  - オブジェクト型列の, 4-323
  - 使用禁止の制約を使用可能にする, 4-417
  - 整合性制約, 4-185
  - 遅延する, 4-200
- 制約の状態, 4-200, 4-416
- 制約の妥当性検査, 4-189
- 制約の有効化および無効化, 4-416
- 制約を使用可能および使用禁止にする, 4-200
- 制約を使用可能にする
  - 主キーおよび一意キー, 4-201
- 制約を遅延する, 4-200
- 制約を有効化する, 4-417
  - 主キーと一意キー, 4-417
- セーブポイント
  - COMMIT コマンドで消去する, 4-182
  - 作成する, 4-484
- セキュリティ
  - ビューが提供する, 4-362
- セキュリティ・ドメイン, 4-150
- セッション
  - SQL トレース機能を使用可能および使用禁止にする, 4-66, 4-67
  - 各国語サポート (NLS) の設定, 4-65, 4-67
  - 終了する, 4-96, 4-103
  - 使用不可索引のエラー・レポートを使用可能および使用禁止にする, 4-66
  - 切断する, 4-96, 4-104
  - データベース・リンクをクローズする, 4-72
  - 変更する, 4-58
  - ロールを使用可能、使用禁止にする, 4-511
- セッション制御コマンド, 4-8
- セッション・カーソル, 4-71
- 切断する
  - セッション, 4-96, 4-104
- 選択リスト, 4-491

## そ

---

相関更新, 4-541  
相関副問合せ, 4-490, 4-528  
挿入する  
    表およびビューに行を, 4-449  
増分する  
    順序値, 2-30, 4-530  
属性, 2-21  
属性参照  
    式の構文, 3-82  
その他の演算子, 3-15

## た

---

対数  
    LN 数値関数, 3-19  
    LOG 数値関数, 3-20  
多重定義する  
    プロシージャとストアド・ファンクション, 4-247  
単純結合  
    例, 4-502  
単純スナップショット, 4-288  
単純スナップショットの更新, 4-287

## ち

---

直積演算, 4-503

## つ

---

追加  
    クラスタへの表の, 4-316  
    データベースに REDO ログ・ファイルを, 4-216  
    表にトリガーを, 4-330  
    表領域へのデータファイル, 4-326  
    列に整合性制約を, 4-115  
追加する  
    REDO ログ・ファイル・グループにメンバーを,  
        4-20  
    REF 表に列を, 4-125  
    オブジェクトにコメントを, 4-180  
    クラスタに表を, 4-208  
    スレッドに REDO ログ・ファイルを, 4-19  
    データベースに REDO ログ・ファイルを, 4-214  
    データベースにデータ・ファイルを, 4-214, 4-218  
    データ・ファイル, 4-135

パッケージにストアド・ファンクションを, 4-247  
パッケージにプロシージャを, 4-247  
表に整合性制約を, 4-115, 4-122  
表に列を, 4-105, 4-115, 4-121  
プロファイルにリソース制限を, 4-43, 4-261  
列に整合性制約を, 4-122

## て

---

定数  
    リテラル, 2-1  
ディスパッチャ・プロセス (DISP)  
    作成と終了, 4-99, 4-100  
ディレクトリ  
    作成する, 4-226  
    定義, 4-226  
ディレクトリ・オブジェクト  
    定義, 4-227  
ディレクトリ・パラメータ  
    AUDIT(スキーマ・オブジェクト)コマンドの,  
        4-176  
    NOAUDIT(スキーマ・オブジェクト)コマンドの,  
        4-461  
データ型, 2-5  
    ANSI, 2-18  
    DB2, 2-18  
    SQL/DS, 2-18  
    SQL 関数での変換, 3-39  
    暗黙の変換, 2-26  
    異なる値間での変換, 2-26  
    式の, 3-73  
    条件の, 3-84  
    明示的な変換, 2-27  
    要約, 2-5  
    列の～を変更する, 4-122  
    列用に指定する, 4-204, 4-311  
    列用を変更する, 4-121  
データ操作言語 (DML), 4-7  
データ定義言語 (DDL), 4-2  
データの独立性  
    シノニムによる, 4-301  
データの複雑性  
    ビューを使用して隠す, 4-362  
データベース  
    REDO ログ・ファイルのアーカイブ, 4-19  
    REDO ログ・ファイルをアーカイブする, 4-217  
    REDO ログ・ファイルを追加する, 4-214, 4-216



- オープンとクローズ, 4-18
- 最大数
  - REDO ログ・ファイルの, 4-214, 4-216
  - インスタンス, 4-217
  - データ・ファイルの, 4-214
- 作成する, 4-214
- データ・ファイルを追加する, 4-214, 4-218
- 変更する, 4-15
- マウントとディスマウント, 4-18
- ロールバック・セグメントの削除, 4-395
- データベース・オブジェクト
  - 定義, 2-40
- データベース・リンク
  - DELETE コマンドでの使用, 4-372
  - INSERT コマンドでの使用, 4-451
  - LOCK TABLE コマンドで使用する, 4-455
  - SELECT コマンドでの使用, 4-490, 4-530
  - UPDATE コマンドでの使用, 4-538
  - クローズする, 4-72
  - 削除する, 4-383
  - 作成する, 4-220
  - シノニムと使用する, 4-224
  - 定義, 4-220
- データ・ファイル
  - 改名する, 4-21, 4-131
  - 最大数
    - データベースの, 4-214
  - 指定する, 4-428
  - 追加する, 4-135
  - データベースに追加する, 4-214, 4-218
  - バックアップをとる, 4-131, 4-136
  - 表領域に追加する, 4-131
  - 表領域への追加, 4-326
- テキスト
  - 定義, 2-2
- デフォルト
  - クラスタ・キー, 4-241
- デフォルトの権限ドメイン, 4-512

## と

---

- 問合せ, 4-525
  - SELECT, 4-486
  - 例, 4-527
- 等価結合, 4-501
- 動的性能ビュー
  - V\$LOG, 4-20

## 動的性能表

- V\$LOG, 4-216
- V\$NLS\_PARAMETERS, 4-67
- トランザクション, 4-183
  - コミットする, 4-182
  - 直列化, 4-64
  - 分散, 4-183, 4-482
  - 読取り一貫性, 4-515
  - 読取り専用, 4-516
  - 読取り専用として設定する, 4-514
  - ロールバックする, 4-481
- トランザクション制御コマンド, 4-8
- トランザクションの直列化, 4-64
- トランザクションの取消し, 4-481
- トリガー
  - INSTEAD OF, 4-339
  - LONG データ型, 2-12
  - 起動, 4-333
  - 作成, 4-330
  - 実行, 4-333
  - 種類, 4-336
  - 使用可能、使用禁止にする, 4-140
  - 定義, 4-330
  - 表から削除する, 4-406
- トリガー・アクション, 4-335
- 取り消す
  - オブジェクト権限をユーザーとロールから, 4-475
  - システム権限とロールをユーザーから, 4-472

## な

---

- 内部結合, 4-503
- ナビゲーション
  - 自動, 1-2
- 名前
  - 引用符で囲まれた, 2-45
  - 小文字と大文字, 2-44
  - スキーマ・オブジェクトの, 2-43
- 名前領域
  - スキーマ・オブジェクトの, 2-44

## ね

---

- ネストされた CURSOR
  - 式の構文, 3-80
- ネストした表の格納, 4-117
  - 作成, 4-322

- ネストした表の型, 2-22
- ネストした表の列
  - 索引, 4-242
  - 表に追加する, 4-124
  - 副問合せでの識別, 4-527
- 年
  - 格納する, 2-12

## は

---

- パーティション
  - 効果的な削除, 4-374
- パーティション化
  - パーティション索引, 4-241
- パーティション索引
  - 定義, 4-233
  - 変更する, 4-34
- パーティション表
  - 作成, 4-317
- パーティション表の更新, 4-540
- パーティション表の作成, 4-317
- パーティション表の分析
  - EXPLAIN PLAN, 4-425
- パーティション・スナップショット, 4-83, 4-293
- パーティション・ビュー, 4-365
- パスワード
  - 変更する, 4-148
  - ユーザーの～を指定する, 4-268, 4-355
- パスワード履歴パラメータ, 4-45
- パターン・マッチング
  - 定義, 3-7
- パッケージ, 4-246, 4-247, 4-250
  - 再定義する, 4-246, 4-251
  - 削除する, 4-389
  - 作成する, 4-246, 4-250
  - ストアド・ファンクションを削除する, 4-385
  - ストアド・ファンクションを追加する, 4-247
  - パッケージ仕様部を作成する, 4-246
  - パッケージ本体を作成する, 4-250
  - プロシージャを削除する, 4-391
  - プロシージャを追加する, 4-247
- パッケージ仕様部, 4-246
- パッケージ本体, 4-250
  - 削除する, 4-389
- パブリック・ロールバック・セグメント, 4-272
- パラレル DML
  - セッションに対して使用可能および使用禁止にする,

- 4-62
- パラレル句
  - ALTER SNAPSHOT コマンドの, 4-80, 4-86
  - ALTER TABLE コマンドの, 4-121
- パラレル問合せ句, 4-13
  - CREATE TABLE コマンドの, 4-317
- パラレル・サーバー
  - インスタンスを設定する, 4-71

## ひ

---

- 比較演算子, 3-5
- 比較規則, 2-22
- 比較方法
  - 空白埋め, 2-23
  - 非空白埋め, 2-23
- 非空白埋め比較方法, 2-23
- 日付
  - 算術, 2-12
  - 比較規則, 2-22
- 日付書式モデル, 3-65
  - 修飾子, 3-70
  - 接尾辞, 3-70
  - デフォルト, 3-65
  - 例, 3-60, 3-72
- 日付書式要素, 3-65
  - 大文字にする, 3-70
- ビュー
  - 改名する, 4-470
  - 行の削除, 4-370
  - 行を更新する, 4-536
  - 行を挿入する, 4-449
  - 結合ビュー, 4-363
    - 更新可能な, 4-363
  - コメントを削除する, 4-180
  - コメントを追加する, 4-180
  - 固有の特性として更新可能な, 4-363
  - 再定義する, 4-412
  - 削除する, 4-412
  - 作成, 4-359
  - 使用方法, 4-362
  - 定義, 4-359
  - と DML コマンド, 4-363
  - パーティション, 4-365
  - ロックする, 4-455
- ビューへの挿入, 4-452
- 表

- LOB 記憶特性, 4-116, 4-315
- LOB 列を追加する, 4-124
- エクステントを割り当てる, 4-105
- オブジェクト権限を付与する, 4-445
- 回復可能, 4-314
- 改名する, 4-470
- 書込みを許可する, 4-105
- 書込みを禁止する, 4-105
- 記憶特性, 4-105, 4-116, 4-303, 4-312, 4-313
- 行の削除, 4-370, 4-402
- 行を更新する, 4-536
- 行を削除する, 4-532
- 行を挿入する, 4-449
- 切り捨てる, 4-532
- クラスタからの削除, 4-381
- クラスタへの追加, 4-316
- コメントを削除する, 4-180
- コメントを追加する, 4-180
- 索引構成表の作成, 4-303
- 削除する, 4-402
- 作成, 4-303
- スナップショット・ログの作成, 4-294
- 整合性制約を削除する, 4-379
- 整合性制約を追加する, 4-115, 4-122
- 定義, 4-303
- データを選択する, 4-486
- トリガーの追加, 4-330
- トリガーを削除する, 4-406
- ビューの作成, 4-359
- 表領域の指定, 4-313
- 別名, 4-490
- 変更する, 4-105
- 列を再定義する, 4-105
- 列を追加する, 4-105, 4-115, 4-121
- ロックする, 4-455
- 表示書式の変更
  - 書式モデル, 3-60
- 標準偏差, 3-56
- 表制約, 4-185
  - 例, 4-198
- 表に対して同時に更新および問合せを行う, 4-515
- 表の別名, 4-541
- 表パーティション
  - 変更する, 4-126
- 表領域, 4-325
  - SYSTEM 表領域, 4-325, 4-328
  - オンラインおよびオフラインの設定, 4-131, 4-135,

- 4-327
- クラスタを作成する, 4-204
- 索引を作成する, 4-237
- 削除する, 4-404
- 作成, 4-325, 4-328
- 将来の記憶領域割当てを変更する, 4-131
- データ・ファイル, 4-131, 4-135, 4-326
- バックアップをとる, 4-131, 4-136
- 表への指定, 4-313
- 変更する, 4-131
- ユーザーに対する表領域割当て制限の設定, 4-355, 4-356
- ユーザーに割り当てる, 4-148
- ユーザーのために一時表領域を設定する, 4-148
- ユーザーのためにデフォルトの表領域を設定する, 4-148
- ロールバック・セグメントの作成, 4-272, 4-273
- ヒント, 2-37
  - DELETE 文の, 4-372
  - SELECT 文の, 4-492
  - UPDATE 文の, 4-536

## ふ

- 複合スナップショット, 4-289
- 副問合せ, 4-525
  - CREATE VIEW, 4-360
  - DELETE, 4-371
  - WITH\_clause, 4-526
  - 相関, 4-528
  - フラット化した, 4-528
- 負の位取り, 2-10
- 付与する
  - ユーザーとロールにオブジェクト権限を, 4-442
  - ユーザーへのシステム権限およびロール, ロール, 4-432
  - ロール, 4-432
- フラット化した副問合せ
  - DELETE 文の, 4-372
  - UPDATE コマンドでの使用, 4-538
  - 使用, 4-528
  - 定義, 4-527
- プロシージャ
  - オーバーロード, 2-45
  - オブジェクト権限を付与する, 4-446
  - 再コンパイルする, 4-41
  - 再定義する, 4-255

- 削除する, 4-244, 4-391
- 作成する, 4-255
- 多重定義する, 4-247
- 定義, 4-255, 4-258
  - パッケージから削除する, 4-391
  - パッケージに追加する, 4-247
- ブロック・サイズ
  - PCTINCREASE での効果, 4-520
- プロファイル
  - DEFAULT プロファイル, 4-265, 4-355, 4-393
  - PUBLIC\_DEFAULT プロファイル, 4-46, 4-48
  - 削除する, 4-393
  - 作成する, 4-261
  - 定義, 4-261, 4-264
  - 変更する, 4-43
  - ユーザーに割り当てる, 4-148
  - ユーザーへの割当て, 4-355
  - リソース制限を追加する, 4-43, 4-261
  - リソース制限を変更する, 4-43
  - リソース制限を削除する, 4-43
- 文監査オプション, 4-169
- 文監査のショートカット, 4-173
- 分散回復
  - 使用禁止にする, 4-88, 4-102
  - シングル・プロセス環境で使用可能にする, 4-88, 4-102
- 分散問合せ, 4-530
  - 制限, 4-530
  - 例, 4-531
- 分散トランザクション, 4-183, 4-482

## へ

---

- 米国規格協会 (ANSI), 1-1
- 米国連邦情報・技術局 (NIST), 1-2
- 別名
  - 表, 4-541
- 変換
  - 文字列から日付へ, 3-72
- 変更する
  - コストベースの最適化の方法の目標, 4-70
  - 最適化の方法, 4-70
  - 索引, 4-31
  - 順序, 4-56
  - スナップショット, 4-76
  - スナップショット・ログ, 4-84
  - データベース, 4-15

- パスワード, 4-148
- 表, 4-105
- プロファイル, 4-43
- リソース制限, 4-43
- リソースのコスト, 4-48
- 列定義, 4-105, 4-115, 4-121

## ま

---

- マウントする
  - データベース, 4-18
- マルチスレッド・サーバー
  - プロセスを管理する, 4-95, 4-99

## め

---

- 命名
  - データベース・オブジェクト, A-3
- メソッド, 2-21
- メソッドの起動
  - 式の構文, 3-82

## も

---

- 文字
  - SQL 関数, 3-24
  - 演算子, 3-3
  - データ型, 2-7
  - 比較規則, 2-22
  - リテラル, 2-1
- 文字列から日付への変換, 3-72

## ゆ

---

- ユーザー
  - 一時表領域を設定する, 4-148
  - オブジェクト権限を取り消す, 4-475
  - オブジェクト権限を付与する, 4-442
  - 削除する, 4-410
  - 作成, 4-353
  - システム権限とロールを取り消す, 4-472
  - システム権限とロールを付与する, 4-432
  - 定義, 4-353
  - デフォルトの表領域を設定する, 4-148
  - デフォルト・ロールを設定する, 4-148, 4-150
  - 認証方式を変更する, 4-148
  - パスワードを指定する, 4-268

- パスワードを設定する, 4-355
- パスワードを変更する, 4-148
- 表領域割当て制限の設定, 4-355, 4-356
- 表領域を割り当てる, 4-148
- プロファイルの割当て, 4-355
- プロファイルを割り当てる, 4-148
- 変更する, 4-148
- リソース制限を割り当てる, 4-148
- ユーザー・アクセス検証
  - セキュリティ・ドメイン, 4-150
- ユーザー関数, 3-57
  - 式の構文, 3-76
- ユーザー定義型, 2-20
  - REF, 2-21
  - VARRAY, 2-21
  - オブジェクト型, 2-21
  - ネストした表, 2-22
- ユーザー認証
  - 変更する, 4-148
- ユーザー関数, 4-228
- 優先順位
  - 定義, 3-2
- ユリウス暦日付, 2-13

## よ

---

- 読取り一貫性
  - デフォルト, 4-515

## り

---

- リソース制限
  - 超える, 4-264
  - 使用可能にする, 4-88, 4-98, 4-265
  - 使用禁止にする, 4-88, 4-98
  - プロファイルから削除する, 4-43
  - プロファイルに追加する, 4-43, 4-261
  - 変更する, 4-43
  - ユーザーに割り当てる, 4-148
  - リソースのコスト, 4-48
- リテラル
  - 数値, 2-1
  - 定義, 2-1
  - 文字, 2-1
- リフレッシュ
  - スナップショット, 4-287
  - スナップショット・ログを使ったスナップショット,

- 4-296
- リフレッシュ時間
  - スナップショット, 4-290
- リフレッシュする
  - スナップショット, 4-80
- リフレッシュ・モード
  - スナップショット, 4-289
- リモート問合せ, 4-530
- リモート表
  - 識別する, 4-530
- 領域
  - 割当て解除, 4-368
- 領域の割当て解除, 4-118
- リンク
  - データベース・リンク, 4-220

## れ

---

- 例
  - コメントの, 2-36
- 列
  - 改名する, 4-471
  - 疑似列, 2-30
  - クラスタ・キーの, 4-206
  - コメントを削除する, 4-180
  - コメントを追加する, 4-180
  - 再定義する, 4-105, 4-115, 4-121
  - 索引における順序, 4-239
  - 索引における最大数, 4-239
  - 整合性制約を削除する, 4-115, 4-122
  - 整合性制約を追加する, 4-115, 4-122
  - 定義, 4-310
  - データ型の指定, 4-311
  - データ型を変更する, 4-115, 4-121, 4-122
  - デフォルト値を変更する, 4-115, 4-122
  - 表から選択する, 4-486
  - 表とスキーマを使用して名前を修飾する, 4-491
  - 表内での最大数, 4-310
  - 表に追加する, 4-105, 4-115, 4-121
- 列制約, 4-185
- 連邦情報処理標準 (FIPS), 1-2

## ろ

---

- ロール
  - CONNECT ロール, 4-269
  - DBA ロール, 4-270

DELETE\_CATALOG\_ROLE ロール, 4-270  
EXECUTE\_CATALOG\_ROLE ロール, 4-270  
Oracle8 によって定義されている, 4-269  
RESOURCE ロール, 4-269  
SELECT\_CATALOG\_ROLE ロール, 4-270  
オブジェクト権限を取り消す, 4-475  
オブジェクト権限を付与する, 4-442  
削除する, 4-394  
作成, 4-268  
システム権限とロールを取り消す, 4-472  
システム権限とロールを付与する, 4-432  
セッションに対して使用可能、使用禁止にする,  
4-511  
定義, 4-268  
付与する, 4-433  
ユーザーとロールから取り消す, 4-472  
ユーザーとロールに付与する, 4-432  
ユーザーのためのデフォルト・ロールを設定する,  
4-148, 4-150  
ロールバック  
同一セーブポイントへの複数の, 4-482  
ロールバックする  
トランザクション, 4-481  
ロールバック・セグメント  
SYSTEM ロールバック・セグメント, 4-325  
オンラインおよびオフラインの設定, 4-54, 4-395  
記憶特性, 4-273  
サイズの縮小, 4-273  
サイズを縮小する, 4-54, 4-521  
最適なサイズ, 4-273, 4-521  
削除する, 4-395  
作成, 4-272  
スナップショット, 4-291  
定義, 4-272  
表領域の指定, 4-272, 4-273  
変更する, 4-53  
ロック  
COMMIT コマンドで解除する, 4-182  
ROLLBACK 文による解除, 4-482  
種類, 4-456  
と問合せ, 4-456  
排他, 4-456  
表, 4-456  
複数の, 4-456  
ロックする  
表およびビュー, 4-455  
論理演算子

条件での使用, 3-84  
定義, 3-10

## わ

---

ワイルド・カード文字  
パターン・マッチング, 3-7, 3-9  
割り当てる  
表にエクステントを, 4-105