

Oracle9i

SQL リファレンス

リリース 1 (9.0.1)

2001 年 10 月

部品番号 : J04117-01

ORACLE®

Oracle9i SQL リファレンス , リリース 1 (9.0.1)

部品番号 : J04117-01

原本名 : Oracle9i SQL Reference, Release 1 (9.0.1)

原本部品番号 : A90126-01 (Vol.1)、A90127-01 (Vol.2)、A90128-01 (Vol.3)

原本著者 : Diana Lorentz

原本協力者 : Nipun Agarwal, David Alpern, Patrick Amor, Rick Anderson, Geeta Arora, Nimar Arora, Lance Ashdown, Cathy Baird, Sandeepan Banerjee, Cailein Barclay, Subhransu Basu, Mark Bauer, Ruth Baylis, Harmeek Bedi, Barbara Benton, Paula Bingham, Steve Bobrowski, Tolga Bozkaya, Allen Brumm, Bridget Burke, Ted Burroughs, Greg Casbolt, Sivasankaran Chandrasekar, Thomas Chang, Eugene Chong, Greg Cook, Michele Cyran, Ravindra Dani, Dinesh Das, Mary Ann Davidson, Norbert Debes, Connie Dialeris, Alan Downing, Amit Ganesh, Bill Gietz, Govind Govindarajan, Ray Guzman, John Haydu, Shelley Higgins, Thuvan Hoang, Wei Hu, Jiansheng Huang, Alexander Hunold, Bob Jenkins, Mark Johnson, Nitin Karkhanis, Vishy Karra, Jonathan Klein, Susan Kotsovolos, Vishu Krishnamurthy, Ramkumar Krishnan, Muralidhar Krishnaprasad, Paul Lane, Simon Law, Shilpa Lawande, Seong Yong Albert Lee, Bill Lee, Yunrui Li, Li-Sen Liu, Shih-Hao Liu, Jing Liu, Phil Locke, Lenore Luscher, Kevin MacDowell, Steve McGee, Colin McGregor, Jack Melnick, Ben Meng, Magdi Morsi, Ari Mozes, Sreedhar Mukkamalla, Subramanian Muralidhar, Ravi Murthy, Sujatha Muthulingam, Gary Ngai, Ron Obermarck, Jeffrey Olkin, Kevin Osinski, Ananth Raghavan, Jack Raitto, Den Raphaely, Siva Ravada, John Russell, Vivian Schupmann, Ajay Sethi, Lei Sheng, Wayne Smith, Ekrem Soylemez, Jagannathan Srinivasan, Jim Stenoish, Mike Stewart, Seema Sundara, Ashish Thusoo, Rosanne Park Toohey, Anh-Tuan Tran, Kothanda Umamageswaran, Randy Urbano, Sandy Venning, Andre Vergison, Steve Vivian, Eric Voss, Rick Wessman, Daniel Wong, Aravind Yalamanchi, Adiel Yoaz, Qin Yu, Mohamed Zait, Fred Zemke, Mohamed Ziauddin

Copyright © 1996, 2001, Oracle Corporation. All rights reserved.

Printed in Japan.

制限付権利の説明

プログラム (ソフトウェアおよびドキュメントを含む) 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。著作権、特許権およびその他の知的財産権に関する法律により保護されています。

当プログラムのリバース・エンジニアリング等は禁止されております。

このドキュメントの情報は、予告なしに変更されることがあります。オラクル社は本ドキュメントの無謬性を保証しません。

*オラクル社とは、Oracle Corporation (米国オラクル) または日本オラクル株式会社 (日本オラクル) を指します。

危険な用途への使用について

オラクル社製品は、原子力、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションに用途として開発されておりません。オラクル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ください。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、日本オラクル株式会社と開発元である Oracle Corporation (米国オラクル) およびその関連会社は一切責任を負いかねます。当プログラムを米国防総省の米国政府機関に提供する際には、『Restricted Rights』と共に提供してください。この場合次の Notice が適用されます。

Restricted Rights Notice

Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

このドキュメントに記載されているその他の会社名および製品名は、あくまでその製品および会社を識別する目的にのみ使用されており、それぞれの所有者の商標または登録商標です。

目次

はじめに	xv
------------	----

SQL リファレンスでの新規事項	xxiii
------------------------	-------

Vol.1

1 概要

SQL の歴史	1-2
SQL 規格	1-2
SQL の特長	1-3
すべてのリレーショナル・データベースに共通の言語	1-3
埋込み SQL	1-4
字句規則	1-5
Oracle のツール製品のサポート	1-5

2 Oracle SQL の基本要素

データ型	2-2
Oracle の組込みデータ型	2-7
文字データ型	2-9
NUMBER データ型	2-12
日時および期間データ型	2-17
ラージ・オブジェクト (LOB) データ型	2-25
ANSI、DB2、SQL/DS のデータ型	2-34
ユーザー定義型	2-36

Oracle が提供する型	2-38
Any 型	2-38
XML 型	2-39
空間型	2-41
メディア型	2-42
データ型の比較規則	2-42
データ変換	2-46
リテラル	2-51
テキスト・リテラル	2-52
整数リテラル	2-53
数値リテラル	2-53
期間リテラル	2-55
書式モデル	2-59
数値書式モデル	2-61
日付書式モデル	2-66
書式モデルの修飾子	2-73
文字列から日付への変換に関する規則	2-75
XML 書式モデル	2-76
NULL	2-77
SQL ファンクションでの NULL	2-77
比較条件での NULL	2-77
条件での NULL	2-78
疑似列	2-79
CURRVAL と NEXTVAL	2-79
LEVEL	2-82
ROWID	2-82
ROWNUM	2-83
コメント	2-85
SQL 文中のコメント	2-85
スキーマ・オブジェクトに関するコメント	2-86
ヒント	2-86
データベース・オブジェクト	2-102
スキーマ・オブジェクト	2-102
非スキーマ・オブジェクト	2-103
スキーマ・オブジェクトの部分	2-103

スキーマ・オブジェクト名および修飾子	2-106
スキーマ・オブジェクトのネーミング規則	2-106
スキーマ・オブジェクトのネーミング例	2-109
スキーマ・オブジェクト名のネーミング計画	2-110
スキーマ・オブジェクトの構文および SQL 文の構成要素	2-110
Oracle によるスキーマ・オブジェクト参照の変換方法	2-111
他のスキーマ内のオブジェクトの参照	2-112
リモート・データベース内のオブジェクトの参照	2-113
オブジェクト型の属性とメソッドの参照	2-116

3 演算子

SQL 演算子	3-2
単項演算子およびバイナリ演算子	3-2
演算子の優先順位	3-2
算術演算子	3-3
連結演算子	3-4
集合演算子	3-5
ユーザー定義演算子	3-6

4 式

SQL 式	4-2
単純式	4-4
複合式	4-5
CASE 式	4-6
CURSOR 式	4-8
日時式	4-10
ファンクション式	4-11
期間式	4-12
オブジェクト・アクセス式	4-12
スカラー副問合せ式	4-13
型コンストラクタ式	4-14
変数式	4-15
式のリスト	4-16

5 条件

SQL 条件	5-2
条件の種類	5-2
条件の優先順位	5-4
比較条件	5-4
単純比較条件	5-6
グループ比較条件	5-7
論理条件	5-8
メンバーシップ条件	5-10
範囲条件	5-11
NULL 条件	5-12
EXISTS 条件	5-12
LIKE 条件	5-13
IS OF <i>type</i> 条件	5-17
複合条件	5-18

6 ファンクション

SQL ファンクション	6-2
単一行ファンクション	6-3
集計ファンクション	6-6
分析ファンクション	6-8
オブジェクト参照ファンクション	6-13
SQL ファンクションのリスト (アルファベット順)	6-14
ABS	6-14
ACOS	6-14
ADD_MONTHS	6-15
ASCII	6-16
ASCIISTR	6-17
ASIN	6-17
ATAN	6-18
ATAN2	6-19
AVG	6-19
BFILENAME	6-21
BIN_TO_NUM	6-22
BITAND	6-23
CAST	6-24

CEIL	6-27
CHARTOROWID	6-27
CHR	6-28
COALESCE	6-29
COMPOSE	6-31
CONCAT	6-31
CONVERT	6-32
CORR	6-34
COS	6-36
COSH	6-36
COUNT	6-37
COVAR_POP	6-39
COVAR_SAMP	6-41
CUME_DIST	6-43
CURRENT_DATE	6-44
CURRENT_TIMESTAMP	6-45
DBTIMEZONE	6-46
DECODE	6-47
DECOMPOSE	6-48
DENSE_RANK	6-49
DEREF	6-51
DUMP	6-52
EMPTY_BLOB、EMPTY_CLOB	6-54
EXISTSNODE	6-54
EXP	6-55
EXTRACT (日時)	6-56
EXTRACT (XML)	6-57
FIRST	6-58
FIRST_VALUE	6-60
FLOOR	6-62
FROM_TZ	6-62
GREATEST	6-63
GROUP_ID	6-64
GROUPING	6-65
GROUPING_ID	6-66
HEXTORAW	6-68
INITCAP	6-69
INSTR	6-70

LAG	6-71
LAST	6-73
LAST_DAY	6-75
LAST_VALUE	6-76
LEAD	6-78
LEAST	6-79
LENGTH	6-80
LN	6-81
LOCALTIMESTAMP	6-81
LOG	6-82
LOWER	6-83
LPAD	6-83
LTRIM	6-84
MAKE_REF	6-85
MAX	6-86
MIN	6-88
MOD	6-89
MONTHS_BETWEEN	6-90
NCHR	6-91
NEW_TIME	6-92
NEXT_DAY	6-93
NLS_CHARSET_DECL_LEN	6-93
NLS_CHARSET_ID	6-94
NLS_CHARSET_NAME	6-95
NLS_INITCAP	6-95
NLS_LOWER	6-97
NLSSORT	6-97
NLS_UPPER	6-99
NTILE	6-99
NULLIF	6-101
NUMTODSINTERVAL	6-102
NUMTOYMINTERVAL	6-103
NVL	6-104
NVL2	6-105
PERCENT_RANK	6-106
PERCENTILE_CONT	6-108
PERCENTILE_DISC	6-110
POWER	6-111

RANK	6-112
RATIO_TO_REPORT	6-114
RAWTOHEX	6-115
RAWTONHEX	6-116
REF	6-116
REFTOHEX	6-117
REGR_ (線形リグレーション) ファンクション	6-118
REPLACE	6-126
ROUND (数値)	6-127
ROUND (日付)	6-128
ROW_NUMBER	6-128
ROWIDTOCHAR	6-130
ROWIDTONCHAR	6-130
RPAD	6-131
RTRIM	6-131
SESSIONTIMEZONE	6-132
SIGN	6-133
SIN	6-133
SINH	6-134
SOUNDEX	6-134
SQRT	6-136
STDDEV	6-136
STDDEV_POP	6-138
STDDEV_SAMP	6-139
SUBSTR	6-141
SUM	6-142
SYS_CONNECT_BY_PATH	6-144
SYS_CONTEXT	6-145
SYS_DBURIGEN	6-150
SYS_EXTRACT_UTC	6-151
SYS_GUID	6-152
SYS_TYPEID	6-153
SYS_XMLAGG	6-154
SYS_XMLGEN	6-155
SYSDATE	6-156
SYSTIMESTAMP	6-157
TAN	6-158
TANH	6-158

TO_CHAR (文字)	6-159
TO_CHAR (日時)	6-160
TO_CHAR (数値)	6-162
TO_CLOB	6-163
TO_DATE	6-164
TO_DSINTERVAL	6-165
TO_LOB	6-166
TO_MULTI_BYTE	6-167
TO_NCHAR (文字)	6-168
TO_NCHAR (日時)	6-169
TO_NCHAR (数値)	6-169
TO_NCLOB	6-170
TO_NUMBER	6-171
TO_SINGLE_BYTE	6-172
TO_TIMESTAMP	6-172
TO_TIMESTAMP_TZ	6-173
TO_YMINTERVAL	6-174
TRANSLATE	6-175
TRANSLATE ...USING	6-176
TREAT	6-178
TRIM	6-179
TRUNC (数値)	6-180
TRUNC (日付)	6-181
TZ_OFFSET	6-182
UID	6-183
UNISTR	6-183
UPPER	6-184
USER	6-185
USERENV	6-185
VALUE	6-187
VAR_POP	6-188
VAR_SAMP	6-189
VARIANCE	6-191
VSIZE	6-192
WIDTH_BUCKET	6-193
ROUND および TRUNC 日付ファンクション	6-195
ユーザー定義ファンクション	6-196
前提条件	6-197

名前の優先順位	6-197
---------------	-------

7 SQL 問合せおよびその他の SQL 文

問合せおよび副問合せ	7-2
単純な問合せの作成	7-3
階層問合せ	7-3
UNION [ALL]、INTERSECT および MINUS 演算子	7-7
問合せ結果のソート	7-9
結合	7-9
副問合せの使用	7-12
ネストされた副問合せのネスト解除	7-13
DUAL 表からの選択	7-14
分散問合せ	7-15
様々な種類の SQL 文	7-16

Vol.2

8 SQL 文 : ALTER CLUSTER ~ ALTER SEQUENCE

ALTER CLUSTER	8-3
ALTER DATABASE	8-9
ALTER DIMENSION	8-43
ALTER FUNCTION	8-46
ALTER INDEX	8-48
ALTER INDEXTYPE	8-69
ALTER JAVA	8-71
ALTER MATERIALIZED VIEW	8-74
ALTER MATERIALIZED VIEW LOG	8-90
ALTER OUTLINE	8-97
ALTER PACKAGE	8-99
ALTER PROCEDURE	8-103
ALTER PROFILE	8-106
ALTER RESOURCE COST	8-110
ALTER ROLE	8-113
ALTER ROLLBACK SEGMENT	8-115
ALTER SEQUENCE	8-119

9 SQL 文 : ALTER SESSION ~ ALTER SYSTEM

ALTER SESSION	9-2
初期化パラメータおよび ALTER SESSION	9-7
セッション・パラメータおよび ALTER SESSION	9-10
ALTER SYSTEM	9-19
初期化パラメータおよび ALTER SYSTEM	9-32

10 SQL 文 : ALTER TABLE ~ ALTER TABLESPACE

ALTER TABLE	10-2
ALTER TABLESPACE	10-82

11 SQL 文 : ALTER TRIGGER ~ *constraint_clause*

ALTER TRIGGER	11-2
ALTER TYPE	11-6
ALTER USER	11-20
ALTER VIEW	11-28
ANALYZE	11-31
ASSOCIATE STATISTICS	11-46
AUDIT	11-50
CALL	11-63
COMMENT	11-66
COMMIT	11-69
<i>constraint_clause</i>	11-72

12 SQL 文 : CREATE CLUSTER ~ CREATE JAVA

CREATE CLUSTER	12-2
CREATE CONTEXT	12-11
CREATE CONTROLFILE	12-14
CREATE DATABASE	12-20
CREATE DATABASE LINK	12-33
CREATE DIMENSION	12-39
CREATE DIRECTORY	12-44
CREATE FUNCTION	12-47
CREATE INDEX	12-58
CREATE INDEXTYPE	12-84
CREATE JAVA	12-86

13 SQL 文 : CREATE LIBRARY ~ CREATE SPFILE

CREATE LIBRARY	13-2
CREATE MATERIALIZED VIEW	13-5
CREATE MATERIALIZED VIEW LOG	13-29
CREATE OPERATOR	13-38
CREATE OUTLINE	13-42
CREATE PACKAGE	13-46
CREATE PACKAGE BODY	13-51
CREATE PFILE	13-56
CREATE PROCEDURE	13-58
CREATE PROFILE	13-65
CREATE ROLE	13-72
CREATE ROLLBACK SEGMENT	13-75
CREATE SCHEMA	13-79
CREATE SEQUENCE	13-81
CREATE SPFILE	13-86

14 SQL 文 : CREATE SYNONYM ~ CREATE TRIGGER

CREATE SYNONYM	14-2
CREATE TABLE	14-6
CREATE TABLESPACE	14-62
CREATE TEMPORARY TABLESPACE	14-73
CREATE TRIGGER	14-77

15 SQL 文 : CREATE TYPE ~ DROP ROLLBACK SEGMENT

CREATE TYPE	15-3
CREATE TYPE BODY	15-24
CREATE USER	15-30
CREATE VIEW	15-37
DELETE	15-49
DISASSOCIATE STATISTICS	15-57
DROP CLUSTER	15-60
DROP CONTEXT	15-62
DROP DATABASE LINK	15-63

DROP DIMENSION	15-65
DROP DIRECTORY	15-67
DROP FUNCTION	15-69
DROP INDEX	15-71
DROP INDEXTYPE	15-73
DROP JAVA	15-75
DROP LIBRARY	15-77
DROP MATERIALIZED VIEW	15-78
DROP MATERIALIZED VIEW LOG	15-80
DROP OPERATOR	15-82
DROP OUTLINE	15-84
DROP PACKAGE	15-85
DROP PROCEDURE	15-87
DROP PROFILE	15-89
DROP ROLE	15-91
DROP ROLLBACK SEGMENT	15-92

16 SQL 文 : DROP SEQUENCE ~ ROLLBACK

DROP SEQUENCE	16-2
DROP SYNONYM	16-4
DROP TABLE	16-6
DROP TABLESPACE	16-9
DROP TRIGGER	16-12
DROP TYPE	16-14
DROP TYPE BODY	16-17
DROP USER	16-19
DROP VIEW	16-21
EXPLAIN PLAN	16-23
<i>filespec</i>	16-27
GRANT	16-31
INSERT	16-56
LOCK TABLE	16-72
MERGE	16-76
NOAUDIT	16-80
RENAME	16-85
REVOKE	16-87
ROLLBACK	16-96

17 SQL 文 : SAVEPOINT ~ UPDATE

SAVEPOINT	17-2
SELECT	17-4
SET CONSTRAINT[S]	17-41
SET ROLE	17-43
SET TRANSACTION	17-46
<i>storage_clause</i>	17-50
TRUNCATE	17-59
UPDATE	17-64

A 構文図の読み方

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

B Oracle と標準 SQL

ANSI 規格	B-2
ISO 規格	B-2
Oracle の規格準拠	B-3
FIPS の規格準拠	B-10
標準 SQL に対する Oracle 拡張機能	B-11
キャラクタ・セットのサポート	B-12

C Oracle の予約語

索引

はじめに

このマニュアルでは、Oracle のデータベースの情報を管理するために使用される Structured Query Language (SQL) について説明します。Oracle SQL は、ANSI (米国規格協会) および ISO (国際標準化機構) の SQL99 規格にさらに多くの内容を盛り込んでいます。

この章では、次の内容を説明します。

- [対象読者](#)
- [構成](#)
- [関連ドキュメント](#)
- [表記規則](#)

対象読者

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

構成

このマニュアルの構成は次のとおりです。

Vol.1

第1章「概要」

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

第2章「Oracle SQL の基本要素」

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

第3章「演算子」

SQL の演算子について説明します。

第4章「式」

SQL の式について説明します。

第5章「条件」

SQL の条件について説明します。

第6章「ファンクション」

SQL ファンクションの使用方法について説明します。

第7章「SQL 問合せおよびその他の SQL 文」

様々な型の SQL 問合せについて説明し、様々な型の SQL 文を示します。

Vol.2

第8章「SQL 文 : ALTER CLUSTER ～ ALTER SEQUENCE」

第9章「SQL 文 : ALTER SESSION ～ ALTER SYSTEM」

第10章「SQL 文 : ALTER TABLE ～ ALTER TABLESPACE」

第 11 章「SQL 文 : ALTER TRIGGER ~ constraint_clause」

第 12 章「SQL 文 : CREATE CLUSTER ~ CREATE JAVA」

Vol.3

第 13 章「SQL 文 : CREATE LIBRARY ~ CREATE SPFILE」

第 14 章「SQL 文 : CREATE SYNONYM ~ CREATE TRIGGER」

第 15 章「SQL 文 : CREATE TYPE ~ DROP ROLLBACK SEGMENT」

第 16 章「SQL 文 : DROP SEQUENCE ~ ROLLBACK」

第 17 章「SQL 文 : SAVEPOINT ~ UPDATE」

すべての SQL 文をアルファベット順に説明します。

付録 A「構文図の読み方」

このマニュアルでの構文図の読み方について説明します。

付録 B「Oracle と標準 SQL」

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

付録 C「Oracle の予約語」

Oracle 内部で使用する予約語を示します。

Oracle9i リリース 1 (9.0.1) マニュアルでの構成変更

前回のマニュアルの式、条件および問合せの章は分割されました。条件および式については 2 つの章に分割され、問合せについては、第 7 章「SQL 問合せおよびその他の SQL 文」で説明しています。

前回のマニュアルでは、式の書式として記載されていた CAST、DECODE および EXTRACT (日時) は、SQL 組込みファンクションとして記載されています。

LIKE および、前回のマニュアルで「比較演算子」および「論理演算子」と呼ばれていた要素は、SQL の条件として記載されています。

すべての SQL 文を含む章 (前回のマニュアルの第 7 章～第 10 章) は、10 個の章に分割されました。

Oracle8iのマニュアルでの構成変更

すべての SQL 文を含む章（前回のマニュアルの第 7 章）は、4 つの章に分割されました。

リリース 8.0 のマニュアルを使用していたユーザーは、次の項が移動またはタイトルが変更されているため注意してください。

- 「書式モデル」の項は、第 2 章（2-59 ページ）に移動しました。

- 第 3 章は、次の章に分割されました。

- 第 3 章「演算子」
- 第 6 章「ファンクション」
- 第 4 章「式」
- 第 5 章「条件」
- 第 7 章「SQL 問合せおよびその他の SQL 文」

この章の 7-2 ページの「問合せおよび副問合せ」は、17-4 ページの **SELECT** の構文および比較情報の背景について説明します。

- 新規の章、第 8 章「SQL 文」は、特定のタスクに対する適切な SQL 文を見つけやすくするために追加されました。
- `archive_log_clause` は、9-19 ページの「**ALTER SYSTEM**」に含まれています。
- `deallocate_unused_clause` は、10-2 ページの「**ALTER TABLE**」、8-3 ページの「**ALTER CLUSTER**」および 8-48 ページの「**ALTER INDEX**」に含まれています。
- `disable_clause` は、14-6 ページの「**CREATE TABLE**」および 10-2 ページの「**ALTER TABLE**」に含まれています。
- `drop_clause` は、**ALTER TABLE** 文の `drop_constraint_clause` として記載されています（**ALTER TABLE** 文の `drop_column_clause` と区別するため）。10-2 ページの「**ALTER TABLE**」を参照してください。
- `enable_clause` は、14-6 ページの「**CREATE TABLE**」および 10-2 ページの「**ALTER TABLE**」に含まれています。
- `parallel_clause` は、単純化され、関連のある様々な文に含まれています。
- `recover_clause` は、**ALTER DATABASE** 文に含まれています。これは、`recover_clause` のリカバリ機能が向上し、**ALTER DATABASE** 文を介して常に実行されるためです。8-9 ページの「**ALTER DATABASE**」を参照してください。

- 「スナップショット」および「スナップショット・ログ」の項は、移動され、タイトルが変更されています。スナップショット機能が大幅に向上し、これらのオブジェクトは **マテリアライズド・ビュー** と呼ばれます。13-5 ページの「**CREATE MATERIALIZED VIEW**」、8-74 ページの「**ALTER MATERIALIZED VIEW**」、15-78 ページの「**DROP MATERIALIZED VIEW**」、13-29 ページの「**CREATE MATERIALIZED VIEW LOG**」、8-90 ページの「**ALTER MATERIALIZED VIEW LOG**」および 15-80 ページの「**DROP MATERIALIZED VIEW LOG**」を参照してください。
- 「**副問合せ**」の項は、「**SELECT**」の項と結合されました。17-4 ページの「**SELECT**」を参照してください。
- SQL 文 `GRANT object_privileges` および `GRANT system_privileges_and_roles` は、`GRANT` 文に結合されました。16-31 ページの「**GRANT**」を参照してください。
- SQL 文 `REVOKE schema_object_privileges` および `REVOKE system_privileges_and_roles` は、`REVOKE` 文に結合されました。16-87 ページの「**REVOKE**」を参照してください。
- SQL 文 `AUDIT sql_statements` および `AUDIT schema_objects` は、`AUDIT` 文に結合されました。11-50 ページの「**AUDIT**」を参照してください。
- SQL 文 `NOAUDIT sql_statements` および `NOAUDIT schema_objects` は、`NOAUDIT` 文に結合されました。16-80 ページの「**NOAUDIT**」を参照してください。

関連ドキュメント

詳細は、次の Oracle ドキュメントを参照してください。

- SQL に対する Oracle 手続き型言語拡張機能である 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。
- Oracle 埋込み SQL の詳細は、『Pro*C/C++ Precompiler プログラマーズ・ガイド』、『Programmer's Guide to SQL*Module for Ada』および『Pro*COBOL Precompiler プログラマーズ・ガイド』を参照してください。

このマニュアルで示すほとんどの例は、Oracle のインストール時にデフォルトでインストールされる、シード・データベースのサンプル・スキーマを使用します。これらのスキーマの作成方法および使用方法については、『Oracle9i サンプル・スキーマ』を参照してください。

リリース・ノート、インストール・マニュアル、ホワイトペーパーまたはその他の関連書籍は、Oracle Technology Network (OTN) に接続すれば、無償でダウンロードできます。OTN を利用するには、オンライン登録をする必要があります。次の URL で登録できます。

<http://otn.oracle.co.jp/membership/>

すでに OTN のユーザー名およびパスワードを所有している場合は、OTN の次の Web サイトのドキュメント・セクションを参照できます。

<http://otn.oracle.co.jp/document/>

表記規則

この項では、このマニュアルの本文およびコード例で使用される表記規則について説明します。この項の内容は次のとおりです。

- [本文中の表記規則](#)
- [コード例中の表記規則](#)

本文中の表記規則

本文では、特別な用語をより迅速に識別できるように、様々な表記規則を使用します。次の表に、それらの表記規則を説明し、その使用例を示します。

規則	意味	例
太字	太字は、本文中で定義されている用語または用語集に記載されている用語（あるいはその両方）を示します。	この句を指定すると、 索引構成表 が作成されます。
大文字の固定幅 フォント	大文字の固定幅フォントは、システムが提供する要素を示します。このような要素には、パラメータ、権限、データ型、 RMAN キーワード、 SQL キーワード、 SQL*Plus またはユーティリティ・コマンド、パッケージおよびメソッドが含まれます。また、システムが提供する列名、データベース・オブジェクト、データベース構造、ユーザー名およびロールも含まれます。	NUMBER 列に対してのみに、この句を指定できません。 BACKUP コマンドを使用して、データベースのバックアップを取ることができます。 USER_TABLES データ・ディクショナリ・ビュー内の TABLE_NAME 列を問い合わせます。 DBMS_STATS.GENERATE_STATS プロシージャを使用します。
小文字の固定幅 フォント	小文字の固定幅フォントは、実行可能ファイル、ファイル名、ディクショナリ名およびユーザーが提供する要素のサンプルを示します。このような要素には、コンピュータ名、データベース名、ネット・サービス名および接続識別子が含まれます。また、ユーザーが提供するデータベース・オブジェクトとデータベース構造、列名、パッケージとクラス、ユーザー名とロール、プログラム・ユニットおよびパラメータ値も含まれます。	sqlplus と入力して、 SQL*Plus を開きます。 パスワードは、 orapwd ファイルで指定します。 /disk1/oracle/dbs ディレクトリ内のデータ・ファイルおよび制御ファイルのバックアップを取ります。 hr.departments 表には、 department_id 、 department_name および location_id 列があります。 QUERY_REWRITE_ENABLED 初期化パラメータを true に設定します。 oe ユーザーとして接続します。 JRepUtil クラスが次のメソッドを実装します。
	注意： 大文字と小文字を組み合わせるプログラム要素もあります。これらの要素は、記載されているとおり入力してください。	

規則	意味	例
小文字の固定幅イタリック体	小文字の固定幅イタリック体は、プレースホルダまたは変数を示します。	<i>parallel_clause</i> を指定できます。 <i>Uold_release</i> .SQL を実行します。ここで、 <i>old_release</i> とはアップグレードの前にインストールしたリリースを示します。

コード例中の表記規則

コード例は、SQL、PL/SQL、SQL*Plus または他のコマンドライン文を説明します。コード例は、固定幅フォントで表示され、この例に示すとおり通常のテキストと区別されます。

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

次の表に、コード例で使用する表記規則を説明し、その使用例を示します。

規則	意味	例
[]	大カッコは、任意に選択する 1 つ以上の項目を囲みます。大カッコは、入力しないでください。	DECIMAL (<i>digits</i> [, <i>precision</i>])
{ }	中カッコは、2 つ以上の項目を囲み、そのうちの 1 つの項目は必須です。中カッコは、入力しないでください。	{ ENABLE DISABLE }
	縦線は、大カッコまたは中カッコ内の 2 つ以上のオプションの選択項目を表します。選択項目から 1 つを入力します。縦線は入力しないでください。	{ ENABLE DISABLE } [COMPRESS NOCOMPRESS]
...	水平の省略記号は、次のいずれかを示します。 <ul style="list-style-type: none"> ■ 例に直接関連しないコードの一部が省略されている。 ■ コードの一部を繰り返すことができる。 	CREATE TABLE ... AS <i>subquery</i> ; SELECT <i>col1</i> , <i>col2</i> , ... , <i>coln</i> FROM employees;
.	垂直の省略記号は、例に直接関連しない複数の行が省略されていることを示します。	
その他の句読点	大カッコ、中カッコ、縦線および省略記号以外の句読点は、表示されているとおり入力する必要があります。	<i>acctbal</i> NUMBER(11,2); <i>acct</i> CONSTANT NUMBER(4) := 3;

規則	意味	例
イタリック体	イタリック体は、特定の値を指定する必要があるプレースホルダや変数を示します。	CONNECT SYSTEM/ <i>system_password</i> DB_NAME = <i>database_name</i>
大文字	大文字は、システムが提供する要素を示します。これらの用語は、ユーザー定義の用語と区別するために大文字で示されます。用語が大カッコ内にかぎり、表示されているとおりの順序および綴りで入力します。ただし、これらの用語は大文字 / 小文字は識別されないため、小文字でも入力できます。	SELECT last_name, employee_id FROM employees; SELECT * FROM USER_TABLES; DROP TABLE hr.employees;
小文字	小文字は、ユーザー定義のプログラム要素を示します。たとえば、表名、列名、ファイル名などです。 注意： 大文字と小文字を組み合わせて使用するプログラム要素もあります。これらの要素は、記載されているとおりの入力してください。	SELECT last_name, employee_id FROM employees; sqlplus hr/hr CREATE USER mjones IDENTIFIED BY ty3MU9;

SQL リファレンスでの新規事項

この項では、Oracle9i リリース 1 (9.0.1) の新機能について説明し、追加情報の参照先を示します。Oracle7 から Oracle9i リリース 1 (9.0.1) へ移行するユーザーのために、Oracle8i での新機能についての情報も含まれています。

この項では、次の内容を説明します。

- [Oracle9i リリース 1 \(9.0.1\) の新機能](#)
- [Oracle8i の新機能](#)

Oracle9i リリース 1 (9.0.1) の新機能

今回のリリースで新しく追加または変更された組込みデータ型は、次のとおりです。

- Oracle は、組込み型または ANSI がサポートする型が不十分な場合に、新しい型を定義するための SQL に基づくインタフェースを提供します。2-38 ページの「[Oracle が提供する型](#)」を参照してください。
- 文字セマンティクスを示す CHAR パラメータおよびバイト・セマンティクスを示す BYTE パラメータについて、2-10 ページの「[CHAR データ型](#)」で説明します。
- 日時機能の追加機能について、2-22 ページの「[INTERVAL YEAR TO MONTH データ型](#)」および 2-22 ページの「[INTERVAL DAY TO SECOND データ型](#)」で説明します。
- 日時機能の追加機能について、2-20 ページの「[TIMESTAMP データ型](#)」で説明します。
- 文字セマンティクスを示す CHAR パラメータおよびバイト・セマンティクスを示す BYTE パラメータについて、2-11 ページの「[VARCHAR2 データ型](#)」で説明します。

今回のリリースでは、次の式の書式が新しく追加または拡張されています。

- 4-6 ページの「[CASE 式](#)」(拡張: 検索 CASE 式)
- 4-8 ページの「[CURSOR 式](#)」(拡張: REF CURSOR 引数としてファンクションに渡すことが可能)
- 4-10 ページの「[日時式](#)」(新規)
- 4-12 ページの「[期間式](#)」(新規)
- 4-13 ページの「[スカラー副問合せ式](#)」(新規)

今回のリリースでは、次の条件が新しく追加されています。

- [IS OF type 条件](#) (5-17 ページ)

今回のリリースでは、次の組込みファンクションが新しく追加されています。

- [ASCIIISTR](#) (6-17 ページ)
- [BIN_TO_NUM](#) (6-22 ページ)
- [COALESCE](#) (6-29 ページ)
- [COMPOSE](#) (6-31 ページ)
- [CURRENT_DATE](#) (6-44 ページ)
- [CURRENT_TIMESTAMP](#) (6-45 ページ)
- [DBTIMEZONE](#) (6-46 ページ)
- [DECOMPOSE](#) (6-48 ページ)

- [EXISTSNODE](#) (6-54 ページ)
- [EXTRACT \(日時\)](#) (6-56 ページ)
- [EXTRACT \(XML\)](#) (6-57 ページ)
- [FIRST](#) (6-58 ページ)
- [FROM_TZ](#) (6-62 ページ)
- [GROUP_ID](#) (6-64 ページ)
- [GROUPING_ID](#) (6-66 ページ)
- [LAST](#) (6-73 ページ)
- [LOCALTIMESTAMP](#) (6-81 ページ)
- [NULLIF](#) (6-101 ページ)
- [PERCENTILE_CONT](#) (6-108 ページ)
- [PERCENTILE_DISC](#) (6-110 ページ)
- [RAWTONHEX](#) (6-116 ページ)
- [ROWIDTONCHAR](#) (6-130 ページ)
- [SESSIONTIMEZONE](#) (6-132 ページ)
- [SYS_CONNECT_BY_PATH](#) (6-144 ページ)
- [SYS_DBURIGEN](#) (6-150 ページ)
- [SYS_EXTRACT_UTC](#) (6-151 ページ)
- [SYS_XMLAGG](#) (6-154 ページ)
- [SYS_XMLGEN](#) (6-155 ページ)
- [SYSTIMESTAMP](#) (6-157 ページ)
- [TO_CHAR \(文字\)](#) (6-159 ページ)
- [TO_CLOB](#) (6-163 ページ)
- [TO_DSINTERVAL](#) (6-165 ページ)
- [TO_NCHAR \(文字\)](#) (6-168 ページ)
- [TO_NCHAR \(日時\)](#) (6-169 ページ)
- [TO_NCHAR \(数値\)](#) (6-169 ページ)
- [TO_NCLOB](#) (6-170 ページ)
- [TO_TIMESTAMP](#) (6-172 ページ)

- [TO_TIMESTAMP_TZ](#) (6-173 ページ)
- [TO_YMINTERVAL](#) (6-174 ページ)
- [TREAT](#) (6-178 ページ)
- [TZ_OFFSET](#) (6-182 ページ)
- [UNISTR](#) (6-183 ページ)
- [WIDTH_BUCKET](#) (6-193 ページ)

今回のリリースでは、次のファンクションが拡張されています。

- [INSTR](#) (6-70 ページ)
- [LENGTH](#) (6-80 ページ)
- [SUBSTR](#) (6-141 ページ)

今回のリリースでは、次の権限が追加されています。

- [EXEMPT ACCESS POLICY](#) システム権限 (16-46 ページ)
- [RESUMABLE](#) システム権限 (16-46 ページ)
- [SELECT ANY DICTIONARY](#) システム権限 (16-46 ページ)
- [UNDER ANY TYPE](#) システム権限 (16-44 ページ)
- [UNDER ANY VIEW](#) システム権限 (16-45 ページ)
- [UNDER](#) オブジェクト権限 (16-49 ページ)

今回のリリースでは、次の SQL 文が新しく追加されています。

- [CREATE PFILE](#) (13-56 ページ)
- [CREATE SPFILE](#) (13-86 ページ)
- [MERGE](#) (16-76 ページ)

次の SQL 文には、新しい構文が追加されています。

- 8-9 ページの [ALTER DATABASE](#) には、データベースのマウント中にホット・バックアップ処理を終了するための新しい構文が追加されています。また、スタンバイ・データベースに関連する構文も追加されています。
- 8-48 ページの [ALTER INDEX](#) 文の構文では、索引を使用して統計を収集できます。
- 8-97 ページの [ALTER OUTLINE](#) では、パブリック・アウトラインおよびプライベート・アウトラインの両方を変更できます。

- 8-113 ページの **ALTER ROLE** では、アプリケーション固有のパッケージを使用したロールを識別できます。
- 9-2 ページの **ALTER SESSION** では、セッション中に発行された文を、いくつかの条件において一時停止可能にするかどうかを指定できます。
- 9-19 ページの **ALTER SYSTEM** では、SET 句が拡張されています。これによって、データベースを停止状態にできます。
- 10-2 ページの **ALTER TABLE** では、指定した値リストによって分割できます。
- 11-6 ページの **ALTER TYPE** では、オブジェクト型の属性またはメソッドの定義を変更できます。
- 11-28 ページの **ALTER VIEW** では、ビューの制約を追加できます。
- 11-31 ページの **ANALYZE** には、VALIDATE STRUCTURE 構文の一部として、ONLINE および OFFLINE 句が追加されています。また、システム統計またはユーザー定義の統計（あるいはその両方）を選択できます。
- 11-72 ページの **constraint_clause** は、制約を削除または使用禁止にするときの索引操作が簡単になっています。
- 12-11 ページの **CREATE CONTEXT** には、LDAP ディレクトリまたは Oracle Call Interface (OCI) インタフェースからコンテキストを初期化する構文、およびインスタンスからコンテキストへのアクセスを可能にする構文が追加されています。
- 12-14 ページの **CREATE CONTROLFILE** では、Oracle 管理ファイルを作成できます。
- 12-20 ページの **CREATE DATABASE** では、データベース作成時に、デフォルトの一時表領域を作成できます。また、UNDO 表領域を作成することもできます。
- 12-47 ページの **CREATE FUNCTION** では、パイプライン表ファンクション、パラレル表ファンクションおよびユーザー定義の集計ファンクションを作成できます。
- 13-42 ページの **CREATE OUTLINE** では、パブリック・アウトラインおよびプライベート・アウトラインの両方を作成できます。
- 13-72 ページの **CREATE ROLE** では、アプリケーション固有のパッケージを使用したロールを識別できます。
- 14-6 ページの **CREATE TABLE** では、外部表（データベースの外部にデータを持つ表）および Oracle 管理ファイルを作成できます。また、指定した値リストの分割もできます。
- 14-62 ページの **CREATE TABLESPACE** では、空きリストと同様に、ビットマップを使用したセグメント領域管理ができます。また、Oracle 管理ファイルおよび UNDO 表領域を作成できます。
- 14-73 ページの **CREATE TEMPORARY TABLESPACE** では、Oracle 管理ファイルを作成できます。

- 15-3 ページの **CREATE TYPE** では、サブタイプを作成できます。
- 15-37 ページの **CREATE VIEW** では、オブジェクト・ビューのサブビューを作成できます。また、ビューに制約を定義できます。
- 16-9 ページの **DROP TABLESPACE** では、削除された表領域の内容を削除するときに、オペレーティング・システム・ファイルを削除する構文が追加されています。
- 16-27 ページの **filespec** では、Oracle 管理ファイルを作成できます。
- 16-56 ページの **INSERT** では、列のデフォルト値を挿入する構文が追加されています。
- 17-4 ページの **SELECT** では、GROUP BY 句の中で複数のグループ化を指定して、複数のディメンションにまたがる選択性の高い分析ができます。また、副問合せブロックに名前を割り当てることができます。ANSI に準拠した結合構文が追加されています。
- 17-46 ページの **SET TRANSACTION** では、トランザクション名を指定できます。
- 17-64 ページの **UPDATE** では、列をデフォルト値に更新する構文が追加されています。

Oracle8i の新機能

このバージョンでは、次の SQL ファンクションが追加されています。

- **BITAND** (6-23 ページ)
- **CORR** (6-34 ページ)
- **COVAR_POP** (6-39 ページ)
- **COVAR_SAMP** (6-41 ページ)
- **CUME_DIST** (6-43 ページ)
- **DENSE_RANK** (6-49 ページ)
- **FIRST_VALUE** (6-60 ページ)
- **LAG** (6-71 ページ)
- **LAST_VALUE** (6-76 ページ)
- **LEAD** (6-78 ページ)
- **NTILE** (6-99 ページ)
- **NUMTOYMINTERVAL** (6-103 ページ)
- **NUMTODSINTERVAL** (6-102 ページ)
- **NVL2** (6-105 ページ)

- [PERCENT_RANK](#) (6-106 ページ)
- [RANK](#) (6-112 ページ)
- [RATIO_TO_REPORT](#) (6-114 ページ)
- [REGR_](#) (線形リグレーション) [ファンクション](#) (6-118 ページ)
- [STDDEV_POP](#) (6-138 ページ)
- [STDDEV_SAMP](#) (6-139 ページ)
- [VAR_POP](#) (6-188 ページ)
- [VAR_SAMP](#) (6-189 ページ)

リリース 8.1.5 では、次の SQL 文が追加されています。

- [ALTER DIMENSION](#) (8-43 ページ)
- [ALTER JAVA](#) (8-71 ページ)
- [ALTER OUTLINE](#) (8-97 ページ)
- [ASSOCIATE STATISTICS](#) (11-46 ページ)
- [CALL](#) (11-63 ページ)
- [CREATE CONTEXT](#) (12-11 ページ)
- [CREATE DIMENSION](#) (12-39 ページ)
- [CREATE INDEXTYPE](#) (12-84 ページ)
- [CREATE JAVA](#) (12-86 ページ)
- [CREATE OPERATOR](#) (13-38 ページ)
- [CREATE OUTLINE](#) (13-42 ページ)
- [CREATE TEMPORARY TABLESPACE](#) (14-73 ページ)
- [DISASSOCIATE STATISTICS](#) (15-57 ページ)
- [DROP CONTEXT](#) (15-62 ページ)
- [DROP DIMENSION](#) (15-65 ページ)
- [DROP INDEXTYPE](#) (15-73 ページ)
- [DROP JAVA](#) (15-75 ページ)
- [DROP OPERATOR](#) (15-82 ページ)
- [DROP OUTLINE](#) (15-84 ページ)

さらに、次の機能が拡張されています。

- 集計ファンクションは機能が拡張されました。6-6 ページの「[集計ファンクション](#)」を参照してください。
- LOB 領域記憶パラメータを指定するとき、読取り専用で LOB キャッシュを指定できます。14-6 ページの「[CREATE TABLE](#)」を参照してください。
- 式に関する項に、新しい式が追加されました。4-6 ページの「[CASE 式](#)」を参照してください。
- 副問合せのネスト解除が可能になりました。7-13 ページの「[ネストされた副問合せのネスト解除](#)」を参照してください。

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

この章では、次の内容を説明します。

- [SQL の歴史](#)
- [SQL 規格](#)
- [埋込み SQL](#)
- [字句規則](#)
- [Oracle のツール製品のサポート](#)

SQL の歴史

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

SQL 規格

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

1999 年 6 月に採用された最新の SQL 規格を SQL:99 といいます。新規格の正式名称は、次のとおりです。

- ANSI X3.135-1999、『Database Language SQL』、Part 1 「Framework」、Part 2 「Foundation」、Part 5 「Bindings」
- ISO/IEC 9075:1999、『Database Language SQL』、Part 1 「Framework」、Part 2 「Foundation」、Part 5 「Bindings」

参照： SQL:99 規格への Oracle の規格準拠については、[付録 B 「Oracle と標準 SQL」](#) で説明します。

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

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

SQL 文を使用して、次の処理を行うことができます。

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

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

すべてのリレーショナル・データベースに共通の言語

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

埋込み SQL

埋込み SQL は、手続き型プログラミング言語に埋め込んで使用する標準の SQL 文です。埋込み SQL 文は、Oracle プリコンパイラ関連のマニュアルに記載されています。

埋込み SQL は、次のコマンドの集まりです。

- 対話形式の Oracle のツール製品を使用した、SQL で使用できるすべての SQL コマンド (SELECT、INSERT など)。
- 手続き型プログラミング言語に標準 SQL コマンドを統合する動的 SQL 実行コマンド (PREPARE、OPEN など)。

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 文が解析され、手続き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手続き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
- SQL*Module for ADA

参照： Oracle プリコンパイラおよび埋込み SQL 文の定義については、『Programmer's Guide to SQL*Module for Ada』、『Pro*C/C++ Precompiler プログラマーズ・ガイド』および『Pro*COBOL Precompiler プログラマーズ・ガイド』を参照してください。

字句規則

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

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

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

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

予約語、キーワード、識別子およびパラメータは、大文字と小文字を区別せずに記述できます。ただし、テキスト・リテラルと引用符で囲んだ名前では、大文字と小文字は区別されます。

参照： 構文の説明は、2-52 ページの「[テキスト・リテラル](#)」を参照してください。

Oracle のツール製品のサポート

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

Oracle SQL の基本要素

この章では、Oracle SQL の基本要素に関する参照情報を説明します。これらの要素は、SQL 文の最も単純な構成ブロックです。したがって、[第 8 章～第 17 章](#)で説明されている文を使用する前に、この章で説明する次の概念を理解し、[第 3 章「演算子」](#)、[第 6 章「ファンクション」](#)、[第 4 章「式」](#) および [第 7 章「SQL 問合せおよびその他の SQL 文」](#) をよく読んでおく必要があります。

この章では、次の内容を説明します。

- [データ型](#)
- [リテラル](#)
- [書式モデル](#)
- [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・オブジェクトの構文および SQL 文の構成要素](#)

データ型

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

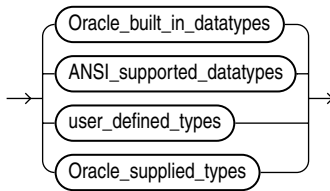
表またはクラスタを作成する場合、各列にデータ型を指定する必要があります。プロシージャまたはストアド・ファンクションを作成する場合は、その各引数にデータ型を指定する必要があります。データ型によって、各列が含むことができる値のドメイン、または各引数が持つことができる値のドメインが決まります。たとえば、DATE 列は、2 月 29 日（うるう年を除く）、2 または 'SHOE' という値を格納できません。列に入る値は、その列のデータ型を受け継ぎます。たとえば、DATE 列に '01-JAN-98' を挿入すると、Oracle はそれが有効な日付に変換されることを確認してから、文字列 '01-JAN-98'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

Oracle には、多くの組込みデータ型およびいくつかのユーザー定義型のカテゴリがあります。Oracle のデータ型の構文については、次の構文図で示します。この項は、次の 4 つの項に分かれています。

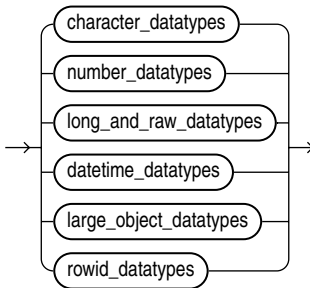
- Oracle の組込みデータ型
- ANSI、DB2、SQL/DS のデータ型
- ユーザー定義型
- Oracle が提供する型

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

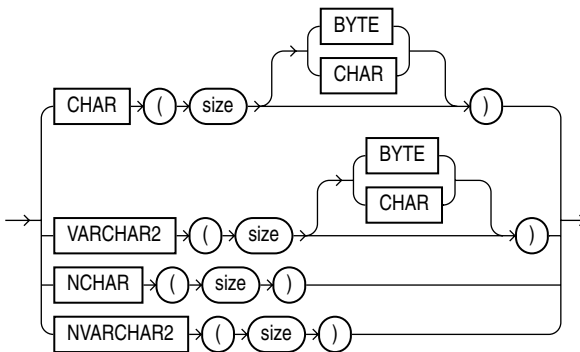
datatypes::=



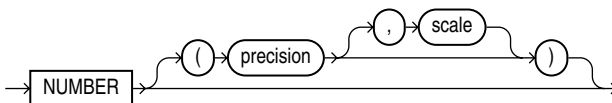
Oracle_built_in_datatypes::=



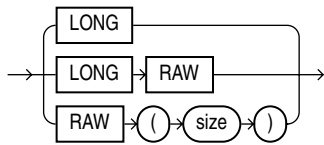
character_datatypes::=



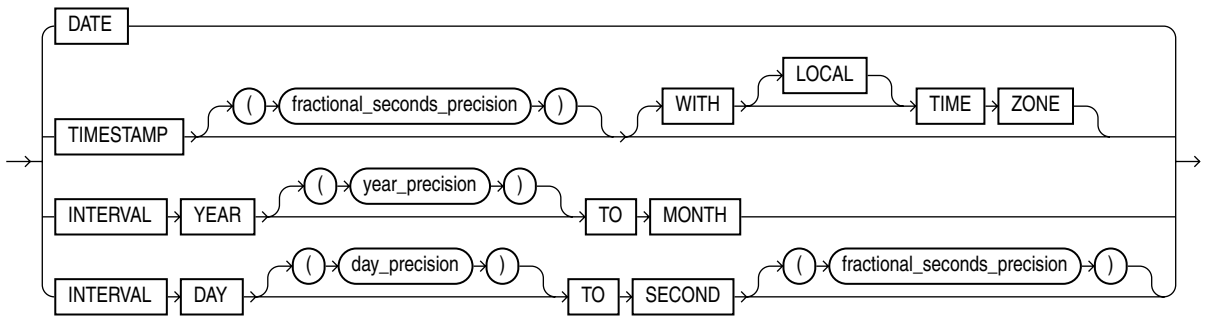
number_datatypes::=



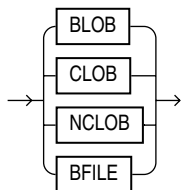
long_and_raw_datatypes::=



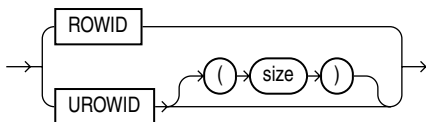
datetime_datatypes::=



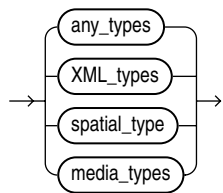
large_object_datatypes::=



rowid_datatypes::=



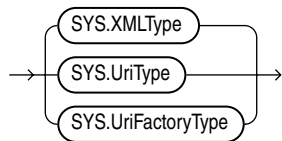
Oracle_supplied_types::=



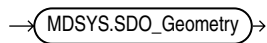
any_types::=



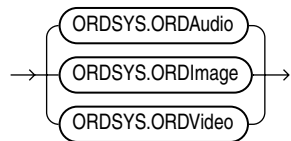
XML_types::=



spatial_type::=



media_types::=



Oracle の組込みデータ型

表 2-1 に Oracle 組込みデータ型の概要を示します。

表 2-1 組込みデータ型の概要

コード ^a	組込みデータ型	説明
1	VARCHAR2(<i>size</i>) [BYTE CHAR]	最大長が <i>size</i> バイトまたは <i>size</i> 文字の可変長文字列。最大サイズは 4000、最小サイズは 1 です。VARCHAR2 には、 <i>size</i> を指定する必要があります。 BYTE は、列がバイト長のセマンティクスを持つことを示し、CHAR は、列が文字のセマンティクスを持つことを示します。
1	NVARCHAR2(<i>size</i>)	最大長が <i>size</i> 文字の可変長文字列。最大サイズは、各国語キャラクタ・セット定義 (上限 4000 バイト) によって決定されます。NVARCHAR2 の <i>size</i> を指定する必要があります。
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 ~ 38 の値を指定できます。位取り <i>s</i> には -84 ~ 127 の値を指定できます。
8	LONG	最大 2GB (2 ³¹ から 1 を引いたバイト数) の可変長文字データ。
12	DATE	紀元前 4712 年 1 月 1 日 ~ 紀元 9999 年 12 月 31 日までの日付を指定します。
180	TIMESTAMP (<i>fractional_</i> <i>seconds_precision</i>)	日付の年、月、日および時刻の時、分、秒の値。 <i>fractional_seconds_precision</i> は、SECOND 日時フィールドの小数部の桁数です。 <i>fractional_seconds_precision</i> の有効範囲は、0 ~ 9 です。デフォルトは 6 です。
181	TIMESTAMP (<i>fractional_</i> <i>seconds_precision</i>) WITH TIME ZONE	タイム・ゾーンの置換値などのすべての TIMESTAMP の値。 <i>fractional_seconds_precision</i> は、SECOND 日時フィールドの小数部の桁数です。有効範囲は 0 ~ 9 です。デフォルトは 6 です。

^a データ型のコードは、Oracle が内部的に使用します。DUMP ファンクションによって、列またはオブジェクト属性のデータ型コードが戻されます。

表 2-1 組み込みデータ型の概要（続き）

コード ^a	組み込みデータ型	説明
231	TIMESTAMP (<i>fractional_seconds_precision</i>) WITH LOCAL TIME ZONE	TIMESTAMP WITH TIME ZONE のすべての値。ただし、次に示す例外があります。 <ul style="list-style-type: none"> ■ データベースへの格納時、データがデータベースのタイム・ゾーンに正規化されている場合。 ■ データの検索時、ユーザーがセッションのタイム・ゾーンでデータを検索する場合。
182	INTERVAL YEAR (<i>year_precision</i>) TO MONTH	年および月で期間を格納。 <i>year_precision</i> は、YEAR 日時フィールドの桁数です。有効範囲は 0～9 です。デフォルトは 2 です。
183	INTERVAL DAY (<i>day_precision</i>) TO SECOND (<i>fractional_seconds_precision</i>)	日、時、分および秒で期間を格納。 <ul style="list-style-type: none"> ■ <i>day_precision</i> は、DAY 日時フィールドの最大桁数です。有効範囲は 0～9 です。デフォルトは 2 です。 ■ <i>fractional_seconds_precision</i> は、SECOND フィールドの小数部の桁数です。有効範囲は 0～9 です。デフォルトは 6 です。
23	RAW(<i>size</i>)	長さ <i>size</i> バイトのバイナリ・データ。最大サイズは 2000 バイトです。RAW 値には、 <i>size</i> を指定する必要があります。
24	LONG RAW	最大 2GB の可変長バイナリ・データ。
69	ROWID	表の中の行のアドレスを一意に表す 16 進文字列。主に、ROWID 疑似列によって戻される値のためのデータ型です。
208	UROWID [(<i>size</i>)]	索引構成表の行の論理アドレスを表す 16 進文字列。オプションの <i>size</i> は、UROWID 型の列のサイズです。最大サイズおよびデフォルトは 4000 バイトです。
96	CHAR(<i>size</i>) [BYTE CHAR]	長さ <i>size</i> バイトの固定長文字データ。最大サイズは 2000 バイトです。デフォルトおよび最小サイズは 1 です。 BYTE および CHAR は、VARCHAR2 と同じセマンティクスを持ちます。

^a データ型のコードは、Oracle が内部的に使用します。DUMP ファンクションによって、列またはオブジェクト属性のデータ型コードが戻されます。

表 2-1 組み込みデータ型の概要（続き）

コード ^a	組み込みデータ型	説明
96	NCHAR(<i>size</i>)	長さが <i>size</i> 文字の固定長文字データ。最大サイズは、各国語キャラクタ・セット定義（上限 2000 バイト）によって決定されます。デフォルトおよび最小サイズは 1 文字です。
112	CLOB	シングルスバイト・キャラクタを含むキャラクタ・ラージ・オブジェクト。固定幅および可変幅のキャラクタ・セットがサポートされます。両方のキャラクタ・セットで CHAR データベース・キャラクタ・セットを使用します。最大サイズは 4GB です。
112	NCLOB	Unicode キャラクタを含むキャラクタ・ラージ・オブジェクト。固定幅および可変幅のキャラクタ・セットがサポートされます。両方のキャラクタ・セットで NCHAR データベース・キャラクタ・セットを使用します。最大サイズは 4GB です。各国語キャラクタ・セットのデータを格納します。
113	BLOB	バイナリ・ラージ・オブジェクト。最大サイズは 4GB です。
114	BFILE	データベース外に保存された大きなバイナリ・ファイルへの参照を格納。データベース・サーバー上に存在する外部 LOB へのバイト・ストリーム I/O アクセスを可能にします。最大サイズは 4GB です。

^a データ型のコードは、Oracle が内部的に使用します。DUMP ファンクションによって、列またはオブジェクト属性のデータ型コードが戻されます。

文字データ型

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

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

次のデータ型が、文字データに対して使用されます。

- [CHAR データ型](#)
- [NCHAR データ型](#)
- [NVARCHAR2 データ型](#)
- [VARCHAR2 データ型](#)

CHAR データ型

CHAR データ型は、固定長の文字列を指定します。Oracle では、その列に格納される値がすべて *size* で指定した長さを持つように調整されます。列の長さより短い値が挿入されると、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と、Oracle はエラーを戻します。

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

CHAR 列を持つ表を作成する場合、デフォルトでは列の長さはバイト単位になります。BYTE 修飾子は、デフォルトと同じです。CHAR 修飾子（たとえば、CHAR(10 CHAR)）を使用すると、列の長さは文字単位になります。技術的な言い方をすると、文字は、データベース・キャラクタ・セットのコードポイントです。サイズは、データベース・キャラクタ・セットによって異なりますが、1 バイトから 4 バイトです。BYTE および CHAR 修飾子は、NLS_LENGTH_SEMANTICS パラメータ（デフォルトはバイト・セマンティクス）で指定したセマンティクスを上書きします。

注意： 異なるキャラクタ・セットを持つデータベース間で適切にデータを変換するには、CHAR データが正しい書式の文字列で構成されていることを確認してください。キャラクタ・セット・サポートの詳細は、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

参照： 比較セマンティクスについては、2-42 ページの「[データ型の比較規則](#)」を参照してください。

NCHAR データ型

Oracle9i から、NCHAR データ型は Unicode のみのデータに再定義されています。NCHAR 列を持つ表を作成する場合、列の長さを文字単位で指定します。使用する各国語キャラクタ・セットは、データベースを作成するときに指定します。

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

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

次の例では、product_descriptions の translated_name 列と各国語キャラクタ・セットの文字列 LCD Monitor 11/PM を比較します。

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

参照： Unicode データ型のサポートについては、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

NVARCHAR2 データ型

Oracle9i から、NVARCHAR2 データ型は Unicode のみのデータ型に再定義されています。NVARCHAR2 列を持つ表を作成する場合、保持できる最大の文字数を指定します。Oracle では、列の最大長を超えないかぎり、各値を指定されたとおりに正確に列に格納します。

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

参照： Unicode データ型のサポートについては、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

VARCHAR2 データ型

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

VARCHAR2 列には最大長を指定する必要があります。保存される文字列の実際の長さは 0 (ゼロ) にできますが、最大長は 1 バイト以上にする必要があります。Oracle は、長さが 0 (ゼロ) の文字列を NULL として処理します。CHAR 修飾子 (たとえば、VARCHAR2(10 CHAR)) を使用すると、バイトではなく、文字で最大長を指定できます。技術的な言い方をすると、文字は、データベース・キャラクタ・セットのコードポイントです。CHAR および BYTE 修飾子は、NLS_LENGTH_SEMANTICS パラメータ (デフォルトはバイト) の設定を上書きします。VARCHAR2 データの最大長は 4000 バイトです。Oracle は、非空白埋め比較セマンティクスを使用して VARCHAR2 値を比較します。

注意： 異なるキャラクタ・セットを持つデータベース間で適切にデータを変換するには、VARCHAR2 データが正しい書式の文字列で構成されていることを確認してください。キャラクタ・セット・サポートの詳細は、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

参照： 比較セマンティクスについては、2-42 ページの「[データ型の比較規則](#)」を参照してください。

VARCHAR データ型

現在、VARCHAR データ型は、VARCHAR2 データ型と同じ意味で使用されています。VARCHAR より VARCHAR2 を使用することをお勧めします。今後、VARCHAR データ型が変更され、異なる比較セマンティクスで比較される別の可変長文字列の型になる可能性があります。

NUMBER データ型

NUMBER データ型は、0（ゼロ）、および精度が 38 桁で、絶対値が $1.0 \times 10^{-130} \sim 9.9...9 \times 10^{125}$ （38 個の 9 の後に 0 が 88 個続く）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します。 1.0×10^{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)

それぞれの意味は、次のとおりです。

- *p* は、精度 (**precision**)、つまり全体の桁数です。Oracle は、38 桁までの精度で数の移植性を保証します。
- *s* は、位取り (**scale**)、つまり小数点の右側にある桁数です。位取りの有効範囲は -84 ～ 127 です。

次の書式で整数を指定できます。

NUMBER (p)

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

次の書式で浮動小数点を指定できます。

NUMBER

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

参照： 2-14 ページの「[浮動小数点数](#)」を参照してください。

位取りと精度

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

次の例では、異なる精度および位取りを使用した Oracle のデータの格納方法を示します。

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、実際のデータは整数部分の右から指定された桁数のみ丸められます。たとえば、(10,-2) と指定すると 100 の位まで丸められます。

精度より大きい位取り

通常はありえないことですが、精度より大きい位取りを指定することもできます。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）である必要があり、小数点以下 5 桁を超える値はすべて丸められます。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。指数は、数字の後に増加する桁数を表すときに、オプションで使用されます（たとえば、 1.777 e^{-20} ）。小数点以下の桁数に制限はないため、浮動小数点数に対して位取りは指定できません。

2-12 ページの「[NUMBER データ型](#)」で説明する範囲で、浮動小数点数を指定できます。書式については、2-53 ページの「[数値リテラル](#)」を参照してください。Oracle は ANSI のデータ型である 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 では、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 (b) は、浮動小数点数をバイナリ精度 *b* に指定します。精度 *b* は、1 ～ 126 の範囲で指定します。2 進精度から 10 進精度に変換するには、*b* に 0.30103 を乗算します。10 進精度から 2 進精度に変換するには、10 進精度に 3.32193 を乗算します。2 進精度の 126 桁は、10 進精度の 38 桁とほぼ等しくなります。

LONG データ型

LONG 列には、最大 2GB (2^{31} 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 列と同じ特徴があります。LONG 列を使用すると、長いテキスト列を格納できます。LONG 値の長さは、ご使用のコンピュータで利用できるメモリによって制限される場合もあります。

注意： LONG 列を LOB 列へ変換することをお勧めします。LOB 列は、LONG 列ほど制限は多くありません。LONG 列から LOB 列への変換については、10-2 ページの「ALTER TABLE」の *modify_column_options* 句および 6-166 ページの「TO_LOB」を参照してください。

SQL 文の中の次の場所で LONG 列を参照できます。

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

LONG 値を使用する場合には、次の制限があります。

- 表には複数の LONG 列を含めることはできません。
- LONG 属性を持つオブジェクトは作成できません。
- LONG 列は、WHERE 句または整合性制約では指定できません (NULL および NOT NULL 制約は除く)。
- LONG 列に索引を付けることはできません。
- ストアド・ファンクションは LONG 値を戻すことはできません。
- LONG データ型を使用して、PL/SQL プログラム単位の変数または引数を宣言できます。ただし、そのプログラムは、SQL からコールできません。
- 単一の SQL 文に指定する、すべての LONG 列、更新された表、ロックされた表は、同一データベース上にある必要があります。
- LONG 列および LONG RAW 列は、分散型の SQL 文で使用できません。また、レプリケートできません。
- LONG と LOB の両方の列を持つ表では、1 つの SQL 文で両方に 4000 バイトを超えるデータをバインドすることはできません。ただし、どちらか片方にバインドすることは可能です。
- LONG 列を持つ表は、自動セグメント領域管理の表領域に格納できません。

また、LONG 列は SQL 文の次のような部分では使用できません。

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

トリガーでは、LONG データ型は次のように使用されます。

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

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

参照：『Oracle Call Interface プログラマーズ・ガイド』を参照してください。

日時および期間データ型

日時データ型には、DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONE および TIMESTAMP WITH LOCAL TIME ZONE があります。日時データ型の値は、「日時」とも呼ばれます。期間データ型には、INTERVAL YEAR TO MONTH および INTERVAL DAY TO SECOND があります。期間データ型の値は、「期間」とも呼ばれます。

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

日時フィールド	日時に有効な値	期間に有効な値
YEAR	-4712 ～ 9999 (0 を除く)	正または負のすべての整数
MONTH	01 ～ 12	0 ～ 11
DAY	01 ～ 31 (現在の NLS カレンダに従った MONTH および YEAR の値の範囲 内)	正または負のすべての整数
HOURL	00 ～ 23	0 ～ 23
MINUTE	00 ～ 59	0 ～ 59
SECOND	00 ～ 59.9(n) (「9(n)」は秒の小数部の精度)	00 ～ 59.9(n) (「9(n)」は秒の小数部の期間の精度)
TIMEZONE_HOUR	-12 ～ 13 (この範囲は夏時間の 変更を保存する)	適用なし
TIMEZONE_MINUTE	00 ～ 59	適用なし

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

DATE データ型

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

日付値をリテラルに指定するか、文字値や数値を TO_DATE ファンクションによって日付値に変換できます。日付値をリテラルに指定する場合は、グレゴリオ暦を使用する必要があります。次のように、ANSI の日付リテラルを指定できます。

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

ANSI の日付リテラルは、時刻部分を含みません。また、'YYYY-MM-DD' という書式で指定する必要があります。また、次のように、Oracle の日付リテラルを指定できます。

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

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

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

日付値を指定する場合に時刻コンポーネントを指定しないと、デフォルト時刻の午前 12:00:00（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。

Oracle の DATE 列には、常に、日付フィールドと時刻フィールドが含まれます。時刻部分を除いた日付書式を使用する問合せを発行する場合は、DATE 列の時刻フィールドが 0（真夜中）に設定されていることを確認してください。そうでない場合、Oracle は、正しい結果を戻さない場合があります。次の例では、数値列 row_num および DATE 列 datecol を持つ表 my_table があると想定します。

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

```
SELECT * FROM my_table;
```

```

  ROW_NUM DATECOL
-----
         1 04-OCT-00
         2 04-OCT-00
```

```
SELECT * FROM my_table
      WHERE datecol = TO_DATE('04-OCT-00','DD-MON-YY');
```

```
      ROW_NUM DATECOL
-----
      2 04-OCT-00
```

DATE 列の時刻フィールドが 0 に設定されている場合は、前述の 2 つ目の例に示すように、DATE 列に対して問い合わせるか、DATE リテラルを使用して問い合わせることができます。

```
SELECT * FROM my_table WHERE datecol = DATE '2000-10-04';
```

ただし、DATE 列が 0（ゼロ）以外の時刻フィールドを含む場合、正しい結果を得るためには、問合せで時刻フィールドを排除する必要があります。次に例を示します。

```
SELECT * FROM my_table WHERE TRUNC(datecol) = DATE '2000-10-04';
```

Oracle は、問合せの各行に TRUNC ファンクションを適用します。これによって、データの時刻フィールドが 0 である場合のパフォーマンスが向上します。時刻フィールドが 0 に設定されていることを確認するには、挿入および更新時に次のいずれかの操作を行います。

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

```
INSERT INTO my_table VALUES
      (3, TO_DATE('4-APR-2000','DD-MON-YYYY'));
```

- DATE リテラルを使用します。

```
INSERT INTO my_table VALUES (4, '04-OCT-00');
```

- TRUNC ファンクションを使用します。

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

日付ファンクション SYSDATE は、現在のシステムの日付および時刻を戻します。

CURRENT_DATE ファンクションは、現在のセッションの日付を戻します。SYSDATE、TO_* 日時ファンクションおよびデフォルト日付書式の詳細は、[第 6 章「ファンクション」](#)を参照してください。

日付算術 日付に対しては、日付の加算や減算のみでなく、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析します。たとえば、SYSDATE+1 は明日です。SYSDATE-7 は 1 週間前です。SYSDATE+(10/1440) は 10 分後です。SYSDATE から employees サンプル表の 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のファンクションを用意しています。たとえば、ADD_MONTHS ファンクションを使用すると、日付に月を加算したり、減算できます。MONTHS_BETWEEN ファンクションは、2 つの日付の間の月数を戻します。結果の小数部は月（1ヶ月は 31 日）を単位として表されます。

各日付には時刻コンポーネントが含まれるため、ほぼすべての日付操作の結果には小数部が含まれます。この小数部は、日を単位として表されています。たとえば、1.5 日は 36 時間です。

参照：

- 日付関クションの詳細は、6-5 ページの「[日時関クション](#)」を参照してください。
 - 算術を含むその他の日時および期間データ型については、2-23 ページの「[日時および期間の演算](#)」を参照してください。
-
-

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。日付関クション `TO_DATE` と `TO_CHAR` で日付書式モデル「J」を使用して、Oracle の `DATE` 値とユリウス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 日をユリウス暦で戻します。

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

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

`DUAL` 表については、7-14 ページの「[DUAL 表からの選択](#)」を参照してください。

TIMESTAMP データ型

`TIMESTAMP` データ型は、`DATE` データ型の拡張機能です。`DATE` データ型の年、月および日に加えて、時、分および秒の値を格納します。`TIMESTAMP` データ型は、次のように指定します。

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

fractional_seconds_precision は、オプションで `SECOND` 日時フィールドの小数部の桁数を指定し、有効範囲は 0～9 です。デフォルトは 6 です。たとえば、リテラルとしての `TIMESTAMP` は、次のように指定します。

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

TIMESTAMP WITH TIME ZONE データ型

TIMESTAMP WITH TIME ZONE は、**タイム・ゾーン置換値**を値に含む TIMESTAMP の可変です。タイム・ゾーン置換値は、ローカルの時刻と UTC との（時および分の）差異です。TIMESTAMP WITH TIME ZONE データ型は、次のように指定します。

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

fractional_seconds_precision は、オプションで SECOND 日時フィールドの小数部の桁数を指定し、有効範囲は 0～9 です。デフォルトは 6 です。たとえば、リテラルとしての TIMESTAMP WITH TIME ZONE は、次のように指定します。

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

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

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

前述の例文は次の例文と同じです。

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

つまり、太平洋標準時の午前 8 時は、東部標準時の午前 11 時と同じです。

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

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

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

```
TIMESTAMP '1999-10-31 01:30:00 US/Pacific PDT'
```

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

参照：

- 夏時間のサポートについては、2-23 ページの「[夏時間のサポート](#)」および 2-67 ページの表 2-12「[日時書式要素](#)」を参照してください。
- ERROR_ON_OVERLAP_TIME セッション・パラメータについては、9-2 ページの「[ALTER SESSION](#)」を参照してください。

TIMESTAMP WITH LOCAL TIME ZONE データ型

TIMESTAMP WITH LOCAL TIME ZONE は、**タイム・ゾーン置換値**を値に含む TIMESTAMP のもう 1 つの可変です。これは、TIMESTAMP WITH TIME ZONE と異なり、データベースに格納されるデータはデータベース・タイム・ゾーンに対して正規化され、タイム・ゾーン置換値は列データの一部として格納されません。ユーザーがデータを検索すると、Oracle はユーザーのローカル・セッション・タイム・ゾーンのデータを戻します。タイム・ゾーン置換値は、ローカルの時刻と UTC との（時および分の）差異です。TIMESTAMP WITH LOCAL TIME ZONE データ型は、次のように指定します。

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

fractional_seconds_precision は、オプションで SECOND 日時フィールドの小数部の桁数を指定し、有効範囲は 0～9 です。デフォルトは 6 です。

TIMESTAMP WITH LOCAL TIME ZONE には、リテラルはありません。

参照： データ型の使用例は、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

INTERVAL YEAR TO MONTH データ型

INTERVAL YEAR TO MONTH は、YEAR および MONTH 日時フィールドを使用して期間を格納します。INTERVAL YEAR TO MONTH は、次のように指定します。

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

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

INTERVAL DAY TO SECOND データ型

INTERVAL DAY TO SECOND は、日付、時、分および秒で期間を格納します。このデータ型は、次のように指定します。

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

それぞれの意味は、次のとおりです。

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

日時および期間の演算

Oracle では、日時および期間の値の式を導出できます。日時の値の式から、日時データ型の値を得ることができます。期間の値の式から、期間データ型の値を得ることができます。表 2-2 に、これらの式で使用する演算子を示します。

表 2-2 日時および期間の値の式の演算子

オペランド 1	演算子	オペランド 2	結果タイプ
日時	+	期間	日時
日時	-	期間	日時
期間	+	日時	日時
日時	-	日時	期間
期間	+	期間	期間
期間	-	期間	期間
期間	*	数値	期間
数値	*	期間	期間
期間	/	数値	期間

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

夏時間のサポート

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

このような境界を解決するために、Oracle は TZR および TZD 書式要素を使用します。詳細は、2-67 ページの表 2-12 を参照してください。TZR は、日時の入力文字列でタイム・ゾーン地域を表します。たとえば、'Australia/North'、'UTC'、'Singapore' などです。TZD は、夏時間情報を含むタイム・ゾーン地域の略称書式です。たとえば、米国 / 太平洋標準時は 'PST'、米国 / 太平洋夏時間は 'PDT' などです。TZR および TZD 書式要素の値を表示するには、V\$TIMEZONE_NAMES 動的パフォーマンス・ビューの TZNAME および TZABBREV 列に問合せを実行してください。

参照：

- 書式要素の詳細は、2-66 ページの「日付書式モデル」を参照してください。
- 動的パフォーマンス・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

日時および期間の例

次の例では、日時および期間データ型の宣言方法を示します。

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

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

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

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

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

注意： LONG RAW 列をバイナリ LOB (BLOB) へ変換することをお勧めします。LOB 列は、LONG 列ほど制限は多くありません。詳細は、6-166 ページの「[TO_LOB](#)」を参照してください。

RAW は、VARCHAR2 と同様に可変長データ型ですが、Oracle Net（ユーザー・セッションとインスタンスを接続します）およびインポート / エクスポート・ユーティリティは、RAW または LONG RAW データの転送時に文字変換を行いません。これに対し、Oracle Net およびインポート / エクスポートは、データベース・キャラクタ・セットとユーザー・セッションのキャラクタ・セット（ALTER SESSION 文の NLS_LANGUAGE パラメータで設定します）が異なる場合に、CHAR、VARCHAR2 および 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 データ間で、データを自動的に変換するとき、バイナリ・データは 16 進数で表されます。1 つの 16 進文字で 4 ビットの RAW データを表します。たとえば、ビット列が 11001011 で表示される 1 バイトの RAW データは、「CB」として表示または入力されます。

ラージ・オブジェクト (LOB) データ型

組込み LOB データ型の BLOB、CLOB、NCLOB（内部ファイルに格納）および BFILE（外部ファイルに格納）には、構造化されていない大きいデータ（text、image、video、spatial data など）を最大 4GB まで格納できます。

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

LOB 列には、アウトライン 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 Oracle Call Interface (OCI) の操作は、これらのロケータを介して行われます。

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

- LOB は、ユーザー定義のデータ型（オブジェクト）の属性に指定できます。
- LOB ロケータは、表の列に格納されます。実際の LOB 値は、表の列に格納される場合と格納されない場合があります。BLOB、NCLOB および CLOB の値は、別々の表領域に格納されます。BFILE データは、サーバー上の外部ファイルに格納されます。
- LOB 列にアクセスしたときに戻されるのはロケータです。
- LOB の最大サイズは 4GB です。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りますが、4GB を超えることはありません。
- LOB では、効果的かつランダムなピース単位のデータ・アクセスおよび操作が可能です。
- 1 つの表内に 2 つ以上の LOB 列を定義できます。
- NCLOB の例外を除いて、1 つのオブジェクトに 1 つ以上の LOB 属性を定義できます。
- LOB バインド変数を宣言できます。
- LOB 列と LOB 属性を選択できます。
- 1 つ以上の LOB 列または 1 つ以上の LOB 属性を持つ、オブジェクトが含まれている新しい行を挿入したり、既存の行を更新できます（内部 LOB 値を NULL つまり空に設定したり、LOB 全体をデータに置き換えられます。BFILE は、NULL に設定したり、別のファイルを指すように設定できます）。
- LOB 行と列の交差部または LOB 属性を、別の LOB 行と列の交差部または LOB 属性を使用して更新できます。
- LOB 列または LOB 属性が含まれている行を削除できます（これによって LOB 値も削除されます）。BFILE の場合、実際のオペレーティング・システム・ファイルは削除されません。

INSERT 文または UPDATE 文を発行するのみで、内部 LOB 列（データベースに格納されている LOB 列）の行にアクセスし、行を挿入できます。ただし、オブジェクト型の一部である LOB 属性にアクセスし、それを移入するには、EMPTY_CLOB ファンクションまたは EMPTY_BLOB ファンクションを使用して、LOB 属性を初期化してください。その後、DBMS_LOB パッケージまたはその他の適切なインタフェースを使用して、空の LOB 属性を選択し、それを移入できます。

LOB 列には、次の制限事項があります。

- 分散された LOB はサポートされていません。そのため、問合せの SELECT または WHERE 句、あるいは DBMS_LOB パッケージのファンクションで、リモート・ロケータを使用できません。

LOB では、次の構文はサポートされていません。

```
SELECT lobcol FROM table1@remote_site;
INSERT INTO lobtable SELECT type1.lobattr FROM table1@remote_site;
SELECT DBMS_LOB.getlength(lobcol) FROM table1@remote_site;
```

ただし、LOB を参照する別の問合せの一部では、リモート・ロケータを使用できます。リモート LOB 列では、次の構文がサポートされています。

```
CREATE TABLE t AS SELECT * FROM table1@remote_site;
INSERT INTO t SELECT * FROM table1@remote_site;
UPDATE t SET lobcol = (SELECT lobcol FROM table1@remote_site);
INSERT INTO table1@remote_site ...
UPDATE table1@remote_site ...
DELETE table1@remote_site ...
```

副問合せを含む最初の 3 つの文では、SELECT 構文のリストでスタンドアロン LOB 列のみが指定できます。SQL ファンクションまたは DBMS_LOB API は、LOB ではサポートされていません。たとえば、次の文はサポートされています。

```
CREATE TABLE AS SELECT clob_col FROM tab@db2;
```

ただし、次の文はサポートされていません。

```
CREATE TABLE AS SELECT dbms_lob.substr(clob_col) from tab@db2;
```

- クラスタには、キー列または非キー列として LOB を含めることができません。
- LOB の VARRAY は作成できません。
- 問合せの ORDER BY 句、GROUP BY 句または集計ファンクションで LOB 列を指定できません。
- SELECT ... DISTINCT 文、SELECT ... UNIQUE 文または結合で LOB 列を指定できません。ただし、列のオブジェクト型に MAP または ORDER ファンクションが定義されている場合は、SELECT ... DISTINCT 文、あるいは UNION または MINUS 集合演算子を使用する問合せで、オブジェクト型列の LOB 属性を指定することはできます。

- 表の作成時に、オブジェクト型列の属性として NCLOB を指定できません。ただし、メソッドで NCLOB パラメータを指定することはできます。
- ANALYZE ... COMPUTE 文または ANALYZE ... ESTIMATE 文で LOB 列を指定できません。
- AUTO セグメント管理表領域には、LOB を格納できません。
- BEFORE ROW DML トリガーの PL/SQL トリガー本体で、LOB の :old 値を読み込むことはできますが、:new 値は読み込むことができません。ただし、AFTER ROW および INSTEAD OF DML トリガーでは、:new と :old の両方を読み込むことができます。
- LOB 列では UPDATE DML トリガーは定義できません。
- LOB 列は主キー列として指定できません。
- LOB 列は索引キーの一部として指定できません。ただし、ファンクション索引のファンクションまたはドメイン索引の索引タイプ指定では、LOB 列を指定できます。また、Oracle Text では、CLOB 列に索引を定義できます。

参照： ドメイン索引へのトリガーの定義については、『Oracle9i Data Cartridge Developer's Guide』を参照してください。

- INSERT または UPDATE 操作では、任意のサイズのデータを LOB 列にバインドできますが、オブジェクト型の LOB 属性にはデータをバインドできません。INSERT ... AS SELECT 操作では、4000 バイトのデータまで LOB 列にバインドできます。

参照： LOB の使用に関連するその他のセマンティクスについては、このマニュアルの各 SQL 文の「キーワードとパラメータ」の項を参照してください。

- LONG と LOB の両方の列を持つ表では、1 つの SQL 文で両方に 4000 バイトより大きいデータをバインドすることはできません。ただし、いずれか片方にバインドすることは可能です。

注意：

- Oracle8i リリース 8.1.6 以上では、LOB の CACHE READS 設定をサポートしています。このような LOB を以前のリリースにダウングレードする場合は、Oracle は警告を表示し、LOB を CACHE READS から CACHE LOGGING に変換します。この後で、NOCACHE LOGGING または NOCACHE NOLOGGING のいずれかに LOB を変更できます。詳細は、『Oracle9i アプリケーション開発者ガイド - ラージ・オブジェクト』を参照してください。
 - DML トリガーを指定した表で、OCI 関数または DBMS_LOB ルーチンを使用して LOB 列の値またはオブジェクト型列の LOB 属性を変更する場合、Oracle は DML トリガーを起動しません。
-

参照：

- 6-54 ページの「[EMPTY_BLOB](#)、[EMPTY_CLOB](#)」を参照してください。
- イメージ、オーディオ、ビデオおよび空間データのその他の格納方法については、2-38 ページの「[Oracle が提供する型](#)」を参照してください。

次の例では、サンプル表 `pm.print_media` を作成しています。この例では、ネストした表である `textdoc_tab` オブジェクト表が `print_media` 表に存在することを想定しています。

```
CREATE TABLE print_media
( product_id      NUMBER(6)
, ad_id           NUMBER(6)
, ad_composite    BLOB
, ad_sourcetext   CLOB
, ad_finaltext    CLOB
, ad_fltextn      NCLOB
, ad_textdocs_ntab textdoc_tab
, ad_photo        BLOB
, ad_graphic      BFILE
, ad_header       adheader_typ
, press_release   LONG
) NESTED TABLE ad_textdocs_ntab STORE AS textdocs_nestedtab;
```

参照：

- これらのインタフェースおよび LOB の詳細は、『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』および『Oracle Call Interface プログラマーズ・ガイド』を参照してください。
- LONG 列から LOB 列への変換については、10-2 ページの「[ALTER TABLE](#)」の `modify_column_options` 句および 6-166 ページの「[TO_LOB](#)」を参照してください。

BFILE データ型

BFILE データ型を使用すると、Oracle 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。

BFILENAME ファンクションを使用すると、実表のデータに影響を与えずに BFILE のファイル名およびパスを変更できます。

参照： この組込み SQL ファンクションの詳細は、6-21 ページの「[BFILENAME](#)」を参照してください。

バイナリ・ファイル LOB は、トランザクションには関係なく、リカバリができません。ファイルの統合性と耐久性を提供しているのは基本にあるオペレーティング・システムです。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、ファイルが存在し、Oracle のプロセスがファイルに対するオペレーティング・システムの読取り権限を持っていることを確認する必要があります。

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

参照：

- LOB の詳細は、『Oracle9i アプリケーション開発者ガイド - ラージ・オブジェクト』および『Oracle Call Interface プログラマーズ・ガイド』を参照してください。
- 12-44 ページの「[CREATE DIRECTORY](#)」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・セットのセマンティクスを持たないビットストリームとして考えることができます。BLOB 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、DBMS_LOB パッケージまたは OCI を介して行った変更は、すべてトランザクションに反映されます。BLOB 値の操作は、コミットおよびロールバックできます。ただし、1 つのトランザクションの PL/SQL または OCI 変数を BLOB ロケータに保存し、そのロケータを別のトランザクションまたはセッションで使用することはできません。

CLOB データ型

CLOB データ型は、シングルバイトおよびマルチバイト・キャラクタ・データを格納するために使用します。固定幅および可変幅のキャラクタ・セットがサポートされます。両方のキャラクタ・セットで CHAR データベース・キャラクタ・セットを使用します。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、DBMS_LOB パッケージまたは OCI を介して行った変更は、すべてトランザクションに反映されます。CLOB 値の操作は、コミットおよびロールバックできます。ただし、1 つのトランザクションの PL/SQL または OCI 変数を CLOB ロケータに保存し、そのロケータを別のトランザクションまたはセッションで使用することはできません。

NCLOB データ型

NCLOB データ型は、各国語キャラクタ・セットを使用する Unicode データを格納するために使用します。固定幅および可変幅のキャラクタ・セットがサポートされます。NCLOB には、4GB までの文字テキスト・データを格納できます。

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

参照： Unicode データ型のサポートについては、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

ROWID データ型

データベース内の各行にはアドレスがあります。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 進文字列です。16 進文字列のデータ型は ROWID です。また、ROWID データ型を持つ実際の列を含む表やクラスタを作成することもできます。Oracle では、このような列の値が有効な ROWID であることは保証されません。

参照： ROWID 疑似列の詳細は、2-79 ページの「[疑似列](#)」を参照してください。

制限 ROWID

Oracle8 以降、Oracle SQL では、パーティション表、索引、および表領域関連のデータ・ブロック・アドレス (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**制限 ROWID**と呼ばれます。制限 ROWID の書式は次のとおりです。

`block.row.file`

それぞれの意味は、次のとおりです。

- `block` は、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 進文字列です。この文字列の長さは、オペレーティング・システムによって異なります。
- `row` は、データ・ブロック内の行を識別する 4 桁の 16 進文字列です。ブロック内の最初の行は 0 になります。
- `file` は、行を含むデータ・ファイルを識別する 16 進文字列です。最初のデータ・ファイルは 1 になります。この文字列の長さは、オペレーティング・システムによって異なります。

拡張 ROWID

ユーザー列に格納される**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`、`DBA_OBJECTS` および `ALL_OBJECTS` から取り出すことができます。同じセグメントを共有するオブジェクト（たとえば、同じクラスタ内のクラスタ化された表など）には、同じオブジェクト番号が付けられます。

拡張 ROWID は、基本となる 64 という値で格納され、文字 A～Z、a～z、0～9、プラス記号 (+) およびスラッシュ (/) を含めることができます。拡張 ROWID は直接利用できません。拡張 ROWID の内容を解析するには、提供されているパッケージ `DBMS_ROWID` を使用します。パッケージ・ファンクションを使用すると、制限 ROWID から直接利用できる情報、および拡張 ROWID に固有の情報が取り出され、提供されます。

参照： `DBMS_ROWID` パッケージで利用できるファンクション、およびこのファンクションの使用方法は、『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

互換性と移行

制限形式の ROWID は、旧リリースとの互換性を保つために Oracle9i 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 が戻されます。

参照： 互換性と移行の問題については、『Oracle9i データベース移行ガイド』を参照してください。

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

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

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

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

参照：

- UROWID データ型および Oracle がユニバーサル ROWID を生成および操作する方法の詳細は、『Oracle9i データベース概要』および『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。
- データベースの行のアドレスの詳細は、2-31 ページの「[ROWID データ型](#)」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 文では、ANSI データ型、および IBM 社の製品 SQL/DS と DB2 のデータ型も使用できます。Oracle では 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 に示す変換に基づいて、Oracle のデータ型で列データを格納します。

表 2-3 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 (n)	CHAR (n)
CHAR (n)	
CHARACTER VARYING (n)	VARCHAR (n)
CHAR VARYING (n)	
NATIONAL CHARACTER (n)	NCHAR (n)
NATIONAL CHAR (n)	
NCHAR (n)	
NATIONAL CHARACTER VARYING (n)	NVARCHAR2 (n)
NATIONAL CHAR VARYING (n)	
NCHAR VARYING (n)	
NUMERIC (p, s)	NUMBER (p, s)
DECIMAL (p, s) ^a	
INTEGER	NUMBER (38)
INT	
SMALLINT	
FLOAT (b) ^b	NUMBER
DOUBLE PRECISION ^c	
REAL ^d	

^a NUMERIC データ型および DECIMAL データ型では、固定小数点数のみを指定できます。これらのデータ型では、s のデフォルトは 0 です。

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

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

^d REAL データ型は 63 桁の 2 進精度または 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 (n)	CHAR (n)
VARCHAR (n)	VARCHAR (n)
LONG VARCHAR (n)	LONG
DECIMAL (p, s) ^a	NUMBER (p, s)
INTEGER	NUMBER (38)
SMALLINT	
FLOAT (b) ^b	NUMBER

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

^b FLOAT データ型は、バイナリ精度 *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 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は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も注意してください。

ユーザー定義型

ユーザー定義のデータ型は、Oracle 組込みデータ型とその他のユーザー定義のデータ型を、アプリケーション内のデータの構造と動作をモデル化する型の構築ブロックとして使用します。ユーザー定義型を作成する方法は2通りあります。

次の項で、ユーザー定義型の様々なカテゴリを説明します。

参照：

- Oracle 組込みデータ型の詳細は、『Oracle9i データベース概要』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、15-3 ページの「[CREATE TYPE](#)」および 15-24 ページの「[CREATE TYPE BODY](#)」を参照してください。
- ユーザー定義型の使用方法については、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

オブジェクト型

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

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

REF

オブジェクト識別子 (OID) を使用することによって、オブジェクトを一意に識別し、他のオブジェクトまたはリレーショナル表からそのオブジェクトを参照できます。REF と呼ばれるデータ型のカテゴリが、そのような参照を表します。REF 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) 状態であるといいます。DANGLING 状態の REF は、NULL である REF と異なります。REF が DANGLING 状態であるかどうかを確認するには、述語 **IS [NOT] DANGLING** を使用します。たとえば、列型が `customer_typ` (属性 `cust_email` を持つ) を指している REF である `customer_ref` 列があるとします。この `customer_ref` 列を含むオブジェクト・ビュー `oc_orders` (サンプル・スキーマ `oe` 内) は、次のように指定します。

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

VARRAY

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

配列内の要素数は、その配列のサイズを表します。Oracle の配列は可変サイズであるため、**VARRAY** と呼ばれます。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 宣言時に領域は割り当てられません。VARRAY では、次のような型を定義します。

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

通常、Oracle は、1 つの配列オブジェクトをインライン形式でその行の他のデータと同じ表領域に、またはアウトライン形式で LOB に格納します。この形式は配列オブジェクトのサイズによって決まります。ただし、VARRAY に個別の記憶特性を指定する場合、Oracle はこれをサイズに関係なくアウトライン形式で格納します。

参照： 詳細は、14-34 ページの「[CREATE TABLE](#)」の [varray_col_properties](#) を参照してください。

ネストした表

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

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

- リレーショナル表の列
- オブジェクト型属性
- PL/SQL の変数、パラメータ、およびファンクションの戻り値

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

Oracle が提供する型

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

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

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

Any 型

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

SYS.AnyType

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

SYS.AnyData

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

SYS.AnyDataSet

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

参照： これらの型の実装および使用のガイドラインは、『Oracle Call Interface プログラマーズ・ガイド』、『PL/SQL ユーザーズ・ガイドおよびリファレンス』および『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

XML 型

eXtensible Markup Language (XML) は、World Wide Web Consortium (W3C) によって開発された、構造化されたデータおよび構造化されていないデータを Web で表示するための標準書式です。Universal Resource Identifiers (URI) は、Web ページなどの Web 上のリソースを識別します。Oracle は、データベース自体に格納されるデータにアクセスするために DBUri-REF と呼ばれる URI のクラスと同様に、XML および URI データを処理するための型を提供します。また、データベースから外部および内部 URI に格納およびアクセスする新しい型のセットを提供します。

SYS.XMLType

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

SYS.XMLType はシステム定義型であるため、ファンクションの引数として、あるいは表またはビューの列のデータ型として使用できます。表に SYS.XMLType 列を作成する場合、列に関連する XML データを格納するために、Oracle は CLOB を内部的に使用します。すべての CLOB データが TRUE である場合、XML ドキュメント全体に対してのみ更新を実行できます。SYS.XMLType 列には、Oracle Text 索引またはファンクション索引を作成できます。

URI データ型

Oracle は、継承階層で関連する URI 型のファミリー (SYS.UriType、SYS.DBUriType および SYS.HttpUriType) を提供します。SYS.UriType はオブジェクト型であり、その他は SYS.UriType のサブタイプです。

- SYS.HttpUriType を使用すると、外部の Web ページまたはファイルの URL を格納できます。Hypertext Transfer Protocol (HTTP プロトコル) を使用して、これらのファイルにアクセスします。
- SYS.DBUriType を使用すると、データベース内のデータを参照する DBUri-REF を格納できます。SYS.UriType はスーパータイプであるため、この型の列を作成し、SYS.DBUriType または SYS.HttpUriType 型のインスタンスをこの列に格納できます。この場合、データベースの内部または外部に格納されたデータを参照し、常にデータにアクセスできます。

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

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

DBUri-REF は、この仮想 XML ドキュメントの単純な XPath 式です。したがって、従業員番号 205 の従業員の SALARY 値を EMPLOYEES 表で参照するには、DBUri-REF を次のように記述します。

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

このモデルを使用すると、CLOB 列またはその他の列に格納されたデータを参照し、それらのデータを外部の URL として外部の世界へ公開できます。Oracle は、このような URL を解析するための標準の URI サブレットを提供します。このサブレットは、Oracle Servlet Engine でインストールおよび実行できます。

SYS.UriFactoryType

`SYS.UriFactoryType` は、他のオブジェクト型を作成および戻すことができるファクトリ型です。URL 文字列が指定されると、`SYS.UriFactoryType` は、`UriType` の様々なサブタイプのインスタンスを作成できます。URL 文字列を分析し、URL の型 (HTTP、DBUri など) を識別し、サブタイプのインスタンスを作成します。

参照：

- オブジェクト型および型の継承については、『Oracle9i アプリケーション開発者ガイド - オブジェクト・リレーショナル機能』を参照してください。
- 提供される型、その実装および URL サブレットについては、『Oracle9i アプリケーション開発者ガイド - XML』を参照してください。
- Oracle Advanced Queuing での `XMLType` の使用については、『Oracle9i アプリケーション開発者ガイド - アドバンスド・キューイング』を参照してください。
- Oracle サブレットについては、『Oracle9i Servlet Engine Developer's Guide』を参照してください。

空間型

Oracle Spatial のオブジェクト・リレーショナル実装は、一連のオブジェクト・データ型、索引メソッド型およびこれらの型の演算子によって構成されます。

MDSYS.SDO_Geometry

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

参照： この型の実装および使用のガイドラインについては、『Oracle Spatial User's Guide and Reference』を参照してください。

メディア型

Oracle *interMedia* は、マルチメディア・データを記述するために、Java または C++ クラスに似たオブジェクト型を使用します。これらのオブジェクト型のインスタンスは、メタデータおよびメディア・データを含む属性とメソッドで構成されます。Oracle *interMedia* 型は次のとおりです。

ORDSYS.ORDAudio

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

ORDSYS.ORDImage

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

ORDSYS.ORDVideo

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

参照： これらの型の実装および使用のガイドラインは、『Oracle *interMedia* 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。

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

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

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-1997'（1997 年 3 月 29 日）に相当する日付は '05-JAN-1998'（1998 年 1 月 5 日）に相当する日付よりも小さく、'05-JAN-1998 1:35pm'（1998 年 1 月 5 日午後 1 時 35 分）に相当する日付は '05-JAN-1998 10:09am'（1998 年 1 月 5 日午前 10 時 9 分）に相当する日付よりも大きいとみなされます。

文字列値

文字値は、次のどちらかの比較セマンティクスを使用して比較されます。

- 空白埋め比較セマンティクス
- 非空白埋め比較セマンティクス

ここでは、これらの比較セマンティクスについて説明します。

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

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

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

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

単一文字

Oracle は、データベース・キャラクタ・セットで指定さ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、ほぼすべてのキャラクタ・セットでいえることです。

次に、一般的なキャラクタ・セットを示します。

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

ASCII と EBCDIC のキャラクタ・セットの一部を表 2-5 と表 2-6 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 2-5 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 2-6 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 つの比較ファンクションのいずれかを使用して比較されます。どちらのファンクションでもオブジェクト型インスタンスは比較されますが、両者は別のものです。これらのファンクションは、オブジェクト型の一部として指定される必要があります。

参照： MAP メソッドと ORDER メソッド、およびこれらのメソッドが戻す値の詳細は、15-3 ページの「[CREATE TYPE](#)」および『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

VARRAY とネストした表

Oracle9i では、VARRAY とネストした表を比較することはできません。

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 に 5 を掛けた値に 'JAMES' を加えることはできません。ただし、Oracle では値のあるデータ型から別のデータ型へ変換する場合の、暗黙的な変換と明示的な変換をサポートしています。

暗黙的なデータ変換と明示的なデータ変換

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

- 明示的なデータ型変換ファンクションを使用すると、SQL 文がわかりやすくなります。
- 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ります。
- 暗黙的な変換はその変換が行われるコンテキストに依存し、どんな場合でも同じように機能するとはかぎりません。たとえば、日付値から VARCHAR2 型へ値を暗黙的に変換すると、NLS_DATE_FORMAT パラメータに指定されている値によって、予期しない年が戻される場合があります。
- 暗黙的な変換のアルゴリズムは、ソフトウェア・リリースや Oracle 製品の変更によって変更されることがあります。明示的な変換を指定しておくと、その動作は将来的にも確実になります。

暗黙的なデータ変換

あるデータ型から別のデータ型への変換が意味を持つ場合、Oracle は値を自動的に変換します。表 2-7 に、Oracle の暗黙的な変換のマトリックスについて示します。この表は、変換の方向または変換されるコンテキストにかかわらず、すべての可能な変換を示します。詳細は、表の後の説明を参照してください。

表 2-7 暗黙的な型変換のマトリックス

	CHAR	VARCHAR2	DATE	DATETIME/ INTERVAL	LONG	NUMBER	RAW	ROWID	CLOB	BLOB	NCHAR	NVARCHAR	NCLOB
CHAR	—	X	X	X	X	X	X		X		X	X	
VARCHAR2	X	—	X	X	X	X	X	X	X		X	X	
DATE	X	X	—								X	X	
DATETIME/INTERVAL	X	X		—	X						X	X	
LONG	X	X		X	—		X		X		X	X	X
NUMBER	X	X				—					X	X	
RAW	X	X			X		—			X	X	X	
ROWID	X	X						—			X	X	
CLOB	X	X			X				—				
BLOB							X			—			
NCHAR	X	X	X	X	X	X	X	X			—	X	X
NVARCHAR2	X	X	X	X	X	X	X	X			X	—	X
NCLOB					X						X	X	—

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

- INSERT および UPDATE 操作中に、Oracle は変更する列のデータ型に値を変換します。
- SELECT FROM 操作中に、Oracle は列からターゲット変数の型にデータを変換します。
- NUMBER の値と文字の値を比較する場合、Oracle は文字データを NUMBER に変換します。
- DATE の値と文字の値を比較する場合、Oracle は文字データを DATE に変換します。
- SQL ファンクションまたは演算子に不当なデータ型の引数を指定して使用する場合、Oracle はその引数を正当なデータ型に変換します。
- 割当てを実行する場合、Oracle は等号 (=) の右側の値を左側の割当てターゲットのデータ型に変換します
- 連結中に、Oracle は非文字データ型を CHAR または NCHAR に変換します。
- 演算の処理および文字データ型と非文字データ型の比較中に、Oracle はすべての文字データ型を数値、日付または ROWID のいずれかの適切なデータ型に変換します。
CHAR/VARCHAR2 と NCHAR/NVARCHAR2 の演算処理では、Oracle は数値に変換します。
- CHAR/VARCHAR2 と NCHAR/NVARCHAR2 の比較では、異なるキャラクタ・セットが必要な場合があります。このような場合のデフォルトの変換の方向は、データベース・キャラクタ・セットから各国語キャラクタ・セットです。表 2-8 に、異なるキャラクタ・タイプ間での暗黙的な変換の方向を示します。
- ほとんどの SQL 文字ファンクションは、CLOB をパラメータとして指定できます。また、Oracle は CLOB と CHAR 型間で暗黙的な変換を実行します。このため、CLOB を使用できないファンクションは、暗黙的な変換を使用して CLOB を受け入れます。このような場合、Oracle はファンクションが起動される前に CLOB を CHAR または VARCHAR2 に変換します。CLOB が 4000 バイトより大きい場合、Oracle は最初の 4000 バイトのみを CHAR に変換します。

表 2-8 異なるキャラクタ・タイプの変換方向

変換前 →	CHAR	VARCHAR2	NCHAR	NVARCHAR2
変換後 ↓				
CHAR	—	VARCHAR2	NCHAR	NVARCHAR2
VARCHAR2	VARCHAR2	—	NVARCHAR2	NVARCHAR2
NCHAR	NCHAR	NCHAR	—	NVARCHAR2
NVARCHAR2	NVARCHAR2	NVARCHAR2	NVARCHAR2	—

暗黙的なデータ変換の例

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

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

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

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

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

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

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

```
SELECT last_name
       FROM employees
      WHERE ROWID = 'AAAFYmAAFAAAAFGAH';
```

明示的なデータ変換

また、SQL 変換ファンクションを使用すると、データ型の変換を明示的に指定できます。次の表に、値のあるデータ型から別のデータ型に明示的に変換する SQL ファンクションを示します。

表 2-9 明示的な型の変換

変換前 →	CHAR、 VARCHAR2、 NCHAR、 NVARCHAR2	NUMBER	Datetime/ Interval	RAW	ROWID	LONG、 LONG RAW	CLOB、 NCLOB、 BLOB
変換後 ↓							
CHAR、 VARCHAR2、 NCHAR、 NVARCHAR2	TO_CHAR (文字)	TO_NUMBER	TO_DATE	HEXTORAW	CHARTO- ROWID		TO_CLOB
			TO_TIMESTAMP				TO_NCLOB
	TO_NCHAR (文字)		TO_TIMESTAMP_TZ				
			TO_YMINTERVAL				
			TO_DSINTERVAL				
NUMBER	TO_CHAR (数値)	—	TO_DATE				
			NUMTOYMINTERVAL				
	TO_NCHAR (数値)		NUMTODSINTERVAL				
Datetime/ Interval	TO_CHAR (日付)		—				
	TO_NCHAR (日時)						
RAW	RAWTOHEX			—			TO_BLOB
	RAWTONHEX						
ROWID	ROWIDTOCHAR				—		
LONG/ LONG RAW						—	TO_LOB
CLOB、 NCLOB、 BLOB	TO_CHAR						TO_CLOB
	TO_NCHAR						TO_NCLOB

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

参照： すべての明示的な変換ファンクションの詳細は、6-5 ページの「[変換ファンクション](#)」を参照してください。

リテラル

リテラルと定数値という用語の意味は同じで、固定データ値のことです。たとえば、'JACK'、'BLUE ISLAND' および '101' はすべて文字リテラルです。文字リテラルは、単一引用符で囲みます。単一引用符を付けることで、Oracle は文字リテラルとスキーマ・オブジェクト名を区別します。

この項では、次の内容を説明します。

- [テキスト・リテラル](#)
- [整数リテラル](#)
- [数値リテラル](#)
- [期間リテラル](#)

多くの SQL 文とファンクションでは、文字リテラルと数値リテラルを指定する必要があります。式と条件の一部として、リテラルを指定できます。文字リテラルは `'text'` の表記法を、各国文字リテラルは `N'text'` の表記法を、数値リテラルはリテラルのコンテキストにより `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。

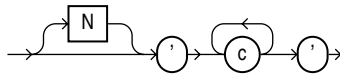
Datetime（日時）または Interval（期間）のデータ型をリテラルとして指定する場合は、データ型に含まれているオプションの精度を考慮する必要があります。Datetime（日時）および Interval（期間）のデータ型をリテラルとして指定する場合の例は、2-2 ページの「[データ型](#)」の関連する項を参照してください。

テキスト・リテラル

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、条件、SQL ファンクションおよび SQL 文に示されている `'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) の構文は次のとおりです。

text::=



それぞれの意味は、次のとおりです。

- `N` は、各国語キャラクタ・セットを使用してリテラルを指定します。この表記法を使用して入力したテキストは、使用時に **Oracle** によって各国語キャラクタ・セットに変換されます。
- `c` は、単一引用符 (') を除く、データベース・キャラクタ・セットの任意の要素です。
- `''` は、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、単一引用符で囲みます。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` データ型の両方のプロパティを持ちます。

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

有効なテキスト・リテラルの例を次に示します。

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

参照：

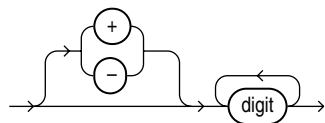
- `expr` の構文の説明は、4-2 ページの「[SQL 式](#)」を参照してください。
- 2-43 ページの「[空白埋め比較セマンティクス](#)」を参照してください。

整数リテラル

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

integer (整数) の構文は次のとおりです。

integer::=



digit は、0、1、2、3、4、5、6、7、8、9 のいずれかです。

整数は最大 38 桁の精度を記憶できます。

有効な *integer* (整数) の例を次に示します。

7
+255

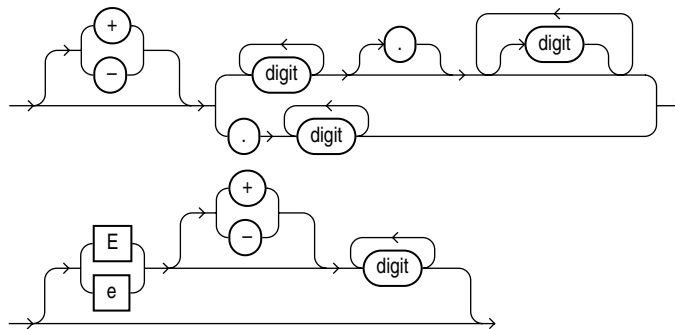
参照： *expr* の構文の説明は、4-2 ページの「SQL 式」を参照してください。

数値リテラル

このマニュアルの他の箇所で、式、条件、SQL ファンクションおよび SQL 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::=



それぞれの意味は、次のとおりです。

- `+` または `-` は、正の値または負の値を示します。符号を指定しない場合、デフォルトは正の値です。
- `digit` は、0、1、2、3、4、5、6、7、8、9 のいずれかです。
- `e` または `E` は、数が科学表記法で指定されることを示します。E の後の数字が指数を示します。指数は -130 ～ 125 の範囲で指定します。

`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` を使用してピリオド (.) 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'
```

参照： 9-2 ページの「[ALTER SESSION](#)」および『Oracle9i データベース・リファレンス』を参照してください。

有効な `number` (数) の例を次に示します。

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

参照： `expr` の構文の説明は、4-2 ページの「[SQL 式](#)」を参照してください。

期間リテラル

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

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

期間リテラルは、主に分析ファンクションとともに使用します。

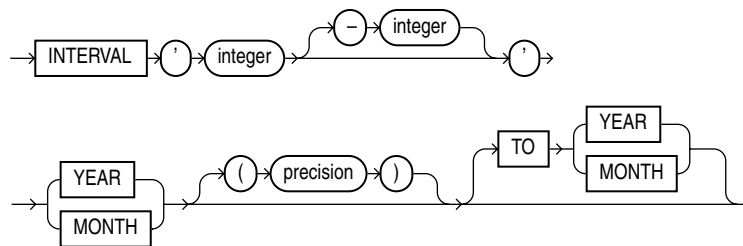
参照：

- 6-8 ページの「[分析ファンクション](#)」および『Oracle9i データ・ウェアハウス・ガイド』を参照してください。
- 6-102 ページの「[NUMTODSINTERVAL](#)」および 6-103 ページの「[NUMTOYMINTERVAL](#)」を参照してください。

INTERVAL YEAR TO MONTH

次の構文を使用して、YEAR TO MONTH 期間リテラルを指定します。

interval_year_to_month::=



それぞれの意味は、次のとおりです。

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

制限事項： 先行フィールドの桁数は、後続フィールドの桁数より多く設定する必要があります。たとえば、INTERVAL '0-1' MONTH TO YEAR は無効です。

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

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

このリテラルの他の書式の例を次に示します。省略バージョンも含みます。

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

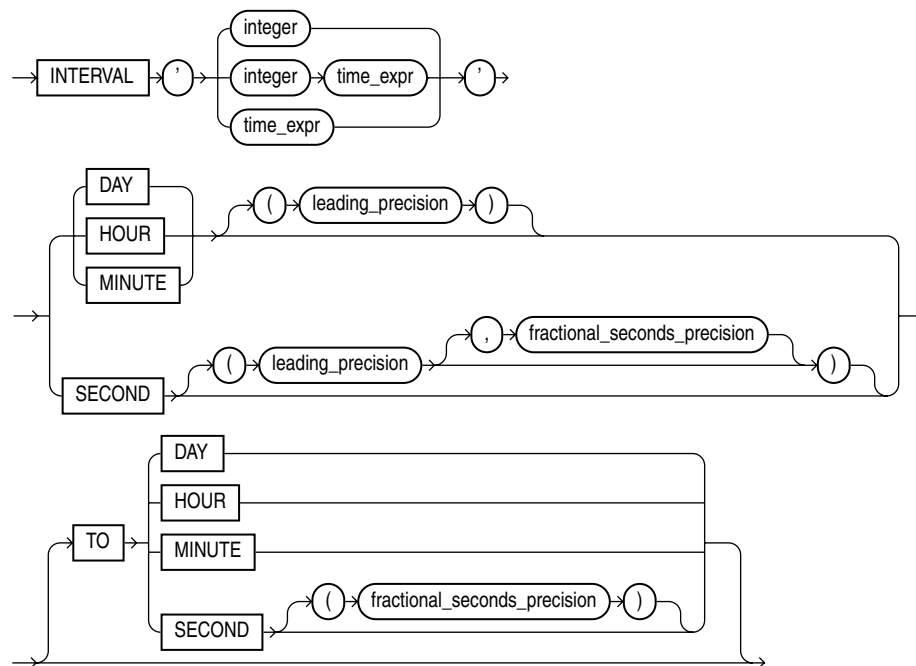
ある INTERVAL YEAR TO MONTH リテラルを別のリテラルに加算または減算して、新しい INTERVAL YEAR TO MONTH リテラルを作成できます。たとえば、次のようにします。

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

INTERVAL DAY TO SECOND

次の構文を使用して、DAY TO SECOND 期間リテラルを指定します。

interval_day_to_second::=



それぞれの意味は、次のとおりです。

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

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

後続フィールドの有効範囲は次のとおりです。

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

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

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

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

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

書式モデル

書式モデルは、文字列に格納される 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。文字列を日付または数値に変換する場合、書式モデルは文字列の変換方法を問いません。SQL 文では、書式モデルは、次の目的で、TO_CHAR ファンクションや TO_DATE ファンクションの引数として使用できます。

- 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します。
- 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します。

注意： 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。

次に例を示します。

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

日付および数値書式モデルの要素のリストは、2-62 ページの表 2-10「数値書式の要素」および 2-67 ページの表 2-12「日時書式要素」を参照してください。

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

参照：

- これらのパラメータの詳細は、『Oracle9i データベース・リファレンス』および『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。
- これらのパラメータの値を変更する場合の詳細は、9-2 ページの「ALTER SESSION」を参照してください。

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

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

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

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

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

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

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

参照： fm 書式要素の説明については、2-73 ページの「[書式モデルの修飾子](#)」を参照してください。

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

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

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

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

この項の後半では、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](#)

参照： 詳細は、6-160 ページの「[TO_CHAR \(日時\)](#)」、6-162 ページの「[TO_CHAR \(数値\)](#)」および 6-164 ページの「[TO_DATE](#)」を参照してください。

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](#) ファンクション
- CHAR データ型または VARCHAR2 データ型の値を NUMBER データ型の値に変換する [TO_NUMBER](#) ファンクション

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

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で構成されます。表 2-10 に数値書式モデルの要素を示します。表 2-11 に、例を示します。

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

表 2-10 数値書式の要素

要素	例	説明
, (カンマ)	9,999	指定した位置にカンマを戻します。1 つの数値書式モデルに複数のカンマを指定できます。 制限事項： <ul style="list-style-type: none">数値書式モデルは、カンマ要素で始めることはできません。数値書式モデルでは、カンマを小数点文字やピリオドの右側に指定できません。
. (ピリオド)	99.99	指定した位置に小数点（ピリオド (.)）を戻します。 制限事項： 1 つの数値書式には、1 つのピリオドのみ指定できます。
\$	\$9999	値の前にドル記号を付けて戻します。
0	0999 9990	先行 0（ゼロ）を戻します。 後続 0（ゼロ）を戻します。
9	9999	正の値の場合は先頭に空白を埋め込み、負の値の場合は先頭に負の符号を埋め込んで、指定の桁数にしてから値を戻します。 固定小数点数の整数部分の場合、先行 0（ゼロ）に対して空白を戻します。ただし、値 0（ゼロ）に対しては 0（ゼロ）を戻します。
B	B9999	整数部が 0（ゼロ）の場合、書式モデル内の 0 にかかわらず、固定小数点数の整数部に対して空白を戻します。
C	C999	指定した位置に ISO 通貨記号（NLS_ISO_CURRENCY パラメータの現在の値）を戻します。

表 2-10 数値書式の要素（続き）

要素	例	説明
D	99D99	指定した位置に小数点文字（NLS_NUMERIC_CHARACTER パラメータの現在の値）を戻します。デフォルトはピリオド（.）です。 制限事項： 1つの数値書式モデルには1つの小数点文字のみ指定できます。
EEEE	9.9EEEE	科学表記法で値を戻します。
FM	FM90.9	前後に空白を付けずに値を戻します。
G	9G999	指定した位置に桁区切り（NLS_NUMERIC_CHARACTER パラメータの現在の値）を戻します。単一の数値書式モデルに複数の桁区切りを指定できます。 制限事項： 数値書式モデルにおいて、桁区切りは小数点文字やピリオドの右側に指定できません。
L	L999	指定した位置にローカル通貨記号（NLS_CURRENCY パラメータの現在の値）を戻します。
MI	9999MI	負の値の後に負の符号（-）を戻します。 正の値の後に空白を戻します。 制限事項： 書式要素 MI は、数値書式モデルの最後の位置にのみ指定できます。
PR	9999PR	山カッコ <> の中に負の値を戻します。 正の値の前後に空白を付けて戻します。 制限事項： 書式要素 PR は、数値書式モデルの最後の位置にのみ指定できます。
RN	RN	大文字のローマ数字で値を戻します。
rn	rn	小文字のローマ数字で値を戻します。 値は1～3999の整数となります。

表 2-10 数値書式の要素（続き）

要素	例	説明
S	S9999	負の値の前に負の符号 (-) を戻します。 正の値の前に正の符号 (+) を戻します。
	9999S	負の値の後に負の符号 (-) を戻します。 正の値の後に正の符号 (+) を戻します。 制限事項： 書式要素 S は、数値書式モデルの最初または最後の位置にのみ指定できます。
TM	TM	テキスト最小値。できるだけ少ない文字数を 10 進数で戻します。 この要素は、大文字と小文字を区別しません。 デフォルトは TM9 です。デフォルトでは、戻り値が 64 文字を超えないかぎり、固定表記法で文字数を戻します。戻り値が 64 文字を超える場合、Oracle は自動的に科学表記法で文字数を戻します。 制限事項： <ul style="list-style-type: none">この要素を他の要素に優先させることはできません。この要素の後には、9、E (1 つ) または e (1 つ) のみを指定できます。
U	U9999	指定した位置にユーロまたは他の第 2 通貨記号 (NLS_DUAL_CURRENCY パラメータの現在の値) を戻します。
V	999V99	値を 10 の ⁿ 乗にして戻します (必要に応じて数値を丸めます)。この例では ⁿ は V の後の 9 の数です。
X	XXXX	指定の桁数の 16 進数値を戻します。指定した値が整数でない場合、Oracle はその値を整数に丸めます。 制限事項： <ul style="list-style-type: none">この要素は、正の値または 0 (ゼロ) のみを受け入れます。負の値の場合、エラーを戻します。この要素は、0 (先行 0 (ゼロ) を戻す) または FM にのみ優先させることができます。その他の要素の場合、エラーを戻します。0 (ゼロ) または FM を X と同時に指定しないと、戻り値の前に常に空白が 1 つ追加されます。
	xxxx	

表 2-11 に、異なる `number` と '`fmt`' に対して次の問合せを行った場合の結果を示します。

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

表 2-11 数値変換例の結果

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

日付書式モデル

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文字値を DATE 値に変換する TO_DATE ファンクション
- デフォルト日付書式以外の書式の DATE 値を文字列に変換する（たとえば、アプリケーションから日付を出力する）TO_CHAR ファンクション

日付書式モデルの合計長は最大 22 文字です。

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。デフォルト日付書式をセッションごとに変更するには、ALTER SESSION 文を使用します。

参照：

- NLS パラメータの詳細は、『Oracle9i データベース・リファレンス』を参照してください。
- 9-2 ページの「ALTER SESSION」を参照してください。

日付書式要素

日時書式モデルは、2-67 ページの表 2-12 に示す、1 つ以上の日付書式要素で構成されます。

- 入力書式モデルの場合、同じ書式項目は 2 回指定できません。また、類似した情報を表す書式項目を組み合わせることもできません。たとえば、'SYYYY' と 'BC' を同一の書式文字列内で使用することはできません。
- 表 2-12 に示すとおり、日時書式要素には、TO_DATE ファンクションで利用できるものと使用できないものがあります。

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位置と同じ位置に現れます。

表 2-12 日時書式要素

要素	TO_* 日時書式で 指定できるか ^a	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	ピリオド付き / なしで西暦を示します。
AM A.M.	はい	ピリオド付き / なしで午前を示します。
BC B.C.	はい	ピリオド付き / なしで紀元前を示します。
CC SCC	いいえ	'1900' の '20' のように、4 桁の年号の上 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「-」が付けられます。
D	はい	曜日 (1 ~ 7)。
DAY	はい	曜日。9 文字分の長さまで空白が埋め込まれます。
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)。
FF		小数部。基数は出力されません (X 書式要素を参照)。 例: 'HH:MI:SS.FF'
HH	はい	時間 (1 ~ 12)。
HH12	いいえ	時間 (1 ~ 12)。
HH24	はい	時間 (0 ~ 23)。

^aTO_* 日時ファンクションとは、TO_CHAR、TO_DATE、TO_TIMESTAMP、TO_TIMESTAMP_TZ、TO_YMINTERVAL および TO_DSINTERVAL です。

表 2-12 日時書式要素（続き）

要素	TO_* 日時書式で 指定できるか ^a	意味
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 規格に基づく 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、整数にしてください。
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埋め込まれます。
PM P.M.	いいえ	ピリオド付き / なしで午前を示します。
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 桁に丸めます。 ■ 指定した年が 50 より小さく、現在の年の下 2 桁が 50 以上の場合、戻された年の上 2 桁は現在の年の上 2 桁よりも 1 多くなります。 ■ 指定した年が 50 以上で、現在の年の下 2 桁が 50 よりも小さい場合、戻された年の上 2 桁は現在の年の上 2 桁よりも 1 少なくなります。 参照: 2-71 ページの表 2-13 を参照してください。
RRRR	はい	年を丸めます。4 桁または 2 桁で入力できます。2 桁の場合、RR の場合と同様の結果が戻ります。年を 4 桁で入力すると、この処理は行われません。
SS	はい	秒（0 ～ 59）。

^aTO_* 日時ファンクションとは、TO_CHAR、TO_DATE、TO_TIMESTAMP、TO_TIMESTAMP_TZ、TO_YMINTERVAL および TO_DSINTERVAL です。

表 2-12 日時書式要素（続き）

要素	TO_*日時書式で 指定できるか ^a	意味
SSSSS	はい	午前 0 時から経過した秒 (0 ~ 86399)。
TZD	はい	夏時間の情報。TZD の値は、夏時間の情報を持つタイム・ゾーン文字列の省略形です。TZR で指定した地域と対応している必要があります。 例： PST (米国 / 太平洋標準時)、PDT (米国 / 太平洋夏時間)
TZH	はい	タイム・ゾーンの時 (後述の TZM 書式要素を参照)。 例： 'HH:MI:SS.FFTZH:TZM'
TZM	はい	タイム・ゾーンの分 (前述の TZH 書式要素を参照)。 例： 'HH:MI:SS.FFTZH:TZM'
TZR	はい	タイム・ゾーン地域の情報。値は、データベースでサポートされるタイム・ゾーン地域である必要があります。 例： US/Pacific
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 日で終了します。
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 日で終了します。
X		ローカル基数文字 例： 'HH:MI:SSXFF'
Y,YYY	はい	指定した位置にカンマを付けた年。
YEAR SYEAR	いいえ	フルスペルで表した年。「S」を指定すると紀元前の日付の先頭に「-」が付けられます。
YYYY SYYYY	はい	4 桁で表した年。「S」を指定すると紀元前の日付の先頭に「-」が付けられます。
YYY YY Y	はい	それぞれ年の下 3 桁、2 桁、1 桁。

^aTO_*日時ファンクションとは、TO_CHAR、TO_DATE、TO_TIMESTAMP、TO_TIMESTAMP_TZ、TO_YMINTERVAL および TO_DSINTERVAL です。

書式文字列に句読点文字がある日付文字列に英数字がある場合、Oracle はエラーを戻します。たとえば、次の場合です。

```
TO_CHAR (TO_DATE('0297','MM/YY'), 'MM/YY')
```

この場合、エラーが戻ります。

日付書式要素およびグローバリゼーション・サポート

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

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

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

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

参照： グローバリゼーション・サポートの初期化パラメータの詳細は、『Oracle9i データベース・リファレンス』および『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

ISO 標準日付書式要素

Oracle は、ISO 規格に従った日時書式要素 IYYYY、IYY、IY、I および IW によって戻される値を計算します。これらの値と日時書式要素 YYYYY、YYY、YY、Y および WW によって戻される値の相違点については、『Oracle9i グローバリゼーション・サポート・ガイド』の「グローバリゼーション・サポート」の章を参照してください。

RR 日付書式要素

RR 日時書式要素は、YY 日時書式要素に似ていますが、20 世紀以外の日付値を格納する場合の柔軟性が優れています。RR 日時書式要素によって、現在が 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、必要に応じて、同じ方法で、21 世紀の時点でも 20 世紀の日付を格納できます。

YY 日時書式要素で TO_DATE ファンクションを使用した場合、日付値は常に現在の日付と同じ上 2 桁で戻されます。そのかわりに RR 日時書式要素を使用した場合、年に指定した 2 桁の数とその年の下 2 桁の数によって戻り値の世紀が変化します。表 2-13 に、RR 日時書式要素の特徴を示します。

表 2-13 RR 日付要素書式

指定された 2 桁の年			
		0 ～ 49	50 ～ 99
現在の年の 下 2 桁	0 ～ 49	戻される日付は、現在の日付と同じ上 2 桁を持ちます。	戻される日付の上 2 桁は、現在の日付の上 2 桁より 1 少なくなります。
	50 ～ 99	戻される日付の上 2 桁は、現在の日付の上 2 桁より 1 大きくなります。	戻される日付は、現在の日付と同じ上 2 桁を持ちます。

次に、RR 日時書式要素の特徴を具体的に説明します。

RR 日付書式の例

次の問合せが、1950 年～ 1999 年の間に発行されるとします。

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

```
Year
----
1998
```

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

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

次の問合せが、2000 年～ 2049 年の間に発行されるとします。

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

Year

1998

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

Year

2017

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

日付書式要素の接尾辞

表 2-14 に、日時書式要素に付加できる接尾辞を示します。

表 2-14 日付書式要素の接尾辞

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

制限事項：

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

書式モデルの修飾子

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

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

FM Fill mode（埋込みモード）です。この修飾子は、TO_CHAR ファンクションの戻り値における空白の埋込みを回避します。

- TO_CHAR ファンクションの日時書式要素では、この修飾子は後続の文字要素（MONTH など）では空白を回避し、日時書式モデルの後続の数要素（MI など）では先行 0（ゼロ）を回避します。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 0（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 ファンクションの数値書式要素では、この修飾子は数値の左に加えられた空白を回避します。これによって、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 ファンクション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）厳密に一致する必要があります。
- 文字引数には余分な空白を含めることはできません。FX を指定していない場合、Oracle は余分な空白を無視します。
- 文字引数における数値データの桁数は、書式モデルの対応する要素と同じ桁数である必要があります。FX を指定していない場合、文字引数における数値により先行 0（ゼロ）が省略されます。

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

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

フォーマット修飾子の例

次の文は、日付書式モデルを使用して文字式を戻します。

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

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

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

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

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

次の文は、2 つの連続した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。

表 2-15 に、char 値と 'fmt' の様々な組合せに対し、次の文が FX を使用したマッチング条件を満たしているかどうかを示します (table という名前の表には DATE データ型の列 date_column があります)。

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

表 2-15 FX 書式モデル修飾子による文字データと書式モデルのマッチング

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

文字列から日付への変換に関する規則

次の追加の書式化規則は、文字列値を日付値に変換する場合に適用されます（ただし、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した場合は適用できません）。

- 先行 0（ゼロ）を含む数値書式要素の桁がすべて指定されている場合は、日付文字列から書式文字列に含まれる句読点を省略できます。たとえば、MM、DD、YY などの 2 桁の書式要素については、2 のかわりに 02 を指定した場合です。
- 日付文字列から、書式文字列の最後にある時刻フィールドを省略できます。
- 日時書式要素と日付文字列内の対応する文字のマッチングに失敗した場合、[表 2-16](#)に示すとおり、元の書式要素のかわりに、別の書式要素の適用が試みられます。

表 2-16 Oracle 書式マッピング

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'

XML 書式モデル

SYS_XMLGEN ファンクションは、XML ドキュメントを含む SYS.XMLType 型のインスタンスを戻します。Oracle は、SYS_XMLGEN ファンクションの結果をフォーマットする XMLGenFormatType オブジェクトを提供します。

表 2-17 に、XMLGenFormatType オブジェクトの属性を示します。表に示すように、ファンクションはこの型を実装します。

参照：

- SYS_XMLGEN ファンクションの詳細は、6-155 ページの「SYS_XMLGEN」を参照してください。
- XMLGenFormatType オブジェクトの実装およびその使用の詳細は、『Oracle9i XML リファレンス』および『Oracle9i アプリケーション開発者ガイド - XML』を参照してください。

表 2-17 XMLGenFormatType オブジェクトの属性

属性	データ型	用途
enclTag	VARCHAR2 (100)	SYS_XMLGEN ファンクションの結果の囲みタグ名です。ファンクションへの入力が列名である場合、デフォルトはその列名です。それ以外の場合は、デフォルトは ROW です。
processingIns	VARCHAR2 (4000)	ファンクションの出力の最上位に、要素の前に追加されるユーザーの処理命令です。

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

```

STATIC MEMBER FUNCTION create(
    enclTag IN varchar2 := null,
    schemaType IN varchar2 := 'NO_SCHEMA'
    schemaName IN varchar2 := null,
    targetNameSpace IN varchar2 := null,
    dburl IN varchar2 := null,
    processingIns IN varchar2 := null)
RETURN XMLGenFormatType;

MEMBER PROCEDURE genSchema(spec IN varchar2);
MEMBER PROCEDURE setSchemaName(schemaName IN varchar2);
MEMBER PROCEDURE setTargetNameSpace(targetNameSpace IN varchar2);
    -- sets the tag name for the ROW element.passing NULL value
    -- supresses ROW element printing, but this is allowed only
    -- if there is only one row in the output or one column per row.
MEMBER PROCEDURE setEnclosingElementName(enclTag IN VARCHAR2);
end;
/
```

NULL

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

NULL は値 0（ゼロ）と同じではないため、0（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 0（ゼロ）の文字値を **NULL** として処理します。ただし、この処理は **Oracle** の今後のバージョンでも継続されるとはかぎらないため、空の文字列を **NULL** として処理しないことをお薦めします。**NULL** を含む算術式は、必ず **NULL** に評価されます。たとえば、**NULL** に 10 を加算しても結果は **NULL** です。実際、オペランドに **NULL** を指定した場合、（連結演算子を除く）すべての演算子は **NULL** を戻します。

SQL ファンクションでの NULL

引数として **NULL** を指定した場合、(**REPLACE**、**NVL** および **CONCAT** を除く) すべてのスカラー・ファンクションでは **NULL** が戻されます。**NVL** ファンクションを使用した場合、**NULL** が発生したときに値を戻すことができます。たとえば、**NVL** (**COMM**, 0) は、**COMM** が **NULL** の場合は 0（ゼロ）を戻し、**COMM** が **NULL** でなければ **COMM** の値を戻します。

ほぼすべての集計ファンクションでは、**NULL** は無視されます。たとえば、1000、**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。

比較条件での NULL

NULL を検査するには、比較条件 **IS NULL** および **IS NOT NULL** のみを使用します。**NULL** を他の条件で使用して、その結果が **NULL** の値に依存する場合、結果は **UNKNOWN** になります。**NULL** はデータの欠落を表すため、任意の値や別の **NULL** との関係で等号や不等号は成り立ちません。ただし、**Oracle** は **DECODE** ファンクションを評価するときに 2 つの **NULL** を等しい値とみなします。

参照： 構文および追加情報の詳細は、6-47 ページの「**DECODE**」を参照してください。

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

条件での NULL

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

表 2-18 は、条件に NULL を含む評価の例です。SELECT 文の WHERE 句で UNKNOWN と評価される条件が使用された場合、その問合せに対して行は戻されません。

表 2-18 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 を含む論理条件の結果を示した真理値表は、5-8 ページの表 5-4、5-9 ページの表 5-5 および 5-9 ページの表 5-6 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、順序の名前で修飾する必要があります。

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

別のユーザーのスキーマ内での順序の現在の値または次の値を参照するには、その順序に対する `SELECT` オブジェクト権限または `SELECT ANY SEQUENCE` システム権限のどちらかが必要です。さらに、その順序は、次に示すとおり、順序を含むスキーマで修飾する必要があります。

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

リモート・データベース上の順序の値を参照するには、次のように、データベース・リンクの完全な名前または部分的な名前で順序を修飾する必要があります。

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

参照： データベース・リンクの参照方法の詳細は、2-113 ページの「[リモート・データベース内のオブジェクトの参照](#)」を参照してください。

順序値の使用場所

次の場所で CURRVAL と NEXTVAL を使用できます。

- 副問合せ、マテリアライズド・ビューまたはビューに含まれていない SELECT 文の SELECT 構文のリスト
- INSERT 文内の副問合せの SELECT 構文のリスト
- INSERT 文の VALUES 句
- UPDATE 文の SET 句

制限事項：次の場所では、CURRVAL と NEXTVAL を使用できません。

- DELETE 文、SELECT 文または UPDATE 文内の副問合せ
- ビューの問合せ、またはマテリアライズド・ビューの問合せ
- DISTINCT 演算子を持つ SELECT 文
- GROUP BY 句または ORDER BY 句を持つ SELECT 文
- 集合演算子 UNION、INTERSECT または MINUS によって別の SELECT 文と結合されている SELECT 文
- SELECT 文の WHERE 句
- CREATE TABLE 文または ALTER TABLE 文の列の DEFAULT 値
- チェック制約の条件

また、CURRVAL または NEXTVAL を使用する単一の SQL 文では、参照された LONG 列、更新された表、ロックされた表がすべて同じデータベース上にある必要があります。

順序値の使用方法

順序を作成するときに、初期値と増分値を定義できます。NEXTVAL の最初の参照によって、順序の初期値が戻されます。その後の参照によって、定義された NEXTVAL 増分値で順序が増加され、その新しい値が戻されます。CURRVAL を参照すると、NEXTVAL への最後の参照で戻された値である、順序の現在の値が常に戻されます。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 文の中では、Oracle は行ごとに 1 回のみ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 1 回のみ順序を増加させ、NEXTVAL が現れるたび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多数のユーザーが同時にアクセスできます。

参照： トランザクションの詳細は、13-81 ページの「[CREATE SEQUENCE](#)」を参照してください。

順序の現在の値の検索例 次の例では、サンプル・スキーマ hr の従業員順序の現在の値を検索します。

```
SELECT employees_seq.currval
FROM DUAL;
```

表への順序値の挿入例 次の例では、従業員順序を増加させ、サンプル表 hr.employees に挿入される新しい従業員のためにその値を使用します。

```
INSERT INTO employees
VALUES (employees_seq.nextval, 'John', 'Doe', 'jdoe',
       '555-1212', TO_DATE(SYSDATE), 'PU_CLERK', 2500, null, null,
       30);
```

順序の現在の値の再利用例 次の例では、次の注文番号を使用して新しい注文をマスター注文表に追加します。その後、この番号を使用して関連する注文をディテール注文表に追加します。

```
INSERT INTO orders (order_id, order_date, customer_id)
VALUES (orders_seq.nextval, TO_DATE(SYSDATE), 106);

INSERT INTO order_items (order_id, line_item_id, product_id)
VALUES (orders_seq.currval, 1, 2359);

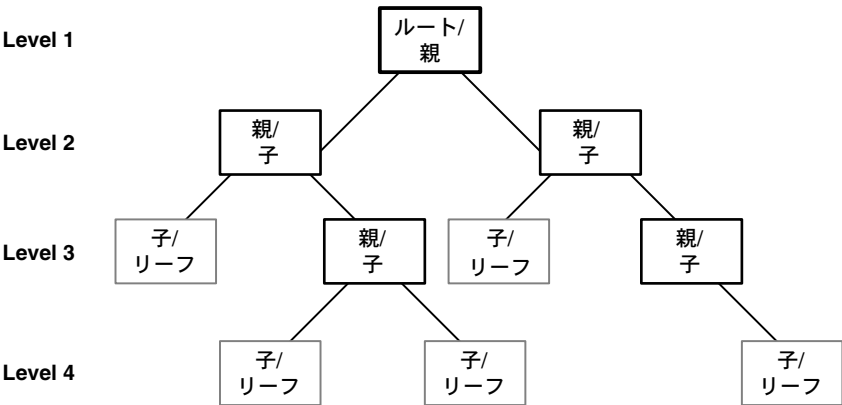
INSERT INTO order_items (order_id, line_item_id, product_id)
VALUES (orders_seq.currval, 2, 3290);

INSERT INTO order_items (order_id, line_item_id, product_id)
VALUES (orders_seq.currval, 3, 2381);
```

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、ルート行に 1 を戻し、ルートの子には 2 を戻します（以降同様に続きます）。**ルート行**は逆ツリー構造の最上位行です。**子である行**は任意の非ルート行です。**親である行**は子を持つ任意の行です。**リーフ行**は子を持たない任意の行です。図 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



問合せの中で階層型の関連を定義するには、START WITH 句および CONNECT BY 句を使用する必要があります。

参照： マテリアライズド・ビューの詳細は、7-3 ページの「[階層問合せ](#)」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 疑似列によって行のアドレスが戻されます。Oracle9i の ROWID 値には、行を検索するために必要な次の情報が含まれています。

- オブジェクトのデータ・オブジェクト番号
- 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 ROWID または UROWID データ型を持ちます。

参照： 2-31 ページの「[ROWID データ型](#)」および 2-33 ページの「[UROWID データ型](#)」を参照してください。

ROWID 値には、次の重要な用途があります。

- 単一の行に最も速くアクセスする方法です。
- 表の行が格納されている様子を示すことができます。
- 表の行に対する一意の識別子です。

表の主キーとして ROWID を使用しないでください。たとえば、インポート・ユーティリティとエクスポート・ユーティリティで行を削除してから再挿入する場合、ROWID が変わる場合があります。行を削除した場合、Oracle は、後から新しく挿入される行にその ROWID を再度割り当てて可能性があります。

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

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

```
SELECT ROWID, last_name
FROM employees
WHERE department_id = 20;
```

ROWNUM

ROWNUM 疑似列は、問合せによって戻される各行について、表や結合処理された行の集合から Oracle が行を選択する順序を示す番号を戻します。つまり、選択される最初の行の ROWNUM は 1、2 番目の行の ROWNUM は 2 です（以降同様に続きます）。

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

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

同じ問合せで ROWNUM に ORDER BY 句が続く場合、ORDER BY 句によって行が再び順序付けられます。結果は、行がアクセスされる方法によって異なります。たとえば、ORDER BY 句の指定によって Oracle が索引を使用してデータにアクセスする場合、索引なしの場合とは異なる順序で行が取り出されることがあります。そのため、後続の文には前述の例と同じ効果はありません。

```
SELECT * FROM employees WHERE ROWNUM < 11 ORDER BY last_name;
```

ORDER BY 句を副問合せに埋め込んで ROWNUM 条件をトップレベル問合せに置いた場合、行の順序付けの後で ROWNUM 条件を強制的に適用させることができます。たとえば、次の問合せは、小さい順から 10 個の従業員数を戻します。これは、上位 N 番までの問合せと呼ばれることがあります。

```
SELECT * FROM
  (SELECT * FROM employees ORDER BY employee_id)
 WHERE ROWNUM < 11;
```

前述の例では、ROWNUM 値はトップレベルの SELECT 文の値です。これらの値は、副問合せ内の employee_id によって行が順序付けられた後で生成されます。

参照： 上位 N 番までの問合せの詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

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

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

最初にフェッチされる行の ROWNUM には 1 が割り当てられるため、条件は偽と判断されません。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、条件も偽と判断されます。このように、後続するすべての行が条件を満たさないため、行は戻されません。

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

```
UPDATE my_table
 SET column1 = ROWNUM;
```

注意： 問合せで ROWNUM を使用した場合、ビューの最適化に影響することがあります。詳細は、『Oracle9i データベース概要』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、パラメータまたは句読点の間に入れることができます。次のどちらかの方法を使用します。

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

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

例 次の文には多くのコメントが含まれています。

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

SELECT last_name,          -- select the name
       salary + NVL(commission_pct, 0),-- total compensation
       job_id,             -- job
       e.department_id     -- and department
FROM employees e,         -- of all employees
     departments d
WHERE e.department_id = d.department_id
```

```
AND salary + NVL(commission_pct, 0) > -- whose compensation
                                         -- is greater than
      (SELECT salary + NVL(commission_pct,0) -- the compensation
FROM employees
WHERE last_name = 'Pataballa')          -- of Pataballa.

;
```

スキーマ・オブジェクトに関するコメント

COMMENT コマンドを使用して、表、ビュー、マテリアライズド・ビューまたは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。

参照： コメントについては、11-66 ページの「[COMMENT](#)」を参照してください。

ヒント

Oracle オプティマイザに指示（**ヒント**）を与えるために、SQL 文中でコメントを使用できます。オプティマイザは、これらのヒントを提案として使用し、文の実行計画を選択します。

文ブロックには、ヒントを含むコメントは1つのみ指定できます。このコメントは、SELECT、UPDATE、INSERT または DELETE のいずれかのキーワードの後でのみ指定できます。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]]...
```

それぞれの意味は、次のとおりです。

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

表 2-19 に、機能のカテゴリに分類したヒントを示します。表の後には、ヒントをアルファベット順に示し、その構文および説明を示します。

注意： Oracle は、誤ったスペルのヒントを標準のコメントとして処理し、エラーを戻しません。

参照： ヒントの詳細は、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』および『Oracle9i データベース概要』を参照してください。

表 2-19 機能のカテゴリに分類したヒント

カテゴリ	ヒント
最適化目標と方法	ALL_ROWS および FIRST_ROWS
	CHOOSE
	RULE
アクセス方法のヒント	AND_EQUAL
	CLUSTER
	FULL
	HASH
	INDEX および NO_INDEX
	INDEX_ASC および INDEX_DESC
	INDEX_COMBINE
	INDEX_FFS
	ROWID
結合順序のヒント	ORDERED
	STAR
結合操作のヒント	DRIVING_SITE
	HASH_SJ、MERGE_SJ および NL_SJ
	LEADING
	USE_HASH および USE_MERGE
	USE_NL

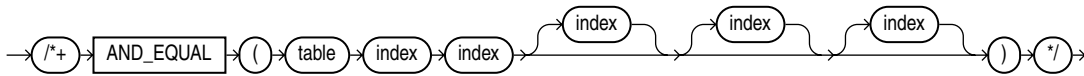
表 2-19 機能のカテゴリに分類したヒント（続き）

カテゴリ	ヒント
パラレル実行のヒント	PARALLEL および NOPARALLEL
	PARALLEL_INDEX
	PQ_DISTRIBUTE
	NOPARALLEL_INDEX
問合せ変換のヒント	FACT および NOFACT
	MERGE
	NO_EXPAND
	NO_MERGE
	REWRITE および NOREWRITE
	STAR_TRANSFORMATION
	USE_CONCAT
その他のヒント	APPEND および NOAPPEND
	CACHE および NOCACHE
	CURSOR_SHARING_EXACT
	NESTED_TABLE_GET_REFS
	UNNEST および NO_UNNEST
	ORDERED_PREDICATES
	PUSH_PRED および NO_PUSH_PRED
	PUSH_SUBQ

all_rows_hint::=



ALL_ROWS ヒントによって、最高のスループットを実現する（リソース使用の合計を最小限にする）という目的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します。

and_equal_hint::=

AND_EQUAL ヒントによって、複数の単一列索引のスキャン結果をマージするアクセス・パスを使用する実行計画を明示的に選択します。

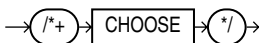
append_hint::=

データベースがシリアル・モードで実行中の場合、APPEND ヒントによって、ダイレクト・パス INSERT を使用可能にできます。(Enterprise Edition を使用していない場合、データベースはシリアル・モードです。シリアル・モードでは、従来の INSERT がデフォルトです。パラレル・モードでは、ダイレクト・パス INSERT がデフォルトです。)

ダイレクト・パス INSERT では、データは、表に割り当てられている既存の領域ではなく、表の最後に追加されます。また、ダイレクト・パス INSERT は、バッファ・キャッシュを回避し、整合性制約を無視します。その結果、ダイレクト・パス INSERT の処理速度は、従来の INSERT の処理速度よりも非常に速くなります。

cache_hint::=

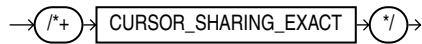
CACHE ヒントによって、フル・テーブル・スキャンの実行時に、この表に取り出されたブロックを、バッファ・キャッシュ内の LRU リストの最高使用頻度側に入れます。このオプションは、小規模な参照表で有効です。

choose_hint::=

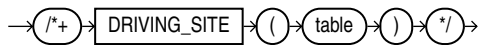
CHOOSE ヒントによって、SQL 文に対して、ルールベース方法とコストベース方法のいずれかをオプティマイザが選択します。オプティマイザは、文によってアクセスされる表の統計情報に基づいて選択します。データ・ディクショナリが、これらの表の 1 つ以上に統計表を持つ場合、オプティマイザはコストベース方法を使用し、最適なスループットを実現するために最適化します。データ・ディクショナリがこれらの表の統計情報を持たない場合、ルールベース方法を使用します。

cluster_hint::=

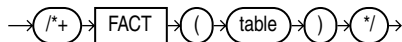
CLUSTER ヒントによって、指定された表にアクセスするために、クラスタ・スキャンを明示的に選択します。このヒントは、クラスタ化されているオブジェクトでのみ適用されます。

cursor_sharing_exact_hint::=

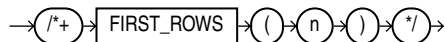
セキュリティ上の問題がなければ、Oracle は、SQL 文のリテラルをバインド変数で置換できます。これは、CURSOR_SHARING 起動パラメータによって制御されます。CURSOR_SHARING_EXACT ヒントによって、この置換処理をオフにできます。つまり、Oracle は、バインド変数によってリテラルを置換せずに SQL 文を実行します。

driving_site_hint::=

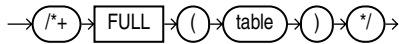
DRIVING_SITE ヒントによって、問合せ処理が、Oracle によって選択されたサイトとは異なるサイトで実行されるようにします。このヒントは、ルールベースまたはコストベースの最適化で使用可能です。

fact_hint::=

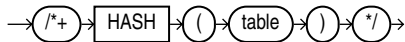
FACT ヒントは、スター型変換のコンテキストで使用され、ヒントに指定された表をファクト表であるとみなすようにスター型変換に指示します。

first_rows_hint::=

FIRST_ROWS(n) (n は正の整数) ヒントまたは FIRST_ROWS によって、Oracle に個々の SQL 文に対する応答を高速化するための最適化を行うように指示します。FIRST_ROWS(n) ヒントは、Oracle に、最初の n 行を最も効果的に戻す計画を選択するように指示するため、高い精度を実現します。FIRST_ROWS ヒントは、最初の 1 行を戻す計画を最適化し、下位互換性およびプラン・スタビリティを保持します。

full_hint::=

FULL ヒントによって、指定された表に対するフル・テーブル・スキャンを明示的に選択します。

hash_hint::=

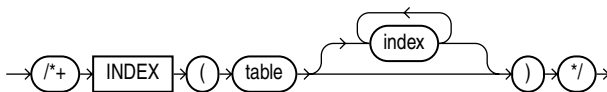
HASH ヒントによって、指定された表にアクセスするために、ハッシュ・スキャンを明示的に選択します。このヒントは、クラスタに格納されている表のみで適用されます。

hash_aj_hint::=

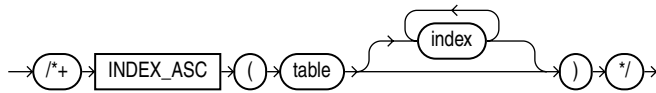
特定の問合せの場合に、NOT IN 副問合せに HASH_AJ、MERGE_AJ または NL_AJ ヒントを指定します。HASH_AJ はハッシュ・アンチジョインを、MERGE_AJ はソート / マージ・アンチジョインを、NL_AJ はネストしたループ・アンチジョインを使用します。

hash_sj_hint::=

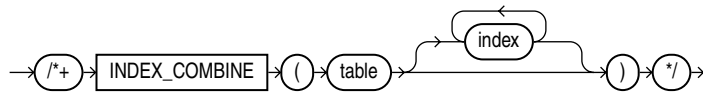
特定の問合せの場合に、EXISTS 副問合せに HASH_SJ、MERGE_SJ または NL_SJ ヒントを指定します。HASH_SJ はハッシュ・セミジョインを、MERGE_SJ はソート / マージ・セミジョインを、NL_SJ はネストしたループ・セミジョインを使用します。

index_hint::=

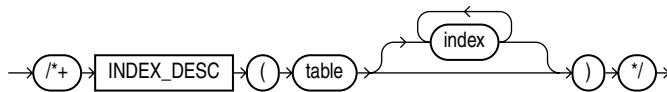
INDEX ヒントによって、指定された表に対する索引スキャンを明示的に選択します。INDEX ヒントは、ドメイン、B ツリー、ビットマップおよびビットマップ結合索引に対して使用できます。ただし、ビットマップ索引に対しては、INDEX ヒントではなく、適応性の高い INDEX_COMBINE ヒントを使用することをお勧めします。

index_asc_hint::=

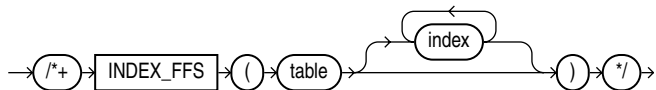
INDEX_ASC ヒントによって、指定された表に対する索引スキャンを明示的に選択します。文が索引レンジ・スキャンを使用する場合、索引付けされた値の昇順に索引エントリをスキャンします。

index_combine_hint::=

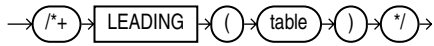
INDEX_COMBINE ヒントによって、表に対するビットマップ・アクセス・パスを明示的に選択します。索引が INDEX_COMBINE ヒントの引数として指定されていない場合は、表にアクセスするための推定コストが最適であるビットマップ索引の組合せをオプティマイザが使用します。特定の索引が引数として指定されている場合、オプティマイザは、指定されたビットマップ索引の組合せを次々に試します。

index_desc_hint::=

INDEX_DESC ヒントによって、指定された表に対する索引スキャンを明示的に選択します。文が索引レンジ・スキャンを使用する場合、索引付けされた値の降順に索引エントリをスキャンします。パーティション索引では、結果はパーティションごとに降順に戻されます。

index_ffs_hint::=

INDEX_FFS ヒントによって、フル・テーブル・スキャンではなく、高速フル索引スキャンを実行します。

leading_hint::=

LEADING ヒントによって、結合順序で最初の表として指定された表を使用できるようにします。

異なる表で LEADING ヒントを指定する場合、すべての LEADING ヒントは無視されます。ORDERED ヒントを指定すると、すべての LEADING ヒントはオーバーライドされます。

merge_hint::=

MERGE ヒントによって、問合せベースでビューをマージできます。

ビューに対する問合せの SELECT 構文のリストに GROUP BY 句または DISTINCT 演算子が含まれる場合、複合ビューがマージ可能であれば、オブティマイザはビューの問合せをアクセス文にマージできます。また、副問合せと相関関係がなければ、IN 副問合せをアクセス文にマージする場合にも複合マージを使用できます。

複合マージはコストベースでは実行されません。アクセス問合せブロックには、MERGE ヒントを含める必要があります。このヒントを含めない場合、オブティマイザは別の方法を使用します。

merge_aj_hint::=

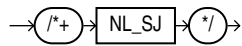
HASH_AJ ヒントを参照してください。

merge_sj_hint::=

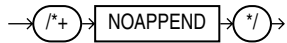
HASH_SJ ヒントを参照してください。

nl_aj_hint::=

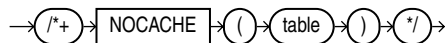
HASH_AJ ヒントを参照してください。

nl_sj_hint::=

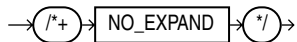
HASH_SJ ヒントを参照してください。

noappend_hint::=

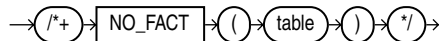
INSERT 文の実行中のパラレル・モードを使用禁止にすると、NOAPPEND ヒントによって従来の INSERT を使用可能にできます（シリアル・モードでは、従来の INSERT がデフォルトです。パラレル・モードでは、ダイレクト・パス INSERT がデフォルトです）。

nocache_hint::=

NOCACHE ヒントによって、フル・テーブル・スキャンの実行時に、この表に取り出されたブロックを、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、バッファ・キャッシュ内のブロックの通常の動作です。

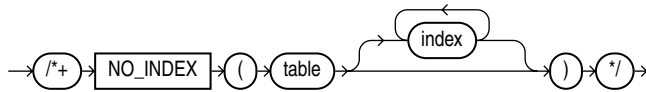
no_expand_hint::=

NO_EXPAND ヒントによって、WHERE 句に OR 条件または IN リストが指定されている問合せに対して、コストベースの 옵ティマイザ가 OR 拡張を検討しないようにします。通常、옵ティマイザは、OR 拡張の使用を検討し、使用しない場合よりコストが低いと判断した場合は、この方法を使用します。

no_fact_hint::=

NO_FACT ヒントは、スター型変換のコンテキストの中で使用され、ヒントに指定された表をファクト表とみなさないように指示します。

no_index_hint::=



NO_INDEX ヒントによって、指定された表に対する一連の索引を明示的に禁止します。

no_merge_hint::=



NO_MERGE ヒントによって、マージ可能なビューをマージしないようにします。

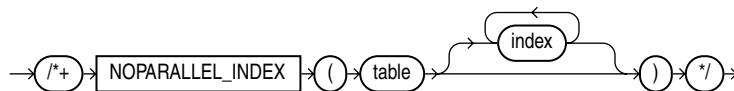
noparallel_hint::=



NOPARALLEL ヒントによって、表内の句の PARALLEL 指定をオーバーライドします。一般的に、ヒントは表内の句より優先されます。

制限事項： ネストした表を含む問合せは、パラレル化できません。

noparallel_index_hint::=



NOPARALLEL_INDEX ヒントによって、パラレル索引スキャン操作を回避するために、索引の PARALLEL 属性をオーバーライドします。

no_push_pred_hint::=



NO_PUSH_PRED ヒントによって、ビューへ述語結合をプッシュしないようにします。

norewrite_hint::=

→ /*+ NOREWRITE */ →

NOREWRITE ヒントによって、QUERY_REWRITE_ENABLED パラメータの設定をオーバーライドして、問合せブロックのクエリー・リライトを使用禁止にします。問合せブロック要求には、NOREWRITE ヒントを使用してください。

no_unnest_hint::=

→ /*+ NO_UNNEST */ →

UNNEST_SUBQUERY パラメータを使用して副問合せのネスト解除を使用可能にした場合、NO_UNNEST ヒントによって、特定の副問合せブロックに対するネスト解除を使用禁止にします。

ordered_hint::=

→ /*+ ORDERED */ →

ORDERED ヒントによって、表が、FROM 句に指定されている順序で結合されるようにします。

結合を実行する SQL 文で ORDERED ヒントを省略した場合、オブティマイザが、表の結合順序を選択します。オブティマイザが選択しない表から選択される行数がわかる場合は、ORDERED ヒントを使用すると、結合順序を指定できる場合があります。この情報によって、オブティマイザより効率的に内部表および外部表を選択できます。

ordered_predicates_hint::=

→ /*+ ORDERED_PREDICATES */ →

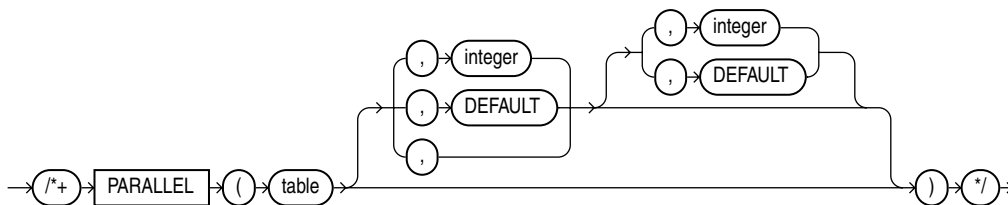
ORDERED_PREDICATES ヒントによって、索引キーとして使用される述語を除く述語の評価順序をオブティマイザが保持できるようにします。SELECT 文の WHERE 句で、このヒントを使用します。

ORDERED_PREDICATES ヒントを使用しない場合、Oracle では、次のルールで指定された順序ですべての述語を評価します。

- 最初に、ユーザー定義ファンクション、型メソッドまたは副問合せのない述語は、WHERE 句に指定されている順序で評価されます。
- 次に、ユーザーが計算したコストを持つユーザー定義ファンクションおよび型メソッドのある述語が、そのコストの昇順で評価されます。
- 次に、ユーザーが計算したコストを持たないユーザー定義ファンクションおよび型メソッドのある述語が、WHERE 句に指定された順序で評価されます。
- 次に、WHERE 句に指定されていない述語（オプティマイザが中間的に生成した述語など）が評価されます。
- 最後に、副問合せのある述語が、WHERE 句に指定されている順序で評価されます。

注意： 前述のとおり、ORDERED_PREDICATES ヒントを使用して、索引キーの述語評価の順序を保持することはできません。

parallel_hint::=



PARALLEL ヒントを使用した場合、パラレル操作に必要な同時サーバーの数を指定できます。ヒントは、INSERT、UPDATE および DELETE 文の一部およびテーブル・スキャン部分に適用されます。

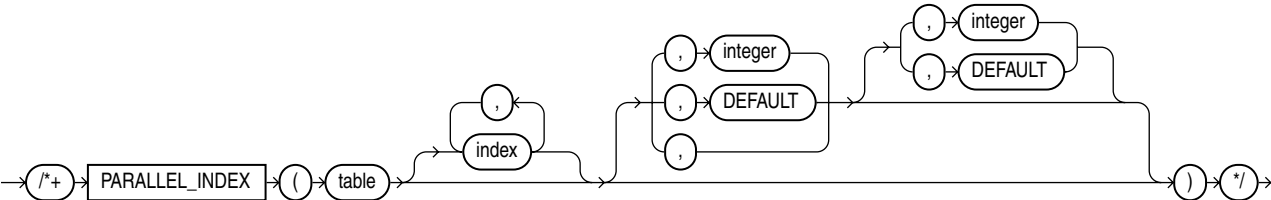
注意： ソートまたはグループ化操作では、使用できるサーバーの数は、PARALLEL ヒントの 2 倍です。

パラレル制限に違反した場合、ヒントは無視されます。

注意： Oracle は、一時表に関するパラレル・ヒントを無視します。

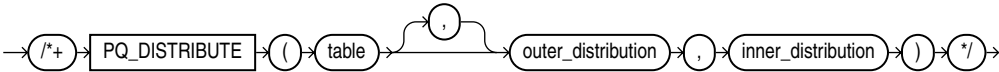
参照： 14-6 ページの「[CREATE TABLE](#)」および『Oracle9i データベース概要』を参照してください。

parallel_index_hint::=



PARALLEL_INDEX ヒントによって、パーティション索引の索引範囲スキャンのパラレル化に必要な同時サーバー数を指定します。

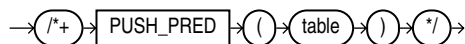
pq_distribute_hint::=



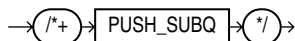
PQ_DISTRIBUTE ヒントによって、パラレル結合操作のパフォーマンスを向上させます。結合された表の行を、生成側の問合せサーバーと受取側の問合せサーバーの間でどのように分散するかを指定します。このヒントによって、オプティマイザが通常行う決定がオーバーライドされます。

EXPLAIN PLAN 文を使用すると、オプティマイザが指定した分散を確認できます。両方の表がシリアルの場合、オプティマイザは分散ヒントを無視します。

参照： 外部および内部結合表への分散で使用可能な組合せについては、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。

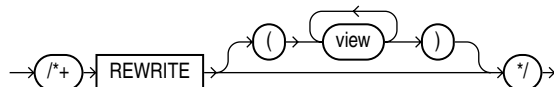
push_pred_hint::=

PUSH_PRED ヒントによって、ビューへの述語結合をプッシュするようにします。

push_subq_hint::=

PUSH_SUBQ ヒントによって、マージされていない副問合せを、実行計画の中でできるだけ早く評価するようにします。一般に、マージされていない副問合せは、実行計画の最後に実行されます。比較的成本が低く、行数を大幅に削減する副問合せの場合、副問合せを早く評価することによってパフォーマンスを改善できます。

副問合せがリモート表に適用される場合、または副問合せがマージ結合を使用して結合される場合は、このヒントを使用しても効果はありません。

rewrite_hint::=

REWRITE ヒントによって、コストを考慮する必要がない場合、コストベースのオプティマイザに、マテリアライズド・ビューの問合せをリライトさせます。ビュー・リストの指定にかかわらず、REWRITE ヒントを使用してください。ビュー・リストを指定して REWRITE ヒントを使用し、そのリストに適切なマテリアライズド・ビューが含まれる場合、Oracle はコストに関係なくそのビューを使用します。

Oracle では、リスト外のビューを考慮しません。ビュー・リストを指定しない場合、Oracle は適切なマテリアライズド・ビューを検索し、コストに関係なくそのビューを使用します。

rowid_hint::=

ROWID ヒントによって、指定された表に対する ROWID によるテーブル・スキャンを明示的に選択します。

rule_hint::=

RULE ヒントによって、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します。これによって、文ブロックに指定されている別のヒントは無視されます。

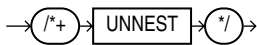
star_hint::=

STAR ヒントによって、スター問合せ計画を強制的に使用します。スター計画の問合せでは、結合順序の最後に最大表があり、その最大表とネステッド・ループ・ジョインを連結索引上で結合します。3 つ以上の表があり、大規模表の連結索引に 3 つ以上の列があり、アクセス方法のヒントまたは結合方法のヒントに矛盾がない場合に、STAR ヒントを適用します。また、オブティマイザは、小規模表の異なる順列についても考慮します。

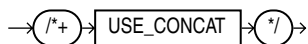
star_transformation_hint::=

STAR_TRANSFORMATION ヒントによって、型変換が適用された最適な計画をオブティマイザが使用するようにします。ヒントを使用しない場合、オブティマイザは、型変換を適用した問合せ用の最適な計画ではなく、型変換を適用せずに生成した最適な計画を使用することを、コストベースで判断します。

ヒントを指定した場合でも、型変換が適用されるとはかぎりません。オブティマイザは、適切であると判断した場合にのみ、副問合せを生成します。副問合せが生成されない場合、型変換が適用された問合せが存在しないため、型変換が適用されない問合せに対する最適な計画が、ヒントの有無にかかわらず使用されます。

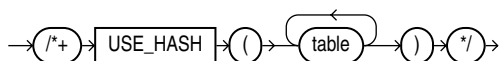
unnest_hint::=

UNNEST_SUBQUERY パラメータが TRUE に設定されている場合、UNNEST ヒントは、副問合せブロックの妥当性のみをチェックします。有効な副問合せブロックであれば、副問合せのネスト解除は使用可能になります。

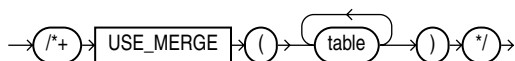
use_concat_hint::=

USE_CONCAT ヒントによって、問合せの WHERE 句に指定されている結合条件 OR を、UNION ALL 集合演算子を使用して複合問合せに変換します。一般的に、この型変換は、連結を使用する問合せのコストが、連結を使用しないコストより少ない場合にのみ発生します。

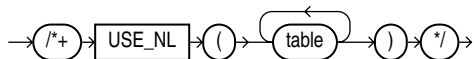
USE_CONCAT ヒントは、IN リストの処理および IN リストを含むすべての論理和に対する OR 拡張をオフにします。

use_hash_hint::=

USE_HASH ヒントによって、ハッシュ結合を使用して、指定された各表を別の行ソースに結合させます。

use_merge_hint::=

USE_MERGE ヒントによって、ソート / マージ結合を使用して、指定された各表を別の行ソースに結合させます。

use_nl_hint::=

USE_NL ヒントによって、ネステッド・ループ・ジョインを使用して、指定された各表を別の行ソースに結合させます。この場合、指定された表を内部表として使用します。

データベース・オブジェクト

次の項で説明するとおり、Oracle は、特定のスキーマに対応付けられたオブジェクトと、特定のスキーマに対応付けられていないオブジェクトを認識します。

スキーマ・オブジェクト

スキーマは、論理的なデータの構造またはスキーマ・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 を使用して作成および操作できます。スキーマ・オブジェクトには次のタイプのオブジェクトがあります。

- クラスタ
- 制約
- データベース・リンク
- データベース・トリガー
- ディメンション
- 外部プロシージャ・ライブラリ
- 索引構成表
- 索引
- 索引タイプ
- Java クラス、Java リソース、Java ソース
- マテリアライズド・ビュー
- マテリアライズド・ビュー・ログ
- オブジェクト表
- オブジェクト型
- オブジェクト・ビュー
- 演算子
- パッケージ
- 順序
- ストアド・ファンクション、ストアド・プロシージャ
- シノニム
- 表
- ビュー

非スキーマ・オブジェクト

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 で作成および操作されますが、スキーマには含まれません。

- コンテキスト
- ディレクトリ
- パラメータ・ファイル (PFILE) およびサーバー・パラメータ・ファイル (SPFILE)
- 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 8 章～第 17 章のデータベース・オブジェクトを作成する文の項で簡単に定義されています。これらの文は、キーワード CREATE で始まります。たとえば、クラスタの定義については、12-2 ページの「CREATE CLUSTER」を参照してください。

参照： データベース・オブジェクトの概要については、『Oracle9i データベース概要』を参照してください。

ほとんどのデータベース・オブジェクトでは、作成時に名前を指定する必要があります。名前は、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、次に示すとおり名前を付けることができる、または名前を付ける必要のある部分で構成されるものもあります。

- 表やビューの中の列
- 索引と表のパーティションおよびサブパーティション
- 表に対する整合性制約
- パッケージ・プロシージャ、パッケージ・ストアド・ファンクション、および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

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

レンジ・メソッドを使用して表または索引をパーティション化する場合、各パーティションのパーティション・キー列の最大値を指定します。リスト・メソッドを使用して表または索引をパーティション化する場合、各パーティションのパーティション・キー列の実際の値を指定します。ハッシュ・メソッドを使用して表または索引をパーティション化する場合、パーティション・キー列に関してシステムが定義するハッシュ・ファンクションに基づいて、Oracle が表の行をパーティションに分割するようにします。コンボジット・パーティション・メソッドを使用して表または索引をパーティション化する場合、パーティションの範囲を指定すると、Oracle はハッシュ・ファンクションに基づいて、各パーティションにある行を1つ以上のハッシュ・サブパーティションに分割します。コンボジット・メソッドを使用してパーティション化された表または索引の各サブパーティションは、同じ論理属性を持ちます。

拡張パーティションおよび拡張サブパーティション名

拡張パーティションおよび拡張サブパーティション名を使用した場合、1つのパーティションまたはサブパーティションのみ、パーティション・レベルおよびサブパーティション・レベルの操作（あるパーティションまたはサブパーティションからのすべての行の削除など）ができます。拡張された名前がない場合、そのような操作には述語（WHERE 句）を指定する必要があります。レンジおよびリスト・パーティション表では、パーティション・レベル操作を述語で表そうとすると（特にレンジ・パーティション・キーで複数の列を使用しているときは）、非常に複雑になる可能性があります。ハッシュ・パーティションおよびサブパーティションの場合、述語の使用はより難しくなります。これは、これらのパーティションおよびサブパーティションが、システムが定義するハッシュ・ファンクションに基づいているためです。

拡張パーティション名を使用した場合、パーティションを表のように使用できます。この方法のメリットは、これらのビューに対する権限を他のユーザーやロールに付与する（または取り消す）ことによって、パーティション・レベルのアクセス制御機構を構築できることです。このメリットは、レンジ・パーティション表に最も有効です。パーティションを表として使用するには、単一のパーティションからデータを選択してビューを作成し、そのビューを表として使用します。

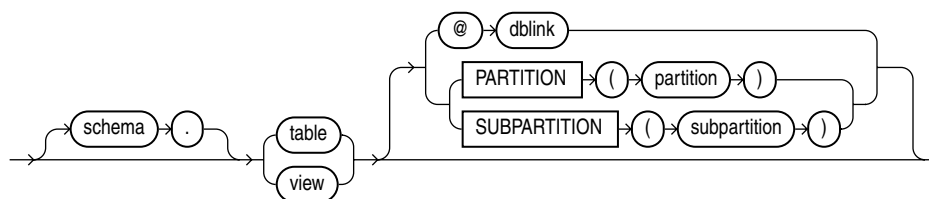
次のデータ操作言語（DML）文では、拡張パーティションまたは拡張サブパーティションの表名を指定できます。

- 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には、ビューを使用することをお勧めします。

構文 拡張パーティション表名および拡張サブパーティション表名を使用する場合の基本的な構文は次のとおりです。

partition_extended_name::=



制限事項 現在、拡張パーティション表名および拡張サブパーティション表名を使用するときには、次の制限があります。

- リモート表は使用できません。拡張パーティション表名または拡張サブパーティション表名には、データベース・リンク (dblink)、または dblink を使用して表に変換するシノニムを含めることはできません。リモート・パーティションおよびリモート・サブパーティションを使用するには、拡張表名の構文を使用してリモート・サイトにビューを作成し、そのリモート・ビューを参照します。
- シノニムは使用できません。拡張パーティションまたは拡張サブパーティションは、実表を使用して指定する必要があります。シノニム、ビューまたはその他のオブジェクトは使用できません。

例 次の文の sales は、パーティション Q1_2000 を持つパーティション表です。単一パーティション Q1_2000 のビューを作成でき、それを表のように使用できます。この例では、パーティションから行が削除されます。

```
CREATE VIEW sales_Q1_2000 AS
  SELECT * FROM sales PARTITION (Q1_2000);
```

```
DELETE FROM sales_Q1_2000 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・オブジェクトの位置修飾子のネーミング規則
- スキーマ・オブジェクトと修飾子のネーミング計画

スキーマ・オブジェクトのネーミング規則

スキーマ・オブジェクトのネーミングには、次の規則があります。

1. 名前は、1 ～ 30 バイトの長さで指定する必要があります。ただし、次の 2 つは例外です。
 - データベースの名前は、8 バイトまでに制限されています。
 - データベース・リンクの名前は、128 バイトまで指定できます。
2. 名前には、引用符を含めることができません。
3. 名前は、大文字と小文字が区別されません。
4. 名前は、二重引用符で囲まれていないかぎり、使用しているデータベース・キャラクタ・セットのアルファベット文字で開始する必要があります。
5. 名前には、使用しているデータベース・キャラクタ・セットの英数字およびアンダースコア (_)、ドル記号 (\$) およびシャープ記号 (#) のみ含めることができます。\$ と # はできるだけ使用しないでください。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。

データベース・キャラクタ・セットがマルチバイト文字を含む場合、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 1 つ以上含めることをお勧めします。

注意： データベース名、グローバル・データベース名またはデータベース・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、ウムラウ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。

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。

参照：

- Oracle の予約語のすべてのリストは、[付録 C「Oracle の予約語」](#)を参照してください。
- 製品の予約語のリストについては、『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。

参照： 7-14 ページの「[DUAL 表からの選択](#)」を参照してください。

8. Oracle の SQL 言語には、特別な意味を持つ文字が含まれています。これらの文字には、データ型、ファンクション名およびキーワード（DIMENSION、SEGMENT、ALLOCATE、DISABLE など、大文字の SQL 文）が含まれます。これらの文字は予約語ではありません。ただし、Oracle はこれらの文字を内部的に使用します。したがって、これらの文字をオブジェクトおよびオブジェクトの部分の名前として使用した場合、使用している SQL 文が読みにくくなり、予期しない結果になることがあります。

特に、「SYS_」で始まる文字をスキーマ・オブジェクト名として使用しないでください。また、SQL 組込みファンクションの名前を、スキーマ・オブジェクトまたはユーザー定義ファンクションの名前として使用しないでください。

参照： 2-2 ページの「[データ型](#)」および 6-2 ページの「[SQL ファンクション](#)」を参照してください。

9. ネームスペース内では、2 つのオブジェクトに同じ名前を付けることはできません。

次のスキーマ・オブジェクトは、1 つのネームスペースを共有します。

- 表
- ビュー
- 順序
- プライベート・シノニム
- スタンドアロン・プロシージャ
- スタンドアロン・ストアド・ファンクション
- パッケージ
- マテリアライズド・ビュー
- ユーザー定義型

次の各スキーマ・オブジェクトは、固有のネームスペースを持ちます。

- 索引
- 制約
- クラスタ
- データベース・トリガー
- プライベート・データベース・リンク
- ディメンション

表およびビューが同じネームスペースにあるため、同じスキーマの表およびビューが同じ名前を持つことはできません。ただし、表と索引は異なるネームスペースに存在します。このため、同じスキーマ内の表と索引には、同じ名前を付けることができます。

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

次の各非スキーマ・オブジェクトは、固有のネームスペースを持ちます。

- ユーザー・ロール
- パブリック・シノニム
- パブリック・データベース・リンク
- 表領域
- ロールバック・セグメント
- プロファイル
- パラメータ・ファイル (PFILE) およびサーバー・パラメータ・ファイル (SPFILE)

これらのネームスペース内のオブジェクトはスキーマに含まれないため、これらのネームスペースはデータベース全体で使用されます。

10. 同じ表やビューでは、複数の列に同じ名前を付けることはできません。ただし、異なる表やビューでは、複数の列に同じ名前を付けることができます。
11. 引数の数およびデータ型が異なる場合、同じパッケージに含まれるプロシージャやファンクションに同じ名前を付けることができます。異なる引数を持ち、同じ名前のプロシージャやファンクションを同じパッケージ内に複数作成することを、オーバーロードといいます。
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～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～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、必ず二重引用符を使用する必要があります。

次の場合は、名前を二重引用符で囲みます。

- 空白を含める場合
- 大文字と小文字を区別する場合
- 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場合
- 英数字、_、\$ および # 以外の文字を名前に含める場合
- 予約語を名前として使用する場合

名前を二重引用符で囲むことによって、同じネームスペース内の異なるオブジェクトに対して次の名前を指定できます。

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

ただし、Oracle は次の名前を同じ名前として解析するため、同じネームスペース内の異なるオブジェクトには、次の名前を使用できません。

```
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・オブジェクトのネーミング例

次に、有効なスキーマ・オブジェクト名の例を示します。

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

列別名、表別名、ユーザー名およびパスワードは、オブジェクトやオブジェクトの部分ではありませんが、同様にこれらのネーミング規則に従う必要があります。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のみで、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。
- パスワードにはネームスペースがないため、規則 9 は適用されません。
- ユーザー名とパスワードで大文字 / 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。

参照： ユーザー名とパスワードのネーミング規則の詳細は、15-30 ページの「[CREATE USER](#)」を参照してください。

スキーマ・オブジェクト名のネーミング計画

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

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

オブジェクトに名前を付ける場合は、短くて簡単な名前とわかりやすい名前のバランスを考えてください。迷ったときには、わかりやすい名前にしてください。これは、データベース内のオブジェクトは、多くの人々が長期間にわたって使用する可能性があるためです。
`payment_due_date` のかわりに `pmdd` という名前を使用すると、10 年後の担当者はデータベースの理解に苦勞することになります。

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

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

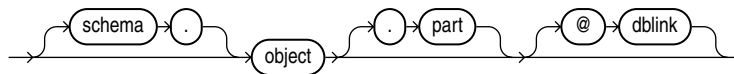
スキーマ・オブジェクトの構文および SQL 文の構成要素

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

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

次に、オブジェクトやそれらの部分を参照するための一般的な構文を示します。

object_part::=



それぞれの意味は、次のとおりです。

- *object* は、オブジェクトの名前です。
- *schema* は、オブジェクトを含むスキーマです。この修飾子を指定することによって、自分のスキーマ以外のスキーマ内のオブジェクトを参照できます。その場合には、自分のスキーマ以外のスキーマ内のオブジェクトを参照するための権限が必要です。この修飾子を指定しないと、自分自身のスキーマ内のオブジェクトを参照するものとみなされます。

スキーマ・オブジェクトのみが *schema* で修飾できます。スキーマ・オブジェクトについては、2-107 ページの規則 9 を参照してください。2-107 ページの規則 9 に示す非スキーマ・オブジェクトはスキーマ・オブジェクトではないため、*schema* では修飾できません。(ただし、パブリック・シノニムは例外で、「PUBLIC」で修飾できます。この場合、引用符が必要です。)

- *part* は、オブジェクトの部分です。この識別子によって、スキーマ・オブジェクトの部分（たとえば、表の列またはパーティション）を参照できます。なお、すべてのタイプのオブジェクトが部分を持っているわけではありません。
- *dblink* は、Oracle の分散オプションを使用している場合にのみ適用されます。オブジェクトを含むデータベースの名前です。この修飾子 *dblink* を指定することによって、ローカル・データベース以外のデータベース内のオブジェクトを参照できます。この *dblink* を指定しないと、自分自身のローカル・データベース内のオブジェクトを参照するものとみなされます。なお、すべての SQL 文でリモート・データベースのオブジェクトにアクセスできるとはかぎりません。

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

Oracle によるスキーマ・オブジェクト参照の変換方法

SQL 文内のオブジェクトが参照される場合、Oracle はその SQL 文のコンテキストを検討して、該当するネームスペース内でそのオブジェクトの位置を確認します。そのオブジェクトの位置を確認してから、そのオブジェクトに対して文の実行を実行します。指定した名前のオブジェクトが適切なネームスペース内に存在しない場合、Oracle はエラーを戻します。

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

```
INSERT INTO departments VALUES (
    280, 'ENTERTAINMENT_CLERK', 206, 1700);
```

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

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

Oracle は、文を発行したユーザーのスキーマ外のネームスペースを考慮する前に、そのユーザーのスキーマ内のネームスペースからオブジェクト参照を変換しようとします。この例では、Oracle は次の方法で名前 departments を変換しようとします。

1. Oracle は、最初に、表、ビューおよびプライベート・シノニムを含む文を発行したユーザーのスキーマ内のネームスペースで、オブジェクトの位置を確認しようとします。オブジェクトがプライベート・シノニムの場合、Oracle はそのシノニムが表すオブジェクトの位置を確認します。このオブジェクトは、ユーザー自身のスキーマ、他のスキーマ、または他のデータベースにあることもあります。このオブジェクトが別のシノニムである場合もあります。その場合、Oracle はそのシノニムが表すオブジェクトの位置を確認します。
2. オブジェクトがネームスペース内に存在する場合、Oracle はそのオブジェクトに対して文を実行しようとします。この例では、Oracle はデータ行を departments に追加しようとします。オブジェクトがその処理にとって正しい型でない場合、Oracle はエラーを戻します。この場合、departments は、表またはビュー、あるいは表またはビューとなるプライベート・シノニムである必要があります。departments が順序である場合、Oracle はエラーを戻します。
3. 前述の処理で検索されたネームスペースにオブジェクトが存在しない場合、Oracle はパブリック・シノニムを含むネームスペースを検索します。オブジェクトがそのネームスペースに存在する場合、Oracle はそのオブジェクトに対して文を実行しようとします。オブジェクトがその処理にとって正しい型でない場合、Oracle はエラーを戻します。この例では、departments が順序のパブリック・シノニムである場合、Oracle はエラーを戻します。

他のスキーマ内のオブジェクトの参照

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

schema.object

たとえば、次の文は、スキーマ hr 内の employees 表を削除します。

```
DROP TABLE hr.employees
```

リモート・データベース内のオブジェクトの参照

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

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

データベース・リンクの作成

12-33 ページの「[CREATE DATABASE LINK](#)」を使用して、データベース・リンクを作成します。この文では、データベース・リンクに関する次の情報を指定できます。

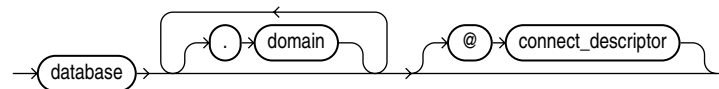
- データベース・リンク名
- リモート・データベースにアクセスするためのデータベース接続文字列
- リモート・データベースに接続するためのユーザー名およびパスワード

これらの情報はデータ・ディクショナリに格納されます。

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

データベース・リンクに付ける名前は、データベース・リンクが参照するデータベースの名前、およびデータベース名の階層内のそのデータベースの位置に一致する必要があります。次に、データベース・リンク名の書式を示します。

dblink::=



それぞれの意味は、次のとおりです。

- **database** には、データベース・リンクが接続するリモート・データベースのグローバル名の名前の部分を指定します。このグローバル名は、リモート・データベースのデータ・ディクショナリに格納されます。この名前は、GLOBAL_NAME ビューで見ることができます。

- `domain` には、データベース・リンクが接続するリモート・データベースのグローバル名のドメイン部分を指定します。データベース・リンクの名前に `domain` を指定しないと、Oracle は、現在、データ・ディクショナリに存在しているローカル・データベースのドメインに、データベース・リンク名を付加します。
- `connect_descriptor` によって、データベース・リンクをさらに修飾できます。接続修飾子を使用する場合、同じデータベースに複数のデータベース・リンクを作成できます。たとえば、接続修飾子を使用して、同じデータベースにアクセスする Real Application Clusters の異なるインスタンスに、複数のデータベース・リンクを作成できます。

`database.domain` の組合せは、「サービス名」と呼ばれることもあります。

参照：『Oracle9i Net Services 管理者ガイド』を参照してください。

ユーザー名およびパスワード リモート・データベースに接続するために、ユーザー名およびパスワードを使用します。データベース・リンクでは、ユーザー名およびパスワードはオプションです。

データベース接続文字列 データベース接続文字列は、Oracle Net がリモート・データベースにアクセスするために使用する仕様です。データベース接続文字列の記述方法については、使用しているネットワーク・プロトコル用の Oracle Net のドキュメントを参照してください。データベース・リンク用のデータベース文字列はオプションです。

データベース・リンクの参照

データベース・リンクは、分散オプションを指定して Oracle を使用している場合にのみ利用できます。データベース・リンクを含む SQL 文の発行時に、次のいずれかの方法でデータベース・リンク名を指定します。

- `complete` は、データ・ディクショナリ内に格納される、`database`、`domain`、およびオプションの `connect_descriptor` コンポーネントを含む完全なデータベース・リンク名を指定します。
- `partial` は、`database`、およびオプションの `connect_descriptor` コンポーネントを含むが、`domain` コンポーネントを含まないように指定します。

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とおり、そのリンク名にローカル・データベースのドメイン名を付加します。現在のグローバル・データベース名は、GLOBAL_NAME データ・ディクショナリ・ビューで見ることができます。
2. Oracle は、最初に、文を発行したユーザーのスキーマ内で、文の中のデータベース・リンクと同じ名前を持つプライベート・データベース・リンクを検索します。必要に応じて、同じ名前を持つパブリック・データベース・リンクを検索します。
 - Oracle は、必ず最初に一致したデータベース・リンク（プライベートまたはパブリック）のユーザー名およびパスワードを採用します。最初に一致したデータベース・リンクに対応付けられているユーザー名およびパスワードがあると、Oracle はそれを使用します。対応付けられているユーザー名およびパスワードがない場合、Oracle は、現在のユーザー名およびパスワードを使用します。
 - 最初に一致したデータベース・リンクに対応付けられているデータベース文字列が存在する場合、Oracle は、そのデータベース文字列を使用します。データベース文字列がない場合、Oracle は一致する次の（パブリック）データベース・リンクを検索します。一致するデータベース・リンクが存在しない場合、または一致するリンクに対応付けられているデータベース文字列が存在しない場合、Oracle はエラーを戻します。
3. Oracle は、リモート・データベースにアクセスするためにデータベース文字列を使用します。リモート・データベースにアクセスした後で、GLOBAL_NAMES パラメータの値が true の場合は、Oracle は、データベース・リンク名の *database.domain* 部分がリモート・データベースの完全なグローバル名に一致しているかどうかを確認します。この条件が満たされている場合、Oracle は手順 2 で選択したユーザー名とパスワードを使用して接続を続行します。それ以外の場合、Oracle はエラーを戻します。
4. データベース文字列、ユーザー名およびパスワードを使用した接続が成功した場合、Oracle は、リモート・データベース上の指定されたオブジェクトにアクセスしようとします。このとき、この項の前半で説明した、オブジェクト参照を変換するための規則、および他のスキーマ内のオブジェクトを参照するための規則が使用されます。

リモート・データベースの完全なグローバル名が、データベース・リンクの *database.domain* 部分と一致する必要があるという要件を無効にするには、初期化パラメータ GLOBAL_NAMES か、ALTER SYSTEM または ALTER SESSION 文の GLOBAL_NAMES パラメータに false を設定します。

参照： リモート・データベースの名前の変換の詳細は、『Oracle9i データベース管理者ガイド』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 文のオブジェクト型の属性とメソッドを参照するには、参照を表の別名で完全に修飾する必要があります。次の例では、`cust_address_typ` 型、および `cust_address_typ` に基づく `cust_address` 列を持つ表 `customers` を含むサンプル・スキーマ `oe` について考えます。

```
CREATE TYPE cust_address_typ AS OBJECT
  ( street_address  VARCHAR2(40)
    , postal_code    VARCHAR2(10)
    , city           VARCHAR2(30)
    , state_province VARCHAR2(10)
    , country_id     CHAR(2)
    );
/
CREATE TABLE customers
  ( customer_id      NUMBER(6)
    , cust_first_name VARCHAR2(20) CONSTRAINT cust_fname_nn NOT NULL
    , cust_last_name  VARCHAR2(20) CONSTRAINT cust_lname_nn NOT NULL
    , cust_address     cust_address_typ
    .
    .
    .
```

次に示すとおり、SQL 文では、`postal_code` 属性への参照は表別名を使用して完全に修飾する必要があります。

```
SELECT c.cust_address.postal_code FROM customers c;

UPDATE customers c SET c.cust_address.postal_code = 'GU13 BE5'
WHERE c.cust_address.city = 'Fleet';
```

引数を取らないオブジェクト型のメンバー・メソッドを参照する場合は、空のカッコを付ける必要があります。たとえば、`distance` が `cust_address` 型のメソッドで、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、空のカッコを付ける必要があります。

```
SELECT c.cust_address.distance() FROM customers c
WHERE c.cust_address.postal_code = '94618';
```

参照： ユーザー定義データ型の詳細は、『Oracle9i データベース概要』を参照してください。

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します。

この章では、次の内容を説明します。

- [SQL 演算子](#)
- [算術演算子](#)
- [連結演算子](#)
- [集合演算子](#)
- [ユーザ定義演算子](#)

SQL 演算子

演算子は、**オペランド**または**引数**と呼ばれる個々のデータ項目を操作します。演算子は、特殊文字またはキーワードで表します。たとえば、乗算演算子は、アスタリスク (*) で表します。

注意： Oracle Text がインストールされている場合は、Oracle Text 問合せで、この製品に含まれる SCORE 演算子を使用できます。この演算子の詳細は、『Oracle Text リファレンス』を参照してください。

単項演算子およびバイナリ演算子

一般に、演算子には次の 2 つのクラスがあります。

- 単項

単項演算子は、1 つのみのオペランドについて操作します。単項演算子の書式は次のとおりです。

`operator operand`
- バイナリ

バイナリ演算子は、2 つのオペランドについて操作します。バイナリ演算子の書式は次のとおりです。

`operand1 operator operand2`

この他、特別な書式を持ち、3 つ以上のオペランドについて操作可能な演算子もあります。演算子のオペランドに NULL が指定された場合、結果は常に NULL になります。この規則に従わない唯一の演算子が連結演算子 (||) です。

演算子の優先順位

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

表 3-1 に、SQL 演算子を優先順位の高い方から順に示します。同じ行にリストされている演算子の優先順位は同じです。

表 3-1 SQL 演算子の優先順位

演算子	操作
+, -	同一、否定
*, /	乗算、除算
+, -,	加算、減算、連結
SQL の条件	5-4 ページの「 条件の優先順位 」を参照してください。

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

```
1+2*3
```

式の中でカッコを使用して演算子の優先順位をオーバーライドできます。Oracle は、カッコの内側の式を評価した後で、外側の式を評価します。

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

算術演算子

算術演算子を式の中で使用することによって、数値を否定（正負を反転）、加算、減算、乗算および除算できます。演算の結果も数値になります。これらの演算子の中には、日付算術で使用するものもあります。表 3-2 に、算術演算子を示します。

表 3-2 算術演算子

演算子	用途	例
+ -	式の正負を示す場合、これらは単項演算子です。	SELECT * FROM order_items WHERE quantity = -1; SELECT * FROM employees WHERE -salary < 0;
	加算、減算を行う場合、これらはバイナリ演算子です。	SELECT commission_pct FROM employees WHERE SYSDATE - hire_date > 365;
* /	乗算、除算を行います。これらはバイナリ演算子です。	UPDATE employees SET salary = salary * 1.1;

二重否定や負の数の減算を表現する場合に、算術式で、連続した負の符号 (--) は使用しないでください。文字 -- は、SQL 文ではコメントの開始を示す場合に使用します。連続した負の符号は、空白またはカッコで区切ってください。

参照： SQL 文中のコメントの詳細は、2-85 ページの「[コメント](#)」を参照してください。

連結演算子

連結演算子は、文字列および CLOB データを操作する場合に使用します。表 3-3 に、連結演算子を示します。

表 3-3 連結演算子

演算子	用途	例
	文字列および CLOB データを連結します。	SELECT 'Name is ' last_name FROM employees;

2 つの文字列を連結した結果は別の文字列になります。両方の文字列が CHAR データ型の場合、結果は CHAR データ型の文字列になり、その最大文字数は 2000 です。どちらかの文字列が VARCHAR2 データ型の場合、結果は VARCHAR2 データ型の文字列になり、最大文字数は 4000 です。どちらかの引数が CLOB データ型の場合、結果は、テンポラリの CLOB になります。データ型が文字列型か CLOB 型かにかかわらず、後続空白は連結後も文字列に残りません。

多くのプラットフォームでは、連結演算子は、表 3-3 に示すとおり 2 本の実線垂直バーで表されます。ただし、IBM 社のプラットフォームの中には、この演算子として破線垂直バーを使用するものもあります。異なるキャラクタ・セットを持つシステム間（たとえば ASCII と EBCDIC 間）で SQL スクリプト・ファイルを移動する場合、垂直バーが、移動先の Oracle 環境で必要な垂直バーに変換されない場合があります。オペレーティング・システムまたはネットワーク・ユーティリティによる変換の制御が困難または不可能である場合に備えて、Oracle では、垂直バー演算子にかわるものとして CONCAT 文字ファンクションが提供されています。異なるキャラクタ・セットを持つ環境間でアプリケーションを移動する場合は、この文字ファンクションを使用することをお勧めします。

Oracle は、長さが 0（ゼロ）の文字列を NULL として処理しますが、長さが 0（ゼロ）の文字列を別のオペランドと連結すると、その結果は常にもう一方のオペランドになります。結果が NULL になるのは、2 つの NULL 文字列を連結したときのみです。ただし、この処理は Oracle の今後のバージョンでも継続されるとはかぎりません。NULL になる可能性がある式を連結する場合は、NVL ファンクションを使用して、その式を長さが 0（ゼロ）の文字列に明示的に変換してください。

参照：

- CHAR データ型と VARCHAR2 データ型の違いの詳細は、2-9 ページの「文字データ型」を参照してください。
- CLOB データ型の詳細は、『Oracle9i アプリケーション開発者ガイド - ラージ・オブジェクト』を参照してください。
- 6-31 ページの「CONCAT」ファンクションおよび 6-104 ページの「NVL」ファンクションを参照してください。

例 次の例では、CHAR 列および VARCHAR2 列を持つ表を作成し、後続空白のある値とない値を挿入してから、これらの値を選択し、連結します。なお、CHAR 列および VARCHAR2 列では、ともに後続空白が保存されます。

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

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

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

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

集合演算子

集合演算子は、2 つのコンポーネントの間合せ結果を 1 つの結果にまとめます。集合演算子を含む間合せを複合間合せと呼びます。表 3-4 に、SQL の集合演算子を示します。これらの演算子の詳細および制限については、7-7 ページの「[UNION \[ALL\]](#)、[INTERSECT](#) および [MINUS 演算子](#)」を参照してください。

表 3-4 集合演算子

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

ユーザー定義演算子

ユーザー定義演算子は、組込み演算子のように、一連のオペランドを入力として受け取り、結果を戻します。ユーザー定義演算子は、ユーザーが `CREATE OPERATOR` 文で作成し、名前で識別されます。これらは、表、ビュー、型およびスタンドアロン・ファンクションとして同じネームスペースに存在します。

新規の演算子を定義すると、他の組込み演算子のように `SQL` 文で使用できます。たとえば、`SELECT` 文の `SELECT` 構文のリスト、`WHERE` 句の条件、`ORDER BY` 句および `GROUP BY` 句でユーザー定義演算子を使用できます。ただし、これはユーザー定義オブジェクトであるため、演算子に対する `EXECUTE` 権限が必要です。

たとえば、指定したキーワードがドキュメントに含まれる場合に、入力としてテキスト・ドキュメントおよびキーワードを受け取り、1 を返す `INCLUDES` 演算子を定義するときは、次のような `SQL` 問合せを作成します。

```
SELECT * FROM emp WHERE includes (resume, 'Oracle and UNIX') = 1;
```

参照： ユーザー定義演算子の詳細は、13-38 ページの「[CREATE OPERATOR](#)」および『[Oracle9i Data Cartridge Developer's Guide](#)』を参照してください。

4

式

この章では、値、演算子およびファンクションを式の中で組み合わせて使用方法について説明します。

この章では、次の内容を説明します。

- SQL 式
- 単純式
- 複合式
- CASE 式
- CURSOR 式
- 日時式
- ファンクション式
- 期間式
- オブジェクト・アクセス式
- スカラー副問合せ式
- 型コンストラクタ式
- 変数式
- 式のリスト

SQL 式

式は、1 つ以上の値、演算子、および値を評価する SQL ファンクションの組合せです。一般に、式のデータ型は、そのコンポーネントのデータ型になります。

次の単純式は、値が 4 になり、データ型は NUMBER（コンポーネントと同じデータ型）になります。

```
2*2
```

次の例は、ファンクションと演算子を使用した複雑な式です。この式は、現在の日付に 7 日を加算し、その合計から時間コンポーネントを削除し、結果を CHAR データ型に変換します。

```
TO_CHAR(TRUNC(SYSDATE+7))
```

次の場所で式を使用できます。

- SELECT 文の SELECT 構文のリスト
- WHERE 句および HAVING 句の条件
- CONNECT BY 句、START WITH 句および ORDER BY 句
- INSERT 文の VALUES 句
- UPDATE 文の SET 句

たとえば、次の UPDATE 文の SET 句で引用符で囲まれた文字列 'Smith' のかわりに式を使用することもできます。

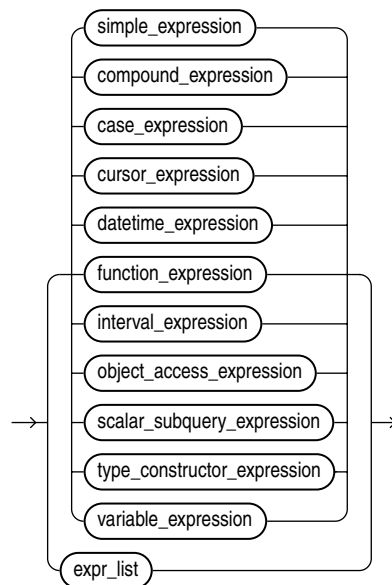
```
SET last_name = 'Smith';
```

この SET 句では、引用符で囲まれた文字列 'Smith' のかわりに、INITCAP(last_name) を使用しています。

```
SET last_name = INITCAP(last_name);
```

次の構文に示すとおり、式にはいくつかの書式があります。

expr::=



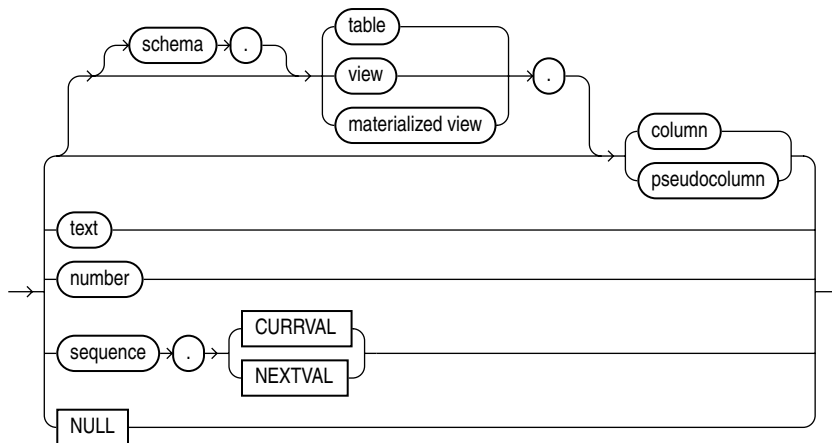
Oracle は、すべての SQL 文のすべての部分で、式のすべての書式を受け入れるわけではありません。このマニュアルの他の箇所で、条件、SQL ファンクションまたは SQL 文に *expr* が示されている場合は、必ず適切な式の表記法を使用してください。次の項では、いくつかの例をあげて、様々な式の書式を説明します。

参照： 文に指定する式の制限については、[第 8 章](#)～[第 17 章](#)の個々の SQL 文の説明を参照してください。

単純式

単純式は、列、疑似列、定数、順序番号または NULL を指定します。

simple_expression::=



スキーマは各ユーザー用の他に、"PUBLIC"（二重引用符が必要）にもなり得ます。その場合、スキーマは表、ビューまたはマテリアライズド・ビューのパブリック・シノニムを修飾する必要があります。"PUBLIC"でのパブリック・シノニムの修飾は、データ操作言語（DML）文でのみサポートされています。データ定義言語（DDL）文ではサポートされていません。

疑似列は、LEVEL、ROWID または ROWNUM のいずれかです。疑似列は、表でのみ使用でき、ビューまたはマテリアライズド・ビューでは使用できません。NCHAR および NVARCHAR2 は、有効な疑似列データ型ではありません。

参照： 疑似列の詳細は、2-79 ページの「[疑似列](#)」を参照してください。

有効な単純式の例を次に示します。

```

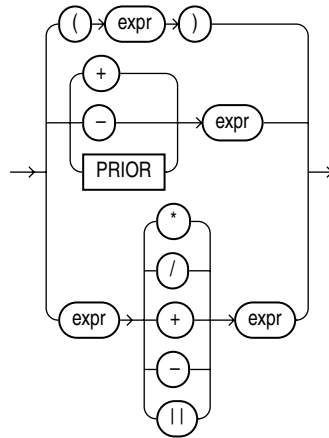
emp.ename
'this is a text string'
10
N'this is an NCHAR string'

```


複合式

複合式は、その他の式の組合せを指定します。

compound_expression::=



ファンクションの組合せによっては、適切でないものや拒否されるものもあるため注意してください。たとえば、LENGTH ファンクションは集計ファンクション内では使用できません。

PRIOR キーワードは、階層問合せの CONNECT BY 句で使用されます。

参照： 7-3 ページの「[階層問合せ](#)」を参照してください。

有効な複合式の例を次に示します。

```
('CLARK' || 'SMITH')
LENGTH('MOOSE') * 57
SQRT(144) + 72
my_fun(TO_CHAR(sysdate, 'DD-MMM-YY'))
```

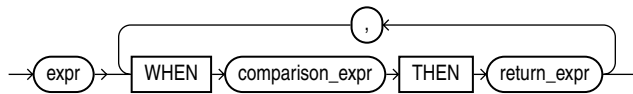
CASE 式

CASE 式を使用すると、プロシージャを起動せずに、SQL 文で IF ... THEN ... ELSE 論理を使用できます。構文は次のとおりです。

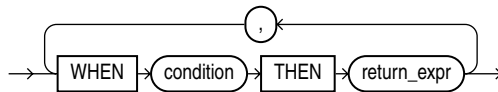
case_expression::=



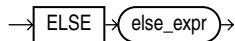
simple_case_expression::=



searched_case_expression::=



else_clause::=



単純 CASE 式では、Oracle は、*expr* と *comparison_expr* が一致する最初の WHEN ... THEN の組合せを検索し、*return_expr* を戻します。WHEN ... THEN の組合せが条件に一致せず、ELSE 句が存在する場合、Oracle は *else_expr* を戻します。それ以外の場合、Oracle は NULL を戻します。*return_expr* および *else_expr* には、リテラル NULL を指定できません。

すべての式 (*expr*、*comparison_expr* および *return_expr*) は、同じデータ型 (CHAR、VARCHAR2、NCHAR または NVARCHAR2) である必要があります。

検索 CASE 式では、Oracle は、*condition* が真である項目を左から右へ検索し、*return_expr* を戻します。真である *condition* がなく、ELSE 句が存在する場合、Oracle は *else_expr* を戻します。それ以外の場合、Oracle は NULL を戻します。

注意： CASE 式の引数の最大数は 255 です。WHEN ... THEN の各組は、2 つの引数として数えます。128 種類の制限を超えないようにするために、CASE 式をネストできます。つまり、`return_expr` 自体を CASE 式にできます。

参照：

- その他の CASE 式の書式については、6-29 ページの「**COALESCE**」および 6-101 ページの「**NULLIF**」を参照してください。
- CASE 式の様々な書式を使用した例は、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

単純 CASE 式の例 次の文は、サンプル表 `oe.customers` のそれぞれの顧客について、クレジット利用限度額を、100 ドルの場合は「Low」、5,000 ドルの場合は「High」、それ以外の場合は「Medium」で表示します。

```
SELECT cust_last_name,
       CASE credit_limit WHEN 100 THEN 'Low'
       WHEN 5000 THEN 'High'
       ELSE 'Medium' END
FROM customers;
```

CUST_LAST_NAME	CASECR
...	
Bogart	Medium
Nolte	Medium
Loren	Medium
Gueney	Medium

検索 CASE 式の例 次の文は、2,000 ドルを最少額の給与として、サンプル表 `oe.employees` の従業員の給与の平均を検出します。

```
SELECT AVG(CASE WHEN e.salary > 2000 THEN e.salary
               ELSE 2000 END) "Average Salary" from employees e;
```

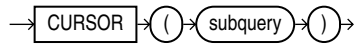
Average Salary

6425

CURSOR 式

CURSOR 式は、ネステッド・カーソルを戻します。この式の書式は、PL/SQL の REF CURSOR と同じで、REF CURSOR 引数としてファンクションに渡せます。

cursor_expression::=



カーソル式が評価されるときに、ネステッド・カーソルは暗黙的にオープンされます。たとえば、カーソル式が SELECT 構文のリストにある場合、問合せによってフェッチされた各行に対して、ネステッド・カーソルがオープンされます。ネステッド・カーソルは、次の場合にのみクローズされます。

- ユーザーによって明示的にクローズされたとき
- 親カーソルが再実行されたとき
- 親カーソルがクローズされたとき
- 親カーソルが取り消されたとき
- 親カーソルの 1 つでのフェッチ時にエラーが発生したとき（クリーンアップの一部としてクローズされる）

制限事項：CURSOR 式には次の制限があります。

- 囲まれる文が SELECT 文以外の文である場合、ネステッド・カーソルは、プロシージャの REF CURSOR 引数としてのみ表示されます。
- 囲まれる文が SELECT 文である場合、ネステッド・カーソルは、問合せ指定の一番外側の SELECT 構文のリストまたは別のネステッド・カーソルの一番外側の SELECT 構文のリストに表示されます。
- ネステッド・カーソルはビューに表示できません。
- ネステッド・カーソルに対して、BIND 操作および EXECUTE 操作は実行できません。

例 問合せの SELECT 構文のリストでの CURSOR 式の使用方法について、次に例を示します。

```

SELECT department_name, CURSOR(SELECT salary, commission_pct
FROM employees e
WHERE e.department_id = d.department_id)
FROM departments d;

```

ファンクションの引数としての CURSOR の使用方法について、次に例を示します。例では、まず、サンプル・スキーマ OE に REF CURSOR 引数を許可するファンクションを作成します。

```
CREATE FUNCTION f (cur SYS_REFCURSOR, mgr_hiredate DATE)
RETURN NUMBER IS
  emp_hiredate DATE;
  before number :=0;
  after number:=0;
begin
  loop
    fetch cur into emp_hiredate;
    exit when cur%NOTFOUND;
    if emp_hiredate > mgr_hiredate then
      after:=after+1;
    else
      before:=before+1;
    end if;
  end loop;
  close cur;
  if before > after then
    return 1;
  else
    return 0;
  end if;
end;
/
```

ファンクションには、カーソルおよび日付を指定できます。ファンクションは、カーソルが日付セットを戻す問合せであることを想定します。次の問合せでは、ファンクションを使用して、サンプル表 employees から、ほとんどの従業員がマネージャよりも前に雇用されているマネージャを検索します。

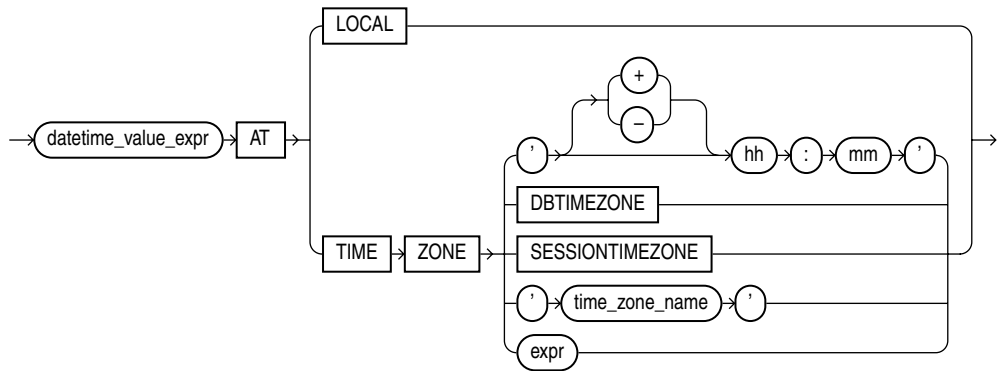
```
SELECT e1.last_name FROM employees e1
WHERE f(
  CURSOR(SELECT e2.hire_date FROM employees e2
  WHERE e1.employee_id = e2.manager_id),
  e1.hire_date) = 1;
```

```
LAST_NAME
-----
De Haan
Mourgos
Cambrault
Zlotkey
Higgins
```

日時式

日時式は、日時データ型の値を戻します。

datetime_expression::=



datetime_value_expression は、日時の値を戻す日時列または複合式です。2-23 ページの表 2-2 で定義される規則に従って、日時および期間を組み合わせたことができます。日時の値を戻す 3 つの組合せは、日時式で有効です。

たとえば、*start_time* に *interval_value_expression* を追加できます。
START_TIME 列を持つ *SCHEDULE* 表について考えます。次の文では、*START_TIME* 列の値に 1 年と 2 か月を追加します。

```
SELECT start_time + INTERVAL '1-2' YEAR TO MONTH FROM schedule;
```

AT LOCAL を指定すると、Oracle は現行のセッションのタイム・ゾーンを使用します。

AT TIME ZONE の設定は、次のように解析されます。

- 文字列 '(+|-)HH:MM': UTC のオフセットとしてタイム・ゾーンを指定します。
- DBTIMEZONE: Oracle は、データベースの作成中に（明示的またはデフォルトで）構築されたデータベース・タイム・ゾーンを使用します。
- SESSIONTIMEZONE: Oracle は、最近の ALTER SESSION 文でデフォルトで構築されたセッション・タイム・ゾーンを使用します。

- `time_zone_name`: Oracle は、`time_zone_name` で指定されたタイム・ゾーンの `datetime_value_expr` を戻します。有効なタイム・ゾーン名を表示するには、`V$TIMEZONE_NAMES` 動的パフォーマンス・ビューに問合せを実行してください。

参照： 動的パフォーマンス・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

- `expr`: `expr` が有効なタイム・ゾーン書式で文字列を戻す場合、Oracle は、そのタイム・ゾーンで入力を戻します。そうでない場合は、エラーが戻ります。

ファンクション式

組込み SQL ファンクションまたはユーザー定義ファンクションを式として使用できます。有効な組込みファンクション式の例を次に示します。

```
LENGTH('BLAKE')
ROUND(1234.567*43)
SYSDATE
```

参照： 組込みファンクションの詳細は、6-2 ページの「SQL ファンクション」および 6-6 ページの「集計ファンクション」を参照してください。

ユーザー定義ファンクション式は、次のものへのコールを指定します。

- オラクル社が提供するパッケージにあるファンクション（『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照）
- ユーザー定義パッケージにあるファンクションまたはスタンドアロン・ユーザー定義ファンクション（6-196 ページの「ユーザー定義ファンクション」を参照）
- ユーザー定義ファンクションおよび演算子（13-38 ページの「CREATE OPERATOR」、12-47 ページの「CREATE FUNCTION」および『Oracle9i Data Cartridge Developer's Guide』を参照）

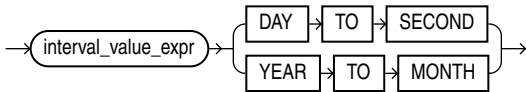
有効なユーザー定義ファンクション式の例を次に示します。

```
circle_area(radius)
payroll.tax_rate(empno)
scott.payrol.tax_rate(dependents, empno)@ny
DBMS_LOB.getlength(column_name)
my_function(DISTINCT a_column)
```

期間式

期間式は、INTERVAL YEAR TO MONTH または INTERVAL DAY TO SECOND の値を戻します。

interval_expression::=



interval_value_expression は、期間の値を戻す期間列または複合式の値です。2-23 ページの表 2-2 で定義される規則に従って、日時および期間を組み合わせることができます。期間の値を戻す 6 つの組合せは、期間式で有効です。

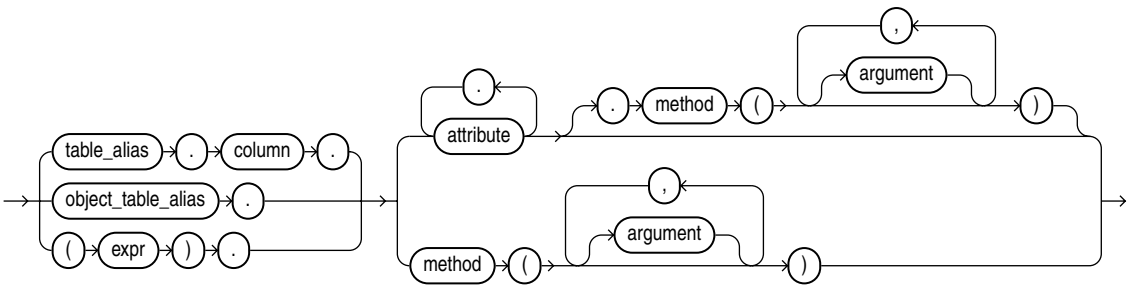
たとえば、次の文は、システム・タイムスタンプ（日時の値）からサンプル表 `orders` の `order_date` 列の値（別の日時の値）を減算して、期間値の式を戻します。

```
SELECT (SYSTIMESTAMP - order_date) DAY TO SECOND from orders;
```

オブジェクト・アクセス式

オブジェクト・アクセス式は、属性の参照およびメソッドの起動を指定します。

object_access_expression::=



`column` パラメータはオブジェクトまたは REF 列です。 `expr` を指定する場合、オブジェクト型に変換する必要があります。

ある型のメンバー・ファンクションが SQL 文のコンテキストでコールされると、SELF 引数が NULL の場合に、Oracle は NULL を戻し、ファンクションは起動されません。

例 次の例では、サンプル・オブジェクト型 `oe.order_item_typ` に基づく表を作成し、オブジェクト列属性から更新および検索する方法を示します。

```
CREATE TABLE short_orders (  
    sales_rep VARCHAR2(25), item order_item_typ);  
  
UPDATE short_orders s SET sales_rep = 'Unassigned';  
  
SELECT o.item.line_item_id, o.item.quantity FROM short_orders o;
```

スカラー副問合せ式

スカラー副問合せ式は、1つの行から1つの列値のみを戻す副問合せです。スカラー副問合せ式の値は、副問合せの **SELECT** 構文リスト項目の値です。副問合せが 0 行を戻す場合、スカラー副問合せ式の値は **NULL** です。副問合せが 2 つ以上の行を戻す場合、**Oracle** はエラーを戻します。

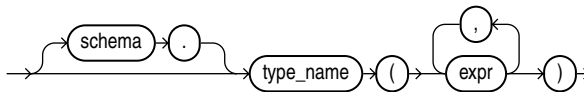
スカラー副問合せ式は、式 (*expr*) をコールするほとんどの構文で使用できます。ただし、次の場所では、スカラー副問合せは無効です。

- 列のデフォルト値
- クラスタのハッシュ式
- DML 文 **RETURNING** 句
- ファンクション索引の基礎
- チェック制約
- **CASE** 式の **WHEN** 条件
- **GROUP BY** 句および **HAVING** 句
- **START WITH** 句および **CONNECT BY** 句
- **CREATE PROFILE** などの問合せに関連しない文

型コンストラクタ式

型コンストラクタ式は、型コンストラクタへのコールを指定します。型コンストラクタの引数は、任意の式です。

type_constructor_expression::=



type_name が**オブジェクト型**の場合、式のリストは、最初の引数の値の型がオブジェクト型の最初の属性と一致する値を取り、2 番目の引数の値の型がオブジェクト型の 2 番目の属性と一致するというように、順序付けられた式のリストになっている必要があります。コンストラクタの引数の合計数は、オブジェクト型の属性の合計数と一致する必要があります。

type_name が **VARRAY 型** または **ネストした表型** の場合、式のリストには 0 個以上の引数を含めることができます。引数が 0 個の場合は、空コレクションの構造であることを示します。それ以外の場合は、各引数が、型がコレクション型の要素型である要素値に対応します。

type_name が**オブジェクト型**、**VARRAY 型**、**ネストした表型**のいずれであっても、引数の最大数は 1000 です。

式の例 次の例では、型コンストラクタのコールに含まれる式の使用方法を示します。

```

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 by Object Type Constructor */
  myaddr address_t = address_t(500, 'Oracle Parkway', 'Redwood Shores', 'CA',
94065);
  /* nested table variable initialized to an empty table by 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;
```

副問合せ例 次に、型コンストラクタへのコールに使用される副問合せの例を示します。

```
CREATE TYPE employee AS OBJECT (
    empno NUMBER,
    ename VARCHAR2(20));

CREATE TABLE empctl of EMPLOYEE;

INSERT INTO empctl VALUES(7377, 'JOHN');

CREATE TYPE project AS OBJECT (
    pname VARCHAR2(25),
    empref REF employee);

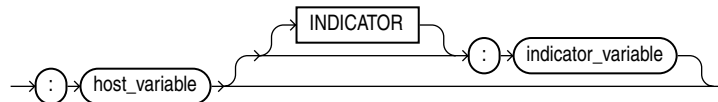
CREATE TABLE depttbl (dno number, proj project);

INSERT INTO depttbl values(10, project('SQL Extensions',
                                         (SELECT REF(p) FROM empctl p
                                          WHERE ename='JOHN')));
```

変数式

変数式は、オプションのインジケータ変数を持つホスト変数を指定します。この書式の式は、埋込み SQL 文または Oracle Call Interface (OCI) プログラムで処理される SQL 文のみで指定できます。

variable_expression::=



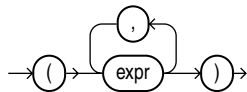
有効な変数式の例を次に示します。

```
:employee_name INDICATOR :employee_name_indicator_var
:department_location
```

式のリスト

式のリストは、カンマで区切られた一連の式です。全体は、カッコで囲みます。

expr_list::=



式のリストには 1000 個まで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)
```

条件は、1 つ以上の式と論理演算子の組合せで指定します。

この章では、次の内容を説明します。

- [SQL 条件](#)
- [比較条件](#)
- [論理条件](#)
- [メンバーシップ条件](#)
- [範囲条件](#)
- [NULL 条件](#)
- [EXISTS 条件](#)
- [LIKE 条件](#)
- [IS OF type 条件](#)
- [複合条件](#)

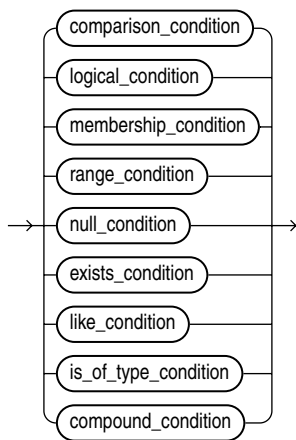
SQL 条件

条件は、1 つ以上の式と論理演算子を組み合わせ、TRUE、FALSE または UNKNOWN のいずれかの値を返します。

条件の種類

条件には、次の構文で示すとおり、複数の書式があります。

condition::=



注意： Oracle Text がインストールされている場合、CONTAINS、CATSEARCH および MATCHES などのこの製品に含まれる組込み条件を使用できます。Oracle Text 要素の詳細は、『Oracle Text リファレンス』を参照してください。

次の項では、様々な書式の条件を説明します。SQL 文に *condition* が含まれる場合は、適切な条件構文を使用する必要があります。

条件は、次の文の WHERE 句で使用できます。

- DELETE
- SELECT
- UPDATE

また、SELECT 文の次の句で 사용할 수도 있습니다。

- WHERE
- START WITH
- CONNECT BY
- HAVING

조건은, 논리 데이터형이라고도 할 수 있습니다.ただし、Oracle で、正式にこのようなデータ型をサポートしているわけではありません。

次のような単純な 조건은,常に TRUE に 평가됩니다。

```
1 = 1
```

次のやや複雑な 조건은, salary の値を salary*commission_pct の値に加算し (NULL は 0 で置き換える)、その合計が定数 25000 より大きいかどうかを判断します。

```
NVL(salary, 0) + NVL(salary*commission_pct, 0) > 25000
```

논리 조건을 사용하면,複数の 조건을 단일의 조건에 결합할 수 있습니다.たとえば、次のように AND 条件を使用して 2 つの 조건을 결합할 수 있습니다。

```
(1 = 1) AND (5 < 7)
```

有効な 조건을 예에次に示します。

```
name = 'SMITH'  
employees.department_id = departments.department_id  
hire_date > '01-JAN-88'  
job_id IN ('SA_MAN', 'SA_REP')  
salary BETWEEN 5000 AND 10000  
commission_pct IS NULL AND salary = 2100
```

参照： 文に指定する条件の制限については、第 8 章～第 17 章にある各文の説明を参照してください。

条件の優先順位

優先順位とは、同じ式の中の異なる条件を **Oracle** が評価する順序を意味します。複数の条件を含む式を評価するとき、**Oracle** は優先順位の高い条件を評価した後で、優先順位の低い条件を評価します。優先順位の等しい条件は、式の中で左から右に評価されます。

表 5-1 に、SQL 条件を優先順位の高い方から順に示します。同じ行に示されている条件の優先順位は同じです。表に示すとおり、**Oracle** は条件の前に演算子を評価します。

表 5-1 SQL 条件の優先順位

条件	用途
SQL 演算子	3-2 ページの「 演算子の優先順位 」を参照してください。
=、!=、<、>、<=、>=	比較
IS [NOT] NULL、LIKE、[NOT] BETWEEN、[NOT] IN、EXISTS、IS OF <i>type</i>	比較
NOT	指数、論理否定
AND	論理積
OR	論理和

比較条件

比較条件は、2 つの式を比較します。比較の結果は、真 (TRUE)、偽 (FALSE) または不明 (UNKNOWN) になります。

注意： ラージ・オブジェクト (LOB) は、比較条件ではサポートされていません。ただし、CLOB データの比較では、PL/SQL プログラムを使用できます。

表 5-2 に、比較条件を示します。

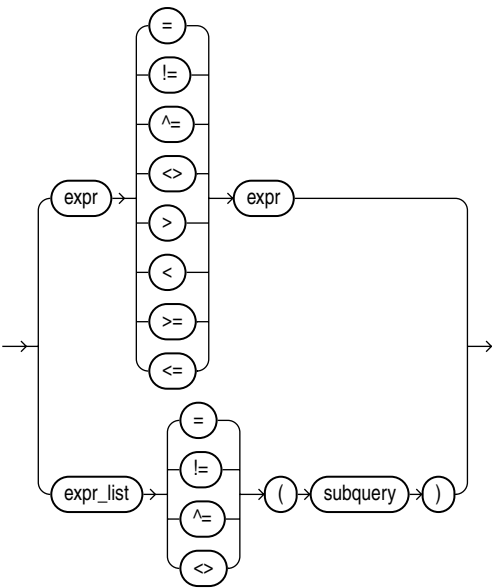
表 5-2 比較条件

条件	用途	例
=	等価性をテストします。	<pre>SELECT * FROM employees WHERE salary = 2500;</pre>
!= ^= <> ¬=	不等性をテストします。ブラットフォームによっては、一部の不等号条件を使用できない場合があります。	<pre>SELECT * FROM employees WHERE salary != 2500;</pre>
>	大 / 小をテストします。	<pre>SELECT * FROM employees WHERE salary > 2500;</pre>
<		<pre>SELECT * FROM employees WHERE salary < 2500;</pre>
>=	以上 / 以下をテストします。	<pre>SELECT * FROM employees WHERE salary >= 2500;</pre>
<=		<pre>SELECT * FROM employees WHERE salary <= 2500;</pre>
ANY SOME	リスト内の各値または問合せによって戻される各値と、ある値を比較します。=、!=、>、<、<=、>= のいずれかを先に指定する必要があります。	<pre>SELECT * FROM employees WHERE salary = ANY (SELECT salary FROM employees WHERE department_id = 30);</pre>
	問合せによって行が戻されない場合には、FALSE と評価されます。	
ALL	リスト内のすべての値または問合せによって戻されるすべての値と、ある値を比較します。=、!=、>、<、<=、>= のいずれかを先に指定する必要があります。	<pre>SELECT * FROM employees WHERE salary >= ALL (1400, 3000);</pre>
	問合せによって行が戻されない場合には、TRUE と評価されます。	

単純比較条件

単純比較条件は、式または副問合せの結果の比較方法を指定します。

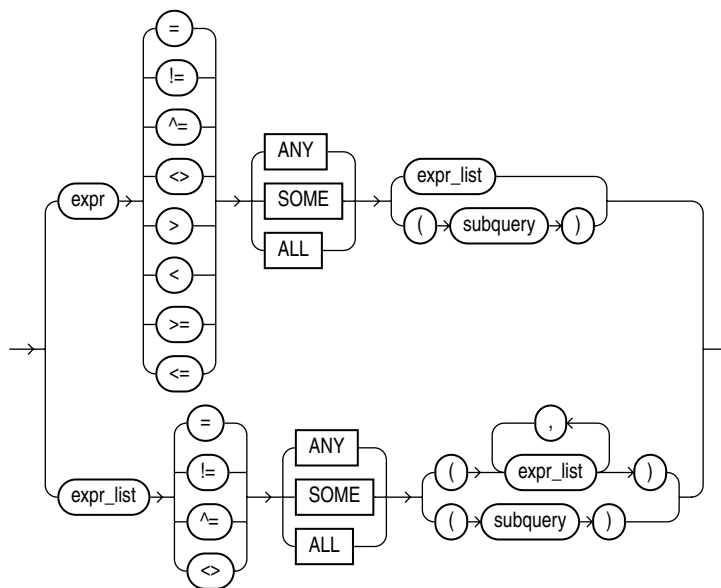
simple_comparison_condition::=



グループ比較条件

グループ比較条件は、リストまたは副問合せ内の任意またはすべてのメンバーの比較方法を指定します。

group_comparison_condition::=



参照： 17-4 ページの「[SELECT](#)」を参照してください。

論理条件

論理条件は、2つのコンポーネント条件の結果を組み合わせ、それらに基づいて単一の結果を生成したり、単一の条件の結果を反転させます。表 5-3 に、論理条件を示します。

表 5-3 論理条件

条件	操作	例
NOT	後続する条件が FALSE の場合に TRUE を返します。TRUE の場合には FALSE を返します。UNKNOWN の場合には UNKNOWN を返します。	<pre>SELECT * FROM employees WHERE NOT (job_id IS NULL); SELECT * FROM employees WHERE NOT (salary BETWEEN 1000 AND 2000);</pre>
AND	コンポーネントの条件が両方とも TRUE の場合に TRUE を返します。どちらかが FALSE の場合には FALSE を返します。それ以外の場合は UNKNOWN を返します。	<pre>SELECT * FROM employees WHERE job_id = 'PU_CLERK' AND department_id = 30;</pre>
OR	コンポーネントの条件のどちらかが TRUE の場合に TRUE を返します。両方とも FALSE の場合は FALSE を返します。それ以外の場合は UNKNOWN を返します。	<pre>SELECT * FROM employees WHERE job_id = 'PU_CLERK' OR department_id = 10;</pre>

表 5-4 に、式に NOT 条件を適用した結果を示します。

表 5-4 NOT 真理値表

	TRUE	FALSE	UNKNOWN
NOT	FALSE	TRUE	UNKNOWN

表 5-5 に、2 つの式に AND 条件を組み合わせた結果を示します。

表 5-5 AND 真理値表

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

たとえば、次の SELECT 文の WHERE 句は、AND 論理条件を使用して、1989 年より前に入社し、月給が 2,500 ドルを超える従業員のみが選択されるように指定しています。

```
SELECT * FROM employees
WHERE hire_date < TO_DATE('01-JAN-1989', 'DD-MON-YYYY')
      AND salary > 2500;
```

表 5-6 に、2 つの式に OR を適用した結果を示します。

表 5-6 OR 真理値表

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

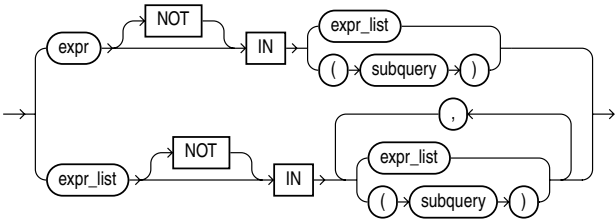
たとえば、次の問合せは、歩合率が 40% または給与が 20,000 ドル以上の従業員を戻します。

```
SELECT employee_id FROM employees
      WHERE commission_pct = .4 OR salary > 20000;
```

メンバーシップ条件

メンバーシップ条件は、リストまたは副問合せ内のメンバーシップをテストします。

membership_condition::=



メンバーシップ条件を使用するときには、次のことに注意してください。

- [NOT] IN の後に、副問合せが 1 回のみ許可されます。
- メンバーシップ条件で IN の左側に複数の式がある場合、[NOT] IN の後に 1 つ以上のリスト (expr_list) を指定できます。ただし、1 つの expr_list のみ指定する場合を含め、常に外側をカッコで囲む必要があります。したがって次のようにします。

```
... WHERE (a, b) IN (subquery);
... WHERE (a, b) IN ((c, d), (e, f));
... WHERE (a, b) IN ((c, d));
```

表 5-7 に、メンバーシップ条件を示します。

表 5-7 メンバーシップ条件

条件	操作	例
IN	メンバーとの等価性をテストします。「= ANY」と同じです。	SELECT * FROM employees WHERE job_id IN ('PU_CLERK', 'SH_CLERK'); SELECT * FROM employees WHERE salary IN (SELECT salary FROM employees WHERE department_id = 30);
NOT IN	「!=ALL」と同じです。メンバーのいずれかが NULL の場合には、FALSE と評価されます。	SELECT * FROM employees WHERE salary NOT IN (SELECT salary FROM employees WHERE department_id = 30); SELECT * FROM employees WHERE job_id NOT IN ('PU_CLERK', 'SH_CLERK');

NOT IN 演算子に続くリストの中のいずれかの項目が NULL の場合は、すべての行は不明 (UNKNOWN) と評価されます (行は戻されません)。たとえば、次の文ではそれぞれの行に対して文字列 'TRUE' が戻されます。

```
SELECT employee_id FROM employees
WHERE department_id IN (10, 20);
```

```
SELECT employee_id FROM employees
WHERE department_id NOT IN (10, 20);
```

ただし、次の文では行は戻されません。

```
SELECT employee_id FROM employees
WHERE department_id NOT IN (10, NULL);
```

この例で行が戻されないのは、WHERE 句の条件が次のように評価されるためです。

```
department_id != 10 AND department_id != 20 AND department_id != null
```

NULL を比較するすべての条件は 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は、このような動作を見逃してしまう可能性があることに注意してください。

範囲条件

範囲条件は、範囲に含まれているかどうかをテストします。

range_condition::=

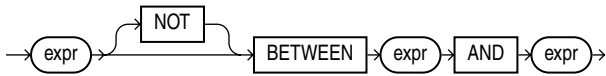


表 5-8 に、範囲条件を示します。

表 5-8 範囲条件

条件	操作	例
[NOT] BETWEEN x AND y	x 以上 y 以下の範囲である [ない] ことをテストします。	SELECT * FROM employees WHERE salary BETWEEN 2000 AND 3000;

NULL 条件

NULL 条件は、NULL かどうかをテストします。

null_condition::=

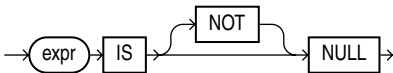


表 5-9 に、NULL 条件を示します。

表 5-9 NULL 条件

条件	操作	例
IS [NOT] NULL	NULL をテストします。NULL のテストに使用する必要がある 唯一の条件です。 参照：2-77 ページの 「NULL」を参照してくだ さい。	SELECT last_name FROM employees WHERE commission_pct IS NULL;

EXISTS 条件

EXISTS 条件は、副問合せに行が存在するかどうかをテストします。

exists_condition::=

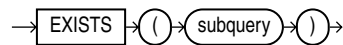


表 5-10 に、EXISTS 条件を示します。

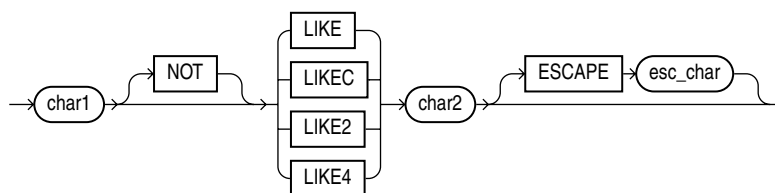
表 5-10 EXISTS 条件

条件	操作	例
EXISTS	副問合せによって行が 1 行以上 戻される場合には、TRUE と評 価されます。	SELECT department_id FROM departments d WHERE EXISTS (SELECT * FROM employees e WHERE d.department_id = e.department_id);

LIKE 条件

LIKE 条件は、パターン一致を含むかどうかをテストします。等号演算子 (=) は、ある文字値を別の文字値と一致させますが、LIKE 条件は、ある文字値の一部を別の文字値と一致させます（ある値が指定したパターンの検索を、もう一方の値に対して行います）。LIKE は、入力キャラクタ・セットによって定義された文字を使用して、文字列を算出します。LIKEC は、完全な Unicode キャラクタを使用します。LIKE2 は、UCS2 コードポイントを使用します。LIKE4 は、UCS4 コードポイントを使用します。

like_condition::=



この構文は、次の特徴があります。

- `char1` は、キャラクタ列などの文字式で、**検索値**と呼ばれます。
- `char2` は、通常はリテラルの文字式で、**パターン**と呼ばれます。
- `esc_char` は、通常はリテラルの文字式で、**エスケープ文字**と呼ばれます。

`esc_char` が指定されていない場合、デフォルトのエスケープ文字はありません。`char1`、`char2` または `esc_char` のいずれかが NULL である場合、結果は不明になります。それ以外の場合は、エスケープ文字（指定されている場合）は、長さが 1 の文字列です。

すべての文字式（`char1`、`char2` および `esc_char`）は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型です。文字式が異なる場合、Oracle はすべての文字式を `char1` のデータ型に変換します。

パターンは、特殊パターン一致文字を含むことができます。

- パーセント (%) は、任意の長さ (0 を含む) のすべての文字列を一致させます。
- アンダースコア () は、すべての単一文字を一致させます。

パーセント (%) およびアンダースコア () の文字を検索するには、それらの文字の前にエスケープ文字を指定してください。たとえば、エスケープ文字がアットマーク (@) である場合、@% を使用してパーセント (%) を、@_ を使用してアンダースコア () を検索できます。

エスケープ文字を検索する場合は、その文字を続けて入力してください。たとえば、アットマーク (@) がエスケープ文字である場合、@@ を使用してアットマーク (@) を検索できます。

パターンでは、エスケープ文字に、パーセント (%)、アンダースコア () またはエスケープ文字自体が続く必要があります。

表 5-11 に、LIKE 条件を示します。

表 5-11 LIKE 条件

条件	操作	例
x [NOT] LIKE y [ESCAPE 'z']	x がパターン y に一致しない場合は TRUE と評価されます。y 内で、文字「%」は 0 以上の NULL 以外の文字を含む文字列と一致します。「_」文字は、任意の 1 文字に一致します。パーセント (%) およびアンダースコア () を除く任意の文字を ESCAPE の後に指定できます。ワイルド・カード文字は、エスケープ文字に指定された文字が前に付いている場合はリテラルとして扱われます。	<pre>SELECT last_name FROM employees WHERE last_name LIKE '%A_B%' ESCAPE '\';</pre>

LIKE 条件を処理するために、Oracle は、パターンを 1 つまたは 2 つの文字で構成されるサブパターンに分割します。2 文字のサブパターンは、エスケープ文字で始まり、もう 1 つの文字はパーセント (%)、アンダースコア () またはエスケープ文字です。

P₁、P₂、...、P_n を、このようなサブパターンと想定します。検査値を部分文字列 S₁、S₂、...、S_n に分割でき、すべての i が 1 ～ n の場合、LIKE 条件は真です。

- P_i がアンダースコア () の場合、S_i は単一文字です。
- P_i がパーセント (%) の場合、S_i は文字列です。
- P_i がエスケープ文字で始まる 2 文字の場合、S_i は P_i の 2 番目の文字です。
- それ以外の場合は、P_i と S_i は同じです。

LIKE 条件では、値を定数ではなくパターンと比較できます。必ず LIKE キーワードの直後に、パターンを指定してください。たとえば、次の問合せを発行することによって、名前が「R」で始まるすべての従業員の給与を検索できます。

```
SELECT salary
FROM employees
WHERE last_name LIKE 'R%';
```

次の問合せは、LIKE 条件ではなく = 演算子を使用しているため、名前が 'SM%' のすべての従業員の給与が検索されます。

```
SELECT salary
  FROM employees
 WHERE last_name = 'SM%';
```

次の問合せでは、名前が 'SM%' のすべての従業員の給与が検索されます。この場合、'SM%' が LIKE キーワードの前にあるため、Oracle は、'SM%' をパターンとしてではなく、テキスト・リテラルとして解析します。

```
SELECT salary
  FROM employees
 WHERE 'SM%' LIKE last_name;
```

パターンでは、値の中の異なる文字に置き換えることができ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）に置き換えることができます。
- パターン中のパーセント記号 (%) は、値の中の 0（ゼロ）を含む任意の数の文字（マルチバイトのキャラクタ・セットでの 1 バイトとは異なる）に置き換えることができます。ただし、パターン「%」は、NULL と一致しないため注意してください。

英大文字または英小文字の区別

LIKE 条件および等号 (=) 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 ファンクションを使用すると、大文字と小文字を区別せずに一致させることができます。

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

索引列でのパターン一致

LIKE を使用して索引列をパターン検索する場合、パターンの先頭文字が「%」または「_」でなければ、Oracle は索引を利用して文のパフォーマンスを向上させることができます。この場合、Oracle はこの先頭文字によって索引をスキャンできます。パターンの先頭文字が「%」または「_」の場合、Oracle は索引をスキャンできないため、問合せのパフォーマンスは向上しません。

一般的な例

次の条件は、「Ma」で始まるすべての last_name 値について TRUE（真）となります。

```
last_name LIKE 'Ma%'
```

次の last_name 値はすべて、条件を TRUE（真）にします。

```
Mallin, Markle, Marlow, Marvins, Marvis, Matos
```

大文字と小文字は区別されるため、「MA」、「ma」および「mA」で始まる last_name 値では条件が FALSE（偽）になります。

次の条件を考えます。

```
last_name LIKE 'SMITH_'
```

この条件は、次の last_name 値について TRUE（真）となります。

```
SMITHE, SMITHY, SMITHS
```

特殊文字「_」は、last_name 値の 1 つの文字に置き換えることができるため、この条件は「SMITH」について偽となります。

ESCAPE 句の例

エスケープ文字を識別する ESCAPE 句を使用すると、パターン中に「%」または「_」を実際の文字として含めることができます。エスケープ文字がパターンの中で文字「%」または「_」の前に指定されている場合、Oracle は、この文字を特殊なパターン一致文字としてではなく、リテラル文字として解析します。

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 last_name
FROM employees
WHERE last_name LIKE '%A\_B%' ESCAPE '\';
```

ESCAPE 句は、エスケープ文字としてバックスラッシュ（\）を識別します。パターンの中でエスケープ文字はアンダースコア（_）に先行します。これによって、Oracle は、アンダースコアを特殊なパターン一致文字としてではなく、リテラルとして解析します。

パーセント (%) なしのパターンの例

パターンに文字「%」が含まれていない場合、両方のオペランドの長さが同じ場合にのみ、条件が TRUE（真）になります。この表の定義および挿入される値について考えます。

```
CREATE TABLE ducks (f CHAR(6), v VARCHAR2(6));
INSERT INTO ducks VALUES ('DUCK', 'DUCK');
SELECT '*' || f || '*' "char",
       '*' || v || '*' "varchar"
FROM ducks;
```

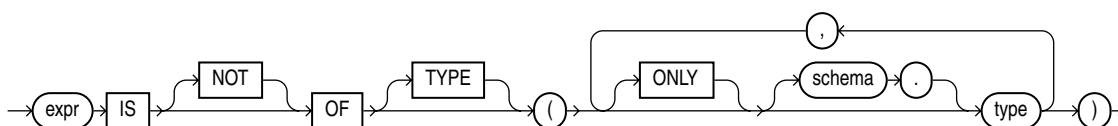
char	varchar
-----	-----
*DUCK	*DUCK*

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

IS OF type 条件

IS OF type 型条件を使用すると、固有の型情報に基づくオブジェクト・インスタンスをテストできます。

is_of_type_condition::=



type によって参照されるすべての型に対する EXECUTE 権限が必要です。また、すべての type は、同じ型ファミリーに属している必要があります。

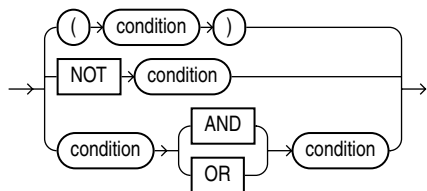
expr が NULL である場合、この条件は NULL と評価されます。expr が NULL でない場合、次に示す環境では、この条件は TRUE（NOT キーワードを指定した場合は FALSE）と評価されます。

1. expr の型が、type リストで指定された型のサブタイプであり、ONLY を指定していない場合
2. expr の型が type リストで明示的に指定されている場合

複合条件

複合条件は、異なる条件の組合せを指定します。

compound_condition::=



参照： NOT、AND および OR 条件の詳細は、5-8 ページの「[論理条件](#)」を参照してください。

ファンクション

ファンクションは、データ項目を操作し結果を戻すという点で演算子と似ています。ファンクションと演算子は引数を指定する書式が異なります。次の書式によって、ファンクションでは0（ゼロ）以上の引数を操作できます。

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

この章では、次の内容を説明します。

- [SQL ファンクション](#)
- [ユーザー定義ファンクション](#)

SQL ファンクション

SQL ファンクションは、Oracle に組み込まれており、適切な SQL 文で使用できます。SQL ファンクションと、PL/SQL で記述されたユーザー・ファンクションを混同しないでください。

SQL ファンクションが戻す値のデータ型以外のデータ型の引数で SQL ファンクションをコールすると、Oracle は SQL ファンクションを実行する前に、その引数を必要なデータ型に暗黙的に変換します。NULL を引数として SQL ファンクションをコールすると、SQL ファンクションは自動的に NULL を戻します。この規則に従わない SQL ファンクションは、CONCAT、NVL および REPLACE のみです。

SQL ファンクションの構文図では、データ型とともに引数が示されています。SQL 構文にパラメータ「function」が指定されている場合は、この項で説明するファンクションの1つに置き換えます。ファンクションは、引数のデータ型および戻り値によってグループ化されています。

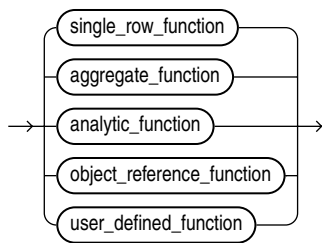
注意： LOB 列に SQL ファンクションを適用すると、Oracle は、SQL および PL/SQL の処理中にテンポラリ LOB を作成します。ご使用のアプリケーションのテンポラリ LOB を格納するために、十分な一時表領域が割り当てられていることを確認してください。

参照：

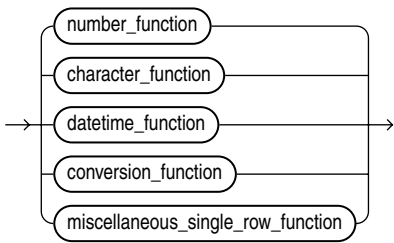
- ユーザー・ファンクションの詳細は、6-196 ページの「[ユーザー定義ファンクション](#)」を参照してください。
- Oracle Text で使用するファンクションの詳細は、『Oracle Text リファレンス』を参照してください。
- データ型の暗黙的な変換については、2-46 ページの「[データ変換](#)」を参照してください。

一般的な構文は次のとおりです。

function::=



single_row_function::=



次の項に、前述の図のユーザー定義ファンクション以外の各グループにおける組込み SQL ファンクションを示します。すべての組込み SQL ファンクションを、アルファベット順に説明します。ユーザー定義ファンクションについては、この章の最後で説明します。

単一行ファンクション

単一行ファンクションは、問合せ対象の表またはビューの各行に対して 1 つの結果行を戻します。これらのファンクションは、SELECT 構文のリスト、WHERE 句、START WITH 句、CONNECT BY 句および HAVING に指定できます。

数値ファンクション

数値ファンクションは入力として数値を受け取り、結果として数値を戻します。これらのファンクションのほとんどは、38 桁（10 進）の値を戻します。超越関数（COS、COSH、EXP、LN、LOG、SIN、SINH、SQRT、TAN および TANH）は、36 桁（10 進）の値を戻します。超越関数の ACOS、ASIN、ATAN、ATAN2 は、30 桁（10 進）の値を戻します。数値ファンクションを次に示します。

表 6-1 数値ファンクション

ABS	EXP	SIN
ACOS	FLOOR	SINH
ASIN	LN	SQRT
ATAN	LOG	TAN
ATAN2	MOD	TANH
BITAND	POWER	TRUNC（数値）
CEIL	ROUND（数値）	WIDTH_BUCKET
COS	SIGN	
COSH		

文字値を戻す文字ファンクション

文字値を戻す文字ファンクションは、入力した引数と同じデータ型の値を戻します。

- CHAR 型の値を戻すファンクションは、値の長さが 2MB に制限されます。
- VARCHAR2 型の値を戻すファンクションは、値の長さが 4MB に制限されます。
これらのファンクションでは、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・メッセージを表示せずにその結果を戻します。
- CLOB 型の値を戻すファンクションは、値の長さが 4GB に制限されます。
CLOB ファンクションの戻り値が制限を超える場合、Oracle はエラーを表示し、データを戻しません。

文字値を戻す文字ファンクションを次に示します。

表 6-2 文字値を戻す文字ファンクション

CHR	NLS_LOWER	SUBSTR
CONCAT	NLSSORT	TRANSLATE
INITCAP	NLS_UPPER	TREAT
LOWER	REPLACE	TRIM
LPAD	RPAD	UPPER
LTRIM	RTRIM	
NLS_INITCAP	SOUNDEX	

数値を戻す文字ファンクション

数値を戻す文字ファンクションの引数には、すべての文字データ型を指定できます。

数値を戻す文字ファンクションを次に示します。

表 6-3 数値を戻す文字ファンクション

ASCII	INSTR	LENGTH
-------	-------	--------

日時ファンクション

日付ファンクションは、DATE データ型の値を操作します。数値を戻す MONTHS_BETWEEN ファンクションを除いて、すべての日付ファンクションは DATE データ型の日時または期間の値を戻します。日付ファンクションを次に示します。

表 6-4 日時ファンクション

ADD_MONTHS	MONTHS_BETWEEN	SYSTIMESTAMP
CURRENT_DATE	NEW_TIME	SYSDATE
CURRENT_TIMESTAMP	NEXT_DAY	TO_DSINTERVAL
DBTIMEZONE	NUMTODSINTERVAL	TO_TIMESTAMP
EXTRACT (日時)	NUMTOYMINTERVAL	TO_TIMESTAMP_TZ
FROM_TZ	ROUND (日付)	TO_YMINTERVAL
LAST_DAY	SESSIONTIMEZONE	TRUNC (日付)
LOCALTIMESTAMP	SYS_EXTRACT_UTC	TZ_OFFSET

変換ファンクション

変換ファンクションは、あるデータ型から他のデータ型に値を変換します。一般に、ファンクション名は *datatype* TO *datatype* の書式で指定されます。最初のデータ型は入力データ型です。2 番目のデータ型は出力データ型です。SQL 変換ファンクションを次に示します。

表 6-5 変換ファンクション

ASCIISTR	RAWTONHEX	TO_NCHAR (文字)
BIN_TO_NUM	ROWIDTOCHAR	TO_NCHAR (日時)
CAST	ROWIDTONCHAR	TO_NCHAR (数値)
CHARTOROWID	TO_CHAR (文字)	TO_NCLOB
COMPOSE	TO_CHAR (日時)	TO_NUMBER
CONVERT	TO_CHAR (数値)	TO_SINGLE_BYTE
DECOMPOSE	TO_CLOB	TO_YMINTERVAL
HEXTORAW	TO_DATE	TRANSLATE ...USING
NUMTODSINTERVAL	TO_DSINTERVAL	UNISTR
NUMTOYMINTERVAL	TO_LOB	
RAWTOHEX	TO_MULTI_BYTE	

その他の単一行ファンクション

次の単一行ファンクションは、他の単一行ファンクションのカテゴリのいずれにも入りません。

表 6-6 その他の単一行ファンクション

BFILENAME	NLS_CHARSET_DECL_LEN	SYS_GUID
COALESCE	NLS_CHARSET_ID	SYS_TYPEID
DECODE	NLS_CHARSET_NAME	SYS_XMLAGG
DUMP	NULLIF	SYS_XMLGEN
EMPTY_BLOB、EMPTY_CLOB	NVL	UID
EXISTSNODE	NVL2	USER
EXTRACT (XML)	SYS_CONNECT_BY_PATH	USERENV
GREATEST	SYS_CONTEXT	VSIZE
LEAST	SYS_DBURIGEN	
	SYS_EXTRACT_UTC	

集計ファンクション

集計ファンクションは、単一行に基づく結果行を戻すのではなく、行のグループに基づく単一結果行を戻します。集計ファンクションは、SELECT 構文のリスト、ORDER BY および HAVING 句に指定できます。集計ファンクションは、通常、SELECT 文の GROUP BY 句で使用され、Oracle は問合せ対象の表またはビューの行をグループ化します。GROUP BY 句を含む問合せでは、SELECT 構文のリストの要素は、これらのいずれかを含む集計ファンクション、GROUP BY 式、定数または式にする必要があります。Oracle は、集計ファンクションを行の各グループに適用し、各グループに単一の結果行を戻します。

GROUP BY 句を指定しないと、Oracle は、SELECT 構文のリスト内の集計ファンクションを、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 句で集計ファンクションを使用して、グループを出力しないこともできます。このとき、出力は、問合せ対象の表またはビューの各行の値ではなく、集計ファンクションの結果に基づきます。

参照： 問合せおよび副問合せにおける GROUP BY 句および HAVING 句の詳細は、17-26 ページの「[GROUP BY の例](#)」および 17-20 ページの「[HAVING 句](#)」を参照してください。

単一の引数を指定する多くの集計ファンクションには、次の句があります。

- `DISTINCT` を指定すると、集計ファンクションは引数式の重複行を排除した値のみを考慮します。
- `ALL` を指定すると、集計ファンクションは重複行を含むすべての値を考慮します。

たとえば、1、1、1、3 の平均値は `DISTINCT` では 2 となり、`ALL` では 1.5 となります。どの句も指定しない場合、デフォルトで `ALL` が使用されます。

`COUNT(*)` および `GROUPING` を除くすべての集計ファンクションは、`NULL` を無視します。集計ファンクションに対する引数に `NVL` ファンクションを使用して、`NULL` をある値で置き換えることができます。`COUNT` は `NULL` を戻しません。数字または 0（ゼロ）を戻します。その他の集計ファンクションでは、データ・セットに行がない場合、または集計ファンクションに対する引数として `NULL` を持つ行のみがある場合は `NULL` を戻します。

集計ファンクションはネストできます。たとえば、次の例では、`scott` スキーマのすべての部門の最高給与の平均を計算しています。

```
SELECT AVG(MAX(salary)) FROM employees GROUP BY department_id;

AVG(MAX(SALARY))
-----
          10925
```

この計算では、`GROUP BY` 句 (`department_id`) で定義されている各グループごとの内部集計 (`MAX(salary)`) を評価し、その結果をもう一度集計しています。

集計ファンクションを次に示します。

表 6-7 集計ファンクション

AVG	GROUPING_ID	STDDEV
CORR	LAST	STDDEV_POP
COUNT	MAX	STDDEV_SAMP
COVAR_POP	MIN	SUM
COVAR_SAMP	PERCENTILE_CONT	VAR_POP
CUME_DIST	PERCENTILE_DISC	VAR_SAMP
DENSE_RANK	PERCENT_RANK	VARIANCE
FIRST	RANK	
GROUP_ID	REGR_ (線形リグレッション)	
GROUPING	ファンクション	

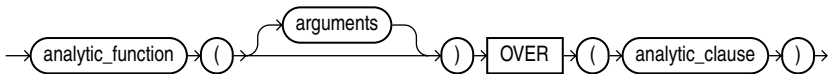
分析ファンクション

分析ファンクションは、行のグループに基づいて集計値を計算します。各グループに対して複数の行を戻す点で、集計ファンクションと異なります。行のグループを **window** といい、分析句で定義されます。各行に対して、行の「スライディング」ウィンドウが定義されます。window は、「カレント行」の計算に使用される行の範囲を決定します。window の大きさは、行の物理数値または時間などの論理間隔に基づきます。

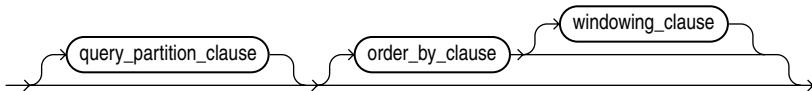
分析ファンクションは、問合せで最後に実行される演算（最後の ORDER BY 句を除く）の集合です。すべての結合およびすべての WHERE、GROUP BY および HAVING 句は、分析ファンクションが処理される前に実行されます。そのため、分析ファンクションは、SELECT 構文のリストまたは ORDER BY 句のみに指定できます。

通常、分析ファンクションは、累積集計、移動集計、センター集計およびレポート集計の実行に使用されます。

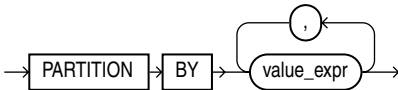
analytic_function::=



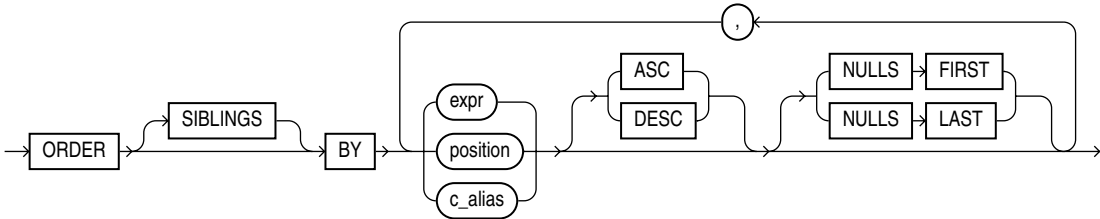
analytic_clause::=



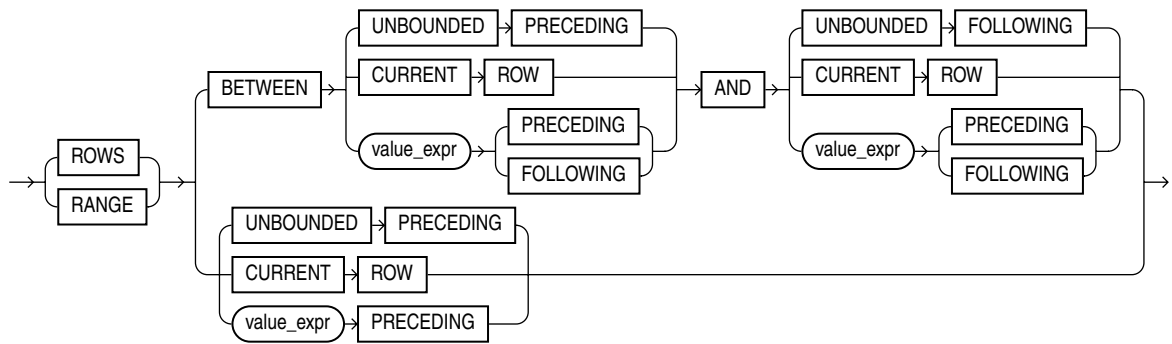
query_partition_clause::=



order_by_clause::=



windowing_clause::=



この構文のキーワードおよびパラメータを次に示します。

analytic_function

分析ファンクションの名前を指定します（6-13 ページの表 6-8 を参照）。

arguments

分析ファンクションには引数を 0 ～ 3 個指定します。

analytic_clause

OVER *analytic_clause* 句は、ファンクションが問合せ結果セットを操作することを示します。FROM、WHERE、GROUP BY および HAVING 句の後に計算されます。SELECT 構文のリストのこの句または ORDER BY 句に分析ファンクションを指定できます。分析ファンクションに基づいて、問合せの結果をフィルタするには、これらのファンクションを親問合せ内でネストした後、ネストされた副問合せの結果をフィルタします。

注意：

- *analytic_clause* のどの部分にも、分析ファンクションを指定できません。つまり、分析ファンクションをネストできません。ただし、副問合せで分析ファンクションを指定して、別の分析ファンクションを計算することはできます。
 - OVER *analytic_clause* には、組込み分析ファンクションと同様に、ユーザー定義の分析ファンクションを指定できます。12-47 ページの「CREATE FUNCTION」を参照してください。
-
-

query_partition_clause

- **PARTITION BY** 句を使用すると、1つ以上の *value_expr* に基づいて、問合せ結果セットをグループに分割できます。この句を省略すると、ファンクションは問合せ結果セットのすべての行を単一のグループとして扱います。

同じまたは異なる **PARTITION BY** キーで、同じ問合せに複数の分析ファンクションを指定できます。

注意： 問い合せているオブジェクトにパラレル属性があり、*query_partition_clause* で分析ファンクションを指定する場合は、ファンクションの計算もパラレル化されます。

- 有効な値の *value_expr* は、定数、列、非分析ファンクション、ファンクション式、またはこれらのいずれかを含む式です。

order_by_clause

order_by_clause を使用すると、パーティション内でのデータの順序付け方法を指定できます。PERCENTILE_CONT および（単一キーのみを適用する）PERCENTILE_DISC 以外の分析ファンクションでは、各キーが *value_expr* で定義され、順序付けシーケンスで修飾された複数キーのパーティションの値を順序付けできます。

各ファンクションには、複数の順序式を指定できます。これは、2番目の式が最初の式にある同一値との間の関連性を変換できるため、値をランク付けするファンクションを使用する場合に特に有効です。

注意： *order_by_clause* の結果が複数行の個々の値である場合、ファンクションは各行の同じ値を戻します。この動作の詳細は、6-142 ページの「[SUM](#)」の分析例を参照してください。

制限事項： *order_by_clause* を分析ファンクションで使用する場合、式 (*expr*) が必要です。SIBLINGS キーワードは無効です（これは、階層問合せでのみ有効です）。位置 (*position*) および列別名 (*c_alias*) は無効です。それ以外で使用する場合、この *order_by_clause* は、問合せまたは副問合せ全体を順序付ける場合に使用するものと同じです。

ASC | DESC 順序付けシーケンス（昇順または降順）を指定します。デフォルトは ASC です。

NULLS FIRST | NULLS LAST NULL 値を含む戻された行が順序の最初にくるか、最後にくるかを指定します。

NULLS LAST は昇順のデフォルトで、NULLS FIRST は降順のデフォルトです。

注意： 分析ファンクションは、常に、ファンクションの `order_by_clause` で指定された順序で行を操作します。ただし、ファンクションの `order_by_clause` は結果の順序を保証しません。最終結果の順序を保証するには、問合せの `order_by_clause` を使用してください。

参照： この句の詳細は、17-21 ページの「`SELECT`」の `order_by_clause` を参照してください。

`windowing_clause`

一部の分析ファンクションでは、`windowing_clause` を使用できます。6-13 ページの表 6-8 に、分析ファンクションを示します。`windowing_clause` を使用できる分析ファンクションには、アスタリスク (*) が付いています。

ROWS | RANGE これらのキーワードは、各行に対して、ファンクションの結果の計算に使用される「window」（行の物理集合または論理集合）を定義します。ファンクションは、window のすべての行に適用されます。window は、問合せ結果セット内またはパーティションの上から下までスライドします。

- ROWS は、物理単位（行）で window を指定します。
- RANGE は、論理オフセットとして window を指定します。

`order_by_clause` を指定しないと、この句を指定できません。

注意： 分析ファンクションが論理オフセットで戻す値は、常に決定的なものです。ただし、分析ファンクションが物理オフセットで戻す値は、順序式の結果が一意の順序にならないかぎり、非決定的な結果を生成することがあります。`order_by_clause` に複数の列を指定して、結果の順序を一意にする必要があります。

BETWEEN ...AND BETWEEN ... AND 句を使用すると、window にスタート・ポイントおよびエンド・ポイントを指定できます。最初の式（AND の前）はスタート・ポイントを定義し、2 番目の式（AND の後）はエンド・ポイントを定義します。

BETWEEN を省略してエンド・ポイントを 1 つのみ指定すると、Oracle はそれをスタート・ポイントとみなし、デフォルトでカレント行をエンド・ポイントに指定します。

UNBOUNDED PRECEDING UNBOUNDED PRECEDING を指定すると、パーティションの最初の行で、window が開始することを指定できます。これはスタート・ポイントの指定で、エンド・ポイントの指定としては使用できません。

UNBOUNDED FOLLOWING UNBOUNDED FOLLOWING を指定して、パーティションの最後の行で、window が終了することを指定できます。これはエンド・ポイントの指定で、スタート・ポイントの指定としては使用できません。

CURRENT ROW スタート・ポイントとして、window がカレント行または値（それぞれ ROW または RANGE を指定したかどうかに基づく）で開始することを指定します。この場合、`value_expr PRECEDING` をエンド・ポイントにできません。

エンド・ポイントとして、window がカレント行または値（それぞれ ROW または RANGE を明示的に指定したかどうかに基づく）で終了することを指定します。この場合、`value_expr FOLLOWING` をスタート・ポイントにできません。

`value_expr PRECEDING` または `value_expr FOLLOWING` RANGE または ROW に対して、次のことがいえます。

- `value_expr FOLLOWING` がスタート・ポイントの場合、エンド・ポイントは `value_expr FOLLOWING` である必要があります。
- `value_expr PRECEDING` がエンド・ポイントの場合、スタート・ポイントは `value_expr PRECEDING` である必要があります。

数値形式の時間間隔で定義されている論理ウィンドウを定義する場合、変換ファンクションを使用する必要があります。

参照： 数値時間から間隔への変換の詳細は、6-103 ページの「[NUMTOYMINTERVAL](#)」および 6-102 ページの「[NUMTODSINTERVAL](#)」を参照してください。

ROWS を指定した場合、次のことがいえます。

- `value_expr` は物理オフセットになります。これは定数または式であり、正数値に評価する必要があります。
- `value_expr` がスタート・ポイントの一部の場合、エンド・ポイントの前にある行に評価する必要があります。

RANGE を指定した場合、次のことがいえます。

- `value_expr` は論理オフセットになります。これは、正数値または期間リテラルに評価する定数または式である必要があります。

参照： 期間リテラルの詳細は、2-51 ページの「[リテラル](#)」を参照してください。

- `order_by_clause` には、式を 1 つのみ指定できます。
- `value_expr` が数値に対して評価を行う場合、ORDER BY `expr` は NUMBER または DATE データ型である必要があります。

- `value_expr` が間隔値に対して評価を行う場合、`ORDER BY expr` は `DATE` データ型である必要があります。

`windowing_clause` を完全に省略した場合、デフォルトで `RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW` になります。

分析ファンクションは、通常、データ・ウェアハウス環境で使用されます。表 6-8 に、分析ファンクションを示します。`windowing_clause` を含む完全な構文を使用できるファンクションには、アスタリスク (*) が付いています。

表 6-8 分析ファンクション

AVG *	LEAD	ROW_NUMBER
CORR *	MAX *	STDDEV *
COVAR_POP *	MIN *	STDDEV_POP *
COVAR_SAMP *	NTILE	STDDEV_SAMP *
COUNT *	PERCENT_RANK	SUM *
CUME_DIST	PERCENTILE_CONT	VAR_POP *
DENSE_RANK	PERCENTILE_DISC	VAR_SAMP *
FIRST	RANK	VARIANCE *
FIRST_VALUE *	RATIO_TO_REPORT	
LAG	REGR_ (線形リグレーション)	
LAST	ファンクション *	
LAST_VALUE *		

参照： これらのファンクションおよびその使用方法の詳細は、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

オブジェクト参照ファンクション

オブジェクト参照ファンクションは、指定されたオブジェクト型のオブジェクトへの参照となる REF を操作します。オブジェクト参照ファンクションを次に示します。

DEREF	REF	VALUE
MAKE_REF	REFTOHEX	

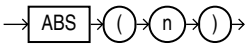
参照： REF の詳細は、『Oracle9i データベース概要』 および 『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

SQL ファンクションのリスト（アルファベット順）

ABS

構文

abs::=



用途

ABS は、 n の絶対値を戻します。

例

次の例では、-15 の絶対値を戻します。

```
SELECT ABS(-15) "Absolute" FROM DUAL;
```

```
  Absolute
-----
         15
```

ACOS

構文

acos::=



用途

ACOS は、 n のアーク・コサインを戻します。入力 は -1 ～ 1 の範囲で、出力は $0 \sim \pi$ （ラジアン）の範囲です。

例

次の例では、.3 のアーク・コサインを戻します。

```
SELECT ACOS(.3) "Arc_Cosine" FROM DUAL;
```

```
Arc_Cosine
-----
1.26610367
```

ADD_MONTHS

構文

add_months::=

→ ADD_MONTHS → (→ d → , → n →) →

用途

ADD_MONTHS は、日付 *d* に *n* か月を加えて戻します。引数 *n* には、任意の整数を指定できます。*d* が月の最終日の場合、または結果の月の日数が *d* の日付コンポーネントよりも少ない場合、戻される値は結果の月の最終日となります。それ以外の場合、結果は *d* と同じ日付コンポーネントを持ちます。

例

次の例では、サンプル表 employees の hire_date 後の月を戻します。

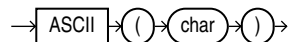
```
SELECT TO_CHAR(
    ADD_MONTHS(hire_date,1),
    'DD-MON-YYYY') "Next month"
FROM employees
WHERE last_name = 'Baer';
```

```
Next Month
-----
07-JUL-1994
```

ASCII

構文

ascii::=



用途

ASCII は、*char* の最初の文字の、データベース・キャラクタ・セットでの 10 進表記を戻します。

char のデータ型は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 です。戻り値のデータ型は NUMBER です。データベース・キャラクタ・セットが 7 ビットの ASCII の場合、このファンクションは ASCII 値を戻します。データベース・キャラクタ・セットが EBCDIC コードの場合、このファンクションは EBCDIC 値を戻します。このファンクションと一致する EBCDIC 文字ファンクションは存在しません。

注意： このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。詳細は、2-42 ページの「[データ型の比較規則](#)」を参照してください。

例

次の例では、文字「Q」を ASCII の 10 進表記で戻します。

```
SELECT ASCII('Q') FROM DUAL;
```

```
ASCII('Q')
```

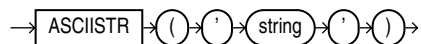
```
-----
```

```
81
```

ASCIISTR

構文

asciistr::=



用途

ASCIISTR は、任意のキャラクタ・セットでの文字列を引数として、データベース・キャラクタ・セットでの ASCII 文字列を戻します。戻り値は、SQL で表示される文字およびスラッシュ (/) のみを含みます。

参照： Unicode キャラクタ・セットおよび文字セマンティクスの詳細は、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

例

次の例では、テキスト文字列「flauwekul」を ASCII 文字列で戻します。

```
SELECT ASCIISTR('flauwekul') FROM DUAL;
```

```
ASCIISTR(
-----
flauwekul
```

ASIN

構文

asin::=



用途

ASIN は、 n のアーク・サインを戻します。入力 は $-1 \sim 1$ の範囲で、出力は $-\pi/2 \sim \pi/2$ (ラジアン) の範囲です。

例

次の例では、.3 のアーク・サインを戻します。

```
SELECT ASIN(.3) "Arc_Sine" FROM DUAL;

Arc_Sine
-----
.304692654
```

ATAN

構文

atan::=



用途

ATAN は、 n のアーク・タンジェントを戻します。入力範囲に制限はなく、出力は $-\pi/2 \sim \pi/2$ (ラジアン) の範囲です。

例

次の例では、.3 のアーク・タンジェントを戻します。

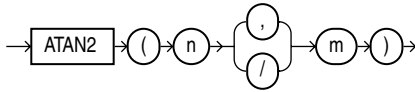
```
SELECT ATAN(.3) "Arc_Tangent" FROM DUAL;

Arc_Tangent
-----
.291456794
```


ATAN2

構文

atan2::=



用途

ATAN2 は、 n および m のアーク・タンジェントを戻します。入力範囲に制限はなく、 n および m の符号により、出力は $-\pi \sim \pi$ (ラジアン) の範囲です。ATAN2(n, m) は、ATAN2(n/m) と同じです。

例

次の例では、.3 および .2 のアーク・タンジェントを戻します。

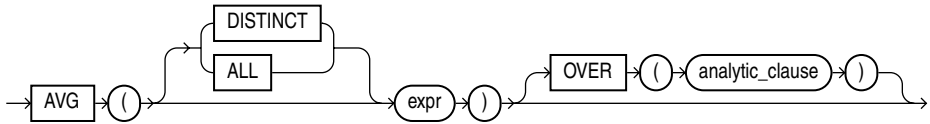
```
SELECT ATAN2(.3, .2) "Arc_Tangent2" FROM DUAL;
```

```
Arc_Tangent2
-----
.982793723
```

AVG

構文

avg::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

AVG は、*expr* の平均値を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

DISTINCT を指定する場合は、*analytic_clause* の *query_partition_clause* のみ指定できます。*order_by_clause* および *windowing_clause* は指定できません。

参照：

- 6-6 ページの「集計ファンクション」を参照してください。
- *expr* の書式の詳細は、4-2 ページの「SQL 式」を参照してください。

集計の例

次の例では、`oe.employees` 表にあるすべての従業員の平均給与を計算します。

```
SELECT AVG(salary) "Average" FROM employees;
```

```
Average
-----
6425
```

分析の例

次の例では、`employees` 表の各従業員について、ある期間内に雇用された従業員の平均給与を所属別に計算します。

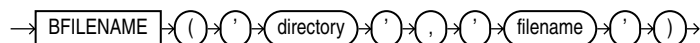
```
SELECT manager_id, last_name, hire_date, salary,
       AVG(salary) OVER (PARTITION BY manager_id ORDER BY hire_date
                          ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS c_mavg
FROM employees;
```

MANAGER_ID	LAST_NAME	HIRE_DATE	SALARY	C_MAVG
100	Kochhar	21-SEP-89	17000	17000
100	De Haan	13-JAN-93	17000	15000
100	Raphaely	07-DEC-94	11000	11966.6667
100	Kaufling	01-MAY-95	7900	10633.3333
100	Hartstein	17-FEB-96	13000	9633.33333
100	Weiss	18-JUL-96	8000	11666.6667
100	Russell	01-OCT-96	14000	11833.3333
.				
.				
.				

BFILENAME

構文

bfilename::=



用途

BFILENAME は、サーバーのファイル・システムの物理 LOB バイナリ・ファイルに対応付けられている BFILE ロケータを戻します。'directory' は、ファイルが実際に格納されているサーバーのファイル・システム上のフルパス名の別名です。また、'filename' は、サーバーのファイル・システムでのファイル名です。

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、BFILE 値を物理ファイルに関連付けておく必要があります。

参照：

- LOB の詳細は、『Oracle9i アプリケーション開発者ガイド - ラージ・オブジェクト』および『Oracle Call Interface プログラマーズ・ガイド』を参照してください。
- 12-44 ページの「[CREATE DIRECTORY](#)」を参照してください。

例

次の例では、サンプル表 pm.print_media に行を挿入します。BFILENAME を使用して、サーバーのファイル・システムのバイナリ・ファイルを識別します。

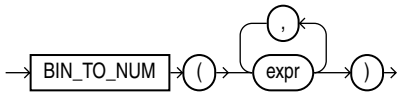
```
INSERT INTO print_media (product_id, ad_id, ad_graphic)
VALUES (3000, 31001,
       bfilename('/demo/schema/product_media', 'modem_comp_ad.gif'));
```

1 row created.

BIN_TO_NUM

構文

bin_to_num::=



用途

BIN_TO_NUM は、ビット・ベクトルを同等の数値に変換します。このファンクションの各引数は、ビット・ベクトルのビットを表します。各 *expr* は、0 または 1 で評価する必要があります。このファンクションは Oracle の数値を戻します。

BIN_TO_NUM は、グルーピング・セットを使用したマテリアライズド・ビューから、対象グループを検索するためのデータ・ウェアハウスのアプリケーションで効果的です。

参照：

- GROUPING SETS 構文の詳細は、17-19 ページの「[group_by_clause](#)」を参照してください。
- データ集計の詳細は、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

例

次の例では、バイナリの値を数値に変換します。

```
SELECT BIN_TO_NUM(1,0,1,0) FROM DUAL;
```

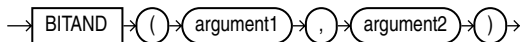
```
BIN_TO_NUM(1,0,1,0)
```

```
-----
```

BITAND

構文

bitand::=



用途

BITAND は、構成要素 *argument1*、*argument2* に対して AND 操作を計算します。
argument1 および *argument2* は、負以外の整数に変換される必要があり、整数を戻します。次に示すとおり、このファンクションは、一般的に DECODE ファンクションで使用されます。

注意： このファンクションは、戻り値のデータ型を判断しません。このため、SQL*Plus では、TO_NUMBER などのデータ型に戻すラッパーの BITAND を指定する必要があります。

例

次の例では、サンプル表 oe.orders の各 order_status をそれぞれの構成要素ごとに表示します。この例では、合計が 7 である order_status のみを指定するため、order_status が 7 よりも大きい行は排除されます。

```

SELECT order_id, customer_id,
       DECODE(BITAND(order_status, 1), 1, 'Warehouse', 'PostOffice')
         Location,
       DECODE(BITAND(order_status, 2), 2, 'Ground', 'Air') Method,
       DECODE(BITAND(order_status, 4), 4, 'Insured', 'Certified') Receipt
FROM orders
WHERE order_status < 8;
  
```

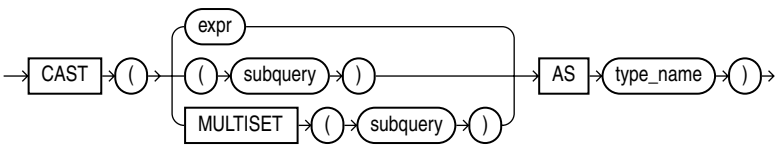
ORDER_ID	CUSTOMER_ID	LOCATION	MET	RECEIPT
2458	101	Postoffice	Air	Certified
2397	102	Warehouse	Air	Certified
2454	103	Warehouse	Air	Certified
2354	104	Postoffice	Air	Certified
2358	105	Postoffice	G	Certified
2381	106	Warehouse	G	Certified
2440	107	Warehouse	G	Certified

2357	108 Warehouse	Air Insured
2394	109 Warehouse	Air Insured
2435	144 Postoffice	G Insured
2455	145 Warehouse	G Insured
.		
.		
.		

CAST

構文

cast::=



用途

CAST ファンクションは、ある組込みデータ型またはコレクション型値を、別の組込みデータ型またはコレクション型値に変換します。

CAST によって、ある組込みデータ型またはコレクション型値を、別の組込みデータ型またはコレクション型値に変換できます。名前のないオペランド（日付や副問合せの結果セットなど）または名前付きのコレクション（VARRAY またはネストした表など）を型互換の名前付きコレクションにキャストできます。type_name は、組込みデータ型またはコレクション型の名前である必要があり、オペランドは、組込みデータ型であるか、またはその値がコレクション値である必要があり。

オペランドでは、expr は組込みデータ型またはコレクション型のいずれかで、subquery がコレクション型または組込み型の単一値を戻す必要があります。MULTISET は、副問合せの結果セットをとり、コレクション値を戻すように Oracle に知らせます。表 6-9 に、どの組込みデータ型が、どの組込みデータ型にキャストできるかを示します（CAST は、LONG 型、LONG RAW 型、すべての LOB データ型または Oracle が提供する型をサポートしていません）。

表 6-9 組み込みデータ型のキャスト

キャスト前→ キャスト後↓	CHAR、 VARCHAR2	NUMBER	DATETIME / INTERVAL ^b	RAW	ROWID、 UROWID	NCHAR、 NVARCHAR2
CHAR、 VARCHAR2	X	X	X	X	X	
NUMBER	X	X				
DATE、 TIMESTAMP、 INTERVAL	X		X			
RAW	X			X		
ROWID、 UROWID	X				X ^a	
NCHAR、 NVARCHAR2		X	X	X	X	X

^a UROWID が索引構成表の ROWID の値を含んでいる場合、UROWID を ROWID にキャストすることはできません。

^b Datetime/Interval には、DATE、TIMESTAMP、TIMESTAMP WITH TIMEZONE、INTERVAL DAY TO SECOND および INTERVAL YEAR TO MONTH が含まれます。

名前付きコレクション型を別の名前付きコレクション型にキャストするには、両方のコレクションの要素が同じ型である必要があります。

副問合せの結果セットが複数行に評価される可能性がある場合は、MULTISET キーワードを指定する必要があります。副問合せの結果である行は、それらの行がキャストされたコレクション値の要素を形成します。MULTISET キーワードを省略すると、副問合せはスカラー副問合せとして処理されます。

組み込みデータ型の例

次の例では、CAST ファンクションをスカラー・データ型とともに使用します。

```
SELECT CAST('22-OCT-1997' AS DATE) FROM dual;
```

```
SELECT product_id,
       CAST(ad_sourcetext AS VARCHAR2(30))
FROM print_media;
```

コレクションの例

次の CAST の例では、次のユーザー定義の型および表を使用します。

```
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 depts (dno NUMBER, addresses address_array_t);
```

この例では、副問合せをキャストします。

```
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;
```

CAST では、VARRAY 型の列をネストした表に変換します。

```
SELECT CAST(d.addresses AS address_book_t)
FROM depts d
WHERE d.dno = 111;
```

次の例では、ORDER BY 句で MULTISET 式をキャストします。

```
CREATE TABLE projects (empid NUMBER, projname VARCHAR2(10));
CREATE TABLE emps (empid NUMBER, ename VARCHAR2(10));
CREATE TYPE projname_table_type AS TABLE OF VARCHAR2(10);
/
```

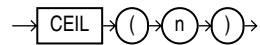
前述のスキーマでの MULTISET 式の例を次に示します。

```
SELECT e.ename, CAST(MULTISET(SELECT p.projname
    FROM projects p
    WHERE p.empid=e.empid
    ORDER BY p.projname)
    AS projname_table_type)
FROM emps e;
```


CEIL

構文

ceil::=



用途

CEIL は、 n 以上の最も小さい整数を返します。

例

次の例では、15.7 以上である最小の整数を返します。

```
SELECT CEIL(15.7) "Ceiling" FROM DUAL;
```

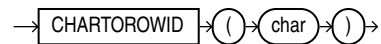
```

Ceiling
-----
      16
  
```

CHARTOROWID

構文

chartorowid::=



用途

CHARTOROWID は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型の値を ROWID データ型に変換します。

注意： この関数は、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。詳細は、2-42 ページの「[データ型の比較規則](#)」を参照してください。

例

次の例では、文字表記の ROWID を ROWID に変換します。（このファンクションは、異なるデータベースの異なる ROWID を戻します。）

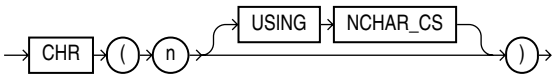
```
SELECT last_name FROM employees
      WHERE ROWID = CHARTOROWID('AAAFYmAAFAAAAFEAP');
```

```
LAST_NAME
-----
Greene
```

CHR

構文

chr::=



用途

CHR は、データベース・キャラクタ・セットまたは各国語キャラクタ・セットの中の *n* に等しい 2 進数を持つ文字を戻します。

USING NCHAR_CS が指定されていない場合、このファンクションは、データベース・キャラクタ・セットの中の *n* に等しい 2 進数を持つ文字を VARCHAR2 値として戻します。

USING NCHAR_CS が指定されている場合、このファンクションは、各国語キャラクタ・セットの中の *n* に等しい 2 進数を持つ文字を NVARCHAR2 値として戻します。

注意：（オプションの USING NCHAR_CS 句の有無にかかわらず）CHR ファンクションを使用すると、ASCII および EBCDIC ベースのマシン・アーキテクチャ間で移植不可能なコードが戻されます。

参照： 6-91 ページの「NCHR」を参照してください。

例

次の例は、データベース・キャラクタ・セットが WE8ISO8859P1 と定義されている ASCII ベースのマシンで実行されています。

```
SELECT CHR(67) || CHR(65) || CHR(84) "Dog" FROM DUAL;
```

```
Dog
---
CAT
```

キャラクタ・セットが WE8EBCDIC1047 の EBCDIC ベースのマシンでも同じ結果を返すには、前述の例を次のように修正する必要があります。

```
SELECT CHR(195) || CHR(193) || CHR(227) "Dog"
       FROM DUAL;
```

```
Dog
---
CAT
```

次の例では、UTF8 キャラクタ・セットを使用します。

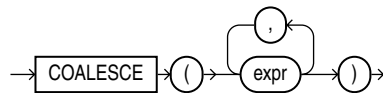
```
SELECT CHR (50052 USING NCHAR_CS) FROM DUAL;
```

```
CH
--
Ä
```

COALESCE

構文

coalesce::=



用途

COALESCE ファンクションは、式のリストの最初の NULL でない *expr* を返します。1 つ以上の *expr* が、リテラル NULL 以外である必要があります。すべての *expr* が NULL と評価された場合、このファンクションは NULL を返します。

このファンクションは NVL ファンクションを一般化したファンクションです。

COALESCE は、様々な CASE 式と同様に使用できます。次に例を示します。

```
COALESCE (expr1, expr2)
```

これは、次の CASE 式と同じです。

```
CASE WHEN expr1 IS NOT NULL THEN expr1 ELSE expr2 END
```

次の例も同様です。

```
COALESCE (expr1, expr2, ..., exprn), for n>=3
```

これは、次の CASE 式と同じです。

```
CASE WHEN expr1 IS NOT NULL THEN expr1
      ELSE COALESCE (expr2, ..., exprn) END
```

参照： 6-104 ページの「[NVL](#)」および 4-6 ページの「[CASE 式](#)」を参照してください。

例

次の例では、サンプル表 `oe.product_information` を使用して、製品の「クリアランス・セール」を企画します。製品の表示価格から 10% 値引きします。表示価格がない場合、セールの価格は最小価格になり、最小価格がない場合、セール価格は「5」です。

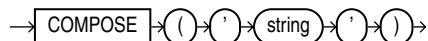
```
SELECT product_id, list_price, min_price,
       COALESCE(0.9*list_price, min_price, 5) "Sale"
FROM   product_information
WHERE  supplier_id = 102050;
```

PRODUCT_ID	LIST_PRICE	MIN_PRICE	Sale
2382	850	731	765
3355			5
1770		73	73
2378	305	247	274.5
1769	48		43.2

COMPOSE

構文

compose::=



用途

COMPOSE は、任意のデータ型の文字列を引数として、入力と同じキャラクタ・セットで定められた書式の Unicode の文字列を戻します。*string* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。たとえば、**o** ウムラウト・コードポイントは、ウムラウト・コードポイントによって修飾された「o」コードポイントとして戻されます。

参照： Unicode キャラクタ・セットおよび文字セマンティクスの詳細は、『Oracle9i データベース概要』を参照してください。

例

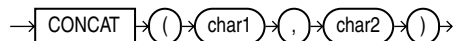
```
SELECT COMPOSE ( 'o' || UNISTR('\0308') ) FROM DUAL;
```

```
CO
--
ö
```

CONCAT

構文

concat::=



用途

CONCAT は、*char2* に連結されている *char1* を戻します。*char1* および *char2* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻される文字列は、VARCHAR2 データ型であり、*char1* と同じキャラクタ・セットです。

このファンクションは、連結演算子 (||) と同等です。

注意： NCHAR 型が CHAR 型に連結される場合、戻される型は、NCHAR です。これは、CHAR から NCHAR への変換が可逆式であるためです。たとえば、CLOB を NCHAR 文字列に連結すると、戻される型は NCLOB です。

参照： CONCAT 演算子の詳細は、3-4 ページの「[連結演算子](#)」を参照してください。

例

次の例では、ネストを使用して 3 つの文字列を連結します。

```

SELECT CONCAT(CONCAT(last_name, ''s job category is '),
              job_id) "Job"
FROM employees
WHERE employee_id = 152;
```

```

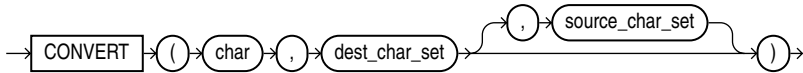
Job
-----
Hall's job category is SA_REP

```

CONVERT

構文

convert::=



用途

CONVERT は、文字列を、あるキャラクタ・セットから別のキャラクタ・セットに変換します。戻り値のデータ型は VARCHAR2 です。

- 引数 *char* は変換する値です。*char* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。
- 引数 *dest_char_set* は *char* が変換されるキャラクタ・セットの名前です。
- 引数 *source_char_set* は、*char* をデータベースに格納しているキャラクタ・セットの名前です。デフォルト値はデータベース・キャラクタ・セットです。

変換先キャラクタ・セットと変換元キャラクタ・セットの引数として、リテラルまたはキャラクタ・セットの名前を含んでいる列を指定できます。

完全に文字を変換するには、変換先キャラクタ・セットが変換元キャラクタ・セットで定義されているすべての文字を表現できる必要があります。文字が変換先キャラクタ・セットに存在しないと、置換文字が使用されます。置換文字は、キャラクタ・セット定義の一部として定義できます。

例

次の例では、Latin-1 文字列を ASCII に変換するキャラクタ・セットの変換を示します。これは、同じ文字列を WE8ISO8859P1 データベースから US7ASCII データベースへインポートした場合と同じ結果が得られます。

```
SELECT CONVERT('Ä Ê Í Õ Ø A B C D E ', 'US7ASCII', 'WE8ISO8859P1')
FROM DUAL;
```

```
CONVERT('ÄÊÍÕØABCDE'
-----
A E I ? ? A B C D E ?
```

一般的なキャラクタ・セットを次に示します。

- US7ASCII: US7 ビット ASCII キャラクタ・セット
- WE8DEC: 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

CORR

構文

corr::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

CORR は、数値の組の集合に対する相関係数を返します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

expr1 および *expr2* は数値式です。Oracle は、*expr1* または *expr2* が NULL である組を排除した後、このファンクションを (*expr1*, *expr2*) の集合に適用します。その後、Oracle は次の計算を行います。

$$\text{COVAR_POP}(\text{expr1}, \text{expr2}) / (\text{STDDEV_POP}(\text{expr1}) * \text{STDDEV_POP}(\text{expr2}))$$

このファンクションは、NUMBER 型の値を返します。ファンクションが空の集合に適用されると、NULL を返します。

参照：

- 6-6 ページの「[集計ファンクション](#)」を参照してください。
- *expr* の書式の詳細は、4-2 ページの「[SQL 式](#)」を参照してください。

集計の例

次の例では、サンプル・ビュー `oe.products` の重さクラスごとの製品の表示価格と最小価格の相関係数を計算します。

```
SELECT weight_class, CORR(list_price, min_price)
FROM products
GROUP BY weight_class;
```

```
WEIGHT_CLASS CORR(LIST_PRICE,MIN_PRICE)
-----
1              .99914795
2              .999022941
3              .998484472
4              .999359909
5              .999536087
```

分析の例

次の例では、1998 年度の `sh.sales` および `sh.times` サンプル表から、1ヶ月の売上利益および1ヶ月の売上個数の相関累積係数を計算します。

```
SELECT t.calendar_month_number,
CORR (SUM(s.amount_sold), SUM(s.quantity_sold))
OVER (ORDER BY t.calendar_month_number) as CUM_CORR
FROM sales s, times t
WHERE s.time_id = t.time_id AND calendar_year = 1998
GROUP BY t.calendar_month_number
ORDER BY t.calendar_month_number;
```

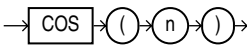
```
CALENDAR_MONTH_NUMBER  CUM_CORR
-----
1
2              1
3 .994309382
4 .852040875
5 .846652204
6 .871250628
7 .910029803
8 .917556399
9 .920154356
10 .86720251
11 .844864765
12 .903542662
```

相関ファンクションには、操作する行が1つ以上必要です。そのため、前述の例にある最初の行には、計算する値はありません。

COS

構文

cos::=



用途

COS は、*n*（ラジアンで表された角度）のコサインを戻します。

例

次の例では、180 度のコサインを戻します。

```
SELECT COS(180 * 3.14159265359/180)
       "Cosine of 180 degrees" FROM DUAL;

Cosine of 180 degrees
-----
                    -1
```

COSH

構文

cosh::=



用途

COSH は、*n* の双曲線コサインを戻します。

例

次の例では、0 の双曲線コサインを戻します。

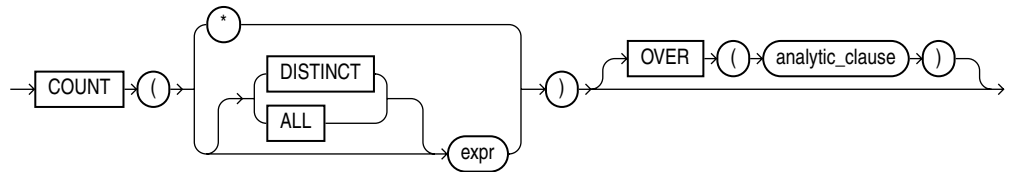
```
SELECT COSH(0) "Hyperbolic cosine of 0" FROM DUAL;

Hyperbolic cosine of 0
-----
                    1
```

COUNT

構文

count::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

COUNT は、問合せ内の行数を返します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

DISTINCT を指定する場合は、*analytic_clause* の *query_partition_clause* のみ指定できます。*order_by_clause* および *windowing_clause* は指定できません。

expr を指定すると、COUNT は *expr* が NULL でない行数を返します。*expr* のすべての行を数えるか、または異なる値のみを数えることができます。

アスタリスク (*) を指定すると、このファンクションは重複値および NULL 値を含むすべての行を返します。COUNT は NULL を返しません。

参照：

- 6-6 ページの「[集計ファンクション](#)」を参照してください。
- *expr* の書式の詳細は、4-2 ページの「[SQL 式](#)」を参照してください。

集計の例

次の例では、COUNT を集計ファンクションとして使用します。

```
SELECT COUNT(*) "Total" FROM employees;
```

```

      Total
-----
      107

```

```
SELECT COUNT(*) "Allstars" FROM employees
```

```
WHERE commission_pct > 0;

Allstars
-----
      35

SELECT COUNT(commission_pct) "Count" FROM employees;

Count
-----
      35

SELECT COUNT(DISTINCT manager_id) "Managers" FROM employees;

Managers
-----
      18
```

分析の例

次の例では、employees 表の各従業員について、その従業員の給与より 50 ドル少ない金額から 150 ドル多い金額の範囲の給与を得ている従業員の数を計算します。

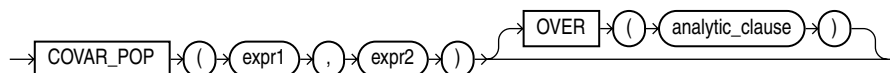
```
SELECT last_name, salary,
       COUNT(*) OVER (ORDER BY salary RANGE BETWEEN 50 PRECEDING
                       AND 150 FOLLOWING) AS mov_count FROM employees;
```

LAST_NAME	SALARY	MOV_COUNT
-----	-----	-----
Olson	2100	3
Markle	2200	2
Philtanker	2200	2
Landry	2400	8
Gee	2400	8
Colmenares	2500	10
Patel	2500	10
.		
.		
.		

COVAR_POP

構文

covar_pop::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

COVAR_POP は、数値の組の集合に対する母集団共分散を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

expr1 および *expr2* は数値式です。Oracle は、*expr1* または *expr2* が NULL であるすべての組を排除した後、このファンクションを (*expr1*, *expr2*) の集合に適用します。その後、Oracle は次の計算を行います。

$$(\text{SUM}(\text{expr1} * \text{expr2}) - \text{SUM}(\text{expr2}) * \text{SUM}(\text{expr1}) / n) / n$$

ここで、*n* は (*expr1*, *expr2*) の組の数値です（ただし、*expr1* および *expr2* は両方とも NULL ではありません）。

ファンクションは、NUMBER 型の値を戻します。ファンクションが空の集合に適用されると、NULL を戻します。

参照：

- 6-6 ページの「[集計ファンクション](#)」を参照してください。
- *expr* の書式の詳細は、4-2 ページの「[SQL 式](#)」を参照してください。

集計の例

次の例では、サンプル表 sh.sales から各年度における売上利益高および売上個数の母集団共分散を計算します。

```

SELECT t.calendar_month_number,
       COVAR_POP(s.amount_sold, s.quantity_sold) AS covar_pop,
       COVAR_SAMP(s.amount_sold, s.quantity_sold) AS covar_samp
FROM sales s, times t
WHERE s.time_id = t.time_id
AND t.calendar_year = 1998
GROUP BY t.calendar_month_number;

```

CALENDAR_MONTH_NUMBER	COVAR_POP	COVAR_SAMP
1	5437.68586	5437.88704
2	5923.72544	5923.99139
3	6040.11777	6040.38623
4	5946.67897	5946.92754
5	5986.22483	5986.4463
6	5726.79371	5727.05703
7	5491.65269	5491.9239
8	5672.40362	5672.66882
9	5741.53626	5741.80025
10	5050.5683	5050.78195
11	5256.50553	5256.69145
12	5411.2053	5411.37709

分析の例

次の例では、デモ・スキーマ oe の製品の表示価格および最小価格の累積標本共分散を計算します。

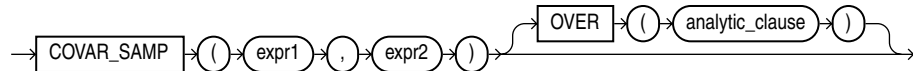
```
SELECT product_id, supplier_id,
       COVAR_POP(list_price, min_price)
         OVER (ORDER BY product_id, supplier_id)
           AS CUM_COVP,
       COVAR_SAMP(list_price, min_price)
         OVER (ORDER BY product_id, supplier_id)
           AS CUM_COVS
FROM product_information p
WHERE category_id = 29
ORDER BY product_id, supplier_id;
```

PRODUCT_ID	SUPPLIER_ID	CUM_COVP	CUM_COVS
1774	103088	0	
1775	103087	1473.25	2946.5
1794	103096	1702.77778	2554.16667
1825	103093	1926.25	2568.33333
2004	103086	1591.4	1989.25
2005	103086	1512.5	1815
2416	103088	1475.97959	1721.97619
.			
.			
.			

COVAR_SAMP

構文

covar_samp::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

COVAR_SAMP は、数値の組の集合の標本共分散を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

expr1 および *expr2* は数値式です。Oracle は、*expr1* または *expr2* が NULL であるすべての組を排除した後、このファンクションを (*expr1*, *expr2*) の集合に適用します。その後、Oracle は次の計算を行います。

$$(\text{SUM}(\text{expr1} * \text{expr2}) - \text{SUM}(\text{expr1}) * \text{SUM}(\text{expr2}) / n) / (n-1)$$

ここで、*n* は (*expr1*, *expr2*) の組の数値です（ただし、*expr1* および *expr2* は両方とも NULL ではありません）。

このファンクションは、NUMBER 型の値を戻します。ファンクションが空の集合に適用されると、NULL を戻します。

参照：

- 6-6 ページの「[集計ファンクション](#)」を参照してください。
- *expr* の書式の詳細は、4-2 ページの「[SQL 式](#)」を参照してください。

集計の例

次の例では、サンプル表 `sh.sales` から各年度における売上利益高および売上個数の母集団共分散を計算します。

```

SELECT t.calendar_month_number,
       COVAR_POP(s.amount_sold, s.quantity_sold) AS covar_pop,
       COVAR_SAMP(s.amount_sold, s.quantity_sold) AS covar_samp
FROM sales s, times t
WHERE s.time_id = t.time_id
AND t.calendar_year = 1998

```

```
GROUP BY t.calendar_month_number;

CALENDAR_MONTH_NUMBER  COVAR_POP  COVAR_SAMP
-----
1  5437.68586  5437.88704
2  5923.72544  5923.99139
3  6040.11777  6040.38623
4  5946.67897  5946.92754
5  5986.22483  5986.4463
6  5726.79371  5727.05703
7  5491.65269  5491.9239
8  5672.40362  5672.66882
9  5741.53626  5741.80025
10 5050.5683  5050.78195
11 5256.50553  5256.69145
12 5411.2053  5411.37709
```

分析の例

次の例では、デモ・スキーマ oe の製品の表示価格および最小価格の累積標本共分散を計算します。

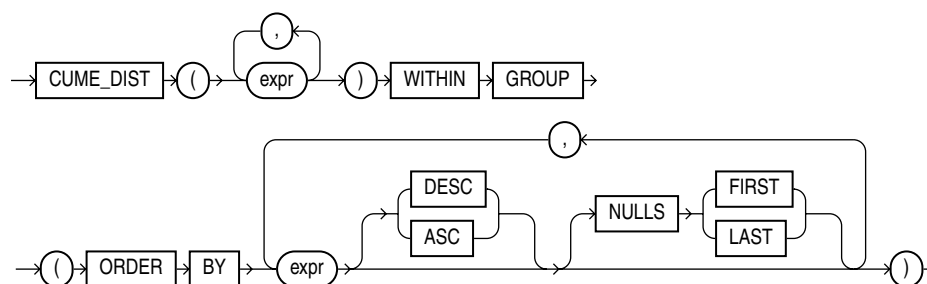
```
SELECT product_id, supplier_id,
       COVAR_POP(list_price, min_price)
         OVER (ORDER BY product_id, supplier_id)
          AS CUM_COVP,
       COVAR_SAMP(list_price, min_price)
         OVER (ORDER BY product_id, supplier_id)
          AS CUM_COVS
FROM   product_information p
WHERE  category_id = 29
ORDER BY product_id, supplier_id;
```

```
PRODUCT_ID SUPPLIER_ID  CUM_COVP  CUM_COVS
-----
1774      103088          0
1775      103087    1473.25    2946.5
1794      103096  1702.77778  2554.16667
1825      103093    1926.25  2568.33333
2004      103086    1591.4    1989.25
2005      103086    1512.5      1815
2416      103088  1475.97959  1721.97619
.
.
.
```


CUME_DIST

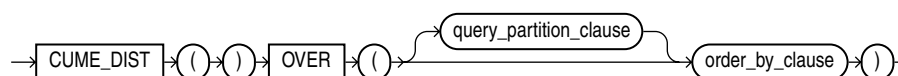
集計の構文

cume_dist_aggregate::=



分析の構文

cume_dist_analytic::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

CUME_DIST は、値のグループにある値の累積分布値を計算します。CUME_DIST が戻す値の範囲は、0 より大きく 1 以下です。連結値は、常に同じ累積分布値に対して評価を行います。

- 集計ファンクションとしての CUME_DIST は、ファンクションの引数および対応するソート指定によって識別される不確定な行 R に対して、集計グループ内の行の中での行 R の関連する位置を計算します。Oracle は、不確定な行 R を集計される行のグループに挿入するように計算します。このファンクションの引数は、各集計グループ内の 1 つの不確定行を識別します。このため、すべての引数は、集計グループ内で定数式と評価される必要があります。定数引数式および集計の ORDER BY 句の式の位置は、一致します。このため、引数の数は同じであり、その型は互換性がある必要があります。
- 分析ファンクションとしての CUME_DIST は、値のグループにある特定の値の相対位置を計算します。行 R について、昇順で順序付けられているとします。R の CUME_DIST は、R の値以下の値の行を、評価される行の数（問合せ結果セットまたはパーティション）で割った数です。

集計の例

次の例では、サンプル表 `oe.employees` の従業員の中から、給与が 15,500 ドルであり、歩合が 5% の不確定な従業員の累積分布を計算します。

```
SELECT CUME_DIST(15500, .05) WITHIN GROUP
      (ORDER BY salary, commission_pct) "Cume-Dist of 15500"
FROM employees;
```

```
Cume-Dist of 15500
-----
.972477064
```

分析の例

次の例では、購入地域の各従業員の給与のパーセンタイルを計算します。たとえば、事務員の 40% が、Himuro の給与以下の給与を得ていることがわかります。

```
SELECT job_id, last_name, salary, CUME_DIST()
      OVER (PARTITION BY job_id ORDER BY salary) AS cume_dist
FROM employees
WHERE job_id LIKE 'PU%';
```

JOB_ID	LAST_NAME	SALARY	CUME_DIST
PU_CLERK	Colmenares	2500	.2
PU_CLERK	Himuro	2600	.4
PU_CLERK	Tobias	2800	.6
PU_CLERK	Baida	2900	.8
PU_CLERK	Khoo	3100	1
PU_MAN	Raphaely	11000	1

CURRENT_DATE

構文

`current_date::=`



用途

`CURRENT_DATE` は、セッション・タイム・ゾーンの現在の日付を `DATE` データ型のグレゴリオ暦の値で戻します。

例

次の例では、CURRENT_DATE がセッション・タイム・ゾーンによって異なることを示します。

```
ALTER SESSION SET TIME_ZONE = '-5:0';
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;
```

```
SESSIONTIMEZONE CURRENT_DATE
-----
-05:00          04-APR-2000 13:14:03
```

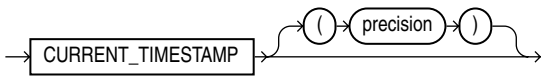
```
ALTER SESSION SET TIME_ZONE = '-8:0';
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;
```

```
SESSIONTIMEZONE CURRENT_DATE
-----
-08:00          04-APR-2000 10:14:33
```

CURRENT_TIMESTAMP

構文

current_timestamp::=



用途

CURRENT_TIMESTAMP は、セッション・タイム・ゾーンの現在の日付および時刻を TIMESTAMP WITH TIME ZONE データ型の値で戻します。タイム・ゾーン置換値は、SQL セッションの現在のローカル時刻を反映します。精度を省略すると、デフォルトは 6 です。CURRENT_TIMESTAMP は、TIMESTAMP WITH TIME ZONE 値を戻します。これに対して、LOCALTIMESTAMP は、TIMESTAMP 値を戻します。

オプションの引数では、precision は、戻される時刻の値の小数部の精度を指定します。

例

次の例では、CURRENT_TIMESTAMP がセッション・タイム・ゾーンによって異なることを示します。

```
ALTER SESSION SET TIME_ZONE = '-5:0';
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
SELECT SESSIONTIMEZONE, CURRENT_TIMESTAMP FROM DUAL;
```

SESSIONTIMEZONE	CURRENT_TIMESTAMP
-----	-----
-05:00	04-APR-00 01.17.56.917550 PM -05:00

```
ALTER SESSION SET TIME_ZONE = '-8:0';
SELECT SESSIONTIMEZONE, CURRENT_TIMESTAMP FROM DUAL;
```

SESSIONTIMEZONE	CURRENT_TIMESTAMP
-----	-----
-08:00	04-APR-00 10.18.21.366065 AM -08:00

DBTIMEZONE

構文

dbtimezone::=



用途

DBTIMEZONE ファンクションは、データベース・タイム・ゾーンの値を戻します。戻り型は、タイム・ゾーン・オフセット ('[+|-]TZH:TZM' 書式の文字型) またはタイム・ゾーン地域名です。これは、最近の CREATE DATABASE または ALTER DATABASE 文でユーザーが指定したデータベース・タイム・ゾーンの値によって異なります。

例

次の例では、データベース・タイム・ゾーンが UTC タイム・ゾーンに設定されていると想定します。

```
SELECT DBTIMEZONE FROM DUAL;
```

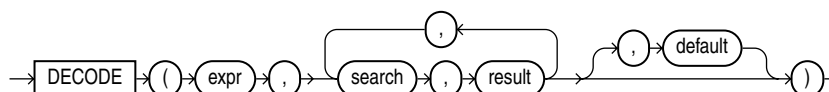
DBTIME

-08:00

DECODE

構文

`decode::=`



用途

DECODE ファンクションは、*expr* と各 *search* の値を 1 つずつ比較します。*expr* が *search* と等しい場合、Oracle は対応する *result* を戻します。一致しない場合は *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 および *search* が文字データを含む場合、Oracle は非空白埋め比較セマンティクスで比較します。*expr*、*search* および *result* 項目は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型のいずれかになります。戻される文字列は、VARCHAR2 データ型で、最初の *result* パラメータと同じキャラクタ・セットの文字列です。

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* と等しい *search* が見つかったら、Oracle はその後の *search* を評価しません。

比較する前に、Oracle は *expr* と各 *search* 値を、最初の *search* 値のデータ型に自動的に変換します。Oracle は、戻り値を最初の *result* と同じデータ型に自動的に変換します。最初の *result* のデータ型が CHAR の場合、または最初の *result* が NULL の場合、Oracle は戻り値を VARCHAR2 データ型の値に変換します。

DECODE ファンクションでは、Oracle は 2 つの NULL を同等とみなします。*expr* が NULL の場合、Oracle は最初の *search* 値の *result* も NULL として戻します。

DECODE ファンクションのコンポーネントの最大数は、*expr*、*search*、*result*、*default* を含めて 255 です。

参照：

- 比較セマンティクスについては、2-42 ページの「[データ型の比較規則](#)」を参照してください。
- データ型変換については、2-46 ページの「[データ変換](#)」を参照してください。
- 暗黙的な変換のデメリットについては、2-46 ページの「[暗黙的なデータ変換と明示的なデータ変換](#)」を参照してください。

例

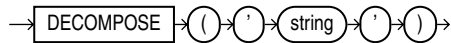
この例では、warehouse_id の値をデコードします。warehouse_id が 1 である場合、ファンクションは 'Southlake' を戻します。warehouse_id が 2 である場合、ファンクションは 'San Francisco' を戻します。warehouse_id が 1、2、3 または 4 以外である場合、ファンクションは 'Non-domestic' を戻します。

```
SELECT product_id,
       DECODE (warehouse_id, 1, 'Southlake',
              2, 'San Francisco',
              3, 'New Jersey',
              4, 'Seattle',
              'Non-domestic')
       quantity_on_hand FROM inventories;
```

DECOMPOSE

構文

decompose::=



用途

DECOMPOSE は、任意のデータ型の文字列を引数として、入力と同じキャラクタ・セットで標準的に分解された Unicode の文字列を戻します。string は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。たとえば、o ウムラウト・コードポイントは、ウムラウト・コードポイントが続く「o」コードポイントとして戻されます。

参照： Unicode キャラクタ・セットおよび文字セマンティクスの詳細は、『Oracle9i データベース概要』を参照してください。

例

次の例では、文字列「Châteaux」をその要素のコードポイントに分解します。

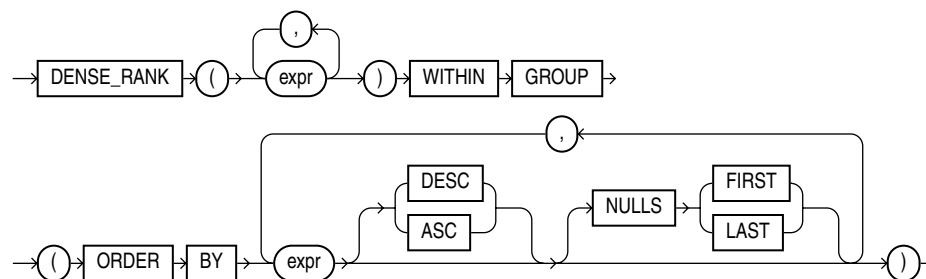
```
SELECT DECOMPOSE ('Châteaux') FROM DUAL;

DECOMPOSE
-----
Cha^teaux
```

DENSE_RANK

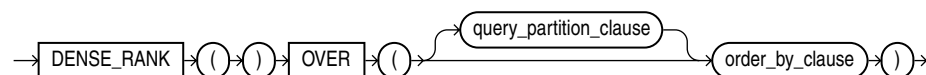
集計の構文

dense_rank_aggregate::=



分析の構文

dense_rank_analytic::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析関数](#)」を参照してください。

用途

DENSE_RANK ファンクションは、順序付けされた行のグループの行のランクを計算します。ランクは 1 から始まる連続した整数です。ランクの最大値は、問合せが戻す一意の数値です。ランクの値は、連続した整数です。ランク付け基準と同じ値を持つ行は、同じランクになります。

- 集計ファンクションとして使用する DENSE_RANK は、ファンクションの引数によって識別される不確定な行の稠密ランクを、与えられたソート指定で計算します。すべてのファンクションの引数は、各グループの単一行を識別するため、各集計グループ内で定数式に評価される必要があります。定数引数式および集計の *order_by_clause* の式の位置は、一致します。このため、引数の数は同じであり、型は互換性がある必要があります。
- 分析ファンクションとして使用する DENSE_RANK は、他の行について、問合せで戻される各行のランクを計算します。この計算は、*order_by_clause* にある *value_exprs* の値に基づいて行われます。

集計の例

次の例では、サンプル表 oe.employees から、給与が 15,500 ドルであり、歩合が 5% の仮想の従業員のランクを計算します。

```
SELECT DENSE_RANK(15500, .05) WITHIN GROUP
      (ORDER BY salary DESC, commission_pct) "Dense Rank"
FROM employees;
```

Dense Rank

3

分析の例

次の文は、HUMAN RESOURCES または PURCHASING 部門で働くすべての従業員の部門名、従業員名および給与を選択します。その後、この 2 つの部門それぞれについて、一意の各給与に対するランクを計算します。同じ給与は同じランクになります。この例と、6-112 ページの「[RANK](#)」の例を比較してください。

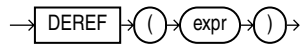
```
SELECT d.department_name, e.last_name, e.salary, DENSE_RANK()
      OVER (PARTITION BY e.department_id ORDER BY e.salary) as drank
FROM employees e, departments d
WHERE e.department_id = d.department_id
AND d.department_id IN ('30', '40');
```

DEPARTMENT_NAME	LAST_NAME	SALARY	DRANK
-----	-----	-----	-----
Purchasing	Colmenares	2500	1
Purchasing	Himuro	2600	2
Purchasing	Tobias	2800	3
Purchasing	Baida	2900	4
Purchasing	Khoo	3100	5
Purchasing	Raphaely	11000	6
Human Resources	Marvis	6500	

DEREF

構文

deref::=



用途

DEREF は、引数 *expr* のオブジェクト参照を戻します。この場合、*expr* はオブジェクトに REF を戻す必要があります。問合せでこのファンクションを使用しない場合、次の例で示すとおり、かわりに REF のオブジェクト ID を戻します。

参照： 6-85 ページの「[MAKE_REF](#)」を参照してください。

例

サンプル・スキーマ oe は、オブジェクト型 *cust_address_typ*（作成方法は、次に示すとおり）を含みます。次の例では、*cust_address_typ* 表を作成し、*cust_address_typ* への REF である 1 つの列を含む別の表を作成します。

```

CREATE TYPE cust_address_typ AS OBJECT
(
  street_address  VARCHAR2(40)
, postal_code     VARCHAR2(10)
, city            VARCHAR2(30)
, state_province  VARCHAR2(10)
, country_id      CHAR(2)
);

/
CREATE TABLE address_table OF cust_address_typ;

CREATE TABLE customer_addresses (
  add_id NUMBER,
  address REF cust_address_typ
  SCOPE IS address_table);

INSERT INTO address_table VALUES
  ('1 First', 'G45 EU8', 'Paris', 'CA', 'US');

INSERT INTO customer_addresses
  SELECT 999, REF(a) FROM address_table a;

SELECT address FROM customer_addresses;
  
```

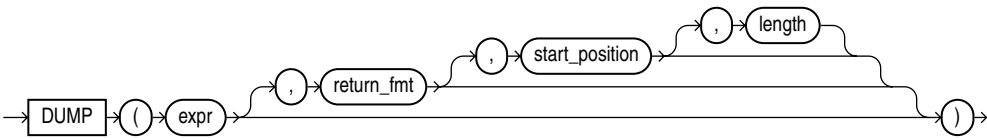
```
ADDRESS
-----
000022020876B2245DBE325C5FE03400400B40DCB176B2245DBE305C5FE03400400B40DCB1

SELECT DEREf(address) FROM customer_addresses;

DEREF(ADDRESS) (STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE, COUNTRY_ID)
-----
CUST_ADDRESS_TYP('1 First', 'G45 EU8', 'Paris', 'CA', 'US')
```

DUMP

構文
dump::=



用途

DUMP は、*expr* のデータ型コード、長さ（バイト単位）および内部表現を含む VARCHAR2 値を戻します。戻される結果は、常にデータベース・キャラクタ・セットの文字です。各コードに対応するデータ型については、2-7 ページの表 2-1 を参照してください。

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

- 8 は、結果を 8 進表記で戻します。
- 10 は、結果を 10 進表記で戻します。
- 16 は、結果を 16 進表記で戻します。
- 17 は、結果を単一文字として戻します。

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、前述の書式のいずれかの値に 1000 を加えて指定します。たとえば、*return_fmt* に 1008 を指定すると、8 進表記で結果が戻り、さらに *expr* のキャラクタ・セット名が得られます。

引数 *start_position* と *length* を組み合わせて、戻される内部表現の部分を指定します。デフォルトでは、10 進表記で全体の内部表現が戻されます。

expr が NULL の場合、このファンクションは NULL を戻します。

注意： このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。詳細は、2-42 ページの「[データ型の比較規則](#)」を参照してください。

例

次の例では、文字列式および列からダンプ情報を抽出する方法を示します。

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

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

```
SELECT DUMP(last_name, 8, 3, 2) "OCTAL"
      FROM employees
      WHERE last_name = 'Hunold';
```

```
OCTAL
-----
Typ=1 Len=6: 156,157
```

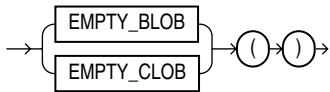
```
SELECT DUMP(last_name, 10, 3, 2) "ASCII"
      FROM employees
      WHERE last_name = 'Hunold';
```

```
ASCII
-----
Typ=1 Len=6: 110,111
```

EMPTY_BLOB、EMPTY_CLOB

構文

empty_lob::=



用途

EMPTY_BLOB および EMPTY_CLOB は、LOB 変数を初期化したり、INSERT または UPDATE 文で LOB 列または属性を EMPTY に初期化できる空の LOB ロケータを返します。EMPTY とは、LOB は初期化されていても、データが移入されていない状態をいいます。

制限事項： このファンクションから戻されるロケータは、DBMS_LOB パッケージまたは OCI へのパラメータとして使用することはできません。

例

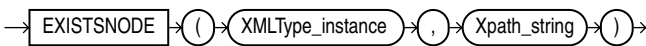
次の例では、サンプル表 pm.print_media の ad_photo 列を EMPTY に初期化します。

```
UPDATE print_media SET ad_photo = EMPTY_BLOB();
```

EXISTSNODE

構文

existsnode::=



用途

EXISTSNODE ファンクションは、パスを使用してドキュメントをトラバースした後にノードが存在するかを判断します。XML ドキュメント、およびパスを指定する VARCHAR2 文字列を含む VARCHAR2 インスタンスを引数として指定します。

戻り値は、NUMBER です。

- ドキュメントに XPath トラバースを適用した後にノードが存在しない場合は、0（ゼロ）が返ります。
- ノードが存在する場合は、0（ゼロ）より大きい値が返ります。

例

次の例では、サンプル表 `oe.warehouses` の `warehouse_spec` 列の XML パスに `/Warehouse/Dock` ノードが存在するかを判断します。

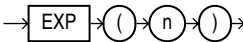
```
SELECT warehouse_id, EXISTSNODE(warehouse_spec, '/Warehouse/Docks')
       "Loading Docks"
FROM warehouses
WHERE warehouse_spec IS NOT NULL;
```

WAREHOUSE_ID	Loading Docks
1	1
2	1
3	1
4	1

EXP

構文

exp::=



用途

EXP は、`e` を `n` 乗した値 (`e = 2.71828183 ...`) を戻します。

例

次の例では、`e` を 4 乗した値を戻します。

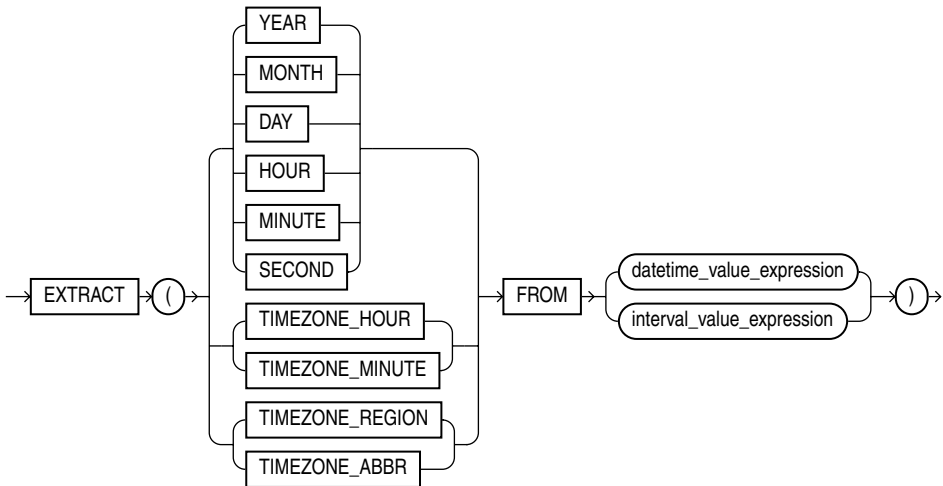
```
SELECT EXP(4) "e to the 4th power" FROM DUAL;
```

e to the 4th power
54.59815

EXTRACT（日時）

構文

extract_datetime::=



用途

EXTRACT（日時）ファンクションは、日時または期間の値の式から指定された日時フィールドの値を抽出し、戻します。TIMEZONE_REGIONを抽出する場合、戻り値は、適切なタイム・ゾーン名を含む文字列です。また、TIMEZONE_ABBR（略称）を抽出する場合、戻り値は、適切な略称を含む文字列です。その他の値を抽出する場合、戻り値はグレゴリオ暦です。タイム・ゾーン値を持つ日時から抽出する場合、戻り値は UTC です。有効なタイム・ゾーン名およびそれに対する略称を表示するには、V\$TIMEZONE_NAMES 動的パフォーマンス・ビューに問合せを実行してください。

注意： 抽出を実行するフィールドは、*datetime_value_expression* または *interval_value_expression* フィールドである必要があります。たとえば、DATE 値からは、YEAR、MONTH および DAY のみを抽出できます。同様に、TIMESTAMP WITH TIME ZONE データ型からは、TIMEZONE_HOUR および TIMEZONE_MINUTE のみを抽出できます。

参照：

- *datetime_value_expression* および *interval_value_expression* の詳細は、2-23 ページの「日時および期間の演算」を参照してください。
- 動的パフォーマンス・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

例

次の例では、数値 1998 を戻します。

```
SELECT EXTRACT(YEAR FROM DATE '1998-03-07') FROM DUAL;

EXTRACT(YEARFROMDATE'1998-03-07')
-----
1998
```

EXTRACT (XML)

構文

extract_xml::=

**用途**

EXTRACT (XML) ファンクションは、EXISTSNODE ファンクションと似ています。VARCHAR2 の XPath 文字列を適用し、XML のフラグメントを含む XMLType インスタンスを戻します。

例

次の例では、サンプル表 `oe.warehouses` の `warehouse_spec` 列の XML パスの `/Warehouse/Dock` ノードの値を抽出します。

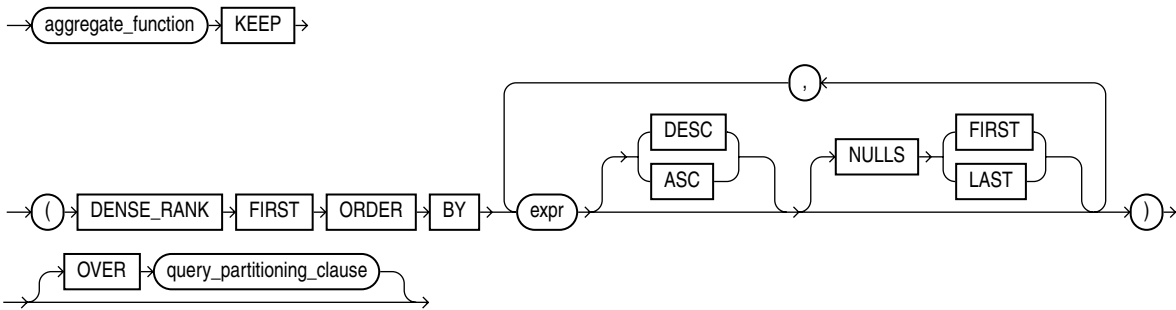
```
SELECT warehouse_name,
       EXTRACT(warehouse_spec, '/Warehouse/Docks').getStringVal()
       "Number of Docks"
FROM warehouses
WHERE warehouse_spec IS NOT NULL;
```

WAREHOUSE_NAME	Number of Docks
-----	-----
Southlake, Texas	<Docks>2</Docks>
San Francisco	<Docks>1</Docks>
New Jersey	
Seattle, Washington	<Docks>3</Docks>

FIRST

構文

first::=



参照： ORDER BY 句および OVER 句の構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

FIRST ファンクションと LAST ファンクションは非常に似ています。両方のファンクションは、与えられたソート指定に対して、FIRST または LAST としてランク付けされた一連の行の一連の値を操作する集計ファンクションおよび分析ファンクションです。1 つの行のみが FIRST または LAST としてランク付けされている場合、集計は、1 つの要素のみを持つセットを操作します。

ソートされたグループの最初または最後の行の値が必要であり、その値がソート・キーでない場合、FIRST および LAST ファンクションは、自己結合またはビューの必要性を排除し、パフォーマンスを向上させます。

- `aggregate_function` は、MIN、MAX、SUM、AVG、COUNT、VARIANCE または STDDEV ファンクションのいずれか 1 つです。FIRST または LAST のいずれかにランク付けされた行の値を操作します。1 つの行のみが FIRST または LAST としてランク付けされている場合、集計は、単一（集計ではない）のセットを操作します。
- `DENSE_RANK FIRST` または `DENSE_RANK LAST` は、最小（FIRST）または最大（LAST）稠密ランク（「オリンピック・ランク」）を持つ行のみを集計することを示します。

OVER 句を指定すると、FIRST および LAST ファンクションを分析ファンクションとして使用できます。`query_partitioning_clause` は、これらのファンクションで有効な OVER 句の一部です。

集計の例

次の例では、デモ表 `hr.employees` の各部門で、歩合が最低である従業員の中での最小給与、および歩合が最高の従業員の中での最大給与を戻します。

```
SELECT department_id,
MIN(salary) KEEP (DENSE_RANK FIRST ORDER BY commission_pct) "Worst",
MAX(salary) KEEP (DENSE_RANK LAST ORDER BY commission_pct) "Best"
  FROM employees
 GROUP BY department_id;
```

DEPARTMENT_ID	Worst	Best
10	4400	4400
20	6000	13000
30	2500	11000
40	6500	6500
50	2100	8200
60	4200	9000
70	10000	10000
80	6100	14000
90	17000	24000
100	6900	12000
110	8300	12000
	7000	7000

分析の例

次の例では、前述の例と同じ計算をしますが、部門の各従業員の結果を戻します。

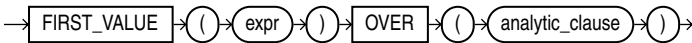
```
SELECT last_name, department_id, salary,
       MIN(salary) KEEP (DENSE_RANK FIRST ORDER BY commission_pct)
         OVER (PARTITION BY department_id) "Worst",
       MAX(salary) KEEP (DENSE_RANK LAST ORDER BY commission_pct)
         OVER (PARTITION BY department_id) "Best"
FROM employees
ORDER BY department_id, salary;
```

LAST_NAME	DEPARTMENT_ID	SALARY	Worst	Best
Whalen	10	4400	4400	4400
Goyal	20	6000	6000	13000
Hartstein	20	13000	6000	13000
.				
.				
.				
Greenberg	100	12000	6900	12000
Gietz	110	8300	8300	12000
Higgins	110	12000	8300	12000

FIRST_VALUE

構文

first_value::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

FIRST_VALUE は分析ファンクションです。これは、順序付けられた値の集合にある最初の値を戻します。

expr には、FIRST_VALUE または他の分析ファンクションを使用できません。他の組込みファンクション式は expr で使用できますが、分析ファンクションはネストできません。

参照： expr の書式の詳細は、4-2 ページの「[SQL 式](#)」を参照してください。

例

次の例では、部門 90 の各従業員について、その部門で給与が一番少ない従業員の名前を選択します。

```
SELECT department_id, last_name, salary, FIRST_VALUE(last_name)
  OVER (ORDER BY salary ASC ROWS UNBOUNDED PRECEDING) AS lowest_sal
  FROM (SELECT * FROM employees WHERE department_id = 90
        ORDER BY employee_id);
```

DEPARTMENT_ID	LAST_NAME	SALARY	LOWEST_SAL
90	Kochhar	17000	Kochhar
90	De Haan	17000	Kochhar
90	King	24000	Kochhar

例では、FIRST_VALUE ファンクションの非決定的な性質が示されています。Kochhar と De Haan の給与は同じであるため、Kochhar の次の行に De Haan があります。Kochhar が最初に表示されているのは、副問合せが戻す行が employee_id で順序付けられているためです。ただし、副問合せが戻す行が employee_id で降順に順序付けられている場合は、次の例に示すとおり、ファンクションは異なる値を戻します。

```
SELECT department_id, last_name, salary, FIRST_VALUE(last_name)
  OVER (ORDER BY salary ASC ROWS UNBOUNDED PRECEDING) as fv
  FROM (SELECT * FROM employees WHERE department_id = 90
        ORDER BY employee_id DESC);
```

DEPARTMENT_ID	LAST_NAME	SALARY	FV
90	De Haan	17000	De Haan
90	Kochhar	17000	De Haan
90	King	24000	De Haan

次の例では、一意キーで順序付けることによって、FIRST_VALUE ファンクションを決定的にする方法を示します。

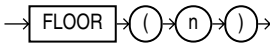
```
SELECT department_id, last_name, salary, hire_date,
  FIRST_VALUE(last_name) OVER
  (ORDER BY salary ASC, hire_date ROWS UNBOUNDED PRECEDING) AS fv
  FROM (SELECT * FROM employees
        WHERE department_id = 90 ORDER BY employee_id DESC);
```

DEPARTMENT_ID	LAST_NAME	SALARY	HIRE_DATE	FV
90	Kochhar	17000	21-SEP-89	Kochhar
90	De Haan	17000	13-JAN-93	Kochhar
90	King	24000	17-JUN-87	Kochhar

FLOOR

構文

floor::=



用途

FLOOR は *n* 以下の最大整数を戻します。

例

次の例では、15.7 以下である最大の整数を戻します。

```
SELECT FLOOR(15.7) "Floor" FROM DUAL;
```

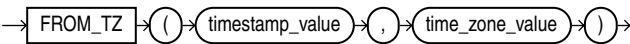
Floor

15

FROM_TZ

構文

from_tz::=



用途

FROM_TZ ファンクションは、タイム・ゾーンのタイムスタンプ値を `TIMESTAMP WITH TIME ZONE` 値に変換します。 *time_zone_value* は、`'TZH:TZM'` 書式の文字列、またはオプションの `TZD` 書式を持つ `TZR` 書式で文字列を戻す文字式です。

例

次の例では、タイムスタンプ値を `TIMESTAMP WITH TIME ZONE` に戻します。

```
SELECT FROM_TZ(TIMESTAMP '2000-03-28 08:00:00', '3:00')
       FROM DUAL;
```

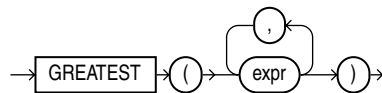
```
FROM_TZ(TIMESTAMP'2000-03-2808:00:00','3:00')
```

```
-----
28-MAR-00 08.00.00 AM +03:00
```

GREATEST

構文

`greatest::=`



用途

`GREATEST` は、`expr` リストの最大値を戻します。比較の前に、2 番目以降のすべての `expr` は最初の `expr` のデータ型に暗黙的に変換されます。**Oracle** は、非空白埋め比較セマンティクスで `expr` を比較します。文字の比較は、データベース・キャラクタ・セットの文字値に基づいて行われます。キャラクタ・セット値の大きい文字が、別の文字より大きい文字とみなされます。このファンクションによって戻される値が文字データの場合、そのデータ型は常に `VARCHAR2` になります。

参照： 2-42 ページの「[データ型の比較規則](#)」を参照してください。

例

次の文では、最大値を持つ文字列を検索します。

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

```
Greatest
```

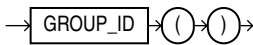
```
-----
```

```
HARRY
```

GROUP_ID

構文

group_id::=



用途

GROUP_ID ファンクションは、GROUP BY 指定の結果から、重複するグループを識別します。このファンクションは、問合せ結果から重複グループを除外する場合に有効です。このファンクションは、重複グループを一意に識別するために、Oracle の数値を戻します。このファンクションは、1つの GROUP BY 句を含む SELECT 文のみに適用できます。

特定のグループ化で *n* 個の重複が存在する場合、GROUP_ID は 0 ～ *n*-1 の範囲の数値を戻します。

例

次の例では、サンプル表 sh.countries および sh.sales の問合せの結果、重複する co.country_region グループ化に値「1」を割り当てます。

```
SELECT co.country_region, co.country_subregion,
       SUM(s.amount_sold) "Revenue",
       GROUP_ID() g
FROM sales s, customers c, countries co
WHERE s.cust_id = c.cust_id AND
      c.country_id = co.country_id AND
      s.time_id = '1-JAN-00' AND
      co.country_region IN ('Americas', 'Europe')
GROUP BY co.country_region,
         ROLLUP (co.country_subregion, co.country_subregion);
```

COUNTRY_REGION	COUNTRY_SUBREGION	Revenue	G
Americas	Northern America	220844	0
Americas	Southern America	10872	0
Europe	Eastern Europe	12751	0
Europe	Western Europe	558686	0
Americas		231716	0
Europe		571437	0
Americas		231716	1
Europe		571437	1

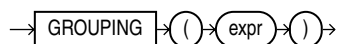
文の最後に次の HAVING 句を追加すると、GROUP_ID が 1 より小さい行のみを戻すことができます。

```
HAVING GROUP_ID() < 1
```

GROUPING

構文

grouping::=



用途

GROUPING ファンクションは、通常のグループ化された行と超集合行を区別します。ROLLUP や CUBE などの GROUP BY の拡張機能は、一連のすべての値を表す NULL を含む超集合行を生成します。GROUPING ファンクションを使用すると、通常の行に含まれる NULL と超集合行で一連のすべての値を表す NULL を区別できます。

GROUPING ファンクションの *expr* は、GROUP BY 句の式のいずれかと一致する必要があります。行の *expr* の値が一連のすべての値を表す NULL の場合、ファンクションは 1 の値を返します。それ以外の場合は、0 (ゼロ) を返します。GROUPING ファンクションは、Oracle の数値を返します。

参照： これらの用語の詳細は、17-19 ページの「SELECT 文」の [group_by_clause](#) を参照してください。

例

次の例では、サンプル表 `hr.departments` および `hr.employees` で、GROUPING ファンクションが 1 (表の通常の行ではなく超集合行) を返す場合、それ以外の場合に表示される NULL のかわりに、文字列「All Jobs」が「JOB」列に表示されます。

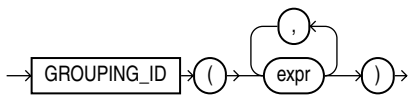
```
SELECT DECODE(GROUPING(department_name), 1, 'All Departments',
             department_name) AS department,
       DECODE(GROUPING(job_id), 1, 'All Jobs', job_id) AS job,
       COUNT(*) "Total Empl", AVG(salary) * 12 "Average Sal"
FROM   employees e, departments d
WHERE  d.department_id = e.department_id
GROUP BY ROLLUP (department_name, job_id);
```

DEPARTMENT	JOB	Total Empl	Average Sal
Accounting	AC_ACCOUNT	1	99600
Accounting	AC_MGR	1	144000

Accounting	All Jobs	2	121800
Administration	AD_ASST	1	52800
Administration	All Jobs	1	52800
Executive	AD_PRES	1	288000
Executive	AD_VP	2	204000
Executive	All Jobs	3	232000
Finance	FI_ACCOUNT	5	95040
Finance	FI_MGR	1	144000
Finance	All Jobs	6	103200
.			
.			
.			

GROUPING_ID

構文
grouping_id::=



用途

GROUPING_ID ファンクションは、行に関連する GROUPING ビット・ベクトルに対応する数値を戻します。GROUPING_ID は、ROLLUP や CUBE など、GROUP BY の拡張機能および GROUPING ファンクションを含む SELECT 文にのみ適用できます。多くの GROUP BY 式を含む問合せでは、多くの GROUPING ファンクションを必要とする特定の行の GROUP BY レベルを指定するため、非常に複雑な SQL になります。GROUPING_ID は、このような場合に有効です。

GROUPING_ID は、複数の GROUPING ファンクションの結果をビット・ベクトル（1 と 0 を組み合せた文字列）に連結したものと同じです。GROUPING_ID を使用すると、複数の GROUPING ファンクションを使用する必要がなくなり、行のフィルタ条件の表記が簡単になります。GROUPING_ID を使用すると、要求する行が単一の条件（GROUPING_ID = n）によって識別されるため、行のフィルタ処理が簡単になります。このファンクションは、1 つの表で複数のレベルの集計を格納する場合に、特に有効です。

例

次の例では、サンプル表 `sh.sales` の問合せからグループ化 ID を抽出する方法を示します。

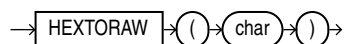
```
SELECT channel_id, promo_id, sum(amount_sold) s_sales,
       GROUPING(channel_id) gc,
       GROUPING(promo_id) gp,
       GROUPING_ID(channel_id, promo_id) gcp,
       GROUPING_ID(promo_id, channel_id) gpc
FROM sales
WHERE promo_id > 496
GROUP BY CUBE(channel_id, promo_id);
```

C	PROMO_ID	S_SALES	GC	GP	GCP	GPC
-	-	-	-	-	-	-
C	498	28024.25	0	0	0	0
C	499	25042	0	0	0	0
C		53066.25	0	1	1	2
I	498	54428.2	0	0	0	0
I	499	72725.25	0	0	0	0
I		127153.45	0	1	1	2
P	498	35801.75	0	0	0	0
P	499	21041.15	0	0	0	0
P		56842.9	0	1	1	2
S	498	95413.05	0	0	0	0
S	499	88706.75	0	0	0	0
S		184119.8	0	1	1	2
T	498	5147	0	0	0	0
T	499	16789.45	0	0	0	0
T		21936.45	0	1	1	2
	498	218814.25	1	0	2	1
	499	224304.6	1	0	2	1
		443118.85	1	1	3	3

HEXTORAW

構文

hextoraw::=



用途

HEXTORAW は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 キャラクタ・セットで 16 進数を含む *char* を RAW 値に変換します。

注意： このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。詳細は、2-42 ページの「[データ型の比較規則](#)」を参照してください。

例

次の例では、RAW 列を含む簡単な表を作成し、RAW に変換された 16 進数値を挿入します。

```
CREATE TABLE test (raw_col RAW(10));
```

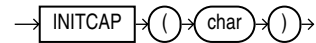
```
INSERT INTO test VALUES (HEXTORAW('7D'));
```

参照： 2-25 ページの「[RAW データ型と LONG RAW データ型](#)」および 6-115 ページの「[RAWTOHEX](#)」を参照してください。

INITCAP

構文

initcap::=



用途

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

char は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型です。戻り値は、*char* と同じデータ型です。

注意： このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。詳細は、2-42 ページの「[データ型の比較規則](#)」を参照してください。

例

次の例では、文字列にある各単語を大文字で始めます。

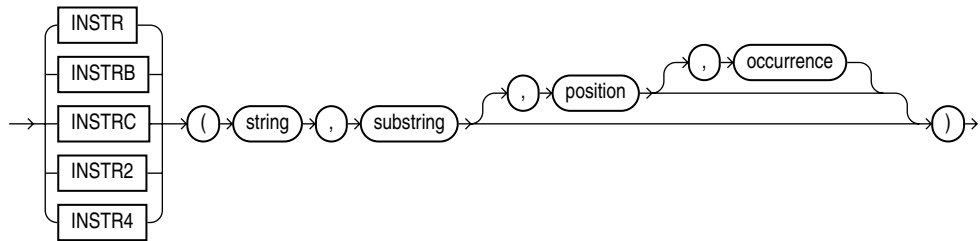
```
SELECT INITCAP('the soap') "Capitals" FROM DUAL;
```

```
Capitals
-----
The Soap
```

INSTR

構文

instr::=



用途

INSTR ファンクションは、*string* の *substring* を検索します。このファンクションは、最初に現れた文字 *string* の位置を示す整数を戻します。INSTR は、入力キャラクター・セットによって定義された文字を使用して、文字列を算出します。INSTRB は、文字のかわりにバイトを使用します。INSTRC は、完全な Unicode キャラクタを使用します。INSTR2 は、UCS2 コードポイントを使用します。INSTR4 は、UCS4 コードポイントを使用します。

- *position* は、Oracle が検索を開始する文字 *string* の位置を示す 0（ゼロ）以外の整数です。*position* が負の場合、Oracle は *string* の終わりから逆方向にカウントおよび検索します。
- *occurrence* は、検索する *string* が現れたことを示す整数です。*occurrence* の値は正である必要があります。

string および *substring* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻り値のデータ型は NUMBER です。

position および *occurrence* のデフォルト値は 1 です。この場合、Oracle は *string* の最初の文字から検索を開始します。検索対象は *substring* が最初に現れる位置です。戻り値は、*position* の値にかかわらず、*string* の先頭に関係し、文字で表されます。検索が失敗した（*string* の *position* 番目の文字の後に *substring* が *occurrence* 回現れない）場合、戻り値は 0 となります。

例

次の例では、文字列「CORPORATE FLOOR」で文字列「OR」の検索を3番目の文字から開始します。文字列 CORPORATE FLOOR で2回目に現れる「OR」の開始位置を戻します。

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

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

次の例では、最後から3番目の文字から検索を開始します。

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

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

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

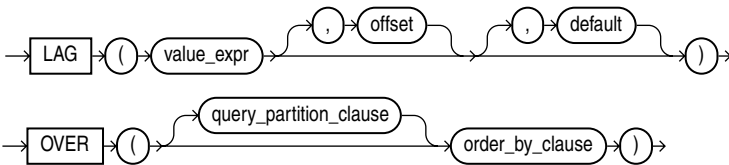
```
SELECT INSTRB('CORPORATE FLOOR','OR',5,2) "Instring in bytes"
       FROM DUAL;
```

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

LAG

構文

lag::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析関数](#)」を参照してください。

用途

LAG は分析ファンクションです。これは、自己結合せずに、表の 1 つ以上の行へ同時アクセスを行います。問合せから戻される一連の行およびカーソル位置を指定すると、LAG は、その位置より前にある指定された物理オフセットにある行へアクセスします。

offset を指定しない場合、デフォルト値は 1 です。オフセットがウィンドウの有効範囲を超えた場合、オプションの *default* 値が戻されます。*default* を指定しない場合、デフォルト値は NULL です。

value_expr には、LAG または他の分析ファンクションを使用できません。他の組込みファンクション式は *expr* で使用できますが、分析ファンクションはネストできません。

参照： *expr* の書式の詳細は、4-2 ページの「SQL 式」を参照してください。

例

次の例では、employees 表の各販売員について、その販売員の直前に雇用された従業員の給与を示します。

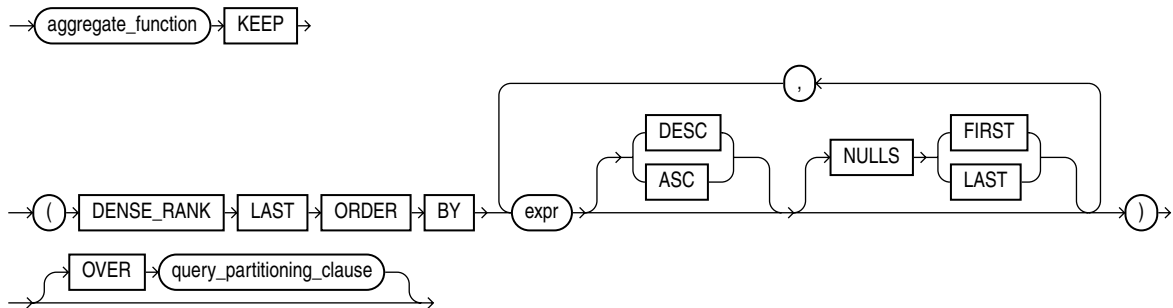
```
SELECT last_name, hire_date, salary,
       LAG(salary, 1, 0) OVER (ORDER BY hire_date) AS prev_sal
FROM employees
WHERE job_id = 'PU_CLERK';
```

LAST_NAME	HIRE_DATE	SALARY	PREV_SAL
-----	-----	-----	-----
Khoo	18-MAY-95	3100	0
Tobias	24-JUL-97	2800	3100
Baida	24-DEC-97	2900	2800
Himuro	15-NOV-98	2600	2900
Colmenares	10-AUG-99	2500	2600

LAST

構文

last::=



参照： `query_partitioning_clause` の構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

FIRST ファンクションと LAST ファンクションは非常に似ています。両方のファンクションは、与えられたソート指定に対して、FIRST または LAST としてランク付けされた一連の行の一連の値を操作する集計ファンクションおよび分析ファンクションです。1 つの行のみが FIRST または LAST としてランク付けされている場合、集計は、1 つの要素のみを持つセットを操作します。

ソートされたグループの最初または最後の行の値が必要であり、その値がソート・キーでない場合、FIRST および LAST ファンクションは、自己結合またはビューの必要性を排除し、パフォーマンスを向上させます。

- `aggregate_function` は、MIN、MAX、SUM、AVG、COUNT、VARIANCE または STDDEV ファンクションのいずれか 1 つです。FIRST または LAST のいずれかにランク付けされた行の値を操作します。1 つの行のみが FIRST または LAST としてランク付けされている場合、集計は、単一（集計ではない）のセットを操作します。
- `DENSE_RANK FIRST` または `DENSE_RANK LAST` は、最小（FIRST）または最大（LAST）稠密ランク（「オリンピック・ランク」）を持つ行のみを集計することを示します。

OVER 句を指定すると、FIRST および LAST ファンクションを分析ファンクションとして使用できます。`query_partitioning_clause` は、これらのファンクションで有効な OVER 句の一部です。

集計の例

次の例では、デモ表 `hr.employees` の各部門で、コミッションが最低である従業員の中での最小給与、およびコミッションが最高の従業員の中での最大給与を戻します。

```
SELECT department_id,
MIN(salary) KEEP (DENSE_RANK FIRST ORDER BY commission_pct) "Worst",
MAX(salary) KEEP (DENSE_RANK LAST ORDER BY commission_pct) "Best"
FROM employees
GROUP BY department_id;
```

DEPARTMENT_ID	Worst	Best
10	4400	4400
20	6000	13000
30	2500	11000
40	6500	6500
50	2100	8200
60	4200	9000
70	10000	10000
80	6100	14000
90	17000	24000
100	6900	12000
110	8300	12000
	7000	7000

分析の例

次の例では、前述の例と同じ計算をしますが、部門の各従業員の結果を戻します。

```
SELECT last_name, department_id, salary,
MIN(salary) KEEP (DENSE_RANK FIRST ORDER BY commission_pct)
OVER (PARTITION BY department_id) "Worst",
MAX(salary) KEEP (DENSE_RANK LAST ORDER BY commission_pct)
OVER (PARTITION BY department_id) "Best"
FROM employees
ORDER BY department_id, salary;
```

LAST_NAME	DEPARTMENT_ID	SALARY	Worst	Best
Whalen	10	4400	4400	4400
Goyal	20	6000	6000	13000
Hartstein	20	13000	6000	13000
.				
.				
.				
Greenberg	100	12000	6900	12000
Gietz	110	8300	8300	12000
Higgins	110	12000	8300	12000

LAST_DAY

構文

last_day::=

→ LAST_DAY ((date)) →

用途

LAST_DAY は、*date* を含む月の最終日の日付を戻します。

例

次の文は、現在の月の残りの日数を確認します。

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

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

次の例では、各従業員の雇用開始日に 5ヶ月を加えて、評価結果を戻します。

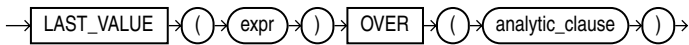
```
SELECT last_name, hire_date, TO_CHAR(
       ADD_MONTHS(LAST_DAY(hire_date), 5)) "Eval Date"
FROM employees;
```

LAST_NAME	HIRE_DATE	Eval Date
King	17-JUN-87	30-NOV-87
Kochhar	21-SEP-89	28-FEB-90
De Haan	13-JAN-93	30-JUN-93
Hunold	03-JAN-90	30-JUN-90
Ernst	21-MAY-91	31-OCT-91
Austin	25-JUN-97	30-NOV-97
Pataballa	05-FEB-98	31-JUL-98
Lorentz	07-FEB-99	31-JUL-99
.		
.		
.		

LAST_VALUE

構文

last_value::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

LAST_VALUE は分析ファンクションです。これは、順序付けられた値の集合にある最後の値を戻します。

expr には、LAST_VALUE または他の分析ファンクションを使用できません。他の組込みファンクション式は expr で使用できますが、分析ファンクションはネストできません。

参照： expr の書式の詳細は、4-2 ページの「[SQL 式](#)」を参照してください。

例

次の例では、給与が一番高い従業員の雇用開始日を各行に戻します。

```
SELECT last_name, salary, hire_date, LAST_VALUE(hire_date) OVER
  (ORDER BY salary
   ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS lv
FROM (SELECT * FROM employees WHERE department_id = 90
      ORDER BY hire_date);
```

LAST_NAME	SALARY	HIRE_DATE	LV
Kochhar	17000	21-SEP-89	17-JUN-87
De Haan	17000	13-JAN-93	17-JUN-87
King	24000	17-JUN-87	17-JUN-87

この例では、LAST_VALUE ファンクションの非決定的でない性質を示しています。Kochhar と De Haan の給与は同じであるため、Kochhar の次の行に De Haan があります。Kochhar が最初に表示されているのは、副問合せの行が hire_date で順序付けられているためです。ただし、行が hire_date で降順に順序付けられている場合は、次の例に示すとおり、ファンクションは異なる値を返します。

```
SELECT last_name, salary, hire_date, LAST_VALUE(hire_date) OVER
  (ORDER BY salary
   ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS lv
FROM (SELECT * FROM employees WHERE department_id = 90
      ORDER BY hire_date DESC);
```

LAST_NAME	SALARY	HIRE_DATE	LV
De Haan	17000	13-JAN-93	17-JUN-87
Kochhar	17000	21-SEP-89	17-JUN-87
King	24000	17-JUN-87	17-JUN-87

次の 2 つの例では、一意キーで順序付けることによって、LAST_VALUE ファンクションを決定的にする方法を示しています。salary および hire_date でファンクション内を順序付けると、副問合せの順序付けにかかわらず、同じ結果が返されます。

```
SELECT last_name, salary, hire_date, LAST_VALUE(hire_date) OVER
  (ORDER BY salary, hire_date
   ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS lv
FROM (SELECT * FROM employees WHERE department_id = 90
      ORDER BY hire_date);
```

LAST_NAME	SALARY	HIRE_DATE	LV
Kochhar	17000	21-SEP-89	17-JUN-87
De Haan	17000	13-JAN-93	17-JUN-87
King	24000	17-JUN-87	17-JUN-87

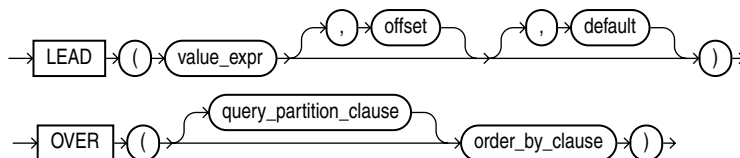
```
SELECT last_name, salary, hire_date, LAST_VALUE(hire_date) OVER
  (ORDER BY salary, hire_date
   ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS lv
FROM (SELECT * FROM employees WHERE department_id = 90
      ORDER BY hire_date DESC);
```

LAST_NAME	SALARY	HIRE_DATE	LV
Kochhar	17000	21-SEP-89	17-JUN-87
De Haan	17000	13-JAN-93	17-JUN-87
King	24000	17-JUN-87	17-JUN-87

LEAD

構文

lead::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

LEAD は分析ファンクションです。これは、自己結合せずに、表の 1 つ以上の行へ同時アクセスを行います。問合せから戻される一連の行およびカーソル位置を指定すると、LEAD は、その位置より後にある指定された物理オフセットの行へアクセスします。

offset を指定しない場合、デフォルト値は 1 です。オフセットが表の有効範囲を超えた場合、オプションの *default* 値が戻されます。*default* を指定しない場合、デフォルト値は NULL です。

value_expr には、LEAD または他の分析ファンクションを使用できません。他の組込みファンクション式は *value_expr* で使用できますが、分析ファンクションはネストできません。

参照： *expr* の書式の詳細は、4-2 ページの「[SQL 式](#)」を参照してください。

例

次の例では、employees 表の各従業員について、その従業員の直後に雇用された従業員の雇用開始日を示します。

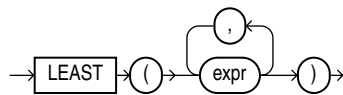
```
SELECT last_name, hire_date,  
       LEAD(hire_date, 1) OVER (ORDER BY hire_date) AS "NextHired"  
FROM employees WHERE department_id = 30;
```

LAST_NAME	HIRE_DATE	NextHired
-----	-----	-----
Raphaely	07-DEC-94	18-MAY-95
Khoo	18-MAY-95	24-JUL-97
Tobias	24-JUL-97	24-DEC-97
Baida	24-DEC-97	15-NOV-98
Himuro	15-NOV-98	10-AUG-99
Colmenares	10-AUG-99	

LEAST

構文

least::=



用途

LEAST は、リストされた *expr* 内の最小値を返します。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、非空白埋め比較セマンティクスで *expr* を比較します。このファンクションによって戻される値が文字データの場合、そのデータ型は常に VARCHAR2 になります。

例

次の文は、LEAST ファンクションの使用例です。

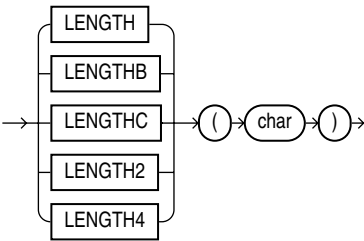
```
SELECT LEAST('HARRY', 'HARRIOT', 'HAROLD') "LEAST"  
FROM DUAL;
```

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

LENGTH

構文

length::=



用途

LENGTH ファンクションは、*char* の長さを戻します。LENGTH は、入力キャラクタ・セットによって定義された文字を使用して、長さを算出します。LENGTHB は、文字のかわりにバイトを使用します。LENGTHC は、完全な Unicode キャラクタを使用します。LENGTH2 は、UCS2 コードポイントを使用します。LENGTH4 は、UCS4 コードポイントを使用します。

char は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻り値のデータ型は NUMBER です。*char* のデータ型が CHAR の場合、その長さにはすべての後続空白が含まれます。*char* が NULL の場合、このファンクションは NULL を戻します。

例

次の例では、シングルバイトおよびマルチバイトのデータベース・キャラクタ・セットを使用する LENGTH ファンクションを使用します。

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

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

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

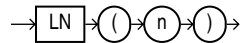
```
SELECT LENGTHB ('CANDIDE') "Length in bytes"
FROM DUAL;
```

```
Length in bytes
-----
14
```

LN

構文

ln::=



用途

LN は、 n の自然対数を返します (n は正の数)。

例

次の例では、95 の自然対数を返します。

```
SELECT LN(95) "Natural log of 95" FROM DUAL;
```

```
Natural log of 95
```

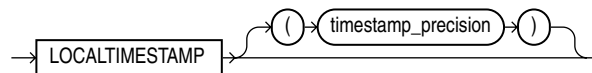
```
-----
```

```
4.55387689
```

LOCALTIMESTAMP

構文

localtimestamp::=



用途

LOCALTIMESTAMP ファンクションは、セッションのタイム・ゾーンの現在の日付および時刻を `TIMESTAMP` データ型の値で返します。LOCALTIMESTAMP は、TIMESTAMP 値を返します。これに対して、CURRENT_TIMESTAMP は、TIMESTAMP WITH TIME ZONE 値を返します。

参照： 6-45 ページの「[CURRENT_TIMESTAMP](#)」を参照してください。

例

次の例では、LOCALTIMESTAMP と CURRENT_TIMESTAMP の相違を示します。

```
ALTER SESSION SET TIME_ZONE = '-5:00';
SELECT CURRENT_TIMESTAMP, LOCALTIMESTAMP FROM DUAL;
```

CURRENT_TIMESTAMP	LOCALTIMESTAMP
04-APR-00 01.27.18.999220 PM -05:00	04-APR-00 01.27.19 PM

```
ALTER SESSION SET TIME_ZONE = '-8:00';
SELECT CURRENT_TIMESTAMP, LOCALTIMESTAMP FROM DUAL;
```

CURRENT_TIMESTAMP	LOCALTIMESTAMP
04-APR-00 10.27.45.132474 AM -08:00	04-APR-00 10.27.451 AM

LOG

構文

log::=



用途

LOG は、 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

次の例では、100 の対数を戻します。

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

Log base 10 of 100
2

LOWER

構文

lower::=



用途

LOWER は、すべての文字を小文字にして *char* を戻します。*char* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻り値は、*char* と同じデータ型です。

例

次の例では、文字列を小文字にして戻します。

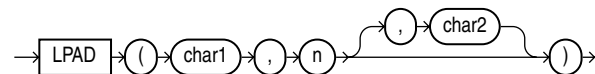
```
SELECT LOWER('MR. SCOTT MCMILLAN') "Lowercase"
      FROM DUAL;
```

```
Lowercase
-----
mr. scott mcmillan
```

LPAD

構文

lpad::=



用途

LPAD は、*char1* の左側に *char2* に指定した文字を連続的に埋め込んで *n* 桁にして戻します。*char2* のデフォルト値は空白 1 個です。*char1* が *n* より長い場合、このファンクションは *n* に収まる *char1* の一部を戻します。

char1 および *char2* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻される文字列は、VARCHAR2 データ型であり、*char1* と同じキャラクタ・セットです。

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、表示される文字列の長さが文字列の文字数と異なる場合もあります。

例

次の例では、文字「*」を左側に埋め込みます。

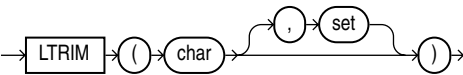
```
SELECT LPAD('Page 1',15,'*.') "LPAD example"
      FROM DUAL;
```

```
LPAD example
-----
*.*.*.*.*Page 1
```

LTRIM

構文

`ltrim::=`



用途

LTRIM は、*char* の左側にあつて *set* に指定された文字を削除します。*set* のデフォルト値は空白 1 個です。*char* が文字リテラルの場合、引用符で囲む必要があります。Oracle は *char* の先頭文字からスキャンし始め、*set* に指定された文字をすべて削除します。*set* に指定された文字以外の文字が見つかった時点で結果を戻します。

char および *set* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻される文字列は、VARCHAR2 データ型であり、*char* と同じキャラクタ・セットです。

例

次の例では、文字列の左端にある *x* および *y* をすべて削除します。

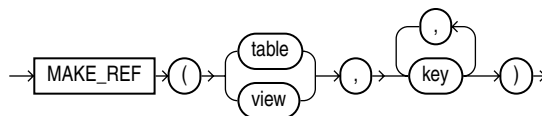
```
SELECT LTRIM('xyxXxyLAST WORD','xy') "LTRIM example"
      FROM DUAL;
```

```
LTRIM example
-----
XxyLAST WORD
```

MAKE_REF

構文

make_ref::=



用途

MAKE_REF は、オブジェクト識別子が主キーに基づいている、オブジェクト・ビューの行またはオブジェクト表の行に対する REF を作成します。

参照：

- オブジェクト・ビューの詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。
- 6-51 ページの「[DEREF](#)」を参照してください。

例

次の例では、オブジェクト・ビューにある行への REF を作成します。

```

CREATE TABLE employee (eno NUMBER, ename VARCHAR2(20),
    salary NUMBER, PRIMARY KEY (eno, ename));
CREATE TYPE emp_type AS OBJECT
    (eno NUMBER, ename CHAR(20), salary NUMBER);
CREATE VIEW emp_view OF emp_type
    WITH OBJECT IDENTIFIER (eno, ename)
    AS SELECT * FROM employee;
SELECT MAKE_REF(emp_view, 1, 'jack') FROM DUAL;

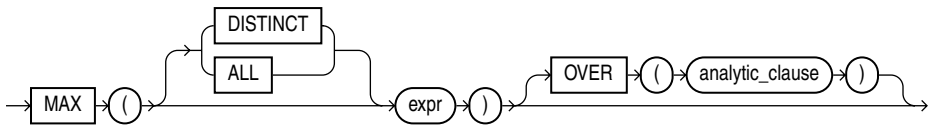
MAKE_REF(EMP_VIEW,1,'JACK')
-----
000067030A0063420D06E06F3C00C1E03400400B40DCB10000001C2601000100020029000000000000F06
00810100140100002A0007000A8401FE0000001F02C102146A61636B2020202020202020202020202020
20200000000000000000000000000000000000000000000000000000000000000000000000000000000

```

MAX

構文

max::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

MAX は *expr* の最大値を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

DISTINCT を指定する場合は、*analytic_clause* の *query_partition_clause* のみ指定できます。 *order_by_clause* および *windowing_clause* は指定できません。

参照：

- 6-6 ページの「[集計ファンクション](#)」を参照してください。
- *expr* の書式の詳細は、4-2 ページの「[SQL 式](#)」を参照してください。

集計の例

次の例では、hr.employees 表の最高給与を検索します。

```
SELECT MAX(salary) "Maximum" FROM employees;
```

```
Maximum
-----
24000
```

分析の例

次の例では、各従業員について、その従業員と所属が同じ従業員のうち、一番高い給与を計算します。

```
SELECT manager_id, last_name, salary,
       MAX(salary) OVER (PARTITION BY manager_id) AS mgr_max
FROM employees;
```

MANAGER_ID	LAST_NAME	SALARY	MGR_MAX
100	Kochhar	17000	17000
100	De Haan	17000	17000
100	Raphaely	11000	17000
100	Kaufling	7900	17000
100	Fripp	8200	17000
100	Weiss	8000	17000
.			
.			
.			

この問合せを述語のある親問合せで囲むと、各部門で給与の一番高い従業員がわかります。

```
SELECT manager_id, last_name, salary
FROM (SELECT manager_id, last_name, salary,
       MAX(salary) OVER (PARTITION BY manager_id) AS rmax_sal
FROM employees)
WHERE salary = rmax_sal;
```

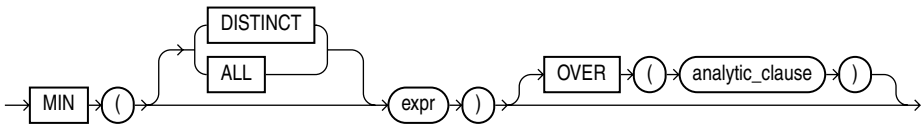
MANAGER_ID	LAST_NAME	SALARY
100	Kochhar	17000
100	De Haan	17000
101	Greenberg	12000
101	Higgins	12000
102	Hunold	9000
103	Ernst	6000
108	Faviet	9000
114	Khoo	3100
120	Nayer	3200
120	Taylor	3200
121	Sarchand	4200
122	Chung	3800
123	Bell	4000
124	Rajs	3500
145	Tucker	10000

146	King	10000
147	Vishney	10500
148	Ozer	11500
149	Abel	11000
201	Goyal	6000
205	Gietz	8300
	King	24000

MIN

構文

min::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

MIN は、*expr* の最小値を返します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

DISTINCT を指定する場合は、*analytic_clause* の *query_partition_clause* のみ指定できます。*order_by_clause* および *windowing_clause* は指定できません。

参照：

- 6-6 ページの「[集計ファンクション](#)」を参照してください。
- *expr* の書式の詳細は、4-2 ページの「[SQL 式](#)」を参照してください。

集計の例

次の例では、hr.employees 表の最初の雇用開始日を返します。

```
SELECT MIN(hire_date) "Earliest" FROM employees;
```

```
Earliest
-----
17-JUN-87
```

分析の例

次の例では、各従業員について、その従業員が雇用された日以前に雇用された従業員を検索します。その従業員と所属が同じ従業員のサブセットを決定し、そのサブセット内で一番低い給与を戻します。

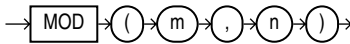
```
SELECT manager_id, last_name, hire_date, salary,
       MIN(salary) OVER(PARTITION BY manager_id ORDER BY hire_date
                        RANGE UNBOUNDED PRECEDING) as p_cmin
FROM employees;
```

MANAGER_ID	LAST_NAME	HIRE_DATE	SALARY	P_CMIN
100	Kochhar	21-SEP-89	17000	17000
100	De Haan	13-JAN-93	17000	17000
100	Raphaely	07-DEC-94	11000	11000
100	Kaufling	01-MAY-95	7900	7900
100	Hartstein	17-FEB-96	13000	7900
100	Weiss	18-JUL-96	8000	7900
100	Russell	01-OCT-96	14000	7900
100	Partners	05-JAN-97	13500	7900
100	Errazuriz	10-MAR-97	12000	7900
.				
.				
.				

MOD

構文

mod::=



用途

MOD は、 m を n で割ったときの余りを戻します。 n が 0 の場合は、 m を戻します。

例

次の例では、11 を 4 で割ったときの余りを戻します。

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

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

このファンクションは、*m* が負の場合には古典数学のモジュール・ファンクションとは異なる動作をします。古典数学のモジュール・ファンクションは、次の公式を用いた MOD ファンクションで表すことができます。

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

次の表は、MOD ファンクションと古典数学のモジュール・ファンクションの相違を示します。

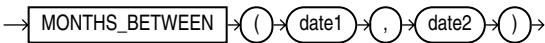
M	N	MOD(M,N)	古典数学のモジュール・ファンクション
11	4	3	3
11	-4	3	-1
-11	4	-3	1
-11	-4	-3	-3

参照： 6-62 ページの「[FLOOR](#)」を参照してください。

MONTHS_BETWEEN

構文

months_between::=



用途

MONTHS_BETWEEN は、日付 *date1* と *date2* の間の月数を戻します。*date1* が *date2* より後の日付の場合、結果は正の値になります。*date1* が *date2* より前の日付の場合、結果は負の値になります。*date1* および *date2* が、月の同じ日または月の最終日の場合、結果は常に整数になります。それ以外の場合、Oracle は結果の小数部を 1 か月 31 日として計算します。コンポーネント *date1* および *date2* の相違も考慮されます。

例

次の例では、2つの日付間の月数を計算します。

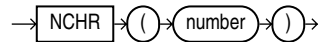
```
SELECT MONTHS_BETWEEN
      (TO_DATE('02-02-1995','MM-DD-YYYY'),
       TO_DATE('01-01-1995','MM-DD-YYYY')) "Months"
FROM DUAL;

      Months
-----
1.03225806
```

NCHR

構文

nchr::=



用途

NCHR は、各国語キャラクタ・セットの *number* と同等のバイナリを持つ文字を戻します。このファンクションは、CHR ファンクションを USING NCHAR_CS 句とともに使用した場合と同じ結果が得られます。

参照： 6-28 ページの「[CHR](#)」を参照してください。

例

次の例では、NCHAR 文字 187 を戻します。

```
SELECT NCHR(187) FROM DUAL;

NC
--
>

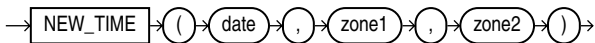
SELECT CHR(187 USING NCHAR_CS) FROM DUAL;

CH
--
>
```

NEW_TIME

構文

new_time::=



用途

NEW_TIME は、タイム・ゾーン *zone1* の日時が *date* の時点のタイム・ゾーン *zone2* の日時を戻します。このファンクションを使用する前に、24 時間で表示されるように、NLS_DATE_FORMAT パラメータを設定する必要があります。

引数 *zone1* および *zone2* には、次のテキスト列のいずれかを指定できます。

- AST、ADT: 大西洋標準時および大西洋夏時間
- BST、BDT: ベーリング標準時およびベーリング夏時間
- CST、CDT: 中央標準時および中央夏時間
- EST、EDT: 東部標準時および東部夏時間
- GMT: グリニッジ標準時
- HST、HDT: アラスカ - ハワイ標準時およびアラスカ - ハワイ夏時間
- MST、MDT: 山岳部標準時および東部夏時間
- NST: ニューファンドランド標準時
- PST、PDT: 太平洋標準時および太平洋夏時間
- YST、YDT: ユーコン標準時およびユーコン夏時間

例

次の例では、指定された太平洋標準時と同等の大西洋標準時を戻します。

```
ALTER SESSION SET NLS_DATE_FORMAT =
'DD-MON-YYYY HH24:MI:SS';

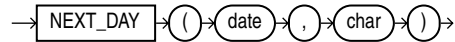
SELECT NEW_TIME(TO_DATE(
'11-10-99 01:23:45', 'MM-DD-YY HH24:MI:SS'),
'AST', 'PST') "New Date and Time" FROM DUAL;

New Date and Time
-----
09-NOV-1999 21:23:45
```

NEXT_DAY

構文

next_day::=



用途

NEXT_DAY は、*char* で指定した曜日で、日付 *date* より後の最初の日付を戻します。引数 *char* は、セッションの日付言語での曜日である必要があります（フルネームでも省略形でも可）。必要最小限の文字数は、省略形の文字数です。有効な省略形の後に続けて文字が入力されていても、それらの文字は無視されます。戻り値は、引数 *date* と同じ時、分および秒のコンポーネントを持っています。

例

次の例では、2001 年 2 月 2 日以降の最初の火曜日の日付を戻します。

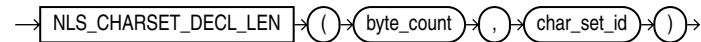
```
SELECT NEXT_DAY('02-FEB-2001','TUESDAY') "NEXT DAY"
      FROM DUAL;
```

```
NEXT DAY
-----
06-FEB-2001
```

NLS_CHARSET_DECL_LEN

構文

nls_charset_decl_len::=



用途

NLS_CHARSET_DECL_LEN は、NCHAR 列の宣言の幅を（文字数で）戻します。*byte_count* 引数は、列の幅です。*char_set_id* 引数は、列のキャラクタ・セット ID です。

例

次の例では、マルチバイト・キャラクタ・セットの列の幅が 200 バイトである文字の数を戻します。

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

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

NLS_CHARSET_ID

構文

nls_charset_id::=



用途

NLS_CHARSET_ID は、キャラクタ・セット名 *text* に対応するキャラクタ・セットの ID 番号を戻します。引数 *text* は、実行時の VARCHAR2 値です。*text* 値 'CHAR_CS' は、サーバーのデータベース・キャラクタ・セットの ID 番号を戻します。*text* 値 'NCHAR_CS' は、サーバーの各国語キャラクタ・セットの ID 番号を戻します。

無効なキャラクタ・セット名を指定すると、NULL が戻されます。

例

次の例では、キャラクタ・セットの ID を戻します。

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

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

参照： キャラクタ・セット名のリストについては、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

NLS_CHARSET_NAME

構文

nls_charset_name::=



用途

NLS_CHARSET_NAME は、ID 番号 *number* に対応するキャラクタ・セット名を返します。キャラクタ・セット名は、データベース・キャラクタ・セットの VARCHAR2 値として戻されます。

number が有効なキャラクタ・セット ID として認識されない場合は、このファンクションは NULL を返します。

例

次の例では、ID 番号が 2 のキャラクタ・セットを返します。

```
SELECT NLS_CHARSET_NAME(2)
FROM DUAL;
```

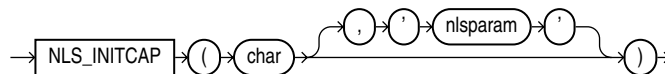
```
NLS_CH
-----
WE8DEC
```

参照： キャラクタ・セット ID のリストについては、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

NLS_INITCAP

構文

nls_initcap::=



用途

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

char および *nlsparam* は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型です。戻される文字列は、VARCHAR2 データ型であり、*char* と同じキャラクタ・セットです。

nlsparam の値は次の書式で指定します。

```
'NLS_SORT = sort'
```

sort は、言語ソート基準または BINARY のいずれかです。言語ソート基準は、大文字と小文字の変換のために特別な言語要件を処理します。これらの要件によって、*char* と異なる長さの値が戻される場合があります。*nlsparam* を省略すると、このファンクションはセッションに対してデフォルトのソート基準を使用します。

注意： このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。詳細は、2-42 ページの「[データ型の比較規則](#)」を参照してください。

例

次に、ファンクションによって言語ソート基準がどのように異なる値を戻すかを示します。

```
SELECT NLS_INITCAP
       ('ijsland') "InitCap" FROM DUAL;

InitCap
-----
Ijsland

SELECT NLS_INITCAP
       ('ijsland', 'NLS_SORT = XDutch') "InitCap"
FROM DUAL;

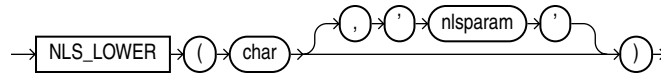
InitCap
-----
IJsland
```

参照： ソート基準の詳細は、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

NLS_LOWER

構文

nls_lower::=



用途

NLS_LOWER は、すべての文字を小文字にして *char* を戻します。

char および *nlsparem* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻される文字列は、VARCHAR2 データ型であり、*char* と同じキャラクター・セットです。

NLS_param の書式および用途は、NLS_INITCAP ファンクションと同じです。

例

次の例では、XGerman 言語ソート順序を使用する文字列 'citta' を戻します。

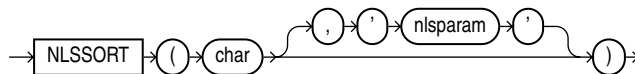
```
SELECT NLS_LOWER
       ('CITTA'', 'NLS_SORT = XGerman') "Lowercase"
FROM DUAL;
```

```
Lowerc
-----
citta'
```

NLSSORT

構文

nlssort::=



用途

NLSSORT は、*char* のソートに使用される文字列のバイトを戻します。

char および *nlsparam* は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型です。戻される文字列は RAW データ型です。

'*nlsparam*' の値は次の書式で指定します。

```
'NLS_SORT = sort'
```

sort は、言語ソート基準または BINARY のいずれかです。'*nlsparam*' を省略すると、このファンクションは、セッションに対してデフォルトのソート基準を使用します。BINARY を指定すると、このファンクションは *char* を戻します。

注意： このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。詳細は、2-42 ページの「[データ型の比較規則](#)」を参照してください。

例

このファンクションを使用すると、文字列の 2 進値を基にした比較ではなく、言語ソート基準を基にした比較を指定できます。次の例では、*name* 列を持つテスト表が 2 つの値を含むことを想定します。

```
SELECT * FROM test ORDER BY name;
```

```
NAME
```

```
-----
```

```
Gaardiner
```

```
Gaberd
```

```
SELECT * FROM test
```

```
ORDER BY NLSSORT(name, 'NLS_SORT = XDanish');
```

```
NAME
```

```
-----
```

```
Gaberd
```

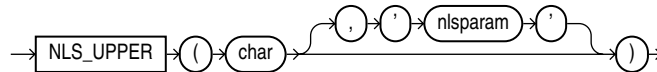
```
Gaardiner
```

参照： ソート基準の詳細は、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

NLS_UPPER

構文

nls_upper::=



用途

NLS_UPPER は、すべての文字を大文字にして *char* を戻します。

char および *nlsparam* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻される文字列は、VARCHAR2 データ型であり、*char* と同じキャラクター・セットです。

NLS_param の書式および用途は、NLS_INITCAP ファンクションと同じです。

例

次の例では、すべての文字を大文字に変換して文字列を戻します。

```
SELECT NLS_UPPER ('große') "Uppercase"
FROM DUAL;
```

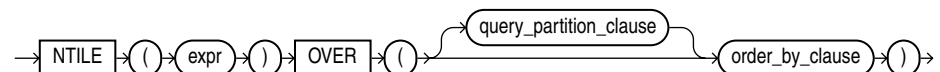
```
Upper
-----
GROßE
```

参照： 6-95 ページの「[NLS_INITCAP](#)」を参照してください。

NTILE

構文

ntile::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

NTILE は分析ファンクションです。これは、順序付けられたデータセットを *expr* に指定した数のバケットに分割し、適切なバケット番号を各行に割り当てます。バケットには 1 ～ *expr* の番号が付けられます。ここで、*expr* には、パーティションごとに、正の定数を指定する必要があります。

バケット内の行数は、最大で 1 異なります。残りの値（バケットで割った行数の余り）は、バケット 1 から順に、1 行ずつ分割されます。

expr が行数より大きい場合、行数と等しい数のバケットに行が入れられ、余りのバケットは空になります。

expr には、NTILE または他の分析ファンクションを使用できません。他の組み込みファンクション式は *expr* で使用できますが、分析ファンクションはネストできません。

参照： *expr* の書式の詳細は、4-2 ページの「SQL 式」を参照してください。

例

次の例では、部門 100 の *oe.employees* 表の *salary* 列の値を 4 つのバケットに分割します。*salary* 列は、この部門に 6 つの値を持つため、2 つの余分な値（6 を 4 で割った余り）は、バケット 1 および 2 に割り当てられます。そのため、これらのバケットは、バケット 3 または 4 より値が 1 つ多くなります。

```
SELECT last_name, salary, NTILE(4) OVER (ORDER BY salary DESC)
       AS quartile FROM employees
       WHERE department_id = 100;
```

LAST_NAME	SALARY	QUARTILE
-----	-----	-----
Greenberg	12000	1
Faviet	9000	1
Chen	8200	2
Urman	7800	2
Sciarra	7700	3
Popp	6900	4

NULLIF

構文

nullif::=



用途

NULLIF ファンクションは、*expr1* と *expr2* を比較します。同じである場合は、NULL を返します。異なる場合は、*expr1* を返します。*expr1* には、リテラル NULL を指定できません。

NULLIF ファンクションは、次の CASE 式と論理的に同じです。

```

CASE WHEN expr1 = expr2 THEN NULL ELSE expr1 END

```

参照： 4-6 ページの「CASE 式」を参照してください。

例

次の例では、サンプル・スキーマ hr から、雇用後に職種が変更した従業員を検索します。これは、employees 表の現行の job_id が job_history 表の job_id と異なるかどうかによって識別されます。

```

SELECT e.last_name, NULLIF(e.job_id, j.job_id) "Old Job ID"
      FROM employees e, job_history j
      WHERE e.employee_id = j.employee_id;

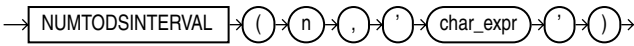
```

LAST_NAME	Old Job ID
De Haan	AD_VP
Kochhar	AD_VP
Kochhar	AD_VP
Hartstein	MK_MAN
Raphaely	PU_MAN
Kaufling	ST_MAN
Whalen	
Taylor	
Taylor	SA_REP
Whalen	AD_ASST

NUMTODSINTERVAL

構文

numtodsinterval::=



用途

NUMTODSINTERVAL は、*n* を INTERVAL DAY TO SECOND リテラルに変換します。*n* は、数値、または結果が数値になる式になります。*char_expr* のデータ型は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 です。*char_expr* の値は *n* の単位を指定し、結果が次の文字列値のいずれかになる必要があります。

- 'DAY'
- 'HOUR'
- 'MINUTE'
- 'SECOND'

char_expr は、大文字と小文字を区別しません。カッコ内の先行値および後続値は無視されます。デフォルトでは、戻り値の精度は 9 です。

例

次の例では、各従業員について、その従業員の雇用日から数えて 100 日以内に雇用された従業員数を所属別に計算します。

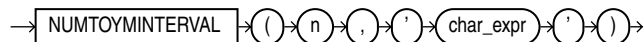
```
SELECT manager_id, last_name, hire_date,
       COUNT(*) OVER (PARTITION BY manager_id ORDER BY hire_date
                      RANGE NUMTODSINTERVAL(100, 'day') PRECEDING) AS t_count
FROM employees;
```

MANAGER_ID	LAST_NAME	HIRE_DATE	T_COUNT
100	Kochhar	21-SEP-89	1
100	De Haan	13-JAN-93	1
100	Raphaely	07-DEC-94	1
100	Kaufling	01-MAY-95	1
100	Hartstein	17-FEB-96	1
.			
.			
.			
149	Grant	24-MAY-99	1
149	Johnson	04-JAN-00	1
201	Goyal	17-AUG-97	1
205	Gietz	07-JUN-94	1
	King	17-JUN-87	1

NUMTOYMINTERVAL

構文

numtoyminterval::=



用途

NUMTOYMINTERVAL は、数値 *n* を INTERVAL YEAR TO MONTH リテラルに変換します。*n* は、数値、または結果が数値になる式になります。*char_expr* のデータ型は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 です。*char_expr* の値は *n* の単位を指定し、結果が次の文字列値のいずれかになる必要があります。

- 'YEAR'
- 'MONTH'

char_expr は、大文字と小文字を区別しません。カッコ内の先行値および後続値は無視されます。デフォルトでは、戻り値の精度は 9 です。

例

次の例では、各従業員について、その従業員の雇用日から 1 年の間に雇用された従業員の給与の合計を計算します。

```
SELECT last_name, hire_date, salary, SUM(salary)
  OVER (ORDER BY hire_date
        RANGE NUMTOYMINTERVAL(1,'year') PRECEDING) AS t_sal
FROM employees;
```

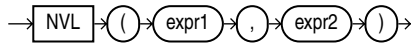
LAST_NAME	HIRE_DATE	SALARY	T_SAL

King	17-JUN-87	24000	24000
Whalen	17-SEP-87	4400	28400
Kochhar	21-SEP-89	17000	17000
.			
.			
Markle	08-MAR-00	2200	112400
Ande	24-MAR-00	6400	106500
Banda	21-APR-00	6200	109400
Kumar	21-APR-00	6100	109400

NVL

構文

nvl::=



用途

expr1 が NULL の場合、NVL は expr2 を返します。expr1 が NULL でない場合、NVL は expr1 を返します。引数 expr1 および expr2 は、任意のデータ型を持つことができます。2 つの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、戻り値のデータ型は VARCHAR2 になり、expr1 のキャラクタ・セットになります。

例

次の例では、従業員が歩合を受け取らない場合、従業員名および従業員の歩合「Not Applicable」を表示します。

```
SELECT last_name, NVL(TO_CHAR(commission_pct), 'Not Applicable')
       "COMMISSION" FROM employees
WHERE last_name LIKE 'B%';
```

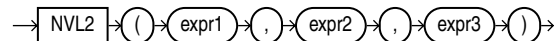
LAST_NAME	COMMISSION

Baer	Not Applicable
Baida	Not Applicable
Banda	.11
Bates	.16
Bell	Not Applicable
Bernstein	.26
Bissot	Not Applicable
Bloom	.21
Bull	Not Applicable

NVL2

構文

nvl2::=



用途

expr1 が NULL でない場合、NVL2 は *expr2* を戻します。*expr1* が NULL の場合、NVL2 は *expr3* を戻します。引数 *expr1* は、任意のデータ型を持つことができます。引数 *expr2* および *expr3* は、LONG 以外の任意のデータ型を持つことができます。

expr2 と *expr3* のデータ型が異なり、*expr3* が NULL 定数でない場合、Oracle は比較前に *expr3* を *expr2* のデータ型に変換します。この場合、データ型の変換は必要ありません。

戻り値のデータ型は、常に *expr2* のデータ型と同じになります。ただし、*expr2* が文字データの場合、戻り値のデータ型は VARCHAR2 になります。

例

次の例では、employees の commission_pct 列が NULL かどうかによって、従業員の収入が給与と歩合か、または給与のみかを示します。

```

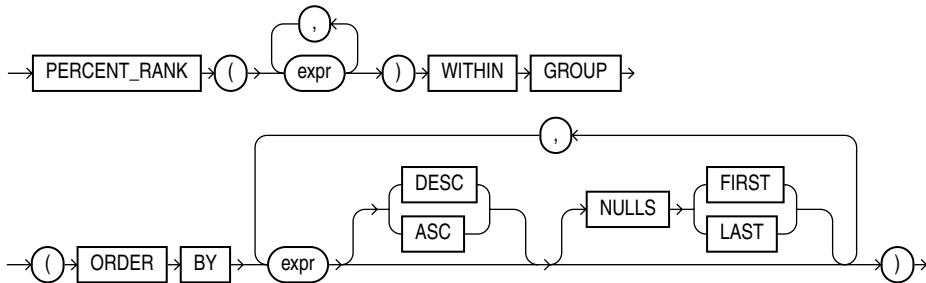
SELECT last_name, salary, NVL2(commission_pct,
    salary + (salary * commission_pct), salary) income
FROM employees WHERE last_name like 'B%';
  
```

LAST_NAME	SALARY	INCOME
-----	-----	-----
Baer	10000	10000
Baida	2900	2900
Banda	6200	6882
Bates	7300	8468
Bell	4000	4000
Bernstein	9500	11970
Bissot	3300	3300
Bloom	10000	12100
Bull	4100	4100

PERCENT_RANK

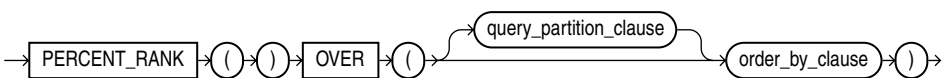
集計の構文

percent_rank_aggregate::=



分析の構文

percent_rank_analytic::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

PERCENT_RANK は、CUME_DIST（累積分布）ファンクションと似ています。
PERCENT_RANK が戻す値の範囲は、0 ～ 1（0 および 1 を含む）です。すべての集合の最初の行の PERCENT_RANK は 0（ゼロ）になります。

- 集計ファンクションとしての PERCENT_RANK は、ファンクションの引数および対応するソート指定によって識別される不確定な行 R を計算し、行 R のランクから 1 を引いて、集計グループ内の行の数で割ります。不確定な行 R を Oracle が集計する行のグループに挿入するように計算します。このファンクションの引数は、各集計グループ内の 1 つの不確定行を識別します。このため、すべての引数は、集計グループ内で定数式と評価される必要があります。定数引数式および集計の ORDER BY 句の式の位置は、一致します。このため、引数の数は同じであり、その型は互換性がある必要があります。
- 分析ファンクションとしての PERCENT_RANK は、行 R に対して、R のランクから 1 引いた数を評価される行数（問合せ結果セット全体またはパーティション）より 1 少ない数で割ります。

集計の例

次の例では、サンプル表 `hr.employees` から、給与が 15,500 ドルであり、歩合が 5% である不確定な従業員のパーセント・ランクを計算します。

```
SELECT PERCENT_RANK(15000, .05) WITHIN GROUP
       (ORDER BY salary, commission_pct) "Percent-Rank"
FROM employees;
```

Percent-Rank

```
-----
.971962617
```

分析の例

次の例では、各従業員について、その従業員の部門内での給与のパーセント・ランクを計算します。

```
SELECT department_id, last_name, salary,
       PERCENT_RANK()
         OVER (PARTITION BY department_id ORDER BY salary DESC) AS pr
FROM employees
ORDER BY pr, salary;
```

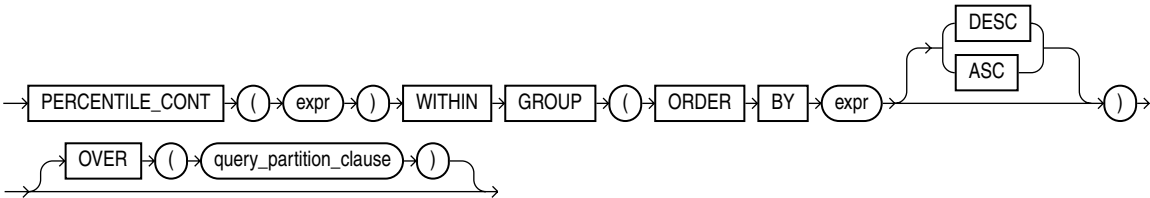
DEPARTMENT_ID	LAST_NAME	SALARY	PR

10	Whalen	4400	0
40	Marvis	6500	0
.			
.			
.			
80	Vishney	10500	.176470588
50	Everett	3900	.181818182
30	Khoo	3100	.2
.			
.			
.			
80	Johnson	6200	.941176471
50	Markle	2200	.954545455
50	Philtanker	2200	.954545455
50	Olson	2100	1

PERCENTILE_CONT

構文

percentile_cont::=



参照： OVER 句の構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

PERCENTILE_CONT ファンクションは、連続分散モデルを想定する逆分散関数です。このファンクションは、パーセンタイル値およびソート指定を用い、そのソート指定に従ってそのパーセンタイル値に該当する補間された値を戻します。計算では、NULL は無視されます。

最初の *expr* は、パーセンタイル値であるため、0 ～ 1 の数値で評価します。この *expr* は、各集計グループ内の定数である必要があります。ORDER BY 句には、Oracle が補間を実行できる型である数値または日時値の単一式を指定します。

PERCENTILE_CONT の結果は、順序付けされた後の値間の直線補間によって計算されます。集計グループで、パーセンタイル値 (P) および行数 (N) を使用すると、ソート指定に従って行を順序付けた後の行数を計算します。この行数 (RN) は、計算式 $RN = (1 + (P * (N - 1)))$ に従って計算されます。集計ファンクションの最終結果は、行番号が $CRN = CEILING(RN)$ および $FRN = FLOOR(RN)$ の行の値間の直線補間によって計算されます。

最終結果は次のとおりです。

```
if (CRN = FRN = RN) then
  (value of expression from row at RN)
else
  (CRN - RN) * (value of expression for row at FRN) +
  (RN - FRN) * (value of expression for row at CRN)
```

PERCENTILE_CONT ファンクションは、分析ファンクションとしても使用できます。その場合、OVER 句には、*query_partitioning_clause* のみを指定できます。各行に対して、各パーティション内の一連の値から、指定されたパーセンタイルに該当する値を戻します。

集計の例

次の例では、各部門の給与の中央値を計算します。

```
SELECT department_id,
       PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY salary DESC)
       "Median cont",
       PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY salary DESC)
       "Median disc"
FROM employees
GROUP BY department_id;
```

DEPARTMENT_ID	Median-cont	Median-disc
10	4400	4400
20	9500	13000
30	2850	2900
40	6500	6500
50	3100	3100
60	4800	4800
70	10000	10000
80	8800	8800
90	17000	17000
100	8000	8200
110	10150	12000

PERCENTILE_CONT および PERCENTILE_DISC は、異なる結果を返す場合があります。PERCENTILE_CONT は、直線補間後の結果を計算します。PERCENTILE_DISC は、集計された一連の値から値のみを返します。この例に示すように、パーセンタイル値が 0.5 の場合、PERCENTILE_CONT は、偶数の要素を持つグループの中間の 2 つの値の平均を返します。それに対して、PERCENTILE_DISC は、中間の 2 つの値の最初の値を返します。奇数の要素を持つ集計グループでは、どちらのファンクションも中間の要素の値を返します。

分析の例

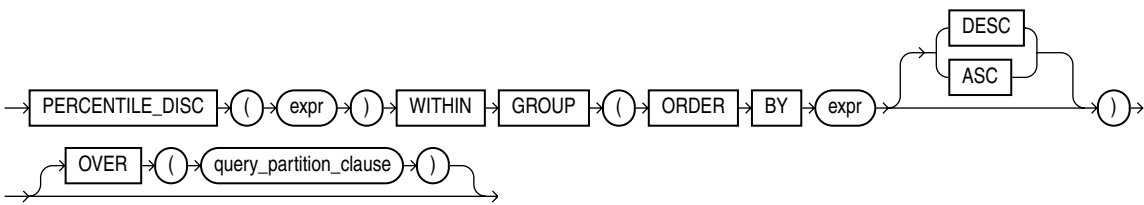
次の例では、0.5 のパーセンタイル (Percent_Rank) に対応する部門 60 の中央値は 4800 です。部門 30 の給与にパーセンタイル 0.5 がないため、2900 (パーセンタイル 0.4) ~ 2800 (パーセンタイル 0.6) の範囲で 2850 に評価される中央値が補間される必要があります。

```
SELECT last_name, salary, department_id,
       PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY salary DESC)
       OVER (PARTITION BY department_id) "Percentile_Cont",
       PERCENT_RANK()
       OVER (PARTITION BY department_id ORDER BY salary DESC) "Percent_Rank"
FROM employees WHERE department_id IN (30, 60);
```

LAST_NAME	SALARY	DEPARTMENT_ID	Percentile_Cont	Percent_Rank
Raphaely	11000	30	2850	0
Khoo	3100	30	2850	.2
Baida	2900	30	2850	.4
Tobias	2800	30	2850	.6
Himuro	2600	30	2850	.8
Colmenares	2500	30	2850	1
Hunold	9000	60	4800	0
Ernst	6000	60	4800	.25
Austin	4800	60	4800	.5
Pataballa	4800	60	4800	.5
Lorentz	4200	60	4800	1

PERCENTILE_DISC

構文
percentile_disc::=



参照： OVER 句の構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「分析ファンクション」を参照してください。

用途

PERCENTILE_DISC ファンクションは、不連続分散モデルを想定する逆分散関数です。このファンクションでは、パーセンタイル値およびソート指定を指定し、そのセットから要素を戻します。計算では、NULL は無視されます。

最初の expr は、パーセンタイル値であるため、0 ～ 1 の数値で評価します。この expr は、各集計グループ内の定数である必要があります。ORDER BY 句には、ソート可能な型の単一式を指定します。

指定されたパーセンタイル値 P に対して、PERCENTILE_DISC ファンクションは、ORDER BY 句の式の値をソートし、P 以上である（同じソート指定に従う）最小 CUME_DIST 値を持つ値を戻します。

集計の例

6-108 ページの「[PERCENTILE_CONT](#)」にある例を参照してください。

分析の例

次の例では、サンプル表 `hr.employees` の各従業員の給与の中央値不連続パーセンタイルを計算します。

```
SELECT last_name, salary, department_id,
       PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY salary DESC)
       OVER (PARTITION BY department_id) "Percentile_Disc",
       CUME_DIST() OVER (PARTITION BY department_id ORDER BY salary DESC)
       "Cume_Dist"
FROM employees where department_id in (20, 60);
```

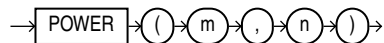
LAST_NAME	SALARY	DEPARTMENT_ID	Percentile_Disc	Cume_Dist
Raphaely	11000	30	2900	.166666667
Khoo	3100	30	2900	.333333333
Baida	2900	30	2900	.5
Tobias	2800	30	2900	.666666667
Himuro	2600	30	2900	.833333333
Colmenares	2500	30	2900	1
Hunold	9000	60	4800	.2
Ernst	6000	60	4800	.4
Austin	4800	60	4800	.8
Pataballa	4800	60	4800	.8
Lorentz	4200	60	4800	1

部門 30 の中央値の値は 2900 です。この値の対応するパーセンタイル (Cume_Dist) は、0.5 以上の最小値です。部門 60 の中央値の値は 4800 です。この値の対応するパーセンタイルは、0.5 以上の最小値です。

POWER

構文

power::=



用途

POWER は、 m を n 乗した値を返します。底 m および指数 n は任意の数です。ただし、 m が負の場合、 n は整数である必要があります。

例

次の例では、3 の 2 乗を戻します。

```
SELECT POWER(3,2) "Raised" FROM DUAL;
```

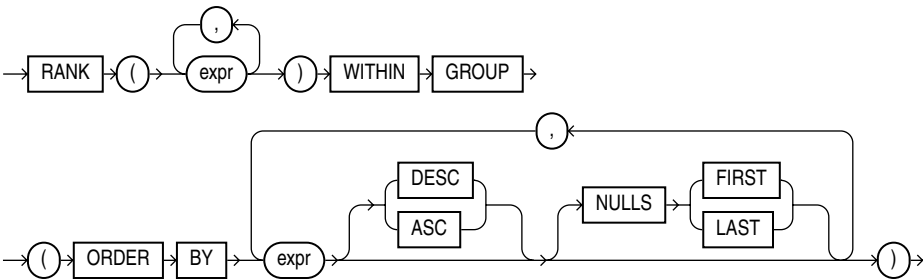
Raised

9

RANK

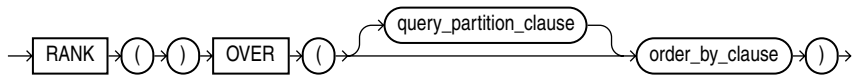
集計の構文

rank_aggregate::=



分析の構文

rank_analytic::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

RANK は、一連の値における値のランクを計算します。ランク付け基準と同じ値を持つ行は、同じランクになります。Oracle は連結行の数を連結ランクに追加して、次のランクを計算します。そのため、ランクは連続した数値でない場合があります。

- 集計ファンクションとして使用する RANK は、ファンクションの引数として識別される不確定な行のランクを、指定されたソート指定に従って計算します。ファンクションの引数は、各集計グループの単一行を識別するため、すべての引数は各集計グループ内で定数式に評価される必要があります。定数引数式および集計の ORDER BY 句の式の位置は、一致します。このため、引数の数は同じであり、その型は互換性がある必要があります。
- 分析ファンクションとして使用する RANK は、ある問合せが戻す他の行について、その問合せが戻す各行のランクを計算します。この計算は、*order_by_clause* にある *value_exprs* の値に基づいて行われます。

集計の例

次の例では、サンプル表 `hr.employees` から、給与が 15,500 ドルであり、歩合が 5% である不確定な従業員のランクを計算します。

```
SELECT RANK(15500, .05) WITHIN GROUP
      (ORDER BY salary, commission_pct) "Rank"
FROM employees;
```

```
Rank
-----
105
```

同様に、次の問合せは、従業員の給与から給与が 15,500 ドルのランクを戻します。

```
SELECT RANK(15500) WITHIN GROUP
      (ORDER BY salary DESC) "Rank of 15500"
FROM employees;
```

```
Rank of 15500
-----
4
```

分析の例

次の例では、サンプル・スキーマ hr の各部門内の従業員を、その給与および歩合に基づいてランク付けします。給与が同じ場合は同じランクになるため、連続しないランクになります。この例と、6-49 ページの「DENSE_RANK」の例を比較してください。

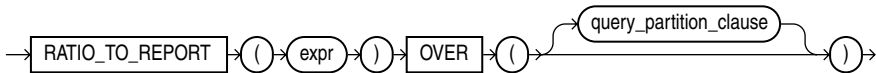
```
SELECT department_id, last_name, salary, commission_pct,
       RANK() OVER (PARTITION BY department_id
                    ORDER BY salary DESC, commission_pct) "Rank"
FROM employees;
```

DEPARTMENT_ID	LAST_NAME	SALARY	COMMISSION_PCT	Rank
10	Whalen	4400		1
20	Hartstein	13000		1
20	Goyal	6000		2
30	Raphaely	11000		1
30	Khoo	3100		2
30	Baida	2900		3
30	Tobias	2800		4
.				
.				
.				

RATIO_TO_REPORT

構文

ratio_to_report::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「分析ファンクション」を参照してください。

用途

RATIO_TO_REPORT は分析ファンクションです。このファンクションは、ある値の集合の合計に対する、その値の比率を計算します。expr が NULL の場合は、値も NULL になります。

値の集合は、query_partition_clause によって決まります。この句を省略すると、比率は、問合せによって戻されるすべての行で計算されます。

`expr` には、`RATIO_TO_REPORT` または他の分析ファンクションを使用できません。他の組み込みファンクション式は `expr` で使用できますが、分析ファンクションはネストできません。

参照： `expr` の書式の詳細は、4-2 ページの「[SQL 式](#)」を参照してください。

例

次の例では、すべての事務員の給与の合計に対する各事務員の給与の比率を計算します。

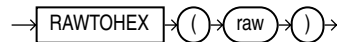
```
SELECT last_name, salary, RATIO_TO_REPORT(salary) OVER () AS rr
FROM employees
WHERE job_id = 'PU_CLERK';
```

LAST_NAME	SALARY	RR
Khoo	3100	.223021583
Baida	2900	.208633094
Tobias	2800	.201438849
Himuro	2600	.18705036
Colmenares	2500	.179856115

RAWTOHEX

構文

`rawtohex::=`



用途

`RAWTOHEX` は、`raw` を 16 進で表した文字値に変換します。`raw` 引数は、`RAW` または `BLOB` のいずれかのデータ型です。

例

```
SELECT RAWTOHEX(raw_column) "Graphics"
FROM graphics;
```

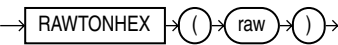
```
Graphics
-----
7D
```

参照： 2-25 ページの「[RAW データ型と LONG RAW データ型](#)」および 6-68 ページの「[HEXTORAW](#)」を参照してください。

RAWTONHEX

構文

rawtonhex::=



用途

RAWTONHEX ファンクションは、*raw* を 16 進で表した NVARCHAR2 文字値に変換します。

例

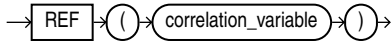
```
SELECT RAWTONHEX(raw_column),
       DUMP ( RAWTONHEX (raw_column) ) "DUMP"
FROM graphics;
```

RAWTONHEX (RA)	DUMP
7D	Typ=1 Len=4: 0,55,0,68

REF

構文

ref::=



用途

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相関変数（表別名）が指定されます。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。サンプル・スキーマ oe には、次のように定義された *cust_address_typ* という型が含まれます。

Attribute	Type
STREET_ADDRESS	VARCHAR2 (40)
POSTAL_CODE	VARCHAR2 (10)
CITY	VARCHAR2 (30)
STATE_PROVINCE	VARCHAR2 (10)
COUNTRY_ID	CHAR (2)

例

次の例では、`cust_address_tpy` に基づく表（address 表）を作成後、その表に行を挿入し、その表からその型のオブジェクト・インスタンスの REF 値を取り出します。

```
CREATE TABLE addresses OF cust_address_tpy;

INSERT INTO addresses VALUES (
    '123 First Street', '4GF H1J', 'Our Town', 'Ourcounty', 'US');

SELECT REF(e) FROM addresses e;

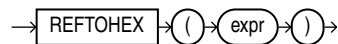
REF(E)
-----
00002802097CD1261E51925B60E0340800208254367CD1261E51905B60E034080020825436010101820000
```

参照：『Oracle9i データベース概要』を参照してください。

REFTOHEX

構文

reftohex::=



用途

REFTOHEX は、引数 `expr` を 16 進で表した文字値に変換します。`expr` は、REF を戻す必要があります。

例

次の例では、値 mgr を 16 進で表した文字値に変換します。

```
CREATE TYPE emp_type AS OBJECT
  (eno NUMBER, ename VARCHAR2(20), salary NUMBER);
CREATE TABLE emp_table OF emp_type
  (primary key (eno, ename));
CREATE TABLE dept
  (dno NUMBER, mgr REF emp_type SCOPE IS emp_table);
INSERT INTO emp_table VALUES (10, 'jack', 50000);
INSERT INTO dept SELECT 10, REF(e) FROM emp_table e;
SELECT REFTOHEX(mgr) FROM dept;

REFTOHEX(MGR)
-----
000022020881D5BDBBC83532FCE03408002082543681D5BDBBC83432FCE034080020825436
```

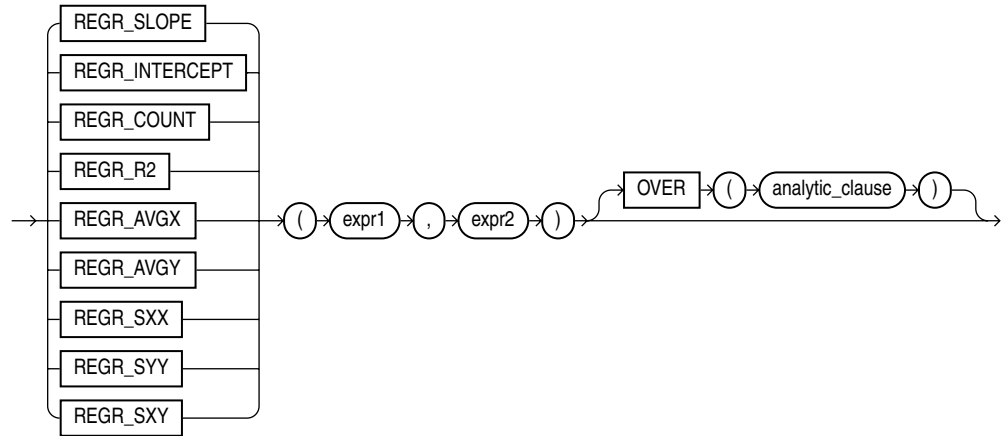
REGR_（線形リグレーション）ファンクション

線形リグレーション・ファンクションは次のとおりです。

- REGR_SLOPE
- REGR_INTERCEPT
- REGR_COUNT
- REGR_R2
- REGR_AVGX
- REGR_AVGY
- REGR_SXX
- REGR_SYY
- REGR_SXY

構文

linear_regr::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

線形リグレッション・ファンクションは、微分最小 2 乗法で求めたリグレッション直線を数値の組の集合に対応付けます。これは、集計ファンクションまたは分析ファンクションとして使用できます。

参照：

- 6-6 ページの「[集計ファンクション](#)」を参照してください。
- `expr` の書式の詳細は、4-2 ページの「[SQL 式](#)」を参照してください。

Oracle は、`expr1` または `expr2` が NULL であるすべての組を排除した後、このファンクションを (`expr1`, `expr2`) の集合に適用します。Oracle は、データでの引渡し中、同時にすべてのリグレッション・ファンクションを計算します。

`expr1` は、従属変数の値 (y 値) として解析されます。`expr2` は、独立変数の値 (x 値) として解析されます。これらの式は、両方とも数値である必要があります。

- `REGR_SLOPE` は、直線の傾きを戻します。戻り値は数値で、NULL になる場合もあります。NULL (`expr1`, `expr2`) の組を排除した後、このファンクションは次の計算を行います。

$$\text{COVAR_POP}(\text{expr1}, \text{expr2}) / \text{VAR_POP}(\text{expr2})$$

- REGR_INTERCEPT は、リグレッション・ファンクションの y 切片を返します。戻り値は数値で、NULL になる場合もあります。NULL (expr1, expr2) の組を排除した後、このファンクションは次の計算を行います。

$$\text{AVG}(\text{expr1}) - \text{REGR_SLOPE}(\text{expr1}, \text{expr2}) * \text{AVG}(\text{expr2})$$

- REGR_COUNT は整数を返します。この整数は、リグレッション・ファンクションに適応させるために使用される NULL でない数値の組です。
- REGR_R2 は、リグレッションに対する確定係数の (「R の 2 乗」または「goodness of fit」) を返します。戻り値は数値で、NULL になる場合もあります。VAR_POP (expr1) および VAR_POP (expr2) は、NULL の組が排除された後に評価されます。戻り値は次のとおりです。

$$\text{NULL if VAR_POP}(\text{expr2}) = 0$$

$$1 \text{ if VAR_POP}(\text{expr1}) = 0 \text{ and} \\ \text{VAR_POP}(\text{expr2}) \neq 0$$

$$\text{POWER}(\text{CORR}(\text{expr1}, \text{expr2}), 2) \text{ if VAR_POP}(\text{expr1}) > 0 \text{ and} \\ \text{VAR_POP}(\text{expr2}) \neq 0$$

これ以外のすべてのリグレッション・ファンクションの戻り値は数値で、NULL になる場合もあります。

- REGR_AVGX は、リグレッション直線の独立変数 (expr2) の平均を求めます。NULL (expr1, expr2) の組を排除した後、このファンクションは次の計算を行います。

$$\text{AVG}(\text{expr2})$$

- REGR_AVGY は、リグレッション直線の従属変数 (expr1) の平均を求めます。NULL (expr1, expr2) の組を排除した後、このファンクションは次の計算を行います。

$$\text{AVG}(\text{expr1})$$

REGR_SXY、REGR_SXX、REGR_SYY は補助ファンクションです。これらは、様々な診断統計の計算に使用されます。

- NULL (expr1, expr2) の組を排除した後、REGR_SXX は次の計算を行います。

$$\text{REGR_COUNT}(\text{expr1}, \text{expr2}) * \text{VAR_POP}(\text{expr2})$$

- NULL (expr1, expr2) の組を排除した後、REGR_SYY は次の計算を行います。

$$\text{REGR_COUNT}(\text{expr1}, \text{expr2}) * \text{VAR_POP}(\text{expr1})$$

- NULL (expr1, expr2) の組を排除した後、REGR_SXY は次の計算を行います。

$$\text{REGR_COUNT}(\text{expr1}, \text{expr2}) * \text{COVAR_POP}(\text{expr1}, \text{expr2})$$

次の例は、サンプル表 sh.sales および sh.products に基づいています。

一般的な線形リグレッションの例

次の例では、様々な線形リグレッション・ファンクションを比較します。

```
SELECT
s.channel_id,
REGR_SLOPE(s.quantity_sold, p.prod_list_price) SLOPE ,
REGR_INTERCEPT(s.quantity_sold, p.prod_list_price) INTCP ,
REGR_R2(s.quantity_sold, p.prod_list_price) RSQR ,
REGR_COUNT(s.quantity_sold, p.prod_list_price) COUNT ,
REGR_AVGX(s.quantity_sold, p.prod_list_price) AVGLISTP ,
REGR_AVGY(s.quantity_sold, p.prod_list_price) AVQSOLD
FROM sales s, products p
WHERE s.prod_id=p.prod_id AND
p.prod_category='Men' AND
s.time_id=to_DATE('10-OCT-2000')
GROUP BY s.channel_id
;
```

C	SLOPE	INTCPT	RSQR	COUNT	AVGLISTP	AVQSOLD
C	-.03529838	16.4548382	.217277422	17	87.8764706	13.3529412
I	-.0108044	13.3082392	.028398018	43	116.77907	12.0465116
P	-.01729665	11.3634927	.026191191	33	80.5818182	9.96969697
S	-.01277499	13.488506	.000473089	71	52.571831	12.8169014
T	-.01026734	5.01019929	.064283727	21	75.2	4.23809524

REGR_SLOPE および REGR_INTERCEPT の例

次の例では、各会計年度の売上高および売上利益について、リグレッション直線の傾きおよび切片を計算します。

```
SELECT t.fiscal_year,
       REGR_SLOPE(s.amount_sold, s.quantity_sold) "Slope",
       REGR_INTERCEPT(s.amount_sold, s.quantity_sold) "Intercept"
FROM sales s, times t
WHERE s.time_id = t.time_id
GROUP BY t.fiscal_year;
```

FISCAL_YEAR	Slope	Intercept
1998	54.7377214	45.3884971
1999	54.4868592	44.3616117
2000	55.4035957	44.717026

次の例では、1998 年の四半期の売上高および売上利益について、リグレーション直線の累積の傾きおよび累積の切片を計算します。

```
SELECT t.fiscal_month_number "Month", t.day_number_in_month "Day",
       REGR_SLOPE(s.amount_sold, s.quantity_sold)
         OVER (ORDER BY t.fiscal_month_desc, t.day_number_in_month) AS CUM_SLOPE,
       REGR_INTERCEPT(s.amount_sold, s.quantity_sold)
         OVER (ORDER BY t.fiscal_month_desc, t.day_number_in_month) AS CUM_ICPT
FROM sales s, times t
WHERE s.time_id = t.time_id AND t.fiscal_year=1998
      AND t.fiscal_quarter_number = 4
ORDER BY t.fiscal_month_desc, t.day_number_in_month;
```

Month	Day	CUM_SLOPE	CUM_ICPT

.			
.			
.			
11	18	47.775583	40.028992
11	18	47.775583	40.028992
11	18	47.775583	40.028992
11	18	47.775583	40.028992
11	18	47.775583	40.028992
11	18	47.775583	40.028992
11	18	47.775583	40.028992
11	18	47.775583	40.028992
11	19	47.6878438	40.6296492
11	19	47.6878438	40.6296492
.			
.			
.			

REGR_COUNT の例

次の例では、customers 表のアカウント・マネージャを持つ顧客の数（合計は 319）を戻します。

```
SELECT REGR_COUNT(customer_id, account_mgr_id) FROM customers;

REGR_COUNT(CUSTOMER_ID,ACCOUNT_MGR_ID)
-----
```


次の例では、1998 年 4 月のトランザクションの累積数を 1 日ごとに計算します。

```
SELECT UNIQUE t.day_number_in_month,
  REGR_COUNT(s.amount_sold, s.quantity_sold)
    OVER (PARTITION BY t.fiscal_month_number
      ORDER BY t.day_number_in_month) "Regr_Count"
FROM sales s, times t
WHERE s.time_id = t.time_id
AND t.fiscal_year = 1998 AND t.fiscal_month_number = 4;
```

DAY_NUMBER_IN_MONTH	Regr_Count
1	825
2	1650
3	2475
4	3300
.	
.	
.	
26	21450
30	22200

REGR_R2 の例

次の例では、5,000 より多い売上高および売上利益に対する、リグレーション直線の確定係数を計算します。

```
SELECT REGR_R2(amount_sold, quantity_sold) FROM sales
  WHERE amount_sold > 5000;
```

```
REGR_R2 (AMOUNT_SOLD, QUANTITY_SOLD)
-----
.005208421
```

次の例では、1998 年の各月の 1 か月の売上高および利益について、リグレーション直線の累積確定係数を計算します。

```
SELECT t.fiscal_month_number,
       REGR_R2(SUM(s.amount_sold), SUM(s.quantity_sold))
         OVER (ORDER BY t.fiscal_month_number) "Regr_R2"
FROM sales s, times t
WHERE s.time_id = t.time_id
AND t.fiscal_year = 1998
GROUP BY t.fiscal_month_number
ORDER BY t.fiscal_month_number;
```

FISCAL_MONTH_NUMBER	Regr_R2
1	
2	1
3	.763816809
4	.581171805
5	.854723188
6	.877870333
7	.907073344
8	.905223336
9	.912142295
10	.858149007
11	.74838262
12	.738707443

REGR_AVGY および REGR_AVGX の例

次の例では、各年度の売上高および売上利益について、リグレーション平均を計算します。

```
SELECT t.fiscal_year,
       REGR_AVGY(s.amount_sold, s.quantity_sold) "Regr_AvgY",
       REGR_AVGX(s.amount_sold, s.quantity_sold) "Regr_AvgX"
FROM sales s, times t
WHERE s.time_id = t.time_id
GROUP BY t.fiscal_year;
```

FISCAL_YEAR	Regr_AvgY	Regr_AvgX
1998	745.788191	12.7955581
1999	741.839772	12.8008509
2000	752.701384	12.7786717

次の例では、1998 年 12 月の売上高および売上利益の累積平均を計算します。

```
SELECT t.day_number_in_month,
       REGR_AVGY(s.amount_sold, s.quantity_sold)
         OVER (ORDER BY t.fiscal_month_desc, t.day_number_in_month)
         "Regr_AvgY",
       REGR_AVGX(s.amount_sold, s.quantity_sold)
         OVER (ORDER BY t.fiscal_month_desc, t.day_number_in_month)
         "Regr_AvgX"
FROM sales s, times t
WHERE s.time_id = t.time_id AND t.fiscal_month_desc = '1998-12'
ORDER BY t.day_number_in_month;
```

DAY_NUMBER_IN_MONTH	Regr_AvgY	Regr_AvgX
1	695.028571	12.9
1	695.028571	12.9
1	695.028571	12.9
.		
.		
.		
27	692.061411	12.9648677
27	692.061411	12.9648677
27	692.061411	12.9648677

REGR_SXY、REGR_SXX および REGR_SYY の例

次の例では、サンプル表 sh.sales の各年度の売上高および売上利益のリグレッション解析について、REGR_SXY、REGR_SXX および REGR_SYY の値を計算します。

```
SELECT t.fiscal_year,
       REGR_SXY(s.amount_sold, s.quantity_sold) "Regr_sxy",
       REGR_SYY(s.amount_sold, s.quantity_sold) "Regr_syy",
       REGR_SXX(s.amount_sold, s.quantity_sold) "Regr_sxx"
FROM sales s, times t
WHERE s.time_id = t.time_id
GROUP BY t.fiscal_year;
```

FISCAL_YEAR	Regr_sxy	Regr_syy	Regr_sxx
1998	1757092061	2.5677E+11	32100204.7
1999	2112447869	3.0619E+11	38769859.3
2000	2338925878	3.4321E+11	42216138.7

次の例では、1998 年の各年および月の値の売上および売上利益に対する、REGR_SXY、REGR_SXX および REGR_SYY の累積統計を計算します。

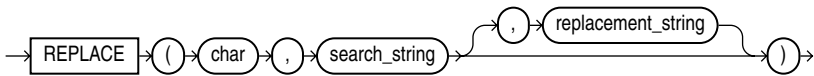
```
SELECT t.day_number_in_month,
       REGR_SXY(s.amount_sold, s.quantity_sold)
         OVER (ORDER BY t.fiscal_year, t.fiscal_month_desc) "Regr_sxy",
       REGR_SYY(s.amount_sold, s.quantity_sold)
         OVER (ORDER BY t.fiscal_year, t.fiscal_month_desc) "Regr_syy",
       REGR_SXX(s.amount_sold, s.quantity_sold)
         OVER (ORDER BY t.fiscal_year, t.fiscal_month_desc) "Regr_sxx"
FROM sales s, times t
WHERE s.time_id = t.time_id AND t.fiscal_month_desc = '1998-02'
ORDER BY t.day_number_in_month;
```

DAY_NUMBER_IN_MONTH	Regr_sxy	Regr_syy	Regr_sxx
1	144226271	2.1996E+10	2497704.77
.			
.			
30	144226271	2.1996E+10	2497704.77

REPLACE

構文

replace::=



用途

REPLACE は、*replacement_string* ですべての *search_string* を変換して *char* を戻します。*replacement_string* を指定しない場合または NULL の場合、すべての *search_string* が削除されます。*search_string* が NULL の場合、*char* が戻されます。

char と同様に、*search_string* および *replacement_string* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻される文字列は、VARCHAR2 データ型であり、*char* と同じキャラクタ・セットです。

このファンクションは、TRANSLATE ファンクションを拡張したものです。TRANSLATE ファンクションは、単一の文字を 1 対 1 で置き換えます。REPLACE ファンクションでは、文字列の置換または削除を実行できます。

参照： 6-175 ページの「[TRANSLATE](#)」を参照してください。

例

次の例では、「J」を「BL」に置換します。

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

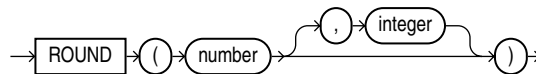
Changes

BLACK and BLUE

ROUND (数値)

構文

round_number::=



用途

ROUND は、*number* を小数点以下 *integer* 桁に丸めた値を戻します。*integer* が省略されると、*number* は小数点以下が丸められます。*integer* が負の場合は小数点の左桁が丸められます。*integer* は整数である必要があります。

例

次の例では、数値を小数点以下 1 桁に丸めます。

```
SELECT ROUND(15.193,1) "Round" FROM DUAL;
```

Round

15.2

次の例では、数値の小数点の左 1 桁を丸めます。

```
SELECT ROUND(15.193,-1) "Round" FROM DUAL;
```

Round

20

ROUND (日付)

構文

round_date::=



用途

ROUND は、*date* を書式モデル *fmt* で指定した単位に丸めた結果を返します。 *fmt* を省略すると、*date* は最も近い日に丸められます。

参照： *fmt* で使用できる書式モデルについては、6-195 ページの「[ROUND および TRUNC 日付ファンクション](#)」を参照してください。

例

次の例では、次の年の最も近い日に丸めます。

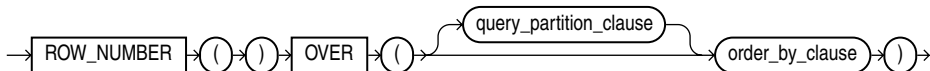
```
SELECT ROUND (TO_DATE ('27-OCT-00'),'YEAR')  
       "New Year" FROM DUAL;
```

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

ROW_NUMBER

構文

row_number::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

ROW_NUMBER は分析ファンクションです。このファンクションは、*order by clause* に指定された行の、1 から始まる順序シーケンスで、このファンクションが適用される各行（パーティションの各行、または問合せが戻す各行）に一意の数値を割り当てます。

expr には、ROW_NUMBER または他の分析ファンクションを使用できません。他の組み込みファンクション式は *expr* で使用できますが、分析ファンクションはネストできません。

参照： *expr* の書式の詳細は、4-2 ページの「SQL 式」を参照してください。

例

次の例では、サンプル表 `oe.employees` 表の各部門について、従業員の雇用日順に数値を各行に割り当てます。

```
SELECT department_id, last_name, employee_id, ROW_NUMBER()
  OVER (PARTITION BY department_id ORDER BY employee_id) AS emp_id
FROM employees;
```

DEPARTMENT_ID	LAST_NAME	EMPLOYEE_ID	EMP_ID
10	Whalen	200	1
20	Hartstein	201	1
20	Goyal	202	2
30	Raphaely	114	1
30	Khoo	115	2
30	Baida	116	3
30	Tobias	117	4
30	Himuro	118	5
30	Colmenares	119	6
40	Marvis	203	1
.			
.			
.			
100	Popp	113	6
110	Higgins	205	1
110	Gietz	206	2

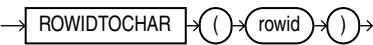
ROW_NUMBER は非決定的なファンクションです。ただし、*employee_id* は一意キーであるため、この場合のファンクションの結果は決定的になります。

参照： 非決定的な動作の例は、6-60 ページの「FIRST_VALUE」および 6-76 ページの「LAST_VALUE」を参照してください。

ROWIDTOCHAR

構文

rowidtochar::=



用途

ROWIDTOCHAR は、ROWID の値を VARCHAR2 データ型に変換します。この変換の結果は常に 18 文字です。

例

次の例では、employees 表の ROWID 値を文字値に変換します。

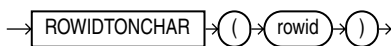
```
SELECT ROWID FROM employees
WHERE ROWIDTOCHAR (ROWID) LIKE '%JAAB%';
```

```
ROWID
-----
AAABIQAAEAAAAEJAAB
```

ROWIDTONCHAR

構文

rowidtonchar::=



用途

ROWIDTONCHAR ファンクションは、ROWID 値を NVARCHAR2 データ型に変換します。この変換の結果は常に 18 文字です。

例

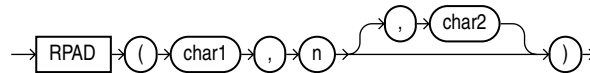
```
SELECT LENGTHB ( ROWIDTONCHAR (ROWID) ), ROWIDTONCHAR (ROWID) FROM emp;
```

```
LENGTHB (ROWIDTONCHAR (ROWID) ) ROWIDTONCHAR (ROWID)
-----
36 AAHymAABAAASuWAAA
```


RPAD

構文

rpad::=



用途

RPAD は、*char1* の右に *char2* で指定した文字を必要に応じて連続的に埋め込み、長さ *n* にして戻します。*char2* のデフォルト値は 1 個の空白です。*char1* が *n* より長い場合、このファンクションは *n* に収まる *char1* の一部を戻します。

char1 および *char2* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻される文字列は、VARCHAR2 データ型であり、*char1* と同じキャラクタ・セットです。

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、表示される文字列の長さが文字列の文字数と異なる場合があります。

例

次の例では、12 文字になるまで、名前の右側に文字「ab」を埋め込みます。

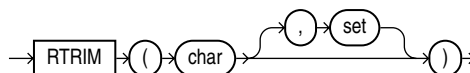
```
SELECT RPAD('MORRISON',12,'ab') "RPAD example"
      FROM DUAL;
```

```
RPAD example
-----
MORRISONabab
```

RTRIM

構文

rtrim::=



用途

RTRIM は、*char* の右側にあって *set* に指定されたすべての文字を削除し、*char* を戻します。*set* のデフォルト値は 1 個の空白です。*char* が文字リテラルの場合、引用符で囲む必要があります。RTRIM は LTRIM と同様の働きをします。

char および *set* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻される文字列は、VARCHAR2 データ型であり、*char* と同じキャラクタ・セットです。

例

次の例では、文字列の右側から文字「xy」を削除します。

```
SELECT RTRIM('BROWNINGxyXxy', 'xy') "RTRIM e.g."
      FROM DUAL;
```

```
RTRIM e.g.
-----
BROWNINGyX
```

参照： 6-84 ページの「[LTRIM](#)」を参照してください。

SESSIONTIMEZONE

構文

sessiontimezone::=



用途

SESSIONTIMEZONE ファンクションは、現行のセッションのタイム・ゾーンの値を戻します。戻り型は、タイム・ゾーン・オフセット（「[+|]TZh:TzM」書式の文字型）またはタイム・ゾーン地域名です。これは、最近の ALTER SESSION 文でユーザーが指定したセッション・タイム・ゾーンの値によって異なります。

例

次の例では、現行のセッションのタイム・ゾーンを戻します。

```
SELECT SESSIONTIMEZONE FROM DUAL;
```

```
SESSION
-----
-08:00
```

SIGN

構文

sign::=



用途

n が 0（ゼロ）より小さい場合、SIGN は -1 を返します。 n が 0（ゼロ）の場合は 0（ゼロ）を返します。 n が 0（ゼロ）より大きい場合、SIGN は 1 を返します。

例

次の例では、ファンクションの引数（-15）が 0 より小さいことを示します。

```
SELECT SIGN(-15) "Sign" FROM DUAL;
```

```

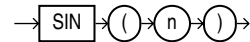
      Sign
-----
      -1

```

SIN

構文

sin::=



用途

SIN は、 n （ラジアンで表された角度）のサインを返します。

例

次の例では、30 度のサインを返します。

```
SELECT SIN(30 * 3.14159265359/180)
       "Sine of 30 degrees" FROM DUAL;
```

```

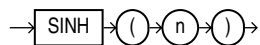
Sine of 30 degrees
-----
                .5

```

SINH

構文

sinh::=



用途

SINH は、*n* の双曲線サインを戻します。

例

次の例では、1 の双曲線サインを戻します。

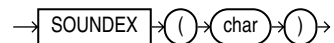
```
SELECT SINH(1) "Hyperbolic sine of 1" FROM DUAL;
```

```
Hyperbolic sine of 1
-----
                1.17520119
```

SOUNDEX

構文

soundex::=



用途

SOUNDEX は、*char* と同じ音声表現を持つ文字列を戻します。このファンクションによって、綴りが異っても発音が似ている英単語を比較できます。

音声表現については、『*The Art of Computer Programming, Volume 3: Sorting and Searching*』(Donald E. Knuth 著) で次のように定義されています。

- 文字列の最初の文字を残し、a、e、h、i、o、u、w、y の文字が出てきた場合にはすべて削除します。

- 残った 2 文字目以降の文字に対し、次のように数値を割り当てます。
b、f、p、v = 1
c、g、j、k、q、s、x、z = 2
d、t = 3
l = 4
m、n = 5
r = 6
- 元の名前（1 つ目の定義を行う前）で、同じ数値を持つ 2 つ以上の文字が並んでいるか、または **h** と **w** の間の文字以外と並んでいる場合は、最初の文字以外のすべての文字を削除します。
- 最初の 4 バイトを 0（ゼロ）で埋めて戻します。

char は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型です。戻り値は、*char* と同じデータ型です。

注意： このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。詳細は、2-42 ページの「[データ型の比較規則](#)」を参照してください。

例

次の例では、「Smyth」と同じ音声表現を持つ従業員を戻します。

```
SELECT last_name, first_name
       FROM hr.employees
       WHERE SOUNDEX(last_name)
             = SOUNDEX('SMYTHE');
```

```
LAST_NAME  FIRST_NAME
-----
Smith      Lindsey
Smith      William
```

SQRT

構文

sqrt::=



用途

SQRT は、*n* の平方根を戻します。値 *n* は負の値にはできません。SQRT は実数を戻します。

例

次の例では、26 の平方根を戻します。

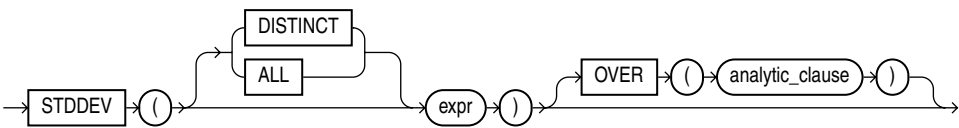
```
SELECT SQRT(26) "Square root" FROM DUAL;
```

```
Square root
-----
5.09901951
```

STDDEV

構文

stddev::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

STDDEV は、数値の集合である *expr* の標本標準偏差を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。このファンクションは、STDDEV_SAMP が NULL を戻すことに対して、入力データが 1 行のみの場合に STDDEV が 0 (ゼロ) を戻すという点で、STDDEV_SAMP と異なります。

Oracle は、VARIANCE 集計ファンクションに対して定義された分散の平方根として標準偏差を計算します。

DISTINCT を指定する場合は、*analytic_clause* の *query_partition_clause* のみ指定できます。*order_by_clause* および *windowing_clause* は指定できません。

参照：

- 詳細は、6-6 ページの「集計ファンクション」、6-191 ページの「VARIANCE」および 6-139 ページの「STDDEV_SAMP」を参照してください。
- *expr* の書式の詳細は、4-2 ページの「SQL 式」を参照してください。

集計の例

次の例では、サンプル表 `hr.employees` の給与値の標本標準偏差を戻します。

```
SELECT STDDEV(salary) "Deviation"
FROM employees;
```

```
Deviation
-----
3909.36575
```

分析の例

次の例では、*hire_date* で順序付けられたサンプル表 `hr.employees` の部門 80 の給与の値の累積標準偏差を戻します。

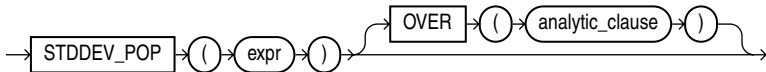
```
SELECT last_name, salary,
       STDDEV(salary) OVER (ORDER BY hire_date) "StdDev"
FROM employees
WHERE department_id = 30;
```

LAST_NAME	SALARY	StdDev
Raphaely	11000	0
Khoo	3100	5586.14357
Tobias	2800	4650.0896
Baida	2900	4035.26125
Himuro	2600	3649.2465
Colmenares	2500	3362.58829

STDDEV_POP

構文

stddev_pop::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

STDDEV_POP は母集団標準偏差を計算し、母集団分散の平方根を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

`expr` は数値式であり、このファンクションは `NUMBER` 型の値を戻します。このファンクションは、`VAR_POP` ファンクションの平方根と同じです。`VAR_POP` が `NULL` を戻す場合、このファンクションも `NULL` を戻します。

参照：

- 6-6 ページの「[集計ファンクション](#)」および 6-188 ページの「[VAR_POP](#)」を参照してください。
- `expr` の書式の詳細は、4-2 ページの「[SQL 式](#)」を参照してください。

集計の例

次の例では、サンプル表 `sh.sales` にある売上高の母集団標準偏差および標本標準偏差を戻します。

```
SELECT STDDEV_POP(amount_sold) "Pop",
       STDDEV_SAMP(amount_sold) "Samp"
FROM sales;
```

Pop	Samp
944.290101	944.290566

分析の例

次の例では、サンプル表 `hr.employees` の部門ごとの給与の母集団標準偏差を戻します。

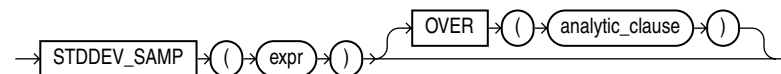
```
SELECT department_id, last_name, salary,
       STDDEV_POP(salary) OVER (PARTITION BY department_id) AS pop_std
FROM employees;
```

DEPARTMENT_ID	LAST_NAME	SALARY	POP_STD
10	Whalen	4400	0
20	Hartstein	13000	3500
20	Goyal	6000	3500
.	.	.	.
100	Sciarra	7700	1644.18166
100	Urman	7800	1644.18166
100	Popp	6900	1644.18166
110	Higgins	12000	1850
110	Gietz	8300	1850

STDDEV_SAMP

構文

stddev_samp::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

STDDEV_SAMP は標本累積標準偏差を計算し、標本分散の平方根を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

`expr` は数値式であり、このファンクションは **NUMBER** 型の値を戻します。このファンクションは、`VAR_SAMP` ファンクションの平方根と同じです。`VAR_SAMP` が **NULL** を戻す場合、このファンクションも **NULL** を戻します。

参照：

- 6-6 ページの「集計ファンクション」および 6-189 ページの「VAR_SAMP」を参照してください。
- expr の書式の詳細は、4-2 ページの「SQL 式」を参照してください。

集計の例

次の例では、サンプル表 sh.sales にある売上高の母集団標準偏差および標本標準偏差を戻します。

```
SELECT STDDEV_POP(amount_sold) "Pop",
       STDDEV_SAMP(amount_sold) "Samp"
FROM sales;
```

Pop	Samp
944.290101	944.290566

分析の例

次の例では、employees 表の部門ごとの給与の標本標準偏差を戻します。

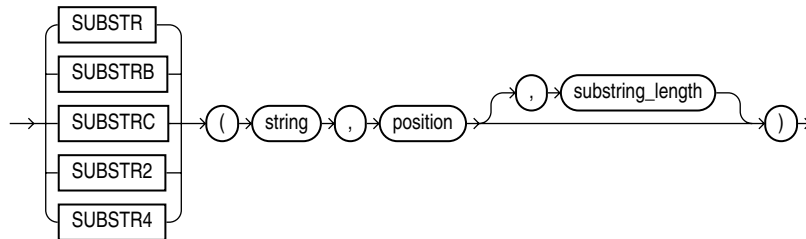
```
SELECT department_id, last_name, hire_date, salary,
       STDDEV_SAMP(salary) OVER (PARTITION BY department_id
                                ORDER BY hire_date
                                ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS cum_sdev
FROM employees;
```

DEPARTMENT_ID	LAST_NAME	HIRE_DATE	SALARY	CUM_SDEV
10	Whalen	17-SEP-87	4400	
20	Hartstein	17-FEB-96	13000	
20	Goyal	17-AUG-97	6000	4949.74747
30	Raphaely	07-DEC-94	11000	
30	Khoo	18-MAY-95	3100	5586.14357
30	Tobias	24-JUL-97	2800	4650.0896
30	Baida	24-DEC-97	2900	4035.26125
.				
.				
.				
100	Chen	28-SEP-97	8200	2003.33056
100	Sciarra	30-SEP-97	7700	1925.91969
100	Urman	07-MAR-98	7800	1785.49713
100	Popp	07-DEC-99	6900	1801.11077
110	Higgins	07-JUN-94	12000	
110	Gietz	07-JUN-94	8300	2616.29509

SUBSTR

構文

substr::=



用途

SUBSTR ファンクションは、*string* の *position* 番目の文字から *substring_length* 文字分の文字列を抜き出して戻します。SUBSTR は、入力キャラクタ・セットによって定義された文字を使用して、長さを計算します。SUBSTRB は、文字のかわりにバイトを使用します。SUBSTRC は、完全な Unicode キャラクタを使用します。SUBSTR2 は、UCS2 コードポイントを使用します。SUBSTR4 は、UCS4 コードポイントを使用します。

- *position* が 0 の場合、1 として処理されます。
- *position* が正の数の場合、Oracle は *string* の始めから数えて最初の文字を検索します。
- *position* が負の場合、Oracle は *string* の終わりにから逆方向にカウントします。
- *substring_length* を指定しないと、Oracle は *string* の終わりまでのすべての文字を戻します。*substring_length* が 1 より小さい場合、NULL を戻します。

string は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻り値は、*string* と同じデータ型です。引数として SUBSTR に渡された浮動小数点数は、自動的に整数に変換されます。

例

次の例では、「ABCDEFGH」の指定されたサブストリングを戻します。

```
SELECT SUBSTR('ABCDEFGH',3,4) "Substring"
       FROM DUAL;
```

```
Substring
-----
CDEF
```

```
SELECT SUBSTR('ABCDEFG',-5,4) "Substring"
FROM DUAL;
```

Substring

CDEF

データベース・キャラクタ・セットがダブルバイトの場合を想定します。

```
SELECT SUBSTRB('ABCDEFG',5,4.2) "Substring with bytes"
FROM DUAL;
```

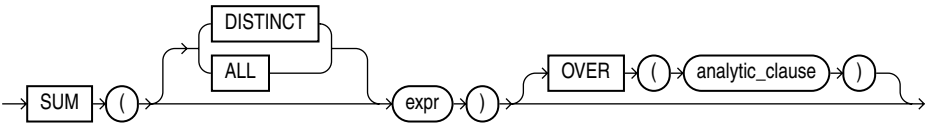
Substring with bytes

CD

SUM

構文

sum::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

SUM は、*expr* の値の合計を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

DISTINCT を指定する場合は、*analytic_clause* の *query_partition_clause* のみ指定できます。*order_by_clause* および *windowing_clause* は指定できません。

参照：

- 6-6 ページの「[集計ファンクション](#)」を参照してください。
- *expr* の書式の詳細は、4-2 ページの「[SQL 式](#)」を参照してください。

集計の例

次の例では、サンプル表 hr.employees にあるすべての給与の合計を計算します。

```
SELECT SUM(salary) "Total"
      FROM employees;
```

Total

691400

分析の例

次の例では、サンプル表 hr.employees の各マネージャについて、そのマネージャの下で働く従業員の現在の給与以下の給与の累積合計を計算します。Raphaely および Zlotkey が同じ累積を持っています。これは、Raphaely および Cambrault が同じ給与であるため、Oracle が給与の値を同時に追加し、同じ累積合計を両方の行に対して適用したためです。

```
SELECT manager_id, last_name, salary,
      SUM(salary) OVER (PARTITION BY manager_id ORDER BY salary
      RANGE UNBOUNDED PRECEDING) l_csum
      FROM employees;
```

MANAGER_ID	LAST_NAME	SALARY	L_CSUM
-----	-----	-----	-----
100	Mourgos	5800	5800
100	Vollman	6500	12300
100	Kaufling	7900	20200
100	Weiss	8000	28200
100	Fripp	8200	36400
100	Zlotkey	10500	46900
100	Raphaely	11000	68900
100	Cambrault	11000	68900
100	Errazuriz	12000	80900
.			
.			
.			
149	Taylor	8600	30200
149	Hutton	8800	39000
149	Abel	11000	50000
201	Goyal	6000	6000
205	Gietz	8300	8300
	King	24000	24000

SYS_CONNECT_BY_PATH

構文

sys_connect_by_path::=

```
→ [SYS_CONNECT_BY_PATH] ( ( column , char ) ) →
```

用途

SYS_CONNECT_BY_PATH は、階層問合せのみで有効です。このファンクションは、ルートからノードへの列の値のパスを、CONNECT BY 条件によって戻された各行を *char* で区切った列の値とともに戻します。

column および *char* は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型です。戻される文字列は、VARCHAR2 データ型であり、*column* と同じキャラクタ・セットです。

参照： 階層問合せおよび CONNECT BY 条件の詳細は、7-3 ページの「[階層問合せ](#)」を参照してください。

例

次の例では、従業員 Kochhar から Kochhar のすべての従業員（およびそれらの従業員の従業員）への従業員名のパスを戻します。

```
SELECT LPAD(' ', 2*level-1) || SYS_CONNECT_BY_PATH(last_name, '/') "Path"
FROM employees
START WITH last_name = 'Kochhar'
CONNECT BY PRIOR employee_id = manager_id;
```

Path

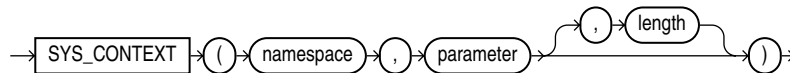
```
-----
/Kochhar
/Kochhar/Greenberg
/Kochhar/Greenberg/Faviet
/Kochhar/Greenberg/Chen
/Kochhar/Greenberg/Sciarra
/Kochhar/Greenberg/Urman
/Kochhar/Greenberg/Popp
/Kochhar/Whalen
/Kochhar/Marvis
/Kochhar/Baer
/Kochhar/Higgins
/Kochhar/Higgins/Gietz
```

12 rows selected.

SYS_CONTEXT

構文

sys_context::=



用途

SYS_CONTEXT は、コンテキスト *namespace* に対応付けられた *parameter* の値を返します。この関数は、SQL 文および PL/SQL 文で使用できます。

注意： **SYS_CONTEXT** は、セッションの属性を返します。そのため、パラレル問合せまたは Real Application Clusters 環境では使用できません。

namespace および *parameter* には、文字列（定数）、あるいはネームスペースまたは属性を指定する文字列に変換する式を指定できます。コンテキスト *namespace* はすでに作成されている必要があり、対応付けられた *parameter* およびその値は DBMS_SESSION.set_context プロシージャを使用して設定されている必要があります。*namespace* は有効な SQL 識別子である必要があります。*parameter* 名には、すべての文字列を指定できます。大 / 小文字を識別しますが、長さは 30 バイト以下です。

戻り値のデータ型は VARCHAR2 です。戻り値のデフォルトの最大サイズは、256 バイトです。オプションの *length* パラメータを指定して、このデフォルトをオーバーライドできます。有効な値の範囲は、1 ～ 4000 バイトです（無効な値を指定すると、Oracle はその値を無視してデフォルト値を使用します）。

Oracle9i では、現行のセッションを記述する USERENV という組込みネームスペースを提供しています。ネームスペース USERENV の事前定義パラメータおよびその戻り文字列の長さについては、6-146 ページの表 6-10 を参照してください。

参照：

- アプリケーション開発でのアプリケーション・コンテキスト機能の使用方法については、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。
- ユーザー定義のコンテキスト・ネームスペースの作成方法については、12-11 ページの「CREATE CONTEXT」を参照してください。
- DBMS_SESSION.set_context プロシージャについては、『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

例

次の例では、データベースにログインしたユーザー名を戻します。

```
CONNECT OE/OE
SELECT SYS_CONTEXT ('USERENV', 'SESSION_USER')
       FROM DUAL;

SYS_CONTEXT ('USERENV', 'SESSION_USER')
-----
OE
```

次の例では、hr_apps の作成時に、コンテキスト hr_apps に対応付けられた PL/SQL パッケージの属性 group_no に対する値として設定されたグループ番号を戻します。

```
SELECT SYS_CONTEXT ('hr_apps', 'group_no') "User Group"
       FROM DUAL;
```

表 6-10 ネームスペース USERENV の事前定義パラメータ

パラメータ	戻り値	戻り値の長さ (バイト)
AUDITED_CURSORID	監査をトリガーによって実行した SQL のカーソル ID を戻します。	NA
AUTHENTICATION_DATA	ログイン・ユーザーの認証に使用されるデータを戻します。X.503 認証セッションでは、このフィールドは HEX2 形式での認証のコンテキストを戻します。 注意： 構文の length パラメータを使用して、AUTHENTICATION_DATA 属性の戻り値を変更できます。最大 4000 までの値を指定できます。この属性は、Oracle がこのような変更を実行する USERENV の唯一の属性です。	256
AUTHENTICATION_TYPE	次に示すとおり、ユーザーの認証方法を戻します。 <ul style="list-style-type: none">■ DATABASE: ユーザー名 / パスワード認証■ OS: オペレーティング・システムの外部ユーザー認証■ NETWORK: ネットワーク・プロトコルまたは ANO 認証■ PROXY: OCI プロキシ接続認証	30

表 6-10 ネームスペース USERENV の事前定義パラメータ (続き)

パラメータ	戻り値	戻り値の長さ (バイト)
BG_JOB_ID	現行のセッションが Oracle のバックグラウンド・プロセスで確立された場合、そのセッションのジョブ ID を戻します。セッションがバックグラウンド・プロセスで確立されていない場合は、NULL を戻します。	30
CLIENT_IDENTIFIER	グローバル・コンテキスト (グローバルにアクセスされたアプリケーション・コンテキスト、または OCI コンテキストの OCI_ATTR_CLIENT_IDENTIFIER 属性) のクライアントのセッション識別子を戻します。関連する識別子がグローバルに設定されていない場合は、NULL を戻します。	64
CLIENT_INFO	DBMS_APPLICATION_INFO パッケージを使用するアプリケーションが格納できる 64 バイトまでのユーザー・セッション情報を戻します。	64
CURRENT_SCHEMA	カレント・スキーマで使用されているデフォルトのスキーマ名を戻します。この値は、セッション中に ALTER SESSION SET CURRENT_SCHEMA 文を使用して変更できます。	30
CURRENT_SCHEMAID	現行のセッションで使用されているデフォルトのスキーマ ID を戻します。	30
CURRENT_SQL	ファイングレイン監査イベントをトリガーによって実行した現行の SQL を戻します。この属性は、ファイングレイン監査機能のイベント・ハンドラ内のみで指定できます。	64
CURRENT_USER	現行のセッションで権限のあるユーザーのユーザー名を戻します。	30
CURRENT_USERID	現行のセッションで権限のあるユーザーのユーザー ID を戻します。	30
DB_DOMAIN	DB_DOMAIN 初期化パラメータで指定されたデータベースのドメインを戻します。	256
DB_NAME	DB_NAME 初期化パラメータで指定されたデータベース名を戻します。	30

表 6-10 ネームスペース USERENV の事前定義パラメータ (続き)

パラメータ	戻り値	戻り値の長さ (バイト)
ENTRYID	使用可能な監査エントリ識別子を返します。この属性を分散 SQL 文で使用することはできません。USERENV でこのキーワードを使用するには、AUDIT_TRAIL 初期化パラメータに true を設定する必要があります。	30
EXTERNAL_NAME	データベース・ユーザーの外部名を返します。v.503 認証を使用する SSL 認証セッションでは、このフィールドは、ユーザー認証に格納されている識別名 (DN) を返します。	256
FG_JOB_ID	現行のセッションが Oracle のフォアグラウンド・プロセスで確立された場合、そのセッションのジョブ ID を返します。セッションがフォアグラウンド・プロセスで確立されていない場合は、NULL を返します。	30
GLOBAL_CONTEXT_MEMORY	コンテキストへのグローバルなアクセスによって、システム・グローバル領域で使用された数値を返します。	NA
HOST	接続中のクライアントのホスト・マシン名を返します。	54
INSTANCE	現行のインスタンスのインスタンス識別番号を返します。	30
IP_ADDRESS	接続中のクライアントのマシンの IP アドレスを返します。	30
ISDBA	オペレーティング・システムまたはパスワード・ファイルによって、ユーザーが DBA 権限を持っていると認証された場合、TRUE を返します。	30
LANG	言語名の ISO 略称を返します。これは、既存の 'LANGUAGE' パラメータを短縮したものです。	62
LANGUAGE	現行のセッションで使用している言語 (language) および地域 (territory) を、データベース・キャラクタ・セット (character set) も含めて次の書式で返します。 language_territory.characterset	52

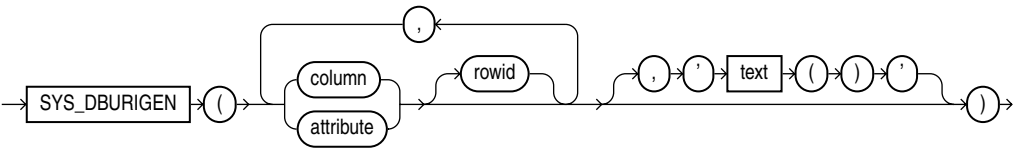
表 6-10 ネームスペース USERENV の事前定義パラメータ (続き)

パラメータ	戻り値	戻り値の長さ (バイト)
NETWORK_PROTOCOL	接続文字列の 'PROTOCOL= <i>protocol</i> ' の部分で指定された、通信に使用されるネットワーク・プロトコルを戻します。	256
NLS_CALENDAR	現行のセッションの現行のカレンダを戻します。	62
NLS_CURRENCY	現行のセッションの通貨を戻します。	62
NLS_DATE_FORMAT	セッションの日付書式を戻します。	62
NLS_DATE_LANGUAGE	日付の表示に使用される言語を戻します。	62
NLS_SORT	BINARY または言語ソート基準を戻します。	62
NLS_TERRITORY	現行のセッションの地域を戻します。	62
OS_USER	データベース・セッションを初期化するクライアント・プロセスのオペレーティング・システム・ユーザー名を戻します。	30
PROXY_USER	SESSION_USER のかわりに現行のセッションをオープンしたデータベース・ユーザー名を戻します。	30
PROXY_USERID	SESSION_USER のかわりに現行のセッションをオープンしたデータベース・ユーザーの ID を戻します。	30
SESSION_USER	現行のユーザーが認証されているデータベース・ユーザー名を戻します。この値は、セッションの存続期間中は同じです。	30
SESSION_USERID	現行のユーザーが認証されているデータベース・ユーザーの ID を戻します。	30
SESSIONID	監査セッション識別子を戻します。この属性を分散 SQL 文で使用することはできません。	30
TERMINAL	現行のセッションのクライアントに対するオペレーティング・システムの識別子を戻します。分散 SQL 文では、この属性はローカル・セッションの識別子を戻します。分散環境では、リモートの SELECT に対してのみこのオプションを使用でき、リモートの INSERT、UPDATE または DELETE には使用できません (このパラメータの戻り値の長さは、オペレーティング・システムによって異なります)。	10

SYS_DBURIGEN

構文

sys_dburigen::=



用途

SYS_DBURIGEN ファンクションでは、引数として 1 つ以上の列または属性、およびオプションの ROWID、特定の列または行オブジェクトへの DBUriType データ型の URL を生成します。これによって、データベースから XML ドキュメントを検索するための URL を使用できるようになります。

参照されるすべての列または属性は、サンプル表に存在する必要があります。これらは、主キーの役割を果たす必要があります。つまり、実際に表の主キーに一致する必要はありませんが、一意の値を参照する必要があります。複数の列を指定すると、最後の列以外のすべての列はデータベースの行を識別し、指定された最後の列は行にある列を識別します。

デフォルトでは、URL はフォーマットされた XML ドキュメントを指します。ドキュメントのテキストのみを指す場合は、オプションの 'text()' を指定します（この XML コンテキストでは、小文字の 'text' はキーワードであり、構文のプレースホルダではありません）。

列または属性を含む表あるいはビューは、問合せのコンテキストで指定されるスキーマを持たない場合、Oracle は、表名またはビュー名をパブリック・シノニムとして解析します。

参照： データベースの UriType データ型および XML ドキュメントの詳細は、『Oracle9i XML リファレンス』および『Oracle9i アプリケーション開発者ガイド - XML』を参照してください。

例

次の例では、SYS_DBURIGEN ファンクションを使用して、サンプル表 hr.employees の employee_id = 206 である行の電子メール列への DBUriType データ型の URL を生成します。

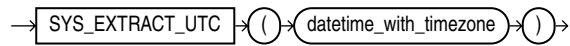
```
SELECT SYS_DBURIGEN(employee_id, email)
       FROM employees
       WHERE employee_id = 206;
```

```
SYS_DBURIGEN(EMPLOYEE_ID,EMAIL) (URL, SPARE)
-----
DBURITYPE('/PUBLIC/EMPLOYEES/ROW[EMPLOYEE_ID = "206"]/EMAIL', NULL)
```

SYS_EXTRACT_UTC

構文

sys_extract_utc::=



用途

SYS_EXTRACT_UTC ファンクションは、タイム・ゾーン置換値を持つ日時から UTC を抽出します。

例

次の例では、指定された日時から UTC を抽出します。

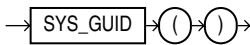
```
SELECT SYS_EXTRACT_UTC(TIMESTAMP '2000-03-28 11:30:00.00 -08:00')
       FROM DUAL;
```

```
SYS_EXTRACT_UTC(TIMESTAMP'2000-03-2811:30:00.00-08:00')
-----
28-MAR-00 07.30.00 PM
```

SYS_GUID

構文

sys_guid::=



用途

SYS_GUID は、16 バイトで構成されたグローバルな一意の識別子（RAW 値）を生成して戻します。多くのプラットフォームでは、生成された識別子は、ホスト識別子とプロセス、またはファンクションを呼び出すプロセスやスレッドのスレッド識別子、およびそのプロセスやスレッドに対する非反復値（バイトの順序）で構成されています。

例

次の例では、サンプル表 hr.locations に列を追加後、一意の識別子を各行に挿入し、グローバルな一意識別子の 16 バイトの RAW 値を 32 文字の 16 進表記で戻します。

```
ALTER TABLE locations ADD (uid_col RAW(32));
```

```
UPDATE locations SET uid_col = SYS_GUID();
```

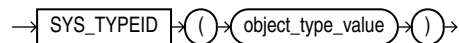
```
SELECT location_id, uid_col FROM locations;
```

LOCATION_ID	UID_COL
1000	7CD5B7769DF75CEFE034080020825436
1100	7CD5B7769DF85CEFE034080020825436
1200	7CD5B7769DF95CEFE034080020825436
1300	7CD5B7769DFA5CEFE034080020825436
.	.
.	.
.	.

SYS_TYPEID

構文

sys_typeid::=



用途

SYS_TYPEID ファンクションは、オペランドで最も指定される型の型 ID を戻します。この値は、主に、置換可能な列の基礎となる型判別式の列を識別するために使用されます。たとえば、型判別式の列に索引を構築するために、SYS_TYPEID によって戻される値を使用できます。

注意：

- このファンクションは、オブジェクト型のオペランドのみで使用してください。
- すべての最終ルート・オブジェクト型（型階層に属さない最終型）は、NULL 型の ID を持ちます。Oracle は、型階層に属するすべての型に、NULL 以外の一意の型 ID を割り当てます。

例

次の例では、14-52 ページの「置換可能な表および列のサンプル」で作成された表 `persons` および `books` を使用します。これらの表は、どちらも 15-20 ページの「型の階層例」で作成された `person_t` 型を使用します。最初の問合せは、`persons` 表に格納されたオブジェクト・インスタンスの最も指定される型を戻します。

```
SELECT name, SYS_TYPEID(VALUE(p)) "Type_id" FROM persons p;
```

NAME	Type_id
-----	-----
Bob	01
Joe	02
Tim	03

次の問合せは、books 表に格納された最も指定された作者の型を戻します。

```
SELECT b.title, b.author.name, SYS_TYPEID(author)
       "Type_ID" FROM books b;
```

TITLE	AUTHOR.NAME	Type_ID
-----	-----	-----
An Autobiography	Bob	01
Business Rules	Joe	02
Mixing School and Work	Tim	03

SYS_TYPEID ファンクションを使用すると、表の型判別式の列に索引を作成できます。
12-82 ページの「置換可能な列の索引の例」の例を参照してください。

SYS_XMLAGG

構文

sys_xmlagg::=



用途

SYS_XMLAGG ファンクションは、*expr* によって表されるすべての XML ドキュメントまたは XML フラグメントを集約し、単一の XML ドキュメントを生成します。このファンクションは、デフォルト名 ROWSET の新しい囲み要素を追加します。XML ドキュメントを別の方法でフォーマットする場合は、SYS.XMLGenFormatType オブジェクトのインスタンスである *fmt* を指定します。

参照：

- SYS_XMLGEN の結果をフォーマットするための XMLGenFormatType 型の属性の使用については、2-76 ページの「XML 書式モデル」を参照してください。
- 6-155 ページの「SYS_XMLGEN」を参照してください。
- XML 型およびそれらの使用については、『Oracle9i XML リファレンス』および『Oracle9i アプリケーション開発者ガイド - XML』を参照してください。

例

次の例では、SYS_XMLGEN ファンクションを使用して、従業員名の最初の文字が R であるサンプル表 employees の各行に対して、XML ドキュメントを生成した後、デフォルトの囲み要素 ROWSET の 1 つの XML ドキュメントにすべての行を集約します。

```
SELECT SYS_XMLAGG(SYS_XMLGEN(last_name)).getClobVal()
       FROM employees
       WHERE last_name LIKE 'R%';
```

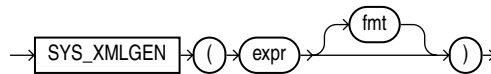
```
SYS_XMLAGG(SYS_XMLGEN(LAST_NAME)).GETCLOBVAL()
```

```
-----
<?xml version="1.0"?>
<ROWSET>
<LAST_NAME>Rajs</LAST_NAME>
<LAST_NAME>Raphaely</LAST_NAME>
<LAST_NAME>Rogers</LAST_NAME>
<LAST_NAME>Russell</LAST_NAME>
</ROWSET>
```

SYS_XMLGEN

構文

sys_xmlgen::=



用途

SYS_XMLGEN ファンクションでは、データベースの特定の行および列を評価する式を指定し、XML ドキュメントを含む SYS.XMLType 型のインスタンスを返します。expr は、スカラー値、ユーザー定義型または XMLType インスタンスです。

- expr がスカラー値である場合、ファンクションはスカラー値を含む XML 要素を返します。
- expr が型である場合、ファンクションは XML 要素へユーザー定義型の属性をマップします。
- expr が XMLType インスタンスである場合、ファンクションはデフォルトのタグ名が ROW である XML 要素でドキュメントを囲みます。

デフォルトでは、XML ドキュメントの要素は `expr` の要素と一致します。たとえば、`expr` が列名に変換される場合、XML の囲み要素は同じ列名になります。XML ドキュメントを別の方法でフォーマットする場合は、`SYS.XMLGenFormatType` オブジェクトのインスタンスである `fmt` を指定します。

参照：

- `XMLGenFormatType` 型の詳細および `SYS_XMLGEN` の結果を書式化するための属性の使用方法については、2-76 ページの「[XML 書式モデル](#)」を参照してください。
- XML 型およびそれらの使用については、『Oracle9i XML リファレンス』および『Oracle9i アプリケーション開発者ガイド - XML』を参照してください。

例

次の例では、サンプル表 `oe.employees` から `employee_id` 値が 205 の従業員の電子メール ID を検出し、EMAIL 要素を持つ XML ドキュメントを含む `XMLType` のインスタンスを生成します。

```
SELECT SYS_XMLGEN(email).getStringVal()
FROM employees
WHERE employee_id = 205;

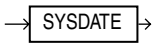
SYS_XMLGEN(EMAIL).GETSTRINGVAL()
-----
<EMAIL>SHIGGENS</EMAIL>
```

注意： `XMLType` データは、`CLOB` データと同様に処理されます。`SQL*Plus` を使用すると、表示されません。そのため、この例では、実際のドキュメントを表示させるために `XMLType.getClobVal` メソッドをファンクションに付けています。たとえば、`XMLType` 値をデータベースに挿入する場合、このメソッドは不要です。

SYSDATE

構文

`sysdate::=`



用途

SYSDATE は現在の日時を戻します。引数は必要ありません。分散 SQL 文では、このファンクションはローカル・データベース上の日時を戻します。チェック制約の条件でこのファンクションは使用できません。

例

次の例では、現在の日付および時刻を戻します。

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

```
NOW
-----
04-13-2001 09:45:51
```

SYSTIMESTAMP

構文

systimestamp::=

→ SYSTIMESTAMP →

用途

SYSTIMESTAMP ファンクションは、データベースが存在するシステムの日付（小数部を含む）を戻します。戻り型は、TIMESTAMP WITH TIME ZONE です。

例

次の例では、システムの日付を戻します。

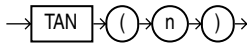
```
SELECT SYSTIMESTAMP FROM DUAL;
```

```
SYSTIMESTAMP
-----
28-MAR-00 12.38.55.538741 PM -08:00
```

TAN

構文

tan::=



用途

TAN は、*n*（ラジアンで表された角度）のタンジェントを戻します。

例

次の例では、135 度のタンジェントを戻します。

```
SELECT TAN(135 * 3.14159265359/180)
      "Tangent of 135 degrees" FROM DUAL;
```

```
Tangent of 135 degrees
-----
                    - 1
```

TANH

構文

tanh::=



用途

TANH は、*n* の双曲線タンジェントを戻します。

例

次の例では、5 の双曲線タンジェントを戻します。

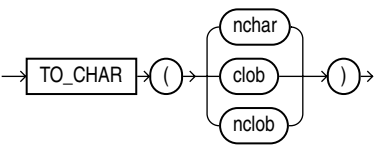
```
SELECT TANH(.5) "Hyperbolic tangent of .5"
      FROM DUAL;
```

```
Hyperbolic tangent of .5
-----
                    .462117157
```

TO_CHAR (文字)

構文

to_char_char::=



用途

TO_CHAR (文字) ファンクションは、NCHAR、NVARCHAR2、CLOB または NCLOB データをデータベース・キャラクタ・セットに変換します。

例

次の例では、pm.print_media 表の CLOB データをデータベース・キャラクタ・セットに変換します。

```
SELECT TO_CHAR(ad_sourcetext) FROM print_media
      WHERE product_id = 2268;
```

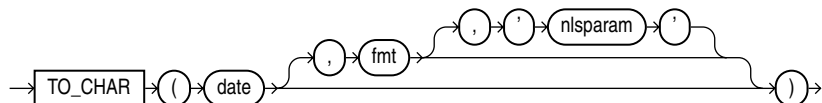
```
TO_CHAR (AD_SOURCETEXT)
-----
*****
TIGER2 2268...Standard Hayes Compatible Modem
Product ID: 2268
The #1 selling modem in the universe! Tiger2's modem includes call management and
Internet voicing. Make real-time full duplex phone calls at the same time you're
online.

*****
```

TO_CHAR (日時)

構文

to_char_date::=



用途

TO_CHAR は、DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONE または TIMESTAMP WITH LOCAL TIME ZONE データ型の *date* を日付書式 *fmt* で指定された書式の VARCHAR2 データ型に変換します。 *fmt* を省略すると、次のように、*date* は VARCHAR2 値に変換されます。

- DATE は、デフォルトの日付書式の値に変換されます。
- TIMESTAMP および TIMESTAMP WITH LOCAL TIME ZONE は、デフォルトのタイムスタンプ書式の値に変換されます。
- TIMESTAMP WITH TIME ZONE は、タイム・ゾーン書式のデフォルトのタイムスタンプの値に変換されます。

'*nlsparam*' には、月と日の名前および略称が戻される言語を指定します。この引数は、次の書式で指定します。

'NLS_DATE_LANGUAGE = language'

nlsparam を指定しないと、このファンクションはセッションのデフォルト日付言語を使用します。

参照： 日付書式については、2-59 ページの「[書式モデル](#)」を参照してください。

例

次の例で使用する表は、次のとおりです。

```
CREATE TABLE my_tab (
  ts_col TIMESTAMP,
  tsltz_col TIMESTAMP WITH LOCAL TIME ZONE,
  tstz_col TIMESTAMP WITH TIME ZONE);
```

次の例では、TO_CHAR を別の TIMESTAMP データ型に適用した結果を示します。
TIMESTAMP WITH LOCAL TIME ZONE 列の結果は、セッションのタイム・ゾーンを識別します。これに対して、TIMESTAMP および TIMESTAMP WITH TIME ZONE 列の結果は、セッションのタイム・ゾーンを識別しません。

```
ALTER SESSION SET TIME_ZONE = '-8:00';
INSERT INTO my_tab VALUES (
    TIMESTAMP'1999-12-01 10:00:00',
    TIMESTAMP'1999-12-01 10:00:00',
    TIMESTAMP'1999-12-01 10:00:00');
INSERT INTO my_tab VALUES (
    TIMESTAMP'1999-12-02 10:00:00 -8:00',
    TIMESTAMP'1999-12-02 10:00:00 -8:00',
    TIMESTAMP'1999-12-02 10:00:00 -8:00');

SELECT TO_CHAR(ts_col, 'DD-MON-YYYY HH24:MI:SSxFF'),
       TO_CHAR(tstz_col, 'DD-MON-YYYY HH24:MI:SSxFF TZH:TZM')
FROM my_tab;

TO_CHAR(TS_COL, 'DD-MON-YYYYHH24:MI:SS') TO_CHAR(TSTZ_COL, 'DD-MON-YYYYHH24:MI:SS')
-----
01-DEC-1999 10:00:00                      01-DEC-1999 10:00:00.000000 -08:00
02-DEC-1999 10:00:00                      02-DEC-1999 10:00:00.000000 -08:00

SELECT SESSIONTIMEZONE,
       TO_CHAR(tsltz_col, 'DD-MON-YYYY HH24:MI:SSxFF')
FROM my_tab;

SESSION TO_CHAR(TSLTZ_COL, 'DD-MON-YYYYHH24:MI:SS')
-----
-08:00  01-DEC-1999 10:00:00
-08:00  02-DEC-1999 10:00:00

ALTER SESSION SET TIME_ZONE = '-5:00';
SELECT TO_CHAR(ts_col, 'DD-MON-YYYY HH24:MI:SSxFF'),
       TO_CHAR(tstz_col, 'DD-MON-YYYY HH24:MI:SSxFF TZH:TZM')
FROM my_tab;

TO_CHAR(TS_COL, 'DD-MON-YYYYHH24:MI:SS') TO_CHAR(TSTZ_COL, 'DD-MON-YYYYHH24:MI:SS')
-----
01-DEC-1999 10:00:00                      01-DEC-1999 10:00:00.000000 -08:00
02-DEC-1999 10:00:00                      02-DEC-1999 10:00:00.000000 -08:00

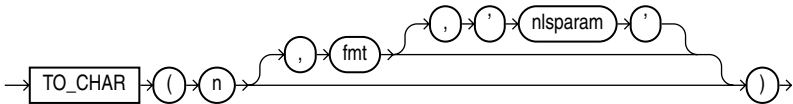
SELECT SESSIONTIMEZONE,
       TO_CHAR(tsltz_col, 'DD-MON-YYYY HH24:MI:SSxFF')
FROM my_tab;
```

```
SESSION TO_CHAR(TSLTZ_COL, 'DD-MON-YYYY')
-----
-05:00 01-DEC-1999 13:00:00
-05:00 02-DEC-1999 13:00:00
```

TO_CHAR（数値）

構文

to_char_number::=



用途

TO_CHAR は、NUMBER データ型の *n* を、オプションの数値書式 *fmt* で指定した書式の 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。

nlsparam には、数値書式要素によって戻される次の文字を指定します。

- 小数点文字
- 桁区切り
- 各国通貨記号
- 国際通貨記号

この引数は、次の書式で指定します。

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

文字 *d* および *g* は、それぞれ小数点文字および桁区切りを表します。これらは、異なるシングルバイト文字である必要があります。引用符付き文字列の中では、パラメータ値を囲む一重引用符を 2 つ使用する必要がある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。

nlsparam またはパラメータのいずれか 1 つを省略すると、このファンクションはセッションのデフォルト・パラメータ値を使用します。

参照： 日付書式については、2-59 ページの「書式モデル」を参照してください。

例

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-
```

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

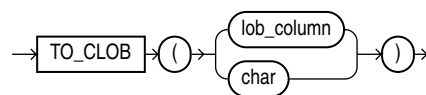
```
Amount
-----
AusDollars10.000,00-
```

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。数値書式要素のすべてのリストは、2-62 ページの表 2-10 を参照してください。

TO_CLOB

構文

to_clob::=



用途

TO_CLOB ファンクションは、LOB 列またはその他の文字列の NCLOB の値を CLOB の値に変換します。char は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。Oracle は、基礎となる LOB データを各国語キャラクタ・セットからデータベース・キャラクタ・セットに変換することによって、このファンクションを実行します。

例

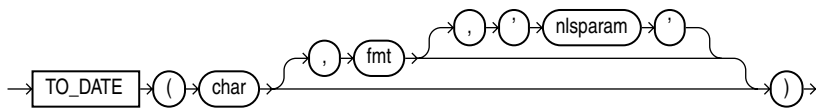
次の文は、サンプル表 `pm.print_media` の NCLOB データを CLOB に変換し、CLOB 列に挿入します。

```
UPDATE PRINT_MEDIA
  SET AD_FINALTEXT = TO_CLOB (AD_FLTEXTN) ;
```

TO_DATE

構文

`to_date::=`



用途

`TO_DATE` は、`CHAR`、`VARCHAR2`、`NCHAR` または `NVARCHAR2` データ型の `char` を、`DATE` データ型の値に変換します。 `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` はデフォルトの日付書式である必要があります。 `fmt` が「J」（ユリウス暦）の場合、`char` は整数である必要があります。

`nlsparam` は、日付変換の `TO_CHAR` ファンクションの場合と同じ用途で使用されます。

引数 `char` に `DATE` 値を持つ `TO_DATE` ファンクションは使用しないでください。 戻される `DATE` 値は、`fmt` またはデフォルトの日付書式によって、元の `char` とは異なる世紀の値を持つことがあります。

注意： このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。詳細は、2-42 ページの「データ型の比較規則」を参照してください。

参照： 2-66 ページの「日付書式モデル」を参照してください。

例

次の例では、文字列を日付に変換します。

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

```
TO_DATE('
-----
15-JAN-89
```

TO_DSINTERVAL

構文

to_dsinterval::=



用途

TO_DSINTERVAL は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型の文字列を、INTERVAL DAY TO SECOND データ型の値に変換します。

- *char* は、変換する文字列です。
- この関数で指定できる有効な NLS パラメータは、NLS_NUMERIC_CHARACTERS のみです。この引数は、次の書式で指定します。

```
NLS_NUMERIC_CHARACTERS = "dg"
```

d および *g* は、それぞれ小数点文字および桁区切りを表します。

例

次の例では、employees 表から 1985 年 1 月 1 日から 100 日以上勤務している従業員を検索します。

```
SELECT employee_id, last_name FROM employees
       WHERE hire_date - TO_DSINTERVAL('100 10:00:00')
          > DATE '1985-01-01';
```

```
EMPLOYEE_ID LAST_NAME
-----
100 King
101 Kochhar
102 De Haan
```

TO_LOB

構文

to_lob::=

→ TO_LOB → ((long_column)) →

用途

TO_LOB は、*long_column* 列の LONG または LONG RAW 値を LOB 値に変換します。このファンクションは LONG または LONG RAW 列に対してのみ、および INSERT 文における副問合せの SELECT 構文のリストにおいてのみ適用できます。

このファンクションを使用する前に、LOB 列を作成して変換された LONG 値を受け取る必要があります。LONG に変換するには、CLOB 列を作成します。LONG RAW に変換するには、BLOB 列を作成します。

参照：

- LONG 列を LOB に変換する別の方法は、10-2 ページの「[ALTER TABLE](#)」の *modify_column_options* 句を参照してください。
- INSERT 文の副問合せについては、16-56 ページの「[INSERT](#)」を参照してください。

例

サンプル表 `pm.print_media` は、LONG 型の列 `press_release` を持ちます。次の例では、`print_media` 列の LOB データを持つ表の一部を再作成します。

```
CREATE TABLE new_print_media (
  product_id      NUMBER(6),
  ad_id           NUMBER(6),
  press_release   CLOB);

INSERT INTO new_print_media
  (SELECT p.product_id, p.ad_id, TO_LOB(p.press_release)
   FROM print_media p);
```

TO_MULTI_BYTE

構文

`to_multi_byte::=`

→ (TO_MULTI_BYTE ((char))) →

用途

`TO_MULTI_BYTE` は、すべてのシングルバイト文字を、対応するマルチバイト文字に変換して `char` を戻します。`char` のデータ型は、`CHAR`、`VARCHAR2`、`NCHAR` または `NVARCHAR2` です。戻り値は、`char` と同じデータ型です。

`char` 内に同等のマルチバイト文字がないシングルバイト文字は、シングルバイト文字として出力されます。このファンクションは、データベース・キャラクタ・セットに、シングルバイト文字およびマルチバイト文字の両方が含まれている場合にのみ有効です。

注意： このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。詳細は、2-42 ページの「[データ型の比較規則](#)」を参照してください。

例

次の例では、シングルバイト 'A' から UTF8 のマルチバイト 'A' への変換を示します。

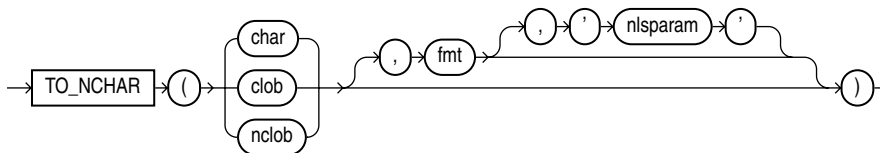
```
SELECT dump(TO_MULTI_BYTE('A')) FROM DUAL;
```

```
DUMP(TO_MULTI_BYTE('A'))
-----
Typ=1 Len=3: 239,188,161
```

TO_NCHAR (文字)

構文

to_nchar_char::=



用途

TO_NCHAR (文字) ファンクションは、文字列、CLOB または NCLOB をデータベース・キャラクタ・セットから各国語キャラクタ・セットに変換します。このファンクションは、各国語キャラクタ・セットで USING 句を指定した TRANSLATE ... USING ファンクションと同じです。

参照： 2-46 ページの「[データ変換](#)」および 6-176 ページの「[TRANSLATE ... USING](#)」を参照してください。

例

次の例では、pm.print_media 表の NCLOB データを各国語キャラクタ・セットに変換します。

```
SELECT TO_NCHAR(ad_fltextn) FROM print_media
       WHERE product_id = 3106;
```

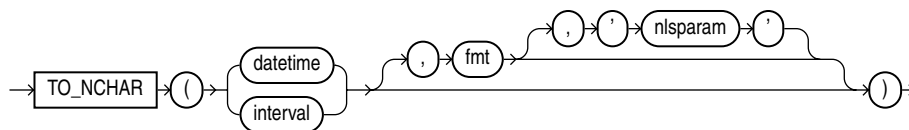
```
TO_NCHAR (AD_FLTEXTN)
```

```
-----
TIGER2 3106 Tastatur
Product Nummer: 3106
Nur 39 EURO!
Die Tastatur KB 101/CH-DE ist eine Standard PC/AT Tastatur mit 102 Tasten. Tasta
turbelegung: Schweizerdeutsch.
・ NEU: Kommt mit ergonomischer Schaumstoffunterlage.
・ Extraflache und ergonomisch-geknickte Versionen verfügbar auf Anfrage.
・ Lieferbar in Elfenbein, Rot oder Schwarz.
```

TO_NCHAR (日時)

構文

to_nchar_date::=



用途

TO_NCHAR (日時) ファンクションは、DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONE、TIMESTAMP WITH LOCAL TIME ZONE、INTERVAL MONTH TO YEAR または INTERVAL DAY TO SECOND データ型の文字列をデータベース・キャラクタ・セットから各国語キャラクタ・セットに変換します。

例

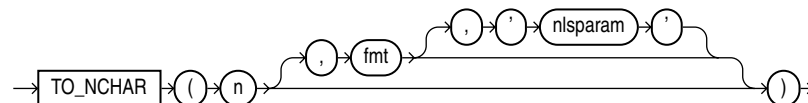
```
SELECT TO_NCHAR(ORDER_DATE) FROM ORDERS
WHERE ORDER_STATUS > 9;
```

```
TO_NCHAR(ORDER_DATE)
-----
14-SEP-99 08.53.40.223345 AM
13-SEP-99 09.19.00.654279 AM
27-JUN-00 08.53.32.335522 PM
26-JUN-00 09.19.43.190089 PM
06-DEC-99 01.22.34.225609 PM
```

TO_NCHAR (数値)

構文

to_nchar_number::=



用途

TO_NCHAR (数値) ファンクションは、数値を NVARCHAR2 キャラクタ・セットの文字列に変換します。オプションの *fmt* および *nlsparam* は、DATE、TIMESTAMP、TIMESTAMP WITH TIME ZONE、TIMESTAMP WITH LOCAL TIME ZONE、INTERVAL MONTH TO YEAR または INTERVAL DAY TO SECOND データ型である *n* に対応しています。

例

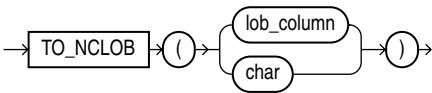
```
SELECT TO_NCHAR(CUSTOMER_ID) "NCHAR_Customer_ID" FROM ORDERS
      WHERE ORDER_STATUS > 9
```

```
NCHAR_Customer_ID
-----
102
103
148
149
148
```

TO_NCLOB

構文

to_nclob::=



用途

TO_NCLOB ファンクションは、LOB 列またはその他の文字列の CLOB の値を NCLOB の値に変換します。*char* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。Oracle は、LOB 列のキャラクタ・セットをデータベース・キャラクタ・セットから各国語キャラクタ・セットに変換することによって、このファンクションを実行します。

例

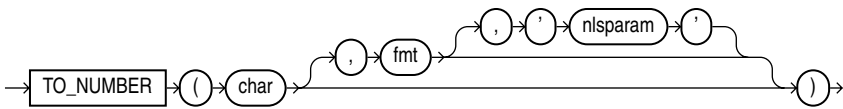
次の例では、TO_NCLOB ファンクションでデータを変換後、pm.print_media 表の NCLOB 列に文字データを挿入します。

```
INSERT INTO print_media (product_id, ad_id, ad_fltextn)
VALUES (3502, 31001,
      TO_NCLOB('Placeholder for new product description'));
```


TO_NUMBER

構文

`to_number::=`



用途

TO_NUMBER は、オプションの書式モデル *fmt* によって指定された書式の数値を含む CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型の値 *char* を、NUMBER データ型の値に変換します。

注意： このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。詳細は、2-42 ページの「[データ型の比較規則](#)」を参照してください。

例

次の例では、文字列データを数値に変換します。

```

UPDATE employees SET salary = salary +
  TO_NUMBER('100.00', '9G999D99')
WHERE last_name = 'Perkins';
  
```

このファンクションの *nlsparam* 文字列は、数値変換の TO_CHAR ファンクションの場合と同じ用途に使用されます。

参照： 6-162 ページの「[TO_CHAR \(数値\)](#)」を参照してください。

```

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

Amount
-----
-100
  
```

TO_SINGLE_BYTE

構文

to_single_byte::=



用途

TO_SINGLE_BYTE は、すべてのマルチバイト文字を、対応するシングルバイト文字に変換して *char* を戻します。*char* のデータ型は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 です。戻り値は、*char* と同じデータ型です。

char 内に同等のシングルバイト文字がないマルチバイト文字は、マルチバイト文字として出力されます。このファンクションは、ご使用のデータベース・キャラクタ・セットに、シングルバイト文字およびマルチバイト文字の両方が含まれている場合にのみ有効です。

注意： このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。詳細は、2-42 ページの「[データ型の比較規則](#)」を参照してください。

例

次の例では、UTF8 のマルチバイト 'A' からシングルバイトの ASCII の 'A' への変換を示します。

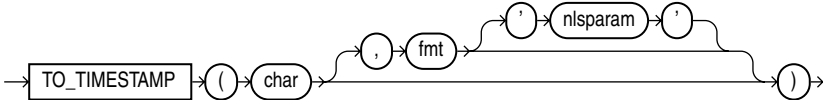
```
SELECT TO_SINGLE_BYTE( CHR(15711393)) FROM DUAL;
```

T
-
A

TO_TIMESTAMP

構文

to_timestamp::=



用途

TO_TIMESTAMP は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型の *char* を、TIMESTAMP データ型の値に変換します。

オプションの *fmt* は、*char* の書式を指定します。*fmt* を指定しない場合、*char* はデフォルト書式の TIMESTAMP データ型である必要があります。オプションの *nlsparam* は、日付変換の TO_CHAR ファンクションの場合と同じ用途で使用されます。

注意： このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。詳細は、2-42 ページの「[データ型の比較規則](#)」を参照してください。

例

```
次の例では、文字列をタイムスタンプに変換します。

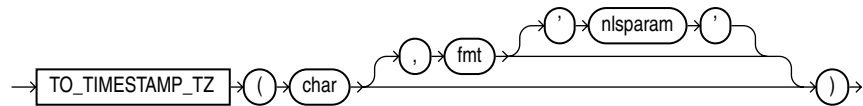
SELECT TO_TIMESTAMP ('1999-12-01 11:00:00', 'YYYY-MM-DD HH:MI:SS')
       FROM DUAL;

TO_TIMESTAMP('1999-12-0111:00:00','YYYY-MM-DDHH:MI:SS')
-----
01-DEC-99 11.00.00.0000000000 AM
```

TO_TIMESTAMP_TZ

構文

to_timestamp_tz::=



用途

TO_TIMESTAMP_TZ は、CHAR、VARCHAR2、NCHAR または NVARCHAR2 データ型の *char* を、TIMESTAMP WITH TIME ZONE データ型の値に変換します。

注意： このファンクションは、文字列を **TIMESTAMP WITH LOCAL TIME ZONE** に変換しません。変換するには、6-24 ページの「[CAST](#)」で示すように、CAST ファンクションを使用してください。

オプションの *fmt* は、*char* の書式を指定します。*fmt* を指定しない場合、*char* はデフォルト書式の `TIMESTAMP WITH TIME ZONE` データ型である必要があります。オプションの *nlsparam* は、日付変換の `TO_CHAR` ファンクションの場合と同じ用途で使用されます。

例

次の例では、文字列を `TIMESTAMP WITH TIME ZONE` 値に変換します。

```
SELECT TO_TIMESTAMP_TZ('1999-12-01 11:00:00 -8:00',
    'YYYY-MM-DD HH:MI:SS TZH:TZM') FROM DUAL;

TO_TIMESTAMP_TZ('1999-12-0111:00:00-08:00','YYYY-MM-DDHH:MI:SSTZH:TZM')
-----
01-DEC-99 11.00.00.000000000 AM -08:00
```

次の例では、サンプル表 `oe.order_items` および `oe.orders` を使用して、`UNION` 操作の `NULL` 列を `TIMESTAMP WITH LOCAL TIME ZONE` としてキャストします。

```
SELECT order_id, line_item_id,
    CAST(NULL AS TIMESTAMP WITH LOCAL TIME ZONE) order_date
    FROM order_items
    UNION
    SELECT order_id, to_number(null), order_date
    FROM orders;
```

TO_YMINTERVAL

構文

to_yminterval::=

→ TO_YMINTERVAL → (char) →

用途

`TO_YMINTERVAL` ファンクションは、`CHAR`、`VARCHAR2`、`NCHAR` または `NVARCHAR2` データ型の文字列を `INTERVAL YEAR TO MONTH` 型に変換します。*char* は、変換する文字列です。

例

次の例では、サンプル表 `hr.employees` の各従業員の雇用された後から 1 年と 2 ヶ月後の日付を計算します。

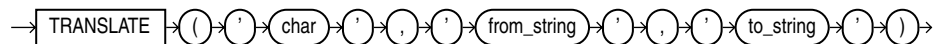
```
SELECT hire_date, hire_date + TO_YMINTERVAL('01-02') "14 months"
FROM employees;
```

```
HIRE_DATE 14 months
-----
17-JUN-87 17-AUG-88
21-SEP-89 21-NOV-90
13-JAN-93 13-MAR-94
03-JAN-90 03-MAR-91
21-MAY-91 21-JUL-92
.
.
.
```

TRANSLATE

構文

translate::=



用途

TRANSLATE は、*from_string* 内のすべての文字を *to_string* 内の対応する文字に置換して *char* を戻します。*from_string* 内に存在しない *char* 内の文字は置換されません。引数 *from_string* には、*to_string* より多い文字を指定できます。この場合、*from_string* の終わりにある余分な文字には、*to_string* 内に対応する文字がありません。これらの余分な文字が *char* 内にある場合、それらの文字は戻り値から削除されます。

戻り値から *from_string* 内の文字をすべて削除するために、*to_string* に空の文字列を使用することはできません。Oracle は空の文字列を NULL と解析し、NULL の引数がある場合、このファンクションは NULL を戻します。

注意： このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。詳細は、2-42 ページの「[データ型の比較規則](#)」を参照してください。

例

次の例では、ライセンス番号を変換します。「ABC...Z」のすべての文字は「X」に変換され、「012 ... 9」のすべての数字は「9」に変換されます。

```
SELECT TRANSLATE('2KRW229',
'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ', '9999999999XXXXXXXXXXXXXXXXXXXXXXXXXXXXX')
"License"
      FROM DUAL;
```

```
License
-----
9XX999
```

次の例では、文字を削除して数値のみになったライセンス番号を戻します。

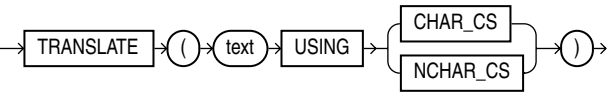
```
SELECT TRANSLATE('2KRW229',
'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ', '0123456789')
"Translate example"
      FROM DUAL;
```

```
Translate example
-----
2229
```

TRANSLATE ...USING

構文

translate_using::=



用途

TRANSLATE ... USING は、text をデータベース・キャラクタ・セットと各国語キャラクタ・セット間の変換に指定されたキャラクタ・セットに変換します。

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 が各国語キャラクタ・セットに変換されます。出力データ型は NVARCHAR2 です。

このファンクションは、Oracle の CONVERT ファンクションと似ていますが、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 ではなくこのファンクションを使用する必要があります。入力に UCS2 コードポイントまたはバックスラッシュ (\) が含まれる場合、UNISTR ファンクションを使用してください。

参照： 6-32 ページの「[CONVERT](#)」および 6-183 ページの「[UNISTR](#)」を参照してください。

例

次に、表 t1 およびその値を使用する例を示します。

```
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

```
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

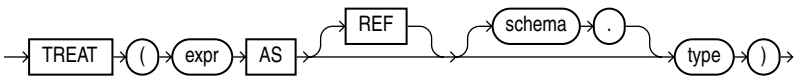
```
UPDATE t1 SET
  nchar_col = TRANSLATE('deo' USING NCHAR_CS);
UPDATE t1 SET
  char_col = TRANSLATE(N'deo' USING CHAR_CS);
SELECT * FROM t1;
```

CHAR_COL	NCHAR_COL
deo	deo

TREAT

構文

treat::=



用途

TREAT ファンクションを使用すると、式の宣言された型を変更できます。

このファンクションを使用する場合は、*type* に対する EXECUTE オブジェクト権限が必要です。

- *expr* の宣言された型が *source_type* である場合、*type* はスーパータイプまたは *source_type* のサブタイプである必要があります。*expr* の最も指定される型が *type* (または *type* のサブタイプ) である場合、TREAT は *expr* を戻します。*expr* の最も指定される型が *type* (または *type* のサブタイプ) ではない場合、TREAT は NULL を戻します。
- *expr* の宣言された型が REF *source_type* である場合、*type* はサブタイプまたは *source_type* のスーパータイプである必要があります。DEREF(*expr*) の最も指定される型が *type* (または *type* のサブタイプ) である場合、TREAT は *expr* を戻します。DEREF(*expr*) の最も指定される型が *type* (または *type* のサブタイプ) ではない場合、TREAT は NULL を戻します。

注意： このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。詳細は、2-42 ページの「[データ型の比較規則](#)」を参照してください。

例

次の例では、14-52 ページの「置換可能な表および列のサンプル」で作成された表 `oe.persons` を使用します。この表は、15-20 ページの「型の階層例」で作成された `person_t` 型に基づいています。次の例では、`persons` 表のすべての人物の給与属性を検索します。従業員ではない人物のインスタンスの値は `NULL` です。

```
SELECT name, TREAT(VALUE(p) AS employee_t).salary salary
FROM persons p;
```

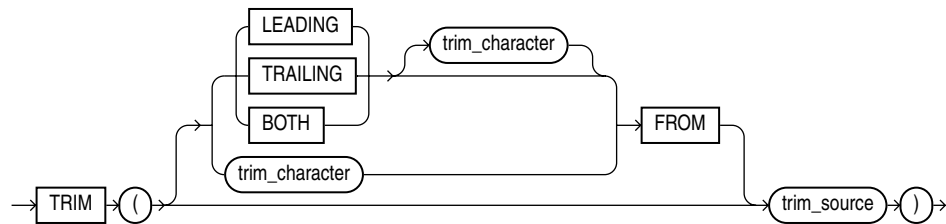
NAME	SALARY
-----	-----
Bob	
Joe	100000
Tim	1000

`TREAT` ファンクションを使用すると、置換可能列のサブタイプ属性に索引を作成できます。12-82 ページの「置換可能な列の索引の例」の例を参照してください。

TRIM

構文

`trim::=`



用途

`TRIM` によって、文字列の先行または後続文字（あるいはその両方）を切り捨てることができます。`trim_character` または `trim_source` が文字リテラルの場合、引用符で囲む必要があります。

- `LEADING` を指定すると、Oracle は `trim_character` と等しい先行文字を削除します。
- `TRAILING` を指定すると、Oracle は `trim_character` と等しい後続文字を削除します。
- `BOTH` を指定するか、またはいずれも指定しない場合、Oracle は `trim_character` と等しい先行および後続文字を削除します。

- *trim_character* を指定しないと、デフォルト値は空白になります。
- *trim_source* のみを指定すると、Oracle は先行および後続空白を削除します。
- このファンクションは、値を VARCHAR2 データ型で返します。値の最大長は、*trim_source* の長さです。
- *trim_source* または *trim_character* のいずれかが NULL 値の場合、TRIM ファンクションは NULL 値を返します。

trim_character および *trim_source* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻される文字列は、VARCHAR2 データ型であり、*trim_source* と同じキャラクタ・セットです。

例

この例では、先行および後続 0（ゼロ）を数値から切り捨てます。

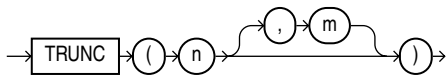
```
SELECT TRIM (0 FROM 0009872348900) "TRIM Example"
FROM DUAL;

TRIM example
-----
      98723489
```

TRUNC（数値）

構文

trunc_number::=



用途

TRUNC は、*n* を小数第 *m* 位までに切り捨てた値を返します。*m* を指定しない場合、*n* の小数点以下を切り捨てます。*m* が負の場合は、小数点の左 *m* 桁を切り捨てて、0（ゼロ）にします。

例

次の例では、数値を切り捨てます。

```
SELECT TRUNC(15.79,1) "Truncate" FROM DUAL;
```

```
Truncate
-----
      15.7
```

```
SELECT TRUNC(15.79,-1) "Truncate" FROM DUAL;
```

```
Truncate
-----
      10
```

TRUNC (日付)

構文

trunc_date::=



用途

TRUNC は、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 *date* を戻します。
fmt を省略すると、*date* は最も近い日に切り捨てられます。

参照： *fmt* で使用できる書式モデルについては、6-195 ページの
「[ROUND](#) および [TRUNC](#) 日付ファンクション」を参照してください。

例

次の例では、日付を切り捨てます。

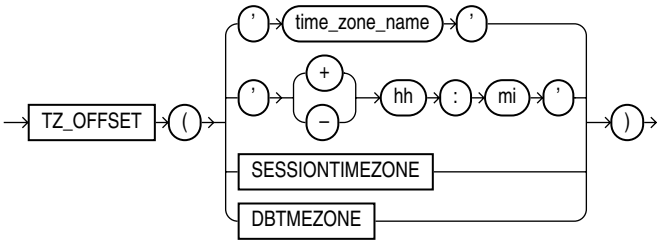
```
SELECT TRUNC(TO_DATE('27-OCT-92','DD-MON-YY'), 'YEAR')
       "New Year" FROM DUAL;
```

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

TZ_OFFSET

構文

tz_offset::=



用途

TZ_OFFSET は、文が実行された日付に基づいて入力された値に対応するタイム・ゾーン・オフセットを戻します。有効なタイム・ゾーン名、UTC からのタイム・ゾーン・オフセット（それ自体を戻します）、あるいはキーワード SESSIONTIMEZONE または DBTIMEZONE を入力できます。有効な値を表示するには、V\$TIMEZONE_NAMES 動的パフォーマンス・ビューの TZNAME 列を問い合わせます。

参照： 動的パフォーマンス・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

例

次の例では、UTC からの米国 / 東部タイム・ゾーンのタイム・ゾーン・オフセットを戻します。

```
SELECT TZ_OFFSET('US/Eastern') FROM DUAL;
```

```
TZ_OFFSET
-----
-04:00
```

UID

構文

`uid::=`



用途

UID は、セッション・ユーザー（ログインしているユーザー）を一意に識別する整数を返します。

例

次の例では、現行のユーザーの UID を返します。

```
SELECT UID FROM DUAL;
```

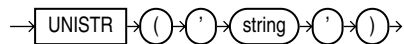
```

      UID
-----
      19
  
```

UNISTR

構文

`unistr::=`



用途

UNISTR は、任意のキャラクタ・セットの文字列を引数として、データベース Unicode キャラクタ・セットの Unicode の文字列を返します。UCS2 コードポイント文字を文字列に含めるには、次の数値の前にエスケープ文字であるバックスラッシュ（\）を使用します。バックスラッシュ自体を含めるには、バックスラッシュをもう 1 つ追加します（\\）。

UNISTR が UCS2 コードポイントおよびバックスラッシュを指定できる点を除くと、このファンクションは、TRANSLATE ... USING ファンクションと似ています。

参照：

- Unicode キャラクタ・セットおよび文字セマンティクスの詳細は、『Oracle9i データベース概要』を参照してください。
- 6-176 ページの「[TRANSLATE ... USING](#)」を参照してください。

例

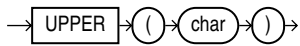
次の例では、文字列と同じ Unicode を戻します。

```
SELECT unistr('\00D6') FROM DUAL;  
  
UN  
--  
Ö
```

UPPER

構文

`upper::=`



用途

UPPER は、すべての文字を大文字にして *char* を戻します。*char* は、CHAR、VARCHAR2、NCHAR、NVARCHAR2、CLOB または NCLOB データ型です。戻り値は、*char* と同じデータ型です。

例

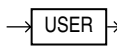
次の例では、文字列を大文字で戻します。

```
SELECT UPPER('Large') "Uppercase"  
      FROM DUAL;  
  
Upper  
-----  
LARGE
```

USER

構文

user::=



用途

USER は、セッション・ユーザー（ログインしているユーザー）の名前を VARCHAR2 データ型で戻します。Oracle は、空白埋め比較セマンティクスでこのファンクションの値を比較します。

分散 SQL 文では、UID ファンクションおよび USER ファンクションは、ローカル・データベース上のユーザーを識別します。チェック制約の条件でこれらのファンクションは使用できません。

例

次の例では、現行のユーザーおよびユーザーの UID を戻します。

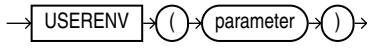
```
SELECT USER, UID FROM DUAL;
```

USER	UID
-----	-----
OE	33

USERENV

構文

userenv::=



用途

注意： USERENV は、下位互換性が保持されるレガシー・ファンクションです。現行の機能に対して組込み USERENV ネームスペースとともに SYS_CONTEXT ファンクションを使用することをお勧めします。詳細は、6-145 ページの「[SYS_CONTEXT](#)」を参照してください。

USERENV は、現行のセッションに関する 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、またはセッションで現在使用されている言語固有の文字を判断する場合に有効です。チェック制約の条件で、USERENV は使用できません。表 6-11 に、parameter 引数の値を示します。

表 6-11 USERENV パラメータ

パラメータ	戻り値
'CLIENT_INFO'	<p>DBMS_APPLICATION_INFO パッケージを使用するアプリケーションが格納できる 64 バイトまでのユーザー・セッション情報を戻します。</p> <p>注意：商業用のアプリケーションによっては、このコンテキスト値を使用する可能性があります。このコンテキスト領域の使用に対する制限については、これらのアプリケーションのドキュメントを確認してください。</p> <p>アプリケーション・コンテキストの機能、または USERENV ネームスペースとともに SYS_CONTEXT ファンクションを使用することをお勧めします。これらは、より安全性が高く、柔軟性があります。</p> <p>参照：</p> <ul style="list-style-type: none">- アプリケーション・コンテキストの詳細は、『Oracle9i データベース概要』を参照してください。- 12-11 ページの「CREATE CONTEXT」および 6-145 ページの「SYS_CONTEXT」を参照してください。
'ENTRYID'	<p>使用可能な監査エントリ識別子を戻します。この属性を分散 SQL 文で使用することはできません。USERENV でこのキーワードを使用するには、AUDIT_TRAIL 初期化パラメータに true を設定する必要があります。</p>
'INSTANCE'	<p>現行のインスタンスのインスタンス識別番号を戻します。</p>
'ISDBA'	<p>ISDBA は、オペレーティング・システムまたはパスワード・ファイルによって、ユーザーが DBA 権限を持っていると認証された場合、'TRUE' を戻します。</p>
'LANG'	<p>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。</p>
'LANGUAGE'	<p>セッションで現在使用している言語 (language) および地域 (territory) を、データベース・キャラクタ・セット (character set) も含めた次の書式で戻します。</p> <p>language_territory.characterset</p>
'SESSIONID'	<p>監査セッション識別子を戻します。この属性を分散 SQL 文で使用することはできません。</p>

表 6-11 USERENV パラメータ（続き）

パラメータ	戻り値
'TERMINAL'	現行のセッションの端末に対するオペレーティング・システム識別子を返します。分散 SQL 文では、この属性はローカル・セッションの識別子を返します。分散環境では、リモートの SELECT に対してのみこのオプションを使用でき、リモートの INSERT、UPDATE または DELETE には使用できません。

例

次の例では、現行のセッションの LANGUAGE パラメータを返します。

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

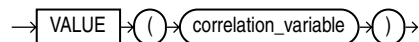
```
Language
```

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

VALUE

構文

value::=



用途

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、オブジェクト表に格納されたオブジェクト・インスタンスを返します。オブジェクト・インスタンスの型は、オブジェクト表と同じ型です。

例

次の例では、オブジェクト表へのオブジェクト・インスタンスの格納方法を示します。

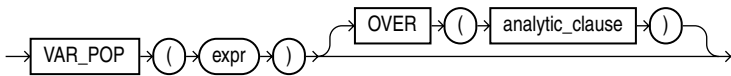
```
CREATE TYPE emp_type AS OBJECT  
    (eno NUMBER, ename VARCHAR2(20), salary NUMBER);  
CREATE TABLE emp_table OF emp_type  
    (primary key (eno, ename));  
INSERT INTO emp_table VALUES (10, 'jack', 50000);  
SELECT VALUE(e) FROM emp_table e;
```

```
VALUE(E) (ENO, ENAME, SALARY)
-----
EMP_TYPE(10, 'jack', 50000)
```

VAR_POP

構文

var_pop::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

VAR_POP は、数値の集合にある NULL を削除した後、この集合の母集団分散を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

expr は数値式であり、このファンクションは NUMBER 型の値を戻します。ファンクションが空の集合に適用されると、NULL を戻します。このファンクションは、次の計算を行います。

$$(\text{SUM}(\text{expr}^2) - \text{SUM}(\text{expr})^2 / \text{COUNT}(\text{expr})) / \text{COUNT}(\text{expr})$$

参照：

- 6-6 ページの「[集計ファンクション](#)」を参照してください。
- expr の書式の詳細は、4-2 ページの「[SQL 式](#)」を参照してください。

集計の例

次の例では、employees 表にある給与の人口分散を計算します。

```
SELECT VAR_POP(salary) FROM employees;
```

```
VAR_POP(SALARY)
-----
15140307.5
```

分析の例

次の例では、1998 年における月ごとの売上について、累積の母集団および標本分散を計算します。

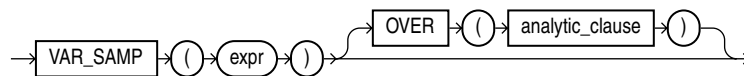
```
SELECT t.calendar_month_desc,
       VAR_POP(SUM(s.amount_sold))
         OVER (ORDER BY t.calendar_month_desc) "Var_Pop",
       VAR_SAMP(SUM(s.amount_sold))
         OVER (ORDER BY t.calendar_month_desc) "Var_Samp"
FROM sales s, times t
WHERE s.time_id = t.time_id AND t.calendar_year = 1998
GROUP BY t.calendar_month_desc;
```

CALENDAR	Var_Pop	Var_Samp
1998-01	0	
1998-02	1.2844E+11	2.5687E+11
1998-03	9.1176E+10	1.3676E+11
1998-04	2.6891E+11	3.5855E+11
1998-05	1.9659E+12	2.4574E+12
1998-06	2.5810E+12	3.0972E+12
1998-07	3.5467E+12	4.1378E+12
1998-08	3.5100E+12	4.0114E+12
1998-09	3.3199E+12	3.7349E+12
1998-10	3.5001E+12	3.8890E+12
1998-11	3.2789E+12	3.6068E+12
1998-12	4.2486E+12	4.6348E+12

VAR_SAMP

構文

var_samp::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析関クション](#)」を参照してください。

用途

VAR_SAMP は、数値の集合にある NULL を削除した後、この集合の標本分散を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

expr は数値式であり、このファンクションは NUMBER 型の値を戻します。ファンクションが空の集合に適用されると、NULL を戻します。このファンクションは、次の計算を行います。

$$(SUM(expr^2) - SUM(expr)^2 / COUNT(expr)) / (COUNT(expr) - 1)$$

このファンクションは、1つの要素の集合を入力すると VARIANCE は 0 を戻し、VAR_SAMP は NULL を戻すということを除いては、VARIANCE に似ています。

参照：

- 6-6 ページの「集計ファンクション」を参照してください。
- expr の書式の詳細は、4-2 ページの「SQL 式」を参照してください。

集計の例

次の例では、サンプル表 employees にある給与の標本分散を計算します。

```
SELECT VAR_SAMP(salary) FROM employees;
```

```
VAR_SAMP(SALARY)
-----
15283140.5
```

分析の例

次の例では、1998 年における月ごとの売上について、累積の母集団および標本分散を計算します。

```
SELECT t.calendar_month_desc,
       VAR_POP(SUM(s.amount_sold))
         OVER (ORDER BY t.calendar_month_desc) "Var_Pop",
       VAR_SAMP(SUM(s.amount_sold))
         OVER (ORDER BY t.calendar_month_desc) "Var_Samp"
FROM sales s, times t
WHERE s.time_id = t.time_id AND t.calendar_year = 1998
GROUP BY t.calendar_month_desc;
```

```
CALENDAR    Var_Pop    Var_Samp
-----
1998-01          0
1998-02  1.2844E+11  2.5687E+11
1998-03  9.1176E+10  1.3676E+11
1998-04  2.6891E+11  3.5855E+11
```

```

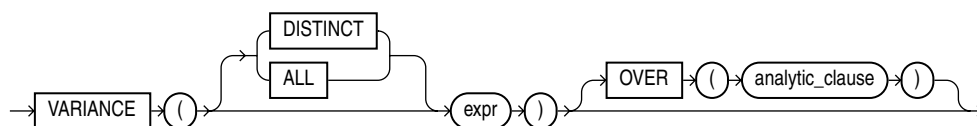
1998-05  1.9659E+12  2.4574E+12
1998-06  2.5810E+12  3.0972E+12
1998-07  3.5467E+12  4.1378E+12
1998-08  3.5100E+12  4.0114E+12
1998-09  3.3199E+12  3.7349E+12
1998-10  3.5001E+12  3.8890E+12
1998-11  3.2789E+12  3.6068E+12
1998-12  4.2486E+12  4.6348E+12

```

VARIANCE

構文

variance::=



参照： 構文、セマンティクスおよび制限事項の詳細は、6-8 ページの「[分析ファンクション](#)」を参照してください。

用途

VARIANCE は *expr* の分散を戻します。これは、集計ファンクションまたは分析ファンクションとして使用できます。

expr の分散の計算結果は、次のようになります。

- *expr* の行数が 1 の場合は 0（ゼロ）
- *expr* の行数が 1 を超える場合は VAR_SAMP

DISTINCT を指定する場合は、*analytic_clause* の *query_partition_clause* のみ指定できます。*order_by_clause* および *windowing_clause* は指定できません。

参照：

- 6-6 ページの「[集計ファンクション](#)」を参照してください。
- *expr* の書式の詳細は、4-2 ページの「[SQL 式](#)」を参照してください。

集計の例

次の例では、サンプル表 `employees` にあるすべての給与の合計を計算します。

```
SELECT VARIANCE(salary) "Variance"
FROM employees;
```

```
Variance
-----
15283140.5
```

分析の例

次の例では、雇用日で順序付けられた部門 30 の給与の値の累積分散を戻します。

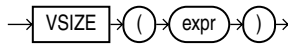
```
SELECT last_name, salary, VARIANCE(salary)
      OVER (ORDER BY hire_date) "Variance"
FROM employees
WHERE department_id = 30;
```

LAST_NAME	SALARY	Variance
-----	-----	-----
Raphaely	11000	0
Khoo	3100	31205000
Tobias	2800	21623333.3
Baida	2900	16283333.3
Himuro	2600	13317000
Colmenares	2500	11307000

VSIZE

構文

`vsize::=`



用途

`VSIZE` は、`expr` の内部表現でのバイト数を戻します。`expr` が `NULL` の場合は `NULL` を戻します。

注意： このファンクションは、CLOB データを直接的にサポートしていません。ただし、暗黙的なデータ変換を使用して CLOB を引数として渡すことはできます。詳細は、2-42 ページの「[データ型の比較規則](#)」を参照してください。

例

次の例では、部門 10 の従業員の `last_name` のバイト数を戻します。

```

SELECT last_name, VSIZE (last_name) "BYTES"
  FROM employees
 WHERE department_id = 10;

```

LAST_NAME	BYTES

Whalen	6

WIDTH_BUCKET

構文

width_bucket::=

```

→ WIDTH_BUCKET ( ( ( expr , min_value , max_value , num_buckets ) ) ) →

```

用途

`WIDTH_BUCKET` ファンクションを使用すると、ヒストグラムの幅が同じサイズに分割された等幅ヒストグラムを作成できます。このファンクションを等高ヒストグラムを作成する `NTILE` と比較してください。各バケットが実際の数値線幅の「closed-open」間隔であることが理想です。たとえば、10 が間隔に含まれ 20 が排除されることを示すために、バケットは 10.00 ～ 19.999... のスコアに割り当てられます。これは、[10, 20] と表すこともできます。

指定された式に対して、`WIDTH_BUCKET` は、この式の値が評価された後に該当するバケット数を戻します。

- `expr` は、ヒストグラムが作成される式です。この式は、数値または日時値に評価される必要があります。`expr` が `NULL` と評価された場合、式は `NULL` を戻します。
- `min_value` および `max_value` は、`expr` の許容域のエンド・ポイントに解決される式です。これらの式は両方とも数値または日付値に評価される必要があります、いずれも `NULL` に評価されません。
- `num_buckets` は、バケット数を示す定数に解決される式です。この式は、正の整数に評価される必要があります。

Oracle は、必要に応じて、0 (ゼロ) の下位バケットおよび `num_buckets+1` の下位バケットを作成します。これらのバケットは、`min_value` 未満および `max_value` より大きい値を処理し、エンド・ポイントの妥当性チェックに有効です。

例

次の例では、サンプル表 `oe.customers` のスイスの顧客の `credit_limit` 列に 10 バケットのヒストグラムを作成し、各顧客のバケット数（「クレジット・グループ」）を戻します。最大値を超えるクレジット利用限度額を持つ顧客は、下位バケット 11 に割り当てられます。

```
SELECT customer_id, cust_last_name, credit_limit,
       WIDTH_BUCKET(credit_limit, 100, 5000, 10) "Credit Group"
FROM customers WHERE nls_territory = 'SWITZERLAND'
ORDER BY "Credit Group";
```

CUSTOMER_ID	CUST_LAST_NAME	CREDIT_LIMIT	Credit Group
825	Dreyfuss	500	1
826	Barkin	500	1
853	Palin	400	1
827	Siegel	500	1
843	Oates	700	2
844	Julius	700	2
835	Eastwood	1200	3
840	Elliott	1400	3
842	Stern	1400	3
841	Boyer	1400	3
837	Stanton	1200	3
836	Berenger	1200	3
848	Olmos	1800	4
849	Kaurusmdki	1800	4
828	Minnelli	2300	5
829	Hunter	2300	5
852	Tanner	2300	5
851	Brown	2300	5
850	Finney	2300	5
830	Dutt	3500	7
831	Bel Geddes	3500	7
832	Spacek	3500	7
838	Nicholson	3500	7
839	Johnson	3500	7
833	Moranis	3500	7
834	Idle	3500	7
845	Fawcett	5000	11
846	Brando	5000	11
847	Streep	5000	11

ROUND および TRUNC 日付ファンクション

表 6-12 に、ROUND および TRUNC 日付ファンクションで使える書式モデル、および日付の丸めと切捨ての単位を示します。デフォルトのモデル DD では、午前 0 時（真夜中）を基準に丸めおよび切捨てを行い、日付を戻します。

表 6-12 ROUND および TRUNC 日付ファンクションの日付書式モデル

書式モデル	丸め単位または切捨て単位
CC SCC	4 桁の年号の上 2 桁より 1 大きい数
SYYYYY YYYY YEAR SYEAR YYY YY Y	年（7 月 1 日に切上げ）
IYYYY IY IY I	ISO 年
Q	四半期（その四半期の 2 番目の月の 16 日に切上げ）
MONTH MON MM RM	月（16 日に切上げ）
WW	年の最初の日と同じ曜日
IW	ISO 年の最初の日と同じ曜日
W	月の最初の日と同じ曜日
DDD DD J	日
DAY DY D	週の開始日
HH HH12 HH24	時
MI	分

書式モデル DAY、DY および D によって使用される週の開始日は、NLS_TERRITORY 初期化パラメータによって暗黙的に指定されています。

参照： このパラメータの詳細は、『Oracle9i データベース・リファレンス』および『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

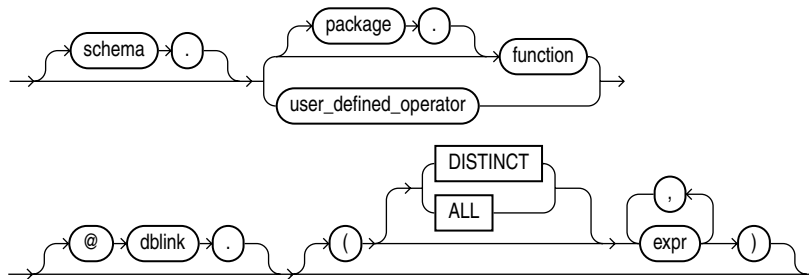
ユーザー定義ファンクション

PL/SQL または Java でユーザー定義ファンクションを作成し、SQL または SQL 組込みファンクションにはない機能を持たせることができます。ユーザー定義ファンクションは、SQL ファンクションを指定できるすべての SQL 文に指定できます。式を指定できる場所であれば、どこでも使用できます。

たとえば、SQL 文の次の場所でユーザー定義ファンクションを使用できます。

- SELECT 文の SELECT 構文のリスト
- WHERE 句の条件
- CONNECT BY 句、START WITH 句、ORDER BY 句および GROUP BY 句
- INSERT 文の VALUES 句
- UPDATE 文の SET 句

user_defined_function::=



オプションの式のリストは、ファンクション、パッケージまたは演算子の属性と一致する必要があります。

制限事項： DISTINCT および ALL キーワードは、ユーザー定義集計ファンクションのみで有効です。

参照：

- ファンクションの作成（ユーザー定義ファンクションの制限を含む）の詳細は、12-47 ページの「[CREATE FUNCTION](#)」を参照してください。
- ユーザー・ファンクションの作成および使用方法の詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

前提条件

ユーザー定義ファンクションを SQL 文で使用するには、トップレベル・ファンクションとして作成するか、またはパッケージ仕様部で宣言する必要があります。

SQL の式の中でユーザー・ファンクションを使用するには、ユーザーがユーザー・ファンクションの EXECUTE 権限を持っている必要があります。ユーザー・ファンクションで定義したビューを問い合わせるには、そのビューに対する SELECT 権限が必要です。ビューから選択するには、個別の EXECUTE 権限は必要ありません。

参照：

- トップレベル・ファンクションの詳細は、12-47 ページの「[CREATE FUNCTION](#)」を参照してください。
- パッケージ・ファンクションの詳細は、13-46 ページの「[CREATE PACKAGE](#)」を参照してください。

名前の優先順位

SQL 文内では、データベースの列名は、パラメータなしのファンクション名より優先順位が高くなります。たとえば、Human Resources のマネージャが、hr スキーマに次の 2 つのオブジェクトを作成する場合は、次のようにします。

```
CREATE TABLE new_emps (new_sal NUMBER, ...);  
CREATE FUNCTION new_sal RETURN NUMBER IS BEGIN ... END;
```

その後、次の 2 つの文のように、new_sal を参照すると new_emps.new_sal 列を参照することになります。

```
SELECT new_sal FROM new_emps;  
SELECT new_emps.new_sal FROM new_emps;
```

new_sal ファンクションにアクセスするには、次のように入力します。

```
SELECT hr.new_sal FROM new_emps;
```

SQL の式内で使用できるユーザー・ファンクションのコール例を次に示します。

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

例 スキーマ hr から tax_rate ユーザー・ファンクションをコールし、tax_table 内の sal 列と SAL 列に対してこのファンクションを実行し、その結果を income_tax 変数に入れるには、次のように指定します。

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

ネーミング規則

オプションのスキーマ名またはパッケージ名を 1 つのみ指定すると、最初の識別子はスキーマ名またはパッケージ名のいずれかになります。たとえば、PAYROLL.TAX_RATE という参照内の PAYROLL がスキーマ名かパッケージ名かを判断するには、Oracle は次の手順を実行します。

1. カレント・スキーマ内の PAYROLL パッケージをチェックします。
2. PAYROLL パッケージが検出されない場合は、トップレベルの TAX_RATE ファンクションを含むスキーマ名 PAYROLL を検索します。このようなファンクションが検出されない場合は、エラーを戻します。
3. カレント・スキーマ内で PAYROLL パッケージが検出されると、PAYROLL パッケージの中で TAX_RATE ファンクションを検索します。このようなファンクションが検出されない場合は、エラーを戻します。

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

SQL 問合せおよびその他の SQL 文

この章では、問合せおよびその他の Oracle SQL 文について説明します。この章では、次の内容を説明します。

- [問合せおよび副問合せ](#)
- [様々な種類の SQL 文](#)

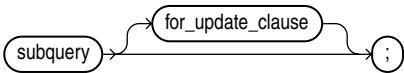
問合せおよび副問合せ

問合せとは、1 つ以上の表またはビューからデータを検索する操作のことです。このマニュアルでは、トップレベルの SELECT 文を**問合せ**といい、他の SQL 文の中でネストされた問合せを**副問合せ**といいます。

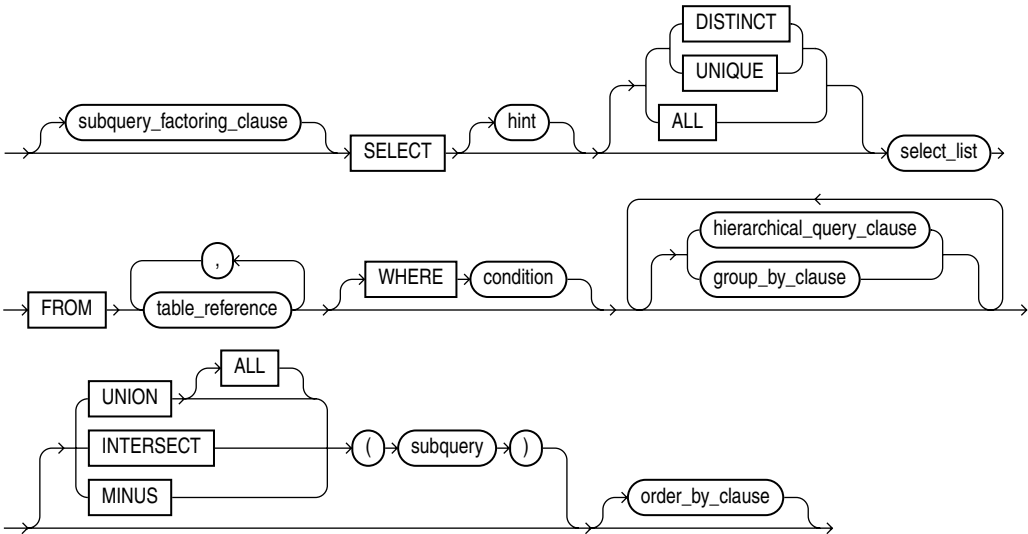
この項では、問合せおよび副問合せの種類およびその使用方法について説明します。この章では、トップレベルの構文について説明します。

参照： すべての句のすべての構文、キーワードおよびパラメータのセマンティクスについては、17-4 ページの「[SELECT](#)」を参照してください。

select::=



subquery::=



単純な問合せの作成

SELECT キーワードの後、FROM 句の前にある式のリストを、**SELECT 構文のリスト**といいます。SELECT 構文のリストに、1 つ以上の表、ビューおよびマテリアライズド・ビューから Oracle が戻す行に含まれる 1 つ以上の列を指定します。SELECT 構文のリストの要素によって、列のデータ型、長さおよび数が決定されます。

複数の表に同じ名前の列がある場合、表の名前でその列名を修飾する必要があります。それ以外の場合は、完全に修飾した列名はオプションとなります。ただし、明示的に表および列の参照を修飾することをお勧めします。表および列名を完全に修飾することで、Oracle の作業が少なくなります。

列の別名 *c_alias* を使用して、SELECT 構文のリストの前の式にラベルを付けると、列が新しい見出し付きで表示されます。別名によって、問合せ中に SELECT 構文のリストの項目の名前を効果的に変更できます。別名は ORDER BY 句の中で使用できますが、問合せ内のその他の句には使用できません。

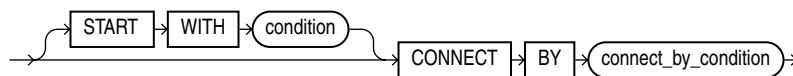
Oracle オプティマイザに指示（ヒント）を与えるために、SELECT 文中でコメントを使用できます。オプティマイザは、これらのヒントを使用して文の実行計画を選択します。

参照： ヒントの詳細は、2-86 ページの「[ヒント](#)」および『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。

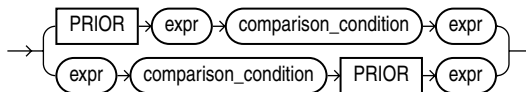
階層問合せ

表に階層データが含まれる場合、階層問合せ句を使用して階層順に行を選択することができます。

hierarchical_query_clause::=



connect_by_condition::=



- START WITH 句では、階層のルート行を指定します。
- CONNECT BY 句では、階層の親 / 子の行の関連を指定します。
connect_by_condition の一部には、PRIOR 演算子を使用して親である行を参照させる必要があるものもあります。
- PRIOR は、階層問合せの現在の行の親である行について、*connect_by_condition* を評価します。PRIOR は単項演算子であり、単項算術演算子の + および - と同じ優先順位を持っています。

階層問合せでの WHERE 句の Oracle の処理は、WHERE 句が結合を含むかどうかによって異なります。

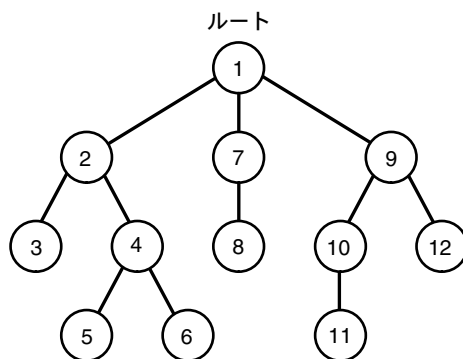
- WHERE 述語が結合を含む場合、Oracle は、CONNECT BY 処理の前に述語結合を適用します。
- すべての述語に結合を含む WHERE 句がない場合、Oracle は、CONNECT BY 処理の後に、非述語結合を適用します。この場合、階層の他の行に影響はありません。

Oracle は階層問合せ句からの情報を使用して、次の手順で階層を形成します。

1. Oracle は、CONNECT BY 句の前または後に WHERE 句を処理します。これは、WHERE 句が述語結合を含むかどうかによって決定されます。
2. Oracle は、階層のルート行を選択します。これらの行は、START WITH 条件を満たすものです。
3. Oracle は、各ルート行の子である行を選択します。子である各行は、1 つのルート行に関して CONNECT BY 条件を満たす必要があります。
4. Oracle は、子である行の連続生成を選択します。まず、手順 3 で戻された子である行を選択し、その行にある子を選択します（以降同様に続きます）。現在の親である行に関する CONNECT BY 条件を評価することによって、常に子を選択します。
5. 問合せに結合を含まない WHERE 句が含まれる場合、Oracle は、階層から WHERE 句の条件を満たさないすべての行を排除します。条件を満たさない子である行をすべて排除するのではなく、各行に関してこの条件をそれぞれ評価します。

6. Oracle は、[図 7-1](#) に示す順序で行を戻します。この図では、親である行の下に子である行が表示されます。階層ツリーの詳細は、2-82 ページの[図 2-1 「階層ツリー」](#)を参照してください。

図 7-1 階層問合せ



親である行に対する子を検索するために、Oracle は、親である行の CONNECT BY 条件の PRIOR 式、および各行の他の式を表の中で評価します。条件が TRUE となる行が、その親である行の子です。CONNECT BY 条件に、問合せによって選択された行をさらにフィルタ処理するための他の条件を含めることができます。CONNECT BY 条件に、副問合せを含めることはできません。

CONNECT BY 条件が階層のループになった場合、Oracle はエラーを戻します。1 つの行が別の行の親（または親の親または祖先）および子（または子の子または子孫）の場合、ループが発生します。

例

次の階層問合せは、CONNECT BY を使用して従業員とマネージャの関係を定義しています。

```
SELECT employee_id, last_name, manager_id
FROM employees
CONNECT BY PRIOR employee_id = manager_id;
```

EMPLOYEE_ID	LAST_NAME	MANAGER_ID
101	Kochhar	100
108	Greenberg	101
109	Faviet	108
110	Chen	108
111	Sciarra	108
112	Urman	108
113	Popp	108
200	Whalen	101

次の例は、前述の例と似ていますが、LEVEL 疑似列を使用して、親および子である行を表示しています。

```
SELECT employee_id, last_name, manager_id, LEVEL
FROM employees
CONNECT BY PRIOR employee_id = manager_id;
```

EMPLOYEE_ID	LAST_NAME	MANAGER_ID	LEVEL
101	Kochhar	100	1
108	Greenberg	101	2
109	Faviet	108	3
110	Chen	108	3
111	Sciarra	108	3
112	Urman	108	3
113	Popp	108	3

参照：

- 階層問合せでの LEVEL 疑似列の処理方法については、2-82 ページの「[LEVEL](#)」を参照してください。
- ルートからノードへの列の値のパスの検索については、6-144 ページの「[SYS_CONNECT_BY_PATH](#)」を参照してください。

UNION [ALL]、INTERSECT および MINUS 演算子

集合演算子 UNION、UNION ALL、INTERSECT および MINUS を使用して、複数の問合せを組み合わせることができます。集合演算子の優先順位はすべて同じです。SQL 文に複数の集合演算子がある場合、カッコによって明示的に別の順序が指定されないかぎり、Oracle は左から右の順に評価します。

複合問合せを構成する各問合せと、それに対応する SELECT 構文のリスト内の各式は、数値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 になります。
- 問合せのどちらか一方または両方が、VARCHAR2 データ型の値を選択する場合、戻される値のデータ型は VARCHAR2 になります。

集合演算子の制限事項：

- 集合演算子は、データ型が BLOB、CLOB、BFILE、VARRAY またはネストした表である列に対しては無効になります。
- UNION、INTERSECT および MINUS 演算子は、LONG 列に対しては無効になります。
- 列を参照する場合は、別名を使用して列に名前を付ける必要があります。
- *for_update_clause* は、これらの集合演算子とともに指定できません。
- これらの演算子の副問合せには、*order_by_clause* を指定できません。
- TABLE コレクション式を含む SELECT 文では、これらの演算子を使用できません。

注意： SQL 標準に準拠するために、Oracle の今後のリリースでは、他の集合演算子より優先順位の高い INTERSECT 演算子が提供されます。したがって、INTERSECT 演算子と他の集合演算子を使用する問合せでは、カッコを使用して評価順序を指定してください。

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、他の表に存在していない列がある場合に、(TO_CHAR ファンクションを使用して) データ型を一致させる必要があることを示しています。

```
SELECT location_id, department_name "Department",
       TO_CHAR(NULL) "Warehouse" FROM departments
UNION
SELECT location_id, TO_CHAR(NULL) "Department", warehouse_name
FROM warehouses;
```

LOCATION_ID	Department	Warehouse
1400	IT	
1400		Southlake, Texas
1500	Shipping	
1500		San Francisco
1600		New Jersey
1700	Accounting	
1700	Administration	
1700	Benefits	
1700	Construction	
.		
.		
.		

UNION ALL の例 UNION ALL 演算子がすべての行を戻すのに対して、UNION 演算子は重複しない行のみを戻します。UNION ALL 演算子は、重複行を評価しません。

```
SELECT product_id FROM order_items
UNION
SELECT product_id FROM inventories;

SELECT location_id FROM locations
UNION ALL
SELECT location_id FROM departments;
```

問合せで複数回戻される location_id 値 (1700 など) は、UNION 演算子では 1 回のみ戻されますが、UNION ALL 演算子では複数回戻されています。

INTERSECT の例 次の文は、INTERSECT 演算子によって 2 つの結果を結合しています。この場合、両方の問合せによって共通に返される行のみが返されます。

```
SELECT product_id FROM inventories
INTERSECT
SELECT product_id FROM order_items;
```

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、最初の間合せでは戻されるが、2 番目の間合せでは戻されない行のみが返されます。

```
SELECT product_id FROM inventories
MINUS
SELECT product_id FROM order_items;
```

問合せ結果のソート

ORDER BY 句を使用して、問合せによって選択された行を順序付けます。位置のソートは次のような場合に有効です。

- 長い SELECT 構文のリストの式によって順序付けるためには、ORDER BY 句で全体の式を複製するのではなく、その位置を指定することができます。
- 複合問合せ（集合演算子 UNION、INTERSECT、MINUS または UNION ALL を含む）では、ORDER BY 句に明示的な式ではなく、位置を使用する必要があります。また ORDER BY 句は、最後のコンポーネントの間合せにのみ使用できます。ORDER BY 句は、複合問合せ全体によって戻されたすべての行を順序付けます。

ORDER BY 句による値のソートは、NLS_SORT 初期化パラメータによって明示的に指定するか、NLS_LANGUAGE 初期化パラメータによって暗黙的に指定します。ALTER SESSION 文を使用して、1 つの言語ソート順序から別の言語ソート順序に変更することができます。ORDER BY 句の NLS_SORT パラメータと NLSSORT ファンクションを使用して、1 つの間合せに特定のソート基準を指定することもできます。

参照： NLS パラメータの詳細は、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

結合

結合とは、2 つ以上の表、ビューまたはマテリアライズド・ビューの行を結合する問合せです。複数の表が問合せの FROM 句に指定される場合、Oracle は結合を実行します。問合せの SELECT 構文のリストは、これらの表のいずれかの任意の列を選択することができます。これらの表のいずれか 2 つに共通の列名を持つものがある場合、問合せの間、これらの列に対してすべての参照を明確にするために表の名前を付けて修飾する必要があります。

結合条件

結合問合せの大部分は、表ごとに異なる 2 つの列を比較する WHERE 句条件を含んでいます。このような条件を**結合条件**と呼びます。結合を実行するために、Oracle は各表に 1 つずつ含まれている列を結合し、結合条件が TRUE になるようにします。結合条件の列を SELECT 構文のリストに表示する必要はありません。

3 つ以上の表を結合するために、Oracle はまず列を比較する結合条件に基づいて 2 つの表を結合し、結合された表と新規の表の列を含む結合条件に基づいて、さらにもう 1 つの表を結合します。すべての表が結果に結合されるまで、このプロセスを継続します。オプティマイザは、Oracle が結合条件に基づいて表を結合する順序、表の索引、およびコストベースの最適化アプローチを行う場合の表の統計を決定します。

結合条件の他に、結合問合せの WHERE 句にも、1 つの表のみの列を示す他の条件を指定することができます。これらの条件は、結合問合せによって戻された列をさらに制限することができます。

注意： 結合を含む WHERE 句には、LOB 列を指定できません。WHERE 句での LOB の使用については、他にも制限事項があります。詳細は、『Oracle9i アプリケーション開発者ガイド - ラージ・オブジェクト』を参照してください。

等価結合

等価結合とは、等価演算子を含む結合条件での結合のことです。等価結合は、指定した列に同等の値を持つ列を結合します。オプティマイザが結合の実行を選択する内部アルゴリズムによって、1 つの表の等価結合条件における列の合計サイズは、データ・ブロックのサイズ以下に制限される可能性があります。データ・ブロックのサイズは、初期化パラメータ DB_BLOCK_SIZE によって指定されます。

参照： 17-31 ページの「[結合の例](#)」を参照してください。

自己結合

自己結合とは、自己の表結合のことです。この表は FROM 句に 2 回指定され、結合条件の列名を修飾する表の別名が続きます。自己結合を実行するために、Oracle は結合条件を満たす表の行を結合して戻します。

参照： 17-32 ページの「[自己結合の例](#)」を参照してください。

デカルト演算

結合問合せの 2 つの表に結合条件がない場合、Oracle は**デカルト演算**を戻します。1 つの表の各列を別の表の各行に結合します。デカルト演算は常に多数の行を生成するため、非常に有効です。たとえば、それぞれが 100 行を持つ 2 つの表のデカルト演算は 10,000 行です。特にデカルト演算を必要としないかぎり、常に結合条件を含みます。問合せが 3 つ以上の表を結合し、特定の組に対して結合条件を指定しない場合、オプティマイザは、中間のデカルト演算を生成しないように結合順序を選択する可能性があります。

内部結合

内部結合（単純結合）とは、結合条件を満たす行のみを戻す、複数の表の結合です。

外部結合

外部結合は、単純結合の結果を拡張します。外部結合は、結合条件を満たすすべての行と、ある特定の表の結合条件を満たす行を除いたすべての行を戻します。

- 表 A および B の外部結合を行い、A からすべての行を戻す問合せを書き込む（**左側外部結合**）ために、ANSI の LEFT [OUTER] JOIN 構文を使用するか、結合条件で外部結合演算子 (+) を B のすべての列に適用します。B に一致する行のない A のすべての行に関して、Oracle は、B の列を含む任意の SELECT 構文のリストの式に NULL を戻します。
- 表 A および B の外部結合を行い、B からすべての行を戻す問合せを書き込む（**右側外部結合**）ために、ANSI の RIGHT [OUTER] JOIN 構文を使用するか、結合条件で外部結合演算子 (+) を A のすべての列に適用します。A に一致する行のない B のすべての行に関して、Oracle は、A の列を含む任意の SELECT 構文のリストの式に NULL を戻します。
- 外部結合を行い、結合条件を満たさない場合に、NULL で拡張する A および B からすべての行を戻す問合せを書き込む（**完全な外部結合**）ために、ANSI の FULL [OUTER] JOIN 構文を使用します。

Oracle の結合演算子よりも ANSI の OUTER JOIN 構文を使用することをお勧めします。ANSI 構文では適用されない、Oracle 結合演算子 (+) を使用した外部結合問合せの規則および制限事項は、次のとおりです。

- ANSI JOIN 構文を含む問合せブロックで、(+) 演算子を指定できません。
- (+) 演算子は、WHERE 句または FROM 句の左相関のコンテキスト（TABLE 句を指定する場合）にのみ指定でき、表またはビューの列にのみ適用されます。
- A および B が複数の結合条件によって結合される場合、これらの条件のすべてにおいて (+) 演算子を使用する必要があります。使用しない場合、Oracle は単純結合の結果である行のみを戻しますが、外部結合の結果がないことを示す警告やエラーは出力しません。
- (+) 演算子は任意の式ではなく、列にのみ適用することができます。ただし、任意の式には (+) 演算子でマークされた列を含めることができます。

- (+) 演算子を含む条件は、OR 論理演算子を使用する他の条件と結合できません。
- 条件は、IN 比較条件を使用して、(+) 演算子でマークされた列と式を比較できません。
- 条件は、(+) 演算子でマークされた列を副問合せと比較できません。

WHERE 句に表 B の列と定数を比較する条件が含まれる場合、Oracle がこの列に対して NULL を生成する表 A の列を戻すように、(+) 演算子をこの列に適用する必要があります。それ以外の場合、Oracle は単純結合の結果のみを戻します。

2 組以上の表の外部結合を行う問合せにおいて、単一表は他の 1 つの表のみに対して NULL 生成された表になることができます。そのため、A と B の結合条件および B と C の結合条件における B の列に、(+) 演算子を適用することはできません。

参照： 外部結合の構文については、17-4 ページの「[SELECT](#)」を参照してください。

副問合せの使用

副問合せは、複数部分の問合せに応答します。たとえば、Taylor の部門で働いている人を判断するには、まず Taylor が働く部門を判断する副問合せを使用できます。その後、親 SELECT 文で元の問合せに応答することができます。SELECT 文の FROM 句の副問合せは、**インライン・ビュー**とも呼ばれます。また、SELECT 文の WHERE 句の副問合せは、**ネストした副問合せ**とも呼ばれます。

副問合せには、別の副問合せを含むことができます。トップレベル問合せの FROM 句内の副問合せレベルの数には、制限がありません。WHERE 句には、最大 255 レベルの副問合せをネストできます。

副問合せにある列が、含まれる文の列と同じ名前を持つ場合、含まれる文の表の列に表名または別名で参照の接頭辞を付ける必要があります。文をさらに読みやすくするには、常に、表、ビューまたはマテリアライズド・ビューの名前または別名で副問合せの列を修飾します。

副問合せが親である文で参照する表の列を参照する場合、Oracle は**相関副問合せ**を行います。相関副問合せは、親である文によって処理された各列を 1 回評価します。親である文は、SELECT、UPDATE または DELETE 文のいずれかです。

相関副問合せは、応答が親である文によって処理された各列の値に依存する問合せに応答します。たとえば、相関副問合せを使用して、部門内で給与が平均給与以上の従業員を判断することができます。この場合、相関副問合せは独自で各部門の平均給与を計算します。

参照： 17-39 ページの「[相関副問合せの例](#)」を参照してください。

副問合せは、次の用途に使用します。

- INSERT または CREATE TABLE 文のターゲット表に挿入する一連の列を定義します。
- CREATE VIEW または CREATE MATERIALIZED VIEW 文のビューまたはマテリアライズド・ビューに含める一連の列を定義します。
- UPDATE 文の既存の列に割り当てる 1 つ以上の値を定義します。
- SELECT、UPDATE および DELETE 文の WHERE 句、HAVING 句または START WITH 句における条件に対する値を定義します。
- 含まれる問合せによって操作される表を定義します。

表名を指定する場合と同様に、問合せを含む FROM 句に副問合せを指定することによってこれらのことを行います。INSERT、UPDATE および DELETE 文においても、このようにして表のかわりに副問合せを使用することができます。

使用された副問合せは、外部参照ではなく、副問合せ内で定義された相関変数のみを使用することができます。外部参照（左相関副問合せ）は、SELECT 文の FROM 句でのみ使用できます。

参照： 17-15 ページの「[table_collection_expression](#)」を参照してください。

1 つの行から 1 つの列の値を戻すスカラー副問合せが、式の有効な書式です。構文の *expr* をコールするほとんどの位置で、スカラー副問合せ式を使用できます。

参照： 4-13 ページの「[スカラー副問合せ式](#)」を参照してください。

ネストされた副問合せのネスト解除

副問合せは、親である文の WHERE 句内にあるときはネストされています。ネストされた副問合せを持つ文を評価する場合、Oracle は、副問合せ部分を複数回評価する必要があり、効果的なアクセス・パスまたは結合を見逃してしまう可能性があります。

副問合せのネスト解除によって、副問合せの本体がネスト解除され、その副問合せを含む文の本体に結合されます。これによって、アクセス・パスおよび結合の評価時に、オブティマイザが副問合せと文を 1 つのものと判断します。オブティマイザは、ほぼすべての副問合せをネスト解除できますが、いくつか例外があります。これらの例外には、階層副問合せおよび ROWNUM 疑似列、集合演算子の 1 つ、ネストした集計ファンクション、または副問合せの直接的な外部問合せブロックではない問合せへの相関参照が含まれます。

制約がない場合、オブティマイザは、次のネストされた副問合せを自動的にネスト解除します（ただし、ネスト解除しない場合もあります）。

- 相関関係のない IN 副問合せ
- IN および EXISTS 相関副問合せ（集計ファンクションまたは GROUP BY 句を含まない場合）

ネスト解除された拡張副問合せを行うには、次のタイプの副問合せをネスト解除するようにオブティマイザに指示します。

- 副問合せに HASH_AJ または MERGE_AJ ヒントを指定して、NOT IN 副問合せをネスト解除します。
- 副問合せに UNNEST ヒントを指定して、その他の副問合せをネスト解除します。

参照： ヒントの詳細は、[第 2 章「Oracle SQL の基本要素」](#)を参照してください。

DUAL 表からの選択

DUAL は、データ・ディクショナリとともに Oracle によって自動的に作成された表です。DUAL は、ユーザー SYS のスキーマにありますが、すべてのユーザーが DUAL という名前でアクセスすることができます。DUAL は、VARCHAR2(1) として定義されている DUMMY 列を持ち、「X」値を持つ行を含みます。DUAL 表から選択することは、定数式を SELECT 文で計算する場合に便利です。DUAL には 1 行以外存在しないため、定数は 1 回のみ戻されます。一方で、任意の表から定数、疑似列または式を選択できますが、値は表の行の数のみ戻されます。

参照： DUAL から定数値を選択する例は、6-2 ページの「[SQL ファンクション](#)」を参照してください。

分散問合せ

Oracle の分散データベース・システム・アーキテクチャによって、Oracle Net および Oracle サーバーを使用するリモート・データベースにアクセスできます。名前の最後に *@dblink* を追加して、リモート表、ビューまたはマテリアライズド・ビューを識別できます。*dblink* は、リモート表、ビューまたはマテリアライズド・ビューを含むデータベースへのデータベース・リンクの完全な名前または部分的な名前である必要があります。

参照：

- データベース・リンクの参照方法の詳細は、2-113 ページの「[リモート・データベース内のオブジェクトの参照](#)」を参照してください。
- リモート・データベースへのアクセスについては、『Oracle9i Net Services 管理者ガイド』を参照してください。

分散問合せの制限

分散問合せには、現在、FOR UPDATE 句によってロックされたすべての表、および問合せによって選択された LONG 列を持つすべての表が、同じデータベース上に位置している必要があります。たとえば、次の文はエラーになります。

```
SELECT employees_ny.*
FROM employees_ny@ny, departments
WHERE employees_ny.department_id = departments.department_id
AND departments.department_name = 'ACCOUNTING'
FOR UPDATE OF employees_ny.salary;
```

次の文は、ny データベースの employees_review 表から LONG 値である long_column を選択し、ローカル・データベースの employees 表をロックするため、エラーになります。

```
SELECT employees.employee_id, review.long_column, employees.salary
FROM employees, employees_review@ny review
WHERE employees.employee_id = employees_review.employee_id
FOR UPDATE OF employees.salary;
```

また、Oracle は現在、リモート表にあるユーザー定義型またはオブジェクト REF を選択する分散問合せをサポートしていません。

様々な種類の SQL 文

次の項にある表は、SQL 文の機能の概要について、次のカテゴリに分類して説明しています。

- データ定義言語（DDL）文
- データ操作言語（DML）文
- トランザクション制御文
- セッション制御文
- システム制御文

データ定義言語（DDL）文

データ定義言語（DDL）文によって、次のタスクを実行できます。

- スキーマ・オブジェクトの作成、変更および削除
- 権限およびロールの付与および取消し
- 表、索引またはクラスタ上の情報の分析
- 監査オプションの構築
- データ・ディクショナリへのコメントの追加

CREATE、ALTER および DROP コマンドは、特定のオブジェクトに対して排他的アクセスを必要とします。たとえば、別のユーザーが特定の表でトランザクションをオープンしている場合、ALTER TABLE 文は実行できません。

GRANT、REVOKE、ANALYZE、AUDIT および COMMENT コマンドは、特定のオブジェクトに対する排他的アクセスを必要としません。たとえば、他のユーザーが表を更新しているときでも、その表を分析できます。

Oracle は、暗黙的にすべての DDL 文の前後で現在のトランザクションをコミットします。

DDL 文の多くは、Oracle にスキーマ・オブジェクトを再コンパイルまたは再認可させることができます。Oracle がスキーマ・オブジェクトを再コンパイルまたは再認可する方法、および DDL 文によってそれを実行する環境については、『Oracle9i データベース概要』を参照してください。

DDL 文は、DBMS_SQL パッケージを使用した PL/SQL によってサポートされます。

参照：『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

表 7-1 に、DDL 文のリストを示します。

表 7-1 データ定義言語文

ALTER CLUSTER	CREATE DIMENSION	DROP CLUSTER
ALTER DATABASE	CREATE DIRECTORY	DROP CONTEXT
ALTER DIMENSION	CREATE FUNCTION	DROP DATABASE LINK
ALTER FUNCTION	CREATE INDEX	DROP DIMENSION
ALTER INDEX	CREATE INDEXTYPE	DROP DIRECTORY
ALTER MATERIALIZED VIEW	CREATE LIBRARY	DROP FUNCTION
ALTER MATERIALIZED VIEW LOG	CREATE MATERIALIZED VIEW	DROP INDEX
ALTER PACKAGE	CREATE MATERIALIZED VIEW LOG	DROP INDEXTYPE
ALTER PROCEDURE	CREATE OPERATOR	DROP LIBRARY
ALTER PROFILE	CREATE PACKAGE	DROP MATERIALIZED VIEW
ALTER RESOURCE COST	CREATE PACKAGE BODY	DROP MATERIALIZED VIEW LOG
ALTER ROLE	CREATE PFILE	DROP OPERATOR
ALTER ROLLBACK SEGMENT	CREATE PROCEDURE	DROP PACKAGE
ALTER SEQUENCE	CREATE PROFILE	DROP PROCEDURE
ALTER TABLE	CREATE ROLE	DROP PROFILE
ALTER TABLESPACE	CREATE ROLLBACK SEGMENT	DROP ROLE
ALTER TRIGGER	CREATE SCHEMA	DROP ROLLBACK SEGMENT
ALTER TYPE	CREATE SEQUENCE	DROP SEQUENCE
ALTER USER	CREATE SYNONYM	DROP SYNONYM
ALTER VIEW	CREATE SPFILE	DROP TABLE
ANALYZE	CREATE TABLE	DROP TABLESPACE
ASSOCIATE STATISTICS	CREATE TABLESPACE	DROP TRIGGER
AUDIT	CREATE TEMPORARY TABLESPACE	DROP TYPE
COMMENT	CREATE TRIGGER	DROP USER
CREATE CLUSTER	CREATE TYPE	DROP VIEW
CREATE CONTEXT	CREATE USER	GRANT
CREATE CONTROLFILE	CREATE VIEW	NOAUDIT
CREATE DATABASE	DISASSOCIATE STATISTICS	RENAME
CREATE DATABASE LINK		REVOKE
		TRUNCATE

データ操作言語（DML）文

データ操作言語（DML）文は、既存スキーマ・オブジェクトのデータを問い合わせ、操作します。次の文は、現在のトランザクションを暗黙的にコミットしません。

表 7-2 データ操作言語文

文
CALL
DELETE
EXPLAIN PLAN
INSERT
LOCK TABLE
MERGE
SELECT
UPDATE

CALL および EXPLAIN PLAN 文は、動的に実行されるときにのみ PL/SQL でサポートされます。他のすべての DML 文は、PL/SQL で完全にサポートされます。

トランザクション制御文

トランザクション制御文は、DML 文で行った変更を管理します。

表 7-3 トランザクション制御文

文
COMMIT
ROLLBACK
SAVEPOINT
SET TRANSACTION

COMMIT および ROLLBACK コマンドの特定書式以外のトランザクション制御文は、PL/SQL でサポートされます。制限については、11-69 ページの「COMMIT」および 16-96 ページの「ROLLBACK」を参照してください。

セッション制御文

セッション制御文は、ユーザー・セッションのプロパティを動的に管理します。次の文は、現在のトランザクションを暗黙的にコミットしません。

PL/SQL は、セッション制御文をサポートしません。

表 7-4 セッション制御文

文
ALTER SESSION
SET ROLE

システム制御文

単一システム制御文は、Oracle インスタンスのプロパティを動的に管理します。次の文は、現在のトランザクションを暗黙的にコミットしません。

ALTER SYSTEM は、PL/SQL でサポートされません。

表 7-5 システム制御文

文
ALTER SYSTEM

埋込み SQL 文

埋込み SQL 文は、DDL、DML およびトランザクション制御文を手続き型言語プログラム内に入れます。埋込み SQL は、Oracle プリコンパイラでサポートされており、次のマニュアルに記載されています。

- 『Pro*COBOL Precompiler プログラマーズ・ガイド』
- 『Pro*C/C++ Precompiler プログラマーズ・ガイド』
- 『Programmer's Guide to SQL*Module for Ada』

SQL 文 : ALTER CLUSTER ～ ALTER SEQUENCE

この章および第 9 章～第 17 章のすべての SQL 文は、次の項で編成されています。

構文	構文図には、文を構成するキーワードおよびパラメータを示します。 注意： すべてのキーワードおよびパラメータがあらゆる環境において有効なわけではありません。構文の制限事項については、文および句の「キーワードとパラメータ」を参照してください。
用途	文の基本的な使用方法を説明します。
前提条件	文の実行に必要な権限と、文を使用する前に実行する手順を示します。特に指定がないかぎり、ご使用のインスタンスでデータベースがオープンされている必要があります。
キーワードとパラメータ	キーワードおよびパラメータの用途を説明します。（この章および第 9 章～第 17 章で使用するキーワードおよびパラメータの表記規則の詳細は、「はじめに」を参照してください。）制限事項および使用時の注意事項についてもこの項で説明します。
例	文の様々な句およびパラメータの使用方法を示します。

この章では、次の SQL 文について説明します。

- ALTER CLUSTER
- ALTER DATABASE
- ALTER DIMENSION
- ALTER FUNCTION
- ALTER INDEX
- ALTER INDEXTYPE
- ALTER JAVA
- ALTER MATERIALIZED VIEW
- ALTER MATERIALIZED VIEW LOG
- ALTER OUTLINE
- ALTER PACKAGE
- ALTER PROCEDURE
- ALTER PROFILE
- ALTER RESOURCE COST
- ALTER ROLE
- ALTER ROLLBACK SEGMENT
- ALTER SEQUENCE

ALTER CLUSTER

用途

ALTER CLUSTER 文を使用すると、クラスタの記憶特性および並列特性を再定義できます。

注意： クラスタ・キーの列番号および列名を変更するためにこの文を使用することはできません。また、クラスタを格納する表領域を変更することはできません。

参照：

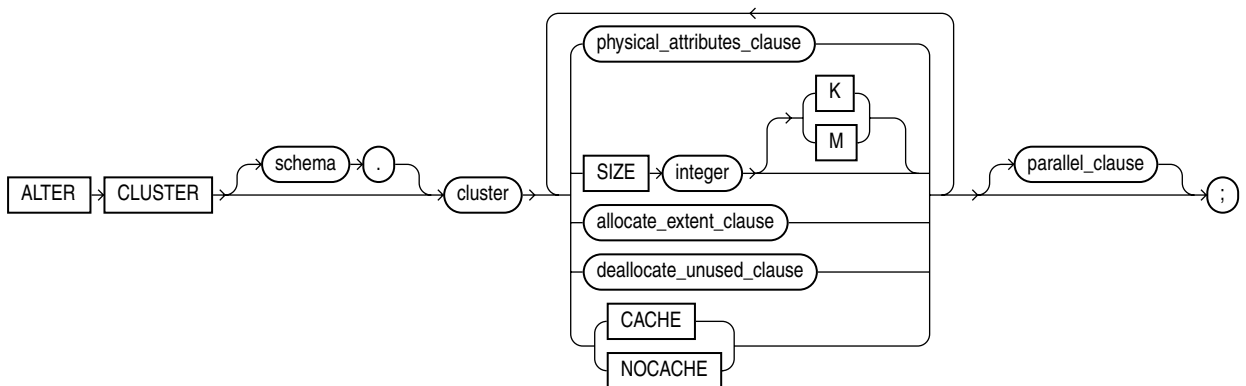
- クラスタの作成については、12-2 ページの「[CREATE CLUSTER](#)」を参照してください。
- クラスタからの表の削除については、15-60 ページの「[DROP CLUSTER](#)」および 16-6 ページの「[DROP TABLE](#)」を参照してください。

前提条件

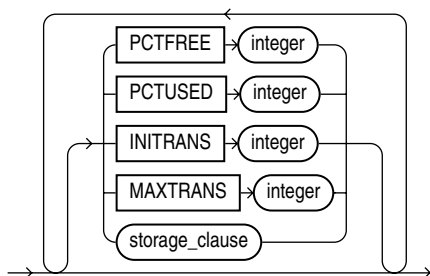
クラスタが自分のスキーマ内にあるか、または ALTER ANY CLUSTER システム権限を持っている必要があります。

構文

alter_cluster::=

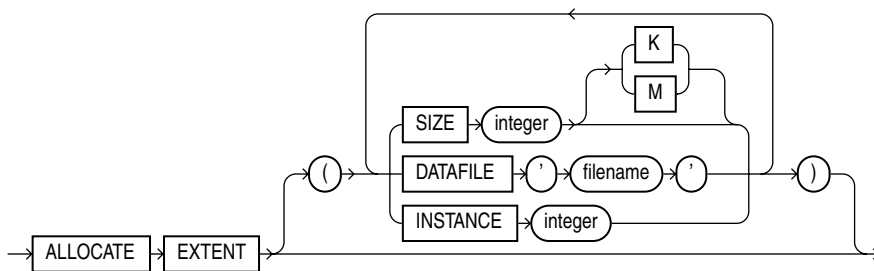


physical_attributes_clause::=

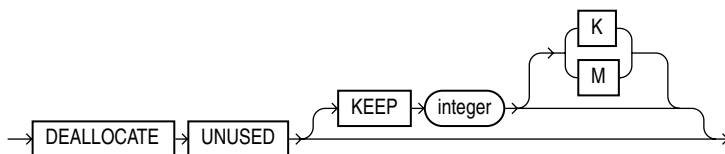


storage_clause: 17-50 ページの「[storage_clause](#)」を参照してください。

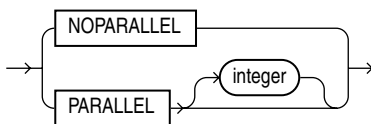
allocate_extent_clause::=



deallocate_unused_clause::=



parallel_clause::=



キーワードとパラメータ

schema

クラスタが含まれているスキーマを指定します。*schema* を指定しない場合、そのクラスタが自分のスキーマにあるとみなされます。

cluster

変更するクラスタの名前を指定します。

physical_attributes_clause

クラスタの PCTUSED パラメータ、PCTFREE パラメータ、INITTRANS パラメータおよび MAXTRANS パラメータの値を変更します。

クラスタの記憶特性を変更するには、**STORAGE** 句を使用します。

制限事項：クラスタの記憶域パラメータ **INITIAL** および **MINEXTENTS** の値は変更できません。

参照：

- これらのパラメータについては、12-2 ページの「[CREATE CLUSTER](#)」を参照してください。
- 17-50 ページの「[storage_clause](#)」を参照してください。

SIZE integer

SIZE 句を使用すると、クラスタに割り当てられたデータ・ブロック中に格納されるクラスタ・キーの数を指定できます。

制限事項：ハッシュ・クラスタではなく、索引クラスタの **SIZE** パラメータのみを変更できます。

参照： **SIZE** パラメータについては、12-2 ページの「[CREATE CLUSTER](#)」を参照してください。

allocate_extent_clause

ALLOCATE EXTENT 句を指定すると、クラスタの新しいエクステンツを明示的に割り当てます。

制限事項：ハッシュ・クラスタではなく、索引クラスタのみに新しいエクステンツを割り当てることができます。

SIZE SIZE パラメータを使用すると、エクステンツのサイズをバイト単位で指定できます。K または M を使用すると、エクステンツ・サイズを KB または MB 単位で指定できます。

この句で明示的にエクステンツを割り当てる場合、Oracle は、クラスツの記憶域パラメータを評価しません。割り当てられる新しいエクステンツの新しいサイズも決定しません（表を作成する際に行います）。したがって、Oracle がデフォルト値を使用しないようにするには、SIZE を指定してください。

DATAFILE DATAFILE パラメータを使用すると、クラスツの表領域内で、新しいエクステンツを含むデータ・ファイルを 1 つ指定できます。このパラメータを省略した場合、データ・ファイルは Oracle によって選択されます。

INSTANCE INSTANCE パラメータを使用すると、指定したインスタンスで新しいエクステンツが使用可能になります。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。INSTANCE を指定しないと、すべてのインスタンスで新しいエクステンツが使用可能になります。

注意： Real Application Clusters 環境で Oracle を使用する場合のみ、このパラメータを使用します。

deallocate_unused_clause

DEALLOCATE UNUSED 句を指定すると、使用クラスツの終わりの未使用領域の割当てを明示的に解除し、解放された領域が他のセグメントで使用可能になります。ただし、解放できるのは、最高水位標を超える未使用領域のみです。

KEEP KEEP パラメータを使用すると、割当てを解除した後にクラスツに残す、最高水位標を超えるバイト数を指定できます。残りのエクステンツ数が MINEXTENTS より少ない場合、MINEXTENTS は現行のエクステンツ数に設定されます。初期エクステンツが INITIAL より小さくなると、INITIAL は初期エクステンツの現行の値に設定されます。KEEP を指定しないと、すべての未使用領域が解放されます。

参照： この句の詳細は、10-2 ページの「ALTER TABLE」を参照してください。

CACHE | NOCACHE

CACHE フル・テーブル・スキャンの実行時に、このクラスタに対して取り出されたブロックを、バッファ・キャッシュ内の最低使用頻度（LRU）リストの最高使用頻度側に入れる場合は、CACHE を指定します。この句は、小規模な参照表で有効です。

NOCACHE フル・テーブル・スキャンの実行時に、このクラスタに対して取り出されたブロックを、バッファ・キャッシュ内の LRU リストの最低使用頻度側に入れる場合は、NOCACHE を指定します。これはデフォルトの動作です。

parallel_clause

parallel_clause を指定すると、クラスタの DML および問合せのデフォルト並列度を変更できます。

制限事項：クラスタの表が LOB 型またはユーザー定義オブジェクト型の列を含む場合、クラスタでその後に行う INSERT、UPDATE または DELETE 操作と同様、この文は通知なしに逐次実行されます。

注意：*parallel_clause* 構文は、以前のリリースの構文に代わるものです。以前のリリースの構文は下位互換用にサポートされていますが、動作がわずかに異なることがあります。

NOPARALLEL シリアル実行を行う場合は、NOPARALLEL を指定します。これはデフォルトです。

PARALLEL 並列度を選択する場合は、PARALLEL を指定します。並列度とは、すべての関係するインスタンスで使用可能な CPU の数に、初期化パラメータ PARALLEL_THREADS_PER_CPU の値を掛けたものです。

PARALLEL integer *integer* にはパラレル操作で使用するパラレル・スレッド数である**並列度**を指定します。各パラレル・スレッドは、1、2 個のパラレル実行サーバーを使用します。通常、最適な並列度が計算されるため、*integer* に値を指定する必要はありません。

参照： 詳細は、14-43 ページの「CREATETABLE」の「[parallel_clause](#)に関する注意事項」を参照してください。

例

次に、12-9 ページの「CREATE CLUSTER」の「例」で作成したクラスタを変更する例を示します。

クラスタの変更例 次の文は、personnel クラスタを変更します。

```
ALTER CLUSTER personnel
  SIZE 1024
  CACHE;
```

この結果、各クラスタ・キー値に 1024 バイトが割り当てられ、CACHE 属性が指定されます。データ・ブロックのサイズを 2KB と想定した場合、このクラスタ内の今後のデータ・ブロックには、各ブロックに 2 つのクラスタ・キー（2KB を 1024 バイトで割った値）が含まれます。

未使用領域の解放例 次の文は、language クラスタから未使用領域の割当てを解除し、後で使用できるように 30KB の未使用領域を保持します。

```
ALTER CLUSTER language
  DEALLOCATE UNUSED KEEP 30 K;
```


ALTER DATABASE

用途

ALTER DATABASE 文を使用すると、既存のデータベースを変更、メンテナンスまたはリカバリできます。

参照：

- データベースをメンテナンスするために ALTER DATABASE 文を使用する場合の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。
- メディア・リカバリの実行例については、『Oracle9i データベース管理者ガイド』、『Oracle9i ユーザー管理バックアップおよびリカバリ・ガイド』、『Oracle9i Recovery Manager ユーザーズ・ガイド』および『Oracle9i Recovery Manager リファレンス』を参照してください。
- データベース作成の詳細は、12-20 ページの「[CREATE DATABASE](#)」を参照してください。

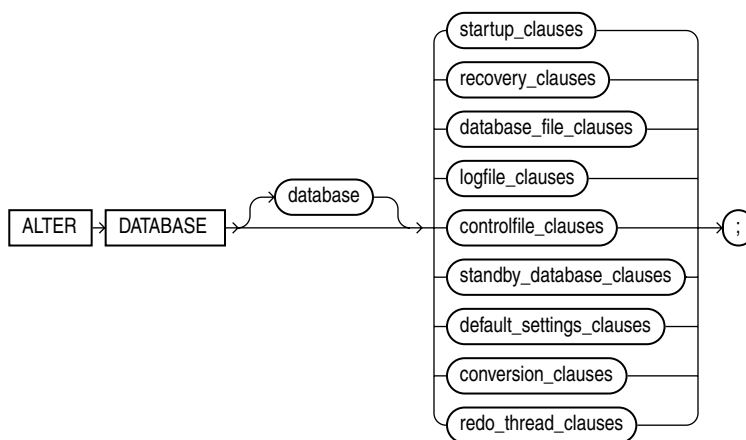
前提条件

ALTER DATABASE システム権限が必要です。

RECOVER 句を指定する場合は、SYSDBA システム権限が必要です。

構文

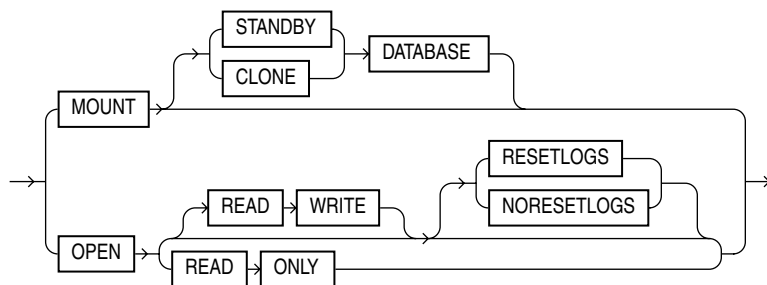
alter_database::=



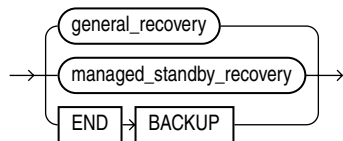
ALTER DATABASE 構文のグループは、次のとおりです。

- **startup_clauses::=** (8-10 ページ)
- **recovery_clauses::=** (8-10 ページ)
- **database_file_clauses::=** (8-13 ページ)
- **logfile_clauses::=** (8-15 ページ)
- **controlfile_clauses::=** (8-16 ページ)
- **standby_database_clauses::=** (8-16 ページ)
- **default_settings_clauses::=** (8-17 ページ)
- **conversion_clauses::=** (8-17 ページ)
- **redo_thread_clauses::=** (8-17 ページ)

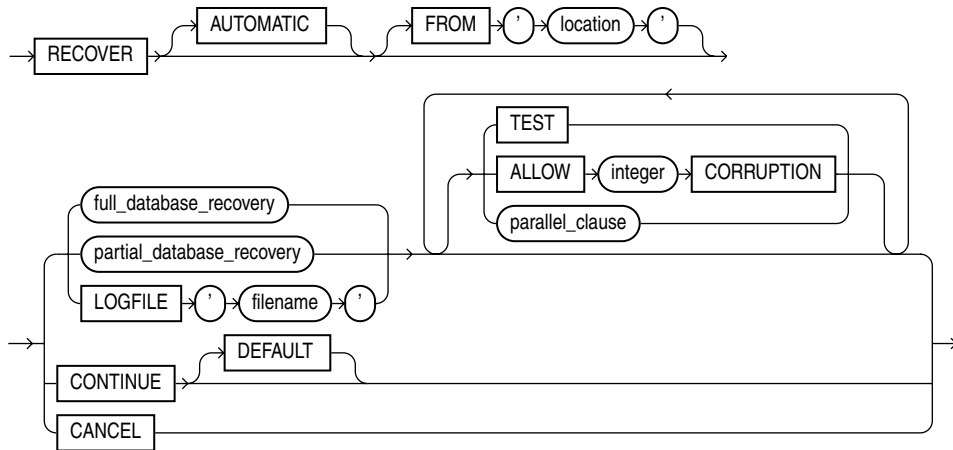
startup_clauses::=



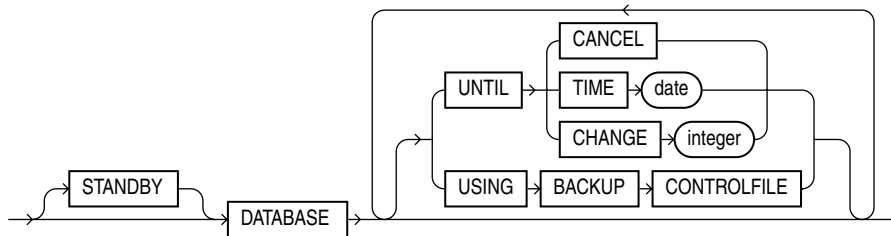
recovery_clauses::=



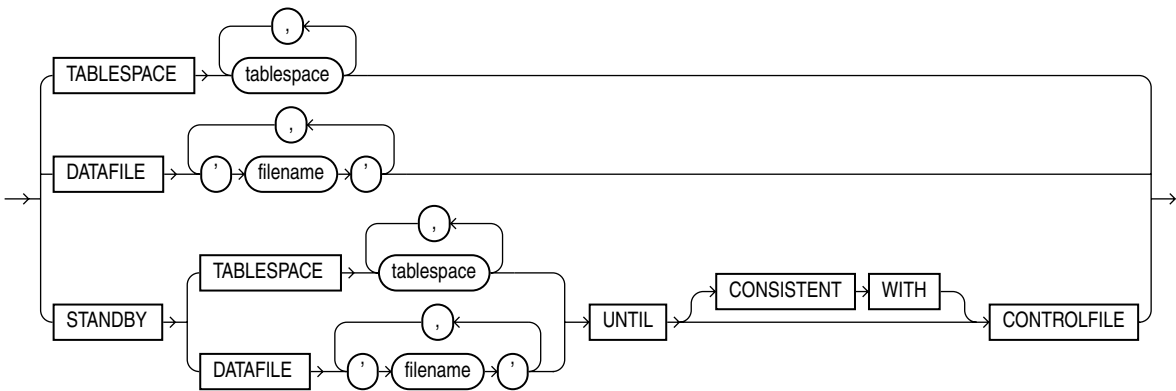
general_recovery::=



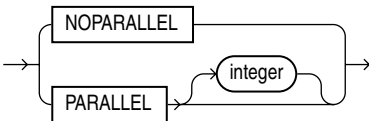
full_database_recovery::=



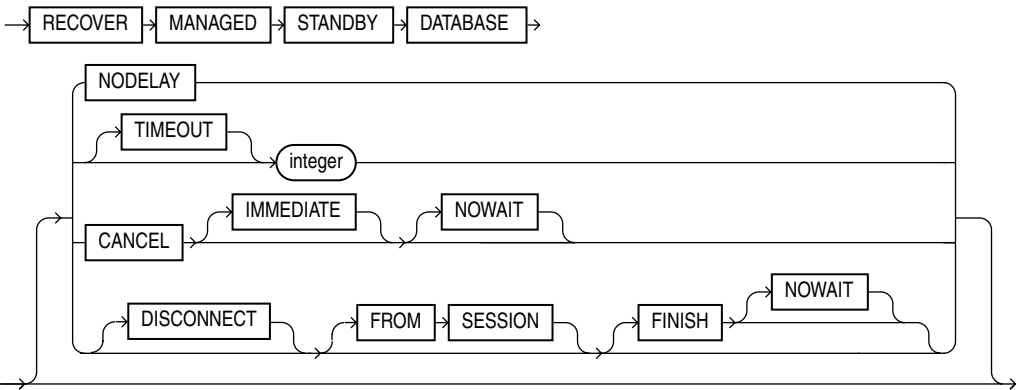
partial_database_recovery::=



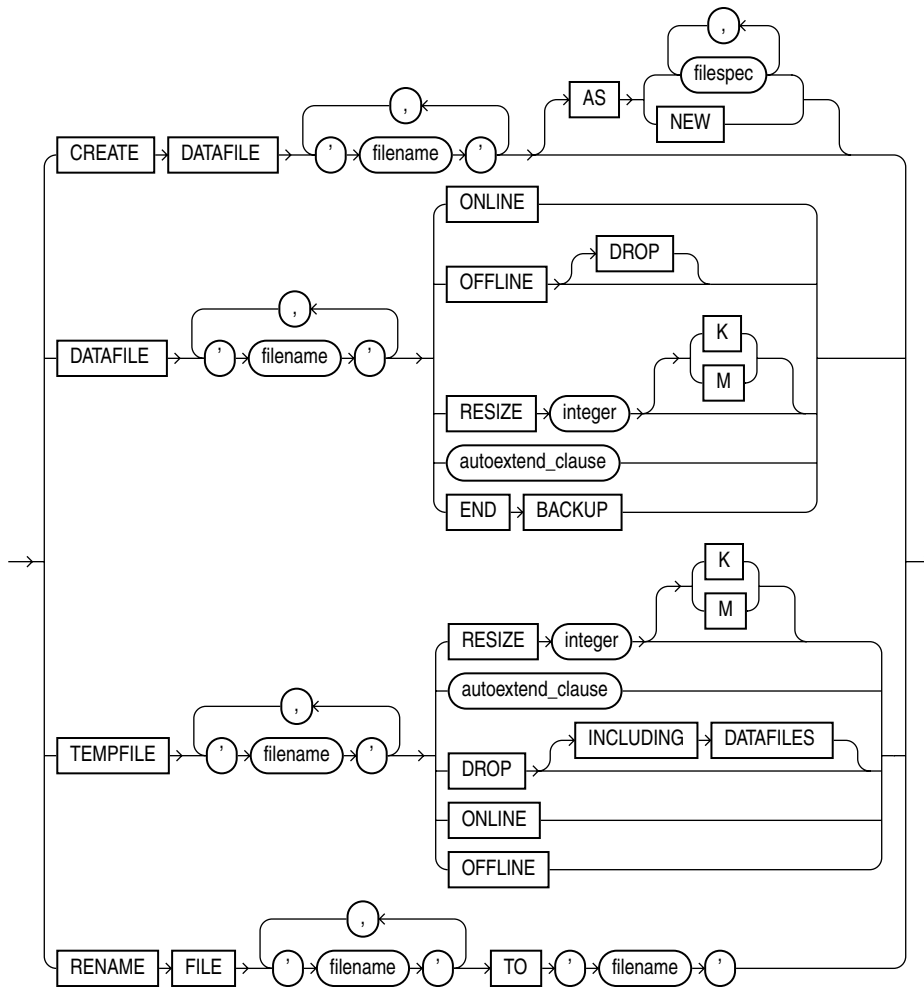
parallel_clause::=



managed_standby_recovery::=

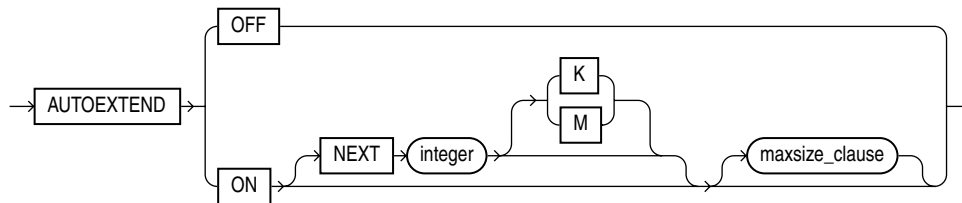


database_file_clauses::=

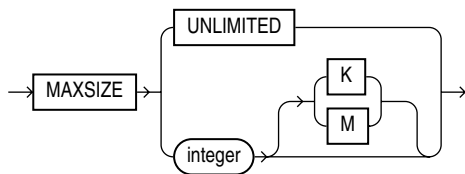


filespec: 16-27 ページの「[filespec](#)」を参照してください。

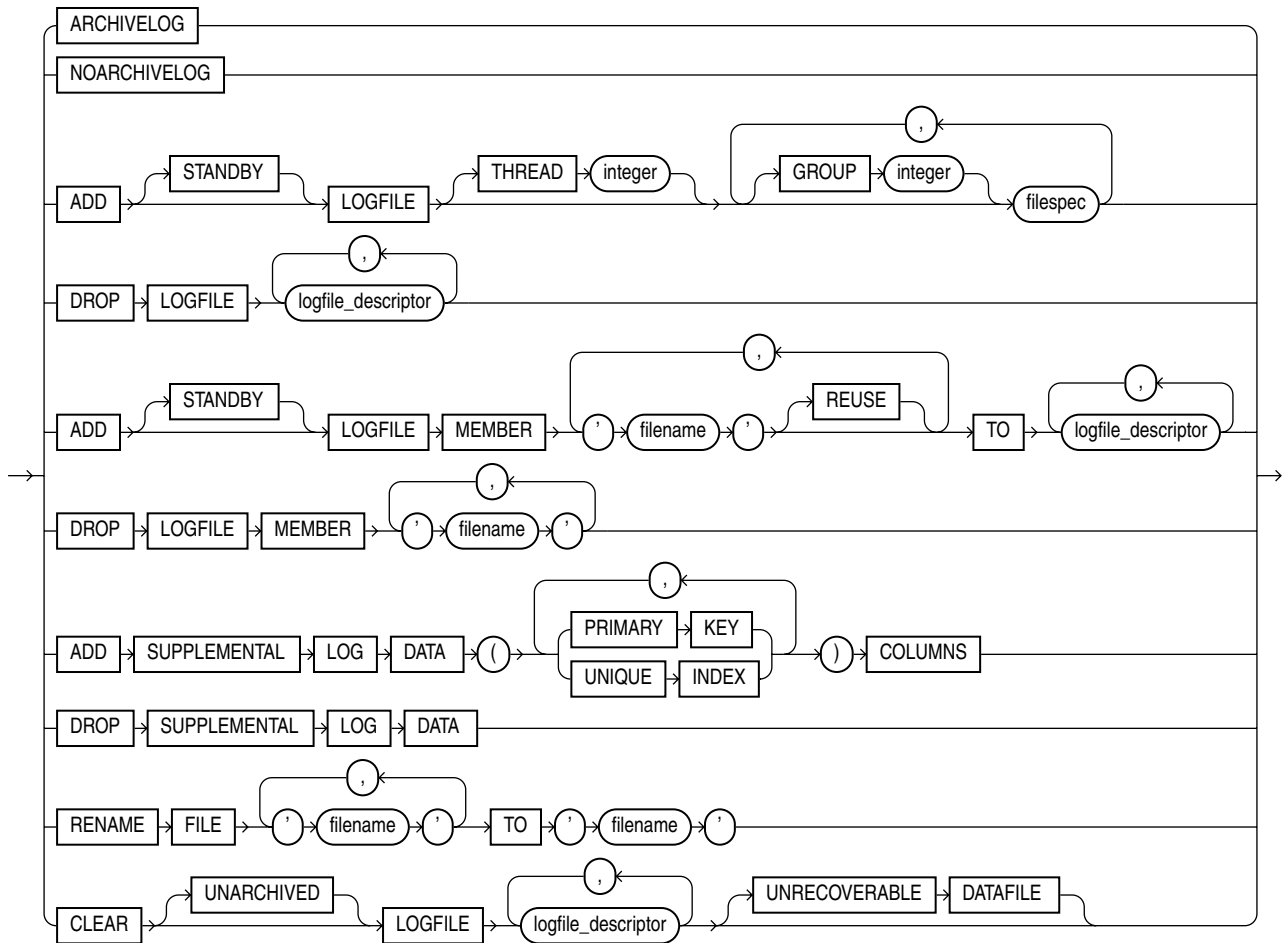
autoextend_clause::=



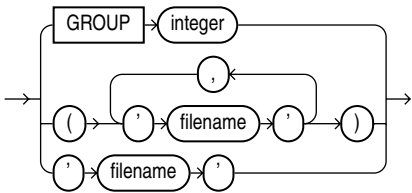
maxsize_clause::=



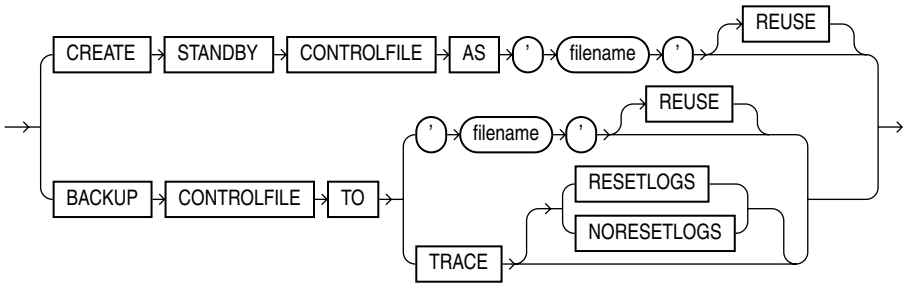
logfile_clauses::=



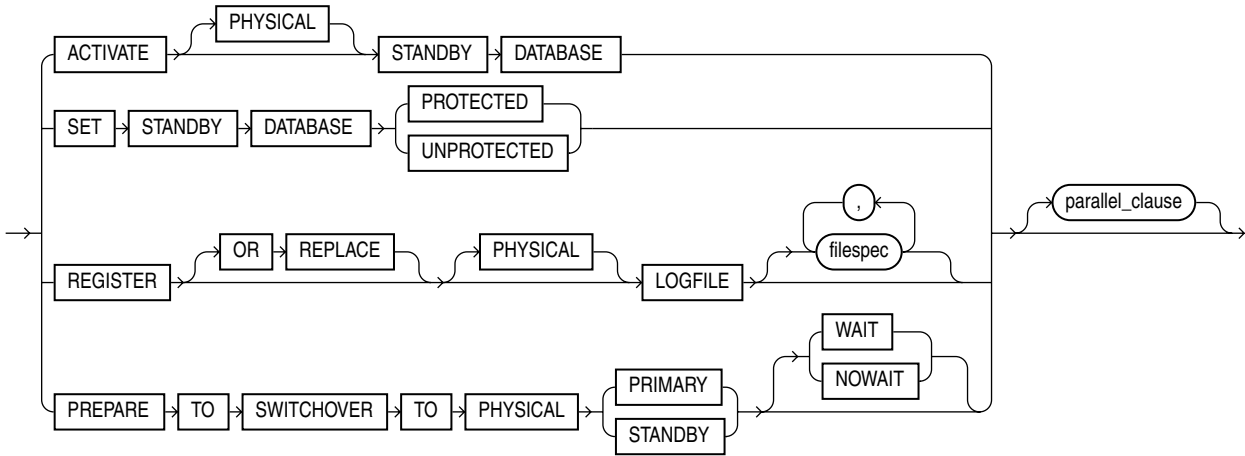
logfile_descriptor::=



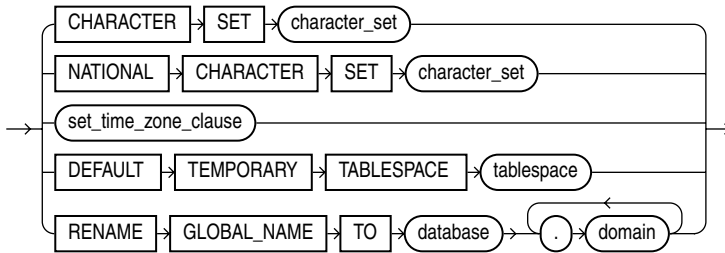
controlfile_clauses::=



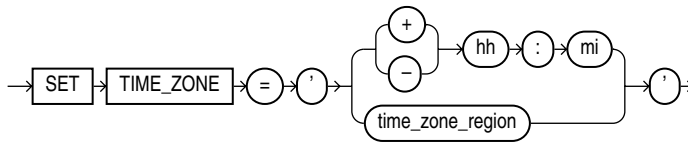
standby_database_clauses::=



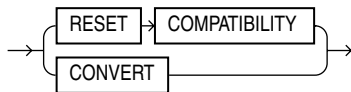
default_settings_clauses::=



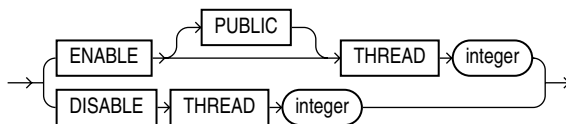
set_time_zone_clause::=



conversion_clauses::=



redo_thread_clauses::=



キーワードとパラメータ

database

変更するデータベースの名前を指定します。データベース名には ASCII 文字のみが使用できます。データベース名を指定しないと、初期化パラメータ `DB_NAME` に指定されているデータベースが変更されます。なお、データベースの制御ファイルが初期化パラメータ `CONTROL_FILES` に指定されている場合にのみ、そのデータベースを変更できます。データベース識別子は、Oracle Net のデータベース指定とは関係ありません。

startup_clauses

startup_clauses を使用すると、データベースがマウントおよびオープンされ、ユーザーからのアクセスが可能になります。

MOUNT 句

MOUNT 句を使用すると、データベースがマウントされます。データベースがマウントされている場合、この句は使用できません。

MOUNT STANDBY DATABASE MOUNT STANDBY DATABASE を指定すると、スタンバイ・データベースがマウントされます。この文の実行直後、スタンバイ・インスタンスはプライマリ・インスタンスからアーカイブ REDO ログを受信し、ログを STANDBY_ARCHIVE_DEST で指定された位置にアーカイブします。

参照：『Oracle9i Data Guard 概要および管理』を参照してください。

MOUNT CLONE DATABASE MOUNT CLONE DATABASE を指定すると、クローン・データベースがマウントされます。

参照：『Oracle9i ユーザー管理バックアップおよびリカバリ・ガイド』を参照してください。

OPEN 句

OPEN 句を使用すると、データベースが使用可能な状態になります。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。スタンバイ・データベースをオープンするには、アクティブにしておく必要があります。

OPEN を他のキーワードを指定せずに単独で指定した場合のデフォルトは、OPEN READ WRITE NORESETLOGS です。

READ WRITE READ WRITE を指定すると、読取り / 書込みモードでデータベースがオープンされ、ユーザーは REDO ログを生成できるようになります。これはデフォルトです。

RESETLOGS RESETLOGS を指定すると、現行のログ順序番号を 1 にリセットし、リカバリ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されます。これによって、REDO ログに記録されている変更は、すべて破棄されますが、データベース内の変更は破棄されません。

RECOVER 句による不完全リカバリや、バックアップ制御ファイルによるメディア・リカバリを行った後は、RESETLOGS を指定してデータベースをオープンしてください。この句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行ってください。

NORESETLOGS NORESETLOGS を指定すると、ログの順序番号および REDO ログ・ファイルの現在の状態が保持されます。

制限事項：バックアップ制御ファイルによる完全メディア・リカバリまたは不完全メディア・リカバリを行った後でのみ RESETLOGS および NORESETLOGS を指定できます。それ以外の場合は、NORESETLOGS が自動的に適用されます。

READ ONLY READ ONLY を指定すると、トランザクションが読取り専用で制限され、ユーザーは REDO ログを生成できなくなります。この句は、アーカイブ・ログをプライマリ・データベース・サイトからコピー中でも、問合せに使用可能なスタンバイ・データベースを作成するために使用できます。

OPEN 句の制限事項

- データベースが他のインスタンスで READ WRITE でオープンされている場合、READ ONLY ではオープンできません。
- データベースをリカバリする必要がある場合、READ ONLY ではオープンできません。
- データベースが READ ONLY でオープンされている間は、表領域をオフラインに切り替えることはできません。ただし、データ・ファイルのオフラインとオンラインの切替えはできます。また、データベースが READ ONLY でオープンされている間は、オフラインのデータ・ファイルおよび表領域をリカバリできます。

recovery_clauses

recovery_clauses には、バックアップ後の操作が含まれます。

参照：データベースのバックアップの詳細は、『Oracle9i バックアップおよびリカバリ概要』、『Oracle9i Recovery Manager ユーザーズ・ガイド』および『Oracle9i Recovery Manager リファレンス』を参照してください。

general_recovery

general_recovery 句を指定すると、データベース、スタンバイ・データベース、または指定した表領域やファイルのメディア・リカバリについて指定できます。インスタンスで、データベースがマウント済、オープン状態、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、この句を使用できます。

制限事項：

- データベースがクローズ状態の場合にのみ、データベース全体をリカバリできます。
- インスタンスは、排他モードでデータベースがマウントされている必要があります。
- リカバリ対象の表領域またはデータ・ファイルがオフラインの場合、データベースがオープン状態でもクローズ状態でも、表領域またはデータ・ファイルをリカバリできます。
- 共有サーバー・アーキテクチャで Oracle に接続している場合、メディア・リカバリは実行できません。

注意： メディアについて特別な要件がない場合は、*general_recovery* 句ではなく、SQL*Plus の RECOVER コマンドを使用することをお勧めします。

参照：

- メディア・リカバリの詳細は、『Oracle9i ユーザー管理バックアップおよびリカバリ・ガイド』を参照してください。
- SQL*Plus の RECOVER コマンドの詳細は、『SQL*Plus ユーザーズ・ガイドおよびリファレンス』を参照してください。

AUTOMATIC

AUTOMATIC を指定すると、リカバリ操作を続けるために必要な新規アーカイブ REDO ログ・ファイルの名前が自動的に生成されます。LOG_ARCHIVE_DEST_n パラメータが定義されている場合、最初のローカル接続先に対して有効で使用可能なパラメータがスキャンされます。その接続先を LOG_ARCHIVE_FORMAT と結合させて使用し、ターゲットの REDO ログ・ファイル名を生成します。LOG_ARCHIVE_DEST_n が定義されていない場合は、かわりに LOG_ARCHIVE_DEST パラメータ値が使用されます。

ファイルが検索されると、そのファイルに含まれている REDO が適用されます。ファイルが検索されない場合は、ファイル名の入力を求めるプロンプトが表示されます。この場合、提案として生成されたファイル名が表示されます。

AUTOMATIC も LOGFILE も指定しない場合、ファイル名の入力を求めるプロンプトが表示されます。この場合、提案として生成されたファイル名が表示されます。次に、生成されたファイル名を受け入れるか、または完全なファイル名に置き換えます。生成されたファイル名とアーカイブ済のファイル名が異なることを認識しておく、LOGFILE 句を使用することによって時間を節約できます。

FROM 'location'

FROM 'location' を指定すると、アーカイブ REDO ログ・ファイル・グループを読み込む位置を指定できます。location には、使用するオペレーティング・システムの表記規則に従って、ファイルの位置を完全に指定する必要があります。このパラメータを指定しないと、そのアーカイブ REDO ログ・グループは、初期化パラメータ LOG_ARCHIVE_DEST または LOG_ARCHIVE_DEST_1 に指定された位置にあるとみなされます。

full_database_recovery

full_database_recovery 句を指定すると、データベース全体がリカバリされます。

DATABASE DATABASE 句を指定すると、データベース全体がリカバリされます。これはデフォルトです。データベースがクローズされている場合にのみ、このオプションを使用できます。

STANDBY DATABASE STANDBY DATABASE 句を指定すると、プライマリ・データベースからコピーされたアーカイブ REDO ログ・ファイルおよび制御ファイルを使用して、スタンバイ・データベースがリカバリされます。スタンバイ・データベースは、マウントされているがオープンされていない状態である必要があります。

注意： この句は、オンライン・データ・ファイルのみをリカバリします。

- UNTIL を使用すると、リカバリ操作の存続期間を指定できます。
 - CANCEL は、取消しベースのリカバリを示します。RECOVER CANCEL 句を指定した ALTER DATABASE 文を発行するまで、データベース・リカバリが続行されます。
 - TIME は、時間ベースのリカバリを示します。このパラメータは、date に指定した時点までデータベースをリカバリします。date は、'YYYY-MM-DD:HH24:MI:SS' の書式の文字リテラルである必要があります。
 - CHANGE は、変更ベースのリカバリを示します。integer に指定したシステム変更番号 (SCN) の直前の、トランザクションの一貫性が保たれるところまでデータベースをリカバリします。
- 現行の制御ファイルのかわりにバックアップ制御ファイルを使用する場合、USING BACKUP CONTROLFILE を指定します。

partial_database_recovery

partial_database_recovery 句を指定すると、個々の表領域およびデータ・ファイルがリカバリされます。

TABLESPACE **TABLESPACE** 句を指定すると、指定した表領域のみがリカバリされます。リカバリの対象となる表領域がオフラインの場合、データベースがオープン状態でもクローズ状態でも、この句を使用できます。

DATAFILE **DATAFILE** 句を指定すると、指定したデータ・ファイルがリカバリされます。リカバリ対象のデータ・ファイルがオフラインの場合、データベースがオープン状態でもクローズ状態でも、この句を使用できます。

STANDBY TABLESPACE **STANDBY TABLESPACE** を指定すると、プライマリ・データベースおよび制御ファイルからコピーしたアーカイブ **REDO** ログ・ファイルを使用して、スタンバイ・データベース内に損失または破損した表領域が再構成されます。

STANDBY DATAFILE **STANDBY DATAFILE** を指定すると、プライマリ・データベースおよび制御ファイルからコピーしたアーカイブ **REDO** ログ・ファイルを使用して、スタンバイ・データベース内に損失または破損したデータ・ファイルが再構成されます。

- 旧スタンバイ・データ・ファイルまたは表領域のリカバリに、現行のスタンバイ・データベースの制御ファイルを使用する場合、**UNTIL [CONSISTENT WITH] CONTROLFILE** を指定します。ただし、スタンバイ制御ファイルに反映されていない **REDO** ログ・ファイルの内容は適用されません。キーワード **CONSISTENT WITH** はオプションであり、意味を明確にするためのものです。

LOGFILE

LOGFILE 'filename' を指定すると、指定した **REDO** ログ・ファイルを使用して、メディア・リカバリが続行されます。

TEST

TEST 句を使用すると、試行リカバリが実行されます。試行リカバリは、通常のリカバリ手順に問題があった場合に有効です。**REDO** ストリームを見ると、続いて発生する可能性のある問題を検出することができます。試行リカバリは、通常のリカバリと同様に **REDO** を適用しますが、ディスクに変更を書き込みません。変更は、試行リカバリの最後にロールバックされます。

ALLOW ... CORRUPTION

ALLOW integer CORRUPTION 句を指定すると、ログ・ファイルが破損した場合に、許容リカバリの続行で許容する破損ブロックの数を指定することができます。

試行リカバリでこの句を使用する（**TEST** 句と結合する）場合は、*integer* に 2 以上の数値を指定できます。通常のリカバリでこの句を使用する場合は、*integer* に 2 以上の数値を指定できません。

参照：

- 一般的なデータベースのリカバリの詳細は、『Oracle9i ユーザー管理 バックアップおよびリカバリ・ガイド』を参照してください。
- スタンバイ・データベースの管理リカバリの詳細は、『Oracle9i Data Guard 概要および管理』を参照してください。

parallel_clause

parallel_clause を使用すると、メディア・リカバリをパラレル化するかどうかを指定できます。

注意： *parallel_clause* 構文は、以前のリリースの構文に代わるものです。以前のリリースの構文は下位互換用にサポートされていますが、動作がわずかに異なることがあります。

NOPARALLEL NOPARALLEL を指定すると、シリアル実行が行われます。これはデフォルトです。

PARALLEL 並列度を選択する場合は、PARALLEL を指定します。並列度とは、すべての関係するインスタンスで使用可能な CPU の数に、初期化パラメータ PARALLEL_THREADS_PER_CPU の値を掛けたものです。

PARALLEL integer *integer* には、パラレル操作で使用するパラレル・スレッド数である**並列度**を指定します。各パラレル・スレッドは、1、2 個のパラレル実行サーバーを使用します。通常、最適な並列度が計算されるため、*integer* に値を指定する必要はありません。

参照： 詳細は、14-43 ページの「CREATETABLE」の「[parallel_clause に関する注意事項](#)」を参照してください。

CONTINUE

CONTINUE を指定すると、スレッドを使用禁止にするために中断されていた複数インスタンス・リカバリが再開されます。

CONTINUE DEFAULT を指定すると、他に指定されたログ・ファイルがない場合、自動的に生成された REDO ログ・ファイルを使用して、リカバリが再開されます。ファイル名の入力を求めるプロンプトが表示されないこと以外は、AUTOMATIC の指定と同じです。

CANCEL

CANCEL を指定すると、取消しベースのリカバリが終了します。

managed_standby_recovery

managed_standby_recovery 句を指定すると、自動スタンバイ・リカバリ・モードを指定できます。このモードでは、自動スタンバイ・データベースは、スタンバイ・データベース・アーキテクチャ全体のアクティブ・コンポーネントであるとみなされます。プライマリ・データベースは、その REDO ログ・ファイルをスタンバイ・サイトにアクティブにアーカイブします。これらのアーカイブ REDO ログは、スタンバイ・サイトにアーカイブされると、管理スタンバイ・リカバリ操作で使用可能になります。自動スタンバイ・リカバリは、メディア・リカバリに制限されます。インスタンスで、データベースがマウント済、オープン状態、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、この句を使用できます。

制限事項：8-20 ページの「[general_recovery](#)」と同じ制限があります。

参照： この句のパラメータの詳細は、『Oracle9i ユーザー管理バックアップおよびリカバリ・ガイド』を参照してください。

NODELAY 遅延アーカイブ・ログをスタンバイ・データベースにただちに適用する必要がある場合、NODELAY を指定します。この句によって、プライマリ・データベースの LOG_ARCHIVE_DEST_n パラメータの DELAY の設定がオーバーライドされます。この句を省略した場合、アーカイブ・ログの適用はパラメータ設定に従って遅延されます。パラメータで DELAY が指定されていない場合、アーカイブ・ログはただちに適用されます。

TIMEOUT TIMEOUT 句を使用すると、管理リカバリ操作の待ち時間を分単位で指定できます。自動スタンバイ・データベースに書き込むために、要求されたアーカイブ REDO ログ・ファイルが使用可能になるまで、リカバリ処理は *integer* 分待ちます。REDO ログ・ファイルがその時間内に使用可能にならなかった場合、リカバリ処理は終了し、エラー・メッセージが表示されます。自動スタンバイ・リカバリ・モードに戻る場合、この文を再発行します。

制限事項：DISCONNECT [FROM SESSION] を指定した場合は、TIMEOUT を指定できません。DISCONNECT 句はバックグラウンドのリカバリ操作を開始しますが、TIMEOUT はフォアグラウンドのリカバリ操作のみを適用対象とします。

TIMEOUT を指定しないと、RECOVER CANCEL 句付きでこの文を再発行するか、またはインスタンスが停止したり障害を発生しないかぎり、データベースは自動スタンバイ・リカバリ・モードのままです。

CANCEL CANCEL 句を指定すると、現在のアーカイブ REDO ファイルのすべての REDO が適用された後、管理スタンバイ・リカバリ操作が終了します。CANCEL のみを指定した場合、セッション制御はリカバリ処理が実際に終了した時点で戻ります。

- CANCEL IMMEDIATE を指定すると、現在のアーカイブ REDO ファイルのすべての REDO が適用された後、または次の REDO ログ・ファイルが読み込まれた後の、いずれか早い方の処理の後に、管理リカバリ操作が終了します。セッション制御は、リカバリ処理が実際に終了した時点で戻ります。

制限事項：CANCEL IMMEDIATE 句は、RECOVER MANAGED STANDBY DATABASE 文を発行したセッションからは発行できません。

- CANCEL IMMEDIATE NOWAIT は、セッション制御がリカバリ処理の終了後ではなくただちに戻されることを除いて CANCEL IMMEDIATE と同じです。
- CANCEL NOWAIT を指定すると、次の REDO ログ・ファイルが読み込まれた後に管理リカバリ操作が終了し、セッション制御がただちに戻ります。

DISCONNECT DISCONNECT を指定すると、バックグラウンド・プロセスの 1 つの管理 REDO プロセス (Managed Redo Process: MRP) が、分離されたバックグラウンド・プロセスとしてアーカイブ REDO ファイルを適用します。これによって、現行のセッションが他のタスクで使用可能になります。(FROM SESSION キーワードはオプションであり、意味を明確にするためのものです。)

参照： 管理 REDO プロセスの詳細は、『Oracle9i Data Guard 概要および管理』を参照してください。

FINISH FINISH を指定すると、スタンバイ・データベースの現行のログのスタンバイ・ログ・ファイルがリカバリされます。この句は、ログ・ライター (LGWR) プロセスが現行のスタンバイ・ログに REDO を転送するときにプライマリ・データベースが破損した場合に有効です。この句は、アーカイブ・ログ用に指定したすべての遅延間隔をオーバーライドし、ただちにログが適用されます。

NOWAIT NOWAIT を指定すると、制御がリカバリ処理の完了後ではなく、ただちに戻されます。

参照：

- スタンバイ・データベースのリカバリの詳細は、『Oracle9i Data Guard 概要および管理』を参照してください。
- LOG_ARCHIVE_DEST_n パラメータの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

END BACKUP 句

END BACKUP 指定すると、オンライン・バックアップ・モードの現行のデータベースにあるすべてのデータ・ファイルが、オンライン・バックアップ・モードから戻されます。この操作を実行するときデータベースはマウントされ、オープンされていない状態である必要があります。

3つの方法でオンライン・バックアップ（ホット・バックアップ）操作を終了できます。通常の操作では、ALTER TABLESPACE ... END BACKUP 文を使用して、表領域をオンライン・バックアップ・モードから戻します。これによって、表領域がオンライン・バックアップ・モードであることによるオーバーヘッドの増加を回避できます。

システム障害、インスタンス障害、または SHUTDOWN ABORT 操作の後には、オンライン・バックアップ・モードのファイルがシステム・クラッシュ時のファイルと一致するかどうかは識別されません。ファイルに一貫性がある場合、個々のデータ・ファイル、またはすべてのデータ・ファイルをオンライン・バックアップ・モードから戻すことができます。これによって、起動時にファイルのメディア・リカバリを回避できます。

- 個々のデータ・ファイルをオンライン・バックアップ・モードから戻すには、ALTER DATABASE DATAFILE ... END BACKUP 文を使用します。8-26 ページの「[database_file_clauses](#)」を参照してください。
- 表領域のすべてのデータ・ファイルをオンライン・バックアップ・モードから戻すには、ALTER TABLESPACE ... END BACKUP 文を使用します。

参照： オンラインの表領域バックアップの終了については、10-82 ページの「[ALTER TABLESPACE](#)」を参照してください。

database_file_clauses

database_file_clauses を指定すると、データ・ファイルおよびテンポラリ・ファイルを変更できます。インスタンスで、データベースがマウント済、オープン状態、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、次の句はどれでも使用できます。

CREATE DATAFILE

CREATE DATAFILE 句を使用すると、元のデータ・ファイルのかわりに新しい空のデータ・ファイルが作成されます。この句を使用すると、バックアップを取らずに失われたデータ・ファイルを再作成できます。'*filename*' には、データベースの一部であるファイル、または以前一部であったファイルを指定します。

- AS NEW を指定すると、データ・ファイル用のデフォルトのファイル・システム位置に、システムが生成するファイル名で、置き換えるファイルと同じサイズの Oracle 管理データ・ファイルが作成されます。
- AS *filespec* を指定すると、新しいデータ・ファイルのファイル名（およびオプションでサイズも）を割り当てることができます。

filename と同一名の Oracle 管理データ・ファイルがすでに存在する状態で *AS filespec* を指定すると、古いファイルは削除されます。*filename* と同一名のユーザー管理データ・ファイルがすでに存在する状態で *AS filespec* を指定すると、ファイルはそのまま残り、エラーが戻されません。

AS 句を指定しない場合、Oracle によって、'*filename*' に指定したファイルと同じ名前およびサイズのファイルが新しく作成されます。

リカバリ時には、元のデータ・ファイルの作成後に書き込まれたアーカイブ REDO ログを、失われたデータ・ファイルに代わる新しい空のデータ・ファイルに適用する必要があります。

新しいファイルは、元のファイルの作成時と同じ状態で作成されます。新しいファイルを元のファイルが失われた時点の状態に戻すには、メディア・リカバリを行ってください。

制限事項：SYSTEM 表領域の最初のデータ・ファイルに基づいて新しいファイルを作成することはできません。

DATAFILE 句

DATAFILE 句を使用すると、次のようにデータベース・ファイルを変更できます。

ONLINE ONLINE を指定すると、データ・ファイルがオンラインになります。

OFFLINE OFFLINE を指定すると、データ・ファイルがオフラインになります。データベースがオープンされている場合、データ・ファイルをオンラインに戻す前に、データ・ファイルのメディア・リカバリを行う必要があります。これは、データ・ファイルがオフラインになる前に、チェックポイントが実行されないためです。

DROP データベースが NOARCHIVELOG モードの場合は、DROP 句を指定して、データ・ファイルをオフラインにする必要があります。ただし、DROP 句は、データベースからデータ・ファイルを削除しません。そのため、データ・ファイルが存在する表領域を削除する必要があります。削除するまで、データ・ファイルは RECOVER または OFFLINE の状態でデータ・ディクショナリに残ります。

データベースが ARCHIVELOG モードの場合は、DROP キーワードは無視されます。

RESIZE RESIZE を指定すると、データ・ファイルのサイズを指定した絶対サイズ（バイト単位）を増やしたり減らすことができます。K または M を使用すると、KB または MB 単位で指定できます。デフォルト値はないため、必ずサイズを指定してください。

増やしたサイズに対して十分なディスク領域がない場合、または減らしたサイズを超えるデータがファイルに含まれる場合、エラー・メッセージが戻されます。

END BACKUP END BACKUP を指定すると、データ・ファイルがオンライン・バックアップ・モードから戻されます。END BACKUP 句は、ALTER DATABASE 構文の最上位に完全に記述されます。8-26 ページの「**END BACKUP 句**」を参照してください。

TEMPFILE 句

TEMPFILE 句を使用すると、テンポラリ・データ・ファイルのサイズを変更できます。また、`autoextend_clause` を指定すると、永続データ・ファイルと同じ効果を持たせることができます。

注意： オペレーティング・システムによっては、テンポラリ・ファイルのブロックが実際にアクセスされるまで、テンポラリ・ファイル用の領域が割り当てられない場合があります。領域の割当ての遅延のため、テンポラリ・ファイルの作成およびサイズ変更を行う必要があります。ただし、後でテンポラリ・ファイルが使用されるときに、十分なディスク領域を使用可能にする必要があります。ご使用のオペレーティング・システムのドキュメントを参照し、このような方法でテンポラリ・ファイル領域が割り当てられるかどうかを判断してください。

制限事項： データベースをオープンしないと、TEMPFILE は指定できません。

DROP DROP を指定すると、データベースからテンポラリ・ファイルが削除されます。表領域はそのまま残ります。

INCLUDING DATAFILES を指定すると、対応付けられたオペレーティング・システム・ファイルは削除され、削除された各ファイルのメッセージがアラート・ログに書き込まれます。

autoextend_clause

`autoextend_clause` は、新しいデータ・ファイルまたはテンポラリ・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、これらのファイルは自動的に拡張されません。

ON ON を指定すると、自動拡張を使用可能にします。

OFF OFF を指定すると、自動拡張を使用禁止にします。

注意： 自動拡張を使用禁止にする場合は、NEXT および MAXSIZE の値を 0（ゼロ）に設定します。後続の文で自動拡張を使用可能に戻す場合は、これらの値を設定しなおす必要があります。

NEXT NEXT 句を使用すると、エクステントがさらに必要になった場合にデータ・ファイルに自動的に割り当てられるディスク領域の増分サイズを、バイト単位で指定できます。K または M を使用すると、KB または MB 単位で指定できます。デフォルトのサイズは 1 データ・ブロックです。

MAXSIZE MAXSIZE を使用すると、データ・ファイルの自動拡張で使用するディスク領域の最大サイズを指定できます。

UNLIMITED データ・ファイルまたはテンポラリ・ファイルに割り当てられるディスク領域を制限しない場合は、UNLIMITED 句を使用します。

RENAME FILE 句

RENAME FILE 句を使用すると、データ・ファイル、テンポラリ・ファイルまたは REDO ログ・ファイル・メンバーの名前を変更できます。この句を指定する前に、オペレーティング・システムのファイル名の表記規則に従って、各ファイル名を指定してください。

- データ・ファイルおよびテンポラリ・ファイルにこの句を使用するには、データベースをマウントしておく必要があります。データベースはオープンしていてもかまいませんが、名前を変更するデータ・ファイルおよびテンポラリ・ファイルは、オフラインにしておく必要があります。
- ログ・ファイル用にこの句を使用するには、データベースはマウントされているがオープンされていない状態である必要があります。

この句は、制御ファイルのファイル名のみを変更します。オペレーティング・システムのファイルの名前は、実際には変更されません。オペレーティング・システムのファイル名は存在したままですが、使用されません。制御ファイルは旧ファイルをデータ・ファイル、テンポラリ・ファイルまたは REDO ログ・ファイルとして指していないため、旧ファイルが Oracle 管理の場合、オペレーティング・システムの旧ファイルはこの句の実行後に削除されます。

logfile_clauses

logfile_clauses を指定すると、ログ・ファイルを追加、削除または変更できます。

ARCHIVELOG | NOARCHIVELOG

ARCHIVELOG 句および NOARCHIVELOG 句は、インスタンスでデータベースがマウントされているがオープンされていない、かつ Real Application Clusters が使用禁止の場合のみに使用できます。

ARCHIVELOG REDO ログ・ファイル・グループを再利用する前に、グループの内容をアーカイブする場合は、ARCHIVELOG を指定します。このモードでは、メディア・リカバリができるようになります。この句は、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、Real Application Clusters を使用禁止にしてデータベースをマウントした場合のみに指定できます。

NOARCHIVELOG REDO ログ・ファイル・グループを再利用する前に、内容をアーカイブする必要がない場合は、NOARCHIVELOG を指定します。このモードでは、メディア障害後のリカバリはできません。

ADD [STANDBY] LOGFILE 句

ADD LOGFILE 句を使用すると、指定したスレッドに1つ以上の REDO ログ・ファイル・グループが追加され、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なります。STANDBY を指定すると、作成された REDO ログ・ファイルは、スタンバイ・データベースのみで使用可能になります。

ログ・ファイルがオンライン用、スタンバイ・データベース用のどちらに設計されたかを確認するには、V\$LOGFILE 動的パフォーマンス・ビューの TYPE 列を問い合わせます。

THREAD THREAD 句は、パラレル・モードの Real Application Clusters で Oracle を使用している場合にのみ適用可能です。integer にはスレッド番号を指定します。作成可能なスレッドの数は、CREATE DATABASE 文で指定した MAXINSTANCES パラメータ値によって制限されます。

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、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。

GROUP GROUP 句を指定すると、すべてのスレッドのすべてのグループ間で、REDO ログ・ファイル・グループを一意に識別できます。この値は、1 ～ MAXLOGFILES の値までの範囲で設定できます。同一の GROUP 値を持つ REDO ログ・ファイル・グループは、複数追加できません。このパラメータを指定しない場合、値が自動的に生成されます。REDO ログ・ファイル・グループの GROUP 値は、動的パフォーマンス・ビュー V\$LOG で確認できます。

filespec filespec には、1 つ以上のメンバー（1 つ以上のコピー）で構成される REDO ログ・ファイル・グループを指定します。

参照：

- 構文の詳細は、16-27 ページの「[filespec](#)」を参照してください。
- 動的パフォーマンス・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

ADD [STANDBY] LOGFILE MEMBER 句

ADD LOGFILE MEMBER 句を使用すると、既存の REDO ログ・ファイル・グループに新しいメンバーを追加できます。新しいメンバーをそれぞれ '*filename*' に指定します。すでにファイルが存在する場合、追加するメンバーはグループ内の他のメンバーと同じサイズである必要があり、REUSE を指定する必要があります。ファイルが存在しない場合、適切なサイズのファイルが作成されます。メディア障害によってグループのすべてのメンバーを失った場合は、そのグループにメンバーを追加することはできません。

STANDBY を指定すると、ログ・ファイル・メンバーがスタンバイ・データベースのみで使用可能になります。ただし、このキーワードは必須ではありません。グループ integer がスタンバイ・データベース用に追加された場合は、すべてのメンバーも同様にスタンバイ・データベースのみに使用されます。

次のいずれかの方法で、既存の REDO ログ・ファイル・グループを指定できます。

GROUP *integer* *integer* には REDO ログ・ファイル・グループを識別する GROUP パラメータの値を指定します。

filename(s) REDO ログ・ファイル・グループのすべてのメンバーを指定します。ご使用のオペレーティング・システムの表記規則に従って、ファイル名を完全に指定する必要があります。

ADD SUPPLEMENTAL LOG DATA 句

ADD SUPPLEMENTAL LOG DATA 句を指定すると、更新操作実行中にログ・ストリームに列データを追加できます。この情報は、LogMiner および LogMiner の技術に基づいた製品のみで使用可能です。

注意： データベースのオープン時にこの文を発行できます。ただし、パフォーマンスに影響するカーソル・キャッシュのすべての DML カーソルが、キャッシュが再移入されるまで無効になります。

PRIMARY KEY COLUMNS PRIMARY KEY COLUMNS を指定すると、主キーを持つすべての表において、主キーのすべての列は更新操作が実行されるたびに REDO ログに置かれます。主キーを定義していない場合は、行を一意に識別する列の集合が REDO ログに置かれます。この集合には、最大サイズが固定長のすべての列が含まれます。

UNIQUE INDEX COLUMNS UNIQUE INDEX COLUMNS を指定すると、一意キーを持つすべての表において、一意キーの列が変更されると、一意に属するその他すべての列も REDO ログに置かれます。

DROP LOGFILE 句

DROP LOGFILE 句を使用すると、REDO ログ・ファイル・グループのすべてのメンバーが削除されます。ADD LOGFILE MEMBER 句と同様に、REDO ログ・ファイル・グループを指定します。

- 現行のログ・ファイル・グループを削除する場合、最初に ALTER SYSTEM SWITCH LOGFILE 文を発行する必要があります。

参照： 9-19 ページの「[ALTER SYSTEM](#)」を参照してください。

- アーカイブが必要な REDO ログ・ファイル・グループは、削除できません。
- REDO ログ・ファイル・グループを削除すると、その REDO スレッドの REDO ログ・ファイル・グループが 2 つ未満になる場合は、削除できません。

DROP LOGFILE MEMBER 句

DROP LOGFILE MEMBER 句を使用すると、1 つ以上の REDO ログ・ファイル・メンバーが削除されます。各 '*filename*' には、ご使用のオペレーティング・システムのファイル名の表記規則に従って、メンバーを完全に指定する必要があります。

- 現行のログのログ・ファイルを削除する場合、最初に ALTER SYSTEM SWITCH LOGFILE 文を発行する必要があります。

参照： 9-19 ページの「[ALTER SYSTEM](#)」を参照してください。

- この句では、有効なデータを含む REDO ログ・ファイル・グループのすべてのメンバーを削除できません。このような操作には、DROP LOGFILE 句を使用してください。

DROP SUPPLEMENTAL LOG DATA 句

DROP SUPPLEMENTAL LOG DATA 句を使用すると、更新操作が発生するたびに REDO ログ・ストリームに追加のログ情報が置かれることを停止することができます。この文は、前述の ADD SUPPLEMENTAL LOG DATA 文によって得られた効果を終了します。

CLEAR LOGFILE 句

CLEAR LOGFILE 句を使用すると、オンライン REDO ログが初期化しなおされます。REDO ログをアーカイブしないオプションもあります。CLEAR LOGFILE は、REDO ログの追加および削除と似ていますが、スレッドに 2 つ以外ログがない場合や、クローズ状態のスレッドの現行の REDO ログ・ファイルに対しても発行できます。

- アーカイブされていない REDO ログを再利用する場合は、UNARCHIVED を指定する必要があります。

注意： リカバリのために REDO ログが必要な場合に UNARCHIVED を指定すると、バックアップが使用できなくなります。

- ARCHIVELOG モードのデータベースでデータ・ファイルをオフラインにする（DROP キーワードを使用せずに ALTER DATABASE ... DATAFILE OFFLINE を指定する）場合、およびそのデータ・ファイルをオンラインに戻す前に、消去するアーカイブされていないログがデータ・ファイルのリカバリに必要な場合は、UNRECOVERABLE DATAFILE を指定する必要があります。この場合、一度 CLEAR LOGFILE 文を実行して、データ・ファイルおよび表領域全体を削除する必要があります。

メディア・リカバリに必要なログを、CLEAR LOGFILE を使用して消去しないでください。データベースのチェックポイント後の REDO を含むログを消去する必要がある場合は、不完全メディア・リカバリを最初に実行する必要があります。オープンしているスレッドの現行の REDO ログは消去できます。クローズしているスレッドの現行のログは、そのスレッド内でログを切り替えれば消去できます。

CLEAR LOGFILE 文が、システム障害またはインスタンス障害による割込みを受けると、データベースがハングする場合があります。このような状況になった場合は、データベースの再起動後に、この文を再発行します。ログ・グループのあるメンバーにアクセスしようとした際、I/O エラーによる障害が発生した場合は、そのメンバーを削除して他のメンバーを追加できます。

controlfile_clauses

`controlfile_clauses` を指定すると、制御ファイルを作成またはバックアップできます。

CREATE STANDBY CONTROLFILE 句

CREATE STANDBY CONTROLFILE 句を使用すると、スタンバイ・データベースを管理するための制御ファイルを作成できます。ファイルがすでに存在している場合は、REUSE を指定してください。

参照：『Oracle9i Data Guard 概要および管理』を参照してください。

BACKUP CONTROLFILE 句

BACKUP CONTROLFILE 句を使用すると、現行の制御ファイルのバックアップが取られます。

TO 'filename' *filename* には制御ファイルのバックアップ先ファイルを指定します。ご使用のオペレーティング・システムの表記規則に従って、ファイル名を完全に指定する必要があります。指定したファイルがすでに存在している場合は、REUSE を指定します。

TO TRACE 制御ファイルの物理バックアップを取るのではなく、データベースのトレース・ファイルに SQL 文を書き込む場合は、TO TRACE を指定します。トレース・ファイルに記述された SQL 文を使用すると、データベースの起動、制御ファイルの再作成、データベースのリカバリやオープンなどの操作を、作成した制御ファイルに基づいて正しく実行できます。この句を使用するとき、データベースをオープンまたはマウントしておく必要があります。

制御ファイルのコピーがすべて失われた場合（または、制御ファイルのサイズを変更する場合）は、トレース・ファイルからスクリプト・ファイルに文をコピーし、必要に応じて文を編集すると、そのデータベースを使用できます。

- NORESETLOGS は、データベースの起動用としてトレース・ファイルに書き込まれた SQL 文が、ALTER DATABASE OPEN NORESETLOGS であることを示します。これはデフォルトです。
- RESETLOGS は、データベースの起動用としてトレース・ファイルに書き込まれた SQL 文が、ALTER DATABASE OPEN RESETLOGS であることを示します。

standby_database_clauses

standby_database_clauses を使用すると、スタンバイ・データベースをアクティブにする、またはプロテクト・モードと非プロテクト・モードのいずれかを指定することができます。

参照： スタンバイ・データベース、およびその管理と使用の詳細は、『Oracle9i Data Guard 概要および管理』を参照してください。

ACTIVATE STANDBY DATABASE 句

ACTIVATE STANDBY DATABASE 句を指定すると、スタンバイ・データベースの状態がアクティブ・データベースに変更され、プライマリ・データベースに設定することができます。この句を指定する前に、データベースをマウントしておく必要があります。キーワード **PHYSICAL** の指定は任意です。

SET STANDBY DATABASE 句

SET STANDBY DATABASE 句を指定すると、データベース環境が**データ非消失モード**かどうかを設定することができます。このモードでは、プライマリ・データベースとスタンバイ・データベースが完全に一致する管理が最優先されます。スタンバイ・データベースをマウントする必要があり、**Real Application Clusters** 以外のインスタンスでは、プライマリ・データベースを排他モードでもオープンできます。

PROTECTED **PROTECTED** を指定すると、プライマリ・データベースからスタンバイ・データベースへの最後の接続が解除されたときに、プライマリ・データベースがオープンおよびオープンし続けるために、スタンバイ・インスタンスが、ログ・ライター (LGWR) ・プロセスによってアーカイブされる 1 つ以上のアーカイブ・ログの出力先を含む必要があることを示します。**Real Application Clusters** 環境では、プライマリ・データベースを持つすべてのインスタンスの LGWR プロセスが、同一のスタンバイ・データベースへのアーカイブをオープンすることが検証されます。

スタンバイ・データベースへの最後の接続が解除されると、プライマリ・インスタンスは停止します。そのため、プライマリ・データベースとスタンバイ・データベースの完全な一致がデータベースの可用性より重要な場合のみにこの設定を使用します。

UNPROTECTED **UNPROTECTED** を指定すると、インスタンスがログ・ライター・プロセスによって管理されるスタンバイ・データベースを必要としないことを示します。これはデフォルトです。

プライマリ・データベースとスタンバイ・データベースの完全な一致がデータベースの可用性より重要でない場合にこの設定を使用します。

データベースが **PROTECTED** と **UNPROTECTED** モードのどちらであるか判断するには、**V\$DATABASE** 動的パフォーマンス・ビューの **STANDBY_DATABASE** 列を問い合わせます。

REGISTER LOGFILE 句

スタンバイ・データベースで REGISTER LOGFILE 句を指定すると、障害があったプライマリ・データベースからログ・ファイルを登録することができます。この操作は、障害があったプライマリ・データベースで消失したログ・ファイルが STANDBY_ARCH_DEST 初期化パラメータで指定されたディレクトリにコピーされていないかぎり、必須です。

OR REPLACE OR REPLACE を指定すると、たとえば位置やファイル仕様が変更された場合に、スタンバイ・データベースの既存のアーカイブ・ログ・エントリが更新されます。エントリの SCN は、正確に一致している必要があります。また、元のエントリは、管理スタンバイ・ログの送信メカニズムによって作成される必要があります。

PREPARE TO SWITCHOVER 句

プライマリ・データベースにおいて、PREPARE TO SWITCHOVER TO STANDBY を指定すると、スイッチオーバー用の現行のプライマリ・データベースがスタンバイ状態になります。スタンバイ・データベースにおいて、PREPARE TO SWITCHOVER TO PRIMARY 文を発行すると、スイッチオーバー用のスタンバイ・データベースがプライマリ状態になります。

default_settings_clauses

データベースのデフォルト設定を変更できます。

CHARACTER SET、NATIONAL CHARACTER SET

CHARACTER SET は、データベースでデータを格納する場合に使用するキャラクタ・セットを変更します。NATIONAL CHARACTER SET は、NCHAR、NCLOB または NVARCHAR2 と定義された列にデータを格納する場合に使用する各国語キャラクタ・セットを変更します。

character_set は、引用符なしで指定します。データベースは、オープンされている必要があります。

注意：

- ALTER DATABASE CHARACTER SET 文または ALTER DATABASE NATIONAL CHARACTER SET 文はロールバックできません。そのため、それらの文を発行する前に、データベース全体のバックアップを取ってください。
 - 既存データベースのキャラクタ・セットを新しいデータベースのキャラクタ・セットへ移行する前に、キャラクタ・セット・スキャナ (CSSCAN) を使用して、データを分析することをお勧めします。これによって、データベースで認識されない非 ASCII データの損失を防ぐことができます。CSSCAN の詳細は、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。
-
-

キャラクタ・セット変更の注意

Oracle9i では、マルチバイト・データベース・キャラクタ・セットの場合、CLOB データは、UCS-2 (2 バイト固定幅の Unicode) として格納されます。シングルのバイト・データベース・キャラクタ・セットの場合、CLOB データはデータベース・キャラクタ・セットに格納されます。ALTER DATABASE 文を使用して、データベースまたは各国語キャラクタ・セットを変更する場合は、データ変換は行われません。そのため、この文を使用してシングルのバイトからマルチバイトへデータベース・キャラクタ・セットを変更しても、CLOB 列は元のデータベース・キャラクタ・セットのままです。これは CLOB 列のデータの非一貫性です。同様に、各国語キャラクタ・セットを Unicode からその他に変更すると、SQL の NCHAR 列 (NCHAR, NVARCHAR2 および NCLOB) は破損します。

次の手順でデータベース・キャラクタ・セットを変更することをお勧めします。

1. CLOB および SQL の NCHAR データ型の列をエクスポートします。
2. CLOB および SQL の NCHAR 列を含む表を削除します。
3. ALTER DATABASE 文を使用して、キャラクタ・セットおよび各国語キャラクタ・セットを変更します。
4. CLOB および SQL の NCHAR 列を再インポートします。

制限事項:

- SYSDBA システム権限が必要です。また、データベースは、制限モードで (たとえば、SQL*Plus の STARTUP RESTRICT コマンドを使用して) 起動してください。
- 現在のキャラクタ・セットは、変更するキャラクタ・セットの厳密なサブセットである必要があります。ソース・キャラクタ・セットのコードポイント値で表される各キャラクタは、ターゲット・キャラクタ・セットの同一コードポイント値で表される必要があります。

参照: データベース・キャラクタ・セットの移行の詳細は、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

set_time_zone_clause

SET TIME_ZONE 句を使用すると、データベースのタイム・ゾーンを設定できます。次の 2 つの方法でタイム・ゾーンを設定します。

- 協定世界時 (UTC) (以前のグリニッジ標準時) からの置換値を指定。hh:mm の有効範囲は、-12:00 ~ +14:00 です。
- タイム・ゾーン地域を指定。有効な地域名を確認するには、V\$TIMEZONE_NAMES 動的パフォーマンス・ビューの TZNAME 列を問い合わせます。

参照: 動的パフォーマンス・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

すべての新しい `TIMESTAMP WITH LOCAL TIME ZONE` データは、ディスクに格納されるときにデータベースのタイム・ゾーンに正規化されます。データベースの既存のデータは、自動的に新しいタイム・ゾーンに更新されません。

この句を使用してタイム・ゾーンを設定または変更した後、新しいタイム・ゾーンを有効にするためにデータベースを再起動する必要があります。

DEFAULT TEMPORARY TABLESPACE 句

データベースのデフォルトの一時表領域を変更します。この操作の完了後、以前のデフォルトの一時表領域が割り当てられているすべてのユーザーに、新しいデフォルトの一時表領域が再度割り当てられます。その後、必要に応じて以前のデフォルトの一時表領域を削除することができます。

現行のデフォルトの一時表領域の名前を確認するには、`DATABASE_PROPERTIES` データ・ディクショナリ表の `PROPERTY_VALUE` 列を、`「PROPERTY_NAME = 'DEFAULT_TEMP_TABLESPACE'」` という条件で問い合わせます。

制限事項：デフォルトの一時表領域として割り当てる表領域、または再度割り当てる表領域は、標準的なブロック・サイズである必要があります。

conversion_clauses

RESET COMPATIBILITY 句

`RESET COMPATIBILITY` を指定すると、次に再起動する際、Oracle の以前のバージョンに再設定する必要があるデータベースにマークが付けられます。データベースがマウントされている場合、この句は使用できません。

注意： `RESET COMPATIBILITY` は、下位互換性に影響する Oracle の機能を使用禁止にしている場合にのみ有効です。

参照： Oracle の以前のバージョンへのダウングレードの詳細は、『Oracle9i データベース移行ガイド』を参照してください。

CONVERT 句

`CONVERT` 句を使用すると、Oracle7 のデータ・ディクショナリが変換されます。この句を使用すると、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。

注意： この句は、Oracle9i へ移行する場合にのみ使用し、データベースがマウントされている場合は使用しないでください。

参照： 『Oracle9i データベース移行ガイド』を参照してください。

redo_thread_clauses

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できます。

ENABLE THREAD 句

Oracle Real Application Clusters 環境において、ENABLE THREAD を指定すると、REDO ログ・ファイル・グループの指定スレッドが使用可能になります。使用可能にできるスレッドは、2 つ以上の REDO ログ・ファイルを持つスレッドのみです。データベースは、オープンされている必要があります。

PUBLIC PUBLIC を指定すると、初期化パラメータ THREAD で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。PUBLIC 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のみがこのスレッドを使用できます。

参照： スレッドを使用可能または使用禁止にする詳細は、『Oracle9i Real Application Clusters 管理』を参照してください。

DISABLE THREAD 句

DISABLE THREAD を指定すると、指定スレッドをすべてのインスタンスで使用禁止にできます。データベースは、オープンされている必要がありますが、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の場合は、そのスレッドを使用禁止にできません。

参照： スレッドを使用可能または使用禁止にする詳細は、『Oracle9i Real Application Clusters 管理』を参照してください。

RENAME GLOBAL_NAME 句

RENAME GLOBAL_NAME を指定すると、データベースのグローバル名を変更できます。database には、データベースの新しい名前を 8 バイト以内の長さで指定します。オプションの domain には、ネットワーク階層におけるデータベースの有効な位置を指定します。データベースがマウントされている場合、この句は使用できません。

注意： データベース名を変更しても、リモート・データベース内の既存のデータベース・リンク、シノニム、ストアド・プロシージャ、ストアド・ファンクションからのユーザーのデータベースに対するグローバル参照は変更されません。これらの参照を変更するのは、リモート・データベース管理者の責任です。

例

READ ONLY および READ WRITE 例 次の最初の文は、データベースを読み取り専用モードでオープンします。2 番目の文は、データベースを読み書き両用モードに戻し、オンライン REDO ログを消去します。

```
ALTER DATABASE OPEN READ ONLY;
```

```
ALTER DATABASE OPEN READ WRITE RESETLOGS;
```

PARALLEL 例 次の文は、パラレル・リカバリ処理を使用して表領域のリカバリを行います。

```
ALTER DATABASE  
  RECOVER TABLESPACE ts1  
  PARALLEL;
```

REDO ログ・ファイル・グループの例 次の文は、2 つのメンバーを含む REDO ログ・ファイル・グループを追加し、GROUP パラメータの値に 3 を指定してこのグループを識別します。

```
ALTER DATABASE stocks  
  ADD LOGFILE GROUP 3  
    ('diska:log3.log' ,  
     'diskb:log3.log') SIZE 50K;
```

REDO ログ・ファイル・グループ・メンバーの例 次の文は、前述の例で追加した REDO ログ・ファイル・グループに 1 つのメンバーを追加します。

```
ALTER DATABASE stocks  
  ADD LOGFILE MEMBER 'diskc:log3.log'  
  TO GROUP 3;
```

ログ・ファイル・メンバーの削除例 次の文は、前述の例で追加した REDO ログ・ファイル・メンバーを削除します。

```
ALTER DATABASE stocks  
  DROP LOGFILE MEMBER 'diskc:log3.log';
```

ログ・ファイル・メンバーの名前の変更例 次の文は、REDO ログ・ファイル・メンバーの名前を変更します。

```
ALTER DATABASE stocks  
  RENAME FILE 'diskb:log3.log' TO 'diskd:log3.log';
```

この例では、REDO ログ・グループ・メンバーのファイルが、別のファイル名に変更されただけです。ファイル名が、実際に 'diskb:log3.log' から 'diskd:log3.log' に変更されたわけではありません。実際のファイル名を変更するには、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。

すべてのログ・ファイル・グループ・メンバーの削除例 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;
```

REDO ログ・ファイル・グループの追加例 次の文は、3 つのメンバーを含む REDO ログ・ファイル・グループをスレッド 5 (Real Application Clusters 環境内) に追加して、このグループに GROUP パラメータ値 4 を割り当てます。

```
ALTER DATABASE stocks
  ADD LOGFILE THREAD 5 GROUP 4
    ('diska:log4.log',
     'diskb:log4:log',
     'diskc:log4.log' );
```

デフォルトの一時表領域の例 次の文は、データベースのデフォルトの一時表領域 temp を作成します。この文は、作成時に何も指定されていない場合にデフォルトの一時表領域を作成するか、既存のデフォルトの一時表領域を temp で置き換えます。

```
ALTER DATABASE
  DEFAULT TEMPORARY TABLESPACE temp;
```

Real Application Clusters スレッドの使用禁止化の例 次の文は、Real Application Clusters 環境のスレッド 5 を使用禁止にします。

```
ALTER DATABASE stocks
  DISABLE THREAD 5;
```

Real Application Clusters スレッドの使用可能化の例 次の文は、Real Application Clusters 環境のスレッド 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;
```

新しいデータ・ファイルの作成例 次の文は、新しいデータ・ファイル 'disk1:db1.dat' を基にデータ・ファイル 'disk2:db1.dat' を作成します。

```
ALTER DATABASE
  CREATE DATAFILE 'disk1:db1.dat' AS 'disk2:db1.dat';
```


グローバル・データベース名の変更例 次の文は、データベースのグローバル名を変更し、データベース名とドメインの両方を指定します。

```
ALTER DATABASE
  RENAME GLOBAL_NAME TO sales.australia.acme.com;
```

キャラクタ・セット例 次の文は、データベース・キャラクタ・セットおよび各国語キャラクタ・セットを UTF8 キャラクタ・セットに変更します。

```
ALTER DATABASE db1 CHARACTER SET UTF8;
ALTER DATABASE db1 NATIONAL CHARACTER SET UTF8;
```

データベース名はオプションです。キャラクタ・セット名は引用符を付けずに指定します。

データ・ファイルのサイズの変更例 次の文は、データ・ファイル 'disk1:db1.dat' のサイズを変更します。

```
ALTER DATABASE
  DATAFILE 'disk1:db1.dat' RESIZE 10 M;
```

ログ・ファイルの消去例 次の文は、ログ・ファイルを消去します。

```
ALTER DATABASE
  CLEAR LOGFILE 'disk3:log.dbf';
```

データベースのリカバリ例 次の文は、データベース全体を完全にリカバリし、必要な新しいアーカイブ REDO ログ・ファイル名を生成します。

```
ALTER DATABASE
  RECOVER AUTOMATIC DATABASE;
```

次の文は、Oracle が適用する REDO ログ・ファイル名を明示的に指定します。

```
ALTER DATABASE
  RECOVER LOGFILE 'diska:arch0006.arc';
```

次の文は、時間ベースのデータベース・リカバリを実行します。

```
ALTER DATABASE
  RECOVER AUTOMATIC UNTIL TIME '1998-10-27:14:00:00';
```

データベースは、1998 年 10 月 27 日の午後 2 時の状態にリカバリされます。

次の文は、表領域 user5 をリカバリします。

```
ALTER DATABASE
  RECOVER TABLESPACE user5;
```

次の文は、元のスタンバイ・データベース内の対応するデータ・ファイル、関連するアーカイブ・ログおよび現行のスタンバイ・データベース制御ファイルを使用して、スタンバイ・データ・ファイル /finance/stbs_21.f をリカバリします。

```
ALTER DATABASE
  RECOVER STANDBY DATAFILE '/finance/stbs_21.f'
  UNTIL CONTROLFILE;
```

管理スタンバイ・データベースの例 次の文は、自動スタンバイ・リカバリ・モードでスタンバイ・データベースをリカバリします。

```
ALTER DATABASE
  RECOVER MANAGED STANDBY DATABASE;
```

次の文は、データベースを自動スタンバイ・リカバリ・モードにします。管理リカバリ処理は、次のアーカイブ・ログまで最長 60 分間待ち状態になります。

```
ALTER DATABASE
  RECOVER MANAGED STANDBY DATABASE TIMEOUT 60;
```

後続のログがその前のログから 60 分以内に作成される場合、手動で終了しないかぎり、リカバリが継続されます。

次の文は、管理リカバリ操作を終了します。

```
ALTER DATABASE
  RECOVER MANAGED STANDBY DATABASE CANCEL IMMEDIATE;
```

管理リカバリ操作は、現行の REDO ログ・ファイルから REDO の次のグループが読み込まれる前に終了します。メディア・リカバリは、現行の REDO ログ・ファイルから REDO を適用している間に終了します。

ALTER DIMENSION

用途

ALTER DIMENSION 文を使用すると、ディメンションの階層関係またはディメンション属性を変更できます。

参照： ディメンションの詳細は、12-39 ページの「[CREATE DIMENSION](#)」を参照してください。

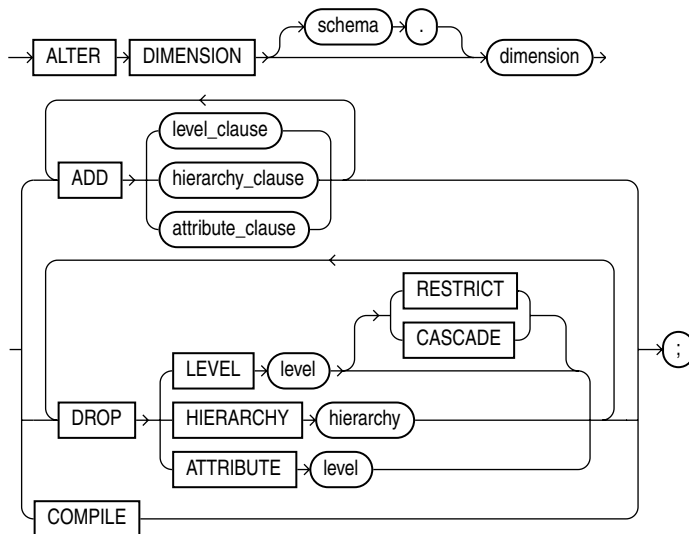
前提条件

この文を使用する場合、ディメンションが自分のスキーマ内に設定されているか、または ALTER ANY DIMENSION システム権限が必要です。

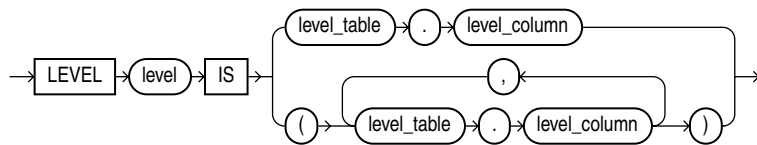
所有者の権限があれば、ディメンションはいつでも変更されます。

構文

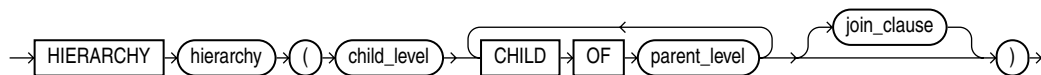
alter_dimension::=



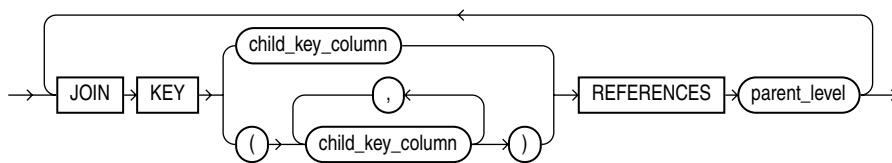
level_clause::=



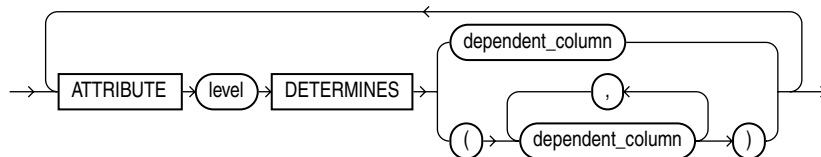
hierarchy_clause::=



join_clause::=



attribute_clause::=



キーワードとパラメータ

次のキーワードおよびパラメータの意味は、ALTER DIMENSION に対してのみ有効です。残りのキーワードおよびパラメータには、CREATE DIMENSION 文での機能と同じ機能があります。

参照： 12-39 ページの「[CREATE DIMENSION](#)」を参照してください。

schema

変更するディメンションのスキーマを指定します。*schema* を指定しない場合、ディメンションが自分のスキーマ内にあるとみなされます。

dimension

ディメンション名を指定します。ディメンションは、すでに存在している必要があります。

ADD

ADD 句を使用すると、ディメンションにレベル、階層または属性を追加できます。これらの要素の 1 つを追加しても、既存のマテリアライズド・ビューは無効になりません。

ADD LEVEL 句は、他の ADD 句より前に処理されます。

DROP

DROP 句を使用すると、ディメンションからレベル、階層または属性を削除できます。指定するすべてのレベル、階層または属性は、すでに存在している必要があります。

制限事項：属性または階層がレベルを参照する場合は、すべての参照している属性および階層を削除するか、または CASCADE を指定しないかぎり、このレベルを削除することはできません。

CASCADE レベルを参照する属性または階層をレベルとともに削除する場合は、CASCADE を指定します。

RESTRICT 属性または階層が参照するレベルを削除できないようにする場合は、RESTRICT を指定します。これはデフォルトです。

COMPILE

COMPILE を指定すると、無効のディメンションを明示的に再コンパイルできます。ADD 句または DROP 句が発行されると、ディメンションは自動的にコンパイルされます。ただし、ディメンションが参照するオブジェクトを変更する（たとえば、ディメンションで参照された表を削除した後再作成する）と、ディメンションが無効になるため、これを明示的に再コンパイルする必要があります。

例

ディメンションの変更例 次の例は、デモ・スキーマ sh の customers_dim ディメンションを変更します。

```
ALTER DIMENSION customers_dim
  DROP ATTRIBUTE country;

ALTER DIMENSION customers_dim
  ADD LEVEL zone IS customers.cust_postal_code
  ADD ATTRIBUTE zone DETERMINES (cust_city);
```

ALTER FUNCTION

用途

ALTER FUNCTION 文を使用すると、無効なスタンドアロンのストアド・ファンクションを再コンパイルできます。明示的に再コンパイルすることによって、実行時に暗黙的に再コンパイルする必要がなくなり、また、実行時のコンパイル・エラーとパフォーマンス上のオーバーヘッドも回避できます。

ALTER FUNCTION 文は、8-103 ページの「ALTER PROCEDURE」と似ています。ファンクションおよびプロシージャの再コンパイルの詳細は、『Oracle9i データベース概要』を参照してください。

注意： この文を使用して既存のファンクションの宣言や定義を変更することはできません。ファンクションを再宣言または再定義する場合は、OR REPLACE 句を指定して CREATE FUNCTION 文を使用します。詳細は、12-47 ページの「CREATE FUNCTION」を参照してください。

前提条件

ファンクションが自分のスキーマ内にあるか、ALTER ANY PROCEDURE システム権限が必要です。

構文

alter_function::=



キーワードとパラメータ

schema

ファンクションが含まれているスキーマを指定します。schema を指定しない場合、ファンクションが自分のスキーマ内に定義されているものとみなされます。

function

再コンパイルするファンクション名を指定します。

COMPILE

COMPILE を指定すると、ファンクションが再コンパイルされます。COMPILE キーワードは必須です。ファンクションが正常にコンパイルされない場合、SQL*Plus コマンド SHOW ERRORS を使用すると、関連するコンパイラ・エラー・メッセージが表示されます。

再コンパイル中、Oracle はすべての永続コンパイラのスイッチ設定を削除し、セッションからそれらを再び取得してコンパイルの終了時に格納します。この手順を回避するには、REUSE SETTINGS 句を指定します。

DEBUG

DEBUG を指定すると、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格納するように指示できます。

REUSE SETTINGS

REUSE SETTINGS を指定すると、Oracle によるコンパイラのスイッチ設定の削除および再取得を回避できます。この句では既存の設定が持続され、その設定で再コンパイルされます。

DEBUG と REUSE SETTINGS の両方を指定する場合は、PLSQL_COMPILER_FLAGS パラメータの永続格納値が INTERPRETED, DEBUG に設定されます。その他のコンパイラのスイッチ値は変更されません。

参照： PLSQL_COMPILER_FLAGS パラメータと COMPILE 句の相互作用の詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』および『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

例

ファンクションの再コンパイル例 次の文は、サンプル・ユーザー oe が所有するファンクション get_bal を明示的に再コンパイルします。

```
ALTER FUNCTION oe.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、実行時にそれを再コンパイルしなくても実行できます。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が戻り、get_bal は無効のままです。

また、get_bal に依存しているすべてのオブジェクトが無効になります。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した場合、Oracle は、実行時にそれらを暗黙的に再コンパイルします。

ALTER INDEX

用途

ALTER INDEX 文を使用すると、既存の索引を変更または再作成できます。

参照： 索引の作成については、12-58 ページの「[CREATE INDEX](#)」を参照してください。

前提条件

索引が自分のスキーマ内にあるか、または ALTER ANY INDEX システム権限が必要です。

MONITORING USAGE 句を実行する場合は、索引は自分のスキーマ内に存在する必要があります。

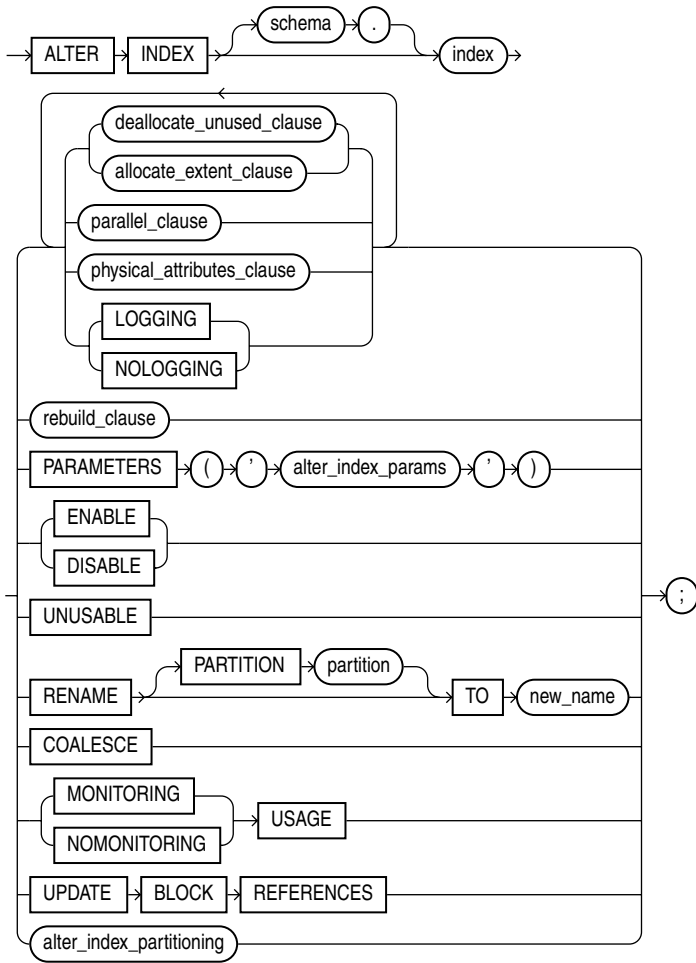
ドメイン索引を変更する場合は、索引の索引タイプに対して EXECUTE オブジェクト権限が必要です。

スキーマ・オブジェクト権限は、個々の索引パーティションまたはサブパーティションではなく、親索引に付与されている必要があります。

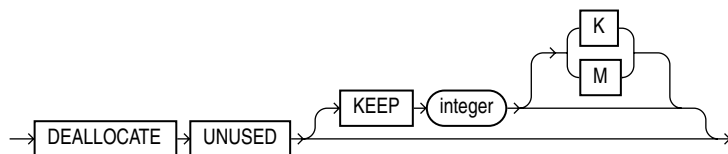
索引パーティションの変更、再作成または分割、あるいは索引サブパーティションの変更または再作成を行う場合、表領域割当て制限が必要です。

参照： ドメイン索引については、12-58 ページの「[CREATE INDEX](#)」および『Oracle9i Data Cartridge Developer's Guide』を参照してください。

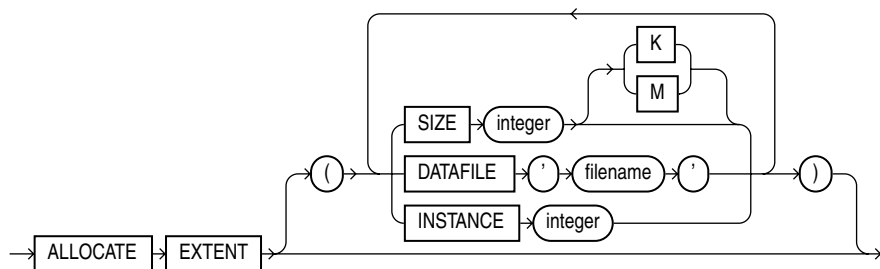
alter_index::=



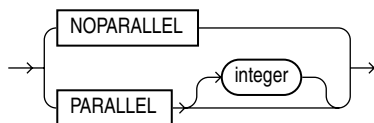
deallocate_unused_clause::=



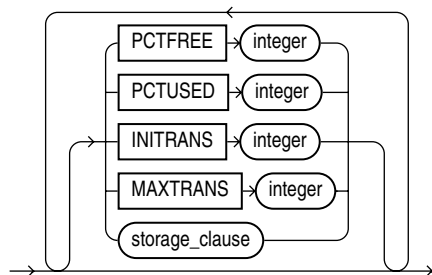
allocate_extent_clause::=



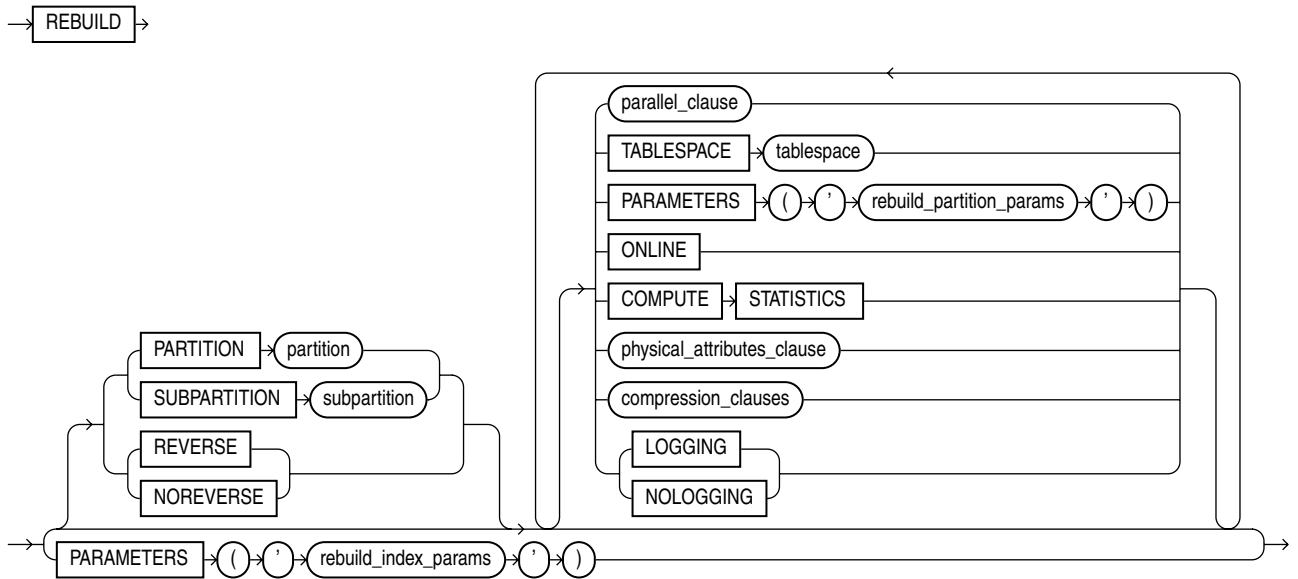
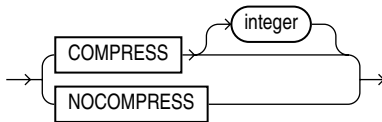
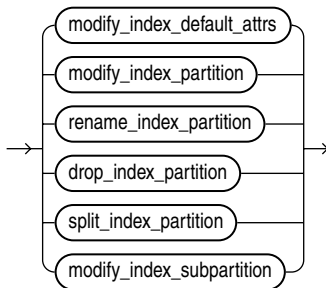
parallel_clause::=



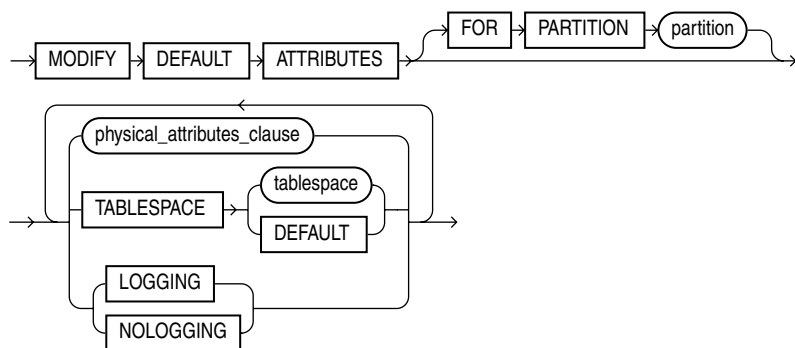
physical_attributes_clause::=



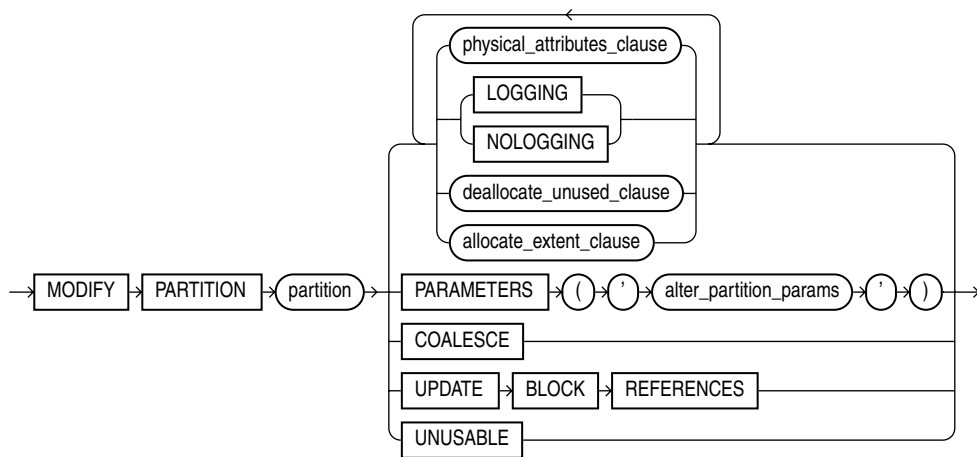
storage_clause: 17-50 ページの「[storage_clause](#)」を参照してください。

rebuild_clause::=**compression_clauses::=****alter_index_partitioning::=**

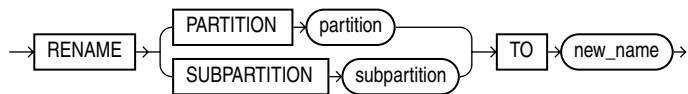
modify_index_default_attrs::=



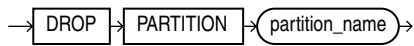
modify_index_partition::=



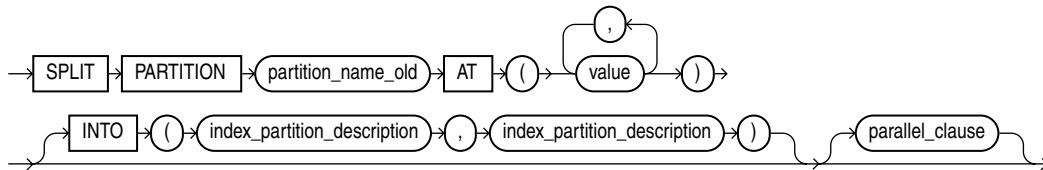
rename_index_partition::=



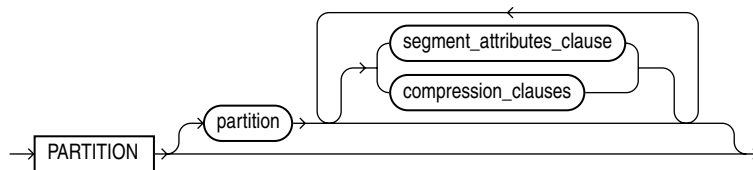
drop_index_partition::=



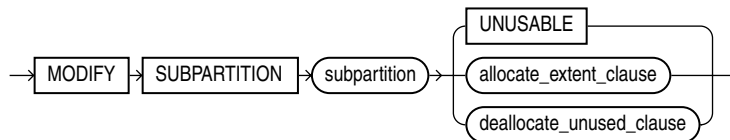
split_index_partition::=



index_partition_description::=



modify_index_subpartition::=



キーワードとパラメータ

schema

索引が含まれているスキーマを指定します。*schema* を指定しない場合、索引が自分のスキーマ内に定義されているものとみなされます。

index

変更する索引の名前を指定します。

制限事項：

- *index* がドメイン索引である場合は、*PARAMETERS* 句、*RENAME* 句または *rebuild_clause* (*PARAMETERS* 句の有無に関係なく) のみ指定できます。その他のすべての句は無効です。
- *LOADING* または *FAILED* のマークが付いているドメイン索引は、変更または名前の変更ができません。索引に *FAILED* のマークが付いている場合、*REBUILD* 句のみ指定できます。

参照： ドメイン索引の *LOADING* および *FAILED* 状態については、『Oracle9i Data Cartridge Developer's Guide』を参照してください。

deallocate_unused_clause

deallocate_unused_clause は、索引の終わりの未使用領域の割当てを明示的に解除し、解放された領域を表領域の他のセグメントで使えるようになります。ただし、解放できるのは、最高水位標を超える未使用領域のみです。

index がレンジ・パーティションまたはハッシュ・パーティションである場合、各索引パーティションの未使用領域の割当てが解除されます。*index* がコンポジット・パーティション表のローカル索引である場合、各索引サブパーティションの未使用領域の割当てが解除されます。

制限事項：

- この句は、一時表の索引に対して指定できません。
- この句および *rebuild_clause* は、指定できません。

参照： この句の詳細は、10-2 ページの「[ALTER TABLE](#)」を参照してください。

KEEP integer KEEP 句は、割当てを解除した後に索引に残す、最高水位標を超えるバイト数を指定できます。残りのエクステント数が MINEXTENTS より少ない場合、MINEXTENTS は現行のエクステント数に設定されます。初期エクステントが INITIAL より小さくなると、INITIAL は初期エクステントの現行の値に設定されます。KEEP を指定しないと、すべての未使用領域が解放されます。

参照： この句の詳細は、10-2 ページの「ALTER TABLE」を参照してください。

allocate_extent_clause

allocate_extent_clause は、索引の新しいエクステントを明示的に割り当てます。ハッシュ・パーティション表のローカル索引に対して、新規エクステントが索引の各パーティションに割り当てられます。

制限事項： この句は、一時表の索引、あるいはレンジ・パーティションまたはコンポジット・パーティション索引に対して指定できません。

SIZE エクステント・サイズをバイト単位で指定します。K または M を使用して、KB または MB 単位でエクステント・サイズを指定することもできます。SIZE を指定しないと、索引の記憶域パラメータの値に基づいてエクステントのサイズが決定されます。

DATAFILE 索引の表領域内で、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。DATAFILE を指定しないと、Oracle がデータ・ファイルを選択します。

INSTANCE INSTANCE 句を使用すると、指定したインスタンスのみが新しいエクステントを使用できるようになります。インスタンス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できます。Oracle を Real Application Clusters とあわせて使用する場合のみ、このパラメータを使用します。

この句を使用してエクステントを割り当てた場合、NEXT および PCTINCREASE 記憶域パラメータの値は変更されません。したがって、割り当てる次のエクステントのサイズには影響ありません。

parallel_clause

PARALLEL 句を使用すると、索引の間合せおよび DML に対するデフォルトの並列度を変更します。

制限事項： この句は、一時表の索引に対して指定できません。

注意： *parallel_clause* 構文は、以前のリリースの構文に代わるものです。以前のリリースの構文は下位互換用にサポートされていますが、動作がわずかに異なることがあります。

NOPARALLEL シリアル実行を行う場合は、NOPARALLEL を指定します。これはデフォルトです。

PARALLEL 並列度を選択する場合は、PARALLEL を指定します。並列度とは、すべての関係するインスタンスで使用可能な CPU の数に、初期化パラメータ PARALLEL_THREADS_PER_CPU の値を掛けたものです。

PARALLEL integer integer を指定すると、パラレル操作で使用するパラレル・スレッド数である**並列度**が指定されます。各パラレル・スレッドは、1、2 個のパラレル実行サーバーを使用します。通常、最適な並列度が計算されるため、integer に値を指定する必要はありません。

参照： 詳細は、14-43 ページの「CREATETABLE」の「[parallel_clause](#)に関する注意事項」を参照してください。

physical_attributes_clause

physical_attributes_clause を使用すると、非パーティション索引、パーティション索引のすべてのパーティションおよびサブパーティション、指定されたパーティション、または指定されたパーティションのすべてのサブパーティションに対するパラメータの値を変更できます。

参照： この句のパラメータの詳細は、14-6 ページの「[CREATE TABLE](#)」を参照してください。

制限事項：

- この句は、一時表の索引に対して指定できません。
- 索引の変更中は、PCTUSED パラメータを指定できません。
- PCTFREE パラメータは、*rebuild_clause*、*modify_index_default_attrs* 句または *split_partition_clause* の一部としてのみ指定できます。

storage_clause

storage_clause を使用すると、非パーティション索引、索引パーティション、またはパーティション索引のすべてのパーティションの記憶域パラメータ、あるいはパーティション索引の記憶域パラメータのデフォルト値を変更できます。

参照： 17-50 ページの「[storage_clause](#)」を参照してください。

LOGGING | NOLOGGING

LOGGING または NOLOGGING は、非パーティション索引、レンジまたはハッシュ索引パーティション、コンポジット・パーティション索引のすべてのパーティションまたはサブパーティションに対する後続のダイレクト・ローダー（SQL*Loader）およびダイレクト・パス 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。

NOLOGGING モードでは、データの変更時に、（新しいエクステンツに無効としてマーク設定し、ディクショナリの変更を記録するために）最小限のログが記録されます。メディア・リカバリ中に NOLOGGING が適用された場合、REDO データのログ記録が中断されるため、エクステン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、損失してはならない索引の場合は、NOLOGGING モードでの操作の後にバックアップを取る必要があります。

データベースを ARCHIVELOG モードで実行する場合、LOGGING モードでの操作の前に行ったバックアップからのメディア・リカバリによって、索引が再作成されます。ただし、NOLOGGING モードでの操作の前に行ったバックアップからのメディア・リカバリでは、索引は再作成されません。

索引セグメントには、実表の属性と異なるロギング属性、および同じ実表の他の索引セグメントと異なるロギング属性を指定できます。

制限事項：この句は、一時表の索引に対して指定できません。

参照： LOGGING およびパラレル DML の詳細は、『Oracle9i データベース概要』および『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

RECOVERABLE | UNRECOVERABLE

これらのキーワードは以前のバージョンのもので、それぞれ LOGGING および NOLOGGING に置き換えられています。RECOVERABLE および UNRECOVERABLE は、下位互換用のためにサポートされていますが、LOGGING および NOLOGGING キーワードを使用することをお勧めします。

RECOVERABLE は、パーティション表または LOB の記憶特性の作成時には無効なキーワードです。UNRECOVERABLE は、パーティション表または索引構成表の作成時には無効なキーワードです。また、CREATE INDEX の AS 副問合せ句を使用してのみ指定できます。

rebuild clause

rebuild clause を使用すると、既存の索引、あるいはパーティションまたはサブパーティションのいずれかを再構築します。索引に UNUSABLE のマークが付いている場合、正常に再構築すると USABLE になります。ファンクション索引も使用可能にします。索引の基になるファンクションが存在しない場合、再構築された文は正確に実行されません。

制限事項：

- 一時表の索引は、再構築できません。
- INVALID のマークが付いているビットマップ索引は再構築できません。制約を削除してからそれを再作成する必要があります。
- パーティション索引全体は、再構築できません。後述のとおり、各パーティションまたはサブパーティションを再構築する必要があります。
- この文の `deallocate_unused_clause` は指定できません。
- 索引全体 (ALTER INDEX) またはパーティション (ALTER INDEX ... MODIFY PARTITION) に対して、PCTFREE パラメータ値を変更できません。ALTER INDEX 文の他のすべての形式では、PCTFREE を指定できます。
- ドメイン索引に対して指定できるのは、PARAMETERS 句のみです (索引または索引のパーティション用)。その他の再構築の句は無効です。
- ローカル索引は再構築できませんが、ALTER INDEX ... REBUILD PARTITION でローカル索引のパーティションを再構築できます。

PARTITION 句

PARTITION 句を使用すると、索引のパーティションが 1 つ再構築されます。この句は、索引パーティションを別の表領域に移動したり、作成時の物理属性を変更するために使用できます。

注意： ブロック・サイズが異なる表領域の、パーティション化されたデータベース・エンティティの記憶域には、制限事項があります。これらの制限事項については、『Oracle9i データベース管理者ガイド』を参照してください。

制限事項： この句は、コンポジット・パーティション表のローカル索引に対して指定できません。かわりに、REBUILD SUBPARTITION 句を使用してください。

参照： パーティション・メンテナンス操作の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。

SUBPARTITION 句

SUBPARTITION 句を使用すると、索引のサブパーティションが 1 つ再構築されます。この句を使用して、索引パーティションを他の表領域に移動することもできます。TABLESPACE を指定しないと、サブパーティションは同じ表領域に再構築されます。

注意： ブロック・サイズが異なる表領域の、パーティション化されたデータベース・エンティティの記憶域には、制限事項があります。これらの制限事項については、『Oracle9i データベース管理者ガイド』を参照してください。

制限事項：

- パラメータ TABLESPACE および *parallel_clause* 以外は、サブパーティションに対して指定できません。
- リスト・パーティションのサブパーティションは再構築できません。

REVERSE | NOREVERSE

索引ブロックのバイトを逆順に格納するかどうかを示します。

- REVERSE は、索引ブロックのバイトを逆順で格納します。また、索引を再構築する場合の ROWID を除外します。
- NOREVERSE は、索引を再構築する場合に逆順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、逆キーの索引になります。

制限事項：

- ビットマップ索引または索引構成表は逆順には格納できません。
- パーティションまたはサブパーティションに対して、REVERSE または NOREVERSE を指定できません。

TABLESPACE 句

再構築された索引、索引パーティションまたは索引サブパーティションが格納される表領域を指定します。デフォルトは、索引またはパーティションを再構築する前に格納されていた表領域です。

COMPRESS | NOCOMPRESS

COMPRESS を指定すると、キー列値の繰返し項目を排除するキー圧縮が使用可能になります。*integer* を使用して、接頭辞の長さ（圧縮する接頭辞列数）を指定します。

- 一意の索引の場合、接頭辞の長さの有効範囲は、1 ～キー列の数から 1 を引いた数までです。接頭辞の長さのデフォルト値は、キー列の数から 1 を引いた数です。
- 一意でない索引の場合、接頭辞の長さの有効範囲は、1 ～キー列の数までです。接頭辞の長さのデフォルト値は、キー列の数です。

非パーティション索引（一意でない索引または 2 列以上の一意索引）のみを圧縮します。

制限事項：ビットマップ索引には、COMPRESS を指定できません。

NOCOMPRESS を指定すると、キー圧縮が使用禁止になります。これはデフォルトです。

ONLINE 句

ONLINE を指定すると、表またはパーティションの DML 操作を索引の再構築中に可能にするかどうかを指定します。

制限事項：

- オンラインで索引を作成中は、パラレル DML はサポートされません。ONLINE を指定し、パラレル DML 文を発行すると、Oracle はエラーを戻します。
- ビットマップ索引またはクラスタ索引には、ONLINE を指定できません。
- 参照整合性制約を適用する索引の再構築中に、ONLINE を指定することはできません。

COMPUTE STATISTICS 句

索引の再構築中に、比較的低コストで統計情報を収集する場合、COMPUTE STATISTICS を指定します。これらの統計情報は、SQL 文の実行計画を選択する際に、オブティマイザによって使用中のデータ・ディクショナリに格納されます。

収集された統計情報のタイプは、再構築している索引のタイプによって異なります。

注意：（表のかわりに）別の索引を使用して索引を作成する場合、元の索引は適切な統計情報を提供しない場合があります。このため、一般的に基となる表を使用して統計を計算します。その結果、統計は改善されますが、パフォーマンスが低下することがあります。

PL/SQL パッケージおよびプロシージャでは、その他の方法で統計を収集できます。

参照：『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

LOGGING | NOLOGGING

ALTER INDEX ... REBUILD 操作をログに記録するかどうかを指定します。

PARAMETERS 句

PARAMETERS 句は、ドメイン索引のみに適用されます。この句を使用すると、ドメイン索引またはドメイン索引のパーティションを変更または再構築するパラメータ文字列を指定できます。索引に UNUSABLE のマークが付いている場合、パラメータが変更されるのみで USABLE になりません。UNUSABLE のマークが付いた索引を再構築し、使用可能にする必要があります。

パラメータ文字列の最大長は 1,000 文字です。この文字列は、適切な索引タイプ・ルーチンに未解析のまま渡されます。

注意： Oracle Text がインストール済の場合、Oracle Text 固有のパラメータを使用する Oracle Text のドメイン索引を再構築することができます。パラメータの詳細は、『Oracle Text リファレンス』を参照してください。

制限事項：

- この句は、ドメイン索引以外の索引には指定できません。
- *index* に IN_PROGRESS または FAILED のマークが付いていない場合にのみ、索引パーティションを変更することができます。索引パーティションに IN_PROGRESS のマークが付くことはなく、変更されたパーティションに FAILED のマークは付きません。
- *index* に IN_PROGRESS のマークが付いていない場合にのみ、索引を再構築できます。
- *index* に IN_PROGRESS または FAILED のマークが付いておらず、かつ *partition* に IN_PROGRESS のマークが付いていない場合にのみ、索引パーティションを再構築できます。

参照：

- 索引タイプ・ルーチンの詳細は、『Oracle9i Data Cartridge Developer's Guide』を参照してください。
- ドメイン索引の詳細は、12-58 ページの「[CREATE INDEX](#)」を参照してください。

ENABLE 句

ENABLE は、索引が使用するユーザ一定義ファンクションが削除または置換されたために使用禁止になったファンクション索引のみに適用されます。次の条件が該当する場合、この句によって、このような索引が使用可能になります。

- ファンクションが現在有効な場合
- 現行のファンクションの署名が、索引が作成された場合のファンクションの署名と一致する場合
- ファンクションに現在 DETERMINISTIC のマークが付いている場合

制限事項：ENABLE と同じ文で、ALTER INDEX の他の句を指定できません。

DISABLE 句

DISABLE は、ファンクション索引のみに適用されます。この句を使用してファンクション索引を使用禁止にします。たとえば、ファンクションの本体を処理する場合に、これを行います。その後、ENABLE キーワードを使用して、索引を再構築、または別の ALTER INDEX 文を指定できます。

UNUSABLE 句

UNUSABLE を指定すると、索引、索引パーティションまたは索引サブパーティションに UNUSABLE のマークを付けます。使用禁止の索引を使用可能にする場合、再構築するか、または削除して再作成する必要があります。1 つのパーティションに UNUSABLE のマークが付いている場合も、同じ索引の他のパーティションは有効です。その索引を必要とする文が使用禁止のパーティションにアクセスしない場合、その文を実行できます。また、使用禁止のパーティションは、分割または名前を変更してから再構築できます。

制限事項：この句は、一時表の索引に対して指定できません。

RENAME 句

RENAME TO を指定すると、索引または索引パーティションの名前を変更できます。
`new_index_name` は単一の識別子で、スキーマ名は含まれません。

制限事項：

- ドメイン索引の場合、`index` および `index` のパーティションに IN_PROGRESS または FAILED のマークが付いていると無効になります。
- ドメイン索引のパーティションの場合、`index` に IN_PROGRESS または FAILED のマークが付いていると無効になります。また、パーティションに IN_PROGRESS のマークが付くことはなく、名前を変更するパーティションに FAILED のマークが付いていると無効になります。

COALESCE 句

COALESCE を指定すると、再利用するために、索引ブロックの内容を空きブロックにマージできます（可能な場合）。

制限事項：

- この句は、一時表の索引に対して指定できません。
- この句は、索引構成表の主キー索引に対して指定できません。かわりに ALTER TABLE の COALESCE 句を使用します。

参照：

- 領域管理および索引の結合の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。
- 索引構成表の領域の結合の詳細は、10-90 ページの「[COALESCE](#)」を参照してください。

MONITORING USAGE | NOMONITORING USAGE

この句を使用すると、索引の使用に関する統計情報の収集を開始または終了できます。この句は、索引が使用されているかどうかを判断するときに有効です。

MONITORING USAGE を指定すると、統計情報の収集が開始されます。まず、*index* の既存の統計情報が削除され、その後で索引の使用に関する統計情報の収集が開始されます。統計情報の収集は、その後の ALTER INDEX ... NOMONITORING USAGE 文が実行されるまで続行します。

索引の統計情報の収集を終了するには、NOMONITORING USAGE を指定します。

収集した統計情報を参照するには、ALL_INDEXES、USER_INDEXES または DBA_INDEXES データ・ディクショナリ・ビューを問い合わせます。統計情報の収集がいつ開始および終了したかを判断するには、V\$OBJECT_USAGE 動的パフォーマンス・ビューを問い合わせます。

参照： データ・ディクショナリおよび動的パフォーマンス・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

UPDATE BLOCK REFERENCES 句

UPDATE BLOCK REFERENCES 句は、索引構成表の通常のドメイン索引に対してのみ有効です。この句を指定すると、主キーによって識別されるブロックの適切なデータベース・アドレスとともに索引行の一部として格納され、失効したと推測されるすべてのデータ・ブロック・アドレスを更新することができます。

注意： ドメイン索引では、AlterIndexUpdBlockRefs に設定された alter_option パラメータで ODCIIndexAlter ルーチンが実行されます。このルーチンはカートリッジ・コードを使用可能にし、失効したと推測される索引のデータ・ブロック・アドレスを更新します。

制限事項： この句を ALTER INDEX の他の句と組み合わせることはできません。

alter_index_partitioning

ALTER INDEX 文のパーティション化句は、パーティション索引に対してのみ有効です。

注意： ブロック・サイズが異なる表領域の、パーティション化されたデータベース・エンティティの記憶域には、制限事項があります。これらの制限事項の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。

制限事項：

- これらの句は、一時表の索引に対して指定できません。
- ベース索引に対するいくつかの操作を1つの ALTER INDEX 文にまとめることはできますが（RENAME および REBUILD は除く）、パーティション操作を、他のパーティション操作またはベース索引に対する操作と組み合わせることはできません。

modify_index_default_attrs

パーティション索引のデフォルト属性に新しい値を指定します。

制限事項： ハッシュ・パーティション表またはコンポジット・パーティション表の索引の属性には、TABLESPACE のみ指定できます。

TABLESPACE 索引の新規パーティション、または索引パーティションのサブパーティションに対して、デフォルトの表領域を指定します。

LOGGING | NOLOGGING パーティション索引または索引パーティションのデフォルトのロギング属性を指定します。

FOR PARTITION FOR PARTITION 句を使用すると、コンポジット・パーティション表にあるローカル索引のパーティションのサブパーティションに対して、デフォルトの属性を指定します。

制限事項： リスト・パーティションに対して FOR PARTITION を指定することはできません。

modify_index_partition

modify_index_partition 句を使用すると、索引パーティション *partition* またはそのサブパーティションの実物理属性、ロギング属性または記憶特性を変更できます。

UPDATE BLOCK REFERENCES UPDATE BLOCK REFERENCES 句は、索引構成表の通常の索引に対してのみ有効です。この句を使用すると、セカンダリ索引パーティションに格納されている、失効したと推測されるすべてのデータ・ブロック・アドレスを更新することができます。

制限事項：

- ハッシュ・パーティション表の索引に対して、*physical_attributes_clause* を指定することはできません。
- ALTER INDEX の他の句とともに、UPDATE BLOCK REFERENCES を指定することはできません。

注意： 索引がコンポジット・パーティション表のローカル索引である場合、ここで指定した変更は、以前に索引のサブパーティションに対して指定したすべての属性をオーバーライドします。また、このパーティションの将来のサブパーティションに対する属性のデフォルト値を設定します。サブパーティションの属性をオーバーライドせずにパーティションのデフォルトの属性を変更する場合は、ALTER TABLE...MODIFY DEFAULT ATTRIBUTES OF PARTITION を使用します。

rename_index_partition

rename_index_partition 句を使用すると、索引パーティションまたは索引サブパーティションの名前を *new_name* に変更できます。

制限事項： リスト・パーティションのサブパーティションは名前を変更できません。

drop_index_partition

drop_index_partition 句を使用すると、グローバル・パーティション索引からパーティションとその中のデータを削除できます。グローバル索引のパーティションを削除する場合、その索引の次のパーティションに UNUSABLE のマークが付けられます。グローバル索引の最上位のパーティションは削除できません。

split_index_partition

split_index_partition 句を使用すると、グローバル・パーティション索引のパーティションを2つのパーティションに分割し、新しいパーティションを索引に追加できます。

UNUSABLE のマークが付いたパーティションを分割すると、2つのパーティションが生成されますが、その両方に UNUSABLE のマークが付けられます。このようなパーティションは、使用前に再構築する必要があります。

使用可能なパーティションを分割した場合、索引データが入っている2つのパーティションが生成されます。両方の新規パーティションは使用可能です。

AT 句 *split_partition_1* に新しい上限（境界は含まない）を指定します。
value_list の値は、*partition_name_old* の分割前のパーティション境界より小さく、その次の最小のパーティション（そのようなパーティションがある場合）のパーティション境界より大きい値である必要があります。

INTO 句 分割の結果、生成される2つのパーティションの名前と物理属性を任意に指定します。

modify_index_subpartition

modify_index_subpartition 句を使用すると、コンポジット・パーティション表にあるローカル索引のサブパーティションに対する領域の UNUSABLE のマーク付け、割当てまたは割当て解除が可能になります。このようなサブパーティションの他のすべての属性は、パーティションレベルのデフォルトの属性から継承されます。

制限事項： リスト・パーティションのサブパーティションは変更できません。

例

実属性の変更例 次の文は、同じ索引に将来追加されるデータ・ブロックが、5つの初期トランザクション・エントリと 100KB の増分エクステントを使用するように、索引 *oe.cust_lname_ix* を変更します。

```
ALTER INDEX oe.cust_lname_ix
  INITTRANS 5
  STORAGE (NEXT 100K);
```

索引 *oe.cust_lname_ix* がパーティション化されている場合、この文は将来追加される索引のパーティションのデフォルト属性も変更します。将来追加される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 100KB の増分エクステントが使用されます。

注意： ローカル管理の表領域を使用する場合、STORAGE 句のエクステン
ト管理を NEXT 100K と設定するとエラーが発生します。

索引パーティションの削除例 次の文は、索引パーティション `ix_antarctica` を削除します。

```
ALTER INDEX sales_area_ix
  DROP PARTITION ix_antarctica;
```

デフォルト属性の変更例 次の文は、ローカル・パーティション索引 `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 100KB の増分エクステン트가使用されます。

```
ALTER INDEX sales_ix3
  MODIFY DEFAULT ATTRIBUTES INITTRANS 5 STORAGE ( NEXT 100K );
```

索引への使用禁止のマーク付けの例 次の文は、索引 `idx_acctno` に UNUSABLE のマークを付けます。

```
ALTER INDEX idx_acctno UNUSABLE;
```

パーティションへの使用禁止のマーク付けの例 次の文は、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;
```

MAXEXTENTS の変更例 次の文は、パーティション `brix_ny` のエクステン트의最大数を変更し、ロギング属性を変更します。

```
ALTER INDEX branch_ix MODIFY PARTITION brix_ny
  STORAGE( MAXEXTENTS 30 ) LOGGING;
```

パラレル問合せの使用禁止例 次の文は、索引 `artist_ix` に対するスキヤン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;
```

パーティションの再構築例 次の文は、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;
```

索引の名前の変更例 次の文は、索引の名前を変更します。

```
ALTER INDEX emp_ix1 RENAME TO employee_ix1;
```

索引パーティションの名前の変更例 次の文は、索引パーティションの名前を変更します。

```
ALTER INDEX employee_ix1 RENAME PARTITION emp_ix1_p3  
    TO emp_ix1_p3;
```

パーティションの分割例 次の文は、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。

索引ブロックの逆順格納例 次の文は、索引ブロックのバイトが逆順に格納されるように、索引 emp_ix を再構築します。

```
ALTER INDEX emp_ix REBUILD REVERSE;
```

索引統計の収集例 次の文は、非パーティション索引 emp_idx の統計情報を収集します。

```
ALTER INDEX emp_idx REBUILD COMPUTE STATISTICS;
```

収集された統計情報のタイプは、再構築している索引のタイプによって異なります。

参照：『Oracle9i データベース概要』を参照してください。

PARALLEL 例 次の文は、パラレル実行プロセスを使用して、既存の索引から索引を再構築し、既存の索引のスキャンおよび新しい索引の構築を行います。

```
ALTER INDEX emp_idx  
    REBUILD  
    PARALLEL;
```

ALTER INDEXTYPE

用途

ALTER INDEXTYPE 文を使用すると、索引タイプの演算子を追加または削除したり、実装タイプを変更したり、索引タイプのプロパティを変更することができます。

前提条件

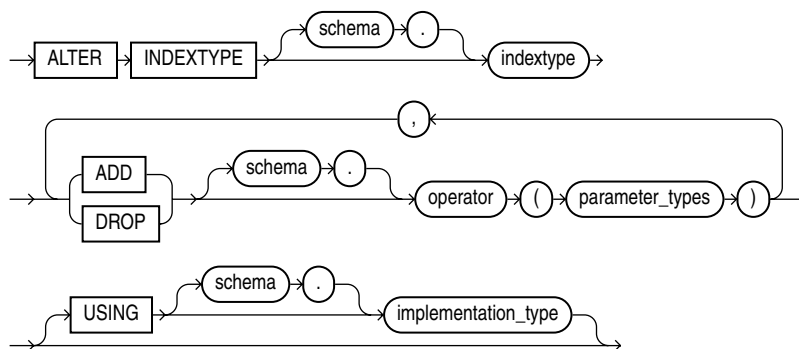
自分のスキーマまたはその他のスキーマの索引タイプを変更する場合は、ALTER ANY INDEXTYPE システム権限が必要です。

新しい演算子を追加する場合は、その演算子に対する EXECUTE オブジェクト権限が必要です。

実装タイプを変更する場合は、新しい実装タイプに対する EXECUTE オブジェクト権限が必要です。

構文

alter_indextype::=



キーワードとパラメータ

schema

索引タイプが存在するスキーマ名を指定します。*schema* を指定しない場合、索引タイプが自分のスキーマ内に定義されているものとみなされます。

indextype

変更する索引タイプの名前を指定します。

ADD | DROP

ADD または DROP 句を使用すると、演算子を追加または削除できます。

- *schema* には、演算子を含むスキーマを指定します。*schema* を指定しない場合、その演算子が自分のスキーマにあるとみなされます。
- *operator* には、索引タイプによってサポートされる演算子の名前を指定します。
この句に指定されるすべての演算子は有効な演算子である必要があります。
- *parameter_type* には、演算子へのパラメータ・タイプを指定します。

USING 句

USING 句を使用すると、索引タイプを実装する新しいタイプを指定できます。

例

次の例では、CREATE INDEXTYPE 文で作成した TextIndexType 索引タイプに他の演算子バインディングを追加します。TextIndexType は、(CLOB, CLOB) バインディングを伴う新しい演算子 lob_contains をサポートします。

```
ALTER INDEXTYPE TextIndexType ADD lob_contains(CLOB, CLOB);
```

ALTER JAVA

用途

ALTER JAVA 文を使用すると、Java クラス・スキーマ・オブジェクトの変換、または Java ソース・スキーマ・オブジェクトのコンパイルを強制実行します（Java 名に対するすべての外部参照を他のクラスと対応付ける前に、Java クラスのメソッドをコールすることはできません）。

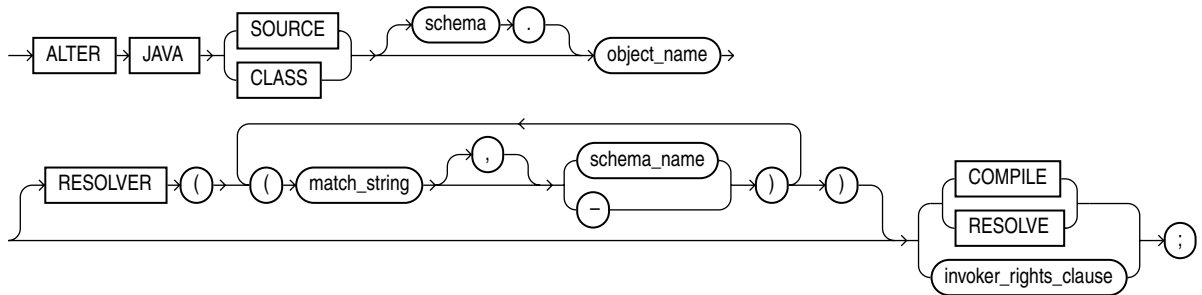
参照： Java クラスの変換および Java ソースのコンパイルの詳細は、『Oracle9i Java Stored Procedures Developer's Guide』を参照してください。

前提条件

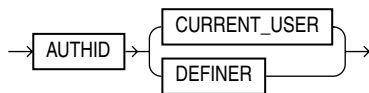
Java ソース、クラスまたはリソースが自分のスキーマ内にあるか、または ALTER ANY PROCEDURE システム権限が必要です。さらに、Java クラスに対する EXECUTE オブジェクト権限も必要です。

構文

alter_java::=



invoker_rights_clause::=



キーワードとパラメータ

JAVA SOURCE

ALTER JAVA SOURCE を使用すると、Java ソース・スキーマ・オブジェクトをコンパイルできます。

JAVA CLASS

ALTER JAVA CLASS を使用すると、Java ソース・スキーマ・オブジェクトを変換できます。

object_name

前回作成した Java クラスまたはソース・スキーマ・オブジェクトを指定します。小文字、または大文字と小文字を組み合わせた名前を付けるには、二重引用符を使用してください。

RESOLVER

RESOLVER 句によって、Java クラスまたはソースが作成されたときに指定したマッピング・ペアを使用して、完全に指定された参照用の Java 名に対するスキーマの検索方法を指定できます。

参照： 12-86 ページの「[CREATE JAVA](#)」を参照してください。

RESOLVE | COMPILE

RESOLVE および COMPILE は、同義のキーワードです。これらの句によって、プライマリ Java クラス・スキーマ・オブジェクトの変換を指定できます。

- クラスに適用された場合、他のクラス・スキーマ・オブジェクトに対する参照名に変換されます。
- ソースに適用された場合、ソースがコンパイルされます。

invoker_rights_clause

invoker_rights_clause は、メソッドを定義したユーザーのスキーマ内で、権限を使用してクラスのメソッドを実行するか、または、CURRENT_USER のスキーマ内で、権限を使用してクラスのメソッドを実行するかを指定します。

また、この句は、問合せ、DML 操作、およびその型のメンバー・ファンクションおよびプロシージャ内の動的 SQL 文の外部名の変換方法も定義します。

AUTHID CURRENT_USER クラスのメソッドが CURRENT_USER 権限で実行されることを指定する場合は、CURRENT_USER を意味します。この句はデフォルトで、実行者権限クラスを作成します。

また、この句は、問合せ、DML 操作、および動的 SQL 文の外部名を CURRENT_USER のスキーマで変換することも指定します。他のすべての文における外部名は、メソッドを含むスキーマで変換します。

AUTHID DEFINER メソッドを定義したユーザーの権限を使用して、クラスのメソッドを実行する場合は、DEFINER を指定します。

さらに、メソッドのあるスキーマ内で外部名を変換するかどうかを指定します。

参照：

- CURRENT_USER を判断する方法については、『Oracle9i データベース概要』および『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。
- 『Oracle9i Java Stored Procedures Developer's Guide』を参照してください。

例

Java クラスの変換例 次の文は、Java クラスを強制変換します。

```
ALTER JAVA CLASS "Agent"  
  RESOLVER (("/home/java/bin/" pm) (* public))  
  RESOLVE;
```

ALTER MATERIALIZED VIEW

用途

マテリアライズド・ビューは、問合せ結果を含むデータベース・オブジェクトです。問合せの FROM 句には、表、ビューおよび他のマテリアライズド・ビューを指定できます。これらをあわせて、**マスター表**（レプリケーション用語）または**ディテール表**（データ・ウェアハウス用語）といいます。このマニュアルでは、「マスター表」を使用します。マスター表が格納されているデータベースを**マスター・データベース**といいます。

ALTER MATERIALIZED VIEW 文を使用すると、既存のマテリアライズド・ビューを次の方法で変更します。

- 記憶特性を変更します。
- リフレッシュ方法、モードまたは時間を変更します。
- 別のタイプのマテリアライズド・ビューになるように構造を変更します。
- クエリー・リライトを使用可能または使用禁止にします。

注意： キーワード SNAPSHOT は、MATERIALIZED VIEW のかわりに下位互換用にサポートされています。

参照：

- マテリアライズド・ビューの作成の詳細は、13-5 ページの「[CREATE MATERIALIZED VIEW](#)」を参照してください。
- レプリケーション環境でのマテリアライズド・ビューの詳細は、『Oracle9i レプリケーション』を参照してください。
- データ・ウェアハウス環境でのマテリアライズド・ビューの詳細は、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

前提条件

マテリアライズド・ビューを変更するために必要な権限は、次のように直接付与される必要があります。

マテリアライズド・ビューが自分のスキーマ内にあるか、または ALTER ANY MATERIALIZED VIEW システム権限が必要です。

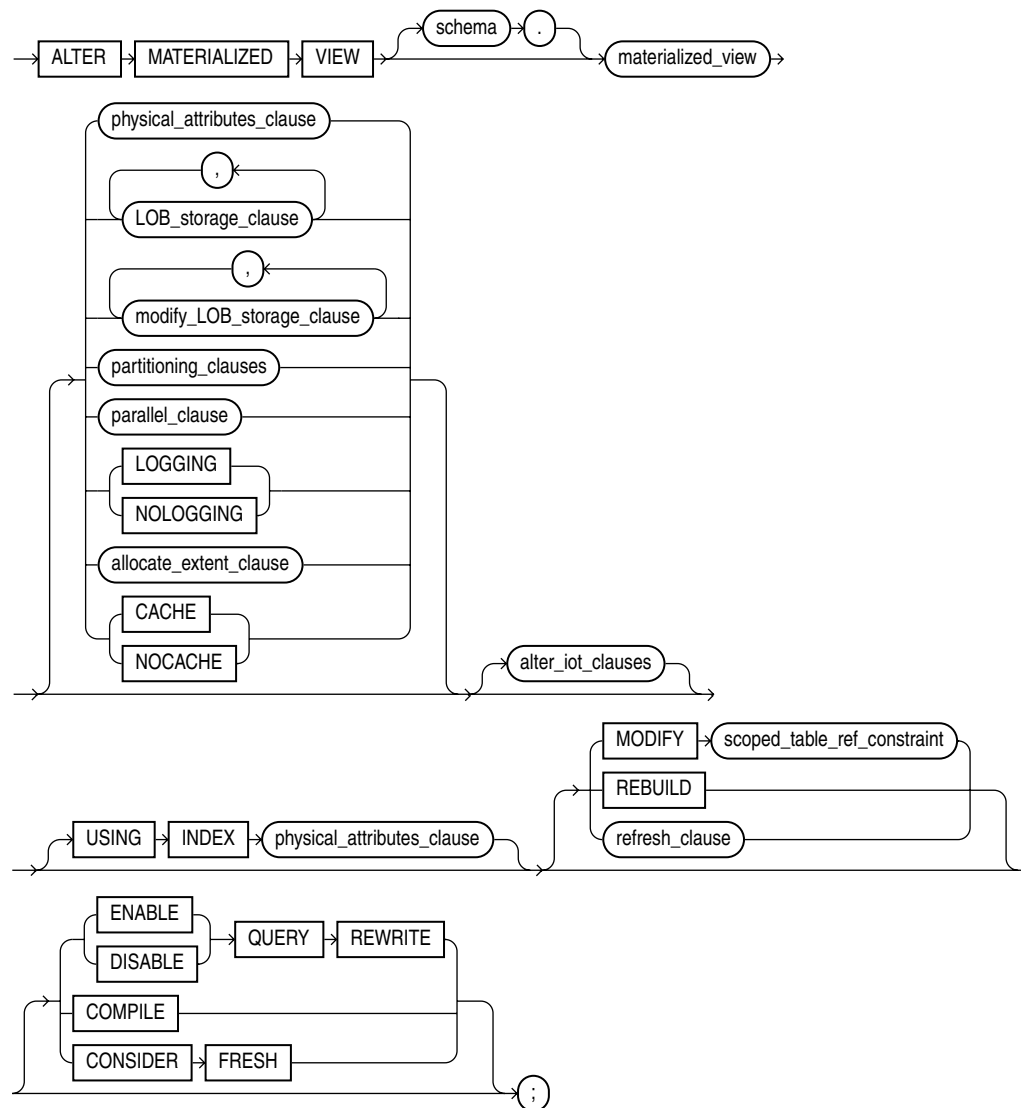
クエリー・リライトでマテリアライズド・ビューを使用可能にする場合、次の条件が必要です。

- マテリアライズド・ビュー内のすべてのマスター表が自分のスキーマ内にある場合、QUERY REWRITE 権限が必要です。
- いずれかのマスター表が別のスキーマ内にある場合、GLOBAL QUERY REWRITE 権限が必要です。
- マテリアライズド・ビューが別のユーザーのスキーマ内にある場合、ユーザーおよびそのスキーマ所有者の両方に、前述の適切な QUERY REWRITE 権限が必要です。また、マテリアライズド・ビューの所有者は、マテリアライズド・ビュー所有者が所有しないすべてのマスター表への SELECT 権限を持っている必要があります。

参照：『Oracle9i レプリケーション』および『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

構文

alter_materialized_view::=

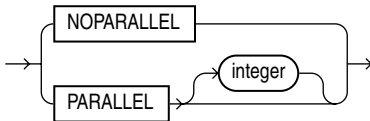


LOB_storage_clause: 10-2 ページの「ALTER TABLE」を参照してください。

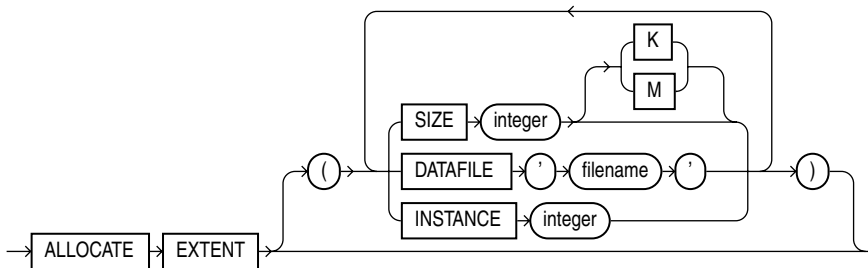
modify_LOB_storage_clause: 10-2 ページの「ALTER TABLE」を参照してください。

partitioning_clauses: 14-36 ページの「table_partitioning_clauses」を参照してください。

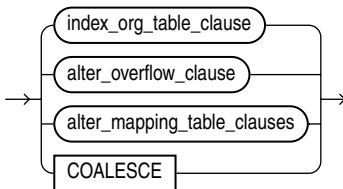
parallel_clause::=



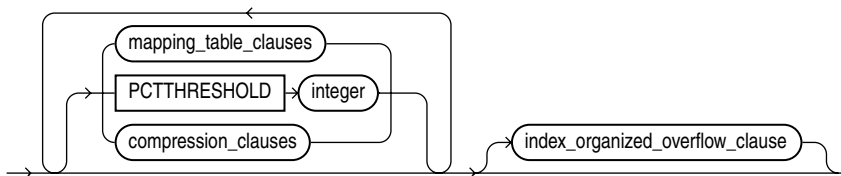
allocate_extent_clause::=



alter_iot_clauses::=



index_org_table_clause::=



mapping_table_clauses: マテリアライズド・ビューではサポートされません。

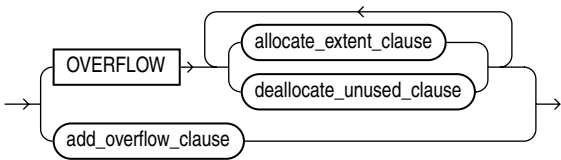
compression_clauses: マテリアライズド・ビューではサポートされません。

alter_mapping_clauses: マテリアライズド・ビューではサポートされません。

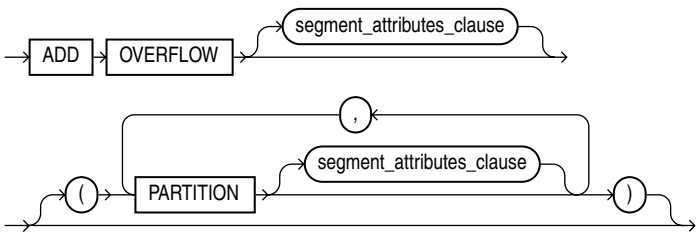
index_org_overflow_clause::=



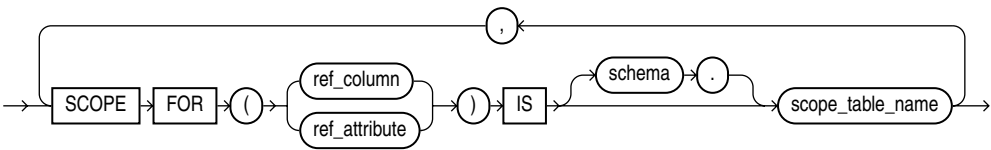
alter_overflow_clause::=



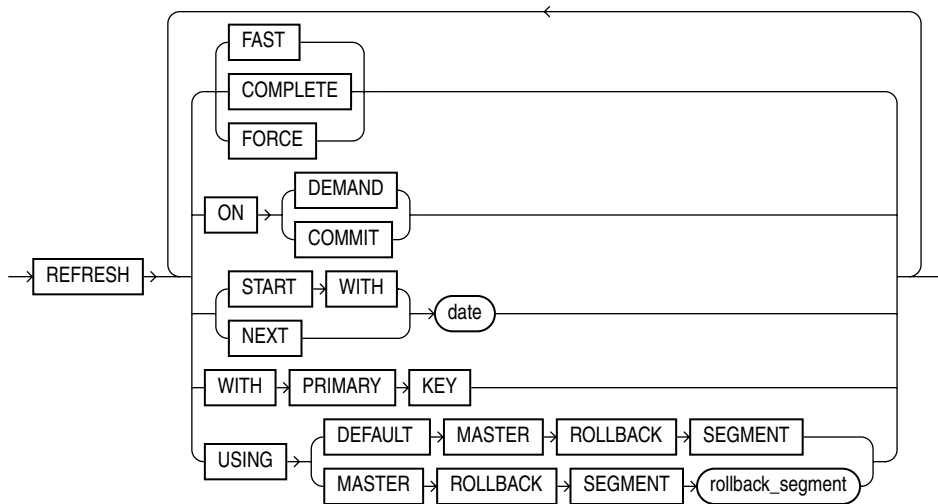
add_overflow_clause::=



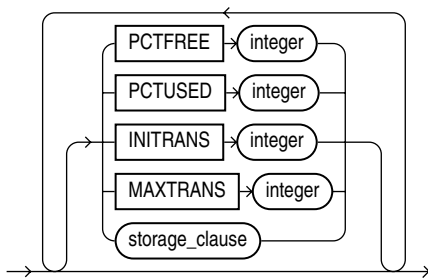
scoped_table_ref_constraint::=



alter_mv_refresh_clause::=



physical_attributes_clause::=



storage_clause: 17-50 ページの「[storage_clause](#)」を参照してください。

キーワードとパラメータ

schema

マテリアライズド・ビューが含まれているスキーマを指定します。*schema* を指定しない場合、マテリアライズド・ビューは自分のスキーマ内にあるとみなされます。

materialized_view

変更するマテリアライズド・ビューの名前を指定します。

physical_attributes_clause

PCTFREE、PCTUSED、INITRANS および MAXTRANS パラメータの値 (USING INDEX 句で使用する場合は、INITRANS および MAXTRANS パラメータ値のみ)、およびマテリアライズド・ビューの記憶特性を指定します。

参照：

- PCTFREE、PCTUSED、INITRANS および MAXTRANS パラメータの詳細は、10-2 ページの「[ALTER TABLE](#)」を参照してください。
- 記憶特性の詳細は、17-50 ページの「[storage_clause](#)」を参照してください。

LOB_storage_clause

LOB_storage_clause では、LOB 記憶特性を指定できます。

参照： この句のパラメータの指定については、10-2 ページの「[ALTER TABLE](#)」を参照してください。

modify_LOB_storage_clause

modify_LOB_storage_clause は、LOB 属性 *lob_item* の物理属性または LOB オブジェクト属性を変更できます。

参照： この句のパラメータの指定については、10-2 ページの「[ALTER TABLE](#)」を参照してください。

partitioning_clauses

マテリアライズド・ビューの *partitioning_clauses* の構文および一般的な機能は、パーティション表と同じです。

参照： 10-2 ページの「[ALTER TABLE](#)」を参照してください。

***partitioning_clauses* の制限事項**

- *partitioning_clauses* では、*LOB_storage_clause* または *modify_LOB_storage_clause* を指定できません。
- マテリアライズド・ビュー・パーティションを削除、切捨てまたは交換すると、Oracle はエラーを戻します。

注意： マテリアライズド・ビューの内容をマスター表の内容と同期させて保持するには、表パーティションを削除または切り捨てた後、表に依存しているすべてのマテリアライズド・ビューを手動で完全リフレッシュすることをお勧めします。

MODIFY PARTITION UNUSABLE LOCAL INDEXES この句を使用すると、*partition* に関連付けられたすべてのローカル索引パーティションに、UNUSABLE のマークを付けます。

MODIFY PARTITION REBUILD UNUSABLE LOCAL INDEXES この句を使用すると、*partition* に関連付けられた、使用禁止のローカル索引パーティションを再構築します。

parallel_clause

parallel_clause は、マテリアライズド・ビューのデフォルトの並列度を変更できます。

注意： *parallel_clause* 構文は、以前のリリースの構文に代わるものです。以前のリリースの構文は下位互換用にサポートされていますが、動作がわずかに異なることがあります。

NOPARALLEL シリアル実行を行う場合は、NOPARALLEL を指定します。これはデフォルト値です。

PARALLEL 並列度を選択する場合は、PARALLEL を指定します。並列度とは、すべての関係するインスタンスで使用可能な CPU の数に、初期化パラメータ PARALLEL_THREADS_PER_CPU の値を掛けたものです。

PARALLEL *integer* *integer* を指定すると、パラレル操作で使用されるパラレル・スレッド数である**並列度**が指定されます。各パラレル・スレッドは、1、2 個のパラレル実行サーバーを使用します。通常、最適な並列度が計算されるため、*integer* に値を指定する必要はありません。

参照： 詳細は、14-43 ページの「CREATETABLE」の「[parallel_clause](#)に関する注意事項」を参照してください。

LOGGING | NOLOGGING

マテリアライズド・ビューのロギング特性を指定または変更します。

参照： ロギング特性の詳細は、10-2 ページの「[ALTER TABLE](#)」を参照してください。

allocate_extent_clause

allocate_extent_clause は、マテリアライズド・ビューの新しいエクステントを明示的に割り当てます。

参照： 10-2 ページの「[ALTER TABLE](#)」を参照してください。

CACHE | NOCACHE

アクセス頻度の高いデータについて、CACHE は、フル・テーブル・スキャンの実行時にこの表用に取り出された各ブロックを、バッファ・キャッシュ内の LRU リストの最高使用頻度側に入れることを指定します。この属性は、小規模な参照表で有効です。NOCACHE は、ブロックを LRU リストの最低使用頻度側に入れることを指定します。

参照： CACHE または NOCACHE の指定については、10-2 ページの「[ALTER TABLE](#)」を参照してください。

alter_iot_clauses

alter_iot_clauses を使用すると、索引構成マテリアライズド・ビューの特性を変更できます。*alter_iot_clauses* コンポーネントのキーワードおよびパラメータは、ALTER TABLE と同じです。また、次の制限事項があります。

制限事項： *index_org_table_clause* の *mapping_table_clauses* または *compression_clauses* は、指定できません。

参照： 索引構成マテリアライズド・ビューの作成については、13-13 ページの「CREATE MATERIALIZED VIEW」の「[ORGANIZATION INDEX 句](#)」を参照してください。

USING INDEX 句

マテリアライズド・ビューのデータをメンテナンスするために使用される索引の INITRANS、MAXTRANS および STORAGE の各パラメータ値を変更します。

制限事項：この句では、PCTUSED または PCTFREE パラメータは指定できません。

MODIFY *scoped_table_ref_constraint*

MODIFY *scoped_table_ref_constraint* 句を使用すると、新しい表に REF 列または属性の有効範囲を再指定できます。

制限事項：ALTER MATERIALIZED VIEW 文では、1 つの REF 列または属性の有効範囲のみを再指定することができます。この句は、この文以外では使用できません。

REBUILD 句

REBUILD を指定すると、*materialized_view* で参照されるタイプが導出された場合に、リフレッシュ操作が再生成されます。

制限事項：同じ ALTER MATERIALIZED VIEW 文で他の句を指定できません。

alter_mv_refresh_clause

alter_mv_refresh_clause を使用すると、自動リフレッシュの方法、モードおよび時間のデフォルト値を変更できます。マテリアライズド・ビューのマスター表の内容が変更された場合、マテリアライズド・ビューのデータを更新し、現在マスター表にあるデータを正確に反映させる必要があります。この句によって、自動的にマテリアライズド・ビューをリフレッシュする時間をスケジューリングし、リフレッシュの方法およびモードを指定できます。

注意： この句では、デフォルトのリフレッシュ・オプションのみを設定します。リフレッシュを実際に実装する手順は、『Oracle9i レプリケーション』および『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

FAST 句

FAST を指定すると、増分リフレッシュ方法を指定できます。これはマスター表に対して行った変更に従ってリフレッシュを行います。この変更は、マスター表に関連付けられたマテリアライズド・ビュー・ログ（従来型 DML 変更の場合）またはダイレクト・ローダー・ログ（ダイレクト・パス INSERT 操作の場合）に格納されます。

従来型 DML の変更の場合も、ダイレクト・パス INSERT の場合も、他の条件によって、高速リフレッシュへのマテリアライズド・ビューの適応性が制限されることがあります。

参照：

- レプリケーション環境における高速リフレッシュの制限については、『Oracle9i レプリケーション』を参照してください。
- データ・ウェアハウス環境における高速リフレッシュの制限については、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

制限事項：

- 作成時に FAST リフレッシュを指定した場合、作成するマテリアライズド・ビューは高速リフレッシュに適応することが検証されています。ALTER MATERIALIZED VIEW 文でリフレッシュ方法を FAST に変更した場合、これは検証されていません。マテリアライズド・ビューが高速リフレッシュに適応しない場合、このビューをリフレッシュするとエラーが戻されます。
- 定義する問合せに分析ファンクションが含まれている場合、マテリアライズド・ビューは高速リフレッシュに適応しません。

参照： 6-8 ページの「[分析ファンクション](#)」を参照してください。

COMPLETE 句

完全リフレッシュを行う場合に、COMPLETE を指定します。これは、マテリアライズド・ビューの定義する問合せを実行することによって実装されます。完全リフレッシュを要求すると、高速リフレッシュが実行可能であっても、完全リフレッシュが実行されます。

FORCE 句

リフレッシュ時、高速リフレッシュが可能な場合は高速リフレッシュを実行し、そうでない場合は完全リフレッシュを実行するときは、FORCE を指定します。

ON COMMIT 句

高速リフレッシュを実行して、マテリアライズド・ビューのマスター表に対するトランザクションをコミットするときは、必ず ON COMMIT を指定します。

制限事項： この句は、マテリアライズド結合ビューおよびマテリアライズド集計ビューにのみサポートされます。

参照： 『Oracle9i レプリケーション』および『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

ON DEMAND 句

マテリアライズド・ビューを、3つの DBMS_MVIEW リフレッシュ・プロシージャのいずれかのコールによる要求でリフレッシュする場合は、ON DEMAND を指定します。ON COMMIT および ON DEMAND のどちらも指定しなかった場合、ON DEMAND がデフォルト値になります。

参照：

- これらのプロシージャの詳細は、『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。
- REFRESH ON DEMAND を指定することによって作成できるマテリアライズド・ビューのタイプについては、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

注意： ON COMMIT または ON DEMAND を指定した場合、START WITH または NEXT を同時に指定できません。

START WITH 句

START WITH *date* を指定すると、最初の自動リフレッシュ時間を表す日付を指定できます。

NEXT 句

NEXT を指定すると、自動リフレッシュの間隔を計算するための日付式を指定できます。

START WITH 値および NEXT 値は、将来の時間に評価される値です。START WITH 値を省略した場合、Oracle はマテリアライズド・ビューの作成時に NEXT 式を評価することによって、最初の自動リフレッシュ時間を判断します。START WITH 値を指定し、NEXT 値を指定しない場合、Oracle は1回のみマテリアライズド・ビューをリフレッシュします。START WITH 値および NEXT 値を両方とも指定しない場合、または `alter my_refresh_clause` 自体を指定しない場合は、マテリアライズド・ビューは自動リフレッシュされません。

WITH PRIMARY KEY 句

WITH PRIMARY KEY を指定すると、ROWID のマテリアライズド・ビューを主キーのマテリアライズド・ビューに変更できます。主キーのマテリアライズド・ビューを使用すると、高速リフレッシュを継続できるマテリアライズド・ビューの機能に影響せずに、マテリアライズド・ビュー・マスター表を再編成できます。マスター表には、使用可能な主キー制約が定義されている必要があります。

参照： 主キーのマテリアライズド・ビューの詳細は、『Oracle9i レプリケーション』を参照してください。

USING ROLLBACK SEGMENT 句

USING ROLLBACK SEGMENT を指定すると、マテリアライズド・ビューのリフレッシュ中に使用するリモート・ロールバック・セグメントを変更できます。使用するロールバック・セグメント名を `rollback_segment` に指定します。

参照： DBMS_REFRESH パッケージを使用するローカル・マテリアライズド・ビュー・ロールバック・セグメントの変更については、『Oracle9i レプリケーション』を参照してください。

DEFAULT 使用するロールバック・セグメントを Oracle に選択させる場合、DEFAULT 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 ... rollback_segment リモート・マスターで個々のマテリアライズド・ビュー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。(ローカル・マテリアライズド・ビュー・ロールバック・セグメントを変更する場合は、DBMS_REFRESH パッケージを使用します。詳細は、『Oracle9i レプリケーション』を参照してください。)

マスター・ロールバック・セグメントは、各マテリアライズド・ビューに格納され、マテリアライズド・ビューの作成およびリフレッシュ時にスキャンされます。複合マテリアライズド・ビューの場合、マスター・ロールバック・セグメントの指定は無視されます。

QUERY REWRITE 句

この句を使用して、マテリアライズド・ビューがクエリー・リライトで使用できるかどうかを判断します。

ENABLE 句

ENABLE を指定すると、クエリー・リライトに対してマテリアライズド・ビューが使用可能になります。

制限事項：

- マテリアライズド・ビューが無効または使用禁止の場合、ENABLE モードでもクエリー・リライトに適応しません。
- マテリアライズド・ビューの全体または一部がビューから作成されている場合、クエリー・リライトを使用可能にできません。
- マテリアライズド・ビューのすべてのユーザー定義ファンクションが DETERMINISTIC である場合のみ、クエリー・リライトを使用可能にできます。

参照： 12-47 ページの「[CREATE FUNCTION](#)」を参照してください。

- 文内の式が反復可能な場合のみ、クエリー・リライトを使用可能にできます。たとえば、CURRENT_TIME または USER を含めることはできません。

参照： クエリー・リライトの詳細は、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

DISABLE 句

マテリアライズド・ビューがクエリー・リライトで使えないようにする場合は、DISABLE を指定します（マテリアライズド・ビューが無効な場合、使用禁止であるかどうかにかかわらず、クエリー・リライトの使用には適応しません）。ただし、使用禁止にされたマテリアライズド・ビューをリフレッシュすることはできます。

COMPILE

COMPILE を指定すると、マテリアライズド・ビューを明示的に再検証します。マテリアライズド・ビューが依存するオブジェクトを削除または変更した場合、マテリアライズド・ビューはアクセス可能のままですが、クエリー・リライトに対しては無効です。再度、明示的にマテリアライズド・ビューの妥当性チェックを行い、クエリー・リライトの使用に適応させるには、この句を使用します。

マテリアライズド・ビューの再妥当性チェックに失敗すると、リフレッシュできなくなるか、またはクエリー・リライトに使用できなくなります。

CONSIDER FRESH

この句を使用すると、マスター表が変更された後のマテリアライズド・ビューの失効状態を管理することができます。CONSIDER FRESH は、マテリアライズド・ビューがフレッシュであり、TRUSTED または STALE_TOLERATED モードでのクエリー・リライトに適応するとみなされるように指定します。Oracle は、マテリアライズド・ビューがフレッシュであるかどうかを保証できないため、ENFORCED モードでのクエリー・リライトはサポートしません。また、この句はマテリアライズド・ビューの失効状態を UNKNOWN に設定します。失効状態は、ALL_MVIEWS、DBA_MVIEWS および USER_MVIEWS の各データ・ディクショナリ・ビューの STALENESS 列に表示されます。

注意： いずれかのマスター表の内容が変更された場合、マテリアライズド・ビューは失効します。この句は、Oracle に対して、マテリアライズド・ビューがフレッシュで、変更されていないものと仮定するように指示します。そのため、リフレッシュが保留されているこれらの表に対する実際の更新内容は、マテリアライズド・ビューから削除されます。

参照： クエリー・リライトの詳細、およびマスター表へのパーティション・メンテナンス操作の影響については、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

例

自動リフレッシュの例 次の文は、sales_by_month_by_state マテリアライズド・ビュー（13-24 ページの「[マテリアライズド集計ビューの例](#)」で作成）のリフレッシュ方法のデフォルトを FAST に変更します。

```
ALTER MATERIALIZED VIEW sales_by_month_by_state
  REFRESH FAST;
```

これ以降のマテリアライズド・ビューの自動リフレッシュは、高速リフレッシュになります。これは、単純なマテリアライズド・ビューであり、そのマスター表には、マテリアライズド・ビューの作成前または最後のリフレッシュ前に作成されたマテリアライズド・ビュー・ログがあります。

REFRESH 句に START WITH または NEXT の値が指定されていないため、sales_by_month_by_state マテリアライズド・ビューを作成したとき、または最後に変更したときに REFRESH 句で設定されたリフレッシュ間隔がそのまま使用されます。

NEXT の例 次の文は、マテリアライズド・ビュー branch_emp の新しい自動リフレッシュ間隔を格納します。

```
ALTER MATERIALIZED VIEW branch_emps
  REFRESH NEXT SYSDATE+7;
```

REFRESH 句に START WITH の値が指定されていないため、マテリアライズド・ビュー branch_emp を作成したとき、または最後に変更したときに、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。

このマテリアライズド・ビューは、次に自動リフレッシュが行われる際に、リフレッシュされます。次に自動リフレッシュが行われる日時は、NEXT に設定した式 SYSDATE+7 が計算されて決まります。その後は、週に 1 回リフレッシュが自動的に行われます。

REFRESH 句にリフレッシュ方法が明示的に指定されていないため、直前の CREATE MATERIALIZED VIEW 文または ALTER MATERIALIZED VIEW 文の REFRESH 句で指定されたリフレッシュ方法が引き続き使用されます。

完全リフレッシュの例 次の文は、sf_emp マテリアライズド・ビューの新しいリフレッシュ方法、NEXT で示す新しいリフレッシュ日時および新しい自動リフレッシュ間隔を指定します。

```
ALTER MATERIALIZED VIEW 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、このマテリアライズド・ビューの次の自動リフレッシュは翌日の午前 9 時に発生するように設定されます。この日時にマテリアライズド・ビューの完全リフレッシュが実行され、NEXT に設定した式が計算されます。その後は、このマテリアライズド・ビューは毎週リフレッシュされます。

クエリー・リライトを使用可能にする例 次の文は、マテリアライズド・ビュー mv1 のクエリー・リライトを使用可能にし、暗黙的に再妥当性チェックを行います。

```
ALTER MATERIALIZED VIEW mv1
  ENABLE QUERY REWRITE;
```

ロールバック・セグメントの例 次の文は、マテリアライズド・ビュー・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 MATERIALIZED VIEW inventory
  REFRESH USING MASTER ROLLBACK SEGMENT master_seg;
```

次の文は、マテリアライズド・ビュー・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 MATERIALIZED VIEW sales_mv
  REFRESH USING DEFAULT MASTER ROLLBACK SEGMENT;
```

主キーの例 次の文は、ROWID のマテリアライズド・ビューを主キーのマテリアライズド・ビューに変更します。

```
ALTER MATERIALIZED VIEW emp_rs
  REFRESH WITH PRIMARY KEY;
```

COMPILE の例 次の文は、マテリアライズド・ビュー store_mv を再検証します。

```
ALTER MATERIALIZED VIEW store_mv COMPILE;
```

リフレッシュ方法の変更例 次の文は、マテリアライズド・ビュー store_mv のリフレッシュ方法を FAST に変更します。

```
ALTER MATERIALIZED VIEW store_mv REFRESH FAST;
```

CONSIDER FRESH の例 次の文は、Oracle にマテリアライズド・ビュー mv1 をフレッシュとみなすように指定します。この文を使用した場合、mv1 のマスター表に対するパーティション・メンテナンス操作を実行した後でも、mv1 は TRUSTED モードでのクエリー・リライトに適応します。

```
ALTER MATERIALIZED VIEW mv1 CONSIDER FRESH;
```

ALTER MATERIALIZED VIEW LOG

用途

ALTER MATERIALIZED VIEW LOG 文を使用すると、記憶特性、リフレッシュ・モードまたはリフレッシュ時刻、または既存のマテリアライズド・ビュー・ログのタイプを変更できます。**マテリアライズド・ビュー・ログ**とは、マテリアライズド・ビューのマスター表に関連付けられる表です。

注意： キーワード SNAPSHOT は、MATERIALIZED VIEW のかわりに下位互換用にサポートされています。

参照：

- マテリアライズド・ビューのリフレッシュ方法など、マテリアライズド・ビューの詳細は、8-74 ページの「[ALTER MATERIALIZED VIEW](#)」を参照してください。
- マテリアライズド・ビューの様々なタイプの詳細は、13-5 ページの「[CREATE MATERIALIZED VIEW](#)」を参照してください。

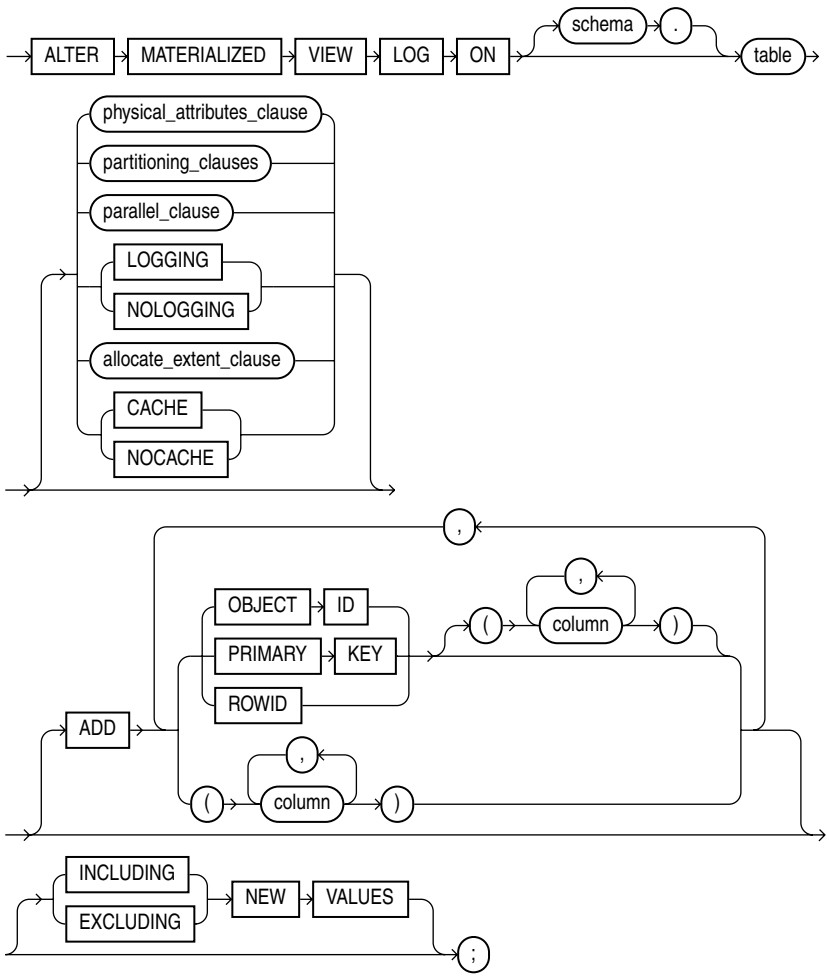
前提条件

マスター表の所有者、またはマスター表に対する SELECT 権限およびマテリアライズド・ビュー・ログに対する ALTER 権限を持つユーザーのみがマテリアライズド・ビュー・ログを変更できます。

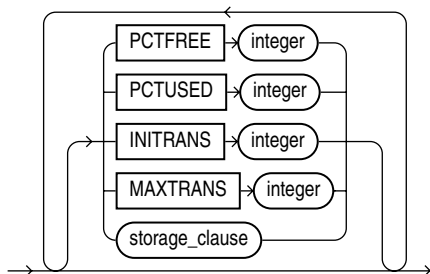
参照： ALTER MATERIALIZED VIEW LOG の前提条件の詳細は、『Oracle9i レプリケーション』を参照してください。

構文

alter_materialized_view_log::=



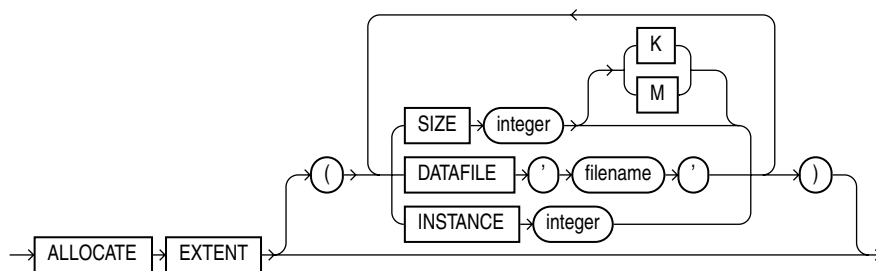
physical_attributes_clause::=



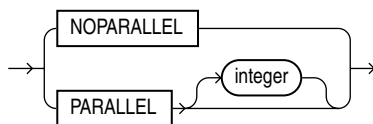
storage_clause: 17-50 ページの「[storage_clause](#)」を参照してください。

partitioning_clauses: 14-36 ページの「[table_partitioning_clauses](#)」を参照してください。

allocate_extent_clause::=



parallel_clause::=



キーワードとパラメータ

schema

マスター表が定義されているスキーマを指定します。*schema* を指定しないと、マテリアライズド・ビュー・ログは自分のスキーマ内にあるとみなされます。

table

変更するマテリアライズド・ビュー・ログに関連付けられたマスター表の名前を指定します。

physical_attributes_clause

physical_attributes_clause は、表、パーティションまたはオーバーフロー・データ・セグメントに対する PCTFREE、PCTUSED、INITRANS および MAXTRANS の各パラメータの値、またはパーティション表のデフォルト特性を変更します。

参照： これらのパラメータについては、14-6 ページの「[CREATE TABLE](#)」および 8-96 ページの「[マテリアライズド・ビュー記憶域の例](#)」を参照してください。

partitioning_clauses

partitioning_clauses の構文および一般的な機能は、ALTER TABLE 文と同じです。

***partitioning_clauses* の制限事項**

- マテリアライズド・ビュー・ログのパーティションを変更する場合、*LOB_storage_clause* または *modify_LOB_storage_clause* は使用できません。
- マテリアライズド・ビュー・ログ・パーティションを削除、切捨てまたは交換すると、Oracle はエラーを戻します。

参照： 10-2 ページの「[ALTER TABLE](#)」を参照してください。

parallel_clause

parallel_clause は、マテリアライズド・ビュー・ログへのパラレル操作がサポートされているかどうかを指定します。

注意： *parallel_clause* 構文は、以前のリリースの構文に代わるものです。以前のリリースの構文は下位互換用にサポートされていますが、動作がわずかに異なることがあります。

NOPARALLEL シリアル実行を行う場合は、NOPARALLEL を指定します。これはデフォルトです。

PARALLEL 並列度を選択する場合は、PARALLEL を指定します。並列度とは、すべての関係するインスタンスで使用可能な CPU の数に、初期化パラメータ PARALLEL_THREADS_PER_CPU の値を掛けたものです。

PARALLEL integer integer を指定すると、パラレル操作で使用されるパラレル・スレッド数である**並列度**が指定されます。各パラレル・スレッドは、1、2 個のパラレル実行サーバーを使用します。通常、最適な並列度が計算されるため、integer に値を指定する必要はありません。

参照： 詳細は、14-43 ページの「CREATE TABLE」の「[parallel_clause](#)に関する注意事項」を参照してください。

LOGGING | NOLOGGING

マテリアライズド・ビュー・ログに対するロギング属性を指定します。

参照： この属性の指定については、10-2 ページの「[ALTER TABLE](#)」を参照してください。

allocate_extent_clause

allocate_extent_clause は、マテリアライズド・ビュー・ログの新しいエクステンツを明示的に割り当てます。

参照： 10-2 ページの「[ALTER TABLE](#)」を参照してください。

CACHE | NOCACHE 句

アクセス頻度が高いデータについて、CACHE は、フル・テーブル・スキャンの実行時にこのログ用に取り出された各ブロックを、バッファ・キャッシュ内の LRU リストの最高使用頻度側に入れることを指定します。この属性は、小規模な参照表で有効です。NOCACHE は、ブロックを LRU リストの最低使用頻度側に入れることを指定します。

参照： CACHE または NOCACHE の指定については、14-6 ページの「[CREATE TABLE](#)」を参照してください。

ADD 句

ADD 句を使用すると、マテリアライズド・ビュー・マスター表内の行が変更される際に、主キー値、ROWID 値またはオブジェクト ID 値も記録できるようにマテリアライズド・ビュー・ログを拡張できます。また、この句は、新しく列を記録するためにも使用できます。

これらの情報の記録を停止する場合は、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。マテリアライズド・ビュー・ログを削除した後再作成した場合、マスター表に依存するすべての既存マテリアライズド・ビューが、次のリフレッシュ時に強制的に完全リフレッシュされます。

制限事項：各マテリアライズド・ビュー・ログに指定できるのは、PRIMARY KEY、ROWID、OBJECT ID および列リストを 1 つずつです。そのため、これら 3 つの値のいずれかが作成時に（暗黙的または明示的に）指定された場合、この ALTER 文にはそれらの値を指定できません。

OBJECT ID 更新されるすべての行の適切なオブジェクト識別子をマテリアライズド・ビュー・ログに記録する場合は、OBJECT ID を指定します。

制限事項：OBJECT ID は、オブジェクト表のログ用のみに指定でき、記憶表用には指定できません。

PRIMARY KEY 更新されるすべての行の主キーをマテリアライズド・ビュー・ログに記録する場合は、PRIMARY KEY を指定します。

ROWID 更新されるすべての行の ROWID 値をマテリアライズド・ビュー・ログに記録する場合は、ROWID を指定します。

column 更新されるすべての行に対して、マテリアライズド・ビュー・ログに記録する値を持つ新しい列を指定します。通常、フィルタ列（マテリアライズド・ビューが参照する非主キー列）および結合列（その後続く WHERE 句で結合を定義する非主キー列）を指定します。

参照：

- マテリアライズド・ビュー・ログの値の明示的および暗黙的な論理和については、13-5 ページの「[CREATE MATERIALIZED VIEW](#)」を参照してください。
- フィルタ列および結合列については、『Oracle9i レプリケーション』を参照してください。

NEW VALUES 句

NEW VALUES 句は、マテリアライズド・ビュー・ログに古い値と新しい値の両方を保存するかを指定できます。ALTER MATERIALIZED VIEW LOG 文で追加した主キー、ROWID または列のみでなく、ログのすべての列にこの句で設定した値を適用します。

INCLUDING INCLUDING を指定すると、ログに新しい値と古い値を保存できます。このログが単一表マテリアライズド集計ビューの表用で、マテリアライズド・ビューに高速リフレッシュを実行する場合、INCLUDING を指定してください。

EXCLUDING EXCLUDING を指定すると、ログに対する新しい値の記録を使用禁止にできます。この句を使用すると、新しい値の記録によるオーバーヘッドを回避できます。ただし、この表に高速リフレッシュが可能な単一表マテリアライズド集計ビューが定義されている場合は、この句を使用しないでください。

例

マテリアライズド・ビュー記憶域の例 次の文は、マテリアライズド・ビュー・ログの MAXEXTENTS の値を変更します。

```
ALTER MATERIALIZED VIEW LOG ON sales
    STORAGE (MAXEXTENTS 50);
```

PRIMARY KEY の例 次の文は、主キー情報も記録するように既存のマテリアライズド・ビュー・ログを変更します。

```
ALTER MATERIALIZED VIEW LOG ON orders
    ADD PRIMARY KEY;
```


ALTER OUTLINE

用途

ストアド・アウトラインの名前を変更する場合、ストアド・アウトラインを異なるカテゴリに再度割り当てる場合、または、アウトラインの SQL 文をコンパイルし、古いアウトライン・データを現行の条件で作成したアウトラインと置き換えて、ストアド・アウトラインを再生成する場合に、ALTER OUTLINE 文を使用します。

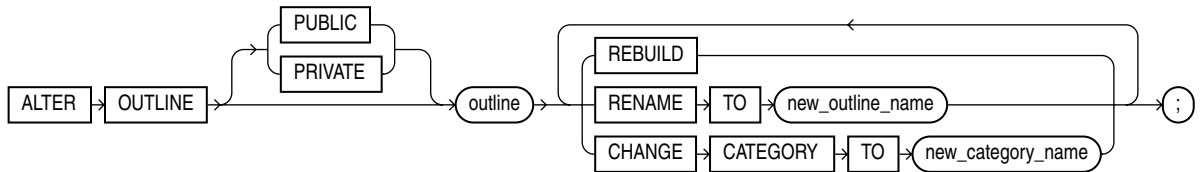
参照： アウトラインの詳細は、13-42 ページの「CREATE OUTLINE」および『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。

前提条件

アウトラインを変更する場合は、ALTER ANY OUTLINE システム権限が必要です。

構文

alter_outline::=



キーワードとパラメータ

PUBLIC | PRIVATE

アウトラインのパブリック・バージョンを変更するには、PUBLIC を指定します。これはデフォルト値です。

現行のセッションに対してプライベートで、現行の解析スキーマにデータが格納されているアウトラインを変更する場合は、PRIVATE を指定します。

outline

変更するアウトラインの名前を指定します。

REBUILD

REBUILD を指定すると、現行の条件で、*outline* の実行計画が再生成されます。

RENAME TO 句

RENAME TO 句を使用すると、*outline* の値と置き換えるアウトライン名を指定できます。

CHANGE CATEGORY TO 句

CHANGE CATEGORY TO 句を使用すると、*outline* の移動先となるカテゴリ名を指定できます。

例

ALTER OUTLINE の例 次の文は、アウトラインのテキストをコンパイルし、古いアウトライン・データを現行の条件で作成したアウトラインと置き換えて、**salaries** というストア・アウトラインを再生成します。

```
ALTER OUTLINE salaries REBUILD;
```

ALTER PACKAGE

用途

ALTER PACKAGE 文を使用すると、パッケージ仕様部またはパッケージ本体のいずれか（あるいはその両方）を明示的に再コンパイルします。明示的に再コンパイルすることによって、実行時に暗黙的に再コンパイルする必要がなくなり、また、実行時のコンパイル・エラーとパフォーマンス上のオーバーヘッドも回避できます。

パッケージ中のすべてのオブジェクトは、1つの単位として格納されているため、ALTER PACKAGE 文によって、すべてのパッケージ・オブジェクトがまとめて再コンパイルされます。ALTER PROCEDURE 文または ALTER FUNCTION 文を使用して、パッケージ中の一部のプロシージャまたはファンクションを再コンパイルすることはできません。

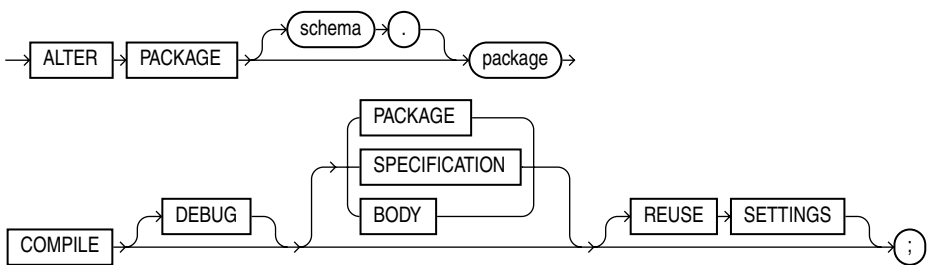
注意： この文を使用して、既存のパッケージの宣言や定義を変更することはできません。パッケージを再宣言または再定義する場合は、「CREATE PACKAGE」または 13-46 ページの「CREATE PACKAGE BODY」に OR REPLACE 句を指定します。

前提条件

パッケージを変更するには、パッケージが自分のスキーマ内にあるか、または ALTER ANY PROCEDURE システム権限が必要です。

構文

alter_package::=



キーワードとパラメータ

schema

パッケージが定義されているスキーマを指定します。*schema* を指定しない場合、パッケージが自分のスキーマ内に定義されているとみなされます。

package

再コンパイルするパッケージの名前を指定します。

COMPILE

パッケージ仕様部およびパッケージ本体を再コンパイルする場合は、**COMPILE** を指定する必要があります。**COMPILE** キーワードは必須です。

再コンパイル中、**Oracle** はすべての永続コンパイラのスイッチ設定を削除し、セッションからそれらを再び取得してコンパイルの終了時に格納します。この手順を回避するには、**REUSE SETTINGS** 句を指定します。

パッケージ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。**SQL*Plus** コマンド **SHOW ERRORS** を使用して、関連するコンパイラ・エラー・メッセージを表示できます。

SPECIFICATION

SPECIFICATION を指定すると、無効かどうかにかかわらず、パッケージ仕様部のみを再コンパイルします。パッケージ仕様部を変更した場合は、コンパイル・エラーの有無を確認するために再コンパイルします。

その結果、そのパッケージ中のプロシージャまたはファンクション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、明示的に再コンパイルせずに、これらの依存オブジェクトを参照した場合、**Oracle** は実行時にそれらを暗黙的に再コンパイルします。

BODY

BODY を指定すると、無効かどうかにかかわらず、パッケージ本体のみを再コンパイルします。パッケージ本体は、変更後に再コンパイルしてください。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なりません。

パッケージ本体を再コンパイルした場合、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に終了した場合、この本体は有効になります。

PACKAGE

PACKAGE を指定すると、有効か無効かにかかわらず、パッケージ仕様部およびパッケージ本体（存在する場合）の両方を再コンパイルします。これはデフォルトです。パッケージ仕様部およびパッケージ本体を再コンパイルした場合、前述の SPECIFICATION および BODY で説明するとおり、再妥当性チェックおよび再コンパイルが必要になります。

参照： リモート・オブジェクトなどのスキーマ・オブジェクト間の依存性を Oracle が管理する方法については、『Oracle9i データベース概要』を参照してください。

DEBUG

DEBUG を指定すると、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格納するように指示できます。

参照： パッケージのデバッグの詳細は、『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

REUSE SETTINGS

REUSE SETTINGS を指定すると、Oracle によるコンパイラ・スイッチ設定の削除および再取得を回避できます。この句では既存の設定が持続され、その設定で再コンパイルされます。

DEBUG と REUSE SETTINGS の両方を指定する場合は、PLSQL_COMPILER_FLAGS パラメータの永続格納値が INTERPRETED, DEBUG に設定されます。その他のコンパイラ・スイッチ値は変更されません。

参照： PLSQL_COMPILER_FLAGS パラメータと COMPILE 句の相互作用の詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』および『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

例

パッケージの再コンパイル例 次の文は、スキーマ blair 内の accounting パッケージの仕様部および本体を明示的に再コンパイルします。

```
ALTER PACKAGE blair.accounting  
    COMPIL PACKAGE;
```

accounting の仕様部および本体の再コンパイル時に、コンパイル・エラーが発生しなかった場合、accounting は有効になります。その後、実行時に再コンパイルしなくても、accounting の仕様部に宣言されたすべてのパッケージ・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が戻り、accounting は無効のままです。

また、`accounting` に依存しているすべてのオブジェクトが無効になります。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した場合、**Oracle** は、実行時にそれらを暗黙的に再コンパイルします。

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、この本体は有効になります。その後、実行時に再コンパイルしなくても、`accounting` の仕様部に宣言されたすべてのパッケージ・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。

ALTER PROCEDURE

用途

ALTER PROCEDURE 文を使用すると、スタンドアロンのストアド・プロシージャを明示的に再コンパイルできます。明示的に再コンパイルすることによって、実行時に暗黙的に再コンパイルする必要がなくなり、また、実行時のコンパイル・エラーとパフォーマンス上のオーバーヘッドも回避できます。

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場合、ALTER PACKAGE 文を使用して、そのパッケージ全体を再コンパイルします (8-99 ページの「ALTER PACKAGE」を参照)。

注意： この文を使用して、既存のプロシージャの宣言または定義を変更することはできません。プロシージャを再宣言または再定義する場合は、OR REPLACE 句を指定して CREATE PROCEDURE 文を使用します (13-58 ページの「CREATE PROCEDURE」を参照)。

ALTER PROCEDURE 文は、ALTER FUNCTION 文と似ています。

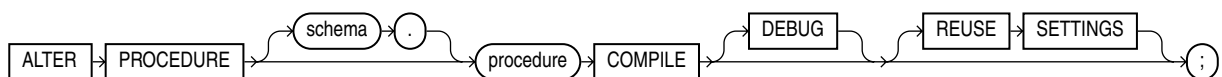
参照： 8-46 ページの「ALTER FUNCTION」を参照してください。

前提条件

プロシージャは、自分のスキーマ内にある必要があります。自分のスキーマ内がない場合は、ALTER ANY PROCEDURE システム権限が必要です。

構文

alter_procedure::=



キーワードとパラメータ

schema

プロシージャが定義されているスキーマを指定します。*schema* を指定しない場合、プロシージャが自分のスキーマ内に定義されているとみなされます。

procedure

再コンパイルするプロシージャの名前を指定します。

COMPILE

COMPILE を指定すると、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。プロシージャが有効か無効かにかかわらず、プロシージャは再コンパイルされます。

- いずれかのオブジェクトが無効の場合、プロシージャが依存するオブジェクトが最初に再コンパイルされます。
- プロシージャに依存するすべてのローカル・オブジェクト（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、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に終了すると、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。SQL*Plus コマンド SHOW ERRORS を使用して、関連するコンパイラ・エラー・メッセージを表示できます。

再コンパイル中、Oracle はすべての永続コンパイラのスイッチ設定を削除し、セッションからそれらを再び取得してコンパイルの終了時に格納します。この手順を回避するには、REUSE SETTINGS 句を指定します。

参照： リモート・オブジェクトを含むスキーマ・オブジェクト間の依存性を Oracle が管理する方法については、『Oracle9i データベース概要』を参照してください。

DEBUG

DEBUG を指定すると、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格納するように指示できます。

参照： これらのプロシージャについては、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

REUSE SETTINGS

REUSE SETTINGS を指定すると、Oracle によるコンパイラ・スイッチ設定の削除および再取得を回避できます。この句では既存の設定が持続され、その設定で再コンパイルされます。

DEBUG と REUSE SETTINGS の両方を指定する場合は、PLSQL_COMPILER_FLAGS パラメータの永続格納値が INTERPRETED, DEBUG に設定されます。その他のコンパイラ・スイッチ値は変更されません。

参照： PLSQL_COMPILER_FLAGS パラメータと COMPILE 句の相互作用の詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』および『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

例

プロシージャの再コンパイル例 次の文は、ユーザー oe が所有するプロシージャ credit を明示的に再コンパイルします。

```
ALTER PROCEDURE oe.credit  
    COMPILE;
```

credit の再コンパイル時にエラーが発生しなければ、credit は有効になります。その後、Oracle は、実行時にそれを再コンパイルしなくても実行できます。credit 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、credit 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。これらのオブジェクトとは、credit をコールするプロシージャ、ファンクション、パッケージ本体などです。その後、明示的に再コンパイルせずに、これらのオブジェクトを参照した場合、Oracle は、実行時にそれらを暗黙的に再コンパイルします。

ALTER PROFILE

用途

ALTER PROFILE 文を使用すると、プロファイルのリソース制限またはパスワード管理パラメータを追加、変更または削除できます。

ALTER PROFILE 文を使用すると、プロファイルに対して行った変更は、このプロファイルの現行のセッションのユーザーには影響しません。後続セッションのユーザーのみに影響します。

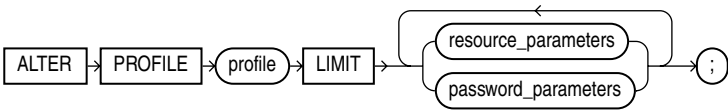
参照： プロファイルの作成の詳細は、13-65 ページの「[CREATE PROFILE](#)」を参照してください。

前提条件

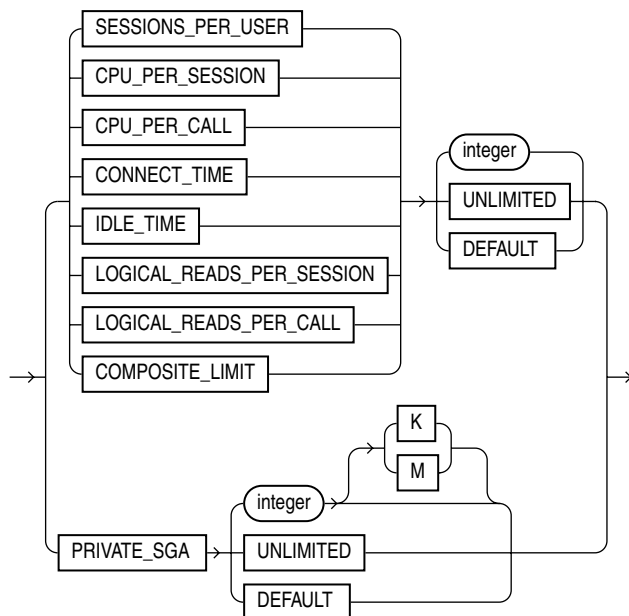
プロファイルのリソース制限を変更する場合は、ALTER PROFILE システム権限が必要です。パスワード制限および保護を変更する場合は、ALTER PROFILE および ALTER USER システム権限が必要です。

構文

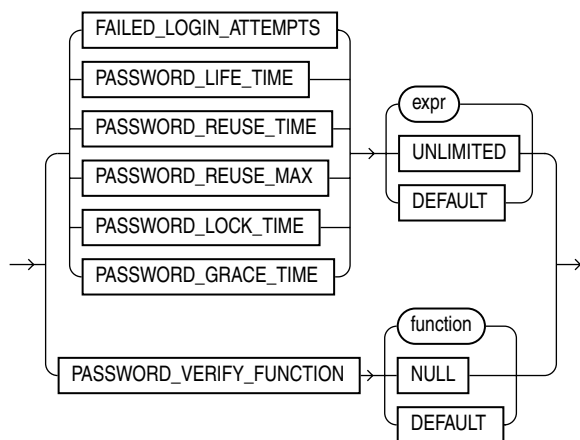
`alter_profile::=`



resource_parameters::=



password_parameters::=



キーワードとパラメータ

ALTER PROFILE 文のキーワードとパラメータの意味は、すべて CREATE PROFILE 文のキーワードとパラメータと同じです。

注意： DEFAULT プロファイルから制限を削除することはできません。

参照： 13-65 ページの「[CREATE PROFILE](#)」を参照してください。

例

パスワードを無効にする例 次の文は、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;
```

デフォルト値の設定例 次の文は、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;
```

ログインおよびパスワード・ロック時刻の制限例 次の文は、プロファイル prof の FAILED_LOGIN_ATTEMPTS を 5 に設定し、PASSWORD_LOCK_TIME を 1 に変更します。

```
ALTER PROFILE prof LIMIT
  FAILED_LOGIN_ATTEMPTS 5
  PASSWORD_LOCK_TIME 1;
```

この文を使用すると、ログインに 5 回失敗した場合に、prof のアカウントは 1 日ロックされます。

パスワードの有効期限および猶予期間の変更例 次の文は、プロファイル 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;
```

同時セッションの制限例 次の文は、プロファイル `engineer` に同時実行のセッションの新しい制限 5 を指定します。

```
ALTER PROFILE engineer LIMIT SESSIONS_PER_USER 5;
```

現在、プロファイル `engineer` に `SESSIONS_PER_USER` の制限が定義されていない場合は、このプロファイルに制限 5 が追加されます。プロファイルに制限が定義されている場合は、前述の文によってその制限が 5 に再定義されます。プロファイル `engineer` が割り当てられているすべてのユーザーは、同時実行のセッションが 5 件に制限されます。

制限の削除例 次の文は、プロファイル `engineer` の `IDLE_TIME` の制限を削除します。

```
ALTER PROFILE engineer LIMIT IDLE_TIME DEFAULT;
```

プロファイル `engineer` が割り当てられているユーザーは、後続セッションからはプロファイル `DEFAULT` に定義された `IDLE_TIME` の制限に従います。

アイドル時間の制限例 次の文は、プロファイル `DEFAULT` にアイドル時間の制限（2 分）を定義します。

```
ALTER PROFILE default LIMIT IDLE_TIME 2;
```

`IDLE_TIME` の制限は、次のユーザーに適用されます。

- プロファイルが明示的に割り当てられていないユーザー
- `IDLE_TIME` の制限が定義されていないプロファイルが明示的に割り当てられているユーザー

次の文は、プロファイル `engineer` に無制限のアイドル時間を設定します。

```
ALTER PROFILE engineer LIMIT IDLE_TIME UNLIMITED;
```

プロファイル `engineer` を割り当てられているすべてのユーザーは、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。

ALTER RESOURCE COST

用途

ALTER RESOURCE COST 文を使用すると、セッションで使用するリソース・コストの合計を算出するための式を指定または変更できます。

Oracle は、その他のリソースの使用も監視しますが、セッションに対するリソース・コストの合計は、この構文の 4 種類のリソースに基づいて算出されます。

リソース・コストの合計を算出するための式を指定した場合、CREATE PROFILE 文の COMPOSITE_LIMIT パラメータを使用して、セッションに対するコストを制限できます。セッションのコストが制限を超えた場合、セッションは異常終了し、エラーが戻ります。各リソースに割り当てた重みを変更するために ALTER RESOURCE COST 文を使用した場合、現行のセッション以降のすべてのセッションで、その新しい重みを基にリソース・コストが計算されます。

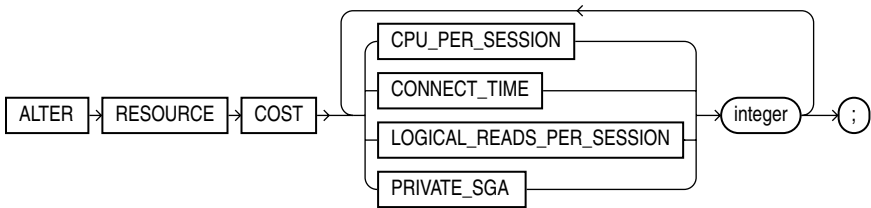
参照： リソースの合計およびリソース制限の設定については、13-65 ページの「CREATE PROFILE」を参照してください。

前提条件

ALTER RESOURCE COST システム権限が必要です。

構文

alter_resource_cost::=



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 時間を、100 分の 1 秒単位で指定します。

CONNECT_TIME

セッションの経過時間を分単位で指定します。

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るデータ・ブロックの数を指定します。

PRIVATE_SGA

セッションごとに使用できるシステム・グローバル領域（SGA）内のプライベート領域のバイト数を指定します。共有サーバー・アーキテクチャを使用して、セッション用として SGA 内でプライベート領域を割り当てている場合のみ、この制限が適用されます。

integer

各リソースの重みを指定します。各リソースに割り当てる重みによって、各リソースがリソース・コストの合計に影響する程度が決定されます。リソースに重みを割り当てない場合は、デフォルト値の 0（ゼロ）が適用され、コストへの影響はありません。重みを割り当てた場合は、データベースの次のセッション以降のすべてのセッションで、その重みが適用されます。

最初にセッションで使用された各リソースの量にそのリソースの重みを乗算し、次に、4 種類のリソースの乗算結果を加算することによって、リソース・コストの合計が計算されます。どのセッションについても、このコストは、ユーザーのプロファイル内の COMPOSITE_LIMIT パラメータの値によって制限されます。乗算結果と総コストは、ともに **サービス単位** と呼ばれる単位で表されます。

例

リソース・コストの変更例 次の文は、リソース CPU_PER_SESSION と CONNECT_TIME に重みを割り当てます。

```
ALTER RESOURCE COST
  CPU_PER_SESSION 100
  CONNECT_TIME      1;
```

この重みによって、セッションごとに次のコスト計算式が設定されます。

$$\text{cost} = (100 * \text{CPU_PER_SESSION}) + (1 * \text{CONNECT_TIME})$$

CPU_PER_SESSION および CONNECT_TIME の値は、DEFAULT プロファイルまたはセッションのユーザーのプロファイルにある値のいずれかです。

ここでは、リソース LOGICAL_READS_PER_SESSION および PRIVATE_SGA に重みを割り当てていないため、これらのリソースは式に含まれません。

プロファイルで COMPOSITE_LIMIT 値として 500 を割り当てた場合、cost が 500 を超えると、必ず、セッションはこの制限を超えます。たとえば、CPU 時間 0.04 秒、経過時間 101 分を使用するセッションは、この制限を超えます。同様に、CPU 時間が 0.0301 秒、経過時間が 200 分のセッションもこの制限を超えます。

一度割り当てた重みは、次のように、別の ALTER RESOURCE 文を発行することによって変更できます。

```
ALTER RESOURCE COST
    LOGICAL_READS_PER_SESSION 2
    CONNECT_TIME 0;
```

新しく割り当てた重みによって、新しいコスト計算式が設定されます。

$$\text{cost} = (100 * \text{CPU_PER_SESSION}) + (2 * \text{LOGICAL_READ_PER_SECOND})$$

CPU_PER_SESSION および LOGICAL_READS_PER_SECOND の値は、DEFAULT プロファイルまたはセッションのユーザーのプロファイルにある値のいずれかです。

この ALTER RESOURCE COST 文によって、式は次のように変更されます。

- CPU_PER_SESSION リソースの重みは指定しないが、このリソースにはすでに重みが割り当てられているため、式では先に指定した重みがそのまま使用されます。
- LOGICAL_READS_PER_SESSION リソースに重みを割り当てたため、このリソースが式で使用されます。
- CONNECT_TIME リソースに 0（ゼロ）を割り当てたため、このリソースは式に含まれていません。
- PRIVATE_SGA リソースの重みを指定せず、かつ、このリソースには重みが指定されていないため、このリソースは式で使用されません。

ALTER ROLE

用途

ALTER ROLE 文を使用すると、ロールを使用可能にするために必要な許可を変更できます。

参照：

- ロールの作成の詳細は、13-72 ページの「[CREATE ROLE](#)」を参照してください。
- セッションのロールを使用可能または使用禁止にする方法については、17-43 ページの「[SET ROLE](#)」を参照してください。

前提条件

ロールに ADMIN OPTION が付与されている必要があります。付与されていない場合は、ALTER ANY ROLE システム権限が必要です。

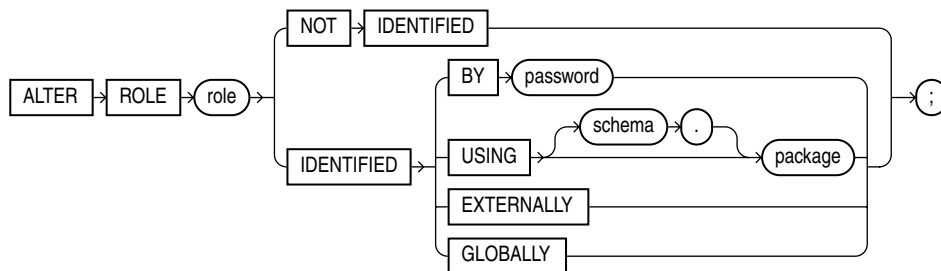
ロールを IDENTIFIED GLOBALLY に変更する前に、次の作業が必要です。

- ロールに対して外部的に識別されたロール権限をすべて取り消します。
- すべてのユーザー、ロールおよび PUBLIC からロールの付与を取り消します。

この規則の唯一の例外として、現在ルールを変更しているユーザーからはそのルールを取り消さないでください。

構文

alter_role::=



キーワードとパラメータ

ALTER ROLE 文のキーワードとパラメータの意味は、すべて CREATE ROLE 文のキーワードとパラメータと同じです。

注意：

- ロールを変更しても、ロールがすでに使用可能になっているユーザー・セッションには影響しません。
 - パスワードによって識別されるロールをアプリケーション・ロールに変更する場合 (USING *package* 句を使用)、ロールに対応付けられたパスワード情報は失われます。次にロールが使用可能になるときから、新しい認証方式が使用されます。
 - ALTER ANY ROLE システム権限を持つユーザーが、IDENTIFIED GLOBALLY ロールを IDENTIFIED BY *password*、IDENTIFIED EXTERNALLY または NOT IDENTIFIED に変更すると、非グローバルなロールを作成した場合と同様に、そのユーザーに変更されたロールが ADMIN OPTION 付きで付与されます。
-
-

参照： 13-72 ページの「[CREATE ROLE](#)」を参照してください。

例

次の文は、ロール `app_user` を IDENTIFIED GLOBALLY に変更します。

```
ALTER ROLE app_user IDENTIFIED GLOBALLY;
```

次の文は、ロール `teller` 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、ロール `teller` 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する必要があります。

次の例は、`analyst` ロールを `hr.admin` パッケージを使用するアプリケーション・ロールに変更します。

```
ALTER ROLE analyst IDENTIFIED USING hr.admin;
```

ALTER ROLLBACK SEGMENT

用途

ALTER ROLLBACK SEGMENT 文を使用すると、ロールバック・セグメントのオンライン / オフライン切替え、記憶特性の変更、またはロールバック・セグメントの最適サイズまたは指定サイズへの縮小を行います。

ここでは、データベースがロールバック UNDO モードで実行されている (UNDO_MANAGEMENT 初期化パラメータを MANUAL に設定、またはすべて設定しない) ことを前提としています。

データベースが自動 UNDO 管理モードで実行されている場合 (初期化パラメータ UNDO_MANAGEMENT を AUTO に設定)、ユーザー作成のロールバック・セグメントを変更することはできません。この場合、CREATE ROLLBACK SEGMENT または ALTER ROLLBACK SEGMENT 文の応答でエラーが戻されます。このエラーを回避するには、UNDO_SUPPRESS_ERRORS パラメータを TRUE に設定します。

参照：

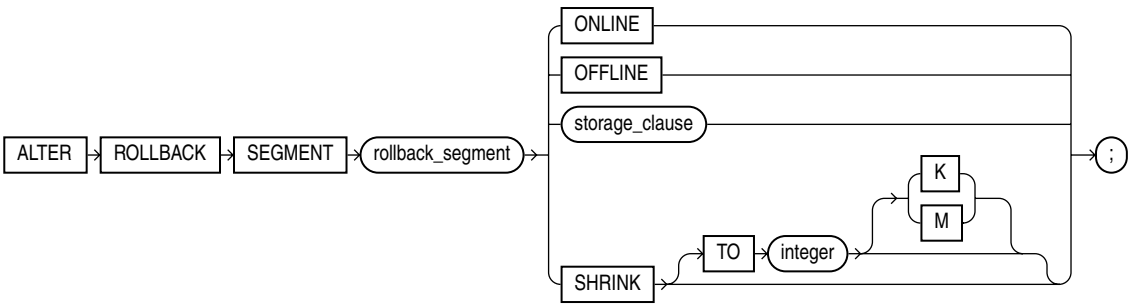
- ロールバック・セグメントの作成の詳細は、13-75 ページの「[CREATE ROLLBACK SEGMENT](#)」を参照してください。
- UNDO_MANAGEMENT パラメータおよび UNDO_SUPPRESS_ERRORS パラメータの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

前提条件

ALTER ROLLBACK SEGMENT システム権限が必要です。

構文

alter_rollback_segment::=



storage_clause: 17-50 ページの「[storage_clause](#)」を参照してください。

キーワードとパラメータ

rollback_segment

既存のロールバック・セグメントの名前を指定します。

ONLINE

ONLINE を指定すると、ロールバック・セグメントをオンラインにできます。ロールバック・セグメントを作成した場合、最初はオフライン状態になり、トランザクションに使用できなくなります。この句を指定した場合、ロールバック・セグメントはオンラインになり、インスタンスは、トランザクションに対してそのロールバック・セグメントを使用できるようになります。また、初期化パラメータ ROLLBACK_SEGMENTS を使用すると、インスタンスの起動時にロールバック・セグメントをオンラインにできます。

OFFLINE

OFFLINE を指定すると、ロールバック・セグメントをオフラインにできます。

- ロールバック・セグメント内に、アクティブ・トランザクションのロールバックに必要な情報が含まれていない場合は、すぐにオフラインになります。
- ロールバック・セグメントにアクティブ・トランザクションについての情報が含まれている場合、このロールバック・セグメントをその後のトランザクションに対して使用できないようにします。また、そのすべてのアクティブ・トランザクションがコミットまたはロールバックされた後、ロールバック・セグメントはオフラインになります。

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す。

ロールバック・セグメントがオンラインかオフラインかを調べるには、データ・ディクショナリ・ビュー `DBA_ROLLBACK_SEGS` に問い合わせます。オンライン・ロールバック・セグメントの `STATUS` 値は `IN_USE` です。オフライン・ロールバック・セグメントの `STATUS` 値は `AVAILABLE` です。

制限事項： `SYSTEM` ロールバック・セグメントをオフラインにすることはできません。

参照： ロールバック・セグメントを使用可能および使用禁止にする方法については、『Oracle9i データベース管理者ガイド』を参照してください。

storage_clause

storage_clause を使用すると、ロールバック・セグメントの記憶特性を変更できます。

制限事項： 既存のロールバック・セグメントに対する `INITIAL` および `MINEXTENTS` の値は変更できません。

参照： 構文および追加情報については、17-50 ページの「*storage_clause*」を参照してください。

SHRINK 句

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する場合に、`SHRINK` を指定します。縮小されるかどうか、および縮小量は、ロールバック・セグメントの使用可能領域およびアクティブ・トランザクションのロールバック・セグメント内での領域保持状態の状況によって異なります。

K または M で単位を KB または MB に指定しなかった場合、*integer* の値の単位はバイトになります。

TO *integer* に値を指定しなかった場合、ロールバック・セグメントを作成した `CREATE ROLLBACK SEGMENT` 文の *storage_clause* の `OPTIMAL` で指定した値が、デフォルトのサイズになります。`OPTIMAL` 値を指定しなかった場合、`CREATE ROLLBACK SEGMENT` 文の *storage_clause* の `MINEXTENTS` で指定した値がデフォルトのサイズになります。

TO *integer* に値を指定するかどうかにかかわらず、次のことがいえます。

- この文の実行時には、ロールバック・セグメントの縮小値が有効です。その後、サイズは `CREATE ROLLBACK SEGMENT` 文の `OPTIMAL` 値に戻ります。
- ロールバック・セグメントは、エクステント数 2 未満には縮小できません。

ロールバック・セグメントを縮小した後でロールバック・セグメントの実際のサイズを判断する場合は、`DBA_SEGMENTS` ビューの `BYTES` 列、`BLOCKS` 列および `EXTENTS` 列に問い合わせます。

制限事項： Real Application Clusters 環境では、インスタンスに対してオンライン状態のロールバック・セグメントのみを縮小できます。

例

ロールバック・セグメントをオンラインにする例 次の文は、ロールバック・セグメント rs_one をオンラインにします。

```
ALTER ROLLBACK SEGMENT rs_one ONLINE;
```

ロールバック・セグメントの記憶域の変更例 次の文は、rs_one の STORAGE パラメータを変更します。

```
ALTER ROLLBACK SEGMENT rs_one  
    STORAGE (NEXT 1000 MAXEXTENTS 20);
```

ロールバック・セグメントのサイズの変更例 次の文は、ロールバック・セグメントのサイズを 100MB に変更します。

```
ALTER ROLLBACK SEGMENT rs_one  
    SHRINK TO 100 M;
```

ALTER SEQUENCE

用途

ALTER SEQUENCE 文を使用すると、既存の順序の増分値、最小値および最大値、キャッシュされる数および動作を変更できます。この文は、順序番号に影響します。

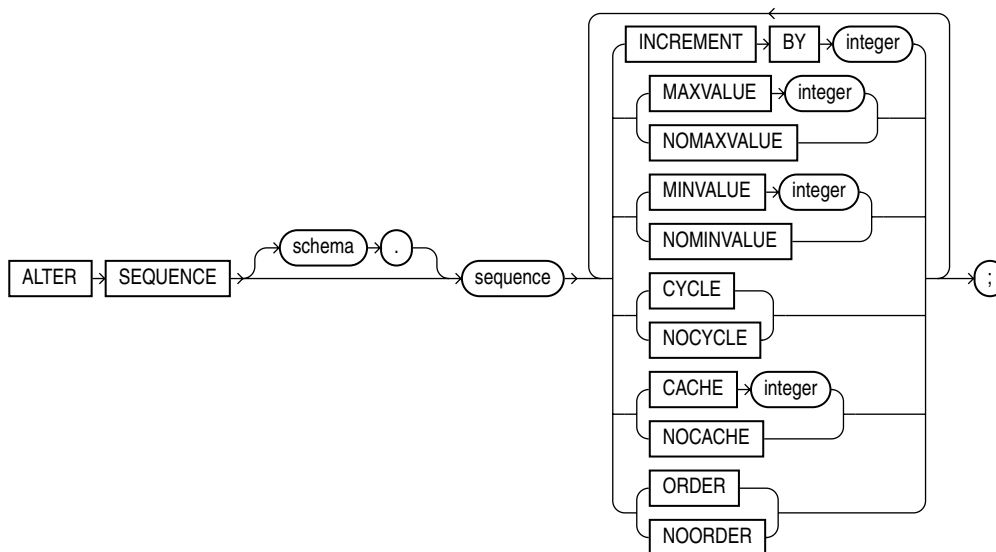
参照： 順序の詳細は、13-81 ページの「[CREATE SEQUENCE](#)」を参照してください。

前提条件

順序が自分のスキーマ内にある必要があります。自分のスキーマ内にはない場合は、順序に対する ALTER オブジェクト権限または ALTER ANY SEQUENCE システム権限が必要です。

構文

alter_sequence::=



キーワードとパラメータ

この文のキーワードおよびパラメータの意味は、順序を作成する場合と同じです。

- 異なる順序番号で再開する場合、順序を削除して再作成する必要があります。
- NEXTVAL を最初に呼び出す前に、INCREMENT BY の値を変更する場合、いくつかの順序番号がスキップされます。このため、元の START WITH の値を保持するには、順序を削除し、これを元の START WITH の値および新しい INCREMENT BY の値を使用して再作成する必要があります。
- いくつかの妥当性チェックが行われます。たとえば、MAXVALUE の値に現行の順序番号より小さい値は指定できません。

参照：

- 順序の作成の詳細は、13-81 ページの「[CREATE SEQUENCE](#)」を参照してください。
- 順序の削除および再作成の詳細は、16-2 ページの「[DROP SEQUENCE](#)」を参照してください。

例

順序の変更例 次の文は、employees_seq 順序に新しい最大値を設定します。

```
ALTER SEQUENCE employees_seq  
    MAXVALUE 1500;
```

次の文は、employees_seq 順序に CYCLE オプションと CACHE オプションを指定します。

```
ALTER SEQUENCE employees_seq  
    CYCLE  
    CACHE 5;
```

SQL 文 : ALTER SESSION ~ ALTER SYSTEM

この章では、次の SQL 文について説明します。

- ALTER SESSION
- ALTER SYSTEM

ALTER SESSION

用途

ALTER SESSION 文を使用すると、データベースへの接続に影響するすべての条件またはパラメータを、指定または変更できます。この文は、データベースとの接続を切断するまで有効です。

前提条件

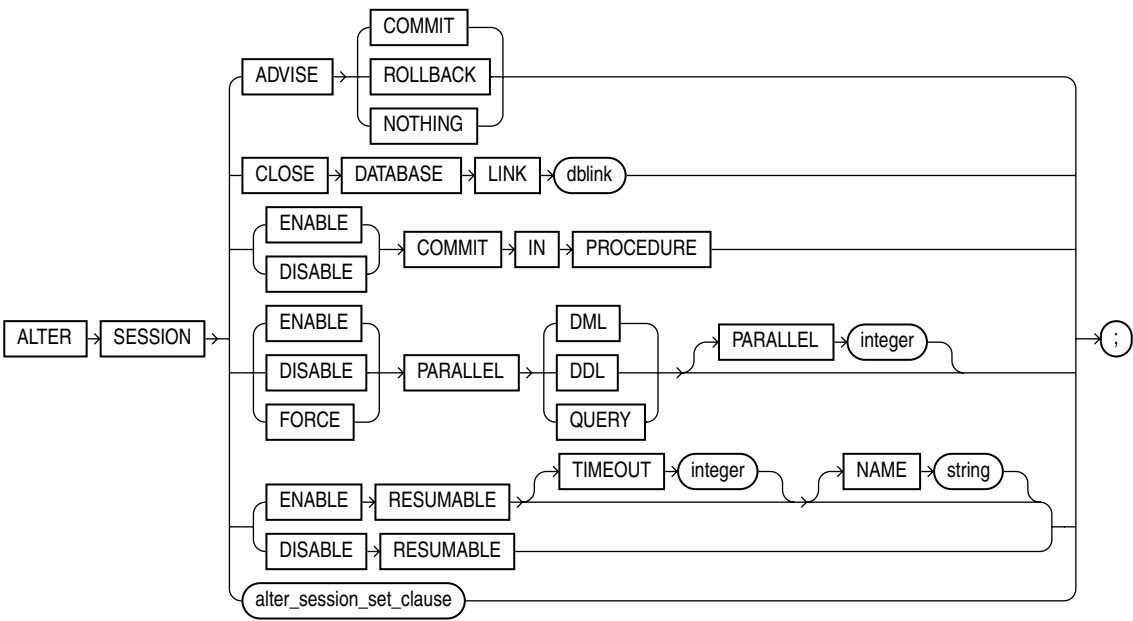
SQL トレース機能を使用可能または使用禁止にするには、ALTER SESSION システム権限が必要です。

再開可能な領域割当てを使用可能または使用禁止にするには、RESUMABLE システム権限が必要です。

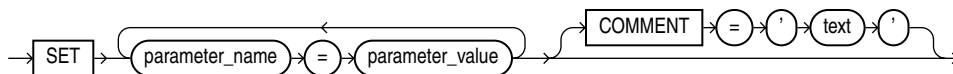
特に指定がないかぎり、これ以外の操作についての権限は不要です。

構文

alter_session::=



`alter_session_set_clause::=`



キーワードとパラメータ

ADVISE 句

ADVISE 句は、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。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（値 'C' が COMMIT、'R' が ROLLBACK、' ' が NOTHING を示します）。トランザクションの状態がインダウトになった場合、データベース管理者は、このアドバイスを使用してトランザクションをコミットするか、ロールバックするかを決定できます。

単一トランザクションにおいて、ADVISE 句を指定した ALTER SESSION 文を複数発行し、リモート・データベースごとに異なるアドバイスを送ることができます。ADVISE 句を指定した文はそれぞれ、ADVISE 句を指定した別の文が発行されるまで、トランザクション内の後続する文で参照されるデータベースに対してアドバイスを送ります。

CLOSE DATABASE LINK 句

CLOSE DATABASE LINK を指定すると、データベース・リンク *dblink* がクローズされます。データベース・リンクを使用する SQL 文を発行した場合、Oracle は、このデータベース・リンクを使用してリモート・データベース上にセッションを作成します。この接続は、セッションの終了またはデータベース・リンクの数が初期化パラメータ OPEN_LINKS の値を超えるまでオープンされています。リンクをオープンしたままにしておくことによって発生するネットワークのオーバーヘッドを減らすには、セッションでデータベース・リンクを再度使用しない場合に、この句を使用してデータベース・リンクを明示的にクローズします。

ENABLE | DISABLE COMMIT IN PROCEDURE

プロシージャおよびストアド・ファンクションは PL/SQL で記述されるため、COMMIT 文と ROLLBACK 文を発行できます。アプリケーション自体が直接発行していない COMMIT 文や ROLLBACK 文によって、アプリケーションが中断される場合、DISABLE COMMIT IN PROCEDURE を指定して、セッション中にコールされるプロシージャおよびストアド・ファンクションがこれらの文を発行しないように制御します。

その後、ENABLE DISABLE COMMIT IN PROCEDURE を指定することによって、セッションでプロシージャおよびストアド・ファンクションが COMMIT および ROLLBACK 文を発行できるようになります。

一部のアプリケーション（SQL*Forms など）は、自動的にプロシージャおよびストアド・ファンクションでの COMMIT 文や ROLLBACK 文を禁止します。詳細は、ご使用のアプリケーションのドキュメントを参照してください。

PARALLEL DML | DDL | QUERY

PARALLEL パラメータは、そのセッションの後続の DML、DDL または問合せ文をパラレル実行とみなすかどうかを指定します。この句は、現行のセッション中に表自体を変更せずに、表の並列度をオーバーライド可能にします。コミットされていないトランザクションは、DML に対してこの句を実行する前に、コミットまたはロールバックされる必要があります。

ENABLE 句

ENABLE を指定すると、セッション内の後続文をパラレルで実行します。これは、DDL 文および問合せ文のデフォルトです。

DML パラレル・ヒントまたはパラレル句が指定されている場合に、セッションの DML 文をパラレル・モードで実行します。

DDL パラレル句が指定されている場合に、セッションの DDL 文をパラレル・モードで実行します。

QUERY パラレル・ヒントまたはパラレル句が指定されている場合に、セッションの問合せをパラレル・モードで実行します。

制限事項：オプションの *PARALLEL integer* に ENABLE を指定することはできません。

DISABLE 句

DISABLE を指定すると、後続文をシリアルに実行します。これは、DML 文のデフォルトです。

DML セッションの DML 文をシリアルに実行します。

DDL セッションの DDL 文をシリアルに実行します。

QUERY セッションの問合せをシリアルに実行します。

制限事項：オプションの *PARALLEL integer* に DISABLE を指定することはできません。

FORCE 句

FORCE は、セッションの後続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、デフォルトの並列度が使用されます。この句は、セッションの後続文に指定されたすべての `parallel_clause` をオーバーライドしますが、パラレル・ヒントによりオーバーライドされます。

DML パラレル DML 制限のどれにも違反していない場合、特定の並列度がこの句に指定されていないかぎり、セッションの後続の DML 文は、デフォルトの並列度で実行されます。

DDL 特定の並列度がこの句に指定されていないかぎり、セッションの後続の DDL 文は、デフォルトの並列度で実行されます。結果のデータベース・オブジェクトは、通常の並列度に対応します。

FORCE DDL を使用した場合、そのセッションで作成されるすべての表は、自動的にデフォルトのパラレル化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を使用して）`parallel_clause` を指定した場合と同じです。

QUERY 特定の並列度がこの句に指定されていないかぎり、後続の間合せはデフォルトの並列度で実行されます。

PARALLEL integer 並列度を明示的に指定する整数を指定します。

- FORCE DDL では、並列度は後続の DDL 文のパラレル句をオーバーライドします。
- FORCE DML および FORCE QUERY では、並列度は、データ・ディクショナリの表に格納されている現行の並列度をオーバーライドします。
- ヒントによって文に指定される並列度は、強制的に実行される並列度をオーバーライドします。

次の DML 操作は、この句に関係なくパラレル化されません。

- クラスタ化表に対する操作
- データベースまたはパッケージの状態を読み書きする埋込みファンクションを使用した操作
- 起動する可能性のあるトリガーを使用した表に対する操作
- オブジェクト型、LONG または LOB データ型が含まれている表またはスキーマ・オブジェクトでの操作

RESUMABLE 句

これらの句を使用すると、再開可能な領域割当てを使用可能および使用禁止にできます。この機能によって、領域不足のエラー条件が発生した場合に操作は停止され、エラー条件が修復されたときに中断したところから自動的に再開されます。

注意： 再開可能な領域割当ては、ローカル管理の表領域での操作で、完全にサポートされます。ディクショナリ管理の表領域を使用する場合は、制限事項があります。制限事項の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。

ENABLE RESUMABLE

この句を使用すると、セッションに対する再開可能な領域割当てが使用可能になります。

TIMEOUT TIMEOUT では、エラー条件が修復されるまで操作を停止する時間を秒単位で指定できます。エラー条件が TIMEOUT で指定した時間までに修復されない場合は、停止操作は異常終了します。

NAME NAME では、ユーザー定義のテキスト文字列を指定することができ、再開可能モードのセッション中に発行される文の識別に有効です。USER_RESUMABLE データ・ディクショナリ・ビューおよび DBA_RESUMABLE データ・ディクショナリ・ビューに、テキスト文字列が挿入されます。NAME を指定しない場合は、デフォルト文字列「User username(userid), Session sessionid, Instance instanceid」が挿入されます。

参照： データ・ディクショナリ・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

DISABLE RESUMABLE

この句を使用すると、セッションに対する再開可能な領域割当てが使用禁止になります。

alter_session_set_clause

alter_session_set_clause を使用すると、ALTER SESSION 文の有効範囲内で動的なセッション・パラメータおよび初期化パラメータを設定できます。同じ *alter_session_set_clause* で複数のパラメータに対する値を設定できます。

COMMENT を使用すると、コメント文字列をパラメータ値の変更に対応付けることができます。

初期化パラメータおよび ALTER SESSION

ALTER SYSTEM 文で設定可能なすべての初期化パラメータについては、9-19 ページの「ALTER SYSTEM」を参照してください。表 9-1 に ALTER SESSION の有効範囲内で動的な初期化パラメータおよび「ALTER SYSTEM」での参照先を示します。ALTER SESSION でパラメータを設定した場合は、設定した値が現行のセッション以外では継続されないことのみが ALTER SYSTEM と異なります。

ALTER SESSION で設定可能な一部のパラメータは、初期化パラメータではありません。つまり、初期化パラメータ・ファイルではなく、ALTER SESSION でのみ設定可能です。これらのセッション・パラメータについては、表 9-1 の後に説明します。

注意： 特に指定がないかぎり、ここで説明するパラメータは初期化パラメータを示し、パラメータの概要のみを説明します。初期化パラメータの値を変更する前に、『Oracle9i データベース・リファレンス』または『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

表 9-1 ALTER SESSION で設定可能な初期化パラメータ

パラメータ	コメント
CURSOR_SHARING (9-42 ページ)	様々な環境での設定の詳細は、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。
DB_BLOCK_CHECKING (9-44 ページ)	ALTER SESSION SET DB_BLOCK_CHECKING による設定は、後続の ALTER SYSTEM SET DB_BLOCK_CHECKING 文でオーバーライドされます。
DB_CREATE_FILE_DEST (9-46 ページ)	
DB_CREATE_ONLINE_LOG_DEST_n (9-46 ページ)	
DB_FILE_MULTIBLOCK_READ_COUNT (9-47 ページ)	
GLOBAL_NAMES (9-56 ページ)	グローバル・ネーム解決およびその実行方法の詳細は、2-113 ページの「リモート・データベース内のオブジェクトの参照」を参照してください。
HASH_AREA_SIZE (9-56 ページ)	
HASH_JOIN_ENABLED (9-57 ページ)	

表 9-1 ALTER SESSION で設定可能な初期化パラメータ（続き）

パラメータ	コメント
LOG_ARCHIVE_DEST_n (9-65 ページ)	
LOG_ARCHIVE_DEST_STATE_n (9-66 ページ)	
LOG_ARCHIVE_MIN_SUCCEED_DEST (9-68 ページ)	
MAX_DUMP_FILE_SIZE (9-73 ページ)	
NLS パラメータ	
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グローバリゼーション・サポートが提供されます。動的パフォーマンス表 V\$NLS_PARAMETERS に問い合わせると、セッションの現在の NLS 属性を参照できます。NLS パラメータの詳細は、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。	
NLS_CALENDAR (9-74 ページ)	
NLS_COMP (9-74 ページ)	
NLS_CURRENCY (9-75 ページ)	
NLS_DATE_FORMAT (9-75 ページ)	有効な日付書式モデルの詳細は、2-66 ページの「日付書式モデル」を参照してください。
NLS_DATE_LANGUAGE (9-75 ページ)	
NLS_DUAL_CURRENCY (9-76 ページ)	数値書式要素の詳細は、2-61 ページの「数値書式モデル」を参照してください。
NLS_ISO_CURRENCY (9-76 ページ)	
NLS_LANGUAGE (9-76 ページ)	
NLS_LENGTH_SEMANTICS (9-77 ページ)	
NLS_NCHAR_CONV_EXCP (9-77 ページ)	
NLS_NUMERIC_CHARACTERS (9-77 ページ)	
NLS_SORT (9-78 ページ)	
NLS_TERRITORY (9-78 ページ)	
NLS_TIMESTAMP_FORMAT (9-78 ページ)	
NLS_TIMESTAMP_TZ_FORMAT (9-79 ページ)	
OBJECT_CACHE_MAX_SIZE_PERCENT (9-79 ページ)	
OBJECT_CACHE_OPTIMAL_SIZE (9-80 ページ)	

表 9-1 ALTER SESSION で設定可能な初期化パラメータ（続き）

パラメータ	コメント
OPTIMIZER_INDEX_CACHING (9-82 ページ)	
OPTIMIZER_INDEX_COST_ADJ (9-82 ページ)	
OPTIMIZER_MAX_PERMUTATIONS (9-82 ページ)	
OPTIMIZER_MODE (9-83 ページ)	アプリケーションの特性に基づく、コストベース方法に対する目標の決定方法は、『Oracle9i データベース概要』および『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。
ORACLE_TRACE_ENABLE (9-84 ページ)	
PARALLEL_BROADCAST_ENABLED (9-87 ページ)	
PARALLEL_INSTANCE_GROUP (9-88 ページ)	
PARALLEL_MIN_PERCENT (9-89 ページ)	
PARTITION_VIEW_ENABLED (9-90 ページ)	パーティション・ビューの詳細は、15-38 ページの「パーティション・ビュー」を参照してください。
PLSQL_COMPILER_FLAGS (9-91 ページ)	
QUERY_REWRITE_ENABLED (9-94 ページ)	
QUERY_REWRITE_INTEGRITY (9-95 ページ)	
REMOTE_DEPENDENCIES_MODE (9-96 ページ)	
SESSION_CACHED_CURSORS (9-101 ページ)	
SORT_AREA_RETAINED_SIZE (9-105 ページ)	
SORT_AREA_SIZE (9-106 ページ)	
STAR_TRANSFORMATION_ENABLED (9-108 ページ)	
TIMED_OS_STATISTICS (9-109 ページ)	
TIMED_STATISTICS (9-110 ページ)	
TRACE_ENABLED (9-110 ページ)	
UNDO_SUPPRESS_ERRORS (9-112 ページ)	
WORKAREA_SIZE_POLICY (9-114 ページ)	

セッション・パラメータおよび ALTER SESSION

次のパラメータは、セッション・パラメータであり、初期化パラメータではありません。

CONSTRAINT[S]

構文：

```
CONSTRAINT[S] = { IMMEDIATE | DEFERRED | DEFAULT }
```

CONSTRAINT[S] は、遅延可能制約によって指定された条件を、いつ適用するかを指定します。

- `immediate` を設定した場合、遅延可能な制約によって指定される条件は、各 DML 文の直後にチェックされます。これは、セッションの各トランザクションの開始時に、SET CONSTRAINTS ALL IMMEDIATE 文を発行することと同じです。
- `deferred` を設定した場合、遅延可能な制約によって指定される条件は、トランザクションのコミット時にチェックされます。これは、セッションの各トランザクションの開始時に、SET CONSTRAINTS ALL DEFERRED 文を発行することと同じです。
- `default` を設定した場合、すべての制約は各トランザクションの開始時に、DEFERRED または IMMEDIATE の初期状態にリストアされます。

CREATE_STORED_OUTLINES

構文：

```
CREATE_STORED_OUTLINES = {TRUE | FALSE | 'category_name'}
```

CREATE_STORED_OUTLINES パラメータは、セッション中に発行された問合せごとに、アウトラインを自動的に作成および格納するかどうかを決定します。

- `TRUE` を設定した場合、同一セッション内で後続の問合せに対するアウトラインは、自動的に作成可能になります。このアウトラインは、一意のシステム生成名を受け取り、DEFAULT カテゴリに格納されます。すでに特定の問合せに定義したアウトラインが DEFAULT カテゴリに存在する場合は、そのアウトラインが保持され、新しいアウトラインは作成されません。
- `FALSE` を設定した場合、セッション中のアウトラインは作成禁止になります。これはデフォルト値です。
- `category_name` を設定した場合、TRUE と同じ動作をしますが、セッション中に作成されたすべてのアウトラインは、`category_name` カテゴリに格納されます。

CURRENT_SCHEMA

構文：

```
CURRENT_SCHEMA = schema
```

CURRENT_SCHEMA は、セッションの現行のスキーマを、指定したスキーマに変更します。セッション中のスキーマ・オブジェクトに対する後続の未修飾の参照は、この指定したスキーマ内でオブジェクトに変換されます。この設定は、現行のセッションの存続期間中、または ALTER SESSION SET CURRENT_SCHEMA 文を再発行するまで保持されます。

この設定によって、現行のユーザーのスキーマ以外にあるオブジェクトに対する操作を、オブジェクトをスキーマ名で修飾することなく簡単に行えます。この設定によって、現行のスキーマは変更されますが、このセッションのユーザーまたは現行のユーザーは変更されません。また、このセッションに対して、追加のシステム権限またはオブジェクト権限は付与されません。

ERROR_ON_OVERLAP_TIME

構文：

```
ERROR_ON_OVERLAP_TIME = {TRUE | FALSE}
```

ERROR_ON_OVERLAP_TIME には、Oracle が不明瞭な境界日時値（日時が標準か夏時間かが明確でない場合）を処理する方法を指定します。

- TRUE を指定すると、不明瞭なオーバーラップ・タイムスタンプに対してエラーが戻されます。
- FALSE を指定すると、不明瞭なオーバーラップ・タイムスタンプは標準時刻のデフォルトになります。これはデフォルト値です。

FLAGGER

構文：

```
FLAGGER = { ENTRY | INTERMEDIATE | FULL | OFF }
```

FLAGGER パラメータは、FIPS のフラグ付けを指定します。このフラグ付けを使用した場合、ANSI SQL92 の拡張要素である SQL 文が発行されたときに、エラー・メッセージが生成されます。FLAGGER は、セッション・パラメータであり、初期化パラメータではありません。

Oracle では、ENTRY レベル、INTERMEDIATE レベル、FULL レベルのそれぞれのフラグ付けに違いはありません。セッションでフラグ付けが設定された場合、これに続く ALTER SESSION SET FLAGGER 文は成功しますが、ORA-00097 のメッセージが生成されます。このため、セッションを切断しなくても FIPS のフラグ付けを変更できます。OFF を設定した場合、フラグ付けの使用は停止されます。

INSTANCE

構文：

```
INSTANCE = integer
```

Real Application Clusters 環境で INSTANCE パラメータを設定すると、セッションが integer で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。INSTANCE はセッション・パラメータであり、初期化パラメータではありません。パフォーマンスの最適化のため、Real Application Clusters 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・グループなどを使用しています。Real Application Clusters 環境では、通常、特定のインスタンスに接続し、使用するパーティション化データにアクセスします。別のインスタンスに接続する必要がある場合は、データのパーティション分割はなくなります。このパラメータを設定した場合、インスタンスに接続している場合と同様に、別のインスタンスにもアクセスできます。

ISOLATION_LEVEL

構文：

```
ISOLATION_LEVEL = {SERIALIZABLE | READ COMMITTED}
```

ISOLATION_LEVEL パラメータは、データベースを変更するトランザクションがどのように処理されるかを指定します。ISOLATION_LEVEL はセッション・パラメータであり、初期化パラメータではありません。

- SERIALIZABLE を設定した場合、セッション内のトランザクションは、SQL92 に規定されているとおりシリアライズ可能トランザクション分離モードを使用します。シリアライズ可能トランザクションが行を更新する DML 文を実行する場合、現在の更新対象の行がそのシリアライズ可能トランザクションの開始時にコミットされていない別のトランザクションによって更新されていたときは、その DML 文は失敗します。シリアライズ可能トランザクションは、同一トランザクション内で行った更新を確認できます。
- READ COMMITTED を設定した場合、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でかけられた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。

PLSQL_DEBUG

構文：

```
PLSQL_DEBUG = { TRUE | FALSE }
```

PLSQL_DEBUG パラメータには、コンパイル操作中にデバッグ情報を含むか含まないかを設定します。このパラメータに TRUE を設定すると、DEBUG キーワードを ALTER {FUNCTION | PROCEDURE | PACKAGE} COMPILE 文に追加したときと同様に動作します。

SKIP_UNUSABLE_INDEXES**構文:**

```
SKIP_UNUSABLE_INDEXES = { TRUE | FALSE }
```

SKIP_UNUSABLE_INDEXES パラメータは、使用禁止の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。SKIP_UNUSABLE_INDEXES は、セッション・パラメータで、初期化パラメータではありません。

- TRUE に設定した場合、UNUSABLE のマークが付けられた索引および索引パーティションのエラー・レポートを使用禁止にします。使用禁止の索引または索引パーティションを持つ表に対するすべての操作（挿入、削除、更新および選択）が認められます。
- FALSE に設定した場合、UNUSABLE 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。使用禁止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、削除および更新は認められません。これはデフォルト値です。

SQL_TRACE**構文:**

```
INSTANCE = integer
```

SQL_TRACE は、初期化パラメータです。ただし、ALTER SESSION 文で値を変更しても、V\$PARAMETER ビューに反映されません。そのため、このコンテキストでは、セッション・パラメータのみを考慮します。

出力の書式設定および解析方法を含む SQL トレース機能の詳細は、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。

TIME_ZONE**構文:**

```
TIME_ZONE = '[+ | -] hh:mm'
             | LOCAL
             | DBTIMEZONE
             | 'time_zone_region'
```

TIME_ZONE パラメータには、現行の SQL セッションのデフォルトのローカル・タイム・ゾーンを指定します。TIME_ZONE はセッション・パラメータであり、初期化パラメータではありません。

- UTC の前または後の時および分を示す書式マスク ('[+|-] hh:mm') を指定します。hh:mm の有効範囲は、-12:00 ~ +14:00 です。
- LOCAL を指定すると、現行の SQL セッションのデフォルトのローカル・タイム・ゾーン置換値が、現行の SQL セッションを起動したときに構築された、元のデフォルトのローカル・タイム・ゾーン置換値に設定されます。

- DBTIMEZONE を指定すると、現行のセッションのタイム・ゾーンにデータベースのタイム・ゾーンと一致する値が設定されます。この設定を指定する場合、DBTIMEZONE ファンクションは、データベースのタイム・ゾーンを UTC オフセットまたはタイム・ゾーン地域として戻します。これはデータベースのタイム・ゾーンの設定に依存します。
- 有効な `time_zone_region` を指定します。有効な地域名を表示するには、`V$TIMEZONE_NAMES` 動的パフォーマンス・ビューの `TZNAME` 列を問い合わせます。この設定を指定する場合、`SESSIONTIMEZONE` ファンクションは地域の名前を戻します。

USE_PRIVATE_OUTLINES

構文：

```
USE_PRIVATE_OUTLINES = { TRUE | FALSE | category_name }
```

`USE_PRIVATE_OUTLINES` パラメータを使用すると、プライベート・アウトラインの使用を制御することができます。このパラメータが使用可能で、アウトライン化された SQL 文が発行された場合、オブティマイザは、`USE_STORED_OUTLINES` が使用可能なときにはパブリック領域ではなくそのセッションのプライベート領域からアウトラインを検索します。そのセッションのプライベート領域にアウトラインが存在しない場合、オブティマイザは、文のコンパイルにアウトラインを使用しません。`USE_PRIVATE_OUTLINES` は、初期化パラメータではありません。

- `TRUE` に設定した場合、要求をコンパイルするときに、オブティマイザは `DEFAULT` カテゴリにストアド・プライベート・アウトラインを使用します。
- `FALSE` に設定した場合、オブティマイザはストアド・プライベート・アウトラインを使用しません。これはデフォルト値です。`USE_STORED_OUTLINES` が使用可能な場合、オブティマイザはストアド・パブリック・アウトラインを使用します。
- `category_name` に設定した場合、要求をコンパイルするときに、オブティマイザは `category_name` カテゴリにストアド・アウトラインを使用します。

制限事項：`USE_STORED_OUTLINES` が使用可能な場合は、このパラメータを使用可能にできません。

USE_STORED_OUTLINES

構文：

```
USE_STORED_OUTLINES = { TRUE | FALSE | category_name }
```

`USE_STORED_OUTLINES` パラメータには、オブティマイザが、ストアド・パブリック・アウトラインを使用して実行計画を生成するかどうかを指定します。`USE_STORED_OUTLINES` は、初期化パラメータではありません。

- `TRUE` に設定した場合、要求をコンパイルするときに、オブティマイザは `DEFAULT` カテゴリにストアド・アウトラインを使用します。

- FALSE に設定した場合、オブティマイザはストアド・アウトラインを使用しません。これはデフォルト値です。
- `category_name` を設定した場合、要求をコンパイルするときに、オブティマイザは `category_name` カテゴリにストアド・アウトラインを使用します。

制限事項：USE_PRIVATE_OUTLINES が使用可能な場合は、このパラメータを使用可能にできません。

例

パラレル DML を使用可能にする例 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場合、次の文を発行します。

```
ALTER SESSION ENABLE PARALLEL DML;
```

分散トランザクションを強制的に実行する例 次のトランザクションによって、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 `employees` 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 `employees` 表から従業員のレコードを削除します。

```
ALTER SESSION
  ADVISE COMMIT;
```

```
INSERT INTO employees@site1
VALUES (8002, 'Juan', 'Fernandez', 'juanf@hr.com', NULL,
        TO_DATE('04-OCT-1992', 'DD-MON-YYYY'), 'SA_CLERK', 3000,
        NULL, 121, 20);
```

```
ALTER SESSION
  ADVISE ROLLBACK;
```

```
DELETE FROM employees@site2
  WHERE employee_id = 8002;
```

```
COMMIT;
```

このトランザクションには、ADVISE 句を指定した ALTER SESSION 文が 2 つあります。このトランザクションが状態不明（インダウト）になった場合、`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。

データベース・リンクをクローズする例 次の文は、データベース・リンクを使用している hq データベース上の employees 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 employees@hq
  SET salary = salary + 200
  WHERE employee_id = 162;
```

```
COMMIT;
```

```
ALTER SESSION
  CLOSE DATABASE LINK hq;
```

日付書式の動的な変更例 次の文は、セッションのデフォルトの日付書式を 'YYYY MM DD-HH24:MI:SS' に動的に変更します。

```
ALTER SESSION
  SET NLS_DATE_FORMAT = 'YYYY MM DD HH24:MI:SS';
```

変更後は、新しい日付書式が次のように適用されます。

```
SELECT TO_CHAR(SYSDATE) Today
  FROM DUAL;
```

```
TODAY
-----
2001 04 12 12:30:38
```

日付言語の動的な変更例 次の文は、日付書式要素の言語をフランス語に変更します。

```
ALTER SESSION
  SET NLS_DATE_LANGUAGE = French;
```

```
SELECT TO_CHAR(SYSDATE, 'Day DD Month YYYY') Today
  FROM DUAL;
```

```
TODAY
-----
Jeudi    12 Avril    2001
```


ISO 通貨の変更例 次の文では、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(salary), 'C999G999D99') Total
  FROM employees;

TOTAL
-----
      USD694,900.00
```

小数点文字と桁区切りの変更例 次の文は、小数点文字をカンマ (,) に、桁区切りをピリオド (.) に動的に変更します。

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

これらの数値書式要素を使用した場合、新しい文字が戻ります。

ALTER SESSION SET NLS_CURRENCY = 'FF';

SELECT TO_CHAR( SUM(salary), 'L999G999D99') Total FROM employees;

TOTAL
-----
      FF694.900,00
```

NLS 通貨の変更例 次の文は、各国通貨記号を 'DM' に動的に変更します。

```
ALTER SESSION
  SET NLS_CURRENCY = 'DM';

SELECT TO_CHAR( SUM(salary), 'L999G999D99') Total
  FROM employees;

TOTAL
-----
      DM694.900,00
```

NLS 言語の変更例 次の文は、表示されたエラー・メッセージの言語をフランス語に動的に変更します。

```
ALTER SESSION
  SET NLS_LANGUAGE = FRENCH;
```

Session modifiee.

```
SELECT * FROM DMP;
```

ORA-00942: Table ou vue inexistante

言語のソート順の変更例 次の文は、言語ソート順序をスペイン語に動的に変更します。

```
ALTER SESSION
  SET NLS_SORT = XSpanish;
```

これによって、文字の値はスペイン語のソート順序に基づいてソートされます。

SQL トレースを使用可能にする例 次の文は、セッションに対して SQL トレース機能を使用可能にします。

```
ALTER SESSION
  SET SQL_TRACE = TRUE;
```

クエリー・リライトを使用可能にする例 次の文は、明示的に使用禁止にされていないすべてのマテリアライズド・ビューに対する現行のセッションのクエリー・リライトを使用可能にします。

```
ALTER SESSION SET QUERY_REWRITE_ENABLED = TRUE;
```

ALTER SYSTEM

用途

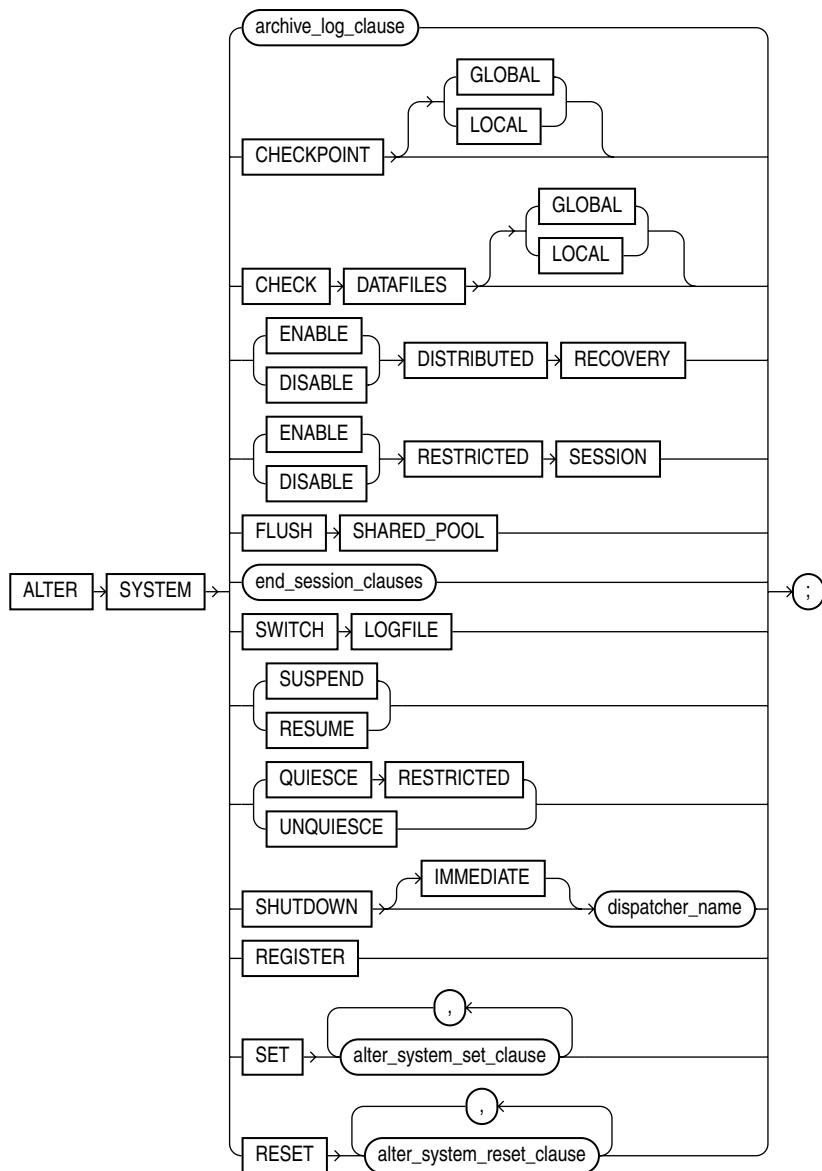
ALTER SYSTEM 文を使用すると、Oracle インスタンスを動的に変更できます。この設定は、データベースがマウントされているかぎり有効です。

前提条件

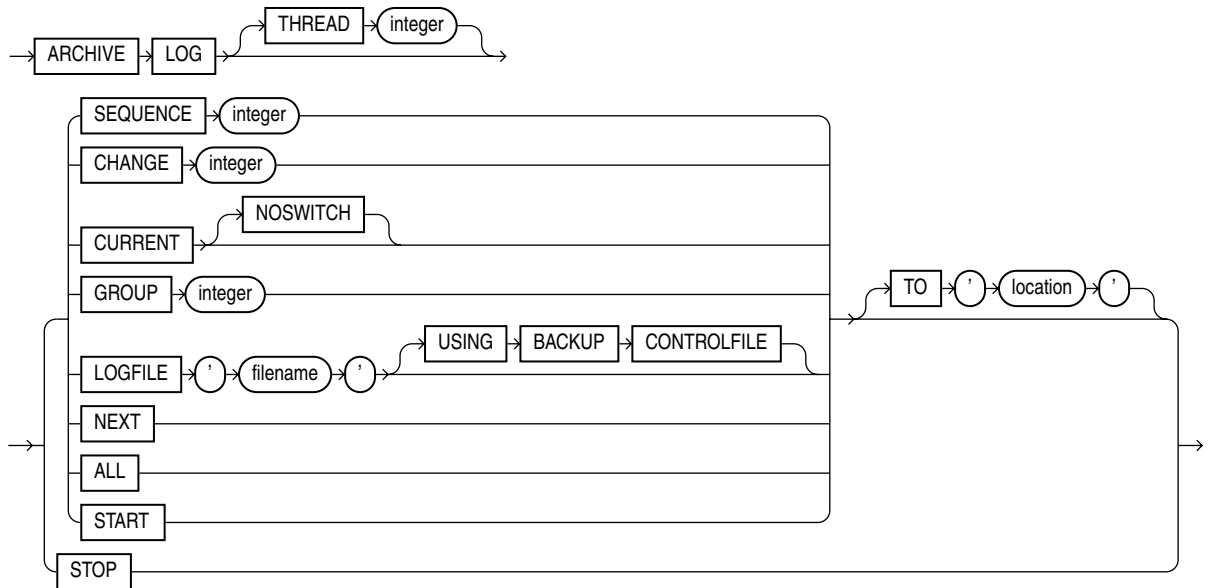
ALTER SYSTEM システム権限が必要です。

archive_log_clause を指定する場合は、SYSDBA または SYSOPER システム権限が必要です。

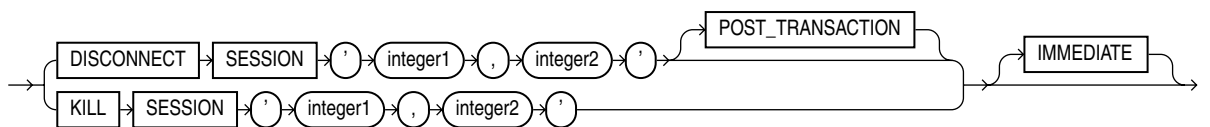
構文

alter_system::=

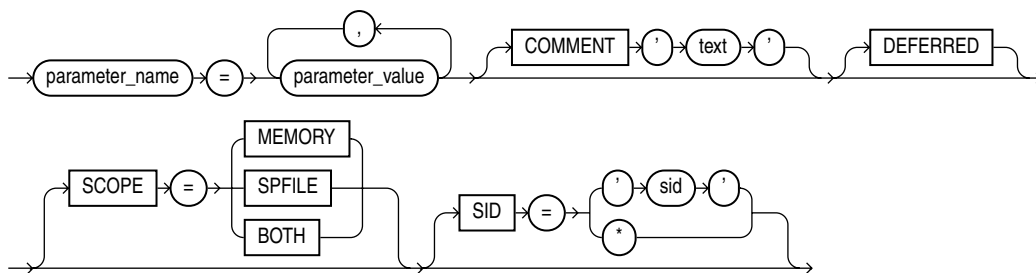
archive_log_clause::=



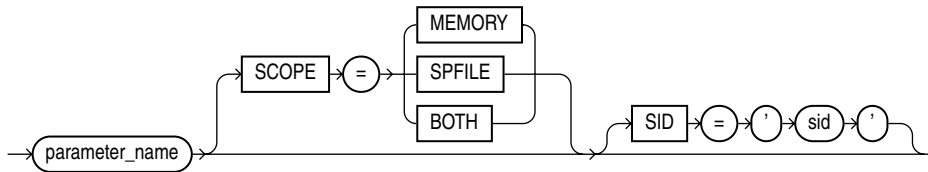
end_session_clauses::=



alter_system_set_clause::=



alter_system_reset_clause::=



キーワードとパラメータ

archive_log_clause

archive_log_clause は、REDO ログ・ファイル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。この句を使用する場合、インスタンスでデータベースをマウントする必要があります。特に指定がない限り、データベースはオープンまたはクローズできます。

THREAD 句

THREAD を指定すると、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を指定できます。

制限事項： Real Application Clusters 環境で Oracle を使用する場合のみ、このパラメータを設定します。

SEQUENCE 句

SEQUENCE を指定すると、指定したスレッド内のログ順序番号 *integer* によって識別されるオンライン REDO ログ・ファイル・グループを手動でアーカイブできます。THREAD パラメータを指定しなかった場合、インスタンスに割り当てられているスレッドから、指定したグループがアーカイブされます。

CHANGE 句

CHANGE を指定すると、オンライン REDO ログ・ファイル・グループを、手動でアーカイブできます。このグループには、指定したスレッド内の *integer* によって識別される SCN を持つ REDO ログ・エントリが含まれます。この SCN が現行の REDO ログ・ファイル・グループ内にある場合、ログ・スイッチが実行されます。THREAD パラメータを指定しない場合、使用可能な状態にあるすべてのスレッドから、この SCN を含むグループがアーカイブされます。

インスタンスでデータベースをオープンしている場合にのみ、この句を使用できます。

CURRENT 句

CURRENT を指定すると、ログ・スイッチを強制的に発生させ、指定したスレッドの現行の REDO ログ・ファイル・グループを手動でアーカイブできます。THREAD パラメータを指定しない場合、すべての使用可能なスレッドから、現行のログ以前のログも含むすべての REDO ログ・ファイル・グループがアーカイブされます。データベースがオープンしているときのみ、CURRENT を指定できます。

ログ・スイッチの強制実行なしで現行の REDO ログ・ファイル・グループを手動でアーカイブする場合は、NOSWITCH を指定します。インスタンスでデータベースをオープンしている場合にのみ、この句を使用できます。データベースがオープンしている場合は、この操作によってデータベースは自動的にクローズされます。再オープンする前にデータベースを手動で停止する必要があります。

GROUP 句

GROUP を指定すると、オンライン REDO ログ・ファイル・グループを手動でアーカイブできます。このグループには、integer によって識別される GROUP 値が含まれます。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 に問い合わせ確認できます。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・グループが、指定するスレッド内に含まれている必要があります。

LOGFILE 句

LOGFILE を指定すると、オンライン REDO ログ・ファイル・グループを手動でアーカイブできます。このグループには、'filename' によって識別される REDO ログ・ファイル・メンバーが含まれます。THREAD パラメータと LOGFILE パラメータの両方を指定する場合は、指定する REDO ログ・ファイル・グループが、指定するスレッド内に含まれている必要があります。

データベースがバックアップ制御ファイルでマウントされている場合は、USING BACKUP CONTROLFILE を指定し、現行のログ・ファイルを含むすべてのオンライン・ログ・ファイルのアーカイブを許可します。

制限事項：REDO ログ・ファイル・グループは、一杯になった順にアーカイブする必要があります。LOGFILE パラメータを使用して REDO ログ・ファイル・グループのアーカイブを指定した場合、それ以前の REDO ログ・ファイル・グループがアーカイブされていないとエラー・メッセージが戻ります。

NEXT 句

NEXT を指定すると、一杯になってもアーカイブされていない次のオンライン REDO ログ・ファイルを、指定したスレッドから手動でアーカイブできます。THREAD パラメータを指定しない場合、使用可能な任意のスレッド上の、アーカイブされていない最初の REDO ログ・ファイル・グループがアーカイブされます。

ALL 句

ALL を指定すると、一杯になってもアーカイブされていないすべてのオンライン REDO ログ・ファイルを、指定したスレッドから手動でアーカイブできます。THREAD パラメータを指定しない場合、使用可能なすべてのスレッドから、一杯でアーカイブされていないすべての REDO ログ・ファイル・グループがアーカイブされます。

START 句

START を指定すると、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できます。

制限事項：インスタンスに割り当てられているスレッドについてのみ、自動アーカイブを使用可能にできます。

TO location 句

TO 'location' を指定すると、REDO ログ・ファイル・グループがアーカイブされる位置を指定できます。このパラメータの値は、オペレーティング・システムの規則に従って、ファイルの位置を完全に指定する必要があります。このパラメータを指定しない場合、REDO ログ・ファイル・グループは初期化パラメータ LOG_ARCHIVE_DEST または LOG_ARCHIVE_DEST_n に指定された場所に格納されます。

STOP 句

STOP を指定すると、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できます。インスタンスに割り当てられているスレッドについてのみ、自動アーカイブを使用禁止にできます。

CHECKPOINT 句

CHECKPOINT を指定すると、チェックポイントを明示的に強制処理させ、コミット済のトランザクションによる変更をディスク上のデータ・ファイルに書き込みます。インスタンスでデータベースがオープンしている場合にのみ、この句を指定できます。チェックポイントが完了するまで、ユーザーに制御は戻りません。

GLOBAL Real Application Clusters 環境で、データベースをオープンしているすべてのインスタンスに対してチェックポイントを実行します。これはデフォルトです。

LOCAL Real Application Clusters 環境で、文を発行するインスタンスの REDO ログ・ファイル・グループのスレッドに対してのみチェックポイントを実行します。

CHECK DATAFILES 句

Real Application Clusters 環境などの分散データベース・システムで、データベース・コントロール・ファイルからインスタンスの SGA を更新し、すべてのオンライン・データ・ファイルに情報を反映します。

- GLOBAL を指定すると、データベースをオープンしているすべてのインスタンスに対して、この同期化を実行します。これはデフォルトです。
 - LOCAL を指定すると、ローカル・インスタンスに対してのみこの同期化を実行します。
- インスタンスでデータベースをオープンしておく必要があります。

end_session_clauses

end_session_clauses を使用すると、現行のセッションを終了することができます。

DISCONNECT SESSION 句

DISCONNECT SESSION 句を使用すると、専用サーバーのプロセス（接続が共有サーバー経由の場合は、仮想回路）を破棄することによって、現行のセッションが切断されます。この句を使用する場合、インスタンスでデータベースをオープンする必要があります。次の両方の値を V\$SESSION ビューで指定して、このセッションを識別する必要があります。

- *integer1* には、SID 列の値を指定します。
- *integer2* には、SERIAL# 列の値を指定します。

システム・パラメータを適切に設定した場合、アプリケーション・フェイルオーバーが有効になります。

- POST_TRANSACTION を設定すると、セッションが切断される前に、実行中のトランザクションを完了します。セッションに実行中のトランザクションがない場合、この句は、後述の KILL SESSION と同様に動作します。
- IMMEDIATE を設定すると、セッションを切断し、実行中のトランザクションの完了を待たずに、すぐにセッション全体の状態をリカバリします。
 - POST_TRANSACTION を指定した場合、セッションに実行中のトランザクションがあれば、IMMEDIATE キーワードは無視されます。
 - POST_TRANSACTION を指定しない場合、または POST_TRANSACTION を指定していてもセッションに実行中のトランザクションがない場合、この句は、後述の KILL SESSION IMMEDIATE と同様の効果があります。

KILL SESSION 句

KILL SESSION は、セッションを使用禁止にし、実行中のトランザクションをロールバックして、すべてのセッション・ロックを解放し、セッション・リソースを一部リカバリします。この句を使用するには、インスタンスでデータベースをオープンしている必要があります。また、セッションおよび中断されるセッションは、同じインスタンスにある必要があります。次の両方の値を V\$SESSION ビューで指定して、このセッションを識別する必要があります。

- `integer1` には、SID 列の値を指定します。
- `integer2` には、SERIAL# 列の値を指定します。

リモート・データベースからの応答を待つ、トランザクションをロールバックするなどの、最後まで完了する必要があるアクティビティをセッションが実行している場合、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されます。待ち時間が 1 分以上続く場合は、中断されるセッションにマークが付けられ、マークが付けら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されます。アクティビティが完了すると、PMON バックグラウンド・プロセスは、セッションに中断されたことを示すマークを付けます。

セッションに実行中のトランザクションがあるかどうかにかかわらず、セッション・ユーザーがセッションに要求を発行してセッションが中断されたことを示すメッセージを受け取るまで、Oracle は、セッション全体の状態をリカバリしません。

IMMEDIATE IMMEDIATE を指定すると、実行中のトランザクションをロールバックしてすべてのセッション・ロックを解放し、セッション全体の状態をリカバリしてから、ユーザーに制御を戻すように Oracle に指示できます。

DISTRIBUTED RECOVERY 句

DISTRIBUTED RECOVERY 句によって、分散リカバリを使用可能または使用禁止にします。この句を使用する場合、インスタンスでデータベースをオープンする必要があります。

ENABLE ENABLE を指定すると、分散リカバリが使用可能になります。シングルプロセス環境では、分散リカバリを開始する場合にこの句を使用する必要があります。

トランザクションに関係があるリモート・ノードにアクセスできない場合、インダウト・トランザクションをリカバリするには、ENABLE DISTRIBUTED RECOVERY 文を複数回発行する必要がある場合もあります。インダウト・トランザクションは、データ・ディクショナリ・ビュー DBA_2PC_PENDING に表示されます。

DISABLE DISABLE を指定すると、分散リカバリが使用禁止になります。

RESTRICTED SESSION 句

RESTRICTED SESSION を指定すると、Oracle にログインできるユーザーを制限できます。

インスタンスでデータベースがマウントされていてもディスマウントされていても、またはオープン状態でもクローズ状態でも、この句を使用できます。

ENABLE ENABLE を指定すると、RESTRICTED SESSION システム権限が付与されているユーザーのみが Oracle にログインできます。既存のセッションは終了しません。

DISABLE DISABLE を指定すると、ENABLE RESTRICTED SESSION 句とは逆の動作をします。つまり、CREATE SESSION システム権限が付与されているすべてのユーザーが Oracle にログインできます。これはデフォルトです。

FLUSH SHARED_POOL 句

FLUSH SHARED POOL 句を指定すると、SGA の共有プール上のすべてのデータが消去されます。共有プールは次のものを格納します。

- キャッシュされたデータ・ディクショナリ情報
- SQL 文の共有 SQL 領域、共有 PL/SQL 領域、ストアド・プロシージャ、ファンクション、パッケージおよびトリガー

この文は、現在実行中の項目に対する共有 SQL 領域および共有 PL/SQL 領域を消去しません。インスタンスでデータベースがマウントされていてもディスマウントされていても、またはオープン状態でもクローズ状態でも、この句を使用できます。

SWITCH LOGFILE 句

SWITCH LOGFILE 句を指定すると、現行の REDO ログ・ファイル・グループのファイルが一杯であるかどうかにかかわらず、新しい REDO ログ・ファイル・グループへの書込みを明示的に強制開始させます。ログ・スイッチを強制的に発生させると、チェックポイントが実行されますが、チェックポイントが完了する前に、ただちに制御が戻されます。この句を使用する場合、インスタンスでデータベースをオープンする必要があります。

SUSPEND | RESUME

SUSPEND 句を指定すると、すべての I/O（データ・ファイル、制御ファイルおよびファイル・ヘッダー）を停止し、すべてのインスタンスで実行中のトランザクションを処理せずにデータベースのコピーを作成可能にします。

制限事項：

- ホット・バックアップ・モードでデータベースの表領域を確保するまで、この句は使用できません。
- システムが停止中に新しいインスタンスを起動する場合、この新しいインスタンスは停止しません。

RESUME 句を指定すると、問合せおよび I/O に対して、再度、データベースを使用可能にします。

QUIESCE RESTRICTED | UNQUIESCE

QUIESCE RESTRICTED 句および UNQUIESCE 句を使用すると、データベースを**休止状態**にしたり、休止状態から戻すことができます。この状態では、データベース管理者は、トランザクション、問合せまたは PL/SQL 操作が同時に存在する状態では安全に実行できない管理操作を実行することができます。

複数の QUIESCE RESTRICTED 文または UNQUIESCE 文が異なるセッションまたはインスタンスで同時に発行された場合、1 つを除いた他のすべてのセッションまたはインスタンスにエラーが戻されます。

QUIESCE RESTRICTED

QUIESCE RESTRICTED を指定すると、データベースは休止状態になります。この句は、データベースがオープン中のすべてのインスタンスに次の影響を与えます。

- Oracle は、すべてのインスタンスのデータベース・リソース・マネージャに、アクティブでないすべてのセッション（SYS および SYSTEM 以外）をアクティブにしないように指示します。SYS および SYSTEM 以外のユーザーは、新しいトランザクション、問合せ、フェッチまたは PL/SQL 操作を起動できません。
- Oracle は、SYS または SYSTEM 以外のユーザーが開始した、すべてのインスタンスの既存のトランザクションが終了するまで（コミットまたは異常終了するまで）待機します。Oracle は、SYS または SYSTEM 以外のユーザーが開始した、内部トランザクションにない、すべてのインスタンスで実行中のすべての問合せ、フェッチおよび PL/SQL プロシージャが終了するまで待機します。連続する複数の Oracle Call インタフェース (OCI) のフェッチによって問合せが実行される場合、Oracle はすべてのフェッチが終了するまで待機しません。現行のフェッチが終了するまで待機しますが、次のフェッチは行われません。Oracle は、エンキューなどの共有リソースを保持するすべてのセッション（SYS および SYSTEM 以外のセッション）がリソースを解放するまで待機します。すべての操作が完了した後、Oracle はデータベースを休止状態にし、QUIESCE RESTRICTED 文の実行を終了します。

- 共有サーバー・モードでインスタンスを実行している場合は、Oracle は、データベース・リソース・マネージャに、SYS または SYSTEM 以外のユーザーによるログインを阻止するように指示します。非共有サーバー・モードでインスタンスを実行している場合、そのインスタンスへのユーザー・ログインに制限はありません。

休止状態中、すべてのインスタンスにおいてリソース・マネージャのプランは変更できません。

UNQUIESCE

UNQUIESCE を指定すると、データベースを休止状態から戻します。これによって、SYS または SYSTEM 以外のユーザーによって開始された、トランザクション、問合せ、フェッチおよび PL/SQL プロシージャを再開できます。UNQUIESCE 文は、QUIESCE RESTRICTED 文を発行したセッションと同じセッションで起動する必要はありません。

SHUTDOWN 句

SHUTDOWN 句は、システムに Oracle の共有サーバー・アーキテクチャを使用している場合のみに関連します。dispatcher_name で識別されたディスパッチャを停止します。dispatcher_name は、'Dxxx' という形式の文字列である必要があります。xxx はディスパッチャの数です。ディスパッチャ名のリストを取得するには、V\$DISPATCHER 動的パフォーマンス・ビューの NAME 列を問い合わせます。

- IMMEDIATE を指定した場合、ディスパッチャは新しい接続の受入れをすぐに中止し、そのディスパッチャによる既存の接続はすべて終了されます。すべてのセッションがクリーンアップされてから、ディスパッチャ・プロセスは停止します。
- IMMEDIATE を指定しない場合、ディスパッチャは新しい接続の受入れをすぐに中止しますが、すべてのユーザーが切断し、すべてのデータベース・リンクが終了されるのを待ちます。その後、ディスパッチャは停止されます。

REGISTER 句

REGISTER を指定すると、PMON バックグラウンド・プロセスによってリスナーにインスタンスがただちに登録されます。この句を指定しない場合、PMON が次に検出ルーチンを実行するまでインスタンスの登録は行われません。その結果、クライアントは、リスナー起動後最大 60 秒間サービスにアクセスできない可能性があります。

参照： PMON バックグラウンド・プロセスおよびリスナーの詳細は、『Oracle9i データベース概要』および『Oracle9i Net Services 管理者ガイド』を参照してください。

alter_system_set_clause

`alter_system_set_clause`を使用すると、すべての初期化パラメータ値を設定またはリセットできます。パラメータの詳細は、9-32 ページの「[初期化パラメータおよび ALTER SYSTEM](#)」を参照してください。

パラメータ値を設定するときに、次の設定も行えます。

COMMENT

COMMENT 句を使用すると、コメント文字列をパラメータ値の変更に対応付けることができます。SPFILE をあわせて指定すると、パラメータ・ファイルにコメントが表示され、そのパラメータに対する直前の変更がわかります。

DEFERRED

DEFERRED キーワードを指定すると、データベースに接続するその後のセッションに対するパラメータの値を設定または変更できます。現行のセッションでは変更前の値が残ります。

SCOPE

SCOPE 句を使用すると、変更が有効になるタイミングを指定できます。有効範囲は、データベースの起動に使用したファイルがパラメータ・ファイル（pfile）か、サーバー・パラメータ・ファイル（spfile）かによって異なります。

MEMORY MEMORY を指定すると、変更がメモリーで行われ、ただちに有効になり、データベースが停止するまで持続されます。パラメータ・ファイル（pfile）を使用してデータベースを起動すると、指定した範囲のみで有効になります。

SPFILE SPFILE を指定すると、変更がサーバー・パラメータ・ファイルで行われます。新しい設定は、データベースが次に停止し、再起動されたときに有効になります。**静的パラメータ値を変更したときには、SPFILE を指定する必要があります。**

BOTH BOTH を指定すると、変更がメモリーとサーバー・パラメータ・ファイルの両方で行われます。新しい設定はただちに有効になり、データベースが停止し、再起動された後も持続します。

データベースの起動でサーバー・パラメータ・ファイルを使用した場合は、BOTH がデフォルトです。データベースの起動でパラメータ・ファイルを使用した場合は、MEMORY がデフォルトで、指定した範囲のみで有効になります。

SID

SID 句は、Real Application Clusters 環境のみに関連します。この句を使用すると、値を有効にするインスタンスの SID を指定することができます。

- すべてのインスタンスに対してパラメータ値を変更する場合、SID = '*' を指定します。
- *sid* のインスタンスのみでパラメータ値を変更する場合、SID = '*sid*' を指定します。この設定は、SID = '*' を指定する前後の ALTER SYSTEM SET 文より優先されます。

この句を指定しない場合は、次のようになります。

- pfile（クライアント側の初期化パラメータ・ファイル）を使用してインスタンスを起動する場合、現行のインスタンスの SID とみなされます。
- spfile（サーバ・パラメータ・ファイル）を使用してインスタンスを起動する場合、SID = '*' を指定したとみなされます。

現行のインスタンス以外のインスタンスを指定すると、そのインスタンスに、メモリーのパラメータ値の変更を通知するメッセージが送信されます。

alter_system_reset_clause

alter_system_reset_clause は、Real Application Clusters 環境で使用されます。サーバ・パラメータ・ファイルですべてのインスタンス用にパラメータが設定されていても、各インスタンスを個々に制御することができます。SCOPE 句は、*alter_system_set_clause* と同様に動作します。

SID SID 句を指定すると、以前、インスタンスに対して ALTER SYSTEM SET ...SID = '*sid*' 文で設定したパラメータを削除できます。インスタンスは、パラメータ値を前後の ALTER SYSTEM SET ...SID = '*' 文で指定されたものとみなします。

参照： Real Application Clusters 環境の各インスタンス用のパラメータ設定の詳細は、『Oracle9i Real Application Clusters 配置およびパフォーマンス』を参照してください。

初期化パラメータおよび ALTER SYSTEM

ここでは、初期化パラメータをアルファベット順に説明します。概要のみを記述しています。パラメータの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

ACTIVE_INSTANCE_COUNT

パラメータ・タイプ	整数
デフォルト値	なし
パラメータ・クラス	静的
値の範囲	1 または クラスタ内のインスタンスの数以上（1 以外の値はインスタンスのアクティブまたはスタンバイ状態に影響しない）
Oracle9i Real Application Clusters	すべてのインスタンスに、このパラメータを設定する必要がある。また、複数インスタンスには、同じ値を指定する必要がある。

注意： このパラメータは、インスタンスを 2 つのみ持つクラスタ内でのみ機能します。

ACTIVE_INSTANCE_COUNT を使用すると、2 インスタンス・クラスタの 1 つのインスタンスをプライマリ・インスタンスとして、別のインスタンスをセカンダリ・インスタンスとして指定できます。このパラメータは 3 つ以上のインスタンスを持つクラスタでは機能しません。

AQ_TM_PROCESSES

パラメータ・タイプ	整数
デフォルト値	0
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	0 ～ 10

AQ_TM_PROCESSES によって、キュー・メッセージの時間監視が使用可能になります。キュー・メッセージの時間は、デレイおよび時間切れプロパティを示すメッセージで使用できます。1 ～ 10 までの値を指定すると、メッセージの監視用のキュー・モニター・プロセスがその数だけ作成されます。AQ_TM_PROCESSES に値を設定しないか、または 0 に設定した場合、キュー・モニターは作成されません。

ARCHIVE_LAG_TARGET

パラメータ・タイプ	整数
デフォルト値	0（使用禁止）
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	0、または 60 ～ 7200 の整数
Oracle9i Real Application Clusters	複数インスタンスには同じ値を使用する必要がある。

ARCHIVE_LAG_TARGET によって、データ損失量を制限できます。また、指定した時間間隔でログ・スイッチを強制することによって、スタンバイ・データベースの可用性を効果的に高めることができます。

AUDIT_FILE_DEST

パラメータ・タイプ	文字列
構文	AUDIT_FILE_DEST = 'directory'
デフォルト値	ORACLE_HOME/rdbms/audit
パラメータ・クラス	静的

AUDIT_FILE_DEST には、監査ファイルが格納されているディレクトリを指定します。

AUDIT_TRAIL

パラメータ・タイプ	文字列
構文	AUDIT_TRAIL = {NONE FALSE DB TRUE OS}
デフォルト値	なし
パラメータ・クラス	静的

AUDIT_TRAIL は、監査証拠への行の自動書込みを使用可能または使用禁止にするために使用します。

BACKGROUND_CORE_DUMP

パラメータ・タイプ	文字列
構文	BACKGROUND_CORE_DUMP = {FULL PARTIAL}
デフォルト値	PARTIAL
パラメータ・クラス	静的

BACKGROUND_CORE_DUMP は、主に UNIX のパラメータです。Oracle バックグラウンド・プロセスのコア・ファイルに SGA を含めるかどうかを指定します。

BACKGROUND_DUMP_DEST

パラメータ・タイプ	文字列
構文	BACKGROUND_DUMP_DEST = {pathname directory}
デフォルト値	オペレーティング・システム依存
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	有効なローカル・パス、ディレクトリまたはディスク

BACKGROUND_DUMP_DEST には、Oracle の操作中に、バックグラウンド・プロセス (LGWR、DBWn など) のデバッグ・トレース・ファイルが書き込まれる (ディレクトリまたはディスクの) パス名を指定します。

BACKUP_TAPE_IO_SLAVES

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	動的: ALTER SYSTEM ... DEFERRED
値の範囲	true false

BACKUP_TAPE_IO_SLAVES には、Recovery Manager がテープへのデータのバックアップ、コピー、リストアに I/O サーバー・プロセス (スレーブ) を使用するかどうかを指定します。値を true に設定すると、I/O サーバー・プロセスはテープ・デバイスに対する書込みまたは読込みに使用されます。値を false (デフォルト) に設定すると、I/O サーバー・プロセスはバックアップに使用されません。そのかわり、バックアップを行うシャドウ・プロセスがテープ・デバイスにアクセスします。

BITMAP_MERGE_AREA_SIZE

パラメータ・タイプ	整数
デフォルト値	1048576 (1MB)
パラメータ・クラス	静的
値の範囲	オペレーティング・システム依存

注意： インスタンス共有サーバー・オプションで構成されていないかぎり、BITMAP_MERGE_AREA_SIZE パラメータを使用することはお薦めしません。かわりに、PGA_AGGREGATE_TARGET を設定して、SQL 作業領域の自動サイズ指定を使用可能にすることをお薦めします。
BITMAP_MERGE_AREA_SIZE は、下位互換性を保つために残されます。

BITMAP_MERGE_AREA_SIZE は、ビットマップ索引を含むシステムのみに関連します。このパラメータには、索引のレンジ・スキャンによって取り出されたビットマップをマージするために使用するメモリー容量を指定します。デフォルト値は 1MB です。シングル・ビットマップにマージするにはビットマップ・セグメントをソートする必要があるため、通常、より大きい値を指定するとパフォーマンスが向上します。

BLANK_TRIMMING

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	静的
値の範囲	true false

BLANK_TRIMMING には、文字データ型のデータ割当て方法を指定します。

BUFFER_POOL_KEEP

パラメータ・タイプ	文字列
構文	<pre>BUFFER_POOL_KEEP = {integer (BUFFERS:integer, LRU_LATCHES:integer)}</pre> <p><i>integer</i> は、バッファ件数、および LRU ラッチ数（オプション）</p>
デフォルト値	なし
パラメータ・クラス	静的

注意： DB_KEEP_CACHE_SIZE パラメータが設定されている場合、このパラメータは無視されます。かわりに、DB_KEEP_CACHE_SIZE を使用することをお薦めします。また、BUFFER_POOL_KEEP は、新しい DB_KEEP_CACHE_SIZE 動的パラメータと組み合わせることはできません。同じパラメータ・ファイルでこの 2 つのパラメータを組み合わせた場合、エラーが発生します。BUFFER_POOL_KEEP は、下位互換性を保つために残されます。

BUFFER_POOL_KEEP は、バッファの合計数（DB_BLOCK_BUFFERS パラメータの値）の近くに KEEP バッファ・プールとして設定することによって、オブジェクトをバッファ・キャッシュに保存できます。また、LRU ラッチの合計数の指定部分を KEEP バッファ・プールに割り当てることができます。

BUFFER_POOL_RECYCLE

パラメータ・タイプ	文字列
構文	<pre>BUFFER_POOL_RECYCLE = {integer (BUFFERS:integer, LRU_LATCHES:integer)}</pre> <p><i>integer</i> は、バッファ件数および LRU ラッチ数（オプション）</p>
デフォルト値	なし
パラメータ・クラス	静的

注意： DB_RECYCLE_CACHE_SIZE パラメータが設定されている場合、このパラメータは無視されます。かわりに、DB_RECYCLE_CACHE_SIZE を使用することをお薦めします。また、BUFFER_POOL_RECYCLE は、新しい DB_RECYCLE_CACHE_SIZE 動的パラメータと組み合わせることはできません。同じパラメータ・ファイルでこの 2 つのパラメータを組み合わせた場合、エラーが発生します。BUFFER_POOL_RECYCLE は、下位互換性を保つために残されます。

BUFFER_POOL_RECYCLE は、バッファの合計数 (DB_BLOCK_BUFFERS パラメータの値) の近くに RECYCLE バッファ・プールとして設定することによって、バッファ・キャッシュ内のオブジェクト・サイズを制限できます。また、LRU ラッチの合計数の指定部分を RECYCLE バッファ・プールに割り当てることができます。

CIRCUITS

パラメータ・タイプ	整数
デフォルト値	次の値が導出される <ul style="list-style-type: none">SESSIONS 値 (共有サーバー・アーキテクチャを使用する場合)0 (共有サーバー・アーキテクチャを使用しない場合)
パラメータ・クラス	静的

CIRCUITS には、インバインドおよびアウトバインド・ネットワーク・セッションに使用可能な、パーチャル・サーキットの合計数を指定します。このパラメータは、インスタンスに必要な SGA の総量に影響するパラメータの 1 つです。

CLUSTER_DATABASE

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	静的
値の範囲	true false
Oracle9i Real Application Clusters	複数インスタンスには、同じ値を指定する必要がある。

CLUSTER_DATABASE は、Oracle9i Real Application Clusters が使用可能かどうかを指定する Oracle9i Real Application Clusters のパラメータです。

CLUSTER_DATABASE_INSTANCES

パラメータ・タイプ	整数
デフォルト値	1
パラメータ・クラス	静的
値の範囲	0（ゼロ）以外の任意の値

CLUSTER_DATABASE_INSTANCES は、クラスタ・データベースの一部として構成されたインスタンスの数を指定する Oracle9i Real Application Clusters のパラメータです。すべてのインスタンスにこのパラメータを設定する必要があります。通常、このパラメータを、Oracle9i Real Application Clusters 環境のインスタンスの数に設定してください。このパラメータを適切に設定すると、メモリー使用を改善できます。

CLUSTER_INTERCONNECTS

パラメータ・タイプ	文字列
構文	CLUSTER_INTERCONNECTS = ifn [: ifn ...]
デフォルト値	なし
パラメータ・クラス	静的
値の範囲	コロンで区切られた、1 つ以上の IP アドレス

CLUSTER_INTERCONNECTS は、Oracle9i Real Application Clusters 環境で使用可能な追加のクラスタ・インターコネクトに関する情報を Oracle に提供します。

COMMIT_POINT_STRENGTH

パラメータ・タイプ	整数
デフォルト値	1
パラメータ・クラス	静的
値の範囲	0 ～ 255

COMMIT_POINT_STRENGTH は、分散データベース・システムのみに関連しています。このパラメータには、分散トランザクション内のコミット・ポイント・サイトを決定する値を指定します。COMMIT_POINT_STRENGTH について最大の値を持つトランザクションのノードが、コミット・ポイント・サイトになります。

COMPATIBLE

パラメータ・タイプ	文字列
構文	COMPATIBLE = <i>release_number</i>
デフォルト値	8.1.0
パラメータ・クラス	静的
値の範囲	今回のリリースに対するデフォルトのリリース
Oracle9i Real Application Clusters	複数インスタンスには、同じ値を指定する必要がある。

COMPATIBLE は、新しいリリースを使用できるようにするとともに、以前のリリースとの下位互換性を保証します。この機能は、以前のリリースに戻す必要がある場合に効果的です。

CONTROL_FILE_RECORD_KEEP_TIME

パラメータ・タイプ	整数
デフォルト値	7 (日)
パラメータ・クラス	動的:ALTER SYSTEM
値の範囲	0 ～ 365 (日)

CONTROL_FILE_RECORD_KEEP_TIME には、制御ファイル内の再利用可能なレコードを再利用するまでの最小経過日数を指定します。再利用可能セクションに新規レコードを追加する必要があり、一番古いレコードがまだ再利用可能になってない場合は、このレコード・セクションは拡張されます。このパラメータを 0 に設定すると、再利用可能セクションは拡張されずに、レコードが必要に応じて再利用されます。

CONTROL_FILES

パラメータ・タイプ	文字列
構文	<code>CONTROL_FILES = filename [, filename [...]]</code> 注意: 制御ファイル名が、Oracle Managed Files となる場合があります。これは、 <code>CREATE CONTROLFILE REUSE</code> 文を使用して制御ファイルが再作成された場合に発生します。
デフォルト値	オペレーティング・システム依存
パラメータ・クラス	静的
値の範囲	1 ～ 8 つのファイル名
Oracle9i Real Application Clusters	複数インスタンスには、同じ値を指定する必要がある。

すべてのデータベースには、データベースの構造（構造体の名前、作成のタイムスタンプ、データ・ファイルおよび REDO ファイルの名前や位置など）を記述する**制御ファイル**があります。CONTROL_FILES には、1 つ以上の制御ファイルの名前を指定します。複数指定する場合は、名前をカンマで区切ります。

CORE_DUMP_DEST

パラメータ・タイプ	文字列
構文	<code>CORE_DUMP_DEST = directory</code>
デフォルト値	ORACLE_HOME/DBS
パラメータ・クラス	動的: ALTER SYSTEM

CORE_DUMP_DEST は、主に UNIX のパラメータであるため、ご使用のプラットフォームではサポートされていない場合があります。CORE_DUMP_DEST には、コア・ファイルをダンプするディレクトリを指定します。

CPU_COUNT

パラメータ・タイプ	整数
デフォルト値	Oracle によって自動的に設定される。
パラメータ・クラス	静的
値の範囲	0 ～無制限

注意： ほとんどのプラットフォームでは、CPU_COUNT の値は Oracle のインスタンスに使用できる CPU の数に自動的に設定されます。CPU_COUNT の値は変更しないでください。

CPU_COUNT には、Oracle が使用できる CPU の数を指定します。シングル CPU コンピュータでは、CPU_COUNT の値は 1 です。

CREATE_BITMAP_AREA_SIZE

パラメータ・タイプ	整数
デフォルト値	8388608 (8 MB)
パラメータ・クラス	静的
値の範囲	オペレーティング・システム依存

注意： インスタンスが共有サーバー・オプションで構成されていないかぎり、CREATE_BITMAP_AREA_SIZE パラメータを使用することはお薦めしません。かわりに、PGA_AGGREGATE_TARGET を設定して、SQL 作業領域の自動サイズ指定を使用可能にすることをお薦めします。CREATE_BITMAP_AREA_SIZE は、下位互換性を保つために残されます。

CREATE_BITMAP_AREA_SIZE はビットマップ索引を含むシステムのみに関連します。このパラメータには、ビットマップの作成に割り当てるメモリー容量（バイト単位）を指定します。デフォルト値は 8MB です。より大きい値を指定すると、その分、索引を速く作成できます。

CREATE_STORED_OUTLINES

構文：

```
CREATE_STORED_OUTLINES = {TRUE | FALSE | category_name} [NOOVERRIDE]
```

CREATE_STORED_OUTLINES パラメータは、セッション中に発行された問合せごとに、自動的にアウトラインを作成および格納するかどうかを決定します。
CREATE_STORED_OUTLINES は、初期化パラメータではありません。

- TRUE に設定した場合、システム内で後続の問合せに対するアウトラインは、自動的に作成可能になります。このアウトラインは、一意のシステム生成名を受け取り、DEFAULT カテゴリに格納されます。すでに特定の問合せに定義したアウトラインが DEFAULT カテゴリに存在する場合は、そのアウトラインが保持され、新しいアウトラインは作成されません。
- FALSE に設定した場合、システムのアウトラインは作成禁止になります。これはデフォルト値です。
- category_name を設定した場合、システムで作成されたアウトラインが category_name で指定したカテゴリに格納されることを除き、TRUE と同じ動作をします。
- NOOVERRIDE に設定した場合、パラメータが明示的に設定されたすべてのセッションの設定はオーバーライドされません。NOOVERRIDE に設定しない場合、この設定は、すべてのセッションで有効になります。

CURSOR_SHARING

パラメータ・タイプ	文字列
構文	CURSOR_SHARING = {SIMILAR EXACT FORCE}
デフォルト値	EXACT
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM

CURSOR_SHARING では、同じカーソルを共有する SQL 文の種類を判断します。

CURSOR_SPACE_FOR_TIME

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	静的
値の範囲	true false

CURSOR_SPACE_FOR_TIME によって、時間を節約するために、カーソルはより多くの領域を使用できるようになります。このパラメータは、共有 SQL 領域およびクライアントのプライベート SQL 領域の両方に影響します。

DB_nK_CACHE_SIZE

パラメータ・タイプ	大整数
構文	DB_nK_CACHE_SIZE = integer [K M G]
デフォルト値	0M（追加のブロック・サイズ・キャッシュは、デフォルトでは構成されない）
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	n = 2、4、8、16、32

DB_nK_CACHE_SIZE では、nK バッファのキャッシュ・サイズを指定します。
DB_BLOCK_SIZE に nK 以外の値が設定されている場合にのみ、このパラメータを設定できます。たとえば、DB_BLOCK_SIZE=4096 の場合、DB_4K_CACHE_SIZE というパラメータの指定は無効です（4 KB ブロックのキャッシュ・サイズは、すでに DB_CACHE_SIZE で指定されているため）。

DB_BLOCK_BUFFERS

パラメータ・タイプ	整数
デフォルト値	次の値が導出される: 48 MB / DB_BLOCK_SIZE
パラメータ・クラス	静的
値の範囲	50 以上。上限は、オペレーティング・システムによって異なる。
Oracle9i Real Application Clusters	複数インスタンスは異なる値を指定でき、必要に応じて値を変更可能。

注意： DB_CACHE_SIZE パラメータが設定されている場合、このパラメータは無視されます。かわりに、DB_CACHE_SIZE を使用することをお勧めします。また、DB_BLOCK_BUFFERS は、新しい DB_CACHE_SIZE 動的パラメータと組み合わせることはできません。同じパラメータ・ファイルでこの 2 つのパラメータを組み合わせた場合、エラーが発生します。
DB_BLOCK_BUFFERS は、下位互換性を保つために残されます。

DB_BLOCK_BUFFERS には、バッファ・キャッシュ内のデータベース・バッファの数を指定します。このパラメータは、インスタンスの SGA の必要メモリー総量の決定にかかわるパラメータの 1 つです。

DB_BLOCK_CHECKING

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM
値の範囲	true false

DB_BLOCK_CHECKING によって、データ・ブロックに対してブロック・チェックを実行するかどうかを制御します。パラメータ値を true に設定すると、すべてのデータ・ブロックのブロック・チェックを実行します。パラメータ値を false に設定すると、ユーザーの表領域に対するブロック・チェックを実行しません。ただし、SYSTEM 表領域に対するブロック・チェックは常にオンです。

DB_BLOCK_CHECKSUM

パラメータ・タイプ	ブール
デフォルト値	true
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	true false

DB_BLOCK_CHECKSUM では、DBWn およびダイレクト・ローダーが、チェックサム（ブロック内に格納されているすべてのバイトから計算された数値）を計算し、ディスクにデータ・ブロックを書き込むときに、すべてのデータ・ブロックのキャッシュ・ヘッダーにそのチェックサムを格納するかどうかを決めます。ブロックが読み込まれると、このパラメータが true でかつブロックの最後の書込みにチェックサムが格納された場合のみ、チェックサムが検証されます。また、すべてのログ・ブロックは、カレント・ログに書き込まれる前に、チェックサムを与えられます。

DB_BLOCK_SIZE

パラメータ・タイプ	整数
デフォルト値	2048
パラメータ・クラス	静的
値の範囲	2048 ～ 32768。ただし、オペレーティング・システムによっては、さらに範囲が狭い場合がある。
Oracle9i Real Application Clusters	すべてのインスタンスに、このパラメータを設定する必要がある。また、複数インスタンスには、同じ値を指定する必要がある。

注意： このパラメータは、データベース作成時に設定してください。
データベース作成後は変更しないでください。

DB_BLOCK_SIZE には、Oracle データベースのブロック・サイズ（バイト単位）を指定します。通常、値は 2048 または 4096 です。データベースを作成した時点で有効な DB_BLOCK_SIZE の値によって、ブロック・サイズが判断されます。値は、初期値に設定しておく必要があります。

DB_CACHE_ADVICE

パラメータ・タイプ	文字列
構文	DB_CACHE_ADVICE = {ON READY OFF}
デフォルト値	OFF
パラメータ・クラス	動的:ALTER SYSTEM

DB_CACHE_ADVICE では、V\$DB_CACHE_ADVICE パフォーマンス・ビューを介して、様々なキャッシュ・サイズでの動作予測用に使用する統計収集が、使用可能または使用禁止にされます。

DB_CACHE_SIZE

パラメータ・タイプ	大整数
構文	DB_CACHE_SIZE = integer [K M G]
デフォルト値	48M
パラメータ・クラス	動的:ALTER SYSTEM

DB_CACHE_SIZE には、プライマリ・ブロック・サイズ (DB_BLOCK_SIZE パラメータで定義済のブロック・サイズ) を持つバッファの DEFAULT バッファ・プール・サイズを指定します。

DB_CREATE_FILE_DEST

パラメータ・タイプ	文字列
構文	DB_CREATE_FILE_DEST = <i>directory</i>
デフォルト値	なし
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM

DB_CREATE_FILE_DEST には、データ・ファイル、制御ファイルおよびオンライン・ログが作成されるデフォルトの位置を設定します。

DB_CREATE_ONLINE_LOG_DEST_n

パラメータ・タイプ	文字列
構文	DB_CREATE_ONLINE_LOG_DEST_n = <i>directory</i> (n は 1 ～ 5)
デフォルト値	なし
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM

DB_CREATE_ONLINE_LOG_DEST_n には、オンライン・ログおよび制御ファイルを作成するデフォルトの位置を設定します。

DB_DOMAIN

パラメータ・タイプ	文字列
構文	DB_DOMAIN = <i>domain_name</i>
デフォルト値	なし
パラメータ・クラス	静的
値の範囲	ピリオドで区切った名前コンポーネントの有効な文字列で、最大 128 文字 (ピリオドを含む)。NULL は指定できない。
Oracle9i Real Application Clusters	すべてのインスタンスに、このパラメータを設定する必要がある。また、複数インスタンスには、同じ値を指定する必要がある。

分散データベース・システムでは、DB_DOMAIN には、ネットワーク構造内でのデータベースの論理上の位置を指定します。このデータベースが分散システムまたはその一部の場合、このパラメータを設定します。この値は、有効な識別子で構成され、ピリオドで区切られたグローバルなデータベース名の拡張コンポーネントで構成されます。DB_DOMAIN には、ドメイン内の各データベースに一意の文字列を指定することをお薦めします。

DB_FILE_MULTIBLOCK_READ_COUNT

パラメータ・タイプ	整数
デフォルト値	8
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM
値の範囲	オペレーティング・システム依存

DB_FILE_MULTIBLOCK_READ_COUNT は、テーブル・スキャン中に、I/O を最小化するために使用できるパラメータの 1 つです。このパラメータには、順次スキャン中に 1 回の I/O 操作で読み取られるブロックの最大数を指定します。フル・テーブル・スキャンに必要な I/O の合計数は、表のサイズ、マルチブロック READ 件数などの条件、およびパラレル実行が操作に対して使用可能な状態になっているかどうかによって異なります。

DB_FILE_NAME_CONVERT

パラメータ・タイプ	文字列
構文	<div>DB_FILE_NAME_CONVERT = [() 'string1' , 'string2' , 'string3' , 'string4' , ... []]</div> <div>この式では、</div> <div>string1 は、プライマリ・データベース・ファイル名のパターン。</div> <div>string2 は、スタンバイ・データベース・ファイル名のパターン。</div> <div>string3 は、プライマリ・データベース・ファイル名のパターン。</div> <div>string4 は、スタンバイ・データベース・ファイル名のパターン。</div> <div>必要な数のプライマリ置換文字列とスタンバイ置換文字列の組合せが使用可能。一重または二重引用符が使用可能。カッコはオプション。</div> <div>設定例：</div> <div>DB_FILE_NAME_CONVERT = ('/dbs/t1/' , '/dbs/t1/s_' , 'dbs/t2/' , 'dbs/t2/s_')</div>
デフォルト値	なし
パラメータ・クラス	静的

DB_FILE_NAME_CONVERT は、リカバリ用のクローン・データベースの作成に役立ちます。このパラメータは、プライマリ・データベース上の新規のデータ・ファイルのファイル名をスタンバイ・データベース上のファイル名に変換します。データ・ファイルをプライマリ・データベースに追加すると、それに対応するファイルをスタンバイ・データベースに追加する必要があります。スタンバイ・データベースを更新するときに、このパラメータは、プライマリ・データベース上のデータ・ファイル名をスタンバイ・データベース上のデータ・ファイル名に変換します。このファイルがスタンバイ・データベース上に書込み可能な状態で存在している必要があります。そうでない場合、リカバリ・プロセスはエラーによって停止します。

DB_FILES

パラメータ・タイプ	整数
デフォルト値	200
パラメータ・クラス	静的
値の範囲	最小値: データベース内のカレント・データ・ファイルの実際の数 最大値: 最後に CREATE DATABASE または CREATE CONTROLFILE が実行されたときに、MAXDATAFILES 句に指定された値
Oracle9i Real Application Clusters	すべてのインスタンスに、このパラメータを設定する必要がある。また、複数インスタンスには、同じ値を指定する必要がある。

DB_FILES には、このデータベースに対してオープンできるデータベース・ファイルの最大数を指定します。最大有効値は、ADD DATAFILE 文によって追加されるファイルを含め、このデータベースに対して指定されるすべてのファイルの最大数です。この値は、オペレーティング・システムによって異なります。

DB_KEEP_CACHE_SIZE

パラメータ・タイプ	大整数
構文	DB_KEEP_CACHE_SIZE = integer [K M G]
デフォルト値	0M (KEEP キャッシュは、デフォルトでは構成されない)
パラメータ・クラス	動的: ALTER SYSTEM

DB_KEEP_CACHE_SIZE では、KEEP バッファ・プール内のバッファ件数を指定します。KEEP バッファ・プール内のバッファのサイズは、プライマリ・ブロック・サイズ (DB_BLOCK_SIZE パラメータで定義済のブロック・サイズ) です。

DB_NAME

パラメータ・タイプ	文字列
構文	DB_NAME = database_name
デフォルト値	なし
パラメータ・クラス	静的
Oracle9i Real Application Clusters	すべてのインスタンスにこのパラメータを設定する必要がある。そうしない場合は、SQL*Plus 文 STARTUP OPEN または SQL 文 ALTER DATABASE MOUNT に同じ値を指定する必要がある。

DB_NAME には、最大 8 文字のデータベース識別子を指定します。このパラメータを指定する場合は、CREATE DATABASE 文で指定された名前に対応する値である必要があります。DB_NAME の使用は任意ですが、一般的に、DB_NAME は CREATE DATABASE 文を発行する前に設定し、その後、その文の中で参照します。

DB_RECYCLE_CACHE_SIZE

パラメータ・タイプ	大整数
構文	DB_RECYCLE_CACHE_SIZE = integer [K M G]
デフォルト値	0M (RECYCLE キャッシュは、デフォルトでは構成されない)
パラメータ・クラス	動的: ALTER SYSTEM

DB_RECYCLE_CACHE_SIZE には、RECYCLE バッファ・プールのサイズを指定します。RECYCLE プール内のバッファのサイズは、DB_BLOCK_SIZE で定義済のプライマリ・ブロック・サイズです。

DB_WRITER_PROCESSES

パラメータ・タイプ	整数
デフォルト値	1
パラメータ・クラス	静的
値の範囲	1 ~ 10

DB_WRITER_PROCESSES は、データを大量に変更するシステムに便利です。このパラメータには、インスタンスに対するデータベース・ライター・プロセス数の初期値を指定します。

DBLINK_ENCRYPT_LOGIN

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	静的
値の範囲	true false

パスワードを使用してデータベースに接続する場合、Oracle は、パスワードをデータベースに送信する前に暗号化します。DBLINK_ENCRYPT_LOGIN には、データベース・リンクを介して他の Oracle サーバーに接続する場合に、暗号化パスワードを使用するかどうかを指定します。

DBWR_IO_SLAVES

パラメータ・タイプ	整数
デフォルト値	0
パラメータ・クラス	静的
値の範囲	0 以上。上限は、オペレーティング・システムによって異なる。

DBWR_IO_SLAVES は、データベース・ライター・プロセス (DBW0) を 1 つのみ持つシステムに関連します。DBW0 プロセスで使用される I/O サーバー・プロセスの数を指定します。DBW0 プロセスおよびそのサーバー・プロセスは、常にディスクに書き込みます。デフォルト値は 0 で、I/O サーバー・プロセスは使用されません。

DISK_ASYNC_IO

パラメータ・タイプ	ブール
デフォルト値	true
パラメータ・クラス	静的
値の範囲	true false

DISK_ASYNC_IO は、データ・ファイル、制御ファイル、およびログ・ファイルへの I/O が非同期かどうかを制御します (Real Application Clusters プロセスでは、表スキャン中に、CPU 処理と I/O 要求をオーバーラップできるかどうかを制御します)。ご使用のプラットフォームがディスクへの非同期 I/O をサポートしている場合は、このパラメータをデフォルト値のままにしておくことをお勧めします。ただし、非同期 I/O の実装が安定していない場合は、このパラメータを false に設定することで、非同期 I/O を使用禁止にできます。プラットフォームがディスクへの非同期 I/O をサポートしていない場合、このパラメータは無効です。

DISPATCHERS

パラメータ・タイプ	文字列
構文	<pre>DISPATCHERS = 'dispatch_clause' dispatch_clause::= (PROTOCOL = protocol) (ADDRESS = address) (DESCRIPTION = description) [options_clause] options_clause::= (DISPATCHERS = integer SESSIONS = integer CONNECTIONS = integer TICKS = seconds POOL = {1 ON YES TRUE BOTH ({IN OUT} = ticks) 0 OFF NO FALSE ticks} MULTIPLEX = {1 ON YES TRUE 0 OFF NO FALSE BOTH IN OUT} LISTENER = tnsname SERVICE = service INDEX = integer)</pre>
デフォルト値	なし
パラメータ・クラス	動的: ALTER SYSTEM

DISPATCHERS は、共有サーバー・アーキテクチャ内のディスパッチャ・プロセスを構成します。Oracle のパラメータ解析部は、属性を順番にかかわりなく大 / 小文字を区別せずに指定できるような名前値の構文をサポートします。たとえば、次のように割り当てることができます。

```
DISPATCHERS = "(PROTOCOL=TCP) (DISPATCHERS=3)"
```

DISTRIBUTED_TRANSACTIONS

パラメータ・タイプ	整数
デフォルト値	25 * TRANSACTIONS
パラメータ・クラス	静的
値の範囲	0 ～ TRANSACTIONS パラメータの値

DISTRIBUTED_TRANSACTIONS は、Oracle の分散システム機能を使用している場合のみ関連します。このパラメータには、当該データベースが同時に関与できる分散トランザクションの最大数を指定します。このパラメータには、パラメータ TRANSACTIONS の値を超える値は指定できません。

DRS_START

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	動的:ALTER SYSTEM
値の範囲	true false

DRS_START で、Oracle は障害時リカバリ・モニター (DRMON) ・プロセスを開始する必要があるかどうかを判断できます。DRMON は、非致命的な Oracle のバックグラウンド・プロセスであり、インスタンスが存在するかぎり存在します。

EVENT

パラメータ・タイプ	文字列
デフォルト値	なし
パラメータ・クラス	静的

EVENT は、システムをデバッグするために使用されるパラメータです。変更する場合は、オラクル社カスタマ・サポート・センターの指示に従ってください。

FAL_CLIENT

パラメータ・タイプ	文字列
構文	FAL_CLIENT = <i>string</i>
デフォルト値	なし
パラメータ・クラス	動的:ALTER SYSTEM

FAL_CLIENT には、フェッチ・アーカイブ・ログ (FAL) ・サービスが使用する FAL クライアント名を指定します。FAL クライアント名は、FAL_SERVER パラメータで構成され、FAL クライアントの参照に使用されます。この値は Oracle Net のサービス名です。このサービス名は、FAL クライアント (スタンバイ・データベース) を指すように、FAL サーバー・システム上で適切に構成されていることを前提とします。

FAL_SERVER

パラメータ・タイプ	文字列
構文	FAL_SERVER = <i>string</i>
デフォルト値	なし
パラメータ・クラス	動的:ALTER SYSTEM

FAL_SERVER には、スタンバイ・データベースの FAL サーバーを指定します。この値は Oracle Net のサービス名です。このサービス名は、必要な FAL サーバーを指すように、スタンバイ・データベース・システムで適切に構成されていることを前提とします。

FAST_START_IO_TARGET

パラメータ・タイプ	整数
デフォルト値	キャッシュ内のすべてのバッファ
パラメータ・クラス	動的:ALTER SYSTEM
値の範囲	1000 ～キャッシュ内のすべてのバッファ。0 に設定すると、リカバリ I/O の制限は使用禁止になる。
Oracle9i/ Real Application Clusters	複数インスタンスには異なる値を指定でき、実行時に値を変更可能。

注意： FAST_START_MTTR_TARGET パラメータが設定されている場合、このパラメータは無視されます。FAST_START_MTTR_TARGET を使用することをお勧めします。FAST_START_IO_TARGET は、下位互換性を保つために残されます。

FAST_START_IO_TARGET (Oracle Enterprise Edition でのみ使用可能) には、クラッシュまたはインスタンスのリカバリ時に必要な I/O の数を指定します。

FAST_START_MTTR_TARGET

パラメータ・タイプ	整数
デフォルト値	0
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	0 ~ 3600 秒
Oracle9i Real Application Clusters	複数インスタンスには異なる値を指定でき、実行時に値を変更可能。

FAST_START_MTTR_TARGET には、データベースがシングル・インスタンスのクラッシュ・リカバリを実行するまでにかかる秒数を指定します。FAST_START_MTTR_TARGET を指定した場合、次のようになります。

- FAST_START_IO_TARGET でオーバーライドされます。
- LOG_CHECKPOINT_INTERVAL でオーバーライドされます。

FAST_START_PARALLEL_ROLLBACK

パラメータ・タイプ	文字列
構文	FAST_START_PARALLEL_ROLLBACK = {HIGH LOW FALSE}
デフォルト値	LOW
パラメータ・クラス	動的: ALTER SYSTEM

FAST_START_PARALLEL_ROLLBACK は、パラレル・ロールバックを実行するために存在するプロセスの最大数を判断します。このパラメータは一部またはすべてのトランザクションの実行に長時間かかるシステムで役立ちます。

FIXED_DATE

パラメータ・タイプ	文字列
構文	FIXED_DATE = YYYY-MM-DD-HH24:MI:SS（またはデフォルトの Oracle 日付書式）
デフォルト値	なし
パラメータ・クラス	動的: ALTER SYSTEM

FIXED_DATE には、SYSDATE が現在の日付のかわりに常に戻す定数の日付を設定します。このパラメータは、主にテストに役立ちます。この値は前述の書式、またはデフォルトの Oracle 日付書式（時刻なし）で設定できます。

GC_FILES_TO_LOCKS

パラメータ・タイプ	文字列
構文	GC_FILES_TO_LOCKS = ' <i>{file_list=lock_count[!blocks] [EACH] [:...]}</i> ' 引用符の中に空白を入れることはできない。
デフォルト値	なし
パラメータ・クラス	静的
Oracle9i Real Application Clusters	すべてのインスタンスに、このパラメータを設定する必要がある。また、複数インスタンスには、同じ値を指定する必要がある。値を変更するには、クラスタ内のすべてのインスタンスを停止し、各インスタンスの値を変更してから、各インスタンスを再起動する必要がある。

注意： このパラメータをデフォルト以外の値に設定すると、Oracle9i Real Application Clusters でキャッシュ・フュージョン処理が使用禁止になります。

GC_FILES_TO_LOCKS は、排他モードで実行中のインスタンスには影響しない、Oracle9i Real Application Clusters のパラメータです。データ・ファイルに対するリリース 9.0.1 より前のパラレル・キャッシュ管理（PCM）ロックのマップを制御します。

GLOBAL_CONTEXT_POOL_SIZE

パラメータ・タイプ	文字列
デフォルト値	1MB
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	MB 単位の任意の整数値

GLOBAL_CONTEXT_POOL_SIZE には、グローバル・アプリケーション・コンテキストの格納および管理用に、SGA に割り当てるメモリー量を指定します。

GLOBAL_NAMES

パラメータ・タイプ	ブール
デフォルト値	true
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM
値の範囲	true false

GLOBAL_NAMES には、データベース・リンクを、接続するデータベースと同じ名前にする必要があるかどうかを指定します。GLOBAL_NAMES の値が false の場合、確認されません。分散処理を使用する、または使用する予定がある場合、ネットワーク環境においてデータベースおよびリンクのネーミング規則の一貫性を保つために、このパラメータを true に設定することをお薦めします。

HASH_AREA_SIZE

パラメータ・タイプ	整数
デフォルト値	次の値が導出される: 2 * SORT_AREA_SIZE
パラメータ・クラス	動的: ALTER SESSION
値の範囲	0 以上。上限は、オペレーティング・システムによって異なる。

注意： インスタンスが共有サーバー・オプションで構成されていない場合、HASH_AREA_SIZE パラメータを使用することはお薦めしません。かわりに、PGA_AGGREGATE_TARGET を設定して、SQL 作業領域の自動サイズ指定を使用可能にすることをお薦めします。HASH_AREA_SIZE は、下位互換性を保つために残されます。

HASH_AREA_SIZE はパラレル実行操作、および DML または DDL 文の間合せ部分に関連します。このパラメータには、ハッシュ結合で使用するメモリーの最大容量（バイト）を指定します。

HASH_JOIN_ENABLED

パラメータ・タイプ	ブール
デフォルト値	true
パラメータ・クラス	動的:ALTER SESSION
値の範囲	true false

HASH_JOIN_ENABLED には、オブティマイザで選択される結合方法にハッシュ結合を使用するかどうかを指定します。false に設定すると、ハッシュ結合を結合方法として選択できません。true に設定すると、オブティマイザによってハッシュ結合とその他の結合とのコストが比較され、コスト効率が最適な場合、ハッシュ結合が採用されます。データ・ウェアハウス・アプリケーション用に、このパラメータを true に設定することをお勧めします。

HI_SHARED_MEMORY_ADDRESS

パラメータ・タイプ	整数
デフォルト値	0
パラメータ・クラス	静的

HI_SHARED_MEMORY_ADDRESS には、SGA の実行時の開始アドレスを指定します。これらのパラメータは、リンク時に SGA の開始アドレスを指定するプラットフォームでは無視されます。

HS_AUTOREGISTER

パラメータ・タイプ	ブール
デフォルト値	true
パラメータ・クラス	動的:ALTER SYSTEM
値の範囲	true false

HS_AUTOREGISTER では、異機種間サービス (HS) エージェントの自動自己登録を使用可能または使用禁止にします。使用可能な場合、サーバーのデータ・ディクショナリに、以前の不明なエージェント・クラスまたは新しいエージェント・バージョンを記述する情報がアップロードされます。

IFILE

パラメータ・タイプ	ファイル
構文	IFILE = <i>parameter_file_name</i>
デフォルト値	なし
パラメータ・クラス	静的
値の範囲	有効なパラメータ・ファイル名
Oracle9i Real Application Clusters	複数インスタンスには、異なる値を指定可能。

IFILE を使用して、現行のパラメータ・ファイルの中に別のパラメータ・ファイルを組み込みます。たとえば、次のように割り当てることができます。

IFILE = COMMON.ORA

INSTANCE_GROUPS

パラメータ・タイプ	文字列
構文	INSTANCE_GROUPS = <i>group_name</i> [, <i>group_name</i> ...]
デフォルト値	なし
パラメータ・クラス	静的
値の範囲	カンマで区切られた、1 つ以上のインスタンス・グループ名
Oracle9i Real Application Clusters	複数インスタンスには、異なる値を指定可能。

INSTANCE_GROUPS は、パラレル・モードでのみ使用できる Oracle9i Real Application Clusters のパラメータです。PARALLEL_INSTANCE_GROUP とともに使用することによって、インスタンスの制限数にパラレル問合せを制限できます。

INSTANCE_NAME

パラメータ・タイプ	文字列
構文	INSTANCE_NAME = instance_id
デフォルト値	インスタンス SID 注意: SID は、ホストでのインスタンスの共有メモリーを識別しますが、他のインスタンスからは一意に識別されません。
パラメータ・クラス	静的
値の範囲	英数字

Oracle9i Real Application Clusters 環境では、複数インスタンスを単一のデータベース・サービスと関連付けることができます。クライアントは、データベースに接続する特定のインスタンスを指定して、Oracle の接続時ロード・バランスをオーバーライドできます。INSTANCE_NAME には、このインスタンスの一意の名前を指定します。

INSTANCE_NUMBER

パラメータ・タイプ	整数
デフォルト値	使用可能な最小番号: インスタンス起動順序、およびその他のインスタンスの INSTANCE_NUMBER の値から導出。Oracle9i Real Application Clusters 構成でない場合は 0。
パラメータ・クラス	静的
値の範囲	1 ～ データベース作成時に指定されたインスタンスの最大数
Oracle9i Real Application Clusters	すべてのインスタンスに異なる値を設定する必要がある。

INSTANCE_NUMBER は、Oracle9i Real Application Clusters のパラメータで、パラレル・モードまたは排他モードで指定できます。記憶域パラメータ FREELIST GROUPS で作成されたデータベース・オブジェクトごとの空きリスト・グループの 1 つにインスタンスをマップする一意の番号を指定します。

JAVA_MAX_SESSIONSPACE_SIZE

パラメータ・タイプ	整数
デフォルト値	0
パラメータ・クラス	静的
値の範囲	0 ～ 4GB

Java セッション・スペースは、1 つのデータベースから他のデータベースをコールする Java を保持するメモリーです。JAVA_MAX_SESSIONSPACE_SIZE には、サーバーで Java プログラムを実行できるセッション・スペースの最大量をバイト単位で指定します。ユーザー・セッション中に Java がこの量を超えそうになると、Java Virtual Machine がメモリー不足障害で、セッションを終了します。

JAVA_POOL_SIZE

パラメータ・タイプ	文字列
デフォルト値	20000 バイト
パラメータ・クラス	静的
値の範囲	1000000 ～ 10000000000 バイト

JAVA_POOL_SIZE には、Java プールのサイズをバイト単位で指定します。この Java プールから、Java メモリー・マネージャは、ランタイム実行時のほとんどの Java を割り当てます。このメモリーには、コールの終わりで Java セッション・スペースに移行される Java オブジェクトの他に、Java メソッドおよびクラス定義のメモリー内での共有表現が含まれます。

JAVA_SOFT_SESSIONSPACE_LIMIT

パラメータ・タイプ	整数
デフォルト値	0
パラメータ・クラス	静的
値の範囲	0 ～ 4GB

Java セッション・スペースは、1 つのデータベースから他のデータベースをコールする Java を保持するメモリーです。JAVA_SOFT_SESSIONSPACE_LIMIT には、セッションでの Java メモリー使用量の**一時的な制限**をバイト単位で指定します。これは、ユーザー・セッション中の Java が、あまりに多くのメモリーを使用している場合の警告手段となります。ユーザー・セッション中の Java がこのサイズを超えた場合、トレース・ファイルに警告が生成されます。

JOB_QUEUE_PROCESSES

パラメータ・タイプ	整数
デフォルト値	0
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	0 ～ 1000
Oracle9i Real Application Clusters	複数インスタンスには、異なる値を指定可能。

JOB_QUEUE_PROCESSES には、ジョブ実行用に作成できるプロセスの最大数を指定します。インスタンスごとのジョブ・キュー・プロセスの数 (000、... 999) を指定します。レプリケーションでは、データ・リフレッシュにジョブ・キューが使用されます。アドバンスト・キューイングでは、メッセージ伝播にジョブ・キューが使用されます。DBMS_JOB ユーティリティを使用して、ユーザー・ジョブ要求を作成できます。

LARGE_POOL_SIZE

パラメータ・タイプ	文字列
構文	LARGE_POOL_SIZE = integer [K M]
デフォルト値	次の両方に該当する場合は 0。 <ul style="list-style-type: none">■ パラレル実行によって、プールが要求されない。■ DBWR_IO_SLAVES が設定されていない。 それ以外の場合は、PARALLEL_MAX_SERVERS、PARALLEL_THREADS_PER_CPU、CLUSTER_DATABASE_INSTANCES、DISPATCHERS および DBWR_IO_SLAVES の値から導出される。
パラメータ・クラス	静的
値の範囲	600KB ～ 2GB (正確な最大値はオペレーティング・システム依存)

LARGE_POOL_SIZE には、ラージ・プールの割当てヒープ・サイズをバイト単位で指定します。ラージ・プールの割当てヒープは、セッション・メモリの共有サーバー・システム、メッセージ・バッファの平行実行およびディスク I/O バッファのバックアップ・プロセスが使用します（平行実行は、PARALLEL_AUTOMATIC_TUNING に TRUE が設定されている場合のみ、ラージ・プールからバッファを割り当てます）。

LICENSE_MAX_SESSIONS

パラメータ・タイプ	整数
デフォルト値	0
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	0 ～セッション・ライセンスの数
Oracle9i Real Application Clusters	複数インスタンスには、異なる値を指定可能。ただし、データベースをマウントするすべてのインスタンスの合計は、そのデータベースにライセンス付与されたセッションの総数以下にする必要がある。

LICENSE_MAX_SESSIONS には、同時ユーザー・セッションの最大数を指定します。最大数に達すると、RESTRICTED SESSION 権限を持つユーザーのみがデータベースに接続できます。接続できないユーザーは、システムが最大容量に達していることを示す警告メッセージを受け取ります。

LICENSE_MAX_USERS

パラメータ・タイプ	整数
デフォルト値	0
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	0 ～ユーザー・ライセンスの数
Oracle9i Real Application Clusters	複数インスタンスには同じ値を指定する必要がある。異なるインスタンスで、このパラメータに異なる値が指定されると、データベースを最初にマウントするインスタンスの値が優先される。

LICENSE_MAX_USERS には、データベースで作成できるユーザーの最大数を指定します。最大数に達すると、それ以上ユーザーを作成できなくなります。ただし、この最大数を引き上げることはできます。

制限事項：ユーザー数の最大値は、データベースに現在登録されているユーザー数より小さくすることができません。

LICENSE_SESSIONS_WARNING

パラメータ・タイプ	整数
デフォルト値	0
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	0 ～ LICENSE_MAX_SESSIONS パラメータの値
Oracle9i Real Application Clusters	複数インスタンスには、異なる値を指定可能。

LICENSE_SESSIONS_WARNING には、同時ユーザー・セッション数に関する警告限度を指定します。この限度に達しても、ユーザーの追加接続はできますが、Oracle は新規の接続のたびにアラート・ファイルにメッセージを書き込みます。この限度に達した後に接続した RESTRICTED SESSION 権限を持つユーザーは、システムがその容量制限に近づいていることを示す警告メッセージを受け取ります。

LOCAL_LISTENER

パラメータ・タイプ	文字列
構文	LOCAL_LISTENER = <i>network_name</i>
デフォルト値	(ADDRESS = (PROTOCOL=TCP) (HOST=) (PORT=1521))
パラメータ・クラス	静的

LOCAL_LISTENER には、Oracle Net ローカル・リスナー（このインスタンスと同じマシン上で実行中のリスナー）のアドレスまたはアドレス・リストを解決するネットワーク名を指定します。アドレスまたはアドレス・リストは、TNSNAMES.ORA ファイルまたはご使用のシステム用に構成されている他のアドレス・リポジトリで指定されます。

LOCK_NAME_SPACE

パラメータ・タイプ	文字列
構文	LOCK_NAME_SPACE = <i>namespace</i>
デフォルト値	なし
パラメータ・クラス	静的
値の範囲	最大 8 文字までの英数字。特殊文字は使用できない。

LOCK_NAME_SPACE には、分散ロック・マネージャ（DLM）がロック名の生成に使用する名前空間を指定します。スタンバイまたはクローン・データベースが、同じクラスタ上で、プライマリ・データベースと同じデータベース名を持つ場合、このパラメータの設定を検討してください。

LOCK_SGA

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	静的
値の範囲	true false

LOCK_SGA は、SGA 全体を物理メモリーの中にロックします。通常は、SGA を実（物理）メモリーにロックすることをお薦めします。特に、仮想メモリーの使用が、SGA の一部をディスク領域に格納してしまう可能性がある場合はそうしてください。このパラメータをサポートしていないプラットフォームでは、このパラメータは無視されます。

LOG_ARCHIVE_DEST

パラメータ・タイプ	文字列
構文	LOG_ARCHIVE_DEST = <i>filespec</i>
デフォルト値	NULL
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	有効なパス名またはデバイス名（RAW パーティションを除く）
Oracle9i Real Application Clusters	複数インスタンスには、異なる値を指定可能。

注意： Enterprise Edition ユーザーを使用している場合、LOG_ARCHIVE_DEST_n パラメータが設定されていると、このパラメータは無視されます。Oracle Enterprise Edition がインストールされていない場合、またはインストールされているが LOG_ARCHIVE_DEST_n パラメータを指定していない場合は、このパラメータは有効です。

LOG_ARCHIVE_DEST は、データベースを ARCHIVELOG モードで起動している場合またはアーカイブ REDO ログからデータベースをリカバリしている場合のみ適用できます。

LOG_ARCHIVE_DEST は LOG_ARCHIVE_DEST_n パラメータと非互換であり、また LOG_ARCHIVE_DEST_n パラメータに NULL 文字列以外の値がある場合、NULL 文字列 ("") または (' ') で定義されている必要があります。REDO ログ・ファイルをアーカイブするとき、ディスク・ファイルまたはテープ・デバイスのデフォルト位置およびルートを指定するには、テキスト文字列を使用してください (テープへのアーカイブは、すべてのオペレーティング・システムでサポートされているわけではありません)。この値には、RAW パーティションは指定できません。

LOG_ARCHIVE_DEST_n

パラメータ・タイプ	文字列
構文	<pre>LOG_ARCHIVE_DEST_[1 2 3 4 5 6 7 8 9 10] = "null_string" ((SERVICE=service LOCATION=location) [AFFIRM NOAFFIRM] [ALTERNATE=destination NOALTERNATE] [ARCH LGWR] [DELAY [=minutes] NODELAY] [DEPENDENCY=destination NODEPENDENCY] [MANDATORY OPTIONAL] [MAX_FAILURE=count NOMAX_FAILURE] [QUOTA_SIZE=blocks NOQUOTA_SIZE] [QUOTA_USED=blocks NOQUOTA_USED] [REGISTER NOREGISTER] [REGISTER=location_format] [REOPEN [=seconds] NOREOPEN] [SYNC ASYNC=blocks]))</pre>
デフォルト値	なし
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM
値の範囲	有効なキーワードの定義

注意： Oracle Enterprise Edition がインストールされている場合のみ、このパラメータは有効です。Oracle Enterprise Edition がインストールされている場合は、LOG_ARCHIVE_DEST を継続して使用できます。ただし、互換性がないので、LOG_ARCHIVE_DEST_n および LOG_ARCHIVE_DEST の両方を使用することはできません。

LOG_ARCHIVE_DEST_n パラメータ (n= 1、2、3、... 10) は、アーカイブ・ログの宛先を 10 まで定義します。パラメータの整数の接尾辞は、V\$ARCHIVE_DEST 動的パフォーマンス・ビューが表示する **ハンドル** として定義されます。

LOG_ARCHIVE_DEST_STATE_n

パラメータ・タイプ	文字列
構文	LOG_ARCHIVE_DEST_STATE_n = {ENABLE DEFER}
デフォルト値	ENABLE
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM

LOG_ARCHIVE_DEST_STATE_n パラメータ (n= 1、2、3、... 10) には、対応する宛先の可用性状態を指定します。パラメータの接尾辞 (1 ~ 10) は、10 個の対応する LOG_ARCHIVE_DEST_n の宛先パラメータのうちの 1 つを指定します。

LOG_ARCHIVE_DUPLEX_DEST

パラメータ・タイプ	文字列
構文	LOG_ARCHIVE_DUPLEX_DEST = <i>filespec</i>
デフォルト値	なし
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	NULL 文字列、有効なパス名またはデバイス名 (RAW パーティションを除く)

注意： Oracle Enterprise Edition を使用している場合、LOG_ARCHIVE_DEST_n パラメータが設定されていると、このパラメータは無視されます。Oracle Enterprise Edition がインストールされていない場合、またはインストールされているが LOG_ARCHIVE_DEST_n パラメータを指定していない場合は、このパラメータは有効です。

LOG_ARCHIVE_DUPLEX_DEST は、初期化パラメータ LOG_ARCHIVE_DEST に似ています。このパラメータには、2 番目のアーカイブ宛先であるを指定します。**複数**アーカイブ宛先この多重アーカイブ宛先は、must-succeed モードまたは best-effort アーカイブ宛先のいずれかです。どちらになるかは、LOG_ARCHIVE_MIN_SUCCEED_DEST パラメータで指定する、成功する必要のあるアーカイブ宛先の数によって決まります。

LOG_ARCHIVE_FORMAT

パラメータ・タイプ	文字列
構文	LOG_ARCHIVE_FORMAT = <i>filename</i>
デフォルト値	オペレーティング・システム依存
パラメータ・クラス	静的
値の範囲	有効なファイル名を解決する文字列
Oracle9i Real Application Clusters	複数インスタンスには異なる値を指定できるが、同じ値を推奨する。

LOG_ARCHIVE_FORMAT は、REDO ログを ARCHIVELOG モードで使用している場合のみ適用できます。REDO ログ・ファイルをアーカイブするときにデフォルトのファイル名書式を指定するには、テキスト文字列および変数を使用してください。この書式から生成された文字列は、LOG_ARCHIVE_DEST パラメータで指定した文字列に追加されます。

LOG_ARCHIVE_MAX_PROCESSES

パラメータ・タイプ	整数
デフォルト値	1
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	1 ～ 10 の任意の整数

LOG_ARCHIVE_MAX_PROCESSES には、最初に起動されるアーカイバ・バックグラウンド・プロセス（ARC0 から ARC9）の数を指定します。

LOG_ARCHIVE_MIN_SUCCEED_DEST

パラメータ・タイプ	整数
デフォルト値	1
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM
値の範囲	LOG_ARCHIVE_DEST_n を使用している場合、1 ～ 10 LOG_ARCHIVE_DEST および LOG_ARCHIVE_DUPLEX_DEST を使用している場合、1 または 2

LOG_ARCHIVE_MIN_SUCCEED_DEST には、オンライン・ログ・ファイルの再利用を可能にするために必要なアーカイブ先の最小数を定義します。

LOG_ARCHIVE_START

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	静的
値の範囲	true false
Oracle9i Real Application Clusters	複数インスタンスには、異なる値を指定可能。

LOG_ARCHIVE_START は、REDO ログを ARCHIVELOG モードで使用しているときのみ適用できます。このパラメータには、インスタンスの起動時にアーカイブが自動、手動のどちらであるかを指定します。

LOG_ARCHIVE_TRACE

パラメータ・タイプ	整数
デフォルト値	0
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	0、1、2、4、8、16、32、64、128
Oracle9i Real Application Clusters	複数インスタンスには、異なる値を指定可能。

LOG_ARCHIVE_TRACE は、アーカイブ・ログ・プロセスで生成される出力結果を制御します。

LOG_BUFFER

パラメータ・タイプ	整数
デフォルト値	最大値は 512KB、または 128KB × CPU_COUNT のいずれか大きい方
パラメータ・クラス	静的
値の範囲	オペレーティング・システム依存

LOG_BUFFER には、REDO エントリを REDO ログ・ファイルにバッファリングするときに使用されるメモリー容量を、バイト単位で指定します。REDO ログ・エントリには、データベース・ブロック・バッファに対する変更の記録が含まれています。LGWR プロセスは、ログ・バッファから REDO ログ・ファイルに REDO ログ・エントリを書き込みます。

LOG_CHECKPOINT_INTERVAL

パラメータ・タイプ	整数
デフォルト値	オペレーティング・システム依存
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	無制限
Oracle9i Real Application Clusters	複数インスタンスには、異なる値を指定可能。

LOG_CHECKPOINT_INTERVAL には、チェックポイントの頻度、つまり増分チェックポイントと REDO ログに書き込まれた最終ブロック間に存在する可能性のある REDO ログ・ファイルのブロックの数を指定します。この数は、データベース・ブロックではなく、オペレーティング・システムの物理ブロックの数です。

LOG_CHECKPOINT_TIMEOUT

パラメータ・タイプ	整数
デフォルト値	Oracle Security Server: 900 秒 Oracle Security Server Enterprise Edition: 1800 秒
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	0 ～無制限
Oracle9i Real Application Clusters	複数インスタンスには、異なる値を指定可能。

LOG_CHECKPOINT_TIMEOUT には、REDO ログへの最後の書込みが発生した場所（ログの最後尾とも呼ばれる）での、増分チェックポイント以降の時間（秒単位）を指定します。また、このパラメータは、すべてのバッファが integer 秒より長い間、使用済で残らないことを示します。

LOG_CHECKPOINTS_TO_ALERT

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	true false

LOG_CHECKPOINTS_TO_ALERT によって、チェックポイントをアラート・ファイルに記録できます。これによって、指定した頻度でチェックポイントが発生しているかどうかを判断できます。

LOG_FILE_NAME_CONVERT

パラメータ・タイプ	文字列
構文	<pre>LOG_FILE_NAME_CONVERT = [()]'string1' , 'string2' , 'string3' , 'string4' , ...[]]</pre> <p>この式では、</p> <p>string1 は、プライマリ・データベース・ファイル名のパターン。</p> <p>string2 は、スタンバイ・データベース・ファイル名のパターン。</p> <p>string3 は、プライマリ・データベース・ファイル名のパターン。</p> <p>string4 は、スタンバイ・データベース・ファイル名のパターン。</p> <p>必要な数のプライマリ置換文字列とスタンバイ置換文字列の組合せを使用可能。一重または二重引用符を使用可能。カッコはオプション。</p> <p>設定例：</p> <pre>LOG_FILE_NAME_CONVERT=('/dbs/t1/', '/dbs/t1/s_', 'dbs/t2/ ' , 'dbs/t2/s_')</pre>
デフォルト値	なし
パラメータ・クラス	静的
値の範囲	文字列

LOG_FILE_NAME_CONVERT によって、プライマリ・データベース上の新規のログ・ファイルのファイル名がスタンバイ・データベース上のログ・ファイルのファイル名に変換されます。ログ・ファイルをプライマリ・データベースに追加すると、それに対応するファイルをスタンバイ・データベースに追加する必要があります。

LOGMNR_MAX_PERSISTENT_SESSIONS

パラメータ・タイプ	整数
デフォルト値	1
パラメータ・クラス	静的
値の範囲	1 ～ LICENSE_MAX_SESSIONS

LOGMNR_MAX_PERSISTENT_SESSIONS には、LogMiner 永続マイニング・セッション（ディスクにバックアップが取られた LogMiner セッション）の最大数を指定します。すべてのセッションが、スタンドアロン・インスタンスによって生成された REDO ログをマイニングしているときに、これらのセッションは同時にアクティブの状態になります。これによって、LogMiner が使用できるように、 $(2 \times \text{LOGMNR_MAX_PERSISTENT_SESSIONS})$ MB の連続するメモリーが SGA に事前に割り当てられます。

MAX_COMMIT_PROPAGATION_DELAY

パラメータ・タイプ	整数
デフォルト値	700
パラメータ・クラス	静的
値の範囲	0 ～ 90000
Oracle9i Real Application Clusters	すべてのインスタンスに、このパラメータを設定する必要がある。また、複数インスタンスには、同じ値を指定する必要がある。

注意： このパラメータを変更するのは、問合せの実行時に、データベースの最新の状態を見る必要がある場合のみにしてください。

MAX_COMMIT_PROPAGATION_DELAY は、Oracle9i Real Application Clusters のパラメータです。この初期化パラメータは、クラスタ・データベースでの特定の状況下以外は変更しないでください。

MAX_DISPATCHERS

パラメータ・タイプ	整数
デフォルト値	5
パラメータ・クラス	静的
値の範囲	5 または構成されるディスパッチャの数のいずれか大きい方

MAX_DISPATCHERS には、同時に実行できるディスパッチャ・プロセスの最大数を指定します。デフォルト値は、システムにディスパッチャが構成された場合のみ適用されます。

MAX_DUMP_FILE_SIZE

パラメータ・タイプ	文字列
構文	MAX_DUMP_FILE_SIZE = {integer [K M] UNLIMITED}
デフォルト値	UNLIMITED
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM
値の範囲	0 ～無制限または UNLIMITED

MAX_DUMP_FILE_SIZE には、トレース・ファイル（アラート・ファイルを除く）の最大サイズを指定します。トレース・ファイルが使用する領域が大きくなりすぎるおそれがある場合は、この値を変更してください。

MAX_ENABLED_ROLES

パラメータ・タイプ	整数
デフォルト値	20
パラメータ・クラス	静的
値の範囲	0 ～ 148

MAX_ENABLED_ROLES には、他のロールに含まれるロールを含め、ユーザーが使用可能にできるデータベース・ロールの最大数を指定します。

MTS Parameters

9-104 ページの「[共有サーバーのパラメータ](#)」を参照してください。

MAX_SHARED_SERVERS

パラメータ・タイプ	整数
デフォルト値	SHARED_SERVERS から導出される (20 または 2 × SHARED_SERVERS)
パラメータ・クラス	静的
値の範囲	オペレーティング・システム依存

MAX_SHARED_SERVERS には、同時に実行できる共有サーバー・プロセスの最大数を指定します。ご使用のシステムで人為的なデッドロックが頻繁に発生する場合、MAX_SHARED_SERVERS の値を増やす必要があります。

NLS_CALENDAR

パラメータ・タイプ	文字列
構文	NLS_CALENDAR = "calendar_system"
デフォルト値	なし
パラメータ・クラス	動的: ALTER SESSION
値の範囲	有効な暦の書式名

NLS_CALENDAR には、Oracle が使用する暦法を指定します。このパラメータは、次の値のうち 1 つをとることができます。

- Arabic Hijrah（イスラム紀元）
- English Hijrah（英語版イスラム紀元）
- Gregorian（グレゴリオ暦）
- Japanese Imperial（日本の元号制）
- Persian（ペルシャ暦）
- ROC Official（台湾の暦）
- Thai Buddha（タイ仏教暦）

NLS_COMP

パラメータ・タイプ	文字列
構文	NLS_COMP = {binary ansi}
デフォルト値	binary
パラメータ・クラス	動的: ALTER SESSION

通常、問合せの WHERE 句内での比較は、NLSSORT 機能を指定しないかぎり、バイナリです。このパラメータを ansi に設定すると、問合せの WHERE 句内の比較で NLS_SORT パラメータで指定された言語ソートを使用する必要があることが示されます。言語ソートする列で索引を定義する必要があります。

NLS_CURRENCY

パラメータ・タイプ	文字列
構文	NLS_CURRENCY = <i>currency_symbol</i>
デフォルト値	NLS_TERRITORY から導出される。
パラメータ・クラス	動的: ALTER SESSION
値の範囲	最大値は 10 バイトまでの有効な文字列 (NULL を含まない)

NLS_CURRENCY には、L 数値書式要素について各国通貨記号として使用する文字列を指定します。このパラメータのデフォルト値は、NLS_TERRITORY によって決まります。

NLS_DATE_FORMAT

パラメータ・タイプ	文字列
構文	NLS_DATE_FORMAT = " <i>format</i> "
デフォルト値	NLS_TERRITORY から導出される。
パラメータ・クラス	動的: ALTER SESSION
値の範囲	固定長を超えない、有効な日付書式マスク

NLS_DATE_FORMAT には、TO_CHAR ファンクションおよび TO_DATE ファンクションで使用するデフォルトの日付書式を指定します。このパラメータのデフォルト値は、NLS_TERRITORY によって決まります。

NLS_DATE_LANGUAGE

パラメータ・タイプ	文字列
構文	NLS_DATE_LANGUAGE = <i>language</i>
デフォルト値	NLS_LANGUAGE から導出される。
パラメータ・クラス	動的: ALTER SESSION
値の範囲	有効な NLS_LANGUAGE の値

NLS_DATE_LANGUAGE には、TO_DATE ファンクションおよび TO_CHAR ファンクションによって戻される曜日名、月名および日付の略称 (a.m.、p.m.、AD、BC) に使用する言語を指定します。

NLS_DUAL_CURRENCY

パラメータ・タイプ	文字列
構文	NLS_DUAL_CURRENCY = <i>currency_symbol</i>
デフォルト値	NLS_TERRITORY から導出される。
パラメータ・クラス	動的: ALTER SESSION
値の範囲	最大 10 文字までの有効な書式名

NLS_DUAL_CURRENCY には、その地域の第二通貨記号（Euro など）を指定します。デフォルトは、現在の言語環境地域での第二通貨記号です。

NLS_ISO_CURRENCY

パラメータ・タイプ	文字列
構文	NLS_ISO_CURRENCY = <i>territory</i>
デフォルト値	NLS_TERRITORY から導出される。
パラメータ・クラス	動的: ALTER SESSION
値の範囲	有効な NLS_TERRITORY の値

NLS_ISO_CURRENCY には、C 数値書式要素について国際通貨記号として使用する文字列を指定します。

NLS_LANGUAGE

パラメータ・タイプ	文字列
構文	NLS_LANGUAGE = <i>language</i>
デフォルト値	オペレーティング・システム依存。環境変数 NLS_LANG から導出される。
パラメータ・クラス	動的: ALTER SESSION
値の範囲	有効な言語名

NLS_LANGUAGE には、データベースのデフォルト言語を指定します。この言語は、メッセージ、曜日名、月名、および AD、BC、a.m.、p.m. に対する記号に使用されます。また、デフォルトのソート・メカニズムにも使用されます。このパラメータによって、パラメータ NLS_DATE_LANGUAGE および NLS_SORT のデフォルト値が決まります。

NLS_LENGTH_SEMANTICS

パラメータ・タイプ	文字列
構文	NLS_LENGTH_SEMANTICS = <i>string</i> 例 :NLS_LENGTH_SEMANTICS = 'CHAR'
デフォルト値	BYTE
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM
値の範囲	BYTE CHAR

NLS_LENGTH_SEMANTICS によって、バイト長または文字長のセマンティクスを使用して、CHAR 列および VARCHAR2 列を作成できます。既存の列には影響しません。

NLS_NCHAR_CONV_EXCP

パラメータ・タイプ	文字列
構文	NLS_NCHAR_CONV_EXCP = {TRUE FALSE}
デフォルト値	FALSE
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM

NLS_NCHAR_CONV_EXCP は、キャラクタ・タイプの暗黙的または明示的な変換中に発生するデータ損害によって、エラーが報告されるかどうかを決定します。

NLS_NUMERIC_CHARACTERS

パラメータ・タイプ	文字列
構文	NLS_NUMERIC_CHARACTERS = "decimal_character_group_separator"
デフォルト値	NLS_TERRITORY から導出される。
パラメータ・クラス	動的: ALTER SESSION

NLS_NUMERIC_CHARACTERS には、桁区切りおよび小数点として使用する文字を指定します。このパラメータは、NLS_TERRITORY で暗黙的に定義された文字を上書きします。桁区切りは、整数グループ（千、100 万、10 億など）を区切ります。小数点は、数値の整数部分と小数部分を区切ります。

NLS_SORT

パラメータ・タイプ	文字列
構文	NLS_SORT = {BINARY <i>linguistic_definition</i> }
デフォルト値	NLS_LANGUAGE から導出される。
パラメータ・クラス	動的:ALTER SESSION
値の範囲	BINARY または有効な言語定義名

NLS_SORT には、ORDER BY 問合せの照合順番を指定します。

NLS_TERRITORY

パラメータ・タイプ	文字列
構文	NLS_TERRITORY = <i>territory</i>
デフォルト値	オペレーティング・システム依存
パラメータ・クラス	動的:ALTER SESSION
値の範囲	有効な地域名

NLS_TERRITORY には、日と週の順序付けについて地域別規則に従う場合のその地域の名前を指定します。

NLS_TIMESTAMP_FORMAT

パラメータ・タイプ	文字列
構文	NLS_TIMESTAMP_FORMAT = " <i>format</i> "
デフォルト値	NLS_TERRITORY から導出される。
パラメータ・クラス	動的:ALTER SESSION
値の範囲	有効な日時書式マスク

NLS_TIMESTAMP_FORMAT には、TO_CHAR ファンクションおよび TO_TIMESTAMP ファンクションで使用するデフォルトのタイムスタンプ書式を定義します。

NLS_TIMESTAMP_TZ_FORMAT

パラメータ・タイプ	文字列
構文	NLS_TIMESTAMP_TZ_FORMAT = " <i>format</i> "
デフォルト値	NLS_TERRITORY から導出される。
パラメータ・クラス	動的: ALTER SESSION
値の範囲	有効な日時書式マスク

NLS_TIMESTAMP_TZ_FORMAT には、TO_CHAR ファンクションおよび TO_TIMESTAMP_TZ ファンクションで使用するデフォルトのタイムスタンプを定義します。このタイムスタンプは、タイム・ゾーン書式を使用します。

O7_DICTIONARY_ACCESSIBILITY

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	静的
値の範囲	true false

O7_DICTIONARY_ACCESSIBILITY は、Oracle7 から Oracle Security Server への移行に使用されます。このパラメータは、SYSTEM 権限の制限を制御します。このパラメータに true を設定すると、SYSTEM 権限による SYS スキーマ内のオブジェクトへのアクセスが許可されます (Oracle7 の動作)。false に設定すると、すべてのスキーマ内のオブジェクトへのアクセスが許可される SYSTEM 権限では、SYS スキーマ内のオブジェクトへアクセスできなくなります。

OBJECT_CACHE_MAX_SIZE_PERCENT

パラメータ・タイプ	整数
デフォルト値	10
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM ... DEFERRED
値の範囲	0 以上。上限は、オペレーティング・システム依存の最大値。

オブジェクト・キャッシュは、クライアント上のメモリー・ブロックで、アプリケーションがすべてのオブジェクトを格納でき、サーバーヘラウンド・トリップせずにオブジェクト間でナビゲートできるようにします。OBJECT_CACHE_MAX_SIZE_PERCENT には、セッション・オブジェクト・キャッシュが、最適なキャッシュ・サイズを、最大で何パーセント超過できるかを指定します。最大サイズは、最適サイズに、このパーセントと最適サイズの積を足したサイズです。キャッシュ・サイズがこの最大サイズを超えると、システムは、キャッシュを最適サイズに縮小しようとします。

OBJECT_CACHE_OPTIMAL_SIZE

パラメータ・タイプ	整数
デフォルト値	102400 (100KB)
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM ... DEFERRED
値の範囲	10KB 以上。上限は、オペレーティング・システム依存の最大値。

オブジェクト・キャッシュは、クライアント上のメモリー・ブロックで、アプリケーションがすべてのオブジェクトを格納でき、サーバーヘラウンド・トリップせずにオブジェクト間でナビゲートできるようにします。OBJECT_CACHE_OPTIMAL_SIZE には、キャッシュ・サイズが最大サイズを超えた場合に、セッション・オブジェクト・キャッシュが縮小されるサイズ (バイト単位) を指定します。

OPEN_CURSORS

パラメータ・タイプ	整数
デフォルト値	50
パラメータ・クラス	静的
値の範囲	1 ~ 4294967295 (4GB-1)

OPEN_CURSORS には、1 つのセッションで同時にオープンできるカーソル (プライベート SQL 領域へのハンドル) の最大数を指定します。このパラメータを使用して、1 つのセッションでカーソルをオープンしすぎないようにできます。このパラメータには、PL/SQL カーソル・キャッシュのサイズも制限します。PL/SQL カーソル・キャッシュは、ユーザーによる文の解析が繰り返されることを防ぐため、PL/SQL が使用します。

OPEN_LINKS

パラメータ・タイプ	整数
デフォルト値	4
パラメータ・クラス	静的
値の範囲	0 ～ 255

OPEN_LINKS には、1 つのセッションでのリモート・データベースに対する同時オープン接続の最大数を指定します。これらの接続には、外部プロシージャおよびカートリッジの他、データベース・リンクが含まれます。これらは、それぞれ個別のプロセスを使用します。

OPEN_LINKS_PER_INSTANCE

パラメータ・タイプ	整数
デフォルト値	4
パラメータ・クラス	静的
値の範囲	0 ～ 4294967295 (4GB-1)
Oracle9i Real Application Clusters	複数インスタンスには、異なる値を指定可能。

OPEN_LINKS_PER_INSTANCE には、データベース・インスタンスごとのグローバルに移行可能なオープン接続の最大数を指定します。XA トランザクションでは、移行可能なオープン接続が使用されるので、トランザクションがコミットされると、接続がキャッシュされます。この接続を別のトランザクションでも使用できます。ただし、その接続を作成したユーザーがそのトランザクションを所有している場合に限定されます。

OPTIMIZER_FEATURES_ENABLE

パラメータ・タイプ	文字列
構文	OPTIMIZER_FEATURES_ENABLE = <i>release_number</i>
デフォルト値	9.0.0
パラメータ・クラス	静的
値の範囲	8.0.0、8.0.3、8.0.4、8.0.5、8.0.6、8.1.0、8.1.3、8.1.4、8.1.5、8.1.6、8.1.7、9.0.1

OPTIMIZER_FEATURES_ENABLE によって、リリース番号に基づく Oracle オプティマイザの動作を変更することができます。たとえば、このパラメータを 8.0.5 に設定すると、8.0.5 オプティマイザ動作を保ったまま、データベースを 8.0.5 から 8.1.5 にアップグレードできます。また、このパラメータを 8.1.5 に設定すると、リリース 8.1.5 で導入された新しい拡張機能を使用することができます。

OPTIMIZER_INDEX_CACHING

パラメータ・タイプ	整数
デフォルト値	0
パラメータ・クラス	動的: ALTER SESSION
値の範囲	0 ～ 100

OPTIMIZER_INDEX_CACHING を設定すると、ネストしたループ結合および IN リスト・イテレータを支援するように、コストベースの最適化動作を調整できます。

OPTIMIZER_INDEX_COST_ADJ

パラメータ・タイプ	整数
デフォルト値	100
パラメータ・クラス	動的: ALTER SESSION
値の範囲	1 ～ 10000

OPTIMIZER_INDEX_COST_ADJ を指定すると、オプティマイザがアクセス・パスを決定するときに、索引の使用を選択する傾向の強さを調整できます。つまり、オプティマイザがフル・テーブル・スキャンを経由して索引アクセス・パスを選択するようになります。

OPTIMIZER_MAX_PERMUTATIONS

パラメータ・タイプ	整数
デフォルト値	80000
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM
値の範囲	4 ～ 2 ³² (～ 4300000000)

OPTIMIZER_MAX_PERMUTATIONS は、オブティマイザが問合せで結合を考慮する表の並べ替えの数を制限します。この制限により、問合せに必要な解析時間を許容できる制限内にすることができます。ただし、この方法には、オブティマイザが別の良い方法を見落とすというデメリットがあります。

OPTIMIZER_MODE

パラメータ・タイプ	文字列
構文	OPTIMIZER_MODE = {first_rows_[1 10 100 1000] first_rows all_rows choose rule}
デフォルト値	choose
パラメータ・クラス	動的: ALTER SESSION

OPTIMIZER_MODE は、インスタンスの最適化方法を選択するためのデフォルトの動作を確立します。

ORACLE_TRACE_COLLECTION_NAME

パラメータ・タイプ	文字列
構文	ORACLE_TRACE_COLLECTION_NAME = <i>collection_name</i>
デフォルト値	なし
パラメータ・クラス	静的
値の範囲	最大 16 文字までの有効なコレクション識別名 (ファイル名が 8 文字までのプラットフォームを除く)

コレクションは、インストルメント製品の実行中に発生したイベントを収集したデータです。ORACLE_TRACE_COLLECTION_NAME には、このインスタンスに対する Oracle Trace のコレクション識別名を指定します。このパラメータは、出力ファイル名 (コレクション定義ファイル .cdf およびデータ・コレクション・ファイル .dat) でも使用されます。ORACLE_TRACE_ENABLE が true に設定されている場合、この値に設定した NULL 以外の文字列が、この値に再度 NULL を設定するまで、デフォルトの Oracle Trace のコレクション機能を起動します。

ORACLE_TRACE_COLLECTION_PATH

パラメータ・タイプ	文字列
構文	ORACLE_TRACE_COLLECTION_PATH = <i>pathname</i>
デフォルト値	オペレーティング・システム依存
パラメータ・クラス	静的
値の範囲	ディレクトリのフルパス名

ORACLE_TRACE_COLLECTION_PATH には、Oracle Trace のコレクション定義ファイル (.cdf) およびデータ・コレクション・ファイル (.dat) を格納するディレクトリ・パス名を指定します。デフォルトの場合、Oracle Trace の .cdf および .dat ファイルは、*ORACLE_HOME*/otrace/admin/cdf に格納されます。

ORACLE_TRACE_COLLECTION_SIZE

パラメータ・タイプ	整数
デフォルト値	5242880
パラメータ・クラス	静的
値の範囲	0 ～ 4294967295

ORACLE_TRACE_COLLECTION_SIZE には、Oracle Trace コレクション・ファイルの最大サイズ（バイト単位）を指定します。このコレクション・ファイルが最大サイズに到達すると、収集は禁止されます。値 0 は、ファイルにサイズ制限がないことを示します。

ORACLE_TRACE_ENABLE

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	動的:ALTER SESSION、ALTER SYSTEM
値の範囲	true false

サーバーの Oracle Trace のコレクション機能を使用可能にするには、ORACLE_TRACE_ENABLE を true に設定します。この設定のみでは Oracle Trace のコレクション機能は起動されませんが、そのサーバーで Oracle Trace を使用できるようになります。

ORACLE_TRACE_FACILITY_NAME

パラメータ・タイプ	文字列
構文	ORACLE_TRACE_FACILITY_NAME = {ORACLED ORACLEE ORACLESM ORACLEC}
デフォルト値	ORACLED
パラメータ・クラス	静的

ORACLE_TRACE_FACILITY_NAME には、Oracle Trace が収集するイベント・セットを指定します。このパラメータ値の後に .fdf 拡張子が付いて、Oracle Trace 製品定義ファイルの名前になります。このファイルは、ORACLE_TRACE_FACILITY_PATH パラメータで指定されたディレクトリに格納する必要があります。製品定義ファイルには、Oracle Trace データ・コレクション API を使用する製品のために収集されるすべてのイベントおよびデータ項目の定義情報が格納されています。

ORACLE_TRACE_FACILITY_PATH

パラメータ・タイプ	文字列
構文	ORACLE_TRACE_FACILITY_PATH = <i>pathname</i>
デフォルト値	オペレーティング・システム依存
パラメータ・クラス	静的
値の範囲	ディレクトリのフルパス名

ORACLE_TRACE_FACILITY_PATH には、Oracle Trace の機能定義ファイルが格納されるディレクトリ・パス名を指定します。Solaris の場合、デフォルトのパス名は、ORACLE_HOME/otrace/admin/fdf/ です。NT の場合、デフォルトのパス名は、%OTRACE80%\ADMIN\FDF\ です。

OS_AUTHENT_PREFIX

パラメータ・タイプ	文字列
構文	OS_AUTHENT_PREFIX = <i>authentication_prefix</i>
デフォルト値	OP\$
パラメータ・クラス	静的

OS_AUTHENT_PREFIX には、サーバーに接続しようとするユーザーの認証に使用される接頭辞を指定します。このパラメータの値は、各ユーザーのオペレーティング・システム・アカウント名およびパスワードの先頭に連結されます。接続が要求されると、接頭辞の付いたユーザー名が、データベース内の Oracle ユーザー名と比較されます。

OS_ROLES

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	静的
値の範囲	true false

OS_ROLES は、Oracle またはオペレーティング・システムのどちらが、各ユーザー名のロールを識別および管理するかを判断します。

PARALLEL_ADAPTIVE_MULTI_USER

パラメータ・タイプ	ブール
デフォルト値	PARALLEL_AUTOMATIC_TUNING の値から導出される。
パラメータ・クラス	動的:ALTER SYSTEM
値の範囲	true false

PARALLEL_ADAPTIVE_MULTI_USER を true に設定すると、パラレル実行を使用するマルチユーザー環境での、パフォーマンス向上を目的とした適応アルゴリズムが使用可能になります。問合せ起動時のシステム・ロードに基づき、アルゴリズムでは、要求された並列度が自動的に減少します。有効な並列度は、デフォルトの並列度、または表またはヒントから得られる並列度を減少要因で割った値に基づきます。

PARALLEL_AUTOMATIC_TUNING

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	静的
値の範囲	true false

注意： このパラメータは、Oracle9i Real Application Clusters 環境および
排他モードでのパラレル実行に適用されます。

PARALLEL_AUTOMATIC_TUNING を true に設定した場合、パラレル実行を制御するパラ
メータのデフォルト値が判断されます。このパラメータの設定に加えて、システムのター
ゲット表に PARALLEL 句を指定する必要があります。その後、すべての連続するパラレル操
作が自動的にチューニングされます。

PARALLEL_BROADCAST_ENABLED

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	動的: ALTER SESSION
値の範囲	true false

注意： このパラメータは、Oracle9i Real Application Clusters ではなくパ
ラレル実行に適用されます。

PARALLEL_BROADCAST_ENABLED を使用すると、大規模な結合結果セットが小規模な結果
セット（行数ではなく、バイト単位で測定したサイズ）に結合される、ハッシュ結合および
マージ結合のパフォーマンスが向上します。

PARALLEL_EXECUTION_MESSAGE_SIZE

パラメータ・タイプ	整数
デフォルト値	オペレーティング・システム依存
パラメータ・クラス	静的
値の範囲	2148 ～ 65535 (64 KB-1)
Oracle9i Real Application Clusters	複数インスタンスには、同じ値を指定する必要がある。

PARALLEL_EXECUTION_MESSAGE_SIZE には、パラレル実行（以前は、パラレル問合せ、
PDML、パラレル・リカバリ、レプリケーションに適用）用のメッセージのサイズを指定し
ます。

PARALLEL_INSTANCE_GROUP

パラメータ・タイプ	文字列
構文	PARALLEL_INSTANCE_GROUP = group_name
デフォルト値	現在アクティブであるすべてのインスタンスで構成されるグループ
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM
値の範囲	すべてのアクティブ・インスタンスの INSTANCE_GROUPS パラメータに指定されたすべてのグループ名
Oracle9i Real Application Clusters	異なるインスタンスには異なる値を指定可能

PARALLEL_INSTANCE_GROUP は、Oracle9i Real Application Clusters のパラメータで、パ
ラレル・モードでのみ使用できます。INSTANCE_GROUP パラメータとともに使用すると、
インスタンスの制限数にパラレル問合せを制限できます。

PARALLEL_MAX_SERVERS

パラメータ・タイプ	整数
デフォルト値	CPU_COUNT、PARALLEL_AUTOMATIC_TUNING および PARALLEL_ADAPTIVE_MULTI_USER の値から導出される。
パラメータ・クラス	静的
値の範囲	0 ～ 3599
Oracle9i Real Application Clusters	複数インスタンスには、同じ値を指定する必要がある。

注意： このパラメータは、Oracle9i Real Application Clusters 環境および
排他モードでのパラレル実行に適用されます。

PARALLEL_MAX_SERVERS には、インスタンスに関するパラレル実行プロセスおよびパ
ラレル・リカバリ・プロセスの最大数を指定します。増加が要求されると、Oracle は、プロセス
数を、インスタンスの始動時に作成された数からこの指定された数以内の範囲で増やしま
す。

PARALLEL_MIN_PERCENT

パラメータ・タイプ	整数
デフォルト値	0
パラメータ・クラス	動的: ALTER SESSION
値の範囲	0 ～ 100
Oracle9i Real Application Clusters	複数インスタンスには、異なる値を指定可能。

PARALLEL_MIN_PERCENT は、PARALLEL_MAX_SERVERS および PARALLEL_MIN_SERVERS と組み合わせて処理されます。パラレル実行に必要な、(PARALLEL_MAX_SERVERS の値の) パラレル実行プロセスの最小の割合を指定できます。このパラメータを設定することで、適切なリソースが使用可能でないかぎり、パラレル操作が順次実行されないようにします。デフォルト値 0 は、プロセスの最小の割合が設定されていないことを意味します。

PARALLEL_MIN_SERVERS

パラメータ・タイプ	整数
デフォルト値	0
パラメータ・クラス	静的
値の範囲	0 ～ PARALLEL_MAX_SERVERS の値
Oracle9i Real Application Clusters	複数インスタンスには、異なる値を指定可能。

注意： このパラメータは、Oracle9i Real Application Clusters 環境および排他モードでのパラレル実行に適用されます。

PARALLEL_MIN_SERVERS には、インスタンスに対するパラレル実行プロセスの最小数を指定します。この値は、インスタンスの起動時に作成されるパラレル実行プロセスの数です。

PARALLEL_THREADS_PER_CPU

パラメータ・タイプ	整数
デフォルト値	オペレーティング・システム依存。通常は 2。
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	0（ゼロ）以外の任意の数値

注意： このパラメータは、Oracle9i Real Application Clusters 環境および
排他モードでのパラレル実行に適用されます。

PARALLEL_THREADS_PER_CPU には、インスタンスのデフォルト並列度を指定し、パラレル適応およびロード・バランス・アルゴリズムを判断します。このパラメータには、パラレル実行中に CPU が処理できるパラレル実行プロセスまたはスレッドの数を記述します。

PARTITION_VIEW_ENABLED

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	動的: ALTER SESSION
値の範囲	true false

注意： パーティション・ビューではなく、パーティション表（Oracle8 以上で
使用可能）を使用することをお勧めします。パーティション・ビューは、
下位互換性のためにサポートされています。

PARTITION_VIEW_ENABLED には、オブティマイザがパーティション・ビューを使用する
かどうかを指定します。このパラメータを true に設定すると、オブティマイザはパーティ
ション・ビューでの不要な表アクセスを排除（つまりスキップ）し、基礎となる表について
の統計からパーティション・ビューについての統計を算出する方法を変更します。

PGA_AGGREGATE_TARGET

パラメータ・タイプ	大整数
構文	PGA_AGGREGATE_TARGET = integer [K M G]
デフォルト値	0（デフォルトでは、自動メモリー管理はOFF）
パラメータ・クラス	動的:ALTER SYSTEM
値の範囲	10MB ～ 4000GB

PGA_AGGREGATE_TARGET には、インスタンスに接続されたすべてのサーバー・プロセスが使用できるターゲット集計 PGA メモリーを指定します。メモリー集中型の SQL 演算子（ソート、グループ化、ハッシュ結合、ビットマップ・マージ、ビットマップ作成など）が使用する SQL 作業領域の自動サイズ指定を有効にするには、このパラメータを設定する必要があります。

PLSQL_COMPILER_FLAGS

パラメータ・タイプ	文字列
構文	PLSQL_COMPILER_FLAGS = { [DEBUG NON_DEBUG] [INTERPRETED NORMAL] }
デフォルト値	INTERPRETED, NON_DEBUG
パラメータ・クラス	動的:ALTER SESSION、ALTER SYSTEM

PLSQL_COMPILER_FLAGS は、PL/SQL コンパイラが使用するパラメータです。このパラメータには、カンマで区切った文字列のリストとして、コンパイラ・フラグを指定します。

PLSQL_NATIVE_C_COMPILER

パラメータ・タイプ	文字列
構文	PLSQL_NATIVE_C_COMPILER = pathname
デフォルト値	なし
パラメータ・クラス	動的:ALTER SYSTEM
値の範囲	有効なパス名

PLSQL_NATIVE_C_COMPILER には、生成された C ファイルをオブジェクト・ファイルにコンパイルするために使用する C コンパイラのフルパス名を指定します。

PLSQL_NATIVE_LIBRARY_DIR

パラメータ・タイプ	文字列
構文	PLSQL_NATIVE_LIBRARY_DIR = <i>directory</i>
デフォルト値	なし
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	有効なディレクトリ・パス

PLSQL_NATIVE_LIBRARY_DIR は、PL/SQL コンパイラが使用するパラメータです。このパラメータには、システム固有のコンパイラが生成した共有オブジェクトが格納されるディレクトリ名を指定します。

PLSQL_NATIVE_LIBRARY_SUBDIR_COUNT

パラメータ・タイプ	整数
デフォルト値	0
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	0 ～ 2 ³² -1 (32 ビットで表される最大値)

PLSQL_NATIVE_LIBRARY_SUBDIR_COUNT には、PLSQL_NATIVE_LIBRARY_DIR で指定したディレクトリに、データベース管理者が作成するサブディレクトリの数を指定します。

PLSQL_NATIVE_LINKER

パラメータ・タイプ	文字列
構文	PLSQL_NATIVE_LINKER = <i>pathname</i>
デフォルト値	なし
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	有効なパス名

PLSQL_NATIVE_LINKER には、リンカー（UNIX での ld、またはオブジェクト・ファイルを共有オブジェクトまたは DLL にリンクするために使用される GNU ld など）のフルパス名を指定します。

PLSQL_NATIVE_MAKE_FILE_NAME

パラメータ・タイプ	文字列
構文	PLSQL_NATIVE_MAKE_FILE_NAME = <i>pathname</i>
デフォルト値	なし
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	有効なパス名

PLSQL_NATIVE_MAKE_FILE_NAME には、Make ファイルのフルパス名を指定します。
make ユーティリティ (PLSQL_NATIVE_MAKE_UTILITY で指定) では、Make ファイルを使用して、共有オブジェクトまたは DLL を作成します。

PLSQL_NATIVE_MAKE_UTILITY

パラメータ・タイプ	文字列
構文	PLSQL_NATIVE_MAKE_UTILITY = <i>pathname</i>
デフォルト値	なし
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	有効なパス名

PLSQL_NATIVE_MAKE_UTILITY には、make ユーティリティ (UNIX の make または gmake (GNU make)) のフルパス名を指定します。生成された C ソースから、共有オブジェクトまたは DLL を生成するには、make ユーティリティが必要です。

PLSQL_V2_COMPATIBILITY

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM
値の範囲	true false

PL/SQL バージョン 2 により、バージョン 8 で禁止されている異常動作が使用できるようになります。下位互換性のためにその動作を保持する場合は、PLSQL_V2_COMPATIBILITY を true に設定します。false に設定すると、PL/SQL バージョン 8 の動作は実行されますが、バージョン 2 の動作は実行されません。

PRE_PAGE_SGA

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	静的
値の範囲	true false

PRE_PAGE_SGA は、インスタンス起動時に SGA 全体がメモリーに読み込まれるかどうかを判断します。オペレーティング・システムのページ表エントリは、SGA のページごとに事前作成されます。この設定によって、インスタンスの起動に必要な時間が長くなる場合がありますが、Oracle のパフォーマンスが最高になるまでの時間が短縮されます。

PROCESSES

パラメータ・タイプ	整数
デフォルト値	PARALLEL_MAX_SERVERS から導出される。
パラメータ・クラス	静的
値の範囲	6 以上。上限は、オペレーティング・システム依存。
Oracle9i Real Application Clusters	複数インスタンスには、異なる値を指定可能。

PROCESSES には、Oracle に同時に接続できるオペレーティング・システムのユーザー・プロセスの最大数を指定します。この値は、ロック、ジョブ・キュー・プロセス、パラレル実行プロセスなどのすべてのバックグラウンド・プロセスを考慮して設定する必要があります。

QUERY_REWRITE_ENABLED

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	動的:ALTER SESSION、ALTER SYSTEM
値の範囲	true false
Oracle9i Real Application Clusters	複数インスタンスには、異なる値を指定可能。

QUERY_REWRITE_ENABLED を使用すると、データベースへのクエリー・リライトを、グローバルに使用可能または使用禁止にすることができます。

QUERY_REWRITE_INTEGRITY

パラメータ・タイプ	文字列
構文	QUERY_REWRITE_INTEGRITY = {stale_tolerated trusted enforced}
デフォルト値	enforced
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM
Oracle9i Real Application Clusters	複数インスタンスには、異なる値を指定可能。

QUERY_REWRITE_INTEGRITY は、クエリー・リライトを行う程度を判断します。最も安全なレベルでは、適用されていないリレーションシップに依存するクエリー・リライト変換は使用されません。

RDBMS_SERVER_DN

パラメータ・タイプ	X.509 識別名
デフォルト値	なし
パラメータ・クラス	静的
値の範囲	X.509 識別名形式の値すべて

RDBMS_SERVER_DN には、Oracle サーバーの識別名 (DN) を指定します。エンタープライズ・ディクショナリ・サービスからエンタープライズ・ロールを取り出すために使用されます。

READ_ONLY_OPEN_DELAYED

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	静的
値の範囲	true false

READ_ONLY_OPEN_DELAYED は、読取り専用表領域内のデータ・ファイルがアクセスされる場合を判断します。

RECOVERY_PARALLELISM

パラメータ・タイプ	整数
デフォルト値	オペレーティング・システム依存
パラメータ・クラス	静的
値の範囲	オペレーティング・システム依存。 上限は、PARALLEL_MAX_SERVERS。

RECOVERY_PARALLELISM には、インスタンス・リカバリまたはクラッシュ・リカバリに
与するプロセスの数を指定します。0 または 1 は、リカバリが 1 つのプロセスによって順次
実行されることを示します。

REMOTE_ARCHIVE_ENABLE

パラメータ・タイプ	ブール
デフォルト値	true
パラメータ・クラス	静的
値の範囲	true false

REMOTE_ARCHIVE_ENABLE は、REDO ログをリモートの宛先にアーカイブできるかどうか
を制御します。デフォルトでは、リモートの宛先へのアーカイブは可能です。

REMOTE_DEPENDENCIES_MODE

パラメータ・タイプ	文字列
構文	REMOTE_DEPENDENCIES_MODE = {TIMESTAMP SIGNATURE}
デフォルト値	TIMESTAMP
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM

REMOTE_DEPENDENCIES_MODE には、リモートの PL/SQL ストアド・プロシージャに対す
る依存性の処理方法を指定します。

REMOTE_LISTENER

パラメータ・タイプ	文字列
構文	REMOTE_LISTENER = <i>network_name</i>
デフォルト値	なし
パラメータ・クラス	静的

REMOTE_LISTENER には、Oracle Net リモート・リスナー（このインスタンスと異なるマシン上で実行中のリスナー）のアドレスまたはアドレス・リストを解決するネットワーク名を指定します。アドレスまたはアドレス・リストは、TNSNAMES.ORA ファイルまたはご使用のシステム用に構成されている他のアドレス・リポジトリで指定されます。

REMOTE_LOGIN_PASSWORDFILE

パラメータ・タイプ	文字列
構文	REMOTE_LOGIN_PASSWORDFILE= {NONE SHARED EXCLUSIVE}
デフォルト値	NONE
パラメータ・クラス	静的
Oracle9i Real Application Clusters	複数インスタンスには、同じ値を指定する必要がある。

REMOTE_LOGIN_PASSWORDFILE には、Oracle がパスワード・ファイルを確認するかどうか、およびそのパスワード・ファイルを使用できるデータベースの数を指定します。

REMOTE_OS_AUTHENT

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	静的
値の範囲	true false

REMOTE_OS_AUTHENT には、OS_AUTHENT_PREFIX パラメータ値でリモート・クライアントが認証されるかどうかを指定します。

REMOTE_OS_ROLES

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	静的
値の範囲	true false

REMOTE_OS_ROLES には、オペレーティング・システム・ロールがリモート・クライアントに与えられるかどうかを指定します。デフォルト値 false では、Oracle によって、リモート・クライアントに対するロールが識別および管理されます。

REPLICATION_DEPENDENCY_TRACKING

パラメータ・タイプ	ブール
デフォルト値	true
パラメータ・クラス	静的
値の範囲	true false

REPLICATION_DEPENDENCY_TRACKING により、データベースへの読取り / 書込み操作に対する依存性追跡を使用可能または使用禁止にできます。依存性追跡は、レプリケート環境でパラレルに変更を伝播するときに重要になります。

RESOURCE_LIMIT

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	動的:ALTER SYSTEM
値の範囲	true false

RESOURCE_LIMIT は、リソース制限がデータベース・プロファイルで行われるかどうかを判断します。

RESOURCE_MANAGER_PLAN

パラメータ・タイプ	文字列
構文	RESOURCE_MANAGER_PLAN = <i>plan_name</i>
デフォルト値	なし
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	有効な任意の文字列

RESOURCE_MANAGER_PLAN には、インスタンスに使用するトップレベルのリソース・プランを指定します。リソース・マネージャは、このトップレベルのプランをすべての子孫（サブプラン、ディレクティブおよびコンシューマ・グループ）にロードします。このパラメータを指定しない場合、リソース・マネージャはデフォルトのオフです。

ROLLBACK_SEGMENTS

パラメータ・タイプ	文字列
構文	ROLLBACK_SEGMENTS = (segment_name [, segment_name] ...)
デフォルト値	このパラメータを指定しない場合、インスタンスでは、パブリック・ロールバック・セグメントがデフォルトで使用される。
パラメータ・クラス	静的
値の範囲	DBA_ROLLBACK_SEGS に示されたすべてのロールバック・セグメント名 (SYSTEM を除く)
Oracle9i Real Application Clusters	複数インスタンスには、異なる値を指定可能

ROLLBACK_SEGMENTS には、インスタンスに割り当てる 1 つ以上のロールバック・セグメントの名前を指定します。このパラメータが設定されると、ロールバック・セグメント数がインスタンスに必要な最小数 (TRANSACTIONS / TRANSACTIONS_PER_ROLLBACK_SEGMENT から算出) を超えていても、インスタンスは、このパラメータで指定されたロールバック・セグメントをすべて取得します。

ROW_LOCKING

パラメータ・タイプ	文字列
構文	ROW_LOCKING = {ALWAYS DEFAULT INTENT}
デフォルト値	ALWAYS
パラメータ・クラス	静的
Oracle9i Real Application Clusters	すべてのインスタンスに、このパラメータを設定する必要がある。また、複数インスタンスには、同じ値を指定する必要がある。

ROW_LOCKING には、UPDATE 操作中に行ロックを取得するかどうかを指定します。

SERIAL_REUSE

パラメータ・タイプ	文字列
構文	SERIAL_REUSE = {DISABLE SELECT DML PLSQL ALL}
デフォルト値	DISABLE
パラメータ・クラス	静的

SERIAL_REUSE には、メモリー・シリアル再利用機能を使用するカーソルの種類を指定します。この機能は、SGA 内のプライベート・カーソル・メモリーを割り当て、同じカーソルを実行するセッションによってそのメモリーが（同時ではなく、シリアルに）再利用できるようになります。

SERVICE_NAMES

パラメータ・タイプ	文字列
構文	SERVICE_NAMES = db_service_name [, db_service_name [...]]
デフォルト値	DB_NAME.DB_DOMAIN（定義されている場合）
パラメータ・クラス	動的:ALTER SYSTEM
値の範囲	ASCII 文字列、またはカンマで区切られた文字列名のリスト

SERVICE_NAMES には、このインスタンスに接続する 1 つ以上のデータベース・サービスの名前を指定します。同じデータベースの異なる使用を区別するために、複数サービス名を指定できます。

SESSION_CACHED_CURSORS

パラメータ・タイプ	整数
デフォルト値	0
パラメータ・クラス	動的: ALTER SESSION
値の範囲	0 以上。上限は、オペレーティング・システムによって異なる。
Oracle9i Real Application Clusters	複数インスタンスには、異なる値を指定可能。

SESSION_CACHED_CURSORS を使用すると、キャッシュするセッション・カーソル数を指定できます。同じ SQL 文に対して解析コールを繰り返し行くと、その文のセッション・カーソルをセッション・カーソル・キャッシュに移動できるようになります。後続の解析コールでは、カーソルはキャッシュ内にあるので、カーソルを再オープンする必要がなくなります。新規のエントリのために領域を空ける必要がある場合、Oracle は、最近使用されたアルゴリズムを使用して、セッション・カーソル・キャッシュ内のエントリを削除します。

SESSION_MAX_OPEN_FILES

パラメータ・タイプ	整数
デフォルト値	10
パラメータ・クラス	静的
値の範囲	1 ～ 50 またはオペレーティング・システムのレベルで定義された MAX_OPEN_FILES 値 (いずれか小さい方)

SESSION_MAX_OPEN_FILES には、すべてのセッションでオープンできる BFILE の最大数を指定します。ここで設定された数に達すると、その後は DBMS_LOB.FILEOPEN() または OCILobFileOpen() を使用してセッションでさらにファイルをオープンしようとしても失敗します。このパラメータの最大値は、使用しているオペレーティング・システムで定義されている対応パラメータに依存します。

SESSIONS

パラメータ・タイプ	整数
デフォルト値	導出される： (1.1 × PROCESSES) + 5
パラメータ・クラス	静的
値の範囲	1 ～ 2 ³¹

SESSIONS には、システムに作成できるセッションの最大数を指定します。すべてのログインにはセッションが必要なため、このパラメータにはシステムでの同時ユーザーの最大数を実際的な値で指定します。このパラメータには、同時ユーザーの推定最大値に、バックグラウンド・プロセスの数を加えて、さらに再帰セッションの約 10% を加えた値を常に明示的に設定しておく必要があります。

SGA_MAX_SIZE

パラメータ・タイプ	大整数
構文	SGA_MAX_SIZE = integer [K M G]
デフォルト値	起動時の SGA サイズは、SGA 内の個々のプール（バッファ・キャッシュ、共有プール、ラージ・プールなど）のサイズによって異なる。
パラメータ・クラス	静的
値の範囲	0 以上。上限は、オペレーティング・システムによって異なる。

SGA_MAX_SIZE には、インスタンスの存続期間を通しての SGA の最大サイズを指定します。

SHADOW_CORE_DUMP

パラメータ・タイプ	文字列
構文	SHADOW_CORE_DUMP = {PARTIAL FULL}
デフォルト値	PARTIAL
パラメータ・クラス	静的

SHADOW_CORE_DUMP は主に UNIX のパラメータなので、ご使用のプラットフォームでは役に立たない場合があります。このパラメータには、フォアグラウンド（クライアント）・プロセスのコア・ファイルに SGA を含めるかどうかを指定します。

SHARED_MEMORY_ADDRESS

パラメータ・タイプ	整数
デフォルト値	0
パラメータ・クラス	静的

SHARED_MEMORY_ADDRESS および HI_SHARED_MEMORY_ADDRESS には、システム・グローバル領域（SGA）の実行時の開始アドレスを指定します。これらのパラメータは、リンク時の SGA の開始アドレスを指定する多くのプラットフォームでは無視されます。

SHARED_POOL_RESERVED_SIZE

パラメータ・タイプ	大整数
構文	SHARED_POOL_RESERVED_SIZE = integer [K M G]
デフォルト値	SHARED_POOL_SIZE の値 5%
パラメータ・クラス	静的
値の範囲	最小値: SHARED_POOL_RESERVED_MIN_ALLOC の値 最大値: SHARED_POOL_SIZE の値の半分（バイト）

SHARED_POOL_RESERVED_SIZE には、大きな共有プール・メモリー領域の要求が連続したときのために確保する共有プール領域（バイト単位）を指定します。このパラメータを使用すると、プールの断片化が発生したときに、Oracle がカレント要求に応じるために未使用のプール空き領域を探して解放する必要があることによって起こる、共有プールでのパフォーマンスの低下を回避できます。

SHARED_POOL_SIZE

パラメータ・タイプ	大整数
構文	SHARED_POOL_SIZE = integer [K M G]
デフォルト値	64 ビットの場合は 64MB、それ以外の場合は 16MB
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	300KB 以上。上限は、オペレーティング・システムによって異なる。

SHARED_POOL_SIZE には、共有プールのサイズ（バイト単位）を指定します。共有プールには、共有カーソル、ストアド・プロシージャ、制御構造体、およびその他の構造体が含まれます。PARALLEL_AUTOMATIC_TUNING を false に設定している場合、共有プールからパラレル実行メッセージ・バッファが割り当てられます。値を大きくするほど、マルチユーザー・システムでのパフォーマンスが改善されます。値が小さいほど、使用されるメモリは少なくなります。

共有サーバーのパラメータ

Oracle9i から、マルチスレッド・サーバー・アーキテクチャは、**共有サーバー・アーキテクチャ**といえます。

インスタンスを起動すると、Oracle は SHARED_SERVERS および DISPATCHERS 初期化パラメータの値に基づく共有サーバー・アーキテクチャの共有サーバー・プロセスおよびディスパッチャ・プロセスを作成します。ALTER SYSTEM 文で SHARED_SERVERS および DISPATCHERS パラメータを設定し、インスタンスの実行中に次の操作のいずれかを実行できます。

- 共有サーバー・プロセスの最小値を増やして、追加の共有サーバー・プロセスを作成します。
- 現行のコールが処理を終了した後、既存の共有サーバー・プロセスを終了します。
- 特定のプロトコルに対するディスパッチャ・プロセスをより多く作成します。プロトコル全体で、初期化パラメータ MAX_DISPATCHERS によって指定される数まで作成できます。
- 現行のユーザー・プロセスがインスタンスから切断した後、特定のプロトコルに対する既存のディスパッチャ・プロセスを終了します。

SHARED_SERVERS

パラメータ・タイプ	整数
デフォルト値	1（共有サーバー・アーキテクチャを使用する場合） 0（共有サーバー・アーキテクチャを使用しない場合）
パラメータ・クラス	動的：ALTER SYSTEM
値の範囲	オペレーティング・システム依存

SHARED_SERVERS には、インスタンスの起動時に作成するサーバー・プロセスの数を指定します。システム負荷が減少する場合は、このサーバーの最小値が保持されます。そのため、SHARED_SERVERS をシステムの起動時にあまり高く設定しないように注意してください。

SHARED_SERVER_SESSIONS

パラメータ・タイプ	整数
デフォルト値	次の値が導出される :CIRCUITS または SESSIONS-5 (いずれか小さい方)
パラメータ・クラス	静的
値の範囲	0 ～ SESSIONS - 5

SHARED_SERVER_SESSIONS には、ユーザー・セッションにより使用可能な、共有サーバー・アーキテクチャの合計数を指定します。このパラメータを設定すると、専用サーバーのユーザー・セッションを確保することが可能になります。

SORT_AREA_RETAINED_SIZE

パラメータ・タイプ	整数
デフォルト値	SORT_AREA_SIZE から導出される。
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM ... DEFERRED
値の範囲	2 個のデータベース・ブロックに相当する値から SORT_AREA_SIZE の値まで。

注意： インスタンスが共有サーバーのオプションで構成されていない場合、SORT_AREA_RETAINED_SIZE パラメータを使用することはお勧めしません。かわりに、PGA_AGGREGATE_TARGET を設定して、SQL 作業領域の自動サイズ指定を使用可能にすることをお勧めします。
SORT_AREA_RETAINED_SIZE は、下位互換性を保つために残されます。

SORT_AREA_RETAINED_SIZE には、ソート完了後にそのまま保持されるユーザー・グローバル領域 (UGA) メモリーの最大量 (バイト単位) を指定します。確保されたメモリー・サイズにより、メモリー内のソート領域の保持に使用される読取りバッファのサイズが制御されます。このメモリーは、最後の行がソート領域から取り出されると解放され、オペレーティング・システムではなく、UGA に戻されます。

SORT_AREA_SIZE

パラメータ・タイプ	整数
デフォルト値	65536
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM ... DEFERRED
値の範囲	最小値: 6 個のデータベース・ブロックに相当する値 最大値: オペレーティング・システム依存

注意： インスタンスが共有サーバーのオプションで構成されていない場合、SORT_AREA_SIZE パラメータを使用することはお薦めしません。かわりに、PGA_AGGREGATE_TARGET を設定して、SQL 作業領域の自動サイズ指定を使用可能にすることをお薦めします。SORT_AREA_SIZE は、下位互換性を保つために残されます。

SORT_AREA_SIZE には、ソートに使用するメモリーの最大量をバイトで指定します。ソート完了後、行が戻される前に、ソート用に割り当てられたすべてのメモリー（SORT_AREA_RETAINED_SIZE パラメータの指定量を除く）が解放されます。最後の行が戻されると、残りのメモリーが解放されます。

SPFILE

パラメータ・タイプ	文字列
構文	SPFILE = <i>spfile_name</i>
デフォルト値	ORACLE_HOME/dbs/spfile.ora
パラメータ・クラス	静的（自動リソース）
値の範囲	任意の有効な SPFILE
Oracle9i Real Application Clusters	複数インスタンスには同じ値を指定する必要がある。

このパラメータには、使用中のカレント・サーバー・パラメータ・ファイル（SPFILE）名を指定します。このパラメータは、クライアント側の PFILE で定義し、使用するサーバー・パラメータ・ファイル名を指定できます。

SQL92_SECURITY

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	静的
値の範囲	true false

SQL92 規格は、WHERE または SET 句で表列値を参照する UPDATE または DELETE 文を実行する場合に、ユーザーが表の SELECT 権限を持つことを、セキュリティ管理者が要求できることを指定します。SQL92_SECURITY により、ユーザーが、UPDATE または DELETE 文などを実行するために、SELECT オブジェクト権限が付与されるべきかどうかを指定できます。

SQL_TRACE

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	静的
値の範囲	true false

SQL_TRACE の値によって、SQL トレース機能が使用禁止または使用可能になります。このパラメータを true に設定すると、パフォーマンスを改善するためのチューニングについての情報が提供されます。この値は、DBMS_SYSTEM パッケージを使用しても変更できます。

STANDBY_ARCHIVE_DEST

パラメータ・タイプ	文字列
構文	STANDBY_ARCHIVE_DEST = <i>filespec</i>
デフォルト値	オペレーティング・システム依存
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	RAW 以外の有効なパス名またはデバイス名

STANDBY_ARCHIVE_DEST は、管理リカバリ・モードのスタンバイ・データベースのみに関連します。このパラメータには、プライマリ・データベースからのアーカイブ・ログの位置を指定します。STANDBY_ARCHIVE_DEST および LOG_ARCHIVE_FORMAT は、完全に修飾されたスタンバイ・ログ・ファイル名を作成するため、およびスタンバイ制御ファイルにファイル名を格納するために使用されます。

STANDBY_FILE_MANAGEMENT

パラメータ・タイプ	文字列
構文	STANDBY_FILE_MANAGEMENT = {MANUAL AUTO}
デフォルト値	MANUAL
パラメータ・クラス	動的:ALTER SYSTEM

STANDBY_FILE_MANAGEMENT によって、スタンバイ・ファイルの自動管理が使用可能になります。AUTO に設定すると、スタンバイ・データベース上で、ファイル管理操作（ファイルの追加、削除など）が自動的に実行されます。

STAR_TRANSFORMATION_ENABLED

パラメータ・タイプ	文字列
構文	STAR_TRANSFORMATION_ENABLED = {TEMP_DISABLE TRUE FALSE}
デフォルト値	FALSE
パラメータ・クラス	動的:ALTER SESSION

STAR_TRANSFORMATION_ENABLED によって、コストベースの問合せ変換を、スター問合せに適用するかどうかが判別されます。

TAPE_ASYNC_IO

パラメータ・タイプ	ブール
デフォルト値	true
パラメータ・クラス	静的
値の範囲	true false

TAPE_ASYNC_IO は、シーケンシャル・デバイスへの I/O（たとえば、テープとの間で行う Oracle データのバックアップまたはリストア）が非同期かどうか、つまり、Real Application Clusters プロセスで、テーブル・スキャン中に、CPU 処理と I/O 要求をオーバーラップできるかどうかを制御します。プラットフォームがシーケンシャル・デバイスへの非同期 I/O をサポートしている場合は、このパラメータをデフォルト値のままにしておくことをお勧めします。プラットフォームがシーケンシャル・デバイスへの非同期 I/O をサポートしている場合は、このパラメータをデフォルト値のままにしておくことをお勧めします。ただし、非同期 I/O の実装が安定してない場合は、TAPE_ASYNC_IO を `false` に設定することで、非同期 I/O を使用禁止にできます。プラットフォームがシーケンシャル・デバイスへの非同期 I/O をサポートしていない場合、このパラメータは無効です。

THREAD

パラメータ・タイプ	整数
デフォルト値	0
パラメータ・クラス	静的
値の範囲	0 ～ 使用可能スレッドの最大数
Oracle9i Real Application Clusters	このパラメータを指定する場合、複数インスタンスには、異なる値を指定する必要があります。

THREAD は、Oracle9i Real Application Clusters のパラメータで、インスタンスで使用される REDO スレッドの番号を指定します。

TIMED_OS_STATISTICS

パラメータ・タイプ	整数
デフォルト値	0
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM
値の範囲	無制限

TIMED_OS_STATISTICS には、クライアントからサーバーへの要求が発生したとき、または要求が完了したときに、オペレーティング・システムの統計情報を収集する間隔（秒単位）を指定します。

TIMED_STATISTICS

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	動的:ALTER SESSION、ALTER SYSTEM
値の範囲	true false

TIMED_STATISTICS には、時刻に関連する統計情報を収集するかどうかを指定します。

TRACE_ENABLED

パラメータ・タイプ	ブール
デフォルト値	true
パラメータ・クラス	動的:ALTER SYSTEM
値の範囲	true false
Oracle9i Real Application Clusters	すべてのインスタンスに、このパラメータを設定する必要がある。また、複数インスタンスには、同じ値を指定する必要がある。

TRACE_ENABLED は、Oracle の実行履歴またはコード・パスのトレースを制御します。オラクル社カスタマ・サポート・センターでは、この情報をデバッグに使用します。

TRACEFILE_IDENTIFIER

パラメータ・タイプ	文字列
構文	TRACEFILE_IDENTIFIER = "traceid"
デフォルト値	なし
パラメータ・クラス	動的:ALTER SESSION
値の範囲	ご使用のプラットフォーム上で、ファイル名に使用される可能性があるすべての文字

TRACEFILE_IDENTIFIER には、Oracle Trace のファイル名となるユーザー定義の識別子を指定します。このようなユーザー定義の識別子を使用すると、単純に名前からトレース・ファイルを識別できます。その場合、トレース・ファイルをオープンしたり、内容を確認する必要はありません。

TRANSACTION_AUDITING

パラメータ・タイプ	ブール
デフォルト値	true
パラメータ・クラス	動的: ALTER SYSTEM ... DEFERRED
値の範囲	true false

TRANSACTION_AUDITING を true に設定すると、ユーザー・ログイン名、ユーザー名、セッション ID、一部のオペレーティング・システムの情報およびクライアント情報を含む特別な REDO レコードが生成されます。逐次トランザクションごとに、セッション ID のみを含むレコードが生成されます。これらの後続のレコードは、セッション ID を含む最初のレコードにリンクされます。

TRANSACTIONS_PER_ROLLBACK_SEGMENT

パラメータ・タイプ	整数
デフォルト値	5
パラメータ・クラス	静的
値の範囲	1 以上。上限は、オペレーティング・システムによって異なる。
Oracle9i Real Application Clusters	複数インスタンスには、異なる値を指定可能。

TRANSACTIONS_PER_ROLLBACK_SEGMENT には、各ロールバック・セグメントを処理する同時トランザクションの数を指定します。起動時に取得されるロールバック・セグメントの最小数は、TRANSACTIONS をこのパラメータの値で割った数です。たとえば、TRANSACTIONS が 101 でこのパラメータが 10 の場合、取得されるロールバック・セグメントの最小数は 101/10 となり、11 に切り上げられます。

UNDO_MANAGEMENT

パラメータ・タイプ	文字列
構文	UNDO_MANAGEMENT = {MANUAL AUTO}
デフォルト値	MANUAL
パラメータ・クラス	静的
Oracle9i Real Application Clusters	複数インスタンスには、同じ値を指定する必要がある。

UNDO_MANAGEMENT には、システムが使用する必要がある UNDO 領域管理モードを指定します。AUTO に設定すると、インスタンスは、自動 UNDO 管理モードで起動します。手動 UNDO 管理モードでは、UNDO 領域がロールバック・セグメントとして外部的に割り当てられます。

UNDO_RETENTION

パラメータ・タイプ	整数
デフォルト値	900
パラメータ・クラス	動的:ALTER SYSTEM
値の範囲	0 ～ 2 ³² -1 (32 ビットで表される最大値)
Oracle9i Real Application Clusters	複数インスタンスには、同じ値を指定する必要がある。

UNDO_RETENTION には、データベースに保存するコミットされたロールバック情報の量 (秒単位) を指定します。UNDO_RETENTION は、データ・ブロックの以前のイメージを作成するために変更をロールバックする場合に、以前のロールバック情報を必要とする間合せて使用します。この値は、インスタンスの起動時に設定できます。

UNDO_SUPPRESS_ERRORS

パラメータ・タイプ	ブール
デフォルト値	false
パラメータ・クラス	動的:ALTER SESSION、ALTER SYSTEM
値の範囲	true false

UNDO_SUPPRESS_ERRORS によって、自動 UNDO 管理モードで手動 UNDO 管理モードの操作 (ALTER ROLLBACK SEGMENT ONLINE など) を実行しているときのエラーを回避できます。このパラメータを設定すると、すべてのアプリケーション・プログラムおよびスクリプトが自動 UNDO 管理モードに変換される前に、UNDO 表領域の特性を使用できます。たとえば、SET TRANSACTION USE ROLLBACK SEGMENT 文を使用するツールの場合、そのツールに「ALTER SESSION SET UNDO_SUPPRESS_ERRORS = TRUE」文を追加して、ORA-30019 エラーを回避できます。

UNDO_TABLESPACE

パラメータ・タイプ	文字列
構文	UNDO_TABLESPACE = <i>undoname</i>
デフォルト値	データベース内の最初に使用可能な UNDO 表領域
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	既存の有効な UNDO 表領域名
Oracle9i Real Application Clusters	複数インスタンスには、異なる値を指定可能。

UNDO_TABLESPACE には、インスタンスの起動時に使用する UNDO 表領域を指定します。インスタンスが手動 UNDO 管理モードのときにこのパラメータを指定すると、エラーが発生して起動できません。

USE_STORED_OUTLINES

構文: USE_STORED_OUTLINES = {TRUE | FALSE | *category_name*}

USE_STORED_OUTLINES パラメータは、オブティマイザが実行計画を生成するために格納されているパブリック・アウトラインを使用するかどうかを決定します。

USE_STORED_OUTLINES は、初期化パラメータではありません。

- TRUE に設定した場合、要求をコンパイルするときに、オブティマイザが DEFAULT カテゴリにストアド・アウトラインを使用します。
- FALSE に設定した場合、オブティマイザがストアド・アウトラインを使用しません。これはデフォルト値です。
- *category_name* を設定した場合、要求をコンパイルする場合に、オブティマイザが *category_name* カテゴリにストアド・アウトラインを使用します。

USER_DUMP_DEST

パラメータ・タイプ	文字列
構文	USER_DUMP_DEST = { <i>pathname</i> <i>directory</i> }
デフォルト値	オペレーティング・システム依存
パラメータ・クラス	動的: ALTER SYSTEM
値の範囲	有効なローカル・パス、ディレクトリまたはディスク

USER_DUMP_DEST には、ユーザー・プロセスのかわりに、サーバーがデバッグ・トレース・ファイルを書き込むディレクトリのパス名を指定します。

UTL_FILE_DIR

パラメータ・タイプ	文字列
構文	UTL_FILE_DIR = <i>pathname</i>
デフォルト値	なし
パラメータ・クラス	静的
値の範囲	有効なディレクトリ・パス

UTL_FILE_DIR により、PL/SQL によるファイル I/O を使用するディレクトリを 1 つ以上指定できます。複数のディレクトリを指定する場合、初期化パラメータ・ファイルの別々の行で各ディレクトリに対する UTL_FILE_DIR パラメータを繰り返す必要があります。

WORKAREA_SIZE_POLICY

パラメータ・タイプ	文字列
構文	WORKAREA_SIZE_POLICY = {AUTO MANUAL}
デフォルト値	AUTO (PGA_AGGREGATE_TARGET が設定されている場合) MANUAL (PGA_AGGREGATE_TARGET が設定されていない場合)
パラメータ・クラス	動的: ALTER SESSION、ALTER SYSTEM

WORKAREA_SIZE_POLICY には、作業領域のサイズ指定方法を指定します。このパラメータは、作業領域のチューニング時に使用するモードを制御します。

例

REDO ログの手動アーカイブの例 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;
```

次の文は、SCN 9356083 の REDO ログ・エントリを含む REDO ログ・ファイル・グループを手動でアーカイブします。

```
ALTER SYSTEM ARCHIVE LOG CHANGE 9356083;
```

次の文は、メンバー 'disk1:log6.log' を含む REDO ログ・ファイル・グループを、'diska:[arch\$]' という場所にあるアーカイブ REDO ログ・ファイルに手動でアーカイブします。

```
ALTER SYSTEM ARCHIVE LOG  
    LOGFILE 'disk1:log6.log'  
    TO 'diska:[arch$]';
```

クエリー・リライトを使用可能にする例 次の文は、明示的に使用禁止にされていないすべてのマテリアライズド・ビューに対するすべてのセッションにおけるクエリー・リライトを使用可能にします。

```
ALTER SYSTEM SET QUERY_REWRITE_ENABLED = TRUE;
```

セッションへのログインを制限する例 たとえば、アプリケーションのメンテナンス中は、RESTRICTED SESSION システム権限が付与されているアプリケーション開発者のみがログインできるように制限できます。このためには、次の文を発行します。

```
ALTER SYSTEM  
    ENABLE RESTRICTED SESSION;
```

次に、ALTER SYSTEM 文の KILL SESSION 句を使用すると、既存のセッションをどれでも終了できます。

アプリケーションのメンテナンスが終了した後で、次の文を発行することによって、CREATE SESSION システム権限が付与されているユーザーもログインできるようになります。

```
ALTER SYSTEM  
    DISABLE RESTRICTED SESSION;
```

共有プールの消去例 共有プールを消去してから、パフォーマンス分析を開始します。共有プールを消去する場合、次の文を発行します。

```
ALTER SYSTEM FLUSH SHARED_POOL;
```

チェックポイントの発生例 次の文は、チェックポイントを強制的に発生させます。

```
ALTER SYSTEM CHECKPOINT;
```

リソース制限を使用可能にする例 次の ALTER SYSTEM 文は、リソース制限を動的に使用可能にします。

```
ALTER SYSTEM SET RESOURCE_LIMIT = TRUE;
```

共有サーバーの例 次の文は、共有サーバー・プロセスの最小数を 25 に変更します。

```
ALTER SYSTEM SET SHARED_SERVERS = 25;
```

共有サーバー・プロセスの数が 25 より少ない場合は追加作成されます。共有サーバー・プロセスが 25 より多く、25 あれば負荷を管理できる場合は、現行のコールによる処理が終了した時点で、25 を超える分の共有サーバー・プロセスは終了します。

次の文は、TCP/IP プロトコルのディスパッチャ・プロセス数を 5 に、DECNet プロトコルのディスパッチャ・プロセス数を 10 に動的に変更します。

```
ALTER SYSTEM
  SET DISPATCHERS =
    ' (INDEX=0) (PROTOCOL=TCP) (DISPATCHERS=5) ',
    ' (INDEX=1) (PROTOCOL=DECNet) (DISPATCHERS=10) ';
```

TCP のディスパッチャ・プロセスの数が 5 より少ない場合、ディスパッチャ・プロセスが新しく作成されます。5 より多い場合は、接続されているユーザーが接続を切断した後に、5 を超える分のディスパッチャ・プロセスは終了します。

DECNet のディスパッチャ・プロセスの数が 10 より少ない場合は、ディスパッチャ・プロセスが新しく作成されます。10 より多い場合は、接続されているユーザーが接続を切断した後に、ディスパッチャ・プロセスは終了します。

これ以外のプロトコル用として既存ディスパッチャがある場合、この文は、そのディスパッチャの数に影響しません。

ライセンス・パラメータの変更例 次の文は、インスタンスにおけるセッションの最大数を 64 に、セッションの警告しきい値を 54 に動的に変更します。

```
ALTER SYSTEM
  SET LICENSE_MAX_SESSIONS = 64
  LICENSE_SESSIONS_WARNING = 54;
```

セッション数が 54 に達した場合、後続の各セッションの ALERT ファイルに警告メッセージが書き込まれます。RESTRICTED SESSION システム権限を持つユーザーも、後続セッションを開始する場合に、警告メッセージを受け取ります。

セッション数が 64 に達した場合、セッション数が再び 64 を下回るまでは、RESTRICTED SESSION システム権限を持つユーザー以外は新しいセッションを開始できません。

次の文は、インスタンスのセッションの最大数を動的に使用禁止にします。この文の実行後は、インスタンスのセッション数に制限がなくなります。

```
ALTER SYSTEM SET LICENSE_MAX_SESSIONS = 0;
```

次の文は、データベースのユーザー数の制限を 200 に動的に変更します。この文の実行後は、データベース・ユーザー数が 200 を超えることはありません。

```
ALTER SYSTEM SET LICENSE_MAX_USERS = 200;
```

ログ・スイッチの発生例 書き込み中のファイルの削除および名前の変更はできません。ただし、ログ・スイッチを強制的に発生させることによって、現行の REDO ログ・ファイル・グループまたはそのメンバーの 1 つを削除または名前の変更ができます。強制的に発生したログ・スイッチは、インスタンスの REDO ログ・スレッドのみに影響します。次の文は、ログ・スイッチを強制的に発生させます。

```
ALTER SYSTEM
  SWITCH LOGFILE;
```

分散リカバリを使用可能にする例 次の文は、分散リカバリを使用可能にします。

```
ALTER SYSTEM ENABLE DISTRIBUTED RECOVERY;
```

デモンストレーションまたはテストの目的で、分散リカバリを使用禁止にする場合があります。次の文を使用すると、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リカバリで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またはテストが終了した場合、ALTER SYSTEM 文に ENABLE DISTRIBUTED RECOVERY 句を指定して実行すると、分散リカバリを再び使用可能にできます。

セッションの中断例 あるユーザーのセッションで、他のユーザーが必要とするリソースを使用している場合、そのユーザーのセッションを中断させる必要があります。このユーザーは、セッションが中断したことを示すエラー・メッセージを受け取ります。このユーザーは、新しいセッションを開始しないかぎり、このデータベースをコールできません。次の V\$SESSION 動的パフォーマンス表のデータについて考えます。

```
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 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';
```

セッションの切断例 次の文は、V\$SESSION の SID と SERIAL# の値を使用して、ユーザー scott のセッションを切断します。

```
ALTER SYSTEM DISCONNECT SESSION '13, 8' POST_TRANSACTION;
```

10

SQL 文 : ALTER TABLE ~ ALTER TABLESPACE

この章では、次の SQL 文について説明します。

- ALTER TABLE
- ALTER TABLESPACE

ALTER TABLE

用途

ALTER TABLE を使用すると、非パーティション表、パーティション表、表パーティションおよび表サブパーティションの定義を変更できます。オブジェクト表またはオブジェクト列を含むリレーショナル表の場合は、ALTER TABLE を使用して型が変更された後に、表を参照する型の最新の定義に変換します。

前提条件

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

パーティション化操作におけるその他の前提条件 表の所有者でない場合、`drop_table_partition` または `truncate_table_partition` 句を使用するには、DROP ANY TABLE 権限が必要です。

`add_table_partition`、`modify_table_partition`、`move_table_partition` および `split_table_partition` 句を使用する場合、領域を確保する表領域に領域割当て制限が必要です。

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

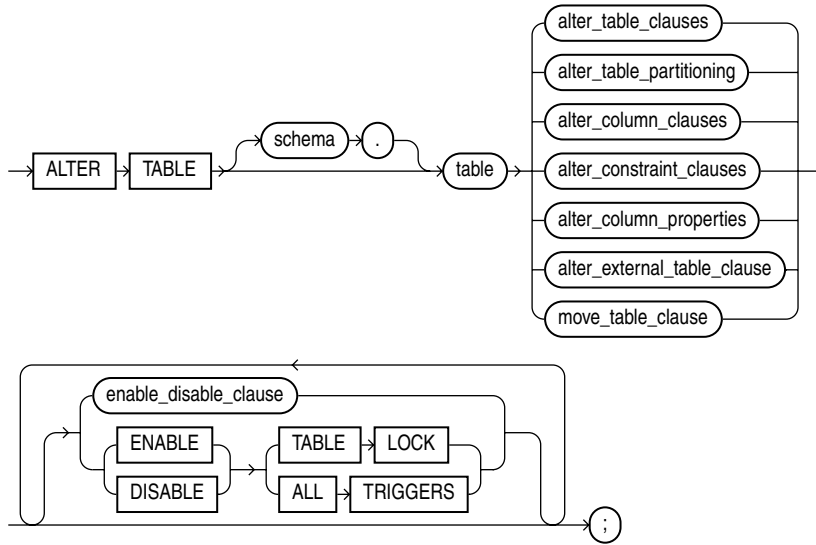
トリガーを使用可能または使用禁止にする場合、トリガーが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、ALTER ANY TRIGGER システム権限が必要です。

オブジェクト型を使用する場合のその他の前提条件 表を変更するときに列定義でオブジェクト型を使用する場合、そのオブジェクトが、変更する表と同じスキーマに属している必要があります。または、EXECUTE ANY TYPE システム権限またはそのオブジェクト型に対する EXECUTE スキーマ・オブジェクト権限が必要です。

参照： 索引を作成する場合に必要な権限については、12-58 ページの「[CREATE INDEX](#)」を参照してください。

構文

alter_table::=

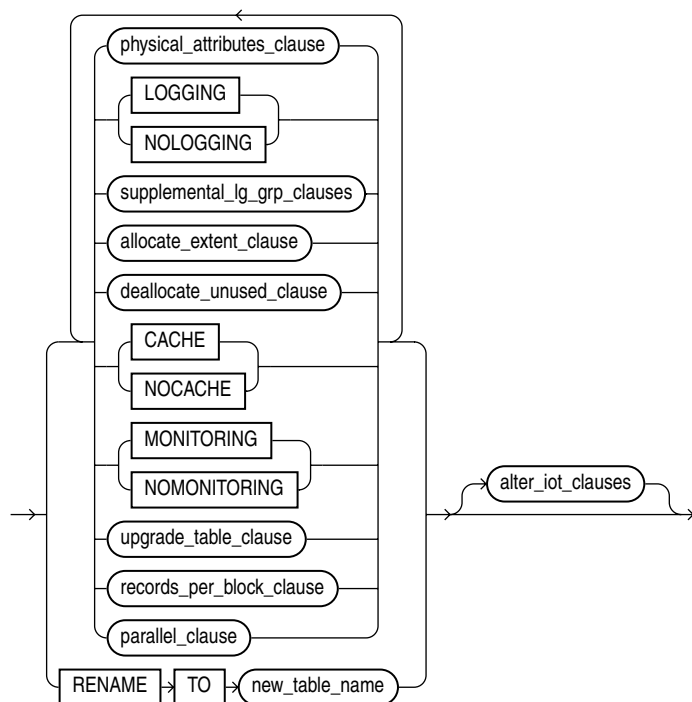


ALTER TABLE 構文のグループは、次のとおりです。

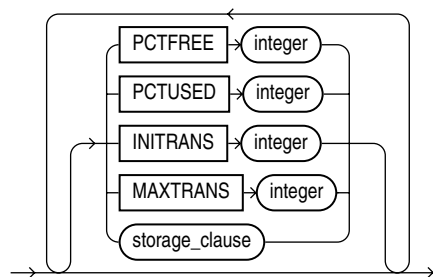
- [alter_table_clauses::=](#) (10-4 ページ)
- [alter_table_partitioning::=](#) (10-9 ページ)
- [alter_column_clauses::=](#) (10-18 ページ)
- [alter_constraint_clauses::=](#) (10-19 ページ)
- [alter_column_properties::=](#) (10-20 ページ)
- [alter_external_table_clause::=](#) (10-24 ページ)
- [move_table_clause::=](#) (10-25 ページ)
- [enable_disable_clause::=](#) (10-25 ページ)

副次句が句のすぐ後に記載されていない場合は、各句の後に副次句への参照先を示します。

alter_table_clauses::=

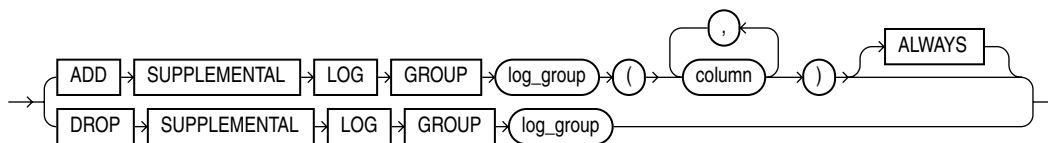


physical_attributes_clause::=

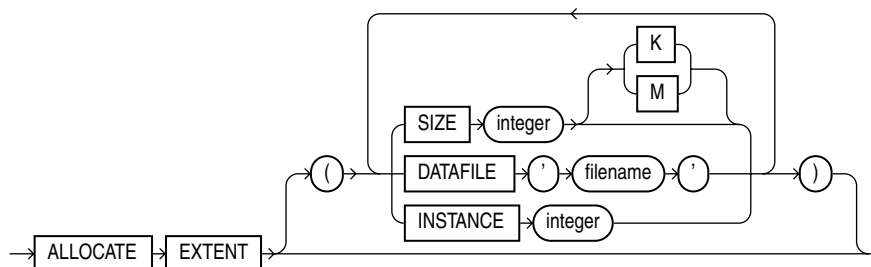


17-50 ページの「[storage_clause](#)」を参照してください。

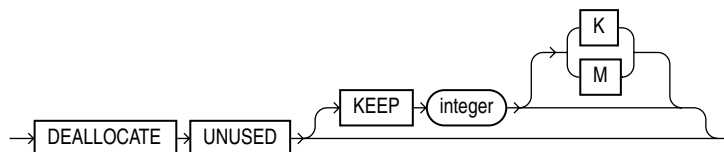
supplemental_lg_grp_clauses::=



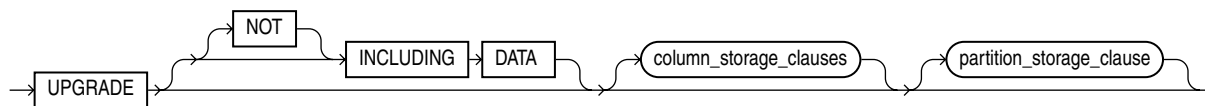
allocate_extent_clause::=



deallocate_unused_clause::=

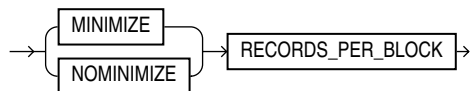


upgrade_table_clause::=

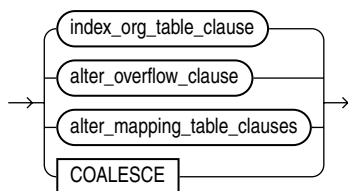


10-20 ページの「[column_properties::=](#)」および 10-23 ページの「[partition_storage_clause::=](#)」を参照してください。

records_per_block_clause::=

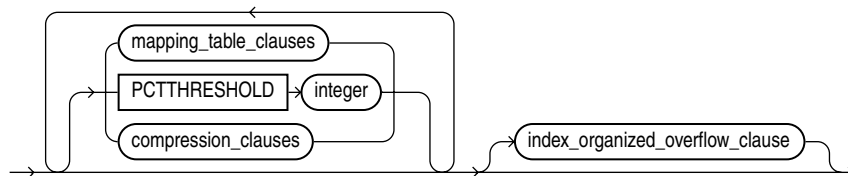


alter_iot_clauses::=

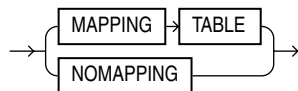


10-7 ページの「[alter_overflow_clause::=](#)」 および 10-8 ページの「[alter_mapping_table_clause::=](#)」を参照してください。

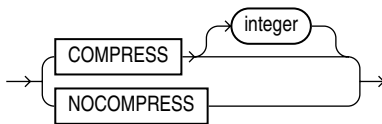
index_org_table_clause::=



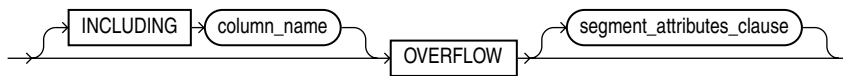
mapping_table_clauses::=



compression_clauses::=

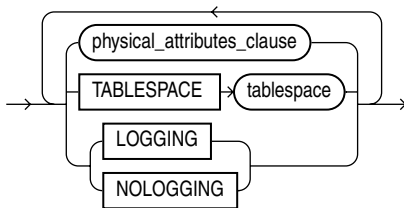


index_org_overflow_clause::=



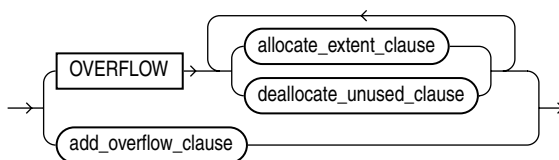
10-7 ページの「[segment_attributes_clause::=](#)」を参照してください。

segment_attributes_clause::=

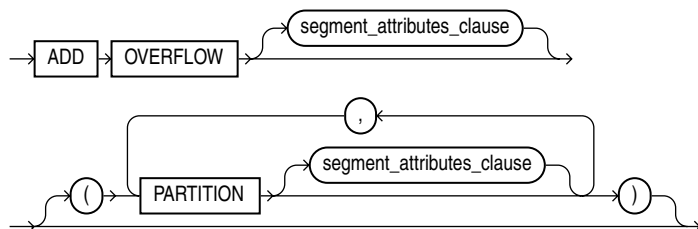


10-4 ページの「[physical_attributes_clause::=](#)」を参照してください。

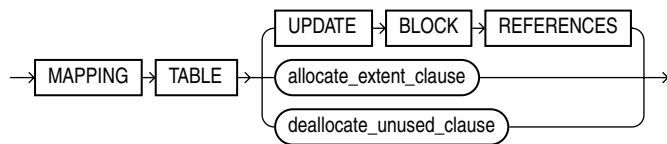
alter_overflow_clause::=



10-7 ページの「[segment_attributes_clause::=](#)」、10-5 ページの「[allocate_extent_clause::=](#)」および 10-5 ページの「[deallocate_unused_clause::=](#)」を参照してください。

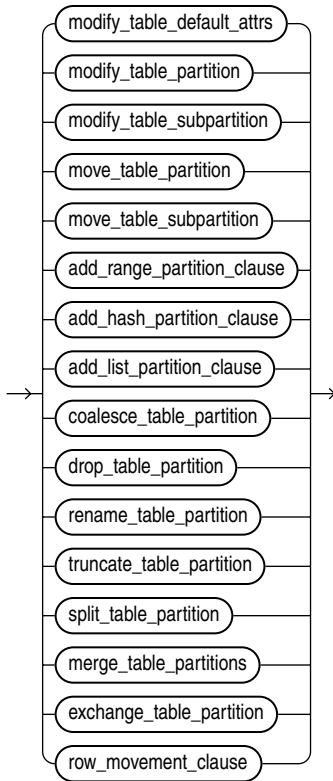
add_overflow_clause::=

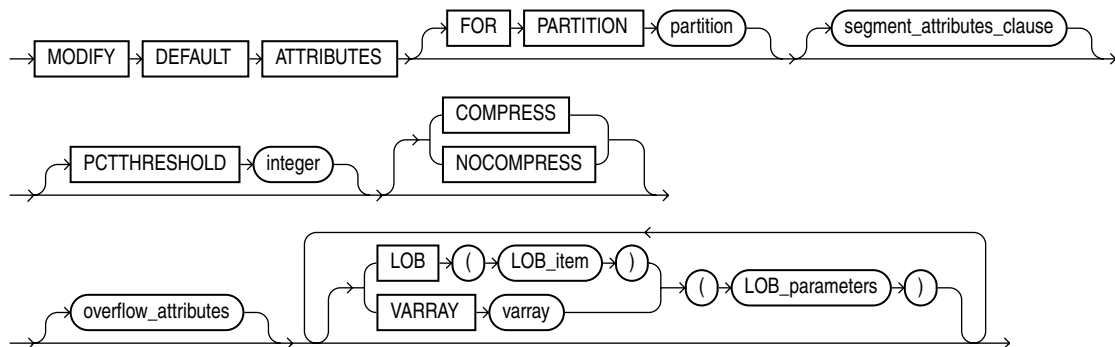
10-7 ページの「[segment_attributes_clause::=](#)」を参照してください。

alter_mapping_table_clause::=

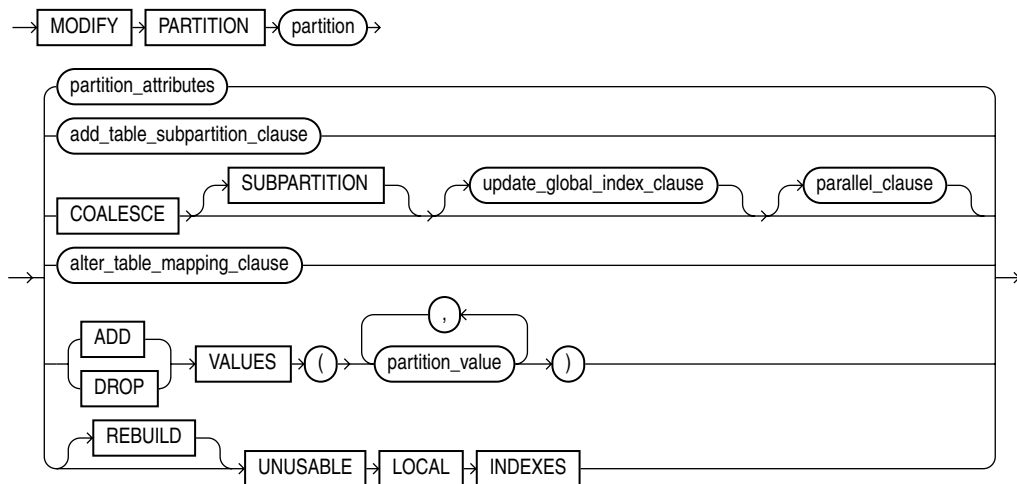
10-5 ページの「[allocate_extent_clause::=](#)」および 10-5 ページの「[deallocate_unused_clause::=](#)」を参照してください。

alter_table_partitioning::=



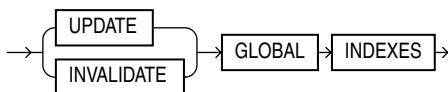
modify_table_default_attrs::=

10-7 ページの「[segment_attributes_clause::=](#)」および 10-23 ページの「[LOB_parameters::=](#)」を参照してください。

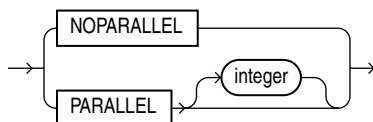
modify_table_partition::=

10-12 ページの「[add_table_subpartition::=](#)」を参照してください。

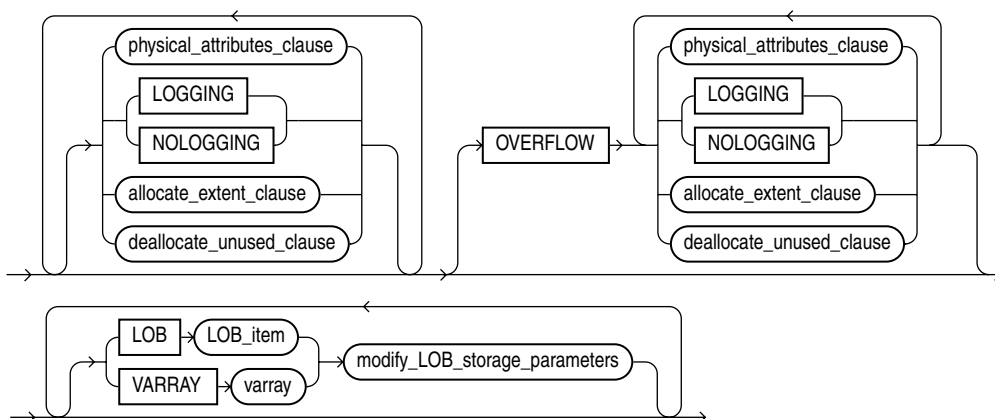
update_global_index_clause::=



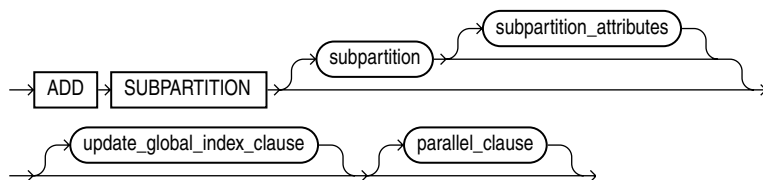
parallel_clause::=



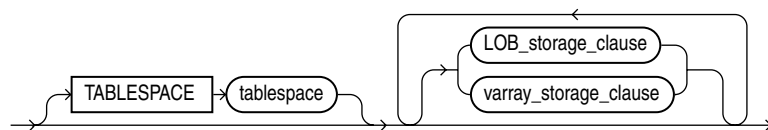
partition_attributes::=



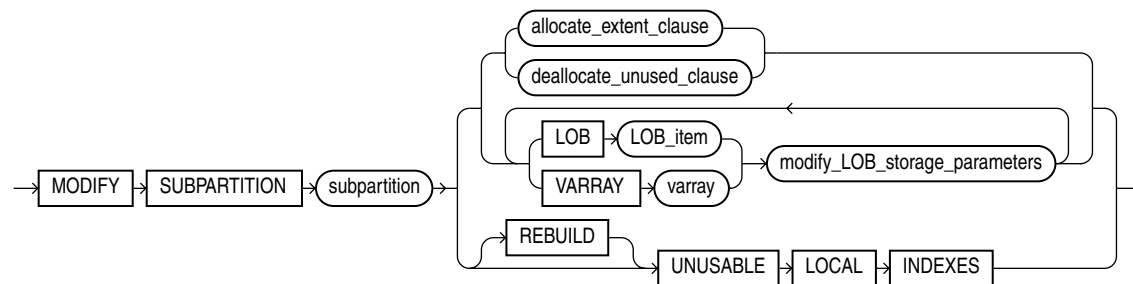
10-4 ページの「[physical_attributes_clause::=](#)」、10-5 ページの「[allocate_extent_clause::=](#)」、10-5 ページの「[deallocate_unused_clause::=](#)」および 10-24 ページの「[modify_lob_parameters::=](#)」を参照してください。

add_table_subpartition::=

10-11 ページの「[update_global_index_clause::=](#)」を参照してください。

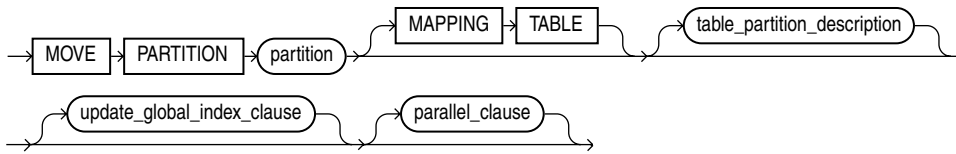
subpartition_attributes::=

10-22 ページの「[LOB_storage_clause::=](#)」、10-22 ページの「[varray_col_properties::=](#)」および 10-11 ページの「[parallel_clause::=](#)」を参照してください。

modify_table_subpartition::=

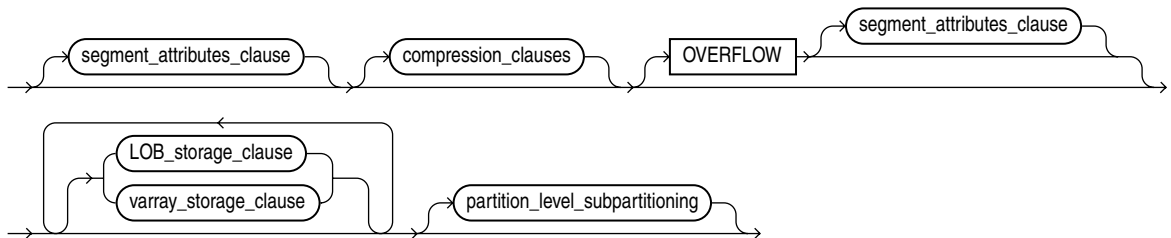
10-5 ページの「[allocate_extent_clause::=](#)」、10-5 ページの「[deallocate_unused_clause::=](#)」および 10-24 ページの「[modify_lob_parameters::=](#)」を参照してください。

move_table_partition::=



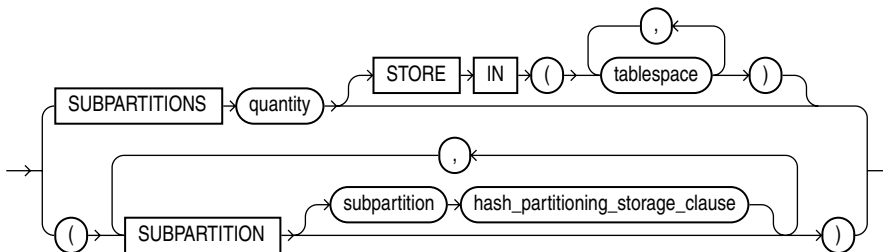
10-13 ページの「[table_partition_description::=](#)」、10-11 ページの「[update_global_index_clause::=](#)」、10-11 ページの「[parallel_clause::=](#)」を参照してください。

table_partition_description::=

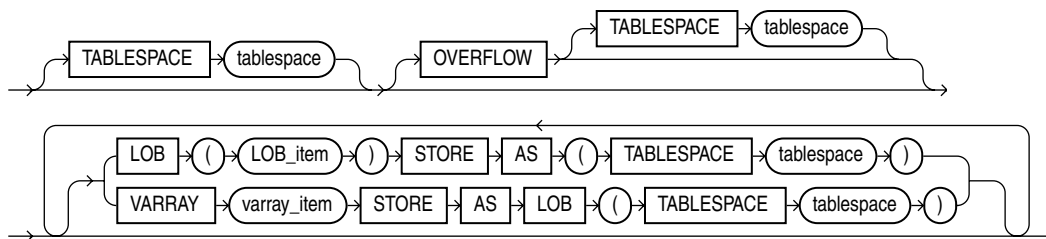


10-7 ページの「[segment_attributes_clause::=](#)」、10-7 ページの「[compression_clauses::=](#)」、10-22 ページの「[LOB_storage_clause::=](#)」および 10-22 ページの「[varray_col_properties::=](#)」を参照してください。

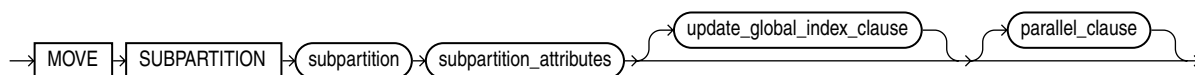
partition_level_subpartition::=



hash_partitioning_storage::=

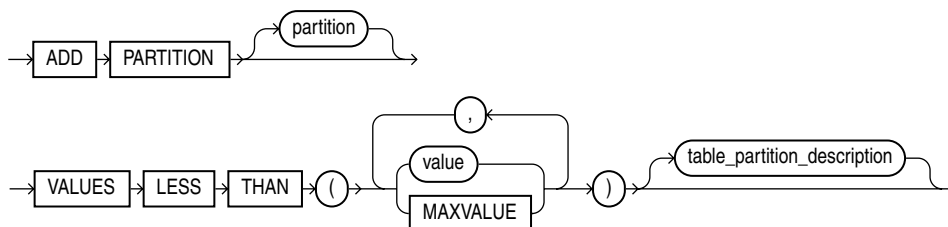


move_table_subpartition::=

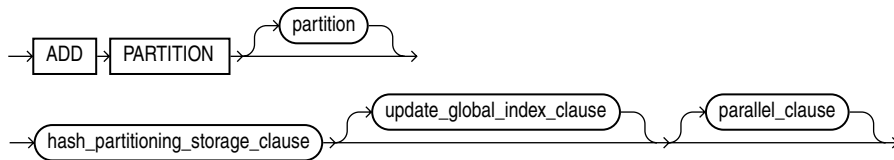


10-12 ページの「[subpartition_attributes::=](#)」、10-11 ページの「[update_global_index_clause::=](#)」、10-11 ページの「[parallel_clause::=](#)」を参照してください。

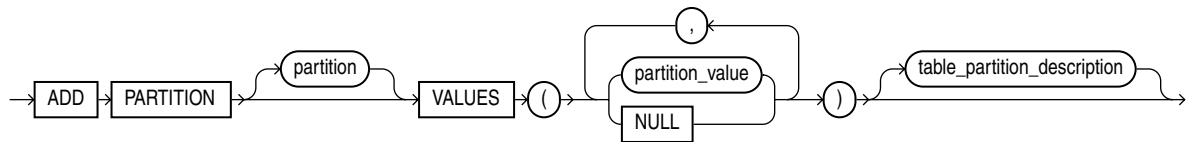
add_range_partition_clause::=



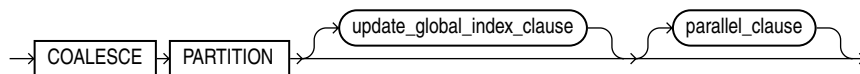
10-13 ページの「[table_partition_description::=](#)」を参照してください。

add_hash_partition_clause::=

10-14 ページの「[hash_partitioning_storage::=](#)」、10-11 ページの「[update_global_index_clause::=](#)」および 10-11 ページの「[parallel_clause::=](#)」を参照してください。

add_list_partition_clause::=

10-13 ページの「[table_partition_description::=](#)」を参照してください。

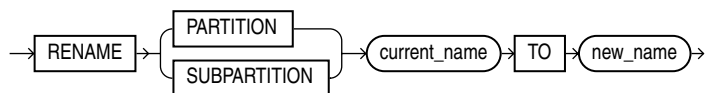
coalesce_table_partition::=

10-11 ページの「[update_global_index_clause::=](#)」および 10-11 ページの「[parallel_clause::=](#)」を参照してください。

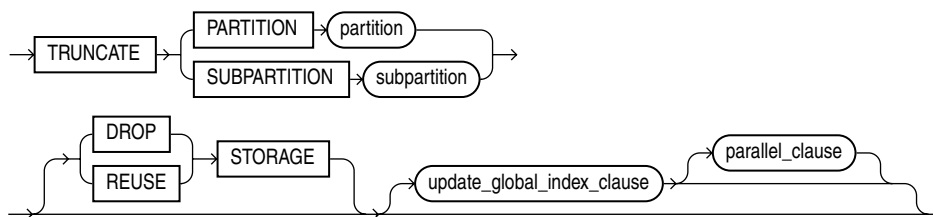
drop_table_partition::=

10-11 ページの「[update_global_index_clause::=](#)」および 10-11 ページの「[parallel_clause::=](#)」を参照してください。

rename_table_partition::=

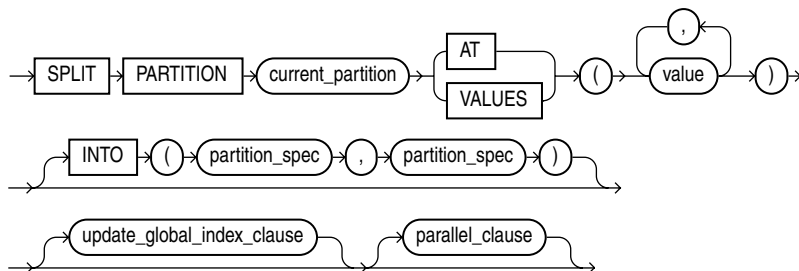


truncate_table_partition::=



10-11 ページの「[update_global_index_clause::=](#)」および 10-11 ページの「[parallel_clause::=](#)」を参照してください。

split_table_partition::=

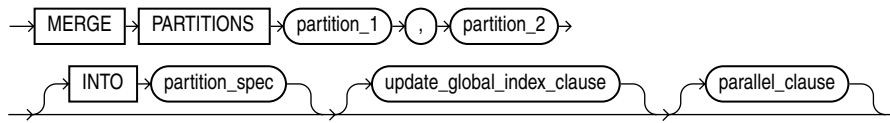


10-11 ページの「[update_global_index_clause::=](#)」および 10-11 ページの「[parallel_clause::=](#)」を参照してください。

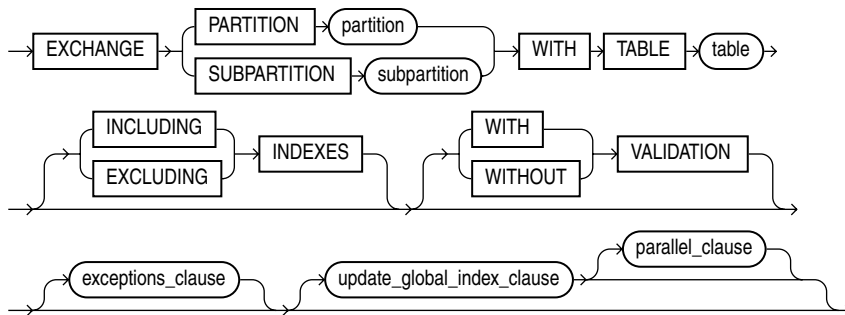
partition_spec::=



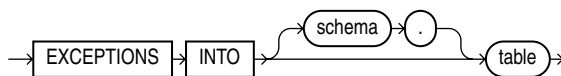
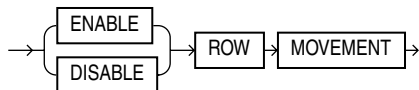
10-13 ページの「[table_partition_description::=](#)」を参照してください。

merge_table_partitions::=

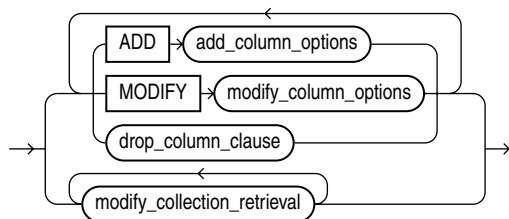
10-11 ページの「[update_global_index_clause::=](#)」および 10-11 ページの「[parallel_clause::=](#)」を参照してください。

exchange_table_partition::=

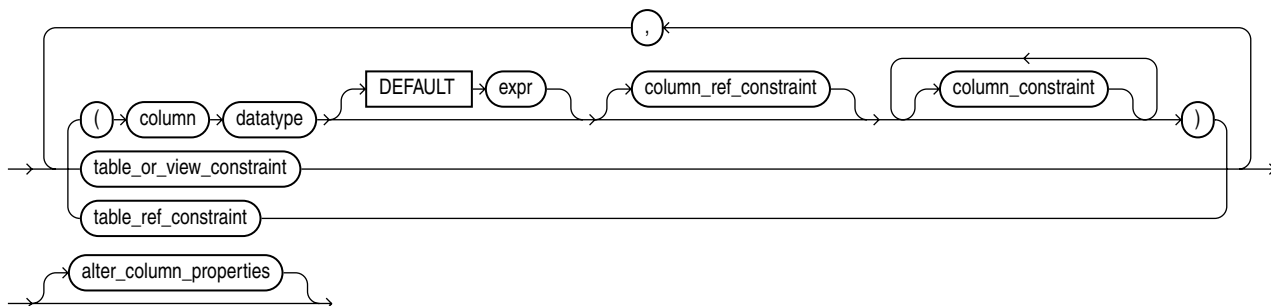
10-11 ページの「[update_global_index_clause::=](#)」および 10-11 ページの「[parallel_clause::=](#)」を参照してください。

exceptions_clause::=**row_movement_clause::=**

alter_column_clauses::=

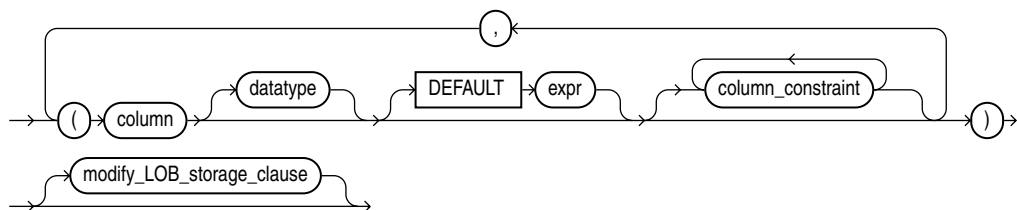


add_column_options::=



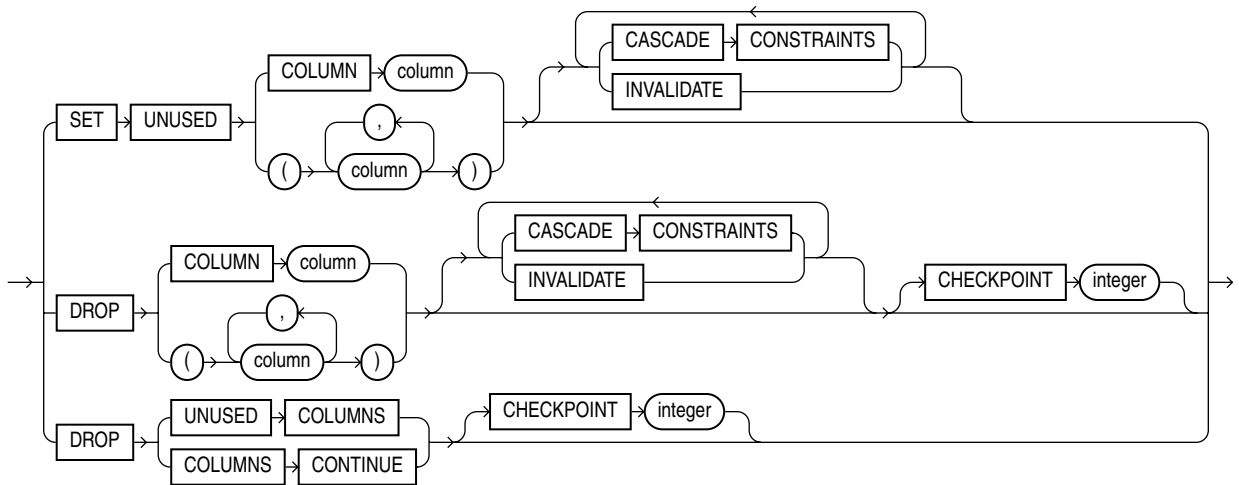
制約については、11-72 ページの「[constraint_clause](#)」および 10-20 ページの「[alter_column_properties::=](#)」を参照してください。

modify_column_options::=

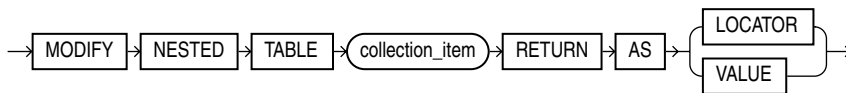


[column_constraint](#) については、11-72 ページの「[constraint_clause](#)」および 10-22 ページの「[LOB_storage_clause::=](#)」を参照してください。

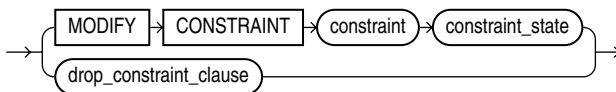
drop_column_clause::=



modify_collection_retrieval::=

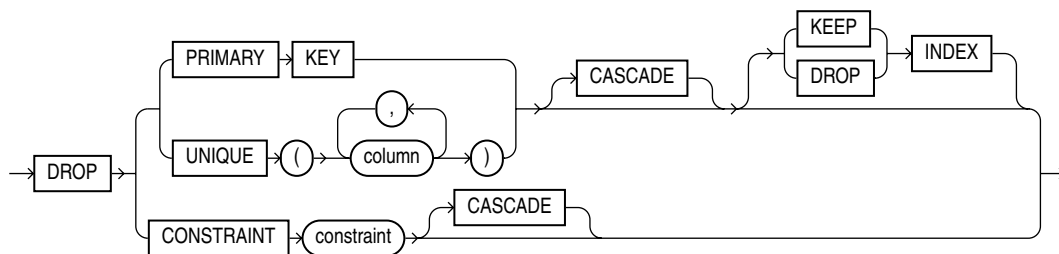


alter_constraint_clauses::=

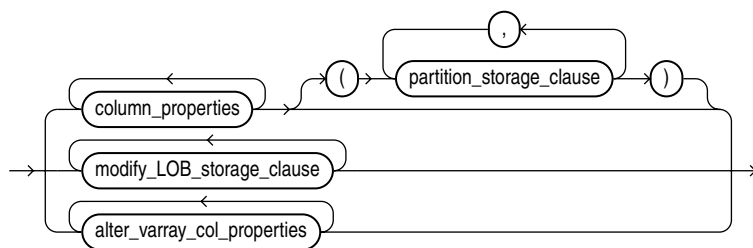


`constraint_state` については、11-72 ページの「[constraint_clause](#)」を参照してください。

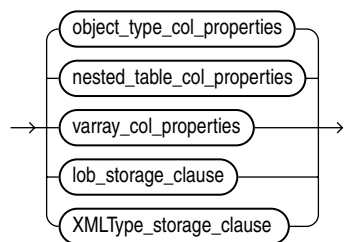
drop_constraint_clause::=



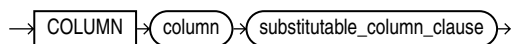
alter_column_properties::=



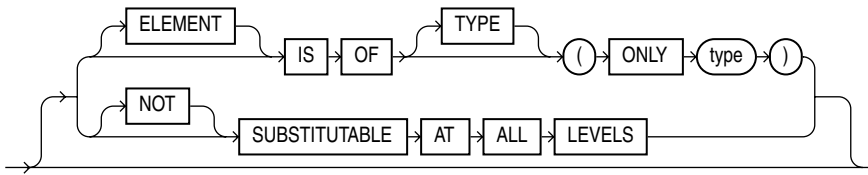
column_properties::=



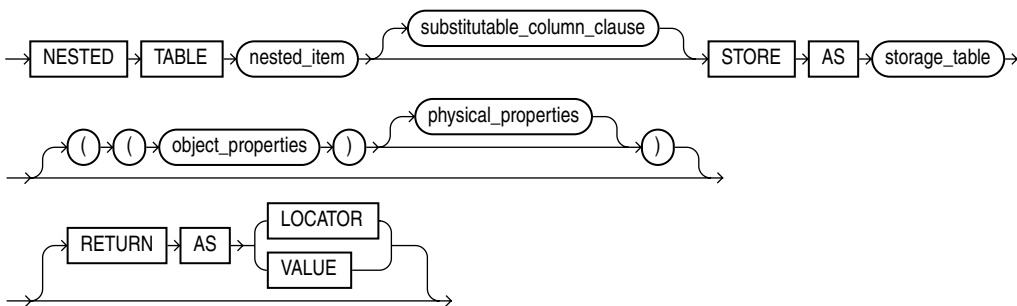
object_type_col_properties::=



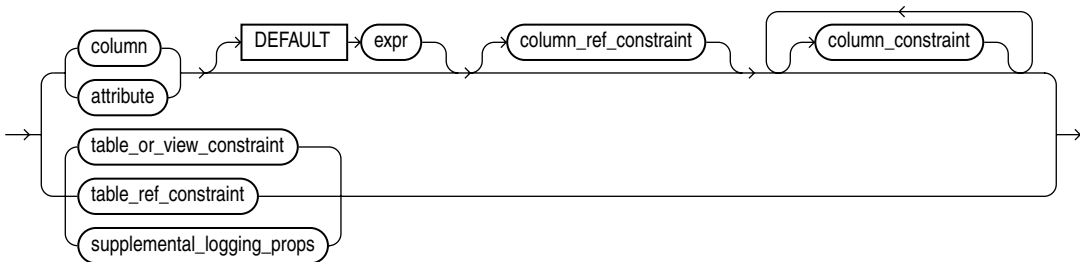
substitutable_column_clause::=



nested_table_col_properties::=

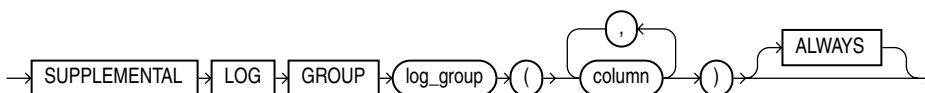


object_properties::=

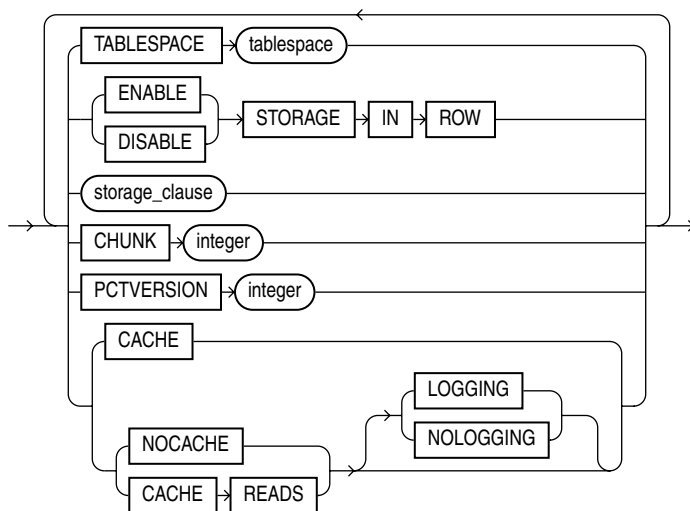


制約については、11-72 ページの「[constraint_clause](#)」を参照してください。

supplemental_logging_props::=

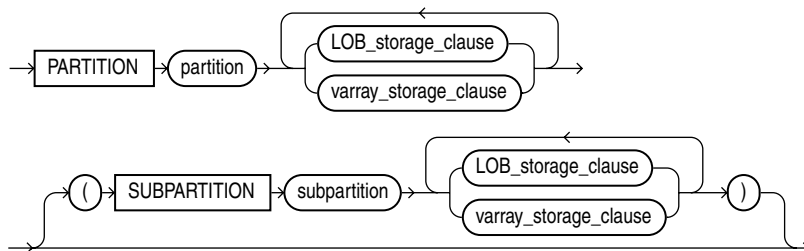


LOB_parameters::=



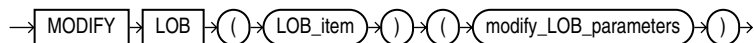
17-50 ページの「[storage_clause](#)」を参照してください。

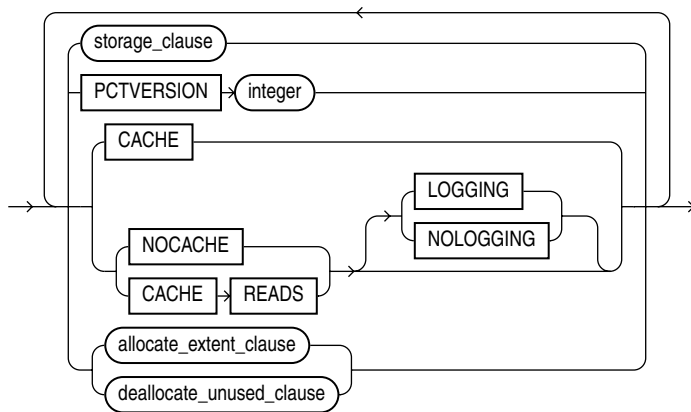
partition_storage_clause::=



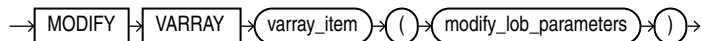
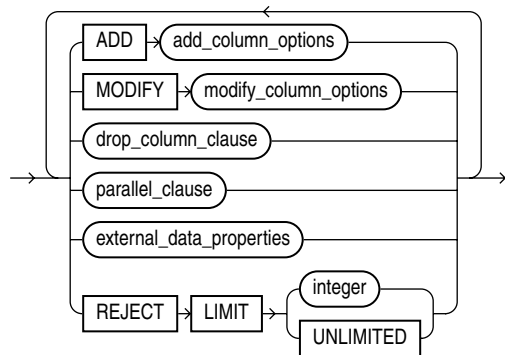
10-22 ページの「[LOB_storage_clause::=](#)」および 10-22 ページの「[varray_col_properties::=](#)」を参照してください。

modify_lob_storage_clause::=



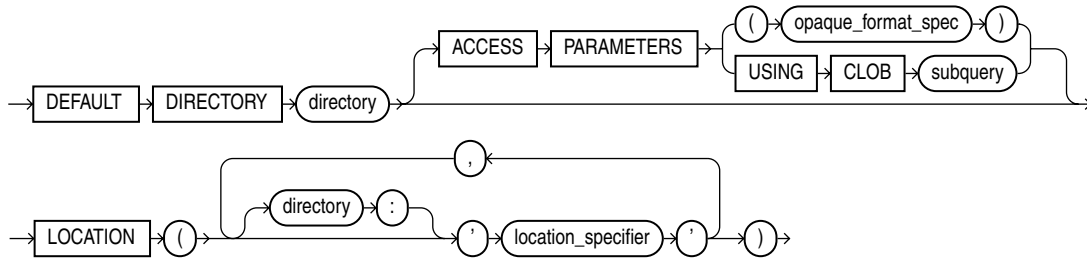
modify_lob_parameters::=

17-50 ページの「[storage_clause](#)」、10-5 ページの「[allocate_extent_clause::=](#)」、10-5 ページの「[deallocate_unused_clause::=](#)」を参照してください。

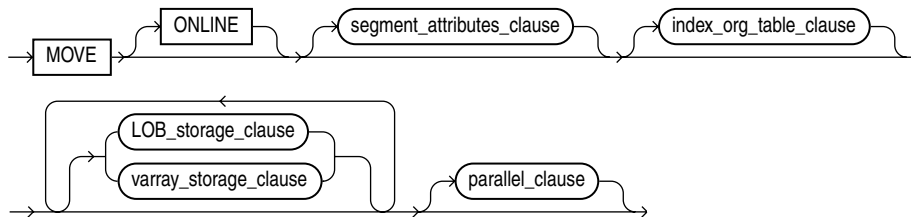
alter_varray_col_properties::=**alter_external_table_clause::=**

10-18 ページの「[add_column_options::=](#)」、10-18 ページの「[modify_column_options::=](#)」、10-19 ページの「[drop_column_clause::=](#)」、10-20 ページの「[drop_constraint_clause::=](#)」および 10-11 ページの「[parallel_clause::=](#)」を参照してください。

external_data_properties::=

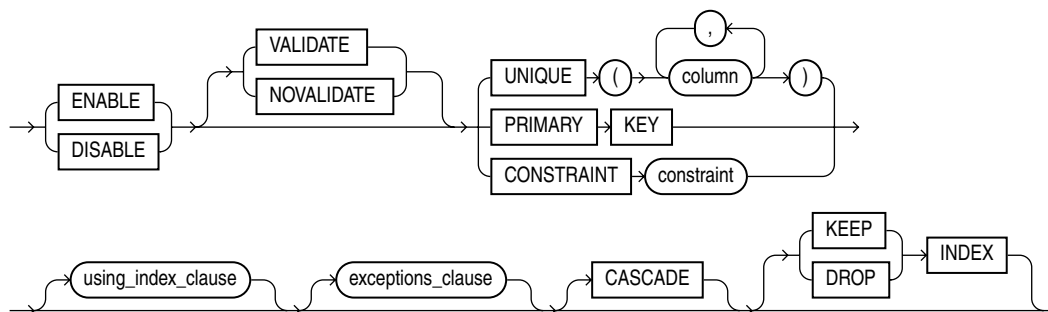


move_table_clause::=



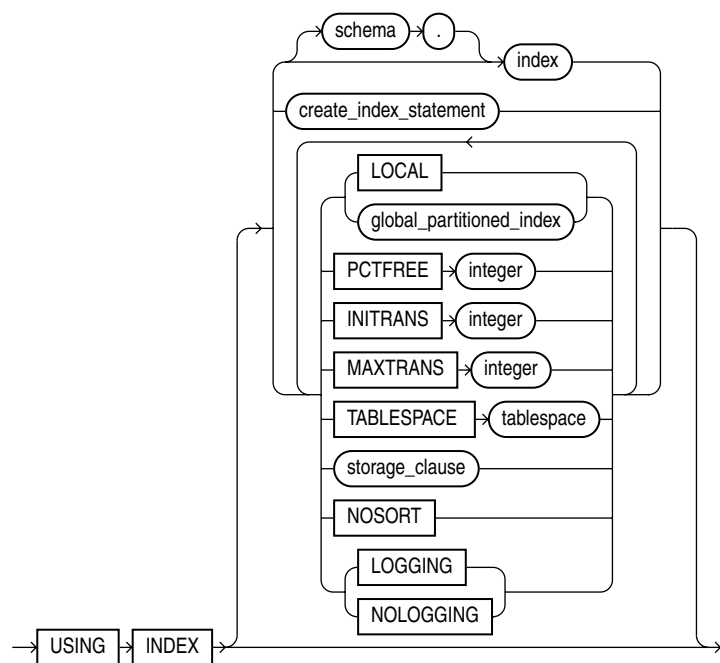
10-7 ページの「[segment_attributes_clause::=](#)」、10-6 ページの「[index_org_table_clause::=](#)」、10-22 ページの「[LOB_storage_clause::=](#)」および 10-22 ページの「[varray_col_properties::=](#)」を参照してください。

enable_disable_clause::=

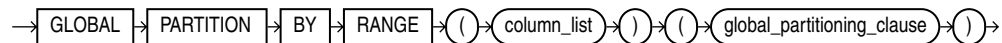


10-17 ページの「[exceptions_clause::=](#)」を参照してください。

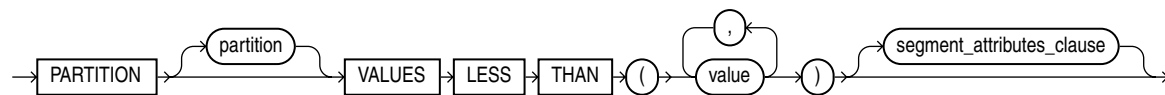
using_index_clause::=



global_partitioned_index::=



global_partitioning_clause::=



キーワードとパラメータ

ALTER TABLE 文の多くの句の機能は、CREATE TABLE 文での機能と同じです。これらの句の詳細は、14-6 ページの「[CREATE TABLE](#)」を参照してください。

注意： ALTER TABLE 文を実行すると、変更した表にアクセスするプロシージャおよびストアド・ファンクションが無効になる場合があります。無効になる条件および具体的な状況については、『Oracle9i データベース概要』を参照してください。

schema

変更する表が定義されているスキーマを指定します。*schema* を指定しない場合、この表が自分のスキーマに存在するとみなされます。

table

変更する表の名前を指定します。

一時表の制限事項

表の変更、表からの列の削除、または一時表の名前の変更ができます。ただし、一時表に対して次のことはできません。

- ネストした表または VARRAY 型の列の追加。その他の型の列の追加。
- 追加または変更された列の参照整合性（外部キー）制約の指定。
- 追加または変更された LOB 列に対する *LOB_storage_clause* の TABLESPACE、*storage_clause*、LOGGING または NOLOGGING、*LOB_index_clause* 句の指定。
- *physical_attributes_clause*、*nested_table_col_properties*、*parallel_clause*、*allocate_extent_clause*、*deallocate_unused_clause* または索引構成表句の指定。
- パーティション表と一時表の間でのパーティションの交換。
- LOGGING または NOLOGGING の指定。
- MOVE の指定。

外部表の制限事項

外部表の列を追加、削除または変更できます。ただし、外部表に対して次のことはできません。

- LONG、LOB またはオブジェクト型の列の追加、あるいは外部表の列のデータ型をこれらのデータ型のいずれかに変換
- 外部表への制約の追加
- 外部表の記憶域パラメータの変更
- LOGGING または NOLOGGING の指定
- MOVE の指定

注意： 1つ以上のマテリアライズド・ビューのマスター表となっている表を変更した場合、マテリアライズド・ビューには INVALID のマークが付けられます。無効なマテリアライズド・ビューは、クエリー・リライトでは使用できません。また、リフレッシュすることもできません。マテリアライズド・ビューを再検証する場合は、8-74 ページの「[ALTER MATERIALIZED VIEW](#)」を参照してください。

参照： マテリアライズド・ビューに関する一般的な情報については、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

alter_table_clauses

alter_table_clauses を使用すると、データベースの表を変更できます。

physical_attributes_clause

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

制限事項：

- 索引構成表の索引セグメントに対しては、PCTUSED パラメータを指定できません。
- ローカル管理表領域の表の記憶域属性を変更しようとすると、エラーが発生します。ただし、ローカル管理表領域にパーティション表のセグメントがあり、ディクショナリ管理表領域に他のセグメントがある場合、ディクショナリ管理表領域のセグメントの記憶域属性は変更されますが、ローカル管理表領域のセグメントの記憶域属性は変更されません。また、エラーも発生しません。

- 自動セグメント領域管理のセグメントの場合は、PCTUSED 設定の変更は無視されます。PCTFREE 設定を変更した場合、その変更をセグメントに割当て済のブロックに実装するには、DBMS_REPAIR.segment_fix_status プロシージャを実行する必要があります。

注意：

- 非パーティション表の場合、作成時に表に指定した値は新しく指定した値によってオーバーライドされます。
 - レンジ・パーティション表、リスト・パーティション表またはハッシュ・パーティション表の場合、新たに指定した値がその表のデフォルト値およびすべての既存パーティションに対する実際の値となり、そのパーティションにすでに設定されていた値はオーバーライドされます。既存パーティションの値をオーバーライドせずにデフォルトの表属性を変更する場合は、*modify_table_default_attrs* を使用してください。
 - コンポジット・パーティション表の場合、新しく指定した値がその表とその表のすべてのパーティションのデフォルト値、およびその表のすべてのサブパーティションに対する実際の値となり、そのサブパーティションにすでに設定されていた値はオーバーライドされます。既存のサブパーティションの値をオーバーライドせずにデフォルトのパーティション属性を変更する場合は、FOR PARTITION 句とともに *modify_table_default_attrs* を使用してください。
-

参照： 14-6 ページの「[CREATE TABLE](#)」の PCTFREE、PCTUSED、INITRANS および MAXTRANS パラメータ、および 17-50 ページの「[storage_clause](#)」を参照してください。

LOGGING | NOLOGGING

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

modify_table_default_attrs 句とともにこの句を使用した場合、パーティション表のロギング属性が影響を受けます。

LOGGING|NOLOGGING は、ALTER TABLE ...MOVE および ALTER TABLE ...SPLIT 操作のログを記録するかどうかも指定します。

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

LOB に対して LOGGING|NOLOGGING を省略した場合、次のようになります。

- CACHE を指定した場合は、LOGGING が使用されます (CACHE NOLOGGING は指定できないため)。
- NOCACHE または CACHE READS を指定した場合は、LOB が存在する表領域の属性がデフォルトのロギング属性として使用されます。

NOLOGGING は、行データとともにインラインに格納された LOB に適用されません。つまり、LOB に対する NOLOGGING を 4000 バイト未満の値に指定し、STORAGE IN ROW を使用禁止にしていなかった場合、NOLOGGING の指定は無視され、LOB データは他の表データと同様に扱われます。

NOLOGGING モードでは、データの変更時に、(新しいエクステンツに INVALID のマークを設定し、ディクショナリの変更を記録するために) 最小限のログが記録されます。メディア・リカバリ中に NOLOGGING が適用された場合、REDO データのログ記録が中断されるため、エクステン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、損失してはならない表の場合は、NOLOGGING 操作の後にバックアップを作成する必要があります。

データベースが ARCHIVELOG モードで実行されている場合、LOGGING モードでの操作前に作成されたバックアップからのメディア・リカバリによって、表がリストアされます。ただし、NOLOGGING モードでの操作の前に作成したバックアップからのメディア・リカバリでは、表はリストアされません。

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。

参照： *logging_clause* およびパラレル DML の詳細は、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

supplemental_lg_grp_clauses

supplemental_lg_grp_clauses を使用すると、補助 REDO ログ・グループを追加および削除できます。

- ADD LOG GROUP 句を使用すると、REDO ログ・グループを追加できます。
- DROP LOG GROUP 句を使用すると、必要でなくなったときに REDO ログ・グループを削除できます。

参照： 補助 REDO ログ・グループの詳細は、『Oracle9i データベース管理者ガイド』を参照してください。

allocate_extent_clause

表、パーティション、サブパーティション、オーバーフロー・データ・セグメント、LOB データ・セグメントまたは LOB 索引に新しいエクステントを明示的に割り当てる場合は、*allocate_extent_clause* を使用します。

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

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

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

DATAFILE 'filename' 新しいエクステントを割り当てるデータ・ファイルを、表の表領域、オーバーフロー・データ・セグメント、LOB データ表領域または LOB 索引の中から 1 つ指定します。このパラメータを省略した場合、Oracle によって選択されます。

INSTANCE integer INSTANCE integer を指定すると、指定したインスタンスに対応付けられた空きリスト・グループが新しいエクステントを使用できるようになります。インスタンス数が空きリスト・グループの最大数を超えた場合、インスタンス数 / 最大数という除算が行われ、その余りから、使用する空きリスト・グループが識別されます。インスタンスは初期化パラメータ INSTANCE_NUMBER の値で識別されます。このパラメータを指定しない場合、表に領域が割り当てられますが、特定の空きリスト・グループからは割り当てられません。かわりにマスター空きリストが使用され、必要に応じて領域が割り当てられます。

注意： Real Application Clusters 環境で Oracle を使用する場合のみ、このパラメータを使用します。

参照：『Oracle9i データベース概要』を参照してください。

deallocate_unused_clause

表、パーティション、サブパーティション、オーバーフロー・データ・セグメント、LOB データ・セグメントまたは LOB 索引の最後にある未使用領域の割当てを明示的に解除し、解放された領域を表領域の他のセグメントから利用できるようにする場合は、***deallocate_unused_clause*** を使用します。最高水位標を超える（データベース・ブロックがデータを受け取るためにフォーマットされていない）未使用領域のみ解放できます。

割当てが解除される表領域のユーザー割当て制限は、解放される領域の量のみ増加します。

未使用領域の割当て解除は、オブジェクトの終わりから開始し、オブジェクトの先頭にある最高水位標に向かって進行します。エクステントが割当て解除の範囲内に完全に含まれる場合、エクステント全体が解放されて再利用可能となります。エクステントの一部が含まれる場合、最高水位標までのすでに使用されている部分がエクステントになり、残りの未使用領域が解放されて再利用可能になります。

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

参照： これらのパラメータについては、17-50 ページの「***storage_clause***」を参照してください。

KEEP integer 割当てを解除した後に表、オーバーフロー・データ・セグメント、LOB データ・セグメントまたは LOB 索引に残す、最高水位標を超えるバイト数を指定します。

- KEEP を省略した場合、最高水位標が INITIAL および MINEXTENTS のサイズを超えると
きは、最高水位標を超えるすべての未使用領域が解放されます。最高水位標が
INITIAL または MINEXTENTS のサイズより小さい場合は、MINEXTENTS を超えるすべ
での未使用領域が解放されます。
- KEEP オプションを指定した場合、指定した量の領域が保持され、残りの領域のみが解
放されます。このとき、残りのエクステント数が MINEXTENTS より小さくなった場合、
MINEXTENTS はそのエクステント数に変更されます。また、初期エクステントが
INITIAL より小さくなった場合、INITIAL がそのサイズに変更されます。
- どちらの場合も、NEXT は、割当てを解除した最後のエクステント・サイズに設定され
ます。

CACHE | NOCACHE

CACHE 句 この句は、アクセス頻度の高いデータについて、フル・テーブル・スキャンの実行時に、この表のために取り出されたブロックを、バッファ・キャッシュ内の LRU リストの最高使用頻度側に置く場合に指定します。この属性は、小規模な参照表で有効です。

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

制限事項：CACHE は、索引構成表に対して指定できません。ただし、索引構成表は暗黙的に CACHE 動作を提供します。

NOCACHE 句 この句は、アクセス頻度の低いデータについて、フル・テーブル・スキャンの実行時に、この表のために取り出されたブロックを、バッファ・キャッシュ内の LRU リストの最低使用頻度側に置く場合に指定します。

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

制限事項：NOCACHE は、索引構成表に対して指定できません。

MONITORING | NOMONITORING

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

参照： この句の使用方法の詳細は、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。

NOMONITORING 句 表の変更統計を収集しない場合は、NOMONITORING を指定します。

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

upgrade_table_clause

upgrade_table_clause は、オブジェクト表およびオブジェクト列を含むリレーショナル表に関連します。この句を使用すると、ターゲットとなる表のメタデータが参照される各型の最新バージョンに準拠する型に変換されます。表が有効な場合、表のメタデータは変更されないままとなります。

INCLUDING DATA 表のデータを型の最新バージョンの形式に変更する場合は、INCLUDING DATA を指定します（型が変更されたときに変換されなかった場合）。[column_properties](#) および [partition_storage_clause](#) を使用して表をアップグレードする間に、新しいすべての列の記憶域を定義できます。これはデフォルトです。

表が古いバージョンの型に基づくデータを含むかどうかを確認するには、USER_TAB_COLUMNS データ・ディクショナリ・ビューの DATA_UPGRADED 列を参照します。

NOT INCLUDING DATA 列データを変更しないままにする場合は、NOT INCLUDING DATA を指定します。

制限事項：表が Oracle8 リリース 8.0.x のイメージ・フォーマットを含む場合、NOT INCLUDING DATA を指定できません。表がこのような列を含むかどうかを確認するには、USER_TAB_COLUMNS データ・ディクショナリ・ビューの V80_FMT_IMAGE 列を参照します。

参照：

- データ・ディクショナリ・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。
- 表が依存する型を変更するときに、依存する表のデータを変換する場合の詳細は、11-6 ページの「ALTER TYPE」を参照してください。
- 表のデータを型の最新バージョンの形式に変換しないことによる影響の詳細は、『Oracle9i アプリケーション開発者ガイド - オブジェクト・リレーショナル機能』を参照してください。

records_per_block_clause

1 ブロックに格納できるレコード数を制限するかどうかを指定する場合は、*records_per_block_clause* を使用します。この句によって、この後、表に作成されるビットマップ索引はできるだけ小さくなり（圧縮され）ます。

制限事項：

- 表にすでにビットマップ索引が定義されている場合は、MINIMIZE または NOMINIMIZE のいずれも指定できません。まず、ビットマップ索引を削除する必要があります。
- この句は、索引構成表またはネストした表に対して指定できません。

MINIMIZE 表の各ブロックの最大レコード数を計算し、ブロックに含まれるレコード数がその数を超えないように挿入操作を制限するために、MINIMIZE を指定します。

制限事項：MINIMIZE は、空の表に対して指定できません。

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

RENAME TO

表の名前を *new_table_name* に変更する場合は、RENAME 句を使用します。

制限事項：マテリアライズド・ビューの名前は変更できません。

注意： この句を使用した場合、依存するすべてのマテリアライズド・ビューは無効になります。マテリアライズド・ビューの詳細は、13-5 ページの「[CREATE MATERIALIZED VIEW](#)」および『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

alter_iot_clauses

index_org_table_clause

14-27 ページの「CREATE TABLE」の「[index_org_table_clause](#)」を参照してください。

alter_overflow_clause

alter_overflow_clause を使用して、索引構成表の定義を変更します。索引構成表は、主キーによってソートされたデータを保持する表であり、主キーに基づくアクセスおよび操作には最適です。

注意： 索引構成表に列を追加した場合、各列の最大サイズが評価され、行の最大値が計算されます。オーバーフロー・セグメントが必要で、OVERFLOW を指定していない場合は、エラーが発生し ALTER TABLE 文は実行されません。このチェック機能によって、索引構成表に対する後続の DML 操作が、オーバーフロー・セグメントがないために失敗することはありません。

PCTTHRESHOLD integer 索引ブロック内で、索引構成表の行を格納するために確保されている領域の割合（パーセント）を指定します。PCTTHRESHOLD は、主キーを保持するために十分な大きさである必要があります。指定したしきい値を超える列から始まる行の後続列はすべて、オーバーフロー・セグメントに格納されます。PCTTHRESHOLD は 1 ～ 50 の値を取る必要があります。PCTTHRESHOLD を指定しない場合のデフォルト値は 50 です。

制限事項：PCTTHRESHOLD は、索引構成表の個別パーティションに対して指定できません。

INCLUDING column_name 索引構成表の行を索引部分とオーバーフロー部分に分割する列を指定します。主キー列は常に索引に格納されます。column_name は、最後の主キー列でもその他の非主キー列でもかまいません。column_name に続くすべての非主キー列は、オーバーフロー・データ・セグメントに格納されます。

制限事項：この句は、索引構成表の個別パーティションに対して指定できません。

注意： column_name で行を分割しようとした場合に、行の索引部分のサイズが、PCTTHRESHOLD の値（指定またはデフォルト）を超えると、Oracle は PCTTHRESHOLD の値に基づいて、行を切り離します。

overflow_attributes 索引構成表に対して、変更するオーバーフロー・データ・セグメントの物理記憶域属性およびロギング属性を指定する場合に、overflow_attributes を使用します。この句に指定するパラメータは、オーバーフロー・データ・セグメントにのみ適用できます。

参照： 14-6 ページの「[CREATE TABLE](#)」を参照してください。

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

STORE IN tablespace 句を使用すると、オーバーフロー・セグメント全体に対する表領域の記憶域を指定できます。PARTITION 句を使用すると、パーティションによるセグメントに対する表領域の記憶域を指定できます。

パーティション化された索引構成表の場合、次の点に注意してください。

- PARTITION を指定しない場合、それぞれのパーティションに自動的にオーバーフロー・セグメントが割り当てられます。これらのセグメントの物理属性は表のレベルから継承されます。
- 1 つ以上のパーティションに別々の物理属性を指定する場合、表の中のパーティションそれぞれに個別に属性を指定する必要があります。パーティションの名前を指定する必要はありませんが、パーティションが作成された順番で属性を指定する必要があります。

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

TABLESPACE を指定していないパーティションがある場合、表に対して指定された表領域が使用されます。表のレベルで TABLESPACE を指定していない場合は、そのパーティションの主キー索引セグメントの表領域が使用されます。

alter_mapping_table_clause

alter_mapping_table_clause は、*table* が索引構成されており、マッピング表を持つ場合のみに有効です。

UPDATE BLOCK REFERENCES UPDATE BLOCK REFERENCES を指定すると、主キーによって識別されるブロックの適切なデータベース・アドレスを持つマッピング表の論理 ROWID 列の一部として格納され、失効したと推測されるすべてのデータ・ブロック・アドレスを更新することができます。

allocate_extent_clause *allocate_extent_clause* を指定すると、新しいエクステンツを索引構成表のマッピング表の終わりに割り当てます。

deallocate_unused_clause *deallocate_unused_clause* を指定すると、索引構成表のマッピング表の終わりにある未使用領域の割当てを解除できます。

参照： 14-28 ページの「CREATE TABLE」の
[「mapping_table_clause」](#) を参照してください。

COALESCE

このキーワードは *table* が索引構成されている場合のみに関連します。COALESCE を指定すると、空きブロックの再利用が可能な索引構成表の主キー索引ブロックを結合することができます。この句は、*parallel_clause* とあわせて指定することができます。

alter_table_partitioning

次の句は、パーティション表にのみ適用されます。1 つの ALTER TABLE 文の中で、パーティション操作を他のパーティション操作または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。

注意：

- ブロック・サイズが異なる表領域のパーティション化されたデータベース・エンティティの記憶領域には、制限事項があります。これらの制限事項の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。
 - 1 つ以上のマテリアライズド・ビューのマスター表で、パーティションを削除、交換、切捨て、移動、変更または分割した場合、その表に関する大量の既存ロード情報が削除されます。したがって、前述の操作のいずれかを行う前に、必ず依存するマテリアライズド・ビューをリフレッシュしてください。
 - table でビットマップ結合索引が定義されている場合、table のパーティションを変更するすべての操作によって、索引に UNUSABLE のマークが付けられます。
-
-

modify_table_default_attrs

modify_table_default_attrs を使用すると、table の属性に対する新しいデフォルト値を指定できます。その後に作成するパーティションおよび LOB パーティションは、パーティションまたは LOB パーティションの作成時に明示的にオーバーライドしないかぎり、この値を継承します。既存のパーティションおよび LOB パーティションは、この句の影響を受けません。

文の中で指定した属性のみが影響を受け、指定されたデフォルト値は、個々のパーティション・レベルで指定された属性でオーバーライドされます。

- FOR PARTITION は、コンポジット・パーティション表にのみ適用されます。*partition* の属性に新しいデフォルト値を指定します。その後に作成する *partition* のサブパーティションおよび LOB パーティションは、サブパーティションまたは LOB パーティションの作成時に明示的にオーバーライドしないかぎり、この値を継承します。既存のサブパーティションは、この句の影響を受けません。
- PCTTHRESHOLD、COMPRESS および OVERFLOW 句は、パーティション化された索引構成表のみに有効です。
- PCTUSED パラメータは、索引構成表の索引セグメントに対して指定できません。
- COMPRESS は、圧縮がすでに表レベルで指定されている場合にのみ指定できます。

modify_table_partition

`modify_table_partition` を使用すると、`partition` の実際の物理属性を変更できます。そのパーティションの 1 つ以上の LOB 項目の記憶域属性を任意に変更できます。パーティションに対して、物理属性であるロギング属性、PCTFREE、PCTUSED、INITRANS または MAXTRANS パラメータ、あるいは記憶域パラメータに新しい値を指定できます。

表がコンポジット・パーティション表の場合は、次の点に注意してください。

- `allocate_extent_clause` を指定した場合、`partition` のそれぞれのサブパーティションにエクステントが割り当てられます。
- `deallocate_unused_clause` を指定した場合、`partition` のそれぞれのサブパーティションから未使用の記憶域の割当てが解除されます。
- この句で変更された他の属性は、`partition` のサブパーティションでも変更され、既存の値はオーバーライドされます。既存のサブパーティションの属性が変更されないようにするには、`modify_table_default_attrs` の FOR PARTITION 句を使用します。

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

add_table_subpartition `add_table_subpartition` を使用すると、ハッシュ・サブパーティションを `partition` に追加できます。Oracle は、ハッシュ・ファンクションによって `partition` の他のサブパーティションから再ハッシュされた行を、新しいサブパーティションに入れます。

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

TABLESPACE を指定しなかった場合、新しいサブパーティションは `partition` のデフォルトの表領域に格納されます。

表のすべてのグローバル索引が無効になります。`update_global_index_clause` を使用すると、操作中にこれらの索引を更新できます。

選択したパーティションに対応するローカル索引パーティションが追加されます。追加したパーティションに対応するローカル索引パーティションに UNUSABLE のマークが付けられ、再構築する必要があります。

制限事項：この句は、リスト・パーティションに対して指定できません。

COALESCE SUBPARTITION COALESCE は、ハッシュ・パーティションおよびハッシュ・サブパーティションのみに適用されます。ハッシュ・サブパーティションが選択され、その内容が 1 つ以上の他のサブパーティション（ハッシュ・ファンクションが決定する）に分割された後、選択されたサブパーティションを削除する場合は、COALESCE SUBPARTITION を使用します。

表のすべてのグローバル索引が無効になります。 [update_global_index_clause](#) を使用すると、操作中にこれらの索引を更新できます。

選択したパーティションに対応するローカル索引パーティションが削除されます。1 つ以上の吸収パーティションに対応するローカル索引パーティションに、UNUSABLE のマークが付けられます。この索引パーティションは、再構築する必要があります。

ADD | DROP VALUES 句 これらの句は、リスト・パーティションを変更する場合のみ有効です。

- ADD VALUES 句を使用すると、*partition* の *partition_value* リストが追加した値まで拡張されます。追加するパーティションの値に、14-38 ページの「CREATE TABLE」の [list_partitioning](#) でリストしているすべての規則および制限事項に準拠する値を設定する必要があります。
- DROP VALUES 句を使用すると、1 つ以上の *partition_value* の排除によって *partition* の *partition_value* リストが削除されます。この句を指定すると、Oracle はこの値の行が存在しないことを検証します。そのような行が存在する場合は、エラーが戻されます。

注意： 表でローカル同一キー索引が定義されている場合、DROP VALUES 操作は機能強化されます。

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

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

制限事項：

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

update_global_index_clause

表パーティションで DDL 文を実行するときに、グローバル索引が *table* で定義されている場合は、DDL 文が実行中のパーティションを除き、索引全体が無効になります。この句を使用すると、DDL 操作中に変更するグローバル索引パーティションを更新でき、DDL 文の後に索引を再構築する必要がなくなります。

UPDATE GLOBAL INDEXES UPDATE GLOBAL INDEXES を指定すると、*table* で定義したグローバル索引を更新することができます。

INVALIDATE GLOBAL INDEXES INVALIDATE GLOBAL INDEXES を指定すると、*table* で定義したグローバル索引を無効にすることができます。

どちらも指定しない場合、グローバル索引は無効になります。

制限事項：この句は、グローバル索引のみをサポートします。ドメイン索引および索引構成表はサポートされません。また、この句では **USABLE** および **VALID** の索引のみが更新されます。**UNUSABLE** の索引は使用禁止のままになり、**INVALID** のグローバル索引は無視されます。

parallel_clause

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

注意： *parallel_clause* 構文は、以前のリリースの構文に代わるものです。以前のリリースの構文は下位互換用にサポートされていますが、動作がわずかに異なることがあります。

NOPARALLEL シリアル実行を行う場合は、**NOPARALLEL** を指定します。これはデフォルトです。

PARALLEL 並列度を選択する場合は、**PARALLEL** を指定します。並列度とは、すべての関係するインスタンスで使用可能な CPU の数に、初期化パラメータ **PARALLEL_THREADS_PER_CPU** の値を掛けたものです。

PARALLEL *integer* *integer* には、パラレル操作で使用するパラレル・スレッド数である**並列度**を指定します。各パラレル・スレッドは、1、2 個のパラレル実行サーバーを使用します。通常、最適な並列度が計算されるため、*integer* に値を指定する必要はありません。

制限事項：

- *table* に LOB 型またはユーザー定義オブジェクト型の列が含まれている場合、この *table* での INSERT、UPDATE および DELETE は、通知なしに逐次実行されます。ただし、後続の問合せはパラレルで実行されます。
- *parallel_clause* を *move_table_clause* と組み合わせて指定する場合、このパラレル化は移動のみに適用され、後続の表での DML 操作および問合せには適用されません。

参照： 14-43 ページの「CREATE TABLE」の「[parallel_clause](#)に関する注意事項」を参照してください。

modify_table_subpartition

modify_table_subpartition 句を使用すると、*table* の個々のサブパーティションに対して、記憶域の割当てまたは割当て解除を行うことができます。

UNUSABLE LOCAL INDEXES UNUSABLE LOCAL INDEXES によって、*subpartition* に関連付けられたすべてのローカル索引サブパーティションに、UNUSABLE のマークが付けられます。

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

制限事項：

- サブパーティションに指定できる *modify_lob_parameters* は、*allocate_extent_clause* および *deallocate_unused_clause* のみです。
- この句は、リスト・パーティション表に対して指定できません。
- リスト・パーティションには、UNUSABLE LOCAL INDEXES のいずれの句も指定できません。

rename_table_partition

rename_table_partition を使用すると、表パーティションまたは表サブパーティションの名前を *current_name* から *new_name* に変更できます。パーティションおよびサブパーティションのどちらの場合も、*new_name* は同じ表に存在するすべてのパーティションおよびサブパーティションと異なる値である必要があります。

table が索引構成されている場合、対応する主キー索引パーティションに、既存のオーバーフロー・パーティションおよびマッピング表パーティションと同じ名前が割り当てられます。

move_table_partition

move_table_partition 句を使用すると、*partition* を別のセグメントへ移動できます。パーティション・データの別の表領域への移動、断片化を削減するためのデータの再クスタ化、および作成時の物理属性の変更ができます。

表に LOB 列が含まれている場合、*LOB_storage_clause* を使用して、このパーティションに関連付けられた LOB データおよび LOB 索引セグメントを移動できます。この場合、指定した LOB のみが影響を受けます。特定の LOB 列に *LOB_storage_clause* を指定しなかった場合、その列の LOB データおよび LOB 索引セグメントは移動されません。

ヒープ構成表のすべてのグローバル索引が無効になります。

update_global_index_clause を使用すると、操作中にこれらの索引を更新できます。索引構成表のグローバル索引は主キー・ベースであるため、使用禁止になりません。

選択したパーティションに対応するローカル索引パーティションが移動されます。移動したパーティションは、空でない場合に UNUSABLE のマークが付けられ、再構築する必要があります。

LOB データ・セグメントを移動する場合、古いデータ・セグメントおよび対応する索引セグメントが削除され、新しい表領域を指定しない場合でも、新しいセグメントが作成されます。

移動操作では、*parallel_clause* (指定されている場合) からパラレル属性が取得されます。*parallel_clause* が指定されていない場合は、表のデフォルトのパラレル属性があれば、これが使用されます。どちらも指定されていない場合は、パラレル化を使用せずに移動が行われます。

MOVE PARTITION の *parallel_clause* を指定すると、*table* のデフォルトのパラレル属性は変更されません。

注意： 索引構成表の場合、主キーのアドレスおよびその値を使用して、論理 ROWID が構成されます。論理 ROWID は、表のセカンダリ索引に格納されます。索引構成表のパーティションを移動した場合、ROWID のアドレス部分に変更され、パフォーマンスの障害になる場合があります。最適なパフォーマンスを維持するには、移動したパーティションのセカンダリ索引を再構築し、ROWID を更新してください。

参照： 論理 ROWID の詳細は、『Oracle9i データベース概要』を参照してください。

MAPPING TABLE MAPPING TABLE 句は、索引構成表用に定義されたマッピング表を持つ索引構成表のみに関連します。索引パーティションとともにマッピング表は移動され、対応するすべてのビットマップ索引パーティションに UNUSABLE のマークが付けられます。

参照： 14-28 ページの「CREATE TABLE」の
「[mapping_table_clause](#)」を参照してください。

表パーティションの移動の制限事項

- `partition` がハッシュ・パーティションである場合、この句に `TABLESPACE` の属性以外は指定できません。
- この句は、サブパーティションを含むパーティションに対して指定できません。ただし、`move_table_subpartition_clause` を使用してサブパーティションを移動できます。

move_table_subpartition

`move_table_subpartition` 句を使用すると、`subpartition` を別のセグメントへ移動できます。TABLESPACE を指定しない場合、サブパーティションは同じ表領域に残ります。

[update_global_index_clause](#) を使用し、操作中に `table` のグローバル索引を更新できます。サブパーティションが空でない場合、UNUSABLE のマークが付けられ、移動したサブパーティションに対応するすべてのローカル索引サブパーティションを再構築する必要があります。

表に LOB 列が含まれている場合、`LOB_storage_clause` を使用して、このサブパーティションに関連付けられた LOB データおよび LOB 索引セグメントを移動できます。この場合、指定した LOB のみが影響を受けます。特定の LOB 列に `LOB_storage_clause` を指定しなかった場合、その列の LOB データおよび LOB 索引セグメントは移動されません。

LOB データ・セグメントを移動する場合、古いデータ・セグメントおよび対応する索引セグメントが削除され、新しい表領域を指定しない場合でも、新しいセグメントが作成されます。

ADD PARTITION 句

ADD PARTITION 句を使用すると、`table` にハッシュ・パーティション、レンジ・パーティションまたはリスト・パーティションを追加できます。

`table` で定義されているローカル索引に、基本の表パーティションと同じ名前で新しいパーティションが追加されます。索引に同じ名前のパーティションがすでに存在する場合、SYS_Pn という形式でパーティションの名前が生成されます。

`table` が索引構成されている場合、表で定義されたすべてのマッピング表およびオーバーフロー領域に、パーティションが追加されます。

add_range_partition_clause

add_range_partition_clauseを使用すると、新しいレンジ・パーティションをパーティション表の一番上（最後の既存のパーティションの後）に追加できます。新しいパーティションに作成時の物理属性を指定できます。表に LOB 列が含まれている場合、1 つ以上の LOB 項目にパーティション・レベルの属性を指定することもできます。

ドメイン索引が **table** で定義されている場合、IN_PROGRESS または FAILED のマークが付いていると無効になります。

1 つの表当たりのパーティションの上限は 64KB-1 です。

制限事項：

- 既存の上位パーティションにある各パーティション化キーのパーティションの上限が MAXVALUE の場合、表にパーティションを追加できません。そのかわり、**split_table_partition** 句を使用して、表の始めまたは中間にパーティションを追加します。
- **compression_clause** および OVERFLOW は、パーティション化された索引構成表に対してのみ有効です。パーティション表にすでにオーバーフロー・セグメントが存在する場合にかぎり、OVERFLOW を指定できます。表レベルで圧縮が使用可能な場合にかぎり、圧縮を指定できます。
- PCTUSED パラメータは、索引構成表の索引セグメントに対して指定できません。

VALUES LESS THAN (value_list) 新しいパーティションの上限を指定します。

value_list は、**column_list** に対応するリテラル値を順序どおりにカンマで区切ったリストです。**value_list** の値は、表内にある既存の最上位パーティションのパーティション境界より大きくする必要があります。

partition_level_subpartition **partition_level_subpartition** 句は、コンポジット・パーティション表のみで使用できます。この句を使用すると、新しいレンジ・パーティションのハッシュ・サブパーティションを指定できます。コンポジット・パーティションは、次のいずれかの方法で指定できます。

- 個々のパーティションを名前で指定できます。また、任意に各パーティションが格納される表領域を指定することもできます。
- サブパーティションの数を指定できます（または、サブパーティションが格納される 1 つ以上の表領域を指定することもできます）。この場合、SYS_SUBPnnn という形式のパーティション名を割り当てます。表領域の数は、サブパーティションの数と等しくなくてもかまいません。サブパーティションの数が表領域の数より多い場合、表領域名は繰り返されます。

サブパーティションは、サブパーティション・レベルで指定可能な TABLESPACE を除き、新しいパーティションに指定されているすべての属性を継承します。サブパーティションまたはパーティション・レベルで指定されなかった属性は、表レベルのデフォルト値から継承されます。

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

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

参照： 14-6 ページの「[CREATE TABLE](#)」を参照してください。

add_hash_partition_clause

パーティション表の一番上に新しいハッシュ・パーティションを追加する場合は、*add_hash_partition_clause* を使用します。Oracle は、ハッシュ・ファンクションによって *table* の他のパーティションから再ハッシュされた行を新しいサブパーティションに入れます。

パーティション名を指定します。また、パーティションが格納される表領域を指定することもできます。名前を指定しない場合、SYS_Pnnn という形式でパーティション名が割り当てられます。TABLESPACE を指定しなかった場合、新しいパーティションは表のデフォルトの表領域に格納されます。他の属性は、常に、表レベルのデフォルトから継承されます。

update_global_index_clause を使用し、操作中に *table* のグローバル索引を更新できます。ヒープ構成表の場合、この操作がパーティション間でデータを再ハッシュすると UNUSABLE のマークが付けられ、対応するローカル索引パーティションを再構築する必要があります。索引構成表の索引は主キー・ベースであるため、使用禁止になりません。

parallel_clause *parallel_clause* を使用すると、新しいパーティションの作成をパラレル化するかどうかを指定できます。

参照： ハッシュ・パーティション化の詳細は、14-6 ページの「[CREATE TABLE](#)」および『Oracle9i データベース概要』を参照してください。

add_list_partition_clause

add_list_partition_clause を使用すると、パーティションの新しい一連の値を使用して、*table* に新しいパーティションを追加できます。新しいパーティションに作成時の物理属性を指定できます。表に LOB 列が含まれている場合、1 つ以上の LOB 項目にパーティション・レベルの属性を指定することもできます。

表にリスト・パーティションを追加すると、表に定義されているすべてのローカル索引に、対応する索引パーティションが同じ値のリストで追加されます。グローバル索引には影響がありません。

参照： リスト・パーティションの詳細および制限事項については、14-38 ページの「[CREATE TABLE](#)」の「[list_partitioning](#)」を参照してください。

coalesce_table_partition

COALESCE は、ハッシュ・パーティションおよびハッシュ・サブパーティションのみに適用されます。*coalesce_table_partition* 句を使用すると、ハッシュ・パーティションが選択され、その内容が 1 つ以上の残りのパーティション（ハッシュ・ファンクションが決定）に分配された後、選択されたパーティションが削除されます。

ヒープ構成表のすべてのグローバル索引が無効になります。

update_global_index_clause を使用し、操作中にこれらの索引を更新できます。索引構成表のグローバル索引は主キー・ベースであるため、使用禁止になりません。

選択したパーティションに対応するローカル索引パーティションが削除されます。1 つ以上の吸収パーティションに対応するローカル索引パーティションに、UNUSABLE のマークが付けられます。この索引パーティションは、再構築する必要があります。

drop_table_partition

drop_table_partition 句を使用すると、パーティション表から、*partition* およびそのパーティション内のデータが削除されます。データを表に残したままパーティションを削除する場合は、そのパーティションを隣接するパーティションにマージする必要があります。

参照： 10-50 ページの「*merge_table_partitions*」を参照してください。

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

table が索引構成されており、定義されたマッピング表を持つ場合、対応するマッピング表のパーティションも同様に削除されます。

partition に対応するローカル索引パーティションおよびサブパーティションは、UNUSABLE のマークが付いている場合でも削除されます。

update_global_index_clause を使用し、操作中にヒープ構成表のグローバル索引を更新できます。*update_global_index_clause* とともに *parallel_clause* を指定すると、削除操作ではなく、索引の更新がパラレル化されます。

パーティションを削除し、その後、削除したパーティションに属していた行を挿入した場合、行は 1 つ後のパーティションに格納されます。ただし、そのパーティションが最上位のパーティションである場合、削除したパーティションが表していた値の範囲が表に対して無効になるため、挿入は失敗します。

制限事項：

- ハッシュ・パーティション表のパーティションは削除できません。
- *table* にパーティションが 1 つのみ存在する場合は、このパーティションを削除できません。その表を削除する必要があります。

truncate_table_partition

TRUNCATE PARTITION を指定すると、*partition* からすべての行が削除されます。表がコンポジット・パーティション化されている場合は、*partition* のサブパーティションからすべての行が削除されます。TRUNCATE SUBPARTITION を指定すると、*subpartition* からすべての行が削除されます。*table* が索引構成されている場合、対応するすべてのマッピング表のパーティションおよびオーバーフロー領域のパーティションが切り捨てられます。

切り捨てるパーティションまたはサブパーティションにデータが含まれている場合は、まず、その表の参照整合性制約を使用禁止にする必要があります。また、別の方法として、行を削除してからパーティションを切り捨てる方法もあります。

表に LOB 列がある場合、このパーティションの LOB データおよび LOB 索引セグメントも切り捨てられます。表がコンポジット・パーティション化されている場合、このパーティションのサブパーティションの LOB データおよび LOB 索引セグメントは切り捨てられません。

table でドメイン索引が定義されている場合、索引に IN_PROGRESS または FAILED のマークが付いていると無効になります。また、切り捨てられる表パーティションに対応する索引パーティションに IN_PROGRESS のマークが付いていると無効になります。

切り捨てられるそれぞれのパーティションまたはサブパーティションでは、対応するローカル索引パーティションおよびサブパーティションも切り捨てられます。これらの索引パーティションまたはサブパーティションに UNUSABLE のマークが付いている場合、これらは切り捨てられ、UNUSABLE のマークは VALID にリセットされます。

[*update_global_index_clause*](#) を使用し、操作中に *table* のグローバル索引を更新できます。[*update_global_index_clause*](#) とともに [*parallel_clause*](#) を指定すると、切捨て操作ではなく、索引の更新がパラレル化されます。

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

REUSE STORAGE パーティションまたはサブパーティションに割り当てられ、削除された行から領域を確保するように、REUSE STORAGE を指定します。この領域は、そのパーティションまたはサブパーティションに対する後続の挿入および更新のためにのみ使用できます。

split_table_partition

split_table_partition 句を使用すると、*current_partition* から新しいセグメント、物理属性および初期エクステンツをそれぞれ含む、2つの新しいパーティションが作成されます。*current_partition* に対応付けられたセグメントは、廃棄されます。

制限事項：この句は、ハッシュ・パーティション表に対して指定できません。

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

current_partition がサブパーティションで、新しいパーティションにサブパーティションを指定しない場合、新しいパーティションは *current_partition* のサブパーティションの数および表領域を継承します。

表が索引構成されている場合、対応するマッピング表のパーティションが分割され、親の索引構成表のパーティションと同じ表領域に配置されます。対応するオーバーフロー領域も分割され、新しいオーバーフロー領域にセグメント属性を指定することができます。

対応するローカル索引パーティションに *UNUSABLE* のマークが付いている場合でも、それらは分割されます。

ヒープ構成表のすべてのグローバル索引が無効になります。

update_global_index_clause を使用し、操作中にこれらの索引を更新できます。索引構成表のグローバル索引は主キー・ベースであるため、使用禁止になりません。

選択したパーティションに対応するローカル索引パーティションが削除されます。分割したパーティションに対応するローカル索引パーティションに *UNUSABLE* のマークが付けられ、再構築する必要があります。

table に LOB 列がある場合、*LOB_storage_clause* を使用して、分割の結果生成された LOB データ・セグメントに対して個々の LOB 記憶域属性を指定できます。

current_partition の LOB データおよび LOB 索引セグメントが削除され、新しい表領域を指定しなくても、各 LOB 列または各パーティションに新しいセグメントが作成されます。

AT (*value_list*) *AT (value_list)* 句は、レンジ・パーティションのみに適用されます。新しい2つのパーティションのうちの最初の方に、新しい上限（境界を含まない）を指定します。*value_list* の値は、*current_partition* の元のパーティション境界より小さく、その次に小さいパーティション（そのようなパーティションがある場合）のパーティション境界より大きい値である必要があります。

VALUES (*values_list*) VALUES (*value_list*) 句は、リスト・パーティションのみに適用されます。新しい2つのパーティションのうちの最初に含まれるパーティションの値を指定します。指定したパーティションの値を使用して新しい最初のパーティションが作成され、*current_partition*に残っているパーティションの値を使用して新しい2番目のパーティションが作成されます。新しいパーティションは、指定されていないすべての物理属性を *current_partition* から継承します。

索引に UNUSABLE のマークが付いている場合でも、*table* で定義されている各ローカル索引の対応するパーティションが分割されます。

制限事項： *value_list* には、*current_partition* のすべてのパーティションの値を含むことができません。また、*current_partition* 用の、すでに存在しないすべてのパーティションの値も含むことができません。

INTO *partition_spec, partition_spec* INTO 句によって、分割の結果生成された2つのパーティションを記述します。キーワード PARTITION は、分割の結果生成される2つのパーティションに任意の名前および物理属性を指定しない場合に必須です。新しいパーティション名を指定しない場合、SYS_Pn という形式の名前が割り当てられます。指定しないすべての属性は、*current_partition* から継承されます。

制限事項：

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

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

merge_table_partitions

merge_table_partitions 句を使用すると、*table* の2つのパーティションの内容が1つの新しいパーティションにマージされ、元の2つのパーティションが削除されます。レンジ・パーティションの場合、マージされる2つのパーティションは、隣接している必要があります。リスト・パーティションをマージする場合は、隣接している必要はありません。

制限事項： この句は、ハッシュ・パーティション表に対して指定できません。

新しいパーティションは、元の2つのパーティションのうち、上位のパーティションのパーティション境界を継承します。

segment_attributes_clause に指定されていない属性はすべて、表レベルのデフォルトから継承されます。

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

2つのリスト・パーティションをマージする場合、結果の `partition_value` リストは、マージされる2つの `partition_value` リストの集合を合わせたものです。

ヒープ構成表のすべてのグローバル索引が無効になります。

`update_global_index_clause` を使用し、操作中にこれらの索引を更新できます。索引構成表のグローバル索引は主キー・ベースであるため、使用禁止になりません。

選択したパーティションに対応するローカル索引パーティションが削除されます。マージされたパーティションに対応するローカル索引パーティションに `UNUSABLE` のマークが付けられ、再構築する必要があります。

`partition_level_subpartition` `partition_level_subpartition` 句を使用すると、マージされた新しいパーティションのハッシュ・サブパーティション属性を指定できます。この句に指定されていない属性はすべて、表レベルのデフォルトから継承されます。この句を指定しない場合、マージされた新しいパーティションは、表レベルのデフォルトからサブパーティション属性を継承します。

`parallel_clause` `parallel_clause` を使用すると、マージ操作がパラレル化されます。

`exchange_table_partition`

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

- 1つのハッシュ・パーティション、リスト・パーティションまたはレンジ・パーティション（またはリスト・パーティション以外の1つのハッシュ・サブパーティション）を含む1つの非パーティション表
- コンポジット・パーティション表のレンジ・パーティションのハッシュ・サブパーティションを持つハッシュ・パーティション表

この句をトランスポータブル表領域とともに使用すると、高速データ・ロードが容易になります。

参照： トランスポータブル表領域については、『Oracle9i データベース管理者ガイド』を参照してください。

`table` に LOB 列がある場合、各 LOB 列の LOB データ、および LOB 索引パーティション・セグメントまたは LOB 索引サブパーティション・セグメントは、`table` の対応する LOB データおよび LOB 索引セグメントと交換されます。

2つのオブジェクトのすべてのセグメント属性（表領域およびロギングを含む）も交換されます。

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

交換されるオブジェクトのすべてのグローバル索引は、無効になります。
[update_global_index_clause](#) をこの句とともに指定すると、交換されるパーティションを含む表のグローバル索引は更新されます。交換される表のグローバル索引は、無効のままになります。[update_global_index_clause](#) とともに [parallel_clause](#) を指定すると、交換操作ではなく、索引の更新がパラレル化されます。

参照： 10-53 ページの「[パーティション交換の制限事項](#)」を参照してください。

WITH TABLE パーティションまたはサブパーティションを交換する表を指定します。

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

EXCLUDING INDEXES 交換された表のすべての標準索引、索引パーティションおよびパーティションに対応するすべての索引パーティションまたはサブパーティションに UNUSABLE のマークを付ける場合に、EXCLUDING INDEXES を指定します。

WITH VALIDATION WITH VALIDATION を指定すると、交換された表にあるすべての行が交換されたパーティションまたはサブパーティションにマップされない場合にエラーが戻されます。

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

exceptions_clause 制約に違反するすべての列の ROWID を格納する表を指定します。
schema を指定しない場合、自分のスキーマ内に例外表があるとみなされます。この句自体を指定しない場合、表の名前は EXCEPTIONS になります。例外表は、ローカル・データベース内にある必要があります。

次のいずれかのスクリプトを使用して、EXCEPTIONS 表を作成できます。

- UTLEXCPT.SQL は、物理 ROWID を使用します。そのため、行は、索引構成表からではなく従来表から収集されます（次の注意を参照）。
- UTLEXPT1.SQL は、ユニバーサル ROWID を使用します。そのため、行は、ヒープ構成表と索引構成表の両方から収集されます。

独自の例外表を作成する場合、これら 2 つのスクリプトのいずれかで規定される形式に従う必要があります。

注意： ユニバーサル ROWID ではなく、主キーに基づく索引構成表から例外を収集する場合、各索引構成表に別の例外表を作成し、主キー記憶域を確保する必要があります。スクリプトを変更および再発行することによって、別の名前の例外表を複数作成できます。

参照：

- SQL スクリプトの詳細は、『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の DBMS_IOT パッケージを参照してください。
- 移行行および連鎖行の削除については、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。
- これらのスクリプトの使用の互換性については、『Oracle9i データベース移行ガイド』を参照してください。

exceptions_clause の制限事項

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

これらの条件が当てはまらない場合、この句は無視されます。

参照： 制約を確認する場合の詳細は、11-72 ページの「[constraint_clause](#)」を参照してください。

パーティション交換の制限事項

- 交換される両方の表は同じ主キーを含む必要があり、参照表が空でないかぎり、どちらの表も有効な外部キーを参照できません。
- ハッシュ・パーティション表とコンポジット・パーティション表のレンジ・パーティションの交換時には、次の制限事項があります。
 - ハッシュ・パーティション表のパーティション化キーは、コンポジット・パーティション表のサブパーティション化キーと同じである必要があります。
 - ハッシュ・パーティション表のパーティション数は、コンポジット・パーティション表のレンジ・パーティションのサブパーティション数と同じである必要があります。

- パーティション化された索引構成表に対して、次の追加の制限事項があります。
 - ソースおよびターゲットの表 / パーティションは、その主キーが同じ列に同じ順序で設定されている必要があります。
 - 圧縮が使用可能な場合は、ソースおよびターゲットの両方で使用可能で、接頭辞の長さは同じである必要があります。
 - ソースおよびターゲットの両方は、索引構成されている必要があります。
 - ソースおよびターゲットの両方に、オーバーフロー・セグメントが必要です。または、ソースおよびターゲットの両方がオーバーフロー・セグメントを持ってはいけません。また、ソースおよびターゲットの両方ともマッピング表を含むか、または両方とも含まない必要があります。
 - ソースおよびターゲットの両方は、LOB 列に対して記憶域属性が同一である必要があります。

row_movement_clause

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

制限事項：この句は、パーティション表に対してのみ指定できます。

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

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

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

alter_column_clauses

add_column_options

ADD *add_column_options* を使用すると、表に列または整合性制約を追加できます。

参照： この句のキーワードとパラメータの詳細は、14-6 ページの「[CREATE TABLE](#)」を参照してください。

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

注意： 列にデフォルト値がある場合、DEFAULT 句を使用してデフォルトを NULL に変更することができますが、デフォルト値を完全に削除することはできません。つまり、列に割り当てられたデフォルト値がある場合、USER_TAB_COLUMNS データ・ディクショナリ・ビューの DATA_DEFAULT 列にはデフォルト値と NULL のどちらかが表示されます。

パーティション化された索引構成表の各パーティションには、オーバーフロー・データ・セグメントを追加できます。

非パーティションおよびパーティション表に LOB 列を追加できます。表、およびパーティションまたはサブパーティションのレベルで LOB 記憶域を指定できます。

「SELECT *」構文を使用して、表からすべての列を選択するように指定した問合せを使用してビューを作成した場合、*table* に列を追加しても、新しい列がビューに自動的に追加されることはありません。ビューに新しい列を追加する場合、CREATE VIEW 文に OR REPLACE 句を指定してビューを再作成してください。

参照： 15-37 ページの「[CREATE VIEW](#)」を参照してください。

制限事項：

- クラスタ化表には、LOB 列を追加できません
- LOB 列をハッシュ・パーティション表に追加する場合、新しいパーティションに対して指定できる属性は、TABLESPACE のみです。
- *table* に行がある場合、DEFAULT 句を指定しないかぎり、NOT NULL 制約のある列を追加できません。
- 索引構成表にこの句を指定した場合、同じ文では他の句を指定できません。

DEFAULT 新しい列にデフォルト値を指定する場合、または既存の列に新しいデフォルト値を指定する場合は、**DEFAULT** 句を使用します。後続の **INSERT** 文で列に値を指定しない場合、この値が自動的に割り当てられます。表に新しい列を追加する場合に、デフォルト値を指定した場合、その表のすべての行にデフォルトの列値が挿入されます。

デフォルト値のデータ型は、列に対して指定したデータ型と一致している必要があります。また、列には、このデフォルト値を保持できる長さが必要です。

制限事項：

- **DEFAULT** 式に、他の列、つまり疑似列 **CURRVAL**、**NEXTVAL**、**LEVEL** および **ROWNUM**、または完全には指定されていない日付定数に対する参照を指定することはできません。
- 式には、スカラー副問合せ式を除くすべての書式が可能です。

table_ref_constraint および **column_ref_constraint** これらの句を使用すると、**REF** 型の列について詳細に指定できます。これらの句の違いは、表レベルで **table_ref** を指定することであり、定義する **REF** 列または属性を識別する必要があります。**REF** 型の列または属性を識別した後、**column_ref** を指定してください。

参照： これらの制約の構文および制限事項については、11-72 ページの「[constraint_clause](#)」を参照してください。

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

参照： 制約の型の構文および制限事項については、11-72 ページの「[constraint_clause](#)」を参照してください。

table_or_view_constraint **table_or_view_constraint** を使用すると、表に対する整合性制約を追加または変更できます。

参照： 制約の型の構文および制限事項については、11-72 ページの「[constraint_clause](#)」を参照してください。

modify_column_options

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

datatype 列に対するすべての行が **NULL** の場合、列のデータ型を変更できます。ただし、マテリアライズド・ビューの記憶表にある列のデータ型を変更した場合、それに対応するマテリアライズド・ビューが無効になります。

参照整合性制約の外部キーの一部として、文で列が指定されている場合のみ、データ型を省略できます。参照整合性制約の参照キーに対応する列のデータ型が、その列に自動的に割り当てられます。

すべての列が NULL 値であるかどうかにかかわらず、文字列と未処理列のサイズまたは数値列の精度は、いつでも大きくすることができます。データの変換が必要ない範囲内で、列のデータ型のサイズを縮小できます。既存のデータがスキャンされ、データが新しい最大長を超える場合はエラーが戻されます。

DATE 列を TIMESTAMP または TIMESTAMP WITH LOCAL TIME ZONE に変更できます。すべての TIMESTAMP WITH LOCAL TIME ZONE を DATE 列に変更できます。

注意： TIMESTAMP WITH LOCAL TIME ZONE 列を DATE 列に変更すると、秒の小数部およびタイム・ゾーンで調整されたデータが失われます。

- TIMESTAMP WITH LOCAL TIME ZONE データに秒の小数部がある場合、秒の小数部を丸めて列の行データが更新されます。
 - TIMESTAMP WITH LOCAL TIME ZONE データに 60 以上の分フィールドがある場合（夏時間の規則が切り換えられた場合に境界で発生）、その分フィールドから 60 を引いて列の行データが更新されます。
-
-

表が空の場合、日時列または期間列の先行フィールドまたは秒の小数部を増やすことも減らすこともできます。表が空でない場合、日時列または期間列の先行フィールドまたは秒の小数部を増やすことのみできます。

LONG 列は CLOB または NCLOB 列に、LONG RAW 列は BLOB 列に変更できます。

- 変更された LOB 列は、元の LONG 列で定義されたすべての制約およびトリガーを継承します。すべての制約を変更する場合、後続の ALTER TABLE 文で変更する必要があります。
- ドメイン索引が LONG 列で定義されている場合、列を LOB に変更する前に削除する必要があります。
- 変更後、表のすべての列にある他のすべての索引を再構築する必要があります。

参照：

- LONG および LOB 列の詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。
- LONG から LOB への移行の詳細は、『Oracle9i アプリケーション開発者ガイド - ラージ・オブジェクト』を参照してください。
- 索引の削除および再構築の詳細は、8-48 ページの「ALTER INDEX」を参照してください。

CHAR および VARCHAR2 列の場合、CHAR（元々バイトで指定されていた列に対して文字のセマンティクスを示す）または BYTE（元々文字で指定されていた列に対してバイトのセマンティクスを示す）を指定することによって長さのセマンティクスを変更できます。既存の列の長さを確認するには、ALL_TAB_COLUMNS、USER_TAB_COLUMNS または DBA_TAB_COLUMNS データ・ディクショナリ・ビューの CHAR_USED 列を問い合わせます。

参照：

- バイトおよび文字のセマンティクスの詳細は、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。
- データ・ディクショナリ・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

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

制限事項：

- 列にドメイン索引が定義されている場合は、表の列を変更できません。最初にドメイン索引を削除してから列を変更する必要があります。
- 索引構成表に対して ROWID データ型の列は指定できませんが、UROWID 型の列は指定できます。
- 列のデータ型を REF に変更できません。

参照： マテリアライズド・ビューの再検証の詳細は、8-74 ページの「ALTER MATERIALIZED VIEW」を参照してください。

drop_column_clause

drop_column_clause は、不要になった列を削除したり、将来、システム・リソースへの要求が少なくなったときに削除するようにマークを付けることによって、データベースの領域を解放します。

- ネストした表の列を削除した場合、その記憶表も削除されます。
- LOB 列を削除した場合、LOB データおよび対応する LOB 索引セグメントも削除されます。
- BFILE 列を削除した場合、その列に格納されたロケータのみ削除され、ロケータによって参照されるファイルは削除されません。
- INCLUDING 列として定義した列を削除（または未使用とマーク）する場合は、この列の直前に格納された列が新しい INCLUDING 列になります。

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

UNUSED のマークが付いた列を持つすべて表は、データ・ディクショナリ・ビュー USER_UNUSED_COL_TABS、DBA_UNUSED_COL_TABS および ALL_UNUSED_COL_TABS で表示できます。

参照： データ・ディクショナリ・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

未使用列は削除されたように処理されますが、その列データは表の行に残っています。UNUSED のマークが付けられた列にはアクセスできなくなります。「SELECT *」問合せでも、未使用列からはデータを取り出すことができません。また、UNUSED のマークが付けられた列の名前および型は、DESCRIBE コマンドでは表示されず、未使用列と同じ名前の新しい列を表に追加できます。

注意： これらの列を実際に削除するまでは、表当たり 1000 列の制限に対して、これらの列もカウント対象になります。ただし、DLL 文でこの句の結果をロールバックすることはできません。つまり、SET USED に相当するものを発行しても SET UNUSED 列を取り出すことはできません。

また、LONG 型の列に UNUSED のマークを付けた場合、この未使用の LONG 列を実際に削除しないかぎり、その表には別の LONG 列を追加できません。

参照： 1000 列の制限の詳細は、14-6 ページの「[CREATE TABLE](#)」を参照してください。

DROP 句 表のそれぞれの行から、対象となる列に関連付けられた列記述子およびデータを削除する場合は、DROP を指定します。特定の列を明示的に削除した場合、対象の表にある UNUSED のマークが付いている列もすべて同時に削除されます。

列データを削除した場合、次のものが削除されます。

- 対象の列に定義されているすべての索引。
- 対象の列を参照しているすべての制約。
- 対象の列に統計タイプが関連付けられている場合、その統計タイプを使用して収集したすべての統計情報。FORCE オプションによって、関連付けは解除されます。

注意： 対象の列が他の列の親キーである場合またはチェック制約が対象である列とその他の列を参照している場合は、Oracle はエラーを戻し、CASCADE CONSTRAINTS 句を指定しないかぎり、列を削除しません。この句を指定した場合、対象である列を参照しているすべての制約が削除されます。

参照： 統計タイプの関連付けの解除の詳細は、15-57 ページの「[DISASSOCIATE STATISTICS](#)」を参照してください。

DROP UNUSED COLUMNS 句 表から未使用にマークされている列をすべて削除する場合は、DROP UNUSED COLUMNS を指定します。表の未使用の列からディスク領域を再利用する場合に、この文を使用します。表に未使用の列がない場合でも、エラーは戻されません。

column 未使用として設定または削除する 1 つ以上の列を指定します。列を 1 つのみ指定する場合にかぎり、COLUMN キーワードを使用します。列リストを指定する場合、リストには重複する列を指定できません。

CASCADE CONSTRAINTS 削除する列に定義されている主キーおよび一意キーを参照する参照整合性制約をすべて削除する場合、および削除する列に定義されている複数列制約をすべて削除する場合は、CASCADE CONSTRAINTS を指定します。他の表の列、または対象である表の他の列が参照している制約がある場合は、CASCADE CONSTRAINTS を指定します。CASCADE CONSTRAINTS を指定しない場合、その文は異常終了し、エラーが戻されます。

INVALIDATE INVALIDATE キーワードはオプションです。ビュー、トリガー、ストアド・プログラム・ユニットなどのすべての依存オブジェクトが自動的に無効になります。オブジェクトの無効化は再帰的プロセスです。したがって、すべての直接的な依存オブジェクトおよび間接的な依存オブジェクトが無効になります。ただし、Oracle では、リモート依存性をローカル依存性と別に管理しているため、無効になるのはローカル依存性のみです。

この文によって無効となったオブジェクトは、次に参照された際に自動的に有効になります。オブジェクトを参照する前に、オブジェクトに存在するエラーは、すべて修正しておく必要があります。

参照： 依存性の詳細は、『Oracle9i データベース概要』を参照してください。

CHECKPOINT CHECKPOINT を指定すると、*integer* 行を処理した後に DROP COLUMN 操作のチェックポイントが適用されます。このとき *integer* は任意の値であり、1 以上である必要があります。*integer* が表の行数より大きい場合、すべての行が処理された後にチェックポイントが適用されます。*integer* を指定しない場合、デフォルトの 512 が設定されます。チェックポイントは、DROP COLUMN 操作中に蓄積された UNDO ログの量を削減し、ロールバック・セグメント領域外での実行を防ぎます。ただし、チェックポイントが適用された後にこの文が中断された場合、表は使用禁止の状態のままになります。表が使用できない間、その表に対して実行可能な操作は、DROP TABLE、TRUNCATE TABLE および ALTER TABLE DROP COLUMNS CONTINUE（後述）のみです。

この句は列データを削除しないため、SET UNUSED と同時に使用できません。

DROP COLUMNS CONTINUE 句 中断されたところから列削除操作を続ける場合は、DROP COLUMNS CONTINUE を指定します。表が有効な状態にあるときにこの文を発行すると、エラーになります。

drop_column_clause の制限事項

- この句の各部分は、文の中で 1 回のみ指定でき、他の ALTER TABLE 句と一緒に使用できません。たとえば、次のような文は許可されません。

```
ALTER TABLE t1 DROP COLUMN f1 DROP (f2);
ALTER TABLE t1 DROP COLUMN f1 SET UNUSED (f2);
ALTER TABLE t1 DROP (f1) ADD (f2 NUMBER);
ALTER TABLE t1 SET UNUSED (f3)
    ADD (CONSTRAINT ck1 CHECK (f2 > 0));
```

- オブジェクト型の列は、エンティティとしてのみ削除できます。オブジェクト型の列から属性を削除するには、ALTER TYPE ... DROP ATTRIBUTE 文を CASCADE INCLUDING TABLE DATA 句とともに使用します。属性の削除は、すべての依存オブジェクトに影響することに注意してください。詳細は、11-15 ページの「[DROP ATTRIBUTE](#)」を参照してください。
- 索引構成表からは、主キー列でない場合にかぎり、列を削除できます。索引構成表の主キー制約は絶対に削除できないため、CASCADE CONSTRAINTS を指定しても主キー列は削除できません。
- 削除した列または未使用の列で表をエクスポートできます。ただし、エクスポート・ファイルに指定されたすべての列が表に存在する（これらの列のいずれも削除または未使用のマークを付けられていない）場合のみ、その表をインポートできます。そうでない場合は、エラーが戻ります。
- ドメイン索引が構築されている列は削除できません。

- 次のものは、この句を使用して削除できません。
 - 擬似列、クラスタ列、またはパーティション列（パーティションが作成されたすべての表領域がオンラインで読取り / 書込みモードである場合、パーティション表から非パーティション列を削除できます）。
- ネストした表の列、オブジェクト表の列または SYS が所有する表の列。

modify_collection_retrieval

modify_collection_retrieval を使用すると、データベースからコレクション型の項目が取り出されたときの戻り値を変更できます。

collection_item ネストした表型または VARRAY 型の列修飾属性の名前を指定します。

RETURN AS 問合せの結果として何を戻り値とするかを指定します。

- LOCATOR は、ネストした表に対して一意のロケータを戻すことを指定します。
- VALUE は、ネストした表のコピーをそのまま戻すことを指定します。

alter_constraint_clauses

alter_constraint_clauses を使用すると、制約の状態を変更したり、制約を削除できます。

MODIFY CONSTRAINT *constraint*

MODIFY CONSTRAINT を使用すると、既存の制約の状態を変更できます。

参照： *constraint_state* のすべてのキーワードとパラメータの詳細は、11-72 ページの「[constraint_clause](#)」を参照してください。

制約の変更の制限事項

- 表索引のパーティション化キーまたはサブパーティション化キーの一部である列の長さまたはデータ型は変更できません。
- CHAR 型の列を VARCHAR2（または VARCHAR）型に変更または VARCHAR2（または VARCHAR）型の列を CHAR 型に変更できるのは、列のすべての行が NULL 値である場合、または列のサイズを変更しない場合のみです。
- クラスタの一部である場合は、LONG または LONG RAW 列を LOB に変更できません。LONG または LONG RAW 列を LOB に変更する場合は、同じ ALTER TABLE 文では、DEFAULT 句および *LOB_storage_clause* 以外は指定できません。

- LONG または LONG RAW 列を LOB に変更する場合のみ、*LOB_storage_clause* を *modify_column_options* の一部として指定できます。
- 索引構成表にこの句を指定した場合、同じ文では他の句を指定できません。

drop_constraint_clause

データベースの整合性制約を削除する場合は、*drop_constraint_clause* を使用します。制約の適用を中止し、データ・ディクショナリから制約が削除されます。

各 *drop_constraint_clause* には、制約を 1 つのみ指定できますが、1 つの文の中では、複数の *drop_constraint_clauses* を指定できます。

PRIMARY KEY PRIMARY KEY を指定すると、表の主キー制約を削除できます。

UNIQUE UNIQUE を指定すると、指定した列の一意制約を削除できます。

注意： ビットマップ結合索引が定義されている列から主キー制約または一意制約を削除すると、索引は無効になります。ビットマップ結合索引の詳細は、12-58 ページの「[CREATE INDEX](#)」を参照してください。

CONSTRAINT CONSTRAINT *constraint* を指定すると、主キー制約または一意制約以外の整合性制約を削除できます。

CASCADE 削除する整合性制約に依存するその他の整合性制約もすべて削除する場合は、CASCADE を指定します。

KEEP | DROP INDEX KEEP または DROP INDEX を指定すると、Oracle が主キーまたは一意制約の適用に使用する索引を残すか削除するかを指定できます。

drop_constraint_clause の制限事項

- 参照整合性制約の一部の主キー制約または一意制約は、外部キーを削除しないと削除できません。参照されたキーと外部キーをともに削除する場合は、CASCADE 句を使用してください。CASCADE を省略すると、いずれかの外部キーが主キー制約または一意制約を参照している場合は、それらは削除されません。
- 主キーをオブジェクト識別子 (OID) として使用している表では、主キー制約は (CASCADE 句を使用しても) 削除できません。
- REF 列の参照整合性制約を削除した場合、REF 列の有効範囲には参照先の表が含まれたままになります。
- 列の有効範囲は削除できません。

alter_column_properties

alter_column_properties 句を使用すると、列またはパーティションの記憶特性を変更できます。

column_properties

column_properties を使用して、オブジェクト、ネストした表、VARRAY または LOB 列の記憶特性を指定します。

object_type_col_properties ***object_type_col_properties*** を使用すると、オブジェクト列、オブジェクト属性、あるいはコレクション列またはコレクション属性の要素の記憶特性を指定できます。

column ***column*** には、オブジェクト列またはオブジェクト属性を指定します。

substitutable_column_clause ***substitutable_column_clause*** を使用すると、同じ階層のオブジェクト列または属性が互いに置換可能かどうかを指定できます。列が特定の型であるか、サブタイプのインスタンスを含むかどうか、またはその両方を指定できます。

- ELEMENT を指定すると、コレクション列またはコレクション属性の要素型を宣言された型のサブタイプに制約できます。
- IS OF [TYPE] (ONLY type) 句を指定すると、オブジェクト列の型を宣言された型のサブタイプに制約できます。
- NOT SUBSTITUTABLE AT ALL LEVELS を指定すると、オブジェクト列がサブタイプに対応するインスタンスを持つことはできません。また、置換は、埋込みオブジェクト属性および埋込みのネストした表と VARRAY の要素には使用禁止です。デフォルトは、SUBSTITUTABLE AT ALL LEVELS です。

***substitutable_column_clause* の制限事項**

- この句は、最上位のオブジェクト列に指定できますが、オブジェクト列の属性には指定できません。
- コレクション型の列の場合、この句で指定できる部分は [NOT] SUBSTITUTABLE AT ALL LEVELS のみです。

nested_table_col_properties *nested_table_col_properties* 句を使用すると、ネストした表に対して別の記憶特性を指定し、そのネストした表を索引構成表として定義できます。ネストした表の型を持つ列または列属性付きで表を作成する場合は、この句を挿入する必要があります（この句の中で、親オブジェクト表に対する場合と同じ働きをする句は、ここでは繰り返されません）。

- *nested_item* には、型がネストした表である列（または、その表のオブジェクト型の最上位の属性）の名前を指定します。
- *storage_table* には、*nested_item* の行を含む表の名前を指定します。記憶表は、親表と同じスキーマ、および親表と同じ表領域内に作成されます。

制限事項：

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

varray_col_properties *varray_col_properties* を使用すると、VARRAY 型のデータが格納されている LOB に対して、別の記憶特性を指定できます。この句を指定する場合、インラインに格納できるほど小さい値でも、VARRAY は必ず LOB に格納されます。

制限事項：VARRAY 列に対し、*LOB_parameters* の一部として TABLESPACE を指定することはできません。VARRAY に対する LOB 表領域は、デフォルトではそれを格納する表の表領域になります。

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

CACHE READS 句 CACHE READS は LOB 記憶域にのみ適用されます。LOB 値が書き込み操作中ではなく読み込み操作中にバッファ・キャッシュに入れられることを指定します。

- *lob_item* には、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 LOB 列名または LOB オブジェクト属性を指定します。
- *lob_segname* には、LOB データ・セグメントの名前を指定します。*lob_item* が複数指定されている場合は、*lob_segname* を使用できません。

新しい LOB 列を追加するときに、CACHE READS でロギング属性を指定できます。作成時に LOB 列を定義するときにも指定できます。

CACHE または NOCACHE から CACHE READS へ、または CACHE READS から CACHE または NOCACHE へ LOB 列を変更するときに、ロギング属性を変更できます。LOGGING または NOLOGGING を指定しない場合、LOB 列の現行のロギング属性がデフォルトになります。

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

制限事項：

- ハッシュ・パーティションまたはハッシュ・サブパーティションに指定できる *LOB_parameters* のパラメータは、TABLESPACE のみです。
- *table* がパーティション化されている場合、*LOB_index_clause* を指定できません。

ENABLE | DISABLE STORAGE IN ROW LOB 値をインライン（行内）またはアウトライン（行外）のどちらに格納するかを指定します（LOB 値が格納されている場所にかかわらず、LOB ロケータは、常に、インラインに格納されます）。

- ENABLE は、LOB 値の長さが、約 4000 バイトからシステム制御情報分を引いた長さより小さい場合、LOB 値をインラインに格納することを指定します。これはデフォルトです。
- DISABLE を指定すると、LOB 値の長さにかかわらず、LOB 値はアウトラインに格納されます。

制限事項： STORAGE IN ROW は、一度設定すると変更できません。したがって、この句は *modify_column_options* 句の一部には指定できません。ただし、新しい列を追加するとき (*add_column_options*)、または表を移動するとき (*move_table_clause*) に、この設定を変更することはできます。

CHUNK integer LOB の操作用に割り当てるバイト数を指定します。*integer* にデータベースのブロック・サイズの倍数を指定しなかった場合、自動的に次に大きい倍数（バイト単位）に切り上げられます。たとえば、データベース・ブロック・サイズが 2048 の場合、*integer* に 2050 を指定すると、4096 バイト（2 ブロック）が割り当てられます。最大値は 32768（32KB）で、これが Oracle で許容されるブロック・サイズの最大値です。デフォルトの CHUNK サイズは、Oracle での 1 データベース・ブロックです。

制限事項：

- CHUNK の値は、一度設定すると変更できません。
- CHUNK の値は、NEXT の値（デフォルト値または記憶域句で指定した値）以下である必要があります。CHUNK の値が NEXT の値を超えると、エラーが戻ります。

PCTVERSION integer LOB の記憶領域全体のうち、新規バージョンの LOB の作成に使用される最大記憶領域の割合（パーセント）を指定します。デフォルト値は 10 です。これは、LOB の記憶領域全体の 10% が使用されるまで以前のバージョンの LOB データが上書きされないことを意味します。

LOB_index_clause この句は、Oracle8i 以降は使用されません。Oracle は各 LOB 列に対して索引を生成します。LOB 索引は、システムによって名前が付けられ管理されており、LOB データ・セグメントと同じ表領域に格納されます。

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

参照： 以前のバージョンから移行された表の LOB 索引の管理方法については、『Oracle9i データベース移行ガイド』を参照してください。

partition_storage_clause

partition_storage_clause を使用すると、各パーティションに、別の *LOB_storage_clause* または *varray_col_properties* を指定できます。パーティションは、その位置の順に指定してください。パーティションの順番は、USER_IND_PARTITIONS ビューの PARTITION_NAME および PARTITION_POSITION 列の問合せから得ることができます。

特定のパーティションに、*LOB_storage_clause* または *varray_col_properties* 句を指定しなかった場合、表レベルで LOB 項目に指定された記憶特性が設定されます。表のレベルでも LOB 項目に記憶特性を指定しなかった場合、LOB データ・パーティションは、対応する表パーティションと同じ表領域に格納されます。

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

modify_lob_storage_clause

modify_lob_storage_clause を使用すると、*lob_item* の物理属性を変更できます。各 *modify_lob_storage_clause* に対して、*lob_item* を 1 つのみ指定できます。

制限事項：

- LOB 記憶域属性を変更する場合、*storage_clause* の INITIAL パラメータの値は変更できません。
- 同じ文で *allocate_extent_clause* と *deallocate_unused_clause* の両方を指定することはできません。

alter_varray_col_properties

alter_varray_col_properties を使用すると、VARRAY が格納されている既存の LOB の記憶特性を変更できます。

制限事項：*LOB_parameters* の TABLESPACE 句は、この句の一部として指定できません。VARRAY に対する LOB 表領域のデフォルトは、表を含む表領域になります。

alter_external_table_clause

*alter_external_table_clause*を使用すると、外部表の特性を変更できます。この句は、外部データには影響しません。*parallel_clause*、*enable_disable_clause*、*external_data_properties*および REJECT LIMIT 句の構文およびセマンティクスは、CREATE TABLE の内容と同じです。14-29 ページの「CREATE TABLE」の「*external_table_clause*」を参照してください。

制限事項：

- この句以外の句を使用して外部表を変更できません。
- LONG、LOB またはオブジェクト型の列は外部表に追加できません。また、外部表の列のデータ型を、これらの型に変更できません。
- 制約は外部表に追加できません。
- 外部表の記憶域パラメータは変更できません。

move_table_clause

*move_table_clause*によって、非パーティション表のデータを新しいセグメントに再配置できます。オプションとして、別の表領域への配置および記憶域属性の変更を行うこともできます。

*LOB_storage_clause*および *varray_col_properties* 句を使用して、その表に関連付けられた LOB データ・セグメントを移動することもできます。この句で指定していない LOB 項目は移動できません。

index_org_table_clause

索引構成表の場合は、*index_org_table_clause* 構文を使用すると、オーバーフロー・セグメント属性も指定できます。*move_table_clause*を使用すると、索引構成表の主キー索引が再構築されます。オーバーフロー・データ・セグメントは、キーワード OVERFLOW が明示的に指定されていないかぎり、再構築されません。ただし、次の場合は例外です。

- ALTER TABLE 文の一部として PCTTHRESHOLD の値または INCLUDING 列を変更する場合は、オーバーフロー・データ・セグメントが再構築されます。
- 索引構成表内のアウトラインの列（LOB 列、VARRAY 列、ネストした表の列）のいずれかを明示的に移動する場合は、オーバーフロー・データ・セグメントも再構築されます。

LOB 列の索引およびデータ・セグメントは、LOB 列を ALTER TABLE 文の一部として明示的に指定しないかぎり、再構築されません。

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

制限事項：

- 同じ文でこの句と他の句は結合できません。
- この句は、パーティション化されていない索引構成表に対してのみ指定できます。
- オンライン MOVE 中のパラレル DML はサポートされていません。**ONLINE** を指定し、パラレル DML 文を発行すると、Oracle はエラーを戻します。

mapping_table_clauses マッピング表が存在していない場合に Oracle にマッピング表を作成させるには、**MAPPING TABLE** を指定します。マッピング表がすでに存在する場合、マッピング表は索引構成表とともに移動され、すべてのビットマップ索引には **UNUSABLE** のマークが付けられます。新しいマッピング表は、親表と同じ表領域に作成されます。

NOMAPPING を指定すると、既存のマッピング表が削除されます。

制限事項：表でビットマップ索引が定義されている場合は、**NOMAPPING** を指定できません。

参照： 14-28 ページの「**CREATE TABLE**」の「[mapping_table_clause](#)」を参照してください。

compression_clause *compression_clause* を使用すると、索引構成表のキー圧縮を使用可能または使用禁止にできます。

- **COMPRESS** は、キー圧縮を使用可能にします。これによって、索引構成表の主キー列の値が重複しなくなります。*integer* を使用して、接頭辞の長さ（圧縮する接頭辞列数）を指定します。

接頭辞の長さの有効範囲は、1 ～（主キー列数 -1）までです。デフォルトでは（主キー列数 -1）になります。

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

TABLESPACE *tablespace* 再構築した索引構成表を格納する表領域を指定します。

move_table_clause の制限事項

- **MOVE** を指定する場合は、最初の句にする必要があります。この句以外では、*physical_attributes_clause*、*parallel_clause* および *LOB_storage_clause* のみが指定できます。
- **LONG** または **LONG RAW** 列を含む表は、移動できません。
- パーティション表（ヒープ表または索引構成表）全体の移動はできません。個々のパーティションまたはサブパーティションを移動してください。

参照： 10-43 ページの「[move_table_partition](#)」および 10-44 ページの「[move_table_subpartition](#)」を参照してください。

LOB についての注意

`move_table_clause` で指定するすべての LOB 列については、次のことに注意してください。

- 新しい表領域が指定されていない場合でも、古い LOB データ・セグメントとこれに対応する索引セグメントは削除され、新しいセグメントが作成されます。
 - 表の LOB 索引がその LOB データと異なる表領域にある場合、移動の後、LOB 索引は LOB データと一緒に LOB データの表領域にまとめて格納されます。
-
-

`enable_disable_clause`

`enable_disable_clause` を使用すると、Oracle が整合性制約を適用するかどうか、およびその方法を指定できます。DROP および KEEP 句は、一意制約または主キー制約を使用禁止にする場合のみに有効です。

参照：

- この文に関連する注意および制限事項については、14-44 ページの「CREATE TABLE」の「[enable_disable_clause](#)」を参照してください。
- 制約を適用する索引の使用については、11-86 ページの「`constraint_clause`」の「[using_index_clause](#)」および「[global_partitioned_index](#)」句を参照してください。

TABLE LOCK

DDL 操作中にロックされた表にのみ DDL 操作を実行できます。このような表ロックは、DML 操作中には必要ありません。

注意： 一時表に表ロックを適用することはできません。

ENABLE TABLE LOCK `ENABLE TABLE LOCK` を指定して表ロックを有効にすることによって、表に対する DDL 操作が可能になります。

DISABLE TABLE LOCK `DISABLE TABLE LOCK` を指定して表ロックを無効にすることによって、表に対する DDL 操作が不可能になります。

ALL TRIGGERS

`ALL TRIGGERS` 句を使用すると、表に関連するすべてのトリガーを使用可能または使用禁止にできます。

ENABLE ALL TRIGGERS `ENABLE ALL TRIGGERS` を指定して、表に関連するすべてのトリガーを使用可能にします。トリガー条件が満たされた場合に、トリガーが起動されます。

単一トリガーを使用可能にする場合は、`ALTER TRIGGER` の *enable_clause* を使用してください。

参照： 14-77 ページの「[CREATE TRIGGER](#)」および 11-2 ページの「[ALTER TRIGGER](#)」を参照してください。

DISABLE ALL TRIGGERS `DISABLE ALL TRIGGERS` を指定して、表に関連するすべてのトリガーを使用禁止にします。トリガー条件が満たされた場合でも、使用禁止のトリガーは起動されません。

例

コレクションの取出し例 次の文は、デモ表 `sh.print_media` のネストした表の列 `ad_textdocs_ntab` を変更し、問合せ時にロケータのかわりに実値を戻します。

```
ALTER TABLE print_media MODIFY NESTED TABLE ad_textdocs_ntab
RETURN AS VALUE;
```

PARALLEL の例 次の文は、デモ表 `oe.customers` への問合せに対してパラレル処理を指定します。

```
ALTER TABLE customers
PARALLEL;
```

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

```
ALTER TABLE employees
ENABLE VALIDATE CONSTRAINT emp_manager_fk
EXCEPTIONS INTO except_table;
```

Oracle が制約を使用可能にするためには、employees 表の各行がこの制約を満たしている必要があります。制約に違反する行があれば、制約は使用禁止のままになります。すべての例外は、except_table 表の中に表示されます。次の文で、employees 表の例外を検出することもできます。

```
SELECT employees.*
FROM employees e, except_table ex
WHERE e.row_id = ex.row_id
      AND ex.table_name = 'EMPLOYEES'
      AND ex.constraint = 'EMP_MANAGER_FK';
```

索引構成表の EXCEPTIONS INTO の例 次の例は、主キー制約に違反する索引構成表 hr.countries からの行を保持する except_table 表を作成します。

```
EXECUTE DBMS_IOT.BUILD_EXCEPTIONS_TABLE ('hr', 'countries', 'except_table');
```

```
ALTER TABLE countries
  ENABLE PRIMARY KEY
  EXCEPTIONS INTO except_table;
```

例外表を指定する場合は、この表に行を挿入する権限が必要です。検出された例外を調べる場合、例外表を問い合わせる権限が必要です。

参照：

- 16-56 ページの「[INSERT](#)」を参照してください。
- 表に行を挿入するために必要な権限の詳細は、17-4 ページの「[SELECT](#)」を参照してください。

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

```
ALTER TABLE employees
  ENABLE NOVALIDATE PRIMARY KEY
  ENABLE NOVALIDATE CONSTRAINT emp_last_name_nn;
```

この文には、次の 2 つの ENABLE 句が含まれています。

- 1 番目の ENABLE 句は、表の主キー制約を ENABLE NOVALIDATE 状態にします。
- 2 番目の ENABLE 句は、emp_last_name_nn という制約を ENABLE NOVALIDATE 状態にします。

この例では、表のそれぞれの行が 2 つの制約を満たす場合にかぎり、その制約が使用可能になります。どちらかの制約に違反する行があった場合、エラーが戻され、どちらの制約も使用禁止のままになります。

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

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

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

チェック制約の例 次の文は、employees 表にチェック制約を定義し、その制約を使用禁止にします。

```
ALTER TABLE employees ADD CONSTRAINT check_comp
  CHECK (salary + (commission_pct*salary) <= 5000)
  DISABLE CONSTRAINT check_comp;
```

check_comp 制約は、給与総額が 5000 ドルを超える従業員がいないことを保証します。ただし、この制約が使用禁止になっているため、従業員の給与をこの制限以上に増やすことができます。

トリガーを使用可能にする例 次の文は、employees 表に対応付けられているすべてのトリガーを使用可能にします。

```
ALTER TABLE employees
  ENABLE ALL TRIGGERS;
```

DEALLOCATE UNUSED の例 次の文は、employees 表で再利用できるように最高水位標が MINEXTENTS を超えるすべての未使用領域を解放します。

```
ALTER TABLE employees
  DEALLOCATE UNUSED;
```

DROP COLUMN の例 次の文は、CASCADE CONSTRAINTS が指定されている drop_column_clause です。表 t1 が次のように作成されているとします。

```
CREATE TABLE t1 (
  pk NUMBER PRIMARY KEY,
  fk NUMBER,
  c1 NUMBER,
  c2 NUMBER,
  CONSTRAINT ri FOREIGN KEY (fk) REFERENCES t1,
  CONSTRAINT ck1 CHECK (pk > 0 and c1 > 0),
  CONSTRAINT ck2 CHECK (c2 > 0)
);
```

次の文に対してエラーが戻されます。

```
ALTER TABLE t1 DROP (pk); -- pk is a parent key
ALTER TABLE t1 DROP (c1); -- c1 is referenced by multicolumn
                           constraint ck1
```

次の文を発行すると、列 `pk`、主キー制約、外部キー制約 `ri` およびチェック制約 `ck1` が削除されます。

```
ALTER TABLE t1 DROP (pk) CASCADE CONSTRAINTS;
```

削除された列に定義した制約が参照する列もすべて削除される場合、`CASCADE CONSTRAINTS` は必要ありません。たとえば、他の表から列 `pk` を参照する他の参照制約が存在していないとします。この場合は、`CASCADE CONSTRAINTS` 句を指定しない次の文が有効になります。

```
ALTER TABLE t1 DROP (pk, fk, c1);
```

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

```
ALTER TABLE countries INITTRANS 4;
```

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

```
ALTER TABLE countries ADD OVERFLOW;
```

次の文は、索引構成表 `countries` のオーバーフロー・データ・セグメントの `INITTRANS` パラメータを変更します。

```
ALTER TABLE countries OVERFLOW INITTRANS 4;
```

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

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

パーティション `p3` の列 `b` および列 `d` のLOBデータおよびLOB索引セグメントは、表領域 `tsz` に格納されます。LOB列の他の属性は、まず表レベルのデフォルトから継承され、次に表領域のデフォルトから継承されます。

列 `c` のLOBデータ・セグメントは `mylobseg` セグメントに格納され、他のすべての属性は、まず表レベルのデフォルトから継承され、次に表領域のデフォルトから継承されます。

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

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

パーティション `p3_1` では、列 `b` と列 `d` の LOB セグメントが表領域 `tsz` 内に作成されます。パーティション `p3_2` では、列 `c` の LOB セグメントが表領域 `ts5` 内に作成されます。パーティション `p3_2` 内の列 `b` と列 `d` の LOB セグメント、およびパーティション `p3_1` 内の列 `c` の LOB セグメントは、元のパーティション `p3` の元の表領域内に残ります。ただし、LOB データおよび LOB 索引セグメントが新しい表領域に移動されない場合でも、これらの新しいセグメントが作成されます。

UPDATE GLOBAL INDEXES の例 次の文は、デモ表 `sh.sales` のパーティション `sales_q1_2000` を分割し、定義されているグローバル索引を更新します。

```
ALTER TABLE sales SPLIT PARTITION sales_q1_2000
  AT (TO_DATE('16-FEB-2000','DD-MON-YYYY'))
  INTO (PARTITION q1a_2000, PARTITION q1b_2000)
  UPDATE GLOBAL INDEXES;
```

ユーザー定義オブジェクト識別子の例 次の文は、オブジェクト型、主キーに基づくオブジェクト識別子に対応するオブジェクト表、およびユーザー定義 REF 列を持つ表を作成します。

```
CREATE TYPE emp_t AS OBJECT (empno NUMBER, address CHAR(30));

CREATE TABLE emp OF emp_t (
  empno PRIMARY KEY)
  OBJECT IDENTIFIER IS PRIMARY KEY;

CREATE TABLE dept (dno NUMBER, mgr_ref REF emp_t SCOPE is emp);
```

次の文は、`emp` 表を参照する制約およびユーザー定義 REF 列を追加します。

```
ALTER TABLE dept ADD CONSTRAINT mgr_cons FOREIGN KEY (mgr_ref)
  REFERENCES emp;
ALTER TABLE dept ADD sr_mgr REF emp_t REFERENCES emp;
```

列の追加例 次の文は、NUMBER データ型の列 `duty_pct`、サイズが 3 の VARCHAR2 データ型の列 `visa_needed` (「yes」または「no」を格納) およびチェック整合性制約を追加します。

```
ALTER TABLE countries
  ADD (duty_pct      NUMBER(2,2)  CHECK (duty_pct < 10.5),
       visa_needed   VARCHAR2(3));
```

列の変更例 次の文は、`duty_pct` 列のサイズを増やします。

```
ALTER TABLE countries
  MODIFY (duty_pct NUMBER(3,2));
```

MODIFY 句には列の定義が 1 つのため、定義を囲むカッコは任意指定です。

次の文は、`employees` 表の PCTFREE パラメータと PCTUSED パラメータの値を、それぞれ 30 と 60 に変更します。

```
ALTER TABLE employees
  PCTFREE 30
  PCTUSED 60;
```

LONG から LOB への変更例 次の例は、デモ表 `pm.print_media` の `press_release` 列のデータ型を LONG から CLOB に変更します。

```
ALTER TABLE print_media MODIFY (press_release CLOB);
```

ALLOCATE EXTENT の例 次の文は、`employees` 表に 5KB のエクステントを割り当て、そのエクステントをインスタンス 4 が使用できるようにします。

```
ALTER TABLE employees
  ALLOCATE EXTENT (SIZE 5K INSTANCE 4);
```

この文は、DATAFILE パラメータを指定していないため、エクステントは `employees` 表が入っている表領域に属するデータ・ファイルの 1 つに割り当てられます。

デフォルト列値の例 次の文は、`product_information` 表の `min_price` 列のデフォルト値を 10 に変更します。

```
ALTER TABLE product_information
  MODIFY (min_price DEFAULT 10);
```

続いて `product_information` 表に値を指定せずに新しい列 `min_price` を追加する場合、`min_price` 列の値は自動的に 0 (ゼロ) になります。

```
INSERT INTO product_information (product_id, product_name,
    list_price)
VALUES (300, 'left-handed mouse', 40.50);
```

```
SELECT product_id, product_name, list_price, min_price
FROM product_information
WHERE product_id = 300;
```

PRODUCT_ID	PRODUCT_NAME	LIST_PRICE	MIN_PRICE
300	left-handed mouse	40.5	0

以前に指定したデフォルト値を中止して、新しく追加する行にその値が自動的に挿入されないようにする場合、次の文に示すように、デフォルト値を `NULL` に置き換えます。

```
ALTER TABLE product_information
MODIFY (min_price DEFAULT NULL);
```

`MODIFY` 句には、列の定義をすべて指定する必要はありません。列名および変更部分のみを指定してください。この文は、既存の行の既存の値には影響しません。

制約の削除例 次の文は、`departments` 表の主キーを削除します。

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

`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 departments
DROP CONSTRAINT pk_dept CASCADE;
```

`CASCADE` 句によって、主キーを参照するすべての外部キーが削除されます。

次の文は、`employees` 表の `email` 列の一意キーを削除します。

```
ALTER TABLE employees
DROP UNIQUE (email);
```

この文の `DROP` 句では `CASCADE` 句を省略します。`CASCADE` オプションを省略することによって、一意キーを参照する外部キーがある場合、その一意キーは削除されません。

LOB の例 次の文は、CLOB 列の `resume` を `employees` 表に追加し、新しい列の LOB 記憶特性を指定します。

```
ALTER TABLE employees ADD (resume CLOB)
  LOB (resume) STORE AS resume_seg (TABLESPACE resume_ts);
```

次の文を入力して、キャッシュを使用できるように LOB 列の `resume` を変更します。

```
ALTER TABLE employees MODIFY LOB (resume) (CACHE);
```

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

```
ALTER TABLE employees ADD (skills skill_table_type)
  NESTED TABLE skills STORE AS nested_skill_table;
```

また、ネストした表の記憶特性も変更できます。変更する場合、`nested_table_col_properties` に指定した記憶表の名前を使用してください。記憶表では、DML 文の問合せまたは実行はできません。記憶表は、ネストした表の列の記憶特性を変更するためにのみ使用します。

次の文は、ネストした表の列 `client` と記憶表 `client_tab` を使用して、表 `vetservice` を作成します。ネストした表 `vetservice` を変更して制約を指定します。

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

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

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

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

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

次の文は、REF 値のネストした表用の記憶表を変更して、REF の範囲が限定されることを指定します。

```
CREATE TYPE emp_t AS OBJECT (eno number, ename char(31));
CREATE TYPE emps_t AS TABLE OF REF emp_t;
CREATE TABLE emptab OF emp_t;
CREATE TABLE dept (dno NUMBER, employees emps_t)
  NESTED TABLE employees STORE AS deptemps;
ALTER TABLE deptemps ADD (SCOPE FOR (column_value) IS emptab);
```

同様に、次の文は、REF を ROWID とともに格納することを指定します。

```
ALTER TABLE deptemps ADD (REF(column_value) WITH ROWID);
```

これらの ALTER TABLE 文を正確に実行するためには、記憶表 deptemps が空である必要があります。また、ネストした表は、スカラー (REF) 表として定義されるため、Oracle は、暗黙的に列名 COLUMN_VALUE を記憶表に設定します。

参照：

- ネストした表の記憶域の詳細は、14-6 ページの「[CREATE TABLE](#)」を参照してください。
- ネストした表の詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

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

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

オブジェクト表 DEPARTMENTS を次のように作成します。

```
CREATE TABLE departments OF dept_t;
```

dept 列は、任意の表に格納された dept_t のオブジェクトに参照を格納できます。次のように dept 列に有効範囲制約を追加することによって、departments 表に格納されたオブジェクトのみが参照されるように制限できます。

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

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

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

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

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

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

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

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

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

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

パーティションの変更例 次の文は、sales 表のパーティション nov96 に対応するすべてのローカル索引パーティションに、UNUSABLE のマークを付けます。

```
ALTER TABLE sales MODIFY PARTITION nov96
  UNUSABLE LOCAL INDEXES;
```

次の文は、UNUSABLE のマークが付けられたすべてのローカル索引パーティションを再構築します。

```
ALTER TABLE sales MODIFY PARTITION jan97
  REBUILD UNUSABLE LOCAL INDEXES;
```

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

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

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

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

パーティションの名前の変更例 次の文は、表の名前を変更します。

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

次の文では、パーティション emp3 の名前が変更されます。

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

パーティションの分割例 次の文は、古いパーティション depot4 を分割して 2 つの新しいパーティションを作成し、1 つには depot9 という名前を付け、もう 1 つには旧パーティションの名前を再利用します。

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

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

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

追加の例 ALTER TABLE 文を使用した整合性制約の定義の例は、11-72 ページの「[constraint_clause](#)」を参照してください。

表の記憶域パラメータの値を変更する例は、17-50 ページの「[storage_clause](#)」を参照してください。

ALTER TABLESPACE

用途

ALTER TABLESPACE を使用すると、既存の表領域、1 つ以上のデータ・ファイルまたはテンポラリ・ファイルを変更できます。

参照： 表領域作成の詳細は、『Oracle9i データベース管理者ガイド』および 14-62 ページの「[CREATE TABLESPACE](#)」を参照してください。

前提条件

ALTER TABLESPACE システム権限を持っている場合、この文のすべての操作を実行できます。MANAGE TABLESPACE システム権限を持っている場合は、次の操作のみを実行できます。

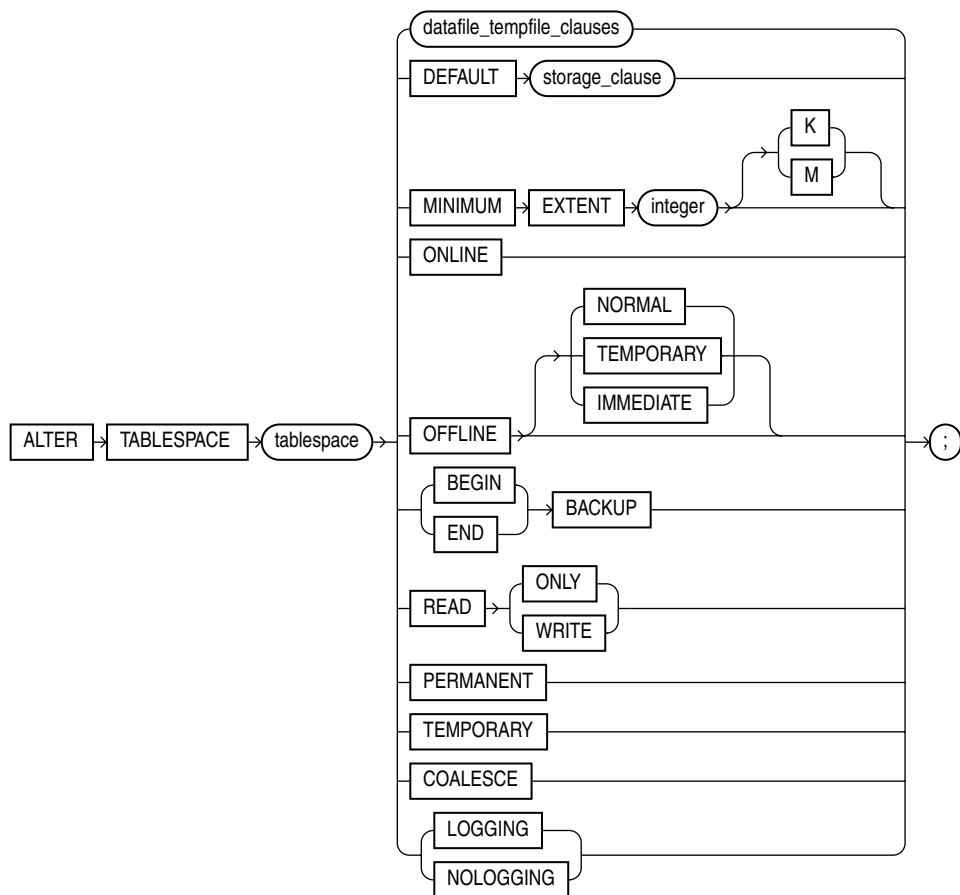
- 表領域をオンラインまたはオフラインにする。
- バックアップを開始または終了する。
- 表領域を読取り専用または読み書き両用にする。

表領域を読取り専用にする場合、次の条件が満たされている必要があります。

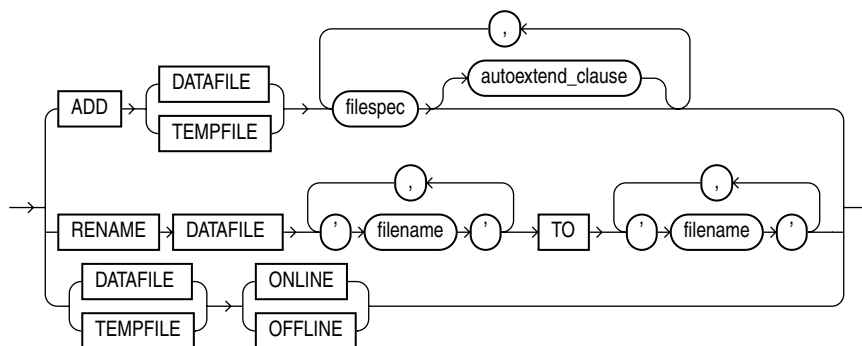
- 表領域がオンラインになっている。
- 表領域にアクティブなロールバック・セグメントがない。SYSTEM 表領域には SYSTEM ロールバック・セグメントがあるため、読取り専用にはできません。また、読取り専用表領域のロールバック・セグメントにはアクセスできないため、ロールバック・セグメントを削除してから、表領域を読取り専用にすることをお勧めします。
- 表領域がオープン・バックアップに使用されていない。バックアップの終わりに表領域内のすべてのデータ・ファイルのヘッダー・ファイルが更新されるためです。

これらの条件を満たす場合、制限モードでこの機能を実行すると便利です。制限モードでは、RESTRICTED SESSION システム権限を持つユーザーのみがログインできます。

構文

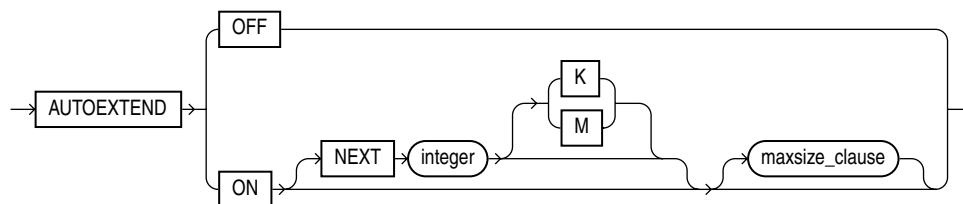
`alter_tablespace::=`

datafile_tempfile_clauses::=

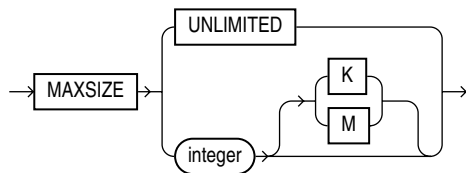


filespec: 16-27 ページの「[filespec](#)」を参照してください。

autoextend_clause::=



maxsize_clause::=



storage_clause: 17-50 ページの「[storage_clause](#)」を参照してください。

キーワードとパラメータ

tablespace

変更する表領域の名前を指定します。

制限事項：

- *tablespace* が UNDO 表領域の場合、この文では ADD DATAFILE、RENAME DATAFILE、DATAFILE ... ONLINE | OFFLINE および BEGIN | END BACKUP のみが指定可能です。
- ローカル管理の一時表領域に対してこの文で指定できるのは、ADD 句のみです。

参照： 自動 UNDO 管理および UNDO 表領域の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。

datafile tempfile clauses

datafile tempfile clauses を使用すると、データ・ファイルまたはテンポラリ・ファイルを追加および変更できます。

ADD DATAFILE | TEMPFILE 句

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

データ・ファイルまたはテンポラリ・ファイルを、オンラインのローカル管理表領域、あるいはオンラインまたはオフラインのディクショナリ管理表領域に追加できます。なお、そのデータ・ファイルが別のデータベースで使用中でないことを確認してください。

制限事項： この句は、ローカル管理の一時表領域に対して、どんな場合でも指定できる唯一の句です。

注意： オペレーティング・システムによっては、テンポラリ・ファイルのブロックが実際にアクセスされるまで、テンポラリ・ファイル用の領域が割り当てられない場合があります。領域の割当ての遅延のため、前もってテンポラリ・ファイルの作成およびサイズ変更を行う必要があります。ただし、後でテンポラリ・ファイルが使用されるときに、十分なディスク領域を使用可能にする必要があります。ご使用のオペレーティング・システムのドキュメントを参照し、このような方法でテンポラリ・ファイル領域が割り当てられるかどうかを判断してください。

参照： 16-27 ページの「[filespec](#)」を参照してください。

RENAME DATAFILE 句

1 つ以上の表領域のデータ・ファイルの名前を変更する場合は、RENAME DATAFILE で指定します。データベースをオープンしておくこと、および名前の変更前に表領域をオフラインにすることが必要です。それぞれの '*filename*' には、ご使用のオペレーティング・システムのファイル名の表記規則に従って、データ・ファイル名を完全に指定してください。

この句では、表領域を古いファイルではなく新しいファイルに対応付けます。オペレーティング・システムのファイル名は実際には変更されません。このため、オペレーティング・システム上でこのファイル名を変更する必要があります。

DATAFILE | TEMPFILE ONLINE | OFFLINE

この句を使用すると、表領域のすべてのデータ・ファイルまたはテンポラリ・ファイルを、オフラインまたはオンラインにできます。この句は、表領域の ONLINE/OFFLINE 状態には影響しません。

データベースは、マウントされている必要があります。表領域が SYSTEM、UNDO 表領域、またはデフォルトの一時表領域の場合、データベースをオープンしないでおく必要があります。

autoextend_clause

注意： この句は新しいデータ・ファイルまたはテンポラリ・ファイルを追加するときのみ有効です。既存のデータ・ファイルで自動拡張を使用可能または使用禁止にするには、ALTER DATABASE DATAFILE 句を使用します。

autoextend_clause は、新しいデータ・ファイルまたはテンポラリ・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、これらのファイルは自動的に拡張されません。

ON ON を指定すると、自動拡張を使用可能にします。

OFF OFF を指定すると、自動拡張を使用禁止にします。

注意： 自動拡張を使用禁止にする場合は、NEXT および MAXSIZE の値を 0（ゼロ）に設定します。後続の文で自動拡張を使用可能に戻す場合は、これらの値を設定しなおす必要があります。

NEXT NEXT 句を使用すると、エクステン트가さらに必要になった場合にデータ・ファイルに自動的に割り当てられるディスク領域の増分サイズを、バイト単位で指定できます。K または M を使用すると、KB または MB 単位で指定できます。デフォルトのサイズは 1 データ・ブロックです。

MAXSIZE MAXSIZE 句を使用すると、データ・ファイルの自動拡張で使用するディスク領域の最大サイズを指定できます。

UNLIMITED データ・ファイルまたはテンポラリ・ファイルに割り当てられるディスク領域を制限しない場合は、UNLIMITED 句を使用します。

DEFAULT *storage_clause*

DEFAULT *storage_clause* には、表領域に作成される後続のオブジェクトに対する新しいデフォルトの記憶域パラメータを指定します。ディクショナリ管理の一時表の場合は、*storage_clause* の NEXT パラメータのみが考慮されます。

制限事項：この句は、ローカル管理の表領域に対して指定できません。

参照： 17-50 ページの「[storage_clause](#)」を参照してください。

MINIMUM EXTENT

MINIMUM EXTENT 句を指定すると、表領域内のすべての使用済エクステン트または未使用エクステン트의大きさが、*integer* で指定したサイズ以上であること、およびその倍数であることが保証され、表領域における空き領域の断片化を制御できます。ディクショナリ管理の一時表領域の場合、この句は関係ありません。

制限事項：この句は、ローカル管理の表領域に対して指定できません。

参照： MINIMUM EXTENT を使用した断片化の制御については、『Oracle9i データベース管理者ガイド』を参照してください。

ONLINE

ONLINE を指定して、表領域をオンラインにします。

OFFLINE

OFFLINE を指定して、表領域をオフラインにし、そのセグメントにアクセスできないようにします。表領域をオフラインにすると、そのすべてのデータ・ファイルもオフラインになります。

提案： 表領域を長期間オフラインにするには、デフォルト表領域または一時表領域としてその表領域を割り当てられているユーザーの表領域割当てを変更した方がよい場合があります。表領域をオフラインにした場合、これらのユーザーは、その表領域内でオブジェクトに対して領域を割り当てたり、領域をソートすることはできません。ユーザーへの表領域の割当ての詳細は、11-20 ページの「[ALTER USER](#)」を参照してください。

制限事項： 一時表領域はオフラインにできません。

NORMAL NORMAL を指定すると、SGA 以外にある表領域のすべてのデータ・ファイルにあるすべてのブロックがフラッシュされます。データ・ファイルをオンラインに戻す前に、表領域のメディア・リカバリを行う必要はありません。これはデフォルトです。

TEMPORARY TEMPORARY を指定すると、Oracle は表領域内のすべてのオンライン・データ・ファイルに対してチェックポイントを実行しますが、すべてのファイルに書き込むことができることを保証しません。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・リカバリを行う必要があります。

IMMEDIATE IMMEDIATE を指定すると、Oracle は表領域のファイルが使用可能であることを保証しません。また、チェックポイントも実行しません。表領域をオンラインに戻す前に、メディア・リカバリを行う必要があります。

注意： ALTER TABLESPACE ... OFFLINE への FOR RECOVER 設定は使用できません。構文は下位互換用にサポートされますが、表領域のリカバリにはトランスポータブル表領域を使用することをお勧めします。

参照： メディア・リカバリを実行するトランスポータブル表領域の使用の詳細は、『Oracle9i ユーザー管理バックアップおよびリカバリ・ガイド』を参照してください。

BEGIN BACKUP

BEGIN BACKUP プロシージャは、表領域を構成するデータ・ファイルのオープン・バックアップを実行することを示します。この句を指定することによって、ユーザーがこの表領域にアクセスできなくなることはありません。オープン・バックアップを開始する前に、この句を指定してください。

制限事項：この句は、読取り専用の表領域またはローカル管理の一時表領域に対して指定できません。

注意： バックアップ中は、表領域の標準オフラインへの切替え、インスタンスの停止または表領域の別のバックアップ処理の開始は実行できません。

END BACKUP

END BACKUP は、表領域のオンライン・バックアップが完了したことを示します。オンライン・バックアップの完了後、できるだけ早くこの句を指定してください。インスタンスに障害または SHUTDOWN ABORT が発生した場合、次のインスタンス起動時にメディア・リカバリ（必要に応じて、アーカイブ REDO ログも）が必要であるとみなされます。

制限事項：この句は、読取り専用表領域に対して使用できません。

参照：

- メディア・リカバリなしでデータベースを再起動する場合の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。
- 個々のデータ・ファイルまたは表領域のすべてのデータ・ファイルをオンライン（ホット）・バックアップ・モード以外にする場合の詳細は、8-26 ページの「ALTER DATABASE [END BACKUP](#) 句」を参照してください。

READ ONLY | READ WRITE

READ ONLY によって、表領域を**読取り専用推移モード**に設定します。この状態では、既存のトランザクションは完了（コミットまたはロールバック）できますが、以前の表領域内のブロックを変更した既存のトランザクションのロールバック以外は、その表領域に対してさらに書き込み操作（DML）を行うことはできません。

表領域を読取り専用にした場合、そのファイルを読取り専用メディアにコピーできます。その場合、SQL 文の ALTER DATABASE ... RENAME を使用して、新しいファイル位置を示すように制御ファイルのデータ・ファイルの名前を変更する必要があります。

参照：

- 読取り専用の表領域の詳細は、『Oracle9i データベース概要』を参照してください。
- 8-9 ページの「[ALTER DATABASE](#)」を参照してください。

読取り専用指定されている表領域に対して書込み操作を可能にする場合は、`READ WRITE`を指定します。

PERMANENT | TEMPORARY

一時表領域を永続表領域に変換する場合は、`PERMANENT`を指定します。永続表領域とは、永続的なデータベース・オブジェクトを格納できる場所です。表領域を作成するときのデフォルトです。

永続表領域を一時表領域に変換する場合は、`TEMPORARY`を指定します。一時表領域とは、永続的なデータベース・オブジェクトを格納できない表領域です。一時表領域の中のオブジェクトはセッション中のみ保持されます。

制限事項：表領域を標準的なブロック・サイズで作成しなかった場合、永続表領域を一時表領域に変換できません。

COALESCE

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

LOGGING | NOLOGGING

表領域内のすべての表、索引およびパーティションのロギング属性を指定する場合、`LOGGING`を指定します。表レベル、索引レベルおよびパーティション・レベルでのロギング指定によって、表領域レベルのロギング属性をオーバーライドできます。

既存の表領域のロギング属性を `ALTER TABLESPACE` 文によって変更した場合、この文の実行後に作成されたすべての表、索引およびパーティションに、新しいデフォルトのロギング属性（これは後でオーバーライドもできます）が適用されます。既存のオブジェクトのロギング属性は変更されません。

次の操作のみが、`NOLOGGING` モードをサポートします。

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

NOLOGGING モードでは、データの変更時に、(新しいエクステン트에 **INVALID** のマークを設定し、ディクショナリの変更を記録するために) 最小限のログが記録されます。メディア・リカバリ中に NOLOGGING が適用された場合、REDO データのログ記録が中断されるため、エクステン트無効化レコードでは、一定のブロック範囲に「論理的に無効」というマークが付きます。このため、損失してはならないオブジェクトの場合は、NOLOGGING 操作の後にバックアップを取る必要があります。

例

バックアップの例 次の文は、バックアップの開始をデータベースに通知します。

```
ALTER TABLESPACE accounting
  BEGIN BACKUP;
```

次の文は、バックアップが終了したことをデータベースに通知します。

```
ALTER TABLESPACE accounting
  END BACKUP;
```

移動および名前の変更例 次の例は、accounting 表領域に対応付けられたデータ・ファイル 'diska:pay1:dat' を移動して 'diskb:receive1:dat' に名前を変更します。

1. OFFLINE 句を指定した ALTER TABLESPACE 文を使用して、この表領域をオフラインにします。

```
ALTER TABLESPACE accounting OFFLINE NORMAL;
```

2. オペレーティング・システムのコマンドを使用して、このファイルを 'diska:pay1.dat' から 'diskb:receive1.dat' にコピーします。

3. RENAME DATAFILE 句を指定した ALTER TABLESPACE 文を使用して、このデータ・ファイルの名前を変更します。

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

4. ONLINE 句を指定した ALTER TABLESPACE 文を使用して、この表領域をオンラインに戻します。

```
ALTER TABLESPACE accounting ONLINE;
```

データ・ファイルの追加例 次の文は、表領域にデータ・ファイルを追加します。さらに多くの領域が必要な場合、10KB の新しいエクステン트가最大 100KB まで追加されます。

```
ALTER TABLESPACE tab_space2
  ADD DATAFILE 'diska:tabspace_file3.dat'
  SIZE 50K
  AUTOEXTEND ON
  NEXT 10K
  MAXSIZE 100K;
```

ロギング属性の変更例 次の例は、表領域のデフォルトのロギング属性を NOLOGGING に変更します。

```
ALTER TABLESPACE tab_space2 NOLOGGING;
```

表領域のロギング属性を変更した場合でも、その表領域内の既存のスキーマ・オブジェクトのロギング属性には影響しません。表レベル、索引レベルおよびパーティション・レベルでのロギング指定によって、表領域レベルのロギング属性をオーバーライドできます。

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

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

Oracle 管理データ・ファイルの追加例 次の例は、Oracle 管理データ・ファイルを omf_ts1 表領域に追加します。(表領域の作成の詳細は、14-72 ページの「[Oracle 管理ファイルの例](#)」を参照してください。) 新しいデータ・ファイルは 100MB で、最大サイズが制限なしで自動拡張されます。

```
ALTER TABLESPACE omf_ts1 ADD DATAFILE;
```

SQL 文 : ALTER TRIGGER ~ *constraint_clause*

この章では、次の SQL 文について説明します。

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

ALTER TRIGGER

用途

ALTER TRIGGER を使用すると、データベース・トリガーを使用可能、使用禁止またはコンパイルできます。

注意： この文を使用して既存のトリガーの宣言や定義は変更できません。トリガーを再宣言または再定義する場合は、OR REPLACE キーワードを指定した CREATE TRIGGER 文を使用します。

参照：

- トリガー作成の詳細は、14-77 ページの「[CREATE TRIGGER](#)」を参照してください。
- トリガー削除の詳細は、16-12 ページの「[DROP TRIGGER](#)」を参照してください。
- トリガーの概要は、『Oracle9i データベース概要』を参照してください。

前提条件

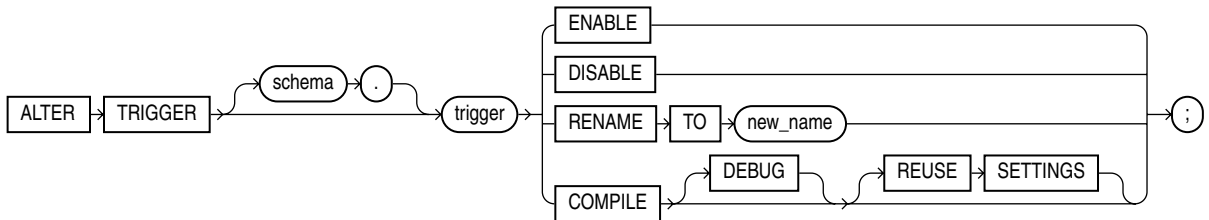
トリガーが自分のスキーマ内にある必要があります。自分のスキーマ内にはない場合は、ALTER ANY TRIGGER システム権限が必要です。

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

参照： DATABASE トリガーに基づいたトリガーの詳細は、14-77 ページの「[CREATE TRIGGER](#)」を参照してください。

構文

alter_trigger::=



キーワードとパラメータ

schema

削除するトリガーが含まれているスキーマを指定します。*schema* を指定しない場合、トリガーは自分のスキーマ内に定義されているとみなされます。

trigger

変更するトリガーの名前を指定します。

ENABLE 句

ENABLE を指定して、トリガーを使用可能にします。また、ALTER TABLE の ENABLE ALL TRIGGERS 句を使用することによって、表に対応付けられたすべてのトリガーを使用可能にできます。

参照： 10-2 ページの「[ALTER TABLE](#)」を参照してください。

DISABLE 句

DISABLE を指定して、トリガーを使用禁止にします。また、ALTER TABLE の DISABLE ALL TRIGGERS 句を使用することによって、表に対応付けられたすべてのトリガーを使用禁止にできます。

参照： 10-2 ページの「[ALTER TABLE](#)」を参照してください。

RENAME 句

RENAME TO *new_name* を指定して、トリガーの名前を変更します。トリガーの名前は変更され、名前が変更される前と同じ状態になります。

COMPILE 句

トリガーが使用可能または使用禁止であるかにかかわらず、トリガーを明示的にコンパイルするには、**COMPILE** を指定します。明示的に再コンパイルすることによって、実行時に暗黙的に再コンパイルする必要がなくなり、また、実行時のコンパイル・エラーとパフォーマンス上のオーバーヘッドもなくなります。

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

再コンパイル中、**Oracle** はすべての永続コンパイラのスイッチ設定を削除し、セッションからそれらを再び取得してコンパイルの終了時に格納します。この手順を回避するには、**REUSE SETTINGS** 句を指定します。

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

DEBUG **DEBUG** を指定すると、**PL/SQL** コンパイラに対して、**PL/SQL** デバッグ用のコードを生成および格納するように指示できます。

参照：

- プロシージャのデバッグの詳細は、『**Oracle9i** アプリケーション開発者ガイド - 基礎編』を参照してください。
- リモート・オブジェクトなどのスキーマ・オブジェクト間の依存性を **Oracle** が管理する方法については、『**Oracle9i** データベース概要』を参照してください。

REUSE SETTINGS **REUSE SETTINGS** を指定すると、**Oracle** によるコンパイラのスイッチ設定の削除および再取得を回避できます。この句では既存の設定が持続され、その設定で再コンパイルされます。

DEBUG と **REUSE SETTINGS** の両方を指定する場合は、**PLSQL_COMPILER_FLAGS** パラメータの永続格納値が **INTERPRETED, DEBUG** に設定されます。その他のコンパイラのスイッチ値は変更されません。

参照： **PLSQL_COMPILER_FLAGS** パラメータと **COMPILE** 句の相互作用の詳細は、『**PL/SQL** ユーザーズ・ガイドおよびリファレンス』および『**Oracle9i** アプリケーション開発者ガイド - 基礎編』を参照してください。

例

トリガーを使用禁止にする例 サンプル・スキーマ hr には、employees 表で作成された update_job_history という名前のトリガーがあります。トリガーは、UPDATE 文によって従業員の job_id が変更されるたびに起動されます。トリガーは、job_history 表に従業員 ID、直前の職種の開始日と終了日、職種 ID および部門を含む行を挿入します。

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

```
ALTER TRIGGER update_job_history DISABLE;
```

使用禁止になると、UPDATE 文によって従業員の job_id が変更されてもトリガーは起動されません。

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

```
ALTER TRIGGER update_job_history ENABLE;
```

再びトリガーを使用可能にした場合、UPDATE 文によって従業員の job_id が変更されるたびにトリガーが起動されます。トリガーが使用禁止の場合、従業員の job_id が変更されても、他のトランザクションが再び job_id を変更するまで、トリガーは自動的に起動されません。

ALTER TYPE

用途

ALTER TYPE 文を使用すると、メンバー属性またはメソッドを追加または削除できます。オブジェクト型の既存のプロパティ（FINAL または INSTANTIABLE）、または型のスカラー属性も変更できます。

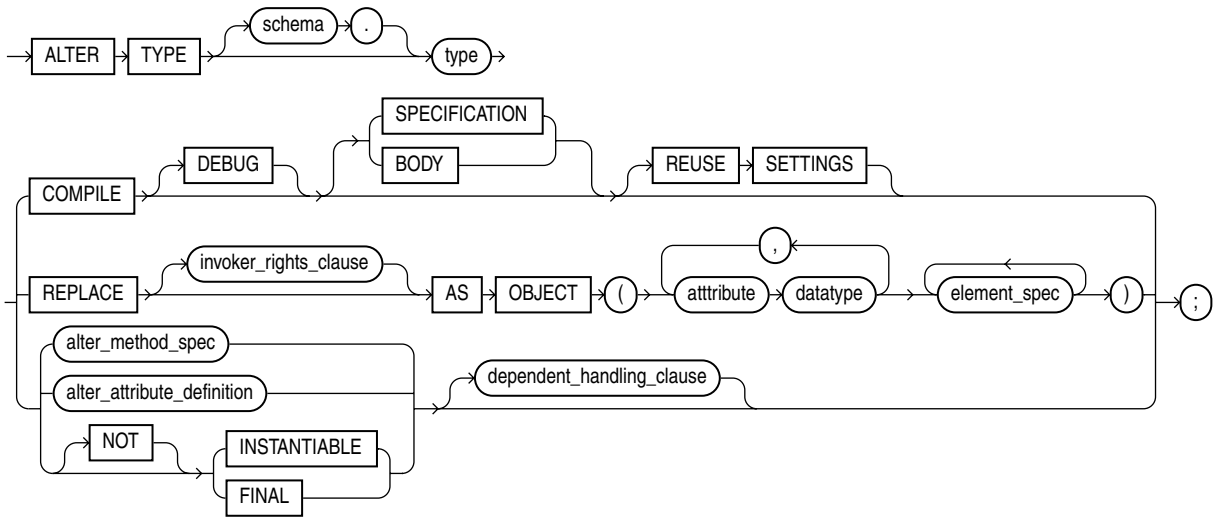
この文を使用すると、新しいオブジェクト・メンバーのサブプログラム仕様を追加することによって、型の仕様部または本体を再コンパイルしたり、オブジェクト型の仕様を変更できます。

前提条件

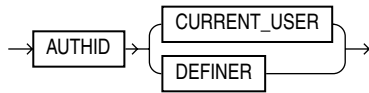
オブジェクト型が自分のスキーマ内にあり、CREATE TYPE または CREATE ANY TYPE システム権限が必要です。または、ALTER ANY TYPE システム権限が必要です。

構文

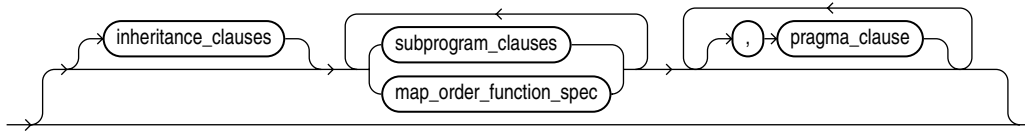
alter_type::=



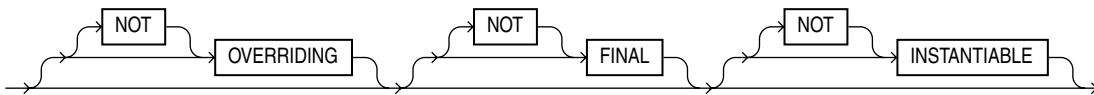
invoker_rights_clause::=



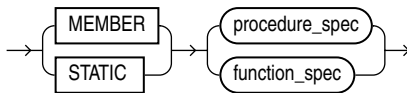
element_spec::=



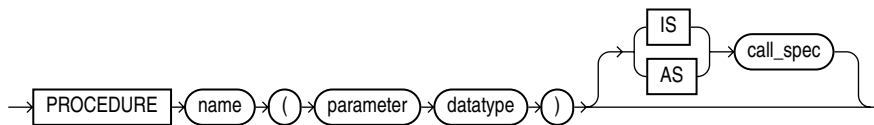
inheritance_clauses::=



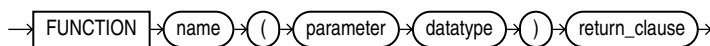
subprogram_clauses::=



procedure_spec::=

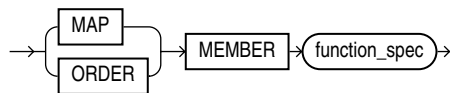


function_spec::=

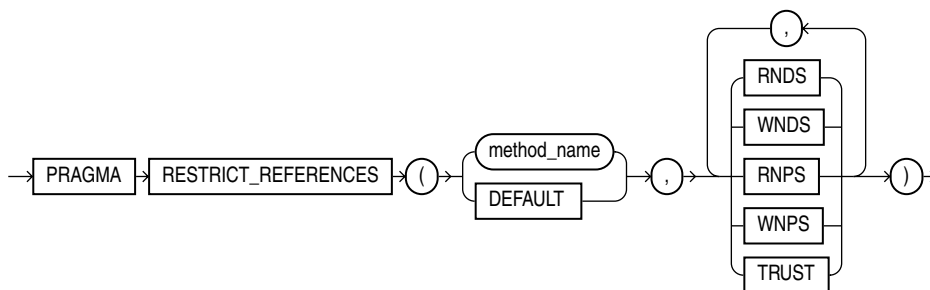


ALTER TYPE

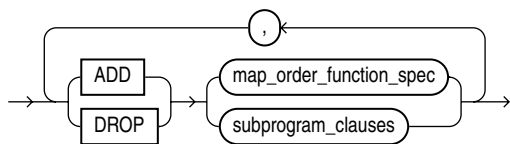
map_order_function_spec::=



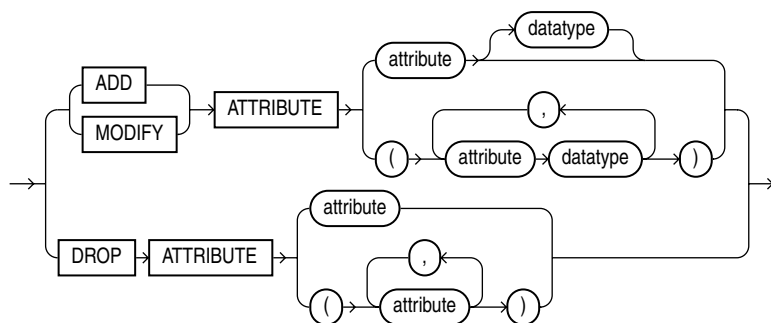
pragma_clause::=



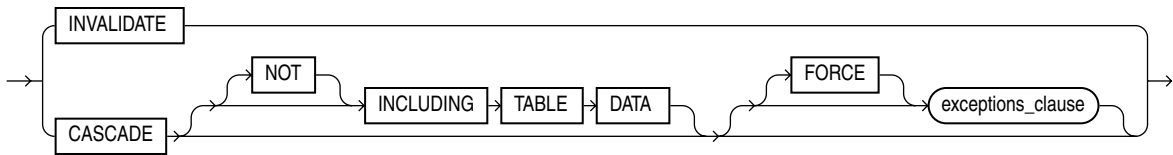
alter_method_spec::=



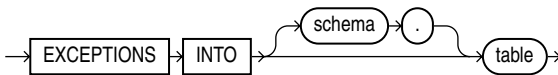
alter_attribute_definition::=



dependent_handling_clause::=



exceptions_clause::=



キーワードとパラメータ

schema

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

type

オブジェクト型、ネストした表型または ROWID 型の名前を指定します。

COMPILE 句

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

再コンパイル中、Oracle はすべての永続コンパイラのスイッチ設定を削除し、セッションからそれらを再び取得してコンパイルの終了時に格納します。この手順を回避するには、REUSE SETTINGS 句を指定します。

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

DEBUG DEBUG を指定すると、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格納するように指示できます。

SPECIFICATION SPECIFICATION を指定して、オブジェクト型の仕様部のみをコンパイルします。

BODY BODY を指定して、オブジェクト型の本体のみをコンパイルします。

REUSE SETTINGS REUSE SETTINGS を指定すると、Oracle によるコンパイラ・スイッチ設定の削除および再取得を回避できます。この句では既存の設定が持続され、その設定で再コンパイルされます。

DEBUG と REUSE SETTINGS の両方を指定する場合は、PLSQL_COMPILER_FLAGS パラメータの永続格納値が INTERPRETED, DEBUG に設定されます。その他のコンパイラのスイッチ値は変更されません。

参照： PLSQL_COMPILER_FLAGS パラメータと COMPILE 句の相互作用の詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』および『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

REPLACE AS OBJECT 句

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

attribute

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

element_spec

再定義するオブジェクトの要素を指定します。

inheritance_clauses *element_spec* の一部として *inheritance_clauses* を使用すると、スーパータイプとサブタイプの間の関係を指定できます。

OVERRIDING OVERRIDING を指定すると、スーパータイプで定義されているメソッドをこのメソッドでオーバーライドします。このキーワードは、スーパータイプのメソッドを再定義する場合に必須です。NOT OVERRIDING がデフォルトです。

制限事項：OVERRIDING 句は、SQLJ オブジェクト型には無効です。

FINAL FINAL を指定すると、メソッドがこの型のサブタイプによってオーバーライドされないようにできます。デフォルトは、NOT FINAL です。

NOT INSTANTIABLE 型がこのメソッドの実装を提供しない場合は、NOT INSTANTIABLE を指定します。デフォルトでは、すべてのメソッドは INSTANTIABLE です。

制限事項：NOT INSTANTIABLE を指定する場合は、FINAL または STATIC を指定できません。

subprogram_clauses MEMBER および STATIC を指定すると、属性として参照されるオブジェクト型に対応付けられたファンクション・サブプログラムまたはプロシージャ・サブプログラムを指定できます。

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

参照：

- メンバー・メソッドとスタティック・メソッドの違い、およびこれらの例については、15-3 ページの「[CREATE TYPE](#)」を参照してください。
- パッケージ内のサブプログラム名のオーバーロードについては、『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。
- 15-24 ページの「[CREATE TYPE BODY](#)」を参照してください。

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

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

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

アプリケーションの下位互換を保つ必要がない場合は、この句を使用しないことをお勧めします。Oracle9i からは、純粋度チェックが実行時に行われるため、この句は使用されなくなります。

制限事項：*pragma_clause* は、メソッドを削除すると無効になります。

参照：『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

map_order_function_spec MAP メソッドまたは ORDER メソッドのいずれかを宣言できますが、両方は宣言できません。ただし、スーパータイプが NOT FINAL MAP メソッドを定義する場合、サブタイプは MAP メソッドをオーバーライドできます。いずれかのメソッドを宣言すると、SQL 内でオブジェクト・インスタンスを比較できます。

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

参照： オブジェクト値の比較の詳細は、2-45 ページの「[オブジェクト値](#)」を参照してください。

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

MAP メソッドの引数が NULL の場合、MAP メソッドは NULL を戻し、メソッドはコールされません。

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

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

サブタイプには、新しい MAP メソッドを定義できません。ただし、継承された MAP メソッドはオーバーライドできます。

- ORDER の場合、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 引数とするメンバー・ファンクション (ORDER メソッド) を指定し、負の整数、0 (ゼロ) または正の整数のいずれかを戻します。負の整数は、暗黙的な SELF 引数が明示的な引数より小さいことを示し、0 (ゼロ) は、両方が等しいことを示します。また、正の整数は、暗黙的な SELF 引数が明示的な引数より大きいことを示します。

ORDER メソッドのどちらかの引数が NULL の場合は、ORDER メソッドは NULL を戻し、メソッドはコールされません。

同じオブジェクト型定義の各インスタンスを ORDER BY 句の中で比較した場合、ORDER メソッド・ファンクションがコールされます。

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

サブタイプは、ORDER メソッドを定義できません。また、継承された ORDER メソッドをオーバーライドすることもできません。

invoker_rights_clause

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

また、この句は、問合せ、DML 操作、およびその型のメンバー・ファンクションおよびプロシージャ内の動的 SQL 文の外部名の変換方法も定義します。

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

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

また、この句は、問合せ、DML 操作、および動的 SQL 文の外部名を CURRENT_USER のスキーマで変換することも指定します。その他すべての文の外部名は、型が存在するスキーマ内で変換します。

注意： 実行者権限を実行者権限状態で作成した場合、この句を指定し、この型への実行者権限を維持する必要があります。指定しないと、この状態は定義者権限に戻ります。

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

参照：

- CURRENT_USER を判断する方法については、『Oracle9i データベース概要』および『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。
- 『PL/SQL 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。

alter_method_spec

alter_method_spec を使用すると、*type* に対してメソッドを追加および削除できます。型に依存するすべてのファンクション索引が使用禁止になります。

1 つの ALTER TYPE 文で複数のメソッドを追加または削除できますが、各メソッドを参照できるのは 1 回のみです。

ADD メソッドを追加する場合は、型の階層内の既存の属性と競合しない名前にする必要があります。

DROP メソッドを削除すると、対象の型からメソッドが削除されます。

制限事項： スーパータイプから継承されたメソッドを、サブタイプで削除できません。かわりに、スーパータイプでメソッドを削除します。

subprogram_clauses MEMBER および STATIC 句を使用すると、オブジェクト型に対してプロシージャ・サブプログラムを追加または削除できます。

制限事項： スーパータイプの MEMBER メソッドを再定義するサブタイプの STATIC メソッドは定義できません。またその逆も同様です。

参照： 15-12 ページの「CREATE TYPE」の「[subprogram_clauses](#)」を参照してください。

map_order_function_spec MAP と ORDER の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。

制限事項： サブタイプに ORDER メソッドを追加できません。

参照： 15-16 ページの「CREATE TYPE」の「[map_order_function_spec](#)」を参照してください。

alter_attribute_definition

alter_attribute_definition 句を使用すると、オブジェクト型の属性を追加、削除または変更できます。1 つの ALTER TYPE 文で複数のメンバー属性またはメソッドを追加、削除または変更できますが、各属性または各メソッドを参照できるのは 1 回のみです。

ADD ATTRIBUTE 新しい属性は、型の階層内にすでに存在する属性またはメソッドと競合しない名前にする必要があります。新しい属性は、ローカル定義の属性リストの終わりに追加されます。

注意： スーパータイプに属性を追加すると、すべてのサブタイプに継承されます。サブタイプでは、継承された属性は、宣言した属性より常に優先されます。そのため、スーパータイプに属性を追加した後、暗黙的に変更されたサブタイプのマッピングの更新が必要な場合があります。

DROP ATTRIBUTE 型から属性を削除すると、削除された属性に対応する列、索引、統計情報および削除する属性を参照する制約も削除されます。

削除する属性のデータ型を指定する必要はありません。

制限事項：

- スーパータイプから継承された属性を削除できません。かわりに、スーパータイプから属性を削除します。
- パーティション、サブパーティションまたはクラスタ・キーの一部になっている属性は、削除できません。
- オブジェクト表、主キーまたは索引構成表の主キー・ベースのオブジェクト識別子の属性は、削除できません。
- ルート型のすべての属性は削除できません。かわりに、型を削除します。ただし、サブタイプのローカルで宣言されたすべての属性は、削除できます。

MODIFY ATTRIBUTE 既存のスカラー属性のデータ型を変更できます。たとえば、VARCHAR2 または RAW の長さ、あるいは数値属性の精度または位取りを増やすことができます。

制限事項： ファンクション索引、ドメイン索引またはクラスタ・キーで参照される属性のサイズを拡張することはできません。

[NOT] FINAL

[NOT] FINAL 句を使用すると、この型のサブタイプを以降で作成させるかどうかを指定できます。

- この型のサブタイプを以降で作成させない場合は、FINAL を指定します。
- この型のサブタイプを以降で作成させる場合は、NOT FINAL を指定します。

注意： FINAL と NOT FINAL との間でプロパティを変更する場合は、CASCADE INCLUDING TABLE DATA 句を指定した *dependent_handling_clause* を指定する必要があります。

制限事項： 型にサブタイプがある場合、ユーザー定義型を NOT FINAL から FINAL に変更できません。

[NOT] INSTANTIABLE

[NOT] INSTANTIABLE 句を使用すると、この型のオブジェクト・インスタンスを構成させるかどうかを指定できます。

- この型のオブジェクト・インスタンスを構成させる場合は、INSTANTIABLE を指定します。
- このオブジェクト型のコンストラクタ（デフォルトまたはユーザー定義の）が存在しない場合は、NOT INSTANTIABLE を指定します。インスタンス化ができないメソッドを持つ型、および属性を持たない型には、これらのキーワードを指定する必要があります（継承、またはこの句で指定）。

制限事項：型に依存する表がある場合、ユーザー定義型を INSTANTIABLE から NOT INSTANTIABLE に変更できません。

dependent_handling_clause

dependent_handling_clause を使用すると、変更された型に依存するオブジェクトの処理方法を Oracle に指示できます。この句を指定しないと、対象の型に依存する型または表がある場合に ALTER TYPE 文が異常終了します。

INVALIDATE 句

INVALIDATE を指定すると、メカニズムの検証なしで、依存するすべてのオブジェクトが無効になります。

注意： 型の変更が検証されないため、この句の使用には注意が必要です。たとえば、パーティション化キーまたはクラスタ・キーの属性を削除すると、表に書込みができなくなります。

CASCADE 句

型の変更を依存する型および表まで広める場合は、CASCADE 句を指定します。依存する型または表でエラーがあったときに、FORCE を指定していない場合は、文は異常終了します。

INCLUDING TABLE DATA INCLUDING TABLE DATA を指定すると、すべてのユーザー定義列に格納されているデータを列の型の最新バージョンに変換します。これはデフォルトです。

注意： 列データが Oracle8 リリース 8.0 のイメージ・フォーマットの場合、この句を指定する必要があります。この句は、FINAL と NOT FINAL の間で型のプロパティを変更する場合にも必須です。

- 列の型に各属性を追加すると、新しい属性はデータに追加され、NULL に初期化されます。
- 参照される型から各属性を削除すると、表の各行から対応する属性データが削除されます。

INCLUDING TABLE DATA を指定する場合、表のデータを含む表領域のすべてを読み取り / 書き込みモードにする必要があります。

NOT INCLUDING TABLE DATA を指定すると、列のメタデータはアップグレードされ、変更は型に反映されますが、依存する列はスキャンされず、この ALTER TYPE 文の一部としてはデータが更新されません。ただし、依存する列データはアクセスすることができ、後続のデータの間合せの結果では、型の変更が反映されます。

参照： 型属性の変更時に表データを含まない実装の詳細は、『Oracle9i アプリケーション開発者ガイド - オブジェクト・リレーショナル機能』を参照してください。

FORCE 依存する表および索引のエラーを無視し、すべてのエラーを指定した例外表のログに記録する場合は、FORCE を指定します。例外表は、DBMS_UTILITY.CREATE_ALTER_TYPE_ERROR_TABLE プロシージャの実行によって作成しておく必要があります。

例

メンバー・ファンクションの追加例 次の例は、15-18 ページの「オブジェクト型の例」で作成した `data_typ` オブジェクト型を使用します。メソッドは、`data_typ` に追加され、型の本体はそれに対応して変更されます。日付書式は、`oe.orders` デモ表の `order_date` 列と一致しています。

```
ALTER TYPE data_typ
  ADD MEMBER FUNCTION qtr(der_qtr DATE)
  RETURN CHAR CASCADE;

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

  MEMBER FUNCTION qtr(der_qtr DATE) RETURN CHAR IS
  BEGIN
    IF (der_qtr < TO_DATE('01-APR', 'DD-MON')) THEN
      RETURN 'FIRST';
    ELSIF (der_qtr < TO_DATE('01-JUL', 'DD-MON')) THEN
      RETURN 'SECOND';
    ELSIF (der_qtr < TO_DATE('01-OCT', 'DD-MON')) THEN
      RETURN 'THIRD';
    ELSE
      RETURN 'FOURTH';
    END IF;
  END;
END;
```

コレクション属性の追加例 次の例は、デモ・スキーマ `oe` の `VARRAY` 型の `phone_list_typ` を、`customers` 表の `customer_address_typ` オブジェクト列に追加します。

```
ALTER TYPE cust_address_typ
  ADD ATTRIBUTE (phone phone_list_typ) CASCADE;
```

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

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

```
ALTER TYPE loan_t COMPILE;
```

型仕様部の再コンパイル 次の文は、link2 の型本体をコンパイルします。

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

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

```
CREATE TYPE BODY link2 AS
  MEMBER FUNCTION p(c1 NUMBER) RETURN NUMBER IS
  BEGIN
    dbms_output.put_line(c1);
    RETURN c1;
  END;
END;
```

次の例は、link2 の仕様部と本体の両方を無効にします。

```
ALTER TYPE link1 ADD ATTRIBUTE (b NUMBER) INVALIDATE;
```

別の文で仕様部と本体を再コンパイルすることによって、型を再コンパイルする必要があります。

```
ALTER TYPE link2 COMPILE SPECIFICATION;
```

```
ALTER TYPE link2 COMPILE BODY;
```

仕様部と本体の両方は同時にコンパイルできます。

```
ALTER TYPE link2 COMPILE;
```

ALTER USER

用途

ALTER USER を使用すると、次の処理ができます。

- データベース・ユーザーの認証またはデータベース・リソースの特性を変更します。
- プロキシ・サーバーが認証なしでクライアントとして接続することを許可します。

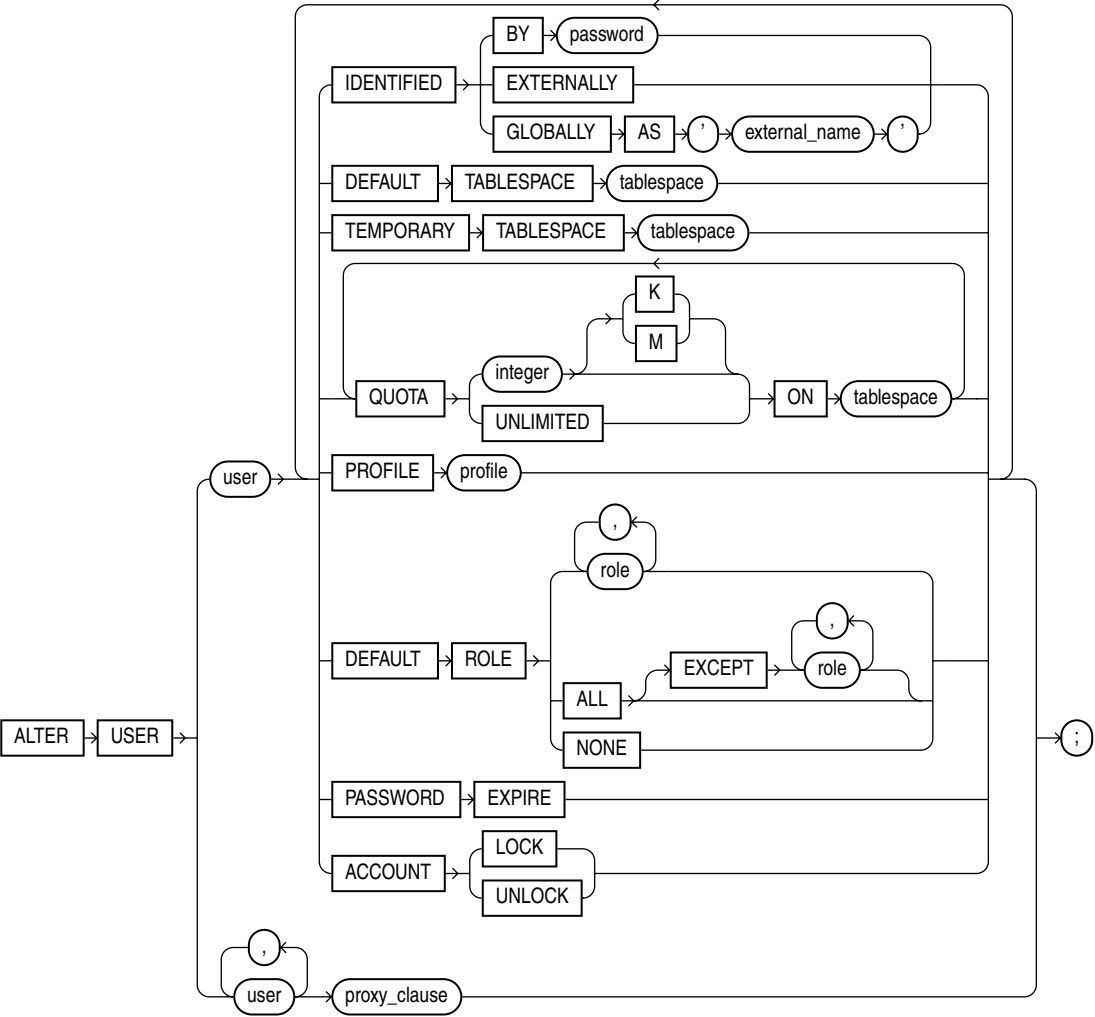
注意： ALTER USER 構文は、古いパスワードを受け入れません。したがって、この構文では古いパスワードを使用して認証することも、新しいパスワードを設定する前に古いパスワードと新しいパスワードを照合することもできません。古いパスワードを確認する必要がある場合、ALTER USER のかわりに OCIPasswordChange () コールを使用してください。詳細は、『Oracle Call Interface プログラマーズ・ガイド』を参照してください。

前提条件

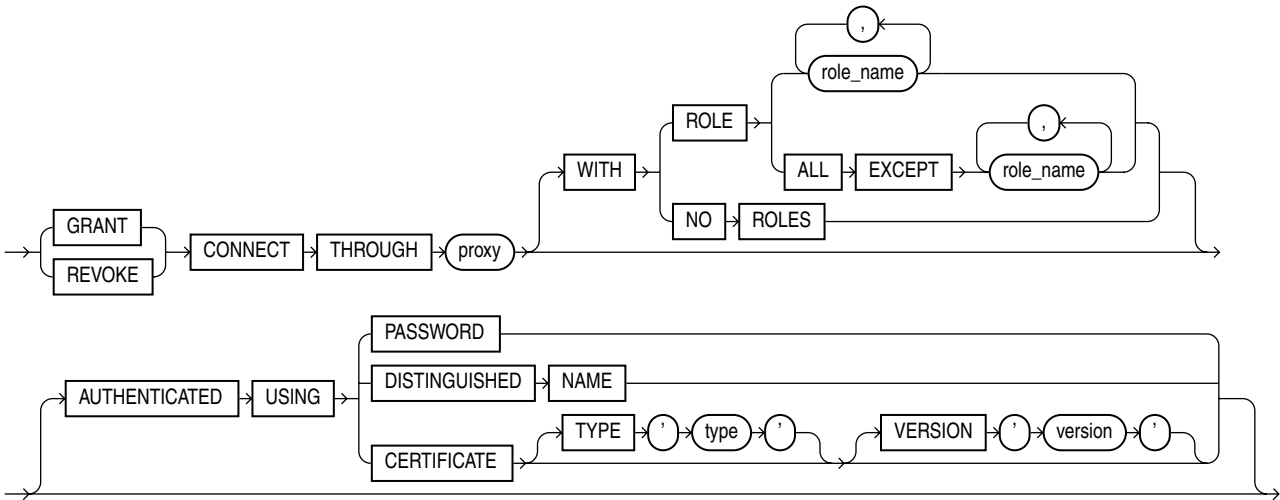
ALTER USER システム権限が必要です。ただし、ユーザー自身のパスワードはこの権限がない場合でも変更できます。

構文

alter_user::=



proxy_clause::=



キーワードとパラメータ

後述のキーワードおよびパラメータは、ALTER USER 独自のものと、CREATE USER にもあるが、セマンティクスが異なるものがあります。ALTER USER 文のその他のすべてのキーワードおよびパラメータは、CREATE USER 文のキーワードとパラメータと同じです。

参照：

- キーワードおよびパラメータについては、15-30 ページの「[CREATE USER](#)」を参照してください。
- ユーザーにデータベース・リソースへの制限を割り当てる方法については、13-65 ページの「[CREATE PROFILE](#)」を参照してください。

IDENTIFIED 句

- `BY password` を指定し、ユーザーの新しいパスワードを指定します。

注意：異なるタイムスタンプで特定のパスワードを再設定する必要があります。1 秒以内に 1 つのパスワードを複数回再設定した場合（たとえば、スクリプトを使用して一連のパスワードの設定を繰り返した場合）、Oracle はパスワードが再利用できないというエラー・メッセージを戻すことがあります。このため、パスワードの再設定には、スクリプトを使用しないことをお勧めします。

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

ユーザーに直接付与されたすべての外部ロールが取り消された場合のみ、ユーザー・アクセス検証方法を `IDENTIFIED GLOBALLY AS 'external_name'` に変更できます。

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

参照： 15-30 ページの「[CREATE USER](#)」を参照してください。

TEMPORARY TABLESPACE 句

ユーザーの一時表領域として割り当てる表領域、または再度割り当てる表領域には、標準以外のブロック・サイズを使用した表領域を指定できません。指定した場合、Oracle からエラーは戻されません。ただし、ユーザーが行う後続の操作で一時セグメントが必要な場合は、エラーになります。

DEFAULT ROLE 句

ログイン時にデフォルトによってユーザーに付与されるロールを指定します。この句では、`GRANT` 文を使用してユーザーに直接付与されているロールのみ指定できます。`DEFAULT ROLE` 句を使用して次のロールを使用可能にすることはできません。

- ユーザーに付与されていないロール
- 他のロールを介して付与されているロール
- 外部サービス（オペレーティング・システムなど）または Oracle Internet Directory によって管理されるロール

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

参照： 13-72 ページの「[CREATE ROLE](#)」を参照してください。

proxy_clause

proxy_clause を使用すると、**プロキシ**（アプリケーションまたはアプリケーション・サーバー）の機能を制御でき、指定したデータベース・ユーザーまたはエンタープライズ・ユーザとして接続し、すべてまたはいくつかのユーザー・ロールをアクティブにする、あるいは非アクティブにできます。

注意： *proxy_clause* は、データベース・ユーザーおよびエンタープライズ・ユーザーに対していくつかのプロキシ認証を提供します。アプリケーション・ユーザーのプロキシ認証の詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

参照： プロキシおよびそれらのデータベースについては、『Oracle9i データベース概要』を参照してください。

GRANT | REVOKE

接続を許可するには、GRANT を指定します。接続を禁止するには、REVOKE を指定します。

CONNECT THROUGH 句

Oracle に接続するプロキシを識別します。AUTHENTICATED USING 句を指定しない場合は、Oracle は、ユーザーの認証にプロキシを想定します。

WITH ROLE WITH ROLE *role_name* を使用すると、プロキシは指定したユーザーとして接続でき、*role_name* で指定されたロールのみをアクティブにできます。

WITH ROLE ALL EXCEPT WITH ROLE ALL EXCEPT *role_name* を使用すると、プロキシは指定したユーザーとして接続でき、*role_name* で指定されたロール以外の、このユーザーに対応付けられたすべてのロールをアクティブにできます。

WITH NO ROLES WITH NO ROLES を使用すると、プロキシは指定したユーザーとして接続できますが、接続後にそのユーザーのロールを 1 つでもアクティブにすることは禁止されます。

WITH 句を指定しない場合、指定したユーザーに付与されているすべてのロールが自動的にアクティブになります。

AUTHENTICATED USING

プロキシ認証をプロキシ以外のソースで処理する場合は、AUTHENTICATED USING を指定します。この句は、GRANT CONNECT THROUGH *proxy* 句の一部のみに関連します。

PASSWORD プロキシに認証用のユーザーのデータベース・パスワードを設定する場合は、PASSWORD を指定します。プロキシはデータベースに依存し、パスワードに基づいてユーザーを認証します。

DISTINGUISHED NAME DISTINGUISHED NAME を指定すると、プロキシが識別名で識別され、グローバルに識別されたユーザーとして動作することが許可されます。

CERTIFICATE CERTIFICATE を指定すると、プロキシが証明書に含まれる識別名を持つ、グローバルに識別されたユーザーとして動作が許可されます。

DISTINGUISHED NAME と CERTIFICATE の両方の場合において、プロキシはすでに認証され、グローバル・データベース・ユーザーとして動作しています。

- `type` には、使用する証明書のタイプを指定します。`type` を指定しない場合、デフォルトは 'X.509' です。
- `version` には、使用する証明書のバージョンを指定します。`version` を指定しない場合、デフォルトは '3' です。

制限事項：この句は、REVOKE CONNECT THROUGH `proxy` 句の一部として指定できません。

参照：

- データベース・セキュリティの概要は、Oracle Security のドキュメントを参照してください。
- 中間層システムおよびプロキシ認証の詳細は、『Oracle9i データベース管理者ガイド』および『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

例

ユーザー識別の例 次の文は、デモ・データベース・ユーザー `oe` のパスワードを `lion` に、デフォルト表領域を `tbs_1` に変更します。

```
ALTER USER oe
  IDENTIFIED BY lion
  DEFAULT TABLESPACE tbs_1;
```

次の文は、`dw_manager` プロファイルをデモ・ユーザー `sh` に割り当てます。

```
ALTER USER sh
  PROFILE dw_manager;
```

後続のセッションでは、`sh` は `dw_manager` プロファイルの制限に従います。

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

```
ALTER USER sh
    DEFAULT ROLE ALL EXCEPT dw_manager;
```

sh の次のセッションの開始時には、dw_manager 以外で sh に直接付与されているすべてのロールが使用可能になります。

ユーザー認証の例 次の文は、デモ・ユーザー hr の認証方法を変更します。

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

次の文は、ユーザー hr のパスワードを期限切れにします。

```
ALTER USER hr PASSWORD EXPIRE;
```

PASSWORD EXPIRE を使用してデータベース・ユーザーのパスワードを期限切れにした場合、そのユーザー（または DBA）は、期限切れの後でデータベースにログインする際、パスワードを変更する必要があります。ただし、SQL*Plus などのツール製品を使用した場合、期限切れの後の最初のログイン時に、パスワードを変更できます。

プロキシ・ユーザーの例 次の文は、ユーザー app_user を変更します。例では、app_user がプロキシ・ユーザー sh を使用して接続できます。また、プロキシ sh を使用して接続したときに、app_user が dw_user ロールを使用可能にできます。

```
ALTER USER app_user
    GRANT CONNECT THROUGH sh
    WITH ROLE dw_user;
```

注意： 基本的な構文を示すため、サンプル・データベース Sales History のユーザー（sh）をプロキシとして使用します。通常、プロキシ・ユーザーは、アプリケーション・サーバーまたは中間層のエンティティに存在します。アプリケーション・サーバーを経由して、アプリケーション・ユーザーとデータベースの間のインタフェースを作成する場合の詳細は、『Oracle Call Interface プログラマーズ・ガイド』を参照してください。

参照：

- app_user ユーザーの作成方法については、15-35 ページの「[外部ユーザーの例](#)」を参照してください。
- dw_user ロールの作成方法については、13-74 ページの「[CREATE ROLE の例](#)」を参照してください。

次の文は、ユーザー `app_user` がプロキシ・ユーザー `sh` を使用して接続する権限を取り消します。

```
ALTER USER app_user REVOKE CONNECT THROUGH sh;
```

次の例は、プロキシ認証の他の方法を示します。

```
ALTER USER grant GRANT CONNECT THROUGH OAS1  
AUTHENTICATED USING PASSWORD;
```

```
ALTER USER green GRANT CONNECT THROUGH WebDB  
AUTHENTICATED USING DISTINGUISHED NAME;
```

```
ALTER USER brown GRANT CONNECT THROUGH WebDB  
AUTHENTICATED USING CERTIFICATE TYPE 'X.509' VERSION '3';
```

ALTER VIEW

用途

ALTER VIEW は、無効なビューを明示的に再コンパイルする場合に使用します。明示的に再コンパイルすると、実行前にコンパイル・エラーを検査できます。再コンパイルは、ビューの実表を変更した後で、その変更がビューまたはそのビューに依存するオブジェクトに影響していないかどうかを確認するときに便利です。

ALTER VIEW を使用すると、制約のビューを定義、変更または削除できます。

ALTER VIEW 文を発行した場合、指定したビューは有効か無効にかかわらず再コンパイルされます。また、そのビューに依存するすべてのローカル・オブジェクトが無効になります。

注意：

- この文を使用して既存のビュー定義を変更することはできません。ビューを再定義する場合、OR REPLACE キーワードを指定した CREATE VIEW を使用する必要があります。
 - 1 つ以上のマテリアライズド・ビューが参照しているビューを変更した場合、これらのマテリアライズド・ビューは無効になります。無効なマテリアライズド・ビューは、クエリー・リライトによって使用できません。また、リフレッシュすることもできません。
-

参照：

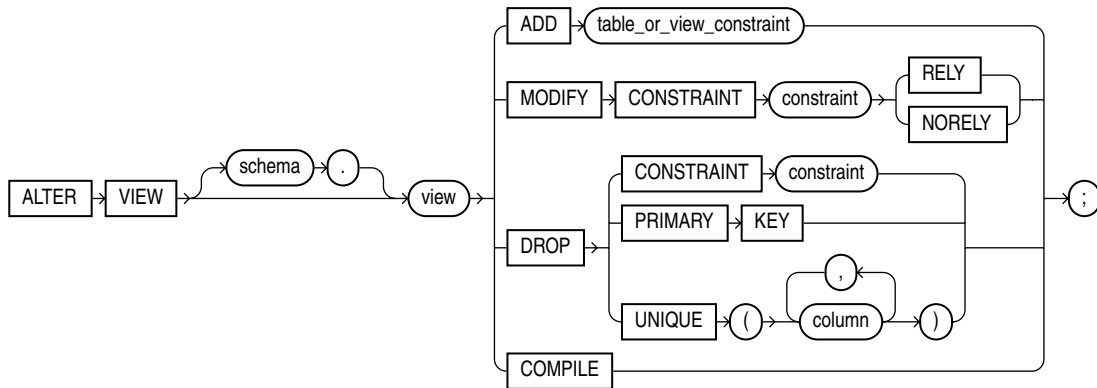
- ビューの再定義の詳細は、15-37 ページの「[CREATE VIEW](#)」を参照してください。
- 無効なマテリアライズド・ビュー・ログの再検証の詳細は、8-74 ページの「[ALTER MATERIALIZED VIEW](#)」を参照してください。
- データ・ウェアハウスの概要は、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。
- スキーマ・オブジェクト間の依存性については、『Oracle9i データベース概要』を参照してください。

前提条件

ビューが自分のスキーマ内にある必要があります。自分のスキーマ内にはない場合は、ALTER ANY TABLE システム権限が必要です。

構文

alter_view::=



table_or_view_constraint: 11-72 ページの「[constraint_clause](#)」を参照してください。

キーワードとパラメータ

schema

削除するビューが含まれているスキーマを指定します。*schema* を指定しない場合、ビューは自分のスキーマ内にあるとみなされます。

view

再コンパイルするビューの名前を指定します。

ADD 句

ADD 句を使用すると、*view* に制約を追加できます。

参照： 制約のビューおよび制限事項については、11-72 ページの「[constraint_clause](#)」を参照してください。

MODIFY CONSTRAINT 句

MODIFY CONSTRAINT 句を使用すると、既存の制約のビューの RELY または NORELY 設定を変更できます。

制限事項：一意制約または主キー制約が参照整合性制約の一部である場合、外部キーの削除または `view` の設定にあわせた変更をせずに、設定を変更することはできません。

参照： RELY および NORELY 設定の使用の詳細は、11-86 ページの「[RELY](#) | [NORELY](#)」を参照してください。

DROP 句

DROP を使用すると、既存の制約のビューを削除できます。

制限事項：一意制約または主キー制約がビューの参照整合性制約の一部である場合は削除できません。

COMPILE

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

例

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

```
ALTER VIEW bombay_inventory  
    COMPILE;
```

`bombay_inventory` の再コンパイル時にエラーが発生しなければ、`bombay_inventory` は有効になります。再コンパイル・エラーが発生した場合、エラー・メッセージが戻り、`bombay_inventory` 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、`bombay_inventory` を参照する、プロシージャ、ファンクション、パッケージ本体、ビューなどです。その後、明示的に再コンパイルせずに、これらのオブジェクトを参照した場合、Oracle は、実行時にそれらを暗黙的に再コンパイルします。

ANALYZE

用途

ANALYZE 使用すると、次の処理ができます。

- 索引または索引パーティション、表または表パーティション、索引構成表、クラスタあるいはスカラー・オブジェクト属性の統計情報を収集または削除します。
- 索引または索引パーティション、表または表パーティション、索引構成表、クラスタあるいはオブジェクト参照（REF）の構造を検証します。
- 表またはクラスタの移行行と連鎖行を識別します。

注意： ほとんどの統計情報収集には、ANALYZE より DBMS_STATS パッケージを使用することをお勧めします。このパッケージを使用すると、パラレルでの統計収集、パーティション化オブジェクトに対するグローバル統計収集、および他の方法での統計収集の詳細なチューニングをすることができます。また、コストベースのオブティマイザは、DBMS_STATS によって収集された統計情報のみを使用します。このパッケージの詳細は、『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

ただし、割合ではなく行数をサンプルとして抽出する場合、および次にあげるようなコストベースのオブティマイザに関連しない統計情報の収集には、DBMS_STATS より ANALYZE を使用します。

- VALIDATE 句、LIST CHAINED ROWS 句の使用
 - 空きリストのブロックに関する情報の収集
-

前提条件

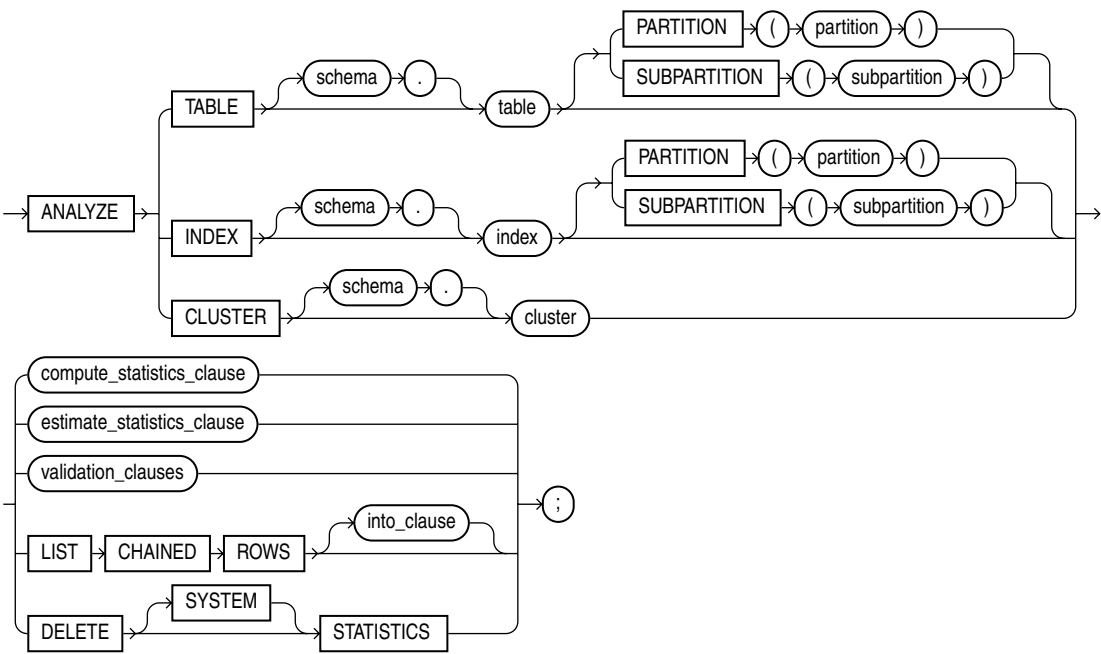
分析するスキーマ・オブジェクトがローカルである必要があります。自分のスキーマ内でない場合は、ANALYZE ANY システム権限が必要です。

表またはクラスタの連鎖行をリスト表へ入れる場合、このリスト表が自分のスキーマ内にある必要があります。自分のスキーマ内でない場合は、そのリスト表の INSERT 権限または INSERT ANY TABLE システム権限が必要です。

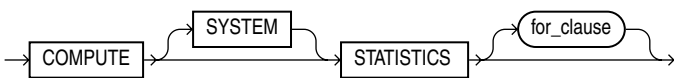
パーティション表の妥当性チェックを行う場合は、分析した ROWID を入れる表に対する INSERT 権限または INSERT ANY TABLE システム権限が必要です。

構文

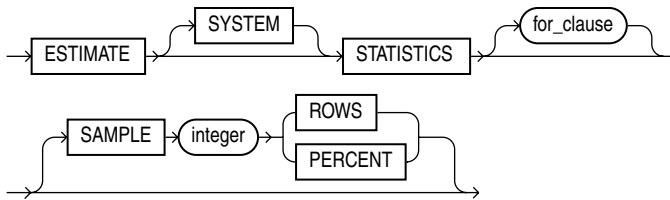
analyze::=



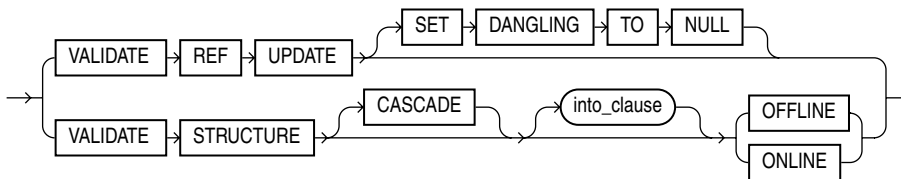
compute_statistics_clause::=



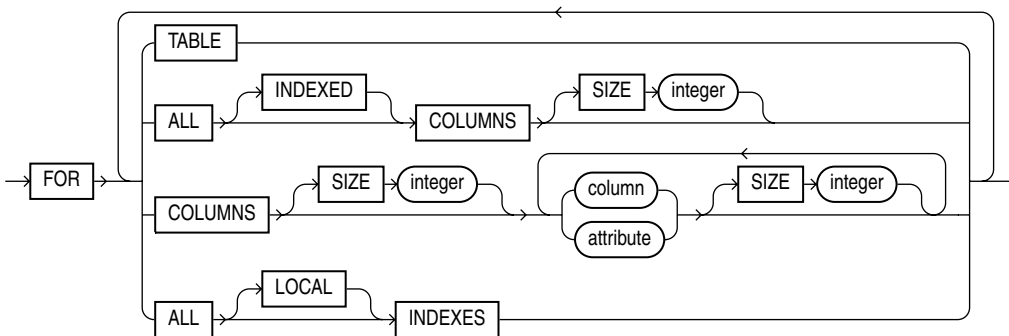
estimate_statistics_clause::=



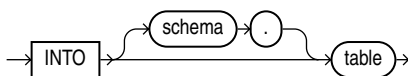
validation_clauses::=



for_clause::=



into_clause::=



キーワードとパラメータ

schema

索引、表またはクラスタが含まれているスキーマを指定します。*schema* を指定しない場合、索引、表またはクラスタは自分のスキーマ内にあるとみなされます。

INDEX *index*

分析する索引を指定します (*for_clause* を使用しない場合)。

索引については、次の統計情報が収集されます。アスタリスクが付いた統計は、常に厳密に計算されます。従来索引の場合、統計情報は、USER_INDEXES、ALL_INDEXES および DBA_INDEXES データ・ディクショナリ・ビューのカッコで示す列に表示されます。

- ルート・ブロックからリーフ・ブロック (BLEVEL) までの索引の深さ *
- リーフ・ブロックの数 (LEAF_BLOCKS)
- 個別索引値の数 (DISTINCT_KEYS)
- 索引の値ごとのリーフ・ブロックの平均数 (AVG_LEAF_BLOCKS_PER_KEY)
- (表に対する索引の) 索引の値ごとのデータ・ブロックの平均数 (AVG_DATA_BLOCKS_PER_KEY)
- クラスタ係数 (索引付きの値についての行が、どれだけ効率的に順序付けられているか) (CLUSTERING_FACTOR)

ドメイン索引の場合、索引に関連付けられた統計タイプに指定したユーザー定義統計収集ファンクションが、この文によってコールされます (11-46 ページの「[ASSOCIATE STATISTICS](#)」を参照)。ドメイン索引に関連付けられた統計タイプがない場合、その索引タイプに関連付けられた統計タイプが使用されます。索引またはその索引タイプの統計タイプがない場合、ユーザー定義統計情報は収集されません。ユーザー定義索引統計情報は、データ・ディクショナリ・ビュー USER_USTATS、ALL_USTATS および DBA_USTATS で STATISTICS 列に表示されます。

制限事項：IN_PROGRESS または FAILED のマークが付いたドメイン索引は分析できません。

参照：

- ドメイン索引の詳細は、12-58 ページの「[CREATE INDEX](#)」を参照してください。
- データ・ディクショナリ・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

TABLE table

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

表を分析すると、すべてのファンクション索引に発生する式について統計情報が収集されます。したがって、表を分析する前に、必ずファンクション索引を作成してください。

参照： ファンクション索引の詳細は、12-58 ページの「[CREATE INDEX](#)」を参照してください。

表を分析すると、LOADING または FAILED のマークが付いたドメイン索引はすべてスキップされます。

索引構成表の場合、マッピング表が分析され、その PCT_ACCESSSS_DIRECT 統計情報も計算されます。これらの統計情報は、マッピング表のローカル ROWID の一部として格納されたと推測されるデータ・ブロック・アドレスの精度を評価します。

表については、次の統計情報が収集されます。アスタリスクが付いた統計は、常に厳密に計算されます。表の統計情報（ドメイン索引の状態を含む）は、データ・ディクショナリ・ビュー USER_TABLES、ALL_TABLES および DBA_TABLES のカッコで示す列に表示されます。

- 行数 (NUM_ROWS)
- 最高水位標を下回るデータ・ブロックの数（現在データを含むか含まないにかかわらず、データを格納するようにフォーマットされているデータ・ブロックの数） * (BLOCKS)
- 未使用の表に対して割り当てられているデータ・ブロックの数 * (EMPTY_BLOCKS)
- 各データ・ブロックにおける使用可能な空き領域サイズの平均値（バイト単位） (AVG_SPACE)
- 連鎖行の数 (CHAIN_COUNT)
- 行のオーバーヘッドを含む、バイト単位での行の平均の長さ (AVG_ROW_LEN)

表の分析の制限事項

- ANALYZE を使用して、データ・ディクショナリ表の統計情報を収集しないでください。
- ANALYZE を使用して、一時表のデフォルト統計情報を収集しないでください。ただし、一時表の 1 つ以上の列とユーザー定義統計タイプを対応付けている場合、ANALYZE を使用して一時表のユーザー定義統計情報を収集できます（ANALYZE 使用前に対応付けが行われている必要があります）。

- REF、VARRAY、ネストした表、LOB（LOB は分析されず、スキップされる）、LONG またはオブジェクト型の列の統計情報は、計算および推定できません。ただし、このような列に統計タイプが対応付けられている場合は、ユーザー定義統計情報が収集されます。

参照：

- 11-46 ページの「[ASSOCIATE STATISTICS](#)」を参照してください。
- データ・ディクショナリ・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

PARTITION | SUBPARTITION

統計を収集するパーティションまたはサブパーティションを指定します。クラスタの分析時にこの句は使用できません。

`table` がコンポジット・パーティションのときに `PARTITION` を指定した場合、指定したパーティション内ですべてのサブパーティションが分析されます。

CLUSTER *cluster*

分析するクラスタを指定します。クラスタの統計情報を収集した場合、すべてのクラスタ表、およびクラスタ索引を含むすべての索引の統計も自動的に収集されます。

索引クラスタとハッシュ・クラスタには、単一クラスタ・キー (`AVG_BLOCKS_PER_KEY`) が使用するデータ・ブロックの平均数が収集されます。これらの統計情報は、データ・ディクショナリ・ビュー `ALL_CLUSTERS`、`USER_CLUSTERS` および `DBA_CLUSTERS` に表示されます。

参照： データ・ディクショナリ・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

compute_statistics_clause

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

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

ユーザー定義統計情報ではなく、システム統計情報のみを計算する場合は、**SYSTEM** を指定します。**SYSTEM** を省略すると、システムが生成した統計情報と統計タイプで宣言した収集ファンクションによって生成された統計情報の両方が収集されます。

参照：

- 統計収集ファンクションの作成の詳細は、『Oracle9i Data Cartridge Developer's Guide』を参照してください。
- これらの統計情報の使用方法の詳細は、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。

for_clause

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

FOR TABLE **FOR TABLE** を指定して、表および列の情報ではなく、表のみの統計情報が収集されるように制限します。

FOR COLUMNS **FOR COLUMNS** を指定して、すべての列または属性の列統計情報ではなく、指定した列およびスカラー・オブジェクト属性のみの列統計情報が収集されるように統計収集ファンクションを制限します。**attribute** には、オブジェクト内の項目の修飾列名を指定します。

FOR ALL COLUMNS **FOR ALL COLUMNS** を指定して、すべての列およびスカラー・オブジェクト属性の列統計情報を収集します。

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

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

Oracle は、次の列統計情報を収集します。

- 列全体として、重複していない値の数
- 各区間の最大値と最小値

参照： ヒストグラムの詳細は、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』および 11-44 ページの「**ヒストグラムの例**」を参照してください。

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

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

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

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

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

FOR ALL INDEXES `FOR ALL INDEXES` を指定して、表に関連付けられたすべての索引を分析します。

FOR ALL LOCAL INDEXES `FOR ALL LOCAL INDEXES` を指定して、すべてのローカル索引パーティションを分析します。`PARTITION` 句および `INDEX` が指定されている場合、キーワード `LOCAL` を指定する必要があります。

SIZE ヒストグラムのバケットの最大数を指定します。デフォルト値は 75 で、最小値は 1、最大値は 254 です。

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

estimate_statistics_clause

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

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

ユーザー定義統計情報ではなく、システム統計情報のみを推定する場合は、SYSTEM を指定します。SYSTEM を省略すると、システムが生成した統計情報と統計タイプで宣言した収集ファンクションによって生成された統計情報の両方が推定されます。

参照： 統計収集ファンクションの作成の詳細は、『Oracle9i Data Cartridge Developer's Guide』を参照してください。

for_clause 11-36 ページの「[compute_statistics_clause](#)」を参照してください。

SAMPLE 統計情報を推定するために、分析対象のオブジェクトからサンプルとして抽出されるデータ量を指定します。このパラメータを指定しない場合、サンプルとして 1064 行が抽出されます。

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

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

参照： これらの統計情報の使用方法の詳細は、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。

validation_clauses

validation_clauses を使用すると、REF および分析したオブジェクトの構造が検証されます。

VALIDATE REF UPDATE 句

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

SET DANGLING TO NULL SET DANGLING TO NULL を指定すると、指定した表内の REF が（範囲が限定されるかどうかにかかわらず）無効なオブジェクトまたは存在しないオブジェクトを指している場合に、REF が NULL に設定されます。

注意： 表の所有者が参照先オブジェクトに対する SELECT オブジェクト権限を持っていない場合、このオブジェクトは無効とみなされ、NULL に設定されます。この結果、オブジェクトに対して適切な権限があるユーザーによって発行された問合せであっても、問合せでこれらの REF を使用することはできません。

VALIDATE STRUCTURE

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

- 表に対して、表のそれぞれのデータ・ブロックと行の整合性を検証します。索引構成表の場合、表の主キー索引に対する圧縮統計情報（最適なプレフィックス圧縮件数）も生成されます。
- クラスタに対して、自動的にクラスタ表の構造を検証します。
- パーティションに対して、行が適切なパーティションに属するかどうかを検証します。行が正しく照合されなかった場合は、ROWID が INVALID_ROWS 表に挿入されます。
- 一時表に対して、現行のセッション中に表の構造および索引を検証します。

- 索引に対して、索引のそれぞれのデータ・ブロックの整合性を検証し、ブロックの破損をチェックします。この句は、表のそれぞれの行が索引エントリを持っていること、またはそれぞれの索引エントリが表の行を指していることを確認するわけではありません。これらを確認する場合は、**CASCADE** 句を使用して表の構造を検証します。

通常のすべての索引用の圧縮統計情報（最適なプレフィックス圧縮件数）も計算されます。

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

参照： これらのビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

オブジェクトの構造に障害がある場合は、エラー・メッセージが戻ります。この場合、オブジェクトを削除して作成しなおす必要があります。

INTO **VALIDATE STRUCTURE** の **INTO** 句は、パーティション表のみに有効です。正しく照合されなかった行を持つパーティションの **ROWID** を格納するリスト表を指定します。*schema* を指定しない場合、このリスト表は自分のスキーマ内にあるとみなされます。この句自体を指定しない場合、表の名前は **INVALID_ROWS** になります。この表を作成するために使用する SQL スクリプトは **UTLVALID.SQL** です。

CASCADE 表またはクラスタに関連付けられた索引の構造を検証する場合は、**CASCADE** を指定します。表を検証するときにこの句を指定すると、その表の索引も検証されます。クラスタを検証するときにこの句を指定した場合、クラスタ化表のすべての索引（クラスタ索引を含む）が検証されます。

この句を使用して使用可能な（以前は使用禁止であった）ファンクション索引を検証すると、検証エラーになる場合があります。この場合は、索引を再構築する必要があります。

ONLINE | OFFLINE **ONLINE** を指定すると、Oracle がオブジェクトの DML 操作中に検証を実行できるようになります。Oracle は、並行して操作が行える程度に、実行する検証の量を減らします。

OFFLINE を指定すると、実行する検証の量を最大化します。この設定は、検証中に **INSERT**、**UPDATE** および **DELETE** 文がオブジェクトに平行してアクセスすることを防ぎますが、問合せは許可されます。これはデフォルトです。

制限事項： **ONLINE** は、クラスタ化されているオブジェクトを分析する場合に指定できません。

LIST CHAINED ROWS

LIST CHAINED ROWS によって、分析対象の表またはクラスタの移行行および連鎖行を識別できます。索引の分析時にこのオプションは使用できません。

INTO 句には、移行行および連鎖行をリストする表を指定します。*schema* を指定しない場合、この表が自分のスキーマにあるものとみなされます。この句自体を指定しない場合、表の名前は CHAINED_ROWS になります。このリスト表はローカル・データベース内にある必要があります。

次のいずれかのスクリプトを使用して、CHAINED_ROWS 表を作成できます。

- UTLCHAIN.SQL は、物理 ROWID を使用します。そのため、行は、索引構成表からではなく従来表から収集されます（次の注意を参照）。
- UTLCHN1.SQL は、ユニバーサル ROWID を使用します。そのため、行は、従来表および索引構成表の両方から収集されます。

独自の連鎖行表を作成する場合、この 2 つのスクリプトのいずれかで規定されるフォーマットに従う必要があります。

注意： ユニバーサル ROWID ではなく、主キーに基づく索引構成表を分析する場合、索引構成表ごとに別の連鎖行表を作成し、主キー記憶域を確保する必要があります。まず、SQL スクリプトの DBMSIOTC.SQL および PRVTIOTC.PLB を使用して、BUILD_CHAIN_ROWS_TABLE プロシージャを定義します。次に、このプロシージャを実行して、索引構成表の IOT_CHAINED_ROWS 表を作成します。

参照：

- これらのスクリプトを使用する場合の互換性については、『Oracle9i データベース移行ガイド』を参照してください。
- パッケージ化された SQL スクリプトの詳細は、『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の DBMS_IOT パッケージを参照してください。
- 移行行および連鎖行の削除については、『Oracle9i データベース管理者ガイド』を参照してください。

DELETE STATISTICS

DELETE STATISTICS は、現在データ・ディクショナリに格納されている、分析対象のオブジェクトの統計情報を削除します。Oracle に統計情報を使用させないようにする場合、この文を使用します。

表を指定してこの句を使用した場合、指定した表のすべての索引の統計情報も自動的に削除されます。クラスタを指定してこの句を使用した場合、指定したクラスタのすべての表、およびこれらの表のすべての索引（クラスタ索引を含む）の統計情報が自動的に削除されます。

ユーザー定義統計情報ではなく、システム統計情報のみを削除する場合は、SYSTEM を指定します。SYSTEM を省略すると、オブジェクトのユーザー定義列または索引統計情報が収集された場合、Oracle は、統計情報を収集するために使用された情報タイプに指定されている統計削除ファンクションを起動して、ユーザー定義統計情報も削除します。

例

統計情報の計算例 次の文は、デモ表 oe.orders の統計情報を計算します。

```
ANALYZE TABLE orders COMPUTE STATISTICS;
```

次の文は、デモ表 oe.orders のシステム統計情報のみを計算します。

```
ANALYZE TABLE orders COMPUTE SYSTEM STATISTICS;
```

次の文は、スカラー・オブジェクト属性の統計情報を計算します。

```
ANALYZE TABLE customers COMPUTE STATISTICS  
FOR COLUMNS cust_address.postal_code;
```

統計情報の推定例 次の文は、デモ表 oe.orders とそのすべての索引の統計情報を推定します。

```
ANALYZE TABLE orders ESTIMATE STATISTICS;
```

統計情報の削除例 次の文は、デモ表 oe.orders とデータ・ディクショナリにあるそのすべての索引の統計情報を削除します。

```
ANALYZE TABLE orders DELETE STATISTICS;
```

ヒストグラム例 次の文は、デモ表 `hr.employees` の `salary` 列について 10 区間のヒストグラムを作成します。

```
ANALYZE TABLE employees
  COMPUTE STATISTICS FOR COLUMNS salary SIZE 10;

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

SELECT NUM_DISTINCT, NUM_BUCKETS, SAMPLE_SIZE
  FROM USER_TAB_COLUMNS
  WHERE TABLE_NAME = 'EMPLOYEES' AND COLUMN_NAME = 'SALARY';

NUM_DISTINCT NUM_BUCKETS SAMPLE_SIZE
-----
          57           10         107
```

表のサイズによっては、`ANALYZE` 文で 10 バケットを指定しても、その `ANALYZE` 文で指定したバケットより少ない数が作成される場合もあります。詳細は、11-39 ページの「[SIZE](#)」を参照してください。

また、表の単一のパーティションのヒストグラムも収集できます。次の文は、デモ表 `sh.sales` のパーティション `sales_q2_2000` を分析します。

```
ANALYZE TABLE sales PARTITION (sales_q2_2000) COMPUTE STATISTICS;
```

索引の分析例 次の文は、デモ索引 `oe.inv_product_ix` の構造を検証します。

```
ANALYZE INDEX inv_product_ix VALIDATE STRUCTURE;
```

表の検証例 次の文は、サンプル表 `hr.employees` およびそのすべての索引を分析します。

```
ANALYZE TABLE employees VALIDATE STRUCTURE CASCADE;
```

表に対する `VALIDATE REF UPDATE` 句は、指定した表の `REF` を検証します。また、それぞれの `REF` の `ROWID` 部分をチェックし、それを真の `ROWID` と比較します。その結果、`ROWID` が誤っていると判断されると、`ROWID` 部分が正しくなるように `REF` が更新されます。

次の文は、デモ表 `oe.customers` の `REF` を検証します。

```
ANALYZE TABLE customers VALIDATE REF UPDATE;
```

次の文は、DML 操作を同時に実行することを許可して、デモ表 `oe.customers` の構造を検証します。

```
ANALYZE TABLE customers VALIDATE STRUCTURE ONLINE;
```

クラスタの分析例 次の文は、personnel クラスタ（12-9 ページの「[クラスタの作成例](#)」で作成）、そのすべての表、クラスタ索引を含むすべての索引を分析します。

```
ANALYZE CLUSTER personnel
      VALIDATE STRUCTURE CASCADE;
```

連鎖行のリスト例 次の文は、orders 表のすべての連鎖行についての情報を収集します。

```
ANALYZE TABLE orders
      LIST CHAINED ROWS INTO chained_rows;
```

前述の文では、情報は表 chained_rows に格納されます。次の問合せでその行を検証できます（表に連鎖行が含まれない場合は、行は戻されません）。

```
SELECT owner_name, table_name, head_rowid, analyze_timestamp
      FROM chained_rows;
```

OWNER_NAME	TABLE_NAME	HEAD_ROWID	ANALYZE_TIMESTAMP
-----	-----	-----	-----
OE	ORDERS	AAAAZzAABAAABrXAAA	25-SEP-2000

ASSOCIATE STATISTICS

用途

ASSOCIATE STATISTICS 文を使用すると、統計収集、選択性またはコストに関するファンクションが含まれた統計タイプ（またはデフォルトの統計）を、1つ以上の列、スタンドアロン・ファンクション、パッケージ、型、ドメイン索引または索引タイプに関連付けることができます。

現在のすべての統計タイプの関連付けの一覧については、USER_ASSOCIATIONS データ・ディクショナリ・ビューに問い合わせます。統計情報に関連付けられたオブジェクトを分析する場合、USER_USTATS ビューでその関連性を問い合わせることができます。

参照： ANALYZE が関連性で使用する優先順位については、11-31 ページの「ANALYZE」を参照してください。

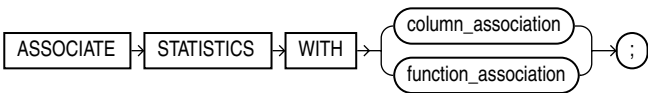
前提条件

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

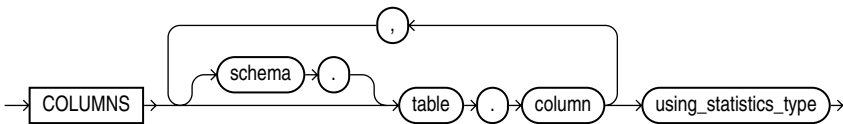
参照： 型の定義の詳細は、15-3 ページの「CREATE TYPE」を参照してください。

構文

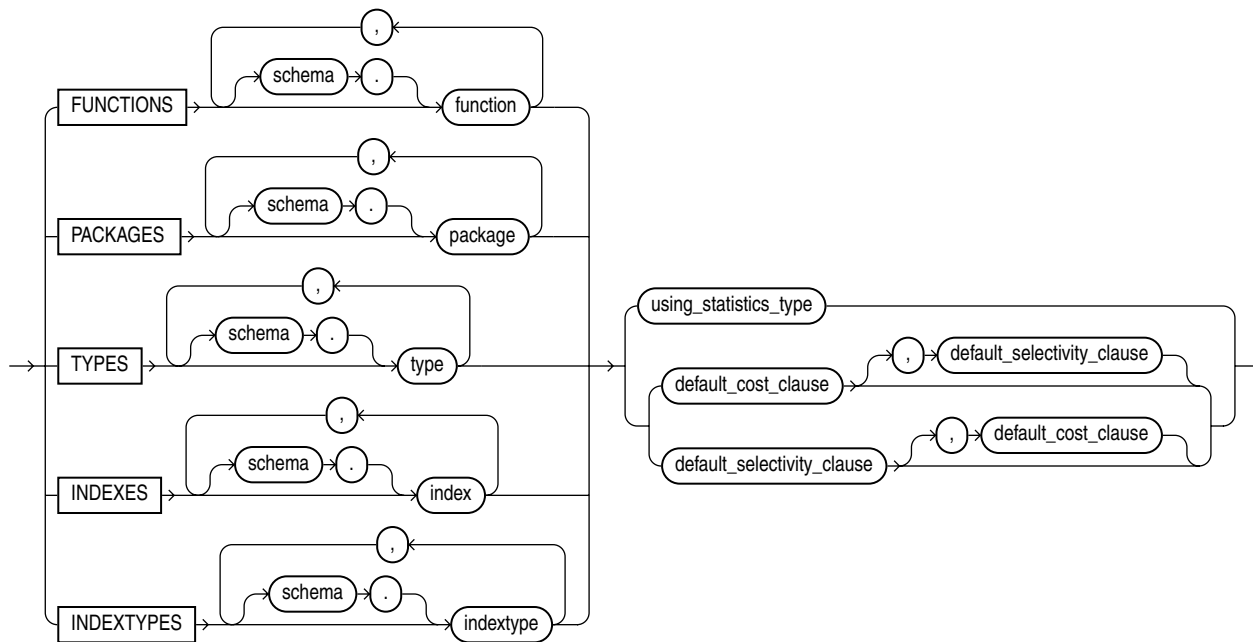
associate_statistics::=



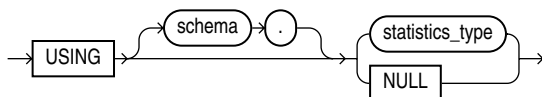
column_association::=



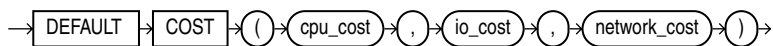
function_association::=



using_statistics_type::=



default_cost_clause::=



default_selectivity_clause::=



キーワードとパラメータ

column_association

1 つ以上の表の列を指定します。 *schema* を指定しない場合、表が自分のスキーマ内にあるとみなされます。

function_association

1 つ以上のスタンドアロン・ファンクション、パッケージ、ユーザー定義データ型、ドメイン索引または索引タイプを指定します。 *schema* で指定しない場合、オブジェクトは自分のスキーマ内にあるとみなされます。

- **FUNCTIONS** は、スタンドアロン・ファンクションのみを参照し、メソッド型または組み込みファンクションは参照しません。
- **TYPES** はユーザー定義型のみを参照し、組み込み SQL データ型は参照しません。

制限事項：すでに関連性を定義してあるオブジェクトには指定できません。まず、このオブジェクトと統計情報の関連性を取り消す必要があります。

参照： 15-57 ページの「[DISASSOCIATE STATISTICS](#)」を参照してください。

using_statistics_type

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

NULL キーワードは、統計情報を列または索引に関連付ける場合のみに有効です。統計タイプをオブジェクト型に関連付ける場合は、そのオブジェクト型の列は、統計タイプを継承します。同様に、統計タイプを索引タイプに関連付ける場合は、索引タイプの索引インスタンスは統計タイプを継承します。列または索引に別の統計タイプを関連付けると、この継承をオーバーライドできます。あるいは、列または索引に統計タイプを関連付けない場合は、*using_statistics_type* 句で NULL を指定します。

制限事項：ファンクション、パッケージ、型または索引タイプには NULL を指定できません。

参照： 統計収集ファンクションの作成の詳細は、『Oracle9i Data Cartridge Developer's Guide』を参照してください。

default_cost_clause

スタンドアロン・ファンクション、パッケージ、型、ドメイン索引または索引タイプのデフォルトのコストを指定します。この句を指定する場合、CPU コスト、I/O コスト、ネットワーク・コストの順でそれぞれに対して 1 つの数を指定する必要があります。それぞれのコストは、ファンクションまたはメソッドを一度実行した場合、またはドメイン索引へ一度アクセスした場合の値です。指定できる値は 0（ゼロ）以上の整数です。

default_selectivity_clause

スタンドアロン・ファンクション、型、パッケージまたはユーザー定義演算子が指定された述語に対するデフォルトの選択性をパーセントで指定します。`default_selectivity` は、0 ～ 100 の整数値である必要があります。範囲外の場合は無視されます。

制限事項：DEFAULT SELECTIVITY は、ドメイン索引または索引タイプに指定できません。

例

スタンドアロン・ファンクションの例 次の文は、スタンドアロン・パッケージ `emp_mgmt` (13-49 ページの「[CREATE PACKAGE の例](#)」で作成) への関連性を作成します。

```
ASSOCIATE STATISTICS WITH PACKAGES emp_mgmt DEFAULT SELECTIVITY 10;
```

デフォルト・コストの例 次の文は、ドメイン索引 `t_a` を使用して任意の述語を実装した場合、常に CPU コストは 100、I/O は 5、およびネットワーク・コストは 0 になるように指定します。

```
ASSOCIATE STATISTICS WITH INDEXES t_a DEFAULT COST (100,5,0);
```

オブティマイザは、コスト・ファンクションをコールするかわりに、これらのデフォルト・コストをそのまま使用します。

AUDIT

用途

AUDIT 文を使用すると、次の処理ができます。

- 後続のユーザー・セッションでの SQL 文の状態変化の監査。特定の SQL 文または特定のシステム権限によって許可されたすべての SQL 文の状態変化を監査します。SQL 文操作の監査は、後続セッションにのみ適用され、現行のセッションには適用されません。
- 特定のスキーマ・オブジェクトに対する操作の監査。スキーマ・オブジェクト操作の監査は、後続のセッションと同様に、現行のセッションにも適用されます。

参照：

- 値に基づいた監査方針を作成および管理する DBMS_FGA パッケージの詳細は、『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。
- SQL 文の監査を使用禁止にする場合の詳細は、16-80 ページの「[NOAUDIT](#)」を参照してください。

前提条件

SQL 文の状態変化を監査するには、AUDIT SYSTEM システム権限が必要です。

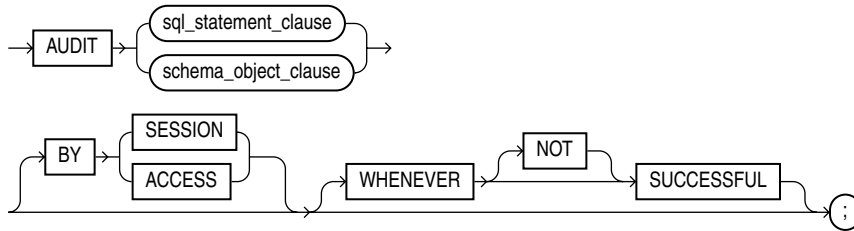
スキーマ・オブジェクト操作を監査するためには、監査対象のオブジェクトが自分のスキーマにあるか、AUDIT ANY システム権限が必要です。また、監査の対象とするオブジェクトがディレクトリ・オブジェクトの場合は、それが自分で作成したものであっても、AUDIT ANY システム権限が必要です。

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

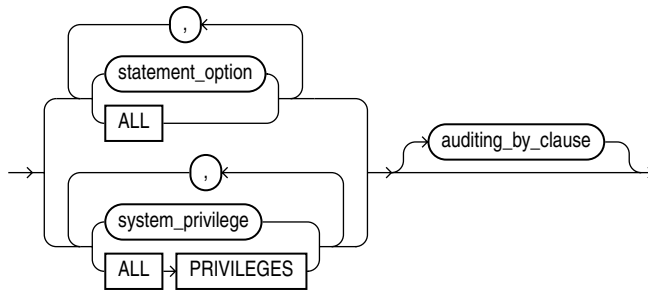
参照： AUDIT_TRAIL パラメータの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

構文

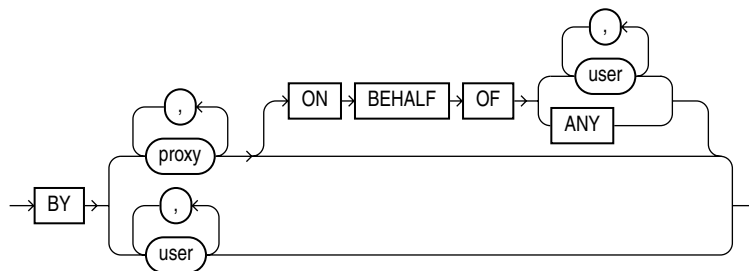
audit::=



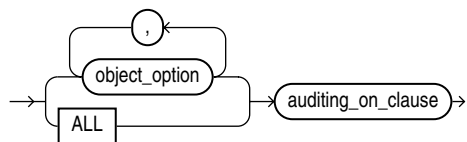
sql_statement_clause::=



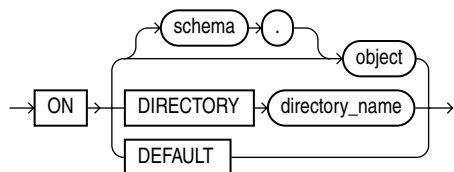
auditing_by_clause::=



schema_object_clause::=



auditing_on_clause::=



キーワードとパラメータ

sql_statement_clause

`sql_statement_clause` を使用し、SQL 文を監査します。

statement_option

特定の SQL 文を監査するには、文オプションを指定します。

監査されるたびに、次の情報を持つ監査レコードが生成されます。

- 操作を行ったユーザー
- 操作の種類
- 操作に関連するオブジェクト
- 操作の日付と時刻

監査レコードは、監査証跡に書き込まれます。監査証跡とは、監査レコードが入っているデータベースの表です。データ・ディクショナリ・ビューを問い合せて監査証跡を調べることによって、データベース・アクティビティを再検討できます。

参照：

- 文オプションおよびそれによって監査される SQL 文のリストは、11-56 ページの表 11-1 および 11-58 ページの表 11-2 を参照してください。
- データ・ディクショナリ・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

system_privilege

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

多くの個々のシステム権限を指定するのではなく、ロール CONNECT、RESOURCE および DBA を指定できます。これは、すべてのシステム権限がそのロールに付与されていることを監査することと同じです。

Oracle には、システム権限と文オプションをまとめて指定するための、次の 2 つのショートカットが用意されています。

ALL ALL は、表 11-1 のすべての文オプションを監査しますが、表 11-2 の追加文オプションを監査しません。

ALL PRIVILEGES システム権限を監査するためには、ALL PRIVILEGES を指定します。

注意： ロールまたはショートカットでなく、監査に個々のシステム権限および文オプションを指定することをお勧めします。ロールおよびショートカットに含まれるシステム権限および文オプションは、リリースごとに異なり、Oracle の今後のバージョンでサポートされない場合があります。

参照：

- すべてのシステム権限とそれによって許可される SQL 文のリストは、表 16-1 「システム権限」を参照してください。
- CONNECT、RESOURCE および DBA ロールの詳細は、16-31 ページの「GRANT」を参照してください。

auditing_by_clause

auditing_by_clause は、特定のユーザーが発行する SQL 文のみを監査します。この句を指定しない場合、すべてのユーザー文が監査されます。

BY user この句は、特定のユーザーによって発行された SQL 文のみを監査するように制限します。

BY proxy この句は、特定のプロキシによって発行された SQL 文のみを監査するように制限します。

参照： プロキシおよびデータベースでプロキシを使用する場合の詳細は、『Oracle9i データベース概要』を参照してください。

ON BEHALF OF *user* は、特定のユーザーのプロキシとして実行された文を監査することを指定します。ANY は、すべてのユーザーのプロキシとして実行された文を監査することを指定します。

schema_object_clause

schema_object_clause は、スキーマ・オブジェクトの操作を監査します。

object_option

監査の対象とする操作を指定します。11-59 ページの表 11-3 に、各オブジェクト・オプションとそれが適用されるオプションのタイプを示します。それぞれのオブジェクト・オプションには、監査の対象となる SQL 文を指定します。たとえば、ALTER オプションを指定して表の監査を選択した場合、その表に対して発行される ALTER TABLE 文がすべて監査されます。また、SELECT オプションを指定して順序の監査を選択した場合、その順序の値を使用するすべての文が監査されます。

ALL

ALL をショートカットに指定することは、オブジェクト・タイプに適用できるオプションをすべて指定することと同じです。

auditing_on_clause

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

schema 監査の対象として選択されたオブジェクトが定義されているスキーマを指定します。*schema* を指定しない場合、オブジェクトは、自分のスキーマ内に定義されているものとみなされます。

object 監査するオブジェクトの名前を指定します。オブジェクトは、表、ビュー、順序、ストアド・プロシージャ、ストアド・ファンクション、ストアド・パッケージ、マテリアライズド・ビューまたはライブラリのいずれかである必要があります。

表、ビュー、順序、プロシージャ、ストアド・ファンクション、パッケージまたはマテリアライズド・ビューについては、それぞれシノニムも指定できます。

ON DEFAULT ON DEFAULT を指定して、オブジェクトを作成した後に特定のオブジェクト・オプションをデフォルト・オブジェクト・オプションとして構築します。デフォルト監査オプションを指定した場合、その後作成されるオブジェクトに対して、これらのオプションが自動的に適用され、監査が行われます。ビューに対するデフォルト監査オプションは、常に、そのビューの実表に対する監査オプションの論理和となります。

ALL_DEF_AUDIT_OPTS データ・ディクショナリ・ビューを問い合わせることによって、現在のデフォルト監査オプションを表示できます。

デフォルト監査オプションを変更した場合でも、以前作成したオブジェクトの監査オプションはそのまま残ります。AUDIT 文の ON 句にオブジェクトを指定した場合のみ、既存のオブジェクトの監査オプションを変更できます。

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

BY SESSION

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

BY ACCESS

監査された各文および操作について、1 つのレコードを書き込む場合は、BY ACCESS を指定します。

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

DDL 文以外の SQL 文を監査する文オプションとシステム権限には、BY SESSION と BY ACCESS のどちらでも指定できます。デフォルトは BY SESSION です。

WHENEVER [NOT] SUCCESSFUL

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

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

この句を省略すると、処理結果にかかわらず監査を実行します。

監査オプションの表

表 11-1 データベース・オブジェクトの文監査オプション

文オプション	SQL 文と操作
CLUSTER	CREATE CLUSTER AUDIT CLUSTER DROP CLUSTER TRUNCATE CLUSTER
CONTEXT	CREATE CONTEXT DROP CONTEXT
DATABASE LINK	CREATE DATABASE LINK DROP DATABASE LINK
DIMENSION	CREATE DIMENSION ALTER DIMENSION DROP DIMENSION
DIRECTORY	CREATE DIRECTORY DROP DIRECTORY
INDEX	CREATE INDEX ALTER INDEX DROP INDEX
NOT EXISTS	指定したオブジェクトが存在しない場合に失敗するすべての SQL 文
PROCEDURE ^a	CREATE FUNCTION CREATE LIBRARY CREATE PACKAGE CREATE PACKAGE BODY CREATE PROCEDURE DROP FUNCTION DROP LIBRARY DROP PACKAGE DROP PROCEDURE
PROFILE	CREATE PROFILE ALTER PROFILE DROP PROFILE
PUBLIC DATABASE LINK	CREATE PUBLIC DATABASE LINK DROP PUBLIC DATABASE LINK

表 11-1 データベース・オブジェクトの文監査オプション (続き)

文オプション	SQL 文と操作
PUBLIC SYNONYM	CREATE PUBLIC SYNONYM DROP PUBLIC SYNONYM
ROLE	CREATE ROLE ALTER ROLE DROP ROLE SET ROLE
ROLLBACK SEGMENT	CREATE ROLLBACK SEGMENT ALTER ROLLBACK SEGMENT DROP ROLLBACK SEGMENT
SEQUENCE	CREATE SEQUENCE DROP SEQUENCE
SESSION	Logons
SYNONYM	CREATE SYNONYM DROP SYNONYM
SYSTEM AUDIT	AUDIT <i>sql_statements</i> NOAUDIT <i>sql_statements</i>
SYSTEM GRANT	GRANT <i>system_privileges_and_roles</i> REVOKE <i>system_privileges_and_roles</i>
TABLE	CREATE TABLE DROP TABLE TRUNCATE TABLE
TABLESPACE	CREATE TABLESPACE ALTER TABLESPACE DROP TABLESPACE
TRIGGER	CREATE TRIGGER ALTER TRIGGER ENABLE および DISABLE 句付き DROP TRIGGER ALTER TABLE ENABLE ALL TRIGGERS 句付き DISABLE ALL TRIGGERS 句付き

表 11-1 データベース・オブジェクトの文監査オプション（続き）

文オプション	SQL 文と操作
TYPE	CREATE TYPE
	CREATE TYPE BODY
	ALTER TYPE
	DROP TYPE
	DROP TYPE BODY
USER	CREATE USER
	ALTER USER
	DROP USER
VIEW	CREATE VIEW
	DROP VIEW

^a Java スキーマ・オブジェクト（ソース、クラスおよびリソース）は、SQL 文の監査ではプロシージャと同じであるとみなされます。

表 11-2 SQL 文のその他の文監査オプション

文オプション	SQL 文と操作
ALTER SEQUENCE	ALTER SEQUENCE
ALTER TABLE	ALTER TABLE
COMMENT TABLE	COMMENT ON TABLE <i>table</i> , <i>view</i> , <i>materialized view</i>
	COMMENT ON COLUMN <i>table.column</i> , <i>view.column</i> , <i>materialized view.column</i>
DELETE TABLE	DELETE FROM <i>table</i> , <i>view</i>
EXECUTE PROCEDURE	CALL
	プロシージャまたはファンクションの実行。または、変数、ライブラリ、パッケージ内のまたはカーソルへのアクセス。
GRANT DIRECTORY	GRANT <i>privilege</i> ON <i>directory</i>
	REVOKE <i>privilege</i> ON <i>directory</i>
GRANT PROCEDURE	GRANT <i>privilege</i> ON <i>procedure</i> , <i>function</i> , <i>package</i>
	REVOKE <i>privilege</i> ON <i>procedure</i> , <i>function</i> , <i>package</i>
GRANT SEQUENCE	GRANT <i>privilege</i> ON <i>sequence</i>
	REVOKE <i>privilege</i> ON <i>sequence</i>

表 11-2 SQL 文のその他の文監査オプション（続き）

文オプション	SQL 文と操作
GRANT TABLE	GRANT <i>privilege</i> ON <i>table, view, materialized view</i> . REVOKE <i>privilege</i> ON <i>table, view, materialized view</i>
GRANT TYPE	GRANT <i>privilege</i> ON TYPE REVOKE <i>privilege</i> ON TYPE
INSERT TABLE	INSERT INTO <i>table, view</i>
LOCK TABLE	LOCK TABLE <i>table, view</i>
SELECT SEQUENCE	<i>sequence.CURRVAL</i> または <i>sequence.NEXTVAL</i> を含む文
SELECT TABLE	SELECT FROM <i>table, view, materialized view</i>
UPDATE TABLE	UPDATE <i>table, view</i>

表 11-3 オブジェクト監査オプション

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

^a Java スキーマ・オブジェクト（ソース、クラスおよびリソース）は、監査オプションではプロシージャ、ファンクションおよびパッケージと同じであるとみなされます。

例

ロールに関連する SQL 文の監査例 次の文は、ロールの作成、変更、削除または設定を行う各 SQL 文が正常に終了したかどうかにかかわらず、それらの文について監査を行います。

```
AUDIT ROLE;
```

次の文は、ロールの作成、変更、削除または設定を行う、正常に終了した各 SQL 文ごとに監査を行います。

```
AUDIT ROLE  
    WHENEVER SUCCESSFUL;
```

次の文は、Oracle エラーが発生した CREATE ROLE 文、ALTER ROLE 文、DROP ROLE 文または SET ROLE 文について監査を行います。

```
AUDIT ROLE  
    WHENEVER NOT SUCCESSFUL;
```

問合せおよび更新を行う SQL 文の監査例 次の文は、表の問合せまたは更新を実行する文について監査を行います。

```
AUDIT SELECT TABLE, UPDATE TABLE;
```

次の文は、ユーザー hr および oe が発行する、表やビューの問合せまたは更新を実行する文について監査を行います。

```
AUDIT SELECT TABLE, UPDATE TABLE  
    BY hr, oe;
```

削除の監査例 次の文は、DELETE ANY TABLE システム権限で発行された文について監査を行います。

```
AUDIT DELETE ANY TABLE;
```

ディレクトリに関連する文の監査例 次の文は、CREATE ANY DIRECTORY システム権限で発行された文について監査を行います。

```
AUDIT CREATE ANY DIRECTORY;
```

CREATE ANY DIRECTORY システム権限を使用しない CREATE DIRECTORY（および DROP DIRECTORY）文を監査する場合は、次の文を発行します。

```
AUDIT DIRECTORY;
```

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

```
AUDIT SELECT
    ON hr.employees;
```

次の文は、スキーマ hr 内の employees 表を問い合わせ正常に終了した各文について監査を行います。

```
AUDIT SELECT
    ON hr.employees
    WHENEVER SUCCESSFUL;
```

次の文は、スキーマ hr 内の employees 表を問い合わせエラーが発生した SQL 文について監査を行います。

```
AUDIT SELECT
    ON hr.employees
    WHENEVER NOT SUCCESSFUL;
```

表の挿入および更新の監査例 次の文は、スキーマ oe 内の customers 表に対して行を挿入または更新するそれぞれの文について監査を行います。

```
AUDIT INSERT, UPDATE
    ON oe.customers;
```

順序に対するすべての操作の監査例 次の文は、スキーマ hr 内の employees_seq 順序に対する操作を行うすべての文について監査を行います。

```
AUDIT ALL
    ON hr.employees_seq;
```

この文は、順序に対して操作を行う次の文について監査を行うため、ALL ショートカットを使用しています。

- ALTER SEQUENCE
- AUDIT
- GRANT
- 疑似列 CURRVAL または NEXTVAL を使用して、順序の値にアクセスするすべての文

ディレクトリに対する読み込み操作の監査例 次の文は、bfile_dir ディレクトリからファイルを読み込む各文について監査を行います。

```
AUDIT READ ON DIRECTORY bfile_dir;
```

デフォルト監査操作の設定例 次の文は、その後作成されるオブジェクトについて、デフォルト監査オプションを指定します。

```
AUDIT ALTER, GRANT, INSERT, UPDATE, DELETE  
ON DEFAULT;
```

その後作成されるオブジェクトについては、監査機能が使用可能な場合、指定したオプションによって自動的に次の監査が行われます。

- 表を作成した場合、その表に対して発行される ALTER 文、GRANT 文、INSERT 文、UPDATE 文または DELETE 文が自動的に監査されます。
- ビューを作成した場合、そのビューに対して発行される GRANT 文、INSERT 文、UPDATE 文または DELETE 文が自動的に監査されます。
- 順序を作成した場合、その順序に対して発行される ALTER 文または GRANT 文が自動的に監査されます。
- プロシージャ、パッケージまたはファンクションを作成した場合、それらに対して発行される ALTER 文または GRANT 文が自動的に監査されます。

CALL

用途

CALL 文を使用すると、SQL 内から**ルーチン**（スタンドアロン・プロシージャ、スタンドアロン・ファンクション、あるいは型またはパッケージ内で定義されたプロシージャまたはファンクション）を実行できます。

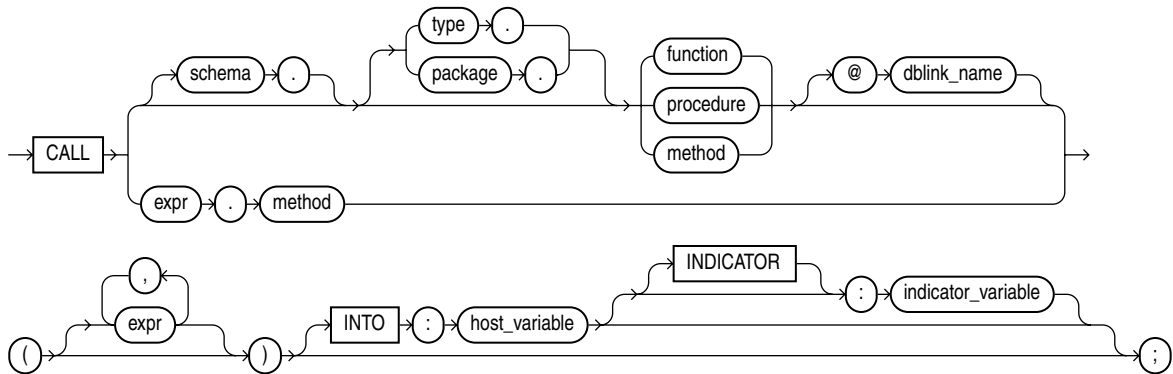
参照： これらのルーチンの作成の詳細は、『PL/SQL ユーザーズ・ガイド およびリファレンス』を参照してください。

前提条件

スタンドアロン・ルーチン、あるいはルーチンが定義されている型またはパッケージに対する EXECUTE 権限が必要です。

構文

call::=



キーワードとパラメータ

schema

スタンドアロン・ルーチン（あるいは、ルーチンが含まれている型またはパッケージ）が存在するスキーマを指定します。*schema* を指定しない場合、ルーチンが自分のスキーマ内にあるとみなされます。

type* または *package

ルーチンが定義されている型またはパッケージを指定します。

function* | *procedure* | *method

コールするファンクション名またはプロシージャ名、あるいはファンクションまたはプロシージャに変換されるシノニムを指定します。

型のメンバーであるファンクションまたはプロシージャをコールする場合、最初の引数（SELF）が NULL の IN OUT の場合、エラーが戻されます。SELF が NULL の IN の場合、NULL が戻されます。どちらの場合も、ファンクションまたはプロシージャはコールされません。

制限事項：ルーチンがファンクションの場合、INTO 句は必須です。

@dblink

分散データベース・システムで、スタンドアロン・ルーチンが含まれているデータベース（あるいは、ルーチンが含まれているパッケージまたはファンクション）の名前を指定します。*dblink* を指定しなかった場合、ローカル・データベースを指定したとみなされます。

expr

ルーチンに引数が必要な場合に、ルーチンの 1 つ以上の引数を指定します。

制限事項：

- *expr* には、疑似列、オブジェクト参照ファンクション VALUE または相関変数 REF は指定できません。
- ルーチンの IN OUT 引数または OUT 引数であるすべての *expr* は、ホスト変数の式に対応する必要があります。

INTO :*host_variable*

INTO 句は、ファンクションのコールにのみ適用されます。ファンクションの戻り値を格納するホスト変数を指定します。

:indicator_variable

ホスト変数の値または状態を指定します。

参照： ホスト変数およびインジケータ変数の詳細は、『Pro*C/C++ Precompiler プログラマーズ・ガイド』を参照してください。

例

プロシージャのコール例 次の文は、プロシージャ updateSalary を作成してからそのプロシージャをコールし、指定された従業員 ID を新しい給与で更新します。

```
CREATE OR REPLACE PROCEDURE updateSalary
(id NUMBER, newsalary NUMBER) IS
BEGIN
    UPDATE employees
    SET salary=newsalary
    WHERE employee_id=id;
END;

CALL updateSalary(1404, 50000);
```

COMMENT

用途

COMMENT を使用すると、表、ビュー、マテリアライズド・ビューまたは列に関するコメントを、データ・ディクショナリに追加できます。

データ・ディクショナリ・ビューの USER_TAB_COMMENTS、DBA_TAB_COMMENTS、ALL_TAB_COMMENTS、USER_COL_COMMENTS、DBA_COL_COMMENTS または ALL_COL_COMMENTS を問い合わせることによって、特定の表または列に関するコメントを表示できます。

データベースからコメントを削除する場合、空の文字列 '' を設定します。

参照：

- 11-66 ページの「[COMMENT](#)」を参照してください。
- データ・ディクショナリ・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

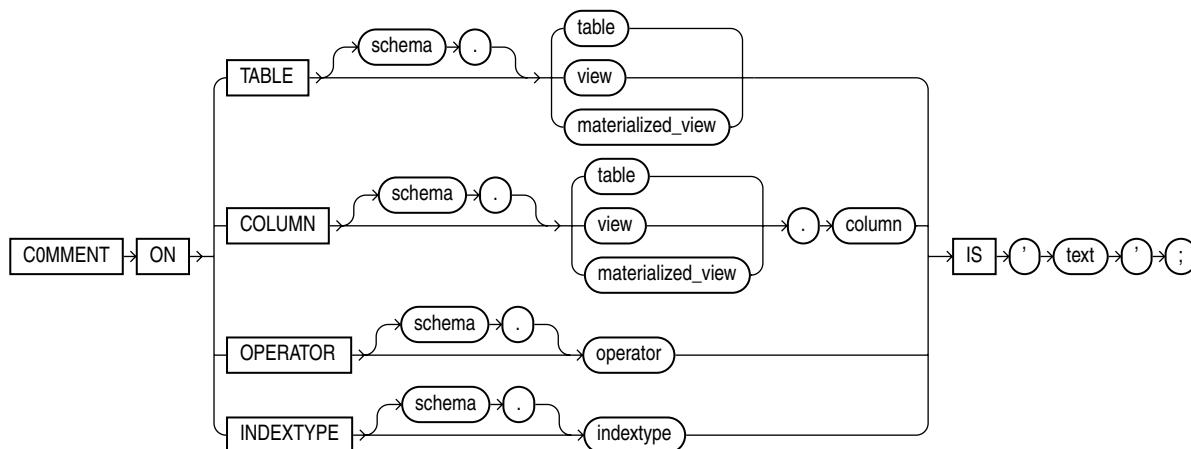
前提条件

コメントを追加するオブジェクトは自分のスキーマ内にあるか、次の条件を満たす必要があります。

- 表、ビューまたはマテリアライズド・ビューにコメントを追加する場合は、COMMENT ANY TABLE システム権限が必要です。
- 索引タイプにコメントを追加する場合は、COMMENT ANY INDEXTYPE システム権限が必要です。
- 演算子にコメントを追加する場合は、COMMENT ANY OPERATOR システム権限が必要です。

構文

comment::=



キーワードとパラメータ

TABLE 句

コメントする表、ビューまたはマテリアライズド・ビューの名前とスキーマです。*schema* を指定しない場合、この表、ビューおよびマテリアライズド・ビューは自分のスキーマ内にあるとみなされます。

COLUMN 句

コメントする表、ビューまたはマテリアライズド・ビューの列の名前を指定します。*schema* を指定しない場合、この表、ビューおよびマテリアライズド・ビューは自分のスキーマ内にあるとみなされます。

IS 'text'

コメントのテキストを指定します。

参照： 'text' の構文の詳細は、2-52 ページの「テキスト・リテラル」を参照してください。

OPERATOR 句

コメントする演算子の名前を指定します。*schema* を指定しない場合、その演算子が自分のスキーマにあるとみなされます。

INDEXTYPE 句

コメントする索引タイプの名前を指定します。*schema* を指定しない場合、索引タイプが自分のスキーマ内に定義されているものとみなされます。

例

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

```
COMMENT ON COLUMN employees.job_id  
    IS 'abbreviated job title';
```

次の文は、データベースからこのコメントを削除します。

```
COMMENT ON COLUMN employees.job_id IS ' ';
```

COMMIT

用途

COMMIT 文を使用すると、現行のトランザクションが終了し、トランザクションで実行したすべての変更が確定されます。トランザクションとは、Oracle が 1 つの単位として扱う一連の SQL 文です。また、この文によって、トランザクション内のセーブポイントがすべて消去され、トランザクションのロックが解除されます。

注意： Oracle では、データ定義言語（DDL）文の前後で暗黙的に COMMIT が発行されます。

この文を使用して、次のことができます。

- インダウト分散トランザクションを手動でコミットします。
- SET TRANSACTION 文で始まる読取り専用トランザクションを終了します。

Oracle との接続を切断する前に、最新のトランザクションを含む、アプリケーション・プログラムのすべてのトランザクションを、COMMIT 文または ROLLBACK 文を使用して明示的に終了してください。トランザクションを明示的にコミットしなかった場合にプログラムが異常終了すると、コミットされていない最後のトランザクションは、自動的にロールバックされます。

Oracle ユーティリティおよび Oracle のツール製品が正常に終了すると、現行のトランザクションがコミットされます。Oracle プリコンパイラ・プログラムが正常に終了した場合は、トランザクションはコミットされず、現行のトランザクションは Oracle によってロールバックされます。

参照：

- トランザクションの詳細は、『Oracle9i データベース概要』を参照してください。
- トランザクションの特性の指定については、17-46 ページの「[SET TRANSACTION](#)」を参照してください。

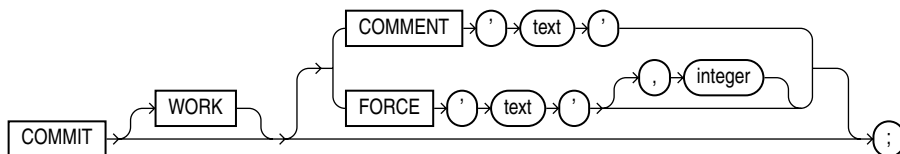
前提条件

現行のトランザクションをコミットするために、必要な権限は特にありません。

自分がコミットしたインダウト分散トランザクションを手動でコミットする場合は、FORCE TRANSACTION システム権限が必要です。別のユーザーがコミットしたインダウト分散トランザクションを手動でコミットする場合は、FORCE ANY TRANSACTION システム権限が必要です。

構文

commit::=



キーワードとパラメータ

WORK

標準 SQL に準拠するために、WORK キーワードがサポートされています。COMMIT 文と COMMIT WORK 文は同じです。

COMMENT 句

現行のトランザクションに関するコメントを指定します。'text' は引用符で囲まれた最大 255 文字のリテラルで、トランザクションの状態が不明（インダウト）になった場合に、そのトランザクション ID とともに、データ・ディクショナリ・ビュー DBA_2PC_PENDING に格納されます。

参照： SQL 文へのコメントの追加については、11-66 ページの「[COMMENT](#)」を参照してください。

FORCE 句

分散データベース・システムでは、FORCE 句によって手動でインダウト分散トランザクションをコミットできます。このトランザクションは、ローカル・トランザクション ID またはグローバル・トランザクション ID を含む 'text' で識別されます。このトランザクションの ID を確認する場合、データ・ディクショナリ・ビュー DBA_2PC_PENDING を問い合わせます。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 SCN を具体的に割り当てることもできます。integer を指定しない場合、このトランザクションは現行の SCN を使用してコミットされます。

注意： FORCE 句を指定して COMMIT 文を発行した場合、指定したトランザクションのみがコミットされます。この文は、現行のトランザクションには影響しません。

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

例

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

```
INSERT INTO regions VALUES (5, 'Antarctica');  
COMMIT WORK;
```

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

```
COMMIT  
  COMMENT 'In-doubt transaction Code 36, Call (415) 555-2637';
```

ネットワーク障害またはマシン障害によって分散トランザクションを適切にコミットできない場合、トランザクション ID とともにデータ・ディクショナリにコメントが格納されます。そのコメントには、障害が発生したアプリケーション部分が示されており、トランザクションがコミットされたデータベースの管理者に連絡する情報が提供されています。

強制インダウト・トランザクションの例 次の文は、インダウト分散トランザクションを手動でコミットします。

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

constraint_clause

用途

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

注意： オブジェクト、ネストした表、VARRAY、REF または LOB 型の列または属性に対する制約はサポートされていません。唯一の例外として、オブジェクト、VARRAY、REF または LOB 型の列または属性に対して NOT NULL 制約がサポートされています。

前提条件

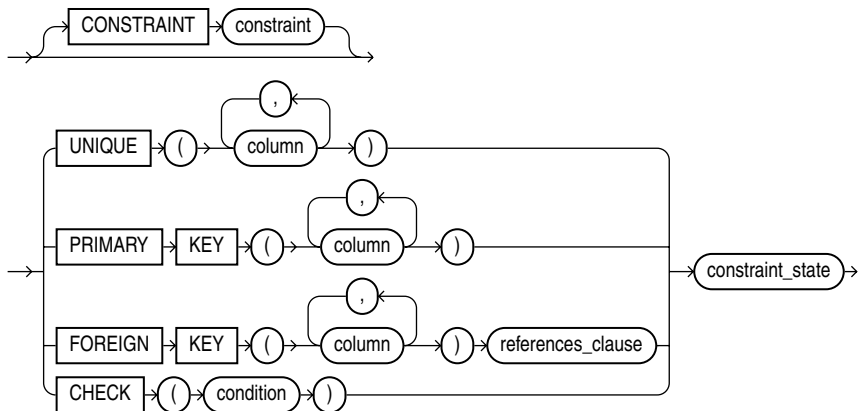
制約句は、CREATE TABLE、ALTER TABLE、CREATE VIEW または ALTER VIEW 文のいずれかで指定できます。整合性制約を定義する場合、これらの文のいずれかを発行する権限が必要です。

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

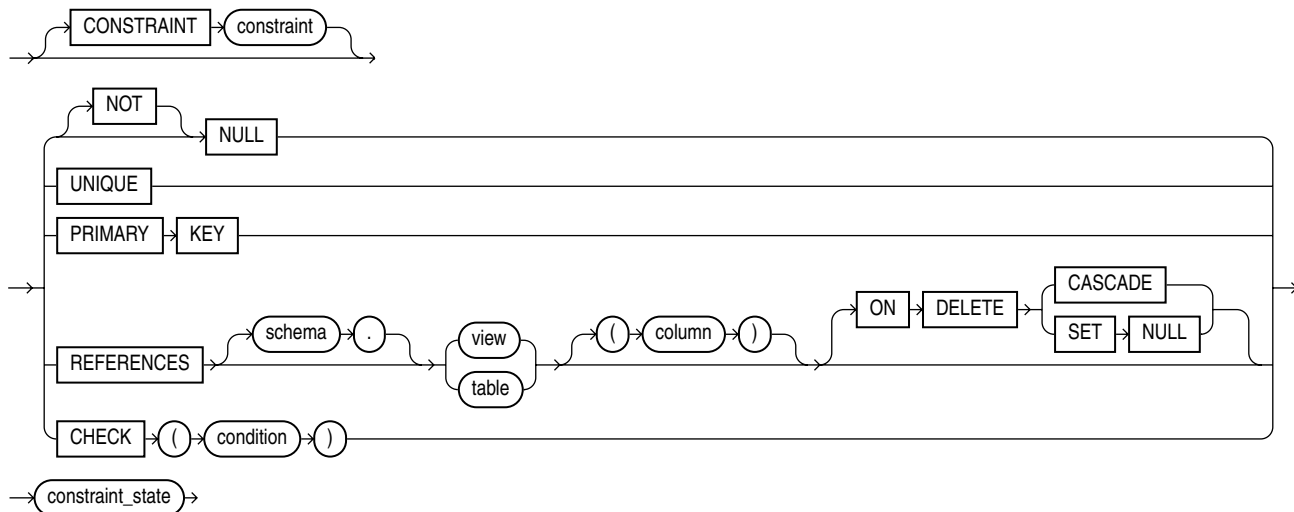
参照： 14-6 ページの「[CREATE TABLE](#)」および 10-2 ページの「[ALTER TABLE](#)」を参照してください。

構文

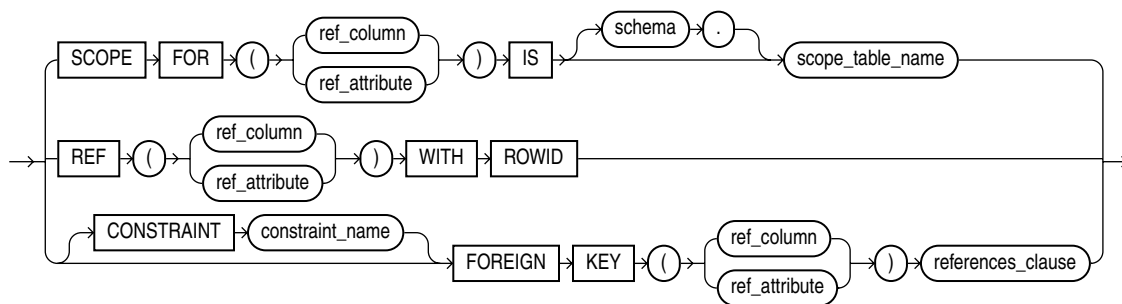
table_or_view_constraint::=



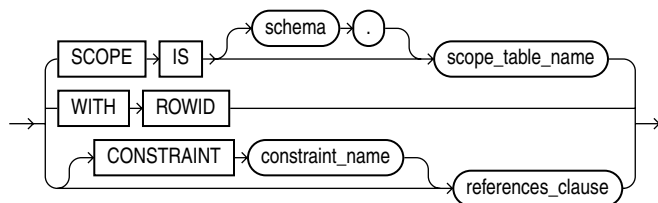
column_constraint::=



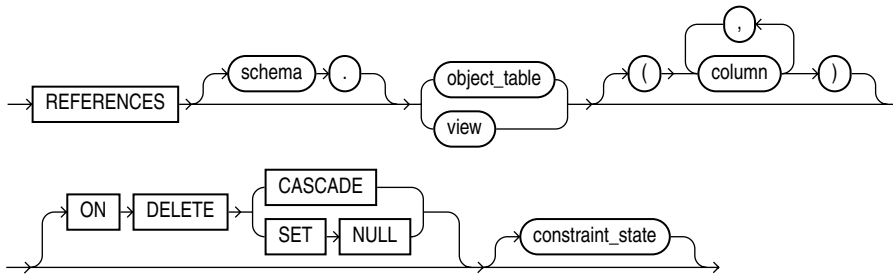
table_ref_constraint::=



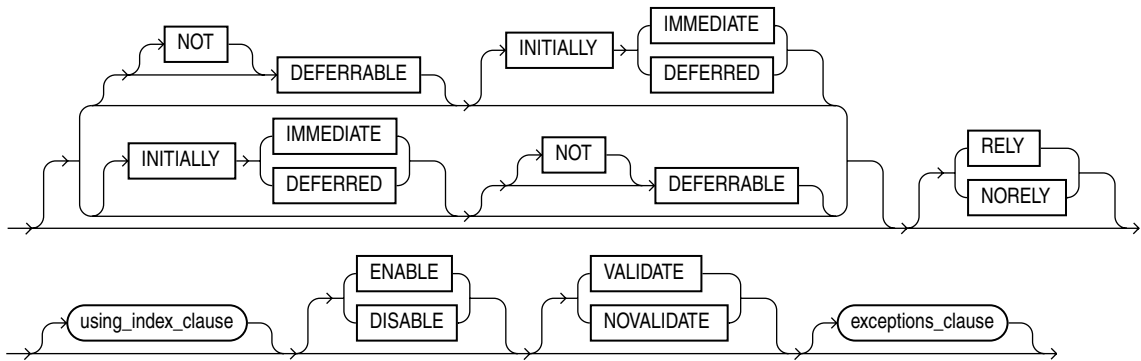
column_ref_constraint::=



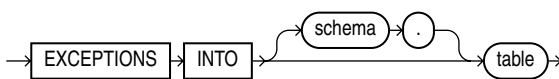
references_clause::=



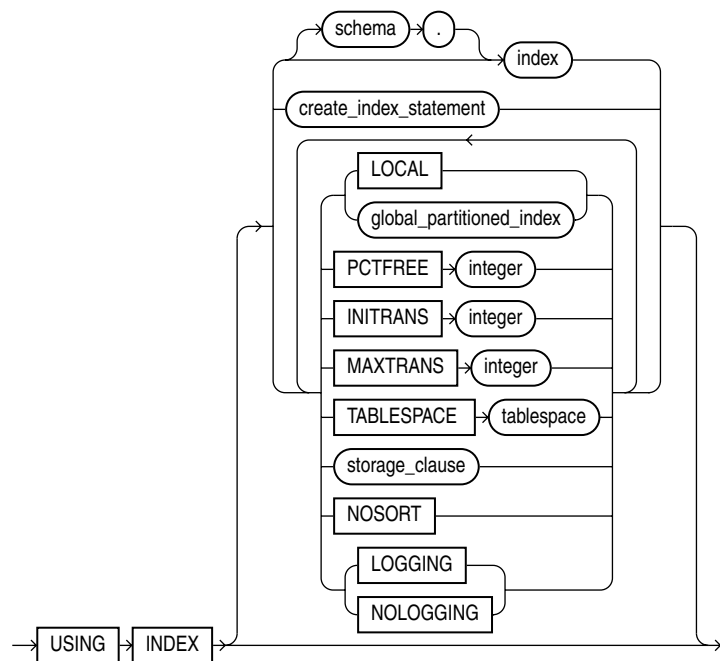
constraint_state::=



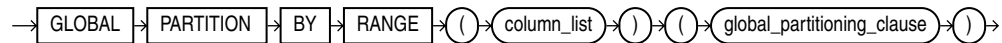
exceptions_clause::=



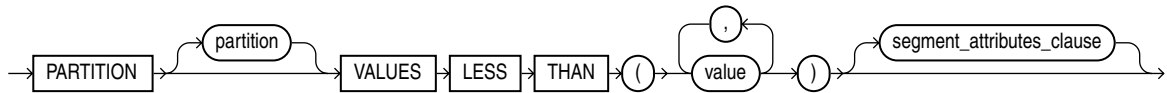
using_index_clause::=



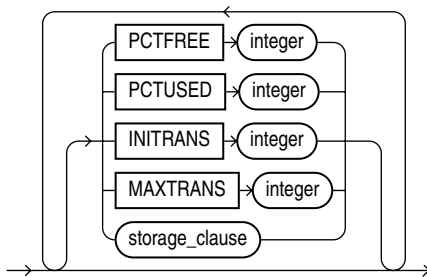
global_partitioned_index::=



global_partitioning_clause::=



physical_attributes_clause::=



storage_clause: 17-50 ページの「[storage_clause](#)」を参照してください。

キーワードとパラメータ

table_or_view_constraint

table_or_view_constraint 構文は、表またはビュー定義の一部です。この構文で定義された整合性制約は、表またはビューのすべての列に規則を適用します。

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

ビュー制約は表制約のサブセットです。ビュー制約構文は CREATE VIEW 文および ALTER VIEW 文で指定できます。ビュー制約は宣言のみです。つまり、適用されません。ただし、ビューの操作は、基礎となる実表に定義した整合性制約に従属します。そのため、実表の制約を使用してビューの制約を適用することができます。

表制約およびビュー制約の制限事項

表制約に対する制限事項

- ユーザー定義型の属性または列に対して制約を定義できません。

ビュー制約に対する追加の制限事項

- ビューには、一意制約、主キー制約および外部キー制約のみ指定できます。
- ビュー制約は直接適用されないため、DEFERRED 句または DEFERRABLE 句は指定できません。
- ビュー制約は、DISABLE NOVALIDATE モードのみでサポートされています。他のモードでは指定できません。
- ビュー制約では、*using_index_clause*、EXCEPTIONS INTO 句または ON DELETE 句を指定できません。
- オブジェクト列の属性またはオブジェクト属性の属性にビュー制約を定義できません。

column_constraint

column_constraint 構文は列定義の一部です。**表**の場合は、この構文で定義された整合性制約は、通常、定義された列に対してのみ規則を適用します。**ビュー**の場合は、制約は宣言のみです。つまり、適用されません。ただし、ビューの操作は、基礎となる実表に定義した整合性制約に従属します。そのため、実表の制約を使用してビューの制約を適用することができます。

- CREATE TABLE 文または ALTER TABLE ADD 文に指定する *column_constraint* 構文は、すべての整合性制約を定義できます。
- ALTER TABLE MODIFY *column_options* 文に指定する *column_constraint* 構文は、NOT NULL 制約の定義または削除のみできます。
- CREATE VIEW または ALTER VIEW 文に指定する *column_constraint* 構文には、ビュー制約と同じ制限事項があります。11-78 ページの「[表制約およびビュー制約の制限事項](#)」を参照してください。

制限事項：

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

CONSTRAINT

制約名を指定します。Oracle は、整合性制約の定義とこの制約名をデータ・ディクショナリに格納します。この識別子を指定しない場合、SYS_Cn の形式で名前が生成されます。

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

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

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

UNIQUE 句

UNIQUE を指定して、列または列の組合せを一意キーとして指定します。一意制約を満たす場合、表の中の 2 つの行が一意キーに対して同じ値を持つことはできません。ただし、単一の列で構成される一意キーの場合は、複数の NULL を持つことができます。

複合一意キーは、列の組合せで構成される一意キーです。複合一意キーを定義する場合、*column_constraint* 構文ではなく *table_or_view_constraint* 構文を使用してください。すべてのキー列に対して NULL を持つ行は、自動的にその制約を満たすことになります。ただし、1 つ以上のキー列に対して NULL を持ち、その他のキー列に対して同じ組合せの値を持つ 2 つの行は、制約に違反します。

注意： 1 つ以上の列に UNIQUE を指定すると、暗黙的に一意キーに索引が作成されます。問合せのパフォーマンスを目的に一意性を定義する場合は、かわりに CREATE UNIQUE INDEX 文を使用して明示的に一意索引を作成することをお勧めします。詳細は、12-58 ページの「[CREATE INDEX](#)」を参照してください。

制限事項：

- 一意キーの列は、データ型 TIMESTAMP WITH TIME ZONE を含むことはできません。ただし、一意キーは TIMESTAMP WITH LOCAL TIME ZONE の列を含むことができます。
- 複合一意キーの場合、表またはビューの 2 つの行が複合キー列に対して同じ組合せの値を持つことはできません。
- 1 つの複合一意キーは、33 以上の列を持つことはできません。キーの全体サイズ（バイト単位）は、すべての索引列の幅と索引列の数を足した値を超えてはいけません。
- 一意キーの列のデータ型は、LONG または LONG RAW であってははいけません。
- 同一の列または列の組合せを一意キーと主キーの両方には指定できません。

PRIMARY KEY 句

PRIMARY KEY を指定して、列または列の組合せを表またはビューの主キーとして指定します。**複合主キー**は、列の組合せで構成される主キーです。複合主キーを定義する場合、`column_constraint` 構文ではなく、`table_or_view_constraint` 構文を使用してください。

制限事項：

- 表またはビューには、主キーを 1 つのみ指定できます。
- 主キー列は、LONG、LONG RAW、VARRAY、ネストした表、オブジェクト、LOB、BFILE、REF または TIMESTAMP WITH TIME ZONE データ型を含むことができません。
- 主キーの値が、表の中の複数行に存在することはできません。
- 主キーを構成する列に、NULL を持たせることはできません。
- 索引構成表の 主キーのサイズは、データベース・ブロック・サイズの半分または 3800 バイトのいずれか小さい方のサイズを超えてはいけません（索引構成表には、主キーが必要です）。
- 1 つの複合主キーは、33 以上の列を持つことはできません。キーの全体サイズ（バイト単位）は、すべての索引列の幅と索引列の数を足した値を超えてはいけません。
- 同一の列または列の組合せを一意キーと主キーの両方には指定できません。
- サブビューの作成時に主キーは指定できません。主キーは、最上位（ルート）のビューのみに指定できます。

NULL | NOT NULL

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

NULL 列に NULL 値を入れることができる場合は、NULL を指定します。NULL キーワードは、実際には、整合性制約を定義しません。NOT NULL または NULL のどちらも指定しない場合、デフォルトでは列に NULL を入れることができます。

NOT NULL 列に NULL 値を入れることができない場合は、NOT NULL を指定します。この制約を満たす場合、表の中のすべての行がその列に対して値を持つ必要があります。

制限事項：

- ビューには NULL または NOT NULL を指定できません。
- オブジェクト属性には NULL または NOT NULL を指定できません。そのかわりに、IS [NOT] NULL 条件でチェック制約を使用してください。

参照： 11-94 ページの「[属性レベル制約の例](#)」を参照してください。

参照整合性制約

参照整合性制約は、外部キーとして列または列の組合せを指定し、その外部キーと、**参照キー**といわれる指定した主キーまたは一意キーの間の関連を設定します。外部キーを持つ表またはビューを**子**オブジェクトといい、参照キーを持つ表を**親**オブジェクトといいます。外部キーと参照キーを、同一の表またはビューに設定することができます。この場合、親表と子表は同一の表になります。

- 表レベルまたはビュー・レベルからは、`table_or_view_constraint` 構文の `FOREIGN KEY` 句を使用して、参照整合性を指定します。この構文で、列の組合せで構成される**複合外部キー**を指定できます。
- 列レベルからは `column_constraint` の `REFERENCES` 句を使用して、1つの列で構成される外部キーの参照整合性制約を指定します。

外部キーと主キーの両方、または外部キーと一意キーの両方に、同一の列または列の組合せを指定できます。また、外部キーとクラスタ・キーの両方にも同じ列または列の組合せを指定できます。

1つの表またはビューで複数の外部キーを定義できます。また、1つの列が複数の外部キーを構成することもできます。

参照整合性制約の制限事項

- 外部キーは、LONG 型または LONG RAW 型にはできません。
- 親表または親ビュー上で参照される一意制約または主キー制約は、あらかじめ定義しておく必要があります。
- 子表と親表は、同一データベース上に設定されている必要があります。分散データベースのノード間の参照整合性制約を使用可能にする場合、データベース・トリガーを使用する必要があります。
- 子オブジェクトと親オブジェクトのどちらかがビューの場合、ビュー制約のすべての制限事項が制約に適用されます。11-78 ページの「[表制約およびビュー制約の制限事項](#)」を参照してください。
- `AS subquery` 句を含む `CREATE TABLE` 文には、参照整合性制約を定義できません。そのかわりに、制約を指定せずに表を作成し、後で `ALTER TABLE` 文を使用してその制約を追加できます。

参照：『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

FOREIGN KEY 句

FOREIGN KEY を指定して、表レベルから列または列の組合せを外部キーとして指定します。この構文を使用して、複合外部キーを定義する必要があります。

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

REFERENCES 句

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

ON DELETE 句

表制約の場合、ON DELETE 句を指定して参照する主キーまたは一意キーの値を削除すると、参照整合性をメンテナンスする方法が自動的に判断されます。この句を指定しない場合、子表に依存する行を持つ親表の中の参照キーの値は削除できません。この句は、ビュー制約には無効です。

- 依存する外部キーの値を削除する場合は、CASCADE を指定します。
- 依存する外部キーの値を NULL に変換する場合は、SET NULL を指定します。

外部キーの制限事項

- 1 つの複合外部キーは、33 以上の列を持つことはできません。キーの全体サイズ（バイト単位）は、すべての索引列の幅と索引列の数を足した値を超えてはいけません。
- 複合外部キーは、複合一意キーまたは複合主キーを参照する必要があります。

チェック制約

CHECK 句によって、表にある各行に必要な条件を指定できます。制約を満たすためには、表のそれぞれの行が、その条件に対して TRUE または不明（NULL のため）のいずれかである必要があります。特定の行に対するチェック制約条件が評価される場合、条件にある列名によって、その行の列値が参照されます。

1 つの列に対して複数のチェック制約を作成する場合は、制約の用途が矛盾しないように注意して設計する必要があります。また、条件の評価について特別な順序を想定しないでください。Oracle は、チェック条件が相互に排他的かどうかについては検証しません。

参照： その他の詳細および構文については、[第 5 章「条件」](#) を参照してください。

チェック制約の制限事項

- ビューに対しては、チェック制約を指定できません。
- チェック制約の条件では、その表の中のすべての列を参照できますが、他の表の列は参照できません。
- チェック制約条件は、次の構造を持つことができません。
 - 副問合せ
 - ファンクション SYSDATE、UID、USER または USERENV へのコール
 - 疑似列 CURRVAL、NEXTVAL、LEVEL または ROWNUM
 - 完全に指定されていない日付定数

table_ref_constraint および *column_ref_constraint*

table_ref_constraint および *column_ref_constraint* を使用すると、REF 型の列を詳細に指定できます。これらの句の違いは、*table_ref_constraint* は表レベルで指定し、定義する REF 列または属性を識別する必要があります。

column_ref_constraint は、REF 型の列または属性を識別した後に指定します。どちらの制約でも、有効範囲制約、WITH ROWID 制約または参照整合性制約を指定できます。

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

REF 列の有効範囲表または参照表が、主キーに基づくオブジェクト識別子を持っている場合、その REF 列は**ユーザー定義 REF 列**です。

参照： REF の詳細は、『Oracle9i データベース概要』および 11-81 ページの「[参照整合性制約](#)」を参照してください。

ref_column

オブジェクトまたはリレーショナル表の REF 列の名前を指定します。

ref_attribute

リレーショナル表のオブジェクト列内の埋込み REF 属性を指定します。

SCOPE 句

表が REF 列を持つ場合は、この列のそれぞれの REF 値が別のオブジェクト表内の行を参照する場合があります。SCOPE 句は、参照の有効範囲を 1 つの表 *scope_table_name* に制限します。REF 列または属性の値は *scope_table_name* 内のオブジェクトを指し、その場所に (REF 列と同じ型の) オブジェクト・インスタンスが格納されます。REF 列ごとに有効範囲表を 1 つのみ指定できます。

SCOPE 句の制限事項

- 表が空でなければ、有効範囲制約を既存の列に追加できます。
- VARRAY 列の REF 要素に対して SCOPE は指定できません。
- AS *subquery* を指定し、この副問合せがユーザー定義 REF を戻す場合、この句を指定する必要があります。
- *scope_table_name* が自分のスキーマ内にあるか、または *scope_table_name* に対する SELECT 権限または SELECT ANY TABLE システム権限が必要です。
- REF 列からは有効範囲表制約を削除できません。

WITH ROWID 句

WITH ROWID を指定して、*ref_column* または *ref_attribute* の REF の値とともに ROWID を格納します。ROWID とともに REF 値を格納した場合、参照解除操作のパフォーマンスは向上しますが、さらに多くの領域が必要になります。デフォルトの REF 値の記憶域では、ROWID は格納されません。

WITH ROWID 句の制限事項

- VARRAY 列の REF 要素に対して、WITH ROWID 制約は指定できません。
- REF 列から WITH ROWID 表制約は削除できません。
- REF 列または属性の範囲が限定される場合、この句は無視され、ROWID は REF とともに格納されません。

references_clause

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

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

***references_clause* の制限事項**

- 範囲が限定されている既存の REF 列に参照整合性制約を追加する場合、参照表は、REF 列の有効範囲表と同じである必要があります。
- 範囲が限定されていない既存の REF 列に参照整合性制約を追加する場合、システムが有効範囲制約を追加します。したがって、有効範囲制約に適用されるすべての制限はこの場合にも適用されます。
- 後で参照整合性制約を削除する場合、REF 列の範囲が参照表に限定されたままになります。

constraint_state

`constraint_state` を使用して、Oracle が制約を適用する方法およびタイミングを指定します。

DEFERRABLE | NOT DEFERRABLE

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

参照：

- 各 DML 文の後の制約チェックについては、17-41 ページの「[SET CONSTRAINT \[S\]](#)」を参照してください。
- 遅延制約の詳細は、『Oracle9i データベース管理者ガイド』および『Oracle9i データベース概要』を参照してください。

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

INITIALLY IMMEDIATE INITIALLY IMMEDIATE を指定すると、各トランザクションの開始時に制約がチェックされます。デフォルトは、各 DML 文の終わりです。INITIALLY を指定しない場合、デフォルト値は INITIALLY IMMEDIATE になります。

INITIALLY DEFERRED INITIALLY DEFERRED を指定すると、この制約が DEFERRABLE であり、デフォルトでは各トランザクションの終了時にのみ制約がチェックされます。

DEFERRABLE および NOT DEFERRABLE の制限事項

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

RELY | NORELY

RELY および NORELY パラメータは、クエリー・リライトを行うアカウントに NOVALIDATE モードの制約を適用するかどうかを指定します。RELY を指定すると、適用されていないクエリー・リライトに対する NOVALIDATE モードの既存の制約は、アクティブになります。その制約は NOVALIDATE モードであるため、適用されません。デフォルト値は NORELY です。

適用されない制約は、通常、マテリアライズド・ビューおよびクエリー・リライトにのみ有効です。QUERY_REWRITE_INTEGRITY モード (9-2 ページの「[ALTER SESSION](#)」を参照) に従って、クエリー・リライトは、VALIDATE モードの制約、または RELY パラメータが設定された NOVALIDATE モードの制約を使用して、結合情報を確認します。

参照： マテリアライズド・ビューおよびクエリー・リライトの詳細は、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

RELY および NORELY の制限事項

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

using_index_clause

using_index_clause を使用すると、Oracle が一意制約および主キー制約の適用に使用する索引の指定、または制約の適用で使用する索引を Oracle に作成するよう指示することができます。

- schema.index を指定すると、指定した索引を使用して制約が適用されます。索引がない、または制約の適用に索引が使用できない場合、エラーが戻されます。
- create_index_statement を指定すると、索引が作成され、制約の適用に使用されます。索引が作成できない、または制約の適用に索引が使用できない場合、エラーが戻されます。
- 既存の索引を指定せず、新しい索引を作成しない場合、索引が作成されます。この場合、次のようになります。
 - 索引には制約と同じ名前が割り当てられます。
 - 索引には、INITRANS、MAXTRANS、TABLESPACE、STORAGE および PCTFREE の各パラメータ値を選択できます。
 - 表がパーティション化されている場合、一意制約または主キー制約にローカル・パーティション索引またはグローバル・パーティション索引を指定できます。

参照： 制約の適用で Oracle が使用する索引の作成方法の例は、11-92 ページの「[明示的な索引の制御例](#)」を参照してください。

***using_index_clause* の制限事項**

- この句は、ビュー制約には指定できません。
- 一意制約および主キー制約が使用可能な場合のみ、*using_index_clause* を指定できます。
- 索引構成表の主キーが使用可能な場合は、索引の指定 (*schema.index*) および作成 (*create_index_statement*) はできません。

参照：

- INITTRANS、MAXTRANS、TABLESPACE、STORAGE および PCTFREE パラメータの詳細は、14-23 ページの「CREATE TABLE」の「[segment_attributes_clause](#)」を参照してください。
- LOCAL と *global_partitioned_index* 句、および索引に関連する NOSORT と LOGGING|NOLOGGING の詳細は、12-58 ページの「[CREATE INDEX](#)」を参照してください。

global_partitioned_index

global_partitioned_index を使用すると、索引のパーティション化がユーザー定義であり、基礎となる表と同一レベルでパーティション化されないことを指定できます。デフォルトでは、非パーティション索引はグローバル索引です。

PARTITION BY RANGE PARTITION BY RANGE によって、*column_list* で指定された列の値の範囲でグローバル索引がパーティション化されるように指定します。ローカル索引にこの句は指定できません。

column_list *column_list* には、索引のパーティション化を行う表の列名を指定します。*column_list* では、索引の列リストの左の接頭辞を指定する必要があります。

column_list には、最大 32 列まで指定できます。なお、ROWID 疑似列および ROWID 型の列は指定できません。

注意： 別のキャラクタ・セットを使用してデータベースを使用しているか、使用する予定がある場合は、キャラクタ列をパーティション化する際に注意してください。文字のソート順序は、すべてのキャラクタ・セットで同一ではありません。

参照： キャラクタ・セットのサポートの詳細は、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

PARTITION PARTITION 句によって、個々の索引パーティションを記述できます。句の数によってパーティションの数が決まります。partition を指定しない場合、名前は SYS_Pn の形式で生成されます。

VALUES LESS THAN VALUES LESS THAN (value_list) には、グローバル索引の現行のパーティションの上限（境界は含まない）を指定します。value_list には、partition_by_range_clause の column_list と対応するリテラル値を順番にカンマで区切って指定します。最後のパーティションの value_list は、必ず MAXVALUE を指定してください。

注意： index が DATE 列でパーティション化されている場合、および日付書式で年の最初の 2 桁の数字が指定されていない場合、年の 4 文字書式マスクで TO_DATE ファンクションを使用する必要があります。日付書式は、NLS_TERRITORY によって暗黙的に決定され、NLS_DATE_FORMAT によって明示的に決定されます。

NOSORT 句

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

ENABLE 句

制約を表のすべての新しいデータに適用させる場合は、ENABLE を指定します。

- **ENABLE VALIDATE** を使用すると、すべての旧データも制約に従うことを指定できます。制約が使用可能で、妥当性チェック済の場合は、すべてのデータが現在有効で、今後でも有効であることが保証されます。

整合性制約に違反する行が表にある場合、制約は使用禁止のままエラーを戻します。すべての行が制約に従っている場合、Oracle は制約を使用可能にします。新規データが整合性制約に違反する場合、その後の文は実行されず、整合性制約の違反を示すエラーが戻されます。

主キー制約を **ENABLE VALIDATE** モードに設定すると、妥当性チェック・プロセスによって主キー列に NULL が含まれないことが検証されます。これによるオーバーヘッドを回避するには、この表の主キー制約を使用可能にする前に、主キーの各列に NOT NULL マークを付けます（最適な結果を得るためには、列にデータを入力してから行ってください）。
- **ENABLE NOVALIDATE** を使用すると、制約データに対して新しく行うすべての DML 操作が制約に従うことが保証されます。この句は、表の既存データが制約に従っていることを保証しないため、表レベルのロックは必要ありません。

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

一意制約または主キー制約を使用可能にした場合で、キーに索引が存在しない場合、Oracle は一意の索引を作成します。その後で制約が使用禁止になった場合、この索引は削除されます。そのため、制約が使用可能になるたびに索引が再作成されます。

索引の再作成を避け、余分な索引を削除するために、最初に使用禁止にした主キー制約および一意制約を新しく作成します。その後、一意でない索引を作成して（または、既存の一意でない索引を使用して）制約を適用してください。制約が使用禁止の場合、一意でない索引は削除されず、後続の ENABLE 操作が容易になります。

ENABLE NOVALIDATE から ENABLE VALIDATE までの単一制約状態を変更すると、パラレルで操作が実行できるため、読取り、書き込みまたはその他の DDL 操作が中断されません。

ENABLE の制限事項 使用禁止になっている一意キーまたは主キーを参照する外部キーを、使用可能にすることはできません。

DISABLE 句

整合性制約を使用禁止にする場合は、DISABLE を指定します。データ・ディクショナリでは、使用禁止になっている整合性制約は、使用可能な制約とともに表示されます。この句を指定せずに制約を作成した場合は、その制約は自動的に使用可能になります。

- DISABLE VALIDATE は、制約を使用禁止にし制約の索引を削除しますが、制約は有効のままです。この機能は大変有効です。データ・ウェアハウスでは、一意キーに重複しない範囲の値を持つ一定量のデータをレンジ・パーティション表にロードする必要があります。このような場合、制約が使用禁止で、妥当性チェック済の場合は、索引を持たないことによって領域を節約できます。ALTER TABLE 文の *exchange_partition_clause* を使用して、データを非パーティション表からパーティション表にロードすることもできます。他の SQL 文を使用した表に対するすべての変更（挿入、更新および削除）は禁止されます。

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

- DISABLE NOVALIDATE は、Oracle によって制約がメンテナンスされないこと（使用禁止になっているため）、および制約が真であると保証されないこと（妥当性チェックが行われていないため）を示します。

外部キー制約が DISABLE NOVALIDATE 状態であっても、外部キーが参照している主キーを持つ表を削除できません。また、オブティマイザは、DISABLE NOVALIDATE 状態でも制約を使用できます。

参照： この設定の使用については、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。

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

一意索引を使用している一意制約または主キー制約を使用禁止にすると、一意索引は削除されます。

exceptions_clause

EXCEPTIONS INTO 句によって、制約の整合性が違反するすべての行の ROWID を格納した表を指定できます。schema を指定しない場合、自分のスキーマ内に例外表があるとみなされます。この句自体を指定しない場合、表の名前は EXCEPTIONS になります。例外表は、ローカル・データベース内にある必要があります。

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

次のいずれかのスクリプトを使用して、EXCEPTIONS 表を作成できます。

- UTLEXCPT.SQL は、物理 ROWID を使用します。そのため、行は、索引構成表からではなく従来表から収集されます（次の注意を参照）。
- UTLEXPT1.SQL は、ユニバーサル ROWID を使用します。そのため、行は、従来表と索引構成表の両方から収集されます。

独自の例外表を作成する場合、これら 2 つのスクリプトのいずれかで規定される形式に従う必要があります。

制限事項：

- この句は、ビュー制約には指定できません。
- この文が正常に終了されるまで ROWID は存在しないため、この句を CREATE TABLE 文に指定することはできません。

注意： ユニバーサル ROWID ではなく、主キーに基づく索引構成表から例外を収集する場合、索引構成表ごとに別の例外表を作成し、主キー記憶域を確保する必要があります。スクリプトを変更および再発行することによって、別の名前の例外表を複数作成できます。

参照：

- これらのスクリプトを使用する場合の互換性については、『Oracle9i データベース移行ガイド』を参照してください。
- SQL スクリプトの詳細は、『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の DBMS_IOT パッケージを参照してください。
- 移行行および連鎖行の削除については、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。

例

一意キーの例 次の文は、デモ表 `sh.promotions` を作成する文の一例です。 `promo_id` 列に一意キーを定義し、使用可能にします（この例では、他の制約は省略されています）。

```
CREATE TABLE promotions
( promo_id          NUMBER(6)
                                CONSTRAINT promo_id_u  UNIQUE
, promo_name        VARCHAR2(20)
, promo_category    VARCHAR2(15)
, promo_cost        NUMBER(10,2)
, promo_begin_date  DATE
, promo_end_date    DATE
) ;
```

制約 `promo_id_u` は、一意キーとして `promo_id` 列を識別します。この制約によって、表に同じ ID を持つ複数の販売促進の行が存在しないことが保証されます。ただし、識別子のない行は許可されます。

また、`table_or_view_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 promotions
( promo_id          NUMBER(6)
, promo_name        VARCHAR2(20)
, promo_category    VARCHAR2(15)
, promo_cost        NUMBER(10,2)
, promo_begin_date  DATE
, promo_end_date    DATE
, CONSTRAINT promo_id_u UNIQUE (promo_id)
USING INDEX PCTFREE 20
          TABLESPACE ts_1
          STORAGE (INITIAL 8K NEXT 6K) );
```

前述の文では、`USING INDEX` 句を使用して、制約を使用可能にするために作成される索引の記憶特性も指定しています。

明示的な索引の制御例 次の文は、Oracle が制約の適用で使用する索引を明示的に制御する一意制約または主キー制約の別の作成方法を示します。

```
CREATE TABLE promotions
( promo_id          NUMBER(6)
, promo_name        VARCHAR2(20)
, promo_category    VARCHAR2(15)
, promo_cost        NUMBER(10,2)
, promo_begin_date  DATE
, promo_end_date    DATE
, CONSTRAINT promo_id_u UNIQUE (promo_id, promo_cost)
      USING INDEX (CREATE UNIQUE INDEX promo_ix1
      ON promotions(promo_id, promo_cost))
, CONSTRAINT promo_id_u2 UNIQUE (promo_cost, promo_id)
      USING INDEX promo_ix1);
```

この例は、1つの制約用に索引を作成できること、およびその索引を使用して同じ文の別の制約を作成および使用可能であることを示します。

複合一意キーの例 次の文は、oe.customers 表の customer_id 列と cust_email 列を組み合わせる複合一意キーを定義し、使用可能にします。

```
ALTER TABLE customers
ADD CONSTRAINT cust_unq UNIQUE (customer_id, cust_email)
USING INDEX PCTFREE 5
EXCEPTIONS INTO bad_cust_name;
```

cust_unq 制約によって、customer_id と cust_email を組み合わせた値が表の中に複数存在しないことが保証されます。

この ADD CONSTRAINT 句には、制約以外のプロパティも指定できます。

- USING INDEX 句は、制約を使用可能にするために作成される索引の記憶特性を指定します。
- EXCEPTIONS INTO 句は、制約に違反する customer 表の行に関する情報を bad_cust_name 表に書き込みます。

主キーの例 次の文は、hr.locations 表を作成する文の一例です。locations 表を作成し、location_id 列に主キーを定義し、使用可能にします（この例では、他の制約は省略されています）。

```
CREATE TABLE locations
( location_id    NUMBER(4) CONSTRAINT loc_id_pk PRIMARY KEY
, street_address VARCHAR2(40)
, postal_code    VARCHAR2(12)
, city           VARCHAR2(30)
, state_province VARCHAR2(25)
, country_id     CHAR(2)
) ;
```

loc_id_pk 制約は、locations 表の主キーとして、location_id 列を識別します。この制約によって、表の中の複数の所在地が同一の所在地番号を持つことはなく、かつ所在地番号が NULL にならないことが保証されます。

また、table_or_view_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 locations
( location_id    NUMBER(4)
, street_address VARCHAR2(40)
, postal_code    VARCHAR2(12)
, city           VARCHAR2(30)
, state_province VARCHAR2(25)
, country_id     CHAR(2)
, CONSTRAINT loc_id_pk PRIMARY KEY (location_id));
```

複合主キーの例 次の文は、sh.sales 表の prod_id 列および cust_id 列を組み合わせで複合主キーを定義します。

```
ALTER TABLE sales
ADD PRIMARY KEY (prod_id, cust_id) DISABLE;
```

この制約は、sales 表の主キーとして prod_id 列と cust_id 列の組合せを識別します。この制約によって、表の中の複数の行が prod_id 列と cust_id 列の両方に対して同じ値を持たないことが保証されます。

この制約句（PRIMARY KEY）では、次の制約のプロパティも指定しています。

- 制約定義によって制約名が指定されていないため、この制約に対する名前が Oracle によって自動的に生成されます。
- DISABLE 句によって制約が定義されますが、使用可能にはなりません。

NOT NULL の例 次の文は、country_id 列に NOT NULL 制約を定義し、使用可能になるように locations (11-93 ページの「主キーの例」を参照) を変更します。

```
ALTER TABLE locations
  MODIFY (country_id CONSTRAINT country_nn NOT NULL);
```

制約 country_nn によって、表に country_id が NULL の所在地の行がないことが保証されます。

属性レベル制約の例 次の文では、students 表の name 列の first_name 属性と last_name 属性の両方に対して値が存在することを保証します。

```
CREATE TYPE person_name AS OBJECT
  (first_name VARCHAR2(30), last_name VARCHAR2(30));

CREATE TABLE students (name person_name, age INTEGER,
  CHECK (name.first_name IS NOT NULL AND
    name.last_name IS NOT NULL));
```

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

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

この fk_deptno 制約によって、emp 表の従業員が属しているすべての部門が dept 表にあることになります。ただし、部門番号が NULL 値の従業員（どの部門にも属さない従業員）がいてもかまいません。すべての従業員が部門に割り当てられたことを保証するには、emp 表の deptno 列に対し NOT NULL 制約を作成し、さらに REFERENCES 制約も作成します。

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

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

制約定義によって親表と参照キーの列の両方が指定されます。参照キーは親表の主キーであるため、参照キーの列名の指定は任意です。

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

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

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

これらの外部キー定義は、両方とも ON DELETE 句を指定していないため、従業員がいる部門は削除できません。

ON DELETE の例 次の文では、emp 表を作成し、2 つの参照整合性制約を定義して使用可能にした後、ON DELETE 句を使用します。

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

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

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

複合参照整合性制約の例 次の文は、phone_calls 表の areaco 列と phoneno 列を組み合わせて、外部キーを定義して使用可能にします。

```
ALTER TABLE phone_calls
  ADD CONSTRAINT fk_areaco_phoneno
    FOREIGN KEY (areaco, phoneno)
    REFERENCES customers(areaco, phoneno)
    EXCEPTIONS INTO wrong_numbers;
```

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

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

チェック制約の例 次の文は、dept 表を作成し、その表の各列にチェック制約を定義します。

```
CREATE TABLE dept
  (deptno NUMBER CONSTRAINT check_deptno
    CHECK (deptno BETWEEN 10 AND 99)
    DISABLE,
  dname VARCHAR2(9) CONSTRAINT check_dname
    CHECK (dname = UPPER(dname))
    DISABLE,
  loc VARCHAR2(10) CONSTRAINT check_loc
    CHECK (loc IN ('DALLAS', 'BOSTON',
    'NEW YORK', 'CHICAGO'))
    DISABLE);
```

列に定義されている各制約によって、列の値が次のように制限されます。

- check_deptno によって、部門番号は必ず 10 ～ 99 の範囲内になります。
- check_dname によって、部門名はすべて大文字になります。
- check_loc によって、部門の所在地は Dallas、Boston、New York または Chicago のいずれかに制限されます。

それぞれの CONSTRAINT 句に DISABLE 句が指定されているため、Oracle は、これらの制約を定義するのみで使用可能にはしません。

次の文は、emp 表を作成し、*table_or_view_constraint_clause* を使用して、チェック制約を定義し、使用可能にします。

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

この制約は、不等式の条件を使用して、従業員の給与の合計（給与と歩合の合計金額）を 5000 ドルに制限します。

- 従業員の給与と歩合が NULL 以外の値の場合、制約を満たすには、これらの値の合計金額が 5000 ドルを超えてはいけません。
- 従業員の給与または歩合が NULL 値の場合、条件の結果は不明となり、その従業員は自動的に制約を満たします。

この例の CONSTRAINT 句には制約名が指定されていないため、Oracle が制約に対して名前を生成します。

次の文は、主キー制約、2 つの参照整合性制約、NOT NULL 制約および 2 つのチェック制約を定義して、使用可能にします。

```
CREATE TABLE order_detail
(CONSTRAINT pk_od PRIMARY KEY (order_id, part_no),
order_id NUMBER
CONSTRAINT fk_oid REFERENCES scott.order (order_id),
part_no    NUMBER
CONSTRAINT fk_pno REFERENCES scott.part (part_no),
quantity   NUMBER
CONSTRAINT nn_qty NOT NULL
CONSTRAINT check_qty CHECK (quantity > 0),
cost        NUMBER
CONSTRAINT check_cost CHECK (cost > 0) );
```

この制約によって、表のデータに対して次の規則を適用できます。

- `pk_od` は、`order_id` 列と `part_no` 列の組合せを表の主キーとして指定します。この制約を満たすためには、`order_id` 列および `part_no` 列の組合せが同じである行が、表内に 2 つあってはいけません。また、`order_id` 列および `part_no` 列では、行に `NULL` を入れることはできません。
- `fk_oid` は、`scott` のスキーマ内の `order` 表にある `order_id` 列を参照する外部キーとして、`order_id` 列を指定します。`order_detail.order_id` 列に追加されるすべての新しい値は、`scott.order.order_id` 列にあらかじめ存在する必要があります。
- `fk_pno` は、`scott` のスキーマ内の `part` 表にある `part_no` 列を参照する外部キーとして、`part_no` 列を指定します。`order_detail.part_no` 列に追加されるすべての新しい値は、`scott.part.part_no` 列にあらかじめ存在する必要があります。
- `nn_qty` は、`quantity` 列に対して `NULL` を禁止します。
- `check_qty` によって、`quantity` 列の値は必ず 0 (ゼロ) より大きくなります。
- `check_cost` によって、`cost` 列の値は必ず 0 (ゼロ) より大きくなります。

この例は、`CONSTRAINT` 句と列定義について、次の点も具体的に示しています。

- `table_or_view_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_or_view_constraint` 構文が列定義より先に指定されています。
- 列定義には、`column_constraint` 構文を複数回使用できます。この例では、`quantity` 列の定義は `nn_qty` 制約と `check_qty` 制約の両方の定義を含んでいます。
- 表には、複数のチェック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 つのチェック制約より、それぞれ 1 つのビジネス・ルールのみを適用する単純な条件を持つ複数のチェック制約を使用してください。矛盾している制約があると、その制約を識別するエラー・メッセージが戻されます。エラーが検出された制約が 1 つのビジネス・ルールのみを有効にする場合、このようなエラー・メッセージの方が、矛盾のあるビジネス・ルールを正確に識別できます。

DEFERRABLE 制約の例 次の文は、`scores` 列に対して `NOT DEFERRABLE INITIALLY IMMEDIATE` のチェック制約を指定し、`games` 表を作成します。

```
CREATE TABLE games (scores NUMBER CHECK (scores >= 0));
```

次の文は、列に対する一意制約を `INITIALLY DEFERRED DEFERRABLE` として定義します。

```
CREATE TABLE orders
(ord_num NUMBER, CONSTRAINT unq_num UNIQUE (ord_num)
INITIALLY DEFERRED DEFERRABLE);
```

SQL 文 : CREATE CLUSTER ~ CREATE JAVA

この章では、次の SQL 文について説明します。

- CREATE CLUSTER
- CREATE CONTEXT
- CREATE CONTROLFILE
- CREATE DATABASE
- CREATE DATABASE LINK
- CREATE DIMENSION
- CREATE DIRECTORY
- CREATE FUNCTION
- CREATE INDEX
- CREATE INDEXTYPE
- CREATE JAVA

CREATE CLUSTER

用途

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

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

参照：

- クラスタの概要については、『Oracle9i データベース概要』を参照してください。
- クラスタのパフォーマンスについては、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。
- クラスタの使用方法については、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。
- データ・ディクショナリ・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

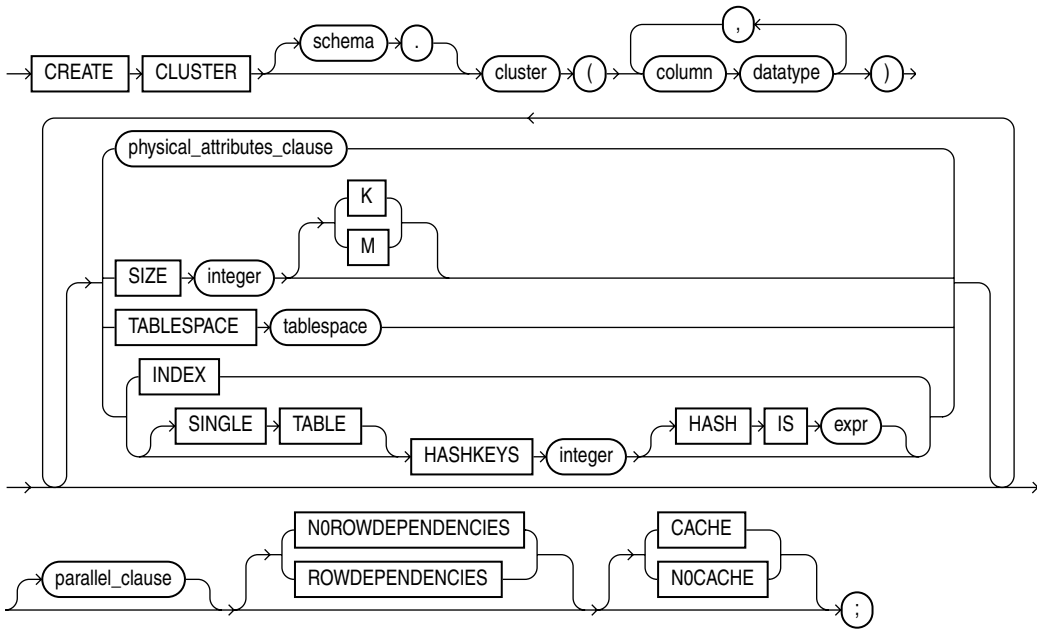
前提条件

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

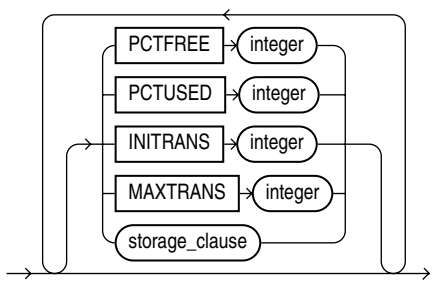
Oracle では、クラスタを最初に作成する場合は、クラスタに対する索引は自動的に作成されません。このため、クラスタ索引を作成するまでは、クラスタ化表に対して、データ操作言語（DML）文を発行できません。

構文

create_cluster::=

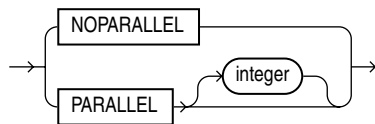


physical_attributes_clause::=



storage_clause: 17-50 ページの「[storage_clause](#)」を参照してください。

parallel_clause::=



キーワードとパラメータ

schema

作成するクラスタが定義されるスキーマを指定します。*schema* を指定しない場合、現行のスキーマにクラスタが作成されます。

cluster

作成するクラスタの名前を指定します。

クラスタを作成した後、そのクラスタに表を追加します。クラスタには、最大 32 個の表を指定できます。クラスタを作成し、そのクラスタに表を追加しても、クラスタを意識する必要はありません。クラスタ化されていない表と同じように、SQL 文を使用してクラスタ化表にアクセスできます。

参照： クラスタへの表の追加については、14-6 ページの「[CREATE TABLE](#)」を参照してください。

column

クラスタ・キーに 1 つ以上の列名を指定します。最大 16 個までクラスタ・キー列を指定できます。これらの列は、データ型およびサイズについて、クラスタ化表の列と対応している必要があります。名前は対応している必要はありません。

クラスタ・キー列の定義の一部として整合性制約は指定できません。そのかわり、クラスタに属している表に整合性制約を対応付けることができます。

datatype

各クラスタ・キー列のデータ型を指定します。

制限事項：

- データ型が LONG、LONG RAW、REF、ネストした表、VARRAY、BLOB、CLOB、BFILE またはユーザー定義オブジェクト型であるクラスタ・キー列は指定できません。
- 列のデータ型が、位取りが 0 である INTEGER 型や NUMBER 型でない場合、HASH IS 句は使用できません。
- ROWID 型の列を指定することはできますが、それらの列の値が有効な列 ID であることは保証されません。

参照： データ型の詳細は、2-2 ページの「[データ型](#)」を参照してください。

physical_attributes_clause

physical_attributes_clause によって、クラスタの記憶特性を指定できます。クラスタ内の各表もこれらの記憶特性を使用します。

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

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

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

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

参照： PCTUSED、PCTFREE、INITRANS および MAXTRANS の各パラメータの詳細は、14-6 ページの「[CREATE TABLE](#)」を参照してください。

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

参照： 17-50 ページの「[storage_clause](#)」を参照してください。

SIZE

同一クラスタ・キー値または同一ハッシュ値を持つすべての行を格納するために確保する領域を、バイト単位で指定します。K または M を使用すると、この領域を KB または MB 単位で指定できます。次に、この領域によって、データ・ブロックごとに格納されるクラスタやハッシュ値の最大値が決まります。SIZE の値がデータ・ブロック・サイズの約数でない場合、Oracle は、次に大きな約数を使用します。SIZE がデータ・ブロック・サイズより大きい場合、Oracle は、クラスタまたはハッシュ値ごとに、1 つ以上のデータ・ブロックを確保し、オペレーティング・システムのブロック・サイズを採用します。

Oracle は、クラスタ・キー値を持つ行に対して確保する必要がある領域を決定する際に、クラスタ・キーの長さを考慮します。クラスタ・キーが大きければ、それに必要なサイズも大きくなります。実際のサイズを参照するには、USER_CLUSTERS データ・ディクショナリ・ビューの KEY_SIZE 列を問い合わせます（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）。

このパラメータを指定しない場合、各クラスタ・キー値またはハッシュ値ごとにデータ・ブロックが 1 つ確保されます。

TABLESPACE

クラスタを作成する表領域を指定します。

INDEX 句

INDEX を指定して、**索引クラスタ**を作成します。索引クラスタには、同一のクラスタ・キー値が指定されている行がまとめて格納されます。それぞれのクラスタ・キー値は、そのキーを持つ表および行の数に関係なく、各データ・ブロックに 1 回のみ格納されます。

索引クラスタの作成後、クラスタ内の表に対してデータ操作言語（DML）文を発行する前に、そのクラスタ・キーに索引を付ける必要があります。この索引を**クラスタ索引**と呼びます。

注意： ハッシュ・クラスタに対してクラスタ索引は作成できないため、ハッシュ・クラスタ・キーで索引を作成する必要はありません。INDEX も HASHKEYS も指定しない場合は、デフォルトで索引クラスタが作成されます。

参照： クラスタ索引の作成方法の詳細は、12-58 ページの「[CREATE INDEX](#)」、索引クラスタの概要は、『Oracle9i データベース概要』を参照してください。

HASHKEYS 句

HASHKEYS を指定して、ハッシュ・クラスタの作成およびハッシュ・クラスタのハッシュ値の数を指定します。ハッシュ・クラスタには、同一のハッシュ・キー値を持つ行がまとめて格納されます。それぞれの行のハッシュ値は、そのクラスタのハッシュ・ファンクションが戻す値です。

Oracle は、ハッシュ値の実際の数を決めるため、HASHKEYS 値を一番近い次の素数に切り上げます。このパラメータの最小値は 2 です。INDEX 句と HASHKEYS パラメータの両方を指定しない場合、デフォルトで索引クラスタが作成されます。

ハッシュ・クラスタの作成時に、Oracle は、SIZE パラメータおよび HASHKEYS パラメータの値に基づいて、クラスタに領域を割り当てます。

参照： Oracle がクラスタに領域を割り当てる方法の詳細は、『Oracle9i データベース概要』を参照してください。

SINGLE TABLE SINGLE TABLE は、クラスタを、表を 1 つのみ持つタイプのハッシュ・クラスタに指定します。この句によって、表がクラスタの一部でない場合より行へのアクセスが高速になります。

制限事項： 同時にクラスタに存在できる表は 1 つのみです。ただし、表を削除して、同一のクラスタに別の表を作成することはできます。

HASH IS *expr* ハッシュ・クラスタに対するハッシュ・ファンクションとして使用する式を指定します。式は、次の条件を満たす必要があります。

- 正の値に評価される必要があります。
- 式全体の値が位取り 0（ゼロ）の数になる場合は、任意のデータ型の列が参照される 1 つ以上の列を持ちます。たとえば、NUM_COLUMN * length(VARCHAR2_COLUMN) です。
- ユーザー定義の PL/SQL ファンクションを参照できません。
- SYSDATE、USERENV、TO_DATE、UID、USER、LEVEL、ROWNUM、CURRENT_DATE、CURRENT_TIMESTAMP、DBTIMEZONE、EXTRACT（日時）、FROM_TZ、LOCALTIMESTAMP、NUMTODSINTERVAL、NUMTOYMINTERVAL、SESSIONTIMEZONE、SYSTIMESTAMP、TO_DSINTERVAL、TO_TIMESTAMP、TO_TIMESTAMP_TZ、TO_YMINTERVAL および TZ_OFFSET を参照できません。
- 定数に評価されることはありません。
- スカラー副問合せ式には指定できません。
- （クラスタ名ではなく）スキーマ名またはオブジェクト名で修飾された列を持つことはできません。

HASH IS 句を指定しない場合、ハッシュ・クラスタに対して内部ハッシュ・ファンクションが使用されます。

既存のクラスタの詳細は、データ・ディクショナリ表 `USER_CLUSTER_HASH_EXPRESSIONS`、`ALL_CLUSTER_HASH_EXPRESSIONS` および `DBA_CLUSTER_HASH_EXPRESSIONS` に問い合わせます。

ハッシュ列のクラスタ・キーは、任意のデータ型で構成される 1 つ以上の列を持つことができます。複合クラスタ・キー、または整数以外の列で構成されるクラスタ・キーを持つハッシュ・クラスタに対しては、内部ハッシュ・ファンクションを使用する必要があります。

参照： データ・ディクショナリ・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

parallel_clause

parallel_clause によって、クラスタ作成をパラレル化します。

注意： *parallel_clause* 構文は、以前のリリースの構文に代わるものです。以前のリリースの構文は下位互換用にサポートされていますが、動作がわずかに異なることがあります。

NOPARALLEL シリアル実行を行う場合は、`NOPARALLEL` を指定します。これはデフォルトです。

PARALLEL 並列度を選択する場合は、`PARALLEL` を指定します。並列度とは、すべての関係するインスタンスで使用可能な CPU の数に、初期化パラメータ `PARALLEL_THREADS_PER_CPU` の値を掛けたものです。

PARALLEL integer *integer* には、パラレル操作で使用されるパラレル・スレッド数である**並列度**を指定します。各パラレル・スレッドは、1、2 個のパラレル実行サーバーを使用します。通常、最適な並列度が計算されるため、*integer* に値を指定する必要はありません。

制限事項： *cluster* の表が LOB またはユーザー定義オブジェクト型の列を含む場合、*cluster* でその後に行う INSERT、UPDATE または DELETE 操作と同様、この文は通知なしに逐次実行されます。

参照： 14-43 ページの「CREATE TABLE」の「[parallel_clause](#)に関する注意事項」を参照してください。

NOROWDEPENDENCIES | ROWDEPENDENCIES

cluster が**行レベル依存の追跡**を使用するかどうかを指定できます。この機能を使用すると、クラスタを構成する表の各行は、行を変更した最後のトランザクションのコミット時刻以降を示す SCN を持つことになります。この設定は、*cluster* の作成後に変更できません。

ROWDEPENDENCIES 行レベル依存の追跡を使用可能にする場合は、ROWDEPENDENCIES を指定します。この設定は、主にレプリケーション環境でパラレル伝播を許可する場合に便利です。各行につきサイズが 6 バイト増えます。

NOROWDEPENDENCIES 行レベル依存の追跡機能を使用しない場合は、NOROWDEPENDENCIES を指定します。これはデフォルトです。

参照： レプリケーション環境での行レベル依存の追跡の使用については、『Oracle9i レプリケーション』を参照してください。

CACHE | NOCACHE

CACHE フル・テーブル・スキャンの実行時に、このクラスタに対して取り出されたブロックを、バッファ・キャッシュ内の LRU リストの最高使用頻度側に入れる場合は、CACHE を指定します。この句は、小規模な参照表で有効です。

NOCACHE フル・テーブル・スキャンの実行時に、このクラスタに対して取り出されたブロックを、バッファ・キャッシュ内の LRU リストの最低使用頻度側に入れる場合は、NOCACHE を指定します。これはデフォルトの動作です。

注意： NOCACHE は、*storage_clause* に KEEP を指定したクラスタには影響しません。

例

クラスタの作成例 次の文は、クラスタ・キー列 department、クラスタ・サイズ 512 バイトおよび記憶域パラメータ値を指定した索引クラスタ personnel を作成します。

```
CREATE CLUSTER personnel
  (department NUMBER(4))
  SIZE 512
  STORAGE (initial 100K next 50K);
```

クラスタ・キーの例 次の文は、personnel のクラスタ・キーにクラスタ索引を作成します。

```
CREATE INDEX idx_personnel ON CLUSTER personnel;
```

クラスタ索引の作成後、索引に表を追加し、その表に対して DML 操作を行うことができます。

クラスタへの表の追加例 次の文は、サンプル表 `hr.employees` から部門表を作成し、前述の例で作成した `personnel` クラスタに追加します。

```
CREATE TABLE dept_10
  CLUSTER personnel (department_id)
  AS SELECT * FROM employees WHERE department_id = 10;

CREATE TABLE dept_20
  CLUSTER personnel (department_id)
  AS SELECT * FROM employees WHERE department_id = 20;
```

ハッシュ・クラスタの例 次の文は、クラスタ・キー列 `cust_language`、最大ハッシュ・キー値 10（各サイズ 512 バイト）および記憶域パラメータ値を指定したハッシュ・クラスタ `language` を作成します。

```
CREATE CLUSTER language (cust_language VARCHAR2(3))
  SIZE 512 HASHKEYS 10
  STORAGE (INITIAL 100k next 50k);
```

この文では、`HASH IS` 句を指定していないため、Oracle は、そのクラスタに対して内部ハッシュ・ファンクションを採用します。

次の文は、`postal_code` と `country_id` 列で構成されるクラスタ・キーを持つ `address` という名前のハッシュ・クラスタを作成し、これらの列を含む SQL 式をハッシュ・ファンクションに使用します。

```
CREATE CLUSTER address
  (postal_code NUMBER, country_id CHAR(2))
  HASHKEYS 20
  HASH IS MOD(postal_code + country_id, 101);
```

単一表ハッシュ・クラスタの例 次の文は、クラスタ・キー `customer_id` および最大ハッシュ・キー値 100（各サイズ 512 バイト）を指定した単一表ハッシュ・クラスタ `cust_orders` を作成します。

```
CREATE CLUSTER cust_orders (customer_id NUMBER(6))
  SIZE 512 SINGLE TABLE HASHKEYS 100;
```

CREATE CONTEXT

用途

次の場合に CREATE CONTEXT 文を使用します。

- コンテキスト（アプリケーションを検証および保護するアプリケーション定義の属性）のネームスペースを作成します。
- ネームスペースを、コンテキストを設定する外部作成パッケージと対応付けます。

指定されたパッケージの DBMS_SESSION.set_context プロシージャを使用して、コンテキスト属性を設定または再設定できます。

参照：

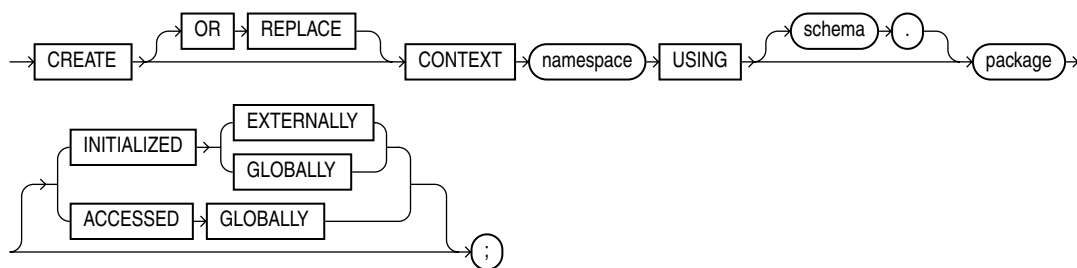
- コンテキストの定義および説明については、『Oracle9i データベース概要』を参照してください。
- DBMS_SESSION.set_context プロシージャについては、『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

前提条件

コンテキスト・ネームスペースを作成する場合、CREATE ANY CONTEXT システム権限が必要です。

構文

create_context::=



キーワードとパラメータ

OR REPLACE

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

namespace

作成または変更するためのコンテキスト・ネームスペース名を指定します。コンテキスト・ネームスペースは、スキーマ SYS に格納されます。

schema

package を所有するスキーマを指定します。*schema* を指定しない場合、Oracle は現在のスキーマを使用します。

package

ユーザー・セッションのネームスペースに基づくコンテキスト属性を設定または再設定する PL/SQL パッケージを指定します。

注意： 柔軟に設計できるように、Oracle は、コンテキスト作成時のスキーマの存在またはパッケージの妥当性を検証しません。

参照： パッケージの設定の詳細は、『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

INITIALIZED 句

INITIALIZED 句を使用すると、コンテキスト・ネームスペースを初期化する Oracle 以外のエンティティを指定することができます。

EXTERNALLY EXTERNALLY を使用すると、セッション確立時に OCI インタフェースを使用してネームスペースが初期化されます。

参照： セッション確立の OCI 使用の詳細は、『Oracle Call Interface プログラムズ・ガイド』を参照してください。

GLOBALY GLOBALY を指定すると、グローバル・ユーザーがデータベースへ接続するときに、LDAP ディレクトリによってネームスペースが初期化されます。

セッション確立後、設定した PL/SQL パッケージのみがネームスペースのすべての属性に書き込みを行うコマンドを発行することができます。

参照：

- グローバルに初期化されるコンテキストの確立の詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。
- LDAP ディレクトリを介したデータベースへの接続の詳細は、『Oracle Internet Directory 管理者ガイド』を参照してください。

ACCESSED GLOBALLY

この句を使用すると、*namespace* に設定したすべてのアプリケーション・コンテキストはすべてのインスタンスからアクセスできます。この設定によって、複数のセッションがアプリケーション属性を共有できます。

例

CREATE CONTEXT の例 この例は、13-49 ページの「[CREATE PACKAGE の例](#)」で作成した hr アプリケーションを検証および保護する PL/SQL パッケージ emp_mgmt を使用します。次の文は、コンテキスト・ネームスペース hr_context を作成し、emp_mgmt パッケージに関連付けます。

```
CREATE CONTEXT hr_context USING emp_mgmt;
```

SYS_CONTEXT ファンクションを使用するこのコンテキストに基づいて、データ・アクセスを制御できます。たとえば、emp_mgmt パッケージが、特定の従業員識別子として、属性 new_empno を定義するものとします。new_empno の値に基づき、アクセスを制限するビューを作成して employees 実表を次のように保護できます。

```
CREATE VIEW hr_org_secure_view AS
  SELECT * FROM employees
  WHERE employee_id = SYS_CONTEXT('hr_context', 'new_empno');
```

参照： 6-145 ページの「[SYS_CONTEXT](#)」を参照してください。

CREATE CONTROLFILE

注意： この文を使用する前に、データベース内のすべてのファイルの全体バックアップを実行することをお勧めします。詳細は、『Oracle9i ユーザー管理バックアップおよびリカバリ・ガイド』を参照してください。

参照： 既存のデータベース制御ファイルに基づくスクリプトの作成の詳細は、8-33 ページの「ALTER DATABASE」の「[BACKUP CONTROLFILE](#) 句」を参照してください。

用途

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

- 既存の制御ファイルのすべてのコピーが、メディア障害により消失してしまった場合。
- データベース名を変更する場合。
- REDO ログ・ファイル・グループ、REDO ログ・ファイル・メンバー、アーカイブ REDO ログ・ファイル、データ・ファイル、またはデータベースを同時にマウントおよびオープンするインスタンスの最大数を変更する場合。

CREATE CONTROLFILE 文を発行した場合、この文にユーザーが指定した情報に基づいて、新しい制御ファイルが作成されます。句を指定しない場合、Oracle は、以前の制御ファイルに対する値ではなく、デフォルト値を使用します。制御ファイルが正常に作成された後、初期化パラメータ CLUSTER_DATABASE で指定したモードでデータベースがマウントされます。データベースをオープンする前に、メディア・リカバリを行う必要があります。その後、インスタンスを停止し、データベースのすべてのファイルの全体バックアップを取ることをお勧めします。

参照： 『Oracle9i ユーザー管理バックアップおよびリカバリ・ガイド』を参照してください。

前提条件

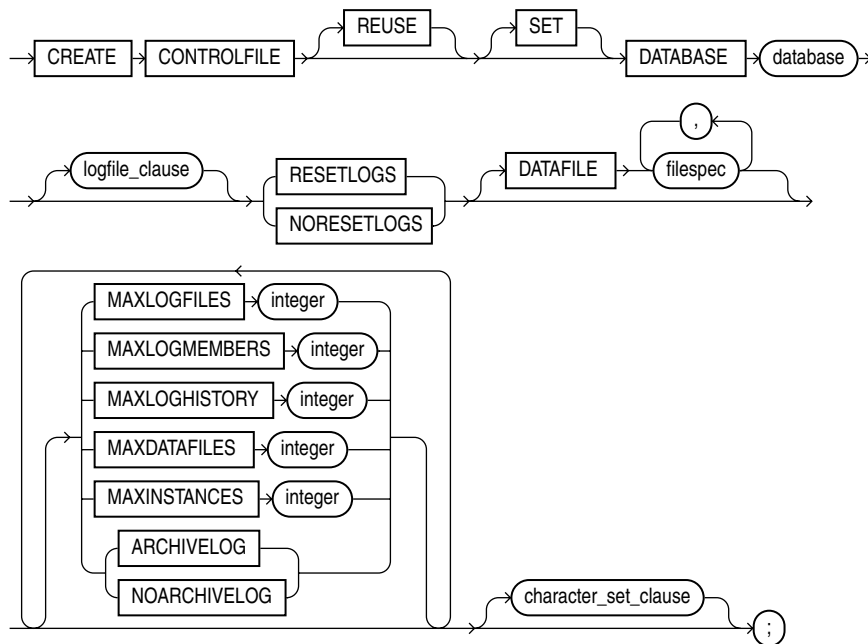
制御ファイルを作成する場合は、SYSDBA システム権限が必要です。データベースをマウントしているインスタンスがあってははいけません。

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

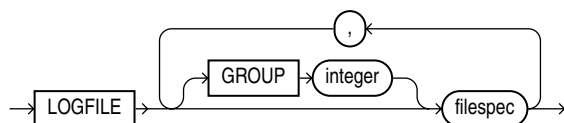
参照： REMOTE_LOGIN_PASSWORDFILE パラメータの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

構文

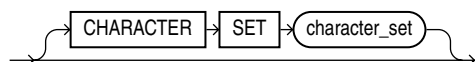
create_controlfile::=



logfile_clause::=



character_set_clause::=



filespec: 16-27 ページの「[filespec](#)」を参照してください。

キーワードとパラメータ

REUSE

初期化パラメータ `CONTROL_FILES` によって特定される既存の制御ファイルを再利用することを指定します。このとき、現在保存されている情報は無視され、上書きされます。この句を指定しないと、既存の制御ファイルがある場合に、エラーが戻ります。

DATABASE 句

データベース名を指定します。このパラメータの値は、事前に `CREATE DATABASE` 文または `CREATE CONTROLFILE` 文で設定した既存のデータベース名である必要があります。

SET DATABASE 句

`SET DATABASE` を使用して、データベース名を変更します。データベース名は最大 8 バイトまで指定可能です。

logfile_clause

logfile_clause を使用すると、データベースの REDO ログ・ファイルを指定できます。すべての REDO ログ・ファイル・グループのすべてのメンバーを指定する必要があります。

GROUP integer ログ・ファイル・グループ番号を指定します。GROUP 値を指定した場合、データベースが前回オープンされたときの GROUP 値を基にして、この値が検証されます。

この句を省略した場合、システムのデフォルト値を使用してログ・ファイルが作成されます。また、`DB_CREATE_ONLINE_LOG_DEST_n` 初期化パラメータと

`DB_CREATE_FILE_DEST` 初期化パラメータのいずれか（あるいはその両方）を設定した場合、および `RESETLOGS` を指定した場合は、`DB_CREATE_ONLINE_LOG_DEST_n` パラメータで指定したログ・ファイルのデフォルトの格納先に 2 つのログ・ファイルが作成されます。

また、`DB_CREATE_ONLINE_LOG_DEST_n` が設定されていない場合は、`DB_CREATE_FILE_DEST` パラメータで指定した格納先に作成されます。

参照： *filespec* の構文の詳細は、16-27 ページの「*filespec*」を参照してください。

RESETLOGS LOGFILE 句にリストされたファイルのコンテキストを無視する場合は、`RESETLOGS` を指定します。LOGFILE 句に指定したファイルは、存在していなくてもかまいません。LOGFILE 句の各 *filespec* で、`SIZE` パラメータを指定する必要があります。Oracle では、スレッド 1 にすべてのオンライン REDO ログ・ファイル・グループを割り当てることによって、このスレッドを任意のインスタンスで共通に使用できるようにします。この句を使用した後は、`RESETLOGS` 句を指定した `ALTER DATABASE` を使用してデータベースをオープンする必要があります。

NORESETLOGS LOGFILE 句に指定したすべてのファイルを、前回データベースをオープンしたときの状態で使用する場合は、**NORESETLOGS** を指定します。LOGFILE 句に指定したファイルは、存在する必要があります。また、バックアップからのリストアではなく、現行のオンライン REDO ログ・ファイルである必要があります。前回割り当てたスレッドに、REDO ログ・ファイル・グループが再度割り当てられ、そのスレッドが前回と同じく再度使用可能になります。

DATAFILE 句

データベースのデータ・ファイルを指定します。すべてのデータ・ファイルを指定する必要があります。これらのファイルは既存のファイルである必要がありますが、メディア・リカバリを必要とするリストア・バックアップでもかまいません。構文の詳細は、16-27 ページの「[filespec](#)」を参照してください。

注意： この句では、テンポラリ・データ・ファイル（テンポラリ・ファイル）ではなく、データ・ファイルのみを指定します。テンポラリ・ファイルの処理の詳細は、『Oracle9i ユーザー管理バックアップおよびリカバリ・ガイド』を参照してください。

MAXLOGFILES 句

データベースに対して作成可能なオンライン REDO ログ・ファイル・グループの最大数を指定します。この値を基にして、制御ファイル内で REDO ログ・ファイル名に割り当てられる領域が決定されます。デフォルト値や最大値は、使用するオペレーティング・システムによって異なります。指定する値は、すべての REDO ログ・ファイル・グループの GROUP の最大値以上である必要があります。

MAXLOGMEMBERS 句

REDO ログ・ファイル・グループのメンバー（同一コピー）の最大数を指定します。この値を基にして、制御ファイル内で REDO ログ・ファイル名に割り当てられる領域が決定されます。最小値は 1 です。最大値およびデフォルト値は、使用するオペレーティング・システムによって異なります。

MAXLOGHISTORY 句

このパラメータは、Real Application Clusters 環境で ARCHIVELOG モードの Oracle を使用している場合に便利です。Real Application Clusters の自動メディア・リカバリに使用するアーカイブ REDO ログ・ファイル・グループの最大数を指定します。この値を基にして、制御ファイル内でアーカイブ REDO ログ・ファイル名に割り当てられる領域が決定されます。最小値は 0（ゼロ）です。デフォルト値は MAXINSTANCES 値の倍数で、使用するオペレーティング・システムによって異なります。最大値は、制御ファイルの最大サイズの制限のみを受けます。

MAXDATAFILES 句

CREATE DATABASE または CREATE CONTROLFILE 実行時の、制御ファイルのデータ・ファイル・セクションの初期サイズ設定を指定します。値が MAXDATAFILES より大きく、DB_FILES 以下のファイルを追加した場合、データ・ファイル・セクションにさらに多くのファイルを格納できるように、Oracle の制御ファイルが自動的に拡張されます。

インスタンスでアクセスできるデータ・ファイルの数は、初期化パラメータ DB_FILES の制限を受けます。

MAXINSTANCES 句

データベースを同時にマウントおよびオープンできるインスタンスの最大数を指定します。この値は、初期化パラメータ INSTANCES の値より優先されます。最小値は 1 です。最大値およびデフォルト値は、使用するオペレーティング・システムによって異なります。

ARCHIVELOG | NOARCHIVELOG

ARCHIVELOG を指定して、REDO ログ・ファイルを再利用する前に、ファイルの内容をアーカイブします。この句を指定した場合、インスタンス・リカバリのみでなくメディア・リカバリもできるようになります。

ARCHIVELOG 句および NOARCHIVELOG 句を省略した場合、デフォルトでは NOARCHIVELOG モードが選択されます。制御ファイルの作成後に、ALTER DATABASE 文を使用した場合、ARCHIVELOG モードと NOARCHIVELOG モードを切り替えることができます。

character_set_clause

キャラクタ・セットを指定した場合、制御ファイルのキャラクタ・セット情報を再構成します。データベースのケース・メディア・リカバリが必要となる場合、データベースがオープンする前にこの情報が使用可能になり、リカバリ時に表領域名が正しく解析されます。この句は、デフォルトの US7ASCII 以外のキャラクタ・セットを使用している場合にのみ有効です。

制御ファイルを再作成し、表領域のリカバリに Recovery Manager を使用している場合、およびデータ・ディクショナリに格納されている異なるキャラクタ・セットを指定する場合、表領域はリカバリされません（ただし、データベースのオープン時、制御ファイルのキャラクタ・セットは、データ・ディクショナリの正しいキャラクタ・セットに更新されます）。

注意： この句ではデータベース・キャラクタ・セットを変更できません。

参照： 表領域のリカバリの詳細は、『Oracle9i Recovery Manager ユーザーズ・ガイド』および『Oracle9i Recovery Manager リファレンス』を参照してください。

例

CREATE CONTROLFILE の例 この文は、制御ファイルを再作成します。この文によって、データベース demo は WE8DEC キャラクタ・セットで作成されます。この例で使用する単語 *path* には、ご使用のシステムでの適切な Oracle ディレクトリへのパスを挿入してください。

```
STARTUP NOMOUNT

CREATE CONTROLFILE REUSE DATABASE "demo" NORESETLOGS NOARCHIVELOG
    MAXLOGFILES 32
    MAXLOGMEMBERS 2
    MAXDATAFILES 32
    MAXINSTANCES 1
    MAXLOGHISTORY 449
LOGFILE
    GROUP 1 '/path/oracle/dbs/t_log1.f' SIZE 500K,
    GROUP 2 '/path/oracle/dbs/t_log2.f' SIZE 500K
# STANDBY LOGFILE
DATAFILE
    '/path/oracle/dbs/t_db1.f',
    '/path/oracle/dbs/dbu19i.dbf',
    '/path/oracle/dbs/tbs_11.f',
    '/path/oracle/dbs/smundo.dbf',
    '/path/oracle/dbs/demo.dbf'
CHARACTER SET WE8DEC
;
```

CREATE DATABASE

注意： この文を実行すると、データベースの初期設定が行われ、指定ファイル内の現行のデータは消去されます。そのことを十分理解したうえで、この文を使用してください。

セキュリティ拡張の登録に関する注意： Oracle9i の今回のリリース以降では、デフォルトのデータベース・ユーザー・アカウントのセキュリティを保証する機能拡張があります。

- 今回のリリースから、Oracle Database Configuration Assistant (DBCA) の初回インストール中、SYS、SYSTEM、SCOTT、DBSNMP、OUTLN、AURORA\$JIS\$UTILITY\$、AURORA\$ORB\$UNAUTHENTICATED および OSE\$HTTP\$ADMIN を除くデータベース・ユーザー・アカウントは、ロックされ、期限切れとなります。ロックされたアカウントをアクティブにするには、データベース管理者は手動でロックを解除し、新しいパスワードを再度割り当てる必要があります。
 - データベース・サーバーの次のリリースでは、ユーザー SYS および SYSTEM にデフォルトのパスワードを割り当てられるのではなく、データベースの初回インストール中、DBCA によってそれらのユーザーのパスワード入力を促すプロンプトが表示されます。また、手動で発行された SQL 文 CREATE DATABASE には、これら 2 ユーザーへのパスワード指定が必須となります。
 - Oracle9i は、あらゆるタイプのインストール中に作成される、または SQL 文 CREATE DATABASE によって作成されるデフォルトのデータベース・ユーザーとしてユーザー SYSTEM をサポートする最後のメジャー・リリースとなる予定です。
-

用途

CREATE DATABASE 文は、汎用的なデータベースを作成する場合に使用します。

この文は、データベースの初期使用に備えて、指定した既存のデータ・ファイル上のデータがすべて消去されます。したがって、既存のデータベースに対してこの文を実行した場合、データ・ファイル上のすべてのデータが失われます。

この文を指定した場合、データベースの作成後、データベースは排他モードまたはパラレル・モード（初期化パラメータ CLUSTER_DATABASE の値に依存する）でマウントおよびオープンされ、通常の用途に使用可能になります。同時に、データベースの表領域およびロールバック・セグメントを作成できます。

参照：

- データベースの変更の詳細は、8-9 ページの「ALTER DATABASE」を参照してください。
- Oracle9i Java Virtual Machine (Java VM) の作成の詳細は、『Oracle9i Java Developer's Guide』を参照してください。
- ロールバック・セグメントおよび表領域の作成の詳細は、13-75 ページの「CREATE ROLLBACK SEGMENT」および 14-62 ページの「CREATE TABLESPACE」を参照してください。

前提条件

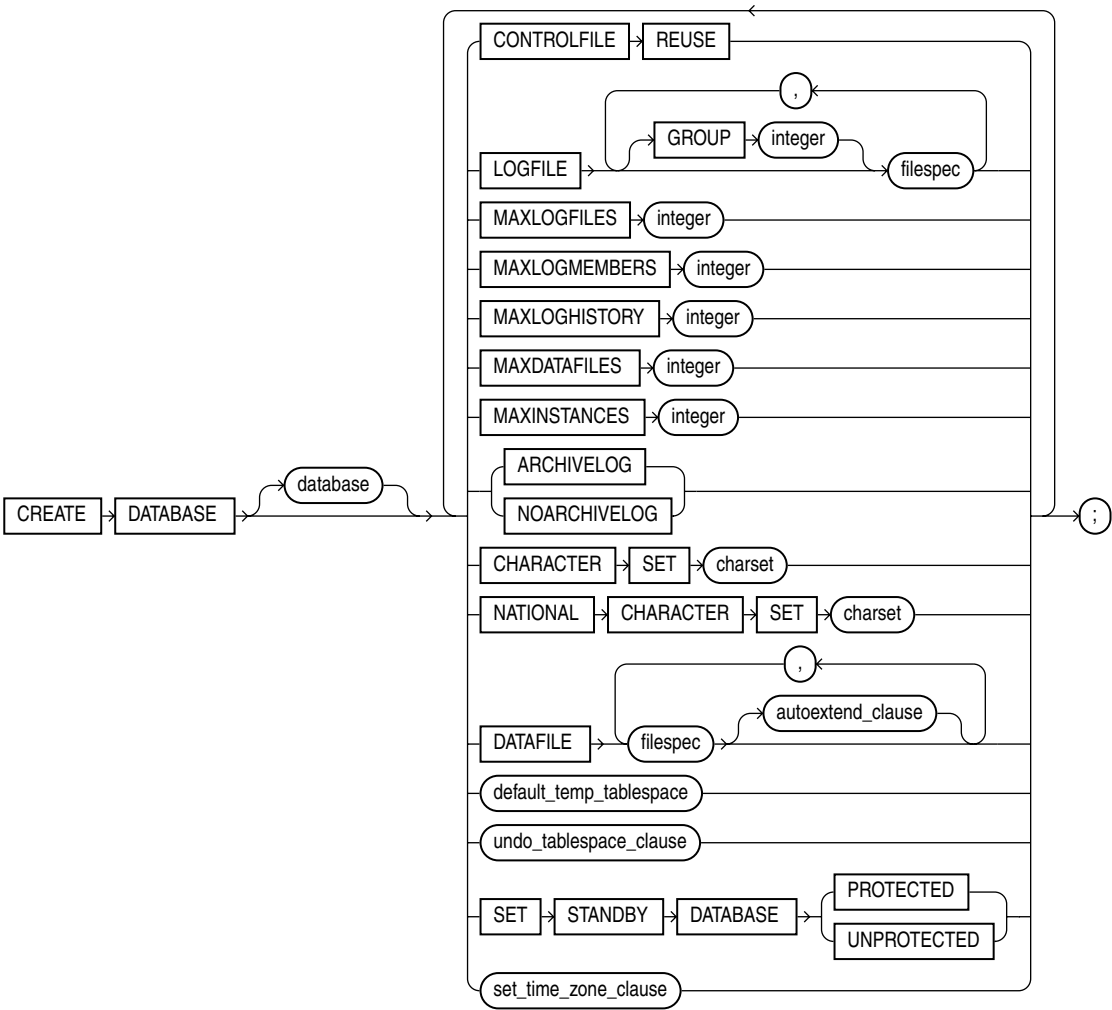
データベースを作成する場合は、SYSDBA システム権限が必要です。

初期化パラメータ REMOTE_LOGIN_PASSWORDFILE が EXCLUSIVE に設定されている場合、Oracle は、データベースを再作成しようとした際にエラーを戻します。エラーを発生させないためには、データベースを再作成する前にパラメータを SHARED に設定するか、またはパスワード・ファイルを再作成します。

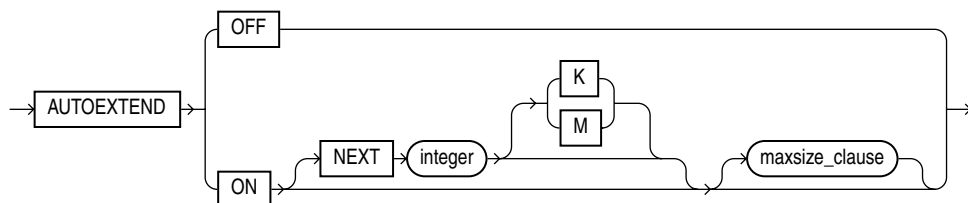
参照： REMOTE_LOGIN_PASSWORDFILE パラメータの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

構文

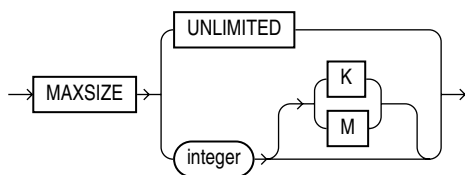
create_database::=



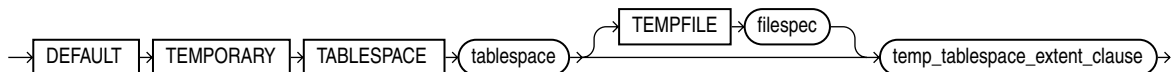
autoextend_clause::=



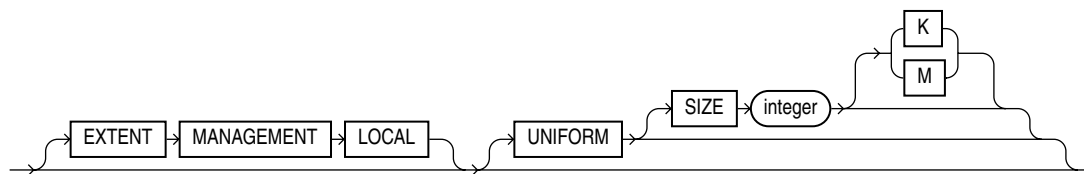
maxsize_clause::=



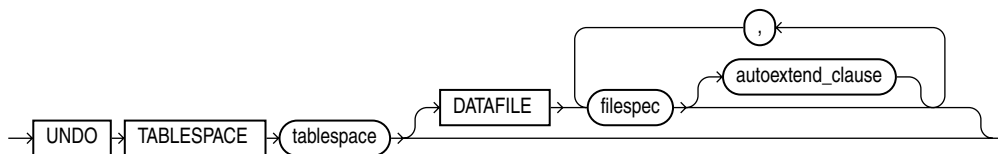
default_temp_tablespace::=



temp_tablespace_extent::=

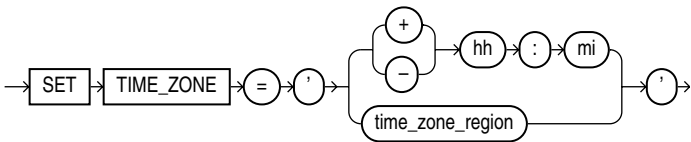


undo_tablespace_clause::=



filespec: 16-27 ページの「[filespec](#)」を参照してください。

set_time_zone_clause::=



キーワードとパラメータ

database

作成するデータベースの名前を指定します。名前の長さは最大 8 バイトです。データベース名には、ASCII 文字のみ指定できます。指定したデータベース名は、Oracle によって制御ファイルに記録されます。後で、ALTER DATABASE 文を発行したときにデータベース名を明示的に指定すると、制御ファイル内の名前に基づいて、そのデータベース名が検証されます。

注意： データベース名には、ヨーロッパやアジアのキャラクタ・セットの中の特文字は使用できません。たとえば、ウムラウト付きの文字は使用できません。

CREATE DATABASE 文でデータベース名を指定しない場合、初期化パラメータ DB_NAME で指定した名前が採用されます。初期化パラメータ DB_NAME が設定されており、そのパラメータの値とは異なる名前を指定した場合、Oracle はエラーを戻します。

参照： データベース名のその他の規則については、2-110 ページの「[スキーマ・オブジェクト名のネーミング計画](#)」を参照してください。

CONTROLFILE REUSE 句

CONTROLFILE REUSE を指定して、初期化パラメータ CONTROL_FILES で特定される既存の制御ファイルを再利用することを指定します。これによって、現在保存されている情報は無視され、上書きされます。通常、この句は、初めてデータベースを作成する際ではなく、データベースを再作成する際に使用します。制御ファイルを既存のファイルより大きくするためのパラメータ値もあわせて指定する場合、この句は使用できません。MAXLOGFILES、MAXLOGMEMBERS、MAXLOGHISTORY、MAXDATAFILES および MAXINSTANCES がこのようなパラメータです。

この句を指定しないと、CONTROL_FILES で指定した制御ファイルのいずれかがすでに存在する場合、Oracle はエラー・メッセージを戻します。

LOGFILE 句

REDO ログ・ファイルとして使用する 1 つ以上のファイルを指定します。各 *filespec* には、1 つ以上の REDO ログ・ファイルのメンバー（コピー）を含む REDO ログ・ファイル・グループを指定します。CREATE DATABASE 文に指定したすべての REDO ログ・ファイルは、REDO ログのスレッド番号 1 に追加されます。

参照： *filespec* の構文については、16-27 ページの「[filespec](#)」を参照してください。

GROUP *integer* REDO ログ・ファイル・グループの識別番号を指定します。*integer* の値は 1 ～ MAXLOGFILES パラメータの値の範囲です。データベースには、2 つ以上の REDO ログ・ファイル・グループが必要です。同一の GROUP 値を持つ REDO ログ・ファイル・グループは複数指定できません。このパラメータを指定しない場合、値が自動的に生成されます。REDO ログ・ファイル・グループの GROUP 値は、動的パフォーマンス・ビュー V\$LOG で調べることができます。

LOGFILE を指定しない場合、次のようになります。

- 初期化パラメータ DB_CREATE_ONLINE_LOG_DEST_n と DB_CREATE_FILE_DEST のいずれか（または両方）を設定した場合、システムが生成する名前で、サイズが 100MB の 2 つの Oracle 管理ログ・ファイルが作成されます。ファイルは DB_CREATE_ONLINE_LOG_DEST_n が指定されていれば、そのディレクトリに、そうでなければ DB_CREATE_FILE_DEST パラメータで指定されるディレクトリに作成されます。
- どちらのパラメータも設定されていない場合、2 つの REDO ログ・ファイル・グループが作成されます。各デフォルト・ファイルの名前およびサイズは、使用するオペレーティング・システムによって異なります。

MAXLOGFILES 句

データベース用に作成可能な REDO ログ・ファイル・グループの最大数を指定します。この値を基にして、制御ファイル内で REDO ログ・ファイル名に割り当てられる領域が決定されます。デフォルト値、最小値および最大値は、使用するオペレーティング・システムによって異なります。

MAXLOGMEMBERS 句

REDO ログ・ファイル・グループのメンバー（コピー）の最大数を指定します。この値を基にして、制御ファイル内で REDO ログ・ファイル名に割り当てられる領域が決定されます。最小値は 1 です。最大値およびデフォルト値は、使用するオペレーティング・システムによって異なります。

MAXLOGHISTORY 句

このパラメータは、Real Application Clusters 環境で ARCHIVELOG モードの Oracle を使用している場合に便利です。Real Application Clusters の自動メディア・リカバリに使用するアーカイブ REDO ログ・ファイルの最大数を指定します。この値を基にして、制御ファイル内でアーカイブ REDO ログ・ファイル名に割り当てられる領域が決定されます。最小値は 0（ゼロ）です。デフォルト値は MAXINSTANCES 値の倍数で、使用するオペレーティング・システムによって異なります。最大値は、制御ファイルの最大サイズの制限のみを受けます。

MAXDATAFILES 句

CREATE DATABASE または CREATE CONTROLFILE 実行時の、制御ファイルのデータ・ファイル・セクションの初期サイズ設定を指定します。値が MAXDATAFILES より大きく、DB_FILES 以下のファイルを追加した場合、データ・ファイル・セクションにさらに多くのファイルを格納できるように、Oracle の制御ファイルが自動的に拡張されます。

インスタンスでアクセスできるデータ・ファイルの数は、初期化パラメータ DB_FILES の制限を受けます。

MAXINSTANCES 句

作成したデータベースを同時にマウントおよびオープンするインスタンスの最大数を指定します。この値は、初期化パラメータ INSTANCES の値より優先されます。最小値は 1 です。最大値およびデフォルト値は、使用するオペレーティング・システムによって異なります。

ARCHIVELOG | NOARCHIVELOG

ARCHIVELOG REDO ログ・ファイル・グループを再利用する前に、グループの内容をアーカイブする場合は、ARCHIVELOG を指定します。この句を指定した場合、メディア・リカバリができるようになります。

NOARCHIVELOG REDO ログ・ファイル・グループを再利用する前に、グループの内容をアーカイブする必要がない場合は、NOARCHIVELOG を指定します。この句を指定した場合、メディア・リカバリはできません。

デフォルトは NOARCHIVELOG モードです。データベースの作成後に、ALTER DATABASE 文を使用した場合、ARCHIVELOG モードと NOARCHIVELOG モードを切り替えることができます。

CHARACTER SET 句

データベースにデータを格納するときのキャラクタ・セットを指定します。サポートされているキャラクタ・セットおよびこのパラメータのデフォルト値は、使用するオペレーティング・システムによって異なります。

制限事項：AL16UTF16 キャラクタ・セットは、データベース・キャラクタ・セットとして指定できません。

参照： キャラクタ・セットの選択の詳細は、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

NATIONAL CHARACTER SET 句

データ型が NCHAR、NCLOB または NVARCHAR2 で定義された列にデータを格納する際に使用する各国語キャラクタ・セット (AL16UTF16 と UTF8 のいずれか) を指定します。デフォルトは 'AL16UTF16' です。

参照： Unicode データ型のサポートの詳細は、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

DATAFILE 句

データ・ファイルとして使用する 1 つ以上のファイルを指定します。ファイルは、すべて SYSTEM 表領域の一部となります。

この句を指定しない場合、次のようになります。

- 初期化パラメータ DB_CREATE_FILE_DEST が設定されている場合、パラメータで指定したデフォルトのファイル格納先に、サイズが 100MB でシステムが生成する名前の、Oracle 管理データ・ファイルが作成されます。
- DB_CREATE_FILE_DEST 初期化パラメータが設定されていない場合、1 つのデータ・ファイルが作成されます。そのファイル名およびサイズは、使用するオペレーティング・システムによって異なります。

注意： SYSTEM 表領域に割り当てる初期領域は、全体で 5MB 以上になるように指定してください。

参照： 構文の詳細は、16-27 ページの「[filespec](#)」を参照してください。

autoextend_clause

autoextend_clause は、新しいデータ・ファイルまたはテンポラリ・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、これらのファイルは自動的に拡張されません。

ON ON を指定すると、自動拡張を使用可能にします。

OFF OFF を指定すると、自動拡張を使用禁止にします。

注意： 自動拡張を使用禁止にする場合は、NEXT および MAXSIZE の値を 0（ゼロ）に設定します。後続の文で自動拡張を使用可能に戻す場合は、これらの値を設定しなおす必要があります。

NEXT NEXT 句を使用すると、エクステントがさらに必要になった場合にデータ・ファイルに自動的に割り当てられるディスク領域の増分サイズを、バイト単位で指定できます。K または M を使用すると、KB または MB 単位で指定できます。デフォルトのサイズは 1 データ・ブロックです。

MAXSIZE MAXSIZE を使用すると、データ・ファイルの自動拡張で使用されるディスク領域の最大サイズを指定できます。

UNLIMITED データ・ファイルまたはテンポラリ・ファイルに割り当てられるディスク領域を制限しない場合は、UNLIMITED 句を使用します。

default_temp_tablespace

データベースのデフォルトの一時表領域を作成します。別の一時表領域を指定しないユーザーに、この一時表領域が割り当てられます。この句を指定しない場合、SYSTEM 表領域がデフォルトの一時表領域になります。

注意： オペレーティング・システムによっては、テンポラリ・ファイルのブロックが実際にアクセスされるまで、テンポラリ・ファイル用の領域が割り当てられない場合があります。領域の割当ての遅延のため、前もってテンポラリ・ファイルの作成およびサイズ変更を行う必要があります。ただし、後でテンポラリ・ファイルが使用されるときに、十分なディスク領域を使用可能にする必要があります。ご使用のオペレーティング・システムのドキュメントを参照し、このような方法でテンポラリ・ファイル領域が割り当てられるかどうかを判断してください。

制限事項：

- この句では、SYSTEM 表領域を指定できません。
- デフォルトの一時表領域は、標準的なブロック・サイズである必要があります。

`temp_tablespace_extent` 句を使用すると、表領域の管理方法を指定できます。

EXTENT MANAGEMENT LOCAL この句は、表領域の一部がビットマップ用に確保されることを示します。すべての一時表領域にはローカル管理のエクステントがあるため、この句はオプションです。

UNIFORM integer 一時表領域のエクステント・サイズをバイトで指定します。表領域のエクステントはすべて同じサイズ（均一）です。この句を指定しない場合は、均一のエクステント 1MB を使用します。

SIZE integer 表領域のエクステントのサイズをバイト単位で指定します。K または M を使用して、KB または MB 単位で指定することもできます。

SIZE を指定しない場合は、デフォルトのエクステント・サイズ 1MB を使用します。

参照： ローカルで管理される表領域の詳細は、『Oracle9i データベース概要』を参照してください。

undo_tablespace_clause

自動 UNDO 管理モードのインスタンスをオープンすると、`undo_tablespace_clause` を指定でき、UNDO データに使用する表領域を作成できます。UNDO 領域管理をロールバック・セグメントによって処理する場合、この句は省略します。

- この句を指定すると、`tablespace` という名前の UNDO 表領域、および UNDO 表領域の一部として指定したデータ・ファイルが作成され、インスタンスの UNDO 表領域としてこの表領域が割り当てられます。Oracle は、この UNDO 表領域を使用して UNDO データを管理します。この句の `DATAFILE` 句は、12-27 ページの「[DATAFILE 句](#)」と同様に動作します。

注意： 初期化パラメータ・ファイル内の `UNDO_TABLESPACE` 初期化パラメータの値を指定する場合は、データベースをマウントする前に、この句で同じ名前を指定してください。この名前が異なると、データベースのオープン時にエラーが戻されます。

- この句を省略すると、SYS_UNDOTBS という名前のデフォルトの UNDO 表領域でデフォルトのデータベースが作成され、インスタンスの UNDO 表領域としてこのデフォルトの表領域が割り当てられます。この UNDO 表領域は、CREATE DATABASE 文で使用するデフォルトのファイルから、初期エクステンツが 10MB のディスク領域を割り当てます。Oracle は、システムが生成するデータ・ファイル (12-27 ページの「[DATAFILE 句](#)」で説明) を処理します。Oracle が UNDO 表領域を作成できない場合、CREATE DATABASE 操作全体が失敗します。

参照：

- UNDO_MANAGEMENT パラメータを使用した自動 UNDO 管理モードのデータベース・インスタンスをオープンする場合の詳細は、『Oracle9i データベース・リファレンス』を参照してください。
- 自動 UNDO 管理および UNDO 表領域の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。
- データベース作成後の UNDO 表領域を作成する場合の詳細は、14-62 ページの「[CREATE TABLESPACE](#)」を参照してください。

SET STANDBY DATABASE 句

SET STANDBY DATABASE 句を指定すると、データベース環境が**データ非消失モード**かどうかを設定することができます。このモードでは、プライマリ・データベースとスタンバイ・データベースが完全に一致する管理が最優先されます。スタンバイ・データベースをマウントする必要があり、Real Application Clusters 以外のインスタンスでは、プライマリ・データベースを排他モードでもオープンできます。

PROTECTED PROTECTED を指定すると、プライマリ・データベースからスタンバイ・データベースへの最後の接続が解除するときに、プライマリ・データベースがオープンおよびオープンし続けるために、スタンバイ・インスタンスが、ログ・ライター (LGWR) ・プロセスによってアーカイブされる 1 つ以上のアーカイブ・ログの出力先を含む必要があることを示します。Real Application Clusters 環境では、プライマリ・データベースを持つすべてのインスタンスの LGWR プロセスが、同一のスタンバイ・データベースへのアーカイブをオープンすることが検証されます。

スタンバイ・データベースへの最後の接続が解除されると、プライマリ・インスタンスは停止します。そのため、プライマリ・データベースとスタンバイ・データベースの完全な一致がデータベースの可用性より重要な場合のみにこの設定を使用します。

UNPROTECTED UNPROTECTED を指定すると、インスタンスがログ・ライター・プロセスによって管理されるスタンバイ・データベースを必要としないことを示します。これはデフォルトです。

プライマリ・データベースとスタンバイ・データベースの完全な一致がデータベースの可用性より重要でない場合にこの設定を使用します。

データベースが PROTECTED と UNPROTECTED モードのどちらであるか判断するには、V\$DATABASE 動的パフォーマンス・ビューの STANDBY_DATABASE 列を問い合わせます。

参照：

- スタンバイ・データベースの管理の詳細は、『Oracle9i Data Guard 概要および管理』を参照してください。
- 動的パフォーマンス・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

set_time_zone_clause

SET TIME_ZONE 句を使用すると、データベースのタイム・ゾーンを設定できます。次の 2 つ方法でタイム・ゾーンを設定します。

- UTC からの置換値を指定。hh:mm の有効範囲は、-12:00 ～ +14:00 です。
- タイム・ゾーン地域を指定。有効な地域名を確認するには、V\$TIMEZONE_NAMES 動的パフォーマンス・ビューの TZNAME 列を問い合わせます。

参照： 動的パフォーマンス・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

すべての TIMESTAMP WITH LOCAL TIME_ZONE データは、ディスクに格納されるときにデータベースのタイム・ゾーンに正規化されます。SET TIME_ZONE 句を指定しない場合、サーバーのオペレーティング・システムのタイム・ゾーンが使用されます。オペレーティング・システムのタイム・ゾーンが Oracle で有効でない場合、データベースのタイム・ゾーンは、デフォルトで UTC になります。

例

CREATE DATABASE の例 次の文は、すべての各引数を完全に指定してデータベースを作成します。

```
CREATE DATABASE sample
  CONTROLFILE REUSE
  LOGFILE
    GROUP 1 ('diskx:log1.log', 'disky:log1.log') SIZE 50K,
    GROUP 2 ('diskx:log2.log', 'disky:log2.log') SIZE 50K
  MAXLOGFILES 5
  MAXLOGHISTORY 100
  MAXDATAFILES 10
  MAXINSTANCES 2
  ARCHIVELOG
  CHARACTER SET UTF8
  NATIONAL CHARACTER SET AL16UTF16
  DATAFILE
    'disk1:df1.dbf' AUTOEXTEND ON,
    'disk2:df2.dbf' AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED
  DEFAULT TEMPORARY TABLESPACE temp_ts
  UNDO TABLESPACE undo_ts
  SET TIME_ZONE = '+02:00';
```

CREATE DATABASE LINK

用途

CREATE DATABASE LINK 文は、データベース・リンクを作成する場合に使用します。**データベース・リンク**とは、リモート・データベース上のオブジェクトにアクセスできるローカル・データベース上のスキーマ・オブジェクトです。リモート・データベースは、Oracle データベースである必要はありません。

データベース・リンクを作成した場合、そのリンクを利用して、リモート・データベース上の表およびビューを参照できます。また、表名またはビュー名に `@dblink` を付けることによって、SQL 文の中でリモート表またはリモート・ビューを参照できます。SELECT 文を使用すると、リモート表またはビューの間合せができます。Oracle 分散オプション付きで使用する場合は、INSERT 文、UPDATE 文、DELETE 文または LOCK TABLE 文を使用してもリモート表およびビューにアクセスできます。

参照：

- PL/SQL ファンクション、プロシージャ、パッケージおよびデータ型を使用してリモート表またはビューへアクセスする方法については、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。
- ALL_DB_LINKS、DBA_DB_LINKS および USER_DB_LINKS データ・ディクショナリ・ビューの既存のデータベース・リンクの詳細、および V\$DBLINK 動的パフォーマンス・ビューを使用して既存のリンクのパフォーマンスを監視する方法については、『Oracle9i データベース・リファレンス』を参照してください。
- 既存のデータベース・リンクの削除については、15-63 ページの「[DROP DATABASE LINK](#)」を参照してください。
- DML 操作でのリンクの使用については、16-56 ページの「[INSERT](#)」、17-64 ページの「[UPDATE](#)」、15-49 ページの「[DELETE](#)」および 16-72 ページの「[LOCK TABLE](#)」を参照してください。

前提条件

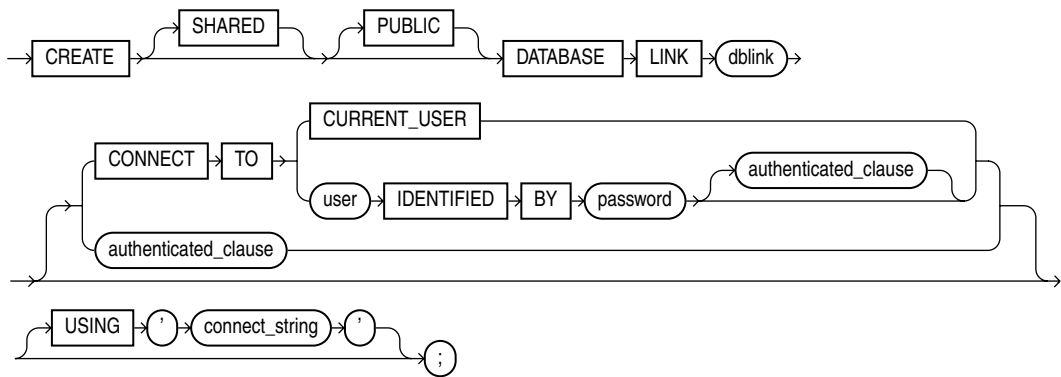
プライベート・データベース・リンクを作成する場合、CREATE DATABASE LINK システム権限が必要です。パブリック・データベース・リンクを作成する場合は、CREATE PUBLIC DATABASE LINK システム権限が必要です。また、リモートの Oracle データベースに対する CREATE SESSION 権限が必要です。

なお、ローカルとリモートの両方の Oracle データベースに、Oracle Net をインストールしておく必要があります。

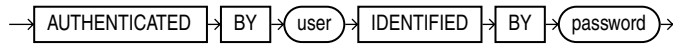
Oracle 以外のシステムにアクセスする場合は、Oracle 異機種間サービスを使用する必要があります。

構文

create_database_link::=



authenticated_clause::=



キーワードとパラメータ

SHARED

SHARED を指定して、1 つのネットワーク接続を使用することによって、複数ユーザーで共有できるパブリック・データベース・リンクを作成します。

PUBLIC

PUBLIC を指定して、すべてのユーザーが使用可能なパブリック・データベース・リンクを作成します。この句を指定しない場合、データベース・リンクはプライベートとなり、作成したユーザー専用になります。

参照： 12-38 ページの「[パブリック・データベース・リンクの例](#)」を参照してください。

dblink

データベース・リンクの完全な名前または名前の一部を指定します。GLOBAL_NAMES 初期パラメータの値は、データベース・リンクが接続するデータベースと同じ名前を持つ必要があるかどうかを指定します。

Real Application Clusters 構成の 1 つのセッションまたは 1 つのインスタンスでオープンできるデータベース・リンクの最大数は、OPEN_LINKS および OPEN_LINKS_PER_INSTANCE 初期パラメータの値で決まります。

制限事項：他のユーザーのスキーマ内にはデータベース・リンクを作成できません。また、*dblink* はスキーマ名を付けて指定できません（データベース・リンク名にはピリオドを指定できるため、*ralph.linktosales* のような名前を付けた場合、スキーマ *ralph* の *linktosales* という名前のデータベース・リンクと解析されるのではなく、名前全体が自分のスキーマにあるデータベース・リンク名と解析されます）。

参照：

- データベース・リンクのネーミングのガイドラインについては、2-113 ページの「[リモート・データベース内のオブジェクトの参照](#)」を参照してください。
- GLOBAL_NAMES、OPEN_LINKS および OPEN_LINKS_PER_INSTANCE 初期パラメータについては、『Oracle9i データベース・リファレンス』を参照してください。

CONNECT TO 句

CONNECT TO によって、リモート・データベースへの接続が可能になります。

CURRENT_USER 句

CURRENT_USER を指定して、**現行のユーザーのデータベース・リンク**を作成します。正常にリンクを作成する場合、現行のユーザーが、リモート・データベースに有効なアカウントを持つグローバル・ユーザーである必要があります。

データベース・リンクが直接使用される場合（ストアド・オブジェクト内から使用されていない場合）、現行のユーザーは接続ユーザーと同じです。

データベース・リンクを開始するストアド・オブジェクト（プロシージャ、ビューまたはトリガーなど）を実行する場合、CURRENT_USER とは、ストアド・オブジェクトを所有するユーザー名であり、オブジェクトをコールしたユーザー名ではありません。たとえば、データベース・リンクが（scott によって作成された）プロシージャ scott.p 内にあり、ユーザー jane がプロシージャ scott.p をコールした場合、現行のユーザーは scott になります。

ただし、ストアド・オブジェクトが実行者権限ファンクション、プロシージャまたはパッケージである場合、実行者認可 ID はリモート・ユーザーとしての接続に使用されます。たとえば、権限を持つデータベース・リンクがプロシージャ scott.p（scott によって作成された実行者権限プロシージャ）内にあり、ユーザー Jane がプロシージャ scott.p をコールした場合、CURRENT_USER は jane であり、プロシージャは、Jane の権限で実行されます。

参照：

- 実行者権限ファンクションの詳細は、12-47 ページの「[CREATE FUNCTION](#)」を参照してください。
- 12-37 ページの「[CURRENT_USER の例](#)」を参照してください。

user IDENTIFIED BY password

固定ユーザー・データベース・リンクを使用して、リモート・データベースに接続するためのユーザー名およびパスワードを指定します。この句を指定しない場合、データベース・リンクでは、データベースに接続している各ユーザーのユーザー名およびパスワードが使用されます。これを**接続ユーザー・データベース・リンク**といいます。

参照： 12-37 ページの「[固定ユーザーの例](#)」を参照してください。

authenticated_clause

ターゲット・インスタンスのユーザー名およびパスワードを指定します。この句は、リモート・サーバーに対してユーザーを認証するもので、セキュリティ上必要です。指定するユーザー名およびパスワードは、リモート・インスタンスで有効なユーザー名およびパスワードである必要があります。ユーザー名およびパスワードは、認証用としてのみ使用されます。このユーザーを対象とした認証以外の操作はありません。

SHARED 句を使用している場合は、必ずこの句を指定してください。

USING 'connect string'

リモート・データベースのサービス名を指定します。

参照： リモート・データベースの指定の詳細は、『Oracle9i Net Services 管理者ガイド』を参照してください。

例

CURRENT_USER の例 次の文は、デモ・データベース表を使用して、現行のユーザーのデータベース・リンクを定義します。

```
CREATE DATABASE LINK sales.hq.acme.com
CONNECT TO CURRENT_USER
USING 'sales';
```

固定ユーザーの例 次の文では、sales.hq.acme.com という名前の固定ユーザー・データベース・リンクを定義します。

```
CREATE DATABASE LINK sales.hq.acme.com
CONNECT TO hr IDENTIFIED BY hr
USING 'sales';
```

データベース・リンクが作成された後は、次の方法でリモート・データベース上のスキーマ scott 内の表に問い合わせることができます。

```
SELECT * FROM employees@sales.hq.acme.com;
```

次のように DML 文を使用して、リモート・データベース上のデータを変更できます。

```
INSERT INTO orders@sales.hq.acme.com
(customer_id, order_id, order_total)
VALUES (5001, 1235, 2000);
```

```
UPDATE orders@sales.hq.acme.com
SET order_total = order_total + 500;
```

```
DELETE FROM order_id@sales.hq.acme.com
WHERE order_id = 2443;
```

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。この文は、現行のユーザーが `hr.departments` 表に対する `SELECT` 権限を持っていることを想定しています。

```
SELECT *
FROM hr.departments@sales.hq.acme.com;
```

この文では、リモート・データベースのユーザー `scott` に接続してから、`adam` の `dept` 表への問合せが行われます。

`departments` 表がリモート・データベース上にあることを隠すために、シノニムを作成できます。次の文は、`dept` を参照すると、必ず、`hr` が所有するリモートの `departments` 表にアクセスします。

```
CREATE SYNONYM dept
FOR hr.departments@sales.hq.acme.com;
```

パブリック・データベース・リンクの例 次の文では、`sales.hq.acme.com` という名前の共有パブリックの固定ユーザー・データベース・リンクを定義します。このデータベース・リンクは、文字列サービス名 `'sales'` で指定したデータベースに対して、ユーザー `scott` (パスワード `tiger`) と対応しています。

```
CREATE SHARED PUBLIC DATABASE LINK sales.hq.acme.com
CONNECT TO hr IDENTIFIED BY hr
AUTHENTICATED BY anupam IDENTIFIED BY bhide
USING 'sales';
```

CREATE DIMENSION

用途

CREATE DIMENSION 文を使用すると、**ディメンション**を作成できます。ディメンションは、列集合の組の親子関係を定義します。列集合の列は、すべて同じ表から得られたものである必要があります。ただし、1つの列集合（またはレベル）の列は、別の集合の列とは異なる表から得ることができます。オプティマイザは、マテリアライズド・ビューとの関係を使用して**クエリー・リライト**を行います。サマリー・アドバイザは、特定のマテリアライズド・ビューの作成の推奨にこの関係を使用します。

注意： Oracle は、ディメンションの作成中に宣言する関係の妥当性チェックを自動的には行いません。*hierarchy_clause* および CREATE DIMENSION の *join_clause* で指定する関係の妥当性チェックを行うには、DBMS_OLAP.validate_dimension プロシージャを実行する必要があります。

参照：

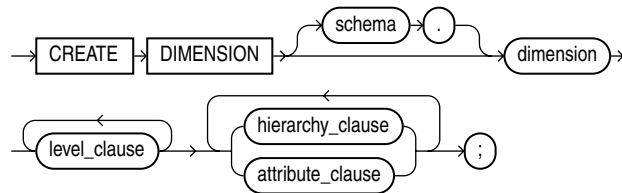
- マテリアライズド・ビューの詳細は、13-5 ページの「[CREATE MATERIALIZED VIEW](#)」を参照してください。
- クエリー・リライト、オプティマイザおよびサマリー・アドバイザの詳細は、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。
- DBMS_OLAP.validate_dimension プロシージャの詳細は、『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

前提条件

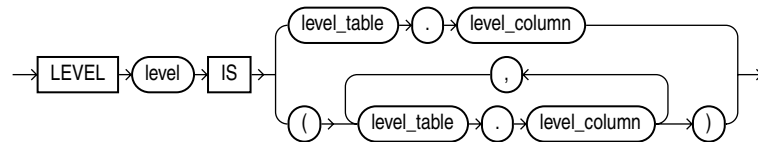
自分のスキーマ内にディメンションを作成する場合は、CREATE DIMENSION システム権限が必要です。他のユーザーのスキーマ内にディメンションを作成する場合は、CREATE ANY DIMENSION システム権限が必要です。どちらの場合も、ディメンションで参照されるオブジェクトに対して、SELECT オブジェクト権限が必要です。

構文

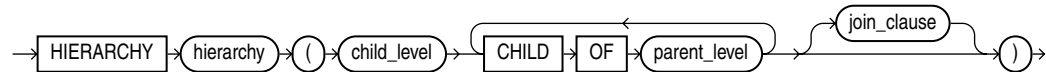
create_dimension::=



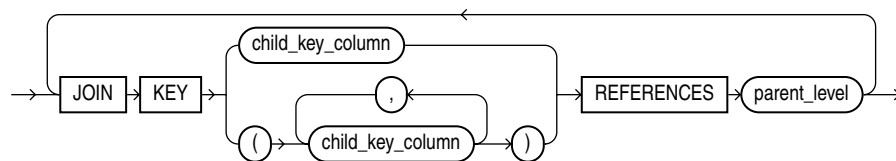
level_clause::=



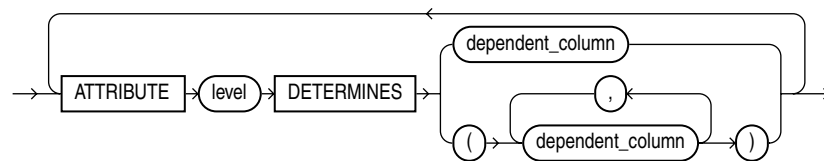
hierarchy_clause::=



join_clause::=



attribute_clause::=



キーワードとパラメータ

schema

ディメンションを作成するスキーマを指定します。*schema* を指定しない場合、自分のスキーマ内にディメンションが作成されます。

dimension

ディメンション名を指定します。名前は、スキーマ内で一意である必要があります。

level_clause

level_clause は、ディメンションのレベルを指定します。レベルは、ディメンション階層および属性を定義します。

level レベル名を指定します。

level_table . level_column レベルの列を指定します。最大 32 列を指定できます。この句で指定する表は、すでに存在している必要があります。

制限事項：

- レベルの列は、すべて同じ表から得られたものである必要があります。
- 異なるレベルの列が異なる表から得られる場合、*join_clause* を指定する必要があります。
- 指定する列の集合は、このレベルに一意である必要があります。
- 指定する列は、他のディメンションでは指定できません。
- 各 *level_column* は、NULL 以外である必要があります（ただし、この列は NOT NULL 制約を受ける必要はありません）。

hierarchy_clause

ディメンションの線形階層レベルを定義します。各階層が、ディメンションのレベル間で親子関係の連鎖を形成します。ディメンションの階層は、互いに依存していません。階層は、共通の列を持つことができます。

ディメンションの各レベルは、句の中で最高 1 回指定され、*level_clause* で名前を付けておく必要があります。

hierarchy 階層名を指定します。この名前は、ディメンションで一意である必要があります。

child_level 親レベルと n:1 の関係を持つレベルの名前を指定します。child_level の level_columns は NULL 以外である必要があります。各 child_level 値は、parent_level という名前の次の値を一意に定義します。

子 level_table が親 level_table と異なる場合、join_clause でそれらの結合関係を指定する必要があります。

parent_level レベル名を指定します。

join_clause

複数の表に列が含まれるディメンションに内部等価結合関係を指定できます。この句は、階層で指定されたすべての列が同じ表にあるとは限らない場合にのみ指定する必要があります、このときのみ指定できます。

child_key_column

親レベルの列と結合互換性のある 1 つ以上の列を指定します。

スキーマおよび各 child_column の表を指定しない場合、hierarchy_clause の CHILD OF 関係からスキーマおよび表が判断されます。child_key_column のスキーマおよび列を指定する場合は、hierarchy_clause の parent_level の子のスキーマおよび表と一致している必要があります。

parent_level

レベル名を指定します。

制限事項：

- 同じ階層の既存のレベルの組に対して、1 つの join_clause のみを指定できます。
- child_key_columns は NULL 以外であり、親キーが一意で NULL 以外である必要があります。条件を適用するために制約を定義する必要はありません。ただし、条件を満たさない場合、問合せが不適切な結果を戻すことがあります。
- 各子キーは、parent_level 表のキーと結合する必要があります。
- 自己結合はできません。つまり、child_key_columns を、parent_level として同じ表に置くことはできません。
- 子キー列は、すべて同じ表から得られたものである必要があります。
- 子キー列数は、parent_level の列数と一致し、列は結合可能である必要があります。
- 親レベルが複数の列で構成されている場合のみ、子キー列を指定します。

attribute_clause

*attribute_clause*によって、階層レベルによって一意に定義されている列を指定できます。*level*の列は、*dependent_columns*として同じ表からすべて得る必要があります。*dependent_columns*は、*level_clause*で指定されている必要はありません。

たとえば、階層レベルが市、都道府県名、および国の場合、市は市長、都道府県名は知事、国は首相を決定します。

例

CREATE DIMENSION の例 次の例は、デモ・スキーマ sh の customers_dim ディメンションを作成します。

```
CREATE DIMENSION customers_dim
  LEVEL customer    IS (customers.cust_id)
  LEVEL city        IS (customers.cust_city)
  LEVEL state       IS (customers.cust_state_province)
  LEVEL country     IS (countries.country_id)
  LEVEL subregion   IS (countries.country_subregion)
  LEVEL region      IS (countries.country_region)
  HIERARCHY geog_rollup (
    customer        CHILD OF
    city            CHILD OF
    state           CHILD OF
    country         CHILD OF
    subregion       CHILD OF
    region
  )
  JOIN KEY (customers.country_id) REFERENCES country
)
ATTRIBUTE customer DETERMINES
(cust_first_name, cust_last_name, cust_gender,
 cust_marital_status, cust_year_of_birth,
 cust_income_level, cust_credit_limit)
ATTRIBUTE country DETERMINES (countries.country_name)
;
```

CREATE DIRECTORY

用途

CREATE DIRECTORY 文は、ディレクトリ・オブジェクトを作成する場合に使用します。ディレクトリ・オブジェクトは、外部バイナリ・ファイル LOB (BFILE) および外部表データが存在するサーバー・ファイル・システム上のディレクトリの別名を示します。PL/SQL コードおよび OCI コールで BFILE を参照する際、オペレーティング・システムのパス名をハードコード化せずにディレクトリ名を使用できます。このため、ファイル管理の汎用性が向上します。

すべてのディレクトリは 1 つのネームスペースに作成されるため、個々のユーザーのスキーマで所有されるわけではありません。ディレクトリに対するオブジェクト権限を特定ユーザーに付与することによって、そのディレクトリ構造内に格納されている BFILE へのアクセスを制限できます。

参照：

- BFILE オブジェクトの詳細は、2-25 ページの「[ラージ・オブジェクト \(LOB\) データ型](#)」を参照してください。
- オブジェクト権限の付与の詳細は、16-31 ページの「[GRANT](#)」を参照してください。
- 14-29 ページの「CREATE TABLE」の「[external_table_clause](#)」を参照してください。

前提条件

ディレクトリを作成する場合は、CREATE ANY DIRECTORY システム権限が必要です。

ディレクトリを作成する場合、そのディレクトリに対する READ オブジェクト権限および WRITE オブジェクト権限が自動的に付与され、他のユーザーおよびロールにこれらの権限を付与できます。DBA も、これらの権限を他のユーザーおよびロールに付与できます。

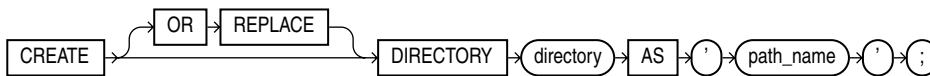
ディレクトリに対する WRITE 権限は、外部表との接続に便利です。これによって、権限受領者は、外部表のエージェントがディレクトリに書き込めるのがログ・ファイルなのか不良ファイルなのかを判断できます。

また、ファイル格納用として、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および書込み権限が正しく設定されていることを確認する必要があります。

ディレクトリに対して付与される権限は、オペレーティング・システムのディレクトリ用に定義されたアクセス権限とは無関係に作成されます。このため、これら2つは正確に対応しない場合があります。たとえば、デモ・ユーザー `hr` に、ディレクトリ・スキーマ・オブジェクトに対する読取り権限が付与されていても、それに対応するオペレーティング・システム上のディレクトリに **Oracle** プロセスに対する読取り権限が付与されていない場合は、エラーが発生します。

構文

create_directory::=



キーワードとパラメータ

OR REPLACE

既存のディレクトリ・データベース・オブジェクトを再作成する場合は、**OR REPLACE** を指定します。この句を指定した場合、既存のディレクトリに付与されているデータベース・オブジェクト権限を削除、再作成および再付与しなくても、そのディレクトリの定義を変更できます。

再定義したディレクトリに対する権限が付与されていたユーザーは、権限が再付与されなくてもそのディレクトリにアクセスできます。

参照： データベースからのディレクトリの削除については、15-67 ページの「**DROP DIRECTORY**」を参照してください。

directory

作成するディレクトリ・オブジェクトの名前を指定します。*directory* の最大長は 30 バイトです。ディレクトリ・オブジェクトは、スキーマ名で修飾できません。

注意： 指定したディレクトリが実際に存在するかどうかは検証されません。このため、オペレーティング・システムに存在するディレクトリを指定してください。また、オペレーティング・システムで使用するパス名が大文字と小文字を区別する場合は、必ず、正しい形式でディレクトリ名を指定してください（ただし、パス名の終わりにスラッシュを指定する必要はありません）。

'path_name'

ファイルが格納されているサーバー上のオペレーティング・システムのディレクトリのフルパス名を指定します。指定するフルパス名は、一重引用符で囲む必要があります。また、パス名の大文字と小文字は区別されます。

例

CREATE DIRECTORY の例 次の文は、サーバーのディレクトリを示す、ディレクトリのデータベース・オブジェクトを作成します。

```
CREATE DIRECTORY admin AS 'oracle/admin';
```

次の文は、オペレーティング・システムのディレクトリ `/private1/lob/files` に格納されている BFILE にアクセスできるように、ディレクトリのデータベース・オブジェクト `bfile_dir` を再定義します。

```
CREATE OR REPLACE DIRECTORY bfile_dir AS '/private1/LOB/files';
```

CREATE FUNCTION

用途

CREATE FUNCTION 文を使用すると、スタンドアロン・ストアド・ファンクションまたはコール仕様を作成できます。(また、CREATE PACKAGE 文を使用して、パッケージの一部としてファンクションを作成することもできます)。

ストアド・ファンクション (ユーザー・ファンクション) は、名前でコールできる PL/SQL 文の集合です。ストアド・ファンクションは、プロシージャとよく似ていますが、ファンクションは、コールした環境に値を戻す点で異なります。ストアド・ファンクションは、SQL 式の一部として使用できます。

コール仕様は、SQL および PL/SQL からコールできるように、Java メソッドまたは 3GL 世代言語 (3GL) ルーチンを宣言します。コール仕様は、コールされたときに起動する Java メソッドまたは共有ライブラリの名前付きファンクションを問い合わせます。引数および戻り値に対する型変換も問い合わせます。

参照：

- プロシージャおよびファンクションの概要は、13-58 ページの「[CREATE PROCEDURE](#)」を参照してください。
- ファンクション作成の例については、12-56 ページの「[例](#)」を参照してください。
- パッケージの作成については、13-46 ページの「[CREATE PACKAGE](#)」を参照してください。
- ファンクションの変更については、8-46 ページの「[ALTER FUNCTION](#)」を参照してください。
- 共有ライブラリについては、13-2 ページの「[CREATE LIBRARY](#)」を参照してください。
- スタンドアロン・ファンクションの削除については、15-69 ページの「[DROP FUNCTION](#)」を参照してください。
- 外部ファンクションの登録については、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

前提条件

ストアド・ファンクションを作成する前に、ユーザー SYS は、SQL スクリプト DBMSSTD_X.SQL を実行する必要があります。このスクリプトの正確な名前および格納位置は、使用するオペレーティング・システムによって異なります。

自分のスキーマ内にディメンションを作成する場合は、CREATE PROCEDURE システム権限が必要です。他のユーザーのスキーマ内にファンクションを作成する場合は、CREATE ANY PROCEDURE システム権限が必要です。他のユーザーのスキーマ内のファンクションを置換する場合は、ALTER ANY PROCEDURE システム権限が必要です。

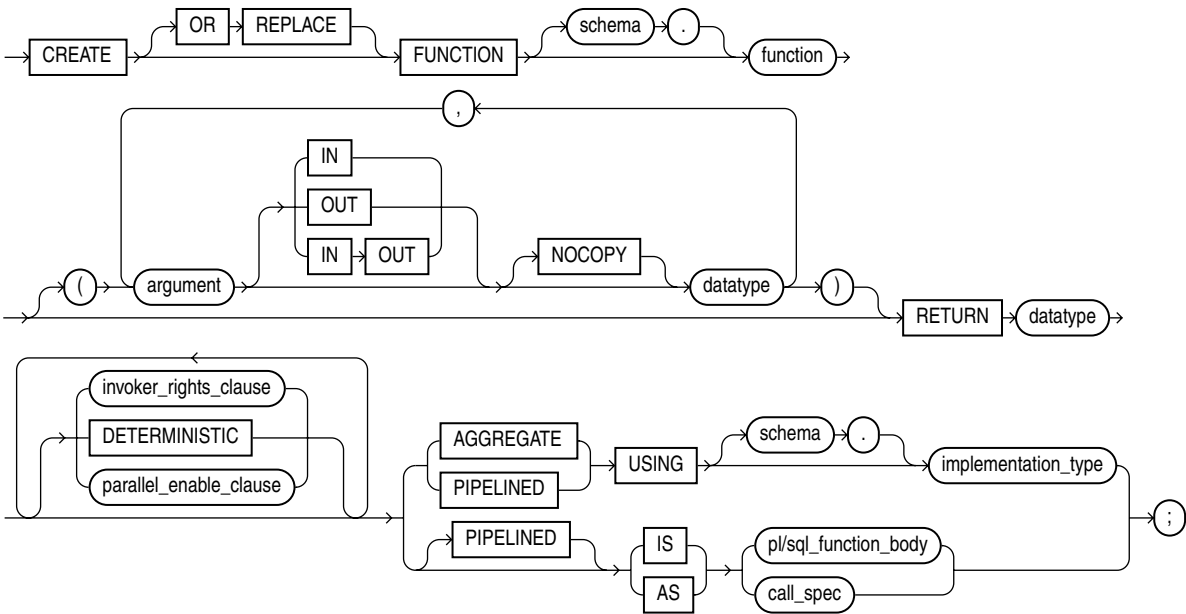
コール仕様を起動する場合、追加権限が必要となることがあります（たとえば、C コール仕様の C ライブラリに対する EXECUTE 権限）。

Oracle プリコンパイラ・プログラム内に CREATE FUNCTION 文を埋め込む場合、キーワード END-EXEC に続けて、各言語の埋込み SQL 文の終了記号を記述して文を終了する必要があります。

参照： このような前提条件の詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』または『Oracle9i Java Stored Procedures Developer's Guide』を参照してください。

構文

create_function::=



```

graph LR
    Input(( )) --> AUTHID[AUTHID]
    AUTHID --> CURRENT_USER[CURRENT_USER]
    AUTHID --> DEFINER[DEFINER]
    CURRENT_USER --> Output(( ))
    DEFINER --> Output
  
```

The diagram illustrates the flow of information from the `AUTHID` block to the `CURRENT_USER` and `DEFINER` blocks. An input arrow points to the `AUTHID` block. From `AUTHID`, two arrows branch out: one to the `CURRENT_USER` block and another to the `DEFINER` block. Both `CURRENT_USER` and `DEFINER` blocks have arrows pointing to a final output arrow on the right.

```

graph LR
    Start(( )) --> ParallelEnable[PARALLEL_ENABLE]
    ParallelEnable --> LParen("(")
    LParen --> Partition[PARTITION]
    Partition --> Argument(argument)
    Argument --> By[BY]
    By --> Any[ANY]
    By --> Hash[HASH]
    By --> Range[RANGE]
    Any --> RParen1(")")
    Hash --> RParen1
    Range --> RParen1
    RParen1 --> LParen2("(")
    LParen2 --> Column(column)
    Column --> RParen2(")")
    Column --> Comma(",")
    Comma --> LParen2
    RParen2 --> RParen3(")")
    RParen3 --> StreamingClause(streaming_clause)
    StreamingClause --> End(( ))
    LParen --> End
  
```

```

graph LR
    Input(( )) --> OrderCluster[ORDER  
CLUSTER]
    OrderCluster --> By[BY]
    By --> LParen(( ))
    LParen --> Column([column])
    Column --> RParen(( ))
    Column --> Column
  
```

```

graph LR
    In(( )) --> LANGUAGE[LANGUAGE]
    LANGUAGE --> Java_declaration(Java_declaration)
    LANGUAGE --> C_declaration(C_declaration)
    Java_declaration --> Out(( ))
    C_declaration --> Out

```

→ [JAVA] → [NAME] → (') → (string) → (') →

```

graph LR
    C[C] --> NAME[NAME]
    NAME --> name(name)
    name --> LIBRARY[LIBRARY]
    LIBRARY --> lib_name(lib_name)
    lib_name --> AGENT[AGENT]
    AGENT --> IN[IN]
    IN --> LP("(")
    LP --> argument(argument)
    argument --> RP(")")
    RP --> WITH[WITH]
    WITH --> CONTEXT[CONTEXT]
    CONTEXT --> PARAMETERS[PARAMETERS]
    PARAMETERS --> LP2("(")
    LP2 --> parameters(parameters)
    parameters --> RP2(")")
    RP2 --> END[...]
  
```

キーワードとパラメータ

OR REPLACE

既存のファンクションを再作成する場合は、OR REPLACE を指定します。この句を指定した場合、既存のファンクションに付与されているオブジェクト権限を削除、再作成および再付与しなくても、そのファンクションの定義を変更できます。ファンクションを再定義した場合、そのファンクションは再コンパイルされます。

再定義したファンクションに対して権限が付与されていたユーザーは、権限を再付与されなくても、そのファンクションにアクセスできます。

ファンクション索引がファンクションに依存している場合は、索引に DISABLED のマークを付けます。

参照： ファンクションの再コンパイルについては、8-46 ページの「ALTER FUNCTION」を参照してください。

schema

ファンクションを定義するスキーマを指定します。*schema* を指定しない場合、現行のスキーマにファンクションが作成されます。

function

作成するファンクションの名前を指定します。コンパイル・エラーでファンクション結果を作成した場合、Oracle はエラーを戻します。SHOW ERRORS コマンドを使用すると、関連するコンパイラ・エラー・メッセージを表示できます。

ユーザー定義ファンクションの制限事項

ユーザー定義ファンクションは、未変更定義を必要とする状況では使用できません。このため、ユーザー定義ファンクションは、次の場所では使用できません。

- CREATE TABLE 文または ALTER TABLE 文のチェック制約句
- CREATE TABLE 文または ALTER TABLE 文の DEFAULT 句

また、ファンクションが問合せまたは DML 文内からコールされる場合、そのファンクションには次の制限があります。

- OUT パラメータまたは IN OUT パラメータは指定できません。
- 現行のトランザクションのコミットやロールバック、セーブポイントの作成やロールバックまたはセッションやシステムの変更はできません。DDL 文が明示的に現行のトランザクションをコミットするため、ユーザー定義ファンクションは DDL 文を実行できません。

- ファンクションが **SELECT** 文からコールされる場合のデータベースへの書込みはできません。ただし、**DML** 文で副問合せからコールされるファンクションは、データベースへの書込みができます。
- ファンクションが **DML** 文からコールされる場合、ファンクションがコールされる文で修正される同じ表への書込みはできません。

OUT パラメータおよび **IN OUT** パラメータの制限を除き、**Oracle** は、**SQL** 文から直接コールされるファンクションのみでなく、そのファンクションがコールするすべてのファンクションや、そのファンクションまたはファンクションがコールすることによって実行される **SQL** 文からコールされるファンクションに対しても、これらの制限を適用します。

argument

ファンクションへの引数の名前を指定します。ファンクションが引数を受け入れない場合は、ファンクション名の後のカッコを省略できます。

制限事項：集計ファンクションを作成する場合、指定できる引数は1つのみです。

IN **IN** は、ファンクションをコールするときに、引数に値を指定する必要があることを示します。これはデフォルトです。

OUT **OUT** は、ファンクションによって引数の値が設定されることを示します。

IN OUT **IN OUT** は、引数の値を、ユーザーが指定することも、ファンクションで設定することも可能であることを示します。

NOCOPY **NOCOPY** は、できるだけ速く引数を渡すように指示します。この句は、**OUT** パラメータや **IN OUT** パラメータに対して、レコード、索引付き表、**VARRAY** などの大きい値を渡す際のパフォーマンスの向上に有効です。（**IN** パラメータ値には、常に **NOCOPY** が渡されます）。

- **NOCOPY** パラメータを指定すると、このパラメータに対応する実際の割当てとしてパッケージ変数が渡された場合に、パッケージ変数に対して行われた割当ては、すぐにこのパラメータに表示されます（またはこのパラメータに対して行われた割当ては、すぐにパッケージ変数に表示されます）。
- このパラメータまたは別のパラメータに対して行われた変更は、同じ変数が両方に渡された場合、両方の名前を介してすぐに参照できます。
- ファンクションが未処理例外で終了した場合、このパラメータに対するすべての割当てはコール元の変数で参照できます。

このような効果がないコールもあります。この効果に問題がない場合にのみ **NOCOPY** を使用してください。

RETURN 句

`datatype` には、ファンクションの戻り値のデータ型を指定します。すべてのファンクションが必ず値を戻すため、この句の指定は必須です。戻り値は、PL/SQL でサポートされているデータ型を持つことができます。

データ型には、データ長、精度および位取りを指定できません。Oracle では、そのファンクションがコールされた環境から戻り値のデータ長、精度および位取りを導出します。

参照： PL/SQL のデータ型については、『PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。

invoker_rights_clause

invoker_rights_clause によって、ファンクションを、スキーマを所有するユーザーの権限でそのスキーマ内で実行するか、または `CURRENT_USER` の権限でそのスキーマ内で実行するかを指定できます。

この句は、問合せ、DML 操作およびファンクションにおける動的 SQL 文の外部名の変換方法も定義します。

AUTHID 句

- ファンクションを `CURRENT_USER` 権限で実行する場合は、`CURRENT_USER` を指定します。この句は**実行者権限ファンクション**を作成します。

また、この句は、問合せ、DML 操作、および動的 SQL 文の外部名を `CURRENT_USER` のスキーマで変換することも指定します。他のすべての文の外部名は、ファンクションを含むスキーマで変換します。

- ファンクションを含むスキーマの所有者権限でファンクションを実行する場合、およびファンクションを含むスキーマで外部名を変換する場合は、`DEFINER` を指定します。これはデフォルトであり、**定義者権限ファンクション**を作成します。

参照：

- `CURRENT_USER` を判断する方法については、『Oracle9i データベース概要』および『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。
- 『PL/SQL 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。

DETERMINISTIC 句

DETERMINISTIC は、ファンクションが戻した結果が保存されたコピー（このようなコピーが使用可能な場合）をシステムが使用できるようにするための最適化のヒントです。保存されたコピーは、マテリアライズド・ビュー、ファンクション索引、または同じ SQL 文内の同じファンクションに対する別の呼出しから得ることができます。問合せオプティマイザは、保存されたコピーを使用するか、またはファンクションを再コールするかを選択できます。

ファンクションは、同じ値の引数でコールされると、必ず同じ結果を示します。したがって、システムがファンクションをコールしないことを選択した場合、結果が得られなくなります。このため、ファンクションが戻した結果に影響するような方法で、パッケージ変数の使用やデータベースへアクセスを行うファンクションは定義しないでください。

ファンクションは、ファンクション索引の式、またはマテリアライズド・ビューが REFRESH FAST または ENABLE QUERY REWRITE でマークされている場合のビューの問合せからコールされるように、DETERMINISTIC を宣言する必要があります。

参照：

- マテリアライズド・ビューについては、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。
- ファンクション索引の詳細は、12-58 ページの「[CREATE INDEX](#)」を参照してください。

parallel_enable_clause

PARALLEL_ENABLE は、パラレル問合せ操作のパラレル実行サーバーからファンクションを実行するための最適化ヒントを指定します。パッケージ変数などの変数は、パラレル実行サーバー内で共有されないことがあるため、ファンクションはそのようなセッション状態を使用してはいけません。

- オプションの PARTITION *argument* BY 句は、REF CURSOR 型を引数とするファンクションのみに使用できます。REF CURSOR 引数からファンクションへの入力のパパーティション化を定義できます。

ファンクションへの入力のパパーティション化は、ファンクションが表ファンクションとして使用される場合（問合せの FROM 句内）の問合せのパラレル化の方法に影響します。ANY を指定すると、データがパラレル実行サーバー間でランダムにパパーティション化されます。また、指定した列リストに RANGE または HASH パパーティション化を指定できます。

- オプションの *streaming_clause* を指定すると、指定した列リストでパラレル処理を順序付けまたはクラスタ化できます。
 - ORDER BY を指定すると、パラレル実行サーバーの行がローカルで順序付けされる必要があります。
 - CLUSTER BY を指定すると、パラレル実行サーバーの行が *column_list* で指定したキーと同じ値である必要があります。

これらのオプションのすべての句で指定した列は、ファンクションの REF CURSOR 引数によって戻された列を参照します。

参照： ユーザー定義の集計ファンクションの詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』、『Oracle9i Data Cartridge Developer's Guide』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。

PIPELINED 句

PIPELINED を使用すると、**表ファンクション**の結果を繰り返し戻すよう、Oracle に指示できます。表ファンクションは、コレクション型（ネストした表または VARRAY）を戻します。問合せの FROM 句のファンクション名の前に TABLE キーワードを使用し、表ファンクションを問い合わせます。たとえば、次のように指定します。

```
SELECT * FROM TABLE(function_name(...))
```

ファンクションによって生成されたものとして、行が戻されます。

- キーワード PIPELINED を単独で指定する場合（PIPELINED IS ...）、PL/SQL ファンクションの本体は PIPE キーワードを使用する必要があります。このキーワードによってコレクション全体を 1 つの値として戻すかわりに、ファンクション外のコレクションの 1 つの要素を戻すように Oracle に指示します。
- 操作の起動、フェッチおよび停止を含むインタフェースを事前に定義する場合、PIPELINED USING *implementation_type* 句を指定できます。実装タイプは、ODCItable インタフェースを実装し、表ファンクションが作成されるときに存在する必要があります。この句は、C++ や Java などの外部言語で実装される表のファンクションに便利です。

参照：

- 表ファンクションの詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』および『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。
- Oracle Data Cartridge インタフェース（ODCI）ルーチンの詳細は、『Oracle9i Data Cartridge Developer's Guide』を参照してください。

AGGREGATE USING 句

AGGREGATE USING を指定すると、このファンクションは**集計ファンクション**、または行のグループを検証して単一の行を戻すファンクションとして識別されます。SELECT 構文のリスト、HAVING 句および ORDER BY 句で集計ファンクションを指定できます。

注意： 問合せでユーザー定義の集計ファンクションを指定する場合、**分析ファンクション**（問合せ結果セットを操作するファンクション）として使用できます。その場合は、組込み分析ファンクションを使用可能にする OVER *analytic_clause* 構文を使用します。構文およびセマンティクスの詳細は、6-8 ページの「**分析ファンクション**」を参照してください。

USING 句で、ファンクションの実装タイプの名前を指定します。実装タイプは、ODCI 集計ルーチンの実装を含むオブジェクト型である必要があります。*schema* を指定しない場合、実装タイプが自分のスキーマ内にあるとみなされます。

制限事項： この句を指定すると、ファンクションに対して入力引数を 1 つのみ指定できます。

参照： ODCI ルーチンの詳細は、『Oracle9i Data Cartridge Developer's Guide』を参照してください。

IS | AS 句

pl/sql_subprogram_body *pl/sql_subprogram_body* を使用すると、PL/SQL サブプログラム本体でファンクションを宣言できます。

参照： PL/SQL サブプログラムの詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

call_spec *call_spec* によって、Java メソッドまたは C 関数名、パラメータ・タイプおよび戻り型を SQL で相当するものにマップできます。*Java_declaration* では、'string' が JAVA 実装メソッドを定義します。

参照：

- 『Oracle9i Java Stored Procedures Developer's Guide』を参照してください。
- パラメータおよび *C_declaration* のセマンティクスの詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

AS EXTERNAL AS EXTERNAL は、C 関数を宣言するもう 1 つの方法です。この句は以前のリリースのもので、下位互換用にのみサポートされています。AS LANGUAGE C 構文を使用することをお勧めします。

例

CREATE FUNCTION の例 次の文は、デモ表 `oe.orders` にファンクション `get_bal` を作成します。

```
CREATE FUNCTION get_bal(acc_no IN NUMBER)
RETURN NUMBER
IS acc_bal NUMBER(11,2);
BEGIN
    SELECT order_total
    INTO acc_bal
    FROM orders
    WHERE customer_id = acc_no;
    RETURN(acc_bal);
END;
```

`get_bal` ファンクションを実行した場合、指定された預金口座の残高が戻ります。

このファンクションをコールする際は、残高を確認する口座番号 `acc_no` を引数として指定する必要があります。`acc_no` のデータ型は `NUMBER` です。

このファンクションでは、口座の残高が戻ります。`CREATE FUNCTION` 文の `RETURN` 句は、戻り値のデータ型が `NUMBER` であることを示しています。

このファンクションでは、`SELECT` 文によって、`order` 表の引数 `acc_no` で特定される行から `balance` 列が選択されます。また、`RETURN` 文によって、ファンクションがコールされる環境にこの値が戻ります。

この例で作成されたファンクションは、`SQL` 文の中で使用できます。たとえば、次のように指定します。

```
SELECT get_bal(165) FROM DUAL;
```

```
GET_BAL(165)
-----
          2519
```

次の文では、C ルーチン `c_get_val` を外部ファンクションとして登録する PL/SQL スタンダードアロン・ファンクション `get_val` が作成されます（この例では、パラメータは省略されています）。

```
CREATE FUNCTION get_val
( x_val IN NUMBER,
  y_val IN NUMBER,
  image IN LONG RAW )
RETURN BINARY_INTEGER AS LANGUAGE C
  NAME "c_get_val"
  LIBRARY c_utils
  PARAMETERS (...);
```

集計ファンクションの例 次の文は、数値を集計する `SecondMax` という名前の集計ファンクションを作成します。オブジェクト型 `SecondMaxImpl` ルーチンが `ODCIAggregate` ルーチンの実装を含んでいるものとします。

```
CREATE FUNCTION SecondMax (input NUMBER) RETURN NUMBER
  PARALLEL_ENABLE AGGREGATE USING SecondMaxImpl;
```

参照： `SecondMaxImpl` の型および型本体の実装の詳細は、『Oracle9i Data Cartridge Developer's Guide』を参照してください。

デモ表 `hr.employees` を問い合わせる次の文のように、問合せで集計ファンクションを使用します。

```
SELECT SecondMax(salary), department_id
  FROM employees
 GROUP BY department_id
 HAVING SecondMax(salary) > 9000;
```

```
SECONDMAX(SALARY) DEPARTMENT_ID
-----
13500             80
17000             90
```

ファンクションでのパッケージ・プロシージャの使用例 次の文は、`DBMS_LOB` プロシージャを使用し、CLOB 列の長さを戻すファンクションを作成します。

```
CREATE OR REPLACE FUNCTION text_length(a clob)
  RETURN NUMBER DETERMINISTIC IS
BEGIN
  RETURN DBMS_LOB.GETLENGTH(a);
END;
```

参照： ファンクション索引を作成するファンクションの使用方法は、12-79 ページの「[LOB 列のファンクション索引の例](#)」を参照してください。

CREATE INDEX

用途

CREATE INDEX 文は、次の索引を作成する場合に使用します。

- 表の 1 つ以上の列、パーティション表、索引構成表またはクラスタ
- 表またはクラスタの 1 つ以上のスカラー型オブジェクト属性
- ネストした表の列の索引を作成するためのネストした表の記憶表

索引は、スキーマ・オブジェクトの 1 つで、索引には、表またはクラスタの索引付き列の中に表示される各値のエントリが入ります。索引を使用した場合、行に直接、かつ高速にアクセスできます。Oracle は、次の索引をサポートしています。

- 通常の索引（デフォルトでは、B ツリー索引が作成されます。）
- **ビットマップ索引**。キー値に関連付けられた ROWID をビットマップとして格納します。
- **パーティション索引**。表の索引付き列に表示される各値のエントリが入るパーティションを構成します。
- **ファンクション索引**。式をベースとしています。式によって戻される値を評価する問合せを組み立てることができます。その式にはファンクション（組込みまたはユーザー定義）が含まれることがあります。
- **ドメイン索引**。アプリケーション固有の *indextype* 索引タイプのインスタンスです。

参照：

- 索引については、『Oracle9i データベース概要』を参照してください。
- 索引の変更については、8-48 ページの「[ALTER INDEX](#)」を参照してください。
- 索引の削除については、15-71 ページの「[DROP INDEX](#)」を参照してください。

前提条件

自分のスキーマ内に索引を作成する場合は、次のいずれかの条件が満たされている必要があります。

- 索引を作成する表またはクラスタが自分のスキーマ内に定義されている。
- 索引を作成する表に対する INDEX 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、UNLIMITED TABLESPACE システム権限、あるいはその索引または索引パーティションを格納するための表領域の割当て制限のいずれかが必要です。

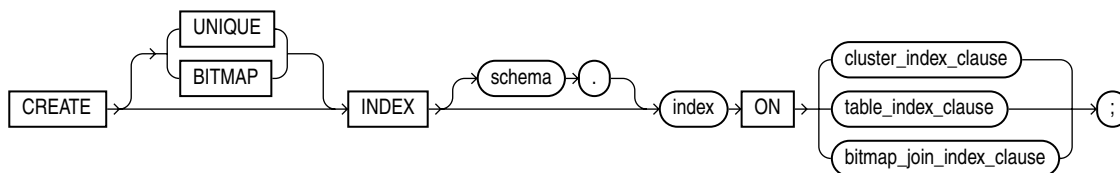
自分のスキーマにドメイン索引を作成する場合、従来索引の作成の前提条件の他に、索引タイプについての EXECUTE 権限が必要です。他のユーザーのスキーマにドメイン索引を作成する場合、索引の所有者にも索引タイプおよびその基礎となる実装タイプに EXECUTE 権限が必要です。ドメイン索引を作成する前に、索引タイプを定義する必要があります。

自表の自分のスキーマにファンクション索引を作成する場合、従来索引の作成の前提条件の他に、QUERY REWRITE システム権限が必要です。別のスキーマまたは別のスキーマの表に索引を作成する場合は、GLOBAL QUERY REWRITE 権限が必要です。どちらの場合も、表の所有者は、ファンクション索引で使用されるファンクションについての EXECUTE オブジェクト権限が必要です。また、問合せでファンクション索引を使用する場合は、QUERY_REWRITE_ENABLED パラメータを TRUE に、QUERY_REWRITE_INTEGRITY パラメータを TRUSTED に設定する必要があります。

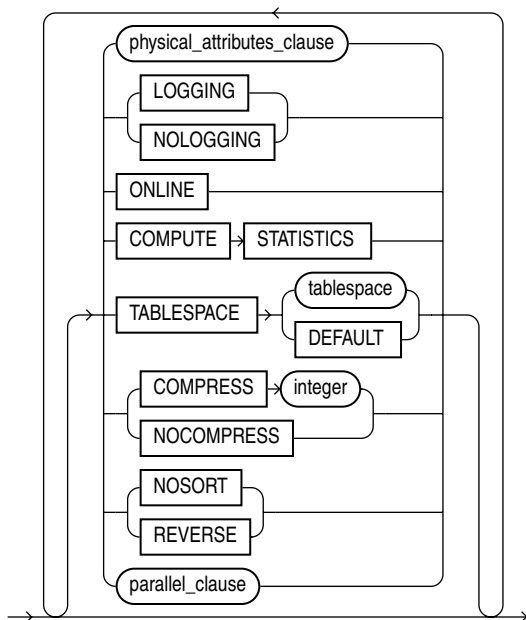
参照： 12-84 ページの「[CREATE INDEXTYPE](#)」を参照してください。

構文

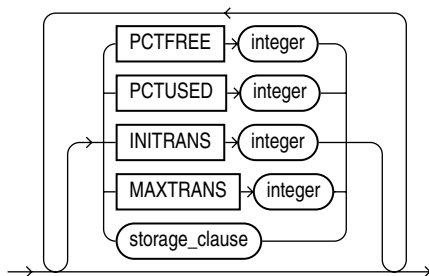
create_index::=



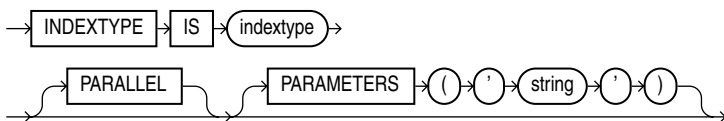
index_attributes::=



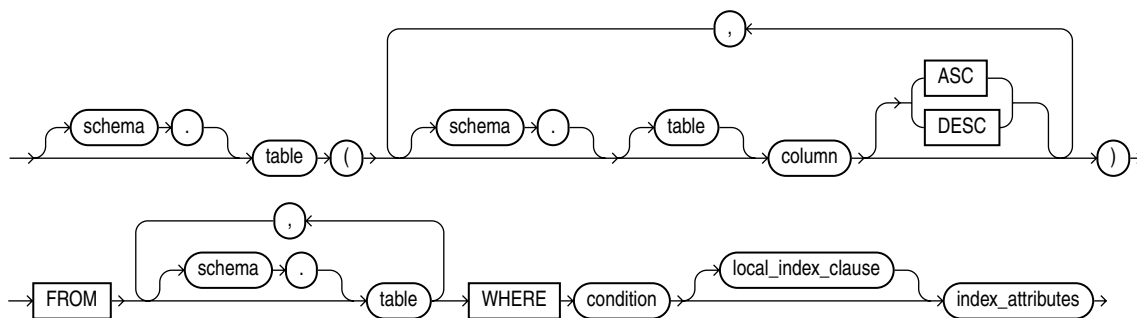
physical_attributes_clause::=



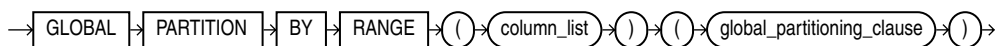
domain_index_clause::=



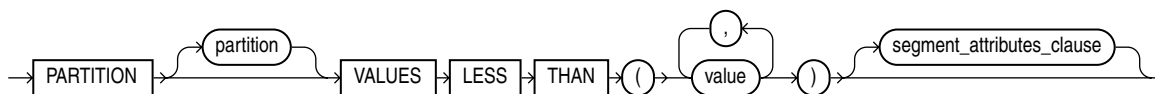
bitmap_join_index_clause::=



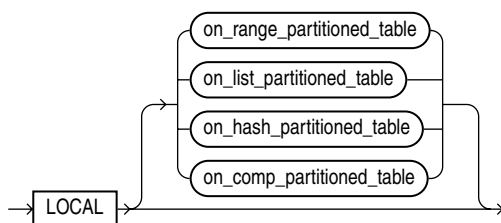
global_partitioned_index::=



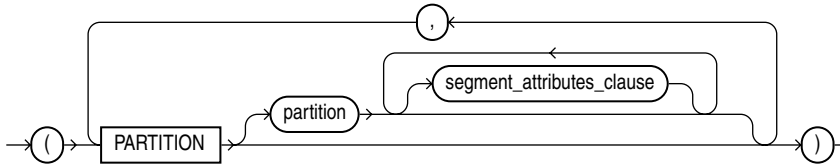
global_partitioning_clause::=



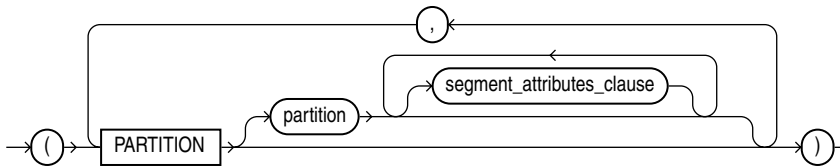
local_partitioned_index::=



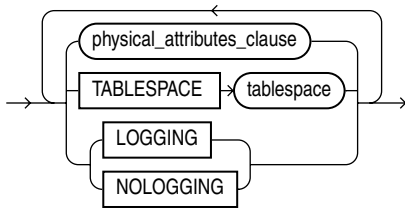
on_range_partitioned_table::=



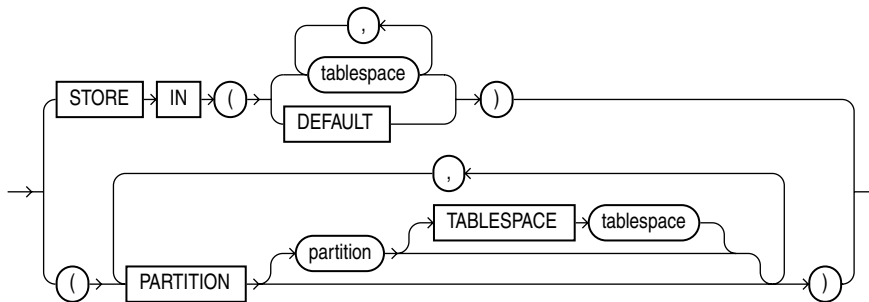
on_list_partitioned_table::=



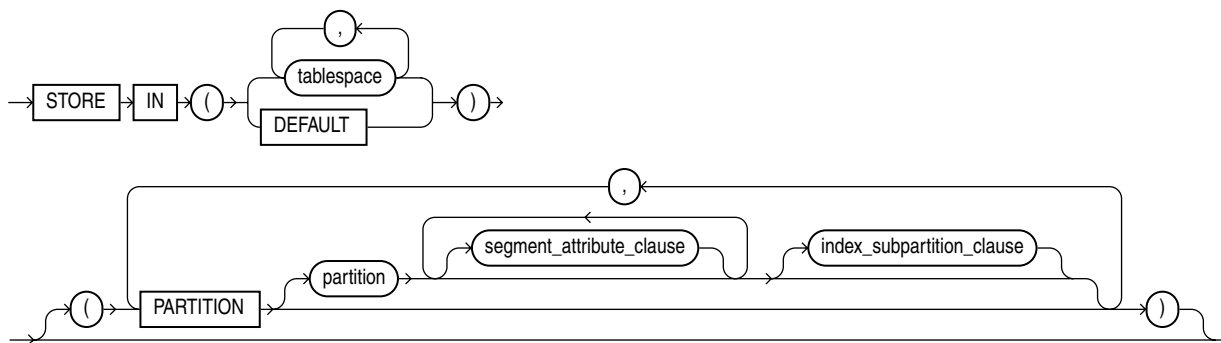
segment_attributes_clause::=



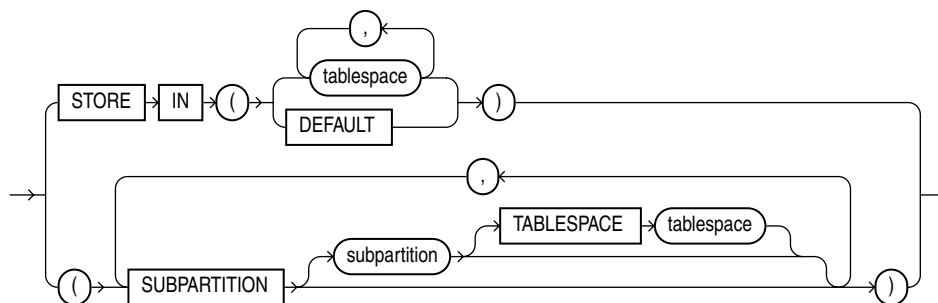
on_hash_partitioned_table::=



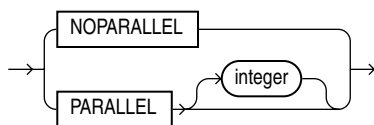
on_comp_partitioned_table::=



index_subpartition_clause::=



parallel_clause::=



storage_clause: 17-50 ページの「[storage_clause](#)」を参照してください。

キーワードとパラメータ

UNIQUE

UNIQUE によって、索引のベースとなっている列の値が一意である必要があることを指定します。索引がローカル非同一キー索引（次の説明を参照）の場合、索引キーはパーティション・キーを含んでいる必要があります。

制限事項：

- UNIQUE および BITMAP は同時に指定できません。
- ドメイン索引には UNIQUE を指定できません。

参照： 整合性制約の詳細は、11-72 ページの「[constraint_clause](#)」を参照してください。

BITMAP

BITMAP によって、*index* が各行を分割した索引付けではなく、各個別キーのビットマップで作成されることを指定します。ビットマップ索引では、キー値にビットマップとして関連付けられた ROWID が格納されます。ビットマップ内の各ビットは使用可能な ROWID に対応しているため、ビットが設定されていれば、それに対応する ROWID を持つ行に、キー値が設定されていることとなります。ビットマップの内部表現は、データ・ウェアハウスなど、低レベルの同時実行トランザクションが実行されるアプリケーションに最適です。

制限事項：

- グローバル・パーティション索引の作成には BITMAP を指定できません。
- 索引構成表がビットマップ化したセカンダリ索引に対応するマッピング表を持たない場合、索引構成表にビットマップ化したセカンダリ索引を作成できません。
- UNIQUE および BITMAP は同時に指定できません。
- ドメイン索引には BITMAP を指定できません。

参照：

- ビットマップ索引の使用の詳細は、『Oracle9i データベース概要』および『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。
- マッピング表の詳細は、14-6 ページの「[CREATE TABLE](#)」を参照してください。

schema

作成する索引が定義されるスキーマを指定します。*schema* を指定しないと、自分のスキーマ内に索引が作成されます。

index

作成する索引の名前を指定します。

cluster_index_clause

cluster_index_clause を使用して、クラスタ索引が作成されるクラスタを識別します。*cluster* を *schema* で修飾しない場合、そのクラスタが自分のスキーマ内に定義されているとみなされます。ハッシュ・クラスタにはクラスタ索引を作成できません。

参照： 12-2 ページの「[CREATE CLUSTER](#)」を参照してください。

table_index_clause

索引を定義している表（およびその属性）を指定します。指定する *table* を *schema* で修飾しない場合は、その表が自分のスキーマに定義されているとみなされます。

ネストした表の記憶表に索引を作成することによって、ネストした表の列に索引を作成します。記憶表の `NESTED_TABLE_ID` 疑似列を組み込んだ一意索引を作成することは、ネストした表の値を持つ行がそれぞれ確実に異なるようにする有効な手段です。

制限事項：

- ローカル・パーティション索引の場合は、*table* をパーティション化する必要があります。
- 索引構成表の場合、この文はセカンダリ索引を作成します。このセカンダリ索引には、`REVERSE` を指定できません。索引キーおよび論理 `ROWID` の結合サイズは、ブロック・サイズの半分未満にする必要があります。
- *table* が一時表の場合、索引も *table* と同様の有効範囲（セッションまたはトランザクション）を持つ一時的なものとなります。一時表の索引には、次の制限があります。
 - 索引は、パーティション索引またはドメイン索引であってははいけません。
 - *physical_attributes_clause* または *parallel_clause* は指定できません。
 - `LOGGING`、`NOLOGGING` または `TABLESPACE` は指定できません。

参照： 一時表の詳細は、14-6 ページの「[CREATE TABLE](#)」および『Oracle9i データベース概要』を参照してください。

t_alias

索引を作成する表に対して相関名（別名）を指定します。

注意： *index_expr* がオブジェクト型属性またはオブジェクト型メソッドを参照する場合、この別名が必要になります。12-80 ページの「[Type Method のファンクション索引の例](#)」を参照してください。

index_expr

index_expr は、索引に基づく列または列の式を指定します。

column 表内の列の名前を指定します。ビットマップ索引には最大 30 列まで指定できません。他の索引には最大 32 列まで指定できます。

スカラー・オブジェクト属性列またはネストした表の記憶表のシステム定義の NESTED_TABLE_ID 列には索引を作成できます。オブジェクト属性列を指定する場合、列名を表名で修飾する必要があります。ネストした表の列属性を指定する場合は、この属性は、一番外側の表の名前、ネストした表が定義されている列の名前、およびそのネストした表の列属性となるすべての中間属性の名前で修飾する必要があります。

制限事項： SCOPE 句で定義されている REF 型の列または属性の索引が Oracle でサポートされている場合を除き、ユーザー定義型、LONG、LONG RAW、LOB または REF 型の列または属性に索引を作成できません。

column_expression *table* の列、定数、SQL ファンクションおよびユーザー定義ファンクションの列から作成された式を指定します。*column_expression* を指定した場合、**ファンクション索引**が作成されます。

参照： 12-68 ページの「[ファンクション索引における注意事項](#)」および 12-69 ページの「[ファンクション索引に関する制限事項](#)」を参照してください。

ファンクションの名前解決は、索引作成者のスキーマに基づきます。*column_expression* で使用されるユーザー定義ファンクションは、CREATE INDEX 操作中に完全に名前解決されます。

ファンクション索引の作成後、ANALYZE 文を使用して索引とその基になる表の両方に関する統計項目を収集します。この統計項目が生成されるまで、ファンクション索引は使用できません。

参照： 11-31 ページの「[ANALYZE](#)」を参照してください。

ファンクション索引における注意事項

ファンクション索引を使用する表を問い合わせる場合、その問合せで `column_expression` が NULL でないことを確認する必要があります。ただし、WHERE 句に指定した列の順序が、ファンクション索引を定義した `column_expression` での順序と異なる場合でも、問合せにファンクション索引が使用されます。

参照： 12-79 ページの「[ファンクション索引の例](#)」を参照してください。

索引の基になるファンクションが無効または削除された場合は、索引に DISABLED がマークが付けられます。オプティマイザが索引の使用を選択した場合、DISABLED 索引の問合せは失敗します。索引が UNUSABLE にマークが付けられ、パラメータ SKIP_UNUSABLE_INDEXES が true に設定された場合を除き、DISABLED 索引の DML 操作は失敗します。

参照： このパラメータの詳細は、9-2 ページの「[ALTER SESSION](#)」を参照してください。

ファンクション索引の使用も、QUERY_REWRITE_ENABLED セッション・パラメータの設定による影響を受けます。

参照： 9-2 ページの「[ALTER SESSION](#)」を参照してください。

ファンクション、パッケージまたは型のパブリック・シノニムが `column_expression` で使用され、後で同じ名前の実際のオブジェクトが表の所有者のスキーマに作成された場合、ファンクション索引は使用禁止になります。その後、ALTER INDEX ... ENABLE または ALTER INDEX ... REBUILD を使用してファンクション索引を使用可能にした場合、`column_expression` で使用されているファンクション、パッケージまたは型は、パブリック・シノニムが最初に指定されたファンクション、パッケージまたは型への変換を続けます。新しいファンクション、パッケージまたは型への変換は行われません。

ファンクション索引の定義によって文字データに内部変換が生成される場合に、NLS パラメータの設定を変更するときは注意が必要です。ファンクション索引は、NLS パラメータの現行のデータベース設定を使用します。セッション・レベルでパラメータを再設定した場合、ファンクション索引を使用して問合せを行うと、無効な結果が戻る場合があります。2 つの照合パラメータ (NLS_SORT および NLS_COMP) は例外です。これらがセッション・レベルで再設定された場合でも、Oracle は正常に変換を処理します。

ファンクション索引に関する制限事項

- `column_expression` で参照されるユーザー定義ファンクションは、DETERMINISTIC である必要があります。
- グローバル・パーティション・ファンクション索引では、`column_expression` がパーティション・キーであってははいけません。
- `column_expression` には、スカラー副問合せ式を除く式のすべての形式が可能です。
- パラメータがない場合も、すべてのファンクションをカッコで指定する必要があります。カッコで指定していない場合は、列名として解析されます。
- `column_expression` で指定するファンクションは、リピータブル値を戻す必要があります。たとえば、SYSDATE や USER ファンクションまたは ROWNUM 疑似列は指定できません。
- `column_expression` に集計ファンクションを含めることはできません。
- ネストした表にはファンクション索引を作成できません。

参照： 12-47 ページの「[CREATE FUNCTION](#)」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。

ASC | DESC

ASC または DESC を使用して、索引を昇順で作成するか降順で作成するかを指定します。文字データの索引は、データベース・キャラクタ・セットの文字値の昇順または降順で作成されます。

Oracle は、降順索引をファンクション索引として扱います。索引作成に QUERY REWRITE または GLOBAL QUERY REWRITE は必要ありません。ただし、他のファンクション索引のように、最初に索引および索引が定義されている表を分析するまで、降順索引は使用しません。この文の `column_expression` 句を参照してください。

制限事項： ドメイン索引にはこれらの句を指定できません。逆順索引には DESC を指定できません。`index` をビットマップ化したり、COMPATIBLE 初期化パラメータに 8.1.0 未満の値を設定すると、DESC は無視されます。

index_attributes

physical_attributes_clause *physical_attributes_clause* を使用して、索引における物理特性および記憶特性の値を設定します。14-6 ページの「**CREATE TABLE**」を参照してください。

制限事項：

- PCTUSED パラメータは、索引に対して指定できません。
- PCTFREE には、索引の各データ・ブロック内で、更新および挿入に備えて確保しておく領域の割合（パーセント）を指定します。
- *storage_clause* を使用して、索引における記憶特性を指定します。

参照： 17-50 ページの「*storage_clause*」を参照してください。

TABLESPACE *tablespace* には、索引、索引パーティションまたは索引サブパーティションを格納する表領域の名前を指定します。この句を指定しない場合、その索引を定義しているスキーマの所有者のデフォルトの表領域内に索引が作成されます。

ローカル索引の場合、*tablespace* のかわりにキーワード **DEFAULT** を指定できます。ローカル索引に追加される新規パーティションまたはサブパーティションは、基礎となる表の対応するパーティションまたはサブパーティションと同じ表領域内に作成されます。

COMPRESS **COMPRESS** を指定すると、キー圧縮が使用可能になります。これによって、キー列値の繰返しがなくなり、記憶領域を削減できます。*integer* を使用して、接頭辞の長さ（圧縮する接頭辞列数）を指定します。

- 一意索引の場合、接頭辞の長さの有効範囲は、1 ～（キー列の数 - 1）です。デフォルトの接頭辞の長さは、（キー列の数 - 1）です。
- 一意でない索引の場合、接頭辞の長さの有効範囲は、1 ～キー列の数です。デフォルトの接頭辞の長さはキー列数です。

非パーティション索引（一意でない索引または 2 列以上の一意索引）のみを圧縮します。

制限事項： ビットマップ索引には、**COMPRESS** を指定できません。

NOCOMPRESS **NOCOMPRESS** を指定すると、キー圧縮が使用禁止になります。これはデフォルトです。

NOSORT **NOSORT** を指定すると、行がデータベース内に昇順で格納されます。そのため、Oracle は索引の作成時に行のソートを行う必要がありません。索引列の行または列が昇順に格納されていない場合、Oracle はエラーを戻します。ソート時間および領域を削減するため、列を表へ初期ロードした直後にこの句を使用します。

NOSORT の制限事項

- この句は、REVERSE と同時には指定できません。
- この句を使用して、クラスタ索引、パーティション索引またはビットマップ索引を作成することはできません。
- 索引構成表のセカンダリ索引には、この句を指定できません。

REVERSE REVERSE を指定すると、ROWID 以外の索引ブロックのバイトが逆順に格納されます。

REVERSE の制限事項

- この句は、NOSORT と同時には指定できません。
- ビットマップ索引または索引構成表は逆順には格納できません。

LOGGING | NOLOGGING 索引作成を、REDO ログ・ファイル内に記録する (LOGGING) か記録しない (NOLOGGING) かを指定します。この設定によって、索引に対する後続のダイレクト・ローダー (SQL*Loader) およびダイレクト・パス INSERT 操作を記録するか記録しないかも決定します。デフォルトは LOGGING です。

非パーティション索引の場合、この句は索引のロギング属性を指定します。

パーティション索引の場合、この句は次のことを決定します。

- CREATE 文で指定されたすべてのパーティションのデフォルト値 (PARTITION 記述句で LOGGING または NOLOGGING を指定する場合を除く)
- 索引パーティションに関連付けられたセグメントに対するデフォルト値
- 後続の ALTER TABLE ... ADD PARTITION 操作中に明示的に追加されたローカル索引パーティションまたはサブパーティションに対するデフォルト値

NOLOGGING モードでは、データの変更時に、(新しいエクステン트를 INVALID としてマーク設定し、ディクショナリの変更を記録するために) 最小限のログが記録されます。メディア・リカバリ中に NOLOGGING が適用された場合、REDO データのログへの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、ブロック範囲に「論理的に無効」というマークが付きます。このため、損失してはならない索引がある場合は、NOLOGGING 操作の後にバックアップを取ってください。

データベースを ARCHIVELOG モードで運用する場合、LOGGING 操作の前に取ったバックアップからのメディア・リカバリによって、索引が再作成されます。ただし、NOLOGGING 操作の前に取ったバックアップからのメディア・リカバリでは、索引は再作成されません。

索引のロギング属性は、その実表の属性に依存しません。

この句を指定しない場合、ロギング属性は表が存在する表領域の属性になります。

参照： ロギングおよびパラレル DML の詳細は、『Oracle9i データベース概要』および『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

ONLINE ONLINE によって、索引作成中に表での DML 操作ができることを指定します。

制限事項：

- オンラインで索引を作成中は、パラレル DML はサポートされません。ONLINE を指定し、パラレル DML 文を発行すると、Oracle はエラーを戻します。
- ビットマップ索引またはクラスタ索引には、ONLINE を指定できません。
- UROWID 列の従来索引には、ONLINE を指定できません。

参照： オンラインでの索引作成および再作成については、『Oracle9i データベース概要』を参照してください。

COMPUTE STATISTICS COMPUTE STATISTICS によって、索引作成中の統計情報収集を指定します。これらの統計情報は、SQL 文の実行計画を選択する際に、オブティマイザによって使用中のデータ・ディクショナリに格納されます。

収集された統計情報のタイプは、作成する索引のタイプによって異なります。

注意： 表のかわりに別の索引を使用して索引を作成する場合、元の索引は適切な統計情報を提供しない場合があります。このため、一般的に基となる表を使用して統計を計算します。その結果、統計は改善されますが、パフォーマンスが低下することがあります。

PL/SQL パッケージおよびプロシージャでは、その他の方法で統計を収集できます。

参照： 『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

parallel_clause

索引の作成をパラレル化する場合は、*parallel_clause* を指定します。

注意： *parallel_clause* 構文は、以前のリリースの構文に代わるものです。以前のリリースの構文は下位互換用にサポートされていますが、動作がわずかに異なることがあります。

NOPARALLEL シリアル実行を行う場合は、**NOPARALLEL** を指定します。これはデフォルトです。

PARALLEL 並列度を選択する場合は、**PARALLEL** を指定します。並列度とは、すべての関係するインスタンスで使用可能な CPU の数に、初期化パラメータ **PARALLEL_THREADS_PER_CPU** の値を掛けたものです。

PARALLEL integer *integer* には、パラレル操作で使用するパラレル・スレッド数である**並列度**を指定します。各パラレル・スレッドは、1、2 個のパラレル実行サーバーを使用します。通常、最適な並列度が計算されるため、*integer* に値を指定する必要はありません。

Index Partitioning 句

global_partitioned_index 句および *local_partitioned_index* 句を使用すると、*index* をパーティション化できます。

注意： ブロック・サイズが異なる表領域のパーティション化されたデータベース・エンティティの記憶域には、制限事項があります。これらの制限事項の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。

global_partitioned_index

global_partitioned_index によって、索引のパーティション化がユーザー定義であり、基礎となる表と同一レベルでパーティション化されないことを指定できます。デフォルトでは、非パーティション索引はグローバル索引です。

PARTITION BY RANGE **PARTITION BY RANGE** によって、*column_list* で指定された列の値の範囲でグローバル索引がパーティション化されるように指定します。ローカル索引にこの句は指定できません。

column_list *column_list* には、索引のパーティション化を行う表の列名を指定します。*column_list* では、索引の列リストの左の接頭辞を指定する必要があります。

column_list には、最大 32 列まで指定できます。なお、ROWID 疑似列および ROWID 型の列は指定できません。

注意：異なるキャラクタ・セットを使用してデータベースを使用しているか、使用する予定がある場合は、キャラクタ列をパーティション化する際に注意してください。文字のソート順序は、すべてのキャラクタ・セットで同一ではありません。

参照：キャラクタ・セットのサポートの詳細は、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

PARTITION PARTITION 句によって、個々の索引パーティションを記述できます。句の数によってパーティションの数が決まります。*partition* を指定しない場合、名前は *SYS_Pn* の形式で生成されます。

VALUES LESS THAN *VALUES LESS THAN (value_list)* には、グローバル索引の現行のパーティションの上限（境界は含まない）を指定します。*value_list* には、*partition_by_range_clause* の *column_list* と対応するリテラル値を順番にカンマで区切って指定します。最後のパーティションの *value_list* は、必ず MAXVALUE を指定してください。

注意：*index* が DATE 列でパーティション化されている場合、および日付書式で年の最初の 2 桁の数字が指定されていない場合、年の 4 文字書式マスクで TO_DATE ファンクションを使用する必要があります。日付書式は、NLS_TERRITORY によって暗黙的に決定され、NLS_DATE_FORMAT によって明示的に決定されます。

参照：

- これらの初期化パラメータの詳細は、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。
- 14-57 ページの「レンジ・パーティション化の例」を参照してください。

local_partitioned_index

local_partitioned_index 句によって、*table* と同じパーティション数および同じパーティション境界を使用し、同じ列で索引をパーティション化できます。Oracle は、基礎となる表が再パーティション化された場合、ローカル索引のパーティションを自動的にメンテナンスします。

on_range_partitioned_table レンジ・パーティション表の索引の名前および属性を指定します。

- PARTITION には、個々のパーティションの名前を指定します。句の数によってパーティションの数が決まります。ローカル索引では、索引のパーティション数は表のパーティション数と同じで、表のパーティションと同じ順序である必要があります。
- *partition* を指定しない場合、対応する表のパーティションと整合性のある名前が生成されます。その名前が既存の索引パーティション名と競合する場合、SYS_Pn の形式が使用されます。

on_hash_partitioned_table ハッシュ・パーティション表の索引の名前および属性を指定します。*partition* を指定しない場合、明示的に指定された別の索引パーティションの名前と競合しないかぎり、対応する実表のパーティション名が使用されます。この場合、SYS_Pnnn の形式の名前を生成します。

索引パーティションまたは 1 つ以上の個々のパーティションに対して、任意に TABLESPACE を指定できます。索引またはパーティション・レベルで TABLESPACE を指定しない場合、同じ表領域の各索引パーティションを対応する表のパーティションとして格納します。

on_comp_partitioned_table コンポジット・パーティション表の索引の名前および属性を指定します。最初の STORE IN 句には、索引のサブパーティションのデフォルトの表領域を指定します。*index_subpartitioning_clause* に異なる表領域を指定して、この記憶領域をオーバーライドできます。

この句または *index_subpartitioning_clause* のサブパーティションに TABLESPACE を指定しない場合、*index* に指定された表領域を使用します。*index* に TABLESPACE を指定しない場合、同じ表領域のサブパーティションを、対応する表のサブパーティションとして格納します。

on_list_partitioned_table *on_list_partitioned_table* 句は、12-75 ページの「[on_range_partitioned_table](#)」と同様です。

STORE IN STORE IN 句によって、索引のハッシュ・パーティション（ハッシュ・パーティション索引用）または索引のサブパーティション（コンポジット・パーティション索引用）が複数の表領域に分散される方法を指定できます。表領域の数は、索引パーティションの数と等しくなる必要はありません。索引パーティションの数が表領域の数より多い場合、表領域名を介して循環します。

- DEFAULT 句は、ハッシュ・パーティション表またはコンポジット・パーティション表のローカル索引にのみ有効です。この句は、パーティションまたはサブパーティションの索引レベルで指定された表領域をオーバーライドし、同じパーティションの索引パーティションまたはサブパーティションを、対応する表パーティションまたはサブパーティションとして格納します。

- *index_subpartition_clause* によって、パーティションにおけるすべてのサブパーティションを格納する 1 つ以上の表領域、またはパーティションにおける 1 つ以上の個々のサブパーティションを指定します。サブパーティションは、パーティションからの他の属性をすべて継承します。パーティションに指定されていない属性は、索引から継承されます。

domain_index_clause

domain_index_clause を使用して、*index* がドメイン索引であることを指定します。

column 索引が定義されている表の列またはオブジェクトの属性を指定します。基礎となる索引タイプが異なり、その索引タイプがユーザー定義操作の分割セットをサポートする場合、1 つの列に複数のドメイン索引を定義できます。

制限事項： REF データ型、VARRAY、ネストした表、LONG または LONG RAW の列にはドメイン索引を作成できません。

indextype *indextype* には、索引タイプの名前を指定します。名前は、すでに定義された有効なスキーマ・オブジェクトです。

注意： Oracle Text がインストールされている場合、各種の組込み索引タイプを使用し、Oracle Text ドメイン索引を作成できます。Oracle Text および Oracle Text が使用する索引の詳細は、『Oracle Text リファレンス』を参照してください。

参照： 12-84 ページの「[CREATE INDEXTYPE](#)」を参照してください。

PARAMETERS PARAMETERS には、未解析のまま適切な ODCI 索引タイプ・ルーチンに渡されたパラメータ文字列を指定します。パラメータ文字列の最大長は 1,000 文字です。

構文の最上位でこの句を指定した場合、パラメータは索引パーティションのデフォルトのパラメータになります。LOCAL [PARTITION] 句の一部としてこの句を指定すると、個々のパーティションのデフォルトのパラメータをオーバーライドできます。

ドメイン索引が作成されると、適切な ODCI ルーチンがコールされます。ルーチンが正常に戻らない場合、ドメイン索引は FAILED のマークが付けられます。失敗したドメイン索引でサポートされる操作は、DROP INDEX および REBUILD INDEX（非ローカル索引引用）のみです。

参照： これらのルーチンについては、『Oracle9i Data Cartridge Developer's Guide』を参照してください。

domain_index_clause の制限事項

- `index_expr` は、単一列のみ指定できます。
- ビットマップ索引または一意ドメイン索引は指定できません。

bitmap_join_index_clause

`bitmap_join_index_clause` を使用すると、**ビットマップ結合索引**を定義できます。ビットマップ結合索引は、単一の表に定義します。ディメンション表の列で構成する索引キーには、そのキーに対応するファクト表の ROWID が格納されます。データ・ウェアハウス環境では、一般的に、索引を定義する表を**ファクト表**といい、ファクト表と結合した表を**ディメンション表**といいます。ただし、結合索引の作成にはスター・スキーマは必須ではありません。

ON ON 句には、まずファクト表を指定し、次に索引を定義するディメンション表の列をカッコ内に指定します。

FROM FROM 句には、結合した表を指定します。

WHERE WHERE 句には、結合条件を指定します。

基礎となるファクト表がパーティション化されている場合、`local_index_clauses` (12-74 ページの「[local_partitioned_index](#)」を参照) のいずれかを指定する必要があります。

制限事項：一般的なビットマップ索引の制限事項 (12-65 ページの「[BITMAP](#)」を参照) に加え、ビットマップ結合索引には次の制限事項があります。

- 索引構成表または一時表にビットマップ結合索引を作成できません。
- FROM 句で表を 2 回指定できません。
- ファンクション結合索引は作成できません。
- ディメンション表の列は、主キー列であるか、または一意制約を含む必要があります。
- ディメンション表が複合主キーを含む場合、主キーの各列は結合の一部である必要があります。
- ファクト表がパーティション化されていない場合は、`local_index_clauses` を指定できません。

参照： ファクト表とディメンション表、およびデータ・ウェアハウス環境でのビットマップ索引の使用については、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

例

一般的な索引の例

PARALLEL の例 次の文は、サンプル表 `oe.orders` の `customer_id` 列に `ord_customer_ix` を作成します。

```
CREATE INDEX ord_customer_ix
ON orders (customer_id);
```

COMPRESS の例 次の文は、`COMPRESS` 句を使用して索引 `ord_customer_ix` を作成します。

```
CREATE INDEX ord_customer_ix ON orders (customer_id, sales_rep_id)
    COMPRESS 1;
```

索引は、`customer_id` 列値の繰返し項目を圧縮します。

統計情報の計算例 次の文は、索引 `ord_customer_ix` の作成中に、その統計情報を収集します。

```
CREATE INDEX ord_customer_ix ON orders (customer_id, sales_rep_id)
    COMPUTE STATISTICS;
```

収集された統計のタイプは、作成する索引のタイプによって異なります。

NOLOGGING の例 サンプル表 `orders` を高速パラレル・ロードで作成した場合、次の文を発行し、早急にパラレルで索引を作成します。(適切な並列度が選択されます)。

```
CREATE INDEX ord_customer_ix
ON orders (customer_id)
NOSORT
NOLOGGING
PARALLEL;
```

クラスタ索引の例 12-9 ページの「[クラスタの作成例](#)」で作成した `personnel` クラスタに対して索引を作成するには、次の文を発行します。

```
CREATE INDEX personnel_ix ON CLUSTER personnel;
```

クラスタ・キーのすべての列に索引が自動的に作成されるため、索引列は指定しません。クラスタ索引の場合は、すべての行に索引が付きます。

NULL の例 次の文を指定するとします。

```
SELECT last_name FROM employees WHERE commission_pct IS NULL;
```

この問合せでは、ビットマップ索引でないかぎり、commission_pct 列に作成された索引は使用されません。

ファンクション索引の例

ファンクション索引の基本例 次の文は、last_name 列の大文字評価に基づく employees 表にファンクション索引を作成します。

```
CREATE INDEX upper_ix ON employees (UPPER(last_name));
```

ファンクション索引の作成に必要な権限およびパラメータ設定の詳細は、12-59 ページの「[前提条件](#)」を参照してください。

フル・テーブル・スキャンの実行ではなく、索引を使用するように、ファンクション値を後続の問合せで NULL 以外にします。たとえば、次の文は、索引の使用を保証します。

```
SELECT first_name, last_name
   FROM employees WHERE UPPER(last_name) IS NOT NULL
   ORDER BY UPPER(last_name);
```

ただし、WHERE 句が指定されていない場合は、フル・テーブル・スキャンが実行されます。

索引の作成および後続の問合せを示す次の文では、問合せで列の順序が逆であっても、Oracle は income_ix を使用します。

```
CREATE INDEX income_ix
   ON employees(salary + (salary*commission_pct));

SELECT first_name||' '||last_name "Name"
   FROM employees
   WHERE (salary*commission_pct) + salary > 15000;
```

LOB 列のファンクション索引の例 次の文は、12-57 ページの「[ファンクションでのパッケージ・プロシージャの使用例](#)」で作成したファンクションを使用し、サンプル・スキーマ pm の LOB 列にファンクション索引を作成します。例では、ファンクション索引の統計情報を収集し、CLOB 列が 1000 字未満のデモ表 print_media から行を検索します。

```
CREATE INDEX src_idx ON print_media(text_length(ad_sourcetext));
ANALYZE INDEX src_idx COMPUTE STATISTICS;

SELECT product_id FROM print_media
   WHERE text_length(ad_sourcetext) < 1000;
```

```
PRODUCT_ID
-----
      3060
      2056
      3106
      2268
```

Type Method のファンクション索引の例 この例には、2つの数値属性（lengthおよびwidth）を含むオブジェクト型 rectangle が必要です。area() メソッドは、四角形の領域を計算します。

```
CREATE TYPE rectangle AS OBJECT
( length NUMBER,
  width NUMBER,
  MEMBER FUNCTION area RETURN NUMBER DETERMINISTIC
);
```

```
CREATE OR REPLACE TYPE BODY rectangle AS
  MEMBER FUNCTION area RETURN NUMBER IS
  BEGIN
    RETURN (length*width);
  END;
END;
```

rectangle 型の表 rect_tab を作成する場合、次のように area() メソッドにファンクション索引を作成できます。

```
CREATE TABLE rect_tab OF rectangle;
CREATE INDEX area_idx ON rect_tab x (x.area());
```

この索引を使用して、効率的に次の形式の問合せを評価できます。

```
SELECT * FROM rect_tab x WHERE x.area() > 100;
```

パーティション索引の例

グローバル・パーティション索引の例 次の文は、コストの範囲を3つのグループに分割した3つのパーティションを含む表 sh_sales にグローバル同一キー索引 amount_sold を作成します。

```
CREATE INDEX cost_ix ON sales (amount_sold)
  GLOBAL PARTITION BY RANGE (amount_sold)
    (PARTITION p1 VALUES LESS THAN (1000),
     PARTITION p2 VALUES LESS THAN (2500),
     PARTITION p3 VALUES LESS THAN (MAXVALUE));
```

ハッシュ・パーティション表の索引の例 次の文は、product_information 表の product_id 列にローカル索引を作成します。STORE IN 句は、product_information がハッシュ・パーティション化されていることを示す LOCAL の直後に記述します。Oracle は、tbs1 表領域と tbs2 表領域にハッシュ・パーティションを分割します。

```
CREATE INDEX prod_idx ON product_information(product_id) LOCAL
STORE IN (tbs1, tbs2);
```

コンポジット・パーティション表の索引の例 次の文は、コンポジット・パーティション化されている composite_sales 表にローカル索引を作成します。STORAGE 句では、索引のデフォルトの記憶領域属性を指定します。ただし、別の TABLESPACE 記憶域が指定されているため、このデフォルトは、パーティション q3_2000 および q4_2000 の5つのサブパーティションにオーバーライドされます。

参照： composite_sales 表の作成の詳細は、14-59 ページの「[コンポジット・パーティション表の例](#)」を参照してください。

```
CREATE INDEX sales_idx ON composite_sales(time_id, prod_id)
STORAGE (INITIAL 1M MAXEXTENTS UNLIMITED)
LOCAL
(PARTITION q1_1998,
PARTITION q2_1998,
PARTITION q3_1998,
PARTITION q4_1998,
PARTITION q1_1999,
PARTITION q2_1999,
PARTITION q3_1999,
PARTITION q4_1999,
PARTITION q1_2000,
PARTITION q2_2000
(SUBPARTITION pq2001, SUBPARTITION pq2002,
SUBPARTITION pq2003, SUBPARTITION pq2004,
SUBPARTITION pq2005, SUBPARTITION pq2006,
SUBPARTITION pq2007, SUBPARTITION pq2008),
PARTITION q3_2000
(SUBPARTITION c1 TABLESPACE tbs1,
SUBPARTITION c2 TABLESPACE tbs2,
SUBPARTITION c3 TABLESPACE tbs3,
SUBPARTITION c4 TABLESPACE tbs4,
SUBPARTITION c5 TABLESPACE tbs5),
PARTITION q4_2000
(SUBPARTITION pq4001 TABLESPACE tbs6,
SUBPARTITION pq4002 TABLESPACE tbs7,
SUBPARTITION pq4003 TABLESPACE tbs8,
SUBPARTITION pq4004 TABLESPACE tbs9)
);
```

ビットマップ索引の例

次の例は、4つのパーティションを含む表にビットマップ・パーティション索引を作成します。

```
CREATE BITMAP INDEX partno_ix
  ON lineitem(partno)
  TABLESPACE ts1
  LOCAL (PARTITION quarter1 TABLESPACE ts2,
         PARTITION quarter2 STORAGE (INITIAL 10K NEXT 2K),
         PARTITION quarter3 TABLESPACE ts2,
         PARTITION quarter4);
```

次の文は、sh.products 表の列に基づくデモ表 sh.sales（パーティション表）にビットマップ結合索引を作成します。

```
CREATE BITMAP INDEX sales_products_ix
  ON sales (p.prod_category, p.prod_status)
  FROM sales s, products p
  WHERE s.prod_id = p.prod_id
  LOCAL;
```

sales はパーティション表であるため、索引はローカル・パーティション・ビットマップ索引である必要があります。

ネストした表の索引の例

次の例では、一意索引 uniq_proj_indx が、記憶表 nested_project_table に作成されます。疑似列 nested_table_id を組み込むことによって、ネストした表の列 projs_managed 内に固有の行が確保されます。

```
CREATE TYPE proj_type AS OBJECT
  (proj_num NUMBER, proj_name VARCHAR2(20));
CREATE TYPE proj_table_type AS TABLE OF proj_type;
CREATE TABLE employee ( emp_num NUMBER, emp_name CHAR(31),
  projs_managed proj_table_type )
  NESTED TABLE projs_managed STORE AS nested_project_table;
CREATE UNIQUE INDEX uniq_proj_indx
  ON nested_project_table ( NESTED_TABLE_ID, proj_num);
```

置換可能な列の索引の例

置換可能な列を宣言した型の属性に、索引を作成できます。また、適切な TREAT ファンクションの使用によって、サブタイプの属性を参照できます。次の例は、14-52 ページの「[置換可能な表および列のサンプル](#)」で作成した表 books を使用します。この文は、books 表の、すべての employee_t 型の author の salary 属性に索引を作成します。

```
CREATE INDEX salary_i
  ON books (TREAT(author AS employee_t).salary);
```

TREAT ファンクションの引数のターゲットとなる型は、参照する属性を追加した型である必要があります。例では、TREAT のターゲットは、employee_t で、salary 属性を追加した型です。

この条件を満たさない場合、TREAT ファンクションがファンクションの定義式として解析され、ファンクション索引が作成されます。たとえば、次の文は型の階層内の他のすべての型のインスタンスに NULL を割り当て、パートタイム従業員の salary 属性にファンクション索引を作成します。

```
CREATE INDEX salary_func_i ON persons p
  (TREAT(VALUE(P) AS part_time_emp_t).salary);
```

SYS_TYPEID ファンクションの使用によって、置換可能な列を基礎とする型判別式の列に索引を作成できます。

注意： Oracle は型判別式の列を使用し、IS OF type 条件を含む問合せを評価します。型 ID 列のカーディナリティが正常な範囲で低い場合、ビットマップ索引を作成することをお勧めします。

次の文は、books 表の author 列の型 ID にビットマップ索引を作成します。

```
CREATE BITMAP INDEX typeid_i ON books (SYS_TYPEID(author));
```

参照：

- books 表に基礎とする型の階層の作成については、15-20 ページの「[型の階層例](#)」を参照してください。
- 6-178 ページの「[TREAT](#)」を参照してください。
- 6-153 ページの「[SYS_TYPEID](#)」を参照してください。
- 5-17 ページの「[IS OF type 条件](#)」を参照してください。

CREATE INDEXTYPE

用途

CREATE INDEXTYPE 文を使用すると、(アプリケーション固有の) ドメイン索引を管理するルーチンを指定するオブジェクトである**索引タイプ**を作成できます。索引タイプは、表、ビューおよび他のスキーマ・オブジェクトと同じネームスペースにあります。この文は、索引タイプ名を実装タイプに結合し、順番に索引タイプを実装するユーザー定義索引ファンクションおよびプロシージャを指定し、参照します。

参照： 索引タイプの実装の詳細は、『Oracle9i Data Cartridge Developer’s Guide』および『Oracle9i データベース概要』を参照してください。

前提条件

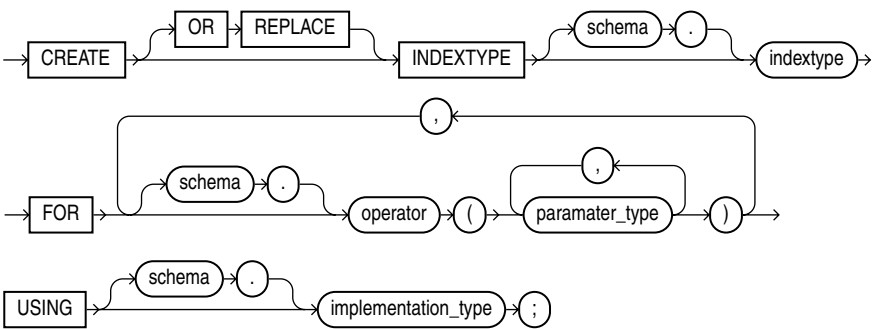
自分のスキーマに索引タイプを作成する場合は、CREATE INDEXTYPE システム権限が必要です。他のユーザーのスキーマ内に索引タイプを作成する場合は、CREATE ANY INDEXTYPE システム権限が必要です。どちらの場合も、実装タイプおよびサポートしている演算子に対する EXECUTE オブジェクト権限が必要です。

索引タイプは、1 つ以上の演算子をサポートしているため、索引タイプを作成する前に、その演算子またはサポートする演算子を設計し、これらの演算子に機能的な実装を指定します。

参照： 13-38 ページの「CREATE OPERATOR」を参照してください。

構文

create_indectype::=



キーワードとパラメータ

schema

索引タイプが存在するスキーマ名を指定します。*schema* を指定しない場合、自分のスキーマ内に索引タイプが作成されます。

indextype

作成する索引タイプの名前を指定します。

FOR 句

FOR 句によって、索引タイプにサポートされる演算子のリストを指定できます。

- *schema* には、演算子を含むスキーマを指定します。*schema* を指定しない場合、その演算子が自分のスキーマにあるとみなされます。
- *operator* には、索引タイプによってサポートされる演算子の名前を指定します。
この句に指定されるすべての演算子は有効な演算子である必要があります。
- *parameter_type* には、演算子へのパラメータ・タイプを指定します。

USING 句

USING 句によって、新しい索引タイプを実装するタイプを指定できます。

implementation_type には、ODCI を実装するタイプ名を指定します。

- ODCI でルーチンを実装する有効なタイプを指定する必要があります。
- 実装タイプは、索引タイプと同じスキーマに存在する必要があります。

参照： このインタフェースの詳細は、『Oracle9i Data Cartridge Developer's Guide』を参照してください。

例

CREATE INDEXTYPE の例 次の文は、TextIndexType という名前の索引タイプを作成し、その索引タイプでサポートされている contains 演算子、および索引インタフェースを実装する TextIndexMethods タイプを指定します。

```
CREATE INDEXTYPE TextIndexType
  FOR contains (VARCHAR2, VARCHAR2)
  USING TextIndexMethods;
```

CREATE JAVA

用途

CREATE JAVA を使用すると、Java ソース、クラスまたはリソースを含むスキーマ・オブジェクトを作成できます。

参照：

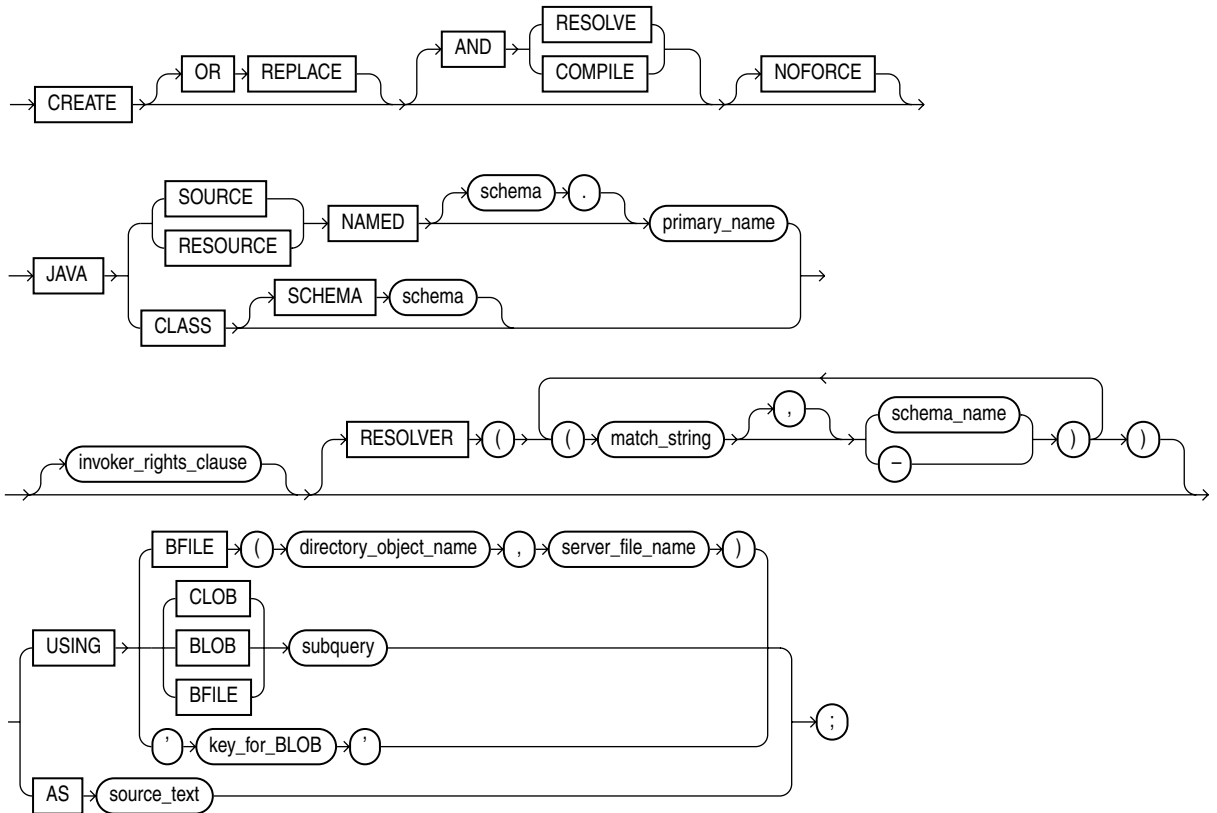
- Java の概要については、『Oracle9i Java Developer's Guide』を参照してください。
- Java ストアド・プロシージャについては、『Oracle9i Java Stored Procedures Developer's Guide』を参照してください。
- SQLJ については、『Oracle9i SQLJ 開発者ガイドおよびリファレンス』を参照してください。
- JDBC については、『Oracle9i JDBC 開発者ガイドおよびリファレンス』を参照してください。
- CORBA および EJB については、『Oracle9i Enterprise JavaBeans Developer's Guide and Reference』を参照してください。

前提条件

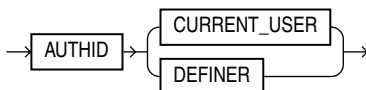
自分のスキーマに Java ソース、クラスまたはリソースを含むスキーマ・オブジェクトを作成または再作成する場合は、CREATE PROCEDURE システム権限が必要です。他のユーザーのスキーマ内にスキーマ・オブジェクトを作成する場合は、CREATE ANY PROCEDURE システム権限が必要です。他のユーザーのスキーマ内にスキーマ・オブジェクトを再作成する場合は、ALTER ANY PROCEDURE システム権限が必要です。

構文

create_java::=



invoker_rights_clause::=



キーワードとパラメータ

OR REPLACE

OR REPLACE を指定して、既存の Java クラス、ソースまたはリソースを含むスキーマ・オブジェクトを再作成します。この句を指定した場合、付与されているオブジェクト権限を削除、再作成および再付与しなくても、既存のオブジェクトの定義を変更できます。

Java スキーマ・オブジェクトを再定義し、RESOLVE または COMPILE を指定する場合、Oracle は、オブジェクトを再コンパイルまたは変換します。正常に変換またはコンパイルされたかどうかにかかわらず、Java スキーマ・オブジェクトを参照するクラスは有効になります。

再定義したファンクションに対して権限が付与されていたユーザーは、権限を再付与されなくても、そのファンクションにアクセスできます。

参照： その他の情報は、8-71 ページの「[ALTER JAVA](#)」を参照してください。

RESOLVE | COMPILE

RESOLVE および COMPILE は、同義のキーワードです。この文が正常に実行されると作成される Java スキーマ・オブジェクトを変換することを指定します。

- クラスに適用された場合、他のクラス・スキーマ・オブジェクトに対する参照名に変換されます。
- ソースに適用された場合、ソースがコンパイルされます。

制限事項：Java リソースには、この句を指定できません。

NOFORCE

RESOLVE または COMPILE を指定しても、正常に変換またはコンパイルできない場合は、NOFORCE を指定して CREATE コマンドの結果をロールバックします。このオプションを省略した場合、正常に変換またはコンパイルできない場合でも処理は行われません（作成されたスキーマ・オブジェクトが残ります）。

JAVA SOURCE 句

JAVA SOURCE を指定して、Java ソース・ファイルをロードします。

JAVA CLASS 句

JAVA CLASS を指定して、Java クラス・ファイルをロードします。

JAVA RESOURCE 句

JAVA RESOURCE を指定して、Java リソース・ファイルをロードします。

NAMED 句

NAMED 句は、Java ソースまたはリソースの場合に指定します。*primary_name* は、二重引用符で囲む必要があります。

- Java ソースの場合、この句にはソース・コードが保持されているスキーマ・オブジェクト名を指定します。正常な CREATE JAVA SOURCE 文は、ソースによって定義された Java クラスをそれぞれ保持するために、追加スキーマ・オブジェクトも作成します。
- Java リソースの場合、この句にはスキーマ・オブジェクト名を指定し、Java リソースを保持します。

primary_name の小文字、または大文字と小文字の組合せを保持するには、二重引用符を使用します。

schema を省略した場合、自分のスキーマ内にオブジェクトが作成されます。

制限事項：

- Java クラスには、NAMED を指定できません。
- *primary_name* は、データベース・リンクを含むことはできません。

SCHEMA 句

SCHEMA 句は、Java クラスにのみ適用されます。このオプションは、Java ファイルを含むオブジェクトが存在するスキーマを指定します。この句を省略した場合、自分のスキーマ内にオブジェクトが作成されます。

invoker_rights_clause

invoker_rights_clause を使用して、クラスの方法が、権限を持つそのクラスを所有するユーザーのスキーマ内で実行されるか、または権限を持つ CURRENT_USER のスキーマ内で実行されるかを指定します。

また、この句は、問合せ、DML 操作、およびその型のメンバー・ファンクションおよびプロシージャ内の動的 SQL 文の外部名の変換方法も定義します。

AUTHID CURRENT_USER

CURRENT_USER は、クラスの方法が CURRENT_USER 権限で実行されることを指定します。この句はデフォルトで、**実行者権限クラス**を作成します。

また、この句は、問合せ、DML 操作、および動的 SQL 文の外部名を CURRENT_USER のスキーマで変換することも指定します。他のすべての文における外部名は、メソッドを含むスキーマで変換します。

AUTHID DEFINER

DEFINER は、クラスが存在するスキーマの所有者権限でクラスのメソッドを実行すること、およびクラスが存在するスキーマで外部名を変換することを指定します。この句によって**定義者権限クラス**が作成されます。

参照：

- 『Oracle9i Java Stored Procedures Developer's Guide』を参照してください。
- CURRENT_USER を判断する方法については、『Oracle9i データベース概要』および『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

RESOLVER 句

次の場合は、RESOLVER 句によって、Java スキーマ・オブジェクトに対する完全修飾 Java 名のマッピングを指定します。

- *match_string* は、完全な Java 名、Java 名と一致するワイルド・カードまたは任意の名前と一致するワイルド・カードを指定します。
- *schema_name* は、対応する Java スキーマ・オブジェクトを検索するスキーマを指定します。
- *schema_name* の代替としてのダッシュ (-) は、*match_string* が有効な Java 名と一致した場合、名前が変換されないことを示します。名前の変換は成功しますが、実行時にクラスがその名前を使用することはできません。

このマッピングは、後の変換で（暗黙的に、または ALTER ... RESOLVE 文で明示的に）使用されるコマンドで作成されるスキーマ・オブジェクトの定義とともに格納されます。

USING 句

USING 句は、Java クラスまたはリソースに対する文字 (CLOB または BFILE) またはバイナリ (BLOB または BFILE) データの順序を指定します。文字の順序を使用して、1 つのファイルが Java クラスまたはリソースに、または 1 つのソース・ファイルおよび 1 つ以上の導出クラスが Java ソースに定義されます。

BFILE 句

順序を含む、オペレーティング・システム (*directory_object_name*) およびサーバー・ファイル (*server_file_name*) であらかじめ作成されているファイルのディレクトリおよびファイル名を指定します。BFILE は、通常、CREATE JAVA SOURCE によって文字順序として、CREATE JAVA CLASS または CREATE JAVA RESOURCE によってバイナリ順序として解析されます。

CLOB | BLOB | BFILE *subquery*

指定した型（CLOB、BLOB または BFILE）の行と列を選択する問合せを提供します。列の値は文字の順序を構成します

注意： USING 句は、暗黙的にキーワード SELECT を提供します。そのため、副問合せではこのキーワードを省略します。

key_for_BLOB

key_for_BLOB 句は、次の暗黙的な問合せを提供します。

```
SELECT LOB FROM CREATE$JAVA$LOB$TABLE
WHERE NAME = 'key_for_BLOB';
```

制限事項：このパラメータを使用する場合、表 CREATE\$JAVA\$LOB\$TABLE が現行のスキーマ内にあり、BLOB 型の LOB 列および VARCHAR2 型の NAME 列が存在する必要があります。

AS *source_text*

Java または SQLJ ソースの文字列を指定します。

例

Java クラスの例 次の文は、Java バイナリ・ファイルにある名前を使用して、Java クラスを含むスキーマ・オブジェクトを作成します。

```
CREATE JAVA CLASS USING BFILE (bfile_dir, 'Agent.class');
```

この例では、Java クラス Agent.class を含むオペレーティング・システム・ディレクトリに指定されるディレクトリ・オブジェクト bfile_dir がすでに存在していることが前提です。この例では、クラス名は Java クラス・スキーマ・オブジェクトの名前になります。

Java ソースの例 次の文は、Java ソース・スキーマ・オブジェクトを作成します。

```
CREATE JAVA SOURCE NAMED "Hello" AS
public class Hello {
    public static String hello() {
        return "Hello World";    } };
```

Java リソースの例 次の文は、bfile から apptext という名前の Java リソース・スキーマ・オブジェクトを作成します。

```
CREATE JAVA RESOURCE NAMED "appText"
USING BFILE (bfile_dir, 'textBundle.dat');
```

SQL 文 : CREATE LIBRARY ~ CREATE SPFILE

この章では、次の SQL 文について説明します。

- CREATE LIBRARY
- CREATE MATERIALIZED VIEW
- CREATE MATERIALIZED VIEW LOG
- CREATE OPERATOR
- CREATE OUTLINE
- CREATE PACKAGE
- CREATE PACKAGE BODY
- CREATE PFILE
- CREATE PROCEDURE
- CREATE PROFILE
- CREATE ROLE
- CREATE ROLLBACK SEGMENT
- CREATE SCHEMA
- CREATE SEQUENCE
- CREATE SPFILE

CREATE LIBRARY

用途

CREATE LIBRARY 文を使用すると、オペレーティング・システム共有ライブラリに関連するスキーマ・オブジェクトを作成できます。このスキーマ・オブジェクトの名前は、CREATE FUNCTION または CREATE PROCEDURE 文の *call_spec* で使用できます。また、パッケージまたはタイプにおけるファンクションまたはプロシージャを宣言する際にも使用できます。これによって、SQL および PL/SQL は、3GL ファンクションおよびプロシージャに対してコールできます。

参照： ファンクションおよびプロシージャの詳細は、12-47 ページの「[CREATE FUNCTION](#)」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。

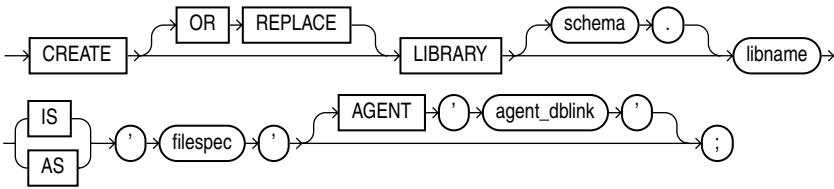
前提条件

自分のスキーマ内にライブラリを作成する場合は、CREATE LIBRARY システム権限が必要です。他のユーザーのスキーマ内にライブラリを作成する場合は、CREATE ANY LIBRARY システム権限が必要です。ライブラリに格納されているプロシージャおよびファンクションを使用する場合は、そのライブラリに対する EXECUTE オブジェクト権限が必要です。

CREATE LIBRARY 文は、共有ライブラリおよび動的リンクをサポートするプラットフォーム上でのみ有効です。

構文

create_library::=



filespec: 16-27 ページの「[filespec](#)」を参照してください。

キーワードとパラメータ

OR REPLACE

既存のライブラリを再作成する場合は、OR REPLACE を指定します。この句を指定した場合、既存のライブラリに付与されているスキーマ・オブジェクト権限を削除、再作成および再付与しなくても、ライブラリの定義を変更できます。

再定義したライブラリに対して権限を付与されていたユーザーは、権限を再付与されなくてもライブラリにアクセスできます。

libname

作成するライブラリの名前を指定します。call_spec でファンクションまたはプロシージャを宣言するときは、この名前を指定します。

'filespec'

引用符で囲まれた文字リテラルを指定します。文字列には、オペレーティング・システムの共有ライブラリの名前となるパスまたはファイル名を指定します。

'filespec' は、CREATE LIBRARY の実行中は解析されません。ライブラリ・ファイルの存在は、そのファイルからルーチンが実行されるまでチェックされません。

AGENT 句

サーバーではなく、データベース・リンクから外部プロシージャを実行する場合は、AGENT 句を指定します。外部プロシージャの実行には、agent_dblink に指定したデータベース・リンクが使用されます。この句を指定しない場合、サーバーのデフォルト・エージェント (extproc) が、外部プロシージャを実行します。

例

CREATE LIBRARY の例 次の文は、ライブラリ ext_lib を作成します。

```
CREATE LIBRARY ext_lib AS '/OR/lib/ext_lib.so';
```

次の文は、ライブラリ ext_lib を再作成します。

```
CREATE OR REPLACE LIBRARY ext_lib IS '/OR/newlib/ext_lib.so';
```

外部プロシージャ・エージェントの例 次の例は、ライブラリ `app_lib` を作成し、パブリック・データベース `sales.hq.acme.com` から外部プロシージャを起動することを指定します。

```
CREATE LIBRARY app_lib as '${ORACLE_HOME}/lib/app_lib.so'  
  AGENT 'sales.hq.acme.com';
```

参照： データベース・リンクの作成の詳細は、12-38 ページの「[パブリック・データベース・リンクの例](#)」を参照してください。

CREATE MATERIALIZED VIEW

用途

CREATE MATERIALIZED VIEW 文を使用すると、**マテリアライズド・ビュー**を作成できます。マテリアライズド・ビューは、問合せ結果を含むデータベース・オブジェクトです。問合せの FROM 句には、表、ビューおよびその他のマテリアライズド・ビューを指定できます。これらをあわせて、**マスター表**（レプリケーション用語）または**ディテール表**（データ・ウェアハウス用語）といいます。このマニュアルでは、「マスター表」を使用します。マスター表が格納されているデータベースを**マスター・データベース**といいます。

注意： キーワード SNAPSHOT は、MATERIALIZED VIEW のかわりに下位互換用にサポートされています。

レプリケーションでは、マテリアライズド・ビューを使用すると、ローカル・ノード上にあるリモート・データのコピーのメンテナンスができます。コピーは、拡張レプリケーション機能によって更新可能となりますが、この機能がない場合は読取り専用です。マテリアライズド・ビューのデータを、表またはビューと同じように選択することができます。レプリケーション環境では、通常作成されるマテリアライズド・ビューは、**主キー**、**ROWID**、**オブジェクト**および**副問合せ**のマテリアライズド・ビューです。

参照： レプリケーションをサポートするマテリアライズド・ビューの詳細は、『Oracle9i レプリケーション』を参照してください。

データ・ウェアハウスでは、通常作成されるマテリアライズド・ビューは、**マテリアライズド集計ビュー**、**単一表マテリアライズド集計ビュー**および**マテリアライズド結合ビュー**です。3つのマテリアライズド・ビューは、クエリー・リライトで使用できます。クエリー・リライトとは、マスター表に関して記述したユーザー要求を、1つ以上のマテリアライズド・ビューを含む同等の要求に変換するための最適化手法です。

参照： データ・ウェアハウスをサポートするマテリアライズド・ビューの詳細は、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

前提条件

マテリアライズド・ビューの作成に必要な権限は、ロールを介してではなく、直接付与する必要があります。

自分のスキーマ内にマテリアライズド・ビューを作成する場合は、次の条件に従う必要があります。

- CREATE MATERIALIZED VIEW システム権限と、CREATE TABLE または CREATE ANY TABLE のいずれかのシステム権限が必要です。
- 所有していないマテリアライズド・ビューのマスター表にアクセスする場合は、各表に対する SELECT オブジェクト権限または SELECT ANY TABLE システム権限が必要です。

他のユーザーのスキーマ内にマテリアライズド・ビューを作成する場合、次の条件に従う必要があります。

- CREATE ANY MATERIALIZED VIEW システム権限が必要です。
- マテリアライズド・ビューの所有者には、CREATE TABLE システム権限が必要です。スキーマ所有者が所有していないマテリアライズド・ビューの任意のマスター表、およびそのマスター表に定義された任意のマテリアライズド・ビュー・ログにアクセスするには、各表に対する SELECT オブジェクト権限、または SELECT ANY TABLE システム権限が必要です。

REFRESH ON COMMIT モードのマテリアライズド・ビューを作成する（ON COMMIT REFRESH 句を使用）場合は、前述の権限の他に、所有しないマスター表に対する ON COMMIT REFRESH オブジェクト権限、または ON COMMIT REFRESH システム権限が必要です。

前述の権限の他にも、**クエリー・リライトが使用可能**なマテリアライズド・ビューを作成する場合は、次の条件に従う必要があります。

- マスター表の所有者には、QUERY REWRITE システム権限が必要です。
- マスター表の所有者でない場合は、GLOBAL QUERY REWRITE システム権限、または自分のスキーマ以外の各表に対する QUERY REWRITE オブジェクト権限が必要です。
- スキーマ所有者がマスター表を所有していない場合は、そのスキーマ所有者には GLOBAL QUERY REWRITE 権限、または自分のスキーマ以外の各表に対する QUERY REWRITE オブジェクト権限が必要です。
- 事前作成コンテナにマテリアライズド・ビューを定義する（ON PREBUILD TABLE を使用）場合は、コンテナ表に対する WITH GRANT OPTION を指定した SELECT 権限が必要です。

マテリアライズド・ビューを含むスキーマのユーザーは、マテリアライズド・ビューのマスター表および索引を格納するターゲット表領域に十分な割当て制限を持つか、または UNLIMITED TABLESPACE システム権限を持つ必要があります。

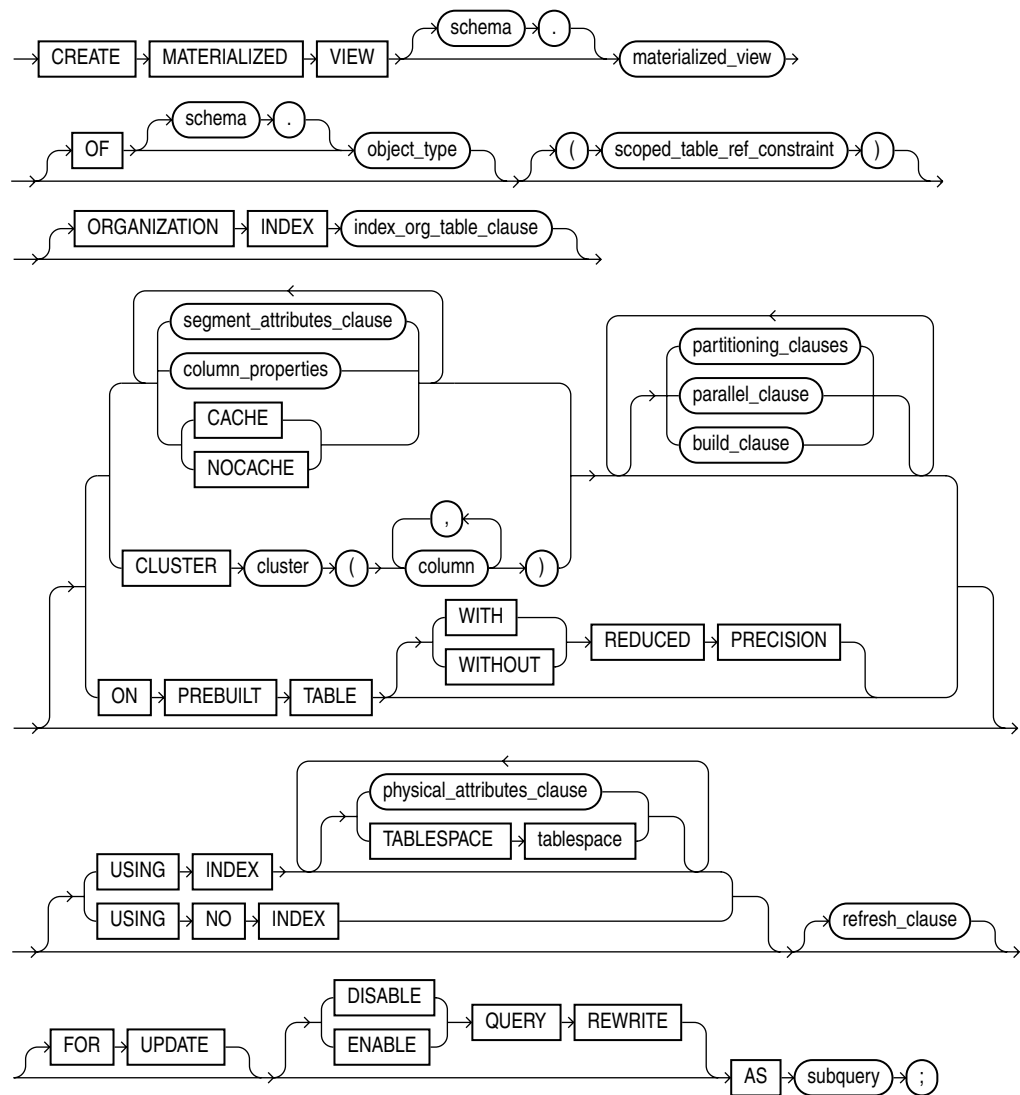
マテリアライズド・ビューを作成する場合、Oracle はマテリアライズド・ビューのスキーマ内すべてに、1 つの内部表および 1 つ以上の索引を作成します。また、1 つのビューを作成することもあります。これらのオブジェクトは、マテリアライズド・ビューのデータをメンテナンスするために使用します。ユーザーは、これらのオブジェクトを作成するための権限が必要です。

参照：

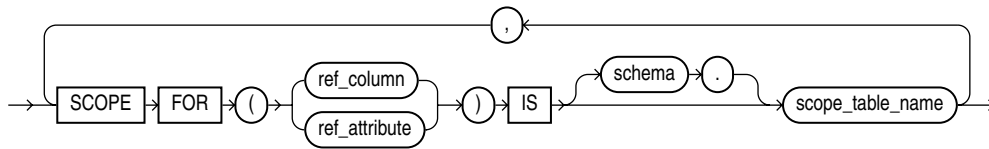
- これらの権限については、14-6 ページの「[CREATE TABLE](#)」、15-37 ページの「[CREATE VIEW](#)」、および 12-58 ページの「[CREATE INDEX](#)」を参照してください。
- レプリケーションのためのマテリアライズド・ビューの作成についての前提条件は、『Oracle9i レプリケーション』を参照してください。
- データ・ウェアハウスのためのマテリアライズド・ビューの作成についての前提条件は、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

構文

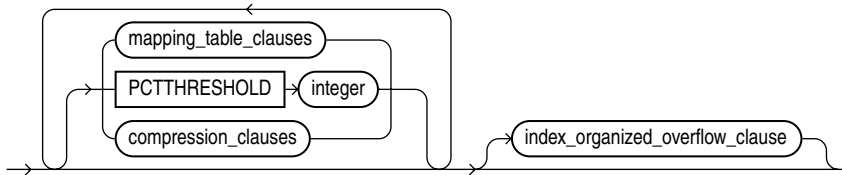
create_materialized_view::=



scoped_table_ref_constraint::=

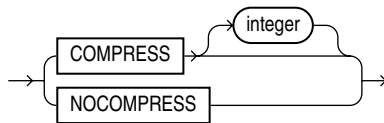


index_org_table_clause::=

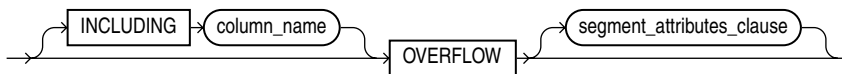


mapping_table_clauses: マテリアライズド・ビューではサポートされていません。

compression_clauses::=

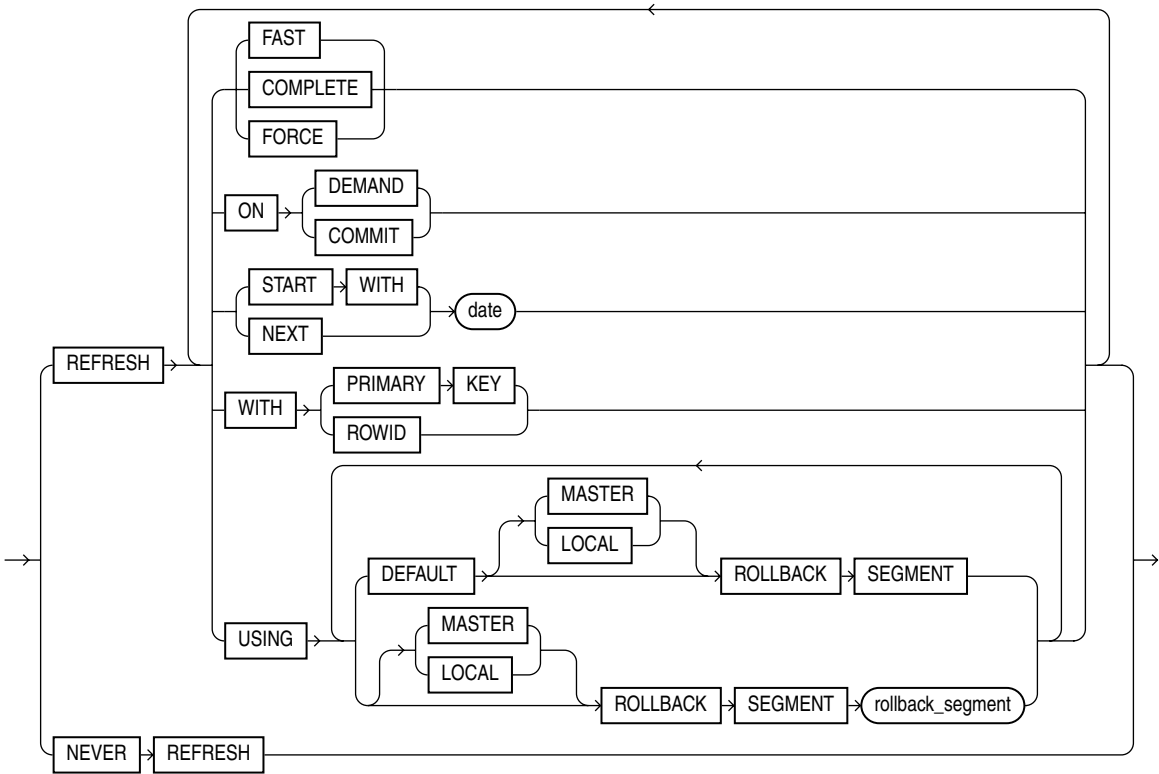


index_org_overflow_clause::=

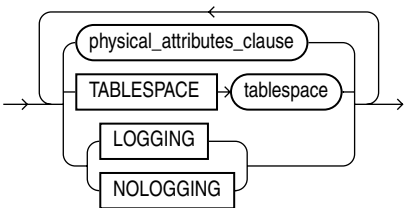


CREATE MATERIALIZED VIEW

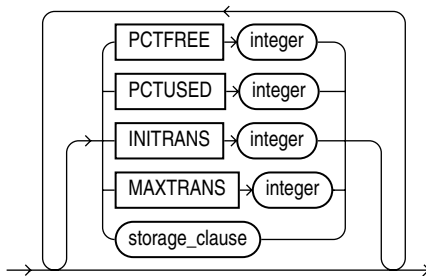
create_mv_refresh_clause::=



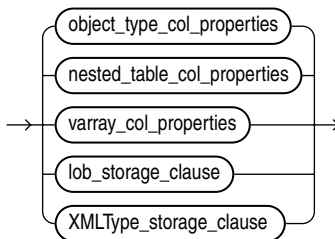
segment_attributes_clause::=



physical_attributes_clause::=

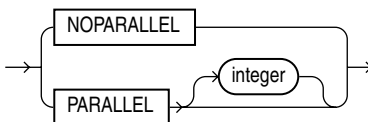


column_properties::=

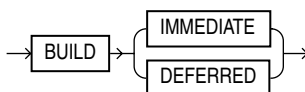


object_type_col_properties、**nested_table_col_properties**、**varray_col_properties**、**lob_storage_clause**、および **XMLType_storage_clause**: 14-6 ページの「[CREATE TABLE](#)」を参照してください。

parallel_clause::=



build_clause::=



subquery: 17-4 ページの「[SELECT](#)」を参照してください。

column_properties: 14-6 ページの「[CREATE TABLE](#)」を参照してください。

partitioning_clauses: 14-6 ページの「[CREATE TABLE](#)」を参照してください。

キーワードとパラメータ

schema

マテリアライズド・ビューを含むスキーマを指定します。*schema* を指定しないと、自分のスキーマ内にマテリアライズド・ビューが作成されます。

materialized_view

作成するマテリアライズド・ビューの名前を指定します。Oracle は、マテリアライズド・ビューを格納するための表、ビューおよび索引の名前を生成する際、マテリアライズド・ビュー名に接頭辞または接尾辞を追加します。

OF type_name

OF *type_name* 句を指定して、*object_type* 型のオブジェクト・マテリアライズド・ビューを明示的に作成します。

参照： OF *type_name* 句の詳細は、14-20 ページの「[CREATE TABLE](#)」の「[object_table](#)」を参照してください。

scoped_table_ref_constraint

SCOPE FOR 句を使用して、参照の有効範囲を 1 つの表 *scope_table_name* に制限します。REF 列または属性の値は、(REF 列と同じ型の) オブジェクト・インスタンスが格納されている *scope_table_name* 内のオブジェクトを指します。

参照： 詳細は、11-83 ページの「*constraint_clause*」の「[SCOPE 句](#)」を参照してください。

ORGANIZATION INDEX 句

ORGANIZATION INDEX を指定して、索引構成マテリアライズド・ビューを作成します。このマテリアライズド・ビューでは、データ行は、マテリアライズド・ビューの主キーに定義した索引に格納されます。次のようなマテリアライズド・ビューの索引構成を指定できません。

- 読取り専用および更新可能なオブジェクト・マテリアライズド・ビュー。マスター表に主キーが含まれている必要があります。
- 読取り専用および更新可能な主キーに基づくマテリアライズド・ビュー。
- 読取り専用 ROWID のマテリアライズド・ビュー。

index_org_table_clause のキーワードおよびパラメータは、CREATE TABLE と同じセマンティクスです。次の制限事項があります。

参照： 詳細は、14-27 ページの「CREATE TABLE」の「*index_org_table_clause*」を参照してください。

制限事項：

- CREATE MATERIALIZED VIEW 句に、CACHE または NOCACHE、CLUSTER あるいは ON PREBUILT TABLE は指定できません。
- *index_org_table_clause* には次の制限事項があります。
 - *mapping_table_clauses* を指定できません。
 - 複合主キーに基づくマテリアライズド・ビューのみに COMPRESS を指定できます。単一主キーと複合主キーのいずれかに基づくマテリアライズド・ビューに NOCOMPRESS を指定できます。

segment_attributes_clause

segment_attributes_clause を使用して、PCTFREE、PCTUSED、INITRANS および MAXTRANS パラメータの値（USING INDEX 句で使用する場合は INITRANS および MAXTRANS パラメータのみ）およびマテリアライズド・ビューの記憶特性の設定、表領域の割当て、ロギングが発生するかどうかの指定を行います。

参照：

- PCTFREE、PCTUSED、INITRANS および MAXTRANS、TABLESPACE および LOGGING|NOLOGGING パラメータの詳細は、14-6 ページの「CREATE TABLE」を参照してください。
- 記憶特性の詳細は、17-50 ページの「*storage_clause*」を参照してください。

TABLESPACE 句

マテリアライズド・ビューを作成する表領域を指定します。この句を指定しないと、マテリアライズド・ビューを含むスキーマのデフォルト表領域内にマテリアライズド・ビューが作成されます。

column_properties

column_properties 句を指定して、LOB、ネストした表、VARRAY または XMLType の列の記憶特性を指定します。

制限事項： *object_type_col_properties* は、マテリアライズド・ビューには関連しません。

参照： この句のパラメータの指定の詳細は、14-6 ページの「[CREATE TABLE](#)」を参照してください。

LOGGING | NOLOGGING

LOGGING または NOLOGGING を指定して、マテリアライズド・ビューのログギング特性を設定します。

参照： ログギング特性については、14-6 ページの「[CREATE TABLE](#)」を参照してください。

CACHE | NOCACHE

アクセス頻度の高いデータについて、CACHE を指定すると、フル・テーブル・スキャンの実行時にこの表用に取り出された各ブロックは、バッファ・キャッシュ内の最低使用頻度 (LRU) リストの最高使用頻度側に置かれます。この属性は、小規模な参照表で有効です。NOCACHE を指定すると、ブロックは LRU リストの最低使用頻度側に置かれます。

注意： NOCACHE は、*storage_clause* に KEEP を指定したマテリアライズド・ビューには、影響しません。

参照： CACHE または NOCACHE の指定の詳細は、14-6 ページの「[CREATE TABLE](#)」を参照してください。

CLUSTER 句

CLUSTER 句を使用して、指定したクラスタの一部としてマテリアライズド・ビューを作成します。クラスタ化マテリアライズド・ビューは、クラスタの領域割当てを使用します。したがって、CLUSTER 句を指定した *physical_attributes_clause* または TABLESPACE を使用しないでください。

partitioning_clauses

partitioning_clauses によって、マテリアライズド・ビューが、指定された範囲の値またはハッシュ・ファンクションでパーティション化されることを指定できます。マテリアライズド・ビューのパーティション化は、表のパーティション化と同じです

参照： 詳細は、14-36 ページの「CREATE TABLE」の「[table_partitioning_clauses](#)」を参照してください。

parallel_clause

parallel_clause によって、マテリアライズド・ビューへのパラレル操作をサポートするかどうかを指定できます。作成後にマテリアライズド・ビューに対する問合せおよび DML のデフォルトの並列度を設定します。

注意： *parallel_clause* 構文は、以前のリリースの構文に代わるものです。以前のリリースの構文は下位互換用にサポートされていますが、動作がわずかに異なることがあります。

NOPARALLEL シリアル実行を行う場合は、NOPARALLEL を指定します。これはデフォルトです。

PARALLEL 並列度を選択する場合は、PARALLEL を指定します。並列度とは、すべての関係するインスタンスで使用可能な CPU の数に、初期化パラメータ PARALLEL_THREADS_PER_CPU の値を掛けたものです。

PARALLEL integer *integer* には、パラレル操作で使用するパラレル・スレッド数である、**並列度**を指定します。各パラレル・スレッドは、1、2 個のパラレル実行サーバーを使用します。通常、最適な並列度が計算されるため、*integer* に値を指定する必要はありません。

参照： 14-43 ページの「CREATE TABLE」の「[parallel_clause](#)に関する注意事項」を参照してください。

build_clause

build_clause によって、マテリアライズド・ビューをいつ移入するかを指定できます。

IMMEDIATE IMMEDIATE を指定すると、マテリアライズド・ビューはすぐに移入されます。これはデフォルトです。

DEFERRED DEFERRED を指定すると、マテリアライズド・ビューは次の REFRESH 操作で移入されます。最初の（遅延）リフレッシュは、常に、完全リフレッシュである必要があります。それ以前のマテリアライズド・ビューの値は UNUSABLE であるため、クエリー・リライトには使用できません。

ON PREBUILT TABLE 句

ON PREBUILT TABLE 句によって、既存の表を再初期化したマテリアライズド・ビューとして登録できます。データ・ウェアハウス環境において、大きいマテリアライズド・ビューを登録する場合に有効です。その表は、結果マテリアライズド・ビューと同じ名前で、同じスキーマにある必要があります。

マテリアライズド・ビューが削除されると、その既存の表は、1 つの表としての元の形に戻ります。

注意： この句は、表オブジェクトが副問合せの具体化を反映することを前提としています。マテリアライズド・ビューがそのマスター表のデータを正しく反映することを保証するために、この前提が満たされていることを確認することをお勧めします。

WITH REDUCED PRECISION 表またはマテリアライズド・ビュー列の精度が、副問合せで戻される精度と一致しない場合の精度の低下を認める場合は、WITH REDUCED PRECISION を指定します。

WITHOUT REDUCED PRECISION 表またはマテリアライズド・ビュー列の精度を副問合せによって戻された精度と一致させるか、または操作を中断するには、WITHOUT REDUCED PRECISION を指定します。これはデフォルトです。

制限事項：

- *subquery* の各列の別名は、*table_name* の列に対応し、対応する列のデータ型が一致している必要があります。
- この句を指定する場合、非管理列（*subquery* で参照されない列）にデフォルト値も指定しないかぎり、その列に NOT NULL 制約は指定できません。

USING INDEX 句

USING INDEX 句を使用して、マテリアライズド・ビューのデータをメンテナンスするために使用されるデフォルトの索引の INITTRANS パラメータ、MAXTRANS パラメータおよび STORAGE パラメータの値を変更します。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用されます。マテリアライズド・ビューの増分リフレッシュ（高速リフレッシュ）を高速に処理するには、デフォルトの索引が使用されます。

制限事項：この句では、PCTUSED または PCTFREE パラメータは指定できません。

USING NO INDEX 句

USING NO INDEX 句を指定して、デフォルトの索引の作成を抑制します。CREATE INDEX 文を使用することによって、代替する索引を明示的に作成できます。USING NO INDEX を指定し、増分リフレッシュ（REFRESH FAST）でマテリアライズド・ビューを作成する場合は、このような索引を作成する必要があります。

create_mv_refresh_clause

create_mv_refresh_clause を使用して、Oracle がマテリアライズド・ビューをリフレッシュするデフォルトの方法、モードおよび時刻を指定できます。マテリアライズド・ビューのマスター表が変更された場合、マテリアライズド・ビューのデータを更新し、その時点でマスター表にあるデータを正確に反映させる必要があります。この句によって、自動的にマテリアライズド・ビューをリフレッシュする時刻をスケジューリングし、リフレッシュの方法およびモードを指定できます。

注意： この句では、デフォルトのリフレッシュ・オプションのみを設定します。リフレッシュを実際に実装する手順は、『Oracle9i レプリケーション』および『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

FAST 句

FAST によって、増分リフレッシュ方法を指定します。これはマスター表に対して行った変更に従ってリフレッシュを行います。この変更は、マスター表に関連付けられたマテリアライズド・ビュー・ログ（従来型 DML 変更の場合）またはダイレクト・ローダー・ログ（ダイレクト・パス INSERT 操作の場合）に格納されます。

REFRESH FAST を指定すると、マテリアライズド・ビューのマスター表のマテリアライズド・ビュー・ログが存在していない場合に、CREATE 文は正常に実行されません。（ダイレクト・パス INSERT が行われると、ダイレクト・ローダー・ログが自動的に作成されます。ユーザーによる手動操作は必要ありません。）

従来型 DML の変更の場合も、ダイレクト・パス INSERT 操作の場合も、他の条件によって、高速リフレッシュへのマテリアライズド・ビューの適応性が制限されることがあります。

定義する副問合せに分析ファンクションが含まれている場合、マテリアライズド・ビューは高速リフレッシュに適応しません。

参照：

- レプリケーション環境における高速リフレッシュの制限については、『Oracle9i レプリケーション』を参照してください。
- データ・ウェアハウス環境における高速リフレッシュの制限については、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。
- 6-8 ページの「[分析ファンクション](#)」を参照してください。

COMPLETE 句

COMPLETE によって、完全リフレッシュ方法を指定します。これは、マテリアライズド・ビューを定義する副問合せを実行することによって実装されます。完全リフレッシュを要求すると、高速リフレッシュが実行可能であっても、完全リフレッシュが実行されます。

FORCE 句

FORCE によって、リフレッシュ時、高速リフレッシュが可能な場合は高速リフレッシュを実行し、そうでない場合は完全リフレッシュを実行することを指定します。リフレッシュ方法（FAST、COMPLETE または FORCE）を指定しないと、デフォルトで FORCE が指定されます。

ON COMMIT 句

ON COMMIT によって、マテリアライズド・ビューのマスター表に対するトランザクションをコミットするときは、必ず高速リフレッシュが実行されるように指定します。この句を指定すると、リフレッシュ操作がコミット処理の一部として行われるため、コミットの完了に時間がかかります。

制限事項：この句は、オブジェクト型を含むマテリアライズド・ビューについてはサポートしません。

参照：『Oracle9i レプリケーション』および『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

ON DEMAND 句

ON DEMAND によって、マテリアライズド・ビューが、3 つの DBMS_MVIEW プロシージャのいずれかのコールによる要求でリフレッシュされることを指定します。ON COMMIT および ON DEMAND のどちらも指定しなかった場合、ON DEMAND がデフォルトになります。

参照：

- これらのプロシージャの詳細は、『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。
- REFRESH ON DEMAND を指定することによって作成できるマテリアライズド・ビューのタイプについては、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

ON COMMIT または ON DEMAND を指定した場合、START WITH または NEXT を同時に指定できません。

START WITH 句

最初の自動リフレッシュ時刻を表す日付式を指定します。

NEXT 句

自動リフレッシュの間隔を計算するための日付式を指定します。

START WITH 値および NEXT 値は、将来の時刻に評価される値です。START WITH 値を省略した場合、Oracle はマテリアライズド・ビューの作成時に NEXT 式を評価することによって、最初の自動リフレッシュ時刻を判断します。START WITH 値を指定し、NEXT 値を指定しない場合、Oracle は 1 回のみマテリアライズド・ビューをリフレッシュします。START WITH 値および NEXT 値のどちらも指定しない場合、または `create_mv_refresh_clause` を指定しない場合は、マテリアライズド・ビューは自動リフレッシュされません。

WITH PRIMARY KEY 句

WITH PRIMARY KEY によって、主キー・マテリアライズド・ビューを作成します。これはデフォルトであり、WITH ROWID に記述される値を除き、すべての場合に使用される必要があります。主キー・マテリアライズド・ビューによって、マテリアライズド・ビューのマスター表は、高速リフレッシュに対するマテリアライズド・ビューの適応性に影響されずに再編成されます。マスター表には、使用可能な主キー制約が定義されている必要があります。

制限事項：この句は、オブジェクト・マテリアライズド・ビューには指定できません。WITH OBJECT ID を指定してマテリアライズされたオブジェクトは暗黙的にリフレッシュされます。

参照：主キー・マテリアライズド・ビューの詳細は、『Oracle9i レプリケーション』を参照してください。

WITH ROWID 句

WITH ROWID によって、ROWID マテリアライズド・ビューを作成します。ROWID のマテリアライズド・ビューは、リリース 8.0 より前のマスター表と互換性があります。

マテリアライズド・ビューがマスター表の主キー列をすべて含まない場合、ROWID マテリアライズド・ビューを使用することができます。ROWID マテリアライズド・ビューは、単一リモート表を元に行っている必要があり、次のいずれも含むことができません。

- 固有ファンクションまたは集計ファンクション
- GROUP BY 句または CONNECT BY 句
- 副問合せ
- 結合
- 集合演算

完全リフレッシュが行われるまでは、マスター表の再編成後に、ROWID マテリアライズド・ビューは高速リフレッシュされません。

制限事項：この句は、オブジェクト・マテリアライズド・ビューには指定できません。WITH OBJECT ID を指定してマテリアライズドされたオブジェクトは暗黙的にリフレッシュされます。

USING ROLLBACK SEGMENT 句

マテリアライズド・ビューのリフレッシュ中に使用するリモート・ロールバック・セグメントを指定します。使用するロールバック・セグメント名を `rollback_segment` に指定します。

DEFAULT DEFAULT は、使用するロールバック・セグメントが自動的に選択されることを指定します。DEFAULT を指定した場合、`rollback_segment` は指定できません。

DEFAULT は、マテリアライズド・ビューを変更（作成ではなく）する場合に有効です。

参照： 8-74 ページの「[ALTER MATERIALIZED VIEW](#)」を参照してください。

MASTER MASTER は、個々のマテリアライズド・ビュー用のリモート・マスター・サイトで使用されるリモート・ロールバック・セグメントを指定します。

LOCAL LOCAL は、マテリアライズド・ビューが含まれているローカル・リフレッシュ・グループで使用されるリモート・ロールバック・セグメントを指定します。

参照： DBMS_REFRESH パッケージを使用するローカル・マテリアライズド・ビュー・ロールバック・セグメントの指定については、『Oracle9i レプリケーション』を参照してください。

MASTER または LOCAL を指定しない場合、デフォルトで LOCAL が使用されます。
rollback_segment を指定しない場合、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。

1 つのマスター・ロールバック・セグメントは、各マテリアライズド・ビューに格納され、マテリアライズド・ビューの作成およびリフレッシュ時にスキャンされます。複合マテリアライズド・ビューの場合、マスター・ロールバック・セグメントの指定は無視されます。

NEVER REFRESH 句

NEVER REFRESH を指定して、Oracle のリフレッシュ・メカニズムまたはパッケージ・プロシージャによるマテリアライズド・ビューのリフレッシュを行わないようにします。このようなプロシージャから発行されるマテリアライズド・ビューの REFRESH 文は、無視されません。この句を戻すには、ALTER MATERIALIZED VIEW ... REFRESH 文を発行する必要があります。

FOR UPDATE 句

FOR UPDATE によって、副問合せ、主キー、オブジェクトまたは ROWID マテリアライズド・ビューを更新します。アドバンスド・レプリケーションと組み合わせて使用した場合、更新はマスターにも影響します。

参照：『Oracle9i レプリケーション』を参照してください。

QUERY REWRITE 句

QUERY REWRITE 句によって、マテリアライズド・ビューがクエリー・リライトで使用できるかどうかを指定します。

ENABLE 句

ENABLE を指定して、クエリー・リライトでマテリアライズド・ビューを使用可能にします。

制限事項：

- マテリアライズド・ビューのすべてのユーザー定義ファンクションが DETERMINISTIC である場合のみ、クエリー・リライトを使用可能にできません。
- 文内の式が反復可能な場合のみ、クエリー・リライトを使用可能にできます。たとえば、CURRENT_TIME、USER、順序値（たとえば、CURRVAL または NEXTVAL 疑似列）、または SAMPLE 句（マテリアライズド・ビューの変更内容と異なる行をサンプルとして抽出します）を含めることはできません。

注意：

- クエリー・リライトでのマテリアライズド・ビューの使用は、デフォルトでは無効です。この句を指定して、マテリアライズド・ビューをクエリー・リライトに適用できるようにする必要があります。
 - マテリアライズド・ビューを作成した後は、必ず分析してください。クエリー・リライトを最適化するために、**ANALYZE** 操作で生成した統計が必要です。
-

参照：

- クエリー・リライトの詳細は、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。
- 12-47 ページの「**CREATE FUNCTION**」を参照してください。

DISABLE 句

DISABLE を指定して、マテリアライズド・ビューをクエリー・リライトで使用禁止にします。ただし、使用禁止にされたマテリアライズド・ビューをリフレッシュすることはできません。

AS subquery

定義するマテリアライズド・ビューの副問合せを指定します。マテリアライズド・ビューの作成時に、この副問合せが実行され、実行結果がマテリアライズド・ビューに格納されます。この副問合せは、有効な SQL 副問合せである必要があります。ただし、すべての問合せが高速リフレッシュされるわけではなく、すべての副問合せがクエリー・リライトに使用できるわけでもありません。

マテリアライズド・ビューの副問合せの定義についての注意

- BUILD DEFERRED を指定する場合、すぐには副問合せの定義は実行されません。
- マテリアライズド・ビュー副問合せを定義する FROM 句に指定する各表およびビューを、それを定義しているスキーマで修飾することをお勧めします。

参照： その他の注意事項については、14-49 ページの「**CREATE TABLE**」の「**AS subquery**」を参照してください。

マテリアライズド・ビューの副問合せの定義についての制限事項

- ユーザー SYS が所有する表、ビューまたはマテリアライズド・ビューからマテリアライズド・ビューの副問合せの定義を選択できますが、このようなマテリアライズド・ビューに対して QUERY REWRITE は使用できません。
- GROUP BY 句を含むマテリアライズド結合ビューおよびマテリアライズド集計ビューは、索引構成表からは選択できません。
- マテリアライズド・ビューに LONG データ型の列を含めることはできません。
- 一時表に対しては、マテリアライズド・ビュー・ログを作成できません。そのため、副問合せの定義が一時表を参照する場合、マテリアライズド・ビューは高速リフレッシュに適応せず、同じ文で QUERY REWRITE を指定することもできません。
- 副問合せを定義する FROM 句が他のマテリアライズド・ビューを参照する場合、この文で作成するマテリアライズド・ビューのリフレッシュの前に、副問合せの定義が参照するマテリアライズド・ビューを常にリフレッシュしておく必要があります。

クエリー・リライトが可能なマテリアライズド・ビューを作成する場合、次の制限があります。

- 副問合せには、ROWNUM、USER、SYSDATE、リモート表、順序、あるいはデータベースまたはパッケージ状態の書込み、読込みを行う PL/SQL ファンクションに対する（直接的またはビューを介しての）参照を含めることはできません。
- マテリアライズド・ビューおよびマテリアライズド・ビューのマスター表はリモートにできません。

マテリアライズド・ビュー・ログを使用した高速リフレッシュを実行する場合は、いくつかの制限事項があります。

参照：

- データ・ウェアハウスに関する制限事項は、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。
- レプリケーションに関する制限事項は、『Oracle9i レプリケーション』を参照してください。

例

次の例は、13-5 ページの「[CREATE MATERIALIZED VIEW](#)」の「例」で作成したマテリアライズド・ログを必須とします。

マテリアライズド集計ビューの例 次の文は、サンプル表 `sh.sales` にマテリアライズド集計ビューを作成して移入し、デフォルトのリフレッシュ方法、モードおよび時刻を指定します。

```
CREATE MATERIALIZED VIEW sales_mv
  BUILD IMMEDIATE
  REFRESH FAST ON COMMIT
  AS SELECT t.calendar_year, p.prod_id,
    SUM(s.amount_sold) AS sum_sales
  FROM times t, products p, sales s
  WHERE t.time_id = s.time_id AND p.prod_id = s.prod_id
  GROUP BY t.calendar_year, p.prod_id;
```

次の文は、サンプル・スキーマ `sh` の表を使用して、マテリアライズド集計ビュー `sales_by_month_by_state` を作成して移入します。マテリアライズド・ビューは、この文が正しく実行されるとすぐに移入されます。デフォルトで、次のマテリアライズド・ビューの間合せを再実行することによって、後続のリフレッシュが行われます。

```
CREATE MATERIALIZED VIEW sales_by_month_by_state
  TABLESPACE demo
  PARALLEL 4
  BUILD IMMEDIATE
  REFRESH COMPLETE
  ENABLE QUERY REWRITE
  AS SELECT t.calendar_month_desc, c.cust_state_province,
    SUM(s.amount_sold) AS sum_sales
  FROM times t, sales s, customers c
  WHERE s.time_id = t.time_id AND s.cust_id = c.cust_id
  GROUP BY t.calendar_month_desc, c.cust_state_province;
```

事前作成したマテリアライズド・ビューの例 次の文は、既存の集計表 `sales_sum_table` に対するマテリアライズド集計ビューを作成します。

```
CREATE TABLE sales_sum_table
  (month DATE, state VARCHAR2(25), sales NUMBER);

CREATE MATERIALIZED VIEW sales_sum_table
  ON PREBUILT TABLE
  ENABLE QUERY REWRITE
```



```

AS SELECT t.month, g.state, SUM(f.sales) AS sum_sales
FROM fact f, time t, geog g
WHERE f.cur_date = t.cur_date AND f.city_id = g.city_id
GROUP BY month, state;

```

この例では、マテリアライズド・ビューは事前作成表と同じ名前を持ち、かつ事前作成表と同じデータ型で同じ数の列を持ちます。

マテリアライズド結合ビューの例 次の文は、マテリアライズド結合ビュー mjv を作成します。

```

CREATE MATERIALIZED VIEW mjv
REFRESH FAST
AS SELECT l.rowid as l_rid, l.pk, l.ofk, l.c1, l.c2,
       o.rowid as o_rid, o.pk, o.cfk, o.c1, o.c2,
       c.rowid as c_rid, c.pd, c.c1, c.c2
FROM l, o, c
WHERE l.ofk = o.pk(+) AND o.ofk = c.pk(+);

```

副問合せマテリアライズド・ビューの例 次の文は、リモート・データベースの sales スキーマの orders および customers 表を基にした副問合せマテリアライズド・ビューを作成します。

```

CREATE MATERIALIZED VIEW sales.orders FOR UPDATE
AS SELECT * FROM sales.orders@dbsl.acme.com o
WHERE EXISTS
  (SELECT * FROM sales.customers@dbsl.acme.com c
   WHERE o.c_id = c.c_id);

```

主キー例 次の文は、主キー・マテリアライズド・ビュー human_genome を作成します。

```

CREATE MATERIALIZED VIEW human_genome
REFRESH FAST START WITH SYSDATE NEXT SYSDATE + 1/4096
WITH PRIMARY KEY
AS SELECT * FROM genome_catalog;

```

ROWID の例 次の文は、ROWID マテリアライズド・ビューを作成します。

```

CREATE MATERIALIZED VIEW emp_data REFRESH WITH ROWID
AS SELECT * FROM emp_table73;

```

定期的リフレッシュの例 次の文は、主キー・マテリアライズド・ビュー emp_sf を作成し、ニューヨークの scott が所有する従業員表のデータを移入します。

```
CREATE MATERIALIZED VIEW emp_sf
  PCTFREE 5 PCTUSED 60
  TABLESPACE users
  STORAGE (INITIAL 50K NEXT 50K)
  REFRESH FAST NEXT sysdate + 7
  AS SELECT * FROM scott.emp@ny;
```

この文では、START WITH パラメータが指定されていないため、現行の SYSDATE を使用して NEXT 値が評価され、最初の自動リフレッシュ時刻が判断されます。現在、ニューヨークの従業員表にマテリアライズド・ビュー・ログが存在する場合、マテリアライズド・ビュー作成の 7 日後から、7 日ごとにマテリアライズド・ビューの高速リフレッシュが実行されます。

マテリアライズド・ビューが高速リフレッシュを行うための条件と一致するため、高速リフレッシュが行われます。また、前述の文では、Oracle がマテリアライズド・ビューをメンテナンスするために使用する表の記憶特性も設定しています。

自動リフレッシュ時刻の例 次の文は、ダラスおよびボルティモアの従業員表を問い合わせる複合マテリアライズド・ビュー all_emps を作成します。

```
CREATE MATERIALIZED VIEW all_emps
  PCTFREE 5 PCTUSED 60
  TABLESPACE users
  STORAGE (INITIAL 50K NEXT 50K)
  USING INDEX STORAGE (INITIAL 25K NEXT 25K)
  REFRESH START WITH ROUND(SYSDATE + 1) + 11/24
  NEXT NEXT_DAY(TRUNC(SYSDATE), 'MONDAY')) + 15/24
  AS SELECT * FROM fran.emp@dallas
      UNION
      SELECT * FROM marco.emp@balt;
```

このマテリアライズド・ビューは、翌日の午前 11:00 に自動的にリフレッシュされ、その後は、毎週月曜日の午後 3:00 にリフレッシュされます。デフォルトのリフレッシュ方法は、FORCE です。all_emps には UNION 演算子が含まれますが、UNION 演算子は高速リフレッシュでサポートされていないため、自動的に完全リフレッシュが実行されます。

前述の文では、マテリアライズド・ビューおよびこのマテリアライズド・ビューをメンテナンスするために使用される索引の記憶特性を次のように設定しています。

- 最初に、マテリアライズド・ビューの 1 番目と 2 番目のエクステンツのサイズをそれぞれ 50KB に設定します。
- 次の USING INDEX 句付きの指定では、索引の 1 番目と 2 番目のエクステンツのサイズをそれぞれ 25KB に変更します。

ロールバック・セグメントの例 次の文では、リモート・マスターにあるロールバック・セグメント master_seg およびマテリアライズド・ビューを含むローカル・リフレッシュ・グループに対するロールバック・セグメント snap_seg を使用して、主キー・マテリアライズド・ビュー sales_emp を作成します。

```
CREATE MATERIALIZED VIEW sales_emp
  REFRESH FAST START WITH SYSDATE NEXT SYSDATE + 7
  USING MASTER ROLLBACK SEGMENT master_seg
  LOCAL ROLLBACK SEGMENT snap_seg
  AS SELECT * FROM bar;
```

次の文には誤りがあり、DEFAULT ロールバック・セグメントを使用してセグメント名を指定しているため、エラーが生成されます。

```
CREATE MATERIALIZED VIEW order_mv
  REFRESH FAST START WITH SYSDATE NEXT SYSDATE + 7
  USING DEFAULT ROLLBACK SEGMENT mv_seg
  AS SELECT * FROM orders;
using default rollback segment mv_seg
  *
```

ERROR at line 3:
ORA-00905: キーワードがありません。

副問合せおよび UNION を伴う高速リフレッシュが可能なマテリアライズド・ビューの例

次の文は、product_information および inventories 表から WHERE 条件で戻される行を制限する UNION 集合演算子を使用して、デモ・スキーマ oe の表 order_items から列を選択する高速リフレッシュが可能なマテリアライズド・ビューを作成します。

```
CREATE MATERIALIZED VIEW warranty_orders REFRESH FAST AS
  SELECT order_id, line_item_id, product_id FROM order_items o
  WHERE EXISTS
    (SELECT * FROM inventories i WHERE o.product_id = i.product_id
     AND i.quantity_on_hand IS NOT NULL)
  UNION
  SELECT order_id, line_item_id, product_id FROM order_items
  WHERE quantity > 5;
```

このマテリアライズド・ビューには、order_items (product_id を結合列とする) および inventories (quantity_on_hand をフィルタ列とする) で定義されたマテリアライズド・ビュー・ログが必須です。13-36 ページの「フィルタ列の例」および「結合列の例」を参照してください。

ネステッド・マテリアライズド・ビューの例 次の例は、前述の例のマテリアライズド・ビューをマスター表として使用し、デモ・スキーマ oe の特定の販売員で構成されるマテリアライズド・ビューを作成します。

```
CREATE MATERIALIZED VIEW my_warranty_orders
  AS SELECT w.order_id, w.line_item_id, o.order_date
  FROM warranty_orders w, orders o
  WHERE o.order_id = w.order_id
  AND o.sales_rep_id = 165;
```

CREATE MATERIALIZED VIEW LOG

用途

CREATE MATERIALIZED VIEW LOG 文を使用すると、マテリアライズド・ビューのマスター表に関連する表**マテリアライズド・ビュー・ログ**を作成できます。

注意： キーワード SNAPSHOT は、MATERIALIZED VIEW のかわりに下位互換用にサポートされています。

マスター表のデータで DML が変更された場合、Oracle は変更を記述する行をマテリアライズド・ビュー・ログに格納し、そのマテリアライズド・ビュー・ログを使用して、マスター表を基にしたマテリアライズド・ビューをリフレッシュします。このプロセスを、増分リフレッシュまたは**高速リフレッシュ**といいます。マテリアライズド・ビュー・ログがない場合、マテリアライズド・ビュー問合せが再実行され、マテリアライズド・ビューがリフレッシュされます。このプロセスが**完全リフレッシュ**です。通常、高速リフレッシュは、完全リフレッシュよりも高速に処理されます。

マテリアライズド・ビュー・ログは、マスター表と同一のスキーマ内のマスター・データベースにあります。マスター表には、単一マテリアライズド・ビュー・ログのみ必要です。Oracle では、このマテリアライズド・ビュー・ログを使用して、マスター表に基づく高速リフレッシュが可能なすべてのマテリアライズド・ビューに対して、高速リフレッシュを実行します。

マテリアライズド結合ビュー（結合を含むマテリアライズド・ビュー）を高速リフレッシュする場合、マテリアライズド・ビューが参照するそれぞれのマスター表に対するマテリアライズド・ビュー・ログを作成する必要があります。

参照：

- マテリアライズド・ビューの概要については、13-5 ページの「[CREATE MATERIALIZED VIEW](#)」、『Oracle9i データベース概要』、『Oracle9i データ・ウェアハウス・ガイド』および『Oracle9i レプリケーション』を参照してください。
- マテリアライズド・ビュー・ログの変更については、8-90 ページの「[ALTER MATERIALIZED VIEW LOG](#)」を参照してください。
- マテリアライズド・ビュー・ログの削除については、15-80 ページの「[DROP MATERIALIZED VIEW LOG](#)」を参照してください。
- ダイレクト・ローダー・ログの使用については、『Oracle9i データベース概要』および『Oracle9i データベース・ユーティリティ』を参照してください。

前提条件

マテリアライズド・ビュー・ログを作成するために必要な権限は、マテリアライズド・ビュー・ログに関連付けられた基礎となるオブジェクトの作成に必要な権限と直接関連しています。

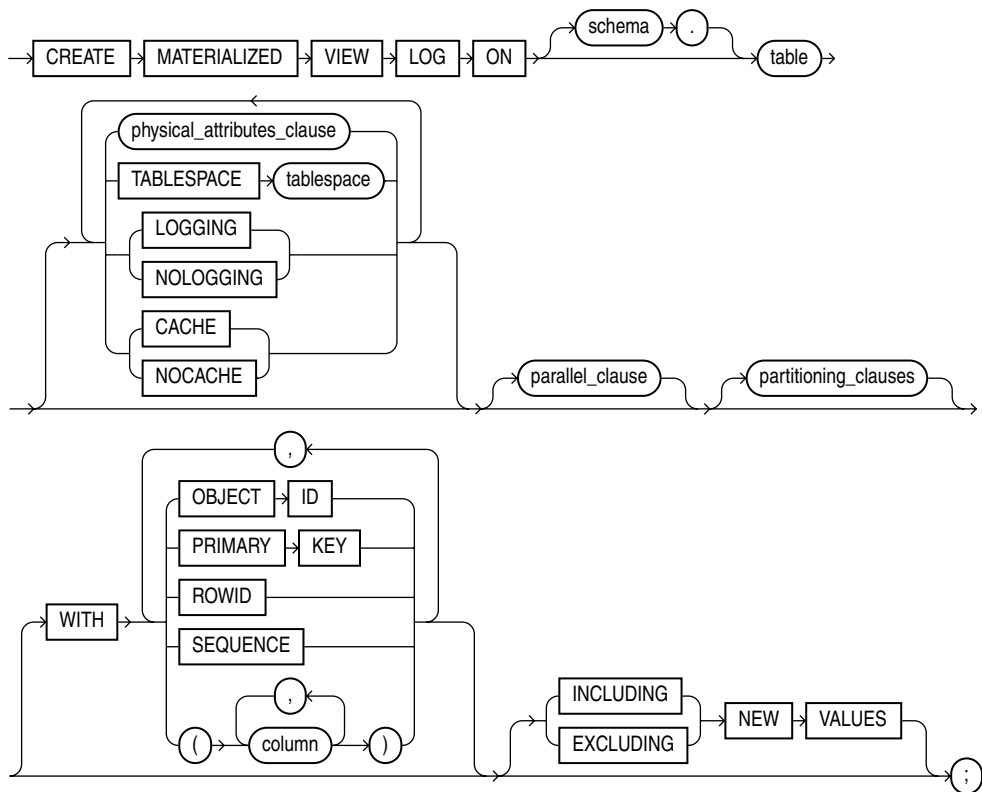
- マスター表を所有しているユーザーに CREATE TABLE 権限がある場合、関連するマテリアライズド・ビュー・ログを作成できます。
- 他のユーザーのスキーマ内に表のマテリアライズド・ビュー・ログを作成する場合は、CREATE ANY TABLE 権限、COMMENT ANY TABLE 権限、およびマスター表に対する SELECT 権限または SELECT ANY TABLE 権限が必要です。

どちらの場合も、マテリアライズド・ビュー・ログの所有者には、マテリアライズド・ビュー・ログを格納するための表領域に十分な割当て制限または UNLIMITED TABLESPACE システム権限が必要です。

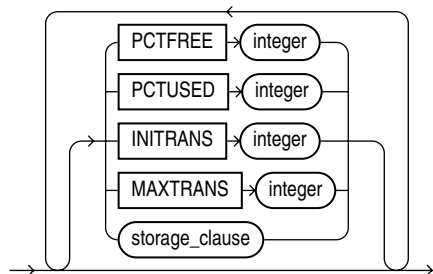
参照： マテリアライズド・ビュー・ログを作成する場合の前提条件については、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

構文

create_materialized_vw_log::=

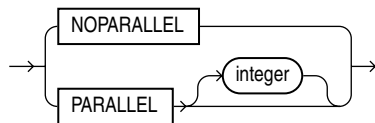


physical_attributes_clause::=



storage_clause: 17-50 ページの「[storage_clause](#)」を参照してください。

parallel_clause::=



partitioning_clauses: 14-6 ページの「[CREATE TABLE](#)」を参照してください。

キーワードとパラメータ

schema

マテリアライズド・ビュー・ログのマスター表が定義されているスキーマを指定します。
schema を省略した場合、マスター表は自分のスキーマ内に定義されているものとみなされます。マテリアライズド・ビュー・ログは、マスター表のスキーマ内に作成されます。なお、ユーザー SYS のスキーマ内の表に対しては、マテリアライズド・ビュー・ログを作成できません。

table

マテリアライズド・ビュー・ログを作成するマスター表の名前を指定します。

制限事項：一時表またはビューに対しては、マテリアライズド・ビュー・ログを作成できません。

physical_attributes_clause

physical_attributes_clause を使用して、マテリアライズド・ビュー・ログにおける物理特性および記憶特性の値の設定をします。

参照： 14-6 ページの「[CREATE TABLE](#)」および 17-50 ページの「[storage_clause](#)」を参照してください。

TABLESPACE 句

マテリアライズド・ビュー・ログを作成する表領域を指定します。この句を省略した場合、マテリアライズド・ビュー・ログのスキーマのデフォルト表領域内にマテリアライズド・ビュー・ログが作成されます。

LOGGING | NOLOGGING

LOGGING または NOLOGGING を指定して、マテリアライズド・ビュー・ログのロギング特性を設定します。デフォルトは、マテリアライズド・ビュー・ログが存在する表領域のロギング特性です。

参照： ロギング特性については、14-6 ページの「[CREATE TABLE](#)」を参照してください。

CACHE | NOCACHE

アクセス頻度の高いデータについて、CACHE を指定すると、フル・テーブル・スキャンの実行時にこのログ用に取り出された各ブロックは、バッファ・キャッシュ内の最低使用頻度 (LRU) リストの最高使用頻度側に置かれます。この属性は、小規模な参照表で有効です。

NOCACHE を指定すると、ブロックは LRU リストの最低使用頻度側に置かれます。デフォルトは NOCACHE です。

注意： NOCACHE は、*storage_clause* に KEEP を指定したマテリアライズド・ビュー・ログには、影響しません。

参照： CACHE または NOCACHE の指定の詳細は、14-6 ページの「[CREATE TABLE](#)」を参照してください。

parallel_clause

parallel_clause によって、パラレル操作でマテリアライズド・ビュー・ログをサポートするかどうかを指定できます。

注意： *parallel_clause* 構文は、以前のリリースの構文に代わるものです。以前のリリースの構文は下位互換用にサポートされていますが、動作がわずかに異なることがあります。

NOPARALLEL シリアル実行を行う場合は、NOPARALLEL を指定します。これはデフォルトです。

PARALLEL 並列度を選択する場合は、PARALLEL を指定します。並列度とは、すべての関係するインスタンスで使用可能な CPU の数に、初期化パラメータ PARALLEL_THREADS_PER_CPU の値を掛けたものです。

PARALLEL *integer* *integer* には、パラレル操作で使用するパラレル・スレッド数である、**並列度**を指定します。各パラレル・スレッドは、1、2 個のパラレル実行サーバーを使用します。通常、最適な並列度が計算されるため、*integer* に値を指定する必要はありません。

参照： 14-43 ページの「CREATE TABLE」の「[parallel_clause](#)に関する注意事項」を参照してください。

partitioning_clauses

partitioning_clauses を使用して、マテリアライズド・ビュー・ログが、指定された範囲の値またはハッシュ・ファンクションでパーティション化されることを指定できます。マテリアライズド・ビュー・ログのパーティション化は、表のパーティション化と同じです。詳細は、14-6 ページの「[CREATE TABLE](#)」を参照してください。

WITH 句

WITH 句を使用して、マスター内の行の更新時に、マテリアライズド・ビュー・ログに主キー、ROWID、オブジェクト ID またはこれらの行識別子の組合せを記録するかどうかを指定します。また、マテリアライズド・ビュー・ログに順序を追加し、レコードに対する追加の順序情報を提供することもできます。

この句を指定すると、マテリアライズド・ビュー・ログが**フィルタ列**（副問合せマテリアライズド・ビューが参照する主キー以外の列）または**結合列**（副問合せの WHERE 句で結合を定義する主キー以外の列）として参照する追加の列を記録するかどうかを指定できます。

この句を指定しない場合、または PRIMARY KEY、ROWID または OBJECT ID なしで句を指定する場合は、デフォルトで主キー値が格納されます。ただし、作成時に OBJECT ID または ROWID のみを指定した場合は、主キー値は暗黙的に格納されません。明示的に、またはデフォルトで作成された主キー・ログによって、主キー制約の追加の検証が実行されます。

OBJECT ID OBJECT ID によって、更新されるすべての行に対するシステム生成またはユーザー定義のオブジェクト識別子をマテリアライズド・ビュー・ログに記録することを指定します。

制限事項： OBJECT ID は、オブジェクト表のログの作成時のみに指定でき、記憶表用には指定できません。

PRIMARY KEY PRIMARY KEY によって、更新されるすべての行の主キーをマテリアライズド・ビュー・ログに記録することを指定します。

ROWID ROWID によって、更新されるすべての行の ROWID をマテリアライズド・ビュー・ログに記録することを指定します。

SEQUENCE SEQUENCE によって、追加の順序情報を提供する順序値をマテリアライズド・ビュー・ログに記録することを指定します。更新操作後の高速リフレッシュには、順序番号が必要です。

参照： マテリアライズド・ビュー・ログの順序番号の使用およびこの句の使用例については、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

column 更新されるすべての行に対して、マテリアライズド・ビュー・ログに値を記録する列を指定します。通常、フィルタ列（マテリアライズド・ビューが参照する主キー以外の列）および結合列（副問合せの WHERE 句で結合を定義する主キー以外の列）を指定します。

WITH 句の制限事項

- 各マテリアライズド・ビュー・ログに指定できるのは、PRIMARY KEY、ROWID、OBJECT ID をそれぞれ 1 回、および 1 つの列リストです。
- 主キー列は、マテリアライズド・ビュー・ログに暗黙的に記録されます。そのため、column が主キー列の 1 つを含む場合は、次の結合はいずれも指定できません。

```
WITH ... PRIMARY KEY ... (column)
WITH ... (column) ... PRIMARY KEY
WITH (column)
```

参照：

- マテリアライズド・ビュー・ログの値の、明示的および暗黙的な論理和については、13-5 ページの「[CREATE MATERIALIZED VIEW](#)」を参照してください。
- フィルタ列および結合列の詳細は、『Oracle9i レプリケーション』を参照してください。

NEW VALUES 句

NEW VALUES 句によって、マテリアライズド・ビュー・ログの古い値と新しい値の両方を保存するかどうかを指定できます。

INCLUDING INCLUDING を指定すると、古い値と新しい値の両方を保存します。このログが単一表マテリアライズド集計ビューの表用で、マテリアライズド・ビューに高速リフレッシュを実行する場合、INCLUDING を指定してください。

EXCLUDING EXCLUDING を指定すると、ログに対する新しい値の記録を使用禁止にします。これはデフォルトです。この句を使用すると、新しい値の記録によるオーバーヘッドを回避できます。ただし、この表に高速リフレッシュが可能な単一表マテリアライズド集計ビューが定義されている場合は、この句を使用しないでください。

例

主キーの例 次の文は、主キー値のみを記録する `oe.customers` 表のマテリアライズド・ビュー・ログを作成します。

```
CREATE MATERIALIZED VIEW LOG ON customers
  WITH PRIMARY KEY;
```

このマテリアライズド・ビューを使用して、`customers` 表に後で作成される単純主キー・マテリアライズド・ビューに対する高速リフレッシュが実行されます。

次の文も、変更された行の主キーのみを記録する `customers` 表のマテリアライズド・ビュー・ログを作成します。ただし、この例では物理特性および記憶特性も指定しています。

```
CREATE MATERIALIZED VIEW LOG ON customers
  PCTFREE 5
  TABLESPACE demo
  STORAGE (INITIAL 10K NEXT 10K);
```

ROWID の例 次の文は、更新された行の主キーおよび ROWID を記録するマテリアライズド・ビュー・ログを作成します。

```
CREATE MATERIALIZED VIEW LOG ON customers WITH ROWID, PRIMARY KEY;
```

フィルタ列の例 次の文は、デモ・スキーマ `oe` の `inventories` 表にマテリアライズド・ビュー・ログを作成します。ログは、主キーおよびフィルタ列 `quantity_on_hand` を記録します。

```
CREATE MATERIALIZED VIEW LOG ON inventories
  WITH PRIMARY KEY (quantity_on_hand);
```

結合列の例 次の文は、デモ・スキーマ `oe` の `order_items` 表にマテリアライズド・ビュー・ログを作成します。ログは、主キーおよび [13-27 ページの「副問合せおよび UNION を伴う高速リフレッシュが可能なマテリアライズド・ビューの例」](#) で結合列として使用した `product_id` を記録します。

```
CREATE MATERIALIZED VIEW LOG ON order_items WITH (product_id);
```

NEW VALUES の例 次の例は、INCLUDING NEW VALUES を指定する
oe.product_information 表にマテリアライズド・ビュー・ログを作成します。

```
CREATE MATERIALIZED VIEW LOG ON product_information
  WITH ROWID (product_id, list_price, min_price, category_id)
  INCLUDING NEW VALUES;
```

次の文は、マテリアライズド集計ビューを作成し、product_information ログを使用します。

```
CREATE MATERIALIZED VIEW products_mv
  REFRESH FAST ON COMMIT
  AS SELECT SUM(list_price - min_price), category_id
  FROM product_information
  GROUP BY category_id;
```

使用するログに古い値と新しい値の両方が含まれるため、このマテリアライズド・ビューでは、高速リフレッシュを実行できます。

CREATE OPERATOR

用途

CREATE OPERATOR 文を使用すると、新しい演算子を作成し、バインディングを定義できます。

演算子は、索引タイプ、DML および問合せ SQL 文によって参照できます。演算子は、ファンクション、パッケージ、型および他のユーザー定義オブジェクトを順番に参照します。

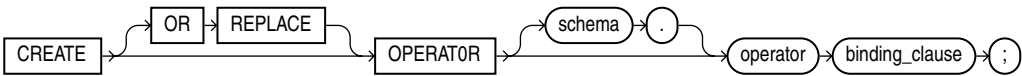
参照： これらの依存性および一般的な演算子については、『Oracle9i Data Cartridge Developer’s Guide』および『Oracle9i データベース概要』を参照してください。

前提条件

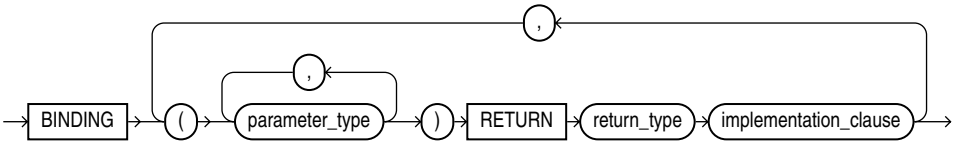
自分のスキーマ内に演算子を作成する場合は、CREATE OPERATOR システム権限が必要です。他のユーザーのスキーマ内に演算子を作成する場合は、CREATE ANY OPERATOR システム権限が必要です。どちらの場合も、参照するファンクションおよび演算子に対する EXECUTE 権限が必要です。

構文

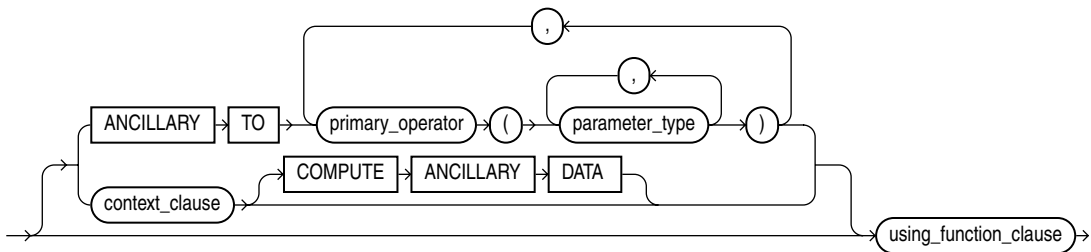
create_operator::=



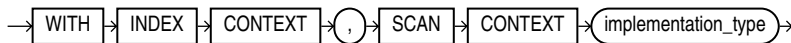
binding_clause::=



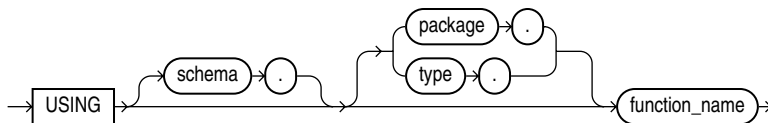
implementation_clause::=



context_clause::=



using_function_clause::=



キーワードとパラメータ

OR REPLACE

OR REPLACE を指定して、演算子スキーマ・オブジェクトの定義を置換します。

制限事項： 演算子に依存オブジェクトがない場合のみ、定義を再作成できます（たとえば、演算子をサポートする索引タイプ）。

schema

演算子が含まれているスキーマを指定します。*schema* を省略した場合、自分のスキーマ内に演算子を作成されます。

operator

作成する演算子の名前を指定します。

binding_clause

binding_clause を使用して、演算子をファンクションにバインドするために、1 つ以上のパラメータ・データ型 (*parameter_type*) を指定します。各バインドの署名 (対応するファンクションの引数のデータ型順序) は、オーバーロードの規則に従って一意である必要があります。

parameter_type 自体をオブジェクト型にできます。この場合、任意にスキーマで修飾することもできます。

制限事項： REF、LONG または LONG RAW は *parameter_type* に指定できません。

参照： オーバーロードの詳細は、『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。

RETURN 句

バインドに戻りデータ型を指定します。

return_type 自体をオブジェクト型にできます。この場合、任意にスキーマで修飾することもできます。

制限事項： REF、LONG または LONG RAW は *return_type* に指定できません。

implementation_clause**ANCILLARY TO 句**

ANCILLARY TO 句を使用して、演算子バインドが、指定した主演算子バインド (*primary_operator*) を補助することを指定します。この句を指定する場合は、1 つのみの数値パラメータで前のバインドを指定しないでください。

context_clause

context_clause を使用して、演算子の機能的な実装により、スキャン・コンテキストとして使用される実装タイプ名を指定します。

COMPUTE ANCILLARY DATA COMPUTE ANCILLARY DATA によって、演算子バインディングが補助データを計算するように指定します。

using_function_clause

using_function_clause によって、バインディングを実装するファンクションを指定します。*function_name* には、スタンドアロン・ファンクション、パッケージ・ファンクション、型メソッドまたはこれらのシノニムを指定できます。

例

CREATE OPERATOR の例 次の例は、2つのバインディングがある `scott` スキーマに `MERGE` という演算子を作成します。1番目のバインディングは、2つの `VARCHAR2` 値をマージし、`VARCHAR2` の結果を戻します。2番目のバインディングは、2つのジオメトリを単一のジオメトリにマージします。バインディングに対応する機能的な実装も指定します。(例では、`text.merge` および `spatial.merge` ファンクションが宣言されているものとします。)

```
CREATE OPERATOR oe.merge
BINDING (varchar2, varchar2) RETURN varchar2
        USING text.merge,
        (spatial.geo, spatial.geo) RETURN spatial.geo
        USING spatial.merge;
```

CREATE OUTLINE

用途

CREATE OUTLINE 文を使用すると、**ストアド・アウトライン**を作成できます。ストアド・アウトラインは、実行計画を生成するためにオブティマイザが使用する属性集合です。最適化に影響する要因に変更があるにもかかわらず、特定の SQL 文が発行された場合、実行計画の生成に影響するアウトラインの集合を使用するように、オブティマイザに指示します。これらの要因における変更を考慮するように、アウトラインを変更することもできます。

注意： 後で発行する SQL 文には、アウトライン作成時に指定した文と一致する文字列を指定する必要があります。

参照：

- 実行計画の詳細は、『Oracle9i データベース・パフォーマンス・ガイド およびリファレンス』を参照してください。
- アウトラインの変更については、8-97 ページの「[ALTER OUTLINE](#)」を参照してください。
- USE_STORED_OUTLINES パラメータおよび USE_PRIVATE_OUTLINES パラメータの詳細は、9-2 ページの「[ALTER SESSION](#)」および 9-19 ページの「[ALTER SYSTEM](#)」を参照してください。

前提条件

パブリック・アウトラインまたはプライベート・アウトラインの作成には、CREATE ANY OUTLINE システム権限が必要です。

ソース・アウトラインからクローン・アウトラインを作成するには、SELECT_CATALOG_ROLE ロールが必要です。

プライベート・アウトラインを作成するには、DBMS_OUTLN_EDIT.CREATE_EDIT_TABLES プロシージャの実行によって自分のスキーマのアウトライン・データを保持するアウトライン編集表が必要です。このプロシージャの実行には、DBMS_OUTLN_EDIT パッケージに対する EXECUTE 権限が必要です。

個々のセッションまたはシステムに対して、ストアド・アウトラインを使用可能または使用禁止にします。

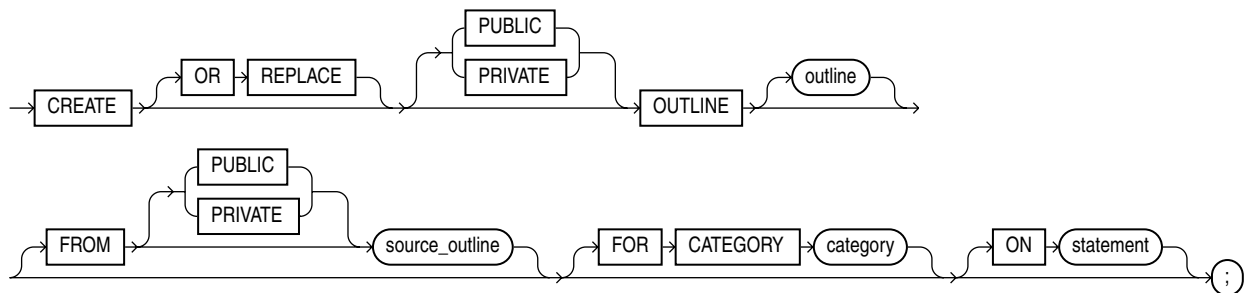
- `USE_STORED_OUTLINES` パラメータを使用可能にすると、パブリック・アウトラインが使用可能になります。
- `USE_PRIVATE_OUTLINES` パラメータを使用可能にすると、プライベート・ストアド・アウトラインが使用可能になります。

参照：

- アウトラインの使用によるパフォーマンス・チューニングの詳細は、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。
- `DBMS_OUTLN_EDIT` パッケージの詳細は、『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

構文

`create_outline::=`



キーワードとパラメータ

OR REPLACE

`OR REPLACE` を指定して、既存のアウトラインを同じ名前の新しいアウトラインと置き換えます。

PUBLIC | PRIVATE

PUBLIC が使用するアウトラインを作成する場合は、PUBLIC を指定します。これはデフォルトです。

現行のセッションのみがプライベートに使用するアウトラインを作成する場合は、PRIVATE を指定します。このアウトラインのデータは、現行のスキーマに格納されます。

注意： 最初にプライベート・アウトラインを作成する前に、OUTLN_PKG.CREATE_EDIT_TABLES プロシージャを実行し、自分のスキーマ内に必要なアウトライン表および索引を作成する必要があります。

outline

ストアド・アウトラインに割り当てる一意の名前を指定します。outline を指定しない場合、システムがアウトライン名を生成します。

FROM ... source_outline 句

FROM 句を使用すると、既存のアウトラインのコピーによって新しいアウトラインが作成されます。デフォルトでは、パブリック領域の source_category が検索されます。PRIVATE を指定すると、現行のスキーマのアウトラインが検索されます。

制限事項： FROM 句を指定する場合は、ON 句は指定できません。

FOR CATEGORY 句

グループ・ストアド・アウトラインに使用される任意の名前を指定します。たとえば、週末に使用するアウトラインの 1 つのカテゴリおよび四半期末に使用する別のカテゴリを指定できます。category を指定しない場合、アウトラインは DEFAULT カテゴリに格納されます。

ON 句

文をコンパイルする際にアウトラインが作成される SQL 文を指定します。この句は、FROM 句を使用して既存のアウトラインのコピーを作成する場合のみに指定するオプションです。

次の文のいずれかを指定できます。

- SELECT
- DELETE
- UPDATE
- INSERT ... SELECT
- CREATE TABLE ... AS SELECT

制限事項：

- ON 句を指定する場合は、FROM 句は指定できません。
- マルチ表の INSERT 文でアウトラインを作成できません。

注意： 単一文に対して複数アウトラインを指定できますが、同じ文に対するアウトラインは別のカテゴリにある必要があります。

例

CREATE OUTLINE の例 次の文は、ON 文をコンパイルすることによってストアド・アウトラインを作成します。アウトラインは `salaries` という名前で、`special` カテゴリに格納されます。

```
CREATE OUTLINE salaries FOR CATEGORY special
ON SELECT last_name, salary FROM employees;
```

USE_STORED_OUTLINES パラメータに `special` が設定されている場合、同じ SELECT 文が後でコンパイルされると、アウトライン `salaries` を作成する場合と同様に実行計画が生成されます。

プライベート・クローン・アウトラインの作成例 次の文は、前述の例で作成したパブリック・アウトライン `salaries` に基づいて、ストアド・プライベート・アウトライン `my_salaries` を作成します。プライベート・アウトラインを作成する場合、プライベート・アウトラインを作成するユーザーには、DBMS_OUTLN_EDIT パッケージに対する EXECUTE 権限が必要です。また、このパッケージの CREATE_EDIT_TABLES プロシージャを実行する必要があります。

```
EXECUTE DBMS_OUTLN_EDIT.CREATE_EDIT_TABLES;
```

```
CREATE OR REPLACE PRIVATE OUTLINE my_salaries
FROM salaries;
```

プライベート・アウトラインのパブリック領域への公開例 次の文は、プライベート編集の後でプライベート・アウトラインをパブリック領域にコピー（または、公開）します。

```
CREATE OR REPLACE OUTLINE public_salaries
FROM PRIVATE my_salaries;
```

CREATE PACKAGE

用途

CREATE PACKAGE 文を使用すると、**ストアド・パッケージ**を指定できます。ストアド・パッケージとは、関連するプロシージャ、ファンクション、およびデータベース上にまとめて格納されるその他のプログラム・オブジェクトの集合のことです。**仕様部**では、これらのオブジェクトを宣言します。

参照：

- パッケージ実装の指定については、13-51 ページの「[CREATE PACKAGE BODY](#)」を参照してください。
- スタンドアロン・ファンクションおよびプロシージャの作成については、12-47 ページの「[CREATE FUNCTION](#)」および 13-58 ページの「[CREATE PROCEDURE](#)」を参照してください。
- パッケージの変更については、8-99 ページの「[ALTER PACKAGE](#)」を参照してください。
- パッケージの削除については、15-85 ページの「[DROP PACKAGE](#)」を参照してください。
- パッケージの詳細および使用方法については、『Oracle9i アプリケーション開発者ガイド - 基礎編』および『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

前提条件

パッケージを作成する前に、ユーザー SYS は、DBMSSTD_X.SQL と呼ばれる SQL スクリプトを起動する必要があります。このスクリプトの正確なファイル名および位置は、使用するオペレーティング・システムによって異なります。

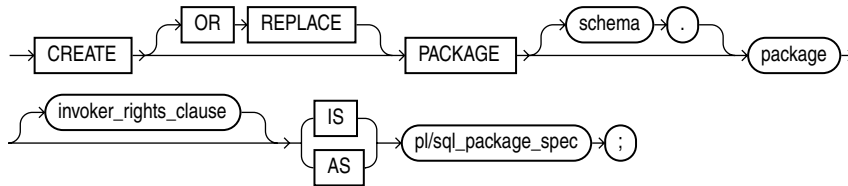
自分のスキーマ内にパッケージを作成する場合は、CREATE PROCEDURE システム権限が必要です。他のユーザーのスキーマ内にパッケージを作成する場合は、CREATE ANY PROCEDURE システム権限が必要です。

Oracle プリコンパイラ・プログラム内に CREATE PACKAGE 文を埋め込む場合、キーワード END-EXEC に続けて、各言語の埋込み SQL 文の終了記号を記述して文を終了する必要があります。

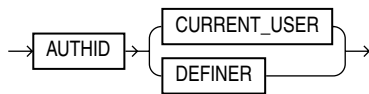
参照：『PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。

構文

create_package::=



invoker_rights_clause::=



キーワードとパラメータ

OR REPLACE

既存のパッケージ仕様部を再作成する場合は、**OR REPLACE** を指定します。この句を指定した場合、パッケージに対して付与されていたオブジェクト権限を削除、再作成および再付与しなくても、既存のパッケージの仕様部を変更できます。パッケージ仕様部を変更した場合、その仕様部は自動的に再コンパイルされます。

再定義したパッケージに対して権限が付与されていたユーザーは、権限が再付与されなくてもそのパッケージにアクセスできます。

ファンクション索引がパッケージに依存している場合、索引に **DISABLED** のマークが付きます。

参照： パッケージ仕様部の再コンパイルについては、8-99 ページの「**ALTER PACKAGE**」を参照してください。

schema

パッケージを含むスキーマを指定します。*schema* を指定しない場合、自分のスキーマ内にパッケージが作成されます。

package

作成するパッケージの名前を指定します。

パッケージの作成がコンパイル・エラーになった場合、Oracle はエラーを戻します。SQL*Plus コマンド SHOW ERRORS を使用して、関連するコンパイラ・エラー・メッセージを表示できます。

invoker_rights_clause

invoker_rights_clause によって、パッケージ内のファンクションおよびプロシージャが、権限を持つユーザーのスキーマ内で実行されるか、または、権限を持つ CURRENT_USER のスキーマ内で実行されるかを指定できます。この仕様部は対応するパッケージ本体にも適用されます。

この句は、問合せ、DML 操作およびパッケージにおける動的 SQL 文の外部名の変換方法も定義します。

AUTHID CURRENT_USER

パッケージを CURRENT_USER 権限で実行する場合は、CURRENT_USER を指定します。この句によって**実行者権限パッケージ**が作成されます。

また、この句は、問合せ、DML 操作、および動的 SQL 文の外部名を CURRENT_USER のスキーマで変換することも指定します。他のすべての文における外部名は、パッケージを含むスキーマで変換します。

AUTHID DEFINER

パッケージを含むスキーマの所有者権限でパッケージを実行する場合、およびパッケージを含むスキーマ内で外部名を変換する場合は、DEFINER を指定します。これはデフォルトで、**定義者権限パッケージ**を作成します。

参照：

- 『PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。
- CURRENT_USER の判断方法については、『Oracle9i データベース概要』および『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

pl/sql_package_spec

型定義、カーソル宣言、変数宣言、定数宣言、例外宣言、PL/SQL サブプログラム仕様およびコール仕様（PL/SQL 内で記述された C または Java ルーチンの宣言）を含むことができるパッケージ仕様部を指定します。

参照：

- PL/SQL パッケージ・プログラム・ユニットの詳細は、『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 が提供するパッケージについては、『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。
- パッケージにおけるユーザー定義ファンクションの制限事項については、12-50 ページの「[ユーザー定義ファンクションの制限事項](#)」を参照してください。

例

CREATE PACKAGE の例 次の SQL 文は、emp_mgmt というパッケージの仕様部を作成します。

```
CREATE PACKAGE emp_mgmt AS
  FUNCTION hire (last_name varchar2(20), job_id varchar2(10),
    manager_id NUMBER(6), salary NUMBER(8,2),
    commission_pct NUMBER(2,2), department_id NUMBER(4))
    RETURN NUMBER;
  FUNCTION create_dept(department_id NUMBER(4), location NUMBER(4))
    RETURN NUMBER;
  PROCEDURE remove_emp(employee_id NUMBER(6));
  PROCEDURE remove_dept(department_id NUMBER(4));
  PROCEDURE increase_sal(employee_id NUMBER(4),
    salary_incr NUMBER);
  PROCEDURE increase_comm(employee_id NUMBER(4), 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 および create_dept の各ファンクション
- remove_emp、remove_dept、increase_sal および increase_comm の各プロシージャ
- no_comm および no_sal の各例外

これらのすべてのオブジェクトは、このパッケージに対するアクセス権限があるユーザーが使用できます。パッケージの作成後は、パッケージのパブリック・プロシージャまたはファンクションをコールしたり、パッケージのパブリック例外を発生させるアプリケーションを開発できます。

このパッケージのプロシージャおよびファンクションをコールする前に、パッケージ本体でこれらのプロシージャおよびファンクションを定義しておく必要があります。emp_mgmt パッケージの本体を作成する CREATE PACKAGE BODY 文の例は、13-51 ページの「[CREATE PACKAGE BODY](#)」を参照してください。

CREATE PACKAGE BODY

用途

CREATE PACKAGE BODY 文を使用すると、**ストアド・パッケージ**を作成できます。ストアド・パッケージとは、関連するプロシージャ、ストアド・ファンクション、およびデータベース上にまとめて格納されるその他のプログラム・オブジェクトの集合のことです。**本体**では、これらのオブジェクトを定義します。

一連のプロシージャやファンクションをスタンドアロンのスキーマ・オブジェクトとして作成するかわりの方法としてパッケージを使用する方法があります。

参照：

- スタンドアロン・ファンクションおよびプロシージャの作成については、12-47 ページの「[CREATE FUNCTION](#)」および 13-58 ページの「[CREATE PROCEDURE](#)」を参照してください。
- 作成方法を含むパッケージの詳細は、13-46 ページの「[CREATE PACKAGE](#)」を参照してください。
- 使用例については、13-53 ページの「[例](#)」を参照してください。
- パッケージの変更については、8-99 ページの「[ALTER PACKAGE](#)」を参照してください。
- データベースからのパッケージの削除については、15-85 ページの「[DROP PACKAGE](#)」を参照してください。

前提条件

パッケージを作成する前に、ユーザー SYS は、DBMSSTD_X.SQL と呼ばれる SQL スクリプトを実行する必要があります。このスクリプトの正確なファイル名および位置は、使用するオペレーティング・システムによって異なります。

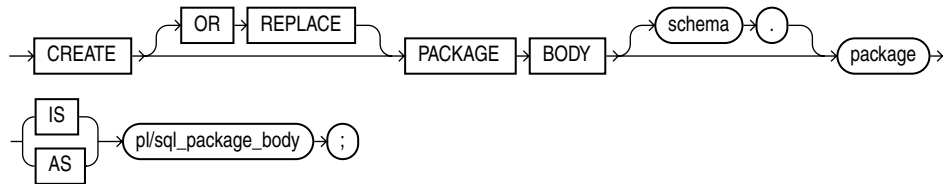
自分のスキーマ内にパッケージを作成する場合は、CREATE PROCEDURE システム権限が必要です。他のユーザーのスキーマ内にパッケージを作成する場合は、CREATE ANY PROCEDURE システム権限が必要です。

Oracle プリコンパイラ・プログラム内に CREATE PACKAGE BODY 文を埋め込む場合、キーワード END-EXEC とその後に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する必要があります。

参照：『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

create_package_body::=



キーワードとパラメータ

OR REPLACE

既存のパッケージ本体を再作成する場合は、`OR REPLACE` を指定します。この句を指定した場合、パッケージに対して付与されていたオブジェクト権限を削除、再作成および再付与しなくても、既存のパッケージの本体を変更できます。パッケージ本体を変更した場合、その本体は自動的に再コンパイルされます。

再定義したパッケージに対して権限が付与されていたユーザーは、権限が再付与されなくてもそのパッケージにアクセスできます。

参照： パッケージ本体の再コンパイルについては、8-99 ページの「[ALTER PACKAGE](#)」を参照してください。

schema

パッケージを含むスキーマを指定します。*schema* を指定しない場合、現行のスキーマ内にパッケージが作成されます。

package

作成するパッケージの名前を指定します。

pl/sql_package_body

PL/SQL サブプログラム本体またはコール仕様（PL/SQL に記述された C または Java ルーチンの宣言）を含むことができるパッケージ本体を指定します。

参照：

- PL/SQL または C パッケージ・プログラム・ユニットへの書込みの詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。
- Java パッケージ・プログラム・ユニットについては、『Oracle9i Java Stored Procedures Developer's Guide』を参照してください。
- パッケージにおけるユーザー定義ファクションの制限事項については、12-50 ページの「[ユーザー定義ファクションの制限事項](#)」を参照してください。

例

CREATE PACKAGE BODY の例 次の SQL 文は、13-49 ページの「[CREATE PACKAGE の例](#)」で作成した emp_mgmt パッケージの本体を作成します。

```
CREATE PACKAGE BODY emp_mgmt AS
    tot_emps NUMBER;
    tot_depts NUMBER;

    FUNCTION hire
        (last_name VARCHAR2(20), job_id VARCHAR2(10),
         manager_id NUMBER(6), salary NUMBER(8,2),
         commission_pct NUMBER(2,2), department_id NUMBER(4))
    RETURN NUMBER IS
        new_empno NUMBER(4);
    BEGIN
        SELECT employees_seq.NEXTVAL
            INTO new_empno
            FROM DUAL;
        INSERT INTO employees
            VALUES (new_empno, last_name, job_id, manager_id, salary,
                    commission_pct, department_id, tot_emps := tot_emps + 1;
        RETURN(new_empno);
    END;

    FUNCTION create_dept(department_id NUMBER(4), location_id NUMBER(4))
    RETURN NUMBER IS
        new_deptno NUMBER(4)
    BEGIN
```

```
        SELECT departments_seq.NEXTVAL
        INTO new_deptno
        FROM dual;
    INSERT INTO departments
        VALUES (new_deptno, department_id, location_id);
        tot_depts := tot_depts | 1;
    RETURN(new_deptno);
END;

PROCEDURE remove_emp (employee_id NUMBER(6)) IS
BEGIN
    DELETE FROM employees
    WHERE employees.employee_id = remove_emp.employee_id;
        tot_emps := tot_emps - 1;
END;

PROCEDURE remove_dept(department_id NUMBER ($)) IS
BEGIN
    DELETE FROM departments
    WHERE departments.department_id = remove_dept.department_id;
        tot_depts := tot_depts - 1;
    SELECT COUNT(*) INTO tot_emps FROM employees;
    /* In case, oracle deleted employees from the EMPLOYEES
    table to enforce referential integrity constraints, reset
    the value of the variable TOT_EMPS to the total number of
    employees in the EMPLOYEES table. */
END;

PROCEDURE increase_sal (employee_id NUMBER(6), sal_incr NUMBER) IS
    curr_sal NUMBER(7,2);
BEGIN
    SELECT salary INTO curr_sal FROM employees
    WHERE employees.employee_id = increase_sal.employee_id;
    IF curr-Sal IS NULL
        THEN RAISE no_sal;
    ELSE
        UPDATE employees
        SET salary = salary + sal_incr
        WHERE employee_id = employee_id;
    ENDIF;
END;

PROCEDURE increase_comm(employee_id NUMBER(6), comm_incr NUMBER) IS
    curr_comm NUMBER(7,2);
BEGIN
    SELECT commission_pct
    INTO curr_comm
```

```

FROM employees
WHERE employees.employee_id = increase_comm.employee_id;
IF curr_comm IS NULL
    THEN RAISE no_comm;
ELSE
    UPDATE employees
    SET commission_pct = commission_pct + comm_incr;
END IF;
END;

END emp_mgmt;

```

パッケージ本体では、パッケージ仕様部で宣言した次のパブリック・プログラム・オブジェクトを定義しています。

- hire および create_dept の各ファンクション
- remove_emp、remove_dept、increase_sal および increase_comm の各プロシージャ

これらのオブジェクトは、パッケージ仕様部で宣言されているため、パッケージ外のアプリケーション・プログラム、プロシージャおよびファンクションからコールできます。たとえば、パッケージに対するアクセス権限がある場合は、increase_comm プロシージャをコールする emp_mgmt パッケージとは別に、increase_all_comms プロシージャを作成できます。

これらのオブジェクトはパッケージ本体で定義されているため、その定義を変更した場合でも、Oracle が依存スキーマ・オブジェクトを無効にすることはありません。たとえば、後で hire の定義を変更した場合に、increase_all_comms は実行するまで再コンパイルされません。

また、この例のパッケージ本体では、プライベート・プログラム・オブジェクトである変数 tot_emps および tot_depts が宣言されています。これらのオブジェクトは、パッケージ仕様部ではなくパッケージ本体で宣言されているため、パッケージ内の他のオブジェクトからアクセスできますが、パッケージ外からはアクセスできません。たとえば、変数 tot_depts の値を明示的に変更するアプリケーションは開発できません。ただし、ファンクション create_dept はパッケージの一部であるため、create_dept で tot_depts の値を変更できます。

CREATE PFILE

用途

CREATE PFILE 文を使用すると、バイナリのサーバー・パラメータ・ファイルをテキストの初期化パラメータ・ファイルにエクスポートできます。テキストのパラメータ・ファイルの作成は、データベースで使用する現行のパラメータ設定リストの取得に便利です。また、テキスト・エディタで簡単に編集でき、CREATE SPFILE 文を使用してサーバー・パラメータ・ファイルに変換して戻すこともできます。

この文が正常に実行されると、サーバーにテキストのパラメータ・ファイルが作成されます。Real Application Clusters 環境では、すべてのインスタンスのすべてのパラメータ設定が含まれます。サーバー・パラメータ・ファイルのパラメータ設定と同じ行のコメントも含まれます。

参照：

- サーバー・パラメータ・ファイルの詳細は、13-86 ページの「[CREATE SPFILE](#)」を参照してください。
- Oracle9i より前のテキストの初期化パラメータ・ファイル、および Oracle9i のバイナリのサーバー・パラメータ・ファイルの詳細は、『Oracle9i データベース管理者ガイド』を参照してください。
- Real Application Clusters 環境でのサーバー・パラメータ・ファイルの使用については、『Oracle9i Real Application Clusters 管理』を参照してください。

前提条件

この文を実行するには、SYSDBA ロールまたは SYSOPER ロールが必要です。この文は、インスタンスの起動前と起動後のいずれかで実行できます。

構文

create_pfile::=



キーワードとパラメータ

pfile_name

作成するテキストのパラメータ・ファイル名を指定します。*pfile_name* を指定しないと、プラットフォーム固有のデフォルトの初期化パラメータ・ファイル名が使用されます。

spfile_name

テキストのファイルを作成する元となるバイナリのサーバー・パラメータ・ファイル名を指定します。

- *spfile_name* を指定する場合、そのファイルがサーバーに存在する必要があります。ファイルがオペレーティング・システムのサーバー・パラメータ・ファイルのデフォルト・ディレクトリに存在しない場合、フルパス名を指定する必要があります。
- *spfile_name* を指定しない場合、オペレーティング・システムのサーバー・パラメータ・ファイルのデフォルト・ディレクトリからプラットフォーム固有のデフォルトのサーバー・パラメータ・ファイル名が検索され、使用されます。そのディレクトリにファイルが存在しない場合、エラーが戻されます。

参照： デフォルトのパラメータ・ファイル名は、オペレーティング・システム固有のドキュメントを参照してください。

例

次の例は、バイナリのサーバー・パラメータ・ファイル `production.ora` からテキストのパラメータ・ファイル `my_init.ora` を作成します。

```
CREATE PFILE = 'my_init.ora' FROM SPFILE = 'production.ora';
```

注意： 通常、オペレーティング・システムのパラメータ・ファイルには、フルパスのファイル名を指定する必要があります。パスについては、ご使用のオペレーティング・システムの Oracle マニュアルを参照してください。

CREATE PROCEDURE

用途

CREATE PROCEDURE 文を使用すると、スタンドアロンのストアド・プロシージャまたはコール仕様を作成できます。

プロシージャとは、名前によってコールできる PL/SQL 文の集合です。コール仕様 (call spec) は、SQL および PL/SQL からコールできるように、Java メソッドまたは 3GL を宣言します。コール仕様は、コールされたときに起動する Java メソッドを問い合わせます。引数および戻り値に対する型変換も問い合わせます。

ストアド・プロシージャには、開発、整合性、セキュリティ、パフォーマンスおよびメモリー割当ての面でいくつかのメリットがあります。

参照：

- コール方法などのストアド・プロシージャの詳細、および外部プロシージャの登録については、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。
- プロシージャと類似点が多いファンクションの詳細は、12-47 ページの「[CREATE FUNCTION](#)」を参照してください。
- パッケージの作成の詳細は、13-46 ページの「[CREATE PACKAGE](#)」を参照してください。(CREATE PROCEDURE 文では、スタンドアロンのスキーマ・オブジェクトとしてプロシージャが作成されます。また、プロシージャをパッケージの一部としても作成できます。)
- スタンドアロン・プロシージャの変更および削除については、8-103 ページの「[ALTER PROCEDURE](#)」および 15-87 ページの「[DROP PROCEDURE](#)」を参照してください。
- 共有ライブラリについては、13-2 ページの「[CREATE LIBRARY](#)」を参照してください。

前提条件

プロシージャを作成する前に、ユーザー SYS は、DBMSSTDY.SQL と呼ばれる SQL スクリプトを実行する必要があります。このスクリプトの正確なファイル名および位置は、使用するオペレーティング・システムによって異なります。

自分のスキーマ内にプロシージャを作成する場合は、CREATE PROCEDURE システム権限が必要です。他のユーザーのスキーマ内にプロシージャを作成する場合は、CREATE ANY PROCEDURE システム権限が必要です。他のユーザーのスキーマ内にプロシージャを再配置する場合は、ALTER ANY PROCEDURE システム権限が必要です。

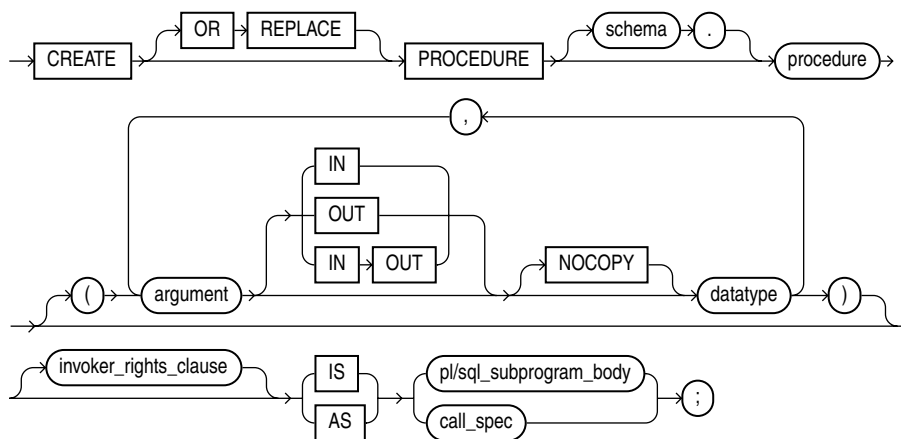
コール仕様を起動する場合、その他の権限が必要になることがあります（たとえば、C コール仕様の C ライブラリには EXECUTE 権限が必要です）。

Oracle プリコンパイラ・プログラム内に CREATE PROCEDURE 文を埋め込む場合、キーワード END-EXEC とその後に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する必要があります。

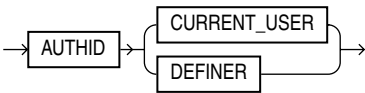
参照： 詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』および『Oracle9i Java Stored Procedures Developer's Guide』を参照してください。

構文

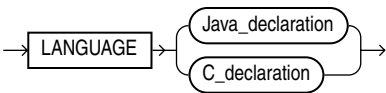
create_procedure::=



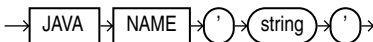
invoker_rights_clause::=



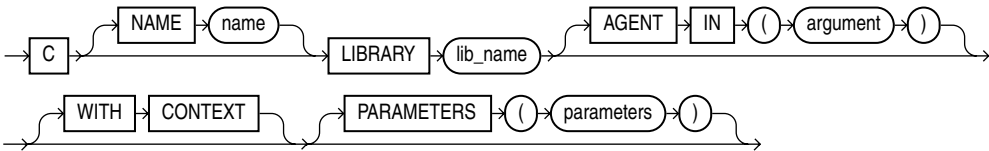
call_spec::=



Java_declaration::=



C_declaration::=



キーワードとパラメータ

OR REPLACE

既存のプロシージャを再作成する場合は、OR REPLACE を指定します。この句を指定した場合、プロシージャに付与されているオブジェクト権限を削除、再作成および再付与なくとも、既存のプロシージャの定義を変更できます。プロシージャを再定義した場合、そのプロシージャは自動的に再コンパイルされます。

再定義したプロシージャに対して権限が付与されていたユーザーは、権限が再付与されなくともそのプロシージャにアクセスできます。

ファンクション索引がパッケージに依存している場合、索引に DISABLED のマークが付きます。

参照： プロシージャの再コンパイルについては、8-103 ページの「ALTER PROCEDURE」を参照してください。

schema

プロシージャを定義するスキーマを指定します。*schema* を省略した場合、現行のスキーマ内にプロシージャが作成されます。

procedure

作成するプロシージャの名前を指定します。

プロシージャの作成でコンパイル・エラーがある場合、Oracle はエラーを戻します。SQL*Plus コマンド SHOW ERRORS を使用して、関連するコンパイラ・エラー・メッセージを表示できます。

argument

プロシージャへの引数の名前を指定します。プロシージャが引数を受け入れない場合は、プロシージャ名の後のカッコを省略できます。

IN IN は、プロシージャをコールするときに、引数に値を指定する必要があることを示します。

OUT OUT は、プロシージャが、実行後にコールの環境に対して、指定した引数の値を戻すことを示します。

IN OUT IN OUT は、プロシージャのコール時に、引数に値を指定し、プロシージャが実行後にコール先の環境に値を渡す必要があることを示します。

IN、OUT および IN OUT のいずれの引数も指定しない場合、デフォルトでは IN が設定されます。

NOCOPY NOCOPY は、できるだけ速く引数を渡すように指示します。この句は、OUT パラメータや IN OUT パラメータに対して、レコード、索引付き表、VARRAY などの大きい値を渡す際のパフォーマンスの向上に有効です。(IN パラメータ値には、常に NOCOPY が渡されます)。

- NOCOPY パラメータを指定すると、このパラメータに対応する実際の割当てとしてパッケージ変数が渡された場合に、パッケージ変数に対して行われた割当ては、すぐにこのパラメータに表示されます (または、このパラメータに対して行われた割当ては、すぐにパッケージ変数に表示されます)。
- このパラメータまたは別のパラメータに対して行われた変更は、同じ変数が両方に渡された場合、両方の名前を介してすぐに参照できます。
- プロシージャが未処理例外で終了した場合、このパラメータに対する割当ては、コール元の変数で参照できます。

このような効果がないコールもあります。この効果に問題がない場合にのみ NOCOPY を使用してください。

datatype 引数のデータ型を指定します。引数には、PL/SQL でサポートされるデータ型を指定できます。

データ型には、データ長、精度または位取りは指定できません。たとえば、VARCHAR2(10) は無効ですが、VARCHAR2 は有効です。Oracle では、そのプロシージャがコールされた環境から引数のデータ長、精度および位取りを導出します。

invoker_rights_clause

invoker_rights_clause によって、プロシージャを、スキーマを所有するユーザーの権限でそのスキーマ内で実行するか、または CURRENT_USER の権限でそのスキーマ内で実行するかを指定できます。

この句は、問合せ、DML 操作およびプロシージャにおける動的 SQL 文の外部名の変換方法も指定します。

AUTHID CURRENT_USER

プロシージャを CURRENT_USER 権限で実行する場合は、CURRENT_USER を指定します。この句によって**実行者権限プロシージャ**が作成されます。

また、この句は、問合せ、DML 操作、および動的 SQL 文の外部名を CURRENT_USER のスキーマで変換することも指定します。他のすべての文における外部名は、プロシージャを含むスキーマで変換します。

AUTHID DEFINER

プロシージャを含むスキーマの所有者権限でプロシージャを実行する場合、およびプロシージャを含むスキーマ内で外部名を変換する場合は、DEFINER を指定します。これはデフォルトで、**定義者権限プロシージャ**を作成します。

参照：

- 『PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。
- CURRENT_USER の判断方法については、『Oracle9i データベース概要』および『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

IS | AS 句

pl/sql_subprogram_body

PL/SQL サブプログラム本体のプロシージャを宣言します。

参照： PL/SQL サブプログラムの詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

call_spec

call_spec を使用して、Java メソッドまたは C 関数名、パラメータ・タイプおよび戻り型を SQL で相当するものにマップします。

Java_declaration では、'string' が JAVA 実装メソッドを定義します。

参照：

- パラメータおよび *Java_declaration* のセマンティクスの詳細は、『Oracle9i Java Stored Procedures Developer's Guide』を参照してください。
- パラメータおよび *C_declaration* のセマンティクスの詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

AS EXTERNAL AS EXTERNAL 句は、C 関数を宣言するもう 1 つの方法です。この句は以前のリリースのもので、下位互換用にのみサポートされています。AS LANGUAGE C 構文を使用することをお勧めします。

例

CREATE PROCEDURE の例 次の文は、スキーマ oe 内にプロシージャ credit を作成します。

```
CREATE PROCEDURE oe.credit
  (acc_no IN NUMBER, amount IN NUMBER) AS
BEGIN
  UPDATE accounts
  SET balance = balance + amount
  WHERE account_id = acc_no;
END;
```

プロシージャ credit を実行すると、指定した金額が指定の預金口座に記入されます。このプロシージャをコールする場合、次の引数を指定する必要があります。

- acc_no には預金口座の番号を指定します。この引数のデータ型は NUMBER です。
- amount には金額を指定します。この引数のデータ型は NUMBER です。

このプロシージャでは、UPDATE 文を使用して引数 acc_no によって特定される口座に対して、accounts 表の balance 列の値を引数 amount の値の分のみ増加させます。

次の例では、外部プロシージャ `c_find_root` によって、ポインタがパラメータとみなされます。プロシージャ `find_root` には、`BY REFERENCE` 句を使用した参照によって、そのパラメータが渡されます。

```
CREATE PROCEDURE find_root
  ( x IN REAL )
  IS LANGUAGE C
    NAME "c_find_root"
    LIBRARY c_utils
    PARAMETERS ( x BY REFERENCE );
```


CREATE PROFILE

用途

CREATE PROFILE 文を使用すると、プロファイルを作成できます。プロファイルとは、データベース・リソースの制限の設定です。あるユーザーに対してプロファイルを割り当てた場合、そのユーザーは、その割当て制限を超えることはできません。

参照： パスワード管理およびパスワード保護の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。

前提条件

プロファイルを作成する場合、CREATE PROFILE システム権限が必要です。

次の方法でユーザーに対するリソース制限を指定します。

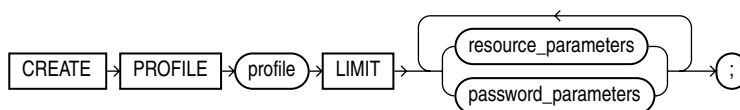
- ALTER SYSTEM 文または初期化パラメータ RESOURCE_LIMIT で動的にリソース制限を使用可能にします（このパラメータは、パスワード・リソースには適用されません。パスワード・リソースは、常に使用可能です）。
- CREATE PROFILE 文を使用して、制限を定義するプロファイルを作成します。
- CREATE USER または ALTER USER 文を使用して、プロファイルを割り当てます。

参照：

- 動的にリソース制限を使用可能にする場合の詳細は、9-19 ページの「ALTER SYSTEM」を参照してください。
- RESOURCE_LIMIT パラメータについては、『Oracle9i データベース・リファレンス』を参照してください。
- プロファイルについては、15-30 ページの「CREATE USER」および 11-20 ページの「ALTER USER」を参照してください。

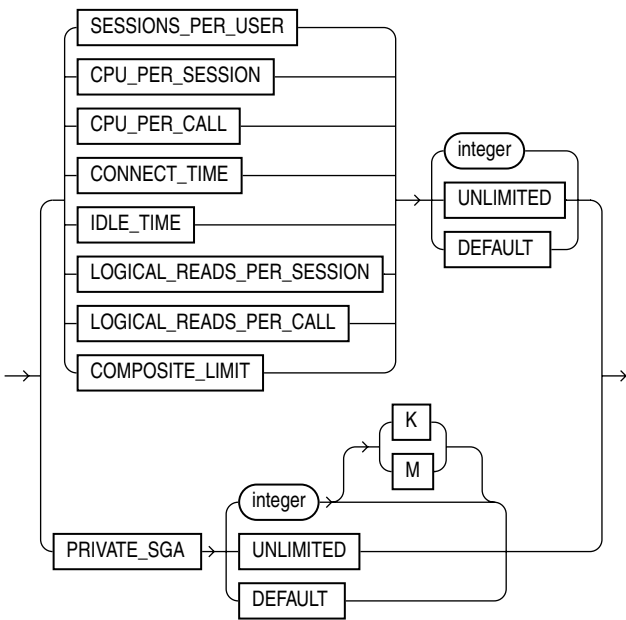
構文

create_profile::=

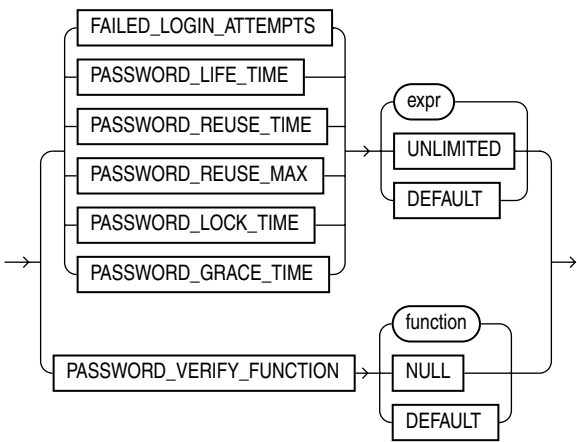


CREATE PROFILE

resource_parameters::=



password_parameters::=



キーワードとパラメータ

profile

作成するプロファイルの名前を指定します。プロファイルを使用した場合、ユーザーが使用可能なデータベース・リソースを1つのコールまたは1つのセッションごとに制限できます。

Oracle では、次の方法でリソース制限を適用します。

- `CONNECT_TIME` または `IDLE_TIME` で指定したセッション・リソース制限を超えた場合、現行のトランザクションが自動的にロールバックされ、セッションが終了します。ユーザー・プロセスで次にコールを実行すると、エラーが戻ります。
- 他のセッション・リソースに対する制限を超える処理を実行しようとした場合、その処理が自動的に判断され、現行の文がロールバックされ、エラーが戻されます。この後、ユーザーは、現行のトランザクションをコミットまたはロールバックできます。そして、セッションを終了する必要があります。
- 1つのコールに対する制限を超える処理を実行しようとした場合、その処理が自動的に中断され、現行の文がロールバックされ、エラーが戻されます。この場合、現行のトランザクションは有効です。

注意：

- 日を単位として、時間を制限するすべてのパラメータに対して、分数で日数を指定できます。たとえば、1時間は1/24、1分は1/1440になります。
 - リソース制限を使用可能にしているかどうかにかかわらず、ユーザーに対してリソース制限を指定できます。ただし、Oracle では、実際にその制限が使用可能になってからリソース制限が適用されます。
-
-

UNLIMITED

リソース・パラメータで指定した場合、UNLIMITED は、このプロファイルを割り当てられたユーザーが無制限にリソースを使用できることを指定します。パスワード・パラメータでUNLIMITED を指定した場合は、パラメータに制限が設定されていないことを示します。

DEFAULT

このプロファイルにあるリソースの制限を指定しない場合は、DEFAULT を指定します。このプロファイルを割り当てられたユーザーは、DEFAULT プロファイルで指定した対象リソースに対する制限を受けます。DEFAULT プロファイルは、最初は無制限のリソースを定義します。ALTER PROFILE 文でこの制限を変更できます。

明示的にプロファイルが割り当てられていないユーザーは、DEFAULT プロファイルに定義されている制限を受けます。ユーザーに明示的に割り当てられているプロファイルでリソースに対する制限が省略されている場合、または制限に対して DEFAULT が指定されている場合、ユーザーは DEFAULT プロファイルで定義されているリソースに関する制限を受けます。

resource_parameters

SESSIONS_PER_USER ユーザーを制限する同時セッションの数を指定します。

CPU_PER_SESSION 1セッション当たりの CPU 時間制限を指定します。この値は 100 分の 1 秒単位で指定します。

CPU_PER_CALL 1 コール（解析、実行またはフェッチ）当たりの CPU 時間制限を指定します。この値は 100 分の 1 秒単位で指定します。

CONNECT_TIME 1セッション当たりの合計経過時間制限を指定します。この値は分単位で指定します。

IDLE_TIME セッション中の連続的な非活動時間の同期を制限します。この値は分単位で指定します。長時間実行の間合せなどの処理は、この制限を受けません。

LOGICAL_READS_PER_SESSION メモリーおよびディスクから読み込まれるブロックなど、1セッション中に読み込まれるデータ・ブロックの数の制限を指定します。

LOGICAL_READS_PER_CALL SQL 文を処理するコール（解析、実行またはフェッチ）で読み込まれるデータ・ブロックの数の制限を指定します。

PRIVATE_SGA 1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 を使用すると、この制限を KB または MB 単位で指定できます。

注意： この制限は、共有サーバー・アーキテクチャを使用している場合のみに適用されます。SGA 内のセッション用のプライベート領域には、プライベート SQL および PL/SQL 領域が含まれますが、共有 SQL および PL/SQL 領域は含まれません。

COMPOSITE_LIMIT 1セッション当たりのリソースの総コストをサービス単位で指定します。サービス単位の合計は、CPU_PER_SESSION、CONNECT_TIME、LOGICAL_READS_PER_SESSIONおよびPRIVATE_SGAの重み付き合計として計算されます。

参照： セッション・リソースの重み付けについては、8-110ページの「ALTER RESOURCE COST」を参照してください。

これらのパラメータに *expr* を指定すると、式はスカラー副問合せ式を除くすべての形式が可能です。

password_parameters

FAILED_LOGIN_ATTEMPTS ユーザー・アカウントがロックされる前に、そのアカウントへのログインに失敗できる回数を指定します。

PASSWORD_LIFE_TIME 同じパスワードを認証に使用できる日数を制限します。この期間内にパスワードを変更しないと、そのパスワードは使用できなくなり、それ以降の接続は拒否されます。

PASSWORD_REUSE_TIME パスワードが再利用できなくなるまでの日数を指定します。PASSWORD_REUSE_TIMEを整数値に設定する場合は、PASSWORD_REUSE_MAXをUNLIMITEDに設定する必要があります。

PASSWORD_REUSE_MAX 現行のパスワードを再利用する前に必要な、パスワードの変更回数を指定します。PASSWORD_REUSE_MAXを整数値に設定する場合は、PASSWORD_REUSE_TIMEをUNLIMITEDに設定する必要があります。

PASSWORD_LOCK_TIME ログインが指定された回数連続して失敗した場合、アカウントがロックされる日数を指定します。

PASSWORD_GRACE_TIME 警告が出され、ログインが許可される猶予期間の日数を指定します。猶予期間中にパスワードが変更されない場合、そのパスワードは使用できなくなります。

PASSWORD_VERIFY_FUNCTION PASSWORD_VERIFY_FUNCTION句によって、PL/SQLの複雑なパスワード検証スクリプトをCREATE PROFILE文の引数として渡すことを可能にします。Oracleにはデフォルトのスクリプトがありますが、ユーザー固有のルーチンを作成することも、サード・パーティのソフトウェアを使用することもできます。

- *function* には、複雑なパスワード検証ルーチンの名前を指定します。
- NULLを指定して、パスワードの検証が行われないことを示します。

パスワード・パラメータに関する制限事項

- PASSWORD_REUSE_TIME を整数値に設定する場合、PASSWORD_REUSE_MAX を UNLIMITED に設定する必要があります。PASSWORD_REUSE_MAX を整数値に設定する場合、PASSWORD_REUSE_TIME を UNLIMITED に設定する必要があります。
- PASSWORD_REUSE_TIME および PASSWORD_REUSE_MAX の両方を UNLIMITED に設定した場合、どちらのパスワードのリソースも使用されません。
- PASSWORD_REUSE_MAX を DEFAULT に設定し、PASSWORD_REUSE_TIME を UNLIMITED に設定した場合、DEFAULT プロファイルに定義された PASSWORD_REUSE_MAX 値が使用されます。
- PASSWORD_REUSE_TIME を DEFAULT に設定し、PASSWORD_REUSE_MAX を UNLIMITED に設定した場合、DEFAULT プロファイルに定義された PASSWORD_REUSE_TIME 値が使用されます。
- PASSWORD_REUSE_TIME および PASSWORD_REUSE_MAX の両方を DEFAULT に設定した場合、DEFAULT プロファイルに定義された値はどちらも使用されます。

例

CREATE PROFILE の例 次の文は、プロファイル new_profile を作成します。

```
CREATE PROFILE new_profile
  LIMIT PASSWORD_REUSE_MAX DEFAULT
        PASSWORD_REUSE_TIME UNLIMITED;
```

リソース制限の設定の例 次の文は、プロファイル app_user を作成します。

```
CREATE PROFILE app_user LIMIT
  SESSIONS_PER_USER          UNLIMITED
  CPU_PER_SESSION            UNLIMITED
  CPU_PER_CALL                3000
  CONNECT_TIME               45
  LOGICAL_READS_PER_SESSION  DEFAULT
  LOGICAL_READS_PER_CALL     1000
  PRIVATE_SGA                15K
  COMPOSITE_LIMIT             5000000;
```

ユーザーに `app_user` プロファイルを割り当てた場合、そのユーザーは、後続のセッションで次の制限を受けます。

- ユーザーは、無制限に同時セッションを使用できます。
- 1つのセッションにおいて、ユーザーは CPU 時間を無制限に消費できます。
- ユーザーが作成した 1つのコールで消費できる CPU 時間は 30 秒以下です。
- 1つのセッションで継続できる時間は 45 分以下です。
- 1つのセッションのメモリーおよびディスクのデータ・ブロック数は、DEFAULT プロファイルで指定した制限を受けます。
- ユーザーが作成した 1つのコールで、メモリーまたはディスクから読み込むことができるデータ・ブロック数の合計は 1000 以下です。
- 1つのセッションで割り当てることができる SGA 内のメモリーは、15KB 以下です。
- 1つのセッションのリソースの総コストは、サービス単位で 500 万を超えることはできません。リソースの総コストの計算式は、ALTER RESOURCE COST 文で指定します。
- `system_manager` プロファイルでは、IDLE_TIME に対する制限が指定されていないため、ユーザーは DEFAULT プロファイルに指定されている対象リソースの制限を受けます。

パスワード制限の設定の例 次の文は、`app_user2` プロファイルに、パスワード制限値を設定して作成します。この例では、`verify_function` がすでに存在しているものとします。

```
CREATE PROFILE app_user2 LIMIT
  FAILED_LOGIN_ATTEMPTS 5
  PASSWORD_LIFE_TIME 60
  PASSWORD_REUSE_TIME 60
  PASSWORD_REUSE_MAX UNLIMITED
  PASSWORD_VERIFY_FUNCTION verify_function
  PASSWORD_LOCK_TIME 1/24
  PASSWORD_GRACE_TIME 10;
```

CREATE ROLE

用途

CREATE ROLE 文を使用すると、**ロール**を作成できます。ロールは、ユーザーまたは他のロールに付与できる権限の集合です。ロールを使用してデータベース権限を管理できます。ロールに権限を追加したうえで、ユーザーにそのロールを付与できます。その結果、ユーザーはロールを使用可能にし、そのロールによって付与された権限を使用できるようになります。

ロールには、そのロールに付与されたすべての権限、およびそのロールに付与された他のロールのすべての権限が含まれています。新しく作成されたロールには、ロールや権限は付与されていません。GRANT 文を使用して、ロールに様々な権限を追加します。

NOT IDENTIFIED、IDENTIFIED EXTERNALLY または BY *password* ロールを作成した場合、そのロールは ADMIN OPTION 付きで付与されます。ただし、ロール IDENTIFIED GLOBALLY を作成した場合、ロールは付与されません。

参照：

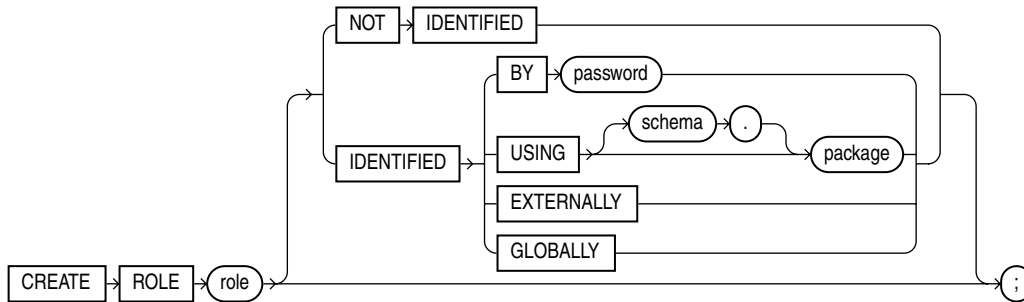
- ロールの付与については、16-31 ページの「[GRANT](#)」を参照してください。
- ロールを使用可能にする場合は、11-20 ページの「[ALTER USER](#)」を参照してください。
- ロールの変更については、8-113 ページの「[ALTER ROLE](#)」を参照してください。
- データベースからのロールの削除については、15-91 ページの「[DROP ROLE](#)」を参照してください。
- 現行のセッションに対するロールの使用可能および使用禁止については、17-43 ページの「[SET ROLE](#)」を参照してください。

前提条件

CREATE ROLE システム権限が必要です。

構文

create_role::=



キーワードとパラメータ

role

作成するロールの名前を指定します。データベース・キャラクタ・セットにマルチバイト文字がサポートされている場合でも、ロールにはシングルバイト文字を1つ以上使用することをお勧めします。

配布メディアで提供されている SQL スクリプトには、いくつかのロールが定義されています。

参照： 事前定義済のロールのリストは、16-31 ページの「[GRANT](#)」を参照してください。

NOT IDENTIFIED 句

NOT IDENTIFIED を指定すると、指定するロールがデータベースによって認可され、パスワードを入力しなくてもこのロールを使用可能にできます。

IDENTIFIED 句

IDENTIFIED 句を使用すると、SET ROLE 文によってロールを使用可能にする前に、指定したメソッドによってユーザーが認可される必要があります。

BY password BY password 句によって、**ローカル・ユーザー**を作成します。また、ロールを使用可能にするときに、ユーザーがパスワードを指定する必要があることを示します。データベース・キャラクタ・セットにマルチバイト文字が含まれている場合でも、データベース・キャラクタ・セットのシングルバイト文字のみで指定することもできます。

USING package USING *package* によって、**アプリケーション・ロール**を作成します。アプリケーション・ロールとは、アプリケーションが認可されたパッケージを使用することによってのみ使用可能になるロールです。*schema* を指定しない場合、パッケージが自分のスキーマ内にあるとみなされます。

EXTERNALLY EXTERNALLY によって、**外部ユーザー**を作成します。ロールを使用可能にする前に、ユーザーがオペレーティング・システムやサード・パーティ・サービスなどの外部サービスによって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できない場合もあります。

GLOBALLY GLOBALLY によって、**グローバル・ユーザー**を作成します。また、SET ROLE 文を使用して、またはログイン時にロールを使用可能にする前に、ユーザーがエンタープライズ・ディレクトリ・サービスによってロールの使用を認可されている必要があることを示します。

NOT IDENTIFIED 句および IDENTIFIED 句を両方とも省略した場合、ロールには NOT IDENTIFIED がデフォルト値として使用されます。

例

CREATE ROLE の例 次の文は、ロール `dw_manager` を作成します。

```
CREATE ROLE dw_manager;
```

この後に `dw_manager` を付与されるユーザーは、このロールに付与されているすべての権限を継承します。

次の例のように、パスワードの指定によってロールにセキュリティのレイヤーを追加できます。

```
CREATE ROLE dw_manager  
  IDENTIFIED BY warehouse;
```

この後、`dw_manager` ロールを付与されたユーザーは、パスワード `warehouse` を指定して、SET ROLE 文でロールを使用可能にする必要があります。

次の文は、グローバル・ロール `warehouse_user` を作成します。

```
CREATE ROLE warehouse_user IDENTIFIED GLOBALLY;
```

CREATE ROLLBACK SEGMENT

用途

CREATE ROLLBACK SEGMENT 文を使用すると、ロールバック・セグメントを作成できます。ロールバック・セグメントとは、トランザクションによる変更を元に戻す（取り消す）ために必要なデータを格納する際に Oracle が使用するオブジェクトです。

ここでは、データベースがロールバック UNDO モードで実行されている（初期化パラメータ UNDO_MANAGEMENT を MANUAL に設定、またはすべて設定しない）ことを前提としています。

データベースが自動 UNDO 管理モードで実行されている場合（初期化パラメータ UNDO_MANAGEMENT を AUTO に設定）、ユーザー作成ロールバック・セグメントを変更することはできません。この場合、CREATE ROLLBACK SEGMENT または ALTER ROLLBACK SEGMENT 文の応答でエラーが戻されます。このエラーを回避するには、UNDO_SUPPRESS_ERRORS パラメータを TRUE に設定します。

注意： SYSTEM 以外の表領域のオブジェクトを使用する場合は、次の注意事項があります。

- ロールバック UNDO モードのデータベースを実行するには、1 つ以上のロールバック・セグメント（SYSTEM ロールバック・セグメント以外の）がオンラインである必要があります。
 - 自動 UNDO 管理モードのデータベースを実行するには、1 つ以上の UNDO 表領域がオンラインである必要があります。
-

参照：

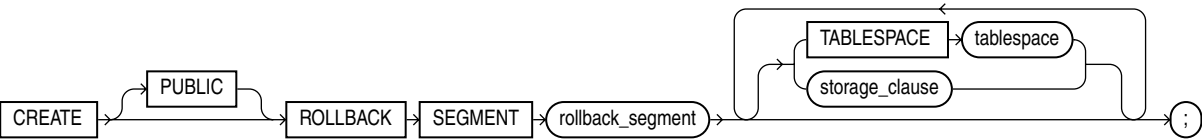
- ロールバック・セグメントの変更については、8-115 ページの「ALTER ROLLBACK SEGMENT」を参照してください。
- ロールバック・セグメントの削除については、15-92 ページの「DROP ROLLBACK SEGMENT」を参照してください。
- UNDO_MANAGEMENT パラメータおよび UNDO_SUPPRESS_ERRORS パラメータの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

前提条件

ロールバック・セグメントを作成するには、CREATE ROLLBACK SEGMENT システム権限が必要です。

構文

create_rollback_segment::=



storage_clause: 17-50 ページの「[storage_clause](#)」を参照してください。

キーワードとパラメータ

PUBLIC

PUBLIC を指定すると、ロールバック・セグメントがパブリックとなり、すべてのインスタンスに対して使用可能になります。この句を省略した場合、ロールバック・セグメントはプライベートになり、インスタンスの初期化パラメータ ROLLBACK_SEGMENTS で指定したインスタンスに対してのみ使用可能になります。

rollback_segment

作成するロールバック・セグメントの名前を指定します。

TABLESPACE

TABLESPACE 句を使用して、ロールバック・セグメントが作成される表領域を識別します。この句を省略した場合、ロールバック・セグメントは SYSTEM 表領域に作成されます。

注意：

- 1つの表領域に複数のロールバック・セグメントを作成できます。一般に、複数のロールバック・セグメントがあると、パフォーマンスが向上します。
- 表領域にロールバック・セグメントを追加する場合、表領域は必ずオンラインである必要があります。
- ロールバック・セグメントを作成した場合、最初はオフライン状態になります。そのロールバック・セグメントを Oracle インスタンスによってトランザクション可能にする場合、ALTER ROLLBACK SEGMENT 文を使用してオンラインの状態にしてください。データベース起動時に自動的にオンライン状態にする場合、初期化パラメータ ROLLBACK_SEGMENTS の値にそのセグメントの名前を追加してください。
- Oracle は、頻繁にロールバック・セグメントにアクセスする必要があります。したがって、明示的か暗黙的（TABLESPACE 句の省略による）かにかかわらず、SYSTEM 表領域にはロールバック・セグメントを作成しないようにしてください。また、ロールバック・セグメントを含む表領域への過剰な競合を避けるには、表や索引などの他のオブジェクトを含まないようにし、エクステンツの割当ておよび割当て解除を最小にする必要があります。このためには、自動割当てを使用禁止にしたローカル管理の表領域、具体的には、UNIFORM 設定で EXTENT MANAGEMENT LOCAL 句を使用して作成された表領域に、ロールバック・セグメントを作成してください（AUTOALLOCATE 設定はサポートされていません）。

参照：

- 14-62 ページの「[CREATE TABLESPACE](#)」を参照してください。
- ロールバック・セグメントを作成および使用可能にする方法の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。

storage_clause

storage_clause によって、ロールバック・セグメントの特性を指定できます。

注意：

- *storage_clause* の OPTIMAL パラメータは、ロールバック・セグメントにのみ適用されるので、特に重要です。
 - *storage_clause* の PCTINCREASE パラメータは、CREATE ROLLBACK SEGMENT では指定できません。
-
-

参照： 17-50 ページの「[storage_clause](#)」を参照してください。

例

CREATE ROLLBACK SEGMENT の例 次の文は、システム表領域内にデフォルトの記憶域値でロールバック・セグメントを作成します。

```
CREATE ROLLBACK SEGMENT rbs_1
    TABLESPACE system;
```

前述の文は、次の文と同じ結果になります。

```
CREATE ROLLBACK SEGMENT rbs_2
    TABLESPACE system
    STORAGE
    ( INITIAL 10K
      NEXT 10K
      MAXEXTENTS UNLIMITED);
```

CREATE SCHEMA

用途

CREATE SCHEMA を使用すると、複数表およびビューを作成し、1つのトランザクションで複数の権限の付与を行うことができます。

CREATE SCHEMA 文を実行すると、挿入されている個々の文が実行されます。すべての文が正常に実行された場合、そのトランザクションがコミットされます。文の結果が1つでもエラーになった場合は、すべての文がロールバックされます。

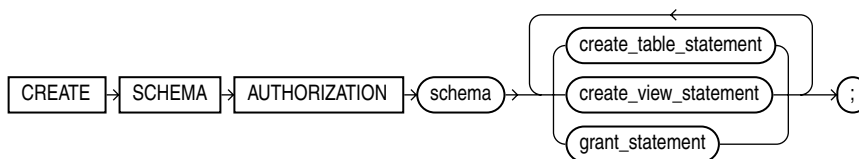
注意： この文では、実際にスキーマが作成されるわけではありません。ユーザーを作成すると、自動的にスキーマが作成されます (15-30 ページの「[CREATE USER](#)」を参照してください)。この文により、表およびビューとともにスキーマが作成され、複数のトランザクションで複数の SQL 文を発行しなくても、これらのオブジェクトに権限が付与されます。

前提条件

CREATE SCHEMA 文には、CREATE TABLE 文、CREATE VIEW 文および GRANT 文を含めることができます。CREATE SCHEMA 文を発行する場合は、挿入した文を発行するための権限が必要です。

構文

`create_schema::=`



キーワードとパラメータ

schema

スキーマの名前を指定します。スキーマ名は、Oracle ユーザー名と一致している必要があります。

create_table_statement

この CREATE SCHEMA 文の一部として発行する CREATE TABLE 文を指定します。この文の終わりには、セミコロン（またはその他の終了文字）を付けないでください。

参照： 14-6 ページの「[CREATE TABLE](#)」を参照してください。

create_view_statement

この CREATE SCHEMA 文の一部として発行する CREATE VIEW 文を指定します。この文の終わりには、セミコロン（またはその他の終了文字）を付けないでください。

参照： 15-37 ページの「[CREATE VIEW](#)」を参照してください。

grant_statement

この CREATE SCHEMA 文の一部として発行する GRANT *object_privileges* 文を指定します。この文の終わりには、セミコロン（またはその他の終了文字）を付けないでください。

参照： 16-31 ページの「[GRANT](#)」を参照してください。

CREATE SCHEMA 文は、Oracle でサポートされている完全な構文ではなく、標準 SQL で定義されている構文のみをサポートします。

CREATE TABLE、CREATE VIEW および GRANT の各文を指定する順序は重要ではありません。CREATE SCHEMA 文の中の文では、既存のオブジェクトまたは同じ CREATE SCHEMA 文の他の文で作成したオブジェクトを参照できます。

制限事項：*parallel_clause* 構文は、CREATE SCHEMA の CREATE TABLE 文で使用できますが、オブジェクトの作成時に並列度は使用されません。

参照： 14-43 ページの「[CREATE TABLE](#)」の「[parallel_clause](#)」を参照してください。

例

CREATE SCHEMA の例 次の文は、サンプルの注文入力ユーザー oe 用の oe という名前のスキーマ、表 `new_product`、ビュー `new_product_view` を作成し、サンプルの人事情報のユーザー `hr` に `new_product_view` に対する SELECT 権限を付与します。

```
CREATE SCHEMA AUTHORIZATION oe
  CREATE TABLE new_product
    (color VARCHAR2(10) PRIMARY KEY, quantity NUMBER)
  CREATE VIEW new_product_view
    AS SELECT color, quantity FROM new_product WHERE color = 'RED'
  GRANT select ON new_product_view TO hr;
```

CREATE SEQUENCE

用途

CREATE SEQUENCE 文を使用すると、**順序**を作成できます。順序とは、データベース・オブジェクトの 1 つで、これを使用して複数のユーザーが一意の整数を生成することができます。順序を使用した場合、主キー値が自動的に生成されます。

順序番号が生成されると、順序はトランザクションのコミットやロールバックとは無関係に増加していきます。2 人のユーザーが、同時に同一の順序を増加させると、ユーザーがそれぞれ順序番号を生成しているため、取得する順序番号間に違いが発生することもあります。他のユーザーが生成した順序番号は取得できません。あるユーザーが順序値を一度生成すると、他のユーザーがその順序を増加させたかどうかに関係なく、順序を生成したユーザーは引続きその値にアクセスすることができます。

順序番号は表から独立して生成されるため、1 つ以上の表に対して同一の順序を使用することができます。生成された順序番号が、最終的にロールバックされるトランザクションで使用されたため、個々の順序番号が連続していないように見える場合があります。また、他のユーザーが同一順序を使用していることを個々のユーザーが認識しない場合もあります。

順序が作成されると、SQL 文の中で CURRVAL 疑似列を使用してその値にアクセスできます（この場合、その順序の現在の値が戻ります）。また、NEXTVAL 疑似列を使用してもアクセスできます（この場合は、順序が増加され、新しい値が戻ります）。

参照：

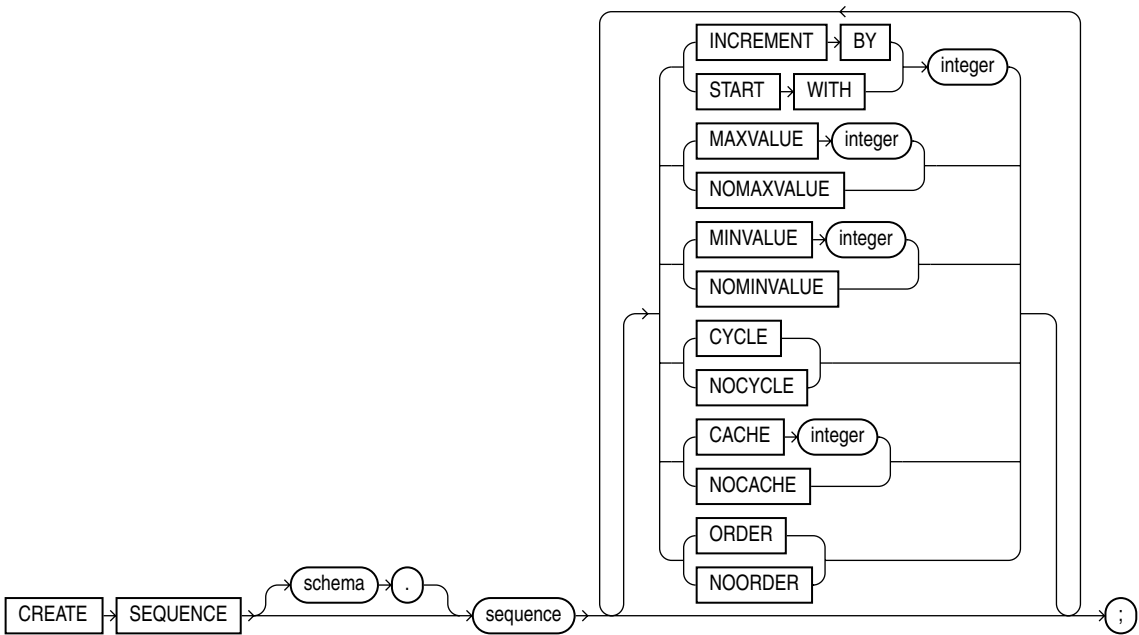
- CURRVAL および NEXTVAL の使用方法については、2-79 ページの「[疑似列](#)」を参照してください。
- 順序の使用については、2-80 ページの「[順序値の使用方法](#)」を参照してください。
- 順序の変更または削除の詳細は、8-119 ページの「[ALTER SEQUENCE](#)」または 16-2 ページの「[DROP SEQUENCE](#)」を参照してください。

前提条件

自分のスキーマ内に順序を作成する場合は、CREATE SEQUENCE 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

create_sequence::=



キーワードとパラメータ

schema

順序を含むスキーマを指定します。schema を省略した場合、自分のスキーマ内に順序が作成されます。

sequence

作成する順序の名前を指定します。

次の句のうちどれも指定しない場合は、1 から始まる昇順の順序が作成され、上限なしで 1 ずつ増加していきます。INCREMENT BY に -1 のみを指定した場合は、初期値を -1 として、下限なしで 1 ずつ減少していきます。

- **昇順で、無制限に増加する順序を作成する場合は**、MAXVALUE パラメータを省略するか、または NOMAXVALUE を指定します。降順の場合は、MINVALUE パラメータを省略するか、または NOMINVALUE を指定します。
- **昇順で、事前に定義した制限で停止する順序を作成する場合は**、MAXVALUE パラメータに値を指定します。降順の場合は、MINVALUE パラメータの値を指定します。NOCYCLE も指定します。順序が制限に達したときに順序番号をさらに生成した場合、エラーが発生します。
- **事前に定義した制限に達した後で初期値に戻る順序を作成する場合は**、MAXVALUE パラメータと MINVALUE パラメータの両方に値を指定します。CYCLE も指定します。MINVALUE を指定しない場合、デフォルトで NOMINVALUE（値 1）が設定されます。

順序パラメータ

INCREMENT BY 順序の番号間の増分間隔を指定します。この値は、0（ゼロ）以外の正の整数または負の整数になります。この値には、28 桁以内の値を指定できます。この値の絶対値は、MAXVALUE と MINVALUE の差未満である必要があります。この値が負の場合、順序は降順になります。この増分値が正の場合、順序は昇順になります。この句を省略した場合、デフォルトで増分間隔は 1 に設定されます。

START WITH 生成する順序番号の初期値を指定します。この句を指定した場合、順序の最小値より大きい値を初期値として昇順を開始することも、最大値よりも小さい値を初期値として降順を開始することもできます。昇順の場合、デフォルト値は順序の最小値になります。降順の場合、デフォルト値は順序の最大値になります。28 桁以内の整数値を指定できます。

注意： この値は、必ずしも、順序の最大値または最小値に達した後に、昇順で循環する順序が戻るときの値ではありません。

MAXVALUE 順序の最大値を指定します。28 桁以内の整数値を指定できます。MAXVALUE 値は、START WITH 以上で、かつ MINVALUE を超える値である必要があります。

NOMAXVALUE NOMAXVALUE は、順序の最大値を、昇順の場合は 10^{27} 、降順の場合は -1 に指定します。これはデフォルトです。

MINVALUE 順序の最小値を指定します。28 桁以内の整数値を指定できます。MINVALUE 値は、START WITH 以下で、かつ MAXVALUE 未満である必要があります。

NOMINVALUE NOMINVALUE は、順序の最小値を、昇順の場合は 1、降順の場合は -10^{26} に指定します。これはデフォルトです。

CYCLE CYCLE は、順序が最大値または最小値に達しても、引き続き値を生成することを示します。つまり、昇順の場合は、最大値に達すると最小値が生成されます。降順の場合は、最小値に達すると最大値が生成されます。

NOCYCLE NOCYCLE は、順序が最大値または最小値に達した場合は、それ以上の値を生成できないことを示します。これはデフォルトです。

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、この値は、そのサイクル内で生成される値の数未満である必要があります。指定したサイクル内で生成される順序番号の数を超える値はキャッシュできません。したがって、CACHE に指定できる値の最大値は、次の式で求められる値未満である必要があります。

$$(\text{CEIL}(\text{MAXVALUE} - \text{MINVALUE})) / \text{ABS}(\text{INCREMENT})$$

システム障害が発生すると、キャッシュされた順序の値のうち、コミットされた DML 文で使用されていないものはすべて失われます。したがって、失われる可能性がある値の数は、CACHE パラメータの値と等しくなります。

注意： Real Application Clusters 環境で順序を使用する場合は、パフォーマンスを向上するために、CACHE の設定を使用することをお勧めします。

NOCACHE NOCACHE は、順序の値が事前に割り当てられていないことを示します。

CACHE および NOCACHE の両方を省略した場合、デフォルトで 20 の順序番号がキャッシュされます。

ORDER ORDER は、要求どおりの順で順序番号を生成することを保証する場合に指定します。順序番号をタイムスタンプとして使用する場合に、この句を使用できます。通常、主キー生成用の順序については、順序どおりに生成するかどうかの保証は重要ではありません。

Real Application Clusters 環境で Oracle を使用する場合、ORDER は順序どおりの生成を保証する場合のみに必要です。排他モードの場合、順序番号は必ず順序どおりに生成されます。

NOORDER NOORDER は、要求どおりの順で順序番号を生成することを保証しない場合に指定します。これはデフォルトです。

例

CREATE SEQUENCE の例 次の文は、サンプル・スキーマ `oe` 内に順序 `orders_seq` を作成します。この順序は、注文表に行が追加されたときの注文 ID 番号となります。

```
CREATE SEQUENCE orders_seq
  START WITH      1000
  INCREMENT BY    1
  NOCACHE
  NOCYCLE;
```

最初に `orders_seq.nextval` を参照したときに 1000 が戻されます。次に参照した場合、1001 が戻されます。同様に、この後の各参照で、前回参照された値より 1 大きい値が戻されます。

CREATE SPFILE

用途

CREATE SPFILE を使用すると、クライアントの初期化パラメータ・ファイルから**サーバー・パラメータ・ファイル**を作成できます。サーバー・パラメータ・ファイルは、サーバーのみに存在し、データベースを起動するためにクライアントから呼ばれるバイナリのファイルです。

サーバー・パラメータ・ファイルを指定すると、個々のパラメータを永続的に変更できます。サーバー・パラメータ・ファイルを使用する場合、新しいパラメータ値を永続化する ALTER SYSTEM SET *parameter* 文を指定できます。新しい値は、現行のインスタンスのみでなく、その後で起動するすべてのインスタンスにおいて適用されます。以前のクライアントのパラメータ・ファイルでは、パラメータ値を永続的に変更できませんでした。

データベースの起動時にサーバー・パラメータ・ファイルを使用するには、CREATE SPFILE 文を使用して以前のテキストの初期化パラメータ・ファイルからサーバー・パラメータ・ファイルを作成しておく必要があります。

Real Application Clusters 環境のすべてのインスタンスは、同一のサーバー・パラメータ・ファイルを使用する必要があります。ただし、個々のインスタンスで 1 つのファイル内の同じパラメータで異なる設定が可能な場合もあります。インスタンス固有のパラメータ定義は、*SID.parameter = value* で指定します。*SID* にはインスタンス識別子を指定します。

サーバー・パラメータ・ファイルを使用したデータベースの起動方法は、作成したサーバー・パラメータ・ファイルをデフォルトで作成したか、または非デフォルトで作成したかによって異なります。サーバー・パラメータ・ファイルの使用方法については、13-88 ページの「**サーバー・パラメータ・ファイルの例**」を参照してください。

参照：

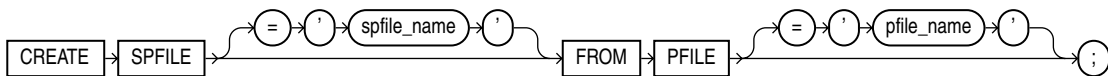
- バイナリのサーバー・パラメータ・ファイルから正規のテキストのパラメータ・ファイルを作成する場合の詳細は、13-56 ページの「**CREATE PFILE**」を参照してください。
- Oracle9i より前の初期化パラメータ・ファイル、および Oracle9i のサーバー・パラメータ・ファイルの詳細は、『Oracle9i データベース管理者ガイド』を参照してください。
- Real Application Clusters 環境でのサーバー・パラメータ・ファイルの使用については、『Oracle9i Real Application Clusters 管理』を参照してください。

前提条件

この文を実行するには、SYSDBA システム権限または SYSOPER システム権限が必要です。この文は、インスタンスの起動前と起動後のいずれかで実行できます。ただし、*spfile_name* を使用して、インスタンスがすでに起動済の場合は、この文で同じ *spfile_name* を指定できません。

構文

create_spfile::=



キーワードとパラメータ

spfile_name

この句を指定すると、作成するサーバー・パラメータ・ファイルの名前を指定できます。

- *spfile_name* を指定しない場合、プラットフォーム固有のデフォルトのサーバー・パラメータ・ファイル名が使用されます。*spfile_name* がサーバーにすでに存在する場合、そのファイルは上書きされます。デフォルトのサーバー・パラメータ・ファイルを使用する場合、ファイル名を参照せずにデータベースが起動されます。
- *spfile_name* を指定する場合、デフォルト以外のサーバー・パラメータ・ファイルを作成します。この場合、データベースを起動するときに、サーバー・パラメータ・ファイルを指す単一行の以前のパラメータ・ファイルを作成し、STARTUP コマンドで単一行のファイルを指定する必要があります。

参照：

- デフォルトおよびデフォルト以外のサーバー・パラメータ・ファイルを使用したデータベースの起動の詳細は、13-88 ページの「[サーバー・パラメータ・ファイルの例](#)」を参照してください。
- デフォルトのパラメータ・ファイル名は、オペレーティング・システム固有のドキュメントを参照してください。

pfile_name

サーバー・パラメータ・ファイルを作成する元になる以前の初期化パラメータ・ファイル名を指定します。

- *pfile_name* を指定する場合、パラメータ・ファイルはサーバーに存在する必要があります。ファイルがオペレーティング・システムのパラメータ・ファイルのデフォルト・ディレクトリに存在しない場合、フルパス名を指定する必要があります。
- *pfile_name* を指定しない場合、オペレーティング・システムのパラメータ・ファイルのデフォルト・ディレクトリからデフォルトのパラメータ・ファイル名が検索され、使用されます。そのディレクトリにファイルが存在しない場合、エラーが戻されます。

注意： Real Application Clusters 環境では、この文でサーバー・パラメータ・ファイルを指定して作成する前に、まず、すべてのインスタンスのパラメータ・ファイルを 1 つのファイルに結合する必要があります。この手順の実行の詳細は、『Oracle9i Real Application Clusters インストレーションおよび構成』を参照してください。

例

サーバー・パラメータ・ファイルの例 次の例は、クライアントの初期化パラメータ・ファイル `t_init1.ora` からデフォルトのサーバー・パラメータ・ファイルを作成します。

```
CREATE SPFILE FROM PFILE = 't_init1.ora';
```

注意： 通常、オペレーティング・システムのパラメータ・ファイルには、フルパスのファイル名を指定する必要があります。パスについては、ご使用のオペレーティング・システムの Oracle マニュアルを参照してください。

デフォルトのサーバー・パラメータ・ファイルを作成する場合、その後のデータベースの起動は、次のように `PFILE` パラメータなしで `SQL*Plus` のコマンド `STARTUP` を実行し、サーバー・パラメータ・ファイルを使用します。

```
STARTUP
```

次の例は、クライアントの初期化パラメータ・ファイル `t_init1.ora` からデフォルト以外のサーバー・パラメータ・ファイル `s_params.ora` を作成します。

```
CREATE SPFILE = 's_params.ora' FROM PFILE = 't_init1.ora';
```


デフォルト以外のサーバー・パラメータ・ファイルを作成する場合、次の単一行を含む以前のパラメータ・ファイルを最初に作成し、その後でデータベースを起動します。

```
spfile = 's_params.ora'
```

このパラメータ・ファイル名は、ご使用のオペレーティング・システムのネーミング規則に従う必要があります。その後、**STARTUP** コマンドで単一行のパラメータ・ファイルを使用します。次の例では、単一行のパラメータ・ファイルの名前が `new_param.ora` であるとした場合のデータベースの起動方法を示します。

```
STARTUP PFILE=new_param.ora
```

SQL 文 : CREATE SYNONYM ~ CREATE TRIGGER

この章では、次の SQL 文について説明します。

- CREATE SYNONYM
- CREATE TABLE
- CREATE TABLESPACE
- CREATE TEMPORARY TABLESPACE
- CREATE TRIGGER

CREATE SYNONYM

用途

CREATE SYNONYM 文を使用すると、**シノニム**を作成できます。シノニムとは、表、ビュー、順序、プロシージャ、ストアド・ファンクション、パッケージ、マテリアライズド・ビュー、Java クラス・スキーマ・オブジェクトおよび別のシノニムに付ける別名です。

シノニムによって、データの独立性および位置の透過性を実現できます。シノニムを使用した場合、どのユーザーが表やビューを所有しているか、どのデータベースに表やビューが格納されているかに関係なく、アプリケーションは機能します。

表 14-1 に、シノニムを参照できる SQL 文を示します。

表 14-1 シノニムの使用方法

DML 文	DDL 文
SELECT	AUDIT
INSERT	NOAUDIT
UPDATE	GRANT
DELETE	REVOKE
EXPLAIN PLAN	COMMENT
LOCK TABLE	

参照： シノニムの概要については、『Oracle9i データベース概要』を参照してください。

前提条件

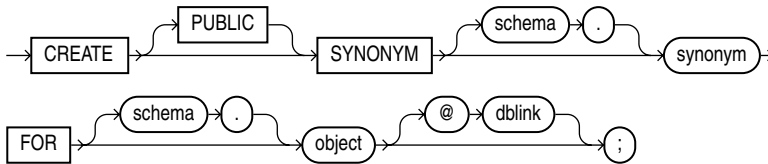
自分のスキーマ内にプライベート・シノニムを作成する場合は、CREATE SYNONYM システム権限が必要です。

他のユーザーのスキーマ内にプライベート・シノニムを作成する場合は、CREATE ANY SYNONYM システム権限が必要です。

パブリック・シノニムを作成する場合は、CREATE PUBLIC SYNONYM システム権限が必要です。

構文

create_synonym::=



キーワードとパラメータ

PUBLIC

PUBLIC を指定して、パブリック・シノニムを作成します。パブリック・シノニムには、すべてのユーザーがアクセスできます。

オブジェクトの先頭にスキーマ名が指定されていない場合、およびオブジェクトの後にデータベース・リンクが指定されていない場合は、オブジェクトの参照を変換するときのみ、パブリック・シノニムが使用されます。

この句を省略した場合、シノニムはプライベートとなり、定義しているスキーマ内にかぎりアクセスできます。プライベート・シノニム名は、スキーマ内で一意である必要があります。

schema

シノニムを含むスキーマを指定します。*schema* を省略した場合、自分のスキーマ内にシノニムが作成されます。**PUBLIC** を指定した場合、スキーマは指定できません。

synonym

作成するシノニムの名前を指定します。

注意： シノニム名の最大長は 32 バイトです。30 バイトを超える名前は、Java の機能にのみ使用できます。30 バイトを超える名前を指定した場合、名前は暗号化され、暗号化された表現でデータ・ディクショナリに格納されます。実際の暗号にはアクセスできません。また、元の指定とデータ・ディクショナリの表現のどちらも、シノニム名としては使用できません。

FOR 句

シノニムを作成するオブジェクトを指定します。オブジェクトに *schema* を指定しなかった場合、そのスキーマ・オブジェクトは自分のスキーマ内にあるとみなされます。スキーマ・オブジェクトには、次のものを指定できます。

- 表またはオブジェクト表
- ビューまたはオブジェクト・ビュー
- 順序
- ストアド・プロシージャ、ファンクションまたはパッケージ
- マテリアライズド・ビュー
- Java クラス・スキーマ・オブジェクト
- シノニム

スキーマ・オブジェクトは、現在存在している必要はなく、スキーマ・オブジェクトへのアクセス権限も必要ありません。

制限事項：

- スキーマ・オブジェクトは、パッケージに入れることはできません。

オブジェクト型にはシノニムを作成できません。

dblink

データベース・リンクを完全に指定するか、またはデータベース・リンクの一部を指定する場合、スキーマ・オブジェクトが格納されているリモート・データベース上でそのスキーマ・オブジェクトのシノニムを作成することができます。*dblink* を指定して、*schema* を省略した場合、シノニムは、データベース・リンクで指定したスキーマ内のオブジェクトを参照します。リモート・データベースでは、オブジェクトを定義しているスキーマを指定することをお勧めします。

dblink を省略した場合、オブジェクトがローカル・データベース上にあるものとみなされます。

制限事項：Java クラス・シノニムには *dblink* を指定できません。

参照：

- データベース・リンクの参照方法の詳細は、2-113 ページの「[リモート・データベース内のオブジェクトの参照](#)」を参照してください。
- データベース・リンクの作成方法の詳細は、12-33 ページの「[CREATE DATABASE LINK](#)」を参照してください。

例

CREATE SYNONYM の例 次の文は、スキーマ hr 内の表 locations に対してシノニム offices を定義します。

```
CREATE SYNONYM offices
FOR hr.locations;
```

次の文は、リモート・データベース sales 上のスキーマ hr 内の employees 表に対してパブリック・シノニムを作成します。

```
CREATE PUBLIC SYNONYM employees
FOR hr.employees@sales;
```

別のスキーマ内に実表が定義されている場合は、実表と同じ名前をシノニムに指定することもできます。

シノニムの変換例 Oracle は、オブジェクトの参照を、パブリック・シノニム・レベルで変換する前に、スキーマ・レベルで変換しようとします。たとえば、スキーマ oe と hr のどちらにも locations という名前の表が存在し、ユーザー SYSTEM が oe.locations にパブリック・シノニム locations を作成する場合は、次の文になります。

```
CREATE PUBLIC SYNONYM locations FOR oe.locations;
```

ユーザー hr が次の文を発行すると、hr.locations から行が戻されます。

```
SELECT * FROM locations;
```

oe.locations から行を取り出すには、ユーザー hr は、locations の前にスキーマ名を指定する必要があります。

```
SELECT * FROM oe.locations;
```

ユーザー pm のスキーマに locations という名前のオブジェクトが存在しない場合、pm は、パブリック・シノニム locations を使用して、oe のスキーマ内の locations 表にアクセスできます。

```
SELECT * FROM locations;
```

CREATE TABLE

用途

CREATE TABLE 文を使用すると、次の型の表を作成できます。

- **リレーショナル表**。ユーザー・データを格納する基本構造です。
- **オブジェクト表**。列の定義にオブジェクト型を使用する表です。オブジェクト表とは、特定の型のオブジェクト・インスタンスを格納するように明示的に定義された表です。

オブジェクト型を作成しておき、リレーショナル表の作成時に列の中でそのオブジェクト型を使用することもできます。

参照： オブジェクト作成の詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』、『Oracle9i データベース管理者ガイド』および 15-3 ページの「[CREATE TYPE](#)」を参照してください。

問合せを指定しない場合、データを含まない表が作成されます。INSERT 文を使用した場合、表に行を追加できます。表を作成した後、ALTER TABLE 文で ADD 句を指定すると、追加する列、パーティションおよび整合性制約を定義できます。ALTER TABLE 文で MODIFY 句を指定すると、既存の列またはパーティションの定義を変更できます。

前提条件

自分のスキーマ内に**リレーショナル表**を作成する場合は、CREATE TABLE システム権限が必要です。他のユーザーのスキーマ内に表を作成する場合は、CREATE ANY TABLE システム権限が必要です。また、表を定義しているスキーマの所有者は、表を格納するため表領域への割当て制限または UNLIMITED TABLESPACE システム権限が必要です。

これらの表権限に加え、**オブジェクト表**またはオブジェクト型の列を持つリレーショナル表を作成する場合は、表の所有者に、表が参照するすべての型にアクセスするための EXECUTE オブジェクト権限が付与されているか、または EXECUTE ANY TYPE システム権限が付与されている必要があります。これらの権限は、ロールを介して取得するのではなく、明示的に付与される必要があります。

さらに、表の所有者が表へのアクセス権限を他のユーザーに付与する場合、所有者には、参照する型に対する GRANT OPTION 付きの EXECUTE 権限、または ADMIN OPTION 付きの EXECUTE ANY TYPE システム権限が必要です。これらの権限を持っていない場合、表の所有者は、表へのアクセス権限を他のユーザーに付与できません。

一意または主キー制約を有効にする場合は、表に索引を作成するための権限が必要です。Oracle は、その表を含むスキーマにある一意または主キー列に索引を作成するので、この権限が必要になります。

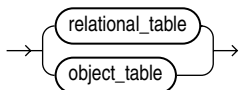
外部表を作成する場合は、外部データが存在するディレクトリに対する READ オブジェクト権限が必要です。

参照：

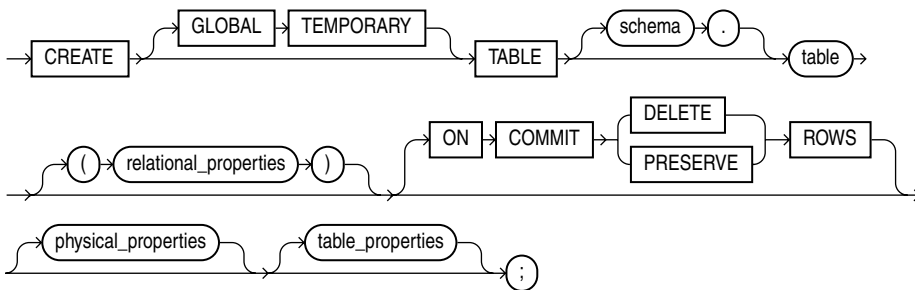
- 12-58 ページの「[CREATE INDEX](#)」を参照してください。
- 型を使用する表の作成に必要な権限については、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

構文

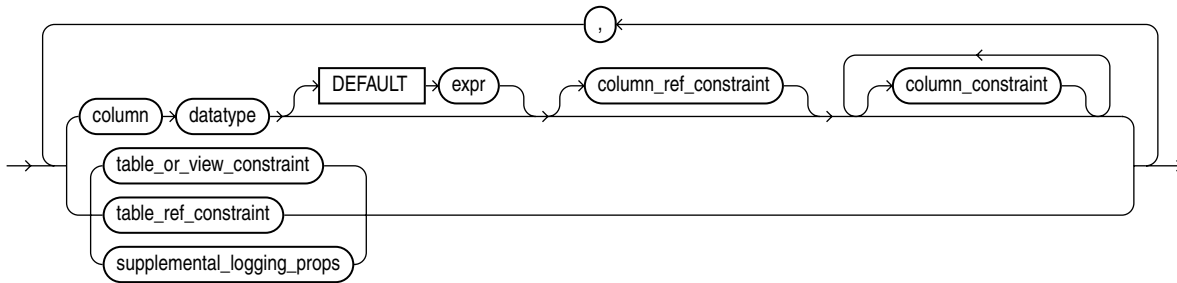
create_table::=



relational_table::=

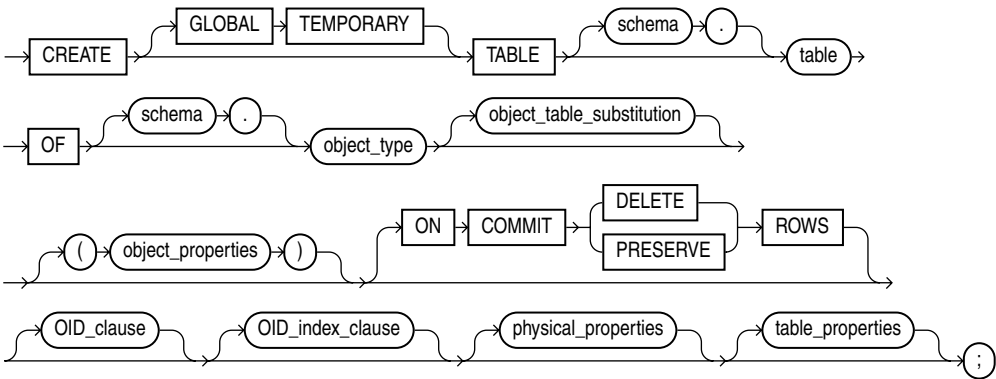


relational_properties::=

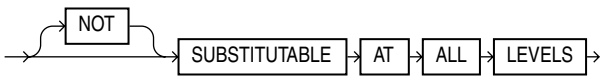


CREATE TABLE

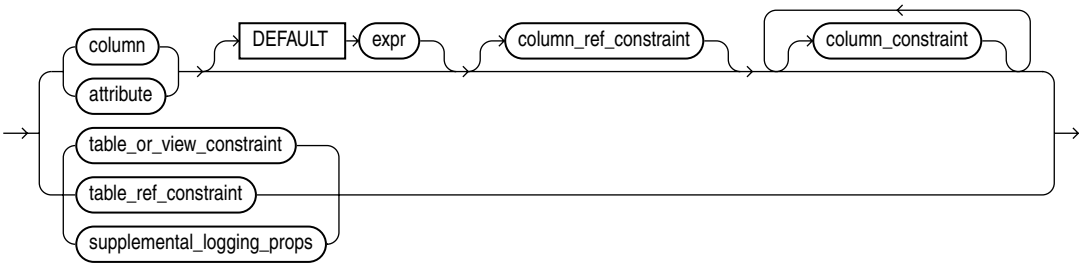
object_table::=



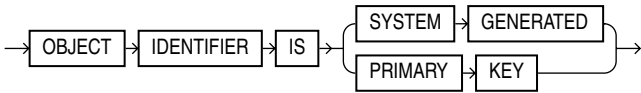
object_table_substitution::=



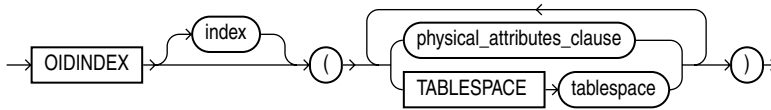
object_properties::=



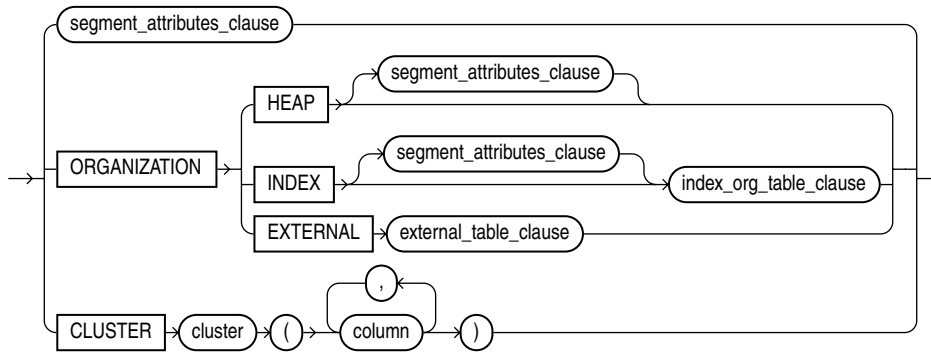
OID_clause::=



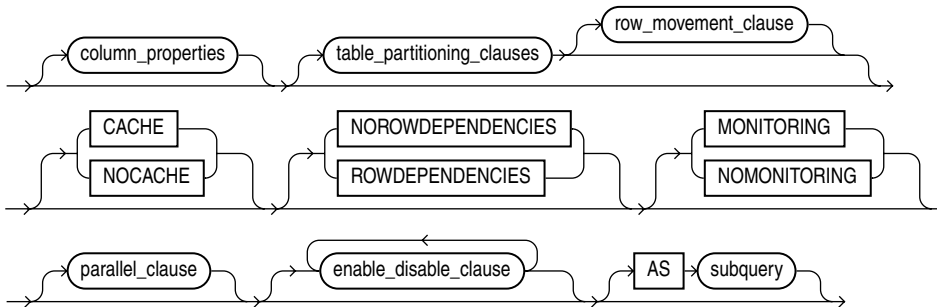
OID_index_clause::=



physical_properties::=

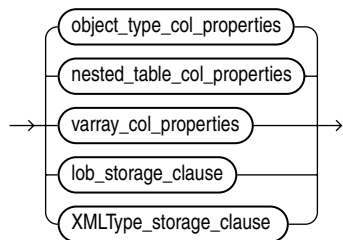


table_properties::=

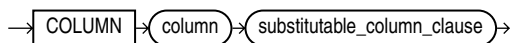


CREATE TABLE

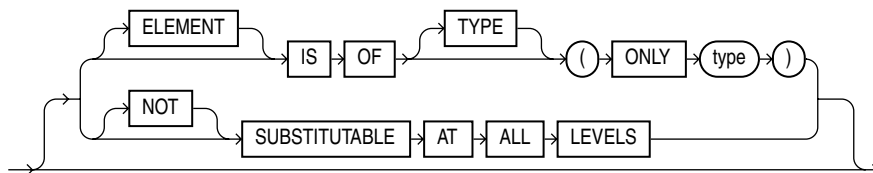
column_properties::=



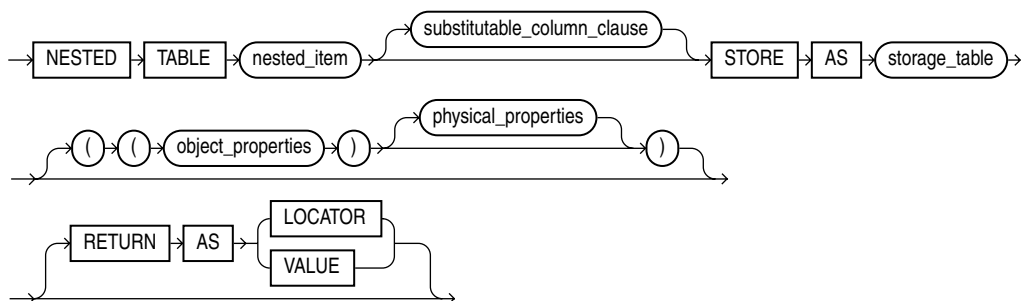
object_type_col_properties::=



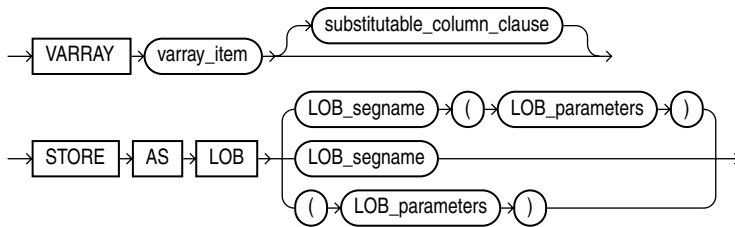
substitutable_column_clause::=



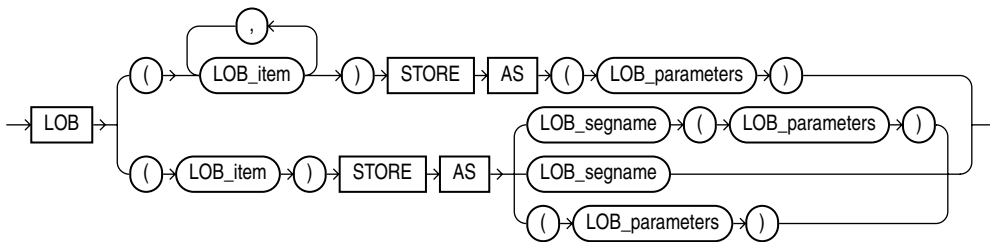
nested_table_col_properties::=



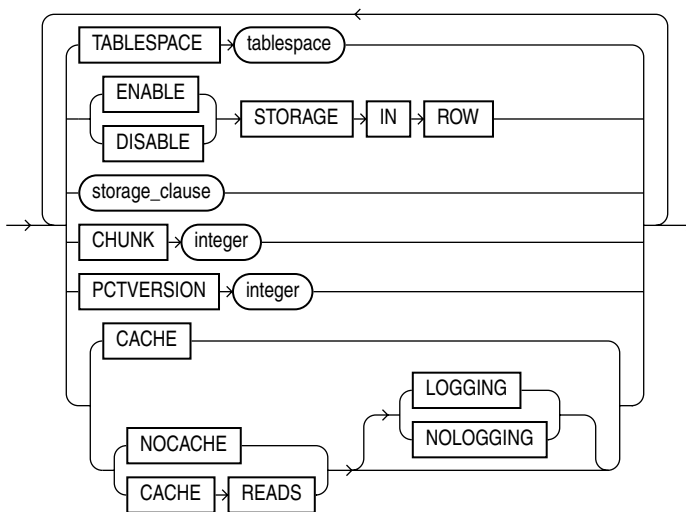
varray_col_properties::=

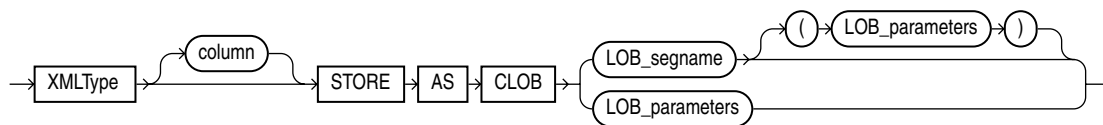


LOB_storage_clause::=



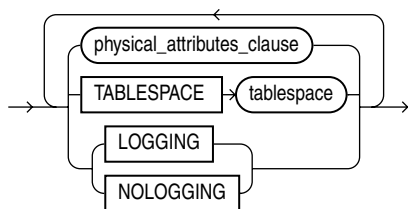
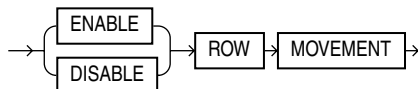
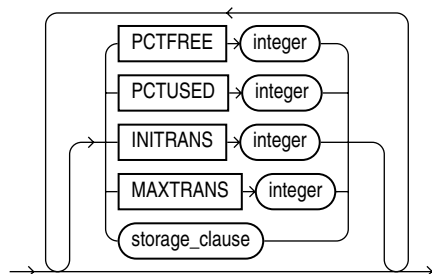
LOB_parameters::=



xmltype_storage_clause::=

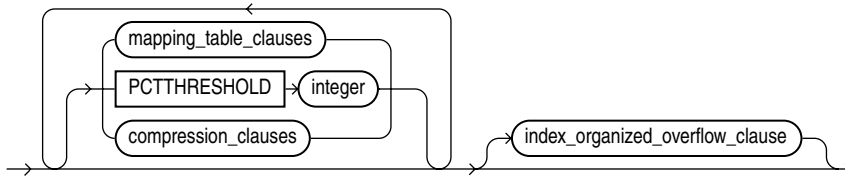
subquery::= 17-4 ページの「[SELECT](#)」を参照してください。

table_or_view_constraint、**column_constraint**、**table_ref_constraint**、**column_ref_constraint**、**constraint_state**:
11-72 ページの「[constraint_clause](#)」を参照してください。

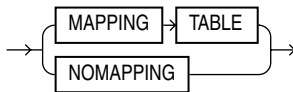
segment_attributes_clause::=**row_movement_clause::=****physical_attributes_clause::=**

storage_clause: 17-50 ページの「[storage_clause](#)」を参照してください。

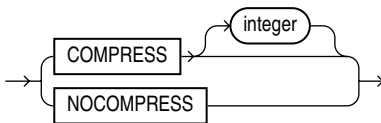
index_org_table_clause::=



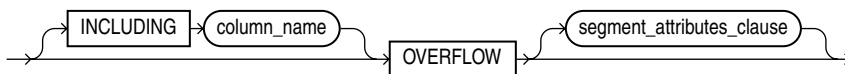
mapping_table_clauses::=



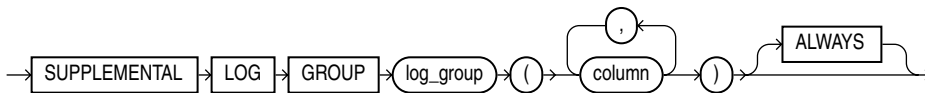
compression_clauses::=



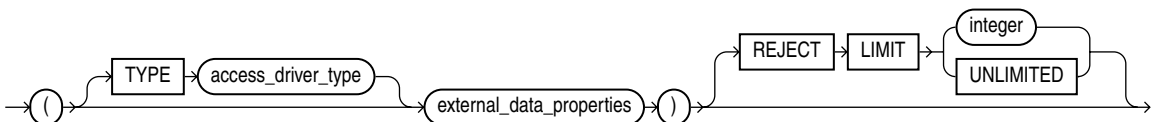
index_org_overflow_clause::=



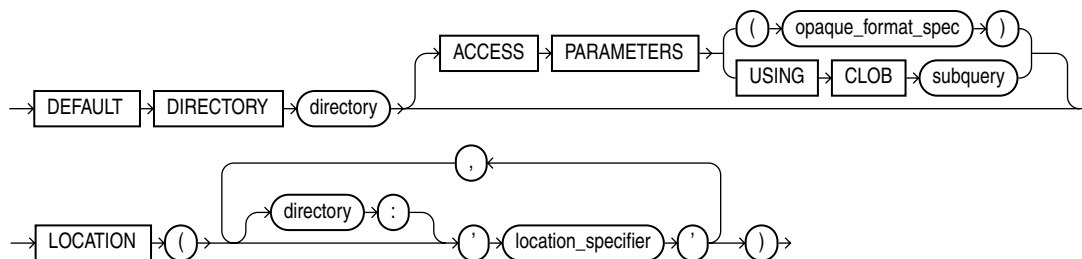
supplemental_logging_props::=



external_table_clause::=

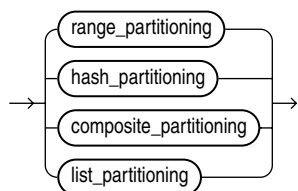


external_data_properties::=

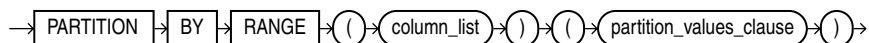


opaque_format_spec: *opaque_format_spec* への値の指定方法の詳細は、『Oracle9i データベース・ユーティリティ』を参照してください。

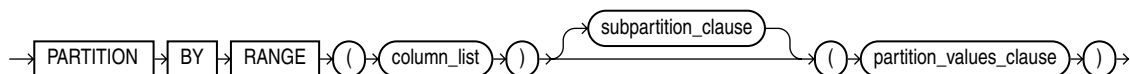
table_partitioning_clauses::=



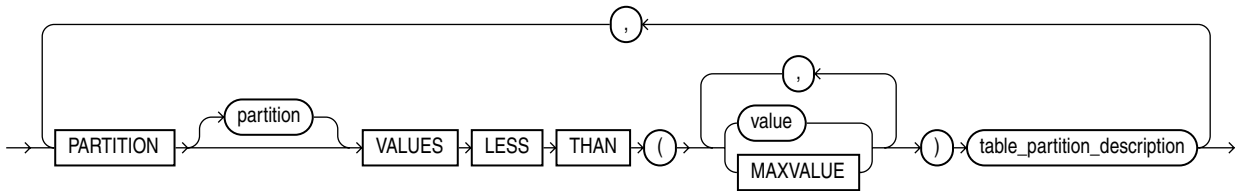
range_partitioning::=



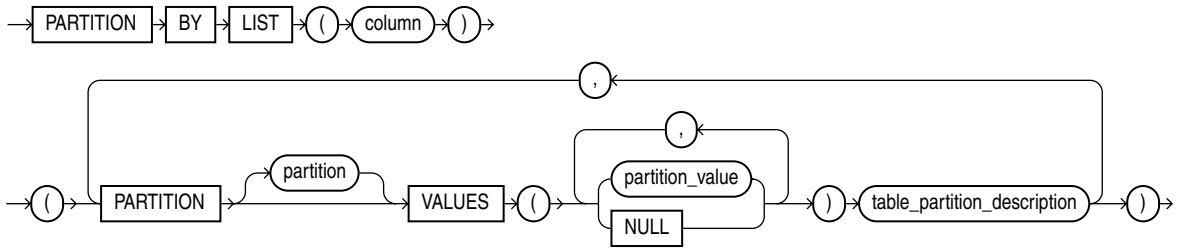
composite_partitioning::=



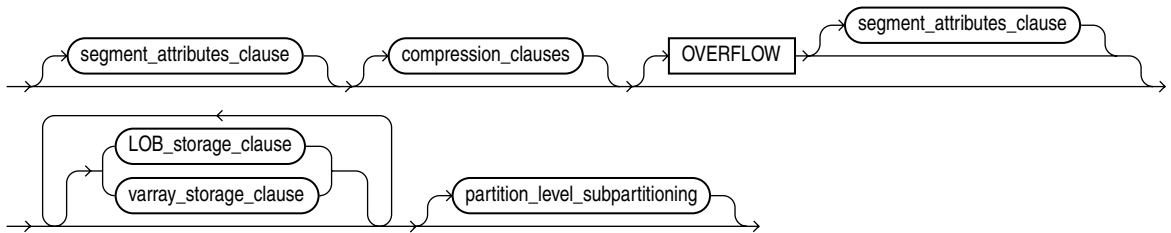
partition_values_clause::=



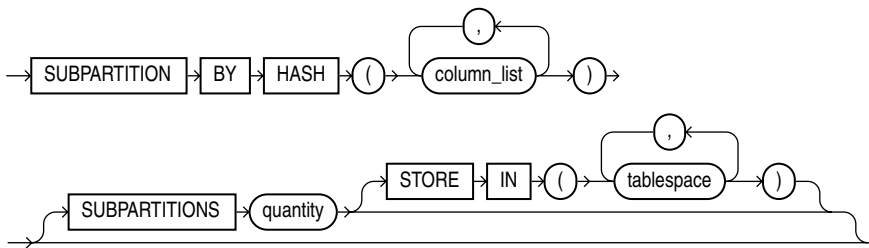
list_partitioning::=



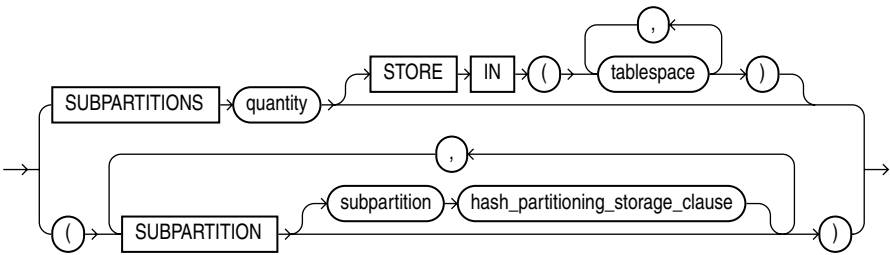
table_partition_description::=



subpartition_clause::=



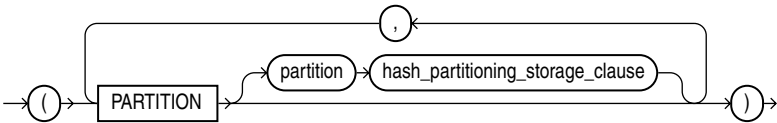
partition_level_subpartition::=



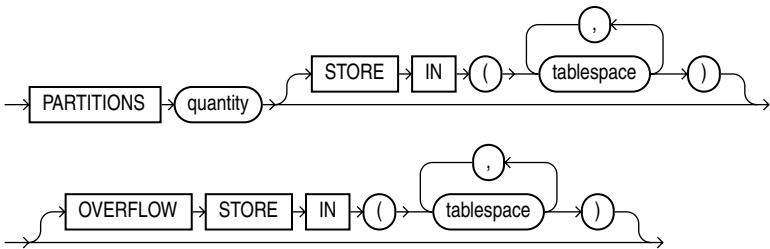
hash_partitioning::=



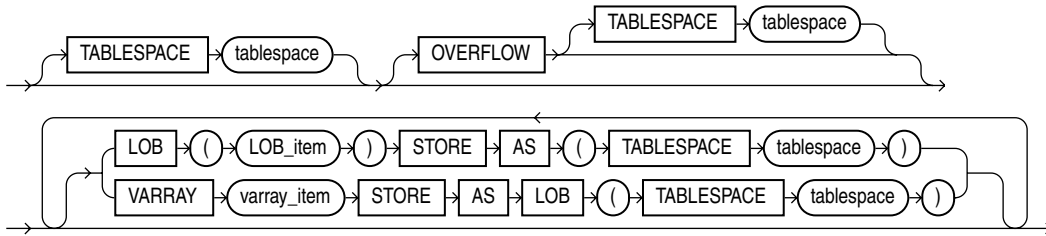
individual_hash_partitions::=



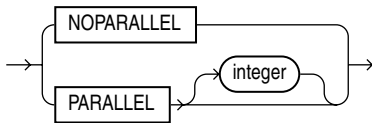
hash_partitions_by_quantity::=



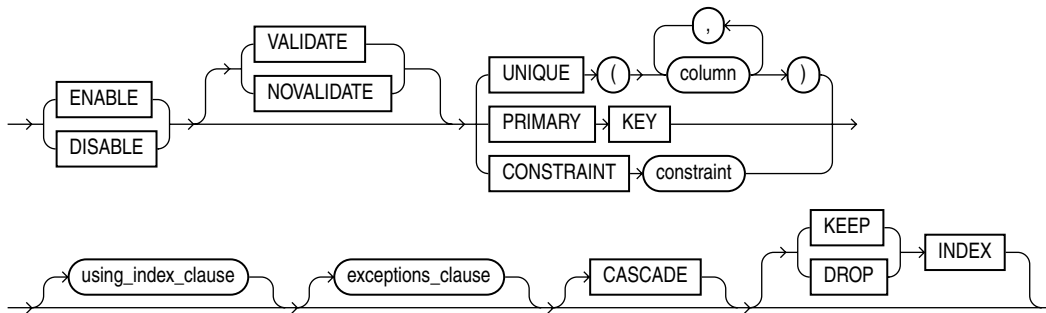
hash_partitioning_storage::=



parallel_clause::=

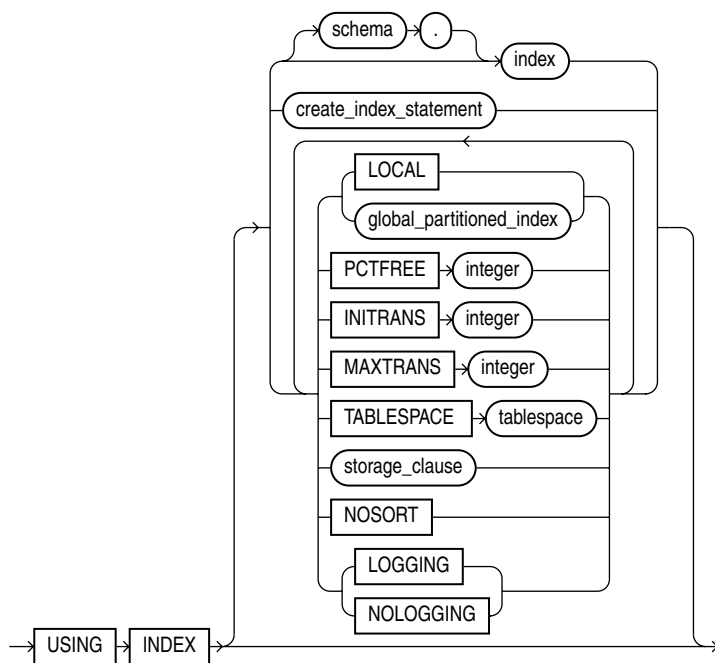


enable_disable_clause::=

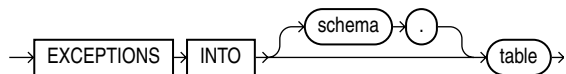


CREATE TABLE

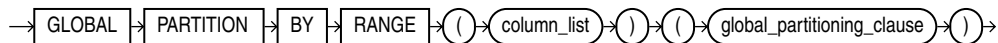
using_index_clause::=



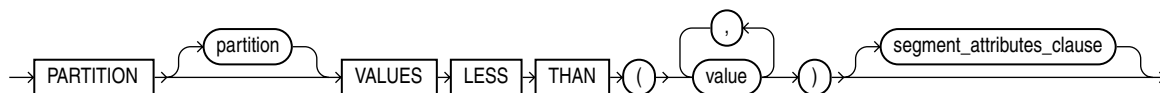
exceptions_clause::=



global_partitioned_index::=



global_partitioning_clause::=



キーワードとパラメータ

GLOBAL TEMPORARY

GLOBAL TEMPORARY を指定して、表が一時的で、その定義がすべてのセッションで参照できることを示します。一時表のデータは、データを表に挿入するセッションでのみ参照できます。

一時表は、標準表の定義に準拠する定義を持ちますが、**セッション固有**または**トランザクション固有**データのいずれかを含みます。データがセッション固有かトランザクション固有かは、ON COMMIT キーワード（後述）で指定します。

参照： 一時表の詳細は、『Oracle9i データベース概要』を参照してください。

制限事項：

- 一時表は、パーティション化、索引構成化またはクラスタ化できません。
- 一時表には、参照整合性（外部キー）制約を指定できません。
- 一時表は、ネストした表または VARRAY 型の列を含むことはできません。
- *LOB_storage_clause* の TABLESPACE、*storage_clause*、LOGGING または NOLOGGING、MONITORING または NOMONITORING あるいは *LOB_index_clause* は指定できません。
- 一時表にパラレル DML およびパラレル問合せはサポートされていません（パラレル・ヒントは無視されます。*parallel_clause* を指定すると、エラーが戻されます）。
- *segment_attributes_clause*、*nested_table_col_properties* または *parallel_clause* は指定できません。
- 一時表での分散トランザクションはサポートされていません。

schema

表を含むスキーマを指定します。*schema* を省略した場合、自分のスキーマ内に表が作成されます。

table

作成する表（またはオブジェクト表）の名前を指定します。

object_table

OF 句によって、明示的に *object_type* 型の**オブジェクト表**を作成できます。オブジェクト表の各列は、*object_type* 型の最上位の属性に対応します。各行には、オブジェクト・インスタンスが入り、また各インスタンスには、行の挿入時に一意のシステム生成オブジェクト識別子 (OID) が割り当てられます。*schema* を省略した場合、オブジェクト表が自分のスキーマ内に作成されます。

オブジェクト表に存在するオブジェクトを参照できます。

object_table_substitution

object_table_substitution 句を使用すると、サブタイプに対応する行オブジェクトの、このオブジェクト表への挿入を許可するかどうかを指定できます。

NOT SUBSTITUTABLE AT ALL LEVELS NOT SUBSTITUTABLE AT ALL LEVELS を指定すると、作成するオブジェクト表は置換できなくなります。また、置換は、すべての埋込みオブジェクト属性および埋込みのネストした表と配列の要素には使用禁止です。デフォルトは、SUBSTITUTABLE AT ALL LEVELS です。

relational_properties

relational_properties 句を使用すると、リレーショナル表のコンポーネントを指定できます。

column

表の列の名前を指定します。

AS *subquery* を指定すると、索引構成表を作成しないかぎり、*column* および *datatype* を省略できます。索引構成表の作成時に AS *subquery* を指定する場合は、*column* を指定し、*datatype* を省略する必要があります。

表の列の絶対最大数は 1000 です。ただし、オブジェクト表（または、オブジェクトの列、ネストした表、VARRAY または REF 型のリレーショナル表）を作成する場合、制限の 1000 列までをカウントする有効な非表示列を作成して、ユーザー定義型の列をリレーショナル列にマップします。

datatype

列のデータ型を指定します。

注意： 次の場合は、*datatype* は省略できます。

- AS *subquery* を指定する場合（索引構成表を作成して AS *subquery* を指定する場合は、データ型を省略する必要があります。）
 - 文が、参照整合性制約で外部キーとして列を指定する場合（Oracle では、参照整合性制約の参照キーに対応する列のデータ型が列に自動的に割り当てられます。）
-
-

制限事項：

- ハッシュ・パーティション化された索引構成表に LOB 列または VARRAY 型の列を指定できません。パーティション化されていない索引構成表またはレンジ・パーティション化された索引構成表に対するデータ型は制限されていません。
- ROWID 型の列を指定することはできますが、それらの列の値が有効な行 ID であることは保証されません。

参照： Oracle が提供するデータ型については、2-2 ページの「[データ型](#)」を参照してください。

DEFAULT

DEFAULT 句によって、後続の INSERT 文が列の値を省略した場合に値が列に割り当てられるように指定できます。式のデータ型は、列のデータ型と一致する必要があります。列には、この式が入る長さが必要です。

DEFAULT 式には、リテラル引数、列の参照またはネストしたファクションの起動を戻さない、任意の SQL ファンクションを含めることができます。

制限事項： DEFAULT 式に、PL/SQL ファンクション、他の列（LEVEL、PRIOR および ROWNUM 疑似列）への参照または完全には指定されていない日付定数を指定することはできません。

参照： *expr* の構文については、4-2 ページの「[SQL 式](#)」を参照してください。

table_ref_constraint | column_ref_constraint

これらの句を使用すると、REF 型の列について詳細に指定できます。これらの句の違いは、表レベルで *table_ref* を指定することであり、定義する REF 型の列または属性を識別する必要があります。REF 型の列または属性を識別した後、*column_ref* を指定してください。

参照： これらの制約の構文および詳細は、11-72 ページの「[constraint_clause](#)」を参照してください。

column_constraint

column_constraint を使用して、整合性制約を列定義の一部として定義します。

オブジェクト型の列のスカラー属性に、一意制約、主キー制約および参照制約を作成できます。また、オブジェクト型の列の NOT NULL 制約、オブジェクト型の列またはオブジェクト型の列の属性を参照するチェック制約も作成できます。

参照： これらの制約の構文および詳細は、11-72 ページの「[constraint_clause](#)」を参照してください。

table_or_view_constraint

table_or_view_constraint を使用して、整合性制約を表定義の一部として定義できます。

注意： DEFERRABLE 以外の主キー制約を索引構成表に指定してください。

参照： *table_or_view_constraint* 構文の詳細は、11-72 ページの「[constraint_clause](#)」を参照してください。

object_properties

オブジェクト表のプロパティは、基本的にリレーショナル表と同じです。ただし、列を指定するかわりに、オブジェクトの属性を指定します。

attribute には、オブジェクト内の項目の修飾した列名を指定します。

ON COMMIT

ON COMMIT 句は、一時表を作成する場合のみに適用されます。この句は、一時表のデータがトランザクションまたはセッションの存続期間中保持されるかどうかを指定します。

DELETE ROWS トランザクション固有の一時表に DELETE ROWS を指定します（デフォルト）。各コミット後に表を切り捨てます（すべての行を削除します）。

PRESERVE ROWS セッション固有の一時表に対して、PRESERVE ROWS を指定します。セッション終了時に表を切り捨てます（すべての行を削除します）。

OID_clause

OID_clause によって、オブジェクト表のオブジェクト識別子（OID）がシステム生成されるか、表の主キーを基に作成されるかを指定できます。デフォルトは SYSTEM GENERATED です。

制限事項：

- 主キー制約を表に指定していないと、OBJECT IDENTIFIER IS PRIMARY KEY は指定できません。
- この句は、ネストした表には指定できません。

注意： 主キー OID はローカルで（グローバルである必要はありません）一意です。グローバルで一意の識別子が必要な場合は、主キーがグローバルで一意であることを確認してください。

OID_index_clause

この句は、*OID_clause* を SYSTEM GENERATED として指定している場合のみに適用されません。非表示のオブジェクト識別子列に索引を指定します。また、任意に記憶特性を指定します。

index には、非表示のシステム生成オブジェクト識別子列の索引の名前を指定します。*index* を省略すると、名前が生成されます。

physical_properties

物理プロパティは、エクステンツおよびセグメントの処理、および表の記憶特性に関係します。

segment_attributes_clause

physical_attributes_clause *physical_attributes_clause* は、PCTFREE、PCTUSED、INITRANS および MAXTRANS パラメータの値、および表の記憶特性を指定します。

- 非パーティション表の場合、指定した各パラメータおよび記憶特性は、表に関連付けられたセグメントの実際の物理属性となります。
- パーティション表の場合、指定した各パラメータおよび記憶特性は、パーティションを作成する文の PARTITION 句で明示的にオーバーライドしないかぎり、この CREATE 文（および後続の ALTER TABLE ... ADD PARTITION 文）で指定されたすべてのパーティションに関連付けられたセグメントのデフォルトの物理属性となります。

PCTFREE integer 表、オブジェクト表の OID 索引、パーティションそれぞれのデータ・ブロックの中で、表の行に対して今後行われる更新用に確保する領域が占める割合（パーセント）を指定します。PCTFREE の値は、0 ～ 99 の値にする必要があります。値に 0 を指定した場合は、ブロック全体が一杯になるまで新しい行を挿入できます。デフォルト値は 10 です。10 を指定した場合、既存の行に対して行われる更新用に各ブロックの 10% が確保されるため、各ブロックでは最大 90% まで表に新しい行を挿入できます。

PARTITION 記述句の中での PCTFREE の機能と、クラスタ、索引、マテリアライズド・ビュー、マテリアライズド・ビュー・ログの作成および変更を行う文の中での PCTFREE の機能は同じです。PCTFREE と PCTUSED の組合せによって、新しい行を既存のデータ・ブロックと新しいデータ・ブロックのどちらに挿入するかが決まります。

PCTUSED integer 使用領域のうち、表、オブジェクト表 OID 索引、索引構成表のオーパーフロー・データ・セグメントのデータ・ブロックごとに、Oracle によって確保される最小限の使用領域の割合（パーセント）を指定します。ブロックは、使用領域が PCTUSED の値を下回ると、行挿入の対象となります。PCTUSED は 0 ～ 99 までの正の整数で指定し、デフォルト値は 40 です。

PARTITION 記述句の中での PCTUSED の機能と、クラスタ、マテリアライズド・ビュー、マテリアライズド・ビュー・ログの作成および変更を行う文の中での PCTUSED の機能は同じです。

PCTUSED は、索引構成表 (ORGANIZATION INDEX) には無効な表記憶特性です。

PCTFREE および PCTUSED の合計は 100 以下である必要があります。PCTFREE と PCTUSED をともに使用して、表内の領域を効果的に利用できます。

参照： PCTUSED および PCTFREE の各値によるパフォーマンスへの効果については、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。

INITRANS integer 表、オブジェクト表 OID 索引、パーティション、LOB の索引セグメントまたはオーバーフロー・データ・セグメントに割り当てられた各データ・ブロック内の初期トランザクション・エントリ数を指定します。この値の範囲は 1 ～ 255 までで、デフォルト値は 1 です。通常、このデフォルトの INITRANS 値を変更する必要はありません。

ブロックを更新するトランザクションごとに、ブロックのトランザクション・エントリが必要です。トランザクション・エントリのサイズは、ご使用のオペレーティング・システムによって異なります。

このパラメータを指定した場合、最小数の同時実行トランザクションでブロックを更新することができます。さらに、トランザクション・エントリを動的に割り当てるときのオーバーヘッドを回避できます。

INITRANS パラメータは、PARTITION 記述、クラスタ、索引、マテリアライズド・ビューおよびマテリアライズド・ビュー・ログの中では、表の場合と同じ働きをします。クラスタまたは索引の場合、INITRANS の最小値およびデフォルト値は 1 ではなく 2 です。

MAXTRANS integer 表、オブジェクト表 OID 索引、パーティション、LOB の索引セグメント、索引構成オーバーフロー・データ・セグメントに割り当てられたデータ・ブロックを更新できる同時実行トランザクションの最大数を指定します。この制限は問合せには適用されません。この値の範囲は 1 ～ 255 までで、デフォルト値はデータ・ブロック・サイズのファンクションとなります。MAXTRANS 値は、変更せずにデフォルトのまま使用してください。

ブロックを同時に更新するトランザクション数が INITRANS 値を超えると、MAXTRANS 値を超えるまで、またはブロックの空き領域がなくなるまで、そのブロックにトランザクション・エントリが動的に割り当てられます。

MAXTRANS パラメータは、PARTITION 記述、クラスタ、索引、マテリアライズド・ビューおよびマテリアライズド・ビュー・ログの中では、表の場合と同じ働きをします。

storage_clause *storage_clause*によって、表、オブジェクト表 OID 索引、パーティション、LOB のデータ・セグメント、LOB の索引セグメントまたは索引構成表のオーバーフロー・データ・セグメントの記憶特性を指定できます。この句は、大規模な表のパフォーマンスに影響します。記憶域は、追加領域の動的割当てを最小限に抑えるように割り当てられます。

参照： 17-50 ページの「[storage_clause](#)」を参照してください。

TABLESPACE Oracle が、表、オブジェクト表 OID 索引、パーティション、LOB のデータ・セグメント、LOB の索引セグメントまたは索引構成表のオーバーフロー・データ・セグメントを作成する表領域を指定します。TABLESPACE を省略した場合、その表を含むスキーマの所有者のデフォルトの表領域内に作成されます。

1 つ以上の LOB 列を持つヒープ構成表の場合、LOB 記憶域に対する TABLESPACE を省略すると、表を作成する表領域に LOB データおよび索引セグメントが作成されます。

ただし、1 つ以上の LOB 列を持つ索引構成表の場合、TABLESPACE を省略すると、索引構成表の主キー索引セグメントが作成された表領域に、LOB データおよび索引セグメントが作成されます。

非パーティション表の場合、TABLESPACE に指定する値は、表に関連付けられたセグメントの実際の物理属性となります。パーティション表の場合、TABLESPACE に指定する値は、PARTITION 記述で TABLESPACE を指定しないかぎり、この CREATE 文（および後続の ALTER TABLE ... ADD PARTITION 文）で指定されたすべてのパーティションに関連付けられたセグメントのデフォルト物理属性となります。

制限事項： *table* が LOB 列を含まない場合は、自動セグメント領域管理を指定した表領域を指定できません。

参照： 表領域の詳細は、14-62 ページの「[CREATE TABLESPACE](#)」を参照してください。

LOGGING | NOLOGGING 表（および制約のために必要な索引）、パーティションまたは LOB の記憶特性の作成を REDO ログ・ファイルに記録する（LOGGING）かしないか（NOLOGGING）を指定します。表のロギング属性は、その索引の属性に依存しません。

表、パーティションまたは LOB の記憶域に対して、後で実行されるダイレクト・ローダー（SQL*Loader）操作およびダイレクト・パス INSERT 操作のログをとる（LOGGING）かとらない（NOLOGGING）かも指定します。

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

LOB に対してこの句を省略した場合は、次のようになります。

- **CACHE** を指定した場合は、**LOGGING** が使用されます (**CACHE NOLOGGING** は指定できないため)。
- **NOCACHE** または **CACHE READS** を指定した場合は、表が存在する表領域の属性がデフォルトのロギング属性として使用されます。

NOLOGGING は、行データとともにインラインに格納された **LOB** に適用されません。つまり、**LOB** に対する **NOLOGGING** を 400 バイト未満の値に指定し、**STORAGE IN ROW** を使用禁止にしていなかった場合、**NOLOGGING** の指定は無視され、**LOB** データは他の表データと同様に扱われます。

非パーティション表の場合、**LOGGING** に指定する値は、表に関連付けられたセグメントの実際の物理属性となります。**パーティション表**の場合、ロギング属性に指定する値は、**PARTITION** 記述でロギング属性を指定しないかぎり、**CREATE** 文 (および後続の **ALTER TABLE ... ADD PARTITION** 文) で指定するすべてのパーティションに関連付けられたセグメントのデフォルト物理属性となります。

NOLOGGING モードでは、データの変更時に、(新しいエクステンツを **INVALID** としてマーク設定し、ディクショナリの変更を記録するために) 最小限のログが記録されます。メディア・リカバリ中に **NOLOGGING** が適用された場合、**REDO** データは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、損失してはならない表の場合は、**NOLOGGING** 操作の後にバックアップを取る必要があります。

NOLOGGING モードでの操作で生成される **REDO** ログのサイズは、**LOGGING** モードで生成されるログより非常に小さくなります。

データベースを **ARCHIVELOG** モードで実行する場合、**LOGGING** 操作の前に取ったバックアップからのメディア・リカバリによって、表がリストアされます。ただし、**NOLOGGING** 操作の前に取ったバックアップからのメディア・リカバリでは、表はリストアされません。

参照： ロギングおよびパラレル DML の詳細は、『Oracle9i データベース概要』および『Oracle9i データベース管理者ガイド』を参照してください。

RECOVERABLE | UNRECOVERABLE これらのキーワードは以前のリリースのもので、それぞれ **LOGGING** および **NOLOGGING** に置き換えられています。**RECOVERABLE** および **UNRECOVERABLE** は、下位互換用のためにサポートされていますが、**LOGGING** および **NOLOGGING** キーワードを使用することをお勧めします。

制限事項：

- パーティション表または **LOB** 記憶特性に **RECOVERABLE** を指定できません。
- パーティション表または索引構成表に **UNRECOVERABLE** を指定できません。
- **AS subquery** でのみ **UNRECOVERABLE** を指定できます。

ORGANIZATION

ORGANIZATION 句によって、表のデータ行が格納される順序を指定できます。

HEAP HEAP は、構成する *table* のデータ行の格納順序を特定しないことを示します。これはデフォルトです。

INDEX INDEX は、*table* を索引構成表として作成することを示します。索引構成表では、表の主キーが定義された索引内にデータ行が格納されます。

EXTERNAL EXTERNAL は、表がデータベースの外部にある読取り専用表であることを示します。

index_org_table_clause

index_org_table_clause を使用すると、索引構成表を作成できます。主キーに基づく索引にある表の行（主キー列の値と非キー列の値の両方）がメンテナンスされます。このため、索引構成表は主キーベースのアクセスおよび操作に最適です。索引構成表は、次のいずれかの表です。

- CREATE INDEX 文を使用して主キーベースで索引付けされるクラスタ化されていない表。
- 索引クラスタに格納されるクラスタ化表。索引クラスタは、表に対する主キーをクラスタ・キーにマップする CREATE CLUSTER 文を使用して作成されます。

索引構成表がパーティション化され、LOB 列を含む場合、最初に *index_org_table_clause*、次に *LOB_storage_clause*、その後に適切な *table_partitioning_clauses* を指定する必要があります。

制限事項：

- ROWID 型の列は、索引構成表に指定できません。
- *composite_partitioning_clause* は、索引構成表に指定できません。

注意： 主キーは行を一意に識別するため、索引構成表には主キーを指定してください。主キーには DEFERRABLE を指定できません。索引構成表の行に直接アクセスする場合は、ROWID のかわりに主キーを使用してください。

PCTTHRESHOLD integer インデックスブロック内で、インデックス構成表の行を格納するために確保されている領域の割合（パーセント）を指定します。PCTTHRESHOLD は、主キーを保持するために十分な大きさである必要があります。指定したしきい値を超える列から始まる行の後続列はすべて、オーバーフロー・セグメントに格納されます。PCTTHRESHOLD は 1 ～ 50 の値を取る必要があります。PCTTHRESHOLD を指定しない場合のデフォルト値は 50 です。

制限事項：インデックス構成表の個々のパーティションに対して PCTTHRESHOLD は指定できません。

mapping_table_clause MAPPING TABLE によって、ローカルのマッピングが物理 ROWID に作成され、ヒープ構成表に格納されます。このマッピングは、インデックス構成表のビットマップ索引の作成に必要です。

マッピング表は、親であるインデックス構成表と同じ表領域に作成されます。問合せ、DML 操作またはマッピング表の記憶特性の変更は実行できません。

制限事項：この句をパーティション化されたインデックス構成表に指定することはできません。

compression_clauses compression_clause によって、インデックス構成表のキー圧縮を使用可能または使用禁止にできます。

- COMPRESS を指定して、**キー圧縮**を使用可能にします。これによって、インデックス構成表の主キー列の値が重複しなくなります。*integer* を使用して、接頭辞の長さ（圧縮する接頭辞列数）を指定します。

接頭辞の長さの有効範囲は、1 ～（主キー列数 -1）までです。デフォルトでは（主キー列数 -1）になります。

制限事項：パーティション・レベルでは、COMPRESS を指定できますが、*integer* で接頭辞の長さを指定できません。

- NOCOMPRESS を指定して、インデックス構成表でのキー圧縮を使用禁止にします。これはデフォルトです。

index_org_overflow_clause index_org_overflow_clause によって、指定されたしきい値を超えるインデックス構成表のデータ行を、この句で指定したデータ・セグメントに格納できます。

- インデックス構成表を作成した場合、各列の最大サイズを評価し、最大限の列を概算します。オーバーフロー・セグメントが必要で、OVERFLOW を指定していない場合はエラーが発生し、CREATE TABLE 文は実行されません。このチェック機能によって、インデックス構成表に対する後続の DML 操作が、オーバーフロー・セグメントがないために失敗することを防げます。
- OVERFLOW キーワードの後の句に指定するすべての物理属性および記憶特性は、表のオーバーフロー・セグメントにのみ適用されます。インデックス構成表自体の物理属性と記憶特性、すべてのパーティションに対するデフォルト値、および各パーティションに対する値は、このキーワードの前に指定する必要があります。

- 索引構成表に 1 つ以上の LOB 列が含まれる場合は、LOB がインラインに格納できるほど小さい場合でも、OVERFLOW を指定しないと、アウトラインに格納されます。
- 表がパーティション化されている場合、オーバーフロー・データ・セグメントが主キー索引セグメントと同一レベルでパーティション化されます。

INCLUDING *column_name* 索引構成表の行を索引部分とオーバーフロー部分に分割する列を指定します。主キー列は常に索引に格納されます。*column_name* は、最後の主キー列でもその他の主キー以外の列でもかまいません。*column_name* に続くすべての主キー以外の列は、オーバーフロー・データ・セグメントに格納されます。

制限事項：索引構成表の個々のパーティションにこの句は指定できません。

注意： *column_name* で行を分割しようとした場合に、行の索引部分のサイズが、PCTTHRESHOLD の値（指定またはデフォルト）を超えると、Oracle は PCTTHRESHOLD の値に基づいて、行を切り離します。

supplemental_logging_props

supplemental_logging_props によって、追加のデータがログ・ストリームに入れられ、ログに基づくツール製品がサポートされます。

external_table_clause

external_table_clause によって、外部表を作成できます。外部表は読取り専用表で、そのメタデータはデータベースに格納されますが、データはデータベースの外部に格納されます。外部表では、データを最初はデータベースにロードせずに、データベースの外部でデータを問い合わせることができます。

参照： 外部表の使用の詳細は、『Oracle9i データ・ウェアハウス・ガイド』、『Oracle9i データベース管理者ガイド』および『Oracle9i データベース・ユーティリティ』を参照してください。

外部表の場合、データベースにデータが存在しないため、表の作成時に通常は使用可能な句の小規模のサブセットを定義します。

- *relational_properties* 句内では、*column*、*datatype* および *column_constraint* のみを指定できます。
- *physical_properties_clause* 内では、表の構成（ORGANIZATION EXTERNAL *external_table_clause*）のみを指定できます。
- *table_properties* 句内では、*parallel_clause* 句のみを指定できます。*parallel_clause* を指定すると、外部データに対する後続の問合せをパラレル化できます。

外部表の制限事項

- `external_table_clause` を指定する場合、同じ CREATE TABLE 文で他の句を指定することはできません。
- 外部表を、一時表にできません。
- 外部表に、オブジェクト型列、LOB 列または LONG 列を含めることはできません。
- 外部表に、制約を指定することはできません。そのため、外部表の作成時に `enable_disable` 句は指定できません。
- 今回のリリースでは、外部表に対して、サード・パーティのアクセス・ドライバはサポートされていません。

TYPE `TYPE access_driver_type` を指定すると、外部表の **アクセス・ドライバ** を指定できます。アクセス・ドライバは、データベースに対する外部表を解析する API です。TYPE を指定しない場合、デフォルトのアクセス・ドライバ ORACLE_LOADER が使用されます。

参照： 外部表に対する ORACLE_LOADER アクセス・ドライバについては、『Oracle9i データベース・ユーティリティ』を参照してください。

DEFAULT DIRECTORY DEFAULT DIRECTORY を指定すると、外部データ・ソースが存在するファイル・システムのディレクトリに対応するデフォルト・ディレクトリ・オブジェクトを 1 つのみ指定できます。複数のデフォルト・ディレクトリは指定できません。デフォルト・ディレクトリは、アクセス・ドライバから使用でき、エラー・ログなどの補助ファイルを格納できます。なお、外部表には、複数の外部データ・ソースを指定できます。

ACCESS PARAMETERS オプションの ACCESS PARAMETERS 句を指定すると、その外部表用のアクセス・ドライバ仕様のパラメータに値を割り当てることができます。

- `opaque_format_spec` を指定すると、パラメータおよびその値をリストできます。`opaque_format_spec` の値の指定方法については、『Oracle9i データベース・ユーティリティ』を参照してください。
- USING CLOB *subquery* を指定すると、副問合せを使用して、パラメータおよびその値を導出できます。副問合せには、集合演算子または ORDER BY 句を含めません。1 つの CLOB データ型を含む単一行を戻します。

`opaque_format_spec` でパラメータを指定する場合、または副問合せを使用してそれらを導出する場合は、この句は解析されません。外部データのコンテキスト情報は、アクセス・ドライバが解析します。

LOCATION LOCATION 句を指定すると、各外部データ・ソースに対する 1 つの外部ロケータを指定することができます。通常、`location_specifier` はファイルで、必要ではありません。Oracle はこの句を解析しません。外部データのコンテキスト情報は、アクセス・ドライバが解析します。

REJECT LIMIT REJECT LIMIT 句を指定すると、Oracle エラーが戻され、問合せが異常終了する前の、外部データの間合せで許容される変換エラーの数を指定できます。デフォルト値は 0 です。

CLUSTER 句

CLUSTER 句は、表がクラスタの一部であることを指定します。この句で指定する各列は、クラスタの各列に対応する表の列となります。一般に、表のクラスタ列は、主キーまたは主キーの一部を構成する 1 つ以上の列です。

参照： 12-2 ページの「[CREATE CLUSTER](#)」を参照してください。

クラスタ・キー内の列ごとに表から 1 つの列を指定します。列は、名前ではなく位置で一致させます。

クラスタ化表はクラスタの領域割当てを使用します。このため、PCTFREE、PCTUSED、INITRANS または MAXTRANS パラメータ、TABLESPACE 句または *storage_clause* を CLUSTER 句とともに使用しないでください。

制限事項：

- オブジェクト表および LOB 列を含む表はクラスタの一部にはできません。
- クラスタが同じ ROWDEPENDENCIES または NOROWDEPENDENCIES 設定で作成されていないかぎり、CLUSTER を ROWDEPENDENCIES または NOROWDEPENDENCIES とともに指定することはできません。

table_properties

column_properties

column_properties 句を使用すると、列の記憶域属性を指定できます。

object_type_col_properties

object_type_col_properties を使用すると、オブジェクト列、属性、あるいは列または属性の集合要素の記憶特性を指定できます。

column *column* には、オブジェクト列または属性を指定します。

substitutable_column_clause *substitutable_column_clause* を使用すると、同じ階層のオブジェクト列または属性が互いに置換可能か否かを指定できます。列が特定の型であるか、サブタイプのインスタンスを含むものであるか、またはその両方を指定できます。

- ELEMENT を指定すると、コレクションの要素型が宣言した型のサブタイプに制約されます。

- IS OF [TYPE] (ONLY type) 句を指定すると、オブジェクト列の型が宣言した型のサブタイプに制約されます。
- NOT SUBSTITUTABLE AT ALL LEVELS を指定すると、オブジェクト列がサブタイプに対応するインスタンスを持つことはできません。また、置換は、埋込みオブジェクト属性、埋込みのネストした表および VARRAY の要素には使用禁止です。デフォルトは、SUBSTITUTABLE AT ALL LEVELS です。

***substitutable_column_clause* の制限事項**

- この句は、最上位のオブジェクト列に指定できますが、オブジェクト列の属性には指定できません。
- コレクション型の列の場合、この句で指定できる部分は [NOT] SUBSTITUTABLE AT ALL LEVELS のみです。

LOB_storage_clause

LOB_storage_clause によって、LOB データ・セグメントの記憶域属性を指定できます。

- 非パーティション表の場合（パーティション句なしで *physical_properties* 句に指定した場合）、この句は、LOB データ・セグメントの表の記憶域属性を指定します。
- 表レベルで指定されたパーティション表の場合（パーティション句とともに *physical_properties* 句で指定した場合）、この句は、各パーティションまたはサブパーティションに対応付けられた LOB データに対するデフォルト記憶域属性を指定します。この記憶域属性は、パーティションまたはサブパーティション・レベルで *LOB_storage_clause* によってオーバーライドされないかぎり、すべてのパーティションまたはサブパーティションに適用されます。
- パーティション表の各パーティションの場合（*table_partition_description* の一部として指定した場合）、この句は、そのパーティションのデータ・セグメントの記憶域属性、またはこのパーティションのサブパーティションのデフォルト記憶域属性を指定します。パーティション・レベルの *LOB_storage_clause* は、表レベルの *LOB_storage_clause* をオーバーライドします。
- パーティション表のそれぞれのサブパーティションの場合（*subpartition_clause* の一部として指定した場合）、この句は、サブパーティションのデータ・セグメントの記憶域属性を指定します。サブパーティション・レベルの *LOB_storage_clause* は、パーティション・レベルおよび表レベルの *LOB_storage_clauses* をオーバーライドします。

制限事項：表がパーティション化されている場合、*LOB_index_clause* を指定できません。

LOB_item 表の表領域および記憶特性とは異なる表領域および記憶特性を明示的に定義する場合に、その LOB 列名または LOB オブジェクト属性を指定します。各 *LOB_item* にシステム管理された索引が自動的に作成されます。

LOB_segname LOB データ・セグメントの名前を指定します。LOB_item が複数指定されている場合は、LOB_segname を使用できません。

LOB_parameters LOB_parameters 句によって、様々な LOB 記憶域の要素を指定できます。

- **ENABLE STORAGE IN ROW**: 行の記憶域を使用可能にした場合、LOB 値の長さが、約 4000 バイトからシステム制御情報分を引いた長さより小さければ、LOB 値をインラインに格納します。これはデフォルトです。

制限事項: index_org_table_clause に OVERFLOW セグメントを指定しないかぎり、索引構成表に対して、このパラメータを指定できません。

- **DISABLE STORAGE IN ROW**: 行の記憶域を使用禁止にした場合、LOB 値の長さに関係なく、LOB 値はアウトライン（行の外側）に格納されます。

注意: LOB 値が格納されている場所にかかわらず、LOB ロケータは常にインライン（行の内側）に格納されます。STORAGE IN ROW の値は、一度設定すると、表を移動しないかぎり、変更できません。10-68 ページの「ALTER TABLE」の「[move_table_clause](#)」を参照してください。

- **CHUNK integer**: LOB の操作用に割り当てるバイト数を指定します。integer にデータベースのブロック・サイズの倍数を指定しなかった場合、自動的に次に大きい倍数（バイト単位）に切り上げられます。たとえば、データベースのブロック・サイズが 2048 バイトのときに integer に 2050 を指定すると、4096 バイト（2 ブロック）が割り当てられます。最大値は 32768（32KB）で、これが Oracle のブロック・サイズとして使用できる最も大きな値です。デフォルトの CHUNK サイズは、Oracle での 1 データベース・ブロックです。

CHUNK の値は、一度設定すると変更できません。

注意: CHUNK 値は、NEXT の値（デフォルト値または storage_clause で指定された値）以下である必要があります。CHUNK の値が NEXT の値を超えると、エラーが戻ります。

- **PCTVERSION integer**: LOB の記憶領域全体のうち、新規バージョンの LOB の作成に使用される最大記憶領域の割合（パーセント）を指定します。デフォルト値は 10 です。これは、LOB の記憶領域全体の 10% が使用されるまで以前のバージョンの LOB データが上書きされないことを意味します。

LOB_index_clause この句の使用は推奨しません。各 LOB の列に対する索引が自動的に生成され、LOB 索引が内部的に指定され、管理されます。

この句を指定することはできますが、指定しないようにしてください。どんな場合でも、LOB 索引を LOB データと異なる表領域に置かないでください。

参照：

- サイズが GB になる LOB を作成する場合のガイドラインを含む LOB の詳細は、『Oracle9i アプリケーション開発者ガイド - ラージ・オブジェクト』を参照してください。
- 14-54 ページの「LOB 列の例」を参照してください。
- 以前のリリースから移行された表での LOB 索引の管理方法の詳細は、『Oracle9i データベース移行ガイド』を参照してください。

varray_col_properties

VARRAY 型のデータが格納される LOB に対して、別々の記憶特性を指定する場合は、*varray_col_properties* を使用します。また、この句を指定した場合、インラインに格納できるほど小さい値でも、VARRAY は必ず LOB に格納されます。

- 非パーティション表の場合（パーティション句なしで *physical_properties* 句に指定した場合）、この句は、VARRAY の LOB データ・セグメントの表の記憶域属性を指定します。
- 表レベルで指定されたパーティション表の場合（パーティション句とともに *physical_properties* 句で指定した場合）、この句は、各パーティション（またはサブパーティション）に対応付けられた VARRAY の LOB データ・セグメントに対するデフォルト記憶域属性を指定します。
- パーティション表の各パーティションの場合（*table_partition_description* の一部として指定した場合）、この句は、そのパーティションの VARRAY の LOB データ・セグメントの記憶域属性、またはこのパーティションのサブパーティションにある VARRAY の LOB データ・セグメントのデフォルト記憶域属性を指定します。パーティション・レベルの *varray_col_properties* は、表レベルの *varray_col_properties* をオーバーライドします。
- パーティション表の各サブパーティションの場合（*subpartition_clause* の一部として指定した場合）、この句は、このサブパーティションの VARRAY のデータ・セグメントの記憶域属性を指定します。サブパーティション・レベルの *varray_col_properties* は、パーティション・レベルおよび表レベルの *varray_col_properties* をオーバーライドします。

制限事項：*LOB_parameters* の TABLESPACE パラメータは、この句の一部として指定できません。VARRAY に対する LOB 表領域は、デフォルトではそれを格納する表の表領域になります。

nested_table_col_properties

ネストした表に対して別々の記憶特性を指定し、そのネストした表を索引構成表として定義できるようにする場合は、*nested_table_col_properties* を使用します。この記憶表は、(デフォルトの記憶特性によって) 親表と同じ表領域内に作成されます。この記憶表には、この表の作成の基になった列のネストした表の値が格納されます。

ネストした表の型を持つ列または列属性付きで表を作成する場合は、この句を挿入する必要があります。*nested_table_col_properties* 句内で、親であるオブジェクト表に対する場合と同じ働きをする句は、ここでは繰り返されません。

nested_item 型がネストした表である列 (または、その表のオブジェクト型の最上位の属性) の名前を指定します。

storage_table *nested_item* の行を含む表の名前を指定します。非パーティション表の場合、記憶表は親表と同じスキーマおよび同じ表領域内に作成されます。パーティション表の場合、記憶表はスキーマのデフォルトの表領域内に作成されます。

***storage_table* の制限事項**

- ネストした表の記憶表はパーティション化できません。
- *storage_table* で DML 文を直接問い合わせたり、実行することはできませんが、その記憶特性は、ALTER TABLE 文で名前を指定することによって変更できます。

参照： ネストした表の列に対する記憶特性の変更方法については、10-2 ページの「ALTER TABLE」を参照してください。

RETURN AS 問合せの結果として何を戻り値とするかを指定します。

- VALUE は、ネストした表自体のコピーを戻します。
- LOCATOR は、コレクション・ロケータをネストした表のコピーに戻します。

注意： ロケータのセッションには有効範囲が決められており、分散したセッションに使用できません。LOB ロケータとは異なり、コレクション・ロケータはコレクション・インスタンスの変更に使用できません。

segment_attributes_clause または *LOB_storage_clause* を指定しない場合、ネストした表はヒープ構成され、デフォルトの記憶特性で作成されます。

***nested_table_col_properties* の制限事項**

- この句は、一時表には指定できません。
- *OID_clause* は指定できません。
- (*segment_attributes_clause* の一部として) *TABLESPACE* をネストした表に指定することはできません。表領域は常に親表の表領域です。
- 作成時、*table_ref_constraint*、*column_ref_constraint* またはネストした表の属性に対する参照制約を (*object_properties* の一部として) 指定することはできません。ただし、ネストした表を修正して *ALTER TABLE* を使用する制約を追加できます。
- 記憶表に対して問合せまたは *DML* 文を直接実行できませんが、*ALTER TABLE* 文に記憶表の名前を指定することで、ネストした表の列の記憶特性を変更できます。

参照： ネストした表の列に対する記憶特性の変更方法については、10-2 ページの「[ALTER TABLE](#)」を参照してください。

xmltype_storage_clause

xmltype_storage_clauses を指定すると、*XMLType* 列に対して記憶域属性を指定できます。*XMLType* は、常にキャラクタ *LOB* として格納されます。*LOB_segname* および *LOB_parameters* の詳細は、14-32 ページの「[LOB_storage_clause](#)」を参照してください。

table_partitioning_clauses

table_partitioning_clauses を使用すると、パーティション表を作成できます。

制限事項： *TIMESTAMP WITH TIME ZONE* 列をパーティション化キーの一部として指定できません。

注意： ブロック・サイズが異なる表領域のパーティション化されたデータベース・エンティティの記憶域には、制限事項があります。これらの制限事項の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。

range_partitioning

range_partitioning を使用して、*column_list* の範囲の値で表をパーティション化します。索引構成表に対して、*column_list* は表の主キー列のサブセットである必要があります。

column_list 行がどのパーティションに属するかを判断するために使用される、列の順序リストを指定します（パーティション化キー）。

制限事項：column_list の列は、ROWID、LONG または LOB 以外の組込みデータ型を指定できます。

PARTITION partition 物理パーティション属性を指定します。partition を省略した場合、名前はパーティションに対して SYS_Pn の形式で生成されます。partition は、スキーマ・オブジェクトのネーミング規則に従っている必要があります。また、各部分は、2-106 ページの「スキーマ・オブジェクトのネーミング規則」の説明に従って指定する必要があります。

注意：

- 64KB - 1 以内のパーティションと 64KB - 1 以内のサブパーティションを指定できます。この数字未満の制限を適用する要因については、『Oracle9i データベース管理者ガイド』を参照してください。
 - パーティションが 1 つのみのパーティション表も作成できます。ただし、パーティションが 1 つのみのパーティション表と、非パーティション表は違うことに注意してください。たとえば、非パーティション表にはパーティションを追加できません。
-
-

VALUES LESS THAN 句 現行のパーティションの上限（境界は含まない）を指定します。value_list は、partition_by_range_clause の column_list に対応するリテラル値の順序リストです。キーワード MAXVALUE を value_list のリテラルで置換できます。MAXVALUE には、常に他の値（NULL を含む）より高位にソートされる最大値を指定します。

パーティション境界の上限に MAXVALUE 以外の値を指定した場合、表に暗黙の整合性制約が課せられます。

注意： 表が DATE 列でパーティション化されている場合、および NLS 日付書式で年の最初の 2 桁の数字が指定されていない場合、年の「YYYY」4 文字書式マスクで TO_DATE ファンクションを使用する必要があります。（「RRRR」書式マスクは、サポートしていません。）NLS 日付書式は、NLS_TERRITORY によって暗黙的に決定され、NLS_DATE_FORMAT によって明示的に決定されます。

参照：

- パーティション境界の詳細は、『Oracle9i データベース概要』を参照してください。
- これらの初期化パラメータの詳細は、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。
- 14-57 ページの「レンジ・パーティション化の例」を参照してください。

list_partitioning

list_partitioning 句を使用すると、*column* のリテラル値のリストで表をパーティション化します。リスト・パーティション化は、個々の行が固有のパーティションにマップする方法に関する制御に便利です。

各 *partition_value* リストは、1 つ以上値を持つ必要があります。MAXVALUE キーワードは、リスト・パーティションには意味がないため無効となります。リスト・パーティションは、順序付けされていません。

各パーティションの値のリストを導出する文字列は、最大 4KB です。すべてのパーティションの *partition_values* の総数を、64KB-1 以内に指定します。

注意： VALUES 句のパーティションの値に、リテラルの NULL を指定できません。ただし、その次の問合せでパーティションのデータにアクセスするには、WHERE 句で比較条件ではなく、NOT NULL 条件を使用する必要があります。

リスト・パーティション化の制限事項

- *column_list* でパーティション化キーを 1 つ指定する必要があります。LOB 列は、指定できません。
- パーティション化キーがオブジェクト型列の場合は、列型の 1 つの属性でもパーティション化できません。
- VALUES 句で MAXVALUE を指定できません。
- VALUES 句の *partition_value* は、表のすべてのパーティションの間で一意である必要があります。
- パーティションおよび索引構成表をリストできません。

hash_partitioning

hash_partitioning を使用して、表がハッシュ方式でパーティション化されるように指定します。列の値にパーティション化キーとして指定されたハッシュ・ファンクションを使用して、列をパーティションに割り当てます。次のいずれかの方法で、ハッシュ・パーティション化できます。

individual_hash_partitions 名前によってそれぞれのパーティションを指定します。TABLESPACE 句は、パーティションが格納される場所を指定します。

注意： 別のキャラクタ・セットを使用してデータベースを使用しているか、使用する予定がある場合は、キャラクタ列をパーティション化する際に注意してください。文字のソート順序は、すべてのキャラクタ・セットで同一ではありません。

hash_partitions_by_quantity パーティション数を指定します。この場合、SYS_Pnnn 形式のパーティション名を割り当てます。STORE IN 句は、ハッシュ・パーティションが格納されている 1 つ以上の表領域を指定します。表領域数はパーティション数と同じである必要はありません。パーティション数が表領域数より多い場合、表領域名を循環させます。

ハッシュ・パーティション化の両方の方法で、ハッシュ・パーティション（またはサブパーティション）に指定できるのは、TABLESPACE のみです。ハッシュ・パーティションは、その他のすべての属性を表レベルのデフォルトから継承します。ハッシュ・パーティションは、パーティション・レベルで指定された属性と表レベルのデフォルトからのその他すべての属性を継承します。

表レベルで指定された表領域の記憶域は、パーティション・レベルで指定された表領域の記憶域でオーバーライドされ、パーティション・レベルで指定された表領域の記憶域は、サブパーティション・レベルで指定された表領域の記憶域でオーバーライドされます。

hash_partitioning 句の STORE IN 句と *hash_partitioning_storage* 句の TABLESPACE 句の両方で表領域の記憶域を指定する場合、パーティションの位置は STORE IN 句で指定する表の作成位置となります。TABLESPACE 句には、後続の操作の表レベルのデフォルト表領域を指定します。

column_list 行がどのパーティションに属するかを判断するために使用される、列の順序リストを指定します（パーティション化キー）。

参照： ハッシュ・パーティション化については、『Oracle9i データベース概要』を参照してください。

ハッシュ・パーティション化の制限事項

- `column_list` に指定できるのは 16 列以下です。
- `column_list` には、ROWID または UROWID 疑似列を含めることはできません。
- `column_list` の列には、ROWID、LONG または LOB 以外の組込みデータ型を指定できません。

参照： キャラクタ・セットのサポートについては、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

composite_partitioning

composite_partitioning を使用すると、表がまずレンジ・パーティション化され、次にそれらのパーティションがハッシュ・サブパーティション化されます。レンジ・パーティション化とハッシュ・サブパーティション化の組合せを、**コンポジット・パーティション化**といいます。

制限事項： 索引構成表にコンポジット・パーティション化を指定できません。

subpartition_clause *subpartition_clause* を使用して、表の各パーティションをハッシュ・サブパーティション化します。`column_list` のサブパーティション化はパーティション化キーには関連しませんが、同じ制限事項が適用されます。

SUBPARTITIONS *quantity* 表の各パーティションにおけるデフォルトのサブパーティション数と、格納されている 1 つ以上の表領域を任意で指定します。

デフォルト値は 1 です。ここで *subpartition_clause* を指定しないと、後で *partition_level_subpartitioning* 句を指定しないかぎり、各パーティションは 1 つのハッシュ・サブパーティションで作成されます。

table_partition_description

LOB_storage_clause *LOB_storage_clause* によって、このパーティションの 1 つ以上の LOB 項目に対して LOB 記憶特性を指定できます。LOB 項目に *LOB_storage_clause* を指定しない場合、各 LOB データ・パーティションに対する名前が生成されます。LOB データおよび LOB 索引パーティションに対するシステム生成名は、それぞれ SYS_LOB_Pn および SYS_IL_Pn の形式をとります。ここで、P はパーティション、n はシステム生成番号です。

varray_col_properties *varray_col_properties* によって、このパーティションの 1 つ以上の VARRAY 項目に対して記憶特性が指定できます。

partition_level_subpartition *partition_level_subpartition* 句によって、パーティションのハッシュ・サブパーティションを指定します。この句は、*subpartition_clause* でのデフォルト設定をオーバーライドします。

制限事項： コンポジット・パーティション表に対してのみこの句を指定できます。

- 個々のパーティションを名前指定できます。また、任意に各パーティションが格納される表領域を指定することもできます。
- サブパーティションの数を指定できます（または、サブパーティションが格納される 1 つ以上の表領域を指定することもできます）。この場合、SYS_SUBPnnn という形式のサブパーティション名を割り当てます。表領域の数は、サブパーティションの数と等しくなくてもかまいません。パーティション数が表領域数より多い場合、表領域名を循環させます。

row_movement_clause

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

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

注意：

- 通常の表（ヒープ構成表）の場合は、UPDATE 操作中に行を移動すると、その行の ROWID が変更されます。
 - 索引構成表での行の移動の場合は、論理 ROWID の物理推測コンポーネントは不正確になりますが、論理 ROWID は有効のままです。
-

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

制限事項： パーティション表に対してのみこの句を指定できます。

CACHE | NOCACHE | CACHE READS

CACHE 句 この句は、アクセス頻度の高いデータについて、フル・テーブル・スキャンの実行時に、この表のために取り出されたブロックを、バッファ・キャッシュ内の最低使用頻度（LRU）リストの最高使用頻度側に置く場合に指定します。この属性は、小規模な参照表で有効です。

LOB_storage_clause のパラメータとして **CACHE** を使用すると、より高速なアクセスのために、LOB データ値がバッファ・キャッシュに配置されます。

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

NOCACHE 句 この句は、アクセス頻度の低いデータについて、フル・テーブル・スキャンの実行時に、この表のために取り出されたブロックを、バッファ・キャッシュ内の LRU リストの最低使用頻度側に置く場合に指定します。

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

制限事項： 索引構成表に **NOCACHE** は指定できません。

CACHE READS **CACHE READS** は LOB 記憶域にのみ適用されます。LOB 値が書き込み操作中ではなく読み込み操作中にバッファ・キャッシュに入れられることを指定します。

NOROWDEPENDENCIES | ROWDEPENDENCIES

表が**行レベル依存の追跡**を使用するかどうかを指定できます。この機能を使用すると、表の各行は、行を変更した最後のトランザクションのコミット時刻以降を示すシステム変更番号（SCN）を持つことになります。この設定は、表の作成後に変更できません。

ROWDEPENDENCIES 行レベル依存の追跡を使用可能にする場合は、**ROWDEPENDENCIES** を指定します。この設定は、主にレプリケーション環境でパラレル伝播を許可する場合に便利です。各行につきサイズが 6 バイト増えます。

NOROWDEPENDENCIES 表を行レベル依存の追跡機能に使用しない場合は、**NOROWDEPENDENCIES** を指定します。これはデフォルトです。

参照： レプリケーション環境での行レベル依存の追跡の使用については、『Oracle9i レプリケーション』を参照してください。

MONITORING | NOMONITORING

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

制限事項：一時表に MONITORING を指定することはできません。

NOMONITORING 表の変更統計を収集しない場合は、NOMONITORING を指定します。これはデフォルトです。

制限事項：一時表に NOMONITORING を指定することはできません。

parallel_clause

parallel_clause によって、表の平行作成および作成後に問合せおよび DML の平行化のデフォルト値の設定ができます。

注意： *parallel_clause* 構文は、以前のリリースの構文に代わるものです。以前のリリースの構文は下位互換用にサポートされていますが、動作がわずかに異なることがあります。

NOPARALLEL シリアル実行を行う場合は、NOPARALLEL を指定します。これはデフォルトです。

PARALLEL 並列度を選択する場合は、PARALLEL を指定します。並列度とは、すべての関係するインスタンスで使用可能な CPU の数に、初期化パラメータ PARALLEL_THREADS_PER_CPU の値を掛けたものです。

PARALLEL integer *integer* には、平行操作で使用する平行・スレッド数である、**並列度**を指定します。各平行・スレッドは、1、2 個の平行実行サーバーを使用します。通常、最適な並列度が計算されるため、*integer* に値を指定する必要はありません。

parallel_clause に関する注意事項

- 表に LOB 型またはユーザー定義オブジェクト型の列が含まれる場合、表に対する後続の INSERT 操作、UPDATE 操作、DELETE 操作およびこの文は通知なしに逐次実行されます。ただし、後続の問合せは平行で実行されます。
- *parallel_clause* の結果は平行・ヒントによってオーバーライドされます。

- DML 文および CREATE TABLE ... AS SELECT 文はパラレルで実行されるリモート・オブジェクトを参照します。ただし、リモート・オブジェクトはリモート・データベースにある必要があります。参照は、ローカル・データベースにあるオブジェクトにループバックできません（たとえば、ローカル・データベースのオブジェクトを指定するリモート・データベースのシノニムから参照することはできません）。

参照： このパラレル化操作の詳細は、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』、『Oracle9i データベース概要』および『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

enable_disable_clause

enable_disable_clause によって、制約が適用されるかどうかを指定できます。デフォルトでは、制約は ENABLE VALIDATE 状態で作成されます。

制限事項：

- 整合性制約を使用可能または使用禁止にする場合、この文または前の文に制約を定義する必要があります。
- 参照された一意制約または主キー制約がすでに使用可能でない場合は、参照整合性制約を使用可能にできません。

参照： 制約の詳細は、11-72 ページの「[constraint_clause](#)」を参照してください。

ENABLE 句

制約を表のすべての新しいデータに適用させる場合は、ENABLE を指定します。

- ENABLE VALIDATE は、すべての旧データも制約に従うことを指定します。制約が使用可能で、妥当性チェック済の場合は、すべてのデータが現在有効で、今後も有効であることが保証されます。

整合性制約に違反する行が表にある場合、制約は使用禁止のままエラーを戻します。すべての行が制約に従っている場合、Oracle は制約を使用可能にします。新規データが整合性制約に違反する場合、その後の文は実行されず、整合性制約の違反を示すエラーが戻されます。

主キー制約を ENABLE VALIDATE モードに設定すると、妥当性チェック・プロセスによって主キー列に NULL が含まれないことが検証されます。これによるオーバーヘッドを回避するには、この表の主キー制約を使用可能にする前に、主キーの各列に NOT NULL マークを付けます（最適な結果を得るためには、列にデータを入力する前に行ってください）。

- ENABLE NOVALIDATE を使用すると、制約データに対して新しく行うすべての DML 操作が制約に従うことが保証されます。この句は、表の既存データが制約に従っていることを保証しないため、表ロックは必要ありません。

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

一意制約または主キー制約を使用可能にした場合で、キーに索引が存在しない場合、Oracle は一意の索引を作成します。その後で制約が使用禁止になった場合、この索引は削除されます。そのため、制約が使用可能になるたびに索引が再作成されます。

索引の再作成を避け、余分な索引を削除するために、最初に使用禁止にした主キー制約および一意制約を新しく作成します。その後、一意でない索引を作成して（または、既存の一意でない索引を使用して）制約を適用してください。制約が使用禁止の場合、一意でない索引は削除されず、後続の `ENABLE` 操作が容易になります。

`ENABLE NOVALIDATE` から `ENABLE VALIDATE` に単一制約状態を変更すると、パラレルで操作が実行できるため、読取り、書込みまたはその他の DDL 操作が中断されません。

ENABLE の制限事項 使用禁止になっている一意キー、または主キーを参照する外部キーを使用可能にすることはできません。

DISABLE 句

整合性制約を使用禁止にする場合は、`DISABLE` を指定します。データ・ディクショナリでは、使用禁止になっている整合性制約は、使用可能な制約とともに表示されます。この句を指定せずに制約を作成した場合は、その制約は自動的に使用可能になります。

- `DISABLE VALIDATE` は、制約を使用禁止にし制約の索引を削除しますが、制約は有効のままです。この機能は大変有効です。データ・ウェアハウスでは、一意キーに重複しない範囲の値を持つ一定量のデータをレンジ・パーティション表にロードする必要があります。このような場合、制約が使用禁止で、妥当性チェック済の場合は、索引を持たないことによって領域を節約できます。`ALTER TABLE` 文の `exchange_partition_clause` または `SQL*Loader` を使用して、データを非パーティション表からパーティション表にロードすることもできます。他の `SQL` 文を使用した表に対するすべての変更（挿入、更新および削除）は禁止されます。

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

- `DISABLE NOVALIDATE` は、Oracle によって制約がメンテナンスされないこと（使用禁止になっているため）、および制約が真であると保証されないこと（妥当性チェックが行われていないため）を示します。

外部キー制約が `DISABLE NOVALIDATE` 状態であっても、外部キーが参照している主キーを持つ表を削除できません。また、オブティマイザは、`DISABLE NOVALIDATE` 状態でも制約を使用できます。

参照： この設定の使用の詳細は、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。

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

一意索引を使用している一意制約または主キー制約を使用禁止にすると、一意索引は削除されます。

UNIQUE

UNIQUE 句によって、指定された列または列の組合せに定義された一意制約を使用可能または使用禁止にできます。

PRIMARY KEY PRIMARY KEY 句によって、表の主キー制約を使用可能または使用禁止にできます。

CONSTRAINT CONSTRAINT 句によって、*constraint* に指定する整合性制約を使用可能または使用禁止にできます。

KEEP | DROP INDEX この句を使用すると、一意制約または主キー制約の適用に使用される索引を保存するか、あるいは削除するかを指定できます。

制限事項：一意制約または主キー制約を使用禁止にする場合のみにこの句を指定できます。

using_index_clause

using_index_clause を使用すると、Oracle が一意制約および主キー制約の適用に使用する索引の指定、または制約の適用で使用する索引を作成するよう、Oracle に指示することができます。

- *schema.index* を指定すると、指定した索引を使用して制約が適用されます。索引がない、または制約の適用に索引が使用できない場合、エラーが戻されます。
- *create_index_statement* を指定すると、索引が作成され、制約の適用に使用されます。索引が作成できない、または制約の適用に索引が使用できない場合、エラーが戻されます。
- 既存の索引を指定せず、新しい索引を作成しない場合、索引が作成されます。この場合、次のようになります。
 - 索引には制約と同じ名前が割り当てられます。
 - 索引には、INITRANS、MAXTRANS、TABLESPACE、STORAGE および PCTFREE の各パラメータ値を選択できます。
 - 表がパーティション化されている場合、一意制約または主キー制約にローカル・パーティション索引またはグローバル・パーティション索引を指定できます。

参照： 制約の適用で Oracle が使用する索引の作成方法の例は、11-92 ページの「[明示的な索引の制御例](#)」を参照してください。

using_index_clause の制限事項

- この句は、ビュー制約には指定できません。
- 一意制約および主キー制約が使用可能な場合のみ、*using_index_clause* を指定できます。
- 索引構成表の主キーが使用可能な場合は、索引の指定 (*schema.index*) および作成 (*create_index_statement*) はできません。

参照：

- LOCAL と *global_index_clause* の詳細、および索引に関する NOSORT と LOGGING|NOLOGGING の詳細は、12-58 ページの「[CREATE INDEX](#)」を参照してください。
- INITRANS、MAXTRANS、TABLESPACE、STORAGE および PCTFREE パラメータの詳細は、14-23 ページの「[segment_attributes_clause](#)」を参照してください。
- 主キー制約および一意制約の詳細は、11-72 ページの「[constraint_clause](#)」を参照してください。

global_partitioned_index

global_partitioned_index を使用すると、索引のパーティション化がユーザー定義であり、基となる表と同一レベルでパーティション化されないことを指定できます。デフォルトでは、非パーティション索引はグローバル索引です。

PARTITION BY RANGE PARTITION BY RANGE によって、*column_list* で指定された列の値の範囲でグローバル索引がパーティション化されるように指定します。ローカル索引にこの句は指定できません。

column_list *column_list* には、索引をパーティション化する表の列名を指定します。*column_list* では、索引の列リストの左の接頭辞を指定する必要があります。

column_list には、最大 32 列まで指定できます。なお、ROWID 疑似列および ROWID 型の列は指定できません。

注意： 別のキャラクタ・セットを使用してデータベースを使用しているか、使用する予定がある場合は、キャラクタ列をパーティション化する際に注意してください。文字のソート順序は、すべてのキャラクタ・セットで同一ではありません。

参照： キャラクタ・セットのサポートについては、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

PARTITION PARTITION 句によって、個々の索引パーティションを記述できます。句の数によってパーティションの数が決まります。*partition* を指定しない場合、名前は *SYS_Pn* の形式で生成されます。

VALUES LESS THAN VALUES LESS THAN (*value_list*) には、グローバル索引の現在のパーティションの上限（境界は含まない）を指定します。*value_list* には、*partition_by_range_clause* の *column_list* と対応するリテラル値を順番にカンマで区切って指定します。最後のパーティションの *value_list* は、必ず MAXVALUE を指定してください。

注意： *index* が DATE 列でパーティション化されている場合、および日付書式で年の最初の 2 桁の数字が指定されていない場合、年の 4 文字書式マスクで TO_DATE ファンクションを使用する必要があります。日付書式は、NLS_TERRITORY によって暗黙的に決定され、NLS_DATE_FORMAT によって明示的に決定されます。

exceptions_clause

制約に違反するすべての列の ROWID を格納する表を指定します。スキーマを指定しない場合、自分のスキーマ内に例外表があるとみなされます。この句自体を指定しない場合、表の名前は EXCEPTIONS になります。例外表は、ローカル・データベース内にある必要があります。

次のいずれかのスクリプトを使用して、EXCEPTIONS 表を作成できます。

- UTLEXCPT.SQL は、物理 ROWID を使用します。そのため、行は、索引構成表からではなく従来表から収集されます（次の注意を参照）。
- UTLEXPT1.SQL は、ユニバーサル ROWID を使用します。そのため、行は、従来表と索引構成表の両方から収集されます。

独自の例外表を作成する場合、これら 2 つのスクリプトのいずれかで規定される形式に従う必要があります。

参照：

- これらのスクリプトの使用に関する互換性については、『Oracle9i データベース移行ガイド』を参照してください。
- 移行行および連鎖行の削除については、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。

注意： ユニバーサル ROWID ではなく、主キーに基づく索引構成表から例外を収集する場合、索引構成表ごとに別の例外表を作成し、主キー記憶域を確保する必要があります。スクリプトを変更および再発行することによって、別の名前の例外表を複数作成できます。

参照：

- 索引構成表に対する SQL スクリプトの詳細は、『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の DBMS_IOT パッケージを参照してください。
 - 10-72 ページの「索引構成表の EXCEPTIONS INTO の例」を参照してください。
-

CASCADE CASCADE を指定して、指定した整合性制約に依存する整合性制約を使用禁止にします。参照整合性制約を構成する主キーまたは一意キーを使用禁止にする場合、この句を指定します。

制限事項： DISABLE を指定した場合のみ、CASCADE を指定できます。

AS subquery

副問合せを指定して表の内容を定義します。表の作成時に、副問合せの結果として戻された行を表の中に挿入します。

parallel_clause この文で *parallel_clause* を指定した場合、INITIAL 記憶域パラメータに対して指定する値は無視され、かわりに NEXT パラメータの値が使用されます。

参照： これらのパラメータの詳細は、17-50 ページの「*storage_clause*」を参照してください。

ORDER BY ORDER BY 句は、文によって戻される行を順序付けます。

注意： この句を CREATE TABLE で指定した場合、この句が表全体にわたるデータを順序付けるとはかぎりません（たとえば、パーティション間での順序付けは行われません）。同じキーの索引を ORDER BY キー列として作成する場合に、この句を指定します。Oracle は、ORDER BY キーのデータをクラスタ化し、索引キーに対応させます。

オブジェクト表の場合、*subquery* には表の型に対応する 1 つの式、または表の型の最上位属性の数のどちらかを設定できます。

参照： 17-4 ページの「*SELECT*」を参照してください。

subquery が、既存のマテリアライズド・ビューと（部分的または完全に）同等のビューを戻す場合、Oracle は、*subquery* に指定した 1 つ以上の表のかわりにマテリアライズド・ビュー（クエリー・リライト用）を使用することがあります。

参照： マテリアライズド・ビューおよびクエリー・リライトの詳細は、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

データ型およびデータ長は、副問合せの結果から導出されます。Oracle では、整合性制約については次の規則も適用されます。

- 副問合せで、列を含む式ではなく列を選択した場合、選択した表の列に NOT NULL 制約が明示的に作成されていると、新しい表の対応する列にも NOT NULL 制約が自動的に定義されます。ただし、選択した表の列（たとえば、主キー用）に Oracle によって暗黙的に作成された NOT NULL 制約は、新しい表には引き継がれません。
- CREATE TABLE 文に AS *subquery* と、CONSTRAINT 句または EXCEPTIONS INTO を指定した ENABLE 句の両方を指定した場合、AS *subquery* は無視されます。制約に違反する行がある場合、表は作成されずエラーが戻されます。

subquery 内のすべての式が、式ではなく列の場合、表定義から列を完全に省略できます。この場合、表の列名は *subquery* の列の名前と同じになります。

TO_LOB ファンクションと組み合わせて *subquery* を使用すると、別の表の LONG 列の値を、作成する表の列の LOB 値に変換できます。

参照：

- LONG を LOB にコピーする理由およびタイミングについては、『Oracle9i データベース移行ガイド』を参照してください。
- TO_LOB ファンクションの使用方法については、6-5 ページの「[変換ファンクション](#)」を参照してください。
- *order_by_clause* の詳細は、17-4 ページの「[SELECT](#)」を参照してください。

制限事項：

- 表中の列数は、副問合せ中の式の数と同じである必要があります。
- 列定義では、列名、デフォルト値および整合性制約のみを指定できます。データ型は指定できません。
- AS *subquery* を含む CREATE TABLE 文には、参照整合性制約を定義できません。そのかわりに、制約を指定せずに表を作成し、後で ALTER TABLE 文を使用してその制約を追加できます。

例

一般的な例

次の文は、人事情報のデモ・スキーマ hr が所有する employees 表を定義します。

```
CREATE TABLE employees
( employee_id    NUMBER(6)
, first_name     VARCHAR2(20)
, last_name      VARCHAR2(25)
  CONSTRAINT emp_last_name_nn NOT NULL
, email          VARCHAR2(25)
  CONSTRAINT emp_email_nn     NOT NULL
, phone_number   VARCHAR2(20)
, hire_date      DATE  DEFAULT SYSDATE
  CONSTRAINT emp_hire_date_nn NOT NULL
, job_id         VARCHAR2(10)
  CONSTRAINT emp_job_nn      NOT NULL
, salary         NUMBER(8,2)
  CONSTRAINT emp_salary_nn   NOT NULL
, commission_pct NUMBER(2,2)
, manager_id     NUMBER(6)
, department_id  NUMBER(4)
, dn            VARCHAR2(300)
, CONSTRAINT emp_salary_min
  CHECK (salary > 0)
, CONSTRAINT emp_email_uk
  UNIQUE (email)
) ;
```

この表は 12 列で構成されます。employee_id 列は、NUMBER データ型です。hire_date 列は、データ型が DATE、デフォルト値が SYSDATE です。last_name 列は、VARCHAR2 型で、NOT NULL 制約などがあります。

一時表の例 次の文は、販売員が使用するサンプル・データベースの一時表 today_sales を作成します。各販売員のセッションは、自身のその日の販売データを表に格納します。一時的なデータは、セッションの終わりに削除されます。

```
CREATE GLOBAL TEMPORARY TABLE today_sales
ON COMMIT PRESERVE ROWS
AS SELECT * FROM orders WHERE order_date = SYSDATE;
```

置換可能な表および列のサンプル 次の文は、15-20 ページの「[型の階層例](#)」で作成した person_t 型から置換可能な表を作成します。

```
CREATE TABLE persons OF person_t;
```

次の文は、person_t 型の置換可能な列で表を作成します。

```
CREATE TABLE books (title VARCHAR2(100), author person_t);
```

persons または books に挿入するときに、person_t またはそのサブタイプの属性に対する値を指定できます。例では、16-69 ページの「[置換可能な表および列への挿入例](#)」で示した文を挿入します。

組込みファンクションおよび条件を使用して、このような表からデータを抽出することができます。その例として、6-178 ページの「[TREAT](#)」、6-153 ページの「[SYS_TYPEID](#)」および 5-17 ページの「[IS OF type 条件](#)」を参照してください。

記憶域の例 記憶域の容量が小さく、割当てに制限のある human_resource 表領域の中にサンプル表 salgrade を定義する場合は、次の文を発行します。

```
CREATE TABLE salgrade
  ( grade  NUMBER    CONSTRAINT pk_salgrade
                                PRIMARY KEY
                                USING INDEX TABLESPACE users_a,
    losal  NUMBER,
    hisal  NUMBER )
TABLESPACE human_resource
STORAGE (INITIAL      6144
         NEXT         6144
         MINEXTENTS    1
         MAXEXTENTS    5 );
```

この文では grade 列に主キー制約を定義し、この制約を適用するために自動的に作成される索引を、users_a 表領域に作成するように指定しています。

参照： 整合性制約の定義の例は、11-72 ページの「[constraint_clause](#)」を参照してください。

PARALLEL の例 次の文は、最適な数のパラレル実行サーバーを使用する表を作成して、employees をスキャンし、dept_80 に移入します。

```
CREATE TABLE dept_80
  PARALLEL
  AS SELECT * FROM employees
  WHERE department_id = 80;
```

パラレル化を使用することによって、表を作成するためにパラレル実行サーバーが使用されるため、表作成が高速化されます。表が作成された後、表へのアクセスに表作成と同じ並列度が使用されるため、表の間合せも高速化します。

NOPARALLEL の例 次の文は、同じ表をシリアルで作成します。後続の DML および表の間合せもシリアルで実行されます。

```
CREATE TABLE dept_80
  AS SELECT * FROM employees
  WHERE department_id = 80;
```

ENABLE VALIDATE の例 次の文は、departments 表を作成し、NOT NULL 制約を定義して、その制約を ENABLE VALIDATE 状態にします。

```
CREATE TABLE departments
  ( department_id    NUMBER(4)
    , department_name VARCHAR2(30)
      CONSTRAINT dept_name_nn NOT NULL
    , manager_id     NUMBER(6)
    , location_id    NUMBER(4)
    , dn             VARCHAR2(300)
  ) ;
```

DISABLE の例 次の文は、同じ departments 表を作成しますが、使用禁止の主キー制約を定義します。

```
CREATE TABLE departments
  ( department_id    NUMBER(4)    PRIMARY KEY DISABLE
    , department_name VARCHAR2(30)
      CONSTRAINT dept_name_nn NOT NULL
    , manager_id     NUMBER(6)
    , location_id    NUMBER(4)
    , dn             VARCHAR2(300)
  ) ;
```

ネストした表の例 次の文は、ネストした表の列 `ad_textdocs_ntab` でサンプル表 `pm.print_media` を作成します。

```
CREATE TABLE print_media
( product_id      NUMBER(6)
, ad_id           NUMBER(6)
, ad_composite    BLOB
, ad_sourcetext   CLOB
, ad_finaltext    CLOB
, ad_fltextn      NCLOB
, ad_textdocs_ntab textdoc_tab
, ad_photo        BLOB
, ad_graphic      BFILE
, ad_header       adheader_typ
, press_release   LONG
) NESTED TABLE ad_textdocs_ntab STORE AS textdocs_nestedtab;
```

LOB 列の例 次の文は、仮想の LOB 記憶特性を持つ、同じ `print_media` 表を作成します。

```
CREATE TABLE print_media
( product_id      NUMBER(6)
, ad_id           NUMBER(6)
, ad_composite    BLOB
, ad_sourcetext   CLOB
, ad_finaltext    CLOB
, ad_fltextn      NCLOB
, ad_textdocs_ntab textdoc_tab
, ad_photo        BLOB
, ad_graphic      BFILE
, ad_header       adheader_typ
, press_release   LONG
) NESTED TABLE ad_textdocs_ntab STORE AS textdocs_nestedtab
LOB (ad_sourcetext, ad_finaltext), STORE AS
(TABLESPACE lob_seg_ts
 STORAGE (INITIAL 6144 NEXT 6144)
 CHUNK 4000
 NOCACHE LOGGING);
```

この例では、CHUNK の値を 4096 (2048 のブロック・サイズの近似倍数) まで切り上げます。

索引構成表の例 次の文は、索引構成表 `hr.countries` を作成します。

```
CREATE TABLE countries
(
  country_id      CHAR(2)
    CONSTRAINT country_id_nn NOT NULL
  , country_name  VARCHAR2(40)
  , currency_name VARCHAR2(25)
  , currency_symbol VARCHAR2(3)
  , region        VARCHAR2(15)
  , CONSTRAINT country_c_id_pk
    PRIMARY KEY (country_id)
)
ORGANIZATION INDEX
INCLUDING country_name
PCTTHRESHOLD 2
STORAGE
(
  INITIAL 4K
  NEXT 2K
  PCTINCREASE 0
  MINEXTENTS 1
  MAXEXTENTS 1 )
OVERFLOW
STORAGE
(
  INITIAL 4K
  NEXT 2K
  PCTINCREASE 0
  MINEXTENTS 1
  MAXEXTENTS 1 );
```

外部表の例 次の文は、サンプル表 `hr.employees` のサブセットを示す外部表を作成します。ORACLE_LOADER アクセス・ドライバの詳細および *opaque_format_spec* への値の指定方法については、『Oracle9i データベース・ユーティリティ』を参照してください。

```
CREATE TABLE emp_external (
  employee_id  NUMBER(6),
  last_name    VARCHAR2(20),
  email        VARCHAR2(25),
  hire_date    DATE,
  job_id       VARCHAR2(10),
  salary       NUMBER(8,2)
)
ORGANIZATION EXTERNAL
(TYPE oracle_loader
DEFAULT DIRECTORY admin
ACCESS PARAMETERS
(
```

```
RECORDS DELIMITED BY newline
BADFILE 'ulcase1.bad'
DISCARDFILE 'ulcase1.dis'
LOGFILE 'ulcase1.log'
SKIP 20
FIELDS TERMINATED BY ","  OPTIONALLY ENCLOSED BY '"'
(
  deptno      INTEGER EXTERNAL,
  dname       CHAR,
  loc         CHAR
)
)
LOCATION ('ulcase1.dat')
)
REJECT LIMIT UNLIMITED;
```

パーティション化の例

ハッシュ・パーティション化の例 サンプル表 `oe.orders` (仮想の名前) は、パーティション化されていません。次の例は、このような大規模な表をハッシュでパーティション化し、注文情報を時間間隔で処理します。

```
CREATE TABLE product_information
( product_id          NUMBER(6)
, product_name        VARCHAR2(50)
, product_description  VARCHAR2(2000)
, category_id         NUMBER(2)
, weight_class        NUMBER(1)
, warranty_period      INTERVAL YEAR TO MONTH
, supplier_id         NUMBER(6)
, product_status      VARCHAR2(20)
, list_price          NUMBER(8,2)
, min_price            NUMBER(8,2)
, catalog_url         VARCHAR2(50)
, CONSTRAINT          product_status_lov
                      CHECK (product_status in ('orderable'
                                                  , 'planned'
                                                  , 'under development'
                                                  , 'obsolete'))
)
PARTITION BY HASH (product_id)
PARTITIONS 5
STORE IN (prod_ts1, prod_ts2, prod_ts3, prod_ts4, prod_ts5);
```

レンジ・パーティション化の例 サンプル・スキーマ sh の販売表は、レンジでパーティション化されています。次の例は、販売表を作成します（この例では、制約および記憶域要素は省略してあります）。

```
CREATE TABLE range_sales
( prod_id          NUMBER(6)
, cust_id          NUMBER
, time_id          DATE
, channel_id       CHAR(1)
, promo_id         NUMBER(6)
, quantity_sold    NUMBER(3)
, amount_sold      NUMBER(10,2)
)
PARTITION BY RANGE (time_id)
(PARTITION SALES_Q1_1998 VALUES LESS THAN (TO_DATE('01-APR-1998', 'DD-MON-YYYY')),
PARTITION SALES_Q2_1998 VALUES LESS THAN (TO_DATE('01-JUL-1998', 'DD-MON-YYYY')),
PARTITION SALES_Q3_1998 VALUES LESS THAN (TO_DATE('01-OCT-1998', 'DD-MON-YYYY')),
PARTITION SALES_Q4_1998 VALUES LESS THAN (TO_DATE('01-JAN-1999', 'DD-MON-YYYY')),
PARTITION SALES_Q1_1999 VALUES LESS THAN (TO_DATE('01-APR-1999', 'DD-MON-YYYY')),
PARTITION SALES_Q2_1999 VALUES LESS THAN (TO_DATE('01-JUL-1999', 'DD-MON-YYYY')),
PARTITION SALES_Q3_1999 VALUES LESS THAN (TO_DATE('01-OCT-1999', 'DD-MON-YYYY')),
PARTITION SALES_Q4_1999 VALUES LESS THAN (TO_DATE('01-JAN-2000', 'DD-MON-YYYY')),
PARTITION SALES_Q1_2000 VALUES LESS THAN (TO_DATE('01-APR-2000', 'DD-MON-YYYY')),
PARTITION SALES_Q2_2000 VALUES LESS THAN (TO_DATE('01-JUL-2000', 'DD-MON-YYYY')),
PARTITION SALES_Q3_2000 VALUES LESS THAN (TO_DATE('01-OCT-2000', 'DD-MON-YYYY')),
PARTITION SALES_Q4_2000 VALUES LESS THAN (MAXVALUE))
;
```

パーティション表のメンテナンス操作については、『Oracle9i データベース管理者ガイド』を参照してください。

リスト・パーティション化の例 次の文は、デモ表 oe.customers をリスト・パーティション表として作成します（この例では、デモ表の一部の列およびすべての制約は省略してあります）。

```
CREATE TABLE list_customers
( customer_id          NUMBER(6)
, cust_first_name      VARCHAR2(20)
, cust_last_name       VARCHAR2(20)
, cust_address         CUST_ADDRESS_TYP
, nls_territory        VARCHAR2(30)
, cust_email           VARCHAR2(30)
)
PARTITION BY LIST (nls_territory) (
PARTITION asia VALUES ('CHINA', 'THAILAND'),
PARTITION europe VALUES ('GERMANY', 'ITALY', 'SWITZERLAND'),
PARTITION west VALUES ('AMERICA'),
PARTITION east VALUES ('INDIA'));
```

LOB 列のあるパーティション表の例 次の文は、2つのパーティション P1 と P2、および3つの LOB 列 B、C、D を持つパーティション表 PT を作成します。

```
CREATE TABLE print_media
( product_id      NUMBER(6)
, ad_id           NUMBER(6)
, ad_composite    BLOB
, ad_sourcetext   CLOB
, ad_finaltext    CLOB
, ad_fltextn      NCLOB
, ad_textdocs_ntab textdoc_tab
, ad_photo        BLOB
, ad_graphic      BFILE
, ad_header       adheader_typ
, press_release   LONG
) NESTED TABLE ad_textdocs_ntab STORE AS textdocs_nestedtab
LOB (ad_composite, ad_photo, ad_finaltext)
  STORE AS (STORAGE (NEXT 20M))
PARTITION BY RANGE (product_id)
(PARTITION P1 VALUES LESS THAN (3000) TABLESPACE ts1
  LOB (ad_composite, ad_photo)
    STORE AS (TABLESPACE tsa STORAGE (INITIAL 20M)),
  PARTITION P2 VALUES LESS THAN (MAXVALUE)
  LOB (ad_composite, ad_finaltext)
    STORE AS (TABLESPACE tsb)
  TABLESPACE tsx;
```

```
CREATE TABLE PT (A NUMBER, B BLOB, C CLOB, D CLOB)
LOB (B,C,D) STORE AS (STORAGE (NEXT 20M))
PARTITION BY RANGE (A)
(PARTITION P1 VALUES LESS THAN (10) TABLESPACE TS1
  LOB (B,D) STORE AS (TABLESPACE TSA STORAGE (INITIAL 20M)),
  PARTITION P2 VALUES LESS THAN (20)
  LOB (B,C) STORE AS (TABLESPACE TSB)
  TABLESPACE TSX;
```

パーティション P1 は、表領域 ts1 にあります。ad_composite および ad_photo に対する LOB データ・パーティションは、表領域 tsa にあります。ad_finaltext に対する LOB データ・パーティションは、表領域 ts1 にあります。LOB 列 ad_composite および ad_photo に、記憶域属性 INITIAL を指定します。他の属性は、デフォルトの表レベルの仕様から継承されます。表レベルで指定されていないデフォルトの LOB 記憶域属性は、ad_composite 列および ad_photo 列は表領域 tsa から、ad_finaltext 列は表領域 ts1 から継承されます。LOB 索引パーティションは、対応する LOB データ・パーティションと同じ表領域に存在します。他の記憶域属性は、LOB データ・パーティションの対応する属性値および索引パーティションがある表領域のデフォルト属性に基づきます。

パーティション P2 は、デフォルトの表領域 `tsx` にあります。 `ad_composite` および `ad_finaltext` に対する LOB データは、表領域 `tsb` にあります。 `ad_photo` に対する LOB データは、表領域 `tsx` にあります。 `ad_composite` 列および `ad_finaltext` 列に対する LOB 索引は、表領域 `tsb` にあります。 `ad_photo` 列に対する LOB 索引は、表領域 `tsx` にあります。

コンポジット・パーティション表の例 14-57 ページの「レンジ・パーティション化の例」で作成した表のデータを販売時刻で分割します。時刻と販売チャネルに従って最近のデータにアクセスする場合、コンポジット・パーティション化は、より適切です。次の文は、同じ `composit_sales` 表を作成しますが、コンポジット・パーティション化しています。最新のデータがあるパーティションは、Oracle 定義とユーザー定義の両方のサブパーティション名でサブパーティション化されます。(この例では、制約および記憶域属性は省略してあります。)

```
CREATE TABLE composite_sales
( prod_id          NUMBER(6)
, cust_id          NUMBER
, time_id          DATE
, channel_id       CHAR(1)
, promo_id         NUMBER(6)
, quantity_sold    NUMBER(3)
, amount_sold      NUMBER(10,2)
)
PARTITION BY RANGE (time_id)
SUBPARTITION BY HASH (channel_id)
(PARTITION SALES_Q1_1998 VALUES LESS THAN (TO_DATE('01-APR-1998', 'DD-MON-YYYY')),
 PARTITION SALES_Q2_1998 VALUES LESS THAN (TO_DATE('01-JUL-1998', 'DD-MON-YYYY')),
 PARTITION SALES_Q3_1998 VALUES LESS THAN (TO_DATE('01-OCT-1998', 'DD-MON-YYYY')),
 PARTITION SALES_Q4_1998 VALUES LESS THAN (TO_DATE('01-JAN-1999', 'DD-MON-YYYY')),
 PARTITION SALES_Q1_1999 VALUES LESS THAN (TO_DATE('01-APR-1999', 'DD-MON-YYYY')),
 PARTITION SALES_Q2_1999 VALUES LESS THAN (TO_DATE('01-JUL-1999', 'DD-MON-YYYY')),
 PARTITION SALES_Q3_1999 VALUES LESS THAN (TO_DATE('01-OCT-1999', 'DD-MON-YYYY')),
 PARTITION SALES_Q4_1999 VALUES LESS THAN (TO_DATE('01-JAN-2000', 'DD-MON-YYYY')),
 PARTITION SALES_Q1_2000 VALUES LESS THAN (TO_DATE('01-APR-2000', 'DD-MON-YYYY')),
 PARTITION SALES_Q2_2000 VALUES LESS THAN (TO_DATE('01-JUL-2000', 'DD-MON-YYYY'))
    SUBPARTITIONS 8,
 PARTITION SALES_Q3_2000 VALUES LESS THAN (TO_DATE('01-OCT-2000', 'DD-MON-YYYY'))
    (SUBPARTITION ch_c,
     SUBPARTITION ch_i,
     SUBPARTITION ch_p,
     SUBPARTITION ch_s,
     SUBPARTITION ch_t),
 PARTITION SALES_Q4_2000 VALUES LESS THAN (MAXVALUE)
    SUBPARTITIONS 4)
;
```

オブジェクト表の例 オブジェクト型 dept_t を指定します。

```
CREATE TYPE dept_t AS OBJECT
  ( dname  VARCHAR2(100),
    address VARCHAR2(200) );
```

オブジェクト表 dept は、dept_t 型の部門オブジェクトを持ちます。

```
CREATE TABLE dept OF dept_t;
```

次の文は、ユーザー定義オブジェクト型 salesrep_t を持つオブジェクト表 salesreps を作成します。

```
CREATE OR REPLACE TYPE salesrep_t AS OBJECT
  ( repId NUMBER,
    repName VARCHAR2(64));
CREATE TABLE salesreps OF salesrep_t;
```

REF の例 次の例は、dept_t 型および表 dept (14-60 ページの「オブジェクト表の例」で作成) を使用します。この文を実行すると、有効範囲付き REF を持つ表が作成されます。

```
CREATE TABLE emp
  ( ename  VARCHAR2(100),
    enumber NUMBER,
    edept  REF dept_t SCOPE IS dept );
```

次の文は、参照制約が定義されている REF 列を含む表を作成します。

```
CREATE TABLE emp
  ( ename  VARCHAR2(100),
    enumber NUMBER,
    edept  REF dept_t REFERENCES dept);
```

ユーザー定義 OID の例 次の文は、オブジェクト型および OID が主キー・ベースの対応するオブジェクト表を作成します。

```
CREATE TYPE emp_t AS OBJECT (empno NUMBER, address CHAR(30));

CREATE TABLE emp OF emp_t (empno PRIMARY KEY)
  OBJECT IDENTIFIER IS PRIMARY KEY;
```

この後は、次の2つのいずれかの方法で emp オブジェクト表を参照できます。

```
CREATE TABLE dept (dno NUMBER,  
                    mgr_ref REF emp_t SCOPE IS emp);  
  
CREATE TABLE dept (  
    dno NUMBER,  
    mgr_ref REF emp_t CONSTRAINT mgr_in_emp REFERENCES emp);
```

タイプ列の制約の例

```
CREATE TYPE address_t AS OBJECT  
    ( hno      NUMBER,  
      street  VARCHAR2(40),  
      city    VARCHAR2(20),  
      zip     VARCHAR2(5),  
      phone   VARCHAR2(10) );  
  
CREATE TYPE person AS OBJECT  
    ( name      VARCHAR2(40),  
      dateofbirth DATE,  
      homeaddress address,  
      manager   REF person );  
  
CREATE TABLE persons OF person  
    ( homeaddress NOT NULL,  
      UNIQUE (homeaddress.phone),  
      CHECK (homeaddress.zip IS NOT NULL),  
      CHECK (homeaddress.city <> 'San Francisco') );
```

CREATE TABLESPACE

用途

CREATE TABLESPACE 文を使用すると、**表領域**を作成できます。表領域とは、永続スキーマ・オブジェクトを格納する、データベース内に割り当てられた領域です。

表領域は、最初は読み書き両用として作成されます。その後、ALTER TABLESPACE 文でその表領域をオフラインまたはオンラインに設定したり、表領域にデータ・ファイルを追加したり、読取り専用に設定することができます。

DROP TABLESPACE 文を使用して、データベースから表領域を削除することもできます。

CREATE TEMPORARY TABLESPACE 文を使用して、セッションの存続期間中のみスキーマ・オブジェクトを格納する表領域を作成できます。

注意： SYSTEM 以外の表領域のオブジェクトを使用する場合は、次の注意事項があります。

- ロールバック UNDO モードのデータベースを実行するには、1 つ以上のロールバック・セグメント（SYSTEM ロールバック・セグメント以外の）がオンラインである必要があります。
 - 自動 UNDO 管理モードのデータベースを実行するには、1 つ以上の UNDO 表領域がオンラインである必要があります。
-

参照：

- 表領域については、『Oracle9i データベース概要』を参照してください。
- 表領域の変更については、10-82 ページの「ALTER TABLESPACE」を参照してください。
- 表領域の削除については、16-9 ページの「DROP TABLESPACE」を参照してください。
- 14-73 ページの「CREATE TEMPORARY TABLESPACE」を参照してください。

前提条件

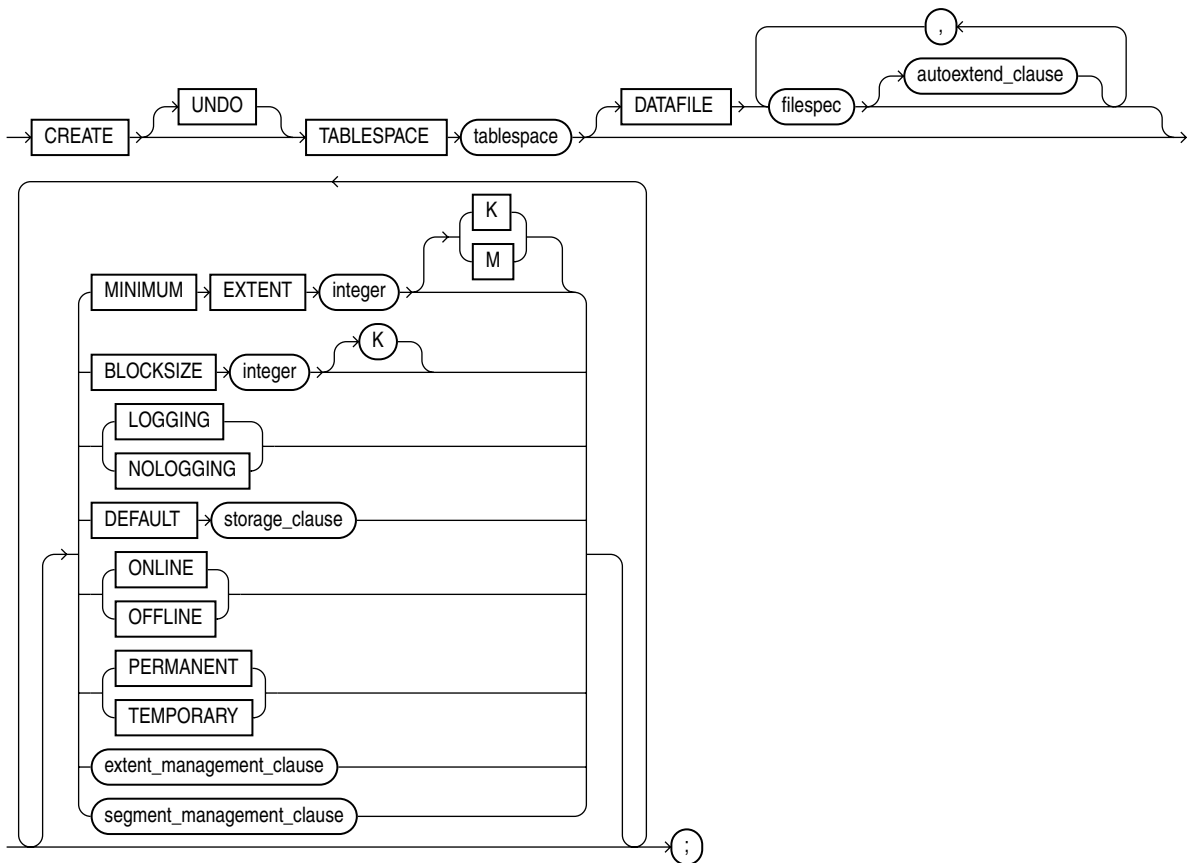
CREATE TABLESPACE システム権限が必要です。

表領域を作成する場合、表領域を格納するデータベースを作成する必要があります。また、そのデータベースをオープンしておく必要もあります。

参照： 12-20 ページの「[CREATE DATABASE](#)」を参照してください。

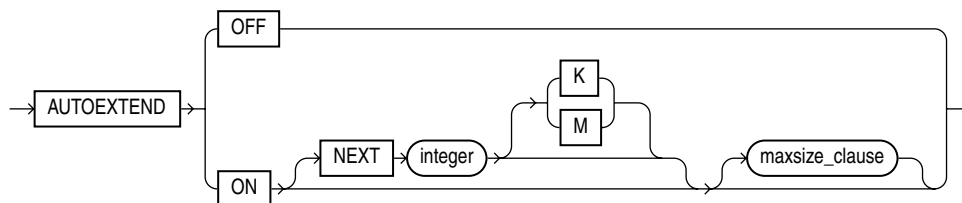
構文

create_tablespace::=

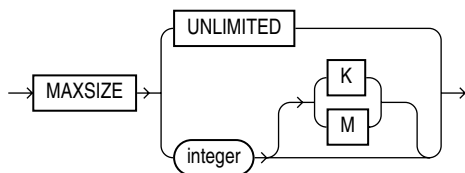


filespec: 16-27 ページの「[filespec](#)」を参照してください。

autoextend_clause::=

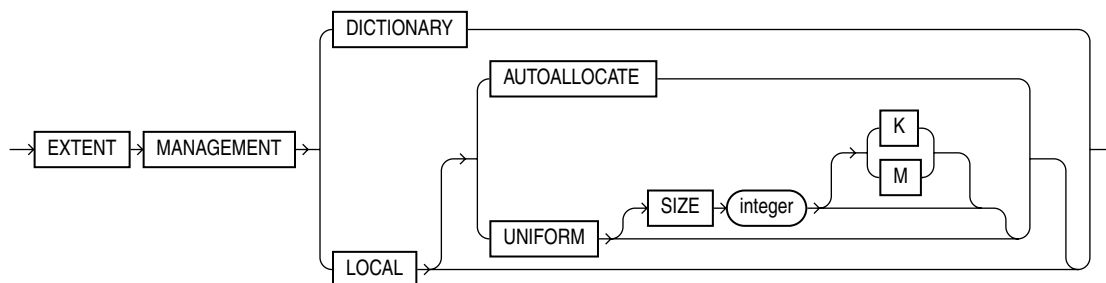


maxsize_clause::=



storage_clause: 17-50 ページの「[storage_clause](#)」を参照してください。

extent_management_clause::=



segment_management_clause::=



キーワードとパラメータ

UNDO

UNDO を指定すると、UNDO 表領域を作成できます。自動 UNDO 管理モードでデータベースを実行する場合、Oracle は、ロールバック・セグメントのかわりに UNDO 表領域を使用して UNDO 領域を管理します。自動 UNDO 管理モードで実行している場合にこの句は便利ですが、データベースは作成されません。

自動 UNDO 管理モードでデータベースを起動すると、UNDO 表領域が割り当てられます。UNDO 表領域がインスタンスに割り当てられない場合、SYSTEM ロールバック・セグメントが使用されます。UNDO 表領域の作成によってこれを回避することができ、他の UNDO 表領域がその時点で割り当てられていない場合、インスタンスに暗黙的に割り当てられます。

制限事項：

- この表領域にはデータベース・オブジェクトを作成できません。システム管理の UNDO データ用に確保されています。
- ローカル・エクステンツ管理を指定するために UNDO 表領域に対して指定できるのは、DATAFILE 句および *extent_management_clause* 句のみです。(*extent_management_clause* を使用して、ディクショナリ・エクステンツ管理を指定できません。) すべての UNDO 表領域は、永続的、読み取り / 書き込み可能およびログギング・モードで作成されます。MINIMUM EXTENT および DEFAULT STORAGE に対する値は、システムで生成されます。

参照：

- 自動 UNDO 管理および UNDO 表領域の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。
- データベース作成中の暗黙的または明示的な UNDO 表領域の作成については、12-20 ページの「[CREATE DATABASE](#)」を参照してください。
- UNDO 表領域の変更の詳細は、「[ALTER TABLESPACE](#)」を参照してください。
- UNDO 表領域の削除の詳細は、「[DROP TABLESPACE](#)」を参照してください。
- UNDO_MANAGEMENT パラメータを使用した自動 UNDO 管理モードのデータベース・インスタンスをオープンする場合の詳細は、『Oracle9i データベース・リファレンス』を参照してください。

tablespace

作成する表領域の名前を指定します。

DATAFILE *filespec*

表領域を構成するデータ・ファイルまたはファイルを指定します。

注意： RAW デバイスをサポートするオペレーティング・システムの場合、*filespec* の REUSE キーワードには、RAW デバイスをデータ・ファイルとして指定する際の意味はありません。このような CREATE TABLESPACE 文は、REUSE の指定の有無にかかわらず実行されます。

DATAFILE 句は、初期化パラメータ DB_CREATE_FILE_DEST を設定する場合のみに使用するオプションです。この場合、パラメータで指定したデフォルトのファイル出力先に、100MB のファイルがシステム名で作成されます。

参照： データ・ファイル指定の詳細は、16-27 ページの「*filespec*」を参照してください。

autoextend_clause

autoextend_clause は、新しいデータ・ファイルまたはテンポラリ・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、これらのファイルは自動的に拡張されません。

ON ON を指定すると、自動拡張を使用可能にします。

OFF OFF を指定すると、自動拡張を使用禁止にします。

注意： 自動拡張を使用禁止にする場合は、NEXT および MAXSIZE の値を 0（ゼロ）に設定します。後続の文で自動拡張を使用可能に戻す場合は、これらの値を設定しなおす必要があります。

NEXT NEXT 句を使用すると、エクステントがさらに必要になった場合にデータ・ファイルに自動的に割り当てられるディスク領域の増分サイズを、バイト単位で指定できます。K または M を使用すると、KB または MB 単位で指定できます。デフォルトのサイズは 1 データ・ブロックです。

MAXSIZE MAXSIZE を使用すると、データ・ファイルの自動拡張で使用されるディスク領域の最大サイズを指定できます。

UNLIMITED データ・ファイルまたはテンポラリ・ファイルに割り当てられるディスク領域を制限しない場合は、UNLIMITED 句を使用します。

MINIMUM EXTENT 句

表領域で使用されるエクステンツの最小サイズを指定します。この句によって、表領域内のすべての使用済エクステンツまたは未使用エクステンツの大きさが、*integer* で指定したサイズ以上であり、その倍数になるように指定することによって、表領域における空き領域の断片化を制御します。

注意： ディクショナリ管理の一時表領域の場合、この句は関係ありません。

参照： MINIMUM EXTENT を使用した断片化制御については、『Oracle9i データベース概要』を参照してください。

BLOCKSIZE 句

BLOCKSIZE 句を使用すると、表領域に対して標準以外のブロック・サイズを指定できます。この句を指定するには、DB_CACHE_SIZE および 1 つ以上の DB_nK_CACHE_SIZE パラメータが設定されている必要があります。また、この句で指定する整数は、DB_nK_CACHE_SIZE パラメータの 1 つの設定と対応している必要があります。

制限事項： 一時表領域（TEMPORARY を指定する場合）の場合、または表領域を一時表領域として任意のユーザーに割り当てる場合は、標準以外のブロック・サイズを指定できません。ユーザーを作成または変更するときに指定した場合、Oracle からエラーは戻されません。ただし、ユーザーが行う後続の操作で一時セグメントが必要な場合は、エラーになります。

参照： バッファ・キャッシュでの複数のブロック・サイズの混在、およびパーティション・オブジェクトでの複数のブロック・サイズの使用に関する制限事項については、『Oracle9i データベース管理者ガイド』を参照してください。

LOGGING | NOLOGGING

表領域内のすべての表、索引および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。デフォルトは LOGGING です。

表レベル、索引レベルおよびパーティション・レベルでのロ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オーバーライドできます。

次の操作でのみ、NOLOGGING モードがサポートされます。

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

NOLOGGING モードでは、データの変更時に、(新しいエクステンツを INVALID としてマーク設定し、ディクショナリの変更を記録するために) 最小限のログが記録されます。メディア・リカバリ中に NOLOGGING が適用された場合、REDO データのログ記録が中断されるため、エクステン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、損失してはならないオブジェクトの場合は、NOLOGGING 操作の後にバックアップを取る必要があります。

DEFAULT *storage_clause*

表領域内に作成されるすべてのオブジェクトに対するデフォルトの記憶域パラメータを指定します。

ディクショナリ管理の一時表領域の場合、*storage_clause* の NEXT パラメータのみを考慮します。

参照： 記憶域パラメータについては、17-50 ページの「[storage_clause](#)」を参照してください。

ONLINE | OFFLINE 句

ONLINE ONLINE を指定して、表領域に対するアクセス権限を付与されているユーザーに対して、作成直後の表領域を使用可能にします。これはデフォルトです。

OFFLINE OFFLINE を指定して、作成直後の表領域を使用禁止にします。

データ・ディクショナリ・ビュー DBA_TABLESPACES は、各表領域がオンラインまたはオフラインのいずれであることを示します。

PERMANENT | TEMPORARY 句

PERMANENT 表領域を永続オブジェクトの格納に使用する場合は、PERMANENT を指定します。これはデフォルトです。

TEMPORARY 表領域を一時オブジェクトの格納にのみ使用する場合は、TEMPORARY を指定します。たとえば、ORDER BY 句の処理での暗黙的なソートに使用されるセグメントの格納などです。

制限事項： TEMPORARY を指定する場合、EXTENT MANAGEMENT LOCAL または BLOCKSIZE 句は指定できません。

extent_management_clause

`extent_management_clause`によって、表領域のエクステントの管理方法を指定できます。

注意： この句でエクステントの管理を指定した場合は、表領域を移行しないかぎり、エクステントの管理を変更できません。

- 表領域をローカルで管理する場合は、`LOCAL` を指定します。ローカル管理の表領域には、ビットマップ用に確保しておいた表領域が一部あります。これはデフォルトです。
- `AUTOALLOCATE` は、表領域がシステム管理されることを指定します。ユーザーはエクステント・サイズを指定できません。初期化パラメータ `COMPATIBLE` が 9.0.0 以上に設定されている場合は、これがデフォルトです。
- `UNIFORM` は、表領域が `SIZE` バイトの均一のエクステントで管理されることを指定します。`K` または `M` を使用して、`KB` または `MB` 単位でエクステント・サイズを指定することもできます。`SIZE` のデフォルト値は `1MB` です。
- ディクショナリ表を使用して表領域を管理する場合は、`DICTIONARY` を指定します。初期化パラメータ `COMPATIBLE` が 9.0.0 未満に設定されている場合は、これがデフォルトです。

`extent_management_clause` を指定しない場合、`COMPATIBLE` の設定、`MINIMUM EXTENT` 句および `DEFAULT storage_clause` が解析され、エクステント管理が判断されます。

- 初期化パラメータ `COMPATIBLE` が 9.0.0 未満の場合は、ディクショナリ管理の表領域が作成されます。初期化パラメータ `COMPATIBLE` が 9.0.0 以上の場合は、表領域はローカル管理となります。
- `DEFAULT storage_clause` を指定しない場合、ローカル管理の自動割当て表領域が作成されます。
- `DEFAULT storage_clause` を指定する場合は、次のようになります。
 - `MINIMUM EXTENT` 句を指定した場合、`MINIMUM EXTENT`、`INITIAL` および `NEXT` の値が等しく、`PCT_INCREASE` の値が 0（ゼロ）であるかどうかの評価されます。その場合、エクステントのサイズが `INITIAL` に等しい、ローカル管理の均一な表領域が作成されます。`MINIMUM EXTENT`、`INITIAL` および `NEXT` パラメータが等しくない場合、または `PCT_INCREASE` が 0（ゼロ）でない場合は、指定したエクステントの記憶域パラメータが無視され、ローカル管理の自動割当て表領域が作成されます。
 - `MINIMUM EXTENT` 句を指定しない場合、`INITIAL` および `NEXT` が等しく、`PCT_INCREASE` が 0（ゼロ）かどうかということのみが評価されます。その場合、表領域はローカル管理され、エクステント・サイズは均一です。そうでない場合、表領域はローカル管理され、エクステント・サイズは自動割当てされます。

参照： ローカル管理の表領域については、『Oracle9i データベース概要』を参照してください。

制限事項：

- 永続的なローカル管理の表領域には、永続オブジェクトを格納できます。ローカル管理の表領域に一時オブジェクトを格納する必要がある場合（たとえば、ユーザーの一時表領域に割り当てる場合）は、CREATE TEMPORARY TABLESPACE 文を使用します。
- LOCAL を指定する場合、DEFAULT *storage_clause*、MINIMUM EXTENT または TEMPORARY は指定できません。

参照： 表領域の移行によってエクステンツの管理を変更する場合は、『Oracle9i データベース移行ガイド』を参照してください。

segment_management_clause

segment_management_clause は、永続的で、ローカル管理の表領域のみに関連します。Oracle が空きリストまたはビットマップを使用する表領域のセグメントにある使用済領域および空き領域を追跡するかどうかを指定できます。

MANUAL Oracle が空きリストを使用して表領域のセグメントにある空き領域を管理する場合は、MANUAL を指定します。

AUTO Oracle がビットマップを使用して表領域のセグメントにある空き領域を管理する場合は、AUTO を指定します。AUTO を指定すると、この表領域のオブジェクトに対して後で指定する記憶域仕様 FREELIST および FREELIST GROUPS は無視されます。この設定を、**自動セグメント領域管理**といいます。

既存の表領域のセグメント管理を確認するには、DBA_TABLESPACES または USER_TABLESPACES データ・ディクショナリ・ビューの SEGMENT_SPACE_MANAGEMENT 列を問い合わせます。

注意： AUTO を指定する場合、次のことに注意します。

- エクステンツ管理を LOCAL UNIFORM に設定する場合は、各エクステンツに 5 以上のデータベース・ブロックがあり、データベースのブロック・サイズが指定されていることを確認する必要があります。
 - エクステンツ管理を LOCAL AUTOALLOCATE に設定する場合、およびデータベースのブロック・サイズが 16KB 以上の場合は、最小で 1MB のエクステンツが作成され、セグメント領域管理が管理されます。
-
-

AUTO の制限事項

- この句は、永続的なローカル管理の表領域に指定できません。
- この句は、SYSTEM 表領域に指定できません。
- AUTO を指定したセグメント管理の表領域には、LOB を格納できません。

参照：

- 自動セグメント領域管理およびその使用については、『Oracle9i データベース管理者ガイド』を参照してください。
- データ・ディクショナリ・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

例

次の例は、データ・ファイル undotbs_1a.f を持つ、10MB の UNDO 表領域 undots1 を作成します。

```
CREATE UNDO TABLESPACE undots1
  DATAFILE 'undotbs_1a.f'
  SIZE 10M AUTOEXTEND ON;
```

DEFAULT 記憶域の例 次の文は、1 つのデータ・ファイルを持つ tbs_1 という名前の表領域を作成します。

```
CREATE TABLESPACE tbs_1
  DATAFILE 'tabspace_file2.dat' SIZE 20M
  DEFAULT STORAGE (INITIAL 10K NEXT 50K
                   MINEXTENTS 1 MAXEXTENTS 999)
  ONLINE;
```

AUTOEXTEND の例 次の文は、1 つのデータ・ファイルを持つ tbs_2 という名前の表領域を作成します。さらに領域が必要な場合、500KB のエクステントが最大サイズ 10MB まで追加されます。

```
CREATE TABLESPACE tbs_2
  DATAFILE 'diskb:tabspace_file5.dat' SIZE 500K REUSE
  AUTOEXTEND ON NEXT 500K MAXSIZE 10M;
```

MINIMUM EXTENT の例 次の文は、1つのデータ・ファイルを持つ `tbs_3` という名前の表領域を作成し、すべてのエクステントを 64KB の倍数として割り当てます。

```
CREATE TABLESPACE tbs_3
  DATAFILE 'tabspace_file3.dbf' SIZE 2M
  MINIMUM EXTENT 64K
  DEFAULT STORAGE (INITIAL 128K NEXT 128K)
  LOGGING;
```

ローカル管理の例 次の文は、データベース・ブロック・サイズが 2KB であると仮定します。

```
CREATE TABLESPACE tbs_4 DATAFILE 'file_1.f' SIZE 10M
  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K;
```

この文で、すべてのエクステントが 128KB で、ビットマップの各ビットが 64 ブロックを示す、ローカル管理の表領域を作成します。

セグメント管理の例 次の例は、自動セグメント領域管理の表領域を作成します。

```
CREATE TABLESPACE auto_seg_ts DATAFILE 'file_2.f' SIZE 1M
  EXTENT MANAGEMENT LOCAL
  SEGMENT SPACE MANAGEMENT AUTO;
```

Oracle 管理ファイルの例 次の例は、データ・ファイルを作成するデフォルトの位置を設定し、デフォルトの位置にあるデータ・ファイルを持つ表領域を作成します。データ・ファイルは 100MB で自動拡張可能であり、最大サイズの制限がありません。

```
ALTER SYSTEM SET DB_CREATE_FILE_DEST = 'u01/oradata/sample';
CREATE TABLESPACE omf_ts1;
```

次の例は、Oracle 管理の、自動拡張されない 100MB のデータ・ファイルを持つ表領域を作成します。

```
CREATE TABLESPACE omf_ts2 DATAFILE AUTOEXTEND OFF;
```

CREATE TEMPORARY TABLESPACE

用途

CREATE TEMPORARY TABLESPACE 文を使用すると、**一時表領域**を作成できます。一時表領域は、セッションの存続期間中に、スキーマ・オブジェクトを含むことができるデータベース中の領域の割当てです。後でこの一時表領域を特定のユーザーに割り当てる場合、そのユーザーが開始するトランザクションのソート操作用にこの表領域が使用されます。

CREATE TABLESPACE 文を使用して、永続スキーマ・オブジェクトを含む表領域を作成します。

注意： メディア・リカバリはテンポラリ・ファイルを認識しません。

参照：

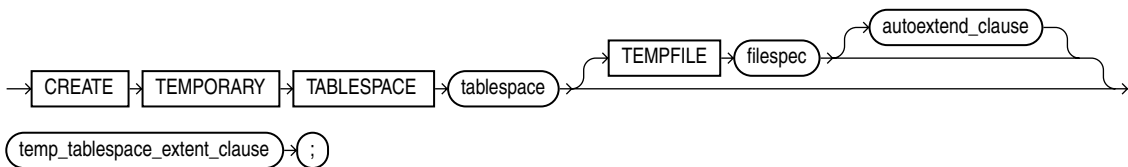
- 永続スキーマ・オブジェクトを格納する表領域の作成については、14-62 ページの「[CREATE TABLESPACE](#)」を参照してください。
- ユーザーに対する一時表領域の割当てについては、15-30 ページの「[CREATE USER](#)」を参照してください。

前提条件

CREATE TABLESPACE システム権限が必要です。

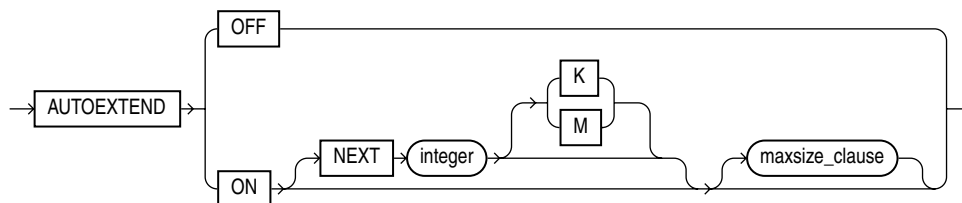
構文

create_temporary_tablespace::=

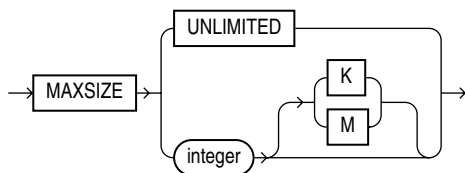


filespec: 16-27 ページの「[filespec](#)」を参照してください。

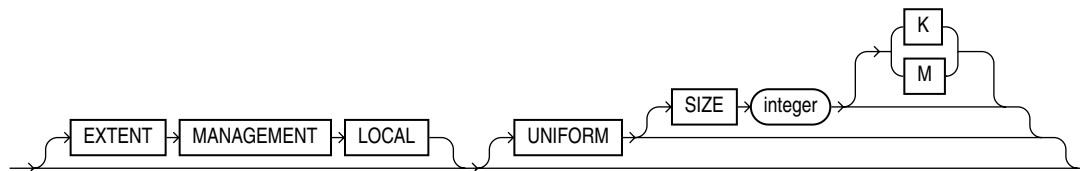
autoextend_clause::=



maxsize_clause::=



temp_tablespace_extent::=



キーワードとパラメータ

tablespace

一時表領域の名前を指定します。

TEMPFILE *filespec*

表領域を構成するテンポラリ・ファイルを指定します。

初期化パラメータ `DB_CREATE_FILE_DEST` が設定されている場合のみ、`TEMPFILE` 句を省略できます。この場合、パラメータで指定するデフォルトのファイル格納先に、Oracle 管理の 100MB のテンポラリ・ファイルが作成されます。このパラメータが設定されていない場合、`TEMPFILE` 句を指定する必要があります。

注意： オペレーティング・システムによっては、テンポラリ・ファイルのブロックが実際にアクセスされるまで、テンポラリ・ファイル用の領域が割り当てられない場合があります。領域の割当ての遅延のため、前もってテンポラリ・ファイルの作成およびサイズ変更を行う必要があります。ただし、後でテンポラリ・ファイルが使用されるときに、十分なディスク領域を使用可能にする必要があります。ご使用のオペレーティング・システムのドキュメントを参照し、このような方法でテンポラリ・ファイル領域が割り当てられるかどうかを判断してください。

参照： 16-27 ページの「[filespec](#)」を参照してください。

autoextend_clause

autoextend_clause は、新しいデータ・ファイルまたはテンポラリ・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、これらのファイルは自動的に拡張されません。

ON ON を指定すると、自動拡張を使用可能にします。

OFF OFF を指定すると、自動拡張を使用禁止にします。

注意： 自動拡張を使用禁止にする場合は、NEXT および MAXSIZE の値を 0（ゼロ）に設定します。後続の文で自動拡張を使用可能に戻す場合は、これらの値を設定しなおす必要があります。

NEXT NEXT 句を使用すると、エクステン트가さらに必要になった場合にデータ・ファイルに自動的に割り当てられるディスク領域の増分サイズを、バイト単位で指定できます。K または M を使用すると、KB または MB 単位で指定できます。デフォルトのサイズは 1 データ・ブロックです。

MAXSIZE MAXSIZE を使用すると、データ・ファイルの自動拡張で使用されるディスク領域の最大サイズを指定できます。

UNLIMITED データ・ファイルまたはテンポラリ・ファイルに割り当てられるディスク領域を制限しない場合は、UNLIMITED 句を使用します。

temp_tablespace_extent

temp_tablespace_extent 句を使用すると、表領域の管理方法を指定できます。

EXTENT MANAGEMENT LOCAL この句は、表領域の一部がビットマップ用に確保されることを示します。すべての一時表領域にはローカル管理のエクステントがあるため、この句はオプションです。

UNIFORM 一時表領域のエクステントはすべて同じサイズ（均一）で、キーワードはオプションです。ただし、**SIZE** を指定する場合は、**UNIFORM** を指定する必要があります。

SIZE integer 表領域のエクステントのサイズをバイト単位で指定します。K または M を使用して、KB または MB 単位で指定することもできます。

SIZE を指定しない場合は、デフォルトのエクステント・サイズ **1MB** を使用します。

参照： ローカル管理の表領域の詳細は、『Oracle9i データベース概要』を参照してください。

例

一時表領域の例 この文は、サンプル・データベースのデータベース・ユーザーに対するデフォルトの一時表領域を作成します。

```
CREATE TEMPORARY TABLESPACE temp
    TEMPFILE 'temp01.dbf' SIZE 5M AUTOEXTEND ON;
```

デフォルトのデータベース・ブロック・サイズを **2KB** とした場合、マップの各ビットは **1** つのエクステントを表し、各ビットは **8000** ブロックをマップします。

次の例は、データ・ファイルを作成するデフォルトの位置を設定し、デフォルトの位置にある Oracle 管理テンポラリ・ファイルを持つ表領域を作成します。テンポラリ・ファイルは **100MB** で自動拡張可能であり、最大サイズの制限がありません（Oracle 管理ファイルのデフォルト値）。

```
ALTER SYSTEM SET DB_CREATE_FILE_DEST = '/u01/oradata/sample';

CREATE TEMPORARY TABLESPACE omf_tempts1;
```

CREATE TRIGGER

用途

CREATE TRIGGER 文を使用すると、次のデータベース・トリガーを作成および使用可能にすることができます。

- 表、スキーマまたはデータベースに対応したストアド PL/SQL ブロック
- 無名 PL/SQL ブロック、あるいは PL/SQL または JAVA で実装されているプロシージャへのコール

指定された条件が発生した場合、トリガーは自動的に実行されます。

トリガーを作成した場合、そのトリガーは自動的に使用可能になります。この後、DISABLE 句および ENABLE 句を指定した ALTER TRIGGER 文または ALTER TABLE 文を使用して、トリガーを使用禁止または使用可能にすることができます。

参照：

- トリガーの様々な型については、『Oracle9i データベース概要』を参照してください。
- 前述の用途のためにトリガーを設計する方法については、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。
- トリガーの使用可能化、使用禁止化またはコンパイルの詳細は、11-2 ページの「ALTER TRIGGER」および 10-2 ページの「ALTER TABLE」を参照してください。
- トリガーの削除については、16-12 ページの「DROP TRIGGER」を参照してください。

前提条件

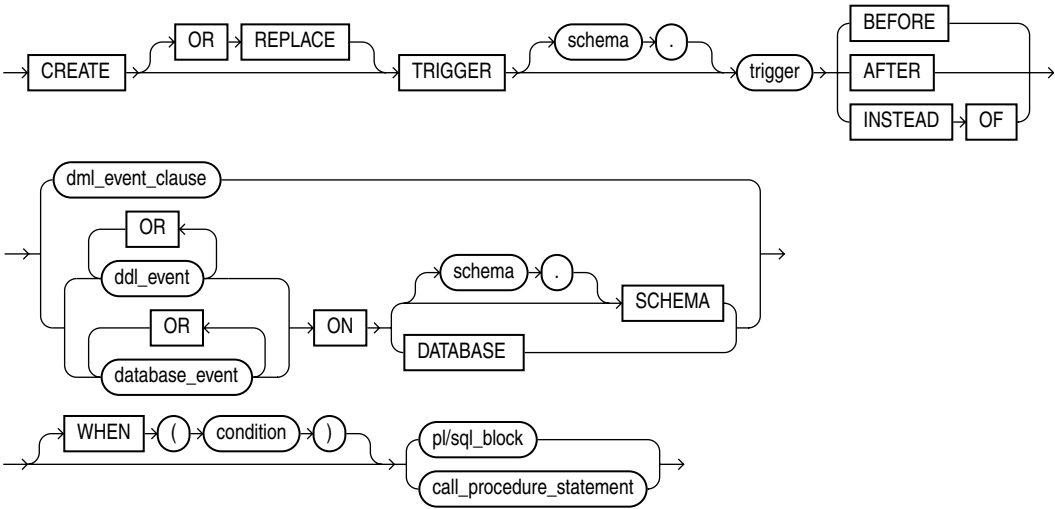
トリガーを作成する前に、ユーザー SYS は、DBMSSTD_X.SQL と呼ばれる SQL スクリプトを実行する必要があります。このスクリプトの正確なファイル名および位置は、使用するオペレーティング・システムによって異なります。

- 自分のスキーマ内の表または自分のスキーマ (SCHEMA) に対するトリガーを自分のスキーマ内に作成する場合は、CREATE TRIGGER 権限が必要です。
- 任意のスキーマ内の表または別のユーザーのスキーマ (schema.SCHEMA) に対するトリガーを任意のスキーマ内に作成する場合は、CREATE ANY TRIGGER 権限が必要です。
- 前述の権限の他にも、データベースに対するトリガーを作成する場合は、ADMINISTER DATABASE TRIGGER システム権限が必要です。

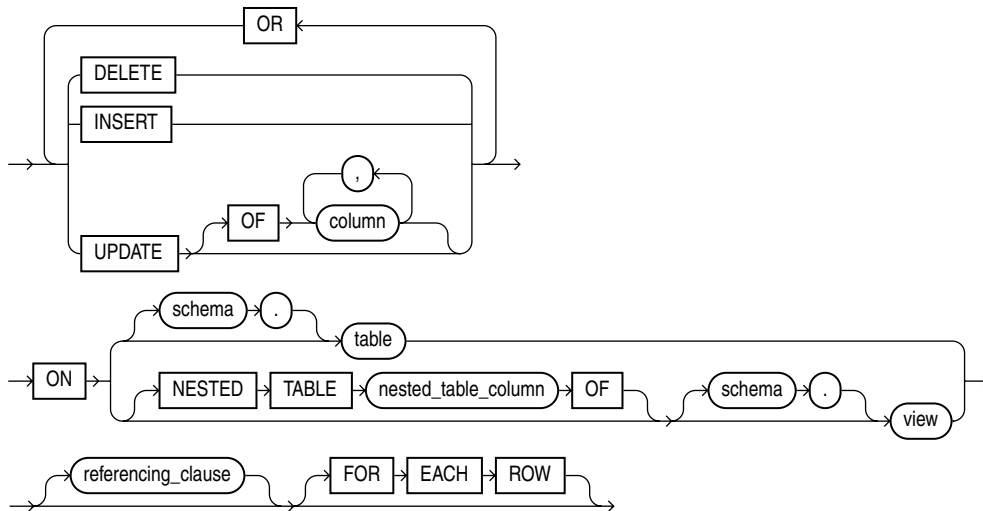
トリガーが SQL 文を発行、またはプロシージャやファンクションをコールする場合、そのトリガーの所有者には、これらの操作を行うための権限が必要です。これらの権限はロールを介して付与するのではなく、所有者に直接付与する必要があります。

構文

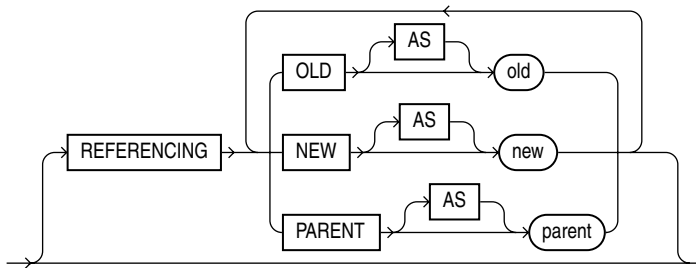
create_trigger::=



dml_event_clause::=



referencing_clause::=



キーワードとパラメータ

OR REPLACE

既存のトリガーを再作成する場合は、`OR REPLACE` を指定します。この句を指定すると、既存のトリガーの定義を削除しなくても変更できます。

schema

作成するトリガーが含まれるスキーマを指定します。*schema* を指定しない場合、自分のスキーマにトリガーが作成されます。

trigger

作成するトリガーの名前を指定します。

トリガーのコンパイル・エラーが発生した場合、このトリガーは作成されますが実行時に正常に処理されません。この場合、コンパイル・エラーになったトリガーが使用禁止にされるか、コンパイル・エラーのないバージョンに置き換えられるか、または削除されるまでは、トリガーを起動するすべての DML 文は実行できません。SQL*Plus コマンド SHOW ERRORS を使用して、関連するコンパイラ・エラー・メッセージを表示できます。

注意： マテリアライズド・ビューの実表にトリガーを作成する場合は、トリガーがマテリアライズド・ビューのリフレッシュ中に起動しないようにする必要があります（リフレッシュ中、DBMS_MVIEW プロシージャ I_AM_A_REFRESH は TRUE を戻します）。

BEFORE

BEFORE を指定すると、トリガーを起動するイベントが実行される前に、トリガーが起動されます。行レベル・トリガーの場合、対象となる各行が変更される前に起動されます。

制限事項：

- ビューおよびオブジェクト・ビューに対しては BEFORE トリガーを指定できません。
- LOB 列に BEFORE トリガーを定義した場合、:OLD 値の読取りはできますが、:NEW 値の読取りはできません。:OLD 値も :NEW 値も書込みはできません。

AFTER

AFTER を指定すると、トリガーを起動するイベント実行後に、トリガーが起動されます。行レベル・トリガーの場合、対象となる各行が変更された後に起動されます。

制限事項：

- ビューおよびオブジェクト・ビューに対しては AFTER トリガーを指定できません。
- LOB 列に AFTER トリガーを定義した場合、:OLD 値の読取りはできますが、:NEW 値の読取りはできません。:OLD 値も :NEW 値も書込みはできません。

注意： 表に対するマテリアライズド・ビュー・ログを作成した場合、その表に AFTER ROW トリガーが暗黙的に作成されます。このトリガーは、INSERT 文、UPDATE 文または DELETE 文で表のデータが変更された場合、マテリアライズド・ビュー・ログに 1 行を挿入します。複数の行レベル・トリガーを起動する順序は制御できません。マテリアライズド・ビューの内容に影響するトリガーは記述しないでください。

参照： マテリアライズド・ビュー・ログの詳細は、13-29 ページの「[CREATE MATERIALIZED VIEW LOG](#)」を参照してください。

INSTEAD OF

INSTEAD OF を指定すると、トリガーを起動するイベントのかわりに、トリガーが起動されます。INSTEAD OF トリガーは、ビューの DML イベントに有効です。DDL またはデータベース・イベントには有効ではありません。

ビューが更新可能であり、そのビューに INSTEAD OF トリガーがある場合、トリガーが優先されます。つまり、ビューで DML を実行するかわりに、Oracle はトリガーを起動します。

ビューが階層に属する場合、サブビューはトリガーを継承しません。

制限事項：

- INSTEAD OF トリガーは、ビューのみに有効です。INSTEAD OF トリガーは表には指定できません。
- あるビューに INSTEAD OF トリガーがある場合、そのビューで作成されるビューが更新可能であっても、そのビューには INSTEAD OF トリガーがある必要があります。
- LOB 列に INSTEAD OF トリガーを定義した場合、:OLD 値および :NEW 値は、どちらも読取りはできますが、書込みはできません。

注意： 同一の表に対する同一の文について、同じ型 (BEFORE、AFTER または INSTEAD OF) のトリガーを複数作成できます。これらのトリガーが起動される順番は不確定です。アプリケーションが同じ文を対象とした同じ種類のトリガーを続けて起動する必要がある場合、これらのトリガーを 1 つのトリガーに結合し、適切な順序で元のトリガーのトリガー・アクションを実行するようにしてください。

dml_event_clause

dml_event_clause によって、トリガーを起動する 3 つの DML 文のうちの 1 つを指定できます。Oracle は、既存のユーザー・トランザクションでトリガーを起動します。

DELETE

DELETE 文が表から行を削除する、またはネストした表から要素を削除するときにトリガーを起動する場合は、DELETE を指定します。

INSERT

INSERT 文が表に行を挿入する、またはネストした表に要素を挿入するときにトリガーを起動する場合は、INSERT を指定します。

UPDATE

OF 句の後に指定した列のいずれかの値を UPDATE 文で変更したときに、トリガーを起動する場合は、UPDATE を指定します。OF 句を省略した場合、UPDATE 文で表またはネストした表の任意の列の値を変更するたびにトリガーが起動されます。

UPDATE トリガーでは、オブジェクト型、VARRAY 型および REF 型の列を OF 句の後に指定した場合、UPDATE 文によってこれらの列のいずれかの値が変更される場合にトリガーが起動されます。ただし、トリガー自体の列の値は変更できません。

注意： Oracle Call Interface (OCI) ファンクションまたは 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、その列または属性を含む表に定義されているトリガーは起動されません。

制限事項：

- UPDATE OF は、INSTEAD OF トリガーに指定できません UPDATE 文でビューの任意の列の値を変更した場合、INSTEAD OF トリガーが起動されます。
- UPDATE OF は、ネストした表または LOB 列に指定できません。

参照： ビューに対する挿入、更新または削除を禁止する構造体については、15-37 ページの「[CREATE VIEW](#)」の「AS subquery」を参照してください。

ネストした表の列に DML 操作を直接実行した場合、そのネストした表の列が入っている表に定義されているトリガーは起動されません。

ddl_event

トリガーを起動する 1 つ以上の DDL 文を指定します。特に指定がないかぎり、DATABASE または SCHEMA のイベント用のトリガーを作成できます。これらのイベント用の BEFORE トリガーおよび AFTER トリガーを作成できます。Oracle は、既存のユーザー・トランザクションでトリガーを起動します。

制限事項：PL/SQL プロシージャを介して実行された DDL 操作は、トリガー・イベントとして指定できません。

次の ddl_event 値が有効です。

ALTER ALTER を指定すると、ALTER 文でデータ・ディクショナリのデータベース・オブジェクトを変更した場合、トリガーが起動されます。

制限事項：トリガーは、ALTER DATABASE 文では起動されません。

ANALYZE ANALYZE を指定すると、統計情報が収集または削除された場合、またはデータベース・オブジェクトの構造が検証された場合、トリガーが起動されます。

ASSOCIATE STATISTICS ASSOCIATE STATISTICS を指定すると、統計タイプがデータベース・オブジェクトと関連付けられた場合、トリガーが起動されます。

AUDIT AUDIT を指定すると、SQL 文の状態変化またはスキーマ・オブジェクトに対する操作が追跡された場合、トリガーが起動されます。

COMMENT COMMENT を指定すると、データベース・オブジェクトに対するコメントがデータ・ディクショナリに追加された場合、トリガーが起動されます。

CREATE CREATE を指定すると、CREATE 文でデータ・ディクショナリに新しいデータベース・オブジェクトを追加した場合、トリガーが起動されます。

制限事項：トリガーは、CREATE DATABASE 文または CREATE CONTROLFILE 文では起動されません。

DISASSOCIATE STATISTICS DISASSOCIATE STATISTICS を指定すると、統計タイプがデータベース・オブジェクトとの関連付けを解除した場合、トリガーが起動されます。

DROP DROP を指定すると、DROP 文でデータ・ディクショナリのデータベース・オブジェクトを削除した場合、トリガーが起動されます。

GRANT GRANT を指定すると、ユーザーが、別のユーザーまたはロールにシステム権限、ロールまたはオブジェクト権限を付与した場合、トリガーが起動されます。

NOAUDIT NOAUDIT を指定すると、NOAUDIT 文で SQL 文またはスキーマ・オブジェクトの操作の追跡を停止させた場合、トリガーが起動されます。

RENAME RENAME を指定すると、RENAME 文でデータベース・オブジェクトの名前を変更した場合、トリガーが起動されます。

REVOKE REVOKE を指定すると、REVOKE 文でユーザーまたはロールからシステム権限、ロールまたはオブジェクト権限を取り消した場合、トリガーが起動されます。

TRUNCATE TRUNCATE を指定すると、TRUNCATE 文で表またはクラスタから行を削除し、記憶特性を再設定した場合、トリガーが起動されます。

DDL DDL を指定すると、前述の DDL 文のいずれかを発行した場合、トリガーが起動されます。

database_event

トリガーを起動する 1 つ以上のデータベースの状態を指定します。特に指定がないかぎり、DATABASE または SCHEMA のイベント用のトリガーを作成できます。これらのトリガー・イベントについて、Oracle が自律型トランザクションの有効範囲をオープンし、トリガーを起動して、(既存のユーザー・トランザクションには関係なく) 別のトランザクションをコミットします。

SERVERERROR SERVERERROR を指定すると、サーバー・エラー・メッセージがログされた場合、トリガーが起動されます。

次のエラーが発生しても、SERVERERROR トリガーは起動されません (*string* には文字列が入ります)。

- ORA-01403: データが見つかりません。
- ORA-01422: 完全フェッチが要求よりもよりも多くの行を返しました
- ORA-01423: 完全フェッチで余分な行をチェック中にエラーが発生しました
- ORA-01034: Oracle は使用できません。
- ORA-04030: *string* バイト (*string, string*) を割り当てようとしてプロセス・メモリーが不足しました。

LOGON LOGON を指定すると、クライアント・アプリケーションがデータベースにログオンした場合、トリガーが起動されます。

LOGOFF LOGOFF を指定すると、クライアント・アプリケーションがデータベースからログオフした場合、トリガーが起動されます。

STARTUP STARTUP を指定すると、データベースがオープンされた場合、トリガーが起動されます。

SHUTDOWN SHUTDOWN を指定すると、データベースのインスタンスが停止された場合、トリガーが起動されます。

SUSPEND SUSPEND を指定すると、サーバー・エラーによってトランザクションが停止した場合、トリガーが起動されます。

注意：

- LOGON、STARTUP、SERVERERROR および SUSPEND に適用されるのは AFTER トリガーのみです。
 - LOGOFF および SHUTDOWN に適用されるのは BEFORE トリガーのみです。
 - DATABASE のみに適用されるのは AFTER STARTUP トリガーおよび BEFORE SHUTDOWN トリガーです。
-
-

参照： 自律型トランザクションの有効範囲の詳細は、『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。

ON table | view

ON 句によって、トリガーが作成されるデータベース・オブジェクトを決定できます。

table | view

トリガーを作成する、次のいずれかのスキーマ、表またはビューの名前を指定します。

- 表またはビュー
- オブジェクト表またはオブジェクト・ビュー
- ネストした表型の列

schema を指定しない場合、この表が自分のスキーマに存在するとみなされます。トリガーは、索引構成表に作成できます。

制限事項：トリガーは、SYS スキーマ内の表には作成できません。

NESTED TABLE 句

ビューの列 *nested_table_column* に定義するトリガーを指定します。そのようなトリガーは、DML の操作対象がネストした表の要素である場合にのみ起動されます。

制限事項：NESTED TABLE は、INSTEAD OF トリガーにのみ指定できます。

DATABASE

DATABASE を指定すると、データベース全体にトリガーを定義します。

SCHEMA

SCHEMA を指定すると、現行のスキーマにトリガーを定義します。

referencing_clause

referencing_clause によって、関連名を指定できます。関連名は、特に現行の行における新旧の値を参照する場合に、PL/SQL ブロックおよび行レベル・トリガーの WHEN 条件で使います。デフォルトの関連名は OLD および NEW です。行レベル・トリガーを OLD または NEW という表に対応付ける場合は、この句を使用して異なる関連名を指定します。これにより、表名と関連名との混乱を避けることができます。

- トリガーがネストした表に定義される場合、OLD および NEW はネストした表の行を参照し、PARENT は親である表の現行の行を参照します。
- オブジェクト表またはビューにトリガーが定義されている場合、OLD および NEW はオブジェクト・インスタンスを参照します。

制限事項：*referencing_clause* は、CREATE DDL イベントの INSTEAD OF トリガーには無効です。

FOR EACH ROW

FOR EACH ROW を指定すると、トリガーを行トリガーとして設定します。行レベル・トリガーは、トリガーを起動する文の対象になり、かつ WHEN 条件で定義したオプションのトリガー制約を満たす行ごとに 1 回ずつ起動されます。

INSTEAD OF トリガー以外のトリガーを指定する際に、この句を省略した場合、そのトリガーは文レベル・トリガーになります。文レベル・トリガーは、トリガーを起動する文が発行されたときにオプションのトリガー制約が満たされていると、1 回のみ起動されます。

INSTEAD OF トリガー文は、各行について暗黙的にアクティブになります。

制限事項：この句は、(DDL およびデータベース・イベント・トリガーではなく) DML イベント・トリガーにのみ有効です。

WHEN 句

トリガー制約 (Oracle がトリガーを起動するために必要な SQL 条件) を指定します。条件の構文は、[第 5 章「条件」](#)を参照してください。この条件には関連名を指定する必要があります。問合せは指定できません。

制限事項：

- DML イベント・トリガーに対してこの句を指定する場合、FOR EACH ROW も指定する必要があります。トリガーを起動する文の対象となる行ごとに、この条件が評価されます。
- INSTEAD OF トリガー文にトリガー制約は指定できません。
- オブジェクト列、オブジェクト列の属性、VARRAY、ネストした表、LOB 列を参照できます。トリガー制約内では PL/SQL ファンクションおよびメソッドは起動できません。

pl/sql_block

Oracle がトリガーを起動するために実行する PL/SQL ブロックを指定します。

データベース・トリガーの PL/SQL ブロックは、システム・イベント属性の抽出用にのみ設計された SYS スキーマ内の一連の組込みファンクションの 1 つを指定できます。これらのファンクションは、データベース・トリガーの PL/SQL ブロック内でのみ使用できます。

制限事項：

- トリガーの PL/SQL ブロックは、ブロックが同一のトランザクションで実行される場合、トランザクション制御 SQL 文（COMMIT、ROLLBACK、SAVEPOINT および SET CONSTRAINT）を指定できません。
- PL/SQL ブロック内部のトリガー・アクションでは LOB 列を参照および使用できますが、トリガー・アクション内部で LOB 列の値を変更することはできません。

参照：

- PL/SQL ブロックの書込み方法など、PL/SQL の詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。
- これらのファンクションの詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

call_procedure_statement

call_procedure_statement 句によって、トリガー・コード・インラインを PL/SQL ブロックとして指定するかわりに、ストアド・プロシージャをコールできるようになります。この文の構文は、次の例外を除いて、11-63 ページの「CALL」の文と同じです。

- ファンクションにのみ適用されるため、CALL の INTO 句は指定できません。
- *expr* 内のバインド変数は指定できません。

- 定義されているトリガーの表の列を参照する場合、:NEW および :OLD を指定する必要があります。

参照： 14-89 ページの「トリガー本体のプロシージャのコール例」を参照してください。

例

DML トリガーの例 次の文は、スキーマ hr に emp_permit_changes という名前の BEFORE 文トリガーを作成します。そのトリガーを書き込み、この表に発行された DML 文に制限を配置します（その文が発行できる場合）。

```
CREATE TRIGGER hr.emp_permit_changes
  BEFORE
  DELETE OR INSERT OR UPDATE
  ON hr.employees
  < pl/sql block >
```

DELETE、INSERT または UPDATE の各文がスキーマ hr 内の employees 表に対して実行されると、このトリガーが起動されます。トリガー emp_permit_changes は BEFORE 文トリガーであるため、emp_permit_changes を起動する文が実行される前に 1 回起動されます。

制限付き DML トリガーの例 次の文は、スキーマ hr 内に salary_check という名前の BEFORE 行レベル・トリガーを作成します。たとえば、PL/SQL ブロックによって、従業員の給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よう指定されたとします。

```
CREATE TRIGGER hr.salary_check
  BEFORE INSERT OR UPDATE OF salary, job_id ON hr.employees
  FOR EACH ROW
  WHEN (new.job_id <> 'AD_VP')
  < pl/sql_block >
```

このトリガーは、次の文のいずれかが発行されるたびに起動されます。

- employees 表に対して行を追加する INSERT 文
- employees 表の salary 列値または job_id 列値を変更する UPDATE 文

salary_check は BEFORE 行レベル・トリガーであるため、UPDATE 文で更新される各行を変更または INSERT 文で挿入される各行を追加する前に、このトリガーが起動されます。

salary_check には、管理部門の副社長（AD_VP）の給与を確認できないトリガー制約が設定されています。

トリガー本体のプロシージャのコール例 PL/SQL ブロック内のトリガー本体を設定するかわりに、コール側プロシージャによって前述の例で記述された salary_check トリガーを作成することができます。従業員の給与が適切な範囲内にあることを確認する hr.salary_check プロシージャを定義したと仮定します。次のようにトリガー salary_check を作成できます。

```
CREATE TRIGGER hr.salary_check
  BEFORE INSERT OR UPDATE OF salary, job_id ON hr.employees
  FOR EACH ROW
  WHEN (new.job_id <> 'AD_VP')
  CALL check_sal(:new.job_id, :new.salary, :new.last_name);
```

check_sal プロシージャは、PL/SQL、C または Java で実装されます。また、:NEW 値のかわりに CALL 句の :OLD 値を指定できます。

データベース・イベント・トリガーの例 次の文は、すべてのエラーをログするトリガーを作成します。PL/SQL ブロックが特定のエラー（無効なログイン：エラー番号は 1017）に対して特別な処理を行うと仮定します。このトリガーは、AFTER 文トリガーで、文の実行に失敗（ログインの失敗など）した後で起動されます。

```
CREATE TRIGGER log_errors AFTER SERVERERROR ON DATABASE
  BEGIN
    IF (IS_SERVERERROR (1017)) THEN
      <special processing of logon error>
    ELSE
      <log error number>
    END IF;
  END;
```

DDL トリガーの例 次の文は、任意の DDL 文 CREATE の AFTER 文トリガーを作成します。このトリガーを使用して、自分のスキーマにある新規データ・ディクショナリ・オブジェクトの作成を監査します。

```
CREATE TRIGGER audit_db_object AFTER CREATE
  ON SCHEMA
  < pl/sql_block >
```

INSTEAD OF トリガーの例 この例では、顧客情報および注文情報を表示する `oe.order_info` ビューを作成します。

```
CREATE VIEW order_info AS
  SELECT c.customer_id, c.cust_last_name, c.cust_first_name,
         o.order_id, o.order_date, o.order_status
  FROM customers c, orders o
  WHERE c.customer_id = o.customer_id;
```

`orders` 表の主キー (`order_id`) が結合ビューの結果セットで一意でないため、通常は、このビューは更新できません。このビューを更新可能にするには、ビューに **INSTEAD OF** トリガーを作成し、ビューに対する **INSERT** 文を処理させます。

```
CREATE OR REPLACE TRIGGER order_info_insert
  INSTEAD OF INSERT ON order_info
  DECLARE
    duplicate_info EXCEPTION;
    PRAGMA EXCEPTION_INIT (duplicate_info, -00001);
  BEGIN
    INSERT INTO customers
      (customer_id, cust_last_name, cust_first_name)
    VALUES (
      :new.customer_id,
      :new.cust_last_name,
      :new.cust_first_name);
    INSERT INTO orders
      (order_id, order_date, customer_id)
    VALUES (
      :new.order_id,
      :new.order_date,
      :new.customer_id);
  EXCEPTION
    WHEN duplicate_info THEN
      RAISE_APPLICATION_ERROR (
        num=> -20107,
        msg=> 'Duplicate customer or order ID');
  END order_info_insert;
/
```

すべての **NOT NULL** 列に値があれば、ビューを介してどちらの実表にも追加できるようになります。

```
INSERT INTO order_info VALUES
  (999, 'Smith', 'John', 2500, '13-MAR-2001', 0);
```

SQL 文 : CREATE TYPE ~ DROP ROLLBACK SEGMENT

この章では、次の SQL 文について説明します。

- CREATE TYPE
- CREATE TYPE BODY
- CREATE USER
- CREATE VIEW
- DELETE
- DISASSOCIATE STATISTICS
- DROP CLUSTER
- DROP CONTEXT
- DROP DATABASE LINK
- DROP DIMENSION
- DROP DIRECTORY
- DROP FUNCTION
- DROP INDEX
- DROP INDEXTYPE
- DROP JAVA
- DROP LIBRARY
- DROP MATERIALIZED VIEW
- DROP MATERIALIZED VIEW LOG

-
- DROP OPERATOR
 - DROP OUTLINE
 - DROP PACKAGE
 - DROP PROCEDURE
 - DROP PROFILE
 - DROP ROLE
 - DROP ROLLBACK SEGMENT

CREATE TYPE

用途

CREATE TYPE 文を使用すると、**オブジェクト型**、**SQLJ オブジェクト型**（オブジェクト型の一種）、名前付きの可変配列（**VARRAY**）、ネストした表の型または**不完全なオブジェクト型**を作成できます。CREATE TYPE 文および CREATE TYPE BODY 文を使用してオブジェクト型を作成します。CREATE TYPE 文では、オブジェクト型の名前、オブジェクトの属性、メソッドおよびその他のプロパティを指定します。CREATE TYPE BODY 文には、その型でのメソッドに対するコードが含まれます。

注意：

- メソッドではなく、属性のみを宣言する型仕様を持つオブジェクト型を作成した場合は、オブジェクト型本体を指定する必要はありません。
 - SQLJ オブジェクト型を作成する場合は、型本体を指定できません。型の実装は Java クラスとして指定されます。
-

ユーザー定義型を作成した場合、Oracle は、暗黙的に、そのコンストラクタ・メソッドを定義します。**コンストラクタ**は、システム提供のプロシージャで、SQL 文または PL/SQL コード内で、その型の値のインスタンスを構成するために使用します。コンストラクタ・メソッドの名前は、ユーザー定義型の名前と同じです。

オブジェクト型のコンストラクタ・メソッドのパラメータは、オブジェクト型のデータ属性です。また、そのオブジェクト型用の属性定義と同じ順序で定義されます。ネストした表または VARRAY コンストラクタのパラメータは、ネストした表または VARRAY の要素です。

不完全型とは、フォワード型定義によって作成される型です。このオブジェクト型には名前がありますが、属性およびメソッドがないため、不完全といわれます。他の型からの参照が可能なため、互いに参照する型の定義に使用できます。ただし、不完全オブジェクト型を使用して表やオブジェクト列またはネストした表型の列を作成する場合は、型を完全に指定しておく必要があります。

参照：

- 型のメンバー・メソッドの作成については、15-24 ページの「[CREATE TYPE BODY](#)」を参照してください。
- オブジェクト、不完全型、VARRAY およびネストした表の詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』、『Oracle9i アプリケーション開発者ガイド - オブジェクト・リレーションアル機能』および『Oracle9i データベース概要』を参照してください。

前提条件

自分のスキーマ内に型を作成する場合は、CREATE TYPE システム権限が必要です。他のユーザーのスキーマ内に型を作成する場合は、CREATE ANY TYPE システム権限が必要です。これらの権限は、明示的に取得することもロールを介して取得することもできます。

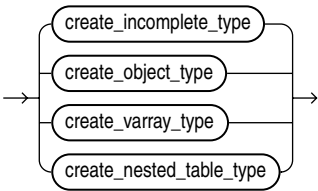
サブタイプを作成する場合は、UNDER ANY TYPE システム権限またはスーパータイプに対する UNDER オブジェクト権限が必要です。

型の所有者には、表定義内で参照する他のすべての型にアクセスするための EXECUTE オブジェクト権限が必要です。または、EXECUTE ANY TYPE システム権限が必要です。所有者は、これらの権限をロールを介して取得することはできません。

型の所有者が、型にアクセスする権限を他のユーザーに付与する場合、所有者には、参照型に対する GRANT OPTION 付きの EXECUTE オブジェクト権限、または ADMIN OPTION 付きの EXECUTE ANY TYPE システム権限が必要です。これらの権限がないと、型の所有者は、型にアクセスする権限を他のユーザーに付与できません。

構文

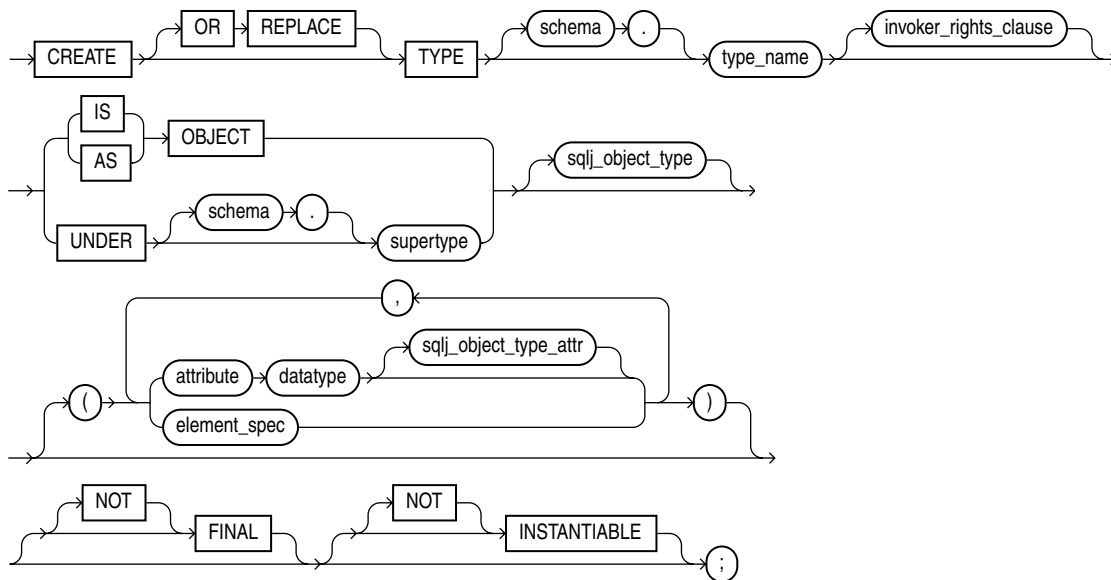
create_type::=



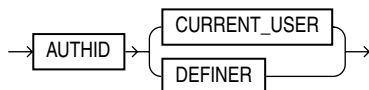
create_incomplete_type::=



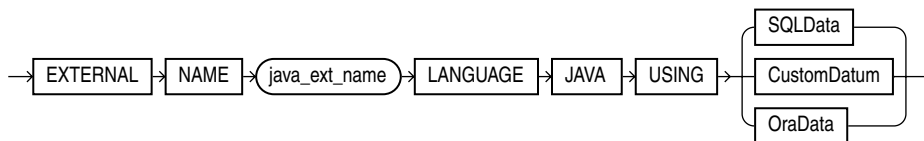
create_object_type::=



invoker_rights_clause::=



sqlj_object_type::=

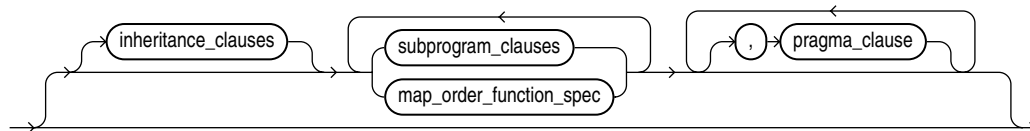


sqlj_object_type_attr::=

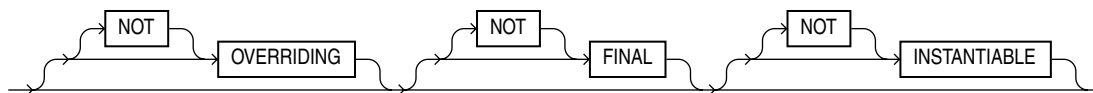


CREATE TYPE

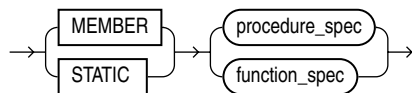
element_spec::=



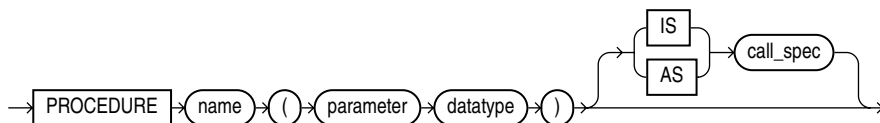
inheritance_clauses::=



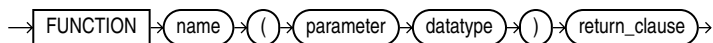
subprogram_clauses::=



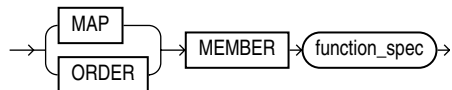
procedure_spec::=



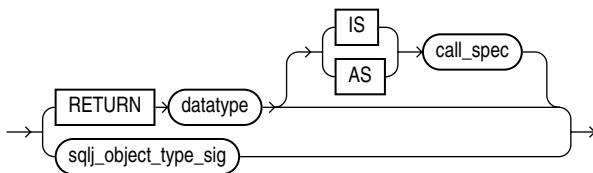
function_spec::=



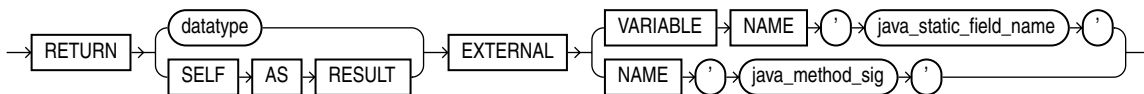
map_order_function_spec::=



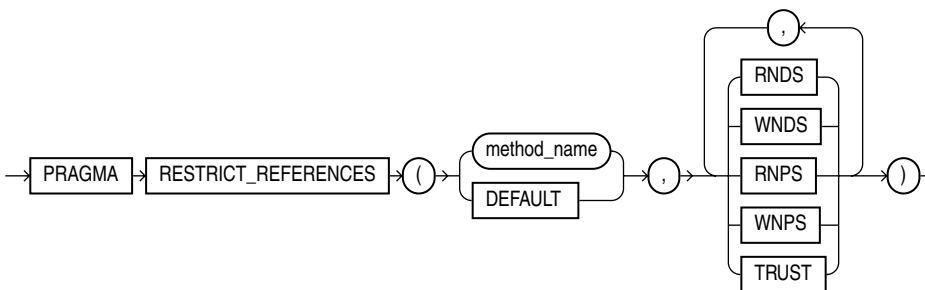
return_clause::=



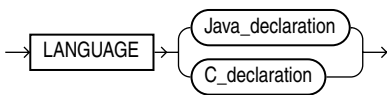
sqlj_object_type_sig::=



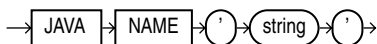
pragma_clause::=



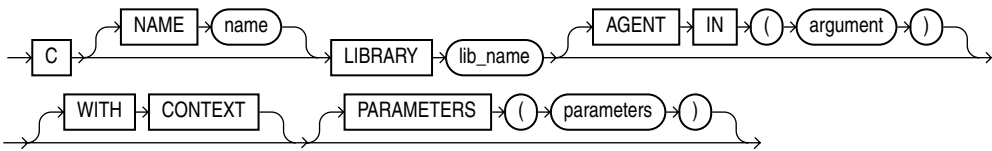
call_spec::=



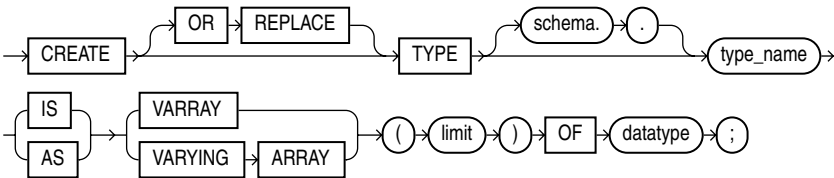
Java_declaration::=



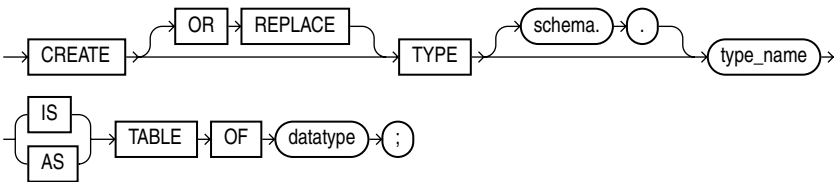
C_declaration::=



create_varray_type::=



create_nested_table_type::=



キーワードとパラメータ

OR REPLACE

既存の型を再作成する場合は、OR REPLACE を指定します。この句を指定した場合、既存のトリガーの定義をはじめに削除しなくても、その定義を変更できます。

再作成するオブジェクト型に対する権限があらかじめ付与されている場合は、再作成後にあらためて権限を付与されなくてもそのオブジェクト型を使用および参照できます。

ファンクション索引が型によって異なる場合、その索引には DISABLED のマークが付きます。

schema

作成する型が定義されるスキーマを指定します。*schema* を指定しない場合、現行のスキーマにその型が作成されます。

type_name

オブジェクト型、ネストした表型または VARRAY 型の名前を指定します。

型の作成時にコンパイル・エラーが発生した場合、エラーを戻します。SQL*Plus コマンド SHOW ERRORS を使用して、関連するコンパイラ・エラー・メッセージを表示できます。

create_object_type

create_object_type 句を使用すると、(不完全型ではなく) ユーザー定義型オブジェクト型を作成できます。データ構造を形成する変数を **属性** といいます。オブジェクトの動作を定義するメンバー・サブプログラムを **メソッド** といいます。キーワード AS OBJECT は、オブジェクト型の作成時に必要です。

invoker_rights_clause

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

また、この句は、問合せ、DML 操作、およびその型のメンバー・ファンクションおよびプロシージャ内の動的 SQL 文の外部名の変換方法も定義します。

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

また、この句は、問合せ、DML 操作、および動的 SQL 文の外部名を CURRENT_USER のスキーマで変換することも指定します。その他すべての文の外部名は、型が存在するスキーマ内で変換します。

- ファンクションおよびプロシージャが存在するスキーマの所有者権限で、オブジェクト型のメンバー・ファンクションおよびプロシージャを実行し、メンバー・ファンクションおよびプロシージャが存在するスキーマで外部名を変換する場合は、AUTHID DEFINER を指定します。これはデフォルトであり、**定義者権限型**を作成します。

***invoker_rights_clause* の制限事項**

- この句はオブジェクト型のみに指定でき、ネストした表および VARRAY 型には指定できません。
- サブタイプを作成する場合は、明確にするためにこの句を指定できます。ただし、サブタイプはスーパータイプの権限モデルを継承するため、スーパータイプに指定した値と異なる値を指定できません。
- スーパータイプが定義者権限で作成された場合、スーパータイプと同じスキーマでサブタイプを作成する必要があります。

参照：

- CURRENT_USER の判断方法については、『Oracle9i データベース概要』および『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。
- 『PL/SQL 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 OBJECT 句

AS OBJECT を指定すると、トップレベル（ルート）のオブジェクト型を作成できます。

UNDER 句

UNDER *supertype* を指定すると、既存の型のサブタイプを作成できます。既存のスーパータイプは、オブジェクト型である必要があります。この文で作成するサブタイプは、そのスーパータイプのプロパティを継承します。また、プロパティの一部がオーバーライドしたり、スーパータイプから識別する新しいプロパティを追加します。

sqlj_object_type

この句を指定すると、**SQLJ オブジェクト型**を作成することができます。SQLJ オブジェクト型の場合、Java クラスを SQL のユーザー定義型へマップします。他のユーザー定義型と同様に、SQLJ オブジェクト型に表または列を定義できます。

1 つの Java クラスを複数の SQLJ オブジェクト型にマップすることもできます。SQLJ オブジェクト型のサブタイプまたはスーパータイプが存在する場合、それも SQLJ オブジェクト型である必要があります。つまり、階層のすべての型は SQLJ オブジェクト型である必要があります。

java_ext_name Java クラスの名前を指定します。クラスが存在する場合、パブリックである必要があります。スキーマを含む Java の外部名は検証されます。

複数の SQLJ オブジェクト型を同一のクラスにマップすることもできます。ただし、次のことに注意してください。

- スーパータイプをマップするクラス直属のサブクラスに、サブタイプをマップする必要があります。
- スーパータイプが共通の 2 つのサブタイプを、同一のクラスにマップすることはできません。

SQLData | CustomDatum | OraData 型の Java インスタンスを作成するメカニズムを選択します。SQLData、CustomDatum および OraData は、使用するメカニズムのインタフェースです。

参照： これらのインタフェースの詳細は、『Oracle9i JDBC 開発者ガイド およびリファレンス』を参照してください。

element_spec

element_spec を指定すると、オブジェクト型の各属性を指定できます。

attribute

attribute には、オブジェクト属性の名前を指定します。属性とは、名前と型指定子を持つオブジェクト構造を形成するデータ項目です。各オブジェクト型には、1 つ以上の属性を指定する必要があります。

サブタイプを作成する場合、連鎖するスーパータイプで宣言されている属性名またはメソッド名と同じ属性名を指定することはできません。

datatype

datatype には、属性の Oracle 組み込みデータ型またはユーザー定義型を指定します。

データ型の制限事項

- ROWID 型、LONG 型または LONG ROW 型の属性は指定できません。
- NCLOB 型、NCHAR 型または NVARCHAR2 型の属性のオブジェクトを作成することはできませんが、メソッドにこれらデータ型のパラメータを指定することはできます。
- ユーザー定義オブジェクト型に対して UROWID のデータ型は指定できません。
- REF 型のオブジェクトを指定する場合、ターゲット・オブジェクトにはオブジェクト識別子が必要です。

参照： 指定可能なデータ型については、2-2 ページの「[データ型](#)」を参照してください。

sqlj_object_type_attr

この句は、*sqlj_object_type* 句を指定する場合（Java クラスを SQLJ オブジェクト型にマップする場合）のみに有効です。SQLJ オブジェクト型の属性に対応する Java フィールドの外部名を指定します。Java の *field_name* は、クラスに存在している必要があります。Java の *field_name* を、同じ型の階層にある 1 つ以上の SQLJ オブジェクト型の属性にマップすることはできません。

この句は、SQLJ オブジェクト型を作成する場合のオプションです。

subprogram_clauses

subprogram_clauses を指定すると、プロシージャ・サブプログラムをオブジェクト型に対応付けることができます。

MEMBER 句

属性として参照されるオブジェクト型に関連付けられたファンクションまたはプロシージャ・サブプログラムを指定します。通常、*object_expression.method()* のように、「自己参照的」スタイルでメンバー・メソッドを起動します。このクラスのメソッドには、メソッド本体で *SELF* として参照される暗黙的な最初の引数があります。この引数は、メソッドが起動されるオブジェクトを表します。

制限事項：Java クラスを SQLJ オブジェクト型へマップする場合は、メンバー・メソッドを指定できません。

STATIC 句

オブジェクト型に関連付けられたファンクションまたはプロシージャ・サブプログラムを指定します。メンバー・メソッドとは異なり、スタティク・メソッドには暗黙的なパラメータがありません（本体で *SELF* を参照できません）。通常、*type_name.method()* として起動されます。

制限事項：Java クラスのメンバー・メソッドを SQLJ オブジェクト型のスタティク・メソッドにマップできません。

メンバー・メソッドとスタティク・メソッドの両方に、各プロシージャまたはファンクションの仕様部について、オブジェクト型本体に対応するメソッド本体を指定する必要があります。

[NOT] FINAL, [NOT] INSTANTIABLE

これらの句を構文の最上位で指定すると、型の継承する属性が指定されます。

[NOT] FINAL 句を使用すると、この型のサブタイプを以降で作成させるかどうかを指定できます。

- この型のサブタイプを以降で作成させない場合は、FINAL を指定します。これはデフォルトです。
- この型のサブタイプを以降で作成させる場合は、NOT FINAL を指定します。

[NOT] INSTANTIABLE 句を使用すると、この型のオブジェクト・インスタンスを構成させるかどうかを指定します。

- この型のオブジェクト・インスタンスを構成させる場合は、INSTANTIABLE を指定します。これはデフォルトです。
- このオブジェクト型のコンストラクタ（デフォルトまたはユーザー定義の）が存在しない場合は、NOT INSTANTIABLE を指定します。インスタンス化ができないメソッドを持つ型、および属性を持たない型には、これらのキーワードを指定する必要があります（継承、またはこの句で指定）。

inheritance_clauses

element_spec の一部として *inheritance_clauses* を使用すると、スーパータイプとサブタイプの間の関係を指定できます。

OVERRIDING OVERRIDING を指定すると、スーパータイプで定義されているメソッドをこのメソッドでオーバーライドします。このキーワードは、スーパータイプのメソッドを再定義する場合に必須です。NOT OVERRIDING がデフォルトです。

制限事項：OVERRIDING 句は、SQLJ オブジェクト型には無効です。

FINAL FINAL を指定し、メソッドがこの型のサブタイプによってオーバーライドされないようにできます。デフォルトは、NOT FINAL です。

NOT INSTANTIABLE 型がこのメソッドの実装を提供しない場合は、NOT INSTANTIABLE を指定します。デフォルトでは、すべてのメソッドは INSTANTIABLE です。

制限事項：NOT INSTANTIABLE を指定する場合は、FINAL または STATIC を指定できません。

参照： 15-16 ページの「[map_order_function_spec](#)」を参照してください。

procedure_spec* または *function_spec

この句を使用すると、プロシージャまたはファンクションのパラメータおよびデータ型を指定できます。このサブプログラムは、プロシージャまたはファンクションの宣言を含みませんが、対応する CREATE TYPE BODY 文は発行する必要があります。

制限事項： サブタイプを作成する場合は、継承しているかどうかにかかわらず、プロシージャまたはファンクションに、連鎖するスーパータイプで宣言されている属性の名前と同じ名前を指定できません。

return_clause *return_clause* の最初の形式はファンクションのみに有効です。構文は省略形です。

参照：

- メソッド起動およびメソッドの詳細は、『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。
- 使用可能なすべての句を含む完全な構文については、13-58 ページの「CREATE PROCEDURE」および 12-47 ページの「CREATE FUNCTION」を参照してください。
- 15-24 ページの「CREATE TYPE BODY」を参照してください。
- ユーザー定義ファンクションの制限事項については、12-50 ページの「ユーザー定義ファンクションの制限事項」を参照してください。

sqlj_object_type_sig SQLJ オブジェクト型のファンクションまたはプロシージャを作成する場合は、*return_clause* のこの形式を使用します。

- Java クラスを SQLJ オブジェクト型にマップし、EXTERNAL NAME を指定する場合は、Java メソッドが戻す値は SQL が戻す値と互換性があり、また Java メソッドがパブリックである必要があります。また、メソッド・シグネチャ（メソッド名とパラメータ・タイプ）は、型の階層で一意である必要があります。
- EXTERNAL VARIABLE NAME を指定する場合、Java のスタティック・フィールドの型は戻り型と互換性がある必要があります。

call_spec

JAVA メソッドまたは C 関数名、パラメータ型および戻り型を SQL にマップするコール仕様を指定します。型のメンバー・メソッドすべてがこの句で定義された場合、対応する CREATE TYPE BODY 文を発行する必要はありません。

Java_declaration では、'string' が JAVA 実装メソッドを定義します。

参照：

- 『Oracle9i Java Stored Procedures Developer’s Guide』を参照してください。
- パラメータおよび *C_declaration* のセマンティクスについては、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

pragma_clause

pragma_clause によって、コンパイラ・ディレクティブを指定できます。PRAGMA RESTRICT_REFERENCES コンパイラ・ディレクティブは、データベースの表またはパッケージ変数（あるいはその両方）に対するメンバー・ファンクションの読み書きアクセスを拒否し、副作用の発生を防止します。

注意： アプリケーションの下位互換を保つ必要がない場合は、この句を使用しないことをお勧めします。この句は使用されません。Oracle9i からは、純粋度チェックが実行時に行われるため、この句は使用されなくなります。

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

参照： 『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

map_order_function_spec

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。また、サブタイプには MAP および ORDER メソッドのどちらも定義できません。ただし、スーパータイプが非最終の MAP メソッドを定義する場合、サブタイプは MAP メソッドをオーバーライドできます。(サブタイプは ORDER をオーバーライドできません。) 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。

Java クラスから SQL 型へのマップ時に、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。ただし、MAP メソッドおよび ORDER メソッドは、Java クラスのメンバー・ファンクションにマップする必要があります。

MAP メソッドも ORDER メソッドも定義されない場合、比較できるのは等価性または非等価性のみです。したがって、オブジェクト・インスタンスに順序を付けることはできません。同じ型定義のインスタンスは、それぞれの対応する属性の各組が等しい場合にのみ等しくなります。2 つのオブジェクト型の等価性を判断するために比較方法を指定する必要はありません。

オブジェクト・インスタンスで大規模のソートまたはハッシュ結合処理を実行する場合に、MAP を使用してください。MAP を使用してオブジェクトをスカラー値にマップすると、そのスカラーは、ソート中およびマージ中に使用されます。MAP メソッドは、各オブジェクトを比較するメソッドを起動する必要がある ORDER メソッドより効率的です。ハッシュ結合には MAP を使用する必要があります。ハッシュ・メカニズムはオブジェクト値でハッシュするため、ORDER メソッドを使用することはできません。

参照： オブジェクト値の比較の詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

MAP MEMBER この句は、オブジェクトの順序付けられたすべてのインスタンスの中から、指定したインスタンスの相対的な位置を戻すメンバー・ファンクション (MAP メソッド) を指定します。MAP メソッドは暗黙的にコールされ、オブジェクト・インスタンスを事前定義済みのスカラー型の値にマップすることによって、それらのオブジェクト・インスタンスに順序を設定します。PL/SQL は、この順序を使用してブール式の計算と比較を行います。

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

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

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

サブタイプには、新しい MAP メソッドを定義できません。ただし、継承された MAP メソッドはオーバーライドできます。

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

ORDER メソッドのどちらかの引数が NULL の場合は、ORDER メソッドは NULL を返し、メソッドは起動されません。

同じオブジェクト型定義の各インスタンスを ORDER BY 句の中で比較した場合、ORDER メソッド `map_order_function_spec` がコールされます。

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

サブタイプは ORDER メソッドを定義およびオーバーライドできません。

create_varray_type

`create_varray_type` は、それぞれが同じデータ型を持つ要素の順序付け集合としての型を作成します。名前および 0（ゼロ）以上の最大限度を指定する必要があります。配列限度は、整数リテラルである必要があります。Oracle では、匿名の VARRAY はサポートされません。

VARRAY 内に含まれるオブジェクトの型名は、次のいずれかである必要があります。

- 組込みデータ型
- REF
- オブジェクト型

制限事項：

- LOB データ型の VARRAY 型は作成できません。
- PL/SQL またはビュー問合せで使用する、XMLType の VARRAY 型を作成できます。ただし、Oracle が XMLType データを CLOB として格納するため、この VARRAY 型の列は作成できません（前述の制限事項を参照）。

create_nested_table_type

create_nested_table_type によって、データ型で名前付きのネストした表を作成します。

- *datatype* がオブジェクト型である場合、ネストした表型は、オブジェクト型の名前および属性と一致する列を持つ表を記述します。
- *datatype* がスカラー型である場合、ネストした表の型は、「column_value」という1つのスカラー型列を持つ表を記述します。

制限事項：*datatype* に NCLOB は指定できません。ただし、CLOB または BLOB は指定できます。

例

オブジェクト型の例 次の例は、customer_typ オブジェクト型をサンプルの注文入力スキーマ oe 用に作成します。

```
CREATE TYPE customer_typ AS OBJECT
( customer_id      NUMBER(6)
  , cust_first_name VARCHAR2(20)
  , cust_last_name  VARCHAR2(20)
  , cust_address    CUST_ADDRESS_TYP
  , phone_numbers   PHONE_LIST_TYP
  , nls_language    VARCHAR2(3)
  , nls_territory    VARCHAR2(30)
  , credit_limit     NUMBER(9,2)
  , cust_email       VARCHAR2(30)
  , cust_orders      ORDER_LIST_TYP
) ;
```

次の例では、CREATE TYPE BODY 文で実装されるメンバー・ファンクション prod を持つ data_typ オブジェクト型を作成します。

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

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

サブタイプの例 次の文は、サンプル・スキーマ oe にサブタイプ `corporate_customer_typ` を作成します。これは前述の例で作成したスーパータイプ `customer_typ` に基づき、`account_mgr_id` 属性を追加しています。

```
CREATE TYPE corporate_customer_typ UNDER customer_typ
    ( account_mgr_id      NUMBER(6)
    );
```

SQLJ オブジェクト型の例 次の例は、SQLJ オブジェクト型およびサブタイプを作成します。`address_t` 型は、Java クラス `Examples.Address` にマップします。サブタイプ `long_address_t` は、Java クラス `Examples.LongAddress` にマップします。この例は、これらの型に対する Java インスタンスの作成に使用するメカニズムとして `SQLData` を指定します。これらの型の各ファンクションの仕様は、Java クラスの実装に対応しています。

参照： これらの型のファンクションの仕様を Java で実装する場合の詳細は、『Oracle9i アプリケーション開発者ガイド - オブジェクト・リレーショナル機能』を参照してください。

```
CREATE TYPE address_t AS OBJECT
    EXTERNAL NAME 'Examples.Address' LANGUAGE JAVA
    USING SQLData(
        street_attr varchar(250) EXTERNAL NAME 'street',
        city_attr varchar(50) EXTERNAL NAME 'city',
        state varchar(50) EXTERNAL NAME 'state',
        zip_code_attr number EXTERNAL NAME 'zipCode',
        STATIC FUNCTION recom_width RETURN NUMBER
            EXTERNAL VARIABLE NAME 'recommendedWidth',
        STATIC FUNCTION create_address RETURN address_t
            EXTERNAL NAME 'create() return Examples.Address',
        STATIC FUNCTION construct RETURN address_t
            EXTERNAL NAME 'create() return Examples.Address',
        STATIC FUNCTION create_address (street VARCHAR, city VARCHAR,
            state VARCHAR, zip NUMBER) RETURN address_t
            EXTERNAL NAME 'create (java.lang.String, java.lang.String, java.lang.String,
int) return Examples.Address',
        STATIC FUNCTION construct (street VARCHAR, city VARCHAR,
            state VARCHAR, zip NUMBER) RETURN address_t
            EXTERNAL NAME
                'create (java.lang.String, java.lang.String, java.lang.String, int) return
Examples.Address',
        MEMBER FUNCTION to_string RETURN VARCHAR
            EXTERNAL NAME 'tojava.lang.String() return java.lang.String',
        MEMBER FUNCTION strip RETURN SELF AS RESULT
            EXTERNAL NAME 'removeLeadingBlanks () return Examples.Address'
    ) NOT FINAL;
```

```
CREATE OR REPLACE TYPE long_address_t
UNDER address_t
EXTERNAL NAME 'Examples.LongAddress' LANGUAGE JAVA
USING SQLData(
    street2_attr VARCHAR(250) EXTERNAL NAME 'street2',
    country_attr VARCHAR (200) EXTERNAL NAME 'country',
    address_code_attr VARCHAR (50) EXTERNAL NAME 'addrCode',
    STATIC FUNCTION create_address RETURN long_address_t
        EXTERNAL NAME 'create() return Examples.LongAddress',
    STATIC FUNCTION construct (street VARCHAR, city VARCHAR,
        state VARCHAR, country VARCHAR, addr_cd VARCHAR)
        RETURN long_address_t
        EXTERNAL NAME
            'create(java.lang.String, java.lang.String,
                java.lang.String, java.lang.String, java.lang.String)
                return Examples.LongAddress',
    STATIC FUNCTION construct RETURN long_address_t
        EXTERNAL NAME 'Examples.LongAddress()'
        return Examples.LongAddress',
    STATIC FUNCTION create_longaddress (
        street VARCHAR, city VARCHAR, state VARCHAR, country VARCHAR,
        addr_cd VARCHAR) return long_address_t
        EXTERNAL NAME
            'Examples.LongAddress (java.lang.String, java.lang.String,
                java.lang.String, java.lang.String, java.lang.String)
                return Examples.LongAddress',
    MEMBER FUNCTION get_country RETURN VARCHAR
        EXTERNAL NAME 'country_with_code () return java.lang.String'
);
```

型の階層例 次の文は、型の階層を作成します。employee_t 型は、person_t 型から name および ssn 属性を継承し、dept_id および salary 属性を追加しています。part_time_emp_t 型は、employee_t および employee_t を介した person_t からすべての属性を継承し、num_hrs 属性を追加しています。part_time_emp_t はデフォルトで最終型であるため、そのサブタイプを作成できません。

```
CREATE TYPE person_t AS OBJECT (name VARCHAR2(100), ssn NUMBER)
    NOT FINAL;

CREATE TYPE employee_t UNDER person_t
    (dept_id NUMBER, salary NUMBER) NOT FINAL;

CREATE TYPE part_time_emp_t UNDER employee_t (num_hrs NUMBER);
```


この型の階層を使用して、置換可能な表および置換可能な列を含む表を作成できます。詳細は、14-52 ページの「[置換可能な表および列のサンプル](#)」を参照してください。

VARRAY 型の例 次の文は、サンプル・スキーマ oe に、5 つの要素を含む VARRAY 型 phone_list_typ を作成します。

```
CREATE TYPE phone_list_typ AS VARRAY(5) OF VARCHAR2(25);
```

名前付き表型の例 次の例は、サンプル・スキーマ pm でオブジェクト型 textdoc_typ の名前付き表型 textdoc_tab を作成します。

```
CREATE TYPE textdoc_typ AS OBJECT
  ( document_typ      VARCHAR2(32)
    , formatted_doc    BLOB
  ) ;

CREATE TYPE textdoc_tab AS TABLE OF textdoc_typ;
```

VARRAY を含むネストした表型の例 次のマルチレベル・コレクションの例は、サンプル表 oe.customers の一例です。この例では、cust_address オブジェクト列は、VARRAY 列 phone_list_typ を埋め込んだネストした表の列になります。

```
CREATE TYPE phone_list_typ AS VARRAY(5) OF VARCHAR2(25);

CREATE TYPE cust_address_typ2 AS OBJECT
  ( street_address    VARCHAR2(40)
    , postal_code      VARCHAR2(10)
    , city             VARCHAR2(30)
    , state_province   VARCHAR2(10)
    , country_id       CHAR(2)
    , phone            phone_list_typ
  ) ;

CREATE TYPE cust_nt_address_typ
  AS TABLE OF cust_address_typ;
```

コンストラクタの例 次の文は、システム定義のコンストラクタを起動して、demo_typ オブジェクトを構成し、構成したオブジェクトを demo_tab 表に入れます。

```
CREATE TYPE demo_typ AS OBJECT (a1 NUMBER, a2 NUMBER);
```

```
CREATE TABLE demo_tab (b1 NUMBER, b2 demo_typ);
```

```
INSERT INTO demo_tab VALUES (1, demo_typ(2,3));
```

参照： コンストラクタの詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』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。

メンバー・メソッドの例 次の文は、メソッド・コンストラクタ col.getbar() を起動します。(例は getbar メソッドがすでに存在するものとします。)

```
CREATE TYPE demo_typ AS OBJECT (a1 NUMBER,  
    MEMBER FUNCTION getbar RETURN NUMBER);
```

```
CREATE TABLE demo_tab(col demo_typ);
```

```
SELECT col.getbar() FROM demo_tab;
```

ファンクションとは異なり、メソッドを起動する場合は、メソッドが他に引数を取らない場合でもカッコが必要です。

スタティック・メソッドの例 次の文は、employee_t 型の定義を変更し、変更した定義を construct_emp ファンクションと関連付けます。例は、最初にオブジェクト型 department_t を作成し、次に department_t 型の属性を含むオブジェクト型 employee_t を作成します。

```
CREATE OR REPLACE TYPE department_t AS OBJECT (  
    deptno number(10),  
    dname CHAR(30));
```

```
CREATE OR REPLACE TYPE employee_t AS OBJECT(  
    empid RAW(16),  
    ename CHAR(31),  
    dept REF department_t,  
    STATIC function construct_emp  
        (name VARCHAR2, dept REF department_t)  
        RETURN employee_t  
);
```

この文は、次の型本体の文が必要です。

```
CREATE OR REPLACE TYPE BODY employee_t IS
    STATIC FUNCTION construct_emp
        (name varchar2, dept REF department_t)
    RETURN employee_t IS
    BEGIN
        return employee_t(SYS_GUID(),name,dept);
    END;
END;
```

次にオブジェクト表を作成し、表に挿入します。

```
CREATE TABLE emptab OF employee_t;

INSERT INTO emptab
VALUES (employee_t.construct_emp('John Smith', NULL));
```

CREATE TYPE BODY

用途

CREATE TYPE BODY を使用すると、オブジェクト型仕様部で定義されたメンバー・メソッドを定義または実装することができます。CREATE TYPE 文および CREATE TYPE BODY 文を使用してオブジェクト型を作成します。CREATE TYPE 文では、オブジェクト型の名前、オブジェクトの属性、メソッドおよびその他のプロパティを指定します。CREATE TYPE BODY 文には、その型でのメソッドに対するコードが含まれます。

`call_spec` を定義していないオブジェクト型仕様部に指定された各メソッドには、オブジェクト型本体の対応するメソッド本体を指定する必要があります。

注意： SQLJ オブジェクト型を作成する場合は、型本体を指定できません。型の実装は Java クラスとして指定されます。

参照： 型仕様部の作成および変更については、15-3 ページの「[CREATE TYPE](#)」および 11-6 ページの「[ALTER TYPE](#)」を参照してください。

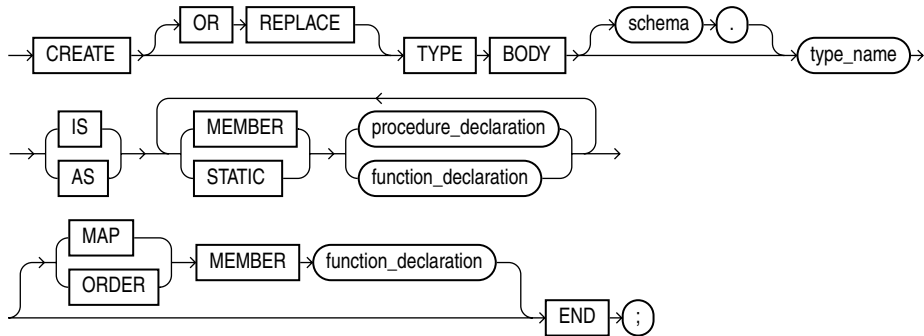
前提条件

オブジェクト型用の CREATE TYPE 仕様部で行われるすべてのメンバー宣言には、それに対応する構造が CREATE TYPE 文または CREATE TYPE BODY 文内に存在する必要があります。

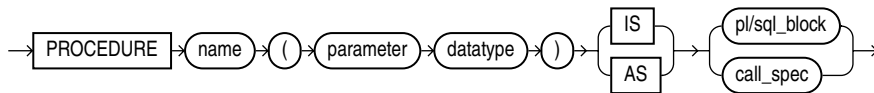
自分のスキーマ内で型本体を作成または再作成する場合は、CREATE TYPE システム権限または CREATE ANY TYPE システム権限が必要です。他のユーザーのスキーマ内でオブジェクト型を作成する場合は、CREATE ANY TYPE システム権限が必要です。他のユーザーのスキーマ内でオブジェクト型を置換する場合は、DROP ANY TYPE システム権限が必要です。

構文

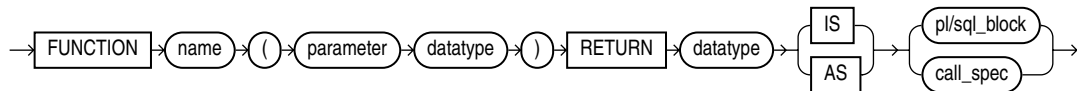
create_type_body::=



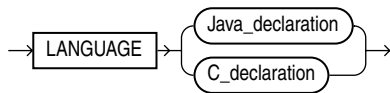
procedure_declaration::=



function_declaration::=



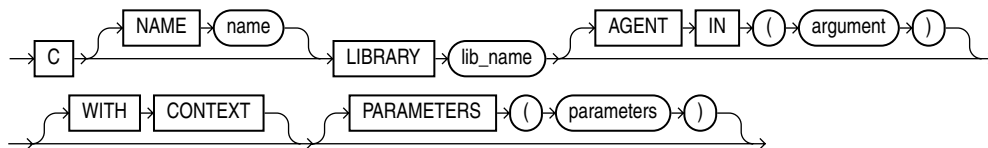
call_spec::=



Java_declaration::=



C_declaration::=



キーワードとパラメータ

OR REPLACE

既存の型本体を再作成する場合は、**OR REPLACE** を指定します。この句を指定した場合、既存の型本体の定義をはじめに削除しなくても、その定義を変更できます。

再作成されたオブジェクト型本体に対する権限を付与されているユーザーは、権限を再付与されなくても、そのオブジェクト型本体を使用および参照できます。

この句を使用した場合、**ALTER TYPE ... REPLACE** 文によって追加された仕様部に、新規メンバー・サブプログラム定義を追加できます。

schema

作成する型本体が定義されるスキーマを指定します。*schema* を省略した場合、Oracle では現行の自分のスキーマ内に型本体が作成されます。

type_name

オブジェクト型の名前を指定します。

IS | AS

MEMBER | STATIC

オブジェクト型仕様部に関連付けられたメソッド・ファンクションまたはプロシージャ・サブプログラムの型を指定します。

各プロシージャまたはファンクション宣言には、対応するメソッド名およびオプション・パラメータ・リスト、また、ファンクションの場合は、オブジェクト型仕様部の戻り型を定義する必要があります。

procedure_declaration* および *function_declaration プロシージャまたはファンクション・サブプログラムを宣言します。

参照：

- ユーザー定義ファンクションの制限事項については、15-3 ページの「[CREATE TYPE](#)」を参照してください。
- パッケージ内のサブプログラム名のオーバーロードについては、『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。
- 13-58 ページの「[CREATE PROCEDURE](#)」、12-47 ページの「[CREATE FUNCTION](#)」および『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

pl/sql_block プロシージャまたはファンクションを宣言します。

参照：『PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。

call_spec JAVA メソッドまたは C 関数名、パラメータ型および戻り型を SQL にマップするコール仕様を指定します。

Java_declaration では、'string' が JAVA 実装メソッドを定義します。

参照：

- 『Oracle9i Java Stored Procedures Developer's Guide』を参照してください。
- パラメータおよび *C_declaration* のセマンティクスについては、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

MAP | ORDER メソッド

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

どちらのメソッドも宣言しない場合、比較できるのはオブジェクト・インスタンスの等価性と非等価のみです。同じ型定義のインスタンスは、それぞれの対応する属性の各組が等しい場合にのみ等しくなります。

MAP MEMBER 句

MAP MEMBER を指定すると、メンバー・ファンクション（MAP メソッド）が定義または実装されます。メンバー・ファンクションは、オブジェクトのすべてのインスタンスの順序付けにおいて、指定したインスタンスの相対的な位置を戻します。MAP メソッドは暗黙的にコールされ、オブジェクト・インスタンスを事前定義済のスカラー型の値にマップすることによって、それらのオブジェクト・インスタンスに順序を指定します。PL/SQL は、この順序を使用してブール式の計算と比較を行います。

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

オブジェクト型本体には、1 つの MAP メソッドのみを指定できます。この MAP メソッドは、ファンクションである必要があります。この MAP ファンクションに指定できる引数は、暗黙的な SELF 引数のみです。

ORDER MEMBER 句

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

ORDER メソッドのどちらかの引数が NULL の場合は、ORDER メソッドは NULL を戻し、メソッドは起動されません。

同じオブジェクト型定義の各インスタンスを ORDER BY 句の中で比較した場合、ORDER MEMBER *function_declaration* がコールされます。

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

function_declaration ファンクション・サブプログラムを宣言します。

参照： 使用可能なすべての句を含む完全な構文については、13-58 ページの「[CREATE PROCEDURE](#)」および 12-47 ページの「[CREATE FUNCTION](#)」を参照してください。

AS EXTERNAL AS EXTERNAL は、C 関数を宣言するもう 1 つの方法です。この句は以前のリリースのもので、下位互換用にのみサポートされています。call_spec 構文は、C_declaration とともに使用することをお勧めします。

例

型本体の作成例は 15-18 ページの「[CREATE TYPE](#)」の「[例](#)」にあります。

型本体の更新例 次の例は、型に属性が追加された場合の `data_typ` オブジェクト型 (15-18 ページの「[オブジェクト型の例](#)」を参照) の型本体の更新方法を示します。

```
ALTER TYPE data_typ
  ADD MEMBER FUNCTION qtr(der_qtr DATE)
  RETURN CHAR CASCADE;

CREATE OR REPLACE TYPE BODY data_typ IS
  MEMBER FUNCTION prod (invent NUMBER) RETURN NUMBER IS
  BEGIN
    RETURN (year + invent);
  END;
  MEMBER FUNCTION qtr(der_qtr DATE) RETURN CHAR IS
  BEGIN
    IF (der_qtr < TO_DATE('01-APR', 'DD-MON')) THEN
      RETURN 'FIRST';
    ELSIF (der_qtr < TO_DATE('01-JUL', 'DD-MON')) THEN
      RETURN 'SECOND';
    ELSIF (der_qtr < TO_DATE('01-OCT', 'DD-MON')) THEN
      RETURN 'THIRD';
    ELSE
      RETURN 'FOURTH';
    END IF;
  END;
END;
/
```

CREATE USER

用途

CREATE USER 文を使用すると、データベース・**ユーザー**（データベースにログイン可能なアカウント）を作成でき、Oracle がそのユーザーによるアクセスを許可する方法が確立されます。

注意： プロキシ（アプリケーションまたはアプリケーション・サーバー）によって、Oracle とユーザーを接続することができます。構文および説明は、11-20 ページの「[ALTER USER](#)」を参照してください。

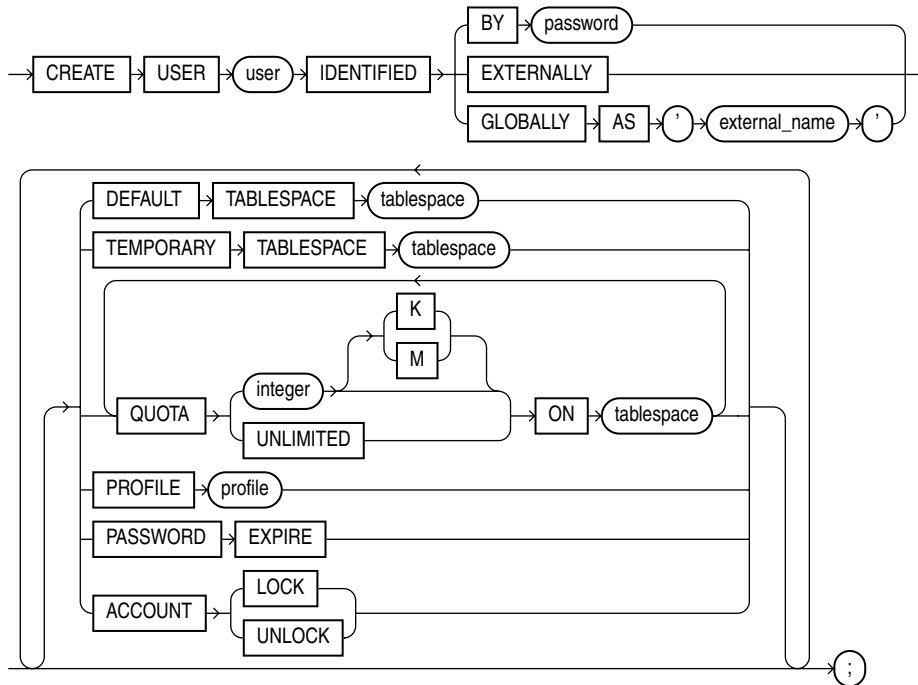
前提条件

CREATE USER システム権限が必要です。CREATE USER 文を使用して作成したユーザーの権限ドメインは空（権限を付与されていない状態）になります。Oracle にログインするユーザーには、CREATE SESSION システム権限が必要です。そのため、ユーザーを作成した場合、必ず CREATE SESSION 権限をそのユーザーに付与してください。

参照： 16-31 ページの「[GRANT](#)」を参照してください。

構文

create_user::=



キーワードとパラメータ

user

作成するユーザー名を指定します。この名前には、使用しているデータベース・キャラクタ・セットの文字のみを指定できます。また、2-106 ページの「[スキーマ・オブジェクトのネーミング規則](#)」で説明した規則に従う必要があります。データベース・キャラクタ・セットがマルチバイト文字を含んでいても、ユーザー名にはシングルバイト文字を1つ以上使用することをお勧めします。

IDENTIFIED 句

IDENTIFIED 句によって、ユーザー認証をどのように行うか示します。

BY password

BY *password* を指定すると、**ローカル・ユーザー**を作成し、ログインするときに、パスワードを指定する必要があることを示します。データベース・キャラクタ・セットにマルチバイト文字が含まれている場合でも、パスワードを指定できるのは、データベース・キャラクタ・セットのシングルバイト文字のみです。

Oracle の複雑なパスワード検証ルーチンを使用していない場合、パスワードは、2-106 ページの「**スキーマ・オブジェクトのネーミング規則**」にある規則に従う必要があります。そのルーチンでは、通常のネーミング規則より複雑な文字の組合せが必要です。このルーチンを UTLPWDMG.SQL スクリプトで実装します。

参照： パスワード管理およびパスワード保護の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。

EXTERNALLY 句

EXTERNALLY を指定すると、**外部ユーザー**を作成できます。このユーザーは、外部サービス（オペレーティング・システムまたはサード・パーティのサービス）で認証する必要があります。この場合、オペレーティング・システムのログイン認証によって、特定のオペレーティング・システム・ユーザーから特定のデータベース・ユーザーへのアクセスが可能になります。

注意： ご使用のオペレーティング・システムにログインする際のセキュリティが十分でない場合は、IDENTIFIED EXTERNALLY を使用しないことをお勧めします。詳細は、『Oracle9i データベース管理者ガイド』を参照してください。

GLOBALLY 句

GLOBALLY 句を使用すると、**グローバル・ユーザー**を作成することができます。このユーザーは、エンタープライズ・ディレクトリ・サービスによって認証される必要があります。'external_name' 文字列は、次のいずれかの形式になります。

- このユーザーを識別するエンタープライズ・ディレクトリ・サービスの X.509 の名前。文字列は、'CN=username,other_attributes' という形式である必要があります。other_attributes は、ディレクトリ内のユーザーの識別名（DN）以外の部分です。
- NULL 文字列（'）は、エンタープライズ・ディレクトリ・サービスが、認証されたグローバル・ユーザーを、適切なデータベース・スキーマに適切なロール付きでマップすることを示します。

注意： 特定のユーザーとして接続し、ALTER USER 文を使用してユーザーのロールをアクティブにするために、アプリケーション・サーバーの機能を制御できます。

参照：

- グローバル・ユーザーの詳細は、『Oracle Advanced Security 管理者ガイド』を参照してください。
- 11-20 ページの「ALTER USER」を参照してください。
- 詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』およびご使用のオペレーティング・システム固有のドキュメントを参照してください。

DEFAULT TABLESPACE 句

ユーザーが作成するオブジェクトを格納するデフォルトの表領域を指定します。この句を省略した場合、オブジェクトはデフォルトで SYSTEM 表領域に格納されます。

制限事項： UNDO 表領域はデフォルトの表領域に指定できません。

参照： 表領域の概要および UNDO 表領域の詳細は、14-62 ページの「CREATE TABLESPACE」を参照してください。

TEMPORARY TABLESPACE 句

ユーザーの一時セグメントが確保される表領域を指定します。この句を省略した場合、一時セグメントはデフォルトの SYSTEM 表領域に確保されます。

制限事項：

- 標準以外のブロック・サイズを使用した表領域をユーザーの一時表領域に指定することはできません。指定した場合も Oracle からエラーは戻されません。ただし、ユーザーが行う後続の操作で一時セグメントが必要な場合は、エラーになります。
- ローカル管理の表領域を指定する場合、一時表領域である必要があります。
- 一時表領域は、UNDO 表領域または自動セグメント領域管理の表領域にできません。

参照： UNDO 表領域およびセグメント管理の詳細は、14-62 ページの「CREATE TABLESPACE」を参照してください。

QUOTA 句

QUOTA 句を使用すると、表領域の最大 *integer* バイトの領域をユーザーに対して割り当てることができます。K または M を使用して、KB または MB 単位で指定することもできます。この割当て制限は、ユーザーが割当て可能な表領域の最大領域です。

CREATE USER 文では、複数の表領域に対して複数の QUOTA 句を指定できます。

UNLIMITED を使用すると、表領域への領域を無制限に割り当てることができます。

PROFILE 句

プロファイルを指定すると、ユーザーへの割当てを解除できます。このプロファイルによって、ユーザーが使用できるデータベース・リソース容量が制限されます。この句を省略した場合、DEFAULT プロファイルがユーザーに割り当てられます。

参照： 16-31 ページの「[GRANT](#)」および 13-65 ページの「[CREATE PROFILE](#)」を参照してください。

PASSWORD EXPIRE 句

ユーザーのパスワードを期限切れにする場合は、PASSWORD EXPIRE を指定します。この設定によって、ユーザーがデータベースにログインする前に、ユーザー（または DBA）にパスワードを変更させます。

ACCOUNT 句

ACCOUNT LOCK を指定すると、ユーザー・アカウントをロックし、アクセスを禁止にします。ACCOUNT UNLOCK を指定すると、ユーザー・アカウントのロックを解除し、アクセスを可能にします。

例

ユーザーの作成例 PASSWORD EXPIRE 指定して新規ユーザーを作成する場合、そのデータベースにログインする前に、そのユーザーのパスワードを変更する必要があります。次の文を発行することによって、sidney ユーザーを作成できます。

```
CREATE USER sidney
  IDENTIFIED BY welcome
  DEFAULT TABLESPACE demo
  QUOTA 10M ON demo
  TEMPORARY TABLESPACE temp
  QUOTA 5M ON system
  PROFILE app_user
  PASSWORD EXPIRE;
```

前述のユーザー `sidney` には次の特性があります。

- パスワード `welcome`
- 10MB の割当て制限のあるデフォルト表領域 `demo`
- 一時表領域 `temp`
- 5MB の割当て制限のある表領域 `SYSTEM` へのアクセス
- プロファイル `app_user` によって定義されているデータベース・リソースの制限
- `sidney` がデータベースにログインする前に変更する必要がある期限切れのパスワード

外部ユーザーの例 次の例は、データベースにアクセスする前に外部ソースによって識別される必要のある、外部ユーザーを作成します。

```
CREATE USER app_user1
  IDENTIFIED EXTERNALLY
  DEFAULT TABLESPACE tbs_1
  QUOTA 5M ON tbs_1
  PROFILE app_user;
```

ユーザー `app_user1` には次の追加の特性があります。

- デフォルト表領域 `tbs_1`
- デフォルト一時表領域 `tbs_1`
- 表領域 `tbs_1` に 5MB の領域、およびデータベースの一時表領域に無制限の割当て
- プロファイル `app_user` によって定義されているデータベース・リソースの制限

オペレーティング・システム・アカウント `app_user2` によってのみアクセス可能な別のユーザーを作成する場合、`app_user2` に初期化パラメータ `OS_AUTHENT_PREFIX` 値の接頭辞を付けます。たとえば、この値が「`ops$`」の場合、次の文でユーザー `ops$app_user2` を作成できます。

```
CREATE USER ops$app_user2
  IDENTIFIED EXTERNALLY
  DEFAULT TABLESPACE tbs_1
  QUOTA 5M ON tbs_1
  PROFILE app_user;
```

グローバル・ユーザーの例 次の例は、グローバル・ユーザーを作成します。グローバル・ユーザーを作成する場合、エンタープライズ・ディレクトリ・サーバーでユーザーを識別する X.509 の名前を指定することができます。

```
CREATE USER global_user
  IDENTIFIED GLOBALLY AS 'CN=analyst, OU=division1, O=oracle, C=US'
  DEFAULT TABLESPACE tbs_1
  QUOTA 5M on tbs_1;
```

CREATE VIEW

用途

CREATE VIEW 文を使用すると、**ビュー**を定義できます。ビューは論理表で、1 つ以上の表またはビューがベースとなります。ビューにデータそのものが格納されているわけではありません。ビューの基礎になる表を**実表**といいます。

LOB およびオブジェクト・データ型（オブジェクト型、REF、ネストした表または VARRAY 型）をサポートする**オブジェクト・ビュー**またはリレーショナル・ビューを既存のビュー・メカニズムで作成します。オブジェクト・ビューとは、ユーザー定義型のビューのことで、ビューの各行に、それぞれが一意のオブジェクト識別子を持つオブジェクトが含まれます。

参照：

- ビューの様々な型およびその使用方法については、『Oracle9i データベース概要』、『Oracle9i アプリケーション開発者ガイド - 基礎編』および『Oracle9i データベース管理者ガイド』を参照してください。
- ビューの変更については、11-28 ページの「[ALTER VIEW](#)」を参照してください。
- データベースからのビューの削除については、16-21 ページの「[DROP VIEW](#)」を参照してください。

前提条件

自分のスキーマ内にビューを作成する場合は、CREATE VIEW システム権限が必要です。他のユーザーのスキーマ内にビューを作成する場合は、CREATE ANY VIEW システム権限が必要です。

サブビューを作成する場合は、UNDER ANY TYPE システム権限またはスーパービューに対する UNDER オブジェクト権限が必要です。

ビューを含むスキーマの所有者は、そのビューの基礎となっているすべての表またはビューに対する行の選択、挿入、更新または削除の権限が必要です。また、所有者には、これらの権限がロールを介して付与されるのではなく、直接付与されている必要があります。

オブジェクト・ビューの作成時にオブジェクト型の基本コンストラクタ・メソッドを使用する場合、次のいずれかが真である必要があります。

- オブジェクト型が作成対象のビューと同じスキーマに属している。
- EXECUTE ANY TYPE システム権限がある。
- そのオブジェクト型に対する EXECUTE オブジェクト権限を持っている。

参照： 作成するビューの基礎となる表またはビューで、ビューの所有者に必要な権限の詳細は、17-4 ページの「[SELECT](#)」、16-56 ページの「[INSERT](#)」、17-64 ページの「[UPDATE](#)」および 15-49 ページの「[DELETE](#)」を参照してください。

パーティション・ビュー

パーティション・ビューは、パーティション化機能を必要とするアプリケーション向けに、リリース 7.3 で導入されたものです。Oracle9i では、パーティション・ビューがサポートされているため、何も変更しなくてもリリース 7.3 からのアプリケーションのアップグレードができます。通常、Oracle9i 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。

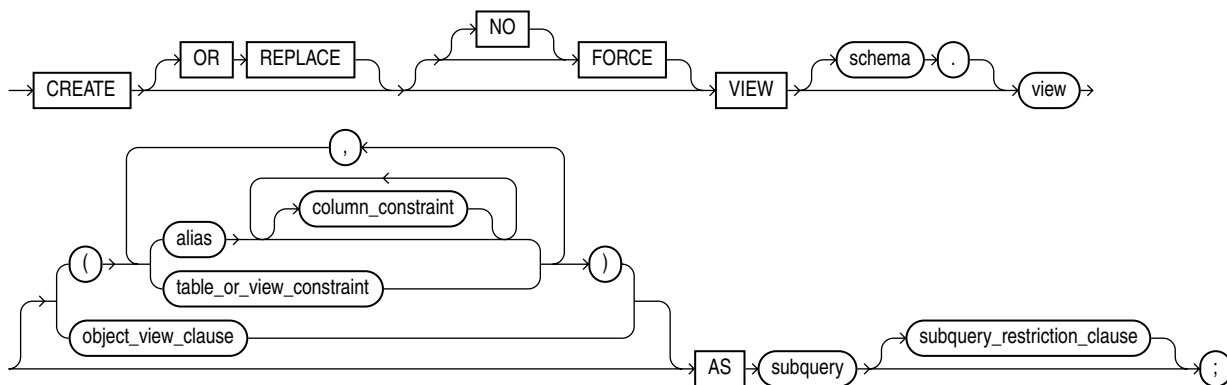
Oracle9i では、CREATE TABLE 文を使用して、パーティション表を簡単に作成できます。パーティション表には、パーティション・ビューと同じメリットがあるのみでなく、パーティション・ビューのデメリットも補います。通常の動作環境では、パーティション・ビューではなく、パーティション表を使用することをお勧めします。

参照：

- パーティション・ビューのデメリットについては、『Oracle9i データベース概要』を参照してください。
- パーティション・ビューのパーティションへの移行については、『Oracle9i データベース管理者ガイド』を参照してください。
- パーティション表の詳細は、14-6 ページの「[CREATE TABLE](#)」を参照してください。

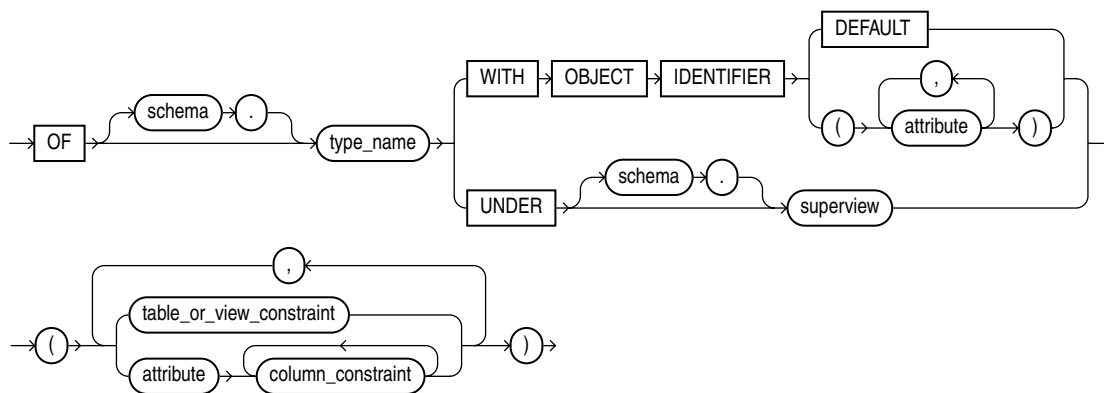
構文

create_view::=



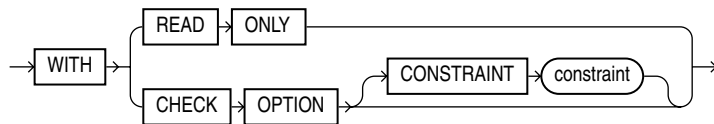
table_or_view_constraint および **column_constraint**: 11-72 ページの「[constraint_clause](#)」を参照してください。

object_view_clause::=



subquery: 17-4 ページの「[SELECT](#)」を参照してください。

subquery_restriction_clause::=



キーワードとパラメータ

OR REPLACE

既存のビューを再作成する場合は、OR REPLACE を指定します。この句を使用した場合、以前に付与されたオブジェクト権限を削除、再作成、再付与しなくても、既存のビュー定義を変更できます。

ビューを再作成した場合、ビュー内で定義された INSTEAD OF トリガーが削除されます。

すべてのマテリアライズド・ビューが view に依存している場合、そのマテリアライズド・ビューには UNUSABLE というマークが付けられ、それらが再利用可能になるようにリストアップする場合は、完全なリフレッシュが必要になります。無効なマテリアライズド・ビューはクエリー・リライトで使用されることはなく、また、再コンパイルされるまでリフレッシュされません。

参照：

- 無効なマテリアライズド・ビュー・ログのリフレッシュについては、8-74 ページの「[ALTER MATERIALIZED VIEW](#)」を参照してください。
- マテリアライズド・ビューの概要は、『Oracle9i データベース概要』を参照してください。
- INSTEAD OF 句については、14-77 ページの「[CREATE TRIGGER](#)」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、またはそのビューを含んでいるスキーマの所有者が、それらの表やオブジェクトに対する権限を持っているかにかかわらず、強制的にビューを作成する場合は、FORCE を指定します。SELECT、INSERT、UPDATE または DELETE 文をビューに対して発行する場合、前述の条件が真である必要があります。

ビュー定義が制約を含む場合、実表または参照するオブジェクトが存在しないときに CREATE VIEW ... FORCE は正常に実行されません。また、ビュー定義が存在しない制約を参照する場合も、CREATE VIEW ... FORCE は正常に実行されません。

NO FORCE

実表が存在し、ビューを含むスキーマの所有者がその権限を持っている場合のみビューを作成する場合は、NOFORCE を指定します。これはデフォルトです。

schema

ビューが定義されるスキーマを指定します。schema を省略した場合、自分のスキーマにビューが作成されます。

view

ビューまたはオブジェクト・ビューの名前を指定します。

制限事項：あるビューに INSTEAD OF トリガーがある場合、そのビューで作成されるビューが更新可能であっても、そのビューには INSTEAD OF トリガーがある必要があります。

alias

ビューの問合せによって選択された式に対して名前を指定します。別名数は、ビューによって選択された式の数と一致している必要があります。別名は、次の Oracle スキーマ・オブジェクトのネーミングの規則に従う必要があります。別名は、ビュー内で一意である必要があります。

別名を省略した場合、Oracle はビューの問合せの列または列の別名から別名を導出します。このため、ビューの問合せが列の名前のみでなく式を含んでいる場合は、別名を使用する必要があります。また、ビュー定義が制約を含む場合は、別名を指定する必要があります。

制限事項：オブジェクト・ビュー作成時に別名は指定できません。

参照： 2-110 ページの「スキーマ・オブジェクトの構文および SQL 文の構成要素」を参照してください。

table_or_view_constraint および column_constraint

ビューおよびオブジェクト・ビューに制約を指定できません。

table_or_view_constraint 句を使用して、ビュー・レベルで制約を定義します。適切な別名の後で column_constraint 句を使用して、列レベルまたは属性レベルで制約を定義します。

ビュー制約（ビュー・レベル、列レベルまたは属性レベル）は宣言のみです。つまり、適用されません。ただし、ビューの操作は、基礎となる実表に定義した整合性制約に従属します。そのため、実表の制約を介してビューの制約を適用することができます。

ビュー制約は表制約のサブセットで、いくつかの制限事項があります。

参照： 制約の概要およびビュー制約の制限については、11-72 ページの「[constraint_clause](#)」を参照してください。

object_view_clause

object_view_clause を指定すると、オブジェクト型にビューを定義できます。

OF type_name 句

type_name 型の**オブジェクト・ビュー**を明示的に作成します。オブジェクト・ビューの列が、*type_name* 型の最上位属性に対応しています。各行にはオブジェクト・インスタンスが含まれ、また、各インスタンスは、WITH OBJECT IDENTIFIER 句で指定したオブジェクト識別子 (OID) に関連付けられます。*schema* を省略した場合、自分のスキーマ内にオブジェクト・ビューが作成されます。

WITH OBJECT IDENTIFIER 句

WITH OBJECT IDENTIFIER 句を使用すると、最上位（ルート）のオブジェクト・ビューを指定できます。オブジェクト・ビュー内の各行を識別するためのキーとして使用されるオブジェクト型の属性を指定します。ほとんどの場合、各属性は実表の主キー列と対応します。属性リストが一意で、ビューの 1 行ずつを識別することを確認する必要があります。

制限事項：

- オブジェクト・ビュー内の複数のインスタンスに変換される主キー REF を参照解除または確保しようとした場合、Oracle はエラーを戻します。
- サブビューはスーパービューのオブジェクト識別子を継承するため、サブビューを作成する場合は、この句を指定できません。

注意： Oracle8 リリース 8.0 の構文 WITH OBJECT OID は、この構文と置き換えられます。キーワード WITH OBJECT OID は下位互換用にサポートされていますが、新しい構文 WITH OBJECT IDENTIFIER を使用することをお勧めします。

オブジェクト・ビューがオブジェクト表またはオブジェクト・ビュー上で定義されている場合は、この句を省略するか、または DEFAULT を指定します。

DEFAULT 各行を一意に識別するために、基になるオブジェクト表またはオブジェクト・ビュー固有のオブジェクト識別子を使用する場合は、DEFAULT を指定します。

attribute attributeには、オブジェクト・ビューに対して作成されるオブジェクト識別子の基になるオブジェクト型の属性を指定します。

UNDER 句

UNDER 句を使用すると、オブジェクト・スーパービューに基づくサブビューを指定できます。

ビューがスーパービューまたはサブビューのどちらであるかを確認するには、USER_VIEWS、ALL_VIEWS または DBA_VIEWS データ・ディクショナリ・ビューの SUPERVIEW_NAME を問い合わせます。

制限事項：

- サブビューは、スーパービューと同じスキーマに作成する必要があります。
- オブジェクト型 type_name は、superview 直属のサブタイプである必要があります。
- 同一のスーパービューには、特定の型のサブビューを 1 つのみ作成できます。

参照：

- オブジェクト作成の詳細は、15-3 ページの「[CREATE TYPE](#)」を参照してください。
- データ・ディクショナリ・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

AS subquery

ビューの基になっている表（実表）の列と行を識別する副問合せを指定します。副問合せの SELECT 構文のリストには 1000 以内の式を指定できます。

リモート表およびビューを参照するビューを作成する場合、指定するデータベース・リンクを CREATE DATABASE LINK 文の CONNECT TO 句を使用して作成する必要があります。また、ビュー副問合せのスキーマ名で指定する必要があります。

ビュー副問合せの制限事項

- ビュー副問合せは、CURRVAL および NEXTVAL 疑似列を選択できません。
- ビュー副問合せが ROWID、ROWNUM または LEVEL の各疑似列を選択する場合、そのビュー副問合せには列の別名が必要です。
- ビュー副問合せがアスタリスク (*) を使用して表のすべての列を選択し、後でその表に新しい列を追加する場合、CREATE OR REPLACE VIEW 文を発行してビューを再作成するまで、そのビューにそれらの列は含まれません。

- オブジェクト・ビューの場合、ビュー副問合せの **SELECT** 構文のリスト内の要素数は、そのオブジェクト型の最上位属性数と同じである必要があります。それぞれの選択要素のデータ型は、対応する最上位属性と同じである必要があります。
- **SAMPLE** 句は指定できません。

前述の制限は、マテリアライズド・ビューにも適用されます。

更新可能なビューの作成に関する注意事項

更新可能なビューとは、実表の行を挿入、更新または削除できるビューです。更新可能なビューを作成するか、またはビューに **INSTEAD OF** トリガーを作成して更新可能にすることができます。

更新可能なビューの列を更新する方法を確認するには、**USER_UPDATABLE_COLUMNS** データ・ディクショナリ・ビューを問い合わせます。（このビューで表示される情報は、更新可能なビューのみに意味があります。）

- ビューを固有の特性として更新可能にする場合、次の構造体を指定しないでください。
 - 集合演算子
 - **DISTINCT** 演算子
 - 集計グループ・ファンクション
 - **GROUP BY**、**ORDER BY**、**CONNECT BY** または **START WITH** 句
 - **SELECT** 構文のリストにあるコレクション式
 - **SELECT** 構文のリストにある副問合せ
 - 結合（次に示す一部の例外を除く）
- また、更新可能なビューが疑似列または式を含む場合は、これらの疑似列または式を参照する **UPDATE** 文を使用して、実表の行を更新できません。
- 結合ビューを変更する場合、次の条件が真である必要があります。
 - **DML** 文は、結合の基になる 1 つの表のみに影響する。
 - **INSERT** 文の場合、**WITH CHECK OPTION** 付きでビューを作成できない。また、値を挿入するすべての列は**キー保存表**の列である。実表の各主キーまたは一意キーの値に対するキー保存表は、結合ビューで一意である。
 - **UPDATE** 文の場合、更新されるすべての列がキー保存表から抽出される。ビューが **WITH CHECK OPTION** 付きで作成された場合、結合列、およびビュー内で 2 回以上参照される表の列は更新できない。
 - **DELETE** 文の場合、結合内のキー保存表は 1 つのみである。この表は、ビューを **WITH CHECK OPTION** 付きで作成しないと、結合内に 2 回以上参照される。

参照：

- 更新可能なビューの詳細は、『Oracle9i データベース管理者ガイド』を参照してください。
- オブジェクト型をサポートするオブジェクト・ビューまたはリレーショナル・ビューの更新の詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。
- 更新可能な結合ビューおよびキー保存表の詳細は、15-47 ページの「[結合ビューの例](#)」を参照してください。
- 更新できないビューに対する INSTEAD OF トリガーの例については、14-90 ページの「[INSTEAD OF トリガーの例](#)」を参照してください。

subquery_restriction_clause

subquery_restriction_clause を使用すると、次のいずれかの方法で副問合せを制限できます。

WITH READ ONLY ビューを介して削除、挿入または更新を行わない場合は、WITH READ ONLY を指定します。

WITH CHECK OPTION WITH CHECK OPTION を指定すると、ビューを介して実行される挿入および更新処理の結果が、ビュー副問合せによって選択できる行であることを保証します。次の場合、CHECK OPTION ではこの条件を保証できません。

- ビュー副問合せまたはビューの基礎になるビュー問合せに副問合せが含まれている場合。
- INSERT、UPDATE または DELETE 操作が INSTEAD OF トリガーを使用して実行された場合。

CONSTRAINT *constraint* CHECK OPTION 制約の名前を指定します。この識別子を省略した場合、その制約に SYS_Cn という形式の名前が自動的に割り当てられます。この場合の *n* は、その制約名をデータベース内で一意の名前にする整数です。

例

基本ビューの例 次の文は、サンプル表 `employees` の `emp_view` という名前のビューを作成します。このビューには、部門 20 の従業員とその年間給与が表示されます。

```
CREATE VIEW emp_view AS
  SELECT last_name, salary*12 annual_salary
  FROM employees
  WHERE department_id = 20;
```

この場合、副問合せで式 `sal*12` に列の別名 (`annual_salary`) を使用しているため、この式に基づく列の名前をビュー宣言で定義する必要はありません。

制約付きビューの例 次の文は、サンプル表 `hr.employees` の制限付きのビューを作成し、ビューの列 `email` に一意制約を定義し、ビューの列 `emp_id` にビューに対する主キー制約を定義します。

```
CREATE VIEW emp_sal (emp_id, last_name,
  email UNIQUE RELY DISABLE NOVALIDATE,
  CONSTRAINT id_pk PRIMARY KEY (emp_id) RELY DISABLE NOVALIDATE)
AS SELECT employee_id, last_name, email FROM employees;
```

更新可能なビューの例 次の文は、`employees` 表内の販売事務員および購買係全員の更新可能なビュー `clerk` を作成します。従業員の ID、姓、部門番号および職種はビューで参照でき、事務長の行のこれらの列のみを更新できます。

```
CREATE VIEW clerk AS
  SELECT employee_id, last_name, department_id, job_id
  FROM employees
  WHERE job_id = 'PU_CLERK'
     or job_id = 'SH_CLERK'
     or job_id = 'ST_CLERK';
```

このビューによって、購買係 `job_id` を購買マネージャ (`PU_MAN`) に変更します。

```
UPDATE clerk SET job_id = 'PU_MAN' WHERE employee_id = 118;
```

次の例は、同じビューを WITH CHECK OPTION 付きで作成します。新しい従業員が事務員ではない場合は、clerk に新しい行を挿入できません。従業員の job_id を他の事務タイプに更新できますが、事務員以外の job_id の場合はビューが employees にアクセスできないため、正常に更新できません。

```
CREATE VIEW clerk AS
  SELECT employee_id, last_name, department_id, job_id
  FROM employees
  WHERE job_id = 'PU_CLERK'
     or job_id = 'SH_CLERK'
     or job_id = 'ST_CLERK'
  WITH CHECK OPTION;
```

結合ビューの例 結合ビューは、結合を含むビューの副問合せです。副問合せの結合において、一意索引を持つ列が 1 列以上ある場合、結合ビューで 1 つの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、次の場合です。

```
CREATE VIEW locations_view AS
  SELECT d.department_id, d.department_name, l.location_id, l.city
  FROM departments d, locations l
  WHERE d.location_id = l.location_id;

SELECT column_name, updatable
  FROM user_updatable_columns
  WHERE table_name = 'LOCATIONS_VIEW';
```

COLUMN_NAME	UPD
DEPARTMENT_ID	YES
DEPARTMENT_NAME	YES
LOCATION_ID	NO
CITY	NO

前述の例では、locations 表の location_id 列に対する主キー索引は、locations_view ビュー内で一意ではありません。そのため、locations はキー保存表ではなく、その実表の列は更新できません。

```
INSERT INTO locations_view VALUES
  (999, 'Entertainment', 87, 'Roma');
INSERT INTO locations_view VALUES
```

★

1 行でエラーが発生しました。

ORA-01776: 結合ビューを介して複数の実表を変更できません。

departments 表にマップするビュー内のすべての列が更新可能とマークされていて、departments の主キーがビューに含まれているため、departments 実表に対して、行の挿入、更新または削除ができます。

```
INSERT INTO locations_view (department_id, department_name)
VALUES (999, 'Entertainment');
```

1 行が作成されました。

注意： NOT NULL 列に DEFAULT 値を指定していない場合は、結合内のすべての表のすべての NOT NULL 列が含まれないかぎり、ビューを使用して表に挿入することはできません。

参照： 結合ビューの更新の詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

読取り専用ビューの例 次の文は、oe.customers 表の customer という名前の読取り専用ビューを作成します。顧客の姓、言語、掛貸限度額のみがこのビューで参照できます。

```
CREATE VIEW customer (name, language, credit)
AS SELECT cust_last_name, nls_language, credit_limit
FROM customers
WITH READ ONLY;
```

オブジェクト・ビューの例 次の例では、oc スキーマ内での inventory_typ 型の作成方法、およびその型を基にした oc_inventories ビューの作成方法を示します。

```
CREATE TYPE inventory_typ AS OBJECT
( product_id          number(6)
, warehouse           warehouse_typ
, quantity_on_hand    number(8)
) ;

CREATE OR REPLACE VIEW oc_inventories OF inventory_typ
WITH OBJECT IDENTIFIER (product_id)
AS SELECT i.product_id,
        warehouse_typ(w.warehouse_id, w.warehouse_name, w.location_id),
        i.quantity_on_hand
FROM inventories i, warehouses w
WHERE i.warehouse_id=w.warehouse_id;
```

DELETE

用途

DELETE を使用すると、表、パーティション表、ビューの実表、またはビューの実表パーティションから行を削除できます。

前提条件

表から行を削除する場合、その表が自分のスキーマ内に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。

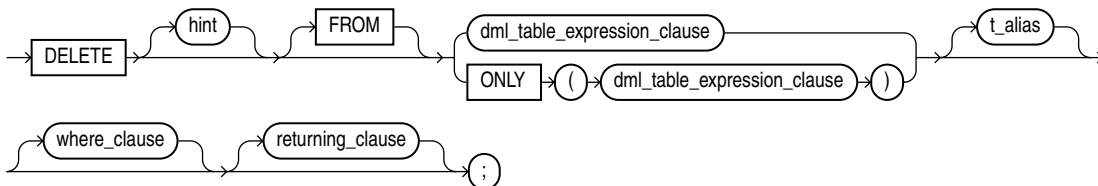
ビューの実表から行を削除する場合、そのビューを含むスキーマの所有者には、その実表の DELETE 権限が必要です。また、他のスキーマ内にビューが存在している場合は、そのビューの DELETE 権限が必要です。

DELETE ANY TABLE システム権限があれば、任意の表、表パーティションまたはビューの実表から行を削除できます。

初期化パラメータ SQL92_SECURITY が TRUE に設定されている場合、表の列 (*where_clause* の列など) を参照する DELETE を実行するには、その表の SELECT 権限が必要です。

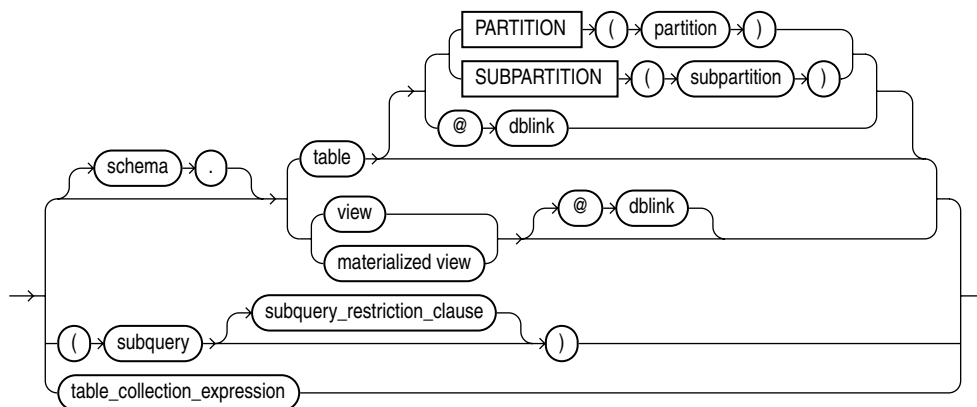
構文

delete::=



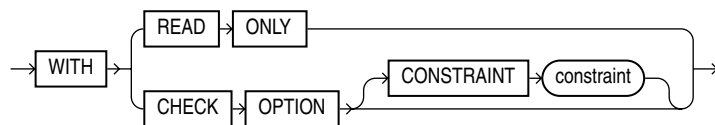
DELETE

dml_table_expression::=

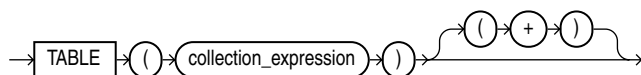


subquery: 17-4 ページの「[SELECT](#)」を参照してください。

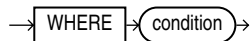
subquery_restriction_clause::=



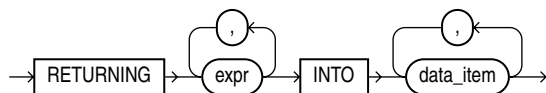
table_collection_expression::=



where_clause::=



returning_clause::=



キーワードとパラメータ

hint

文に対して実行計画を選択する際の、オプティマイザへ指示を渡すコメントを指定します。

参照： ヒントの構文および説明については、2-86 ページの「[ヒント](#)」および『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。

from_clause

FROM 句を使用すると、行を削除するデータベース・オブジェクトを指定できます。

ONLY 構文は、ビューのみに関連します。FROM 句のビューがビューの階層に属し、サブビューから行を削除しない場合は、ONLY 句を使用します。

dml_table_expression_clause

schema

表またはビューが含まれているスキーマを指定します。*schema* を指定しない場合、表またはビューが自分のスキーマにあるとみなされます。

table | view | materialized view | subquery

行が削除される表、ビュー、列または副問合せから生成される列の名前を指定します。*view* を指定した場合、ビューの実表から行が削除されます。

表（またはビューの実表）が 1 つ以上のドメイン索引列を含む場合、この文は、適切な索引タイプ削除ルーチンを実行します。

参照： これらのルーチンの詳細は、『Oracle9i Data Cartridge Developer's Guide』を参照してください。

表に対して DELETE 文を実行すると、その表に定義されている DELETE トリガーが起動されます。

行の削除によって解放される表や索引のすべての領域は、表および索引によって引き続き保持されます。

PARTITION (*partition_name*) および SUBPARTITION (*subpartition_name*)

更新対象の表内にあるパーティションまたはサブパーティションの名前を指定します。

パーティション表から値を削除する場合、そのパーティション名を指定する必要はありません。ただし、パーティション名を指定した方が、複雑な *where_clause* より効率が上がる場合があります。

dblink

表またはビューが存在しているリモート・データベースとのデータベース・リンクの完全名または部分名を指定します。Oracle の分散機能を使用している場合にのみ、リモート表またはリモート・ビューから行を削除できます。

参照： データベース・リンクの参照方法の詳細は、2-113 ページの「[リモート・データベース内のオブジェクトの参照](#)」を参照してください。

dblink を省略した場合、その表またはビューはローカル・データベースにあるとみなされます。

subquery_restriction_clause

subquery_restriction_clause を使用すると、次のいずれかの方法で副問合せを制限できます。

- WITH READ ONLY によって、副問合せを更新禁止にすることを指定します。
- WITH CHECK OPTION によって、副問合せに存在しない行を生成する表変更の禁止を指定します。

参照： 17-30 ページの「[WITH CHECK OPTION の例](#)」を参照してください。

table_collection_expression

問合せおよび DML 操作で、*collection_expression* 値を表として扱う場合に、*table_collection_expression* を指定します。*collection_expression* は、副問合せ、列、組込みファンクションまたはコレクション・コンストラクタにすることができます。その形式にかかわらず、集合値（ネストした表型または VARRAY 型の値）を戻す必要があります。このようなコレクションの要素抽出プロセスを **コレクション・ネスト解除** といいます。

注意： 以前のリリースの Oracle では、*collection_expression* が副問合せの場合、*table_collection_expression* を「`THE subquery`」と表現していました。現在、このような表現方法はされていません。

table_collection_expression を使用して他の表にもある行のみを削除します。

collection_expression ネストした表の列を *table* または *view* から選択する副問合せを指定します。

注意： 以前のリリースでは、*table_collection_expression* は「*THE subquery*」と表現されていました。現在、このような表現方法はされていません。

dml_table_expression_clauseに関する制限事項

- *table* (または *view* の実表) に、IN_PROGRESS または FAILED とマークされたドメイン索引がある場合は、この文は実行できません。
- 関係する索引パーティションが UNUSABLE とマークされている場合は、パーティションに挿入できません。
- *dml_table_expression_clause* の副問合せでは、ORDER BY 句を指定できません。
- ビューを定義する問合せに次のいずれかの構造体が含まれている場合は、INSTEAD OF トリガーを使用する場合を除いて、ビューからの削除はできません。
 - 集合演算子
 - DISTINCT 演算子
 - 集計グループ・ファンクション
 - GROUP BY、ORDER BY、CONNECT BY または START WITH 句
 - SELECT 構文のリストにあるコレクション式
 - SELECT 構文のリストにある副問合せ
 - 結合（一部の例外を除く）。詳細は、『Oracle9i データベース管理者ガイド』を参照してください。

UNUSABLE のマークが付いている索引、索引パーティションまたは索引サブパーティションを指定する場合、SKIP_UNUSABLE_INDEXES パラメータが true に設定されていないかぎり、DELETE 文は正常に実行されません。

参照： 9-2 ページの「ALTER SESSION」を参照してください。

table_collection_expression***where_clause***

where_clause を使用すると、条件を満たす行のみが削除されます。この条件は、表を参照したり、副問合せを含むことができます。Oracle の分散機能を使用している場合にのみ、リモート表またはリモート・ビューから行を削除できます。

参照： 条件の構文については、[第5章「条件」](#)を参照してください。

注意： この句がリモート・オブジェクトを参照する *subquery* を含む場合、参照がローカル・データベース上でオブジェクトにループバックしないかぎり、DELETE はパラレルで実行されます。ただし、*dml_table_expression_clause* の *subquery* がリモート・オブジェクトを参照する場合は、UPDATE はシリアルで実行されます。

参照： 14-43 ページの「CREATE TABLE」の「[parallel_clause](#)」を参照してください。

dblink を省略した場合、その表またはビューはローカル・データベースにあるとみなされます。

where_clause を省略した場合、表またはビューのすべての行が削除されます。

t_alias 文中で参照する表、ビュー、副問合せまたはコレクション値の**相関名**です。通常、別名は相関問合せを持つ DELETE 文で使用されます。

注意： *dml_table_expression_clause* がいずれかのオブジェクト型属性またはオブジェクト型メソッドを参照する場合、この別名が必要です。

returning_clause

DML (INSERT、UPDATE または DELETE) 文に影響される行を取り出します。この句は、表、マテリアライズド・ビュー、および単一の実表を持つビューに指定できます。

単一行で処理する場合、*returning_clause* 付きの DML 文で、処理された行および ROWID を使用している列式、処理された行への REF を取り出し、ホスト変数または PL/SQL 変数に格納します。

複数行で処理する場合、*returning_clause* 付きの DML 文で、式の値、ROWID および処理された行に関連する REF をバインド配列に格納します。

expr *expr* リストの各項目は、適切な構文で表す必要があります。スカラー副問合せ式を除くすべての形式が有効です。

INTO 変更された行の値を、*data_item* リストに指定する変数に格納することを示します。

data_item 取り出された *expr* 値を格納するホスト変数または PL/SQL 変数です。

RETURNING リストの各式については、INTO リストに、対応する型に互換がある PL/SQL 変数またはホスト変数を指定する必要があります。

制限事項：

- *returning_clause* は、マルチ表の挿入に指定できません。
- パラレル DML またはリモート・オブジェクトでは、この句を使用できません。
- この句で LONG 型を取り出すことはできません。
- INSTEAD OF トリガーが定義されたビューに対しては、この句を指定できません。

参照： BULK COLLECT 句を使用してコレクション変数に複数の値を戻す場合は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。

注意： この句を使用すると、削除された列から値を戻すことができるため、DELETE 文の後で SELECT を実行する必要がなくなります。

例

基本的な例 次の文は、サンプル表 `oe.product_descriptions` からすべての行を削除します。

```
DELETE FROM product_descriptions;
```

次の文は、サンプル表 `hr.employees` から手数料率が 10% 未満の購買係を削除します。

```
DELETE FROM employees
WHERE job_id = 'PU_CLERK'
AND commission_pct < .1;
```

次の文は前述の例と同じ効果がありますが、副問合せを使用します。

```
DELETE FROM (SELECT * FROM employees)
  WHERE job_id = 'PU_CLERK'
  AND commission_pct < .1;
```

リモート・データベースの例 次の文は、データベース・リンク `dallas` によってアクセス可能なデータベースの、ユーザー `blake` が所有する `accounts` 表からすべての行を削除します。

```
DELETE FROM blake.accounts@dallas;
```

ネストした表の例 次の文は、ネストした表 `projs` の行のうち、部門番号が `123` か `456` のいずれかであるもの、または部門の予算が `456.78` を超えるものを削除します。

```
DELETE TABLE(SELECT projs FROM dept d WHERE d.dno = 123) p
  WHERE p.pno IN (123, 456) OR p.budgets > 456.78;
```

パーティションの例 次の例は、`sh.sales` 表のパーティション `sales_q1_1998` から行を削除します。

```
DELETE FROM sales PARTITION (sales_q1_1998)
  WHERE amount_sold != 0;
```

RETURNING 句の例 次の例は、削除された行から `salary` 列を戻し、その結果をバインド配列 `:1` に格納します。(バインド配列は事前に宣言しておく必要があります。)

```
DELETE FROM employees
  WHERE job_id = 'PU_CLERK' AND commission_pct < 100
  RETURNING salary INTO :1;
```

DISASSOCIATE STATISTICS

用途

DISASSOCIATE STATISTICS 文を使用すると、統計タイプ（またはデフォルト統計）と、列、スタンドアロン・ファンクション、パッケージ、型、ドメイン索引または索引タイプとの関連付けを解除できます。

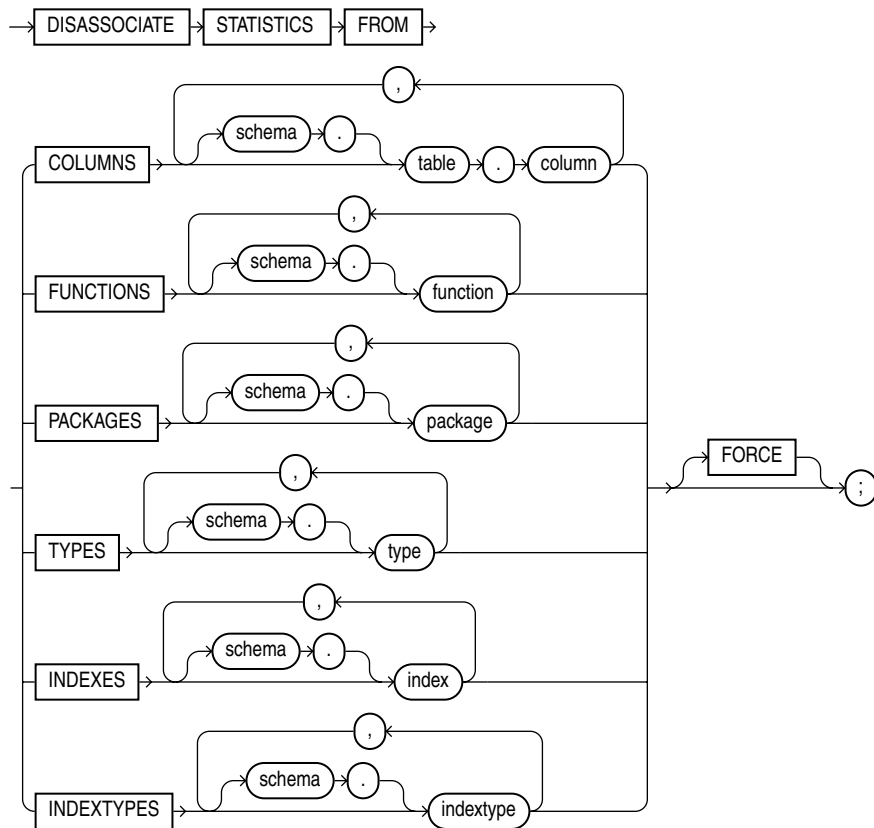
参照： 統計タイプの関連付けの詳細は、11-46 ページの「[ASSOCIATE STATISTICS](#)」を参照してください。

前提条件

この文を発行する場合は、ベース・オブジェクト（表、ファンクション、パッケージ、型、ドメイン索引または索引タイプ）を変更する適切な権限が必要です。

構文

disassociate_statistics::=



キーワードとパラメータ

FROM COLUMNS | FUNCTIONS | PACKAGES | TYPES | INDEXES | INDEXTYPES

統計との関連付けを解除する 1 つ以上の列、スタンドアロン・ファンクション、パッケージ、型、ドメイン索引または索引タイプを指定します。

schema を指定しない場合、オブジェクトは自分のスキーマ内にあるとみなされます。

オブジェクトのユーザー定義の統計を収集する場合、FORCE を指定しないかぎり文は実行されません。

FORCE

FORCE を指定すると、統計タイプを使用するオブジェクトでの統計の有無にかかわらず、関連付けを解除します。統計が存在する場合、関連付けが解除される前に統計は削除されます。

注意： 統計タイプが関連付けられているオブジェクトを削除する場合、FORCE オプションを持つ統計タイプの関連付けは自動的に解除され、統計タイプで収集したすべての統計は削除されます。

例

統計との関連付けの解除例 次の文は、oe スキーマの emp_mgmt パッケージと統計との関連付けを解除します。

```
DISASSOCIATE STATISTICS FROM PACKAGES oe.emp_mgmt;
```

DROP CLUSTER

用途

- DROP CLUSTER 文を使用すると、データベースからクラスタを削除できます。
- 個々の表は非クラスタ化できません。そのかわり、次の手順を実行します。
1. 既存の表と同じ構成および内容で、新しい表を CLUSTER 句なしで作成します。
 2. 既存の表を削除します。
 3. RENAME 文を使用して、新しい表に既存の表の名前を指定します。
 4. 既存のクラスタ化表に対して付与された権限は適用されないため、新しく作成した非クラスタ化表に権限を付与します。

参照： これらの手順の詳細は、14-6 ページの「[CREATE TABLE](#)」、16-6 ページの「[DROP TABLE](#)」、16-85 ページの「[RENAME](#)」および 16-31 ページの「[GRANT](#)」を参照してください。

前提条件

削除するクラスタが自分のスキーマ内にあるか、または DROP ANY CLUSTER システム権限が必要です。

構文

drop_cluster::=



キーワードとパラメータ

schema

クラスタが含まれているスキーマを指定します。*schema* を指定しない場合、そのクラスタが自分のスキーマにあるとみなされます。

cluster

削除するクラスタの名前を指定します。クラスタを削除するとクラスタの索引も削除され、その索引のデータ・ブロックを含むクラスタ領域が表領域に戻されます。

INCLUDING TABLES

INCLUDING TABLES を指定すると、クラスタに属するすべての表が削除されます。

CASCADE CONSTRAINTS

CASCADE CONSTRAINTS を指定すると、クラスタに含まれる表の主キーまたは一意キーを参照するクラスタ外の表にあるすべての参照整合性制約が削除されます。このような参照整合性制約がある場合にこの句の指定を省略した場合、エラー・メッセージが表示され、クラスタは削除されません。

例

次に、12-9 ページの「CREATE CLUSTER」の「例」で作成したクラスタを変更する例を示します。

次の文は、language クラスタを削除します。

```
DROP CLUSTER language;
```

次の文は、dept_10、dept_20 およびこれらの表の主キーまたは一意キーを参照する参照整合性制約と同様に personnel クラスタを削除します。

```
DROP CLUSTER personnel
INCLUDING TABLES
CASCADE CONSTRAINTS;
```

DROP CONTEXT

用途

DROP CONTEXT 文を使用すると、データベースからコンテキスト・ネームスペースを削除できます。

注意： コンテキスト・ネームスペースを削除した場合でも、ユーザー・セッションに設定されたネームスペースのコンテキストは無効になりません。ただし、ユーザーがそのコンテキストを設定しようとした場合、そのコンテキストは無効になります。

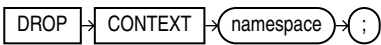
参照： コンテキストの詳細は、12-11 ページの「[CREATE CONTEXT](#)」および『Oracle9i データベース概要』を参照してください。

前提条件

DROP ANY CONTEXT システム権限が必要です。

構文

drop_context::=



キーワードとパラメータ

namespace

削除するコンテキスト・ネームスペースを指定します。組込みネームスペース USERENV は削除できません。

参照： USERENV の詳細は、6-145 ページの「[SYS_CONTEXT](#)」を参照してください。

例

DROP CONTEXT の例 次の文は、12-11 ページの「[CREATE CONTEXT](#)」で作成したコンテキストを削除します。

```
DROP CONTEXT hr_context;
```

DROP DATABASE LINK

用途

DROP DATABASE LINK 文を使用すると、データベースからデータベース・リンクを削除できます。

参照： データベース・リンクの作成方法については、12-33 ページの「[CREATE DATABASE LINK](#)」を参照してください。

前提条件

プライベート・データベース・リンクは自分のスキーマにある必要があります。パブリック・データベース・リンクを削除する場合は、DROP PUBLIC DATABASE LINK システム権限が必要です。

構文

drop_database_link::=



キーワードとパラメータ

PUBLIC

パブリック・データベース・リンクを削除する場合は、PUBLIC を指定する必要があります。

dblink

削除するデータベース・リンクの名前を指定します。

制限事項： 他のユーザーのスキーマ内のデータベース・リンクは削除できません。また、スキーマ名で *dblink* を修飾することはできません。これは、データベース・リンクの名前で、ピリオドが解析されるためです。そのため、たとえば、`ralph.linktosales` のような名前を付けた場合、スキーマ `ralph` の中の `linktosales` という名前のデータベース・リンクであると解析されるため、名前全体が自分のスキーマにあるデータベース・リンク名であると解析されます。

例

データベース・リンクの削除例 次の文は、sales.hq.acme.com という名前のプライベート・データベース・リンクを削除します。

```
DROP DATABASE LINK sales.hq.acme.com;
```

DROP DIMENSION

用途

DROP DIMENSION を使用すると、名前付けしたディメンションを削除できます。

注意： この文によって、ディメンションに指定された関連を使用するマテリアライズド・ビューが無効になるわけではありません。ただし、クエリー・リライトによって再書き込みされた要求が無効になり、そのようなビューに対する後続の操作の処理速度が遅くなることがあります。

参照：

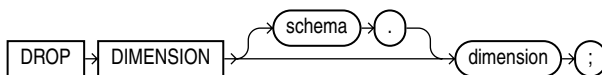
- ディメンションの作成については、12-39 ページの「[CREATE DIMENSION](#)」を参照してください。
- ディメンションの変更については、8-43 ページの「[ALTER DIMENSION](#)」を参照してください。
- 『Oracle9i データベース概要』を参照してください。

前提条件

この文を使用する場合、ディメンションが自分のスキーマ内に設定されているか、または DROP ANY DIMENSION システム権限が必要です。

構文

drop_dimension::=



キーワードとパラメータ

schema

ディメンションが格納されているスキーマの名前を指定します。*schema* を指定しない場合、そのディメンションは自分のスキーマにあるとみなされます。

dimension

削除するディメンションの名前を指定します。そのディメンションはすでに存在している必要があります。

例

DROP DIMENSION の例 この例は、sh.customers_dim を削除します。

```
DROP DIMENSION customers_dim;
```

参照： ディメンションの作成および変更の例については、12-43 ページの「[CREATE DIMENSION の例](#)」および 8-45 ページの「[ディメンションの変更例](#)」を参照してください。

DROP DIRECTORY

用途

DROP DIRECTORY 文を使用すると、データベースからディレクトリ・オブジェクトを削除できます。

参照： ディレクトリの作成については、12-44 ページの「[CREATE DIRECTORY](#)」を参照してください。

前提条件

ディレクトリを削除する場合は、DROP ANY DIRECTORY システム権限が必要です。

注意： PL/SQL または OCI プログラムによる関連ファイル・システム内のファイルへのアクセス中は、DROP によるディレクトリの削除はできません。

構文

drop_directory::=

DROP → DIRECTORY → *directory_name* → ;

キーワードとパラメータ

directory_name

削除するディレクトリ・データベース・オブジェクトの名前を指定します。

Oracle はディレクトリ・オブジェクトを削除しますが、サーバーのファイル・システム上の関連するオペレーティング・システム・ディレクトリは削除しません。

例

DROP DIRECTORY の例 次の文は、ディレクトリ・オブジェクト bfile_dir を削除します。

```
DROP DIRECTORY bfile_dir;
```

参照： 12-46 ページの「[CREATE DIRECTORY の例](#)」を参照してください。

DROP FUNCTION

用途

DROP FUNCTION 文を使用すると、データベースからスタンドアロン・ストアド・ファンクションを削除できます。

注意： この文を使用してパッケージの一部のファンクションを削除しないでください。DROP PACKAGE 文を使用して完全なパッケージを削除するか、OR REPLACE 句付きの CREATE PACKAGE 文を使用するファンクションを含まないパッケージを再定義します。

参照：

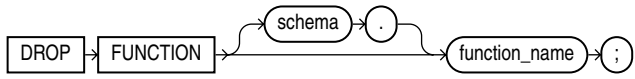
- ファンクションの作成については、12-47 ページの「[CREATE FUNCTION](#)」を参照してください。
- ファンクションの変更については、8-46 ページの「[ALTER FUNCTION](#)」を参照してください。

前提条件

削除するファンクションが自分のスキーマ内にあるか、または DROP ANY PROCEDURE システム権限が必要です。

構文

drop_function::=



キーワードとパラメータ

schema

ファンクションが含まれているスキーマを指定します。*schema* を指定しない場合、ファンクションが自分のスキーマ内に定義されているものとみなされます。

function_name

削除するファンクションの名前を指定します。

Oracle は、削除したファンクションに依存するローカル・オブジェクト、または削除したファンクションをコールするローカル・オブジェクトを無効にします。後でこのようなオブジェクトを参照した場合、Oracle は、そのオブジェクトを再コンパイルしようとします。削除したファンクションを再作成しないかぎり、エラーが戻されます。

任意の統計タイプがファンクションに関連付けられている場合、FORCE 句を使用して統計タイプの関連付けが解除され、統計タイプを使用して収集されたユーザー定義のすべての統計が削除されます。

参照：

- リモート・オブジェクトなどのスキーマ・オブジェクト間の依存性を Oracle が管理する方法の詳細は、『Oracle9i データベース概要』を参照してください。
- 統計タイプの関連の詳細は、11-46 ページの「[ASSOCIATE STATISTICS](#)」および 15-57 ページの「[DISASSOCIATE STATISTICS](#)」を参照してください。

例

DROP FUNCTION の例 次の文は、サンプル・スキーマ oe 内のファンクション SecondMax を削除します。この場合、SecondMax に依存するすべてのオブジェクトは無効になります。

```
DROP FUNCTION oe.SecondMax;
```

参照： SecondMax ファンクションの作成の詳細は、12-56 ページの「[CREATE FUNCTION の例](#)」を参照してください。

DROP INDEX

用途

DROP INDEX 文を使用すると、データベースから索引またはドメイン索引を削除できます。

索引を削除すると、ビュー、パッケージ、パッケージ本体、ファンクション、プロシージャなど、基になる表に依存するすべてのオブジェクトが削除されます。

グローバル・パーティション索引、レンジ・パーティション索引またはハッシュ・パーティション索引を削除した場合、すべての索引パーティションが同様に削除されます。コンポジット・パーティション索引を削除した場合、すべての索引パーティションおよびサブパーティションが同様に削除されます。

また、ドメイン索引を削除すると、次のようになります。

- 適切なルーチンが起動されます。これらのルーチンの詳細は、『Oracle9i Data Cartridge Developer's Guide』を参照してください。
- 統計タイプがドメイン索引に関連付けられている場合、FORCE 句を使用して統計タイプの関連付けが解除され、統計タイプを使用して収集されたユーザー定義の統計が削除されます。

参照：

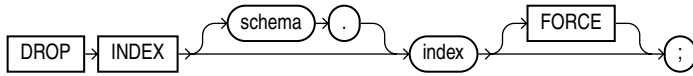
- 索引の作成については、12-58 ページの「[CREATE INDEX](#)」を参照してください。
- 索引の変更については、8-48 ページの「[ALTER INDEX](#)」を参照してください。
- ドメイン索引の詳細は、12-58 ページの「[CREATE INDEX](#)」の「[domain_index_clause](#)」を参照してください。
- 統計タイプの関連の詳細は、11-46 ページの「[ASSOCIATE STATISTICS](#)」および 15-57 ページの「[DISASSOCIATE STATISTICS](#)」を参照してください。

前提条件

索引が自分のスキーマ内にあるか、または DROP ANY INDEX システム権限が必要です。

構文

drop_index::=



キーワードとパラメータ

schema

削除する索引が含まれているスキーマを指定します。*schema* を指定しない場合、索引が自分のスキーマ内に定義されているものとみなされます。

index

削除する索引の名前を指定します。索引を削除した場合、その索引に割り当てられていたすべてのデータ・ブロックが索引の表領域に戻されます。

制限事項：索引または索引パーティションのいずれかに **IN_PROGRESS** のマークが付けられている場合は、ドメイン索引を削除できません。

FORCE

FORCE は、ドメイン索引のみに適用されます。この句を指定すると、索引タイプ・ルーチンの起動がエラーを戻した場合、または索引に **IN_PROGRESS** のマークが付けられている場合でも、ドメイン索引を削除できます。**FORCE** がないと、その索引タイプ・ルーチンの起動がエラーを戻す場合、または索引に **IN_PROGRESS** マークが付けられている場合にドメイン索引を削除できません。

例

DROP INDEX の例 この文は、12-78 ページの「[一般的な索引の例](#)」で作成した `ord_customer_ix` という名前の索引を削除します。

```
DROP INDEX ord_customer_ix;
```

DROP INDEXTYPE

用途

DROP INDEXTYPE 文を使用すると、索引タイプを削除できます。同時に、統計タイプとの関連付けも解除されます。

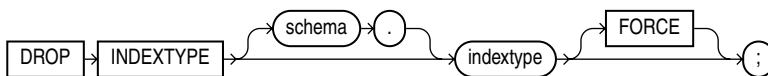
参照： 索引タイプの詳細は、12-84 ページの「[CREATE INDEXTYPE](#)」を参照してください。

前提条件

索引タイプが自分のスキーマ内にあるか、または DROP ANY INDEXTYPE システム権限が必要です。

構文

drop_indextype::=



キーワードとパラメータ

schema

索引タイプが含まれているスキーマを指定します。*schema* を指定しない場合、索引タイプが自分のスキーマ内に定義されているものとみなされます。

indextype

削除する索引タイプの名前を指定します。

任意の統計タイプが索引タイプに関連付けられている場合、統計タイプと索引タイプの関連付けが解除され、統計タイプを使用して収集されたすべての統計が削除されます。

参照： 統計情報の関連付けの詳細は、11-46 ページの「[ASSOCIATE STATISTICS](#)」および 15-57 ページの「[DISASSOCIATE STATISTICS](#)」を参照してください。

FORCE

FORCE を指定すると、索引タイプが 1 つ以上のドメイン索引から現在参照されている状態でも、索引タイプを削除できます。これらのドメイン索引に INVALID のマークが付けられます。FORCE を指定しない場合、任意のドメイン索引が索引タイプを参照していると索引タイプを削除できません。

例

DROP INDEXTYPE の例 次の文は、索引タイプ `textindextype` を削除し、この索引タイプで定義されているすべてのドメイン索引に INVALID のマークを付けます。

```
DROP INDEXTYPE textindextype FORCE;
```

DROP JAVA

用途

DROP JAVA 文を使用すると、Java ソース、クラスまたはリソース・スキーマ・オブジェクトを削除できます。

参照：

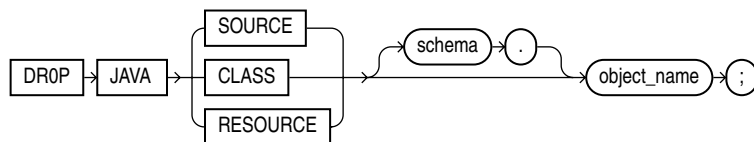
- Java オブジェクトの作成については、12-86 ページの「[CREATE JAVA](#)」を参照してください。
- Java ソース、クラスおよびリソースの変換の詳細は、『Oracle9i Java Stored Procedures Developer's Guide』を参照してください。

前提条件

Java ソース、クラスまたはリソースが自分のスキーマ内にあるか、または DROP ANY PROCEDURE システム権限が必要です。また、このコマンドを使用する場合は、Java クラスに対する EXECUTE オブジェクト権限が必要です。

構文

drop java::=



キーワードとパラメータ

JAVA SOURCE

SOURCE を指定すると、Java ソース・スキーマ・オブジェクトおよびそのオブジェクトから導出されたすべての Java クラス・スキーマ・オブジェクトが削除されます。

JAVA CLASS

CLASS を指定すると、Java クラス・スキーマ・オブジェクトが削除されます。

JAVA RESOURCE

RESOURCE を指定すると、Java リソース・スキーマ・オブジェクトが削除されます。

object_name

既存の Java クラス、ソースまたはリソース・スキーマ・オブジェクトの名前を指定します。
object_name を二重引用符で囲むと、小文字の名前、または大文字と小文字を組み合わせた名前を指定できます。

例

DROP JAVA CLASS の例 次の文は、Java クラス `MyClass` を削除します。

```
DROP JAVA CLASS "MyClass";
```

DROP LIBRARY

用途

DROP LIBRARY を使用すると、データベースから外部プロシージャ・ライブラリを削除できます。

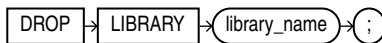
参照： ライブラリ作成については、13-2 ページの「[CREATE LIBRARY](#)」を参照してください。

前提条件

DROP LIBRARY システム権限が必要です。

構文

drop_library::=



キーワードとパラメータ

library_name

削除対象の外部プロシージャ・ライブラリの名前を指定します。

例

DROP LIBRARY の例 次の文は、13-3 ページの「[CREATE LIBRARY の例](#)」で作成した ext_lib ライブラリを削除します。

```
DROP LIBRARY ext_lib;
```

DROP MATERIALIZED VIEW

用途

DROP MATERIALIZED VIEW 文を使用すると、データベースから既存のマテリアライズド・ビューを削除できます。

注意： キーワード `SNAPSHOT` は、`MATERIALIZED VIEW` のかわりに下位互換用にサポートされています。

参照：

- 種類の説明を含むマテリアライズド・ビューの詳細は、13-5 ページの「[CREATE MATERIALIZED VIEW](#)」を参照してください。
- マテリアライズド・ビューの変更については、8-74 ページの「[ALTER MATERIALIZED VIEW](#)」を参照してください。
- レプリケーション環境でのマテリアライズド・ビューについては、『Oracle9i レプリケーション』を参照してください。
- データ・ウェアハウス環境でのマテリアライズド・ビューについては、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

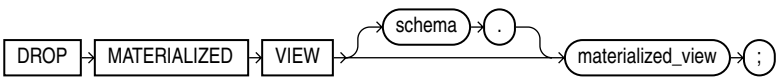
前提条件

マテリアライズド・ビューが自分のスキーマ内にあるか、または `DROP ANY MATERIALIZED VIEW` システム権限が必要です。また、Oracle がマテリアライズド・ビューのデータを管理するために使用する内部表、ビュー、索引を削除する権限も必要です。

参照： マテリアライズド・ビューを管理するために使用するオブジェクトの削除に必要な権限については、16-6 ページの「[DROP TABLE](#)」、16-21 ページの「[DROP VIEW](#)」および 15-71 ページの「[DROP INDEX](#)」を参照してください。

構文

drop_materialized_view::=



キーワードとパラメータ

schema

マテリアライズド・ビューが含まれているスキーマを指定します。*schema* を指定しない場合、マテリアライズド・ビューは自分のスキーマ内にあるとみなされます。

materialized_view

削除する既存のマテリアライズド・ビューの名前を指定します。

- マスター表から、最後にリフレッシュされた単純マテリアライズド・ビューを削除した場合、マスター表のマテリアライズド・ビュー・ログのうち、削除されたマテリアライズド・ビューのリフレッシュに必要な行のみが自動的に削除されます。
- マスター表を削除した場合、Oracle は、その表に基づくマテリアライズド・ビューを自動的に削除しません。ただし、削除したマスター表に基づくマテリアライズド・ビューをリフレッシュしようとした場合、エラー・メッセージが戻されます。
- マテリアライズド・ビューを削除した場合、マテリアライズド・ビューを使用するために再書き込みされたコンパイル済の要求が無効になり、自動的に再コンパイルされます。事前にマテリアライズド・ビューが作成されている場合、その表は削除されませんが、マテリアライズド・ビューのリフレッシュ・メカニズムによる管理ができなくなります。

例

DROP MATERIALIZED VIEW の例 次の文は、サンプル・スキーマ sh のマテリアライズド・ビュー mv1 を削除します。

```
DROP MATERIALIZED VIEW mv1;
```

次の文は、sales_by_month_by_state マテリアライズド・ビューおよびマテリアライズド・ビューの基礎となる表を削除します（基礎となる表が ON PREBUILT TABLE 句付きで CREATE MATERIALIZED VIEW 文に登録されていない場合）。

```
DROP MATERIALIZED VIEW sales_by_month_by_state;
```

DROP MATERIALIZED VIEW LOG

用途

DROP MATERIALIZED VIEW LOG 文を使用すると、データベースからマテリアライズド・ビュー・ログを削除できます。

注意： キーワード `SNAPSHOT` は、`MATERIALIZED VIEW` のかわりに下位互換用にサポートされています。

参照：

- マテリアライズド・ビューについては、13-5 ページの「[CREATE MATERIALIZED VIEW](#)」および 8-74 ページの「[ALTER MATERIALIZED VIEW](#)」を参照してください。
- マテリアライズド・ビュー・ログについては、13-29 ページの「[CREATE MATERIALIZED VIEW LOG](#)」を参照してください。
- レプリケーション環境でのマテリアライズド・ビューについては、『Oracle9i レプリケーション』を参照してください。
- データ・ウェアハウス環境でのマテリアライズド・ビューについては、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

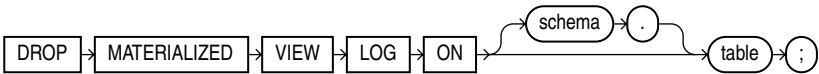
前提条件

マテリアライズド・ビュー・ログを削除する場合は、表を削除する権限が必要です。

参照： 16-6 ページの「[DROP TABLE](#)」を参照してください。

構文

`drop_materialized_view_log::=`



キーワードとパラメータ

schema

削除するマテリアライズド・ビュー・ログおよびそのマスター表が含まれているスキーマを指定します。*schema* を指定しない場合、マテリアライズド・ビュー・ログおよびマスター表は自分のスキーマ内にあるとみなされます。

table

削除するマテリアライズド・ビュー・ログに関連付けられたマスター表の名前を指定します。

マテリアライズド・ビュー・ログを削除した場合、マテリアライズド・ビュー・ログのマスター表に基づく一部のマテリアライズド・ビューが高速リフレッシュできなくなります。これらのマテリアライズド・ビューは、ROWID マテリアライズド・ビュー、主キー・マテリアライズド・ビューおよび副問合せビューを含みます。

参照： マテリアライズド・ビューの種類については、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

例

DROP MATERIALIZED VIEW LOG の例 次の文は、customers マスター表のマテリアライズド・ビュー・ログを削除します。

```
DROP MATERIALIZED VIEW LOG ON customers;
```

DROP OPERATOR

用途

DROP OPERATOR 文を使用すると、ユーザー定義演算子を削除できます。

参照：

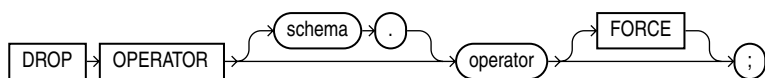
- 演算子の作成については、13-38 ページの「[CREATE OPERATOR](#)」を参照してください。
- 演算子の概要については、3-6 ページの「[ユーザー定義演算子](#)」および『Oracle9i Data Cartridge Developer's Guide』を参照してください。
- ユーザー定義索引タイプの演算子の削除については、8-69 ページの「[ALTER INDEXTYPE](#)」を参照してください。

前提条件

演算子が自分のスキーマ内にあるか、または DROP ANY OPERATOR システム権限が必要です。

構文

drop_operator::=



キーワードとパラメータ

schema

演算子が含まれているスキーマを指定します。 *schema* を指定しない場合、その演算子が自分のスキーマにあるとみなされます。

operator

削除する演算子の名前を指定します。

FORCE

FORCE を指定すると、演算子が現在 1 つ以上のスキーマ・オブジェクト（索引タイプ、パッケージ、ファンクション、プロシージャなど）によって参照されている場合でも、その演算子を削除し、演算子に依存するオブジェクトに INVALID のマークが付きます。FORCE を使用しないと、任意のスキーマ・オブジェクトが演算子を参照しているときに演算子を削除できません。

例

DROP OPERATOR の例 次の文は、演算子 merge を削除します。

```
DROP OPERATOR oe.merge;
```

FORCE 句が指定されていないため、この演算子のバインディングのいずれかが索引タイプによって参照されている場合、この操作は実行されません。

DROP OUTLINE

用途

DROP OUTLINE 文を使用すると、ストアド・アウトラインを削除できます。

参照：

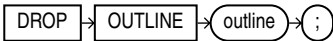
- アウトラインの作成については、13-42 ページの「[CREATE OUTLINE](#)」を参照してください。
- アウトラインについては、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。

前提条件

アウトラインを削除する場合は、DROP ANY OUTLINE システム権限が必要です。

構文

drop_outline::=



キーワードとパラメータ

outline

削除するアウトラインの名前を指定します。

そのアウトラインが削除された後、ストアド・アウトラインが作成された SQL 文がコンパイルされると、オブティマイザはアウトラインの影響なしに新しい実行計画を生成します。

例

DROP OUTLINE の例 次の文は、ストアド・アウトライン salaries を削除します。

```
DROP OUTLINE salaries;
```


DROP PACKAGE

用途

DROP PACKAGE 文を使用すると、データベースからストアド・パッケージを削除できます。この文は、パッケージの本体および仕様部を削除します。

注意： この文を使用してパッケージから単一のオブジェクトを削除しないでください。かわりに、CREATE PACKAGE 文および CREATE PACKAGE BODY 文に OR REPLACE 句を指定して、そのオブジェクトがないパッケージを再作成してください。

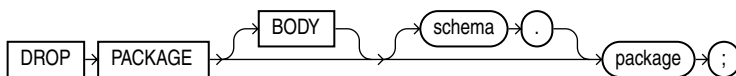
参照： 13-46 ページの「[CREATE PACKAGE](#)」を参照してください。

前提条件

削除するパッケージが自分のスキーマ内にあるか、または DROP ANY PROCEDURE システム権限が必要です。

構文

drop_package::=



キーワードとパラメータ

BODY

BODY を指定すると、パッケージ本体のみを削除できます。この句を省略した場合、パッケージ本体と仕様部の両方が削除されます。

パッケージ本体のみを削除して仕様部は削除しない場合、依存するオブジェクトは無効になりません。ただし、パッケージ本体を再作成しないかぎり、パッケージ仕様部で宣言したプロシージャやストアド・ファンクションはコールできません。

schema

パッケージが含まれているスキーマを指定します。*schema* を指定しない場合、パッケージが自分のスキーマ内に定義されているとみなされます。

package

削除するパッケージの名前を指定します。

そのパッケージ仕様部に依存するすべてのローカル・オブジェクトが無効になります。後でこのようなオブジェクトのいずれかを参照した場合、Oracle がそのオブジェクトを再コンパイルしようとします。削除したパッケージを再作成しないかぎり、エラーが戻されます。

統計タイプがパッケージに関連付けられている場合、FORCE 句を使用して統計タイプの関連付けが解除され、統計タイプを使用して収集されたユーザー定義のすべての統計が削除されます。

参照：

- リモート・オブジェクトなどのスキーマ・オブジェクト間の依存性を Oracle が管理する方法の詳細は、『Oracle9i データベース概要』を参照してください。
- 11-46 ページの「[ASSOCIATE STATISTICS](#)」および 15-57 ページの「[DISASSOCIATE STATISTICS](#)」を参照してください。

例

DROP PACKAGE の例 次の文は、13-49 ページの「[CREATE PACKAGE の例](#)」で作成した emp_mgmt パッケージの仕様部および本体を削除し、その仕様部に依存するすべてのオブジェクトを無効にします。

```
DROP PACKAGE emp_mgmt;
```

DROP PROCEDURE

用途

DROP PROCEDURE 文を使用すると、データベースからスタンドアロンのストアド・プロシージャを削除できます。この文を使用してパッケージの一部であるプロシージャを削除しないでください。DROP PACKAGE 文を使用して完全にパッケージを削除するか、OR REPLACE 句付きの CREATE PACKAGE 文を使用して、プロシージャを含まないパッケージを再定義します。

参照：

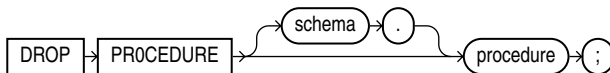
- プロシージャの作成については、13-58 ページの「[CREATE PROCEDURE](#)」を参照してください。
- プロシージャの変更については、8-103 ページの「[ALTER PROCEDURE](#)」を参照してください。

前提条件

削除するプロシージャが自分のスキーマ内にあるか、または DROP ANY PROCEDURE システム権限が必要です。

構文

drop_procedure::=



キーワードとパラメータ

schema

プロシージャが含まれているスキーマを指定します。*schema* を指定しない場合、プロシージャが自分のスキーマ内に定義されているとみなされます。

procedure

削除するプロシージャの名前を指定します。

プロシージャを削除すると、そのプロシージャに依存するすべてのローカル・オブジェクトは無効になります。後でこのようなオブジェクトのいずれかを参照した場合、**Oracle** がそのオブジェクトを再コンパイルしようとします。削除したプロシージャを再作成しないかぎり、エラー・メッセージが戻されます。

参照： リモート・オブジェクトなどのスキーマ・オブジェクト間の依存性を **Oracle** が管理する方法の詳細は、『**Oracle9i データベース概要**』を参照してください。

例

DROP PROCEDURE の例 次の文は、ユーザー **oe** が所有するプロシージャ **credit** を削除し、**transfer** に依存するすべてのオブジェクトを無効にします。

```
DROP PROCEDURE oe.credit;
```

DROP PROFILE

用途

DROP PROFILE 文を使用すると、データベースからプロファイルを削除できます。

参照：

- プロファイルの作成については、13-65 ページの「[CREATE PROFILE](#)」を参照してください。
- プロファイルの変更については、8-106 ページの「[ALTER PROFILE](#)」を参照してください。

前提条件

DROP PROFILE システム権限が必要です。

構文

drop_profile::=



キーワードとパラメータ

profile

削除するプロファイルの名前を指定します。

制限事項：DEFAULT プロファイルは削除できません。

CASCADE

CASCADE を指定すると、削除するプロファイルが割り当てられているすべてのユーザーから、そのプロファイルの割当てが解除されます。このようなユーザーに対しては、DEFAULT プロファイルが自動的に割り当てられます。現在、複数のユーザーに割り当てられているプロファイルを削除する場合は、必ずこの句を指定します。

例

DROP PROFILE の例 次の文は、13-70 ページの「[CREATE PROFILE の例](#)」で作成したプロファイル `app_user` を削除します。

```
DROP PROFILE app_user CASCADE;
```

`app_user` プロファイルが削除され、現在、`app_user` プロファイルが割り当てられているユーザーについては、`DEFAULT` プロファイルが割り当てられます。

DROP ROLE

用途

DROP ROLE 文を使用すると、データベースからロールを削除できます。ロールを削除した場合、そのロールが付与されていたすべてのユーザーからそのロールが取り消され、データベースからも削除されます。すでに使用可能なロールのユーザー・セッションには影響しません。ただし、削除後は新しいユーザー・セッションはロールを使用可能にできません。

参照：

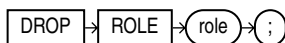
- ロールの作成の詳細は、13-72 ページの「[CREATE ROLE](#)」を参照してください。
- ロールを使用可能にするときに必要な認証の変更については、8-113 ページの「[ALTER ROLE](#)」を参照してください。
- 現行のセッションに対するロールの使用禁止については、17-43 ページの「[SET ROLE](#)」を参照してください。

前提条件

ADMIN OPTION 付きのロールが付与されているか、または DROP ANY ROLE システム権限が必要です。

構文

drop_role::=



キーワードとパラメータ

role

削除するロールの名前を指定します。

例

DROP ROLE の例 次の文は、13-74 ページの「[CREATE ROLE の例](#)」で作成したロール dw_manager を削除します。

```
DROP ROLE dw_manager;
```

DROP ROLLBACK SEGMENT

用途

DROP ROLLBACK SEGMENT 文を使用すると、データベースからロールバック・セグメントを削除できます。ロールバック・セグメントを削除した場合、ロールバック・セグメントに割り当てられていたすべての領域が表領域に戻されます。

注意： データベースが自動 UNDO 管理モードで起動されている場合、ロールバック・セグメントに有効な操作はこの文のみです。このモードの場合、ロールバック・セグメントを作成または変更できません。

参照：

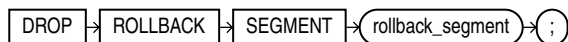
- ロールバック・セグメントの作成については、13-75 ページの「[CREATE ROLLBACK SEGMENT](#)」を参照してください。
- ロールバック・セグメントの変更については、8-115 ページの「[ALTER ROLLBACK SEGMENT](#)」を参照してください。
- 14-62 ページの「[CREATE TABLESPACE](#)」を参照してください。

前提条件

DROP ROLLBACK SEGMENT システム権限が必要です。

構文

`drop_rollback_segment::=`



キーワードとパラメータ

rollback_segment

削除するロールバック・セグメントの名前を指定します。

制限事項：

- ロールバック・セグメントがオフラインになっている場合にのみ削除できます。ロールバック・セグメントがオフラインであるかどうかを判断するには、データ・ディクショナリ・ビュー `DBA_ROLLBACK_SEGS` に問い合わせます。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` 文に `OFFLINE` 句を指定して、ロールバック・セグメントをオフラインにすることもできます。
- `SYSTEM` ロールバック・セグメントは削除できません。

例

DROP ROLLBACK SEGMENT の例 次の文は、ロールバック・セグメント `rhs_1` を削除します。

```
DROP ROLLBACK SEGMENT rhs_1;
```

SQL 文 : DROP SEQUENCE ~ ROLLBACK

この章では、次の SQL 文について説明します。

- DROP SEQUENCE
- DROP SYNONYM
- DROP TABLE
- DROP TABLESPACE
- DROP TRIGGER
- DROP TYPE
- DROP TYPE BODY
- DROP USER
- DROP VIEW
- EXPLAIN PLAN
- *filespec*
- GRANT
- INSERT
- LOCK TABLE
- MERGE
- NOAUDIT
- RENAME
- REVOKE
- ROLLBACK

DROP SEQUENCE

用途

DROP SEQUENCE を使用すると、データベースから順序を削除できます。

この文を使用した場合、順序の削除および再作成を行って、順序を再開できます。たとえば、現在、値が 150 の順序を初期値 27 で再開する場合、順序を削除して同じ名前で再作成し、START WITH 値を 27 にします。

参照：

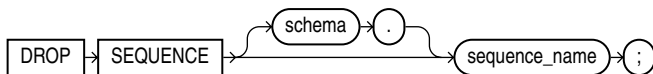
- 順序の作成の詳細については、13-81 ページの「[CREATE SEQUENCE](#)」を参照してください。
- 順序の変更については、8-119 ページの「[ALTER SEQUENCE](#)」を参照してください。

前提条件

削除する順序が自分のスキーマ内にあるか、または DROP ANY SEQUENCE システム権限が必要です。

構文

drop_sequence::=



キーワードとパラメータ

schema

順序が含まれているスキーマを指定します。*schema* の指定を省略した場合、順序が自分のスキーマ内に定義されているとみなされます。

sequence_name

削除する順序の名前を指定します。

例

DROP SEQUENCE の例 次の文は、13-85 ページの「[CREATE SEQUENCE の例](#)」で作成した、ユーザー oe が所有する順序 orders_seq を削除します。この文を発行する場合は、ユーザー oe として接続するか、または DROP ANY SEQUENCE システム権限が必要です。

```
DROP SEQUENCE oe.orders_seq;
```

DROP SYNONYM

用途

DROP SYNONYM 文を使用すると、データベースからシノニムを削除できます。また、シノニムの削除および再作成を行って、シノニムの定義を変更することもできます。

参照： シノニムの詳細は、14-2 ページの「[CREATE SYNONYM](#)」を参照してください。

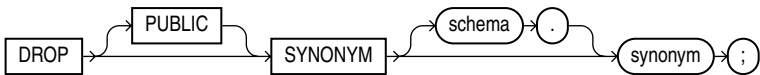
前提条件

プライベート・シノニムを削除する場合は、そのシノニムが自分のスキーマ内にあるか、または DROP ANY SYNONYM システム権限が必要です。

パブリック・シノニムを削除する場合は、DROP PUBLIC SYNONYM システム権限が必要です。

構文

drop_synonym::=



キーワードとパラメータ

PUBLIC

パブリック・シノニムを削除する場合は、PUBLIC を指定する必要があります。PUBLIC を指定した場合、*schema* は指定できません。

schema

シノニムが含まれているスキーマを指定します。*schema* の指定を省略した場合、シノニムが自分のスキーマ内に定義されているとみなされます。

synonym

削除するシノニムの名前を指定します。

マテリアライズド・ビューのシノニム、表またはマテリアライズド・ビューを含むマテリアライズド・ビュー、あるいはマテリアライズド・ビューが依存する表が削除された場合、マテリアライズド・ビューは無効になります。

例

DROP SYNONYM の例 次の文は、14-5 ページの「[シノニムの変換例](#)」で作成した、locations という名前のパブリック・シノニムを削除します。

```
DROP PUBLIC SYNONYM locations;
```

DROP TABLE

用途

DROP TABLE 文を使用すると、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データベースから削除できます。

注意： DROP TABLE 文はロールバックできません。

注意： 外部表の場合、この文はデータベースにある表のメタデータのみを削除します。データベースの外部に存在する実際のデータには影響しません。

表を削除すると、表の依存オブジェクトが無効になり、表のオブジェクト権限が削除されます。表を再作成する場合、表のオブジェクト権限を再び付与し、表の索引、整合性制約およびトリガーを作成し、記憶域パラメータを再指定する必要があります。表の切捨ての場合、このような影響はありません。そのため、表を削除して再作成するより、TRUNCATE 文で行を削除するほうが効果的です。

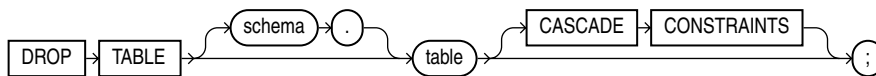
- 参照：**
- 表の作成については、14-6 ページの「[CREATE TABLE](#)」を参照してください。
 - 表の変更については、10-2 ページの「[ALTER TABLE](#)」を参照してください。
 - 表を削除せずに表からデータを削除する場合の詳細は、17-59 ページの「[TRUNCATE](#)」および 15-49 ページの「[DELETE](#)」を参照してください。

前提条件

削除する表が自分のスキーマ内にあるか、または DROP ANY TABLE システム権限が必要です。

構文

drop_table::=



キーワードとパラメータ

schema

変更する表が定義されているスキーマを指定します。*schema* を指定しない場合、この表が自分のスキーマに存在するとみなされます。

table

削除する表、オブジェクト表または索引構成表の名前を指定します。Oracle によって次の操作が自動的に実行されます。

- 表からすべての行が削除されます。
- 表のすべての索引およびドメイン索引が、作成者やスキーマの所有者とは関係なく削除されます。*table* がパーティション化されている場合、対応するローカル索引パーティションも同様に削除されます。
- *table* のネストした表および LOB のすべての記憶表が削除されます。
- レンジ・パーティション表またはハッシュ・パーティション表を削除すると、すべての表パーティションも削除されます。コンポジット・パーティション表を削除した場合、すべてのパーティションおよびサブパーティションが同様に削除されます。
- 索引構成表の場合、索引構成表に定義したマッピング表も削除されます。
- ドメイン索引の場合は、この文によって適切な削除ルーチンが起動されます。

参照： これらのルーチンの詳細は、『Oracle9i Data Cartridge Developer's Guide』を参照してください。

- 任意の統計タイプが表に関連付けられている場合、FORCE 句を使用して統計タイプの関連付けが解除され、統計タイプを使用して収集されたユーザー定義のすべての統計が削除されます。

参照： 統計タイプの関連の詳細は、11-46 ページの「[ASSOCIATE STATISTICS](#)」および 15-57 ページの「[DISASSOCIATE STATISTICS](#)」を参照してください。

- 表がクラスタの一部でない場合は、表とその索引に割り当てられていたすべてのデータ・ブロックを、表とその索引が格納されていた表領域に戻します。

注意： DROP CLUSTER 文に INCLUDING TABLES 句を指定した場合、クラスタおよびそのすべての表を削除できます。これによって、表を1つずつ削除する必要がなくなります。詳細は、15-60 ページの「[DROP CLUSTER](#)」を参照してください。

- 表がビュー、マテリアライズド・ビューのコンテンツ表またはマスター表の実表である場合、あるいは表がストアド・プロシージャ、ファンクションまたはパッケージで参照されている場合、依存するオブジェクトは無効になりますが、削除はされません。表を再作成するか、これらのオブジェクトを一度削除してその表に依存しない形で再作成しないかぎり、これらのオブジェクトは使用できません。
- 表を再作成する場合、ストアド・プロシージャ、ファンクションまたはパッケージで参照する列、およびマテリアライズド・ビューの定義で使用されていた副問合せで選択されるすべての列が、その表に含まれている必要があります。ビュー、ストアド・プロシージャ、ファンクション、パッケージに対するオブジェクト権限を付与されていたユーザーに、これらの権限を再付与する必要はありません。
- 表がマテリアライズド・ビューのマスター表の場合、マテリアライズド・ビューの間合せはできます。ただし、そのマテリアライズド・ビューの副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かぎり、リフレッシュはできません。
- 表がマテリアライズド・ビュー・ログを持つ場合、このログおよびその表に関連付けられているダイレクト・パス INSERT のリフレッシュ情報は削除されます。

制限事項： ネストした表の記憶表は直接削除できません。かわりに、ALTER TABLE ... DROP COLUMN 句を使用して、ネストした表の列を削除する必要があります。

CASCADE CONSTRAINTS

CASCADE CONSTRAINTS は、削除した表の主キーまたは一意キーを参照するすべての参照整合性制約を削除します。このような参照整合性制約があるときにこの句の指定を省略した場合、エラーが戻され、表は削除されません。

例

DROP TABLE の例 次の文は、oe.orders 表を削除します。

```
DROP TABLE oe.orders;
```

DROP TABLESPACE

用途

DROP TABLESPACE を使用すると、データベースから表領域を削除できます。

参照：

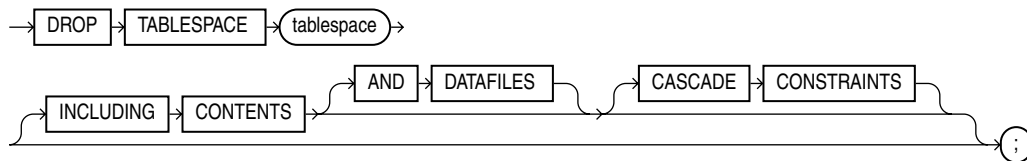
- 表領域の作成については、14-62 ページの「[CREATE TABLESPACE](#)」を参照してください。
- 表領域の変更については、10-82 ページの「[ALTER TABLESPACE](#)」を参照してください。

前提条件

DROP TABLESPACE システム権限が必要です。アクティブ・トランザクションを保持するロールバック・セグメントを含む場合は、表領域を削除できません。

構文

drop_tablespace::=



キーワードとパラメータ

tablespace

削除する表領域の名前を指定します。

表領域の状態がオンラインまたはオフラインのどちらであっても、その表領域を削除できません。実行中のトランザクション内の SQL 文で、表領域内のいずれかのオブジェクトにアクセスすることがないように、表領域はオフラインにしてから削除することをお勧めします。

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することはできません。ALTER USER 文を使用すると、ユーザーに新しいデフォルト表領域および一時表領域を割り当てることができます。

表領域にあるすべてのデータ・ファイルおよびテンポラリ・ファイルに関するすべてのメタデータが、データ・ディクショナリから削除されます。表領域にある Oracle 管理データ・ファイルおよびテンポラリ・ファイルが、オペレーティング・システムから自動的に削除されます。その他のデータ・ファイルおよびテンポラリ・ファイルは、INCLUDING CONTENTS AND DATAFILES を指定しないかぎり、オペレーティング・システムから削除されません。

制限事項：

- SYSTEM 表領域は削除できません。
- ドメイン索引またはドメイン索引によって作成されたオブジェクトを格納している表領域は削除できません。
- いずれかのインスタンスによって使用されている場合、またはコミットされていないトランザクションのロールバックに必要な UNDO データを含む場合は、UNDO 表領域を削除できません。

参照： ドメイン索引の詳細は、『Oracle9i Data Cartridge Developer's Guide』および『Oracle9i データベース概要』を参照してください。

INCLUDING CONTENTS

INCLUDING CONTENTS は、表領域の中のすべてのデータベース・オブジェクトを削除します。データベース・オブジェクトを格納している表領域を削除する場合は、必ずこの句を指定します。表領域が空でない場合にこの句を省略した場合、エラーが戻され、表領域は削除されません。

パーティション表の場合、次のパーティションが表領域に（すべてではなく）一部含まれていると、INCLUDING CONTENTS を指定しても DROP TABLESPACE コマンドが正常に実行されません。

- レンジ・パーティション表またはハッシュ・パーティション表のパーティション
- コンポジット・パーティション表のサブパーティション

注意： パーティション表のすべてのパーティションが *tablespace* に存在する場合、DROP TABLESPACE ... INCLUDING CONTENTS は、*tablespace* を削除し、関連付けられた索引セグメント、LOB データ・セグメントおよび他の表領域にある LOB 索引セグメントも削除します。

パーティション化された索引構成表の場合、すべての主キー索引セグメントがこの表領域に存在する場合、この句は、他の表領域にあるオーバーフロー・セグメントも、他の表領域にある関連するマッピング表と同様に削除します。キー索引セグメントのいくつかが存在しない場合、その文は実行されません。その場合、その表領域を削除する前に、`ALTER TABLE ... MOVE PARTITION` を使用して、それらの主キー索引セグメントをこの表領域に移動し、この表領域にオーバーフロー・データ・セグメントの存在しないパーティションを削除します。また、パーティション化された索引構成表も削除します。

表領域がマテリアライズド・ビューのマスター表を含む場合、マテリアライズド・ビューは無効になります。

表領域がマテリアライズド・ビュー・ログを含む場合、このログおよびその表に関連付けられているダイレクト・パス `INSERT` のリフレッシュ情報は削除されます。

AND DATAFILES

`INCLUDING CONTENTS` を指定すると、`AND DATAFILES` 句によって、関連するオペレーティング・システム・ファイルも同様に削除されます。アラート・ログに各オペレーティング・システム・ファイルに対するメッセージが書き込まれます。この句は Oracle 管理ファイルには必要ではありません。

CASCADE CONSTRAINTS

`CASCADE CONSTRAINTS` は、*tablespace* に含まれる表の主キーまたは一意キーを参照する、*tablespace* の外の表にあるすべての参照整合性制約を削除します。このような参照整合性制約がある際にこの句の指定を省略した場合、エラーが戻され、表領域は削除されません。

例

DROP TABLESPACE の例 次の文は、`tbs_1` 表領域とそのすべての内容を削除します。

```
DROP TABLESPACE tbs_1
INCLUDING CONTENTS
CASCADE CONSTRAINTS;
```

オペレーティング・システム・ファイルの削除例 次の例は、`tbs_2` 表領域およびそれに関連するすべてのオペレーティング・システムのデータ・ファイルを削除します。

```
DROP TABLESPACE tbs_2
INCLUDING CONTENTS AND DATAFILES;
```

DROP TRIGGER

用途

DROP TRIGGER を使用すると、データベースからデータベース・トリガーを削除できます。

参照：

- トリガーの作成については、14-77 ページの「[CREATE TRIGGER](#)」を参照してください。
- トリガーの使用可能化、使用禁止化またはコンパイルの詳細は、11-2 ページの「[ALTER TRIGGER](#)」を参照してください。

前提条件

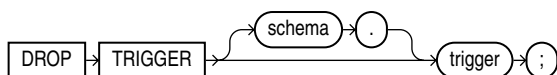
削除するトリガーが自分のスキーマ内にあるか、または DROP ANY TRIGGER システム権限が必要です。

他のユーザーのスキーマ内のデータベースにあるトリガーを削除する場合は、ADMINISTER DATABASE TRIGGER システム権限が必要です。

参照： これらの権限の詳細は、14-77 ページの「[CREATE TRIGGER](#)」を参照してください。

構文

drop_trigger::=



キーワードとパラメータ

schema

削除するトリガーが含まれているスキーマを指定します。*schema* を指定しない場合、トリガーは自分のスキーマ内に定義されているとみなされます。

trigger

削除するトリガーの名前を指定します。データベースからトリガーが削除され、再度、起動されることはありません。

例

DROP TRIGGER の例 次の文は、スキーマ `oe` 内のトリガー `order` を削除します。

```
DROP TRIGGER oe.order;
```

DROP TYPE

用途

DROP TYPE 文を使用すると、オブジェクト型、VARRAY 型またはネストした表型の仕様部および本体を削除できます。

参照：

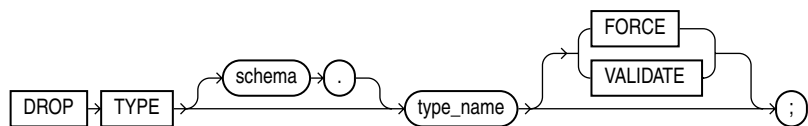
- オブジェクト型の本体のみの削除については、16-17 ページの「[DROP TYPE BODY](#)」を参照してください。
- 型の作成については、15-3 ページの「[CREATE TYPE](#)」を参照してください。

前提条件

削除するオブジェクト型、VARRAY 型またはネストした表型が自分のスキーマ内にあるか、または DROP ANY TYPE システム権限が必要です。

構文

drop_type::=



キーワードとパラメータ

schema

型が含まれているスキーマを指定します。*schema* を指定しない場合、その型が自分のスキーマに存在するとみなされます。

type_name

削除するオブジェクト型、VARRAY 型またはネストした表型の名前を指定します。型依存性または表依存性のない型のみ削除できます。

type_name がスーパータイプの場合、FORCE も指定しないと、この文は正常に実行されません。FORCE を指定すると、そのスーパータイプに依存するすべてのサブタイプが無効になります。

type_name が統計タイプの場合、FORCE も指定しないと、この文は正常に実行されません。FORCE を指定した場合、最初に *type_name* に関連付けられたすべてのオブジェクトの関連付けが解除され、*type_name* が削除されます。

参照： 統計タイプの詳細は、11-46 ページの「[ASSOCIATE STATISTICS](#)」および 15-57 ページの「[DISASSOCIATE STATISTICS](#)」を参照してください。

type_name が統計タイプに関連付けられたオブジェクト型の場合、最初にこの統計タイプから *type_name* の関連付けが解除され、*type_name* が削除されます。ただし、統計タイプを使用して統計が収集された場合、この統計タイプから *type_name* の関連付けを解除することはできません。このため、この文は正常に実行されません。

type_name が索引タイプの実装タイプの場合、索引タイプに INVALID のマークが付けられます。

FORCE オプションを指定していない場合に削除できるのは、依存性のないスタンドアロンのスキーマ・オブジェクトとして定義されているオブジェクト型またはネストした表型または VARRAY 型のみです。これはデフォルトの動作です。

参照： 12-84 ページの「[CREATE INDEXTYPE](#)」を参照してください。

FORCE

FORCE は、依存性オブジェクトがある型でも強制的に削除します。削除する型に依存するすべての列に UNUSED のマークが付けられ、それらの列にアクセスできなくなります。

注意： FORCE を使用して、依存性のある型を削除することはお勧めしません。この操作は、リカバリが不可能であり、依存表または列のデータにアクセスできなくなる場合があります。型の依存性の詳細は、『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

VALIDATE

型を削除するときに **VALIDATE** を指定すると、格納されているこの型のインスタンスがスーパータイプのいずれかの置換可能な列の範囲であることが検証されます。このようなインスタンスがない場合、**Oracle** は削除操作を完了します。

この句はサブタイプのみに意味があります。明示的な型または表依存性のないサブタイプを安全に削除するために、このオプションの使用をお勧めします。

例

DROP TYPE の例 次の文は、15-20 ページの「[型の階層例](#)」で作成したオブジェクト型 `person_t` を削除します。

```
DROP TYPE person_t;
```

DROP TYPE BODY

用途

DROP TYPE BODY 文を使用すると、オブジェクト型、VARRAY 型またはネストした表型の本体を削除できます。型本体を削除しても、オブジェクト型の仕様部は残ります。また、削除した型本体は再作成できます。その本体を再作成する場合、型本体を削除したオブジェクト型は引き続き使用できますが、メンバー・ファンクションはコールできません。

参照：

- オブジェクトの本体および仕様部を削除する場合は、16-14 ページの「[DROP TYPE](#)」を参照してください。
- 型本体の詳細は、15-24 ページの「[CREATE TYPE BODY](#)」を参照してください。

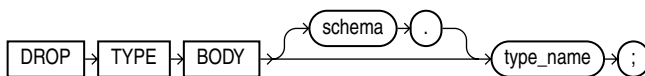
前提条件

オブジェクト型の本体が自分のスキーマ内にある必要があります。また、次の権限が必要です。

- CREATE TYPE または CREATE ANY TYPE システム権限
- DROP ANY TYPE システム権限

構文

drop_type_body::=



キーワードとパラメータ

schema

オブジェクト型が含まれているスキーマを指定します。*schema* を指定しない場合、このオブジェクト型が自分のスキーマに存在するとみなされます。

type_name

削除するオブジェクト型の本体の名前を指定します。

制限事項：型依存性および表依存性がない場合にのみ型の本体を削除できます。

例

DROP TYPE BODY の例 次の文は、15-29 ページの「[型本体の更新例](#)」で作成したオブジェクト型の本体 `data_typ` を削除します。

```
DROP TYPE BODY data_typ;
```

DROP USER

用途

DROP USER 文を使用すると、データベース・ユーザーを削除できます。また、オプションでユーザーのオブジェクトを削除することもできます。

参照：

- ユーザーの作成については、15-30 ページの「[CREATE USER](#)」を参照してください。
- ユーザー定義の変更については、11-20 ページの「[ALTER USER](#)」を参照してください。

前提条件

DROP USER システム権限が必要です。

構文

drop_user::=



キーワードとパラメータ

user

削除するユーザーを指定します。CASCADE を指定しない場合、またはユーザーのオブジェクトを最初に明示的に削除しない場合、所有するスキーマにオブジェクトが含まれているユーザーは削除されません。

CASCADE

CASCADE は、ユーザーを削除する前に、そのユーザーのスキーマ内にあるすべてのオブジェクトを削除します。所有するスキーマにオブジェクトが含まれているユーザーを削除する場合は、必ずこの句を指定します。

- ユーザーのスキーマに表がある場合、表が削除され、その表の主キーまたは一意キーを参照している他のユーザーのスキーマ内にある表の参照整合性制約も自動的に削除されます。
- この句で表が削除される場合、その表の列で作成されたすべてのドメイン索引も削除され、適切な削除ルーチンが起動されます。

参照： これらのルーチンの詳細は、『Oracle9i Data Cartridge Developer's Guide』を参照してください。

- Oracle は、他のスキーマにある、削除された自分のスキーマ内のオブジェクトのビューまたはシノニム、および削除された自分の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、ファンクション、パッケージのオブジェクトを削除せず、無効にします。
- 削除されたユーザー・スキーマ内の表またはビューに対するマテリアライズド・ビューは削除されません。ただし、CASCADE を指定した場合、このようなマテリアライズド・ビューはリフレッシュできなくなります。
- ユーザーのスキーマにあるすべてのトリガーは削除されます。
- ユーザーが作成したロールは削除されません。

注意： Oracle では、ユーザーが所有するすべての型も FORCE で削除します。詳細は、16-15 ページの「[DROP TYPE](#)」にある [FORCE](#) キーワードを参照してください。

例

DROP USER の例 ユーザー `sidney` のスキーマ内にオブジェクトがない場合、次の文を発行すると、`sidney` を削除できます。

```
DROP USER sidney;
```

`sidney` のスキーマ内にオブジェクトがある場合は、次の文のように CASCADE 句を指定して、`sidney` とそのオブジェクトを削除する必要があります。

```
DROP USER sidney CASCADE;
```

DROP VIEW

用途

DROP VIEW 文を使用すると、データベースからビューまたはオブジェクト・ビューを削除できます。ビューを削除して再作成すると、ビューの定義を変更できます。

参照：

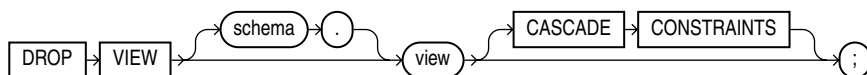
- ビューの作成については、15-37 ページの「[CREATE VIEW](#)」を参照してください。
- ビューの変更については、11-28 ページの「[ALTER VIEW](#)」を参照してください。

前提条件

削除するビューが自分のスキーマ内にあるか、または DROP ANY VIEW システム権限が必要です。

構文

drop_view::=



キーワードとパラメータ

schema

ビューを含むスキーマを指定します。*schema* を指定しない場合、ビューは自分のスキーマ内にあるとみなされます。

view

削除するビューの名前を指定します。

ビューを参照するビュー、マテリアライズド・ビューおよびシノニムは削除されませんが、INVALID のマークが付けられます。このようなビューおよびシノニムは削除または再定義するか、無効なビューやシノニムをもう一度有効にするビューを別に定義します。

ビューにサブビューが定義してある場合、サブビューも同様に無効になります。ビューがサブビューを持つかどうかを確認するには、`USER_VIEWS`、`ALL_VIEWS` または `DBA_VIEWS` データ・ディクショナリ・ビューの `SUPERVIEW_NAME` 列を問い合わせます。

参照：

- 14-6 ページの「[CREATE TABLE](#)」および 14-2 ページの「[CREATE SYNONYM](#)」を参照してください。
- 無効なマテリアライズド・ビューの再検証の詳細は、8-74 ページの「[ALTER MATERIALIZED VIEW](#)」を参照してください。

CASCADE CONSTRAINTS

CASCADE CONSTRAINTS は、削除するビューの主キーまたは一意キーを参照するすべての参照整合性制約を削除します。このような制約が存在する状態でこの句を省略すると、DROP 文は正常に実行されません。

例

DROP VIEW の例 次の文は、15-46 ページの「[基本ビューの例](#)」で作成した `emp_view` ビューを削除します。

```
DROP VIEW emp_view;
```

EXPLAIN PLAN

用途

EXPLAIN PLAN 文を使用すると、指定した SQL 文を実行するために Oracle が使用する実行計画を決定できます。この文によって、実行計画の各ステップを記述している行が、指定した表に挿入されます。コストベースの最適化を使用している場合、この文によって文を実行するコストも決まります。表にドメイン索引が定義されている場合、ユーザー定義の CPU および I/O コストが挿入されます。

配布メディアの SQL スクリプトの中に、サンプル出力表 PLAN_TABLE の定義があります。使用する出力表は、列の名前およびデータ型がこのサンプル表と同じである必要があります。このスクリプトの一般的な名前は、UTLXPLAN.SQL です。正確な名前および位置は、使用するオペレーティング・システムによって異なります。

SQL トレース機能の一部として EXPLAIN PLAN 文を発行することもできます。

キャッシュされたカーソルの実行計画の詳細は、V\$SQL_PLAN 動的パフォーマンス・ビューを問い合わせると確認できます。

参照：

- 実行計画の出力、SQL トレース機能の使用方法、および実行計画の生成と解析方法については、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。
- V\$SQL_PLAN を含む動的パフォーマンス・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

前提条件

EXPLAIN PLAN 文を実行する場合、実行計画を格納する既存の出力表に行を挿入するための権限が必要です。

出力する実行計画の適用対象である SQL 文を実行するための権限も必要です。この SQL 文でビューにアクセスする場合は、このビューの基礎になっているすべての表およびビューへのアクセス権限が必要です。このビューが別のビューに基づき、さらにこの別のビューが、ある表に基づいている場合、別のビューとそのビューの基礎になっている表へのアクセス権限が必要です。

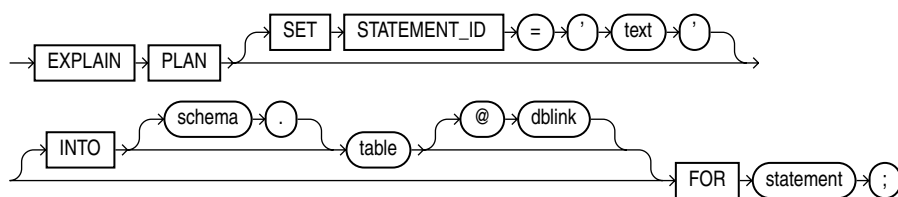
EXPLAIN PLAN で作成された実行計画を検証する場合は、出力表へ問い合わせる権限が必要です。

EXPLAIN PLAN 文はデータ操作言語 (DML) 文であり、データ定義言語 (DDL) 文ではありません。そのため、EXPLAIN PLAN 文で加えられた変更内容は暗黙的にコミットされません。出力表の EXPLAIN PLAN 文で生成された行を保存する場合は、この文を指定したトランザクションをコミットする必要があります。

参照： 計画表の移入および問合せに必要な権限の詳細は、16-56 ページの「[INSERT](#)」および 17-4 ページの「[SELECT](#)」を参照してください。

構文

explain_plan::=



キーワードとパラメータ

SET STATEMENT_ID 句

実行計画が出力される表の中で、この実行計画に該当する行にある STATEMENT_ID 列の値を指定します。この値によって、実行計画の行を出力表の中の他の行と区別できます。ユーザーの出力表に多数の実行計画の行が含まれている場合は、必ず、STATEMENT_ID の値を指定します。この句を省略した場合、デフォルトで STATEMENT_ID 値が NULL に設定されます。

INTO table 句

出力表の名前を指定し、オプションとしてそのスキーマおよびデータベースの名前も指定します。この表は、EXPLAIN PLAN 文を使用する前に作成しておく必要があります。

schema を指定しない場合、この表が自分のスキーマに存在するとみなされます。

dblink には、出力表が格納されているリモートの Oracle データベースに対するデータベース・リンクの完全な名前または名前の一部を指定します。Oracle の分散機能を使用している場合にのみ、リモート出力表を指定できます。*dblink* の指定を省略した場合、表がローカル・データベース上にあるとみなされます。

参照： データベース・リンクの参照方法の詳細は、2-113 ページの「[リモート・データベース内のオブジェクトの参照](#)」を参照してください。

INTO 句を省略した場合、出力表は、ローカル・データベース上の自分のスキーマ内にある PLAN_TABLE であるとみなされます。

FOR statement 句

実行計画生成の対象となる SELECT、INSERT、UPDATE、DELETE、CREATE TABLE、CREATE INDEX または ALTER INDEX ... REBUILD 文を指定します。

注意：

- *statement* が *parallel_clause* を持つ場合、結果として生成される実行計画はパラレルで実行されます。ただし、EXPLAIN PLAN コマンドによって計画表に実際に文が挿入されるため、ユーザーが発行したパラレル DML 文は、トランザクションの最初の DML 文ではなくなります。これは、トランザクション 1 つにつきパラレル DML 文は 1 つという Oracle の制限に違反するため、この文はシリアルで実行されます。文をパラレルで実行するには、EXPLAIN PLAN 文をコミットまたはロールバックしてから、パラレル DML 文を発行する必要があります。
 - 一時表上で操作するための実行計画を判断する場合、EXPLAIN PLAN は、同一セッションから実行する必要があります。これは、一時表内のデータがセッション固有であるためです。
-
-

例

EXPLAIN PLAN の例 次の文は、UPDATE 文の実行計画およびコストを決定し、実行計画を記述した行を STATEMENT_ID 値 'Raise in Tokyo' とともに、指定した plan_table 表に挿入します。

```
EXPLAIN PLAN
  SET STATEMENT_ID = 'Raise in Tokyo'
  INTO plan_table
  FOR UPDATE employees
    SET salary = salary * 1.10
    WHERE department_id =
      (SELECT department_id FROM departments
       WHERE location_id = 1200);
```

次の SELECT 文は、plan_table 表への問合せを実行し、実行計画およびコストを戻します。

```
SELECT LPAD(' ',2*(LEVEL-1))||operation operation, options,
object_name, position
FROM plan_table
START WITH id = 0 AND statement_id = 'Raise in Tokyo'
CONNECT BY PRIOR id = parent_id AND
statement_id = 'Raise in Tokyo';
```

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			
UPDATE		EMPLOYEES	
INDEX	RANGE SCAN	EMP_DEPARTMENT_IX	
TABLE ACCESS	BY INDEX ROWID	DEPARTMENTS	
INDEX	RANGE SCAN	DEPT_LOCATION_IX	

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 のパーティション化例 stock_num 列に従って 8 つにパーティション化されている stocks 表があり、stock_num 列上にローカルの同一キー索引 stock_ix が作成されているとします。パーティションの上限は、1000、2000、3000、4000、5000、6000、7000 および 8000 です。

次の問合せを考えてみます。

```
SELECT * FROM stocks WHERE stock_num BETWEEN 3800 AND :h;
```

ここで、:h はバインド変数を指します。EXPLAIN PLAN は、plan_table を出力表とする次の問合せを実行します。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;
```

filespec

用途

filespec 構文を使用すると、ファイルをデータ・ファイルまたはテンポラリ・ファイルとして指定できます、また、1 つ以上のファイルのグループを REDO ログ・ファイル・グループとして指定することもできます。

前提条件

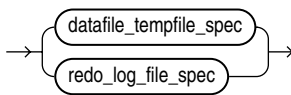
filespec は、CREATE DATABASE、ALTER DATABASE、CREATE TABLESPACE、ALTER TABLESPACE、CREATE CONTROLFILE、CREATE LIBRARY および CREATE TEMPORARY TABLESPACE 文で使用できます。これらの文の発行には、その文の実行権限が必要です。

参照： 詳細は、次の項を参照してください。

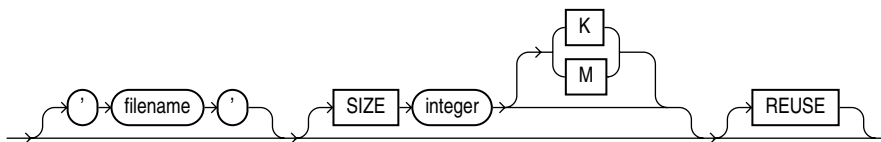
- 12-20 ページの「CREATE DATABASE」
- 8-9 ページの「ALTER DATABASE」
- 14-62 ページの「CREATE TABLESPACE」
- 10-82 ページの「ALTER TABLESPACE」
- 12-14 ページの「CREATE CONTROLFILE」
- 13-2 ページの「CREATE LIBRARY」
- 14-73 ページの「CREATE TEMPORARY TABLESPACE」

構文

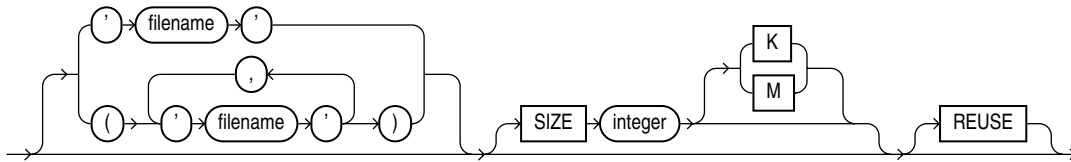
filespec::=



datafile_tempfile_spec::=



redo_log_file_spec::=



キーワードとパラメータ

'filename'

`filespec` が新しいファイルの場合、次のようになります。

- データ・ファイルまたはテンポラリ・ファイルの場合、初期化パラメータ `DB_CREATE_FILE_DEST` が設定されていると、`filename` はオプションで、`filespec` の残りの句となります。
- ログ・ファイルの場合、`DB_CREATE_ONLINE_LOG_DEST_n` または `DB_CREATE_FILE_DEST` パラメータが設定されていると、`filename` はオプションで、`filespec` の残りの句となります。

どちらの場合も、作成するファイル・タイプのデフォルトのシステム位置に **Oracle** 管理ファイルが作成され、一意のファイル名が生成されます。SIZE 句を省略すると、100MB の自動拡張可能ファイルが、最大サイズの指定なしで作成されます。filespec を含む操作が正常に実行されない場合、部分的に作成された **Oracle** 管理ファイルは削除されます。いずれのパラメータも設定されていない状態で filename を省略すると、操作は正常に実行されません。

`filespec` が既存のファイルの場合、ファイル名を指定する必要があります。データ・ファイル、テンポラリ・ファイルまたは REDO ログ・ファイル・メンバーの名前を指定します。`filename` には、7 ビット ASCII キャラクタ・セットまたは EBCDIC キャラクタ・セットのシングルスバイト文字のみを使用できます。マルチバイト文字は無効です。

REDO ログ・ファイル・グループは、1つ以上のメンバー（コピー）で構成されます。
filename は、使用するオペレーティング・システムの表記規則に従って、完全な名前
で指定する必要があります。

SIZE 句

ファイルのサイズをバイト単位で指定します。K または M を使用して、KB または MB 単位で指定することもできます。

- UNDO 表領域の場合、各データ・ファイルに対して SIZE 句を指定する必要があります。その他の表領域の場合、ファイルがすでに存在するとき、または Oracle 管理ファイルを作成するときには、このパラメータを省略できます。
- 表領域は、中に含まれるオブジェクトのサイズの合計より 1 ブロック大きいサイズである必要があります。

参照： 自動 UNDO 管理および UNDO 表領域の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。

REUSE

REUSE は、既存ファイルの再利用を可能にします。すでに存在している *filename* を指定する場合は、REUSE を指定する必要があります。

- ファイルがすでに存在する場合、ファイル名が再利用され、新しいサイズが適用されるか (SIZE を指定する場合)、または元のサイズのままとなります。
- ファイルが存在していない場合、この句は無視され、ファイルが作成されます。

制限事項： *filename* を指定しないかぎり、REUSE は指定できません。

注意： 既存のファイルが使用される場合は、そのファイルに入っていた内容は失われます。

例

ログ・ファイルの指定例 次の文は、payable という名前のデータベースを作成します。このデータベースには各グループにメンバーを 2 つ持つ REDO ログ・ファイル・グループが 2 つと、データ・ファイルが 1 つ設定されています。

```
CREATE DATABASE payable
  LOGFILE GROUP 1 ('diska:log1.log', 'diskb:log1.log') SIZE 50K,
  GROUP 2 ('diska:log2.log', 'diskb:log2.log') SIZE 50K
  DATAFILE 'diskc:dbone.dat' SIZE 30M;
```

LOGFILE 句の最初の *filespec* は、REDO ログ・ファイル・グループに GROUP1 の値を指定します。このグループには、'diska:log1.log' および 'diskb:log1.log' というメンバーがあり、サイズはそれぞれ 50KB です。

LOGFILE 句の 2 番目の *filespec* は、REDO ログ・ファイル・グループに GROUP2 の値を指定します。このグループには、'diska:log2.log' および 'diskb:log2.log' というメンバーがあり、サイズはそれぞれ 50KB です。

DATAFILE 句の *filespec* では、'diskc:dbone.dat' という名前のデータ・ファイルが指定されています。このファイルのサイズは 30MB です。

各 *filespec* は SIZE パラメータの値を指定し、REUSE 句は指定していないため、指定のファイルがすでに定義されていることはありません。Oracle が新しくファイルを作成します。

ログ・ファイルの追加例 次の文は、2 つのメンバーで構成される新しい REDO ログ・ファイル・グループを、payable データベースに追加します。

```
ALTER DATABASE payable
  ADD LOGFILE GROUP 3 ('diska:log3.log', 'diskb:log3.log')
  SIZE 50K REUSE;
```

ADD LOGFILE 句の *filespec* は、REDO ログ・ファイル・グループに GROUP3 の値を指定します。このグループには、'diska:log3.log' および 'diskb:log3.log' というメンバーがあり、サイズはそれぞれ 50KB です。この *filespec* では REUSE 句が指定されているため、各メンバーはすでに存在している可能性があります（存在しなくてもよい）。

データ・ファイルの指定例 次の文は、stocks という名前の表領域を作成します。また、この表領域にはデータ・ファイルが 3 つあります。

```
CREATE TABLESPACE stocks
  DATAFILE 'diskc:stock1.dat',
            'diskc:stock2.dat',
            'diskc:stock3.dat';
```

データ・ファイルの *filespecs* は、'diskc:stock1.dat'、'diskc:stock2.dat' および 'diskc:stock3.dat' という名前のファイルを指定します。各 *filespec* の指定に SIZE パラメータが指定されていないため、各ファイルはすでに存在している必要があります。

データ・ファイルの追加例 次の文は、stocks 表領域を変更し、新しいデータ・ファイルを追加します。

```
ALTER TABLESPACE stocks
  ADD DATAFILE 'diskc:stock4.dat' REUSE;
```

filespec は、'diskc:stock4.dat' という名前のデータ・ファイルを指定します。*filespec* の指定に SIZE パラメータが指定されていないため、このデータ・ファイルはすでに存在している必要があります。また、REUSE 句は無効です。

GRANT

用途

GRANT 文を使用すると、次の権限を付与できます。

- ユーザーおよびロールに対するシステム権限
- ユーザーおよびロールに対するロール。権限とロールは、ともにローカル、グローバルまたは外部になります。表 16-1 に、システム権限のリストを、操作対象のデータベース・オブジェクト別に示します。表 16-2 に、Oracle で事前定義されているロールを示します。
- ユーザー、ロールおよび PUBLIC に対する、特定のオブジェクトに対するオブジェクト権限。表 16-3 に、各オブジェクトに付与できるオブジェクト権限を示します。表 16-4 に、オブジェクト権限とその権限によって許可される操作を示します。

注意： データベース・ユーザーにロールの使用を許可する方法は、データベースや GRANT 文を介して行う以外にもあります。たとえば、オペレーティング・システムによっては、Oracle ユーザーに対して、初期化パラメータ OS_ROLES でロールを付与する機能を備えている場合もあります。オペレーティング・システム機能を使用してユーザーにロールを付与する場合、GRANT 文を使用してユーザーにシステム権限を付与したり、その他のロールにシステム権限およびロールを付与することはできますが、そのユーザーにロールを付与することはできません。

参照：

- ローカル、グローバルおよび外部権限の定義については、15-30 ページの「CREATE USER」および 13-72 ページの「CREATE ROLE」を参照してください。
- その他の認証方法の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。
- 権限付与の取消しについては、16-87 ページの「REVOKE」を参照してください。

前提条件

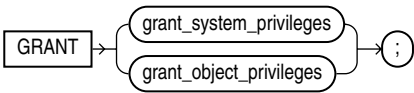
システム権限を付与する場合は、ADMIN OPTION 付きのシステム権限または GRANT ANY PRIVILEGE システム権限が付与されている必要があります。

ロールを付与する場合は、ADMIN OPTION 付きのロールまたは GRANT ANY ROLE システム権限が付与されているか、あるいは付与するロールが自分で作成したロールである必要があります。

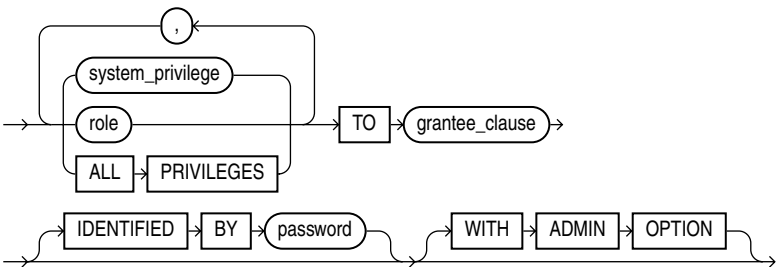
オブジェクト権限を付与する場合は、オブジェクトの所有者である必要があります。そうでない場合、オブジェクトの所有者から GRANT OPTION 付きのオブジェクト権限が付与されている必要があります。この規則は、DBA ロールを持つユーザーに適用されます。

構文

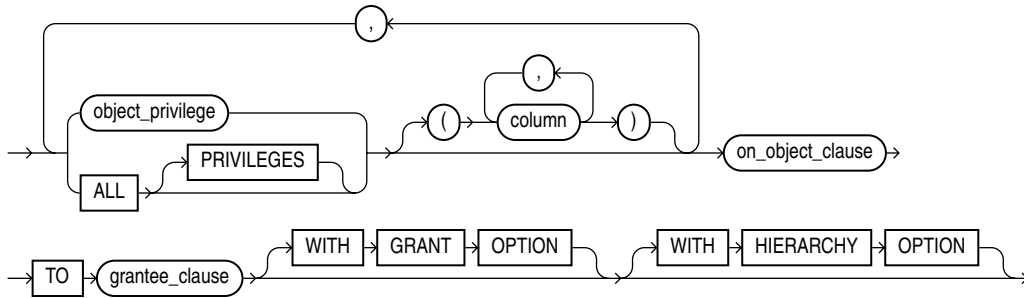
grant::=



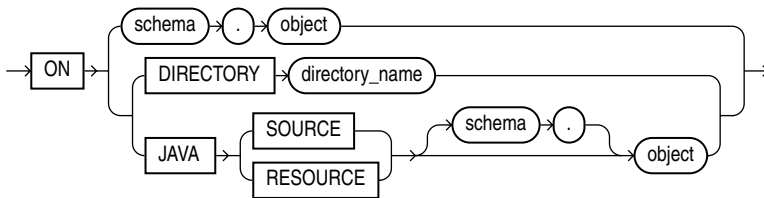
grant_system_privileges::=



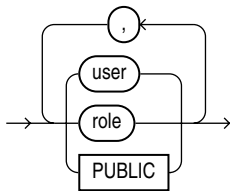
grant_object_privileges::=



on_object_clause::=



grantee_clause::=



キーワードとパラメータ

grant_system_privileges

system_privilege

付与するシステム権限を指定します。表 16-1 に、システム権限のリストを、操作対象のデータベース・オブジェクト別に示します。

- **ユーザー**に権限を付与する場合、ユーザーの権限ドメインに権限が登録されます。このユーザーはその権限をすぐに使用できます。
- **ロール**に権限を付与する場合、ロールの権限ドメインに権限が登録されます。ロールが付与されているまたは使用可能となっているユーザーは、その権限をすぐに使用できます。なお、ロールが付与されている他のユーザーも、そのロールを使用可能にしてその権限を使用できます。
- **PUBLIC** に権限を付与する場合、各ユーザーの権限ドメインに権限が登録されます。すべてのユーザーは、その権限によって許可される操作をすぐに実行できます。

すべてのシステム権限を一度に指定できるショートカットがあります。

- **ALL PRIVILEGES**。ALL PRIVILEGES を指定すると、SELECT ANY DICTIONARY 権限を除き、16-38 ページの表 16-1「システム権限」に示すすべてのシステム権限を付与できます。

role

付与するロールを指定します。事前に定義されたロールまたはユーザーが定義したロールを付与できます。表 16-2 は、事前に定義されたロールのリストです。

- **ユーザー**にロールを付与する場合、ユーザーに付与されたロールを使用できます。ユーザーはそのロールをすぐに使用可能にし、ロールの権限ドメインに登録されている権限を使用できます。
- 他の**ロール**にロールを付与する場合、権限受領者のロールの権限ドメインに、権限付与者のロールの権限ドメインが登録されます。権限受領者のロールを付与されているユーザーは、それを使用可能にした場合、付与されたロールの権限ドメイン内の権限を使用できます。
- **PUBLIC** にロールを付与する場合、すべてのユーザーがロールを使用できます。ユーザーはそのロールを使用可能にし、ロールの権限ドメインに登録されている権限をすぐに使用できます。

参照： ユーザー定義ロールの作成については、13-72 ページの「**CREATE ROLE**」を参照してください。

IDENTIFIED BY 句

IDENTIFIED BY 句を使用すると、パスワードによって既存ユーザーが明確に識別できるか、または存在しないユーザーを作成できます。この句は、権限受領者がロールまたは PUBLIC の場合は無効です。grantee_clause に指定するユーザーが存在しない場合、パスワードおよびこの句で指定する権限とロール付きでユーザーが作成されます。

参照： ユーザー名およびパスワードの制限事項については、15-30 ページの「CREATE USER」を参照してください。

WITH ADMIN OPTION

WITH ADMIN OPTION を指定すると、権限受領者は次のことができます。

- ロールが GLOBAL ロールでない場合、そのロールを他のユーザーまたはロールに付与できます。
- ロールを他のユーザーまたは他のロールから取り消すことができます。
- ロールへのアクセスに必要な認証を変更するため、ロールを変更できます。
- ロールを削除できます。

たとえば、WITH ADMIN OPTION を指定せずにユーザーにロールまたはシステム権限を付与し、後で WITH ADMIN OPTION を指定してその権限およびロールを付与した場合、そのユーザーはその権限またはロールに関して ADMIN OPTION を持つことになります。

システム権限またはロールの ADMIN OPTION をユーザーから取り消す場合、そのユーザーの権限またはロールを完全に取り消し、その次に ADMIN OPTION を指定せずに権限またはロールをユーザーに付与します。

grantee_clause

TO grantee_clause は、システム権限、ロールまたはオブジェクト権限が付与されているユーザーまたはロールを識別します。

制限事項： TO grantee_clause にユーザー、ロールおよび PUBLIC を指定できるのは 1 回のみです。

PUBLIC すべてのユーザーに権限を付与する場合は、PUBLIC を指定します。

システム権限およびロールの付与に関する制限事項

- 付与する権限およびロールのリストに、権限またはロールを指定できるのは 1 回のみです。
- ロールに対してそのロールと同じロールは付与できません。
- ロール IDENTIFIED GLOBALLY は、どのようなユーザーまたはロールに対しても付与できません。

- ロール IDENTIFIED EXTERNALLY は、グローバル・ユーザーまたはグローバル・ロールに対して付与できません。
- ロールは交互に付与できません。たとえば、ロール `banker` をロール `teller` に付与した場合、逆に `teller` を `banker` に付与することはできません。

grant_object_privileges

object_privilege

付与するオブジェクト権限を指定します。表 16-3 に示すいずれかの値を指定します。表 16-4 も参照してください。

制限事項：付与する権限のリストに権限を指定できるのは 1 回のみです。

ALL [PRIVILEGES]

GRANT OPTION 付きで付与されているオブジェクト権限に対するすべての権限を付与する場合に、ALL を指定します。オブジェクトが定義されているスキーマを所有しているユーザーは、自動的に GRANT OPTION 付きのオブジェクトに関するすべての権限を持っています（キーワード PRIVILEGES はセマンティクスを明確にするためのものであり、指定は任意です。）

column

権限を付与する表またはビューの列を指定します。INSERT、REFERENCES または UPDATE の各権限を付与する場合にのみ、列を指定できます。列を指定しない場合、権限受領者には表またはビューのすべての列に対して指定した権限が付与されます。

既存の列オブジェクトに関する権限の付与については、データ・ディクショナリ・ビュー USER_COL_PRIVS、ALL_COL_PRIVS および DBA_COL_PRIVS に問い合わせます。

参照： データ・ディクショナリ・ビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

on_object_clause

on_object_clause は、権限が付与されているオブジェクトを識別します。ディレクトリ・スキーマ・オブジェクト、Java ソースおよびリソース・スキーマ・オブジェクトは、異なるネームスペースに格納されるため、別々に識別されます。

WITH GRANT OPTION

WITH GRANT OPTION は、権限受領者が、他のユーザーまたはロールに対してオブジェクト権限を付与する場合に指定します。

制限事項：WITH GRANT OPTION は、ユーザーまたは PUBLIC に権限を付与する場合にのみ指定できます。ロールに付与する場合は指定できません。

WITH HIERARCHY OPTION

WITH HIERARCHY OPTION を指定すると、この文の後に作成されるサブオブジェクトも含め、*object* のすべてのサブオブジェクト（ビューを基に作成したサブビューなど）に対する指定したオブジェクト権限を付与できます。

注意： この句は、SELECT オブジェクト権限とあわせて指定する場合のみ意味があります。

object 権限を付与するスキーマ・オブジェクトを指定します。*object* に *schema* を指定しない場合、自分のスキーマ内にあるとみなされます。オブジェクトには次のタイプがあります。

- 表、ビューまたはマテリアライズド・ビュー
- 順序
- プロシージャ、ファンクションまたはパッケージ
- ユーザー定義型
- これらの項目のシノニム
- ディレクトリ、ライブラリ、演算子または索引タイプ
- Java ソース、クラスまたはリソース

注意： パーティション表の単一パーティションに直接権限を付与することはできません。単一パーティションに間接的に権限を付与する方法の詳細は、『Oracle9i データベース概要』を参照してください。

DIRECTORY *directory_name* 権限を付与するディレクトリ・スキーマ・オブジェクトを指定します。*directory_name* にはスキーマ名を指定できません。

参照： 12-44 ページの「[CREATE DIRECTORY](#)」を参照してください。

JAVA SOURCE | RESOURCE JAVA 句によって、権限が付与された Java ソースまたはリソース・スキーマ・オブジェクトを指定します。

参照： 12-86 ページの「[CREATE JAVA](#)」を参照してください。

システム権限およびオブジェクト権限

表 16-1 システム権限

システム権限名	許可される操作
クラスタ	
CREATE CLUSTER	権限を付与したスキーマ内でのクラスタの作成
CREATE ANY CLUSTER	任意のスキーマ内でのクラスタの作成 CREATE ANY TABLE と同様に動作します。
ALTER ANY CLUSTER	任意のスキーマ内でのクラスタの変更
DROP ANY CLUSTER	任意のスキーマ内でのクラスタの削除
コンテキスト	
CREATE ANY CONTEXT	任意のコンテキスト・ネームスペースの作成
DROP ANY CONTEXT	任意のコンテキスト・ネームスペースの削除
データベース	
ALTER DATABASE	データベースの変更
ALTER SYSTEM	ALTER SYSTEM 文の発行
AUDIT SYSTEM	AUDIT <i>sql_statements</i> 文の発行
データベース・リンク	
CREATE DATABASE LINK	権限を付与したスキーマ内でのプライベート・データベース・リンクの作成
CREATE PUBLIC DATABASE LINK	パブリック・データベース・リンクの作成
DROP PUBLIC DATABASE LINK	パブリック・データベース・リンクの削除
ディメンション	
CREATE DIMENSION	権限を付与したスキーマ内でのディメンションの作成
CREATE ANY DIMENSION	任意のスキーマ内でのディメンションの作成
ALTER ANY DIMENSION	任意のスキーマ内でのディメンションの変更
DROP ANY DIMENSION	任意のスキーマ内でのディメンションの削除

注意: ANY オブジェクトの権限 (CREATE ANY CLUSTER など) を付与した場合、SYS スキーマを含めたすべてのスキーマ内で、そのタイプのオブジェクトにアクセスできるようになります。SYS スキーマ内のオブジェクトへのアクセスを禁止するには、初期化パラメータ O7_DICTIONARY_ACCESSIBILITY を FALSE に設定します。ANY オブジェクトに付与された権限によって、SYS 以外のスキーマにアクセスできるようになります。

表 16-1 システム権限（続き）

システム権限名	許可される操作
ディレクトリ	
CREATE ANY DIRECTORY	ディレクトリ・データベース・オブジェクトの作成
DROP ANY DIRECTORY	ディレクトリ・データベース・オブジェクトの削除
索引タイプ	
CREATE INDEXTYPE	権限を付与したスキーマ内での索引タイプの作成
CREATE ANY INDEXTYPE	任意のスキーマ内での索引タイプの作成
ALTER ANY INDEXTYPE	任意のスキーマ内での索引タイプの変更
DROP ANY INDEXTYPE	任意のスキーマ内での索引タイプの削除
EXECUTE ANY INDEXTYPE	任意のスキーマ内での索引タイプの参照
索引	
CREATE ANY INDEX	任意のスキーマでの任意の表へのドメイン索引または索引の作成
ALTER ANY INDEX	任意のスキーマでの索引の変更
DROP ANY INDEX	任意のスキーマでの索引の削除
QUERY REWRITE	マテリアライズド・ビューまたは索引が権限を付与したスキーマ内の表およびビューを参照している場合の次の操作: そのマテリアライズド・ビューを使用したリライト、ファンクション索引の作成
GLOBAL QUERY REWRITE	マテリアライズド・ビューまたは索引が任意のスキーマ内の表またはビューを参照している場合の次の操作: そのマテリアライズド・ビューを使用したリライト、ファンクション索引の作成
ライブラリ	
CREATE LIBRARY	権限を付与したスキーマ内での外部プロシージャまたはファンクション・ライブラリの作成
CREATE ANY LIBRARY	任意のスキーマ内での外部プロシージャまたはファンクション・ライブラリの作成
DROP LIBRARY	権限を付与したスキーマ内での外部プロシージャまたはファンクション・ライブラリの削除
DROP ANY LIBRARY	任意のスキーマ内での外部プロシージャまたはファンクション・ライブラリの削除

注意: ANY オブジェクトの権限 (CREATE ANY CLUSTER など) を付与した場合、SYS スキーマを含めたすべてのスキーマ内で、そのタイプのオブジェクトにアクセスできるようになります。SYS スキーマ内のオブジェクトへのアクセスを禁止するには、初期化パラメータ O7_DICTIONARY_ACCESSIBILITY を FALSE に設定します。ANY オブジェクトに付与された権限によって、SYS 以外のスキーマにアクセスできるようになります。

表 16-1 システム権限（続き）

システム権限名	許可される操作
マテリアライズド・ビュー	
CREATE MATERIALIZED VIEW	権限を付与したスキーマ内でのマテリアライズド・ビューの作成
CREATE ANY MATERIALIZED VIEW	任意のスキーマでのマテリアライズド・ビューの作成
ALTER ANY MATERIALIZED VIEW	任意のスキーマでのマテリアライズド・ビューの変更
DROP ANY MATERIALIZED VIEW	任意のスキーマでのマテリアライズド・ビューの削除
QUERY REWRITE	マテリアライズド・ビューまたは索引が権限を付与したスキーマ内の表およびビューを参照している場合の次の操作: そのマテリアライズド・ビューを使用したリライト、ファンクション索引の作成
GLOBAL QUERY REWRITE	マテリアライズド・ビューまたは索引が任意のスキーマ内の表またはビューを参照している場合の次の操作: そのマテリアライズド・ビューを使用したリライト、ファンクション索引の作成
ON COMMIT REFRESH	データベースの任意の表に対する REFRESH ON COMMIT モードのマテリアライズド・ビューの作成 データベースの任意の表に対する REFRESH ON DEMMAND モードのマテリアライズド・ビューの、REFRESH ON COMMIT モードのマテリアライズド・ビューへの変更
演算子	
CREATE OPERATOR	権限を付与したスキーマ内での演算子およびバインディングの作成
CREATE ANY OPERATOR	任意のスキーマ内での演算子およびバインディングの作成
DROP ANY OPERATOR	任意のスキーマ内での演算子の削除
EXECUTE ANY OPERATOR	任意のスキーマ内での演算子の参照

注意: ANY オブジェクトの権限 (CREATE ANY CLUSTER など) を付与した場合、SYS スキーマを含めたすべてのスキーマ内で、そのタイプのオブジェクトにアクセスできるようになります。SYS スキーマ内のオブジェクトへのアクセスを禁止するには、初期化パラメータ O7_DICTIONARY_ACCESSIBILITY を FALSE に設定します。ANY オブジェクトに付与された権限によって、SYS 以外のスキーマにアクセスできるようになります。

表 16-1 システム権限（続き）

システム権限名	許可される操作
アウトライン	
CREATE ANY OUTLINE	アウトラインを使用する任意のスキーマ内で使用するパブリック・アウトラインの作成
ALTER ANY OUTLINE	アウトラインの変更
DROP ANY OUTLINE	アウトラインの削除
SELECT ANY OUTLINE	パブリック・アウトラインからのプライベート・クローン・アウトラインの作成
プロシージャ	
CREATE PROCEDURE	権限を付与したスキーマ内でのストアド・プロシージャ、ストアド・ファンクション、ストアド・パッケージの作成
CREATE ANY PROCEDURE	任意のスキーマ内でのストアド・プロシージャ、ストアド・ファンクション、ストアド・パッケージの作成
ALTER ANY PROCEDURE	任意のスキーマ内でのストアド・プロシージャ、ストアド・ファンクション、ストアド・パッケージの変更
DROP ANY PROCEDURE	任意のスキーマ内でのストアド・プロシージャ、ストアド・ファンクション、ストアド・パッケージの削除
EXECUTE ANY PROCEDURE	プロシージャまたはファンクションの実行（スタンドアロンまたはパッケージ） 任意のスキーマ内でのパブリック・パッケージ変数の参照
プロファイル	
CREATE PROFILE	プロファイルの作成
ALTER PROFILE	プロファイルの変更
DROP PROFILE	プロファイルの削除
ロール	
CREATE ROLE	ロールの作成
ALTER ANY ROLE	データベース内の任意のロールの変更
DROP ANY ROLE	ロールの削除
GRANT ANY ROLE	データベース内の任意のロールの付与

注意：ANY オブジェクトの権限（CREATE ANY CLUSTER など）を付与した場合、SYS スキーマを含めたすべてのスキーマ内で、そのタイプのオブジェクトにアクセスできるようになります。SYS スキーマ内のオブジェクトへのアクセスを禁止するには、初期化パラメータ `O7_DICTIONARY_ACCESSIBILITY` を `FALSE` に設定します。ANY オブジェクトに付与された権限によって、SYS 以外のスキーマにアクセスできるようになります。

表 16-1 システム権限（続き）

システム権限名	許可される操作
ロールバック・セグメント	
CREATE ROLLBACK SEGMENT	ロールバック・セグメントの作成
ALTER ROLLBACK SEGMENT	ロールバック・セグメントの変更
DROP ROLLBACK SEGMENT	ロールバック・セグメントの削除
順序	
CREATE SEQUENCE	権限を付与したスキーマ内での順序の作成
CREATE ANY SEQUENCE	任意のスキーマ内での順序の作成
ALTER ANY SEQUENCE	データベース内の任意の順序の変更
DROP ANY SEQUENCE	任意のスキーマ内での順序の削除
SELECT ANY SEQUENCE	任意のスキーマ内での順序の参照
セッション	
CREATE SESSION	データベースへの接続
ALTER RESOURCE COST	セッション・リソースに対するコストの設定
ALTER SESSION	ALTER SESSION 文の発行
RESTRICTED SESSION	SQL*Plus の STARTUP RESTRICT 文を使用した、インスタンス起動後のログイン
スナップショット「マテリアライズド・ビュー」を参照。	
シノニム	
CREATE SYNONYM	権限を付与したスキーマ内でのシノニムの作成
CREATE ANY SYNONYM	任意のスキーマ内でのプライベート・シノニムの作成
CREATE PUBLIC SYNONYM	パブリック・シノニムの作成
DROP ANY SYNONYM	任意のスキーマ内でのプライベート・シノニムの削除
DROP PUBLIC SYNONYM	パブリック・シノニムの削除

注意: ANY オブジェクトの権限 (CREATE ANY CLUSTER など) を付与した場合、SYS スキーマを含めたすべてのスキーマ内で、そのタイプのオブジェクトにアクセスできるようになります。SYS スキーマ内のオブジェクトへのアクセスを禁止するには、初期化パラメータ O7_DICTIONARY_ACCESSIBILITY を FALSE に設定します。ANY オブジェクトに付与された権限によって、SYS 以外のスキーマにアクセスできるようになります。

表 16-1 システム権限（続き）

システム権限名	許可される操作
表	
注意： 外部表の場合、有効なシステム権限は、CREATE ANY TABLE、ALTER ANY TABLE、DROP ANY TABLE および SELECT ANY TABLE です。	
CREATE TABLE	権限を付与したスキーマ内での表の作成
CREATE ANY TABLE	任意のスキーマ内での表の作成 なお、表が設定されるスキーマの所有者は、表領域内にその表を定義するための割当て制限が必要です。
ALTER ANY TABLE	スキーマ内の任意の表またはビューの変更
BACKUP ANY TABLE	エクスポート・ユーティリティを使用した他のユーザーのスキーマからのオブジェクトの増分エクスポート
DELETE ANY TABLE	任意のスキーマ内での表、表パーティション、またはビューからの行の削除
DROP ANY TABLE	任意のスキーマ内での表または表パーティションの削除または切捨て
INSERT ANY TABLE	任意のスキーマ内の表またはビューへの行の挿入
LOCK ANY TABLE	任意のスキーマ内の表またはビューのロック
SELECT ANY TABLE	任意のスキーマ内の表、ビューまたはマテリアライズド・ビューの間合せ
UPDATE ANY TABLE	任意のスキーマ内の表またはビューの行の更新
表領域	
CREATE TABLESPACE	表領域の作成
ALTER TABLESPACE	表領域の変更
DROP TABLESPACE	表領域の削除
MANAGE TABLESPACE	表領域のオンラインとオフラインの切替え、表領域のバックアップの開始および終了の制御
UNLIMITED TABLESPACE	任意の表領域の無制限な使用 この権限は、設定されている任意の割当て制限をオーバーライドします。ユーザーからこの権限を取り消した場合、ユーザーのスキーマ・オブジェクトはそのまま残りますが、表領域の割当て制限が許可されないかぎり、それ以上表領域を割り当てることはできません。このシステム権限をロールに付与することはできません。

注意：ANY オブジェクトの権限（CREATE ANY CLUSTER など）を付与した場合、SYS スキーマを含めたすべてのスキーマ内で、そのタイプのオブジェクトにアクセスできるようになります。SYS スキーマ内のオブジェクトへのアクセスを禁止するには、初期化パラメータ O7_DICTIONARY_ACCESSIBILITY を FALSE に設定します。ANY オブジェクトに付与された権限によって、SYS 以外のスキーマにアクセスできるようになります。

表 16-1 システム権限（続き）

システム権限名	許可される操作
トリガー	
CREATE TRIGGER	権限を付与したスキーマ内でのデータベース・トリガーの作成
CREATE ANY TRIGGER	任意のスキーマ内でのデータベース・トリガーの作成
ALTER ANY TRIGGER	任意のスキーマ内でのデータベース・トリガーの使用可能化、使用禁止化またはコンパイル
DROP ANY TRIGGER	任意のスキーマ内でのデータベース・トリガーの削除
ADMINISTER DATABASE TRIGGER	データベース内でのトリガーの作成（CREATE TRIGGER または CREATE ANY TRIGGER 権限が必要）
型	
CREATE TYPE	権限を付与したスキーマ内でのオブジェクト型およびオブジェクト型本体の作成
CREATE ANY TYPE	任意のスキーマ内でのオブジェクト型およびオブジェクト型本体の作成
ALTER ANY TYPE	任意のスキーマ内でのオブジェクト型の変更
DROP ANY TYPE	任意のスキーマ内でのオブジェクト型およびオブジェクト型本体の削除
EXECUTE ANY TYPE	特定のユーザーに付与した場合、任意のスキーマ内でのオブジェクト型およびコレクション型を使用および参照した、任意のスキーマ内のオブジェクト型メソッドの起動 EXECUTE ANY TYPE をロールに付与した場合、使用可能なロールを保持したユーザーは、任意のスキーマ内のオブジェクト型メソッドを起動できません。
UNDER ANY TYPE	非最終オブジェクト型のサブタイプの作成

注意：ANY オブジェクトの権限（CREATE ANY CLUSTER など）を付与した場合、SYS スキーマを含めたすべてのスキーマ内で、そのタイプのオブジェクトにアクセスできるようになります。SYS スキーマ内のオブジェクトへのアクセスを禁止するには、初期化パラメータ O7_DICTIONARY_ACCESSIBILITY を FALSE に設定します。ANY オブジェクトに付与された権限によって、SYS 以外のスキーマにアクセスできるようになります。

表 16-1 システム権限（続き）

システム権限名	許可される操作
ユーザー	
CREATE USER	<p>ユーザーの作成</p> <p>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"> ■ 任意の表領域に対する割当て制限の設定 ■ デフォルトの表領域および一時表領域の設定 ■ CREATE USER 文の一部としてのプロファイルの設定
ALTER USER	<p>任意のユーザーの変更</p> <p>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"> ■ 他のユーザーのパスワードまたは認証方法の変更 ■ 任意の表領域に対する割当て制限の設定 ■ デフォルトの表領域および一時表領域の設定 ■ プロファイルおよびデフォルト・ロールの設定
BECOME USER	別のユーザーになること（全データベース・インポートを実行するユーザーに必要）
DROP USER	ユーザーの削除
ビュー	
CREATE VIEW	権限を付与したスキーマ内でのビューの作成
CREATE ANY VIEW	任意のスキーマ内でのビューの作成
DROP ANY VIEW	任意のスキーマ内でのビューの削除
UNDER ANY VIEW	オブジェクト・ビューのサブビューの作成

注意：ANY オブジェクトの権限（CREATE ANY CLUSTER など）を付与した場合、SYS スキーマを含めたすべてのスキーマ内で、そのタイプのオブジェクトにアクセスできるようになります。SYS スキーマ内のオブジェクトへのアクセスを禁止するには、初期化パラメータ O7_DICTIONARY_ACCESSIBILITY を FALSE に設定します。ANY オブジェクトに付与された権限によって、SYS 以外のスキーマにアクセスできるようになります。

表 16-1 システム権限（続き）

システム権限名	許可される操作
その他	
ANALYZE ANY	任意のスキーマ内の任意の表、クラスタ、索引の分析
AUDIT ANY	AUDIT <i>schema_objects</i> 文を使用した、任意のスキーマ内の任意のオブジェクトの監査
COMMENT ANY TABLE	任意のスキーマ内の任意の表、ビュー、列についてのコメントの記述
EXEMPT ACCESS POLICY	ファイングレイン・アクセス・コントロールの回避 注意： 強力なシステム権限で、権限受領者がアプリケーション駆動のセキュリティ・ポリシーを回避できます。データベース管理者がこの権限を付与する場合は、注意が必要です。
FORCE ANY TRANSACTION	ローカル・データベース内の、インダウト分散トランザクションのコミットまたはロールバック 分散トランザクション・エラーを意図的に発生させます。
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
RESUMABLE	再開可能な領域割当ての使用可能化
SELECT ANY DICTIONARY	SYS スキーマ内のデータ・ディクショナリ・オブジェクトへの問合せ 初期化パラメータ O7_DICTIONARY_ACCESSIBILITY のデフォルトの FALSE 設定を選択的にオーバーライドします。 注意： この権限は個々に付与する必要があります。GRANT ALL PRIVILEGES には含まれません。また、ロールを介して付与できません。

注意：ANY オブジェクトの権限（CREATE ANY CLUSTER など）を付与した場合、SYS スキーマを含めたすべてのスキーマ内で、そのタイプのオブジェクトにアクセスできるようになります。SYS スキーマ内のオブジェクトへのアクセスを禁止するには、初期化パラメータ O7_DICTIONARY_ACCESSIBILITY を FALSE に設定します。ANY オブジェクトに付与された権限によって、SYS 以外のスキーマにアクセスできるようになります。

表 16-1 システム権限（続き）

システム権限名	許可される操作
SYSDBA	STARTUP および SHUTDOWN 操作の実行
	ALTER DATABASE: オープン、マウント、バックアップまたはキャラクタ・セットの変更
	CREATE DATABASE
	ARCHIVELOG および RECOVERY
	CREATE SPFILE
	RESTRICTED SESSION 権限を含みます。
SYSOPER	STARTUP および SHUTDOWN 操作の実行
	ALTER DATABASE OPEN MOUNT BACKUP
	ARCHIVELOG および RECOVERY
	CREATE SPFILE
	RESTRICTED SESSION 権限を含みます。
注意： ANY オブジェクトの権限（CREATE ANY CLUSTER など）を付与した場合、SYS スキーマを含めたすべてのスキーマ内で、そのタイプのオブジェクトにアクセスできるようになります。SYS スキーマ内のオブジェクトへのアクセスを禁止するには、初期化パラメータ O7_DICTIONARY_ACCESSIBILITY を FALSE に設定します。ANY オブジェクトに付与された権限によって、SYS 以外のスキーマにアクセスできるようになります。	

表 16-2 Oracle によって定義されたロール

定義されたロール	用途
CONNECT、RESOURCE および DBA	<p>前回のリリースとの互換性を確保します。DBA_SYS_PRIVS データ・ディクショナリ・ビューを問い合わせることによって、これらのロールにまとめられた権限を確認できます。</p> <p>参照：このビューの詳細は、『Oracle9i データベース・リファレンス』を参照してください。</p> <p>注意：データベースのセキュリティを維持するために、独自のロールを設計することをお薦めします。これらのロールは、今後の Oracle のリリースでは自動的に作成されない可能性があります。</p>
DELETE_CATALOG_ROLE EXECUTE_CATALOG_ROLE SELECT_CATALOG_ROLE	<p>データ・ディクショナリ・ビューおよびパッケージにアクセスします。</p> <p>参照：これらのロールの詳細は、『Oracle9i データベース管理者ガイド』を参照してください。</p>
EXP_FULL_DATABASE IMP_FULL_DATABASE	<p>インポート / エクスポート・ユーティリティに有効です。</p> <p>参照：これらのロールの詳細は、『Oracle9i データベース・ユーティリティ』を参照してください。</p>
AQ_USER_ROLE AQ_ADMINISTRATOR_ROLE	<p>Oracle のアドバンスト・キューイング機能を使用する場合、これらのロールが必要です。</p> <p>参照：これらのロールの詳細は、『Oracle9i アプリケーション開発者ガイド - アドバンスト・キューイング』を参照してください。</p>
SNMPAGENT	<p>このロールは、Enterprise Manager/Intelligent Agent で使用します。</p> <p>参照：Enterprise Manager/Intelligent Agent の詳細は、『Oracle Enterprise Manager 管理者ガイド』を参照してください。</p>
RECOVERY_CATALOG_OWNER	<p>リカバリ・カタログを所有するユーザーを作成する場合、このロールが必要です。</p> <p>参照：リカバリ・カタログの詳細は、『Oracle9i ユーザー管理バックアップおよびリカバリ・ガイド』を参照してください。</p>
HS_ADMIN_ROLE	<p>Oracle の異種機間サービス機能を使用する DBA がデータ・ディクショナリにある適切な表にアクセスし、その表を DBMS_HS パッケージと一緒に操作する場合、このロールが必要です。</p> <p>参照：詳細は、『Oracle9i PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。</p>

表 16-3 特定のオブジェクトで使用可能なオブジェクト権限

オブジェクト 表 権限		ビュー	順序	プロシージャ、 ファンクション、 パッケージ ^a	マテリア ライズド・ ビュー	ディレ クトリ	ライブ ラリ	ユーザ― 定義型	演算子	索引 タイプ
ALTER	X		X							
DELETE	X	X			X ^b					
EXECUTE				X			X	X	X	X
INDEX	X									
INSERT	X	X			X ^b					
ON COMMIT REFRESH	X									
QUERY REWRITE	X									
READ						X				
REFERENCES	X	X								
SELECT	X	X	X		X					
UNDER		X						X		
UPDATE	X	X			X ^b					
WRITE						X				

^a Oracle では、Java クラス、ソースまたはリソースを、オブジェクト権限の付与のためのプロシージャとして扱います。

^b DELETE、INSERT および UPDATE 権限は、更新可能なマテリアライズド・ビューにのみ付与できます。

表 16-4 オブジェクト権限とその権限によって許可される操作

オブジェクト権限	許可される操作
次の表権限は、表の操作を許可します。次のいずれかのオブジェクト権限がある場合は、LOCK TABLE 文を使用して任意のロック・モードで表をロックできます。	
ALTER	ALTER TABLE 文での表定義の変更
DELETE	DELETE 文での表の行の削除 注意： 表に対する DELETE 権限とともに SELECT 権限を付与する必要があります。
INDEX	CREATE INDEX 文での表の索引の作成
INSERT	INSERT 文での表への新しい行の追加
REFERENCES	表参照制約の作成 この権限はロールには付与できません。
SELECT	SELECT 文での表の間合せ
UPDATE	UPDATE 文での表のデータの変更 注意： 表に対する UPDATE 権限とともに SELECT 権限を付与する必要があります。
次のビュー権限は、ビューの操作を許可します。次のいずれかのオブジェクト権限がある場合は、LOCK TABLE 文を使用して任意のロック・モードでビューをロックできます。	
ビューの権限を付与する場合、そのビューのすべての実表に関して GRANT OPTION 付きの権限が必要です。	
DELETE	DELETE 文でのビューの行の削除
INSERT	INSERT 文でのビューへの新しい行の追加
REFERENCES	ビューへの外部キー制約の定義
SELECT	SELECT 文でのビューの間合せ
UNDER	ビューのサブビューの作成 このビューの直属のスーパービューに WITH GRANT OPTION 付きの UNDER ANY VIEW 権限を持つ場合のみ、このオブジェクト権限を付与できます。
UPDATE	UPDATE 文でのビューのデータの変更
次の順序権限は、順序の操作を許可します。	
ALTER	ALTER SEQUENCE 文での順序定義の変更
SELECT	CURRVAL 疑似列および NEXTVAL 疑似列を使用した順序の値の検査および増分

表 16-4 オブジェクト権限とその権限によって許可される操作（続き）

オブジェクト権限	許可される操作
<p>プロシージャ、ファンクションおよびパッケージ権限は、プロシージャ、ファンクションおよびパッケージの操作を許可します。この権限は、Java ソース、クラスおよびリソースにも適用されます。Oracle では、これらはオブジェクト権限の付与のために生成されたプロシージャとして扱われます。</p>	
EXECUTE	<p>プロシージャまたはファンクションのコンパイルまたは直接実行、またはパッケージの仕様部に宣言されたプログラム・オブジェクトへのアクセス</p> <p>注意：プロシージャ、ファンクションまたはパッケージを間接的に実行する場合、ユーザーはこの権限を持つ必要はありません。</p> <p>参照：『Oracle9i データベース概要』および『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。</p>
<p>次のマテリアライズド・ビュー権限は、マテリアライズド・ビューについての操作を許可します。</p>	
ON COMMIT REFRESH	指定した表に対する REFRESH ON COMMIT モードのマテリアライズド・ビューの作成
QUERY REWRITE	指定した表で使用するクエリー・リライトに対するマテリアライズド・ビューの作成
SELECT	SELECT 文でのマテリアライズド・ビューの間合せ
<p>シノニム権限は、基本オブジェクトに対して付与される権限と同じです。シノニムに権限を付与することは、基本オブジェクトに権限を付与することと同じです。同様に、基本オブジェクトに対して権限を付与することは、オブジェクトのすべてのシノニムに権限を付与することと同じです。あるユーザーにシノニムの権限を付与した場合、そのユーザーは、シノニム名または基本オブジェクト名を SQL 文に指定して、その権限を使用できます。</p>	
<p>次のディレクトリ権限では、ディレクトリ・オブジェクトをポインタとして使用することにより、オペレーティング・システムのディレクトリに格納されている各ファイルにデータベースから安全にアクセスできるようになります。このディレクトリ・オブジェクトには、ファイルが格納されているオペレーティング・システムのディレクトリへのフルパス名が定義されています。これらのファイルは実際にはデータベース外に格納されているため、Oracle サーバーの各プロセスは、ファイル・システム・サーバーに対して適切なファイル・アクセス権限も持っている必要があります。オペレーティング・システムに対するオブジェクト権限ではなく、ディレクトリ・データベース・オブジェクトに対するオブジェクト権限を個々のデータベース・ユーザーに付与することによって、Oracle はファイル運用時のセキュリティを実現できます。</p>	
READ	ディレクトリ内のファイルの読取り
WRITE	<p>ディレクトリ内のファイルの書込み</p> <p>外部表に接続する場合のみに役立ちます。これによって、権限受領者は、外部表のエージェントがディレクトリに書き込めるのがログ・ファイルなのか不良ファイルなのかを判断できます。</p> <p>制限事項：この権限によって、権限受領者が BFILE に書き込むことはできません。</p>

表 16-4 オブジェクト権限とその権限によって許可される操作（続き）

オブジェクト権限	許可される操作
次の オブジェクト型権限 は、オブジェクト型の操作を許可します。	
EXECUTE	特定のオブジェクトの使用および参照、また、そのメソッドの呼出し
UNDER	型のサブタイプの作成 この型の直属のスーパータイプに WITH GRANT OPTION 付きの UNDER ANY TYPE 権限を持つ場合のみ、このオブジェクト権限を付与できます。
次の 索引タイプ権限 は、索引タイプの操作を許可します。	
EXECUTE	索引タイプの参照
次の 演算子権限 は、ユーザー定義演算子の操作を許可します。	
EXECUTE	演算子の参照

例

ユーザーに対してシステム権限を付与する例 サンプル・ユーザー hr に CREATE SESSION システム権限を付与し、hr が Oracle にログインできるようにするには、次の文を発行します。

```
GRANT CREATE SESSION
  TO hr;
```

ロールに対してシステム権限を付与する例 次の文は、13-74 ページの「[CREATE ROLE の例](#)」で作成したデータ・ウェアハウス管理者ロールに、適切なシステム権限を付与します。

```
GRANT
  CREATE ANY MATERIALIZED VIEW
  , ALTER ANY MATERIALIZED VIEW
  , DROP ANY MATERIALIZED VIEW
  , QUERY REWRITE
  , GLOBAL QUERY REWRITE
  TO dw_manager
  WITH ADMIN OPTION;
```

dw_manager の権限ドメインには、マテリアライズド・ビューに関連するシステム権限が含まれます。

ADMIN OPTION 付きロールを付与する例 サンプル・ユーザー sh に ADMIN OPTION 付きの dw_manager ロールを付与するには、次の文を発行します。

```
GRANT dw_manager
  TO sh
  WITH ADMIN OPTION;
```

dw_manager ロールによって、sh は次の操作を実行できます。

- ロールを使用可能にして、CREATE MATERIALIZED VIEW システム権限を含むそのロールの権限ドメインに登録されている権限の使用
- 他のユーザーに対するそのロールの付与および取消し
- ロールの削除

ロールに対してオブジェクト権限を付与する例 次の文は、13-74 ページの「[CREATE ROLE の例](#)」で作成したデータ・ウェアハウス・ユーザー・ロールに、SELECT オブジェクト権限を付与します。

```
GRANT SELECT ON sh.sales TO warehouse_user;
```

ロールに対してロールを付与する例 次の文は、dw_manager ロールに、dw_user ロールを付与します（両方のロールとも、13-74 ページの「[CREATE ROLE の例](#)」で作成）。

```
GRANT dw_user TO dw_manager;
```

dw_manager ロールは、dw_user ロールのドメインにあるすべての権限を含むことになります。

ディレクトリへのオブジェクト権限を付与する例 ユーザー oe にディレクトリ bfile_dir に対する READ 権限を GRANT OPTION 付きで付与するには、次の文を発行します。

```
GRANT READ ON DIRECTORY bfile_dir TO oe
  WITH GRANT OPTION;
```

ユーザーに対して表へのオブジェクト権限を付与する例 ユーザー hr に対して、16-78 ページの「[MERGE の例](#)」で作成した oe.bonuses 表についてのすべての権限を GRANT OPTION 付きで付与するには、次の文を発行します。

```
GRANT ALL ON bonuses TO hr
  WITH GRANT OPTION;
```

この結果、hr は次の操作が実行できます。

- bonuses 表に対するすべての権限の使用
- 他のユーザーまたはロールに対する、bonuses 表についての権限の付与

ビューへのオブジェクト権限を付与する例 15-46 ページの「[基本ビューの例](#)」で作成したビュー emp_view についての SELECT 権限および UPDATE 権限をすべてのユーザーに付与するには、次の文を発行します。

```
GRANT SELECT, UPDATE
    ON emp_view TO PUBLIC;
```

この結果、すべてのユーザーが、従業員の詳細についてのビューを問合せおよび更新できるようになります。

別のスキーマの順序に対してオブジェクト権限を付与する例 ユーザー hr に対して、スキーマ oe 内の orders_seq 順序の SELECT 権限を付与するには、次の文を発行します。

```
GRANT SELECT
    ON oe.orders_seq TO hr;
```

ユーザー hr は、次の文を指定して、順序の次の値を作成できるようになります。

```
SELECT oe.orders_seq.NEXTVAL
    FROM DUAL;
```

別々の列に複数のオブジェクト権限を付与する例 ユーザー oe に、スキーマ hr にある employees 表の employee_id 列に対する REFERENCES 権限、および employee_id、salary、commission_pct 列に対する UPDATE 権限を付与するには、次の文を発行します。

```
GRANT REFERENCES (employee_id),
    UPDATE (employee_id, salary, commission_pct)
    ON hr.employees
    TO oe;
```

この結果、ユーザー oe は、employee_id 列、salary 列および commission_pct 列の値を更新できるようになります。oe は、employee_id 列を参照する参照整合性制約を定義することもできます。ただし、GRANT 文にはこれらの列のみが指定されているため、ユーザー oe は employees 表の他の列を操作できません。

たとえば、oe は制約付きの表を作成できます。

```
CREATE TABLE dependent
  (dependno    NUMBER,
   dependname  VARCHAR2(10),
   employee    NUMBER
   CONSTRAINT in_emp REFERENCES oe.employees(employee_id) );
```

スキーマ hr 内の employees 表の従業員に対応する dependent 表の依存性が、制約 in_emp によって保証されます。

INSERT

用途

INSERT 文を使用すると、表、ビューの実表、パーティション表のパーティション、コンポジット・パーティション表のサブパーティション、オブジェクト表またはオブジェクト・ビューの実表に、行を追加できます。

前提条件

表に行を挿入する場合は、その表が自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、その表に対する INSERT 権限が必要です。

ビューの実表に行を挿入する場合、ビューが定義されているスキーマの所有者には、その実表に対する INSERT 権限が必要です。また、他のユーザーのスキーマ内のビューに行を挿入する場合は、そのビューに対する INSERT 権限が必要です。

INSERT ANY TABLE システム権限があれば、任意の表または任意のビューの実表に行を挿入できます。

従来型 INSERT およびダイレクト・パス INSERT

表、パーティションまたはビューにデータを挿入するために使用する INSERT 文には、従来型 INSERT およびダイレクト・パス INSERT の 2 種類があります。従来型 INSERT 文を発行すると、表の空き領域を再利用して挿入され、参照整合性制約が維持されます。ダイレクト・パス INSERT の場合、表の既存データの後に挿入するデータが追加されます。データは、バッファ・キャッシュを回避してデータ・ファイルに直接書き込まれます。既存データの空き領域は再利用されません。これによって挿入操作中のパフォーマンスが向上します。また、これは Oracle のダイレクト・パスのローダー・ユーティリティ、SQL*Loader の機能に似ています。

ダイレクト・パス INSERT には、次の制限があります。これらの制限に違反する場合、他にエラーがないかぎりメッセージが戻されず、従来型 INSERT がシリアルで実行されます。

- DML 文の有無にかかわらず、1 つのトランザクションで複数のダイレクト・パス INSERT 文を使用できます。ただし、ある DML 文が特定の表、パーティションまたは索引を変更した後は、トランザクションの他の DML 文は表、パーティションまたは索引にアクセスできません。
- 同じ表、パーティションまたは索引にアクセスする問合せは、ダイレクト・パス INSERT 文の前であれば許可されますが、後は許可されません。

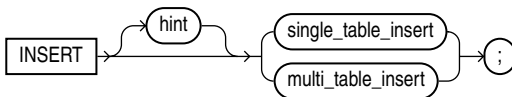
- シリアルまたはパラレル文が、同じトランザクションからダイレクト・パス INSERT によって変更された表にアクセスしようとする場合、エラーが戻され、文が拒否されます。
- 初期化パラメータ ROW_LOCKING は、INTENT に設定されません。
- 対象となる表は、索引構成化またはクラスタ化できません。
- 対象となる表は、オブジェクト型または LOB 列を含むことはできません。
- 対象となる表には、トリガーまたは参照整合性制約を定義できません。
- 対象となる表は、レプリケートできません。
- ダイレクト・パス INSERT 文を含むトランザクションは、分散できません。

参照：

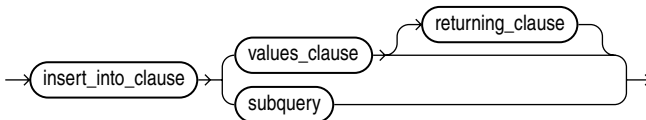
- ダイレクト・パス INSERT の詳細は、『Oracle9i データベース概要』を参照してください。
- SQL*Loader の詳細は、『Oracle9i データベース・ユーティリティ』を参照してください。
- パラレル・ダイレクト・パス INSERT をチューニングする方法の詳細は、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。

構文

insert::=

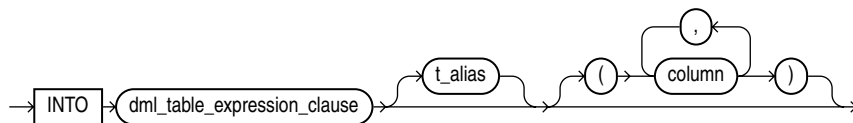


single_table_insert::=

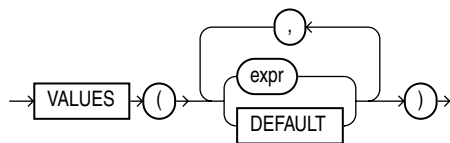


INSERT

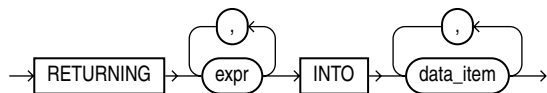
insert_into_clause::=



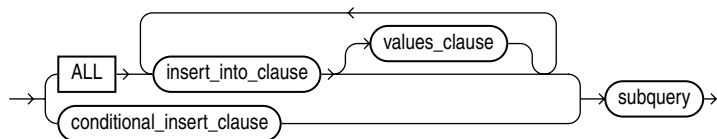
values_clause::=



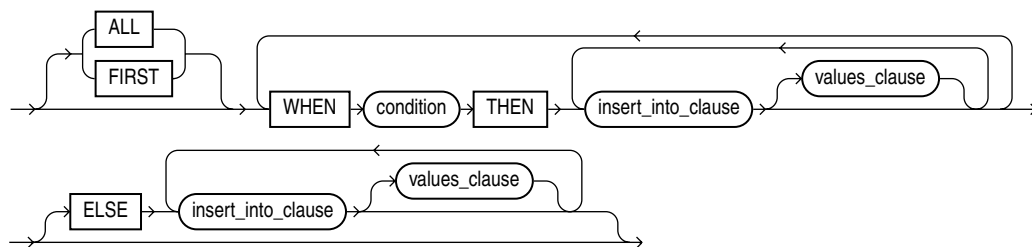
returning_clause::=



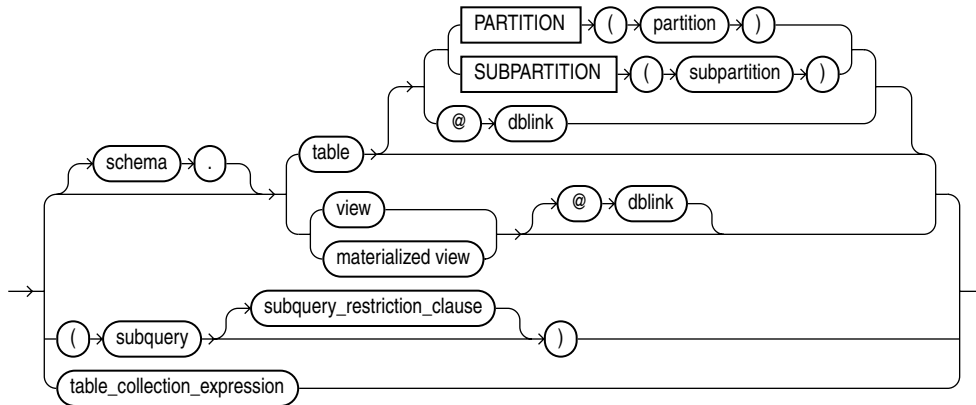
multi_table_insert::=



conditional_insert_clause::=

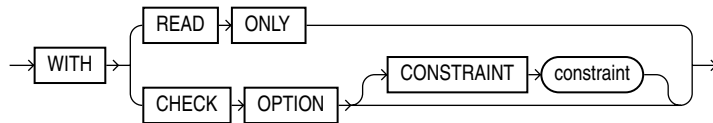


dml_table_expression_clause::=

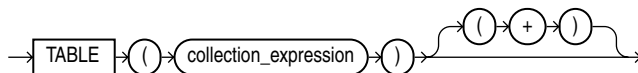


subquery: 17-4 ページの「[SELECT](#)」を参照してください。

subquery_restriction_clause::=



table_collection_expression::=



キーワードとパラメータ

hint

文に対して実行計画を選択する際の、オプティマイザへ指示を渡すコメントを指定します。

マルチ表の挿入の場合、対象となる表に対して `PARALLEL` ヒントを指定すると、表が `PARALLEL` の指定付きで作成または変更されていなくても、マルチ表の `INSERT` 文全体がパラレル化されます。`PARALLEL` ヒントを指定しない場合、対象となるすべての表が `PARALLEL` の指定付きで作成または変更されていないかぎり、挿入操作はパラレル化されません。

参照：

- ヒントの構文および説明については、2-86 ページの「[ヒント](#)」および『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。
- 16-66 ページの「[マルチ表の挿入に関する制限事項](#)」を参照してください。

single_table_insert

単一表の挿入の場合、明示的に値を指定する、または副問合せで値を検索することによって、表、ビューまたはマテリアライズド・ビューの 1 行に値を挿入します。

制限事項：副問合せで値を検索する場合、副問合せの `SELECT` 構文のリストには `INSERT` 文の列リストと同じ数の列が含まれている必要があります。列リストを指定しない場合は、副問合せで表の各列の値を指定する必要があります。

insert_into_clause

`INSERT INTO` 句を使用すると、Oracle がデータを挿入する対象となる表またはオブジェクトを指定できます。

dml_table_expression_clause

`INTO dml_table_expression_clause` を使用すると、データを挿入するオブジェクトを指定できます。

schema 表、ビューまたはマテリアライズド・ビューが含まれるスキーマを指定します。`schema` を指定しない場合、オブジェクトは、自分のスキーマ内に定義されているものとみなされます。

table | view | subquery 行を挿入する表またはオブジェクト表の名前、ビューまたはオブジェクト・ビューの名前、マテリアライズド・ビューの名前、あるいは副問合せから戻された列の名前を指定します。ビューまたはオブジェクト・ビューを指定した場合、Oracle はそのビューの実表に行を挿入します。

挿入される値がオブジェクト表に対する REF の場合、およびそのオブジェクト表に主キー・オブジェクト識別子がある場合、REF を挿入する列は、オブジェクト表に対する参照整合性制約または有効範囲制約を持つ REF 列である必要があります。

table (または **view** の実表) に、1 列以上のドメイン索引がある場合は、この文が適切な索引タイプの挿入ルーチンを実行します。

表に対して INSERT 文を発行した場合、その表に対して定義されている INSERT トリガーが起動します。

参照： これらのルーチンの詳細は、『Oracle9i Data Cartridge Developer's Guide』を参照してください。

PARTITION (partition_name) | SUBPARTITION (subpartition_name) 挿入先の **table** (または **view** の実表) 内にあるパーティションまたはサブパーティションの名前を指定します。

挿入する行が特定のパーティションまたはサブパーティションにマップされない場合、Oracle はエラーを戻します。

制限事項： この句は、オブジェクト表またはオブジェクト・ビューでは無効です。

dblink 表またはビューが格納されているリモート・データベースへのデータベース・リンクの完全名または部分名を指定します。Oracle の分散機能を使用している場合にのみ、リモート表またはリモート・ビューに行を挿入できます。

dblink を指定しない場合、Oracle は、その表またはビューがローカル・データベース内にあるとみなします。

参照： データベース・リンクの参照方法の詳細は、2-110 ページの「[スキーマ・オブジェクトの構文および SQL 文の構成要素](#)」を参照してください。

subquery_restriction_clause *subquery_restriction_clause* を使用すると、次のいずれかの方法で副問合せを制限できます。

- WITH READ ONLY で、副問合せを更新禁止にすることを指定します。
- WITH CHECK OPTION で、副問合せに存在しない行を生成する表変更の禁止を指定します。

参照： 17-30 ページの「[WITH CHECK OPTION の例](#)」を参照してください。

table_collection_expression 問合せおよび DML 操作で、*collection_expression* 値を表として扱う場合に、*table_collection_expression* を指定します。*collection_expression* は副問合せ、列、組込みファンクションまたはコレクション・コンストラクタにすることができます。その形式にかかわらず、集合値（ネストした表型または VARRAY 型の値）を戻す必要があります。このようなコレクションの要素抽出プロセスを **コレクション・ネスト解除** といいます。

注意： 以前のリリースの Oracle では、*collection_expression* が副問合せの場合、*table_collection_expression* を「`THE subquery`」と表現していました。現在、このような表現方法はされていません。

参照： 17-34 ページの「[表コレクションの例](#)」を参照してください。

t_alias 文内で参照する表、ビューまたは副問合せの**相関名**（別名）を指定します。表の別名は、SELECT 構文のリストに定義されません。そのため、SELECT 構文のリストに依存した句では、表の別名は不明です。たとえばこれは、式の中でオブジェクト列を指定する場合に発生する可能性があります。表の別名とともに式を使用するには、列の別名を使用して SELECT 構文のリストに式を置き、複数表の INSERT の VALUE 句または WHEN 条件で、列の別名を指定する必要があります。

制限事項： *t_alias* は、マルチ表の挿入に指定できません。

dml_table_expression_clause に関する制限事項

- *table*（または *view* の実表）に、IN_PROGRESS または FAILED とマークされたドメイン索引がある場合は、この文は実行できません。
- 関係する索引パーティションが UNUSABLE とマークされている場合は、パーティションに挿入できません。
- *dml_table_expression_clause* の *subquery* の ORDER BY 句に関して、順序付けは、挿入された行または表の各エクステント内のみに保証されています。既存の行に関連する新しい行の順序付けは保証されていません。

- WITH CHECK OPTION を指定してビューを作成した場合、ビューに定義されている問合せを満たす行のみビューに挿入されます。
- 単一の実表を使用してビューを作成した場合、行をビューに挿入し、`returning_clause` を使用してその行の値を取り出せます。
- ビューを定義する問合せに、INSTEAD OF トリガー以外の次のいずれかの構造体が含まれる場合は、そのビューは挿入できません。
 - 集合演算子
 - DISTINCT 演算子
 - 集計ファンクションまたは分析ファンクション
 - GROUP BY、ORDER BY、CONNECT BY または START WITH 句
 - SELECT 構文のリストにあるコレクション式
 - SELECT 構文のリストにある副問合せ
 - 結合（一部の例外を除く）

詳細は、『Oracle9i データベース管理者ガイド』を参照してください。
- UNUSABLE のマークが付いている索引、索引パーティションまたは索引サブパーティションを指定する場合、SKIP_UNUSABLE_INDEXES セッション・パラメータが TRUE に設定されていないかぎり、INSERT 文は正常に実行されません。

参照： SKIP_UNUSABLE_INDEXES セッション・パラメータの詳細は、9-2 ページの「ALTER SESSION」を参照してください。

column

表またはビューの列を指定します。挿入された行では、このリストにある各列に `values_clause` または副問合せの値が代入されます。

表のいずれかの列も指定しない場合、挿入された行の列の値には、表の作成時または最終更新時に指定したデフォルト値が使用されます。列のいずれかに NOT NULL 制約がある場合、制約違反のエラーが発生して INSERT 文がロールバックされます。

列リストを指定しない場合は、`values_clause` または問合せに、表の列をすべて指定する必要があります。

参照： 列のデフォルト値の詳細は、14-6 ページの「CREATE TABLE」を参照してください。

values_clause

単一表の挿入操作の場合、表またはビューに挿入する値の行を指定します。なお、値は、列リスト内の各列について *values_clause* に指定する必要があります。列リストを指定しない場合、*values_clause* または副問合せで、表の各列の値を指定する必要があります。

マルチ表の挿入操作の場合、*values_clause* の各式は、副問合せの *SELECT* 構文のリストによって戻される列を参照する必要があります。また、*conditional_insert_clause* の *WHEN* 条件の各式についても、副問合せの *SELECT* 構文のリストによって戻される列を参照する必要があります。*values_clause* を省略すると、副問合せの *SELECT* 構文のリストによって挿入する値が決定されるため、*insert_into_clause* に対応する列リストと同じ列数を含んでいる必要があります。*insert_into_clause* で列リストを指定しない場合、対象となる表のすべての列に対する値を計算した行を指定する必要があります。

どちらの挿入操作の場合も、*insert_into_clause* で列リストを指定すると、リストの各列に値の句または副問合せからの対応する値が割り当てられます。*values_clause* の任意の値に対して *DEFAULT* を指定できます。表またはビューの対応する列に対してデフォルト値を指定すると、その値が挿入されます。対応する列に対してデフォルト値を指定しない場合、*NULL* が挿入されます。

制限事項：

- オブジェクトにある内部 *LOB* 属性を空または *NULL* 以外の値で初期化することはできません。つまり、リテラルを使用することはできません。
- *BFILE* ロケータを *NULL* に、またはディレクトリ別名およびファイル名に初期化するまで、*BFILE* 値を挿入できません。
- リスト・パーティション表に挿入する場合、1 つのパーティションの *partition_value* リストに存在していないパーティション化キー列に値を挿入できません。
- ビューに挿入する場合は、*DEFAULT* を指定できません。

注意： 後続の問合せ中に文字リテラルを *RAW* 列に挿入する場合、*RAW* 列にある索引は使用されずに、フル・テーブル・スキャンが行われます。

参照：

- 16-70 ページの「[BFILE への挿入例](#)」を参照してください。
- *BFILE* ロケータの初期化の詳細は、『Oracle Call Interface プログラマーズ・ガイド』および『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。
- 有効な式の構文の詳細は、4-2 ページの「[SQL 式](#)」および 17-4 ページの「[SELECT](#)」を参照してください。

returning_clause

returning_clause は DML (INSERT、UPDATE または DELETE) 文に影響される行を取り出します。この句は、表、マテリアライズド・ビューおよび単一の実表を持つビューに指定できます。

単一行で処理する場合、*returning_clause* 付きの DML 文で、処理された行および ROWID を使用している列式、処理された行への REF を取り出し、ホスト変数または PL/SQL 変数に格納します。

複数行で処理する場合、*returning_clause* 付きの DML 文で、式の値、ROWID および処理された行に関連する REF をバインド配列に格納します。

expr *expr* リストの各項目は、適切な構文で表す必要があります。スカラー副問合せ式を除くすべての形式が有効です。

INTO 変更された行の値を、*data_item* リストに指定する変数に格納することを示します。

data_item 取り出された *expr* 値を格納するホスト変数または PL/SQL 変数です。

RETURNING リストの各式については、INTO リストに、対応する型に互換がある PL/SQL 変数またはホスト変数を指定する必要があります。

制限事項：

- *returning_clause* は、マルチ表の挿入に指定できません。
- パラレル DML またはリモート・オブジェクトでは、この句を使用できません。
- この句で LONG 型を取り出すことはできません。
- INSTEAD OF トリガーが定義されたビューに対しては、この句を指定できません。

参照： BULK COLLECT 句を使用してコレクション変数に複数の値を戻す場合の詳細は、『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。

multi_table_insert

マルチ表の挿入では、副問合せの評価によって戻された行から導出され、計算された行が 1 つ以上の表に挿入されます。

ALL into_clause

複数の *insert_into_clause* を後ろに伴った ALL を指定すると、**無条件のマルチ表の挿入**が実行されます。副問合せによって戻される各行に対して、各 *insert_into_clause* が 1 回実行されます。

conditional_insert_clause

conditional_insert_clause を指定すると、**条件付きのマルチ表の挿入**が実行されます。実行する *insert_into_clause* を判断する WHEN 条件を使用して、各 *insert_into_clause* が抽出されます。1 つのマルチ表の挿入には、最大 127 個まで WHEN 句の指定が可能です。

ALL ALL を指定すると、他の WHEN 句の評価結果にかかわらず、各 WHEN 句が評価されます。条件が真と評価された各 WHEN 句に対して、対応する INTO 句リストが実行されます。

FIRST FIRST を指定すると、文で指定されている順序で各 WHEN 句が評価されます。真と評価された最初の WHEN 句に対して、対応する INTO 句が実行され、指定された行に対する後続の WHEN 句はスキップされます。

ELSE 句 指定された行に対して、WHEN 句が真と評価されない場合、次のようになります。

- ELSE 句を指定する場合、ELSE 句に関連する INTO 句リストが実行されます。
- 他の句を指定しない場合、行に対して何も実行されません。

マルチ表の挿入に関する制限事項

- 表のみにマルチ表の挿入が実行でき、ビューまたはマテリアライズド・ビューには実行できません。
- リモート表には、マルチ表の挿入を実行できません。
- マルチ表の挿入の実行時、表のコレクション式は指定できません。
- マルチ表の挿入では、すべての *insert_into_clause* に最大 999 列を組み合わせで指定できます。
- マルチ表の挿入は、Real Application Clusters 環境の場合、対象となる表が索引構成されている場合、または対象となる表にビットマップ索引が定義されている場合は、パラレル化されません。
- プラン・スタビリティは、マルチ表の INSERT 文にはサポートされません。
- マルチ表の INSERT 文の副問合せに、順序は使用できません。

subquery

表に挿入される行を戻す副問合せを指定します。副問合せによって、INSERT 文の対象となる表を含む任意の表、ビューおよびマテリアライズド・ビューを参照できます。副問合せで選択された行が 1 行もない場合、表に行は挿入されません。

subquery を TO_LOB ファンクションと組み合わせて、LONG 列にある値を、同じ表の異なる列または別の表にある LOB 値に変換できます。ビュー内で LONG を LOB に移行する場合、実表内で移行を実行してからビューに LOB を追加する必要があります。

注意：

- *subquery* が、既存のマテリアライズド・ビューと（部分的または完全に）同じビューを戻す場合、Oracle は、*subquery* に指定された 1 つ以上の表のかわりにマテリアライズド・ビュー（クエリー・リライト用）を使用することがあります。

参照：マテリアライズド・ビューおよびクエリー・リライトの詳細は、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。

- *subquery* がリモート・オブジェクトを参照する場合、参照がローカル・データベース上でオブジェクトにループバックしないかぎり、INSERT はパラレルで実行されます。ただし、*dml_table_expression_clause* の *subquery* がリモート・オブジェクトを参照する場合は、INSERT はシリアルで実行されます。

参照：14-43 ページの「[CREATE TABLE](#)」の「[parallel_clause](#)」を参照してください。

参照：

- 16-70 ページの「[BFILE への挿入例](#)」を参照してください。
- BFILE の初期化の詳細は、『Oracle Call Interface プログラマーズ・ガイド』および『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。
- 有効な式の構文の詳細は、4-2 ページの「[SQL 式](#)」および 17-4 ページの「[SELECT](#)」を参照してください。

例

値の挿入例 次の文は、サンプル表 `departments` に行を挿入します。

```
INSERT INTO departments
VALUES (280, 'Recreation', 121, 1700);
```

`manager_id` 列のデフォルト値が 121 として `departments` 表が作成された場合、次の文を発行できます。

```
INSERT INTO departments
VALUES (280, 'Recreation', DEFAULT, 1700);
```

次の文は、employees 表に 6 つの列で構成される行を挿入します。NULL または科学表記の数値を設定されている列がそれぞれ 1 つ含まれています。

```
INSERT INTO employees (employee_id, last_name, email,
    hire_date, job_id, salary, commission_pct)
VALUES (207, 'Gregory', 'pgregory@oracle.com',
    sysdate, 'PU_CLERK', 1.2E3, NULL);
```

次の文は、前述の例と同じ結果を表しますが、`dml_table_expression_clause`にある副問合せを使用します。

```
INSERT INTO
    (SELECT employee_id, last_name, email, hire_date, job_id,
    salary, commission_pct FROM employees)
VALUES (207, 'Gregory', 'pgregory@oracle.com',
    sysdate, 'PU_CLERK', 1.2E3, NULL);
```

副問合せを持つ値の挿入例 次の文は、歩合給が給与の 25% 以上の従業員を bonuses 表 (16-78 ページの「[MERGE の例](#)」で作成) にコピーします。

```
INSERT INTO bonuses
    SELECT employee_id, salary*1.1
    FROM employees
    WHERE commission_pct > 0.25 * salary;
```

リモート・データベースへの挿入例 次の文は、データベース・リンク sales がアクセスできるデータベース上の、ユーザー scott が所有する accounts 表に行を挿入します。

```
INSERT INTO scott.accounts@sales (acc_no, acc_name)
VALUES (5001, 'BOWER');
```

accounts 表に balance 列がある場合、INSERT 文に balance の値が指定されていないため、新しく挿入された行にはこの列のデフォルト値が割り当てられます。

順序値の挿入例 次の文は、従業員順序の次の値を持つ行を、departments 表に挿入します。

```
INSERT INTO departments
VALUES (departments_seq.nextval, 'Entertainment', 162, 1400);
```

パーティションへの挿入例 次の文は、latest_data の行を sales 表のパーティション oct98 に挿入します。

```
INSERT INTO sales PARTITION (oct98)
SELECT * FROM latest_data;
```

バインド変数を使用した挿入例 次の文は、出力バインド変数 bnd1 および bnd2 に挿入された行の値を戻します。(バインド変数は最初に宣言しておく必要があります。)

```
INSERT INTO employees
(employee_id, last_name, email, hire_date, job_id, salary)
VALUES
(employee_seq.nextval, 'Doe', 'john.doe@oracle.com',
SYSDATE, 'SA_CLERK', 2400)
RETURNING salary*12, job_id INTO :bnd1, :bnd2;
```

バインド配列への戻り値の例 次の文は、バインド配列 :1 に挿入された行の参照値を戻します。

```
INSERT INTO employee
VALUES ('Kitty Mine', 'Peaches Fuzz', 'Meena Katz')
RETURNING REF(employee) INTO :1;
```

置換可能な表および列への挿入例 次の例は、14-52 ページの「置換可能な表および列のサンプル」で作成した persons 表に挿入します。最初の文は、ルート型 person_t を使用します。2 番目の挿入は、person_t のサブタイプ employee_t を使用し、3 番目の挿入は employee_t のサブタイプ part_time_emp_t を使用します。

```
INSERT INTO persons VALUES (person_t('Bob', 1234));
INSERT INTO persons VALUES (employee_t('Joe', 32456, 12, 100000));
INSERT INTO persons VALUES (
part_time_emp_t('Tim', 5678, 13, 1000, 20));
```

次の例は、14-52 ページの「置換可能な表および列のサンプル」で作成した books 表に挿入します。属性値の指定が、置換可能な表の例と同一であることに注意してください。

```
INSERT INTO books VALUES (
'An Autobiography', person_t('Bob', 1234));
INSERT INTO books VALUES (
'Business Rules', employee_t('Joe', 3456, 12, 10000));
INSERT INTO books VALUES (
'Mixing School and Work',
part_time_emp_t('Tim', 5678, 13, 1000, 20));
```

組み込み関数および条件を使用して、置換可能な表および列からデータを抽出することができます。その例として、6-178 ページの「TREAT」、6-153 ページの「SYS_TYPEID」および 5-17 ページの「IS OF type 条件」を参照してください。

TO_LOB を使用した挿入例 次の例は、LONG データを次の long_tab 表にある LOB 列にコピーします。

```
CREATE TABLE long_tab (long_pics LONG RAW);
```

まず、LOB を持つ表を作成します。

```
CREATE TABLE lob_tab (lob_pics BLOB);
```

次に、INSERT ... SELECT を使用して、LONG 列のすべての行にあるデータを、新しく作成した LOB 列にコピーします。

```
INSERT INTO lob_tab (lob_pics)
  SELECT TO_LOB(long_pics) FROM long_tab;
```

移行が問題なく終了したことを確認した後、long_pics 表を削除できます。別の方法として、表が他の列を含む場合、次のように入力して表から LONG 列を削除できます。

```
ALTER TABLE long_tab DROP COLUMN long_pics;
```

BFILE への挿入例 BFILE に対して挿入または更新する場合、次の例に示すとおり NULL に、またはディレクトリ別名およびファイル名に初期化する必要があります。emp 表の BFILE 列の次に number 列がある場合、次のように入力します。

```
INSERT INTO emp
  VALUES (1, BFILENAME ('a_dir_alias', 'a_filename'));
```

マルチ表の挿入例 次の例はマルチ表の挿入の構文を使用して、サンプル表 sh.sales に、異なる構造の入力表からデータを挿入します。

注意： 例を簡潔に示すために表の列が無視されているため、sales 表の制約は使用禁止になっています。

入力表は次のように構成されています。

```
SELECT * FROM sales_input_table;
```

PRODUCT_ID	CUSTOMER_ID	WEEKLY_ST	SALES_SUN	SALES_MON	SALES_TUE	SALES_WED	SALES_THU	SALES_FRI	SALES_SAT
111	222	01-OCT-00	100	200	300	400	500	600	700
222	333	08-OCT-00	200	300	400	500	600	700	800
333	444	15-OCT-00	300	400	500	600	700	800	900

マルチ表の INSERT 文を次に示します。

```
INSERT ALL
  INTO sales (prod_id, cust_id, time_id, amount)
  VALUES (product_id, customer_id, weekly_start_date, sales_sun)
  INTO sales (prod_id, cust_id, time_id, amount)
  VALUES (product_id, customer_id, weekly_start_date+1, sales_mon)
  INTO sales (prod_id, cust_id, time_id, amount)
  VALUES (product_id, customer_id, weekly_start_date+2, sales_tue)
  INTO sales (prod_id, cust_id, time_id, amount)
  VALUES (product_id, customer_id, weekly_start_date+3, sales_wed)
  INTO sales (prod_id, cust_id, time_id, amount)
  VALUES (product_id, customer_id, weekly_start_date+4, sales_thu)
  INTO sales (prod_id, cust_id, time_id, amount)
  VALUES (product_id, customer_id, weekly_start_date+5, sales_fri)
  INTO sales (prod_id, cust_id, time_id, amount)
  VALUES (product_id, customer_id, weekly_start_date+6, sales_sat)
SELECT product_id, customer_id, weekly_start_date, sales_sun,
       sales_mon, sales_tue, sales_wed, sales_thu, sales_fri, sales_sat
FROM sales_input_table;
```

sales 表には次の行のみが存在するものとします。内容は次のとおりです。

```
SELECT * FROM sales;
```

PROD_ID	CUST_ID	TIME_ID	C	PROMO_ID	QUANTITY_SOLD	AMOUNT	COST
111	222	01-OCT-00				100	
111	222	02-OCT-00				200	
111	222	03-OCT-00				300	
111	222	04-OCT-00				400	
111	222	05-OCT-00				500	
111	222	06-OCT-00				600	
111	222	07-OCT-00				700	
222	333	08-OCT-00				200	
222	333	09-OCT-00				300	
222	333	10-OCT-00				400	
222	333	11-OCT-00				500	
222	333	12-OCT-00				600	
222	333	13-OCT-00				700	
222	333	14-OCT-00				800	
333	444	15-OCT-00				300	
333	444	16-OCT-00				400	
333	444	17-OCT-00				500	
333	444	18-OCT-00				600	
333	444	19-OCT-00				700	
333	444	20-OCT-00				800	
333	444	21-OCT-00				900	

LOCK TABLE

用途

LOCK TABLE 文を使用すると、1 つ以上の表、表パーティションまたはサブパーティションを特定のモードでロックできます。操作中の表またはビューに対する他のユーザーによるアクセスを許可または制限するため、自動ロックを手動で無効にします。

ロックによっては、同じ表に同時に設定できる場合、または表ごとに 1 つのみ設定できる場合があります。

ロックされた表は、トランザクションをコミットするか、全体をロールバックするか、または表をロックする前のセーブポイントにロールバックするまでロックされています。

ロックした場合でも他のユーザーが表を問い合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き込みプログラムをロックすることではなく、書き込みプログラムが読取りプログラムをロックすることもあります。

参照：

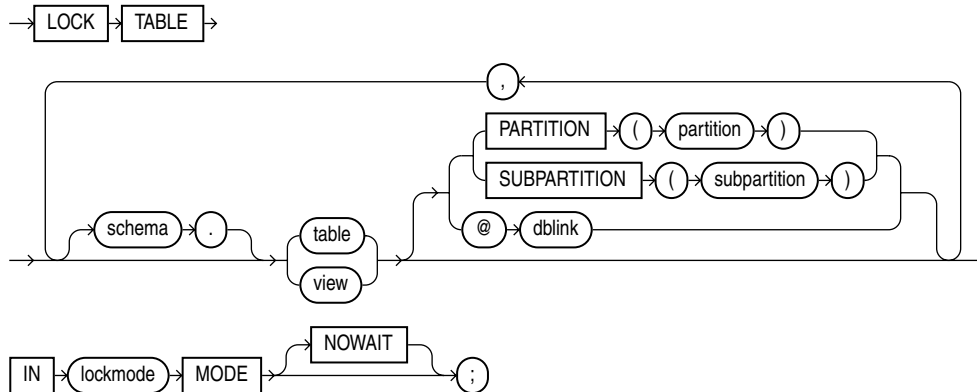
- ロック・モードの相互作用については、『Oracle9i データベース概要』を参照してください。
- 11-69 ページの「COMMIT」を参照してください。
- 16-96 ページの「ROLLBACK」を参照してください。
- 17-2 ページの「SAVEPOINT」を参照してください。

前提条件

表またはビューが自分のスキーマ内にある必要があります。自分のスキーマ内でない場合は、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

lock_table::=



キーワードとパラメータ

schema

表またはビューが含まれているスキーマを指定します。*schema* を指定しない場合、表またはビューが自分のスキーマにあるとみなされます。

table / view

ロックする表の名前を指定します。

view を指定した場合、ビューの実表がロックされます。

PARTITION (*partition*) または **SUBPARTITION** (*subpartition*) を指定する場合、最初にその表を暗黙的にロックします。表ロックは、*partition* または *subpartition* に指定したロックと同じです。ただし、次の 2 つの例外があります。

- **SHARE** ロックをサブパーティションに指定する場合、Oracle は、表を暗黙的に **ROW SHARE** ロックします。
- **EXCLUSIVE** ロックをサブパーティションに指定する場合、Oracle は、表を暗黙的に **ROW EXCLUSIVE** ロックします。

PARTITION を指定し、*table* がコンポジット・パーティション化されている場合、Oracle は、パーティションのすべてのサブパーティションをロックします。

制限事項：*table* が階層の一部である場合、階層のルートである必要があります。

dblink

表またはビューが格納されている、Oracle のリモート・データベースに対するデータベース・リンクを指定します。Oracle 分散機能を使用している場合のみ、リモート・データベースで表およびビューをロックできます。LOCK TABLE 文を使用してロックする表は、すべて同じデータベース上にある必要があります。

dblink を指定しない場合、その表またはビューは、ローカル・データベース内にあるとみなされます。

参照： データベース・リンクの指定方法の詳細は、2-113 ページの「[リモート・データベース内のオブジェクトの参照](#)」を参照してください。

***lockmode* 句**

次のいずれかのモードを指定します。

ROW SHARE ROW SHARE は、ロックされた表への同時アクセスを可能にしますが、排他アクセスのために表全体をロックすることはできなくなります。ROW SHARE は、SHARE UPDATE と同じ意味で、以前のリリースの Oracle との互換性を保つために用意されています。

ROW EXCLUSIVE ROW EXCLUSIVE は、ROW SHARE と同じですが、SHARE モードでロックはできません。ROW EXCLUSIVE ロックは、更新、挿入、削除の実行時に自動的に適用されます。

SHARE UPDATE 「ROW SHARE」を参照してください。

SHARE SHARE は、同時問合せは実行できますが、ロックされた表は更新できません。

SHARE ROW EXCLUSIVE SHARE ROW EXCLUSIVE は、表全体を見る場合に使用します。これを使用すると他のユーザーがその表内の行を見ることはできますが、SHARE モードで表のロックまたは行の更新を行うことはできません。

EXCLUSIVE EXCLUSIVE は、ロックされた表上で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。

NOWAIT

指定した表（あるいは指定したパーティションまたはサブパーティション）が他のユーザーによってすでにロックされている場合、制御をすぐに戻すには NOWAIT を指定します。この場合、表、パーティションまたはサブパーティションが他のユーザーによってロックされていることを示すエラー・メッセージが戻ります。

この句を指定しない場合、Oracle は、表が使用可能になるまで待ち状態になり、表をロックした後に、発行元に制御を戻します。

例

LOCK TABLE の例 次の文は、employees 表を排他モードでロックします。他のユーザーがすでに表をロックしている場合でも、待ち状態にはなりません。

```
LOCK TABLE employees  
    IN EXCLUSIVE MODE  
    NOWAIT;
```

次の文は、データベース・リンク boston を介してアクセスできるリモート表 accounts をロックします。

```
LOCK TABLE accounts@boston  
    IN SHARE MODE;
```

MERGE

用途

MERGE 文を使用すると、1 つの表から行を選択し、その他の表に対して更新および挿入できます。対象の表に更新と挿入のどちらを実行するかは、ON 句の条件に基づきます。

この文は、2 つ以上の操作を組み合わせるときに便利です。DML 文 INSERT および UPDATE を複数指定する必要がなくなります。

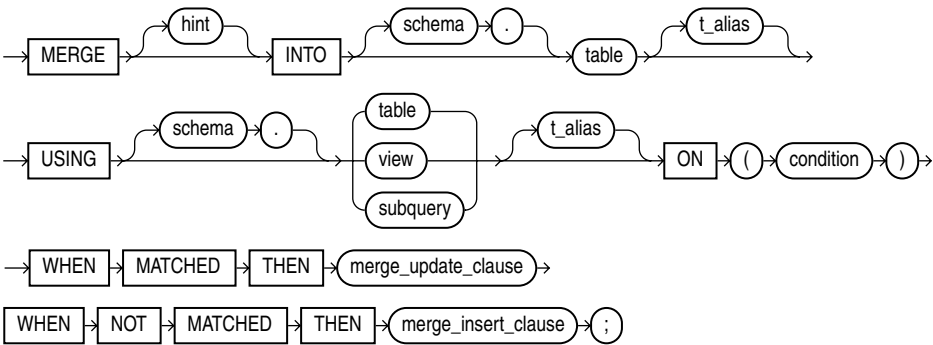
MERGE は、決定的な文です。つまり、同一の MERGE 文で対象となる表の同じ行を複数回更新できません。

前提条件

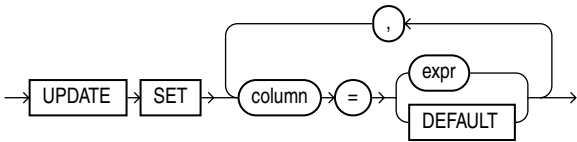
対象となる表に対する INSERT オブジェクト権限と UPDATE オブジェクト権限、およびソース表に対する SELECT 権限が必要です。

構文

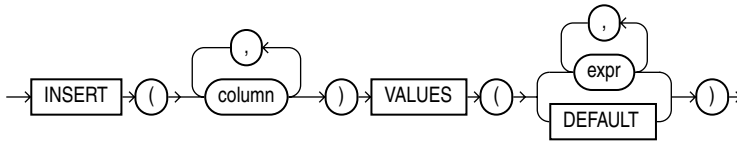
merge::=



merge_update_clause::=



merge_insert_clause::=



キーワードとパラメータ

INTO 句

INTO 句を使用すると、更新または挿入の対象となる表を指定できます。

USING 句

USING 句を使用すると、更新または挿入の対象となるデータのソースを指定できます。ソースには、表、ビューまたは副問合せの結果を指定できます。

ON 句

ON 句を使用すると、MERGE の更新操作または挿入操作の条件を指定できます。対象となる表の中で検索条件が真となる各行は、ソース表の対応するデータに基づいて行が更新されます。どの行も条件が真とならない場合、ソース表の対応する行に基づいて対象となる表に行が挿入されます。

WHEN MATCHED | NOT MATCHED

この句を使用すると、*join_condition* の結果への応答方法を Oracle に指示できます。これら 2 つの句は、いずれかの順序で指定できます。

merge_update_clause

merge_update_clause を指定すると、対象となる表の新しい列値を指定できます。ON 句の条件が真となる場合、更新が実行されます。更新が実行されると、対象となる表に定義されているすべての更新トリガーがアクティブになります。

制限事項：ビューを更新する場合は、DEFAULT を指定できません。

merge_insert_clause

merge_insert_clause を指定すると、ON 句の条件が偽となる場合に対象となる表の列に挿入する値を指定できます。挿入が実行されると、対象となる表に定義されているすべての挿入トリガーがアクティブになります。

制限事項：ビューを更新する場合は、DEFAULT を指定できません。

例

MERGE の例 次の例は、サンプル・スキーマ oe 内に bonus 列のデフォルト値を 100 として bonuses 表を作成します。次に、bonuses 表に販売実績があった (oe.orders 表の sales_rep_id 列に基づく) すべての従業員を挿入します。最終的に、人事情報マネージャがすべての従業員にボーナスを支給することを決定します。販売実績がなかった従業員には、給与の 1% がボーナスとして支給されます。販売実績があった従業員には、給与の 1% がボーナスに加算されて支給されます。MERGE 文は、これらの変更を 1 行で実装します。

```
CREATE TABLE bonuses (employee_id NUMBER, bonus NUMBER DEFAULT 100);

INSERT INTO bonuses(employee_id)
  (SELECT e.employee_id FROM employees e, orders o
   WHERE e.employee_id = o.sales_rep_id
   GROUP BY e.employee_id);
```

```
SELECT * FROM bonuses;
```

EMPLOYEE_ID	BONUS
-----	-----
153	100
154	100
155	100
156	100
158	100
159	100
160	100
161	100
163	100

```
MERGE INTO bonuses D
  USING (SELECT employee_id, salary, department_id FROM employees
   WHERE department_id = 80) S
  ON (D.employee_id = S.employee_id)
  WHEN MATCHED THEN UPDATE SET D.bonus = D.bonus + S.salary*.01
  WHEN NOT MATCHED THEN INSERT (D.employee_id, D.bonus)
  VALUES (S.employee_id, S.salary*0.1);
```

EMPLOYEE_ID	BONUS
-----	-----
153	180
154	175
155	170
156	200
158	190
159	180
160	175

161	170
163	195
157	950
145	1400
170	960
179	620
152	900
169	1000

.
.
.

NOAUDIT

用途

NOAUDIT を使用すると、AUDIT 文によって有効になった監査を停止できます。

NOAUDIT 文は先に発行した AUDIT 文と同じ構文である必要があります。また、NOAUDIT 文は、その特定の AUDIT 文のみを無効にします。たとえば、1 番目の AUDIT 文 (A) は特定のユーザーに対する監査を有効にするものとします。2 番目の文 (B) が、すべてのユーザーに対して監査を有効にします。すべてのユーザーに対して監査を無効にする NOAUDIT 文 (C) は、文 B を無効にします。ただし、文 A は無効にされず、文 A が指定したユーザーの監査は継続されます。

参照： 監査の詳細は、11-50 ページの「[AUDIT](#)」を参照してください。

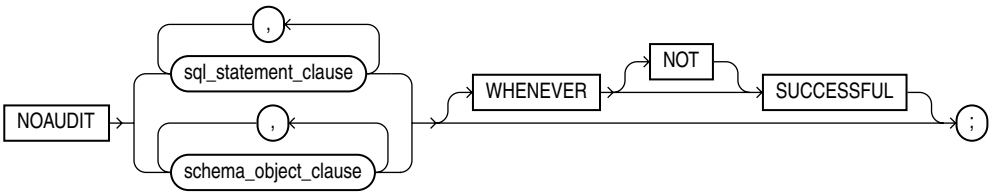
前提条件

SQL 文の監査を停止するには、AUDIT SYSTEM システム権限が必要です。

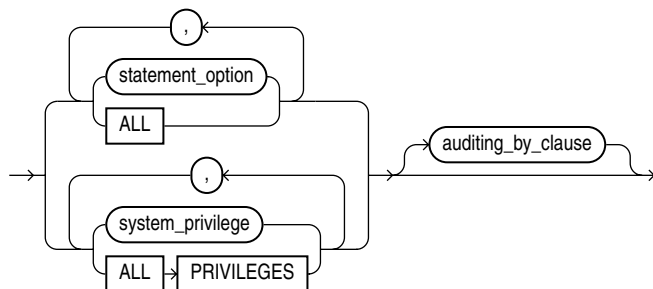
スキーマ・オブジェクトの監査を停止するには、監査を停止するオブジェクトが自分のスキーマ内にある必要があります。自分のスキーマ内にない場合は、AUDIT ANY システム権限が必要です。監査の対象として選択するオブジェクトがディレクトリの場合、自分が作成したディレクトリであっても、AUDIT ANY システム権限が必要です。

構文

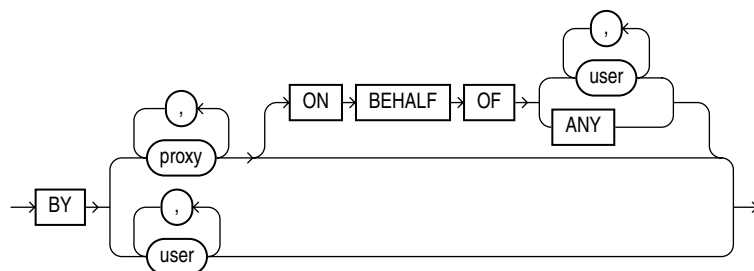
`noaudit::=`



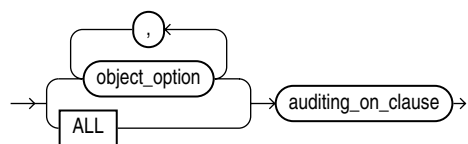
sql_statement_clause::=



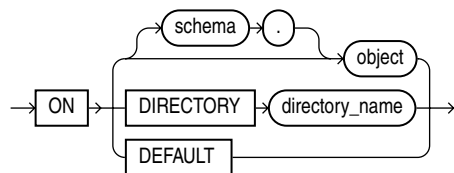
auditing_by_clause::=



schema_object_clause::=



auditing_on_clause::=



キーワードとパラメータ

sql_statement_clause

sql_statement_clause を使用し、特定の SQL 文の監査を停止します。

statement_option *statement_option* には、監査を停止する文のオプションを指定します。

参照： 文のオプションおよびそれによって監査される SQL 文のリストは、11-56 ページの表 11-1 および 11-58 ページの表 11-2 を参照してください。

ALL 現在監査されているすべての文オプションの監査を停止する場合に、ALL を指定します。

system_privilege *system_privilege* には、監査を停止するシステム権限を指定します。

参照： システム権限および各システム権限によって許可される文については、16-38 ページの表 16-1 を参照してください。

ALL PRIVILEGES 現在監査されているすべてのシステム権限の監査を停止する場合に、ALL PRIVILEGES を指定します。

auditing_by_clause *auditing_by_clause* は、特定のユーザーが発行する SQL 文のみを監査します。この句を指定しない場合、すべてのユーザー文の監査が停止されます。

- 指定したユーザーのそれ以降のセッションで発行される SQL 文の監査のみを取り消す場合に、BY *user* を指定します。この句を指定しない場合、すべてのユーザー文の監査を停止します。ただし、WHENEVER SUCCESSFUL で説明する状況は除きます。
- 特定のユーザーまたは任意のユーザーのかわりに指定されたプロキシによって発行された SQL 文の監査のみを停止する場合に、BY *proxy* を指定します。

schema_object_clause

schema_object_clause を使用し、特定のデータベース・オブジェクトの監査を停止します。

object_option *object_option* には、ON 句で指定したオブジェクトへの監査を停止する操作の種類を指定します。

参照： これらのオプションのリストは、11-59 ページの表 11-3 を参照してください。

ALL ALL をショートカットに指定することは、オブジェクト・タイプに適用できるオプションをすべて指定することと同じです。

auditing_on_clause *auditing_on_clause* では、監査を停止する特定のスキーマ・オブジェクトを指定できます。

- オブジェクトには、表、ビュー、順序、ストアド・プロシージャ、ファンクション、パッケージ、マテリアライズド・ビューまたはライブラリのオブジェクト名を指定します。object に *schema* を指定しない場合、自分のスキーマ内にあるとみなされます。

参照： 特定のスキーマ・オブジェクトの監査については、11-50 ページの「**AUDIT**」を参照してください。

- DIRECTORY では、監査を停止するディレクトリ名を指定できます。
- DEFAULT を指定して、オブジェクトを作成した後に、特定のオブジェクト・オプションをデフォルト・オブジェクト・オプションとして削除します。

WHENEVER [NOT] SUCCESSFUL 正常に実行されたスキーマ・オブジェクトに対する SQL 文および操作の監査のみを停止する場合は、WHENEVER SUCCESSFUL を指定します。

WHENEVER NOT SUCCESSFUL は、Oracle エラーとなった文および操作の監査のみを停止します。

この句を指定しない場合、正常に実行されたかどうかにかかわらず、すべての文および操作の監査が停止されます。

例

ロールに関連する SQL 文の監査の停止例 次の文は、ロールを作成または削除するすべての SQL 文の監査を停止します。

```
NOAUDIT ROLE;
```

特定ユーザーが所有するオブジェクトに対する更新または問合せの監視の停止例 次の文は、ユーザー hr および oe によって発行された、表の問合せまたは更新を実行する文を監査している場合、hr の問合せの監査のみを停止します。

```
NOAUDIT SELECT TABLE BY hr;
```

この結果、ユーザー hr の問合せの監査のみが停止されます。oe の問合せと更新、および hr の更新の監査は継続されます。

特定のオブジェクト権限で許可された文の監査の停止例 次の文は、DELETE ANY TABLE システム権限に許可されたすべての文の監査を停止します。

```
NOAUDIT DELETE ANY TABLE;
```

特定のオブジェクトに対する問合せの監査の停止例 スキーマ hr 内の employees 表に問い合わせるすべての SQL 文の監査を選択していた場合、次の文を発行すると、この監査が停止されます。

```
NOAUDIT SELECT
  ON hr.employees;
```

正常に実行される問合せの監査の停止例 次の文は、正常に終了した問合せの監査を停止します。

```
NOAUDIT SELECT
  ON hr.employees
  WHENEVER SUCCESSFUL;
```

この文は、正常に終了した問合せの監査のみ停止します。Oracle は、Oracle エラーの結果発生した問合せの監査を継続します。

RENAME

用途

RENAME 文を使用すると、表、ビュー、順序または（表、ビューまたは順序の）プライベート・シノニムの名前を変更できます。

- 古いオブジェクトの整合性制約、索引および権限付与は、新しいオブジェクトに自動的に移行されます。
- 名前を変更した表を参照するビュー、シノニム、ストアド・プロシージャ、ストアド・ファンクションなど、名前を変更したオブジェクトに依存するオブジェクトはすべて無効になります。

この文を使用してパブリック・シノニムの名前を変更しないでください。そのかわり、該当するパブリック・シノニムを削除し、新しい名前でのパブリック・シノニムを作成してください。

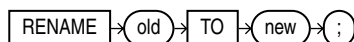
参照： 14-2 ページの「[CREATE SYNONYM](#)」および 16-4 ページの「[DROP SYNONYM](#)」を参照してください。

前提条件

オブジェクトが自分のスキーマ内にある必要があります。

構文

rename::=



キーワードとパラメータ

old

既存の表、ビュー、順序またはプライベート・シノニムの名前を指定します。

new

既存のオブジェクトに指定する新しい名前を指定します。新しい名前は、同じネームスペース内の他のスキーマ・オブジェクトに使用されている名前以外を指定する必要があります。また、スキーマ・オブジェクトのネーミング規則に従って指定する必要があります。

参照： 2-106 ページの「[スキーマ・オブジェクトのネーミング規則](#)」を参照してください。

例

データベース・オブジェクトの名前の変更例 表の名前を departments から emp_departments に変更する場合、次の文を発行します。

```
RENAME departments TO emp_departments;
```

この文では列名を直接変更できません。ただし、AS 副問合せを指定した CREATE TABLE 文とともにこの文を使用すると、列名を変更できます。次の文は、job_history 表を再作成し、department_id から dept_id に列名を変更します。

```
CREATE TABLE temporary
    (employee_id, start_date, end_date, job_id, dept_id)
AS SELECT
    employee_id, start_date, end_date, job_id, department_id
FROM job_history;

DROP TABLE job_history;

RENAME temporary TO job_history;
```


REVOKE

用途

REVOKE 文は、次の処理を行う場合に使用します。

- ユーザーおよびロールからのシステム権限の取消し
- ユーザーおよびロールからのロールの取消し
- ユーザーまたはロールからの特定のオブジェクトに対するオブジェクト権限の取消し

参照：

- システム権限およびロールの付与については、16-31 ページの「[GRANT](#)」を参照してください。
- それぞれのオブジェクト型に対するオブジェクト権限の概要については、16-49 ページの表 16-3 を参照してください

前提条件

システム権限または**ロール**を取り消すには、ADMIN OPTION 付きの権限が必要です。

ロールを取り消すには、ADMIN OPTION 付きのロールが必要です。なお、GRANT ANY ROLE システム権限がある場合は、ロールを自由に取り消すことができます。

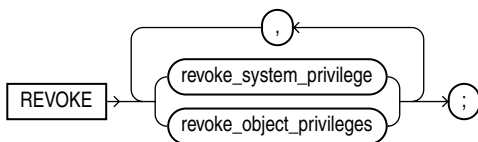
オブジェクト権限を取り消すには、各ユーザーおよびロールに、オブジェクト権限が事前に付与されている必要があります。

REVOKE 文によって取り消すことができる権限およびロールは、GRANT 文によって直接付与されているものに限られます。この句では、次の権限を取り消すことはできません。

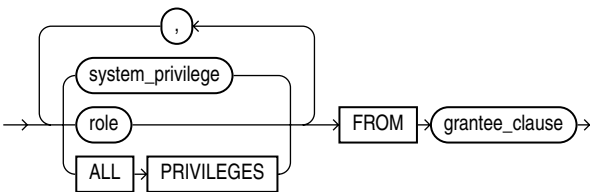
- 取消し側に付与されていない権限またはロール
- オペレーティング・システムを介して付与されているロールまたはオブジェクト権限
- ロールを介して取消し側に付与されている権限またはロール

構文

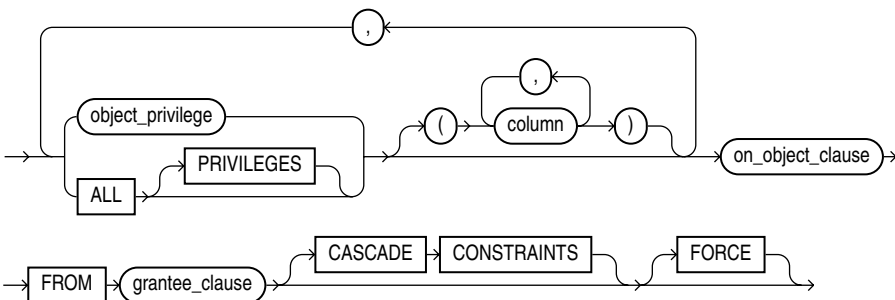
revoke::=



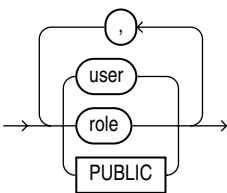
revoke_system_privileges::=



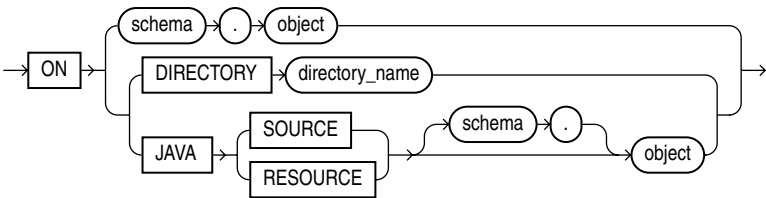
revoke_object_privileges::=



grantee_clause::=



on_object_clause::=



キーワードとパラメータ

revoke_system_privileges

system_privilege

取り消すシステム権限を指定します。

参照： システム権限のリストは、16-38 ページの表 16-1 を参照してください。

- **ユーザーの権限**を取り消す場合は、ユーザーの権限ドメインの権限を取り消します。この取消しはすぐに有効になるため、このユーザーはその権限を使用できなくなります。
- **ロールの権限**を取り消す場合は、ロールの権限ドメインの権限を取り消します。この取消しはすぐに有効になるため、そのロールが使用可能となっている場合でも、この権限は使用できません。また、そのロールが権限付与されている他のユーザーは、ロールを使用可能にしても、その権限を使用できなくなります。
- **PUBLIC の権限**を取り消す場合は、PUBLIC を介して権限を付与されている各ユーザーの権限ドメインの権限を取り消します。この取消しはすぐに有効になるため、ユーザーは権限を使用できなくなります。ただし、直接またはロールを介して権限が付与されているユーザーからは、権限を取り消すことはできません。

制限事項： 取り消す権限のリストに権限を指定できるのは 1 回のみです。

すべてのシステム権限を一度に指定できるショートカットがあります。

- **ALL PRIVILEGES**。ALL PRIVILEGES を指定すると、16-38 ページの表 16-1 に示すすべてのシステム権限を取り消すことができます。

role

取り消すロールを指定します。

- **ユーザーからロール**を取り消す場合は、ユーザーによるロールの使用を禁止します。そのロールが使用可能となっている場合に、ロールの権限ドメインの権限が使用可能な場合はその権限を使用できます。ただし、ユーザーが後からロールを使用可能にできません。
- **他のロールからロール**を取り消す場合、Oracle は取消し側ロールの権限ドメインから、取り消されたロールの権限ドメインを削除します。被取消し側ロールの権限が付与されており、そのロールの権限が使用可能になっているユーザーは、そのロールの権限が使用可能な間は、取り消されたロールの権限ドメインの権限を引続き使用できます。ただし、被取消し側の権限を付与されていても、ロールの取消し操作の後でそれを使用可能にしたユーザーは、取り消されたロールの権限ドメインの権限を使用できません。

- **PUBLIC** のロールを取り消す場合、**PUBLIC** を介してロールが付与されているすべてのユーザーに対して、そのロールを使用禁止にします。そのロールを使用可能としているユーザーは、権限ドメインの権限が使用可能であるかぎり、権限ドメインでその権限を引き続き使用できます。ただし、ユーザーが後からロールを使用可能にすることはできません。直接またはロールを介して権限が付与されているユーザーからは、ロールを取り消すことはできません。

制限事項：取り消すロールのリストにロールを指定できるのは1回のみです。

参照： 事前定義されたロールのリストは、16-48 ページの表 16-2 を参照してください。

grantee_clause

FROM *grantee_clause* は、システム権限、ロールまたはオブジェクト権限が取り消されるユーザーまたはロールを識別します。

PUBLIC すべてのユーザーから権限を取り消す場合は、**PUBLIC** を指定します。

revoke_object_privileges

object_privilege

取り消すオブジェクト権限を指定します。次のいずれかの値に置換することができます。

ALTER、DELETE、EXECUTE、INDEX、INSERT、READ、REFERENCES、SELECT
または UPDATE。

注意： 操作は権限によって許可されます。権限を取り消すことによって、取り消されたユーザーは、その操作ができなくなります。ただし、複数のユーザーが、同じ権限を同一ユーザー、ロールまたは **PUBLIC** に対して付与できます。権限受領者の権限ドメインの権限を取り消す場合、すべての権限付与者が権限を取り消す必要があります。権限を取り消さない権限付与者が1人でもいれば、権限受領者は引き続きその権限を使用できます。

ユーザーの権限を取り消す場合は、ユーザーの権限ドメインの権限を取り消します。この取消しはすぐに有効になるため、このユーザーはその権限を使用できなくなります。

- ユーザーがその権限を他のユーザーまたはロールに付与している場合、他のユーザーまたはロールの権限も取り消されます。
- 権限を使用する SQL 文を記述したプロシージャ、ファンクションまたはパッケージが定義されているスキーマを持つユーザーのオブジェクト権限を取り消すと、それらのプロシージャ、ファンクションまたはパッケージは実行できなくなります。

- そのユーザーのスキーマにオブジェクトのビューが含まれる場合、ビューは無効になります。
- 参照整合性制約の定義権限を使用したユーザーの REFERENCES 権限を取り消す場合、CASCADE CONSTRAINTS 句も指定する必要があります。

ロールの権限を取り消す場合は、ロールの権限ドメインの権限を取り消します。この取消しはすぐに有効になるため、そのロールが使用可能となっている場合でも、この権限は使用できません。ロールが権限付与されている他のユーザーは、ロールを使用可能にした場合でも、権限を使用できなくなります。

PUBLIC の権限を取り消す場合は、PUBLIC を介して権限を付与されている各ユーザーの権限ドメインの権限を取り消します。この取消しはすぐに有効になるため、それらのすべてのユーザーは、その権限を使用できなくなります。ただし、直接またはロールを介して権限が付与されているユーザーからは、権限を取り消すことはできません。

制限事項：取り消す権限のリストに権限を指定できるのは1回のみです。FROM 句にユーザー、ロールまたは PUBLIC を指定できるのは1回のみです。

ALL [PRIVILEGES]

ユーザーまたはロールに付与されているすべてのオブジェクト権限を取り消す場合に、ALL を指定します（キーワード PRIVILEGES はセマンティクスを明確にするためのものであり、指定は任意です。）

注意： オブジェクトに権限が付与されていない場合、処理は行われず、エラーも戻りません。

CASCADE CONSTRAINTS

この句は、REFERENCES 権限または ALL [PRIVILEGES] を取り消すときにのみ適用されます。取消し側で REFERENCES 権限（ALL [PRIVILEGES] を付与して明示的または暗黙的に付与された権限）を使用して定義した参照整合性制約を削除します。

FORCE

表または型に依存するユーザー定義型オブジェクトで、EXECUTE オブジェクト権限を取り消す場合に、FORCE を指定します。表に依存するユーザー定義型オブジェクトでは、FORCE を使用して EXECUTE オブジェクト権限を取り消します。

FORCE を指定した場合、すべての権限が取り消されますが、すべての依存するオブジェクトには INVALID のマークが付けられ、依存する表のデータにはアクセスできなくなります。また、すべての依存するファンクション索引には、UNUSABLE のマークが付けられます（必要な型の権限を再付与した場合、表に対して再度妥当性チェックが行われます）。

参照： 型依存性およびユーザー定義オブジェクト権限の詳細は、『Oracle9i データベース概要』を参照してください。

on_object_clause

on_object_clause は、権限を取り消すオブジェクトを識別します。

object オブジェクト権限を取り消すオブジェクトを指定します。取り消すことができるオブジェクトは次のとおりです。

- 表、ビュー、順序、プロシージャ、ストアド・ファンクション、パッケージまたはマテリアライズド・ビュー
- 表、ビュー、順序、プロシージャ、ストアド・ファンクション、パッケージまたはマテリアライズド・ビューに対するシノニム
- ライブラリ、索引タイプまたはユーザー定義演算子

schema を指定しなかった場合、自分のスキーマ内にあるとみなされます。

(GRANT OPTION の有無にかかわらず) SELECT オブジェクト権限をマテリアライズド・ビューまたは表を含むマテリアライズド・ビューから取り消すと、マテリアライズド・ビューは無効になります。

(GRANT OPTION の有無にかかわらず) マテリアライズド・ビューのマスター表のいずれかにおける SELECT オブジェクト権限を取り消すと、マテリアライズド・ビューとそれに含まれる表またはマテリアライズド・ビューの両方が無効になります。

DIRECTORY *directory_name* 権限を取り消すディレクトリ・オブジェクトを指定します。*directory_name* にはスキーマを指定できません。このオブジェクトはディレクトリである必要があります。

参照： 12-44 ページの「[CREATE DIRECTORY](#)」を参照してください。

JAVA SOURCE | RESOURCE JAVA 句によって、権限が取り消された Java ソースまたはリソース・スキーマ・オブジェクトを指定します。

例

ユーザーからシステム権限を取り消す例 次の文は、ユーザー hr および oe の DROP ANY TABLE システム権限を取り消します。

```
REVOKE DROP ANY TABLE
FROM hr, oe;
```

この結果、hr および oe は他のユーザーのスキーマに定義されている表を削除できなくなります。

ユーザーからロールを取り消す例 次の文は、ユーザー sh のロール dw_manager を取り消します。

```
REVOKE dw_manager
FROM sh;
```

この結果、sh はロール dw_manager を使用できなくなります。

ロールからシステム権限を取り消す例 次の文は、ロール dw_manager の CREATE TABLESPACE システム権限を取り消します。

```
REVOKE CREATE TABLESPACE
FROM dw_manager;
```

ロール dw_manager を使用可能にしても、ユーザーは表領域を作成できません。

ロールからロールを取り消す例 次の文は、ロール dw_manager からロール dw_user を取り消します。

```
REVOKE dw_user
FROM dw_manager;
```

この結果、dw_user の権限はロール dw_manager に付与されなくなります。

ユーザーからオブジェクト権限を取り消す例 次の文は、orders 表に対する DELETE、INSERT、SELECT および UPDATE 権限をユーザー hr に付与します。

```
GRANT ALL
ON orders TO hr;
```

次の文は、ユーザー hr から表 orders に対する DELETE 権限を取り消します。

```
REVOKE DELETE
ON orders FROM hr;
```

ユーザーからすべてのオブジェクト権限を取り消す例 次の文は、ユーザー hr の表 orders に対する残りのすべての権限を取り消します。

```
REVOKE ALL
ON orders FROM hr;
```

PUBLIC からオブジェクト権限を取り消す例 次の文は、ロール public に権限を付与することによって、すべてのユーザーにビュー emp_details_view に対する SELECT 権限および UPDATE 権限を付与します。

```
GRANT SELECT, UPDATE
  ON emp_details_view TO public;
```

次の文は、すべてのユーザーから emp_details_view に対する UPDATE 権限を取り消します。

```
REVOKE UPDATE
  ON emp_details_view FROM public;
```

ユーザーは、emp_details_view ビューへの問合せはできますが、更新はできなくなります。ただし、emp_details_view に対する UPDATE 権限も任意のユーザーに直接またはロールを介して付与している場合は、これらのユーザーはその権限を保持します。

ユーザーから順序のオブジェクト権限を取り消す例 次の文は、ユーザー oe にスキーマ hr 内の departments_seq 順序に対する SELECT 権限を付与します。

```
GRANT SELECT
  ON hr.departments_seq TO oe;
```

oe から departments_seq に対する SELECT 権限を取り消す場合、次の文を発行します。

```
REVOKE SELECT
  ON hr.departments_seq FROM oe;
```

ただし、ユーザー hr が departments に対する SELECT 権限を sh に付与している場合、sh は、hr からの権限付与によって departments を使用できます。

CASCADE CONSTRAINTS でオブジェクト権限を取り消す例 次の文は、oe に、スキーマ hr 内の employees 表に対する REFERENCES 権限および UPDATE 権限を付与します。

```
GRANT REFERENCES, UPDATE
  ON hr.employees TO oe;
```

oe は、REFERENCES 権限を使用して、スキーマ hr 内の employees 表を参照する dependent 表の制約を定義できます。

```
CREATE TABLE dependent
(dependno    NUMBER,
dependname  VARCHAR2(10),
employee    NUMBER
  CONSTRAINT in_emp REFERENCES hr.employees(employee_id) );
```


CASCADE CONSTRAINTS 句を指定した次の文を発行することによって、oe の hr.employees に対する REFERENCES 権限を取り消すことができます。

```
REVOKE REFERENCES
  ON hr.employees
  FROM oe
  CASCADE CONSTRAINTS;
```

oe の hr.employees に対する REFERENCES 権限を取り消した場合、oe は制約を定義する権限が必要になるため、in_emp 制約が自動的に削除されます。

ただし、oe が他のユーザーから hr.employees に対する REFERENCES 権限を付与されている場合は、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、oe は制約に対して必要な権限を保持しています。

ユーザーからディレクトリのオブジェクト権限を取り消す例 次の文は、hr のディレクトリ bfile_dir に対する READ 権限を取り消します。

```
REVOKE READ ON DIRECTORY bfile_dir FROM hr;
```

ROLLBACK

用途

ROLLBACK 文を使用すると、現行のトランザクションで実行された処理を取り消すことができます。また、インダウト分散トランザクションで実行された処理を手動で取り消すこともできます。

注意： アプリケーション・プログラムでは、COMMIT または ROLLBACK 文を使用してトランザクションを明示的に終了することをお薦めします。トランザクションを明示的にコミットせずにプログラムが異常終了した場合、コミットされていない最後のトランザクションがロールバックされます。

参照：

- トランザクションの詳細は、『Oracle9i データベース概要』を参照してください。
- 現行トランザクションの特性の設定については、17-46 ページの「[SET TRANSACTION](#)」を参照してください。
- 11-69 ページの「[COMMIT](#)」を参照してください。
- 17-2 ページの「[SAVEPOINT](#)」を参照してください。

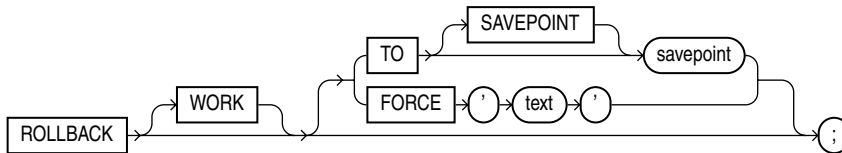
前提条件

現行のトランザクションをロールバックする場合、権限は不要です。

コミットしたインダウト分散トランザクションを手動でロールバックする場合は、FORCE TRANSACTION システム権限が必要です。他のユーザーがコミットしたインダウト分散トランザクションを手動でロールバックする場合は、FORCE ANY TRANSACTION システム権限が必要です。

構文

rollback::=



キーワードとパラメータ

WORK

キーワード WORK の指定は任意です。ANSI 互換性のために提供されています。

TO SAVEPOINT 句

現行のトランザクションをロールバックするセーブポイントを指定します。この句を指定しない場合、ROLLBACK 文によってトランザクション全体がロールバックされます。

TO SAVEPOINT 句を指定しない 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
- 現行のトランザクションに対するすべての変更の取消し
- トランザクション中のセーブポイントの消去
- トランザクションのロックの解除

参照： 17-2 ページの「[SAVEPOINT](#)」を参照してください。

TO SAVEPOINT 句を指定する 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後のトランザクションにロールバックします。
- 指定したセーブポイントより後に作成されたセーブポイントはすべて消去されます。指定したセーブポイントはそのまま残るため、そのセーブポイントまで繰り返しロールバックできます。指定したセーブポイントより前に作成されたセーブポイントも残ります。
- 指定したセーブポイント以降に設定された表と行のロックを解除します。セーブポイント後にロックされた行へのアクセスを要求した他のトランザクションは、コミットまたはロールバックされるまで待つ必要があります。行を要求していない他のトランザクションは、すぐに行の要求およびアクセスができます。

制限事項： インダウト・トランザクションは、セーブポイントまで手動でロールバックできません。

FORCE 句

インダウト分散トランザクションを手動でロールバックする場合に、FORCE を指定します。このトランザクションは、ローカル・トランザクション ID またはグローバル・トランザクション ID を含む *text* で識別されます。このトランザクションの ID を確認する場合、データ・ディクショナリ・ビュー DBA_2PC_PENDING を問い合わせます。

FORCE 句を指定した ROLLBACK 文では、指定したトランザクションのみをロールバックするため注意してください。この文は、現行のトランザクションには影響しません。

制限事項: PL/SQL では、FORCE 句を指定する ROLLBACK 文はサポートされていません。

例

次の文は、現行のトランザクション全体をロールバックします。

```
ROLLBACK;
```

次の文は、現行のトランザクションをセーブポイント sp5 にロールバックします。

```
ROLLBACK TO SAVEPOINT sp5;
```

次の文は、インダウト分散トランザクションを手動でロールバックします。

```
ROLLBACK WORK  
  FORCE '25.32.87';
```

SQL 文 : SAVEPOINT ~ UPDATE

この章では、次の SQL 文について説明します。

- `SAVEPOINT`
- `SELECT`
- `SET CONSTRAINT [S]`
- `SET ROLE`
- `SET TRANSACTION`
- `storage_clause`
- `TRUNCATE`
- `UPDATE`

SAVEPOINT

用途

SAVEPOINT 文を使用すると、トランザクション内でロールバックされる位置を指定できます。

参照：

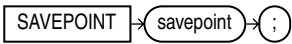
- セーブポイントの詳細は、『Oracle9i データベース概要』を参照してください。
- トランザクションのロールバックの詳細は、16-96 ページの「[ROLLBACK](#)」を参照してください。
- 現行のトランザクションの特性の設定については、17-46 ページの「[SET TRANSACTION](#)」を参照してください。

前提条件

ありません。

構文

savepoint::=



キーワードとパラメータ

savepoint

作成するセーブポイントの名前を指定します。

同一トランザクション内のセーブポイント名は、区別する必要があります。同じ識別子のセーブポイントを指定した場合、最初のセーブポイントは消去されます。セーブポイントを作成した後は、処理の継続、作業のコミット、トランザクション全体のロールバックまたはセーブポイントまでのロールバックができます。

例

デモ表 `hr.employees` の Banda と Greene の給与を更新するために、部門の給与合計が 314,000 ドルを超えていないことを確認してから、次のように Greene の給与を再入力します。

```
UPDATE employees
  SET salary = 7000
  WHERE last_name = 'Banda';
SAVEPOINT banda_sal;

UPDATE employees
  SET salary = 12000
  WHERE last_name = 'Greene';
SAVEPOINT greene_sal;

SELECT SUM(salary) FROM employees;

ROLLBACK TO SAVEPOINT banda_sal;

UPDATE employees
  SET salary = 11000
  WHERE last_name = 'Greene';

COMMIT;
```

SELECT

用途

SELECT 文または副問合せを使用すると、1 つ以上の表、オブジェクト表、ビュー、オブジェクト・ビューまたはマテリアライズド・ビューからデータを取り出すことができます。

注意： SELECT 文の結果（またはその一部）が既存のマテリアライズド・ビューと同じ場合、そのマテリアライズド・ビューを SELECT 文で指定した 1 つ以上の表のかわりに使用できます。このような置換を**クエリー・リライト**といいます。これは、コストの最適化が使用可能で、QUERY_REWRITE_ENABLED パラメータが TRUE に設定されている場合にのみ行われます。クエリー・リライトが行われるかどうかを確認する場合は、EXPLAIN PLAN 文を使用してください。

参照：

- 問合せおよび副問合せの詳細は、[第 7 章「SQL 問合せおよびその他の SQL 文」](#)を参照してください。
- マテリアライズド・ビューおよびクエリー・リライトの詳細は、『[Oracle9i データ・ウェアハウス・ガイド](#)』を参照してください。
- 16-23 ページの「[EXPLAIN PLAN](#)」を参照してください。

前提条件

表またはマテリアライズド・ビューからデータを選択する場合、表またはマテリアライズド・ビューが自分のスキーマ内にあるか、またはその表またはマテリアライズド・ビューに対する SELECT 権限が必要です。

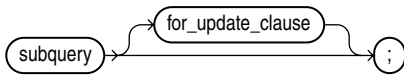
ビューの実表から行を選択する場合、次の条件を 2 つとも満たしている必要があります。

- ビューに対する SELECT 権限を持っている。
- ビューを含むスキーマの所有者が、実表に対する SELECT 権限を持っている。

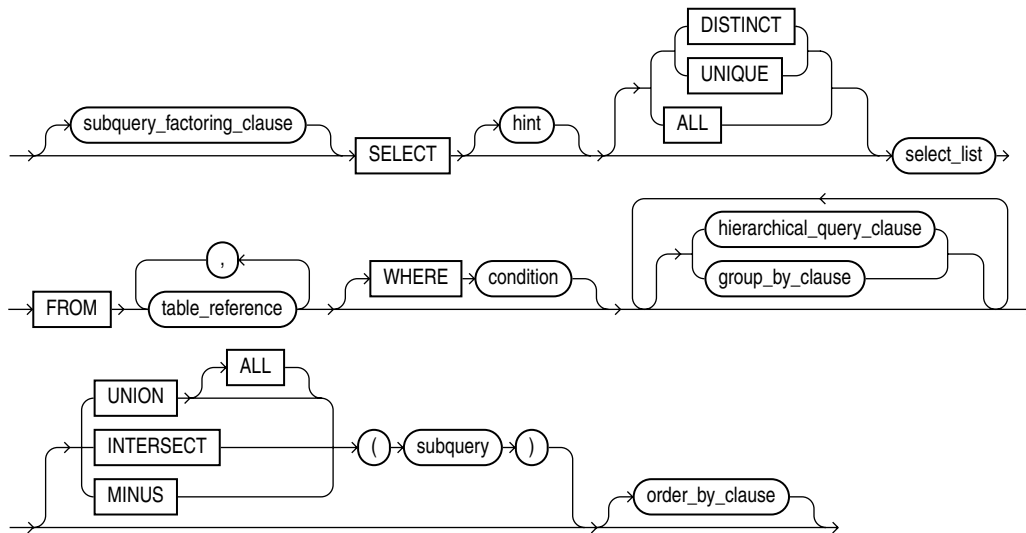
SELECT ANY TABLE システム権限を持っている場合、任意の表、マテリアライズド・ビューまたはビューの実表からデータを選択できます。

構文

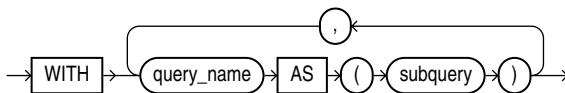
select::=



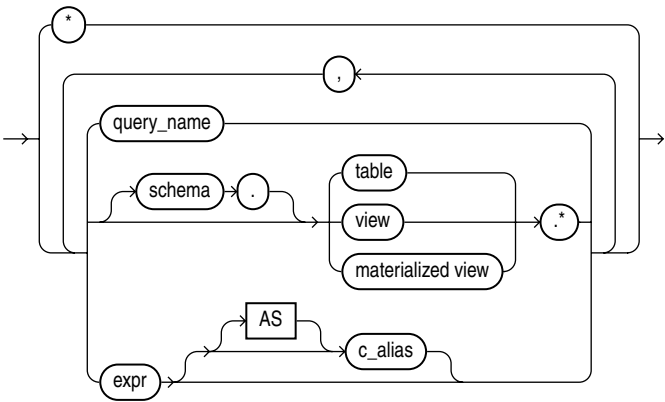
subquery::=



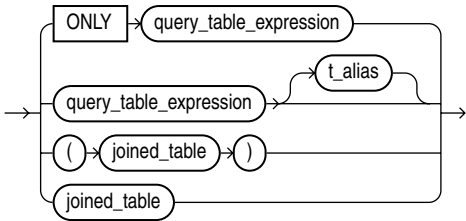
subquery_factoring_clause::=



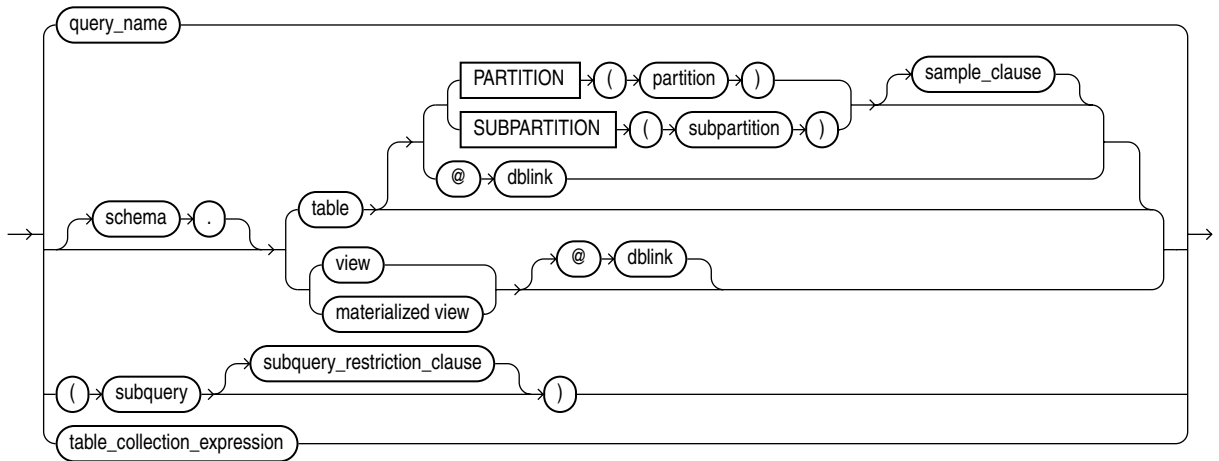
select_list::=



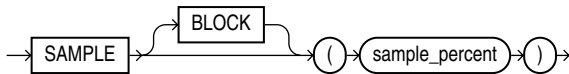
table_reference::=



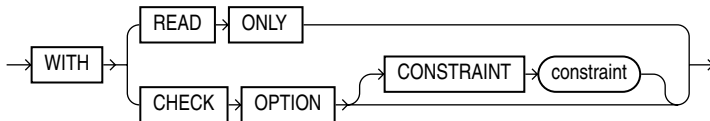
query_table_expression::=



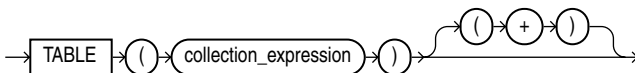
sample_clause::=



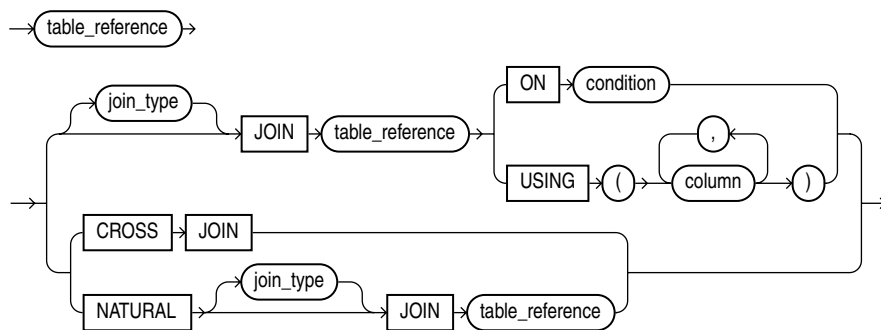
subquery_restriction_clause::=



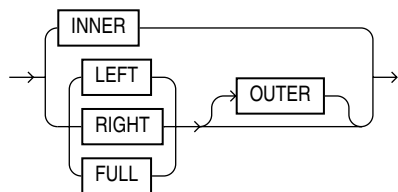
table_collection_expression::=



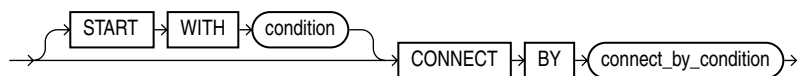
joined_table::=



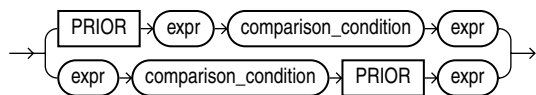
join_type::=



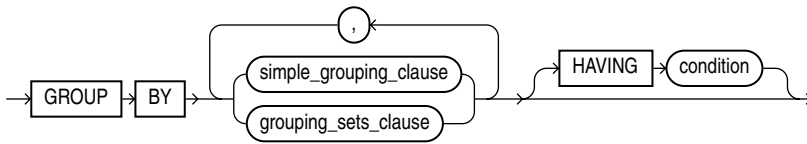
hierarchical_query_clause::=



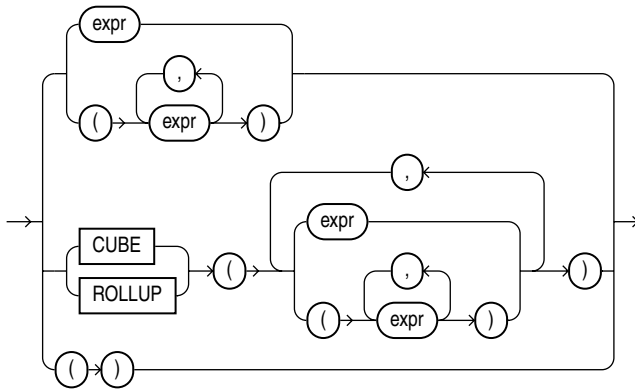
connect_by_condition::=



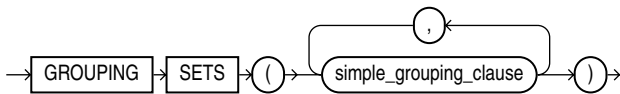
group_by_clause::=



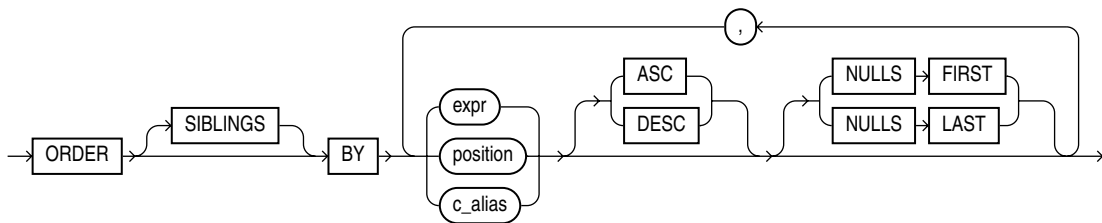
simple_grouping_clause::=



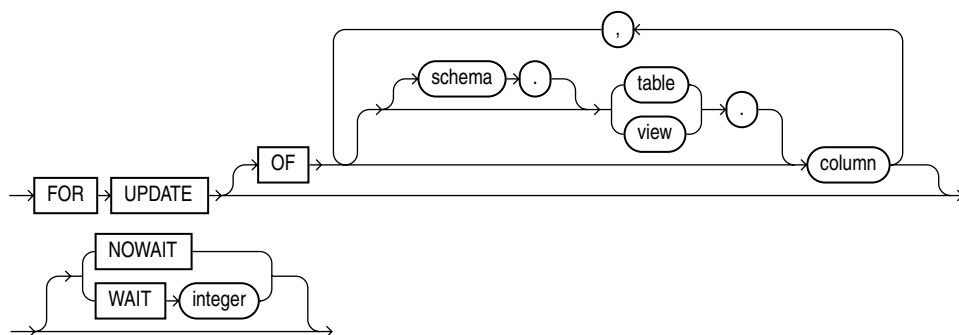
grouping_sets_clause::=



order_by_clause::=



for_update_clause::=



キーワードとパラメータ

subquery_factoring_clause

subquery_factoring_clause (WITH *query_name*) を指定すると、副問合せのブロックに名前を割り当てることができます。問合せの名前を指定することによって、問合せに複数存在する副問合せブロックを参照することができます。問合せの名前をインライン・ビューまたは一時表として扱うことによって、問合せが最適化されます。

最上位の SELECT 文およびほとんどの副問合せでこの句を指定できます。問合せの名前は、後続のすべての副問合せ（自身の問合せ名を定義する副問合せを除く）および主問合せから参照できます。

制限事項：

- この句はネストできません。つまり、他の *subquery_factoring_clause* の副問合せとして *subquery_factoring_clause* を指定できません。
- 集合演算子を指定した問合せでは、集合演算子の副問合せでは *subquery_factoring_clause* を指定できませんが、FROM 副問合せでは *subquery_factoring_clause* を指定することができます。

参照：

- インライン・ビューの詳細は、『Oracle9i データベース概要』を参照してください。
- 副問合せのファクタリング機能の使用については、『Oracle9i データ・ウェアハウス・ガイド』および『Oracle9i アプリケーション開発者ガイド - 基礎編』を参照してください。

hint

文に対して実行計画を選択する際の、オブティマイザへ指示を渡すコメントを指定します。

参照： ヒントの構文および説明については、2-86 ページの「[ヒント](#)」および『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。

DISTINCT | UNIQUE

選択された行に重複行の 1 行のみを戻す場合に、DISTINCT または UNIQUE（これらの 2 つのキーワードは同義）を指定します。重複行とは、SELECT 構文のリスト中のそれぞれの式で一致する値を持つ行のことです。

制限事項：

- DISTINCT または UNIQUE を指定する場合、SELECT 構文のリスト中の式すべての総バイト数は、データ・ブロックのサイズからオーバーヘッド分を引いたサイズに制限されます。このサイズは、初期化パラメータ DB_BLOCK_SIZE によって指定されます。
- `select_list` に LOB 列が含まれている場合、DISTINCT は指定できません。

ALL

重複行を含め、選択されたすべての行を戻す場合に、ALL を指定します。デフォルトは ALL です。

★

FROM 句に指定されているすべての表、ビューまたはマテリアライズド・ビューのすべての列を選択する場合に、アスタリスクを指定します。

注意： 表から選択する（FROM 句に、ビューやマテリアライズド・ビューではなく表を指定する）場合、ALTER TABLE SET UNUSED 文によって UNUSED のマークが付けられた列は選択されません。

参照： 10-2 ページの「[ALTER TABLE](#)」を参照してください。

select_list

select_list を指定すると、データベースから取り出す列を指定できます。

query_name

query_name には、*subquery_factoring_clause* ですでに指定されている名前を指定します。SELECT 構文のリストで *query_name* を指定するためには、*subquery_factoring_clause* を指定する必要があります。

table.* | view.* | materialized view.*

指定した表、ビューまたはマテリアライズド・ビューのすべての列を選択する場合に、ピリオドおよびアスタリスクの後にオブジェクト名を指定します。結合とは、2 つ以上の表、ビューまたはマテリアライズド・ビューの行を選択する問合せです。

他のユーザーのスキーマの表、ビューまたはマテリアライズド・ビューから選択する場合には、スキーマ修飾子を使用します。*schema* を指定しない場合、この表、ビューおよびマテリアライズド・ビューは自分のスキーマ内にあるとみなされます。

参照： 7-9 ページの「[結合](#)」を参照してください。

expr

選択する情報を表す式を指定します。リスト中の列が含まれている表、ビューまたはマテリアライズド・ビューが FROM 句で *schema* 名で指定されている場合のみ、その列名を *schema* 名で指定できます。

c_alias 列式に他の名前（別名）を指定します。この別名は、列のヘッダーで使います。AS キーワードはオプションです。別名によって、問合せ中に SELECT 構文のリストの項目名を効果的に変更できます。問合せにおいて、別名は *order_by_clause* で使用できますが、他の句では使用できません。

参照： *expr* の構文については、4-2 ページの「[SQL 式](#)」を参照してください。

***select_list* の制限事項**

- この文に [group_by_clause](#) も指定されている場合、この SELECT 構文のリストには次の式のみ指定できます。
 - 定数
 - 集計ファンクション、USER ファンクション、UID ファンクションおよび SYSDATE ファンクション
 - *group_by_clause* に指定されているものと同じ式
 - グループ内のすべての行が同じ値に評価される前述の式を伴っている式

- 結合内のキー保存表が1つのみの場合、結合ビューから ROWID を選択することができます。表の ROWID がビューの ROWID になります。

参照： キー保存表の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。

- 複数の表に同じ名前の列がある場合、表の名前でその列名を修飾する必要があります。

FROM 句

FROM 句を指定すると、どのオブジェクトからデータを選択するかを指定できます。

query_table_expression

query_table_expression を使用すると、表、ビュー、マテリアライズド・ビュー、またはパーティション、あるいはオブジェクトを識別する副問合せの指定を識別することができます。

ONLY ONLY 句は、ビューのみに適用されます。FROM 句のビューがビューの階層に属し、サブビューに行を含まない場合は、ONLY 句を使用します。

PARTITION | SUBPARTITION PARTITION または SUBPARTITION には、データを取り出すパーティションまたはサブパーティションを指定します。*partition* パラメータには、データ検索対象の *table* の中のパーティションの名前を指定するか、または検索を表の1つのパーティションのみに限定するより複雑な述語を指定できます。

dblink *dblink* には、表、ビューまたはマテリアライズド・ビューが存在するリモート・データベースのデータベース・リンクの完全名または部分名を指定します。このデータベースは、Oracle のデータベースである必要はありません。

参照：

- データベース・リンクの参照方法の詳細は、2-113 ページの「[リモート・データベース内のオブジェクトの参照](#)」を参照してください。
- 分散問合せの詳細は、7-15 ページの「[分散問合せ](#)」を参照してください。

dblink を指定しない場合、その表、ビューまたはマテリアライズド・ビューは、ローカル・データベースに存在するものとみなされます。

制限事項： リモート表のユーザー定義型またはオブジェクト REF を問い合わせることはできません。

table | view | materialized view table、view または materialized view には、データを選択する表、ビューまたはマテリアライズド・ビューの名前を指定します。

sample_clause

sample_clause では、表全体の行ではなく、表からランダムなサンプル行を選択します。

BLOCK BLOCK を指定した場合、ランダムな行サンプリングのかわりに、ランダムなブロック・サンプリングが行われます。

参照： 相違点については、『Oracle9i データベース概要』を参照してください。

sample_percent sample_percent は、サンプルに含まれているとみなされる行またはブロックの割合（パーセント）を指定する数字です。値は .000001 ～ 99 の範囲内である必要があります。

sample_clause の制限事項

- 1 つの表から選択する問合せでのみ SAMPLE を指定できます。結合はサポートされません。ただし、CREATE TABLE ...AS SELECT を使用して、基礎となる表のサンプルをマテリアライズし、新しく作成されたサンプルを参照するように元の問合せを書きなおしても、同じ結果になります。他の表に対してサンプルをマテリアライズするために、追加問合せを書き込むことができます。

参照： 17-25 ページの「[SAMPLE の例](#)」を参照してください。

- SAMPLE を指定した場合、自動的にコストベースのオプティマイザが使用されます。ルールベースのオプティマイザは、この句ではサポートされません。

注意： 統計的に適切でない想定値でこの機能を使用した場合、不正確な、または望ましくない結果になります。

subquery_restriction_clause 副問合せを次のように制限する場合に、subquery_restriction_clause を使用します。

WITH READ ONLY 副問合せが更新禁止であることを示す場合に、WITH READ ONLY を指定します。

WITH CHECK OPTION WITH CHECK OPTION は、INSERT、UPDATE または DELETE 文で、表のかわりに副問合せが使用された場合に、副問合せに含めることができない行を生成する表変更を禁止します。

参照： 17-30 ページの「[WITH CHECK OPTION の例](#)」を参照してください。

table_collection_expression

問合せおよび DML 操作で、*collection_expression* 値を表として扱う場合に、*table_collection_expression* を指定します。*collection_expression* は副問合せ、列、組み込みファンクションまたはコレクション・コンストラクタにすることができます。その形式にかかわらず、集合値（ネストした表型または VARRAY 型の値）を戻す必要があります。このようなコレクションの要素抽出プロセスを **コレクション・ネスト解除** といいます。

注意： 以前のリリースの Oracle では、*collection_expression* が副問合せの場合、*table_collection_expression* を「THE subquery」と表現していました。現在、このような表現方法はされていません。

collection_expression は、FROM 句で左側に定義された表の列を参照できます。これを **左相関** といいます。左相関は *table_collection_expression* のみで行われます。その他の副問合せは、その副問合せ以外で定義された列を参照することはできません。

オプションの (+) を使用した場合、コレクションが NULL または空である場合、すべてのフィールドに NULL が設定された行を *table_collection_expression* が戻すように指定できます。この (+) は *collection_expression* が左相関を使用する場合にのみ有効です。結果は、外部結合の結果と似ています。

参照：

- 7-11 ページの「[外部結合](#)」を参照してください。
- 17-35 ページの「[コレクション・ネスト解除の例](#)」を参照してください。

t_alias

表、ビュー、または問合せを評価するための副問合せの **相関名**（別名）を指定します。相関名は、相関問合せ内で頻繁に使用されます。表、ビューまたはマテリアライズド・ビューを参照する問合せでは、この別名を参照する必要があります。

注意： *query_table_expr_clause* がオブジェクト型の属性またはオブジェクト型のメソッドを参照する場合、この別名が必要です。

joined_table

joined_table 構文を使用すると、データが選択され、結合の一部となる表を識別できます。

参照： 結合の詳細は、7-9 ページの「[結合](#)」を参照してください。

join_type *join_type* には、実行する結合の種類を指定します。

- INNER を指定すると、内部結合が明示的に実行されます。これはデフォルトです。
- RIGHT を指定すると、右側外部結合が実行されます。
- LEFT を指定すると、左側外部結合が実行されます。
- FULL を指定すると、完全な外部結合または両側外部結合が実行されます。内部結合に加え、内部結合の結果が戻されない両方の表からの行は、保持され、NULL で拡張されます。
- RIGHT、LEFT または FULL の後にオプションの OUTER キーワードを指定し、外部結合の実行を明示的に示すことができます。

JOIN JOIN キーワードは、結合の実行を明示的に示します。この構文を使用すると、Oracle の結合で使用する、カンマで区切った表の式を ANSI の構文に置き換えることができます。

ON condition ON 句を使用すると、結合条件を指定できます。これによって、WHERE 句の検索またはフィルタ条件とは分離して結合条件を指定できます。

USING column 両方の表で同じ名前の列同士を等価結合する場合、USING *column* 句に使用する列を指定します。両方の表で同じ名前の列同士を結合する場合のみ、この句を使用できます。列名を表の名前および別名で修飾しないでください。

制限事項： LOB 列またはコレクション列は、USING *column* 句で指定できません。

CROSS JOIN CROSS キーワードは、クロス結合の実行を示します。クロス結合は、2 つの関係のクロス積を生成します。カンマで区切った Oracle の表記法と基本的に同じです。

NATURAL JOIN NATURAL キーワードは、自然結合の実行を示します。自然結合は、2 つの表の間で同じ名前のすべての列に基づきます。2 つの表から関連する列の値が等しい行を選択します。自然結合で使用する列を指定する場合は、表の名前または別名で列名を修飾しないでください。

制限事項： LOB 列またはコレクション列は、自然結合の一部として指定できません。

クロス結合および自然結合に関する注意事項

自然結合またはクロス結合の表の組合せが不明瞭な場合があります。たとえば、次のような場合です。

```
a NATURAL LEFT JOIN b LEFT JOIN c ON b.c1 = c.c1
```

次のいずれかの方法に解析されます。

```
a NATURAL LEFT JOIN (b LEFT JOIN c ON b.c1 = c.c1)
```

```
(a NATURAL LEFT JOIN b) LEFT JOIN c ON b.c1 = c.c1
```

このような不明瞭さを回避するには、カッコを使用して結合する表の組合せを指定してください。このようなカッコがないと、左から右へ表が組み合せられ、左の結合が使用されます。

WHERE condition

WHERE 条件を指定すると、選択する行を 1 つ以上の条件を満たす行のみに制限することができます。condition には、有効な SQL 条件を指定します。

参照： 条件の構文については、[第 5 章「条件」](#)を参照してください。

この句を省略した場合、FROM 句に指定されている表、ビューまたはマテリアライズド・ビューのすべての行が戻されます。

注意： この句がパーティション表または索引の DATE 列を参照する場合、Oracle は次の 2 つの条件を満たす場合にのみパーティション・プルーニングを行います。

1. 4 桁書式マスクの TO_DATE ファンクションを使用して年を完全に指定し、表または索引パーティションを作成する。
2. 2 または 4 桁書式マスクの TO_DATE ファンクションを使用して、問合せの where_clause に日付を指定する。

参照： 17-25 ページの「[PARTITION の例](#)」を参照してください。

hierarchical_query_clause

hierarchical_query_clause では、階層順序で行を選択できます。階層問合せの詳細は、7-3 ページの「[階層問合せ](#)」を参照してください。

階層問合せを含む SELECT 文では、LEVEL 疑似列を使用できます。LEVEL は、ルート・ノードには 1 を、ルート・ノードの子であるノードには 2 を、孫であるノードには 3 を戻します（以下同様）。階層問合せによって戻されるレベルの数値は、使用可能なユーザー・メモリーによって制限されます。

START WITH 句

階層問合せのルートとして使用される行を識別する場合に条件を指定します。Oracle では、この条件を満たすすべての行がルートで使用されます。この句を省略した場合、表内のすべての行がルート行として使用されます。START WITH 条件には、副問合せを含めることができますが、スカラー副問合せ式を含めることはできません。

CONNECT BY 句

階層の親 / 子の行の関連を識別する条件を指定します。*connect_by_condition* には、[第 5 章「条件」](#) のすべての条件を含めることができます。ただし、親である行を参照するための PRIOR 演算子を使用する必要があります。

制限事項：*connect_by_condition* には、正規の副問合せおよびスカラー副問合せ式を含めることはできません。

参照：

- LEVEL の詳細は、2-79 ページの「[疑似列](#)」を参照してください。
- 階層問合せの概要については、7-3 ページの「[階層問合せ](#)」を参照してください。

階層問合せに関する注意事項

ORDER BY 句とともに階層問合せを指定する場合、ORDER BY 句に SIBLINGS キーワードを指定しないかぎり、ORDER BY 句は、階層問合せによって指定される順序より優先されます。

階層問合せでの WHERE 句の Oracle の処理は、WHERE 句が結合を含むかどうかによって異なります。

- WHERE 述語が結合を含む場合、Oracle は、CONNECT BY 処理の前に述語結合を適用します。
- すべての述語に結合を含む WHERE 句がない場合、Oracle は、CONNECT BY 処理の後に、非述語結合を適用します。この場合、階層の他の行に影響はありません。

group_by_clause

それぞれの行の *expr* の値に基づいて選択した行をグループ化し、各グループのサマリー情報を 1 行戻す場合に、GROUP BY 句を使用します。この句に CUBE または ROLLUP 拡張要素が指定された場合、標準グループ化の他に超集合グループ化が生成されます。

GROUP BY 句の式には、SELECT 構文のリストに指定されている列であるかどうかにかかわらず、FROM 句の表、ビューおよびマテリアライズド・ビューの列を指定できます。

参照：

- データを集約化する SQL グループ化構文の詳細および例については、『Oracle9i データ・ウェアハウス・ガイド』を参照してください。
- この句の例については、6-64 ページの「[GROUP_ID](#)」、[GROUPING](#) および [GROUPING_ID](#) ファンクションを参照してください。

ROLLUP *simple_grouping_clause* の ROLLUP 操作は、GROUP BY で指定した式 *n*、*n*-1、*n*-2、... 0 の最初の値に基づいて選択した行をグループ化します。また、それぞれのグループのサマリー情報を 1 行戻します。ROLLUP 操作を SUM ファンクションとともに使用して、**小計値**を出力できます。ROLLUP を SUM とともに使用すると、最も詳細なレベルの小計から総計までが生成されます。COUNT などの集計ファンクションは、他の種類の超集合の出力に使用できます。

たとえば、*simple_grouping_clause* の ROLLUP 句に式を 3 つ指定した場合 (*n*=3)、操作の結果は $n+1=3+1=4$ グループになります。

最初の '*n*' 式の値でグループ化した行を**標準行**、その他を**超集合行**といいます。

CUBE *simple_grouping_clause* の CUBE 操作は、指定した式のあらゆる組合せの値に基づいて選択した行をグループ化します。また、それぞれのグループのサマリー情報を 1 行戻します。CUBE 操作を使用して、**クロス集計値**を出力できます。

たとえば、*simple_grouping_clause* の CUBE 句に式を 3 つ指定した場合 (*n*=3)、操作の結果は $2^n = 2^3 = 8$ グループになります。'*n*' 式の値でグループ化した行を**標準行**、その他を**超集合行**といいます。

参照： 17-26 ページの「[CUBE の例](#)」を参照してください。

GROUPING SETS GROUPING SETS は、データを複数にグループ化する GROUP BY 句をさらに拡張したものです。これは、効果的な集計に役立ちます。必要なグループを指定すると、Oracle が CUBE または ROLLUP によって生成された集計のすべてを実行する必要がなくなります。GROUPING SETS 句で指定したすべてのグループ化が計算され、UNION ALL 操作で個々のグループ化の結果が組み合わせられます。UNION ALL は、結果セットが重複行を含むことを許可します。

GROUP BY 句では、様々な方法で式を組み合わせることができます。

- **複合列**を指定するには、カッコで列をグループ化します。Oracle は、ROLLUP 操作または CUBE 操作の計算でこれらを一単位として処理します。
- **グルーピング・セットの連結**を指定するには、カンマで複数のグルーピング・セット、ROLLUP 操作および CUBE 操作を分割します。Oracle は、これらを 1 つの GROUP BY 句に結合します。結果は、各グルーピング・セットからのグループ化のクロス積です。

参照： 17-26 ページの「[GROUPING SETS の例](#)」を参照してください。

HAVING 句

戻す行のグループを、指定した条件が真である行のグループのみに制限する場合に、HAVING 句を使用します。この句を省略した場合、すべてのグループのサマリー行が戻されます。

where_clause および CONNECT BY 句の後に、GROUP BY および HAVING を指定します。GROUP BY および HAVING 句を両方指定する場合は、どちらを先に指定してもかまいません。

制限事項： HAVING 条件に、スカラー副問合せ式を含めることはできません。

GROUP BY 句の制限事項

- *group_by_clause* には最大 255 個の式を指定できます。
- 式には、スカラー副問合せ式を除くすべての形式が可能です。
- LOB 列、ネストした表または VARRAY を *expr* の一部として指定できません。
- *group_by_clause* に指定したすべての式の総バイト数は、データ・ブロックのサイズ（初期化パラメータ DB_BLOCK_SIZE によって指定されたもの）からオーバーヘッドを引いた値になります。
- *group_by_clause* がオブジェクト列を参照する場合、問合せはパラレル化されません。

参照： *expr* の構文については、4-2 ページの「[SQL 式](#)」を、条件の構文については、[第 5 章「条件」](#)を参照してください。

集合演算子：UNION、UNION ALL、INTERSECT および MINUS

これらの集合演算子は、2 つの SELECT 文によって戻された行を 1 つの結果に結合します。それぞれのコンポーネント問合せで選択される列の数とデータ型は同じである必要がありますが、列の長さは異なってもかまいません。

集合演算子で 2 つ以上の問合せを結合する場合、隣接する問合せを左から右へと評価します。この評価順序を変更する場合、カッコを使用します。

参照： これらの演算子の詳細は、7-7 ページの「[UNION \[ALL\]、INTERSECT および MINUS 演算子](#)」を参照してください。

集合演算子の制限事項

- 集合演算子は、データ型が BLOB、CLOB、BFILE、VARRAY またはネストした表である列に対しては無効になります。
- UNION、INTERSECT および MINUS 演算子は、LONG 列に対しては無効になります。
- 列を参照する場合は、別名を使用して列に名前を付ける必要があります。
- *for_update_clause* は、これらの集合演算子とともに指定できません。
- これらの演算子の副問合せには、*order_by_clause* を指定できません。
- TABLE コレクション式を含む SELECT 文では、これらの演算子を使用できません。

注意： SQL 規格に準拠するために、Oracle の今後のリリースでは、他の集合演算子より優先順位の高い INTERSECT 演算子が提供されます。したがって、INTERSECT 演算子と他の集合演算子を使用する問合せでは、カッコを使用して評価順序を指定してください。

order_by_clause

ORDER BY 句を使用して、文によって戻された行を順序付けます。*order_by_clause* を指定しない場合、同じ問合せで取り出される行の順序が異なることがあります。

SIBLINGS SIBLINGS キーワードは、*hierarchical_query_clause* (CONNECT BY) を指定する場合のみに有効です。Oracle は階層問合せ句で指定した任意の順序を保持し、兄弟関係の階層に *order_by_clause* を適用します。

expr *expr* は、*expr* の値を基準にして行に順番を付けます。式は、SELECT 構文のリストの列、FROM あるいはビューまたはマテリアライズド・ビューの列に基づきます。

position *position* は、SELECT 構文のリストのその位置にある式の値に基づいて行を順序付けます。*position* には整数を指定する必要があります。

参照： 問合せ結果の順位付けの詳細は、7-9 ページの「[問合せ結果のソート](#)」を参照してください。

order_by_clause には複数の式を指定できます。この場合、まず、最初の式の値に基づいて行がソートされ、次に、最初の式と同じ値を持つ行が 2 番目の式の値に基づいてソートされる、というように処理が行われます。NULL 値は昇順では最後に、降順では先頭にソートされます。

ASC | DESC 昇順か降順かを指定します。デフォルトは ASC です。

NULLS FIRST | NULLS LAST NULL 値を含む戻された行が順序の最初にくるか、最後にくるかを指定します。

NULLS LAST は昇順のデフォルトで、NULLS FIRST は降順のデフォルトです。

***order_by_clause* の制限事項**

- この文中で DISTINCT 演算子を指定した場合、SELECT 構文のリストに指定された列でないかぎり、この句は列を参照することはできません。
- *order_by_clause* には最大 255 個の式を指定できます。
- LOB 列、ネストした表または VARRAY を使用しての順位付けはできません。
- 同じ文中で *group_by_clause* を指定する場合、この *order_by_clause* は次の式に制限されます。
 - 定数
 - 集計ファンクション
 - 分析ファンクション
 - USER ファンクション、UID ファンクションおよび SYSDATE ファンクション
 - *group_by_clause* に指定されているものと同じ式
 - グループ内のすべての行が同じ値に評価されるこれらの式を導出する式

for_update_clause

FOR UPDATE 句によって、トランザクションが終了する前に、選択した行が別のユーザーによってロックまたは更新されることがないように、選択した行をロックします。最上位の SELECT 文でのみ、この句を指定できます。副問合せでは指定できません。

LOB 値を更新する場合、その LOB を含む行をロックしておく必要があります。行をロックする方法の 1 つに、SELECT ... FOR UPDATE 文があります。

参照： 17-30 ページの「[LOB ロックの例](#)」を参照してください。

親表の行がロックされても、ネストした表の行はロックされません。ネストした表の行をロックする場合、ネストした表を明示的にロックする必要があります。

制限事項：

- この句を DISTINCT または CURSOR 演算子、集合演算子、*group_by_clause*、または集計ファンクションの構造体とともに指定することはできません。
- この句がロックした表は、同じ文で参照された LONG 列および順序と同じデータベース内にある必要があります。

OF ... column

結合内の特定の表の選択された行のみをロックする場合に、OF ... column 句を使用します。OF 句の列は、どの表またはビューの行をロックするかを識別する場合にのみ使用します。指定する列は重要ではありません。ただし、列の別名ではなく、実際の列名を指定する必要があります。この句を省略した場合、問合せ内のすべての表の選択された行がロックされます。

NOWAIT | WAIT

NOWAIT および WAIT 句は、他のユーザによってロックされている行を SELECT 文がロックしようとする場合に処理する方法を Oracle に指示します。

NOWAIT NOWAIT を指定すると、ロックされている場合に制御がただちにに戻ります。

WAIT WAIT を指定すると、行が使用可能になるまで *integer* 秒待機した後で制御が戻されます。

WAIT および NOWAIT のどちらも指定しない場合、行が使用可能になるまで待機した後で SELECT 文の結果が戻されます。

例

副問合せのファクタリングの例 次の文は、結合を含む初期問合せブロックに対する問合せの名前 dept_costs および avg_cost を作成し、主問合せの本体でその問合せの名前を使用します。

```
WITH
  dept_costs AS (
    SELECT department_name, SUM(salary) dept_total
      FROM employees e, departments d
     WHERE e.department_id = d.department_id
    GROUP BY department_name),
  avg_cost AS (
    SELECT SUM(dept_total)/COUNT(*) avg
      FROM dept_costs)
SELECT * FROM dept_costs
  WHERE dept_total >
        (SELECT avg FROM avg_cost)
   ORDER BY department_name;
```

DEPARTMENT_NAME	DEPT_TOTAL
-----	-----
Sales	313800
Shipping	156400

単純問合せの例 次の文は、部門番号 30 の従業員表 employees の行を選択します。

```
SELECT *
  FROM employees
  WHERE department_id = 30;
```

次の文は、部門番号 30 の購買係を除くすべての従業員の名前、職種、給与および部門番号を選択します。

```
SELECT last_name, job_id, salary department_id
  FROM employees
  WHERE NOT (job_id = 'PU_CLERK' AND department_id = 30);
```

次の文は、FROM 句の副問合せから、部門のすべての従業員数と給与合計がすべての部門に占める割合を算出します。

```
SELECT a.department_id "Department",
       a.num_emp/b.total_count "%_Employees",
       a.sal_sum/b.total_sal "%_Salary"
FROM
  (SELECT department_id, COUNT(*) num_emp, SUM(salary) sal_sum
   FROM employees
   GROUP BY department_id) a,
  (SELECT COUNT(*) total_count, SUM(salary) total_sal
   FROM employees) b;
```

PARTITION の例 FROM 句にキーワード PARTITION を指定することによって、パーティション表の 1 つのパーティションから行を選択できます。次の SQL 文は、デモ表 sh.sales の sales_q2_2000 パーティションへの別名の割当ておよび行の取出しを行います。

```
SELECT * FROM sales PARTITION (sales_q2_2000) s
WHERE s.amount_sold > 1000;
```

次の文は、oe.orders 表から指定した日付より前の注文の行を選択します。

```
SELECT * FROM orders
WHERE order_date < TO_DATE('1999-06-15', 'YYYY-MM-DD');
```

SAMPLE の例 次の問合せは、oe.orders 表の注文数を推定します。

```
SELECT COUNT(*) * 100 FROM orders SAMPLE BLOCK (1);
```

次の文は、デモ表 hr.employees のサンプル・サブセットを作成し、結果のサンプル表と departments で結合します。この操作によって、結合問合せで sample_clause を指定できないという制限を回避できます。

```
CREATE TABLE sample_emp AS
  SELECT employee_id, department_id FROM employees SAMPLE(10);

SELECT e.employee_id FROM sample_emp e, departments d
WHERE e.department_id = d.department_id
AND d.department_name = 'Sales';
```

GROUP BY の例 次の文は、employees 表の各部門について最低給与と最高給与を戻します。

```
SELECT department_id, MIN(salary), MAX (salary)
  FROM employees
  GROUP BY department_id;
```

次の文は、各部門の事務員について最高給与と最低給与を戻します。

```
SELECT department_id, MIN(salary), MAX (salary)
  FROM employees
 WHERE job_id = 'PU_CLERK'
  GROUP BY department_id;
```

CUBE の例 部門および職種のすべての組合せについて、従業員数と平均年収を戻すには、デモ表 hr.employees および hr.departments に次の問合せを発行します。

```
SELECT DECODE(GROUPING(department_name), 1, 'All Departments',
              department_name) AS department_name,
       DECODE(GROUPING(job_id), 1, 'All Jobs', job_id) AS job_id,
       COUNT(*) "Total Empl", AVG(salary) * 12 "Average Sal"
  FROM employees e, departments d
 WHERE d.department_id = e.department_id
  GROUP BY CUBE (department_name, job_id);
```

DEPARTMENT_NAME	JOB_ID	Total Empl	Average Sal
-----	-----	-----	-----
Accounting	AC_ACCOUNT	1	99600
Accounting	AC_MGR	1	144000
Accounting	All Jobs	2	121800
Administration	AD_ASST	1	52800
.			
.			
.			
All Departments	ST_MAN	5	87360
All Departments	All Jobs	107	77798.1308

GROUPING SETS の例 次の例は、指定した 3 つのグループで集計した販売の合計を示します。

- (channel_desc, calendar_month_desc, country_id)
- (channel_desc, country_id)
- (calendar_month_desc, country_id)

GROUPING SETS 構文を指定しない場合、SQL はより複雑になり、問合せの効果は低くなります。たとえば、3つの分散した問合せを実行し、重複を含まないすべての行を戻す、または CUBE(channel_desc, calendar_month_desc, country_id) 操作を指定した問合せを実行し、生成される 8 つのグループから 5 つを除去します。

```
SELECT channel_desc, calendar_month_desc, co.country_id,
       TO_CHAR(sum(amount_sold) , '9,999,999,999') SALES$
FROM sales, customers, times, channels, countries co
WHERE sales.time_id=times.time_id
      AND sales.cust_id=customers.cust_id
      AND sales.channel_id= channels.channel_id
      AND customers.country_id = co.country_id
      AND channels.channel_desc IN ('Direct Sales', 'Internet')
      AND times.calendar_month_desc IN ('2000-09', '2000-10')
      AND co.country_id IN ('UK', 'US')
GROUP BY GROUPING SETS(
    (channel_desc, calendar_month_desc, co.country_id),
    (channel_desc, co.country_id),
    ( calendar_month_desc, co.country_id) );
```

CHANNEL_DESC	CALENDAR	CO	SALES\$
-----	-----	--	-----
Direct Sales	2000-09	UK	1,378,126
Direct Sales	2000-10	UK	1,388,051
Direct Sales	2000-09	US	2,835,557
Direct Sales	2000-10	US	2,908,706
Internet	2000-09	UK	911,739
Internet	2000-10	UK	876,571
Internet	2000-09	US	1,732,240
Internet	2000-10	US	1,893,753
Direct Sales		UK	2,766,177
Direct Sales		US	5,744,263
Internet		UK	1,788,310
Internet		US	3,625,993
	2000-09	UK	2,289,865
	2000-09	US	4,567,797
	2000-10	UK	2,264,622
	2000-10	US	4,802,459

参照： これらのファンクションの詳細は、6-64 ページの「[GROUP_ID](#)」、[「GROUPING」](#) および [「GROUPING_ID」](#) ファンクションを参照してください。

階層問合せの例 CONNECT BY 句を指定した次の問合せは、親である行の employee_id 値が子である行の manager_id 値と等しいという階層関係を定義します。

```
SELECT last_name, employee_id, manager_id FROM employees
       CONNECT BY employee_id = manager_id;
```

次の CONNECT BY 句では、PRIOR 演算子が employee_id 値のみに適用されます。この条件を評価するために、Oracle は親である行に対しては employee_id の値を評価し、子である行に対しては manager_id、salary および commission_pct のそれぞれの値を評価します。

```
SELECT last_name, employee_id, manager_id FROM employees
       CONNECT BY PRIOR employee_id = manager_id
       AND salary > commission_pct;
```

子である行を限定する場合、manager_id の値と親である行の employee_id の値が等しく、salary の値が commission_pct の値より大きい必要があります。

HAVING の例 次の文は、従業員の最低給与が 5,000 ドル未満の部門についての最高給与と最低給与を戻します。

```
SELECT department_id, MIN(salary), MAX (salary)
       FROM employees
       GROUP BY department_id
       HAVING MIN(salary) < 5000;
```

DEPARTMENT_ID	MIN(SALARY)	MAX(SALARY)
10	4400	4400
30	2500	11000
50	2100	8200
60	4200	9000

ORDER BY の例 次の文は、employees 表から営業担当員のすべてのレコードを選択し、その歩合によって降順にソートします。

```
SELECT *
  FROM employees
 WHERE job_id = 'PU_CLERK'
 ORDER BY commission_pct DESC;
```

次の文は、employees 表から情報を選択し、最初に部門番号で昇順にソートした後、給与で降順にソートします。

```
SELECT last_name, department_id, salary
  FROM employees
 ORDER BY department_id ASC, salary DESC;
```

次の文は、先の SELECT 文と同じ情報を選択し、位置に基づく ORDER BY 句の指定を使用します。

```
SELECT last_name, department_id, salary
  FROM employees
 ORDER BY 2 ASC, 3 DESC;
```

FOR UPDATE の例 次の文は、employees 表中のオックスフォード勤務 (location_id は 2500) の購買係の行をロックし、departments 表中の購買係が存在するオックスフォードの部門の行をロックします。

```
SELECT e.employee_id, e.salary, e.commission_pct
  FROM employees e, departments d
 WHERE job_id = 'SA_REP'
 AND e.department_id = d.department_id
 AND location_id = 2500
 FOR UPDATE;
```

次の文は、employees 表中のオックスフォード勤務 (location_id は 2500) の購買係の行のみをロックします。departments 表では、行のロックはできません。

```
SELECT e.employee_id, e.salary, e.commission_pct
  FROM employees e, departments d
 WHERE job_id = 'SA_REP'
 AND e.department_id = d.department_id
 AND location_id = 2500
 FOR UPDATE OF e.salary;
```

LOB ロックの例 次の文は、SELECT ... FOR UPDATE 文を使用して、LOB 値を更新する前にその LOB が含まれている行をロックします。

```
INSERT INTO t_table VALUES (1, 'abcd');

COMMIT;

DECLARE
    num_var      NUMBER;
    clob_var      CLOB;
    clob_locked   CLOB;
    write_amount  NUMBER;
    write_offset  NUMBER;
    buffer        VARCHAR2(20) := 'efg';

BEGIN
    SELECT clob_col INTO clob_locked FROM t_table
    WHERE num_col = 1 FOR UPDATE;

    write_amount := 3;
    dbms_lob.write(clob_locked, write_amount, write_offset, buffer);
END;
```

WITH CHECK OPTION の例 次の文は、2 番目の値が副問合せ *where_clause* の条件に違反していても有効です。

```
INSERT INTO
    (SELECT employee_id, last_name, email, hire_date, job_id, salary
     FROM employees WHERE department_id < 10)
VALUES (99999, 'Taylor', 'Taylor@oracle.com',
        TO_DATE('07-JUN-99', 'DD-MON-YY'), 'PU_CLERK', 5000);
```

ただし、次の文は WITH CHECK OPTION 句を含むため、無効になります。

```
INSERT INTO
    (SELECT employee_id, last_name, email, hire_date, job_id, salary
     FROM employees WHERE department_id < 10 WITH CHECK OPTION)
VALUES (99999, 'Taylor', 'Taylor@oracle.com',
        TO_DATE('07-JUN-99', 'DD-MON-YY'), 'PU_CLERK', 5000);
insert into
    *
```

1 行でエラーが発生しました。

ORA-01402: ビューの WITH CHECK OPTION WHERE 句でエラーが発生しました。

結合の例 次の例は、問合せにおける様々な表の結合方法を示します。最初の例では、等価結合は、それぞれの従業員の名前と職種、およびその従業員が属する部門の番号と名前を戻します。

```
SELECT last_name, job_id, departments.department_id, department_name
FROM employees, departments
WHERE employees.department_id = departments.department_id;
```

LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
...			
Sciarra	FI_ACCOUNT	100	Finance
Urman	FI_ACCOUNT	100	Finance
Popp	FI_ACCOUNT	100	Finance
...			

従業員の名前および職種は部門名とは別の表に格納されているため、このデータを戻す場合は結合を使用する必要があります。次の結合条件に従って、2つの表の行が結合されます。

```
employees.department_id = departments.department_id
```

次の等価結合は、すべての販売マネージャの名前、職種、部門番号および部門名を戻します。

```
SELECT last_name, job_id, departments.department_id, department_name
FROM employees, departments
WHERE employees.department_id = departments.department_id
AND job_id = 'SA_MAN';
```

LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
...			
Russell	SA_MAN	80	Sales
Partners	SA_MAN	80	Sales
Errazuriz	SA_MAN	80	Sales
Cambrault	SA_MAN	80	Sales
Zlotkey	SA_MAN	80	Sales
...			

この問合せは、次の *where_clause* を使用して 'SA_MAN' という job 値を持つ行のみを戻すこと以外は、前述の例と同じです。

副問合せの例 次の文は、従業員 'Lorentz' と同じ部門で働く従業員を判断します。

```
SELECT last_name, department_id FROM employees
WHERE department_id =
  (SELECT department_id FROM employees
   WHERE last_name = 'Lorentz');
```

employees 表の職種を変更した (job_history 表に示される) すべての従業員の給与を 10% アップする場合、次の文を発行します。

```
UPDATE employees
SET salary = salary * 1.1
WHERE employee_id IN (SELECT employee_id FROM job_history);
```

departments 表から 3 つの列だけを伴って新しい表 new_departments を作成するには、次の文を発行します。

```
CREATE TABLE new_departments
(department_id, department_name, location_id)
AS SELECT department_id, department_name, location_id
FROM departments;
```

自己結合の例 次の問合せは、自己結合を使用して、それぞれの従業員の名前と一緒にその従業員の上司の名前を戻します (出力を短くするために WHERE 句は追加されています)。

```
SELECT e1.last_name||' works for '||e2.last_name
"Employees and Their Managers"
FROM employees e1, employees e2
WHERE e1.manager_id = e2.employee_id
AND e1.last_name LIKE 'R%';
```

```
Employees and Their Managers
-----
Rajs works for Mourgos
Raphaely works for King
Rogers works for Kaufling
Russell works for King
```

この問合せの結合条件では、サンプル表 employees に対する別名 e1 および e2 を使用します。

```
e1.manager_id = e2.employee_id
```

外部結合の例 次の例は、左側外部結合を使用し、従業員がいない場合も含めてすべての部門名を戻します。

```
SELECT d.department_id, e.last_name
FROM departments d LEFT OUTER JOIN employees e
ON d.department_id = e.department_id
ORDER BY d.department_id;
```

これは、次に示す以前の Oracle の外部結合の構文と同じ問合せです。

```
SELECT d.department_id, e.last_name
FROM departments d, employees e
WHERE d.department_id(+) = e.department_id
ORDER BY d.department_id;
```

これよりも、その前の例で示した、より柔軟性が高い Oracle9i の ANSI 互換の構文を使用することをお勧めします。

次の例は、右側外部結合を使用し、そこを本拠地とする部門がない場合も含めてすべての勤務地名を戻します。

```
SELECT d.department_name, d.manager_id, l.city
FROM departments d RIGHT OUTER JOIN locations l
ON d.location_id = l.location_id
ORDER BY d.department_name;
```

次の問合せは完全な外部結合を使用し、customers 表からすべての行を戻し、orders 表からもすべての行を戻します。ON 条件を満たさない行は NULL で拡張されます。

```
SELECT c.customer_id, c.o.order_id, c.account_mgr_id, o.sales_rep_id
FROM customers c FULL OUTER JOIN orders o
ON c.customer_id = o.customer_id
ORDER BY c.customer_id;
```

```
CUSTOMER_ID    ORDER_ID ACCOUNT_MGR_ID SALES_REP_ID
```

```
-----
```

.			
.			
.			
133			149
134			
135			
136			149
137			149
138			149
139			
140			
141	2377		145
142	2378		149

143	2380	149	
144	2435	149	159
144	2445	149	158
144	2363	149	
144	2422	149	153
144	2382	149	
145	2455	145	160
.			
.			
.			

表コレクションの例 DML 操作は、表の列として定義された場合にのみ、ネストした表で実行できます。したがって、INSERT、DELETE または UPDATE 文の *query_table_expr_clause* が *table_collection_expression* の場合、コレクション式は、表のネストした表の列を選択する副問合せである必要があります。次の例は、この使用例に基づいています。

```
CREATE TYPE ProjectType AS OBJECT(  
    pno    NUMBER,  
    pname  CHAR(31),  
    budget NUMBER);  
CREATE TYPE ProjectSet AS TABLE OF ProjectType;  
  
CREATE TABLE dept_work (dno NUMBER, dname CHAR(31), projs ProjectSet)  
    NESTED TABLE projs STORE AS  
        ProjectSetTable ((Primary Key(Nested_Table_Id, pno)) ORGANIZATION  
INDEX COMPRESS 1);  
  
INSERT INTO dept_work VALUES (1, 'Engineering', ProjectSet());
```

次の例では、Engineering 部門のネストした表 projs に挿入します。

```
INSERT INTO TABLE(SELECT d.projs  
    FROM    dept_work d  
    WHERE   d.dno = 1)  
VALUES (1, 'Collection Enhancements', 10000);
```

次の例では、Engineering 部門のネストした表 projs を更新します。

```
UPDATE TABLE(SELECT d.projs  
    FROM    dept_work d  
    WHERE   d.dno = 1) p  
SET  p.budget = p.budget + 1000;
```

次の例では、Engineering 部門のネストした表 `projs` から削除します。

```
DELETE TABLE(SELECT d.projs
                FROM   dept_work d
                WHERE  d.dno = 1) p
WHERE p.budget > 100000;
```

コレクション・ネスト解除の例 データベースに、`dept` 列、`location` 列、`mgr` 列を持つ `hr_info` 表と、`name` 列、`dept` 列および `sal` 列を持つネストした表型 `people` の列が含まれていると仮定します。次の文を使用して、`hr_info` および `people` のすべての列を選択できます。

```
SELECT t1.dept, t2.* FROM hr_info t1, TABLE(t1.people) t2
WHERE t2.dept = t1.dept;
```

`people` は、`hr_info` のネストした表の列ではなく、`name`、`dept`、`address`、`hiredate` および `sal` 列と別の表であると仮定します。次の文を使用して、前述の例と同じ行を抽出できます。

```
SELECT t1.department, t2.*
FROM hr_info t1, TABLE(CAST(MULTISET(
    SELECT t3.name, t3.dept, t3.sal FROM people t3
    WHERE t3.dept = t1.dept)
AS NESTED_PEOPLE)) t2;
```

最後に、`people` は `hr_info` 表のネストした表の列でも、表そのものでもないと仮定します。かわりに、すべての従業員の名前、部門および給与を様々な情報から抽出する `people_func` ファンクションを作成しておきます。次の問合せを使用して、前述の例と同様の情報を得ることができます。

```
SELECT t1.dept, t2.* FROM hr_info t1, TABLE(CAST
    (people_func( ... ) AS NESTED_PEOPLE)) t2;
```

参照： コレクション・ネスト解除の詳細は、『Oracle9i アプリケーション 開発者ガイド - 基礎編』を参照してください。

LEVEL の例 次の文を実行すると、すべての従業員が階層順序で戻されます。職種が 'AD_VP' である従業員がルート行となるように定義されています。また、親である行の従業員番号が上司の従業員番号となるように、親である行の子である行が定義されています。

```
SELECT LPAD(' ',2*(LEVEL-1)) || last_name org_chart,
       employee_id, manager_id, job_id
FROM employees
START WITH job_id = 'AD_VP'
CONNECT BY PRIOR employee_id = manager_id;
```

ORG_CHART	EMPLOYEE_ID	MANAGER_ID	JOB_ID
-----	-----	-----	-----
Kochhar	101	100	AD_VP
Greenberg	108	101	FI_MGR
Faviet	109	108	FI_ACCOUNT
Chen	110	108	FI_ACCOUNT
Sciarra	111	108	FI_ACCOUNT
Urman	112	108	FI_ACCOUNT
Popp	113	108	FI_ACCOUNT
Whalen	200	101	AD_ASST
Mavris	203	101	HR_REP
Baer	204	101	PR_REP
Higgins	205	101	AC_MGR
Gietz	206	205	AC_ACCOUNT
De Haan	102	100	AD_VP
Hunold	103	102	IT_PROG
Ernst	104	103	IT_PROG
Austin	105	103	IT_PROG
Pataballa	106	103	IT_PROG
Lorentz	107	103	IT_PROG

次の文は、前述の例とほぼ同じですが、職種が 'FI_MGR' である従業員は選択されません。

```
SELECT LPAD(' ', 2*(LEVEL-1)) || last_name org_chart,
       employee_id, manager_id, job_id
FROM employees
WHERE job_id != 'FI_MGR'
START WITH job_id = 'AD_VP'
CONNECT BY PRIOR employee_id = manager_id;
```

ORG_CHART	EMPLOYEE_ID	MANAGER_ID	JOB_ID
-----	-----	-----	-----
Kochhar	101	100	AD_VP
Faviet	109	108	FI_ACCOUNT
Chen	110	108	FI_ACCOUNT
Sciarra	111	108	FI_ACCOUNT
Urman	112	108	FI_ACCOUNT
Popp	113	108	FI_ACCOUNT
Whalen	200	101	AD_ASST
Mavris	203	101	HR_REP
Baer	204	101	PR_REP
Higgins	205	101	AC_MGR
Gietz	206	205	AC_ACCOUNT
De Haan	102	100	AD_VP
Hunold	103	102	IT_PROG
Ernst	104	103	IT_PROG
Austin	105	103	IT_PROG
Pataballa	106	103	IT_PROG
Lorentz	107	103	IT_PROG

Greenberg が管理する従業員は戻されますが、マネージャ Greenberg は戻されません。

次の文も、前述の例と同じですが、LEVEL 疑似列を使用して管理階層の最初の 2 つのレベルのみが選択されます。

```
SELECT LPAD(' ',2*(LEVEL-1)) || last_name org_chart,
employee_id, manager_id, job_id
FROM employees
START WITH job_id = 'AD_PRES'
CONNECT BY PRIOR employee_id = manager_id AND LEVEL <= 2;
```

ORG_CHART	EMPLOYEE_ID	MANAGER_ID	JOB_ID
-----	-----	-----	-----
King	100		AD_PRES
Kochhar	101	100	AD_VP
De Haan	102	100	AD_VP
Raphaely	114	100	PU_MAN
Weiss	120	100	ST_MAN
Fripp	121	100	ST_MAN
Kaufling	122	100	ST_MAN
Vollman	123	100	ST_MAN
Mourgos	124	100	ST_MAN
Russell	145	100	SA_MAN
Partners	146	100	SA_MAN
Errazuriz	147	100	SA_MAN
Cambrault	148	100	SA_MAN
Zlotkey	149	100	SA_MAN
Hartstein	201	100	MK_MAN

分散問合せの例 次の文は、ローカル・データベース上の departments 表と、houston データベース上の employees 表を結合します。

```
SELECT last_name, department_name
FROM employees@houston, departments
WHERE employees.department_id = departments.department_id;
```

相関副問合せの例 次に、相関副問合せの構文の一般的な例を示します。

```
SELECT select_list
  FROM table1 t_alias1
 WHERE expr operator
       (SELECT column_list
         FROM table2 t_alias2
         WHERE t_alias1.column
              operator t_alias2.column);

UPDATE table1 t_alias1
  SET column =
      (SELECT expr
       FROM table2 t_alias2
       WHERE t_alias1.column = t_alias2.column);

DELETE FROM table1 t_alias1
  WHERE column operator
       (SELECT expr
       FROM table2 t_alias2
       WHERE t_alias1.column = t_alias2.column);
```

次の文は、部門内の平均給与を超える給与を支給されている従業員の情報を戻します。給与情報が格納されている `employees` 表に別名を割り当て、相関副問合せではその別名を使用します。

```
SELECT department_id, last_name, salary
  FROM employees x
 WHERE salary > (SELECT AVG(salary)
                 FROM employees
                 WHERE x.department_id = department_id)
 ORDER BY department_id;
```

親問合せでは、相関副問合せを使用して同一部門の従業員の平均給与を、`employees` 表の行ごとに計算します。相関副問合せは、`employees` 表の各行について次の手順を実行します。

1. 行の `department_id` を判断します。
2. `department_id` に基づいて親問合せが評価されます。
3. 行の部門の平均給与より高い給与の行がある場合は、その行を戻します。

副問合せは、`employees` 表の各行につき 1 回ずつ評価されます。

DUAL 表の例 次の文は、現在の日付を戻します。

```
SELECT SYSDATE FROM DUAL;
```

`employees` 表から簡単に `SYSDATE` を選択できますが、このとき、`employees` 表のすべての行に対して 1 件ずつ 14 行の同じ `SYSDATE` が戻ります。このため、`DUAL` から選択する方が便利です。

順序の例 次の文は、`employees_seq` 順序を増分し、新しい値を戻します。

```
SELECT employees_seq.nextval  
FROM dual;
```

次の文は、`employees_seq` の現在値を選択します。

```
SELECT employees_seq.currval  
FROM dual;
```

SET CONSTRAINT[S]

用途

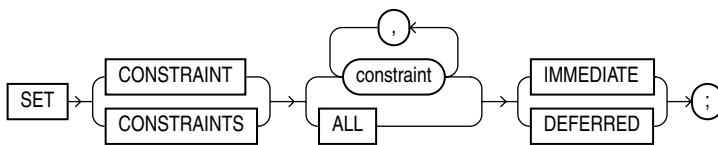
SET CONSTRAINTS 文を使用すると、遅延可能な制約の検証を、各 DML 文の実行後に行うか、トランザクションのコミット時に行うかをトランザクションごとに指定できます。

前提条件

遅延可能な制約を検証する時期を指定する場合は、制約が適用される表に対する SELECT 権限を持っているか、またはその表が自分のスキーマ内にある必要があります。

構文

set_constraints::=



キーワードとパラメータ

constraint

1 つ以上の整合性制約の名前を指定します。

ALL

このトランザクション内のすべての遅延可能な制約を設定する場合に、ALL を指定します。

IMMEDIATE

遅延可能な制約によって指定された条件が、各 DML 文の直後に検証されるようにする場合に、IMMEDIATE を指定します。

DEFERRED

遅延可能な制約によって指定された条件が、トランザクションのコミット時に検証されるようにする場合に、DEFERRED を指定します。

注意： SET CONSTRAINTS ALL IMMEDIATE 文を発行することによって、遅延可能な制約をコミットする前に、それらの制約が完全に適用されたかどうかを検証できます。

例

制約の設定例 次の文は、このトランザクション内のすべての遅延可能な制約が、各 DML 文の直後に検証されるように設定します。

```
SET CONSTRAINTS ALL IMMEDIATE;
```

次の文は、トランザクションのコミット時に 3 つの遅延制約を検証します。

```
SET CONSTRAINTS emp_job_nn, emp_salary_min,  
  hr.emp_job_fk@houston DEFERRED;
```

SET ROLE

用途

SET ROLE 文を使用すると、現行のセッションのロールを使用可能化、使用禁止化またはコンパイルすることができます。

ユーザー・ログイン時に、Oracle は、ユーザーに明示的に付与されたすべての権限およびユーザーのすべてのデフォルトのロールを使用可能にします。セッション中、ユーザーまたはアプリケーションはそのセッションに対して使用可能になっているロールを変更するために何度でも SET ROLE 文を使用できます。ただし、同時に使用可能にできるロールの数は、初期化パラメータ MAX_ENABLED_ROLES の値によって制限されます。

SESSION_ROLES データ・ディクショナリ・ビューを検索することにより、現在使用可能なロールを参照できます。

参照：

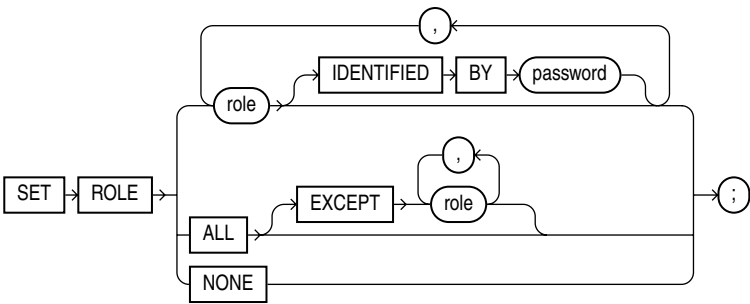
- ロールの作成については、13-72 ページの「[CREATE ROLE](#)」を参照してください。
- ユーザーのデフォルト・ロールの変更については、11-20 ページの「[ALTER USER](#)」を参照してください。
- SESSION_ROLES セッション・パラメータの詳細は、『Oracle9i データベース・リファレンス』を参照してください。

前提条件

SET ROLE 文に指定するロールが付与されている必要があります。

構文

set_role::=



キーワードとパラメータ

role

現行のセッションで使用可能にするロールを指定します。リストされないロールおよび使用不可のロールは、現行のセッションで使用禁止になります。

IDENTIFIED BY password 句では、ロールに対するパスワードを指定します。ロールにパスワードが設定されている場合は、指定する必要があります。

制限事項：ロールは、直接的または他のロールを介して付与されていないかぎり、指定できません。

ALL

現行のセッションに対して付与されているすべてのロールを使用可能にする場合に、ALL を指定します。ただし、EXCEPT 句に任意に指定されているロールは除きます。

EXCEPT 句に指定するロールは、ユーザーに直接付与されている必要があります。他のロールによってユーザーに付与されたものは無効です。

直接付与されているロール、および他のロールを介してユーザーに付与されているロールを EXCEPT 句に指定した場合、そのロールの付与先のロールにより、そのロールは使用可能のままになります。

制限事項：また、このオプションを使用して、ユーザーに直接付与されているパスワード付きのロールを使用可能にすることはできません。

NONE

現行のセッションで、DEFAULT ロールを含むすべてのロールを使用禁止にする場合に、NONE を指定します。

例

ロールの設定例 次の文は、現行のセッションのパスワード marigolds によって識別されるロール gardener を使用可能にします。

```
SET ROLE gardener IDENTIFIED BY marigolds;
```

次の文は、現行のセッションで付与されているロールをすべて使用可能にします。

```
SET ROLE ALL;
```

次の文は、dw_manager を除くロールをすべて使用可能にします。

```
SET ROLE ALL EXCEPT dw_manager;
```

次の文は、現行のセッションで付与されているすべてのロールを使用禁止にします。

```
SET ROLE NONE;
```

SET TRANSACTION

用途

SET TRANSACTION 文は、現行のトランザクションを読取り専用または読み書き両用として設定する場合、分離レベルとして設定する場合、または指定したロールバック・セグメントにトランザクションを割り当てる場合に使用します。

SET TRANSACTION 文によって実行される処理は、現行のトランザクションのみに影響します。他のユーザーまたは他のトランザクションには影響しません。COMMIT 文または ROLLBACK 文を発行すると、トランザクションは終了します。なお、現行のトランザクションは、データ定義文が実行される前後に暗黙的にコミットされます。

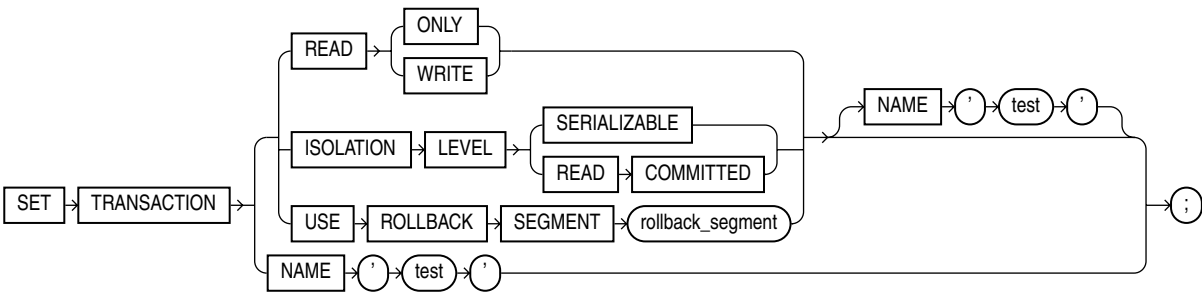
参照： 11-69 ページの「COMMIT」および 16-96 ページの「ROLLBACK」を参照してください。

前提条件

SET TRANSACTION 文を使用する場合、トランザクションの先頭に記述する必要があります。ただし、SET TRANSACTION 文が必要ないトランザクションもあります。

構文

set_transaction::=



キーワードとパラメータ

READ ONLY

READ ONLY 句は、現行のトランザクションを読取り専用に設定します。この句では、**トランザクション・レベルの読取り一貫性**を設定します。

そのトランザクションの後続のすべての問合せでは、そのトランザクション開始前にコミットされた変更のみが参照されます。読取り専用トランザクションは、他のユーザーが更新中の 1 つ以上の表に対して、複数の問合せを実行するレポートに便利です。

注意： ユーザー SYS は、この句を使用できません。SYS による問合せでは、SYS がトランザクションを READ ONLY に設定した場合でも、トランザクション中の変更が戻されます。

制限事項： 読取り専用トランザクションは、次の文のみを使用できます。

- 副問合せ (*for_update_clause* を指定しない SELECT 文)
- LOCK TABLE
- SET ROLE
- ALTER SESSION
- ALTER SYSTEM

参照：『Oracle9i データベース概要』を参照してください。

READ WRITE

現行のトランザクションを読み書き両用に設定する場合に、READ WRITE を指定します。この句では、**文レベルの読取り一貫性**を設定します。これはデフォルトです。

制限事項： 同一トランザクション内では、読取り一貫性のレベル（トランザクション・レベルおよび文レベル）を切り替えることができません。

ISOLATION LEVEL 句

データベースを変更するトランザクションがどのように処理されるかを指定する場合に、ISOLATION LEVEL 句を指定します。

- SQL92 規格に定義されているシリアライズ可能トランザクション分離モードを設定する場合に、SERIALIZABLE を指定します。シリアライズ可能トランザクションに、データ操作言語（DML）が含まれている場合、その DML が、シリアライズ可能トランザクションの開始時にコミットされていないトランザクションですでに更新されているリソースを更新すると、その DML 文は正常に実行されません。

注意： SERIALIZABLE モードで運用する場合は、初期化パラメータ COMPATIBLE を 7.3.0 以上に設定してください。

- Oracle トランザクションでは、デフォルトで READ COMMITTED が設定されています。別のトランザクションで行ロックを保持しておく必要がある DML がトランザクションに指定されていると、DML 文は行ロックが解除されるまで待ち状態になります。

USE ROLLBACK SEGMENT 句

現行のトランザクションを、指定したロールバック・セグメントに割り当てる場合に、USE ROLLBACK SEGMENT を指定します。この句によって、現行のトランザクションは暗黙的に読み書き両用トランザクションに設定されます。

この句では、トランザクションのタイプごとに、異なるサイズのロールバック・セグメントを割り当てることができます。たとえば、次のように割り当てることができます。

- 実行時間が長い複数の問合せが、同時に同じ表を読み取っていない場合は、小さいトランザクションを、小さいロールバック・セグメントに割り当てることができます。ロールバック・セグメントが小さいと、メモリー内に保持される可能性が高くなります。
- 実行時間が長い複数の問合せによって同時に読み取られる表を変更するトランザクションには、大きいロールバック・セグメントを割り当てることができます。これは、読取り一貫性問合せのために必要なロールバック情報が、上書きされないようにするためです。
- 大量のデータを挿入、更新または削除するトランザクションは、そのトランザクションのロールバック情報を保持するための十分な大きさを持つロールバック・セグメントに割り当てることができます。

1 つの SET TRANSACTION 文または同じトランザクション内の異なる文に、READ ONLY 句および USE ROLLBACK SEGMENT 句は指定できません。読取り専用トランザクションはロールバック情報を生成しないため、ロールバック・セグメントは割り当てられません。

NAME 句

NAME 句を使用すると、現行のトランザクションに名前を割当てることができます。この句は、分散データベース環境でインダウト・トランザクションを識別および変換する場合に便利です。text 文字列の最大長は 255 バイトです。

分散トランザクションに対して名前を指定する場合、トランザクションのコミット時に、名前はコミットのコメントとなり、COMMIT 文で明示的に指定した任意のコメントを上書きします。

例

次の文は、サンプル・スキーマ **Order Entry** (oe) のトロントのウェアハウスにある在庫の製品と量を計算するもので、各月の最後の日の真夜中に起動されます。このレポートは、異なるウェアハウスに対して在庫を追加および削除する他のユーザーの影響は受けません。

```
COMMIT;
SET TRANSACTION READ ONLY NAME 'Toronto';
SELECT product_id, quantity_on_hand FROM inventories
    WHERE warehouse_id = 5;
COMMIT;
```

最初の COMMIT によって、SET TRANSACTION がトランザクションの最初の文であることが保証されます。最後の COMMIT 文は、データベースに対する変更を保存するためではありません。単に、読取り専用トランザクションを終了するためのものです。

次の文は、ユーザーの現行のトランザクションを、ロールバック・セグメント rs_1 に割り当てます。

```
SET TRANSACTION USE ROLLBACK SEGMENT rs_1;
```

storage_clause

用途

`storage_clause` を使用すると、次のスキーマ・オブジェクトの記憶特性を指定できます。

- クラスタ
- 索引
- ロールバック・セグメント
- マテリアライズド・ビュー
- マテリアライズド・ビュー・ログ
- 表
- 表領域
- パーティション

記憶域パラメータは、データベースのデータへのアクセス時間およびデータベース内での領域の効率的な利用に影響します。これらのパラメータの影響の詳細は、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。

表領域の作成時に、記憶域パラメータの値を指定できます。指定した値は、表領域に割り当てられたセグメントのデフォルト値になります。

表領域を変更する場合、記憶域パラメータの値を変更できます。指定した値は、それ以降に割り当てられるセグメント（または、それ以降に作成されるオブジェクト）のデフォルト値になります。

注意： ローカル管理の表領域の場合、`storage_clause` は異なって解析されます。作成時には、`MAXEXTENTS` は無視され、他のパラメータ値を使用してセグメントの初期サイズが計算されます。詳細は、14-62 ページの「[CREATE TABLESPACE](#)」を参照してください。

クラスタ、索引、ロールバック・セグメント、マテリアライズド・ビュー、マテリアライズド・ビュー・ログ、表またはパーティションを作成する場合、これらのオブジェクトに割り当てられるセグメントに、記憶域パラメータの値を指定できます。記憶域パラメータを指定しない場合、表領域に指定されている記憶域パラメータの値が使用されます。

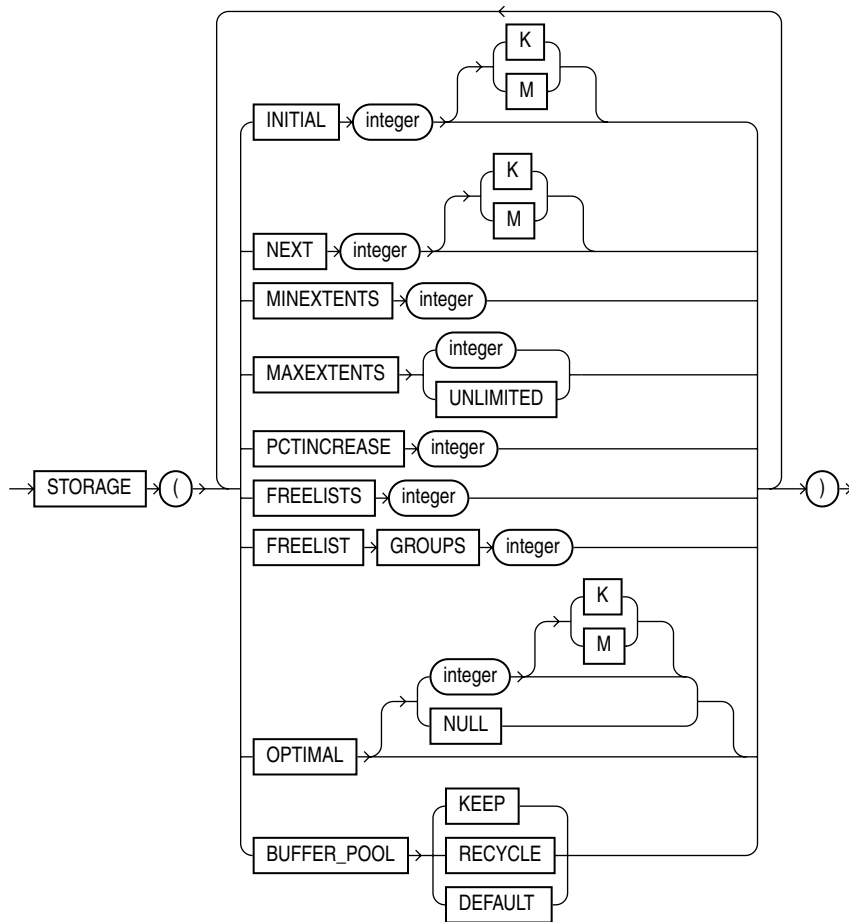
クラスタ、索引、ロールバック・セグメント、マテリアライズド・ビュー、マテリアライズド・ビュー・ログ、表またはパーティションを変更する場合、記憶域パラメータの値を変更できます。この値の変更は、それ以降のエクステンツの割当てにのみ影響します。

前提条件

STORAGE パラメータの値を変更する場合は、適切な CREATE 文または ALTER 文を使用するための権限が必要です。

構文

storage_clause::=



キーワードとパラメータ

INITIAL

オブジェクトの第 1 エクステンツ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、このエクステンツに領域が割り当てられます。K または M を使用すると、KB または MB 単位で指定できます。

デフォルトのサイズは 5 データ・ブロックです。セグメントを手動で領域管理する場合、最小値は 2 データ・ブロック・サイズに指定した各空きリスト・グループの 1 データ・ブロック・サイズを加えた値になります。セグメントを自動的に領域管理する場合、最小値は 3 データ・ブロック・サイズになります。最大値は、ご使用のオペレーティング・システムによって異なります。5 データ・ブロックより小さい値が指定された場合、データ・ブロック・サイズの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切り上げられます。

制限事項： ALTER 文では、INITIAL を指定できません。

参照： 空きリスト・グループの詳細は、17-54 ページの「[FREELIST GROUPS](#)」を参照してください。

NEXT

オブジェクトに割り当てる次のエクステンツ・サイズをバイト単位で指定します。K または M を使用して、KB または MB 単位で指定することもできます。デフォルトのサイズは 5 データ・ブロックです。最小のサイズは 1 データ・ブロックです。最大値は、ご使用のオペレーティング・システムによって異なります。5 データ・ブロックより小さい値が指定された場合、データ・ブロック・サイズの一番近い倍数に丸められます。5 データ・ブロックを超える値の場合は、断片化を最小限に抑える値に切り上げられます。

NEXT パラメータの値を変更した場合（ALTER 文で指定した場合）、次に割り当てられるエクステンツのサイズは、直前に割り当てられたエクステンツのサイズおよび PCTINCREASE パラメータの値とは関係なく、指定したサイズになります。

参照： 断片化の最小化の詳細は、『Oracle9i データベース管理者ガイド』を参照してください。

PCTINCREASE

3 番目以降の各エクステントが、直前のエクステントに対して増加する割合（パーセント）を指定します。デフォルト値は 50 です。この場合、3 番目以降のエクステントは、それぞれその直前のエクステントより 50% ずつ大きくなります。最小値は 0 で、この場合、第 2 エクステント以降のエクステントのサイズはすべて同じになります。最大値は、ご使用のオペレーティング・システムによって異なります。

計算された各エクステントのサイズは、データ・ブロック・サイズの一番近い倍数に切り上げられます。

また、PCTINCREASE パラメータの値を変更した場合（ALTER 文で指定した場合）、この変更した値と直前に割り当てられたエクステントのサイズを使用して、次に割り当てるエクステントのサイズが計算されます。

提案： すべてのエクステントを同じサイズにする場合は、PCTINCREASE の設定を 0（ゼロ）にして、SMON によるエクステントの結合を回避します。通常、設定は 0（ゼロ）にすることをお勧めします。そうすることで、断片化を最小限に抑え、処理中に一時セグメントの極端な拡大化を防ぐことができます。

制限事項： ロールバック・セグメントに PCTINCREASE は指定できません。ロールバック・セグメントでは、PCTINCREASE の値は常に 0（ゼロ）です。

MINEXTENTS

オブジェクトの作成時に割り当てられる合計エクステント数を指定します。このパラメータを使用した場合、使用可能な領域が連続していない場合でも、オブジェクト作成時にたくさんの領域を割り当てることができます。最小値（デフォルト）は 1 です。この場合、第 1 エクステントのみが割り当てられます。ただし、ロールバック・セグメントの場合、最小値（デフォルト）は 2 です。最大値は、ご使用のオペレーティング・システムによって異なります。

MINEXTENTS の値が 1 より大きい場合、INITIAL、NEXT および PCTINCREASE 記憶域パラメータの値に基づいて、次のエクステントのサイズが計算されます。

ALTER 文で MINEXTENTS の値を変更する場合、現行の値より小さくすることはできません、大きくすることはできません。MINEXTENTS をより小さい値に再設定すると便利な場合があります。たとえば、TRUNCATE ... DROP STORAGE 文の前で、TRUNCATE 操作の後もセグメントがエクステントの最小数を変更しない場合です。

制限事項： ローカル管理の表領域に存在するオブジェクトに対する MINEXTENTS の値は変更できません。

MAXEXTENTS

第1エクステントを含めて、Oracle がオブジェクトに割り当てることができるエクステントの総数を指定します。最小値は1です（最小値が常に2のロールバック・セグメントは除きます）。デフォルト値は、データ・ブロックのサイズによって異なります。

制限事項：ローカル管理の表領域に存在するオブジェクトに対する MAXEXTENTS の値は変更できません。

UNLIMITED 必要に応じてエクステントが自動的に割り当てられるようにする場合に、UNLIMITED を指定します。断片化を最小限に抑えるため、この設定をお勧めします。

ただし、ロールバック・セグメントにこの句は使用しないでください。長時間にわたって挿入、更新または削除を続ける特殊なトランザクションでは、ディスクが一杯になるまで新規エクステントの作成を続けます。

注意： *storage_clause* を指定せずに作成したロールバック・セグメントは、ロールバック・セグメントを作成した表領域の記憶域パラメータと同じ設定になります。MAXEXTENTS UNLIMITED を指定して表領域を作成する場合、ロールバック・セグメントのデフォルトは同じ設定になります。

FREELIST GROUPS

作成するデータベース・オブジェクトに対する空きリスト・グループ数を指定します。このパラメータの最小値（デフォルト）は1です。Oracle は、Real Application Clusters インスタンスのインスタンス番号を使用して、各インスタンスを空きリスト・グループにマップします。

1つの空きリスト・グループに、それぞれデータベース・ブロックを1つずつ使用します。したがって、次のことがいえます。

- 各空きリスト・グループの最小値に、1データ・ブロックを加えた値を格納できるだけの十分な大きさの値を INITIAL の値に指定していない場合、INITIAL の値は必要な分だけ引き上げられます。
- 均一なローカル管理の表領域にオブジェクトを作成する場合、空きリスト・グループ数に適応するだけの十分なエクステント・サイズがないと、作成操作は正常に実行されません。

注意： オブジェクトが存在する表領域が自動セグメント領域管理モードの場合、FREELIST GROUPS の設定は無視されます。

制限事項： FREELIST GROUPS パラメータは、CREATE TABLE、CREATE CLUSTER、CREATE MATERIALIZED VIEW、CREATE MATERIALIZED VIEW LOG および CREATE INDEX 文でのみ指定できます。

参照： 『Oracle9i Real Application Clusters 管理』を参照してください。

FREELISTS

表領域以外のオブジェクトについて、表、パーティション、クラスタまたは索引の各空きリスト・グループの空きリスト数を指定します。このパラメータの最小値（デフォルト）は1です。各空きリスト・グループには、空きリストが1つ割り当てられます。最大値は、データ・ブロックのサイズによって異なります。FREELISTS に指定した値が大きすぎた場合、最大値を示すエラーが戻ります。

注意： オブジェクトが存在する表領域が自動セグメント領域管理モードの場合、FREELIST の設定は無視されます。

制限事項： 表領域またはロールバック・セグメントを作成または変更するとき以外は、すべての文の *storage_clause* に FREELISTS を指定できます。

OPTIMAL

OPTIMAL キーワードは、ロールバック・セグメントのみに指定します。ロールバック・セグメントの最適なサイズをバイト単位で指定します。K または M を使用すると、KB または MB 単位で指定できます。エクステンツのデータがアクティブ・トランザクションで不要になった場合、Oracle は、そのエクステンツの割当てを動的に解除することによって、指定されたロールバック・セグメントのサイズを維持します。ロールバック・セグメントのサイズの合計を OPTIMAL 値より小さくせずに、できるだけ多くのエクステンツの割当てを解除します。

OPTIMAL には、MINEXTENTS、INITIAL、NEXT および PCTINCREASE パラメータで最初に割り当てた領域より小さい値を指定できません。最大値は、ご使用のオペレーティング・システムによって異なります。値は、データ・ブロック・サイズが一番近い倍数に丸められます。

NULL ロールバック・セグメントに対する最適なサイズがないことを示す場合に NULL を指定します。これは、ロールバック・セグメントのエクステンツ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。

BUFFER_POOL

BUFFER_POOL 句では、スキーマ・オブジェクト用のデフォルトのバッファ・プール（キャッシュ）を定義します。オブジェクトのすべてのブロックは、指定されたキャッシュに格納されます。

- バッファ・プールがパーティション表またはパーティション索引用に定義されている場合は、パーティションは、パーティション・レベル定義で変更されないかぎり、表定義または索引定義のバッファ・プールを継承します。
- 索引構成表の場合、索引セグメントおよびオーバーフロー・セグメントに対して個々にバッファ・プールを指定できます。

制限事項：

- この句は、クラスタ化表には指定できません。ただし、クラスタには指定できます。
- この句は、表領域またはロールバック・セグメントには指定できません。

KEEP **KEEP** を指定すると、ブロックがセグメントから **KEEP** バッファ・プールへ移されます。**KEEP** バッファ・プールに適切なサイズを維持すると、メモリーのスキーマ・オブジェクトが保持され、I/O 操作を避けることができます。**KEEP** は、表、クラスタ、マテリアライズド・ビューまたはマテリアライズド・ビュー・ログに指定する **NOCACHE** 句より優先されます。

RECYCLE **RECYCLE** を指定すると、ブロックがセグメントから **RECYCLE** プールへ移されます。**RECYCLE** プールに適切なサイズを指定すると、不要なキャッシュ領域が利用されず、デフォルト・プールが **RECYCLE** プールであるオブジェクト数が削減されます。

DEFAULT デフォルトのバッファ・プールを識別する場合に、**DEFAULT** を指定します。これは、**KEEP** も **RECYCLE** も指定しないオブジェクトのデフォルトです。

参照： 複数のバッファ・プールの使用方法については、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』を参照してください。

例

記憶域属性を使用した表の作成 次の文は、表の作成時に記憶域パラメータの値を指定します。

```
CREATE TABLE divisions
  (div_no      NUMBER(2),
   div_name    VARCHAR2(14),
   location    VARCHAR2(13) )
  STORAGE ( INITIAL 100K NEXT      50K
            MINEXTENTS 1 MAXEXTENTS 50 PCTINCREASE 5);
```

STORAGE パラメータに指定した値に基づいて、次に示すとおり表に領域が割り当てられます。

- MINEXTENTS の値は 1 のため、表作成時にエクステン트가 1 つ割り当てられます。
- INITIAL の値は 100KB のため、第 1 エクステン트의サイズは 100KB になります。
- 表データが第 1 エクステントを超えた場合、第 2 エクステン트가割り当てられます。NEXT の値が 50KB のため、第 2 エクステン트의サイズは 50KB になります。
- 表データが増加して最初の 2 つのエクステンंतを超えた場合、第 3 エクステン트가割り当てられます。PCTINCREASE の値は 5 のため、第 3 エクステン트의値は第 2 エクステン트의 5% 増の 52.5KB になります。このとき、データ・ブロック・サイズが 2KB の場合は、値は丸められて 52KB になります。

これ以降、表データの増加に応じて、直前に割り当てられたエクステン트의サイズより 5% 大きいサイズのエクステン트가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트의最大数は 50 になります。

記憶域属性を使用したロールバック・セグメントの作成 次の文は、表の作成時に記憶域パラメータの値を指定します。

```
CREATE ROLLBACK SEGMENT rs_store
  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 10KB のため、第 1 エクステントのサイズは 10KB となります。
- NEXT の値は 10KB のため、第 2 エクステントのサイズは 10KB となります。
- ロールバック・データが 2 つのエクステントを超えた場合、第 3 エクステントが割り当てられます。ロールバック・セグメントの PCTINCREASE の値は常に 0（ゼロ）のため、第 3 エクステントのサイズは第 2 エクステントのサイズと同じ 10KB です。
- MAXEXTENTS の値は 25 のため、ロールバック・セグメントに割り当てられるエクステントの最大数は 25 になります。
- OPTIMAL の値は 50KB のため、ロールバック・セグメントが 50KB を超えた場合、エクステントの割当てが解除されます。割当てが解除されるエクステントは、アクティブでなくなったトランザクションのデータが入っているエクステントのみです。

TRUNCATE

注意： TRUNCATE 文はロールバックできません。

用途

TRUNCATE 文を使用すると、表またはクラスタのすべての行を削除でき、`STORAGE` パラメータを表またはクラスタが作成されたときの値にリセットすることができます。

表を削除して再作成するより、TRUNCATE 文で行を削除する方が効果的です。表を削除して再作成した場合、その表に依存するオブジェクトが無効になり、表に対するオブジェクト権限を再度付与する必要があります。また、表の索引、整合性制約およびトリガーを再作成し、その記憶域パラメータを再指定する必要があります。TRUNCATE の場合、このような影響はありません。

参照：

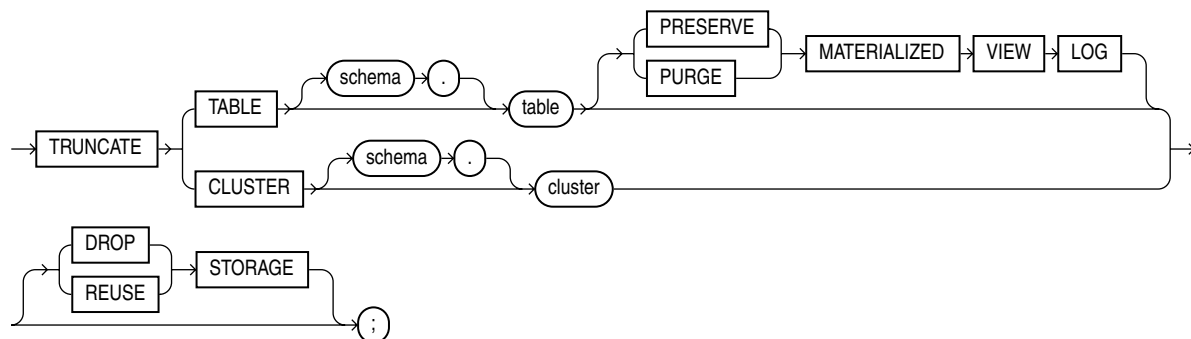
- データベースから表のデータを削除する他の方法については、15-49 ページの「[DELETE](#)」および 16-6 ページの「[DROP TABLE](#)」を参照してください。
- クラスタ化表の削除については、15-60 ページの「[DROP CLUSTER](#)」を参照してください。

前提条件

表またはクラスタを切り捨てるには、表またはクラスタが自分のスキーマ内にあるか、または `DROP ANY TABLE` システム権限が必要です。

構文

truncate::=



キーワードとパラメータ

TABLE 句

切り捨てる表が設定されているスキーマおよびその表の名前を指定します。クラスタを構成する表は、切り捨てることができません。*schema* を指定しない場合、この表が自クラスタ内に存在するとみなされます。

- 索引構成表や一時表も切り捨てることができます。一時表を切り捨てた場合、現行のセッションで作成された行のみが削除されます。
- 表を切り捨てると、記憶域パラメータ **NEXT** が、切捨てプロセス中にセグメントから最後に削除されたエクステンツのサイズに変更されます。
- **table** に対する索引（ローカル索引のレンジ・パーティションとハッシュ・パーティション、およびローカル索引のサブパーティション）の **UNUSABLE** のインジケータも、自動的に切捨ておよびリセットされます。
- **table** が空でない場合は、表中の非パーティション索引およびグローバル・パーティション索引のすべてのパーティションに **UNUSABLE** のマークが付けられます。
- ドメイン索引の場合は、この文が、適切な **TRUNCATE** ルーチンを起動し、ドメイン索引のデータを切り捨てます。

参照: ドメイン索引の詳細は、『Oracle9i Data Cartridge Developer's Guide』を参照してください。

- `table` (索引構成表および標準表) が LOB 列を含む場合、すべての LOB データおよび LOB 索引セグメントは切り捨てられます。
- `table` がパーティション化されている場合、各パーティションまたはサブパーティションの LOB データ・セグメントおよび LOB 索引セグメントと同様に、パーティションおよびサブパーティションも切り捨てられます。

注意： 表を切り捨てた場合、表中の索引データおよび表に対応付けされたマテリアライズド・ビュー・ダイレクト・パス INSERT 情報もすべて、自動的に削除されます (この情報は、マテリアライズド・ビュー・ログのいずれにも依存していません)。このダイレクト・パス INSERT 情報を削除した場合、マテリアライズド・ビューの増分リフレッシュのデータが失われる場合があります。

制限事項：

- クラスタを構成する表は、個別に切り捨てることはできません。これを行うには、クラスタを切り捨てるか、表のすべての行を削除するか、または表を削除して再作成する必要があります。
- 使用可能になっている参照整合性制約の親である表は、切り捨てることはできません。その表を切り捨てる場合、制約を無効にしておく必要があります (整合性制約が自己参照型の場合は、例外として表を切り捨てることができます)。
- `table` が階層に属する場合、階層のルートである必要があります。
- ドメイン索引が `table` で定義されている場合、IN_PROGRESS のマークが付いている索引および索引パーティションは切り捨てることはできません。

MATERIALIZED VIEW LOG 句

MATERIALIZED VIEW LOG 句では、表が切り捨てられた場合に、この表に定義されているマテリアライズド・ビュー・ログを保存するか、または削除するかを指定できます。この句を使用した場合、マテリアライズド・ビュー・マスター表を、エクスポート / インポートにより再編成できます。この場合、マスター表で定義された主キー・マテリアライズド・ビューを高速リフレッシュする機能は影響を受けません。主キー・マテリアライズド・ビューの連続高速リフレッシュをサポートする場合、マテリアライズド・ビュー・ログに主キー情報を記録する必要があります。

注意： キーワード SNAPSHOT は、MATERIALIZED VIEW のかわりに下位互換用にサポートされています。

PRESERVE マスター表を切り捨てたときにマテリアライズド・ビュー・ログを保存する場合は、PRESERVE を指定します。これはデフォルトです。

PURGE マスター表を切り捨てたときにマテリアライズド・ビュー・ログを削除する場合は、PURGE を指定します。

参照： マテリアライズド・ビュー・ログおよび TRUNCATE 文の詳細は、『Oracle9i レプリケーション』を参照してください。

CLUSTER 句

切り捨てるクラスタが設定されているスキーマと、そのクラスタの名前を指定します。なお、索引クラスタは切り捨てられますが、ハッシュ・クラスタは切り捨てられません。*schema* を指定しない場合、そのクラスタが自分のスキーマにあるとみなされます。

クラスタを切り捨てた場合、そのクラスタにある表のすべての索引データも自動的に削除されます。

STORAGE 句

STORAGE 句を指定すると、行の切捨てによって空いた領域をどのようにするかを指定できます。DROP STORAGE 句および REUSE STORAGE 句は、対応する索引から削除されたデータの空き領域にも適用されます。

DROP STORAGE 表またはクラスタの MINEXTENTS パラメータで割り当てられた領域を除き、表またはクラスタから削除された行から、すべての領域の割当てを解除する場合に、DROP STORAGE を指定します。解放された領域は、表領域の他のオブジェクトに使用されます。これはデフォルトです。

REUSE STORAGE 表またはクラスタに割り当てられた削除行から領域を確保する場合に、REUSE STORAGE を指定します。STORAGE の値は、表またはクラスタを作成したときの値にリセットされません。この領域は、挿入操作または更新操作によってその表またはクラスタ内に作成される新規データによってのみ使用されます。

注意： 切り捨てるオブジェクトに対して、2 つ以上の空きリストを指定している場合は、REUSE STORAGE 句によって、インスタンスへの空きリストのマッピングも削除され、最高水位標は第 1 エクステントの始まりにリセットされます。

例

簡単な TRUNCATE の例 次の文は、employees 表のすべての行を削除して、解放された領域を employees 表が定義されている表領域に戻します。

```
TRUNCATE TABLE employees;
```

ここでは、employees 表の索引データもすべて削除され、解放された領域は、それらの索引が定義されていた表領域に戻されます。

切捨て後の空き領域の保持例 次の文は、personnel クラスタ内の表のすべての行を削除しますが、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 personnel REUSE STORAGE
```

この文では、personnel クラスタにある表のすべての索引データも削除されます。

切捨て後のマテリアライズド・ビュー・ログの保存例 次の文は、マテリアライズド・ビュー・ログを保存する TRUNCATE 文の使用例です。

```
TRUNCATE TABLE sales PRESERVE MATERIALIZED VIEW LOG;  
TRUNCATE TABLE orders;
```

UPDATE

用途

UPDATE 文を使用すると、表またはビューの実表の既存の値を変更できます。

前提条件

表の値を更新する場合は、表が自分のスキーマ内にあるか、またはその表に対する UPDATE 権限が必要です。

ビューの実表の値を更新する場合は、次のことが必要です。

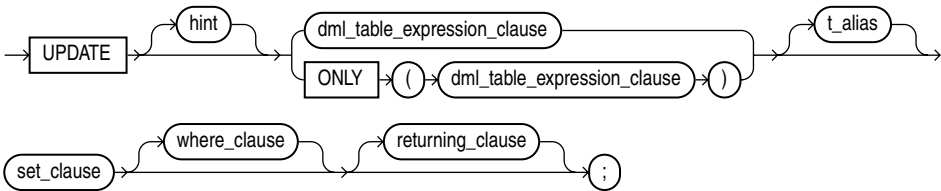
- そのビューに対する UPDATE 権限を持っている。
- そのビューが設定されているスキーマの所有者が、実表に対する UPDATE 権限を持っている。

初期化パラメータ SQL92_SECURITY が TRUE に設定されている場合、UPDATE を実行するには、(where_clause の列などの) 列値を参照している表に対する SELECT 権限が必要です。

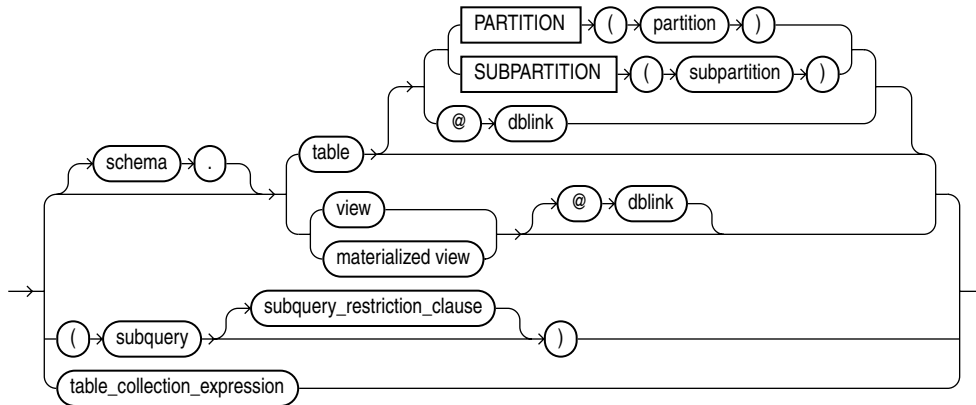
UPDATE ANY TABLE システム権限がある場合は、任意の表または任意の実表の値を更新できます。

構文

update::=

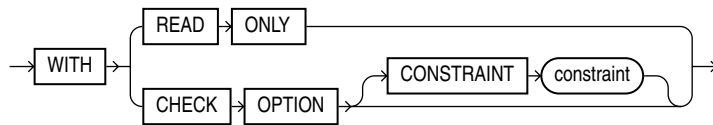


dml_table_expression_clause::=

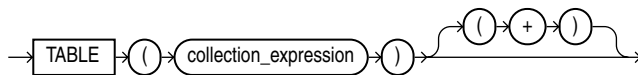


subquery: 17-4 ページの「[SELECT](#)」を参照してください。

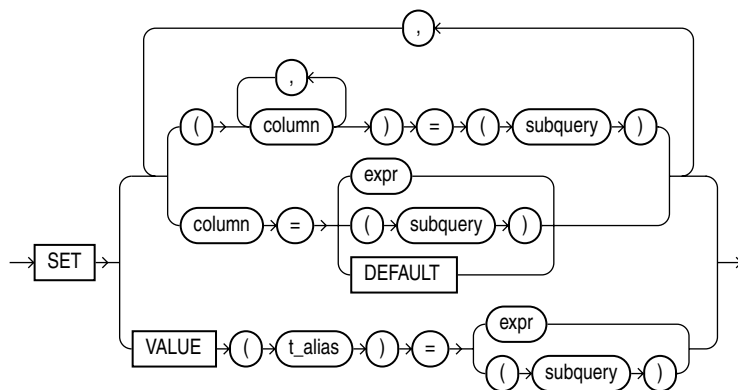
subquery_restriction_clause::=



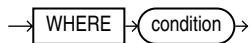
table_collection_expression::=



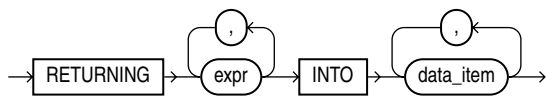
update_set_clause::=



where_clause::=



returning_clause::=



キーワードとパラメータ

hint

文に対して実行計画を選択する際の、オブティマイザへ指示を渡すコメントを指定します。

UPDATE キーワードに続けてパラレル・ヒントを指定した場合、基礎となるスキャンおよび UPDATE 操作の両方をパラレル化できます。

参照：

- ヒントの構文および説明については、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』および 2-86 ページの「[ヒント](#)」を参照してください。
- パラレル DML の詳細は、『Oracle9i データベース・パフォーマンス・ガイドおよびリファレンス』および『Oracle9i データベース概要』を参照してください。

dml_table_expression_clause

ONLY 句は、ビューのみに適用されます。UPDATE 句のビューが階層に属し、任意のサブビューから行を変更しない場合は、ONLY 構文を指定します。

参照： 17-69 ページの「[dml_table_expression_clause に関する制限事項](#)」を参照してください。

schema

表またはビューが含まれているスキーマを指定します。*schema* を指定しない場合、表またはビューが自分のスキーマにあるとみなされます。

table | view | subquery

表またはビュー、または副問合せから戻された列の名前を指定します。指定した値は、更新する必要があります。表に対して UPDATE 文を実行した場合、その表に対応付けられた UPDATE トリガーが起動します。*view* を指定した場合、Oracle はそのビューの実表を更新します。

table (または *view* の実表) に、1 列以上のドメイン索引がある場合は、この文が適切な索引タイプの更新ルーチンを実行します。

参照： これらのルーチンの詳細は、『Oracle9i Data Cartridge Developer's Guide』を参照してください。

PARTITION (*partition*) | SUBPARTITION (*subpartition*)

更新対象の表内にあるパーティションまたはサブパーティションの名前を指定します。パーティション表内の値を更新する場合は、パーティション名を指定する必要はありません。ただし、パーティション名を指定した方が、複雑な *where_clause* を使用するよりも効果的な場合もあります。

dblink

表またはビューが格納されているリモート・データベースへのデータベース・リンクの完全名または部分名を指定します。Oracle の分散機能を使用している場合にかぎり、データベース・リンクを使用して、リモート表またはリモート・ビューを更新できます。

dblink を指定しない場合、その表またはビューは、ローカル・データベース内にあるとみなされます。

参照： データベース・リンクの参照方法の詳細は、2-113 ページの「[リモート・データベース内のオブジェクトの参照](#)」を参照してください。

subquery_restriction_clause

subquery_restriction_clause を使用すると、次のいずれかの方法で副問合せを制限できます。

WITH READ ONLY WITH READ ONLY で、副問合せを更新禁止にすることを指定します。

WITH CHECK OPTION WITH CHECK OPTION で、副問合せに存在しない行を生成する表を変更禁止にすることを指定します。

参照： 17-30 ページの「[WITH CHECK OPTION の例](#)」を参照してください。

table_collection_expression

問合せおよび DML 操作で、*collection_expression* 値を表として扱う場合に、*table_collection_expression* を指定します。*collection_expression* は副問合せ、列、組込みファンクションまたはコレクション・コンストラクタにすることができます。その形式にかかわらず、集合値（ネストした表型または VARRAY 型の値）を戻す必要があります。このようなコレクションの要素抽出プロセスを **コレクション・ネスト解除** といいます。

注意： 以前のリリースの Oracle では、*collection_expression* が副問合せの場合、*table_collection_expression* を「*THE subquery*」と表現していました。現在、このような表現方法はされていません。

*table_collection_expression*を使用して、ある表の行を別の表の行を基にして更新できます。たとえば、四半期ごとの売上表を、年度ごとの売上表にまとめることができます。

t_alias

文内で参照する表、ビューまたは副問合せの**相関名**（別名）を指定します。

注意： *dml_table_expression_clause* がオブジェクト型の属性またはオブジェクト型のメソッドを参照する場合、この別名が必要です。

***dml_table_expression_clause*に関する制限事項**

- *table*（または *view* の実表）に、IN_PROGRESS または FAILED とマークされたドメイン索引がある場合は、この文は実行できません。
- 関係する索引パーティションが UNUSABLE とマークされている場合は、パーティションに挿入できません。
- *dml_table_expression_clause* の副問合せには *order_by_clause* を指定できません。
- ビューを定義する問合せに次のいずれかの要素が含まれる場合は、INSTEAD OF トリガーを除き、そのビューを更新することはできません。
 - 集合演算子
 - DISTINCT 演算子
 - 集計ファンクションまたは分析ファンクション
 - GROUP BY、ORDER BY、CONNECT BY または START WITH 句
 - SELECT 構文のリストにあるコレクション式
 - SELECT 構文のリストにある副問合せ
 - 結合（一部の例外を除く）。詳細は、『Oracle9i データベース管理者ガイド』を参照してください。
- WITH CHECK OPTION を指定してビューを作成した場合、実行結果がビューを定義する問合せの条件を満たす場合にのみ、ビューを更新できます。
- UNUSABLE のマークが付いている索引、索引パーティションまたは索引サブパーティションを指定する場合、SKIP_UNUSABLE_INDEXES セッション・パラメータが TRUE に設定されていないかぎり、UPDATE 文は正常に実行されません。

参照： SKIP_UNUSABLE_INDEXES セッション・パラメータの詳細は、9-2 ページの「ALTER SESSION」を参照してください。

update_set_clause

`update_set_clause` によって、列の値を設定します。

column

更新する表またはビューの列の名前を指定します。`update_set_clause` に表の列を指定しない場合、その列の値は変更されません。

`column` が LOB オブジェクト属性を参照している場合、まず空または NULL の値で初期化する必要があります。リテラルで更新はできません。また、UPDATE 以外の SQL 文を使用して LOB 値を更新する場合は、LOB を含む最初の行をロックしておく必要があります。

参照： 17-30 ページの「[LOB ロックの例](#)」を参照してください。

`column` がパーティション表のパーティション化キーに含まれる場合、別のパーティションまたはサブパーティションに行を移動する列の値を変更すると、行の移動を可能にしないかぎり、UPDATE は正常に実行されません。

参照： 詳細は、14-6 ページの「[CREATE TABLE](#)」の「`row_movement_clause`」または 10-2 ページの「[ALTER TABLE](#)」を参照してください。

また、`column` がリスト・パーティション表のパーティション化キーの一部である場合、パーティションの `partition_value` リストに存在していない列の値を指定すると、UPDATE は正常に実行されません。

subquery

更新される行ごとに 1 行ずつ戻す副問合せを指定します。

- `update_set_clause` で 1 列のみを指定した場合、副問合せは 1 つの値のみを戻します。
- `update_set_clause` で複数の列を指定した場合、副問合せは指定した列の数の値を戻します。

副問合せが行を戻さなかった場合は、列には NULL が割り当てられます。

注意： この `subquery` がリモート・オブジェクトを参照する場合、参照がローカル・データベース上でオブジェクトにループバックしないかぎり、UPDATE はパラレルで実行されます。ただし、`dml_table_expression_clause` の `subquery` がリモート・オブジェクトを参照する場合は、UPDATE はシリアルで実行されます。

参照：

- 17-4 ページの「[SELECT](#)」および 7-12 ページの「[副問合せの使用](#)」を参照してください。
- 14-43 ページの「[CREATE TABLE](#)」の「[parallel_clause](#)」を参照してください。

expr

対応する列に割り当てられた値を変換する式を指定します。

参照： *expr* の構文については、[第 4 章「式」](#)を参照してください。

DEFAULT DEFAULT を指定すると、以前に列のデフォルト値として指定した値を列に設定できます。対応する列に対してデフォルト値を指定していない場合、列に NULL が設定されます。

制限事項：ビューを更新する場合は、DEFAULT を指定できません。

VALUE

VALUE 句によって、オブジェクト表の行全体を指定できます。

制限事項：この句は、オブジェクト表に対してのみ指定できます。

注意： 後続の問合せ中に文字リテラルを RAW 列に挿入する場合、RAW 列にある索引は使用されずに、フル・テーブル・スキャンが行われます。

参照： 17-74 ページの「[SET VALUE の例](#)」を参照してください。

where_clause

where_clause によって、指定した条件が真の行のみが更新されるように制限できます。この句を指定しない場合、表またはビューのすべての行が更新されます。

where_clause は、値を更新する行を決定します。*where_clause* を指定しない場合、すべての行が更新されます。*where_clause* の条件を満たす各行ごとに、*update_set_clause* の等号 (=) の左側にある列に、対応する右側の式の値が設定されます。式は行が更新される場合に評価されます。

参照： 条件の構文については、[第 5 章「条件」](#)を参照してください。

returning_clause

returning_clause は DML (INSERT、UPDATE または DELETE) 文に影響される行を取り出します。この句は、表、マテリアライズド・ビューおよび単一の実表を持つビューに指定できます。

単一行で処理する場合、*returning_clause* 付きの DML 文で、処理された行および ROWID を使用している列式、処理された行への REF を取り出し、ホスト変数または PL/SQL 変数に格納します。

複数行で処理する場合、*returning_clause* 付きの DML 文で、式の値、ROWID および処理された行に関連する REF をバインド配列に格納します。

expr *expr* リストの各項目は、適切な構文で表す必要があります。スカラー副問合せ式を除くすべての形式が有効です。

INTO 変更された行の値を、*data_item* リストに指定する変数に格納することを示します。

data_item 取り出された *expr* 値を格納するホスト変数または PL/SQL 変数です。

RETURNING リストの各式については、INTO リストに、対応する型に互換がある PL/SQL 変数またはホスト変数を指定する必要があります。

制限事項：

- *returning_clause* は、マルチ表の挿入に指定できません。
- パラレル DML またはリモート・オブジェクトでは、この句を使用できません。
- この句で LONG 型を取り出すことはできません。
- INSTEAD OF トリガーが定義されたビューに対しては、この句を指定できません。

参照： BULK COLLECT 句を使用してコレクション変数に複数の値を戻す場合の詳細は、『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

簡単な例 次の文は、職種が SA_CLERK のすべての従業員の歩合に NULL 値を指定します。

```
UPDATE employees
  SET commission_pct = NULL
  WHERE job = 'SA_CLERK';
```

次の文は、Douglas Grant を部門 20 の管理者に昇格させ、給与を 1,000 ドル引き上げます。

```
UPDATE employees SET
  job_id = 'SA_MAN', salary = salary + 1000, department_id = 120
  WHERE first_name||' '||last_name = 'Douglas Grant';
```

次の文は、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;
```

PARTITION の例 次の文は、sales 表の 1 つのパーティションの値を更新します。

```
UPDATE sales PARTITION (sales_q1_1999) s
  SET s.promo_id = 494;
```

複雑な例 次の例は、UPDATE 文の次の構文要素を示します。

- 単一文にまとめた *update_set_clause* の 2 つの形式
- 関連副問合せ
- 更新された行を制限する *where_clause*

```
UPDATE employees a
  SET department_id =
    (SELECT department_id
     FROM departments
     WHERE location_id = '2100'),
    (salary, commission_pct) =
    (SELECT 1.1*AVG(salary), 1.5*AVG(commission_pct)
     FROM employees b
     WHERE a.department_id = b.department_id)
```

```
WHERE department_id IN
  (SELECT department_id
   FROM departments
   WHERE location_id = 2900
     OR location_id = 2700);
```

この UPDATE 文によって、次の処理が実行されます。

- ジュネーブまたはミュンヘン (location_id は 2900 または 2700) で働く従業員のみを更新します。
- これらの従業員の department_id をボンベイ (location_id は 2100) の対応する department_id に設定します。
- 各従業員の給与を、その部門の平均給与の 10% 引き上げます。
- 各従業員の歩合を、その部門の平均歩合の 50% 引き上げます。

SET VALUE の例 次の文は、異なるオブジェクト表 table2 の行を選択して、オブジェクト表 table1 の行を更新します。

```
UPDATE table1 p SET VALUE(p) =
  (SELECT VALUE(q) FROM table2 q WHERE p.id = q.id)
WHERE p.id = 10;
```

この副問合せでは、式に value オブジェクト参照ファンクションを使用します。

相関更新の例 次の文は、部門番号 123 の部門に関連するネストした表 projs の特定の行を更新します。

```
UPDATE TABLE(SELECT projs
  FROM dept d WHERE d.dno = 123) p
SET p.budgets = p.budgets + 1
WHERE p.pno IN (123, 456);
```

RETURNING 句の例 次の文は、更新された行の値を戻し、PL/SQL 変数 bnd1、bnd2、bnd3 に結果を格納します。

```
UPDATE employees
SET job_id = 'SA_MAN', salary = salary + 1000, department_id = 140
WHERE last_name = 'Jones'
RETURNING salary*0.25, last_name, department_id
INTO :bnd1, :bnd2, :bnd3;
```

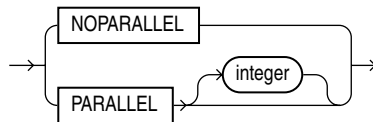
構文図の読み方

構文図とは、SQL の有効な構文を図で示したものです。構文図は、矢印が示す方向に左から右へ読んでください。

コマンドおよびキーワードは、四角形の中に大文字で書かれています。コマンドおよびキーワードは、四角形の中に書かれているとおりに指定してください。パラメータは、楕円形の中に小文字で書かれています。パラメータには変数を指定します。句読点、デリミタ、終了記号は円の中に書かれています。

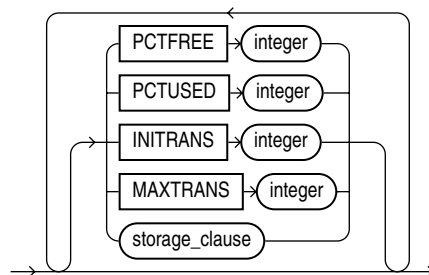
構文図の中にパスが複数ある場合は、ユーザーがとるパスを選択できます。次の例では、NOPARALLEL または PARALLEL のいずれかを指定できます。

parallel_clause::=



キーワード、演算子、パラメータに複数の選択肢がある場合は、選択できるオプションが縦に並べて書かれています。次の例では、スタックにある 4 つのパラメータから 1 つ以上を指定することができます。

physical_attributes_clause::=



構文図で使用するパラメータと、各文でそのパラメータに代入する値の例を次の表に示します。

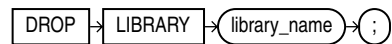
パラメータ	説明	例
<i>table</i>	パラメータによって指定された型のオブジェクト名で置き換える必要があります。オブジェクト型の一覧は、2-102 ページの「スキーマ・オブジェクト」を参照してください。	<code>emp</code>
<i>c</i>	ご使用のデータベース・キャラクタ・セットの単一文字で置き換える必要があります。	<code>T</code> <code>s</code>
<code>'text'</code>	一重引用符で囲んだテキスト文字列で置き換える必要があります。 <code>'text'</code> の構文は、2-52 ページの「テキスト・リテラル」を参照してください。	<code>'Employee records'</code>
<i>char</i>	CHAR または VARCHAR2 データ型の式か、一重引用符で囲んだ文字リテラルで置き換える必要があります。	<code>ename</code> <code>'Smith'</code>
<i>condition</i>	TRUE または FALSE に評価される条件で置き換える必要があります。 <i>condition</i> の構文は、第 5 章「条件」を参照してください。	<code>ename > 'A'</code>
<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る必要があります。	<code>TO_DATE (</code> <code>'01-Jan-1994',</code> <code>'DD-MON-YYYY')</code>
<i>expr</i>	4-2 ページの「SQL 式」にある <i>expr</i> 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。	<code>sal + 1000</code>
<i>integer</i>	2-53 ページの「整数リテラル」にある <i>integer</i> の構文の説明で定義されている整数で置き換える必要があります。	<code>72</code>
<i>number</i> <i>m</i> <i>n</i>	NUMBER データ型の式、または 2-53 ページの「数値リテラル」にある <i>number</i> の構文の説明で定義されている数値定数で置き換える必要があります。	<code>AVG(sal)</code> <code>15 * 7</code>
<i>raw</i>	RAW データ型の式で置き換える必要があります。	<code>HEXTORAW('7D')</code>
<i>subquery</i>	SELECT 文で置き換える必要があります。この SELECT 文は、別の SQL 文中で使用されます。17-4 ページの「SELECT」を参照してください。	<code>SELECT ename</code> <code>FROM emp</code>
<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名で置き換える必要があります。	<code>sales_db</code>

パラメータ	説明	例
<code>db_string</code>	Oracle Net データベース接続のデータベース識別文字列で置き換える必要があります。詳細は、ご使用の Oracle Net プロトコルのユーザーズ・ガイドを参照してください。	

必須キーワードとパラメータ

必須キーワードおよびパラメータは、単独で示されているか、または選択肢のリストとして縦に並べて書かれています。必須キーワードおよびパラメータが 1 つの場合は、メイン・パス、つまり現在選択しているパスの線上に書かれています。次の例では、`library_name` が必須パラメータです。

drop_library::=

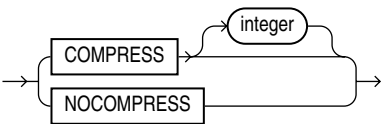


HQ_LIB という名前のライブラリがあるとする、この図は、次の文を示しています。

DROP LIBRARY hq_lib;

メイン・パスに交わる縦に並んだリストの中に、複数のキーワードまたはパラメータがある場合、そのうちの 1 つが必須です。つまり、複数のキーワードまたはパラメータから、いずれか 1 つを選択する必要があります。ただし、メイン・パスに存在しなくてもかまいません。次の例では、4 つの設定値から 1 つを選択する必要があります。

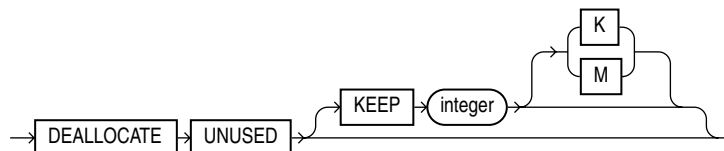
compression_clauses::=



オプションのキーワードとパラメータ

キーワードおよびパラメータが、メイン・パスより上に縦に並べてリストされている場合は、それらのキーワードおよびパラメータがオプションであることを示しています。次の例では、上のパスへ進んでも、続けてメイン・パスに沿って進んでもかまいません。

deallocate_unused_clause::=



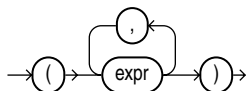
この図に従うと、次の文はすべて有効です。

```
DEALLOCATE UNUSED;
DEALLOCATE UNUSED KEEP 1000;
DEALLOCATE UNUSED KEEP 10M;
```

構文のループ

ループがある場合、ループ内の構文を何度でも繰り返して実行できます。次の例では、式を1つ選択した後、繰り返し別の式を選択できます。式と式の間はカンマで区切ります。

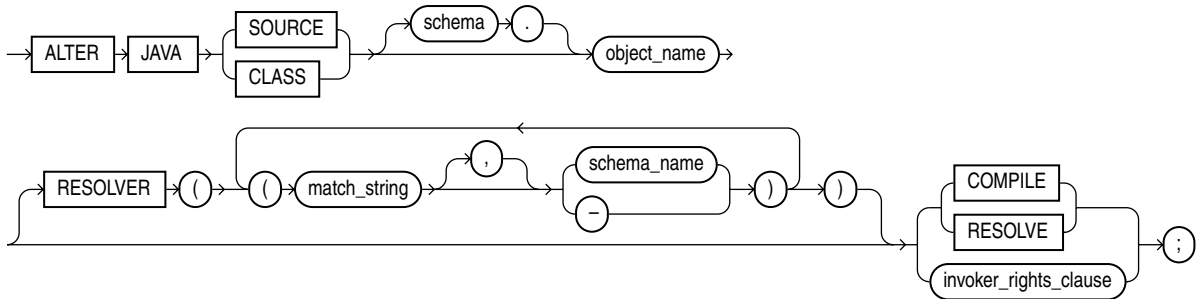
expr_list::=



複数の部分に分割された構文図

複数の部分に分割された構文図は、すべてのメイン・パスの端と端が結合されているものとしてみてください。次の例は、2 つに分割された構文図です。

`alter_java::=`



この図に従うと、次の文は有効です。

`ALTER JAVA SOURCE COMPILE;`

データベース・オブジェクト

表や列などの Oracle 識別子名は、30 文字以内に制限されています。先頭文字は英字である必要がありますが、それ以外は、英字、数字、ドル記号 (\$)、シャープ記号 (#) およびアンダースコア (_) を任意に組み合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、すべての有効な文字 (空白を含む) を組み合わせて指定できます。

Oracle の識別子は、二重引用符で囲んだとき以外は、大文字、小文字を区別しません。

詳細は、2-106 ページの「スキーマ・オブジェクトのネーミング規則」を参照してください。

Oracle と標準 SQL

この付録では、SQL:1999 規格への Oracle の規格準拠について説明します。SQL:1999 規格の必須部分は、Core SQL:1999 として知られ、SQL:1999 の Part 2「Foundation」および Part 5「Bindings」に記載されています。基礎的な機能は、Part 2 の Annex F の「SQL/Foundation feature taxonomy and definition for Core SQL」表で分析されています。バインディング機能は、Part 5 の Annex F の「SQL/Bindings feature taxonomy and definition for Core SQL」表で分析されています。

この付録では、ANSI（米国規格協会）および ISO（国際標準化機構）によって確立された SQL 規格への Oracle の規格準拠について説明します（ANSI および ISO の SQL 規格は同一です）。この付録では、次の内容を説明します。

- ANSI 規格
- ISO 規格
- Oracle の規格準拠
- FIPS の規格準拠
- 標準 SQL に対する Oracle 拡張機能
- キャラクタ・セットのサポート

ANSI 規格

SQL に関連する ANSI のドキュメントは、次のとおりです。

- ANSI/ISO/IEC 9075-1:1999, Information technology—Database languages—SQL—Part 1: Framework (SQL/Framework)
 - ANSI/ISO/IEC 9075-1:1999/Amd.1:2000
- ANSI/ISO/IEC 9075-2:1999, Information technology—Database languages—SQL—Part 2: Foundation (SQL/Foundation)
- ANSI/ISO/IEC 9075-5:1999, Information technology—Database languages—SQL—Part 5: Host Language Bindings (SQL/Bindings)

ANSI 規格のコピーについては、次の宛先に申し込んでください。

American National Standards Institute
11 West 42nd Street
New York, NY 10036 USA
電話番号 : +1.212.642.4900
FAX 番号 : +1.212.398.0023

Web サイトからも入手できます。URL は、次のとおりです。

<http://webstore.ansi.org/ansidocstore/default.asp>

SQL 規格を含む ANSI 規格のサブセットは、X3 または NCITS 規格です。これらは、NCITS (National Committee for Information Technology Standards) から入手できます。URL は、次のとおりです。

<http://www.cssinfo.com/ncitsgate.html>

ISO 規格

SQL に関連する ISO のドキュメントは、次のとおりです。

- ISO/IEC 9075-1:1999, Information technology—Database languages—SQL—Part 1: Framework (SQL/Framework)
 - ISO/IEC 9075-1:1999/Amd.1:2000
- ISO/IEC 9075-2:1999, Information technology—Database languages—SQL—Part 2: Foundation (SQL/Foundation)
- ISO/IEC 9075-5:1999, Information technology—Database languages—SQL—Part 5: Host Language Bindings (SQL/Bindings)

ISO 規格のコピーについては、次の宛先に申し込んでください。

International Organization for Standardization
1 Rue de Varembe
Case postale 56
CH-1211, Geneva 20, Switzerland
電話番号 : +41.22.749.0111
FAX 番号 : +41.22.733.3430
URL: <http://www.iso.ch/>

Web ストアからも入手できます。URL は、次のとおりです。

<http://www.iso.ch/cate/cat.html>

Oracle の規格準拠

ANSI および ISO の SQL 規格では、規格準拠性について明示する箇所に、準拠の種類および実装されている機能について記載することを義務付けています。Oracle9i サーバー、Oracle プリコンパイラ (C/C++ 用) リリース 9.0.1、Oracle プリコンパイラ (COBOL 用) リリース 9.0.1 および SOL*Module (ADA 用) リリース 9.0.1 は、次の表に示す ANSI および ISO 規格に完全 (または部分的) に準拠しています。

表 B-1 に、Oracle が完全にサポートする Core SQL:1999 機能を示します。

表 B-1 完全にサポートする Core SQL:1999 機能

機能識別番号	機能
E011	数値データ型
E031	識別子
E061	基本的な述語および検索条件
E081	基本的な権限
E091	集合ファンクション
E101	基本的なデータ操作
E111	単一行の SELECT 文
E131	NULL 値のサポート (値のかわりの NULL)
E141	基本的な整合性制約
E151	トランザクションのサポート
E152	基本的な SET TRANSACTION 文
E153	副問合せを持つ更新可能な問合せ

表 B-1 完全にサポートする Core SQL:1999 機能（続き）

機能識別番号	機能
E161	先頭に 2 つの負の符号 (-) を付けた SQL 文のコメント
E171	SQLSTATE のサポート
F041	基本的な結合表
F051	基本的な日付および時刻
F081	ビューの UNION および EXCEPT
F131	グループ操作
F181	複数モジュールのサポート
F201	CAST ファンクション
F221	明示的なデフォルト
F261	CASE 式
F311	スキーマ定義文
F471	スカラー副問合せの値
F481	拡張 NULL 述語
B011	埋込み ADA
B012	埋込み C
B013	埋込み COBOL
B014	埋込み Fortran
T431	拡張グループ化機能
T611	基本的な OLAP 演算子
T621	拡張数値ファンクション

表 B-2 に、Oracle が部分的にサポートする Core SQL:1999 機能を示します。

表 B-2 部分的にサポートする Core SQL:1999 機能

機能識別子、 機能	部分的なサポート
E021、文字 データ型	<p>次の副機能を完全にサポートします。</p> <ul style="list-style-type: none">■ E021-01、CHARACTER データ型■ E021-07、文字の連結■ E021-08、UPPER および LOWER ファンクション■ E021-09、TRIM ファンクション■ E021-10、文字データ型間の暗黙的な加算■ E021-11、文字の比較 <p>次の副機能を部分的にサポートします。</p> <ul style="list-style-type: none">■ E021-02、CHARACTER VARYING データ型 (Oracle は、長さ 0 の VARCHAR 列と NULL を区別しません)■ E021-03、文字リテラル (Oracle は、長さ 0 のリテラル '' を NULL とみなします) <p>Oracle は、次の副機能と同等の機能を持ちます。</p> <ul style="list-style-type: none">■ E021-04、CHARACTER_LENGTH ファンクションのかわりに、LENGTH ファンクションを使用します。■ E021-05、OCTET_LENGTH ファンクションのかわりに、LENGTHB ファンクションを使用します。■ E021-06、SUBSTRING ファンクションのかわりに、SUBSTR ファンクションを使用します。■ E021-11、POSITION ファンクションのかわりに、INSTR ファンクションを使用します。

表 B-2 部分的にサポートする Core SQL:1999 機能（続き）

機能識別子、機能	部分的なサポート
F031、基本的なスキーマ操作	<p>次の副機能を完全にサポートします。</p> <ul style="list-style-type: none">■ F031-01、実表を作成するための CREATE TABLE 文■ F031-02、CREATE VIEW 文■ F031-03、GRANT 文 <p>次の副機能を部分的にサポートします。</p> <ul style="list-style-type: none">■ F031-04、ALTER TABLE 文の ADD COLUMN 句（Oracle は、この構文のオプションのキーワード COLUMN をサポートしません） <p>次の副機能はサポートしません（Oracle は、キーワード RESTRICT をサポートしません）。</p> <ul style="list-style-type: none">■ F031-13、DROP TABLE 文の RESTRICT 句■ F031-16、DROP VIEW 文の RESTRICT 句■ F031-19、REVOKE 文の RESTRICT 句
E051、基本的な問合せ指定	<p>次の副機能を完全にサポートします。</p> <ul style="list-style-type: none">■ E051-01、SELECT DISTINCT■ E05102、GROUP BY 句■ E051-04、<SELECT 構文のリスト> にない列を GROUP BY に含めることができます。■ E051-06、HAVING 句■ E051-07、SELECT 構文のリストでの * の修飾 <p>次の副機能を部分的にサポートします。</p> <ul style="list-style-type: none">■ E051-05、SELECT 構文のリスト項目の名前の変更（Oracle は、列の別名をサポートしますが、オプションの AS キーワードはサポートしません）■ E051-08、FROM 句の関連名（Oracle は、関連名をサポートしますが、オプションの AS キーワードはサポートしません） <p>次の副機能はサポートされません。</p> <ul style="list-style-type: none">■ E051-09、FROM 句での列の名前の変更

表 B-2 部分的にサポートする Core SQL:1999 機能（続き）

機能識別子、 機能	部分的なサポート
E071、基本的な問合せ式	<p>次の副機能を完全にサポートします。</p> <ul style="list-style-type: none"> ■ E071-01、UNION DISTINCT 表演算子 ■ E071-02、UNION ALL 表演算子 ■ E071-05、表演算子によって結合された列の型は同じでなくてもかまいません。 ■ E071-06、副問合せの表演算子 <p>Oracle は、次の副機能と同等の機能を持ちます。</p> <ul style="list-style-type: none"> ■ E071-03、EXCEPT DISTINCT 表演算子。EXCEPT DISTINCT のかわりに、MINUS を使用します。
E121、基本的なカーソルのサポート	<p>次の副機能を完全にサポートします。</p> <ul style="list-style-type: none"> ■ E121-01、DECLARE CURSOR ■ E121-02、ORDER BY 列は、SELECT 構文のリストになってもかまいません。 ■ E121-03、ORDER BY 句の値の式 ■ E121-04、OPEN 文 ■ E121-06、位置付けされた UPDATE 文 ■ E121-07、位置付けされた DELETE 文 ■ E121-08、CLOSE 文 ■ E121-10、FETCH 文および暗黙的な NEXT <p>次の副機能を部分的にサポートします。</p> <ul style="list-style-type: none"> ■ E121-17、WITH HOLD カーソル（規格では、カーソルは ROLLBACK 中、保持されませんが、Oracle では、ROLLBACK 中も保持します）
F812、基本的なフラグ付け	<p>Oracle は、フラガーを持っていますが、SQL:1999 よりも SQL-92 に準拠しています。</p>

表 B-2 部分的にサポートする Core SQL:1999 機能（続き）

機能識別子、機能	部分的なサポート
T321、基本的な SQL 起動ルーチン	<p>次の副機能を完全にサポートします。</p> <ul style="list-style-type: none">■ T321-03、ファンクションの起動■ T321-04、CALL 文 <p>Oracle は、次の副機能を異なる構文でサポートします。</p> <ul style="list-style-type: none">■ T321-01、オーバーロードしないユーザー定義ファンクション■ T321-02、オーバーロードしないユーザー定義プロシージャ <p>Oracle の構文 CREATE FUNCTION および CREATE PROCEDURE と規格の構文の違いは、次のとおりです。</p> <ul style="list-style-type: none">■ 規格の構文では、パラメータのモード（IN、OUT または INOUT）はパラメータ名の前ですが、Oracle 構文では、パラメータ名の後です。■ 規格の構文では、INOUT を使用しますが、Oracle の構文では、IN OUT を使用します。■ Oracle の構文では、復帰型の後およびルーチン本体の定義の前に IS または AS が必要ですが、規格の構文では、必要ありません。■ ルーチンの本体が C である場合、そのルーチンに名前を付けるとき、規格の構文では、キーワード LANGUAGE C EXTERNAL NAME を使用しますが、Oracle の構文では、LANGUAGE C NAME を使用します。■ ルーチンの本体が SQL である場合、Oracle では、プロシージャを独自に拡張した PL/SQL を使用します。 <p>Oracle は、次の副機能を PL/SQL ではサポートしますが、Oracle SQL ではサポートしません。</p> <ul style="list-style-type: none">■ T321-05、RETURN 文
T612、拡張 OLAP 演算子	<p>次の副機能を完全にサポートします。</p> <ul style="list-style-type: none">■ T612.b: WIDTH_BUCKET ファンクション■ T612.c.ii: PERCENT_RANK および CUME_DIST ファンクション■ T612.f.i: 仮想集合関数および逆分散関数

Oracle には、表 B-3 で示す機能と同等の機能があります。

表 B-3 Core SQL:1999 機能と同等の機能

機能識別子、機能	同等の機能
F021、基本的な情報スキーマ	<p>Oracle は、この機能のビューを持ちませんが、同じ情報を別のメタデータ・ビューで使用可能にします。</p> <ul style="list-style-type: none"> ■ TABLES のかわりに、ALL_TABLES を使用します。 ■ COLUMNS のかわりに、ALL_TAB_COLUMNS を使用します。 ■ VIEWS のかわりに、ALL_VIEWS を使用します。 <p>ただし、ユーザーのビューに WITH CHECK OPTION が定義されていたり、更新可能な場合は、Oracle の ALL_VIEWS は表示されません。ビューに WITH CHECK OPTION が定義されているかどうかを確認する場合は、ALL_CONSTRAINTS を使用し、TABLE_NAME をそのビュー名にして、CONSTRAINT_TYPE が 'V' であるビューを検索します。</p> <ul style="list-style-type: none"> ■ TABLE_CONSTRAINTS、REFERENTIAL_CONSTRAINTS および CHECK_CONSTRAINTS のかわりに、ALL_CONSTRAINTS を使用します。 <p>ただし、Oracle の ALL_CONSTRAINTS は、制約が遅延可能か初期遅延かは表示しません。</p>

表 B-4 に、Oracle がサポートしない Core SQL:1999 機能を示します。

表 B-4 サポートしない Core SQL:1999 機能

機能識別番号	機能
F501	機能および準拠するビュー
S011	DISTINCT データ型

注意： Oracle は、「E182、モジュール言語」をサポートしません。この機能は、規格の表 31 に示されていますが、Core にはモジュール言語と埋込み言語の選択肢があることが示されているのみです。モジュール言語および埋込み言語は、機能は同じで、SQL 文がホスト・プログラミング言語と関連付けられている点のみが異なります。Oracle は、埋込み言語をサポートします。

FIPS の規格準拠

Oracle は、最新の FIPS（Federal Information Processing Standard）である FIPS PUB 127-2 に完全に準拠しています。現在、この規格は、公開されていません。ただし、FIPS PUB 127-2 で定義されたデータベース要素のサイズに関する情報に依存するアプリケーションを使用するユーザーのために、準拠性についての詳細を[表 B-5](#) に示します。

表 B-5 データベース要素のサイズ設定

データベース要素	FIPS	Oracle9i
識別子の長さ（バイト単位）	18	30
CHARACTER データ型の長さ（バイト単位）	240	2000
NUMERIC データ型の 10 進精度	15	38
DECIMAL データ型の 10 進精度	15	38
INTEGER データ型の 10 進精度	9	38
SMALLINT データ型の 10 進精度	4	38
FLOAT データ型の 2 進精度	20	126
REAL データ型の 2 進精度	20	63
DOUBLE PRECISION データ型の 2 進精度	30	126
表の中の列	100	1000
INSERT 文の中の値	100	1000
UPDATE 文内の SET 句 ^(a)	20	1000
行の長さ ^(b,c)	2,000	2,000,000
UNIQUE 制約の中の列	6	32
UNIQUE 制約の長さ ^(b)	120	(d)
外部キー列リストの長さ ^(b)	120	(d)
GROUP BY 句の中の列	6	255 ^(e)
GROUP BY 列のリストの長さ	120	(e)
ORDER BY 句の中のソート指定	6	255 ^(e)
ORDER BY 列のリストの長さ	120	(e)
参照整合性制約内の列	6	32
SQL 文で参照される表	15	無制限

表 B-5 データベース要素のサイズ設定 (続き)

データベース要素	FIPS	Oracle9i
同時にオープンできるカーソル	10	(f)
SELECT 構文のリストの項目	100	1000

(a) UPDATE 文の SET 句の数とは、SET キーワードの後に続くカンマで区切られる項目の数のことです。

(b) FIPS PUB では、列セットの長さを次の値の合計として規定しています。つまり、列の数を 2 倍した値、各文字列の長さ (バイト単位)、各真数値列の 10 進精度に 1 を加えた値、各概数値列の 2 進精度を 4 で割って 1 を加えた値の合計です。

(c) 行の最大長に対する Oracle の制限は、長さ 2GB の LONG 値とそれぞれの長さが 4000 バイトである 999 の VARCHAR2 値を含む行の最大長に基づいています。
 $2(254) + 231 + (999(4000))$

(d) UNIQUE 制約に対する Oracle の制限は、Oracle データ・ブロックのサイズ (初期化パラメータ DB_BLOCK_SIZE によって指定される) の半分からオーバーヘッドを引いたものになります。

(e) Oracle は、GROUP BY 句の列数や ORDER BY 句のソート指定の数に対して制限を設定しません。ただし、GROUP BY 句や ORDER BY 句のすべての式のサイズの合計は、Oracle データ・ブロックのサイズ (初期化パラメータ DB_BLOCK_SIZE によって指定される) からオーバーヘッドを引いたサイズに制限されています。

(f) 同時にオープンできるカーソルの数に対する Oracle の制限は、初期化パラメータ OPEN_CURSORS によって指定されます。このパラメータの最大値は、使用しているオペレーティング・システムで使用可能なメモリーによって異なりますが、どんな場合でも 100 を超えます。

標準 SQL に対する Oracle 拡張機能

Oracle では、標準 SQL 以外にも様々な機能をサポートしています。Oracle アプリケーションでは、Core SQL:1999 の使用と同様に、これらの拡張機能を使用できます。

他の SQL 処理系へのアプリケーションの移植性を考慮する場合、Oracle の FIPS フラガーを使用して、埋込み SQL プログラムでの Entry SQL92 に Oracle の拡張機能を位置付けてください。FIPS フラガーは、Oracle プリコンパイラと SQL*Module コンパイラの一部です。

参照： FIPS フラガーの使用方法の詳細は、『Pro*C/C++ Precompiler プログラマーズ・ガイド』を参照してください。

キャラクタ・セットのサポート

Oracle は、ほとんどの各国語およびベンダー固有のエンコードされたキャラクタ・セットの規格をサポートしています。Oracle がサポートするキャラクタ・セットの詳細は、『Oracle9i グローバリゼーション・サポート・ガイド』の付録 A「ロケール・データ」を参照してください。

Unicode は、エンコードされたユニバーサル・キャラクタ・セットで、単一キャラクタ・セットを使用したすべての言語の情報を格納できます。Unicode は、XML、Java、JavaScript、LDAP、CORBA 3.0 などの最新の規格で必要です。Unicode は、ISO/IEC 規格 10646 に準拠しています。ISO/IEC 規格 10646 のコピーについては、次の宛先に申し込んでください。

International Organization for Standardization
1 Rue de Varembe
Case postale 56
CH-1211, Geneva 20, Switzerland
電話番号 : +41.22.749.0111
FAX 番号 : +41.22.733.3430
URL: <http://www.iso.ch/>

Oracle9i は、Unicode 規格の最新バージョンである Unicode 3.0 に完全に準拠しています。この規格の最新情報については、次の Unicode Consortium の Web サイトを参照してください。

<http://www.unicode.org>

Oracle は、3 つのデータベース・キャラクタ・セット（ASCII ベースのプラットフォーム用の UTF8 と AL32UTF8、および EBCDIC プラットフォーム用の UTFE）によってエンコードされた UTF-8（8 ビット）を使用します。Unicode のサポートを段階的に実装する場合は、SQL の NCHAR データ型（NCHAR、NVARCHAR2 および NCLOB）に対して、Unicode データを、各国語キャラクタ・セットで、UTF-16 または UTF-8 エンコード形式によって格納できます。

参照： Oracle のキャラクタ・セットのサポートの詳細は、『Oracle9i グローバリゼーション・サポート・ガイド』を参照してください。

Oracle の予約語

この付録では、Oracle の予約語を示します。アスタリスク（*）が付いた語は、ANSI の予約語でもあります。

注意： Oracle では、次の表に示す予約語の他に、暗黙的に生成されるスキーマ・オブジェクトとサブオブジェクトには「SYS_」で始まるシステム生成名を使用します。名前解決での競合を避けるため、この接頭辞は明示的に指定するスキーマ・オブジェクト名やサブオブジェクト名では使用しないでください。

表 C-1 Oracle の予約語

ACCESS	CHAR *	DEFAULT *
ADD *	CHECK *	DELETE *
ALL *	CLUSTER	DESC *
ALTER *	COLUMN	DISTINCT *
AND *	COMMENT	DROP *
ANY *	COMPRESS	ELSE *
AS *	CONNECT *	EXCLUSIVE
ASC *	CREATE *	EXISTS
AUDIT	CURRENT *	FILE
BETWEEN *	DATE *	FLOAT *
BY *	DECIMAL *	FOR *

表 C-1 Oracle の予約語 (続き)

FROM *	NOT *	SHARE
GRANT *	NOWAIT	SIZE *
GROUP *	NULL *	SMALLINT *
HAVING *	NUMBER	START
IDENTIFIED	OF *	SUCCESSFUL
IMMEDIATE *	OFFLINE	SYNONYM
IN *	ON *	SYSDATE
INCREMENT	ONLINE	TABLE *
INDEX	OPTION *	THEN *
INITIAL	OR *	TO *
INSERT *	ORDER *	TRIGGER
INTEGER *	PCTFREE	UID
INTERSECT *	PRIOR *	UNION *
INTO *	PRIVILEGES *	UNIQUE *
IS *	PUBLIC *	UPDATE *
LEVEL *	RAW	USER *
LIKE *	RENAME	VALIDATE
LOCK	RESOURCE	VALUES *
LONG	REVOKE *	VARCHAR *
MAXEXTENTS	ROW	VARCHAR2
MINUS	ROWID	VIEW *
MISLABEL	ROWNUM	WHENEVER *
MODE	ROWS *	WHERE
MODIFY	SELECT *	WITH *
NOAUDIT	SESSION *	
NOCOMPRESS	SET *	

記号

- \$ (ドル記号)
 - 数値書式要素, 2-62
- % (パーセント) LIKE 演算子で使用, 5-14
- , (カンマ)
 - 数値書式要素, 2-62
 - 日時書式要素, 2-67
- (ダッシュ)
 - 日時書式要素, 2-67
- . (ピリオド)
 - 数値書式要素, 2-62
 - 日時書式要素, 2-67
- / (スラッシュ)
 - 日時書式要素, 2-67
- : (コロン)
 - 日時書式要素, 2-67
- ; (セミコロン)
 - 日時書式要素, 2-67

数字

- 0 (ゼロ)
 - 数値書式要素, 2-62
- 16 進数値
 - 戻す, 2-64
- 20 世紀, 2-71
 - 指定, 2-71
- 21 世紀, 2-71
 - 指定, 2-71
- 9
 - 数値書式要素, 2-62

A

- A.D. 日時書式要素, 2-67, 2-70
- A.M. 日時書式要素, 2-67, 2-70
- ABS ファンクション, 6-14
- ACCESSED GLOBALLY 句
 - CREATE CONTEXT, 12-12
- ACCOUNT LOCK 句
 - ALTER USER
 - 「CREATE USER」を参照
 - CREATE USER, 15-34
- ACCOUNT UNLOCK 句
 - ALTER USER
 - 「CREATE USER」を参照
 - CREATE USER, 15-34
- ACOS ファンクション, 6-14
- ACTIVATE STANDBY DATABASE 句
 - ALTER DATABASE, 8-34
- ACTIVE_INSTANCE_COUNT 初期化パラメータ
 - ALTER SYSTEM で設定, 9-32
- ADD DATAFILE 句
 - ALTER TABLESPACE, 10-85
- ADD LOG GROUP 句
 - ALTER TABLE, 10-30
- ADD LOGFILE GROUP 句
 - ALTER DATABASE, 8-30
- ADD LOGFILE MEMBER 句
 - ALTER DATABASE, 8-15, 8-30
- ADD LOGFILE THREAD 句
 - ALTER DATABASE, 8-30
- ADD LOGFILE 句
 - ALTER DATABASE, 8-15
- ADD OVERFLOW 句
 - ALTER TABLE, 10-36

ADD PARTITION 句
 ALTER TABLE, 10-45, 10-46
 ADD PRIMARY KEY 句
 ALTER MATERIALIZED VIEW LOG, 8-95
 ADD ROWID 句
 ALTER MATERIALIZED VIEW, 8-95
 ALTER MATERIALIZED VIEW LOG, 8-95
 ADD SUPPLEMENTAL LOG DATA 句
 ALTER DATABASE, 8-31
 ADD TEMPFILE 句
 ALTER TABLESPACE, 10-85
 ADD VALUES 句
 ALTER TABLE MODIFY PARTITION, 10-40
 ADD_MONTHS ファンクション, 6-15
 ADD 句
 ALTER DIMENSION, 8-45
 ALTER INDEXTYPE, 8-70
 ALTER TABLE, 10-55
 ALTER VIEW, 11-29
 ADMINISTER DATABASE TRIGGER システム権限,
 16-44
 ADVISE 句
 ALTER SESSION, 9-3
 AD 日時書式要素, 2-67, 2-70
 AFTER 句
 CREATE TRIGGER, 14-80
 AFTER トリガー, 14-80
 AGENT 句
 CREATE LIBRARY, 13-3
 ALL EXCEPT 句
 SET ROLE, 17-44
 ALL PRIVILEGES 句
 GRANT, 16-36
 REVOKE, 16-91
 ALL PRIVILEGES ショートカット
 AUDIT, 11-53
 ALL_COL_COMMENTS データ・ディクショナリ・
 ビュー, 11-66
 ALL_ROWS ヒント, 2-88
 ALL_TAB_COMMENTS データ・ディクショナリ・
 ビュー, 11-66
 ALLOCATE EXTENT 句
 ALTER CLUSTER, 8-4, 8-5
 ALTER INDEX, 8-50, 8-55
 ALTER MATERIALIZED VIEW, 8-77
 ALTER TABLE, 10-31
 ALLOW CORRUPTION 句
 ALTER DATABASE RECOVER, 8-22
 ALL 演算子, 5-5
 ALL 句
 SELECT, 17-11
 SET CONSTRAINTS, 17-41
 SET ROLE, 17-44
 ALL ショートカット
 AUDIT, 11-53
 ALTER ANY CLUSTER システム権限, 16-38
 ALTER ANY DIMENSION システム権限, 16-38
 ALTER ANY INDEXTYPE システム権限, 16-39
 ALTER ANY INDEX システム権限, 16-39
 ALTER ANY MATERIALIZED VIEW システム権限,
 16-40
 ALTER ANY OUTLINE システム権限, 16-41
 ALTER ANY PROCEDURE システム権限, 16-41
 ALTER ANY ROLE システム権限, 16-41
 ALTER ANY SEQUENCE システム権限, 16-42
 ALTER ANY TABLE システム権限, 16-43
 ALTER ANY TRIGGER システム権限, 16-44
 ALTER ANY TYPE システム権限, 16-44
 ALTER CLUSTER 文, 8-3
 ALTER DATABASE システム権限, 16-38
 ALTER DATABASE 文, 8-9
 ALTER DIMENSION 文, 8-43
 ALTER FUNCTION 文, 8-46
 ALTER INDEXTYPE 文, 8-69
 ALTER INDEX 文, 8-48
 ALTER JAVA CLASS 文, 8-71
 ALTER JAVA SOURCE 文, 8-71
 ALTER MATERIALIZED VIEW LOG 文, 8-90
 ALTER MATERIALIZED VIEW 文, 8-74
 ALTER OUTLINE 文, 8-97
 ALTER PACKAGE 文, 8-99
 ALTER PROCEDURE 文, 8-103
 ALTER PROFILE システム権限, 16-41
 ALTER PROFILE 文, 8-106
 ALTER RESOURCE COST システム権限, 16-42
 ALTER RESOURCE COST 文, 8-110
 ALTER ROLE 文, 8-113
 ALTER ROLLBACK SEGMENT システム権限, 16-42
 ALTER ROLLBACK SEGMENT 文, 8-115
 ALTER SEQUENCE 文, 8-119
 ALTER SESSION システム権限, 16-42
 ALTER SESSION 文, 9-2

- ALTER SNAPSHOT
 - 「ALTER MATERIALIZED VIEW」を参照
- ALTER SNAPSHOT LOG
 - 「ALTER MATERIALIZED VIEW LOG」を参照
- ALTER SYSTEM システム権限, 16-38
- ALTER SYSTEM 文, 9-19
- ALTER TABLESPACE システム権限, 16-43
- ALTER TABLESPACE 文, 10-82
- ALTER TABLE 文, 10-2
- ALTER TRIGGER 文, 11-2
- ALTER TYPE 文, 11-6
- ALTER USER システム権限, 16-45
- ALTER USER 文, 11-20
- ALTER VIEW 文, 11-28
- alter_column_properties 句
 - ALTER TABLE, 10-20
- alter_constraint_clauses
 - ALTER TABLE, 10-19
- alter_external_table_clause
 - ALTER TABLE, 10-24
- ALTER オブジェクト権限, 16-49
 - 順序, 16-50
 - 表, 16-50
- ALTER 文
 - トリガー, 14-83
- AM 日時書式要素, 2-67, 2-70
- ANALYZE ANY システム権限, 16-46
- ANALYZE CLUSTER 文, 11-31
- ANALYZE INDEX 文, 11-31
- ANALYZE TABLE 文, 11-31
- ANCILLARY TO 句
 - CREATE OPERATOR, 13-40
- AND DATAFILES 句
 - DROP TABLESPACE, 16-11
- AND_EQUAL ヒント, 2-89
- AND 条件, 5-8, 5-9
- ANSI (米国規格協会), B-1
 - 暗黙的なデータ型変換, 2-34
 - 規格, xv, 1-2, B-2
 - サポートしているデータ型, 2-5
 - データ型, 2-34
 - Oracle データ型への変換, 2-34
- ANY 演算子, 5-5
- APPEND ヒント, 2-89
- AQ_ADMINISTRATOR_ROLE ロール, 16-48
- AQ_TM_PROCESSES 初期化パラメータ
 - ALTER SYSTEM で設定, 9-32
- AQ_USER_ROLE ロール, 16-48
- ARCHIVE LOG 句
 - ALTER SYSTEM, 9-22
- ARCHIVE_LAG_TARGET 初期化パラメータ
 - ALTER SYSTEM で設定, 9-33
- ARCHIVELOG 句
 - ALTER DATABASE, 8-15, 8-29
 - CREATE CONTROLFILE, 12-18
 - CREATE DATABASE, 12-26
- AS EXTERNAL 句
 - CREATE FUNCTION, 13-63
 - CREATE TYPE BODY, 15-28
- AS OBJECT 句
 - CREATE TYPE, 15-9
- AS TABLE 句
 - CREATE TYPE, 15-18
- AS VARRAY 句
 - CREATE TYPE, 15-17
- AS 'filespec' 句
 - CREATE LIBRARY, 13-3
- ASCII
 - キャラクタ・セット, 2-44
- ASCIISTR ファンクション, 6-17
- ASCII ファンクション, 6-16
- ASC 句
 - CREATE INDEX, 12-69
- ASIN ファンクション, 6-17
- ASSOCIATE STATISTICS 文, 11-46
- AS 句
 - CREATE JAVA, 12-91
- AS 副問合せ句
 - CREATE MATERIALIZED VIEW, 13-12, 13-22
 - CREATE TABLE, 14-49
 - CREATE VIEW, 15-43
- ATAN2 ファンクション, 6-19
- ATAN ファンクション, 6-18
- ATTRIBUTE 句
 - ALTER DIMENSION, 8-44
 - CREATE DIMENSION, 12-40, 12-43
- AUDIT ANY システム権限, 16-46
- AUDIT SYSTEM システム権限, 16-38
- AUDIT_FILE_DEST 初期化パラメータ
 - ALTER SYSTEM で設定, 9-33
- AUDIT_TRAIL 初期化パラメータ
 - ALTER SYSTEM で設定, 9-33
- auditing_by_clause
 - AUDIT, 11-51

- auditing_on_clause
 - AUDIT, 11-52
- AUTHENTICATED BY 句
 - CREATE DATABASE LINK, 12-37
- AUTHENTICATED 句
 - ALTER USER, 11-24
- AUTHID CURRENT_USER 句
 - ALTER JAVA, 8-72
 - CREATE FUNCTION, 12-52
 - CREATE JAVA, 12-87, 12-89
 - CREATE PACKAGE, 13-48
 - CREATE PROCEDURE, 13-62
 - CREATE TYPE, 11-13, 15-9
- AUTHID DEFINER 句
 - ALTER JAVA, 8-72
 - CREATE FUNCTION, 12-52
 - CREATE JAVA, 12-87, 12-89
 - CREATE PACKAGE, 13-48
 - CREATE PROCEDURE, 13-62
 - CREATE TYPE, 11-13, 15-9
- AUTOALLOCATE 句
 - CREATE TABLESPACE, 14-69
- AUTOEXTEND 句
 - ALTER DATABASE, 8-14
 - ALTER TABLESPACE, 10-84, 10-86
 - CREATE DATABASE, 12-23
 - CREATE TABLESPACE, 14-64, 14-66
 - CREATE TEMPORARY TABLESPACE, 14-74, 14-75
- AVG ファンクション, 6-19

B

- B
 - 数値書式要素, 2-62
- B.C. 日時書式要素, 2-67, 2-70
- BACKGROUND_CORE_DUMP 初期化パラメータ
 - ALTER SYSTEM で設定, 9-34
- BACKGROUND_DUMP_DEST 初期化パラメータ
 - ALTER SYSTEM で設定, 9-34
- BACKUP ANY TABLE システム権限, 16-43
- BACKUP CONTROLFILE 句
 - ALTER DATABASE, 8-16, 8-33
- BACKUP_TAPE_IO_SLAVES 初期化パラメータ
 - ALTER SYSTEM で設定, 9-34
- BC 日時書式要素, 2-67, 2-70
- BECOME USER システム権限, 16-45

- BEFORE 句
 - CREATE TRIGGER, 14-80
- BEFORE トリガー, 14-80
- BEGIN BACKUP 句
 - ALTER TABLESPACE, 10-89
- BFILE
 - データ型, 2-30
 - ロケータ, 2-30
- BFILENAME ファンクション, 6-21
- BIN_TO_NUM ファンクション, 6-22
- BINDING 句
 - CREATE OPERATOR, 13-38, 13-40
- BITAND ファンクション, 6-23
- bitmap_join_index_clause
 - CREATE INDEX, 12-62
- BITMAP_MERGE_AREA_SIZE 初期化パラメータ
 - ALTER SYSTEM で設定, 9-35
- BITMAP 句
 - CREATE INDEX, 12-65
- BLANK_TRIMMING 初期化パラメータ
 - ALTER SYSTEM で設定, 9-35
- BLOB データ型, 2-30
 - トランザクションのサポート, 2-30
- BLOCKSIZE 句
 - CREATE TABLESPACE, 14-67
- BODY 句
 - ALTER PACKAGE, 8-100
- BUFFER_POOL_KEEP 初期化パラメータ
 - ALTER SYSTEM で設定, 9-36
- BUFFER_POOL_RECYCLE 初期化パラメータ
 - ALTER SYSTEM で設定, 9-36
- BUFFER_POOL パラメータ
 - STORAGE 句, 17-56
- BUILD DEFERRED 句
 - CREATE MATERIALIZED VIEW, 13-16
- BUILD IMMEDIATE 句
 - CREATE MATERIALIZED VIEW, 13-16
- BY ACCESS 句
 - AUDIT, 11-55
- BY proxy 句
 - AUDIT, 11-54
- BY SESSION 句
 - AUDIT, 11-55
- BY user 句
 - AUDIT, 11-54
- BYTE の長さのセマンティクス, 10-58
- BYTE の文字のセマンティクス, 2-11, 2-10

C

C

- 数値書式要素, 2-62
- CACHE READS 句
 - ALTER TABLE, 10-65
 - CREATE TABLE, 14-42
- CACHE 句
 - ALTER MATERIALIZED VIEW, 8-82
 - ALTER MATERIALIZED VIEW LOG, 8-94
 - ALTER TABLE, 10-33, 14-42
 - CREATE CLUSTER, 12-9
 - CREATE MATERIALIZED VIEW, 13-14
 - CREATE MATERIALIZED VIEW LOG, 13-33
- CACHE パラメータ
 - ALTER SEQUENCE
 - 「CREATE SEQUENCE」を参照
 - CREATE SEQUENCE, 13-84
- CACHE ヒント, 2-89
- CALL 句
 - CREATE TRIGGER, 14-87
- CALL プロシージャ文
 - CREATE TRIGGER, 14-87
- CALL 文, 11-63
- CASCADE CONSTRAINTS 句
 - DROP CLUSTER, 15-61
 - DROP TABLE, 16-8
 - DROP TABLESPACE, 16-11
 - of DROP VIEW, 16-22
 - REVOKE, 16-91
- CASCADE 句
 - CREATE TABLE, 14-49
 - DROP PROFILE, 15-89
 - DROP USER, 16-20
- CASE 式, 4-6
 - 検索, 4-6
 - 単純, 4-6
- CAST ファンクション, 6-24
 - MULTISET パラメータ, 6-24
- CATSEARCH 条件, 5-2
- CC 日時書式要素, 2-67
- CEIL ファンクション, 6-27
- CHANGE CATEGORY 句
 - ALTER OUTLINE, 8-98
- CHAR VARYING データ型, ANSI, 2-34
- CHARACTER SET パラメータ
 - ALTER DATABASE, 8-35
 - CREATE CONTROLFILE, 12-18
 - CREATE DATABASE, 12-27
- CHARACTER VARYING データ型
 - ANSI, 2-34
- CHARACTER データ型
 - ANSI, 2-34
 - DB2, 2-35
 - SQL/DS, 2-35
- CHARTOROWID ファンクション, 6-27
- CHAR データ型, 2-10
 - ANSI, 2-34
 - VARCHAR2 への変換, 2-61
- CHAR の長さのセマンティクス, 10-58
- CHAR の文字のセマンティクス, 2-10, 2-11
- CHECK DATAFILES 句
 - ALTER SYSTEM, 9-25
- CHECKPOINT 句
 - ALTER SYSTEM, 9-24
- CHECK 句
 - constraint_clause, 11-82
 - CREATE TABLE, 14-21
- CHOOSE ヒント, 2-89
- CHR ファンクション, 6-28
- CHUNK 句
 - ALTER TABLE, 10-66
 - CREATE TABLE, 14-33
- CIRCUITS 初期化パラメータ
 - ALTER SYSTEM で設定, 9-37
- CLEAR LOGFILE 句
 - ALTER DATABASE, 8-15, 8-32
- CLOB データ型, 2-31
 - トランザクションのサポート, 2-31
- CLOSE DATABASE LINK 句
 - ALTER SESSION, 9-3
- CLUSTER_DATABASE_INSTANCES 初期化パラメータ
 - ALTER SYSTEM で設定, 9-38
- CLUSTER_DATABASE 初期化パラメータ
 - ALTER SYSTEM で設定, 9-37
- cluster_index_clause
 - CREATE INDEX, 12-60
- CLUSTER_INTERCONNECTS 初期化パラメータ
 - ALTER SYSTEM で設定, 9-38
- CLUSTER 句
 - ANALYZE, 11-36
 - CREATE INDEX, 12-60, 12-66
 - CREATE TABLE, 14-31

- TRUNCATE, 17-62
- CLUSTER ヒント, 2-90
- COALESCE SUBPARTITION 句
 - ALTER TABLE, 10-40
- COALESCE 句
 - ALTER INDEX, 8-63
 - ALTER TABLE, 10-37, 10-40
 - ALTER TABLESPACE, 10-90
 - パーティション, 10-47
- COALESCE ファンクション, 6-29
 - 様々な CASE 式, 6-29
- column_properties 句
 - ALTER TABLE, 10-20, 10-64
- COLUMNS 句
 - ASSOCIATE STATISTICS, 11-46, 11-48
- COMMENT ANY TABLE システム権限, 16-46
- COMMENT 句
 - COMMIT, 11-70
- COMMENT 文, 11-66
- COMMIT IN PROCEDURE 句
 - ALTER SESSION, 9-3
- COMMIT_POINT_STRENGTH 初期化パラメータ
 - ALTER SYSTEM で設定, 9-38
- COMMIT 文, 11-69
- COMPATIBLE 初期化パラメータ
 - ALTER SYSTEM で設定, 9-39
- COMPILE 句
 - ALTER DIMENSION, 8-45
 - ALTER FUNCTION, 8-47
 - ALTER JAVA SOURCE, 8-72
 - ALTER MATERIALIZED VIEW, 8-87
 - ALTER PACKAGE, 8-100
 - ALTER PROCEDURE, 8-104
 - ALTER TRIGGER, 11-4
 - ALTER TYPE, 11-9
 - ALTER VIEW, 11-30
 - CREATE JAVA, 12-88
- COMPOSE ファンクション, 6-31
- COMPOSITE_LIMIT パラメータ
 - ALTER PROFILE, 8-107
 - CREATE PROFILE, 13-69
- COMPRESS 句
 - ALTER INDEX, 8-51
 - ALTER INDEX ... REBUILD, 8-60
 - ALTER TABLE, 10-69
 - CREATE TABLE, 14-28
- COMPUTE STATISTICS 句
 - ALTER INDEX REBUILD, 8-60
 - ANALYZE, 11-36
 - CREATE INDEX, 12-72
- CONCAT ファンクション, 6-31
- conditional_insert_clause
 - INSERT, 16-66
- CONNECT BY 句
 - SELECT, 7-5, 17-18
 - 問合せおよび副問合せ, 17-18
- CONNECT THROUGH 句
 - ALTER USER, 11-24
- CONNECT TO 句
 - CREATE DATABASE LINK, 12-36
- CONNECT_TIME パラメータ
 - ALTER PROFILE, 8-107
 - ALTER RESOURCE COST, 8-111
- CONNECT 句
 - SELECT および副問合せ, 17-8
- CONNECT ロール, 16-48
- CONSIDER FRESH 句
 - ALTER MATERIALIZED VIEW, 8-87
- CONSTRAINT(S) セッション・パラメータ, 9-10
- CONTAINS 条件, 5-2
- CONTROL_FILE_RECORD_KEEP_TIME 初期化パラメータ
 - ALTER SYSTEM で設定, 9-39
- CONTROL_FILES 初期化パラメータ
 - ALTER SYSTEM で設定, 9-40
- CONTROLFILE REUSE 句
 - CREATE DATABASE, 12-24
- controlfile_clauses
 - ALTER DATABASE, 8-16
- CONVERT 句
 - ALTER DATABASE, 8-37
- CONVERT ファンクション, 6-32
- CORE_DUMP_DEST 初期化パラメータ
 - ALTER SYSTEM で設定, 9-40
- CORR ファンクション, 6-34
- COSH ファンクション, 6-36
- COS ファンクション, 6-36
- COUNT ファンクション, 6-37
- COVAR_POP ファンクション, 6-39
- COVAR_SAMP ファンクション, 6-41
- CPU_COUNT 初期化パラメータ
 - ALTER SYSTEM で設定, 9-41

CPU_PER_CALL パラメータ
 ALTER PROFILE, 8-107
 CREATE PROFILE, 13-68
 CPU_PER_SESSION パラメータ
 ALTER PROFILE, 8-107
 ALTER RESOURCE COST, 8-111
 CREATE PROFILE, 13-68
 CREATE ANY CLUSTER システム権限, 16-38
 CREATE ANY CONTEXT システム権限, 16-38
 CREATE ANY DIMENSION システム権限, 16-38
 CREATE ANY DIRECTORY システム権限, 16-39
 CREATE ANY INDEXTYPE システム権限, 16-39
 CREATE ANY INDEX システム権限, 16-39
 CREATE ANY LIBRARY システム権限, 16-39
 CREATE ANY MATERIALIZED VIEW システム権限,
 16-40
 CREATE ANY OPERATOR システム権限, 16-40
 CREATE ANY OUTLINE システム権限, 16-41
 CREATE ANY PROCEDURE システム権限, 16-41
 CREATE ANY SEQUENCE システム権限, 16-42
 CREATE ANY SYNONYM システム権限, 16-42
 CREATE ANY TABLE システム権限, 16-43
 CREATE ANY TRIGGER システム権限, 16-44
 CREATE ANY TYPE システム権限, 16-44
 CREATE ANY VIEW システム権限, 16-45
 CREATE CLUSTER システム権限, 16-38
 CREATE CLUSTER 文, 12-2
 CREATE CONTEXT 文, 12-11
 CREATE CONTROLFILE 文, 12-14
 CREATE DATABASE LINK システム権限, 16-38
 CREATE DATABASE LINK 文, 12-33
 CREATE DATABASE 文, 12-20
 CREATE DATAFILE 句
 ALTER DATABASE, 8-13, 8-26
 CREATE DIMENSION
 システム権限, 16-38
 CREATE DIMENSION 文, 12-39
 CREATE DIRECTORY 文, 12-44
 CREATE FUNCTION 文, 12-47
 CREATE INDEX
 文, 12-58
 CREATE INDEXTYPE
 文, 12-84
 CREATE INDEXTYPE システム権限, 16-39
 CREATE JAVA 文, 12-86
 CREATE LIBRARY システム権限, 16-39
 CREATE LIBRARY 文, 13-2
 CREATE MATERIALIZED VIEW LOG 文, 13-29
 CREATE MATERIALIZED VIEW システム権限, 16-40
 CREATE MATERIALIZED VIEW 文, 13-5
 CREATE OPERATOR システム権限, 16-40
 CREATE OPERATOR 文, 13-38
 CREATE OUTLINE 文, 13-42
 CREATE PACKAGE BODY 文, 13-51
 CREATE PACKAGE 文, 13-46
 CREATE PFILE 文, 13-56
 CREATE PROCEDURE システム権限, 16-41
 CREATE PROCEDURE 文, 13-58
 CREATE PROFILE システム権限, 16-41
 CREATE PROFILE 文, 13-65
 CREATE PUBLIC DATABASE LINK システム権限,
 16-38
 CREATE PUBLIC SYNONYM システム権限, 16-42
 CREATE ROLE システム権限, 16-41
 CREATE ROLE 文, 13-72
 CREATE ROLLBACK SEGMENT システム権限, 16-42
 CREATE ROLLBACK SEGMENT 文, 13-75
 CREATE SCHEMA 文, 13-79
 CREATE SESSION システム権限, 16-42
 CREATE SEQUENCE 文, 13-81
 CREATE SESSION システム権限, 16-42
 CREATE SPFILE 文, 13-86
 CREATE STANDBY CONTROLFILE 句
 ALTER DATABASE, 8-16, 8-33
 CREATE SYNONYM システム権限, 16-42
 CREATE SYNONYM 文, 14-2
 CREATE TABLESPACE システム権限, 16-43
 CREATE TABLESPACE 文, 14-62
 CREATE TABLE システム権限, 16-43
 CREATE TABLE 文, 14-6
 CREATE TEMPORARY TABLESPACE 文, 14-73
 CREATE TRIGGER システム権限, 16-44
 CREATE TRIGGER 文, 14-77
 CREATE TYPE BODY 文, 15-24
 CREATE TYPE システム権限, 16-44
 CREATE TYPE 文, 15-3
 CREATE USER システム権限, 16-45
 CREATE USER 文, 15-30
 CREATE VIEW システム権限, 16-45
 CREATE VIEW 文, 15-37
 CREATE_BITMAP_AREA_SIZE 初期化パラメータ
 ALTER SYSTEM で設定, 9-41
 CREATE_STORED_OUTLINES 初期化パラメータ
 ALTER SYSTEM で設定, 9-42

- CREATE_STORED_OUTLINES セッション・パラメータ, 9-10
- CREATE 文
 - トリガー, 14-83
- CUBE 句
 - SELECT 文, 17-19
- CUME_DIST ファンクション, 6-43
- CURRENT_DATE ファンクション, 6-44
- CURRENT_SCHEMA セッション・パラメータ, 9-11
- CURRENT_TIMESTAMP ファンクション, 6-45
- CURRENT_USER 句
 - CREATE DATABASE LINK, 12-36
- CURRVAL 疑似列, 2-79, 13-81
- CURSOR_SHARING 初期化パラメータ
 - ALTER SESSION で設定, 9-7
- CURSOR_SPACE_FOR_TIME 初期化パラメータ
 - ALTER SYSTEM で設定, 9-42
- CURSOR 式, 4-8
- CustomDatum Java 記憶形式, 15-11
- CYCLE パラメータ
 - ALTER SEQUENCE
 - 「CREATE SEQUENCE」を参照
 - CREATE SEQUENCE, 13-84
- C 関数
 - オブジェクト型へのマップ, 15-14
- C 句
 - CREATE TYPE, 15-14
 - CREATE TYPE BODY, 15-27

D

- D
 - 数値書式要素, 2-62
- DATAFILE END BACKUP 句
 - ALTER DATABASE, 8-26
- DATAFILE OFFLINE 句
 - ALTER DATABASE, 8-27
- DATAFILE ONLINE 句
 - ALTER DATABASE, 8-27
- DATAFILE RESIZE 句
 - ALTER DATABASE, 8-27
- DATAFILE 句
 - ALTER DATABASE, 8-13, 8-27
 - CREATE DATABASE, 12-27
- DATE データ型, 2-18
 - ユリウス, 2-20
- DATE 列
 - 日時列への変換, 10-57
- DAY 日時書式要素, 2-70
- DB_CACHE_SIZE パラメータ
 - ALTER SYSTEM, 9-68, 9-82
- DB_BLOCK_BUFFERS 初期化パラメータ
 - ALTER SYSTEM で設定, 9-43
- DB_BLOCK_CHECKING 初期化パラメータ
 - ALTER SESSION で設定, 9-7
 - ALTER SYSTEM で設定, 9-44
- DB_BLOCK_CHECKSUM 初期化パラメータ
 - ALTER SYSTEM で設定, 9-44
- DB_BLOCK_SIZE 初期化パラメータ
 - ALTER SYSTEM で設定, 9-45
- DB_CACHE_ADVICE 初期化パラメータ
 - ALTER SYSTEM で設定, 9-45
- DB_CACHE_SIZE 初期化パラメータ
 - ALTER SYSTEM で設定, 9-45, 9-68, 9-82
- DB_CREATE_FILE_DEST 初期化パラメータ
 - ALTER SESSION で設定, 9-7
 - ALTER SYSTEM で設定, 9-46
- DB_CREATE_ONLINE_LOG_DEST_n 初期化パラメータ
 - ALTER SESSION で設定, 9-7
 - ALTER SYSTEM で設定, 9-46
- DB_DOMAIN 初期化パラメータ
 - ALTER SYSTEM で設定, 9-46
- DB_FILE_MULTIBLOCK_READ_COUNT 初期化パラメータ
 - ALTER SESSION で設定, 9-7
 - ALTER SYSTEM で設定, 9-47
- DB_FILE_NAME_CONVERT 初期化パラメータ
 - ALTER SYSTEM で設定, 9-47
- DB_FILES 初期化パラメータ
 - ALTER SYSTEM で設定, 9-48
- DB_KEEP_CACHE_SIZE 初期化パラメータ
 - ALTER SYSTEM で設定, 9-48
- DB_NAME 初期化パラメータ
 - ALTER SYSTEM で設定, 9-49
- DB_nK_CACHE_SIZE 初期化パラメータ
 - ALTER SYSTEM で設定, 9-43
- DB_RECYCLE_CACHE_SIZE 初期化パラメータ
 - ALTER SYSTEM で設定, 9-49
- DB_WRITER_PROCESSES 初期化パラメータ
 - ALTER SYSTEM で設定, 9-49
- DB2 データ型, 2-34
 - Oracle データ型への変換, 2-35

- 暗黙的な変換, 2-35
- 制限, 2-35
- DBA_2PC_PENDING データ・ディクショナリ・ビュー, 9-3
- DBA_COL_COMMENTS データ・ディクショナリ・ビュー, 11-66
- DBA_ROLLBACK_SEGS データ・ディクショナリ・ビュー, 15-93
- DBA_TAB_COMMENTS データ・ディクショナリ・ビュー, 11-66
- DBA ロール, 16-48
- DBLINK_ENCRYPT_LOGIN 初期化パラメータ
 - ALTER SYSTEM で設定, 9-50
- DBMS_OUTPUT パッケージ, 11-4
- DBMS_ROWID パッケージ
 - 拡張 ROWID, 2-32
- DBMSSTD.SQL スクリプト, 12-48, 13-46, 13-51, 13-59
 - トリガー, 14-78
- DBTIMEZONE ファンクション, 6-46
- DBWR_IO_SLAVES 初期化パラメータ
 - ALTER SYSTEM で設定, 9-50
- DDAY 日時書式要素, 2-67
- DDD 日時書式要素, 2-67
- DDL
 - 「データ定義言語 (DDL)」を参照
- DD 日時書式要素, 2-67
- DEALLOCATE UNUSED 句
 - ALTER CLUSTER, 8-4, 8-6
 - ALTER INDEX, 8-50
 - ALTER TABLE, 10-32
- DEBUG 句
 - ALTER FUNCTION, 8-47
 - ALTER PACKAGE, 8-101
 - ALTER PROCEDURE, 8-104
 - ALTER TRIGGER, 11-4
 - ALTER TYPE, 11-9
- DECIMAL データ型
 - ANSI, 2-34
 - DB2, 2-35
 - SQL/DS, 2-35
- DECODE ファンクション, 6-47
- DECOMPOSE ファンクション, 6-48
- DEFAULT COST 句
 - ASSOCIATE STATISTICS, 11-47, 11-49
- DEFAULT ROLE 句
 - ALTER USER, 11-23
- DEFAULT SELECTIVITY 句
 - ASSOCIATE STATISTICS, 11-47, 11-49
- DEFAULT TABLESPACE 句
 - ALTER USER
 - 「CREATE USER」を参照
 - CREATE USER, 15-33
- DEFAULT TEMPORARY TABLESPACE 句
 - ALTER DATABASE, 8-37
 - CREATE DATABASE, 12-23
- default_cost_clause
 - ASSOCIATE STATISTICS, 11-47
- default_selectivity_clause
 - ASSOCIATE STATISTICS, 11-47
- default_temporary_tablespace_clause
 - CREATE DATABASE, 12-23
- DEFAULT 記憶域句
 - CREATE TABLESPACE, 14-68
- DEFAULT 句
 - ALTER TABLE, 10-56
 - CREATE TABLE, 14-21
- DEFAULT プロファイル
 - ユーザーへの割当て, 15-89
- DEFAULT 記憶域句
 - ALTER TABLESPACE, 10-87
- DEFERRABLE 句
 - constraint_clause, 11-85
- DEFERRED 句
 - SET CONSTRAINTS, 17-42
- DELETE ANY TABLE システム権限, 16-43
- DELETE STATISTICS 句
 - ANALYZE, 11-43
- DELETE_CATALOG_ROLE ロール, 16-48
- DELETE オブジェクト権限, 16-49
 - ビュー, 16-50
 - 表, 16-50
- DELETE 文, 15-49
 - トリガー, 14-82
- DENSE_RANK ファンクション, 6-49
- DEREF ファンクション, 6-51
- DESC 句
 - CREATE INDEX, 12-69
- DETERMINISTIC 句
 - CREATE FUNCTION, 12-53
- DISABLE [constraint] 句
 - CREATE TABLE, 11-89, 14-45
- DISABLE ALL TRIGGERS 句
 - ALTER TABLE, 10-71

DISABLE DISTRIBUTED RECOVERY 句
 ALTER SYSTEM, 9-26
 DISABLE NOVALIDATE 制約状態, 14-45
 DISABLE NOVALIDATE 制約文, 11-89
 DISABLE PARALLEL DML 句
 ALTER SESSION, 9-4
 DISABLE QUERY REWRITE 句
 ALTER MATERIALIZED VIEW, 8-86
 CREATE MATERIALIZED VIEW, 13-21
 DISABLE RESTRICTED SESSION 句
 ALTER SYSTEM, 9-27
 DISABLE RESUMABLE 句
 ALTER SESSION, 9-6
 DISABLE ROW MOVEMENT 句
 CREATE TABLE, 14-12, 14-41
 ENABLE ROW MOVEMENT 句
 ALTER TABLE, 10-17, 10-54
 DISABLE STORAGE IN ROW 句
 ALTER TABLE, 10-66
 CREATE TABLE, 14-33
 DISABLE TABLE LOCK 句
 ALTER TABLE, 10-71
 DISABLE THREAD 句
 ALTER DATABASE, 8-38
 DISABLE VALIDATE 制約状態, 14-45
 DISABLE VALIDATE 制約文, 11-89
 DISABLE 句
 ALTER INDEX, 8-62
 ALTER TRIGGER, 11-3
 CREATE TABLE, 14-44
 DISASSOCIATE STATISTICS 文, 15-57
 DISCONNECT SESSION 句
 ALTER SYSTEM, 9-25
 DISK_ASYNCH_IO 初期化パラメータ
 ALTER SYSTEM で設定, 9-50
 DISPATCHERS 初期化パラメータ
 ALTER SYSTEM で設定, 9-51
 DISTINCT 句
 SELECT, 17-11
 DISTRIBUTED_TRANSACTIONS 初期化パラメータ
 ALTER SYSTEM で設定, 9-52
 DML
 「データ操作言語 (DML)」を参照
 dml_event_clause
 CREATE TRIGGER, 14-79
 domain_index_clause
 CREATE INDEX, 12-61
 DOUBLE PRECISION データ型 (ANSI), 2-34
 DROP ANY CLUSTER システム権限, 16-38
 DROP ANY CONTEXT システム権限, 16-38
 DROP ANY DIMENSION システム権限, 16-38
 DROP ANY DIRECTORY システム権限, 16-39
 DROP ANY INDEXTYPE システム権限, 16-39
 DROP ANY INDEX システム権限, 16-39
 DROP ANY LIBRARY システム権限, 16-39
 DROP ANY MATERIALIZED VIEW システム権限,
 16-40
 DROP ANY OPERATOR システム権限, 16-40
 DROP ANY OUTLINE システム権限, 16-41
 DROP ANY PROCEDURE システム権限, 16-41
 DROP ANY ROLE システム権限, 16-41
 DROP ANY SEQUENCE システム権限, 16-42
 DROP ANY SYNONYM システム権限, 16-42
 DROP ANY TABLE システム権限, 16-43
 DROP ANY TRIGGER システム権限, 16-44
 DROP ANY TYPE システム権限, 16-44
 DROP ANY VIEW システム権限, 16-45
 DROP CLUSTER 文, 15-60
 DROP COLUMN 句
 ALTER TABLE, 10-58
 DROP CONSTRAINT 句
 ALTER TABLE, 10-63
 DROP CONTEXT 文, 15-62
 DROP DATABASE LINK 文, 15-63
 DROP DIMENSION 文, 15-65
 DROP DIRECTORY 文, 15-67
 DROP FUNCTION 文, 15-69
 drop_index_partition_clause
 ALTER INDEX, 8-53
 DROP INDEXTYPE 文, 15-73
 DROP INDEX 文, 15-71
 DROP JAVA 文, 15-75
 DROP LIBRARY システム権限, 16-39
 DROP LIBRARY 文, 15-77
 DROP LOG GROUP 句
 ALTER TABLE, 10-30
 DROP LOGFILE MEMBER 句
 ALTER DATABASE, 8-15, 8-32
 DROP LOGFILE 句
 ALTER DATABASE, 8-15, 8-31
 DROP MATERIALIZED VIEW LOG 文, 15-80
 DROP MATERIALIZED VIEW 文, 15-78
 DROP OPERATOR 文, 15-82
 DROP OUTLINE 文, 15-84

- DROP PACKAGE BODY 文, 15-85
- DROP PACKAGE 文, 15-85
- DROP PARTITION 句
 - ALTER INDEX, 8-53, 8-65
 - ALTER TABLE, 10-47
- DROP PRIMARY 制約句
 - ALTER TABLE, 10-63
- DROP PROCEDURE 文, 15-87
- DROP PROFILE システム権限, 16-41
- CREATE PROFILE 文, 15-89
- DROP PUBLIC DATABASE LINK システム権限, 16-38
- DROP PUBLIC SYNONYM システム権限, 16-42
- DROP ROLE 文, 15-91
- DROP ROLLBACK SEGMENT システム権限, 16-42
- DROP ROLLBACK SEGMENT 文, 15-92
- DROP SEQUENCE 文, 16-2
- DROP SUPPLEMENTAL LOG DATA 句
 - ALTER DATABASE, 8-32
- DROP SYNONYM 文, 16-4
- DROP TABLESPACE システム権限, 16-43
- DROP TABLESPACE 文, 16-9
- DROP TABLE 文, 16-6
- DROP TRIGGER 文, 16-12
- DROP TYPE BODY 文, 16-17
- DROP TYPE 文, 16-14
- DROP UNIQUE 制約句
 - ALTER TABLE, 10-63
- DROP USER システム権限, 16-45
- DROP USER 文, 16-19
- DROP VALUES 句
 - ALTER TABLE MODIFY PARTITION, 10-40
- DROP VIEW 文, 16-21
- drop_constraint_clause
 - ALTER TABLE, 10-19
- DROP 句
 - ALTER DIMENSION, 8-45
 - ALTER INDEXTYPE, 8-70
- DROP 制約句
 - ALTER VIEW, 11-30
- DROP 文
 - トリガー, 14-83
- DRS_START 初期化パラメータ
 - ALTER SYSTEM で設定, 9-52
- DUAL ダミー表, 2-107, 7-14
- DUMP ファンクション, 6-52
- DY 日時書式要素, 2-67, 2-70

E

- E
 - 数値書式要素, 2-62
- EBCDIC キャラクタ・セット, 2-44
- EE 日時書式要素, 2-67
- EMPTY_BLOB ファンクション, 6-54
- EMPTY_CLOB ファンクション, 6-54
- ENABLE ALL TRIGGERS 句
 - ALTER TABLE, 10-71
- enable_disable_clause
 - CREATE TABLE, 14-17
- ENABLE DISTRIBUTED RECOVERY 句
 - ALTER SYSTEM, 9-26
- ENABLE NOVALIDATE 制約状態, 14-44
- ENABLE NOVALIDATE 制約文, 11-88
- ENABLE PARALLEL DML 句
 - ALTER SESSION, 9-4
- ENABLE QUERY REWRITE 句
 - ALTER MATERIALIZED VIEW, 8-86
 - CREATE MATERIALIZED VIEW, 13-21
- ENABLE RESTRICTED SESSION 句
 - ALTER SYSTEM, 9-27
- ENABLE RESUMABLE 句
 - ALTER SESSION, 9-6
- ENABLE ROW MOVEMENT 句
 - ALTER TABLE, 10-17, 10-54
 - CREATE TABLE, 14-12, 14-41
- ENABLE STORAGE IN ROW 句
 - ALTER TABLE, 10-66
 - CREATE TABLE, 14-33
- ENABLE TABLE LOCK 句
 - ALTER TABLE, 10-70
- ENABLE THREAD 句
 - ALTER DATABASE, 8-38
- ENABLE VALIDATE 制約状態, 14-44
- ENABLE VALIDATE 制約文, 11-88
- enable_disable_clause
 - ALTER TABLE, 10-70
- ENABLE 句
 - ALTER INDEX, 8-62
 - ALTER TRIGGER, 11-3
 - CREATE TABLE, 14-44
- END BACKUP 句
 - ALTER DATABASE DATAFILE, 8-27
 - ALTER TABLESPACE, 10-89

ERROR_ON_OVERLAP_TIME セッション・パラメータ, 9-11
ESTIMATE STATISTICS 句
ANALYZE, 11-39
EVENT 初期化パラメータ
ALTER SYSTEM で設定, 9-52
EXCEPTIONS INTO 句
ALTER TABLE, 10-52
制限事項, 10-53
EXCHANGE PARTITION 句
ALTER TABLE, 10-17, 10-51
EXCHANGE SUBPARTITION 句
ALTER TABLE, 10-17, 10-51
exchange_table_partition 句
ALTER TABLE, 10-17
exchange_table_subpartition 句
ALTER TABLE, 10-17
EXCLUDING NEW VALUES 句
ALTER MATERIALIZED VIEW LOG, 8-96
CREATE MATERIALIZED VIEW LOG, 13-35
EXCLUSIVE ロック・モード, 16-74
EXECUTE ANY INDEXTYPE システム権限, 16-39
EXECUTE ANY OPERATOR システム権限, 16-40
EXECUTE ANY PROCEDURE システム権限, 16-41
EXECUTE ANY TYPE システム権限, 16-44
EXECUTE_CATALOG_ROLE ロール, 16-48
EXECUTE オブジェクト権限, 16-49
演算子, 16-52
オブジェクト型, 16-52
索引タイプ, 16-52
ファンクション、プロシージャまたはパッケージ,
16-51
EXEMPT ACCESS POLICY システム権限, 16-46
EXISTSNode ファンクション, 6-54
EXISTS 条件, 5-12
EXP_FULL_DATABASE ロール, 16-48
EXPLAIN PLAN 文, 16-23
EXP ファンクション, 6-55
EXTENT MANAGEMENT DICTIONARY 句
CREATE TABLESPACE, 14-69
EXTENT MANAGEMENT LOCAL 句
CREATE TABLESPACE, 14-69
CREATE TEMPORARY TABLESPACE, 14-76
EXTENT MANAGEMENT 句
CREATE DATABASE, 12-23
CREATE TABLESPACE, 14-64, 14-69
CREATE TEMPORARY TABLESPACE, 14-74

一時表領域, 14-75
EXTRACT (XML) ファンクション, 6-57
EXTRACT (日時) ファンクション, 6-56
E 日時書式要素, 2-67

F

FAILED_LOGIN_ATTEMPTS パラメータ
ALTER PROFILE, 8-107
CREATE PROFILE, 13-69
FAL_CLIENT 初期化パラメータ
ALTER SYSTEM で設定, 9-53
FAL_SERVER 初期化パラメータ
ALTER SYSTEM で設定, 9-53
FAST_START_IO_TARGET 初期化パラメータ
ALTER SESSION で設定, 9-53
FAST_START_MTTR_TARGET 初期化パラメータ
ALTER SYSTEM で設定, 9-54
FAST_START_PARALLEL_ROLLBACK 初期化パラメータ
ALTER SYSTEM で設定, 9-54
FF 日時書式要素, 2-67
filespec 句, 16-27
CREATE CONTROLFILE, 12-16
CREATE DATABASE, 12-24
CREATE LIBRARY, 13-2
CREATE TABLESPACE, 14-63
CREATE TEMPORARY TABLESPACE, 14-73
FINAL 句
CREATE TYPE, 15-13
FIPS
準拠, B-10
フラグ付け, 9-11
FIRST_ROWS(n) ヒント, 2-90
FIRST_VALUE ファンクション, 6-60
FIRST ファンクション, 6-58
FIXED_DATE 初期化パラメータ
ALTER SYSTEM で設定, 9-55
FLAGGER セッション・パラメータ, 9-11
FLOAT データ型, 2-14
DB2, 2-35
SQL/DS, 2-35
FLOAT データ型 (ANSI), 2-34
FLOOR ファンクション, 6-62
FLUSH SHARED POOL 句
ALTER SYSTEM, 9-27
FM 書式モデル修飾子, 2-73

- FM 数値書式要素, 2-62
- FOR EACH ROW 句
 - CREATE TRIGGER, 14-86
- FOR UPDATE 句
 - CREATE MATERIALIZED VIEW, 13-21
 - SELECT, 17-10, 17-23
- FORCE ANY TRANSACTION システム権限, 16-46
- FORCE PARALLEL DML 句
 - ALTER SESSION, 9-4
- FORCE TRANSACTION システム権限, 16-46
- FORCE 句
 - COMMIT, 11-70
 - CREATE VIEW, 15-40
 - DISASSOCIATE STATISTICS, 15-59
 - DROP INDEX, 15-72
 - DROP INDEXTYPE, 15-74
 - DROP OPERATOR, 15-83
 - DROP TYPE, 16-15
 - REVOKE, 16-91
 - ROLLBACK, 16-98
- FOREIGN KEY 句
 - constraint_clause, 11-82
- FOR 句
 - COMPUTE STATISTICS, 11-37
 - CREATE INDEXTYPE, 12-85
 - ESTIMATE STATISTICS, 11-37
 - EXPLAIN PLAN, 16-25
- FREELIST GROUPS パラメータ
 - STORAGE 句, 17-54
- FREELISTS パラメータ
 - STORAGE 句, 17-55
- FROM COLUMNS 句
 - DISASSOCIATE STATISTICS, 15-59
- FROM FUNCTIONS 句
 - DISASSOCIATE STATISTICS, 15-59
- FROM INDEXES 句
 - DISASSOCIATE STATISTICS, 15-59
- FROM INDEXTYPES 句
 - DISASSOCIATE STATISTICS, 15-59
- FROM PACKAGES 句
 - DISASSOCIATE STATISTICS, 15-59
- FROM TYPES 句
 - DISASSOCIATE STATISTICS, 15-59
- FROM_TZ ファンクション, 6-62
- FROM 句
 - 問合せ, 7-10
- FULL ヒント, 2-91

- FUNCTIONS 句
 - ASSOCIATE STATISTICS, 11-47, 11-48
- FX 書式モデル修飾子, 2-73

G

- GC_FILES_TO_LOCKS 初期化パラメータ
 - ALTER SYSTEM で設定, 9-55
- GLOBAL PARTITION BY RANGE 句
 - CREATE INDEX, 11-87, 12-62, 12-73, 14-47
- GLOBAL QUERY REWRITE システム権限, 16-39, 16-40
- GLOBAL TEMPORARY 句
 - CREATE TABLE, 14-19
- GLOBAL_CONTEXT_POOL_SIZE 初期化パラメータ
 - ALTER SYSTEM で設定, 9-56
- GLOBAL_NAMES 初期化パラメータ
 - ALTER SESSION で設定, 9-7
 - ALTER SYSTEM で設定, 9-56
- global_partitioned_index_clause
 - CREATE INDEX, 12-62
- GRANT ANY PRIVILEGE システム権限, 16-46
- GRANT ANY ROLE システム権限, 16-41
- GRANT CONNECT THROUGH 句
 - ALTER USER, 11-22, 11-24
- grantee_clause
 - REVOKE, 16-88
- GRANT 句
 - ALTER USER, 11-24
- GRAPHIC データ型
 - DB2, 2-35
 - SQL/DS, 2-35
- GREATEST ファンクション, 6-63
- GROUP BY 句
 - CUBE 拡張, 17-19
 - ROLLUP 拡張, 17-19
 - SELECT および副問合せ, 17-9, 17-19
 - 重複するグループ化の識別, 6-64
- GROUP_ID ファンクション, 6-64
- GROUPING SETS 句
 - SELECT および副問合せ, 17-20
- GROUPING_ID ファンクション, 6-66
- GROUPING ファンクション, 6-65
- G 数値書式要素, 2-62

H

HASH IS 句

- CREATE CLUSTER, 12-7

HASH_AREA_SIZE 初期化パラメータ

- ALTER SYSTEM で設定, 9-56

HASH_AJ ヒント, 2-91

HASH_AREA_SIZE 初期化パラメータ

- ALTER SESSION で設定, 9-7

HASH_JOIN_ENABLED 初期化パラメータ

- ALTER SESSION で設定, 9-7

- ALTER SYSTEM で設定, 9-57

HASHKEYS 句

- CREATE CLUSTER, 12-6

HASH ヒント, 2-91

HAVING 条件

- GROUP BY 句, 17-20

HEXTORAW ファンクション, 6-68

HH12 日時書式要素, 2-67

HH24 日時書式要素, 2-67

HH 日時書式要素, 2-67

HI_SHARED_MEMORY_ADDRESS 初期化パラメータ

- ALTER SYSTEM で設定, 9-57

HIERARCHY 句

- CREATE DIMENSION, 12-40, 12-41

HS_ADMIN_ROLE ロール, 16-48

HS_AUTOREGISTER 初期化パラメータ

- ALTER SYSTEM で設定, 9-57

I

IDENTIFIED BY 句

- ALTER ROLE

- 「CREATE ROLE」を参照

- CREATE DATABASE LINK, 12-36

- SET ROLE, 17-44

IDENTIFIED EXTERNALLY 句

- ALTER ROLE

- 「CREATE ROLE」を参照

- ALTER USER

- 「CREATE USER」を参照

- CREATE ROLE, 13-73

- CREATE USER, 15-32

IDENTIFIED GLOBALLY 句

- ALTER ROLE

- 「CREATE ROLE」を参照

- ALTER USER, 11-23

- CREATE ROLE, 13-73

- CREATE USER, 15-32

IDLE_TIME パラメータ

- ALTER PROFILE, 8-107

IFILE 初期化パラメータ

- ALTER SYSTEM で設定, 9-58

IMMEDIATE 句

- SET CONSTRAINTS, 17-41

IMP_FULL_DATABASE ロール, 16-48

implementation_clause

- CREATE OPERATOR, 13-39

IN OUT パラメータ

- CREATE FUNCTION, 12-51

- CREATE PROCEDURE, 13-61

INCLUDING CONTENTS 句

- DROP TABLESPACE, 16-10

INCLUDING DATAFILES 句

- ALTER DATABASE TEMPFILE DROP 句, 8-28

INCLUDING NEW VALUES 句

- ALTER MATERIALIZED VIEW LOG, 8-96

- CREATE MATERIALIZED VIEW LOG, 13-35

INCLUDING TABLES 句

- DROP CLUSTER, 15-61

INCREMENT BY 句

- ALTER SEQUENCE

- 「CREATE SEQUENCE」を参照

INCREMENT BY パラメータ

- CREATE SEQUENCE, 13-83

INDEX_ASC ヒント, 2-92

INDEX_DESC ヒント, 2-92

index_subpartition_clause

- CREATE INDEX, 12-64

INDEXES 句

- ASSOCIATE STATISTICS, 11-47, 11-48

INDXTYPES 句

- ASSOCIATE STATISTICS, 11-47, 11-48

INDXTYPE 句

- CREATE INDEX, 12-61, 12-76

- コメント, 11-68

INDEX オブジェクト権限, 16-49

- 表, 16-50

INDEX 句

- ANALYZE, 11-34

- CREATE CLUSTER, 12-6

INDEX ヒント, 2-91

INITCAP ファンクション, 6-69

INITIALIZED EXTERNALLY 句
 CREATE CONTEXT, 12-12
 INITIALIZED GLOBALLY 句
 CREATE CONTEXT, 12-12
 INITIALLY DEFERRED 句
 constraint_clause, 11-85
 INITIALLY IMMEDIATE 句
 constraint_clause, 11-85
 INITIAL パラメータ
 STORAGE 句, 17-52
 INITTRANS パラメータ
 ALTER CLUSTER, 8-5
 ALTER INDEX, 8-50, 8-56
 ALTER MATERIALIZED VIEW, 8-79
 ALTER MATERIALIZED VIEW LOG, 8-92
 ALTER TABLE, 10-28
 CREATE INDEX
 「CREATE TABLE」を参照
 CREATE MATERIALIZED VIEW
 「CREATE TABLE」を参照
 CREATE MATERIALIZED VIEW LOG
 「CREATE TABLE」を参照
 CREATE TABLE, 14-24
 INSERT ANY TABLE システム権限, 16-43
 INSERT オブジェクト権限, 16-49
 ビュー, 16-50
 表, 16-50
 INSERT 句
 MERGE, 16-77
 INSERT 文, 16-56
 APPEND, 2-89
 トリガー, 14-82
 INSTANCE_GROUPS 初期化パラメータ
 ALTER SYSTEM で設定, 9-58
 INSTANCE_NAME 初期化パラメータ
 ALTER SYSTEM で設定, 9-59
 INSTANCE_NUMBER 初期化パラメータ
 ALTER SYSTEM で設定, 9-59
 INSTANCE セッション・パラメータ, 9-12
 INSTANTIABLE 句
 CREATE TYPE, 15-13
 INSTEAD OF 句
 CREATE TRIGGER, 14-81
 INSTEAD OF トリガー, 14-81
 INSTR2 ファンクション, 6-70
 INSTR4 ファンクション, 6-70
 INSTRB ファンクション, 6-70
 INSTRC ファンクション, 6-70
 INSTR ファンクション, 6-70
 INTEGER データ型
 ANSI, 2-34
 DB2, 2-35
 SQL/DS, 2-35
 INTERSECT 集合演算子, 3-5, 17-21
 INTERVAL DAY TO SECOND データ型, 2-22
 INTERVAL YEAR TO MONTH データ型, 2-22
 INTO 句
 EXPLAIN PLAN, 16-24
 INSERT, 16-60
 INT データ型 (ANSI), 2-34
 INVALIDATE GLOBAL INDEXES 句
 ALTER TABLE, 10-41
 invoker_rights_clause
 ALTER JAVA, 8-72
 CREATE FUNCTION, 12-52
 CREATE JAVA, 12-87, 12-89
 CREATE PACKAGE, 13-47
 CREATE PROCEDURE, 13-60
 CREATE TYPE, 11-13, 15-9
 IN パラメータ
 CREATE PROCEDURE, 13-61
 CREATE FUNCTION, 12-51
 IN リスト, 2-101
 IS NOT NULL 演算子, 5-12
 IS NULL 演算子, 5-12
 IS OF *type* 条件, 5-17
 ISOLATION_LEVEL セッション・パラメータ, 9-12
 ISO (国際標準化機構), B-1
 規格, xv, 1-2, B-2
 IW 日時書式要素, 2-67
 IYYY 日時書式要素, 2-67
 IYY 日時書式要素, 2-67
 IY 日時書式要素, 2-67
 I 日時書式要素, 2-67

J

Java
 Java ソース・スキーマ・オブジェクト
 作成, 12-88
 記憶形式
 CustomDatum, 15-11
 SQLData, 15-11

クラス
 削除, 15-75
 作成, 12-86, 12-88
 変換, 8-71, 12-88
スキーマ・オブジェクト
 名前解決, 12-90
ソース
 コンパイル, 8-71, 12-88
 削除, 15-75
 作成, 12-86
メソッド
 戻り型, 15-14
リソース
 削除, 15-75
 作成, 12-86, 12-89
JAVA_MAX_SESSIONSPACE_LIMIT 初期化パラメータ
 ALTER SYSTEM で設定, 9-60
JAVA_MAX_SESSIONSPACE_SIZE 初期化パラメータ
 ALTER SYSTEM で設定, 9-60
JAVA_POOL_SIZE 初期化パラメータ
 ALTER SYSTEM で設定, 9-60
JAVA 句
 CREATE TYPE, 15-14
 CREATE TYPE BODY, 15-27
Java メソッド
 オブジェクト型へのマップ, 15-14
JOB_QUEUE_PROCESSES 初期化パラメータ
 ALTER SYSTEM で設定, 9-61
JOIN KEY 句
 ALTER DIMENSION, 8-44
 CREATE DIMENSION, 12-42
JOIN 句
 CREATE DIMENSION, 12-40
J 日時書式要素, 2-67

K

KILL SESSION 句
 ALTER SYSTEM, 9-26

L

LAG ファンクション, 6-71
LANGUAGE 句
 CREATE PROCEDURE, 13-63
 CREATE TYPE, 15-14
 CREATE TYPE BODY, 15-27
 LARGE_POOL_SIZE 初期化パラメータ
 ALTER SYSTEM で設定, 9-61
 LAST_DAY ファンクション, 6-75
 LAST_VALUE ファンクション, 6-76
 LAST ファンクション, 6-73
 LEAD ファンクション, 6-78
 LEAST ファンクション, 6-79
 LENGTH2 ファンクション, 6-80
 LENGTH4 ファンクション, 6-80
 LENGTHB ファンクション, 6-80
 LENGTHC ファンクション, 6-80
 LENGTH ファンクション, 6-80
 level_clause
 CREATE DIMENSION, 12-40
 LEVEL 疑似列, 2-82, 17-18
 階層問合せ, 2-82
 LEVEL 句
 ALTER DIMENSION, 8-44
 CREATE DIMENSION, 12-40, 12-41
 LICENSE_MAX_SESSIONS 初期化パラメータ
 ALTER SYSTEM で設定, 9-62
 LICENSE_SESSIONS_WARNING 初期化パラメータ
 ALTER SYSTEM で設定, 9-63
 LIKE 条件, 5-13
 LIST CHAINED ROWS 句
 ANALYZE, 11-42
 LN ファンクション, 6-81
 LOB
 CACHE READS 設定, 2-28
 外部, 2-25
 記憶域
 インライン, 14-33
 属性, 14-32
 特性, 14-25
 キャッシュへの値の保存, 10-65, 14-42
 索引, 14-34
 処理されるバイト数, 14-33
 属性の初期化, 2-26
 ディレクトリの指定, 12-44
 内部, 2-25
 表領域
 定義, 14-25
 物理属性の変更, 10-67
 列
 LONG と LONG RAW の違い, 2-26
 移入, 2-26

- ロギング属性, 14-25
- ロケータ, 2-25
- LOB 記憶域句
 - ALTER MATERIALIZED VIEW, 8-77, 8-80
 - ALTER TABLE, 10-22, 10-65
 - CREATE MATERIALIZED VIEW, 13-11, 13-12, 13-14
 - CREATE TABLE, 14-11, 14-32
 - パーティション, 10-67
- LOB 索引句
 - ALTER TABLE, 10-67
 - CREATE TABLE, 14-34
- LOB データ型, 2-25
- LOB 列
 - LONG 列からの作成, 2-15, 10-57
 - 記憶域の変更, 10-65
 - 結合の制限, 7-10
 - 追加, 10-55
 - 変更, 10-56
 - マテリアライズド・ビューの記憶特性, 8-80
- LOCAL_LISTENER 初期化パラメータ
 - ALTER SYSTEM で設定, 9-63
- local_partitioned_index_clause
 - CREATE INDEX, 12-62
- LOCALTIMESTAMP ファンクション, 6-81
- LOCAL 句
 - CREATE INDEX, 12-62, 12-74
- LOCK ANY TABLE システム権限, 16-43
- LOCK TABLE 文, 16-72
- LOCK_NAME_SPACE 初期化パラメータ
 - ALTER SYSTEM で設定, 9-63
- LOCK_SGA 初期化パラメータ
 - ALTER SYSTEM で設定, 9-64
- LOG_ARCHIVE_START パラメータ
 - ALTER SYSTEM, 9-68
- LOG_ARCHIVE_DEST_n 初期化パラメータ
 - ALTER SESSION で設定, 9-8, 9-65
 - DELAY の設定のオーバーライド, 8-24
- LOG_ARCHIVE_DEST_STATE_n 初期化パラメータ
 - ALTER SESSION で設定, 9-8
 - ALTER SYSTEM で設定, 9-66
- LOG_ARCHIVE_DEST 初期化パラメータ
 - ALTER SYSTEM で設定, 9-64
- LOG_ARCHIVE_DUPLEX_DEST 初期化パラメータ
 - ALTER SYSTEM で設定, 9-66
- LOG_ARCHIVE_FORMAT 初期化パラメータ
 - ALTER SYSTEM で設定, 9-67
- LOG_ARCHIVE_MAX_PROCESSES 初期化パラメータ
 - ALTER SYSTEM で設定, 9-67
- LOG_ARCHIVE_MIN_SUCCEED_DEST 初期化パラメータ
 - ALTER SESSION で設定, 9-8
 - ALTER SYSTEM で設定, 9-68
- LOG_ARCHIVE_TRACE 初期化パラメータ
 - ALTER SYSTEM で設定, 9-68
- LOG_BUFFER 初期化パラメータ
 - ALTER SYSTEM で設定, 9-69
- LOG_CHECKPOINT_INTERVAL 初期化パラメータ
 - ALTER SYSTEM で設定, 9-69
- LOG_CHECKPOINT_TIMEOUT 初期化パラメータ
 - ALTER SYSTEM で設定, 9-70
- LOG_CHECKPOINTS_TO_ALERT 初期化パラメータ
 - ALTER SYSTEM で設定, 9-70
- LOG_FILE_NAME_CONVERT 初期化パラメータ
 - ALTER SYSTEM で設定, 9-71
- LOGFILE GROUP 句
 - CREATE CONTROLFILE, 12-16
- LOGFILE 句
 - ALTER DATABASE, 8-15
 - CREATE DATABASE, 12-25
- LOGGING 句
 - ALTER INDEX, 8-57
 - ALTER INDEX ... REBUILD, 8-61
 - ALTER MATERIALIZED VIEW, 8-82
 - ALTER MATERIALIZED VIEW LOG, 8-94
 - ALTER TABLE, 10-29
 - ALTER TABLESPACE, 10-90
 - CREATE MATERIALIZED VIEW, 13-14
 - CREATE MATERIALIZED VIEW LOG, 13-33
 - CREATE TABLE, 14-25
 - CREATE TABLESPACE, 14-67
- LOGICAL_READS_PER_CALL パラメータ
 - ALTER PROFILE, 8-107
- LOGICAL_READS_PER_SESSION パラメータ
 - ALTER PROFILE, 8-107
- ALTER RESOURCE COST, 8-111
- LOGMNR_MAX_PERSISTENT_SESSIONS 初期化パラメータ
 - ALTER SYSTEM で設定, 9-71
- LOGOFF データベース・イベント
 - トリガー, 14-84
- LOGON データベース・イベント
 - トリガー, 14-84
- LOG ファンクション, 6-82

LONG RAW データ型, 2-25
 CHAR データからの変換, 2-25
LONG VARCHAR データ型
 DB2, 2-35
 SQL/DS, 2-35
LONG VARCHARIC データ型
 DB2, 2-35
 SQL/DS, 2-35
LONG データ型, 2-15
 トリガー内, 2-16
LONG 列
 LOB への変換, 2-15, 10-57
 参照先, 2-15
 制限, 2-15
 テキスト列の格納, 2-15
 ドメイン索引, 10-57
 ビュー定義の格納, 2-15
LOWER ファンクション, 6-83
LPAD ファンクション, 6-83
LTRIM ファンクション, 6-84
L 数値書式要素, 2-62

M

MAKE_REF ファンクション, 6-85
MANAGE TABLESPACE システム権限, 16-43
MANAGED STANDBY RECOVERY 句
 ALTER DATABASE, 8-24
MAP MEMBER 句
 ALTER TYPE, 11-11
 CREATE TYPE, 15-28
MAPPING TABLE 句
 ALTER TABLE, 10-44, 10-69
MAP メソッド
 指定, 11-11
MATCHES 条件, 5-2
MAX_COMMIT_PROPAGATION_DELAY 初期化パラメータ
 ALTER SYSTEM で設定, 9-72
MAX_DISPATCHERS 初期化パラメータ
 ALTER SYSTEM で設定, 9-72
MAX_DUMP_FILE_SIZE 初期化パラメータ
 ALTER SESSION で設定, 9-8
 ALTER SYSTEM で設定, 9-73
MAX_ENABLED_ROLES 初期化パラメータ
 ALTER SYSTEM で設定, 9-73

MAX_SHARED_SERVERS 初期パラメータ
 ALTER SYSTEM で設定, 9-73
MAXDATAFILES パラメータ
 CREATE CONTROLFILE, 12-18
 CREATE DATABASE, 12-26
MAXEXTENTS パラメータ
 STORAGE 句, 17-54
MAXINSTANCES パラメータ
 CREATE CONTROLFILE, 12-18
 CREATE DATABASE, 12-26
MAXLOGFILES パラメータ
 CREATE CONTROLFILE, 12-17
 CREATE DATABASE, 12-25
MAXLOGHISTORY パラメータ
 CREATE CONTROLFILE, 12-17
 CREATE DATABASE, 12-26
MAXLOGMEMBERS パラメータ
 CREATE CONTROLFILE, 12-17
 CREATE DATABASE, 12-25
MAXSIZE 句
 ALTER DATABASE, 8-14
 CREATE DATABASE, 12-23
 CREATE TABLESPACE, 14-64
 CREATE TEMPORARY TABLESPACE, 14-74
MAXTRANS パラメータ
 ALTER CLUSTER, 8-5
 ALTER INDEX, 8-50, 8-56
 ALTER MATERIALIZED VIEW, 8-79
 ALTER MATERIALIZED VIEW LOG, 8-92
 ALTER TABLE, 10-28
 CREATE INDEX
 「CREATE TABLE」を参照
 CREATE MATERIALIZED VIEW
 「CREATE TABLE」を参照
 CREATE MATERIALIZED VIEW LOG
 「CREATE TABLE」を参照
 CREATE TABLE, 14-24
MAXVALUE パラメータ
 ALTER SEQUENCE
 「CREATE SEQUENCE」を参照
 CREATE SEQUENCE, 13-83
MAX ファンクション, 6-86
MEMBER 句
 ALTER TYPE, 11-11
 CREATE TYPE, 15-12
 CREATE TYPE BODY, 15-26

- MERGE PARTITIONS 句
 - ALTER TABLE, 10-50
- MERGE_AJ ヒント, 2-91
- merge_insert_clause
 - MERGE, 16-77
- merge_update_clause
 - MERGE, 16-76, 16-77
- MERGE ヒント, 2-93
- MERGE 文, 16-76
- MINEXTENTS パラメータ
 - STORAGE 句, 17-53
- MINIMIZE RECORDS PER BLOCK 句
 - ALTER TABLE, 10-34
- MINIMUM EXTENT 句
 - ALTER TABLESPACE, 10-87
 - CREATE TABLESPACE, 14-67
- MINUS 集合演算子, 3-5, 17-21
- MINVALUE パラメータ
 - ALTER SEQUENCE
 - 「CREATE SEQUENCE」を参照
 - CREATE SEQUENCE, 13-84
- MIN ファンクション, 6-88
- MI 数値書式要素, 2-62
- MI 日時書式要素, 2-67
- MM 日時書式要素, 2-67
- MODE 句
 - LOCK TABLE, 16-74
- MODIFY CONSTRAINT 句
 - ALTER TABLE, 10-19, 10-62
 - ALTER VIEW, 11-30
- MODIFY DEFAULT ATTRIBUTES 句
 - ALTER INDEX, 8-52, 8-64
 - ALTER TABLE, 10-38
- MODIFY LOB 記憶域句
 - ALTER MATERIALIZED VIEW, 8-77, 8-80
 - ALTER TABLE, 10-67
- MODIFY LOB 句
 - ALTER TABLE, 10-67
- MODIFY NESTED TABLE 句
 - ALTER TABLE, 10-19, 10-62
- MODIFY PARTITION 句
 - ALTER INDEX, 8-52, 8-65
 - ALTER MATERIALIZED VIEW, 8-81
 - ALTER TABLE, 10-39
- MODIFY scoped_table_ref_constraint 句
 - ALTER MATERIALIZED VIEW, 8-83

- MODIFY SUBPARTITION 句
 - ALTER INDEX, 8-53, 8-66
 - ALTER TABLE, 10-42
- MODIFY VARRAY 句
 - ALTER TABLE, 10-24, 10-67
- modify_collection_retrieval_clause
 - ALTER TABLE, 10-19
- modify_varray_storage_clause
 - ALTER TABLE, 10-24
- MODIFY 句
 - ALTER TABLE, 10-56
- MOD ファンクション, 6-89
- MONITORING USAGE 句
 - ALTER INDEX, 8-63
- MONITORING 句
 - ALTER TABLE, 10-33
 - CREATE TABLE, 14-43
- MONTHS_BETWEEN ファンクション, 6-90
- MONTH 日時書式要素, 2-67, 2-70
- MON 日時書式要素, 2-67, 2-70
- MOUNT 句
 - ALTER DATABASE, 8-18
- MOVE ONLINE 句
 - ALTER TABLE, 10-69
- MOVE PARTITION 句
 - ALTER TABLE, 10-43
- MOVE SUBPARTITION 句
 - ALTER TABLE, 10-44
- move_table_clause
 - ALTER TABLE, 10-25
- MOVE 句
 - ALTER TABLE, 10-25, 10-68
- MTS
 - 「共有サーバー」を参照
- MULTISET パラメータ
 - CAST ファンクション, 6-24

N

- NAMED 句
 - CREATE JAVA, 12-89
- NAME 句
 - SET TRANSACTION, 17-49
- NATIONAL CHAR VARYING データ型 (ANSI), 2-34
- NATIONAL CHARACTER SET パラメータ
 - ALTER DATABASE, 8-35

CREATE DATABASE, 12-27
 NATIONAL CHARACTER VARYING データ型
 (ANSI), 2-34
 NATIONAL CHARACTER データ型 (ANSI), 2-34
 NATIONAL CHAR データ型 (ANSI), 2-34
 NCHAR VARYING データ型 (ANSI), 2-34
 NCHAR データ型, 2-11
 ANSI, 2-34
 NCHR ファンクション, 6-91
 NCLOB データ型, 2-31
 トランザクションのサポート, 2-31
 NESTED TABLE 句
 ALTER TABLE, 10-21, 10-65
 CREATE TABLE, 14-10, 14-35
 CREATE TRIGGER, 14-85
 nested_table_col_properties 句
 ALTER TABLE, 10-21
 CREATE MATERIALIZED VIEW, 13-11
 NEW_TIME ファンクション, 6-92
 NEXT_DAY ファンクション, 6-93
 NEXTVAL 疑似列, 2-79, 13-81
 NEXT 句
 ALTER MATERIALIZED VIEW...REFRESH, 8-85
 NEXT パラメータ
 STORAGE 句, 17-52
 NL_SJ ヒント, 2-91
 NLS_CALENDAR 初期化パラメータ
 ALTER SESSION で設定, 9-8
 ALTER SYSTEM で設定, 9-74
 NLS_CHARSET_DECL_LEN ファンクション, 6-93
 NLS_CHARSET_ID ファンクション, 6-94
 NLS_CHARSET_NAME ファンクション, 6-95
 NLS_COMP 初期化パラメータ
 ALTER SESSION で設定, 9-8
 ALTER SYSTEM で設定, 9-74
 NLS_CURRENCY 初期化パラメータ
 ALTER SESSION で設定, 9-8
 ALTER SYSTEM で設定, 9-75
 NLS_DATE_FORMAT 初期化パラメータ
 ALTER SESSION で設定, 9-8
 ALTER SYSTEM で設定, 9-75
 NLS_DATE_LANGUAGE 初期化パラメータ, 2-70
 ALTER SESSION で設定, 9-8
 ALTER SYSTEM で設定, 9-75
 NLS_DUAL_CURRENCY 初期化パラメータ
 ALTER SESSION で設定, 9-8
 ALTER SYSTEM で設定, 9-76
 NLS_INITCAP ファンクション, 6-95
 NLS_ISO_CURRENCY 初期化パラメータ
 ALTER SESSION で設定, 9-8
 ALTER SYSTEM で設定, 9-76
 NLS_LANGUAGE 初期化パラメータ, 2-70, 7-9
 ALTER SESSION で設定, 9-8
 ALTER SYSTEM で設定, 9-76
 NLS_LENGTH_SEMANTICS 初期化パラメータ
 ALTER SESSION で設定, 9-8
 ALTER SYSTEM で設定, 9-77
 変更, 2-10
 NLS_LOWER ファンクション, 6-97
 NLS_NCHAR_CONV_EXCP 初期化パラメータ
 ALTER SESSION で設定, 9-8
 ALTER SYSTEM で設定, 9-77
 NLS_NUMERIC_CHARACTERS 初期化パラメータ
 ALTER SESSION で設定, 9-8
 ALTER SYSTEM で設定, 9-77
 NLS_SORT 初期化パラメータ, 7-9
 ALTER SESSION で設定, 9-8
 ALTER SYSTEM で設定, 9-78
 NLS_TERRITORY 初期化パラメータ, 2-70
 ALTER SESSION で設定, 9-8
 ALTER SYSTEM で設定, 9-78
 NLS_TIMESTAMP_FORMAT 初期化パラメータ
 ALTER SESSION で設定, 9-8
 ALTER SYSTEM で設定, 9-78
 NLS_TIMESTAMP_TZ_FORMAT 初期化パラメータ
 ALTER SESSION で設定, 9-8
 ALTER SYSTEM で設定, 9-79
 NLS_UPPER ファンクション, 6-99
 NLSSORT ファンクション, 6-97
 NO FORCE 句
 CREATE VIEW, 15-41
 NO_EXPAND ヒント, 2-94
 NO_INDEX ヒント, 2-95
 NO_MERGE ヒント, 2-95
 NO_PUSH_PRED ヒント, 2-95
 NOAPPEND ヒント, 2-94
 NOARCHIVELOG 句
 ALTER DATABASE, 8-15, 8-29
 CREATE CONTROLFILE, 12-18
 CREATE DATABASE, 8-20, 12-26
 NOAUDIT 文, 16-80
 NOCACHE 句
 ALTER CLUSTER, 8-7
 ALTER MATERIALIZED VIEW, 8-82

- ALTER MATERIALIZED VIEW LOG, 8-94
- ALTER SEQUENCE
 - 「CREATE SEQUENCE」を参照
- ALTER TABLE, 10-33, 14-42
- CREATE CLUSTER, 12-9
- CREATE MATERIALIZED VIEW, 13-14
- CREATE MATERIALIZED VIEW LOG, 13-33
- CREATE SEQUENCE, 13-84
- NOCACHE ヒント, 2-94
- NOCOMPRESS 句
 - ALTER INDEX ... REBUILD, 8-60
 - ALTER TABLE, 10-69
 - CREATE INDEX, 12-70
 - CREATE TABLE, 14-28
- NOCOPY 句
 - CREATE FUNCTION, 12-51
 - CREATE PROCEDURE, 13-61
- NOCYCLE パラメータ
 - ALTER SEQUENCE
 - 「CREATE SEQUENCE」を参照
 - CREATE SEQUENCE, 13-84
- NOFORCE 句
 - CREATE JAVA, 12-88
- NOLOGGING 句
 - ALTER INDEX, 8-57
 - ALTER INDEX ...REBUILD, 8-61
 - ALTER MATERIALIZED VIEW, 8-82
 - ALTER MATERIALIZED VIEW LOG, 8-94
 - ALTER TABLE, 10-29
 - ALTER TABLESPACE, 10-90
 - CREATE MATERIALIZED VIEW, 13-14
 - CREATE MATERIALIZED VIEW LOG, 13-33
 - CREATE TABLE, 14-25
 - CREATE TABLESPACE, 14-67
- NOMAXVALUE パラメータ
 - ALTER SEQUENCE
 - 「CREATE SEQUENCE」を参照
 - CREATE SEQUENCE, 13-83
- NOMINIMIZE RECORDS PER BLOCK 句
 - ALTER TABLE, 10-34
- NOMINVALUE パラメータ
 - ALTER SEQUENCE
 - 「CREATE SEQUENCE」を参照
 - CREATE SEQUENCE, 13-84
- NOMONITORING USAGE 句
 - ALTER INDEX, 8-63

- NOMONITORING 句
 - ALTER TABLE, 10-33
 - CREATE TABLE, 14-43
- NONE 句
 - SET ROLE, 17-45
- NOORDER パラメータ
 - ALTER SEQUENCE
 - 「CREATE SEQUENCE」を参照
 - CREATE SEQUENCE, 13-84
- NOPARALLEL_INDEX ヒント, 2-95
- NOPARALLEL 句
 - CREATE INDEX, 8-7, 8-23, 8-56, 8-81, 8-94, 10-41, 12-8, 12-73, 13-15, 13-33, 14-43
- NOPARALLEL ヒント, 2-95
- NORELY 句
 - constraint_clause, 11-86
- NORESETLOGS 句
 - CREATE CONTROLFILE, 12-17
- NOREVERSE パラメータ
 - ALTER INDEX ... REBUILD, 8-59
- NOREWRITE ヒント, 2-96
- NOROWDEPENDENCIES 句
 - CREATE CLUSTER, 12-8
 - CREATE TABLE, 14-42
- NOSORT 句
 - ALTER INDEX, 12-70
 - constraint_clause, 11-88
- NOT DEFERRABLE 句
 - constraint_clause, 11-85
- NOT FINAL 句
 - CREATE TYPE, 15-13
- NOT IDENTIFIED 句
 - ALTER ROLE
 - 「CREATE ROLE」を参照
 - CREATE ROLE, 13-73
- NOT INSTANTIABLE 句
 - CREATE TYPE, 15-13
- NOT NULL 句
 - constraint_clause, 11-80
 - CREATE TABLE, 14-21
- NOT NULL 制約, 11-80
- NOT 条件, 5-8
- NOWAIT 句
 - LOCK TABLE, 16-74
- NTILE ファンクション, 6-99
- NULL, 2-77
 - 0（ゼロ）との違い, 2-77

- 条件, 2-78
 - 表, 2-78
- 比較条件, 2-77
- ファンクション, 2-77
- NULLIF ファンクション, 6-101
 - CASE 式の書式, 6-101
- NULL 句
 - constraint_clause, 11-80
- NULL 条件, 5-12
- NUMBER データ型, 2-12
 - VARCHAR2 への変換, 2-61
 - 位取り, 2-12
 - 精度, 2-12
- NUMERIC データ型 (ANSI), 2-34
- NUMTODSINTERVAL ファンクション, 6-102
- NUMTOYMINTERVAL ファンクション, 6-103
- NVARCHAR2 データ型, 2-11
- NVL2 ファンクション, 6-105
- NVL ファンクション, 6-104

O

- O7_DICTIONARY_ACCESSIBILITY 初期化パラメータ
 - ALTER SYSTEM で設定, 9-79
- OBJECT IDENTIFIER 句
 - CREATE TABLE, 14-22
- OBJECT_CACHE_MAX_SIZE_PERCENT 初期化パラメータ
 - ALTER SESSION で設定, 9-8
 - ALTER SYSTEM で設定, 9-79
- OBJECT_CACHE_OPTIMAL_SIZE 初期化パラメータ
 - ALTER SESSION で設定, 9-8
 - ALTER SYSTEM で設定, 9-80
- object_type_col_properties 句
 - ALTER TABLE, 10-20, 10-64
 - CREATE MATERIALIZED VIEW, 13-11
- OF object_type 句
 - CREATE TABLE, 14-20
- OFFLINE 句
 - ALTER ROLLBACK SEGMENT, 8-116
 - ALTER TABLESPACE, 10-88
 - CREATE TABLESPACE, 14-68
- OF 句
 - CREATE VIEW, 15-42
- OID
 - 「オブジェクト識別子」を参照
- OIDINDEX 句
 - CREATE TABLE, 14-23
- ON COMMIT REFRESH オブジェクト権限, 16-49
 - マテリアライズド・ビュー, 16-51
- ON COMMIT REFRESH システム権限, 16-40
- ON COMMIT 句
 - CREATE TABLE, 14-22
- ON DATABASE 句
 - CREATE TRIGGER, 14-85
- ON DEFAULT 句
 - AUDIT, 11-55
 - NOAUDIT, 16-83
- ON DELETE CASCADE 句
 - constraint_clause, 11-82
- ON DELETE SET NULL 句
 - constraint_clause, 11-82
- ON DIRECTORY 句
 - AUDIT, 11-55
 - NOAUDIT, 16-83
- ON NESTED TABLE 句
 - CREATE TRIGGER, 14-85
- ON object 句
 - NOAUDIT, 16-83
 - REVOKE, 16-92
- ON PREBUILT TABLE
 - CREATE MATERIALIZED VIEW, 13-16
- ON SCHEMA 句
 - CREATE TRIGGER, 14-85
- on_composite_partitioned_table_clause
 - CREATE INDEX, 12-64
- on_hash_partitioned_table_clause
 - CREATE INDEX, 12-63
- on_list_partitioned_table_clause
 - CREATE INDEX, 12-63
- on_object_clause
 - REVOKE, 16-88
- on_range_partitioned_table_clause
 - CREATE INDEX, 12-63
- ONLINE 句
 - ALTER ROLLBACK SEGMENT, 8-116
 - ALTER TABLESPACE, 10-87
 - CREATE INDEX, 12-72
 - CREATE TABLESPACE, 14-68
- ONLINE パラメータ
 - ALTER INDEX ... REBUILD, 8-60
- ON 句
 - CREATE OUTLINE, 13-44

- OPEN NORESETLOGS 句
 - ALTER DATABASE, 8-19
- OPEN READ ONLY 句
 - ALTER DATABASE, 8-19
- OPEN READ WRITE 句
 - ALTER DATABASE, 8-18
- OPEN RESETLOGS 句
 - ALTER DATABASE, 8-19
- OPEN_CURSORS 初期化パラメータ
 - ALTER SYSTEM で設定, 9-80
- OPEN_LINKS_PER_INSTANCE 初期化パラメータ
 - ALTER SYSTEM で設定, 9-81
- OPEN_LINKS 初期化パラメータ
 - ALTER SYSTEM で設定, 9-81
- OPEN 句
 - ALTER DATABASE, 8-18
- OPERATOR 句
 - コメント, 11-68
- OPTIMAL パラメータ
 - STORAGE 句, 17-55
- OPTIMIZER_FEATURES_ENABLE 初期化パラメータ
 - ALTER SYSTEM で設定, 9-81
- OPTIMIZER_INDEX_CACHING 初期化パラメータ
 - ALTER SESSION で設定, 9-9
 - ALTER SYSTEM で設定, 9-82
- OPTIMIZER_INDEX_COST_ADJ 初期化パラメータ
 - ALTER SESSION で設定, 9-9
 - ALTER SYSTEM で設定, 9-82
- OPTIMIZER_MAX_PERMUTATIONS 初期化パラメータ
 - ALTER SESSION で設定, 9-9
 - ALTER SYSTEM で設定, 9-82
- OPTIMIZER_MODE 初期化パラメータ
 - ALTER SESSION で設定, 9-9
 - ALTER SYSTEM で設定, 9-83
- OR REPLACE 句
 - CREATE CONTEXT, 12-12
 - CREATE DIRECTORY, 12-45
 - CREATE FUNCTION, 12-50, 12-88
 - CREATE LIBRARY, 13-3
 - CREATE OUTLINE, 13-43
 - CREATE PACKAGE, 13-47
 - CREATE PACKAGE BODY, 13-52
 - CREATE PROCEDURE, 13-60
 - CREATE TRIGGER, 14-79
 - CREATE TYPE, 15-8
 - CREATE TYPE BODY, 15-26
 - CREATE VIEW, 15-40
- ORACLE_TRACE_COLLECTION_NAME 初期化パラメータ
 - ALTER SYSTEM で設定, 9-83
- ORACLE_TRACE_COLLECTION_PATH 初期化パラメータ
 - ALTER SYSTEM で設定, 9-84
- ORACLE_TRACE_COLLECTION_SIZE 初期化パラメータ
 - ALTER SYSTEM で設定, 9-84
- ORACLE_TRACE_ENABLE 初期化パラメータ
 - ALTER SYSTEM で設定, 9-84
- ORACLE_TRACE_FACILITY_NAME 初期化パラメータ
 - ALTER SYSTEM で設定, 9-85
- ORACLE_TRACE_FACILITY_PATH 初期化パラメータ
 - ALTER SYSTEM で設定, 9-85
- Oracle9i Text
 - CATSEARCH, 5-2
 - CONTAINS, 5-2
 - MATCHES, 5-2
 - SCORE 演算子, 3-2
 - 組込み条件, 5-2
 - ドメイン索引の作成, 12-76
- Oracle のツール製品
 - SQL のサポート, 1-5
- Oracle の予約語, C-1
- ORDER BY 句
 - SELECT, 7-9, 17-9, 17-21
 - ROWNUM, 2-84
 - 問合せ, 7-9
- ORDER MEMBER 句
 - ALTER TYPE, 11-11
 - CREATE TYPE BODY, 15-28
- ORDER SIBLINGS BY 句
 - SELECT, 17-21
- ORDERED_PREDICATES ヒント, 2-96
- ORDERED ヒント, 2-96
- ORDER 句
 - ALTER SEQUENCE
 - 「CREATE SEQUENCE」を参照
- ORDER パラメータ
 - CREATE SEQUENCE, 13-84
- ORDER メソッド
 - 指定, 11-11

ORGANIZATION EXTERNAL 句
CREATE TABLE, 14-27, 14-29

ORGANIZATION HEAP 句
CREATE TABLE, 14-27

ORGANIZATION INDEX 句
CREATE TABLE, 14-27

OR 条件, 5-8, 5-9

OS_AUTHENT_PREFIX 初期化パラメータ
ALTER SYSTEM で設定, 9-85

OS_ROLES 初期化パラメータ
ALTER SYSTEM で設定, 9-86

OUT パラメータ
CREATE FUNCTION, 12-51
CREATE PROCEDURE, 13-61

OVERFLOW 句
ALTER INDEX, 8-53
ALTER TABLE, 10-36
CREATE TABLE, 14-28

OVERRIDING 句
CREATE TYPE, 15-13

OVER 句
分析ファンクション, 6-8, 6-9

P

P.M. 日時書式要素, 2-67, 2-70

PACKAGES 句
ASSOCIATE STATISTICS, 11-47, 11-48
PARALLEL_ADAPTIVE_MULTI_USER 初期化パラメータ
ALTER SYSTEM で設定, 9-86
PARALLEL_AUTOMATIC_TUNING 初期化パラメータ
ALTER SYSTEM で設定, 9-86
PARALLEL_BROADCAST_ENABLED 初期化パラメータ
ALTER SESSION で設定, 9-9
ALTER SYSTEM で設定, 9-87
PARALLEL_ENABLE 句
CREATE FUNCTION, 12-53
PARALLEL_EXECUTION_MESSAGE_SIZE 初期化パラメータ
ALTER SYSTEM で設定, 9-87
PARALLEL_INSTANCE_GROUP 初期化パラメータ
ALTER SESSION で設定, 9-9
ALTER SYSTEM で設定, 9-88

PARALLEL_MAX_SERVERS 初期化パラメータ
ALTER SYSTEM で設定, 9-88

PARALLEL_MIN_PERCENT 初期化パラメータ
ALTER SESSION で設定, 9-9
ALTER SYSTEM で設定, 9-89

PARALLEL_MAX_SERVERS 初期化パラメータ
ALTER SYSTEM で設定, 9-89

PARALLEL_THREADS_PER_CPU 初期化パラメータ
ALTER SYSTEM で設定, 9-90

PARALLEL 句
ALTER CLUSTER, 8-4, 8-7
ALTER DATABASE, 8-23
ALTER INDEX, 8-50, 8-55
ALTER MATERIALIZED VIEW, 8-77, 8-81
ALTER MATERIALIZED VIEW LOG, 8-92, 8-93
ALTER TABLE, 10-41
CREATE CLUSTER, 12-8
CREATE INDEX, 12-72
CREATE MATERIALIZED VIEW, 13-11, 13-15
CREATE MATERIALIZED VIEW LOG, 13-32, 13-33
CREATE TABLE, 14-17, 14-43

PARALLEL ヒント, 2-97

PARAMETERS 句
ALTER INDEX ... REBUILD, 8-61
CREATE INDEX, 12-76

PARTITION ... LOB 記憶域句
ALTER TABLE, 10-67

PARTITION BY HASH 句
CREATE TABLE, 14-39

PARTITION BY LIST 句
CREATE TABLE, 14-38

PARTITION BY RANGE 句
CREATE TABLE, 14-14, 14-36

partition_storage_clause
ALTER TABLE, 10-23

PARTITION_VIEW_ENABLED 初期化パラメータ
ALTER SESSION で設定, 9-9
ALTER SYSTEM で設定, 9-90, 9-108

PARTITION 句
ANALYZE, 11-36
CREATE INDEX, 11-88, 12-74, 14-48
CREATE TABLE, 14-37
DELETE, 15-52
INSERT, 16-61
LOCK TABLE, 16-73
UPDATE, 17-68

PASSWORD EXPIRE 句
 ALTER USER
 「CREATE USER」を参照
 CREATE USER, 15-34

PASSWORD_GRACE_TIME パラメータ
 ALTER PROFILE, 8-107
 CREATE PROFILE, 13-69

PASSWORD_LIFE_TIME パラメータ
 ALTER PROFILE, 8-107
 CREATE PROFILE, 13-69

PASSWORD_LOCK_TIME パラメータ
 ALTER PROFILE, 8-107
 CREATE PROFILE, 13-69

PASSWORD_REUSE_MAX パラメータ
 ALTER PROFILE, 8-107
 CREATE PROFILE, 13-69

PASSWORD_REUSE_TIME パラメータ
 ALTER PROFILE, 8-107
 CREATE PROFILE, 13-69

PASSWORD_VERIFY_FUNCTION パラメータ
 ALTER PROFILE, 8-107
 CREATE PROFILE, 13-69

PCT_ACCESS_DIRECT 統計情報
 索引構成表, 11-35

PCTFREE パラメータ
 ALTER CLUSTER, 8-5
 ALTER INDEX, 8-50, 8-56
 ALTER MATERIALIZED VIEW, 8-79
 ALTER MATERIALIZED VIEW LOG, 8-92
 ALTER TABLE, 10-28
 CREATE MATERIALIZED VIEW
 「CREATE TABLE」を参照
 CREATE MATERIALIZED VIEW LOG
 「CREATE TABLE」を参照
 CREATE TABLE, 14-23

PCTINCREASE パラメータ
 STORAGE 句, 17-53

PCTTHRESHOLD パラメータ
 CREATE TABLE, 10-36, 14-28

PCTUSED パラメータ
 ALTER CLUSTER, 8-5
 ALTER INDEX, 8-50, 8-56
 ALTER MATERIALIZED VIEW, 8-79
 ALTER MATERIALIZED VIEW LOG, 8-92
 ALTER TABLE, 10-28
 CREATE INDEX
 「CREATE TABLE」を参照

CREATE MATERIALIZED VIEW
 「CREATE TABLE」を参照
 CREATE MATERIALIZED VIEW LOG
 「CREATE TABLE」を参照
 CREATE TABLE, 14-23

PCTVERSION パラメータ
 CREATE TABLE, 14-33

LOB 記憶域句, 10-66

PERCENT_RANK ファンクション, 6-106

PERCENTILE_CONT ファンクション, 6-108

PERCENTILE_DISC ファンクション, 6-110

PERMANENT 句
 ALTER TABLESPACE, 10-90
 CREATE TABLESPACE, 14-68

PGA_AGGREGATE_TARGET 初期化パラメータ
 ALTER SYSTEM で設定, 9-91

PIPELINED 句
 CREATE FUNCTION, 12-54

PL/SQL
 前回のリリースとの互換性, 9-93
 プログラム本体
 CREATE FUNCTION, 12-55

PLAN_TABLE サンプル表, 16-23

PLSQL_COMPILER_FLAGS 初期化パラメータ
 ALTER SESSION で設定, 9-9
 ALTER SYSTEM で設定, 9-91

PLSQL_DEBUG セッション・パラメータ, 9-12

PLSQL_NATIVE_C_COMPILER 初期化パラメータ
 ALTER SYSTEM で設定, 9-91

PLSQL_NATIVE_LIBRARY_DIR 初期化パラメータ
 ALTER SYSTEM で設定, 9-92

PLSQL_NATIVE_LIBRARY_SUBDIR_COUNT 初期化
 パラメータ
 ALTER SYSTEM で設定, 9-92

PLSQL_NATIVE_LINKER 初期化パラメータ
 ALTER SYSTEM で設定, 9-92

PLSQL_NATIVE_MAKE_FILE_NAME 初期化パラメータ
 ALTER SYSTEM で設定, 9-93

PLSQL_NATIVE_MAKE UTILITY 初期化パラメータ
 ALTER SYSTEM で設定, 9-93

PLSQL_V2_COMPATIBILITY 初期化パラメータ
 ALTER SYSTEM で設定, 9-93

PM 日時書式要素, 2-67, 2-70

POWER ファンクション, 6-111

PQ_DISTRIBUTE ヒント, 2-98

PRAGMA RESTRICT_REFERENCES, 11-11

PRAGMA 句

ALTER TYPE, 11-11

CREATE TYPE, 15-7, 15-15

PRE_PAGE_SGA 初期化パラメータ

ALTER SYSTEM で設定, 9-94

PREPARE TO SWITCHOVER 句

ALTER DATABASE, 8-35

PRIMARY KEY 句

constraint_clause, 11-80

CREATE TABLE, 14-21

PRIVATE_SGA パラメータ

ALTER PROFILE, 8-107

ALTER RESOURCE COST, 8-111

PRIVATE 句

CREATE OUTLINE, 13-44

PROCESSES 初期化パラメータ

ALTER SYSTEM で設定, 9-94

PROFILE 句

ALTER USER

「CREATE USER」を参照

CREATE USER, 15-34

PR 数値書式要素, 2-62

PUBLIC 句

CREATE OUTLINE, 13-44

CREATE ROLLBACK SEGMENT, 13-76

CREATE SYNONYM, 14-3

DROP DATABASE LINK, 15-63

PUSH_PRED ヒント, 2-99

Q

QUERY REWRITE オブジェクト権限, 16-49

マテリアライズド・ビュー, 16-51

QUERY REWRITE システム権限, 16-39, 16-40

QUERY_REWRITE_ENABLED 初期化パラメータ

ALTER SESSION で設定, 9-9

ALTER SYSTEM で設定, 9-94

QUERY_REWRITE_INTEGRITY 初期化パラメータ

ALTER SYSTEM で設定, 9-95

ALTER SESSION で設定, 9-9

QUIESCE RESTRICTED 句

ALTER SYSTEM, 9-28

QUOTA 句

ALTER USER

「CREATE USER」を参照

CREATE USER, 15-34

Q 日時書式要素, 2-67

R

RANK ファンクション, 6-112

RATIO_TO_REPORT ファンクション, 6-114

RAWTOHEX ファンクション, 6-115

RAWTONHEX ファンクション, 6-116

RAW データ型, 2-25

CHAR データからの変換, 2-25

RDBMS_SERVER_DN 初期化パラメータ

ALTER SYSTEM で設定, 9-95

READ ONLY 句

ALTER TABLESPACE, 10-89

READ WRITE 句

ALTER TABLESPACE, 10-89

READ_ONLY_OPEN_DELAYED 初期化パラメータ

ALTER SYSTEM で設定, 9-95

READ オブジェクト権限, 16-49

マテリアライズド・ディレクトリ, 16-51

REAL データ型 (ANSI), 2-34

REBUILD PARTITION 句

ALTER INDEX, 8-58

REBUILD SUBPARTITION 句

ALTER INDEX, 8-59

REBUILD UNUSABLE LOCAL INDEXES 句

ALTER TABLE, 10-40

REBUILD 句

ALTER INDEX, 8-51, 8-57

ALTER MATERIALIZED VIEW, 8-83

ALTER OUTLINE, 8-98

RECOVER AUTOMATIC 句

ALTER DATABASE, 8-20

RECOVER CANCEL 句

ALTER DATABASE, 8-11, 8-24

RECOVER CONTINUE 句

ALTER DATABASE, 8-11, 8-23

RECOVER DATABASE 句

ALTER DATABASE, 8-11, 8-21

RECOVER DATAFILE 句

ALTER DATABASE, 8-11, 8-22

RECOVER LOGFILE 句

ALTER DATABASE, 8-11, 8-22

RECOVER MANAGED STANDBY DATABASE 句

ALTER DATABASE, 8-12

RECOVER STANDBY DATAFILE 句

ALTER DATABASE, 8-22

RECOVER STANDBY TABLESPACE 句

ALTER DATABASE, 8-22

- RECOVER TABLESPACE 句
 - ALTER DATABASE, 8-11, 8-22
- RECOVERABLE, 8-57, 14-26
 - 「LOGGING 句」を参照
- recovery_clauses
 - ALTER DATABASE, 8-10
- RECOVERY_CATALOG_OWNER ロール, 16-48
- RECOVERY_PARALLELISM 初期化パラメータ
 - ALTER SYSTEM で設定, 9-96
- RECOVER 句
 - ALTER DATABASE, 8-20
- REDO ログ, 8-18
 - アーカイブ位置, 9-24
 - アーカイブ・モードの指定, 12-26
 - クラスタ・データベースの指定スレッドの使用可能化, 8-38
 - クラスタ・データベースの指定スレッドの使用禁止化, 8-38
 - グループの切替え, 9-27
 - サイズ, 16-29
 - 再利用, 16-29
 - 削除, 8-29, 8-31
 - 指定, 12-25, 16-27
 - メディア・リカバリ用, 8-22
 - 自動アーカイブ, 9-22
 - 起動, 9-24
 - 停止, 9-24
 - 自動名前生成, 8-20
 - 手動アーカイブ, 9-22
 - SCN, 9-22
 - グループ番号, 9-23
 - 現行, 9-23
 - 順序番号, 9-22
 - すべて, 9-24
 - 次, 9-24
 - ファイル名, 9-23
 - 消去, 8-29
 - スレッド, 9-22
 - スレッドの使用可能化および使用禁止化, 8-29
 - 追加, 8-29, 8-30
 - 変更の削除, 8-18
 - メンバー
 - 既存のグループへの追加, 8-30
 - 削除, 8-32
 - 名前の変更, 8-29
- REDO ログ・グループ
 - 補助 REDO ログ・グループの削除, 10-30
 - 補助 REDO ログ・グループの追加, 10-30
- REF, 2-37, 11-83
 - DANGLING, 11-40
 - OID のコンテナ, 2-37
 - 検証, 11-40
 - 更新, 11-40
- REFERENCES オブジェクト権限, 16-49
 - ビュー, 16-50
 - 表, 16-50
- REFERENCES 句
 - constraint_clause, 11-82
 - CREATE TABLE, 14-21
- REFERENCING 句
 - CREATE TRIGGER, 14-79, 14-86
- REFRESH COMPLETE 句
 - ALTER MATERIALIZED VIEW, 8-84
 - CREATE MATERIALIZED VIEW, 13-17
- REFRESH FAST 句
 - ALTER MATERIALIZED VIEW, 8-83
 - CREATE MATERIALIZED VIEW, 13-17
- REFRESH FORCE 句
 - ALTER MATERIALIZED VIEW, 8-84
 - CREATE MATERIALIZED VIEW, 13-17
- REFRESH ON COMMIT 句
 - ALTER MATERIALIZED VIEW, 8-84
 - CREATE MATERIALIZED VIEW, 13-17
- REFRESH ON DEMAND 句
 - ALTER MATERIALIZED VIEW, 8-85
 - CREATE MATERIALIZED VIEW, 13-17
- REFRESH 句
 - ALTER MATERIALIZED VIEW, 8-79, 8-83
 - CREATE MATERIALIZED VIEW, 13-10
- REFTOHEX ファンクション, 6-117
- REF 表制約, 11-74, 11-83
 - ALTER TABLE, 10-56
 - CREATE TABLE, 14-21
- REF ファンクション, 6-116
- REF 列
 - 指定, 14-21
 - 範囲の最限定, 8-83
 - 表や列レベルからの指定, 14-21
- REGISTER LOGFILE 句
 - ALTER DATABASE, 8-35
- REGISTER 句
 - ALTER SYSTEM, 9-29
- REGR_AVGX ファンクション, 6-118
- REGR_AVGY ファンクション, 6-118

REGR_COUNT ファンクション, 6-118
 REGR_INTERCEPT ファンクション, 6-118
 REGR_R2 ファンクション, 6-118
 REGR_SLOPE ファンクション, 6-118
 REGR_SXX ファンクション, 6-118
 REGR_SXY ファンクション, 6-118
 REGR_SYY ファンクション, 6-118
 RELY 句
 constraint_clause, 11-86
 REMOTE_ARCHIVE_ENABLE 初期化パラメータ
 ALTER SYSTEM で設定, 9-96
 REMOTE_DEPENDENCIES_MODE 初期化パラメータ
 ALTER SESSION で設定, 9-9
 ALTER SYSTEM で設定, 9-96
 REMOTE_LISTENER 初期化パラメータ
 ALTER SYSTEM で設定, 9-97
 REMOTE_LOGIN_PASSWORDFILE 初期化パラメータ
 ALTER SYSTEM で設定, 9-97
 制御ファイル, 12-14
 データベース, 12-21
 REMOTE_OS_AUTHENT 初期化パラメータ
 ALTER SYSTEM で設定, 9-97
 REMOTE_OS_ROLES 初期化パラメータ
 ALTER SYSTEM で設定, 9-98
 RENAME DATAFILE 句
 ALTER TABLESPACE, 10-85
 RENAME FILE 句
 ALTER DATABASE, 8-9, 8-29
 RENAME GLOBAL_NAME 句
 ALTER DATABASE, 8-38
 RENAME PARTITION 句
 ALTER INDEX, 8-52, 8-65
 ALTER TABLE, 10-42
 RENAME SUBPARTITION 句
 ALTER INDEX, 8-52, 8-65
 ALTER TABLE, 10-42
 RENAME 句
 ALTER INDEX, 8-62
 ALTER OUTLINE, 8-98
 ALTER TABLE, 10-35
 ALTER TRIGGER, 11-3
 RENAME 文, 16-85
 REPLACE AS OBJECT 句
 ALTER TYPE, 11-10
 REPLACE ファンクション, 6-126
 REPLICATION_DEPENDENCY_TRACKING 初期化パラメータ
 ALTER SYSTEM で設定, 9-98
 RESET COMPATIBILITY 句
 ALTER DATABASE, 8-37
 RESETLOGS パラメータ
 CREATE CONTROLFILE, 12-16
 RESOLVER 句
 ALTER JAVA CLASS, 8-72
 ALTER JAVA SOURCE, 8-72
 CREATE JAVA, 12-90
 RESOLVE 句
 ALTER JAVA CLASS, 8-72
 CREATE JAVA, 12-88
 RESOURCE_LIMIT 初期化パラメータ
 ALTER SYSTEM で設定, 9-98
 RESOURCE_MANAGER_PLAN 初期化パラメータ
 ALTER SYSTEM で設定, 9-99
 RESOURCE ロール, 16-48
 RESTRICT_REFERENCES プラグマ
 ALTER TYPE, 11-11
 RESTRICTED SESSION システム権限, 16-42
 RESUMABLE システム権限, 16-46
 RESUME 句
 ALTER SYSTEM, 9-27
 RETURNING 句
 DELETE, 15-54
 INSERT, 16-58, 16-65
 UPDATE, 17-66, 17-72
 RETURN 句
 CREATE FUNCTION, 12-52
 CREATE OPERATOR, 13-40
 CREATE TYPE, 15-14
 CREATE TYPE BODY, 15-28
 REUSE SETTINGS 句
 ALTER FUNCTION, 8-47
 ALTER PACKAGE, 8-101
 ALTER PROCEDURE, 8-105
 ALTER TRIGGER, 11-4
 ALTER TYPE, 11-10
 REUSE 句
 CREATE CONTROLFILE, 12-16
 filespec 句, 16-29
 REVERSE 句
 CREATE INDEX, 12-71
 REVERSE パラメータ
 ALTER INDEX ... REBUILD, 8-59

- REVOKE CONNECT THROUGH 句
 - ALTER USER, 11-22, 11-24
- REVOKE 句
 - ALTER USER, 11-24
- REVOKE 文, 16-87
- REWRITE ヒント, 2-99
- RM 日時書式要素, 2-67
- RNDS 属性
 - PRAGMA RESTRICT_REFERENCES, 15-15
- RNPS 属性
 - PRAGMA RESTRICT_REFERENCES, 15-15
- RN 数値書式要素, 2-62
- ROLLBACK_SEGMENTS 初期化パラメータ
 - ALTER SYSTEM で設定, 9-99
- ROLLBACK 文, 16-96
- ROLLUP 句
 - SELECT 文, 17-19
- ROUND ファンクション
 - 書式モデル, 6-195
 - 数値ファンクション, 6-127
 - 日付ファンクション, 6-128
- ROW EXCLUSIVE ロック・モード, 16-74
- ROW SHARE ロック・モード, 16-74
- ROW_CACHE_CURSORS 初期化パラメータ
 - ALTER SESSION で設定, 9-42
- ROW_LOCKING 初期化パラメータ
 - ALTER SYSTEM で設定, 9-100
- ROW_NUMBER ファンクション, 6-128
- ROWDEPENDENCIES 句
 - CREATE CLUSTER, 12-8
 - CREATE TABLE, 14-42
- ROWID
 - ROW 部分, 2-32
 - 外部キー表, 2-33
 - 拡張, 2-32
 - 基本, 2-32
 - 直接利用できない, 2-32
 - 索引構成表, 2-33
 - 使用方法, 2-83
 - 制限, 2-32
 - 互換性と移行, 2-33
 - 説明, 2-31
 - 非物理, 2-33
 - ファイル部分, 2-32
 - ブロック部分, 2-32
- ROWIDTOCHAR ファンクション, 6-130
- ROWIDTONCHAR ファンクション, 6-130

- ROWID 疑似列, 2-31, 2-33, 2-82
- ROWID データ型, 2-31
- ROWID ヒント, 2-99
- ROWNUM 疑似列, 2-83
 - 使用方法, 2-84
- RPAD ファンクション, 6-131
- RRRR 日時書式要素, 2-67
- RR 日時書式要素, 2-67, 2-71
 - 解析, 2-71
- RTRIM ファンクション, 6-131
- RULE ヒント, 2-100

S

- SAMPLE 句
 - SELECT, 17-14
 - SELECT および副問合せ, 17-7
- SAVEPOINT 文, 17-2
- SCC 日時書式要素, 2-67
- SCHEMA 句
 - CREATE JAVA, 12-89
- SCOPE FOR 句
 - ALTER MATERIALIZED VIEW, 8-78
 - CREATE MATERIALIZED VIEW, 13-12
- scoped_table_ref_constraint 句
 - ALTER MATERIALIZED VIEW, 8-78
- SCOPE 句
 - REF 列制約, 11-83
- SCORE 演算子, 3-2
- SEGMENT MANAGEMENT FREELISTS 句
 - CREATE TABLESPACE, 14-70
- SEGMENT MANAGEMENT PAGETABLE 句
 - CREATE TABLESPACE, 14-70
- segment_management_clause
 - CREATE TABLESPACE, 14-70
- SELECT ANY DICTIONARY システム権限, 16-46
- SELECT ANY OUTLINE システム権限, 16-41
- SELECT ANY SEQUENCE システム権限, 16-42
- SELECT ANY TABLE システム権限, 16-43
- SELECT_CATALOG_ROLE ロール, 16-48
- SELECT オブジェクト権限, 16-49
 - 順序, 16-50
 - ビュー, 16-50
 - 表, 16-50
 - マテリアライズド・ビュー, 16-51
- SELECT 構文のリスト, 7-3
 - 順序付け, 7-9

SELECT 文, 7-2, 17-4
 SERIAL_REUSE 初期化パラメータ
 ALTER SYSTEM で設定, 9-100
 SERVERERROR イベント
 トリガー, 14-84
 SERVICE_NAMES 初期化パラメータ
 ALTER SYSTEM で設定, 9-100
 SESSION_CACHED_CURSORS 初期化パラメータ
 ALTER SESSION で設定, 9-9
 ALTER SYSTEM で設定, 9-101
 SESSION_MAX_OPEN_FILES 初期化パラメータ
 ALTER SYSTEM で設定, 9-101
 SESSION_ROLES ビュー, 17-43
 SESSIONS_PER_USER パラメータ
 ALTER PROFILE, 8-107
 SESSIONS 初期化パラメータ
 ALTER SYSTEM で設定, 9-102
 SESSIONTIMEZONE ファンクション, 6-132
 SET CONSTRAINT(S) 文, 17-41
 SET Dangling TO NULL 句
 ANALYZE, 11-40
 SET DATABASE 句
 CREATE CONTROLFILE, 12-16
 SET ROLE 文, 17-43
 SET STANDBY DATABASE 句
 ALTER DATABASE, 8-34
 SET STATEMENT_ID 句
 EXPLAIN PLAN, 16-24
 SET TIME_ZONE 句
 ALTER DATABASE, 8-17, 8-36
 ALTER SESSION, 9-13
 CREATE DATABASE, 12-24
 SET TRANSACTION 文, 17-46
 SET UNUSED 句
 ALTER TABLE, 10-58
 SET 句
 ALTER SESSION, 9-6
 ALTER SYSTEM, 9-30
 UPDATE, 17-70
 SGA
 「システム・グローバル領域」を参照
 SGA_MAX_SIZE 初期化パラメータ
 ALTER SYSTEM で設定, 9-102
 SHADOW_CORE_DUMP 初期化パラメータ
 ALTER SYSTEM で設定, 9-102
 SHARE ROW EXCLUSIVE ロック・モード, 16-74
 SHARE UPDATE ロック・モード, 16-74
 SHARED_MEMORY_ADDRESS 初期化パラメータ
 ALTER SYSTEM で設定, 9-103
 SHARED_POOL_RESERVED_SIZE 初期化パラメータ
 ALTER SYSTEM で設定, 9-103
 SHARED_POOL_SIZE 初期化パラメータ
 ALTER SYSTEM で設定, 9-103
 SHARED_SERVER_SESSIONS 初期化パラメータ
 ALTER SYSTEM で設定, 9-105
 SHARED_SERVERS 初期化パラメータ
 ALTER SYSTEM で設定, 9-104
 SHARED 句
 CREATE DATABASE LINK, 12-35
 SHRINK 句
 ALTER ROLLBACK SEGMENT, 8-117
 SHUTDOWN イベント
 トリガー, 14-84
 SHUTDOWN 句
 ALTER SYSTEM, 9-29
 SIGN ファンクション, 6-133
 SINGLE TABLE 句
 CREATE CLUSTER, 12-7
 SINH ファンクション, 6-134
 SIN ファンクション, 6-133
 SIZE 句
 ALTER CLUSTER, 8-5
 CREATE CLUSTER, 12-6
 filespec 句, 16-29
 SKIP_UNUSABLE_INDEXES セッション・パラメータ, 9-13
 SMALLINT データ型
 ANSI, 2-34
 DB2, 2-35
 SQL/DS, 2-35
 SNMPAGENT ロール, 16-48
 SOME 演算子, 5-5
 SORT_AREA_RETAINED_SIZE 初期化パラメータ
 ALTER SESSION で設定, 9-9
 ALTER SYSTEM で設定, 9-105
 SORT_AREA_SIZE 初期化パラメータ
 ALTER SESSION で設定, 9-9
 ALTER SYSTEM で設定, 9-106
 SOUNDEX ファンクション, 6-134
 SPECIFICATION 句
 ALTER PACKAGE, 8-100
 SPFILE 初期化パラメータ
 ALTER SYSTEM で設定, 9-106

SPLIT PARTITION 句
 ALTER INDEX, 8-53, 8-66
 ALTER TABLE, 10-49
 SPTH 日時書式要素の接尾辞, 2-72
 SP 日時書式要素の接尾辞, 2-72
 SQL*Loader INSERT、LOGGING, 8-56
 SQL:99 規格, 1-2
 SQL_TRACE 初期化パラメータ
 ALTER SYSTEM で設定, 9-107
 SQL_TRACE セッション・パラメータ, 9-13
 SQL92_SECURITY 初期化パラメータ
 ALTER SYSTEM で設定, 9-107
 SQLData Java 記憶形式, 15-11
 SQL/DS データ型, 2-34
 Oracle データ型への変換, 2-35
 暗黙的な変換, 2-35
 制限, 2-35
 sqlj_object_type_attr 句
 CREATE TYPE, 15-12
 SQLJ オブジェクト型
 作成, 15-10
 SQL ファンクション
 ABS, 6-14
 ACOS, 6-14
 ADD_MONTHS, 6-15
 ASCII, 6-16
 ASCIISTR, 6-17
 ASIN, 6-17
 ATAN, 6-18
 ATAN2, 6-19
 AVG, 6-19
 BFILENAME, 6-21
 BIN_TO_NUM, 6-22
 BITAND, 6-23
 CAST, 6-24
 CEIL, 6-27
 CHARTOROWID, 6-27
 CHR, 6-28
 COALESCE, 6-29
 COMPOSE, 6-31
 CONCAT, 6-31
 CONVERT, 6-32
 CORR, 6-34
 COS, 6-36
 COSH, 6-36
 COUNT, 6-37
 COVAR_POP, 6-39
 COVAR_SAMP, 6-41
 CUME_DIST, 6-43
 CURRENT_DATE, 6-44
 CURRENT_TIMESTAMP, 6-45
 DBTIMEZONE, 6-46
 DECOMPOSE, 6-48
 DENSE_RANK, 6-49
 DEREF, 6-51
 DUMP, 6-52
 EMPTY_BLOB, 6-54
 EMPTY_CLOB, 6-54
 EXISTSNODE, 6-54
 EXP, 6-55
 EXTRACT (XML), 6-57
 EXTRACT (日時), 6-56
 FIRST, 6-58
 FIRST_VALUE, 6-60
 FLOOR, 6-62
 FROM_TZ, 6-62
 GREATEST, 6-63
 GROUP_ID, 6-64
 GROUPING, 6-65
 GROUPING_ID, 6-66
 HEXTORAW, 6-68
 INITCAP, 6-69
 INSTR, 6-70
 INSTR2, 6-70
 INSTR4, 6-70
 INSTRB, 6-70
 INSTRC, 6-70
 LAG, 6-71
 LAST, 6-73
 LAST_DAY, 6-75
 LAST_VALUE, 6-76
 LEAD, 6-78
 LEAST, 6-79
 LENGTH, 6-80
 LENGTH2, 6-80
 LENGTH4, 6-80
 LENGTHB, 6-80
 LENGTHC, 6-80
 LN, 6-81
 LOB 列への適用, 6-2
 LOCALTIMESTAMP, 6-81
 LOG, 6-82
 LOWER, 6-83
 LPAD, 6-83

LTRIM, 6-84
 MAKE_REF, 6-85
 MAX, 6-86
 MIN, 6-88
 MOD, 6-89
 MONTHS_BETWEEN, 6-90
 NCHR, 6-91
 NEW_TIME, 6-92
 NEXT_DAY, 6-93
 NLS_CHARSET_DECL_LEN, 6-93
 NLS_CHARSET_ID, 6-94
 NLS_CHARSET_NAME, 6-95
 NLS_INITCAP, 6-95
 NLS_LOWER, 6-97
 NLS_UPPER, 6-99
 NLSSORT, 6-97
 NLV2, 6-105
 NTILE, 6-99
 NULLIF, 6-101
 NUMTODSINTERVAL, 6-102
 NUMTOYMINTERVAL, 6-103
 NVL, 6-104
 PERCENT_RANK, 6-106
 PERCENTILE_CONT, 6-108
 PERCENTILE_DISC, 6-110
 POWER, 6-111
 RANK, 6-112
 RATIO_TO_REPORT, 6-114
 RAWTOHEX, 6-115
 RAWTONHEX, 6-116
 REF, 6-116
 REFTOHEX, 6-117
 REGR_AVGX, 6-118
 REGR_AVGY, 6-118
 REGR_COUNT, 6-118
 REGR_INTERCEPT, 6-118
 REGR_R2, 6-118
 REGR_SLOPE, 6-118
 REGR_SXX, 6-118
 REGR_SXY, 6-118
 REGR_SYY, 6-118
 REPLACE, 6-126
 ROUND (数值), 6-127
 ROUND (日付), 6-128
 ROW_NUMBER, 6-128
 ROWIDTOCHAR, 6-130
 ROWIDTONCHAR, 6-130

RPAD, 6-131
 RTRIM, 6-131
 SESSIONTIMEZONE, 6-132
 SIGN, 6-133
 SIN, 6-133
 SINH, 6-134
 SOUNDEX, 6-134
 SQRT, 6-136
 STDDEV, 6-136
 STDDEV_POP, 6-138
 STDDEV_SAMP, 6-139
 SUBSTR, 6-141
 SUBSTR2, 6-141
 SUBSTR4, 6-141
 SUBSTRB, 6-141
 SUBSTRC, 6-141
 SUM, 6-142
 SYS_CONNECT_BY_PATH, 6-144
 SYS_CONTEXT, 6-145
 SYS_DBURIGEN, 6-150
 SYS_EXTRACT_UTC, 6-151
 SYS_GUID, 6-152
 SYS_TYPEID, 6-153
 SYS_XMLAGG, 6-154
 SYS_XMLGEN, 6-155
 SYSDATE, 6-156
 SYSTIMESTAMP, 6-157
 TAN, 6-158
 TANH, 6-158
 TO_CHAR (数值), 6-162
 TO_CHAR (日時), 6-160
 TO_CHAR (文字), 6-159
 TO_CLOB, 6-163
 TO_DATE, 6-164
 TO_DSINTERVAL, 6-165
 TO_LOB, 6-166
 TO_MULTI_BYTE, 6-167
 TO_NCHAR (数值), 6-169
 TO_NCHAR (日時), 6-169
 TO_NCHAR (文字), 6-168
 TO_NCLOB, 6-170
 TO_NUMBER, 6-171
 TO_SINGLE_BYTE, 6-172
 TO_TIMESTAMP, 6-172
 TO_YMINTERVAL, 6-174
 TRANSLATE, 6-175
 TRANSLATE...USING, 6-176

- TREAT, 6-178
- TRIM, 6-179
- TRUNC (数値), 6-180
- TRUNC (日付), 6-181
- TZ_OFFSET, 6-182
- UID, 6-183
- UNISTR, 6-183
- UPPER, 6-184
- USER, 6-185
- USERENV, 6-185
- VALUE, 6-187
- VAR_POP, 6-188
- VAR_SAMP, 6-189
- VARIANCE, 6-191
- VSIZE, 6-192
- WIDTH_BUCKET, 6-193
- オブジェクト参照, 6-13
 - 集計, 6-6
 - 数値, 6-3
 - 線形リグレッション, 6-118
 - 単一行, 6-3
 - その他, 6-6
 - 日付, 6-5
 - 分析, 6-8
 - 変換, 6-5
 - 文字
 - 数値, 6-4
 - 数値を戻す, 6-4
 - 文字値, 6-4
- SQL 文
 - DDL, 7-16
 - DML, 7-18
 - 監査
 - アクセス, 11-55
 - 正常終了, 11-55
 - セッション, 11-55
 - 停止, 16-80
 - プロキシ, 11-54
 - ユーザー, 11-54
 - 再開可能な領域割当て, 9-6
 - システム制御, 7-19
 - 実行計画の判断, 16-23
 - セッション制御, 7-19
 - セッションでの発生, 11-50
 - 停止およびコンパイル, 9-6
 - トランザクション制御, 7-18
 - 取消し, 16-96
 - ロールバック, 16-96
 - SQRT ファンクション, 6-136
 - SSSS 日時書式要素, 2-67
 - SS 日時書式要素, 2-67
 - STANDBY_ARCHIVE_DEST 初期化パラメータ
 - ALTER SYSTEM で設定, 9-107
 - STANDBY_FILE_MANAGEMENT 初期化パラメータ
 - ALTER SYSTEM で設定, 9-108
 - STAR_TRANSFORMATION_ENABLED 初期化パラメータ
 - ALTER SESSION で設定, 9-9
 - STAR_TRANSFORMATION ヒント, 2-100
 - START WITH 句
 - ALTER MATERIALIZED VIEW...REFRESH, 8-85
 - SELECT および副問合せ, 17-8
 - 問合せおよび副問合せ, 17-18
 - START WITH パラメータ
 - CREATE SEQUENCE, 13-83
 - startup_clauses
 - ALTER DATABASE, 8-10
 - STARTUP イベント
 - トリガー, 14-84
 - STATIC 句
 - ALTER TYPE, 11-11
 - CREATE TYPE, 15-12
 - CREATE TYPE BODY, 15-26
 - STDDEV_POP ファンクション, 6-138
 - STDDEV_SAMP ファンクション, 6-139
 - STDDEV ファンクション, 6-136
 - STORAGE IN ROW 句
 - ALTER TABLE, 10-66
 - STORAGE 句, 17-50
 - ALTER CLUSTER, 8-5
 - ALTER INDEX, 8-50, 8-56
 - ALTER MATERIALIZED VIEW, 8-79
 - ALTER MATERIALIZED VIEW LOG, 8-92
 - ALTER ROLLBACK SEGMENT, 8-116, 8-117
 - CREATE CLUSTER, 12-6
 - CREATE MATERIALIZED VIEW
 - 「CREATE TABLE」を参照
 - CREATE MATERIALIZED VIEW LOG, 13-32
 - 「CREATE TABLE」を参照
 - CREATE ROLLBACK SEGMENT, 13-78
 - CREATE TABLE, 14-12, 14-25
 - CREATE TABLESPACE, 14-64
 - STORE IN DEFAULT 句
 - CREATE INDEX, 12-75

STORE IN 句
 ALTER TABLE, 10-36, 14-39
STORE IN 表領域句
 CREATE INDEX, 12-75
Structured Query Language (SQL)
 構文, 8-1
 文
 監査, 11-56
 コストの判断, 16-23
 Oracle のツール製品のサポート, 1-5
 埋込み, 1-4
 キーワード, A-3
 規格, 1-2, B-1
 構文, A-1
 説明, 1-3
 パラメータ, A-3
 ファンクション, 6-2
SUBPARTITION BY HASH 句
 CREATE TABLE, 14-15, 14-40
SUBPARTITION 句
 ANALYZE, 11-36
 CREATE INDEX, 12-76
 CREATE TABLE, 14-41
 DELETE, 15-52
 INSERT, 16-61
 LOCK TABLE, 16-73
 UPDATE, 17-68
SUBSTR2 ファンクション, 6-141
SUBSTR4 ファンクション, 6-141
SUBSTRB ファンクション, 6-141
SUBSTRC ファンクション, 6-141
SUBSTR ファンクション, 6-141
SUM ファンクション, 6-142
SUSPEND 句
 ALTER SYSTEM, 9-27
SWITCH LOGFILE 句
 ALTER SYSTEM, 9-27
SYEAR 日時書式要素, 2-67
SYS_CONNECT_BY_PATH ファンクション, 6-144
SYS_CONTEXT ファンクション, 6-145
SYS_DBURIGEN ファンクション, 6-150
SYS_EXTRACT_UTC ファンクション, 6-151
SYS_GUID ファンクション, 6-152
SYS_TYPEID ファンクション, 6-153
SYS_XMLAGG ファンクション, 6-154
SYS_XMLGEN ファンクション, 6-155
SYSDATE ファンクション, 6-156

SYSDBA システム権限, 16-47
SYSOPER システム権限, 16-47
SYSTIMESTAMP ファンクション, 6-157
SYS スキーマ
 格納されたデータベース・トリガー, 14-87
 格納されたファンクション, 14-87
YYYYY 日時書式要素, 2-67
S 数値書式要素, 2-62

T

table_index_clause
 CREATE INDEX, 12-60
TABLESPACE 句
 ALTER INDEX ... REBUILD, 8-59
 CREATE CLUSTER, 12-6
 CREATE INDEX, 12-70
 CREATE MATERIALIZED VIEW, 13-14
 CREATE MATERIALIZED VIEW LOG, 13-32
 CREATE ROLLBACK SEGMENT, 13-77
 CREATE TABLE, 14-25
TABLE 句
 ANALYZE, 11-35
 DELETE, 15-54
 INSERT, 16-62
 SELECT, 17-15
 TRUNCATE, 17-60
 UPDATE, 17-67, 17-68, 17-70
TANH ファンクション, 6-158
TAN ファンクション, 6-158
TAPE_ASYNC_IO 初期化パラメータ
 ALTER SYSTEM で設定, 9-108
temp_tablespace_extent_clause
 CREATE DATABASE, 12-23
 CREATE TEMPORARY TABLESPACE, 14-74
TEMPFILE 句
 ALTER DATABASE, 8-13, 8-28
 CREATE TEMPORARY TABLESPACE, 14-74
TEMPORARY TABLESPACE 句
 ALTER USER
 「CREATE USER」を参照
 CREATE USER, 15-33
TEMPORARY 句
 ALTER TABLESPACE, 10-90
 CREATE TABLESPACE, 14-68
TEST 句
 ALTER DATABASE RECOVER, 8-22

THREAD 初期化パラメータ
 ALTER SYSTEM で設定, 9-109
THSP 日時書式要素の接尾辞, 2-72
TH 日時書式要素の接尾辞, 2-72
TIME_ZONE セッション・パラメータ, 9-13
TIMED_OS_STATISTICS 初期化パラメータ
 ALTER SESSION で設定, 9-9
 ALTER SYSTEM で設定, 9-109
TIMED_STATISTICS 初期化パラメータ
 ALTER SESSION で設定, 9-9
 ALTER SYSTEM で設定, 9-110
TIMESTAMP WITH LOCAL TIME ZONE データ型,
 2-22
TIMESTAMP WITH TIME ZONE データ型, 2-21
TIMESTAMP データ型, 2-20
 DB2, 2-35
 SQL/DS, 2-35
TIME データ型
 DB2, 2-35
 SQL/DS, 2-35
TM 数値書式要素, 2-62
TO SAVEPOINT 句
 ROLLBACK, 16-97
TO_CHAR
 数値変換ファンクション, 6-162
 日時変換ファンクション, 6-160
TO_CHAR ファンクション, 2-61, 2-66, 2-73
TO_CHAR (文字変換) ファンクション, 6-159
TO_CLOB ファンクション, 6-163
TO_DATE ファンクション, 2-66, 2-71, 2-73, 6-164
TO_DSINTERVAL ファンクション, 6-165
TO_LOB ファンクション, 6-166
TO_MULTI_BYTE ファンクション, 6-167
TO_NCHAR (数値) ファンクション, 6-169
TO_NCHAR (日時) ファンクション, 6-169
TO_NCHAR (文字) ファンクション, 6-168
TO_NCLOB ファンクション, 6-170
TO_NUMBER ファンクション, 2-61, 6-171
TO_SINGLE_BYTE ファンクション, 6-172
TO_TIMESTAMP_TZ ファンクション
 SQL ファンクション
 TO_TIMESTAMP_TZ, 6-173
TO_TIMESTAMP ファンクション, 6-172
TO_YMINTERVAL ファンクション, 6-174
TRACE_ENABLED 初期化パラメータ
 ALTER SYSTEM で設定, 9-110

TRACEFILE_IDENTIFIER 初期化パラメータ
 ALTER SESSION で設定, 9-9
 ALTER SYSTEM で設定, 9-110
TRANSACTION_AUDITING 初期化パラメータ
 ALTER SYSTEM で設定, 9-111
TRANSACTIONS_PER_ROLLBACK_SEGMENT 初期
 化パラメータ
 ALTER SYSTEM で設定, 9-111
TRANSLATE ...USING ファンクション, 6-176
TRANSLATE ファンクション, 6-175
TREAT ファンクション, 6-178
TRIM ファンクション, 6-179
TRUNCATE PARTITION 句
 ALTER TABLE, 10-48
TRUNCATE SUBPARTITION 句
 ALTER TABLE, 10-48
TRUNCATE 文, 17-59
TRUNC ファンクション
 書式モデル, 6-195
 数値ファンクション, 6-180
 日付ファンクション, 6-181
TRUST 属性
 PRAGMA RESTRICT_REFERENCES, 15-15
TYPES 句
 ASSOCIATE STATISTICS, 11-47, 11-48
TZ_OFFSET ファンクション, 6-182
TZD 日時書式要素, 2-67
TZH 日時書式要素, 2-67
TZM 日時書式要素, 2-67
TZR 日時書式要素, 2-67

U

UID ファンクション, 6-183
UNDER ANY TABLE システム権限, 16-44
UNDER ANY VIEW システム権限, 16-45
UNDER オブジェクト権限, 16-49
 型, 16-52
 ビュー, 16-50
UNDER 句
 CREATE VIEW, 15-43
UNDO_MANAGEMENT 初期化パラメータ
 ALTER SYSTEM で設定, 9-111
UNDO_RETENTION 初期化パラメータ
 ALTER SYSTEM で設定, 9-112
UNDO_SUPPRESS_ERRORS 初期化パラメータ
 ALTER SESSION で設定, 9-9

- ALTER SYSTEM で設定, 9-112
- UNDO_TABLESPACE 初期化パラメータ
 - ALTER SYSTEM で設定, 9-113
- UNDO のロールバック, 8-115, 12-29
- UNDO 表領域
 - 削除, 16-10
 - 作成, 12-29, 14-65
 - 変更, 10-85
- UNDO 表領域句
 - CREATE DATABASE, 12-29
 - CREATE TABLESPACE, 14-65
- UNIFORM 句
 - CREATE TABLESPACE, 14-69
- UNION ALL 集合演算子, 3-5, 17-21
- UNION 集合演算子, 3-5, 17-21
- UNIQUE 句
 - constraint_clause, 11-79
 - CREATE INDEX, 12-65
 - CREATE TABLE, 14-21
 - SELECT, 17-11
- UNISTR ファンクション, 6-183
- UNLIMITED TABLESPACE システム権限, 16-43
- UNQUIESCE 句
 - ALTER SYSTEM, 9-28
- UNRECOVERABLE, 8-57, 14-26
 - 「NOLOGGING 句」を参照
- UNUSABLE LOCAL INDEXES 句
 - ALTER MATERIALIZED VIEW, 8-81
 - ALTER TABLE, 10-40
- UNUSABLE 句
 - ALTER INDEX, 8-62
- UPDATE ANY TABLE システム権限, 16-43
- UPDATE BLOCK REFERENCES 句
 - ALTER INDEX, 8-64, 8-65
 - ALTER TABLE, 10-37
- UPDATE GLOBAL INDEXES 句
 - ALTER TABLE, 10-41
- UPDATE SET 句
 - MERGE, 16-76
- UPDATE オブジェクト権限, 16-49
 - ビュー, 16-50
 - 表, 16-50
- UPDATE 文, 17-64
 - トリガー, 14-82
- UPGRADE 句
 - ALTER TABLE, 10-34
- UPPER ファンクション, 6-184

- URL
 - 生成, 6-150
- UROWID
 - 外部キー表, 2-33
 - 索引構成表, 2-33
 - 説明, 2-33
- UROWID データ型, 2-33
- USE_STORED_OUTLINES 初期化パラメータ
 - ALTER SESSION で設定, 9-113
- USE_CONCAT ヒント, 2-101
- USE_MERGE ヒント, 2-101
- USE_NL ヒント, 2-101
- USE_PRIVATE_OUTLINES セッション・パラメータ, 9-14
- USE_STORED_OUTLINES セッション・パラメータ, 9-14, 9-113
- USER_COL_COMMENTS データ・ディクショナリ・ビュー, 11-66
- USER_DUMP_DEST 初期化パラメータ
 - ALTER SYSTEM で設定, 9-113
- USER_TAB_COMMENTS データ・ディクショナリ・ビュー, 11-66
- USERENV ファンクション, 6-185
- USER ファンクション, 6-185
- USING BFILE 句
 - CREATE JAVA, 12-90
- USING BLOB 句
 - CREATE JAVA, 12-90
- USING CLOB 句
 - CREATE JAVA, 12-90
- USING INDEX 句
 - ALTER MATERIALIZED VIEW, 8-83
 - ALTER TABLE, 10-26
 - constraint_clause, 11-86
 - CREATE MATERIALIZED VIEW, 13-17
 - CREATE TABLE, 14-18, 14-46
- USING NO INDEX 句
 - CREATE MATERIALIZED VIEW, 13-17
- USING ROLLBACK SEGMENT 句
 - ALTER MATERIALIZED VIEW...REFRESH, 8-86
 - CREATE MATERIALIZED VIEW, 13-20
- USING 句
 - ALTER INDEXTYPE, 8-70
 - ASSOCIATE STATISTICS, 11-47, 11-48
 - CREATE DATABASE LINK, 12-37
 - CREATE INDEXTYPE, 12-85
 - CREATE OPERATOR, 13-40

UTC

日時値から抽出, 6-151

UTC オフセット

タイム・ゾーン地域に置換, 2-21

UTL_FILE_DIR 初期化パラメータ

ALTER SYSTEM で設定, 9-114

UTLCHN.SQL スクリプト, 11-42

UTLEXPT1.SQL スクリプト, 10-52

UTLXPLAN.SQL スクリプト, 16-23

U 数値書式要素, 2-62

V

VALIDATE REF UPDATE 句

ANALYZE, 11-40

VALIDATE STRUCTURE 句

ANALYZE, 11-40

VALIDATE 句

DROP TYPE, 16-16

VALUES LESS THAN 句

CREATE TABLE, 14-37

VALUES 句

CREATE INDEX, 11-88, 12-74, 14-48

INSERT, 16-64

VALUE ファンクション, 6-187

VAR_POP ファンクション, 6-188

VAR_SAMP ファンクション, 6-189

VARCHAR2 データ型, 2-11

NUMBER への変換, 2-61

VARCHAR データ型, 2-12

DB2, 2-35

SQL/DS, 2-35

VARGRAPHIC データ型

DB2, 2-35

SQL/DS, 2-35

VARIANCE ファンクション, 6-191

VARRAY, 2-37

アウトラインでの格納, 2-37

記憶特性, 10-65, 10-67, 14-34

作成, 15-3, 15-8, 15-17

仕様部の削除, 16-14

ネストした表との比較, 2-45

比較規則, 2-45

本体の削除, 16-17

戻り値の変更, 10-62

VARRAY 句

ALTER TABLE, 10-22

VARRAY 列プロパティ

ALTER TABLE, 10-22, 10-65

CREATE MATERIALIZED VIEW, 13-11

CREATE TABLE, 14-11, 14-34

VARYING 配列

「VARRAY」を参照

VSIZE ファンクション, 6-192

V 数値書式要素, 2-62

W

WHEN MATCHED 句

MERGE, 16-77

WHEN NOT MATCHED 句

MERGE, 16-77

WHENEVER NOT SUCCESSFUL 句

NOAUDIT, 16-83

WHENEVER SUCCESSFUL 句

AUDIT sql_statements, 11-55

NOAUDIT, 16-83

WHEN 句

CREATE TRIGGER, 14-86

WHERE 句

DELETE, 15-54

SELECT, 7-4

UPDATE, 17-71

WIDTH_BUCKET ファンクション, 6-193

WITH ADMIN OPTION 句

GRANT, 16-35

WITH CHECK OPTION 句

CREATE VIEW, 15-40, 15-45

DELETE, 15-52

INSERT, 16-63

SELECT, 17-7, 17-14

UPDATE, 17-68

WITH GRANT OPTION 句

GRANT, 16-36

WITH HIERARCHY OPTION

GRANT, 16-37

WITH INDEX CONTEXT 句

CREATE OPERATOR, 13-40

WITH OBJECT IDENTIFIER 句

CREATE VIEW, 15-42

WITH OBJECT ID 句

CREATE MATERIALIZED VIEW LOG, 13-34

WITH OBJECT OID

「WITH OBJECT IDENTIFIER」を参照

WITH PRIMARY KEY 句

ALTER MATERIALIZED VIEW, 8-85

CREATE MATERIALIZED VIEW LOG, 13-34

CREATE MATERIALIZED VIEW...REFRESH,
13-17

WITH query_name 句

SELECT, 17-10

WITH READ ONLY 句

CREATE VIEW, 15-40, 15-45

DELETE, 15-52

INSERT, 16-63

SELECT, 17-7, 17-14

UPDATE, 17-68

WITH ROWID 句

CREATE MATERIALIZED VIEW LOG, 13-34

CREATE MATERIALIZED VIEW...REFRESH,
13-17

REF 列制約, 11-84

WITH SEQUENCE 句

CREATE MATERIALIZED VIEW LOG, 13-34

WNDS 属性

PRAGMA RESTRICT_REFERENCES, 15-15

WNPS 属性

PRAGMA RESTRICT_REFERENCES, 15-15

WORKAREA_SIZE_POLICY 初期化パラメータ

ALTER SESSION で設定, 9-9

ALTER SYSTEM で設定, 9-114

WRITE オブジェクト権限

ディレクトリ, 16-51

WW 日時書式要素, 2-67

W 日時書式要素, 2-67

X

XMLGenFormatType オブジェクト, 2-76

XMLType 記憶域句

CREATE TABLE, 14-36

XML 書式モデル, 2-76

XML データ

記憶域, 14-36

XML ドキュメント

XML のフラグメントから生成, 6-154

データベースからの検索, 6-150

XML のフラグメント, 6-57

X 数値書式要素, 2-62

X 日時書式要素, 2-67

Y

Y,YYY 日時書式要素, 2-67

YEAR 日時書式要素, 2-67

YYYY 日時書式要素, 2-67

YYY 日時書式要素, 2-67

YY 日時書式要素, 2-67

Y 日時書式要素, 2-67

あ

アーカイブ REDO ログ

位置, 8-21

格納場所, 9-65

アーカイブ・モード

指定, 12-26

アーカイブ・ログ

スタンバイ・データベースへの適用, 8-24

アウトライン

PUBLIC による使用, 13-44

置換え, 13-43

オブティマイザによる使用, 9-113

オブティマイザによるプライベート・アウトライン
の使用, 9-14

グループの格納, 13-44

現行のセッションによる使用, 13-44

異なるカテゴリへの割当て, 8-97, 8-98, 8-99

コピー, 13-44

再構築, 8-97, 8-99

再コンパイル, 8-98

作成, 13-42

実行計画の生成での使用, 13-42, 9-14

自動作成および自動格納, 9-42

セッション中のリストア, 9-10

データベースからの削除, 15-84

動的な使用可能化または使用禁止化, 13-43

名前の変更, 8-97, 8-98, 8-99

付与

システム権限, 16-41

文に作成, 13-44

空きリスト

表、パーティション、クラスタまたは索引の指定,
17-54

アプリケーション

検証, 12-11

保護, 12-11

ユーザーとしての接続の許可, 11-24

アプリケーション・サーバー

ユーザーとしての接続の許可, 11-24

暗黙的なデータ変換, 2-46, 2-47, 2-49

い

移行行

クラスタ, 11-36

リスト, 11-42

以上 / 以下のテスト, 5-5

一意索引, 12-65

一意制約

索引, 14-46

使用可能, 14-46

一意の問合せ, 17-11

一時表

作成, 14-6, 14-19

セッション固有, 14-19

トランザクション固有, 14-19

一時表領域

SQL 例, 14-76

作成, 14-73

デフォルト, 8-37

ユーザーへの指定, 15-33

位置の透過性, 14-2

一般リカバリ句

ALTER DATABASE, 8-11, 8-20

インスタンス

グローバル・ネーム解決, 9-56

索引エクステンツの使用可能化, 8-55

パラメータの設定, 9-30

メモリー要件, 9-43

インスタンス・リカバリ

中断後の継続, 8-20

インダウト・トランザクション

強制コミット, 11-70

強制実行, 11-70

ロールバック, 16-96

ロールバックの強制, 16-98

引用符

データベース・オブジェクト名での使用, 2-108

インライン・ビュー, 7-12

う

埋込み SQL, 1-4, 7-19

プリコンパイラのサポート, 7-19

え

英大文字または英小文字の区別

スキーマ・オブジェクト名, 2-108

エクステンツ

インスタンスを使用したアクセス制限, 8-55

オブジェクト作成時に割り当てられるエクステンツ数の指定, 17-53

オブジェクトの最大エクステンツ数の指定, 17-54

オブジェクトの第1エクステンツの指定, 17-52

オブジェクトの第2エクステンツの指定, 17-52

サイズ増加の割合の指定, 17-53

サブパーティションへの割当て, 10-31

パーティションへの割当て, 10-31

表への割当て, 10-31

エラー・メッセージ

言語の設定, 9-8

演算子, 3-1

コメント, 11-68

索引タイプからの削除, 8-70

索引タイプへの追加, 8-70

算術, 3-3

集合, 3-5, 17-21

単項, 3-2

バイナリ, 3-2

付与

システム権限, 16-40

ユーザー定義, 3-6

削除, 15-82

作成, 13-38

実装タイプ, 13-40

実装を設定するファンクション, 13-40

バインド実装方法, 13-40

バインドの戻り型, 13-40

ファンクションへのバインド, 13-40

優先順位, 3-2

連結, 3-4

お

応答時間

最適化, 2-90

オブジェクト

「オブジェクト型」または「データベース・オブジェクト」を参照

オブジェクト・アクセス式, 4-12

- オブジェクト・インスタンス
 - タイプ, 5-17
- オブジェクト型, 2-36
 - 「REF」を参照
 - SQL 例, 15-18
 - VARRAY, 15-8
 - 値
 - 比較, 15-28
 - 新しいメンバーのサブプログラムの追加, 11-10
 - 依存する型の処理, 11-16
 - 依存する型の無効化, 11-16
 - 継承, 15-13
 - コンポーネント, 2-36
 - 作成, 15-3, 15-5
 - サブタイプ, 11-10
 - サブタイプの権限, 16-37
 - サブタイプの指定, 15-10
 - 参照の再構築の展開, 8-83
 - システム権限の付与, 16-44
 - 使用可能なオブジェクト・インスタンス, 15-13
 - 使用可能なサブタイプ, 15-13
 - 仕様部および本体のコンパイル, 11-9
 - 仕様部の削除, 16-14
 - スーパータイプ, 11-10
 - ステティック・メソッド, 15-12
 - 属性, 2-116
 - 階層のメンバーシップ, 10-64
 - 型の階層, 14-31
 - 置換可能, 10-64
 - 統計タイプ, 11-46
 - 統計タイプの削除, 16-15
 - トップレベル, 15-10
 - ネストした表, 15-8
 - 比較規則, 2-45
 - MAP ファンクション, 2-45
 - ORDER ファンクション, 2-45
 - ファンクション・サブプログラム, 11-11, 15-12, 15-26
 - 宣言, 15-28
 - 不完全, 15-3, 15-4
 - プロシージャ・サブプログラム, 11-11, 15-12, 15-26
 - 宣言, 15-28
 - 本体
 - SQL 例, 15-29
 - 再作成, 15-26
 - 作成, 15-24
 - 本体の削除, 16-17
 - メソッド, 2-116
 - メソッドの削除, 11-14
 - メソッドの追加, 11-14
 - メンバー・メソッドの定義, 15-24
 - ユーザー定義
 - 作成, 15-9
 - ルートの指定, 15-10
 - オブジェクト型マテリアライズド・ビュー
 - 作成, 13-12
 - オブジェクト型列
 - 階層のメンバーシップ, 10-64
 - 型の階層, 14-31
 - 置換可能, 10-64
 - オブジェクト・キャッシュ, 9-8, 9-80
 - オブジェクト権限
 - ON COMMIT REFRESH, 16-49
 - QUERY REWRITE, 16-49
 - UNDER, 16-49
 - データベース・オブジェクト
 - 取消し, 16-92
 - 取消し
 - PUBLIC, 16-91
 - ユーザー, 16-87, 16-90
 - ロール, 16-87, 16-91
 - 付与, 13-72
 - 特定の列, 16-36
 - 複数, 13-79
 - オブジェクト参照ファンクション, 6-13
 - オブジェクト識別子
 - REF に含まれる, 2-37
 - オブジェクト・ビュー, 15-42
 - 索引の指定, 14-23
 - システム生成, 14-22
 - 指定, 14-22
 - 主キー, 14-22
 - オブジェクト・ビュー
 - 作成, 15-42
 - サブビューの作成, 15-43
 - 実表
 - 行の追加, 16-56
 - 定義, 15-37
 - オブジェクト表
 - アップグレード, 10-34
 - 階層の一部, 14-20
 - 行の追加, 16-56
 - 最新バージョンに更新, 10-34

- 作成, 14-8, 14-20
- オペランド, 3-1
- オペレーティング・システム・ファイル
 - 削除, 8-28, 16-11
- オンライン REDO ログ
 - 初期化しなおし, 8-32
- オンライン索引, 12-72
 - 再構築, 10-69
- オンライン・バックアップ
 - 表領域のオンライン・バックアップの終了, 10-89

か

- カーソル
 - キャッシュ, 16-23
 - 共有ブール, 9-103
- 階層
 - ディメンションからの削除, 8-45
 - ディメンションの追加, 8-45
 - ディメンションの定義, 12-41
- 階層問合せ, 2-82, 7-3, 17-18
 - 親である行, 2-82, 7-4
 - 子である行, 2-82, 7-4
 - 説明, 2-82
 - リーフ行, 2-82
 - ルートおよびノードの値の検出, 6-144
- 階層問合せ句
 - SELECT および副問合せ, 17-8
- 外部 LOB, 2-25
- 外部キー制約, 11-82
- 外部キー表
 - ROWID, 2-33
- 外部結合, 7-11
 - 制限事項, 7-11
- 外部表, 14-27
 - 作成, 14-29
 - 制限, 14-30
 - 変更, 10-68
- 外部ファンクション, 12-47, 13-58
- 外部プロシージャ, 13-56, 13-58
 - リモート・データベースからの実行, 13-3
- 外部ユーザー, 13-73, 15-32
- 科学表記法, 2-63
- 拡張 ROWID, 2-32
 - 基本, 2-32
 - 直接利用できない, 2-32
- 拡張サブパーティション表名, 2-104

- DML 文, 2-104
- 構文, 2-105
- 制限, 2-105
- 拡張パーティション表名, 2-104
 - DML 文, 2-104
 - 構文, 2-105
 - 制限, 2-105

数

- SQL 構文内, 2-53
- 構文, 2-53
- 精度, 2-54
- 比較規則, 2-42
- 浮動小数点, 2-12, 2-14
- フルスペルでの表記, 2-72
- 丸め, 2-13

型

- 「オブジェクト型」または「データ型」を参照
- 型コンストラクタ式, 4-14
- 型メソッド
 - 戻り型, 15-14
- 各国語キャラクタ・セット
 - 可変長文字列, 2-11
 - 固定幅と可変幅, 2-11
 - 変更, 8-35
 - マルチバイト・キャラクタ・セット, 2-11
 - マルチバイト・キャラクタ・データ, 2-31

監査

- SQL 文, 11-56
- SQL 文の停止, 16-80
- オプション
 - SQL 文, 11-58
 - データベース・オブジェクト, 11-56
- 方針
 - 値に基づいた方針, 11-50
- 完全な外部結合, 17-16
- 管理スタンバイ・リカバリ
 - 終了, 8-25
 - 制御の復帰, 8-25
 - バックグラウンド・プロセス, 8-25
 - 遅延のオーバーライド, 8-25
- 管理リカバリ
 - データベース, 8-12
 - 待ち時間, 8-24

き

- キー圧縮, 14-28

索引
 使用禁止, 8-60
索引構成表, 14-28
索引再構築, 10-69
使用可能, 8-57
使用禁止, 8-60, 12-70
定義, 8-60
キー保存表, 15-44
キーワード, 2-107
 オプション, A-4
 必須, A-3
記憶域パラメータ
 デフォルトの変更, 10-87
 リセット, 17-59
期間式, 4-12
期間データ型, 2-17
疑似列, 2-79
 CURRVAL, 2-79
 LEVEL, 2-82
 NEXTVAL, 2-79
 ROWID, 2-82
 ROWNUM, 2-83
 使用方法, 2-84
機能
 新規, xxiii
逆索引, 12-71
逆分散関数, 6-108, 6-110
キャッシュされたカーソル
 実行計画, 16-23
キャラクタ・セット
 共通, 2-44
 データベースへの指定, 12-27
 変更, 8-35
 マルチバイト・キャラクタ, 2-106
キャラクタ・ラージ・オブジェクト
「CLOB」を参照
行
 違反制約の格納, 10-52
 階層順序の選択, 7-3
 削除
 クラスタ, 17-59
 パーティションおよびサブパーティション,
 15-52
 表, 17-59
 表およびビュー, 15-49
 昇順で格納, 11-88
 制約の指定, 11-82

挿入
 サブパーティション, 16-61
 パーティション, 16-61
 リモート・データベース, 16-61
 パーティション間の移動, 14-12, 14-41
 表への追加, 16-56
共有サーバー
 システム・パラメータ, 9-104
 パラメータ
 DISPATCHERS, 9-51
プロセス
 終了, 9-104
 追加プロセスの作成, 9-104
共有プール
 フラッシュ, 9-27
行レベル依存の追跡, 12-8, 14-42

<

空白埋め
 書式モデルへの指定, 2-73
 回避, 2-73
空白埋め比較セマンティクス, 2-44
クエリー・リライト
 使用可能, 9-94
 使用可能および使用禁止, 9-9
 使用禁止, 9-94
 定義済, 17-4
 ディメンション, 12-39
 ファンクション索引, 9-9
 ルールベースの最適化, 9-9
グローバリゼーション・サポート
 セッション設定の変更, 9-8
位取り
 NUMBER データ型, 2-12
 精度より大きい, 2-14
 負, 2-13
クラスタ
 SQL 例, 15-61
 移行行および連鎖行, 11-36, 11-42
 エクステンツの割当て, 8-4, 8-5
キー値
 領域の変更, 8-5
 領域の割当て, 12-6
記憶域属性
 指定, 12-6, 17-50
 変更, 8-4

- 記憶特性の変更, 8-5
- クラスタ索引, 12-66
- 構造の検証, 11-40
- 索引, 12-6
- 作成, 12-2
- 作成表領域, 12-6
- システム権限の付与, 16-38
- データ・ブロックの割当て, 12-6
- データベースからの削除, 15-60
- 統計情報の収集, 11-36
- 取り出されたブロックのキャッシュ, 12-9
- ハッシュ, 12-6
 - 1 つの表, 12-7
- 表の削除, 15-61
- 表への割当て, 14-31
- 物理属性
 - 指定, 12-5
 - 変更, 8-4
- 並列度
 - 作成時, 12-8
 - 変更, 8-4, 8-7
- 変更, 8-3
- 未使用エクステンツの割当て解除, 8-4
- 未使用領域の解放, 8-6
- グルーピング・セット, 17-20
- グループ化
 - 重複の排除, 6-64
- グループ比較条件, 5-7
- グローバル索引
 - 「グローバル・パーティション索引」を参照
- グローバル・データベース名
 - 解決, 9-56
- グローバル・パーティション索引, 11-87, 12-73, 12-74, 14-47
- グローバル・ユーザー, 13-73, 15-32
- クローン・データベース
 - マウント, 8-18
- クロス結合, 17-16

け

- 結合, 7-9
 - 外部, 7-11
 - 制限事項, 7-11
 - 完全な外部結合, 17-16
 - クロス, 17-16
 - 結合条件のない, 7-11

- 自己, 7-10, 7-11
- 自然, 17-16
- 条件
 - 定義, 7-10
- 等価結合, 7-10
- 内部, 17-16
- ネステッド・ループの最適化, 9-9
- パラレルおよび PQ_DISTRIBUTE ヒント, 2-98
- 左側外部結合, 17-16
- 右側外部結合, 17-16
- 結合ビュー
 - 更新可能化, 15-44
 - 変更, 15-53, 16-63, 17-69
 - 例, 15-47
- 権限
 - 「システム権限」または「オブジェクト権限」を参照
 - オブジェクト型のサブタイプ, 16-37
- 検証
 - オンラインのデータベース・オブジェクト, 11-41
 - クラスタ, 11-40
 - 索引, 11-40
 - データベース・オブジェクト
 - オフライン, 11-41
 - 表, 11-40

こ

- 更新
 - 同時の挿入, 16-76
- 更新操作
 - 補助ログ・データの収集, 8-31
- 構文図, A-1
 - 複数の部分に分割された図, A-5
 - ループ, A-4
- コール
 - CPU 時間の制限, 13-68
 - 読み込まれたデータ・ブロックの制限, 13-68
- コール仕様
 - CREATE PROCEDURE, 13-63
 - CREATE TYPE, 15-14
 - CREATE TYPE BODY, 15-27
 - プロシージャ, 13-58
- コミット
 - 自動, 11-69
- コメント, 2-85
 - SQL 文, 2-85

- 演算子, 11-68
- オブジェクトからの削除, 11-66
- オブジェクトへの追加, 11-66
- 索引タイプ, 11-68
- 指定, 2-85
- スキーマ・オブジェクト, 2-86
- データ・ディクショナリからの削除, 11-66
- トランザクションとの関連, 11-70
- 表示, 11-66
- 固有の問合せ, 17-11
- コレクション
 - 行の挿入, 16-62
 - ネスト解除, 17-15
 - 例, 17-35
 - 表としての扱い, 16-62, 17-15, 17-67, 17-68
- コレクション型値
 - データ型への変換, 6-24
- コンストラクタ・メソッド
 - オブジェクト型, 15-3
- コンテキスト
 - システム権限の付与, 16-38
 - ネームスペースの作成, 12-11
- コンテキスト・ネームスペース
 - LDAP ディレクトリを使用する初期化, 12-12
 - OCI を使用する初期化, 12-12
 - インスタンスへのアクセス性, 12-13
 - データベースからの削除, 15-62
 - パッケージへの関連付け, 12-11
- コンパイラ・スイッチ
 - 削除および保持, 8-47, 8-101, 8-105, 11-4, 11-10
- コンボジット・パーティション化句
 - CREATE TABLE, 14-14, 14-40

さ

- サーバー・パラメータ・ファイル
 - 作成, 13-86
- サービス名
 - リモート・データベース, 12-37
- 再開可能な領域割当て, 9-6
- 最高水位標
 - クラスタ, 8-6
 - 索引, 8-54
 - 表, 10-32, 11-35
- 再構築, 8-83
- 索引, 8-56
 - B* ツリー, 12-58

- UNUSABLE のマーク付け, 8-62
- アクセス・パスの最適化, 9-9
- アプリケーション固有, 12-84
- 一意, 12-65
- 移動, 8-57
- オンライン, 12-72
- オンラインの場合の再構築, 8-60
- キー圧縮, 8-60
- キー圧縮の使用可能化, 8-57
- キーの反復の排除, 8-57
- 記憶域属性, 12-70, 17-50
- 逆, 8-57, 8-59, 12-71
- クラスタ, 12-66
- グローバル・パーティション, 11-87, 12-73, 12-74, 14-47
- 更新, 10-41
- 降順, 12-69
 - クエリー・リライト, 12-69
 - ファンクション索引, 12-69
- 構造の検証, 11-40
- コンボジット・パーティション表, 12-75
- 再構築, 8-57
- 再構築操作のログの記録, 8-57, 8-61
- 再構築の統計, 8-60
- 再作成, 8-57
- 索引構成表, 12-66
- 索引に基づく, 12-76
- 索引パーティションの削除, 15-72
- 索引ブロックの内容のマージ, 8-63
- 作成, 12-58
- 作成のパラレル化, 12-72
- サブパーティション
 - UNUSABLE のマーク付け, 8-66
 - 移動, 8-57
 - エクステンツの割当て, 8-66
 - 記憶特性の変更, 8-64
 - 再構築, 8-57
 - 再作成, 8-57
 - 使用禁止, 8-62
 - デフォルト属性の変更, 8-64
 - 名前の変更, 8-65
 - 表領域の指定, 8-57, 8-59
 - 物理属性の変更, 8-56
 - 変更, 8-57
 - 未使用領域の解放, 8-54, 8-66
- システム権限の付与, 16-39
- 使用禁止, 8-62

- 昇順, 12-69
- 使用の統計, 8-63
- 新規エクステンツの割当て, 8-55
- スカラー型オブジェクト属性, 12-66
- 制約の適用, 10-63, 14-46
- 属性の変更, 8-56, 8-57
- ダイレクト・パス INSERT、LOGGING, 8-56
- データベースからの削除, 15-71
- 統計情報, 12-72
- 統計情報の収集, 11-34
- 統計タイプの削除, 15-72
- ドメイン, 12-58, 12-76, 12-84
- 名前の変更, 8-57, 8-62
- ネストした表の記憶表, 12-66
- パーティション, 2-104, 12-58, 12-73
 - UNUSABLE のマーク付け, 8-65, 10-40
 - 記憶特性の変更, 8-64
 - 再構築, 8-57
 - 再作成, 8-57
 - 削除, 8-64, 8-65
 - 実特性の変更, 8-65
 - 使用禁止, 8-62
 - 使用禁止の再構築, 10-40
 - 新規の追加, 8-66
 - デフォルト属性の変更, 8-64
 - 名前の変更, 8-65
 - 表領域の指定, 8-57, 8-59
 - 物理属性の変更, 8-56
 - 分割, 8-64, 8-66
 - 未使用領域の解放, 8-54
 - ユーザー定義, 11-87, 12-73, 14-47
- パーティション化, 12-73
- パーティション表, 12-66
- ハッシュ・パーティション表, 12-75
- ビットマップ, 12-65
- ビットマップ結合, 12-77
- 表の列, 12-66
- 表領域, 12-70
- 表領域の指定, 8-57, 8-59
- ファンクション, 12-58
 - 作成, 12-67
- 物理属性, 12-70
- ブロック内容のマージ, 8-57
- 並列性の変更, 8-55
- 未使用領域の解放, 8-54
- 未ソート, 12-70
- 例, 12-78
- レンジ・パーティション表, 12-75
- ローカル・パーティション, 12-74
- 索引クラスタ
 - 作成, 12-6
- 索引構成表
 - PCT_ACCESS_DIRECT 統計情報, 11-35
 - ROWID, 2-33
 - 移動, 10-68
 - オーバーフロー・セグメント
 - 記憶域の指定, 10-36, 14-39
 - 再構築, 10-68
 - 作成, 14-6
 - 主キー索引
 - 合わせる, 10-37
 - 更新, 10-37
 - セカンダリ索引の更新, 8-64
 - パーティション化されたセカンダリ索引の更新, 8-65
 - ビットマップ索引の作成, 14-28
 - 変更, 10-35
 - マッピング表, 10-69
 - 移動, 10-44
 - マッピング表の作成, 14-28
- 索引タイプ
 - インスタンス, 12-58
 - コメント, 11-68
 - 索引タイプに基づく索引, 12-76
 - 削除ルーチンの起動, 15-72
 - 作成, 12-84
 - システム権限の付与, 16-39
 - 実装タイプの変更, 8-69
 - 操作の追加, 8-69
 - データベースからの削除, 15-73
 - 統計情報の関連付け, 11-48
 - 統計情報のタイプの関連性の削除, 15-73
 - 変更, 8-69
- サブタイプ, 11-10
 - 安全な削除, 16-16
- サブパーティション
 - 値の変更, 17-68
 - 合わせる, 10-40
 - エクステンツの割当て, 10-31, 10-42
 - 行の削除, 10-48, 15-52
 - 行の挿入, 16-61
 - 行の追加, 16-56
 - 作成, 14-15, 14-41
 - 指定, 14-40

- 挿入操作のロギング, 10-29
- 追加, 10-39
- 名前の変更, 10-42
- 非パーティション表への変換, 10-51
- 物理属性
 - 変更, 10-28
- 別のセグメントへの移動, 10-44
- 未使用領域の解放, 10-32, 10-42
- ロック, 16-72
- 算術
 - DATE 値, 2-19
 - 演算子, 3-3
- 参照整合性制約, 11-81, 11-82

し

- 式
 - CASE, 4-6
 - CURSOR, 4-8
 - DUAL 表の計算, 7-14
 - SQL 構文内, 4-2
 - オブジェクト・アクセス, 4-12
 - 型コンストラクタ, 4-14
 - 期間, 4-12
 - スカラー副問合せ, 4-13
 - 宣言型の変更, 6-178
 - 単純, 4-4
 - 日時, 4-10
 - 比較, 6-47
 - 複合, 4-5
 - 変数, 4-15
 - リスト, 4-16
- 自己結合, 7-10, 7-11
- システム・イベント
 - 属性, 14-87
 - トリガー, 14-84
- システム・グローバル領域
 - 更新, 9-25
 - フラッシュ, 9-27
- システム権限
 - ADMINISTER DATABASE TRIGGER, 16-44
 - ALTER ANY CLUSTER, 16-38
 - ALTER ANY DIMENSION, 16-38
 - ALTER ANY INDEX, 16-39
 - ALTER ANY INDEXTYPE, 16-39
 - ALTER ANY MATERIALIZED VIEW, 16-40
 - ALTER ANY OUTLINE, 16-41
 - ALTER ANY PROCEDURE, 16-41
 - ALTER ANY ROLE, 16-41
 - ALTER ANY SEQUENCE, 16-42
 - ALTER ANY TABLE, 16-43
 - ALTER ANY TRIGGER, 16-44
 - ALTER ANY TYPE, 16-44
 - ALTER DATABASE, 16-38
 - ALTER PROFILE, 16-41
 - ALTER RESOURCE COST, 16-42
 - ALTER ROLLBACK SEGMENT, 16-42
 - ALTER SESSION, 16-42
 - ALTER SYSTEM, 16-38
 - ALTER TABLESPACE, 16-43
 - ALTER USER, 16-45
 - ANALYZE ANY, 16-46
 - AUDIT ANY, 16-46
 - AUDIT SYSTEM, 16-38
 - BACKUP ANY TABLE, 16-43
 - BECOME USER, 16-45
 - COMMENT ANY TABLE, 16-46
 - CREATE ANY CLUSTER, 16-38
 - CREATE ANY CONTEXT, 16-38
 - CREATE ANY DIMENSION, 16-38
 - CREATE ANY DIRECTORY, 16-39
 - CREATE ANY INDEX, 16-39
 - CREATE ANY INDEXTYPE, 16-39
 - CREATE ANY LIBRARY, 16-39
 - CREATE ANY MATERIALIZED VIEW, 16-40
 - CREATE ANY OPERATOR, 16-40
 - CREATE ANY OUTLINE, 16-41
 - CREATE ANY PROCEDURE, 16-41
 - CREATE ANY SEQUENCE, 16-42
 - CREATE ANY SYNONYM, 16-42
 - CREATE ANY TABLE, 16-43
 - CREATE ANY TRIGGER, 16-44
 - CREATE ANY TYPE, 16-44
 - CREATE ANY VIEW, 16-45
 - CREATE CLUSTER, 16-38
 - CREATE DATABASE LINK, 16-38
 - CREATE DIMENSION, 16-38
 - CREATE INDEXTYPE, 16-39
 - CREATE LIBRARY, 16-39
 - CREATE MATERIALIZED VIEW, 16-40
 - CREATE OPERATOR, 16-40
 - CREATE PROCEDURE, 16-41
 - CREATE PROFILE, 16-41
 - CREATE PUBLIC DATABASE LINK, 16-38

CREATE PUBLIC SYNONYM, 16-42
 CREATE ROLE, 16-41
 CREATE ROLLBACK SEGMENT, 16-42
 CREATE SEQUENCE, 16-42
 CREATE SESSION, 16-42
 CREATE SYNONYM, 16-42
 CREATE TABLE, 16-43
 CREATE TABLESPACE, 16-43
 CREATE TRIGGER, 16-44
 CREATE TYPE, 16-44
 CREATE USER, 16-45
 CREATE VIEW, 16-45
 DELETE ANY TABLE, 16-43
 DROP ANY CLUSTER, 16-38
 DROP ANY CONTEXT, 16-38
 DROP ANY DIMENSION, 16-38
 DROP ANY DIRECTORY, 16-39
 DROP ANY INDEX, 16-39
 DROP ANY INDEXTYPE, 16-39
 DROP ANY LIBRARY, 16-39
 DROP ANY MATERIALIZED VIEW, 16-40
 DROP ANY OPERATOR, 16-40
 DROP ANY OUTLINE, 16-41
 DROP ANY PROCEDURE, 16-41
 DROP ANY ROLE, 16-41
 DROP ANY SEQUENCE, 16-42
 DROP ANY SYNONYM, 16-42
 DROP ANY TABLE, 16-43
 DROP ANY TRIGGER, 16-44
 DROP ANY TYPE, 16-44
 DROP ANY VIEW, 16-45
 DROP LIBRARY, 16-39
 DROP PROFILE, 16-41
 DROP PUBLIC DATABASE LINK, 16-38
 DROP PUBLIC SYNONYM, 16-42
 DROP ROLLBACK SEGMENT, 16-42
 DROP TABLESPACE, 16-43
 DROP USER, 16-45
 EXECUTE ANY INDEXTYPE, 16-39
 EXECUTE ANY OPERATOR, 16-40
 EXECUTE ANY PROCEDURE, 16-41
 EXECUTE ANY TYPE, 16-44
 EXEMPT ACCESS POLICY, 16-46
 FORCE ANY TRANSACTION, 16-46
 FORCE TRANSACTION, 16-46
 GLOBAL QUERY REWRITE, 16-39, 16-40
 GRANT ANY PRIVILEGE, 16-46
 GRANT ANY ROLE, 16-41
 INSERT ANY TABLE, 16-43
 LOCK ANY TABLE, 16-43
 MANAGE TABLESPACE, 16-43
 ON COMMIT REFRESH, 16-40
 QUERY REWRITE, 16-39, 16-40
 RESTRICTED SESSION, 16-42
 RESUMABLE, 16-46
 SELECT ANY DICTIONARY, 16-46
 SELECT ANY OUTLINE, 16-41
 SELECT ANY SEQUENCE, 16-42
 SELECT ANY TABLE, 16-43
 SYSDBA, 16-47
 SYSOPER, 16-47
 UNDER ANY TYPE, 16-44
 UNDER ANY VIEW, 16-45
 UNLIMITED TABLESPACE, 16-43
 UPDATE ANY TABLE, 16-43
 一覧, 16-38
 取消し, 16-87
 PUBLIC, 16-89
 ユーザー, 16-89
 ロール, 16-89
 付与, 13-72, 16-31
 PUBLIC, 16-34
 ユーザー, 16-34
 ロール, 16-34
 システム制御文, 7-19
 PL/SQL サポート, 7-19
 システムの日付
 変更, 9-55
 システム・リソース
 使用可能および使用禁止, 9-98
 自然結合, 17-16
 実行計画
 削除するアウトライン, 15-84
 判断, 16-23
 保存, 13-42
 実行時の再コンパイル
 回避, 8-103, 11-28
 実行者定義ファンクション
 定義, 12-52
 自動 UNDO 管理モード, 8-115, 12-29
 自動セグメント領域管理, 2-15, 14-25, 14-70
 シノニム
 作成, 14-2
 シノニム, 14-2

- 定義の変更, 16-4
- データベースからの削除, 16-4
- 名前の変更, 16-85
- パブリック, 14-3
 - 削除, 16-4
- 付与
 - システム権限, 16-42
- プライベート・シノニムの削除, 16-4
- リモート, 14-4
- ローカル, 14-4
- 集計ファンクション, 6-6
 - ユーザー定義型, 12-55
- 集合
 - VARRAY, 2-37
 - ネストした表, 2-38
 - 表としての扱い, 15-54
 - 変更, 10-62
- 集合演算子, 3-5, 17-21
 - INTERSECT, 3-5
 - MINUS, 3-5
 - UNION, 3-5
 - UNION ALL, 3-5
- 主キー
 - 値の生成, 13-81
- 主キー制約, 11-80
 - 索引, 14-46
 - 使用可能, 14-46
- 順序, 2-79, 13-81
 - 値の順序付け, 8-119
 - 値のリサイクル, 8-119
 - 値へのアクセス, 13-81
 - キャッシュされた値の数の変更, 8-119
 - 再起動, 16-2
 - 値, 13-84
 - 異なる番号, 8-120
 - 事前定義の制限, 13-83
- 最小値
 - 設定, 13-84
 - 設定または変更, 8-119
 - 排除, 8-119
- 最大値
 - 設定, 13-83
 - 設定または変更, 8-119
 - 排除, 8-119
- 再利用, 13-81
- 作成, 13-81
- 事前割当ての値, 13-84
- シノニム, 14-2
- 使用場所, 2-80
- 使用方法, 2-80
- 初期値の設定, 13-83
- 所定の制限での停止, 13-83
- 増分, 13-81
- 増分値の設定, 13-83
- データベースからの削除, 16-2
- 名前の変更, 16-85
- 付与
 - システム権限, 16-42
- 変更
 - 増分値, 8-119
 - 無制限での作成, 13-83
 - 連続する値の保証, 13-84
- 順序のリセット, 8-18
- 上位 N 番までの問合せ, 2-84
- 小計値
 - 導出, 17-19
- 条件
 - EXISTS, 5-12
 - IS OF *type*, 5-17
 - LIKE, 5-13
 - NULL, 5-12
 - SQL 構文内, 5-1
 - グループ比較, 5-7
 - 単純比較, 5-6
 - 比較, 5-4
 - 複合, 5-18
 - メンバーシップ, 5-10
 - レンジ, 5-11
 - 論理, 5-8
- 小数点文字, 2-54
 - 指定, 2-63
 - セッション用にリセット, 9-8
- 小のテスト, 5-5
- 初期化パラメータ
 - CIRCUITS, 9-37
 - セッション設定の変更, 9-6
- 書式
 - データベースからの戻り値, 2-59
 - データベースに格納された値, 2-59
 - 日付および数値
 - 「書式モデル」を参照
- 書式モデル, 2-59
 - XML, 2-76
 - 指定, 2-60

- 修飾子, 2-73
- 数値, 2-61
- 数値、要素, 2-62
- 日付, 2-66
 - 最大長, 2-66
 - 書式要素, 2-66
 - デフォルト書式, 2-66
 - 変更, 2-66
- 戻り値の書式の変更, 2-60
- 序数
 - 指定, 2-72
 - フルスペルでの表記, 2-72
- 新機能, xxiii

す

- 数値書式モデル, 2-61
- 数値ファンクション, 6-3
- スーパータイプ, 11-10
- スカラー副問合せ, 4-13
- スカラー副問合せ式, 4-13
- スキーマ
 - 作成, 13-79
 - セッションの変更, 9-11
 - 定義, 2-102
- スキーマ・オブジェクト, 2-102
 - 位置の保護, 14-2
 - オブジェクト型, 2-36
 - 監査
 - オプション, 11-59
 - 構造の検証, 11-40
 - 再コンパイル, 7-16
 - 再認可, 7-16
 - 削除, 16-19
 - 参照, 2-110, 9-11
 - 所有者の保護, 14-2
 - 代替名の提供, 14-2
 - デフォルトのバッファ・プールの定義, 17-56
 - 名前解決, 2-111
 - ネーミング, 2-106
 - 規則, 2-106
 - 計画, 2-110
 - 例, 2-109
 - ネームスペース, 2-107
 - パーティション索引, 2-104
 - パーティション表, 2-104
 - 部分, 2-103

- 他のスキーマ内, 2-112
- リスト, 2-102
- リモート・データベース, 2-113
- リモート・ビューへのアクセス, 12-33
- スター型変換, 2-100
- スタンドアロン・プロシージャ
 - 削除, 15-87
- スタンバイ・データベース
 - アーカイブ・ログの適用, 8-24
 - アクティブ, 8-34
 - 指定, 12-30
 - プライマリ状態の準備, 8-35
 - マウント, 8-18
 - メディア・リカバリの設計, 8-20
 - リカバリ, 8-21, 8-22
- ストアド・ファンクション, 12-47
- スナップショット
 - 「マテリアライズド・ビュー」を参照
- スナップショット・ログ
 - 「マテリアライズド・ビュー・ログ」を参照
- スループット
 - 最適化, 2-88

せ

- 制御ファイル
 - 再作成, 12-14
 - 再利用可能, 12-16, 12-24
 - スタンバイの作成, 8-33
 - バックアップ, 8-33
- 制限 ROWID, 2-32
 - 互換性と移行, 2-33
- 整合性制約
 - 「制約」を参照
- 整数
 - SQL 構文内, 2-53
 - 一意の値の生成, 13-81
 - 構文, 2-53
 - 指定, 2-12
 - 精度, 2-53
- 精度
 - NUMBER データ型, 2-12
 - 桁数, 2-54
- 制約
 - NOT NULL, 11-80
 - REF 表, 11-83
 - REF 列, 11-83

- 一意, 11-79
 - 索引の属性, 11-86
 - 使用可能, 14-46
 - 複合, 11-79
- 違反の行の格納, 10-52
- 外部キー, 11-82
- 既存の制約の変更, 10-62
- 削除, 10-63, 16-11
- 参照整合性, 11-81, 11-82
- 主キー, 11-80
 - 索引の属性, 11-86
 - 使用可能, 14-46
- 使用可能, 10-70, 14-44, 14-46
- 使用禁止, 10-70, 14-44
 - CASCADE, 14-49
- 制限事項, 11-79
- チェック, 11-82
 - 各 DML 文の終わり, 11-85
 - トランザクションの終わり, 11-85
 - トランザクションの始まり, 11-85
- 遅延可能, 11-85, 17-41
 - 適用, 9-10
- 追加, 10-55
- 定義, 11-72, 14-6
 - 表, 14-22
 - 列, 14-21
- トランザクション内の状態設定, 17-41
- ビュー, 11-77
 - 削除, 11-30, 16-22
 - 変更, 11-30
- 表, 11-77
- 複合一意, 11-79
- 有効範囲, 11-83
- 列, 11-78

セーブポイント

- 指定, 17-2
- 消去, 11-69
- ロールバック, 16-97

セキュリティ

- 規定, 14-77

セグメント

- 領域管理
 - 空きリストの使用, 14-70
 - 自動, 14-70
 - 手動, 14-70
 - ビットマップの使用, 14-70

セグメント属性句

- CREATE TABLE, 14-12

セッション

- CPU 時間の制限, 8-111
- オブジェクト・キャッシュ, 9-8
- グローバル・ネーム解決, 9-7
- 権限を持つユーザーに制限, 9-27
- 合計経過時間の制限, 8-111
- 終了, 9-26
- 制限, 9-28
- 切断, 9-25
- タイム・ゾーンの設定, 9-13
- 同時セッションの数, 9-62
- 特性の変更, 9-6
- 非アクティブ期間の制限, 8-106
- 付与
 - システム権限, 16-42
- プライベート SGA 領域の制限, 8-111
- 別のインスタンスへの切替え, 9-12
- 読み込まれたデータ・ブロックの制限, 8-111
- リソース・コスト制限の計算, 8-110
- リソース・コスト制限の変更, 8-110
- リソース・コストの制限, 8-110
- リソースの合計の制限, 8-106
- ロールの影響, 8-114

セッション制御文, 7-19

- PL/SQL サポート, 7-19

セッション・パラメータ

- INSTANCE, 9-12
- PLSQL_DEBUG, 9-12
- 設定の変更, 9-10

セッション・ロック

- 解放, 9-26

線形リグレーション・ファンクション, 6-118

そ

相関副問合せ, 7-12

相関名

- DELETE, 15-54
- SELECT, 17-15
- 索引の実表, 12-67

挿入

- 従来型, 16-56
- 条件付きマルチ表, 16-66
- ダイレクト・パス, 16-56
- 単一表, 16-60

- 同時の更新, 16-76
- マルチ表, 16-65
- ソート操作
 - 言語処理手順の変更, 9-8
- 属性
 - オブジェクト型の最大数, 14-20
 - ディメンションからの削除, 8-45
 - ディメンションの追加, 8-45
 - ディメンションの定義, 12-43
 - ユーザー定義型
 - Java フィールドへのマップ, 15-12

た

- 大 / 小のテスト, 5-5
- タイム・ゾーン
 - セッションの確認, 6-132
 - データベースへの指定, 12-31
- ダイレクト・パス INSERT, 16-56
- 単一行ファンクション, 6-3
 - その他, 6-6
- 単一表, 16-60
- 単項演算子, 3-2
- 単純式, 4-4
- 単純比較条件, 5-6
- ダンプ・ファイル
 - サイズの制限, 9-8

ち

- チェック制約, 11-82
- チェックポイント
 - 強制実行, 9-24
- 遅延可能制約, 17-41
- 中央値の値, 6-110
- 長期スタンバイ・リカバリ・モード, 8-24

つ

- 通貨記号
 - ISO, 2-62
 - セッションの設定, 9-8
 - 連合, 2-64
 - ローカル, 2-63

て

- 定義者権限ファンクション, 12-52
- 定数値
 - 「リテラル」を参照
- ディスパッチャ・プロセス
 - 終了, 9-104
 - 追加プロセスの作成, 9-104
- ディメンション
 - 階層
 - 削除, 8-45
 - 追加, 8-45
 - 定義, 12-41
 - 変更, 8-43
 - 作成, 12-39
 - システム権限の付与, 16-38
- 属性
 - 削除, 8-45
 - 追加, 8-45
 - 定義, 12-43
 - 変更, 8-43
 - データベースからの削除, 15-65
 - 無効のコンパイル, 8-45
 - 例, 12-43
- レベル
 - 削除, 8-45
 - 追加, 8-45
 - 定義, 12-41
- ディレクトリ
 - 「ディレクトリ・オブジェクト」を参照
- ディレクトリ・オブジェクト
 - システム権限の付与, 16-39
 - データベースからの削除, 15-67
 - オペレーティング・システムのディレクトリの別名, 12-44
 - 監査, 11-55
 - 再定義, 12-45
 - 作成, 12-44
- データ
 - UNDO
 - 保存, 13-75
 - 検索, 7-2
 - 集計
 - GROUP BY のグルーピング・セットの連結, 17-20
 - GROUP BY の複合列, 17-20
 - グルーピング・セット, 17-20

- テンポラリ・ファイルとしての指定, 14-19
- 入力に対する整合性チェック, 2-13
- 非依存, 14-2
- 頻繁に使用されるデータのキャッシュ, 10-33, 14-42
- データ・オブジェクト番号
 - 拡張 ROWID, 2-32
- データ型, 2-2
 - ANSI のサポート, 2-5
 - Any 型, 2-38
 - BFILE, 2-9, 2-30
 - BLOB, 2-9, 2-30
 - CHAR, 2-8, 2-10
 - CLOB, 2-9, 2-31
 - DATE, 2-7, 2-18
 - INTERVAL DAY TO SECOND, 2-22
 - INTERVAL YEAR TO MONTH, 2-22
 - LONG, 2-7, 2-15
 - LONG RAW, 2-8, 2-25
 - NCHAR, 2-9, 2-11
 - NCLOB, 2-9, 2-31
 - NUMBER, 2-7, 2-12
 - NVARCHAR2, 2-7, 2-11
 - Oracle が提供する型, 2-38
 - RAW, 2-8, 2-25
 - ROWID, 2-8, 2-31
 - TIMESTAMP, 2-20
 - TIMESTAMP WITH LOCAL TIME ZONE, 2-22
 - TIMESTAMP WITH TIME ZONE, 2-21
 - UROWID, 2-8, 2-33
 - VARCHAR, 2-12
 - VARCHAR2, 2-7, 2-11
 - XML 型, 2-39
 - 期間, 2-17
 - 空間型, 2-41
 - 組込み, 2-7
 - コレクション型値への変換, 6-24
 - 統計情報の関連付け, 11-48
 - 長さのセマンティクス, 2-10, 2-11
 - 日時, 2-17
 - 比較規則, 2-42
 - 別のデータ型への変換, 6-24
 - メディア型, 2-42
 - 文字, 2-9
 - ユーザー定義, 2-36
- データ操作言語 (DML)
 - 関係する行の取出し, 15-54, 16-65, 17-72
- 行の取出し, 15-54, 16-65, 17-72
- 索引付け中の許可, 8-57
- 操作
 - 索引再構築中, 10-69
 - 索引作成中, 12-72
 - トリガー, 14-82
- 操作の制限, 9-28
- トリガー
 - LOB 列および属性, 2-28
- パラレル化, 14-43
- 文, 7-18
 - PL/SQL サポート, 7-18
- データ定義言語 (DDL)
 - イベントおよびトリガー, 14-83
 - 排他的アクセスを必要とする文, 7-16
 - 文, 7-16
 - PL/SQL サポート, 7-16
 - 再コンパイルの原因, 7-16
 - 暗黙的コミット, 7-16
- データ・ディクショナリ
 - コメントの追加, 11-66
- データ非消失モード
 - 設定, 12-30
- データ・ファイル
 - オフライン化, 8-27
 - オンライン化, 8-27
 - オンラインの情報更新, 9-25
 - オンライン・バックアップ, 10-89
 - オンライン・バックアップの終了, 8-27, 10-89
 - サイズ, 16-29
 - サイズの再変更, 8-27
 - サイズの変更, 8-27
 - 再利用, 16-29
 - 削除, 16-11
 - システム生成, 8-26
 - 指定, 16-27
 - 表領域, 14-66
 - 自動拡張の使用可能化, 14-66
 - 新規作成, 8-26
 - 損失または破損した場合の再作成, 8-26
 - データベースへの指定, 12-27
 - 名前の変更, 8-29
 - 破損のリカバリ, 8-20
 - メディア・リカバリの設計, 8-20
 - リカバリ, 8-22

データベース

Oracle7 のデータ・ディクショナリからの変換,
8-37

REDO ログ生成の可能化, 8-18

アーカイブ・モード

指定, 12-26

アカウント

作成, 15-30

アクティビティの再開, 9-27

アクティビティの停止, 9-27

アップグレード, 8-37

インスタンス, 12-26

オープン, 8-18, 12-21

メディア・リカバリ後, 8-19

オンライン

ログ・ファイルの追加, 8-30

各国語キャラクタ・セット

指定, 12-27

管理リカバリ, 8-12

キャッシュ

バッファ, 9-43

キャラクタ・セット

指定, 12-27

変更, 8-35

キャラクタ・セットの指定, 12-27

キャラクタ・セットの変更, 8-35

休止状態, 9-28

グローバル名の変更, 8-38

再作成の準備, 8-33

作成, 12-21

時間ベースのリカバリ, 8-21

システム権限の付与, 16-38

自動拡張の使用可能化, 12-28

スクリプトの作成, 8-33

スタンバイ

PROTECTED モード, 8-34

UNPROTECTED モード, 8-34

指定, 12-30

ログ・ファイルの追加, 8-30

スタンバイ状態の準備, 8-35

すべてのデータの消去, 12-21

制御ファイルの再作成, 12-14

制御ファイルの再利用, 12-24

接続文字列, 2-114

タイム・ゾーン

設定, 8-36, 12-31

判断, 6-46

データ非消失モード, 8-34, 12-30

データ・ファイル

指定, 12-27

変更, 8-26

テンポラリ・ファイル

変更, 8-26

特性の変更, 12-14

取消しベースのリカバリ, 8-21

終了, 8-24

名前の変更, 12-14, 12-16

ネーミング, 8-18

破損の再構成, 8-20

バックアップの終了, 8-26

ブロック

サイズの指定, 14-67

別のデータベースへのサブセットの移動, 10-51

変更, 8-9

変更ベースのリカバリ, 8-21

マウント, 8-18, 12-21

メディア・リカバリの設計, 8-20

ユーザーに対する無制限のリソース, 13-67

ユーザーに対するリソース制限, 13-65

ユーザーへの読取り専用トランザクションの制限,
8-19

読み書き両用, 8-18

読取り専用, 8-18

リカバリ, 8-19, 8-21

テスト, 8-22

破損ブロックの許容, 8-22

バックアップ制御ファイルの使用, 8-21

リセット

現行のログ順序, 8-19

前回のバージョン, 8-37

リモート

アクセス, 7-15

サービス名, 12-37

接続, 12-36

挿入, 16-61

表ロック, 16-74

ユーザーの認証, 12-37

ログ・ファイル

指定, 12-25

変更, 8-29

データベース・イベント

監査, 14-84

透過的なログイン, 14-84

トリガー, 14-84

データベース・オブジェクト

削除, 16-20

スキーマ, 2-102

非スキーマ, 2-103

データベース・トリガー

「トリガー」を参照

データベース・リンク, 7-15

共有, 12-35

クローズ, 9-3

現行のユーザー, 12-36

構文, 2-113

作成, 2-113, 12-33

参照, 2-114

システム権限の付与, 16-38

シノニムの作成, 14-4

データベースからの削除, 15-63

ネーミング, 2-113

パブリック, 12-35

削除, 15-63

ユーザー名およびパスワード, 2-114

データ変換, 2-46

暗黙的

デメリット, 2-46

暗黙的と明示的, 2-46

暗黙的に行う場合, 2-47, 2-49

キャラクタ・データ型, 2-48

明示的に行う場合, 2-50

デカルト演算, 7-11

テキスト

CHAR および VARCHAR2 データ型のプロパティ,
2-52

SQL 構文内, 2-52

構文, 2-52

日付および数値書式, 2-59

デフォルトの索引の抑制, 13-17

テンポラリ・ファイル

オフライン化, 8-28

オンライン化, 8-28

サイズ, 16-29

サイズの再変更, 8-28

再利用, 16-29

削除, 8-28

指定, 14-74, 16-27

自動拡張, 14-75

自動拡張の使用可能化, 8-28

自動拡張の使用禁止化, 8-28

名前の変更, 8-29

と

問合せ, 7-2, 17-4

SELECT 構文のリスト, 7-3

値で戻された行のグループ化, 17-19

階層

「階層問合せ」を参照

階層の順序付け, 17-21

外部結合, 17-15

行ロック, 17-23

結果のソート, 7-9

結合, 7-9, 17-16

構文, 7-2

コメント, 7-3

上位 N 番, 2-84

関連

左関連, 17-15

定義済, 7-2

トップレベル, 7-2

任意の行からの選択, 17-14

ヒント, 7-3

複合, 7-9

複数表の参照, 7-9

分散, 7-15

戻された行の順序, 17-21

等価結合, 7-10

ディメンションの定義, 12-42

等価性のテスト, 5-5, 5-10

統計情報

関連付けの削除, 15-59

厳密な計算, 11-36

索引, 12-72

索引再構築中の収集, 8-57

索引の使用, 8-63

推定, 11-39

スカラー・オブジェクト属性

収集, 11-31

スキーマ・オブジェクト

削除, 11-31

収集, 11-31

データ・ディクショナリからの削除, 11-43

ユーザー定義

削除, 15-72, 15-73, 15-86, 16-7, 16-15

統計タイプ

関連付け

索引タイプ, 11-48

データ型, 11-48

- ドメイン索引, 11-48
- パッケージ, 11-48
- ファンクション, 11-48
- 列, 11-48
- 関連付けの解除
 - 型, 15-57
 - 索引タイプ, 15-57
 - ドメイン索引, 15-57
 - パッケージ, 15-57
 - ファンクション, 15-57
 - 列, 15-57
- 特殊文字
 - パスワード, 13-69
- ドメイン索引, 12-58, 12-76, 12-84
- LONG 列, 10-57
- 再構築, 8-57
- 削除ルーチンの起動, 16-7
- データベースからの削除, 15-71
- 統計情報の関連付け, 11-48
- 変更, 8-61
- 文字列変更の指定, 8-61
- ユーザ定義の CPU および I/O コストの判断,
16-23
- トランザクション
 - 暗黙的コミット, 7-16, 7-18, 7-19
 - インダウト
 - 強制実行, 11-70
 - コミット, 11-69
 - 変換, 17-49
 - 完了, 9-25
 - コメント, 11-70
 - 自動コミット, 11-69
 - 終了, 11-69
 - セーブポイント, 17-2
 - ネーミング, 17-49
 - 分散トランザクションの強制実行, 9-3
 - 分離レベル, 17-46
 - 変更の確定, 11-69
 - 読み書き両用, 17-46
 - 読取り専用, 17-46
 - ロールバック, 9-26, 13-75, 16-96
 - セーブポイント, 16-97
 - ロックの解除, 11-69
 - 割当て
 - ロールバック・セグメント, 17-46
- トランザクション制御文, 7-18
- PL/SQL サポート, 7-18

- トリガー
 - AFTER, 14-80
 - BEFORE, 14-80
 - DDL イベント, 14-83
 - DML 操作, 14-82
 - INSTEAD OF, 14-81
 - 削除, 15-40
 - SQL 例, 14-88
 - 起動の順序, 14-81
 - 行値
 - 新旧, 14-86
 - 行の指定, 14-86
 - コンパイル, 11-2, 11-4
 - 再作成, 14-79
 - 作成, 14-77
 - 複数, 14-81
 - 実行
 - PL/SQL ブロック, 14-87
 - 外部プロシージャでの実行, 14-87
 - 使用可能, 10-71, 11-2, 11-3, 14-77
 - 使用禁止, 10-71, 11-2, 11-3
 - 制限, 14-86
 - データベース
 - 削除, 16-12, 16-20
 - 変更, 11-3
 - データベース・イベント, 14-84
 - データベースからの削除, 16-12
 - 名前の変更, 11-3
 - ビュー, 14-81
 - 付与
 - システム権限, 16-44
 - 文, 14-86
- 取消し
 - システム管理, 8-115, 12-29
 - ロールバック, 8-115, 12-29

な

- 内部 LOB, 2-25
- 内部結合, 17-16
- 夏時間, 2-23

に

- 日時式, 4-10
- 日時書式要素, 2-66
 - ISO 規格, 2-70

- RR, 2-71
- 大文字化, 2-66
- グローバルゼーション・サポート, 2-70
- 接尾辞, 2-72
- 日時データ型, 2-17
 - 夏時間, 2-23
- 日時の演算
 - 夏時間の演算, 2-23
- 日時ファンクション, 6-5
- 日時フィールド
 - 日時または期間値からの抽出, 6-56
- 日時列
 - DATE 列からの作成, 10-57

ね

- ネームスペース
 - オブジェクトのネーミング規則, 2-107
 - スキーマ・オブジェクト, 2-107
 - 非スキーマ・オブジェクト, 2-108
- ネステッド・ループ・ジョイン
 - 最適化, 9-9
- ネスト解除集合, 17-15
 - 例, 17-35
- ネストした表, 2-38
 - VARRAY との比較, 2-45
 - 記憶特性, 10-65, 14-35
 - 索引構成表としての定義, 10-65
 - 索引列, 12-67
 - 作成, 15-3, 15-8
 - 仕様部の削除, 16-14
 - 比較規則, 2-45
 - ビュー内での更新, 14-81
 - 変更, 10-62
 - 本体の削除, 16-17
 - 戻り値の変更, 10-62
- ネストした副問合せ, 7-12

は

- パーティション
 - LOB 記憶特性, 10-67
 - 値の変更, 17-68
 - エクステンツ
 - 索引への割当て, 8-55
 - エクステンツの割当て, 10-31
 - 記憶特性, 14-25

- 行の削除, 10-48, 15-52
- 行の挿入, 16-61
- 行の追加, 16-56
- コンボジット, 2-104
 - 指定, 14-40
- 索引, 12-73
- 削除, 10-47
- 挿入操作のロギング, 10-29
- 追加, 10-38
- 名前の変更, 10-42
- ハッシュ, 2-104
 - 合わせる, 10-47
 - 指定, 14-39
 - 追加, 10-46
- 非パーティション表への変換, 10-51
- 表領域
 - 定義, 14-25
- 物理属性
 - 変更, 10-28
- 分割, 10-49
- 別のセグメントへの移動, 10-43
- 変更, 10-38, 10-39
- マージ, 10-50
- 未使用領域の解放, 10-32
- リスト・パーティションの追加, 10-46
- リテラル値に基づくパーティション, 14-38
- レンジ, 2-104
 - 指定, 14-36
 - 追加, 10-45
- ロギング属性, 14-25
- ロック, 16-72
- パーティション化
 - 句
 - ALTER INDEX, 8-51
 - ALTER MATERIALIZED VIEW, 8-77, 8-81
 - ALTER MATERIALIZED VIEW LOG, 8-92, 8-93
 - ALTER TABLE, 10-38
 - CREATE MATERIALIZED VIEW, 13-12, 13-15
 - CREATE MATERIALIZED VIEW LOG, 13-32, 13-34
 - レンジ, 14-14
- パーティション化された索引構成表
 - セカンダリ索引の更新, 8-65
- パーティション交換
 - 制限, 10-53
- パーティション索引, 2-104, 12-58, 12-74

- ユーザー定義, 11-87, 12-73, 14-47
- パーティション表, 2-104
- バイナリ演算子, 3-2
- バイナリ・ラージ・オブジェクト
 - 「BLOB」を参照
- 配布
 - ヒント, 2-98
- パスワード
 - 期限切れ, 15-34
 - 使用および再利用の制限, 13-69
 - 使用禁止化, 13-69
 - 特殊文字, 13-69
 - パラメータ
 - ALTER PROFILE, 13-70
 - CREATE PROFILE, 13-66
 - 複雑さに対する保証, 13-69
 - 猶予期間, 13-69
 - ロック, 13-69
- パッケージ
 - 再定義, 13-47
 - 作成, 13-46
 - 実行時のコンパイルの回避, 8-100
 - 実行者権限, 13-48
 - シノニム, 14-2
 - スキーマと権限の指定, 13-48
 - データベースからの削除, 15-85
 - 統計情報の関連付け, 11-48
 - 統計タイプの削除, 15-86
 - 明示的な再コンパイル, 8-100
- パッケージ・プロシージャ
 - 削除, 15-87
- パッケージ本体
 - 再作成, 13-52
 - 作成, 13-51
 - データベースからの削除, 15-85
- ハッシュ・クラスタ
 - 作成, 12-6
 - 単一表、作成, 12-7
 - ハッシュ・ファンクションの指定, 12-7
- ハッシュ結合
 - 使用可能および使用禁止, 9-7
 - メモリーの割当て, 9-7
- ハッシュ・パーティション
 - 合わせる, 10-40
 - 追加, 10-46
- ハッシュ・パーティション化句
 - CREATE TABLE, 14-16, 14-39

- パフォーマンス
 - 共有プール, 9-103
 - 索引アクセス・パスに対する最適化, 9-9
 - セッション・オブティマイザ・アプローチ, 9-9
 - ネステッド・ループ・ジョインに対する最適化, 9-9
- パブリック・シノニム, 14-3
 - 削除, 16-4
- パブリック・データベース・リンク
 - 削除, 15-63
- パブリック・ロールバック・セグメント, 13-76
- パラメータ
 - 構文
 - オプション, A-4
 - 必須, A-3
- パラメータ・ファイル
 - 作成, 13-56
- パラレル結合
 - PQ_DISTRIBUTE ヒント, 2-98
- パラレル実行
 - DDL 文, 9-4
 - DML 文, 9-4
 - ヒント, 2-97
- 範囲条件, 5-11

ひ

- ヒープ構成表
 - 作成, 14-6
- 比較条件, 5-4
- 比較セマンティクス
 - 空白埋め, 2-44
 - 非空白埋め, 2-43
 - 文字列, 2-43
- 比較ファンクション
 - MAP, 15-28
 - ORDER, 15-28
- 引数
 - 演算子, 3-1
- 非空白埋め比較セマンティクス, 2-43
- 非スキーマ・オブジェクト
 - ネームスペース, 2-108
 - リスト, 2-103
- ヒストグラム
 - 等幅の作成, 6-193
- 左側外部結合, 17-16

- 日付
 - 算術, 2-19
 - 比較規則, 2-42
- 日時の演算
 - 境界ケース, 9-11
- 日付書式モデル, 2-66
 - 句読点, 2-66
 - テキスト, 2-66
- 日付ファンクション, 6-5
- ビット・ベクトル
 - 数値への変換, 6-22
- ビットマップ索引, 12-65
- ビュー
 - オブジェクト・サブビューの作成, 15-43
 - オブジェクトの作成, 15-42
 - 結合
 - キー保存表, 15-44
 - 結合を使用した更新可能化, 15-44
 - 再コンパイル, 11-28
 - 再作成, 15-40
 - 削除
 - 実表の行, 15-49
 - データベース, 16-21
 - 作成
 - コメント, 11-66
 - 実表の前, 15-40
 - 複数, 13-79
 - 実表
 - 行の追加, 16-56
 - シノニム, 14-2
 - 制約, 11-77
 - 制約の削除, 11-30
 - 制約の変更, 11-30
 - 定義, 15-37
 - データ検索, 17-4
 - 名前の変更, 16-85
 - 副問合せ, 15-43
 - 制限, 15-45
 - 付与
 - システム権限, 16-45
 - 変更
 - 実表の値, 17-64
 - 定義, 16-21
 - リモート・ビューへのアクセス, 12-33
 - 更新可能, 15-44
- ビュー制約, 11-77
 - 削除, 16-22

- 表
 - LOB 記憶特性, 14-25
 - SQL 例, 14-51
 - 新しいセグメントへの移動, 10-68
 - 移行行および連鎖行, 11-42
 - エクステンツの割当て, 10-31
 - オブジェクト
 - 作成, 14-8
 - 外部, 14-27
 - 作成, 14-29
 - 制限, 14-30
 - 外部構成, 14-27
 - 記憶域属性, 17-50
 - 定義, 14-6
 - 記憶域プロパティ, 14-23, 14-31
 - 記憶特性
 - 定義, 14-25
 - 既存値の変更, 17-64
 - キャッシュへのブロックの保存, 10-33, 14-42
 - 行の削除, 15-49
 - 行の追加, 16-56
 - クラスタへの割当て, 14-31
 - 構成の定義, 14-27
 - 構造の検証, 11-40
 - コメントの作成, 11-66
 - 索引構成, 14-27
 - オーバーフロー・セグメント, 14-28
 - 索引ブロックの領域, 10-36, 14-28
 - 索引構成表の移動, 10-68
 - 削除
 - クラスタ, 15-61
 - 索引, 16-7
 - 所有者, 16-20
 - パーティション, 16-7
 - 作成, 14-6
 - 複数, 13-79
 - サブパーティション属性, 10-38
 - シノニム, 14-2
 - 使用禁止の索引, 9-13
 - 制限
 - 参照, 11-83
 - ブロックのレコード, 10-34
 - 制約, 11-77
 - ダイレクト・パスを使用した挿入, 16-56
 - データ検索, 17-4
 - データベースからの削除, 16-6
 - データベースの外部へのデータの格納, 14-29

- デフォルト物理属性
 - 変更, 10-28
- テンポラリ
 - セッション固有, 14-19
 - データの存続期間, 14-22
 - トランザクション固有, 14-19
- 問合せ内で結合, 17-16
- 統計情報の収集, 10-33, 11-35
- 統計タイプの削除, 16-7
- 名前の変更, 16-85
- ネスト
 - 記憶特性, 14-35
 - 作成, 15-18
- パーティション化, 2-104, 14-6, 14-36
 - 行のパーティション間での移動可能化, 10-54
 - デフォルト属性, 10-38
- パーティション属性, 10-38
- パラレル化
 - デフォルト値の設定, 14-43
- パラレル作成, 14-43
- ヒープ構成, 14-27
- 非クラスタ化, 15-60
- ビューを介した更新, 15-44
- 表領域
 - 定義, 14-6, 14-25
- 副問合せへの行の挿入, 14-49
- 物理属性
 - 変更, 10-28
- 付与
 - システム権限, 16-43
- 並列度
 - 指定, 14-6
- 並列度の変更, 10-41
- 別名, 2-116
 - CREATE INDEX, 12-67
 - DELETE, 15-54
- 未使用領域の解放, 10-32
- リモート・ビューへのアクセス, 12-33
- リレーショナル
 - 作成, 14-7
- ロギング
 - 挿入操作, 10-29
 - 表作成, 14-25
- ロック, 16-72
- 表名前の変更, 10-35
- 標準 SQL, B-1
 - Oracle 拡張機能, B-11
- 表制約, 11-77
 - ALTER TABLE, 10-56
 - CREATE TABLE, 14-22
 - 定義済, 11-73
- 表のファンクション
 - 作成, 12-54
- 表領域, 10-87
 - UNDO
 - 削除, 16-10
 - 作成, 14-65
 - 一時オブジェクト, 14-68
 - 永続オブジェクト, 14-68
 - エクステント管理, 14-69, 14-75
 - エクステント・サイズ, 14-67
 - オフライン化, 10-88, 14-68
 - オンライン化, 10-87, 14-68
 - オンライン・バックアップの終了, 10-89
 - 書込み操作の実行可能化, 10-89
 - コンテンツの削除, 16-10
 - 作成, 14-62
 - システム権限の付与, 16-43
- 指定
 - 索引再構築, 10-69
 - データ・ファイル, 14-66
 - 表, 14-23
 - ユーザー, 15-33
- 自動拡張の使用可能化, 10-86
- 自動セグメント領域管理, 2-15, 14-25, 14-70
- セッション存続期間, 14-73
- 損失または破損した場合の再構成, 8-20, 8-26
- データ・ファイル
 - 追加, 10-85
 - 名前の変更, 10-85
- データ・ファイルのバックアップ, 10-89
- データベースからの削除, 16-9
- デフォルトの一時, 8-37
 - 名前の認識, 8-37
- デフォルトの記憶域属性, 17-50
- テンポラリ
 - 作成, 14-73
 - ユーザーへの指定, 15-33
- テンポラリ・ファイル
 - 追加, 10-85
- 取消し
 - 作成, 12-29
 - 変更, 10-85

変換

一時的から永続的, 10-90

永続的から一時的, 10-90

未使用エクステンツの結合, 10-90

未使用エクステンツ・サイズ, 10-87

メディア・リカバリの設計, 8-20

ユーザーの領域の割当て, 15-34

読取り専用, 10-89

リカバリ, 8-20, 8-22

ローカル管理, 17-50

テンポラリ, 14-75

変更, 10-85

ロギング属性, 10-90, 14-67

表ロック

EXCLUSIVE, 16-73, 16-74

ROW EXCLUSIVE, 16-73, 16-74

ROW SHARE, 16-73, 16-74

SHARE, 16-73

SHARE ROW EXCLUSIVE, 16-74

SHARE UPDATE, 16-74

継続期間, 16-72

サブパーティション, 16-73

使用可能, 10-70

使用禁止, 10-71

問合せ, 16-72

パーティション, 16-73

モード, 16-74

リモート・データベース, 16-74

ヒント, 7-3

ALL_ROWS ヒント, 2-88

AND_EQUAL ヒント, 2-89

CACHE ヒント, 2-89

CLUSTER ヒント, 2-90

FIRST_ROWS ヒント, 2-90

FULL ヒント, 2-91

HASH ヒント, 2-91

INDEX_ASC ヒント, 2-92

INDEX_DESC ヒント, 2-92

INDEX ヒント, 2-91

NO_EXPAND ヒント, 2-94

NO_MERGE ヒント, 2-95

NO_PUSH_PRED ヒント, 2-95

NOCACHE ヒント, 2-94

NOPARALLEL ヒント, 2-95

NOREWRITE ヒント, 2-96

ORDERED ヒント, 2-96

PARALLEL ヒント, 2-97

PQ_DISTRIBUTE ヒント, 2-98

PUSH_PRED ヒント, 2-99

PUSH_SUBQ ヒント, 2-99

REWRITE ヒント, 2-99

ROWID ヒント, 2-99

RULE ヒント, 2-100

SQL 文, 2-86

USE_CONCAT ヒント, 2-101

USE_MERGE ヒント, 2-101

USE_NL ヒント, 2-101

オブティマイザの受渡し, 17-64

構文, 2-87

ふ

ファイル

REDO ログ・ファイル・グループとしての指定, 16-27

データ・ファイルとしての指定, 16-27

テンポラリ・ファイルとしての指定, 16-27

ファンクション

user_defined

式, 4-11

3GL、コール, 13-2

「SQL ファンクション」を参照

built_in

式, 4-11

COMMIT または ROLLBACK 文の発行, 9-3

DECODE, 6-47

外部, 12-47, 13-58

逆分散, 6-108, 6-110

結果の戻りの繰返し, 12-54

権限, 11-13, 15-9

コール, 11-63

コレクションの戻り, 12-54

再作成, 12-50, 12-88

実行, 11-63

パラレル問合せプロセス, 12-53

実行時のコンパイルの回避, 8-46

シノニム, 14-2

スキーマ, 11-13, 15-9

スキーマとユーザー権限の指定, 12-52

ストアド, 12-47

線形リグレーション, 6-118

宣言の変更, 12-50

データベースからの削除, 15-69

統計情報の関連付け, 11-48

- 日時, 6-5
- ネーミング規則, 2-108
- パーティション化
 - パラレル問合せプロセス, 12-53
- 表, 12-54
- 分析
 - ユーザー定義, 12-55
- 保存されたコピーの使用, 12-53
- 無効の再コンパイル, 8-46
- 明示的な再コンパイル, 8-47
- 戻り値の格納, 11-64
- 戻り値のデータ型, 12-52
- ユーザー定義, 6-196
- 集計, 12-55
- 例, 12-56
- ファンクション索引の定義, 12-67
- ファンクション定義の変更, 12-50
- ファンクション索引, 12-58
 - クエリー・リライト, 9-9
- 作成, 12-67
- 使用可能, 8-57, 8-62, 9-94
- 使用可能および使用禁止, 8-57
- 使用禁止, 9-94
- ファンクション式
 - ユーザー定義, 4-11
 - 組込み, 4-11
- 不完全なオブジェクト型, 15-3
 - 作成, 15-3, 15-4
- 複合一意制約, 11-79
- 複合外部キー, 11-81
- 複合式, 4-5
- 複合主キー, 11-80
- 複合条件, 5-18
- 副問合せ, 7-2, 7-12, 17-4
 - インライン・ビュー, 7-12
 - 拡張された副問合せのネスト解除, 7-14
 - 式のかわりに使用, 4-13
 - スカラー, 4-13
 - 式, 4-13
 - 制限, 17-14
 - 相関, 7-12
 - 定義済, 7-2
 - 名前の割当て, 17-10
 - ネスト解除, 7-13
 - ネストした, 7-12
 - 表データの挿入, 14-49
 - ファクタリング, 17-10
 - 副問合せを含む, 7-12
 - 副問合せのネスト解除, 7-13
 - 物理属性句
 - ALTER CLUSTER, 8-4
 - ALTER INDEX, 8-50, 8-56
 - ALTER MATERIALIZED VIEW, 8-79
 - ALTER MATERIALIZED VIEW LOG, 8-92
 - ALTER TABLE, 10-28
 - CREATE CLUSTER, 12-3
 - CREATE MATERIALIZED VIEW, 13-10
 - CREATE MATERIALIZED VIEW LOG, 13-32
 - CREATE TABLE, 14-12, 14-23
 - 制約, 11-77
 - 不等価性のテスト, 5-10
 - 浮動小数点数, 2-12, 2-14
 - 不等性のテスト, 5-5
 - 負の位取り, 2-13
 - プライベート・アウトライン
 - オブティマイザによる使用, 9-14
 - プラン・スタビリティ, 13-42
 - プリコンパイラ
 - Oracle, 1-4
 - フルスペルで表した数
 - 指定, 2-72
 - プロキシ句
 - ALTER USER, 11-22, 11-24
 - プロシージャ
 - 3GL、コール, 13-2
 - COMMIT または ROLLBACK 文の発行, 9-3
 - 依存するローカル・オブジェクトの無効化, 15-88
 - 外部, 13-56, 13-58
 - リモート・データベースからの実行, 13-3
 - 共有プール, 9-103
 - 権限, 11-13, 15-9
 - コール, 11-63
 - 再コンパイル, 8-103
 - 再作成, 13-60
 - 作成, 13-56, 13-58
 - 実行, 11-63
 - 実行時のコンパイルの回避, 8-104
 - シノニム, 14-2
 - スキーマ, 11-13, 15-9
 - スキーマと権限の指定, 13-62
 - 宣言
 - C 関数, 13-63
 - Java メソッド, 13-63
 - データベースからの削除, 15-87

- ネーミング規則, 2-108
- 付与
 - システム権限, 16-41
- 明示的なコンパイル, 8-104
- プロファイル
 - 作成, 13-65
 - 例, 13-70
- データベースからの削除, 15-89
- 付与
 - システム権限, 16-41
- 変更の例, 8-108
- ユーザーへの割当て, 15-34
- ユーザーへの割当ての削除, 15-89
- リソース制限の削除, 8-106
- リソース制限の追加, 8-106
- リソース制限の変更, 8-106
- 分散問合せ, 7-15
 - 制限, 7-15
- 分析関数
 - 逆分散, 6-108, 6-110
- 分析ファンクション, 6-8
 - AVG, 6-19
 - CORR, 6-34
 - COUNT, 6-37
 - COVAR_POP, 6-39
 - COVAR_SAMP, 6-41
 - CUME_DIST, 6-43
 - DENSE_RANK, 6-49
 - FIRST, 6-58
 - FIRST_VALUE, 6-60
 - LAG, 6-71
 - LAST, 6-73
 - LAST_VALUE, 6-76
 - LEAD, 6-78
 - MAX, 6-86
 - MIN, 6-88
 - NTILE, 6-99
 - OVER 句, 6-8, 6-9
 - PERCENT_CONT, 6-108
 - PERCENT_DISC, 6-110
 - PERCENT_RANK, 6-106
 - RANK, 6-112
 - RATIO_TO_REPORT, 6-114
 - ROW_NUMBER, 6-128
 - STDDEV, 6-136
 - STDDEV_POP, 6-138
 - STDDEV_SAMP, 6-139

- SUM, 6-142
- VAR_POP, 6-188
- VAR_SAMP, 6-189
- VARIANCE, 6-191
- 構文, 6-8
- 線形リグレーション, 6-118
- ユーザー定義, 6-9, 12-55

へ

- 別名
 - 問合せおよび副問合せでの指定, 17-15
 - ビュー問合せの式, 15-41
 - 列, 7-3
- 変換
 - 文字列から日付への規則, 2-75
- 変換ファンクション, 6-5
- 変数式, 4-15

ま

- マスター・データベース, 13-5
- マスター表, 13-5
- マッピング表
 - 索引構成表, 10-69
- マテリアライズド結合ビュー, 13-29
- マテリアライズド・ビュー, 8-83, 13-17
 - LOB 記憶域属性, 8-80
 - ROWID, 13-20
 - ROWID の値
 - マスター表への記録, 8-95
 - ROWID ベースから主キー・ベースへの変更, 8-85
 - 移入, 13-16
 - オブジェクト型マテリアライズド・ビューの作成, 13-12
 - 完全リフレッシュ, 8-84, 13-18
 - 記憶域属性, 13-13
 - 変更, 8-80
 - キャッシュへのブロックの保存, 8-82
 - 強制リフレッシュ, 8-84
 - クエリー・リライト
 - 資格, 11-86
 - 使用可能および使用禁止, 8-86
 - クエリー・リライトの使用可能 / 使用禁止, 13-21
 - 結合, 13-29
 - 更新可能, 13-21
 - 高速リフレッシュ, 8-83, 13-17, 13-18

- コメントの作成, 11-66
- 再検証, 8-87
- 索引特性
 - 変更, 8-81
- 作成, 13-5
- システム権限の付与, 16-40
- シノニム, 14-2
- 主キー, 13-19
 - マスター表の値の記録, 8-95
- 主キー・ベースへの変更, 8-95
- 制約, 11-86
- 次の COMMIT でのリフレッシュ, 8-84, 13-18
- データ・ウェアハウス, 13-5
- データ検索, 17-4
- データベースからの削除, 15-78
- デフォルトの索引の作成の抑制, 13-17
- パーティション, 8-81
- 副問合せ, 13-22
- 物理属性, 13-13
 - 変更, 8-80
- 並列度, 8-81, 8-93
 - 作成, 13-15
- マスター表の DML 後でのリフレッシュ, 8-85, 13-19
- マスター表の削除, 15-79
- メンテナンスする索引, 13-17
- 有効範囲の制限, 13-12
- リフレッシュ時刻
 - 変更, 8-83
- リフレッシュ中に再作成, 8-84
- リフレッシュ・モード
 - 変更, 8-83
- 例, 13-24, 13-36
- レプリケーション, 13-5
- ロギングの変更, 8-82
- マテリアライズド・ビュー・ログ, 13-29
 - ROWID に基づく, 8-95
 - 新しい値の除外, 8-96
 - 新しい値の保存, 8-96
 - オブジェクト ID に基づく, 8-95
- 記憶域属性
 - 指定, 13-32
- 高速リフレッシュに必要な, 13-29
- 作成, 13-29
- 作成の平行化, 13-33
- データベースからの削除, 15-80
- パーティション, 13-34

- 物理属性
 - 指定, 13-32
- 物理属性の変更, 8-93
- 古い値の保存, 13-35
- 列の追加, 8-95
 - ロギングの変更, 8-94
- マルチスレッド・サーバー
 - 「共有サーバー」を参照
- マルチ表の挿入, 16-65
 - 条件, 16-65
 - 無条件, 16-65

み

- 右側外部結合, 17-16
- 未ソート索引, 12-70

め

- 明示的なデータ変換, 2-46, 2-50
- メソッド
 - サブタイプでのオーバーライドの回避, 15-13
 - 実装なし, 15-13
 - スーパータイプのメソッドのオーバーライド, 15-13
 - スタティック, 15-12
- メディア・リカバリ
 - 起動時のメディア・リカバリの回避, 8-27
 - 指定の REDO ログ, 8-20
 - 準備, 8-29
 - 使用禁止, 8-26
 - 使用中の実行, 8-24
 - スタンバイ・データベース, 8-20
 - 制限事項, 8-20
 - 設計, 8-20
 - 長期スタンバイ・リカバリ, 8-24
 - データ・ファイル, 8-20
 - データベース, 8-20
 - 平行化, 8-23
 - 表領域, 8-20
- メンバーシップ条件, 5-10

も

- 文字
 - 単一文字の比較規則, 2-44
- 文字長のセマンティクス, 10-58

- 文字ファンクション, 6-4
- 文字リテラル
 - 「テキスト」を参照
- 文字列
 - ASCII 値への変換, 6-17
 - Unicode への変換, 6-31
 - 各国語キャラクタ・セット, 2-11
 - 可変長, 2-11, 2-15
 - 完全一致, 2-73
 - 固定長, 2-10
 - 長さが 0 (ゼロ), 2-10
 - 比較規則, 2-43

ゆ

- 有効範囲制約, 11-83
- ユーザー
 - SQL 例, 15-34
 - アカウントのロック, 15-34
 - 一時表領域, 15-33
 - 外部, 13-73, 15-32
 - グローバル, 13-73, 15-32
 - グローバル認証の変更, 11-23
 - 最大同時ユーザー数, 9-62
 - 作成, 15-30
 - データベースからの削除, 16-19
 - データベース・リンク, 12-36
 - デフォルトの表領域, 15-33
 - 認証の変更, 11-24
 - パスワードの期限切れ, 15-34
 - 表およびビューへのアクセスの制限, 16-72
 - 付与
 - システム権限, 16-45
 - リモート・サーバーの認証, 12-37
 - 領域の割当て, 15-34
 - ローカル, 13-73, 15-32
 - 割当て
 - デフォルト・ロール, 11-23
 - プロファイル, 15-34
- ユーザー定義演算子, 3-6
- ユーザー定義型, 2-36
 - Java クラスへのマップ, 15-10
 - 定義, 15-9
- ユーザー定義の集計ファンクション, 12-55
- ユーザー定義の統計情報
 - 削除, 15-72, 15-73, 15-86, 16-7, 16-15
- ユーザー定義ファンクション, 6-196

- 制限, 12-50
- 名前の優先順位, 6-197
- ネーミング規則, 6-198
- 優先順位
 - 演算子, 3-2
 - 条件, 5-4
- ユニバーサル ROWID
 - 「UROWID」を参照
- ユリウス暦, 2-20

よ

- 予約語, 2-106, C-1

ら

- ラージ・オブジェクト
 - 「LOB データ型」を参照
- ライセンス
 - 制限の変更, 9-62, 9-63
- ライブラリ
 - 再作成, 13-3
 - 作成, 13-2
 - システム権限の付与, 16-39
 - データベースからの削除, 15-77
- ライブラリ・ユニット
 - 「Java スキーマ・オブジェクト」を参照

り

- リカバリ
 - 使用中のメディアの実行, 8-24
 - 中断後のインスタンスの継続, 8-20
 - データの廃棄, 8-18
 - データベース, 8-11
 - パラレル化, 8-23
 - 分散、可能, 9-26
 - メディアの設計, 8-20
- リスト・パーティション化
 - 値の削除, 10-40
 - 値の追加, 10-40
 - パーティションの作成, 14-38
 - パーティションの追加, 10-46
- リスナー
 - 登録, 9-29
- リソース・パラメータ
 - CREATE PROFILE, 13-66

リソース・マネージャ, 9-28
リテラル
 SQL 構文内, 2-51
 SQL 文およびファンクション, 2-51
リレーショナル表
 作成, 14-7

る

累積分布, 6-43
ルーチン
 コール, 11-63
 実行, 11-63

れ

列
 LOB
 記憶域属性, 10-65, 13-11
 NULL の禁止, 11-80
 REF
 記述, 11-83
 VARRAY
 記憶域属性, 13-11
 値の制限, 11-72
 一意値, 11-79
 オブジェクト型
 マテリアライズド・ビューへの格納, 13-11
 親子関係, 12-39
 外部キーとしての指定, 11-82
 記憶域の変更, 10-64
 記憶域プロパティ, 14-31
 既存の制約の変更, 10-56
 コメントの作成, 11-66
 最大数, 14-20
 索引に基づく, 12-67
 修飾名, 7-3
 主キーとしての指定, 11-80
 制約の指定, 14-21
 置換可能な識別型, 6-153
 追加, 10-55
 定義, 14-6
 デフォルト値の指定, 14-21
 統計情報の関連付け, 11-48
 統計情報の収集, 11-37
 ネストした表
 記憶域属性, 13-11

表からの削除, 10-58
別名, 7-3
列 REF 制約, 11-74, 11-83
 ALTER TABLE, 10-56
 CREATE TABLE, 14-21
列制約, 11-73, 11-78
 ALTER TABLE, 10-56
 CREATE TABLE, 14-21
 制限, 10-58
レプリケーション
 行レベル依存の追跡, 12-8, 14-42
レベル
 ディメンションからの削除, 8-45
 ディメンションの追加, 8-45
 ディメンションの定義, 12-41
連結演算子, 3-4
連鎖行
 クラスタ, 11-36
 リスト, 11-42
レンジ・パーティション
 作成, 14-36
 追加, 10-45

ろ

ローカル管理の表領域
 記憶域属性, 17-50
 変更, 10-85
ローカル・パーティション索引, 12-74
ローカル・ユーザー, 13-73, 15-32
ロール
 AQ_ADMINISTRATOR_ROLE, 16-48
 AQ_USER_ROLE, 16-48
 CONNECT, 16-48
 DBA, 16-48
 DELETE_CATALOG_ROLE, 16-48
 EXECUTE_CATALOG_ROLE, 16-48
 EXP_FULL_DATABASE, 16-48
 HS_ADMIN_ROLE, 16-48
 IMP_FULL_DATABASE, 16-48
 RECOVERY_CATALOG_OWNER, 16-48
 RESOURCE, 16-48
 SNMPAGENT, 16-48
 アプリケーション, 8-114
 エンタープライズ・ディレクトリ・サービスを使用
 した識別, 13-73
 外部からの識別, 13-73

- 作成, 13-72
- 使用可能
 - 現在のセッション, 17-43, 17-44
- 使用禁止
 - 現在のセッション, 17-43, 17-45
- データベースからの削除, 15-91
- 取消し, 16-87
 - PUBLIC, 16-90
 - 別のロール, 15-91, 16-89
 - ユーザー, 15-91, 16-89
- 認可
 - 変更, 8-113
- 認証
 - エンタープライズ・ディレクトリ・サービス, 13-73
 - 外部サービス, 13-73
 - データベース, 13-73
 - パスワード, 13-73
 - パスワードによる識別, 13-73
 - パッケージを使用した識別, 13-73
- 付与, 16-31
 - PUBLIC, 16-34
 - システム権限, 16-41
 - 別のロール, 16-34
 - ユーザー, 16-34
 - ユーザー・セッションへの効果, 8-114
- ロールバック・セグメント
 - SQL 例, 13-78
 - オフライン化, 8-115, 8-116
 - オンライン化, 8-115, 8-116
 - 記憶域属性, 17-50
 - 記憶特性, 13-78
 - 記憶特性の変更, 8-115, 8-117
 - サイズの縮小, 8-115, 8-117
 - 最適なサイズの指定, 17-55
 - 作成, 13-75
 - システム生成, 13-75
 - データベースからの削除, 15-92
 - パブリック, 13-76
 - 表領域の指定, 13-77
 - 付与
 - システム権限, 16-42
- ロギング
 - REDO ログ・サイズ, 14-26
 - 最小限度の指定, 14-26
- ログ・グループ
 - 削除, 10-30

- ログ・データ
 - 更新操作中の収集, 8-31
 - 補助
 - 削除, 8-32
- ログ・ファイル
 - 削除, 8-29
 - セッション・パスの設定, 9-8
 - 追加, 8-29
 - データベースへの指定, 12-25
 - 登録, 8-35
 - 名前の変更, 8-29
 - 変更, 8-29
- ロック
 - 自動
 - 変更, 16-72
 - 「表ロック」を参照
- 論理条件, 5-8