Oracle9*i*

データベース管理者ガイド

リリース 2（9.2）

2002 年 7 月

部品番号：J06242-01

ORACLE®

Oracle9*i* データベース管理者ガイド , リリース 2（9.2）

部品番号：J06242-01

原本名：Oracle9*i* Database Administrator's Guide, Release 2 (9.2)

原本部品番号：A96522-01（Vol.1）、A96523-01（Vol.2）

原本著者：Ruth Baylis

原本協力者：Kathy Rich, Valarie Moore, Lance Ashdown, Allen Brumm, Michele Cyran, Mary Ann Davidson, Harvey Eneman, Amit Ganesh, Carolyn Gray, Wei Huang, Robert Jenkins, Mark Kennedy, Sushil Kumar, Bill Lee, Yunrui Li, Diana Lorentz, Sujatha Muthulingam, Gary Ngai, Waleed Ojeil, Lois Price, Ananth Raghavan, Ann Rhee, Rajiv Sinha, Jags Srinivasan, Anh-Tuan Tran, Deborah Steiner, Janet Stern, Michael Stewart, Alex Tsukerman, Kothanda Umamageswaran, Steven Wertheimer, Daniel Wong

Printed in Japan

目次

2 Oracle データベースの作成

3 Oracle Managed Files の使用

4 起動と停止

第 II 部　Oracle サーバー・プロセスと記憶域構造

5　Oracle プロセスの管理

6 制御ファイルの管理

7 オンライン REDO ログの管理

8 アーカイブ REDO ログの管理

9 LogMiner を使用した REDO ログの分析

10 ジョブ・キューの管理

11 表領域の管理

12 データ・ファイルの管理

13 UNDO 領域の管理

第 III 部　スキーマ・オブジェクト

14 スキーマ・オブジェクトの領域の管理

15 表の管理

16 索引の管理

17 パーティション表と索引の管理

18 クラスタの管理

19 ハッシュ・クラスタの管理

20 ビュー、順序およびシノニムの管理

21 スキーマ・オブジェクトの一般的な管理

22 データ・ブロック破損の検出と修復

第 IV 部 データベース・セキュリティ

23 セキュリティ・ポリシーの設定

24 ユーザーとリソースの管理

25 ユーザー権限とロールの管理

26 データベース使用の監査

第 V 部 データベース・リソースの管理

27 Database Resource Manager の使用

第 VI 部　分散データベースの管理

28　分散データベースの概念

29 分散データベースの管理

30 分散データベース・システムのアプリケーション開発

31 分散トランザクションの概念

32 分散トランザクションの管理

索引

はじめに

このマニュアルは、Oracle データベース・システムの運用を管理するユーザーを対象として記述しています。このようなユーザーはデータベース管理者（DBA）と呼ばれ、Oracle データベースの作成、円滑な運用、使用状況の監視などの役割を持ちます。

この項の内容は、次のとおりです。

- 対象読者
- 構成
- 関連文書
- 表記規則

注意：『Oracle9*i* データベース管理者ガイド』は、Oracle9*i* Standard Edition、Oracle9*i* Enterprise Edition および Oracle9*i* Personal Edition 製品の特長と機能について説明しています。これらの製品は同じ基本機能を備えています。ただし、最新の機能のいくつかは、Oracle9*i* Enterprise Edition または Oracle9*i* Personal Edition のみで使用可能であり、オプションのものもあります。たとえば、パーティション表やパーティション索引を作成するには、Oracle9*i* Enterprise Edition または Oracle9*i* Personal Edition が必要です。

Oracle9*i* の各エディション間の違い、および使用可能な機能とオプションの詳細は、『Oracle9*i* データベース新機能』を参照してください。

対象読者

このマニュアルの読者は、リレーショナル・データベースの概念をよく理解しているものと想定しています。また、Oracle を稼働しているオペレーティング・システム環境もよく理解している必要があります。

インストールとアップグレードに関する情報が必要な読者

管理者は、Oracle データベースのインストールや、既存の Oracle データベースの新しい形式へのアップグレード作業（たとえば、Oracle8 または Oracle8*i* データベースを Oracle9*i* 形式に移行する作業）にかかわることがあります。このマニュアルでは、インストールやシステムのアップグレードについては説明しません。

インストールに関する情報が必要な場合は、使用しているオペレーティング・システム固有の Oracle インストレーション・ガイドを参照してください。

データベースまたはアプリケーションのアップグレードに関する情報が必要な場合は、『Oracle9*i* データベース移行ガイド』を参照してください。

アプリケーションの設計に関する情報が必要な読者

このマニュアルには、管理者のみでなく、Oracle をよく理解しているユーザーと上級のデータベース・アプリケーション設計者にとっても有用な情報が記載されています。

ただし、データベース・アプリケーションの開発者は、『Oracle9*i* アプリケーション開発者ガイドー基礎編』と、Oracle データベース・アプリケーションの開発に使用するツールまたは言語製品のマニュアルも参照してください。

構成

このマニュアルは、次の部と章で構成されています。

第 I 部「基本データベース管理」

第 1 章「Oracle データベース管理者」

ソフトウェアのインストール、データベースの計画などのデータベース管理者が実行する一般的な作業の概要について説明します。

第 2 章「Oracle データベースの作成」

データベースを作成する際の考慮点とデータベースの作成手順について説明します。データベースの計画および作成を行うときに参照してください。

第 3 章「Oracle Managed Files の使用」

Oracle データベースで次のファイルを直接作成し、管理する方法について説明します。

- データ・ファイル
- 一時ファイル
- オンライン REDO ログ・ファイル
- 制御ファイル

第 4 章「起動と停止」

データベースの起動、可用性の変更または停止を行うときに参照してください。起動と停止に関連したパラメータ・ファイルについても説明しています。

第 II 部「Oracle サーバー・プロセスと記憶域構造」

第 5 章「Oracle プロセスの管理」

専用サーバー・プロセスや共有サーバー・プロセスなどの様々な Oracle プロセスの識別に役立ちます。プロセスの構成、変更、追跡および管理を行うときに参照してください。

第 6 章「制御ファイルの管理」

制御ファイルを管理するすべての作業（制御ファイルの命名、作成、トラブルシューティングおよび削除）について説明します。

第 7 章「オンライン REDO ログの管理」

オンライン REDO ログを管理するすべての作業（オンライン REDO ログ・ファイルの計画、作成、名前変更、削除および消去）について説明します。

第 8 章「アーカイブ REDO ログの管理」

アーカイブ・モードとアーカイブのチューニングについて説明します。

第 9 章「LogMiner を使用した REDO ログの分析」

LogMiner を使用して REDO ログ・ファイルを解析する方法について説明します。

第 10 章「ジョブ・キューの管理」

ジョブ・キューを使用する前に参照してください。ジョブ・キューの発行、削除、変更、調整に関するすべての作業について説明しています。

第 11 章「表領域の管理」

表領域の管理についてガイドラインを示し、表領域の作成、管理、変更、削除および表領域間のデータ移動の方法を説明します。

第 12 章「データ・ファイルの管理」

データ・ファイルの管理についてガイドラインを示し、データ・ファイルの作成、変更、名前変更およびデータ・ファイルに関する情報の表示の方法を説明します。

第 13 章「UNDO 領域の管理」

UNDO 表領域またはロールバック・セグメントを使用して UNDO 領域を管理する方法について説明します。

第 III 部「スキーマ・オブジェクト」

第 14 章「スキーマ・オブジェクトの領域の管理」

記憶域パラメータの設定、領域の割当て解除および管理などの一般的な作業について説明します。

第 15 章「表の管理」

一般的な表を管理するためのガイドライン、および表の作成、変更、メンテナンス、削除について説明します。

第 16 章「索引の管理」

作成、変更、監視、削除など、索引についての一般的なガイドラインを示します。

第 17 章「パーティション表と索引の管理」

パーティション表とパーティション索引、およびその作成および管理の方法について説明します。

第 18 章「クラスタの管理」

クラスタの作成、変更および削除などについての一般的なガイドラインを示します。

第 19 章「ハッシュ・クラスタの管理」

ハッシュ・クラスタの作成、変更および削除などについての一般的なガイドラインを示します。

第 20 章「ビュー、順序およびシノニムの管理」

ビュー、順序およびシノニムの管理に関するすべての作業について説明します。

第 21 章「スキーマ・オブジェクトの一般的な管理」

スキーマ管理に関する様々な事項について説明します。この章に記載されている操作は、特定のスキーマ・オブジェクトに固有のものではありません。オブジェクトの分析、表とクラスタの切捨て、データベース・トリガー、整合性制約およびオブジェクト依存性についての情報が必要なときに参照してください。

第 22 章「データ・ブロック破損の検出と修復」

データ・ブロックの破損を検出して修正する方法について説明します。

第 IV 部「データベース・セキュリティ」

第 23 章「セキュリティ・ポリシーの設定」

パスワードの管理に関連した特定の作業のみでなく、システム、データ、ユーザーの各セキュリティ・ポリシーなど、データベース・セキュリティに関するすべての作業について説明します。

第 24 章「ユーザーとリソースの管理」

セッション・ライセンスとユーザー・ライセンスおよびユーザー認証について説明し、ユーザーとリソースの管理に関連した特定の作業例を示します。

第 25 章「ユーザー権限とロールの管理」

ユーザー権限とロールの管理に関するすべての情報が含まれます。権限とロールの付与および取消しの方法について説明します。

第 26 章「データベース使用の監査」

監査情報の作成、管理および参照方法について説明します。

第 V 部「データベース・リソースの管理」

第 27 章「Database Resource Manager の使用」

Database Resource Manager を使用してリソースを割り当てる方法を説明します。

第 VI 部「分散データベースの管理」

第 28 章「分散データベースの概念」

Oracle の分散データベース・アーキテクチャの基本概念と用語について説明します。

第 29 章「分散データベースの管理」

分散データベース・システムの管理とメンテナンスの方法について説明します。

第 30 章「分散データベース・システムのアプリケーション開発」

分散データベース・システムで稼働するアプリケーションを開発するときの重要な考慮点について説明します。

第 31 章「分散トランザクションの概念」
分散トランザクションの概要とその整合性を維持する Oracle の仕組みについて説明します。

第 32 章「分散トランザクションの管理」
分散トランザクションの管理とトラブルシューティングの方法について説明します。

関連文書

詳細は、次の Oracle マニュアルを参照してください。

- 『Oracle9*i* データベース概要』

  『Oracle9*i* データベース概要』の第 1 章では、Oracle に関する概念と用語の概要について説明し、このマニュアルの詳細な情報の基礎を提供しています。この章は Oracle データベースを理解するための出発点であるため、『Oracle9*i* データベース管理者ガイド』を読む前に一読されることをお薦めします。『Oracle9*i* データベース概要』の他の章では、Oracle のアーキテクチャと各機能、および各機能の動作について詳しく説明しています。

- 『Oracle9*i* バックアップおよびリカバリ概要』

  このマニュアルは、バックアップとリカバリの概要について説明しています。

- 『Oracle9*i* ユーザー管理バックアップおよびリカバリ・ガイド』

  このマニュアルは、バックアップとリカバリの詳細について説明しています。データ・ファイル、制御ファイルおよびアーカイブ REDO ログのバックアップ、リストアおよびリカバリを行う際に使用します。

- 『Oracle9*i* Recovery Manager ユーザーズ・ガイド』

  このマニュアルは、Recovery Manager の詳細について説明しています。Recovery Manager は、バックアップとリカバリの操作を管理し、自動化するための Oracle です。

- 『Oracle9*i* データベース・パフォーマンス・プランニング』

  このマニュアルは、データベース・システムを設定する際に重要な考慮点について説明しており、データベースのチューニング方法を理解するのに役立ちます。主に概要、用語の定義、アーキテクチャおよび設計原理を取り扱い、事前および事後チューニング方法の概要を示しています。

- 『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』

  このマニュアルは、Oracle データベース・システムをチューニングする際のリファレンス・ガイドとして使用できます。

- 『Oracle9*i* アプリケーション開発者ガイドー基礎編』

  DBA が行う作業の多くはアプリケーション開発者と共通しています。場合によっては、アプリケーション・レベルのマニュアルに記載されている作業の説明が役立つことがあります。そのような場合は、このマニュアルが主なリファレンスになります。

このマニュアルの多数の例では、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・スキーマを使用しています。これらのスキーマがどのように作成されたかと、その使用方法の詳細は、『Oracle9*i* サンプル・スキーマ』を参照してください。

リリース・ノート、インストレーション・マニュアル、ホワイト・ペーパーまたはその他の関連文書は、OTN-J（Oracle Technology Network Japan）に接続すれば、無償でダウンロードできます。OTN-J を使用するには、オンラインでの登録が必要です。次の URL で登録できます。

`http://otn.oracle.co.jp/membership/`

OTN-J のユーザー名とパスワードを取得済みであれば、次の OTN-J Web サイトの文書セクションに直接接続できます。

`http://otn.oracle.co.jp/document/`

表記規則

このマニュアル・セットの本文とコード例に使用されている表記規則について説明します。

- 本文の表記規則
- コード例の表記規則
- Windows オペレーティング・システムの表記規則

本文の表記規則

本文中には、特別な用語が一目でわかるように様々な表記規則が使用されています。次の表は、本文の表記規則と使用例を示しています。

規則	意味	例
太字	太字は、本文中に定義されている用語または用語集に含まれている用語、あるいはその両方を示します。	この句を指定する場合は、**索引構成表**を作成します。

規則	意味	例
固定幅フォントの大文字	固定幅フォントの大文字は、システムにより指定される要素を示します。この要素には、パラメータ、権限、データ型、Recovery Manager 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、パッケージとメソッドの他、システム指定の列名、データベース・オブジェクトと構造体、ユーザー名、およびロールがあります。	この句は、NUMBER 列に対してのみ指定できます。 BACKUP コマンドを使用すると、データベースのバックアップを作成できます。 USER_TABLES データ・ディクショナリ・ビューの TABLE_NAME 列を問い合せます。 DBMS_STATS.GENERATE_STATS プロシージャを使用します。
固定幅フォントの小文字	固定幅フォントの小文字は、実行可能ファイル、ファイル名、ディレクトリ名およびサンプルのユーザー指定要素を示します。この要素には、コンピュータ名とデータベース名、ネット・サービス名、接続識別子の他、ユーザー指定のデータベース・オブジェクトと構造体、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があります。 **注意 :** 一部のプログラム要素には、大文字と小文字の両方が使用されます。この場合は、記載されているとおりに入力してください。	sqlplus と入力して、SQL*Plus をオープンします。 パスワードは orapwd ファイルに指定されています。 データ・ファイルと制御ファイルのバックアップを /disk1/oracle/dbs ディレクトリに作成します。 department_id、department_name および location_id の各列は、hr.departments 表にあります。 初期化パラメータ QUERY_REWRITE_ENABLED を true に設定します。 oe ユーザーで接続します。 これらのメソッドは JRepUtil クラスに実装されます。
固定幅フォントの小文字のイタリック	固定幅フォントの小文字のイタリックは、プレースホルダまたは変数を示します。	*parallel_clause* を指定できます。 U*old_release*.SQL を実行します。 *old_release* は、アップグレード以前にインストールしたリリースです。

コード例の表記規則

コード例は、SQL、PL/SQL、SQL*Plus またはその他のコマンドラインを示します。次のように、固定幅フォントで、通常の本文とは区別して記載されています。

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

次の表は、コード例の記載上の表記規則とその使用例を示しています。

規則	意味	例
[ ]	大カッコで囲まれている項目は、1 つ以上のオプション項目を示します。大カッコ自体は入力しないでください。	DECIMAL (*digits* [, *precision*])
{ }	中カッコで囲まれている項目は、そのうちの 1 つのみが必要であることを示します。中カッコ自体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複数の選択肢を区切るために使用します。オプションのうち 1 つを入力します。縦線自体は入力しないでください。	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
. . .	水平の省略記号は、次のどちらかを示します。	
	■ 例に直接関係のないコード部分が省略されていること。	CREATE TABLE ... AS *subquery*;
	■ コードの一部が繰返し可能であること。	SELECT *col1*, *col2*, ... , *coln* FROM employees;
. . .	垂直の省略記号は、例に直接関係のない数行のコードが省略されていることを示します。	SQL> SELECT NAME FROM V$DATAFILE; NAME ---------- /fsl/dbs/tbs_01.dbf /fs1/dbs/tbs_02.dbf . . . /fsl/dbs/tbs_09.dbf 9 rows selected.
その他の表記	大カッコ、中カッコ、縦線および省略記号以外の記号は、示されているとおりに入力してください。	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;
イタリック	イタリックの文字は、特定の値を指定する必要のあるプレースホルダまたは変数を示します。	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*
大文字	大文字は、システムにより指定される要素を示します。これらの用語は、ユーザー定義用語と区別するために大文字で記載されています。大カッコで囲まれている場合を除き、記載されているとおりの順序とスペルで入力してください。ただし、この種の用語は大 / 小文字区別がないため、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;

規則	意味	例

小文字	小文字は、ユーザー指定のプログラム要素を示します。たとえば、表名、列名またはファイル名を示します。
	注意 : 一部のプログラム要素には、大文字と小文字の両方が使用されます。この場合は、記載されているとおりに入力してください。
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Windows オペレーティング・システムの表記規則

次の表は、Windows オペレーティング・システムの表記規則と使用例を示しています。

規則	意味	例
「スタート」→を選択	プログラムの起動方法。	Database Configuration Assistant を起動するには、「スタート」→「プログラム」→「Oracle - *HOME_NAME*」→「Configuration and Migration Tools」→「Database Configuration Assistant」を選択します。
ファイル名とディレクトリ名	ファイル名とディレクトリ名では、大 / 小文字は区別されません。特殊文字のうち左山カッコ（<)、右山カッコ（>)、コロン（:)、二重引用符（")、スラッシュ（/)、パイプ（|）およびハイフン（-）は使用できません。特殊文字のうち円記号（¥）は、引用符で囲まれている場合にも要素のセパレータとして扱われます。ファイル名が¥¥で始まる場合、Windows では汎用命名規則を使用しているものとみなされます。	`c:¥winnt"¥"system32` は `C:¥WINNT¥SYSTEM32` と同じです。
`C:¥>`	現行のハード・ディスク・ドライブを示す Windows のコマンド・プロンプトを表します。コマンド・プロンプト内のエスケープ文字はカレット（^）です。プロンプトには、現在作業中のサブディレクトリが反映されます。このマニュアルでは、コマンド・プロンプトと呼んでいます。	`C:¥oracle¥oradata>`

特殊文字	特殊文字のうち円記号（¥）は、Windows コマンド・プロンプトで二重引用符（"）のエスケープ文字として必要な場合があります。カッコと一重引用符（'）には、エスケープ文字は不要です。エスケープ文字と特殊文字の詳細は、Windows オペレーティング・システムのマニュアルを参照してください。	`C:¥>exp scott/tiger TABLES=emp QUERY=¥"WHERE job='SALESMAN' and sal<1600¥"`
		`C:¥>imp SYSTEM/`*`password`*
		`FROMUSER=scott TABLES=(emp, dept)`

規則	意味	例
HOME_NAME	Oracle ホーム名を表します。ホーム名は、英数字で 16 文字以内です。ホーム名に使用できる特殊文字は、アンダースコアのみです。	`C:¥> net start Oracle`*HOME_NAME*`TNSListener`
ORACLE_HOME と *ORACLE_BASE*	Oracle8 リリース 8.0 以前では、Oracle コンポーネントをインストールすると、すべてのサブディレクトリはデフォルトで次のいずれかの名前のトップレベルの *ORACLE_HOME* ディレクトリに置かれていました。 ■ Windows NT の場合は `C:¥orant` ■ Windows 98 の場合は `C:¥orawin98` このリリースは、Optimal Flexible Architecture（OFA）のガイドラインに準拠しています。すべてのサブディレクトリがトップレベルの *ORACLE_HOME* ディレクトリにあるとはかぎりません。*ORACLE_BASE* というトップレベル・ディレクトリがあり、デフォルトでは `C:¥oracle` です。他の Oracle ソフトウェアがインストールされていないコンピュータに最新の Oracle リリースをインストールする場合、最初の Oracle ホーム・ディレクトリのデフォルト設定は `C:¥oracle¥ora`*nn* で、*nn* は最新リリース番号です。Oracle ホーム・ディレクトリは、*ORACLE_BASE* の直下にあります。 このマニュアルでは、すべてのディレクトリ・パスの例が、OFA の表記規則に従って示されています。	`%`*ORACLE_HOME*`%¥rdbms¥admin` ディレクトリにアクセスします。

Oracle9*i* の新機能

この章では、このマニュアルで解説されている Oracle9*i* リリース 2（9.2）の新しい管理機能について紹介し、追加情報の参照先を示します。

Oracle9*i* に新たに導入されたすべての機能の概要は、『Oracle9*i* データベース新機能』を参照してください。

次の項では、この『Oracle9*i* データベース管理者ガイド』に記載されている新機能について説明します。

- Oracle9i リリース 2（9.2）の新機能
- Oracle9i リリース 1（9.0.1）の新機能

Oracle9*i* リリース 2（9.2）の新機能

Oracle9*i* リリース 2（9.2）では、Oracle9*i* リリース 1（9.0.1）で達成された目標がさらに拡張され、詳細化されています。

このマニュアルに記述されている Oracle9*i* リリース 2（9.2）の新機能の概要は、次のとおりです。

- **データベース作成時の SYS と SYSTEM のパスワード指定**

  Oracle では、次の `CREATE DATABASE` 句を使用して、ユーザー `SYS` および `SYSTEM` のパスワードを指定できます。

  – `USER SYS IDENTIFIED BY` *password*

  – `USER SYSTEM IDENTIFIED BY` *password*

  関連項目： 2-23 ページ「データベースの保護 : ユーザー SYS および SYSTEM のパスワードの指定」

- **FORCE LOGGING モードの指定**

  `CREATE DATABASE`、`CREATE CONTROLFILE` および `CREATE TABLESPACE` 文の `FORCE LOGGING` 句を使用すると、DDL 文に `NOLOGGING` が指定されていても、REDO ログ・レコードを強制的に書き込ませることができます。

  関連項目：

  - 2-29 ページ「FORCE LOGGING モードの指定」
  - 6-7 ページ「CREATE CONTROLFILE 文」
  - 11-19 ページ「REDO レコードの書込みの制御」

- **Recovery Manager によるサーバー・パラメータ・ファイルのバックアップ**

  Recovery Manager を使用して、サーバー・パラメータ・ファイルのバックアップを作成できるようになりました。

  関連項目： 2-49 ページ「サーバー・パラメータ・ファイルのバックアップの作成」

- **リリース 2（9.2）の LogMiner の新機能**

  LogMiner 9.2 には複数の新機能のサポートが追加され、デフォルト動作が次のように変更されました。

  – リリース 2（9.2）以降の Oracle データベースで生成された REDO ログ用に、`LONG` および `LOB` データ型がサポートされます。

- サプリメンタル・ロギングは、デフォルトでオフになっています。これはリリース 1（9.0.1）からの変更点であり、リリース 1（9.0.1）ではデフォルトで最小限のサプリメンタル・ロギングがオンになっていました。リリース 2（9.2）では、必要なサプリメンタル・ロギング・レベルを指定する必要があります。
- サプリメンタル・ロギングは、データベース・レベルと表レベルで使用可能です。データベースのサプリメンタル・ロギングでは、最小限のロギングまたは識別キーのロギングを選択できます。表のサプリメンタル・ロギングでは、条件付きログ・グループと条件なしのログ・グループのどちらを使用するかを選択できます。

  関連項目： 9-2 ページ「サプリメンタル・ロギング」

- 戻されるデータの書式に影響する 2 つのオプションが追加されました。`DBMS_LOGMNR.NO_SQL_DELIMITER` オプションを使用すると、`SQL_REDO` 文と `SQL_UNDO` 文の末尾のセミコロンが抑制されます。`DBMS_LOGMNR.PRINT_PRETTY_SQL` オプションを使用すると、再構成された SQL 文が書式化されて読みやすくなります。

  関連項目： 9-17 ページ「戻されるデータの書式」

- 新しいオプション `DBMS_LOGMNR.CONTINUOUS_MINE` は LogMiner に対して、セッションの開始後にアーカイブされる REDO ログ・ファイルを自動的に追加して抽出するように指示します。

  関連項目： 9-26 ページ「連続的マイニング」

- `DBMS_LOGMNR.NO_DICT_RESET_ONSELECT` オプションを使用する必要がなくなりました。DDL の追跡が使用可能になっている場合、LogMiner では古いメタデータ定義が格納されるため、2 度目の選択操作では必要なメタデータのバージョンをすべて使用できます。
- 新しいプロシージャ `DBMS_LOGMNR_D.SET_TABLESPACE` では、LogMiner のすべての表はデフォルトである `SYSTEM` 以外の表領域に再作成されます。

  関連項目： 『Oracle9*i* 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』

■ **ローカル管理 SYSTEM 表領域の作成**

ローカル管理 `SYSTEM` 表領域を作成できるようになりました。そのためには、データベースの作成時に `CREATE DATABASE` 文の `EXTENT MANAGEMENT LOCAL` 句を指定する方法と、`DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL` プロシージャを使用して既存の `SYSTEM` 表領域をローカル管理表領域に移行する方法があります。

関連項目：

- 2-27 ページ「ローカル管理の SYSTEM 表領域の作成」
- 11-33 ページ「ローカル管理表領域への SYSTEM 表領域の移行」

- **ファイルと物理デバイスのマッピング**

  ホスト・ベースの論理ボリューム・マネージャ（LVM）と、RAID 機能を提供する洗練された記憶域サブシステムの導入により、ファイルからデバイスへのマッピングを判断するのが難しくなっています。そこで、ファイルを物理デバイスにマップできるように、新規のビューと新規の DBMS_STORAGE_MAP パッケージが作成されました。

  関連項目： 12-15 ページ「ファイルと物理デバイスのマッピング」

- **列名の変更**

  既存の表の列名を変更できるようになりました。

  関連項目： 15-14 ページ「表の列名の変更」

- **制約名の変更**

  表に対する既存の制約の名前を変更できるようになりました。

  関連項目： 21-19 ページ「制約名の変更」

- **デフォルトのリスト・パーティションの指定**

  リスト・パーティション表に定義するパーティションの値リスト記述子として、キーワード DEFAULT を使用できるようになりました。このパーティションは、パーティション・キー列が、そのパーティションの値リスト記述子に指定されているリテラル値と一致しない場合に、リスト・パーティション表に行を挿入するために使用されます。

  関連項目： 17-13 ページ「リスト・パーティション表の作成」

- **レンジ - リスト・コンポジット・パーティション化の指定**

  このリリースには、新しいタイプのコンポジット・パーティション化が導入されています。レンジ - リスト・パーティション化方法を使用して、表をパーティションに分割できるようになりました。この場合、パーティションはレンジ・パーティションとして定義され、サブパーティションはリスト・パーティションとして定義されます。

  関連項目： 17-8 ページ「レンジ - リスト・コンポジット・パーティション化方法を使用する場合」

- **SPLIT PARTITION および SPLIT SUBPARTITION 操作の最適化**

  特定の条件下では、Oracle で高速分割操作を実行できます。パーティションやサブパーティションの場合は、この操作の方が通常の分割操作よりも効率的です。

  関連項目： 17-57 ページ「SPLIT PARTITION および SPLIT SUBPARTITION 操作の最適化」

- **ユーザー SYS の監査**

  ユーザー SYS が行ったすべての操作を（すべての AS SYSDBA および AS SYSOPER 接続を含めて）監査できるようになりました。

  関連項目： 26-6 ページ「管理ユーザーの監査」

- **オブジェクト所有者にかわるオブジェクト権限の付与**

  新しいシステム権限 GRANT ANY OBJECT PRIVILEGE を使用すると、所有者にかわってオブジェクト権限を付与できます。この操作は、（ビューには）オブジェクト所有者が権限を付与したかのように表示されますが、監査レコードには実際に権限付与を行ったユーザーが表示されます。

  関連項目：

  - 25-14 ページ「オブジェクト所有者にかわるオブジェクト権限の付与」
  - 25-17 ページ「オブジェクト所有者にかわるオブジェクト権限の取消し」

- **LOB 列の制限の解除**

  自動セグメント領域管理を指定する表領域に、LOB 列を作成できるようになりました。

- **特定の分散データベース初期化パラメータの排除**

  以前のリリースの Oracle では、DISTRIBUTED_TRANSACTIONS 初期化パラメータを使用すると、データベースが同時に参加できる分散トランザクションの最大数を指定できました。このパラメータは排除され、同時分散トランザクションの数に制限がなくなりました。DISTRIBUTED_TRANSACTIONS 初期化パラメータは、指定しても無視されます。

  また、以前のリリースの Oracle では、各分散トランザクションの最大分岐数を MAX_TRANSACTION_BRANCHES 初期化パラメータで指定していました。このパラメータは Oracle8*i* で排除されましたが、分散トランザクションの最大分岐数は引き続き 32 に制限されていました。MAX_TRANSACTION_BRANCHES 初期化パラメータは、指定しても無視されます。

Oracle9*i* リリース 1（9.0.1）の新機能

Oracle9*i* は、Oracle データベースの新しいバージョンです。Oracle9*i* には、データベースの可用性を向上させる様々な機能が組み込まれています。オンラインで可能な操作が増えたことにより、オフラインでのメンテナンスの必要性が減少しました。データベース管理作業に必要な労力も少なくなりました。Oracle9*i* では、データベースが必要とする基本的なオペレーティング・システム・ファイルを自動的に作成および管理します。このバージョンは、自動管理のテーマに基づいています。

パフォーマンスが向上しました。Database Resource Manager には、リソースを細部に渡って制御できる新しいオプションが用意されています。これにより、リソース・コンシューマ・グループに要求されるパフォーマンス・レベルを適切に維持できます。パーティション化の拡張によって、パフォーマンスを向上させる表および索引のパーティション化が向上しました。セキュリティの拡張は、このバージョンの重要な要素です。アプリケーションにセキュリティと監査を実装する際に、これまで以上にきめの細かい方法を使用できます。

このマニュアルに記述されている Oracle9*i* の新機能の概要は、次のとおりです。

- **表のオンライン再定義**

  新しい PL/SQL パッケージ `DBMS_REDEFINITION` を使用すると、オンラインで表を再定義できます。オンラインで表を再定義する際、その再定義プロセス中ずっと、データ操作言語（DML）を使用してその表にアクセスできます。これにより、表を再定義するためにオフライン化する必要があった従来の方法に比べて、可用性が大幅に向上しました。

  関連項目： 15-16 ページ「表のオンライン再定義」

- **ANALYZE VALIDATE STRUCTURE 文の ONLINE オプション**

  解析するオブジェクト内で DML を実行している間も、`ANALYZE` 文によって妥当性をチェックできるようになりました。

  関連項目： 21-7 ページ「表、索引、クラスタおよびマテリアライズド・ビューの妥当性チェック」

- **アーカイブ・タイムラグの制御**

  現行のオンライン REDO ログ・グループの切替えに時間ベースの方法が導入されました。プライマリ / スタンバイ構成では、プライマリ・サイトのカレント以外のログがすべてアーカイブされて、スタンバイ・データベースに転送されますが、この機能を使用すると、時間で測定した場合のように、スタンバイ・データベースに適用されない REDO レコードの数を実質上制限できます。

  関連項目： 7-11 ページ「アーカイブ・タイムラグの制御」

- **データベースの一時停止**

  Oracle9*i* には、データベースの一時停止および再開機能があります。`ALTER SYSTEM SUSPEND` 文を使用すると、データ・ファイルおよび制御ファイルへの入出力（I/O）がすべて停止し、データベースが一時停止します。データベースを一時停止すると、実行中のすべての I/O 操作の完了が許可され、新しいデータベース・アクセスはキューに待機した状態になります。通常のデータベース操作を再開するには、`ALTER SYSTEM RESUME` 文を使用します。

  関連項目： 4-17 ページ「データベースの一時停止と再開」

- **データベースの静止**

  Oracle9*i* では、データベースを静止状態にすることができます。静止状態中は、DBA によるトランザクション、問合せまたは PL/SQL 文の実行のみが許可されています。データベースを静止状態にすると、それ以外の状態では安全に実行できない管理処理を実行できます。データベースを静止状態にするには、`ALTER SYSTEM QUIESCE RESTRICTED` 文を使用します。

  関連項目： 4-14 ページ「データベースの静止」

- **再開可能領域割当て**

  Oracle では、領域割当てが失敗した場合に大規模なデータベース処理を一時停止し、後で再開するための方法が提供されています。これにより、Oracle データベースがユーザーにエラーを返すかわりに、DBA が訂正処理を実行できます。エラー条件が訂正されると、一時停止していた処理が自動的に再開します。

  関連項目： 14-14 ページ「再開可能領域割当ての管理」

- **アーカイブ先の増加**

  オンライン REDO ログのアーカイブ先の最大数が 5 から 10 に増加しました。

  関連項目： 8-11 ページ「アーカイブ先の指定」

- **自動セグメント領域管理**

  ローカル管理表領域を使用すると、Oracle によってエクステントを自動的に管理できます。Oracle9*i* では、ローカル管理表領域に格納されたセグメントの空き領域と使用済み領域も自動的に管理できます。Oracle が使用するセグメント領域管理のタイプを指定するには、`CREATE TABLESPACE` 文の `SEGMENT SPACE MANAGEMENT` 句で `AUTO` または `MANUAL` を指定します。

  関連項目： 11-7 ページ「ローカル管理表領域のセグメント領域管理の指定」

- **パーティションのメンテナンス実行時のグローバル索引の更新**

  デフォルトでは、パーティション表のメンテナンス操作を実行すると、多くの場合グローバル索引が無効になります（UNUSABLE マークが設定されます）。このような場合は、グローバル索引全体を再作成する必要があります。また、パーティション化されている場合は、そのパーティションをすべて再作成する必要があります。Oracle9*i* では、このデフォルトの動作を無効にできます。メンテナンス操作で ALTER TABLE 文の UPDATE GLOBAL INDEX 句を指定すると、実表の操作と同時にグローバル索引が更新されます。

  関連項目： 17-22 ページ「パーティション表のメンテナンス」

- **複数のブロック・サイズ**

  複数のブロック・サイズがサポートされています。DB_BLOCK_SIZE 初期化パラメータによって指定される標準ブロック・サイズの他に、最大 4 つの非標準ブロック・サイズを指定できます。非標準ブロック・サイズは、表領域の作成時に指定します。標準ブロック・サイズは、SYSTEM 表領域およびその他の大部分の表領域で使用されます。複数ブロック・サイズのサポートにより、ブロック・サイズの異なる表領域をデータベース間でトランスポートできます。

  関連項目： 2-36 ページ「データベース・ブロック・サイズの指定」

- **動的バッファ・キャッシュ**

  システム・グローバル領域（SGA）のバッファ・キャッシュ・サブコンポーネントのサイズが動的になりました。DB_BLOCK_BUFFERS 初期化パラメータは、新しい動的パラメータ DB_CACHE_SIZE に置き換えられました。このパラメータでは、標準データベース・ブロック・サイズ用のバッファ・サブキャッシュのサイズを指定します。データベースで複数のブロック・サイズを指定した場合、バッファ・キャッシュはサブキャッシュで構成されます。最大 4 つの DB_*n*K_CACHE_SIZE 初期化パラメータを使用して、追加したブロック・サイズ用のバッファ・サブキャッシュのサイズを指定できます。

  関連項目： 2-38 ページ「SGA のサイズに影響する初期化パラメータの設定」

- **動的 SGA**

  SGA のサイズを制御する初期化パラメータを動的に指定できるようになりました。ALTER SYSTEM SET 文を使用して、SGA のサイズを動的に変更できます。

  関連項目： 2-38 ページ「SGA のサイズに影響する初期化パラメータの設定」

- **自動 UNDO 管理**

  Oracle ではこれまで、UNDO を格納するためにロールバック・セグメントを使用していました。UNDO とは、必要時にデータベースの変更をロールバックまたは取り消すために使用できる情報を指します。新しいバージョンでは、UNDO を格納するための UNDO 表領域を作成できます。UNDO 表領域を使用すると、ロールバック・セグメント領域の複雑な管理が不要になり、UNDO を上書きするまでの保存期間を制御できます。

  関連項目： 第 13 章「UNDO 領域の管理」

- **Oracle Managed Files**

  Oracle9*i* の Oracle Managed Files 機能を使用すると、Oracle データベースを構成するファイルを直接管理する必要がなくなります。`DB_CREATE_FILE_DEST` および `DB_CREATE_ONLINE_LOG_DEST_n` 初期化パラメータを使用して、表領域を構成するファイル、オンライン REDO ログ・ファイル、制御ファイルといった特定タイプのファイルに使用するファイル・システム・ディレクトリを指定します。これにより、一意の Oracle Managed Files が作成され、不要になると削除されます。

  関連項目： 第 3 章「Oracle Managed Files の使用」

- **データ・ファイルの自動削除**

  Oracle9*i* では、`DROP TABLESPACE` 文を使用して表領域を削除するときに、表領域のオペレーティング・システム・ファイル（データ・ファイル）を自動的に削除するオプションが用意されています。`ALTER DATABASE TEMPFILE` 文でも、同様のオプションによって、一時ファイルに対応するオペレーティング・システム・ファイルを削除できます。

  関連項目：

  - 11-28 ページ「表領域の削除」
  - 11-13 ページ「ローカル管理の一時表領域の変更」

- **メタデータ API**

  新しい PL/SQL パッケージ `DBMS_METADATA.GET_DDL` を使用すると、スキーマ・オブジェクトに関するメタデータを（オブジェクトの作成に使用する DDL の形式で）取得できます。

  関連項目： 21-30 ページ「PL/SQL パッケージを使用したスキーマ・オブジェクト情報の表示」

- **外部表**

  Oracle9*i* では、外部表のデータに読取り専用でアクセスできます。外部表はデータベース内に存在しない表として定義されており、アクセス・ドライバが提供されていればどのようなフォーマットにすることもできます。`CREATE TABLE ... ORGANIZATION EXTERNAL` 文を使用して、外部表を記述したメタデータを指定します。Oracle では現在、SQL*Loader 制御ファイル構文のサブセットであるデータ・マッピング機能を備えた `ORACLE_LOADER` アクセス・ドライバを提供しています。

  関連項目： 15-33 ページ「外部表の管理」

- **制約の拡張**

  `CREATE TABLE` または `ALTER TABLE` 文の `USING INDEX` 句が拡張され、一意キー制約または主キー制約を作成または使用可能にするときに、特定の索引を指定できるようになりました。また、制約を削除または使用禁止にするときに、一意キー制約または主キー制約を規定している索引が削除されないように指定できます。

  関連項目：

  - 16-12 ページ「制約に対応付けられた索引の作成」
  - 21-14 ページ「整合性制約の管理」

- **サーバー・パラメータ・ファイル**

  Oracle ではこれまで、テキスト形式の初期化パラメータ・ファイルに初期化パラメータを格納していました。Oracle9*i* からは、初期化パラメータをサーバー・パラメータ・ファイルに保持することを選択できます。サーバー・パラメータ・ファイルとは、データベース・サーバーに格納されたバイナリ形式のパラメータ・ファイルです。サーバー・パラメータ・ファイルに格納された初期化パラメータは永続的で、インスタンスの実行中に行ったパラメータの変更は、インスタンスを停止し、起動しても有効です。

  関連項目： 2-43 ページ「サーバー・パラメータ・ファイルを使用した初期化パラメータの管理」

- **デフォルトの一時表領域**

  `CREATE DATABASE` 文の新しい `DEFAULT TEMPORARY TABLESPACE` 句を使用すると、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。この表領域は、他の方法で一時表領域を割り当てられていないユーザー用のデフォルトの一時表領域として使用されます。

  関連項目： 2-25 ページ「デフォルト一時表領域の作成」

- **データベース・タイム・ゾーンの設定**

  CREATE DATABASE 文に SET TIME_ZONE 句が追加されました。この句を使用すると、データベースのタイム・ゾーンを UTC（協定世界時－旧称グリニッジ標準時）から変更できます。Oracle では、データがディスクに格納されるときに、すべての TIMESTAMP WITH LOCAL TIME ZONE データがデータベースのタイム・ゾーンに標準化されます。また、ALTER SESSION の SET 句に新しいセッション・パラメータ TIME_ZONE が追加されました。

  関連項目： 2-18 ページ「手順 6: CREATE DATABASE 文の発行」

- **トランザクションの命名**

  トランザクションへの名前の割当てが可能になりました。トランザクション名はインダウト分散トランザクションの解決に役立ち、COMMIT COMMENT の代替機能になります。

  関連項目： 32-3 ページ「トランザクションの命名」

- **Database Configuration Assistant の変更**

  Database Configuration Assistant の設計が変更されました。データベースの定義を保存したテンプレートが用意され、それを使用してデータベースを生成できます。Oracle が提供する事前定義のテンプレートを使用できます。また、独自のテンプレートを作成することもできます。テンプレートの作成方法には、既存のテンプレートを変更する、新しいテンプレートを定義する、既存データベースの定義を取得するなどがあります。

  Database Configuration Assistant でデータベースを作成すると、Oracle の新しいサンプル・スキーマを最初から組み込むことも、オプションとして後で追加することもできます。Oracle マニュアルで使用されているほとんどの例は、これらのスキーマに基づいています。

  関連項目： 2-6 ページ「Database Configuration Assistant の使用」

- **索引の使用状況の監視**

  ALTER INDEX 文に MONITORING USAGE 句が追加されました。これにより、索引を監視して、有効に使用されているかどうかを判断できます。

  関連項目： 16-21 ページ「索引の使用状況の監視」

- **リスト・パーティション化**

  リスト・パーティション化が導入されました。これを使用すると、各パーティションの記述にパーティション化列の離散値のリストを指定できます。リスト・パーティション化方法は、特に離散値に従ってデータ配分をモデル化するために設計されています。このモデル化は、レンジ・パーティション化またはハッシュ・パーティション化では困難です。

  関連項目： 第 17 章「パーティション表と索引の管理」

- **索引構成表のハッシュ・パーティション化**

  このバージョンでは、ハッシュ方法による索引構成表のパーティション化をサポートしています。これまでの索引構成表のパーティション化はレンジ方法のみ可能でした。

  関連項目： 17-19 ページ「パーティション化された索引構成表の作成」

- **動的ジョブ・キュー・プロセス**

  ジョブ・キュー・プロセスが動的に作成されるため、実行準備のできているジョブの実行に必要な数のプロセスのみが作成されます。ジョブ・キュー・コーディネータ・バックグラウンド・プロセス（CJQ）によって、ジョブを実行する J*nnn* プロセスが動的に起動されます。

  関連項目： 10-2 ページ「ジョブの実行に使用されるプロセスの有効化」

- **Oracle9*i* の Database Resource Manager の新機能**

  Database Resource Manager に、次の新機能が追加されました。

  – アクティブなセッション・プールの作成機能。このプールは、ユーザー・グループ内で同時にアクティブ化できる最大数のユーザー・セッションで構成されています。最大数を超える追加セッションは実行待ちのキューに送られますが、キューで待機しているジョブが終了するまでのタイムアウト周期を指定できます。

  – 管理者が定義した基準に基づいた、グループ間でのユーザーの自動切替え機能。特定のユーザー・グループのメンバーが、指定された時間より長時間実行されるセッションを作成した場合、そのセッションをリソース要件が異なる他のユーザー・グループに自動的に切り替えることができます。

  – ある操作の実行時間の見積りが事前定義の制限を超える場合に、その操作を実行しない機能。

  – UNDO プールの作成機能。このプールは、ユーザー・グループで消費可能な量の UNDO 領域から構成されます。

  関連項目： 第 27 章「Database Resource Manager の使用」

- **プロキシ認証および認可**

  Oracle9*i* では、クライアントのかわりに中間層サーバーが動作することを許可できます。この機能を指定するには、`ALTER USER` 文の `GRANT CONNECT THROUGH` 句を使用します。また、クライアントとして接続したときに、中間層でアクティブにできるロールも指定できます。

  関連項目： 24-16 ページ「プロキシ認証および認可」

- **アプリケーション・ロール**

  指定された PL/SQL パッケージを使用して、アプリケーション・ユーザーに付与されたロールを有効にするメカニズムが提供されます。この機能を使用するには、`CREATE ROLE` 文で `IDENTIFIED USING` *`package`* 句を使用します。

  関連項目： 25-9 ページ「アプリケーションによるロールの認可」

- **ファイングレイン監査**

  Oracle の従来の監査方法では、監査証跡に固定セットのファクトが記録されていました。監査オプションは、オブジェクトのアクセスまたは権限を監視する場合のみ設定できました。新しい PL/SQL パッケージ `DBMS_FGA` を使用すると、内容に基づいてデータ・アクセスを監査するファイングレイン監査をアプリケーションに実装できます。

  関連項目： 26-17 ページ「ファイングレイン監査」

- **リリース 1（9.0.1）の LogMiner の新機能**

  リリース 1（9.0.1）の LogMiner では、多くの新機能のサポートが追加されました。新機能の一部は、Oracle8 以上のデータベースの REDO ログ・ファイルに対応しています。その他の機能は、Oracle9*i* 以上で生成された REDO ログ・ファイルのみに対応しています。

  Oracle9*i* 以上で生成された REDO ログ・ファイル用の新機能

  LogMiner は、Oracle9*i* 以上で生成された REDO ログ・ファイルに対して次のサポートを提供します。

  - 索引クラスタ。
  - 連鎖行と移行行。
  - `ARCHIVELOG` モード使用時のダイレクト・パス挿入。
  - REDO ログ・ファイルへのデータ・ディクショナリの抽出。9-6 ページの「REDO ログへのディクショナリの抽出」を参照してください。
  - データ・ディクショナリとしてのオンライン・カタログの使用。9-8 ページの「オンライン・カタログの使用」を参照してください。

– すべてのデータ定義言語（DDL）操作の追跡。これにより、スキーマの変更を監視できます。9-9 ページの「DDL 文の追跡」を参照してください。

– ユーザーが実行した DDL の SQL_REDO 列への表示。オリジナルのデータベース・ユーザーに関する情報も返されます。

– update 用の主キー情報を含む SQL_REDO および SQL_UNDO の生成。つまり、更新された行が主キーと ROWID によって識別されるので（サプリメンタル・ロギングが使用可能な場合）、異なるデータベースへの SQL 文の適用が容易になります。

Oracle8 以上で生成された REDO ログ・ファイル用の新機能

LogMiner は、Oracle8 以上で生成された REDO ログ・ファイルに対して次のサポートを提供します。

– V$LOGMNR_CONTENTS のデータを、コミットされたトランザクションに属する行のみに限定するオプション。このオプションにより、ロールバックされたトランザクションと処理中のトランザクションを除外できます。オプションに関する情報は、9-26 ページの「LogMiner セッションの開始」を参照してください。

– REDO ログ・ファイル内の実際のデータ値に基づく問合せの実行。9-18 ページの「REDO ログからの実際のデータ値の抽出」を参照してください。

関連項目： 第 9 章「LogMiner を使用した REDO ログの分析」

第 I 部

基本データベース管理

第 I 部では、データベース管理者の概要、データベース作成、データベース・インスタンスの起動および停止方法について説明します。第 I 部の構成は、次のとおりです。

1

Oracle データベース管理者

この章では、Oracle データベースを管理するデータベース管理者（DBA）の役割について説明します。

この章の内容は、次のとおりです。

- Oracle ユーザーのタイプ
- DBA のタスク
- Oracle データベース・ソフトウェアのリリースの識別
- DBA のセキュリティと権限
- DBA の認証
- パスワード・ファイルの作成とメンテナンス
- DBA のユーティリティ

Oracle ユーザーのタイプ

ユーザーのタイプとその役割および責任は、サイトによって異なります。小規模のサイトでは、1 名の DBA を配置して、アプリケーション開発者およびユーザー向けのデータベースを管理することがありえます。大規模なサイトでは、DBA の役割を複数の人および複数の専門グループに分割する必要があります。

この項の内容は、次のとおりです。

- データベース管理者
- セキュリティ管理者
- ネットワーク管理者
- アプリケーション開発者
- アプリケーション管理者
- データベース・ユーザー

データベース管理者

各データベースには、データベースを管理する DBA が少なくとも 1 名は必要です。Oracle データベース・システムの場合は規模が大きく、多数のユーザーにより使用されるため、1 名の担当者ではデータベースを管理できない場合があります。このような場合は、DBA のグループを編成して、役割を分担させます。

DBA が担当するタスクは次のとおりです。

- Oracle データベースとアプリケーション・ツールをインストールおよびアップグレードします。
- データベース・システムにシステム記憶域を割り当て、将来の記憶域要件を計画します。
- アプリケーション開発者がアプリケーションを設計した後、プライマリ・データベースの記憶域構造（表領域）を作成します。
- アプリケーション開発者がアプリケーションを設計した後、プライマリ・オブジェクト（表、ビュー、索引）を作成します。
- アプリケーション開発者から得た情報に基づき、必要に応じてデータベース構造を修正します。
- ユーザーを登録し、システム・セキュリティをメンテナンスします。
- Oracle のライセンス契約に従っていることを確認します。
- データベースに対するユーザー・アクセスを制御し、監視します。
- データベースのパフォーマンスを監視し、最適化します。

- データベース情報のバックアップおよびリカバリの計画を立てます。
- テープ上のアーカイブ済みデータをメンテナンスします。
- データベースをバックアップおよびリストアします。
- 技術サポートについてオラクル社カスタマ・サポート・センターに連絡します。

セキュリティ管理者

サイトによっては、データベースに 1 名以上のセキュリティ管理者が必要です。セキュリティ管理者は、ユーザーの登録、データベースに対するユーザー・アクセスの制御と監視およびシステム・セキュリティのメンテナンスを実施します。したがって、サイトにセキュリティ管理者が別にいる場合、DBA はこれらの業務に対する役割を持ちません。

ネットワーク管理者

サイトによっては、1 名以上のネットワーク管理者がいる場合があります。ネットワーク管理者は、Oracle Net Services などの Oracle ネットワーク製品を管理できます。

> **関連項目：** 分散環境でのネットワーク管理の詳細は、第 VI 部「分散データベースの管理」を参照してください。

アプリケーション開発者

アプリケーション開発者は、データベース・アプリケーションを設計し、実装します。アプリケーション開発者が担当するタスクは、次のとおりです。

- データベース・アプリケーションを設計、開発します。
- アプリケーションのためのデータベース構造を設計します。
- アプリケーションに対する記憶域要件を見積ります。
- アプリケーションのためのデータベース構造の変更を指定します。
- DBA にこれらの情報を伝えます。
- 開発中にアプリケーションをチューニングします。
- 開発中にアプリケーションのセキュリティ手段を確立します。

アプリケーション開発者は、これらのタスクの一部を DBA と協力して実施する場合があります。

アプリケーション管理者

Oracle のサイトでは、特定のアプリケーションを管理するために 1 名以上のアプリケーション管理者が必要な場合があります。アプリケーションごとに専門の管理者を置く場合があります。

データベース・ユーザー

データベース・ユーザーは、アプリケーションまたはユーティリティを介してデータベースと対話します。一般ユーザーが担当するタスクは、次のとおりです。

- 許された範囲でデータを入力、修正および削除します。
- データのレポートを作成します。

DBA のタスク

次のタスクは、Oracle データベースを設計、実装およびメンテナンスするためのアプローチの優先順位を表しています。

タスク 1: データベース・ハードウェアの評価

タスク 2: Oracle ソフトウェアのインストール

タスク 3: データベースの計画

タスク 4: データベースの作成とオープン

タスク 5: データベースのバックアップ

タスク 6: システム・ユーザーの登録

タスク 7: データベース設計の実装

タスク 8: 実行データベースのバックアップ

タスク 9: データベースのパフォーマンス・チューニング

これらのタスクについて、次の項で説明します。

注意： 新しいリリースにアップグレードする場合は、既存の本番データベースのバックアップを作成してからインストールしてください。既存の本番データベースの保存方法に関する詳細は、『Oracle9*i* データベース移行ガイド』を参照してください。

タスク 1: データベース・ハードウェアの評価

Oracle とそのアプリケーションが、使用可能なコンピュータ・リソースを最大限に活用するための評価を行います。この評価では、次のような情報を明らかにする必要があります。

- Oracle とそのデータベースに使用可能なディスク・ドライブ数
- Oracle とそのデータベースに使用可能な専用テープ・ドライブ数（テープ・ドライブがある場合）
- Oracle インスタンスで使用可能なメモリーの量（システムの構成マニュアルを参照）

タスク 2: Oracle ソフトウェアのインストール

DBA は、Oracle データベースとすべてのフロントエンド・ツール、およびデータベースにアクセスするデータベース・アプリケーションをインストールします。分散処理環境インストールでは、データベースが中核となるコンピュータ（データベース）によって制御され、データベース・ツールとアプリケーションがリモート・マシン（クライアント）で実行される場合があります。このような場合には、Oracle を実行するコンピュータにリモート・マシンを接続するために必要な Oracle Net コンポーネントもインストールする必要があります。

インストールするソフトウェアの詳細は、1-8 ページの「Oracle データベース・ソフトウェアのリリースの識別」を参照してください。

関連項目： インストールの特定の要件や指示の詳細は、次のマニュアルを参照してください。

- 使用しているオペレーティング・システム固有の Oracle マニュアル
- フロントエンド・ツールおよび Oracle Net ドライバのインストレーション・ガイド

タスク 3: データベースの計画

DBA は、次のことを計画する必要があります。

- データベースの論理記憶域構造
- データベース全体の設計
- データベースのバックアップ計画

重要なことは、データベースの論理記憶域構造が、システムのパフォーマンスと様々なデータベース管理操作にどのように影響を及ぼすかについて計画することです。たとえば、表領域を構成するデータ・ファイルの数、各表領域に格納される情報のタイプ、およびデータ・ファイルの物理的な格納先ディスク・ドライブについて、表領域を作成する前に把握しておく必要があります。データベース構造の論理記憶域全体を計画する際は、実際にデータベースを作成し、稼働したときに、この構造によって生じる影響を考慮します。次の項目について、データベースの論理記憶域構造が及ぼす影響を考慮してください。

- Oracle を実行しているコンピュータのパフォーマンス
- データ・アクセス操作中のデータベースのパフォーマンス
- データベースのバックアップおよびリカバリの手順の効率

データベース・オブジェクトのリレーショナル設計と、各オブジェクトの記憶特性について計画します。オブジェクトを作成する前に、オブジェクトと各オブジェクトの物理記憶域間の関連について計画を立てることによって、1 つの単位としてのデータベースのパフォーマンスを直接制御できます。データベースの拡張計画についても必ず検討してください。

分散データベース環境では、この計画段階が非常に重要です。頻繁にアクセスされるデータの物理的な位置が、アプリケーションのパフォーマンスに大きな影響を及ぼします。

計画段階中に、データベースのバックアップ計画を作成します。バックアップの効率を改善するために、必要に応じて論理記憶域やデータベースの設計を変更します。

リレーショナル・データベース設計および分散データベース設計については、このマニュアルでは取り扱っていません。このような設計上の問題は、業界標準として認められている資料を参照してください。

データベースの論理記憶域構造、オブジェクトおよび整合性制約の作成方法は、第 II 部「Oracle サーバー・プロセスと記憶域構造」および第 III 部「スキーマ・オブジェクト」を参照してください。

タスク 4: データベースの作成とオープン

データベース設計が完了すると、データベースを作成し、オープンできます。データベースを作成するには、Database Configuration Assistant（DBCA）を使用してインストール時に作成する方法と、データベースを作成するためのスクリプトを用意する方法があります。

どちらの場合も、データベースの作成方法については第 2 章「Oracle データベースの作成」、データベース起動時のガイドラインについては第 4 章「起動と停止」を参照してください。

タスク 5: データベースのバックアップ

データベース構造の作成後、データベースに対して予定していたバックアップ計画を実行します。追加の REDO ログ・ファイルを作成し、データベース全体の最初のバックアップ（オンラインまたはオフライン）を作成して、その後の定期的なデータベース・バックアップをスケジュールします。

関連項目： バックアップ操作をカスタマイズする方法とリカバリ手順の実行方法の詳細は、次のいずれかを参照してください。

- 『Oracle9*i* ユーザー管理バックアップおよびリカバリ・ガイド』
- 『Oracle9*i* Recovery Manager ユーザーズ・ガイド』

タスク 6: システム・ユーザーの登録

データベース構造のバックアップが完了した後、Oracle のライセンス契約に従ってデータベースのユーザーを登録し、登録したユーザーに対して適切なロールを作成して付与できます。

これらの操作方法については、次の各章を参照してください。

- 第 23 章「セキュリティ・ポリシーの設定」
- 第 24 章「ユーザーとリソースの管理」
- 第 25 章「ユーザー権限とロールの管理」

タスク 7: データベース設計の実装

データベースを作成して起動し、システム・ユーザーを登録した後、必要なすべての表領域を作成して、計画したデータベースの論理構造を実装できます。このタスクを完了すると、データベースのオブジェクトを作成できます。

データベースの論理記憶域構造とオブジェクトの作成方法は、第 II 部「Oracle サーバー・プロセスと記憶域構造」および第 III 部「スキーマ・オブジェクト」を参照してください。

タスク 8: 実行データベースのバックアップ

この段階でデータベースはすべて実装されたので、再度バックアップを作成します。定期的にバックアップを作成するようにスケジュールし、また、データベース構造の変更を実装した直後にも必ずデータベースのバックアップを作成するようにします。

タスク 9: データベースのパフォーマンス・チューニング

データベースのパフォーマンスの最適化は、DBA の日常的な作業の 1 つです。Oracle では、DBA が各種ユーザー・グループへのリソースの割当てを制御できるように、データベース・リソース管理機能を提供しています。

Database Resource Manager については、第 27 章「Database Resource Manager の使用」を参照してください。

関連項目：『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』には、データベースとアプリケーションのチューニング情報が記載されています。

Oracle データベース・ソフトウェアのリリースの識別

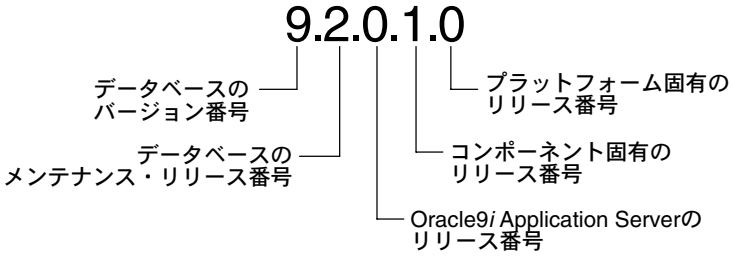
Oracle データベースは継続的に開発が進められ、メンテナンスが必要になるため、定期的にデータベースの新リリースが作成されています。新リリースを最初に利用したり、特定のメンテナンスを必要とするユーザーは一部なので、同時に複数の製品リリースが存在することになります。

特定のリリースを完全に識別するには、5 つの番号が必要です。ここでは、各番号の意味について説明します。

リリース番号の形式

Oracle で使用されているリリース・レベルの命名体系を理解するために、Oracle データベースのリリース番号「リリース 9.2.0.1.0」を例として説明します。

図 1-1 Oracle のリリース番号の例



注意： リリース 2（9.2）からは、Oracle のメンテナンス・リリースはリリース番号の 2 桁目の変更で示されます。以前のリリースでは、3 桁目が特定のメンテナンス・リリースを示していました。

データベースのバージョン番号

これは最も一般的な識別子です。この番号は、重要な新機能が含まれる、ソフトウェアの主な新規版（またはバージョン）を表します。

データベースのメンテナンス・リリース番号

この数字は、メンテナンス・リリース・レベルを表します。新機能がいくつか含まれている場合もあります。

Oracle9*i* Application Server のリリース番号

この数字には、Oracle9*i* Application Server（Oracle9*i*AS）のリリース・レベルが反映されます。

コンポーネント固有のリリース番号

この数字では、コンポーネント固有のリリース・レベルが識別されます。この桁の番号は、パッチ・セットや暫定リリースなどに応じてコンポーネントごとに異なる場合があります。

プラットフォーム固有のリリース番号

この数字では、プラットフォーム固有のリリースが識別されます。通常、これはパッチ・セットです。異なるプラットフォームに同等のパッチ・セットが必要な場合、この数字は影響を受けるプラットフォーム間で同じになります。

現行のリリース番号のチェック

現在インストールされている Oracle データベースのリリースを識別し、使用している他の Oracle コンポーネントのリリース・レベルを確認するには、データ・ディクショナリ・ビュー PRODUCT_COMPONENT_VERSION を問い合せます。問合せの例は、次のとおりです。他の製品のリリース・レベルも、データベースと関係なく表示される場合があります。

```
COL PRODUCT FORMAT A35
COL VERSION FORMAT A15
COL STATUS FORMAT A15
SELECT * FROM PRODUCT_COMPONENT_VERSION;

PRODUCT                             VERSION         STATUS
----------------------------------- --------------- ---------------
NLSRTL                              9.2.0.1.0       Production
Oracle9i Enterprise Edition         9.2.0.1.0       Production
PL/SQL                              9.2.0.1.0       Production
TNS for Solaris:                    9.2.0.1.0       Production
```

オラクル社にソフトウェアの問題を報告する際は、この問合せによって表示される情報を伝えることが重要です。

必要な場合は、V$VERSION ビューを問い合せてコンポーネント・レベルの情報を確認できます。

DBA のセキュリティと権限

Oracle データベース管理者の管理タスクを実施するには、データベースとデータベースが稼働するサーバーのオペレーティング・システムの両方で特別な権限が必要です。DBA アカウントへのアクセスは厳しく管理する必要があります。

この項の内容は、次のとおりです。

- DBA のオペレーティング・システム・アカウント
- DBA のユーザー名

DBA のオペレーティング・システム・アカウント

データベースに対する管理業務の多くを実行するには、オペレーティング・システム・コマンドを実行できる必要があります。Oracle が稼働するオペレーティング・システムによって異なりますが、オペレーティング・システムにアクセスするには、オペレーティング・システム・アカウント（ID）が必要になります。その場合、そのオペレーティング・システム・アカウントに対しては、多くのデータベース・ユーザーに必要な権限と比べて、より多くのオペレーティング・システム権限やアクセス権（Oracle ソフトウェアのインストールの実行など）が必要になります。Oracle ファイルを自分のアカウント内に格納する必要はありませんが、それらに対するアクセス権は必要です。

> **関連項目：** 使用しているオペレーティング・システム固有の Oracle マニュアルを参照してください。DBA のアカウントを識別する方法は、オペレーティング・システムによって異なります。

DBA のユーザー名

データベースとともに、次の 2 つのユーザー・アカウントが自動的に作成されます。

- SYS（デフォルトのパスワード : CHANGE_ON_INSTALL）
- SYSTEM（デフォルトのパスワード : MANAGER）

注意： SYS と SYSTEM のパスワードには、これらのデフォルト・パスワードを使用するのではなく、データベースの作成時に指定することをお薦めします。この操作については、2-23 ページの「データベースの保護 : ユーザー SYS および SYSTEM のパスワードの指定」を参照してください。

デフォルトのパスワードを使用する場合は、データ・ディクショナリ表への不当なアクセスや、データベースに対するその他の改ざんを防ぐために、Oracle データベースの作成直後に SYS および SYSTEM ユーザー名のパスワードを変更する必要があります。

日常的な管理タスクを実施するときに使用する管理者ユーザーを最低 1 つ追加で作成し、そのユーザーに DBA ロールを付与することをお薦めします。また、これらのユーザーには SYS および SYSTEM を使用しないでください。

セキュリティ拡張に関する注意： このリリース以上の Oracle 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確保するために、セキュリティ機能が拡張されています。

- Database Configuration Assistant（DBCA）を使用した初期インストール時に、SYS、SYSTEM、SCOTT、DBSNMP、OUTLN、AURORAJISUTILITY$、AURORA$ORB$UNAUTHENTICATED および OSE$HTTP$ADMIN を除くデフォルトのデータベース・ユーザー・アカウントはすべてロックされ、期限切れとなっています。ロックされたアカウントをアクティブにするには、DBA が手動でロックを解除して、新しいパスワードを再設定する必要があります。
- また、DBCA による初期インストール時に、SYS および SYSTEM に対してデフォルトのパスワードを割り当てるのではなく、ユーザーにパスワードの入力を求めます。この 2 つのユーザーのパスワードは、CREATE DATABASE 文を発行して手動で指定することもできます。

SYS

データベースの作成時に、ユーザー SYS が自動的に作成されて、DBA ロールが付与されます。

データベースのデータ・ディクショナリの実表とビューはすべて、SYS スキーマに格納されます。この実表とビューは、Oracle の操作に非常に重要なものです。データ・ディクショナリの整合性を維持するために、SYS スキーマ内の表は Oracle によってのみ操作されます。ユーザーや DBA はそれらの表を修正したり、ユーザー SYS のスキーマ内に表を作成しないでください。(ただし、必要に応じてデータ・ディクショナリ設定の記憶域パラメータを変更することは可能です)。

ほとんどのデータベース・ユーザーに対して、SYS アカウントによる接続を禁止する必要があります。

SYSTEM

データベースの作成時に、ユーザー SYSTEM が自動的に作成されて、DBA ロールが付与されます。

ユーザー名 SYSTEM を使用して、管理情報を表示する追加表とビュー、および各種の Oracle オプションと Oracle のツール製品が使用する内部表とビューが作成されます。個々のユーザーに関係する表は、SYSTEM スキーマ内に作成しないでください。

DBA ロール

事前に定義された DBA という名前のロールは、Oracle データベースで自動的に作成されます。このロールには、ほとんどのデータベース・システム権限が含まれています。これは非常に強力な権限であるため、専任の DBA にのみ付与してください。

注意： DBA ロールには、SYSDBA または SYSOPER システム権限が含まれていません。これらは、データベースとインスタンスの起動や停止など、基本的なデータベース管理タスクの実行を管理者に許可する特別な管理権限です。これらのシステム権限については、1-13 ページの「管理権限」を参照してください。

DBA の認証

DBA は、データベースの起動や停止などの特別な操作を実行します。これらの操作は DBA のみが実行する必要があるため、DBA のユーザー名には安全性の高い認証方式が必要です。

この項の内容は、次のとおりです。

- 管理権限
- 認証方式の選択
- オペレーティング・システム（OS）認証の使用
- パスワード・ファイル認証の使用

管理権限

管理者が基本的なデータベース操作を実行するために必要な管理権限は、SYSDBA および SYSOPER という 2 つの特別なシステム権限によって付与されます。必要な認可レベルに応じて、どちらか一方の権限を DBA に付与する必要があります。

注意： SYSDBA および SYSOPER システム権限を使用すると、データベースがオープンしていなくてもデータベース・インスタンスにアクセスできます。これらの権限の制御は、完全にデータベース機能の範囲外にあります。

システム権限と呼ばれる SYSDBA と SYSOPER も、他の方法では権限を付与できない特定のデータベース操作を実行するための接続の一種とみなすことができます（たとえば、CONNECT *AS* SYSDBA と指定する場合など）。

SYSDBA と SYSOPER

SYSDBA および SYSOPER システム権限で許可されている操作は、次のとおりです。

システム権限	許可されている操作
SYSDBA	■ STARTUP および SHUTDOWN 操作の実行 ■ ALTER DATABASE OPEN/MOUNT/BACKUP/CHARACTER SET ■ CREATE DATABASE ■ CREATE SPFILE ■ ARCHIVELOG と RECOVERY ■ RESTRICTED SESSION 権限を含む 実際は、このシステム権限を使用することにより、ユーザー SYS として接続できます。
SYSOPER	■ STARTUP および SHUTDOWN 操作の実行 ■ CREATE SPFILE ■ ALTER DATABASE OPEN/MOUNT/BACKUP ■ ARCHIVELOG と RECOVERY ■ RESTRICTED SESSION 権限を含む この権限を使用すると、ユーザー・データの表示を除く基本操作を実行できます。

これらの権限の使用を許可する方法は、選択した認証方式によって異なります。

SYSDBA または SYSOPER 権限で接続した場合は、一般的にユーザー名に対応付けられているスキーマではなく、デフォルトのスキーマで接続します。デフォルトのスキーマは、SYSDBA の場合は SYS、SYSOPER の場合は PUBLIC です。

管理権限での接続 : 例

この例では、ユーザーが SYSDBA システム権限で接続したときに、他のスキーマ（SYS）に割り当てられることを説明します。

ユーザー scott が次の文を発行したとします。

```
CONNECT scott/password
CREATE TABLE admin_test(name VARCHAR2(20));
```

後で、scott が次の文を発行します。

```
CONNECT scott/password AS SYSDBA
SELECT * FROM admin_test;
```

ユーザー scott は次のエラー・メッセージを受け取ります。

```
ORA-00942: 表またはビューが存在しません。
```

これは、scott が現在デフォルトで SYS スキーマを参照しているためです。表は scott スキーマ内に作成されています。

関連項目：

- 1-17 ページ「オペレーティング・システム（OS）認証の使用」
- 1-18 ページ「パスワード・ファイル認証の使用」

認証方式の選択

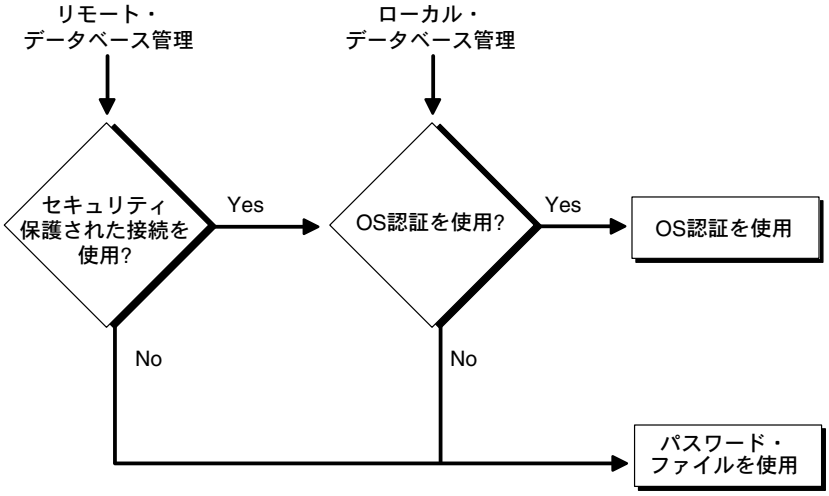
DBA の認証には、次の方式を利用できます。

- オペレーティング・システム（OS）認証
- パスワード・ファイル

注意： これらの認証方式は、旧バージョンの Oracle で提供されていた CONNECT INTERNAL 構文にかわるものです。CONNECT INTERNAL は使用できなくなりました。

データベースと同じマシン上でデータベースをローカルで管理するか、または単一のリモート・クライアントから複数の異なるデータベースを管理するかによって、どちらの認証方式を選択するかが決まります。図 1-2 は、DBA 用に選択できる認証方式を示しています。

図 1-2　DBA の認証方式



リモート・データベース管理の場合は、Oracle Net 関連マニュアルを参照して、セキュリティで保護された接続を使用しているかどうかを判断してください。TCP/IP や DECnet などの最も一般的な接続プロトコルは、セキュリティによって保護されていません。

関連項目：

- ユーザー認証の詳細は『Oracle9*i* データベース概要』を参照してください。
- 23-2 ページ「ユーザー認証」
- 24-9 ページ「ユーザー認証方式」
- 『Oracle9*i* Net Services 管理者ガイド』

セキュリティで保護されていないリモート接続

セキュリティで保護されていない接続で、権限を持つユーザーとして Oracle に接続するには、パスワード・ファイルによる認証を受ける必要があります。パスワード・ファイル認証を使用すると、SYSDBA または SYSOPER システム権限を付与されたデータベース・ユーザー名を追跡管理するために、パスワード・ファイルが使用されます。

この認証方式については、1-18 ページの「パスワード・ファイル認証の使用」を参照してください。

ローカル接続およびセキュリティで保護されたリモート接続

ローカル接続またはセキュリティで保護されたリモート接続で、権限を持つユーザーとして Oracle に接続するには、次のオプションがあります。

- データベースにパスワード・ファイルがあり、ユーザーが SYSDBA または SYSOPER システム権限を付与されている場合は、パスワード・ファイルを使用して接続し、認証を受けることができます。
- データベースがパスワード・ファイルを使用していない場合、あるいはユーザーが SYSDBA または SYSOPER 権限を付与されていないため、パスワード・ファイルに登録されていない場合は、OS 認証を使用できます。ほとんどのオペレーティング・システムでは、DBA の OS 認証のために、DBA の OS ユーザー名を特別なグループに登録します。このグループは一般に OSDBA と呼ばれます。

オペレーティング・システム（OS）認証の使用

ここでは、オペレーティング・システムを使用した管理者の認証方法について説明します。

OS 認証を使用するための準備

オペレーティング・システムを使用して管理ユーザーを認証するには、次の手順を実行する必要があります。

1. ユーザーのオペレーティング・システム・アカウントを作成します。
2. オペレーティング・システムで定義された OSDBA または OSOPER グループにユーザーを追加します。
3. REMOTE_LOGIN_PASSWORDFILE 初期化パラメータが NONE に設定されていることを確認します。これは、このパラメータのデフォルト値です。

OS 認証を使用した接続

次のどちらかの SQL*Plus コマンドを入力すると、ユーザーが管理ユーザーとして認証され、ローカル・データベースに接続できます。

```
CONNECT / AS SYSDBA
CONNECT / AS SYSOPER
```

安全な接続を介したリモート・データベース接続の場合、ユーザーはリモート・データベースのネット・サービス名も指定する必要があります。

```
CONNECT /@net_service_name AS SYSDBA
CONNECT /@net_service_name AS SYSOPER
```

関連項目： CONNECT コマンドの構文は、『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。

OSDBA と OSOPER

OS 認証を使用する場合は、2 つの特別なオペレーティング・システム・グループによって DBA の接続が制御されます。これらのグループは一般に OSDBA および OSOPER と呼ばれます。各グループはデータベースのインストール・プロセスで作成され、特定の名前が割り当てられます。グループの名前は、次の表に示すようにオペレーティング・システムによって異なります。

オペレーティング・システム・グループ	UNIX	Windows
OSDBA	dba	ORA_DBA
OSOPER	oper	ORA_OPER

Oracle Universal Installer で提示されるデフォルト名は変更できます。OSDBA および OSOPER グループは、オペレーティング・システム固有の方法で作成されます。

OSDBA または OSOPER グループでのメンバーシップは、Oracle への接続に次のように影響します。

- OSDBA グループに属するユーザーが AS SYSDBA を指定してデータベースに接続すると、SYSDBA システム権限が付与されます。
- OSOPER グループに属するユーザーが AS SYSOPER を指定してデータベースに接続すると、SYSOPER システム権限が付与されます。
- SYSDBA または SYSOPER システム権限に対応するオペレーティング・システム・グループに属していないユーザーが CONNECT コマンドを発行すると、エラーが発生します。

関連項目： OSDBA および OSOPER グループの作成方法の詳細は、使用しているオペレーティング・システム固有の Oracle マニュアルを参照してください。

パスワード・ファイル認証の使用

ここでは、パスワード・ファイル認証を使用した管理ユーザーの認証方法について説明します。

パスワード・ファイル認証を使用するための準備

パスワード・ファイル認証を使用して管理ユーザーを認証するには、次の手順を実行する必要があります。

1. ユーザーのオペレーティング・システム・アカウントを作成します。

2. 作成していない場合は、次のように ORAPWD ユーティリティを使用してパスワード・ファイルを作成します。

   ```
   ORAPWD FILE=filename PASSWORD=password ENTRIES=max_users
   ```

3. REMOTE_LOGIN_PASSWORDFILE 初期化パラメータを EXCLUSIVE に設定します。
4. ユーザー SYS（または管理権限を持つ他のユーザー）としてデータベースに接続します。
5. データベース内にユーザーが存在しない場合は、ユーザーを作成します。次のコマンドを入力して、ユーザーに SYSDBA または SYSOPER システム権限を付与します。

   ```
   GRANT SYSDBA to scott;
   ```

   この文を実行すると、パスワード・ファイルにユーザーが追加され、AS SYSDBA として接続できるようになります。

   関連項目： パスワード・ファイルの作成とメンテナンスの方法は、1-20 ページの「パスワード・ファイルの作成とメンテナンス」を参照してください。

パスワード・ファイル認証を使用した接続

SQL*Plus の CONNECT コマンドを使用すると、管理ユーザーが認証され、ローカルまたはリモート・データベースに接続できます。接続時には、ユーザー名とパスワード、および AS SYSDBA または AS SYSOPER 句を指定します。たとえば、ユーザー scott は SYSDBA 権限を付与されたので、次のコマンドで接続できます。

```
CONNECT scott/tiger AS SYSDBA
```

しかし、scott は SYSOPER 権限を付与されていないので、次のコマンドは失敗します。

```
CONNECT scott/tiger AS SYSOPER
```

注意： オペレーティング・システム認証は、パスワード・ファイル認証より優先されます。特に、オペレーティング・システムの OSDBA または OSOPER グループに属しているユーザーが、SYSDBA または SYSOPER で接続すると、指定したユーザー名とパスワードに関係なく、関連付けられた管理権限を使用して接続されます。

ユーザーが OSDBA グループにも OSOPER グループにも属しておらず、パスワード・ファイルにも指定されていない場合、接続は失敗します。

関連項目： CONNECT コマンドの構文は、『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。

パスワード・ファイルの作成とメンテナンス

パスワード・ファイル作成ユーティリティ ORAPWD を使用して、パスワード・ファイルを作成できます。一部のオペレーティング・システムでは、標準インストール時にこのファイルを作成できます。

この項の内容は、次のとおりです。

- ORAPWD の使用方法
- REMOTE_LOGIN_PASSWORDFILE の設定
- パスワード・ファイルへのユーザーの追加
- パスワード・ファイルのメンテナンス

関連項目： インストーラでパスワード・ファイルをインストールする方法の詳細は、オペレーティング・システム固有の Oracle マニュアルを参照してください。

ORAPWD の使用方法

パラメータを指定せずにパスワード・ファイル作成ユーティリティを起動すると、次の出力例のようなコマンドの正しい使用方法を示すメッセージが表示されます。

```
orapwd
Usage: orapwd file=<fname> password=<password> entries=<users>
where
file - name of password file (mand),
password - password for SYS (mand),
entries - maximum number of distinct DBAs and OPERs (opt),
There are no spaces around the equal-to (=) character.
```

次のコマンドを使用すると、acct.pwd という名前のパスワード・ファイルが作成されます。このパスワード・ファイルでは、30 人までの権限を持つユーザーと各ユーザー固有のパスワードを設定できます。この例のファイルは、最初は SYS で接続するユーザー用にパスワード secret を使用して作成されます。

```
ORAPWD FILE=acct.pwd PASSWORD=secret ENTRIES=30
```

次に、ORAPWD ユーティリティのパラメータについて説明します。

FILE

このパラメータは、作成するパスワード・ファイルの名前を設定します。ファイルにはフルパス名を指定する必要があります。このファイルの内容は暗号化されているため、ユーザーは直接読むことができません。これは必須パラメータです。

パスワード・ファイルで許可されているファイル名のタイプは、オペレーティング・システムによって異なります。一部のオペレーティング・システムでは、パスワード・ファイルを

特定の形式で作成して特定のディレクトリに格納する必要があります。また、環境変数を使用してパスワード・ファイルの名前と位置を指定できるオペレーティング・システムもあります。プラットフォームで許可されているファイルの名前と位置については、オペレーティング・システム固有の Oracle マニュアルを参照してください。

Oracle9*i* Real Application Clusters を使用して複数の Oracle インスタンスを稼働している場合は、各インスタンスの環境変数が同じパスワード・ファイルを指している必要があります。

注意： パスワード・ファイルやパスワード・ファイルの位置を特定する環境変数の保護は、システムのセキュリティにとって非常に重要です。パスワード・ファイルや環境変数にアクセスできるユーザーは、接続に対するセキュリティを脅かす潜在的な可能性を持っています。

PASSWORD

このパラメータは、ユーザー SYS 用のパスワードを設定します。データベースに接続した後、ALTER USER 文を発行して SYS のパスワードを変更すると、データ・ディクショナリに格納されているパスワードとパスワード・ファイルに格納されているパスワードの両方が更新されます。これは必須パラメータです。

ENTRIES

このパラメータは、パスワード・ファイルで受け入れるエントリの数を指定します。この数は、データベースに SYSDBA または SYSOPER として接続できるユーザーの数に対応します。ORAPWD ユーティリティでは、オペレーティング・システム・ブロックがいっぱいになるまでパスワード・エントリの割当てが継続されるため、実際にはユーザー数より多くのエントリを入力できます。たとえば、オペレーティング・システムのブロック・サイズが 512 バイトの場合は、4 つのパスワード・エントリが格納されます。そのため、割り当てられたパスワード・エントリの数は常に 4 の倍数になります。

パスワード・ファイルにユーザーを追加したり、パスワード・ファイルからユーザーを削除したりすると、エントリは再利用されます。REMOTE_LOGON_PASSWORDFILE=EXCLUSIVE を指定し、ユーザーに SYSDBA および SYSOPER 権限を付与する場合、このパラメータは必須です。

注意： 割り当てられたパスワード・エントリ数を超える場合は、新しいパスワード・ファイルを作成する必要があります。新しいパスワード・ファイルを作成しなくて済むように、必要と思われる数よりも大きいエントリ数を指定してください。

REMOTE_LOGIN_PASSWORDFILE の設定

パスワード・ファイルを作成する他に、初期化パラメータ REMOTE_LOGIN_PASSWORDFILE を適切な値に設定する必要があります。認識される値は、次のとおりです。

値	説明
NONE	このパラメータを NONE に設定すると、Oracle はパスワード・ファイルがない場合と同じように動作します。つまり、セキュリティで保護されていない接続では、権限付きの接続は許可されません。NONE は、このパラメータのデフォルト値です。
EXCLUSIVE	EXCLUSIVE パスワード・ファイルは、1 つのデータベースでしか使用できません。SYS 以外のユーザー名は、EXCLUSIVE ファイルのみに登録できます。EXCLUSIVE パスワード・ファイルを使用すると、個々のユーザーに SYSDBA および SYSOPER システム権限を付与し、それぞれの権限で接続させることができます。
SHARED	SHARED パスワード・ファイルは、複数のデータベースで使用できます。ただし、SHARED パスワード・ファイルで認識されるユーザーは SYS のみです。SHARED パスワード・ファイルにはユーザーを追加できません。SYSDBA または SYSOPER システム権限を必要とするすべてのユーザーは、必ず SYS ユーザーとそのパスワードを使用して接続してください。このオプションは、1 人の DBA が複数のデータベースを管理する場合に役立ちます。

提案： 最も高いレベルのセキュリティを実現するには、パスワード・ファイルの作成直後に REMOTE_LOGIN_PASSWORDFILE 初期化パラメータを EXCLUSIVE に設定してください。

パスワード・ファイルへのユーザーの追加

ユーザーに SYSDBA または SYSOPER 権限を付与すると、そのユーザーの名前と権限情報がパスワード・ファイルに追加されます。データベースに EXCLUSIVE パスワード・ファイルがない場合、つまり、初期化パラメータ REMOTE_LOGIN_PASSWORDFILE が NONE あるいは SHARED である場合は、これらの権限を付与しようとするとエラー・メッセージが出力されます。

ユーザーがこの 2 つの権限のうち 1 つでも持っている間は、そのユーザーの名前がパスワード・ファイルに残っています。これらの権限を両方とも取り消すと、そのユーザーはパスワード・ファイルから削除されます。

パスワード・ファイルを作成して新規ユーザーを追加する手順

1. 1-20 ページの「ORAPWD の使用方法」に記載された指示に従って、パスワード・ファイルを作成します。
2. REMOTE_LOGIN_PASSWORDFILE 初期化パラメータを EXCLUSIVE に設定します。
3. 次の例に示すように、SYSDBA 権限で接続します。

   ```
   CONNECT SYS/password AS SYSDBA
   ```

4. 必要であれば、インスタンスを起動してデータベースを作成します。または、既存のデータベースをマウントしてオープンします。
5. 必要に応じて、ユーザーを登録します。DBA 自身、および必要に応じて他のユーザーに SYSDBA または SYSOPER 権限を付与します。次の「SYSDBA および SYSOPER 権限の付与と取消し」を参照してください。

ユーザーに SYSDBA または SYSOPER 権限を付与すると、そのユーザーの名前がパスワード・ファイルに追加されます。これにより、ユーザーは（SYS ではなく）ユーザー名とパスワードを使用して SYSDBA または SYSOPER としてデータベースに接続できます。ユーザーが OS 認証の基準を満たしていれば、パスワード・ファイルを使用しても、OS 認証ユーザーの接続には問題ありません。

SYSDBA および SYSOPER 権限の付与と取消し

データベースで EXCLUSIVE パスワード・ファイルを使用している場合は、次の例に示すように、GRANT 文を使用して、ユーザーに SYSDBA または SYSOPER システム権限を付与します。

```
GRANT SYSDBA TO scott;
```

ユーザーの SYSDBA または SYSOPER システム権限を取り消すには、次のように REVOKE 文を使用します。

```
REVOKE SYSDBA FROM scott;
```

SYSDBA と SYSOPER は最も強力なデータベース権限であるため、ADMIN OPTION は使用されません。現在 SYSDBA として接続しているユーザーのみが、別のユーザーに SYSDBA または SYSOPER システム権限を付与できます。また、別のユーザーのシステム権限を取り消すこともできます。この 2 つの権限は、ロールには付与できません。ロールはデータベースを起動しないかぎり使用できないためです。SYSDBA と SYSOPER のデータベース権限とオペレーティング・システム・ロールを混同しないでください。これらはまったく別の機能です。

> **関連項目：** システム権限の詳細は、第 25 章「ユーザー権限とロールの管理」を参照してください。

パスワード・ファイル・メンバーの表示

データベースの SYSDBA または SYSOPER システム権限を付与されているユーザーを確認するには、V$PWFILE_USERS ビューを使用します。このビューで表示される列は、次のとおりです。

列	説明
USERNAME	パスワード・ファイルで認識されるユーザー名。
SYSDBA	この列の値が TRUE の場合、そのユーザーは SYSDBA システム権限でログインできます。
SYSOPER	この列の値が TRUE の場合、そのユーザーは SYSOPER システム権限でログインできます。

パスワード・ファイルのメンテナンス

この項の内容は、次のとおりです。

- パスワード・ファイルがいっぱいになった場合の、パスワード・ファイルのユーザー数の追加
- パスワード・ファイルの削除
- パスワード・ファイルの状態変更の回避

パスワード・ファイルのユーザー数の追加

ユーザーに SYSDBA または SYSOPER システム権限を付与しようとしたときに、パスワード・ファイルが満杯（ORA-01996）というエラーが出力された場合は、よりサイズの大きいパスワード・ファイルを作成し、ユーザーに再度権限を付与する必要があります。

パスワード・ファイルを置換する手順

1. V$PWFILE_USERS ビューを問い合せて、どのユーザーが SYSDBA または SYSOPER 権限を持っているかを確認します。
2. データベースを停止します。
3. 既存のパスワード・ファイルを削除します。
4. 1-20 ページの「ORAPWD の使用方法」の指示に従って、ORAPWD ユーティリティを使用してパスワード・ファイルを新しく作成します。ENTRIES パラメータには、必要と思われる数よりも大きい数を指定してください。
5. 1-22 ページの「パスワード・ファイルへのユーザーの追加」の指示に従います。

パスワード・ファイルの削除

ユーザー認証にパスワード・ファイルを使用する必要がなくなったと判断した場合は、パスワード・ファイルを削除して、REMOTE_LOGIN_PASSWORDFILE 初期化パラメータを NONE にリセットできます。このファイルを削除した後は、オペレーティング・システムによって認証されたユーザーのみがデータベース管理の操作を実行できます。

注意： REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE（または SHARED）を使用してデータベースまたはインスタンスをマウントしている場合は、パスワード・ファイルを削除または変更しないでください。パスワード・ファイルを削除または変更すると、パスワード・ファイルを使用してリモートから再接続できなくなります。かわりのパスワード・ファイルを作成しても、タイムスタンプとチェックサムが正しくないため、新しいパスワード・ファイルは使用できません。

パスワード・ファイルの状態変更

パスワード・ファイルの状態は、パスワード・ファイルに格納されます。パスワード・ファイルを初めて作成したときのデフォルトの状態は、SHARED です。パスワード・ファイルの状態を変更するには、初期化パラメータ REMOTE_LOGIN_PASSWORDFILE を設定します。インスタンスを起動すると、クライアント・マシンに格納された初期化パラメータ・ファイルからこのパラメータの値が検索されます。データベースをマウントすると、このパラメータの値とパスワード・ファイルに格納された値が比較されます。これらの値が一致しない場合は、パスワード・ファイルの値が上書きされます。

注意： EXCLUSIVE パスワード・ファイルを誤って SHARED に変更しないように注意してください。複数のクライアントからインスタンスを起動できるようにするには、各クライアントに初期化パラメータ・ファイルが存在し、各クライアントのファイルにある REMOTE_LOGIN_PASSWORDFILE パラメータの値が一致する必要があります。パラメータの値が異なるクライアントがある場合は、インスタンスを起動した場所によって、パスワード・ファイルの状態が変わります。

DBA のユーティリティ

Oracle データベースには、データのメンテナンスに役立つ様々なユーティリティが用意されています。ここでは、それらのユーティリティの中から次の 2 つを紹介します。

- SQL*Loader
- エクスポートとインポート

関連項目：『Oracle9*i* データベース・ユーティリティ』

SQL*Loader

SQL*Loader は、DBA と Oracle の他のユーザーにより使用されます。SQL*Loader は、オペレーティング・システムの標準ファイル（テキストや C データ形式のファイルなど）から Oracle データベース表にデータをロードします。

エクスポートとインポート

エクスポート・ユーティリティとインポート・ユーティリティを使用すると、Oracle データベースの間で Oracle 形式の既存データを移動できます。たとえば、エクスポート・ファイルを使用すると、データベースのデータをアーカイブしたり、同一のオペレーティング・システムまたは異なるオペレーティング・システム上で稼働している複数の Oracle データベース間でデータを移動できます。

2

Oracle データベースの作成

この章では、Oracle データベースの作成プロセスについて説明します。この章の内容は、次のとおりです。

- データベースを作成する前の考慮点
- Database Configuration Assistant の使用
- Oracle データベースの手動作成
- CREATE DATABASE 文の理解
- データベース作成のトラブルシューティング
- データベースの削除
- データベースを作成した後の考慮点
- 初期化パラメータとデータベースの作成
- サーバー・パラメータ・ファイルを使用した初期化パラメータの管理

関連項目：

- 基礎となるオペレーティング・システム・ファイルが自動的に作成され、Oracle データベースによって管理されるデータベースの作成方法の詳細は、第 3 章「Oracle Managed Files の使用」を参照してください。
- Oracle Real Application Clusters 環境に固有の追加情報は、『Oracle9*i* Real Application Clusters セットアップおよび構成 』を参照してください。

データベースを作成する前の考慮点

データベースの作成では、いくつかのオペレーティング・システム・ファイルを準備することで、それらのファイルを Oracle データベースとして動作できるようにします。データベースは、データ・ファイルの数やアクセスするインスタンスの数にかかわらず、一度だけ作成します。データベースの作成では、既存のデータベース内の情報を消去し、同じ名前と物理構造を持つ新規データベースを作成することもできます。

この項の内容は、次のとおりです。

- データベース作成計画
- 作成の前提条件
- Oracle データベースの作成方法の決定

データベース作成計画

データベースの作成を準備するには、調査と綿密な計画が必要です。推奨される処理を次に示します。

処理	詳細
■ データベース表と索引を計画し、それらが必要とする領域を見積ります。	第 II 部「Oracle サーバー・プロセスと記憶域構造」 第 III 部「スキーマ・オブジェクト」
■ データベースを構成する基礎となるオペレーティング・システム・ファイルのレイアウトを計画します。ファイルを適切に分散すると、ファイル・アクセスの I/O が分散されるため、データベースのパフォーマンスを大幅に向上できます。Oracle をインストールしてデータベースを作成するときに I/O を分散させるには、いくつかの方法があります。たとえば、REDO ログ・ファイルを異なるディスクに配置する、ストライプ化する、競合が少なくなるようにデータ・ファイルを配置する、データの密度（データ・ブロック当たりの行数）を制御するなどです。	『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』 オペレーティング・システム固有の Oracle マニュアル
■ Oracle Managed Files 機能の使用を検討します。Oracle Managed Files 機能を使用すると、データベース記憶域を構成するオペレーティング・システム・ファイルが作成および管理されるため、各ファイルの管理が容易になります。	第 3 章「Oracle Managed Files の使用」
■ **グローバル・データベース名**を選択します。グローバル・データベース名とは、ネットワーク構造内におけるデータベースの名前と位置です。グローバル・データベース名を作成するには、`DB_NAME` と `DB_DOMAIN` の 2 つの初期化パラメータを設定します。	「グローバル・データベース名の決定」2-35 ページ

処理	詳細
■ 初期化パラメータ・ファイルを構成する初期化パラメータについて理解します。特に**サーバー・パラメータ・ファイル**の概要と操作について理解します。サーバー・パラメータ・ファイルを使用すると、初期化パラメータをサーバー側のディスク・ファイルに永続的に格納し、管理できます。	「初期化パラメータとデータベースの作成」2-34 ページ 「サーバー・パラメータ・ファイルの概要」2-43 ページ 『Oracle9*i* データベース・リファレンス』
■ データベース・キャラクタ・セットを選択します。 データ・ディクショナリ内のデータも含め、すべての文字データは、そのデータベースのキャラクタ・セットで格納されます。データベースの作成時には、データベース・キャラクタ・セットを指定する必要があります。 異なるキャラクタ・セットを使用したクライアントがデータベースにアクセスする場合は、クライアント・キャラクタ・セットをすべて含むスーパーセットを選択します。そうしないと、キャラクタの変換が必要な場合、オーバーヘッドが増大し、データが消失する危険性が大きくなります。 代替キャラクタ・セットを指定することもできます。	『Oracle9*i* Database グローバリゼーション・サポート・ガイド』
■ データベースでサポートする必要のあるタイム・ゾーンを検討します。 Oracle では、Oracle ホーム・ディレクトリにあるタイム・ゾーン・ファイルが、有効なタイム・ゾーンのソースとして使用されます。デフォルトのタイム・ゾーン・ファイル（`timezone.dat`）にはなく、定義数の多いタイム・ゾーン・ファイル（`timezlrg.dat`）に存在するタイム・ゾーンを使用する必要があると判断した場合は、後者のファイルを指すように環境変数 `ORA_TZFILE` を設定する必要があります。	「データベースのタイム・ゾーン・ファイルの指定」2-28 ページ 『Oracle9*i* Database グローバリゼーション・サポート・ガイド』
■ データベースの標準ブロック・サイズを選択します。このサイズはデータベースの作成時に `DB_BLOCK_SIZE` 初期化パラメータによって指定しますが、データベースの作成後は変更できません。 標準ブロック・サイズは、`SYSTEM` 表領域およびその他の大部分の表領域で使用されます。また、表領域の作成時に、最大 4 つの非標準ブロック・サイズを指定できます。	「データベース・ブロック・サイズの指定」2-36 ページ
■ UNDO レコードを管理するために、ロールバック・セグメントではなく、UNDO 表領域を使用します。	第 13 章「UNDO 領域の管理」

処理	詳細
■ バックアップおよびリカバリ計画を作成し、データベースを障害から保護します。重要な点として、多重化による制御ファイルの保護、適切なバックアップ・モードの選択、およびオンライン REDO ログとアーカイブ REDO ログの管理があげられます。	第 7 章「オンライン REDO ログの管理」 第 8 章「アーカイブ REDO ログの管理」 第 6 章「制御ファイルの管理」 『Oracle9*i* バックアップおよびリカバリ概要』
■ インスタンスの起動と停止、データベースのマウントとオープンの原理およびオプションについて理解します。	第 4 章「起動と停止」

作成の前提条件

新しいデータベースを作成するには、次の前提条件を満たす必要があります。

- 必要な Oracle ソフトウェアがインストールされていること。これには、オペレーティング・システム固有の各環境変数の設定と、ソフトウェアおよびデータベース・ファイルのディレクトリ構造の設定が含まれます。
- データベース管理者（DBA）に必要な操作を実行できるオペレーティング・システム権限があること。DBA は、データベースを作成またはオープンする前に、オペレーティング・システムまたはパスワード・ファイルで特別な認証を受ける必要があります。これにより、インスタンスを起動および停止できるようになります。この認証については、1-13 ページの「DBA の認証」を参照してください。
- Oracle インスタンスを起動するために十分なメモリーが使用可能であること。
- Oracle を実行するコンピュータ上に、設計したデータベースのための十分なディスク記憶域があること。

各前提条件については、オペレーティング・システム固有の Oracle インストレーション・ガイドを参照してください。また、Oracle Universal Installer を使用すると、表示される手順に従ってインストールでき、環境変数、ディレクトリ構造および認可の設定に関するヘルプが表示されます。

Oracle データベースの作成方法の決定

データベース作成では、次のような操作を実行します。

- データ・ディクショナリなど、Oracle がデータベースにアクセスし、使用するために必要となる情報の構造を作成します。
- データベースの制御ファイルと REDO ログ・ファイルを作成して初期化します。
- 新しいデータ・ファイルを作成するか、既存のデータ・ファイル内のデータを消去します。

これらの操作には `CREATE DATABASE` 文を使用しますが、実行可能なデータベースを作成する前に他の処理が必要です。この処理には、ユーザー表領域と一時表領域の作成、データ・ディクショナリ表のビューの作成、Oracle 組込みパッケージのインストールなどがあります。これは、データベース作成プロセスで、準備されたスクリプトを実行する必要があるためです。ただし、このスクリプトをユーザーが準備する必要はありません。

新規 Oracle データベースの作成には、次の方法があります。

- Database Configuration Assistant（DBCA）の使用

  DBCA は、選択したインストール・タイプに応じて Oracle Universal Installer から起動でき、GUI を使用して、データベースの作成プロセスを説明しています。DBCA を使用しない選択もできます。データベースを作成する場合は、スタンドアロン・ツールとしていつでも DBCA を起動できます。2-6 ページの「Database Configuration Assistant の使用」を参照してください。

- スクリプトによるデータベースの手動作成

  データベース作成用のスクリプトがすでにある場合は、従来どおり手動でデータベースを作成できます。ただし、Oracle の新機能を利用するために、既存のスクリプトの編集を検討してください。データベース・ソフトウェア・ファイルとともにサンプル・データベース作成スクリプトとサンプル初期化パラメータ・ファイルが配布されており、どちらもニーズにあわせて編集できます。2-14 ページの「Oracle データベースの手動作成」を参照してください。

- 既存データベースのアップグレード

  旧リリースの Oracle を使用している場合、データベースの作成が必要になるのは、まったく新しく作成するときのみです。既存の Oracle データベースをアップグレードすると、新しいリリースの Oracle で使用できます。データベースのアップグレードについては、このマニュアルでは説明しません。既存の Oracle データベースのアップグレードについては、『Oracle9*i* データベース移行ガイド』を参照してください。

Database Configuration Assistant の使用

Database Configuration Assistant（DBCA）はオラクル社が提供するツールです。DBCA を使用すると、Oracle データベースの作成、既存の Oracle データベースのデータベース・オプションの構成、Oracle データベースの削除またはデータベース・テンプレートの管理ができます。DBCA は Oracle Universal Installer により自動的に起動されますが、スタンドアロンとして起動することもできます。そのためには、Windows オペレーティング・システムの場合は「スタート」メニューの「Configuration and Migration Tools」から選択し、UNIX の場合はコマンドラインから次のように入力します。

```
dbca
```

DBCA には、次の 3 つの実行モードがあります。

モード	説明
Interactive	これは、何もパラメータを指定しない場合のデフォルト・モードです。このモードでは、ウィザードのような GUI インタフェースが表示され、DBCA の機能をすべて使用できます。オンライン・ヘルプが用意されています。
Progress Only	通常、このモードは他のデータベース作成ツールで使用されます。たとえば、Oracle Universal Installer、Enterprise Manager Configuration Assistant および Oracle Internet Directory Configuration Assistant で使用されます。このモードはデータベースとテンプレートの作成時に使用され、プログレス・バーのみが表示されます。
Silent	このモードは、パラメータを指定するコマンドライン・インタフェースのみで構成されます。他のユーザー・インタフェースはありません。情報、エラーおよび警告の各メッセージがログ・ファイルに書き込まれます。データベースのカスタマイズまたは作成に使用するテンプレートを選択して指定します。

DBCA では、シングル・インスタンス・データベースの作成や、Oracle Real Application Clusters 環境でのインスタンスの作成または追加が可能です。

この項では、DBCA をインタラクティブ・モードで使用する方法に重点を置いて説明します。この章の内容は、次のとおりです。

- DBCA を使用する利点
- DBCA を使用したデータベースの作成
- データベース・オプションの構成
- DBCA を使用したデータベースの削除
- DBCA テンプレートの管理
- DBCA のサイレント・モードの使用

DBCA を使用する利点

DBCA を使用すると、次のような利点があります。

- ウィザードのガイドに従ってオプションを選択できるため、データベースを容易に作成しカスタマイズできます。これにより、様々な詳細レベルを指定できます。必要最低限の設定を入力し、残りの設定を Oracle によって自動的に決定できるので、最適なパラメータ設定やデータベース構造を決定するために時間をかける必要はありません。必要に応じて、パラメータ設定やファイルの割当てを細かく指定することもできます。
- Oracle の新機能を利用した効果的なデータベースが作成されます。
- Optimal Flexible Architecture（OFA）の採用により、初期化ファイルなどのデータベース・ファイルや管理ファイルを標準的な方法に従って命名し、配置できます。

DBCA を使用したデータベースの作成

DBCA を使用すると、Oracle が提供する事前定義のテンプレート、または独自に作成したテンプレートを使用して、データベースを作成できます。テンプレートは、データベースの記述です。テンプレートの詳細は、2-9 ページの「DBCA テンプレートの管理」を参照してください。

テンプレートの選択

DBCA には、DBCA 製品に付属するテンプレートなど、使用可能なテンプレートが表示されます。これらのテンプレートについては、2-12 ページの「オラクル社が提供する DBCA テンプレート」を参照してください。独自に作成したテンプレートがある場合は、それも表示されます。作成するデータベースに適したテンプレートを選択してください。「Show Details...」ボタンをクリックすると、テンプレートで定義されたデータベースについて、次のような特定の情報が表示されます。

データ・ファイルが含まれるかどうか

テンプレートを選択すると、データベース定義にデータ・ファイルが含まれるかどうかが指定されます。これにより、データベースの作成にシード・テンプレート（データ・ファイルを含む）を使用するか、または非シード・テンプレート（データ・ファイルなし）を使用するかが判断されます。

グローバル・データベース名とデータベース・オプションの指定

DBCA の次のページでは、グローバル・データベース名と SID を指定できます。

データベース・オプションの指定

「Database Features」ページは、非シード・テンプレートを選択した場合にのみ表示されます。このページでは、データベース・オプションを組み込むことができます。

次のリストは、データベースにインストールできる代表的な Oracle オプションを示しています。選択したデータベース・テンプレートによっては、一部のオプションがすでに組み込まれている場合があります。すでにインストール済みのオプションはグレー表示されます。

- Oracle Spatial
- Oracle Ultra Search
- Oracle Label Security
- Oracle Data Mining
- Oracle OLAP
- Sample Schema

標準的なデータベース・オプションのリストも表示できます。これらのオプションを常にインストールすることをお薦めしますが、除外するかどうかをオプションで指定できます。次のものが含まれます。

- Oracle JVM
- Oracle Text
- Oracle *inter*Media
- Oracle XML DB

モード、初期化パラメータおよびデータ・ファイルの指定

次のページでは、データベースの詳細を定義できます。モード（専用サーバーまたは共有サーバー）を指定し、初期化パラメータを設定し、データ・ファイルの場所を指定します。Oracle は、作成中のデータベースの記述に基づいて特定の値を判断できます。たとえば、標準的なデータベースとカスタム・データベースのどちらを選択したかに応じて、SGA メモリーのサイズ設定パラメータに適切な設定を選択できます。

データベース作成の完了

データベースを定義するパラメータの仕様をすべて入力した後、次のことを実行できます。

- データベースの即時作成
- データベース・テンプレートとして設定を保存
- データベース作成スクリプトの生成

スクリプトを生成するように選択した場合は、後でそれを使用して、DBCA を使用せずにデータベースを作成できます。また、スクリプトをチェックリストとして使用することもできます。

データベース・オプションの構成

データベース・オプションの構成を選択すると、まだデータベースで使用するよう構成されていない Oracle オプションを追加できます。これにより、データベースの作成時に組み込まなかったオプションと機能を追加できます。これらのオプションについては、2-8 ページの「データベース・オプションの指定」を参照してください。

DBCA を使用したデータベースの削除

DBCA を使用すると、データベースを削除できます。DBCA でデータベースを削除すると、データベース・インスタンスとその制御ファイル、REDO ログ・ファイルおよびデータ・ファイルが削除されます。データベースで使用されるサーバー・パラメータ・ファイル（SPFILE）や初期化パラメータ・ファイルも削除されます。

DBCA テンプレートの管理

DBCA テンプレートは、データベースの作成に必要な情報を含む XML ファイルです。DBCA では、テンプレートを使用して新規データベースが作成され、既存のデータベースのクローンとなります。テンプレート内の情報には、データベースの（データ・ファイル、表領域、制御ファイルおよび REDO ログに関する）オプション、初期化パラメータおよび記憶域属性が含まれています。

テンプレートの使用方法はスクリプトと同様で、サイレント・モードで使用できます。ただし、クローン・データベースを作成するオプションが含まれているため、スクリプトよりも高機能です。このオプションを使用すると、以前に作成したシード・データベースのファイルが適切な位置にコピーされるため、新規に作成する場合に比べるとデータベースを短時間で作成できます。

テンプレートは、次のディレクトリに格納されています。

```
$ORACLE_HOME/assistants/dbca/templates
```

テンプレートを使用する利点

テンプレートを使用すると、次のような利点があります。

- 時間を節約できます。テンプレートを使用すると、データベースの定義が不要になります。
- データベース設定を含むテンプレートを作成することで、再び同じパラメータを指定しなくても、複製データベースを容易に作成できます。
- 簡単に編集できます。データベース・オプションをテンプレートの設定から迅速に変更できます。
- テンプレートを容易に共有できます。テンプレートはマシン間でコピーが可能です。

テンプレートのタイプ

テンプレートには、次の 2 つのタイプがあります。

- シード・テンプレート
- 非シード・テンプレート

次の表に、それぞれのタイプの特性を示します。

タイプ	ファイル拡張子	データ・ファイルを含む	データベース構造
シード	.dbc	Yes	このタイプのテンプレートには、既存（シード）データベースの構造と物理的なデータ・ファイルが含まれています。シード・テンプレートを選択すると、データベースの物理ファイルとスキーマがすでに作成されているため、データベースを短時間で作成できます。データベースはシード・データベースのコピーとして起動するため、作成する必要はありません。 ユーザーは、次の設定のみ変更できます。 ■ データベースの名前 ■ データ・ファイルの場所 ■ 制御ファイル数 ■ REDO ログ・グループ数 ■ 初期化パラメータ 他の設定は、データベースの作成後に変更できます。その場合、DBCA から起動できるカスタム・スクリプトを使用する方法、コマンドラインから SQL 文を入力する方法、Oracle Enterprise Manager を使用する方法があります。 シード・データベースのデータ・ファイルと REDO ログ・ファイルは、`.dfj` 拡張子を持つ別のファイルに圧縮（zip）形式で格納されています。通常、`.dbc` ファイルとそれに対応する `.dfj` ファイルには同じ名前が付いていますが、`.dbc` ファイルには対応する `.dfj` ファイルの場所が格納されるため、これは必須要件ではありません。
非シード	.dbt	No	このタイプのテンプレートは、まったく新しくデータベースを作成する場合に使用します。このテンプレートには、作成するデータベースの特性が含まれます。非シード・テンプレートを使用すると、すべてのデータ・ファイルと REDO ログは（コピーされるのではなく）指定に従って作成され、名前、サイズおよびその他の属性を必要に応じて変更できるため、シード・テンプレートに比べて柔軟性があります。

オラクル社が提供する DBCA テンプレート

Oracle では、次の環境向けのテンプレートが用意されています。

環境	環境の説明
DSS （データ・ウェアハウス）	ユーザーは、大量のデータを処理する多数の複雑な問合せを実行します。応答時間、正確さおよび可用性がポイントになります。 これらの問合せ（通常は読取り専用）には、少数レコードの単純フェッチや、多数の異なる表から膨大なレコードをソートする多数の複雑な問合せなどがあります。
OLTP（オンライン・トランザクション処理）	多数の同時ユーザーが、データへの高速アクセスが必要な多数のトランザクションを実行します。可用性、速度、同時実行性およびリカバリ能力がポイントになります。 トランザクションは、データベース表のデータの読込み（SELECT 文）、書込み（INSERT 文と UPDATE 文）および削除（DELETE 文）からなっています。
General Purpose	このテンプレートを使用すると、汎用に設計されたデータベースを作成できます。このテンプレートでは、DSS および OLTP データベース・テンプレート機能が併用されます。
New Database	このテンプレートを使用すると、最も柔軟にデータベースを定義できます。

DBCA を使用したテンプレートの作成

「Template Management」ページには 3 つのオプションが表示され、既存のテンプレートを変更するか、カスタム・テンプレートを独自に作成できます。選択肢は次のとおりです。

- From an existing template

  既存のテンプレートを使用し、事前定義されたテンプレート設定に基づいて新しいテンプレートを作成できます。初期化パラメータ、記憶域パラメータ、ユーザー定義スクリプトの使用などのテンプレート設定を追加または変更できます。

- From an existing database（structure only）

  データベース・オプション、表領域、データ・ファイルおよび初期化パラメータなど、ソース・データベース内で既存データベースに関して指定されている構造情報が含まれた、新規テンプレートを作成できます。作成されたテンプレートには、ユーザー定義スキーマとそのデータは含まれません。ソース・データベースは、ローカルでもリモートでもかまいません。

- From an existing database（structure as well as data--a seed database）

  既存データベースの構造情報と物理データ・ファイルの両方を持つ新規テンプレートを作成できます。このテンプレートを使用すると、ソース・データベースと同一のデータベースが作成されます。作成されたテンプレートには、ユーザー定義スキーマとそのデータが含まれます。ソース・データベースは、ローカルである必要があります。

テンプレートは XML ファイルとして保存されます。

既存データベースからテンプレートを作成する場合は、オプションでファイル・パスを Optimal Flexible Architecture（OFA）に変換するか、既存のファイル・パスをそのまま使用するかを選択できます。テンプレートを使用してデータベースを作成する予定のマシンのディレクトリ構造が異なる場合は、OFA を使用することをお薦めします。ターゲット・マシンのディレクトリ構造が同一の場合は、非 OFA を使用できます。

DBCA テンプレートの削除

「Template Management」ページでは、既存のテンプレートの削除も可能です。

DBCA のサイレント・モードの使用

サイレント・モードには、ユーザー・インタフェース（コマンドラインから最初に入力するインタフェース以外）はありません。情報、エラーおよび警告など、すべてのメッセージはログ・ファイルに出力されます。

コマンドラインから次のコマンドを入力すると、サイレント・モードで使用可能なすべての DBCA オプションが表示されます。

```
dbca -help
```

ここでは、サイレント・モードの使用例を示します。

DBCA のサイレント・モードの例 1: クローン・データベースの作成

クローン・データベースを作成するには、コマンドラインから次のように入力します。

```
% dbca -silent -createDatabase -templateName Transaction_Processing.dbc
-gdbname ora9i -sid ora9i -datafileJarLocation
/private/oracle9i/ora9i/assistants/dbca/templates -datafileDestination
/private/oracle9i/ora9i/oradata -responseFile NO_VALUE
-characterset WE8ISO8859P1
```

DBCA のサイレント・モードの例 2: シード・テンプレートの作成

シード・テンプレートを作成するには、コマンドラインから次のように入力します。

```
% dbca -silent -createCloneTemplate -sourceDB ora9I -sysDBAUserName
sys -sysDBAPassword change_on_install -templateName copy_of_ora9i.dbc
-datafileJarLocation /private/oracle/ora9i/assistants/dbca/templates
```

Oracle データベースの手動作成

ここでは、データベースを手動で作成する際に必要な手順について説明します。これらの手順は、説明されている順序どおりに行う必要があります。Oracle ソフトウェアのインストール・プロセスの一部として、オペレーティング・システム固有の環境変数など、Oracle データベースの作成環境をあらかじめ作成しておきます。

各手順に示されている例では、データベース mynewdb を作成しようとしています。

注意： この時点では、この項に記載されている初期化パラメータとデータベース構造について、すべての説明が終わっていません。各手順には、他のマニュアルへの相互参照が多数記載されています。これらの相互参照を使用して、不明なパラメータや構造について理解してください。

手順 1: インスタンス識別子（SID）の決定

インスタンスに対して一意の Oracle システム識別子（SID）を決定し、環境変数 ORACLE_SID に設定します。この識別子は、後で作成してシステムで同時に実行する他の Oracle インスタンスとの競合を回避するために使用されます。

次の例では、これから作成するインスタンスとデータベースの SID を設定しています。

```
% setenv ORACLE_SID mynewdb
```

DB_NAME 初期化パラメータの値は、SID の設定と一致させる必要があります。

手順 2: DBA の認証方式の設定

データベースを作成するには、作成するユーザーが認証を受け、適切なシステム権限が付与されている必要があります。認証方式には、パスワード・ファイル方式またはオペレーティング・システム方式を使用できます。DBA の認証と認可については、このマニュアルの次の項を参照してください。

- 1-10 ページ「DBA のセキュリティと権限」
- 1-13 ページ「DBA の認証」
- 1-20 ページ「パスワード・ファイルの作成とメンテナンス」

手順 3: 初期化パラメータ・ファイルの作成

Oracle データベースのインスタンス（SGA とバックグラウンド・プロセス）は、初期化パラメータ・ファイルを使用して起動されます。初期化パラメータ・ファイルを作成するには、インストールされているサンプルの初期化パラメータ・ファイルをコピーするか、このマニュアルのサンプルを編集します。

操作を簡単にするために、デフォルトのファイル名を使用して、Oracle のデフォルトの場所に初期化パラメータ・ファイルを配置します。こうすることにより、Oracle が自動的に初期化パラメータ・ファイルのデフォルトの場所を参照するので、データベースの起動時に PFILE パラメータを指定する必要がなくなります。

パラメータ・ファイルのデフォルトの場所は、次の表のとおりです。

プラットフォーム	デフォルト名	デフォルトの場所
UNIX	init$ORACLE_SID.ora たとえば、mynewdb データベースの初期化パラメータ・ファイル名は次のようになります。 initmynewdb.ora	$ORACLE_HOME/dbs たとえば、mynewdb データベースの初期化パラメータ・ファイルは次の場所に配置されます。 /vobs/oracle/dbs/initmynewdb.ora
Windows	init%ORACLE_SID%.ora	%ORACLE_HOME%¥database

次に、mynewdb データベースの作成に使用する初期化パラメータ・ファイルの内容を示します。

初期化パラメータ・ファイルのサンプル

```
# Cache and I/O
DB_BLOCK_SIZE=4096
DB_CACHE_SIZE=20971520

# Cursors and Library Cache
CURSOR_SHARING=SIMILAR
OPEN_CURSORS=300

# Diagnostics and Statistics
BACKGROUND_DUMP_DEST=/vobs/oracle/admin/mynewdb/bdump
CORE_DUMP_DEST=/vobs/oracle/admin/mynewdb/cdump
TIMED_STATISTICS=TRUE
USER_DUMP_DEST=/vobs/oracle/admin/mynewdb/udump

# Control File Configuration
CONTROL_FILES=("/vobs/oracle/oradata/mynewdb/control01.ctl",
               "/vobs/oracle/oradata/mynewdb/control02.ctl",
               "/vobs/oracle/oradata/mynewdb/control03.ctl")

# Archive
LOG_ARCHIVE_DEST_1='LOCATION=/vobs/oracle/oradata/mynewdb/archive'
LOG_ARCHIVE_FORMAT=%t_%s.dbf
LOG_ARCHIVE_START=TRUE

# Shared Server
# Uncomment and use first DISPATCHES parameter below when your listener is
# configured for SSL
# (listener.ora and sqlnet.ora)
# DISPATCHERS = "(PROTOCOL=TCPS)(SER=MODOSE)",
#               "(PROTOCOL=TCPS)(PRE=oracle.aurora.server.SGiopServer)"
DISPATCHERS="(PROTOCOL=TCP)(SER=MODOSE)",
            "(PROTOCOL=TCP)(PRE=oracle.aurora.server.SGiopServer)",
             (PROTOCOL=TCP)

# Miscellaneous
COMPATIBLE=9.2.0
DB_NAME=mynewdb

# Distributed, Replication and Snapshot
DB_DOMAIN=us.oracle.com
REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE
```

```
# Network Registration
INSTANCE_NAME=mynewdb

# Pools
JAVA_POOL_SIZE=31457280
LARGE_POOL_SIZE=1048576
SHARED_POOL_SIZE=52428800

# Processes and Sessions
PROCESSES=150

# Redo Log and Recovery
FAST_START_MTTR_TARGET=300

# Resource Manager
RESOURCE_MANAGER_PLAN=SYSTEM_PLAN

# Sort, Hash Joins, Bitmap Indexes
SORT_AREA_SIZE=524288

# Automatic Undo Management
UNDO_MANAGEMENT=AUTO
UNDO_TABLESPACE=undotbs
```

関連項目：

- これらのパラメータおよびその他の初期化パラメータの詳細は、2-34 ページの「初期化パラメータとデータベースの作成」を参照してください。

手順 4: インスタンスへの接続

SQL*Plus を起動して、AS SYSDBA として Oracle インスタンスに接続します。

```
$ SQLPLUS /nolog
CONNECT SYS/password AS SYSDBA
```

手順 5: インスタンスの起動

データベースをマウントせずにインスタンスを起動します。通常、この方法で起動するのはデータベースの作成時またはメンテナンス時のみです。STARTUP コマンドで、NOMOUNT オプションを指定します。この例では、初期化パラメータ・ファイルがデフォルトの場所に配置されているため、PFILE 句を指定する必要はありません。

```
STARTUP NOMOUNT
```

この時点では、データベースが存在しません。新しいデータベースの作成に備えて、SGA のみが作成され、バックグラウンド・プロセスが起動します。

関連項目：

- 2-43 ページ「サーバー・パラメータ・ファイルを使用した初期化パラメータの管理」
- STARTUP コマンドの使用方法は、第 4 章「起動と停止」を参照してください。

手順 6: CREATE DATABASE 文の発行

新しいデータベースを作成するには、CREATE DATABASE 文を使用します。次の文では、データベース mynewdb を作成します。

```
CREATE DATABASE mynewdb
   USER SYS IDENTIFIED BY pz6r58
   USER SYSTEM IDENTIFIED BY y1tz5p
   LOGFILE GROUP 1 ('/vobs/oracle/oradata/mynewdb/redo01.log') SIZE 100M,
           GROUP 2 ('/vobs/oracle/oradata/mynewdb/redo02.log') SIZE 100M,
           GROUP 3 ('/vobs/oracle/oradata/mynewdb/redo03.log') SIZE 100M
   MAXLOGFILES 5
   MAXLOGMEMBERS 5
   MAXLOGHISTORY 1
   MAXDATAFILES 100
   MAXINSTANCES 1
   CHARACTER SET US7ASCII
   NATIONAL CHARACTER SET AL16UTF16
   DATAFILE '/vobs/oracle/oradata/mynewdb/system01.dbf' SIZE 325M REUSE
   EXTENT MANAGEMENT LOCAL
   DEFAULT TEMPORARY TABLESPACE tempts1
      DATAFILE '/vobs/oracle/oradata/mynewdb/temp01.dbf'
      SIZE 20M REUSE
   UNDO TABLESPACE undotbs
      DATAFILE '/vobs/oracle/oradata/mynewdb/undotbs01.dbf'
      SIZE 200M REUSE AUTOEXTEND ON NEXT 5120K MAXSIZE UNLIMITED;
```

次の特性を持つデータベースが作成されます。

- データベースには mynewdb という名前が付けられます。グローバル・データベース名は、mynewdb.us.oracle.com です。2-36 ページの「DB_NAME 初期化パラメータ」および「DB_DOMAIN 初期化パラメータ」を参照してください。
- CONTROL_FILES 初期化パラメータで指定された 3 つの制御ファイルが作成されます。2-36 ページの「制御ファイルの指定」を参照してください。

- ユーザー SYS のパスワードは pz6r58、SYSTEM のパスワードは y1tz5p です。この 2 つは SYS および SYSTEM のパスワードを指定する句で、このリリースの Oracle9*i* ではオプションです。ただし、指定する場合は、両方の句を指定する必要があります。これらの句の使用方法の詳細は、2-23 ページの「データベースの保護 : ユーザー SYS および SYSTEM のパスワードの指定」を参照してください。
- 新しいデータベースには、LOGFILE 句で指定された 3 つのオンライン REDO ログ・ファイルがあります。MAXLOGFILES、MAXLOGMEMBERS および MAXLOGHISTORY は、REDO ログの制限を定義します。第 7 章「オンライン REDO ログの管理」を参照してください。
- MAXDATAFILES によって、このデータベースでオープンできるデータ・ファイルの最大数が指定されます。この数は、制御ファイルの初期サイズに影響を及ぼします。

注意： データベースの作成時に、制限をいくつか設定できます。これらの制限には、オペレーティング・システムの制限に置き換わるものもあり、それらの制限の影響を受ける可能性があります。たとえば、MAXDATAFILES を設定すると、Oracle は、初期のデータベースにデータ・ファイルが 1 つしかなくても、制御ファイルに MAXDATAFILES 個のファイル名を格納できるだけの領域を割り当てます。ただし、制御ファイルの最大サイズは制限されており、オペレーティング・システムによって異なるので、CREATE DATABASE 文のパラメータをすべて理論的な最大値で設定できるとはかぎりません。

データベース作成時に制限を設定する方法の詳細は、『Oracle9*i* SQL リファレンス』および使用しているオペレーティング・システム固有の Oracle マニュアルを参照してください。

- MAXINSTANCES の指定により、このデータベースをマウントし、オープンできるインスタンスが 1 つのみに制限されます。
- このデータベースにデータを格納する際は、US7ASCII キャラクタ・セットが使用されます。
- NATIONAL CHARACRTER SET として AL16UTF16 キャラクタ・セットが指定されます。このキャラクタ・セットは、特に NCHAR、NCLOB または NVARCHAR2 として定義された列にデータを格納する際に使用されます。
- DATAFILE 句で指定したとおりに、SYSTEM 表領域（オペレーティング・システム・ファイル /vobs/oracle/oradata/mynewdb/system01.dbf からなる）が作成されます。ファイルがすでに存在する場合は上書きされます。
- SYSTEM 表領域はローカル管理になります。2-27 ページの「ローカル管理の SYSTEM 表領域の作成」を参照してください。

- DEFAULT_TEMPORARY_TABLESPACE 句により、指定した名前で、このデータベース用のデフォルト一時表領域が作成されます。2-25 ページの「デフォルト一時表領域の作成」を参照してください。
- UNDO_TABLESPACE 句により、UNDO 表領域が指定の名前で作成されます。初期化パラメータ・ファイルで UNDO_MANAGEMENT=AUTO を指定した場合は、このデータベース用の UNDO レコードがこの UNDO 表領域に格納されます。2-24 ページの「自動 UNDO 管理の使用 : UNDO 表領域の作成」を参照してください。
- この CREATE DATABASE 文では ARCHIVELOG 句が指定されていないので、初期状態では REDO ログ・ファイルはアーカイブされません。これはデータベース作成時の慣例です。後で ALTER DATABASE 文を使用して、ARCHIVELOG モードに切り替えることができます。mynewdb のアーカイブに関連する初期化パラメータ・ファイル内の初期化パラメータは、LOG_ARCHIVE_DEST_1、LOG_ARCHIVE_FORMAT および LOG_ARCHIVE_START です。第 8 章「アーカイブ REDO ログの管理」を参照してください。

関連項目：

- 2-23 ページ「CREATE DATABASE 文の理解」
- CREATE DATABASE 文で指定できる句およびパラメータ値の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

手順 7: 追加の表領域の作成

データベースを稼働させるには、ユーザー用のファイルと表領域を追加作成する必要があります。次のサンプル・スクリプトは、追加の表領域を作成します。

```
CONNECT SYS/password AS SYSDBA
-- create a user tablespace to be assigned as the default tablespace for users
CREATE TABLESPACE users LOGGING
     DATAFILE '/vobs/oracle/oradata/mynewdb/users01.dbf'
     SIZE 25M REUSE AUTOEXTEND ON NEXT  1280K MAXSIZE UNLIMITED
     EXTENT MANAGEMENT LOCAL;
-- create a tablespace for indexes, separate from user tablespace
CREATE TABLESPACE indx LOGGING
     DATAFILE '/vobs/oracle/oradata/mynewdb/indx01.dbf'
     SIZE 25M REUSE AUTOEXTEND ON NEXT  1280K MAXSIZE UNLIMITED
     EXTENT MANAGEMENT LOCAL;
EXIT
```

表領域の作成方法の詳細は、第 11 章「表領域の管理」を参照してください。

手順 8: スクリプトの実行によるデータ・ディクショナリ・ビューの作成

ビュー、シノニムおよび PL/SQL パッケージの作成に必要なスクリプトを実行します。

```
CONNECT SYS/password AS SYSDBA
@/vobs/oracle/rdbms/admin/catalog.sql
@/vobs/oracle/rdbms/admin/catproc.sql
EXIT
```

次の表に、スクリプトの説明を示します。

スクリプト	説明
CATALOG.SQL	データ・ディクショナリ表のビュー、動的パフォーマンス・ビューおよび多くのビューに対するパブリック・シノニムを作成します。シノニムに PUBLIC アクセスを付与します。
CATPROC.SQL	PL/SQL に必要なスクリプトや PL/SQL とともに使用されるスクリプトをすべて実行します。

必要に応じて、他のスクリプトも実行できます。実行するスクリプトは、使用またはインストール対象として選択する機能とオプションによって異なります。使用可能なスクリプトは、『Oracle9*i* データベース・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格納場所については、オペレーティング・システム固有の Oracle インストレーション・ガイドを参照してください。

手順 9: スクリプトの実行による追加オプションのインストール（オプション）

このデータベースとともに稼働する他の Oracle 製品をインストールする場合は、それらの製品のインストール指示を参照してください。製品によっては、追加のデータ・ディクショナリ表を作成する必要があります。通常は、これらの表を作成して、データベースのデータ・ディクショナリにロードするためのコマンド・ファイルが提供されます。

インストールを計画している製品のインストールと管理に関する指示は、製品固有の Oracle マニュアルを参照してください。

手順 10: サーバー・パラメータ・ファイルの作成（推奨）

初期化パラメータを動的にメンテナンスする方法として、サーバー・パラメータ・ファイルの作成をお薦めします。サーバー・パラメータ・ファイルについては、2-43 ページの「サーバー・パラメータ・ファイルを使用した初期化パラメータの管理」を参照してください。

次のスクリプトは、テキスト初期化パラメータ・ファイルからサーバー・パラメータ・ファイルを作成し、デフォルトの場所に書き込みます。インスタンスをいったん停止し、デフォルトの場所にあるサーバー・パラメータ・ファイルを使用して再起動します。

```
CONNECT SYS/password AS SYSDBA
-- create the server parameter file
CREATE SPFILE='/vobs/oracle/dbs/spfilemynewdb.ora' FROM
       PFILE='/vobs/oracle/admin/mynewdb/scripts/init.ora';
SHUTDOWN
-- this time you will start up using the server parameter file
CONNECT SYS/password AS SYSDBA
STARTUP
EXIT
```

手順 11: データベースのバックアップ

メディア障害が発生した場合にリカバリするための完全なファイルのセットが確実に存在するように、データベースの全体バックアップを作成する必要があります。データベースのバックアップの詳細は、『Oracle9*i* バックアップおよびリカバリ概要』を参照してください。

CREATE DATABASE 文の理解

CREATE DATABASE 文を実行すると、Oracle は（最低限）次の操作を実行します。実際の操作の大部分は、CREATE DATABASE 文で指定した句、または初期化パラメータの設定に応じて実行されます。

- データベース用のデータ・ファイルの作成
- データベース用の制御ファイルの作成
- データベース用の REDO ログ・ファイルの作成と ARCHIVELOG モードの設定
- SYSTEM 表領域と SYSTEM ロールバック・セグメントの作成
- データ・ディクショナリの作成
- データベースへのデータの格納に使用するキャラクタ・セットの設定
- データベース・タイム・ゾーンの設定
- データベースのマウントとオープン

この項では、CREATE DATABASE 文の複数の句について説明します。これには、2-18 ページの「手順 6: CREATE DATABASE 文の発行」で説明した句と、その他の句が含まれます。

この項の内容は、次のとおりです。

- データベースの保護 : ユーザー SYS および SYSTEM のパスワードの指定
- データベースの作成と管理を簡単にする句
- ローカル管理の SYSTEM 表領域の作成
- データベースのタイム・ゾーンとタイム・ゾーン・ファイルの指定
- FORCE LOGGING モードの指定

データベースの保護 : ユーザー SYS および SYSTEM のパスワードの指定

CREATE DATABASE 文でユーザー SYS および SYSTEM のパスワード指定に使用する句は、次のとおりです。

- USER SYS IDENTIFIED BY *password*
- USER SYSTEM IDENTIFIED BY *password*

パスワードを指定しなければ、これらのユーザーにはそれぞれデフォルトのパスワード change_on_install および manager が割り当てられます。デフォルトのパスワードが使用された場合は、そのことを示すレコードがアラート・ファイルに書き込まれます。データベースを保護するには、データベース作成後に ALTER USER 文を使用して、これらのパスワードを変更する必要があります。

これらの句は、このリリースではオプションですが、指定することをお薦めします。デフォルトのパスワードは一般に知られており、後で変更するのを怠ると、自分自身を悪意のユーザーによる攻撃にさらすことになります。

関連項目： 2-31 ページ「セキュリティに関する考慮点」

データベースの作成と管理を簡単にする句

Oracle9*i* には、データベース作成時における DBCA の使用以外にも、データベースの作成、操作および管理を簡単にするオプションが用意されています。これらのオプションとそれに関連する CREATE DATABASE 句については、次の各項で概要を説明します。詳細は、以降の項を参照してください。

- 自動 UNDO 管理の使用 : UNDO 表領域の作成
- デフォルト一時表領域の作成
- Oracle Managed Files の使用

自動 UNDO 管理の使用 : UNDO 表領域の作成

データベースでロールバック・セグメントを使用するかわりに、UNDO 表領域を使用することをお薦めします。そのためには、異なる初期化パラメータ・セットを使用する必要があります。また、オプションで CREATE DATABASE 文に UNDO TABLESPACE 句を含めます。

データベースを自動 UNDO 管理モードで操作する場合は、次の初期化パラメータを組み込む必要があります。

```
UNDO_MANAGEMENT=AUTO
```

このモードでは、**UNDO** と呼ばれるロールバック情報がロールバック・セグメントではなく UNDO 表領域に格納され、Oracle によって管理されます。UNDO 表領域用に特定の表領域を作成して名前を付ける必要がある場合は、データベース作成時に UNDO TABLESPACE 句を指定ができます。この句を省略して自動 UNDO 管理を指定すると、デフォルトの UNDO 表領域 SYS_UNDOTBS が作成されます。

関連項目：

- 2-41 ページ「UNDO 領域管理方法の指定」
- UNDO 表領域の作成と使用の詳細は、第 13 章「UNDO 領域の管理」を参照してください。

デフォルト一時表領域の作成

CREATE DATABASE 文で DEFAULT TEMPORARY TABLESPACE 句を指定すると、データベース作成時に一時表領域が作成されます。この表領域は、他の方法で一時表領域を割り当てられていないユーザー用のデフォルトの一時表領域として使用されます。

デフォルト一時表領域は、CREATE USER 文で明示的に割り当てることができます。しかし、一時表領域を指定しない場合は、デフォルトで SYSTEM 表領域に設定されます。一時データを SYSTEM 表領域に格納するのはよい方法ではありません。この問題を回避し、CREATE USER 文の実行時に各ユーザーへのデフォルト一時表領域の割当てを不要にするには、CREATE DATABASE 文で DEFAULT TEMPORARY TABLESPACE 句を使用します。

デフォルトの一時表領域を後で変更したり、データベース作成後に最初のデフォルト表領域を作成することも可能です。これを行うには、CREATE TEMPORARY TABLESPACE 文で新しい一時表領域を作成してから、ALTER DATABASE DEFAULT TEMPORARY TABLESPACE 文を使用して、作成した一時表領域を割り当てます。ユーザーの一時表領域は自動的に新しいデフォルト一時表領域に切り替えられます（または割り当てられます）。

次の文は、新しいデフォルト一時表領域を割り当てます。

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE tempts2;
```

新しいデフォルト一時表領域には、既存の一時表領域を指定する必要があります。ローカル管理の SYSTEM 表領域を使用している場合は、新しいデフォルト一時表領域もローカル管理にする必要があります。

デフォルト一時表領域は削除できませんが、新しいデフォルト一時表領域を割り当ててから、古いほうの一時表領域を削除することは可能です。デフォルト一時表領域を永続表領域に変更することはできません。また、デフォルト一時表領域をオフラインにすることもできません。

現行のデフォルト一時表領域の名前を取得するには、DATABASE_PROPERTIES ビューを使用します。PROPERTY_NAME 列に値「DEFAULT_TEMP_TABLESPACE」が表示され、PROPERTY_VALUE 列にデフォルト一時表領域の名前が表示されます。

関連項目：

- CREATE DATABASE および ALTER DATABASE の DEFAULT TEMPORARY TABLESPACE 句の構文は、『Oracle9*i* SQL リファレンス』を参照してください。
- 一時表領域の作成と使用の詳細は、11-11 ページの「一時表領域」を参照してください。

Oracle Managed Files の使用

Oracle Managed Files 機能を使用すると、CREATE DATABASE 文で指定する句とパラメータの数を最小限に抑えることができます。

初期化パラメータ・ファイルに DB_CREATE_FILE_DEST または DB_CREATE_ONLINE_LOG_DEST_*n* 初期化パラメータを設定すると、データベースの基礎となるオペレーティング・システム・ファイルを Oracle によって作成および管理することができます。初期化パラメータおよび CREATE DATABASE 文で指定した句に応じて、次のデータベース構造のオペレーティング・システム・ファイルが自動的に作成および管理されます。

- 表領域
- 一時表領域
- 制御ファイル
- オンライン REDO ログ・ファイル

次の CREATE DATABASE 文を例として、Oracle Managed Files 機能の動作を示します。

```
CREATE DATABASE rbdb1
     USER SYS IDENTIFIED BY pz6r58
     USER SYSTEM IDENTIFIED BY y1tz5p
     UNDO TABLESPACE undotbs
     DEFAULT TEMPORARY TABLESPACE tempts1;
```

- DATAFILE 句が指定されていないので、Oracle Managed Files の SYSTEM 表領域用のデータ・ファイルが作成されます。
- LOGFILE 句が指定されていないので、Oracle が管理するオンライン REDO ログ・ファイルのグループが 2 つ作成されます。
- UNDO TABLESPACE 句に DATAFILE 副次句が指定されていないので、Oracle Managed Files の UNDO 表領域用のデータ・ファイルが作成されます。
- DEFAULT TEMPORARY TABLESPACE 句に TEMPFILE 副次句が指定されていないので、Oracle Managed Files の一時ファイルが作成されます。
- また、初期化パラメータ・ファイルに CONTROL_FILES 初期化パラメータが指定されていない場合は、Oracle Managed Files の制御ファイルが作成されます。
- サーバー・パラメータ・ファイル（2-43 ページの「サーバー・パラメータ・ファイルを使用した初期化パラメータの管理」を参照）を使用している場合は、作成されたファイルに応じて初期化パラメータが自動的に設定されます。

> **関連項目：** Oracle Managed Files 機能と使用方法の詳細は、第 3 章「Oracle Managed Files の使用」を参照してください。

ローカル管理の SYSTEM 表領域の作成

CREATE DATABASE 文に EXTENT MANAGEMENT LOCAL 句を指定すると、Oracle によりエクステント・サイズが決定されるローカル管理の SYSTEM 表領域を作成できます。この文を正常に実行するには、COMPATIBLE 初期化パラメータを 9.2 以上に設定する必要があります。EXTENT MANAGEMENT LOCAL 句を指定しない場合は、デフォルトでディレクトリ管理の SYSTEM 表領域が作成されます。

ローカル管理表領域を使用すると、ディクショナリ管理表領域に比べてパフォーマンスが向上し、管理が容易になります。ローカル管理の SYSTEM 表領域は、デフォルトで AUTOALLOCATE により作成されます。これは、エクステント・サイズが Oracle により判断され、制御されるシステム管理の表領域です。ローカル管理の SYSTEM 表領域では、自動割当てポリシーのため、作成されるオブジェクトの初期サイズが大きくなる場合があります。ローカル管理の SYSTEM 表領域を作成する場合に、UNIFORM エクステント・サイズを指定することはできません。

ローカル管理の SYSTEM 表領域を指定してデータベースを作成する場合は、次の条件が満たされているかどうかを確認してください。

- 必ずデフォルトの一時表領域があること。その表領域を SYSTEM 表領域にしないこと。
- ディクショナリ管理表領域にロールバック・セグメントを作成しないこと。SYSTEM 表領域がローカル管理の場合に、ディクショナリ管理表領域にロールバック・セグメントを作成すると失敗します。

最初の条件を満たすために、CREATE DATABASE 文に DEFAULT TEMPORARY TABLESPACE 句を指定できます。句を指定しない場合は、Oracle でデフォルトの名前とデフォルトの位置を使用して表領域が自動的に作成されます。

2 番目の条件を満たすために、データベースの UNDO レコードの管理には、ロールバック・セグメントではなく自動 UNDO 管理を使用することをお薦めします。CREATE DATABASE 文に UNDO TABLESPACE 句を指定すると、特定の UNDO 表領域を作成できます。この句を指定しない場合は、デフォルトの名前とデフォルトの位置を使用して、Oracle によりローカル管理の UNDO 表領域が自動的に作成されます。

注意： SYSTEM 表領域がローカル管理の場合、データベースの他の表領域には次の制限が適用されます。

- データベースには、ディクショナリ管理表領域を作成できません。
- ローカル管理表領域はディクショナリ管理表領域に移行できません。
- ディクショナリ管理表領域をデータベースにトランスポートできますが、読取り / 書込みモードには変更できません。
- 既存のディクショナリ管理表領域は、READ ONLY モードの場合にのみデータベースに残すことができます。READ WRITE モードには変更できません。

また、Oracle では、DBMS_SPACE_ADMIN パッケージを使用すると、既存のディクショナリ管理 SYSTEM 表領域をローカル管理表領域に移行できます。ただし、下位へと移行するためのプロシージャは用意されていません。

関連項目：

- SYSTEM 表領域に対して EXTENT MANAGEMENT LOCAL を指定する場合の DEFAULT TEMPORARY TABLESPACE および UNDO TABLESPACE 句の使用の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。
- 11-5 ページ「ローカル管理表領域」
- 11-33 ページ「ローカル管理表領域への SYSTEM 表領域の移行」

データベースのタイム・ゾーンとタイム・ゾーン・ファイルの指定

Oracle では、データベースのデフォルト・タイム・ゾーンを指定し、オプションでサポートするタイム・ゾーン・ファイルのサイズを選択できます。

データベースのタイム・ゾーンの指定

データベースのデフォルト・タイム・ゾーンを設定するには、CREATE DATABASE 文で SET TIME_ZONE 句を指定します。この句を省略した場合、デフォルトのデータベース・タイム・ゾーンはオペレーティング・システムのタイム・ゾーンと同じです。ALTER SESSION 文を使用すると、セッションのデータベース・タイム・ゾーンを変更できます。

関連項目： データベース・タイム・ゾーンの設定の詳細は、『Oracle9*i* Database グローバリゼーション・サポート・ガイド』を参照してください。

データベースのタイム・ゾーン・ファイルの指定

Oracle9*i* では、CREATE DATABASE 文で SET TIME_ZONE 句を使用して、データベースのデフォルト・タイム・ゾーンを指定できます。ここでは、特に Solaris プラットフォームについて、この機能のサポートに使用されるタイム・ゾーン・ファイルの詳細を説明します。ディレクトリ名、ファイル名および環境変数はプラットフォームごとに異なりますが、UNIX プラットフォームではおそらくすべて同じです。

タイム・ゾーン・ファイルには有効なタイム・ゾーン名が含まれており、ゾーンごとに次の情報が含まれています（略称は必ずゾーン名と組み合せて使用します）。

- UTC からのオフセット
- 夏時間への移行期間
- 標準時間の略称
- 夏時間の略称

Oracle がインストールされているディレクトリに、次の 2 つのタイム・ゾーン・ファイルがあります。

- `$ORACLE_HOME/oracore/zoneinfo/timezone.dat`

  これはデフォルトです。このファイルには最も一般的に使用されるタイム・ゾーンが含まれています。登録数が少ないので、このファイルを使用するとデータベースのパフォーマンスが向上します。

- `$ORACLE_HOME/oracore/zoneinfo/timezlrg.dat`

  このファイルには、デフォルトより多くの定義済みタイム・ゾーンが保存されています。このファイルは、デフォルトの `timezone.dat` ファイルで定義されていないゾーンを必要とするユーザーが使用します。ゾーン情報セットは登録数が多いので、パフォーマンスに影響を与える場合があります。

定義数の多いタイム・ゾーン・データ・ファイルを使用可能にする場合は、次のことを実行する必要があります。

1. データベースを停止します。
2. 環境変数 `ORA_TZFILE` に、`timezlrg.dat` ファイルの位置のフルパス名を設定します。
3. データベースを再起動します。

定義数の多い `timezlrg.dat` を使用すると、データベースに格納されているデータがすべてデフォルトのゾーンを使用していることが確認されないかぎり、このファイルを継続して使用する必要があります。また、情報を共有しているすべてのデータベースが同じタイム・ゾーン・データ・ファイルを使用する必要があります。

タイム・ゾーン名を表示するには、次の問合せを実行します。

```
SELECT * FROM V$TIMEZONE_NAMES;
```

FORCE LOGGING モードの指定

一部の DDL 文（`CREATE TABLE` など）では `NOLOGGING` 句を使用できますが、ある種のデータベース操作ではデータベース REDO ログに REDO レコードが生成されません。`NOLOGGING` 句を指定すると、データベース・リカバリ・メカニズムの外で容易にリカバリする操作は高速化されますが、メディア・リカバリとスタンバイ・データベースに問題が発生します。

Oracle には、DDL 文に `NOLOGGING` が指定されていても、データベースに対する変更について REDO レコードを強制的に書き込ませる手段が用意されています。Oracle では、一時表領域と一時セグメントの REDO レコードは生成されないため、この場合、FORCE LOGGING は影響しません。

関連項目：

- NOLOGGING モードの詳細は、『Oracle9*i* データベース概要』を参照してください。
- NOLOGGING モードで実行できる操作の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

FORCE LOGGING 句の使用

データベースを FORCE LOGGING モードにするには、CREATE DATABASE 文で FORCE LOGGING 句を使用します。この句を指定しない場合、データベースは FORCE LOGGING モードになりません。

データベースの作成後に、ALTER DATABASE 文を使用してデータベースを FORCE LOGGING モードにします。この文は、ロギングなしの直接書込みがすべて完了するまで待機するため、完了までに長時間の待機が発生する可能性があります。

次の SQL 文を使用すると、FORCE LOGGING モードを取消しできます。

```
ALTER DATABASE NO FORCE LOGGING
```

データベースに対して FORCE LOGGING を指定するかどうかに関係なく、表領域レベルで FORCE LOGGING または NO FORCE LOGGING を選択的に指定できます。ただし、データベースの FORCE LOGGING モードが有効になっている場合は、表領域のモード設定より優先されます。データベースに対して有効になっていない場合は、個々の表領域の設定が実行されます。データベース全体を FORCE LOGGING モードにするか、個々の表領域を FORCE LOGGING モードにすることをお薦めします。一度に両方の設定を行わないでください。

FORCE LOGGING モードは、データベースの永続属性です。つまり、データベースが停止され、再起動されても、同じロギング・モードのままです。ただし、制御ファイルを再作成すると、CREATE CONTROL FILE 文で FORCE LOGGING 句を指定しないかぎり、データベースは FORCE LOGGING モードで再起動しません。

関連項目： 表領域の作成に FORCE LOGGING 句を使用する方法の詳細は、11-19 ページの「REDO レコードの書込みの制御」を参照してください。

FORCE LOGGING モードのパフォーマンスに関する考慮点

FORCE LOGGING モードはパフォーマンスの低下を伴います。アクティブになっているスタンバイ・データベースがなく、主として完全メディア・リカバリを確実に行うために FORCE LOGGING を指定する場合は、次の点を考慮する必要があります。

- 発生すると思われるメディア障害の数
- ロギングなしの直接書込みをリカバリできない場合のダメージの重大度
- FORCE LOGGING モードによるパフォーマンスの低下は許容範囲内かどうか

データベースが NOARCHIVELOG モードで稼働している場合、通常は FORCE LOGGING モードにする利点はありません。これは、このモードではメディア・リカバリが不可能なので、ほとんど利点はない一方、パフォーマンスが低下するためです。

データベース作成のトラブルシューティング

なんらかの理由でデータベースの作成が失敗した場合は、インスタンスを停止し、CREATE DATABASE 文によって作成されたファイルをすべて削除した後で、データベースを作成しなおしてください。データベースの作成に失敗した原因となるエラーを訂正し、スクリプトを再度実行してください。

データベースの削除

データベースを削除するには、データベースのデータ・ファイル、REDO ログ・ファイルおよびその他の関連ファイル（制御ファイル、初期化パラメータ・ファイルおよびアーカイブ・ログ・ファイル）すべてを削除する必要があります。データベースのデータ・ファイル、REDO ログ・ファイルおよび制御ファイルの名前を表示するには、それぞれデータ・ディクショナリ・ビュー V$DATAFILE、V$LOGFILE および V$CONTROLFILE を問い合せます。

データベースがアーカイブ・ログ・モードの場合は、初期化パラメータ LOG_ARCHIVE_DEST_*n* または LOG_ARCHIVE_DEST と LOG_ARCHIVE_DUPLEX_DEST を調べてアーカイブ・ログの保存先を特定します。

DBCA を使用してデータベースを作成した場合は、データベースの削除とファイルのクリーン・アップに DBCA を使用できます。

関連項目： これらのビューと初期化パラメータの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

データベースを作成した後の考慮点

データベースの作成後は、インスタンスが稼働しており、データベースがオープンしているので、通常どおりにデータベースを使用できます。必要に応じて、他の処理を実行できます。ここでは、その処理の一部について説明します。

セキュリティに関する考慮点

新しく作成したデータベースには、データベースの管理に役立つユーザーが少なくとも 3 つ登録されています。SYS、SYSTEM および OUTLN（ストアド・アウトラインが格納されているスキーマの所有者）です。

注意： データベースへの不当なアクセスを防ぎ、データベースの整合性を保護するため、データベースを作成した直後に、SYS および SYSTEM のデフォルト・パスワードを変更してください。

インストールした機能やオプションによっては、他のユーザーが存在する場合があります。次にそのようなユーザーの例を示します。

- `MDSYS`（Spatial）
- `ORDSYS`（*inter*Media Audio）
- `ORDPLUGINS`（*inter*Media Audio）
- `CTXSYS`（Oracle Text）
- `DBSNMP`（Enterprise Manager Intelligent Agent）

ユーザー DBSNMP のパスワードを変更する場合は、『Oracle Intelligent Agent ユーザーズ・ガイド』を参照してください。

セキュリティ拡張に関する注意： このリリース以上の Oracle 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確保するために、セキュリティ機能が拡張されています。

- Database Configuration Assistant（DBCA）を使用した初期インストール時に、`SYS`、`SYSTEM`、`SCOTT`、`DBSNMP`、`OUTLN`、`AURORA$JIS$UTILITY$`、`AURORA$ORB$UNAUTHENTICATED` および `OSE$HTTP$ADMIN` を除くデフォルトのデータベース・ユーザー・アカウントはすべてロックされ、期限切れとなっています。ロックされたアカウントをアクティブにするには、DBA が手動でロックを解除して、新しいパスワードを再設定する必要があります。
- また、DBCA によるデータベースの初期インストール時に、`SYS` および `SYSTEM` に対してデフォルトのパスワードを割り当てるのではなく、ユーザーにパスワードの入力を求めます。この 2 つのユーザーのパスワードは、`CREATE DATABASE` 文を発行して手動で指定することもできます。

関連項目：

- Oracle データベースを安全な方法で構成するためのガイドラインは、23-20 ページの「セキュリティ・チェックリスト」を参照してください。
- ユーザー SYS および SYSTEM の詳細は、1-11 ページの「DBA のユーザー名」を参照してください。
- 新しいユーザーの追加方法とパスワードの変更方法は、24-6 ページの「ユーザーの変更」を参照してください。
- ユーザー・アカウントのロック解除に使用する ALTER USER 文の構文は、『Oracle9*i* SQL リファレンス』を参照してください。

Oracle のサンプル・スキーマのインストール

Oracle データベースの配布メディアには、各種の SQL ファイルが格納されています。この SQL ファイルでは、システムの試用、SQL の学習、追加の表、ビュー、シノニムの作成などが可能です。

Oracle9*i* から、Oracle の機能の理解に役立つサンプル・スキーマが用意されています。いくつかの Oracle マニュアルでは、例を示す際にこのサンプル・スキーマを使用しています。大部分の Oracle マニュアルで、サンプル・スキーマに基づく例を使用するように変更作業が進められています。

次の表は、サンプル・スキーマについて簡単に説明しています。

スキーマ	説明
Human Resources	Human Resources（HR）スキーマは、基本的なリレーショナル・データベース・スキーマです。HR スキーマには、Employees、Departments、Locations、Countries、Jobs および Job_History という 6 つの表があります。Order Entry（OE）スキーマは、HR スキーマにリンクしています。
Order Entry	Order Entry（OE）スキーマは、純粋なリレーショナル・スキーマである Human Resources（HR）スキーマを基に、オブジェクト・リレーショナル機能とオブジェクト指向機能を使用して作成されています。OE スキーマには、Customers、Product_Descriptions、Product_Information、Order_Items、Orders、Inventories および Warehouses という 7 つの表があります。OE スキーマは、HR スキーマと PM スキーマにリンクしています。このスキーマには、ユーザーに透過的なアクセスを提供するために、HR オブジェクトに関するシノニムが定義されています。

スキーマ	説明
Product Media	Product Media（PM）には、2 つの表 online_media と print_media、1 つのオブジェクト型 adheader_typ および 1 つのネストした表 textdoc_typ があります。PM スキーマでは、*inter*Media 列型と LOB 列型が使用されています。 **注意：**Oracle Text を使用する場合は、Oracle Text の索引を作成する必要があります。
Sales History	Sales History（SH）スキーマは、リレーショナル・スター・スキーマの一例です。sales というレンジ・パーティション化された大きなファクト表 1 つと、5 つのディメンション表、times、promotions、channels、products および customers から構成されます。customers にリンクされた追加の表 countries が単純なスノーフレークを示します。
Queued Shipping	Queued Shipping（QS）スキーマは、実際には、メッセージ・キューを含む複数のスキーマです。

サンプル・スキーマは DBCA によって自動的にインストールするか、手動でインストールします。スキーマとインストール手順の詳細は、『Oracle9*i* サンプル・スキーマ』を参照してください。

初期化パラメータとデータベースの作成

オラクル社では、データベース・ソフトウェアとともに、適切な値が設定された開始用の初期化パラメータ・ファイルを提供しており、DBCA でも自動的に作成されます。構成やオプション、およびデータベースのチューニング計画によっては、オラクル社が提供するこれらの初期化パラメータを編集し、他の初期化パラメータを追加することが可能です。関連する初期化パラメータのうち、初期化パラメータ・ファイル内に特に設定されていないものには、デフォルト値が適用されます。

Oracle データベースを初めて作成する場合は、変更するパラメータ値の数を最小限にとどめておくことをお薦めします。データベースと環境に慣れてから、多数の初期化パラメータを `ALTER SYSTEM` 文で動的にチューニングしてください。従来のテキスト形式の初期化パラメータ・ファイルを使用している場合、変更できるのは現行のインスタンスについてのみです。永続的に変更するには、初期化パラメータ・ファイル内で手動で更新する必要があります。それ以外の場合は、データベースを次回に停止して起動すると、変更内容が失われます。

サーバー・パラメータ・ファイルを使用している場合、`ALTER SYSTEM` 文で行った初期化パラメータ・ファイルの変更内容は、停止して起動した後も持続します。この操作については、2-43 ページの「サーバー・パラメータ・ファイルを使用した初期化パラメータの管理」を参照してください。

ここでは、新しいデータベースを作成する前に追加または編集対象として選択できるいくつかの初期化パラメータについて説明します。

この項の内容は、次のとおりです。

- グローバル・データベース名の決定
- 制御ファイルの指定
- データベース・ブロック・サイズの指定
- SGA のサイズに影響する初期化パラメータの設定
- 最大プロセス数の指定
- UNDO 領域管理方法の指定
- ライセンスに関するパラメータの設定

関連項目： デフォルト設定など、すべての初期化パラメータに関する説明は、『Oracle9*i* データベース・リファレンス』を参照してください。

グローバル・データベース名の決定

データベースのグローバル・データベース名は、割り当てたローカル・データベース名と、ネットワーク構造内でのデータベースの位置から構成されます。データベース名のローカル名構成要素は DB_NAME 初期化パラメータによって決まり、ネットワーク構造内のドメイン（論理的な位置）は DB_DOMAIN パラメータによって決まります。これら 2 つのパラメータの設定を組み合せて、ネットワーク内で一意となるデータベース名を形成する必要があります。

たとえば、test.us.acme.com というグローバル・データベース名を持つデータベースを作成するには、新しいパラメータ・ファイルのパラメータを次のように編集します。

```
DB_NAME = test
DB_DOMAIN = us.acme.com
```

ALTER DATABASE RENAME GLOBAL_NAME 文を使用すると、データベースの GLOBAL_NAME を変更できます。ただし、最初に DB_NAME および DB_DOMAIN 初期化パラメータを変更し、制御ファイルを再作成した後に、データベースを停止して再起動する必要があります。

関連項目： データベース名の変更手段である DBNEWID ユーティリティの使用方法は、『Oracle9*i* データベース・ユーティリティ』を参照してください。

DB_NAME 初期化パラメータ

DB_NAME には、8 文字以内のテキスト文字列を設定する必要があります。DB_NAME に指定した名前は、データベースの作成時に、データベースのデータ・ファイル、REDO ログ・

ファイルおよび制御ファイルに記録されます。データベース・インスタンスの起動時に、パラメータ・ファイル内の DB_NAME パラメータの値と制御ファイル内のデータベース名が一致しないと、データベースは起動しません。

DB_DOMAIN 初期化パラメータ

DB_DOMAIN は、データベースが作成されるネットワーク・ドメインを指定するテキスト文字列です。通常、これはデータベースを所有する組織の名前です。作成しようとしているデータベースが分散データベース・システムの一部である場合は、データベースを作成する前に、この初期化パラメータに特に注意してください。

> **関連項目：** 分散データベースの詳細は、第 VI 部「分散データベースの管理」を参照してください。

制御ファイルの指定

新しいパラメータ・ファイルに CONTROL_FILES パラメータを指定し、その値として新しいデータベースで使用する制御ファイル名のリストを設定します。CREATE DATABASE 文を実行すると、CONTROL_FILES パラメータに記述した制御ファイルが作成されます。CONTROL_FILES パラメータにファイル名のリストを記述しないと、オペレーティング・システム固有のデフォルト・ファイル名が使用されます。

データベース用の制御ファイルを作成するときに新しいオペレーティング・システム・ファイルを作成する場合は、CONTROL_FILES パラメータに記述されているファイル名が現在のシステム上に存在するいずれのファイル名とも一致しないことを確認してください。データベース用の制御ファイルを作成するときに既存のファイルを再利用または上書きする場合は、CONTROL_FILES パラメータに記述されているファイル名が、現在のシステム上に存在するファイル名と一致することを確認してください。

注意： このオプションを設定する場合は十分注意してください。不注意から意図しなかったファイルを指定して CREATE DATABASE 文を実行すると、そのファイルの内容は上書きされます。

データベースごとに、少なくとも 2 つの制御ファイルを別々の物理ディスク・ドライブに格納して使用することをお薦めします。

> **関連項目：** 第 6 章「制御ファイルの管理」

データベース・ブロック・サイズの指定

データベースの標準ブロック・サイズは、DB_BLOCK_SIZE 初期化パラメータで指定します。標準ブロック・サイズは SYSTEM 表領域で使用され、その他の表領域ではデフォルトとして使用されます。Oracle は、最大 4 つの非標準ブロック・サイズをサポートします。

DB_BLOCK_SIZE 初期化パラメータ

標準ブロック・サイズには、最も一般的に使用するブロック・サイズを選択します。多くの場合、設定が必要なブロック・サイズは標準ブロック・サイズのみです。通常、DB_BLOCK_SIZE は 4K または 8K に設定します。パラメータを指定しないとオペレーティング・システム固有のデフォルト・データ・ブロック・サイズが使用されますが、ほとんどの場合、このデフォルト・ブロック・サイズで十分です。

データベースの作成後は、データベースを再作成する以外にブロック・サイズを変更する手段はありません。データベースのブロック・サイズがオペレーティング・システムのブロック・サイズと異なる場合は、データベースのブロック・サイズをオペレーティング・システムのブロック・サイズの倍数にする必要があります。

たとえば、使用しているオペレーティング・システムのブロック・サイズが 2KB（2048 バイト）である場合、次の DB_BLOCK_SIZE 初期化パラメータの設定は有効です。

```
DB_BLOCK_SIZE=4096
```

オペレーティング・システムのブロック・サイズより大きいブロック・サイズを指定できます。データ・ブロック・サイズを大きくすると、ディスクとメモリーの I/O（データのアクセスと格納）の効率が向上します。このケースに該当するのは、次のような使用例です。

- Oracle が大容量メモリーと高速ディスク・ドライブを装備した大型コンピュータ・システム上にある場合。たとえば、莫大なハードウェア資源を有するメインフレーム・コンピュータによって制御されるデータベースは、通常 4KB 以上のデータ・ブロック・サイズを使用します。
- Oracle が動作するオペレーティング・システムのブロック・サイズが小さい場合。たとえば、オペレーティング・システムのブロック・サイズが 1KB で、この値がデフォルトのデータ・ブロック・サイズと一致する場合、Oracle は通常の処理で過度のディスク I/O を実行している可能性があります。この場合にパフォーマンスを最高にするには、データベース・ブロックを複数のオペレーティング・システム・ブロックから構成する必要があります。

関連項目： デフォルトのブロック・サイズの詳細は、オペレーティング・システム固有の Oracle マニュアルを参照してください。

非標準ブロック・サイズ

CREATE TABLESPACE 文で BLOCKSIZE 句を指定すると、非標準のブロック・サイズを持つ表領域を作成できます。これらの非標準ブロック・サイズには、2K から 32K の間にある 2 の累乗、つまり 2K、4K、8K、16K、32K のいずれかを指定します。最大ブロック・サイズに関するプラットフォーム固有の制限が適用されるので、プラットフォームによっては、これらのサイズの一部は指定できない場合があります。

非標準ブロック・サイズを使用する場合は、使用するすべての非標準ブロック・サイズについて、SGA メモリーのバッファ・キャッシュ領域内にサブキャッシュを構成する必要があり

ます。これらのサブキャッシュの構成に使用する初期化パラメータについては、次の項の「SGA のサイズに影響する初期化パラメータの設定」を参照してください。

データベースに複数のブロック・サイズを指定できる機能は、特にデータベース間で表領域をトランスポートする場合に役立ちます。たとえば、OLTP 環境から、8KB の標準ブロック・サイズを使用するデータ・ウェアハウス環境に、4KB のブロック・サイズを使用する表領域をトランスポートできます。

関連項目：

- 11-4 ページ「表領域の作成」
- 11-34 ページ「データベース間での表領域のトランスポート」

SGA のサイズに影響する初期化パラメータの設定

ここで説明する初期化パラメータは、SGA に割り当てられるメモリーの量に影響を及ぼします。SGA_MAX_SIZE 以外の初期化パラメータは、ALTER SYSTEM 文で値を変更できる動的なパラメータです。SGA のサイズは動的で、これらのパラメータを動的に変更することにより、拡大または縮小できます。

注意： SGA の動的コンポーネント用のメモリーは、グラニュル単位で割り当てられます。グラニュル・サイズは、合計 SGA サイズにより決定されます。一般に、ほとんどのプラットフォームでは、合計 SGA サイズが 128MB 以下であればグラニュル・サイズは 4MB で、それ以外の場合は 16MB です。

ただし、一部にプラットフォーム依存性が存在する場合があります。たとえば、32 ビットの Windows NT の場合、128MB より大きい SGA のグラニュル・サイズは 8MB です。詳細は、使用しているオペレーティング・システム固有のマニュアルを参照してください。

V$SGA_DYNAMIC_COMPONENTS ビューを問い合せると、1 つのインスタンスで使用中のグラニュル・サイズを確認できます。SGA のすべての動的コンポーネントに、同じグラニュル・サイズが使用されます。

コンポーネントに対してグラニュル・サイズの倍数でないサイズを指定すると、そのサイズは最も近い倍数に繰り上げられます。たとえば、グラニュル・サイズが 4MB の場合に DB_CACHE_SIZE を 10MB として指定すると、実際には 12MB が割り当てられます。

SGA の動的コンポーネントに関するサマリー情報は、V$SGA_DYNAMIC_COMPONENTS ビューで確認できます。進行中の SGA サイズ変更操作に関する情報は V$SGA_CURRENT_RESIZE_OPS ビューに表示され、最後に完了した 100 件の SGA サイズ変更操作に関する情報は V$SGA_RESIZE_OPS ビューに表示されます。将来の動的な SGA のサイズ変更操作に

使用可能な SGA メモリー容量を調べるには、V$SGA_DYNAMIC_FREE_MEMORY ビューを問い合せます。

関連項目：

- SGA コンポーネントの監視とチューニングの詳細は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。
- SGA のサイズの監視に使用する動的パフォーマンス・ビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。
- SGA の概要は、『Oracle9*i* データベース概要』を参照してください。

SGA のサイズの制限

インスタンスの存続期間中の SGA の最大サイズは、SGA_MAX_SIZE 初期化パラメータによって決まります。バッファ・キャッシュ、共有プールおよびラージ・プールのサイズに影響を与える初期化パラメータは動的に変更できますが、これらのサイズと SGA の他の構成要素（固定 SGA、可変 SGA および REDO ログ・バッファ）のサイズとの合計が、SGA_MAX_SIZE で指定された値を超えるような変更はできません。

SGA_MAX_SIZE が指定されていない場合は、初期化時に指定またはデフォルト設定された構成要素すべての合計がデフォルト値として設定されます。

バッファ・キャッシュ初期化パラメータの設定

SGA の構成要素であるバッファ・キャッシュのサイズは、バッファ・キャッシュ初期化パラメータによって決まります。これらのパラメータを使用して、データベースで使用される各ブロック・サイズのキャッシュ・サイズを指定します。これらの初期化パラメータはすべて動的です。

データベースで複数のブロック・サイズを使用する場合は、DB_CACHE_SIZE と、少なくとも 1 つの DB_*n*K_CACHE_SIZE パラメータを設定する必要があります。Oracle は、DB_CACHE_SIZE パラメータに適切なデフォルト値を割り当てますが、DB_*n*K_CACHE_SIZE パラメータはデフォルトで 0（ゼロ）に設定され、追加のブロック・サイズ・キャッシュは構成されません。

バッファ・キャッシュのサイズはパフォーマンスに影響を及ぼします。一般に、キャッシュ・サイズを大きくすると、ディスクの読取りと書込みの回数が少なくなります。ただし、キャッシュを大きくすると、メモリーを過度に消費してページングやスワッピングが発生する可能性があります。

DB_CACHE_SIZE 初期化パラメータ DB_CACHE_SIZE 初期化パラメータは、旧リリースで使用されていた DB_BLOCK_BUFFERS 初期化パラメータを置換したものです。DB_CACHE_SIZE 初期化パラメータは、標準ブロック・サイズ・バッファのキャッシュ・サイズを指定します。標準ブロック・サイズは、DB_BLOCK_SIZE で指定されます。

下位互換性を維持するため、DB_BLOCK_BUFFERS パラメータもまだ機能しますが、これは静的パラメータのままであり、動的なサイズ設定パラメータと組み合せることはできません。

DB_*n*K_CACHE_SIZE 初期化パラメータ 非標準ブロック・サイズ・バッファのサイズと数は、次の初期化パラメータによって指定します。

- DB_2K_CACHE_SIZE
- DB_4K_CACHE_SIZE
- DB_8K_CACHE_SIZE
- DB_16K_CACHE_SIZE
- DB_32K_CACHE_SIZE

各パラメータは、対応するブロック・サイズのバッファ・キャッシュ・サイズを指定します。たとえば、次のように入力します。

```
DB_BLOCK_SIZE=4096

DB_CACHE_SIZE=12M
DB_2K_CACHE_SIZE=8M
DB_8K_CACHE_SIZE=4M
```

この例では、パラメータの指定によってデータベースの標準ブロック・サイズは 4KB になります。標準ブロック・サイズ・バッファのキャッシュ・サイズは 12MB になります。また、2KB および 8KB のキャッシュ・サイズがそれぞれ 8MB と 4MB で構成されます。

注意： これらのパラメータを使用して、標準ブロック・サイズのバッファ・キャッシュ・サイズを指定することはできません。たとえば、DB_BLOCK_SIZE の値が 2K の場合に DB_2K_CACHE_SIZE を設定しても無効です。標準ブロック・サイズのキャッシュ・サイズは、常に DB_CACHE_SIZE の値によって決まります。

共有プールのサイズの調整

SHARED_POOL_SIZE 初期化パラメータは、SGA の構成要素である共有プールのサイズを指定または調整する動的なパラメータです。Oracle によって適切なデフォルト値が選択されます。

ラージ・プールのサイズの調整

LARGE_POOL_SIZE 初期化パラメータは、SGA の構成要素であるラージ・プールのサイズを指定または調整する動的なパラメータです。Oracle によって適切なデフォルト値が選択されます。

最大プロセス数の指定

Oracle に同時に接続できるオペレーティング・システム・プロセスの最大数は、PROCESSES 初期化パラメータによって決まります。このパラメータには、6 以上（バックグラウンド・プロセスに対して 5、ユーザー・プロセスごとに 1）の値を指定する必要があります。たとえば、同時に 50 ユーザーが実行できるよう計画している場合は、このパラメータを最低でも 55 に設定します。

UNDO 領域管理方法の指定

Oracle データベースでは、データベースの変更をロールバックまたは取り消すために使用する情報の管理方法が必要とされます。これらの情報は、主にコミットされる前のトランザクションの処理レコードから構成されます。Oracle では、これらのレコードを総称して **UNDO** と呼びます。UNDO は、UNDO 表領域またはロールバック・セグメントに格納できます。

関連項目： 第 13 章「UNDO 領域の管理」

UNDO_MANAGEMENT 初期化パラメータ

UNDO_MANAGEMENT 初期化パラメータは、インスタンスを自動 UNDO 管理モード（UNDO が UNDO 表領域に格納される）と手動 UNDO 管理モード（UNDO がロールバック・セグメントに格納される）のどちらで起動するかを指定します。値として AUTO を指定すると自動 UNDO 管理モード、MANUAL を指定すると手動 UNDO 管理モードがそれぞれ有効になります。下位互換性を維持するために、デフォルトは MANUAL になっています。

UNDO_TABLESPACE 初期化パラメータ

インスタンスを自動 UNDO 管理モードで起動すると、UNDO を格納するために、インスタンス内で最初に使用可能になった UNDO 表領域が選択されます。UNDO_MANAGEMENT 初期化パラメータを AUTO に設定して CREATE DATABASE 文を実行すると、SYS_UNDOTS という名前のデフォルト UNDO 表領域が自動的に作成されます。これは、通常、データベースを起動したときに Oracle が必ず選択する UNDO 表領域です。

必要に応じて、UNDO_TABLESPACE 初期化パラメータを指定できます。このパラメータを指定すると、インスタンスは、パラメータで指定された UNDO 表領域を使用します。Oracle Real Application Clusters 環境で UNDO_TABLESPACE パラメータを使用すると、インスタンスに特定の UNDO 表領域を割り当てることができます。

使用可能な UNDO 表領域がない場合でもインスタンスは起動しますが、その場合は SYSTEM ロールバック・セグメントが使用されます。通常の環境では、これはお薦めできません。アラート・ファイルには、システムが UNDO 表領域のない状態で稼働していることを伝える警告メッセージが書き込まれます。

オラクル社は、ロールバック・セグメントよりも UNDO 表領域の使用をお薦めします。UNDO 表領域のほうが、管理が容易であり、UNDO 保存期間を明示的に設定できるためです。

ROLLBACK_SEGMENTS 初期化パラメータ

ROLLBACK_SEGMENTS パラメータは、データベースが手動 UNDO 管理モードで動作する場合、Oracle インスタンスがデータベース起動時に取得するシステム・ロールバック・セグメント以外のロールバック・セグメントのリストです。このパラメータには、値としてロールバック・セグメントのリストを指定します。ロールバック・セグメントを指定しない場合は、システム・ロールバック・セグメントが使用されます。

ROLLBACK_SEGMENTS 初期化パラメータは、下位互換性を維持するためにサポートされています。オラクル社は、ロールバック・セグメントよりも UNDO 表領域の使用をお薦めします。

ライセンスに関するパラメータの設定

注意： Oracle では、同時セッションの数によるライセンス提供がなくなりました。したがって、LICENSE_MAX_SESSIONS および LICENSE_SESSIONS_WARNING 初期化パラメータは廃止になっており、このマニュアルでは説明しません。

指名ユーザー・ライセンスを使用している場合、Oracle ではこの形式のライセンスを施行できます。データベース内に作成するユーザーの数に対して、制限を設定できます。この制限に達すると、それ以上のユーザーは作成できません。

注意： このメカニズムでは、データベースにアクセスする人がそれぞれ一意のユーザー名を持ち、ユーザー名を共有していないものと想定しています。したがって、名前付きユーザー・ライセンスによって Oracle のライセンス契約に従っていることを保証できるように、複数のユーザーが同じ名前を使用してログインすることを許可しないでください。

データベース内に作成できるユーザー数を制限するには、次の例に示すように、そのデータベースの初期化パラメータ・ファイルに LICENSE_MAX_USERS 初期化パラメータを設定します。

```
LICENSE_MAX_USERS = 200
```

サーバー・パラメータ・ファイルを使用した初期化パラメータの管理

Oracle ではこれまで、テキスト形式の初期化パラメータ・ファイルに初期化パラメータを格納していました。Oracle9*i* からは、バイナリ形式のサーバー・パラメータ・ファイルで初期化パラメータをメンテナンスすることを選択できます。

ここでは、サーバー・パラメータ・ファイルの概要を示し、各パラメータ格納方式を使用した初期化パラメータの管理方法について説明します。この項の内容は、次のとおりです。

- サーバー・パラメータ・ファイルの概要
- サーバー・パラメータ・ファイルへの移行
- サーバー・パラメータ・ファイルの作成
- SPFILE 初期化パラメータ
- ALTER SYSTEM を使用した初期化パラメータ値の変更
- サーバー・パラメータ・ファイルのエクスポート
- サーバー・パラメータ・ファイルのバックアップの作成
- サーバー・パラメータ・ファイルのエラーおよびリカバリ
- パラメータ設定の表示

サーバー・パラメータ・ファイルの概要

サーバー・パラメータ・ファイル（SPFILE）は、Oracle データベースが稼働するマシンで管理される初期化パラメータのリポジトリと考えることができます。これは、サーバー側初期化パラメータ・ファイルとして設計されています。サーバー・パラメータ・ファイルに格納された初期化パラメータは永続的で、インスタンスの実行中に行ったパラメータの変更は、インスタンスを停止し、起動しても有効です。これにより、`ALTER SYSTEM` 文による変更を持続させるために、初期化パラメータを手動で更新する必要がなくなります。また、Oracle データベースによる自己チューニングの基礎ともなります。

最初のサーバー・パラメータ・ファイルは、`CREATE SPFILE` 文を使用して、従来のテキスト初期化パラメータ・ファイルから作成します。このファイルは、テキスト・エディタで表示または編集できないバイナリ・ファイルです。パラメータの設定を表示および変更するために、他のインタフェースが用意されています。

注意： テキスト・エディタでバイナリ形式のサーバー・パラメータ・ファイルをオープンし、そのテキストを表示することは可能ですが、手動で編集しないでください。手動で編集すると、ファイルが破損します。また、インスタンスが起動できなくなり、たとえインスタンスが起動しても失敗します。

システム起動時に STARTUP コマンドが行うデフォルトの動作は、サーバー・パラメータ・ファイルを読み込んで、初期化パラメータの設定を取得することです。PFILE 句を指定せずに STARTUP コマンドを実行すると、オペレーティング・システム固有の位置からサーバー・パラメータ・ファイルが読み込まれます。従来のテキスト・パラメータ・ファイルを使用する場合は、STARTUP コマンドの発行時に必ず PFILE 句を指定します。サーバー・パラメータ・ファイルを使用してインスタンスを起動する方法の詳細は、4-2 ページの「データベースの起動」を参照してください。

サーバー・パラメータ・ファイルへの移行

現在、従来の初期化パラメータ・ファイルを使用している場合は、次の手順に従ってサーバー・パラメータ・ファイルに移行します。

1. 初期化パラメータ・ファイルがクライアント・マシン上にある場合は、FTP などを使用して、クライアント・マシンからサーバー・マシンにファイルを転送してください。

注意： Oracle9*i* Real Application Clusters を使用している場合は、インスタンス固有の初期化パラメータ・ファイルをすべて結合して、1 つの初期化パラメータ・ファイルを作成する必要があります。この操作の方法や、Oracle Real Application Clusters インスタンスでサーバー・パラメータ・ファイルを使用する際の固有の処理については、次のマニュアルを参照してください。

- 『Oracle9*i* Real Application Clusters セットアップおよび構成』
- 『Oracle9*i* Real Application Clusters 管理』

---

2. CREATE SPFILE 文を使用して、サーバー・パラメータ・ファイルを作成します。この文を実行すると、初期化パラメータ・ファイルが読み込まれて、サーバー・パラメータ・ファイルが作成されます。CREATE SPFILE 文を発行するために、データベースを起動する必要はありません。
3. 新しく作成したサーバー・パラメータ・ファイルを使用して、インスタンスを起動します。

サーバー・パラメータ・ファイルの作成

最初のサーバー・パラメータ・ファイルは、従来のテキスト初期化パラメータ・ファイルから作成する必要があります。必ず、STARTUP コマンドで使用する前に作成します。サーバー・パラメータ・ファイルを作成するには、CREATE SPFILE 文を使用します。この文を実行するには、SYSDBA または SYSOPER システム権限が必要です。

次の例では、初期化パラメータ・ファイル /u01/oracle/dbs/init.ora からサーバー・パラメータ・ファイルを作成しています。この例では SPFILE の名前を指定していないので、プラットフォーム固有のデフォルトの場所に、spfile$ORACLE_SID.ora という名前でファイルが作成されます。

```
CREATE SPFILE FROM PFILE='/u01/oracle/dbs/init.ora';
```

次の例では、名前を指定してサーバー・パラメータ・ファイルを作成しています。

```
CREATE SPFILE='/u01/oracle/dbs/test_spfile.ora'
       FROM PFILE='/u01/oracle/dbs/test_init.ora';
```

サーバー・パラメータ・ファイルは、必ずデータベースが稼働しているマシン上に作成されます。サーバー上に同じファイル名のサーバー・パラメータ・ファイルがすでに存在する場合は、新しい情報で上書きされます。

サーバー・パラメータ・ファイルの名前と場所には、データベースによるデフォルト設定を使用することをお薦めします。これにより、データベースの管理が容易になります。たとえば、STARTUP コマンドはこのデフォルトの場所を想定して、パラメータ・ファイルを読み込みます。

初期化パラメータ・ファイルからサーバー・パラメータ・ファイルを作成すると、初期化パラメータ・ファイル内のパラメータ設定と同じ行に記述されているコメントもサーバー・パラメータ・ファイルで管理されます。他のコメントはすべて無視されます。

CREATE SPFILE 文は、インスタンスの起動前後に実行できます。ただし、すでにサーバー・パラメータ・ファイルを使用してインスタンスを起動している場合に、インスタンスで現在使用されているのと同じサーバー・パラメータ・ファイルを再作成しようとすると、エラーが発生します。

注意： DBCA を使用してデータベースを作成した場合は、サーバー・パラメータ・ファイルが自動的に作成されます。

SPFILE 初期化パラメータ

SPFILE 初期化パラメータには、現在のサーバー・パラメータ・ファイルの名前を指定します。サーバーでデフォルトのサーバー・パラメータ・ファイルが使用されている場合（つまり、PFILE を指定せずに STARTUP コマンドを発行した場合）、SPFILE の値はサーバーによって内部的に設定されます。SQL*Plus コマンドの SHOW PARAMETERS SPFILE（またはパラメータの値を問い合せる他の方法）を使用すると、現在使用中のサーバー・パラメータ・ファイルの名前が表示されます。

従来のパラメータ・ファイルでも、使用するサーバー・パラメータ・ファイルを指定するために、SPFILE パラメータを設定できます。SPFILE パラメータは、デフォルト以外の場所にあるサーバー・パラメータ・ファイルを指定する際に使用します。従来の初期化パラメータ・ファイルで、サーバー・パラメータ・ファイルを設定するために IFILE 初期化パラメータを使用しないでください。かわりに、SPFILE パラメータを使用してください。次の操作の詳細は、4-2 ページの「データベースの起動」を参照してください。

- 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
- SPFILE パラメータを使用して、インスタンス起動時に使用するサーバー・パラメータ・ファイルの名前を指定する方法

ALTER SYSTEM を使用した初期化パラメータ値の変更

ALTER SYSTEM 文を使用すると、初期化パラメータ値を設定、変更または削除（デフォルト値へのリストア）できます。ALTER SYSTEM 文を使用して、従来の初期化パラメータ・ファイルのパラメータ設定を変更した場合は、現行のインスタンスのみに変更が適用されます。これは、ディスク上の初期化パラメータを自動的に更新するメカニズムがないためです。以後起動するインスタンスにも同じパラメータを渡すためには、ディスク上の初期化パラメータを手動で更新する必要があります。サーバー・パラメータ・ファイルを使用している場合は、このような制限はありません。

初期化パラメータ値の設定または変更

初期化パラメータ値を設定または変更するには、ALTER SYSTEM 文で SET 句を指定します。また、次の表のように、変更の適用範囲を指定する場合は、SCOPE 句を指定します。

SCOPE 句	説明
SCOPE = SPFILE	サーバー・パラメータ・ファイルのみに変更が適用されます。次のような効果があります。 ■ 動的パラメータの場合は、次の起動時に変更が有効になり、以後持続します。 ■ 静的パラメータの場合も、動的パラメータと同じように動作します。静的パラメータで使用できる SCOPE 指定はこれだけです。

SCOPE 句	説明
SCOPE = MEMORY	メモリーのみに変更が適用されます。次のような効果があります。 ■ 動的パラメータの場合は、変更が即時に有効になりますが、サーバー・パラメータ・ファイルは更新されないので、変更は持続しません。 ■ 静的パラメータでは、この指定は使用できません。
SCOPE = BOTH	サーバー・パラメータ・ファイルとメモリーの両方に変更が適用されます。次のような効果があります。 ■ 動的パラメータの場合は、変更が即時に有効になり、持続します。 ■ 静的パラメータでは、この指定は使用できません。

サーバーがサーバー・パラメータ・ファイルを使用していない場合に SCOPE=SPFILE または SCOPE=BOTH を指定すると、エラーが発生します。デフォルトは、インスタンス起動時にサーバー・パラメータ・ファイルを使用した場合は SCOPE=BOTH、従来の初期化パラメータ・ファイルを使用した場合は MEMORY になります。

動的パラメータの場合は、DEFERRED キーワードを指定することもできます。このキーワードを指定すると、これから確立するセッションでのみ変更が有効になります。

COMMENT 句を使用すると、パラメータの更新にコメント文字列を対応付けることができます。SCOPE を SPFILE または BOTH に指定した場合に、サーバー・パラメータ・ファイルにコメントが記述されます。

次の文は、インスタンスで許可するジョブ・キュー・プロセスの最大数を変更します。また、コメントを指定し、変更をメモリーにのみ適用する（つまり、インスタンスを停止し、起動すると変更は無効になる）ことを明示します。

```
ALTER SYSTEM SET JOB_QUEUE_PROCESSES=50
                 COMMENT='temporary change on Nov 29'
                 SCOPE=MEMORY;
```

次の例では、文字列のリストをとる複雑な初期化パラメータの設定を説明しています。値を設定するパラメータは、LOG_ARCHIVE_DEST_n 初期化パラメータです。この例では、既存のパラメータが新しい値に変更されるか、または新しいアーカイブ先が追加されます。

```
ALTER SYSTEM
     SET LOG_ARCHIVE_DEST_4='LOCATION=/u02/oracle/rbdb1/',MANDATORY,'REOPEN=2'
         COMMENT='Add new destimation on Nov 29'
         SCOPE=SPFILE;
```

値が文字列のリストからなる場合、ALTER SYSTEM SET 文の構文では、位置または序数によるリストの各要素の編集をサポートしていません。パラメータを更新するたびに、完全な値のリストを指定する必要があります。これにより、新しいリストで古いリストが完全に置換されます。

初期化パラメータ値の削除

文字列値を持つ初期化パラメータでは、次の構文を使用することにより、パラメータをデフォルト値にリストア（実質的には削除）できます。

```
ALTER SYSTEM SET parameter = '';
```

数値パラメータまたはブール値パラメータでは、元のデフォルト値に戻すようにパラメータを設定する必要があります。

サーバー・パラメータ・ファイルのエクスポート

従来のテキスト形式の初期化パラメータ・ファイルを作成するために、サーバー・パラメータ・ファイルをエクスポートできます。これを行うのは、次のような場合です。

- サーバー・パラメータ・ファイルのバックアップを作成する場合。
- 診断のために、現在インスタンスで使用しているすべてのパラメータのリストを作成する場合。これは、SQL*Plus の SHOW PARAMETERS コマンド、あるいは V$PARAMETER または V$PARAMETER2 ビューの選択と同じです。
- サーバー・パラメータ・ファイルをエクスポートし、出力ファイルを編集して、サーバー・パラメータ・ファイルを再作成するという手順で、サーバー・パラメータ・ファイルを変更する場合。

PFILE オプションにエクスポート・ファイルを指定して、インスタンスを起動することもできます。

サーバー・パラメータ・ファイルをエクスポートするには、CREATE PFILE 文を使用します。この文を実行するには、SYSDBA または SYSOPER システム権限が必要です。エクスポート・ファイルは、データベース・サーバー・マシン上に作成されます。パラメータ設定と同じ行に記述されている、パラメータに関するコメントがすべて含まれます。

次の例では、サーバー・パラメータ・ファイルからテキスト形式の初期化パラメータ・ファイルを作成しています。

```
CREATE PFILE FROM SPFILE;
```

この例ではファイル名を指定していないので、プラットフォーム固有のデフォルト・サーバー・パラメータ・ファイルから、プラットフォーム固有の名前で初期化パラメータ・ファイルが作成されます。

次の例では、ファイル名を指定して、サーバー・パラメータ・ファイルからテキスト形式の初期化パラメータ・ファイルを作成しています。

```
CREATE PFILE='/u01/oracle/dbs/test_init.ora'
       FROM SPFILE='/u01/oracle/dbs/test_spfile.ora';
```

サーバー・パラメータ・ファイルのバックアップの作成

サーバー・パラメータ・ファイルをエクスポートして、そのバックアップを作成できます。詳細は、2-48 ページの「サーバー・パラメータ・ファイルのエクスポート」を参照してください。データベースのバックアップおよびリカバリ計画が Recovery Manager を使用して実装されている場合は、Recovery Manager を使用してバックアップを作成できます。サーバー・パラメータ・ファイルのバックアップは、データベースのバックアップ作成時に自動的に作成されますが、Recovery Manager を使用すると現在アクティブになっているサーバー・パラメータ・ファイルのバックアップを作成できます。

関連項目：『Oracle9*i* Recovery Manager ユーザーズ・ガイド』

サーバー・パラメータ・ファイルのエラーおよびリカバリ

起動時やエクスポート操作時にサーバー・パラメータ・ファイルの読込みでエラーが発生した場合、または作成時にサーバー・パラメータ・ファイルの書込みでエラーが発生した場合は、エラーが出力されて操作が終了します。

パラメータの更新時にサーバー・パラメータ・ファイルの読込みまたは書込みでエラーが発生した場合は、アラート・ファイルにエラーが記録されて、サーバー・パラメータ・ファイルに対する後続のパラメータ更新がすべて無視されます。この時点では、次の方法を選択できます。

- インスタンスを停止し、サーバー・パラメータ・ファイルをリカバリしてからインスタンスを再起動する。
- 後続のパラメータ更新を持続的にするための処置を講じずに、実行を継続する。

パラメータ設定の表示

パラメータ設定を表示するには、いくつかの方法があります。

方法	説明
SHOW PARAMETERS	この SQL*Plus コマンドを使用すると、現在使用中のパラメータ値が表示されます。
CREATE PFILE	この SQL 文は、バイナリ形式のサーバー・パラメータ・ファイルからテキスト形式の初期化パラメータ・ファイルを作成します。
V$PARAMETER	このビューを使用すると、現在有効なパラメータ値が表示されます。
V$PARAMETER2	このビューを使用すると、現在有効なパラメータ値が表示されます。各リスト・パラメータ値が行として出力されるので、このビューのほうがリスト・パラメータ値を容易に識別できます。
V$SPPARAMETER	このビューを使用すると、サーバー・パラメータ・ファイルの現在の内容が表示されます。インスタンスでサーバー・パラメータ・ファイルが使用されていない場合は、NULL が返されます。

関連項目： ビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

3

Oracle Managed Files の使用

この章では、Oracle Managed Files の使用方法について説明します。この章の内容は、次のとおりです。

- Oracle Managed Files の概要
- Oracle Managed Files の作成および使用の有効化
- Oracle Managed Files の作成
- Oracle Managed Files の動作
- Oracle Managed Files の使用例

Oracle Managed Files の概要

Oracle Managed Files を使用すると、Oracle データベースの管理が簡単になります。Oracle データベースを構成するオペレーティング・システム・ファイルをデータベース管理者（DBA）が直接管理する必要はありません。ファイル名ではなくデータベース・オブジェクトによって操作を指定します。Oracle は内部的に標準ファイル・システム・インタフェースを使用し、必要に応じて、次のデータベース構造のファイルを作成および削除します。

- 表領域
- オンライン REDO ログ・ファイル
- 制御ファイル

初期化パラメータによって、特定のタイプのファイルに使用するファイル・システム・ディレクトリを指定します。これにより、一意の Oracle Managed Files が作成され、不要になると削除されます。

この機能は、トレース・ファイル、監査ファイル、アラート・ファイルおよびコア・ファイルなどの管理ファイルの作成および命名には影響を与えません。

Oracle Managed Files の使用対象

Oracle Managed Files は、次のタイプのデータベースに適しています。

- 次のものによってサポートされるデータベース
  - ストライプ化 /RAID および動的に拡張可能な論理ボリュームをサポートする論理ボリューム・マネージャ
  - サイズが大きく拡張可能なファイルを提供するファイル・システム
- ローエンド・データベースまたはテスト・データベース

Oracle Managed Files の目的は、RAW ディスクを使用したシステムの管理を容易にすることではありません。この機能は、ディスク領域割当てについて、オペレーティング・システム機能との統合を実現します。オペレーティング・システムは RAW ディスクの割当てをサポートしていない（割当ては手動で行う）ので、この機能は利用できません。その一方で、Oracle Managed Files では（RAW ディスクではなく）必ずオペレーティング・システムのファイル・システムを使用する必要があるため、ディスク上のファイルの配置方法を DBA が管理できず、一部の I/O をチューニングできなくなります。

論理ボリューム・マネージャ（LVM）の概要

LVM は、ほとんどのオペレーティング・システムで利用できるソフトウェア・パッケージです。論理ディスク・マネージャ（LDM）と呼ばれることもあります。LVM を使用すると、複数の物理ディスクの断片を、ソフトウェアの高層部で 1 つのディスクとして表される、連続した単一のアドレス空間に結合できます。LVM では、基盤となるどの物理ディスクよりも、容量、パフォーマンス、信頼性および可用性の特性に優れた論理ボリュームを作成できます。LVM は、ミラー化、ストライプ化、連結および RAID5 などの手法を使用して、これらの特性を実装します。

LVM の中には、論理ボリュームを作成後、そのボリュームの使用中に特性を変更できるものがあります。ボリュームはサイズ変更やミラー化できるだけでなく、別の物理ディスクに再配置することもできます。

ファイル・システムの概要

ファイル・システムとは、連続するディスク・アドレス空間内に構築されたデータ構造です。ファイル・マネージャ（FM）は、ファイル・システムを操作するソフトウェア・パッケージですが、これがファイル・システムと呼ばれる場合もあります。オペレーティング・システムには必ず FM が組み込まれています。FM の主要なタスクは、ファイル・システム内のファイルにディスク領域への割当てまたは割当て解除です。

ファイル・システムを使用すると、多数のファイルにディスク領域を割り当てることができます。各ファイルは、Oracle などのアプリケーションに連続するアドレス空間を提供するために作成されます。実際には、ファイル・システムのディスク領域の中でファイルは連続していない場合があります。ファイルは、作成、読取り、書込み、サイズ変更および削除ができます。各ファイルには対応する名前があり、ファイルを参照する際に使用します。

ファイル・システムは通常、LVM が作成する論理ボリュームの最上部に構築されます。したがって、特定のファイル・システム内にあるファイルはすべて、基盤となる論理ボリュームから継承された同じパフォーマンス、信頼性および可用性の特性を持ちます。ファイル・システムは、その中のすべてのファイルによって共有される、単一の記憶域のプールです。ファイル・システムの領域がなくなると、そのファイル・システム内にあるファイルを増やすことはできません。1 つのファイル・システムで使用可能な領域が、他のファイル・システムの領域に影響を及ぼすことはありません。ただし、LVM と FM の組合せによっては、ファイル・システムの領域を追加または削除できます。

オペレーティング・システムは、複数のファイル・システムをサポートできます。別々のファイルに異なる記憶特性を与える場合や、使用可能なディスク領域を分割して互いに影響を及ぼさないプールを作成する場合に、複数のファイル・システムが作成されます。

Oracle Managed Files の使用上の利点

Oracle Managed Files を使用すると、次のような利点があります。

- データベースの管理が容易になります。

  ファイル名を考えて、特定の記憶域要件を定義する必要はありません。一貫性のある一連のルールに基づいて、すべての関連ファイルが命名されます。記憶域の特性と記憶域を割り当てるプールは、ファイル・システムによって定義されます。

- 管理者による誤ったファイルの指定が原因で破損することが少なくなります。

  Oracle Managed Files とファイル名はすべて一意です。一般的によくある間違いは、2 つの異なるデータベースで同じファイルを使用することであり、これが長時間にわたるシステム・ダウンを引き起こし、コミット済みトランザクションが失われる原因となります。1 つのファイルを参照するために 2 つの異なる名前を使用することは、重大な破損の原因となるもう 1 つの間違いです。

- 不要なファイルの存在によるディスク領域の浪費が減少します。

  Oracle Managed Files が不要になったときは、古いファイルが自動的に削除されます。大規模なシステムでは、特定のファイルがまだ必要かどうか誰も確信できないという理由だけで、大量のディスク領域が浪費されています。Oracle Managed Files の削除機能は、ディスク上の不要ファイルの削除という管理タスクを容易にし、ファイルを誤って削除することを防止します。

- テスト・データベースおよび開発データベースを容易に作成できます。

  ファイル構造と命名について検討する時間を最小限にとどめることができ、実行するファイル管理タスクも従来より少なくて済みます。これにより、テスト・データベースまたは開発データベースの実際の要件を満たす作業に集中できます。

- 移植可能なサード・パーティ製ツールの開発が容易になります。

  Oracle Managed Files では、SQL スクリプト内でオペレーティング・システム固有のファイル名を指定する必要がありません。

Oracle Managed Files と既存の機能

Oracle Managed Files を使用しても、既存の機能が不要になるわけではありません。既存データベースは、常に従来どおり操作できます。古いファイルはそれまでの方法で管理し、その一方で新しいファイルは管理ファイルとして作成できます。したがって、データベースには Oracle Managed Files とそれ以外のファイルがともに存在する状態になります。

Oracle Managed Files の作成および使用の有効化

次の初期化パラメータを使用すると、データベースで Oracle Managed Files 機能を使用できます。

パラメータ	説明
DB_CREATE_FILE_DEST	作成操作でファイル仕様を指定しなかった場合に、Oracle によってデータ・ファイルまたは一時ファイルが作成される、デフォルトのファイル・システム・ディレクトリの場所を定義します。DB_CREATE_ONLINE_LOG_DEST_*n* を指定していない場合は、オンライン REDO ログ・ファイルおよび制御ファイルのデフォルトのファイル・システム・ディレクトリとしても使用されます。
DB_CREATE_ONLINE_LOG_DEST_*n*	作成操作でファイル仕様を指定しなかった場合に、オンライン REDO ログ・ファイルおよび制御ファイルが作成される、デフォルトのファイル・システム・ディレクトリの場所を定義します。この初期化パラメータは複数回使用できます。その際は、*n* にオンライン REDO ログ・ファイルまたは制御ファイルの多重コピーを指定します。多重コピーは最大 5 つ指定できます。

これらのパラメータで指定したファイル・システム・ディレクトリは、すでに存在している必要があります。Oracle はディレクトリを作成しません。ディレクトリには、Oracle によるファイル作成を可能にする権限が必要です。

ファイル作成操作で場所を明示的に指定しなかった場合は、必ずデフォルトの場所が使用されます。ファイル名は Oracle が作成するため、作成されたファイルは Oracle Managed Files になります。

これら 2 つの初期化パラメータはどちらも動的であり、ALTER SYSTEM または ALTER SESSION 文を使用して設定できます。

関連項目：

- 初期化パラメータの詳細は『Oracle9*i* データベース・リファレンス』を参照してください。
- 3-8 ページ「Oracle Managed Files の命名方法」

DB_CREATE_FILE_DEST 初期化パラメータの設定

初期化パラメータ・ファイルに DB_CREATE_FILE_DEST 初期化パラメータを設定して、データベースが次のファイルを作成するデフォルトの場所を識別できるようにします。

- データ・ファイル
- 一時ファイル
- オンライン REDO ログ・ファイル
- 制御ファイル

ファイル・システム・ディレクトリの名前を指定して、これらに対するオペレーティング・システム・ファイルを作成するためのデフォルトの場所にします。次の例では、Oracle Managed Files を作成する際に使用するデフォルトのディレクトリとして、/u01/oradata/payroll を設定しています。

```
DB_CREATE_FILE_DEST = '/u01/oradata/payroll'
```

DB_CREATE_ONLINE_LOG_DEST_*n* 初期化パラメータの設定

初期化パラメータ・ファイルに DB_CREATE_ONLINE_LOG_DEST_*n* 初期化パラメータを設定して、データベースが次のファイルを作成するデフォルトの場所を識別できるようにします。

- オンライン REDO ログ・ファイル
- 制御ファイル

ファイル・システム・ディレクトリの名前を指定して、これらに対するオペレーティング・システム・ファイルを作成するためのデフォルトの場所にします。多重コピーを配置する場所は最大 5 つまで指定できます。

オンライン REDO ログ・ファイルおよび制御ファイルを作成する場合のみ、このパラメータは、DB_CREATE_FILE_DEST 初期化パラメータで指定されているデフォルトの場所を上書きします。DB_CREATE_FILE_DEST パラメータを指定せず、このパラメータのみを指定している場合は、オンライン REDO ログ・ファイルと制御ファイルのみが Oracle Managed Files として作成されます。

少なくとも 2 つのパラメータを指定することをお薦めします。たとえば、次のように設定します。

```
DB_CREATE_ONLINE_LOG_DEST_1 = '/u02/oradata/payroll'
DB_CREATE_ONLINE_LOG_DEST_2 = '/u03/oradata/payroll'
```

これにより、ファイルを多重化できるので、オンライン REDO ログ・ファイルまたは制御ファイルの保存先の一方で障害が発生した場合のフォルト・トレランスが向上します。

Oracle Managed Files の作成

次の条件のいずれかを満たしている場合で、作成操作でファイル仕様を指定しなかったときに、必要に応じて Oracle Managed Files が作成されます。

- 初期化パラメータ・ファイルに、DB_CREATE_FILE_DEST および DB_CREATE_ONLINE_LOG_DEST_*n* 初期化パラメータの一方または両方を指定している場合
- DB_CREATE_FILE_DEST および DB_CREATE_ONLINE_LOG_DEST_*n* 初期化パラメータの一方または両方を動的に設定するために、ALTER SYSTEM または ALTER SESSION 文を発行した場合

Oracle Managed Files を作成する文がエラーを検出した場合、またはなんらかの障害のために完了しなかった場合は、その文によって作成された Oracle Managed Files はすべて、エラーまたは障害のリカバリの一部として自動的に削除されます。ただし、ファイル・システムや記憶域サブシステムで発生する多数の潜在的なエラーが原因で、オペレーティング・システムのコマンドを使用した手動でのファイル削除が必要になる場合があります。Oracle Managed Files が作成されると、アラート・ファイルにそのファイル名が書き込まれます。手動でのファイル削除が必要な場合は、この情報を使用してファイルを特定してください。

この項の内容は、次のとおりです。

- Oracle Managed Files の命名方法
- データベース作成時の Oracle Managed Files の作成
- 表領域用データ・ファイルの作成
- 一時表領域用一時ファイルの作成
- 制御ファイルの作成
- オンライン REDO ログ・ファイルの作成

関連項目：『Oracle9*i* SQL リファレンス』

Oracle Managed Files の命名方法

Oracle Managed Files のファイル名は、ファイル命名に関する Optimal Flexible Architecture（OFA）標準に準拠しています。割り当てられた名前は、次の要件を満たしています。

- データベース・ファイルが他のすべてのファイルと容易に区別できます。
- 制御ファイル、オンライン REDO ログ・ファイルおよびデータ・ファイルがファイル名で識別できます。
- 表領域に対するデータ・ファイルの関連が明確に示されています。

同じ名前を持つ Oracle Managed Files は 1 つもありません。Oracle Managed Files の作成に使用される名前は、次の 3 つのソースから構成されます。

- デフォルトのファイル・システム・ディレクトリの場所。
- ファイルのタイプに基づいて選択されたポート固有のファイル名テンプレート。
- Oracle データベースまたはオペレーティング・システムによって作成された一意の文字列。これにより、ファイル作成によって既存のファイルが破損しないこと、および誤って他のファイルが選択されないことが保証されます。

特定の例として、Solaris における Oracle Managed Files のファイル名の書式を次に示します。

ファイル・タイプ	書式	例
データ・ファイル	`o1_mf_%t_%u_.dbf`	`/u01/oradata/payroll/o1_mf_tbs1_2ixfh90q_.dbf`
一時ファイル	`o1_mf_%t_%u_.tmp`	`/u01/oradata/payroll/o1_mf_temp1_6dygh80r_.tmp`
REDO ログ・ファイル	`o1_mf_%g_%u_.log`	`/u01/oradata/payroll/o1_mf_1_wo94n2xi_.log`
制御ファイル	`o1_mf_%u_.ctl`	`/u01/oradata/payroll/o1_mf_cmr7t30p_.ctl`

各書式の意味は次のとおりです。

- %t は表領域の名前を表します。最大 8 文字の表領域名が使用されます。名前が長すぎる場合は、表領域の名前は切り捨てられます。一意の文字列の前に表領域名が付いているので、アルファベット順のファイル・リストでは、特定の表領域のデータ・ファイルがすべて連続して並びます。
- %u は一意性を保証する 8 文字の文字列を表します。
- %g はオンライン REDO ログ・ファイルのグループ番号を表します。

他のプラットフォームでもファイル名はほぼ同じですが、各プラットフォームのネーミング規則の制約を受けます。

注意： Oracle Managed Files の名前は変更しないでください。Oracle Managed Files は名前に基づいて識別されます。ファイル名を変更すると、Oracle では Oracle Managed Files として認識できなくなり、ファイルは適切に管理されません。

データベース作成時の Oracle Managed Files の作成

ここでは、Oracle Managed Files 使用時に、CREATE DATABASE 文を発行してデータベース構造を作成したときの動作について説明します。

データベース作成時の制御ファイルの指定

データベース作成時には、CONTROL_FILES 初期化パラメータによって指定されたファイルで、制御ファイルが作成されます。CONTROL_FILES パラメータが設定されておらず、Oracle Managed Files の作成に必要な初期化パラメータが少なくとも 1 つ設定されている場合は、制御ファイルのデフォルトの保存先に Oracle Managed Files の制御ファイルが作成されます。デフォルトの保存先は、次の優先順位に従って定義されます。

- DB_CREATE_ONLINE_LOG_DEST_*n* 初期化パラメータが設定されている場合は、指定された各ディレクトリに Oracle Managed Files の制御ファイルのコピーが作成されます。最初のディレクトリに作成されたファイルが主制御ファイルになります。
- DB_CREATE_FILE_DEST 初期化パラメータが設定されており、DB_CREATE_ONLINE_LOG_DEST_*n* 初期化パラメータが設定されていない場合は、指定されたディレクトリに Oracle Managed Files の制御ファイルが作成されます。

CONTROL_FILES パラメータが設定されておらず、前述のどの初期化パラメータも設定されていない場合、Oracle のデフォルトの動作はオペレーティング・システムによって異なります。少なくとも 1 つの制御ファイルのコピーが、オペレーティング・システム固有のデフォルトの場所に作成されます。この方法で作成された制御ファイルのコピーは Oracle Managed Files ではありません。そのため、初期化パラメータ・ファイルに CONTROL_FILES 初期化パラメータを追加する必要があります。

Oracle Managed Files の制御ファイルが作成された場合で、サーバー・パラメータ・ファイルが存在するときは、サーバー・パラメータ・ファイルに CONTROL_FILES 初期化パラメータのエントリが追加されます。サーバー・パラメータ・ファイルが存在しない場合は、CONTROL_FILES 初期化パラメータのエントリをテキスト形式の初期化パラメータ・ファイルに手動で追加する必要があります。

関連項目： 第 6 章「制御ファイルの管理」

データベース作成時のオンライン REDO ログ・ファイルの指定

CREATE DATABASE 文で LOGFILE 句は必須でなく、単純にこれを省略すると Oracle Managed Files のオンライン REDO ログ・ファイルが作成されます。LOGFILE 句を省略すると、デフォルトのオンライン REDO ログ・ファイルの保存先にオンライン REDO ログ・ファイルが作成されます。デフォルトの保存先は、次の優先順位に従って定義されます。

- DB_CREATE_ONLINE_LOG_DEST_*n* 初期化パラメータが設定されている場合は、指定された各ディレクトリに 2 つのオンライン REDO ファイルが作成されます。より厳密に言うと、指定された各ディレクトリに対応するメンバーを持つ 2 つのオンライン REDO グループが作成されます。これらのオンライン REDO ログ・ファイルは Oracle Managed Files です。
- DB_CREATE_FILE_DEST 初期化パラメータが設定されており、DB_CREATE_ONLINE_LOG_DEST_*n* 初期化パラメータが設定されていない場合は、指定されたディレクトリに 2 つのオンライン REDO ログ・ファイル（それぞれ 1 つのメンバーを持つ 2 つのグループ）が作成されます。これらのオンライン REDO ログ・ファイルは Oracle Managed Files です。
- LOGFILE 句を省略し、前述のどの初期化パラメータも設定されていない場合は、オペレーティング・システム固有のデフォルトの場所に 2 つのオンライン REDO ログ・ファイルが作成されます。この方法で作成されたオンライン REDO ログ・ファイルは Oracle Managed Files ではありません。

Oracle Managed Files のオンライン REDO ログ・ファイルのデフォルト・サイズは 100MB です。

必要に応じて、ファイル名を省略した LOGFILE 句を指定することにより、デフォルトの属性を変更した Oracle Managed Files のオンライン REDO ログ・ファイルを作成できます。オンライン REDO ログ・ファイルは前述のとおりに作成されますが、次のような例外があります。CREATE DATABASE 文の LOGFILE 句にファイル名を指定せず、Oracle Managed Files の作成に必要な初期化パラメータが 1 つも設定されていない場合は、CREATE DATABASE 文が失敗します。

関連項目： 第 7 章「オンライン REDO ログの管理」

データベース作成時の SYSTEM 表領域用データ・ファイルの指定

CREATE DATABASE 文で DATAFILE 句は必須でなく、単純にこれを省略すると SYSTEM 表領域用の Oracle Managed Files のデータ・ファイルが作成されます。DATAFILE 句を省略すると、次の処理のいずれかが実行されます。

- DB_CREATE_FILE_DEST が設定されている場合は、SYSTEM 表領域用の Oracle Managed Files のデータ・ファイルが DB_CREATE_FILE_DEST ディレクトリに作成されます。
- DB_CREATE_FILE_DEST が設定されていない場合は、オペレーティング・システム固有の名前とサイズで、SYSTEM 表領域のデータ・ファイルが 1 つ作成されます。この方

法で作成された SYSTEM 表領域のデータ・ファイルは Oracle Managed Files ではありません。

Oracle Managed Files のデータ・ファイルのデフォルト・サイズは 100MB です。このファイルは自動拡張可能で、最大サイズに制限はありません。

必要に応じて、SYSTEM 表領域用の Oracle Managed Files のデータ・ファイルを作成し、デフォルトの属性を上書きできます。そのためには、ファイル名を省略し、上書きする属性を指定して、DATAFILE 句を指定します。ファイル名を指定せずに、DB_CREATE_FILE_DEST パラメータを設定すると、SYSTEM 表領域用の Oracle Managed Files のデータ・ファイルが DB_CREATE_FILE_DEST ディレクトリに作成され、指定した属性が上書きされます。ただし、ファイル名を指定せず、DB_CREATE_FILE_DEST パラメータを設定しないと、CREATE DATABASE 文が失敗します。

Oracle Managed Files のデフォルトの属性を上書きするときに、SIZE 値を指定しても AUTOEXTEND 句を指定しない場合、データ・ファイルは自動拡張可能になりません。

データベース作成時の UNDO 表領域データ・ファイルの指定

UNDO TABLESPACE 句の DATAFILE 副次句は必須でなく、ファイル仕様に必ずしもファイル名を指定する必要はありません。ファイル名を指定せず、DB_CREATE_FILE_DEST が設定されている場合は、Oracle Managed Files のデータ・ファイルが DB_CREATE_FILE_DEST ディレクトリに作成されます。DB_CREATE_FILE_DEST が設定されていない場合は、構文エラーで文が失敗します。

UNDO TABLESPACE 句自体は、CREATE DATABASE 文のオプションです。この句を指定せず、自動 UNDO 管理モードが使用可能な場合は、次のルールに従って、SYS_UNDOTBS という名前のデフォルトの UNDO 表領域が作成され、自動拡張可能な 10MB のデータ・ファイルが割り当てられます。

- DB_CREATE_FILE_DEST が設定されている場合は、指定されたディレクトリに Oracle Managed Files のデータ・ファイルが作成されます。
- DB_CREATE_FILE_DEST が設定されていない場合は、オペレーティング・システム固有のデフォルトの場所にデータ・ファイルが作成されます。

関連項目： 第 13 章「UNDO 領域の管理」

データベース作成時のデフォルト一時表領域用一時ファイルの指定

DEFAULT TEMPORARY TABLESPACE 句の TEMPFILE 副次句は必須でなく、ファイル仕様に必ずしもファイル名を指定する必要はありません。ファイル名を指定せず、DB_CREATE_FILE_DEST が設定されている場合は、Oracle Managed Files の一時ファイルが DB_CREATE_FILE_DEST ディレクトリに作成されます。DB_CREATE_FILE_DEST が設定されていない場合は、構文エラーで CREATE DATABASE 文が失敗します。

DEFAULT TEMPORARY TABLESPACE 句自体はオプションであり、この句を指定していない場合は、デフォルト一時表領域は作成されません。

Oracle Managed Files の一時ファイルのデフォルト・サイズは 100MB です。このファイルは自動的に拡張可能で、最大サイズに制限はありません。

Oracle Managed Files を使用した CREATE DATABASE 文の例

ここでは、Oracle Managed Files 機能を使用した CREATE DATABASE 文の例を示します。

CREATE DATABASE: 例 1

この例では、次の Oracle Managed Files を含むデータベースが作成されます。

- ディレクトリ /u01/oradata/sample の SYSTEM 表領域用データ・ファイル。サイズは 100MB で、無制限に自動拡張可能です。
- それぞれ 100MB のメンバーを 2 つ含む 2 つのオンライン・ログ・グループ。/u02/oradata/sample と /u03/oradata/sample に 1 つずつ作成されます。
- 自動 UNDO 管理モードが使用可能の場合は、ディレクトリ /u01/oradata/sample の UNDO 表領域用データ・ファイル。サイズは 10MB で、無制限に自動拡張可能です。SYS_UNDOTS という名前の UNDO 表領域が作成されます。
- CONTROL_FILES 初期化パラメータが指定されていない場合は、2 つの制御ファイルが /u02/oradata/sample と /u03/oradata/sample に 1 つずつ作成されます。/u02/oradata/sample の制御ファイルが主制御ファイルになります。

初期化パラメータ・ファイルで、次のパラメータを設定します。

```
DB_CREATE_FILE_DEST = '/u01/oradata/sample'
DB_CREATE_ONLINE_LOG_DEST_1 = '/u02/oradata/sample'
DB_CREATE_ONLINE_LOG_DEST_2 = '/u03/oradata/sample'
```

SQL プロンプトから次の文を発行します。

```
SQL> CREATE DATABASE sample;
```

CREATE DATABASE: 例 2

この例では、次の Oracle Managed Files を含むデータベースが作成されます。

- ディレクトリ /u01/oradata/sample2 の SYSTEM 表領域用データ・ファイル。サイズは 100MB で、無制限に自動拡張可能です。
- ディレクトリ /u01/oradata/sample2 の 2 つのオンライン REDO ログ・ファイル（各 100MB）。これらのファイルは多重化されていません。
- ディレクトリ /u01/oradata/sample2 の UNDO 表領域用データ・ファイル。サイズは 10MB で、無制限に自動拡張可能です。SYS_UNDOTS という名前の UNDO 表領域が作成されます。
- /u01/oradata/sample2 の 1 つの制御ファイル。

この例では、次のように想定されています。

- 初期化パラメータ・ファイルに DB_CREATE_ONLINE_LOG_DEST_*n* 初期化パラメータは 1 つも指定されていません。
- 初期化パラメータ・ファイルに CONTROL_FILES 初期化パラメータは指定されていません。
- 自動 UNDO 管理モードは使用可能になっています。

SQL プロンプトから次の文を発行します。

```
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST = '/u01/oradata/sample2';
SQL> CREATE DATABASE sample2;
```

このデータベース構成は、本番データベースにはお薦めしません。この例は、非常にローエンドのデータベースか、単純なテスト・データベースを簡単に作成する方法を示しています。このデータベースの耐障害性を高めるには、制御ファイルを少なくとももう 1 つ作成し、オンライン REDO ログを多重化する必要があります。

CREATE DATABASE: 例 3

この例では、デフォルト一時表領域および UNDO 表領域用の Oracle Managed Files のファイル・サイズを指定しています。次の Oracle Managed Files を持つデータベースが作成されます。

- ディレクトリ /u01/oradata/sample3 の SYSTEM 表領域用データ・ファイル（400MB）。SIZE が指定されているため、このファイルは自動拡張可能ではありません。
- それぞれ 100MB のメンバーを 2 つ含む 2 つのオンライン REDO ログ・グループ。/u02/oradata/sample3 と /u03/oradata/sample3 に 1 つずつ作成されます。
- デフォルト一時表領域 dflt_ts に対して、ディレクトリ /u01/oradata/sample3 の一時ファイル（10MB）。SIZE が指定されているため、このファイルは自動拡張可能ではありません。
- UNDO 表領域 undo_ts に対して、ディレクトリ /u01/oradata/sample3 のデータ・ファイル（10MB）。SIZE が指定されているため、このファイルは自動拡張可能ではありません。
- CONTROL_FILES 初期化パラメータが指定されていない場合は、2 つの制御ファイルが /u02/oradata/sample3 と /u03/oradata/sample3 に 1 つずつ作成されます。/u02/oradata/sample3 の制御ファイルが主制御ファイルになります。

初期化パラメータ・ファイルで、次のパラメータを設定します。

```
DB_CREATE_FILE_DEST = '/u01/oradata/sample3'
DB_CREATE_ONLINE_LOG_DEST_1 = '/u02/oradata/sample3'
DB_CREATE_ONLINE_LOG_DEST_2 = '/u03/oradata/sample3'
```

SQL プロンプトから次の文を発行します。

```
SQL> CREATE DATABASE sample3 DATAFILE SIZE 400M
2>   DEFAULT TEMPORARY TABLESPACE dflt_ts TEMPFILE SIZE 10M
3>   UNDO TABLESPACE undo_ts DATAFILE SIZE 10M;
```

表領域用データ・ファイルの作成

ここでは、データ・ファイルを作成する次の文について説明します。

- CREATE TABLESPACE
- CREATE UNDO TABLESPACE
- ALTER TABLESPACE ... ADD DATAFILE

表領域を作成するときは、通常の表領域と UNDO 表領域のどちらの場合でも、DATAFILE 句はオプションです。DATAFILE 句を指定する場合、ファイル名はオプションです。DATAFILE 句またはファイル名を省略すると、次のルールが適用されます。

- DB_CREATE_FILE_DEST 初期化パラメータが設定されている場合は、パラメータで指定された場所に Oracle Managed Files のデータ・ファイルが作成されます。
- DB_CREATE_FILE_DEST 初期化パラメータが設定されていない場合は、データ・ファイルを作成する文が失敗します。

ALTER TABLESPACE ... ADD DATAFILE 文で表領域にデータ・ファイルを追加する場合、ファイル名はオプションです。ファイル名を省略すると、前の段落で説明したのと同じルールが適用されます。

デフォルトでは、通常表領域用の Oracle Managed Files のデータ・ファイルのサイズは 100MB です。このファイルは自動的に拡張可能で、最大サイズに制限はありません。ただし、DATAFILE 句で SIZE 値を指定して（AUTOEXTEND 句を指定せずに）これらのデフォルトを変更すると、データ・ファイルは自動拡張可能になりません。

関連項目：

- 3-10 ページ「データベース作成時の SYSTEM 表領域用データ・ファイルの指定」
- 3-11 ページ「データベース作成時の UNDO 表領域データ・ファイルの指定」
- 第 11 章「表領域の管理」

CREATE TABLESPACE: 例

ここでは、Oracle Managed Files を持つ表領域の作成例をいくつか示します。

CREATE TABLESPACE: 例 1

次の例では、データ・ファイルを作成するデフォルトの場所を /u01/oradata/sample に設定してから、そのディレクトリ上のデータ・ファイルを含む表領域 tbs_1 を作成しています。データ・ファイルは 100MB で、無制限に自動拡張可能です。

```
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST = '/u01/oradata/sample';
SQL> CREATE TABLESPACE tbs_1;
```

CREATE TABLESPACE: 例 2

この例では、ディレクトリ /u01/oradata/sample2 上のデータ・ファイルを含む表領域 tbs_2 を作成しています。このデータ・ファイルの初期サイズは 400MB で、自動拡張可能ではありません。

初期化パラメータ・ファイルで、次のパラメータを設定します。

```
DB_CREATE_FILE_DEST = '/u01/oradata/sample2'
```

SQL プロンプトから次の文を発行します。

```
SQL> CREATE TABLESPACE tbs_2 DATAFILE SIZE 400M AUTOEXTEND OFF;
```

CREATE TABLESPACE: 例 3

この例では、ディレクトリ /u01/oradata/sample3 上のデータ・ファイルを含む表領域 tbs_3 を作成しています。作成されるデータ・ファイルは初期サイズが 100MB、最大サイズが 800MB で自動拡張可能です。

初期化パラメータ・ファイルで、次のパラメータを設定します。

```
DB_CREATE_FILE_DEST = '/u01/oradata/sample3'
```

SQL プロンプトから次の文を発行します。

```
SQL> CREATE TABLESPACE tbs_3 DATAFILE AUTOEXTEND ON MAXSIZE 800M;
```

CREATE TABLESPACE: 例 4

次の例では、データ・ファイルを作成するデフォルトの場所を /u01/oradata/sample4 に設定してから、そのディレクトリ上の 2 つのデータ・ファイルを含む表領域 tbs_4 を作成しています。どちらのデータ・ファイルも初期サイズは 200MB で、SIZE 値が指定されているため自動拡張可能ではありません。

```
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST = '/u01/oradata/sample4';
SQL> CREATE TABLESPACE tbs_4 DATAFILE SIZE 200M, SIZE 200M;
```

CREATE UNDO TABLESPACE: 例

次の例では、ディレクトリ /u01/oradata/sample 上のデータ・ファイルを含む UNDO 表領域 undotbs_1 を作成しています。UNDO 表領域用のデータ・ファイルは 100MB で、無制限に自動拡張可能です。

初期化パラメータ・ファイルで、次のパラメータを設定します。

```
DB_CREATE_FILE_DEST = '/u01/oradata/sample'
```

SQL プロンプトから次の文を発行します。

```
SQL> CREATE UNDO TABLESPACE undotbs_1;
```

ALTER TABLESPACE: 例

この例では、自動拡張可能な Oracle Managed Files のデータ・ファイルを tbs_1 表領域に追加しています。デフォルトの初期サイズは 100MB で、最大サイズは 800MB です。

初期化パラメータ・ファイルで、次のパラメータを設定します。

```
DB_CREATE_FILE_DEST = '/u01/oradata/sample'
```

SQL プロンプトから次の文を入力します。

```
SQL> ALTER TABLESPACE tbs_1 ADD DATAFILE AUTOEXTEND ON MAXSIZE 800M;
```

一時表領域用一時ファイルの作成

ここでは、一時ファイルを作成する次の文について説明します。

- CREATE TEMPORARY TABLESPACE
- ALTER TABLESPACE ... ADD TEMPFILE

一時表領域を作成する場合、TEMPFILE 句はオプションです。TEMPFILE 句を指定する場合、ファイル名はオプションです。TEMPFILE 句またはファイル名を省略すると、次のルールが適用されます。

- DB_CREATE_FILE_DEST 初期化パラメータが設定されている場合は、パラメータで指定された場所に Oracle Managed Files の一時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は、一時ファイルを作成する文が失敗します。

ALTER TABLESPACE ... ADD TEMPFILE 文で表領域に一時ファイルを追加する場合、ファイル名はオプションです。ファイル名を省略すると、前の段落で説明したのと同じルールが適用されます。

Oracle Managed Files のデフォルト属性を上書きするときに、SIZE 値を指定しても AUTOEXTEND 句を指定しない場合、データ・ファイルは自動拡張可能になりません。

関連項目： 3-11 ページ「データベース作成時のデフォルト一時表領域用一時ファイルの指定」

CREATE TEMPORARY TABLESPACE: 例

次の例では、データ・ファイルを作成するデフォルトの場所を /u01/oradata/sample に設定してから、そのディレクトリ上の一時ファイルを含む表領域 temptbs_1 を作成しています。一時ファイルは 100MB で、無制限に自動拡張可能です。

```
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST = '/u01/oradata/sample';
SQL> CREATE TEMPORARY TABLESPACE temptbs_1;
```

ALTER TABLESPACE ... ADD TEMPFILE: 例

次の例では、データ・ファイルを作成するデフォルトの場所を /u03/oradata/sample に設定してから、デフォルトの場所にある一時ファイルを表領域 temptbs_1 に追加しています。一時ファイルの初期サイズは 100MB です。このファイルは自動的に拡張可能で、最大サイズに制限はありません。

```
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST = '/u03/oradata/sample';
SQL> ALTER TABLESPACE TBS_1 ADD TEMPFILE;
```

制御ファイルの作成

CREATE CONTROLFILE 文を発行すると、CONTROL_FILES 初期化パラメータによって指定されたファイルで、制御ファイルが作成（REUSE を指定した場合は再利用）されます。CONTROL_FILES パラメータが設定されていない場合は、制御ファイルのデフォルトの保存先に制御ファイルが作成されます。デフォルトの保存先は、次の優先順位に従って定義されます。

- DB_CREATE_ONLINE_LOG_DEST_*n* 初期化パラメータが設定されている場合は、指定された各ディレクトリに Oracle Managed Files の制御ファイルのコピーが作成されます。最初のディレクトリに作成されたファイルが主制御ファイルになります。
- DB_CREATE_FILE_DEST 初期化パラメータが設定されており、DB_CREATE_ONLINE_LOG_DEST_*n* 初期化パラメータが設定されていない場合は、指定されたディレクトリに Oracle Managed Files の制御ファイルが作成されます。
- DB_CREATE_FILE_DEST および DB_CREATE_ONLINE_LOG_DEST_*n* 初期化パラメータがどちらも設定されていない場合は、オペレーティング・システム固有のデフォルトの場所に制御ファイルが 1 つ作成されます。この制御ファイルは Oracle Managed Files ではありません。

Oracle Managed Files の制御ファイルが作成された場合で、サーバー・パラメータ・ファイルが存在するときは、サーバー・パラメータ・ファイルに CONTROL_FILES 初期化パラメータが作成されます。サーバー・パラメータ・ファイルが存在しない場合は、CONTROL_FILES 初期化パラメータを手動で作成して、初期化パラメータ・ファイルに追加する必要があります。

データベースのデータ・ファイルが Oracle Managed Files の場合は、文の DATAFILE 句に、そのファイルの Oracle 生成ファイル名を指定する必要があります。

オンライン REDO ログ・ファイルが Oracle Managed Files の場合は、NORESETLOGS または RESETLOGS キーワードによって、LOGFILE 句に指定できるパラメータが決まります。

- NORESETLOGS キーワードを使用する場合は、Oracle Managed Files のオンライン REDO ログ・ファイル用に生成されるファイル名を LOGFILE 句に指定する必要があります。
- RESETLOGS キーワードを使用する場合は、オンライン REDO ログ・ファイル名を CREATE DATABASE 文の場合と同様に指定できます。3-10 ページの「データベース作成時のオンライン REDO ログ・ファイルの指定」を参照してください。

Oracle Managed Files を使用した CREATE CONTROLFILE 文の使用例については、次の項を参照してください。

関連項目： 3-9 ページ「データベース作成時の制御ファイルの指定」

NORESETLOGS キーワードを使用した CREATE CONTROLFILE: 例

次の CREATE CONTROLFILE 文は、Oracle Managed Files のデータ・ファイルおよびオンライン REDO ログ・ファイルを含むデータベースで ALTER DATABASE BACKUP CONTROLFILE TO TRACE 文を発行したときに生成されます。

```
CREATE CONTROLFILE
     DATABASE sample
     LOGFILE GROUP 1 ('/u01/oradata/sample/ora_1_o220rtt9.log',
                      '/u02/oradata/sample/ora_1_v2o0b2i3.log') SIZE 100M,
             GROUP 2 ('/u01/oradata/sample/ora_2_p22056iw.log',
                      '/u02/oradata/sample/ora_2_p02rcyg3.log') SIZE 100M
     NORESETLOGS
     DATAFILE '/u01/oradata/sample/ora_system_xu34ybm2.dbf' SIZE 100M
     MAXLOGFILES 5
     MAXLOGHISTORY 100
     MAXDATAFILES 10
     MAXINSTANCES 2
     ARCHIVELOG;
```

RESETLOGS キーワードを使用した CREATE CONTROLFILE: 例

次の文は、RESETLOGS オプションを指定した CREATE CONTROLFILE 文の例です。DB_CREATE_ONLINE_LOG_DEST_*n* または DB_CREATE_FILE_DEST を設定する必要があります。

```
CREATE CONTROLFILE
     DATABASE sample
     RESETLOGS
     DATAFILE '/u01/oradata/sample/ora_system_aawbmz51.dbf' SIZE 100M
     MAXLOGFILES 5
     MAXLOGHISTORY 100
     MAXDATAFILES 10
     MAXINSTANCES 2
     ARCHIVELOG;
```

後で、ALTER DATABASE OPEN RESETLOGS 文を発行して、オンライン REDO ログ・ファイルを再作成する必要があります。この操作については、3-20 ページの「ALTER DATABASE OPEN RESETLOGS 文の使用」を参照してください。使用していたログ・ファイルが Oracle Managed Files である場合、そのファイルは削除されません。

オンライン REDO ログ・ファイルの作成

オンライン REDO ログ・ファイルはデータベース作成時に作成されます。また、次の文のどちらかを発行したときにも作成できます。

- ALTER DATABASE ADD LOGFILE
- ALTER DATABASE OPEN RESETLOGS

ALTER DATABASE ADD LOGFILE 文の使用

ALTER DATABASE ADD LOGFILE 文を使用すると、現行のオンライン REDO ログ・ファイルに後から新しいグループを追加できます。Oracle Managed Files を使用している場合、ADD LOGFILE 句のファイル名はオプションです。ファイル名を省略した場合は、ログ・ファイルのデフォルトの保存先に REDO ログ・ファイルが作成されます。デフォルトの保存先は、次の優先順位に従って定義されます。

- DB_CREATE_ONLINE_LOG_DEST_*n* 初期化パラメータが設定されている場合は、パラメータで指定された各ディレクトリに Oracle Managed Files のログ・ファイルのメンバーが作成されます（最大数はデータベースの MAXLOGMEMBERS）。
- DB_CREATE_FILE_DEST 初期化パラメータが設定されており、DB_CREATE_ONLINE_LOG_DEST_*n* 初期化パラメータが設定されていない場合は、パラメータで指定されたディレクトリに Oracle Managed Files のログ・ファイルのメンバーが作成されます。

ファイル名を指定せず、Oracle Managed Files の作成に必要な初期化パラメータが 1 つも指定されていない場合は、文はエラーを戻します。

Oracle Managed Files のログ・ファイルのデフォルト・サイズは 100MB です。

完全ファイル名を指定すると、オンライン REDO ログ・ファイルのメンバーを引き続き追加および削除できます。

関連項目：

- 3-10 ページ「データベース作成時のオンライン REDO ログ・ファイルの指定」
- 3-17 ページ「制御ファイルの作成」

新しいオンライン REDO ログ・ファイルの追加の例

次の例では、一方のメンバーがディレクトリ /u01/oradata/sample、もう一方のメンバーが /u02/oradata/sample に存在するログ・ファイルを作成します。ログ・ファイルのサイズは 100MB です。

初期化パラメータ・ファイルで、次のパラメータを設定します。

```
DB_CREATE_ONLINE_LOG_DEST_1 = '/u01/oradata/sample'
DB_CREATE_ONLINE_LOG_DEST_2 = '/u02/oradata/sample'
```

SQL プロンプトから次の文を発行します。

```
SQL> ALTER DATABASE ADD LOGFILE;
```

ALTER DATABASE OPEN RESETLOGS 文の使用

前に RESETLOGS を指定して制御ファイルを作成しており、その際、ファイル名を指定しなかった場合、または存在しないファイル名を指定した場合は、ALTER DATABASE OPEN RESETLOGS 文を発行したときに、オンライン REDO ログ・ファイルが作成されます。制御ファイル内に何も指定されていない場合に、REDO ログ・ファイルの格納ディレクトリを決めるルールは、3-10 ページの「データベース作成時のオンライン REDO ログ・ファイルの指定」に記載されているルールと同じです。

Oracle Managed Files の動作

ファイル名を使用して既存ファイルを識別する SQL 文では、Oracle Managed Files のファイル名が受け入れられます。これらのファイル名は他のファイル名と同様に、制御ファイルに格納されています。また、バックアップとリカバリ用に Recovery Manager を使用している場合は、Recovery Manager カタログに格納されています。これらのファイル名を表示するには、データ・ファイルと一時ファイルの監視に使用できる通常の固定パフォーマンス・ビューまたは動的パフォーマンス・ビュー（たとえば、V$DATAFILE や DBA_DATA_FILES など）のいずれかを使用します。

次に、Oracle 生成ファイル名を使用した文の例を示します。

```
SQL> ALTER DATABASE RENAME FILE 'ora_tbs01_ziw3bopb.dbf' TO 'tbs0101.dbf';

SQL> ALTER DATABASE DROP LOGFILE 'ora_1_wo94n2xi.log';

SQL> ALTER TABLE emp ALLOCATE EXTENT ( DATAFILE 'ora_tbs1_2ixfh90q.dbf' );
```

Oracle Managed Files のデータ・ファイル、一時ファイルおよび制御ファイルは、Oracle Managed Files 以外の対応するファイルと同様にバックアップおよびリストアを実行できます。Oracle 生成ファイル名を使用しても、エクスポート・ファイルなどの論理バックアップ・ファイルの使用には影響しません。これは特に、表領域の Point-in-Time リカバリ（TSPITR）およびトランスポータブル表領域のエクスポート・ファイルにとって重要です。

Oracle Managed Files の動作が Oracle Managed Files 以外のファイルと異なる場合があります。次の項では、それらの場合について説明します。

データ・ファイルおよび一時ファイルの削除

Oracle 管理ではないファイルとは異なり、Oracle Managed Files のデータ・ファイルまたは一時ファイルを削除すると、制御ファイルからファイル名が削除されて、ファイル・システムからファイルが自動的に削除されます。Oracle Managed Files を削除する文は、次のとおりです。

- DROP TABLESPACE
- ALTER DATABASE TEMPFILE ...DROP

オンライン REDO ログ・ファイルの削除

Oracle Managed Files のオンライン REDO ログ・ファイルを削除すると、その Oracle Managed Files が削除されます。削除するグループまたはメンバーを指定します。次の文は、オンライン REDO ログ・ファイルを削除します。

- ALTER DATABASE DROP LOGFILE
- ALTER DATABASE DROP LOGFILE MEMBER

ファイルの名前変更

ファイルの名前を変更するには、次の文が使用されます。

- `ALTER DATABASE RENAME FILE`
- `ALTER TABLESPACE ... RENAME DATAFILE`

これらの文は、実際にはオペレーティング・システム上のファイルの名前を変更しませんが、そのかわりに制御ファイル内の名前が変更されます。変更前のファイルが Oracle Managed Files で、そのファイルが存在している場合は削除されます。この文を発行するときは、オペレーティング・システムのファイル名の規則を使用して各ファイルを指定する必要があります。

スタンバイ・データベースの管理

スタンバイ・データベースのデータ・ファイル、制御ファイルおよびオンライン REDO ログ・ファイルは、Oracle Managed Files にすることが可能です。プライマリ・データベースで Oracle Managed Files が使用されているかどうかは関係ありません。

スタンバイ・データベースのリカバリでデータ・ファイルを作成する REDO を検出したとき、そのデータ・ファイルが Oracle Managed Files の場合は、リカバリ・プロセスによって、ローカル・ファイル・システムのデフォルトの場所に空のファイルが作成されます。これにより、管理者が操作することなく、新しいファイルの REDO が即時に適用されます。

スタンバイ・データベースのリカバリで表領域を削除する REDO を検出した場合は、ローカル・ファイル・システム内にある Oracle Managed Files のデータ・ファイルがすべて削除されます。プライマリ・データベースで `INCLUDING DATAFILES` オプションを発行したかどうかは関係ありません。

Oracle Managed Files の使用例

ここでは、使用例を示して、Oracle Managed Files の使用方法をさらに詳しく説明します。

使用例 1: 多重オンライン REDO ログを含むデータベースの作成および管理

この使用例では、DBA が、データ・ファイルとオンライン REDO ログ・ファイルが異なるディレクトリに存在するデータベースを作成します。オンライン REDO ログ・ファイルと制御ファイルは多重化されています。データベースは UNDO 表領域を使用し、デフォルト一時表領域を持っています。このデータベースの作成とメンテナンスに関するタスクは、次のとおりです。

1. 初期化パラメータの設定

   DBA は、データベースを作成する前に、初期化パラメータ・ファイルに 3 つの汎用的なファイル作成デフォルトを設定します。自動 UNDO 管理モードも使用可能にします。

   ```
   DB_CREATE_FILE_DEST = '/u01/oradata/sample'
   DB_CREATE_ONLINE_LOG_DEST_1 = '/u02/oradata/sample'
   DB_CREATE_ONLINE_LOG_DEST_2 = '/u03/oradata/sample'
   UNDO_MANAGEMENT = AUTO
   ```

   `DB_CREATE_FILE_DEST` パラメータは、データ・ファイルと一時ファイルのデフォルトのファイル・システム・ディレクトリを設定します。

   `DB_CREATE_ONLINE_LOG_DEST_1` および `DB_CREATE_ONLINE_LOG_DEST_2` パラメータは、オンライン REDO ログ・ファイルと制御ファイルを作成するためのデフォルトのファイル・システム・ディレクトリを設定します。オンライン REDO ログ・ファイルと制御ファイルは、2 つのディレクトリの間で多重化されます。

2. データベースの作成

   初期化パラメータの設定が完了すると、次の文でデータベースを作成できます。

   ```
   SQL> CREATE DATABASE sample
   2>   DEFAULT TEMPORARY TABLESPACE dflt_tmp;
   ```

   `DATAFILE` 句が指定されておらず、`DB_CREATE_FILE_DEST` 初期化パラメータが設定されているので、`SYSTEM` 表領域のデータ・ファイルはデフォルトのファイル・システム（この使用例では `/u01/oradata/sample`）に作成されます。ファイル名は、Oracle によって一意に生成されます。データ・ファイルは初期サイズが 100MB で、無制限に自動拡張可能です。このファイルは Oracle Managed Files です。

   `LOGFILE` 句が指定されていないので、2 つのオンライン REDO ログ・グループが作成されます。各グループにはそれぞれ 2 つのメンバーがあり、一方のメンバーは `DB_CREATE_ONLINE_LOG_DEST_1`、もう一方のメンバーは `DB_CREATE_ONLINE_LOG_DEST_2` に作成されます。ファイル名は、Oracle によって一意に生成されます。ログ・ファイルのサイズは 100MB です。ログ・ファイルのメンバーは Oracle Managed Files です。

同様に、CONTROL_FILES 初期化パラメータが設定されておらず、2 つの DB_CREATE_ONLINE_LOG_DEST_*n* 初期化パラメータが設定されているので、2 つの制御ファイルが作成されます。DB_CREATE_ONLINE_LOG_DEST_1 に配置された制御ファイルが主制御ファイルになります。DB_CREATE_ONLINE_LOG_DEST_2 に配置された制御ファイルは多重コピーです。ファイル名は、Oracle によって一意に生成されます。これらのファイルは Oracle Managed Files です。サーバー・パラメータ・ファイルが存在する場合は、CONTROL_FILES 初期化パラメータが生成されます。

自動 UNDO 管理モードが設定されていますが、UNDO 表領域が指定されておらず、DB_CREATE_FILE_DEST 初期化パラメータが設定されているので、DB_CREATE_FILE_DEST で指定されたディレクトリに、SYS_UNDOTS という名前のデフォルト UNDO 表領域が作成されます。データ・ファイルは 10MB で、自動拡張可能です。このファイルは Oracle Managed Files です。

最後に、dflt_tmp という名前のデフォルト一時表領域が指定されています。パラメータ・ファイルに DB_CREATE_FILE_DEST が設定されているので、このパラメータで指定されたディレクトリに dflt_tmp の一時ファイルが作成されます。一時ファイルは 100MB で、無制限に自動拡張可能です。このファイルは Oracle Managed Files です。

作成されたファイルを、生成ファイル名によるファイル・ツリーで表現すると次のようになります。

```
/u01
     /oradata
          /sample
               /ora_system_cmr7t30p.dbf
               /ora_sys_undo_2ixfh90q.dbf
               /ora_dflt_tmp_157se6ff.tmp
/u02
     /oradata
          /sample
               /ora_1_0orrm31z.log
               /ora_2_2xyz16am.log
               /ora_cmr7t30p.ctl
/u03
     /oradata
          /sample
               /ora_1_ixfvm8w9.log
               /ora_2_q89tmp28.log
               /ora_x1sr8t36.ctl
```

内部的に生成されたファイル名は、通常のビューを選択して表示できます。次に例を示します。

```
SQL> SELECT NAME FROM V$DATAFILE;

NAME
----------------------------------------------------
/u01/oradata/sample/ora_system_cmr7t30p.dbf
/u01/oradata/sample/ora_sys_undo_2ixfh90q.dbf

2 rows selected
```

ファイル名は、ファイルが作成されたときにアラート・ファイルにも出力されます。

3. 制御ファイルの管理

データベースの作成時に制御ファイルが作成され、パラメータ・ファイルに CONTROL_FILES 初期化パラメータが追加されました。必要に応じて、DBA は CREATE CONTROLFILE 文を使用して、データベース用の制御ファイルを再作成したり、新しい制御ファイルを作成できます。

DATAFILE 句および LOGFILE 句には、正しい Oracle Managed Files のファイル名を指定する必要があります。ALTER DATABASE BACKUP CONTROLFILE TO TRACE 文は、正しいファイル名を含むスクリプトを生成します。また、ファイル名は、V$DATAFILE、V$TEMPFILE および V$LOGFILE ビューを選択して確認することもできます。次の例では、サンプル・データベースの制御ファイルを再作成しています。

```
SQL> CREATE CONTROLFILE REUSE
2>       DATABASE sample
3>       LOGFILE GROUP 1('/u02/oradata/sample/ora_1_0orrm31z.log',
4>                       '/u03/oradata/sample/ora_1_ixfvm8w9.log'),
5>               GROUP 2('/u02/oradata/sample/ora_2_2xyz16am.log',
6>                       '/u03/oradata/sample/ora_2_q89tmp28.log')
7>       NORESETLOGS
8>       DATAFILE '/u01/oradata/sample/ora_system_cmr7t30p.dbf',
9>                '/u01/oradata/sample/ora_sys_undo_2ixfh90q.dbf',
10>               '/u01/oradata/sample/ora_dflt_tmp_157se6ff.tmp'
11>      MAXLOGFILES 5
12>      MAXLOGHISTORY 100
13>      MAXDATAFILES 10
14>      MAXINSTANCES 2
15>      ARCHIVELOG;
```

この文で作成される制御ファイルは、データベースを作成したときに生成された CONTROL_FILES 初期化パラメータの指定どおりに配置されます。REUSE 句が指定されているので、既存のファイルがすべて上書きされます。

4. オンライン REDO ログの管理

   オンライン REDO ログ・ファイルの新しいグループを作成するには、DBA が ALTER DATABASE ADD LOGFILE 文を使用します。次の文は、DB_CREATE_ONLINE_LOG_DEST_1 と DB_CREATE_ONLINE_LOG_DEST_2 にメンバーを持つログ・ファイルを追加します。これらのファイルは Oracle Managed Files です。

   ```
   SQL> ALTER DATABASE ADD LOGFILE;
   ```

   オンライン REDO ログ・ファイルのメンバーは、完全なファイル名を指定することにより、追加および削除できます。

   GROUP 句を使用して、ログ・ファイルを削除できます。次の例では、Oracle Managed Files のログ・ファイルの各メンバーに対応するオペレーティング・システム・ファイルが自動的に削除されます。

   ```
   SQL> ALTER DATABASE DROP LOGFILE GROUP 3;
   ```

5. 表領域の管理

   sample データベースで今後表領域を作成する際、すべてのデータ・ファイルがデフォルトで配置される記憶域は、DB_CREATE_FILE_DEST 初期化パラメータで指定された場所（この使用例では /u01/oradata/sample）です。ファイル名を指定せずにデータ・ファイルを作成すると、そのファイルは初期化パラメータ DB_CREATE_FILE_DEST で指定されたファイル・システムに配置されます。次に例を示します。

   ```
   SQL> CREATE TABLESPACE tbs_1;
   ```

   この文は、/u01/oradata/sample を記憶域とする表領域を作成します。作成されるデータ・ファイルは初期サイズが 100MB で、無制限に自動拡張可能です。このデータ・ファイルは Oracle Managed Files です。

   表領域を削除すると、その表領域に対応する Oracle Managed Files も自動的に削除されます。次の文は、表領域とその格納に使用されているすべての Oracle Managed Files を削除します。

   ```
   SQL> DROP TABLESPACE tbs_1;
   ```

   最初のデータ・ファイルがいっぱいになっても、新しいデータ・ファイルは自動的に作成されません。別の Oracle Managed Files のデータ・ファイルを追加することによって、表領域を拡張できます。次の文は、DB_CREATE_FILE_DEST で指定された場所に別のデータ・ファイルを追加します。

   ```
   SQL> ALTER TABLESPACE tbs_1 ADD DATAFILE;
   ```

デフォルトのファイル・システムは、初期化パラメータを変更することによって変更できます。これを行っても、既存のデータベースは変更されません。今後の作成にのみ影響を与えます。次の文を使用すると、初期化パラメータを動的に変更できます。

```
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST='/u04/oradata/sample';
```

6. REDO 情報のアーカイブ

   オンライン REDO ログ・ファイルのアーカイブは、Oracle Managed Files と Oracle Managed Files 以外のファイルの間で違いはありません。アーカイブするログ・ファイルのファイル・システム上のアーカイブ先は、LOG_ARCHIVE_DEST_*n* 初期化パラメータで指定できます。ファイル名は、LOG_ARCHIVE_FORMAT パラメータまたはそのデフォルトに基づいて生成されます。アーカイブ・ログは Oracle Managed Files ではありません。

7. バックアップ、リストアおよびリカバリ

   Oracle Managed Files は標準オペレーティング・システム・ファイルと互換性があるため、オペレーティング・システム・ユーティリティを使用してバックアップまたはリストアを実行できます。データベースのバックアップ、リストアおよびリカバリを実行する既存の方法はすべて、Oracle Managed Files に対しても機能します。

使用例 2: 既存のデータベースへの Oracle Managed Files の追加

この例では、Oracle Managed Files が含まれていない既存のデータベースに対して、DBA が Oracle Managed Files を含む新しい表領域を作成し、/u03/oradata/sample2 ディレクトリにその表領域を配置しようとしていると想定しています。

1. 初期化パラメータの設定

   データ・ファイルの自動作成を可能にするために、DB_CREATE_FILE_DEST 初期化パラメータを、データ・ファイルを作成するファイル・システム・ディレクトリに設定します。これは、次のように動的に実行できます。

```
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST = '/u03/oradata/sample2';
```

2. 表領域の作成

   DB_CREATE_FILE_DEST の設定が完了すると、CREATE TABLESPACE 文から DATAFILE 句を省略できます。データ・ファイルは、DB_CREATE_FILE_DEST で指定された場所にデフォルトで作成されます。次に例を示します。

```
SQL> CREATE TABLESPACE tbs_2;
```

   tbs_2 表領域を削除すると、データ・ファイルが自動的に削除されます。

4

起動と停止

この章では、Oracle データベースの起動と停止の手順について説明します。この章の内容は、次のとおりです。

- データベースの起動
- データベースの実行モードの変更
- データベースの停止
- データベースの静止
- データベースの一時停止と再開

関連項目： Oracle Real Application Clusters 環境に固有の追加情報は、次のマニュアルを参照してください。

- 『Oracle9*i* Real Application Clusters 管理』
- 『Oracle9*i* Real Application Clusters セットアップおよび構成』

データベースの起動

データベースの起動時に、そのデータベースのインスタンスを作成し、データベースの起動状態を選択します。通常は、データベースをマウントおよびオープンしてインスタンスを起動し、有効なユーザーが接続して典型的なデータ・アクセス操作を実行できるようにします。ここでは他の起動方法についても説明します。

この項の内容は、次のとおりです。

データベースの起動方法

データベース・インスタンスを起動（および管理）するには、いくつかの方法があります。

SQL*Plus の使用

データベースを起動するには、SQL*Plus を使用して管理者権限のあるユーザーで Oracle に接続し、STARTUP コマンドを発行します。ここでは 3 つの方法を示していますが、このマニュアルの対象は SQL*Plus を使用する方法のみです。

Recovery Manager の使用

Recovery Manager を使用して、STARTUP（および SHUTDOWN）コマンドを実行する方法もあります。この方法を選択するのは、Recovery Manager 環境で SQL*Plus を起動しない場合です。

関連項目：『Oracle9*i* Recovery Manager ユーザーズ・ガイド』

Oracle Enterprise Manager の使用

起動や停止など、データベース管理の目的で Oracle Enterprise Manager を使用できます。Oracle Enterprise Manager は独立した Oracle 製品で、グラフィカルなコンソール、エージェント、共有サービスおよび Oracle のツール製品を組み合せて、Oracle 製品の管理のために統合された包括的なシステム管理プラットフォームを提供します。Oracle Enterprise Manager を使用することにより、このマニュアルで説明されている機能を、コマンドラインではなく GUI を使用して実行できます。

関連項目：

- 『Oracle Enterprise Manager 概要』
- 『Oracle Enterprise Manager 管理者ガイド』

インスタンス起動の準備

SQL*Plus を使用してデータベース・インスタンスを起動する前に、次の準備手順を実行する必要があります。

1. データベースに接続せずに、SQL*Plus を起動します。

   ```
   SQLPLUS /NOLOG
   ```

2. SYSDBA として Oracle に接続します。

   ```
   CONNECT username/password AS SYSDBA
   ```

これで Oracle に接続され、データベース・インスタンスを起動する準備が完了します。

> **関連項目：** CONNECT、STARTUP および SHUTDOWN コマンドの説明と構文は、『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。これらのコマンドは SQL*Plus コマンドです。

SQL*Plus を使用したデータベースの起動

データベース・インスタンスを起動するには、STARTUP コマンドを使用します。Oracle は、インスタンスを起動するために、サーバー・パラメータ・ファイルまたは従来のテキスト形式の初期化パラメータ・ファイルからインスタンス構成パラメータ（初期化パラメータ）を読み込む必要があります。

PFILE 句を指定せずに STARTUP コマンドを発行すると、オペレーティング・システム固有のデフォルトの場所にあるサーバー・パラメータ・ファイル（SPFILE）から初期化パラメータが読み込まれます。

注意： UNIX の場合、サーバー・パラメータ・ファイル（またはテキスト形式の初期化パラメータ・ファイル）が配置されるプラットフォーム固有のデフォルトの場所（ディレクトリ）は、次のとおりです。

```
$ORACLE_HOME/dbs
```

Windows NT および Windows 2000 の場合は、次のとおりです。

```
%ORACLE_HOME%¥database
```

Oracle は、プラットフォーム固有のデフォルトの場所にあるファイル名を次の順序で検査し、初期化パラメータ・ファイルを特定します。

1. spfile$ORACLE_SID.ora
2. spfile.ora
3. init$ORACLE_SID.ora

注意： spfile.ora ファイルがこの検索パスに含まれているのは、Real Application Clusters 環境では、すべてのインスタンスの初期化パラメータ設定が 1 つのサーバー・パラメータ・ファイルに格納されるためです。サーバー・パラメータ・ファイルにはインスタンス固有の格納場所はありません。

Real Application Clusters 環境におけるサーバー・パラメータ・ファイルの詳細は、『Oracle9*i* Real Application Clusters 管理』を参照してください。

STARTUP コマンドで PFILE 句を指定すると、従来のテキスト形式の初期化パラメータ・ファイルから初期化パラメータを直接 Oracle に読み込ませることができます。次に例を示します。

```
STARTUP PFILE = /u01/oracle/dbs/init.ora
```

また、PFILE 句を次のように使用すると、デフォルト以外のサーバー・パラメータ・ファイルでインスタンスを起動できます。

1. SPFILE パラメータのみを記述した 1 行のテキスト形式の初期化パラメータ・ファイルを作成します。パラメータの値には、デフォルト以外のサーバー・パラメータ・ファイルの場所を指定します。

   たとえば、次のパラメータのみを記述したテキスト形式の初期化パラメータ・ファイル /u01/oracle/dbs/spf_init.ora を作成します。

   ```
   SPFILE = /u01/oracle/dbs/test_spfile.ora
   ```

   注意： 従来のテキスト形式の初期化パラメータ・ファイルで、サーバー・パラメータ・ファイルを設定するために IFILE 初期化パラメータを使用することはできません。この場合は、必ず SPFILE 初期化パラメータを使用してください。

   ---

2. この初期化パラメータ・ファイルを指定して、インスタンスを起動します。

   ```
   STARTUP PFILE = /u01/oracle/dbs/spf_init.ora
   ```

サーバー・パラメータ・ファイルは必ずデータベース・サーバーが稼働しているマシン上に存在するため、この方法ではクライアント・マシンでサーバー・パラメータ・ファイルを使用してデータベースを起動する手段も提供します。また、クライアント・マシンで、クライアント側の初期化パラメータ・ファイルをメンテナンスする必要はありません。クライアント・マシンが SPFILE パラメータを含む初期化パラメータ・ファイルを読み込むと、サーバーにその値が渡され、指定されたサーバー・パラメータ・ファイルが読み込まれます。

インスタンスは、次のような様々なモードで起動できます。

- インスタンスを起動するが、データベースはマウントしないモード。このモードで起動すると、データベースにはアクセスできません。通常、このモードで起動するのは、データベースの作成時または制御ファイルの再作成時のみです。
- インスタンスを起動し、データベースをマウントするが、クローズしたままにするモード。この状態では、一部の DBA アクティビティは可能ですが、データベースへの汎用アクセスはできません。
- インスタンスを起動し、データベースをマウントしてオープンするモード。この操作を非制限モードで実行してユーザー全員にアクセスを許可するか、制限モードで実行して DBA のみにアクセスを許可できます。

注意： 共有サーバー・プロセスを介してデータベースに接続してデータベースのインスタンスを起動することはできません。

また、インスタンスを強制的に起動したり、インスタンスの起動直後に完全メディア・リカバリを開始したりできます。この状態をアーカイブするために指定する STARTUP オプションは、次の項を参照してください。

関連項目： 初期化パラメータ、初期化パラメータ・ファイルおよびサーバー・パラメータ・ファイルの詳細は、第 2 章「Oracle データベースの作成」を参照してください。

インスタンスの起動例

次の例では、DBA がインスタンスを起動できるいくつかの状態を具体的に説明します。STARTUP コマンドのオプションを組み合せると、制限が適用されます。

注意： 制御ファイル、データベース・ファイルまたは REDO ログ・ファイルが使用できない場合は、インスタンスの起動で問題が発生することがあります。データベースをマウントするときに CONTROL_FILES 初期化パラメータで指定された 1 つ以上のファイルが存在しない場合、またはオープンできない場合は、警告メッセージが表示され、データベースはマウントされません。データベースをオープンするときに 1 つ以上のデータ・ファイルまたは REDO ログ・ファイルを使用できない場合、またはオープンできない場合は、警告メッセージが表示され、データベースはオープンされません。

関連項目： STARTUP コマンドのオプションを組み合せるときに適用される制限の詳細は、『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。

インスタンスを起動し、データベースをマウントしてオープンする方法

通常のデータベース操作とは、インスタンスを起動し、データベースをマウントおよびオープンすることを意味します。このモードでは、有効なユーザーがデータベースに接続して、典型的なデータ・アクセス操作を実行できます。

インスタンスを起動し、デフォルトの場所にあるサーバー・パラメータ・ファイルから初期化パラメータを読み込んで、データベースをマウントおよびオープンするには、STARTUP コマンドを単独で使用します（必要に応じて、PFILE または SPFILE 句を指定できます）。

```
STARTUP
```

インスタンスを起動するが、データベースをマウントしない方法

インスタンスは、データベースをマウントしなくても起動できます。通常、この方法で起動するのはデータベースの作成時のみです。STARTUP コマンドで、NOMOUNT オプションを指定します。

```
STARTUP NOMOUNT
```

インスタンスを起動し、データベースをマウントする方法

インスタンスを起動し、データベースをオープンしないでマウントして、特定のメンテナンス操作を実行できます。たとえば、次のようなタスクの実行時は、データベースのマウントは必要ですが、オープンしてはなりません。

タスク	詳細
データ・ファイルの名前変更	第 12 章「データ・ファイルの管理」
REDO ログ・ファイルの追加、削除または名前変更	第 7 章「オンライン REDO ログの管理」
REDO ログ・アーカイブ・オプションの有効化および無効化	第 8 章「アーカイブ REDO ログの管理」
全データベース・リカバリの実行	『Oracle9*i* ユーザー管理バックアップおよびリカバリ・ガイド』 『Oracle9*i* Recovery Manager ユーザーズ・ガイド』

インスタンスを起動してデータベースをマウントし、クローズしたままにするには、STARTUP コマンドで MOUNT オプションを指定します。

```
STARTUP MOUNT
```

起動時にデータベースへのアクセスを制限する方法

データベースの使用を管理担当者にのみ許可し、一般ユーザーの使用を禁止するには、制限モードでインスタンスを起動して、データベースをマウントおよびオープンします。次のいずれかのタスクを実行するときは、このデータベース起動モードを使用してください。

- データベース・データのエクスポートまたはインポートを実行する場合
- SQL*Loader を使用してデータをロードする場合
- 一時的に一般ユーザーがデータを使用できないようにする場合
- 特定の移行およびアップグレード操作を実行する場合

通常、CREATE SESSION システム権限を持つすべてのユーザーは、オープンしているデータベースに接続できます。制限モードでデータベースをオープンすると、CREATE SESSION システム権限と RESTRICTED SESSION システム権限の両方を持つユーザーのみがデータベースにアクセスできます。したがって、DBA のみが RESTRICTED SESSION システム権限を持つようにしてください。

制限モードでインスタンスを起動（必要に応じて、データベースをマウントしてオープン）するには、STARTUP コマンドで RESTRICT オプションを指定します。

```
STARTUP RESTRICT
```

RESTRICTED SESSION 機能を無効にするには、ALTER SYSTEM 文を使用します。

```
ALTER SYSTEM DISABLE RESTRICTED SESSION;
```

データベースを非制限モードでオープンし、後でアクセス制限が必要であると判明した場合は、ALTER SYSTEM 文を使用して制限できます。4-10 ページの「オープンしているデータベースへのアクセスを制限する方法」を参照してください。

> **関連項目：** ALTER SYSTEM 文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

インスタンスを強制的に起動する方法

通常と異なる状況では、データベース・インスタンスを起動しようとしたときに、問題が発生することがあります。次の問題が発生している場合以外は、データベースを強制的に起動しないでください。

- 現行インスタンスを SHUTDOWN NORMAL、SHUTDOWN IMMEDIATE または SHUTDOWN TRANSACTIONAL コマンドで停止できない場合
- インスタンス起動時に問題が発生した場合

このような問題が発生した場合、STARTUP コマンドで FORCE オプションを指定して新しいインスタンスを起動（必要に応じて、データベースをマウントおよびオープン）すると、通常は問題を解決できます。

```
STARTUP FORCE
```

インスタンスの実行中に STARTUP FORCE を使用すると、ABORT モードで停止した後に再起動します。

> **関連項目：** 現行インスタンスの強制終了による影響の詳細は、4-13 ページの「ABORT オプションによる停止」を参照してください。

インスタンスを起動し、データベースをマウントして、完全メディア・リカバリを開始する方法

メディア・リカバリが必要な場合には、STARTUP コマンドで RECOVER オプションを指定すると、インスタンスを起動し、データベースをインスタンスにマウントして、リカバリ処理を自動的に開始できます。

```
STARTUP OPEN RECOVER
```

必要ない場合にリカバリを実行しようとすると、エラー・メッセージが表示されます。

オペレーティング・システム起動時にデータベースを自動的に起動する方法

多くのサイトでは、システムの起動直後に 1 つ以上の Oracle インスタンスおよびデータベースを自動的に起動する手順を使用しています。そのための手順は、オペレーティング・システムによって異なります。自動起動の詳細は、オペレーティング・システム固有の Oracle マニュアルを参照してください。

リモート・インスタンスを起動する方法

ローカルの Oracle データベースが分散データベースの一部を構成している場合は、リモート・インスタンスとデータベースを起動できます。リモート・インスタンスの起動と停止の手順は、通信プロトコルとオペレーティング・システムによって大きく異なります。

データベースの実行モードの変更

データベースの実行モードを変更できます。たとえば、メンテナンスのためにアクセスを制限したり、データベースを読取り専用にする目的で、データベースの実行モードを変更できます。次の項では、データベースの実行モードの変更方法について説明します。

- インスタンスにデータベースをマウントする方法
- クローズしているデータベースをオープンする方法
- データベースを読取り専用モードでオープンする方法
- オープンしているデータベースへのアクセスを制限する方法

インスタンスにデータベースをマウントする方法

特定の管理操作を実行する場合は、データベースを起動してインスタンスにマウントし、クローズしたままにする必要があります。そのためには、インスタンスを起動してデータベースをマウントします。

あらかじめ起動したインスタンスにデータベースをマウントするには、次のように、SQL 文 ALTER DATABASE で MOUNT オプションを指定します。

```
ALTER DATABASE MOUNT
```

> **関連項目：** データベースをマウントし、クローズしておくことが必要な操作（およびインスタンスの起動とデータベースのマウントを一度に実行する手順）の詳細は、4-6 ページの「インスタンスを起動し、データベースをマウントする方法」を参照してください。

クローズしているデータベースをオープンする方法

データベースをオープンすることによって、マウントされ、クローズしているデータベースを一般的な用途のために使用可能にできます。マウントされたデータベースをオープンするには、ALTER DATABASE 文で OPEN オプションを使用します。

```
ALTER DATABASE OPEN
```

この文の実行後は、CREATE SESSION システム権限を持つ有効な Oracle ユーザーであれば、誰でもデータベースに接続できます。

データベースを読取り専用モードでオープンする方法

データベースを読取り専用モードでオープンすると、オープンしたデータベースを問い合せることができますが、その間にデータの内容がオンラインで変更されることはありません。これにより、データ・ファイルと REDO ログ・ファイルにデータが書き込まれないことが保証されますが、データベース・リカバリや、REDO を生成せずにデータベースの状態を変更する操作が制限されることはありません。たとえば、データ・ファイルをオフラインとオンラインの間で切り替えてもデータの内容には影響しないので、このような切替えが可能です。

ディスク・ソートの実行時など、読取り専用モードでデータベースを問い合せるときに一時表領域を使用する場合、問合せの発行者には、デフォルト一時表領域としてローカル管理表領域が割り当てられている必要があります。割り当てられていない場合は、問合せが失敗します。この操作については、11-12 ページの「ローカル管理の一時表領域の作成」を参照してください。

データベースは、スタンバイ・データベースを読取り専用モードとリカバリ・モードの間で交互に切り替えながら、読取り専用モードでオープンするのが理想的です。この 2 つのモードは相互排他的なので注意してください。

次の文では、データベースが読取り専用モードでオープンします。

```
ALTER DATABASE OPEN READ ONLY;
```

また、次のように読取り / 書込みモードでデータベースをオープンすることもできます。

```
ALTER DATABASE OPEN READ WRITE;
```

ただし、読取り / 書込みはデフォルトのモードです。

注意： RESETLOGS 句と READ ONLY 句は併用できません。

関連項目： ALTER DATABASE 文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

オープンしているデータベースへのアクセスを制限する方法

インスタンスを制限モードにするには、SQL 文の ALTER SYSTEM で ENABLE RESTRICTED SESSION 句を指定します。インスタンスを制限モードにした場合は、管理タスクを実行する前に現行のユーザー・セッションをすべて停止する必要があります。インスタンスの制限モードを解除するには、ALTER SYSTEM で DISABLE RESTRICTED SESSION オプションを指定します。

関連項目： インスタンスを制限モードにする理由は、4-7 ページの「起動時にデータベースへのアクセスを制限する方法」を参照してください。

データベースの停止

データベースを停止するには、SQL*Plus の SHUTDOWN コマンドを使用します。停止が完了するまで、データベース停止を開始したセッションに制御が戻りません。停止処理の進行中に接続しようとするユーザーは、次のようなメッセージを受け取ります。

```
ORA-01090: シャットダウン処理中 - 接続はできません
```

注意： 共有サーバー・プロセスを介してデータベースに接続している場合は、データベースを停止できません。

データベースとインスタンスを停止するには、最初に SYSOPER または SYSDBA として接続する必要があります。データベースにはいくつかの停止モードがあります。次の項では、各モードについて説明します。

- NORMAL オプションによる停止
- IMMEDIATE オプションによる停止
- TRANSACTIONAL オプションによる停止
- ABORT オプションによる停止

NORMAL オプションによる停止

データベースを通常の状況で停止するには、次のように SHUTDOWN コマンドで NORMAL オプションを指定します。

```
SHUTDOWN NORMAL
```

通常のデータベース停止では、次のように処理が進みます。

- 文が発行された後は、新しい接続は許可されません。
- データベースが停止される前に、Oracle は現在データベースに接続しているすべてのユーザーが切断されるのを待ちます。

次にデータベースを起動するときに、インスタンス・リカバリ手順は必要ありません。

IMMEDIATE オプションによる停止

データベースの即時停止は、次のような状況のときにのみ使用します。

- 自動バックアップおよび無人バックアップを開始する場合
- 電源が間もなく停止する場合
- データベースやデータベース・アプリケーションの一部が異常に動作している場合で、ユーザーにログオフを依頼できない場合、またはユーザーがログオフできない場合

データベースを即時に停止するには、SHUTDOWN コマンドで IMMEDIATE オプションを指定します。

```
SHUTDOWN IMMEDIATE
```

このデータベース停止では、次のように処理が進みます。

- 文が発行された後は、新しい接続や新しいトランザクションの開始は許可されません。
- コミットされていないトランザクションはすべてロールバックされます。（コミットされていない大規模なトランザクションが存在する場合、この停止方法では、その名前に反してただちに完了しないことがあります。）
- 現在データベースに接続しているユーザーが切断されるのを待ちません。アクティブなトランザクションは暗黙的にロールバックされ、接続中のユーザーはすべて切断されます。

次にデータベースを起動するときに、インスタンス・リカバリ手順は必要ありません。

TRANSACTIONAL オプションによる停止

アクティブなトランザクションを完了してから、予定どおりにインスタンスを停止する場合は、SHUTDOWN コマンドで TRANSACTIONAL オプションを指定します。

```
SHUTDOWN TRANSACTIONAL
```

このデータベース停止では、次のように処理が進みます。

- 文が発行された後は、新しい接続や新しいトランザクションの開始は許可されません。
- すべてのトランザクションが完了すると、まだインスタンスに接続されているすべてのクライアントが切断されます。
- この時点で、インスタンスは SHUTDOWN IMMEDIATE 文を発行した場合と同じように停止します。

次にデータベースを起動するときに、インスタンス・リカバリ手順は必要ありません。

TRANSACTIONAL オプションによる停止では、クライアントの作業内容が失われずに済み、すべてのユーザーがログオフする必要がなくなります。

ABORT オプションによる停止

データベースのインスタンスを強制終了することにより、データベースをただちに停止できます。このタイプの停止は、次のような状況でのみ実行してください。

- データベースまたはデータベース・アプリケーションの一部が異常に動作している場合で、他のタイプの停止がどれも機能しないとき
- データベースを即時停止する必要がある場合（たとえば、電源停止が 1 分以内に起こることがわかっている場合）
- データベース・インスタンスを起動するときに問題が発生した場合

トランザクションとユーザーの接続を強制終了してデータベースを停止する場合は、次のように SHUTDOWN コマンドで ABORT オプションを指定します。

```
SHUTDOWN ABORT
```

このデータベース停止では、次のように処理が進みます。

- 文が発行された後は、新しい接続や新しいトランザクションの開始は許可されません。
- Oracle によって処理されている現行のクライアント SQL 文がただちに終了します。
- コミットされていないトランザクションはロールバックされません。
- 現在データベースに接続しているユーザーが切断されるのを待ちません。接続中のユーザーはすべて暗黙的に切断されます。

次にデータベースを起動するときに、インスタンス・リカバリ手順が必要になります。

データベースの静止

場合によっては、データベースを、DBA によるトランザクション、問合せ、フェッチまたは PL/SQL 文の実行のみ許可された状態にする必要があります。このような状態は、システム上で DBA 以外によるトランザクション、問合せ、フェッチまたは PL/SQL 文が実行されていないという意味で、静止状態と呼びます。データベースを静止状態にすると、それ以外の状態では安全に実行できない管理処理を DBA が実行できます。これらの処理は次のように分類されます。

- 同時ユーザー・トランザクションが同じオブジェクトにアクセスした場合に、失敗する可能性がある処理。たとえば、データベース表のスキーマの変更や、NOWAIT ロックが必要な既存表への列の追加などがこれに該当します。
- 途中で、同時ユーザー・トランザクションに望ましくない影響を与えるおそれがある処理。たとえば、最初に表をエクスポートし、削除してから、最後にインポートするというように、複数の手順で表を再編成する場合などがこれに該当します。同時ユーザーが表を削除してからインポートするまでの間に表にアクセスしようすると、望ましくない結果が生じます。

データベースの静止機能がない場合は、データベースの停止と制限モードでの再オープンが必要になります。これは、特に 24 時間、365 日の可用性が必要なシステムにとっては重大な制限です。データベースを静止させると、データベースの停止と再起動に伴うユーザーの遮断と停止時間の発生を回避できるので、制限が大幅に緩和されます。

注意： Oracle9*i* のこのリリースの場合、静止中データベース・コンテキストでは、DBA がユーザー `SYS` または `SYSTEM` として定義されています。`DBA` ロールを持つユーザーを含む他のユーザーには、`ALTER SYSTEM QUIESCE DATABASE` 文の発行や、データベース静止後の処理の継続は許可されていません。

データベースの静止状態への変更

データベースを静止状態にするには、次の文を発行します。

```
ALTER SYSTEM QUIESCE RESTRICTED;
```

DBA 以外のアクティブなセッションはすべて、非アクティブになるまで処理が継続されます。アクティブなセッションとは、現在トランザクション、問合せ、フェッチまたは PL/SQL 文を実行中のセッション、または現在なんらかの共有リソース（エンキューなど）を保持しているセッションとして定義されます。非アクティブなセッションがアクティブになることはできません。たとえば、ユーザーが非アクティブなセッションを強制的にアクティブにさせようとして SQL 問合せを発行すると、その問合せは停止したようになります。後でデータベースが静止解除されると、セッションが再開され、ブロックされていた処理（たとえば、前述の SQL 問合せ）が実行されます。

DBA 以外のセッションがすべて非アクティブになると、ALTER SYSTEM QUIESCE RESTRICTED 文が完了し、データベースは静止状態とみなされます。Oracle Real Application Clusters 環境でこの文を発行すると、文を発行したインスタンスだけでなく、すべてのインスタンスが影響を受けます。

注意： ALTER SYSTEM QUIESCE RESTRICTED 文を正常に発行するには、Database Resource Manager 機能をアクティブにしておく必要があります。この機能は、インスタンス（Oracle Real Application Clusters 環境ではすべてのインスタンス）の起動時以降、継続的にアクティブであることが必要です。DBA 以外のセッションがアクティブにならないようにするために、Database Resource Manager の機能が使用されています。また、この文が有効な間に現行のリソース・プランを変更しようとすると、その処理はシステムが静止解除されるまでキューに待機します。

Database Resource Manager の詳細は、第 27 章「Database Resource Manager の使用」を参照してください。

ALTER SYSTEM QUIESCE RESTRICTED 文は、アクティブなセッションが非アクティブになるまで、長時間待機する場合があります。DBA が要求を中断した場合、またはアクティブなセッションがすべて静止する前になんらかの理由でセッションが異常終了した場合は、この文による部分的な影響がすべて自動的に取り消されます。

複数の Oracle Call Interface（OCI）の連続したフェッチによって問合せが実行されている場合、ALTER SYSTEM QUIESCE RESTRICTED 文はすべてのフェッチが完了するまで待機しません。現行のフェッチの完了のみを待機します。

専用サーバー接続の場合も、共有サーバー接続の場合も、この文の発行後、DBA 以外のユーザーがログインしようとすると、その処理は Database Resource Manager によってすべてキューに送られ、進行しません。ユーザーにはログインが停止したように見えます。データベースが静止解除されると、ログインは再開されます。

文を発行したセッションが終了しても、データベースは静止状態のままです。DBA は、データベースを静止解除される文を明示的に発行するために、データベースにログインする必要があります。

静止状態の間は、ファイル・システム・コピーを使用して、コールド・バックアップとしてデータベースのデータ・ファイルをバックアップすることができません。これは、各インスタンスでチェックポイントを実行している場合でも同様です。その理由は、静止状態時にオンライン・データ・ファイルのファイル・ヘッダーを見ると、常にデータ・ファイルがアクセス中であるように見えるためです。このファイル・ヘッダーの状態は、データベースを正しく停止した場合とは異なります。同様に、データベースが静止状態の間にオンライン表領域のデータ・ファイルのホット・バックアップを実行するには、最初に ALTER TABLESPACE... BEGIN BACKUP 文を使用して表領域をバックアップ・モードにする必要があります。

通常操作へのシステムのリストア

次の文は、データベースを通常の操作にリストアさせます。

```
ALTER SYSTEM UNQUIESCE;
```

DBA 以外のアクティビティの処理がすべて許可されます。Oracle Real Application Clusters 環境では、データベースを静止状態に変更したのと同じセッションまたは同じインスタンスでこの文を発行する必要はありません。ALTER SYSTEM UNQUIESCE 文を発行したセッションが異常終了した場合、Oracle データベース・サーバーは静止解除を確実に完了させます。

インスタンスの静止状態の表示

V$INSTANCE ビューを問い合せると、インスタンスの現在の状態を表示できます。このビューには ACTIVE_STATE という列があります。値は次の表のとおりです。

ACTIVE_STATE	**説明**
NORMAL	通常の静止していない状態。
QUIESCING	静止途中の状態。まだ DBA 以外のアクティブなセッションが動作している。
QUIESCED	静止状態。DBA 以外のセッションは非アクティブで、アクティブになることができない。

データベースの一時停止と再開

ALTER SYSTEM SUSPEND 文は、データ・ファイル（ファイル・ヘッダーとファイル・データ）および制御ファイルへの入出力（I/O）をすべて停止して、データベースを一時停止します。これにより、I/O に干渉されずにデータベースのバックアップを作成できます。データベースを一時停止すると、実行中のすべての I/O 操作の完了が許可され、新しいデータベース・アクセスはキューに待機した状態になります。

SUSPEND コマンドは、インスタンスではなくデータベースを一時停止します。したがって、Oracle Real Application Clusters 環境では、あるシステムで SUSPEND コマンドを入力すると、内部ロッキング・メカニズムを通じてインスタンス間で停止要求が伝播し、特定のクラスタのアクティブ・インスタンスがすべて停止します。ただし、新しいインスタンスは一時停止されないため、インスタンスの一時停止中に新しいインスタンスを起動しないでください。

通常のデータベース操作を再開するには、ALTER SYSTEM RESUME 文を使用します。SUSPEND と RESUME は、異なるインスタンスから指定できます。たとえば、インスタンス 1、2 および 3 の実行中に、インスタンス 1 から ALTER SYSTEM SUSPEND 文を発行した場合は、インスタンス 1、2 または 3 から同様に RESUME 文を発行できます。

一時停止 / 再開機能は、ディスクやファイルをミラー化してそのミラーを分割できるシステムで役立ち、バックアップとリストアの代替ソリューションを提供します。書込み中に既存のデータベースからミラー化されたディスクを分割できないシステムを使用している場合は、この一時停止 / 再開機能を使用すると容易に分割できます。

ただし、一時停止したデータベースのコピーにはコミット前の更新が含まれるため、一時停止 / 再開機能は通常の停止操作の簡易版ではありません。

注意： 表領域をホット・バックアップ・モードに設定する代替手段として ALTER SYSTEM SUSPEND 文を使用しないでください。データベースの一時停止操作ではなく、ALTER TABLESPACE BEGIN BACKUP 文を使用してください。

次の文は、ALTER SYSTEM SUSPEND/RESUME の使用方法を示しています。データベースの状態を確認するために、V$INSTANCE ビューを問い合せています。

```
SQL> ALTER SYSTEM SUSPEND;
System altered
SQL> SELECT DATABASE_STATUS FROM V$INSTANCE;
DATABASE_STATUS
---------
SUSPENDED

SQL> ALTER SYSTEM RESUME;
System altered
SQL> SELECT DATABASE_STATUS FROM V$INSTANCE;
DATABASE_STATUS
---------
ACTIVE
```

関連項目： データベースの一時停止 / 再開機能を使用してデータベースをバックアップする方法の詳細は、『Oracle9*i* 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。

第 II 部

Oracle サーバー・プロセスと記憶域構造

第 II 部では、Oracle データベース・サーバー・プロセスと、その操作をサポートする基盤のデータベース記憶域構造について説明します。第 II 部の構成は、次のとおりです。

5

Oracle プロセスの管理

この章では、Oracle インスタンスのプロセスを管理する方法について説明します。この章の内容は、次のとおりです。

- サーバー・プロセス
- Oracle の共有サーバー構成
- Oracle バックグラウンド・プロセスの概要
- Oracle インスタンスのプロセスの監視
- パラレル実行用プロセスの管理
- 外部プロシージャのプロセスの管理
- セッションの停止

サーバー・プロセス

Oracle では、インスタンスに接続されているユーザー・プロセスの要求を処理するために、サーバー・プロセスが作成されます。サーバー・プロセスには、単一のサーバー・プロセス・サービスが単一のユーザー・プロセスのみを処理する**専用サーバー・プロセス**と、データベース・サーバーが**共有サーバー**用に構成されている場合に、単一のサーバー・プロセスが複数のユーザー・プロセスを処理できる**共有サーバー・プロセス**があります。

関連項目：『Oracle9*i* データベース概要』

専用サーバー・プロセス

図 5-1「Oracle 専用サーバー・プロセス」は、専用サーバー・プロセスの動作の仕組みを示しています。この図では、専用サーバー・プロセスを介して、2 つのユーザー・プロセスが Oracle に接続されています。

一般的には、共有サーバーを使用し、ディスパッチャを介して接続するほうがよいとされています。これは、図 5-2「Oracle 共有サーバー・プロセス」に図示されています。共有サーバー・プロセスは、実行中のインスタンスに必要なプロセスの数を少なくすることができるため、効率が向上します。

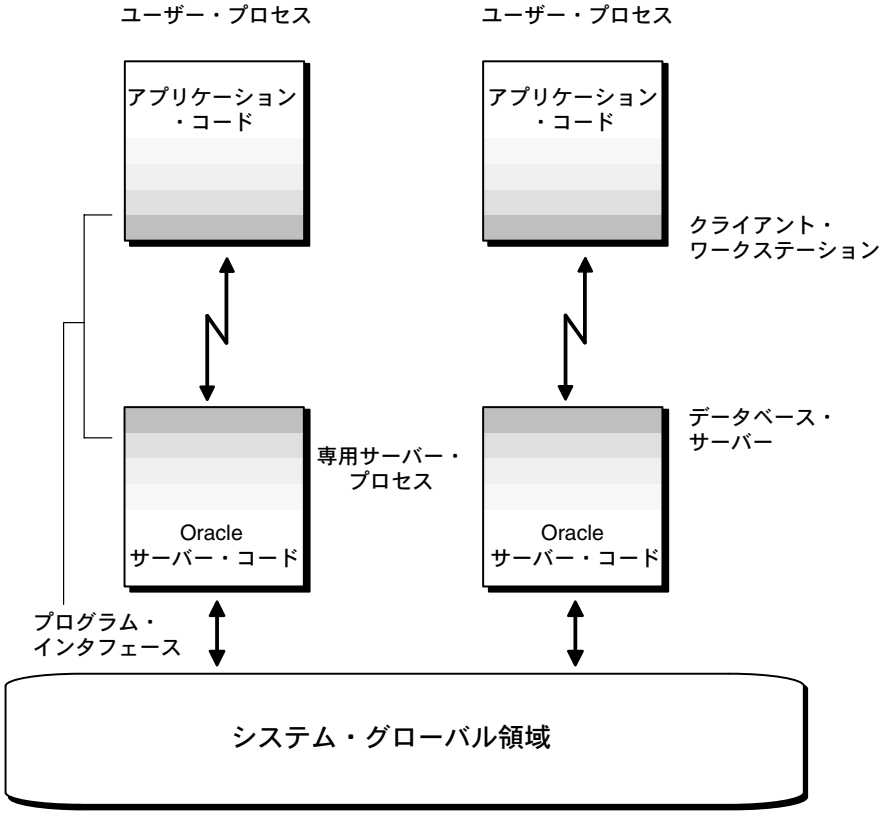
ただし、次の状況では、ユーザーと管理者は、専用サーバー・プロセスを使用して明示的にインスタンスに接続する必要があります。

- バッチ・ジョブを実行する場合（たとえば、ジョブがサーバー・プロセスに対して持つアイドル時間がほとんどないか、まったくない場合）
- Recovery Manager を使用して、データベースをバックアップ、リストアまたはリカバリする場合

Oracle が共有サーバー用に構成されている場合に専用サーバー接続を要求するには、専用サーバーを使用するように構成されているネット・サービス名を使用して接続する必要があります。具体的に言うと、ネット・サービス名の接続記述子に `SERVER=DEDICATED` 句を含めます。

関連項目： 専用サーバー接続を要求する方法の詳細は、『Oracle9*i* Net Services 管理者ガイド』を参照してください。

図 5-1　Oracle 専用サーバー・プロセス

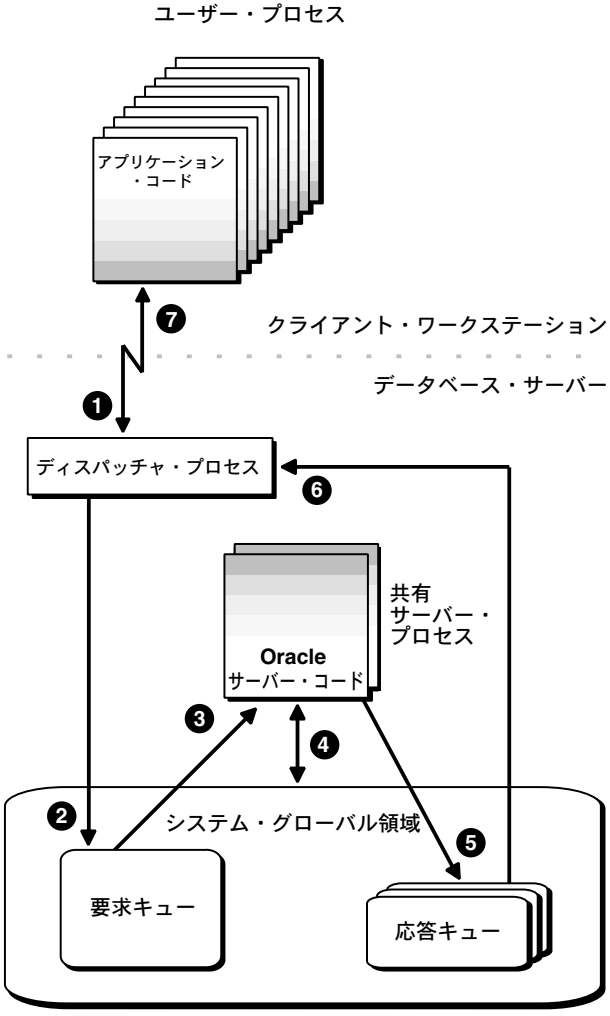


共有サーバー・プロセス

専用サーバー・プロセスを使用した注文入力システムを例にとって考えてみます。顧客からの発注を受けると、担当者がオーダーをデータベースに入力します。トランザクションのほとんどの間、担当者は顧客と電話で話しており、担当者のユーザー・プロセス専用のサーバー・プロセスはアイドル状態のままです。サーバー・プロセスはトランザクションのほとんどの間必要とされていないにもかかわらず、アイドル状態のサーバー・プロセスがシステム・リソースを保持したままなので、他の担当者がオーダーを入力する際にシステムのパフォーマンスが低下します。

共有サーバー・アーキテクチャでは、接続ごとに専用サーバー・プロセスは必要ありません（図 5-2 を参照）。

図 5-2　Oracle 共有サーバー・プロセス



共有サーバー構成では、クライアントのユーザー・プロセスはディスパッチャに接続します。ディスパッチャには、同時に複数のクライアント接続をサポートする機能があります。各クライアント接続は、バーチャル・サーキットにバインドされます。バーチャル・サーキットとは、ディスパッチャがクライアントのデータベース接続要求と応答に使用する共有メモリーの一部です。ディスパッチャは、要求が到着すると、バーチャル・サーキットを共通キューに入れます。

アイドル状態の共有サーバーは、共通キューからバーチャル・サーキットを選択して要求を処理し、そのバーチャル・サーキットを解放して、共通キューから別のバーチャル・サーキットを取り出します。このアプローチでは、小さいサーバー・プロセス・プールで大量のクライアントを処理することが可能です。専用サーバー・モデルと比較した共有サーバー・アーキテクチャの大きな利点は、システム・リソースが少なくて済むため、ユーザー数の増加に対応できることです。

共有サーバー・アーキテクチャには、Oracle Net Services が必要です。共有サーバーを使用するユーザー・プロセスは、Oracle インスタンスと同じマシン上にある場合でも、必ず Oracle Net Services を介して接続してください。

システムを共有サーバー用に構成するには、いくつかの必要な操作があります。次の項では、各操作について説明します。

> **関連項目：** 接続プーリングなどの追加機能を含む共有サーバーの詳細は、『Oracle9*i* Net Services 管理者ガイド』を参照してください。

Oracle の共有サーバー構成

共有サーバーをアクティブにするには、データベース初期化パラメータを設定します。共有サーバーでは、Oracle Net Services のリスナー・プロセスがアクティブである必要があります。ここでは、共有サーバーに関する初期化パラメータの設定と変更方法について説明します。

この項の内容は、次のとおりです。

- 共有サーバー用初期化パラメータ
- 初期ディスパッチャ数（DISPATCHERS）の設定
- 初期共有サーバー数（SHARED_SERVERS）の設定
- ディスパッチャ・プロセスとサーバー・プロセスの変更
- 共有サーバーの監視

共有サーバー用初期化パラメータ

共有サーバーを制御する初期化パラメータは、次のとおりです。

パラメータ	説明
必須：	
DISPATCHERS	共有サーバー・アーキテクチャのディスパッチャ・プロセスを構成します。
オプション（次のパラメータを指定しないと、デフォルトが選択されます）：	
MAX_DISPATCHERS	同時に実行可能なディスパッチャ・プロセスの最大数を指定します。
SHARED_SERVERS	インスタンスの起動時に作成する共有サーバー・プロセスの数を指定します。
MAX_SHARED_SERVERS	同時に実行可能な共有サーバー・プロセスの最大数を指定します。
CIRCUITS	受信および発信用のネットワーク・セッションに使用可能なバーチャル・サーキットの合計数を指定します。
SHARED_SERVER_SESSIONS	許可される共有サーバー・ユーザー・セッションの合計数を指定します。このパラメータを設定すると、専用サーバー用のユーザー・セッションを予約できます。
共有サーバーの影響を受け、調整を必要とする他のパラメータ：	
LARGE_POOL_SIZE	ラージ・プール割当てヒープのバイト数を指定します。共有サーバーでは、デフォルト値の設定が大きすぎると、パフォーマンスが低下したり、データベースの起動時に問題が発生したりすることがあります。
SESSIONS	システムで作成可能な最大セッション数を指定します。共有サーバーにあわせて調整が必要な場合があります。

関連項目：

- 『Oracle9*i* Net Services リファレンス・ガイド』
- 『Oracle9*i* データベース・リファレンス』

初期ディスパッチャ数（DISPATCHERS）の設定

インスタンス起動時に開始されるディスパッチャ・プロセスの数は、DISPATCHERS 初期化パラメータによって制御されます。初期化ファイルには複数の DISPATCHERS パラメータを指定できますが、それらは互いに隣接していることが必要です。各 DISPATCHERS パラメータには、内部で INDEX 値が割り当てられるため、後で ALTER SYSTEM 文を使用して、特定の DISPATCHERS パラメータを参照できます。

各インスタンスに対するディスパッチャ・プロセスの適切な数は、必要なデータベースのパフォーマンス、プロセスごとの接続数に関するホスト・オペレーティング・システムの制限（オペレーティング・システムによって異なる）、およびネットワーク・プロトコルごとに必要な接続数に依存します。インスタンスは、データベース・システム上の同時ユーザー数と同数の接続を提供する必要があります。インスタンスの起動後には、必要に応じて追加のディスパッチャ・プロセスを開始できます。この操作については、5-8 ページの「ディスパッチャ・プロセスの追加と削除」を参照してください。

標準的なシステムの場合は、1000 接続に対し 1 つのディスパッチャという割合で十分ですが、端数は次の整数に切り上げてください。たとえば、ピーク時に 1500 の接続が予想される場合は、2 つのディスパッチャを構成する必要があります。構成するディスパッチャが多すぎるとパフォーマンスが低下するおそれがあるため、極端に大きく見積るのは無意味です。この割合を参考にして、個々の状況にあわせてチューニングしてください。

次に、DISPATCHERS 初期化パラメータの設定例をいくつか示します。

例 :Typical

これは、DISPATCHERS 初期化パラメータの標準的な設定例です。

```
DISPATCHERS="(PROTOCOL=TCP)"
```

例 : ディスパッチャが使用する IP アドレスの設定

ディスパッチャが使用する IP アドレスを設定するには、次のように入力します。

```
DISPATCHERS="(ADDRESS=(PROTOCOL=TCP)\
     (HOST=144.25.16.201))(DISPATCHERS=2)"
```

この例では、指定した IP アドレスで 2 つのディスパッチャがリスニングを開始します。これは、インスタンスが稼働しているホストの有効な IP アドレスである必要があります。

例 : ディスパッチャが使用するポートの設定

ディスパッチャの位置を正確に設定するには、次のように PORT を追加します。

```
DISPATCHERS="(ADDRESS=(PROTOCOL=TCP)(PORT=5000))"
DISPATCHERS="(ADDRESS=(PROTOCOL=TCP)(PORT=5001))"
```

初期共有サーバー数（SHARED_SERVERS）の設定

SHARED_SERVERS パラメータでは、インスタンスの起動時に作成する共有サーバー・プロセスの数を指定します。共有サーバー・プロセス数は、要求キューの長さに基づいて動的に調整されます。作成できる共有サーバー・プロセスの数の上限と下限は、初期化パラメータ SHARED_SERVERS および MAX_SHARED_SERVERS の値で指定します。標準的なシステムは、多くの場合、10 接続ごとに共有サーバー 1 つという割合で安定します。

OLTP アプリケーションでは、接続数 / サーバー数の割合が高くなります。このようになるのは、要求率が低い場合、または要求に対するサーバー使用の比率が低い場合です。逆に、要求率が高いか、要求に対するサーバー使用の比率が大きいアプリケーションの場合は、接続数 / サーバー数の割合が低くなります。

MAX_SHARED_SERVERS は、使用するアプリケーションに基づいて適切な値に設定してください。SHARED_SERVERS および MAX_SHARED_SERVERS には、標準的な構成に合わせた適切なデフォルト値が用意されていますが、これらの設定に最適な値はアプリケーションごとに異なります。

注意： Windows NT の場合、各サーバーは共通プロセス内のスレッドであるため、MAX_SHARED_SERVERS を大きい値に設定する場合は注意が必要です。

MAX_SHARED_SERVERS は静的初期化パラメータであるため、変更するにはデータベースを停止する必要があります。しかし、SHARED_SERVERS は動的初期化パラメータであるため、ALTER SYSTEM 文を使用して変更できます。

ディスパッチャ・プロセスとサーバー・プロセスの変更

DISPATCHERS および SHARED_SERVERS の設定は、インスタンスの稼働中に動的に変更できます。ALTER SYSTEM 権限を持っている場合は、ALTER SYSTEM 文を使用して動的な変更が可能です。

関連項目： ALTER SYSTEM 文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

ディスパッチャ・プロセスの追加と削除

特定のインスタンスに対するディスパッチャ・プロセス数を制御できます。V$QUEUE、V$DISPATCHER および V$DISPATCHER_RATE ビューの監視により、ディスパッチャ・プロセスに対する負荷が一貫して高いことが判明した場合は、追加のディスパッチャ・プロセスを起動してユーザー要求をルーティングすると、パフォーマンスを改善できます。逆に、ディスパッチャの負荷が一貫して低い場合は、ディスパッチャの数を少なくすることにより、パフォーマンスを改善できます。

ディスパッチャ・プロセス数を変更するには、SQL 文 ALTER SYSTEM を使用します。既存の DISPATCHERS 値に対して新しいディスパッチャ・プロセスを起動する方法と、新しい DISPATCHERS 値を追加する方法があります。ディスパッチャは、MAX_DISPATCHERS で指定した上限の数まで追加できます。

特定の共有サーバー・ディスパッチャ値に対するディスパッチャ数を少なくしても、ディスパッチャが即時に削除されるわけではありません。ユーザーの切断にあわせてディスパッチャが終了し、最終的に DISPATCHERS で指定した制限に達するまでディスパッチャが減ります。

次の文では、TCP/IP プロトコル用のディスパッチャ・プロセス数が 5 に動的に変更され、SSL 付き TCP/IP（TCPS）プロトコル用のディスパッチャ・プロセスが追加されます。TCPS プロトコルに関する DISPATCHERS 初期化パラメータは設定されていない（DISPATCHERS パラメータは TCP プロトコル用のものが 1 つしかなかった）ため、この文ではディスパッチャ・パラメータが 1 つ追加されます。

```
ALTER SYSTEM
SET DISPATCHERS =
'(PROTOCOL=TCP)(DISPATCHERS=5) (INDEX=0)',
'(PROTOCOL=TCPS)(DISPATCHERS=2) (INDEX=1)';
```

現在起動している TCP 用のディスパッチャ・プロセスが 4 以下の場合は、新規作成されます。現行の数が 5 以上の場合は、その一部が接続ユーザーの切断後に終了します。

注意： INDEX キーワードを使用すると、変更する DISPATCHERS パラメータを識別できます。INDEX 値の範囲は 0 ～ *n* で、*n* は定義済みの DISPATCHERS パラメータの数から 1 を差し引いた値です。ALTER SYSTEM 文で INDEX 値に *n+1* を指定すると（*n* は現行のディスパッチャの数）、新しい DISPATCHERS パラメータが追加されます。DISPATCHERS パラメータに割り当てられている索引番号を識別するには、V$DISPATCHER ビューの CONF_INDX 値を問い合せます。

特定のディスパッチャ・プロセスの停止

特定のディスパッチャ・プロセスを停止できます。停止するディスパッチャ・プロセスの名前を識別するには、V$DISPATCHER 動的パフォーマンス・ビューを使用します。

```
SELECT NAME, NETWORK FROM V$DISPATCHER;

NAME  NETWORK
----  ----------------------------------------------------------------
D000  (ADDRESS=(PROTOCOL=tcp)(HOST=rbaylis-hpc.us.oracle.com)(PORT=3499))
D001  (ADDRESS=(PROTOCOL=tcp)(HOST=rbaylis-hpc.us.oracle.com)(PORT=3531))
D002  (ADDRESS=(PROTOCOL=tcp)(HOST=rbaylis-hpc.us.oracle.com)(PORT=3532))
```

各ディスパッチャは、D*nnn* 形式の名前で一意に識別されます。

ディスパッチャ D002 を停止するには、次の文を発行します。

```
ALTER SYSTEM SHUTDOWN IMMEDIATE 'D002';
```

IMMEDIATE キーワードを指定すると、ディスパッチャによる新規接続の受入れが停止し、そのディスパッチャを介した既存のすべての接続が即時に終了します。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・プロセスが停止します。IMMEDIATE を指定しなかった場合、ディスパッチャは、接続ユーザーがすべて切断され、接続がすべて終了するまで待ってから停止します。

共有サーバー・プロセスの最小数の変更

インスタンスを起動した後は、SQL 文 ALTER SYSTEM を使用して共有サーバー・プロセスの最小数を変更できます。指定した最小限度より多くの共有サーバーが起動している場合は、最終的に、アイドル状態のサーバーが終了します。

SHARED_SERVERS を 0 に設定した場合は、現行の共有サーバーすべてがアイドル状態になると、Oracle はこれらのサーバーをすべて終了させ、ユーザーが SHARED_SERVERS の値を大きくしないかぎり新しくサーバーを起動しません。つまり、SHARED_SERVERS を 0 に設定することによって、共有サーバーを効果的に使用禁止にすることができます。

次の文は、共有サーバー・プロセスの最小数を動的に 2 に設定します。

```
ALTER SYSTEM SET SHARED_SERVERS = 2;
```

共有サーバーの監視

次のビューは、共有サーバー構成情報の取得やパフォーマンスの監視に使用できます。

ビュー	説明
V$DISPATCHER	名前、ネットワーク・アドレス、状態、各種使用統計、索引番号などのディスパッチャ・プロセス情報を提供します。
V$DISPATCHER_RATE	ディスパッチャ・プロセスのレート統計が含まれています。
V$QUEUE	共有サーバー・メッセージ・キューについての情報が含まれています。
V$SHARED_SERVER	共有サーバー・プロセスについての情報が含まれています。
V$CIRCUIT	バーチャル・サーキットについての情報が含まれています。バーチャル・サーキットとは、ディスパッチャおよびサーバーを介したデータベースへのユーザー接続です。
V$SHARED_SERVER_MONITOR	共有サーバーのチューニング情報が含まれています。

ビュー	説明
V$SGA	各種のシステム・グローバル領域（SGA）グループに関するサイズ情報が含まれています。この情報は、共有サーバーのチューニング時に役立ちます。
V$SGASTAT	チューニングに役立つ SGA 関連の詳細な統計情報が含まれています。
V$SHARED_POOL_RESERVED	共有プール内で予約済みのプールと領域をチューニングする際に役立つ統計リストが含まれています。

関連項目：

- これらのビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。
- 共有サーバーの監視とチューニングの詳細は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。

Oracle バックグラウンド・プロセスの概要

マルチプロセスの Oracle システムでは、最高のパフォーマンスを提供し、多くのユーザーが同時に使用できるように、「バックグラウンド・プロセス」と呼ばれるいくつかの追加プロセスが使用されています。バックグラウンド・プロセスでは、ユーザー・プロセスごとに実行される複数の Oracle プログラムによって処理される機能が統合されます。バックグラウンド・プロセスは、非同期的に I/O を実行して他の Oracle プロセスを監視することにより、並列性を高め、パフォーマンスと信頼性を向上させます。

次の表は、基本的な Oracle バックグラウンド・プロセスを示しています。これらの多くは、このマニュアルの別の箇所に詳細な解説があります。Oracle データベースの機能またはオプションを追加すると、バックグラウンド・プロセスの数が増える場合があります。たとえば、アドバンスト・キューイングを使用している場合は、キュー・モニター（QMN*n*）バックグラウンド・プロセスが存在します。また、データ・ファイルを記憶域サブシステムの物理デバイスにマッピングするために FILE_MAPPING 初期化パラメータを指定している場合は、LMON プロセスが存在します。

プロセス名	説明
データベース・ライター（DBW*n*）	データベース・ライターは、変更があったブロックをデータベース・バッファ・キャッシュからデータ・ファイルに書き込みます。Oracle では、最大 20 のデータベース・ライター・プロセス（DBW0 ～ DBW9 および DBWa ～ DBWj）を使用できます。DBW*n* プロセスの数は、初期化パラメータ `DB_WRITER_PROCESSES` で指定します。Oracle では、CPU 数とプロセッサ・グループ数に基づいて、この初期化パラメータに適切なデフォルト設定が選択（またはユーザー指定の設定が調整）されます。 `DB_WRITER_PROCESSES` 初期化パラメータの設定の詳細は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。
ログ・ライター（LGWR）	ログ・ライター・プロセスは、REDO ログ・エントリをディスクに書き込みます。REDO ログ・エントリは、SGA の REDO ログ・バッファ内で生成され、LGWR によってオンライン REDO ログ・ファイルに順次書き込まれます。データベースの REDO ログが多重化されている場合、LGWR は REDO ログ・エントリをオンライン REDO ログ・ファイルのグループに書き込みます。ログ・ライター・プロセスの詳細は、第 7 章「オンライン REDO ログの管理」を参照してください。
チェックポイント（CKPT）	SGA 内で変更があったすべてのデータベース・バッファは、特定の時点で DBW*n* によってデータ・ファイルに書き込まれます。このイベントはチェックポイントと呼ばれます。チェックポイント・プロセスは、最新のチェックポイントを示すために、チェックポイントで DBW*n* にシグナルを出し、データベースのすべてのデータ・ファイルと制御ファイルを更新する役割を持ちます。
システム・モニター（SMON）	システム・モニターは、障害の発生したインスタンスの再起動時にクラッシュ・リカバリを実行します。クラスタ・データベース（Oracle9*i* Real Application Clusters）では、1 つのインスタンスの SMON プロセスが、障害を起こした他のインスタンスのインスタンス・リカバリを実行できます。また、SMON により、不要になった一時セグメントがクリーン・アップされ、ファイル読込みエラーやオフライン・エラーのためにクラッシュ・リカバリやインスタンス・リカバリ時にスキップされたデッド・トランザクションがリカバリされます。これらのトランザクションは、表領域またはファイルがオンラインに戻るときに、SMON によって最後にリカバリされます。 さらに、SMON は、領域の割当てを容易にするために、データベースのディクショナリ管理表領域内の使用可能エクステントを結合して連続する空き領域を作成します（11-15 ページの「ディクショナリ管理表領域の空き領域の結合」を参照してください）。
プロセス・モニター（PMON）	プロセス・モニターは、ユーザー・プロセスが失敗したときにプロセス・リカバリを実行します。PMON は、キャッシュをクリーン・アップし、プロセスで使用されていたリソースを解放する役割を持ちます。また、ディスパッチャ・プロセス（後述）とサーバー・プロセスをチェックし、障害プロセスがある場合は再起動します。

プロセス名	説明
アーカイバ（ARC*n*）	オンライン REDO ログ・ファイルは、ログ・ファイルがいっぱいになるか、ログ・スイッチが発生すると、1 つ以上のアーカイバ・プロセスによってアーカイブ記憶域にコピーされます。アーカイバ・プロセスについては、第 8 章「アーカイブ REDO ログの管理」を参照してください。
リカバラ（RECO）	リカバラ・プロセスは、ネットワーク障害やシステム障害が原因で分散データベース内で保留されている分散トランザクションを解決するために使用されます。ローカル RECO は、一定の間隔でリモート・データベースに接続し、保留されている分散トランザクションのローカル部分のコミットまたはロールバックを自動的に完了しようとします。このプロセスの詳細と開始方法は、第 32 章「分散トランザクションの管理」を参照してください。
ディスパッチャ（D*nnn*）	ディスパッチャは、オプションのバックグラウンド・プロセスです。これが存在するのは、共有サーバー構成を使用している場合のみです。共有サーバーについては、5-5 ページの「Oracle の共有サーバー構成」を参照してください。
グローバル・キャッシュ・サービス（LMS）	グローバル・キャッシュ・サービスは、Oracle Real Application Clusters 環境でリソースを管理し、インスタンス間のリソース制御機能を提供します。次のマニュアルを参照してください。 ■ 『Oracle9*i* Real Application Clusters 概要 』 ■ 『Oracle9*i* Real Application Clusters セットアップおよび構成』
コーディネータ・ジョブ・キュー・プロセス（CJQ0）	このプロセスは、インスタンスのジョブ・キュー・プロセスのコーディネータです。JOB$ 表（ジョブ・キュー内にあるジョブの表）を監視し、ジョブ実行のために必要に応じてジョブ・キュー・プロセス（J*nnn*）を起動します。J*nnn* プロセスは、DBMS_JOBS パッケージによって作成されたジョブ要求を実行します。詳細は、第 10 章「ジョブ・キューの管理」を参照してください。 また、最大 1000 の J*nnn* プロセスによって、マテリアライズド・ビューを自動的にリフレッシュできます。これらのプロセスは定期的に起動され、リフレッシュ・スケジュールが設定されているマテリアライズド・ビューをリフレッシュします。マテリアライズド・ビューの作成とリフレッシュについては、次のマニュアルを参照してください。 ■ 『Oracle9*i* レプリケーション』 ■ 『Oracle9*i* レプリケーション・マネージメント API リファレンス』 J*nnn* プロセスのもう 1 つの機能は、キューに入っているメッセージを他のデータベースのキューに伝播させることです。キューに入っているメッセージの伝播については、『Oracle9*i* アプリケーション開発者ガイドーアドバンスト・キューイング』を参照してください。 多くの Oracle バックグラウンド・プロセスとは異なり、ジョブ・キュー・プロセスまたはコーディネータ（CJQ0）が失敗してもインスタンス障害は発生しません。

関連項目： Oracle バックグラウンド・プロセスの詳細は、『Oracle9*i* データベース概要 』を参照してください。

Oracle インスタンスのプロセスの監視

ここでは、Oracle インスタンスの監視に使用できるいくつかのデータ・ディクショナリ・ビューについて説明します。これらのビューは一般的なものです。プロセス固有の他のビューについては、そのプロセスに関する項を参照してください。また、ここではロックの状態を監視するためのスクリプトとビューについても説明します。

関連項目：

- これらのビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。
- 『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス 』には、これらのビューの監視を通じて明らかになるパフォーマンス上の問題と競合の解決方法が記載されています。

プロセスおよびセッション・ビュー

これらのビューには、プロセスおよびセッション固有の情報が含まれています。

ビュー	説明
V$PROCESS	現在アクティブになっているプロセスの情報が含まれています。
V$SESSION	現行のセッションごとにセッション情報がリストされます。
V$SESS_IO	各ユーザー・セッションの I/O 統計情報が含まれています。
V$SESSION_LONGOPS	このビューには、6 秒（絶対時間）以上実行されていた各種操作の状態が表示されます。現在、状態が表示される操作は、多数のバックアップおよびリカバリ機能、統計収集および問合せの実行などです。Oracle のリリースごとにさらに操作が追加されます。
V$SESSION_WAIT	アクティブ・セッションが待機しているリソースまたはイベントが表示されます。
V$SYSSTAT	システム統計情報が含まれています。
V$RESOURCE_LIMIT	一部のシステム・リソースについて、現行および最大のグローバル・リソース使用率が表示されます。
V$SQLAREA	共有 SQL 領域に関する統計情報が、SQL 文字列ごとに 1 行ずつ含まれています。また、メモリー内にあり、解析済みで、実行準備のできている SQL 文に関する統計情報も提供します。
V$LATCH	非親ラッチの統計情報と、親ラッチのサマリー統計情報が含まれています。

ロックの監視

utllockt.sql スクリプトは、ロックを待機しているシステム上のセッションとそのロックをツリー構造の形式で表示します。このスクリプトは、SQL*Plus などの非定型問合せツールを使用して、ロックを待機しているシステム上のセッションと、対応するブロッキング・ロックを出力します。このスクリプト・ファイルの位置は、オペレーティング・システムによって異なります。オペレーティング・システム固有の Oracle のマニュアルを参照してください。第 2 のスクリプト catblock.sql は utllockt.sql に必要なロック・ビューを作成するスクリプトであるため、utllockt.sql を実行する前にを実行する必要があります。

ロックの監視に使用できるビューは、次のとおりです。

ビュー	説明
V$LOCK	現在 Oracle データベースによって保持されているロックと、未処理のロック要求またはラッチ要求が表示されます。

トレース・ファイルとアラート・ファイル

各サーバー・プロセスとバックグラウンド・プロセスは、対応する**トレース・ファイル**に情報を書き込むことができます。プロセスによって内部エラーが検出されると、エラー情報が関連トレース・ファイルにダンプされます。トレース・ファイルに書き込まれる情報の一部は DBA 用であり、その他の情報はオラクル社カスタマ・サポート・センター用です。また、トレース・ファイルの情報は、アプリケーションとインスタンスのチューニングにも使用されます。

アラート・ファイル（**アラート・ログ**）は特殊なトレース・ファイルです。データベースのアラート・ファイルはメッセージとエラーの履歴ログであり、次のような情報が含まれます。

- 発生したすべての内部エラー（ORA-600）、ブロック破損エラー（ORA-1578）およびデッドロック・エラー（ORA-60）
- たとえば、CREATE/ALTER/DROP 文、STARTUP/SHUTDOWN 文、ARCHIVELOG 文などの管理操作
- 共有サーバーとディスパッチャ・プロセスの機能に関係するメッセージとエラー
- マテリアライズド・ビューの自動リフレッシュ中に発生したエラー
- データベースおよびインスタンス起動時のすべての初期化パラメータの値

Oracle は、オペレータのコンソール上にこのような情報を表示するかわりに（多くのシステムがコンソール上に情報を表示する）、アラート・ファイルを使用して、これらの特別な操作のログを保持します。操作が成功すると、タイムスタンプとともに「completed」というメッセージがアラート・ファイルに書き込まれます。

トレース・ファイルの位置とサイズを制御する初期化パラメータは、次のとおりです .

- BACKGROUND_DUMP_DEST
- USER_DUMP_DEST
- MAX_DUMP_FILE_SIZE

次の項では、これらのパラメータについて説明します。

> **関連項目：** トレース・ファイルへの書込みを制御する初期化パラメータの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

トレース・ファイルの使用

バックグラウンド・プロセスでエラーが発生していないかを確認するために、インスタンスのアラート・ファイルとその他のトレース・ファイルを定期的にチェックする必要があります。たとえば、ログ・ライター・プロセス（LGWR）でグループのメンバーに書き込むことができないと、LGWR トレース・ファイルとデータベースのアラート・ファイルにエラー・メッセージが書き込まれます。このようなエラー・メッセージが見つかった場合は、メディアまたは I/O の問題が発生しているため、ただちに解決が必要です。

アラート・ファイルには、他の重要な統計に加えて、初期化パラメータの値も書き込まれます。たとえば、通常どおりに、または即時にインスタンスを停止すると、インスタンスの起動以降、インスタンスに同時に接続されたセッションの最大数がアラート・ファイルに書き込まれます。この数に基づいて、Oracle セッション・ライセンスをアップグレードする必要があるかどうかを確認できます。

トレース・ファイルの位置の指定

バックグラウンド・プロセスに対応するトレース・ファイルすべてとアラート・ファイルは、初期化パラメータ BACKGROUND_DUMP_DEST によって指定された保存先ディレクトリに書き込まれます。サーバー・プロセスに対応するすべてのトレース・ファイルは、初期化パラメータ USER_DUMP_DEST によって指定された保存先ディレクトリに書き込まれます。トレース・ファイルの名前はオペレーティング・システムによって異なりますが、通常は、ファイルに情報を書き込んでいるプロセスの名前（LGWR、RECO など）が含まれます。

> **関連項目：** トレース・ファイルの名前の詳細は、オペレーティング・システム固有の Oracle マニュアルを参照してください。

トレース・ファイル・サイズの制御

すべてのトレース・ファイル（アラート・ファイルを除く）の最大サイズは、初期化パラメータ MAX_DUMP_FILE_SIZE を使用して制御できます。この制限は、オペレーティング・システム・ブロックの数で設定します。アラート・ファイルのサイズを制御するには、不要になったファイルを手動で削除する必要があります。そうしないと、Oracle はファイルに情報を追加し続けます。インスタンスの実行中にアラート・ファイルを削除しても問題はありませんが、必要に応じて、削除する前にアラート・ファイルのアーカイブ・コピーを作成してください。

Oracle がトレース・ファイルに書き込む時期の制御

バックグラウンド・プロセスは適宜、トレース・ファイルに情報を書き込みます。ARC*n* バックグラウンド・プロセスの場合は、生成されるトレース情報の量とタイプを初期化パラメータで制御できます。この操作については、8-24 ページの「ARCHIVELOG プロセスによって生成されるトレース出力の制御」を参照してください。他のバックグラウンド・プロセスには、このような柔軟性はありません。

内部エラーが発生したときは必ず、サーバー・プロセスのためにトレース・ファイルに情報が書き込まれます。また、初期化パラメータ SQL_TRACE = TRUE を設定すると、SQL トレース機能が有効になり、インスタンスに対するすべての SQL 文の処理についてパフォーマンス統計が生成され、USER_DUMP_DEST ディレクトリに書き込まれます。

必要に応じて、ユーザー要求時にサーバー・プロセスに対するトレース・ファイルを生成できます。SQL 文 ALTER SESSION SET SQL_TRACE を使用すると、SQL_TRACE の現行の値にかかわらず、対応するサーバー・プロセスのために各セッションのトレース・ロギングを使用可能または使用禁止にできます。次の例は、特定の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;
```

> **注意：** サーバー・プロセスの SQL トレース機能は、著しいシステム・オーバーヘッドを引き起こし、パフォーマンスに重大な影響を及ぼします。統計を収集するときにのみ、この機能を使用可能にしてください。

共有サーバーでは、ディスパッチャを使用している各セッションは共有サーバー・プロセスに転送され、セッションでトレースが使用可能な場合（または、エラーが発生した場合）にかぎり、トレース情報がサーバーのトレース・ファイルに書き込まれます。ディスパッチャを使用して接続する特定のセッションのトレースを追跡するには、いくつかの共有サーバーのトレース・ファイルを調べることが必要な場合があります。

セッションの SQL トレースを制御するために、DBMS_SESSION および DBMS_SYSTEM パッケージも使用できます。

> **関連項目：** SQL トレース機能の使用方法、および TKPROF を使用して、生成されたトレース・ファイルを解析する方法の詳細は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。

パラレル実行用プロセスの管理

この項では、SQL 文のパラレル実行の管理方法について説明します。この構成では、SQL 文の処理作業を複数のパラレル・プロセスに分割できます。

多数の SQL 文の実行をパラレル化できます。**並列度**は、単一の処理に対応付け可能なパラレル実行サーバーの数で表されます。並列度は、次の要素によって決まります。

- 文の PARALLEL 句
- 問合せ内で参照されているオブジェクトの場合は、オブジェクトが作成または変更されたときに使用された PARALLEL 句
- 文に挿入されたパラレル・ヒント
- Oracle によって決定されたデフォルト

パラレル実行の使用例は、15-8 ページの「表作成のパラレル化」を参照してください。

この項の内容は、次のとおりです。

- パラレル実行サーバーの管理
- セッションのパラレル実行の変更

注意： この項に記載されているパラレル実行機能は、Oracle9*i* Enterprise Edition および Oracle9*i* Personal Edition で利用できます。

関連項目：

- パラレル実行に関する追加情報は、『Oracle9*i* データベース概要 』および『Oracle9*i* データ・ウェアハウス・ガイド』を参照してください。
- パラレル・ヒントの使用方法の詳細は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス 』を参照してください。

パラレル実行サーバーの管理

パラレル実行機能では、**パラレル実行コーディネータ**と呼ばれるプロセスが**パラレル実行サーバー**のプールの実行をディスパッチし、これらすべてのパラレル実行サーバーからユーザーへの結果の返信を調整します。1 つの文の実行フェーズ全体を通して、複数のパラレル実行サーバー・プロセスが対応付けられます。文の処理が完了すると、その文に対応付けられていたプロセスが他の文を処理できるようになります。

パラレル実行は、PARALLEL_AUTOMATIC_TUNING 初期化パラメータに TRUE を設定することによって、自動的にチューニングできます。このパラメータを設定すると、パラレル実

行のパフォーマンスに影響を与える他の初期化パラメータのデフォルト値が Oracle によって決定されます。

セッションのパラレル実行の変更

ALTER SESSION 文を使用して、セッションのパラレル実行を制御できます。

パラレル実行を使用禁止にする方法

ALTER SESSION DISABLE PARALLEL DML|DDL|QUERY 文を発行すると、後続のすべての DML（INSERT、UPDATE、DELETE）、DDL（CREATE、ALTER）または問合せ（SELECT）文はパラレル化されません。それらの文に PARALLEL 句やパラレル・ヒントを指定しても、それとは無関係に文は連続して実行されます。

次の文は、パラレル DDL を使用禁止にします。

```
ALTER SESSION DISABLE PARALLEL DDL;
```

パラレル実行を使用可能にする方法

ALTER SESSION ENABLE PARALLEL DML|DDL|QUERY 文を発行後、PARALLEL 句またはパラレル・ヒントを文に対応付けると、その DML、DDL または問合せはパラレルで実行されます。これは、DDL 文と問合せ文のデフォルトです。

DML 文は、この文を明示的に発行した場合のみ、パラレル化できます。次の文は、DML 文のパラレル処理を使用可能にします。

```
ALTER SESSION ENABLE PARALLEL DML;
```

注意： パラレル DML を使用できるのは、Oracle の Partitioning Option をインストールした場合のみです。

パラレル実行の強制

ALTER SESSION FORCE PARALLEL DML|DDL|QUERY 文を発行すると、後続のすべての DML、DDL または問合せ文をパラレルで実行するように強制できます。また、特定の並列度を強制的に適用して、後続の文に対応付けられた PARALLEL 句を無効にできます。この文で並列度を指定しなかった場合は、デフォルトの並列度が使用されます。ただし、ヒントによって文に指定された並列度は、強制された並列度を無効にします。

次の文は、後続の文のパラレル実行を強制し、優先する並列度を 5 に設定します。

```
ALTER SESSION FORCE PARALLEL DDL PARALLEL 5;
```

DML のパラレル化を強制するには、「パラレル実行を使用可能にする方法」で示されたようにパラレル化を使用可能にすることも必要です。

外部プロシージャのプロセスの管理

外部プロシージャとは、別のプログラムからコールされるプロシージャです。ただし、異なる言語で記述されています。たとえば、PL/SQL プログラムから、特別な用途の処理に必要な 1 つ以上の C ルーチンをコールする場合がその例です。

これらのコール可能ルーチンは、Dynamic Link Library（DLL）または Java クラス・メソッドの場合はライブラリ・ユニットに格納されており、ベース言語で登録されています。Oracle ではコール仕様という特別な用途のインタフェースが用意されており、ユーザーはこのインタフェースを使用して他の言語から外部プロシージャをコールできます。

簡単に言うと、外部プロシージャをコールするには、アプリケーションが、外部プロシージャを含む DLL または共有ライブラリを知っている必要があります。アプリケーションはネットワーク・リスナー・プロセスにアラートを送り、続いてネットワーク・リスナー・プロセスが外部プロシージャ・エージェント（デフォルト名は `extproc`）を起動します。アプリケーションは、リスナーによって確立されたネットワーク接続を使用して、外部プロシージャ・エージェントに DLL の名前、外部プロシージャの名前およびパラメータを渡します。次に、外部プロシージャ・エージェントは DLL をロードして外部プロシージャを実行し、外部プロシージャから返された値をアプリケーションに返送します。

エージェントは、データベース・サーバーと同じコンピュータ上に存在する必要があります。

DLL へのアクセスを制御するには、DBA がアプリケーション開発者に適切な DLL の実行権限を付与します。アプリケーション開発者は外部プロシージャを作成し、他のユーザーに特定の外部プロシージャの実行権限を付与します。

注意： 外部ライブラリ（DLL ファイル）は、静的にリンクする必要があります。つまり、他の外部ライブラリ（DLL ファイル）の外部シンボルを参照しないようにします。これらのシンボルは解決されないため、外部プロシージャが失敗する原因となります。

外部プロシージャをコールするための環境は `tnsnames.ora` および `listener.ora` エントリで構成され、データベースのインストール中にデフォルトで構成されます。セキュリティ・レベルを上げるには、追加のネットワーク構成手順の実行が必要になることがあります。その操作の詳細は、『Oracle9*i* Net Services 管理者ガイド』を参照してください。

関連項目： 外部プロシージャの詳細は、『Oracle9*i* アプリケーション開発者ガイドー基礎編 』を参照してください。

セッションの停止

状況によっては、現行のユーザー・セッションを停止できます。たとえば、管理操作を実行するため、管理に関係のないセッションをすべて停止する必要がある場合などです。

ここでは、セッションの停止について説明します。この項の内容は、次のとおりです。

- 停止するセッションの識別
- アクティブ・セッションの停止
- 非アクティブ・セッションの停止

セッションが停止すると、そのセッションのトランザクションがロールバックされ、そのセッションが保持していたリソース（ロックやメモリー領域など）がただちに解放されて、他のセッションで使用可能になります。

現行のセッションを停止するには、SQL 文 `ALTER SYSTEM KILL SESSION` を使用します。

次の文は、システム識別子が 7 でシリアル番号が 15 のセッションを停止します。

```
ALTER SYSTEM KILL SESSION '7,15';
```

停止するセッションの識別

停止するセッションを識別するには、セッションの索引番号とシリアル番号を指定します。セッションのシステム識別子（SID）とシリアル番号を識別するには、`V$SESSION` 動的パフォーマンス・ビューを問い合せます。

次の問合せは、ユーザー `jward` のすべてのセッションを識別します。

```
SELECT SID, SERIAL#, STATUS
  FROM V$SESSION
  WHERE USERNAME = 'JWARD';

SID    SERIAL#    STATUS
-----  ---------  --------
    7         15  ACTIVE
   12         63  INACTIVE
```

Oracle に対して SQL コールを実行しているとき、セッションは `ACTIVE` です。Oracle に対して SQL コールを実行していないとき、セッションは `INACTIVE` です。

> **関連項目：** セッションの状態値については、『Oracle9*i* データベース・リファレンス』を参照してください。

アクティブ・セッションの停止

停止時にユーザー・セッションがトランザクションを処理している場合（STATUS が ACTIVE）、トランザクションはロールバックされ、ユーザーはただちに次のメッセージを受け取ります。

```
ORA-00028: セッションは強制終了されました。
```

ユーザーが、ORA-00028 のメッセージを受け取った後、データベースに再接続する前に追加の文を実行すると、Oracle は次のメッセージを返します。

```
ORA-01012: ログオンされていません。
```

アクティブ・セッションを中断できない（ネットワーク I/O やトランザクションのロールバックを実行中の）場合は、操作が完了するまでそのセッションは停止できません。この場合、セッションは、停止するまですべてのリソースを保持します。また、セッションを停止させるために ALTER SYSTEM 文を発行したセッションは、対象のセッションが停止するまで最大 60 秒待機します。中断できなかった操作が 1 分たっても終了しない場合、ALTER SYSTEM 文の発行者は、セッションが停止されることを示すマークが設定されたというメッセージを受け取ります。停止マークが付けられたセッションは、V$SESSION での状態（STATUS）が KILLED になり、サーバー（SERVER）が PSEUDO 以外の値になります。

非アクティブ・セッションの停止

停止時にセッションが Oracle に対して SQL コールを実行していない場合（STATUS が INACTIVE）、ORA-00028 のメッセージはただちには返されません。このメッセージは、その後ユーザーが停止したセッションを使用しようとしたときに返されます。

非アクティブ・セッションを停止すると、V$SESSION ビューの STATUS が KILLED になります。停止したセッションの行は、ユーザーが再びそのセッションを使用しようとして ORA-00028 のメッセージを受け取った後で、V$SESSION から削除されます。

次の例では、非アクティブ・セッションを停止しています。最初に V$SESSION を問い合せてセッションの SID と SERIAL# を識別してから、セッションを停止しています。

```
SELECT SID,SERIAL#,STATUS,SERVER
   FROM V$SESSION
   WHERE USERNAME = 'JWARD';

SID    SERIAL#   STATUS     SERVER
-----  --------  ---------  ---------
    7        15  INACTIVE   DEDICATED
   12        63  INACTIVE   DEDICATED
2 rows selected.

ALTER SYSTEM KILL SESSION '7,15';
Statement processed.
```

```
SELECT SID, SERIAL#, STATUS, SERVER
   FROM V$SESSION
   WHERE USERNAME = 'JWARD';

SID    SERIAL#   STATUS     SERVER
-----  --------  ---------  ---------
    7        15  KILLED     PSEUDO
   12        63  INACTIVE   DEDICATED
2 rows selected.
```

6

制御ファイルの管理

この章では、データベースの制御ファイルを作成し、メンテナンスする方法について説明します。この章の内容は、次のとおりです。

- 制御ファイルの概要
- 制御ファイルのガイドライン
- 制御ファイルの作成
- 制御ファイル作成後のトラブルシューティング
- 制御ファイルのバックアップ
- 現行のコピーを使用した制御ファイルのリカバリ
- 制御ファイルの削除
- 制御ファイル情報の表示

関連項目： Oracle データベース・サーバーによって作成および管理される制御ファイルの作成方法は、第 3 章「Oracle Managed Files の使用」を参照してください。

制御ファイルの概要

すべての Oracle データベースは、**制御ファイル**を持っています。制御ファイルはデータベースの物理構造を記録した小さなバイナリ・ファイルであり、次の情報が格納されています。

- データベース名
- 対応するデータ・ファイルとオンライン REDO ログ・ファイルの名前と位置
- データベース作成のタイムスタンプ
- 現行のログ順序番号
- チェックポイント情報

制御ファイルは、データベースがオープンしているときに必ず Oracle データベースが書き込めるように、使用可能にしておく必要があります。制御ファイルがないと、データベースがマウントできず、リカバリが困難になります。

Oracle データベースの制御ファイルはデータベースとともに作成されます。デフォルトでは、データベースの作成時に、制御ファイルのコピーが少なくとも 1 つ作成されます。デフォルトで複数のコピーが作成されるオペレーティング・システムもあります。データベース作成時に、制御ファイルのコピーを 2 つ以上作成することをお薦めします。その後も、制御ファイルを失ったり、制御ファイル内の設定を変更する場合には、制御ファイルを作成する必要があります。

制御ファイルのガイドライン

ここでは、データベースの制御ファイルを管理するためのガイドラインについて説明します。この項の内容は、次のとおりです。

- 制御ファイルのファイル名の指定
- 異なるディスク上での制御ファイルの多重化
- 制御ファイルの適切な配置
- 制御ファイルのバックアップ
- 制御ファイルのサイズ管理

制御ファイルのファイル名の指定

データベースの初期化パラメータ・ファイルの CONTROL_FILES 初期化パラメータを使用して、制御ファイルの名前を指定します（6-5 ページの「初期制御ファイルの作成」を参照してください）。インスタンス起動処理によって、指定されたすべてのファイルが認識され、オープンされます。データベースの稼働中、インスタンスは指定されたすべての制御ファイルに情報を書き込み、メンテナンスします。

データベースを作成する前に CONTROL_FILES に対してファイルを指定しておらず、Oracle Managed Files 機能を使用していない場合は、デフォルトのファイル名で制御ファイルが作成されます。デフォルトのファイル名はオペレーティング・システムによって異なります。

異なるディスク上での制御ファイルの多重化

Oracle データベースには少なくとも 2 つの制御ファイルを作成し、各ファイルを異なるディスクに配置するようにします。ディスク障害によって制御ファイルが破損した場合は、対応するインスタンスを必ず停止します。ディスク・ドライブを修復後、他のディスク上にある制御ファイルの正常なコピーを使用して破損した制御ファイルをリストアすると、インスタンスを再起動できます。この場合は、メディア・リカバリは不要です。

多重制御ファイルは、次のように動作します。

- Oracle は、データベースの初期化パラメータ・ファイルの初期化パラメータ CONTROL_FILES にリストされているすべてのファイル名に対して、情報を書き込みます。
- データベースの稼働中に Oracle データベースによって読み込まれるのは、CONTROL_FILES パラメータにリストされている最初のファイルのみです。
- データベースの稼働中に制御ファイルのいずれかが使用できなくなった場合、インスタンスは動作不能になり、異常終了します。

注意： データベースには最低 2 つ以上の制御ファイルを作成し、それらを異なるディスク上に配置することをお薦めします。

制御ファイルの適切な配置

前述のように、制御ファイルのコピーはそれぞれ異なるディスク・ドライブに格納してください。一例として、オンライン REDO ログを多重化している場合、オンライン REDO ログ・グループのメンバーが格納されているすべてのディスク・ドライブに制御ファイルのコピーを格納するという方法があります。このようにファイルを配置することによって、単一のディスク障害のために制御ファイルとオンライン REDO ログ・グループがすべて失われる危険が少なくなります。

制御ファイルのバックアップ

制御ファイルのバックアップは非常に重要です。初期設定時およびデータベースの物理構造を変更した後は、必ずバックアップを作成してください。たとえば、次のような構造上の変更を行った場合は、バックアップを作成する必要があります。

- データ・ファイルの追加、削除または名前変更
- 表領域の追加または削除、表領域の読取り / 書込み状態の変更
- REDO ログ・ファイルまたは REDO ログ・グループの追加と削除

制御ファイルのバックアップ方法については、6-11 ページの「制御ファイルのバックアップ」を参照してください。

制御ファイルのサイズ管理

制御ファイルのサイズを決定する主な要因は、対応するデータベースを作成した `CREATE DATABASE` 文の `MAXDATAFILES`、`MAXLOGFILES`、`MAXLOGMEMBERS`、`MAXLOGHISTORY` および `MAXINSTANCES` パラメータに設定された値です。これらのパラメータの値を大きくすると、対応するデータベースの制御ファイルのサイズも大きくなります。

関連項目：

- 制御ファイルの最大サイズの詳細は、使用しているオペレーティング・システム固有の Oracle マニュアルを参照してください。
- `CREATE DATABASE` 文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

制御ファイルの作成

ここでは、制御ファイルを作成する方法について説明します。この項の内容は、次のとおりです。

- 初期制御ファイルの作成
- 制御ファイルの追加コピーの作成、名前変更および再配置
- 新しい制御ファイルの作成

初期制御ファイルの作成

Oracle データベースの初期制御ファイルは、CREATE DATABASE 文を発行したときに作成されます。制御ファイルの名前は、データベースの作成時に使用される初期化パラメータ・ファイルの CONTROL_FILES パラメータで指定します。CONTROL_FILES で指定するファイル名はパスを含めて完全に指定する必要があり、オペレーティング・システムによって異なります。CONTROL_FILES 初期化パラメータの例を次に示します。

```
CONTROL_FILES = (/u01/oracle/prod/control01.ctl,
                 /u02/oracle/prod/control02.ctl,
                 /u03/oracle/prod/control03.ctl)
```

指定した名前のファイルがデータベース作成時にすでに存在する場合は、CREATE DATABASE 文で CONTROLFILE REUSE 句を指定してください。そうしないとエラーが発生します。なお、古い制御ファイルのサイズと新しい制御ファイルの SIZE パラメータが異なる場合には、REUSE オプションは使用できません。

制御ファイルのサイズは、Oracle の一部のリリース間で異なることがあります。また、制御ファイル内に指定されたファイルの数が変わると、制御ファイルのサイズも変わります。制御ファイルのサイズには、MAXLOGFILES、MAXLOGMEMBERS、MAXLOGHISTORY、MAXDATAFILES、MAXINSTANCES などの構成パラメータが影響します。

CONTROL_FILES 初期化パラメータの値を後で変更して、制御ファイルを追加したり、既存の制御ファイルの名前や位置を変更できます。

> **関連項目：** 制御ファイルの指定方法の詳細は、使用しているオペレーティング・システム固有の Oracle マニュアルを参照してください。

制御ファイルの追加コピーの作成、名前変更および再配置

制御ファイルのコピーを追加作成するには、既存の制御ファイルを新しい位置にコピーし、そのファイル名を制御ファイルのリストに追加します。同様に、制御ファイルの名前を変更するには、既存の制御ファイルを新しい名前や位置にコピーし、制御ファイル・リストのファイル名を変更します。どちらの場合も、作業中に制御ファイルが変更されないように、制御ファイルをコピーする前にインスタンスを停止してください。

現行の制御ファイルの追加コピーを多重化または移動する手順

1. データベースを停止します。
2. オペレーティング・システムのコマンドを使用して、既存の制御ファイルを異なる位置にコピーします。
3. データベースの初期化パラメータ・ファイルの CONTROL_FILES パラメータを編集して、新しい制御ファイルの名前を追加するか、または既存の制御ファイル名を変更します。
4. データベースを再起動します。

新しい制御ファイルの作成

ここでは、新しい制御ファイルを作成する場合とその方法について説明します。

新しい制御ファイルを作成する場合

次の状況の場合に、新しい制御ファイルを作成する必要があります。

- データベースの制御ファイルがすべて破損し、制御ファイルのバックアップがない場合。
- CREATE DATABASE 文で最初に指定した永続データベース・パラメータ設定のいずれかを変更する場合。これらの設定には、データベースの名前とパラメータ MAXLOGFILES、MAXLOGMEMBERS、MAXLOGHISTORY、MAXDATAFILES および MAXINSTANCES が含まれます。

  たとえば、分散環境でデータベースの名前が別のデータベースの名前と競合する場合、その名前を変更することがあります。また、元の設定が小さすぎる場合に、MAXLOGFILES の値を変更する場合があります。

CREATE CONTROLFILE 文

CREATE CONTROLFILE 文を使用して、データベースの新しい制御ファイルを作成できます。次の文は、prod データベース（以前は別のデータベース名を使用していたデータベース）用に新しい制御ファイルを作成します。

```
CREATE CONTROLFILE
   SET DATABASE prod
   LOGFILE GROUP 1 ('/u01/oracle/prod/redo01_01.log',
                    '/u01/oracle/prod/redo01_02.log'),
           GROUP 2 ('/u01/oracle/prod/redo02_01.log',
                    '/u01/oracle/prod/redo02_02.log'),
           GROUP 3 ('/u01/oracle/prod/redo03_01.log',
                    '/u01/oracle/prod/redo03_02.log')
   NORESETLOGS
   DATAFILE '/u01/oracle/prod/system01.dbf' SIZE 3M,
            '/u01/oracle/prod/rbs01.dbs' SIZE 5M,
            '/u01/oracle/prod/users01.dbs' SIZE 5M,
            '/u01/oracle/prod/temp01.dbs' SIZE 5M
   MAXLOGFILES 50
   MAXLOGMEMBERS 3
   MAXLOGHISTORY 400
   MAXDATAFILES 200
   MAXINSTANCES 6
   ARCHIVELOG;
```

注意：

- CREATE CONTROLFILE 文では、指定したデータ・ファイルとオンライン REDO ログ・ファイルを破損する可能性があります。ファイル名を省略すると、そのファイルのデータが失われたり、データベース全体にアクセスできなくなる場合があります。この文を使用するときには十分に注意し、必ず「新しい制御ファイルの作成手順」の手順に従ってください。
- 新しい制御ファイルを作成する前にデータベースの FORCE LOGGING を使用可能にしていて、この設定を引き続き有効にする場合は、CREATE CONTROLFILE 文で FORCE LOGGING 句を指定する必要があります。2-29 ページの「FORCE LOGGING モードの指定」を参照してください。

関連項目： CREATE CONTROLFILE 文の詳細な構文は、『Oracle9*i* SQL リファレンス』を参照してください。

新しい制御ファイルの作成手順

新しい制御ファイルを作成するには、次の手順を実行します。

1. データベースのデータ・ファイルとオンライン REDO ログ・ファイルすべてのリストを作成します。

   6-11 ページの「制御ファイルのバックアップ」に記載されている制御ファイルのバックアップの推奨事項に従っている場合は、現在のデータベース構造を反映するデータ・ファイルとオンライン REDO ログ・ファイルのリストがすでに作成されています。しかし、そのようなリストがない場合は、次の文を実行することでリストが生成されます。

   ```
   SELECT MEMBER FROM V$LOGFILE;
   SELECT NAME FROM V$DATAFILE;
   SELECT VALUE FROM V$PARAMETER WHERE NAME = 'CONTROL_FILES';
   ```

   このようなリストが手元になく、制御ファイルが破損してデータベースがオープンできない場合は、データベースを構成するデータ・ファイルとオンライン REDO ログ・ファイルをすべて特定してください。新しい制御ファイルを作成すると、手順 5 で指定したファイル以外はリカバリできなくなります。さらに、SYSTEM 表領域を構成するファイルを 1 つでも省略すると、データベースをリカバリできないことがあります。

2. データベースを停止します。

   データベースがオープンしている場合は、できるかぎり通常モードでデータベースを停止します。IMMEDIATE オプションまたは ABORT オプションは、他に方法がない場合にのみ使用してください。

3. データベースのデータ・ファイルとオンライン REDO ログ・ファイルすべてのバックアップを作成します。

4. 新しいインスタンスを起動します。ただし、データベースのマウントとオープンは行いません。

   ```
   STARTUP NOMOUNT
   ```

5. CREATE CONTROLFILE 文を使用して、データベースの新しい制御ファイルを作成します。

   制御ファイルに加えて、オンライン REDO ログ・グループも失ってしまった場合は、新しい制御ファイルの作成時に RESETLOGS オプションを選択します。この場合には、失った REDO ログをリカバリする必要があります（手順 8）。データベースの名前を変更した場合も、必ず RESETLOGS オプションも指定してください。それ以外の場合には、NORESETLOGS オプションを選択してください。

6. 新しい制御ファイルのバックアップをオフラインの記憶デバイスに格納します。バックアップ作成の手順については、6-11 ページの「制御ファイルのバックアップ」を参照してください。

7. データベースの CONTROL_FILES 初期化パラメータを編集し、手順 5 で作成したすべての制御ファイルがデータベースの新しい一部となるように指定します。ただし、バックアップ制御ファイルは含めません。データベースの名前を変更する場合は、DB_NAME パラメータを編集して新しい名前を指定します。

8. 必要に応じて、データベースをリカバリします。データベースをリカバリしない場合は、手順 9 にスキップしてください。

   リカバリの一部として制御ファイルを作成している場合は、データベースをリカバリしてください。NORESETLOGS オプションを使用して新しい制御ファイルを作成した場合（手順 5）は、クローズ状態の完全なデータベース・リカバリ操作によってデータベースをリカバリできます。

   RESETLOGS オプションを使用して新しい制御ファイルを作成した場合は、USING BACKUP CONTROL FILE を指定する必要があります。オンラインまたはアーカイブ REDO ログあるいはデータ・ファイルが失われた場合は、これらのファイルのリカバリ手順に従ってください。

9. 次のいずれかの方法で、データベースをオープンします。

   - リカバリを実行しなかった場合、または手順 8 でクローズ状態の完全なデータベース・リカバリを実行した場合は、通常の手順でデータベースをオープンします。

     ```
     ALTER DATABASE OPEN;
     ```

   - 制御ファイルの作成時に RESETLOGS を指定した場合は、ALTER DATABASE 文で RESETLOGS を指定します。

     ```
     ALTER DATABASE OPEN RESETLOGS;
     ```

これでデータベースはオープンされ、使用可能になります。

関連項目： 次のことに関する詳細は、『Oracle9*i* 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。

- データベース・ファイルのリスト表示
- データベースのすべてのデータ・ファイルとオンライン REDO ログ・ファイルのバックアップ作成
- オンラインまたはアーカイブ REDO ログ・ファイルのリカバリ
- クローズ状態のデータベースのリカバリ

制御ファイル作成後のトラブルシューティング

CREATE CONTROLFILE 文を実行した後で、一般的なエラーが発生する場合があります。ここでは、制御ファイルの使用に関連する最も一般的なエラーについて説明します。この項の内容は、次のとおりです。

- 欠落したファイルや余分なファイルのチェック
- CREATE CONTROLFILE でのエラー処理

欠落したファイルや余分なファイルのチェック

新しい制御ファイルを作成し、それを使用してデータベースをオープンした後、アラート・ファイルをチェックして、データ・ディクショナリと制御ファイルの間で不整合（データ・ディクショナリにはデータ・ファイルがあるが、制御ファイルには含まれていないなど）が検出されていないかを確認してください。

データ・ディクショナリ内にはデータ・ファイルが存在していて、新しい制御ファイルには含まれていない場合、Oracle は制御ファイル内に MISSING*nnnn*（*nnnn* は 10 進数のファイル番号）という名前のプレースホルダ・エントリを作成します。制御ファイル内の MISSING*nnnn* には、オフラインであること、およびメディア・リカバリを必要とすることを示すフラグが設定されます。

MISSING*nnnn* に対応する実際のデータ・ファイルにアクセス可能にするには、データ・ファイルを指すように MISSING*nnnn* の名前を変更します。ただし、この方法を使用できるのは、実際のデータ・ファイルが読取り専用または通常のオフラインであった場合のみです。MISSING*nnnn* が、読取り専用と通常のオフラインのどちらでもないデータ・ファイルに対応している場合は、名前変更操作を行ってもデータ・ファイルはアクセス可能になりません。これは、RESETLOGS の結果によって使用できなくなったデータ・ファイルには、メディア・リカバリが必要なためです。この場合は、データ・ファイルを含む表領域を削除する必要があります。

反対に、制御ファイルに指定されているデータ・ファイルがデータ・ディクショナリに存在しない場合、Oracle は新しい制御ファイルからそのファイルへの参照を削除します。どちらの場合でも、検出された状態を通知するメッセージが alert.log ファイルに書き込まれます。

CREATE CONTROLFILE でのエラー処理

新しい制御ファイルの作成後、データベースをマウントしてオープンしようとしたときに、エラー（通常、ORA-01173、ORA-01176、ORA-01177、ORA-01215、ORA-01216 のいずれか）が出力された場合、最も可能性の高い原因は、CREATE CONTROLFILE 文でファイルを省略したか、またはリストに示されていないファイルを指定したかのいずれかです。この場合は、6-8 ページの手順 3 で作成したバックアップのファイルをリストアし、正しいファイル名を使用して手順 4 以降を再実行してください。

制御ファイルのバックアップ

制御ファイルをバックアップするには、ALTER DATABASE BACKUP CONTROLFILE 文を使用します。次の 2 つの方法があります。

1. 次の文を使用して、制御ファイルをバイナリ・ファイル（既存の制御ファイルの複製）にバックアップを作成します。

   ```
   ALTER DATABASE BACKUP CONTROLFILE TO '/oracle/backup/control.bkp';
   ```

2. 後で制御ファイルの再作成に使用できる SQL 文を生成します。

   ```
   ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
   ```

   このコマンドは、データベースのトレース・ファイルに SQL スクリプトを書き込みます。書き込まれた SQL スクリプトは、制御ファイルを再作成するために、トレース・ファイルから取得して編集できます。

   関連項目： 制御ファイルのバックアップの詳細は、バックアップ計画に応じて次のいずれかのマニュアルを参照してください。

   - 『Oracle9*i* ユーザー管理バックアップおよびリカバリ・ガイド』
   - 『Oracle9*i* Recovery Manager ユーザーズ・ガイド』

現行のコピーを使用した制御ファイルのリカバリ

ここでは、現行のバックアップまたは多重化コピーから制御ファイルをリカバリする方法について説明します。

制御ファイルのコピーを使用した制御ファイル破損からのリカバリ

この手順では、CONTROL_FILES パラメータで指定されている制御ファイルの 1 つが破損した場合を想定しています。制御ファイルのディレクトリにはまだアクセス可能で、制御ファイルの多重化コピーがあるとします。

1. インスタンスを停止してから、オペレーティング・システム・コマンドを使用して、破損した制御ファイルに正常なコピーを上書きします。

   ```
   % cp /u01/oracle/prod/control03.ctl  /u01/oracle/prod/control02.ctl
   ```

2. SQL*Plus を起動してデータベースをオープンします。

   ```
   SQL> STARTUP
   ```

制御ファイルのコピーを使用した永続的なメディア障害からのリカバリ

この手順では、永続的なメディア障害のために、CONTROL_FILES パラメータで指定されている制御ファイルの 1 つにアクセスできない場合を想定しています。制御ファイルの多重化コピーがあるとします。

1. インスタンスを停止してから、オペレーティング・システム・コマンドを使用して、制御ファイルの現行のコピーをアクセス可能な新しい位置にコピーします。

```
% cp /u01/oracle/prod/control01.ctl  /u04/oracle/prod/control03.ctl
```

2. 初期化パラメータ・ファイルの CONTROL_FILES パラメータを編集し、破損したファイルの位置を新しい位置に置き換えます。

```
CONTROL_FILES = (/u01/oracle/prod/control01.ctl,
                 /u02/oracle/prod/control02.ctl,
                 /u04/oracle/prod/control03.ctl)
```

3. SQL*Plus を起動してデータベースをオープンします。

```
SQL> STARTUP
```

多重制御ファイルがあり、最短時間でデータベースをリカバリする必要がある場合は、CONTROL_FILES 初期化パラメータを編集して破損した制御ファイルを削除し、即時にデータベースを再起動します。次に、破損した制御ファイルを再作成し、CONTROL_FILES 初期化パラメータにリカバリした制御ファイルを設定してから、データベースを停止し、再起動します。

制御ファイルの削除

制御ファイルはデータベースから削除できます。たとえば、制御ファイルの位置が不適切な場合は、その制御ファイルを削除できます。ただし、データベースには常に少なくとも 2 つの制御ファイルが存在する必要があります。

1. データベースを停止します。
2. データベースの初期化パラメータ・ファイルの CONTROL_FILES パラメータを編集して、古い制御ファイル名を削除します。
3. データベースを再起動します。

注意： この操作では、不要な制御ファイルをディスクから物理的に削除することはできません。データベースから制御ファイルを削除した後、オペレーティング・システムのコマンドを使用して不要なファイルを削除してください。

制御ファイル情報の表示

次のビューには、制御ファイルに関する情報が表示されます。

ビュー	説明
V$DATABASE	制御ファイル内のデータベース情報が表示されます。
V$CONTROLFILE	制御ファイル名が一覧表示されます。
V$CONTROLFILE_RECORD_SECTION	制御ファイルのレコード・セクションに関する情報が表示されます。
V$PARAMETER	CONTROL_FILES 初期化パラメータで指定されている制御ファイルの名前が表示されます。

この例では、制御ファイル名が一覧表示されます。

```
SQL> SELECT NAME FROM V$CONTROLFILE;

NAME
------------------------------------
/u01/oracle/prod/control01.ctl
/u02/oracle/prod/control02.ctl
/u03/oracle/prod/control03.ctl
```

7

オンライン REDO ログの管理

この章では、オンライン REDO ログを管理する方法について説明します。この章の内容は、次のとおりです。

- オンライン REDO ログの概要
- オンライン REDO ログの計画
- オンライン REDO ログ・グループおよびメンバーの作成
- オンライン REDO ログ・メンバーの再配置および名前変更
- オンライン REDO ログ・グループおよびメンバーの削除
- ログ・スイッチの強制
- REDO ログ・ファイル内のブロックの検証
- オンライン REDO ログ・ファイルの初期化
- オンライン REDO ログ情報の表示

関連項目：

- Oracle データベースによって作成および管理されるオンライン REDO ログ・ファイルの作成の詳細は、第 3 章「Oracle Managed Files の使用」を参照してください。
- Oracle Real Application Clusters の使用中にインスタンスのオンライン REDO ログを管理する方法の詳細は、『Oracle9*i* Real Application Clusters 管理』を参照してください。
- チェックポイントと REDO ログがインスタンス・リカバリに及ぼす影響については、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。

オンライン REDO ログの概要

オンライン REDO ログは、リカバリ操作にとって最も重要な構造です。これは、データベースに加えられたすべての変更を発生時に格納する、2 つ以上の事前割当てファイルから構成されます。Oracle データベースの各インスタンスには、インスタンスの障害時にデータベースを保護するためのオンライン REDO ログが 1 つずつ対応付けられています。

REDO スレッド

各データベース・インスタンスは、専用のオンライン **REDO ログ・グループ**を持ちます。これらのオンライン REDO ログ・グループは、多重化されているかどうかに関係なく、インスタンスのオンライン REDO の**スレッド**と呼ばれます。標準的な構成では、Oracle データベースにアクセスするデータベース・インスタンスは 1 つのみであるため、スレッドは 1 つしか存在しません。ただし、Oracle Real Application Clusters の実行時には、複数のインスタンスが単一のデータベースに同時にアクセスし、各インスタンスが専用スレッドを持ちます。

この章では、Oracle9*i* Real Application Clusters を使用していない場合に、オンライン REDO ログを構成し、管理する方法について説明します。以降、文のすべての説明と例では、スレッド番号を 1 と想定します。

オンライン REDO ログの内容

オンライン REDO ログ・ファイルには、**REDO レコード**が書き込まれます。REDO レコードは **REDO エントリ**とも呼ばれ、**変更ベクトル**（データベース内の単一ブロックに加えられた変更の記述）のグループからなっています。たとえば、従業員表の給与値を変更する場合は、その表のデータ・セグメント・ブロック、ロールバック・セグメント・データ・ブロックおよびロールバック・セグメントのトランザクション表の変更内容を記述する変更ベクトルを含む REDO レコードが生成されます。

REDO エントリには、ロールバック・セグメントなど、データベースに対するすべての変更の再構築に使用できるデータが記録されます。したがって、オンライン REDO ログによってロールバック・データも保護されます。REDO データベースを使用してデータベースをリカバリさせるときには、Oracle は REDO レコード内の変更ベクトルを読み込んで変更内容を関連ブロックに適用します。

REDO レコードは、循環方式でシステム・グローバル領域（SGA）の REDO ログ・バッファに入れられ（次の項の「Oracle によるオンライン REDO ログの書込み」を参照）、Oracle バックグラウンド・プロセスであるログ・ライター（LGWR）によってオンライン REDO ログ・ファイルの 1 つに書き込まれます。トランザクションがコミットされると、LGWR によってそのトランザクションの REDO レコードが SGA の REDO ログ・バッファからオンライン REDO ログ・ファイルに書き込まれ、コミットされた各トランザクションの REDO レコードを識別するために**システム変更番号**（SCN）が割り当てられます。特定のトランザクションに対応付けられたすべての REDO ログ・レコードがディスク上のオンライン・ログに安全に書き込まれた場合にのみ、ユーザー・プロセスはトランザクションがコミットされたことを示す通知を受け取ります。

また、REDO レコードは、対応するトランザクションがコミットされる前にオンライン REDO ログ・ファイルに書き込むこともできます。REDO ログ・バッファがいっぱいになるか、別のトランザクションがコミットされると、一部の REDO レコードがコミットされていない可能性があっても、LGWR は REDO ログ・バッファ内のすべての REDO ログ・エントリをオンライン REDO ログ・ファイルにフラッシュします。必要に応じて、Oracle はこれらの変更をロールバックできます。

Oracle によるオンライン REDO ログの書込み

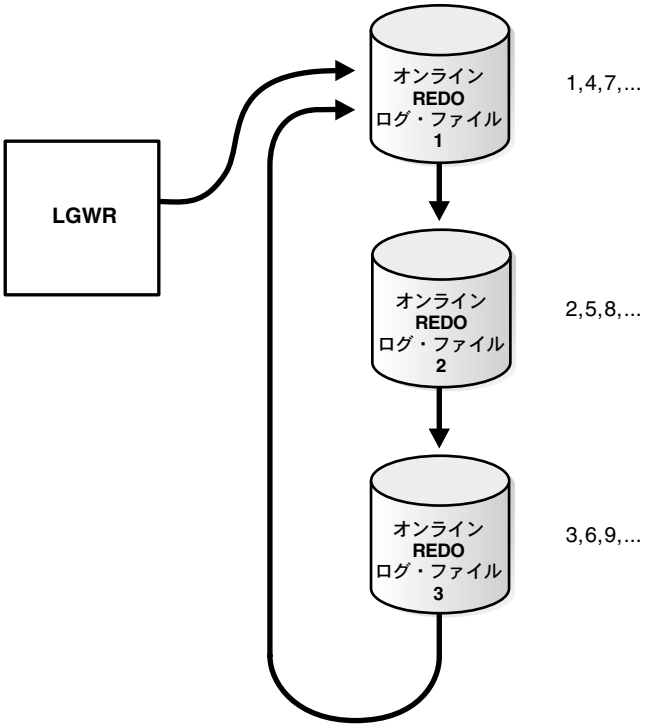
データベースのオンライン REDO ログは、2 つ以上のオンライン REDO ログ・ファイルから構成されます。Oracle は、一方のファイルのアーカイブ中にも（ARCHIVELOG モード時）、他方のファイルが常に書込み可能であることを保証するために、最低 2 つのファイルを必要とします。

LGWR は、オンライン REDO ログ・ファイルに循環方式で書き込みます。つまり、現行のオンライン REDO ログ・ファイルがいっぱいになると、LGWR は次に使用可能なオンライン REDO ログ・ファイルへの書込みを開始します。使用可能な最後のオンライン REDO ログ・ファイルがいっぱいになると、LGWR は最初のオンライン REDO ログ・ファイルに戻って書込みを行い、再び循環を開始します。図 7-1 は、オンライン REDO ログ・ファイルへの循環方式の書込みを示しています。各ラインの隣の番号は、LGWR が各オンライン REDO ログ・ファイルに書き込む順序を示しています。

いっぱいになったオンライン REDO ログ・ファイルは、アーカイブが使用可能になっているかどうかに応じて、LGWR で再利用できます。

- アーカイブが使用禁止になっている場合（NOARCHIVELOG モード）、いっぱいになったオンライン REDO ログ・ファイルは、そこに記録された変更がデータ・ファイルに書き込まれた後に使用可能になります。
- アーカイブが使用可能になっている場合（ARCHIVELOG モード）、いっぱいになったオンライン REDO ログ・ファイルは、そこに記録された変更がデータ・ファイルに書き込まれ、かつ、そのファイルがアーカイブされた後に、LGWR で使用可能になります。

図 7-1 LGWR によるオンライン REDO ログ・ファイルの循環使用



アクティブ（カレント）および非アクティブなオンライン REDO ログ・ファイル

Oracle は、どの時点でもオンライン REDO ログ・ファイルを 1 つのみ使用して、REDO ログ・バッファから書き込まれた REDO レコードを格納します。LGWR が書込み中のオンライン REDO ログ・ファイルを**現行のオンライン REDO ログ**・ファイルと呼びます。

インスタンス・リカバリに必要なオンライン REDO ログ・ファイルを**アクティブ**なオンライン REDO ログ・ファイルと呼びます。インスタンス・リカバリに必要でないオンライン REDO ログ・ファイルを**非アクティブ**と呼びます。

アーカイブを使用可能にしている場合は（`ARCHIVELOG` モード）、ARC*n* によって内容がアーカイブされるまで、Oracle はアクティブなオンライン・ログ・ファイルの再利用または上書きができません。アーカイブが使用禁止になっている場合は（`NOARCHIVELOG` モード）、最後のオンライン REDO ログ・ファイルがいっぱいになると、使用可能な最初のアクティブ・ファイルが上書きされて書込みが継続します。

ログ・スイッチとログ順序番号

ログ・スイッチは、Oracle があるオンライン REDO ログ・ファイルへの書込みを終了して他のファイルへの書込みを開始するポイントです。現行のオンライン REDO ログ・ファイルが完全にいっぱいになり、引き続き次のオンライン REDO ログ・ファイルへの書込みが必要になると、通常はログ・スイッチが発生します。ただし、現行のオンライン REDO ログ・ファイルが完全にいっぱいになっているかどうかに関係なく、時間ベースでログ・スイッチの発生を指定することもできます。ログ・スイッチは、手動で強制的に発生させることもできます。

Oracle は、ログ・スイッチが発生して LGWR が書込みを開始するたびに、各オンライン REDO ログ・ファイルに新しい**ログ順序番号**を割り当てます。Oracle がオンライン REDO ログ・ファイルをアーカイブしても、そのファイルのログ順序番号は変わりません。一巡して再び使用可能になったオンライン REDO ログ・ファイルには、次に使用可能なログ順序番号が割り当てられます。

各オンライン REDO ログ・ファイルまたはアーカイブ REDO ログ・ファイルは、そのログ順序番号で一意に識別されます。クラッシュ、インスタンスまたはメディア・リカバリ中に、Oracle は必要なアーカイブおよびオンライン REDO ログ・ファイルのログ順序番号を使用して、REDO ログ・ファイルを昇順に正しく適用します。

オンライン REDO ログの計画

ここでは、データベース・インスタンスのオンライン REDO ログを構成するときに考慮すべきガイドラインについて説明します。この項の内容は、次のとおりです。

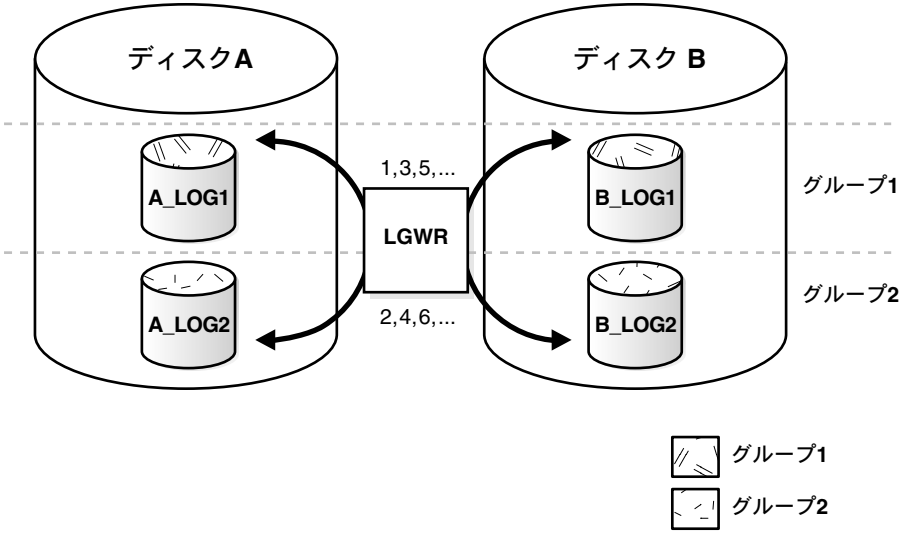
- オンライン REDO ログ・ファイルの多重化
- 異なるディスクへのオンライン REDO ログ・メンバーの配置
- オンライン REDO ログ・メンバーのサイズの設定
- 適切なオンライン REDO ログ・ファイル数の選択
- アーカイブ・タイムラグの制御

オンライン REDO ログ・ファイルの多重化

Oracle は、オンライン REDO ログ・ファイルが破損した場合に備えて、インスタンスのオンライン REDO ログ・ファイルを**多重化**する機能を持っています。オンライン REDO ログ・ファイルを多重化すると、LGWR は同じ REDO ログ情報を複数のオンライン REDO ログ・ファイルに書き込むため、REDO ログの単一の障害箇所は問題になりません。

注意： REDO ログ・ファイルは多重化することをお薦めします。これは、リカバリが必要になったときにログ・ファイルのデータが失われていると、致命的な事態を招くおそれがあるためです。

図 7-2　多重オンライン REDO ログ・ファイル



対応するオンライン REDO ログ・ファイルを**グループ**と呼びます。グループ内の各オンライン REDO ログ・ファイルを**メンバー**と呼びます。図 7-2 では、A_LOG1 と B_LOG1 はどちらもグループ 1 のメンバーで、A_LOG2 と B_LOG2 はどちらもグループ 2 のメンバーです。グループ内の各メンバーのサイズは、完全に一致させてください。

LGWR から同一のログ順序番号が割り当てられることが示すように、グループの各メンバーは同時にアクティブになり、LGWR によって同時に書き込まれます。図 7-2 では、LGWR は最初に A_LOG1 と B_LOG1 に同時に書き込み、次に A_LOG2 と B_LOG2 に同時に書き込みます。LGWR が異なるグループのメンバー（A_LOG1 と B_LOG2 など）に同時に書き込むことはありません。

オンライン REDO ログの障害への対処

LGWR がグループのメンバーに書き込めない場合、Oracle はそのメンバーに `INVALID` を示すマークを付け、LGWR トレース・ファイルとデータベースのアラート・ファイルに、アクセス不可能ファイルの問題を示すエラー・メッセージを書き込みます。特定のオンライン REDO ログ・メンバーが使用できない場合、LGWR の動作はその原因に応じて異なります。

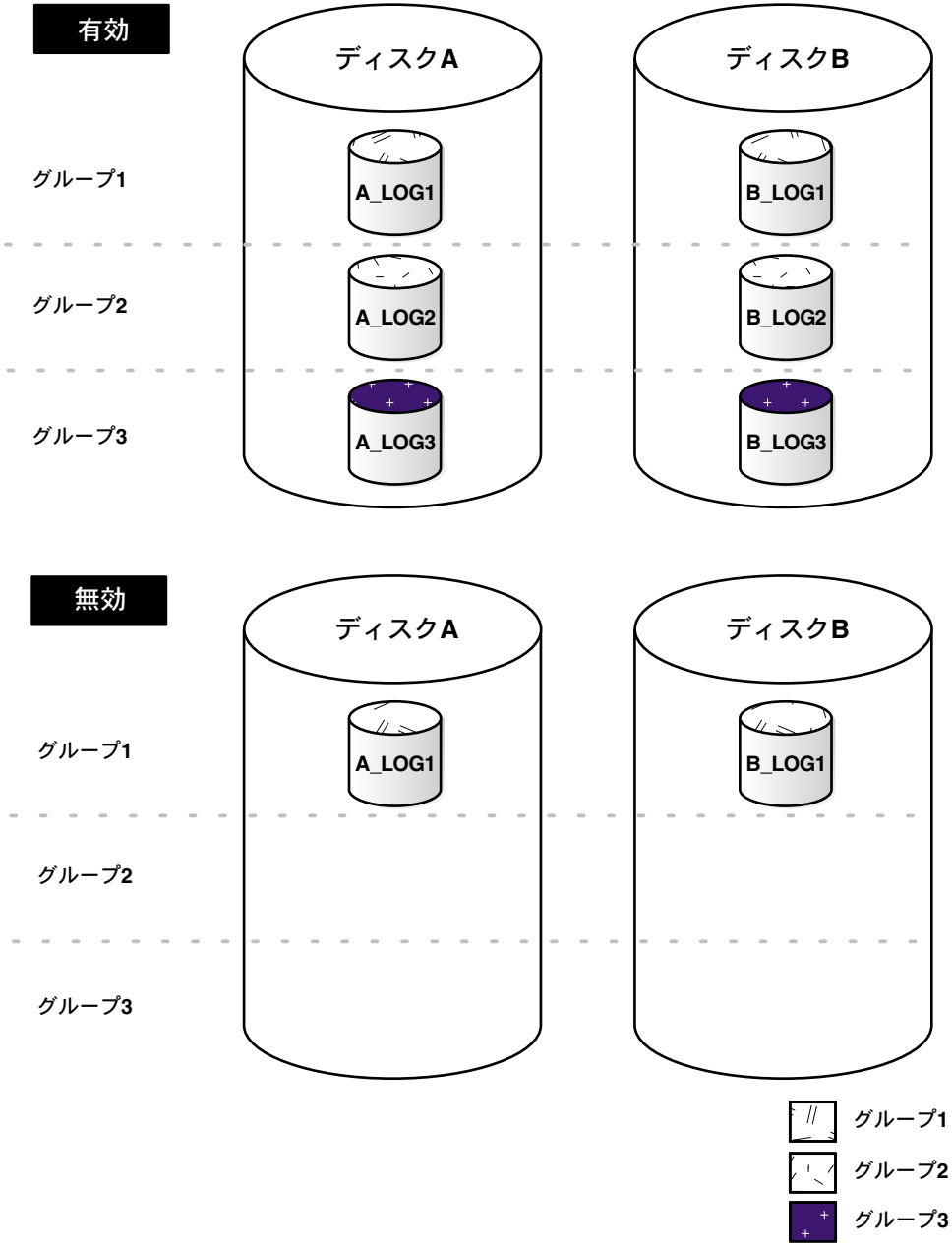
状況	対処
LGWR がグループ内の最低 1 つのメンバーに正常に書き込める場合	通常どおり書込みが行われます。LGWR はグループのうち使用可能なメンバーにのみ書き込み、使用不可のメンバーは無視します。
グループをアーカイブする必要があるため、LGWR がログ・スイッチの発生時に次のグループにアクセスできない場合	データベース操作は、グループが使用可能になるか、グループがアーカイブされるまで一時的に停止します。
メディア障害のため、ログ・スイッチの発生時に LGWR が次のグループのどのメンバーにもアクセスできない場合	Oracle はエラーを返し、データベース・インスタンスは停止します。この場合は、データベース上でオンライン REDO ログ・ファイルの損失からのメディア・リカバリを実行する必要があります。 データベースのチェックポイントが実行済みで、損失した REDO ログがそのチェックポイント以前のものの場合、その REDO ログに記録されたデータは Oracle によってデータ・ファイルに保存されているので、メディア・リカバリの必要はありません。単純に、アクセスできない REDO ログ・グループを削除してください。不良ログがアーカイブされていない場合は、`ALTER DATABASE CLEAR UNARCHIVED LOG` を使用してアーカイブを使用禁止にしてから、ログを削除してください。
LGWR が書込み中に、グループのすべてのメンバーに突然アクセスできなくなった場合	Oracle はエラーを返し、データベース・インスタンスは即時に停止します。この場合は、メディア・リカバリを実行する必要があります。ログのドライブを不注意からオフにした場合など、ログを含むメディアが実際には失われていなければ、メディア・リカバリは不要です。この場合は、ドライブをオンに戻して、Oracle にインスタンス・リカバリを実行します。

有効な構成と無効な構成

オンライン REDO ログの単一の障害箇所からデータベースを保護するには、多重オンライン REDO ログ・ファイルを対称型にしておくのが理想的です。つまり、オンライン REDO ログのどのグループも、メンバーが同数になるようにします。ただし、Oracle では、必ずしも多重オンライン REDO ログ・ファイルを対称型にする必要はありません。たとえば、あるグループのメンバーは 1 つ、他のグループのメンバーは 2 つでもかまいません。この構成は、一部のオンライン REDO ログ・メンバーには一時的に影響しても、他のメンバーには影響しないディスク障害からデータベースを保護します。

インスタンスのオンライン REDO ログに関する唯一の要件は、最低 2 つのグループを持つことです。図 7-3 は、多重オンライン REDO ログに有効な構成と無効な構成を示しています。2 番目の構成は、グループが 1 つしかないため無効です。

図 7-3 多重オンライン REDO ログ・ファイルの有効な構成と無効な構成



異なるディスクへのオンライン REDO ログ・メンバーの配置

多重オンライン REDO ログ・ファイルを設定する場合は、グループのメンバーを異なるディスク上に配置します。このようにすると、1 つのディスクで障害が発生しても、LGWR が使用できなくなるのはグループの 1 つのメンバーのみで、それ以外のメンバーには引き続きアクセスできるので、インスタンスは継続して機能します。

REDO ログをアーカイブする場合は、バックグラウンド・プロセス LGWR と ARC*n* 間の競合をなくすために、オンライン REDO ログ・メンバーを複数のディスクに分散します。たとえば、多重化したオンライン REDO ログ・メンバーのグループが 2 つある場合、各メンバーを異なるディスク上に配置し、アーカイブ先を 5 つ目のディスクに設定します。このようにすると、LGWR（メンバーへの書込み）と ARC*n*（メンバーの読込み）間の競合は起こりません。

データ・ブロックや REDO レコードの書込み時の競合を減らすために、データ・ファイルとオンライン REDO ログ・ファイルも異なるディスク上に配置することをお薦めします。

> **関連項目：** オンライン REDO ログがバックアップとリカバリに及ぼす影響の詳細は、『Oracle9*i* バックアップおよびリカバリ概要 』を参照してください。

オンライン REDO ログ・メンバーのサイズの設定

オンライン REDO ログ・ファイルのサイズを設定するときには、REDO ログをアーカイブするかどうかを考慮してください。オンライン REDO ログ・ファイルのサイズは、いっぱいになったグループを 1 つのオフライン記憶メディア（テープやディスクなど）にアーカイブできるとともに、そのメディア上の未使用領域が最小になるように設定します。たとえば、いっぱいになった 1 つのオンライン REDO ログ・グループを 1 本のテープにアーカイブし、そのテープの記憶容量の 49% が未使用のまま残っているとします。この場合には、オンライン REDO ログ・ファイルのサイズを小さくして、テープごとに 2 つずつオンライン REDO ログ・グループがアーカイブされるように構成することをお薦めします。

オンライン REDO ログの多重グループでは、同一グループのメンバーはすべて同じサイズにする必要があります。異なるグループのメンバーは異なるサイズにすることができます。ただし、グループ間でファイル・サイズを変えても特に利点はありません。ログ・スイッチ間にチェックポイントが発生するように設定していない場合は、グループのサイズを同一に設定して、チェックポイントが一定の間隔で発生することを保証してください。

> **関連項目：** 使用しているオペレーティング・システム固有の Oracle マニュアルを参照してください。オンライン REDO ログ・ファイルのデフォルトのサイズは、オペレーティング・システムによって異なります。

適切なオンライン REDO ログ・ファイル数の選択

データベース・インスタンスに対するオンライン REDO ログ・ファイルの適切な数を決定する最良の方法は、様々な構成をテストすることです。最適な構成では、LGWR が REDO ログ情報を書き込むのを妨げない最小の数がグループ数になります。

データベース・インスタンスに必要なグループが 2 つのみの場合もあります。また、LGWR が使用可能な再利用グループが常に存在するようにするため、データベース・インスタンスにグループを追加する必要がある場合もあります。テスト中に、現行のオンライン REDO ログ構成に問題がないかどうかを確認するには、LGWR トレース・ファイルおよびデータベースのアラート・ログの内容を調べるのが最も簡単です。チェックポイントやグループのアーカイブが完了していないために LGWR が待機する必要があるというメッセージが頻繁に示された場合は、グループを追加します。

インスタンスのオンライン REDO ログ構成を設定または変更する前に、オンライン REDO ログ・ファイル数を制限するパラメータを検討してください。次のパラメータは、データベースに追加できるオンライン REDO ログ・ファイルの数を制限します。

- `CREATE DATABASE` 文の `MAXLOGFILES` パラメータは、データベース当たりのオンライン REDO ログ・ファイルの最大グループ数を決定します。グループの値は 1 ～ `MAXLOGFILES` です。この上限値を変更する唯一の方法は、データベースまたは制御ファイルを再作成することです。したがって、データベースを作成する前にこの上限値を十分検討してください。`CREATE DATABASE` 文に `MAXLOGFILES` パラメータが指定されていない場合は、オペレーティング・システム固有のデフォルト値が使用されます。
- `CREATE DATABASE` 文の `MAXLOGMEMBERS` パラメータは、グループに含まれるメンバーの最大値を決定します。この上限値を変更する唯一の方法は、`MAXLOGFILES` の場合と同様、データベースまたは制御ファイルを再作成することです。したがって、データベースを作成する前にこの上限値を十分検討してください。`CREATE DATABASE` 文に `MAXLOGMEMBERS` パラメータが指定されていない場合は、オペレーティング・システムのデフォルト値が使用されます。

関連項目： `MAXLOGFILES` パラメータおよび `MAXLOGMEMBERS` パラメータのデフォルト値と有効な値については、オペレーティング・システム固有の Oracle マニュアルを参照してください。

アーカイブ・タイムラグの制御

すべての有効なオンライン REDO ログ・スレッドについて、そのカレント・ログを時間ベースで強制的に切り替えることができます。プライマリ / スタンバイ構成では、プライマリ・サイトのログをアーカイブしてスタンバイ・データベースに転送することにより、スタンバイ・データベースを変更できます。スタンバイ・データベースに適用される変更には、プライマリ・データベースで発生している変更に対するタイムラグがある場合があります。

プライマリ・データベースでオンライン REDO ログがアーカイブ REDO ログにアーカイブされてから、スタンバイ・データベースに転送されるまで、ある程度時間がかかります。この間、スタンバイ・データベースは変更を待機する必要があるため、このタイムラグが発生します。このタイムラグの制御や制限には、ARCHIVE_LAG_TARGET 初期化パラメータを使用します。このパラメータを設定することにより、許容するタイムラグの長さを時間で制限できます。

ARCHIVE_LAG_TARGET 初期化パラメータの設定

ARCHIVE_LAG_TARGET 初期化パラメータを設定すると、Oracle はインスタンスの現行のオンライン REDO ログを定期的に調べます。次の条件が満たされると、インスタンスはログを切り替えます。

- カレント・ログが *n* 秒前に作成され、カレント・ログのアーカイブ見積り時間が *m* 秒（この時間はカレント・ログで使用されている REDO ブロック数に比例する）の場合に、*n*+*m* が ARCHIVE_LAG_TARGET 初期化パラメータの値を超えている。
- カレント・ログに REDO レコードが含まれている。

Oracle Real Application Clusters 環境では、他のスレッドが遅れている場合、前述の条件を検出したインスタンスによって他のスレッドも切り替えられて、ログがアーカイブされます。これは、Oracle Real Application Clusters で 2 つのノードを持つプライマリ / セカンダリ構成を実行しているときのように、クラスタ内に他のインスタンスよりも稼働率の低いインスタンスがある場合に特に有効です。

初期化パラメータ ARCHIVE_LAG_TARGET は、Data Guard 環境が非データ消失モードで構成されていない場合に、プライマリ側で停止やクラッシュが起こったときに、スタンバイ側で失ってもかまわない REDO 時間（秒）のターゲットを指定します。また、このパラメータは、プライマリ・データベースのカレント・ログが使用される時間の上限（秒数）も指定します。アーカイブ見積り時間も考慮されるため、これはログ・スイッチ時間そのものではありません。

次の初期化パラメータ設定では、ログ・スイッチ間隔を 30 分（標準的な値）に設定します。

```
ARCHIVE_LAG_TARGET = 1800
```

0（ゼロ）を指定すると、時間ベースのログ・スイッチ機能は使用禁止になります。これはデフォルトの設定です。

スタンバイ・データベースが存在していなくても、ARCHIVE_LAG_TARGET 初期化パラメータを設定できます。たとえば、強制的にログ・スイッチを発生させてアーカイブを行うために、ARCHIVE_LAG_TARGET 初期化パラメータを設定できます。

ARCHIVE_LAG_TARGET は動的パラメータで、ALTER SYSTEM SET 文を使用して設定できます。

注意： Oracle Real Application Clusters 環境では、すべてのインスタンスの ARCHIVE_LAG_TARGET パラメータに同じ値を指定する必要があります。異なる値を設定すると指定外の動作が起こるため、お薦めできません。

ARCHIVE_LAG_TARGET の設定に影響する要因

ARCHIVE_LAG_TARGET パラメータを設定するかどうか、またはその設定値を決定するには、次の要因を考慮してください。

- 切替え（およびアーカイブ）に伴うオーバーヘッド
- ログがいっぱいになったときに発生する通常のログ・スイッチの頻度
- スタンバイ・データベースで許容できる REDO 損失の量

指定した間隔より短い頻度ですでにログ・スイッチが自然に発生している状況では、ARCHIVE_LAG_TARGET を設定しても特に利点はありません。ただし、REDO の生成速度が一定でない場合は、時間間隔を指定することで、各カレント・ログが使用される時間範囲の上限を設定できます。

ARCHIVE_LAG_TARGET 初期化パラメータに極端に小さい値を指定すると、パフォーマンスに悪影響を及ぼすことがあります。これは、小さい値によって頻繁にログ・スイッチが発生するためです。このパラメータには、プライマリ・データベースのパフォーマンスを低下させないように適切な値を設定してください。

オンライン REDO ログ・グループおよびメンバーの作成

データベースのオンライン REDO ログを事前に計画し、データベースの作成時に必要なオンライン REDO ログ・ファイルのグループとメンバーをすべて作成してください。しかし、グループやメンバーの追加作成が必要な状況もあります。たとえば、オンライン REDO ログにグループを追加することによって、REDO ログ・グループの可用性の問題を解決できます。

新たにオンライン REDO ログ・グループおよびメンバーを作成するには、ALTER DATABASE システム権限が必要です。データベースには、最大 MAXLOGFILES 個までグループを作成できます。

> **関連項目：** ALTER DATABASE 文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

オンライン REDO ログ・グループの作成

オンライン REDO ログ・ファイルの新しいグループを作成するには、SQL 文 ALTER DATABASE で ADD LOGFILE 句を指定します。

次の文は、データベースに新しい REDO ログ・グループを追加します。

```
ALTER DATABASE
  ADD LOGFILE ('/oracle/dbs/log1c.rdo', '/oracle/dbs/log2c.rdo') SIZE 500K;
```

> **注意：** オペレーティング・システム上のファイルを作成する位置を指定するには、新しいログ・メンバーのファイル名を絶対パスで指定してください。そうしないと、ファイルはオペレーティング・システムごとに異なるデータベースのデフォルトのディレクトリまたはカレント・ディレクトリに作成されます。

また、次のように GROUP オプションを使用して、グループの識別番号を指定することもできます。

```
ALTER DATABASE
  ADD LOGFILE GROUP 10 ('/oracle/dbs/log1c.rdo', '/oracle/dbs/log2c.rdo')
      SIZE 500K;
```

グループ番号を使用することによって、REDO ログ・グループの管理がより簡単になります。ただし、グループ番号には 1 から MAXLOGFILES の範囲の値を使用する必要があります。REDO ログ・ファイルのグループ番号をスキップする（つまり、グループに 10、20、30 などの番号を付ける）と、データベースの制御ファイル内で領域が消費されてしまうので、グループ番号はスキップしないでください。

オンライン REDO ログ・メンバーの作成

完全なオンライン REDO ログ・ファイルのグループを作成する必要がない場合もあります。つまり、グループはすでに存在しているが、そのグループの 1 つ以上のメンバーが（ディスク障害などにより）削除されたために完全ではない場合です。このような場合は、既存のグループに新しいメンバーを追加できます。

既存のグループ用に新しいオンライン REDO ログ・メンバーを作成するには、SQL 文 ALTER DATABASE で ADD LOG MEMBER パラメータを指定します。次の文は、REDO ログ・グループ 2 に新しい REDO ログ・メンバーを追加します。

```
ALTER DATABASE ADD LOGFILE MEMBER '/oracle/dbs/log2b.rdo' TO GROUP 2;
```

REDO ログ・メンバーを追加する際、ファイル名は指定する必要がありますが、サイズを指定する必要はありません。新しいメンバーのサイズは、既存グループのメンバーのサイズによって決まります。

ALTER DATABASE 文を使用するときは、次の例のように、TO パラメータにグループの他のメンバーをすべて指定することによって、目的のグループを選択的に識別できます。

```
ALTER DATABASE ADD LOGFILE MEMBER '/oracle/dbs/log2c.rdo'
    TO ('/oracle/dbs/log2a.rdo', '/oracle/dbs/log2b.rdo');
```

注意： オペレーティング・システム上のファイルを作成する位置を指定するには、新しいログ・メンバーのファイル名を絶対パスで指定してください。そうしないと、ファイルはオペレーティング・システムごとに異なるデータベースのデフォルトのディレクトリまたはカレント・ディレクトリに作成されます。また、新しいログ・メンバーの状態は INVALID として表示されるので注意してください。これは正常であり、最初に使用するときにアクティブ（空白）に変更されます。

オンライン REDO ログ・メンバーの再配置および名前変更

オペレーティング・システムのコマンドを使用してオンライン REDO ログを再配置し、ALTER DATABASE 文を使用してデータベースにそのオンライン REDO ログの新しい名前（位置）を通知できます。たとえば、オンライン REDO ログ・ファイルが現在保存されているディスクを削除する場合や、複数のデータ・ファイルやオンライン REDO ログ・ファイルが同一のディスク上に格納されていて、競合を少なくするためにそれらを分離する場合に、この手順が必要となります。

オンライン REDO ログ・メンバーの名前を変更するには、ALTER DATABASE システム権限が必要です。さらに、ファイルを目的の位置にコピーするためのオペレーティング・システム権限と、データベースをオープンしてバックアップするための権限が必要となることもあります。

REDO ログを再配置したり、データベースにその他の構造上の変更を加えたりする前には、各操作の実行中に発生する問題に備えて、データベース全体のバックアップを作成してください。また、今後発生する可能性のある問題に備えて、一連のオンライン REDO ログ・ファイルを名前変更または再配置した後、ただちにデータベースの制御ファイルのバックアップを作成してください。

REDO ログを再配置する手順は、次のとおりです。これらの手順では、次の状況を想定しています。

- ログ・ファイルは、diska および diskb という 2 つのディスク上にあります。
- オンライン REDO ログは多重化されています。最初のグループはメンバー /diska/logs/log1a.rdo と /diskb/logs/log1b.rdo からなり、2 番目のグループはメンバー /diska/logs/log2a.rdo と /diskb/logs/log2b.rdo からなっています。
- diska にあるオンライン REDO ログ･ファイルを diskc に再配置する必要があります。新しいファイル名には新しい位置が反映され、/diskc/logs/log1c.rdo および /diskc/logs/log2c.rdo となります。

オンライン REDO ログ・メンバーの名前変更の手順

1. データベースを停止します。

   ```
   SHUTDOWN
   ```

2. オンライン REDO ログ・ファイルを新しい位置にコピーします。

   オンライン REDO ログ・メンバーなどのオペレーティング・システム・ファイルは、適切なオペレーティング・システム・コマンドを使用してコピーする必要があります。ファイルのコピーに関する説明は、オペレーティング・システム固有のマニュアルを参照してください。

注意： SQL*Plus の HOST コマンドを使用すると、既存の SQL*Plus を終了せずにオペレーティング・システム・コマンドを使用してファイルをコピーできます（その他のオペレーティング・システムのコマンドも実行できます）。HOST という語のかわりに 1 文字を使用するオペレーティング・システムもあります。たとえば、UNIX では！（感嘆符）が使用できます。

次の例では、オペレーティング・システム・コマンド（UNIX）を使用して、オンライン REDO ログ・メンバーを新しい位置に移動しています。

```
mv /diska/logs/log1a.rdo /diskc/logs/log1c.rdo
mv /diska/logs/log2a.rdo /diskc/logs/log2c.rdo
```

3. データベースを起動して、マウントします。ただし、オープンはしません。

```
CONNECT / as SYSDBA
STARTUP MOUNT
```

4. オンライン REDO ログ・メンバーの名前を変更します。

ALTER DATABASE 文で RENAME FILE 句を使用して、データベースのオンライン REDO ログ・ファイルの名前を変更します。

```
ALTER DATABASE
  RENAME FILE '/diska/logs/log1a.rdo', '/diska/logs/log2a.rdo'
           TO '/diskc/logs/log1c.rdo', '/diskc/logs/log2c.rdo';
```

5. 通常の操作を実行するためにデータベースをオープンします。

オンライン REDO ログの変更は、データベースがオープンされたときに有効となります。

```
ALTER DATABASE OPEN;
```

オンライン REDO ログ・グループおよびメンバーの削除

場合によっては、オンライン REDO ログ・メンバーを含むグループ全体を削除できます。たとえば、インスタンスのオンライン REDO ログのグループ数を少なくする場合などです。また、1 つ以上の特定のオンライン REDO ログ・メンバーの削除が必要になる場合もあります。たとえば、ディスク障害が発生した場合には、アクセスできないファイルに書き込まれないように、障害のあったディスク上のオンライン REDO ログ・ファイルをすべて削除します。これ以外にも、特定のオンライン REDO ログ・ファイルが不要になることがあります。たとえば、適切ではない位置にファイルを配置した場合です。

ログ・グループの削除

オンライン REDO ログ・グループを削除するには、ALTER DATABASE システム権限が必要です。オンライン REDO ログ・グループを削除する前に、次の制限と注意点について検討してください。

- グループ内のメンバー数にかかわらず、インスタンスには少なくとも 2 つのオンライン REDO ログ・ファイルのグループが必要です（1 つのグループは 1 つ以上のメンバーから構成されます）。
- オンライン REDO ログ・グループは、非アクティブである場合にのみ削除できます。カレントのグループを削除する必要がある場合は、最初にログ・スイッチを発生させる必要があります。
- 削除する前に、オンライン REDO ログ・グループがアーカイブされていることを確認します（アーカイブが使用可能になっている場合）。アーカイブされているかどうかを確認するには、V$LOG ビューを使用します。

```
SELECT GROUP#, ARCHIVED, STATUS FROM V$LOG;

   GROUP# ARC STATUS
--------- --- ----------------
        1 YES ACTIVE
        2 NO  CURRENT
        3 YES INACTIVE
        4 YES INACTIVE
```

SQL 文 ALTER DATABASE に DROP LOGFILE 句を指定して、オンライン REDO ログ・グループを削除します。

次の文は、REDO ログ・グループ 3 を削除します。

```
ALTER DATABASE DROP LOGFILE GROUP 3;
```

データベースからオンライン REDO ログ・グループを削除するときは、Oracle Managed Files 機能を使用していないかぎり、オペレーティング・システム・ファイルはディスクから削除されません。より正確に言えば、対応するデータベースの制御ファイルが更新されて、そのグループのメンバーがデータベース構造から削除されます。オンライン REDO ログ・グループを削除した後で、この処理が正常に終了したことを確認し、適切なオペレーティング・システム・コマンドを使用して、削除したオンライン REDO ログ・ファイルを実際に削除します。

Oracle Managed Files 機能を使用している場合は、オペレーティング・システム・ファイルのクリーン・アップが自動的に実行されます。

オンライン REDO ログ・メンバーの削除

オンライン REDO ログ・メンバーを削除するには、`ALTER DATABASE` システム権限が必要です。各オンライン REDO ログ・メンバーを削除する前に、次の制限と注意点について検討してください。

- オンライン REDO ログ・ファイルを削除することにより、多重オンライン REDO ログ・ファイルを一時的に非対称にしても問題はありません。たとえば、多重化したオンライン REDO ログ・ファイルのグループを使用している場合、他のすべてのグループにメンバーが 2 つずつ残っていても、あるグループのメンバーを 1 つ削除できます。ただし、すべてのグループに少なくともメンバーが 2 つ存在するように、この状態をただちに訂正し、オンライン REDO ログの単一の障害箇所が発生する可能性を取り除いてください。
- グループ内のメンバー数にかかわらず、インスタンスには常に少なくとも 2 つの有効なオンライン REDO ログ・ファイルのグループが必要です（1 つのグループは 1 つ以上のメンバーから構成されます）。削除するメンバーがグループの最後の有効なメンバーである場合は、他のメンバーが有効にならないかぎり、そのメンバーを削除できません。REDO ログ・ファイルの状態を確認するには、`V$LOGFILE` ビューを使用します。REDO ログ・ファイルは、Oracle がアクセスできないと `INVALID` になります。Oracle がそのログ・ファイルを完全でない、または正しくないと判断すると、そのログ・ファイルは `STALE` になります。この失効したログ・ファイルは、次にそのグループがアクティブ・グループになったときに、再び有効になります。
- オンライン REDO ログ・メンバーは、アクティブ・グループまたはカレント・グループの一部でない場合にのみ削除できます。アクティブ・グループのメンバーを削除する場合は、最初にログ・スイッチを発生させます。
- メンバーを削除する前に、そのオンライン REDO ログ・メンバーが属するグループがアーカイブされていることを確認します（アーカイブが使用可能になっている場合）。アーカイブされているかどうかを確認するには、`V$LOG` ビューを使用します。

非アクティブである特定のオンライン REDO ログ・メンバーを削除するには、`ALTER DATABASE` 文で `DROP LOGFILE MEMBER` 句を指定します。

次の文は、REDO ログ /oracle/dbs/log3c.rdo を削除します。

```
ALTER DATABASE DROP LOGFILE MEMBER '/oracle/dbs/log3c.rdo';
```

データベースからオンライン REDO ログ・メンバーを削除するときは、そのオペレーティング・システム・ファイルはディスクから削除されません。より正確に言えば、対応するデータベースの制御ファイルが更新されて、データベース構造からメンバーが削除されます。オンライン REDO ログ・ファイルを削除した後で、処理が正常終了したことを確認し、適切なオペレーティング・システム・コマンドを使用して、削除したオンライン REDO ログ・ファイルを実際に削除します。

アクティブ・グループのメンバーを削除するには、最初にログ・スイッチを発生させる必要があります。

ログ・スイッチの強制

ログ・スイッチは、LGWR があるオンライン REDO ログ・グループへの書込みを中止して、別のログ・グループへの書込みを開始するときに発生します。デフォルトでは、現行のオンライン REDO ログ・ファイル・グループがいっぱいになると、ログ・スイッチが自動的に発生します。

オンライン REDO ログのメンテナンス操作を実行するために、ログ・スイッチを強制的に発生させて、現在アクティブなグループを非アクティブの状態に変更することができます。たとえば、現在アクティブなグループを削除する場合は、アクティブでない状態になるまでそのグループを削除できません。また、現在アクティブなグループのメンバーが完全にいっぱいになる前に、特定の時点でそのグループをアーカイブする必要がある場合にも、ログ・スイッチの強制的な実行が必要です。このオプションは、いっぱいになるまで長い時間を必要とする大きなオンライン REDO ログ・ファイルが含まれた構成で有効です。

ログ・スイッチを強制するには、ALTER SYSTEM 権限が必要です。ALTER SYSTEM 文で SWITCH LOGFILE 句を指定します。

次の文は、ログ・スイッチを強制します。

```
ALTER SYSTEM SWITCH LOGFILE;
```

REDO ログ・ファイル内のブロックの検証

Oracle は、チェックサムを使用して REDO ログ・ファイル内のブロックを検証するように構成できます。初期化パラメータ DB_BLOCK_CHECKSUM を TRUE に設定すると、ディスクに書き込まれているすべての Oracle データベース・ブロック（REDO ログ・ブロックを含む）に対するブロック・チェックが使用可能になります。DB_BLOCK_CHECKSUM のデフォルト値は FALSE です。

ブロック・チェックを使用可能にすると、カレント・ログに書き込まれた各 REDO ログ・ブロックのチェックサムが算出されます。チェックサムは、ブロックのヘッダーに書き込まれます。Oracle は、チェックサムを使用して REDO ログ・ブロック内の破損を検出します。REDO ログ・ブロックをアーカイブ・ログ・ファイルに書き込むとき、およびリカバリ処理中にブロックがアーカイブ・ログから読み込まれるときに、そのブロックの検証が行われます。

Oracle は、REDO ログ・ブロックのアーカイブ時に破損を検出すると、グループ内の別のメンバーからそのブロックを読み込もうとします。REDO ログ・グループ内のすべてのメンバーでブロックが破損していると、アーカイブ処理は継続できません。

注意： DB_BLOCK_CHECKSUM を使用可能にすると、多少のオーバーヘッドとデータベースのパフォーマンスの低下が発生します。データベースのパフォーマンスを監視して、パフォーマンスを犠牲にしてでもデータ・ブロックのチェックサムを使用して破損を検出する利点があるかを判断してください。

関連項目： DB_BLOCK_CHECKSUM 初期化パラメータの説明は、『Oracle9*i* データベース・リファレンス』を参照してください。

オンライン REDO ログ・ファイルの初期化

データベースがオープンしている間にオンライン REDO ログ・ファイルが破損し、その結果アーカイブが継続できなくなり、データベース・アクティビティが停止することがあります。このような状況では、ALTER DATABASE CLEAR LOGFILE 文を使用して、データベースを停止せずにファイルを再初期化できます。

次の文は、REDO ログ・グループ 3 のログ・ファイルを初期化します。

```
ALTER DATABASE CLEAR LOGFILE GROUP 3;
```

この文は、REDO ログの削除が不可能な次の 2 つの状況に対応できます。

- ログ・グループが 2 つのみの場合
- 破損した REDO ログ・ファイルがカレント・グループに属する場合

破損した REDO ログ・ファイルがアーカイブされていない場合は、この文に UNARCHIVED キーワードを使用します。

```
ALTER DATABASE CLEAR UNARCHIVED LOGFILE GROUP 3;
```

この文によって、破損した REDO ログは初期化され、アーカイブを回避できます。初期化された REDO ログは、アーカイブされていなくても使用できます。

バックアップのリカバリに必要なログ・ファイルを初期化すると、そのバックアップからのリカバリ処理ができなくなります。Oracle はアラート・ログに、そのバックカップからのリカバリ処理ができないことを示すメッセージを書き込みます。

注意： アーカイブされていない REDO ログ・ファイルを初期化する場合は、データベースのバックアップをもう 1 つ作成する必要があります。

オフライン表領域をオンラインにするために必要な、アーカイブされていない REDO ログを初期化するには、ALTER DATABASE CLEAR LOGFILE 文で UNRECOVERABLE DATAFILE 句を指定します。

オフライン表領域をオンラインにするために必要な REDO ログを初期化すると、その表領域は二度とオンラインにはできません。表領域を削除するか、不完全リカバリを実行する必要があります。正常にオフライン化された表領域には、リカバリは必要ありません。

オンライン REDO ログ情報の表示

オンライン REDO ログ情報を表示するには、次のビューを使用します。

ビュー	説明
V$LOG	制御ファイルの REDO ログ・ファイル情報が表示されます。
V$LOGFILE	REDO ログ・グループとメンバーおよびメンバーの状態を識別します。
V$LOG_HISTORY	ログの履歴情報が含まれます。

次の問合せは、データベースのオンライン REDO ログに関する制御ファイル情報を返します。

```
SELECT * FROM V$LOG;

GROUP#  THREAD#    SEQ    BYTES  MEMBERS  ARC STATUS     FIRST_CHANGE# FIRST_TIM
------  -------  -----  -------  -------  --- ---------  ------------- ---------
     1        1  10605  1048576        1  YES ACTIVE          11515628 16-APR-00
     2        1  10606  1048576        1  NO  CURRENT         11517595 16-APR-00
     3        1  10603  1048576        1  YES INACTIVE        11511666 16-APR-00
     4        1  10604  1048576        1  YES INACTIVE        11513647 16-APR-00
```

グループのすべてのメンバーの名前を表示するには、次の問合せを使用します。

```
SELECT * FROM V$LOGFILE;

GROUP#   STATUS  MEMBER
------  -------  ---------------------------------
     1           D:\ORANT\ORADATA\IDDB2\REDO04.LOG
     2           D:\ORANT\ORADATA\IDDB2\REDO03.LOG
     3           D:\ORANT\ORADATA\IDDB2\REDO02.LOG
     4           D:\ORANT\ORADATA\IDDB2\REDO01.LOG
```

メンバーの STATUS が空白の場合、そのファイルは使用中です。

関連項目： これらのビューの詳細情報は、『Oracle9*i* データベース・リファレンス』を参照してください。

8

アーカイブ REDO ログの管理

この章では、REDO データをアーカイブする方法について説明します。この章の内容は、次のとおりです。

- アーカイブ REDO ログの概要
- NOARCHIVELOG モードと ARCHIVELOG モードの選択
- アーカイブの制御
- アーカイブ先の指定
- ログ転送モードの指定
- アーカイブ先の障害管理
- ARCn プロセスの複数指定によるアーカイブ・パフォーマンスのチューニング
- ARCHIVELOG プロセスによって生成されるトレース出力の制御
- アーカイブ REDO ログに関する情報の表示

関連項目： Oracle Real Application Clusters 環境におけるアーカイブに固有の情報は、『Oracle9*i* Real Application Clusters 管理』を参照してください。

アーカイブ REDO ログの概要

Oracle では、いっぱいになったオンライン REDO ログ・ファイルのグループを 1 つ以上のオフライン・アーカイブ先に保存できます。これを総称して、**アーカイブ REDO ログ**または単にアーカイブ・ログと呼びます。オンライン REDO ログをアーカイブ REDO ログに変更するプロセスを**アーカイブ**と呼びます。このプロセスを実行できるのは、データベースが **ARCHIVELOG モード**で稼働しているときのみです。自動アーカイブと手動アーカイブのいずれかを選択できます。

アーカイブ REDO ログ・ファイルは、オンライン REDO ログ・グループのうちいっぱいになった同一メンバーの 1 つのコピーです。このファイルには、REDO ログ・グループの同一メンバー内に存在する REDO エントリが含まれ、そのグループの一意のログ順序番号も格納されています。たとえば、オンライン REDO ログを多重化しており、グループ 1 にメンバー・ファイル `a_log1` および `b_log1` が含まれている場合、アーカイバ・プロセス（ARC*n*）によってこれらの同一メンバーの 1 つがアーカイブされます。万一 `a_log1` が破損した場合でも、ARC*n* によってそれと同一の `b_log1` をアーカイブできます。このアーカイブ REDO ログには、アーカイブを使用可能にした後に作成された各グループのコピーが含まれます。

`ARCHIVELOG` モードで稼働しているときは、オンライン REDO ログ・グループがアーカイブされないかぎり、ログ・ライター・プロセス（LGWR）は REDO ログ・グループを再利用（上書き）できません。自動アーカイブが使用可能な場合は、バックグラウンド・プロセス ARC*n* によってアーカイブ操作が自動的に実行されます。Oracle は必要に応じて複数のアーカイバ・プロセスを起動して、いっぱいになったオンライン REDO ログのアーカイブが遅れないようにします。

アーカイブ REDO ログは、次の操作に使用できます。

- データベースのリカバリ
- スタンバイ・データベースの更新
- LogMiner ユーティリティを使用してデータベースの履歴情報を取得する操作

NOARCHIVELOG モードと ARCHIVELOG モードの選択

ここでは、データベースを NOARCHIVELOG モードまたは ARCHIVELOG モードで稼働する際の考慮点について説明します。この項の内容は、次のとおりです。

- NOARCHIVELOG モードによるデータベースの実行
- ARCHIVELOG モードによるデータベースの実行

NOARCHIVELOG モードによるデータベースの実行

データベースを NOARCHIVELOG モードで実行すると、オンライン REDO ログはアーカイブされません。データベースの制御ファイルは、グループがいっぱいになってもアーカイブする必要がないことを示します。したがって、ログ・スイッチが発生して、いっぱいになったグループがアクティブでなくなると、そのグループは LGWR で再利用できるようになります。

いっぱいになったオンライン REDO ログ・ファイル・グループをアーカイブ可能にするかどうかは、データベース上で実行されているアプリケーションの可用性と信頼性の要件によって決まります。ディスク障害の発生時にもデータベース内のデータが失われないようにする場合は、ARCHIVELOG モードを使用します。いっぱいになったオンライン REDO ログ・ファイルをアーカイブすると、管理作業が増えます。

NOARCHIVELOG モードでは、データベースはインスタンス障害からのみ保護され、メディア障害からは保護されません。インスタンスのリカバリに使用できるのは、オンライン REDO ログのグループに格納されているデータベースへの最新の変更のみです。つまり、NOARCHIVELOG モードでは、途中でメディア障害が発生すると、最後にデータベース全体のバックアップを行ったときの状態までしかデータベースをリストア（リカバリではなく）できません。それ以降のトランザクションはリカバリできません。

また、NOARCHIVELOG モードでは、オンライン表領域のバックアップを実行できません。さらに、データベースを ARCHIVELOG モードで操作していたときに作成されたオンライン表領域のバックアップも使用できません。NOARCHIVELOG モードで操作しているデータベースのリストアに使用できるのは、データベースがクローズされているときに作成されたデータベース全体のバックアップのみです。したがって、NOARCHIVELOG モードでデータベースを操作する場合は、データベース全体のバックアップを短い間隔で定期的に作成してください。

ARCHIVELOG モードによるデータベースの実行

データベースを ARCHIVELOG モードで実行するときは、オンライン REDO ログのアーカイブを指定します。データベースの制御ファイルは、いっぱいになったオンライン REDO ログ・ファイルのグループがアーカイブされるまでは、LGWR でこのグループを使用できないことを示します。いっぱいになったグループは、ログ・スイッチの発生直後からアーカイブに使用できます。

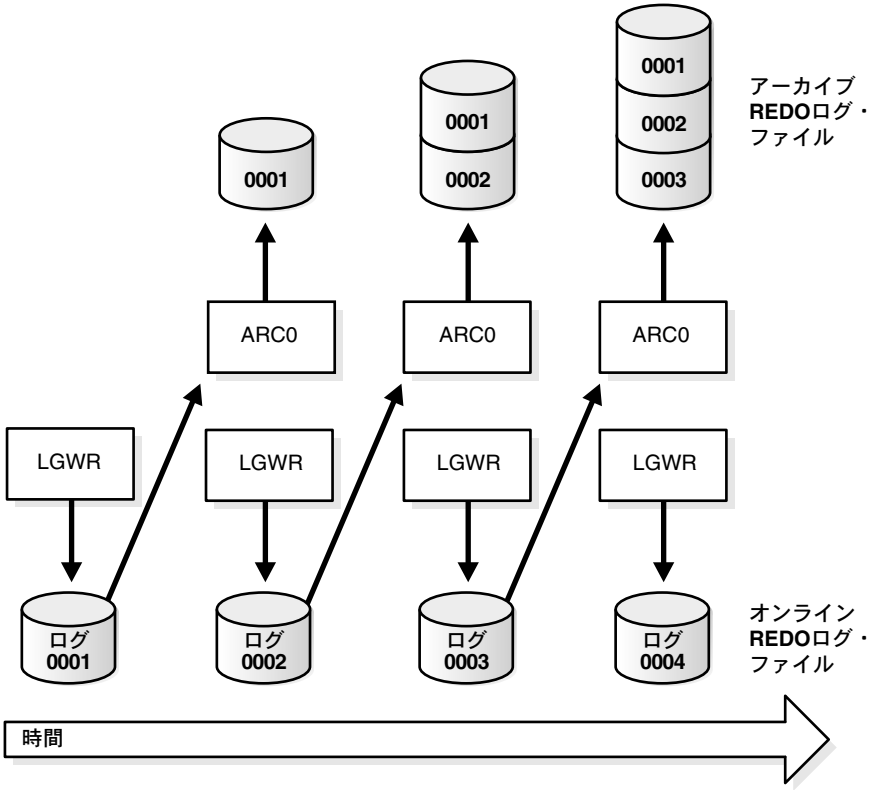
いっぱいになったグループのアーカイブには、次のような利点があります。

- データベースのバックアップ、オンライン REDO ログおよびアーカイブ REDO ログ・ファイルが揃っていると、オペレーティング・システムやディスクに障害が発生しても、コミットされたすべてのトランザクションをリカバリできることが保証されます。
- アーカイブ・ログを保管していれば、オープンしているデータベースを通常どおり使用している状態で取得したバックアップを使用できます。
- オリジナル・データベースのアーカイブ REDO ログを絶えずスタンバイ・データベースに適用し、スタンバイをオリジナルとともに最新の状態に保つことができます。

いっぱいになったオンライン REDO ログ・グループのアーカイブ計画を作成してください。いっぱいになったオンライン REDO ログ・ファイルを自動的にアーカイブするようにインスタンスを構成する方法と、手動でアーカイブする方法があります。通常は、自動アーカイブのほうが便利で効率的です。図 8-1 は、アーカイバ・プロセス（この図では ARC0）によって、いっぱいになったオンライン REDO ログ・ファイルがデータベースのアーカイブ REDO ログに書き込まれる過程を示しています。

分散データベース内のデータベースをすべて ARCHIVELOG モードで操作している場合は、調整式分散データベース・リカバリを実行できます。ただし、分散データベース内のデータベースのいずれかが NOARCHIVELOG モードで操作されている場合、（すべてのデータベースの整合性を維持するために）グローバルな分散データベースのリカバリは、NOARCHIVELOG モードで操作しているデータベース全体の最新のバックアップによって制限されます。

図 8-1　ARCHIVELOG モードによるオンライン REDO ログ・ファイルの使用



アーカイブの制御

この項では、データベースのアーカイブ・モードを制御する方法と、アーカイブ・プロセスを制御する方法について説明します。この項の内容は、次のとおりです。

- 初期データベース・アーカイブ・モードの設定
- データベース・アーカイブ・モードの変更
- 自動アーカイブの使用可能
- 自動アーカイブの使用禁止
- 手動アーカイブの実行

関連項目： アーカイブ・モードの制御に関する追加情報は、オペレーティング・システム固有の Oracle マニュアルを参照してください。

初期データベース・アーカイブ・モードの設定

データベースの最初のアーカイブ・モードは、CREATE DATABASE 文でデータベース作成の一部として設定します。多くの場合、データベース作成時に生成される REDO 情報はアーカイブする必要がないため、データベース作成時には NOARCHIVELOG モード（デフォルト）を使用できます。初期のアーカイブ・モードを変更するかどうかは、データベースの作成後に決定します。

注意： Oracle をインストールするときにデータベースが自動的に作成される場合、そのデータベースの最初のアーカイブ・モードは、オペレーティング・システムによって異なります。

データベース・アーカイブ・モードの変更

データベースのアーカイブ・モードを切り替えるには、ARCHIVELOG または NOARCHIVELOG オプションを指定して ALTER DATABASE 文を使用します。次の手順は、データベースのアーカイブ・モードを NOARCHIVELOG から ARCHIVELOG に切り替えます。

1. データベース・インスタンスを停止します。

   ```
   SHUTDOWN
   ```

   データベースがオープンされている場合は、アーカイブ・モードを切り替える前にクローズし、対応するインスタンスを停止する必要があります。メディア・リカバリを必要とするデータ・ファイルがある場合は、アーカイブを使用禁止にできません。

2. データベースのバックアップを作成します。

   データベースに重要な変更をする前に、データベースのデータを保護するため必ずバックアップを作成してください。これは NOARCHIVELOG モードでのデータベースの最終バックアップとなり、ARCHIVELOG モードへの切替え中に問題が生じた場合に使用できます。『Oracle9*i* ユーザー管理バックアップおよびリカバリ・ガイド』または『Oracle9*i* Recovery Manager ユーザーズ・ガイド』を参照してください。

3. 初期化パラメータ・ファイルを編集して、自動アーカイブを使用可能にするかどうか（8-8 ページの「自動アーカイブの使用可能」を参照）とアーカイブ・ログ・ファイルのアーカイブ先（8-11 ページの「アーカイブ先の指定」を参照）を初期化パラメータで指定します。

4. 新しいインスタンスを起動し、データベースをマウントします。オープンはしません。

   ```
   STARTUP MOUNT
   ```

   アーカイブを使用可能または使用禁止にするには、データベースをマウントして、オープンしないようにする必要があります。

5. データベースのアーカイブ・モードを切り替えます。通常の操作を実行するためにデータベースをオープンします。

   ```
   ALTER DATABASE ARCHIVELOG;
   ALTER DATABASE OPEN;
   ```

6. データベースを停止します。

   ```
   SHUTDOWN IMMEDIATE
   ```

7. データベースのバックアップを作成します。

   データベースのアーカイブ・モードを変更すると、制御ファイルが更新されます。変更後は、すべてのデータベース・ファイルと制御ファイルのバックアップを作成する必要があります。以前のバックアップは NOARCHIVELOG モードで作成されているため、使用できなくなります。

関連項目： Oracle9*i* Real Application Clusters 使用時のアーカイブ・モード切替えの詳細は、『Oracle9*i* Real Application Clusters 管理』を参照してください。

自動アーカイブの使用可能

オンライン REDO ログの自動アーカイブを使用可能にできます。自動アーカイブを使用可能にすると、グループがいっぱいになってもグループをコピーする操作は不要であり、自動的にアーカイブされます。ただし、自動アーカイブが使用可能になっていても、手動アーカイブは実行できます。8-10 ページの「手動アーカイブの実行」を参照してください。

自動アーカイブは、インスタンスの起動前後に使用可能にできます。インスタンスの起動後に自動アーカイブを使用可能にするには、Oracle に管理者権限（AS SYSDBA）で接続するか、ALTER SYSTEM システム権限を持つ必要があります。

自動アーカイブを使用可能にする前に、アーカイブ REDO ログのアーカイブ先とファイル名の形式を指定したことを確認します。この操作については、8-11 ページの「アーカイブ先の指定」を参照してください。

注意： データベースが ARCHIVELOG モードでない場合、ログ・ファイルは自動的にアーカイブされません。

インスタンス起動時の自動アーカイブの使用可能

インスタンスを起動するたびに、いっぱいになったグループの自動アーカイブを使用可能にするには、次のように、データベースの初期化パラメータ・ファイルで初期化パラメータ LOG_ARCHIVE_START を TRUE に設定します。

```
LOG_ARCHIVE_START=TRUE
```

新しい値は、次回データベースを起動するときに有効になります。

インスタンス起動後の自動アーカイブの使用可能

現行のインスタンスを停止せずに、いっぱいになったオンライン REDO ログ・グループの自動アーカイブを使用可能にするには、ARCHIVE LOG START 句を指定した ALTER SYSTEM 文を使用します。次に例を示します。

```
ALTER SYSTEM ARCHIVE LOG START;
```

この場合、アーカイブ先を任意に指定できます。

注意： ALTER SYSTEM 文を使用して自動アーカイブを使用可能にした後に、インスタンスを停止して再起動すると、そのインスタンスは初期化パラメータ・ファイルの設定によって再度初期化されます。自動アーカイブは、設定に従って使用可能または使用禁止になります。REDO ログ・ファイルを常に自動的にアーカイブすることを意図している場合は、初期化パラメータに LOG_ARCHIVE_START = TRUE を含める必要があります。

アーカイバ・プロセス数の制御

Oracle は必要に応じて追加のアーカイバ・プロセス（ARC*n*）を起動して、いっぱいになった REDO ログの自動処理が遅れないようにします。ただし、追加の ARC*n* プロセスの起動に伴う実行時のオーバーヘッドを回避するには、LOG_ARCHIVE_MAX_PROCESSES 初期化パラメータを使用して、インスタンス起動時に開始するプロセスの数を指定します。最大 10 の ARC*n* プロセスを開始できます。

このパラメータを使用して、インスタンスで起動できる ARC*n* プロセスの数を制限することもできます。これにより、指定した数を超えるプロセスを起動することはできなくなります。

LOG_ARCHIVE_MAX_PROCESSES は動的で、ALTER SYSTEM 文を使用して変更できます。次の文は、現在実行されている ARC*n* プロセスの数を増加（または減少）させます。

```
ALTER SYSTEM SET LOG_ARCHIVE_MAX_PROCESSES=3;
```

Oracle はシステムの処理負荷に従って ARC*n* プロセスを適切に調整するため、通常、LOG_ARCHIVE_MAX_PROCESSES 初期化パラメータをデフォルト値の 2 から変更する必要はありません。

自動アーカイブの使用禁止

オンライン REDO ログ・グループの自動アーカイブは、いつでも使用禁止にできます。一度自動アーカイブを使用禁止にすると、オンライン REDO ログ・ファイルのグループを適時手動でアーカイブ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、自動アーカイブが使用禁止になっている場合は、オンライン REDO ログ・ファイルのグループがすべていっぱいになったときにアーカイブを行わないと、LGWR はオンライン REDO ログ・グループの非アクティブなグループを再利用できなくなります。したがって、必要なアーカイブが完了するまでデータベース操作は一時的に停止します。

自動アーカイブは、インスタンスの起動時または起動後に使用禁止にできます。インスタンスの起動後に自動アーカイブを使用禁止にするには、管理者権限で接続するか、ALTER SYSTEM 権限を持つ必要があります。

インスタンス起動時の自動アーカイブの使用禁止

いっぱいになったオンライン REDO ログ・グループの自動アーカイブをデータベースの起動時に使用禁止にするには、LOG_ARCHIVE_START 初期化パラメータを FALSE に設定します。

```
LOG_ARCHIVE_START=FALSE
```

インスタンス起動後の自動アーカイブの使用禁止

現行のインスタンスを停止せずにいっぱいのオンライン REDO ログ・グループの自動アーカイブを使用禁止にするには、ARCHIVE LOG STOP パラメータを指定した SQL 文 ALTER SYSTEM を使用します。次の文はアーカイブを停止します。

```
ALTER SYSTEM ARCHIVE LOG STOP;
```

自動アーカイブを使用禁止にする際に、ARC*n* が REDO ログ・グループをアーカイブしている場合、ARC*n* はカレント・グループのアーカイブを完了しますが、次にいっぱいになったオンライン REDO ログ・グループのアーカイブは開始しません。

自動アーカイブを使用禁止にするためにインスタンスを停止する必要はありません。ただし、自動アーカイブが使用禁止になった後にインスタンスを停止して再起動すると、そのインスタンスは初期化パラメータ・ファイルの設定によって再度初期化されます。自動アーカイブは、設定に従って使用可能または使用禁止になります。

手動アーカイブの実行

データベースを ARCHIVELOG モードで操作しており、自動アーカイブが使用可能になっていない場合は、いっぱいになった REDO ログ・ファイルの非アクティブ・グループをアーカイブしないと、データベース操作が一時的に停止する可能性があります。

また、自動アーカイブが使用可能になっていても、手動アーカイブを使用できます。その場合、いっぱいになったオンライン REDO ログ・メンバーの非アクティブ・グループは、別の位置に再度アーカイブされます。ただし、この場合は、手動アーカイブが完了していなくてもインスタンスによって REDO ログ・グループを再利用できるので、ファイルは上書きされる場合があります。このような場合は、アラート・ファイルにエラー・メッセージが書き込まれます。

いっぱいのオンライン REDO ログ・グループを手動でアーカイブするには、管理者権限を使用して接続してください。手動アーカイブを実行するには、ALTER SYSTEM 文で ARCHIVE LOG 句を指定します。次の文は、アーカイブされていないログ・ファイルをすべてアーカイブします。

```
ALTER SYSTEM ARCHIVE LOG ALL;
```

アーカイブ先の指定

REDO ログをアーカイブする場合は、アーカイブ先を指定し、アーカイブ先の様々な状態を理解する必要があります。8-25 ページの「アーカイブ REDO ログに関する情報の表示」に示す動的パフォーマンス・ビュー（V$）を使用して、アーカイブ情報にアクセスする方法を決めておきます。

この項の内容は、次のとおりです。

- アーカイブ先の指定
- アーカイブ先の状態の理解

アーカイブ先の指定

ログのアーカイブ先を**単一**にするか、または**多重化**するかを決める必要があります。アーカイブ先を多重化すると、ログは複数の場所にアーカイブされます。次のどちらかの方法で初期化パラメータを設定します。

方法	初期化パラメータ	ホスト	例
1	LOG_ARCHIVE_DEST_*n* *n* は 1 ～ 10 の整数	ローカルまたはリモート	LOG_ARCHIVE_DEST_1 = 'LOCATION = /disk1/arc' LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1'
2	LOG_ARCHIVE_DEST および LOG_ARCHIVE_DUPLEX_DEST	ローカルのみ	LOG_ARCHIVE_DEST = '/disk1/arc' LOG_ARCHIVE_DUPLEX_DEST ='/disk2/arc'

関連項目：

- REDO ログのアーカイブ制御に使用される初期化パラメータの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。
- スタンバイ・アーカイブ先の指定に使用する LOG_ARCHIVE_DEST_*n* 初期化パラメータの使用方法は、『Oracle9*i* Data Guard 概要および管理』を参照してください。この初期化パラメータには他にも指定できるキーワードがありますが、このマニュアルでは説明されていません。

方法 1: LOG_ARCHIVE_DEST_*n* パラメータの使用

最初の方法では、LOG_ARCHIVE_DEST_*n* パラメータ（*n* は 1 ～ 10 の整数）を使用して、1 ～ 10 の異なるアーカイブ先を指定します。末尾に番号が付いた各パラメータによって、特定のアーカイブ先を一意に識別します。

LOG_ARCHIVE_DEST_*n* の位置は、次のキーワードを使用して指定します。

キーワード	指定内容	例
LOCATION	ローカル・ファイル・システムの位置	LOG_ARCHIVE_DEST_1 = 'LOCATION = /disk1/arc'
SERVICE	Oracle Net のサービス名を介したリモート・アーカイブ	LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1'

LOCATION キーワードを使用する場合は、オペレーティング・システムに有効なパス名を指定します。SERVICE を指定すると、tnsnames.ora ファイルを介してネット・サービス名が接続記述子に変換されます。この記述子には、リモート・データベースへの接続に必要な情報が含まれています。Oracle がスタンバイ・データベースの制御ファイルのログ履歴を正しく更新できるように、サービス名には対応するデータベース SID が必要です。

LOG_ARCHIVE_DEST_*n* 初期化パラメータを使用してアーカイブ REDO ログのアーカイブ先を設定する手順は、次のとおりです。

1. SQL*Plus を使用してデータベースを停止します。

   ```
   SHUTDOWN
   ```

2. LOG_ARCHIVE_DEST_*n* パラメータを編集し、1 ～ 10 のアーカイブ先を指定します。LOCATION キーワードには、オペレーティング・システム固有のパス名を指定します。たとえば、次のように入力します。

   ```
   LOG_ARCHIVE_DEST_1 = 'LOCATION = /disk1/archive'
   LOG_ARCHIVE_DEST_2 = 'LOCATION = /disk2/archive'
   LOG_ARCHIVE_DEST_3 = 'LOCATION = /disk3/archive'
   ```

   スタンバイ・データベースにアーカイブする場合は、SERVICE キーワードを使用して、tnsnames.ora ファイルに含まれる有効なネット・サービス名を指定します。たとえば、次のように入力します。

   ```
   LOG_ARCHIVE_DEST_4 = 'SERVICE = standby1'
   ```

3. LOG_ARCHIVE_FORMAT 初期化パラメータを編集します。ファイル名にログ順序番号を含めるには %s を使用し、スレッド番号を含めるには %t を使用します。番号の左をゼロで埋めるには、大文字（%S および %T）を使用します。たとえば、次のように入力します。

```
LOG_ARCHIVE_FORMAT = arch%s.arc
```

前述の設定では、ログ順序番号 100、101 および 102 について次のようなアーカイブ・ログが生成されます。

```
/disk1/archive/arch100.arc, /disk1/archive/arch101.arc,
/disk1/archive/arch102.arc

/disk2/archive/arch100.arc, /disk2/archive/arch101.arc,
/disk2/archive/arch102.arc

/disk3/archive/arch100.arc, /disk3/archive/arch101.arc,
/disk3/archive/arch102.arc
```

方法 2: LOG_ARCHIVE_DEST および LOG_ARCHIVE_DUPLEX_DEST の使用

2 番目の方法では、最大 2 つのアーカイブ先ディレクトリを指定できます。この方法では、LOG_ARCHIVE_DEST パラメータを使用して **1 次**アーカイブ先を指定し、必要に応じて LOG_ARCHIVE_DUPLEX_DEST で **2 次**アーカイブ先を指定します。Oracle では、REDO ログはどちらかのパラメータで指定したすべてのアーカイブ先ディレクトリにアーカイブされます。

方法 2 を使用する手順は、次のとおりです。

1. SQL*Plus を使用してデータベースを停止します。

```
SHUTDOWN
```

2. LOG_ARCHIVE_DEST および LOG_ARCHIVE_DUPLEX_DEST パラメータにアーカイブ先を指定します。ALTER SYSTEM 文を使用して、LOG_ARCHIVE_DUPLEX_DEST を動的に指定することもできます。たとえば、次のように入力します。

```
LOG_ARCHIVE_DEST = '/disk1/archive'
LOG_ARCHIVE_DUPLEX_DEST = '/disk2/archive'
```

3. LOG_ARCHIVE_FORMAT パラメータを編集します。ファイル名にログ順序番号を含めるには %s を使用し、スレッド番号を含めるには %t を使用します。番号の左をゼロで埋めるには、大文字（%S および %T）を使用します。たとえば、次のように入力します。

```
LOG_ARCHIVE_FORMAT = arch_%t_%s.arc
```

たとえば、前述の設定では、スレッド 1 のログ順序番号 100 および 101 について次のようなアーカイブ・ログが生成されます。

```
/disk1/archive/arch_1_100.arc, /disk1/archive/arch_1_101.arc
/disk2/archive/arch_1_100.arc, /disk2/archive/arch_1_101.arc
```

関連項目： アーカイブとスタンバイ・データベースの詳細は、次のマニュアルを参照してください。

- 『Oracle9*i* ユーザー管理バックアップおよびリカバリ・ガイド』
- 『Oracle9*i* Recovery Manager ユーザーズ・ガイド』
- 『Oracle9*i* Data Guard 概要および管理』

アーカイブ先の状態の理解

各アーカイブ先は次のような可変特性を持っており、これらの特性によってその状態が決まります。

- **Valid/Invalid** －ディスクの位置またはサービス名情報が指定されているかどうかと、それらが有効かどうかを示します。
- **Enabled/Disabled** －位置の使用可能状態と、Oracle がアーカイブ先を使用できるかどうかを示します。
- **Active/Inactive** －アーカイブ先へのアクセスに問題があったかどうかを示します。

これらの特性は、何通りかの組合せが可能です。インスタンスの各アーカイブ先について現在の状態などの情報を取得するには、V$ARCHIVE_DEST ビューを問い合せます。

ビューで表示される位置の状態は、表 8-1 に示す特性によって決まります。アーカイブを使用する際は、その特性が Valid、Enabled および Active である必要があります。

表 8-1 アーカイブ先の状態

	特性			
状態	**Valid**	**Enabled**	**Active**	**意味**
VALID	○	○	○	ユーザーがアーカイブ先を適切に初期化しているため、アーカイブ操作に使用できます。
INACTIVE	×	N/A	N/A	ユーザーがアーカイブ先情報を指定していないか、または削除しました。
ERROR	○	○	×	アーカイブ先ファイルの作成または書込み中にエラーが発生しました。エラー・データを参照してください。
FULL	○	○	×	アーカイブ先がいっぱいです（ディスク領域が残っていません）。

表 8-1 アーカイブ先の状態（続き）

	特性			
状態	**Valid**	**Enabled**	**Active**	**意味**
DEFERRED	○	×	○	ユーザーがアーカイブ先を手動で一時的に使用禁止にしています。
DISABLED	○	×	×	ユーザーがエラーの発生後にアーカイブ先を手動で一時的に使用禁止にしています。エラー・データを参照してください。
BAD PARAM	N/A	N/A	N/A	パラメータ・エラーが発生しました。エラー・データを参照してください。通常、この状態は LOG_ARCHIVE_START 初期化パラメータが設定されていない場合にのみ発生します。

LOG_ARCHIVE_DEST_STATE_*n* 初期化パラメータ（*n* は 1 ～ 10 の整数）を使用すると、*n* で指定したアーカイブ先の使用可能状態を制御できます。アーカイブ先の状態を示す値は、ENABLE、DEFER または ALTERNATE の 3 つです。値 ENABLE は、アーカイブ先として Oracle が使用できることを示します。DEFER は、その位置が一時的に使用禁止になっていることを示します。3 番目の値 ALTERNATE は、代替アーカイブ先を意味します。その使用可能状態は DEFER で、親アーカイブ先に障害が発生すると ENABLE になります。

ログ転送モードの指定

アーカイブ・ログをアーカイブ先に転送する場合、**ノーマル・アーカイブ転送**および**スタンバイ転送**という 2 つのモードがあります。ノーマル転送では、ファイルはローカル・ディスクに転送されます。スタンバイ転送では、ファイルはネットワークを介してローカルまたはリモートのスタンバイ・データベースに転送されます。

ノーマル転送モード

ノーマル転送モードでは、アーカイブ先はデータベースの別のディスク・ドライブです。この構成では、アーカイブがインスタンスに必要な他のファイルと競合せず、短時間で完了します。アーカイブ先は、`LOG_ARCHIVE_DEST_n` または `LOG_ARCHIVE_DEST` パラメータで指定します。

アーカイブ REDO ログ・ファイルとそれに対応するデータベース・バックアップは、ローカル・ディスクからテープなどの安価なオフライン記憶メディアに永続的に移動しておくことが理想的です。アーカイブ・ログは主としてデータベース・リカバリに使用されるので、プライマリ・データベースに障害が発生した場合でも、これらのログが安全であることを保証する必要があります。

スタンバイ転送モード

スタンバイ転送モードでは、アーカイブ先はローカルまたはリモートのスタンバイ・データベースです。

注意： ローカル・ディスク上でスタンバイ・データベースをメンテナンスすることも可能ですが、スタンバイ・データベースはリモート・サイトでメンテナンスし、最大限の障害対策を講じることをお薦めします。

スタンバイ・データベースを**管理リカバリ・モード**で操作している場合は、転送されたアーカイブ・ログを自動的に適用して、スタンバイ・データベースとソース・データベースの同期状態を維持できます。

ファイルをスタンバイ・データベースに正常に転送するには、ARC*n* またはサーバー・プロセスが次の処理を実行する必要があります。

- リモートの位置の認識
- リモート・サーバー上にある**リモート・ファイル・サーバー**（RFS）プロセスの併用によるアーカイブ・ログの転送

各 ARC*n* プロセスには、スタンバイ・アーカイブ先ごとに対応する RFS があります。たとえば、3 つの ARC*n* プロセスを 2 つのスタンバイ・データベースにアーカイブする場合、Oracle は 6 つの RFS 接続を確立します。

Oracle Net Services を使用すると、ネットワークを介してアーカイブ・ログをリモートの位置に転送できます。リモート・アーカイブを指定するには、アーカイブ先の属性としてネット・サービス名を指定します。このサービス名は、`tnsnames.ora` ファイルを介して接続記述子に変換されます。この記述子には、リモート・データベースへの接続に必要な情報が含まれています。Oracle がスタンバイ・データベースの制御ファイルのログ履歴を正しく更新できるように、サービス名には対応するデータベース SID が必要です。

RFS プロセスは接続先ノード上で実行され、ARC*n* クライアントへのネットワーク・サーバーとして機能します。最終的には、ARC*n* は情報を RFS にプッシュし、RFS がそれをスタンバイ・データベースに転送します。

RFS プロセスは、リモート接続先へのアーカイブ時に必要であり、次のタスクを受け持ちます。

- ARC*n* プロセスからのネットワーク I/O の消費
- `STANDBY_ARCHIVE_DEST` パラメータを使用した、スタンバイ・データベース上でのファイル名の作成
- リモート・サイトでのログ・ファイルの移入
- スタンバイ・データベースの制御ファイルの更新（Recovery Manager によるリカバリで使用可能にするため）

アーカイブ REDO ログは、オリジナルのデータベースの完全なレプリカであるスタンバイ・データベースをメンテナンスするうえで重要です。データベースは、スタンバイ・アーカイブ・モードで操作できます。このモードでは、スタンバイ・データベースがオリジナル・データベースからのアーカイブ REDO ログで自動的に更新されます。

関連項目：

- 『Oracle9*i* Data Guard 概要および管理』
- サービス名を使用してリモート・データベースに接続する方法の詳細は、『Oracle9*i* Net Services 管理者ガイド』を参照してください。

アーカイブ先の障害管理

アーカイブ先で発生した障害が、自動アーカイブ・モードで操作している場合のエラー原因となることがあります。Oracle では、アーカイブ先の障害に関連する問題を最小限に抑えるために、いくつかのオプションが用意されています。次の項では、これらのオプションについて説明します。

- 正常なアーカイブ先の最小数の指定
- 障害アーカイブ先への再アーカイブ

正常なアーカイブ先の最小数の指定

オプションの初期化パラメータ LOG_ARCHIVE_MIN_SUCCEED_DEST=*n*（*n* は 1 ～ 10 の整数、多重化を使用するように選択している場合は 1 ～ 2 の整数）によって、Oracle がオンライン・ログ・ファイルを再利用できるようになるまでに REDO ログ・グループを正常にアーカイブすることが必要なアーカイブ先の最小数が決まります。デフォルト値は 1 です。

必須およびオプションのアーカイブ先の指定

LOG_ARCHIVE_DEST_*n* パラメータを使用すると、アーカイブ先の属性として OPTIONAL（デフォルト）または MANDATORY を指定できます。LOG_ARCHIVE_MIN_SUCCEED_DEST=*n* パラメータでは、すべての MANDATORY アーカイブ先と、OPTIONAL の非スタンバイ・アーカイブ先をいくつか使用して、LGWR がオンライン・ログを上書きできるかどうかが判断されます。

パラメータの設定を判断するときには、次の点に注意してください。

- アーカイブ先に MANDATORY を指定しない場合は、OPTIONAL が指定されます。
- ローカル・アーカイブ先を少なくとも 1 つは指定する必要があります。この場合は、OPTIONAL または MANDATORY を宣言できます。
- LOG_ARCHIVE_MIN_SUCCEED_DEST の最小値は 1 なので、少なくとも 1 つのローカル・アーカイブ先で LOG_ARCHIVE_MIN_SUCCEED_DEST=*n* を使用すると、操作上、MANDATORY として扱われます。
- MANDATORY のスタンバイ・アーカイブ先を含め、MANDATORY のアーカイブ先のいずれかが障害を起こすと、LOG_ARCHIVE_MIN_SUCCEED_DEST パラメータの意味が失われます。
- LOG_ARCHIVE_MIN_SUCCEED_DEST には、アーカイブ先を超える値や、MANDATORY のアーカイブ先の数と OPTIONAL のローカル・アーカイブ先の数との合計を超える値は指定できません。
- MANDATORY のアーカイブ先に DEFER を指定した場合で、アーカイブ・ログがスタンバイ・サイトに転送されないままオンライン・ログが上書きされるときは、ログを手動でスタンバイ・サイトに転送する必要があります。

また、LOG_ARCHIVE_DEST および LOG_ARCHIVE_DUPLEX_DEST パラメータを使用して、アーカイブ先が必須かオプションかを指定することもできます。次の規則に注意してください。

- LOG_ARCHIVE_DEST によって宣言されたアーカイブ先は必須です。
- LOG_ARCHIVE_DUPLEX_DEST によって宣言されたアーカイブ先は、LOG_ARCHIVE_MIN_SUCCEED_DEST = 1 であればオプション、LOG_ARCHIVE_MIN_SUCCEED_DEST = 2 であれば必須です。

使用例 : 正常なアーカイブ先の数の指定

LOG_ARCHIVE_DEST_*n* および LOG_ARCHIVE_MIN_SUCCEED_DEST パラメータの関係は、使用例を見ると理解しやすくなります。

使用例 1 この例では、それぞれ OPTIONAL として宣言している 3 つのローカル・アーカイブ先にアーカイブします。表 8-2 に、この場合の LOG_ARCHIVE_MIN_SUCCEED_DEST=*n* に考えられる値を示します。

表 8-2 使用例 1 の LOG_ARCHIVE_MIN_SECCEED_DEST の値

値	意味
1	最低 1 つの OPTIONAL のアーカイブ先へのアーカイブに成功した場合にのみ、Oracle はログ・ファイルを再利用できます。
2	最低 2 つの OPTIONAL のアーカイブ先へのアーカイブに成功した場合にのみ、Oracle はログ・ファイルを再利用できます。
3	OPTIONAL のすべてのアーカイブ先へのアーカイブに成功した場合にのみ、Oracle はログ・ファイルを再利用できます。
4 以上	エラーです。値がアーカイブ先数を超えています。

この例は、LOG_ARCHIVE_DEST_*n* パラメータを使用してアーカイブ先を明示的に MANDATORY に設定していない場合でも、LOG_ARCHIVE_MIN_SUCCEED_DEST が 1、2 または 3 に設定されていれば、Oracle は必ずこれらの位置の 1 つ以上に正常にアーカイブすることを示しています。

使用例 2 この例では、次のような状況を考えます。

- MANDATORY のアーカイブ先は 2 つ指定されている。
- OPTIONAL のアーカイブ先は 2 つ指定されている。
- アーカイブ先は、いずれもスタンバイ・データベースではない。

表 8-3 に、LOG_ARCHIVE_MIN_SUCCEED_DEST=*n* に考えられる値を示します。

表 8-3 使用例 2 の LOG_ARCHIVE_MIN_SUCCEED_DEST の値

値	意味
1	Oracle は、この値を無視して MANDATORY のアーカイブ先数（この例では 2）を使用します。
2	Oracle は、OPTIONAL のアーカイブ先へのアーカイブに失敗しても、ログ・ファイルを再利用できます。
3	Oracle は、最低 1 つの OPTIONAL のアーカイブ先へのアーカイブに成功した場合にのみ、ログを再利用できます。
4	Oracle は、OPTIONAL のアーカイブ先へのアーカイブに両方とも成功した場合にのみ、ログを再利用できます。
5 以上	エラーです。値がアーカイブ先数を超えています。

この例は、アーカイブ先が少なくなるように LOG_ARCHIVE_MIN_SUCCEED_DEST を設定した場合でも、Oracle はその設定とは無関係に、MANDATORY として指定されているアーカイブ先に必ずアーカイブすることを示しています。

障害アーカイブ先への再アーカイブ

LOG_ARCHIVE_DEST_*n* パラメータの REOPEN 属性を使用して、エラーの発生後に ARC*n* が障害アーカイブ先への再アーカイブを試行するかどうかと、その時期を指定します。REOPEN は、OPEN エラーのみでなく、すべてのエラーに適用されます。

REOPEN=*n* では、ARC*n* が障害アーカイブ先の再オープンを試行するまでの最小秒数を設定します。*n* のデフォルト値は 300 秒です。値として 0（ゼロ）を指定すると、REOPEN オプションはオフになります。つまり、ARC*n* は障害発生後にアーカイブを試行しません。REOPEN キーワードを指定しない場合、ARC*n* はエラー発生後にアーカイブ先を再オープンしません。

REOPEN は、再接続とアーカイブ・ログ転送の試行回数の制限を指定するときには使用できません。REOPEN は成功または失敗で終了し、REOPEN 情報がリセットされます。

OPTIONAL アーカイブ先に REOPEN を指定すると、Oracle はエラーがある場合にオンライン・ログを上書きできます。MANDATORY のアーカイブ先に REOPEN を指定すると、正常にアーカイブできない場合に本番データベースの機能が停止します。この状況では、次の方法を検討してください。

- 障害アーカイブ先に手動でアーカイブする。
- アーカイブ先を遅延させる、アーカイブ先をオプションとして指定する、サービスを変更するのいずれかの方法によってアーカイブ先を変更する。
- アーカイブ先を削除する。

REOPEN キーワードを使用する場合は、次の点に注意してください。

- ARC*n* は、アーカイブ操作をログ・ファイルの先頭から開始する場合にのみアーカイブ先を再オープンします。現行のアーカイブ操作中に再オープンすることはありません。ARC*n* は、常に先頭からログ・コピーを再試行します。
- REOPEN 時間を指定するか、またはデフォルト設定された場合、ARC*n* は記録されたエラー発生時刻から REOPEN 間隔が経過した時刻が現在時刻より前かどうかをチェックします。現在時刻より前であれば、ARC*n* はログ・コピーを再試行します。
- REOPEN 句は、ACTIVE=TRUE のアーカイブ先状態に影響を及ぼします。VALID および ENABLED 状態は変化しません。

ARC*n* プロセスの複数指定によるアーカイブ・パフォーマンスのチューニング

ほとんどのデータベースでは、ARC*n* がシステム・パフォーマンス全体に影響を及ぼすことはありません。ただし、大規模なデータベース・サイトでは、アーカイブがシステム・パフォーマンスに影響を及ぼす場合があります。たとえば、ARC*n* の処理が非常に速い場合は、CPU サイクルがアーカイブに使用されるため、ARC*n* の実行中に全体のシステム・パフォーマンスが低下する可能性があります。逆に、ARC*n* の処理が極端に遅い場合は、システム・パフォーマンスへの悪影響はほとんどありませんが、REDO ログ・ファイルのアーカイブに時間がかかります。これにより、すべての REDO ログ・グループが使用不可になったときに REDO ログ・ファイルのアーカイブ待ちが発生するため、それがボトルネックになる可能性があります。

データベース・インスタンスごとに、最大 10 の ARC*n* プロセスを指定できます。起動時または実行時に複数処理機能を使用可能にするには、初期化パラメータ LOG_ARCHIVE_MAX_PROCESSES=*n*（*n* は 1 ～ 10 の整数）を設定します。デフォルトでは、このパラメータは 2 に設定されます。

現行の ARC*n* プロセス数が足りないために現行の作業負荷を処理できないときは、LGWR によって追加の ARC プロセスが自動的に起動されます。そのため、このパラメータの役割は、初期の ARC*n* プロセス数を指定することと、現行の数を増減させることです。初期の ARC*n* プロセス数が 4 に設定されていた場合、次の文ではプロセス数が 2 に減ります。

```
ALTER SYSTEM SET LOG_ARCHIVE_MAX_PROCESSES=2;
```

ARC*n* プロセス数を減らす場合、どのプロセスが停止するかを正確には判断できません。また、パラメータ値を 0（ゼロ）に変更することはできないため、常に少なくとも 1 つの ARC*n* プロセスがアクティブになっています。各アーカイブ・プロセスの状態情報を表示するには、V$ARCHIVE_PROCESSES ビューを問い合せます。停止したプロセスは、IDLE 状態として表示されます。

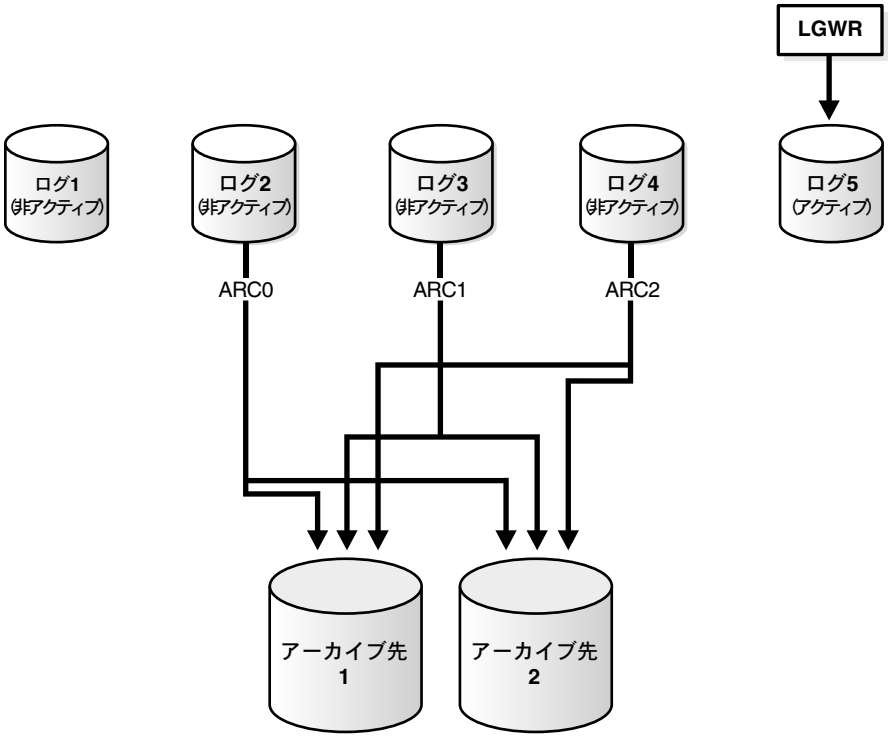
複数プロセスの作成が特に有効なケースは、次のような場合です。

- 複数のオンライン REDO ログを使用する場合
- 複数のアーカイブ先にアーカイブする場合

単一の ARC*n* プロセスが非アクティブなログを複数のアーカイブ先に書き込むよりも、LGWR が複数のオンライン REDO ログ間でログを切り替えるほうが速い場合は、ボトルネックが発生します。このボトルネックは、複数の ARC*n* プロセスによって防止できます。各 ARC*n* プロセスは一度に 1 つの非アクティブなログしか処理しませんが、必ず指定された各アーカイブ先にアーカイブします。

たとえば、5 つのオンライン REDO ログ・ファイルをメンテナンスする場合に、3 つの ARC*n* プロセスを使用してインスタンスを起動するよう設定できます。LGWR はログ・ファイルの 1 つにアクティブな状態で書き込むので、ARC*n* プロセスは各アーカイブ先に最高 3 つの非アクティブなログ・ファイルを同時にアーカイブできます。ARC*n* の各インスタンスは、図 8-2 のようにログ・ファイルを 1 つずつ担当し、定義されたすべてのアーカイブ先にアーカイブします。

図 8-2 複数の ARC*n* プロセスの使用



関連項目： アーカイブ・プロセスのチューニングの詳細は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。

ARCHIVELOG プロセスによって生成されるトレース出力の制御

バックグラウンド・プロセスは適宜、トレース・ファイルに情報を書き込みます。5-15 ページの「トレース・ファイルとアラート・ファイル」を参照してください。ARCHIVELOG プロセスの場合は、生成される出力を制御できます。

トレース・レベルを指定するには、LOG_ARCHIVE_TRACE 初期化パラメータを設定します。設定可能な値は、次のとおりです。

トレース・レベル	意味
0	アーカイブ・ログのトレースは出力されません。これはデフォルト設定です。
1	REDO ログ・ファイルのアーカイブを追跡します。
2	アーカイブ・ログのアーカイブ先ごとにアーカイブ状態を追跡します。
4	アーカイブ操作のフェーズを追跡します。
8	アーカイブ・ログのアーカイブ先のアクティビティを追跡します。
16	アーカイブ・ログのアーカイブ先の詳細アクティビティを追跡します。
32	アーカイブ・ログのアーカイブ先パラメータの変更を追跡します。
64	ARC*n* プロセスの状態のアクティビティを追跡します。
128	FAL（フェッチ・アーカイブ・ログ）サーバー関連のアクティビティを追跡します。
256	将来のリリースでサポートされる予定です。
512	非同期の LGWR アクティビティを追跡します。
1024	RFS 物理クライアントを追跡します。
2048	ARCn/RFS ハートビートを追跡します。

パラメータ値として、必要な各トレース・レベルの合計を設定することにより、トレース・レベルを組み合せることができます。たとえば、LOG_ARCHIVE_TRACE=12 に設定すると、トレース・レベル 8 および 4 の出力が生成されます。また、プライマリ・データベースとスタンバイ・データベースには、異なる値を設定できます。

LOG_ARCHIVE_TRACE パラメータのデフォルト値は 0（ゼロ）ですが、このレベルでもエラー条件が発生すると適切なアラートおよびトレース・エントリが生成されます。

このパラメータの値は、ALTER SYSTEM 文で動的に変更できます。次に例を示します。

```
ALTER SYSTEM SET LOG_ARCHIVE_TRACE=12;
```

この方法で行った変更は、次回のアーカイブ操作の開始時に有効になります。

関連項目： このパラメータをスタンバイ・データベースで使用する方法については、『Oracle9*i* Data Guard 概要および管理』を参照してください。

アーカイブ REDO ログに関する情報の表示

アーカイブ REDO ログに関する情報を表示するには、次の方法を使用します。

- 動的パフォーマンス・ビュー
- ARCHIVE LOG LIST コマンド

動的パフォーマンス・ビュー

アーカイブ REDO ログに関して役立つ情報を含む動的パフォーマンス・ビューがいくつかあります。

動的パフォーマンス・ビュー	説明
V$DATABASE	データベースが ARCHIVELOG モードと NOARCHIVELOG モードのいずれであるかを識別します。
V$ARCHIVED_LOG	制御ファイルに格納されたアーカイブ・ログ履歴情報が表示されます。リカバリ・カタログを使用している場合は、RC_ARCHIVED_LOG ビューにも同様の情報が含まれます。
V$ARCHIVE_DEST	現行インスタンス、すべてのアーカイブ先、各アーカイブ先の現行の値、モードおよび状態が表示されます。
V$ARCHIVE_PROCESSES	インスタンスの各アーカイブ・プロセスの状態情報が表示されます。
V$BACKUP_REDOLOG	アーカイブ・ログのバックアップ情報が含まれます。リカバリ・カタログを使用している場合は、RC_BACKUP_REDOLOG ビューにも同様の情報が含まれます。
V$LOG	データベースのオンライン REDO ログ・グループをすべて表示し、その中でアーカイブする必要があるグループを示します。
V$LOG_HISTORY	アーカイブ済みログや各アーカイブ・ログの SCN 範囲などのログ履歴情報が含まれます。

たとえば、次の問合せでは、アーカイブする必要があるオンライン REDO ログ・グループが表示されます。

```
SELECT GROUP#, ARCHIVED
   FROM SYS.V$LOG;

GROUP#     ARC
--------   ---
       1   YES
       2   NO
```

現行のアーカイブ・モードを確認するには、V$DATABASE ビューを問い合せます。

```
SELECT LOG_MODE FROM SYS.V$DATABASE;

LOG_MODE
------------
NOARCHIVELOG
```

関連項目： データ・ディクショナリ・ビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

ARCHIVE LOG LIST コマンド

SQL*Plus コマンドの ARCHIVE LOG LIST を使用して、接続されているインスタンスのアーカイブ情報を表示できます。次に例を示します。

```
SQL> ARCHIVE LOG LIST

Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            D:¥ORANT¥oradata¥IDDB2¥archive
Oldest online log sequence     11160
Next log sequence to archive   11163
Current log sequence           11163
```

この表示は、現行インスタンスのアーカイブ REDO ログの設定に関して必要なすべての情報を示します。

- このデータベースは現在 ARCHIVELOG モードで操作されています。
- 自動アーカイブは使用可能です。
- アーカイブ REDO ログのアーカイブ先は、D:¥ORANT¥oradata¥IDDB2¥archive です。
- いっぱいになったオンライン REDO ログ・グループのうち、最も古いものの順序番号は 11160 です。

- いっぱいになったオンライン REDO ログ・グループのうち、次にアーカイブされるものの順序番号は 11163 です。
- 現行のオンライン REDO ログ・ファイルの順序番号は 11163 です。

関連項目： `ARCHIVE LOG LIST` 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。

9

LogMiner を使用した REDO ログの分析

LogMiner ユーティリティを使用すると、SQL インタフェースを介して REDO ログを問合せできます。REDO ログには、データベース上でのアクティビティの履歴に関する情報が格納されています。

この章の内容は、次のとおりです。

- REDO ログに格納されているデータの使用
- REDO ログに格納されている情報へのアクセス
- REDO ログとディクショナリ・ファイル
- LogMiner に関する推奨事項と制限事項
- 戻されるデータのフィルタ処理
- LogMiner 情報へのアクセス
- V$LOGMNR_CONTENTS の問合せ
- REDO ログからの実際のデータ値の抽出
- サプリメンタル・ロギング
- 標準的な LogMiner セッションにおける手順
- LogMiner の使用例

この章では、コマンドラインから使用する LogMiner の機能について説明します。LogMiner の機能は、Oracle LogMiner Viewer の GUI を介して利用することもできます。LogMiner Viewer は Oracle Enterprise Manager の一部です。

REDO ログに格納されているデータの使用

ユーザー・データまたはデータ・ディクショナリに対するすべての変更内容は、REDO ログに記録されます。したがって、REDO ログには、リカバリ操作の実行に必要な情報がすべて含まれています。REDO ログ・データはアーカイブ・ファイルに保存されていることが多く、それらのデータは最初から使用可能です。REDO ログに有効な情報が確実に格納されるように、少なくとも最低限のサプリメンタル・ロギングを使用可能にする必要があります。

関連項目： 9-20 ページ「サプリメンタル・ロギング」

REDO ログに格納されているデータの使用例は、次のとおりです。

- アプリケーション・レベルで発生したエラーなど、データベースの論理的な破損が始まった時期を正確に特定するとき。アプリケーション・レベルで発生したエラーの例としては、ユーザーが誤ってデータベースを更新し、全従業員に対して 10 パーセントではなく 100 パーセントの給与増額を与えてしまった場合などが考えられます。重要なのは、時間ベースまたは変更ベースのリカバリをいつの時点から開始すればよいかを判断できるように、破損が始まった時期を正確に知ることです。これにより、データベースを破損の直前の状態にリストアできます。

  関連項目： この操作に LogMiner を使用する方法の詳細は、9-18 ページの「REDO ログからの実際のデータ値の抽出」を参照してください。

- ユーザー・エラーを検出し、できるだけ訂正するとき。これは、論理的な破損よりも可能性の高い使用例です。ユーザー・エラーには、`WHERE` 句に不適切な値が指定されていることによる誤った行の検出、不適切な値による行の更新、誤った索引の削除などがあります。
- トランザクション・レベルでのファイングレイン・リカバリを実行するために必要なアクションを判断するとき。既存の依存性を十分に理解して考慮すると、表ベースの UNDO 操作を実行して一連の変更をロールバックできる場合があります。通常は、表を以前の状態にリストアしてから、アーカイブ REDO ログを適用してロールフォワードする必要があります。
- 傾向分析を通じてパフォーマンス・チューニングと容量計画を行うとき。更新および挿入が多い表を判別できます。この情報からディスク・アクセス統計の履歴情報が得られ、チューニングに使用できます。
- 監査後処理の実行。REDO ログには、データベース上で実行される DML 文と DDL 文、実行順序および実行者の追跡に必要な情報がすべて含まれています。

REDO ログに格納されている情報へのアクセス

Oracle データベースの一部として、LogMiner を介して REDO ログに SQL でアクセスする機能が用意されています。LogMiner では、V$LOGMNR_CONTENTS 固定ビューを介して REDO ログ内の情報が表示されます。このビューには、データベースに対して行われた変更に関する次のような履歴情報が含まれています。

- データベースに対する変更のタイプ（INSERT、UPDATE、DELETE または DDL）。
- 変更が行われた SCN（SCN 列）。
- 変更がコミットされた SCN（COMMIT_SCN 列）。
- 変更が属するトランザクション（XIDUSN、XIDSLT および XIDSQN 列）。
- 変更されたオブジェクトの表名とスキーマ名（SEG_NAME および SEG_OWNER 列）。
- 変更を行うために DDL 文または DML 文を発行したユーザーの名前（USERNAME 列）。
- REDO レコードの生成に使用された SQL 文と等価の（ただし、同一とはかぎらない）SQL 文を示すように再構成された SQL 文（SQL_REDO 列）。SQL_REDO 列の文にパスワードが含まれている場合、そのパスワードは暗号化されています。
- 変更を元に戻すために必要な SQL 文を示すように再構成された SQL 文（SQL_UNDO 列）。DDL 文に対応する SQL_UNDO 列は、常に NULL です。同様に、一部のデータ型やロールバックされた操作の場合は、SQL_UNDO 列が NULL になることがあります。

REDO ログには、表とそれに関連する列を識別するために内部的に生成された数値識別子が含まれています。LogMiner では、SQL 文を再構成するために、内部識別子からユーザー定義名へのマッピングを認識する必要があります。このマッピング情報は、データベースのデータ・ディクショナリに格納されています。LogMiner には、データ・ディクショナリを抽出するプロシージャ（DBMS_LOGMNR_D.BUILD）が用意されています。

> **関連項目：** DBMS_LOGMNR_D.BUILD プロシージャの詳細は、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

次の項では、REDO ログとディクショナリ・ファイルの詳細について説明します。

REDO ログとディクショナリ・ファイル

LogMiner を使用する前に、LogMiner が REDO ログとディクショナリ・ファイルをどのように操作するのかについて理解しておくことが重要です。これは、操作の正確な結果を取得して、システム・リソースの使用を計画する際に役立ちます。この項で説明する概念は、次のとおりです。

- REDO ログ
- ディクショナリ・オプション
- DDL 文の追跡

REDO ログ

LogMiner を実行するときは、分析する REDO ログの名前を指定します。LogMiner はこれらの REDO ログから情報を取得し、V$LOGMNR_CONTENTS ビューを通じて結果を戻します。REDO ログに必要な値情報が確実に格納されるように、少なくとも最低限のサプリメンタル・ロギングを使用可能にする必要があります。9-20 ページの「サプリメンタル・ロギング」を参照してください。

その後、他のビューと同様に SQL を使用して、V$LOGMNR_CONTENTS を問合せできます。V$LOGMNR_CONTENTS ビューに対して選択操作を実行すると、そのたびに REDO ログが順次読み込まれます。

REDO ログについては、常に次の点に注意してください。

- REDO ログは、必ず Oracle8 以上のデータベースのものである必要があります。ただし、リリース 1（9.0.1）で導入された LogMiner の一部の機能は、Oracle9*i* 以上のデータベースで生成された REDO ログでのみ動作します。9-11 ページの「制限事項」を参照してください。
- リリース 2（9.2）では LOB および LONG データ型のサポート機能を使用できますが、リリース 2（9.2）の Oracle データベースで生成された REDO ログの場合にかぎられます。
- REDO ログでは、LogMiner を実行しているデータベースのキャラクタ・セットと互換性のあるデータベース・キャラクタ・セットを使用する必要があります。
- 一般に、REDO ログの分析には、REDO ログを生成したのと同じデータベースから生成されたディクショナリが必要です。
- フラット・ファイル形式のディクショナリまたは REDO ログに格納されているディクショナリを使用する場合は、LogMiner を実行しているデータベースまたは他のデータベースの REDO ログを分析できます。
- オンライン・カタログを LogMiner ディクショナリとして使用する場合は、LogMiner を実行しているデータベースの REDO ログのみを分析できます。

- LogMiner は、分析する REDO ログを生成したのと同じハードウェア・プラットフォーム上で実行する必要があります。ただし、必ずしも同じシステムで実行する必要はありません。
- LogMiner を実行するときは、正しい REDO ログを指定することが重要です。必要なデータの一部を含む REDO ログを省略すると、V$LOGMNR_CONTENTS ビューの問合せ時に得られる結果が不正確になります。

現行の LogMiner セッションで分析する REDO ログを判断する際は、V$LOGMNR_LOGS ビューを参照できます。このビューには各 REDO ログに対応する 1 行が含まれています。

関連項目： 9-25 ページ 「分析する REDO ログの指定」

ディクショナリ・オプション

LogMiner では、REDO ログの内容を完全に変換するために、データベース・ディクショナリへのアクセスが必要です。

LogMiner はディクショナリを使用して、内部オブジェクト識別子とデータ型をオブジェクト名と外部データ形式に変換します。ディクショナリがない場合、LogMiner は内部オブジェクト ID を返し、データを 16 進バイトとして表現します。

たとえば、次の SQL 文を考えます。

```
INSERT INTO emp(name, salary) VALUES ('John Doe', 50000);
```

この場合、LogMiner では次のように表示されます。

```
insert into Object#2581(col#1, col#2) values (hextoraw('4a6f686e20446f65'),
hextoraw('c306'));"
```

LogMiner のディクショナリ・ファイルには、作成元のデータベースと作成された時間を識別する情報が含まれます。この情報を使用して、ディクショナリは選択した REDO ログと比較検証され、LogMiner の内部ディクショナリと REDO ログの不一致が自動的に検出されます。

ディクショナリ・ファイルは、分析する REDO ログと同じデータベース・キャラクタ・セットを持ち、同じデータベースから作成されたものである必要があります。ただし、ディクショナリが抽出されると、ソース・データベースに接続しなくても、そのデータベースの REDO ログを別のデータベース・インスタンスにマイニングできます。

また、ディクショナリ・ファイルを抽出すると、現行のデータ・ディクショナリに最新の表定義しか含まれていない場合に発生する問題を回避できます。たとえば、検索対象の表がいずれかの時点で削除された場合、現行のディクショナリにはその表の参照が含まれないことになります。

LogMiner では、ソース・ディクショナリに関して次の 3 つのオプションがあります。

- フラット・ファイルへのディクショナリの抽出
- REDO ログへのディクショナリの抽出
- オンライン・カタログの使用

フラット・ファイルへのディクショナリの抽出

ディクショナリがフラット・ファイル形式の場合は、REDO ログ形式の場合よりも使用するシステム・リソースは少なくなります。古い REDO ログでも正しく分析できるように、定期的にディクショナリ抽出のバックアップを作成することをお薦めします。

データベースのディクショナリ情報をフラット・ファイルに抽出するには、STORE_IN_FLAT_FILE オプションを指定して DBMS_LOGMNR_D.BUILD プロシージャを使用します。

ディクショナリの作成中に DDL 操作が行われないようにしてください。

次の手順では、ディクショナリをフラット・ファイルに抽出する方法を説明します（Oracle8 を使用している場合に従う必要のある追加の手順を含んでいます）。手順 1 ～ 4 は準備手順です。この準備手順を一度行うと、その後は必要な回数だけディクショナリをフラット・ファイルに抽出できます。

1. DBMS_LOGMNR_D.BUILD プロシージャでは、ディクショナリ・ファイルを配置することのできるディレクトリへのアクセスが必要です。通常、PL/SQL プロシージャはユーザー・ディレクトリにアクセスしないため、DBMS_LOGMNR_D.BUILD プロシージャで使用するディレクトリを指定する必要があります。これを指定しないと、プロシージャは失敗します。ディレクトリを指定するには、init.ora ファイルに初期化パラメータ UTL_FILE_DIR を設定します。

   関連項目： init.ora ファイルの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

   たとえば、ディクショナリ・ファイルを配置するディレクトリとして /oracle/database を使用するように UTL_FILE_DIR を設定するには、init.ora ファイルに次のように入力します。

   ```
   UTL_FILE_DIR = /oracle/database
   ```

   init.ora ファイルの変更を有効にするために、必ずデータベースをいったん停止して再起動してください。

2. **Oracle8 の場合のみ。それ以外の場合は、次の手順に進んでください。** オペレーティング・システムのコピー・コマンドを使用して、Oracle8*i* データベースの $ORACLE_HOME/rdbms/admin ディレクトリにある dbmslmd.sql スクリプトを、Oracle8 データベースの同じディレクトリにコピーします。たとえば、次のように入力します。

   ```
   % cp /8.1/oracle/rdbms/admin/dbmslmd.sql /8.0/oracle/rdbms/admin/dbmslmd.sql
   ```

3. データベースがクローズしている場合は、SQL*Plus を使用してマウントしてから、REDO ログを分析するデータベースをオープンします。たとえば、STARTUP コマンドを入力し、データベースをマウントしてオープンします。

```
SQL> STARTUP
```

4. **Oracle8 の場合のみ。それ以外の場合は、次の手順に進んでください。**コピーした dbmslmd.sql スクリプトを Oracle8 上で実行し、DBMS_LOGMNR_D パッケージをインストールします。たとえば、次のように入力します。

```
@dbmslmd.sql
```

スクリプトへの完全なパスの入力が必要な場合もあります。

5. PL/SQL プロシージャ DBMS_LOGMNR_D.BUILD を実行します。ディクショナリのファイル名とファイルのディレクトリ・パス名を指定します。このプロシージャによってディクショナリ・ファイルが作成されます。たとえば、次のように入力して、/oracle/database にファイル dictionary.ora を作成します。

```
SQL> EXECUTE DBMS_LOGMNR_D.BUILD('dictionary.ora', -
  2 '/oracle/database/', -
  3 OPTIONS => DBMS_LOGMNR_D.STORE_IN_FLAT_FILE);
```

STORE_IN_FLAT_FILE オプションを指定せずに、ファイル名と位置のみを指定することもできます。結果はどちらも同じです。

REDO ログへのディクショナリの抽出

ディクショナリを REDO ログに抽出するには、データベースをオープンし、ARCHIVELOG モードでアーカイブを使用可能にする必要があります。ディクショナリを REDO ログ・ストリームに抽出している間は、DDL 文を実行できません。したがって、REDO ログに抽出されたディクショナリのスナップショットには一貫性のあることが保証されますが、フラット・ファイルに抽出されたディクショナリには、この保証がありません。

データベースのディクショナリ情報を REDO ログに抽出するには、STORE_IN_REDO_FILES オプションを指定して DBMS_LOGMNR_D.BUILD プロシージャを使用します。ファイル名や位置は指定しないでください。

```
SQL> EXECUTE DBMS_LOGMNR_D.BUILD ( -
  2 OPTIONS=>DBMS_LOGMNR_D.STORE_IN_REDO_LOGS);
```

REDO ログに必要な値情報が確実に格納されるように、少なくとも最低限のサプリメンタル・ロギングを使用可能にする必要があります。9-20 ページの「サプリメンタル・ロギング」を参照してください。

関連項目： ARCHIVELOG モードの詳細は、『Oracle9*i* Recovery Manager ユーザーズ・ガイド』を参照してください。

ディクショナリを REDO ログに抽出するプロセスでは、データベース・リソースを消費します。しかし、抽出をオフピーク時に制限した場合、これが問題になることはなく、フラット・ファイルよりも高速に抽出できます。ディクショナリのサイズによっては、複数の REDO ログにディクショナリが格納されることがあります。関連する REDO ログがアーカイブされている場合は、抽出されたディクショナリの先頭と末尾を含む REDO ログを検索できます。そのためには、次のように V$ARCHIVED_LOG ビューを問い合せます。

```
SQL> SELECT NAME FROM V$ARCHIVED_LOG WHERE DICTIONARY_BEGIN='YES';
SQL> SELECT NAME FROM V$ARCHIVED_LOG WHERE DICTIONARY_END='YES';
```

先頭と末尾の REDO ログの名前と、両者間に含まれる他のログは、LogMiner セッションの開始準備中に ADD_LOGFILE プロシージャで指定します。

情報を保存して後で使用可能にしておくために、定期的に REDO ログのバックアップを作成することをお薦めします。データベースを適切に管理していれば、アーカイブ REDO ログのバックアップとリストアを行うためのプロセスがすでに適切に実行されているので、他の手順は必要ありません。繰り返しますが、処理にかかる時間を考慮して、この作業をオフピーク時間帯に実行することをお薦めします。

オンライン・カタログの使用

現在データベースで使用中のディクショナリを LogMiner で直接使用するには、LogMiner の起動時に次のようにディクショナリ・ソースとしてオンライン・カタログを指定します。

```
SQL> EXECUTE DBMS_LOGMNR.START_LOGMNR(OPTIONS => -
  2 DBMS_LOGMNR.DICT_FROM_ONLINE_CATALOG);
```

オンライン・カタログを使用すると、ディクショナリをフラット・ファイルや REDO ログに抽出する必要がありません。オンライン・カタログを使用すると、オンライン REDO ログを分析するのみでなく、アーカイブ REDO ログの生成時と同じシステムにアクセスしている場合はそれも分析できます。

オンライン・カタログには、データベースに関する最新の情報が含まれており、最も短時間で分析を開始できます。重要な表を変更する DDL 操作はまれであるため、通常、オンライン・カタログには分析に必要な情報が含まれています。

ただし、オンライン・カタログでできる操作は、最新バージョンの表に対して実行される SQL 文の再構成のみであることに注意してください。表に変更があると、オンライン・カタログにはその表の以前のバージョンは反映されなくなります。つまり、LogMiner では、以前のバージョンの表に対して実行された SQL 文は再構成できません。かわりに、LogMiner では次のように、実行可能でない SQL が SQL_REDO 列に（16 進から RAW への 2 進値の形式を含め）生成されます。

```
insert into Object#2581(col#1, col#2) values (hextoraw('4a6f686e20446f65'),
hextoraw('c306'));"
```

オンライン・カタログ・オプションを使用するには、データベースをオープンする必要があります。

オンライン・カタログ・オプションは DDL_DICT_TRACKING オプションと同時に指定したときは無効です。

DDL 文の追跡

LogMiner は、起動時に指定したソース・ディクショナリ（フラット・ファイル・ディクショナリ、REDO ログ内のディクショナリまたはオンライン・カタログ）から独自の内部ディクショナリを自動的に作成します。

ソース・ディクショナリがフラット・ファイル・ディクショナリまたは REDO ログ内のディクショナリである場合は、DDL_DICT_TRACKING オプションを使用して、DDL 文を追跡するように LogMiner に指示できます。DDL 文の追跡は、デフォルトで使用禁止になっています。使用可能にするには、LogMiner の起動時に OPTIONS パラメータを使用して DDL_DICT_TRACKING を指定します。次に例を示します。

```
SQL> EXECUTE DBMS_LOGMNR.START_LOGMNR(OPTIONS => -
  2 DBMS_LOGMNR.DDL_DICT_TRACKING);
```

このオプションを設定すると、LogMiner は REDO ログ内にあるすべての DDL 文をその内部ディクショナリに適用します。たとえば、ユーザー SYS により実行された DDL 文をすべて表示するには、次の問合せを発行します。

```
SQL> SELECT USERNAME, SQL_REDO
  2 FROM V$LOGMNR_CONTENTS
  3 WHERE USERNAME = 'SYS' AND OEPRATION = 'DDL';
```

次のような情報が戻されますが、画面に表示される実際の情報とその体裁は異なります。

```
USERNAME   SQL_REDO
SYS        ALTER TABLE SCOTT.ADDRESS ADD CODE NUMBER;
SYS        CREATE USER KATHY IDENTIFIED BY VALUES 'E4C8B920449B4C32' DEFAULT
           TABLESPACE TS1;
```

DDL_DICT_TRACKING オプションを使用する場合は、次の点に注意してください。

- DDL_DICT_TRACKING オプションは DICT_FROM_ONLINE_CATALOG オプションと同時に指定したときは無効です。
- DDL_DICT_TRACKING オプションを指定するには、データベースをオープンする必要があります。

DDL 文の追跡機能は、スキーマの変化を監視するために役立ちます。これは、（列の追加や削除などの DDL 操作のために）表の論理構造を変更する SQL 文を再構成できるためです。また、ディクショナリの抽出後に作成された新しい表に対して実行したデータ操作言語（DML）の操作も表示できます。

注意： DDL 文の追跡機能が使用禁止になっていると、DDL イベントが発生した場合に LogMiner では一部の REDO データが 16 進バイトとして戻されるため、通常はこの機能を使用可能にすることをお薦めします。また、使用可能にしないと、メタデータ・バージョンの不一致が発生する場合もあります。

LogMiner ではデータベースのメタデータにバージョンが自動的に割り当てられるため、内部ディクショナリと REDO ログに不一致があると検出されて通知されます。

注意： LogMiner の内部ディクショナリは、フラット・ファイルまたは REDO ログに格納されている LogMiner ディクショナリとは異なることを理解する必要があります。LogMiner は、それ自体の内部ディクショナリは更新しますが、フラット・ファイルまたは REDO ログに格納されているディクショナリは更新しません。

LogMiner に関する推奨事項と制限事項

LogMiner を使用する際には、次の推奨事項と制限事項に注意してください。

推奨事項

LogMiner の使用時には、次の点を考慮してください。

- すべてのデータベースでは、LogMiner 表に代替表領域を使用する必要があります。デフォルトでは、すべての LogMiner 表は `SYSTEM` 表領域を使用するように作成されます。`DBMS_LOGMNR_D.SET_TABLESPACE` ルーチンを使用すると、すべての LogMiner 表が代替表領域に再作成されます。たとえば、次の文では、すべての LogMiner 表が `logmnrts$` 表領域を使用するように再作成されます。

  ```
  SQL> EXECUTE DBMS_LOGMNR_D.SET_TABLESPACE('logmnrts$');
  ```

  関連項目： `DBMS_LOGMNR_D.SET_TABLESPACE` ルーチンの詳細は、『Oracle9*i* 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。

制限事項

LogMiner の使用時には、次の制限が適用されます。

- 次の操作対象はサポートされていません。
  - 単純な抽象データ型とネストした抽象データ型（ADT）
  - コレクション（ネストした表と VARRAY）
  - オブジェクト参照
  - 索引構成表（IOT）
  - クラスタ・キーを持つ表の CREATE TABLE AS SELECT
- LogMiner は Oracle8*i* 以上のデータベースでしか動作しませんが、Oracle8 データベースの REDO ログの分析に使用できます。ただし、LogMiner で REDO ログから取り出すことができる情報は、使用中のデータベースではなくログのバージョンに応じて異なります。たとえば、サプリメンタル・ロギングが使用可能になっている場合は、Oracle9*i* の REDO ログを引数として使用し、追加情報を取得できます。これにより、LogMiner の機能を最大限まで活用できます。旧リリースの Oracle で作成された REDO ログには追加データがないため、LogMiner でサポートされる操作とデータ型に制限を伴う場合があります。

  たとえば、次の機能を使用するには、サプリメンタル・ロギングをオンにする必要があります。Oracle9*i* リリース 1（9.0.1）では、サプリメンタル・ロギングが常にオンになっていたことに注意してください（リリース 1（9.0.1）より前のリリースでは、常に使用可能ではありませんでした）。しかし、リリース 2（9.2）では、サプリメンタル・ロギングは、明示的にオンにしないと使用可能になりません。

  - 索引クラスタ、連鎖行および移行行のサポート（連鎖行の場合は、データベースに設定されている互換性レベルに関係なく、サプリメンタル・ロギングが必須です）。
  - ダイレクト・パス・インサートのサポート（ARCHIVELOG モードを使用可能にする必要があります）。
  - REDO ログへのデータ・ディクショナリの抽出。
  - DDL 文の追跡。
  - 更新用の主キー情報を含む SQL_REDO および SQL_UNDO の生成。
  - LONG および LOB データ型のサポート（サプリメンタル・ロギングが使用可能になっている場合のみ）。

    関連項目： 9-20 ページ「サプリメンタル・ロギング」

戻されるデータのフィルタ処理

LogMiner では、大量の情報を取り扱う可能性があります。V$LOGMNR_CONTENTS ビューに戻される情報と戻されるときのスピードを制限するには、複数の方法があります。これらのオプションは、LogMiner の起動時に指定します。

- コミット済みトランザクションのみの表示
- REDO 破損部分のスキップ
- 時間指定によるデータのフィルタ処理
- SCN 指定によるデータのフィルタ処理

コミット済みトランザクションのみの表示

COMMITTED_DATA_ONLY オプションを使用すると、コミット済みトランザクションに属する行のみが V$LOGMNR_CONTENTS ビューに表示されます。これにより、ロールバックされたトランザクション、処理中のトランザクションおよび内部操作を除外できます。

このオプションを使用可能にするには、LogMiner の起動時に次のように指定します。

```
SQL> EXECUTE DBMS_LOGMNR.START_LOGMNR(OPTIONS => -
  2 DBMS_LOGMNR.COMMITTED_DATA_ONLY);
```

COMMITTED_DATA_ONLY オプションを指定すると、LogMiner では、同じトランザクションに属するすべての DML 操作がグループ化されます。トランザクションは、コミットされた順に戻されます。

分析中の REDO ログ内に長時間実行のトランザクションがある場合は、このオプションを使用すると「メモリー不足」エラーが発生することがあります。

デフォルトでは、すべてのトランザクションに対応する行が、REDO ログ内で発生する順に返されます。

たとえば、COMMITTED_DATA_ONLY を指定せずに LogMiner を起動し、次の問合せを実行するとします。

```
SQL> SELECT (XIDUSN || '.' || XIDSLT || '.' || XIDSQN) AS XID,
  2 USERNAME AS USER,
  3 SQL_REDO AS SQL_REDO
  4 FROM V$LOGMNR_CONTENTS;
```

出力は次のようになります。コミット済みトランザクションとコミットされていないトランザクションの両方が戻され、各トランザクションからの行が混在した状態で表示されます。

```
XID       USER      SQL_REDO
1.5.123   SCOTT     SET TRANSACTION READ WRITE;
1.5.123   SCOTT     INSERT INTO "SCOTT"."EMP"("EMPNO","ENAME")
                    VALUES (8782, 'Frost');
1.6.124   KATHY     SET TRANSACTION READ WRITE;
1.6.124   KATHY     INSERT INTO "SCOTT"."CUSTOMER"("ID","NAME","PHONE_DAY")
                    VALUES (8839, 'Cummings', '415-321-1234');
1.6.124   KATHY     INSERT INTO "SCOTT"."CUSTOMER"("ID","NAME","PHONE_DAY")
                    VALUES (7934, 'Yeats', '033-334-1234');
1.5.123   SCOTT     INSERT INTO "SCOTT"."EMP" ("EMPNO","ENAME")
                    VALUES (8566, 'Browning');
1.6.124   KATHY     COMMIT;
1.7.234   GOUTAM    SET TRANSACTION READ WRITE;
1.5.123   SCOTT     COMMIT;
1.7.234   GOUTAM    INSERT INTO "SCOTT"."CUSTOMER"("ID","NAME","PHONE_DAY")
                    VALUES (8499, 'Emerson', '202-334-1234');
```

今度は COMMITTED_DATA_ONLY オプションを指定して LogMiner を起動するとします。前述の問合せを再び実行すると、出力は次のようになります。

```
1.6.124   KATHY     SET TRANSACTION READ WRITE;
1.6.124   KATHY     INSERT INTO "SCOTT"."CUSTOMER"("ID","NAME","PHONE_DAY")
                    VALUES (8839, 'Cummings', '415-321-1234');
1.6.124   KATHY     INSERT INTO "SCOTT"."CUSTOMER"("ID","NAME","PHONE_DAY")
                    VALUES (7934, 'Yeats', '033-334-1234');
1.6.124   KATHY     COMMIT;
1.5.123   SCOTT     SET TRANSACTION READ WRITE;
1.5.123   SCOTT     INSERT INTO "SCOTT"."EMP" ("EMPNO","ENAME")
                    VALUES (8566, 'Browning');
1.5.123   SCOTT     INSERT INTO "SCOTT"."EMP"("EMPNO","ENAME")
                    VALUES (8782, 'Frost');
1.5.123   SCOTT     COMMIT;
```

1.6.124 トランザクションは 1.5.123 トランザクションより前にコミットされるため、1.6.124 トランザクション全体が最初に戻されます。これは、1.5.123 トランザクションが 1.6.124 トランザクションより前に開始された場合も同じです。1.7.234 トランザクションは、コミットが発行されていないため、戻されません。

REDO 破損部分のスキップ

SKIP_CORRUPTION オプションを使用すると、REDO ログ内の破損部分は V$LOGMNR_CONTENTS ビューの選択操作中にスキップされます。破損後に取得された行には、「ログ・ファイルの破損が検出された」というメッセージ付きのフラグが設定されます。また、破損が検出されたすべての REDO レコードについて、スキップしたブロック数を示す情報行が返されます。

デフォルトでは、REDO ログ内で最初に破損が検出された時点で選択操作は終了します。

このオプションを使用可能にするには、LogMiner の起動時に次のように指定します。

```
SQL> EXECUTE DBMS_LOGMNR.START_LOGMNR(OPTIONS => -
  2 DBMS_LOGMNR.SKIP_CORRUPTION);
```

時間指定によるデータのフィルタ処理

時間を指定してデータをフィルタ処理するには、STARTTIME および ENDTIME パラメータを設定します。この場合は日付値が必要です。次の例のように、TO_DATE ファンクションを使用して、日付と時刻を指定します。

```
SQL> EXECUTE DBMS_LOGMNR.START_LOGMNR( -
  2 DICTFILENAME => '/oracle/dictionary.ora', -
  3 STARTTIME => TO_DATE('01-Jan-1998 08:30:00', 'DD-MON-YYYY HH:MI:SS'), -
  4 ENDTIME => TO_DATE('01-Jan-1998 08:45:00', 'DD-MON-YYYY HH:MI:SS'));
```

STARTTIME および ENDTIME パラメータをまったく指定しない場合は、SELECT 文が発行されるたびに REDO ログ全体が最初から最後まで読み込まれます。

タイムスタンプを使用して REDO レコードの順序を推論しないでください。REDO レコードの順序を推論するには、SCN を使用します。

SCN 指定によるデータのフィルタ処理

SCN（システム変更番号）を指定してデータをフィルタ処理するには、次の例のように STARTSCN および ENDSCN パラメータを使用します。

```
SQL> EXECUTE DBMS_LOGMNR.START_LOGMNR( -
  2 DICTFILENAME => '/oracle/dictionary.ora', -
  3 STARTSCN => 100, -
  4 ENDSCN => 150);
```

STARTSCN、ENDSCN、STARTTIME および ENDTIME パラメータをすべて指定した場合は、STARTTIME および ENDTIME パラメータが上書きされます。

STARTSCN および ENDSCN パラメータをまったく指定しない場合は、SELECT 文が発行されるたびに REDO ログ全体が最初から最後まで読み込まれます。

LogMiner 情報へのアクセス

LogMiner 情報は、次のビューに含まれています。これらは、他のビューと同様に SQL を使用して問合せできます。

- `V$LOGMNR_CONTENTS`

  ユーザーおよび表情報に加えられた変更を表示します。

- `V$LOGMNR_DICTIONARY`

  ディクショナリが `STORE_IN_FLAT_FILE` オプションを使用して作成された場合に、LogMiner のディクショナリ・ファイルに関する情報を表示します。表示される情報には、データベース名とステータス情報が含まれます。

- `V$LOGMNR_LOGS`

  指定された REDO ログに関する情報を表示します。各行がそれぞれ 1 つの REDO ログに対応します。

- `V$LOGMNR_PARAMETERS`

  LogMiner のオプション・パラメータに関する情報を表示します。これには、開始および終了システム変更番号（SCN）、開始時刻、終了時刻などが含まれます。

  関連項目： これらのビューの内容に関する詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

これ以降は、LogMiner 情報へのアクセスに関する次の内容について説明します。

- V$LOGMNR_CONTENTS の問合せ
- REDO ログからの実際のデータ値の抽出

V$LOGMNR_CONTENTS の問合せ

LogMiner の出力は V$LOGMNR_CONTENTS ビューに格納されます。LogMiner の起動後、コマンドラインで SQL 文を発行し、V$LOGMNR_CONTENTS に含まれるデータを問合せできます。

V$LOGMNR_CONTENTS ビューに対して SQL の選択操作を実行すると、REDO ログが順次読み込まれます。REDO ログから変換された情報は、V$LOGMNR_CONTENTS ビュー内の行として戻されます。この処理は、起動時に指定したフィルタ条件が満たされる間、または REDO ログの最後に達するまで続きます。

LogMiner は、COMMITTED_DATA_ONLY オプションを使用してコミット済みのトランザクションのみを取得することを指定しないかぎり、すべての行を SCN 順に返します。SCN 順序は、通常メディア・リカバリにおいて適用される順序です。

たとえば、ユーザー Ron により scott.orders 表に対して実行されたすべての削除操作に関する情報を検索する必要があるとします。次のような問合せを発行できます。

```
SQL> SELECT OPERATION, SQL_REDO, SQL_UNDO
  2 FROM V$LOGMNR_CONTENTS
  3 WHERE SEG_OWNER = 'SCOTT' AND SEG_NAME = 'ORDERS' AND
  4 OPERATION = 'DELETE' AND USERNAME = 'RON';
```

次の出力が生成されます。実際に表示される書式は、次の例とは異なる場合があります。

```
OPERATION   SQL_REDO                        SQL_UNDO

DELETE      delete from "SCOTT"."ORDERS"    insert into "SCOTT"."ORDERS"
                where "ORDER_NO" = 2 and        ("ORDER_NO", "QTY", "EXPR_SHIP")
                "QTY" = 3 and                    values(2,3,'Y');
                "EXPR_SHIP" = 'Y' and
                ROWID = 'AAABM8AABAAALm/AAA'
DELETE          delete from "SCOTT"."ORDERS"     insert into "SCOTT"."ORDERS"
                where "ORDER_NO" = 4 and         ("ORDER_NO","QTY","EXPR_SHIP")
                "QTY" = 7 and                    values(4,7,'Y');
                "EXPR_SHIP" = 'Y' and
                ROWID = 'AAABM8AABAAALm/AAC';
```

この出力は、Ron が scott.orders 表から 2 つの行を削除したことを示しています。再構成された SQL 文は実際に Ron が発行した文と等価ですが、同一であるとはかぎりません。これは、元の WHERE 句は REDO ログに書き込まれないので、LogMiner では削除（または更新、挿入）された個々の行しか表示できないためです。

したがって、1 つの DELETE 文によって両方の行が削除された場合にも、それは V$LOGMNR_CONTENTS の出力には反映されません。そのため、実際の DELETE 文は、DELETE FROM SCOTT.ORDERS WHERE EXPR_SHIP = 'Y' または DELETE FROM SCOTT.ORDERS WHERE QTY < 8 の可能性があります。

再構成された SQL 文の実行

デフォルトでは、SQL_REDO および SQL_UNDO 文の末尾にはセミコロンが 1 つ付いています。再構成された文の使用方法に応じて、セミコロンを含めても、含めなくてもかまいません。セミコロンを抑制するには、LogMiner の起動時に DBMS_LOGMNR.NO_SQL_DELIMITER オプションを指定します。

V$LOGMNR_CONTENTS の STATUS フィールドに dbms_logmnr.invalid_sql が含まれている場合は、SQL を実行できないことに注意してください。

戻されるデータの書式

問合せの結果、再構成される SQL 文を含む多数の列が戻される場合があります。このような結果では情報が読みづらくなります。LogMiner には、この問題に対処するための DBMS_LOGMNR.PRINT_PRETTY_SQL オプションが用意されています。PRINT_PRETTY_SQL オプションを使用すると、再構成された SQL 文が次のように書式化されて読みやすくなります。

```
insert into "SCOTT"."EMP" values
  "EMPNO": 5505,
  "ENAME": "Parker",
  "SAL":   9000
  "DEPTNO":  NULL;
update "SCOTT"."EMP"
  set
  "EMPNO" = 5505 and
  "SAL"   = 9000
  where
  "EMPNO" = 5505 and
  "SAL"     = 9000 and
  "ROWID"   = AABBCEXFGHA;
```

PRINT_PRETTY_SQL オプションが使用可能になっているときに再構成された SQL 文は、標準の SQL 構文を使用していないため実行できません。

REDO ログからの実際のデータ値の抽出

LogMiner では、問合せを実際のデータ値に基づいて実行できます。たとえば、scott.emp に対して sal の値をある一定の金額よりも増やした更新をすべて表示する問合せを実行できます。このようなデータは、システムの動作分析や監査作業の実行に使用できます。

LogMiner による REDO ログからのデータ抽出は、2 つのマイニング・ファンクション DBMS_LOGMNR.MINE_VALUE および DBMS_LOGMNR.COLUMN_PRESENT を使用して実行されます。これらのファンクションは、DBMS_LOGMNR パッケージの一部です。これらのマイニング・ファンクションのサポートは、V$LOGMNR_CONTENTS ビューの REDO_VALUE および UNDO_VALUE 列で提供されます。

次の例に、MINE_VALUE ファンクションを使用して、scott.emp に対して sal 列を元の値の 2 倍以上に増やした更新をすべて選択する方法を示します。

```
SQL> SELECT SQL_REDO FROM V$LOGMNR_CONTENTS
  2 WHERE
  3 SEG_NAME = 'emp' AND
  4 SEG_OWNER = 'SCOTT' AND
  5 OPERATION = 'UPDATE' AND
  6 DBMS_LOGMNR.MINE_VALUE(REDO_VALUE, 'SCOTT.EMP.SAL') >
  7 2*DBMS_LOGMNR.MINE_VALUE(UNDO_VALUE, 'SCOTT.EMP.SAL');
```

この例のように、MINE_VALUE ファンクションは 2 つの引数を取ります。最初の引数では、データの REDO（REDO_VALUE）部分と UNDO（UNDO_VALUE）部分のうち、どちらをマイニングするかを指定します。第 2 の引数は、マイニングする列の完全修飾名（この場合は SCOTT.EMP.SAL）を指定する文字列です。MINE_VALUE ファンクションは常に、元のデータ型に変換できる文字列を戻します。

MINE_VALUE ファンクションからの NULL の戻り

MINE_VALUE ファンクションが NULL 値を戻す場合は、次のいずれかを意味する可能性があります。

- 指定した列がデータの REDO 部分または UNDO 部分に存在しない場合。
- 指定した列が存在し、その値が NULL の場合。

この 2 つの場合を区別するために、DBMS_LOGMNR.COLUMN_PRESENT ファンクションを使用します。このファンクションは、データの REDO 部分または UNDO 部分に列が存在する場合は 1 を戻し、それ以外の場合は 0 を戻します。たとえば、sal 列の値が変更されたときの増分と、それに対応するトランザクション ID を検索するとします。次の問合せを発行できます。

```
SQL> SELECT
  2 (XIDUSN || '.' || XIDSLT || '.' || XIDSQN) AS XID,
  3 (DBMS_LOGMNR.MINE_VALUE(REDO_VALUE, 'SCOTT.EMP.SAL') -
  4 DBMS_LOGMNR.MINE_VALUE(UNDO_VALUE, 'SCOTT.EMP.SAL')) AS INCR_SAL
  5 FROM V$LOGMNR_CONTENTS
  6 WHERE
  7 DBMS_LOGMNR.COLUMN_PRESENT(REDO_VALUE, 'SCOTT.EMP.SAL') = 1 AND
  8 DBMS_LOGMNR.COLUMN_PRESENT(UNDO_VALUE, 'SCOTT.EMP.SAL') = 1 AND
  9 OPERATION = 'UPDATE';
```

MINE_VALUE および COLUMN_PRESENT ファンクションの使用規則

MINE_VALUE および COLUMN_PRESENT ファンクションには、次の使用規則が適用されます。

- 使用できるのは、LogMiner セッション内のみです。
- V$LOGMNR_CONTENTS ビューからの選択操作のコンテキスト内で起動する必要があります。
- LONG、LOB、ADT または COLLECTION データ型はサポートされません。
- 列の引数が DATE 型の場合、戻される文字列は現行セッションの日付書式に関係なく標準書式（DD-MON-YYYY HH24:MI:SS.SS）で書式化されます。

関連項目： DBMS_LOGMNR パッケージの詳細は、『Oracle9*i* 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。このパッケージには、MINE_VALUE および COLUMN_PRESENT ファンクションが含まれています。

サプリメンタル・ロギング

通常、REDO ログはインスタンス・リカバリとメディア・リカバリに使用されます。この種の操作に必要なデータは、REDO ログに自動的に記録されます。ただし、REDO ベースのアプリケーションでは、REDO ログに追加情報を記録することが必要になる場合があります。次に、サプリメンタル・データを必要とする状況の例を示します。

- 再構成された SQL 文を別のデータベースに適用する必要があるアプリケーションでは、UPDATE 文を LogMiner で使用される通常の方法である ROWID ではなく主キーで識別する必要があります。（デフォルトでは、キー自体が更新により変更されないかぎり、主キーは REDO ログに記録されません。）
- アプリケーションで行の変更を効率的に追跡するには、変更された列のみでなく行全体のビフォア・イメージをログに記録する必要があります。

Oracle データベースのデフォルト動作では、サプリメンタル・ロギングはまったく提供されません。これは、特定の機能がサポートされないことを意味します（9-11 ページの「制限事項」を参照）。LogMiner のサポート機能全体を使用するには、サプリメンタル・ロギングを使用可能にする必要があります。

最小限のサプリメンタル・ロギングを使用可能にして LogMiner を使用すると、REDO ログを生成するインスタンスのパフォーマンスにはほとんど影響しません。ただし、データベース単位のサプリメンタル・ロギングを使用可能にして LogMiner を使用する場合は、オーバーヘッドが大きく、パフォーマンスが大幅に低下します。

サプリメンタル・ロギングには、データベースのサプリメンタル・ロギングと表のサプリメンタル・ロギングという 2 つのタイプがあります。以降は、この 2 つのタイプについて個別に説明します。

データベースのサプリメンタル・ロギング

データベースのサプリメンタル・ロギングには、最小限のロギングと識別キーによるロギングという 2 つのタイプがあります。

最小限のサプリメンタル・ロギングでは、LogMiner で DML の変更に関連する REDO 操作を識別、グループ化およびマージするために必要な、最小限の情報が記録されます。この場合、LogMiner（および、LogMiner テクノロジに基づく製品）は、連鎖行とクラスタ化表などの各種記憶域配置をサポートできるだけの十分な情報が得られます。ほとんどの場合は、少なくとも最小限のサプリメンタル・ロギングを使用可能にする必要があります。そのためには、次の文を実行します。

```
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA
```

注意： リリース 1（9.0.1）の LogMiner では、最小限のサプリメンタル・ロギングがデフォルト動作でした。リリース 2（9.2）の場合、デフォルトではサプリメンタル・ロギングはオフになっており、明示的に使用可能にする必要があります。

識別キーによるロギングでは、すべての更新について、主キーまたは一意索引（主キーが存在しない場合）のデータベース全体のビフォア・イメージをロギングできます。このタイプのロギングを使用すると、アプリケーションでは更新された行を ROWID に再ソートするのではなく、論理的に識別できます。

識別キーによるロギングが必要になるのは、サプリメンタル・ロギングで得られたデータが、論理スタンバイなど、他のデータベースでの変更のソースになる場合です。

識別キーによるロギングを使用可能にするには、次の文を実行します。

```
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY, UNIQUE INDEX) COLUMNS;
```

この文を実行すると、データベース全体ですべての主キーの値が、変更があったかどうかに関係なくロギングされます。

表に主キーがなくても、NULL でない一意キー制約が 1 つ以上あると、更新対象となる行を識別する手段として、その 1 つがロギング用に任意に選択されます。

表に主キーも一意索引もない場合は、LONG および LOB を除くすべての列がサプリメンタル・ロギングの対象となります。したがって、サプリメンタル・ロギングを使用する場合は、すべてまたはほとんどの表に主キーまたは一意キーを定義しておくことをお薦めします。

注意： 識別キーによるロギングが使用可能になっているかどうかに関係なく、LogMiner で戻される SQL 文には常に ROWID 句が含まれます。再構成される SQL 文に RTRIM ファンクションと適切な引数を使用すると、ROWID 句を除外できます。

最小限のロギングまたは識別キーによるロギングを使用禁止にするには、次の文を実行します。

```
SQL> ALTER DATABASE DROP SUPPLEMENTAL LOG DATA;
```

識別キーによるロギングの使用方法

識別キーによるロギングを使用する場合は、次のことに注意してください。

- DELETE 文には行の識別に必要な列の値がすべて含まれているため、識別キーによるロギングは削除操作には不要です。
- 識別キーによるロギングを使用可能にしている場合に、データベースがオープンしていると、カーソル・キャッシュ内のすべての DML 文が無効になります。これは、キャッシュが再移入されるまでパフォーマンスに影響する可能性があります。

表のサプリメンタル・ロギング

表のサプリメンタル・ロギングでは、ログ・グループを使用してサプリメンタル情報がロギングされます。ログ・グループには、次の 2 つのタイプがあります。

- 無条件のログ・グループ － 表が更新されるたびに、その更新が指定の列に影響するかどうかに関係なく、指定した列のビフォア・イメージがロギングされます。このタイプは ALWAYS ログ・グループとも呼ばれます。
- 条件付きログ・グループ － ログ・グループ内の少なくとも 1 つの列が更新される場合にのみ、指定したすべての列のビフォア・イメージがロギングされます。

無条件のログ・グループ

無条件のログ・グループを使用するサプリメンタル・ロギングを使用可能にするには、次の例のように ALWAYS 句を使用します。

```
SQL> ALTER TABLE scott.emp
  2 ADD SUPPLEMENTAL LOG GROUP emp_parttime (empno, ename, deptno) ALWAYS;
```

この句により、列 empno、ename および deptno を含む scott.emp に、ログ・グループ emp_parttime が作成されます。これらの列は、scott.emp に対して UPDATE 文が実行されるたびに、更新の影響を受けるかどうかに関係なくロギングされます。更新のたびに行全体のイメージをロギングする必要がある場合は、表のすべての列を含むログ・グループを作成できます。

注意： LOB、LONG および ADT は、ログ・グループに含めることができません。

条件付きログ・グループ

条件付きログ・グループを使用するサプリメンタル・ロギングを使用可能にするには、次の例のように ALTER TABLE 文の ALWAYS 句を省略します。

```
SQL> ALTER TABLE scott.emp
  2 ADD SUPPLEMENTAL LOG GROUP emp_fulltime (empno, ename, deptno);
```

この文では、scott.emp にログ・グループ emp_fulltime が作成されます。前述の例と同様に、scott.emp は列 empno、ename および deptno で構成されています。ただし、ALWAYS 句が省略されているため、列のビフォア・イメージは少なくとも列の 1 つが更新される場合にのみロギングされます。

ログ・グループの使用方法

ログ・グループを使用する場合は、次のことに注意してください。

- 1 つの列が複数のログ・グループに属することができます。ただし、列のビフォア・イメージがロギングされるのは 1 度のみです。
- REDO ログには、列が属しているログ・グループや、列のビフォア・イメージがログ・グループのロギングと識別キーによるロギングのどちらでロギングされるかに関する情報は含まれません。
- 同じ列を条件付きと無条件の両方でロギングするように指定すると、その列は無条件でロギングされます。

標準的な LogMiner セッションにおける手順

この項では、標準的な LogMiner セッションにおける手順について説明します。各手順は、それぞれ独立したセクションで説明します。

1. 初期セットアップ・アクティビティの実行
2. ディクショナリの抽出（オンライン・カタログを使用する予定の場合以外）
3. 分析する REDO ログの指定
4. LogMiner セッションの開始
5. V$LOGMNR_CONTENTS の問合せ
6. LogMiner セッションの終了

LogMiner を実行するには、DBMS_LOGMNR PL/SQL パッケージを使用します。また、オンライン・カタログを使用するのではなくディクショナリを抽出するように選択する場合は、DBMS_LOGMNR_D パッケージを使用することもできます。

DBMS_LOGMNR パッケージには、REDO ログ名、フィルタ基準およびセッション特性を指定するためのインタフェースなど、LogMiner の初期化と実行に使用されるプロシージャが含まれています。DBMS_LOGMNR_D パッケージは、現行データベースのディクショナリ表を問い合せて LogMiner ディクショナリ・ファイルを作成します。

LogMiner パッケージは、SYS スキーマが所有しています。したがって、ユーザー SYS として接続していない場合は、コールの中に SYS を含める必要があります。次に例を示します。

```
EXECUTE SYS.DBMS_LOGMNR.END_LOGMNR
```

関連項目：

- これらの LogMiner パッケージの構文とパラメータの詳細は、『Oracle9*i* 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。
- PL/SQL プロシージャの実行の詳細は、『Oracle9*i* アプリケーション開発者ガイドー基礎編』を参照してください。

初期セットアップ・アクティビティの実行

これは、LogMiner を初めて使用する前に実行する必要のある初期セットアップ・アクティビティです。これらのアクティビティを実行するのは 1 度のみです。LogMiner を使用するたびに実行する必要はありません。

- 使用するタイプのサプリメンタル・ロギングを使用可能にします。少なくとも、次のように最小限のサプリメンタル・ロギングを使用可能にすることをお薦めします。

  ```
  SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA
  ```

  詳細は、9-20 ページの「サプリメンタル・ロギング」を参照してください。

- DBMS_LOGMNR_D.SET_TABLESPACE ルーチンを使用すると、すべての LogMiner 表が代替表領域に再作成されます。次に例を示します。

  ```
  SQL> EXECUTE DBMS_LOGMNR_D.SET_TABLESPACE('logmnrts$');
  ```

  詳細は、9-10 ページの「推奨事項」を参照してください。

ディクショナリの抽出

LogMiner を使用するには、次のいずれかを実行して、LogMiner にディクショナリを提供する必要があります。

- データベースのディクショナリ情報をフラット・ファイルに抽出する。9-6 ページの「フラット・ファイルへのディクショナリの抽出」を参照してください。
- データベースのディクショナリ情報を REDO ログに抽出する。9-7 ページの「REDO ログへのディクショナリの抽出」を参照してください。
- LogMiner の起動時に DICT_FROM_ONLINE_CATALOG オプションを使用して、オンライン・カタログの使用を指定する。9-8 ページの「オンライン・カタログの使用」を参照してください。

分析する REDO ログの指定

LogMiner を実行する前に、分析する REDO ログの名前を指定する必要があります。そのためには、後述の手順のように DBMS_LOGMNR.ADD_LOGFILE プロシージャを実行します。REDO ログは任意の順序で追加および削除できます。

注意： REDO ログを生成するのと同じインスタンスにマイニングする場合は、LogMiner の起動時にアーカイブ REDO ログを 1 つと CONTINUOUS_MINE オプションを指定します。9-26 ページの「連続的マイニング」を参照してください。

---

1. SQL*Plus を使用して Oracle インスタンスを起動します。データベースはマウントしてもしなくてもどちらでもかまいません。たとえば、次のように入力します。

```
SQL> STARTUP
```

2. REDO ログのリストを作成します。DBMS_LOGMNR.ADD_LOGFILE プロシージャの NEW オプションを指定して、これが新規リストの先頭であることを示します。たとえば、/oracle/logs/log1.f を指定するには、次のように入力します。

```
SQL> EXECUTE DBMS_LOGMNR.ADD_LOGFILE( -
  2 LOGFILENAME => '/oracle/logs/log1.f', -
  3 OPTIONS => DBMS_LOGMNR.NEW);
```

3. 必要に応じて、DBMS_LOGMNR.ADD_LOGFILE プロシージャの ADDFILE オプションを指定して、REDO ログをさらに追加します。たとえば、/oracle/logs/log2.f を追加するには、次のように入力します。

```
SQL> EXECUTE DBMS_LOGMNR.ADD_LOGFILE( -
  2 LOGFILENAME => '/oracle/logs/log2.f', -
  3 OPTIONS => DBMS_LOGMNR.ADDFILE);
```

さらに REDO ログを追加する場合、OPTIONS パラメータはオプションです。たとえば、単に次のように入力できます。

```
SQL> EXECUTE DBMS_LOGMNR.ADD_LOGFILE( -
  2 LOGFILENAME=>'/oracle/logs/log2.f');
```

4. 必要に応じて、DBMS_LOGMNR.ADD_LOGFILE プロシージャの REMOVEFILE オプションを指定して、REDO ログを削除します。たとえば、/oracle/logs/log2.f を削除するには、次のように入力します。

```
SQL> EXECUTE DBMS_LOGMNR.ADD_LOGFILE( -
  2 LOGFILENAME => '/oracle/logs/log2.f', -
  3 OPTIONS => DBMS_LOGMNR.REMOVEFILE);
```

連続的マイニング

連続的マイニング・オプションは、REDO ログを生成するのと同じインスタンスでマイニングする場合に役立ちます。連続的マイニング・オプションを使用する予定の場合は、LogMiner の起動前にアーカイブ REDO ログを 1 つ指定します。その後、LogMiner の起動時に DBMS_LOGMNR.CONTINUOUS_MINE オプションを指定します。このオプションにより LogMiner に対して、後続のアーカイブ REDO ログとオンライン・カタログを自動的に追加し、マイニングするように指示します。

注意： 連続的マイニングは、Real Application Clusters 環境では使用できません。

LogMiner セッションの開始

ディクショナリ・ファイルを作成し、分析する REDO ログの指定が完了した後、LogMiner セッションを開始できます。次の手順を実行します。

1. DBMS_LOGMNR.START_LOGMNR プロシージャを実行して、LogMiner を起動します。

   起動時はディクショナリ・オプションを指定することをお薦めします。ディクショナリ・オプションを指定しない場合、LogMiner は内部オブジェクト識別子とデータ型をオブジェクト名と外部データ形式に変換できません。したがって、内部オブジェクト ID を返し、データを 16 進バイトとして表現します。また、MINE_VALUE および COLUMN_PRESENT ファンクションは、ディクショナリを指定せずに使用することはできません。

   フラット・ファイル・ディクショナリの名前を指定する場合は、ディクショナリ・ファイルの完全修飾ファイル名を指定する必要があります。たとえば、/oracle/database/dictionary.ora を使用して LogMiner を起動するには、次のコマンドを発行します。

   ```
   SQL> EXECUTE DBMS_LOGMNR.START_LOGMNR( -
     2 DICTFILENAME =>'/oracle/database/dictionary.ora');
   ```

   フラット・ファイル・ディクショナリ名を指定しない場合は、OPTIONS パラメータで DICT_FROM_REDO_LOGS または DICT_FROM_ONLINE_CATALOG オプションを指定します。

   DICT_FROM_REDO_LOGS を指定した場合、LogMiner は、DBMS_LOGMNR.ADD_LOGFILE プロシージャで指定した REDO ログ内でディクショナリを検索します。ディクショナリが格納されている REDO ログを判断するには、V$ARCHIVED_LOG ビューを参照します。例については、9-7 ページの「REDO ログへのディクショナリの抽出」を参照してください。

注意： LogMiner セッションの開始後に REDO ログを追加した場合は、LogMiner を再起動する必要があります。必要に応じて、新規の起動パラメータを指定できます。指定しない場合、LogMiner では前回のセッションに指定したパラメータが使用されます。

オンライン・カタログの使用方法の詳細は、9-8 ページの「オンライン・カタログの使用」を参照してください。

2. 必要に応じて、問合せを時間または SCN でフィルタ処理できます。9-14 ページの「時間指定によるデータのフィルタ処理」または 9-14 ページの「SCN 指定によるデータのフィルタ処理」を参照してください。

3. また、OPTIONS パラメータを使用して LogMiner セッションの特性を追加指定することもできます。たとえば、次のように、ディクショナリとしてオンライン・カタログを使用し、コミット済みトランザクションのみを V$LOGMNR_CONTENTS ビューに表示します。

```
SQL> EXECUTE DBMS_LOGMNR.START_LOGMNR(OPTIONS => -
  2 DBMS_LOGMNR.DICT_FROM_ONLINE_CATALOG + -
  3 DBMS_LOGMNR.COMMITTED_DATA_ONLY);
```

次のリストは、OPTIONS パラメータで指定できる LogMiner 設定と詳細情報の参照先をまとめたものです。

- DBMS_LOGMNR.DICT_FROM_ONLINE_CATALOG — 9-8 ページの「オンライン・カタログの使用」を参照してください。
- DBMS_LOGMNR.DICT_FROM_REDO_LOGS — このリストの手順 1 を参照してください。
- DBMS_LOGMNR.COMMITTED_DATA_ONLY — 9-12 ページの「コミット済みトランザクションのみの表示」を参照してください。
- DBMS_LOGMNR.SKIP_CORRUPTION — 9-14 ページの「REDO 破損部分のスキップ」を参照してください。
- DBMS_LOGMNR.DDL_DICT_TRACKING — 9-9 ページの「DDL 文の追跡」を参照してください。
- DBMS_LOGMNR.NEW、DBMS_LOGMNR.ADDFILE および DBMS_LOGMNR.REMOVEFILE — 9-25 ページの「分析する REDO ログの指定」を参照してください。
- DBMS_LOGMNR.NO_SQL_DELIMITER — 9-17 ページの「戻されるデータの書式」を参照してください。

- DBMS_LOGMNR.PRINT_PRETTY_SQL － 9-17 ページの「戻されるデータの書式」を参照してください。
- DBMS_LOGMNR.CONTINUOUS_MINE － 9-26 ページの「連続的マイニング」を参照してください。

DBMS_LOGMNR.START_LOGMNR プロシージャを複数回実行し、そのたびに異なるオプションを指定できます。この方法は、V$LOGMNR_CONTENTS の問合せから必要な結果を取得しておらず、異なるオプションを指定して LogMiner を再起動する必要がある場合などに役立ちます。すでに以前のセッションで追加した REDO ログを再び追加する必要はありません。

V$LOGMNR_CONTENTS の問合せ

この時点で LogMiner が起動し、V$LOGMNR_CONTENTS ビューに対して問合せを実行できます。この例は、9-16 ページの「V$LOGMNR_CONTENTS の問合せ」を参照してください。

LogMiner セッションの終了

LogMiner セッションを適切に終了するには、次のように DBMS_LOGMNR.END_LOGMNR プロシージャを使用します。

```
SQL> EXECUTE DBMS_LOGMNR.END_LOGMNR;
```

このプロシージャは、すべての REDO ログをクローズし、LogMiner によって割り当てられたすべてのデータベースとシステム・リソースを解放します。

このプロシージャを実行しない場合、LogMiner は起動元の Oracle セッションが終了しないかぎり、割り当てられたすべてのリソースをそのまま保持します。特に、DDL_DICT_TRACKING または DICT_FROM_REDO_LOGS のどちらかのオプションを使用した場合には、このプロシージャを使用して LogMiner を終了することが重要です。

LogMiner の使用例

ここでは、次のような LogMiner の使用例について説明します。

- 例 : LogMiner を使用した特定のユーザーによる変更の追跡
- 例 : LogMiner を使用した表アクセス統計の計算

例 : LogMiner を使用した特定のユーザーによる変更の追跡

この例は、ユーザーの 1 人である joedevo が特定の期間内にデータベースに対して行ったすべての変更を表示する方法を示しています。データベースに接続し、次の手順を実行します。

- 手順 1: ディクショナリ・ファイルの作成
- 手順 2: REDO ログの追加
- 手順 3: LogMiner の起動と検索範囲の制限
- 手順 4: V$LOGMNR_CONTENTS の問合せ

手順 1: ディクショナリ・ファイルの作成 LogMiner を使用して joedevo のデータを分析するには、joedevo が変更を行う前にディクショナリ・ファイルを作成するか、LogMiner の起動時にオンライン・カタログの使用を指定する必要があります。ディクショナリの作成例は、9-24 ページの「ディクショナリの抽出」を参照してください。

手順 2: REDO ログの追加 joedevo がデータベースに対してなんらかの変更を行っているとします。次のように、分析する REDO ログの名前を指定できます。

```
SQL> EXECUTE DBMS_LOGMNR.ADD_LOGFILE( -
  2 LOGFILENAME => 'log1orc1.ora', -
  3 OPTIONS => DBMS_LOGMNR.NEW);
```

必要に応じて、次のようにさらに REDO ログを追加します。

```
SQL> EXECUTE DBMS_LOGMNR.ADD_LOGFILE( -
  2 LOGFILENAME => 'log2orc1.ora', -
  3 OPTIONS => DBMS_LOGMNR.ADDFILE);
```

手順 3: LogMiner の起動と検索範囲の制限 LogMiner を起動し、検索範囲を指定の時間範囲に限定します。

```
SQL> EXECUTE DBMS_LOGMNR.START_LOGMNR( -
  2 DICTFILENAME => 'orcldict.ora', -
  3 STARTTIME => TO_DATE('01-Jan-1998 08:30:00', 'DD-MON-YYYY HH:MI:SS'), -
  4 ENDTIME => TO_DATE('01-Jan-1998 08:45:00', 'DD-MON-YYYY HH:MI:SS'));
```

手順 4: V$LOGMNR_CONTENTS の問合せ この時点で、V$LOGMNR_CONTENTS ビューは問合せのために使用可能です。ユーザー joedevo が salary 表に対して行ったすべての変更を検索します。次の SELECT 文を実行します。

```
SQL> SELECT SQL_REDO, SQL_UNDO FROM V$LOGMNR_CONTENTS
  2 WHERE USERNAME = 'joedevo' AND SEG_NAME = 'salary';
```

SQL_REDO および SQL_UNDO の両方の列について、2 つの行が返されます（データの表示形式は画面上では異なります）。joedevo が 2 つの操作を要求したことがわかりました。自分の過去の給与を削除し、前より高額の給与を新規に挿入しています。現在は、この操作の取消しに必要なデータを取得できます。

```
SQL_REDO                                SQL_UNDO
--------                                --------
delete * from SALARY                    insert into SALARY(NAME, EMPNO, SAL)
where EMPNO = 12345                     values ('JOEDEVO', 12345, 500)
and ROWID = 'AAABOOAABAAEPCABA';

insert into SALARY(NAME, EMPNO, SAL)    delete * from SALARY
values('JOEDEVO',12345, 2500)           where EMPNO = 12345
                                        and ROWID = 'AAABOOAABAAEPCABA';
2 rows selected
```

例 : LogMiner を使用した表アクセス統計の計算

この例では、直販データベースを管理しており、8 月の 2 週間の消費者売上につながった生産性を判断する場合を考えます。すでにディクショナリを作成し、検索に必要な REDO ログを追加しているとします（前述の例を参照）。次の手順を実行します。

1. LogMiner を起動して、時間範囲を指定します。

   ```
   SQL> EXECUTE DBMS_LOGMNR.START_LOGMNR( -
     2 STARTTIME => TO_DATE('07-Aug-1998 08:30:00', 'DD-MON-YYYY HH:MI:SS'), -
     3 ENDTIME => TO_DATE('21-Aug-1998 08:45:00', 'DD-MON-YYYY HH:MI:SS'), -
     4 DICTFILENAME => '/usr/local/dict.ora');
   ```

2. 次の例のように V$LOGMNR_CONTENTS ビューを問い合せて、指定した時間範囲内にどの表が変更されたのかを判断します。この問合せでは、慣習として表名に $ が含まれているシステム表を除外しています。

   ```
   SQL> SELECT SEG_OWNER, SEG_NAME, COUNT(*) AS Hits FROM
     2 V$LOGMNR_CONTENTS WHERE SEG_NAME NOT LIKE '%$' GROUP BY
     3 SEG_OWNER, SEG_NAME;
   ```

3. 次のデータが表示されます。（表示形式は異なる場合があります。）

```
SEG_OWNER          SEG_NAME          Hits
---------          --------          ----
CUST               ACCOUNT            384
SCOTT              EMP                 12
SYS                DONOR               12
UNIV               DONOR              234
UNIV               EXECDONOR          325
UNIV               MEGADONOR           32
```

`Hits` 列の値は、問合せで指定した 2 週間という期間中に、指定した表に対して挿入、削除または更新操作が実行された回数を示します。

10

ジョブ・キューの管理

この章では、ジョブ・キューを使用してユーザー・ジョブの定期的な実行をスケジュールする方法について説明します。この章の内容は、次のとおりです。

- ジョブの実行に使用されるプロセスの有効化
- ジョブ・キューの管理
- ジョブ・キューに関する情報の表示

ジョブの実行に使用されるプロセスの有効化

ジョブ・キューを使用すると、ルーチン（ジョブ）が定期的に実行されるようにスケジュールできます。ジョブをスケジュールするには、オラクル社が提供する DBMS_JOB パッケージを使用してジョブをジョブ・キューに送り、ジョブを実行する頻度を指定します。追加の機能を使用すると、以前に送ったジョブの変更、無効化または削除ができます。

ジョブ・キュー（J*nnn*）プロセスは、ジョブ・キュー内のジョブを実行します。これらのジョブ・キュー・プロセスは、インスタンスごとに、コーディネータ・ジョブ・キュー（CJQ0）バックグラウンド・プロセスによって動的に起動されます。コーディネータは、DBA_JOBS ビューに表示されているジョブから、実行準備が完了しているジョブを定期的に選択します。コーディネータはジョブを時間別に順序付けてから、J*nnn* プロセスを起動し、選択したジョブを実行します。各 J*nnn* プロセスは、選択されたジョブの 1 つを実行します。

JOB_QUEUE_PROCESSES 初期化パラメータは、コーディネータ・ジョブ・キュー・プロセスがインスタンスによって起動されるかどうかを制御します。このパラメータを 0（ゼロ）に設定すると、データベースの起動時にコーディネータ・ジョブ・キュー・プロセスは起動されず、その結果ジョブ・キューのジョブは実行されません。また、JOB_QUEUE_PROCESSES 初期化パラメータは、1 つのインスタンスで同時に実行できる J*nnn* プロセスの最大数を指定します。指定可能なプロセス最大数は 1,000 です。

次の初期化パラメータ設定は、データベースの起動時にコーディネータ・ジョブ・キュー・プロセスを起動し、最大 60 個の同時 J*nnn* プロセスの起動を可能にします。

```
JOB_QUEUE_PROCESSES = 60
```

コーディネータ・ジョブ・キュー・プロセスは、DBA_JOBS ビューに表示されているジョブをスキャンする一定の周期内で、最大でも選択したジョブの実行に必要な数の J*nnn* プロセスしか起動しません。上の例では 60 個の同時 J*nnn* プロセスの起動が可能ですが、実行するために選択したジョブの数が 20 個の場合、コーディネータは 20 個（最低 20 個）のジョブの実行に必要な数の J*nnn* プロセスのみ起動または再利用します。すでに起動しておりアイドル状態の J*nnn* プロセスはすべて、再利用のために使用可能であるとみなされます。

J*nnn* プロセスは、ジョブの実行を完了すると、別のジョブを実行するためにポーリングします。実行のために選択されているジョブがない場合はアイドル状態に入りますが、定期的にウェイクアップして再びポーリングします。事前に決められた回数試行しても実行するジョブが見つからない場合、J*nnn* プロセスは終了します。

JOB_QUEUE_PROCESSES 初期化パラメータは動的であり、ALTER SYSTEM 文によって変更できます。たとえば、次の文は同時 J*nnn* プロセスの許容最大数を 20 に設定します。

```
ALTER SYSTEM SET JOB_QUEUE_PROCESSES = 20;
```

新しい値が前の設定よりも小さく、現在実行中の J*nnn* プロセスの数よりも少ない場合は、まだ終了していない余分なプロセスを完了できます。

インスタンスが制限モードで実行されている場合、J*nnn* プロセスはジョブを実行しません。

関連項目： 制限モードの設定および解除の詳細は、4-10 ページの「オープンしているデータベースへのアクセスを制限する方法」を参照してください。

ジョブ・キューの管理

ここでは、ジョブ・キューの管理について説明します。この項の内容は、次のとおりです。

- DBMS_JOB パッケージ
- ジョブをジョブ・キューに送る方法
- ジョブの実行方法
- ジョブ・キューからのジョブの削除
- ジョブの変更
- 中断されたジョブ
- ジョブの強制的な実行
- ジョブの終了

DBMS_JOB パッケージ

ジョブ・キュー内のジョブをスケジュールし、管理するには、DBMS_JOB パッケージのプロシージャを使用します。ジョブ・キューの使用に関連するデータベース権限はありません。ジョブ・キュー・プロシージャを実行できるユーザーはすべて、ジョブ・キューを使用できます。

DBMS_JOB パッケージのプロシージャを次に示します。ここでは、これらのプロシージャについて説明します。

プロシージャ	説明
SUBMIT	ジョブをジョブ・キューに送ります。10-4 ページの「ジョブをジョブ・キューに送る方法」を参照してください。
REMOVE	指定したジョブをジョブ・キューから削除します。10-10 ページの「ジョブ・キューからのジョブの削除」を参照してください。
CHANGE	すでにジョブ・キューに送られている指定のジョブを変更します。ジョブの記述、ジョブの実行時刻、ジョブの実行間隔を変更できます。10-10 ページの「ジョブの変更」を参照してください。
WHAT	指定したジョブの記述を変更します。10-10 ページの「ジョブの変更」を参照してください。

プロシージャ	説明
NEXT_DATE	指定したジョブの次の実行時刻を変更します。10-10 ページの「ジョブの変更」を参照してください。
INTERVAL	指定したジョブの実行間隔を変更します。10-10 ページの「ジョブの変更」を参照してください。
BROKEN	ジョブ中断フラグを設定またはリセットします。ジョブに中断状態を示すマークを付けると、Oracle はそのジョブを実行しません。10-12 ページの「中断されたジョブ」を参照してください。
RUN	指定したジョブを強制的に実行します。10-13 ページの「ジョブの強制的な実行」を参照してください。

関連項目：

- DBMS_JOB パッケージの構文情報と、Oracle Real Application Clusters 環境で DBMS_JOB パッケージを使用する際に使用可能なその他のオプションの詳細は、『Oracle9*i* 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。

ジョブをジョブ・キューに送る方法

新しいジョブをジョブ・キューに送るには、DBMS_JOB パッケージの SUBMIT プロシージャを使用します。SUBMIT プロシージャでは、次のパラメータを指定します。

パラメータ	説明
JOB	出力パラメータ。作成するジョブに割り当てる識別子。ジョブを変更または削除するときは、必ずこのジョブ番号を使用する必要があります。10-6 ページの「ジョブ番号」を参照してください。
WHAT	実行する PL/SQL コード。10-6 ページの「ジョブ定義」を参照してください。
NEXT_DATE	次にジョブを実行する日付。デフォルト値は SYSDATE です。
INTERVAL	次にジョブを実行する時刻を計算する日付関数。デフォルト値は NULL です。INTERVAL は、将来のある時点または NULL として評価される必要があります。10-7 ページの「ジョブの実行間隔」を参照してください。
NO_PARSE	フラグ。NO_PARSE を FALSE（デフォルト）に設定すると、Oracle はそのジョブに対応付けられているプロシージャを解析します。NO_PARSE を TRUE に設定すると、Oracle はそのジョブの最初の実行時にそのジョブに対応付けられたプロシージャを解析します。たとえば、ジョブに関連する表を作成する前に、そのジョブを送る場合は、NO_PARSE を TRUE に設定します。

たとえば、新しいジョブをジョブ・キューに送ってジョブ番号を出力する次の文を考えます。このジョブはプロシージャ DBMS_DDL.ANALYZE_OBJECT をコールして、表 hr.employees の統計を生成します。統計は、employees 表の行の半分をサンプルとします。このジョブは、24 時間ごとに実行されます。

```
VARIABLE jobno NUMBER
BEGIN
   DBMS_JOB.SUBMIT(:jobno,
      'DBMS_DDL.ANALYZE_OBJECT(''TABLE'',
      ''HR'', ''EMPLOYEES'',
      ''ESTIMATE'', NULL, 50);',
      SYSDATE, 'SYSDATE + 1');
   COMMIT;
END;
/
PRINT jobno

JOBNO
----------
14144
```

注意： 送ったジョブを実行するには、DBMS_JOB.SUBMIT 文の直後に COMMIT 文を発行する必要があります。

ジョブ環境

ジョブがジョブ・キューに送られるか、ジョブの定義が変更されると、Oracle は次の環境特性を記録します。

- 現行ユーザー
- ジョブを送るユーザーまたはジョブを変更するユーザー
- 現行スキーマ（ALTER SESSION SET CURRENT_SCHEMA 文が発行された場合、これは現行ユーザーまたはジョブを送るユーザーと異なる場合があります）

次の NLS パラメータも記録されます。

- NLS_LANGUAGE
- NLS_TERRITORY
- NLS_CURRENCY
- NLS_ISO_CURRENCY
- NLS_NUMERIC_CHARACTERS
- NLS_DATE_FORMAT

- NLS_DATE_LANGUAGE
- NLS_SORT

これらの環境特性は、ジョブが実行されるたびにすべてリストアされます。NLS_LANGUAGE パラメータと NLS_TERRITORY パラメータによって、未指定の NLS パラメータのデフォルトが決まります。

DBMS_SQL パッケージと ALTER SESSION 文を使用して、ジョブの環境を変更できます。

関連項目：

- DBMS_SQL パッケージの詳細は、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。
- ALTER SESSION 文を使用してジョブの環境を変更する方法については、『Oracle9*i* SQL リファレンス』を参照してください。

ジョブとインポート / エクスポート

ジョブは、インポートまたはエクスポートできます。そのため、あるデータベースで定義したジョブを別のデータベースに転送できます。ジョブをインポートまたはエクスポートするときに、ジョブ番号、環境および定義が変更されることはありません。

注意： インポートするジョブのジョブ番号がデータベース内にすでに存在するジョブの番号と同じ場合、そのジョブはインポートできません。そのジョブをデータベースの新しいジョブとして送ってください。

ジョブの所有者

ジョブをジョブ・キューに送ったユーザーは、ジョブの所有者として識別されます。ジョブの変更や強制実行、キューからのジョブの削除ができるのは、ジョブの所有者のみです。

ジョブ番号

キューに入っているジョブは、ジョブ番号によって識別されます。ジョブを送ると、ユーザー SYS が所有する JOBSEQ 順序を使用してジョブ番号が自動的に生成されます。一度ジョブにジョブ番号が割り当てられると、その番号は変わりません。ジョブをインポートまたはエクスポートしても、同じジョブ番号のままです。

ジョブ定義

ジョブ定義とは、SUBMIT プロシージャの WHAT パラメータに指定された PL/SQL コードです。標準では、ジョブ定義は 1 つのプロシージャを 1 度コールします。プロシージャ・コールには、任意の数のパラメータを指定できます。

注意： ジョブ定義では、文字列の前後を一重引用符で囲んでください。ジョブ定義の末尾には必ずセミコロンを入れてください。

次に、有効なジョブ定義の例を示します。

- `'myproc(''10-JAN-99'', next_date, broken);'`
- `'scott.emppackage.give_raise(''JFEE'', 3000.00);'`
- `'dbms_job.remove(job);'`

注意： ジョブからジョブを実行する動作はサポートされていません。実行しようとするとエラー・メッセージが表示されます。たとえば、次の文を実行すると、「ORA-32317 あるジョブから他のジョブの実行はできません」というエラー・メッセージがアラート・ファイルに出力されます。

```
DECLARE
  jobno number;
BEGIN
  DBMS_JOB.SUBMIT(jobno, 'DBMS_JOB.RUN(23587);');
  DBMS_JOB.RUN(jobno);
END;
/
```

ジョブの実行間隔

設定した間隔でジョブを定期的に実行するには、INTERVAL パラメータで 'SYSDATE + 7' のような日付式を使用します。次の表は、ジョブの実行間隔を求めるための一般的な日付式を示しています。

日付式	評価
'SYSDATE + 7'	前回の実行からちょうど 7 日目
'SYSDATE + 1/48'	30 分ごと
'NEXT_DAY(TRUNC(SYSDATE), ''MONDAY'') + 15/24'	毎週月曜日午後 3 時
'NEXT_DAY(ADD_MONTHS(TRUNC(SYSDATE, ''Q''), 3), ''THURSDAY'')'	四半期ごとの第一木曜日

注意： NEXT_DATE または INTERVAL を指定するときは、日付リテラルと日付文字列を必ず一重引用符で囲んでください。また、INTERVAL の値も一重引用符で囲む必要があります。

ジョブの実行の直前に、INTERVAL 日付関数が評価されます。ジョブが正常に実行されると、INTERVAL で計算された日付が新しい NEXT_DATE になります。たとえば、月曜日に実行間隔を 'SYSDATE + 7' と設定したときに、なんらかの理由（ネットワーク障害など）でジョブが木曜日まで実行されない場合は、'SYSDATE + 7' が月曜日ではなく毎週木曜日に実行されます。INTERVAL 日付関数が NULL に評価され、ジョブが正常に終了すると、そのジョブはキューから削除されます。

ジョブを必ず特定の時間に自動的に実行するには、前回の実行時期（たとえば、毎週月曜日）に関係なく、INTERVAL パラメータと NEXT_DATE パラメータで 'NEXT_DAY(TRUNC(SYSDATE), ''MONDAY'')' のような日付式を指定する必要があります。

データベース・リンクとジョブ

送られたジョブがデータベース・リンクを使用する場合は、そのリンクにユーザー名とパスワードを含める必要があります。名前のないデータベース・リンクは正常に機能しません。

ジョブの実行方法

ジョブは、*Jnnn* プロセスによって実行されます。このプロセスは、ジョブを実行するために、ジョブ実行用のセッションを作成します。*Jnnn* プロセスがジョブを実行するとき、ジョブは送られたときと同じ環境で、所有者のデフォルトの権限で実行されます。所有者には、ジョブ定義内部で参照されるすべてのオブジェクトに対して必要なオブジェクト権限が明示的に付与されている必要があります。

DBMS_JOB.RUN プロシージャを使用してジョブを強制的に実行する場合、ジョブはユーザー・プロセスによって、デフォルトの権限のみで実行されます。ロールを介してユーザーに付与された権限は、使用されません。ユーザーには、ジョブ定義内部で参照されるすべてのオブジェクトに対して必要なオブジェクト権限が明示的に付与されている必要があります。

ジョブ・キューのロック

Oracle はジョブ・キューのロックを使用して、1 つのジョブが一度に 1 つのセッションのみで実行されるようにします。ジョブの実行中、セッションはそのジョブのジョブ・キュー（JQ）をロックします。データ・ディクショナリのロック・ビューを使用することにより、現在セッションによって保持されているロックについての情報を調べることができます。

次の問合せは、JQ ロックを保持しているすべてのセッションのセッション識別子、ロック・タイプおよびロック識別子をリストします。

```
SELECT SID, TYPE, ID1, ID2
   FROM V$LOCK
   WHERE TYPE = 'JQ';

      SID TY       ID1       ID2
--------- -- --------- ---------
       12 JQ         0     14144
1 row selected.
```

この問合せでは、ロックを保持しているセッションの識別子は 12 です。JQ ロックの場合、ID1 列は常に 0（ゼロ）になります。ID2 列は、セッションで実行中のジョブのジョブ番号を表します。このビューを DBA_JOBS_RUNNING ビューと結合すると、ジョブの詳細情報が得られます。

関連項目：

- ビューの詳細は、10-14 ページの「ジョブ・キューに関する情報の表示」を参照してください。
- V$LOCK ビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。
- ロックの詳細は、『Oracle9*i* データベース概要』を参照してください。

ジョブ実行のエラー

ジョブの実行に失敗すると、トレース・ファイルとアラート・ログにその失敗に関する情報が記録されます。Oracle はメッセージ番号 ORA-12012 と、失敗したジョブのジョブ番号を書き込みます。

キューに入っているジョブの正常な実行を妨げる原因には、次のものがあります。

- ネットワークまたはインスタンスの障害
- ジョブ実行時の例外

実行中のジョブがエラーを返した場合、Oracle はそのジョブを再実行しようとします。1 分後に最初の実行が試みられ、2 分後に 2 度目、4 分後に 3 度目というようにそれぞれの試行の間隔は 2 倍になっていきます。ジョブの失敗回数が 16 回になると、そのジョブには自動的に中断のマークが付けられ、それ以上そのジョブは実行されなくなります。ただし、各試行の間に、ジョブの実行を妨げている問題を訂正できる可能性があります。この訂正処理によって再試行のサイクルが中断することはなく、ジョブの実行が再度試みられます。

ジョブ・キューからのジョブの削除

ジョブ・キューからジョブを削除するには、DBMS_JOB パッケージの REMOVE プロシージャを使用します。

次の文は、ジョブ・キューからジョブ番号 14144 のジョブを削除します。

```
BEGIN
DBMS_JOB.REMOVE(14144);
END;
/
```

制限事項：

- 現在実行中のジョブをジョブ・キューから削除できます。ただし、そのジョブは中断されず、現行の実行が完了するまで処理が継続されます。
- 削除できるのは、自分が所有しているジョブのみです。自分が所有していないジョブを削除しようとすると、そのジョブがジョブ・キューに存在しないことを示すメッセージが表示されます。

ジョブの変更

ジョブ・キューに送られているジョブを変更するには、DBMS_JOB パッケージの CHANGE、WHAT、NEXT_DATE または INTERVAL プロシージャを使用します。

制限事項：

- 変更できるのは、自分の所有しているジョブのみです。自分の所有していないジョブを変更しようとすると、そのジョブがジョブ・キューに存在しないことを示すメッセージが表示されます。

CHANGE

DBMS_JOB.CHANGE プロシージャをコールすると、ユーザー定義可能なパラメータで、ジョブに対応付けられているものであれば、どのパラメータでも変更できます。

次の例では、ジョブ番号 14144 のジョブを 3 日ごとに実行するように変更しています。

```
BEGIN
DBMS_JOB.CHANGE(14144, NULL, NULL, 'SYSDATE + 3');
END;
/
```

プロシージャ DBMS_JOB.CHANGE をコールするときに、WHAT、NEXT_DATE、INTERVAL に NULL を指定すると、現行の設定値は変更されません。

注意： プロシージャ DBMS_JOB.CHANGE で WHAT パラメータを使用してジョブの定義を変更すると、Oracle は現行の環境を記録します。この環境が、そのジョブの新しい環境になります。

WHAT

DBMS_JOB.WHAT プロシージャをコールして、ジョブの定義を変更できます。

次の例は、ジョブ番号 14144 の定義を変更しています。

```
BEGIN
DBMS_JOB.WHAT(14144,
      'DBMS_DDL.ANALYZE_OBJECT(''TABLE'',
      ''HR'', ''DEPARTMENTS'',
      ''ESTIMATE'', NULL, 50);');
END;
/
```

注意： プロシージャ DBMS_JOB.WHAT を実行すると、Oracle は現行の環境を記録します。この環境が、そのジョブの新しい環境になります。

NEXT_DATE

次の例のように DBMS_JOB.NEXT_DATE プロシージャをコールすると、次にジョブを実行する日付を変更できます。

```
BEGIN
DBMS_JOB.NEXT_DATE(14144, 'SYSDATE + 4');
END;
/
```

INTERVAL

次の例は、DBMS_JOB.INTERVAL プロシージャをコールして、ジョブの実行間隔を変更する方法を示しています。

```
BEGIN
DBMS_JOB.INTERVAL(14144, 'NULL');
END;
/
```

この場合、ジョブは一度正常終了した後は実行されず、ジョブ・キューから削除されます。

中断されたジョブ

ジョブには、中断または非中断状態を示すラベルが付けられます。中断されたジョブは実行されません。ただし、DBMS_JOB.RUN プロシージャをコールすると、中断されたジョブを強制的に実行できます。

ジョブが中断される原因

ジョブをキューに送ったとき、そのジョブは中断されていないとみなされます。

ジョブは、次の 2 つの場合に中断します。

- Oracle がジョブの実行を 16 回試みても、ジョブが正常に実行されなかった場合
- 次のように、DBMS_JOB.BROKEN プロシージャを使用して、ジョブに中断されたことを示すマークを付けた場合

```
BEGIN
DBMS_JOB.BROKEN(14144, TRUE);
END;
/
```

中断のマークを付けられたジョブは、非中断のマークを付けるか、または DBMS_JOB.RUN プロシージャをコールしてジョブを強制的に実行しないかぎり、Oracle はこのジョブを実行しません。

次の例では、ジョブ 14144 に非中断のマークを付け、次回の実行の日付を次の月曜日に設定しています。

```
BEGIN
DBMS_JOB.BROKEN(14144, FALSE, NEXT_DAY(SYSDATE, 'MONDAY'));
END;
/
```

制限事項 :

- 中断のマークを付けることができるのは、自分が所有しているジョブのみです。自分が所有していないジョブに対して DBMS_JOB.BROKEN をコールすると、そのジョブがジョブ・キューに存在しないことを示すメッセージが表示されます。

中断されたジョブの実行

問題が発生してジョブの実行が 16 回失敗すると、Oracle はそのジョブに中断のマークを付けます。発生した問題を訂正した後、次のどちらかの方法でこのジョブを実行できます。

- DBMS_JOB.RUN をコールして、ジョブを強制実行する。
- DBMS_JOB.BROKEN をコールしてジョブに非中断のマークを付け、Oracle がジョブを実行するのを待機する。

DBMS_JOB.RUN プロシージャをコールしてジョブを強制実行すると、そのジョブはただちに実行されます。ジョブが正常に実行されると、Oracle はそのジョブに非中断のマークを付け、そのジョブが実行に失敗した回数のカウントを 0（ゼロ）にリセットします。

RUN または BROKEN をコールしてジョブの中断フラグをリセットした後は、そのジョブに対してスケジュールされた実行間隔に従ってジョブの実行が再開されます。

ジョブの強制的な実行

ジョブを手動で実行することが必要な場合があります。たとえば、中断されたジョブを修正したときに、そのジョブを強制実行してただちにテストする場合です。ジョブをただちに強制実行するには、DBMS_JOB パッケージのプロシージャ RUN を使用します。

DBMS_JOB.RUN を使用してジョブを実行するとき、Oracle は次の実行の日付を再計算します。たとえば、SYSDATE の NEXT_DATE 値と 'SYSDATE + 7' の INTERVAL 値を使用して月曜日にジョブを作成すると、そのジョブは月曜日から 7 日ごとに実行されます。ただし、水曜日に RUN を実行すると、次の実行の日付が次の水曜日になります。

次の文を発行すると、ジョブ 14144 が現行セッションで実行され、次の実行の日付が再計算されます。

```
BEGIN
DBMS_JOB.RUN(14144);
END;
/
```

注意： ジョブを強制的に実行すると、そのジョブは現行セッションで実行されます。ジョブを実行すると、セッションのパッケージが再度初期化されます。

制限事項：

- 実行できるのは、自分が所有しているジョブのみです。自分が所有していないジョブを実行しようとすると、そのジョブがジョブ・キューに存在しないことを示すメッセージが表示されます。
- RUN プロシージャには、暗黙的コミットが含まれています。RUN を使用してジョブを実行した後は、ロールバックできません。

ジョブの終了

ジョブに中断のマークを付け、そのジョブを実行しているセッションを識別して、そのセッションを切断すると、実行中のジョブを停止できます。Oracle がジョブを再実行しないように、そのジョブに中断のマークを付ける必要があります。

前述の V$SESSION または V$LOCK を使用して、ジョブを実行中のセッションを識別した後に、SQL 文 ALTER SYSTEM を使用してそのセッションを切断できます。ジョブとセッションについての情報の表示例は、「ジョブ・キューに関する情報の表示」を参照してください。

関連項目：

- V$SESSION の詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。
- 10-2 ページ「セッションの停止」

ジョブ・キューに関する情報の表示

次のデータ・ディクショナリ・ビューを使用して、ジョブ・キュー内のジョブについての情報を表示できます。

ビュー	説明
DBA_JOBS ALL_JOBS USER_JOBS	DBA ビューには、データベース内のすべてのジョブが表示されます。ALL ビューには、現行ユーザーからアクセス可能なすべてのジョブが表示されます。USER ビューには、現行ユーザーが所有しているすべてのジョブが表示されます。
DBA_JOBS_RUNNING	データベース内の実行中のジョブをすべてリストします。このビューを V$LOCK と結合すると、ロックを保持しているジョブを識別できます。

ジョブに関する情報の表示

次の問合せでは、キューに送られている各ジョブのジョブ番号、次の実行時刻、失敗回数および中断の状態がリストされています。

```
SELECT JOB, NEXT_DATE, NEXT_SEC, FAILURES, BROKEN
   FROM DBA_JOBS;

JOB      NEXT_DATE  NEXT_SEC  FAILURES  B
-------  ---------  --------  --------  -
   9125  01-JUN-01  00:00:00         4  N
  14144  24-OCT-01  16:35:35         0  N
   9127  01-JUN-01  00:00:00        16  Y
3 rows selected.
```

ジョブの実行に関する情報の表示

また、現在実行中のジョブのみに限定した情報も表示できます。次の問合せでは、現在実行中のすべてのジョブについて、セッション識別子、ジョブ番号、ジョブを送ったユーザーおよび開始時刻がリストされています。

```
SELECT SID, r.JOB, LOG_USER, r.THIS_DATE, r.THIS_SEC
  FROM DBA_JOBS_RUNNING r, DBA_JOBS j
  WHERE r.JOB = j.JOB;

SID        JOB        LOG_USER      THIS_DATE  THIS_SEC
-----  ----------   -------------   ---------  --------
   12       14144   HR              24-OCT-94  17:21:24
   25        8536   QS              24-OCT-94  16:45:12
2 rows selected.
```

関連項目： データ・ディクショナリ・ビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

11

表領域の管理

この章では、表領域の管理について説明します。この章の内容は、次のとおりです。

- 表領域を管理するためのガイドライン
- 表領域の作成
- ディクショナリ管理表領域の空き領域の結合
- 表領域の非標準のブロック・サイズの指定
- REDO レコードの書込みの制御
- 表領域の可用性の変更
- 読取り専用表領域の使用
- 表領域の削除
- ローカル管理表領域の問題の診断と修復
- ローカル管理表領域への SYSTEM 表領域の移行
- データベース間での表領域のトランスポート
- 表領域情報の表示

関連項目： Oracle データベースによって作成および管理されるデータ・ファイルと一時ファイルの作成方法は、第 3 章の「Oracle Managed Files の使用」を参照してください。

表領域を管理するためのガイドライン

Oracle データベースの表領域を使用して作業する前に、次の項で説明するガイドラインについて理解してください。

- 複数の表領域の使用
- 表領域のデフォルト記憶域パラメータの指定
- ユーザーに対する表領域割当て制限の割当て

関連項目： データベース構造、領域管理、表領域およびデータ・ファイルの詳細は、『Oracle9*i* データベース概要』を参照してください。

複数の表領域の使用

データベース操作を実行する際に複数の表領域を使用すると、システムの柔軟性が向上します。たとえば、データベースに複数の表領域があるときには、次のことが可能です。

- ユーザー・データをデータ・ディクショナリ・データから分離し、同じデータ・ファイルに関するディクショナリ・オブジェクトとスキーマ・オブジェクトの競合を減らす。
- あるアプリケーションのデータを別のアプリケーションのデータから分離し、表領域をオフライン化する必要が生じた場合に、複数のアプリケーションが影響を受けないようにする。
- I/O の競合を低減するために、別々のディスク・ドライブ上に異なる表領域のデータ・ファイルを配置する。
- 単一のディスク障害によってデータが永久に失われないように、ロールバック・セグメントのデータをユーザー・データから分離する。
- 別の表領域をオンライン状態に維持しながら、個々の表領域をオフライン化して、全体の可用性を高める。
- 高い更新アクティビティ、読取り専用アクティビティ、一時セグメント記憶域など、異なるタイプのデータベース利用のために異なる表領域を確保する。これにより、表領域の使用を最適化できます。
- 表領域のバックアップを個別に作成する。

一部のオペレーティング・システムでは、同時にオープン可能なファイルの数に制限が設けられています。このような制限によって、同時にオンライン化可能な表領域の数に影響が出ることがあります。そのため、使用しているオペレーティング・システムの制限を超えないように、表領域を効率よく計画する必要があります。表領域はデータベースの要件を満たすために必要な数だけ作成し、構成するファイル数もできるかぎり少なくなるようにしてください。表領域のサイズを大きくする必要がある場合は、小さいデータ・ファイルを多数作成するのではなく、1 つまたは 2 つの大きなデータ・ファイルを追加するか、または自動拡張オプションをオンに設定してデータ・ファイルを作成します。

これらの要素を考慮に入れてデータを再検討し、データベース設計に必要となる表領域の数を決定してください。

表領域のデフォルト記憶域パラメータの指定

新しいディクショナリ管理表領域を作成するときは、その表領域に作成するオブジェクトに対して、デフォルトの記憶域パラメータを指定できます。オブジェクトが格納される表領域のデフォルトの記憶域パラメータは、オブジェクトの作成時に指定した記憶域パラメータによって上書きされます。オブジェクトの作成時に記憶域パラメータを指定しないと、そのオブジェクトのセグメントは自動的に表領域のデフォルト記憶域パラメータを使用します。

表領域のデフォルト記憶域パラメータは、その表領域に格納される一般的なオブジェクトのサイズを見積った上で設定してください。例外的なオブジェクトに対する記憶域パラメータは、そのオブジェクトを作成するときに指定できます。また、後でデフォルトの記憶域パラメータを変更することも可能です。

ローカル管理として作成する表領域には、デフォルトの記憶域パラメータを指定できません。

注意： 新しいディクショナリ管理表領域にデフォルトの記憶域パラメータを指定しないと、オペレーティング・システムに適切なデフォルトの記憶域パラメータが選択されます。

ユーザーに対する表領域割当て制限の割当て

表、クラスタ、マテリアライズド・ビュー、索引およびその他のオブジェクトを作成しようとするユーザーには、そのオブジェクトを作成するための権限と、そのオブジェクトのセグメントを格納する表領域の**割当て制限**（領域の許容または制限）を付与します。オブジェクトを作成するために必要な権限をデータベース・ユーザーに付与し、また必要に応じて表領域の割当て制限をデータベース・ユーザーに割り当てるのは、セキュリティ管理者の役割です。

関連項目： 24-4 ページ「表領域割当て制限の割当て」

表領域の作成

表領域を作成する前に、それを格納するデータベースを作成する必要があります。どのデータベースでも、最初の表領域は常に SYSTEM 表領域です。そのため、データベースの作成時には、データベースの最初のデータ・ファイルが SYSTEM 表領域に自動的に割り当てられます。

表領域を作成する手順は、オペレーティング・システムによって異なります。ただし、いずれの場合も、データ・ファイルが割り当てられるディレクトリ構造は、オペレーティング・システムを通じて作成する必要があります。ほとんどのオペレーティング・システムでは、新しい表領域を作成するとき、またはデータ・ファイルを加えて表領域を変更するときに、データ・ファイルのサイズと完全なファイル名を指定します。Oracle は、それぞれの状況で、指定されたとおりにデータ・ファイルを自動的に割り当てて、フォーマットします。

新しい表領域を作成するには、SQL 文 CREATE TABLESPACE または CREATE TEMPORARY TABLESPACE を使用します。表領域を作成するには、CREATE TABLESPACE システム権限が必要です。後で、ALTER TABLESPACE または ALTER DATABASE 文を使用して、この表領域を変更できます。そのためには、ALTER TABLESPACE または ALTER DATABASE システム権限が必要です。

Oracle8*i* より前のリリースでは、すべての表領域は**ディクショナリ管理**表領域として作成されていました。ディクショナリ管理表領域は、データ・ディクショナリ表を使用して領域の使用率を追跡します。Oracle8*i* からは、**ローカル管理**表領域を作成できるようになりました。この表領域では、データ・ディクショナリ表ではなくビットマップを使用して、使用済み領域と空き領域が追跡されます。これらのローカル管理表領域を使用すると、パフォーマンスが向上し、管理が容易になります。

注意： Oracle9*i* からは、次の条件の両方が満たされる場合、SYSTEM 表領域以外の永続表領域はローカル管理表領域がデフォルトになります。

- EXTENT MANAGEMENT 句が指定されていない場合。
- COMPATIBLE 初期化パラメータが 9.0.0 以上に設定されている場合。

また、UNDO 表領域と呼ばれる特別なタイプの表領域も作成できます。この表領域は、UNDO レコードを格納するために特別に設計されています。UNDO レコードとは Oracle が生成するレコードで、リカバリや読込み一貫性のために、または ROLLBACK 文の要求によって、データベースの変更をロールバックまたは取り消す際に Oracle によって使用されます。UNDO 表領域の作成と管理については、第 13 章の「UNDO 領域の管理」を参照してください。

永続表領域および一時表領域については、次の項で説明します。

- ローカル管理表領域
- ディクショナリ管理表領域
- 一時表領域

関連項目：

- データベース作成時に作成される表領域については、第 2 章の「Oracle データベースの作成」およびオペレーティング・システム固有の Oracle インストレーション・ガイドを参照してください。
- CREATE TABLESPACE、CREATE TEMPORARY TABLESPACE、ALTER TABLESPACE および ALTER DATABASE 文の構文と使用方法の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。
- 非標準のブロック・サイズを持つ表領域の作成に必要な初期化パラメータの詳細は、2-36 ページの「データベース・ブロック・サイズの指定」を参照してください。

ローカル管理表領域

ローカル管理表領域では、その表領域内のすべてのエクステント情報がビットマップを使用して追跡されるため、次のような利点があります。

- 領域の割当てと割当て解除では、エンキューなど一元管理されるリソースを必要とせず、ローカル管理のリソース（ヘッダー・ファイルに格納されているビットマップ）のみが変更されるため、領域操作の並行性と速度が改善されます。
- ディクショナリ管理の領域割当てに必要な場合があった再帰的操作が不要となるため、パフォーマンスが改善されます。
- ローカル管理の一時表領域（ソートなどに使用）が文字通りローカルに管理されることにより、UNDO や REDO が生成されなくなるため、読取り可能なスタンバイ・データベースを実現できます。
- 領域割当てが簡素化されます。AUTOALLOCATE 句を指定すると、適切なエクステント・サイズが自動的に選択されます。
- 必要な情報はファイル・ヘッダーとビットマップ・ブロックに格納されるため、データ・ディクショナリに対するユーザーの依存性が低下します。

SYSTEM 表領域を含め、すべての表領域をローカルに管理できます。

また、DBMS_SPACE_ADMIN パッケージにより、ローカル管理表領域のメンテナンス手順が提供されます。

関連項目：

- 2-27 ページ「ローカル管理の SYSTEM 表領域の作成」
- 11-29 ページ 「ローカル管理表領域の問題の診断と修復」

ローカル管理表領域の作成

ローカル管理表領域を作成するには、CREATE TABLESPACE 文の EXTENT MANAGEMENT 句に LOCAL を指定します。その他に、次の 2 つのオプションがあります。AUTOALLOCATE（デフォルト）を指定してエクステントを自動的に管理するオプションと、UNIFORM SIZE を指定して表領域を特定サイズの均一エクステントで管理するオプションです。

異なるエクステント・サイズが必要で、多数のエクステントを持つ様々なサイズのオブジェクトが表領域に格納されると予想される場合は、AUTOALLOCATE を選択してください。領域の割当てと割当て解除を厳密に制御しなくてもよい場合は、AUTOALLOCATE を選択すると表領域の管理作業が簡素化されます。ある程度の領域が無駄になるという欠点もありますが、ほとんどの場合、Oracle によって領域が管理されるという利点のほうが重要です。

これに対して、未使用領域の厳密な制御が必要で、オブジェクトに割り当てられる領域、エクステントの数およびサイズを正確に予測できる場合は、UNIFORM を選択してください。これにより、表領域から使用できない領域がなくなります。

注意： エクステント管理のタイプを明示的に指定せず、デフォルトでローカル管理表領域が作成される場合は、Oracle によりエクステント管理が次のように判断されます。

CREATE TABLESPACE 文に DEFAULT 記憶域句を指定しない場合は、自動割当てのローカル管理表領域が作成されます。

CREATE TABLESPACE 文に DEFAULT 記憶域句を指定した場合、Oracle では次のことが考慮されます。

- MINIMUM EXTENT 句を指定した場合は、MINIMUM EXTENT、INITIAL および NEXT の値が等しいかどうかと、PCTINCREASE の値が 0 かどうかが評価されます。3 つの値が等しく、PCTINCREASE の値が 0 の場合は、エクステント・サイズ = INITIAL で均一なローカル管理表領域が作成されます。MINIMUM EXTENT、INITIAL および NEXT パラメータの値が等しくない場合、または PCTINCREASE が 0 でない場合は、指定したエクステント記憶域パラメータが無視され、自動割当てのローカル管理表領域が作成されます。
- MINIMUM EXTENT 句を指定しない場合は、INITIAL および NEXT の記憶域パラメータの値が等しいかどうかと、PCTINCREASE が 0 かどうかのみが評価されます。2 つの値が等しく、PCTINCREASE が 0 の場合は、均一なローカル管理表領域が作成されます。それ以外の場合は、ローカル管理の表領域が作成され、自動的に割り当てられます。

次の文は、ローカル管理表領域 lmtbsb を作成し、AUTOALLOCATE を指定しています。

```
CREATE TABLESPACE lmtbsb DATAFILE '/u02/oracle/data/lmtbsb01.dbf' SIZE 50M
    EXTENT MANAGEMENT LOCAL AUTOALLOCATE;
```

AUTOALLOCATE を指定すると、表領域はシステム管理になり、最小エクステント・サイズは 64KB となります。自動割当て表領域内のオブジェクトに割り当てられる初期領域が大きくなります。これは、ディクショナリ管理表領域では、オブジェクトの最小サイズは 2 ブロックですが、自動割当てのローカル管理表領域では、最小オブジェクト・サイズが 64KB であるためです。

また、この表領域は、UNIFORM 句を指定して作成することもできます。UNIFORM SIZE を指定すると、表領域は指定した SIZE の均一サイズのエクステントを持つように管理されます。デフォルトの SIZE は 1MB です。

次の例では、128KB のエクステント・サイズを指定します。128KB の各エクステント（表領域のブロック・サイズが 2KB の場合は 64 個の Oracle ブロックに相当）が、このファイルのエクステント・ビットマップの 1 ビットで表されます。

```
CREATE TABLESPACE lmtbsb DATAFILE '/u02/oracle/data/lmtbsb01.dbf' SIZE 50M
    EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K;
```

EXTENT MANAGEMENT LOCAL を明示的に指定する場合は、DEFAULT の記憶域句、MINIMUM EXTENT または TEMPORARY を指定できません。ローカル管理の一時表領域を作成するには、CREATE TEMPORARY TABLESPACE 文を使用します。

注意： ローカル管理表領域にデータ・ファイルを割り当てる場合は、領域管理に使用されるメタデータ（エクステント・ビットマップまたは領域ヘッダー・セグメント）用の領域を考慮する必要があります。これはユーザー領域の一部です。たとえば、UNIFORM を指定するときに、エクステント管理句に SIZE パラメータを指定しないと、デフォルトのエクステント・サイズは 1MB になります。したがって、データ・ファイルには 1MB より大きいサイズ（少なくとも 1 ブロックとビットマップ用の領域分は大きいサイズ）を指定する必要があります。

ローカル管理表領域のセグメント領域管理の指定

CREATE TABLESPACE 文を使用してローカル管理表領域を作成するときに SEGMENT SPACE MANAGEMENT 句を使用すると、セグメント内の使用可能領域および使用済み領域の管理方法を指定できます。選択肢は次のとおりです。

- MANUAL

  MANUAL を指定すると、セグメント内の空き領域の管理に空きリストが使用されます。空きリストとは、行挿入用に使用可能な領域を持つデータ・ブロックのリストです。この形式でセグメント内の領域を管理する方法は、手動セグメント領域管理と呼ばれます。これは、表領域内で作成されるスキーマ・オブジェクトについて、PCTUSED、

FREELISTS および FREELISTS GROUPS 記憶域パラメータを指定してチューニングする必要があるためです。

MANUAL はデフォルトです。

- AUTO

  このキーワードを指定すると、セグメント内の空き領域の管理にビットマップが使用されます。この場合のビットマップとは、行挿入用に使用可能なブロック内の領域の量を基準にしてセグメント内の各データ・ブロックの状態を表したマップのことです。データ・ブロックで使用可能な領域が増加または減少すると、その新しい状態がビットマップに反映されます。ビットマップにより、Oracle による空き領域の管理をさらに自動化できます。そのため、このような形式の領域管理を自動セグメント領域管理と呼びます。

自動セグメント領域管理の方が、セグメント内の領域を管理する方法としては単純で効率的です。この方法では、表領域内で作成されるスキーマ・オブジェクトについて PCTUSED、FREELISTS および FREELISTS GROUPS 記憶域パラメータを指定してチューニングする必要がありません。仮にこれらの属性を指定したとしても、無視されます。

自動セグメント領域管理では、手動セグメント領域管理に比べて領域の使用効率が向上します。また、ユーザー数やインスタンス数の増加につれて拡張されるという点で自己チューニング型です。Real Application Clusters 環境の場合は、自動セグメント領域管理ではインスタンスに対する領域の動的アフィニティに対処できるため、空きリスト・グループの使用に伴う領域のハード・パーティショニングを回避できます。

多くの標準的な処理負荷の場合、自動セグメント領域管理を使用した場合のアプリケーションのパフォーマンスは、手動セグメント領域管理を使用して適切にチューニングされたアプリケーションよりも向上します。

次の文は、自動セグメント領域管理を行う lmtbsb 表領域を作成します。

```
CREATE TABLESPACE lmtbsb DATAFILE '/u02/oracle/data/lmtbsb01.dbf' SIZE 50M
    EXTENT MANAGEMENT LOCAL
    SEGMENT SPACE MANAGEMENT AUTO;
```

表領域作成時に指定する、セグメント内で使用可能な領域の管理方法は、その後表領域で作成されるすべてのセグメントに対して適用されます。選択した方法を後で変更することはできません。自動セグメント領域管理を指定できるのは、ローカル管理の永続表領域のみです。

ローカル管理表領域の変更

ローカル管理表領域をローカル管理の一時表領域に変更することはできません。また、セグメント領域の管理方法を変更することもできません。

ローカル管理表領域に対して ALTER TABLESPACE 文を使用するのは、次のような場合です。

- データ・ファイルを追加する場合。次に例を示します。

```
ALTER TABLESPACE lmtbsb
   ADD DATAFILE '/u02/oracle/data/lmtbsb02.dbf' SIZE 1M;
```

- 表領域の可用性（ONLINE/OFFLINE）を変更する場合。11-20 ページの「表領域の可用性の変更」を参照してください。
- 表領域を読取り専用または読取り / 書込み用にする場合。11-23 ページの「読取り専用表領域の使用」を参照してください。
- データ・ファイルの名前を変更したり、表領域内のデータ・ファイルのサイズの自動拡張を使用可能 / 使用禁止にする場合。第 12 章の「データ・ファイルの管理」を参照してください。

ローカル管理表領域では、使用可能エクステントを結合する必要はありません。

ディクショナリ管理表領域

Oracle9*i* から、表領域作成時のエクステント管理のデフォルトはローカル管理になりました。ただし、ディクショナリ管理表領域を作成することを明示的に指定できます。ディクショナリ管理表領域では、エクステントが割り当てられるか、再利用できるように解放されるたびに、Oracle によってデータ・ディクショナリ内の適切な表が更新されます。

ディクショナリ管理表領域の作成

例として、次の特性を持つ表領域 tbsa を作成します。

- 新しい表領域のデータには、1 つのデータ・ファイル（サイズは 50MB）が含まれます。
- EXTENT MANAGEMENT DICTIONARY を指定して、表領域をディクショナリ管理表領域として明示的に作成します。
- この表領域に作成されるすべてのセグメントに対して、デフォルト記憶域パラメータが指定されます。

次の文で、表領域 tbsb を作成します。

```
CREATE TABLESPACE tbsb
    DATAFILE '/u02/oracle/data/tbsa01.dbf' SIZE 50M
    EXTENT MANAGEMENT DICTIONARY
    DEFAULT STORAGE (
        INITIAL 50K
        NEXT 50K
        MINEXTENTS 2
        MAXEXTENTS 50
        PCTINCREASE 0);
```

前述の例に示した次のパラメータにより、表領域内のセグメントの記憶域割当てが決定されます。これらのパラメータは、データベースに格納されているデータへのアクセス所要時間と、そのデータベースの領域の使用効率に影響します。この種のパラメータは、記憶域パラメータと呼ばれます。

記憶域パラメータ	説明
INITIAL	セグメントの最初のエクステントのバイト数（KB または MB）を定義します。
NEXT	第 2 のエクステントのバイト数（KB または MB）を定義します。
PCTINCREASE	第 2（NEXT）エクステントより後に各エクステントが拡張される割合（百分率）。
MINEXTENTS	表領域の最初のセグメントの作成時に割り当てられるエクステントの数。
MAXEXTENTS	セグメントに許される最大エクステント数を決定します。UNLIMITED も指定できます。

CREATE TABLESPACE 文のもう 1 つのパラメータ MINIMUM EXTENT も、セグメント割当てに影響します。このパラメータを指定すると、表領域内のすべての使用可能エクステントと割当て済みエクステントのサイズが、少なくとも指定したバイト数（KB または MB）と同じか、またはその倍数になります。これにより、表領域内の空き領域の断片化を制御できます。

関連項目：

- 14-8 ページ「記憶域パラメータの設定」
- これらのパラメータの効果は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。
- 記憶域パラメータの詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

ディクショナリ管理表領域の変更

ALTER TABLESPACE 文を使用するケースの 1 つとして、データ・ファイルを追加する場合があります。次の文は、tbsa 表領域の新しいデータ・ファイルを作成します。

```
ALTER TABLESPACE tbsa
   ADD DATAFILE '/u02/oracle/data/tbsa02.dbf' SIZE 1M;
```

デフォルトの記憶域パラメータの変更が必要になる場合もあります。

次の例のように、ALTER TABLESPACE 文を使用すると、表領域のデフォルトの記憶域パラメータを変更できます。

```
ALTER TABLESPACE users
    DEFAULT STORAGE (
       NEXT 100K
       MAXEXTENTS 20
       PCTINCREASE 0);
```

表領域のデフォルト記憶域パラメータの新しい値は、その表領域内の既存のセグメントに対してその後作成されるオブジェクトまたは割り当てられるエクステントにのみ影響します。

その他に ALTER TABLESPACE 文を発行するのは、次のような場合です。

- 表領域の空き領域を結合する場合。11-15 ページの「ディクショナリ管理表領域の空き領域の結合」を参照してください。
- 表領域の可用性（ONLINE/OFFLINE）を変更する場合。11-20 ページの「表領域の可用性の変更」を参照してください。
- 表領域を読取り専用または読取り / 書込み用にする場合。11-23 ページの「読取り専用表領域の使用」を参照してください。
- データ・ファイルを追加または変更したり、表領域内のデータ・ファイルのサイズの自動拡張を使用可能 / 使用禁止にしたりする場合。第 12 章の「データ・ファイルの管理」を参照してください。

一時表領域

複数のソート操作の並行性の改善、オーバーヘッドの軽減または Oracle による領域管理操作の回避を実現するには、**一時表領域**を作成します。一時表領域は複数のユーザーで共有でき、データベース内にユーザーを作成するときに CREATE USER 文でユーザーに割り当てることができます。

インスタンスと表領域のソート操作はすべて、一時表領域内の 1 つの**ソート・セグメント**を共有します。ソート・セグメントは、その表領域の中でソート操作を実行するすべてのインスタンスに存在します。ソート・セグメントは、一時表領域を使用してソート処理を行う最初の文によってインスタンスの起動後に作成され、停止時にのみ解放されます。複数のトランザクションでエクステントを共有することはできません。

一時表領域のソート・セグメントの領域割当てと割当て解除は、V$SORT_SEGMENT ビューを使用して表示できます。V$TEMPSEG_USAGE ビューでは、そのセグメント内の現行のソート・ユーザーが識別されます。

一時表領域には、オブジェクトを明示的に作成できません。

関連項目：

- 一時表領域のユーザーへの割当ての詳細は、第 24 章の「ユーザーとリソースの管理」を参照してください。
- V$SORT_SEGMENT ビューおよびV$TEMPSEG_USAGE ビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。
- ソートのチューニングの詳細は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。

ローカル管理の一時表領域の作成

ローカル管理表領域では、領域の管理がより簡単で効率的であるため、一時表領域には理想的です。ローカル管理の一時表領域では、**一時ファイル**が使用されます。一時表領域の外側のデータは変更されず、一時表領域データの REDO は発生しません。したがって、スタンバイ・データベースまたは読取り専用データベースに使用できます。

一時ファイルの情報を表示するには、データ・ファイルの場合とは異なるビューを使用します。V$TEMPFILE および DBA_TEMP_FILES ビューは、V$DATAFILE および DBA_DATA_FILES ビューに相当します。

ローカル管理の一時表領域を作成するには、CREATE TEMPORARY TABLESPACE 文を使用します。この文を発行するには、CREATE TABLESPACE システム権限が必要です。

次の文では、各エクステントが 16MB の一時表領域が作成されます。16MB の各エクステント（標準ブロック・サイズが 2KB のときは 8000 個のブロックに相当）は、このファイルのビットマップに 1 ビットで表されます。

```
CREATE TEMPORARY TABLESPACE lmtemp TEMPFILE '/u02/oracle/data/lmtemp01.dbf'
     SIZE 20M REUSE
   EXTENT MANAGEMENT LOCAL UNIFORM SIZE 16M;
```

すべての一時表領域は均一サイズのローカル管理エクステントを使用して作成されるため、一時表領域の場合、エクステント管理句はオプションです。Oracle のデフォルトの SIZE は 1MB です。ただし、SIZE に別の値を指定する必要がある場合は、前述の文を使用します。

AUTOALLOCATE 句は、一時表領域には使用できません。

注意： 一部のオペレーティング・システムでは、一時ファイルのブロックが実際にアクセスされるまで、Oracle は一時ファイル用の領域を割り当てません。このような領域割当ての遅延により、一時ファイルの作成やサイズ変更が短時間で済みますが、一時ファイルを後で使用するときに十分なディスク領域が使用可能である必要があります。使用しているシステムにおいて Oracle がこのように一時ファイルを割り当てているかどうかは、オペレーティング・システムのマニュアルを参照の上で判断してください。

ローカル管理の一時表領域の変更

次の例のように一時ファイルを追加する場合を除き、ALTER TABLESPACE 文はローカル管理の一時表領域に使用できません。

```
ALTER TABLESPACE lmtemp
   ADD TEMPFILE '/u02/oracle/data/lmtemp02.dbf' SIZE 2M REUSE;
```

> **注意：** ALTER TABLESPACE 文に TEMPORARY キーワードを指定して、ローカル管理の永続表領域をローカル管理の一時表領域に変更することはできません。ローカル管理の一時表領域を作成するには、CREATE TEMPORARY TABLESPACE 文を使用する必要があります。

ただし、ALTER DATABASE を使用して一時ファイルを変更することはできます。

次の文では、一時ファイルをオフラインにしてから、オンラインに戻しています。

```
ALTER DATABASE TEMPFILE '/u02/oracle/data/lmtemp02.dbf' OFFLINE;
ALTER DATABASE TEMPFILE '/u02/oracle/data/lmtemp02.dbf' ONLINE;
```

次の文では、一時ファイルのサイズが変更されます。

```
ALTER DATABASE TEMPFILE '/u02/oracle/data/lmtemp02.dbf' RESIZE 4M;
```

次の文では、一時ファイルが削除され、オペレーティング・システム・ファイルが削除されます。

```
ALTER DATABASE TEMPFILE '/u02/oracle/data/lmtemp02.dbf' DROP
     INCLUDING DATAFILES;
```

この一時ファイルが属していた表領域は残ります。アラート・ファイルには、データ・ファイルが削除されたことを示すメッセージが書き込まれます。オペレーティング・システム・エラーによってファイルが削除されなかった場合でも文は正常終了しますが、エラーを示すメッセージがアラート・ファイルに書き込まれます。

また、例には示されていませんが、ALTER DATABASE 文を使用して、既存の一時ファイルの自動拡張を使用可能または使用禁止にしたり、一時ファイル名を変更（RENAME FILE）できます。

ディクショナリ管理の一時表領域の作成

表領域の作成時に一時表領域として指定するには、CREATE TABLESPACE 文に TEMPORARY キーワードを指定します。この方法で作成した一時表領域には、EXTENT MANAGEMENT LOCAL を指定できません。ローカル管理の一時表領域を作成するには、CREATE TEMPORARY TABLESPACE 文を使用します。一時表領域を作成するには、この方法の使用をお薦めします。

次の文は、ディクショナリ管理の一時表領域を作成します。

```
CREATE TABLESPACE sort
     DATAFILE '/u02/oracle/data/sort01.dbf' SIZE 50M
        DEFAULT STORAGE (
        INITIAL 2M
        NEXT 2M
        MINEXTENTS 1
        PCTINCREASE 0)
     EXTENT MANAGEMENT DICTIONARY
     TEMPORARY;
```

ディクショナリ管理の一時表領域の変更

ディクショナリ管理の一時表領域を作成するには、ディクショナリ管理の永続表領域の場合と同様のキーワードと句を使用して ALTER TABLESPACE 文を発行します。制限事項については、『Oracle9*i* SQL リファレンス』を参照してください。

注意： ALTER TABLESPACE ... OFFLINE 文を使用してディクショナリ管理の一時表領域をオフラインにしてからオンラインに戻しても、それらの一時ステータスに影響を及ぼすことはありません。

既存のディクショナリ管理の永続表領域を一時表領域に変更するには、ALTER TABLESPACE 文を使用します。次に例を示します。

```
ALTER TABLESPACE tbsa TEMPORARY;
```

ディクショナリ管理表領域の空き領域の結合

ディクショナリ管理表領域の空き領域が時間経過とともに断片化され、新しいエクステントの割当てが困難になる場合があります。ここでは、この空き領域の断片化を解消する方法について説明します。

この項の内容は、次のとおりです。

- Oracle による空き領域の結合方法
- 手動による空き領域の結合
- 空き領域の監視

Oracle による空き領域の結合方法

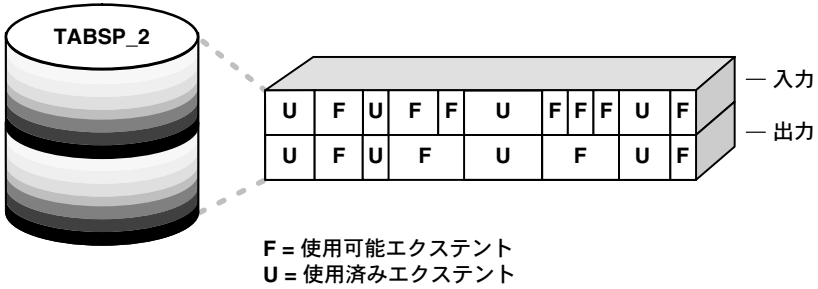
ディクショナリ管理表領域の使用可能エクステントは、連続する空きブロックの集合から構成されます。新しいエクステントを表領域セグメントに割り当てるときには、必要なエクステントに最も近いサイズの使用可能エクステントが使用されます。場合によっては、セグメントが削除されると、そのエクステントの割当てが解除されて使用可能マークが設定されますが、隣接する使用可能エクステントが即時に再結合され、より大きな使用可能エクステントになることはありません。その結果、断片化が生じ、より大きなエクステントの割当てが困難になります。

この断片化に対処するには、次のように複数の方法があります。

- セグメントに新規エクステントを割り当てるときに、最初にその新規エクステントが収まるサイズの使用可能エクステントが検索されます。十分な大きさの使用可能エクステントが見つからない場合は、表領域内で隣接する使用可能エクステントが結合されてから、再検索されます。この結合は、新規エクステントが収まるサイズの使用可能エクステントが見つからない場合に必ず実行されます。
- SMON バックグラウンド・プロセスは、表領域の PCTINCREASE の値が 0（ゼロ）以外に設定されている場合、隣接する使用可能エクステントどうしを定期的に結合します。PCTINCREASE=0 に設定すると、使用可能エクステントを結合する操作は発生しません。SMON の結合操作によるオーバーヘッドが問題になる場合は、PCTINCREASE=0 に設定し、定期的に手動で空き領域を結合してください。
- セグメントの PCTINCREASE の値が 0（ゼロ）以外に設定されている場合、そのセグメントを削除または切り捨てると、限定的に結合操作が実行されます。この動作は、そのセグメントを含む表領域に PCTINCREASE=0 が設定されている場合も実行されます。
- ALTER TABLESPACE ... COALESCE 文を使用すると、隣接する使用可能エクステントを手動で結合できます。

次の図は、空き領域を結合するプロセスを示しています。

図 11-1 空き領域の結合



注意： ローカル管理表領域では空き領域の結合は不要です。これは、ビットマップが隣接する空き領域を自動的に追跡するためです。

関連項目： エクステントの割当て方法と空き領域を結合する方法の詳細は、『Oracle9*i* データベース概要』を参照してください。

手動による空き領域の結合

表領域内の領域が著しく断片化されている（ディスク上の連続した領域が連続していないように見える）場合は、ALTER TABLESPACE ... COALESCE 文を使用して空き領域を結合できます。表領域を結合するには、ALTER TABLESPACE システム権限が必要です。

この文は、PCTINCREASE=0 の場合に使用するか、SMON およびエクステント割当ての結合操作を補完するために使用できます。表領域内のエクステントがすべて同一サイズの場合は、結合する必要はありません。これは、表領域の PCTINCREASE のデフォルト値が 0（ゼロ）で、すべてのセグメントにその表領域のデフォルトの記憶域パラメータが使用され、INITIAL=NEXT=MINIMUM EXTENT に設定されている場合が該当します。

次の文は、表領域 tabsp_4 の空き領域を結合します。

```
ALTER TABLESPACE tabsp_4 COALESCE;
```

ALTER TABLESPACE 文の他のオプションと同様に、COALESCE オプションも排他的であり、このオプションを指定すると他のオプションは指定できません。

この文では、データ・エクステントで区切られた使用可能エクステントは結合されません。データ・エクステントの間に多数の使用可能エクステントが存在する場合は、表領域を（たとえば、そのデータをエクスポートおよびインポートして）再編成し、使用可能な空き領域のエクステントを作成する必要があります。

空き領域の監視

表領域内の空き領域を監視するには、次のビューを使用します。

- DBA_FREE_SPACE
- DBA_FREE_SPACE_COALESCED

次の文は、表領域 tabsp_4 の空き領域を表示します。

```
SELECT  BLOCK_ID, BYTES, BLOCKS
     FROM  DBA_FREE_SPACE
     WHERE TABLESPACE_NAME = 'TABSP_4'
     ORDER BY BLOCK_ID;

BLOCK_ID   BYTES      BLOCKS
---------- ---------- ----------
         2      16384          2
         4      16384          2
         6      81920         10
        16      16384          2
        27      16384          2
        29      16384          2
        31      16384          2
        33      16384          2
        35      16384          2
        37      16384          2
        39       8192          1
        40       8192          1
        41     196608         24
13 rows selected.
```

このビューは、tabsp_4 内に、結合されていない隣接する空き領域（BLOCK_ID2、4、6、16 で始まるブロックなど）があることを示しています。前述の ALTER TABLESPACE 文を使用して表領域を結合すると、この問合せの結果は次のようになります。

```
BLOCK_ID   BYTES      BLOCKS
---------- ---------- ----------
         2     131072         16
        27     311296         38
2 rows selected.
```

DBA_FREE_SPACE_COALESCED ビューには、結合アクティビティの統計が表示されます。このビューは、領域を結合する必要があるかどうかを判断する際にも役立ちます。

> **関連項目：** これらのビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

表領域の非標準のブロック・サイズの指定

DB_BLOCK_SIZE 初期化パラメータで指定された標準のデータベース・ブロック・サイズとは異なるブロック・サイズの表領域を作成できます。この機能により、ブロック・サイズの異なる表領域をデータベース間でトランスポートできます。

CREATE TABLESPACE 文の BLOCKSIZE 句を使用すると、データベースの標準とは異なるブロック・サイズを指定して表領域を作成できます。ただし、システム・グローバル領域（SGA）メモリー内のバッファ・キャッシュを非標準のブロック・サイズに設定する必要があります。

次の文は表領域 lmtbsb を作成しますが、ブロック・サイズを DB_BLOCK_SIZE 初期化パラメータで指定されている標準のデータベース・ブロック・サイズとは異なるサイズにします。

```
CREATE TABLESPACE lmtbsb DATAFILE '/u02/oracle/data/lmtbsb01.dbf' SIZE 50M
    EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K;
    BLOCKSIZE 8K;
```

注意： BLOCKSIZE 句を正しく実行するために、DB_CACHE_SIZE と、少なくとも 1 つの DB_*n*K_CACHE_SIZE 初期化パラメータを設定し、この句で指定する整数を 1 つの DB_*n*K_CACHE_SIZE パラメータの設定に対応付ける必要があります。冗長な指定になりますが、BLOCKSIZE を DB_BLOCK_SIZE 初期化パラメータで指定されている標準のブロック・サイズと同じに指定することも可能です。

これらのパラメータの詳細は、2-39 ページの「バッファ・キャッシュ初期化パラメータの設定」を参照してください。

関連項目：

- 2-36 ページ「データベース・ブロック・サイズの指定」
- 11-34 ページ「データベース間での表領域のトランスポート」

REDO レコードの書込みの制御

一部のデータベース操作については、REDO レコードが生成されるかどうかを制御できます。REDO の生成を抑制するとパフォーマンスが改善され、簡単にリカバリできる操作に適している場合があります。これには CREATE TABLE...AS SELECT 文などが含まれます。これにより、データベース障害やインスタンス障害が発生した場合に、操作を繰り返すことができます。REDO を使用しないと、メディア・リカバリはできません。

これらの操作を表領域内のオブジェクトに対して実行するときに REDO を抑制する必要がある場合は、CREATE TABLESPACE 文に NOLOGGING 句を指定します。この句を指定しないか、かわりに LOGGING を指定した場合は、表領域内のオブジェクトに変更が行われると REDO が生成されます。一時セグメントや一時表領域の場合は、ロギング属性に関係なく REDO は生成されません。

表領域レベルで指定するロギング属性は、その表領域内で作成されるオブジェクトのデフォルト属性になります。このデフォルトのロギング属性は、CREATE TABLE 文を使用するなど、スキーマ・オブジェクト・レベルで LOGGING または NOLOGGING を指定すると上書きできます。

スタンバイ・データベースがある場合は、NOLOGGING 句を指定すると、スタンバイ・データベースの可用性と精度に問題が生じます。この問題を克服するために、FORCE LOGGING モードを指定できます。CREATE TABLESPACE 文に FORCE LOGGING 句を指定すると、表領域内のオブジェクトを変更するすべての操作について、REDO レコードを強制的に生成させることができます。これにより、オブジェクト・レベルでの指定が上書きされます。

FORCE LOGGING モードの表領域を別のデータベースにトランスポートすると、新しい表領域では FORCE LOGGING モードは維持されません。

関連項目：

- NOLOGGING モードの詳細は、『Oracle9*i* データベース概要』を参照してください。
- NOLOGGING モードで実行できる操作の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。
- FORCE LOGGING モードの詳細と、CREATE DATABASE 文で FORCE LOGGING 句を使用する効果の詳細は、2-29 ページの「FORCE LOGGING モードの指定」を参照してください。

表領域の可用性の変更

オンラインの表領域をオフライン化し、データベースのこの部分について一般的な使用を一時的に禁止することができます。データベースの残りの部分はオープンしていて使用可能であり、ユーザーはデータにアクセスできます。逆に、オフライン状態の表領域をオンライン化して、データベース・ユーザーがその表領域内のスキーマ・オブジェクトを使用できるようにすることもできます。データベースはオープンしている必要があります。

表領域の可用性を変更するには、SQL 文 ALTER TABLESPACE を使用します。そのためには、ALTER TABLESPACE または MANAGE TABLESPACE システム権限が必要です。

表領域内のすべてのデータ・ファイルまたは一時ファイルを、表領域自体の OFFLINE または ONLINE の状態に影響を与えずにオフライン化して、オンラインに戻すこともできます。

表領域のオフライン化

表領域は、次のような場合にオフライン化できます。

- データベースの一部のみを使用できないようにし、残りの部分には正常にアクセスできるようにする場合
- オフライン表領域のバックアップを実行する場合（ただし、表領域はオンラインで使用中の場合でもバックアップは可能）
- アプリケーションの更新時またはメンテナンス時に、アプリケーションとその表のグループを一時的に使用できないようにする場合

表領域をオフライン化すると、その関連ファイルがすべてオフライン化されます。SYSTEM 表領域をオフライン化することはできません。

表領域をオフライン化するときには、次のオプションのいずれかを指定できます。

オプション	説明
NORMAL	表領域のどのデータ・ファイルにもエラー条件が存在していない場合は、この表領域を通常の方法でオフライン化できます。書込みエラーが発生していると、現時点では表領域のデータ・ファイルをオフライン化することはできません。OFFLINE NORMAL を指定すると、Oracle は表領域のデータ・ファイルすべてのチェックポイントを取ってから、それらのファイルをオフライン化します。NORMAL はデフォルトです。

オプション	説明
TEMPORARY	表領域の 1 つまたは複数のデータ・ファイルについてエラー条件が存在している場合でも、表領域を一時的にオフライン化できます。OFFLINE TEMPORARY を指定すると、Oracle はまだオフライン化されていないデータ・ファイルのチェックポイントを取ってから、これらのファイルをオフライン化します。 オフラインになっているファイルがないときに表領域を一時的にオフライン化する場合は、表領域をオンラインに戻す前にメディア・リカバリを実行する必要はありません。しかし、表領域の 1 つまたは複数のファイルが書込みエラーのためにオフラインになっており、この表領域を一時的にオフライン化する場合は、表領域をオンラインに戻す前にリカバリする必要があります。
IMMEDIATE	表領域が即時にオフライン化されます。Oracle はデータ・ファイルのチェックポイントを取りません。OFFLINE IMMEDIATE を指定すると、表領域をオンライン化する前に表領域のメディア・リカバリが必要です。データベースを NOARCHIVELOG モードで運用している場合は、表領域を即時にオフライン化することはできません。
FOR RECOVER	表領域の Point-in-Time リカバリのために、リカバリ・セット内のデータベースの表領域をオフライン化できます。詳細は、『Oracle9*i* 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。

注意： 表領域をオフライン化する必要がある場合は、可能なかぎり NORMAL オプション（デフォルト）を使用してください。これにより、表領域をオンラインに戻すためのリカバリが不要になることが保証されます。不完全リカバリの後に ALTER DATABASE OPEN RESETLOGS 文を使用して REDO ログ順序をリセットした場合でも、リカバリは不要になります。

TEMPORARY は、表領域を通常の方法でオフライン化できないときのみ指定してください。この場合、エラーのためにオフライン化されたファイルのみをリカバリする必要があり、その後に表領域をオンライン化できます。IMMEDIATE は、NORMAL オプションと TEMPORARY オプションを試した後にのみ指定してください。

次の例では、users 表領域を通常の方法でオフライン化しています。

```
ALTER TABLESPACE users OFFLINE NORMAL;
```

オンライン表領域をオフライン化する前に、次の操作を検討してください。

- 表領域にアクティブなロールバック・セグメントが含まれていないことを確認します。この種の表領域はオフライン化できません。

- 表領域がデフォルトまたは一時表領域としてすでに割り当てられているユーザーについて、その表領域割当てを変更できます。このようなユーザーは表領域がオフラインの間その中のオブジェクトやソート領域にアクセスできないため、表領域割当ての変更をお薦めします。

  関連項目： 13-24 ページ「ロールバック・セグメントをオフライン化する方法」

表領域のオンライン化

Oracle データベースがオープンされている場合は、いつでもデータベース内の任意の表領域をオンライン化できます。通常、表領域は、データベース・ユーザーがその中のデータを使用できるようにオンラインになっています。

注意： オンライン化しようとする表領域が、正常に（ALTER TABLESPACE OFFLINE 文の NORMAL オプションを使用して）オフライン化されていない場合は、最初にメディア・リカバリをしないかぎりオンライン化できません。メディア・リカバリを実行しないと、エラーが返されて表領域はオフラインのままになります。

メディア・リカバリの実行方法の詳細は、アーカイブ計画に応じて次のいずれかのマニュアルを参照してください。

- 『Oracle9*i* ユーザー管理バックアップおよびリカバリ・ガイド』
- 『Oracle9*i* Recovery Manager ユーザーズ・ガイド』

次の文は、users 表領域をオンライン化します。

```
ALTER TABLESPACE users ONLINE;
```

データ・ファイルまたは一時ファイルの可用性の変更

ALTER TABLESPACE 文の句により、表領域内にあるすべてのデータ・ファイルまたは一時ファイルのオンラインまたはオフラインの状態を変更できます。具体的には、オンライン / オフラインの状態に影響を与える文として次のものがあります。

- ALTER TABLESPACE ... DATAFILE {ONLINE|OFFLINE}
- ALTER TABLESPACE ... TEMPFILE {ONLINE|OFFLINE}

入力が必要なのは表領域名のみであり、個々のデータ・ファイルや一時ファイルを入力する必要はありません。すべてのデータ・ファイルまたは一時ファイルが影響を受けますが、表領域そのもののオンライン / オフラインの状態は変わりません。

ほとんどの場合、データベースがマウントされていれば、オープンしていなくても、前述の ALTER TABLESPACE 文を発行できます。ただし、表領域が SYSTEM 表領域、UNDO 表領域またはデフォルト一時表領域である場合は、データベースをオープンしないでください。ALTER DATABASE DATAFILE 文および ALTER DATABASE TEMPFILE 文にも ONLINE/OFFLINE 句がありますが、これらの文では表領域のすべてのファイル名を入力する必要があります。

この操作は表領域の可用性を変更する ALTER TABLESPACE ... ONLINE|OFFLINE 文とは操作が異なるため、構文も異なります。ALTER TABLESPACE 文は表領域だけでなくデータ・ファイルもオフラインにしますが、一時表領域または一時ファイルの状態を変更するためには使用できません。

読取り専用表領域の使用

表領域を読取り専用にすると、表領域のデータ・ファイルに対して書込み操作ができなくなります。読取り専用表領域の主な目的は、データベース内の大規模かつ静的部分のバックアップおよびリカバリを実行しなくて済むようにすることですが、どのユーザーでもデータを変更できないように履歴データを完全に保護する手段でもあります。表領域を読取り専用にすると、その表領域内のすべての表はユーザーの更新権限レベルに関係なく更新できません。

注意： 表領域は、それが作成されたデータベース内でしかオンライン化できないため、読取り専用にすること自体でアーカイブ要件やデータ公開要件を満たすことはできません。ただし、トランスポータブル表領域機能を使用すると、これらの要件を満たすことができます。

表や索引などの項目は読取り専用表領域から削除できますが、表領域内のオブジェクトは作成または変更できません。ALTER TABLE ... ADD または ALTER TABLE ... MODIFY など、データ・ディクショナリ内のファイル記述を更新する文は実行できますが、新しい記述は表領域を読取り / 書込み用にするまでは使用できません。

読取り専用表領域は、他のデータベースにトランスポートすることもできます。読取り専用表領域は更新できないため、CD-ROM または Write Once-Read Many（WORM）デバイスに格納できます。

この項の内容は、次のとおりです。

- 表領域を読取り専用にする方法
- 読取り専用表領域を書込み可能にする方法
- WORM デバイスでの読取り専用表領域の作成
- 読取り専用表領域内にあるデータ・ファイルのオープンの遅延

関連項目：

- 読取り専用表領域の詳細は、『Oracle9*i* データベース概要』を参照してください。
- 11-34 ページ「データベース間での表領域のトランスポート」

表領域を読取り専用にする方法

すべての表領域は、最初は読取り / 書込み用として作成されます。表領域を読取り専用に変更するには、ALTER TABLESPACE 文で READ ONLY キーワードを使用します。そのためには、ALTER TABLESPACE または MANAGE TABLESPACE システム権限が必要です。

表領域を読取り専用にするには、あらかじめ次の条件を満たす必要があります。

- 表領域は必ずオンラインにする。

  これにより、表領域に適用する必要のあるロールバック情報がないことが保証されます。

- 表領域にアクティブなロールバック・セグメントを格納しない（データ表領域にはロールバック・セグメントは格納されないため、これは通常の状況です）。

  SYSTEM 表領域には SYSTEM ロールバック・セグメントが含まれているため、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、表領域を読取り専用にする前にロールバック・セグメントを削除しておく必要があります。

- 表領域を現行のオンライン・バックアップに含めない（オンライン・バックアップは、終了時に表領域内にあるすべてのデータ・ファイルのヘッダー・ファイルを更新するためです）。

読取り専用表領域のデータにアクセスする際のパフォーマンスを向上させるため、表領域を読取り専用にする直前に、表領域内の表のブロックすべてにアクセスする問合せを発行することをお薦めします。各表に対して SELECT COUNT (*) などの単純な問合せを実行しておくと、それ以降、表領域のデータ・ブロックに最も効率的にアクセスできるようになります。これにより、最後にブロックを変更したトランザクションの状態を Oracle が確認する必要がなくなるからです。

次の文は、flights 表領域を読取り専用にします。

```
ALTER TABLESPACE flights READ ONLY;
```

トランザクションが完了するまで待たなくても、ALTER TABLESPACE ... READ ONLY 文を発行できます。この文を発行すると、ターゲット表領域は、これ以上の書込み操作（DML 文）が許可されない推移読取り専用モードになります。その表領域を変更した既存トランザクションは、コミットまたはロールバックできます。データベース内のすべてのトランザクションが完了すると、表領域は読取り専用になります。

注意： この推移読取り専用状態になるのは、初期化パラメータ COMPATIBLE の値が 8.1.0 以上の場合のみです。このパラメータが 8.1.0 より小さい値に設定されている場合は、アクティブなトランザクションが存在していると、ALTER TABLESPACE ... READ ONLY 文は失敗します。

表領域の停止までに長時間かかる場合は、読取り専用状態になるのを妨げているトランザクションを識別できます。これらのトランザクションの所有者に通知し、必要に応じてトランザクションを終了させることができます。次の例に、ブロックしているトランザクションの識別方法を示します。

- ALTER TABLESPACE ... READ ONLY 文に対応するトランザクション・エントリを識別し、そのセッション・アドレス（saddr）をメモしておきます。

```
SELECT SQL_TEXT, SADDR
    FROM V$SQLAREA,V$SESSION
    WHERE V$SQLAREA.ADDRESS = V$SESSION.SQL_ADDRESS
        AND SQL_TEXT LIKE 'alter tablespace%';

SQL_TEXT                                 SADDR
---------------------------------------- --------
alter tablespace tbs1 read only          80034AF0
```

- 各アクティブ・トランザクションの開始システム変更番号（SCN）は、V$TRANSACTION ビューに格納されています。このビューを開始 SCN の昇順でソートして表示すると、トランザクションが実行順にリストされます。読取り専用文のトランザクション・エントリのセッション・アドレスがわかっているので、V$TRANSACTION ビューで特定できます。開始 SCN よりも小さい番号を持つトランザクションはすべて、表領域の停止とその後の読取り専用状態になるのを妨げている可能性があります。

```
SELECT SES_ADDR, START_SCNB
    FROM V$TRANSACTION
    ORDER BY START_SCNB;

SES_ADDR START_SCNB
-------- ----------
800352A0       3621   --> waiting on this txn
80035A50       3623   --> waiting on this txn
80034AF0       3628   --> this is the ALTER TABLESPACE statement
80037910       3629   --> don’t care about this txn
```

表領域を読取り専用にした後は、その表領域のバックアップをただちに作成することをお薦めします。表領域は読取り専用になっているかぎり変更できないため、それ以後のバックアップは不要です。

関連項目： 読取り専用データ・ファイルを含むデータベースのリカバリの詳細は、バックアップおよびリカバリ計画に応じて次のいずれかのマニュアルを参照してください。

- 『Oracle9*i* ユーザー管理バックアップおよびリカバリ・ガイド』
- 『Oracle9*i* Recovery Manager ユーザーズ・ガイド』

読取り専用表領域を書込み可能にする方法

表領域を書込み可能に変更するには、ALTER TABLESPACE 文で READ WRITE キーワードを指定します。そのためには、ALTER TABLESPACE または MANAGE TABLESPACE システム権限が必要です。

表領域を読取り / 書込み用にするには、前提条件として、表領域のみでなく、そのすべてのデータ・ファイルをオンライン化する必要があります。データ・ファイルをオンライン化するには、ALTER DATABASE 文の DATAFILE ... ONLINE 句を使用します。データ・ファイルの現行の状態を確認するには、V$DATAFILE ビューを使用します。

次の文は、flights 表領域を書込み可能にします。

```
ALTER TABLESPACE flights READ WRITE;
```

読取り専用表領域を書込み可能に変更すると、データ・ファイルの制御ファイル・エントリが更新されるため、読取り専用バージョンのデータ・ファイルをリカバリの開始点として使用できます。

WORM デバイスでの読取り専用表領域の作成

CD-ROM または WORM デバイスに読取り専用表領域を作成する手順は、次のとおりです。

1. 別のデバイスに書込み可能表領域を作成します。その表領域に属するオブジェクトを作成して、データを挿入します。
2. 表領域を読取り専用に変更します。
3. 表領域のデータ・ファイルを WORM デバイスにコピーします。ファイルをコピーするには、オペレーティング・システムのコマンドを使用します。
4. 表領域をオフライン化します。
5. データ・ファイルの名前を、WORM デバイスにコピーしたファイルと一致するように名前変更します。これには、RENAME DATAFILE 句を指定した ALTER TABLESPACE 文を使用します。データ・ファイルの名前を変更すると、制御ファイルに記述されているこれらのファイルの名前も変更されます。
6. 表領域をオンライン化します。

読取り専用表領域内にあるデータ・ファイルのオープンの遅延

大規模データベースのほとんどが、アクセス速度の遅いデバイスや階層形式の記憶デバイス上にある読取り専用表領域に格納されている場合は、READ_ONLY_OPEN_DELAYED 初期化パラメータを TRUE に設定することを検討する必要があります。これにより、読取り専用表領域内のデータ・ファイルは、そこに格納されたデータの読取り試行時に初めてアクセスされるため、データベースのオープンなど、特定の操作が高速になります。

READ_ONLY_OPEN_DELAYED=TRUE に設定すると、次のような副次的な影響があります。

- オープン時に、読取り専用の欠落ファイルや不良ファイルが検出されません。これらのファイルは、アクセス試行時にのみ検出されます。
- ALTER SYSTEM CHECK DATAFILES では、読取り専用ファイルはチェックされません。
- ALTER TABLESPACE ... ONLINE および ALTER DATABASE DATAFILE ... ONLINE では、読取り専用ファイルはチェックされません。最初のアクセス時にのみチェックされます。
- V$RECOVER_FILE、V$BACKUP および V$DATAFILE_HEADER は、読取り専用ファイルにアクセスしません。読取り専用ファイルは結果リスト上に「DELAYED OPEN」というエラーで示され、他の列の値は 0（ゼロ）になります。
- V$DATAFILE は読取り専用ファイルにアクセスしません。読取り専用ファイルにはサイズ「0」がリストされます。
- V$RECOVER_LOG は読取り専用ファイルにアクセスしません。リカバリに必要な可能性があるログは、リストに追加されません。
- ALTER DATABASE NOARCHIVELOG は読取り専用ファイルにアクセスしません。リカバリが必要な読取り専用ファイルがある場合でも、処理が継続します。

注意：

- RECOVER DATABASE および ALTER DATABASE OPEN RESETLOGS は、パラメータ値に関係なく、すべての読取り専用データ・ファイルに引き続きアクセスします。これらの操作で読取り専用ファイルへのアクセスを回避する場合は、該当ファイルをオフライン化する必要があります。
- バックアップ制御ファイルを使用すると、一部のファイルの読取り専用状態が不正確になる場合があります。これにより、これらの操作の一部で予期しない結果が返されることがあります。この状況には注意が必要です。

表領域の削除

表領域とその内容が不要になった場合は、その表領域と内容（表領域に含まれるセグメント）をデータベースから削除できます。Oracle データベースでは、どの表領域でも削除できます（ただし SYSTEM 表領域は除く）。表領域を削除するには、DROP TABLESPACE システム権限が必要です。

注意： 削除された表領域のデータはリカバリできません。そのため、削除しようとしている表領域に含まれているデータはすべて、将来的に必要のないことを確かめてください。また、表領域をデータベースから削除する直前および直後に、データベースの完全バックアップを作成する必要があります。表領域を誤って削除した場合、または表領域を削除した後にデータベースで問題が発生した場合、データベースをリカバリできるように、必ずバックアップを作成することをお薦めします。

表領域を削除すると、対応付けられたデータベースの制御ファイル中のファイル・ポインタのみが削除されます。必要に応じて、削除された表領域を構成していたオペレーティング・システム・ファイル（データ・ファイル）を削除するように Oracle に指示することもできます。表領域の削除と同時にデータ・ファイルを削除するように Oracle に指示しない場合は、後でオペレーティング・システムの適切なコマンドを使用して削除する必要があります。

アクティブなセグメントを含む表領域は削除できません。たとえば、表領域内の表が現在使用されている場合、またはアクティブなロールバック・セグメントが表領域に含まれている場合、その表領域は削除できません。表領域はオンラインでもオフラインでもかまいませんが、削除する前にオフラインにすることをお薦めします。

表領域を削除するには、DROP TABLESPACE 文を使用します。次の文は、users 表領域を、その中のセグメントも含めて削除します。

```
DROP TABLESPACE users INCLUDING CONTENTS;
```

表領域が空の場合（表、ビューまたは他の構造が格納されていない場合）は、INCLUDING CONTENTS オプションを指定する必要はありません。CASCADE CONSTRAINTS オプションを使用すると、表領域内の表の主キーと一意キーを参照する別の表領域の表から、すべての参照整合性制約を削除できます。

表領域の削除と同時に表領域に対応付けられたデータ・ファイルを削除するには、INCLUDING CONTENTS AND DATAFILES 句を使用します。次の文は、users 表領域とそれに対応付けられているデータ・ファイルを削除します。

```
DROP TABLESPACE users INCLUDING CONTENTS AND DATAFILES;
```

アラート・ファイルには、削除された各データ・ファイルのメッセージが書き込まれます。オペレーティング・システム・エラーによってファイルが削除されなかった場合でも DROP TABLESPACE 文は正常終了しますが、エラーを示すメッセージがアラート・ファイルに書き込まれます。

ローカル管理表領域の問題の診断と修復

注意： DBMS_SPACE_ADMIN パッケージを使用すると、管理者はローカル管理表領域の欠陥を診断および修復できます。このパッケージは、ディクショナリ管理表領域に関する欠陥の診断および修復には使用できません。

このパッケージには、ディクショナリ管理表領域とローカル管理表領域の間で移行するためのプロシージャも用意されています。

DBMS_SPACE_ADMIN パッケージには、次のプロシージャが含まれています。

プロシージャ	説明
SEGMENT_VERIFY	セグメントのエクステント・マップの整合性を検証します。
SEGMENT_CORRUPT	適切なエラー・リカバリを実行できるように、セグメントに破損または有効マークを付けます。ローカル管理の SYSTEM 表領域には使用できません。
SEGMENT_DROP_CORRUPT	現在、破損マークが設定されているセグメントを削除します（領域は再生しません）。ローカル管理の SYSTEM 表領域には使用できません。
SEGMENT_DUMP	特定セグメントのセグメント・ヘッダーとエクステント・マップをダンプします。
TABLESPACE_VERIFY	表領域のセグメントのビットマップとエクステント・マップが同期しているかどうかを検証します。
TABLESPACE_REBUILD_BITMAPS	適切なビットマップを再作成します。ローカル管理の SYSTEM 表領域には使用できません。
TABLESPACE_FIX_BITMAPS	適切なデータ・ブロック・アドレス範囲（エクステント）にビットマップ内で使用可能マークまたは使用済みマークを付けます。ローカル管理の SYSTEM 表領域には使用できません。
TABLESPACE_REBUILD_QUOTAS	特定の表領域の割当て制限を再作成します。
TABLESPACE_MIGRATE_FROM_LOCAL	ローカル管理表領域をディクショナリ管理表領域に移行します。ローカル管理の SYSTEM 表領域からディクショナリ管理の SYSTEM 表領域への移行には使用できません。

プロシージャ	説明
TABLESPACE_MIGRATE_TO_LOCAL	表領域をディクショナリ管理形式からローカル管理形式に移行します。
TABLESPACE_RELOCATE_BITMAPS	ビットマップを指定の保存先に再配置します。ローカル管理の SYSTEM 表領域には使用できません。
TABLESPACE_FIX_SEGMENT_STATES	移行が異常終了した表領域内のセグメントの状態を修正します。

次の使用例では、DBMS_SPACE_ADMIN パッケージを使用して問題を診断し、解決できる代表的な状況について説明します。

注意： 前述の一部のプロシージャは、正しく使用しないとデータが消失してリカバリ不能になる場合があります。これらのプロシージャに不明な点がある場合は、オラクル社カスタマ・サポート・センターと共同で作業を行ってください。

関連項目： DBMS_SPACE_ADMIN パッケージの詳細は、『Oracle9*i* 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。

使用例 1: 割当て済みブロックが空き（オーバーラップなし）とマークされているときのビットマップの修復

TABLESPACE_VERIFY プロシージャの使用時に、ビットマップ内で「空き」マークが付いているブロックがセグメントに割り当てられたにもかかわらず、セグメント間のオーバーラップがレポートされていないことが検出された場合。

この使用例では、次のタスクを実行してください。

1. SEGMENT_DUMP プロシージャをコールして、管理者がそのセグメントに割り当てた範囲をダンプします。
2. 範囲ごとに、TABLESPACE_EXTENT_MAKE_USED オプションを指定して TABLESPACE_FIX_BITMAPS プロシージャをコールし、領域に使用済みのマークを付けます。
3. TABLESPACE_REBUILD_QUOTAS をコールして割当て制限を修復します。

使用例 2: 破損したセグメントの削除

ビットマップに「空き」マークが付いたセグメント・ブロックがあるため、セグメントを削除できない場合。このセグメントには、自動的に「破損」マークが付けられます。

この使用例では、次のタスクを実行してください。

1. SEGMENT_VERIFY_EXTENTS_GLOBAL オプションを指定して SEGMENT_VERIFY プロシージャをコールします。オーバーラップがレポートされない場合は、手順 2 から 5 までを実行します。
2. SEGMENT_DUMP プロシージャをコールして、そのセグメントに割り当てられたデータ・ブロック・アドレス範囲をダンプします。
3. 範囲ごとに、TABLESPACE_EXTENT_MAKE_FREE オプションを指定して TABLESPACE_FIX_BITMAPS プロシージャをコールし、領域に「空き」のマークを付けます。
4. SEGMENT_DROP_CORRUPT をコールして SEG$ エントリを削除します。
5. TABLESPACE_REBUILD_QUOTAS をコールして割当て制限を修復します。

使用例 3: オーバーラップがレポートされたビットマップの修復

TABLESPACE_VERIFY プロシージャで、いくつかオーバーラップがレポートされる場合。前の内部エラーに基づいて、一部の実データを削除する必要があります。

この場合、表 t1 などの削除するオブジェクトを選択してから、次のタスクを実行します。

1. t1 がオーバーラップしているすべてのオブジェクトのリストを作成します。
2. 表 t1 を削除します。必要に応じて、SEGMENT_DROP_CORRUPT プロシージャをコールしてフォローアップします。
3. t1 がオーバーラップしていたすべてのオブジェクトに対して、SEGMENT_VERIFY プロシージャをコールします。必要に応じて、TABLESPACE_FIX_BITMAPS プロシージャをコールして該当するビットマップに使用済みを示すマークを付けます。
4. TABLESPACE_VERIFY プロシージャを再度実行し、問題が解決したかどうかを検証します。

使用例 4: ビットマップ・ブロックのメディア破損の訂正

ビットマップ・ブロックの集合にメディア破損がある場合。

この使用例では、次のタスクを実行してください。

1. すべてのビットマップ・ブロック、または 1 つしか破損していない場合はそのブロックに対して、TABLESPACE_REBUILD_BITMAPS プロシージャをコールします。
2. TABLESPACE_REBUILD_QUOTAS をコールして割当て制限を再作成します。
3. TABLESPACE_VERIFY プロシージャをコールして、ビットマップの整合性を検証します。

使用例 5: ディクショナリ管理表領域からローカル管理表領域への移行

ディクショナリ管理表領域をローカル管理表領域に移行する場合。TABLESPACE_MIGRATE_TO_LOCAL プロシージャを使用します。

データベースのブロック・サイズは 2KB、表領域 tbs_1 の既存のエクステント・サイズは 10、50 および 10,000 ブロック（それぞれ使用済み、使用済みおよび使用可能）とします。MINIMUM EXTENT 値は 20KB（10 ブロック）です。この使用例では、ビットマップの割当て単位をシステムに選択させることができます。MINIMUM EXTENT を超えない範囲の最大公約数であることから、値 10 ブロックが選択されます。

tbs_1 をローカル管理表領域に変換する文は、次のとおりです。

```
EXEC DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL ('tbs_1');
```

割当て単位のサイズを指定する場合は、必ずシステムによって計算される単位サイズの因数にします。そうしないと、エラー・メッセージが発行されます。

ローカル管理表領域への SYSTEM 表領域の移行

DBMS_SPACE_ADMIN パッケージを使用して、SYSTEM 表領域をディクショナリ管理からローカル管理に移行します。次の文は、この移行を実行します。

```
SQL> EXECUTE DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL('SYSTEM');
```

移行を実行する前に、次の条件を満たす必要があります。

- データベースのデフォルト一時表領域が SYSTEM ではないこと。
- ディクショナリ管理表領域にロールバック・セグメントがないこと。
- ローカル管理表領域に 1 つ以上のオンライン・ロールバック・セグメントがあるか、自動 UNDO 管理を使用している場合は、UNDO 表領域がオンラインになっていること。
- UNDO 領域を含む表領域（つまり、ロールバック・セグメントを含む表領域または UNDO 表領域）を除き、すべての表領域が読取り専用モードになっていること。
- データベースのコールド・バックアップがあること。
- システムが制限モードになっていること。

コールド・バックアップを除き、前述のすべての条件は TABLESPACE_MIGRATE_TO_LOCAL プロシージャにより施行されます。

注意： SYSTEM 表領域をローカル管理に移行すると、データベース内のディクショナリ管理表領域を READ WRITE モードにできなくなります。ディクショナリ管理表領域を READ-WRITE モードで使用できるようにする必要がある場合は、これらの表領域をローカル管理に移行してから、SYSTEM 表領域を移行することをお薦めします。

データベース間での表領域のトランスポート

ここでは、データベース間で表領域をトランスポートする方法について説明します。この項の内容は、次のとおりです。

- トランスポータブル表領域の概要
- 制限事項
- トランスポータブル表領域の互換性に関する注意事項
- データベース間で表領域をトランスポートする手順
- オブジェクトの動作
- トランスポータブル表領域の使用方法

トランスポータブル表領域の概要

注意： トランスポータブル表領域セットを生成するには、Oracle8*i* 以上の Enterprise Edition を使用する必要があります。ただし、Oracle8*i* 以上であれば、どのエディションでもトランスポータブル表領域セットを Oracle データベースにプラグインできます。

表領域の移動に伴うリリース・レベルでのデータベースの互換性の詳細は、11-36 ページの「トランスポータブル表領域の互換性に関する注意事項」を参照してください。

トランスポータブル表領域を使用すると、Oracle データベースのサブセットを移動して別の Oracle データベースにプラグインできます。これにより、実際にはデータベース間で表領域が移動されます。ディクショナリ管理またはローカル管理のどちらの表領域でもトランスポートできます。Oracle9*i* から、トランスポートする表領域をターゲット・データベースの標準ブロック・サイズと同じブロック・サイズにする必要はなくなりました。表領域をトランスポートする操作は、特に次の場合に使用します。

- OLTP システムからデータ・ウェアハウス・ステージング・システムにデータを移動する場合
- データ・ウェアハウスとデータ・マートをステージング・システムから更新する場合
- データ・マートを中央のデータ・ウェアハウスからロードする場合
- OLTP およびデータ・ウェアハウス・システムを効率的にアーカイブする場合
- 社内と外部のカスタマにデータを公開する場合
- 表領域の Point-in-Time リカバリ（TSPITR）を実行する場合

表領域をトランスポートすると、データ・ファイルをコピーして表領域の構造情報を統合するだけで済むため、同じデータのエクスポート / インポートやアンロード / ロードを使用するよりも、トランスポータブル表領域を使用してデータを移動するほうが高速です。また、トランスポータブル表領域を使用して索引データを移動することで、表データのインポート時やロード時に必要となる索引の再作成を回避することもできます。

関連項目：

- トランスポータブル表領域と、データ・マートおよびデータ・ウェアハウスでの使用の詳細は、『Oracle9*i* データベース概要』を参照してください。
- トランスポータブル表領域の異なる Oracle リリース間での互換性に関する問題については、『Oracle9*i* データベース移行ガイド』を参照してください。
- Recovery Manager を使用して Recovery Manager 表領域のバックアップを他のデータベースにトランスポートする方法の詳細は、『Oracle9*i* Recovery Manager ユーザーズ・ガイド』を参照してください。この方法を使用すると、元のデータベース内で表領域を読取り専用にする必要がありません。

制限事項

トランスポータブル表領域の使用計画を作成する場合は、次の制限事項に注意してください。

- ソース・データベースとターゲット・データベースは、同じハードウェア・プラットフォーム上に存在する必要があります。たとえば、Sun Solaris の Oracle データベース間、または Windows NT の Oracle データベース間では、表領域をトランスポートできます。ただし、Sun Solaris の Oracle データベースから Windows NT の Oracle データベースには、表領域をトランスポートできません。
- ソース・データベースとターゲット・データベースでは、同じキャラクタ・セットおよび各国語キャラクタ・セットを使用する必要があります。
- 同じ名前を持つ表領域がすでに存在しているターゲット・データベースには、表領域をトランスポートできません。
- トランスポータブル表領域は、次のものをサポートしていません。
  - マテリアライズド・ビュー / レプリケーション
  - ファンクション・ベース索引
  - 有効範囲付 `REF`
  - 複数受信者を持つ 8.0 互換のアドバンスト・キュー

トランスポータブル表領域の互換性に関する注意事項

トランスポータブル表領域の機能を使用するには、ソースおよびターゲットの両方のデータベースの COMPATIBLE 初期化パラメータを 8.1 以上に設定する必要があります。トランスポート対象の表領域のブロック・サイズがターゲット・データベースの標準ブロック・サイズと異なる場合は、ターゲット・データベースの COMPATIBLE 初期化パラメータを 9.0 以上に設定する必要があります。ソースおよびターゲットの両方のデータベースで同じリリースの Oracle を実行する必要はありません。Oracle では、トランスポータブル表領域セットがターゲット・データベースと互換性を持つことが保証されています。互換性がない場合は、プラグイン操作の開始時にエラーが通知されます。

古いリリースの Oracle（ただし、Oracle8*i* 以上）で実行しているデータベースから新しいリリースの Oracle（Oracle9*i* など）で実行しているデータベースに表領域を移動することは、常に可能です。

Oracle では、トランスポータブル表領域セットを作成するときに、ターゲット・データベースで実行する必要のある最も低い互換性レベルが計算されます。このレベルのことを、トランスポータブル・セットの互換性レベルと呼びます。トランスポータブル・セットをターゲット・データベースにプラグインするときに、トランスポータブル・セットの互換性レベルがターゲット・データベースの互換性レベルよりも高い場合は、エラーが通知されます。

データベース間で表領域をトランスポートする手順

表領域セットを移動またはコピーするには、次の手順を実行します。これらの手順については、この後の表領域 sales_1 と sales_2 のデータベース間でのトランスポート方法を示した項でさらに詳しく説明します。

1. 自己完結型の表領域セットの選択
2. トランスポータブル表領域セットの生成

   トランスポータブル表領域セットは、トランスポートされる表領域セットのデータ・ファイルと、そのセットの構造情報を含むファイルからなっています。

3. 表領域セットのトランスポート

   データ・ファイルとエクスポート・ファイルを、ターゲット・データベースにコピーします。このタスクには、フラット・ファイルをコピーする機能（オペレーティング・システムのコピー・ユーティリティ、ftp または CD での配布など）を使用できます。

4. 表領域のプラグイン

   インポート・ユーティリティを起動して、表領域セットをターゲット・データベースにプラグインします。

手順 1: 自己完結した表領域セットの選択

トランスポータブル・セット内のオブジェクトとセット外のオブジェクトとの間に、論理的または物理的な依存関係が存在することがあります。トランスポートできるのは、自己完結した表領域セットのみです。この場合、自己完結とは、表領域セット内から外部への参照がないことを意味します。次に、自己完結した表領域に違反する例を示します。

- 表領域セット内に、そのセットに含まれない表に関する索引が含まれている場合

  注意： 表に対応する索引が表領域セットの外部にある場合は、違反になりません。

- パーティション表の一部が表領域セットに含まれている場合

  コピーする表領域セットは、パーティション化した表のすべてのパーティションが含まれている状態、またはまったく含まれていない状態にしてください。パーティション表のサブセットをトランスポートする場合は、パーティションを表に変換する必要があります。

- 参照整合性制約がセット境界を越えて別の表を指している場合

  表領域セットをトランスポートするときには、参照整合性制約を含めるかどうかを選択できます。ただし、参照整合性制約を含めることによって、表領域セットの自己完結性に影響を与える場合があります。制約をトランスポートしなければ、その制約はポインタとは見なされません。

- 表領域セット内の表に、そのセットに含まれない LOB を指す LOB 列が含まれている場合

表領域セットが自己完結型かどうかを判断するには、オラクル社が提供するパッケージ DBMS_TTS の TRANSPORT_SET_CHECK プロシージャをコールします。このプロシージャを実行するには、EXECUTE_CATALOG_ROLE ロール（最初は SYS に付与されている）を付与されている必要があります。

DBMS_TTS パッケージをコールするときは、自己完結かどうかを調べるトランスポータブル・セットの表領域のリストを指定します。制約を含むかどうかを指定することもできます。**厳密または完全な完結**であるかを調べる場合は、TTS_FULL_CHECK パラメータを TRUE に設定する必要があります。

厳密または完全な完結のチェックは、トランスポータブル・セットから外部への参照のみではなく、外部からトランスポータブル・セットへの参照も捕捉する必要がある場合に実行します。依存オブジェクトがトランスポータブル・セットに完全に含まれているか、またはトランスポータブル・セットの外部にのみ存在することが必要な場合は、TSPITR を実行します。

たとえば、表 t を含んでいるが、その索引 i を含んでいない表領域に対して TSPITR を実行すると、トランスポート後に索引とデータの整合性がなくなるため、これは違反になります。完全完結チェックを実行することにより、トランスポータブル・セットからの依存関係またはトランスポータブル・セットへの依存関係がないことが保証されます。詳細は、

『Oracle9*i* ユーザー管理バックアップおよびリカバリ・ガイド』の TSPITR の例を参照してください。

注意： デフォルトでは、トランスポータブル表領域は、完全完結しているかどうかではなく自己完結しているかどうかがチェックされます。

ここでは、表領域 sales_1 および sales_2 が自己完結しているかどうかを、参照整合性制約を考慮して（TRUE を指定して）調べます。

```
EXECUTE dbms_tts.transport_set_check('sales_1,sales_2', TRUE);
```

この PL/SQL パッケージをコールした後に、TRANSPORT_SET_VIOLATIONS ビューからすべての違反を選択して表示できます。表領域セットが自己完結している場合、このビューは空になります。次の問合せは、2 つの違反がある場合を示しています。1 つ目は表領域セットの境界を越えている外部キー定数 dept_fk で、2 つ目は表領域セットに部分的に含まれているパーティション表 jim.sales です。

```
SELECT * FROM TRANSPORT_SET_VIOLATIONS;

VIOLATIONS
---------------------------------------------------------------------------
Constraint DEPT_FK between table JIM.EMP in tablespace SALES_1 and table
JIM.DEPT in tablespace OTHER
Partitioned table JIM.SALES is partially contained in the transportable set
```

これらの違反は、sales_1 および sales_2 をトランスポータブルにする前に解決する必要があります。次の手順で説明するように、整合性制約違反を回避するための選択肢の 1 つとして、整合性制約をエクスポートしない方法があります。

表領域セットにまたがるオブジェクト参照（REF など）は、違反とは見なされません。REF は、TRANSPORT_SET_CHECK ルーチンでチェックされません。参照先がない REF を含む表領域をデータベースにプラグインすると、その REF に従った問合せはユーザー・エラーを返します。

関連項目：

- REF の詳細は、『Oracle9*i* アプリケーション開発者ガイドー基礎編』を参照してください。
- DBMS_TTS パッケージの詳細は、『Oracle9*i* 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。
- TSPITR における DBMS_TTS パッケージの使用に関する詳細は、『Oracle9*i* 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。

手順 2: トランスポータブル表領域セットの生成

トランスポートする表領域セットが自己完結であることを確認した後に、次のタスクを実行してトランスポータブル表領域セットを生成します。

1. コピーするセット内のすべての表領域を読取り専用にします。

```
ALTER TABLESPACE sales_1 READ ONLY;
ALTER TABLESPACE sales_2 READ ONLY;
```

2. 次のように、エクスポート・ユーティリティを起動して、トランスポータブル・セットに含める表領域を指定します。

```
EXP TRANSPORT_TABLESPACE=y TABLESPACES=(sales_1,sales_2)
   TRIGGERS=y CONSTRAINTS=n GRANTS=n FILE=expdat.dmp
```

注意： エクスポート・ユーティリティを使用しますが、エクスポートするのは表領域のデータ・ディクショナリの構造情報（メタデータ）のみです。したがって、この操作は大規模な表領域の場合でもすぐに完了します。

プロンプトが表示された後、SYSDBA システム権限で SYS（またはその他の管理ユーザーとして）接続します。

```
CONNECT SYS/password AS SYSDBA
```

TABLESPACES は必ず指定します。この例では、次の指定も行います。

- トリガーをエクスポートします。

  TRIGGERS=y を設定すると、トリガーは妥当性チェックなしでエクスポートされます。無効なトリガーがあると、この後のインポート中にコンパイル・エラーが発生します。TRIGGERS=n を設定すると、トリガーはエクスポートされません。

- 参照整合性制約はエクスポートしません。
- 権限付与をエクスポートします。
- 作成する構造情報エクスポート・ファイルの名前を expdat.dmp にします。

TSPITR または厳密完結チェック付きのトランスポートを実行する場合は、次のようなコマンドを使用します。

```
EXP TRANSPORT_TABLESPACE=y TABLESPACES=(sales_1,sales_2)
   TTS_FULL_CHECK=Y FILE=expdat.dmp
```

トランスポートする表領域セットが自己完結していない場合は、エクスポートは失敗し、トランスポータブル・セットが自己完結していないことがわかります。その場合は、手順 1 に戻ってすべての違反を解決する必要があります。

> **関連項目：** エクスポート・ユーティリティの使用方法は、『Oracle9*i* データベース・ユーティリティ』を参照してください。

手順 3: 表領域セットのトランスポート

表領域のデータ・ファイルとエクスポート・ファイルの両方を、ターゲット・データベースからアクセスできる場所にトランスポートします。このタスクには、フラット・ファイルをコピーする機能（オペレーティング・システムのコピー・ユーティリティ、ftp または CD での配布など）を使用できます。

手順 4: 表領域セットのプラグイン

注意： 表領域セットを受け取るデータベースの標準ブロック・サイズと異なるブロック・サイズの表領域をトランスポートする場合は、最初に DB_*n*K_CACHE_SIZE 初期化パラメータ・エントリを受取り側データベースのパラメータ・ファイル内に設定する必要があります。

たとえば、ブロック・サイズが 8KB の表領域を標準ブロック・サイズが 4KB のデータベースにトランスポートする場合は、DB_8K_CACHE_SIZE 初期化パラメータ・エントリをパラメータ・ファイルに含める必要があります。このエントリがすでにパラメータ・ファイルに含まれている場合は、ALTER SYSTEM SET 文を使用してこのパラメータを設定できます。

DB_*n*K_CACHE_SIZE 初期化パラメータの値の指定方法は、『Oracle9*i* SQL リファレンス』を参照してください。

表領域セットをプラグインするには、次のタスクを実行します。

1. 次のインポート文を使用し、表領域をプラグインして構造情報を統合します。

```
IMP TRANSPORT_TABLESPACE=y FILE=expdat.dmp
    DATAFILES=('/db/sales_jan','/db/sales_feb',...)
    TABLESPACES=(sales_1,sales_2) TTS_OWNERS=(dcranney,jfee)
    FROMUSER=(dcranney,jfee) TOUSER=(smith,williams)
```

プロンプトが表示された後、SYSDBA システム権限で SYS（またはその他の管理ユーザーとして）接続します。

```
CONNECT SYS/password AS SYSDBA
```

この例では、次の指定を行います。

- TRANSPORT_TABLESPACE=y を指定して、表領域のトランスポートをエクスポート・ユーティリティに指示します。
- 表領域のメタデータが含まれるエクスポート・ファイルとして expdat.dmp を指定します。
- DATAFILES によって、トランスポートする表領域のデータ・ファイルを指定します。これは必ず指定する必要があります。
- 表領域名として sales_1 および sales_2 を指定します。

  TABLESPACES を指定すると、指定した表領域名がエクスポート・ファイル内の表領域名と比較されます。不一致があれば、エラーが返されます。指定しない場合は、エクスポート・ファイルから表領域名が抽出されます。
- TTS_OWNERS を指定して、表領域セット内のデータを所有する全ユーザーをリストします。

  TTS_OWNERS を指定すると、ユーザー名がエクスポート・ファイル内のユーザー名と比較されます。不一致があれば、エラーが返されます。指定しない場合は、エクスポート・ファイルから所有者名が抽出されます。
- データベース・オブジェクトの所有権を変更するために、FROMUSER および TOUSER を指定します。

  FROMUSER と TOUSER を指定しない場合は、すべてのデータベース・オブジェクト（表や索引など）が、ソース・データベース内と同じユーザーとして作成されます。これらのユーザーは、ターゲット・データベース内にすでに存在している必要があります。存在しない場合は、一部の必須ユーザーがターゲット・データベースに存在しないことを示すエラーが返されます。

  FROMUSER と TOUSER を使用すると、オブジェクトの所有者を変更できます。この例では、FROMUSER=(dcranney,jfee) および TOUSER=(smith, williams) を指定しています。ソース・データベース内で dcranney が所有している表領域セット内のオブジェクトは、表領域セットをプラグインした後のターゲット・データベース内では smith の所有となります。同様に、ソース・データベース内で jfee が所有しているオブジェクトは、ターゲット・データベース内では williams の所有となります。この場合、ターゲット・データベースにユーザー dcranney および jfee は存在しなくてもかまいませんが、ユーザー smith および williams は存在する必要があります。

この文が正常に実行されると、コピーするセット内のすべての表領域は読取り専用モードのままになります。インポート・ログをチェックして、エラーが発生しなかったかどうかを確認してください。

多数のデータ・ファイルを扱う場合、データ・ファイル名のリストを文の行で指定することは煩雑です。また、文の行制限を超える場合もあります。このような場合には、インポート・パラメータ・ファイルを使用できます。たとえば、次のようにしてインポート・ユーティリティを起動できます。

```
IMP PARFILE='par.f'
```

ファイル par.f の内容は、次のとおりです。

```
TRANSPORT_TABLESPACE=y
FILE=expdat.dmp
DATAFILES=('/db/sales_jan','/db/sales_feb',...)
TABLESPACES=(sales_1,sales_2)
TTS_OWNERS=(dcranney,jfee)
FROMUSER=(dcranney,jfee)
TOUSER=(smith,williams)
```

2. 必要に応じて、次のように、コピーした領域内の表領域を読取り / 書込みモードに戻します。

```
ALTER TABLESPACE sales_1 READ WRITE
ALTER TABLESPACE sales_1 READ WRITE
```

関連項目： インポート・ユーティリティの使用方法は、『Oracle9*i* データベース・ユーティリティ』を参照してください。

オブジェクトの動作

ほとんどのオブジェクトは、表領域内のデータや、それに対応付けられている構造情報に関係なく、異なるデータベースへのトランスポート後も正常に動作します。ただし、次のオブジェクトは例外です。

- ROWID
- REF
- 権限
- パーティション表
- オブジェクト
- アドバンスト・キュー
- 索引
- トリガー
- マテリアライズド・ビュー / レプリケーション

ROWID

データベースに（他のデータベースから）プラグインされた表領域が含まれている場合、そのデータベース内の ROWID は一意でなくなります。ROWID は、表内でのみ一意であることが保証されます。

REF

Oracle によって表領域セットが自己完結していると判断された場合、REF はチェックされません。その結果、プラグインされた表領域には、参照先のない REF が含まれることがあります。参照先のない REF に従った問合せでは、ユーザー・エラーが返されます。

権限

エクスポート時に GRANTS=y を指定すると、権限がトランスポートされます。インポート中には、一部の権限がインポートされないことがあります。たとえば、一定の権限を付与されているユーザーが存在しない場合や、特定の権限を付与されているロールが存在しない場合があります。

パーティション表

表領域セットにパーティション表のサブセットしか含まれていない場合、そのパーティション表はトランスポータブル表領域を使用して移動できません。表内のすべてのパーティションが表領域セットに含まれていることを確認するか、表領域セットをコピーする前にパーティションを表に変換する必要があります。ただし、パーティションを表に変換すると、パーティション表のグローバル索引が無効になるので注意してください。

ターゲット・データベースでは、そこに含まれる列と正確に一致する既存のパーティション表がある場合は、表をパーティションに変換できます。その表のすべてのパーティションが同じ外部データベースからのものであれば、変換操作が成功することは保証されています。まれなケースですが、同じ外部データベースからのものでない場合は、変換操作でデータ・オブジェクト番号の競合を示すエラーが返されることがあります。

表をパーティションに変換するときにデータ・オブジェクト番号の競合エラーが返される場合は、ALTER TABLE MOVE PARTITION 文を使用して、競合しているパーティションを移動できます。その後で、変換操作を再試行してください。

変換文で WITHOUT VALIDATION オプションを指定すると、構造情報のみが操作されるので即時に結果が返されます。ただし、パーティション内のデータがコピーされる場合があるので、パーティションの移動には時間がかかることがあります。

> **関連項目：** パーティション表のトランスポートの例については、11-45 ページの「データ・ウェアハウスのためのパーティションのトランスポートと連結」を参照してください。

オブジェクト

トランスポータブル表領域には、次のオブジェクトを含めることができます。

- 表
- 索引
- ドメイン索引
- ビットマップ索引
- 索引構成表
- LOB
- ネストした表
- VARRAY
- ユーザー定義型の列を含む表

表領域セットに BFILE へのポインタが含まれている場合は、その BFILE を移動して、ターゲット・データベース内に正しくディレクトリを設定する必要があります。

アドバンスト・キュー

Oracle アドバンスト・キューが複数受信者を持つ 8.0 互換キューでないかぎり、これらのキューはトランスポータブル表領域を使用して移動またはコピーできます。ターゲット・データベースにトランスポートしたキューは、最初は使用禁止になっています。トランスポートされた表領域をターゲット・データベース内で読取り / 書込みにしてから、組込みの PL/SQL ルーチン DBMS_AQADM.START_QUEUE を使用して起動すると、キューを使用可能にすることができます。

索引

通常の索引、ドメイン索引およびビットマップ索引をトランスポートできます。トランスポータブル・セットにパーティション表全体が含まれている場合は、そのパーティション表のグローバル索引もトランスポートできます。

ファンクション・ベース索引はサポートされていません。この種の索引が表領域に存在する場合は、表領域をトランスポートする前に削除する必要があります。

トリガー

トリガーは、妥当性チェックなしでエクスポートされます。つまり、Oracle では、トリガーがトランスポータブル・セット内のオブジェクトのみを参照しているかどうかは検証されません。無効なトリガーがあると、後続のインポート中にコンパイル・エラーになります。

マテリアライズド・ビュー / レプリケーション

マテリアライズド・ビューまたはレプリケーションの構造情報のトランスポートは、サポートされていません。表領域のトランスポート時には、表領域内の表に対応付けられたマテリアライズド・ビューまたはレプリケーションのメタデータはエクスポートされないため、ターゲット・データベースでは使用できません。

トランスポータブル表領域の使用方法

トランスポータブル表領域の用途は、次のとおりです。

データ・ウェアハウスのためのパーティションのトランスポートと連結

標準的な企業のデータ・ウェアハウスには、1 つ以上の大きいファクト表が含まれています。これらのファクト表は、企業データ・ウェアハウスを履歴データベースにするために、日付別にパーティション化されていることがあります。この場合、索引を作成すると、スター・クエリーを高速化できます。事実、オラクル社は、履歴データベースから最も古いパーティションを削除するたびにグローバル索引を再作成しなくても済むように、この種の履歴パーティション表についてローカル索引を作成することをお薦めしています。

たとえば、1 か月分のデータをデータ・ウェアハウスに毎月ロードする場合を考えます。データ・ウェアハウスには、sales という大型のファクト表があり、次の列が含まれています。

```
CREATE TABLE sales (invoice_no NUMBER,
   sale_year  INT NOT NULL,
   sale_month INT NOT NULL,
   sale_day   INT NOT NULL)
   PARTITION BY RANGE (sale_year, sale_month, sale_day)
     (partition jan98 VALUES LESS THAN (1998, 2, 1),
      partition feb98 VALUES LESS THAN (1998, 3, 1),
      partition mar98 VALUES LESS THAN (1998, 4, 1),
      partition apr98 VALUES LESS THAN (1998, 5, 1),
      partition may98 VALUES LESS THAN (1998, 6, 1),
      partition jun98 VALUES LESS THAN (1998, 7, 1));
```

次のようにして、ローカルの非同一キー索引を作成します。

```
CREATE INDEX sales_index ON sales(invoice_no) LOCAL;
```

最初は、すべてのパーティションは空で、同じデフォルト表領域にあります。パーティションを毎月 1 つ作成し、パーティション表 sales に連結する必要があります。

現在が 1998 年 7 月で、7 月分の売上データをパーティション表にロードするとします。ステージング・データベース内で、新しい表領域 ts_jul を作成します。また、その表領域内で、sales 表と正確に同じ列タイプを持つ表 jul_sales も作成します。表 jul_sales は、CREATE TABLE ... AS SELECT 文を使用して作成できます。jul_sales を作成して移入した後に、sales 表内のローカル索引と同じ列に索引を付けて、この表の索引 jul_

sale_index を作成することもできます。索引の作成後に、表領域 ts_jul をデータ・ウェアハウスにトランスポートします。

データ・ウェアハウスでは、7 月分の売上データ用の sales 表にパーティションを追加します。これにより、ローカルの非同一キー索引用にも 1 つのパーティションが作成されます。

```
ALTER TABLE sales ADD PARTITION jul98 VALUES LESS THAN (1998, 8, 1);
```

トランスポートされた表 jul_sales を新しいパーティションに変換して、表 sales に連結します。

```
ALTER TABLE sales EXCHANGE PARTITION jul98 WITH TABLE jul_sales
   INCLUDING INDEXES
   WITHOUT VALIDATION;
```

この文により、新しいデータがパーティション表に連結され、7 月分の売上データが新しいパーティション jul98 に格納されます。また、索引 jul_sale_index が sales 表のローカル索引のパーティションに変換されます。この文では、構造情報を操作するだけであり、データベースのポインタを切り替えれば済むので、結果は即時に返されます。新しいパーティション内のデータが旧パーティション内のデータとオーバーラップしないことがわかっている場合は、WITHOUT VALIDATION オプションを指定してください。このオプションを指定しなければ、新しいパーティションの範囲を検証するために、そこに含まれる新しいデータがすべて検査されます。

sales 表のすべてのパーティションが同じステージング・データベースから取り込まれる場合（ステージング・データベースが破壊されることはありません）、変換文は常に成功します。ただし、一般に、パーティション表のデータが異なるデータベースから取り込まれる場合は、変換操作が失敗する可能性があります。たとえば、sales の jan98 パーティションが同じステージング・データベースから取り込まれていなければ、前述の変換操作が失敗して次のエラーが返されることがあります。

```
ORA-19728: 表 JUL_SALES とパーティション JAN98（表 SALES）間で、データ・オブジェクト番号が競合しています。
```

この競合を解決するには、次の文を発行して、競合しているパーティションを移動します。

```
ALTER TABLE sales MOVE PARTITION jan98;
```

次に、変換操作を再試行してください。

交換が成功した後は、jul_sales と jul_sale_index を削除しても安全です（どちらも空になっています）。これで、7 月分の売上データはデータ・ウェアハウスに正常にロードされたことになります。

構造化データの CD での公開

トランスポータブル表領域を使用すると、構造化データを CD で公開できます。データ・プロバイダは、公開するデータを含む表領域をロードし、トランスポータブル・セットを生成し、トランスポータブル・セットを CD にコピーできます。これにより、この CD を配布できます。

顧客は、この CD を受け取って既存のデータベースにプラグインできます。CD からディスク記憶域にデータ・ファイルをコピーする必要がありません。たとえば、Windows NT マシンの D ドライブが CD ドライブであるとします。次のようにして、データ・ファイル catalog.f とエクスポート・ファイル expdat.dmp を含むトランスポータブル・セットをプラグインできます。

```
IMP TRANSPORT_TABLESPACE=y DATAFILES='D:¥catalog.f' FILE='D:¥expdat.dmp'
```

CD は、データベースの稼働中に取り出すことができます。この場合、その表領域への後続の問合せでは、CD 上のデータ・ファイルをオープンできないことを示すエラーが返されます。ただし、このデータベースの他の部分への操作は影響を受けません。CD をドライブに戻すと、表領域は再び読込み可能になります。

CD を取り出すのは、読取り専用表領域のデータ・ファイルを削除するのと同じことです。データベースを停止して再起動すると、削除されたデータ・ファイルが見つからず、データベースをオープンできないことを示すメッセージが表示されます（初期化パラメータ READ_ONLY_OPEN_DELAYED を TRUE に設定していない場合）。READ_ONLY_OPEN_DELAYED が TRUE に設定されている場合は、プラグインされている表領域を問い合せるときにのみ、このファイルが読み込まれます。したがって、CD 上の表領域をプラグインする場合は、その CD がデータベースに永続的に連結されないかぎり、常に READ_ONLY_OPEN_DELAYED 初期化パラメータを TRUE に設定しておく必要があります。

複数データベースで同じ表領域を読取り専用でマウントする方法

トランスポータブル表領域を使用すると、複数のデータベースで 1 つの表領域を読取り専用でマウントできます。これにより、データを別々のディスクに複製しなくても、異なるデータベースで同じデータを共有できます。表領域のデータ・ファイルは、どのデータベースからもアクセス可能にする必要があります。データベースの破損を回避するために、表領域はマウント先のすべてのデータベース内で読取り専用のままにしてください。

複数のデータベースで同じ表領域を読取り専用でマウントするには、次の 2 つの方法があります。

- 表領域をマウント先の各データベースにプラグインする方法。単一データベース内でトランスポータブル・セットを生成します。トランスポータブル・セット内のデータ・ファイルを、すべてのデータベースからアクセス可能なディスクに格納します。構造情報を各データベースにインポートします。
- あるデータベース内でトランスポータブル・セットを生成し、それを他のデータベースにプラグインする方法。この方法を使用する場合は、データ・ファイルがすでに共有ディスク上にあり、あるデータベースの既存の表領域に属していることが前提となりま

す。この表領域を読取り専用にしてトランスポータブル・セットを生成すると、データ・ファイルは共有ディスク上の同じ位置に残したまま、表領域を他のデータベースにプラグインできます。

このディスクを複数のコンピュータからアクセス可能にするには、いくつかの方法があります。まず、クラスタ・ファイル・システムまたは RAW ディスクのいずれかを使用できます。これは、Oracle9*i* Real Application Clusters で必要とされるためです。また、Oracle は共有ディスク上でこのタイプのデータ・ファイルしか読み込まないので、NFS を使用することも可能です。ただし、NFS の停止中にユーザーが共有表領域を問い合せると、NFS 操作がタイムアウトになるまでデータベースが停止することがあります。

後で、一部のデータベースから読取り専用表領域を削除できます。読取り専用表領域を削除しても、表領域のデータ・ファイルは変更されません。したがって、削除操作によって表領域が破損することはありません。表領域をマウントしているデータベースが 1 つしかない場合を除き、表領域は読取り / 書込み可能にしないでください。

トランスポータブル表領域を使用した履歴データのアーカイブ

トランスポータブル表領域セットは、任意の Oracle データベースにプラグインできる自己完結したファイル・セットなので、この章で説明するトランスポータブル表領域の手順を使用して、企業データ・ウェアハウスに旧 / 履歴データをアーカイブできます。

関連項目： 詳細は、『Oracle9*i* データ・ウェアハウス・ガイド』を参照してください。

トランスポータブル表領域を使用した TSPITR の実行

トランスポータブル表領域を使用して、表領域の TSPITR を実行できます。

関連項目： トランスポータブル表領域を使用して TSPITR を実行する方法は、『Oracle9*i* 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。

表領域情報の表示

次のデータ・ディクショナリ・ビューおよび動的パフォーマンス・ビューは、データベースの表領域に関して役立つ情報を提供します。

ビュー	説明
V$TABLESPACE	制御ファイルに記述されているすべての表領域の名前と番号
DBA_TABLESPACES、USER_TABLESPACES	すべての（またはユーザーがアクセス可能な）表領域の説明
DBA_SEGMENTS、USER_SEGMENTS	すべての（またはユーザーがアクセス可能な）表領域内のセグメントに関する情報
DBA_EXTENTS、USER_EXTENTS	すべての（またはユーザーがアクセス可能な）表領域内のデータ・エクステントに関する情報
DBA_FREE_SPACE、USER_FREE_SPACE	すべての（またはユーザーがアクセス可能な）表領域内の使用可能エクステントに関する情報
V$DATAFILE	所有する表領域の表領域番号など、すべてのデータ・ファイルに関する情報
V$TEMPFILE	所有する表領域の表領域番号など、すべての一時ファイルに関する情報
DBA_DATA_FILES	表領域に属するファイル（データ・ファイル）
DBA_TEMP_FILES	一時表領域に属するファイル（一時ファイル）
V$TEMP_EXTENT_MAP	ローカル管理の一時表領域すべての全エクステントに関する情報
V$TEMP_EXTENT_POOL	ローカル管理の一時表領域の場合、キャッシュされ、各インスタンスで使用されている一時領域の状態
V$TEMP_SPACE_HEADER	各一時ファイルの使用済み領域 / 空き領域
DBA_USERS	すべてのユーザーのデフォルト表領域と一時表領域
DBA_TS_QUOTAS	すべてのユーザーの表領域割当て制限
V$SORT_SEGMENT	特定インスタンス内のすべてのソート・セグメントに関する情報。このビューは、表領域が TEMPORARY タイプの場合にのみ更新されます。
V$SORT_USER	ユーザーおよび一時 / 永続表領域に使用される一時ソート領域

次に示すのは、これらのビューのごく一部の使用例です。

関連項目： これらのビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

表領域とデフォルト記憶域パラメータの表示例

データベースに含まれるすべての表領域の名前とデフォルト記憶域パラメータをすべて表示するには、DBA_TABLESPACES ビューに対して次の問合せを使用します。

```
SELECT TABLESPACE_NAME "TABLESPACE",
   INITIAL_EXTENT "INITIAL_EXT",
   NEXT_EXTENT "NEXT_EXT",
   MIN_EXTENTS "MIN_EXT",
   MAX_EXTENTS "MAX_EXT",
   PCT_INCREASE
   FROM DBA_TABLESPACES;

TABLESPACE  INITIAL_EXT  NEXT_EXT  MIN_EXT   MAX_EXT    PCT_INCREASE
----------  -----------  --------  -------   -------    ------------
RBS             1048576   1048576        2        40               0
SYSTEM           106496    106496        1        99               1
TEMP             106496    106496        1        99               0
TESTTBS           57344     16384        2        10               1
USERS             57344     57344        1        99               1
```

データ・ファイルとデータベースの対応する表領域の表示例

データ・ファイルの名前、サイズおよびデータベースの対応する表領域を表示するには、DBA_DATA_FILES ビューに対して次の問合せを入力します。

```
SELECT  FILE_NAME, BLOCKS, TABLESPACE_NAME
   FROM DBA_DATA_FILES;

FILE_NAME                                      BLOCKS  TABLESPACE_NAME
------------                                ----------  -------------------
/U02/ORACLE/IDDB3/RBS01.DBF                       1536  RBS
/U02/ORACLE/IDDB3/SYSTEM01.DBF                    6586  SYSTEM
/U02/ORACLE/IDDB3/TEMP01.DBF                      6400  TEMP
/U02/ORACLE/IDDB3/TESTTBS01.DBF                   6400  TESTTBS
/U02/ORACLE/IDDB3/USERS01.DBF                      384  USERS
```

各表領域の空き領域（エクステント）の統計の表示例

データベース内の各表領域について、使用可能エクステントと結合アクティビティの統計を生成するには、次の問合せを入力します。

```
SELECT TABLESPACE_NAME "TABLESPACE", FILE_ID,
   COUNT(*)    "PIECES",
   MAX(blocks) "MAXIMUM",
   MIN(blocks) "MINIMUM",
   AVG(blocks) "AVERAGE",
   SUM(blocks) "TOTAL"
   FROM DBA_FREE_SPACE
GROUP BY TABLESPACE_NAME, FILE_ID;

TABLESPACE    FILE_ID  PIECES   MAXIMUM    MINIMUM  AVERAGE    TOTAL
----------    -------  ------   -------    -------  -------   ------
RBS                 2       1       955        955      955      955
SYSTEM              1       1       119        119      119      119
TEMP                4       1      6399       6399     6399     6399
TESTTBS             5       5      6364          3     1278     6390
USERS               3       1       363        363      363      363
```

PIECES は表領域ファイル内の空き領域エクステント数を示し、MAXIMUM および MINIMUM はデータベース・ブロック内の領域で連続する最大領域と最小領域を示します。また、AVERAGE は空き領域があるエクステントの平均ブロック・サイズを示し、TOTAL は各表領域ファイル内の空き領域のブロック数を示します。新しいオブジェクトを作成しようとしているとき、またはセグメントを拡張予定であり、表領域に十分な領域があることを確かめるときに、この問合せを使用します。

12

データ・ファイルの管理

この章では、データ・ファイルの管理について説明します。この章の内容は、次のとおりです。

- データ・ファイルを管理するためのガイドライン
- データ・ファイルの作成および表領域への追加
- データ・ファイルのサイズ変更
- データ・ファイルの可用性の変更
- データ・ファイルの名前変更および再配置
- データ・ファイルの削除
- データ・ファイル内のデータ・ブロックの検証
- ファイルと物理デバイスのマッピング
- データ・ファイル情報の表示

関連項目： Oracle データベースによって作成および管理されるデータ・ファイルと一時ファイルの作成方法は、第 3 章「Oracle Managed Files の使用」を参照してください。

データ・ファイルを管理するためのガイドライン

データ・ファイルは、データベース内に存在するすべての論理構造のデータを格納する、オペレーティング・システムの物理ファイルです。データ・ファイルは、表領域ごとに明示的に作成する必要があります。Oracle は、絶対ファイル番号および相対ファイル番号という 2 つの関連するファイル番号を各データ・ファイルに割り当てます。これらは、データ・ファイルを一意に識別するために使用されます。これらの番号について、次の表で説明します。

ファイル番号のタイプ	説明
絶対	データベース内のデータ・ファイルを一意に識別します。Oracle の旧リリースでは、絶対ファイル番号を単に「ファイル番号」と呼んでいる場合があります。
相対	表領域内のデータ・ファイルを一意に識別します。小規模および中規模サイズのデータベースでは、多くの場合、相対ファイル番号と絶対ファイル番号は同じです。しかし、データベース内のデータ・ファイル数が一定のしきい値（通常は 1023）を超えている場合は、相対ファイル番号と絶対ファイル番号が異なります。

ファイル番号は、多くのデータ・ディクショナリ・ビューで表示されます。必要に応じてファイル名のかわりにファイル番号を使用し、SQL 文でデータ・ファイルまたは一時ファイルを識別できます。ファイル番号を使用する場合は、`V$DATAFILE` または `V$TEMPFILE` ビューの `FILE#` 列に表示されるファイル番号を指定します。このファイル番号は、`DBA_DATA_FILES` または `DBA_TEMP_FILES` ビューの `FILE_ID` 列にも表示されます。

ここでは、データ・ファイルの管理について説明します。この項の内容は、次のとおりです。

- データ・ファイル数の決定
- データ・ファイルのサイズ設定
- 適切なデータ・ファイルの配置
- REDO ログ・ファイルから分離したデータ・ファイルの格納

データ・ファイル数の決定

データベースの SYSTEM 表領域には、少なくとも 1 つのデータ・ファイルが必要です。小規模なシステムでは、データ・ファイルが 1 つの場合があります。データベースのデータ・ファイルの数を決める際に考慮すべきいくつかのガイドラインを次に示します。

DB_FILES 初期化パラメータの値の決定

DB_FILES 初期化パラメータは、Oracle インスタンスを起動するときに、データ・ファイル情報用に予約するシステム・グローバル領域（SGA）の容量を指定します。これにより、インスタンスで作成可能なデータ・ファイルの最大数が決まります。この制限が適用されるのは、インスタンスが存続する期間内です。DB_FILES の値は（初期化パラメータ設定を変更することで）変更できますが、新しい値はインスタンスをいったん停止して再起動するまでは有効になりません。

注意： DB_FILES のデフォルト値は、オペレーティング・システムによって異なります。

DB_FILES の値を決めるときは、次の点を考慮してください。

- DB_FILES の値が小さすぎる場合は、最初にデータベースを停止しないと、DB_FILES 制限を超えてデータ・ファイルを追加できません。
- DB_FILES の値が大きすぎると、メモリーが不必要に消費されます。

データ・ファイルを表領域に追加するときの制限事項

データ・ファイルを表領域に追加するときには、次の制限事項があります。

- ほとんどのオペレーティング・システムでは、1 つのプロセスで同時にオープンできるファイルの数に制限があります。オープン・ファイル数がオペレーティング・システム制限に達すると、それ以上データ・ファイルを作成できなくなります。
- オペレーティング・システムでは、データ・ファイルの数とサイズに制限があります。
- Oracle では、インスタンスによってオープンされる Oracle データベースのデータ・ファイルの最大数が制限されます。この制限はオペレーティング・システムによって異なります。
- DB_FILES 初期化パラメータで指定したデータ・ファイルの数を超えることはできません。
- CREATE DATABASE 文または CREATE CONTROLFILE 文を発行するときに、MAXDATAFILES パラメータによって制御ファイルのデータ・ファイル部分の初期サイズを指定します。ただし、新しく追加するファイルの番号が MAXDATAFILES より大きく DB_FILES 以下であれば、データ・ファイルのセクションにより多数のファイルを格納できるように、制御ファイルが自動的に拡張されます。

パフォーマンスへの影響の考慮

表領域、そして最終的にはデータベースを構成するデータ・ファイルの数は、パフォーマンスに影響を与える可能性があります。

Oracle では、オペレーティング・システムで定義されている制限よりも多くの数のデータ・ファイルをデータベース内に作成できます。Oracle の DBW*n* プロセスは、すべてのオンライン・データ・ファイルをオープンできます。Oracle には、オープン・ファイル記述子をキャッシュとして処理し、オープン・ファイル記述子の数がオペレーティング・システムで定義されている制限に達したときに自動的にファイルをクローズする機能があります。この機能は、パフォーマンスに悪影響を与えるおそれがあります。できれば、オープン・ファイル記述子に関するオペレーティング・システムの制限を調整して、それがデータベース内のオンライン・データ・ファイルの数より大きくなるようにしてください。

関連項目：

- オペレーティング・システムの制限の詳細は、オペレーティング・システム固有の Oracle マニュアルを参照してください。
- `CREATE DATABASE` 文または `CREATE CONTROLFILE` 文の `MAXDATAFILES` パラメータの詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

データ・ファイルのサイズ設定

最初のデータ・ファイル（オリジナルの `SYSTEM` 表領域内にあるデータ・ファイル）の大きさは、初期のデータ・ディクショナリとロールバック・セグメントを収容するために、最低でも 150MB 必要です。他の Oracle 製品をインストールする場合は、`SYSTEM` 表領域内に追加の領域が必要になる場合があります。これらの製品の領域要件に関する詳細は、それぞれのインストール手順を参照してください。

適切なデータ・ファイルの配置

表領域の位置は、その表領域を構成するデータ・ファイルの物理的な位置によって決まります。コンピュータのハードウェア資源を適切に使用してください。

たとえば、データベースを格納するためのディスク・ドライブが複数使用可能な場合は、競合の可能性のあるデータ・ファイルを別のディスクに配置することを検討してください。このようにすると、ユーザーが情報を問い合せるときに両方のディスク・ドライブが同時に動作し、同時にデータを取得できます。

REDO ログ・ファイルから分離したデータ・ファイルの格納

データ・ファイルは、データベースの REDO ログ・ファイルが格納されているディスク・ドライブに格納しないでください。データ・ファイルと REDO ログ・ファイルが同じディスク・ドライブに格納されていて、このディスク・ドライブで障害が発生すると、これらのファイルをデータベースのリカバリ手順で使用できなくなります。

REDO ログ・ファイルを多重化すると、全 REDO ログ・ファイルが失われる可能性が低くなるので、データ・ファイルを一部の REDO ログ・ファイルと同じドライブに格納できます。

データ・ファイルの作成および表領域への追加

表領域を作成するときは、データベース・オブジェクトの将来的なサイズを見積り、十分な数のデータ・ファイルを作成する必要があります。必要に応じて、後で表領域に割り当てられたディスク領域のすべての容量（その結果としてデータベースの容量）を大きくするために、データ・ファイルを作成して表領域に追加できます。可能であれば、データがすべてのデバイスに均等に配分されるように、データ・ファイルを複数のデバイスに作成してください。

データ・ファイルを作成して表領域を対応付けるには、次の表にリストされている文のいずれかを使用します。どの文でも、作成するデータ・ファイルのファイル仕様を指定するか、または Oracle Managed Files 機能を使用してデータベースが作成し管理するファイルを作成できます。表には、データ・ファイルの作成に使用する文の簡単な説明と、このマニュアル内に記載されている文の詳細説明への参照が示されています。

SQL 文	説明	詳細
CREATE TABLESPACE	表領域とそれを構成するデータ・ファイルを作成します。	「表領域の作成」11-4 ページ
CREATE TEMPORARY TABLESPACE	ローカル管理の一時表領域とそれを構成する一時ファイルを作成します。一時ファイルは特殊なデータ・ファイルです。	「ローカル管理の一時表領域の作成」11-12 ページ
ALTER TABLESPACE ... ADD DATAFILE	データ・ファイルを作成して表領域に追加します。	「ディクショナリ管理表領域の変更」11-10 ページ
ALTER TABLESPACE ... ADD TEMPFILE	一時ファイルを作成して一時表領域に追加します。	「ローカル管理の一時表領域の作成」11-12 ページ
CREATE DATABASE	データベースとそれに対応付けられたデータ・ファイルを作成します。	「Oracle データベースの手動作成」2-14 ページ

SQL 文	説明	詳細
ALTER DATABASE ... CREATE DATAFILE	古いデータ・ファイルにかわる空のデータ・ファイルを作成します。この文は、バックアップがない状態で失われたデータ・ファイルを再作成する場合に利用します。	このマニュアルでは説明していません。『Oracle9*i* 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。

表領域に新しいデータ・ファイルを追加するときにファイル名を完全に指定しないと、データ・ファイルは、オペレーティング・システムに応じてデフォルトのデータベース・ディレクトリまたはカレント・ディレクトリに作成されます。データ・ファイルには、常に完全修飾名を使用することをお薦めします。既存のファイルを再利用する場合以外は、新しいファイル名が他のファイルと競合しないことを確認してください。すでに削除済みの旧ファイルは上書きされます。

データ・ファイルを作成する文が失敗した場合は、作成されたオペレーティング・システム・ファイルがすべて削除されます。ただし、ファイル・システムや記憶域サブシステムで発生する多数の潜在的なエラーが原因で、オペレーティング・システムのコマンドを使用した手動でのファイル削除が必要になる場合があります。

データ・ファイルのサイズ変更

ここでは、データ・ファイルのサイズを変更する様々な方法について説明します。この項の内容は、次のとおりです。

- データ・ファイルの自動拡張機能の使用可能および使用禁止
- 手動によるデータ・ファイルのサイズ変更

データ・ファイルの自動拡張機能の使用可能および使用禁止

データベースに追加の領域が必要になった場合に、自動的にサイズを拡張するデータ・ファイルを作成できます。また、既存のデータ・ファイルを自動拡張するように変更することも可能です。ファイルのサイズは、指定されている最大値に達するまで、指定の増分値ずつ大きくなります。

データ・ファイルを自動的に拡張するように設定しておくと、次のような利点があります。

- 表領域で領域が足りなくなった場合に、管理者が即時に介入する必要性が減ります。
- エクステントの割当て失敗が原因でアプリケーションが停止することがなくなります。

データ・ファイルが自動的に拡張できるかどうか確認するには、DBA_DATA_FILES ビューを問い合せ、AUTOEXTENSIBLE 列を調べます。

自動ファイル拡張を指定するには、次の SQL 文を使用してデータ・ファイルを作成するときに AUTOEXTEND ON 句を指定します。

- CREATE DATABASE
- CREATE TABLESPACE
- ALTER TABLESPACE

既存のデータ・ファイルの自動ファイル拡張機能を使用可能または使用禁止にしたり、データ・ファイルのサイズを手動で変更するには、ALTER DATABASE 文を使用します。

次の例は、users 表領域に追加するデータ・ファイルの自動拡張機能を使用可能にします。

```
ALTER TABLESPACE users
    ADD DATAFILE '/u02/oracle/rbdb1/users03.dbf' SIZE 10M
      AUTOEXTEND ON
      NEXT 512K
      MAXSIZE 250M;
```

NEXT の値は、データ・ファイルの拡張時にこのファイルに追加される増分値の最小サイズです。MAXSIZE の値は、自動拡張可能なファイルの最大サイズです。

次の例は、データ・ファイルの自動拡張機能を使用禁止にします。

```
ALTER DATABASE DATAFILE '/u02/oracle/rbdb1/users03.dbf'
    AUTOEXTEND OFF;
```

関連項目： データ・ファイルを作成または変更するための SQL 文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

手動によるデータ・ファイルのサイズ変更

手動でデータ・ファイルのサイズを増減させるには、ALTER DATABASE 文を使用します。

データ・ファイルのサイズを変更できるため、データ・ファイルをさらに追加しなくてもデータベースに領域を追加できます。この機能は、データベースで許容されているデータ・ファイルの最大数に達することが懸念される場合に有効です。

また、データ・ファイルのサイズを手動で縮小することで、データベース内の未使用領域を再生できます。これは、領域要件の見積りの誤りを訂正する際に有効です。

次の例では、データ・ファイル /u02/oracle/rbdb1/stuff01.dbf が 250MB まで拡張されていることを想定しています。ただし、その表領域には現在小さなオブジェクトが格納されているので、データ・ファイルのサイズを縮小できます。

次の文は、データ・ファイル /u02/oracle/rbdb1/stuff01.dbf のサイズを縮小します。

```
ALTER DATABASE DATAFILE '/u02/oracle/rbdb1/stuff01.dbf'
   RESIZE 100M;
```

注意： 必ずしもファイルのサイズを指定した値まで縮小できるわけではありません。

データ・ファイルの可用性の変更

表領域の個々のデータ・ファイルまたは一時ファイルはオフライン化でき、同様にオンライン化できます。オフラインのデータ・ファイルはデータベースから使用できず、オンライン化されるまでアクセスできません。表領域の名前を指定するだけで、表領域を構成しているすべてのデータ・ファイルまたは一時ファイルをオフライン化またはオンライン化することもできます。

データ・ファイルの可用性の変更が必要な例として、データ・ファイルへの書込みに問題が発生し、データ・ファイルが自動的にオフライン化される場合があります。この場合は、後で問題を解決してから、データ・ファイルを手動でオンライン化できます。

読取り専用表領域のファイルも、読取り / 書込み表領域の場合と同様に、個別にオフライン化またはオンライン化できます。読取り専用表領域のデータ・ファイルをオンライン化すると、そのデータ・ファイルは読取り可能になります。このファイルに対応付けられた表領域が読取り / 書込み可能な状態に戻らないかぎり、このファイルには書き込めません。

データ・ファイルをオンライン化またはオフライン化するには、`ALTER DATABASE` システム権限が必要です。`ALTER TABLESPACE` 文を使用してすべてのデータ・ファイルまたは一時ファイルをオフライン化するには、`ALTER TABLESPACE` または `MANAGE TABLESPACE` システム権限が必要です。Oracle Real Application Clusters 環境では、データベースを排他モードでオープンする必要があります。

ここでは、データ・ファイルの可用性を変更する様々な方法について説明します。この項の内容は、次のとおりです。

- ARCHIVELOG モードでデータ・ファイルをオンライン化またはオフライン化する方法
- NOARCHIVELOG モードでデータ・ファイルをオフライン化する方法
- 表領域内のすべてのデータ・ファイルおよび一時ファイルの可用性の変更

注意： 表領域をオフライン化することによって、表領域内のすべてのデータ・ファイル（`SYSTEM` 表領域内のファイルを除く）を一時的に使用禁止にすることができます。表領域をオンラインに戻すために、表領域内のデータ・ファイルはすべてそのままにしておく必要があります。

表領域をオフラインにする方法の詳細は、11-20 ページの「表領域のオフライン化」を参照してください。

ARCHIVELOG モードでデータ・ファイルをオンライン化またはオフライン化する方法

個々のデータ・ファイルをオンライン化するには、DATAFILE 句を指定して ALTER DATABASE 文を発行します。次の文では、指定したデータ・ファイルがオンライン化されます。

```
ALTER DATABASE DATAFILE '/u02/oracle/rbdb1/stuff01.dbf' ONLINE;
```

これと同じファイルをオフライン化するには、次の文を発行します。

```
ALTER DATABASE DATAFILE '/u02/oracle/rbdb1/stuff01.dbf' OFFLINE;
```

注意： この形式の ALTER DATABASE 文を使用するには、データベースを ARCHIVELOG モードにしてください。NOARCHIVELOG モードでデータ・ファイルをオフライン化すると、ファイルが失われる可能性があります。ARCHIVELOG モードを使用することで、データ・ファイルの不慮の損失を防止できます。

NOARCHIVELOG モードでデータ・ファイルをオフライン化する方法

データベースが NOARCHIVELOG モードのときにデータ・ファイルをオフライン化するには、DATAFILE 句および OFFLINE DROP 句を指定して ALTER DATABASE 文を使用します。これにより、データ・ファイルをオフライン化し、即時に削除できます。たとえば、データベースが NOARCHIVELOG モードで運用されていて、データ・ファイルが一時セグメントのデータのみを含み、バックアップが作成されていない場合に利用できます。

次の文は、指定したデータ・ファイルをオフライン化します。

```
ALTER DATABASE DATAFILE '/u02/oracle/rbdb1/users03.dbf' OFFLINE DROP;
```

表領域内のすべてのデータ・ファイルおよび一時ファイルの可用性の変更

ALTER TABLESPACE 文で句を指定することにより、表領域内にあるすべてのデータ・ファイルまたは一時ファイルのオンラインまたはオフラインの状態を変更できます。具体的には、オンライン / オフラインの状態に影響を与える文として次のものがあります。

- ALTER TABLESPACE ... DATAFILE {ONLINE|OFFLINE}
- ALTER TABLESPACE ... TEMPFILE {ONLINE|OFFLINE}

入力が必要なのは表領域名のみであり、個々のデータ・ファイルや一時ファイルを入力する必要はありません。すべてのデータ・ファイルまたは一時ファイルが影響を受けますが、表領域そのもののオンライン / オフラインの状態は変わりません。

ほとんどの場合、データベースがマウントされていれば、オープンしていなくても、前述の ALTER TABLESPACE 文を発行できます。ただし、表領域がシステム表領域、UNDO 表領域またはデフォルト一時表領域である場合は、データベースをオープンしないでください。ALTER DATABASE DATAFILE 文および ALTER DATABASE TEMPFILE 文にも ONLINE/OFFLINE 句がありますが、これらの文では表領域のファイル名をすべて入力する必要があります。

この操作は表領域の可用性を変更する ALTER TABLESPACE ... ONLINE|OFFLINE 文とは操作が異なるため、構文も異なります。ALTER TABLESPACE 文は表領域だけでなくデータ・ファイルもオフラインにしますが、一時表領域または一時ファイルの状態を変更するためには使用できません。

データ・ファイルの名前変更および再配置

データ・ファイルの名前を変更して、それらの名前や位置を変更できます。次の項では、オプションの一部と使用可能な手順について説明します。

- 単一の表領域のデータ・ファイルの名前変更および再配置

  たとえば、データベースの残りの部分をオープンしたまま、*tablespace1* 内の *filename1* および *filename2* の名前を変更します。

- 複数の表領域のデータ・ファイルの名前変更および再配置

  たとえば、データベースをマウントし、クローズしたまま、*tablespace1* 内の *filename1* と *tablespace2* 内の *filename2* の名前を変更します。

注意： SYSTEM 表領域のデータ・ファイルを名前変更または再配置する場合、SYSTEM 表領域はオフライン化できないため、前述の後者のオプションを使用する必要があります。

これらの手順を使用してデータ・ファイルを名前変更または再配置すると、データベースの制御ファイルに記録されている、データ・ファイルへのポインタのみが変わります。これらの手順では、オペレーティング・システム・ファイルを物理的に名前変更したり、ファイルをオペレーティング・システム・レベルでコピーすることはありません。データ・ファイルの名前変更と再配置には、複数の手順が必要です。手順と例題をよく理解してから実行してください。

単一の表領域のデータ・ファイルの名前変更および再配置

ここでは、単一の表領域のデータ・ファイルを名前変更および再配置するための手順をいくつか示します。単一の表領域のデータ・ファイルの名前を変更するには、ALTER TABLESPACE システム権限が必要です。

単一の表領域のデータ・ファイルの名前を変更する手順

単一の表領域のデータ・ファイルの名前を変更する手順は、次のとおりです。

1. データ・ファイルを含む SYSTEM 表領域以外の表領域をオフライン化します。

   次に例を示します。

   ```
   ALTER TABLESPACE users OFFLINE NORMAL;
   ```

2. オペレーティング・システムを使用してデータ・ファイルの名前を変更します。

3. ALTER TABLESPACE 文に RENAME DATAFILE 句を指定して、データベース内のファイル名を変更します。

   たとえば、次の文はデータ・ファイル /u02/oracle/rbdb1/user1.dbf および /u02/oracle/rbdb1/user2.dbf をそれぞれ /u02/oracle/rbdb1/users01.dbf および /u02/oracle/rbdb1/users02.dbf に名前変更します。

   ```
   ALTER TABLESPACE users
       RENAME DATAFILE '/u02/oracle/rbdb1/user1.dbf',
                       '/u02/oracle/rbdb1/user2.dbf'
                    TO '/u02/oracle/rbdb1/users01.dbf',
                       '/u02/oracle/rbdb1/users02.dbf';
   ```

   新しいファイルはすでに存在している必要があります。この文ではファイルは作成されません。また、古いデータ・ファイルと新しいデータ・ファイルを正しく識別するために、必ず完全なファイル名（パスを含む）を指定してください。特に、古いデータ・ファイル名は、データ・ディクショナリの DBA_DATA_FILES ビューに表示されるとおり、正確に指定してください。

4. データベースのバックアップを作成します。データベースの構造を変更した後は、即時にデータベースの完全バックアップを実行してください。

単一の表領域のデータ・ファイルの名前変更および再配置

ここでは、データ・ファイルを再配置する手順の例を示します。

想定する条件は、次のとおりです。

- オープンしているデータベースに users という表領域が存在し、すべて同じディスク上に配置されたデータ・ファイルによって構成されています。
- users 表領域のデータ・ファイルを、別の分離されたディスク・ドライブに再配置します。

- 現在、オープンしているデータベースに管理者権限で接続しています。
- データベースの現行のバックアップは取得済みです。

次の手順を実行します。

1. 対象のデータ・ファイル名を確認します。

   次のようにデータ・ディクショナリ・ビュー DBA_DATA_FILES を問い合せると、users 表領域のデータ・ファイル名とそれぞれのサイズ（バイト単位）が表示されます。

```
SELECT FILE_NAME, BYTES FROM DBA_DATA_FILES
WHERE TABLESPACE_NAME = 'USERS';

FILE_NAME                                BYTES
---------------------------------------- ----------------
/U02/ORACLE/RBDB1/USERS01.DBF            102400000
/U02/ORACLE/RBDB1/USERS02.DBF            102400000
```

2. データ・ファイルが含まれる表領域をオフライン化するか、またはデータベースを停止してから再起動し、マウントしてクローズしたままにします。どちらの場合でも、表領域のデータ・ファイルはクローズされます。

3. オペレーティング・システムを使用し、データ・ファイルを新しい位置にコピーして名前変更します。

   注意： SQL*Plus の HOST コマンドを使用すると、オペレーティング・システム・コマンドを実行してファイルをコピーできます。

4. Oracle 内のデータ・ファイルの名前を変更します。

   users 表領域を構成するファイルのデータ・ファイル・ポインタは、対応付けられているデータベースの制御ファイルに記録されていますが、これらのポインタをこの時点で旧ファイル名から新ファイル名に変更する必要があります。

   表領域がオフラインになっていても、データベースがオープンしている場合は、ALTER TABLESPACE ... RENAME DATAFILE 文を使用します。データベースがマウントされているがクローズしている場合は、ALTER DATABASE ... RENAME FILE 文を使用します。

```
ALTER TABLESPACE users
    RENAME DATAFILE '/u02/oracle/rbdb1/users01.dbf',
                    '/u02/oracle/rbdb1/users02.dbf'
                 TO '/u03/oracle/rbdb1/users01.dbf',
                    '/u04/oracle/rbdb1/users02.dbf';
```

5. 表領域をオンライン化するか、またはデータベースをオープンします。

   users 表領域がオフラインで、データベースがオープンしている場合は、表領域をオンラインに戻します。データベースがマウントされているが、クローズしている場合は、データベースをオープンします。

6. データベースのバックアップを作成します。データベースの構造を変更した後は、即時にデータベースの完全バックアップを実行してください。

複数の表領域のデータ・ファイルの名前変更および再配置

1 つ以上の表領域のデータ・ファイルは、RENAME FILE 句を指定した ALTER DATABASE 文を使用して、名前変更および再配置できます。1 回の操作で複数の表領域のデータ・ファイルを名前変更または再配置する場合、あるいは SYSTEM 表領域のデータ・ファイルを名前変更または再配置する場合は、このオプションを使用する以外に方法はありません。データベースをオープンしておく必要がある場合は、前項で説明した手順を検討してください。

1 回の操作で複数の表領域のデータ・ファイルの名前を変更する、または SYSTEM 表領域のデータ・ファイルの名前を変更するには、ALTER DATABASE システム権限が必要です。

複数の表領域のデータ・ファイルの名前を変更する手順は、次のとおりです。

1. データベースがマウントされ、クローズされていることを確認します。

2. オペレーティング・システムで新しい位置と名前を指定して、名前変更するデータ・ファイルをコピーします。

3. ALTER DATABASE を使用して、データベースの制御ファイル内のファイル・ポインタの名前を変更します。

   たとえば、次の文はデータ・ファイル /u02/oracle/rbdb1/sort01.dbf および /u02/oracle/rbdb1/user3.dbf をそれぞれ /u02/oracle/rbdb1/temp01.dbf および /u02/oracle/rbdb1/users03.dbf に名前変更します。

   ```
   ALTER DATABASE
       RENAME FILE '/u02/oracle/rbdb1/sort01.dbf',
                   '/u02/oracle/rbdb1/user3.dbf'
                TO '/u02/oracle/rbdb1/temp01.dbf',
                   '/u02/oracle/rbdb1/users03.dbf;
   ```

   新しいファイルはすでに存在している必要があります。この文ではファイルは作成されません。また、古いデータ・ファイルと新しいデータ・ファイルを正しく識別するために、必ず完全なファイル名（パスを含む）を指定してください。特に、古いデータ・ファイル名は、データ・ディクショナリの DBA_DATA_FILES ビューに表示されるとおり、正確に指定してください。

4. データベースのバックアップを作成します。データベースの構造を変更した後は、即時にデータベースの完全バックアップを実行してください。

データ・ファイルの削除

データ・ファイルの削除専用の SQL 文はありません。データ・ファイルを削除するには、それを含む表領域を削除する必要があります。たとえば、表領域からデータ・ファイルを削除するには、次の手順で操作します。

1. 新しい表領域を作成します。
2. 古い表領域から新しい表領域にデータを移動します。
3. 古い表領域を削除します。

ただし、ALTER DATABASE 文を使用すると、一時ファイルを削除できます。次に例を示します。

```
ALTER DATABASE TEMPFILE '/u02/oracle/data/lmtemp02.dbf' DROP
     INCLUDING DATAFILES;
```

関連項目： 11-28 ページ「表領域の削除」

データ・ファイル内のデータ・ブロックの検証

チェックサムを使用してデータ・ブロックを検証するように Oracle を構成する場合は、初期化パラメータ DB_BLOCK_CHECKSUM を TRUE に設定します。このパラメータ値は、動的に変更するか、または初期化パラメータ・ファイルに設定できます。DB_BLOCK_CHECKSUM のデフォルト値は FALSE です。このパラメータの設定に関係なく、システム表領域のデータ・ブロックは常にチェックサムを使用して検証されます。

ブロックのチェックを使用可能にすると、Oracle がディスクに書き込まれる各ブロックのチェックサムを計算します。チェックサムは一時ブロックを含むすべてのデータ・ブロックについて計算されます。

DBW*n* プロセスが各ブロックのチェックサムを計算して、ブロック・ヘッダーにこのチェックサムを格納します。チェックサムはダイレクト・ローダによっても計算されます。

Oracle は次にデータ・ブロックを読み込むときに、チェックサムを使用してブロックの破損を検出します。破損が検出されると、メッセージ ORA-01578 が返され、破損に関する情報がトレース・ファイルに書き込まれます。

注意： DB_BLOCK_CHECKSUM を TRUE に設定すると、パフォーマンスのオーバーヘッドが発生することがあります。このパラメータを TRUE に設定するのは、データ破損の問題を診断するためにオラクル社カスタマ・サポート・センターの指示があった場合のみにしてください。

関連項目： チェックサムと DB_BLOCK_CHECKSUM 初期化パラメータの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

ファイルと物理デバイスのマッピング

データ・ファイルが単なるファイル・システム・ファイルである環境や、RAW デバイス上で直接作成される環境では、表領域と基礎となるデバイスとの関連付けを調べるのは比較的簡単です。Oracle には、ファイルとデバイスとのマッピングを提供する DBA_TABLESPACES、DBA_DATA_FILES および V$DATAFILE などのビューが用意されています。これらのマッピングをデバイス統計と併用して、I/O パフォーマンスを評価できます。

ただし、ホスト・ベースの論理ボリューム・マネージャ（LVM）と、Redundant Array of Independent Disks（RAID）機能を提供する洗練された記憶域サブシステムの導入により、ファイルからデバイスへのマッピングを判別するのが難しくなっています。最新ファイルがブラック・ボックスに隠れていると、そのファイルを判断するのが難くなるため、問題が発生します。この項では、この問題を解決するための Oracle のアプローチについて説明します。

この項の内容は、次のとおりです。

- Oracle のファイル・マッピング・インタフェースの概要
- Oracle のファイル・マッピング・インタフェースの動作
- Oracle のファイル・マッピング・インタフェースの使用
- ファイル・マッピングの例

注意： この項では、Oracle のファイル・マッピング・インタフェースの概要と、DBMS_STORAGE_MAP パッケージおよび動的パフォーマンス・ビューを使用してファイルから物理デバイスへのマッピングを公開する方法について説明します。この機能にアクセスするには、Oracle Enterprise Manager（OEM）を使用する方が簡単です。OEM には、ファイルから物理デバイスへのマッピング用に、使用しやすいグラフィカル・インタフェースが用意されています。

詳細は、OEM のドキュメント・セットを参照してください。

Oracle のファイル・マッピング・インタフェースの概要

I/O パフォーマンスを把握するには、ファイルが格納されている記憶域の階層の詳細を知る必要があります。Oracle には、ファイル、論理ビューの中間レイヤーおよび実際の物理デバイスのマッピング全体を表示するメカニズムが用意されています。この表示には、動的パフォーマンス・ビュー（V$ ビュー）のセットが使用されます。これらのビューを使用すると、ファイル・ブロックがあるディスクを正確に特定できます。

これらのビューを作成するために、ストレージ・ベンダーは特定の I/O スタック要素のマッピングを受け持つマッピング・ライブラリを提供する必要があります。Oracle は、Oracle バックグラウンド・プロセス FMON により起動される外部の非 Oracle プロセスを介して、これらのライブラリと通信します。FMON は、マッピング情報の管理を受け持ちます。Oracle には PL/SQL パッケージ `DBMS_STORAGE_MAP` が用意されており、このパッケージを使用して、マッピング・ビューを移入するマッピング操作を起動します。

Oracle のファイル・マッピング・インタフェースの動作

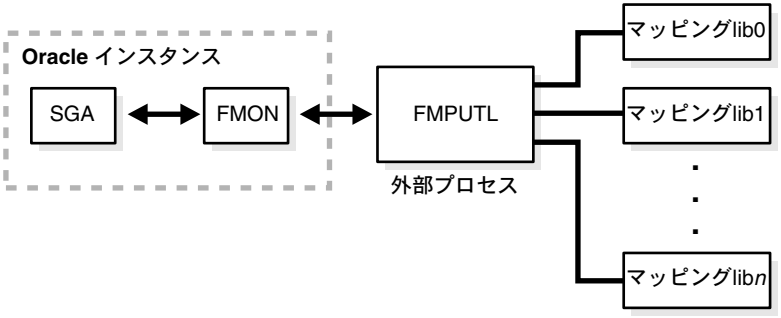
この項では、Oracle のファイル・マッピング・インタフェースの構成要素と、インタフェースの動作について説明します。この章の内容は、次のとおりです。

- ファイル・マッピングの構成要素
- マッピング構造
- マッピング構造の例
- 構成 ID

ファイル・マッピングの構成要素

次の図は、ファイル・マッピング・メカニズムの構成要素を示しています。

図 12-1 ファイル・マッピングの構成要素



ここでは、これらの構成要素と、各構成要素が連動してマッピング・ビューを移入する動作について説明します。

- FMON
- 外部プロセス（FMPUTL）
- マッピング・ライブラリ

FMON FMON は、FILE_MAPPING 初期化パラメータが TRUE に設定されている場合に、Oracle により起動されるバックグラウンド・プロセスです。FMON の役割は、次のとおりです。

- SGA に格納されるマッピング情報を作成します。この情報は、次の構造で構成されます。
  - ファイル
  - ファイル・システムのエクステント
  - 要素
  - 副要素

  これらの構造については、12-18 ページの「マッピング構造」を参照してください。
- 次の原因で変更が発生した場合にマッピング情報をリフレッシュします。
  - Oracle データ・ファイル（サイズ）の変更
  - データ・ファイルの追加または削除
  - 記憶域の構成変更（低頻度）
- マッピング情報をデータ・ディクショナリに保存して、起動操作と停止操作の間も持続する情報のビューを保持します。
- インスタンスの起動時にマッピング情報を SGA にリストアします。これにより、インスタンスを起動するたびにマッピング情報全体を再作成するという、高コストの操作が不要になります。

DBMS_STORAGE_MAP パッケージで起動されるプロシージャを使用すると、このマッピングを制御しやすくなります。

外部プロセス（FMPUTL） FMON は外部の非 Oracle プロセス FMPUTL を起動し、このプロセスはベンダーが提供するマッピング・ライブラリと直接通信します。このプロセスは、I/O スタックのすべてのレベルにマッピング・ライブラリが存在していれば、すべてのレベルを通じてマッピング情報を取得します。一部のプラットフォームでは、I/O マッピング・スタックの全レベルを通じてマッピングするにはルート権限が必要であるため、外部プロセスの SETUID ビットを ON に設定する必要があります。

この外部プロセスの役割は、マッピング・ライブラリを検出してアドレス空間に動的にロードすることです。

マッピング・ライブラリ Oracle はマッピング・ライブラリを使用して、特定のマッピング・ライブラリが所有する要素のマッピング情報を検出します。これらのマッピング・ライブラリを通じて、個々の I/O スタック要素に関する情報が伝達されます。この情報を使用して、ユーザーが問合せできる動的パフォーマンス・ビューが移入されます。

マッピングを完成するには、すべてのスタック・レベルにマッピング・ライブラリが存在する必要があり、各ライブラリが I/O マッピング・スタックの独自部分を所有できます。たとえば、VERITAS VxVM ライブラリは VERITAS ボリューム・マネージャに関連するスタック要素を所有し、EMC ライブラリは I/O マッピング・スタックのうちすべての EMC ストレージ固有レイヤーを所有します。

マッピング・ライブラリはベンダーから提供されます。ただし、現在、Oracle には EMC ストレージ用のマッピング・ライブラリが用意されています。データベース・サーバーに使用可能なマッピング・ライブラリは、特殊ファイル filemap.ora 内で識別されます。

マッピング構造

この項では、マッピング構造とその Oracle 表現について説明します。マッピング・ビューに表示される情報を解析するには、この情報を理解する必要があります。

マッピング情報を構成する基本構造は、次のとおりです。

- ファイル

  すべてのマッピング構造は、ファイル・サイズ、ファイルを構成するファイル・システムのエクステント数およびファイル・タイプなど、ファイルの属性セットを提供します。

- ファイル・システムのエクステント

  ファイル・システムのエクステントのマッピング構造では、1 つの要素にあるブロックの連続するチャンクが記述されます。これには、デバイス・オフセット、エクステント・サイズ、ファイル・オフセット、タイプ（データまたはパリティ）およびエクステントがある要素の名前が含まれます。

注意： ファイル・システムのエクステントは、Oracle のエクステントとは異なります。ファイル・システムのエクステントは、そのファイル・システムで管理されるデバイスに書き込まれる連続する物理データ・ブロックです。Oracle のエクステントは、表領域エクステントなど、Oracle で管理される論理構造です。

- 要素

  要素のマッピング構造は、I/O スタック内の記憶域コンポーネントを記述する抽象マッピング構造です。要素には、ミラー、ストライプ、パーティション、RAID5、連結要素およびディスクがあります。これらの構造は、マッピングのビルディング・ブロックです。

- 副要素

  副要素のマッピング構造では、I/O マッピング・スタック内のある要素と次の要素のリンクが記述されます。この構造には、副要素番号、サイズ、副要素が存在する要素の名前および要素のオフセットが含まれます。

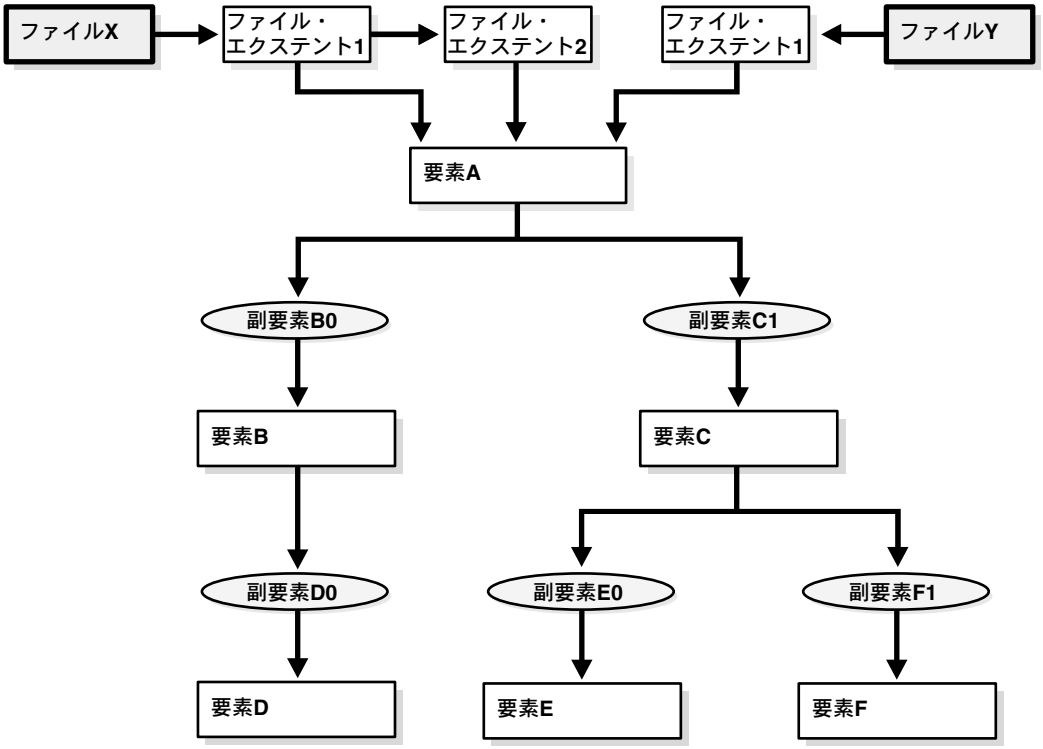
次の例は、これらのマッピング構造すべてを示しています。

マッピング構造の例

2 つのデータ・ファイル X および Y で構成される Oracle データベースを考えてみます。ファイル X と Y はどちらもボリューム A にマウントされたファイル・システム上に存在し、ファイル X は 2 つのエクステントで構成され、ファイル Y は 1 つのエクステントのみで構成されているとします。要素 A は 2 つの要素 B および C にまたがってストライプ化されています。要素 B は要素 D のパーティションで、要素 C は要素 E および F にまたがってミラー化されています。要素 D、E および F が物理ディスクであることに注意してください。副要素 B0 は親要素 A を要素 B に接続し、副要素 C1 は A を C に接続しています。

図 12-2 は、すべてのマッピング構造を示しています。

図 12-2 マッピング構造の図



この図が示すマッピング構造は、Oracle インスタンスのマッピング情報全体を記述するには十分であり、ファイル内の各論理ブロックを I/O スタック内の各レベルで 1 つ（またはミラー化の場合は 1 つ以上）の（要素名、要素オフセットの）タプルにマップしていることに注意してください。

構成 ID

構成 ID では、要素またはファイルに関連付けられたバージョン情報を取得します。ベンダーのライブラリには構成 ID が用意されており、変更があるたびに更新されます。構成 ID がなければ、Oracle ではマッピングに変更があったかどうかを指示できません。

構成 ID には、次の 2 種類があります。

- 永続

  この種の構成 ID は、インスタンスが停止されても持続します。

- 非永続

  この種の構成 ID は、インスタンスが停止すると持続しません。Oracle では、インスタンスが稼働している間にのみマッピング情報をリフレッシュできます。

Oracle のファイル・マッピング・インタフェースの使用

この項では、Oracle のファイル・マッピング・インタフェースの使用方法について説明します。この章の内容は、次のとおりです。

- ファイル・マッピングの有効化
- DBMS_STORAGE_MAP パッケージの使用
- ファイル・マッピング・ビューからの情報の取得

ファイル・マッピングの有効化

ファイル・マッピング機能を使用可能にする手順は、次のとおりです。

1. `$ORACLE_HOME/rdbms/filemap/etc` ディレクトリに有効な `filemap.ora` ファイルが存在することを確認します。

   注意： `filemap.ora` ファイルの形式と内容についてはこの項で説明しますが、これはあくまでも参考情報です。filemap.ora ファイルは、システムのインストール時に Oracle により作成されます。ベンダーから独自ライブラリが提供されるまで、filemap.ora ファイルのエントリは 1 つのみで、オラクル社が提供する EMC ライブラリに関するものです。このエントリをコメント解除して手動で変更する必要があるのは、EMC Symmetrix 配列が使用可能な場合のみです。

   `filemap.ora` ファイルは、使用可能なすべてのマッピング・ライブラリが記述されている構成ファイルです。FMON を使用するには、`filemap.ora` ファイルが存在し、マッピング・ライブラリへの有効なパスを指している必要があります。それ以外の場合は、正常に起動しません。

   ライブラリごとに、次の行を含める必要があります。

   `lib=`*`vendor_name`*`:`*`mapping_library_path`*

   各項目の意味は次のとおりです。

   – *`vendor_name`* には、EMC Symmetric ライブラリの場合は Oracle を指定します。

   – *`mapping_library_path`* には、マッピング・ライブラリのフルパスを指定します。

このファイル内のライブラリの順序がきわめて重要であることに注意してください。各ライブラリは、構成ファイル内での順序に基づいて問合せされます。

ファイル・マッピング・サービスは、使用可能なマッピング・ライブラリがなくても起動できます。filemap.ora ファイルは、空であっても存在する必要があります。この場合、マッピング・サービスは、新しいマッピング情報を検出できないという制約を伴います。この種の構成で許可されるのは、リストア操作と削除操作のみです。

2. FILE_MAPPING 初期化パラメータを TRUE に設定します。

```
FILE_MAPPING=TRUE
```

このパラメータを設定するためにインスタンスを停止する必要はありません。ALTER SYSTEM 文を使用して設定できます。

3. 適切な DBMS_STORAGE_MAP マッピング・プロシージャを起動します。次の 2 つの方法があります。

- コールド・スタートの使用例では、Oracle データベースが起動するのみで、まだマッピング操作は起動されていません。DBMS_STORAGE_MAP.MAP_ALL プロシージャを実行して、Oracle データベースに関連する I/O サブシステム全体のマッピング情報を作成します。
- ウォーム・スタートの使用例では、マッピング情報はすでに作成されており、DBMS_STORAGE_MAP.MAP_SAVE プロシージャを起動してマッピング情報をデータ・ディクショナリに保存するかどうかをオプションで選択できます。（このプロシージャは、デフォルトで DBMS_STORAGE_MAP.MAP_ALL() 内で起動します。）これにより、SGA 内のすべてのマッピング情報がディスクに強制的にフラッシュされます。

  データベースの再起動後に、DBMS_STORAGE_MAP.RESTORE() を使用してマッピング情報を SGA にリストアします。必要な場合は、DBMS_STORAGE_MAP.MAP_ALL() をコールしてマッピング情報をリフレッシュできます。

DBMS_STORAGE_MAP パッケージの使用

DBMS_STORAGE_MAP パッケージを使用すると、マッピング操作を制御できます。次の表に、使用可能な各種プロシージャを示します。

プロシージャ	用途
MAP_OBJECT	オブジェクト名、所有者およびタイプで識別される Oracle オブジェクトのマッピング情報を作成します。
MAP_ELEMENT	指定した要素のマッピング情報を作成します。
MAP_FILE	指定したファイル名のマッピング情報を作成します。
MAP_ALL	すべてのタイプの Oracle ファイル（アーカイブ・ログ以外）のマッピング情報全体を作成します。

プロシージャ	用途
DROP_ELEMENT	指定した要素のマッピング情報を削除します。
DROP_FILE	指定したファイル名のファイル・マッピング情報を削除します。
DROP_ALL	このインスタンスの SGA からすべてのマッピング情報を削除します。
SAVE	マッピング全体の再生成に必要な情報をデータ・ディクショナリに保存します。
RESTORE	マッピング情報全体をデータ・ディクショナリからインスタンスの共有メモリーにロードします。
LOCK_MAP	このインスタンスの SGA 内でマッピング情報をロックします。
UNLOCK_MAP	このインスタンスの SGA 内でマッピング情報のロックを解除します。

関連項目：

- DBMS_STORAGE_MAP パッケージの詳細は、『Oracle9*i* 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。
- DBMS_STORAGE_MAP パッケージの使用例は、12-25 ページの「ファイル・マッピングの例」を参照してください。

ファイル・マッピング・ビューからの情報の取得

DBMS_STORAGE_MAP パッケージにより生成されたマッピング情報は、動的パフォーマンス・ビューで取得されます。次の表に、これらのビューの概要を示します。

ビュー	説明
V$MAP_LIBRARY	外部プロセスにより動的にロードされたすべてのマッピング・ライブラリのリストが含まれています。
V$MAP_FILE	インスタンスの共有メモリーにあるすべてのファイル・マッピング構造のリストが含まれています。
V$MAP_FILE_EXTENT	インスタンスの共有メモリーにあるすべてのファイル・システム・エクステントのマッピング構造のリストが含まれています。
V$MAP_ELEMENT	インスタンスの SGA にあるすべての要素マッピング構造のリストが含まれています。
V$MAP_EXT_ELEMENT	すべての要素マッピングの補足情報が含まれています。

ビュー	説明
V$MAP_SUBELEMENT	インスタンスの共有メモリーにあるすべての副要素マッピング構造のリストが含まれています。
V$MAP_COMP_LIST	すべての要素マッピング構造の補足情報が含まれています。
V$MAP_FILE_IO_STACK	ファイルの記憶域コンテナの階層配列が一連の行として表示されます。各行は 1 つの階層レベルを表します。

関連項目： 動的パフォーマンス・ビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

ただし、DBMS_STORAGE_MAP.MAP_OBJECT プロシージャにより生成された情報は、グローバルな一時表 MAP_OBJECT に取得されます。この表には、オブジェクトの記憶域コンテナの階層配置が表示されます。表の各行は 1 つの階層レベルを表します。MAP_OBJECT 表の内容は、次のとおりです。

列	データ型	説明
OBJECT_NAME	VARCHAR2(2000)	オブジェクトの名前。
OBJECT_OWNER	VARCHAR2(2000)	オブジェクトの所有者。
OBJECT_TYPE	VARCHAR2(2000)	オブジェクト型。
FILE_MAP_IDX	NUMBER	ファイル索引（V$MAP_FILE 内の FILE_MAP_IDX に対応）。
DEPTH	NUMBER	I/O スタック内の要素の深さ。
ELEM_IDX	NUMBER	要素に対応する索引。
CU_SIZE	NUMBER	要素上に連続して存在する、ファイルの論理ブロックの連続するセット（単位 HKB）。
STRIDE	NUMBER	この要素上で連続しているファイルの連続単位（CU）間の HKB 数。RAID5 ファイルとストライプ化されたファイルに使用されます。
NUM_CU	NUMBER	ファイル内で STRIDE HKB で区切られた、この要素上で相互に隣接する連続単位の数。RAID5 では、パリティ・ストライプも連続単位数に含まれます。
ELEM_OFFSET	NUMBER	HKB 単位による要素オフセット。
FILE_OFFSET	NUMBER	ファイルの先頭から連続単位の先頭バイトまでのオフセット（HKB 単位）。

列	データ型	説明
DATA_TYPE	VARCHAR2(2000)	データ型（DATA、PARITY または DATA AND PARITY）。
PARITY_POS	NUMBER	パリティの桁。RAID5 のみ。このフィールドは、データ部分とパリティを区別するために必要です。
PARITY_PERIOD	NUMBER	パリティ間隔。RAID5 のみ。

ファイル・マッピングの例

次の例では、Oracle のファイル・マッピング機能の強力な機能について説明します。次のような機能があります。

- 特定のデバイスにまたがるすべての Oracle ファイルをマップする機能
- 特定のファイルを対応するデバイスにマップする機能
- I/O スタックのすべてのレベルでブロックを配分するなど、特定の Oracle オブジェクトをマップする機能

次の 2 つのデータ・ファイルで構成される Oracle インスタンスを考えてみます。

- t_db1.f
- t_db2.f

この 2 つのファイルは、VERITAS VxVM ホスト・ベースのストライプ化ボリューム /dev/vx/dsk/ipfdg/ipf-vol1 にマウントされた Solaris UFS ファイル・システム上で作成されており、このボリュームは EMC Symmetrix 配列から外部化された次のホスト・デバイスで構成されているとします。

- /dev/vx/rdmp/c2t1d0s2
- /dev/vx/rdmp/c2t1d1s2

次の例では、MAP_ALL() 操作を実行する必要があることに注意してください。

例 1: 1 つのデバイスにまたがるすべての Oracle ファイルのマッピング

次の問合せでは、/dev/vx/rdmp/c2t1d1s2 ホスト・デバイスに関連付けられたすべての Oracle ファイルが戻されます。

```
SELECT UNIQUE me.ELEM_NAME, mf.FILE_NAME
   FROM V$MAP_FILE_IO_STACK fs, V$MAP_FILE mf, V$MAP_ELEMENT me
   WHERE mf.FILE_MAP_IDX = fs.FILE_MAP_IDX
   AND me.ELEM_IDX = fs.ELEM_IDX
   AND me.ELEM_NAME = /dev/vx/rdmp/c2t1d1s2;
```

問合せ結果は次のとおりです。

```
ELEM_NAME                 FILE_NAME
------------------------  --------------------------------
/dev/vx/rdmp/c2t1d1s2     /oracle/dbs/t_db1.f
/dev/vx/rdmp/c2t1d1s2     /oracle/dbs/t_db2.f
```

例 2: ファイルから対応するデバイスへのマッピング

次の問合せでは、/oracle/dbs/t_db1.f データ・ファイルのトポロジ・グラフが表示されます。

```
WITH fv AS
  (SELECT FILE_MAP_IDX, FILE_NAME FROM V$MAP_FILE
   WHERE FILE_NAME = /oracle/dbs/t_db1.f)
SELECT fv.FILE_NAME, LPAD(' ', 4 * (LEVEL - 1)) || el.ELEM_NAME ELEM_NAME
   FROM V$MAP_SUBELEMENT sb, V$MAP_ELEMENT el, fv,
     (SELECT UNIQUE ELEM_IDX FROM V$MAP_FILE_IO_STACK io, fv
      WHERE io.FILE_MAP_IDX = fv.FILE_MAP_IDX) fs
   WHERE el.ELEM_IDX = sb.CHILD_IDX
   AND fs.ELEM_IDX = el.ELEM_IDX
   START WITH sb.PARENT_IDX IN
     (SELECT DISTINCT ELEM_IDX
      FROM V$MAP_FILE_EXTENT fe, fv
      WHERE fv.FILE_MAP_IDX = fe.FILE_MAP_IDX)
   CONNECT BY PRIOR sb.CHILD_IDX = sb.PARENT_IDX;
```

表示されるトポロジ・グラフは次のとおりです。

```
FILE_NAME                     ELEM_NAME
-----------------------       --------------------------------------------------
/oracle/dbs/t_db1.f           _sym_plex_/dev/vx/rdsk/ipfdg/ipf-vol1_-1_-1
/oracle/dbs/t_db1.f               _sym_subdisk_/dev/vx/rdsk/ipfdg/ipf-vol1_0_0_0
/oracle/dbs/t_db1.f                   /dev/vx/rdmp/c2t1d0s2
/oracle/dbs/t_db1.f                       _sym_symdev_000183600407_00C
/oracle/dbs/t_db1.f                           _sym_hyper_000183600407_00C_0
/oracle/dbs/t_db1.f                           _sym_hyper_000183600407_00C_1
/oracle/dbs/t_db1.f               _sym_subdisk_/dev/vx/rdsk/ipfdg/ipf-vol1_0_1_0
/oracle/dbs/t_db1.f                   /dev/vx/rdmp/c2t1d1s2
/oracle/dbs/t_db1.f                       _sym_symdev_000183600407_00D
/oracle/dbs/t_db1.f                           _sym_hyper_000183600407_00D_0
/oracle/dbs/t_db1.f                           _sym_hyper_000183600407_00D_1
```

例 3: Oracle オブジェクトのマッピング

この例では、scott.bonus 表について I/O スタックの全レベルにおけるブロックの分散を表示します。

次のように、最初に MAP_OBJECT() 操作を実行する必要があります。

```
EXECUTE DBMS_STORAGE_MAP.MAP_OBJECT('BONUS','SCOTT','TABLE');
```

問合せは次のとおりです。

```
SELECT io.OBJECT_NAME o_name, io.OBJECT_OWNER o_owner, io.OBJECT_TYPE o_type,
       mf.FILE_NAME, me.ELEM_NAME, io.DEPTH,
      (SUM(io.CU_SIZE * (io.NUM_CU - DECODE(io.PARITY_PERIOD, 0, 0,
                         TRUNC(io.NUM_CU / io.PARITY_PERIOD)))) / 2) o_size
   FROM MAP_OBJECT io, V$MAP_ELEMENT me, V$MAP_FILE mf
   WHERE io.OBJECT_NAME =  'BONUS'
   AND   io.OBJECT_OWNER = 'SCOTT'
   AND   io.OBJECT_TYPE =  'TABLE'
   AND   me.ELEM_IDX = io.ELEM_IDX
   AND   mf.FILE_MAP_IDX = io.FILE_MAP_IDX
   GROUP BY io.ELEM_IDX, io.FILE_MAP_IDX, me.ELEM_NAME, mf.FILE_NAME, io.DEPTH,
         io.OBJECT_NAME, io.OBJECT_OWNER, io.OBJECT_TYPE
   ORDER BY io.DEPTH;
```

問合せの結果は次のとおりです。o_size 列が KB 単位で表されていることに注意してください。

```
O_NAME O_OWNER O_TYPE  FILE_NAME            ELEM_NAME                      DEPTH   O_SIZE
------ ------- ------  -------------------  -----------------------------  ------  ------
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  /dev/vx/dsk/ipfdg/ipf-vol1          0      20
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  _sym_plex_/dev/vx/rdsk/ipf          1      20
                                            pdg/if-vol1_-1_-1
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  _sym_subdisk_/dev/vx/rdsk/          2      12
                                            ipfdg/ipf-vol1_0_1_0
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  _sym_subdisk_/dev/vx/rdsk/ipf       2       8
                                            dg/ipf-vol1_0_2_0
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  /dev/vx/rdmp/c2t1d1s2               3      12
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  /dev/vx/rdmp/c2t1d2s2               3       8
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  _sym_symdev_000183600407_00D        4      12
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  _sym_symdev_000183600407_00E        4       8
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  _sym_hyper_000183600407_00D_0       5      12
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  _sym_hyper_000183600407_00D_1       5      12
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  _sym_hyper_000183600407_00E_0       6       8
BONUS  SCOTT   TABLE   /oracle/dbs/t_db1.f  _sym_hyper_000183600407_00E_1       6       8
```

データ・ファイル情報の表示

次のデータ・ディクショナリ・ビューは、データベースのデータ・ファイルに関して役立つ情報を提供します。

ビュー	説明
DBA_DATA_FILES	属している表領域やファイル ID など、各データ・ファイルに関する記述情報が表示されます。ファイル ID を使用すると、他のビューと結合して詳細情報を得ることができます。
DBA_EXTENTS USER_EXTENTS	DBA ビューには、データベース内のすべてのセグメントを構成するエクステントが表示されます。エクステントを含むデータ・ファイルのファイル ID が含まれます。USER ビューには、現行ユーザーの所有するオブジェクトに属するセグメントのエクステントが表示されます。
DBA_FREE_SPACE USER_FREE_SPACE	DBA ビューには、すべての表領域の使用可能エクステントが表示されます。エクステントを含むデータ・ファイルのファイル ID が含まれます。USER ビューには、現行ユーザーからアクセス可能な表領域の使用可能エクステントが表示されます。
V$DATAFILE	制御ファイル内のデータ・ファイル情報が含まれます。
V$DATAFILE_HEADER	データ・ファイル・ヘッダーからの情報が含まれます。

ここでは、これらのビューの 1 つである V$DATAFILE の使用例を示します。

```
SELECT NAME,
    FILE#,
    STATUS,
    CHECKPOINT_CHANGE# "CHECKPOINT"
  FROM   V$DATAFILE;

NAME                                    FILE#     STATUS       CHECKPOINT
--------------------------------        -----     -------       ----------
/u01/oracle/rbdb1/system01.dbf              1     SYSTEM              3839
/u02/oracle/rbdb1/temp01.dbf                2     ONLINE              3782
/u02/oracle/rbdb1/users03.dbf               3     OFFLINE             3782
```

FILE# は各データ・ファイルのファイル番号を示します。データベースとともに作成される SYSTEM 表領域内の最初のデータ・ファイルは、常にファイル 1 になります。STATUS は、データ・ファイルに関する他の情報を示します。データ・ファイルが SYSTEM 表領域の一部である場合、このファイルの STATUS は SYSTEM になります（ただし、ファイルがリカバリを必要とする場合を除きます）。SYSTEM 表領域以外の表領域内のデータ・ファイルがオンラインの場合、このファイルの STATUS は ONLINE になります。SYSTEM 表領域以外の表領域内のデータ・ファイルがオフラインの場合、このファイルの STATUS は OFFLINE または

RECOVER になります。CHECKPOINT は、データ・ファイルの最新のチェックポイントに対して書き込まれた最後の SCN（システム変更番号）を示します。

関連項目： これらのビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

13

UNDO 領域の管理

この章では、UNDO 表領域の使用またはロールバック・セグメントの使用による UNDO 領域の管理方法について説明します。この章の内容は、次のとおりです。

- UNDO の概要
- UNDO 領域管理用モードの指定
- UNDO 表領域の管理
- ロールバック・セグメントの管理

関連項目：

- Oracle データベースによってデータ・ファイルが作成および管理される UNDO 表領域の作成方法は、第 3 章「Oracle Managed Files の使用」を参照してください。
- Oracle Real Application Clusters 環境における UNDO 領域の管理の詳細は、『Oracle9*i* Real Application Clusters 管理』を参照してください。

UNDO の概要

Oracle データベースでは、データベースの変更をロールバックまたは取り消すために使用する情報の管理方法が必要とされます。これらの情報は、主にコミットされる前のトランザクションの処理レコードから構成されます。Oracle では、これらのレコードを総称して **UNDO** と呼びます。

UNDO レコードは次の処理に使用されます。

- ROLLBACK 文を発行したときのトランザクションのロールバック
- データベースのリカバリ
- 読込み一貫性の提供

ロールバック文を発行すると、コミットされていないトランザクションによってデータベースに加えられた変更が、UNDO レコードを使用して取り消されます。データベース・リカバリ時は、REDO ログからデータ・ファイルに適用されたコミットされていない変更が、UNDO レコードを使用してすべて取り消されます。UNDO レコードは、あるユーザーがデータを変更しているときに同じデータに同時にアクセスしようとしている別のユーザーのために、そのデータの変更前のイメージを維持することによって読込み一貫性を提供します。

Oracle ではこれまで、UNDO を格納するためにロールバック・セグメントを使用していました。これらのロールバック・セグメントの領域管理は非常に複雑です。現在 Oracle では、ロールバック・セグメント領域の複雑な管理を不要にする別の UNDO 格納方法を用意しており、UNDO を上書きするまでの保存期間をデータベース管理者（DBA）が制御できるようになっています。この方法では、UNDO 表領域を使用しています。この章では、これらの 2 つの UNDO 領域管理方法について説明します。

2 つの方法を同じデータベース・インスタンスで使用することはできません。しかし、移行の目的から、たとえばロールバック・セグメントを使用しているデータベースで UNDO 表領域を作成したり、または UNDO 表領域を使用しているデータベースでロールバック・セグメントを削除することは可能です。ただし、別の UNDO 管理方法に切り替えて有効にするには、データベースを停止して再起動する必要があります。

注意： Oracle では、システム・トランザクションを実行するために必ず SYSTEM ロールバック・セグメントを使用します。SYSTEM ロールバック・セグメントは 1 つのみ存在し、CREATE DATABASE の実行時に自動的に作成され、インスタンスの起動時に常にオンライン化されています。SYSTEM ロールバック・セグメントを管理するためになんらかの操作を実行する必要はありません。

関連項目： UNDO および UNDO 領域の管理の詳細は、『Oracle9*i* データベース概要』を参照してください。

UNDO 領域管理用モードの指定

ロールバック・セグメントの方法を使用して UNDO 領域を管理する場合は、その操作を「手動 UNDO 管理モード」と呼びます。UNDO 表領域の方法を使用する場合は、「自動 UNDO 管理モード」と呼びます。このモードは、UNDO_MANAGEMENT 初期化パラメータを使用して、インスタンス起動時に決定します。

自動 UNDO 管理モードでのインスタンスの起動

次のように初期化パラメータを設定すると、STARTUP コマンドの実行時に自動 UNDO 管理モードでインスタンスが起動します。

```
UNDO_MANAGEMENT = AUTO
```

UNDO 表領域は、Oracle が UNDO レコードを格納するために使用可能である必要があります。デフォルトの UNDO 表領域は、データベースの作成時に作成されます。または、UNDO 表領域を明示的に作成することもできます。UNDO 表領域の作成方法については 13-6 ページの「UNDO 表領域の作成」を参照してください。

インスタンスが起動すると、Oracle は最初に使用可能になった UNDO 表領域を自動的に選択して使用します。使用可能な UNDO 表領域がない場合でもインスタンスは起動しますが、その場合は SYSTEM ロールバック・セグメントが使用されます。通常の環境では、これはお薦めできません。アラート・ファイルには、システムが UNDO 表領域のない状態で稼働していることを伝える警告メッセージが書き込まれます。

必要に応じて、Oracle インスタンスが起動時に特定の UNDO 表領域を使用するように指定することもできます。これには、UNDO_TABLESPACE 初期化パラメータを設定します。次に例を示します。

```
UNDO_TABLESPACE = undotbs_01
```

この場合、UNDO 表領域（この例では undotbs_01）がまだ作成されていない場合は、STARTUP コマンドは失敗します。Oracle Real Application Clusters 環境で UNDO_TABLESPACE パラメータを使用すると、インスタンスに特定の UNDO 表領域を割り当てることができます。

次に、自動 UNDO 管理モード用の初期化パラメータの概要を示します。

初期化パラメータ	説明
UNDO_MANAGEMENT	AUTO の場合は、自動 UNDO 管理モードを使用します。MANUAL の場合は、手動 UNDO 管理モードを使用します。
UNDO_TABLESPACE	使用する UNDO 表領域の名前を指定する動的パラメータ。
UNDO_RETENTION	UNDO の保存期間を指定する動的パラメータ。デフォルトは 900 秒です。
UNDO_SUPPRESS_ERRORS	TRUE の場合は、自動 UNDO 管理モードで操作中に手動 UNDO 管理 SQL 文を発行してもエラー・メッセージが出力されません。FALSE の場合は、エラー・メッセージが出力されます。これは動的パラメータです。

初期化パラメータ・ファイルに手動 UNDO 管理に関するパラメータが含まれていても、それらは無視されます。

UNDO 表領域の管理方法の詳細は、13-5 ページの「UNDO 表領域の管理」を参照してください。

関連項目： 自動 UNDO 管理モードで使用する初期化パラメータの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

手動 UNDO 管理モードでのインスタンスの起動

次のように初期化パラメータを設定すると、STARTUP コマンドの実行時に手動 UNDO 管理モードでインスタンスが起動します。

```
UNDO_MANAGEMENT = MANUAL
```

UNDO_MANAGEMENT 初期化パラメータを指定しない場合、インスタンスは手動 UNDO 管理モードで起動します。UNDO_TABLESPACE 初期化パラメータが指定されていても無視されます。手動 UNDO 管理モードでデータベースを稼働しようとしている DBA は、既存の初期化パラメータ・ファイルを変更せずにそのまま使用できます。

インスタンスが起動すると、次のいずれかで指定されている数のロールバック・セグメントがオンライン化されます。

- ROLLBACK_SEGMENTS 初期化パラメータ
- TRANSACTIONS および TRANSACTIONS_PER_ROLLBACK_SEGMENT 初期化パラメータ

次に、手動 UNDO 管理モードで指定できる初期化パラメータの概要を示します。

初期化パラメータ	説明
ROLLBACK_SEGMENTS	インスタンスの起動時に取得するロールバック・セグメントを指定します。
TRANSACTIONS	同時トランザクションの最大数を指定します。
TRANSACTIONS_PER_ROLLBACK_SEGMENT	各ロールバック・セグメントで処理する同時トランザクションの数を指定します。
MAX_ROLLBACK_SEGMENTS	インスタンスに対してオンライン化可能なロールバック・セグメントの最大数を指定します。

ロールバック・セグメントの管理方法の詳細は、13-13 ページの「ロールバック・セグメントの管理」を参照してください。

関連項目： 手動 UNDO 管理モードで使用する初期化パラメータの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

UNDO 表領域の管理

オラクル社は、できるだけ自動 UNDO 管理モードでの運用をお薦めします。自動 UNDO 管理モードでは、データベース・サーバーによって UNDO をより効率的に管理でき、手動 UNDO 管理ほど実装や管理が複雑ではありません。次の項では、UNDO 表領域の管理について説明します。

- UNDO 表領域の作成
- UNDO 表領域の変更
- UNDO 表領域の削除
- UNDO 表領域の切替え
- UNDO 領域に対するユーザー割当ての確立
- UNDO 情報の保存期間の指定
- UNDO 領域に関する情報の表示

関連項目： 次の項で説明する SQL 文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

UNDO 表領域の作成

UNDO 表領域の作成には 2 つの方法があります。1 つは、CREATE DATABASE 文の発行時に UNDO 表領域を作成する方法です。これは、データベースを新規作成中にインスタンスが自動 UNDO 管理モードで起動したとき（UNDO_MANAGEMENT = AUTO）に実行されます。もう 1 つは、既存のデータベースで使用する方法です。この場合は CREATE UNDO TABLESPACE 文を使用します。

UNDO 表領域にはデータベース・オブジェクトは作成できません。UNDO 表領域は、システムが管理している UNDO データ用に予約されています。

CREATE DATABASE を使用した UNDO 表領域の作成

CREATE DATABASE 文で UNDO TABLESPACE 句を使用すると、特定の UNDO 表領域を作成できます。ただし、この句は必須ではありません。

UNDO TABLESPACE 句を指定せずに CREATE DATABASE 文を自動 UNDO 管理モードで実行した場合は、SYS_UNDOTS という名前のデフォルトの UNDO 表領域が作成されます。この表領域は CREATE DATABASE 文が使用するデフォルトのファイル・セットから割り当てられ、その属性は Oracle によって決められます。初期サイズは 10MB で、自動拡張可能です。この方法による UNDO 表領域の作成は、UNDO 領域割当てのための仕様要件を持たないユーザーにのみお薦めします。

次の文は、CREATE DATABASE 文での UNDO TABLESPACE 句の使用例を示しています。ここでは、UNDO 表領域に undotbs_01 という名前を付け、/u01/oracle/rbdb1/undo0101.dbf という 1 つのデータ・ファイルを割り当てています。

```
CREATE DATABASE rbdb1
     CONTROLFILE REUSE
     .
     .
     .
     UNDO TABLESPACE undotbs_01 DATAFILE '/u01/oracle/rbdb1/undo0101.dbf';
```

CREATE DATABASE の実行中に UNDO 表領域を正常に作成できない場合は、CREATE DATABASE 操作全体が失敗します。データベース・ファイルをクリーン・アップし、エラーを訂正して、再度 CREATE DATABASE 操作を実行する必要があります。

CREATE UNDO TABLESPACE 文の使用

CREATE UNDO TABLESPACE 文は CREATE TABLESPACE 文とほぼ同じですが、UNDO キーワードを指定します。UNDO 表領域の属性のほとんどは Oracle が決定します。DBA が指定できるのは DATAFILE 句のみです。

この例では、undotbs_02 UNDO 表領域を作成しています。

```
CREATE UNDO TABLESPACE undotbs_02
     DATAFILE '/u01/oracle/rbdb1/undo0201.dbf' SIZE 2M REUSE AUTOEXTEND ON;
```

UNDO 表領域の変更

UNDO 表領域を変更するには、ALTER TABLESPACE 文を使用します。ただし、UNDO 表領域のほとんどはシステムが管理しているため、考慮が必要になるのは次の操作のみです。

- データ・ファイルの追加
- データ・ファイルの名前変更
- データ・ファイルのオンライン化またはオフライン化
- データ・ファイルのオープン状態のバックアップの開始または終了

DBA が変更可能な属性もこれらの属性のみです。

UNDO 表領域が領域不足の場合、または領域不足の発生を防止する場合は、さらにファイルを追加したり、既存のデータ・ファイルのサイズを変更できます。

次の例では、UNDO 表領域 undotbs_01 にデータ・ファイルを 1 つ追加しています。

```
ALTER TABLESPACE undotbs_01
     ADD DATAFILE '/u01/oracle/rbdb1/undo0102.dbf' AUTOEXTEND ON NEXT 1M
         MAXSIZE UNLIMITED;
```

ALTER DATABASE ... DATAFILE 文を使用すると、データ・ファイルのサイズを変更または拡張できます。

関連項目： 12-6 ページ「データ・ファイルのサイズ変更」

UNDO 表領域の削除

UNDO 表領域を削除するには、DROP TABLESPACE 文を使用します。次の例では、UNDO 表領域 undotbs_01 を削除しています。

```
DROP TABLESPACE undotbs_01;
```

UNDO 表領域は、現在どのインスタンスでも使用されていない場合にのみ削除できます。UNDO 表領域に処理中のトランザクションが含まれている場合（トランザクションが失敗してまだリカバリされていない場合など）、DROP TABLESPACE 文は失敗します。しかし、DROP TABLESPACE は、UNDO 表領域に期限切れでない（保存期間内である）ロールバック情報が含まれている場合でも UNDO 表領域を削除するため、既存の問合せでロールバック情報を必要とする場合は、UNDO 表領域を削除しないように注意する必要があります。

UNDO 表領域に対する DROP TABLESPACE は、DROP TABLESPACE ... INCLUDING CONTENTS と同じように動作します。つまり、UNDO 表領域の内容はすべて削除されます。

UNDO 表領域の切替え

ある UNDO 表領域から別の UNDO 表領域に切り替えることができます。UNDO_TABLESPACE 初期化パラメータは動的パラメータであるため、ALTER SYSTEM SET 文を使用して新しい UNDO 表領域を割り当てることができます。

次の文は、新しい UNDO 表領域に効率的に切り替えます。

```
ALTER SYSTEM SET UNDO_TABLESPACE = undotbs_02;
```

undotbs_01 が現行の UNDO 表領域であるとすると、このコマンドが正常に実行された後、インスタンスは undotbs_01 のかわりに undotbs_02 を UNDO 表領域として使用します。

切替え先の表領域が次のいずれかの条件を満たす場合はエラーがレポートされ、切替えは行われません。

- 表領域が存在しない場合
- 表領域が UNDO 表領域ではない場合
- 表領域が別のインスタンスによってすでに使用されている場合

切替え操作が実行されている間、データベースはオンラインであり、このコマンドの実行中でもユーザー・トランザクションを実行できます。切替え操作が正常に完了すると、切替え操作開始後に開始されたすべてのトランザクションが新しい UNDO 表領域内のトランザクション表に割り当てられます。

切替え操作は、古い UNDO 表領域内のトランザクションがコミットされるまで待機しません。古い UNDO 表領域内に未処理のトランザクションがある場合、古い UNDO 表領域は PENDING OFFLINE モード（状態）になります。このモードでは、既存のトランザクションは引き続き実行できますが、新しいユーザー・トランザクションの UNDO レコードをこの UNDO 表領域に格納することはできません。

UNDO 表領域は、切替え操作が正常に完了した後も、この PENDING OFFLINE モードのまま存在できます。PENDING OFFLINE の UNDO 表領域は、別のインスタンスが使用することも、削除することもできません。最終的に、すべてのアクティブなトランザクションがコミットされた後、UNDO 表領域は自動的に PENDING OFFLINE モードから OFFLINE モードに移行します。それ以降は、他のインスタンスが（Oracle Real Application Clusters 環境で）その UNDO 表領域を使用できます。

UNDO TABLESPACE のパラメータ値を「''」（2 つの一重引用符）に設定した場合、現行の UNDO 表領域が次の使用可能な UNDO 表領域に切り替わります。この文の使用には注意が必要です。使用可能な UNDO 表領域がない場合には、SYSTEM ロールバック・セグメントが使用され、アラート・ファイルにシステムが UNDO 表領域のない状態で稼働していることを伝える警告メッセージが書き込まれます。

次の例では、現行の UNDO 表領域の割当てを解除しています。

```
ALTER SYSTEM SET UNDO_TABLESPACE = '';
```

UNDO 領域に対するユーザー割当ての確立

Database Resource Manager を使用すると、UNDO 領域に対するユーザー割当てを確立できます。DBA は、Database Resource Manager のディレクティブである UNDO_POOL を使用して、ユーザーのグループ（リソース・コンシューマ・グループ）が消費する UNDO 表領域の量を制限できます。

UNDO プールは、コンシューマ・グループごとに指定できます。UNDO プールによって、コンシューマ・グループが生成できる UNDO の合計量が制御されます。コンシューマ・グループが生成する UNDO の合計量がその UNDO 制限を超えると、REDO を生成している現行の UPDATE トランザクションが終了します。コンシューマ・グループの他のメンバーは、UNDO 領域がプールから解放されるまで、新たに更新を実行できなくなります。

UNDO_POOL ディレクティブが明示的に定義されていないときは、ユーザーは無制限に UNDO 領域を使用できます。

関連項目： 第 27 章「Database Resource Manager の使用」

UNDO 情報の保存期間の指定

通常、コミットされたロールバック情報は、UNDO 領域が新しいトランザクションによって上書きされた時点で失われます。しかし、長時間実行の問合せ中にデータ・ブロックの変更を取り消し、変更前のイメージを生成する場合は、読込み一貫性を保証するために古いロールバック情報が必要になることがあります。初期化パラメータ UNDO_RETENTION は、保存するロールバック情報の量を明示的に指定するための手段を提供します。このパラメータを適切に設定すると、長時間実行の問合せの際に「スナップショットが古すぎます」というエラーが出力される危険はありません。

UNDO_RETENTION 初期化パラメータの設定

保存期間は、たとえば 500 秒というように秒単位で指定します。保存期間は永続的で、システム・クラッシュが発生しても継続します。つまり、インスタンスがクラッシュする前に生成された UNDO は、インスタンスを再起動したとしても、保存期間が経過するまで失われることはありません。インスタンスがリカバリすると、現行の UNDO_RETENTION 初期化パラメータの設定に基づいて、ロールバック情報が保存されます。

UNDO_RETENTION パラメータの初期値は、STARTUP プロセスが使用する初期化パラメータ・ファイルで設定できます。

```
UNDO_RETENTION = 10
```

UNDO_RETENTION パラメータの値は、ALTER SYSTEM コマンドを使用していつでも動的に変更できます。

```
ALTER SYSTEM SET UNDO_RETENTION = 5;
```

UNDO_RETENTION パラメータの変更は即時に反映されますが、その効果が表れるのは、現行の UNDO 表領域にアクティブ・トランザクションのための十分な領域がある場合のみです。アクティブ・トランザクションが UNDO 領域を必要とし、UNDO 表領域に使用可能な領域がない場合、システムは期限の切れていない UNDO 領域の再利用を開始します。このような処理によって、一部の問合せが「スナップショットが古すぎます」エラーで失敗する可能性があります。

UNDO_RETENTION 初期化パラメータが指定されていない場合、デフォルト値は 900 秒になります。

フラッシュバック問合せの保存期間の選択

ロールバック情報の保存期間は、フラッシュバック問合せの実行にとって重要な要素です。Oracle のフラッシュバック問合せ機能を使用すると、指定した過去の時点でのデータベースの一貫性のあるバージョンを見ることができます。過去のある時点のデータベースに対して問合せやアプリケーションを実行できます。この機能は、オラクル社が提供する DBMS_FLASHBACK パッケージによりセッション・レベルで実装されています。オブジェクト・レベルでは、フラッシュバック問合せには SELECT 文の AS OF 句を使用して、データを表示する期間の直前の時点を指定します。

フラッシュバック問合せで過去のどの時点までさかのぼってデータベース・バージョンを確立できるかは、保存期間によって決まります。具体的には、対象とするデータベースの最も古いバージョンのデータベースのスナップショットを作成できるだけの十分長い UNDO 保存期間を選択する必要があります。たとえば、アプリケーションで 12 時間前の内容を反映したデータベースのバージョンを使用する場合は、UNDO_RETENTION を 43,200 に設定する必要があります。

自動 UNDO 管理を使用している場合、LOB 列の RETENTION 値は UNDO_RETENTION の値に設定されます。

関連項目：

- フラッシュバック問合せ機能の使用の詳細は、『Oracle9*i* アプリケーション開発者ガイド－基礎編』を参照してください。
- DBMS_FLASHBACK パッケージの詳細は、『Oracle9*i* 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。
- SELECT 文の AS OF 句の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

UNDO の保存に必要な領域要件の計算

具体的な UNDO_RETENTION パラメータ設定とシステム統計により、次の式を使用して、UNDO 保存の要件を満たすために必要な UNDO 領域の量を見積ることができます。

UndoSpace = UR * UPS + overhead

各項目の意味は次のとおりです。

- UndoSpace は、UNDO ブロック数です。
- UR は秒単位の UNDO_RETENTION です。
- UPS は 1 秒間に使用する UNDO ブロックの数です。
- overhead は、メタデータ（トランザクション表やビットマップなど）使用時の多少のオーバーヘッドです。

例として、UNDO_RETENTION を 2 時間に設定し、トランザクション率（UPS）を 200UNDO ブロック毎秒、ブロック・サイズを 4KB とすると、必要な UNDO 領域は次のように計算されます。

（2 × 3,600 × 200 × 4KB）=5.8GB

このような計算を行うには、V$UNDOSTAT ビュー内の情報を使用します。安定状態のときに、ビューを問い合せてトランザクション率を取得します。オーバーヘッド値も同じビューから取得できます。

UNDO 領域に関する情報の表示

ここでは、自動 UNDO 管理モードにおいて UNDO 領域に関する情報を表示する際に役立つビューについて説明します。ここで紹介したビュー以外にも、表領域やデータ・ファイルの情報を表示するビューを使用して、情報を取得できます。

関連項目：

- 11-49 ページ「表領域情報の表示」
- 12-28 ページ「データ・ファイル情報の表示」

UNDO 領域のビュー

UNDO 領域情報を取得するために、次のビューが使用できます。

ビュー	説明
V$UNDOSTAT	UNDO 領域の監視とチューニングのための統計情報が含まれます。このビューは、現行の作業負荷に必要な UNDO 領域の量を見積る際に利用できます。また、Oracle はこの情報を使用して、システム内の UNDO の使用方法をチューニングします。このビューは、自動 UNDO 管理および手動 UNDO 管理のどちらのモードでも使用できます。
V$ROLLSTAT	自動 UNDO 管理モードの場合、このビューの情報は、UNDO 表領域内の UNDO セグメントの動作を反映します。
V$TRANSACTION	UNDO セグメント情報が含まれます。
DBA_UNDO_EXTENTS	UNDO 表領域内の各エクステントのコミット時間を示します。

関連項目： 自動 UNDO 管理モードで使用するビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

UNDO 領域の監視

V$UNDOSTAT ビューは、現行インスタンス内の UNDO 領域におけるトランザクションの実行の効果を監視する際に役立ちます。UNDO 領域の消費、トランザクションの並行性、インスタンス内の問合せの長さに関する統計が使用できます。

ビュー内の各行には、インスタンス内で 10 分ごとに収集された統計が表示されます。行は、BEGIN_TIME 列の値の降順に並びます。各行は、BEGIN_TIME と END_TIME によってマーク付けされた時間間隔に基づいています。各列は、その時間間隔で収集された特定の統計データを表します。ビューの最初の行には、（部分的な）現在の時間間隔に対応する統計が含まれます。ビューには、7 日周期にわたる合計 1008 の行があります。

次の例は、V$UNDOSTAT ビューに対する問合せの結果を示したものです。

```
SELECT  BEGIN_TIME, END_TIME, UNDOTSN, UNDOBLKS, TXNCOUNT,
        MAXCONCURRENCY AS "MAXCON"
     FROM V$UNDOSTAT;
```

結果は次のとおりです。

```
BEGIN_TIME           END_TIME             UNDOTSN UNDOBLKS TXNCOUNT MAXCON
-------------------- -------------------- ------- -------- -------- ------
07/28/2000 18:26:28  07/28/2000 18:32:13        2      709       55      2
07/28/2000 18:16:28  07/28/2000 18:26:28        2      448       12      2
07/28/2000 14:36:28  07/28/2000 18:16:28        1        0        0      0
07/28/2000 14:26:28  07/28/2000 14:36:28        1        1        1      1
07/28/2000 14:16:28  07/28/2000 14:26:28        1       10        1      1
...
```

この例は、18:32:13 から 24 時間前までの間に、システムで UNDO 領域がどのように消費されたのかを示しています。

ロールバック・セグメントの管理

UNDO の格納にロールバック・セグメントの使用を選択する場合は、それらの管理について次の項を参照してください。

- ロールバック・セグメントを管理するためのガイドライン
- ロールバック・セグメントの作成
- ロールバック・セグメントの変更
- ロールバック・セグメントへのトランザクションの明示的な割当て
- ロールバック・セグメントの削除
- ロールバック・セグメント情報の表示

注意： UNDO 領域の管理にロールバック・セグメントを使用する機能は、将来のリリースで廃止される予定です。できるだけ自動 UNDO 管理を使用し、UNDO_TABLESPACE で UNDO 領域を管理するようにしてください。

ロールバック・セグメントを管理するためのガイドライン

ここでは、データベースのロールバック・セグメントの作成や管理を行う前に考慮すべきガイドラインについて説明します。この項の内容は、次のとおりです。

- 複数ロールバック・セグメントの使用
- パブリックとプライベートのロールバック・セグメントの選択
- 自動的に取得するロールバック・セグメントの指定
- ロールバック・セグメント・サイズの概算
- 等しいサイズのエクステントを数多く持つロールバック・セグメントの作成
- 各ロールバック・セグメントに対するエクステントの最適数の設定
- 異なる表領域へのロールバック・セグメントの配置

関連項目： ロールバック・セグメントの詳細は、『Oracle9*i* データベース概要』を参照してください。

複数ロールバック・セグメントの使用

複数のロールバック・セグメントを使用すると、ロールバック・セグメントの競合が多くのセグメントに分散され、システム・パフォーマンスが改善されます。Oracle は、ラウンドロビン法によってトランザクションをロールバック・セグメントに割り当てます。その結果、各ロールバック・セグメントのトランザクションの数がほぼ一様に分散されます。特定のロールバック・セグメントにトランザクションを割り当てることも可能ですが、通常は行いません。

データベースが作成されると、SYSTEM という名前の単一ロールバック・セグメントが SYSTEM 表領域に作成されます。このロールバック・セグメントは Oracle データベース・サーバーによる特殊な用途での使用を目的としており、一般的な用途では使用されません。SYSTEM 以外の表領域で作成されたオブジェクトに書き込む際は、その前に少なくとも 1 つのロールバック・セグメントを SYSTEM 以外の表領域に作成し、オンライン化する必要があります。

注意： データベースを初めて作成するときに、追加の表領域およびロールバック・セグメントを作成する場合は、SYSTEM 表領域に 2 番目のロールバック・セグメントを作成する必要があります。追加のロールバック・セグメントを作成した後、新しいロールバック・セグメントをアクティブにして、2 番目のロールバック・セグメントを使用不可にしてください。

インスタンスは起動時に、必要なロールバック・セグメントと取得するように指示されているロールバック・セグメントに加えて、SYSTEM ロールバック・セグメントも必ず取得（オンライン化）します。複数のロールバック・セグメントがあると、Oracle は特殊なシステム・トランザクション専用に SYSTEM ロールバック・セグメントを使用し、他のロールバック・セグメント間でユーザー・トランザクションを分配します。SYSTEM 以外のロールバック・セグメントのトランザクション数が多すぎる場合、Oracle は SYSTEM セグメントを使用します。このような状況が起こらないように、ロールバック・セグメントの数を決定してください。

インスタンスの起動時に複数のロールバック・セグメントをアクティブ化するには、次の 2 つの方法があります。

- パブリックのロールバック・セグメントを使用し、初期化パラメータ・ファイルに TRANSACTIONS および TRANSACTIONS_PER_ROLLBACK_SEGMENT 初期化パラメータを設定する。
- プライベートまたはパブリックのロールバック・セグメントを使用し、ROLLBACK_SEGMENTS 初期化パラメータにそれらの名前を指定する。

これらの方法については、後述の別のガイドラインで説明します。

同時にオープンできるロールバック・セグメントの数には上限があります。この上限は、MAX_ROLLBACK_SEGMENTS 初期化パラメータで設定します。このパラメータには、ROLLBACK_SEGMENTS 初期化パラメータで指定したロールバック・セグメントの数よりも必ず大きい値を設定してください。

> **関連項目：** TRANSACTIONS、TRANSACTIONS_PER_ROLLBACK_SEGMENT および ROLLBACK_SEGMENT の各初期化パラメータの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

パブリックとプライベートのロールバック・セグメントの選択

プライベート・ロールバック・セグメントは、インスタンスで明示的に取得する必要があります。この操作は、初期化パラメータ・ファイルの ROLLBACK_SEGMENTS パラメータにロールバック・セグメント名を設定することにより、データベースの起動時に実行できます。また、該当する文を手動で発行し、明示的にオンライン化することによって、プライベート・ロールバック・セグメントを取得することもできます。Oracle Real Application Clusters 環境では、プライベート・ロールバック・セグメントを使用すると、各インスタンスが特定のロールバック・セグメントを取得できます。

パブリック・ロールバック・セグメントは、ロールバック・セグメントを必要とする任意のインスタンスが使用可能なロールバック・セグメントのプールを形成します。インスタンスが起動時にこれらのロールバック・セグメントを自動的に何個取得するかは、TRANSACTIONS および TRANSACTIONS_PER_ROLLBACK_SEGMENT の初期化パラメータの値に基づいて決まります。パブリック・ロールバック・セグメントは、Oracle Real Application Clusters のインスタンス間で共有できます。

Oracle9*i* Real Application Clusters 機能を使用しない場合は、プライベートとパブリックのロールバック・セグメントはどちらも同じように機能します。

自動的に取得するロールバック・セグメントの指定

多数のトランザクションが同時に処理されると、ロールバック情報が同時に生成されます。インスタンスの起動時に適切な数のロールバック・セグメントを自動的に取得するよう指定する方法として、TRANSACTIONS および TRANSACTIONS_PER_ROLLBACK_SEGMENT の初期化パラメータを設定する方法があります。また、パブリック・ロールバック・セグメントを使用することも必要です。

初期化パラメータ TRANSACTIONS には、インスタンスに対して見積った同時トランザクション数を指定します。初期化パラメータ TRANSACTIONS_PER_ROLLBACK_SEGMENT には、各ロールバック・セグメントで処理する必要があるトランザクション数を指定します。インスタンスはデータベースのオープン時に、少なくとも *n* 個のロールバック・セグメントを取得しようとします（*n*=TRANSACTIONS/TRANSACTIONS_PER_ROLLBACK_SEGMENT）。そのため、データベースの作成時または作成後に、少なくとも *n* 個のパブリック・ロールバック・セグメントを作成する必要があります。

プライベート・ロールバック・セグメントの作成を選択した場合は、インスタンスの起動時にロールバック・セグメントを自動的に取得するため、インスタンスのパラメータ・ファイルの ROLLBACK_SEGMENTS 初期化パラメータにロールバック・セグメントの名前を指定します。

プライベートとパブリックの両方のロールバック・セグメントを使用すると、Oracle は次のように動作することがあります。TRANSACTIONS/TRANSACTIONS_PER_ROLLBACK_SEGMENT で算出される数よりも多くのロールバック・セグメントが指定されていても、インスタンスは ROLLBACK_SEGMENTS 初期化パラメータ内にリストされたセグメントをすべて取得します。

ロールバック・セグメント・サイズの概算

ロールバック・セグメント全体のサイズは、データベースに対して発行される最も一般的なトランザクションのサイズを基準にして設定する必要があります。一般に、バッチ・ジョブなどの長時間実行のトランザクションでは、ロールバック・セグメントが大きいほどパフォーマンスは向上しますが、短いトランザクションでは、データベースに小さなロールバック・セグメントが数多く存在する方が、パフォーマンスが向上します。一般に、ロールバック・セグメントはどんなサイズのトランザクションでも容易に処理できます。ただし、トランザクションが非常に短いまたは非常に長いなどの極端なケースでは、適切なサイズのロールバック・セグメントを使用してもかまいません。

システムが短いトランザクションのみを実行する場合、ロールバック・セグメントは、主メモリー内に常時キャッシュされるように小さくする必要があります。LRU アルゴリズムでは、ロールバック・セグメントが十分小さい場合にシステム・グローバル領域（SGA）内にキャッシュされる可能性が高くなり、必要なディスク I/O が少なくなるため、データベース・パフォーマンスが改善されます。小さいロールバック・セグメントの主な欠点は、他のトランザクションが頻繁に更新するレコードを必要とする長時間の問合せを実行していると

きに、「スナップショットが古すぎます」というエラーが起こる可能性が増大することです。このエラーが発生するのは、他の更新エントリがロールバック・セグメントを折り返したときに、読込み一貫性の実現に必要なロールバック・エントリが上書きされるためです。アプリケーションのトランザクションを設計する際はこの点を考慮し、問題が発生しないように短い基本作業単位でトランザクションを設計してください。

一方、長時間実行のトランザクションのロールバック・エントリは、大きなロールバック・セグメントの事前割当済みエクステントに適合するので、長時間実行のトランザクションは、より大きなロールバック・セグメントで効率よく実行できます。

データベース・システムのアプリケーションが、非常に短いトランザクションと非常に長いトランザクションを混合して同時に発行するときは、トランザクション / ロールバック・セグメント・サイズに基づいてロールバック・セグメントにトランザクションが明示的に割り当てられた場合に、最適なパフォーマンスが得られます。つまり、ロールバック・セグメントに対する動的なエクステント割当てと切捨てを最小限に抑えることができます。ほとんどのシステムでは、これは必要ありません。極端に大きい、または小さいトランザクションを処理するシステムのみを対象としています。

非常に小さいトランザクションと非常に大きいトランザクションを混合して発行する際のパフォーマンスを最適化するには、トランザクションのタイプ（小規模、中規模、大規模など）ごとにロールバック・セグメントを適切なサイズに設定します。この場合、ほとんどのロールバック・セグメントを標準的なトランザクション用とし、例外的なトランザクション用のロールバック・セグメントは少数にしてください。例外的なトランザクション用のロールバック・セグメントには、そのサイズが拡張した場合でも意図したサイズに戻るように、それぞれ `OPTIMAL` を設定します。

トランザクションのタイプによって対応するロールバック・セグメントが異なることについて、ユーザーに説明してください。多くの場合、特定のロールバック・セグメントにトランザクションを明示的に割り当てても利点はありません。ただし、例外的なトランザクション用に作成した適切なロールバック・セグメントには、そのトランザクションを割り当てることができます。たとえば、大きなバッチ・ジョブを含むトランザクションを、大きなロールバック・セグメントに割り当てることができます。

トランザクションの混在がそれほど多くない場合は、各ロールバック・セグメントのサイズをデータベースで一番大きな表のサイズの 10% にします。これは、ほとんどの SQL 文は表の 10% 以下にしか作用しないためです。ほとんどの SQL 文が実行する処理を格納するためには、このサイズのロールバック・セグメントで十分です。

一般的に、ロールバック・セグメントには大きな `MAXEXTENTS` を設定します。これにより、ロールバック・セグメントは、必要に応じて次のエクステントを割り当てることができます。

等しいサイズのエクステントを数多く持つロールバック・セグメントの作成

それぞれのロールバック・セグメントに割り当てられた全体の領域は、同じサイズの複数のエクステントに分ける必要があります。一般に、ロールバック I/O パフォーマンスが最適になるのは、インスタンスの各ロールバック・セグメント内に等しいサイズのエクステントが 10 ～ 20 個ある場合です。

ロールバック・セグメントの各エクステントのサイズ（*s*）を算出するには、ロールバック・セグメントの初期サイズの合計とセグメントの初期エクステント数を決定してから、次の式を使用します。

$$s = T / n$$

各項目の意味は次のとおりです。

s = 最初に割り当てられる各エクステントの算出サイズ（単位はバイト）

T = ロールバック・セグメントの初期サイズの合計（単位はバイト）

n = 最初に割り当てられる初期エクステントの数

s の計算後、ロールバック・セグメントを作成し、記憶域パラメータ INITIAL および NEXT として *s* を指定し、MINEXTENTS として *n* を指定します。ロールバック・セグメントには PCTINCREASE は指定できないので、デフォルトで 0（ゼロ）に設定されます。また、エクステントのサイズ *s* がデータ・ブロック・サイズの整数倍でない場合は、その次の整数倍に切り上げられます。

各ロールバック・セグメントに対するエクステントの最適数の設定

各ロールバック・セグメントに OPTIMAL パラメータを設定するときは、システムが実行するトランザクションの種類を慎重に評価してください。長時間実行のトランザクションを頻繁に実行するシステムの場合、Oracle がエクステントの縮小と割当てを頻繁に行う必要がないように、OPTIMAL を大きくする必要があります。また、アクティブ・データに対して長い問合せを実行するシステムでも、「スナップショットが古すぎます」というエラーを避けるために、OPTIMAL を大きくしてください。主に短いトランザクションと問合せを実行するシステムの場合には、ロールバック・セグメントがメモリー内にキャッシュされるほど十分小さくなるように、OPTIMAL をさらに小さく設定してください。これにより、システムのパフォーマンスが向上します。

V$ROLLNAME および V$ROLLSTAT の動的パフォーマンス・ビューを監視することで、OPTIMAL に適切な設定を判断する上で役立つ統計情報を収集できます。13-28 ページの「ロールバック・セグメント統計の監視」を参照してください。

異なる表領域へのロールバック・セグメントの配置

可能であれば、すべてのロールバック・セグメントを保持する専用の表領域を 1 つ以上作成してください。このようにすると、すべてのロールバック・セグメント・データは、他のタイプのデータから分離して格納されます。このようなロールバック・セグメント表領域を作成することには、次の利点があります。

- ロールバック・セグメントを保持している表領域は常にオンライン状態にしておくことができるため、ロールバック・セグメントの合計記憶域容量を常に最大限利用できます。ただし、ロールバック・セグメントの中に利用できないものがある場合には、データベース操作全体に影響が及ぶ可能性があります。
- アクティブ・ロールバック・セグメントを持つ表領域は、オフライン化できません。したがって、ある表領域がデータベースのすべてのロールバック・セグメントを保持するように設計することにより、他の表領域内に格納されたデータをデータベースのロールバック・セグメントとは無関係にオフライン化できます。
- 表領域にエクステントの割当てと割当て解除が頻繁に実行されるロールバック・セグメントが含まれる場合は、使用可能エクステントが断片化する可能性が高くなります。

ロールバック・セグメントの作成

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。CREATE ROLLBACK SEGMENT 文を使用します。新しいロールバック・セグメントを格納する表領域は、オンライン化する必要があります。通常、ロールバック・セグメントは、データベース作成スクリプトまたはプロセスの一部として作成されますが、後から追加することもできます。

この項の内容は、次のとおりです。

- CREATE ROLLBACK SEGMENT 文
- 新しいロールバック・セグメントをオンライン化する方法
- ロールバック・セグメントを作成するときに記憶域パラメータを設定する方法

CREATE ROLLBACK SEGMENT 文

次の文は、rbsspace 表領域のデフォルトの記憶域パラメータを使用して、rbsspace 表領域内にロールバック・セグメント rbs_02 を作成します。これは Oracle Real Application Clusters 環境ではないため、PRIVATE または PUBLIC を指定する必要はありません。デフォルト値は PRIVATE です。

```
CREATE ROLLBACK SEGMENT rbs_02 TABLESPACE rbsspace;
```

> **関連項目：** ロールバック・セグメントの管理に使用する SQL 文の正確な構文、制限および認可要件については、『Oracle9*i* SQL リファレンス』を参照してください。

新しいロールバック・セグメントをオンライン化する方法

新しいロールバック・セグメントは、最初はオフラインになっています。ALTER ROLLBACK SEGMENT を発行してオンライン化し、インスタンスのトランザクションで使用可能にする必要があります。この操作については、13-22 ページの「ロールバック・セグメントの ONLINE/OFFLINE 状態の変更」を参照してください。

新しいプライベート・ロールバック・セグメントを作成する場合は、そのロールバック・セグメントの名前をデータベースの初期化パラメータ・ファイル内の ROLLBACK_SEGMENTS 初期化パラメータに追加します。これにより、そのプライベート・ロールバック・セグメントは、インスタンス起動時にインスタンスによって自動的に取得されます。たとえば、2 つの新しいプライベート・ロールバック・セグメントを作成し、それぞれ rbs_01 および rbs_02 という名前を付けた場合は、ROLLBACK_SEGMENTS 初期化パラメータを次のように指定します。

```
ROLLBACK_SEGMENTS = (rbs_01, rbs_02)
```

ロールバック・セグメントを作成するときに記憶域パラメータを設定する方法

次のような記憶域パラメータと最適サイズが設定されたロールバック・セグメント rbs_01 を作成する場合を想定します。

- ロールバック・セグメントに割り当てられる初期エクステント : 100KB
- ロールバック・セグメントに割り当てられる第 2 エクステント : 100KB
- ロールバック・セグメントの最適サイズ : 4MB
- 最小エクステント数、およびセグメントが作成されるときに最初に割り当てられるエクステント数 : 20
- ロールバック・セグメントの割当て可能な最大エクステント数（初期エクステントを含む）: 100

次の文は、このような特性を持つロールバック・セグメントを作成します。

```
CREATE ROLLBACK SEGMENT rbs_01
                TABLESPACE rbsspace
                STORAGE (
      INITIAL 100K
      NEXT 100K
      OPTIMAL 4M
      MINEXTENTS 20
      MAXEXTENTS 100 );
```

PCTINCREASE 記憶域パラメータの値は設定できません。ロールバック・セグメントの場合は常に 0（ゼロ）に設定されます。OPTIMAL 記憶域パラメータは、ロールバック・セグメントにのみ指定可能です。記憶域パラメータの詳細は、14-8 ページの「記憶域パラメータの設定」を参照してください。

オラクル社の推奨事項は、次のとおりです。

- すべてのエクステントが同じサイズになるように、INITIAL と NEXT を同じ値に設定します。
- 動的拡張の可能性を最小限に抑えるために、多数の初期エクステントを作成します。推奨値は、MINEXTENTS = 20 です。
- MAXEXTENTS = UNLIMITED に設定するのは避けてください。この設定では、プログラミング・エラーが原因でロールバック・セグメントやデータ・ファイルが不必要に拡張されるおそれがあります。UNLIMITED を指定する必要がある場合は、そのセグメントのエクステントが最低でも 4 つのデータ・ブロックを必ず持つことになるので注意してください。また、MAXEXTENTS が制限されているロールバック・セグメントを後で UNLIMITED に変換しようとしても、データ・ブロック数が 4 より小さいエクステントがあると、そのロールバック・セグメントは変換できません。制限付きフォーマットから UNLIMITED に変換し、エクステント内のデータ・ブロック数を 3 以下にする場合は、そのロールバック・セグメントを削除して再作成するしかありません。

関連項目： 記憶域パラメータの詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

ロールバック・セグメントの変更

ここでは、ロールバック・セグメントのメンテナンス時に実行できる各種のアクションについて説明します。これらのメンテナンス・アクティビティでは、いずれも ALTER ROLLBACK SEGMENT 文を使用します。この文を使用するには、ALTER ROLLBACK SEGMENT システム権限が必要です。

この項の内容は、次のとおりです。

- ロールバック・セグメント記憶域パラメータの変更
- 手動によるロールバック・セグメントの縮小
- ロールバック・セグメントの ONLINE/OFFLINE 状態の変更

ロールバック・セグメント記憶域パラメータの変更

一部のロールバック・セグメント記憶域パラメータは、作成後に変更することができます。値を変更できるパラメータは、OPTIMAL と MAXEXTENTS です。次の文は、rbs_01 ロールバック・セグメントで割当て可能なエクステントの最大数を変更します。

```
ALTER ROLLBACK SEGMENT rbs_01
    STORAGE (MAXEXTENTS 120);
```

他のロールバック・セグメントの設定変更と同じように、SYSTEM ロールバック・セグメントでも、OPTIMAL パラメータなどの設定を変更できます。

手動によるロールバック・セグメントの縮小

ALTER ROLLBACK SEGMENT 文を使用して、ロールバック・セグメントのサイズを手動で小さくできます。縮小するロールバック・セグメントは、オンライン化しておく必要があります。

次の文は、ロールバック・セグメント rbs1 を 100KB に縮小します。

```
ALTER ROLLBACK SEGMENT rbs1 SHRINK TO 100K;
```

この文はロールバック・セグメントのサイズを指定のサイズまで縮小しようとしますが、エクステントがアクティブであるために割当て解除できない場合は途中で停止します。

ロールバック・セグメントの ONLINE/OFFLINE 状態の変更

ここでは、ロールバック・セグメントをオンライン化およびオフライン化する方法について説明します。この項の内容は、次のとおりです。

- ロールバック・セグメントを手動でオンライン化する方法
- ロールバック・セグメントを自動的にオンライン化する方法
- ロールバック・セグメントをオフライン化する方法

ロールバック・セグメントの状態は、オンラインでトランザクションによって利用できるか、またはオフラインでトランザクションによって利用できないかのどちらかです。ほとんどの場合、ロールバック・セグメントはオンラインであり、トランザクションによって使用可能な状態にあります。

次のような状況では、オンライン・ロールバック・セグメントをオフライン化できます。

- 表領域をオフライン化しようとしており、その表領域にロールバック・セグメントが含まれています。トランザクションによって使用されているロールバック・セグメントが表領域に含まれている場合は、その表領域をオフライン化できません。対応するロールバック・セグメントが使用されないようにするには、表領域をオフライン化する前にロールバック・セグメントをオフライン化します。
- ロールバック・セグメントを削除しようとしていますが、現在トランザクションによって使用されているので削除できません。ロールバック・セグメントが使用されないようにするには、削除する前にロールバック・セグメントをオフライン化します。

注意： SYSTEM ロールバック・セグメントはオフライン化できません。

オフライン・ロールバック・セグメントを後からオンライン化して、トランザクションで使用できます。作成直後のロールバック・セグメントはオフラインになっています。新しく作成したロールバック・セグメントをインスタンスのトランザクションで使用するには、明示的にオンライン化する必要があります。そのロールバック・セグメントを含むデータベースにアクセスする任意のインスタンスを使用して、オフラインのロールバック・セグメントをオンライン化できます。

ロールバック・セグメントを手動でオンライン化する方法 オンライン化できるロールバック・セグメントは、現在の状態が OFFLINE または PARTLY AVAILABLE であるものにかぎられます（現在の状態は DBA_ROLLBACK_SEGS データ・ディクショナリ・ビューで表示できます)。オフライン・ロールバック・セグメントをオンライン化するには、ONLINE オプションを指定した ALTER ROLLBACK SEGMENT 文を使用します。

次の文は、ロールバック・セグメント user_rs_2 をオンライン化します。

```
ALTER ROLLBACK SEGMENT user_rs_2 ONLINE;
```

オンライン化すると、データ・ディクショナリ・ビュー DBA_ROLLBACK_SEGS でのロールバック・セグメントの状態が ONLINE になります。ロールバック・セグメントの状態をチェックする問合せについては、13-27 ページの「ロールバック・セグメント情報の表示」を参照してください。

PARTLY AVAILABLE 状態のロールバック・セグメントには、インダウトまたはリカバリした分散トランザクションのデータや、まだリカバリされていないトランザクションのデータが含まれています。データ・ディクショナリ・ビュー DBA_ROLLBACK_SEGS でこの状態を見ると、PARTLY AVAILABLE と表示されます。ロールバック・セグメントは、トランザクションが RECO によって自動的に解決されるか、または DBA によって手動で解決されるまで、通常はこの状態のままです。

ただし、すべてのロールバック・セグメントが PARTLY AVAILABLE になる場合もあります。この場合は、PARTLY AVAILABLE セグメントをオンライン化できます。ロールバック・セグメントがインダウト・トランザクションのために使用するいくつかのリソースは、そのトランザクションが解決されるまでアクセスできない状態になります。その結果、他のトランザクションが追加の領域を必要とする場合は、必要に応じてロールバック・セグメントのサイズが大きくなります。

PARTLY AVAILABLE ロールバック・セグメントをオンライン化するかわりに、インダウト・トランザクションが解決されるまで、一時的に新しいロールバック・セグメントを作成する方が効率的な場合もあります。

ロールバック・セグメントを自動的にオンライン化する方法 データベースを起動するたびにロールバック・セグメントを自動的にオンライン化する場合は、データベースのパラメータ・ファイルの ROLLBACK_SEGMENTS パラメータにそのセグメントの名前を追加します。あるいは、パブリック・ロールバック・セグメントを使用して、TRANSACTIONS および TRANSACTIONS_PER_ROLLBACK_SEGMENT 初期化パラメータを使用します。

これらの操作については、13-16 ページの「自動的に取得するロールバック・セグメントの指定」を参照してください。

ロールバック・セグメントをオフライン化する方法 オンラインのロールバック・セグメントをオフライン化するには、OFFLINE オプションを指定した ALTER ROLLBACK SEGMENT 文を使用します。変更するロールバック・セグメントは、DBA_ROLLBACK_SEGS データ・ディクショナリ・ビュー内で ONLINE の状態になっている必要があり、現行インスタンスによって取得されている必要があります。

たとえば、次の例ではロールバック・セグメント user_rs_2 がオフライン化されます。

```
ALTER ROLLBACK SEGMENT user_rs_2 OFFLINE;
```

アクティブなロールバック・セグメントを含まないロールバック・セグメントをオフライン化しようとすると、Oracle はただちにそのセグメントをオフライン化し、その状態を OFFLINE に変更します。

これに対して、アクティブ・トランザクション（ローカル、リモートまたは分散）のロールバック・データを含むロールバック・セグメントを使用する場合、Oracle はロールバック・セグメントが将来のトランザクションで使用されないようにし、そのロールバック・セグメントを使用しているアクティブ・トランザクションがすべて完了してから、オフライン化します。トランザクションが完了するまで、オフライン化しようとしているインスタンス以外のどのインスタンスも、そのロールバック・セグメントをオンライン化できません。

ロールバック・セグメントがオフライン化されるまで待機している間、ビュー DBA_ROLLBACK_SEGS でのロールバック・セグメントの状態は ONLINE のままです。しかし、ビュー V$ROLLSTAT でのロールバック・セグメントの状態は PENDING OFFLINE になります。ロールバック・セグメントの状態の表示については、13-27 ページの「ロールバック・セグメント情報の表示」を参照してください。

ロールバック・セグメントのオフライン化を試み、その状態を PENDING OFFLINE に変更したインスタンスは、いつでもそのロールバック・セグメントをオンラインに戻すことができます。オンラインに戻されたロールバック・セグメントは、通常どおり機能します。

パブリックまたはプライベートのロールバック・セグメントをオフライン化すると、それを明示的にオンラインに戻すか、またはインスタンスを再起動するまで、オフライン状態のままです。

ロールバック・セグメントへのトランザクションの明示的な割当て

トランザクションには特定のロールバック・セグメントを明示的に割り当てることができます。これを行うのは、次のような場合です。

- トランザクションが生成するロールバック情報の量を予測できる場合。ロールバック情報がセグメントの現在のエクステントに収まることがあらかじめわかっている場合は、そのロールバック・セグメントにトランザクションを割り当てることができます。これにより、動的に割り当てられて後で切り捨てられる追加のエクステントのオーバーヘッドを減らすことができます。
- 長時間実行の問合せが同時に同じ表を読み込むことがない場合。この場合は、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たときに、それらのセグメントがメモリー内に残る可能性が高くなります。
- 長時間実行の問合せによって同時に読み込まれる表を変更するトランザクションがある場合。これらのトランザクションを大きなロールバック・セグメントに割り当てることで、読込み一貫性のある問合せに必要なロールバック情報が上書きされることを回避できます。

トランザクションを特定のロールバック・セグメントに明示的に割り当てるには、`USE ROLLBACK SEGMENT` 句を指定した `SET TRANSACTION` 文を使用します。ロールバック・セグメントは現行インスタンスに対してオンラインであり、`SET TRANSACTION USE ROLLBACK SEGMENT` 文がトランザクションの最初の文である必要があります。指定したロールバック・セグメントがオンラインでない場合、または `SET TRANSACTION USE ROLLBACK SEGMENT` 句がトランザクション内の最初の文でない場合には、エラーが返されます。

たとえば、大量（通常のトランザクションよりも多く）の作業を含むトランザクションを開始する場合は、次の文を使用して、そのトランザクションを大きなロールバック・セグメントに割り当てることができます。

```
SET TRANSACTION USE ROLLBACK SEGMENT large_rs1;
```

トランザクションがコミットされた後、ユーザーが新しいトランザクションを特定のロールバック・セグメントに明示的に割り当てないかぎり、Oracle は次のトランザクションを利用可能なロールバック・セグメントに自動的に割り当てます。

ロールバック・セグメントの削除

ディスク上でセグメントのエクステントが著しく断片化された場合や、セグメントを異なる表領域に再割当てする必要があるときには、ロールバック・セグメントを削除できます。ロールバック・セグメントを削除する前に、その状態が OFFLINE であることを確認してください。削除するロールバック・セグメントが OFFLINE 以外の状態になっている場合は、削除できません。状態が INVALID の場合、そのセグメントはすでに削除されています。

ロールバック・セグメントを削除するには、DROP ROLLBACK SEGMENT 文を使用します。そのためには、DROP ROLLBACK SEGMENT システム権限が必要です。次の文は、rbs1 ロールバック・セグメントを削除します。

```
DROP ROLLBACK SEGMENT rbs1;
```

注意： ROLLBACK_SEGMENTS に指定されたロールバック・セグメントを削除する場合は、データベースのパラメータ・ファイルを編集して、ROLLBACK_SEGMENTS パラメータのリストから削除したロールバック・セグメントの名前を必ず削除してください。この手順が次のインスタンス起動の前に実行されていないと、削除されたロールバック・セグメントを取得できないため、起動は失敗に終わります。

ロールバック・セグメントが削除されると、その状態は INVALID になります。次に作成されたロールバック・セグメントは、削除されたロールバック・セグメントによって空き状態になっている行が使用できる場合、その行を使用します。これにより、削除されたロールバック・セグメントの行が DBA_ROLLBACK_SEGS ビューから消去されます。

ロールバック・セグメント情報の表示

ここでは、ロールバック・セグメント情報の取得と監視に使用できるビューと、その使用に関する情報および使用例について説明します。

この章の内容は、次のとおりです。

- ロールバック・セグメントのビュー
- ロールバック・セグメント情報の表示
- ロールバック・セグメント統計の監視
- すべてのロールバック・セグメントの表示
- ロールバック・セグメントがオフラインになったかどうかを表示する方法

関連項目： ここで説明するデータ・ディクショナリ・ビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

ロールバック・セグメントのビュー

次のビューは、ロールバック・セグメントに関する情報を表示する際に役立ちます。

ビュー	説明
DBA_ROLLBACK_SEGS	名前と表領域を含むロールバック・セグメントの説明が表示されます。
DBA_SEGMENTS	セグメントがロールバック・セグメントとして識別され、セグメントの追加情報が表示されます。
V$ROLLNAME	オンラインのロールバック・セグメントの名前がすべてリストされます。
V$ROLLSTAT	ロールバック・セグメントの統計が含まれます。
V$TRANSACTION	UNDO セグメント情報が含まれます。

ロールバック・セグメント情報の表示

DBA_ROLLBACK_SEGS データ・ディクショナリ・ビューには、データベースのロールバック・セグメントに関する情報が格納されています。たとえば、次の問合せは、データベース内の各ロールバック・セグメントの名前、対応する表領域および状態をリストします。

```
SELECT SEGMENT_NAME, TABLESPACE_NAME, STATUS
        FROM DBA_ROLLBACK_SEGS;

SEGMENT_NAME  TABLESPACE_NAME     STATUS
------------- ----------------    ------
SYSTEM        SYSTEM              ONLINE
PUBLIC_RS     SYSTEM              ONLINE
USERS_RS      USERS               ONLINE
```

また、次のデータ・ディクショナリ・ビューには、データベースのセグメントに関する情報が含まれています。

- USER_SEGMENTS
- DBA_SEGMENTS

ロールバック・セグメント統計の監視

V$ROLLSTAT 動的パフォーマンス・ビューを問い合せて、ロールバック・セグメント統計を監視できます。このビューを V$ROLLNAME ビューと結合して、セグメント番号を名前にマッピングする必要があります。

V$ROLLSTAT ビューで重要ないくつかの列を次に示します。

列名	説明
USN	ロールバック・セグメント番号。このビューを V$ROLLNAME ビューに結合すると、ロールバック・セグメント名を判断できます。
WRITES	ロールバック・セグメントに書き込まれたバイト数。
XACTS	アクティブ・トランザクション数。
GETS	ロールバック・セグメントのヘッダー要求数。
WAITS	待機状態になったロールバック・セグメントのヘッダー要求数。
OPTSIZE	ロールバック・セグメントの OPTIMAL パラメータの値。
HWMSIZE	使用中にロールバック・セグメント・サイズが到達した最大バイト数（最高水位標）。
SHRINKS	ロールバック・セグメントが最適なサイズを保つために実行する必要があった縮小回数。
WRAPS	ロールバック・セグメント・エントリがエクステント間で折り返した回数。
EXTENDS	ロールバック・セグメントが新規エクステントを取得する必要があった回数。
AVESHRINK	縮小時に解放された平均バイト数。
AVEACTIVE	ロールバック・セグメント内のアクティブ・エクステントの平均バイト数（一定時間内での計測値）。

これらの統計は、システム起動時にリセットされます。

このビューの非定型問合せの実行結果は、OPTIMAL パラメータの最適な設定を判断する上で役立ちます。インスタンスに、同等サイズのエクステントを持つ同一サイズのロールバック・セグメントがある場合は、特定のロールバック・セグメントの OPTIMAL パラメータを AVEACTIVE より少し大きい値に設定してください。次のチャートは、このビューに表示される統計の解析方法に関する追加情報を示しています。

SHRINKS	AVESHRINK	分析と推奨事項
Low	Low	AVEACTIVE が OPTSIZE に近い場合、OPTIMAL は正しい値に設定されています。そうでない場合は、OPTIMAL が大きすぎます（縮小はあまり実行されていません）。
Low	High	良好。OPTIMAL の設定は理想的です。
High	Low	OPTIMAL が小さすぎます。縮小が必要以上に頻繁に実行されています。
High	High	おそらく定期的な大規模トランザクションがこの統計の原因です。SHRINKS の値が小さくなるまで OPTIMAL パラメータの設定を大きくします。

すべてのロールバック・セグメントの表示

次の問合せは、各ロールバック・セグメントの名前、ロールバック・セグメントを含む表領域およびサイズを返します。

```
SELECT SEGMENT_NAME, TABLESPACE_NAME, BYTES, BLOCKS, EXTENTS
   FROM DBA_SEGMENTS
   WHERE SEGMENT_TYPE = 'ROLLBACK';

SEGMENT_NAME   TABLESPACE_NAME    BYTES      BLOCKS     EXTENTS
------------   ---------------    -------    ------     -------
SYSTEM         SYSTEM              409600       200           8
RB_TEMP        SYSTEM             1126400       550          11
RB1            RBS                 614400       300           3
RB2            RBS                 614400       300           3
RB3            RBS                 614400       300           3
RB4            RBS                 614400       300           3
RB5            RBS                 614400       300           3
RB6            RBS                 614400       300           3
RB7            RBS                 614400       300           3
RB8            RBS                 614400       300           3
10 rows selected.
```

ロールバック・セグメントがオフラインになったかどうかを表示する方法

ロールバック・セグメントをオフライン化するときは、アクティブ・トランザクションがすべて完了するまで、実際にはオフラインになりません。オフライン化しようとしてから実際にオフラインになるまでの間、V$ROLLSTAT の状態は PENDING OFFLINE になっており、新しいトランザクションでは使用されません。インスタンスに対するロールバック・セグメントがこの状態になっているかどうかを判断するには、次の問合せを使用します。

```
SELECT NAME, XACTS "ACTIVE TRANSACTIONS"
      FROM V$ROLLNAME, V$ROLLSTAT
      WHERE STATUS = 'PENDING OFFLINE'
        AND V$ROLLNAME.USN = V$ROLLSTAT.USN;

NAME         ACTIVE TRANSACTIONS
----------   --------------------
RS2                             3
```

インスタンスが Oracle Real Application Clusters の一部を構成している場合、この問合せは現行インスタンスのみのロールバック・セグメント情報を表示し、その他のインスタンスの情報は表示しません。

第 III 部

スキーマ・オブジェクト

第 III 部では、Oracle データベース内のスキーマ・オブジェクトの作成とメンテナンスについて説明します。第 III 部の構成は、次のとおりです。

14

スキーマ・オブジェクトの領域の管理

ここでは、スキーマ・オブジェクトの領域を管理する際のガイドラインについて説明します。この章の内容は、次のとおりです。

- データ・ブロックの領域管理
- 記憶域パラメータの設定
- 再開可能領域割当ての管理
- 領域の割当て解除
- データ型の領域使用の理解

後の章の説明に従って特定のスキーマ・オブジェクトを管理する前に、この章で説明する概念をよく理解しておく必要があります。

データ・ブロックの領域管理

ここでは、データ・ブロックの領域を管理する方法について説明します。データ・ブロックは、データベース・データをディスク上に格納する構造の中で最も細分化レベルの細かい構造です。データ・ブロックのサイズは、データベースの作成時に指定します（またはデフォルトで設定されます）。

PCTFREE および PCTUSED パラメータは、スキーマ・オブジェクトを作成または変更するときに指定できる物理属性です。これらのパラメータを使用すると、データ・ブロック内の空き領域の使用方法を制御できます。空き領域は、データ行の挿入や更新に使用できます。

PCTFREE および PCTUSED パラメータを使用すると、次のことができます。

- データの書込み時および取得時のパフォーマンスを改善する。
- データ・ブロック内の未使用領域を少なくする。
- データ・ブロック間の行連鎖を少なくする。

INITRANS および MAXTRANS パラメータも、スキーマ・オブジェクトを作成または変更するときに指定できる物理属性です。これらのパラメータは、スキーマ・オブジェクトのデータ・ブロックに割り当てられる同時更新トランザクションの数を制御します。これにより、データ・ブロック・ヘッダーの領域の使用と、データ・ブロックの空き領域を制御できます。

この項の内容は、次のとおりです。

- PCTFREE パラメータの指定
- PCTUSED パラメータの指定
- 関連する PCTUSED と PCTFREE の値の選択
- トランザクション・エントリ・パラメータの指定 : INITRANS と MAXTRANS

関連項目：

- データ・ブロックの詳細は、『Oracle9*i* データベース概要』を参照してください。
- PCTFREE、PCTUSED、INITRANS および MAXTRANS の各物理属性パラメータの構文などの詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

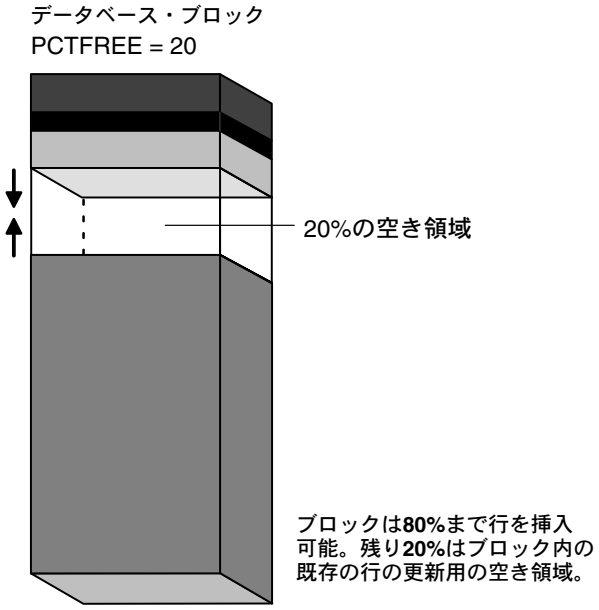
PCTFREE パラメータの指定

PCTFREE パラメータは、すでにブロックに含まれている行の更新用に確保しておくブロックの割合を設定します。たとえば、CREATE TABLE 文で次のようなパラメータを指定するとします。

```
PCTFREE 20
```

これは、この表のデータ・セグメントに使用される各データ・ブロックの 20 パーセントが空きのまま保持され、各ブロックにすでに存在する行を更新する際に使用できることを示します。図 14-1 は PCTFREE を示しています。

図 14-1　PCTFREE



ブロックが PCTFREE に達するまで、新しい行の挿入とデータ・ブロック・ヘッダーの増加によって、データ・ブロックの空き領域が埋められていきます。

PCTFREE を設定する前に必ず、表や索引データの性質を把握してください。更新により行が大きくなることがあります。新しい値は、それらが置き換わる値と同じサイズの場合も、そうでない場合もあります。データ値が増加していく更新を何度も実行する場合は、PCTFREE を大きくする必要があります。行に対する更新が全体の行幅に影響を与えない場合は、PCTFREE は小さくてもかまいません。データベース管理者（DBA）の目標は、高密度でまとめたデータと優れた更新パフォーマンスとの間で満足のいく妥協点を見いだすことです。

PCTFREE のデフォルト値は 10 パーセントです。PCTFREE と PCTUSED の合計が 100 を超えない範囲であれば、0 ～ 99 の任意の整数（0 と 99 を含む）を指定できます。

PCTFREE を小さく指定した場合の影響

PCTFREE の値を小さくすると、次のような影響が表れます。

- 既存の表の行を更新するために確保される領域が小さくなります。
- 挿入によってより完全にブロックが埋められます。
- 少ないブロックで表または索引の全データが格納される（各ブロックの行またはエントリは多くなる）ので、領域が節約されることもあります。

ほとんど変更されないセグメントなどでは、PCTFREE の値は小さいほうが適しています。

PCTFREE を大きく指定した場合の影響

PCTFREE の値を大きくすると、次のような影響が表れます。

- 既存の表の行を更新するために確保される領域が大きくなります。
- 同じ量の挿入データに必要なブロックが多くなる（各ブロックに挿入する行は少なくなる）ことがあります。
- Oracle は行断片を頻繁に連鎖する必要がないため、更新パフォーマンスが改善されることがあります。

頻繁に更新されるセグメントなどでは、PCTFREE の値が大きいほうが適しています。

非クラスタ化表の PCTFREE

非クラスタ化表の行のデータ・サイズが増加しそうな場合は、更新のための領域をいくらか確保してください。そうしないと、更新された行がブロック間で連鎖してしまう可能性が高くなります。

クラスタ化表の PCTFREE

非クラスタ化表での説明は、クラスタ化表にも当てはまります。ただし、PCTFREE に達すると、同じクラスタ・キーに含まれている任意の表の新しい行が、既存のクラスタ・キーに連鎖される新しいデータ・ブロックに格納されます。

索引の PCTFREE

索引の PCTFREE を指定できるのは、初めて索引を作成するときのみです。

PCTUSED パラメータの指定

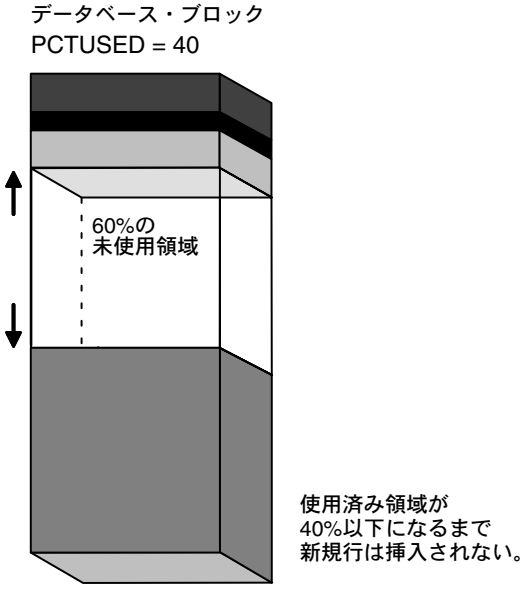
注意： セグメント領域管理が AUTO に指定されているローカル管理表領域に作成されたオブジェクトでは、PCTUSED パラメータは無視されます。この方式によるセグメント領域管理の詳細は、11-7 ページの「ローカル管理表領域のセグメント領域管理の指定」を参照してください。

一度データ・ブロックが PCTFREE まで達すると、使用されているブロックの割合が PCTUSED パラメータより小さくなるまで、そのブロックに新しい行は挿入されません。この値に達するまでは、データ・ブロックの空き領域はそのデータ・ブロックにすでに含まれている行の更新用にしか使用されません。たとえば、CREATE TABLE 文で次のようなパラメータを指定するとします。

```
PCTUSED 40
```

この場合、この表のデータ・セグメントに使用されるデータ・ブロックには、ブロック内で使用されている領域が 39 パーセント以下になるまでは、新しい行は挿入されません（ブロックが使用する領域がすでに PCTFREE に達している場合）。図 14-2 は、この状態を示しています。

図 14-2　PCTUSED



PCTUSED のデフォルト値は 40 パーセントです。一度データ・ブロックの空き領域が PCTFREE に達すると、使用されている領域の割合が PCTUSED を下回るまで、そのブロックに新しい行は挿入されません。このパーセント値は、合計領域からオーバーヘッドを差し引いた後の、データに使用できるブロック領域に対するものです。

PCTUSED と PCTFREE の合計が 100 を超えない範囲であれば、PCTUSED には 0 ～ 99 の間の任意の整数（0 と 99 を含む）を指定できます。

PCTUSED を小さく指定した場合の影響

PCTUSED の値を小さくすると、次のような影響が表れます。

- その使用の割合を下回ったときに、ブロックを空きリストに移動するために UPDATE 文と DELETE 文で発生する処理コストが少なくなります。
- データベース内の未使用領域が増大します。

PCTUSED を大きく指定した場合の影響

PCTUSED の値を大きくすると、次のような影響が表れます。

- 領域効率が改善されます。
- INSERT および UPDATE 文の処理コストが増えます。

関連する PCTUSED と PCTFREE の値の選択

PCTFREE または PCTUSED のデフォルト値を使用しない場合は、次のガイドラインを参考にしてください。

- PCTFREE と PCTUSED の和は 100 以下にしてください。
- 和が 100 の場合、Oracle は PCTFREE の空き領域を維持しようとし、処理コストは最大になります。
- PCTUSED が 75 で PCTFREE が 20 というように、PCTFREE および PCTUSED の和と 100 との差が小さいほど、領域使用の効率はそれだけよくなりますが、パフォーマンスは多少犠牲になります。

次の例では、表に対する PCTFREE および PCTUSED の値の指定方法とその理由を示します。

例	使用例	設定	説明
1	一般的なアクティビティに、行のサイズを大きくする UPDATE 文が含まれています。	PCTFREE=20 PCTUSED=40	更新の結果サイズが増える行のための領域を確保するため、PCTFREE を 20 に設定します。高い更新アクティビティ時の処理を少なくしてパフォーマンスを改善するために、PCTUSED を 40 に設定します。
2	ほとんどのアクティビティに、INSERT 文や DELETE 文、および処理の対象となる行のサイズを大きくしない UPDATE 文が含まれています。	PCTFREE=5 PCTUSED=60	ほとんどの UPDATE 文で行のサイズが増加しないため、PCTFREE を 5 に設定します。DELETE 文によって解放された領域をすぐに使用し、同時に処理の最小化を図るため、PCTUSED を 60 に設定します。
3	表が非常に大きく、記憶域が主な問題となります。ほとんどのアクティビティに、読取り専用トランザクションが含まれています。	PCTFREE=5 PCTUSED=40	表が大きく、各ブロックを完全に埋めるのが望ましいため、PCTFREE を 5 に設定します。

トランザクション・エントリ・パラメータの指定 : INITRANS と MAXTRANS

INITRANS は、データ・ブロック・ヘッダー内で領域を最初に確保するデータ操作言語（DML）トランザクション・エントリの数を指定します。この領域は、対応するセグメントに含まれるすべてのデータ・ブロックのヘッダー内に確保されます。

複数のトランザクションが同時に同じデータ・ブロックの行にアクセスするために、ブロック内の各 DML トランザクションのエントリに対して領域が割り当てられます。INITRANS によって確保された領域が使い果たされると、利用できる場合には、追加のトランザクション・エントリのための領域が、ブロック内の空き領域から割り当てられます。いったん割り当てられると、この領域は実質上ブロック・ヘッダーの永続部分となります。MAXTRANS パラメータは、データ・ブロック内のデータを同時に使用できるトランザクション・エントリの数を制限します。したがって、MAXTRANS を使用して、データ・ブロック内のトランザクション・エントリに割り当てることのできる空き領域を制限できます。

特定のスキーマ・オブジェクトに割り当てたデータ・ブロックの INITRANS パラメータおよび MAXTRANS パラメータは、各スキーマ・オブジェクトごとに次の条件に基づいて個別に設定してください。

- データベース・データ用に確保する領域と比較した、トランザクション・エントリ用に確保する領域
- ある特定の時間に同じデータ・ブロックにアクセスする同時実行トランザクションの数

たとえば、かなり大きな表でも、わずかなユーザーしか同時にアクセスしない場合は、複数の同時実行のトランザクションが同じデータ・ブロックへのアクセスを必要とする確率は低くなります。したがって、特にデータベースで領域が貴重な場合には、INITRANS を小さく設定できます。

一方、日常的に多くのユーザーが表に同時にアクセスする場合もあります。このような場合は、高い INITRANS を使用して、トランザクション・エントリ領域を事前に割り当てておくことを検討します。こうすると、オブジェクトが使用中のときに必要となる、トランザクション・エントリ領域の割当てに伴うオーバーヘッドが解消されます。また、ユーザーが必要なデータ・ブロックにアクセスするまでに待機することがないように、MAXTRANS にさらに大きい値を設定します。

記憶域パラメータの設定

ここでは、各種のデータ構造に対して設定できる記憶域パラメータについて説明します。これらの記憶域パラメータは、次のタイプの構造とスキーマ・オブジェクトに適用されます。

- 表領域（すべてのセグメントの記憶域パラメータのデフォルトとして使用される）
- 表、パーティション、クラスタ、マテリアライズド・ビューおよびマテリアライズド・ビュー・ログ（データ・セグメント）
- 索引（索引セグメント）
- ロールバック・セグメント

この項の内容は、次のとおりです。

- 記憶域パラメータの識別
- 表領域内のセグメントのデフォルト記憶域パラメータの設定
- データ・セグメントの記憶域パラメータの設定
- 索引セグメントの記憶域パラメータの設定
- LOB、VARRAY およびネストした表の記憶域パラメータの設定
- 記憶域パラメータの値の変更
- 記憶域パラメータの優先順位の理解
- 記憶域パラメータが領域割当てに影響を与える例

記憶域パラメータの識別

記憶域パラメータは、ディクショナリ管理表領域内で作成されるオブジェクトに対する領域割当てを決定します。ローカル管理表領域には、より単純な領域割当て手段があり、そのコンテキスト内ではほとんどの記憶域パラメータは無効です。

ディクショナリ管理表領域を作成する場合は、デフォルトの記憶域パラメータを指定できます。これらの値によりシステム・デフォルトが上書きされ、その表領域に作成されるオブジェクト専用のデフォルトとなります。記憶域パラメータのデフォルト値は、CREATE または ALTER TABLESPACE 文の DEFAULT STORAGE 句で指定します。

また、ディクショナリ管理表領域内で作成されるオブジェクトの場合は、スキーマ・オブジェクトごとに記憶域パラメータを指定できます。これらのパラメータ設定により、デフォルトの記憶域設定が上書きされます。そのためには、個々のオブジェクトに対して、CREATE または ALTER 文の STORAGE 句を使用して記憶域パラメータを指定します。次の例は、表作成時における記憶域パラメータの指定方法を示しています。

```
CREATE TABLE  players
        (code  NUMBER(10) PRIMARY KEY,
         lastname  VARCHAR(20),
         firstname VARCHAR(15),
         position  VARCHAR2(20),
         team VARCHAR2(20))
      PCTFREE 10
      PCTUSED 40
      STORAGE
         (INITIAL 25K
          NEXT 10K
          MAXEXTENTS 10
          MINEXTENTS 3);
```

すべての記憶域パラメータをどのタイプのデータベース・オブジェクトにも指定できるわけではなく、CREATE 文と ALTER 文の一方でしか指定できない記憶域パラメータもあります。

次の表に、各記憶域パラメータの簡単な説明を示します。デフォルト設定、最小設定および最大設定など、これらのパラメータの詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

パラメータ	説明
INITIAL	セグメントの作成時に割り当てられる第 1 エクステントのサイズ。バイト単位で指定します。このパラメータは、ALTER 文では指定できません。
NEXT	セグメントに次に割り当てられる増分エクステントのサイズ。バイト単位で指定します。第 2 エクステントは、NEXT のオリジナルの設定値と同じサイズになります。それ以降、NEXT は 1 つ前の NEXT のサイズに（1 + PCTINCREASE/100）を掛けたサイズに設定されます。
PCTINCREASE	セグメントに割り当てられた最後の増分エクステントに対する、新しい増分エクステントの増加率。PCTINCREASE が 0（ゼロ）の場合、各増分エクステントのサイズは一定になります。PCTINCREASE が 0（ゼロ）より大きい場合、NEXT は計算されるたびに PCTINCREASE だけ大きくなります。PCTINCREASE に負の値は設定できません。 新しい NEXT は、1 + PCTINCREASE/100 に、最後の増分エクステントのサイズ（古い NEXT）を乗算して、ブロック・サイズの次の倍数に切り上げられます。
MINEXTENTS	セグメントの作成時に割り当てられるエクステントの合計数。これにより、連続した領域が使用できなくても、作成時に大きな領域を割り当てることができます。
MAXEXTENTS	最初のエクステントも含めて、セグメントに割り当てられるエクステントの合計数。
FREELIST GROUPS	作成するデータベース・オブジェクトの空きリストのグループ数。Oracle は、Oracle Real Application Clusters インスタンスのインスタンス番号を使用して、各インスタンスを単一の空きリスト・グループにマッピングします。このパラメータの使用方法については、『Oracle9*i* Real Application Clusters 管理』を参照してください。 **注意 :** セグメント領域管理が AUTO に指定されているローカル管理表領域に作成されたオブジェクトでは、このパラメータは無視されます。
FREELISTS	スキーマ・オブジェクトの各空きリスト・グループの空きリスト数を指定します。このパラメータは表領域には無効です。このパラメータの使用方法は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。 **注意 :** セグメント領域管理が AUTO に指定されているローカル管理表領域に作成されたオブジェクトでは、このパラメータは無視されます。

パラメータ	説明
OPTIMAL	ロールバック・セグメントにのみ関連があります。このパラメータの使用方法については、第 13 章「UNDO 領域の管理」を参照してください。
BUFFER POOL	スキーマ・オブジェクト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。表領域またはロールバック・セグメントには無効です。このパラメータの使用方法については、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。

表領域内のセグメントのデフォルト記憶域パラメータの設定

データベースの表領域ごとにデフォルトの記憶域パラメータを設定できます。表領域にセグメントを作成するとき、または後で変更するときに明示的に設定しなかった記憶域パラメータは、そのセグメントが存在する表領域の対応するデフォルト記憶域パラメータに自動的に設定されます。

表領域レベルで MINEXTENTS を指定すると、表領域に割り当てられるエクステントは最小のエクステント数の倍数に四捨五入されます。

データ・セグメントの記憶域パラメータの設定

非クラスタ化表、マテリアライズド・ビューまたはマテリアライズド・ビュー・ログのデータ・セグメントに対する記憶域パラメータは、表、マテリアライズド・ビューまたはマテリアライズド・ビュー・ログの CREATE 文または ALTER 文の STORAGE 句を使用して設定します。

それに対して、クラスタのデータ・セグメントに対する記憶域パラメータは、CREATE CLUSTER 文または ALTER CLUSTER 文の STORAGE 句を使用して設定します。クラスタ内に表やマテリアライズド・ビューを設定する個々の CREATE 文または ALTER 文には STORAGE 句は指定しません。クラスタ化された表やマテリアライズド・ビューの作成時または変更時に指定された記憶域パラメータは、無視されます。クラスタに設定された記憶域パラメータは、表の記憶域パラメータを上書きします。

パーティション表では、表レベルでデフォルト記憶域パラメータを設定できます。表のパーティションの新規作成時に個々のパーティションに対してデフォルト記憶域パラメータを指定しない場合は、それらのパラメータが表レベルから継承されます。表レベルの記憶域パラメータが指定されていない場合は、表領域から継承されます。

索引セグメントの記憶域パラメータの設定

表の索引のために作成される索引セグメントの記憶域パラメータは、CREATE INDEX 文または ALTER INDEX 文の STORAGE 句を使用して設定できます。

索引を主キー制約または一意キー制約を規定するために使用する場合、その索引のために作成する索引セグメントの記憶域パラメータは、次のどちらかの方法で設定できます。

- CREATE TABLE 文または ALTER TABLE 文の ENABLE ... USING INDEX 句
- ALTER INDEX 文の STORAGE 句

LOB、VARRAY およびネストした表の記憶域パラメータの設定

表またはマテリアライズド・ビューには、LOB、VARRAY およびネストした表の列タイプを格納できます。これらのエントリは、専用セグメントに格納できます。LOB と VARRAY は LOB セグメントに格納されますが、ネストした表は記憶表に格納されます。これらのセグメントに対して STORAGE 句を指定し、表レベルで指定した記憶域パラメータを上書きできます。

関連項目： LOB、VARRAY およびネストした表を含む表の作成方法は、次のマニュアルを参照してください。

- 『Oracle9*i* アプリケーション開発者ガイドーラージ・オブジェクト』
- 『Oracle9*i* アプリケーション開発者ガイドー基礎編』
- 『Oracle9*i* SQL リファレンス』

記憶域パラメータの値の変更

必要に応じて、表領域に対するデフォルト記憶域パラメータと、個々のセグメントに対する特定の記憶域パラメータを変更できます。デフォルトの記憶域パラメータは、表領域用のパラメータにリセットできます。ただし、変更は、その表領域に作成される新しいオブジェクトまたはセグメントに割り当てられる新しいエクステントにしか反映されません。

既存の表、クラスタ、索引またはロールバック・セグメントの INITIAL 記憶域パラメータと MINEXTENTS 記憶域パラメータは変更できません。セグメントの NEXT のみを変更すると、次の増分エクステントは新しい NEXT のサイズとなり、後続するエクステントは、通常どおりに PCTINCREASE だけ大きくなります。

セグメントの NEXT と PCTINCREASE を両方とも変更すると、次のエクステントは NEXT の新しい値となり、それ以降は通常どおり PCTINCREASE を使用して NEXT が計算されます。

記憶域パラメータの優先順位の理解

ある時点で有効な記憶域パラメータは、SQL 文のタイプによって決まります。次に、SQL 文を優先順位の高いものから順に記述します（小さい番号のほうが大きい番号よりも優先されます）。

1. ALTER [TABLE|CLUSTER|MATERIALIZED VIEW|MATERIALIZED VIEW LOG|INDEX|ROLLBACK] SEGMENT 文
2. CREATE [TABLE|CLUSTER|MATERIALIZED VIEW|MATERIALIZED VIEW LOG|INDEX|ROLLBACK] SEGMENT 文
3. ALTER TABLESPACE 文
4. CREATE TABLESPACE 文
5. Oracle のデフォルト値

オブジェクト・レベルで指定された記憶域パラメータは、表領域レベルで設定された対応するオプションを上書きします。記憶域パラメータがオブジェクト・レベルで明示的に設定されていない場合は、表領域レベルでデフォルトが適用されます。記憶域パラメータが表領域レベルで設定されていないときは、Oracle のシステム・デフォルトが適用されます。記憶域パラメータを変更すると、新しいオプションは、まだ割り当てられていないエクステントにのみ適用されます。

注意： 一時セグメントの記憶域パラメータは、対応する表領域のために設定されたデフォルト記憶域パラメータを常に使用します。

記憶域パラメータが領域割当てに影響を与える例

次の文が実行されたと想定します。

```
CREATE TABLE test_storage
   ( . . . )
   STORAGE (INITIAL 100K   NEXT 100K
      MINEXTENTS 2   MAXEXTENTS 5
      PCTINCREASE 50);
```

また、初期化パラメータ DB_BLOCK_SIZE は 2KB に設定されているものとします。次の表は、エクステントが test_storage 表に割り当てられる様子を示しています。また、増分エクステントの値も示します。この値は、USER_SEGMENTS データ・ディクショナリ・ビューまたは DBA_SEGMENTS データ・ディクショナリ・ビューの NEXT 列で参照できます。

表 14-1　エクステントの割当て

エクステント数	エクステント・サイズ	NEXT の値
1	50 ブロックまたは 102400 バイト	50 ブロックまたは 102400 バイト
2	50 ブロックまたは 102400 バイト	75 ブロックまたは 153600 バイト
3	75 ブロックまたは 153600 バイト	113 ブロックまたは 231424 バイト
4	115 ブロックまたは 235520 バイト	170 ブロックまたは 348160 バイト
5	170 ブロックまたは 348160 バイト	NEXT の値はなし、MAXEXTENTS=5

NEXT または PCTINCREASE 記憶域パラメータを ALTER 文（ALTER TABLE など）で変更すると、データ・ディクショナリに格納されている現行の値が指定した値に置き換えられます。たとえば、3 番目のエクステントが表に割り当てられる前に、次の文によって test_storage 表の NEXT 記憶域パラメータを変更するとします。

```
ALTER TABLE test_storage STORAGE (NEXT 500K);
```

その結果、3 番目のエクステントは割当て時に 500KB となり、それ以降、4 番目のエクステントは (500KB*1.5) = 750KB のように計算されます。

再開可能領域割当ての管理

Oracle では、領域割当てが失敗した場合に大規模なデータベース処理を一時停止し、後で再開するための方法が提供されています。これにより、Oracle データベースがユーザーにエラーを返すかわりに対処措置を講じることができます。エラー条件が訂正されると、一時停止していた処理が自動的に再開します。この機能のことを、**再開可能領域割当て**と呼びます。また、影響を受ける文のことを、再開可能文と呼びます。

この項の内容は、次のとおりです。

- 再開可能領域割当ての概要
- 再開可能領域割当ての有効化および無効化
- 一時停止文の検出
- 再開可能領域割当ての例 : AFTER SUSPEND トリガーの登録

再開可能領域割当ての概要

ここでは、再開可能領域割当ての概要について説明します。再開可能文の動作について説明し、修飾文とエラー条件を具体的に定義します。

再開可能文の動作

再開可能文の動作の概要は、次のとおりです。詳細はこの後の各項で説明します。

1. クライアントが ALTER SESSION 文を使用してセッションの再開可能セマンティクスを明示的に有効にしたときのみ、文が再開可能モードで実行されます。
2. 次のいずれかの条件が成立すると、再開可能文が一時停止します（非再開可能文では、これらの条件に対応するエラーが通知されます）。
   - 領域不足条件
   - 最大エクステント数到達条件
   - スペース割当制限超過条件
3. 再開可能文の実行が一時停止すると、ユーザー指定の操作の実行、エラーの記録および文の実行状態の問合せを行うメカニズムがただちに動作します。再開可能文が一時停止すると、次の処理が実行されます。
   - エラーがアラート・ログに記録されます。
   - ユーザーが AFTER SUSPEND システム・イベントに対してトリガーを登録していた場合は、そのユーザー・トリガーが実行されます。ユーザー指定の PL/SQL プロシージャは、DBMS_RESUMABLE パッケージと DBA/USER_RESUMABLE ビューを使用して、エラー・メッセージ・データにアクセスできます。
4. 文の一時停止によって自動的にトランザクションが一時停止します。その結果、すべてのトランザクション・リソースが文の一時停止から再開までの間保持されます。
5. ユーザーの介入や他の問合せによってソート領域が解放されるなどの結果としてエラー条件がなくなると、一時停止していた文が自動的に実行を再開します。
6. 一時停止した文は、DBMS_RESUMABLE.ABORT() プロシージャを使用して、強制的に例外を発生できます。このプロシージャは、DBA または文を発行したユーザーがコールできます。
7. 一時停止のタイムアウト間隔は、再開可能文に対応付けられています。タイムアウト間隔（デフォルトは 2 時間）の間一時停止していた再開可能文がリストアすると、ユーザーに例外が返されます。
8. 再開可能文は、実行中に一時停止と再開を複数回繰り返すことができます。

再開可能な操作

注意： 再開可能領域割当ては、ローカル管理表領域の使用時に完全にサポートされます。ディクショナリ管理表領域の使用時には、特定の制限事項があります。詳細は、14-17 ページの「ディクショナリ管理表領域に対する再開可能領域割当ての制限」を参照してください。

再開可能な操作は、次のとおりです。

- 問合せ

  一時領域（ソート領域）を使い果たした SELECT 文は、再開可能な実行の候補になります。Oracle Call Interface（OCI）の使用時は、LNOCIStmtExecute() および LNOCIStmtFetch() の各コールが候補になります。

- DML

  INSERT、UPDATE および DELETE の各文が候補になります。各文の実行に使用されているインタフェースには関係ありません。OCI、JSQL、PL/SQL やその他のインタフェースでも有効です。外部表からの INSERT INTO ... SELECT も再開可能にできます。

- インポート / エクスポート

  SQL*Loader については、リカバリ可能なエラーの後に文が再開可能かどうかをコマンドライン・パラメータで制御します。

- DDL

  次の文が、再開可能な実行の候補になります。

  - CREATE TABLE ... AS SELECT
  - CREATE INDEX
  - ALTER INDEX ... REBUILD
  - ALTER TABLE ... MOVE PARTITION
  - ALTER TABLE ... SPLIT PARTITION
  - ALTER INDEX ... REBUILD PARTITION
  - ALTER INDEX ... SPLIT PARTITION
  - CREATE MATERIALIZED VIEW
  - CREATE MATERIALIZED VIEW LOG

訂正可能なエラー

訂正可能なエラーには、次の 3 つのクラスがあります。

- 領域不足条件

  データベースの操作によって、表領域内の表、索引、一時セグメント、ロールバック・セグメント、UNDO セグメント、クラスタ、LOB、表パーティションまたは索引パーティション用のエクステントをこれ以上取得できません。たとえば、次のエラーはこのカテゴリに属します。

  ```
  ORA-01650 ロールバック・セグメント ... を拡張できません（... 分、表領域 ...）。
  ORA-01653 表 ... を拡張できません（... 分、表領域 ...）。
  ORA-01654 索引 ... を拡張できません（... 分、表領域 ...）。
  ```

- 最大エクステント数到達条件

  表、索引、一時セグメント、ロールバック・セグメント、UNDO セグメント、クラスタ、LOB、表パーティションまたは索引パーティション内のエクステント数が、オブジェクトで定義されている最大エクステント数に等しくなりました。たとえば、次のエラーはこのカテゴリに属します。

  ```
  ORA-01628 最大エクステント数（...）に達しました（ロールバック・セグメント ...）
  ORA-01631 最大エクステント数（...）に達しました（表 ...）
  ORA-01654 最大エクステント数（...）に達しました（索引 ...）
  ```

- スペース割当制限超過条件

  ユーザーが自分に割り当てられている表領域内のスペース割当制限を超過しました。具体的には、次のエラーによって示されます。

  ```
  ORA-01536 表領域 string に対して割り当てられた領域を使い果たしました。
  ```

ディクショナリ管理表領域に対する再開可能領域割当ての制限

ディクショナリ管理表領域の使用時には、再開可能領域割当てに対して特定の制限があります。それらの制限は、次のとおりです。

1. CREATE TABLE や CREATE INDEX などのデータ定義言語（DDL）操作を、実行中に領域不足エラーの原因となる明示的な MAXEXTENTS 設定で実行した場合、操作は一時停止しません。この場合は、操作が強制終了します。このエラーは修復不可能として扱われます。これは、MAXEXTENTS などのオブジェクトのプロパティをオブジェクトの作成前に変更できないためです。しかし、DML 操作によって、すでに存在している表や索引が MAXEXTENTS 上限に達した場合には、操作は一時停止し、後で再開できます。MAXEXTENTS 句を UNLIMITED に設定するか、またはローカル管理表領域を使用することにより、このような制限を回避できます。

2. ロールバック・セグメントがディクショナリ管理表領域にある場合、ロールバック・セグメントに対する領域割当ては再開可能ではありません。しかし、ユーザー・オブジェクト（表や索引など）に対する領域割当ては、引き続き再開可能です。この制限を回避するために、自動 UNDO 管理を使用するか、またはロールバック・セグメントをローカル管理表領域に配置することをお薦めします。

再開可能文と分散操作

リモート操作は、再開可能モードではサポートされていません。

パラレル実行と再開可能文

パラレル実行では、パラレル実行サーバー・プロセスの 1 つで訂正可能なエラーが発生した場合、そのサーバー・プロセスの実行が一時停止します。他のパラレル実行サーバー・プロセスでは、エラーが発生するまで、または一時停止したサーバー・プロセスに（直接または間接に）ブロックされるまで、個々のタスクの実行が継続されます。訂正可能なエラーが解決すると、一時停止したプロセスが実行を再開し、パラレル操作の実行は継続されます。一時停止したプロセスが終了した場合、パラレル操作は中断し、ユーザーに対してエラーが返されます。

異なるパラレル実行プロセスで、1 つ以上の訂正可能なエラーが発生することがあります。この結果、AFTER SUSPEND トリガーが複数回、パラレルに発行される場合があります。また、あるパラレル実行サーバー・プロセスが一時停止中に他のパラレル実行サーバー・プロセスで訂正不可能なエラーが発生すると、一時停止していた文はただちに強制終了します。

パラレル実行については、すべてのパラレル実行コーディネータ・プロセスとサーバー・プロセスが DBA/USER_RESUMABLE ビューに独自のエントリを持っています。

再開可能領域割当ての有効化および無効化

再開可能領域割当ては、再開可能モードを有効にしたセッションの中で文を実行するときのみ可能です。

セッションの再開可能モードを有効化するには、次の SQL 文を使用します。

```
ALTER SESSION ENABLE RESUMABLE;
```

一時停止した文はなんらかのシステム・リソースを保持している可能性があるため、ユーザーが再開可能モードを有効化し、再開可能文を実行するには、RESUMABLE システム権限が付与されている必要があります。

再開可能モードを無効化するには、次の文を使用します。

```
ALTER SESSION DISABLE RESUMABLE;
```

新しいセッションでは、デフォルトで再開可能モードが無効になっています。

タイムアウト間隔や、再開可能文の識別に使用する名前を指定することもできます。次の項では、各操作について説明します。

関連項目： 14-19 ページ「デフォルトの再開可能モードの設定」

タイムアウト間隔の指定

セッションの再開可能モードを有効化するときに、タイムアウト間隔を指定することもできます。この時間が経過するまでに介入操作がまったく発生しなかった場合、一時停止した文はエラーになります。次の文は、再開可能トランザクションが 3600 秒後にタイムアウトし、エラーになることを指定します。

```
ALTER SESSION ENABLE RESUMABLE TIMEOUT 3600;
```

TIMEOUT の値は、別の ALTER SESSION ENABLE RESUMABLE 文や他の手段によって変更されるまで、またはセッションが終了するまで有効です。デフォルトのタイムアウト間隔は、7200 秒です。

関連項目： 再開可能文のタイムアウト間隔を変更するその他の方法の詳細は、14-20 ページの「タイムアウト間隔の変更」を参照してください。

再開可能文の命名

再開可能文は、名前で識別するように設定できます。次の文は、再開可能文に名前を割り当てます。

```
ALTER SESSION ENABLE RESUMABLE TIMEOUT 3600 NAME 'insert into table';
```

NAME の値は、別の ALTER SESSION ENABLE RESUMABLE 文によって変更されるまで、またはセッションが終了するまで有効です。NAME のデフォルト値は、次のとおりです。

```
User USERNAME(USERID), Session SESSIONID, Instance INSTANCEID
```

文の名前は、DBA_RESUMABLE ビューおよび USER_RESUMABLE ビューで再開可能文を識別する際に使用します。

デフォルトの再開可能モードの設定

デフォルトの再開可能モードを設定するために、DBA はデータベース・レベルの LOGON トリガーを登録して、ユーザーのセッションの変更、再開可能モードの有効化およびタイムアウト間隔の設定を実行できます。

注意： 再開可能文のデフォルトのモードとタイムアウトを変更する登録済みトリガーが複数ある場合、その結果は予測できません。これは、トリガーの起動順序が保証されていないためです。

タイムアウト間隔の変更

ALTER SESSION ENABLE RESUMABLE 文以外にも、タイムアウト間隔を設定または変更する方法があります。

DBMS_RESUMABLE パッケージには、特定のセッションまたは現行セッションのタイムアウト間隔を設定するためのプロシージャがあります。DBA は、DBMS_RESUMABLE をコールしてデフォルトのシステム・タイムアウトを設定する、システム全体で有効な AFTER SUSPEND トリガーを作成することにより、デフォルトのシステム・タイムアウトを変更できます。たとえば、次のコード例は、システム全体で有効なデフォルト・タイムアウトを 1 時間に設定します。

```
CREATE OR REPLACE TRIGGER resumable_default_timeout
AFTER SUSPEND
ON DATABASE
BEGIN
   DBMS_RESUMABLE.SET_TIMEOUT(3600);
END;
```

一時停止文の検出

再開可能文が一時停止するとき、クライアントにはエラーは通知されません。訂正処理を実行するため、Oracle ではユーザーにエラーを通知して状況に関する情報を提供するかわりの手段が提供されています。

AFTER SUSPEND システム・イベントおよびトリガー

再開可能文で訂正可能なエラーが発生すると、システムは内部的に AFTER SUSPEND システム・イベントを生成します。ユーザーは、このイベントに対するトリガーをデータベース・レベルとスキーマ・レベルの両方で登録できます。ユーザーがこのシステム・イベントを処理するトリガーを登録した場合、トリガーは SQL 文が一時停止した後に実行されます。

AFTER SUSPEND トリガー内部で実行された SQL 文は、再開不可能であり、かつ自律型です。トリガー内部で開始されたトランザクションは、SYSTEM ロールバック・セグメントを使用します。これらの条件が課されるのは、デッドロックを回避し、文と同じエラー条件に直面する可能性を少なくするためです。

ユーザーは USER_RESUMABLE ビュー、DBA_RESUMABLE ビューまたは DBMS_RESUMABLE.SPACE_ERROR_INFO ファンクションをトリガー内部で使用して、再開可能文に関する情報を取得できます。

また、トリガーでは DBMS_RESUMABLE パッケージをコールして、一時停止した文の強制終了や再開可能タイムアウト値の変更を実行できます。

> **関連項目：** システム・イベント、トリガーおよび属性ファンクションの詳細は、『Oracle9*i* アプリケーション開発者ガイド－基礎編』を参照してください。

再開可能文に関する情報を含むビュー

次のビューを問い合せることにより、再開可能文の状態に関する情報を取得できます。

ビュー	説明
DBA_RESUMABLE USER_RESUMABLE	これらのビューには、現在実行中または一時停止しているすべての再開可能文に関する行が含まれます。これらは、再開可能文の実行状態を監視したり、再開可能文に関する特定の情報を取得するために、DBA、AFTER SUSPEND トリガーまたは別のセッションが使用できます。
V$SESSION_WAIT	ある文が一時停止すると、その文を起動したセッションは待機状態になります。このビューには、「文が一時停止され、エラーの消去を待機しています」という内容の EVENT 列を持つセッションに対して 1 行が挿入されます。

関連項目： これらのビューに含まれる列の詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

DBMS_RESUMABLE パッケージ

DBMS_RESUMABLE パッケージは、再開可能文の管理に役立ちます。次のプロシージャが使用できます。

プロシージャ	説明
ABORT(sessionID)	このプロシージャは、一時停止した再開可能文を強制終了します。パラメータ sessionID は、文が実行されているセッション ID です。パラレル DML/DDL の場合、sessionID はパラレル DML/DDL に参加している任意のセッション ID です。 ABORT 操作は常に正常終了することが保証されています。ABORT は、AFTER SUSPEND トリガーの内部または外部のどちらでもコールできます。 ABORT のコール元は、sessionID を持つセッションの所有者、ALTER SYSTEM 権限を持っているユーザー、DBA 権限を持っているユーザーのいずれかである必要があります。
GET_SESSION_ TIMEOUT(sessionID)	このファンクションは、sessionID を持つセッションで設定されている再開可能文の現行のタイムアウト値を返します。タイムアウトは秒数で返されます。セッションが存在しない場合、このファンクションは -1 を返します。
SET_SESSION_ TIMEOUT(sessionID, timeout)	このプロシージャは、sessionID を持つセッションに対して再開可能文のタイムアウト間隔を設定します。パラメータ timeout の単位は秒数です。新しい timeout 設定は、即時にセッションに適用されます。セッションが存在しない場合は何も起こりません。

プロシージャ	説明
GET_TIMEOUT()	このファンクションは、現行セッションで設定されている再開可能文の現行の timeout 値を返します。値は秒数で返されます。
SET_TIMEOUT(timeout)	このプロシージャは、現行セッションに対して再開可能文の timeout 値を設定します。パラメータ timeout の単位は秒数です。新しいタイムアウト設定は、即時にセッションに適用されます。

関連項目：『Oracle9*i* 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』

再開可能領域割当ての例 : AFTER SUSPEND トリガーの登録

次の例では、システム全体で有効な AFTER SUSPEND トリガーを作成し、ユーザー SYS としてデータベース・レベルで登録します。任意のセッションで再開可能文が一時停止すると、このトリガーは次の 2 つのうちどちらかの処理を実行します。

- ロールバック・セグメントが領域上限に達した場合は、メッセージが DBA に送られ、文が強制終了します。
- 他のリカバリ可能なエラーが発生した場合には、タイムアウト間隔が 8 時間にリセットされます。

この例で使用する文を次に示します。

```
CREATE OR REPLACE TRIGGER resumable_default
AFTER SUSPEND
ON DATABASE
DECLARE
   /* declare transaction in this trigger is autonomous */
   /* this is not required because transactions within a trigger
      are always autonomous */
   PRAGMA AUTONOMOUS_TRANSACTION;
   cur_sid           NUMBER;
   cur_inst          NUMBER;
   errno             NUMBER;
   err_type          VARCHAR2;
   object_owner      VARCHAR2;
   object_type       VARCHAR2;
   table_space_name  VARCHAR2;
   object_name       VARCHAR2;
   sub_object_name   VARCHAR2;
   error_txt         VARCHAR2;
   msg_body          VARCHAR2;
```

```
   ret_value         BOOLEAN;
   mail_conn         UTL_SMTP.CONNECTION;
BEGIN
   -- Get session ID
   SELECT DISTINCT(SID) INTO cur_SID FROM V$MYSTAT;

   -- Get instance number
   cur_inst := userenv('instance');

   -- Get space error information
   ret_value :=
   DBMS_RESUMABLE.SPACE_ERROR_INFO(err_type,object_type,object_owner,
        table_space_name,object_name, sub_object_name);
   /*
   -- If the error is related to rollback segments, log error, send email
   -- to DBA, and abort the statement. Otherwise, set timeout to 8 hours.
   --
   -- sys.rbs_error is created by DBA manually and defined as
   -- sql_text VARCHAR2(1000), error_msg VARCHAR2(4000),
   -- suspend_time DATE)
   */

   IF OBJECT_TYPE = 'ROLLBACK SEGMENT' THEN
       /* LOG ERROR */
       INSERT INTO sys.rbs_error (
           SELECT SQL_TEXT, ERROR_MSG, SUSPEND_TIME
           FROM DBMS_RESUMABLE
           WHERE SESSION_ID = cur_sid AND INSTANCE_ID = cur_inst
        );
       SELECT ERROR_MSG INTO error_txt FROM DBMS_RESUMABLE
           WHERE SESSION_ID = cur_sid and INSTANCE_ID = cur_inst;

        -- Send email to receipient via UTL_SMTP package
        msg_body:='Subject: Space Error Occurred

                   Space limit reached for rollback segment ' || object_name ||
                   on ' || TO_CHAR(SYSDATE, 'Month dd, YYYY, HH:MIam') ||
                   '. Error message was ' || error_txt;

        mail_conn := UTL_SMTP.OPEN_CONNECTION('localhost', 25);
        UTL_SMTP.HELO(mail_conn, 'localhost');
        UTL_SMTP.MAIL(mail_conn, 'sender@localhost');
        UTL_SMTP.RCPT(mail_conn, 'recipient@localhost');
        UTL_SMTP.DATA(mail_conn, msg_body);
        UTL_SMTP.QUIT(mail_conn);

        -- Abort the statement
```

```
        DBMS_RESUMABLE.ABORT(cur_sid);
    ELSE
        -- Set timeout to 8 hours
        DBMS_RESUMABLE.SET_TIMEOUT(28800);
    END IF;

    /* commit autonomous transaction */
    COMMIT;
END;
```

領域の割当て解除

セグメントに領域を割り当てたものの、その領域が使用されていない場合があります。たとえば、PCTINCREASE を高い値に設定したことによって、一部しか使用されない大きなエクステントが作成されることがあります。また、ALTER TABLE ... ALLOCATE EXTENT 文を発行して、明示的に必要以上の領域を割り当てる可能性もあります。未使用の領域または必要以上に割り当てた領域があることが判明した場合は、その領域を解放して、未使用領域を他のセグメントで使用可能にすることができます。

ここでは、未使用領域の割当て解除について説明します。

最高水位標の表示

割当てを解除する前に、DBMS_SPACE パッケージを使用できます。このパッケージには、最高水位標の位置とセグメント内の未使用領域の量についての情報を返すプロシージャ（UNUSED_SPACE）が含まれています。

最高水位標は、セグメント内の使用済み領域、つまりデータを受け取るためにフォーマットされている領域の量を示します。割当てを解除する領域にデータが存在しなくても、最高水位標より下の領域は解放できません。ただし、セグメントが完全に空の場合は、TRUNCATE ... DROP STORAGE 文を使用して領域を解放できます。

セグメント領域管理が AUTO に指定されているローカル管理表領域内のセグメントでは、次の出力パラメータによって最高水位標が決まりますが、その意味は少し変更されています。

- LAST_USED_EXTENT_FILE_ID
- LAST_USED_EXTENT_BLOCK_ID
- LAST_USED_BLOCK

特に、最高水位標より下のブロックがフォーマット解除される可能性があります。セグメント領域管理が AUTO に指定されている場合、DBMS_SPACE の UNUSED_SPACE プロシージャと FREE_SPACE プロシージャはどちらも未使用領域を正確に表しません。かわりに SPACE_USAGE プロシージャを使用してください。

関連項目： DBMS_SPACE パッケージの詳細は、『Oracle9*i* 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。

領域割当て解除文の発行

次の文は、セグメント（表、索引またはクラスタ）内の未使用領域の割当てを解除します。KEEP 句はオプションです。

```
ALTER TABLE table DEALLOCATE UNUSED KEEP integer;
ALTER INDEX index DEALLOCATE UNUSED KEEP integer;
ALTER CLUSTER cluster DEALLOCATE UNUSED KEEP integer;
```

未使用領域の量を KEEP に明示的に指定すると、残りの未使用領域の割当てが解除されても、この領域は保持されます。残りのエクステントの数が MINEXTENTS よりも少なくなると、MINEXTENTS の値は新しい数を反映するように変更されます。初期エクステントが小さくなった場合は、初期エクステントの新しいサイズを反映するように INITIAL の値が変更されます。

KEEP 句を指定しない場合は、初期エクステントのサイズと MINEXTENTS が保たれるかぎり、すべての未使用領域（最高水位標より上のすべて）の割当てが解除されます。したがって、最高水位標が MINEXTENTS の境界の内部にある場合でも、MINEXTENTS は保持され、初期エクステントのサイズは減少しません。

DBA_FREE_SPACE ビューを調べることにより、割当て解除によって解放された領域を確認できます。

関連項目：

- 未使用領域の割当て解除に関連する構文とオプションの詳細は、『Oracle9*i* SQL リファレンス』を参照してください。
- DBA_FREE_SPACE ビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

割当て解除の例

ここでは、領域の割当てを解除する例をいくつか示します。

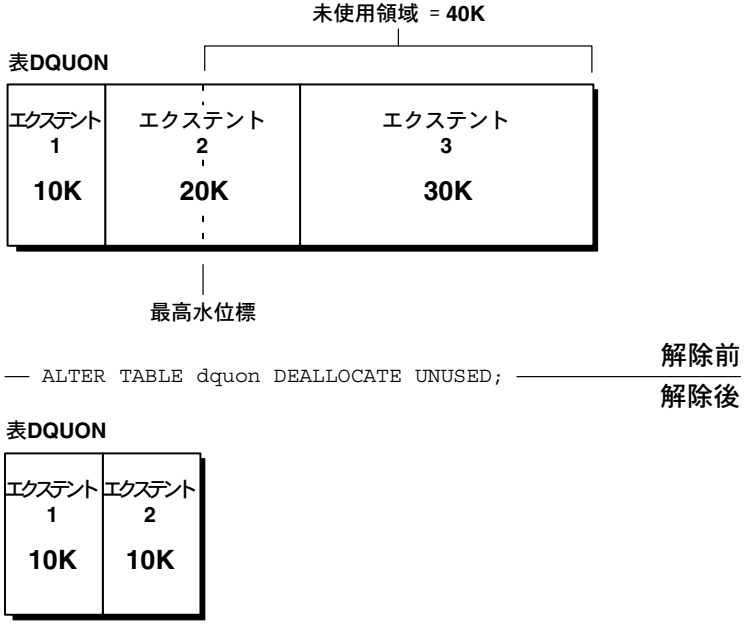
領域の割当て解除の例 1:

表は 3 つのエクステントから構成されています。第 1 エクステントは 10KB、第 2 エクステントは 20KB、第 3 エクステントは 30KB です。最高水位標は第 2 エクステントの中央にあり、40KB の未使用領域があります。図 14-3 に、次の文を発行した場合の効果を示します。

```
ALTER TABLE dquon DEALLOCATE UNUSED;
```

すべての未使用領域の割当てが解除され、表 dquon には 2 つのエクステントが残ります。第 3 エクステントはなくなり、第 2 エクステントのサイズは 10KB になります。

図 14-3　すべての未使用領域の割当てを解除する



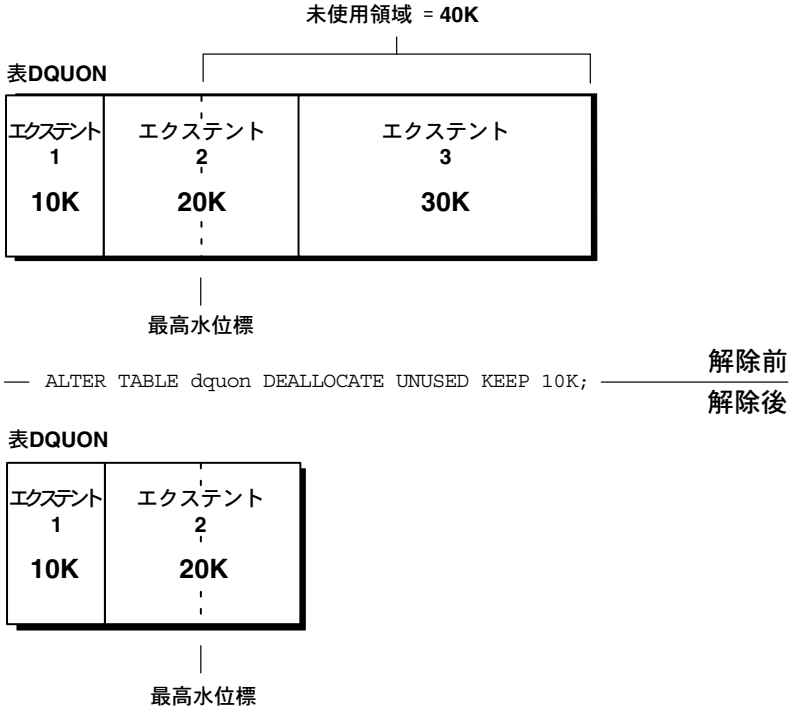
ただし、KEEP キーワードを指定して次の文を発行すると、最高水位標の上の 10KB は残り、残りの未使用領域は dquon から割当て解除されます。

```
ALTER TABLE dquon DEALLOCATE UNUSED KEEP 10K;
```

実際には、第 3 エクステントの割当てが解除され、第 2 エクステントはそのまま残ります。

図 14-4 に、この状況を示します。

図 14-4　未使用領域の割当てを解除し、KEEP 10K を指定する



次の文のように、dquon のすべての未使用領域の割当てを解除して KEEP 20K を指定すると、第 3 エクステントは 10KB に縮小され、第 2 エクステントのサイズは変わりません。

```
ALTER TABLE dquon DEALLOCATE UNUSED KEEP 20K;
```

領域の割当て解除の例 2:

図 14-3 の状況を考えます。第 3 エクステントの割当ては完全に解除され、第 2 エクステントには 10KB 残ります。また、次に割り当てられるエクステントのデフォルトのサイズは、最後に完全に割当て解除されたエクステントのサイズ、つまり、この場合は 30KB に設定されます。この設定を変更する場合は、ALTER TABLE 文の STORAGE 句に NEXT の新しい値を指定して、次のエクステントのサイズを明示的に設定します。

次の文は、表 dquon の次のエクステントのサイズを 20KB に設定します。

```
ALTER TABLE dquon STORAGE (NEXT 20K);
```

領域の割当て解除の例 3:

DEALLOCATE では、MINEXTENTS に設定した数のエクステントを保つため、そのセグメントに割り当てられていた元のエクステントを保持できます。この容量は、前述のように KEEP パラメータの影響を受けます。

表 dquon の MINEXTENTS 値が 2 の場合は、図 14-3 と図 14-4 の各文の結果は変わらず、MINEXTENTS の初期値が保たれます。

ただし、MINEXTENTS の値が 3 の場合、図 14-4 の文の結果は同じ（第 3 エクステントが削除される）ですが、MINEXTENTS の値は 2 に変更されます。また、図 14-3 の文では異なる結果になります。この場合、この文は効果がありません。

データ型の領域使用の理解

表またはその他のデータ構造を作成する際には、必要となる領域の大きさを把握しておく必要があります。領域要件は、データ型ごとに異なります。データ型とその領域要件の詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』および『Oracle9*i* SQL リファレンス』を参照してください。

15

表の管理

この章では、表を管理する方法について説明します。この章の内容は、次のとおりです。

- 表を管理するためのガイドライン
- 表の作成
- 表の変更
- 表のオンライン再定義
- 表の削除
- 索引構成表の管理
- 外部表の管理
- 表情報の表示

関連項目：

- この章のタスクを実行する前に、第 14 章「スキーマ・オブジェクトの領域の管理」を一読されることをお薦めします。
- 整合性制約の指定や表の分析など、表の管理に関するその他の情報については、第 21 章「スキーマ・オブジェクトの一般的な管理」を参照してください。
- パーティション表については、第 17 章「パーティション表と索引の管理」を参照してください。

表を管理するためのガイドライン

ここでは、表を管理するときに従うべきガイドラインについて説明します。これらのガイドラインに従うことで、表の作成やその後の問合せまたは更新を行うときに、表の管理が容易になり、パフォーマンスの向上にもつながります。

この項の内容は、次のとおりです。

- 作成前の表の設計
- データ・ブロック領域の使用方法の指定
- 各表の位置の指定
- 表作成のパラレル化
- 表作成時の NOLOGGING の使用
- 表のサイズの見積りと記憶域パラメータの設定
- 大規模な表の計画
- 表の制限事項

作成前の表の設計

通常、アプリケーション開発者は、表などのアプリケーションの要素を設計する必要があります。データベース管理者（DBA）は、アプリケーション開発者から得たアプリケーションの動作と予想されるデータのタイプに関する情報に基づいて、記憶域パラメータを設定し、表のクラスタを定義する必要があります。

次の考慮事項に従って、アプリケーション開発者と共同で綿密に各表の計画を作成してください。

- 表を正規化します。
- 各列に適切なデータ型を割り当てます。
- 記憶域を節約するために、NULL を許可する列を最後に定義します。
- 記憶域を節約し、SQL 文のパフォーマンスを最適化するために、適切であれば必ず表をクラスタ化します。クラスタ化表については、第 18 章「クラスタの管理」を参照してください。

データ・ブロック領域の使用方法の指定

個々の表を作成するときに、PCTFREE パラメータと PCTUSED パラメータを指定することによって、表のデータ・セグメントのデータ・ブロック内での領域使用の効率、およびカレント・データの更新に使用する予約領域を調整できます。PCTFREE および PCTUSED パラメータについては、14-2 ページの「データ・ブロックの領域管理」を参照してください。

注意： 自動セグメント領域管理が使用可能になっているローカル管理表領域に表を作成する場合、PCTFREE（または FREELISTS）パラメータを指定する必要はありません。自動セグメント領域管理は、表領域レベルで指定します。Oracle データベースでは、この種の表領域に作成されたオブジェクト内の空き領域と使用済み領域が自動的かつ効率的に管理されます。

ローカル管理表領域と自動セグメント領域管理の詳細は、15-2 ページの「ローカル管理表領域」を参照してください。

各表の位置の指定

適切な権限と表領域割当て制限を持つユーザーであれば、オンライン表領域内に新しい表を作成できます。新しい表を格納する表領域を識別するには、CREATE TABLE 文に TABLESPACE 句を指定します。CREATE TABLE 文で表領域を指定しない場合は、作成したユーザーのデフォルト表領域内に表が作成されます。

新しい表を含む表領域を指定するときには、その選択が意味することを必ず理解しておいてください。各表の作成時に表領域を適切に指定することによって、次のことが可能になります。

- データベース・システムのパフォーマンスの向上
- データベース管理に必要な時間の短縮

次のように、スキーマ・オブジェクトの記憶域の位置を適切に指定しない場合は、データベースに悪影響が及びます。

- ユーザーのオブジェクトを SYSTEM 表領域に作成すると、データ・ディクショナリ・オブジェクトとユーザー・オブジェクトの両方が同じデータ・ファイルを求めて競合し、Oracle のパフォーマンスが低下するおそれがあります。
- アプリケーションに関係する表をいろいろな表領域に無計画に格納すると、DBA がアプリケーションのデータ管理操作（バックアップやリカバリなど）に要する時間が増大する可能性があります。

ユーザーにデフォルトの表領域および表領域割当て制限を割り当てる方法については、第 24 章「ユーザーとリソースの管理」を参照してください。

表作成のパラレル化

CREATE TABLE 文で副問合せ（AS SELECT）を使用して表を作成する際は、パラレル実行を利用できます。複数のプロセスが同時に動作して表を作成するため、表を作成するときのパフォーマンスが向上します。

表作成のパラレル化については、15-8 ページの「表作成のパラレル化」を参照してください。

表作成時の NOLOGGING の使用

表を最も効率よく作成するには、CREATE TABLE ... AS SELECT 文で NOLOGGING 句を使用します。NOLOGGING 句を指定すると、表の作成中に最小限の REDO 情報しか生成されません。これには、次のような利点があります。

- REDO ログ・ファイルの領域を節約できます。
- 表の作成に要する時間が削減できます。
- 大規模な表のパラレル作成のパフォーマンスが向上します。

また、NOLOGGING 句を指定することで、SQL*Loader を使用した後続のダイレクト・ロードおよびダイレクト・ロード INSERT 操作がロギングされなくなります。後続のデータ操作文（DML）文（UPDATE、DELETE および従来型パスの挿入）は、表の NOLOGGING 属性の影響を受けず、REDO を生成します。

表の作成後にその表の損失（たとえば、表の作成に使用したデータにアクセスできなくなるなど）を避ける必要がある場合は、作成直後に表のバックアップを取得してください。一時的に使用するために作成する表など、そのような予防策が不要な場合もあります。

一般に、NOLOGGING を指定して表を作成するときは、小規模な表より大規模な表のほうが相対的にパフォーマンスの向上が大きくなります。小規模な表の場合は、NOLOGGING を指定しても、表作成に要する時間にほとんど影響はありません。一方、大規模な表では、特に表作成をパラレル化したときにパフォーマンスが著しく向上します。

表のサイズの見積りと記憶域パラメータの設定

次のような理由で、表を作成する前に表のサイズを見積ると有効です。

- 表の見積りサイズの合計と、索引、UNDO 領域および REDO ログ・ファイルの見積りを使用して、作成するデータベースを格納するために必要なディスク容量を決定できます。この見積りを利用して適切なハードウェアを購入できます。
- 個々の表の見積りサイズを使用することで、表が使用するディスク領域をより適切に管理できます。表を作成するときに、適切な記憶域パラメータを設定し、その表を使用するアプリケーションの I/O パフォーマンスを改善できます。たとえば、表を作成する前に表の最大サイズを見積る場合を想定します。表を作成するときに記憶域パラメータを設定すると、その表のデータ・セグメントに割り当てるエクステントを少なくできま

す。そのため、表のデータすべてが比較的ディスク領域の連続した部分に格納されます。これによって、この表に関係したディスク I/O 操作に要する時間が短くなります。

表を作成する前に表サイズを見積るかどうかにかかわらず、表を作成するときは記憶域パラメータを個々に明示的に設定できます。（クラスタ化表はクラスタの記憶域パラメータを自動的に使用します。クラスタ化表については、第 18 章「クラスタの管理」を参照してください）表を作成するとき、または表を変更するときに記憶域パラメータを明示的に設定しない場合は、その表が存在する表領域に設定されたデフォルト記憶域パラメータが自動的に使用されます。記憶域パラメータについては、14-8 ページの「記憶域パラメータの設定」を参照してください。

表のデータ・セグメントのエクステントに対して記憶域パラメータを明示的に設定する場合は、多数の小さなエクステントではなく、少数の大きなエクステントに表のデータを格納するようにします。

大規模な表の計画

表とエクステントの物理的なサイズに制限はありません。MAXEXTENTS にキーワード UNLIMITED を指定すると、大きなオブジェクトの計画を単純化し、無駄な領域や断片化を少なくして、領域の再利用率を向上させることができます。ただし、表内のエクステントの数が非常に多くなると、その表に対してなんらかの操作を実行するときにパフォーマンスに大きな影響を及ぼすことがあります。

注意： 許容されるブロックの最大値よりも MAXEXTENTS が大きくなるようにデータ・ディクショナリ表を変更することはできません。

データベース内に大規模な表がある場合は、次の推奨事項を検討してください。

- 表と索引の分離

  索引を他のオブジェクトとは別の表領域に配置し、できれば別のディスク上に配置してください。大規模な表の索引を削除して再作成する必要がある場合（制約の使用可能 / 使用禁止時や表の再作成時など）は、索引を別々の表領域に分離することによって、他のオブジェクトと同じ表領域に配置した場合よりも容易に連続領域が検出できるようになります。

- 十分な一時領域の割当て

  大規模な表のデータにアクセスするアプリケーションが大規模なソートを実行する場合は、大きな一時セグメントに使用可能な領域が十分にあることを確認してください（一時セグメントは、常にその表領域のデフォルトの STORAGE 設定を使用します）。

表の制限事項

表を作成する前に、次の制限事項について確認してください。

- オブジェクト型を含む表は、Oracle8 より古いバージョンのデータベースにインポートできません。
- オリジナルのデータがデータベースにまだ存在するときは、型とエクステント表を異なるスキーマには移動できません。
- エクスポートされた表は、異なるスキーマで同じ名前の付いた既存の表にマージできません。
- Oracle には、表が持つ列（またはオブジェクト型の属性）の合計数に制限があります。この制限については、『Oracle9*i* データベース・リファレンス』を参照してください。

  ユーザー定義型のデータを含む表を作成すると、ユーザー定義型の列はその型データを格納するリレーショナル列にマップされます。これにより、追加のリレーショナル列が作成されます。これらのリレーショナル列は「非表示」で、`DESCRIBE` 表の文では表示されず、`SELECT *` 文でも返されません。したがって、オブジェクト表、`REF` の列を持つリレーショナル表、VARRAY、ネストした表またはオブジェクト型を作成するときは、Oracle が表に対して実際に作成した列の合計数が、指定した数よりも多くなることがあるので注意してください。

  関連項目： ユーザー定義型の詳細は、『Oracle9*i* アプリケーション開発者ガイドーオブジェクト・リレーショナル機能』を参照してください。

表の作成

自分のスキーマに新しい表を作成するには、CREATE TABLE システム権限が必要です。別のユーザーのスキーマに表を作成するには、CREATE ANY TABLE システム権限が必要です。また、表の所有者には、その表を含む表領域に対する割当て制限または UNLIMITED TABLESPACE システム権限が必要です。

表は SQL 文 CREATE TABLE を使用して作成します。

この項の内容は、次のとおりです。

- 表の作成
- 一時表の作成
- 表作成のパラレル化
- 表に関する統計の自動収集

関連項目： この章で説明している CREATE TABLE などの SQL 文の正確な構文については、『Oracle9*i* SQL リファレンス』を参照してください。

表の作成

次の文を発行すると、表 admin_emp が hr スキーマに作成され、admin_tbs 表領域に格納されます。

```
CREATE TABLE        hr.admin_emp (
         empno      NUMBER(5) PRIMARY KEY,
         ename      VARCHAR2(15) NOT NULL,
         job        VARCHAR2(10),
         mgr        NUMBER(5),
         hiredate   DATE DEFAULT (sysdate),
         sal        NUMBER(7,2),
         comm       NUMBER(7,2),
         deptno     NUMBER(3) NOT NULL
                     CONSTRAINT admin_dept_fkey REFERENCES hr.departments
                     (department_id))
   TABLESPACE admin_tbs
   STORAGE ( INITIAL 50K
             NEXT 50K
             MAXEXTENTS 10
             PCTINCREASE 25 );
```

この例では、表の複数の列で整合性制約が定義されています。整合性制約については、21-14 ページの「整合性制約の管理」を参照してください。また、表には複数のセグメント属性も明示的に指定されています。これらについては、第 14 章「スキーマ・オブジェクトの領域の管理」を参照してください。

一時表の作成

一時表を作成することもできます。一時表の定義はすべてのセッションで参照できますが、一時表内のデータを参照できるのは、そのデータを表に挿入するセッションのみです。一時表を作成するには、CREATE GLOBAL TEMPORARY TABLE 文を使用します。ON COMMIT キーワードは、表のデータが**トランザクション固有**（デフォルト）であるか、**セッション固有**であるかを示します。

- ON COMMIT DELETE ROWS は、一時表がトランザクション固有であり、表は各コミット後に切り捨てられる（すべての行が削除される）ことを示します。
- ON COMMIT PRESERVE ROWS は、一時表がセッション固有であり、表はセッションの終了時に切り捨てられることを示します。

次の例では、トランザクション固有の一時表を作成しています。

```
CREATE GLOBAL TEMPORARY TABLE admin_work_area
        (startdate DATE,
         enddate DATE,
         class CHAR(20))
      ON COMMIT DELETE ROWS;
```

一時表には索引を作成できます。この索引も一時索引であり、索引内のデータのセッションまたはトランザクションの有効範囲は、基礎となる表のデータと同じです。

関連項目：

- 一時表の詳細は、『Oracle9*i* データベース概要』を参照してください。
- 一時表の他の使用例については、『Oracle9*i* アプリケーション開発者ガイドー基礎編』を参照してください。

表作成のパラレル化

表の作成時に AS SELECT 句を指定すると、パラレル実行を利用できます。CREATE TABLE ... AS SELECT 文には、CREATE 部分（DDL）と SELECT 部分（問合せ）の 2 つの部分があります。Oracle では、この文の両方の部分をパラレル化できます。次の条件が 1 つでも成り立つ場合は、CREATE 部分がパラレル化されます。

- CREATE TABLE ... AS SELECT 文に PARALLEL 句が含まれている。
- ALTER SESSION FORCE PARALLEL DDL 文が指定されている。

次の条件がすべて成り立つ場合は、問合せ部分がパラレル化されます。

- 問合せにパラレル・ヒント指定（PARALLEL または PARALLEL_INDEX）が含まれている、または CREATE 部分に PARALLEL 句が含まれている、または問合せの中で参照されるスキーマ・オブジェクトに対応付けられた PARALLEL 宣言がある。
- 問合せで指定した表のうち少なくとも 1 つで、全表スキャンまたは複数のパーティションにまたがる索引レンジ・スキャンが必要である。

表の作成をパラレル化した場合、その表には対応付けられたパラレル宣言（PARALLEL 句）が付きます。表に対するその後のすべての DML または問合せでは、パラレル化が可能な場合、パラレル実行の使用が試みられます。

表の作成をパラレル化する簡単な例を次に示します。

```
CREATE TABLE hr.admin_emp_dept
     PARALLEL
     AS SELECT * FROM hr.employees
     WHERE department_id = 10;
```

この例では、PARALLEL 句により、表の作成時に最適な数のパラレル実行サーバーを選択するよう Oracle に指示しています。

関連項目：

- パラレル実行の詳細は、『Oracle9*i* データベース概要』を参照してください。
- パラレル実行の使用に関する詳細は、『Oracle9*i* データ・ウェアハウス・ガイド』を参照してください。
- 5-18 ページ「パラレル実行用プロセスの管理」

表に関する統計の自動収集

PL/SQL パッケージ DBMS_STATS を使用すると、コストベースの最適化に関する統計を生成および管理できます。このパッケージを使用して、統計の収集、変更、表示、エクスポート、インポートおよび削除ができます。また、すでに収集した統計を識別または命名する際も、このパッケージを使用できます。

DBMS_STATS を有効化して統計を自動収集するには、CREATE（または ALTER）TABLE 文で MONITORING 句を指定します。その後、たとえば、アプリケーションにとって適切な間隔で、GATHER STALE オプションを指定した DBMS_STATS.GATHER_TABLE_STATS を起動する再帰的ジョブ（通常はジョブ・キューを使用）を準備することにより、統計収集を自動化できます。

監視では、統計が最後に収集された時点以降表に対して実行された INSERT、UPDATE および DELETE の概数が追跡されます。影響を受ける行数に関する情報は、SMON が周期的に（およそ 3 時間ごとに）データをデータ・ディクショナリに取り込むまで、システム・グローバル領域（SGA）に保持されます。このデータ・ディクショナリ情報は、DBA_TAB_MODIFICATIONS、ALL_TAB_MODIFICATIONS または USER_TAB_MODIFICATIONS を通じて参照できます。Oracle はこれらのビューを使用して、失効した統計を持つ表を識別します。

MONITORING 句と DBMS_STATS パッケージを使用することで、オプティマイザは正確な実行計画を生成できます。

表の監視を無効化するには、NOMONITORING 句を指定します。

関連項目： 統計の収集に MONITORING 句と DBMS_STATS パッケージを使用する正確なメカニズムは、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。

表の変更

表を変更するには ALTER TABLE 文を使用します。表を変更するには、その表が自分のスキーマに含まれているか、その表の ALTER オブジェクト権限または ALTER ANY TABLE システム権限のいずれかを持っている必要があります。

表を変更するには、次のように様々な理由があります。

- 物理的な特性（PCTFREE、PCTUSED、INITRANS、MAXTRANS または記憶域パラメータ）を変更する場合
- 表を新しいセグメントまたは表領域に移動する場合
- 明示的にエクステントを割り当てるか、未使用領域の割当てを解除する場合
- 列を追加、削除または名前変更する場合、あるいは既存の列の定義（データ型、長さ、デフォルト値および NOT NULL 整合性制約）を変更する場合
- 表のロギング属性を変更する場合
- CACHE/NOCACHE 属性を変更する場合
- 表に対応付けられた整合性制約を追加、変更または削除する場合
- 表に対応付けられた整合性制約またはトリガーを使用可能にするか、使用禁止にする場合
- 表の並列度を変更する場合
- 統計収集を有効化または無効化する場合（MONITORING/NOMONITORING）
- 表の名前を変更する場合
- 索引構成表の特性を追加または変更する場合
- 外部表の特性を変更する場合
- LOB 列を追加または変更する場合
- オブジェクト型、ネストした表または VARRAY の列を追加または変更する場合

ALTER TABLE 文の使用例については、次の各項を参照してください。

- 表の物理属性の変更
- 新規セグメントまたは表領域への表の移動
- 表の記憶域の手動割当て
- 既存の列の定義の変更

- 表の列の追加
- 表の列名の変更
- 表の列の削除

注意： 表を変更する前に、表を変更した結果についてよく理解しておいてください。これらの結果については、『Oracle9*i* SQL リファレンス』の ALTER TABLE 句の説明を参照してください。

パッケージのビュー、マテリアライズド・ビュー、トリガー、ドメイン索引、ファンクション・ベース索引、CHECK 制約、ファンクション、プロシージャが実表に依存する場合は、その実表または列を変更すると依存するオブジェクトに影響する可能性があります。Oracle による依存性管理の詳細は、21-23 ページの「オブジェクト依存性の管理」を参照してください。

表の物理属性の変更

表のデータ・ブロック・スペース使用パラメータ（PCTFREE と PCTUSED）を変更するときには、すでに割り当てられているブロックと今後割り当てられるブロックを含めて、その表が使用するすべてのデータ・ブロックに新しい設定が適用されます。ただし、すでに割り当てられているブロックは、スペース使用パラメータが変更されてただちに再編成されるのではなく、変更した後で必要に応じて再編成されます。データ・ブロック記憶域パラメータについては、14-2 ページの「データ・ブロックの領域管理」を参照してください。

表のトランザクション・エントリ設定（INITRANS、MAXTRANS）を変更するときには、INITRANS の新しい設定はその後表に割り当てられるデータ・ブロックにのみ適用されますが、MAXTRANS の新しい設定は表のすべてのブロック（すでに割り当てられたブロックとその後割り当てられるブロック）に適用されます。これらのトランザクション・エントリ設定パラメータの詳細は、14-7 ページの「トランザクション・エントリ・パラメータの指定：INITRANS と MAXTRANS」を参照してください。

記憶域パラメータ INITIAL と MINEXTENTS は変更できません。他の記憶域パラメータ（たとえば NEXT や PCTINCREASE）の新しい設定はすべて、その後に表に割り当てられるエクステントにのみ影響します。割り当てられる次のエクステントのサイズは、NEXT と PCTINCREASE の現行値によって決まります。前の値に基づいて決まるわけではありません。記憶域パラメータについては、14-8 ページの「記憶域パラメータの設定」を参照してください。

新規セグメントまたは表領域への表の移動

ALTER TABLE ... MOVE 文を使用すると、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。必要に応じて、割当て制限を持つ別の表領域に再配置することもできます。また、この文では、表の記憶域属性を変更することも可能です。変更可能な属性には、ALTER TABLE 文で変更できないものも含まれます。

次の文は、新しい記憶域パラメータを指定して、hr.admin_emp 表を新しいセグメントに移動します。

```
ALTER TABLE hr.admin_emp MOVE
      STORAGE ( INITIAL 20K
                NEXT 40K
                MINEXTENTS 2
                MAXEXTENTS 20
                PCTINCREASE 0 );
```

表に LOB 列が含まれている場合は、この文を使用して、ユーザーが明示的に指定できる LOB データと、表に関連した LOB 索引セグメントを、表とともに移動できます。特に指定しない場合、デフォルトでは LOB データと LOB 索引セグメントは移動されません。

表の記憶域の手動割当て

Oracle は、必要に応じて表のデータ・セグメントに追加のエクステントを動的に割り当てます。ただし、表に追加のエクステントを明示的に割り当てることもできます。たとえば、Oracle Real Application Clusters 環境で、表のエクステントを特定のインスタンスに対して明示的に割り当てることが可能です。

新しいエクステントは、ALTER TABLE ... ALLOCATE EXTENT 句を使用して表に割り当てることができます。

また、ALTER TABLE 文の DEALLOCATE UNUSED 句を使用して、未使用領域の割当てを明示的に解除することもできます。この操作については、14-24 ページの「領域の割当て解除」を参照してください。

> **関連項目：** Oracle Real Application Clusters 環境での ALLOCATE EXTENT 句の使用方法は、『Oracle9*i* Real Application Clusters 管理』を参照してください。

既存の列の定義の変更

既存の列の定義を変更するには、ALTER TABLE ... MODIFY 文を使用します。列のデータ型、デフォルト値または列制約を変更できます。

既存のデータがすべて新しい長さを満たしている場合は、既存の列の長さを拡張または縮小できます。列は、バイト・セマンティクスから CHAR セマンティクスに、あるいはその逆に変更できます。空でない CHAR 列の長さを拡張するには、初期化パラメータ BLANK_TRIMMING=TRUE を設定する必要があります。

データ型 CHAR の列長を拡張するために表を変更している場合、特に表の行数が多い場合は、この操作は時間がかかり、さらに相当な追加記憶域を必要とする可能性があります。これは、各行の CHAR 値に空白を埋めて、新しい列長に合わせる必要があるためです。

> **関連項目：** 表の列の変更とその他の制限事項の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

表の列の追加

既存の表に列を追加するには、ALTER TABLE ... ADD 文を使用します。

次の文は、hr.admin_emp 表を変更して新しい列 bonus を追加します。

```
ALTER TABLE hr.admin_emp
      ADD (bonus NUMBER (7,2));
```

表に新しい列を追加すると、DEFAULT 句を指定しないかぎり、その列は最初は NULL です。デフォルト値を指定すると、新しい列の各行が指定した値で更新されます。

NOT NULL 制約付きの列を追加できるのは、表に行がまったく含まれていない場合、またはデフォルト値を指定する場合のみです。

> **関連項目：** 表の列の追加とその他の制限事項の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

表の列名の変更

Oracle では、表の既存の列の名前を変更できます。列名を変更するには、ALTER TABLE 文の RENAME COLUMN 句を使用します。新しい名前には、表の既存の列名と競合しない名前を指定する必要があります。RENAME COLUMN 句とともに他の句は使用できません。

次の文は、hr.admin_emp 表の comm 列の名前を変更します。

```
ALTER TABLE hr.admin_emp
      RENAME COLUMN comm TO commission;
```

前述のように、表の列を変更すると、依存するオブジェクトが無効になる可能性があります。ただし、列名を変更すると、ファンクション・ベース索引と CHECK 制約が引き続き有効になるように、関連するデータ・ディクショナリ表が更新されます。

また、Oracle では列制約の名前も変更できます。この操作については、21-19 ページの「制約名の変更」を参照してください。

注意： ALTER TABLE の RENAME TO 句の構文は RENAME COLUMN 句に似ていますが、表自体の名前の変更に使用します。

表の列の削除

索引構成表などの表から、不要になった列を削除できます。これにより、データベースの領域を解放でき、データをエクスポート / インポートしてから索引と制約を再作成する必要がなくなります。

表からすべての列を削除することはできません。また、SYS が所有している表の列も削除できません。削除しようとするとエラーが発生します。

関連項目： 表からの列の削除に関するその他の制限事項およびオプションの詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

表から列を削除する方法

ALTER TABLE ... DROP COLUMN 文を発行すると、列記述子およびターゲット列に対応付けられたデータが表の各行から削除されます。1 つの文で複数の列を削除できます。次の文は、hr.admin_emp 表から列を削除する操作の例を示しています。

この文は、sal 列のみを削除します。

```
ALTER TABLE hr.admin_emp DROP COLUMN sal;
```

次の文は、bonus 列と comm 列を両方とも削除します。

```
ALTER TABLE hr.admin_emp DROP (bonus, commission);
```

列に未使用マークを付ける方法

大きい表のすべての行から列データを削除するための所要時間が重要な場合は、ALTER TABLE ... SET UNUSED 文を使用できます。この文は 1 つ以上の列に未使用マークを付けますが、実際にターゲット列を削除したり該当列が占めるディスク領域をリストアすることはありません。ただし、未使用マークが付けられた列は、問合せやデータ・ディクショナリ・ビューに表示されなくなり、その名前が削除されて新しい列に再利用できるようになります。その列に定義されている制約、索引および統計も、すべて削除されます。

hiredate 列と mgr 列に未使用マークを付けるには、次の文を実行します。

```
ALTER TABLE hr.admin_emp SET UNUSED (hiredate, mgr);
```

後で ALTER TABLE ... DROP UNUSED COLUMNS 文を発行し、未使用マークが付いている列を削除できます。表の特定列の明示的な削除文を発行すると、未使用列もターゲット表から削除されます。

データ・ディクショナリ・ビュー USER_UNUSED_COL_TABS、ALL_UNUSED_COL_TABS または DBA_UNUSED_COL_TABS を使用すると、未使用の列を含むすべての表を表示できます。COUNT フィールドには、表の未使用の列数が表示されます。

```
SELECT * FROM DBA_UNUSED_COL_TABS;

OWNER                          TABLE_NAME                   COUNT
------------------------------ ------------------------------ -----
HR                             ADMIN_EMP                        2
```

未使用列の削除

未使用列に対して実行できるのは、ALTER TABLE ... DROP UNUSED COLUMNS 文のみです。この文では、表から未使用の列が物理的に削除され、ディスク領域が再生されます。

次の例では、オプションのキーワード CHECKPOINT が指定されています。このオプションを指定すると、指定した行数（この場合は 250 行）が処理された後に、チェックポイントが適用されます。チェックポイントによって、列削除操作中に累積される UNDO ログの量が減少し、UNDO 領域が使い果たされるおそれがなくなります。

```
ALTER TABLE hr.admin_emp DROP UNUSED COLUMNS CHECKPOINT 250;
```

表のオンライン再定義

高可用性システムでは、しばしば大規模で使用頻度の高い表に対する問合せや DML のパフォーマンスを向上させるために、それらの表を再定義する必要が生じます。Oracle では、表をオンライン再定義するための仕組みが提供されています。この仕組みでは、表を再定義するためにオフライン化する必要があった従来の方法に比べて、可用性が大幅に向上します。

オンラインで表を再定義している間でも、その再定義プロセスの大部分で、DML を使用してその表にアクセスできます。表は、そのサイズや再定義の複雑さにかかわらず、わずかな間のみ、排他モードでロックされます。

この項の内容は、次のとおりです。

- 表のオンライン再定義の機能
- DBMS_REDEFINITION パッケージ
- 表のオンライン再定義の手順
- 中間での同期化
- エラー後の強制終了およびクリーン・アップ
- 表のオンライン再定義の例
- 制限事項

表のオンライン再定義の機能

表のオンライン再定義では、次のことが可能です。

- 表の記憶域パラメータの変更
- 同じスキーマ内の異なる表領域への表の移動
- パラレル問合せのサポートの追加
- パーティション化サポートの追加または削除
- 表の再作成による断片化の低減
- 通常の表（ヒープ構成表）から索引構成表へ、または索引構成表から通常の表への、編成の変更
- 列の追加または削除

DBMS_REDEFINITION パッケージ

オンライン再定義を実行するための仕組みは、PL/SQL パッケージ DBMS_REDEFINITION にあります。このパッケージには、実行権限として EXECUTE_CATALOG_ROLE が付与されています。実行ユーザーは、このパッケージの実行権限以外に、次の権限が付与されていることが必要です。

- CREATE ANY TABLE
- ALTER ANY TABLE
- DROP ANY TABLE
- LOCK ANY TABLE
- SELECT ANY TABLE

関連項目：『Oracle9*i* 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』

表のオンライン再定義の手順

表をオンラインで再定義するには、次の手順を実行する必要があります。

1. 次の 2 つの再定義方法から一方を選択します。
   - 第 1 の再定義方法は、主キーを使用して再定義を実行することです。この方法を使用することをお薦めします。この方法の場合、表の再定義前のバージョンと再定義後のバージョンの主キー列は同じになります。これはデフォルトの再定義方法です。
   - 第 2 の再定義方法は、ROWID を使用することです。この方法の場合、索引構成表は再定義できません。また、この方法では、表の再定義後のバージョンに非表示列（M_ROW$$）が追加されます。再定義の完了後に、この列を未使用としてマークするか、削除することをお薦めします。
2. DBMS_REDEFINITION.CAN_REDEF_TABLE() プロシージャを起動し、使用する再定義方法を指定して、表をオンラインで再定義できることを確認します。表がオンライン再定義の候補でない場合、このプロシージャは表をオンライン再定義できない理由を示すエラーを出力します。
3. 必要な属性をすべて持つ空の仮表を（再定義する表と同じスキーマ内に）作成します。削除される列の場合は、仮表の定義に含めないでください。列を追加する場合は、その列の定義を仮表に追加します。

   表の再定義はパラレルに実行できます。両方の表の並列度を指定し、セッションでのパラレル実行が使用可能になっていることを確認すると、可能な場合はパラレル実行を使用して再定義が実行されます。

4. 次のものを指定して DBMS_REDEFINITION.START_REDEF_TABLE() をコールし、再定義プロセスを開始します。
   - 再定義する表
   - 仮表名
   - 列マッピング
   - 再定義方法

   列マッピング情報を指定しない場合は、すべての列が（列名はそのままで）仮表に含まれるものとみなされます。列マッピング情報を指定した場合は、列マッピングで明示的に指定した列のみが対象になります。再定義方法を指定しないと、主キーを使用するデフォルトの再定義方法とみなされます。

5. 仮表のトリガー、索引、権限付与および制約をすべて作成します。仮表に関与する参照制約（つまり、仮表は参照制約の親表または子表のどちらかになります）は、すべて使用禁止で作成する必要があります。再定義プロセスが完了または強制終了するまでは、仮表で定義されたトリガーはどれも実行されません。

   再定義が完了すると、仮表に対応付けられたトリガー、制約、索引および権限付与によって、再定義中の表のトリガー、制約、索引および権限付与が置き換えられます。仮表に関与する参照制約（使用禁止で作成）は再作成中の表に移動し、再定義の完了後に使用可能になります。

6. DBMS_REDEFINITION.FINISH_REDEF_TABLE() プロシージャを実行して、表の再定義を完了します。このプロシージャの実行中、元の表はわずかな間、排他モードでロックされます。この時間の長さは、元の表のデータ量には関係ありません。また、このプロシージャの一部として次の処理が発生します。

   a. 元の表は仮表の属性、索引、制約、権限付与およびトリガーをすべて持つように再定義されます。

   b. 仮表に関与する参照制約は、再定義後の表に関与して使用可能になります。

7. 必要であれば、仮表に対して作成し、再定義された表で現在定義されている索引、トリガーおよび制約の名前を変更します。ROWID を使用して再定義した場合は、再定義後の表に非表示列（M_ROW$$）ができます。この非表示列を次のように未使用に設定することをお薦めします。

   ```
   ALTER TABLE table_name SET UNUSED (M_ROW$$)
   ```

8. 再定義プロセスの最終的な結果は、次のようになります。
   - 元の表が仮表の属性と機能で再定義されます。
   - START_REDEF_TABLE() を実行してから FINISH_REDEF_TABLE() を実行するまでの間に定義したトリガー、権限付与、索引および制約が、再定義後の表に対して定義されます。再定義プロセスを完了する前に仮表に対して作成した参照制約がすべて再定義後の表に関与し、使用可能になります。

- （再定義前に）元の表に定義されていた索引、トリガー、権限付与および制約がすべて仮表に移動し、ユーザーが仮表を削除したときに同時に削除されます。再定義する前に元の表に関与していた参照制約がすべて仮表に関与し、使用禁止になります。
- （再定義前に）元の表に定義されていた PL/SQL プロシージャおよびカーソルがすべて無効になります。これらは、次回使用されるときに必ず自動的に再検証されます（再定義プロセスの結果表の形式が変更された場合は、この再検証が失敗することがあります）。

中間での同期化

START_REDEF_TABLE() をコールして再定義プロセスを開始してから FINISH_REDEF_TABLE() コールが完了するまでの間に、元の表に対して多数の DML 文が実行される可能性があります。これが問題になることがわかっている場合は、定期的に仮表を元の表と同期化することをお薦めします。これには、DBMS_REDEFINITION.SYNC_INTERIM_TABLE() プロシージャをコールします。このプロシージャをコールすると、FINISH_REDEF_TABLE() によって再定義プロセスを完了するための時間が短縮されます。

FINISH_REDEF_TABLE() の実行中に元の表がロックされるわずかな時間は、SYNC_INTERIM_TABLE() がコールされたかどうかには関係ありません。

エラー後の強制終了およびクリーン・アップ

再定義プロセス中にエラーが発生した場合、または再定義プロセスの強制終了を選択した場合は、DBMS_REDEFINITION.ABORT_REDEF_TABLE() をコールしてください。このプロシージャは、再定義プロセスに対応付けられた一時ログおよび一時表を削除します。このプロシージャをコールした後、ユーザーは仮表とそれに対応付けられたオブジェクトを削除できます。

表のオンライン再定義の例

この例は、以前に作成した表 hr.admin_emp のオンライン再定義を示しています。この表の列は、この時点では empno、ename、job、deptno のみです。表を次のように再定義します。

- 新しい列 mgr、hiredate、sal および bonus を追加します（これらの列は元の表に存在していましたが、前述の例で削除されました）。
- 新しい列 bonus を 0 に初期化します。
- 列 deptno の値を 10 増やしています。
- 再定義された表を empno の範囲でパーティション化します。

この再定義の手順は、次のとおりです。

1. 表がオンライン再定義の候補であることを確認します。

```
BEGIN
DBMS_REDEFINITION.CAN_REDEF_TABLE('hr','admin_emp',
        dbms_redefinition.cons_use_pk);
END;
/
```

2. 仮表 hr.int_admin_emp を作成します。

```
CREATE TABLE hr.int_admin_emp
       (empno      NUMBER(5) PRIMARY KEY,
         ename      VARCHAR2(15) NOT NULL,
         job        VARCHAR2(10),
         mgr        NUMBER(5),
         hiredate   DATE DEFAULT (sysdate),
         sal        NUMBER(7,2),
         deptno     NUMBER(3) NOT NULL,
         bonus      NUMBER (7,2) DEFAULT(1000))
     PARTITION BY RANGE(empno)
       (PARTITION emp1000 VALUES LESS THAN (1000) TABLESPACE admin_tbs,
        PARTITION emp2000 VALUES LESS THAN (2000) TABLESPACE admin_tbs2);
```

3. 再定義プロセスを開始します。

```
BEGIN
DBMS_REDEFINITION.START_REDEF_TABLE('hr', 'admin_emp','int_admin_emp',
       'empno empno, ename ename, job job, deptno+10 deptno, 0 bonus',
        dbms_redefinition.cons_use_pk);
END;
/
```

4. hr.int_admin_emp に対するトリガー、索引および制約がある場合は、それらを作成します。これらは、再定義の最後の手順で元の表に戻されます。hr.int_admin_emp に関与する参照制約は、すべて使用禁止にします。仮表に対応付ける任意の権限付与を定義できます。再定義後、これらの権限付与によって元の表の権限付与が置き換えられます。

```
ALTER TABLE hr.int_admin_emp ADD CONSTRAINT admin_dept_fkey2
     FOREIGN KEY (deptno) REFERENCES hr.departments (department_id);
ALTER TABLE hr.int_admin_emp MODIFY CONSTRAINT admin_dept_fkey2
     DISABLE KEEP INDEX;
```

使用禁止にした制約 admin_dept_fkey2 は、再定義完了プロセスの一部として自動的に使用可能になり、新しく再定義された表 admin_emp に関与します。

5. 必要に応じて、仮表 hr.int_admin_emp を同期化します。

```
BEGIN
DBMS_REDEFINITION.SYNC_INTERIM_TABLE('hr', 'admin_emp', 'int_admin_emp');
END;
/
```

6. 再定義を完了します。

```
BEGIN
DBMS_REDEFINITION.FINISH_REDEF_TABLE('hr', 'admin_emp', 'int_admin_emp');
END;
/
```

この手順が終了するまでに、わずかな間のみ、表 hr.admin_emp が排他モードでロックされます。このコールの後、表 hr.admin_emp は hr.int_admin_emp 表のすべての属性を持つように再定義されます。

7. 仮表を削除します。

制限事項

表のオンライン再定義には、次の制限が適用されます。

- 表の再定義に主キーを使用する場合、再定義する表と再定義後の表の主キー列は同じにする必要があります。表の再定義に ROWID を使用する場合、索引構成表は再定義できません。
- マテリアライズド・ビューとそれらに定義されているマテリアライズド・ビュー・ログを持つ表は、オンライン再定義できません。
- マテリアライズド・ビュー・コンテナ表とアドバンスト・キューイング表は、オンライン再定義できません。
- 索引構成表のオーバーフロー表は、オンライン再定義できません。
- ユーザー定義型（オブジェクト、REF、コレクション、型付き表）を持つ表は、オンライン再定義できません。
- BFILE 列を持つ表は、オンライン再定義できません。
- LONG 列を持つ表は、オンライン再定義できません。LOB 列を持つ表は、オンライン再定義可能です。
- 再定義する表は、クラスタの一部でないことが必要です。
- SYS および SYSTEM スキーマ内の表は、オンライン再定義できません。
- 一時表は再定義できません。
- 水平サブセット化はサポートされていません。

- 仮表の列を元の表の列にマッピングするときに使用できるのは、値がすぐに決定される単純な式のみです。たとえば、副問合せは使用できません。
- 新しい列（元の表の既存データでインスタンス化されていない列）を再定義の一部として追加しようとする場合、それらの列は再定義が完了するまで NOT NULL と宣言されている必要があります。
- 再定義しようとする表と仮表の間では参照制約を作成できません。
- 表の再定義は、NOLOGGING モードでは実行できません。

表の削除

表を削除するには、その表が自分のスキーマに含まれているか、または DROP ANY TABLE システム権限を持っている必要があります。

不要になった表を削除するには、DROP TABLE 文を使用します。次の文は、hr.int_admin_emp 表を削除します。

```
DROP TABLE hr.int_admin_emp;
```

削除する表に、他の表の外部キーが参照している主キーまたは一意キーが含まれていて、その子表の FOREIGN KEY 制約を削除する場合は、次のように DROP TABLE 文に CASCADE 句を指定します。

```
DROP TABLE hr.admin_emp CASCADE CONSTRAINTS;
```

注意： 表を削除する前に、表を削除した結果についてよく理解しておいてください。

- 表を削除すると、その表定義はデータ・ディクショナリから削除されます。その結果、表のすべての行はアクセスできなくなります。
- 表に対応付けられている索引とトリガーは、すべて削除されます。
- 削除した表に依存しているビューと PL/SQL プログラム・ユニットはすべてそのまま残りますが、無効になります（使用できません）。Oracle による依存性管理の詳細は、21-23 ページの「オブジェクト依存性の管理」を参照してください。
- 削除する表のシノニムはすべてそのまま残りますが、使用するとエラーが返されます。
- 削除した表に割り当てられていたエクステントは表領域の空き領域にすべて戻され、新しいエクステントまたは新しいオブジェクトを必要とするその他のオブジェクトによって再利用されます。クラスタ化表に対応する行はすべて、そのクラスタのブロックから削除されます。クラスタ化表については、第 18 章「クラスタの管理」を参照してください。

表は、削除するかわりに切り捨てることができます。TRUNCATE 文を使用すると、表からすべての行を効率よく高速に削除できます。この操作は、切り捨てる表に対応付けられた構造（列定義、制約、トリガーなど）や認可には影響しません。TRUNCATE 文については、21-10 ページの「表とクラスタの切捨て」を参照してください。

索引構成表の管理

ここでは、索引構成表の管理について説明します。この項の内容は、次のとおりです。

- 索引構成表の概要
- 索引構成表の作成
- 索引構成表のメンテナンス
- 索引構成表の分析
- 索引構成表での ORDER BY 句の使用
- 索引構成表の標準的な表への変換

索引構成表の概要

索引構成表は、プライマリ B ツリーの異形である記憶域編成を持っています。順序付けされていないコレクション（ヒープ）としてデータを格納する通常の（ヒープ構成）表とは異なり、索引構成表のデータは B ツリーの索引構造に主キー・ソート方式で格納されます。B ツリーの各索引エントリには、索引構成表の行の主キー列値以外に、非キー列値も格納されます。

索引構成表を使用する理由

索引構成表は、完全一致検索および範囲検索を含む問合せの発行時に、キー・ベースでより高速に表データにアクセスできます。新しい行の追加、行の更新、行の削除などにより表データを変更すると、索引構造の更新のみが実行されます。これは、索引構造以外に表記憶域がないためです。

また、キー列が表と索引の両方で重複しないため、記憶域要件も少なくて済みます。残りの非キー列は、索引構造に格納されます。

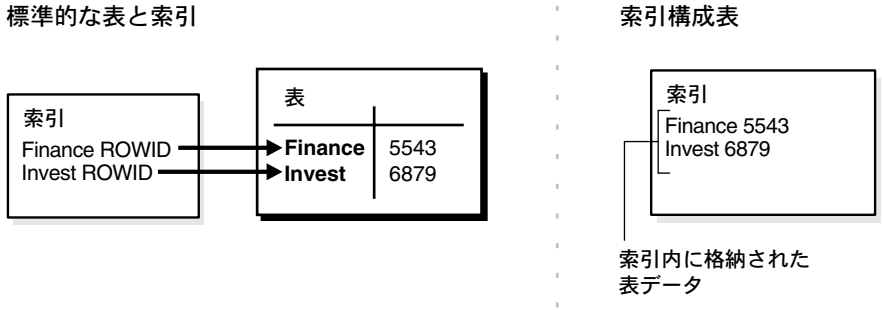
索引構成表は、特に主キーに基づいてデータを検索する必要があるアプリケーションを使用しているときに有効です。また、アプリケーション固有の索引構造をモデル化するのに適しています。たとえば、テキスト、イメージおよびオーディオ・データを含むコンテンツ・ベースの情報検索アプリケーションには、索引構成表を使用して有効にモデル化できる逆索引が必要です。

索引構成表と標準的な表との違い

図 15-1 に示すように、索引構成表は、通常の表と、1 つまたは複数の表列の索引からなる構成に類似しています。しかし、データベース・システムは、表用および B ツリー索引用の 2 つの記憶域構造を別々に維持するかわりに、単一の B ツリー索引を維持します。また、索引エントリに行の ROWID ではなく、非キー列値が格納されます。このように、B ツリー索引エントリには、それぞれ次が格納されます。

primary_key_value、*non_primary_key_column_values*

図 15-1　標準的な表と索引構成表の構造



アプリケーションは、通常の表と同様に、SQL 文を使用して索引構成表を操作します。しかし、データベース・システムは対応する B ツリー索引を操作することですべての操作を実行します。

関連項目：

- 索引構成表の詳細は、『Oracle9*i* データベース概要』を参照してください。
- 索引構成表を作成する構文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

索引構成表の作成

索引構成表を作成するには、`CREATE TABLE` 文を使用します。ただし、次の追加情報を指定する必要があります。

- `ORGANIZATION INDEX` 修飾子。これによって、索引構成表であることを示します。
- 主キー。主キーは、単一列主キーの場合は列制約句、複数列主キーの場合は表制約句によって指定します。索引構成表では必ず主キーを指定してください。
- 必要な場合は、行オーバーフロー仕様句（`OVERFLOW`）。この句は、指定したしきい値を超える行列値を別のオーバーフロー・データ・セグメントに格納することにより、B ツリー索引の稠密なクラスタを保ちます。また、`INCLUDING` 句を指定し、オーバーフロー・データ・セグメントに格納される（非キー）列を指定することもできます。
- `PCTTHRESHOLD` 値。この値は、索引構成表の索引ブロックに保持される領域の割合（百分率）を定義します。行のうち、指定したしきい値を超える部分は、オーバーフロー・セグメントに格納されます。つまり、行は、列の境界で先頭と後尾の 2 つの断片に分けられます。先頭の断片は指定したしきい値に収まり、索引のリーフ・ブロック内にキーとともに格納されます。後尾の断片は、1 つ以上の行断片としてオーバーフロー領域に格納されます。したがって、索引エントリには、キー値、指定したしきい値に収まる非キー列値および行の残りの部分へのポインタが含まれています。

次に、索引構成表の作成例を示します。

```
CREATE TABLE admin_docindex(
        token char(20),
        doc_id NUMBER,
        token_frequency NUMBER,
        token_offsets VARCHAR2(512),
        CONSTRAINT pk_admin_docindex PRIMARY KEY (token, doc_id))
    ORGANIZATION INDEX
    TABLESPACE admin_tbs
    PCTTHRESHOLD 20
    OVERFLOW TABLESPACE admin_tbs2;
```

この例では、ORGANIZATION INDEX 修飾子で索引構成表を指定しています。キー列と非キー列は、表の主キー（token、doc_id）を指定する列で定義される索引内に格納されます。

索引構成表は、オブジェクト型を格納できます。次の例は、オブジェクト型 admin_typ を作成し、オブジェクト型 admin_typ の列を含む索引構成表を作成しています。

```
CREATE OR REPLACE TYPE admin_typ AS OBJECT
    (col1 NUMBER, col2 VARCHAR2(6));
CREATE TABLE admin_iot (c1 NUMBER primary key, c2 admin_typ)
    ORGANIZATION INDEX;
```

オブジェクト型の索引構成表を作成することもできます。次に例を示します。

```
CREATE TABLE admin_iot2 OF admin_typ (col1 PRIMARY KEY)
    ORGANIZATION INDEX;
```

関連項目： パーティション化された索引構成表の作成方法は、17-19 ページの「パーティション化された索引構成表の作成」を参照してください。

AS 副問合せの使用

AS 副問合せを使用して、索引構成表を作成できます。この方法で作成する索引構成表は、PARALLEL オプションを使用してパラレルにデータをロードできます。

次の文は、従来型の表 hr.jobs から行を選択し、索引構成表を（パラレルに）作成します。

```
CREATE TABLE admin_iot3(i PRIMARY KEY, j, k, l)
     ORGANIZATION INDEX PARALLEL (DEGREE 2)
     AS SELECT * FROM hr.jobs;
```

OVERFLOW 句の使用

前の例で指定した OVERFLOW 句は、ブロック・サイズの 20% を超える行の非キー列が、admin_tbs2 表領域内のデータ・セグメントに配置されることを示しています。キー列は、指定したしきい値に収まる必要があります。

非キー列の更新によって行のサイズが小さくなる場合は、その更新が適用される行断片（先頭または後尾）が識別され、その断片が書きなおされます。

非キー列の更新によって行のサイズが大きくなる場合は、その更新が適用される行断片（先頭または後尾）が識別され、その断片が書きなおされます。更新のターゲットが先頭の断片であることがわかった場合は、指定したしきい値以下に行のサイズを保つため、この断片が再度 2 つに分割されます。

索引リーフ・ブロックに収まる非キー列は行の先頭断片として格納され、その断片にはオーバーフロー・データ・セグメントに格納された次の行断片に先頭断片をリンクする ROWID フィールドが含まれています。オーバーフロー領域に格納されるのは、収まらない列のみです。

しきい値の選択と監視 キー列とともに最初のいくつかの非キー列が頻繁にアクセスされる場合は、その非キー列を取り込めるしきい値を選択してください。

しきい値を選択した後、指定した値が適切な値であることを確認するために、表を監視できます。ANALYZE TABLE ... LIST CHAINED ROWS 文を使用して、しきい値を超える行の数と、どの行がしきい値を超えているかを判断できます。

> **関連項目：** ANALYZE 文のこの使用方法の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

INCLUDING 句の使用 PCTTHRESHOLD を指定する以外に、INCLUDING 句を使用して、キー列とともに格納する非キー列を制御できます。Oracle では、索引リーフ・ブロック内に INCLUDING 句で指定した列までのすべての非キー列を取り込むことができます。ただし、その列が指定したしきい値を超えない場合にかぎります。INCLUDING 句で指定した列より後のすべての非キー列は、オーバーフロー領域に格納されます。

注意： Oracle では、主キー・ベースのアクセス効率を高めるために、索引構成表のすべての主キー列が、表の先頭に（キー順に）移動されます。次に例を示します。

```
CREATE TABLE admin_iot4(a INT, b INT, c INT, d INT,
                primary key(c,b))
    ORGANIZATION INDEX;
```

格納後の列順は、a b c d ではなく、c b a d となります。格納された列順に基づき、最後の主キー列は b になります。INCLUDING 列には、最後の主キー列（この例では b）と非キー列（つまり、格納後の列順で b の後の任意の列）のどちらでも指定できます。

前述の例を変更し、token_offsets 列の値が常にオーバーフロー領域に格納される索引構成表を作成できます。

```
CREATE TABLE admin_docindex2(
        token CHAR(20),
        doc_id NUMBER,
        token_frequency NUMBER,
        token_offsets VARCHAR2(512),
        CONSTRAINT pk_admin_docindex2 PRIMARY KEY (token, doc_id))
    ORGANIZATION INDEX
    TABLESPACE admin_tbs
    PCTTHRESHOLD 20
    INCLUDING token_frequency
    OVERFLOW TABLESPACE admin_tbs2;
```

この例では、索引リーフ・ブロック内のキー列値とともに、token_offsets までの非キー列のみ（この場合は 1 つの列のみ）が格納されます。

キー圧縮の使用

キー圧縮を使用して索引構成表を作成すると、キー列の接頭辞が同じ値で繰り返し格納されるのを避けることができます。

キー圧縮によって、索引キーは接頭辞および接尾辞エントリに分割されます。圧縮するために、接頭辞エントリは索引ブロック内のすべての接尾辞エントリ間で共有されます。このような共有によって、領域が大幅に節約され、各索引ブロックに格納できるキー数が増え、パフォーマンスが向上します。

キー圧縮を使用可能にするには、次の操作を行う際に COMPRESS 句を使用します。

- 索引構成表の作成
- 索引構成表の移動

また、接頭辞の長さをキー列の数で指定できます。これにより、キー列が接頭辞および接尾辞エントリにどのように分割されるかが決まります。

```
CREATE TABLE admin_iot5(i INT, j INT, k INT, l INT, PRIMARY KEY (i, j, k))
    ORGANIZATION INDEX COMPRESS;
```

この文は、次の文と等価です。

```
CREATE TABLE admin_iot6(i INT, j INT, k INT, l INT, PRIMARY KEY(i, j, k))
    ORGANIZATION INDEX COMPRESS 2;
```

値リスト (1,2,3)、(1,2,4)、(1,2,7)、(1,3,5)、(1,3,4)、(1,4,4) では、(1,2)、(1,3) の反復的な発生が圧縮されます。

また、次のように、圧縮に使用されるデフォルトの接頭辞の長さを変更することもできます。

```
CREATE TABLE admin_iot7(i INT, j INT, k INT, l INT, PRIMARY KEY (i, j, k))
    ORGANIZATION INDEX COMPRESS 1;
```

値リスト (1,2,3)、(1,2,4)、(1,2,7)、(1,3,5)、(1,3,4)、(1,4,4) では、1 の反復的な発生が圧縮されます。

圧縮は、次のように使用禁止にすることができます。

```
ALTER TABLE admin_iot5 MOVE NOCOMPRESS;
```

関連項目： キー圧縮の詳細は、『Oracle9*i* データベース概要』を参照してください。

索引構成表のメンテナンス

索引構成表と標準的な表の相違点は、物理的な構成のみです。論理的には、どちらの表も同じように操作されます。INSERT 文、SELECT 文、DELETE 文および UPDATE 文では、標準的な表のかわりに索引構成表を使用できます。

索引構成表の変更

ALTER TABLE 文を使用すると、主キー索引セグメントとオーバーフロー・データ・セグメントの物理属性と記憶域属性を変更できます。OVERFLOW キーワードより前に指定したすべての属性は、主キー索引セグメントに適用できます。OVERFLOW キーワードより後に指定したすべての属性は、オーバーフロー・データ・セグメントに適用できます。たとえば、次のようにして、主キー索引セグメントの INITRANS を 4 に、オーバーフロー・データ・セグメントの INITRANS を 6 に設定できます。

```
ALTER TABLE admin_docindex INITRANS 4 OVERFLOW INITRANS 6;
```

また、PCTTHRESHOLD および INCLUDING 列の値も変更できます。後続の操作では、新しい設定を使用して、先頭部分とオーバーフローの後尾の部分に行が分割されます。たとえば、admin_docindex 表の PCTHRESHOLD および INCLUDING 列の値を次のように変更できます。

```
ALTER TABLE admin_docindex PCTTHRESHOLD 15 INCLUDING doc_id;
```

INCLUDING 列を doc_id に設定すると、その後のすべての列、つまり token_frequency および token_offsets はオーバーフロー・データ・セグメントに格納されます。

オーバーフロー・データ・セグメントなしで作成された索引構成表の場合は、ADD OVERFLOW 句を使用してオーバーフロー・データ・セグメントを追加できます。たとえば、次のように表 admin_iot3 にオーバーフロー・セグメントを追加できます。

```
ALTER TABLE admin_iot3 ADD OVERFLOW TABLESPACE admin_tbs2;
```

索引構成表の移動（再作成）

索引構成表は主として B ツリー索引に格納されるため、増分更新の結果として断片化が生じることがあります。しかし、ALTER TABLE ... MOVE 文を使用して索引を再作成することで、このような断片化を低減できます。

次の文は、索引構成表 admin_docindex を再作成します。

```
ALTER TABLE admin_docindex MOVE;
```

ONLINE キーワードを使用して、索引構成表をオンラインで再作成できます。OVERFLOW キーワードを指定すると、オーバーフロー・データ・セグメントが存在する場合はそれが再作成されます。たとえば、admin_docindex 表を再作成し、オーバーフロー・データ・セグメントを再作成しない場合は、次のようにオンラインで移動します。

```
ALTER TABLE admin_docindex MOVE ONLINE;
```

admin_docindex 表とオーバーフロー・データ・セグメントを再作成するには、次の文のようにオンラインで移動します。この文は、表とオーバーフロー・データ・セグメントを新しい表領域に移動する方法も示しています。

```
ALTER TABLE admin_docindex MOVE TABLESPACE admin_tbs2
    OVERFLOW TABLESPACE admin_tbs3;
```

次の例では、LOB 列（CLOB）を持つ索引構成表が作成されます。表が移動する間に、LOB 索引とデータ・セグメントが再作成され新しい表領域に移動します。

```
CREATE TABLE admin_iot_lob
   (c1 number (6) primary key,
    admin_lob CLOB)
   ORGANIZATION INDEX
   LOB (admin_lob) STORE AS (TABLESPACE admin_tbs2);
ALTER TABLE admin_iot_lob MOVE LOB (admin_lob) STORE AS (TABLESPACE admin_tbs3);
```

キー列の更新

キー列の更新は、論理的には、古いキー値を持つ行を削除し、主キーの順序を保つ適切な位置に新しいキー値を持つ行を挿入することと同じです。

次の例では、論理的には、admin_docindex 表から token='coins' と doc_id=10 の行が削除され、token='medals' と doc_id=10 の新しい 1 行が挿入されます。

```
UPDATE admin_docindex
    SET token='medals'
    WHERE token='coins' and doc_id=10;
```

索引構成表の分析

従来型の表と同様に、索引構成表の分析には ANALYZE 文を使用します。たとえば、次の文は admin_docindex 表の統計を収集します。

```
ANALYZE TABLE admin_docindex COMPUTE STATISTICS;
```

注意： オプティマイザ統計の収集には、DBMS_STATS パッケージを使用することをお薦めします。オプティマイザ統計の収集、ANALYZE 文を使用した非オプティマイザ統計の収集、オブジェクト構造の妥当性チェックおよび連鎖行のリストについては、21-4 ページの「表、索引およびクラスタの分析」を参照してください。

ANALYZE 文では、主キー索引セグメントとオーバーフロー・データ・セグメントの両方が分析され、表の論理統計と物理統計が算出されます。

- 論理統計は、USER_TABLES、ALL_TABLES または DBA_TABLES を使用して問合せできます。
- 主キー索引セグメントの物理統計を問い合せるには、USER_INDEXES、ALL_INDEXES または DBA_INDEXES（および主キー索引名）を使用します。たとえば、表 admin_docindex の主キー索引セグメントの物理統計は、次のようにして取得できます。

  ```
  SELECT LAST_ANALYZED, BLEVEL,LEAF_BLOCKS, DISTINCT_KEYS
     FROM DBA_INDEXES WHERE INDEX_NAME= 'PK_ADMIN_DOCINDEX';
  ```

- オーバーフロー・データ・セグメントの物理統計を問い合せるには、USER_TABLES、ALL_TABLES または DBA_TABLES を使用します。IOT_TYPE = 'IOT_OVERFLOW' で検索すると、オーバーフロー・エントリを識別できます。たとえば、admin_docindex 表に対応付けられたオーバーフロー・データ・セグメントの物理属性は、次のようにして取得できます。

  ```
  SELECT LAST_ANALYZED, NUM_ROWS, BLOCKS, EMPTY_BLOCKS
     FROM DBA_TABLES WHERE IOT_TYPE='IOT_OVERFLOW'
            and IOT_NAME= 'ADMIN_DOCINDEX';
  ```

索引構成表での ORDER BY 句の使用

ORDER BY 句が主キー列またはその接頭辞のみを参照する場合、行は主キー列でソートされた状態で返されるので、オプティマイザはソートのオーバーヘッドを回避します。

データはすでに主キーでソートされているので、次の 2 つの問合せはソートのオーバーヘッドを回避します。

```
SELECT * FROM admin_docindex2 ORDER BY token, doc_id;
SELECT * FROM admin_docindex2 ORDER BY token;
```

ただし、主キー列の接尾辞または非主キー列に ORDER BY 句がある場合は、別のソートが必要になります（他の 2 次索引が定義されていない場合）。

```
SELECT * FROM admin_docindex2 ORDER BY doc_id;
SELECT * FROM admin_docindex2 ORDER BY token_frequency
```

索引構成表の標準的な表への変換

索引構成表を標準的な表に変換するには、Oracle のインポートまたはエクスポート・ユーティリティ、あるいは CREATE TABLE ... AS SELECT 文を使用します。

索引構成表を標準的な表に変換する手順

- 従来型パスを使用して、索引構成表のデータをエクスポートします。
- 同じ定義で、標準的な表の定義を作成します。
- IGNORE=y（オブジェクト存在エラーを無視する）を指定して、索引構成表のデータをインポートします。

注意： 索引構成表を標準的な表に変換する前に、Oracle8 より古いバージョンのエクスポート・ユーティリティでは索引構成表をエクスポートできないことに注意してください。

関連項目： インポートおよびエクスポート・ユーティリティの使用方法は、『Oracle9*i* データベース・ユーティリティ』を参照してください。

外部表の管理

Oracle では、外部表内のデータへの読取り専用アクセスが可能です。外部表はデータベース内に存在しない表として定義されており、アクセス・ドライバが提供されていればどのようなフォーマットにすることもできます。外部表を記述するメタデータを提供することで、外部表内のデータをあたかも標準的なデータベース表内に存在しているデータのように公開できます。外部データは、SQL を使用して直接およびパラレルに問合せできます。

外部表のデータは、選択、結合、ソートなどが行えます。外部表のビューやシノニムも作成できます。ただし、外部表に対して DML 操作（UPDATE、INSERT または DELETE）は実行できず、索引も作成できません。

注意： 外部表の統計収集には、DBMS_STATS パッケージを使用できます。ANALYZE 文による外部表の統計収集はサポートされていません。

DBMS_STATS パッケージの詳細は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。

外部表のメタデータは、CREATE TABLE ... ORGANIZATION EXTERNAL 文を通じて定義します。外部表の定義は、外部データを最初にデータベースにロードしなくても外部データに対して任意の SQL 問合せを実行できるビューとみなすことができます。表内の外部データを読み込むために実際に使用されているメカニズムが、アクセス・ドライバです。

Oracle は、外部表のためのアクセス・ドライバを提供しています。アクセス・ドライバは、Oracle のローダー・テクノロジを使用して外部ファイルからデータを読み込むことができます。ORACLE_LOADER アクセス・ドライバは、SQL*Loader ユーティリティの制御ファイル構文のサブセットであるデータ・マッピング機能を提供します。

Oracle の外部表機能は、データ・ウェアハウスで一般的な、抽出、変換および移送（ETT）の基本タスクを実行する際に役立つ手段を提供します。

次の項では、外部表のためにサポートされているデータ定義言語（DDL）文について説明します。サポートされている DDL 文はここで説明しているもののみですが、これらの文の句がすべてサポートされているわけではありません。

- 外部表の作成
- 外部表の変更
- 外部表の削除
- 外部表のシステム権限およびオブジェクト権限

関連項目：

- 外部表、アクセス・ドライバおよびそのアクセス・パラメータの詳細は、『Oracle9*i* データベース・ユーティリティ』を参照してください。
- データ・ウェアハウス環境での外部表の使用方法は、『Oracle9*i* データ・ウェアハウス・ガイド』を参照してください。

外部表の作成

外部表は、CREATE TABLE 文の ORGANIZATION EXTERNAL 句を使用して作成します。実際には表が作成されるわけではなく、外部表にはエクステントが対応付けられません。そのかわりに、外部データへのアクセスを可能にするメタデータをデータ・ディクショナリに作成します。

次の例では、外部表を作成してから、データをデータベース表にアップロードしています。

例 : 外部表の作成とデータのロード

ファイル empxt1.dat には、次のサンプル・データが収められています。

```
360,Jane,Janus,ST_CLERK,121,17-MAY-2001,3000,0,50,jjanus
361,Mark,Jasper,SA_REP,145,17-MAY-2001,8000,.1,80,mjasper
362,Brenda,Starr,AD_ASST,200,17-MAY-2001,5500,0,10,bstarr
363,Alex,Alda,AC_MGR,145,17-MAY-2001,9000,.15,80,aalda
```

ファイル empxt2.dat には、次のサンプル・データが収められています。

```
401,Jesse,Cromwell,HR_REP,203,17-MAY-2001,7000,0,40,jcromwel
402,Abby,Applegate,IT_PROG,103,17-MAY-2001,9000,.2,60,aapplega
403,Carol,Cousins,AD_VP,100,17-MAY-2001,27000,.3,90,ccousins
404,John,Richardson,AC_ACCOUNT,205,17-MAY-2001,5000,0,110,jrichard
```

次の SQL 文は、スキーマ hr に外部表 admin_ext_employees を作成し、そのデータを hr.employees 表にロードします。

```
CONNECT  /  AS SYSDBA;
-- Set up directories and grant access to hr
CREATE OR REPLACE DIRECTORY admin_dat_dir
    AS '/net/dlsun301/private6/examples/submitted/ADMIN/flatfiles/data';
CREATE OR REPLACE DIRECTORY admin_log_dir
    AS '/net/dlsun301/private6/examples/submitted/ADMIN/flatfiles/log';
CREATE OR REPLACE DIRECTORY admin_bad_dir
    AS '/net/dlsun301/private6/examples/submitted/ADMIN/flatfiles/bad';
GRANT READ ON DIRECTORY admin_dat_dir TO hr;
GRANT WRITE ON DIRECTORY admin_log_dir TO hr;
GRANT WRITE ON DIRECTORY admin_bad_dir TO hr;
-- hr connects
CONNECT hr/hr
```

```
-- create the external table
CREATE TABLE admin_ext_employees
                   (employee_id       NUMBER(4),
                    first_name        VARCHAR2(20),
                    last_name         VARCHAR2(25),
                    job_id            VARCHAR2(10),
                    manager_id        NUMBER(4),
                    hire_date         DATE,
                    salary            NUMBER(8,2),
                    commission_pct    NUMBER(2,2),
                    department_id     NUMBER(4),
                    email             VARCHAR2(25)
                   )
     ORGANIZATION EXTERNAL
     (
       TYPE ORACLE_LOADER
       DEFAULT DIRECTORY admin_dat_dir
       ACCESS PARAMETERS
       (
         records delimited by newline
         badfile admin_bad_dir:'empxt%a_%p.bad'
         logfile admin_log_dir:'empxt%a_%p.log'
         fields terminated by ','
         missing field values are null
         ( employee_id, first_name, last_name, job_id, manager_id,
           hire_date char date_format date mask "dd-mon-yyyy",
           salary, commission_pct, department_id, email
         )
       )
       LOCATION ('empxt1.dat', 'empxt2.dat')
     )
     PARALLEL
     REJECT LIMIT UNLIMITED;
-- enable parallel for loading (good if lots of data to load)
ALTER SESSION ENABLE PARALLEL DML;
-- load the data in hr employees table
INSERT INTO employees (employee_id, first_name, last_name, job_id, manager_id,
                       hire_date, salary, commission_pct, department_id, email)
            SELECT * FROM admin_ext_employees;
```

この例について、次の各段落で説明します。

この例で、最初の数行の文は、データ・ソースを保存するオペレーティング・システム・ディレクトリ用のディレクトリ・オブジェクトと、アクセス・パラメータで指定される不良レコードやログ・ファイル用のディレクトリ・オブジェクトを作成します。また、必要に応じて READ または WRITE のディレクトリ・オブジェクト権限を付与する必要があります。

注意： ディレクトリ・オブジェクトまたは BFILE を作成する場合は、次の条件が満たされているかどうかを確認してください。

- オペレーティング・システム・ファイルが、シンボリック・リンクまたはハード・リンクでないこと。
- DIRECTORY で指定されているオペレーティング・システムのディレクトリ・パスが、既存のオペレーティング・システムのディレクトリ・パスであること。
- DIRECTORY で指定されているオペレーティング・システムのディレクトリ・パスの構成要素に、シンボリック・リンクが含まれていないこと。

TYPE 仕様は、具体的な使用方法を示す目的のみで指定されています。指定しない場合は、ORACLE_LOADER がデフォルトのアクセス・ドライバになります。ACCESS PARAMETERS 句で指定するアクセス・パラメータは、Oracle には不透明です。これらのアクセス・パラメータはアクセス・ドライバによって定義されるもので、Oracle が外部表にアクセスするときにアクセス・ドライバに提供されます。ORACLE_LOADER アクセス・パラメータの詳細は、『Oracle9*i* データベース・ユーティリティ』を参照してください。

PARALLEL 句は、データ・ソースに対するパラレル問合せを可能にします。パラレル化の最小単位はデフォルトではデータ・ソースですが、データ・ソース内部でのパラレル・アクセスは可能なかぎり実装されます。たとえば、PARALLEL=3 と指定すると、データ・ソースに対して複数のパラレル実行サーバーを稼働しておくことができます。しかし、データ・ソース内部でのパラレル・アクセスは、次の条件がすべて成り立つ場合にのみ、アクセス・ドライバによって提供されます。

- メディアが、データ・ソース内部でのランダムな位置指定をサポートしている。
- レコード境界をランダムな位置から検索できる。
- データ・ファイルが、複数のチャンクに分割することが適切なほど十分大きい。

注意： PARALLEL 句の指定は、大量のデータを扱うときのみ意味があります。データが大量でない場合には、PARALLEL 句を指定すると悪影響を及ぼす可能性が高いので、お薦めできません。

REJECT LIMIT 句は、外部データの問合せ中に発生する可能性のあるエラーの数に上限を設けないことを指定します。パラレル・アクセスの場合、この上限は各パラレル実行サーバーに個別に適用されます。たとえば、REJECT LIMIT 10 と指定すると、各パラレル問合せプロセスで 10 個の拒否が許可されます。したがって、パラレル問合せに関して正確に規定される REJECT LIMIT の値は、0（ゼロ）および UNLIMITED のみです。

この例では、INSERT INTO TABLE 文によって外部データ・ソースから Oracle SQL エンジンへのデータフローが生成され、そこでデータが処理されます。外部表ソースからのデータがアクセス・ドライバで解析されて外部表インタフェースに提供されると、外部データがその外部表現から Oracle の内部データ型に変換されます。

関連項目： 外部表の作成と句の使用に関する制限の指定を行うための CREATE TABLE 文の構文については、『Oracle9*i* SQL リファレンス』を参照してください。

外部表の変更

外部表の特性を変更するには、次のいずれかの ALTER TABLE 句を使用します。これ以外の句は使用できません。

ALTER TABLE 句	説明	例
REJECT LIMIT	拒否の上限を変更します。	ALTER TABLE admin_ext_employees REJECT LIMIT 100;
DEFAULT DIRECTORY	デフォルトのディレクトリ指定を変更します。	ALTER TABLE admin_ext_employees DEFAULT DIRECTORY admin_dat2_dir;
ACCESS PARAMETERS	外部表のメタデータの削除と再作成を行わずにアクセス・パラメータを変更できます。	ALTER TABLE admin_ext_employees ACCESS PARAMETERS (FIELDS TERMINATED BY ';');
LOCATION	外部表のメタデータの削除と再作成を行わずにデータ・ソースを変更できます。	ALTER TABLE admin_ext_employees LOCATION ('empxt3.txt', 'empxt4.txt');
PARALLEL	標準的な表の場合と同じです。並列度を変更できます。	新しい構文はありません。
ADD COLUMN	標準的な表の場合と同じです。外部表に列を追加できます。	新しい構文はありません。
MODIFY COLUMN	標準的な表の場合と同じです。外部表の列を変更できます。	新しい構文はありません。
DROP COLUMN	標準的な表の場合と同じです。外部表の列を削除できます。	新しい構文はありません。
RENAME TO	標準的な表の場合と同じです。外部表の名前を変更できます。	新しい構文はありません。

外部表の削除

外部表では、DROP TABLE 文によってデータベース内の表メタデータのみ削除されます。実際のデータはデータベースの外側に存在しているため、影響はありません。

外部表のシステム権限およびオブジェクト権限

外部表のシステム権限およびオブジェクト権限は、標準的な表のサブセットになります。外部表に適用できるシステム権限は、次のものにかぎられます。

- CREATE ANY TABLE
- ALTER ANY TABLE
- DROP ANY TABLE
- SELECT ANY TABLE

外部表に適用できるオブジェクト権限は、次のものにかぎられます

- ALTER
- SELECT

ただし、ディレクトリには次のオブジェクト権限が対応付けられています。

- READ
- WRITE

外部表では、データ・ソースのあるディレクトリ・オブジェクトに対して READ 権限が必要であり、同時に、不良ファイル、ログ・ファイルまたは廃棄ファイルのあるディレクトリ・オブジェクトに対して WRITE 権限が必要です。

表情報の表示

次のビューを使用して、表に関する情報にアクセスできます。

ビュー	説明
DBA_TABLES ALL_TABLES USER_TABLES	DBA ビューには、データベース内のすべてのリレーショナル表が表示されます。ALL ビューには、ユーザーがアクセス可能なすべての表が表示されます。USER ビューは、ユーザーが所有する表のみに制限されます。これらのビューの一部の列には、DBMS_STATS パッケージまたは ANALYZE 文によって生成される統計が含まれます。
DBA_TAB_COLUMNS ALL_TAB_COLUMNS USER_TAB_COLUMNS	これらのビューには、データベース内の表の列、ビューおよびクラスタが表示されます。これらのビューの一部の列には、DBMS_STATS パッケージまたは ANALYZE 文によって生成される統計が含まれます。
DBA_ALL_TABLES ALL_ALL_TABLES USER_ALL_TABLES	これらのビューには、データベース内のすべてのリレーショナル表およびオブジェクト表が表示されます。オブジェクト表については、このマニュアルでは詳しく説明していません。
DBA_TAB_COMMENTS ALL_TAB_COMMENTS USER_TAB_COMMENTS	これらのビューには、表およびビューのコメントが表示されます。コメントは、COMMENT 文を使用して入力します。
DBA_COL_COMMENTS ALL_COL_COMMENTS USER_COL_COMMENTS	これらのビューには、表およびビューの列のコメントが表示されます。コメントは、COMMENT 文を使用して入力します。
DBA_EXTERNAL_TABLES ALL_EXTERNAL_TABLES USER_EXTERNAL_TABLES	これらのビューには、データベースで定義されている外部表の特定の属性がリストされます。
DBA_EXTERNAL_LOCATIONS ALL_EXTERNAL_LOCATIONS USER_EXTERNAL_LOCATIONS	これらのビューには、外部表のデータ・ソースがリストされます。
DBA_TAB_HISTOGRAMS ALL_TAB_HISTOGRAMS USER_TAB_HISTOGRAMS	これらのビューには、表およびビューに関するヒストグラムが表示されます。
DBA_TAB_COL_STATISTICS ALL_TAB_COL_STATISTICS USER_TAB_COL_STATISTICS	これらのビューは、関連する TAB_COLUMNS ビューから抽出された列の統計およびヒストグラム情報を提供します。

ビュー	説明
DBA_TAB_MODIFICATIONS ALL_TAB_MODIFICATIONS USER_TAB_MODIFICATIONS	これらのビューには、表統計が最後に収集された時点以降変更された表が表示されます。これらのビューにデータが移入されるのは、MONITORING 属性を持つ表についてのみです。これらのビューは即時には移入されず、ある程度の時間（通常は 3 時間）が経過した後に移入されます。
DBA_UNUSED_COL_TABS ALL_UNUSED_COL_TABS USER_UNUSED_COL_TABS	これらのビューには、ALTER TABLE ... SET UNUSED 文によって未使用のマークが付けられた列を持つ表がリストされます。
DBA_PARTIAL_DROP_TABS ALL_PARTIAL_DROP_TABS USER_PARTIAL_DROP_TABS	これらのビューには、DROP COLUMN 操作が一部完了している表がリストされます。これらの操作は、ユーザーによる中断やシステム・クラッシュが原因で不完全になることがあります。

関連項目：

- 15-39 ページ「表情報の表示」
- これらのビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。
- オブジェクト表の詳細は、『Oracle9*i* アプリケーション開発者ガイドーオブジェクト・リレーショナル機能』を参照してください。
- ヒストグラムおよび表の統計生成の詳細は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。
- 21-4 ページ「表、索引およびクラスタの分析」

16

索引の管理

この章では、索引の管理について説明します。この章の内容は、次のとおりです。

- 索引を管理するためのガイドライン
- 索引の作成
- 索引の変更
- 索引の領域使用の監視
- 索引の削除
- 索引情報の表示

関連項目： この章のタスクを実行する前に、第 14 章「スキーマ・オブジェクトの領域の管理」を一読されることをお薦めします。

索引を管理するためのガイドライン

索引とは、表とクラスタに対応付けられるオプションの構造です。索引を使用することで、表に対して SQL 文を高速に実行できます。このマニュアルの索引によって情報を素早く検索できるのと同じように、Oracle の索引は表データへの高速なアクセス・パスを提供します。索引を使用するために、問合せを書きなおす必要はありません。索引を使用しなくても処理結果は同じですが、索引を使用するとより短時間で結果が表示されます。

Oracle は、補完的なパフォーマンス機能を持つ複数の索引付け方法を提供します。索引には、次のようなタイプがあります。

- B ツリー索引－デフォルトの設定で、最も一般的です。
- B ツリー・クラスタ索引－特にクラスタ用に定義されます。
- ハッシュ・クラスタ索引－特にハッシュ・クラスタ用に定義されます。
- グローバル索引とローカル索引－パーティション表とパーティション索引に関連します。
- 逆キー索引－ Oracle Real Application Clusters アプリケーションで最も役立ちます。
- ビットマップ索引－サイズが小さいので、小さい値の集合を持つ列に効果的です。
- ファンクション・ベース索引－事前計算された関数や式の値を含みます。
- ドメイン索引－アプリケーションまたはカートリッジに固有の索引です。

索引は、対応付けられた表内のデータから論理的にも物理的にも独立しています。索引は独立した構造体であり、記憶域を必要とします。索引は、実表、データベース・アプリケーションまたはその他の索引に影響を与えることなく、作成または削除できます。索引に対応する表に対して行の挿入、更新および削除が発生すると、Oracle は索引を自動的にメンテナンスします。索引を削除しても、すべてのアプリケーションは引き続き動作可能です。ただし、それまで索引が付けられていたデータへのアクセスが遅くなります。

ここでは、索引管理のガイドラインについて説明します。この項の内容は、次のとおりです。

- 表データ挿入後の索引の作成
- 正しい表および列への索引付け
- パフォーマンスのための索引列の順序付け
- 表当たりの索引数の制限
- 不必要な索引の削除
- 索引ブロックの領域使用の指定
- 索引サイズの見積りと記憶域パラメータの設定
- 各索引の表領域の指定

- 索引作成のパラレル化
- 索引作成時の NOLOGGING の使用
- 索引の結合と再作成に関するコストと利点の検討
- 制約を使用禁止または削除する前のコストの検討

関連項目：

- Oracle が提供する各種索引付け方法の説明など、索引と索引付けの概念に関する情報は、『Oracle9*i* データベース概要』を参照してください。
- ビットマップ索引の詳細は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』および『Oracle9*i* データ・ウェアハウス・ガイド』を参照してください。
- ドメイン固有のオペレータと索引付け方法の定義方法および Oracle データベース・サーバーへの統合方法については、『Oracle9*i* Data Cartridge Developer's Guide』を参照してください。

表データ挿入後の索引の作成

SQL*Loader またはインポート・ユーティリティを使用して、表にデータを挿入またはロードすることがあります。表の索引作成は、データを挿入またはロードした後のほうが効率的です。データをロードする前に索引を 1 つ以上作成すると、行が挿入されるたびにすべての索引の更新が必要になります。

すでにデータが格納されている表に索引を作成するには、ソート領域が必要です。索引の作成ユーザーに割り当てられているメモリーから確保されるソート領域もあります。各ユーザーのソート領域の大きさは、初期化パラメータ `SORT_AREA_SIZE` によって決まります。また、Oracle では、索引作成のときにのみユーザーの一時表領域に割り当てられる一時セグメントとの間でソート情報のスワップが行われます。

特定の条件下では、SQL*Loader のダイレクト・パス・ロードを使用してデータを表にロードし、データがロードされたときに索引が作成されることが可能です。

関連項目： SQL*Loader を使用したダイレクト・パス・ロードの詳細は、『Oracle9*i* データベース・ユーティリティ』を参照してください。

正しい表および列への索引付け

索引を作成するかどうか判断する際は、次のガイドラインを参考にしてください。

- 大きな表で頻繁に検索する行数が総行数の 15% 未満の場合に索引を作成します。この割合は、表をスキャンする相対速度と、索引キーについて行データがどのようにクラスタ化されているかによって大きく異なります。表のスキャンが速いほどこの割合は低くなります。また、行データがクラスタ化されているほど、この割合は高くなります。
- 複数の表を結合するパフォーマンスを改善するには、結合に使用する列に索引を付けます。

> **注意：** 主キーおよび一意キーには自動的に索引が作成されますが、外部キーには必要に応じて索引を作成できます。

- 小さい表に索引は不要です。問合せに時間がかかる場合は、表のサイズが大きくなっていることが考えられます。

列のタイプによっては、索引を付けるほうが望ましいものがあります。次のような特性を 1 つ以上持つ列には、索引の作成を検討してください。

- 一意の値が比較的多い。
- 値の範囲が広い（通常の索引が適している）。
- 値の範囲が狭い（ビットマップ索引が適している）。
- 列に多くの NULL が含まれているが、通常の問合せでは必ず値を持つ列を選択する。この場合は次の句を使用します。

  ```
  WHERE COL_X > -9.99 * power(10,125)
  ```

  この句は、次の句よりも優れています。

  ```
  WHERE COL_X IS NOT NULL
  ```

  これは、最初の句では COL_X の索引を使用しているためです（COL_X は数値列とします）。

次のような特性を持つ列は、索引付けには適していません。

- 列に NULL が多く含まれていて、NULL 以外の値を検索することがない。

LONG 列および LONG RAW 列には索引を作成できません。

単一の索引エントリのサイズは、データ・ブロック内の使用可能な領域のおよそ 2 分の 1（さらに多少のオーバーヘッドを差し引く必要がある）を超えることはできません。

パフォーマンスのための索引列の順序付け

CREATE INDEX 文の列の順序は、問合せのパフォーマンスに影響を与えます。一般的には、最も頻繁に使用する列を最初に指定します。

たとえば、col1、col2 および col3 の各列にアクセスする問合せを高速にするために、列にまたがる単一の索引を作成すると、col1 のみにアクセスする問合せ、または col1 と col2 にアクセスする問合せが速くなります。しかし、col2 のみにアクセスする問合せ、col3 のみにアクセスする問合せ、および col2 と col3 にアクセスする問合せは速くなりません。

表当たりの索引数の制限

表は、多数の索引を持つことができます。ただし、索引の数が多いほど、表を変更するときに発生するオーバーヘッドが増加します。特に、行を挿入したり削除したりするときは、その表の索引もすべて更新する必要があります。また、列を更新するときには、その列を含む索引もすべて更新する必要があります。

このように、表からデータを検索する速度とその表を更新する速度は二律背反的です。たとえば、表が主に読取り専用である場合、索引を増やすと有効ですが、表が頻繁に更新される場合は、索引を少なくすることをお薦めします。

不必要な索引の削除

次のような状況では、索引の削除を検討してください。

- 索引を使用しても問合せが速くならない場合。これには、たとえば表が非常に小さい場合や、表の行数は多いものの、索引エントリが非常に少ない場合などがあります。
- アプリケーションの問合せが索引を使用しない場合。
- 索引を再作成する前にいったん削除する必要がある場合。

関連項目： 16-21 ページ「索引の使用状況の監視」

索引ブロックの領域使用の指定

表の索引が作成されるとき、その索引のデータ・ブロックは、その表にある既存の値で PCTFREE まで満たされます。PCTFREE によって確保されている索引ブロックの領域は、表に新しい行が挿入され、その行に対応する索引エントリを正しい索引ブロック（前の索引エントリと次の索引エントリの間）に配置する必要があるときにのみ使用されます。

該当する索引ブロックにそれ以上領域がない場合、索引の値は（字句設定順に基づいて）別の索引ブロックに配置されます。そのため、索引を作成した表に多くの行を挿入する予定であれば、PCTFREE は新しい索引値を格納するために大きくする必要があります。また、あまりデータが挿入されず、表が比較的静的である場合は、索引データを保持するために必要なブロックが少なくなるように、対応する索引の PCTFREE を小さくします。

PCTUSED は、索引には指定できません。

関連項目： PCTFREE パラメータの詳細は、14-2 ページの「データ・ブロックの領域管理」を参照してください。

索引サイズの見積りと記憶域パラメータの設定

索引を作成する前にそのサイズを見積っておくと、ディスク領域の計画と管理がいっそう容易になります。索引の見積りサイズの合計と、表、ロールバック・セグメントおよび REDO ログ・ファイルの見積りを使用して、作成するデータベースを格納するために必要なディスク容量を決定できます。この見積りを利用して適切なハードウェアを購入できます。

個々の索引の見積りサイズを使用することで、索引が使用するディスク領域をより適切に管理できます。索引を作成するときに、適切な記憶域パラメータを設定し、その索引を使用するアプリケーションの I/O パフォーマンスを改善できます。たとえば、索引を作成する前に索引の最大サイズを見積る場合を想定します。索引を作成するときに記憶域パラメータを設定すると、その表のデータ・セグメントに割り当てるエクステントを少なくできます。そのため、表のデータすべてが比較的ディスク領域の連続した部分に格納されます。これによって、この索引に関係したディスク I/O 操作に要する時間が短くなります。

単一の索引エントリの最大サイズは、データ・ブロック・サイズのおよそ 2 分の 1 です。

関連項目： 記憶域パラメータの詳細は、14-8 ページの「記憶域パラメータの設定」を参照してください。

各索引の表領域の指定

索引はどの表領域にも作成できます。索引は、その索引を付けた表と同じ表領域にも、異なる表領域にも作成できます。表とその索引に対して同じ表領域を使用すると、（表領域やファイルのバックアップなどの）データベースのメンテナンスやアプリケーションの可用性確保の面で便利です。これは、すべての関連するデータが常にまとまってオンラインになっているためです。

表とその索引に対して（異なるディスク上にある）異なる表領域を使用すると、表と索引を同じ表領域に格納するよりも、パフォーマンスが向上します。これは、ディスクの競合が解消されるためです。表とその索引に対して異なる表領域を使用し、一方の（データまたは索引のいずれかを含む）表領域がオフラインになっている場合には、その表を参照している文が動作する保証はありません。

索引作成のパラレル化

表作成をパラレル化できるのと同様に、索引の作成もパラレル化できます。複数のプロセスが同時に動作して索引を作成するため、1 つのサーバー・プロセスが順に索引を作成する場合よりも高速に索引を作成できます。

索引を並行して作成する場合、問合せサーバー・プロセスごとに別々の記憶域パラメータが使用されます。したがって、INITIAL 値を 5MB、並行度を 12 で索引を作成する場合は、作成時に 60MB 以上の記憶域を使用します。

関連項目：

- パラレル実行の詳細は、『Oracle9*i* データベース概要』を参照してください。
- データ・ウェアハウス環境でのパラレル実行の使用方法は、『Oracle9*i* データ・ウェアハウス・ガイド』を参照してください。

索引作成時の NOLOGGING の使用

CREATE INDEX 文で NOLOGGING を指定すると、索引の作成時に最小限の REDO ログ・レコードしか生成されません。

注意： NOLOGGING を使用して作成された索引はアーカイブされないため、索引作成後にバックアップを実行してください。

NOLOGGING を使用して索引を作成すると、次のような利点があります。

- REDO ログ・ファイルの領域を節約できます。
- 索引の作成に要する時間が削減できます。
- 大規模な索引のパラレル作成のパフォーマンスが向上します。

一般に、LOGGING を指定しないで索引を作成した場合、小規模な索引より大規模な索引のほうが相対的にパフォーマンスの向上が大きくなります。小規模な索引を LOGGING を指定しないで作成しても、索引作成に要する時間にはほとんど影響しません。一方、大規模な索引では、特に索引作成のパラレル化もあわせて指定したときに、パフォーマンスが著しく向上します。

索引の結合と再作成に関するコストと利点の検討

不適切な索引サイズの設定やサイズの拡大によって、索引の断片化が生じることがあります。断片化を解消または低減するには、索引を再作成するか、索引を結合します。ただし、どちらの作業を行う場合も、事前に各選択肢のコストと利点を分析し、状況に最も有効な方法を選択してください。表 16-1 は、索引を再作成する場合と結合する場合のコストと利点を示しています。

表 16-1　再作成および結合の特徴

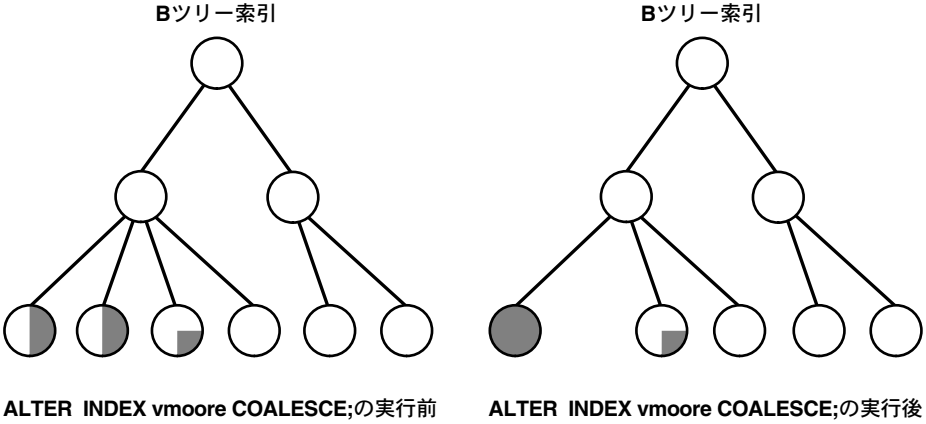
索引の再作成	索引の結合
索引を別の表領域に迅速に移動できる。	索引を別の表領域に移動することはできない。
多くのディスク領域を必要とし、コストが高い。	必要なディスク領域が少ないため、コストが低い。
新しいツリーを作成して、可能であればその高さを縮小する。	ツリーの同じブランチ内のリーフ・ブロックを結合する。
オリジナルの索引を削除せずに、記憶域パラメータと表領域パラメータを迅速に変更できる。	索引のリーフ・ブロックを迅速に解放できる。

再利用のために解放できる B ツリー索引のリーフ・ブロックがある場合は、次の文を使用してそのようなリーフ・ブロックをマージできます。

```
ALTER INDEX vmoore COALESCE;
```

図 16-1 は、索引 vmoore に対する ALTER INDEX COALESCE の効果を示しています。処理を実行する前は、最初の 2 つのリーフ・ブロックが半分満たされています。これは、断片化を低減できる余地があることを示しています。つまり、1 番目のブロックをいっぱいにして、2 番目のブロックを空にすることができます。この例では、PCTFREE=0 と想定しています。

図 16-1　索引の結合



制約を使用禁止または削除する前のコストの検討

一意キーと主キーには対応する索引があるため、一意キー制約や主キー制約を使用禁止または削除するかどうかを検討するときには、索引の削除と作成にかかわるコストを分析してください。また、一意キー制約や主キー制約に対する索引が大きい場合には、その索引を削除して再作成するよりも、その制約を使用可能な状態のままにするほうが時間を節約できます。一意キー制約や主キー制約を削除または使用禁止にするときには、索引を保持するか削除するかを明示的に指定することもできます。

関連項目： 21-14 ページ「整合性制約の管理」

索引の作成

ここでは、索引の作成方法について説明します。自分のスキーマに索引を作成するには、次の条件の少なくとも 1 つが成り立つことが必要です。

- 索引を付ける表またはクラスタが、自分のスキーマに格納されている。
- 索引を付ける表に対する INDEX 権限を持っている。
- CREATE ANY INDEX システム権限を持っている。

自分のスキーマ以外のスキーマに索引を作成するには、次の条件がすべて成り立つことが必要です。

- CREATE ANY INDEX システム権限を持っている。
- 目的のスキーマの所有者が、索引または索引パーティションを作成する表領域への割当て制限を持っているか、UNLIMITED TABLESPACE システム権限を持っている。

この項の内容は、次のとおりです。

- 索引の明示的な作成
- 一意索引の明示的な作成
- 制約に対応付けられた索引の作成
- 索引作成時の付随的統計の収集
- 大きな索引の作成
- 索引のオンラインでの作成
- ファンクション・ベース索引の作成
- キー圧縮型索引の作成

関連項目： CREATE INDEX 文、ALTER INDEX 文および DROP INDEX 文の構文と制限事項については、『Oracle9*i* SQL リファレンス』を参照してください。

索引の明示的な作成

SQL 文 CREATE INDEX を使用して、索引を明示的に（整合性制約の他に）作成できます。次の文は、emp 表の ename 列に対して emp_ename という名前の索引を作成します。

```
CREATE INDEX emp_ename ON emp(ename)
      TABLESPACE users
      STORAGE (INITIAL 20K
      NEXT 20k
      PCTINCREASE 75)
      PCTFREE 0;
```

索引に対して、複数の記憶域設定および 1 つの表領域が明示的に指定されています。索引に記憶域オプション（INITIAL や NEXT など）を指定しない場合、デフォルトの表領域または指定された表領域のデフォルトの記憶域オプションが自動的に使用されます。

一意索引の明示的な作成

索引は、一意にすることも、重複を許可することもできます。一意索引を使用すると、表の複数行のキー列に重複した値が入らないことが保証されます。非一意索引では、列の値にこのような制限はありません。

一意索引を作成するには、CREATE UNIQUE INDEX 文を使用します。次の例では、一意索引を作成しています。

```
CREATE UNIQUE INDEX dept_unique_index ON dept (dname)
      TABLESPACE indx;
```

別の方法として、目的の列に一意整合性制約を定義することもできます。Oracle では、一意のキーに対して自動的に一意索引を定義することによって、一意整合性制約を規定します。この操作については、次の項を参照してください。ただし、問合せのパフォーマンス向上のために必要な索引は、一意索引も含め、明示的に作成することをお薦めします。

> **関連項目：** パフォーマンス向上のための索引作成の詳細は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。

制約に対応付けられた索引の作成

Oracle は、表に一意キー整合性制約または主キー整合性制約を規定するために、一意キーまたは主キーの一意索引を作成します。この索引は、制約を使用可能にしたときに、Oracle によって自動的に作成されます。CREATE TABLE 文または ALTER TABLE 文を発行して索引を作成する場合は、それ以外に必要なアクションはありませんが、必要に応じて、USING INDEX 句を指定して索引作成を制御することができます。これは、制約を定義して使用可能にする場合、および定義したが使用禁止にしていた制約を使用可能にする場合のどちらでも可能です。

一意キー制約または主キー制約を使用可能にし、対応する索引を作成するには、表の所有者が索引を格納する表領域の割当て制限または UNLIMITED TABLESPACE システム権限を持っている必要があります。特に指定しないかぎり、制約の対応する索引には、常に制約と同じ名前が付けられます。

制約に対応付けられた索引に対する記憶域オプションの指定

USING INDEX 句を使用すると、一意キー制約または主キー制約に対応する索引の記憶域オプションを設定できます。次の CREATE TABLE 文は、主キー制約を使用可能にして、対応する索引の記憶域オプションを指定します。

```
CREATE TABLE emp (
     empno NUMBER(5) PRIMARY KEY, age INTEGER)
     ENABLE PRIMARY KEY USING INDEX
     TABLESPACE users
     PCTFREE 0;
```

制約に対応付けられた索引の指定

一意キー制約および主キー制約に対応付けられた索引をより明示的に制御する場合、Oracle では次のことが可能です。

- 制約を規定するために使用する既存の索引の指定
- 索引の作成と制約の規定に使用する索引の作成文の指定

これらのオプションは、USING INDEX 句を使用して指定します。次にいくつかの例を示します。

例 1:

```
CREATE TABLE a (
     a1 INT PRIMARY KEY USING INDEX (create index ai on a (a1)));
```

例 2:

```
CREATE TABLE b(
     b1 INT,
     b2 INT,
     CONSTRAINT bu1 UNIQUE (b1, b2)
                    USING INDEX (create unique index bi on b(b1, b2)),
     CONSTRAINT bu2 UNIQUE (b2, b1) USING INDEX bi);
```

例 3:

```
CREATE TABLE c(c1 INT, c2 INT);
CREATE INDEX ci ON c (c1, c2);
ALTER TABLE c ADD CONSTRAINT cpk PRIMARY KEY (c1) USING INDEX ci;
```

単一の文で制約とともに索引を作成し、同じ文でその索引を別の制約にも使用する場合、Oracle では、索引を再使用する前に索引を作成するための句の再編成を試みます。

関連項目： 21-14 ページ「整合性制約の管理」

索引作成時の付随的統計の収集

Oracle では、索引の作成または再作成の際、リソース・コストをほとんど使用せずに統計を収集できます。これらの統計はデータ・ディクショナリに格納され、SQL 文の実行計画を選択する際にオプティマイザによって常時使用されます。次の文では、表 emp の列 ename に対して索引 emp_ename を作成すると同時に、索引、表および列の各統計を計算します。

```
CREATE INDEX emp_ename ON emp(ename)
     COMPUTE STATISTICS;
```

関連項目：

- 統計の収集とオプティマイザによるその使用の詳細は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。
- 21-4 ページ「表、索引およびクラスタの分析」

大きな索引の作成

極端に大きい索引を作成するときは、次の手順を使用して、索引の作成に大きな一時表領域を割り当てることを検討してください。

1. CREATE TABLESPACE 文または CREATE TEMPORARY TABLESPACE 文を使用して、新しい一時表領域を作成します。
2. ALTER USER 文で TEMPORARY TABLESPACE オプションを指定して、この一時表領域を自分の新しい一時表領域として割り当てます。
3. CREATE INDEX 文を使用して、索引を作成します。
4. DROP TABLESPACE 文を使用して、この表領域を削除します。次に ALTER USER 文を使用して、自分の一時表領域を元の一時表領域に再設定します。

この手順により、通常使用している一時表領域（ほとんどの場合、共有されている）が極度に肥大化して今後のパフォーマンスに影響を及ぼす問題を回避できます。

索引のオンラインでの作成

索引はオンラインで作成または再作成できます。これにより、実表に索引を作成または再作成しながら、同じ実表を更新できます。索引の作成中でもデータ操作言語（DML）操作が実行できます。しかし、データ定義言語（DDL）操作は実行できません。また、索引の作成中または再作成中のパラレル実行はサポートされていません。

次の文は、オンラインでの索引作成操作を示しています。

```
CREATE INDEX emp_name ON emp (mgr, emp1, emp2, emp3) ONLINE;
```

注意： オンライン索引ビルド中に DML 操作を実行することは可能ですが、この処理の間に重要または大規模な DML 操作を実行しないことをお薦めします。これは、実表に対して DML 操作を実行すると、リソースがロックされるためです。この場合は、実表に作用しているトランザクションをコミットまたはロールバックして、リソースのロックを解放するまで、索引を作成する DDL は進行できません。

たとえば、合計で既存の表サイズの 30% を占める行をロードする場合は、オンライン索引ビルドの前にこのロードを実行する必要があります。

関連項目： 16-21 ページ「既存の索引の再作成」

ファンクション・ベース索引の作成

ファンクション・ベース索引を使用すると、関数や式から返される値を修飾する問合せが可能です。関数や式の値は、事前に計算されて索引に格納されます。

> **関連項目：** ファンクション・ベース索引に関する追加情報は、次のマニュアルを参照してください。
>
> - 『Oracle9*i* データベース概要』
> - 『Oracle9*i* データ・ウェアハウス・ガイド』

ファンクション・ベース索引の機能

ファンクション・ベース索引を使用すると、次のことが可能になります。

- より強力なソートの作成

  UPPER 関数および LOWER 関数を使用した大 / 小文字を区別しないソート、DESC キーワードによる降順ソート、NLSSORT 関数を使用した言語ベースのソートが可能です。

- 処理の負荷が大きい関数の値の事前計算と索引への格納

  アクセス頻度が高いものの、計算処理の負荷が大きい式を索引に格納できます。値が必要になったときにはすでに計算済みのため、問合せの実行パフォーマンスが大幅に向上します。

- オプティマイザが全表スキャンではなくレンジ・スキャンを実行する機会の増加

  たとえば、次のような WHERE 句の式を考えます。

  ```
  CREATE INDEX idx ON Example_tab(column_a + column_b);
  SELECT * FROM example_tab WHERE column_a + column_b < 10;
  ```

  索引が（column_a+column_b）に対して作成されているため、オプティマイザはレンジ・スキャンを実行できます。大きな表から全体の 15% 未満の行を選択する述語を使用する状況では、レンジ・スキャンは迅速な応答を示します。式がファンクション・ベース索引でマテリアライズされている場合は、その式によって選択される行数がオプティマイザによってより正確に見積られます。（ファンクション・ベース索引の式は仮想列として表され、DBMS_STATS パッケージを使用する分析操作によって仮想列のヒストグラムが作成できます。）

- 真の降順索引の使用

  この種の索引は、ファンクション・ベース索引の特殊ケースとして扱われます。

注意： DESC キーワードを使用すると、列が降順でソートされます。この場合の索引は、ファンクション・ベース索引として扱われます。降順索引はビットマップ索引または逆キー索引にすることはできず、またビットマップによる最適化には使用できません。Oracle 8 以前のリリースの DESC と同じ動作を実行するには、CREATE INDEX 文の DESC キーワードを外してください。

- オブジェクト列および REF 列に対する索引の作成

  オブジェクトを記述するメソッドを、そのオブジェクトに索引を作成する関数として使用できます。たとえば、MAP メソッドを使用してオブジェクト型の列に索引を作成できます。

  関連項目：

  - NLSSORT 関数の詳細は、『Oracle9*i* Database グローバリゼーション・サポート・ガイド』を参照してください。
  - オプティマイザの詳細は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。
  - オブジェクト列および REF 列の詳細は、『Oracle9*i* アプリケーション開発者ガイドーオブジェクト・リレーショナル機能』を参照してください。

ファンクション・ベース索引の動作

自分のスキーマにファンクション・ベース索引を作成するには、QUERY REWRITE システム権限が必要です。別のスキーマ内に索引を作成する場合、または別のスキーマにある表の索引を作成する場合は、CREATE ANY INDEX 権限および GLOBAL QUERY REWRITE 権限が必要です。

ファンクション・ベース索引を作成するには、次の初期化パラメータを定義しておく必要があります。

- QUERY_REWRITE_INTEGRITY を TRUSTED に設定します。
- QUERY_REWRITE_ENABLED を TRUE に設定します。
- COMPATIBLE を 8.1.0.0.0 以上の値に設定します。

また、ファンクション・ベース索引を使用するには、次のことを行います。

- 索引の作成後に表を分析します。
- NULL 値は索引に格納されないため、問合せには索引を作成した式からの NULL 値を必要としないことを保証します。

注意： CREATE INDEX では、ファンクション・ベース索引で最後に使用された関数のタイムスタンプが格納されます。このタイムスタンプは、索引の妥当性チェック時に更新されます。ファンクション・ベース索引について表領域の Point-in-Time リカバリを実行する場合、索引で最後に使用された関数のタイムスタンプが索引に格納されたタイムスタンプより新しい場合は、その索引には無効を示すマークが設定されます。ANALYZE INDEX ... VALIDATE STRUCTURE 文を使用して、この索引の妥当性をチェックする必要があります。

ファンクション・ベース索引の具体例として、ファンクション area(geo) に定義されているファンクション・ベース索引（area_index）を定義する次の文を示します。

```
CREATE INDEX area_index ON rivers (area(geo));
```

次の SQL 文では、area(geo) が WHERE 句で参照されているため、オプティマイザは索引 area_index の使用を考慮します。

```
SELECT id, geo, area(geo), desc
     FROM rivers
     WHERE Area(geo) >5000;
```

表の所有者は、ファンクション・ベース索引で使用されるファンクションに対して EXECUTE 権限を持つ必要があります。

ファンクション・ベース索引は使用するファンクションに依存するので、ファンクションの変更時に無効にできます。ファンクションが有効な場合は、ALTER INDEX...ENABLE 文を使用して、使用禁止になっているファンクション・ベース索引を使用可能にすることができます。ALTER INDEX...DISABLE 文では、ファンクション・ベース索引を使用禁止にすることができます。ファンクションの本体で作業をする場合は、ファンクション・ベース索引を使用禁止にするか検討してください。

ファンクション・ベース索引の例

次に、ファンクション・ベース索引の使用例を示します。

例 : 大 / 小文字を区別しない検索でのファンクション・ベース索引 次の文は、ename 列の大文字評価に基づいて、表 emp にファンクション・ベース索引 idx を作成します。

```
CREATE INDEX idx ON emp (UPPER(ename));
```

次の SELECT 文は、UPPER(ename) のファンクション・ベース索引を使用して、名前が JOH で始まるすべての従業員を取り出します。

```
SELECT * FROM emp WHERE UPPER(ename) LIKE 'JOH%';
```

これは、大 / 小文字を区別しない検索例でもあります。

例 : ファンクション・ベース索引を使用した算術式の事前計算 次の文は、式のファンクション・ベース索引を作成します。

```
CREATE INDEX idx ON t (a + b * (c - 1), a, b);
```

SELECT 文では、索引レンジ・スキャン（次の SELECT 文では、式が索引の接頭辞になっている）または索引フル・スキャン（索引で高い並列度が指定されている場合に使用するのが望ましい）を使用できます。

```
SELECT a FROM t WHERE a + b * (c - 1) < 100;
```

例 : 言語依存ソートに対するファンクション・ベース索引 ファンクション・ベース索引は、言語ソート索引のサポートに使用できます。NLSSORT は、文字列に与えられているソート・キーを返すファンクションです。したがって、NLSSORT を使用して name の索引を作成する場合は、次の文を発行します。

```
CREATE INDEX nls_index ON t_table (NLSSORT(name, 'NLS_SORT = German'));
```

この文は、照合順序 German を使用し、表 t_table の索引 nls_index を作成します。

次の文は、NLS_SORT 索引を使用して t_table から選択します。

```
SELECT * FROM t_table ORDER BY name;
```

行の順序は、German の照合順序を使用して決定されます。

次の例では、大 / 小文字を区別しないソートと言語ソートを併用しています。

```
CREATE INDEX empi ON emp
   UPPER ((ename), NLSSORT(ename));
```

ここでは、NLSSORT 引数に NLS_SORT 仕様を指定していません。こうすると、NLSSORT は言語ソート・キーの言語に関するセッションの設定を調べます。前述の例は、NLS_SORT を指定した場合を示しています。

キー圧縮型索引の作成

キー圧縮を使用して索引を作成すると、キー列の接頭辞が同じ値で繰り返し格納されるのを避けることができます。

キー圧縮によって、索引キーは接頭辞および接尾辞エントリに分割されます。圧縮するために、接頭辞エントリは索引ブロック内のすべての接尾辞エントリ間で共有されます。このような共有によって、領域が大幅に節約され、各索引ブロックに格納できるキー数が増え、パフォーマンスが向上します。

キー圧縮は、次のような状況で役立ちます。

- ROWID を追加してキーを一意にしている非一意索引がある場合。このような状況でキー圧縮を使用すると、重複キーは接頭辞エントリとして索引ブロックに ROWID なしで格納されます。残りの行は、ROWID のみからなる接尾辞エントリとなります。
- 一意の複数列索引がある場合。

キー圧縮を使用可能にするには、COMPRESS 句を使用します。また、接頭辞の長さをキー列の数で指定できます。これにより、キー列が接頭辞および接尾辞エントリにどのように分割されるかが決まります。たとえば、次の文は、索引リーフ・ブロック内のキーの重複発生を圧縮します。

```
CREATE INDEX  emp_ename ON emp(ename)
   TABLESPACE users
   COMPRESS 1;
```

索引の再作成中に COMPRESS 句を指定することもできます。たとえば、次のようにして、再作成中に圧縮を使用禁止にすることができます。

```
ALTER INDEX emp_ename REBUILD NOCOMPRESS;
```

関連項目： キー圧縮の詳細は、『Oracle9*i* データベース概要』を参照してください。

索引の変更

索引を変更するには、その索引が自分のスキーマに含まれているか、または ALTER ANY INDEX システム権限を持っている必要があります。ALTER INDEX 文で実行できるアクションには、次のようなものがあります。

- 既存の索引の再作成または結合
- 未使用領域の割当て解除または新規エクステントの割当て
- パラレル実行の指定（または指定解除）およびその並列度の変更
- 記憶域パラメータまたは物理属性の変更
- LOGGING または NOLOGGING の指定
- キー圧縮の使用可能または使用禁止の設定
- 索引への UNUSABLE マークの設定
- 索引使用状況の監視の開始または停止

索引の列構造は変更できません。

これらの操作のいくつかについて、次の項で詳しく説明します。

- 索引の記憶域特性の変更
- 既存の索引の再作成
- 索引の使用状況の監視

索引の記憶域特性の変更

主キーと一意キーの整合性制約を規定するために Oracle によって作成される索引も含めて、どの索引の記憶域パラメータも、ALTER INDEX 文を使用して変更できます。たとえば、次の文は emp_ename 索引を変更します。

```
ALTER INDEX emp_ename
     STORAGE (PCTINCREASE 50);
```

記憶域パラメータ INITIAL と MINEXTENTS は変更できません。他の記憶域パラメータの新しい設定はすべて、その後に索引に割り当てられるエクステントにのみ影響します。

整合性制約を実装する索引では、ENABLE 句の USING INDEX 副次句を指定した ALTER TABLE 文を発行することによって、記憶域パラメータを調整することもできます。たとえば、次の文は、表 emp に作成された索引の記憶域オプションを変更して、主キー制約を規定します。

```
ALTER TABLE emp
     ENABLE PRIMARY KEY USING INDEX
     PCTFREE 5;
```

既存の索引の再作成

既存の索引を再作成する前に、16-8 ページの表 16-1 の説明に従って、索引を再作成する方法と結合する方法のコストと利点を比較してください。

索引を再作成するときは、既存の索引をデータ・ソースとして使用できます。このようにして索引を作成すると、記憶特性の変更や新しい表領域への移動ができます。既存のデータ・ソースに基づいて索引を再作成すると、ブロック内の断片化も解消されます。索引を削除して CREATE INDEX 文を使用するよりも、既存の索引を再作成した方がパフォーマンスは優れています。

次の文は、既存の索引 emp_name を再作成します。

```
ALTER INDEX emp_name REBUILD;
```

REBUILD 句は、索引名の直後で他のオプションより前に指定する必要があります。この句を DEALLOCATE UNUSED 句と組み合せて使用することはできません。

索引はオンラインで再作成できます。次の文は、emp_name 索引をオンラインで再作成します。

```
ALTER INDEX emp_name REBUILD ONLINE;
```

索引の再作成に必要な大きさの領域がない場合、かわりに索引を結合する方法が使用できます。索引の結合もオンラインで可能です。

関連項目：

- 16-14 ページ「索引のオンラインでの作成」
- 16-22 ページ「索引の領域使用の監視」

索引の使用状況の監視

Oracle では、索引を監視し、それらが使用されているかを判断する手段が用意されています。未使用の索引があることがわかれば、それを削除して不要な文によるオーバーヘッドを解消できます。

索引の使用状況の監視を開始するには、次の文を発行します。

```
ALTER INDEX index MONITORING USAGE;
```

その後、監視を停止するには、次の文を発行します。

```
ALTER INDEX index NOMONITORING USAGE;
```

監視中の索引について、それが使用されているかどうか確認するには、ビュー V$OBJECT_USAGE を問い合せます。このビューには USED 列があり、監視期間中に索引が使用されたかどうかによって YES または NO の値を持ちます。また、このビューには監視の開始時間と停

止時間も記録され、MONITORING 列（YES/NO）には使用状況の監視が現在アクティブであるかどうかが示されます。

MONITORING USAGE を指定するたびに、指定した索引の V$OBJECT_USAGE ビューはリセットされます。前回の使用方法の情報は消去またはリセットされ、新しい開始時間が記録されます。NOMONITORING USAGE を指定すると、これ以上の監視は実行されず、監視期間の終了時間が記録されます。次に ALTER INDEX ... MONITORING USAGE 文が発行されるまで、ビューの情報は変更されずに保持されます。

索引の領域使用の監視

索引のキー値を頻繁に挿入、更新および削除していると、索引がその取得した領域を効果的に使用しなくなる可能性があります。そこで、最初に ANALYZE INDEX ... VALIDATE STRUCTURE 文を使用して索引の構造を分析してから、次のように INDEX_STATS ビューを問い合せることにより、定期的な間隔で索引のスペース使用の効率を監視します。

```
SELECT PCT_USED FROM INDEX_STATS WHERE NAME = 'index';
```

索引のスペース使用の割合は、どれくらい頻繁に索引キーが挿入、更新または削除されるかによって変化します。次の一連の操作を何度か実行し、索引の平均的なスペース使用効率の履歴を作成してください。

- 統計分析
- 索引の検証
- PCTUSED のチェック
- 索引の削除および再作成（または結合）

索引のスペース使用がその平均を下回っているときは、索引を削除してから再作成または結合することによって、索引の領域を圧縮できます。

関連項目： 21-4 ページ「表、索引およびクラスタの分析」

索引の削除

索引を削除するには、その索引が自分のスキーマに含まれているか、または DROP ANY INDEX システム権限を持っている必要があります。

索引を削除するのは、次のような場合です。

- 索引が不要になった場合。
- 対応する表に対して発行した問合せで、索引が予想されたパフォーマンスの改善を達成していない場合。たとえば、表が非常に小さい、または表には多くの行があるものの、索引エントリが非常に少ないなどがこれに該当します。
- アプリケーションに索引を使用するデータ問合せが含まれない場合。
- 索引が無効になり、再作成する前に削除する必要がある場合。
- 索引がかなり断片化し、再作成する前に削除する必要がある場合。

索引が削除されると、その索引のセグメントのエクステントはすべて、索引を含んでいる表領域に戻され、表領域内の他のオブジェクトで利用できます。

索引を削除する方法は、索引の作成方法、つまり CREATE INDEX 文によって索引を明示的に作成したか、または表にキー制約を定義することによって索引を暗黙的に作成したかによって異なります。CREATE INDEX 文を使用して明示的に作成した索引は、DROP INDEX 文で削除できます。次の文は、emp_ename 索引を削除します。

```
DROP INDEX emp_ename;
```

使用可能になっている一意キー制約や主キー制約に対応付けられた索引のみを削除することはできません。制約に対応付けられた索引を削除するには、制約自体を使用禁止にするかまたは削除します。

注意： 表を削除すると、対応する索引はすべて自動的に削除されます。

関連項目： 21-14 ページ「整合性制約の管理」

索引情報の表示

次のビューには、索引に関する情報が表示されます。

ビュー	説明
DBA_INDEXES ALL_INDEXES USER_INDEXES	DBA ビューには、データベース内にあるすべての表の索引が表示されます。ALL ビューには、ユーザーがアクセス可能なすべての表の索引が表示されます。USER ビューは、ユーザーが所有する索引のみに制限されます。これらのビューの一部の列には、DBMS_STATS パッケージまたは ANALYZE 文によって生成される統計が含まれます。
DBA_IND_COLUMNS ALL_IND_COLUMNS USER_IND_COLUMNS	これらのビューには、表の索引の列が表示されます。これらのビューの一部の列には、DBMS_STATS パッケージまたは ANALYZE 文によって生成される統計が含まれます。
DBA_IND_EXPRESSIONS ALL_IND_EXPRESSIONS USER_IND_EXPRESSIONS	これらのビューには、表のファンクション・ベース索引の式が表示されます。
INDEX_STATS	最後に発行された ANALYZE INDEX ... VALIDATE STRUCTURE 文の情報が格納されています。
INDEX_HISTOGRAM	最後に発行された ANALYZE INDEX ... VALIDATE STRUCTURE 文の情報が格納されています。
V$OBJECT_USAGE	ALTER INDEX ... MONITORING USAGE 機能で生成された索引使用状況の情報が含まれます。

関連項目： これらのビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

17

パーティション表と索引の管理

ここでは、パーティション表と索引の管理について説明します。この章の内容は、次のとおりです。

- パーティション表と索引の概要
- パーティション化の方法
- パーティション表の作成
- パーティション表のメンテナンス
- パーティション表および索引の例
- パーティション化された表および索引の情報の表示

関連項目： この章のタスクを実行する前に、第 14 章「スキーマ・オブジェクトの領域の管理」を一読されることをお薦めします。

パーティション表と索引の概要

今日の企業では、数百 GB のデータ（数 TB のデータになることも多い）を持つ業務上重要なデータベースが多く稼働しています。これらの企業には、大規模データベース（VLDB）のサポートおよびメンテナンスが要求されており、それらの要求を満たすような方法が必要になります。

VLDB 要求を満たす 1 つの方法は、**パーティション表および索引**を作成し、それを使用することです。パーティション表では、データを**パーティション**と呼ばれる管理が容易な単位に分割し、さらにそれを**サブパーティション**に分割できます。索引も同様にパーティション化できます。各パーティションを専用セグメントに格納し、個別に管理できます。各パーティションは互いに独立して機能します。これにより、可用性やパフォーマンスのために適切にチューニング可能な構造を使用できます。

パラレル実行を使用している場合、パーティションはパラレル化のもう 1 つの手段を提供します。パーティション表および索引に対する操作は、表や索引の様々なパーティションに異なるパラレル実行サーバーを割り当てることによって並列に実行されます。

表や索引のパーティションとサブパーティションは、すべてが同じ論理属性を共有します。たとえば、表中のすべてのパーティション（またはサブパーティション）は同じ列および制約定義を共有し、索引内のすべてのパーティション（またはサブパーティション）は同じ索引オプションを共有しています。ただし、それぞれが異なる物理属性（`TABLESPACE` など）を持つことができます。

各表や索引のパーティション（またはサブパーティション）をそれぞれ別個の表領域に格納することは必須ではありませんが、利点があります。パーティションを別個の表領域に格納すると、次のことが可能になります。

- 複数のパーティションでのデータ破損の可能性を低減する。
- 各パーティションを個別にバックアップおよびリカバリする。
- パーティションとディスク・ドライブのマッピングを制御する（I/O ロードの均衡化に重要）。
- 管理作業を軽減し、可用性とパフォーマンスを改善する。

パーティション化は既存のアプリケーションに対して透過的で、標準データ操作言語（DML）文はパーティション表に対しても動作します。また、アプリケーションでは、DML 内で拡張パーティション表名または索引名を使用して、パーティション化を利用するようにプログラミングできます。

SQL*Loader、インポート・ユーティリティおよびエクスポート・ユーティリティを使用すると、パーティション表に格納されるデータをロードまたはアンロードできます。これらのユーティリティでは、いずれもパーティションとサブパーティションが認識されます。

関連項目：

- パーティション化の詳細は、『Oracle9*i* データベース概要』を参照してください。初めてパーティション表または索引を作成したり、パーティション表のメンテナンス操作を実行する場合は、このマニュアルの情報を検討することをお薦めします。
- パラレル実行の詳細は、『Oracle9*i* データ・ウェアハウス・ガイド』および『Oracle9*i* データベース概要』を参照してください。
- SQL*Loader、インポート・ユーティリティおよびエクスポート・ユーティリティについては、『Oracle9*i* データベース・ユーティリティ』を参照してください。

パーティション化の方法

パーティション化には、次のような方法があります。

- レンジ・パーティション化
- ハッシュ・パーティション化
- リスト・パーティション化
- レンジ - ハッシュ・コンポジット・パーティション化
- レンジ - リスト・コンポジット・パーティション化

表のみでなく索引もパーティション化できます。グローバル索引はレンジ・パーティション化のみできますが、どのタイプのパーティション表または非パーティション表にも定義できます。ローカル索引より多くのメンテナンスを必要とする場合があります。

ローカル索引は、基礎となる表の構造を反映するように構成されています。ローカル索引は基礎となる表と同一レベルでパーティション化されます。つまり、基礎となる表と同じ列でパーティション化され、同数のパーティションまたはサブパーティションが作成され、基礎となる表の対応するパーティションと同じパーティション・バウンドが設定されます。ローカル索引の場合、索引パーティション化は、パーティションがメンテナンス・アクティビティの影響を受ける場合に自動的にメンテナンスされます。これにより、索引は、基礎となる表と同一レベルでパーティション化されている状態に保たれます。

次の項では、要求に適したパーティション化の方法を決定する際に役立つ情報について説明します。

- レンジ・パーティション化方法を使用する場合
- ハッシュ・パーティション化方法を使用する場合
- リスト・パーティション化方法を使用する場合

- レンジ - ハッシュ・コンポジット・パーティション化方法を使用する場合
- レンジ - リスト・コンポジット・パーティション化方法を使用する場合

レンジ・パーティション化方法を使用する場合

レンジ・パーティション化を使用すると、列値の範囲に基づいて行をパーティションにマップできます。このタイプのパーティション化が役立つのは、年度の各月など、分散できる論理範囲を持つデータを取り扱う場合です。パフォーマンスは、範囲内にデータが均等に分散しているときに最高になります。不均等に分散しているために、レンジ・パーティション化によってパーティションのサイズに大きなばらつきが生じる場合は、他のいずれかのパーティション化方法を検討する必要があります。

レンジ・パーティションを作成する場合は、次の情報を指定します。

- パーティション化の方法 : レンジ
- パーティション化列
- パーティション・バウンドを識別するパーティション記述

次の例では、四半期の売上ごとに 1 つずつ、4 つのパーティションから構成される表を作成しています。**パーティション化列**は sale_year、sale_month および sale_day の各列で、その値は特定の行の**パーティション化キー**を構成します。VALUES LESS THAN 句によって**パーティション・バウンド**が決定されます。各行のパーティション化キー値がこの句に指定された順序付きの値リストと比較され、値リストより小さい場合は、その行がパーティションに格納されます。各パーティションには名前（sales_q1、sales_q2、...）が付けられ、別個の表領域（tsa、tsb、...）に格納されます。

```
CREATE TABLE sales
    ( invoice_no NUMBER,
      sale_year  INT NOT NULL,
      sale_month INT NOT NULL,
      sale_day   INT NOT NULL )
  PARTITION BY RANGE (sale_year, sale_month, sale_day)
    ( PARTITION sales_q1 VALUES LESS THAN (1999, 04, 01)
        TABLESPACE tsa,
      PARTITION sales_q2 VALUES LESS THAN (1999, 07, 01)
        TABLESPACE tsb,
      PARTITION sales_q3 VALUES LESS THAN (1999, 10, 01)
        TABLESPACE tsc,
      PARTITION sales_q4 VALUES LESS THAN (2000, 01, 01)
        TABLESPACE tsd );
```

たとえば、sale_year=1999、sale_month=8 および sale_day=1 の行のパーティション化キーは (1999, 8, 1) となり、パーティション sales_q3 に格納されます。

ハッシュ・パーティション化方法を使用する場合

ハッシュ・パーティション化を使用するのは、データ自体のレンジ・パーティション化は困難でも、パフォーマンス上および管理上の理由からパーティション化しようとする場合です。ハッシュ・パーティション化では、データは指定した数のパーティションに均等に分散されます。行は、パーティション化キーのハッシュ値に基づいてパーティションにマップされます。ハッシュ・パーティションを作成して使用すると、これらの均等サイズのパーティションを I/O デバイス間に分散させて（ストライプ化）可用性とパフォーマンスを改善できるため、データ配置を高度にチューニングできます。

ハッシュ・パーティションを作成するには、次の情報を指定します。

- パーティション化の方法 : ハッシュ
- パーティション化列
- パーティション数または個々のパーティション記述

次の例では、ハッシュ・パーティション表を作成しています。パーティション化列は id で、4 つのパーティションが作成され、システム生成名が割り当てられて、4 つの名前付き表領域（gear1、gear2、...）に配置されます。

```
CREATE TABLE scubagear
     (id NUMBER,
      name VARCHAR2 (60))
   PARTITION BY HASH (id)
   PARTITIONS 4
   STORE IN (gear1, gear2, gear3, gear4);
```

リスト・パーティション化方法を使用する場合

リスト・パーティション化を使用するのは、行がパーティションにマップされる方法を明示的に制御する必要がある場合です。パーティション化列に含まれる離散値のリストを、各パーティションの記述で指定できます。これは、レンジ・パーティション化やハッシュ・パーティション化と異なる点です。レンジ・パーティション化では値の範囲がパーティションと関連付けられており、ハッシュ・パーティション化ではパーティションに対する行のマッピングをユーザーが制御することはできません。

リスト・パーティション化方法は、特に離散値に従ってデータ配分をモデル化するために設計されています。このモデル化は、次のような理由により、レンジ・パーティション化またはハッシュ・パーティション化では困難です。

- レンジ・パーティション化では、パーティション化列の値が自然な範囲にあることを想定しています。範囲外の値のパーティションをグループ化することはできません。
- ハッシュ・パーティション化では、システムのハッシュ関数を使用してデータが様々なパーティションに分散されるため、データの分散を制御できません。この場合も、パーティション化列の離散値を論理的にパーティションにグループ化することはできません。

リスト・パーティション化では、順序付けも関連付けもされていないデータのセットをグループ化して、自然に編成できます。

レンジ・パーティション化およびハッシュ・パーティション化と異なり、リスト・パーティション化では複数列のパーティション化はサポートされていません。表がリスト・パーティション化されている場合、パーティション化キーはその表の単一の列のみで構成されています。リスト・パーティション化されていない場合、レンジ・パーティション化またはハッシュ・パーティション化できるすべての列はリスト・パーティション化できます。

リスト・パーティションを作成する場合は、次の情報を指定します。

- パーティション化の方法 : リスト
- パーティション化列
- パーティション記述。これには、リテラル値のリスト（**値リスト**）を指定します。リテラル値とは、パーティションに含む行を識別するパーティション化列の離散値です。

次の例では、リスト・パーティション表を作成しています。ここでは、州のグループで構成されるリージョン別にパーティション化された表 `q1_sales_by_region` を作成します。

```
CREATE TABLE q1_sales_by_region
      (deptno number,
       deptname varchar2(20),
       quarterly_sales number(10, 2),
       state varchar2(2))
   PARTITION BY LIST (state)
      (PARTITION q1_northwest VALUES ('OR', 'WA'),
       PARTITION q1_southwest VALUES ('AZ', 'UT', 'NM'),
       PARTITION q1_northeast VALUES  ('NY', 'VM', 'NJ'),
       PARTITION q1_southeast VALUES ('FL', 'GA'),
       PARTITION q1_northcentral VALUES ('SD', 'WI'),
       PARTITION q1_southcentral VALUES ('OK', 'TX'));
```

パーティションを記述する値リストの値と行のパーティション化列の値が合致するかどうかをチェックすることによって、その列の値を含む行がパーティションにマップされます。

たとえば、いくつかのサンプル行が次のように挿入されます。

- (10, 'accounting', 100, 'WA') はパーティション `q1_northwest` にマップされます。
- (20, 'R&D', 150, 'OR') はパーティション `q1_northwest` にマップされます。
- (30, 'sales', 100, 'FL') はパーティション `q1_southeast` にマップされます。
- (40, 'HR', 10, 'TX') はパーティション `q1_southwest` にマップされます。
- (50, 'systems engineering', 10, 'CA') は表のどのパーティションにもマップされず、エラーが返されます。

リスト・パーティション化では、（レンジ・パーティション化と異なり）パーティション間の順序に明確な意味がないので注意してください。他のどのパーティションにもマップされない行について、マップ先となる**デフォルト・パーティション**を指定することもできます。前述の例でデフォルト・パーティションを指定すると、州 CA はそのパーティションにマップされます。

レンジ - ハッシュ・コンポジット・パーティション化方法を使用する場合

レンジ - ハッシュ・パーティション化では、データがレンジ方式でパーティション化され、各パーティション内ではハッシュ方式でサブパーティション化されます。これらのコンポジット・パーティションは、履歴データやストライプ化には理想的であり、レンジ・パーティション化とデータ配置が管理しやすくなるのみでなく、ハッシュ・パーティション化による並列性を利用できるという利点もあります。

レンジ - ハッシュ・パーティションの作成時には、次の情報を指定します。

- パーティション化の方法 : レンジ
- パーティション化列
- パーティション・バウンドを識別するパーティション記述
- サブパーティション化の方法 : ハッシュ
- サブパーティション化列
- 各パーティションのサブパーティション数またはサブパーティション記述

次の文では、レンジ - ハッシュ・パーティション表が作成されます。この例では、それぞれ 8 個のサブパーティションを含む 3 個のレンジ・パーティションが作成されます。サブパーティションには名前が指定されていないため、システム生成名が割り当てられますが、`STORE IN` 句によって指定した 4 つの表領域（`ts1`、…、`ts4`）に分散されます。

```
CREATE TABLE scubagear (equipno NUMBER, equipname VARCHAR(32), price NUMBER)
  PARTITION BY RANGE (equipno) SUBPARTITION BY HASH(equipname)
    SUBPARTITIONS 8 STORE IN (ts1, ts2, ts3, ts4)
      (PARTITION p1 VALUES LESS THAN (1000),
       PARTITION p2 VALUES LESS THAN (2000),
       PARTITION p3 VALUES LESS THAN (MAXVALUE));
```

レンジ - ハッシュ・パーティション表のパーティションは、そのデータをサブパーティションのセグメントに格納するための単なる論理構造です。パーティションの場合と同様に、これらのサブパーティションは同じ論理属性を共有します。レンジ・パーティション表内のレンジ・パーティションとは異なり、サブパーティションは所有パーティションと異なる物理属性を持つことはできません。同じ表領域に格納する必要はありません。

レンジ - リスト・コンポジット・パーティション化方法を使用する場合

レンジ - ハッシュ・コンポジット・パーティション化方法と同様に、レンジ - リスト・コンポジット・パーティション化方法では、2 レベルの階層に基づいてパーティション化できます。最初のパーティション化レベルはレンジ・パーティション化と同様に値の範囲に基づき、第 2 レベルはリスト・パーティション化と同様に離散値に基づきます。この形式のコンポジット・パーティション化は履歴データに適していますが、順序付けも関連付けもされていない列値に基づいてデータ行をさらにグループ化できます。

レンジ - リスト・パーティションの作成時には、次の情報を指定します。

- パーティション化の方法 : レンジ
- パーティション化列
- パーティション・バウンドを識別するパーティション記述
- サブパーティション化の方法 : リスト
- サブパーティション化列
- サブパーティション記述。これには、リテラル値のリスト（**値リスト**）を指定します。リテラル値とは、サブパーティションに含む行を識別するサブパーティション化列の離散値です。

次の例は、レンジ - リスト・パーティション化の使用方法を示しています。この例では、製品の売上データを四半期別に追跡し、各四半期内では指定した州別にグループ化します。

```
CREATE TABLE quarterly_regional_sales
      (deptno number, item_no varchar2(20),
       txn_date date, txn_amount number, state varchar2(2))
  TABLESPACE ts4
  PARTITION BY RANGE (txn_date)
    SUBPARTITION BY LIST (state)
      (PARTITION q1_1999 VALUES LESS THAN (TO_DATE('1-APR-1999','DD-MON-YYYY'))
         (SUBPARTITION q1_1999_northwest VALUES ('OR', 'WA'),
          SUBPARTITION q1_1999_southwest VALUES ('AZ', 'UT', 'NM'),
          SUBPARTITION q1_1999_northeast VALUES ('NY', 'VM', 'NJ'),
          SUBPARTITION q1_1999_southeast VALUES ('FL', 'GA'),
          SUBPARTITION q1_1999_northcentral VALUES ('SD', 'WI'),
          SUBPARTITION q1_1999_southcentral VALUES ('OK', 'TX')
         ),
       PARTITION q2_1999 VALUES LESS THAN ( TO_DATE('1-JUL-1999','DD-MON-YYYY'))
         (SUBPARTITION q2_1999_northwest VALUES ('OR', 'WA'),
          SUBPARTITION q2_1999_southwest VALUES ('AZ', 'UT', 'NM'),
          SUBPARTITION q2_1999_northeast VALUES ('NY', 'VM', 'NJ'),
          SUBPARTITION q2_1999_southeast VALUES ('FL', 'GA'),
          SUBPARTITION q2_1999_northcentral VALUES ('SD', 'WI'),
          SUBPARTITION q2_1999_southcentral VALUES ('OK', 'TX')
         ),
```

```
    PARTITION q3_1999 VALUES LESS THAN (TO_DATE('1-OCT-1999','DD-MON-YYYY'))
      (SUBPARTITION q3_1999_northwest VALUES ('OR', 'WA'),
       SUBPARTITION q3_1999_southwest VALUES ('AZ', 'UT', 'NM'),
       SUBPARTITION q3_1999_northeast VALUES ('NY', 'VM', 'NJ'),
       SUBPARTITION q3_1999_southeast VALUES ('FL', 'GA'),
       SUBPARTITION q3_1999_northcentral VALUES ('SD', 'WI'),
       SUBPARTITION q3_1999_southcentral VALUES ('OK', 'TX')
      ),
    PARTITION q4_1999 VALUES LESS THAN ( TO_DATE('1-JAN-2000','DD-MON-YYYY'))
      (SUBPARTITION q4_1999_northwest VALUES ('OR', 'WA'),
       SUBPARTITION q4_1999_southwest VALUES ('AZ', 'UT', 'NM'),
       SUBPARTITION q4_1999_northeast VALUES ('NY', 'VM', 'NJ'),
       SUBPARTITION q4_1999_southeast VALUES ('FL', 'GA'),
       SUBPARTITION q4_1999_northcentral VALUES ('SD', 'WI'),
       SUBPARTITION q4_1999_southcentral VALUES ('OK', 'TX')
      )
  );
```

行のパーティション化列の値が特定のパーティションの範囲に合致するかどうかをチェックすることによって、その列の値を含む行がパーティションにマップされます。行は、記述子の値リストにサブパーティション列の値と一致する値が含まれているサブパーティションを識別することによって、そのパーティション内のサブパーティションにマップされます。

たとえば、いくつかのサンプル行が次のように挿入されます。

- (10, 4532130, '23-Jan-1999', 8934.10, 'WA') はサブパーティション `q1_1999_northwest` にマップされます。
- (20, 5671621, '15-May-1999', 49021.21, 'OR') はサブパーティション `q2_1999_northeast` にマップされます。
- (30, 9977612, ,'07-Sep-1999', 30987.90, 'FL') はサブパーティション `q3_1999_southeast` にマップされます。
- (40, 9977612, '29-Nov-1999', 67891.45, 'TX') はサブパーティション `q4_1999_southwest` にマップされます。
- (40, 4532130, '5-Jan-2000', 897231.55, 'TX') は表のどのパーティションにもマップされず、エラーになります。
- (50, 5671621, '17-Dec-1999', 76123.35, 'CA') は表のどのサブパーティションにもマップされず、エラーになります。

レンジ - リスト・パーティション表のパーティションは、そのデータをサブパーティションのセグメントに格納するための単なる論理構造です。リスト・サブパーティションの特性は、リスト・パーティションと同じです。リスト・パーティション化についてデフォルト・パーティションを指定するのと同様に、デフォルト・サブパーティションを指定できます。

パーティション表の作成

パーティション表または索引を作成する手順は、（第 15 章「表の管理」で説明されている）標準的な表や索引の場合とほとんど同じですが、パーティション化句を使用する点が異なります。指定するパーティション句と副次句は、パーティション化のタイプによって異なります。

LOB 列を持つ表も含め、通常の（ヒープ構成）表および索引構成表はどちらもパーティション化できます。パーティション表には、非パーティション・グローバル索引、レンジ・パーティション・グローバル索引およびローカル索引を作成できます。

パーティション表を作成または変更する場合は、行移動句 ENABLE ROW MOVEMENT または DISABLE ROW MOVEMENT を指定できます。この句は、対応するキーが更新された場合に、新規パーティションへの行の移行を使用可能または使用禁止にします。デフォルトの行移動句は、DISABLE ROW MOVEMENT です。

次の項では、各種のパーティション表およびパーティション索引用にパーティションを作成する方法の詳細と例を示します。

- レンジ・パーティション表の作成
- ハッシュ・パーティション表の作成
- リスト・パーティション表の作成
- レンジ - ハッシュ・コンポジット・パーティション表の作成
- レンジ - リスト・コンポジット・パーティション表の作成
- サブパーティション・テンプレートを使用したコンポジット・パーティション表の記述
- パーティション化された索引構成表の作成
- 複数のブロック・サイズに関するパーティション化の制限

関連項目：

- パーティション表および索引の作成と変更に使用するパーティション化句の正確な構文、使用制限、表の作成と変更に必要な権限については、『Oracle9*i* SQL リファレンス』を参照してください。
- LOB を持つ列や LOB として格納されているその他のオブジェクトを含むパーティション表の作成方法は、『Oracle9*i* アプリケーション開発者ガイドーラージ・オブジェクト』および『Oracle9*i* アプリケーション開発者ガイドー基礎編』を参照してください。

レンジ・パーティション表の作成

表をレンジ・パーティション化するには、CREATE TABLE 文で PARTITION BY RANGE 句を指定します。PARTITION 句では個々のパーティション・レンジを識別し、PARTITION 句のオプションの副次句では、パーティションのセグメントに固有の物理属性と他の属性を識別できます。パーティション・レベルで上書きされない場合、各パーティションはその基礎となる表の属性を継承します。

この例は、前述のレンジ・パーティション表の例をさらに複雑にしたものです。記憶域パラメータと LOGGING 属性が表レベルで指定されています。これらの指定により、表自体に対して表領域レベルから継承された対応するデフォルトが置換され、その値がレンジ・パーティションによって継承されます。ただし、第 1 四半期には取引が少なかったため、パーティション sales_q1 の記憶域属性は小さくなっています。ENABLE ROW MOVEMENT 句が指定されているため、行が別のパーティションに格納されるようなキー値の更新が発生した場合は、その行を新規パーティションに移行できます。

```
CREATE TABLE sales
     ( invoice_no NUMBER,
       sale_year  INT NOT NULL,
       sale_month INT NOT NULL,
       sale_day   INT NOT NULL )
   STORAGE (INITIAL 100K NEXT 50K) LOGGING
   PARTITION BY RANGE ( sale_year, sale_month, sale_day)
     ( PARTITION sales_q1 VALUES LESS THAN ( 1999, 04, 01 )
        TABLESPACE tsa STORAGE (INITIAL 20K, NEXT 10K),
       PARTITION sales_q2 VALUES LESS THAN ( 1999, 07, 01 )
        TABLESPACE tsb,
       PARTITION sales_q3 VALUES LESS THAN ( 1999, 10, 01 )
        TABLESPACE tsc,
       PARTITION sales q4 VALUES LESS THAN ( 2000, 01, 01 )
        TABLESPACE tsd)
   ENABLE ROW MOVEMENT;
```

レンジ・パーティション化されたグローバル索引の作成ルールは、レンジ・パーティション表の場合に似ています。次に、前述の表の sales_month に対して、レンジ・パーティション化されたグローバル索引を作成する例を示します。各索引パーティションには名前が指定されていますが、索引用のデフォルト表領域に格納されます。

```
CREATE INDEX month_ix ON sales(sales_month)
   GLOBAL PARTITION BY RANGE(sales_month)
      (PARTITION pm1_ix VALUES LESS THAN (2)
       PARTITION pm2_ix VALUES LESS THAN (3)
       PARTITION pm3_ix VALUES LESS THAN (4)
       PARTITION pm4_ix VALUES LESS THAN (5)
       PARTITION pm5_ix VALUES LESS THAN (6)
       PARTITION pm6_ix VALUES LESS THAN (7)
       PARTITION pm7_ix VALUES LESS THAN (8)
       PARTITION pm8_ix VALUES LESS THAN (9)
```

```
PARTITION pm9_ix VALUES LESS THAN (10)
PARTITION pm10_ix VALUES LESS THAN (11)
PARTITION pm11_ix VALUES LESS THAN (12)
PARTITION pm12_ix VALUES LESS THAN (MAXVALUE));
```

注意： 文字のソート順はキャラクタ・セットごとに異なるため、異なるキャラクタ・セットを使用しているデータベースの場合、キャラクタ列をパーティション化するときには注意してください。詳細は、『Oracle9*i* Database グローバリゼーション・サポート・ガイド』を参照してください。

ハッシュ・パーティション表の作成

ハッシュ・パーティション化する表を識別するには、CREATE TABLE 文で PARTITION BY HASH 句を指定します。PARTITIONS 句を使用すると、作成するパーティション数、および必要に応じてそれを格納する表領域を指定できます。また、PARTITION 句を使用して、個々のパーティションとその表領域に名前を付けることもできます。

ハッシュ・パーティションに指定できる属性は、TABLESPACE のみです。表のすべてのハッシュ・パーティションは、表レベルから継承される同じセグメント属性（TABLESPACE 以外）を共有します。

次の例は、ハッシュ・パーティション表 dept について、2 つの作成方法を示しています。最初の例ではパーティション数を指定していますが、システム生成名が割り当てられ、表のデフォルト表領域に格納されます。

```
CREATE TABLE dept (deptno NUMBER, deptname VARCHAR(32))
     PARTITION BY HASH(deptno) PARTITIONS 16;
```

第 2 の例では、個々のパーティション名と、それぞれが格納される表領域が指定されています。各ハッシュ・パーティション（セグメント）の初期エクステント・サイズも、表レベルで明示的に指定されており、すべてのパーティションはこの属性を継承します。

```
CREATE TABLE dept (deptno NUMBER, deptname VARCHAR(32))
     STORAGE (INITIAL 10K)
     PARTITION BY HASH(deptno)
       (PARTITION p1 TABLESPACE ts1, PARTITION p2 TABLESPACE ts2,
        PARTITION p3 TABLESPACE ts1, PARTITION p4 TABLESPACE ts3);
```

この表にローカル索引を作成すると、その索引は基礎となる表と同一レベルでパーティション化されます。また、基礎となる表のメンテナンス操作が実行されると索引が自動的にメンテナンスされるように構成されます。次に、表 dept 上でローカル索引を作成する例を示します。

```
CREATE INDEX loc_dept_ix ON dept(deptno) LOCAL;
```

必要に応じて、ハッシュ・パーティションと、ローカル索引パーティションが格納される表領域の名前を指定できます。指定しない場合は、対応するベース・パーティションの名前が索引パーティション名として使用され、索引パーティションは表パーティションと同じ表領域に格納されます。

リスト・パーティション表の作成

リスト・パーティションを作成するためのセマンティクスは、レンジ・パーティションを作成するためのセマンティクスとほぼ同じです。ただし、リスト・パーティションを作成するには、CREATE TABLE 文で PARTITION BY LIST 句を指定し、PARTITION 句にはリテラル値のリストを指定します。リテラル値とは、パーティションに含む行を識別するパーティション化列の離散値です。リスト・パーティション化の場合、パーティション化キーにはその表の単一の列の名前しか使用できません。

リスト・パーティション化の場合にのみ、キーワード DEFAULT を使用してパーティションの値リストを記述できます。これにより、他のどのパーティションにもマップされない行を格納するパーティションが識別されます。

レンジ・パーティションの場合と同様に、PARTITION 句のオプションの副次句では、パーティションのセグメントに固有の物理属性と他の属性を識別できます。パーティション・レベルで上書きされない場合、各パーティションはその基礎となる表の属性を継承します。

次の例では、表 sales_by_region を作成し、リスト方法を使用してその表をパーティション化しています。最初の 2 つの PARTITION 句には物理属性を指定し、表レベルのデフォルトの属性を変更しています。他の PARTITION 句には属性を指定していないため、これらのパーティションの物理属性は表レベルのデフォルト属性から継承されます。デフォルト・パーティションは指定されています。

```
CREATE TABLE sales_by_region (item# INTEGER, qty INTEGER,
             store_name VARCHAR(30), state_code VARCHAR(2),
             sale_date DATE)
     STORAGE(INITIAL 10K NEXT 20K) TABLESPACE tbs5
     PARTITION BY LIST (state_code)
     (
     PARTITION region_east
        VALUES ('MA','NY','CT','NH','ME','MD','VA','PA','NJ')
        STORAGE (INITIAL 20K NEXT 40K PCTINCREASE 50)
        TABLESPACE tbs8,
     PARTITION region_west
        VALUES ('CA','AZ','NM','OR','WA','UT','NV','CO')
        PCTFREE 25 NOLOGGING,
     PARTITION region_south
        VALUES ('TX','KY','TN','LA','MS','AR','AL','GA'),
     PARTITION region_central
        VALUES ('OH','ND','SD','MO','IL','MI','IA'),
     PARTITION region_null
        VALUES (NULL),
```

```
     PARTITION region_unknown
        VALUES (DEFAULT)
     );
```

レンジ - ハッシュ・コンポジット・パーティション表の作成

レンジ - ハッシュ・パーティション表を作成するには、最初に CREATE TABLE 文の PARTITION BY RANGE 句を使用します。次に、PARTITION BY HASH 句と同じ構文およびルールに従って、SUBPARTITION BY HASH 句を指定します。その後に、個々の PARTITION 句および SUBPARTITION または SUBPARTITIONS 句と、オプションの SUBPARTITION TEMPLATE 句を続けて指定します。

レンジ・パーティションに指定した属性は、そのパーティションのすべてのサブパーティションに適用されます。レンジ・パーティションごとに異なる属性を指定できます。また、そのパーティションのサブパーティションが分散される表領域のリストが他のパーティションと異なる場合は、STORE IN 句をパーティション・レベルで指定できます。次の例に、このすべての操作を示します。

```
CREATE TABLE emp (deptno NUMBER, empname VARCHAR(32), grade NUMBER)
     PARTITION BY RANGE(deptno) SUBPARTITION BY HASH(empname)
        SUBPARTITIONS 8 STORE IN (ts1, ts3, ts5, ts7)
    (PARTITION p1 VALUES LESS THAN (1000) PCTFREE 40,
     PARTITION p2 VALUES LESS THAN (2000)
        STORE IN (ts2, ts4, ts6, ts8),
     PARTITION p3 VALUES LESS THAN (MAXVALUE)
       (SUBPARTITION p3_s1 TABLESPACE ts4,
        SUBPARTITION p3_s2 TABLESPACE ts5));
```

サブパーティション・テンプレートを使用してコンポジット・パーティション表の指定を簡素化する方法は、17-16 ページの「サブパーティション・テンプレートを使用したコンポジット・パーティション表の記述」を参照してください。

次の文は、emp 表に対して、索引セグメントが表領域 ts7、ts8 および ts9 に分散するローカル索引の作成例を示しています。

```
CREATE INDEX emp_ix ON emp(deptno)
     LOCAL STORE IN (ts7, ts8, ts9);
```

このローカル索引は、次のように実表と同一レベルでパーティション化されます。

- 実表と同数のパーティションから構成されます。
- 各索引パーティションは、対応する実表のパーティションと同数のサブパーティションから構成されます。
- 実表の特定サブパーティションの行に関する索引エントリは、索引の対応するサブパーティションに格納されます。

レンジ - リスト・コンポジット・パーティション表の作成

レンジ - リスト・パーティション化の概念は、もう 1 つのコンポジット・パーティション化方法であるレンジ - ハッシュ方法に似ていますが、この場合はサブパーティションをハッシュするのではなくリストするように指定します。特に、CREATE TABLE ... PARTITION BY RANGE 句に続けて、PARTITION BY LIST 句と同様の構文およびルールに従って SUBPARTITION BY LIST 句を指定します。その後に、個々の PARTITION 句および SUBPARTITION 句と、オプションの SUBPARTITION TEMPLATE 句を続けて指定します。

コンポジット・パーティション表のレンジ・パーティションは、非コンポジット・レンジ・パーティション表と同様に記述されます。このため、PARTITION 句のオプションの副次句では、表領域など、パーティションのセグメントに固有の物理属性と他の属性を識別できます。パーティション・レベルで上書きされない場合、各パーティションはその基礎となる表の属性を継承します。

リスト・サブパーティション記述は、SUBPARTITION 句に非コンポジット・リスト・パーティションの場合と同様に記述しますが、指定できる物理属性は表領域（オプション）のみです。サブパーティションは、他のすべての物理属性をパーティション記述から継承します。

次の例は、パーティション・レベルとサブパーティション・レベルで表領域を指定する表の作成方法を示しています。各パーティション内のサブパーティション数は異なり、デフォルト・サブパーティションを指定しています。

```
CREATE TABLE sample_regional_sales
      (deptno number, item_no varchar2(20),
       txn_date date, txn_amount number, state varchar2(2))
  PARTITION BY RANGE (txn_date)
    SUBPARTITION BY LIST (state)
      (PARTITION q1_1999 VALUES LESS THAN (TO_DATE('1-APR-1999','DD-MON-YYYY'))
          TABLESPACE tbs_1
         (SUBPARTITION q1_1999_northwest VALUES ('OR', 'WA'),
          SUBPARTITION q1_1999_southwest VALUES ('AZ', 'UT', 'NM'),
          SUBPARTITION q1_1999_northeast VALUES ('NY', 'VM', 'NJ'),
          SUBPARTITION q1_1999_southeast VALUES ('FL', 'GA'),
          SUBPARTITION q1_others VALUES (DEFAULT) TABLESPACE tbs_4
         ),
       PARTITION q2_1999 VALUES LESS THAN ( TO_DATE('1-JUL-1999','DD-MON-YYYY'))
          TABLESPACE tbs_2
         (SUBPARTITION q2_1999_northwest VALUES ('OR', 'WA'),
          SUBPARTITION q2_1999_southwest VALUES ('AZ', 'UT', 'NM'),
          SUBPARTITION q2_1999_northeast VALUES ('NY', 'VM', 'NJ'),
          SUBPARTITION q2_1999_southeast VALUES ('FL', 'GA'),
          SUBPARTITION q2_1999_northcentral VALUES ('SD', 'WI'),
          SUBPARTITION q2_1999_southcentral VALUES ('OK', 'TX')
         ),
       PARTITION q3_1999 VALUES LESS THAN (TO_DATE('1-OCT-1999','DD-MON-YYYY'))
          TABLESPACE tbs_3
         (SUBPARTITION q3_1999_northwest VALUES ('OR', 'WA'),
```

```
      SUBPARTITION q3_1999_southwest VALUES ('AZ', 'UT', 'NM'),
      SUBPARTITION q3_others VALUES (DEFAULT) TABLESPACE tbs_4
    ),
  PARTITION q4_1999 VALUES LESS THAN ( TO_DATE('1-JAN-2000','DD-MON-YYYY'))
      TABLESPACE tbs_4
);
```

この例では、サブパーティション記述は次のようになります。

- すべてのサブパーティションは、表領域を除く物理属性を表領域レベルのデフォルトから継承します。これは、パーティションまたはサブパーティションに対して指定されている物理属性が表領域のみであるためです。表レベルの物理属性は指定されていないので、すべてのレベルで表領域レベルのデフォルトが継承されます。
- パーティション q1_1999 の最初の 4 つのサブパーティションは、すべて tbs_1 に含まれていますが、サブパーティション q1_others は tbs_4 に格納されており、他のどのパーティションにもマップされない行がすべて含まれています。
- パーティション q2_1999 の 6 つのサブパーティションは、すべて tbs_2 に格納されています。
- パーティション q3_1999 の最初の 2 つのサブパーティションは、すべて tbs_3 に含まれていますが、サブパーティション q3_others は tbs_4 に格納されており、他のどのパーティションにもマップされない行がすべて含まれています。
- パーティション q4_1999 のサブパーティション記述はありません。このため、デフォルト・サブパーティションが 1 つ作成され、tbs_4 に格納されます。このサブパーティションには、SYS_SUBP*n* 形式のシステム生成名が使用されます。

サブパーティション・テンプレートを使用してコンポジット・パーティション表の指定を簡素化する方法は、「サブパーティション・テンプレートを使用したコンポジット・パーティション表の記述」を参照してください。

サブパーティション・テンプレートを使用したコンポジット・パーティション表の記述

サブパーティション・テンプレートを使用すると、コンポジット・パーティション表にサブパーティションを作成できます。サブパーティション・テンプレートでは、表の各パーティションのサブパーティション記述子を指定する必要がないため、サブパーティションの指定が簡素化されます。かわりに、テンプレート内でサブパーティションを一度記述し、そのサブパーティション・テンプレートを表のすべてのパーティションに適用します。

サブパーティション・テンプレートは、パーティションのサブパーティション記述子が指定されていない場合に使用されます。サブパーティション記述子が指定されている場合は、そのパーティションのサブパーティション・テンプレートのかわりに使用されます。サブパーティション・テンプレートもパーティションのサブパーティション記述子も指定されていない場合は、デフォルト・サブパーティションが 1 つ作成されます。

レンジ - ハッシュ・パーティション表のサブパーティション・テンプレートの指定

レンジ - ハッシュ・パーティション表の場合は、サブパーティション・テンプレートでサブパーティションの詳細を記述する方法と、ハッシュ・サブパーティションの数のみを指定する方法があります。

次の例では、サブパーティション・テンプレートを使用してレンジ - ハッシュ・パーティション表を作成しています。

```
CREATE TABLE emp_sub_template (deptno NUMBER, empname VARCHAR(32), grade NUMBER)
     PARTITION BY RANGE(deptno) SUBPARTITION BY HASH(empname)
     SUBPARTITION TEMPLATE
         (SUBPARTITION a TABLESPACE ts1,
          SUBPARTITION b TABLESPACE ts2,
          SUBPARTITION c TABLESPACE ts3,
          SUBPARTITION d TABLESPACE ts4
         )
    (PARTITION p1 VALUES LESS THAN (1000),
     PARTITION p2 VALUES LESS THAN (2000),
     PARTITION p3 VALUES LESS THAN (MAXVALUE)
    );
```

この例では、次の表記述が生成されます。

- 各パーティションには、サブパーティション・テンプレートに記述されているように 4 つのサブパーティションがあります。
- 各サブパーティションには表領域が指定されています。あるサブパーティションの表領域をサブパーティション・テンプレートで指定する場合は、すべてのサブパーティション用に表領域を指定する必要があります。
- サブパーティション名は、パーティション名とサブパーティション名を次の書式で連結して生成されます。

  partition name_subpartition name

次の問合せでは、サブパーティション名と表領域が表示されます。

```
SQL> SELECT TABLESPACE_NAME, PARTITION_NAME, SUBPARTITION_NAME
  2  FROM DBA_TAB_SUBPARTITIONS WHERE TABLE_NAME='EMP_SUB_TEMPLATE'
  3  ORDER BY TABLESPACE_NAME;

TABLESPACE_NAME PARTITION_NAME  SUBPARTITION_NAME
--------------- --------------- ------------------
TS1             P1              P1_A
TS1             P2              P2_A
TS1             P3              P3_A
TS2             P1              P1_B
TS2             P2              P2_B
```

```
TS2             P3              P3_B
TS3             P1              P1_C
TS3             P2              P2_C
TS3             P3              P3_C
TS4             P1              P1_D
TS4             P2              P2_D
TS4             P3              P3_D

12 rows selected.
```

レンジ - リスト・パーティション表のサブパーティション・テンプレートの指定

レンジ - リスト・パーティション表に関する次の例は、サブパーティション・テンプレートを使用して表領域間でデータをストライプ化する方法を示しています。この例で作成される表では、サブパーティションが垂直にストライプ化されています。これは、各パーティションのサブパーティションが同じ表領域にあることを意味します。

```
CREATE TABLE stripe_regional_sales
            ( deptno number, item_no varchar2(20),
              txn_date date, txn_amount number, state varchar2(2))
   PARTITION BY RANGE (txn_date)
   SUBPARTITION BY LIST (state)
   SUBPARTITION TEMPLATE
      (SUBPARTITION northwest VALUES ('OR', 'WA') TABLESPACE tbs_1,
       SUBPARTITION southwest VALUES ('AZ', 'UT', 'NM') TABLESPACE tbs_2,
       SUBPARTITION northeast VALUES ('NY', 'VM', 'NJ') TABLESPACE tbs_3,
       SUBPARTITION southeast VALUES ('FL', 'GA') TABLESPACE tbs_4,
       SUBPARTITION midwest VALUES ('SD', 'WI') TABLESPACE tbs_5,
       SUBPARTITION south VALUES ('AL', 'AK') TABLESPACE tbs_6,
       SUBPARTITION others VALUES (DEFAULT ) TABLESPACE tbs_7
      )
  (PARTITION q1_1999 VALUES LESS THAN ( TO_DATE('01-APR-1999','DD-MON-YYYY')),
   PARTITION q2_1999 VALUES LESS THAN ( TO_DATE('01-JUL-1999','DD-MON-YYYY')),
   PARTITION q3_1999 VALUES LESS THAN ( TO_DATE('01-OCT-1999','DD-MON-YYYY')),
   PARTITION q4_1999 VALUES LESS THAN ( TO_DATE('1-JAN-2000','DD-MON-YYYY'))
  );
```

パーティション・レベルで表領域を指定し（パーティション q1_1999 には tbs_1、パーティション q1_1999 には tbs_2、パーティション q3_1999 には tbs_3 およびパーティション q4_1999 には tbs_4 など）、サブパーティション・テンプレートで指定しないと、表は水平にストライプ化されます。すべてのサブパーティションは、所有パーティションの表領域に格納されます。

パーティション化された索引構成表の作成

索引構成表では、レンジ・パーティション化方法またはハッシュ・パーティション化方法が使用できます。ただし、LOB を使用した列を含むことができるのは、レンジ・パーティション化された索引構成表のみです。レンジ・パーティション化またはハッシュ・パーティション化された索引構成表を作成するセマンティクスは、通常の表を作成するセマンティクスとほぼ同じですが、次の点が異なります。

- 索引構成表を作成するときは、ORGANIZATION INDEX 句を指定し、さらに必要に応じて INCLUDING 句および OVERFLOW 句を指定します。
- PARTITION 句または PARTITIONS 句には、パーティション・レベルでオーバーフロー・セグメントの属性を指定できる OVERFLOW 副次句を指定できます。

OVERFLOW 句を指定すると、オーバーフロー・データ・セグメント自体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。したがって、オーバーフローを指定してパーティション化された索引構成表では、各パーティションが索引セグメントとオーバーフロー・データ・セグメントを持っています。

索引構成表では、パーティション化列のセットは主キー列のサブセットであることが必要です。索引構成表の行はその表の主キー索引に格納されるため、パーティション化の基準が可用性に影響を及ぼします。パーティション化キーを主キーのサブセットとすることによって、単一パーティション内で主キーの一意性が確認されると行を挿入できます。これにより、パーティションの独立性が維持されます。

索引構成表上での 2 次索引のサポートは、通常の表に対するサポートとほぼ同じです。ただし、特定のメンテナンス操作では、通常の表の場合と異なり、グローバル索引に UNUSABLE マークが付けられません。

関連項目：

- 15-24 ページ「索引構成表の管理」
- 17-22 ページ「パーティション表のメンテナンス」
- 索引構成表の詳細は、『Oracle9*i* アプリケーション開発者ガイドー基礎編』および『Oracle9*i* データベース概要』を参照してください。

レンジ・パーティション化された索引構成表の作成

索引構成表とその 2 次索引は、レンジ・パーティション化方法でパーティション化できます。次の例では、レンジ・パーティション化された索引構成表 sales が作成されます。INCLUDING 句では、week_no より後の列がすべてオーバーフロー・セグメントに格納されるよう指定しています。各パーティションにはオーバーフロー・セグメントが 1 つあり、すべて同じ表領域に格納されます（overflow_here）。必要に応じて、個々のパーティション・レベルで OVERFLOW TABLESPACE を指定できます。その場合、一部またはすべてのオーバーフロー・セグメントに、別個の TABLESPACE 属性を指定できます。

```
CREATE TABLE sales(acct_no NUMBER(5),
                   acct_name CHAR(30),
                   amount_of_sale NUMBER(6),
                   week_no INTEGER,
                   sale_details VARCHAR2(1000),
             PRIMARY KEY (acct_no, acct_name, week_no))
     ORGANIZATION INDEX
             INCLUDING week_no
             OVERFLOW TABLESPACE overflow_here
     PARTITION BY RANGE (week_no)
            (PARTITION VALUES LESS THAN (5)
                   TABLESPACE ts1,
             PARTITION VALUES LESS THAN (9)
                   TABLESPACE ts2 OVERFLOW TABLESPACE overflow_ts2,
             ...
             PARTITION VALUES LESS THAN (MAXVALUE)
                   TABLESPACE ts13);
```

ハッシュ・パーティション化された索引構成表の作成

索引構成表をパーティション化する方法として、ハッシュ・パーティション化方法を使用することもできます。次の例では、索引構成表 sales がハッシュ・パーティション化方法でパーティション化されています。

```
CREATE TABLE sales(acct_no NUMBER(5),
                   acct_name CHAR(30),
                   amount_of_sale NUMBER(6),
                   week_no INTEGER,
                   sale_details VARCHAR2(1000),
             PRIMARY KEY (acct_no, acct_name, week_no))
     ORGANIZATION INDEX
             INCLUDING week_no
     OVERFLOW
    PARTITION BY HASH (week_no)
             PARTITIONS 16
             STORE IN (ts1, ts2, ts3, ts4)
             OVERFLOW STORE IN (ts3, ts6, ts9);
```

注意： 正しく設計されたハッシュ関数では行がパーティション間でバランスよく分散されるため、行の主キー列を更新すると、その行が別のパーティションに移動することがあります。したがって、変更可能なパーティション化キーを持つハッシュ・パーティション化された索引構成表を作成するときは、ROW MOVEMENT ENABLE 句を明示的に指定することをお薦めします。デフォルトでは、ROW MOVEMENT ENABLE は使用禁止になっています。

複数のブロック・サイズに関するパーティション化の制限

複数のブロック・サイズの表領域を持つデータベースにパーティション化されたオブジェクトを作成するときは、注意が必要です。このような表領域に格納されたパーティション・オブジェクトの記憶域には、いくつかの制限が適用されます。具体的には、次のエンティティのパーティションはすべて同じブロック・サイズの表領域内にあることが必要です。

- 従来型の表
- 索引
- 索引構成表の主キー索引セグメント
- 索引構成表のオーバーフロー・セグメント
- 行外に格納されている LOB 列

したがって、次のことが必要になります。

- 従来型の各表のパーティションはすべて同じブロック・サイズの表領域に格納する必要があります。
- 各索引構成表では、主キー索引のパーティションはすべて同じブロック・サイズの表領域内にあることが必要です。また、オーバーフロー・パーティションもすべて同じブロック・サイズの表領域内にあることが必要です。ただし、索引パーティションとオーバーフロー・パーティションが異なるブロック・サイズの表領域にあってもかまいません。
- 各索引（グローバルまたはローカル）の各パーティションは、同じブロック・サイズの表領域にあることが必要です。ただし、同じオブジェクトに対して定義されている異なる索引のパーティションは、異なるブロック・サイズの表領域にあってもかまいません。
- 各 LOB 列の各パーティションは、同じブロック・サイズの表領域に格納する必要があります。ただし、異なる LOB 列は異なるブロック・サイズの表領域に格納してもかまいません。

パーティション表または索引を作成または変更するときに、各エンティティのパーティションおよびサブパーティションに対して表領域を明示的に指定する場合、その表領域はすべて同じブロック・サイズであることが必要です。また、エンティティに対して表領域を明示的に指定しない場合は、デフォルトで使用される表領域が同じブロック・サイズであることが必要です。このため、パーティション・オブジェクトの各レベルにおけるデフォルトの表領域に注意する必要があります。

パーティション表のメンテナンス

ここでは、表と索引の両方について、パーティションとサブパーティションのメンテナンス操作の実行方法を説明します。

表 17-1 は、表パーティション（またはサブパーティション）に対して実行できるメンテナンス操作と、パーティション化のタイプごとにそのメンテナンス操作の実行に使用する ALTER TABLE 文の特定の句を示しています。

表 17-1 ALTER TABLE による表パーティションのメンテナンス操作

メンテナンス操作	レンジ	ハッシュ	リスト	コンポジット：レンジ / ハッシュ	コンポジット：レンジ / リスト
パーティションの追加	ADD PARTITION	ADD PARTITION	ADD PARTITION	ADD PARTITION MODIFY PARTITION...ADD SUBPARTITION	ADD PARTITION MODIFY PARTITION...ADD SUBPARTITION
パーティションの結合	N/A	COALESCE PARTITION	N/A	MODIFY PARTITION... COALESCE SUBPARTITION	N/A
パーティションの削除	DROP PARTITION	N/A	DROP PARTITION	DROP PARTITION	DROP PARTITION DROP SUBPARTITION
パーティションの交換	EXCHANGE PARTITION	EXCHANGE PARTITION	EXCHANGE PARTITION	EXCHANGE PARTITION EXCHANGE SUBPARTITION	EXCHANGE PARTITION EXCHANGE SUBPARTITION
パーティションのマージ	MERGE PARTITIONS	N/A	MERGE PARTITIONS	MERGE PARTITIONS	MERGE PARTITIONS MERGE SUBPARTITIONS
デフォルト属性の変更	MODIFY DEFAULT ATTRIBUTES	MODIFY DEFAULT ATTRIBUTES	MODIFY DEFAULT ATTRIBUTES	MODIFY DEFAULT ATTRIBUTES MODIFY DEFAULT ATTRIBUTES FOR PARTITION	MODIFY DEFAULT ATTRIBUTES MODIFY DEFAULT ATTRIBUTES FOR PARTITION
パーティションの実属性の変更	MODIFY PARTITION	MODIFY PARTITION	MODIFY PARTITION	MODIFY PARTITION MODIFY SUBPARTITION	MODIFY PARTITION MODIFY SUBPARTITION
リスト・パーティションの変更：値の追加	N/A	N/A	MODIFY PARTITION... ADD VALUES	N/A	MODIFY SUBPARTITION... ADD VALUES
リスト・パーティションの変更：値の削除	N/A	N/A	MODIFY PARTITION... DROP VALUES	N/A	MODIFY SUBPARTITION... DROP VALUES

表 17-1 ALTER TABLE による表パーティションのメンテナンス操作（続き）

メンテナンス操作	レンジ	ハッシュ	リスト	コンポジット：レンジ / ハッシュ	コンポジット：レンジ / リスト
サブパーティション・テンプレートの変更	N/A	N/A	N/A	SET SUBPARTITION TEMPLATE	SET SUBPARTITION TEMPLATE
パーティションの移動	MOVE PARTITION	MOVE PARTITION	MOVE PARTITION	MOVE SUBPARTITION	MOVE SUBPARTITION
パーティションの名前変更	RENAME PARTITION	RENAME PARTITION	RENAME PARTITION	RENAME PARTITION RENAME SUBPARTITION	RENAME PARTITION RENAME SUBPARTITION
パーティションの分割	SPLIT PARTITION	N/A	SPLIT PARTITION	SPLIT PARTITION	SPLIT PARTITION SPLIT SUBPARTITION
パーティションの切捨て	TRUNCATE PARTITION	TRUNCATE PARTITION	TRUNCATE PARTITION	TRUNCATE PARTITION TRUNCATE SUBPARTITION	TRUNCATE PARTITION TRUNCATE SUBPARTITION

注意： ビットマップ索引を持ち、現在は非圧縮パーティションのみが含まれているパーティション表に、圧縮パーティションを初めて導入する場合は、次の操作が必要です。

- 既存のビットマップ索引とビットマップ索引パーティションをすべて削除するか、UNUSABLE マークを付けます。
- 圧縮属性を設定します。
- 索引を再作成します。

これらのアクションは、パーティションにデータが含まれているかどうかや圧縮パーティションの導入操作からは独立しています。

これらのアクションは、B ツリー索引を持つパーティション表には適用されません。

詳細は、『Oracle9*i* データ・ウェアハウス・ガイド』を参照してください。

表 17-2 は、索引パーティションに対して実行できるメンテナンス操作と、各操作を実行できる索引のタイプ（グローバルまたはローカル）を示しています。メンテナンス操作には、ALTER INDEX 句が示されています。

グローバル索引は基礎となる表の構造を反映せず、パーティション化する場合はレンジ・パーティション化のみ有効です。レンジ・パーティション索引には、レンジ・パーティション表に対して実行できるパーティション・メンテナンス操作の一部（すべてではありません）を実行できます。

ローカル索引は基礎となる表の構造を反映するため、表パーティションとサブパーティションがメンテナンス・アクティビティの影響を受ける場合にパーティション化は自動的にメンテナンスされます。したがって、ローカル索引のパーティションをメンテナンスする機会はそれほどなく、オプションも少数です。

表 17-2 ALTER INDEX による索引パーティションのメンテナンス操作

		索引パーティション化のタイプ		
メンテナンス操作	**索引のタイプ**	**レンジ**	**ハッシュおよびリスト**	**コンポジット**
索引パーティションの削除	グローバル	DROP PARTITION	-	-
	ローカル	N/A	N/A	N/A
索引パーティションのデフォルト属性の変更	グローバル	MODIFY DEFAULT ATTRIBUTES	-	-
	ローカル	MODIFY DEFAULT ATTRIBUTES	MODIFY DEFAULT ATTRIBUTES	MODIFY DEFAULT ATTRIBUTES MODIFY DEFAULT ATTRIBUTES FOR PARTITION
索引パーティションの実属性の変更	グローバル	MODIFY PARTITION	-	-
	ローカル	MODIFY PARTITION	MODIFY PARTITION	MODIFY PARTITION MODIFY SUBPARTITION
索引パーティションの再作成	グローバル	REBUILD PARTITION	-	-
	ローカル	REBUILD PARTITION	REBUILD PARTITION	REBUILD SUBPARTITION
索引パーティションの名前変更	グローバル	RENAME PARTITION	-	-
	ローカル	RENAME PARTITION	RENAME PARTITION	RENAME PARTITION RENAME SUBPARTITION
索引パーティションの分割	グローバル	SPLIT PARTITION	-	-
	ローカル	N/A	N/A	N/A

注意： 次の項では、パーティション表に対するメンテナンス操作について説明します。メンテナンス操作の影響を受ける索引または索引パーティションの使用可能性についての記述では、次のことを考慮してください。

- UNUSABLE でマークされる対象は、空ではない索引および索引パーティションのみです。これらが空である場合、USABLE/UNUSABLE 状態は変更されません。
- 後続の DML で更新されるのは、USABLE 状態の索引または索引パーティションのみです。

グローバル索引の自動更新

パーティション表および索引に対する個々のメンテナンス操作を説明する前に、ALTER TABLE 文に指定できる UPDATE GLOBAL INDEXES 句の効果について説明する必要があります。

デフォルトでは、パーティション表のメンテナンス操作を実行すると、多くの場合グローバル索引が無効になります（UNUSABLE マークが設定されます）。このような場合は、グローバル索引全体を再作成する必要があります。また、パーティション化されている場合は、そのパーティションをすべて再作成する必要があります。メンテナンス操作に対して、ALTER TABLE 文で UPDATE GLOBAL INDEXES 句を指定することにより、このデフォルト動作を無効にできます。この句を指定すると、メンテナンス操作のデータ定義言語（DDL）文の実行時にグローバル索引が更新されます。この方法には、次のような利点があります。

- 実表の操作と同時にグローバル索引が更新されます。後で個別にグローバル索引を再作成する必要はありません。
- グローバル索引が UNUSABLE でマークされないため、グローバル索引の可用性が向上します。パーティションの DDL を実行している間でもグローバル索引は使用可能であり、表中の他のパーティションへのアクセスにも使用できます。
- グローバル索引を再作成するために、無効になっている索引の名前を検索する必要がありません。

ただし、UPDATE GLOBAL INDEXES 句の指定時には、次のようなパフォーマンスに関する考慮事項があります。

- すでに UNUSABLE でマークされている索引が更新されるため、パーティションの DDL 文の実行には時間がかかります。ただし、この時間は、索引を更新せずに DDL 文を実行して、その後すべての索引を再作成する作業に必要な時間と比較する必要があります。一般に、パーティションのサイズが表のサイズの 5% 未満であれば、索引を更新したほうが処理時間は短くなります。
- DROP、TRUNCATE および EXCHANGE の各操作は、決して速くありません。繰り返しますが、DDL を実行した後、すべてのグローバル索引を再作成するのに必要な時間と比較してください。

- 索引の更新がログに記録され、REDO レコードと UNDO レコードが生成されます。索引全体を再作成する場合は、NOLOGGING で実行することもできます。
- 索引全体を再作成すると、領域を有効に利用して索引サイズが縮小されるため、より効率的な索引になります。また、索引の再作成では記憶域オプションを変更できます。

注意： パーティション化された索引構成表では、UPDATE GLOBAL INDEXES 句はサポートされていません。

UPDATE GLOBAL INDEXES 句がサポートされているのは、次の操作です。

- ADD PARTITION|SUBPARTITION（ハッシュのみ）
- COALESCE PARTITION|SUBPARTITION
- DROP PARTITION
- EXCHANGE PARTITION|SUBPARTITION
- MERGE PARTITION
- MOVE PARTITION|SUBPARTITION
- SPLIT PARTITION
- TRUNCATE PARTITION|SUBPARTITION

パーティションの追加

ここでは、新しいパーティションをパーティション表に追加する方法と、パーティションを特定のグローバル・パーティション索引またはローカル索引に追加できない理由について説明します。

レンジ・パーティション表へのパーティションの追加

ALTER TABLE ... ADD PARTITION 文を使用すると、新しいパーティションが最後尾（既存の最後のパーティションの次の位置）に追加されます。パーティションを表の先頭または途中に追加する場合は、SPLIT PARTITION 句を使用します。

たとえば、sales という表があり、今月と過去 12 か月分のデータが含まれているとします。1999 年 1 月 1 日に、1 月用のパーティションを表領域 tsx に追加します。

```
ALTER TABLE sales
      ADD PARTITION jan96 VALUES LESS THAN ( '01-FEB-1999' )
      TABLESPACE tsx;
```

レンジ・パーティション表に関連付けられたローカル索引およびグローバル索引は、使用可能のままです。

ハッシュ・パーティション表へのパーティションの追加

ハッシュ・パーティション表にパーティションを追加すると、新しいパーティションには、ハッシュ関数で決定された既存のパーティション（Oracle が選択）から再ハッシュされた行が移入されます。

次の文は、表 scubagear にハッシュ・パーティションを追加する 2 つの方法を示しています。最初の文を選択すると、システムによって生成されたパーティション名を持つ新しいハッシュ・パーティションが追加され、表のデフォルト表領域に配置されます。第 2 の文でも新しいハッシュ・パーティションが追加されますが、そのパーティションは明示的に p_named と命名され、表領域 gear5 に作成されます。

```
ALTER TABLE scubagear ADD PARTITION;

ALTER TABLE scubagear
      ADD PARTITION p_named TABLESPACE gear5;
```

索引は、次の表に示すように UNUSABLE にマークされる場合があります。

表のタイプ	索引の動作
通常の表（ヒープ）	■ 新しいパーティションおよび行が再分散される元の既存パーティションのローカル索引は UNUSABLE でマークされるため、再作成する必要があります。 ■ UPDATE GLOBAL INDEXES を指定しないかぎり、すべてのグローバル索引、またはパーティション化されたグローバル索引のすべてのパーティションには UNUSABLE マークが付けられるため、それらを再作成する必要があります。
索引構成表	■ ローカル索引については、ヒープ表の場合と同じように動作します。 ■ グローバル索引はすべて使用可能のままです。

リスト・パーティション表へのパーティションの追加

次の文は、リスト・パーティション表に新しいパーティションを追加する方法を示しています。この例では、追加するパーティションに物理属性および NOLOGGING を指定しています。

```
ALTER TABLE q1_sales_by_region
   ADD PARTITION q1_nonmainland VALUES ('HI', 'PR')
      STORAGE (INITIAL 20K NEXT 20K) TABLESPACE tbs_3
      NOLOGGING;
```

追加するパーティションを記述するリテラル値のセットには、表の他のパーティションに存在しない値を指定する必要があります。

デフォルト・パーティションを持つリスト・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が、デフォルト・パーティションを分割することはできます。分割すると、実際には指定した値で定義される新しいパーティションが作成され、2 番目のパーティションが引き続きデフォルト・パーティションとなります。

リスト・パーティション表に関連付けられたローカル索引およびグローバル索引は、使用可能のままです。

レンジ - ハッシュ・パーティション表へのパーティションの追加

パーティションは、レンジ・パーティション・レベルとハッシュ・サブパーティション・レベルのどちらでも追加できます。

レンジ - ハッシュ・パーティション表へのパーティションの追加 レンジ - ハッシュ・パーティション表に新しいレンジ・パーティションを追加する方法は、すでに「レンジ・パーティション表へのパーティションの追加」で説明されています。また、SUBPARTITIONS 句を指定して特定の数のサブパーティションを追加したり、SUBPARTITION 句を指定して特定のサブパーティションを命名することができます。SUBPARTITIONS 句または SUBPARTITION 句を指定しない場合、パーティションはサブパーティションに表レベルのデフォルトを継承します。

次の例では、表 sales にレンジ・パーティション q1_2000 を追加しています。このパーティションには、2000 年の第 1 四半期のデータが移入されます。表領域 tbs5 には、8 個のサブパーティションが格納されます。

```
ALTER TABLE sales ADD PARTITION q1_2000
      VALUES LESS THAN (2000, 04, 01)
      SUBPARTITIONS 8 STORE IN tbs5;
```

レンジ - ハッシュ・パーティション表へのサブパーティションの追加 レンジ - ハッシュ・パーティション表にハッシュ・サブパーティションを追加するには、ALTER TABLE 文の MODIFY PARTITION ... ADD SUBPARTITION 句を使用します。新しく追加したサブパーティションには、ハッシュ関数で決定されたのと同じパーティションの他のサブパーティションから再ハッシュされた行が移入されます。

次の例では、表領域 us1 に格納されている新しいハッシュ・サブパーティション us_loc5 が、表 diving のレンジ・パーティション locations_us に追加されます。

```
ALTER TABLE diving MODIFY PARTITION locations_us
      ADD SUBPARTITION us_locs5 TABLESPACE us1;
```

追加されて再ハッシュされたサブパーティションに対応するローカル索引のパーティションは、再作成する必要があります。UPDATE GLOBAL INDEXES を指定しないかぎり、すべてのグローバル索引、またはパーティション化されたグローバル索引のすべてのパーティションには UNUSABLE マークが付けられるため、それらを再作成する必要があります。

レンジ - リスト・パーティション表へのパーティションの追加

パーティションは、レンジ・パーティション・レベルとリスト・サブパーティション・レベルのどちらでも追加できます。

レンジ - リスト・パーティション表へのパーティションの追加 レンジ - リスト・パーティション表に新しいレンジ・パーティションを追加する方法は、すでに「レンジ・パーティション表へのパーティションの追加」で説明されています。ただし、SUBPARTITION 句でサブパーティションの名前と値リストを指定できます。SUBPARTITION 句を指定しないと、パーティションはサブパーティション・テンプレートを継承します。サブパーティション・テンプレートが存在しない場合は、デフォルト・サブパーティションが 1 つ作成されます。

次の文は、レンジ - リスト方法によってパーティション化されている quarterly_regional_sales 表に新しいパーティションを追加します。この新しいパーティションには新しい物理属性がいくつか指定されていますが、未指定の物理属性については表レベルのデフォルトが継承されます。

```
ALTER TABLE quarterly_regional_sales
   ADD PARTITION q1_2000 VALUES LESS THAN (TO_DATE('1-APR-2000','DD-MON-YYYY'))
      STORAGE (INITIAL 20K NEXT 20K) TABLESPACE ts3 NOLOGGING
         (
          SUBPARTITION q1_2000_northwest VALUES ('OR', 'WA'),
          SUBPARTITION q1_2000_southwest VALUES ('AZ', 'UT', 'NM'),
          SUBPARTITION q1_2000_northeast VALUES ('NY', 'VM', 'NJ'),
          SUBPARTITION q1_2000_southeast VALUES ('FL', 'GA'),
          SUBPARTITION q1_2000_northcentral VALUES ('SD', 'WI'),
          SUBPARTITION q1_2000_southcentral VALUES ('OK', 'TX')
         );
```

レンジ - リスト・パーティション表へのサブパーティションの追加 レンジ - リスト・パーティション表にリスト・サブパーティションを追加するには、ALTER TABLE 文の MODIFY PARTITION ... ADD SUBPARTITION 句を使用します。

次の文は、レンジ - リスト・パーティション表 quarterly_regional_sales の既存のサブパーティション・セットに、新しいサブパーティションを追加します。新しいサブパーティションは、表領域 ts2 に作成されます。

```
ALTER TABLE quarterly_regional_sales
   MODIFY PARTITION q1_1999
      ADD SUBPARTITION q1_1999_south
         VALUES ('AR','MS','AL') tablespace ts2;
```

索引パーティションの追加

ローカル索引には、パーティションを明示的に追加できません。新しいパーティションをローカル索引に追加できるのは、パーティションをその基礎となる表に追加するときのみです。具体的には、表にローカル索引が定義されているときに、ALTER TABLE 文を発行してパーティションを追加すると、それに対応するパーティションもローカル索引に追加されます。新しい索引パーティションには、Oracle によって名前とデフォルトの物理記憶域属性が割り当てられますが、ADD PARTITION 操作が完了した後にそれらを改名または変更できます。

実際には、最初に索引のデフォルト属性を変更し、ADD PARTITION 操作で索引パーティション用に新しい表領域を指定できます。たとえば、リスト・パーティション表 q1_sales_by_region のローカル索引 q1_sales_by_region_locix が作成されているとします。17-27 ページの「リスト・パーティション表へのパーティションの追加」のように新しいパーティション q1_nonmainland を追加する前に、次の文を発行すると、対応する索引パーティションが表領域 tbs_4 に作成されます。

```
ALTER INDEX q1_sales_by_region_locix
   MODIFY DEFAULT ATTRIBUTES TABLESPACE tbs_4;
```

それ以外の場合は、次の文を使用して、索引パーティションを追加した後に tbs_4 に移動する必要があります。

```
ALTER INDEX q1_sales_by_region_locix
   REBUILD PARTITION q1_nonmainland TABLESPACE tbs_4;
```

最高位のパーティションのパーティション・バウンドは常に MAXVALUE であるため、グローバル索引にはパーティションを追加できません。最高位のパーティションを新しく追加する場合は、ALTER INDEX ... SPLIT PARTITION 文を使用してください。

パーティションの結合

パーティションを結合すると、ハッシュ・パーティション表のパーティション数や、レンジ - ハッシュ・パーティション表のサブパーティション数を減らすことができます。ハッシュ・パーティションを結合すると、その内容はハッシュ関数で決定された残りの 1 つ以上のパーティションに再分散されます。結合する特定のパーティションは Oracle によって選択され、その内容が再分散された後に削除されます。

索引は、次の表に示すように UNUSABLE にマークされる場合があります。

表のタイプ	索引の動作
通常の表（ヒープ）	■ 選択されたパーティションに対応するローカル索引のパーティションも削除されます。結合した側の 1 つ以上の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は UNUSABLE マークが付けられるため、それらを再作成する必要があります。 ■ UPDATE GLOBAL INDEXES を指定しないかぎり、すべてのグローバル索引、またはパーティション化されたグローバル索引のすべてのパーティションには UNUSABLE マークが付けられるため、それらを再作成する必要があります。
索引構成表	■ 前述のように、ローカル索引には UNUSABLE マークが付けられる場合があります。 ■ グローバル索引はすべて使用可能のままです。

ハッシュ・パーティション表のパーティションの結合

ハッシュ・パーティション表のパーティションを結合するには、ALTER TABLE ... COALESCE PARTITION 文を使用します。次の文は、パーティションを結合することによって、表のパーティション数を 1 つ減らします。

```
ALTER TABLE ouu1
     COALESCE PARTITION;
```

レンジ - ハッシュ・パーティション表のサブパーティションの結合

次の文は、パーティション us_locations のサブパーティションの内容を、同じパーティション内にある残りの 1 つ以上のサブパーティション（ハッシュ関数で決定）に分散させます。基本的に、この操作は、17-28 ページの「レンジ - ハッシュ・パーティション表へのサブパーティションの追加」で説明した MODIFY PARTITION ... ADD SUBPARTITION 句とは逆の効果を持ちます。

```
ALTER TABLE diving MODIFY PARTITION us_locations
     COALESCE SUBPARTITION;
```

パーティションの削除

レンジ・パーティション表、コンポジット・パーティション表、リスト・パーティション表またはレンジ - リスト・コンポジット・パーティション表からは、パーティションを削除できます。ハッシュ・パーティション表、またはレンジ - ハッシュ・パーティション表のハッシュ・サブパーティションでは、かわりに結合操作を実行する必要があります。

表パーティションの削除

次のいずれかの文を使用して、表のパーティションまたはサブパーティションを削除します。

- 表パーティションを削除するには、ALTER TABLE ... DROP PARTITION 文を使用します。
- レンジ - リスト・パーティション表のサブパーティションを削除するには、ALTER TABLE ... DROP SUBPARTITION 文を使用します。

パーティション内のデータを保つ場合は、DROP PARTITION 文のかわりに MERGE PARTITION 文を使用します。

表にローカル索引が定義されている場合は、この文によって、ローカル索引から対応するパーティションまたはサブパーティションも削除されます。次のいずれかの条件に該当しないかぎり、すべてのグローバル索引、またはパーティション化されたグローバル索引のすべてのパーティションには UNUSABLE マークが付けられます。

- UPDATE GLOBAL INDEXES を指定している場合（索引構成表には指定できません）
- 削除する索引またはそのサブパーティションが空の場合

注意： 表に 1 つしかないパーティションは削除できません。かわりに、その表を削除する必要があります。

ここでは、表パーティションの削除方法をいくつか示します。

データとグローバル索引を含む表からのパーティションの削除 パーティションにデータが含まれており、表でグローバル索引が 1 つ以上定義されている場合、表パーティションの削除には次のいずれかの方法を使用してください。

方法 1:

グローバル索引の更新を指定せずに、ALTER TABLE ... DROP PARTITION 文を実行します。文の実行後、索引（または索引パーティション）には UNUSABLE マークが付けられるため、グローバル索引を（パーティション化されているかどうかに関係なく）再作成する必要があります。次の文は、sales 表からパーティション dec98 を削除し、パーティション化されていないグローバル索引を再作成する例を示しています。

```
ALTER TABLE sales DROP PARTITION dec98;
ALTER INDEX sales_area_ix REBUILD;
```

索引 sales_area_ix がレンジ・パーティション化されたグローバル索引である場合は、そのすべてのパーティションを再作成する必要があります。1 つの文で索引のすべてのパーティションを再作成することはできません。索引のパーティションごとに個別の REBUILD 文を実行する必要があります。次の文は、索引パーティション jan99_ix、feb99_ix、mar99_ix、…、dec99_ix を再作成します。

```
ALTER INDEX sales_area_ix REBUILD PARTITION jan99_ix;
ALTER INDEX sales_area_ix REBUILD PARTITION feb99_ix;
ALTER INDEX sales_area_ix REBUILD PARTITION mar99_ix;
...
ALTER INDEX sales_area_ix REBUILD PARTITION dec99_ixi
```

削除するパーティションに、その表の全データの大部分が含まれるような大規模な表の場合には、この方法が最適です。

方法 2:

ALTER TABLE ... DROP PARTITION 文を発行する前に、DELETE 文を発行し、パーティションからすべての行を削除します。DELETE 文でグローバル索引が更新され、さらにトリガーが起動されて、REDO ログおよび UNDO ログが生成されます。

たとえば、パーティション・バウンド 10,000 の最初のパーティションを削除する場合は、次の文を発行します。

```
DELETE FROM sales WHERE TRANSID < 10000;
ALTER TABLE sales DROP PARTITION dec98;
```

これは、小さい表の場合、または削除するパーティションにその表の全データのうちごく一部分のみが含まれるような大規模な表の場合に最適な方法です。

方法 3:

ALTER TABLE 文で UPDATE GLOBAL INDEXES を指定します。これにより、パーティションの削除時にグローバル索引が更新されるようになります。

```
ALTER TABLE sales DROP PARTITION dec98
     UPDATE GLOBAL INDEXES;
```

データおよび参照整合性制約を含むパーティションの削除 パーティションにデータおよび参照整合性制約が含まれる場合、表パーティションを削除するには次のいずれかの方法を使用します。この表にはローカル索引しかないため、索引を再作成する必要はありません。

方法 1:

整合性制約を使用禁止にし、ALTER TABLE ... DROP PARTITION 文を発行してから、整合性制約を使用可能にします。

```
ALTER TABLE sales
   DISABLE CONSTRAINT dname_sales1;
ALTER TABLE sales DROP PARTITTION dec98;
ALTER TABLE sales
   ENABLE CONSTRAINT dname_sales1;
```

削除するパーティションに、その表の全データの大部分が含まれるような大規模な表の場合には、この方法が最適です。

方法 2:

ALTER TABLE ... DROP PARTITION 文を発行する前に、DELETE 文を発行し、パーティションからすべての行を削除します。DELETE 文によって参照整合性制約が適用され、さらにトリガーが起動されて、REDO ログおよび UNDO ログが生成されます。

```
DELETE FROM sales WHERE TRANSID < 10000;
ALTER TABLE sales DROP PARTITION dec94;
```

これは、小さい表の場合、または削除するパーティションにその表の全データのごく一部分のみが含まれるような大規模な表の場合に最適な方法です。

索引パーティションの削除

ローカル索引のパーティションは、明示的には削除できません。ローカル索引のパーティションを削除できるのは、パーティションをその基礎である表から削除するときのみです。

グローバル索引では、パーティションが空の場合に ALTER INDEX ... DROP PARTITION 文を発行して明示的に削除できます。ただし、グローバル索引のパーティションにデータが含まれている場合にそのパーティションを削除すると、次の最高位パーティションに UNUSABLE マークが付けられます。たとえば、索引パーティション P1 を削除する場合に、P2 が次の最高位パーティションであるとすると、次の文を発行する必要があります。

```
ALTER INDEX npr DROP PARTITION P1;
ALTER INDEX npr REBUILD PARTITION P2;
```

注意： グローバル索引では、最高位のパーティションは削除できません。

パーティションの交換

表およびパーティション（またはサブパーティション）のデータ・セグメントを交換することによって、パーティション（またはサブパーティション）を非パーティション表に変換したり、表をパーティション表のパーティション（サブパーティション）に変換できます。また、ハッシュ・パーティション表からレンジ - ハッシュ・パーティション表のパーティションへの変換や、レンジ - ハッシュ・パーティション表のパーティションからハッシュ・パーティション表への変換も可能です。同様に、リスト・パーティション表からレンジ - リスト・パーティション表のパーティションへの変換や、レンジ - リスト・パーティション表のパーティションからリスト・パーティション表への変換も可能です。

表のパーティションの交換は、非パーティション表を使用するアプリケーションがあり、その非パーティション表をパーティション表のパーティションに変換する場合に役立ちます。たとえば、パーティション表に移行するパーティション・ビューがすでに存在する場合があります。パーティションの交換をトランスポータブル表領域と併用すると、高速にデータをロードできます。

パーティションを交換したときは、ロギング属性が保たれます。必要に応じて、ローカル索引の交換（INCLUDING INDEXES 句）や行のマッピングが正しいかどうかの検証を実行するように（WITH VALIDATION 句）指定できます。

注意： パーティション交換操作に対して WITHOUT VALIDATION を指定すると、データ・ディクショナリの更新のみが関係するため通常は操作が高速になります。ただし、交換操作に関係する表またはパーティション表に主キーまたは一意制約が使用可能になっている場合、交換操作は WITH VALIDATION を指定した場合と同様に実行されます。これは、制約の整合性を維持するためです。

この妥当性チェック・アクティビティによるオーバーヘッドを回避するために、制約ごとに次の文を発行してから、パーティション交換操作を実行します。

```
ALTER TABLE table_name
     DISABLE CONSTRAINT constraint_name KEEP INDEX
```

交換後に、制約を使用可能にします。

UPDATE GLOBAL INDEXES を指定しないかぎり（索引構成表には指定できません）、パーティションを交換する表のグローバル索引またはすべてのグローバル索引パーティションに UNUSABLE マークが付けられます。交換する表のグローバル索引またはグローバル索引パーティションにもすべて、UNUSABLE マークが付けられます。

関連項目：

- 17-61 ページ「パーティション・ビューからパーティション表への変換」
- トランスポータブル表領域の詳細は、11-45 ページの「トランスポータブル表領域の使用方法」を参照してください。

レンジ・パーティション、ハッシュ・パーティションまたはリスト・パーティションの交換

レンジ・パーティション表、ハッシュ・パーティション表またはリスト・パーティション表のパーティションと非パーティション表との間で交換するには、ALTER TABLE ... EXCHANGE PARTITION 文を使用します。パーティションを非パーティション表に変換する例を次に示します。この例では、表 stocks は、レンジ・パーティション、ハッシュ・パーティションまたはリスト・パーティションのいずれにも変換できます。

```
ALTER TABLE stocks
    EXCHANGE PARTITION p3 WITH stock_table_3;
```

ハッシュ・パーティション表とレンジ - ハッシュ・パーティションの交換

この例では、ハッシュ・パーティション表をそのすべてのパーティションとともに、レンジ - ハッシュ・パーティション表のレンジ・パーティションおよびそのすべてのハッシュ・サブパーティションと交換します。次の例に、この操作を示します。

最初に、ハッシュ・パーティション表を作成します。

```
CREATE TABLE t1 (i NUMBER, j NUMBER)
     PARTITION BY HASH(i)
       (PARTITION p1, PARTITION p2);
```

この表にデータを移入してから、次のようにレンジ - ハッシュ・パーティション表を作成します。

```
CREATE TABLE t2 (i NUMBER, j NUMBER)
     PARTITION BY RANGE(j)
     SUBPARTITION BY HASH(i)
        (PARTITION p1 VALUES LESS THAN (10)
            SUBPARTITION t2_pls1
            SUBPARTITION t2_pls2,
         PARTITION p2 VALUES LESS THAN (20)
            SUBPARTITION t2_p2s1
            SUBPARTITION t2_p2s2));
```

表 t1 のパーティション化キーが、表 t2 のサブパーティション化キーと同じである点に注意してください。

t1 のデータを t2 に移行し、各行を検証するには、次の文を使用します。

```
ALTER TABLE t1 EXCHANGE PARTITION p1 WITH TABLE t2
     WITH VALIDATION;
```

レンジ - ハッシュ・パーティション表のサブパーティションの交換

レンジ - ハッシュ・パーティション表のハッシュ・サブパーティションから非パーティション表への変換またはその逆の変換を行うには、ALTER TABLE ... EXCHANGE SUBPARTITION 文を使用します。次の例では、表 sales のサブパーティション q3_1999_s1 を非パーティション表 q3_1999 に変換しています。ローカル索引のパーティションは、q3_1999 の対応する索引と交換されます。

```
ALTER TABLE sales EXCHANGE SUBPARTITION q3_1999_s1
      WITH TABLE q3_1999 INCLUDING INDEXES;
```

リスト・パーティション表とレンジ - リスト・パーティションの交換

ALTER TABLE ... EXCHANGE PARTITION 文のセマンティクスは、すでに「ハッシュ・パーティション表とレンジ - ハッシュ・パーティションの交換」で説明したものと同じです。前述の例では、CREATE TABLE 文の構文を変更するのみで、それぞれリスト・パーティション表とレンジ - リスト・パーティション表を作成しました。関連するアクションも同じです。

レンジ - リスト・パーティション表のサブパーティションの交換

ALTER TABLE ... EXCHANGE SUBPARTITION 文のセマンティクスは、すでに「レンジ - ハッシュ・パーティション表のサブパーティションの交換」で説明したものと同じです。

パーティションのマージ

2 つのパーティションの内容を 1 つのパーティションにマージするには、ALTER TABLE ... MERGE PARTITIONS 文を使用します。元の 2 つのパーティションは、対応するローカル索引とともに削除されます。

この文は、ハッシュ・パーティション表や、レンジ - ハッシュ・パーティション表のハッシュ・サブパーティションには使用できません。

マージの対象となるパーティションまたはサブパーティションが空でないかぎり、次の表に示すように、索引には UNUSABLE マークが付けられます。

表のタイプ	索引の動作
通常の表（ヒープ）	■ マージされた対応する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はすべて、UNUSABLE マークが付けられます。 ■ UPDATE GLOBAL INDEXES を指定しないかぎり、すべてのグローバル索引、またはパーティション化されたグローバル索引のすべてのパーティションには UNUSABLE マークが付けられるため、それらを再作成する必要があります。
索引構成表	■ マージされた対応する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はすべて、UNUSABLE マークが付けられます。 ■ グローバル索引はすべて使用可能のままです。

レンジ・パーティションのマージ

隣接する 2 つのレンジ・パーティションの内容は、1 つのパーティションにマージできます。隣接していないレンジ・パーティションはマージできません。1 つにマージされたパーティションは、マージ前の上位のパーティションのバウンドを継承します。

レンジ・パーティションをマージする理由の 1 つに、履歴データを大きなパーティションでオンライン化しておくことがあります。たとえば、日付別のパーティションがある場合、最も古いパーティションを週次のパーティションにロールアップし、さらに月次パーティションにロールアップできます。

次のスクリプトは、レンジ・パーティションのマージ例を作成します。

最初に、パーティション表を作成し、ローカル索引を作成します。

```
-- Create a Table with four partitions each on its own tablespace
-- Partitioned by range on the data column.
--
CREATE TABLE four_seasons
(
        one DATE,
        two VARCHAR2(60),
        three NUMBER
)
PARTITION  BY RANGE ( one )
(
PARTITION quarter_one
   VALUES LESS THAN ( TO_DATE('01-apr-1998','dd-mon-yyyy'))
   TABLESPACE quarter_one,
PARTITION quarter_two
   VALUES LESS THAN ( TO_DATE('01-jul-1998','dd-mon-yyyy'))
   TABLESPACE quarter_two,
PARTITION quarter_three
```

```
   VALUES LESS THAN ( TO_DATE('01-oct-1998','dd-mon-yyyy'))
   TABLESPACE quarter_three,
PARTITION quarter_four
   VALUES LESS THAN ( TO_DATE('01-jan-1999','dd-mon-yyyy'))
   TABLESPACE quarter_four
);
--
-- Create local PREFIXED index on Four_Seasons
-- Prefixed because the leftmost columns of the index match the
-- Partition key
--
CREATE INDEX i_four_seasons_l ON four_seasons ( one,two )
LOCAL (
PARTITION i_quarter_one TABLESPACE i_quarter_one,
PARTITION i_quarter_two TABLESPACE i_quarter_two,
PARTITION i_quarter_three TABLESPACE i_quarter_three,
PARTITION i_quarter_four TABLESPACE i_quarter_four
);
```

次に、各パーティションをマージします。

```
--
-- Merge the first two partitions
--
ALTER TABLE four_seasons
MERGE PARTITIONS quarter_one, quarter_two INTO PARTITION quarter_two;
```

次に、影響を受けたパーティションのローカル索引を再作成します。

```
-- Rebuild index for quarter_two, which has been marked unusable
-- because it has not had all of the data from Q1 added to it.
-- Rebuilding the index will correct this.
--
ALTER TABLE four_seasons MODIFY PARTITION
quarter_two REBUILD UNUSABLE LOCAL INDEXES;
```

リスト・パーティションのマージ

リスト・パーティションでは、任意の 2 つのパーティションがマージできます。リスト・パーティション化ではパーティションの順序が想定されていないため、レンジ・パーティションのようにマージするパーティションが隣接している必要はありません。マージされたパーティションは、元の 2 つのパーティションに含まれていたすべてのデータから構成されます。デフォルトのリスト・パーティションを他のパーティションとマージすると、マージされたパーティションがデフォルト・パーティションとなります。

次の文は、リスト・パーティション化方法でパーティション化されている表の 2 つのパーティションを 1 つのパーティションにマージします。マージ後のパーティションは表レベル

のデフォルト属性をすべて継承しますが、この文で指定されている PCTFREE および MAXEXTENTS の各属性は継承されません。

```
ALTER TABLE q1_sales_by_region
   MERGE PARTITIONS q1_northcentral, q1_southcentral
   INTO PARTITION q1_central
      PCTFREE 50 STORAGE(MAXEXTENTS 20);
```

元の 2 つのパーティションの値リストは、次のように指定されていました。

```
PARTITION q1_northcentral VALUES ('SD','WI')
PARTITION q1_southcentral VALUES ('OK','TX')
```

マージされた sales_west パーティションの値リストは、これら 2 つのパーティションの値リストを結合したもので構成されます。具体的には次のようになります。

- ('SD','WI','OK','TX')

レンジ - ハッシュ・パーティションのマージ

レンジ - ハッシュ・パーティションをマージすると、サブパーティションは SUBPARTITIONS 句または SUBPARTITION 句に指定した数のサブパーティションに再ハッシュされます。また、どちらの句も指定されていない場合は、表レベルのデフォルトが使用されます。

1 つのレンジ - ハッシュ・パーティションを分割する場合（17-54 ページの「レンジ - ハッシュ・パーティションの分割」を参照）と 2 つのレンジ - ハッシュ・パーティションをマージする場合では、プロパティの継承が異なる点に注意してください。パーティションの分割では親が 1 つのみのため、新しいパーティションは元のパーティションのプロパティを継承できます。しかし、パーティションのマージでは親が 2 つあり、一方を犠牲にして他方からプロパティを継承することはできません。このため、パーティションには表レベルのデフォルトのプロパティが継承されます。

次の例では、2 つのレンジ - ハッシュ・パーティションをマージしています。

```
ALTER TABLE all_seasons
   MERGE PARTITIONS quarter_1, quarter_2 INTO PARTITION quarter_2
   SUBPARTITIONS 8;
```

レンジ - リスト・パーティションのマージ

パーティションをレンジ・パーティション・レベルでマージし、サブパーティションをリスト・サブパーティション・レベルでマージできます。

レンジ - リスト・パーティション表のパーティションのマージ レンジ - リスト・パーティション表のレンジ・パーティションをマージする方法は、すでに 17-38 ページの「レンジ・パーティションのマージ」で説明されています。ただし、2 つのレンジ - リスト・パーティションをマージすると、マージされた新しいパーティションは、サブパーティション・テンプレートが存在していれば、そのテンプレートからサブパーティション記述を継承します。

サブパーティション・テンプレートが存在しない場合は、新しいパーティション用にデフォルト・サブパーティションが 1 つ作成されます。

次の文は、レンジ - リスト・パーティション表 stripe_regional_sales の 2 つのパーティションをマージします。この表にはサブパーティション・テンプレートが存在します。

```
ALTER TABLE stripe_regional_sales
   MERGE PARTITIONS q1_1999, q2_1999 INTO PARTITION q1_q2_1999
      PCTFREE 50 STORAGE(MAXEXTENTS 20);
```

この新しいパーティションには新しい物理属性がいくつか指定されていますが、未指定の物理属性については表レベルのデフォルトが継承されます。マージされた新しいパーティション q1_q2_1999 は、パーティション q2_1999 の上限値と、表のサブパーティション・テンプレート記述からのサブパーティションの値リスト記述を継承します。

マージされたパーティション内のデータは、両方のパーティションからのデータで構成されます。ただし、Oracle でエラーが返される場合があります。これは、次の両方の条件が存在すると、データが新しいパーティションの外にマップされる場合があるためです。

- マージされたサブパーティションのリテラル値の一部が、サブパーティション・テンプレートに含まれていなかった場合
- サブパーティション・テンプレートにデフォルト・パーティションの定義が含まれていない場合

このエラー条件は、デフォルトのサブ・パーティション・テンプレートで常にデフォルト・パーティションを指定すると排除できます。

レンジ - リスト・パーティション表のサブパーティションのマージ 同じレンジ・パーティションに属している任意の 2 つのリスト・サブパーティションの内容をマージできます。マージされたサブパーティションの値リスト記述子には、マージ対象となったパーティションの値リストにあるすべてのリテラル値が含まれます。

次の文は、レンジ - リスト方法を使用してパーティション化された表の 2 つのサブパーティションを、表領域 ts4 にある新しいサブパーティションにマージします。

```
ALTER TABLE quarterly_regional_sales
   MERGE SUBPARTITIONS q1_1999_northwest, q1_1999_southwest
      INTO SUBPARTITION q1_1999_west
         TABLESPACE ts4;
```

元の 2 つのパーティションの値リストは、次のように指定されていました。

- サブパーティション q1_1999_northwest は、('WA','OR') として記述されていました。
- サブパーティション q1_1999_southwest は、('AZ','NM','UT') として記述されていました。

マージされたサブパーティションの値リストは、次のように、この 2 つのサブパーティションの値リストを結合したもので構成されます。

- サブパーティション q1_1999_west の値リストは、('WA','OR','AZ','NM','UT') として記述されます。

マージされたサブパーティションが格納されている表領域と、サブパーティションの属性は、明示的に指定されたものを除きパーティション・レベルのデフォルト属性により決定されます。既存のサブパーティション名のいずれかが再利用されている場合、新しいサブパーティションは名前が再利用されているサブパーティションのサブパーティション属性を継承します。

デフォルト属性の変更

表またはコンポジット・パーティション表のパーティションのデフォルト属性は変更可能です。デフォルト属性を変更すると、新しい属性はその後に作成するパーティションまたはサブパーティションにのみ反映されます。パーティションまたはサブパーティションの作成時に値を指定して、デフォルト値を上書きすることもできます。

表のデフォルト属性の変更

レンジ・パーティション、リスト・パーティションまたはハッシュ・パーティションに継承されるデフォルト属性を変更するには、ALTER TABLE の MODIFY DEFAULT ATTRIBUTES 句を使用します。次の例では、新しくパーティションを作成するために、表 emp の PCTFREE のデフォルト値を変更しています。

```
ALTER TABLE emp
      MODIFY DEFAULT ATTRIBUTES PCTFREE 25;
```

ハッシュ・パーティション表の場合、変更できる属性は TABLESPACE のみです。

パーティションのデフォルト属性の変更

サブパーティションの作成時に継承されるデフォルト属性を変更するには、ALTER TABLE ... MODIFY DEFAULT ATTRIBUTES FOR PARTITION を使用します。次の文は、レンジ - ハッシュ・パーティション表 emp 内のパーティション p1 について、今後作成するサブパーティションを格納する TABLESPACE を変更します。

```
ALTER TABLE emp
     MODIFY DEFAULT ATTRIBUTES FOR PARTITION p1 TABLESPACE ts1;
```

変更できる属性は TABLESPACE のみです。これは、レンジ - ハッシュ・パーティション表のすべてのサブパーティションは、この属性を除き、同じ属性を共有するためです。

索引パーティションのデフォルト属性の変更

表パーティションと同じ方法で、レンジ・パーティション化されたグローバル索引のパーティションに継承されるデフォルト属性を変更できます。同様に、パーティション表のローカル索引パーティションに継承されるデフォルト属性も変更できます。この場合は、ALTER INDEX ... MODIFY DEFAULT ATTRIBUTES 文を使用します。コンポジット・パーティション表のサブパーティションに継承されるデフォルト属性を変更する場合は、ALTER INDEX ... MODIFY DEFAULT ATTRIBUTES FOR PARTITION 文を使用します。

パーティションの実属性の変更

表または索引の既存パーティションの属性は変更可能です。

TABLESPACE 属性は変更できません。パーティションまたはサブパーティションを新しい表領域に移動するには、ALTER TABLESPACE ... MOVE PARTITION/SUBPARTITION を使用します。

レンジ・パーティションまたはリスト・パーティションの実属性の変更

レンジ・パーティションまたはリスト・パーティションの既存の属性を変更するには、ALTER TABLE ... MODIFY PARTITION 文を使用します。この文では、セグメント属性（TABLESPACE を除く）の変更、エクステントの割当てと割当て解除、ローカル索引パーティションへの UNUSABLE マークの設定、UNUSABLE マークが付いたローカル索引の再作成などが可能です。

これがレンジ - ハッシュ・パーティション表のレンジ・パーティションの場合は、次のことに注意してください。

- エクステントを割り当てるか、割当てを解除する場合、このアクションは指定したパーティションのすべてのサブパーティションに対して実行されます。
- 同様に、他の属性を変更すると、そのパーティションのすべてのサブパーティションで、対応する属性が変更されます。また、パーティション・レベルのデフォルト属性も変更されます。既存のサブパーティションの属性が変更されないようにするには、MODIFY DEFAULT ATTRIBUTES 文の FOR PARTITION 句を使用します。

次に、パーティションの実属性を変更する操作の例をいくつか示します。

次の例では、表 sales のレンジ・パーティション sales_q1 の MAXEXTENTS 記憶域属性を変更しています。

```
ALTER TABLE sales MODIFY PARTITION sales_Q1
     STORAGE (MAXEXTENTS 10);
```

次の例では、レンジ - ハッシュ・パーティション表 scubagear 内にあるパーティション ts1 のすべてのローカル索引サブパーティションに、UNUSABLE マークが付けられます。

```
ALTER TABLE scubagear MPDIFY PARTITION ts1 UNUSABLE LOCAL INDEXES;
```

ハッシュ・パーティションの実属性の変更

ハッシュ・パーティションの属性も、ALTER TABLE ... MODIFY PARTITION 文を使用して変更できます。ただし、個々のハッシュ・パーティションの物理属性は必ずすべて同じ（TABLESPACE を除く）であるため、変更できるのは次の属性に限定されます。

- 新規エクステントの割当て
- 未使用エクステントの割当て解除
- ローカル索引サブパーティションへの UNUSABLE マークの設定
- UNUSABLE マークが付いたローカル索引サブパーティションの再作成

次の例では、表 dept のハッシュ・パーティション p1 に関連付けられている、使用禁止状態のローカル索引パーティションを再作成しています。

```
ALTER TABLE dept MODIFY PARTITION p1
     REBUILD UNUSABLE LOCAL INDEXES;
```

サブパーティションの実属性の変更

ALTER TABLE の MODIFY SUBPARTITION 句を使用すると、前述したパーティションに対する操作と同じことを、特定のコンポジット・パーティション表のサブパーティション・レベルで実行できます。次に例を示します。

```
ALTER TABLE emp MODIFY SUBPARTITION p3_s1
     REBUILD UNUSABLE LOCAL INDEXES;
```

索引パーティションの実属性の変更

ALTER INDEX の MODIFY PARTITION 句を使用すると、索引パーティションまたはそのサブパーティションの実属性を変更できます。適用されるルールは表パーティションの場合とほぼ同じですが、ALTER TABLE の MODIFY PARTITION 句とは異なり、使用禁止状態の索引パーティションを再作成するための副次句はありません。ただし、索引パーティションまたはそのサブパーティションを結合するための副次句はあります。この場合、結合するとは、可能であれば索引ブロックをマージし、使用されなくなった索引ブロックを再利用のために解放することを意味します。

また、MODIFY SUBPARTITION 句を使用して、ローカル索引のサブパーティションに対する記憶域の割当てや割当て解除、UNUSABLE マークの設定ができます。

リスト・パーティションの変更 : 値の追加

リスト・パーティション化では、定義されている値リストに対してリテラル値を追加できます。

リスト・パーティションに対する値の追加

既存パーティションの値リストを拡張するには、ALTER TABLE 文の MODIFY PARTITION ... ADD VALUES 句を使用します。他のパーティションの値リストに含まれているリテラル値は追加できません。対応するローカル索引パーティションのパーティション値リストも、それに応じて拡張されます。グローバル索引、グローバル索引パーティションまたはローカル索引パーティションは使用可能のままです。

次の文は、既存のパーティション・リストに状態コードの新しいセット（'OK'、'KS'）を追加します。

```
ALTER TABLE sales_by_region
   MODIFY PARTITION region_south
      ADD VALUES ('OK', 'KS');
```

デフォルト・パーティションがあると、他のパーティションに値を追加するときのパフォーマンスに影響する可能性があります。これは、リスト・パーティションに値を追加するために、Oracle は追加する値がデフォルト・パーティションに存在しないかどうかをチェックする必要があるためです。値のいずれかがデフォルト・パーティションに存在する場合は、エラーになります。

注意： 追加するリテラル値に対応するデフォルト・パーティション内に行が存在するかどうかをチェックする問合せが実行されるので、対象の表にローカルな同一キー索引を作成することをお薦めします。これにより、問合せと操作全体の実行を高速化できます。

デフォルトのリスト・パーティションには値を追加できません。

リスト・サブパーティションに対する値の追加

この操作は「リスト・パーティションの変更 : 値の追加」で説明した操作と同じですが、MODIFY PARTITION 句のかわりに MODIFY SUBPARTITION 句を使用します。たとえば、サブパーティション q1_1999_southeast の値リスト内でリテラル値の範囲を拡張するには、次の文を使用します。

```
ALTER TABLE quarterly_regional_sales
   MODIFY SUBPARTITION q1_1999_southeast
      ADD VALUES ('KS');
```

所有パーティションの他のサブパーティションの値リストに含まれているリテラル値は追加できません。ただし、表の他のパーティションのサブパーティションの値リストにあるリテラル値と重複していてもかまいません。

リスト・パーティションの変更 : 値の削除

リスト・パーティション化では、定義されている値リストからリテラル値を削除できます。

リスト・パーティションからの値の削除

既存パーティションの値リストからリテラル値を削除するには、ALTER TABLE 文の MODIFY PARTITION ... DROP VALUES 句を使用します。この文を実行したときは必ずデータが検証されます。つまり、この文は、削除する値セットに対応するパーティション内に行が存在するかどうかをチェックします。行が存在するとエラー・メッセージが返され、文の実行は失敗します。必要に応じて、値を削除する前に DELETE 文を実行して、対応する行を削除してください。

注意 : この方法では、パーティションを記述している値リストからすべてのリテラル値を削除することはできません。このためには、ALTER TABLE ... DROP PARTITION 文を使用します。

対応するローカル索引パーティションのパーティション値リストには、新しい値リストが反映されます。グローバル索引、グローバル索引パーティションまたはローカル索引パーティションは使用可能のままです。

次の例では、既存のパーティションの値リストから状態コードのセット（'OK' および 'KS'）を削除しています。

```
ALTER TABLE sales_by_region
   MODIFY PARTITION region_south
      DROP VALUES ('OK', 'KS');
```

注意 : 削除するリテラル値に対応するパーティション内に行が存在するかどうかをチェックする問合せが実行されるので、対象の表にローカルな同一キー索引を作成することをお薦めします。これにより、問合せと操作全体の実行を高速化できます。

デフォルトのリスト・パーティションからは値を削除できません。

リスト・サブパーティションからの値の削除

この操作は「リスト・パーティションの変更 : 値の削除」で説明した操作と同じですが、MODIFY PARTITION 句のかわりに MODIFY SUBPARTITION 句を使用します。たとえば、サブパーティション q1_1999_southeast の値リストからリテラル値のセットを削除するには、次の文を使用します。

```
ALTER TABLE quarterly_regional_sales
   MODIFY SUBPARTITION q1_1999_southeast
      DROP VALUES ('KS');
```

サブパーティション・テンプレートの変更

コンポジット・パーティション表のサブパーティション・テンプレートを変更するには、新規のサブパーティション・テンプレートで置換します。サブパーティション・テンプレートを使用する以降の操作（ADD PARTITION または MERGE PARTITIONS など）では、新しいサブパーティション・テンプレートが使用されます。既存のサブパーティションが変更されることはありません。

ALTER TABLE ... SET SUBPARTITION TEMPLATE 文を使用して、新しいサブパーティション・テンプレートを指定します。次に例を示します。

```
ALTER TABLE emp_sub_template
   SET SUBPARTITION TEMPLATE
         (SUBPARTITION e, TABLESPACE ts1,
          SUBPARTITION f, TABLESPACE ts2,
          SUBPARTITION g, TABLESPACE ts3,
          SUBPARTITION h, TABLESPACE ts4
         );
```

空のリストを指定すると、サブパーティション・テンプレートを削除できます。

```
ALTER TABLE emp_sub_template
   SET SUBPARTITION TEMPLATE ( );
```

パーティションの移動

ALTER TABLE 文の MOVE PARTITION 句を使用すると、次のことができます。

- データの再クラスタ化による断片化の低減
- 別の表領域へのパーティションの移動
- 作成時間属性の変更

一般に、ALTER TABLE/INDEX ... MODIFY PARTITION 文を使用すると、パーティションの物理的な記憶域属性を 1 ステップで変更できます。しかし、TABLESPACE のように、MODIFY PARTITION では変更できない物理属性もあります。このような場合は、MOVE PARTITION 句を使用します。

移動するパーティションにデータが含まれている場合は、次の表のルールに従って、索引に UNUSABLE マークが付けられます。

表のタイプ	索引の動作
通常の表（ヒープ）	■ 各ローカル索引内の対応するパーティションには、UNUSABLE マークが付けられます。したがって、MOVE PARTITION を発行した後で、それらの索引パーティションを再作成する必要があります。 ■ UPDATE GLOBAL INDEXES を指定しないかぎり、すべてのグローバル索引、またはパーティション化されたグローバル索引のすべてのパーティションには UNUSABLE マークが付けられます。
索引構成表	移動するパーティションに対して定義されているローカル索引またはグローバル索引は主キー・ベースの論理 ROWID であるため、使用可能のままです。しかし、これらの ROWID に対する推測情報は不適切になります。

表パーティションの移動

パーティションを移動するには、MOVE PARTITION 句を使用します。たとえば、I/O のバランスを調整するために、最もアクティブなパーティションを専用ディスク上にある表領域に移動し、そのアクションをログに記録しない場合は、次の文を発行します。

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

この文は、新しい表領域を指定しなくても、常にパーティションの旧セグメントを削除し、新しいセグメントを作成します。

サブパーティションの移動

次の文は、表のサブパーティション内にあるデータを移動する方法を示しています。この例では、PARALLEL 句も指定されています。

```
ALTER TABLE scuba_gear MOVE SUBPARTITION bcd_types
     TABLESPACE tbs23 PARALLEL (DEGREE 2);
```

索引のパーティションの移動

通常の表に対して ALTER TABLE ... MOVE PARTITION 文を実行すると、グローバル索引のすべてのパーティションに UNUSABLE マークが付けられます。この場合は、ALTER INDEX ... REBUILD PARTITION 文を使用して各パーティションを個別に再作成することにより、索引全体を再作成できます。このような索引の再作成は、同時に実行できます。

また、単に索引を削除して再作成するという方法もあります。

索引パーティションの再作成

索引パーティションを再作成するのは、次のような場合です。

- 領域をリカバリしてパフォーマンスを改善する場合
- メディア障害のために破損した索引パーティションを修復する場合
- インポート・ユーティリティまたは SQL*Loader で基礎となる表パーティションをロードした後に、ローカル索引パーティションを再作成する場合
- UNUSABLE マークが付いている索引パーティションを再作成する場合

ここでは、索引パーティションおよびサブパーティションを再作成する場合のオプションについて説明します。

グローバル索引パーティションの再作成

グローバル索引のパーティションを再作成するには、次の 2 つの方法があります。

1. ALTER INDEX ... REBUILD PARTITION 文を発行することによって、各パーティションを再作成する（再作成は同時実行可能）。
2. 一度グローバル索引全体を削除し、再作成する。

> **注意：** 第 2 の方法では表が 1 回しかスキャンされないため、最初の方法より効率的です。

グローバル索引付きパーティション表に対するほとんどのメンテナンス操作では、DDL 文に UPDATE GLOBAL INDEXES を指定することによって、グローバル索引の再作成が不要になります。

ローカル索引パーティションの再作成

ローカル索引を再作成するには、次の ALTER INDEX または ALTER TABLE を使用します。

- ALTER INDEX ... REBUILD PARTITION/SUBPARTITION

  この文は、索引のパーティションまたはサブパーティションを無条件で再作成します。

- ALTER TABLE ... MODIFY PARTITION/SUBPARTITION ... REBUILD UNUSABLE LOCAL INDEXES

  この文は指定された表のパーティションまたはサブパーティションで使用禁止状態の索引をすべて検索し、それらを再作成します。索引パーティションは、UNUSABLE マークが付いている場合にのみ再作成されます。

ALTER INDEX を使用してパーティションを再作成する場合 ALTER INDEX ... REBUILD PARTITION 文は、1 つの索引の 1 つのパーティションを再作成します。レンジ - ハッシュ・パーティション表には使用できません。索引を再作成するときは、パーティションを新しい表領域へ移動したり、属性を変更できます。

レンジ - ハッシュ・パーティション表の場合は、ALTER INDEX ... REBUILD SUBPARTITION を使用して、索引のサブパーティションを再作成します。サブパーティションを別の表領域に移動したり、PARALLEL 句を指定できます。次の文は、表のローカル索引のサブパーティションを再作成し、索引サブパーティションを別の表領域に移動します。

```
ALTER INDEX scuba
   REBUILD SUBPARTITION bcd_types
   TABLESPACE tbs23 PARALLEL (DEGREE 2);
```

ALTER TABLE を使用して索引パーティションを再作成する場合 ALTER TABLE ... MODIFY PARTITION の REBUILD UNUSABLE LOCAL INDEXES 句では、再作成する索引パーティションの新しい属性は指定できません。次の例では、表 scubagear について、使用禁止状態のローカル索引パーティションであるパーティション p1 を検索し、再作成しています。

```
ALTER TABLE scubagear
   MODIFY PARTITION p1 REBUILD UNUSABLE LOCAL INDEXES;
```

使用禁止状態のローカル索引サブパーティションを再作成するために、同様の機能を持つ ALTER TABLE ... MODIFY SUBPARTITION 句があります。

パーティションの名前変更

表と索引のパーティションとサブパーティションは名前変更できます。パーティションの名前を変更する理由の 1 つは、別のメンテナンス操作でパーティションに割り当てられたデフォルトのシステム名のかわりに、意味のある名前を割り当てることです。

表パーティションの名前変更

レンジ・パーティション、ハッシュ・パーティションまたはリスト・パーティションの名前を変更するには、ALTER TABLE ... RENAME PARTITION 文を使用します。次に例を示します。

```
ALTER TABLE scubagear RENAME PARTITION sys_p636 TO tanks;
```

表のサブパーティションの名前変更

同様に、表のサブパーティションにも新しい名前を割り当てることができます。この場合は、ALTER TABLE ... RENAME SUBPARTITION 構文を使用します。

索引パーティションの名前変更

索引パーティションおよびサブパーティションも同様に名前変更できますが、ALTER INDEX 構文を使用します。

索引パーティションの名前変更 索引パーティションの名前を変更するには、ALTER INDEX ... RENAME PARTITION 文を使用します。

索引のサブパーティションの名前変更 次の文は、基礎となる表にパーティションを追加した後で、システム生成名を持つサブパーティションの名前を変更する方法を示しています。

```
ALTER INDEX scuba RENAME SUBPARTITION sys_subp3254 TO bcd_types;
```

パーティションの分割

パーティションの内容を 2 つの新しいパーティションに再分散させるには、ALTER TABLE 文または ALTER INDEX 文の SPLIT PARTITION 句を使用します。パーティションのサイズが大きくなり、バックアップ、リカバリまたはメンテナンス操作に時間がかかる場合に、この方法を検討してください。また、SPLIT PARTITION 句を使用して I/O ロードを再分散させることもできます。

この句は、ハッシュ・パーティションまたはサブパーティションには使用できません。

分割するパーティションにデータが含まれている場合は、次の表に示すように、索引に UNUSABLE マークが付けられます。

表のタイプ	索引の動作
通常の表（ヒープ）	■ 各ローカル索引内にある 2 つの新しいパーティションには、UNUSABLE マークが付けられます。 ■ UPDATE GLOBAL INDEXES を指定しないかぎり、すべてのグローバル索引、またはパーティション化されたグローバル索引のすべてのパーティションには UNUSABLE マークが付けられるため、それらを再作成する必要があります。
索引構成表	■ 各ローカル索引内にある 2 つの新しいパーティションには、UNUSABLE マークが付けられます。 ■ グローバル索引はすべて使用可能のままです。

レンジ・パーティション表のパーティションの分割

レンジ・パーティションを分割するには、ALTER TABLE ... SPLIT PARTITION 文を使用します。分割するパーティションの範囲内でパーティション化キー列の値を指定します。分割後の 2 つの新しいパーティションの一方には、元のパーティション内でパーティション化キー列の値が指定した値より下位にマップされる行がすべて含まれます。他方のパーティションには、パーティション化キー列の値が指定した値と同等または上位にマップされる行がすべて含まれます。

必要であれば、分割後の 2 つのパーティションに新しい属性を指定できます。表にローカル索引が定義されている場合は、この文によって、各ローカル索引内の対応するパーティションも分割されます。

次の例では、表 vet_cats に fee_katy というパーティションがあります。この表には、jaf1 というローカル索引があります。この表には、vet というグローバル索引もあります。vet には、vet_parta と vet_partb という 2 つのパーティションがあります。

パーティション fee_katy を分割し、索引のパーティションを再作成するには、次の文を発行します。

```
ALTER TABLE vet_cats SPLIT PARTITION
      fee_katy at (100) INTO ( PARTITION
      fee_katy1 ..., PARTITION fee_katy2 ...);
ALTER INDEX JAF1 REBUILD PARTITION fee_katy1;
ALTER INDEX JAF1 REBUILD PARTITION fee_katy2;
ALTER INDEX VET REBUILD PARTITION vet_parta;
ALTER INDEX VET REBUILD PARTITION vet_partb;
```

注意： 新しいパーティション名を指定しない場合は、SYS_P*n* という形式の名前が割り当てられます。データ・ディクショナリを調べると、新しいローカル索引のパーティションに割り当てられた名前を確認できます。必要に応じて、それらの名前を変更できます。指定しなかった属性は、元のパーティションから継承されます。

リスト・パーティション表のパーティションの分割

リスト・パーティションを分割するには、ALTER TABLE ... SPLIT PARTITION 文を使用します。SPLIT PARTITION 句によって、リテラル値の値リストを指定できます。このリストの値に対応するパーティション化キー値を持つ行が、定義されたパーティションに挿入されます。元のパーティションの残りの行は、2 番目のパーティションに挿入されます。値リスト全体が、元のパーティションからの残りの値となります。

必要であれば、分割後の 2 つのパーティションに新しい属性を指定できます。

次の文は、region_east パーティションを 2 つのパーティションに分割します。

```
ALTER TABLE sales_by_region
   SPLIT PARTITION region_east VALUES ('CT', 'VA', 'MD')
   INTO
    ( PARTITION region_east_1
         PCTFREE 25 TABLESPACE tbs2,
      PARTITION region_east_2
        STORAGE (NEXT 2M PCTINCREASE 25))
   PARALLEL 5;
```

元の region_east パーティションに対するリテラル値リストは、次のように指定されていました。

```
PARTITION region_east VALUES ('MA','NY','CT','NH','ME','MD','VA','PA','NJ')
```

2 つの新しいパーティションのリテラル値リストは、次のようになります。

- region_east_1 のリテラル値リストは、('CT','VA','MD') です。
- region_east_2 には、残りのリテラル値のリスト ('NY', 'NH', 'ME', 'VA', 'PA', 'NJ') が継承されます。

個々のパーティションは、パーティション・レベルで指定された新しい物理属性を持ちます。この操作は、並列度 5 で実行されます。

他のリスト・パーティションと同様に、デフォルトのリスト・パーティションを分割できます。デフォルト・パーティションを含むリスト・パーティション表にパーティションを追加する場合も、この操作が必要です。デフォルト・パーティションを分割すると、指定した値で定義される新しいパーティションが作成され、2 番目のパーティションが引き続きデフォルト・パーティションとなります。

次の例では、sales_by_region のデフォルト・パーティションを分割して新しいパーティションを作成しています。

```
ALTER TABLE sales_by_region
   SPLIT PARTITION region_unknown VALUES ('MT', 'WY', 'ID')
   INTO
    ( PARTITION region_wildwest,
      PARTITION region_unknown);
```

レンジ - ハッシュ・パーティションの分割

レンジ - ハッシュ・パーティションをマージする操作とは逆の操作です。レンジ - ハッシュ・パーティションを分割すると、新しいサブパーティションは、SUBPARTITIONS 句または SUBPARTITION 句のいずれかで指定される数のサブパーティションに再ハッシュされます。これらの句が指定されていない場合は、分割されるパーティションから新しいパーティションにサブパーティションの数（および表領域）が継承されます。

1 つのレンジ - ハッシュ・パーティションを分割する場合と 2 つのレンジ - ハッシュ・パーティションをマージする場合では、プロパティの継承が異なる点に注意してください。パーティションの分割では親が 1 つのみのため、新しいパーティションは元のパーティションのプロパティを継承できます。しかし、パーティションのマージでは親が 2 つあり、一方を犠牲にして他方からプロパティを継承することはできません。このため、パーティションには表レベルのデフォルトのプロパティが継承されます。

次の例では、1 つのレンジ - ハッシュ・パーティションを分割しています。

```
ALTER TABLE all_seasons SPLIT PARTITION quarter_1
     AT (TO_DATE('16-dec-1997','dd-mon-yyyy'))
     INTO (PARTITION q1_1997_1 SUBPARTITIONS 4 STORE IN (ts1,ts3),
           PARTITION q1_1997_2);
```

レンジ - リスト・パーティション表のパーティションの分割

パーティションは、レンジ・パーティション・レベルとリスト・サブパーティション・レベルのどちらでも分割できます。

レンジ - リスト・パーティションの分割 レンジ - リスト・パーティション表のレンジ・パーティションを分割する操作は、17-52 ページの「レンジ・パーティション表のパーティションの分割」で説明した操作と同様です。新しいパーティションについては、サブパーティションのリテラル値のリストを指定できません。新しいパーティションは、分割元のパーティションからサブパーティション記述を継承します。

次の例では、quarterly_regional_sales 表の q1_1999 パーティションを分割しています。

```
ALTER TABLE quarterly_regional_sales SPLIT PARTITION q1_1999
   AT (to_date('15-Feb-1999','dd-mon-yyyy'))
   INTO ( PARTITION q1_1999_jan_feb
             PCTFREE 25 TABLESPACE ts1,
          PARTITION q1_1999_feb_mar
             STORAGE (NEXT 2M PCTINCREASE 25) TABLESPACE ts2)
   PARALLEL 5;
```

この操作により、パーティション q1_1999 が 2 つのパーティション q1_1999_jan_feb および q1_1999_feb_mar に分割されます。どちらのパーティションも、元のパーティションからサブパーティション記述を継承します。個々のパーティションは、表領域など、パーティション・レベルで指定された新しい物理属性を持ちます。この新しい属性は、新しいパーティションのデフォルト属性となります。この操作は、並列度 5 で実行されます。

ALTER TABLE ... SPLIT PARTITION 文では、コンポジット・パーティション表のパーティションを分割して作成するサブパーティションの名前を指定できません。ただし、"*partition name_subpartition name*" 形式の名前を持つ親パーティション内のサブパーティションの場合は、Oracle により新しく作成されたサブパーティション内で新しいパーティション名を使用して対応する名前が生成されます。他のすべてのサブパーティションには、SYS_SUBP*n* 形式のシステム生成名が割り当てられます。システム生成名は、名前を指定しないで分割して作成したパーティションのサブパーティションにも割り当てられます。名前のないパーティションには、SYS_P*n* 形式のシステム生成名が割り当てられます。

次の問合せでは、表 quarterly_regional_sales に対する前述の分割操作で得られたサブパーティション名が表示されます。この表を 17-8 ページの「レンジ - リスト・コンポジット・パーティション化方法を使用する場合」で作成した後、各項で実行した他の操作の結果も反映されます。

```
SQL> SELECT PARTITION_NAME, SUBPARTITION_NAME, TABLESPACE_NAME
  2       FROM DBA_TAB_SUBPARTITIONS WHERE TABLE_NAME='QUARTERLY_REGIONAL_SALES'
  3       ORDER BY PARTITION_NAME;

PARTITION_NAME       SUBPARTITION_NAME              TABLESPACE_NAME
-------------------- ------------------------------ ---------------
Q1_1999_FEB_MAR      Q1_1999_FEB_MAR_WEST           TS2
Q1_1999_FEB_MAR      Q1_1999_FEB_MAR_NORTHEAST      TS2
Q1_1999_FEB_MAR      Q1_1999_FEB_MAR_SOUTHEAST      TS2
Q1_1999_FEB_MAR      Q1_1999_FEB_MAR_NORTHCENTRAL   TS2
Q1_1999_FEB_MAR      Q1_1999_FEB_MAR_SOUTHCENTRAL   TS2
Q1_1999_FEB_MAR      Q1_1999_FEB_MAR_SOUTH          TS2
Q1_1999_JAN_FEB      Q1_1999_JAN_FEB_WEST           TS1
Q1_1999_JAN_FEB      Q1_1999_JAN_FEB_NORTHEAST      TS1
Q1_1999_JAN_FEB      Q1_1999_JAN_FEB_SOUTHEAST      TS1
Q1_1999_JAN_FEB      Q1_1999_JAN_FEB_NORTHCENTRAL   TS1
Q1_1999_JAN_FEB      Q1_1999_JAN_FEB_SOUTHCENTRAL   TS1
Q1_1999_JAN_FEB      Q1_1999_JAN_FEB_SOUTH          TS1
Q1_2000              Q1_2000_NORTHWEST              TS3
Q1_2000              Q1_2000_SOUTHWEST              TS3
Q1_2000              Q1_2000_NORTHEAST              TS3
Q1_2000              Q1_2000_SOUTHEAST              TS3
Q1_2000              Q1_2000_NORTHCENTRAL           TS3
Q1_2000              Q1_2000_SOUTHCENTRAL           TS3
Q2_1999              Q2_1999_NORTHWEST              TS4
Q2_1999              Q2_1999_SOUTHWEST              TS4
Q2_1999              Q2_1999_NORTHEAST              TS4
```

```
Q2_1999              Q2_1999_SOUTHEAST             TS4
Q2_1999              Q2_1999_NORTHCENTRAL          TS4
Q2_1999              Q2_1999_SOUTHCENTRAL          TS4
Q3_1999              Q3_1999_NORTHWEST             TS4
Q3_1999              Q3_1999_SOUTHWEST             TS4
Q3_1999              Q3_1999_NORTHEAST             TS4
Q3_1999              Q3_1999_SOUTHEAST             TS4
Q3_1999              Q3_1999_NORTHCENTRAL          TS4
Q3_1999              Q3_1999_SOUTHCENTRAL          TS4
Q4_1999              Q4_1999_NORTHWEST             TS4
Q4_1999              Q4_1999_SOUTHWEST             TS4
Q4_1999              Q4_1999_NORTHEAST             TS4
Q4_1999              Q4_1999_SOUTHEAST             TS4
Q4_1999              Q4_1999_NORTHCENTRAL          TS4
Q4_1999              Q4_1999_SOUTHCENTRAL          TS4

36 rows selected.
```

レンジ - リスト・サブパーティションの分割 レンジ - リスト・パーティション表のリスト・サブパーティションを分割する操作は、17-53 ページの「リスト・パーティション表のパーティションの分割」で説明した操作と同様ですが、PARTITION ではなく SUBPARTITION の構文を使用します。たとえば、次の文は quarterly_regional_sales 表のサブパーティションを分割します。

```
ALTER TABLE quarterly_regional_sales SPLIT SUBPARTITION q2_1999_southwest
   VALUES ('UT') INTO
      ( SUBPARTITION q2_1999_utah
          TABLESPACE ts2,
        SUBPARTITION q2_1999_southwest
          TABLESPACE ts3
      )
   PARALLEL;
```

この操作では、サブパーティション q2_1999_southwest が次の 2 つのサブパーティションに分割されます。

- リテラル値のリスト ('UT') を持つ q2_1999_utah
- 残りのリテラル値のリスト ('AZ','NM') を持つ q2_1999_southwest

個々のサブパーティションは、分割対象となったサブパーティションから継承した新しい物理属性を持ちます。

索引パーティションの分割

ローカル索引のパーティションは、明示的に分割できません。ローカル索引のパーティションを分割できるのは、その基礎となる表パーティションを分割するときのみです。ただし、グローバル索引のパーティションは、次のような方法で分割できます。

```
ALTER INDEX quon1 SPLIT
    PARTITION canada AT ( 100 ) INTO
    PARTITION canada1 ..., PARTITION canada2 ...);
ALTER INDEX quon1 REBUILD PARTITION canada1;
ALTER INDEX quon1 REBUILD PARTITION canada2;
```

分割する索引には索引データが含まれていてもかまいません。また、元のパーティションで事前に UNUSABLE マークが付けられていないかぎり、分割されたパーティションで再作成する必要はありません。

SPLIT PARTITION および SPLIT SUBPARTITION 操作の最適化

Oracle では、2 つの新しいパーティションを作成し、分割対象となったパーティションの行を 2 つの新しいパーティションに再分散することで、SPLIT PARTITION 操作が実装されます。この操作は、分割対象となったパーティションのすべての行をスキャンして、新しいパーティションに 1 行ずつ挿入する必要があるため高コストです。新しいパーティションに対応するローカル索引パーティションも、再作成する必要があります。また、UPDATE GLOBAL INDEXES 句を使用しない場合は、グローバル索引の再作成も必要になります。

分割操作後に、新しいパーティションの 1 つに分割対象となったパーティションのすべての行が含まれ、他のパーティションに行がまったく含まれない場合があります。通常、これは表の最初のパーティションを分割する場合です。Oracle では、このような状況を検出して分割操作を最適化できます。この最適化により、分割操作がパーティション追加操作と同様に動作し、高速化されます。

特に、Oracle では次の 2 つの条件が満たされている場合に、SPLIT PARTITION 操作を最適化し高速化できます。

- 分割後の 2 つのパーティションの一方が空になる必要があります。
- 分割後に空にならないパーティションの場合は、その記憶特性が分割対象となったパーティションと同じである必要があります。具体的には、次のようになります。
  - コンポジット・パーティションを分割する場合、分割後に空にならない新しいパーティションの各サブパーティションの記憶特性は、分割対象となったパーティションのサブパーティションと同じである必要があります。
  - 分割対象となったパーティションに LOB 列が含まれている場合、分割後に空にならない新しいパーティションの各 LOB（サブ）パーティションの記憶特性は、分割対象となったパーティションの LOB（サブ）パーティションと同じである必要があります。

分割後にこの 2 つの条件が満たされていれば、UPDATE GLOBAL INDEXES 句を指定しなかった場合にも、すべてのグローバル索引は引き続き使用可能です。分割後の両方のパーティションに関連するローカル索引（サブ）パーティションは、分割前に使用可能だった場合は引き続き使用可能です。分割後の空でないパーティションに対応するローカル索引（サブ）パーティションは、分割元パーティションのローカル索引（サブ）パーティションと同じになります。

SPLIT SUBPARTITION 操作の場合も、同じ最適化が保持されます。

パーティションの切捨て

表パーティションからすべての行を切り捨てるには、ALTER TABLE ... TRUNCATE PARTITION 文を使用します。パーティションの切捨てはパーティションの削除に似ていますが、パーティションが物理的に削除されるのではなく、そのデータが空になります。

索引パーティションの切捨てはできません。ただし、表にローカル索引が定義されている場合は、ALTER TABLE TRUNCATE PARTITION を実行することで、各ローカル索引から対応するパーティションを切り捨てることができます。UPDATE GLOBAL INDEXES を指定しないかぎり（索引構成表には指定できません）、すべてのグローバル索引、またはパーティション化されたグローバル索引のすべてのパーティションには UNUSABLE マークが付けられるため、それらを再作成する必要があります。

表パーティションの切捨て

表パーティションからすべての行を切り捨てるには、ALTER TABLE ... TRUNCATE PARTITION 文を使用します。領域を再生することも、再生しないことも可能です。

データおよびグローバル索引を含む表のパーティションの切捨て パーティションにデータとグローバル索引が含まれる場合、表パーティションを切り捨てるには次のいずれかの方法を使用します。

方法 1:

グローバル索引の更新を指定せずに、ALTER TABLE TRUNCATE PARTITION 文を実行します。この例では、表 sales にグローバル索引 sales_area_ix があり、それを再作成しています。

```
ALTER TABLE sales TRUNCATE PARTITION dec98;
ALTER INDEX sales_area_ix REBUILD;
```

切り捨てるパーティションに、その表の全データの大部分が含まれるような大規模な表の場合には、この方法が最適です。

方法 2:

ALTER TABLE ... TRUNCATE PARTITION 文を発行する前に DELETE 文を発行し、パーティションからすべての行を削除します。DELETE 文でグローバル索引が更新され、さらにトリガーが起動されて、REDO ログおよび UNDO ログが生成されます。

たとえば、パーティション・バウンド 10,000 の最初のパーティションを切り捨てる場合は、次の文を発行します。

```
DELETE FROM sales WHERE TRANSID < 10000;
ALTER TABLE sales TRUNCATE PARTITION dec98;
```

これは、小さい表の場合、または切り捨てるパーティションにその表の全データのごく一部分のみが含まれるような大規模な表の場合に最適な方法です。

方法 3:

ALTER TABLE 文で UPDATE GLOBAL INDEXES を指定します。これにより、パーティションの切捨て時にグローバル索引も切り捨てられるようになります。

```
ALTER TABLE sales TRUNCATE PARTITION dec98
     UPDATE GLOBAL INDEXES;
```

データおよび参照整合性制約を含むパーティションの切捨て パーティションにデータおよび参照整合性制約が含まれる場合、表パーティションを切り捨てるには次のいずれかの方法を使用します。

方法 1:

整合性制約を使用禁止にし、ALTER TABLE ... TRUNCATE PARTITION 文を発行してから、整合性制約を再度使用可能にします。

```
ALTER TABLE sales
    DISABLE CONSTRAINT dname_sales1;
ALTER TABLE sales TRUNCATE PARTITTION dec94;
ALTER TABLE sales
    ENABLE CONSTRAINT dname_sales1;
```

切り捨てるパーティションに、その表の全データの大部分が含まれるような大規模な表の場合には、この方法が最適です。

方法 2:

ALTER TABLE ... TRUNCATE PARTITION 文を発行する前に DELETE 文を発行し、パーティションからすべての行を削除します。DELETE 文によって参照整合性制約が適用され、さらにトリガーが起動されて、REDO ログおよび UNDO ログが生成されます。

注意： パーティションの行をすべて削除する前にパーティションの NOLOGGING 属性を設定する（ALTER TABLE ... MODIFY PARTITION ... NOLOGGING）ことにより、ロギングの量を大幅に削減できます。

```
DELETE FROM sales WHERE TRANSID < 10000;
ALTER TABLE sales TRUNCATE PARTITION dec94;
```

これは、小さい表の場合、または切り捨てるパーティションにその表の全データのごく一部分のみが含まれるような大規模な表の場合に最適な方法です。

サブパーティションの切捨て

コンポジット・パーティション表のサブパーティションからすべての行を切り捨てるには、ALTER TABLE ... TRUNCATE SUBPARTITION 文を使用します。対応するローカル索引のサブパーティションも切り捨てられます。

次の文は、表のサブパーティション内にあるデータを切り捨てる方法を示しています。この例で、削除される行が占めていた領域は、表領域内の他のスキーマ・オブジェクトで使用可能になります。

```
ALTER TABLE diving
   TRUNCATE SUBPARTITION us_locations
      DROP STORAGE;
```

パーティション表および索引の例

ここでは、パーティション表および索引の使用方法について、いくつかの例を示します。

履歴表での時間枠の移動

履歴表とは、ある期間にわたる企業の業務上の取引を記録したものです。履歴表は売上、小切手、注文などの基礎情報を含んでおり、**実**表になります。履歴表は、GROUP BY、AVERAGE または COUNT などの操作によって基礎情報から導出されるサマリー情報を取り込んだ**まとめ**表にもなります。

多くの場合、履歴表の時間間隔は、ローリング・ウィンドウのようなものです。データベース管理者（DBA）は、最も古いトランザクションを記録する一連の行を定期的に削除し、最近のトランザクションを記録する一連の行に領域を割り当てます。たとえば、DBA は、1995 年 4 月 30 日の業務終了時に、1994 年 4 月のトランザクションの行（およびそれをサポートする索引項目）を削除し、1995 年 4 月のトランザクションのために領域を割り当てます。

ここで、具体的な例について考えてみます。今月分の注文と 1 年分の履歴データをまとめた 13 か月分のトランザクションを含む表 order があるとします。この表には、月ごとにパーティションが 1 つあります。これら月ごとのパーティションには order_*yymm* という名前が付けられており、それらが格納されている表領域も同じ名前を持ちます。

order 表には 2 つのローカル索引が含まれています。1 つは order_ix_onum で、これは注文番号に対するローカルな一意の同一キー索引です。もう 1 つは order_ix_supp で、これは業者番号に対するローカルな非同一キー索引です。ローカル索引のパーティション名には、基礎となる表と一致する接尾辞が付いています。また、顧客名を表すグローバルな一意索引、order_ix_cust もあります。order_ix_cust には、アルファベット 3 文字ごとに 1 つずつ、3 つのパーティションが含まれています。このようなデータベースにおいて、1994 年 10 月 31 日に order の時間枠を変更するには、次の手順を実行します。

1. 最も古い時間間隔のデータのバックアップを作成します。

```
ALTER TABLESPACE order_9310 BEGIN BACKUP;
...
ALTER TABLESPACE order_9310 END BACKUP;
```

2. 最も古い時間間隔のパーティションを削除します。

```
ALTER TABLE order DROP PARTITION order_9310;
```

3. 最新の時間間隔のパーティションを追加します。

```
ALTER TABLE order ADD PARTITION order_9411;
```

4. グローバル索引のパーティションを再作成します。

```
ALTER INDEX order_ix_cust REBUILD PARTITION order_ix_cust_AH;
ALTER INDEX order_ix_cust REBUILD PARTITION order_ix_cust_IP;
ALTER INDEX order_ix_cust REBUILD PARTITION order_ix_cust_QZ;
```

通常、Oracle は、`ALTER TABLE ... DROP PARTITION` などの個別の DDL 文が別の操作（DML、DDL またはユーティリティ）によって妨げられないようにするため、十分なロックを取得します。しかし、パーティションのメンテナンス操作に複数の手順が必要な場合、DBA は、アプリケーション（またはその他のメンテナンス操作）によって、進行中の複数手順の操作が妨げられないように注意する必要があります。そのための方法を次にいくつか示します。

- 明確に規定された時間帯に、ユーザー・レベルのアプリケーションをすべて停止する。
- すべてのアプリケーションで使用されているロールのアクセス権限を取り消して、表 `order` に誰もアクセスできないようにする。

パーティション・ビューからパーティション表への変換

ここでは、パーティション・ビュー（「マニュアル・パーティション」とも呼ばれる）をパーティション表に変換する方法について説明します。パーティション・ビューは、次のように定義されています。

```
CREATE VIEW accounts AS
      SELECT * FROM accounts_jan98
      UNION ALL
      SELECT * FROM accounts_feb98
      UNION ALL
      ...
      SELECT * FROM accounts_dec98;
```

パーティション・ビューをパーティション表に段階的に移行する手順は、次のとおりです。

1. 最初に、パーティション表を作成し、最新の 2 つのパーティション accounts_nov98 と accounts_dec98 のみをビューから表に移行します。各パーティションは、2 ブロックのセグメントを（プレースホルダとして）取得します。

```
CREATE TABLE accounts_new (...)
    TABLESPACE ts_temp STORAGE (INITIAL 2)
    PARTITION BY RANGE (opening_date)
         (PARTITION jan98 VALUES LESS THAN ('01-FEB-1998'),
          ...
         PARTITION dec98 VALUES LESS THAN ('01-JAN-1999'));
```

2. EXCHANGE PARTITION 文を使用して、対応するパーティションに表を移行します。

```
ALTER TABLE accounts_new
    EXCHANGE PARTITION nov98 WITH TABLE
        accounts_nov98 WITH VALIDATION;

ALTER TABLE accounts_new
    EXCHANGE PARTITION dec98 WITH TABLE
        accounts_dec98 WITH VALIDATION;
```

これにより、nov98 パーティションおよび dec98 パーティションに対応付けられたプレースホルダ・データ・セグメントが、accounts_nov98 表と accounts_dec98 表に対応付けられたデータ・セグメントと交換されます。

3. accounts ビューを再定義します。

```
CREATE OR REPLACE VIEW accounts AS
    SELECT * FROM accounts_jan98
    UNION ALL
    SELECT * FROM accounts_feb_98
    UNION ALL
    ...
    UNION ALL
    SELECT * FROM accounts_new PARTITION (nov98)
    UNION ALL
    SELECT * FROM accounts_new PARTITION (dec98);
```

4. accounts_nov98 表と accounts_dec98 表を削除します。これらの表には、nov98 パーティションと dec98 パーティションに含まれていた元のプレースホルダ・セグメントが格納されています。

5. UNION ALL ビュー内のすべての表をパーティションに変換した後、ビューを削除し、パーティション化した表を、削除したビューの名前に変更します。

```
DROP VIEW accounts;
RENAME accounts_new TO accounts;
```

パーティション化された表および索引の情報の表示

次のビューには、パーティション化された表および索引に固有の情報が表示されます。

ビュー	説明
DBA_PART_TABLES ALL_PART_TABLES USER_PART_TABLES	DBA ビューには、データベース内にあるすべてのパーティション表のパーティション化情報が表示されます。ALL ビューには、ユーザーがアクセス可能なすべてのパーティション表のパーティション化情報が表示されます。USER ビューは、ユーザーが所有するパーティション表のパーティション化情報のみに制限されます。
DBA_TAB_PARTITIONS ALL_TAB_PARTITIONS USER_TAB_PARTITIONS	DBMS_STATS パッケージまたは ANALYZE 文で生成されるパーティション・レベルのパーティション化情報、パーティションの記憶域パラメータおよびパーティションの統計が表示されます。
DBA_TAB_SUBPARTITIONS ALL_TAB_SUBPARTITIONS USER_TAB_SUBPARTITIONS	DBMS_STATS パッケージまたは ANALYZE 文で生成されるサブパーティション・レベルのパーティション化情報、サブパーティションの記憶域パラメータおよびサブパーティションの統計が表示されます。
DBA_PART_KEY_COLUMNS ALL_PART_KEY_COLUMNS USER_PART_KEY_COLUMNS	パーティション表のパーティション化キー列が表示されます。
DBA_SUBPART_KEY_COLUMNS ALL_SUBPART_KEY_COLUMNS USER_SUBPART_KEY_COLUMNS	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列（およびコンポジット・パーティション表のローカル索引）が表示されます。
DBA_PART_COL_STATISTICS ALL_PART_COL_STATISTICS USER_PART_COL_STATISTICS	表のパーティションについて、列の統計およびヒストグラム情報が表示されます。
DBA_SUBPART_COL_STATISTICS ALL_SUBPART_COL_STATISTICS USER_SUBPART_COL_STATISTICS	表のサブパーティションについて、列の統計およびヒストグラム情報が表示されます。
DBA_PART_HISTOGRAMS ALL_PART_HISTOGRAMS USER_PART_HISTOGRAMS	表のパーティションのヒストグラムに関するヒストグラム・データ（各ヒストグラムのエンドポイント）が表示されます。
DBA_SUBPART_HISTOGRAMS ALL_SUBPART_HISTOGRAMS USER_SUBPART_HISTOGRAMS	表のサブパーティションのヒストグラムに関するヒストグラム・データ（各ヒストグラムのエンドポイント）が表示されます。

ビュー	説明
DBA_PART_INDEXES ALL_PART_INDEXES USER_PART_INDEXES	パーティション索引のパーティション化情報が表示されます。
DBA_IND_PARTITIONS ALL_IND_PARTITIONS USER_IND_PARTITIONS	索引パーティションについて、DBMS_STATS パッケージまたは ANALYZE 文で収集されたパーティション・レベルのパーティション化情報、パーティションの記憶域パラメータ、統計が表示されます。
DBA_IND_SUBPARTITIONS ALL_IND_SUBPARTITIONS USER_IND_SUBPARTITIONS	索引サブパーティションについて、DBMS_STATS パッケージまたは ANALYZE 文で収集されたパーティション・レベルのパーティション化情報、パーティションの記憶域パラメータ、統計が表示されます。

関連項目：

- これらのビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。
- ヒストグラムおよび表の統計生成の詳細は、『Oracle9*i* データベース・パフォーマンス・プランニング』および『 Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。
- 21-4 ページ「表、索引およびクラスタの分析」

18

クラスタの管理

この章では、索引付きクラスタ、クラスタ化表およびクラスタ索引の管理など、クラスタを管理する方法について説明します。この章の内容は、次のとおりです。

- クラスタを管理するためのガイドライン
- クラスタの作成
- クラスタの変更
- クラスタの削除
- クラスタ情報の表示

関連項目：

- もう 1 つのタイプのクラスタ（ハッシュ・クラスタ）の詳細は、第 19 章「ハッシュ・クラスタの管理」を参照してください。
- この章のタスクを実行する前に、第 14 章「スキーマ・オブジェクトの領域の管理」を一読されることをお薦めします。

クラスタを管理するためのガイドライン

クラスタは、表データを格納するために選択可能なオプションの方法を提供します。クラスタは、同じデータ・ブロックを共有する表のグループで構成されています。表をグループ化する理由は、各表が共通の列を共有しており、一緒に使用されるケースが多いためです。たとえば、emp 表と dept 表が deptno 列を共有しているとします。emp 表および dept 表をクラスタ化する場合（図 18-1 を参照）、emp 表および dept 表の各部門の行はすべて、物理的に同じデータ・ブロックに格納されます。

クラスタは、異なる表の関連する行を同じデータ・ブロックに格納します。そのため、クラスタを正しく使用することには、主に次のような利点があります。

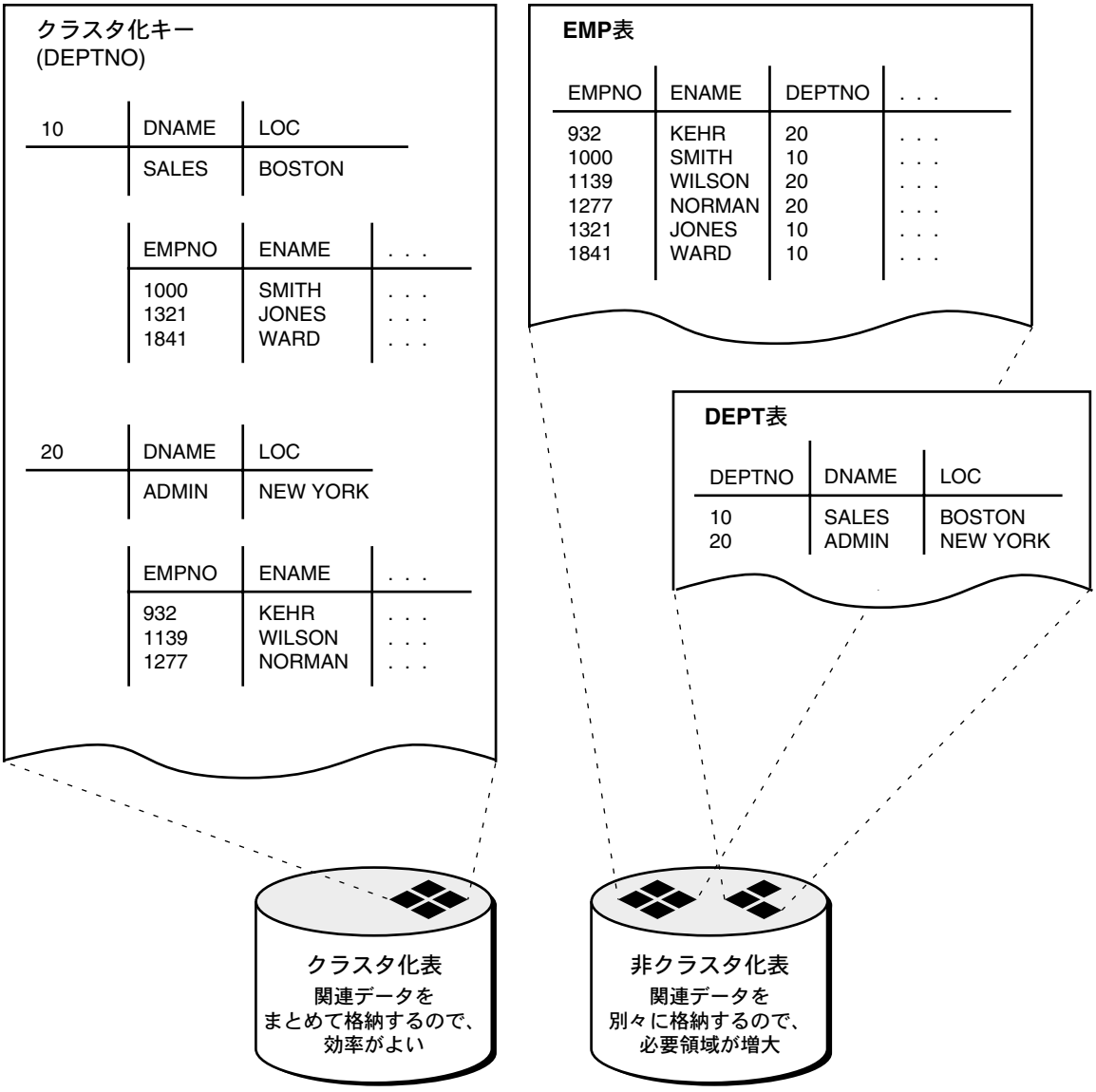
- クラスタ化表の結合によって、ディスク I/O が減少し、アクセス時間が改善されます。
- **クラスタ・キー**は、クラスタ化表が共有する列または列のグループです。最初にクラスタを作成するときに、クラスタ・キー列を指定します。その後、そのクラスタに追加する表を作成するたびに同じ列を指定します。各クラスタ・キー値は、その値が含まれている異なる表行数に関係なく、クラスタとクラスタ索引のそれぞれに 1 度しか格納されません。

  そのため、関連する表および索引データをクラスタに格納するために必要な記憶域は、クラスタ化されていない表形式の場合に比べて少なくて済みます。図 18-1 に、クラスタ・キーの格納方法を示します。各クラスタ・キー（各 deptno）は、emp 表と dept 表の両方に同じ値が含まれている多数の行について 1 度しか格納されていません。

クラスタを作成した後、そのクラスタ内に表を作成できます。ただし、クラスタ化表に行を挿入する前に、クラスタ索引を作成する必要があります。クラスタを使用しても、クラスタ化表に対して索引を追加することには影響しません。索引は通常どおり作成および削除できます。

別々にアクセスされることの多い表には、クラスタを使用しないでください。

図 18-1　クラスタ化表データ



次の項では、クラスタを管理する際に考慮すべきガイドラインについて説明します。この項の内容は、次のとおりです。

- クラスタに適した表の選択
- クラスタ・キーに適した列の選択
- データ・ブロック領域使用の指定
- 平均クラスタ・キーとその対応行が必要とする領域の指定
- 各クラスタとクラスタ索引の行の位置の指定
- クラスタ・サイズの見積りと記憶域パラメータの設定

関連項目：

- クラスタの詳細は、『Oracle9*i* データベース概要』を参照してください。
- どのようなときにクラスタを使用するかについてのガイドラインは、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。

クラスタに適した表の選択

次のような条件が成り立つ場合は、表にクラスタを使用します。

- 表に対する主な操作が挿入や更新ではなく、問合せである場合
- 複数の表のレコードを頻繁に問い合せたり、結合する場合

クラスタ・キーに適した列の選択

クラスタ・キー列の選択には注意が必要です。表を結合する問合せで複数列を使用する場合、クラスタ・キーはコンポジット・キーにします。一般に、適切なクラスタ索引を表す特性は、適切な索引を表す特性と同じです。適切な索引を表す特性の詳細は、16-2 ページの「索引を管理するためのガイドライン」を参照してください。

適切なクラスタ・キーの条件は、1 つのキー値に対応している行のグループで 1 つのデータ・ブロックがほぼいっぱいになるような、一意の値を持つことです。クラスタ・キー値ごとの行が少なすぎると、領域を浪費し、結果的にパフォーマンスが低下します。共通の値を共有する行がわずかしかないクラスタ・キーは、クラスタを作成するときに `SIZE` に小さい値を指定しないかぎり、ブロック内の領域を浪費する可能性があります（18-5 ページの「平均クラスタ・キーとその対応行が必要とする領域の指定」を参照してください）。

クラスタ・キー値ごとの行が多すぎると、そのキーを持つ行を見つけるために余分な検索が起こる可能性があります。クラスタ・キーの値があまりにも一般的な場合（たとえば、male と female といった性別など）は、過度の検索が発生し、クラスタ化していない場合よりパフォーマンスが低下するおそれがあります。

クラスタ索引は一意にできません。また、LONG 型として定義されている列を含むことはできません。

データ・ブロック領域使用の指定

クラスタの作成時に PCTFREE および PCTUSED パラメータを指定することにより、領域使用率と、現在行を更新するためにクラスタのデータ・セグメントのデータ・ブロック内に確保される領域の大きさを制御できます。クラスタ内の表に設定された PCTFREE および PCTUSED パラメータは、無視されます。つまり、クラスタ化表はクラスタの設定を自動的に使用します。

関連項目： PCTFREE および PCTUSED パラメータ設定の詳細は、14-2 ページの「データ・ブロックの領域管理」を参照してください。

平均クラスタ・キーとその対応行が必要とする領域の指定

CREATE CLUSTER 文にはオプションの引数 SIZE があります。これは、平均クラスタ・キーとそれに対応する行で必要なバイト数の見積りです。Oracle では、次の処理を実行するときに、SIZE パラメータが使用されます。

- クラスタ化したデータ・ブロック内に収めることのできるクラスタ・キー（および対応する行）の数を見積る場合。
- クラスタ化したデータ・ブロックに配置するクラスタ・キーの数を制限する場合。これにより、クラスタ内のキーの格納効率が最大になります。

SIZE は、クラスタ・キーが使用できる領域を制限しません。たとえば、2 つのクラスタ・キーが 1 つのデータ・ブロックに収まるように SIZE が設定された場合、どちらのクラスタ・キーでも、その利用可能なデータ・ブロック領域をいくらでも使用できます。

デフォルトでは、Oracle は 1 つのクラスタ・キーとそれに対応する行をそのクラスタのデータ・セグメントの 1 データ・ブロックに格納します。ブロック・サイズはオペレーティング・システムによって異なることがありますが、クラスタ化表が異なるマシン上にある他のデータベースにインポートされるときにも、ブロック当たり 1 つのキーというルールは守られます。

クラスタ・キー値に対応するすべての行を 1 つのブロック内に収めることができない場合は、ブロックをまとめて連鎖することにより、特定のキーを持つすべての値へのアクセスの高速化が図られます。クラスタ索引は、クラスタ・キー値と対応する行をそれぞれに含むブロックの連鎖の始点を示します。クラスタの SIZE が、複数のキーが 1 つのブロックに収まる大きさに設定されている場合は、ブロックが複数の連鎖に属する可能性があります。

各クラスタとクラスタ索引の行の位置の指定

適切な権限と表領域割当て制限を持つユーザーであれば、現在オンライン状態の表領域内に新しいクラスタおよび関連するクラスタ索引を作成できます。新しいクラスタまたは索引を格納する表領域を識別するには、CREATE CLUSTER/INDEX 文に必ず TABLESPACE オプションを指定します。

クラスタとそのクラスタ索引は、異なる表領域内に作成できます。実際、クラスタとその索引を、それぞれ異なる記憶デバイス上に格納された異なる表領域内に作成すると、ディスク競合を最小限に抑えて、表データと索引データを同時に検索できます。

クラスタ・サイズの見積りと記憶域パラメータの設定

クラスタを作成する前にクラスタのサイズを見積る利点は、次のとおりです。

- クラスタの見積りサイズの合計と、索引、ロールバック・セグメントおよび REDO ログ・ファイルの見積りを使用して、作成するデータベースを格納するために必要なディスク容量を決定できます。この見積りを利用して適切なハードウェアを購入できます。
- 個々のクラスタの見積りサイズを使用することで、クラスタが使用するディスク領域をより適切に管理できます。クラスタを作成するときに、適切な記憶域パラメータを設定し、そのクラスタを使用するアプリケーションの I/O パフォーマンスを改善できます。

表を作成する前に表サイズを見積るかどうかにかかわらず、クラスタ化されていない表を作成するときは記憶域パラメータを明示的に設定できます。表を作成するとき、または表を変更するときに記憶域パラメータを明示的に設定しない場合は、その表が存在する表領域に設定されたデフォルト記憶域パラメータが自動的に使用されます。クラスタ化表は、クラスタの記憶域パラメータも自動的に使用します。

クラスタの作成

自分のスキーマにクラスタを作成するには、CREATE CLUSTER システム権限とそのクラスタを格納する表領域に対する割当て制限を持っているか、または UNLIMITED TABLESPACE システム権限を持っている必要があります。

別のユーザーのスキーマにクラスタを作成するには、CREATE ANY CLUSTER システム権限が必要です。さらに、所有者はそのクラスタを格納する表領域に対する割当て制限を持っているか、または UNLIMITED TABLESPACE システム権限を持っている必要があります。

クラスタを作成するには、CREATE CLUSTER 文を使用します。次の文は、deptno 列によってクラスタ化された、emp 表と dept 表を格納するクラスタ emp_dept を作成します。

```
CREATE CLUSTER emp_dept (deptno NUMBER(3))
   PCTUSED 80
   PCTFREE 5
   SIZE 600
   TABLESPACE users
   STORAGE (INITIAL 200K
      NEXT 300K
      MINEXTENTS 2
      MAXEXTENTS 20
      PCTINCREASE 33);
```

この例のように INDEX キーワードを指定しない場合は、デフォルトで索引クラスタが作成されます。また、ハッシュ・パラメータ（HASHKEYS、HASH IS または SINGLE TABLE HASHKEYS）を指定すると、ハッシュ・クラスタを作成できます。ハッシュ・クラスタについては、第 19 章「ハッシュ・クラスタの管理」を参照してください。

関連項目： この章に記載されている SQL 文の構文、制限および必要な認可の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

クラスタ化表の作成

クラスタに表を作成するには、CREATE TABLE システム権限または CREATE ANY TABLE システム権限のどちらかが必要です。なお、クラスタ内に表を作成するために、表領域割当て制限または UNLIMITED TABLESPACE システム権限は必要ありません。

クラスタに表を作成するには、CLUSTER オプションを指定した CREATE TABLE 文を使用します。次の文を使用すると、emp_dept クラスタ内に emp 表と dept 表を作成できます。

```
CREATE TABLE emp (
   empno NUMBER(5) PRIMARY KEY,
   ename VARCHAR2(15) NOT NULL,
   . . .
   deptno NUMBER(3) REFERENCES dept)
   CLUSTER emp_dept (deptno);

CREATE TABLE dept (
   deptno NUMBER(3) PRIMARY KEY, . . . )
   CLUSTER emp_dept (deptno);
```

注意： クラスタ化表のスキーマは、CREATE TABLE 文で指定できます。クラスタ化表は、そのクラスタを含むスキーマとは異なるスキーマに配置できます。また、列名は一致しなくてもかまいませんが、その構造は同じである必要があります。

クラスタ索引の作成

クラスタ索引を作成するには、次の条件のいずれかが成り立つ必要があります。

- スキーマにクラスタが含まれている。
- CREATE ANY INDEX システム権限を持っている。

どちらの場合も、クラスタ索引を格納する表領域に対する割当て制限または UNLIMITED TABLESPACE システム権限が必要です。

クラスタ索引は、クラスタ化表に行を挿入する前に作成する必要があります。次の文は、emp_dept クラスタに対するクラスタ索引を作成します。

```
CREATE INDEX emp_dept_index
   ON CLUSTER emp_dept
   INITRANS 2
   MAXTRANS 5
   TABLESPACE users
   STORAGE (INITIAL 50K
      NEXT 50K
      MINEXTENTS 2
      MAXEXTENTS 10
      PCTINCREASE 33)
   PCTFREE 5;
```

クラスタ索引句（ON CLUSTER）では、クラスタ索引を作成するクラスタ emp_dept を識別します。この文では、クラスタとクラスタ索引の記憶域設定も明示的に指定しています。

クラスタの変更

クラスタを変更するには、そのクラスタが自分のスキーマに含まれているか、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。既存のクラスタを変更することにより、次の設定を変更できます。

- 物理属性（PCTFREE、PCTUSED、INITRANS、MAXTRANS および記憶特性）
- クラスタ・キー値のすべての行を格納するために必要な平均使用可能空き領域（SIZE）
- デフォルトの並列度

また、クラスタに新しいエクステントを明示的に割り当てたり、クラスタの最後にある未使用のエクステントの割当てを解除できます。Oracle は、必要に応じてクラスタのデータ・セグメントに追加のエクステントを動的に割り当てます。ただし、状況によっては、クラスタに追加のエクステントを明示的に割り当てることもできます。たとえば、Oracle9*i* Real Application Clusters を使用している場合、クラスタのエクステントを特定のインスタンスに明示的に割り当てることができます。クラスタに新しいエクステントを割り当てるには、ALLOCATE EXTENT 句を指定した ALTER CLUSTER 文を使用します。

クラスタのデータ・ブロック・スペース使用パラメータ（PCTFREE と PCTUSED）またはクラスタ・サイズ・パラメータ（SIZE）を変更すると、そのクラスタにすでに割り当てられているブロックと今後割り当てられるブロックを含め、そのクラスタが使用するすべてのデータ・ブロックに対して新しい設定が適用されます。すでに表に割り当てられているブロックは、即時ではなく、必要なときに再編成されます。

クラスタのトランザクション・エントリ設定（INITRANS および MAXTRANS）を変更したとき、MAXTRANS の新しい設定はクラスタのすべてのデータ・ブロック（すでに割り当てられたブロックとその後割り当てられるブロック）に適用されますが、INITRAN の新しい設定はその後クラスタに割り当てられるブロックのみに適用されます。

記憶域パラメータ INITIAL と MINEXTENTS は変更できません。他の記憶域パラメータの新しい設定はすべて、その後にクラスタに割り当てられるエクステントにのみ影響します。

クラスタを変更するには、ALTER CLUSTER 文を使用します。次の文は、emp_dept クラスタを変更します。

```
ALTER CLUSTER emp_dept
   PCTFREE 30
   PCTUSED 60;
```

関連項目： Oracle Real Application Clusters 環境での ALTER CLUSTER 文固有の使用方法は、『Oracle9*i* Real Application Clusters 管理』を参照してください。

クラスタ化表の変更

クラスタ化表を変更するには、ALTER TABLE 文を使用します。ただし、クラスタ化表に対して ALTER TABLE 文でデータ・ブロック領域パラメータ、トランザクション・エントリ・パラメータまたは記憶域パラメータを設定しても、エラー・メッセージ（ORA-01771「クラスタ表に対するオプションが無効です。」）が出力されます。これは、クラスタ化表では必ずそのクラスタのパラメータが使用されるためです。そのため、クラスタ化表に対して ALTER TABLE 文を使用できるのは、列の追加や変更、クラスタ化されていないキー列の削除、整合性制約やトリガーの追加、削除、使用可能または使用禁止のみに限定されます。表の変更方法については、15-10 ページの「表の変更」を参照してください。

クラスタ索引の変更

クラスタ索引は、他の索引と同じように変更できます。16-20 ページの「索引の変更」を参照してください。

注意： クラスタ索引のサイズを見積るときには、索引が実際の行ではなく各クラスタ・キーに付いていることに注意してください。したがって、各キーは索引内に 1 度しか現れません。

クラスタの削除

クラスタ内の表が不要になった場合は、そのクラスタを削除できます。クラスタを削除すると、そのクラスタ内の表および対応するクラスタ索引も削除されます。クラスタのデータ・セグメントとクラスタ索引の索引セグメントの両方に属するすべてのエクステントは、それらを含んでいる表領域に戻され、その表領域内の他のセグメントで使用可能になります。

表を含まないクラスタとそのクラスタ索引を削除するには、DROP CLUSTER 文を使用します。たとえば、次の文は空のクラスタ emp_dept を削除します。

```
DROP CLUSTER emp_dept;
```

クラスタに 1 つ以上のクラスタ化表が含まれており、その表も同様に削除する場合は、次のように、DROP CLUSTER 文に INCLUDING TABLES オプションを追加します。

```
DROP CLUSTER emp_dept INCLUDING TABLES;
```

INCLUDING TABLES オプションを指定していない場合に、クラスタに表が含まれていると、エラーが返されます。

クラスタ内の 1 つ以上の表が、そのクラスタ外の表の FOREIGN KEY 制約によって参照される主キーまたは一意キーを含んでいる場合、依存する FOREIGN KEY 制約が削除されないかぎり、そのクラスタを削除できません。この削除は、DROP CLUSTER 文の CASCADE CONSTRAINTS オプションを使用すると簡単に実行できます。次に例を示します。

```
DROP CLUSTER emp_dept INCLUDING TABLES CASCADE CONSTRAINTS;
```

制約が存在している場合に、CASCADE CONSTRAINTS オプションを使用しないと、Oracle はエラーを返します。

クラスタ化表の削除

クラスタを削除するには、そのクラスタが自分のスキーマに含まれているか、または DROP ANY CLUSTER システム権限を持っている必要があります。クラスタ化表をそのクラスタの所有者が所有していなくても、その表を含むクラスタを削除するために特別な権限は必要ありません。

クラスタ化表は、その表のクラスタ、他のクラスタ化表またはクラスタ索引に影響を及ぼすことなく、個別に削除できます。クラスタ化表は、クラスタ化されていない表を削除する場合と同じように、DROP TABLE 文を使用して削除します。15-22 ページの「表の削除」を参照してください。

注意： 単一の表をクラスタから削除するときは、表の各行が個別に削除されます。クラスタ全体を最も効率よく削除するには、INCLUDING TABLES オプションを指定した DROP CLUSTER 文を使用して、そのクラスタを表も含めて削除します。クラスタの残りの部分をそのままにしておく場合のみ、（DROP TABLE 文を使用して）クラスタから表を個別に削除してください。

クラスタ索引の削除

クラスタ索引は、クラスタまたはそのクラスタ化表に影響を及ぼすことなく削除できます。ただし、クラスタ索引が存在しないと、クラスタ化表を使用できません。クラスタへのアクセスを可能にするには、クラスタ索引の再作成が必要です。断片化したクラスタ索引を再作成する手順の一部として、クラスタ索引を削除することがあります。索引を削除する方法の詳細は、16-23 ページの「索引の削除」を参照してください。

クラスタ情報の表示

次のビューには、クラスタに関する情報が表示されます。

ビュー	説明
DBA_CLUSTERS ALL_CLUSTERS USER_CLUSTERS	DBA ビューには、データベース内のすべてのクラスタが表示されます。ALL ビューには、ユーザーがアクセス可能なすべてのクラスタが表示されます。USER ビューは、ユーザーが所有するクラスタのみに制限されます。これらのビューの一部の列には、DBMS_STATS パッケージまたは ANALYZE 文によって生成される統計が含まれます。
DBA_CLU_COLUMNS USER_CLU_COLUMNS	これらのビューでは、表の列とクラスタの列がマップされています。

関連項目： これらのビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

19

ハッシュ・クラスタの管理

この章では、ハッシュ・クラスタを管理する方法について説明します。この章の内容は、次のとおりです。

- ハッシュ・クラスタを使用する場合
- ハッシュ・クラスタの作成
- ハッシュ・クラスタの変更
- ハッシュ・クラスタの削除
- ハッシュ・クラスタ情報の表示

関連項目： この章のタスクを実行する前に、第 14 章「スキーマ・オブジェクトの領域の管理」を一読されることをお薦めします。

ハッシュ・クラスタを使用する場合

ハッシュ・クラスタに表を格納することは、データ検索のパフォーマンスを改善するための 1 つの選択肢です。ハッシュ・クラスタは、索引付きの非クラスタ化表または索引クラスタの代替手段を提供します。索引付きの表または索引クラスタでは、別個の索引に格納されるキー値を使用して、表内の行の位置を決定します。ハッシングを使用するには、ハッシュ・クラスタを作成し、そこに表をロードします。表の行は物理的にはハッシュ・クラスタに格納され、**ハッシュ関数**の結果に従って検索されます。

Oracle はハッシュ関数を使用して、特定のクラスタ・キー値に基づく、**ハッシュ値**と呼ばれる数値の分布を生成します。ハッシュ・クラスタのキーは、索引クラスタのキーと同じように単一列キーでもコンポジット・キー（複数列キー）でもかまいません。ハッシュ・クラスタ内で行を検索または格納する場合、Oracle は行のクラスタ・キー値にハッシュ関数を適用します。結果として生成されるハッシュ値はクラスタ内のデータ・ブロックに対応しており、Oracle は、発行された文のためにそのデータ・ブロックに対して読取りまたは書込みを行います。

索引付きの表または索引クラスタ内の行を検索または格納するためには、少なくとも 2 回（通常はそれ以上）の I/O を実行する必要があります。

- 1 回以上の I/O による、索引内でのキー値の検索または格納
- 別の I/O による、表またはクラスタ内での行の読取りまたは書込み

一方、ハッシュ・クラスタでは、ハッシュ関数を使用して行の位置が特定されるので、I/O は必要ありません。結果として、ハッシュ・クラスタ内の行の読込みや書込みに必要とされるのは最小限の I/O 操作のみになります。

ここでは、ハッシングが最も有効な場合と有効でない場合を対比することによって、ハッシュ・クラスタを使用すべき状況を判断する際に役立つ情報を提供します。ハッシングではなく索引を使用する場合は、表を個別に格納するのか、またはクラスタの一部として格納するのかを考慮してください。

注意： ハッシングを使用する場合でも、クラスタ・キーを含む表のどの列にも異なる索引を設定できます。

関連項目：

- ハッシュ・クラスタの詳細は、『Oracle9*i* データベース概要』を参照してください。
- ハッシュ・クラスタの使用に関する追加の推奨事項は、『Oracle9*i* アプリケーション開発者ガイドー基礎編』を参照してください。

ハッシングが有効な状況

ハッシングは、次のような状況で有効です。

- ほとんどの問合せが次のようなクラスタ・キーとの等式を含んでいる場合。

  ```
  SELECT ... WHERE cluster_key = ...;
  ```

  このような場合、等価条件内のクラスタ・キーがハッシュされ、対応するハッシュ・キーが通常 1 回の読込みで検索されます。それに対して、索引付きの表では、最初にキー値を索引内で検索する必要があります（通常複数回の読込み）。その後で行が表から読み込まれます（別の読込み）。

- ハッシュ・クラスタ内の表のサイズが最初から固定されていて、行数とそのクラスタ内の表が必要とする領域を決定できる場合。ハッシュ・クラスタ内の表でそのクラスタの初期割当てより多くの領域が必要な場合、オーバーフロー・ブロックが必要となるためにパフォーマンスがかなり低下するおそれがあります。

ハッシングが不利な状況

ハッシングは次のような状況では有効ではありません。

- ほとんどの問合せがクラスタ・キー値全体にわたって行を検索する場合。たとえば、全表スキャンや次のような問合せでは、ハッシュ関数は特定のハッシュ・キーの位置を決定するために使用できません。そのかわりに、全表スキャンと同等の機能を実行して問合せの行をフェッチする必要があります。

  ```
  SELECT . . . WHERE cluster_key < . . . ;
  ```

  索引では、キー値はその索引内で順序付けられており、問合せの WHERE 句を満たすクラスタ・キー値を、比較的少ない I/O で見つけることができます。

- 表が固定でなく、継続的に拡大する場合。表が無制限に拡大する場合、表（そのクラスタ）の存続期間にわたって必要な領域を事前に定義することはできません。

- アプリケーションが頻繁に表の全表スキャンを実行し、その表内でデータが散在している場合。この状況でハッシングを使用すると、全表スキャンの処理時間が長くなります。

- 最終的にハッシュ・クラスタが必要とする領域を事前に割り当てることができない場合。

ハッシュ・クラスタの作成

ハッシュ・クラスタを作成するには、HASHKEYS 句を指定した CREATE CLUSTER 文を使用します。次の例では、trial 表を格納するクラスタ trial_cluster を作成し、trialno 列（クラスタ・キー）によってクラスタ化する文と、クラスタ内に表を作成する文を示しています。

```
CREATE CLUSTER trial_cluster (trialno NUMBER(5,0))
    PCTUSED 80
    PCTFREE 5
    TABLESPACE users
    STORAGE (INITIAL 250K     NEXT 50K
      MINEXTENTS 1     MAXEXTENTS 3
      PCTINCREASE 0)
    HASH IS trialno HASHKEYS 150;

CREATE TABLE trial (
    trialno NUMBER(5,0) PRIMARY KEY,
    ...)
    CLUSTER trial_cluster (trialno);
```

索引クラスタの場合と同様に、ハッシュ・クラスタのキーは、単一列でもコンポジット・キー（複数列のキー）でもかまいません。この例では、単一列が使用されています。

HASHKEYS 値（この例では 150）は、クラスタに使用されるハッシュ関数で生成できる一意のハッシュ値の数を指定し、制限します。指定した値は最も近い素数に丸められます。

HASH IS 句を指定しない場合は、内部ハッシュ関数が使用されます。すでにクラスタ・キーがその範囲に均一に分布する一意識別子の場合は、内部ハッシュ関数を無視し、前述の例のようにクラスタ・キーをハッシュ値として指定できます。また、HASH IS 句を使用して、ユーザー定義のハッシュ関数を指定することもできます。

ハッシュ・クラスタのクラスタ索引は作成できませんが、ハッシュ・クラスタ・キーに索引を作成する必要はありません。

クラスタ内に表を作成する方法、索引とハッシュ・クラスタに共通する CREATE CLUSTER 文のパラメータ設定のガイドライン、およびクラスタを作成するために必要な権限の追加情報は、第 18 章「クラスタの管理」を参照してください。次の項では、ハッシュ・クラスタ固有の CREATE CLUSTER 文のパラメータの設定とそのガイドラインについて説明します。

- 単一表ハッシュ・クラスタの作成
- ハッシュ・クラスタ内の領域使用の制御
- ハッシュ・クラスタに必要なサイズの見積り

関連項目：

- ハッシュ関数の説明とユーザー定義ハッシュ関数の指定は、『Oracle9*i* データベース概要』を参照してください。
- SQL 文 CREATE CLUSTER および CREATE TABLE の構文、制限および必要な認可の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

単一表ハッシュ・クラスタの作成

表中の行への高速なアクセスを提供する**単一表ハッシュ・クラスタ**を作成できます。ただし、この表はハッシュ・クラスタ内の唯一の表にする必要があります。ハッシュ・キーとデータ行の間に 1 対 1 のマッピングが必要となるためです。次の文は、クラスタ・キー variety を持つ単一表ハッシュ・クラスタ peanut を作成します。

```
CREATE CLUSTER peanut (variety NUMBER)
   SIZE 512 SINGLE TABLE HASHKEYS 500;
```

HASHKEYS 値は最も近い素数に丸められるため、このクラスタはそれぞれサイズ 512 バイトのハッシュ・キー値を最大 503 個持ちます。

注意： SINGLE TABLE オプションは、ハッシュ・クラスタにのみ有効です。また、必ず HASHKEYS も指定する必要があります。

ハッシュ・クラスタ内の領域使用の制御

ハッシュ・クラスタを作成するときは、パフォーマンスと使用領域が最適になるように、クラスタ・キーを正しく選択し、HASH IS、SIZE および HASHKEYS の各パラメータを設定することが重要です。次に示すガイドラインでは、これらのパラメータを設定する方法について説明します。

キーの選択

正しいクラスタ・キーの選択は、クラスタ化表に対して最もよく発行される問合せのタイプによって決まります。たとえば、ハッシュ・クラスタ内の emp 表について検討します。問合せが従業員番号によって行を頻繁に選択する場合、最適なクラスタ・キーは empno 列です。問合せが部門によって行を頻繁に選択する場合、最適なクラスタ・キーは deptno 列です。単一の表を含むハッシュ・クラスタの場合は、通常、含まれる表の主キー全体をクラスタ・キーにします。

ハッシュ・クラスタのキーは、索引クラスタのキーと同じように単一列でもコンポジット・キー（複数列キー）でもかまいません。コンポジット・キーによるハッシュ・クラスタは、Oracle の内部ハッシュ関数を使用する必要があります。

HASH IS パラメータの設定

HASH IS パラメータを指定するのは、クラスタ・キーが NUMBER データ型の単一の列であり、均一に分布した整数を含む場合のみです。この条件を満たしていれば、それぞれの一意のクラスタ・キー値が衝突（同じハッシュ値を持つクラスタ・キーが 2 つ発生すること）せずに一意のハッシュ値にハッシュするように、クラスタ内に行を分散させることができます。この条件が当てはまらない場合は、このオプションを指定せずに内部ハッシュ関数を使用してください。

SIZE パラメータの設定

SIZE は、ハッシュ・キーに対応するすべての行を保持するために必要な領域の平均サイズに設定します。そのため、SIZE を適切に決定するには、格納するデータの特性をよく理解する必要があります。

- ハッシュ・クラスタに含まれている表が 1 つで、その表の行のハッシュ・キー値が一意（1 つの値につき 1 行）である場合、SIZE はクラスタ内の平均の行サイズに設定できます。
- ハッシュ・クラスタが複数の表を含む場合、SIZE は代表的なハッシュ値に対応するすべての行を保持するために必要な領域の平均サイズに設定できます。

また、SIZE の見積り値を決定した後で、次の点を考慮してください。SIZE の値が小さい場合（データ・ブロックごとに 5 つ以上のハッシュ・キーを割り当てることができる場合）は、その値を CREATE CLUSTER 文の SIZE に使用できます。しかし、SIZE の値が大きい場合（データ・ブロックごとに割当て可能なハッシュ・キーが 4 つ以下の場合）は、衝突の発生頻度を予測し、データ検索パフォーマンスと領域使用効率のどちらを重視するか検討する必要があります。

- ハッシュ・クラスタで内部ハッシュ関数を使用せず（HASH IS を指定した場合）、衝突がわずかであるか、または衝突のないことが予想される場合は、見積り値をそのまま SIZE に設定できます。この場合、衝突は発生せず、領域は可能なかぎり効率的に使用されます。
- 挿入時に頻繁な衝突が予想される場合、行を格納するためにオーバーフロー・ブロックが割り当てられる可能性は高くなります。衝突が頻繁に起こる場合にブロックのオーバーフローの可能性を軽減し、最大のパフォーマンスを引き出すには、次のように SIZE を調整する必要があります。

ブロック当たりの使用可能領域 / 算出された SIZE	SIZE の設定
1	SIZE
2	SIZE+15%
3	SIZE+12%
4	SIZE+8%
>4	SIZE

ただし、SIZE の値を過大に見積ると、クラスタ内の未使用領域を増やすことになります。領域効率がデータ検索のパフォーマンスよりも重要な場合は、この調整を無視して SIZE に元の値を使用してください。

HASHKEYS パラメータの設定

ハッシュ・クラスタ内の行を最大限まで分散させるために、Oracle は HASHKEYS 値を最も近い素数に丸めます。

ハッシュ・クラスタ内の使用領域の制御例

次に示す例では、クラスタ・キーを正しく選択し、HASH IS、SIZE および HASHKEYS の各パラメータを設定する方法を示します。すべての例において、データ・ブロック・サイズは 2KB で、利用可能なデータ領域（ブロック・サイズからオーバーヘッドを差し引いた領域）の平均は各ブロックの 1,950 バイトとします。

例 1 ハッシュ・クラスタに emp 表をロードすることになりました。ほとんどの問合せは、従業員番号によって従業員レコードを検索します。emp 表の最大行数は常に 10,000、平均の行サイズは 55 バイトと見積られています。

この場合は、empno をクラスタ・キーにします。この列には一意の整数が格納されているので、内部ハッシュ関数は無視できます。SIZE は平均の行サイズ（55 バイト）に設定できます。これにより、各データ・ブロックに 34 のハッシュ・キーが割り当てられます。HASHKEYS は、表の行数である 10,000 に設定できます。この値は、10,000 より大きい最初の素数である 10,007 に切り上げられます。

```
CREATE CLUSTER emp_cluster (empno
NUMBER)
. . .
SIZE 55
HASH IS empno HASHKEYS 10000;
```

例 2 前述の例と同様の条件を考えます。ただし、この例では、ほとんどの場合、行が部門番号によって検索されるとします。平均 10 名の従業員を持つ部門が最大で 1,000 部門存在します。部門番号は 10 ずつ増加します（0、10、20、30、...）。

この場合は、deptno をクラスタ・キーにします。この列には一様に分布する整数が格納されているので、内部ハッシュ関数は無視できます。事前に見積った SIZE（各部門のすべての行を保持するために必要な領域の平均サイズ）は 55 × 10 バイト、つまり 550 バイトです。SIZE にこの値を使用して、各データ・ブロックに 3 つのハッシュ・キーのみを割り当てることができます。いくらかの衝突が予想される状況で、最大限のデータ検索パフォーマンスが必要な場合、オーバーフロー・ブロックを必要とする衝突を防ぐために、見積りの SIZE を少し変更してください。ここでは、SIZE を 12% 調整して 620 バイトにします（19-6 ページの 19-6 ページ「SIZE パラメータの設定」を参照）。これにより、予想される衝突を考慮した上で行の領域をより多く確保できます。

HASHKEYS は、一意の部門番号の数である 1,000 に設定できます。この値は、1,000 より大きい最初の素数である 1,009 に切り上げられます。

```
CREATE CLUSTER emp_cluster (deptno NUMBER)
. . .
SIZE 620
HASH IS deptno HASHKEYS 1000;
```

ハッシュ・クラスタに必要なサイズの見積り

索引クラスタの場合と同じように、ハッシュ・クラスタ内のデータに必要な記憶域を見積ることは重要です。

Oracle は、SIZE および HASHKEYS の設定に従って、ハッシュ表の格納に十分な領域の初期割当てを保証します。ハッシュ表サイズを考慮せずに記憶域パラメータ INITIAL、NEXT および MINEXTENTS を設定した場合は、少なくとも SIZE*HASHKEYS に達するまで増分（追加の）エクステントが割り当てられます。たとえば、データ・ブロック・サイズが 2KB、各ブロックの使用可能なデータ領域（ブロック・サイズからオーバーヘッドを差し引いた領域）が約 1,900 バイトの場合に、CREATE CLUSTER 文で STORAGE パラメータと HASH パラメータを次のように指定したとします。

```
STORAGE (INITIAL 100K
    NEXT 150K
    MINEXTENTS 1
    PCTINCREASE 0)
SIZE 1500
HASHKEYS 100
```

この例では、各データ・ブロックにハッシュ・キーを 1 つのみ割り当てることができます。そのため、ハッシュ・クラスタが必要とする初期領域は少なくとも 200KB（100 × 2KB）です。しかし、記憶域パラメータの設定にはこの要件が考慮されていません。そのため、100KB の初期のエクステントと 150KB の増分のエクステントがハッシュ・クラスタに割り当てられます。

一方、HASH パラメータが次のように指定されたとします。

```
SIZE 500 HASHKEYS 100
```

この場合、各データ・ブロックに 3 つのハッシュ・キーが割り当てられます。そのため、ハッシュ・クラスタが必要とする初期領域は少なくとも 68KB（34 × 2KB）です。記憶域パラメータの初期設定はこの要件を満たしているため、100KB の初期エクステントがハッシュ・クラスタに割り当てられます。

ハッシュ・クラスタの変更

ハッシュ・クラスタは、ALTER CLUSTER 文を使用して変更できます。

```
ALTER CLUSTER emp_dept . . . ;
```

ハッシュ・クラスタの変更に関係する問題は、18-8 ページの 18-8 ページ「クラスタの変更」で説明されている索引クラスタを変更する場合と同じです。ただし、SIZE、HASHKEYS および HASH IS の各パラメータは ALTER CLUSTER 文に指定できません。これらのパラメータを変更するには、クラスタを再作成して、元のクラスタからデータをコピーする必要があります。

ハッシュ・クラスタの削除

ハッシュ・クラスタは、DROP CLUSTER 文を使用して削除できます。

```
DROP CLUSTER emp_dept;
```

ハッシュ・クラスタ内の表は、DROP TABLE 文を使用して削除されます。ハッシュ・クラスタとハッシュ・クラスタ内の表の削除についての問題は、索引クラスタの場合と同じです。

関連項目： 18-10 ページ「クラスタの削除」

ハッシュ・クラスタ情報の表示

次のビューには、ハッシュ・クラスタに関する情報が表示されます。

ビュー	説明
DBA_CLUSTERS ALL_CLUSTERS USER_CLUSTERS	DBA ビューには、データベース内のすべてのクラスタ（ハッシュ・クラスタを含む）が表示されます。ALL ビューには、ユーザーがアクセス可能なすべてのクラスタが表示されます。USER ビューは、ユーザーが所有するクラスタのみに制限されます。これらのビューの一部の列には、DBMS_STATS パッケージまたは ANALYZE 文によって生成される統計が含まれます。
DBA_CLU_COLUMNS USER_CLU_COLUMNS	これらのビューでは、表の列とクラスタの列がマップされています。
DBA_CLUSTER_HASH_EXPRESSIONS ALL_CLUSTER_HASH_EXPRESSIONS USER_CLUSTER_HASH_EXPRESSIONS	これらのビューには、ハッシュ・クラスタのハッシュ関数がリストされます。

関連項目： これらのビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

20

ビュー、順序およびシノニムの管理

この章では、ビュー、順序およびシノニムの管理について説明します。この章の内容は、次のとおりです。

- ビューの管理
- 順序の管理
- シノニムの管理
- ビュー、シノニムおよび順序の情報の表示

ビューの管理

ビューとは、1 つ以上の表（または他のビュー）に含まれているデータをユーザーの必要にあわせて調整したデータ表現であり、問合せの出力を収集し、それを表として扱います。ビューは、「ストアド・クエリー」または「仮想表」と考えることができます。表を使用できる場合は、ほとんどの場合、ビューも使用できます。

ここでは、ビューを管理する方法について説明します。この項の内容は、次のとおりです。

- ビューの作成
- 結合ビューの更新
- ビューの変更
- ビューの削除
- ビューの置換

ビューの作成

ビューを作成するには、次に示す要件を満たす必要があります。

- 自分のスキーマに新しいビューを作成するには、CREATE VIEW システム権限が必要です。別のユーザーのスキーマ内にビューを作成するには、CREATE ANY VIEW システム権限が必要です。これらの権限は明示的に取得するか、またはロールを介して取得できます。
- どのスキーマのビューであっても、その所有者は、ビュー定義で参照されるすべてのオブジェクトにアクセスする権限を明示的に付与されている必要があります。ロールを介してこれらの権限を取得することはできません。また、ビューの機能はビューの所有者の権限によって決まります。たとえば、ビューの所有者に Scott の emp 表の INSERT 権限しかない場合、emp 表に新しい行を挿入するためにはビューを使用できますが、このビューの行を選択（SELECT）、更新（UPDATE）または削除（DELETE）するためには使用できません。
- ビューの所有者がビューにアクセスする権限を他のユーザーに付与しようとする場合は、ベース・オブジェクトに対する GRANT OPTION 付きのオブジェクト権限、または ADMIN OPTION 付きのシステム権限が必要です。

ビューを作成するには、CREATE VIEW 文を使用します。ビューはそれぞれ、表、マテリアライズド・ビューまたは他のビューを参照する問合せによって定義されます。すべての副問合せと同様に、ビューを定義する問合せには、FOR UPDATE 句を指定できません。

次の文は、emp 表のデータのサブセットに対してビューを作成します。

```
CREATE VIEW sales_staff AS
      SELECT empno, ename, deptno
      FROM emp
      WHERE deptno = 10
    WITH CHECK OPTION CONSTRAINT sales_staff_cnst;
```

sales_staff ビューを定義する問合せは、部門番号 10 の行のみを参照します。また、CHECK OPTION は、そのビューが選択できない行に対して INSERT および UPDATE 文を発行できないという制約（sales_staff_cnst）付きでビューを作成します。たとえば、次の INSERT 文では、sales_staff ビュー（部門番号が 10 の行のみを含む）によって emp 表に行が正常に挿入されます。

```
INSERT INTO sales_staff VALUES (7584, 'OSTER', 10);
```

しかし、次の INSERT 文は、sales_staff ビューを使用しても選択できない部門番号 30 の行を挿入しようとしているため、エラーが返されます。

```
INSERT INTO sales_staff VALUES (7591, 'WILLIAMS', 30);
```

必要に応じて、WITH READ ONLY 句を指定してビューを作成できます。これにより、このビューからは、実表の更新、挿入または削除ができなくなります。WITH 句を指定しない場合、一部の制限を伴いますが、ビューは従来どおり更新可能です。

> **関連項目：** ビューの作成とメンテナンスに関する構文、制限および認可情報の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

結合ビュー

FROM 句で複数の実表またはビューを指定するビューを作成することもできます。この種のビューを**結合ビュー**と呼びます。次の文は、emp 表と dept 表のデータを結合する division1_staff ビューを作成します。

```
CREATE VIEW division1_staff AS
      SELECT ename, empno, job, dname
      FROM emp, dept
      WHERE emp.deptno IN (10, 30)
         AND emp.deptno = dept.deptno;
```

更新可能な結合ビューは、UPDATE、INSERT および DELETE 操作が可能な結合ビューです。詳細は、20-5 ページの「結合ビューの更新」を参照してください。

ビュー作成時の問合せ定義の展開

ビューが作成されるときに、Oracle は最上位のビュー問合せのワイルドカード（*）を列リストに展開します。結果の問合せはデータ・ディクショナリに格納され、副問合せはそのまま残されます。展開された列リスト中の列名は、引用符で囲まれています。これは、ベース・オブジェクトの列がもともと引用符付きで入力された可能性があり、問合せの構文を正しいものにするためには引用符が必要であることを示しています。

たとえば、dept ビューが次のように作成される場合を想定します。

```
CREATE VIEW dept AS SELECT * FROM scott.dept;
```

Oracle は、dept ビューを定義している問合せを次のように格納します。

```
SELECT "DEPTNO", "DNAME", "LOC" FROM scott.dept;
```

エラー付きで作成されたビューでは、ワイルドカードは展開されません。エラーなしでビューがコンパイルされると、定義された問合せのワイルドカードが展開されます。

エラー付きビューの作成

CREATE VIEW 文に構文エラーがない場合は、そのビューを定義している問合せを実行できなくても、Oracle はビューを作成できます。この場合、ビューは、エラー付きで作成されたと見なされます。たとえば、存在しない表や既存の表の無効な列を参照するビューを作成するとき、またはビューの所有者が必要な権限を持っていないときでも、ビューを作成し、データ・ディクショナリに登録できます。ただし、そのビューは使用できません。

エラー付きのビューを作成するには、CREATE VIEW 文の FORCE オプションを指定する必要があります。

```
CREATE FORCE VIEW AS ...;
```

デフォルトでは、エラー付きのビューは VALID としては作成されません。このようなビューを作成しようとすると、ビューがエラー付きで作成されたことを示すメッセージが返されます。エラー付きで作成されたビューの状態は、INVALID です。状況が変化して無効なビューの問合せが実行可能になると、ビューは再コンパイルされ、有効（使用可能）になります。条件の変更とビューに及ぼす影響の詳細は、21-23 ページの「オブジェクト依存性の管理」を参照してください。

結合ビューの更新

更新可能な結合ビュー（**変更可能な結合ビュー**と呼ぶこともあります）とは、SELECT 文の最上位の FROM 句に複数の表を含むビューで、WITH READ ONLY 句の制限を受けないものです。

注意： 結合ビューが更新可能かどうかについては、いくつかの制限と条件が影響します。これらの制限と条件については、『Oracle9*i* SQL リファレンス』の CREATE VIEW 文の説明を参照してください。

また、ビューが別のネストされたビュー上の結合である場合は、そのネストされたビューをトップレベル・ビューにマージ可能である必要があります。マージ可能なビューとマージ不可能なビューの詳細、およびオプティマイザによってビューを参照する文が最適化される方法の概要は、『Oracle9*i* データベース概要』および『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。

結合ビュー内の列が更新可能かどうかを示すデータ・ディクショナリ・ビューがあります。このようなビューについては、20-9 ページの「UPDATABLE_COLUMNS ビューの使用」を参照してください。

更新可能な結合ビューに関するルールは、次のとおりです。

規則	説明
一般規則	結合ビューに対する INSERT、UPDATE または DELETE 操作は、基礎となる実表を一度に 1 つしか変更できません。
UPDATE 規則	結合ビューの更新可能な列はすべて、**キー保存表**の列にマップする必要があります。キー保存表については、20-6 ページの「キー保存表」を参照してください。ビューの定義に WITH CHECK OPTION 句が使用されている場合は、すべての結合列および繰返し表の列はいずれも更新できません。
DELETE 規則	結合の中にキー保存表が 1 つしかない場合には、結合ビューから行を削除できます。ビューの定義に WITH CHECK OPTION 句が使用されていて、キー保存表が繰り返される場合は、そのビューから行を削除できません。
INSERT 規則	INSERT 文では、明示的にも暗黙的にも**非キー保存表**の列を参照しないでください。結合ビューの定義に WITH CHECK OPTION 句が使用されている場合は、INSERT 文を使用できません。

ここでは、これらの規則の例を示し，キー保存表について説明します。

それぞれの例は、表に主キーと外部キーを明示的に定義した場合、または一意索引を定義した場合にのみ動作します。次に、制約が適切に設定された emp と dept の表定義を示します。

```
CREATE TABLE dept (
      deptno        NUMBER(4) PRIMARY KEY,
      dname         VARCHAR2(14),
      loc           VARCHAR2(13));

CREATE TABLE emp (
      empno        NUMBER(4) PRIMARY KEY,
      ename        VARCHAR2(10),
      job          VARCHAR2(9),
      mgr          NUMBER(4),
      sal          NUMBER(7,2),
      comm         NUMBER(7,2),
      deptno       NUMBER(2),
      FOREIGN KEY (DEPTNO) REFERENCES DEPT(DEPTNO));
```

また、上の文の主キーと外部キーの制約を省略し、dept（deptno）に UNIQUE INDEX を作成した場合でも、後続の例は動作可能です。

次の文は、例で参照される emp_dept 結合ビューを作成します。

```
CREATE VIEW emp_dept AS
      SELECT emp.empno, emp.ename, emp.deptno, emp.sal, dept.dname, dept.loc
      FROM emp, dept
      WHERE emp.deptno = dept.deptno
         AND dept.loc IN ('DALLAS', 'NEW YORK', 'BOSTON');
```

キー保存表

キー保存表の概念は、結合ビューを更新する上での制限を理解するために重要です。表のすべてのキーが結合の結果のキーでもある場合、その表はキー保存になります。つまり、キー保存表とは、結合後もそのキーを保存している表のことです。

注意： 表をキー保存にするために，表の 1 つ以上のキーを選択する必要はありません。1 つ以上のキーを選択した場合に、そのキーが結合の結果のキーであれば十分です。

表のキー保存特性は、表内の実際のデータには依存しません。これは、そのスキーマの特性です。たとえば、emp 表で各部門に多くても 1 人の従業員しか含まれていない場合、emp と dept の結合の結果では deptno は一意ですが、dept はキー保存表ではありません。

emp_dept から SELECT ですべての行を選択すると、結果は次のようになります。

```
EMPNO      ENAME      DEPTNO  DNAME          LOC
---------- ---------- ------- -------------- -----------
      7782 CLARK           10 ACCOUNTING     NEW YORK
      7839 KING            10 ACCOUNTING     NEW YORK
      7934 MILLER          10 ACCOUNTING     NEW YORK
      7369 SMITH           20 RESEARCH       DALLAS
      7876 ADAMS           20 RESEARCH       DALLAS
      7902 FORD            20 RESEARCH       DALLAS
      7788 SCOTT           20 RESEARCH       DALLAS
      7566 JONES           20 RESEARCH       DALLAS
8 rows selected.
```

このビューでは、empno は emp 表のキーであり、結合結果のキーでもあるので、emp はキー保存表となります。dept がキー保存表でないのは、deptno が dept 表のキーであっても結合のキーではないためです。

DML 文と結合ビュー

一般規則では、結合ビューにおいて、UPDATE、INSERT または DELETE 文では、基礎となる実表を 1 つしか変更できません。以降の例は、UPDATE、DELETE および INSERT 文に固有の規則を示しています。

UPDATE 文 次の例は、emp_dept ビューを正常に変更する UPDATE 文を示したものです。

```
UPDATE emp_dept
     SET sal = sal * 1.10
     WHERE deptno = 10;
```

次に示す UPDATE 文は、emp_dept ビューには使用できません。

```
UPDATE emp_dept
     SET loc = 'BOSTON'
     WHERE ename = 'SMITH';
```

この文は dept 実表を変更しようとしますが、dept 表は emp_dept ビューのキー保存表でないため、エラー番号 ORA-01779「キー保存されていない表にマップする列は変更できません。」が出力され、文を実行できません。

一般に、結合ビューの更新可能な列はすべて、キー保存表の列にマップする必要があります。ビューの定義に WITH CHECK OPTION 句が使用されている場合は、すべての結合列およびビューで 2 回以上参照されている表から取得したすべての列はいずれも更新できません。

したがって、たとえば emp_dept ビューの定義に WITH CHECK OPTION が使用されている場合、次に示す UPDATE 文は失敗します。

```
UPDATE emp_dept
     SET deptno = 10
     WHERE ename = 'SMITH';
```

結合列を更新しようとするため、この文は失敗となります。

DELETE 文 結合の中にキー保存表が 1 つしかない場合には、結合ビューから削除できます。

次の DELETE 文は、emp_dept ビューに対して動作します。

```
DELETE FROM emp_dept
     WHERE ename = 'SMITH';
```

emp_dept ビューに対するこの DELETE 文は、実表 emp に対する DELETE 操作に変換でき、表 emp は結合ビュー内の唯一のキー保存表であるため、この文は有効です。

次のビューを作成した場合、e1 と e2 がともにキー保存表であるため、このビューに対して DELETE 操作を実行できません。

```
CREATE VIEW emp_emp AS
     SELECT e1.ename, e2.empno, deptno
     FROM emp e1, emp e2
     WHERE e1.empno = e2.empno;
```

ビューの定義に WITH CHECK OPTION 句が使用されていて、キー保存表が繰り返される場合は、そのビューから行を削除できません。

```
CREATE VIEW emp_mgr AS
     SELECT e1.ename, e2.ename mname
     FROM emp e1, emp e2
     WHERE e1.mgr = e2.empno
     WITH CHECK OPTION;
```

このビューはキー保存される表の自己結合を伴うので、このビューに対して削除を実行できません。

INSERT 文 emp_dept ビューに対する次の INSERT 文は正常に動作します。

```
INSERT INTO emp_dept (ename, empno, deptno)
   VALUES ('KURODA', 9010, 40);
```

変更されるキー保存実表は 1 つのみであり（emp）、40 は dept 表の有効な deptno であるため（つまり、emp 表に対する FOREIGN KEY 整合性制約を満たすため）、この文は動作します。

次の INSERT 文は、実表 emp に対する UPDATE が失敗するのと同じ理由で失敗します。つまり、77 という deptno が存在しないため、emp 表に対する FOREIGN KEY 整合性制約に違反します。

```
INSERT INTO emp_dept (ename, empno, deptno)
   VALUES ('KURODA', 9010, 77);
```

次の INSERT 文はエラー番号 ORA-01776「結合ビューを介して複数の実表を変更できません。」が出力されて失敗します。

```
INSERT INTO emp_dept (empno, ename, loc)
   VALUES (9010, 'KURODA', 'BOSTON');
```

INSERT は、暗黙的にも明示的にも、非キー保存表の列を参照できません。結合ビューの定義に WITH CHECK OPTION 句が使用されている場合は、その結合ビューに対して INSERT を実行できません。

UPDATABLE_COLUMNS ビューの使用

次の表に示すビューは、結合ビューを変更する際に利用できます。

ビュー	説明
DBA_UPDATABLE_COLUMNS	変更可能なすべての表とビューのすべての列が表示されます。
ALL_UPDATABLE_COLUMNS	ユーザーがアクセス可能で変更可能なすべての表とビューのすべての列が表示されます。
USER_UPDATABLE_COLUMNS	ユーザーのスキーマ内で変更可能なすべての表とビューのすべての列が表示されます。

次に、emp_dept ビュー内の更新可能な列を示します。

```
SELECT COLUMN_NAME, UPDATABLE
      FROM USER_UPDATABLE_COLUMNS
      WHERE TABLE_NAME = 'EMP_DEPT';

COLUMN_NAME                    UPD
------------------------------ ---
EMPNO                          YES
ENAME                          YES
DEPTNO                         YES
SAL                            YES
DNAME                          NO
LOC                            NO

6 rows selected.
```

ビューの変更

ALTER VIEW 文は、無効なビューを明示的に再コンパイルする場合にのみ使用します。ビューの定義を変更する場合は、20-10 ページの「ビューの置換」を参照してください。

ALTER VIEW 文を使用すると、実際に使用する前に再コンパイル・エラーを調べることができます。ビューの実表の 1 つを変更したとき、変更がビューまたはそれに依存する他のオブジェクトに影響しないようにするには、そのビューを明示的に再コンパイルします。

ALTER VIEW 文を使用するには、そのビューが自分のスキーマに含まれているか、または ALTER ANY TABLE システム権限を持っている必要があります。

ビューの削除

自分のスキーマにあるビューはすべて削除できます。別のユーザーのスキーマ内にあるビューを削除するには、DROP ANY VIEW システム権限が必要です。ビューを削除するには、DROP VIEW 文を使用します。たとえば、次の文は emp_dept ビューを削除します。

```
DROP VIEW emp_dept;
```

ビューの置換

ビューを置換するには、ビューの削除および作成に必要なすべての権限が必要です。ビューの定義を変更する場合は、そのビューを置換する必要があります。これは、ビューの定義を直接変更できないことを意味します。次の方法でビューを置換できます。

- ビューを削除してから再作成します。

  注意： ビューを削除するときに、ロールおよびユーザーに付与された対応するオブジェクト権限はすべて取り消されます。ビューを再作成してから、再度権限を付与してください。

- OR REPLACE オプションを含む CREATE VIEW 文によって、ビューを再定義します。OR REPLACE オプションは、ビューの現行の定義を置換し、現行のセキュリティ認可を保存します。たとえば、前述のように、sales_staff ビューを作成し、いくつかのオブジェクト権限をロールと他のユーザーに付与した場合を想定します。ただし、ここでは sales_staff ビューを再定義して、WHERE 句に指定されている部門番号を変更するものとします。この場合は、次の文によって、sales_staff ビューの現行バージョンを置換できます。

```
CREATE OR REPLACE VIEW sales_staff AS
     SELECT empno, ename, deptno
     FROM emp
     WHERE deptno = 30
     WITH CHECK OPTION CONSTRAINT sales_staff_cnst;
```

ビューを置換する前に、次の影響を検討してください。

- ビューを置換することによって、データ・ディクショナリ内のビュー定義が置換されます。ビューによって参照される基礎となるオブジェクトは影響を受けません。
- 前のビューには CHECK OPTION で制約を定義していたが、新しいビュー定義には指定しない場合、その制約は削除されます。
- 置換されたビューに依存するビューと PL/SQL プログラム・ユニットはすべて無効（使用不可）になります。Oracle でオブジェクトの依存性を管理する方法の詳細は、21-23 ページの「オブジェクト依存性の管理」を参照してください。

順序の管理

順序とは、複数のユーザーが一意の整数を生成するために使用できるデータベース・オブジェクトです。順序を使用して、主キー値を自動的に生成できます。ここでは、順序を管理する方法について説明します。この項の内容は、次のとおりです。

- 順序の作成
- 順序の変更
- 順序の削除

関連項目：

- 順序の詳細は、『Oracle9*i* データベース概要』を参照してください。
- CURRVAL および NEXTVAL 疑似列を使用して順序番号にアクセスする文の構文と詳細は、『Oracle9*i* SQL リファレンス』を参照してください。
- アプリケーションで順序を使用する方法の詳細は、『Oracle9*i* アプリケーション開発者ガイドー基礎編』を参照してください。

順序の作成

自分のスキーマに順序を作成するには、CREATE SEQUENCE システム権限が必要です。別のユーザーのスキーマ内に順序を作成するには、CREATE ANY SEQUENCE 権限が必要です。

順序を作成するには、CREATE SEQUENCE 文を使用します。たとえば、次の文は、emp 表の empno 列に対して従業員番号を生成するために使用する順序を作成します。

```
CREATE SEQUENCE emp_sequence
      INCREMENT BY 1
      START WITH 1
      NOMAXVALUE
      NOCYCLE
      CACHE 10;
```

CACHE オプションは順序番号により高速にアクセスできるように、順序番号の集合をメモリーに事前に割り当て、維持します。キャッシュ内の最後の順序番号が使用されると、別の順序の集合がキャッシュ内に読み込まれます。

順序番号の集合をキャッシュする場合に、順序番号がスキップされることがあります。たとえば、インスタンスが異常停止すると（たとえばインスタンス障害が発生したり、SHUTDOWN ABORT 文が発行されたりすると）、キャッシュされているが使用されていない順序番号は失われます。また、使用されても保存されなかった順序番号も失われます。さらに、エクスポートとインポートの後、Oracle がキャッシュされた順序番号をスキップすることもあります。詳細は、『Oracle9*i* データベース・ユーティリティ』を参照してください。

関連項目：

- Oracle Real Application Clusters 環境での順序番号のキャッシュによるパフォーマンス向上の詳細は、『Oracle9*i* Real Application Clusters 配置およびパフォーマンス』を参照してください。
- アプリケーションで順序を使用する方法と、順序番号をキャッシュする場合のパフォーマンスに関する考慮点については、『Oracle9*i* アプリケーション開発者ガイドー基礎編』を参照してください。

順序の変更

順序を変更するには、その順序が自分のスキーマに含まれているか、または ALTER ANY SEQUENCE システム権限を持っている必要があります。順序を変更することにより、順序番号の生成方法を定義するパラメータを変更できます。ただし、順序の開始番号は変更できません。順序の開始番号を変更するには、順序を削除してから、再作成します。

順序を変更するには、ALTER SEQUENCE 文を使用します。たとえば、次の文は、emp_sequence を変更します。

```
ALTER SEQUENCE emp_sequence
    INCREMENT BY 10
    MAXVALUE 10000
    CYCLE
    CACHE 20;
```

順序の削除

自分のスキーマ内の順序はどれでも削除できます。別のスキーマ内の順序を削除するには、DROP ANY SEQUENCE システム権限が必要です。不要になった順序は、DROP SEQUENCE 文を使用して削除できます。たとえば、次の文は order_seq 順序を削除します。

```
DROP SEQUENCE order_seq;
```

順序を削除すると、その定義がデータ・ディクショナリから削除されます。順序のシノニムはそのまま残りますが、参照時にエラーが返されます。

シノニムの管理

シノニムは、スキーマ・オブジェクトの別名です。シノニムは、オブジェクトの名前と所有者をマスクし、分散データベースのリモート・オブジェクトに位置透過性を提供することによって、あるレベルのセキュリティを提供します。また、SQL 文で使用できるため、データベース・ユーザーにとって SQL 文の複雑さが軽減されます。

シノニムを使用することで、基礎となるオブジェクトの名前変更または移動が可能になります。その場合、シノニムの再定義のみで、そのシノニムに基づくアプリケーションは変更しなくてもそのまま機能します。

パブリック・シノニムとプライベート・シノニムの両方を作成できます。**パブリック**・シノニムは PUBLIC という名前の特別なユーザー・グループによって所有され、データベース内のすべてのユーザーがアクセスできます。**プライベート**・シノニムは、特定のユーザーのスキーマ内に含まれ、そのユーザーとそのユーザーの権限受領者のみが使用できます。

ここでは、次のようなシノニム管理の情報について説明します。

- シノニムの作成
- シノニムの削除

関連項目：

- シノニムの詳細は、『Oracle9*i* データベース概要』を参照してください。
- 文の構文については、『Oracle9*i* SQL リファレンス』を参照してください。

シノニムの作成

自分のスキーマに新しいプライベート・シノニムを作成するには、CREATE SYNONYM システム権限が必要です。別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には、CREATE ANY SYNONYM 権限が必要です。パブリック・シノニムを作成するには、CREATE PUBLIC SYNONYM システム権限が必要です。

シノニムを作成するには、CREATE SYNONYM 文を使用します。基礎となるスキーマ・オブジェクトは、必要ありません。また、そのオブジェクトにアクセスする権限も不要です。次の文は、jward のスキーマに含まれる emp 表のパブリック・シノニム PUBLIC_EMP を作成します。

```
CREATE PUBLIC SYNONYM public_emp FOR jward.emp;
```

シノニムの削除

自分のスキーマ内のプライベート・シノニムはどれでも削除できます。別のユーザーのスキーマ内にあるプライベート・シノニムを削除するには、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には、DROP PUBLIC SYNONYM システム権限が必要です。

不要になったシノニムを削除するには、DROP SYNONYM 文を使用します。プライベート・シノニムを削除する場合は、PUBLIC キーワードを省略します。パブリック・シノニムを削除する場合は、PUBLIC キーワードを指定します。

たとえば、次の文はプライベート・シノニム emp を削除します。

```
DROP SYNONYM emp;
```

次の文は、パブリック・シノニム public_emp を削除します。

```
DROP PUBLIC SYNONYM public_emp;
```

シノニムを削除すると、その定義がデータ・ディクショナリから削除されます。削除したシノニムを参照するオブジェクトはすべて残ります。ただし、それらのオブジェクトは無効（使用不可）になります。シノニムの削除が他のスキーマ・オブジェクトに与える影響の詳細は、「オブジェクト依存性の管理」を参照してください。

ビュー、シノニムおよび順序の情報の表示

次のビューには、ビュー、シノニムおよび順序に関する情報が表示されます。

ビュー	説明
DBA_VIEWS ALL_VIEWS USER_VIEWS	DBA ビューには、データベース内のすべてのビューが表示されます。ALL ビューは、現行ユーザーがアクセスできるビューのみに制限されます。USER ビューは、現行ユーザーが所有するビューのみに制限されます。
DBA_SYNONYMS ALL_SYNONYMS USER_SYNONYMS	これらのビューには、シノニムが表示されます。
DBA_SEQUENCES ALL_SEQUENCES USER_SEQUENCES	これらのビューには、順序が表示されます。
DBA_UPDATABLE_COLUMNS ALL_UPDATABLE_COLUMNS USER_UPDATABLE_COLUMNS	これらのビューには、更新可能な結合ビューの列がすべて表示されます。

関連項目： これらのビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

21

スキーマ・オブジェクトの一般的な管理

この章では、複数のタイプのスキーマ・オブジェクトに共通する管理上の問題点について説明します。この章の内容は、次のとおりです。

- 一度の操作で複数の表やビューを作成する方法
- スキーマ・オブジェクトの名前変更
- 表、索引およびクラスタの分析
- 表とクラスタの切捨て
- トリガーの使用可能および使用禁止
- 整合性制約の管理
- オブジェクト依存性の管理
- オブジェクトの名前解決の管理
- データ・ディクショナリの記憶域パラメータの変更
- スキーマ・オブジェクト情報の表示

関連項目： この章で説明する SQL 文の構文、認可および制限の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

一度の操作で複数の表やビューを作成する方法

CREATE SCHEMA 文を使用すると、一度の操作で複数の表やビューを作成し、権限を付与できます。CREATE SCHEMA 文は、複数の表とビューの作成、および権限の付与を一度の操作で確実に行う必要がある場合に便利です。個々の表やビューの作成が失敗したり、権限の付与が失敗したりすると、文全体がロールバックされます。オブジェクトは作成されず、権限も付与されません。

CREATE SCHEMA 文に指定できるのは、CREATE TABLE、CREATE VIEW および GRANT 文のみです。指定した文を発行するための権限を持っている必要があります。この文を実行しても実際にスキーマが作成されるわけではありません。スキーマが作成されるのは、CREATE USER 文でユーザーを作成したときです。そのかわりに、この文はスキーマを移入します。

次の文は、2 つの表とそれらのデータを結合するビューを作成します。

```
CREATE SCHEMA AUTHORIZATION scott
    CREATE TABLE dept (
        deptno NUMBER(3,0) PRIMARY KEY,
        dname VARCHAR2(15),
        loc VARCHAR2(25)
    CREATE TABLE emp (
        empno NUMBER(5,0) PRIMARY KEY,
        ename VARCHAR2(15) NOT NULL,
        job VARCHAR2(10),
        mgr NUMBER(5,0),
        hiredate DATE DEFAULT (sysdate),
        sal NUMBER(7,2),
        comm NUMBER(7,2),
        deptno NUMBER(3,0) NOT NULL
        CONSTRAINT dept_fkey REFERENCES dept)
   CREATE VIEW sales_staff AS
        SELECT empno, ename, sal, comm
        FROM emp
        WHERE deptno = 30
        WITH CHECK OPTION CONSTRAINT sales_staff_cnst
        GRANT SELECT ON sales_staff TO human_resources;
```

CREATE SCHEMA 文は、STORAGE 句など、ANSI の CREATE TABLE 文と CREATE VIEW 文を拡張した Oracle 独自の機能をサポートしていません。

スキーマ・オブジェクトの名前変更

オブジェクト名を変更するには、そのオブジェクトが自分のスキーマ内に存在する必要があります。スキーマ・オブジェクトは、次のいずれかの方法で名前を変更できます。

- オブジェクトを削除して再作成する。
- RENAME 文を使用してオブジェクトの名前を変更する。

オブジェクトを削除して再作成する場合、そのオブジェクトに付与された権限はすべて失われます。オブジェクトを再作成するときに、再度権限を付与してください。

また、RENAME 文を使用して、表、ビュー、順序またはそれらのプライベート・シノニムの名前を変更することもできます。RENAME 文を使用すると、その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名前に引き継がれます。たとえば、次の文は sales_staff ビューの名前を変更します。

```
RENAME sales_staff TO dept_30;
```

注意： ストアド PL/SQL プログラム・ユニット、パブリック・シノニム、索引またはクラスタは名前を変更できません。これらのオブジェクトの名前を変更するには、削除してから再作成してください。

スキーマ・オブジェクト名を変更する前に、次のような影響について検討する必要があります。

- 名前を変更されたオブジェクトに依存しているビューと PL/SQL プログラム・ユニットはすべて無効になるため、次に使用する前に再コンパイルする必要があります。
- 名前を変更されたオブジェクトのシノニムを使用すると、必ずエラーが返されます。

関連項目： Oracle によるオブジェクト依存性の管理の詳細は、21-23 ページの「オブジェクト依存性の管理」を参照してください。

表、索引およびクラスタの分析

次の目的でスキーマ・オブジェクト（表、索引またはクラスタ）を分析します。

- 統計の収集と管理
- 記憶形式の妥当性の検証
- 表またはクラスタの移行行と連鎖行の識別

注意： オプティマイザ統計の収集には、ANALYZE ではなく DBMS_STATS パッケージを使用することをお薦めします。このパッケージを使用すると、統計をパラレルに収集し、パーティション・オブジェクトのグローバル統計を収集し、他の方法で統計収集を細かくチューニングできます。また、統計に依存するコストベース・オプティマイザの場合、最終的には DBMS_STATS で収集された統計のみが使用されます。このパッケージの詳細は、『Oracle9*i* 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、次のように、コストベース・オプティマイザに関連しない統計収集には、DBMS_STATS ではなく ANALYZE 文を使用する必要があります。

- VALIDATE または LIST CHAINED ROWS 句を使用する場合
- 空きリスト・ブロックの情報を収集する場合

この項の内容は、次のとおりです。

- 表、索引およびクラスタの統計の収集
- 表、索引、クラスタおよびマテリアライズド・ビューの妥当性チェック
- 表とクラスタの連鎖行のリスト

関連項目： 索引構成表の分析の詳細は、15-31 ページの「索引構成表の分析」を参照してください。

表、索引およびクラスタの統計の収集

DBMS_STATS パッケージまたは ANALYZE 文を使用して、表、索引またはクラスタの物理記憶特性の統計を収集できます。これらの統計はデータ・ディクショナリに格納され、オプティマイザで使用して、分析対象オブジェクトにアクセスする SQL 文に最も効率的な実行計画を選択できます。

オプティマイザ統計の収集にはより多様性のある DBMS_STATS パッケージを使用することをお薦めしますが、空きブロックや平均容量など、非オプティマイザ統計の収集には ANALYZE 文を使用する必要があります。

DBMS_STATS パッケージを使用した統計の計算

DBMS_STATS パッケージを使用すると、パラレル実行を利用した統計収集と統計の外部操作ができます。統計をデータ・ディクショナリ以外の表に格納し、オプティマイザに影響を与えずにその統計を操作できます。統計をデータベース間でコピーしたり、バックアップ・コピーを作成できます。

次の DBMS_STATS プロシージャにより、オプティマイザ統計を収集できます。

- GATHER_INDEX_STATS
- GATHER_TABLE_STATS
- GATHER_SCHEMA_STATS
- GATHER_DATABASE_STATS

関連項目：

- DBMS_STATS を使用してオプティマイザ統計を収集する方法は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。
- DBMS_STATS パッケージの詳細は、『Oracle9*i* 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。

ANALYZE 文を使用した統計の計算

次の文は、emp 表の統計を計算します。

```
ANALYZE TABLE emp COMPUTE STATISTICS;
```

次の問合せは、1,064 行のデフォルト統計サンプルを使用して、emp 表に関する統計を見積ります。

```
ANALYZE TABLE emp ESTIMATE STATISTICS;
```

Oracle が使用する統計サンプルを指定するには、ESTIMATE STATISTICS オプションとともに SAMPLE オプションを指定します。行や索引値の数、または表における行や索引値の割合（百分率）を示す整数を指定できます。次に、各オプションの指定例を示します。

```
ANALYZE TABLE emp
   ESTIMATE STATISTICS
      SAMPLE 2000 ROWS;

ANALYZE TABLE emp
   ESTIMATE STATISTICS
      SAMPLE 33 PERCENT;
```

どちらの場合でも、50 より大きい百分率や、そのオブジェクト内の行または索引値の数の 50% を超える数を指定すると、Oracle は見積りではなく、正確な統計を計算します。

ANALYZE 文を発行したときに、指定されたオブジェクトに対する統計がデータ・ディクショナリに含まれていると、新しい統計によってデータ・ディクショナリ内の古い統計が置き換えられます。

その他の統計計算方法

ANALYZE 文を効率的に実行できる PL/SQL パッケージのプロシージャがいくつかあります。索引には、次のようなタイプがあります。

- DBMS_UTILITY.ANALYZE_SCHEMA
- DBMS_UTILITY.ANALYZE_DATABASE
- DBMS_DDL.ANALYZE_OBJECT

これらのパッケージのプロシージャは、非オプティマイザ統計の収集にのみ使用してください。

関連項目：

- DBMS_UTILITY パッケージの詳細は、『Oracle9*i* 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。
- DBMS_DDL パッケージの詳細は、『Oracle9*i* 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、索引、クラスタおよびマテリアライズド・ビューの妥当性チェック

表、索引、クラスタまたはマテリアライズド・ビューの構造の整合性を検証するには、VALIDATE STRUCTURE オプションを指定した ANALYZE 文を使用します。構造が有効な場合、エラーは返されません。しかし、構造が破損していると、エラー・メッセージが出力されます。

たとえば、ハードウェアやその他のシステムに障害が発生した場合、索引が破損し、正しく機能しなくなる可能性があります。索引の妥当性をチェックすると、索引内のすべてのエントリが、対応付けられた表の正しい行を示しているかを確認できます。索引が破損した場合は、その索引を削除して再作成できます。

表、索引またはクラスタが破損している場合は、削除して再作成する必要があります。マテリアライズド・ビューが破損している場合は、完全リフレッシュを実行し、問題が修正されたことを確認します。問題が修正されない場合は、マテリアライズド・ビューを削除して再作成します。

次の文は、emp 表を分析します。

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

CASCADE オプションを含めると、オブジェクトとそれに関連するすべてのオブジェクト（索引など）の妥当性をチェックできます。次の文は、emp 表と、それに対応付けられているすべての索引の妥当性をチェックします。

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

妥当性をチェックするオブジェクトに対して DML を実行している間でも、オンラインで構造の妥当性をチェックするように指定できます。オブジェクトに影響を与える DML 文と並行して妥当性チェックを実行すると、パフォーマンスがわずかに低下しますが、これはオンラインで ANALYZE 文を実行できる柔軟性によって相殺されます。次の文は、emp 表と、それに対応付けられているすべての索引の妥当性をオンラインでチェックします。

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

表とクラスタの連鎖行のリスト

表またはクラスタの連鎖行と移行行は、LIST CHAINED ROWS 句を指定した ANALYZE 文を使用して検出できます。この文の結果は、LIST CHAINED ROWS 句によって返される情報を受け入れるために明示的に作成した指定の表に格納されます。この結果は、行を更新するための領域が十分であるかどうかを判断する上で役立ちます。たとえば、この情報を見れば、表やクラスタの PCTFREE が適切に設定されているかどうかがわかります。

CHAINED_ROWS 表の作成

ANALYZE...LIST CHAINED ROWS 文によって返されるデータを格納する表を作成するには、UTLCHAIN.SQL または UTLCHN1.SQL スクリプトを実行します。これらのスクリプト

は、Oracle に付属しています。これらのスクリプトは、スクリプトを実行するユーザーのスキーマ内に CHAINED_ROWS という名前の表を作成します。

注意： CHAINED_ROWS 表を作成するためにどちらのスクリプトを実行するかは、データベースの互換性レベルと分析する表のタイプによって決まります。詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

CHAINED_ROWS 表を作成した後、ANALYZE 文の INTO 句にその表を指定します。たとえば、次の文は、CHAINED_ROWS 表に、emp_dept クラスタ内の連鎖行に関する情報を含む行を挿入します。

```
ANALYZE CLUSTER emp_dept LIST CHAINED ROWS INTO CHAINED_ROWS;
```

関連項目： CHAINED_ROWS 表の詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

表内の移行行または連鎖行の解消

CHAINED_ROWS 表の情報を使用すると、既存表内にある移行行と連鎖行を低減または解消できます。これには、次の手順を使用します。

1. ANALYZE 文を使用して、移行行と連鎖行に関する情報を収集します。

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

2. 出力表を問い合せます。

   ```
   SELECT *
   FROM CHAINED_ROWS
   WHERE TABLE_NAME = 'ORDER_HIST';

   OWNER_NAME  TABLE_NAME  CLUST... HEAD_ROWID          TIMESTAMP
   ----------  ----------  -----... ------------------  ---------
   SCOTT       ORDER_HIST       ... AAAAluAAHAAAAA1AAA  04-MAR-96
   SCOTT       ORDER_HIST       ... AAAAluAAHAAAAA1AAB  04-MAR-96
   SCOTT       ORDER_HIST       ... AAAAluAAHAAAAA1AAC  04-MAR-96
   ```

   移行行または連鎖行がすべてリストされます。

3. 出力表の問合せによって、移行行または連鎖行が多数存在することがわかれば、以降の手順を実行して移行行を解消します。

4. 既存表と同じ列を持つ中間表を作成して、移行行と連鎖行を格納します。

   ```
   CREATE TABLE int_order_hist
      AS SELECT *
         FROM order_hist
         WHERE ROWID IN
   ```

```
         (SELECT HEAD_ROWID
            FROM CHAINED_ROWS
            WHERE TABLE_NAME = 'ORDER_HIST');
```

5. 既存表から移行行と連鎖行を削除します。

```
DELETE FROM order_hist
   WHERE ROWID IN
      (SELECT HEAD_ROWID
         FROM CHAINED_ROWS
         WHERE TABLE_NAME = 'ORDER_HIST');
```

6. 中間表の行を既存表に挿入します。

```
INSERT INTO order_hist
   SELECT *
   FROM int_order_hist;
```

7. 中間表を削除します。

```
DROP TABLE int_order_history;
```

8. 手順 1 で収集した情報を出力表から削除します。

```
DELETE FROM CHAINED_ROWS
   WHERE TABLE_NAME = 'ORDER_HIST';
```

9. 再度 ANALYZE 文を使用してから、出力表を問い合せます。
10. 出力表に表示された行は連鎖しています。連鎖行を解消するには、データ・ブロックのサイズを大きくする以外にありません。すべての状況において連鎖を回避することはほぼ不可能です。LONG 列や長い CHAR 列または VARCHAR2 列を持つ表では、ほとんどの場合、連鎖の発生は避けられません。

表とクラスタの切捨て

表（またはクラスタ）は残したままで、内容が完全に空になるように、表のすべての行またはクラスタ化表のグループ内のすべての行を削除できます。たとえば、月ごとのデータが含まれている表では、各月の終わりにそのデータをアーカイブした後で、表を空にする（すべての行を削除する）必要があります。

表からすべての行を削除するには、次の 3 通りの方法があります。

- DELETE 文を使用する。
- DROP 文と CREATE 文を使用する。
- TRUNCATE 文を使用する。

後続の項では、これらの方法について説明します。

DELETE の使用

DELETE 文を使用して表の行を削除できます。たとえば、次の文は emp 表からすべての行を削除します。

```
DELETE FROM emp;
```

DELETE 文を使用するときに、表またはクラスタに多数の行が存在していると、それらの行を削除する際に相当のシステム・リソースが使用されます。たとえば、CPU 時間、その表と対応付けられた索引の REDO ログ領域、ロールバック・セグメント領域などのリソースが必要です。また、各行が削除されるときに、トリガーが起動される場合があります。結果的に空になる表またはクラスタに事前に割り当てられた領域は、行を削除してもそのオブジェクトに対応付けられたままです。DELETE を使用すると削除する行を選択できますが、TRUNCATE と DROP の場合はオブジェクト全体が削除されます。

DROP と CREATE の使用

表を削除してから再作成します。たとえば、次の例では、emp 表を削除してから再作成しています。

```
DROP TABLE emp;
CREATE TABLE emp ( ... );
```

表やクラスタを削除してから再作成すると、対応付けられた索引、整合性制約およびトリガーもすべて削除され、削除された表またはクラスタ化表に依存するオブジェクトはすべて無効になります。また、削除された表またはクラスタ化表に対する権限付与もすべて削除されます。

TRUNCATE の使用

TRUNCATE 文を使用して表のすべての行を削除できます。たとえば、次の文は emp 表を切り捨てます。

```
TRUNCATE TABLE emp;
```

TRUNCATE 文は、表またはクラスタからすべての行を削除するための高速で効率的な方法を提供します。TRUNCATE 文はロールバック情報を生成せず、即時にコミットします。この文はデータ定義言語（DDL）であり、ロールバックできません。TRUNCATE 文を実行しても、切り捨てられる表に対応付けられている構造（制約およびトリガー）または認可は影響を受けません。また、TRUNCATE 文では、表を切り捨てた後で、表に現在割り当てられている領域を、その表を含む表領域に戻すかどうかも指定できます。

自分のスキーマにある表またはクラスタは切り捨てることができます。DROP ANY TABLE システム権限を持っているユーザーは、どのスキーマ内の表またはクラスタでも切り捨てることができます。

親キーを含む表またはクラスタ化表を切り捨てる際は、別の表で定義されているすべての参照外部キーを事前に使用禁止にする必要があります。自己参照制約を使用禁止にする必要はありません。

TRUNCATE 文によって表から行を削除する場合、表に対応付けられているトリガーは起動されません。また、TRUNCATE 文は、監査が使用可能の場合でも、DELETE 文に対応するどのような監査情報も生成しません。そのかわりに、発行された TRUNCATE 文に対して、単一の監査レコードが生成されます。監査の詳細は、第 26 章「データベース使用の監査」を参照してください。

ハッシュ・クラスタや、ハッシュ・クラスタまたは索引クラスタ内の表を個別に切り捨てることはできません。索引クラスタを切り捨てると、そのクラスタ内のすべての表からすべての行が削除されます。個々のクラスタ化表からすべての行を削除する必要がある場合は、DELETE 文を使用するか、または表を削除してから再作成してください。

TRUNCATE 文の REUSE STORAGE または DROP STORAGE オプションは、切り捨てた後に、表またはクラスタに現在割り当てられている領域を、その表を含む表領域に戻すかどうかを制御します。デフォルトのオプション DROP STORAGE は、文実行後の表に割り当てられたエクステントの数を MINEXTENTS の元の設定まで減らします。解放されたエクステントはシステムに戻され、他のオブジェクトによって使用できます。

一方、REUSE STORAGE オプションを指定すると、表またはクラスタに対して現在割り当てられているすべての領域は割り当てられたままになります。たとえば、次の文は emp_dept クラスタを切り捨てて、クラスタに対してそれまでに割り当てられているすべてのエクステントを、今後の挿入と削除のためにそのまま残します。

```
TRUNCATE CLUSTER emp_dept REUSE STORAGE;
```

REUSE または DROP STORAGE オプションは、対応付けられたすべての索引に適用されます。表またはクラスタを切り捨てると、対応付けられた索引もすべて切り捨てられます。切

り捨てられた表、クラスタまたは対応付けられた索引の記憶域パラメータは、切捨て後も変わりません。

トリガーの使用可能および使用禁止

データベース・トリガーとは、データベースに格納されており、表に行を追加するなどの特定の条件が発生したときにアクティブ化（起動）されるプロシージャです。トリガーを使用して Oracle の標準機能を補完することにより、データベース管理システムを高度にカスタマイズできます。たとえば、表に対する DML 操作を制限するトリガーを作成して、通常の営業時間中に発行された文のみ許可できます。

データベース・トリガーは、表、スキーマまたはデータベースに対応付けることができます。データベース・トリガーは、次の場合に暗黙的に起動されます。

- 対応付けられている表に対して DML 文（INSERT、UPDATE、DELETE）が実行されたとき
- データベースまたはスキーマ内のオブジェクトに対して、特定の DDL 文（ALTER、CREATE、DROP など）が実行されたとき
- 指定したデータベース・イベントが発生したとき（STARTUP、SHUTDOWN、SERVERERROR など）

このリストがすべてではありません。トリガーを起動する文とデータベース・イベントの詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

トリガーを作成するには、CREATE TRIGGER 文を使用します。トリガーは、トリガー・イベントの前（BEFORE）、後（AFTER）またはトリガー・イベントのかわりに（INSTEAD OF）起動するように定義できます。次の文は、表 scott.emp に対してトリガー scott.emp_permit_changes を作成します。このトリガーは、指定されたいずれかの文が実行される前に起動します。

```
CREATE TRIGGER scott.emp_permit_changes
     BEFORE
     DELETE OR INSERT OR UPDATE
     ON scott.emp
     .
pl/sql block
     .
```

後で DROP TRIGGER 文を発行し、トリガーをデータベースから削除できます。

トリガーには、次の 2 つのモードがあります。

- 使用可能

  トリガーが起動される文を発行したときに、トリガー制限（存在する場合）が TRUE と評価された場合は、使用可能トリガーによってトリガー本体が実行されます。デフォルトでは、トリガーを最初に作成したときに使用可能に設定されます。

- 使用禁止

  トリガーが起動される文を発行したときに、トリガー制限（存在する場合）が TRUE と評価された場合でも、使用禁止トリガーはトリガー本体を実行しません。

ALTER TABLE 文を使用してトリガーを使用可能または使用禁止にするには、表を所有しているか、表に対する ALTER オブジェクト権限があるか、または ALTER ANY TABLE システム権限があることが必要です。また、ALTER TRIGGER 文を使用してトリガーを個別に使用可能または使用禁止にするには、トリガーを所有しているか、または ALTER ANY TRIGGER システム権限を持っている必要があります。

関連項目：

- トリガーの詳細は、『Oracle9*i* データベース概要』を参照してください。
- トリガーの作成と管理に使用する SQL 文の構文、制限事項および認可要件については、『Oracle9*i* SQL リファレンス』を参照してください。
- トリガーの作成と使用の詳細は、『Oracle9*i* アプリケーション開発者ガイドー基礎編』を参照してください。

トリガーを使用可能にする方法

使用禁止のトリガーを使用可能にするには、ENABLE オプションを指定した ALTER TRIGGER 文を使用します。たとえば、inventory 表に定義されている reorder という使用禁止のトリガーを使用可能にするには、次の文を入力します。

```
ALTER TRIGGER reorder ENABLE;
```

ENABLE ALL TRIGGERS オプションを指定した ALTER TABLE 文を使用すれば、特定の表に定義されているトリガーをすべて使用可能にできます。たとえば、inventory 表に定義されているトリガーをすべて使用可能にするには、次の文を入力します。

```
ALTER TABLE inventory
    ENABLE ALL TRIGGERS;
```

トリガーを使用禁止にする方法

次の条件のいずれか 1 つが成り立つ場合は、一時的にトリガーを使用禁止にすることを検討してください。

- トリガーの参照するオブジェクトが使用可能でない場合
- 大規模なデータ・ロードを実行する際に、トリガーを起動せずに迅速にデータをロードする場合
- トリガーが適用される表にデータをロードする場合

トリガーを使用禁止にするには、DISABLE オプションを指定した ALTER TRIGGER 文を使用します。たとえば、inventory 表に定義されているトリガー reorder を使用禁止にするには、次の文を入力します。

```
ALTER TRIGGER reorder DISABLE;
```

DISABLE ALL TRIGGERS オプションを指定した ALTER TABLE 文を使用すれば、表に関連するトリガーをすべて同時に使用禁止にできます。たとえば、inventory 表に定義されているトリガーをすべて使用禁止にするには、次の文を入力します。

```
ALTER TABLE inventory
    DISABLE ALL TRIGGERS;
```

整合性制約の管理

整合性制約とは、表の 1 つ以上の列に格納される値を制限するルールです。CREATE TABLE 文または ALTER TABLE 文に制約句を指定することにより、その制約の影響を受ける列と、制約の条件を識別できます。

ここでは、制約の概念と、整合性制約の定義および管理に使用する SQL 文について説明します。この項の内容は、次のとおりです。

- 整合性制約の状態
- 定義時の整合性制約の設定
- 既存の整合性制約の変更、名前の変更または削除
- 制約チェックの遅延
- 制約例外のレポート
- 制約情報の表示

関連項目：

- 整合性制約の詳細は、『Oracle9*i* データベース概要』を参照してください。
- アプリケーションで整合性制約を使用する際の詳細と使用例については、『Oracle9*i* アプリケーション開発者ガイド－基礎編』を参照してください。

整合性制約の状態

制約は、使用可能（ENABLE）と使用禁止（DISABLE）のいずれの状態にするかを指定できます。制約が使用可能になっている場合は、データベース内でデータが入力または更新されるときにチェックされ、制約のルールに従っていないデータは入力されません。制約が使用禁止になっている場合は、ルールに従っていないデータでもデータベースに入力できます。

また、表の既存データが必ず制約に従うように指定できます（VALIDATE）。逆に NOVALIDATE を指定すると、既存データが制約に従っていることは保証されません。

表に定義されている整合性制約は、次のいずれかの状態にあります。

- ENABLE、VALIDATE
- ENABLE、NOVALIDATE
- DISABLE、VALIDATE
- DISABLE、NOVALIDATE

これらの状態の意味と組合せの結果の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。ここでは、いくつかの組合せの結果について説明します。

制約を使用禁止にする方法

整合性制約によって定義したルールを施行するには、その制約を常に使用可能にしておく必要があります。しかし、次のような場合は、パフォーマンス上の理由から、表の整合性制約を一時的に使用禁止にすることを検討してください。

- 表に大量のデータをロードする場合
- 表に大規模な変更を加えるバッチ操作を実行する場合（たとえば、既存の番号に 1000 を加えてすべての従業員番号を変更する場合）
- 1 つの表を一度にインポートまたはエクスポートする場合

これら 3 つの場合には、整合性制約を一時的に使用禁止にすることにより、操作のパフォーマンスを改善できます。これは、特にデータ・ウェアハウス構成に当てはまります。

制約が使用禁止である間は、その制約に違反するデータを入力できます。したがって、前述の操作を終了した後に、制約を必ず使用可能にする必要があります。

制約を使用可能にする方法

制約が使用可能になっている場合、制約に違反する行は表に挿入されません。しかし、制約が使用禁止の場合は、制約に違反する行でも表に挿入できます。このような行を制約の例外と呼びます。制約が妥当性チェックなしで使用可能な状態にある場合、制約が使用禁止になっていた間に入力された違反データはそのまま残っています。制約を妥当性チェック済みの状態にするためには、制約に違反する行を更新または削除する必要があります。

制約を使用可能にするときに、特定の整合性制約に対する例外を指定できます。21-21 ページの「制約例外のレポート」を参照してください。制約に違反している行はすべて EXCEPTIONS 表に格納され、検証できます。

妥当性チェックなしで使用可能な制約の状態

制約が妥当性チェックなしで使用可能な状態にある場合、それ以後の文はすべて、制約に従っているかどうかがチェックされます。ただし、表の既存データはチェックされません。妥当性チェックなしで使用可能な状態の制約を持つ表には、無効なデータが含まれる可能性がありますが、無効なデータを新たに追加することはできません。妥当性チェックなしで使用可能な制約は、有効なオンライン・トランザクション処理（OLTP）データをアップロードしているデータ・ウェアハウス構成で役立ちます。

制約を使用可能にする場合に、妥当性チェックは必ずしも必要ではありません。妥当性チェックなしで制約を使用可能にする方が、妥当性チェックありで制約を使用可能にするよりはるかに高速です。また、すでに使用可能になっている制約の妥当性をチェックする場合、妥当性チェック中の DML ロックは必要ありません（すでに使用禁止にした制約の妥当性をチェックする場合とは異なります）。これは、制約の規定により、妥当性チェック中に違反データが挿入されないことが保証されているためです。したがって、妥当性チェックなしで使用可能にすれば、制約を使用可能にすることによって一般に生じる停止時間を短縮できます。

整合性制約の状態：手順と利点

整合性制約の状態を次の順序で使用したときに、最も大きな利点が得られます。

1. 使用禁止状態
2. 操作（ロード、エクスポート、インポート）の実行
3. 妥当性チェックなしで使用可能な状態
4. 使用可能状態

制約をこの順序で使用する際の利点は、次のとおりです。

- ロックが保持されません。
- すべての制約を同時に使用可能状態にすることができます。
- 制約を使用可能にする処理がパラレルで行われます。
- 表での同時アクティビティを実行できます。

定義時の整合性制約の設定

CREATE TABLE 文または ALTER TABLE 文で整合性制約を定義するときに ENABLE/DISABLE 句を指定して、その制約を使用可能 / 使用禁止、妥当性チェックあり / 妥当性チェックなしの状態にすることができます。制約の定義時に ENABLE/DISABLE 句を指定しなければ、自動的にその制約は妥当性チェックありで使用可能な状態になります。

定義時に制約を使用禁止にする方法

次の CREATE TABLE 文と ALTER TABLE 文は、整合性制約を定義して、使用禁止にします。

```
CREATE TABLE emp (
    empno NUMBER(5) PRIMARY KEY DISABLE,   . . . ;

ALTER TABLE emp
   ADD PRIMARY KEY (empno) DISABLE;
```

整合性制約を定義して使用禁止にする ALTER TABLE 文は、表の行がその整合性制約に違反しているために失敗することはありません。制約のルールが施行されていないので、制約の定義が許可されます。

定義時に制約を使用可能にする方法

次の CREATE TABLE 文と ALTER TABLE 文は、整合性制約を定義して、使用可能にします。

```
CREATE TABLE emp (
    empno NUMBER(5) CONSTRAINT emp.pk PRIMARY KEY,   . . . ;

ALTER TABLE emp
    ADD CONSTRAINT emp.pk PRIMARY KEY (empno);
```

整合性制約を定義して使用可能にする ALTER TABLE 文は、表の行が整合性制約に違反しているために失敗する場合があります。この場合、その文はロールバックされ、制約定義は格納されず、使用可能にもなりません。

UNIQUE または PRIMARY KEY 制約を使用可能にすると、対応する索引が作成されます。

関連項目： 16-12 ページ「制約に対応付けられた索引の作成」

既存の整合性制約の変更、名前の変更または削除

ALTER TABLE 文では、制約を使用可能または使用禁止にする他、制約を変更または削除することもできます。制約を規定するために UNIQUE または PRIMARY KEY 索引が使用されている場合、その索引に対応する制約を削除または使用禁止にすると、明示的に指定しないかぎり、索引は削除されます。

使用可能な外部キーが PRIMARY キーまたは UNIQUE キーを参照している場合、PRIMARY キーまたは UNIQUE キーの制約またはその索引を削除したり使用禁止にしたりすることはできません。

使用可能状態の制約を使用禁止にする方法

次の文は、使用可能状態の整合性制約を使用禁止にします。2 番目の文では、対応する索引を保持するように指定しています。

```
ALTER TABLE dept
    DISABLE CONSTRAINT dname_ukey;

ALTER TABLE dept
    DISABLE PRIMARY KEY KEEP INDEX,
    DISABLE UNIQUE (dname, loc) KEEP INDEX;
```

次の文は、使用禁止状態の整合性制約を妥当性チェックなしで使用可能な状態にします。

```
ALTER TABLE dept
    ENABLE NOVALIDATE CONSTRAINT dname_ukey;

ALTER TABLE dept
    ENABLE NOVALIDATE PRIMARY KEY,
    ENABLE NOVALIDATE UNIQUE (dname, loc);
```

次の文は、使用禁止状態の整合性制約を使用可能にするか、または妥当性チェックありの状態にします。

```
ALTER TABLE dept
    MODIFY CONSTRAINT dname_key VALIDATE;
ALTER TABLE dept
    MODIFY PRIMARY KEY ENABLE NOVALIDATE;
```

次の文は、使用禁止状態の整合性制約を使用可能にします。

```
ALTER TABLE dept
    ENABLE CONSTRAINT dname_ukey;
ALTER TABLE dept
    ENABLE PRIMARY KEY,
    ENABLE UNIQUE (dname, loc);
```

UNIQUE キーまたは PRIMARY KEY 制約、およびすべての依存する FOREIGN KEY 制約を一度に使用禁止または削除するには、DISABLE 句または DROP 句の CASCADE オプションを使用します。たとえば、次の文は PRIMARY KEY 制約とこれに依存する FOREIGN KEY 制約を使用禁止にします。

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

制約名の変更

ALTER TABLE ... RENAME CONSTRAINT 文を使用すると、表に対する既存の制約の名前を変更できます。新しい制約名には、ユーザーの既存の制約名と競合しない名前を指定する必要があります。

次の文は、表 dept に対する dname_ukey 制約の名前を変更します。

```
ALTER TABLE dept
    RENAME CONSTRAINT dname_ukey TO dname_unikey;
```

制約名を変更しても、実表に対するすべての依存性は引き続き有効です。

RENAME CONSTRAINT 句を使用すると、制約のシステム生成名を変更できます。

制約の削除

整合性制約は、規定するルールが成立しなくなった場合、またはその制約が不要になった場合に削除できます。制約を削除するには、ALTER TABLE 文で次のいずれかの句を指定します。

- DROP PRIMARY KEY
- DROP UNIQUE
- DROP CONSTRAINT

次の 2 つの文は、整合性制約を削除します。2 番目の文は、PRIMARY KEY 制約に対応する索引を保持します。

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

ALTER TABLE emp
    DROP PRIMARY KEY KEEP INDEX,
    DROP CONSTRAINT dept_fkey;
```

FOREIGN KEY が UNIQUE または PRIMARY KEY を参照している場合は、DROP 文に CASCADE CONSTRAINTS 句を指定しないかぎり、制約を削除できません。

制約チェックの遅延

Oracle が制約をチェックしたときに制約が満たされていない場合は、エラーが通知されます。制約の妥当性チェックは、トランザクションが終わるまで遅延できます。

SET CONSTRAINTS 文を発行すると、トランザクションの実行中、または別の SET CONSTRAINTS 文によってモードが再設定されるまで、SET CONSTRAINTS モードが継続します。

注意：

- SET CONSTRAINT 文は、トリガーの内部では発行できません。
- 遅延可能な一意キーと主キーは、必ず非一意索引を使用する必要があります。

すべての制約を遅延に設定する方法

データ操作に使用するアプリケーションでは、実際にデータの処理を始める前にすべての制約を遅延に設定する必要があります。遅延可能制約をすべて遅延に設定するには、次の DML 文を使用します。

```
SET CONSTRAINTS ALL DEFERRED;
```

注意： SET CONSTRAINTS 文は、現行のトランザクションにのみ適用されます。制約を作成したときに指定したデフォルトは、その制約が存在するかぎり保持されています。ALTER SESSION SET CONSTRAINTS 文は、現行のセッションにしか適用されません。

コミットのチェック（オプション）

COMMIT の発行直前に SET CONSTRAINTS ALL IMMEDIATE 文を発行することにより、制約違反をチェックできます。制約になんらかの問題があると、この文は失敗し、エラーの原因となっている制約が識別されます。制約違反のままコミットすると、トランザクションはロールバックされ、エラー・メッセージが返されます。

制約例外のレポート

制約の妥当性チェック時に例外が存在すると、エラーが返され、整合性制約は妥当性チェックなしの状態のままになります。整合性制約の例外が存在しているために文が正常に実行されない場合、文はロールバックされます。例外が存在している場合は、制約の例外をすべて更新または削除するまで、制約の妥当性はチェックできません。

どの行が違反しているかを判断するために、CREATE TABLE 文を使用することはできません。整合性制約に違反している行を判断するには、ENABLE 句に EXCEPTIONS オプションを指定して ALTER TABLE 文を発行します。EXCEPTIONS オプションにより、例外を含むすべての行の ROWID、表所有者、表名および制約名が指定した表に格納されます。

制約を使用可能にする前に、ENABLE 句の EXCEPTIONS オプションからの情報を格納する適切な例外レポート表を作成する必要があります。例外表を作成するには、UTLEXCPT.SQL スクリプトまたは UTLEXPT1.SQL スクリプトを実行します。

注意： EXCEPTIONS 表を作成するためにどちらのスクリプトを実行するかは、データベースの互換性レベルと分析する表のタイプによって決まります。詳細は、『Oracle9*i* SQL 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どちらを使用しても、EXCEPTIONS という名前の表が作成されます。また、スクリプトを変更して再実行すると、新たに別の名前の例外表を作成できます。

次の文は、dept 表の PRIMARY KEY を検証します。例外が存在すると、EXCEPTIONS 表に情報が挿入されます。

```
ALTER TABLE dept ENABLE PRIMARY KEY EXCEPTIONS INTO EXCEPTIONS;
```

dept 表に重複する主キー値が存在し、dept の PRIMARY KEY 制約の名前が sys_c00610 である場合は、この文を実行することにより、次のような行が EXCEPTIONS 表に挿入されます。

```
SELECT * FROM EXCEPTIONS;

ROWID               OWNER      TABLE_NAME      CONSTRAINT
------------------  ---------  --------------  -----------
AAAAZ9AABAAABvqAAB  SCOTT      DEPT            SYS_C00610
AAAAZ9AABAAABvqAAG  SCOTT      DEPT            SYS_C00610
```

次の例のように、例外レポート表とマスター表の行を結合した詳細な問合せを実行すれば、特定の制約に違反している実際の行を表示できます。

```
SELECT deptno, dname, loc FROM dept, EXCEPTIONS
    WHERE EXCEPTIONS.constraint = 'SYS_C00610'
    AND dept.rowid = EXCEPTIONS.row_id;
```

```
DEPTNO     DNAME             LOC
---------- --------------    -----------
10         ACCOUNTING        NEW YORK
10         RESEARCH          DALLAS
```

制約に違反している行はすべて更新するか、または制約を含む表から削除する必要があります。例外を更新する場合は、制約に違反する値を、制約を満たす値または NULL に変更します。マスター表の行を更新または削除した後、以後取得する例外レポートとの混同を避けるために、例外レポート表の例外に対応する行は削除します。マスター表と例外レポート表を更新する文は、トランザクションの一貫性を保証するために、同じトランザクション内で実行してください。

前述の例の例外を訂正するために、次のトランザクションを発行できます。

```
UPDATE dept SET deptno = 20 WHERE dname = 'RESEARCH';
DELETE FROM EXCEPTIONS WHERE constraint = 'SYS_C00610';
COMMIT;
```

例外管理の最終的な目的は、例外レポート表の例外をすべて取り除くことにあります。

注意： 制約が使用禁止になっている表の現在の例外を訂正している間に、他のユーザーが新しい例外を作成する文を発行する可能性があります。これを避けるには、例外を取り除く前に、制約を妥当性チェックなしで使用可能な状態に変更します。

関連項目： EXCEPTIONS 表の詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

制約情報の表示

表の制約定義を表示し、制約で指定されている列を識別できるように、次のビューが用意されています。

ビュー	説明
DBA_CONSTRAINTS ALL_CONSTRAINTS USER_CONSTRAINTS	DBA ビューには、データベース内のすべての制約定義が表示されます。ALL ビューには、現行ユーザーがアクセス可能な制約定義が表示されます。USER ビューには、現行ユーザーが所有している制約定義が表示されます。
DBA_CONS_COLUMNS ALL_CONS_COLUMNS USER_CONS_COLUMNS	DBA ビューには、制約で指定されているデータベース内のすべての列が表示されます。ALL ビューには、制約で指定されていて、現行ユーザーがアクセス可能な列のみが表示されます。USER ビューには、制約で指定されていて、現行ユーザーが所有している列のみが表示されます。

関連項目： これらのビューの列の詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

オブジェクト依存性の管理

ここでは、様々なオブジェクト依存性について説明します。この項の内容は、次のとおりです。

- 手動によるビューの再コンパイル
- 手動によるプロシージャとファンクションの再コンパイル
- 手動によるパッケージの再コンパイル

最初に、表 21-1 を検討します。この表は、オブジェクトが依存している他のオブジェクトの変更によって、そのオブジェクトがどのような影響を受けるかを示したものです。

表 21-1 オブジェクトの状態に影響を与える操作

操作	操作後のオブジェクトの状態	操作後の依存オブジェクトの状態
CREATE [TABLE | SEQUENCE | SYNONYM]	エラーがない場合、VALID になる	変化なし[1]
ALTER TABLE [ADD | RENAME | MODIFYcolumns] RENAME [TABLE | SEQUENCE | SYNONYM | VIEW]	エラーがない場合、VALID になる	INVALID
DROP [TABLE | SEQUENCE | SYNONYM | VIEW | PROCEDURE | FUNCTION | PACKAGE]	なし（オブジェクトが削除される）	INVALID
CREATE [VIEW | PROCEDURE][2]	エラーがない場合は VALID、構文エラーまたは認可エラーの場合は INVALID	変化なし[1]
CREATE OR REPLACE [VIEW | PROCEDURE][2]	エラーがない場合は VALID、構文エラーまたは認可エラーの場合は INVALID	INVALID
REVOKE *object privilege*[3] ON *object* TO | FROM *user*	変化なし	オブジェクトに依存するユーザーのすべてのオブジェクトが INVALID[3] になる
REVOKE *object privilege*[3] ON *object* TO | FROM PUBLIC	変化なし	オブジェクトに依存するデータベース内のすべてのオブジェクトが INVALID[3] になる

表 21-1 オブジェクトの状態に影響を与える操作 （続き）

操作	操作後の オブジェクトの状態	操作後の依存 オブジェクトの状態
REVOKE *system privilege*[4] TO|FROM *user*	変化なし	ユーザーのすべてのオブジェクトが INVALID[4] になる
REVOKE *system privilege*[4] TO|FROM PUBLIC	変化なし	データベース内のすべてのオブジェクトが INVALID[4] になる

[1] **操作の前にオブジェクトが存在していなかった場合は、依存オブジェクトが** INVALID **になることがあります。**
[2] **スタンドアロンのプロシージャ、ファンクション、パッケージおよびトリガー。**
[3] SELECT、INSERT、UPDATE、DELETE、EXECUTE **などの** DML **オブジェクト権限のみ。妥当性の再チェックでの再コンパイルは必要ありません。**
[4] SELECT、INSERT、UPDATE、DELETE ANY TABLE、EXECUTE ANY PROCEDURE **などの** DML **システム権限のみ。妥当性の再チェックでの再コンパイルは必要ありません。**

無効なビューまたは PL/SQL プログラム・ユニットは、次に使用されるときに自動的に再コンパイルされます。また、適切な SQL 文に COMPILE 句を指定して、ビューまたはプログラム・ユニットを強制的に再コンパイルすることもできます。強制コンパイルは、通常、依存しているビューまたはプログラム・ユニットが無効であることがわかっていて、現在使用していない場合に、エラーを検査する目的で使用します。

このような場合は、ビューまたはプログラム・ユニットが実行されないかぎり、自動再コンパイルは実行されません。無効な依存オブジェクトを識別するには、ビュー USER_OBJECTS/ALL_OBJECTS/DBA_OBJECTS を問い合せます。

手動によるビューの再コンパイル

ビューを手動で再コンパイルするには、そのビューが自分のスキーマに含まれているか、または ALTER ANY TABLE システム権限を持っている必要があります。ビューを再コンパイルするには、COMPILE 句を指定した ALTER VIEW 文を使用します。次の文は、自分のスキーマ内にある emp_dept ビューを再コンパイルします。

```
ALTER VIEW emp_dept COMPILE;
```

手動によるプロシージャとファンクションの再コンパイル

スタンドアロン・プロシージャを手動で再コンパイルするには、そのプロシージャが自分のスキーマに含まれているか、または ALTER ANY PROCEDURE システム権限を持っている必要があります。スタンドアロン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には、COMPILE 句を指定した ALTER PROCEDURE/FUNCTION 文を使用します。次の文は、自分のスキーマ内にあるストアド・プロシージャ update_salary を再コンパイルします。

```
ALTER PROCEDURE update_salary COMPILE;
```

手動によるパッケージの再コンパイル

パッケージを手動で再コンパイルするには、そのパッケージが自分のスキーマに含まれているか、または ALTER ANY PROCEDURE システム権限を持っている必要があります。パッケージ本体、またはパッケージ本体とパッケージ仕様部の両方を再コンパイルするには、COMPILE 句を指定した ALTER PACKAGE 文を使用します。次の文は、パッケージ acct_mgmt の本体のみを再コンパイルします。

```
ALTER PACKAGE acct_mgmt COMPILE BODY;
```

次の例では、パッケージ acct_mgmt の本体と仕様部を再コンパイルしています。

```
ALTER PACKAGE acct_mgmt COMPILE PACKAGE;
```

オブジェクトの名前解決の管理

SQL 文で参照されるオブジェクト名は、ピリオドで区切られた複数の断片から構成できます。ここでは、Oracle でオブジェクト名を解決する方法を説明します。

1. Oracle は、SQL 文で参照される名前の最初の断片を識別しようとします。たとえば、scott.emp の最初の断片は scott です。断片が 1 つしか存在しない場合、その断片は最初の断片とみなされます。

   a. 現行スキーマ内で、オブジェクト名の最初の断片に一致するオブジェクトが検索されます。そのようなオブジェクトが見つからない場合は、手順 b に進みます。

   b. オブジェクト名の最初の断片に一致するパブリック・シノニムが検索されます。そのようなシノニムが見つからない場合は、手順 c に進みます。

   c. オブジェクト名の最初の断片に一致するスキーマが検索されます。スキーマが検出されたら手順 b に戻り、次はオブジェクト名の 2 番目の断片を使用して、識別されたスキーマ内が検索されます。2 番目の断片が識別されたスキーマ内のオブジェクトに一致しない場合、または 2 番目の断片がない場合は、エラーが返されます。

   手順 c でスキーマが検出されない場合、そのオブジェクトは識別できず、エラーが返されます。

2. スキーマ・オブジェクトが識別されました。SQL 文で与えられた名前の残りの断片は、見つかったオブジェクトの有効な部分に一致する必要があります。たとえば、名前が scott.emp.deptno で、scott がスキーマとして識別され、emp が表として識別された場合は、（emp が表であるため）deptno は列に対応する必要があります。また、emp がパッケージとして識別された場合、deptno はそのパッケージのパブリック定数、変数、プロシージャまたはファンクションに対応する必要があります。

分散データベースにおいて、明示的またはシノニム内で間接的にグローバル・オブジェクト名が使用されているとき、ローカルの Oracle はローカルで参照を解決します。たとえば、シノニムをリモート表のグローバル・オブジェクト名として解決します。部分的に解決された文はリモート・データベースに転送され、前述の手順に従って、リモートの Oracle でオブジェクトの解決が行われます。

Oracle による参照の解決方法の関係で、あるオブジェクトが、他のオブジェクトが存在しないことに依存している可能性があります。この状況が発生するのは、依存するオブジェクトが使用している参照の解析方法が、他のオブジェクトが存在しているときには異なる場合です。たとえば、次のような場合を考えてみます。

- 現時点では、company スキーマに表 emp が含まれています。
- company.emp に対してパブリック・シノニム emp が作成され、company.emp に対する SELECT 権限が PUBLIC ロールに付与されます。
- jward スキーマには、表またはプライベート・シノニム emp は含まれていません。
- ユーザー jward が、次の文を使用して自分のスキーマにビューを作成します。

```
CREATE VIEW dept_salaries AS
     SELECT deptno, MIN(sal), AVG(sal), MAX(sal) FROM emp
     GROUP BY deptno
     ORDER BY deptno;
```

jward が dept_salaries ビューを作成すると、emp への参照は、jward.emp を表、ビューまたはプライベート・シノニムとして検索し、いずれも見つからない場合はパブリック・シノニム emp として検索して見つけることで解決されます。その結果、jward.dept_salaries は、jward.emp が存在しないことと、public.emp が存在することに依存していることがわかります。

ここで、jward が次の文を使用して自分のスキーマに新しいビュー emp を作成するとします。

```
CREATE VIEW emp AS
     SELECT empno, ename, mgr, deptno
     FROM company.emp;
```

jward.emp の構造が company.emp とは異なることに注意してください。

オブジェクト定義内で参照を解決するときに、Oracle は新しい依存オブジェクトが存在しないオブジェクト（スキーマ・オブジェクト）に対して持っている依存性に内部的に注目します。このスキーマ・オブジェクトが存在する場合は、オブジェクトの定義の解析が変化しま

す。存在しないオブジェクトを後で作成する場合は、この種の依存性に注意する必要があります。存在しないオブジェクトを作成する場合は、依存オブジェクトを再コンパイルして検証できるように、すべての依存オブジェクトを無効にする必要があります。また、依存するすべてのファンクション・ベース索引を使用禁止としてマークする必要があります。

したがって、前述の例では、jward.emp が作成されると、jward.dept_salaries は jward.emp に依存するため無効になります。その後、jward.dept_salaries が使用されると、Oracle はビューの再コンパイルを試みます。emp への参照を解決するときに、jward.emp が見つかります（public.emp は参照先のオブジェクトではなくなっています）。jward.emp には sal 列がないため、ビューを置換するときにエラーが見つかり、ビューは無効のままになります。

要約すると、存在しないオブジェクトを後で作成する場合は、オブジェクトの解決中にチェックされる存在しないオブジェクトへの依存性を管理する必要があります。

データ・ディクショナリの記憶域パラメータの変更

データベースが非常に大規模であるか、または著しい数のオブジェクト、表の列、制約定義、ユーザーなどの定義を含んでいる場合は、ある時点でデータ・ディクショナリを構成する表が追加のエクステントを取得できなくなる可能性があります。たとえば、データ・ディクショナリ表が追加のエクステントを必要としていても、SYSTEM 表領域内に十分な連続領域がない場合です。このような場合は、オブジェクトを保持する表領域に十分な領域があるように見えても、新しいオブジェクトを作成できません。こうした状況を改善するために、ユーザーが作成したセグメントの記憶域設定を変更する場合と同じ方法で、基礎となるデータ・ディクショナリ表の記憶域パラメータを変更できます。これにより、データ・ディクショナリ表に追加のエクステントを割り当てることが可能になります。たとえば、データ・ディクショナリ表の NEXT または PCTINCREASE の値を調整できます。

注意： データ・ディクショナリ・オブジェクトの記憶域設定を変更するときは、注意してください。不適切な設定を選択すると、データ・ディクショナリの構造が損傷し、データベース全体の再作成を余儀なくされるおそれがあります。たとえば、データ・ディクショナリ表 USER$ の PCTINCREASE を 0（ゼロ）、NEXT を 2KB に設定すると、この表はすぐにセグメントの最大エクステント数に到達します。こうなると、データベース全体をエクスポートし、再作成して、インポートしないかぎり、新しいユーザーやロールを作成できません。

ここでは、データ・ディクショナリの記憶域パラメータについて説明します。この項の内容は、次のとおりです。

- データ・ディクショナリの構造
- データ・ディクショナリの記憶域の変更が必要なエラー

データ・ディクショナリの構造

次の表とクラスタには、ユーザーがデータベース内に作成したすべてのオブジェクトの定義が格納されています。

表またはクラスタ	定義の対象
SEG$	データベース内に定義されているセグメント（一時セグメントを含む）。
OBJ$	データベース内のユーザー定義オブジェクト（クラスタ化表を含む）。I_OBJ1 と I_OBJ2 による索引付き。
UNDO$	データベース内に定義されているロールバック・セグメント。I_UNDO1 による索引付き。
FET$	どのセグメントにも割り当てられていない使用可能な空きエクステント。
UET$	セグメントに割り当てられているエクステント。
TS$	データベース内に定義されている表領域。
FILE$	データベースを構成するファイル。I_FILE1 による索引付き。
FILEXT$	AUTOEXTEND オプションがオンに設定されているデータ・ファイル。
TAB$	データベース内に定義されている表（クラスタ化表を含む）。I_TAB1 による索引付き。
CLU$	データベース内に定義されているクラスタ。
IND$	データベース内に定義されている索引。I_IND1 による索引付き。
ICOL$	索引が定義されている列（コンポジット索引内の各列の個別エントリを含む）。I_ICOL1 による索引付き。
COL$	データベース内の表に定義されている列。I_COL1 と I_COL2 による索引付き。
CON$	データベース内に定義されている制約（制約の所有者に関する情報を含む）。I_CON1 と I_CON2 による索引付き。
CDEF$	CON$ 内の制約の定義。I_CDEF1、I_CDEF2 および I_CDEF3 による索引付き。
CCOL$	制約が定義されている列（コンポジット・キー内の各列の個別エントリを含む）。I_CCOL1 による索引付き。
USER$	データベース内に定義されているユーザーとロール。I_USER1 による索引付き。
TSQ$	ユーザーの表領域割当て制限（各ユーザーに定義されている表領域割当て制限ごとに 1 エントリを含む）。

表またはクラスタ	定義の対象
C_OBJ#	TAB$、CLU$、ICOL$、IND$ および COL$ を含むクラスタ。I_OBJ# による索引付き。
C_TS#	FET$、TS$ および FILE$ を含むクラスタ。I_TS# による索引付き。
C_USER#	USER$ と TSQ$ を含むクラスタ。I_USER# による索引付き。
C_COBJ#	CDEF$ と CCOL$ を含むクラスタ。I_COBJ# による索引付き。

すべてのデータ・ディクショナリ・セグメントの中で、変更が必要となる可能性があるセグメントは次のとおりです。

表またはクラスタ	コメント
C_TS#	データベース内の空き領域の断片化が進んでいる場合
C_OBJ#	表に多くの索引または多くの列がある場合
CON$, C_COBJ#	整合性制約を頻繁に使用する場合
C_USER#	データベースに多数のユーザーが定義されている場合

クラスタ化表については、その表の記憶域設定ではなく、クラスタの記憶域設定を変更する必要があります。

データ・ディクショナリの記憶域の変更が必要なエラー

ユーザーが新しいオブジェクトを作成するときに、Oracle がデータ・ディクショナリに追加のエクステントを割り当てようとして、それが失敗すると、エラーが返されます。次のエラー・メッセージは、この種の問題を示しています。

ORA-1653 表 *name* を拡張できません（*num* 分、表領域 *name*）。

このエラー・メッセージが出力されたときに、変更しようとしていたセグメント（表やロールバック・セグメントなど）がその定義に指定された制限に到達していない場合は、その定義を含むオブジェクトの記憶域設定をチェックしてください。

たとえば、表に対して新しい PRIMARY KEY 制約を定義しようとしたときに、キーの索引を作成できるだけの領域が存在しているにもかかわらず、ORA-1653 が出力された場合は、CON$ または C_COBJ# に別のエクステントを割り当てることができるかをチェックします。そのためには、DBA_SEGMENTS を問い合せます。別のセグメントを割り当てることができない場合は、CON$ または C_COBJ# の記憶域パラメータの変更を検討します。21-35 ページの「例 7: 追加のエクステントを割り当てることのできないセグメントの表示」を参照してください。

スキーマ・オブジェクト情報の表示

Oracle には、スキーマ・オブジェクトに関する情報を表示するためのデータ・ディクショナリ・ビューと PL/SQL パッケージが用意されています。特定のスキーマ・オブジェクトに固有のビューとパッケージは、このマニュアルの各オブジェクトに関連した章に記載されています。ここでは、汎用的な性質を持ち、複数のスキーマ・オブジェクトに適用されるビューとパッケージについて説明します。

PL/SQL パッケージを使用したスキーマ・オブジェクト情報の表示

オラクル社が提供する PL/SQL パッケージを使用すると、スキーマ・オブジェクトに関する次の情報を取得できます。

パッケージとプロシージャ / ファンクション	説明
DBMS_METADATA.GET_DDL	スキーマ・オブジェクトに関するメタデータを（オブジェクトの作成に使用する DDL の形式で）取得します。
次のパッケージ・プロシージャは、スキーマ・オブジェクト内のスペース使用と空きブロックに関する情報を提供します。	
DBMS_SPACE.UNUSED_SPACE	オブジェクト（表、索引またはクラスタ）の未使用領域に関する情報を返します。
DBMS_SPACE.FREE_BLOCKS	セグメントの空き領域が空きリストで管理されている（つまり、セグメント領域管理が MANUAL である）オブジェクト（表、索引またはクラスタ）の空きデータ・ブロックに関する情報を返します。
DBMS_SPACE.SPACE_USAGE	セグメント領域管理が AUTO であるオブジェクト（表、索引またはクラスタ）の空きデータ・ブロックに関する情報を返します。

次に、これらのパッケージの使用例を示します。

関連項目： PL/SQL パッケージの詳細は、『Oracle9*i* 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 1: DBMS_METADATA パッケージの使用

DBMS_METADATA パッケージは、スキーマ・オブジェクトの完全な定義を取得できる強力なツールです。このパッケージを使用すると、あるオブジェクトのすべての属性を 1 回のパスで取得できます。オブジェクトは、その作成（再作成）に使用できる DDL で表されます。

この例では、GET_DDL ファンクションを使用して、現行スキーマ内にあるすべての表の DDL をフェッチし、ネストした表とオーバーフロー・セグメントを除外しています。また、

DDL で記憶域句が返されないようにするため、SET_TRANSFORM_PARAM（ハンドル値として「現行セッション用」を意味する DBMS_METADATA.SESSION_TRANSFORM をとる）を使用してそれを指定しています。セッション・レベルの変換パラメータは、最後にデフォルトにリセットされています。変換パラメータ値は、いったん設定すると、明示的にデフォルトにリセットされるまで有効です。

```
EXECUTE DBMS_METADATA.SET_TRANSFORM_PARAM(
     DBMS_METADATA.SESSION_TRANSFORM,'STORAGE',false);
SELECT DBMS_METADATA.GET_DDL('TABLE',u.table_name)
     FROM USER_ALL_TABLES u
     WHERE u.nested='NO'
     AND (u.iot_type is null or u.iot_type='IOT');
EXECUTE DBMS_METADATA.SET_TRANSFORM_PARAM(
     DBMS_METADATA.SESSION_TRANSFORM,'DEFAULT');
```

関連項目： DBMS_METADATA パッケージの使用に関する詳細とその他の使用例については、『Oracle9*i* XML Developer's Kit ガイド - XDK』を参照してください。

例 2: DBMS_SPACE.UNUSED_SPACE の使用

次の SQL*Plus の例では、DBMS_SPACE パッケージを使用して、未使用領域情報を取得しています。

```
SQL> VARIABLE total_blocks NUMBER
SQL> VARIABLE total_bytes NUMBER
SQL> VARIABLE unused_blocks NUMBER
SQL> VARIABLE unused_bytes NUMBER
SQL> VARIABLE lastextf NUMBER
SQL> VARIABLE last_extb NUMBER
SQL> exec DBMS_SPACE.UNUSED_SPACE('SCOTT', 'EMP', 'TABLE', :total_blocks, -
>    :total_bytes,:unused_blocks, :unused_bytes, :lastextf, -
>    :last_extb, :lastusedblock);

PL/SQL procedure successfully completed.

SQL> PRINT

TOTAL_BLOCKS
------------
           5

TOTAL_BYTES
-----------
      10240

...
```

```
LASTUSEDBLOCK
-------------
            3
```

ビューを使用したスキーマ・オブジェクト情報の表示

次のビューには、スキーマ・オブジェクトに関する情報が表示されます。

ビュー	説明
DBA_OBJECTS ALL_OBJECTS USER_OBJECTS	DBA ビューには、データベース内のすべてのスキーマ・オブジェクトが表示されます。ALL ビューには、現行ユーザーがアクセス可能なオブジェクトが表示されます。USER ビューには、現行ユーザーが所有しているオブジェクトが表示されます。
DBA_CATALOG ALL_CATALOG USER_CATALOG	データベース内にあるすべての表、ビュー、シノニムおよび順序の名前、タイプおよび所有者（USER ビューでは所有者は表示されない）がリストされます。
DBA_DEPENDENCIES ALL_DEPENDENCIES USER_DEPENDENCIES	プロシージャ、パッケージ、ファンクション、パッケージ本体およびトリガーの間の依存性（データベース・リンクを持たないビューへの依存性など）がすべて表示されます。
次のビューには、データベースのセグメントに関する情報が含まれています。	
DBA_SEGMENTS USER_SEGMENTS	すべてのデータベース・セグメントまたは現行ユーザーのセグメントに割り当てられた記憶域が表示されます。
次のビューには、データベースのエクステントに関する情報が含まれています。	
DBA_EXTENTS USER_EXTENTS	データベース内のすべてのセグメントまたは現行ユーザーのセグメントを構成するエクステントが表示されます。
DBA_FREE_SPACE USER_FREE_SPACE	すべての表領域または現行ユーザーが所有する表領域の使用可能エクステントが表示されます。

次に、これらのビューの使用例を示します。

関連項目： データ・ディクショナリ・ビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

例 1: スキーマ・オブジェクトのタイプ別表示

次の問合せは、問合せを発行しているユーザーが所有しているオブジェクトをすべてリストします。

```
SELECT OBJECT_NAME, OBJECT_TYPE
    FROM USER_OBJECTS;

OBJECT_NAME                OBJECT_TYPE
-------------------------  -------------------
EMP_DEPT                   CLUSTER
EMP                        TABLE
DEPT                       TABLE
EMP_DEPT_INDEX             INDEX
PUBLIC_EMP                 SYNONYM
EMP_MGR                    VIEW
```

例 2: 列情報の表示

_COLUMNS 接尾辞で終わるビューのいずれかを使用すれば、名前、データ型、長さ、精度、位取り、デフォルト・データ値などの列情報を表示できます。たとえば、次の問合せは、emp 表と dept 表のデフォルトの列値をすべてリストします。

```
SELECT TABLE_NAME, COLUMN_NAME, DATA_DEFAULT
    FROM USER_TAB_COLUMNS
    WHERE TABLE_NAME = 'DEPT' OR TABLE_NAME = 'EMP';

TABLE_NAME   COLUMN_NAME   DATA_DEFAULT
----------   ------------- --------------------
DEPT         DEPTNO
DEPT         DNAME
DEPT         LOC           'NEW YORK'
EMP          EMPNO
EMP          ENAME
EMP          JOB
EMP          MGR
EMP          HIREDATE       SYSDATE
EMP          SAL
EMP          COMM
EMP          DEPTNO
```

必ずしもすべての列がユーザー指定のデフォルトを持つとはかぎりません。そのような列には、デフォルトとして自動的に NULL が設定されます。

例 3: ビューとシノニムの依存性の表示

ビューまたはシノニムを作成するとき、ビューやシノニムはその基礎になるベース・オブジェクトに基づきます。ビューの依存性を明確にするには、ALL_DEPENDENCIES/USER_DEPENDENCIES/DBA_DEPENDENCIES データ・ディクショナリ・ビューを使用します。シノニムのベース・オブジェクトのリストを表示するには、ALL_SYNONYMS/USER_SYNONYMS/DBA_SYNONYMS データ・ディクショナリ・ビューを使用します。たとえば、次の問合せは、ユーザー jward によって作成されたシノニムのベース・オブジェクトをリストします。

```
SELECT TABLE_OWNER, TABLE_NAME, SYNONYM_NAME
    FROM DBA_SYNONYMS
    WHERE OWNER = 'JWARD';

TABLE_OWNER             TABLE_NAME   SYNONYM_NAME
----------------------  -----------  -----------------
SCOTT                   DEPT         DEPT
SCOTT                   EMP          EMP
```

例 4: 一般的なセグメント情報の表示

次の問合せは、各ロールバック・セグメントの名前、各ロールバック・セグメントを含む表領域および各ロールバック・セグメントのサイズを返します。

```
SELECT SEGMENT_NAME, TABLESPACE_NAME, BYTES, BLOCKS, EXTENTS
    FROM DBA_SEGMENTS
    WHERE SEGMENT_TYPE = 'ROLLBACK';

SEGMENT_NAME  TABLESPACE_NAME   BYTES       BLOCKS     EXTENTS
------------  ---------------   ---------   -------    ---------
RS1           SYSTEM                20480        10           2
RS2           TS1                   40960        20           3
SYSTEM        SYSTEM               184320        90           3
```

例 5: エクステントに関する一般的な情報の表示

データベース内に現在割り当てられているエクステントに関する一般的な情報は、DBA_EXTENTS データ・ディクショナリ・ビューに格納されています。たとえば、次の問合せによって、ロールバック・セグメントに対応付けられているエクステントとそれらのエクステントのサイズが識別されます。

```
SELECT SEGMENT_NAME, BYTES, BLOCKS
    FROM DBA_EXTENTS
    WHERE SEGMENT_TYPE = 'ROLLBACK';

SEGMENT_NAME      BYTES          BLOCKS
---------------   ---------      --------
RS1                   10240            5
RS1                   10240            5
SYSTEM                51200           25
SYSTEM                51200           25
SYSTEM                51200           25
```

SYSTEM ロールバック・セグメントは、等しいサイズ（50KB）の 3 つのエクステントから構成され、rs1 ロールバック・セグメントは、それぞれ 10KB の 2 つのエクステントから構成されていることがわかります。

例 6: データベースの空き領域（エクステント）の表示

データベース内の使用可能エクステント（どのセグメントにも割り当てられていないエクステント）に関する情報は、DBA_FREE_SPACE データ・ディクショナリ・ビューに格納されています。たとえば、次の問合せは、各表領域内の使用可能エクステントとして使用可能な空き領域を示します。

```
SELECT TABLESPACE_NAME, FILE_ID, BYTES, BLOCKS
    FROM DBA_FREE_SPACE;

TABLESPACE_NAME        FILE_ID     BYTES       BLOCKS
-------------------    ---------   --------    ----------
SYSTEM                         1    8120320          3965
SYSTEM                         1      10240             5
TS1                            2   10432512          5094
```

例 7: 追加のエクステントを割り当てることのできないセグメントの表示

セグメントに追加のエクステントを割り当てることができない場合は、DBA_FREE_SPACE と、DBA_SEGMENTS、DBA_TABLES、DBA_CLUSTERS、DBA_INDEXES および DBA_ROLLBACK_SEGS の各ビューを組み合せて使用し、その原因がデータ・ディクショナリ・オブジェクトのみにあるのかどうかを判断できます。

セグメントは、次のいずれかの理由でエクステントを割り当てることができない場合があります。

- セグメントを含む表領域に次のエクステントのための十分な領域がない場合
- セグメント内のエクステント数が、データ・ディクショナリ（SEG.MAX_EXTENTS）に記録されている最大のエクステント数に到達した場合

- セグメント内のエクステント数が、データ・ブロック・サイズ（オペレーティング・システム固有）によって許可される最大のエクステント数に到達した場合

> **注意：** MAXEXTENTS の STORAGE 句の値に UNLIMITED を設定することはできますが、データ・ディクショナリ表は許容されるブロックの最大数より多くの MAXEXTENTS を持つことはできません。したがって、データ・ディクショナリ表は制限のないフォーマットに変換できません。

次の問合せは、前述の基準のいずれかを満たすすべてのセグメントの名前、所有者および表領域を返します。

```
SELECT a.SEGMENT_NAME, a.SEGMENT_TYPE, a.TABLESPACE_NAME, a.OWNER
    FROM DBA_SEGMENTS a
    WHERE a.NEXT_EXTENT >= (SELECT MAX(b.BYTES)
        FROM DBA_FREE_SPACE b
        WHERE b.TABLESPACE_NAME = a.TABLESPACE_NAME)
    OR a.EXTENTS = a.MAX_EXTENTS
    OR a.EXTENTS = 'data_block_size' ;
```

> **注意：** この問合せを使用するときは、*data_block_size* をシステムのデータ・ブロック・サイズに置き換えてください。

追加のエクステントを割り当てることのできないセグメントを識別した後、その原因に応じて、次のどちらかの方法で問題を解決できます。

- 表領域がいっぱいの場合は、表領域にデータ・ファイルを追加します。
- セグメント内に存在するエクステントが多すぎるものの、セグメントの MAXEXTENTS を増やせない場合は、次の手順を実行します。

  1. セグメント内のデータをエクスポートします。
  2. セグメントを削除して、再作成します。その際、多数のエクステントを割り当てる必要がないように、INITIAL に大きい値を設定します。
  3. セグメントにデータをインポートします。

22

データ・ブロック破損の検出と修復

この章では、DBMS_REPAIR PL/SQL パッケージを使用して、データベースのスキーマ・オブジェクト内にあるデータ・ブロックの破損を修復する方法について説明します。この章の内容は、次のとおりです。

- データ・ブロック破損を修復するオプション
- DBMS_REPAIR パッケージの内容
- DBMS_REPAIR パッケージの使用方法
- DBMS_REPAIR の例

注意： DBMS_REPAIR パッケージについて詳しくない場合は、このパッケージに含まれる修復プロシージャを実行する際に、オラクル社カスタマ・サポート・センターのアナリストと共同で作業することをお薦めします。

データ・ブロック破損を修復するオプション

Oracle には、データ・ブロックの破損を検出して修正するために、複数の方法が用意されています。その 1 つは、破損の検出後にオブジェクトを削除して再作成することです。しかし、この方法が必ずしも可能とはかぎらず、またそれが望ましくない場合もあります。データ・ブロックの破損が行のサブセットにかぎられている場合は、破損した行を除くすべてのデータを選択して表を再作成する方法があります。

また、DBMS_REPAIR パッケージを使用してデータ・ブロック破損を管理する方法もあります。DBMS_REPAIR を使用すると、表と索引の破損ブロックを検出して修復できます。このアプローチを使用すると、可能なかぎり破損に対処できるとともに、再作成または修復中でもオブジェクトを引き続き使用できます。

注意： データの損失を伴う破損の場合は、そのデータがデータベース・システム全体にどのように格納されているかを分析して理解する必要があります。DBMS_REPAIR は万能ではなく、従来どおり、このパッケージで提供される修復アプローチが特定の破損に対して適切かどうかを判断する必要があります。修復の内容によっては、データを失ったり、論理的一貫性が損なわれる場合があります。そのため、DBMS_REPAIR の使用に伴う利害得失を比較検討することが必要です。

DBMS_REPAIR パッケージの内容

ここでは、パッケージに含まれている DBMS_REPAIR プロシージャと、その使用に伴う制約および制限事項について説明します。

関連項目： DBMS_REPAIR プロシージャの構文、制限事項および例外の詳細は、『Oracle9*i* 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。

DBMS_REPAIR プロシージャ

次の表は、DBMS_REPAIR パッケージに含まれているプロシージャの一覧を示します。

プロシージャ名	説明
CHECK_OBJECT	表または索引内の破損を検出してレポートします。
FIX_CORRUPT_BLOCKS	ブロックにソフトウェア破損を示すマークを付けます。これらのブロックは、従来は CHECK_OBJECT プロシージャで識別されていました。
DUMP_ORPHAN_KEYS	破損データ・ブロック内の行を指す索引エントリを（孤立キー表に）レポートします。
REBUILD_FREELISTS	オブジェクトの空きリストを再作成します。
SEGMENT_FIX_STATUS	セグメント領域管理が AUTO の場合に、ビットマップ・エントリの破損状態を修正する機能を提供します。
SKIP_CORRUPT_BLOCKS	このプロシージャを使用すると、表と索引のスキャン時に、破損マークが付いたブロックが無視されます。使用しない場合は、破損マークが付いたブロックが検出されたときにエラー ORA-1578 が返されます。
ADMIN_TABLES	修復表および孤立キー表の管理機能（作成、削除、パージ）を提供します。 **注意：**これらの表は常に SYS スキーマに作成されます。

これらのプロシージャの詳細と使用例は、22-9 ページの「DBMS_REPAIR の例」を参照してください。

制約と制限事項

DBMS_REPAIR プロシージャには、次の制約があります。

- LOB、ネストした表および VARRAY を含む表はサポートされますが、表外格納の列は無視されます。
- クラスタは、SKIP_CORRUPT_BLOCKS および REBUILD_FREELISTS プロシージャではサポートされますが、CHECK_OBJECT プロシージャではサポートされません。
- 索引構成表および LOB 索引はサポートされません。
- DUMP_ORPHAN_KEYS プロシージャは、ビットマップ索引またはファンクション・ベース索引には機能しません。
- DUMP_ORPHAN_KEYS プロシージャで処理される最大キー長は、3,950 バイトです。

DBMS_REPAIR パッケージの使用方法

データ・ブロック破損に対処する手段として DBMS_REPAIR を検討する場合は、次のアプローチに従うことをお薦めします。

タスク 1: 破損の検出とレポート

タスク 2: DBMS_REPAIR の使用に伴うコストと利点の評価

タスク 3: オブジェクトの使用可能化

タスク 4: 破損の修復および失われたデータの再作成

これらのタスクについて、次の項で説明します。

タスク 1: 破損の検出とレポート

最初のタスクとして、DBMS_REPAIR を使用する前に、破損を検出してレポートします。レポートでは、ブロックに関する問題が明らかになるだけでなく、それに対応する修復ディレクティブも識別されます。DBMS_REPAIR の他にも、破損を検出するためにいくつかのオプションがあります。表 22-1 は、各種の検出方法を示しています。

表 22-1　破損検出方法の比較

検出方法	説明
DBMS_REPAIR	指定した表、パーティションまたは索引のブロック・チェックが実行されます。修復表に結果が移入されます。
DB_VERIFY	オフライン・データベースのブロック・チェックを実行する外部コマンドライン・ユーティリティ。
ANALYZE	VALIDATE STRUCTURE オプションを指定すると、索引、表またはクラスタの構造の整合性が検証され、表と索引が同期しているかどうかがチェックまたは検証されます。
DB_BLOCK_CHECKING	初期化パラメータ DB_BLOCK_CHECKING=TRUE のときに実行されます。実際に破損マークを付ける前に、破損ブロックを識別します。チェックは、ブロックの変更時に実行されます。

DBMS_REPAIR:CHECK_OBJECT および ADMIN_TABLES プロシージャの使用

CHECK_OBJECT プロシージャは、指定されたオブジェクトのブロック破損をチェックしてレポートします。 索引と表に対する ANALYZE...VALIDATE STRUCTURE 文と同様に、索引とデータ・ブロックに対してブロック・チェックが実行されます。

CHECK_OBJECT では、破損がレポートされるだけでなく、そのオブジェクトに対して後で FIX_CORRUPT_BLOCKS を実行した場合に行われる修正も識別されます。この情報は修復表への移入によって使用可能になるため、最初に ADMIN_TABLES プロシージャで修復表を作成しておく必要があります。

CHECK_OBJECT プロシージャを実行した後は、修復表の簡単な問合せによってそのオブジェクトの破損および修復ディレクティブが表示されます。この情報に基づいて、レポートされた問題に最も適切な対処方法を評価できます。

DB_VERIFY: オフライン・データベース・チェックの実行

通常、データ破損の問題が発生した場合は、オフライン診断ユーティリティとして DB_VERIFY を使用します。

> **関連項目：** DB_VERIFY の詳細は、『Oracle9*i* データベース・ユーティリティ』を参照してください。

ANALYZE: 破損のレポート

ANALYZE TABLE...VALIDATE STRUCTURE 文は、分析するオブジェクトの構造の妥当性をチェックします。構造の妥当性チェックが正常に終了した場合は、それを確認するメッセージが表示されます。オブジェクトの構造内で破損が検出されると、エラー・メッセージが表示されます。この場合は、オブジェクトを削除して再作成する必要があります。

> **関連項目：** ANALYZE 文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

DB_BLOCK_CHECKING（ブロック・チェック初期化パラメータ）

DB_BLOCK_CHECKING 初期化パラメータ（デフォルト値は FALSE）を使用して、インスタンスのブロック・チェックを設定できます。これにより、データ・ブロックおよび索引ブロックが変更された際に、必ずそのブロックがチェックされます。DB_BLOCK_CHECKING は、ALTER SYSTEM SET 文で変更可能な動的パラメータです。システム表領域では、ブロック・チェックは常に使用可能になっています。

> **関連項目：** DB_BLOCK_CHECKING 初期化パラメータの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

タスク 2: DBMS_REPAIR の使用に伴うコストと利点の評価

DBMS_REPAIR を使用する前に、その利害得失を検討する必要があります。また、破損オブジェクトの対応手段として使用可能な他のオプションも検討してください。

最初のステップは、次の質問に答えることです。

1. 破損の範囲はどの程度ですか。

   破損の有無と修復アクションの要不要を判断するには、CHECK_OBJECT プロシージャを実行して修復表を問い合せます。

2. ブロック破損の対応手段として使用可能な他のオプションがありますか。この質問については、次の対応が可能かどうかを検討します。

   - 他のソースからのデータが使用可能であれば、そのオブジェクトを削除し、再作成して再移入する。
   - CREATE TABLE...AS SELECT 文を発行して、破損表から新しい表を作成する。
   - SELECT 文から破損行を除外して、破損を無視する。
   - メディア・リカバリを実行する。

3. DBMS_REPAIR を使用してオブジェクトを使用可能にした場合に、どのような論理的な破損や副作用が生じますか。それらの問題に対処できますか。そのためにはどんな作業が必要ですか。

   破損マークが付いたブロックの行にアクセスできない場合があります。また、正常にアクセスできる行が含まれているブロックでも、破損マークが付いている場合があります。

   ブロックに破損マークが付いている場合は、参照整合性制約が壊れていることがあります。この場合は、制約を使用禁止にし、再び使用可能にすると、不整合がレポートされます。すべての問題を解決すれば、再び制約を正常に使用できるようになります。

   表にトリガーが定義されている場合は、論理的な破損が生じることがあります。たとえば、行を再度挿入したときに、挿入トリガーが起動されるかどうかを確認します。これらの問題に対処するには、インストレーションでどのトリガーがどのように使用されているかを理解する必要があります。

   空きリスト・ブロックにアクセスできなくなる場合があります。破損ブロックが空きリストの先頭または最後にあると、領域管理によって空きリストが再初期化されます。これにより、空きリストにあるはずのブロックが含まれていない状態になる場合があります。この問題に対処するには、REBUILD_FREELISTS プロシージャを実行します。

   索引と表が同期していない場合があります。この問題に対処するには、最初に DUMP_ORPHAN_KEYS プロシージャを実行します（破損データの再作成に役立つ情報をキーから取得するため）。次に、ALTER INDEX REBUILD ONLINE 文を発行し、表と索引を同期化します。

4. 修復によってデータが失われる場合に、このデータを取り出すことができますか。

   データ・ブロックに破損マークが付いている場合は、索引からデータを取り出すことができます。この情報を取り出すには、DUMP_ORPHAN_KEYS プロシージャを利用します。ただし、この方法でデータを取り出せるかどうかは、索引と表の間にどの程度の冗長性があるかによります。

タスク 3: オブジェクトの使用可能化

このタスクでは、DBMS_REPAIR を使用して表と索引のスキャン時に破損を無視することにより、オブジェクトを使用可能にします。

破損の修復 :FIX_CORRUPT_BLOCKS および SKIP_CORRUPT_BLOCKS プロシージャの使用

DBMS_REPAIR の修復機能の有効範囲外にある破損をスキップする環境を設定し、それによって破損オブジェクトを使用可能にします。

データ・ブロック内の不良行のように、破損によってデータが失われている場合は、FIX_CORRUPT_BLOCKS プロシージャを使用して、この種のすべてのブロックに破損マークを設定します。次に、SKIP_CORRUPT_BLOCKS プロシージャを実行します。このプロシージャは、破損マークが付いているオブジェクトのブロックをスキップするように設定します。スキップを設定すると、表および索引のスキャン時に破損マーク付きのすべてのブロックがスキップされます。これは、メディア破損ブロックとソフトウェア破損ブロックのどちらにも適用されます。

破損ブロックをスキップする操作の意味

索引と表が同期化されていない場合は、ある問合せで索引のみをプローブし、後続の問合せで索引と表の両方をプローブするような状況において、SET TRANSACTION READ ONLY トランザクションの一貫性が保たれないことがあります。表ブロックに破損マークが付いている場合、この 2 つの問合せは異なる結果を返すので、読取り専用トランザクションのルールに違反します。この場合の対処方法の 1 つは、SET TRANSACTION READ ONLY トランザクション内で破損をスキップしないことです。

これと同様の問題は、連鎖している行の選択時にも発生します。実際には、同じ行を問い合せても、破損にアクセスできる場合とできない場合があるため、異なった結果が生じます。

タスク 4: 破損の修復および失われたデータの再作成

オブジェクトを使用可能にした後で、次の修復アクティビティを実行できます。

DUMP_ORPHAN_KEYS プロシージャを使用したデータのリカバリ

DUMP_ORPHAN_KEYS プロシージャは、破損データ・ブロック内の行を指す索引エントリをレポートします。この種の索引エントリがすべて、破損のキーと ROWID を格納する孤立キー表に挿入されます。

索引エントリ情報を取り出した後、ALTER INDEX REBUILD ONLINE 文を使用して索引を再作成できます。

REBUILD_FREELISTS プロシージャを使用した空きリストの修復

セグメントの空き領域が空きリストで管理されている場合（SEGMENT SPACE MANAGEMENT MANUAL）は、このプロシージャを使用します。

破損マークが付いているブロックが空きリストの先頭または最後に検出されると、その空きリストが再初期化されて、エラーが返されます。これにより破損ブロックは空きリストから除去されますが、破損ブロックに続くすべてのブロックに空きリストからアクセスできなくなります。

空きリストを再初期化するには、REBUILD_FREELISTS プロシージャを使用します。オブジェクトがスキャンされ、空きリストに入れる適切なブロックがマスター空きリストに追加されます。複数の空きリスト・グループがあるときは、一度に 1 ブロックずつ均等に分配されます。空きリストの再作成時、破損マークの付いたオブジェクト内のブロックは無視されます。

SEGMENT_FIX_STATUS プロシージャを使用したセグメント・ビットマップの修正

セグメントの空き領域がビットマップで管理されている場合（SEGMENT SPACE MANAGEMENT AUTO）は、このプロシージャを使用します。

このプロシージャは、対応するブロックの現在の内容に基づいてビットマップ・エントリの状態を再計算します。また、ビットマップ・エントリを特定の値に設定するように指定することもできます。通常は、状態が適切に再計算されるので、値を強制的に設定する必要はありません。

DBMS_REPAIR の例

ここでは、DBMS_REPAIR プロシージャの使用例を示します。

- ADMIN_TABLES を使用した修復表または孤立キー表の作成
- CHECK_OBJECT プロシージャを使用した破損の検出
- FIX_CORRUPT_BLOCKS プロシージャを使用した破損ブロックの修正
- 破損データ・ブロックを指す索引エントリの検索 :DUMP_ORPHAN_KEYS
- REBUILD_FREELISTS プロシージャを使用した空きリストの再作成
- 破損ブロックのスキップの使用可能 / 使用禁止 :SKIP_CORRUPT_BLOCKS

ADMIN_TABLES を使用した修復表または孤立キー表の作成

修復表は、CHECK_OBJECT プロシージャによって検出された破損の内容と、FIX_CORRUPT_BLOCKS プロシージャを実行した場合にこれらの破損がどのように処理されるかを示す情報を提供します。また、FIX_CORRUPT_BLOCKS プロシージャの実行が必要かを判断する際にも使用されます。

孤立キー表は、DUMP_ORPHAN_KEYS プロシージャの実行時に使用され、破損行を指す索引エントリが格納されます。DUMP_ORPHAN_KEYS プロシージャは、そのアクティビティをロギングし、索引情報を使用可能な形にして、孤立キー表に移入します。

ADMIN_TABLE プロシージャは、修復表または孤立キー表の作成、パージまたは削除に使用します。

修復表の作成

次の例では、users 表領域の修復表を作成しています。

```
BEGIN
DBMS_REPAIR.ADMIN_TABLES (
     TABLE_NAME => 'REPAIR_TABLE',
     TABLE_TYPE => dbms_repair.repair_table,
     ACTION     => dbms_repair.create_action,
     TABLESPACE => 'USERS');
END;
/
```

修復表または孤立キー表それぞれについて、存在しなくなったオブジェクトに関連する行を除外するビューも作成されます。ビュー名は、修復表または孤立キー表の名前に対応していますが、接頭辞 DBA_ が付いています（DBA_REPAIR_TABLE や DBA_ORPHAN_KEY_TABLE など）。

次の問合せでは、前述の例で作成された修復表の定義を表示しています。

```
SQL> DESC REPAIR_TABLE
 Name                          Null?    Type
 ----------------------------- -------- --------------
 OBJECT_ID                     NOT NULL NUMBER
 TABLESPACE_ID                 NOT NULL NUMBER
 RELATIVE_FILE_ID              NOT NULL NUMBER
 BLOCK_ID                      NOT NULL NUMBER
 CORRUPT_TYPE                  NOT NULL NUMBER
 SCHEMA_NAME                   NOT NULL VARCHAR2(30)
 OBJECT_NAME                   NOT NULL VARCHAR2(30)
 BASEOBJECT_NAME                        VARCHAR2(30)
 PARTITION_NAME                         VARCHAR2(30)
 CORRUPT_DESCRIPTION                    VARCHAR2(2000)
 REPAIR_DESCRIPTION                     VARCHAR2(200)
 MARKED_CORRUPT                NOT NULL VARCHAR2(10)
 CHECK_TIMESTAMP               NOT NULL DATE
 FIX_TIMESTAMP                          DATE
 REFORMAT_TIMESTAMP                     DATE
```

孤立キー表の作成

次の例は、users 表領域の孤立キー表の作成方法を示しています。

```
BEGIN
DBMS_REPAIR.ADMIN_TABLES (
     TABLE_NAME => 'ORPHAN_KEY_TABLE',
     TABLE_TYPE => dbms_repair.orphan_table,
     ACTION     => dbms_repair.create_action,
     TABLESPACE => 'USERS');
END;
/
```

次の問合せでは、孤立キー表の定義を表示しています。

```
SQL> DESC ORPHAN_KEY_TABLE

 Name                          Null?    Type
 ----------------------------- -------- -----------------
 SCHEMA_NAME                   NOT NULL VARCHAR2(30)
 INDEX_NAME                    NOT NULL VARCHAR2(30)
 IPART_NAME                             VARCHAR2(30)
 INDEX_ID                      NOT NULL NUMBER
 TABLE_NAME                    NOT NULL VARCHAR2(30)
 PART_NAME                              VARCHAR2(30)
 TABLE_ID                      NOT NULL NUMBER
 KEYROWID                      NOT NULL ROWID
```

```
 KEY                           NOT NULL ROWID
 DUMP_TIMESTAMP                NOT NULL DATE
```

CHECK_OBJECT プロシージャを使用した破損の検出

CHECK_OBJECT プロシージャは、指定されたオブジェクトをチェックし、破損および修復ディレクティブに関する情報を修復表に移入します。オブジェクトの一部をチェックする場合は、必要に応じて範囲、パーティション名またはサブパーティション名を指定できます。

妥当性チェックでは、オブジェクト内部でそれまでに破損マークが付けられていないブロックがすべてチェックされます。ブロックごとに、トランザクションおよびデータ・レイヤー部分の自己整合性がチェックされます。CHECK_OBJECT の実行中に、破損バッファ・キャッシュ・ヘッダーを持つブロックが検出されると、そのブロックはスキップされます。

scott.dept 表に対する CHECK_OBJECT プロシージャの実行例を次に示します。

```
SET SERVEROUTPUT ON
DECLARE num_corrupt INT;
BEGIN
num_corrupt := 0;
DBMS_REPAIR.CHECK_OBJECT (
     SCHEMA_NAME => 'SCOTT',
     OBJECT_NAME => 'DEPT',
     REPAIR_TABLE_NAME => 'REPAIR_TABLE',
     CORRUPT_COUNT =>  num_corrupt);
DBMS_OUTPUT.PUT_LINE('number corrupt: ' || TO_CHAR (num_corrupt));
END;
/
```

SQL*Plus には、1 つの破損を示す次の行が出力されます。

```
number corrupt: 1
```

修復表を問い合せると、破損の説明および修復アクションに関する提案を含む情報が表示されます。

```
SELECT OBJECT_NAME, BLOCK_ID, CORRUPT_TYPE, MARKED_CORRUPT,
       CORRUPT_DESCRIPTION, REPAIR_DESCRIPTION
     FROM REPAIR_TABLE;

OBJECT_NAME                      BLOCK_ID CORRUPT_TYPE MARKED_COR
------------------------------ ---------- ------------ ----------
CORRUPT_DESCRIPTION
--------------------------------------------------------------------------------
REPAIR_DESCRIPTION
--------------------------------------------------------------------------------
DEPT                                    3            1 FALSE
```

```
kdbchk: row locked by non-existent transaction
        table=0   slot=0
        lockid=32   ktbbhitc=1
mark block software corrupt
```

この時点では、まだ破損ブロックに破損マークが付いていないため、ここで意味を持つデータを抽出します。ブロックに破損マークが付けられた後は、そのブロック全体がスキップされます。

FIX_CORRUPT_BLOCKS プロシージャを使用した破損ブロックの修正

CHECK_OBJECT プロシージャによって生成済みの修復表内の情報に基づいて FIX_CORRUPT_BLOCKS プロシージャを使用し、指定したオブジェクトの破損ブロックを修正します。ブロックの変更を有効にする前に、ブロックがチェックされてまだ破損していることが確認されます。破損ブロックは、ソフトウェア破損マークを付けることによって修復されます。修復が実行されると、修復表内の対応する行が修正タイムスタンプで更新されます。

次の例では、CHECK_OBJECT プロシージャによってレポートされた表 scott.dept の破損ブロックを修正しています。

```
SET SERVEROUTPUT ON
DECLARE num_fix INT;
BEGIN
num_fix := 0;
DBMS_REPAIR.FIX_CORRUPT_BLOCKS (
     SCHEMA_NAME => 'SCOTT',
     OBJECT_NAME=> 'DEPT',
     OBJECT_TYPE => dbms_repair.table_object,
     REPAIR_TABLE_NAME => 'REPAIR_TABLE',
     FIX_COUNT=> num_fix);
DBMS_OUTPUT.PUT_LINE('num fix: ' || TO_CHAR(num_fix));
END;
/

SQL*Plus outputs the following line:

num fix: 1
```

次の問合せによって、修復が完了していることを確認します。

```
SELECT OBJECT_NAME, BLOCK_ID, MARKED_CORRUPT
     FROM REPAIR_TABLE;

OBJECT_NAME                      BLOCK_ID MARKED_COR
------------------------------ ---------- ----------
DEPT                                    3 TRUE
```

破損データ・ブロックを指す索引エントリの検索 :DUMP_ORPHAN_KEYS

DUMP_ORPHAN_KEYS プロシージャは、破損データ・ブロック内の行を指す索引エントリをレポートします。この種の索引エントリが検出されるたびに、指定した孤立キー表に 1 行ずつ挿入されます。この孤立キー表は、事前に作成しておく必要があります。

この情報は、表内の失われた行を再作成する場合や診断に使用します。

注意： このプロシージャは、修復表で識別された表に対応付けられている索引ごとに実行する必要があります。

この例で、pk_dept は scott.dept 表の索引です。この索引をスキャンして、破損データ・ブロック内の行を指す索引エントリの有無を判別しています。

```
SET SERVEROUTPUT ON
DECLARE num_orphans INT;
BEGIN
num_orphans := 0;
DBMS_REPAIR.DUMP_ORPHAN_KEYS (
     SCHEMA_NAME => 'SCOTT',
     OBJECT_NAME => 'PK_DEPT',
     OBJECT_TYPE => dbms_repair.index_object,
     REPAIR_TABLE_NAME => 'REPAIR_TABLE',
     ORPHAN_TABLE_NAME=> 'ORPHAN_KEY_TABLE',
     KEY_COUNT => num_orphans);
DBMS_OUTPUT.PUT_LINE('orphan key count: ' || TO_CHAR(num_orphans));
END;
/
```

次のように、孤立キーが 3 つあることを示す行が出力されます。

```
orphan key count: 3
```

孤立キー表の索引エントリは、索引の再作成が必要であることを示しています。索引の再作成により、表プローブと索引プローブが同じ結果セットを返すことが保証されます。

REBUILD_FREELISTS プロシージャを使用した空きリストの再作成

REBUILD_FREELISTS プロシージャは、指定されたオブジェクトの空きリストを再作成します。空きブロックはすべて、マスター空きリストに入れられます。他の空きリストはすべてゼロになります。オブジェクトに複数の空きリスト・グループがある場合は、空きブロックがラウンドロビン方式でそれぞれのグループに割り当てられ、すべての空きリスト間で分配されます。

次の例では、表 scott.dept の空きリストが再作成されます。

```
BEGIN
DBMS_REPAIR.REBUILD_FREELISTS (
     SCHEMA_NAME => 'SCOTT',
     OBJECT_NAME => 'DEPT',
     OBJECT_TYPE => dbms_repair.table_object);
END;
/
```

破損ブロックのスキップの使用可能 / 使用禁止 :SKIP_CORRUPT_BLOCKS

SKIP_CORRUPT_BLOCKS プロシージャは、指定されたオブジェクトの索引および表のスキャン時に破損ブロックをスキップするかしないかを指定します。オブジェクトが表であれば、スキップは表とその索引に適用されます。オブジェクトがクラスタであれば、スキップはクラスタ内のすべての表とそれぞれの索引に適用されます。

次の例では、scott.dept 表のソフトウェア破損ブロックのスキップを可能に設定しています。

```
BEGIN
DBMS_REPAIR.SKIP_CORRUPT_BLOCKS (
     SCHEMA_NAME => 'SCOTT',
     OBJECT_NAME => 'DEPT',
     OBJECT_TYPE => dbms_repair.table_object,
     FLAGS => dbms_repair.skip_flag);
END;
/
```

DBA_TABLES ビューを使用して scott の表を問い合せると、表 scott.dept の SKIP_CORRUPT が使用可能になっていることが示されます。

```
SELECT OWNER, TABLE_NAME, SKIP_CORRUPT FROM DBA_TABLES
    WHERE OWNER = 'SCOTT';

OWNER                          TABLE_NAME                     SKIP_COR
------------------------------ ------------------------------ --------
SCOTT                          ACCOUNT                        DISABLED
SCOTT                          BONUS                          DISABLED
SCOTT                          DEPT                           ENABLED
```

```
SCOTT                          DOCINDEX                       DISABLED
SCOTT                          EMP                            DISABLED
SCOTT                          RECEIPT                        DISABLED
SCOTT                          SALGRADE                       DISABLED
SCOTT                          SCOTT_EMP                      DISABLED
SCOTT                          SYS_IOT_OVER_12255             DISABLED
SCOTT                          WORK_AREA                      DISABLED

10 rows selected.
```

第IV部

データベース・セキュリティ

第 IV 部では、データベースのセキュリティに影響するユーザーおよび権限管理の問題について説明します。第 IV 部の構成は、次のとおりです。

23

セキュリティ・ポリシーの設定

この章では、データベースの運用におけるセキュリティ・ポリシーを作成するためのガイドラインについて説明します。この章の内容は、次のとおりです。

- システム・セキュリティ・ポリシー
- データ・セキュリティ・ポリシー
- ユーザー・セキュリティ・ポリシー
- パスワード管理ポリシー
- 監査方針
- セキュリティ・チェックリスト

システム・セキュリティ・ポリシー

ここでは、システム・セキュリティ・ポリシーについて説明します。この項の内容は、次のとおりです。

- データベース・ユーザー管理
- ユーザー認証
- オペレーティング・システムのセキュリティ

各データベースには、セキュリティ・ポリシーのあらゆる側面のメンテナンスを担当する管理者、つまりセキュリティ管理者が 1 名以上必要です。小規模なデータベース・システムの場合は、データベース管理者（DBA）がセキュリティ管理者を兼務できます。ただし、大規模なデータベース・システムの場合は、専任者または専任グループがセキュリティ管理者に限定される責務を担当するようにしてください。

システムのセキュリティ管理担当者を決定した後、すべてのデータベースに対してセキュリティ・ポリシーを設定してください。データベースのセキュリティ・ポリシーには、後述の説明に従って、いくつかの詳細な方針を設定する必要があります。

データベース・ユーザー管理

データベース・ユーザーとは、Oracle データベースの情報へのアクセス・パスです。このため、データベース・ユーザーの管理に関しては厳密なセキュリティを確保する必要があります。データベース・システムのサイズやデータベース・ユーザーの管理に必要な作業量によっては、データベース・ユーザーの作成、変更または削除に必要な権限を持つユーザーはセキュリティ管理者のみである場合があります。また、データベース・ユーザーを管理する権限を持つ管理者が多数存在する場合もあります。どちらの場合でも、信頼のおける担当者のみに、データベース・ユーザーを管理するための強力な権限を付与する必要があります。

ユーザー認証

Oracle では、データベース・パスワード、ホスト・オペレーティング・システム、ネットワーク・サービスまたは Secure Sockets Layer（SSL）を使用して、データベース・ユーザーを**認証**（正しいユーザーであることを確認すること）できます。

注意： ネットワーク認証サービスまたは SSL を使用して認証を受ける場合は、事前に Oracle Advanced Security をインストールしておく必要があります。これらの認証タイプの詳細は、『Oracle Advanced Security 管理者ガイド』を参照してください。

ユーザー認証とその指定方法は、24-9 ページの「ユーザー認証方式」を参照してください。

オペレーティング・システムのセキュリティ

Oracle とその他のデータベース・アプリケーションが稼働するオペレーティング・システム環境では、状況に応じて、次のようなセキュリティ上の問題を考慮する必要があります。

- DBA には、ファイルを作成および削除するためのオペレーティング・システム権限が必要です。
- 通常のデータベース・ユーザーには、データベースに関連するファイルを作成または削除するためのオペレーティング・システム権限を付与する必要はありません。
- オペレーティング・システムを使用してユーザーのデータベース・ロールを識別する場合、セキュリティ管理者には、オペレーティング・システム・アカウントのセキュリティ・ドメインを変更するためのオペレーティング・システム権限が必要です。

関連項目： オペレーティング・システムのセキュリティ問題に関する詳細は、オペレーティング・システム固有の Oracle マニュアルを参照してください。

データ・セキュリティ・ポリシー

データ・セキュリティには、データベースへのアクセスおよびデータベースの使用をオブジェクト・レベルで制御するメカニズムが含まれます。データ・セキュリティ・ポリシーによって、特定のスキーマ・オブジェクトにアクセスできるユーザーと、各ユーザーがオブジェクトに対して許可されているアクションのタイプが決まります。たとえば、ユーザー `scott` は、`emp` 表を使用して `SELECT` 文および `INSERT` 文を発行できますが、`DELETE` 文は発行できません。また、データ・セキュリティ・ポリシーでは、スキーマ・オブジェクトごとに監査が必要なアクション（ある場合）も定義します。

データ・セキュリティ・ポリシーは、主にデータベースのデータに対して設定するセキュリティのレベルによって決まります。たとえば、どのユーザーでも自由にスキーマ・オブジェクトを作成できるようにする場合、またはあるユーザーが所有するオブジェクトへのアクセス権限をシステム上の他のユーザーにも付与する場合は、データベースのデータ・セキュリティを厳密に設定する必要はありません。一方、DBA またはセキュリティ管理者のみがオブジェクトを作成し、オブジェクトのアクセス権限をロールおよびユーザーに付与できるようにするには、データ・セキュリティの管理を強化する必要があります。

全体的なデータ・セキュリティは、データの機密性に基づいています。機密性の低い情報に対して、厳しいデータ・セキュリティ・ポリシーを設定する必要はありません。しかし、機密性の高いデータには、セキュリティ・ポリシーを慎重に設定し、オブジェクトに対するアクセスを厳しく制御してください。

データ・セキュリティを実装する方法には、システム権限とオブジェクト権限、およびロールを使用する方法があります。ロールとは、ユーザーにまとめて付与できるようにグループ化されている権限のセットです。権限とロールについては、第 25 章「ユーザー権限とロールの管理」を参照してください。

また、ビューの定義によって表データへのアクセスを制限できるため、ビューを使用してデータ・セキュリティを実装する方法もあります。たとえば、重要なデータを含む列を除外したビューを作成できます。ビューについては、第 20 章「ビュー、順序およびシノニムの管理」を参照してください。

また、ファイングレイン・アクセス・コントロールと、関連するアプリケーション・コンテキストを使用して、データ・セキュリティを実装する方法もあります。ファイングレイン・アクセス・コントロールは、ファンクションを使用してセキュリティ・ポリシーを実装し、そのセキュリティ・ポリシーを表またはビューに対応付けることができる Oracle の機能です。実際には、セキュリティ・ポリシー・ファンクションによって SQL 文に追加される `WHERE` 条件が生成され、表またはビュー内のデータ行へのユーザー・アクセスが制限されます。アプリケーション・コンテキストは、アクセス制御の決定に使用される情報を格納するための保護データ・キャッシュです。

関連項目：

ファイングレイン・アクセス・コントロールとアプリケーション・コンテキストの実装の詳細は、次のマニュアルを参照してください。

- 『Oracle9*i* アプリケーション開発者ガイドー基礎編』
- 『Oracle9*i* 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』

ユーザー・セキュリティ・ポリシー

ここでは、ユーザー・セキュリティ・ポリシーについて説明します。この項の内容は、次のとおりです。

- 一般的なユーザー・セキュリティ
- エンド・ユーザーのセキュリティ
- 管理者のセキュリティ
- アプリケーション開発者のセキュリティ
- アプリケーション管理者のセキュリティ

一般的なユーザー・セキュリティ

すべてのタイプのデータベース・ユーザーについて、次の一般的なユーザー・セキュリティの問題を検討してください。

- パスワード・セキュリティ
- 権限管理

パスワード・セキュリティ

ユーザー認証をデータベースで管理している場合は、データベース・アクセスのセキュリティを確保するために、パスワード・セキュリティ・ポリシーを設定する必要があります。たとえば、各データベース・ユーザーは、パスワードが別のユーザーに漏れた場合はもちろん、そうでなくても定期的にパスワードを変更する必要があります。このような状況でユーザーにパスワードの変更を強制することにより、データベースへの不正なアクセスを減らすことができます。

パスワードの機密性の保護を強化するため、クライアント / サーバー間およびサーバー / サーバー間の接続に暗号化されたパスワードを使用するように Oracle を構成できます。

注意： クライアント / サーバー間およびサーバー / サーバー間の接続でパスワードを暗号化するように Oracle を構成することをお薦めします。このように構成しなければ、ネットワーク上を覗き見している悪意あるユーザーが暗号化されていないパスワードを入手し、それを使用して他のユーザーとしてデータベースに接続することにより、そのユーザーになりすます危険性があります。

次の値を設定すると、接続の確認に使用するパスワードを常に暗号化できます。

- クライアント・マシンの環境変数 `ORA_ENCRYPT_LOGIN` を `TRUE` に設定します。
- サーバーの `DBLINK_ENCRYPT_LOGIN` 初期化パラメータを `TRUE` に設定します。

クライアントとサーバーの両方で暗号化を使用可能にした場合、パスワードはネットワーク内を平文で送信されるのではなく、DES（データ暗号化規格）アルゴリズムの修正版を使用して暗号化されます。

`DBLINK_ENCRYPT_LOGIN` 初期化パラメータは、2 つの Oracle サーバーを接続するときに使用されます（分散問合せを実行する場合など）。クライアントから接続する場合は、環境変数 `ORA_ENCRYPT_LOGIN` がチェックされます。

パスワードを使用してサーバーに接続しようとすると、そのパスワードはサーバーに送信される前に必ず暗号化されます。接続が失敗すると、監査が使用可能であれば監査ログにその失敗が記録されます。次に、適切な `DBLINK_ENCRYPT_LOGIN` 値または `ORA_ENCRYPT_LOGIN` 値がチェックされます。この値が `FALSE` に設定されていると、Oracle は暗号化されていないパスワードを使用して再度接続を試みます。接続が成功すると、監査ログに記録さ

れた失敗が接続に置き換わり、接続が維持されます。不正なユーザーが使用した暗号化されていないパスワードによって Oracle が接続を再試行することを防ぐために、対応するパラメータの値を必ず TRUE に設定してください。

権限管理

セキュリティ管理者は、すべてのタイプのユーザーについて権限管理に関する問題を検討する必要があります。たとえば、多数のユーザー名が登録されているデータベースで、ユーザーが使用できる権限を管理するには、ロール（関連する権限の名前付きグループ。ユーザーまたは他のロールに付与できる）を使用すると便利な場合があります。しかし、登録されているユーザー名が少ないデータベースでは、ユーザーに権限を明示的に付与し、ロールは使用しない方が容易に管理できます。

セキュリティ管理者は、多数のユーザー、アプリケーションまたはオブジェクトを扱うデータベースを管理する場合、ロールの特長を利用する必要があります。ロールは、複雑な環境における権限管理のタスクを大幅に簡素化します。

エンド・ユーザーのセキュリティ

セキュリティ管理者は、エンド・ユーザーのセキュリティ・ポリシーを定義する必要があります。データベースに多数のユーザーが登録されている場合、セキュリティ管理者はユーザーのどの集合をユーザー・グループに分類するかを決定し、各グループのユーザー・ロールを作成できます。また、各ユーザー・ロールに必要な権限またはアプリケーション・ロールを付与し、そのユーザー・ロールをユーザーに割り当てることができます。例外を考慮して、個々のユーザーに対して明示的に付与する必要がある権限も合せて決定してください。

エンド・ユーザー権限管理のためのロールの使用

ロールは、データベース・ユーザーの異なるグループで必要となる共通の権限をまとめて付与および管理する最も簡単な方法です。

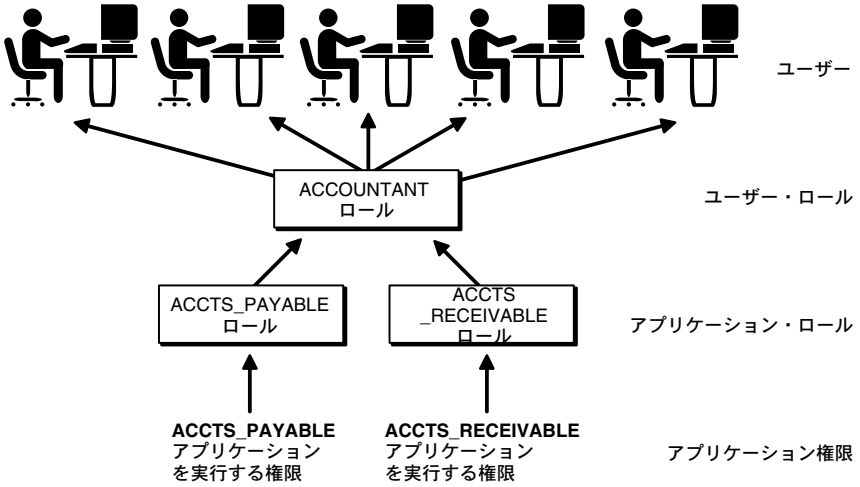
ある会社の経理部門のユーザー全員がデータベース・アプリケーション accts_receivable と accts_payable を実行するための権限を必要としている状況を考えてみます。この場合、ロールには両方のアプリケーションを対応付け、それらのアプリケーションを実行するために必要なオブジェクト権限を含めます。

このような単純なセキュリティを必要とする状況に対処するには、DBA またはセキュリティ管理者は次のアクションを実施します。

1. accountant という名前のロールを作成します。
2. データベース・アプリケーション accts_receivable と accts_payable のロールを accountant ロールに付与します。
3. accountant ロールを経理部門の各ユーザーに付与します。

このセキュリティ・モデルを図 23-1 に示します。

図 23-1　ユーザー・ロール



このモデルは、次のような状況に対処します。

- その後、経理部員に新しいデータベース・アプリケーションのロールが必要となった場合は、そのアプリケーションのロールを accountant ロールに付与します。これにより、経理部門のすべてのユーザーに新しいデータベース・アプリケーションの権限が自動的に付与されます。アプリケーションを使用する個々のユーザーに、アプリケーションのロールを付与する必要はありません。
- 同様に、経理部門で特定のアプリケーションが不要になった場合も、そのアプリケーションのロールを accountant ロールから削除できます。
- accts_receivable または accts_payable の各アプリケーションで必要な権限が変更された場合は、新しい権限をアプリケーション・ロールに付与またはロールから削除できます。accountant ロールのセキュリティ・ドメイン、および accountant ロールを付与されたすべてのユーザーにその権限の変更が自動的に反映されます。

可能な状況では必ずロールを使用して、エンド・ユーザーの権限管理の効率化および簡素化を行ってください。

エンド・ユーザー権限管理のためのディレクトリ・サービスの使用

Oracle Advanced Security のエンタープライズ・ユーザーおよびエンタープライズ・ロール機能を使用して、ユーザーとその認証をディレクトリ・サービスで集中管理することもできます。この機能の詳細は、『Oracle Advanced Security 管理者ガイド』を参照してください。

管理者のセキュリティ

セキュリティ管理者には、DBA のセキュリティに対処する方針が必要です。たとえば、データベースが大規模で様々なタイプの DBA が存在している場合、セキュリティ管理者は、関連のある管理権限をいくつかの管理ロールに分類できます。分類した管理ロールは、適切な管理者ユーザーに付与できます。一方、データベースの規模が小さく、管理者が少ない場合は、管理ロールを 1 つだけ作成し、その権限を管理者全員に付与するのが便利です。

関連項目： 管理者のセキュリティの詳細は、第 1 章「Oracle データベース管理者」を参照してください。

SYS および SYSTEM としての接続に対する保護

SYS および SYSTEM にデフォルト・パスワードを使用した場合は、データベースの作成後、ただちに SYS および SYSTEM 管理ユーザー名のパスワードを変更してください。SYS または SYSTEM として接続すると、データベースを変更できる強力な権限がユーザーに与えられます。たとえば、SYS として接続すると、ユーザーはデータ・ディクショナリ表を変更できます。これらのユーザー名に対応付けられている権限は機密に関わるものであるため、DBA を選択したときのみ使用可能にしてください。

他の管理用ユーザー名が作成されるオプションをインストールした場合、そのユーザー名は、最初はロックされた状態で作成されます。これらのアカウントのロックを解除するには、ALTER USER 文を使用します。また、これらのアカウントに対応するパスワードを変更する際にも、ALTER USER 文を使用します。

これらのアカウントのパスワードを変更する手順は、24-6 ページの「ユーザーの変更」を参照してください。

管理者の接続に対する保護

管理者権限を使用してデータベースに接続できるのは、DBA のみでなければなりません。次に例を示します。

```
CONNECT username/password AS SYSDBA/SYSOPER
```

SYSOPER として接続したユーザーには、STARTUP、SHUTDOWN、リカバリ操作などの基本操作を実行する権限が与えられます。SYSDBA として接続したユーザーには、これらの権限に加えて、データベースやデータベース内のオブジェクトに対してあらゆる操作（CREATE、DROP、DELETE など）を無制限に実行できる権限が与えられます。SYSDBA として接続したユーザーは SYS スキーマに割り当てられ、SYS スキーマ内のデータ・ディクショナリ表を変更できます。

管理者権限管理のためにロールを使用する方法

ロールは、データベースの管理者に必要とされる強力なシステム権限やロールを制限するための最も簡単な方法です。

大規模なインストール環境におけるデータベース管理者の業務を数名のデータベース管理者で分担し、各データベース管理者がそれぞれ次のような特定のデータベース管理業務を担当する例を考えてみます。

- オブジェクトの作成とメンテナンス
- データベースのチューニングとパフォーマンス
- 新規ユーザーの作成と、データベース・ユーザーへのロールおよび権限の付与
- 日常的なデータベース操作（STARTUP、SHUTDOWN、バックアップおよびリカバリ操作など）
- データベース・リカバリなどの緊急事態

データベース管理の経験がなく、権限の制限を必要とする新任の DBA もいます。

この例では、セキュリティ管理者は DBA に対して次のようなセキュリティを組織的に構成する必要があります。

1. 6 つのロールを定義して、各ロールにそれぞれのタイプの業務を遂行するために必要な異なる権限を含めます（たとえば、dba_objects、dba_tune、dba_security、dba_maintain、dba_recov、dba_new など）。
2. 各ロールに適切な権限を付与します。
3. 各タイプの DBA に対応するロールを付与します。

この計画では、状況の変化に対して次のように対応できるので、今後問題が発生する可能性が低くなります。

- あるデータベース管理者の業務範囲を変更して担当業務を追加する場合、その DBA には新しい担当業務に対応する他の管理ロールを付与できます。
- あるデータベース管理者の業務範囲を変更して担当業務を減らす場合、その DBA の該当する管理ロールを取り消すことができます。
- データ・ディクショナリには必ず各ロールと各ユーザーの情報が格納され、各 DBA のタスクを明らかにする目的で利用できます。

アプリケーション開発者のセキュリティ

セキュリティ管理者は、データベースを使用するアプリケーション開発者に対して特別なセキュリティ・ポリシーを定義する必要があります。セキュリティ管理者は、アプリケーション開発者に、必要なオブジェクトを作成するための権限を付与できます。また、開発者からのオブジェクト作成要求を受け付ける DBA に対してのみ、オブジェクトの作成権限を付与することもできます。

アプリケーション開発者とその権限

データベース・アプリケーション開発者は、開発作業を行うために特別な権限グループを必要とする独特なデータベース・ユーザーです。エンド・ユーザーとは異なり、開発者は CREATE TABLE や CREATE PROCEDURE などのシステム権限を必要とします。ただし、データベース内で開発者が実行できる機能全般を制限するために、開発者には特定のシステム権限のみを付与するようにします。

アプリケーション開発者の環境 : テストおよび本番データベース

多くの場合、アプリケーション開発は、テスト・データベースを対象とするように制限されており、本番データベースを使用したアプリケーション開発は許可されていません。この制限によって、アプリケーション開発者はエンド・ユーザーと競合せずにデータベース・リソースを獲得できるため、本番データベースに悪影響を及ぼすことはありません。

アプリケーションの開発およびテストが完了すると、そのアプリケーションは本番データベースにアクセスすることが許可され、対象のエンド・ユーザーがアプリケーションを使用できるようになります。

無制限および制限付きのアプリケーション開発

データベース管理者はアプリケーション開発者に付与する権限を決定する際に、次のオプションを定義できます。

- 無制限の開発

  アプリケーション開発者は、表、索引、プロシージャ、パッケージなど、スキーマ・オブジェクトの新規作成が許可されています。このオプションを使用すると、他のオブジェクトとは独立したアプリケーションを開発できます。

- 制限付きの開発

  アプリケーション開発者は、スキーマ・オブジェクトの新規作成が許可されていません。必要な表、索引、プロシージャなどはすべて、アプリケーション開発者の要求に従ってデータベース管理者が作成します。このオプションを使用すると、データベースのスペース使用やデータベース内での情報のアクセス・パスをデータベース管理者が完全に制御できます。

データベース・システムの中には、どちらか 1 つのオプションしか使用できないものもありますが、それ以外のシステムでは、この 2 つのオプションは混合して使用できます。たとえば、アプリケーション開発者にストアド・プロシージャとパッケージの新規作成は許可しても、表と索引の作成は許可しないようにすることが可能です。セキュリティ管理者は、次の条件に基づいてこの問題を決定します。

- データベースのスペース使用について必要な制御。
- スキーマ・オブジェクトへのアクセス・パスについて必要な制御。
- アプリケーション開発に使用するデータベース。アプリケーション開発にテスト・データベースを使用する場合は、より自由な開発方針が必要となります。

アプリケーション開発者のためのロールと権限

セキュリティ管理者は、標準的なアプリケーション開発者が必要とする権限を管理するためのロールを作成できます。たとえば、APPLICATION_DEVELOPER という名前を付けた標準的なロールに、CREATE TABLE、CREATE VIEW、CREATE PROCEDURE などの各システム権限を指定できます。アプリケーション開発者のロールを定義する際は、次の点を考慮してください。

- 通常は、開発者独自のオブジェクトを作成できるように、アプリケーション開発者に CREATE システム権限を付与します。ただし、CREATE ANY システム権限は、任意のユーザーのスキーマ内にオブジェクトを作成することを許可するための権限なので、通常、開発者には付与しません。この制限により、新しいオブジェクトを作成できるのが、開発者のユーザー・アカウントのみに限定されます。
- アプリケーション開発者が使用するロールにオブジェクト権限を付与することはほとんどありません。これは、ロールを介してオブジェクト権限を付与すると、他のオブジェクト（主にビューとストアド・プロシージャ）を作成するときの利便性が制限されるためです。むしろ、アプリケーション開発者が開発の目的で独自のオブジェクトを作成できるように許可する方が実用的です。

アプリケーション開発者に課される領域の制限

一般に、アプリケーション開発者は開発プロセスの一部としてオブジェクトを作成する権限が付与されていますが、セキュリティ管理者は、各アプリケーション開発者が使用可能なデータベースの領域とサイズの制限をメンテナンスする必要があります。たとえば、個々のアプリケーション開発者に次の制限を設定してください。

- 開発者が表または索引を作成できる表領域
- 開発者がアクセス可能な各表領域の割当て制限

どちらの制限も、開発者のセキュリティ・ドメインを変更することで設定できます。この操作については、24-6 ページの「ユーザーの変更」を参照してください。

アプリケーション管理者のセキュリティ

多くのデータベース・アプリケーションを使用する大規模なデータベース・システムでは、アプリケーション管理者が必要となる場合があります。アプリケーション管理者は、次のような種類の業務を担当します。

- アプリケーションに対するロールの作成と各アプリケーション・ロールの権限の管理
- データベース・アプリケーションによって使用されるオブジェクトの作成と管理
- 必要に応じて、アプリケーション・コードや Oracle プロシージャおよびパッケージのメンテナンスと更新

多くの場合、アプリケーション管理者は、アプリケーションを設計したアプリケーション開発者でもあります。ただし、データベース・アプリケーションに通じた人であれば、だれでもアプリケーション管理者にすることができます。

パスワード管理ポリシー

パスワードに依存しているデータベース・セキュリティ・システムでは、パスワードの機密を常に保つことが必要です。しかし、パスワードは盗難、偽造、悪用などに弱い性質を持っています。データベース・セキュリティの管理を強化するには、DBA およびセキュリティ管理者が、ユーザー・プロファイルを介して Oracle のパスワード管理ポリシーを制御する必要があります。

ユーザー・プロファイルを作成するには、`CREATE PROFILE` 文を使用します。プロファイルは、`CREATE USER` または `ALTER USER` 文を使用してユーザーに割り当てます。ここでは、データベース・ユーザーの作成と変更の詳細については説明しません。この項で取り上げるのは、`CREATE PROFILE`（または `ALTER PROFILE`）文で指定できるパスワード・パラメータです。

ここでは、Oracle のパスワード管理のうち次の側面について説明します。

- アカウントのロック
- パスワード・エイジングおよび期限切れ
- パスワード履歴
- パスワードの複雑度の検証

関連項目：

- 24-18 ページ「プロファイルによるリソースの管理」
- 24-2 ページ「Oracle ユーザーの管理」
- この項で説明する SQL 文の構文と固有の情報は、『Oracle9*i* SQL リファレンス』を参照してください。

アカウントのロック

特定のユーザーが、指定された回数以上ログインに失敗した場合、サーバーはそのユーザーのアカウントを自動的にロックします。DBA は、CREATE PROFILE 文を使用して、ログインの失敗が許容される回数を指定します。また、アカウントがロックされる時間の長さも指定できます。

次の例では、ユーザー ashwini に対して許容されているログイン失敗の最大回数は 4 回、アカウントがロックされる時間の長さは 30 日です。アカウントのロックは、30 日が経過すると自動的に解除されます。

```
CREATE PROFILE prof LIMIT
    FAILED_LOGIN_ATTEMPTS 4
    PASSWORD_LOCK_TIME 30;
ALTER USER ashwini PROFILE prof;
```

アカウントのロックを解除する時間間隔を指定しないと、PASSWORD_LOCK_TIME はデフォルト・プロファイルで指定されている値になります。PASSWORD_LOCK_TIME を UNLIMITED に指定した場合は、ALTER USER 文を使用して明示的にロックを解除する必要があります。たとえば、ashwini の PASSWORD_LOCK_TIME を UNLIMITED に指定したときは、次の文を使用してアカウントのロックを解除します。

```
ALTER USER ashwini ACCOUNT UNLOCK;
```

ユーザーが正常にアカウントにログインできると、そのユーザーが失敗したログインの回数は 0（ゼロ）にリセットされます。

セキュリティ管理者が明示的にユーザー・アカウントをロックすることもできます。そのようにした場合、アカウントのロックは自動的には解除されません。セキュリティ管理者がアカウントのロックを解除することが必要です。ユーザー・アカウントを明示的にロックまたはロック解除するには、CREATE USER 文または ALTER USER 文を使用します。たとえば、次の文はユーザー・アカウント susan をロックします。

```
ALTER USER susan ACCOUNT LOCK;
```

パスワード・エイジングおよび期限切れ

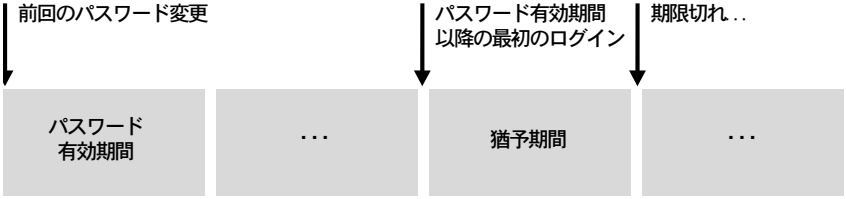
CREATE PROFILE 文を使用して、パスワードの最長有効期間を指定します。指定された時間が経過してパスワードの期限が切れると、ユーザーまたは DBA はパスワードを変更する必要があります。次の文は、プロファイルを作成し、ユーザー ashwini に割り当てます。PASSWORD_LIFE_TIME 句の指定により、ashwini はパスワードの期限が切れるまで 90 日間同じパスワードを使用できます。

```
CREATE PROFILE prof LIMIT
   FAILED_LOGIN_ATTEMPTS 4
   PASSWORD_LOCK_TIME 30
   PASSWORD_LIFE_TIME 90;
ALTER USER ashwini PROFILE prof;
```

パスワードの期限切れの猶予期間を指定することもできます。ユーザーは、パスワードの期限が切れた後、初めてデータベース・アカウントにログインしようとしたときに、猶予期間に入ります。猶予期間中は、ユーザーがアカウントにログインしようとするたびに警告メッセージが表示され、猶予期間が終わるまで表示され続けます。ユーザーは、猶予期間内にパスワードを変更する必要があります。猶予期間内にパスワードを変更しなければ、その後ユーザーがアカウントにアクセスしようとするたびに、新しいパスワードの入力を求められます。新しいパスワードを入力しないかぎり、アカウントへのアクセスは拒否されます。

図 23-2 は、パスワードの有効期間と猶予期間の推移を示しています。

図 23-2　パスワードの有効期間と猶予期間の推移



次の例では、ashwini に割り当てられたプロファイルに、猶予期間（PASSWORD_GRACE_TIME = 3）が指定されています。ashwini が 90 日経過後（90 日目以降の任意の日。70 日目、100 日目なども指定可能）初めてデータベースにログインしようとすると、パスワードが 3 日以内に期限切れになることを示す警告メッセージが表示されます。3 日経過してもパスワードを変更しない場合は、パスワードの期限が切れます。その後、ashwini はログインしようとするたびにパスワードの変更を求められ、パスワードを変更するまでログインできません。

```
CREATE PROFILE prof LIMIT
   FAILED_LOGIN_ATTEMPTS 4
   PASSWORD_LOCK_TIME 30
   PASSWORD_LIFE_TIME 90
   PASSWORD_GRACE_TIME 3;
ALTER USER ashwini PROFILE prof;
```

Oracle には、明示的にパスワードを期限切れにする手段が用意されています。この機能を提供するのは、CREATE USER 文と ALTER USER 文です。次の文は、期限切れのパスワードを持つユーザーを作成します。この設定により、ユーザーがデータベースにログイン可能になる前に、強制的にパスワードを変更させることができます。

```
CREATE USER jbrown
     IDENTIFIED BY zX83yT
     ...
     PASSWORD EXPIRE;
```

パスワード履歴

CREATE PROFILE 文を使用して、ユーザーがパスワードを再利用できない期間を指定します。次の文で定義されるプロファイルでは、PASSWORD_REUSE_TIME 句の指定により、ユーザーは 60 日間パスワードを再利用できません。

```
CREATE PROFILE prof LIMIT
   PASSWORD_REUSE_TIME 60
   PASSWORD_REUSE_MAX UNLIMITED;
```

次の文では、PASSWORD_REUSE_MAX 句によって、ユーザーが現在のパスワードを再利用できるようになるまでにパスワードを最低 3 回変更しなければならないことを指定しています。

```
CREATE PROFILE prof LIMIT
   PASSWORD_REUSE_MAX 3
   PASSWORD_REUSE_TIME UNLIMITED;
```

注意： PASSWORD_REUSE_TIME または PASSWORD_REUSE_MAX を指定する場合は、どちらか一方を UNLIMITED に設定するか、またはどちらも指定しないでください。

パスワードの複雑度の検証

Oracle のパスワード複雑度検証ルーチンは、デフォルトのプロファイル・パラメータを設定する PL/SQL スクリプト（UTLPWDMG.SQL）を使用して指定できます。

パスワード複雑度検証ルーチンは、次の点をチェックします。

- パスワードの長さが 4 文字以上であること。
- パスワードがユーザー名と同じでないこと。
- パスワードに少なくとも 1 つのアルファベット文字、1 つの数字および 1 つの句読点文字が含まれていること。
- パスワードが、welcome、account、database、user などの簡単または明白な単語でないこと。
- 以前のパスワードとの違いが 3 文字以上あること。

注意： ALTER USER 文はパスワード検証機能を完全にサポートしていないため、ALTER USER 文でパスワードを変更することはお薦めできません。かわりに、LNOCIPasswordChange() を使用してパスワードを変更してください。

パスワード検証ルーチンのフォーマットに関するガイドライン

既存のパスワード複雑度検証ルーチンを拡張したり、PL/SQL またはサード・パーティ製のツールを使用して独自のパスワード検証ルーチンを作成することが可能です。

PL/SQL のコールは、次のフォーマットに従う必要があります。

```
routine_name
(
userid_parameter IN VARCHAR(30),
password_parameter IN VARCHAR (30),
old_password_parameter IN VARCHAR (30)
)
RETURN BOOLEAN
```

新しいルーチンの作成後、ユーザーのプロファイルまたはシステムのデフォルト・プロファイルを使用して、それをパスワード検証ルーチンとして割り当てます。

```
CREATE/ALTER PROFILE profile_name LIMIT
PASSWORD_VERIFY_FUNCTION routine_name
```

パスワード検証ルーチンは、SYS が所有する必要があります。

パスワード検証ルーチンのサンプル

新しいパスワードに対する独自の複雑度チェックを開発する際は、このサンプルのパスワード検証ルーチンをモデルとして使用できます。

デフォルトのパスワード複雑度ファンクションは、次のような最低限の複雑度チェックを実行します。

- パスワードが長さの最低条件を満たしていること。
- パスワードがユーザー名ではないこと。このファンクションは、要件に基づいて変更できます。

このファンクションは SYS スキーマ内に作成する必要があるため、スクリプトを実行する前に connect SYS/*password* AS SYSDBA として接続します。

```
CREATE OR REPLACE FUNCTION verify_function
(username varchar2,
   password varchar2,
   old_password varchar2)
   RETURN boolean IS
   n boolean;
   m integer;
   differ integer;
   isdigit boolean;
   ischar  boolean;
   ispunct boolean;
   digitarray varchar2(20);
   punctarray varchar2(25);
   chararray varchar2(52);

BEGIN
   digitarray:= '0123456789';
   chararray:= 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
   punctarray:='!"#$%&()''*+,-/:;<=>?_';

--Check if the password is same as the username
IF password = username THEN
   raise_application_error(-20001, 'Password same as user');
END IF;

--Check for the minimum length of the password
IF length(password) < 4 THEN
   raise_application_error(-20002, 'Password length less than 4');
END IF;

--Check if the password is too simple. A dictionary of words may be
--maintained and a check may be made so as not to allow the words
--that are too simple for the password.
```

```
IF NLS_LOWER(password) IN ('welcome', 'database', 'account', 'user',
   'password', 'oracle', 'computer', 'abcd')
   THEN raise_application_error(-20002, 'Password too simple');
END IF;

--Check if the password contains at least one letter,
--one digit and one punctuation mark.
--1. Check for the digit
--You may delete 1. and replace with 2. or 3.
isdigit:=FALSE;
m := length(password);
FOR i IN 1..10 LOOP
  FOR j IN 1..m LOOP
    IF substr(password,j,1) = substr(digitarray,i,1) THEN
      isdigit:=TRUE;
        GOTO findchar;
    END IF;
   END LOOP;
END LOOP;
IF isdigit = FALSE THEN
  raise_application_error(-20003, 'Password should contain at least one \
  digit, one character and one punctuation');
END IF;
--2. Check for the character

<<findchar>>
ischar:=FALSE;
FOR i IN 1..length(chararray) LOOP
  FOR j IN 1..m LOOP
    IF substr(password,j,1) = substr(chararray,i,1) THEN
       ischar:=TRUE;
         GOTO findpunct;
       END IF;
    END LOOP;
END LOOP;
IF ischar = FALSE THEN
  raise_application_error(-20003, 'Password should contain at least one digit,\
    one character and one punctuation');
END IF;
--3. Check for the punctuation

<<findpunct>>
ispunct:=FALSE;
FOR i IN 1..length(punctarray) LOOP
  FOR j IN 1..m LOOP
    IF substr(password,j,1) = substr(punctarray,i,1) THEN
       ispunct:=TRUE;
```

```
          GOTO endsearch;
        END IF;
    END LOOP;
END LOOP;
IF ispunct = FALSE THEN raise_application_error(-20003, 'Password should \
 contain at least one digit, one character and one punctuation');
END IF;

<<endsearch>>
--Check if the password differs from the previous password by at least 3 letters
IF old_password = '' THEN
  raise_application_error(-20004, 'Old password is null');
END IF;
--Everything is fine; return TRUE ;
differ := length(old_password) - length(password);
IF abs(differ) < 3 THEN
  IF length(password) < length(old_password) THEN
    m := length(password);
  ELSE
    m:= length(old_password);
  END IF;
  differ := abs(differ);
  FOR i IN 1..m LOOP
    IF substr(password,i,1) != substr(old_password,i,1) THEN
             differ := differ + 1;
    END IF;
  END LOOP;
  IF differ < 3 THEN
    raise_application_error(-20004, 'Password should differ by at \
      least 3 characters');
    END IF;
  END IF;
--Everything is fine; return TRUE ;
  RETURN(TRUE);
END;
```

監査方針

セキュリティ管理者は、各データベースの監査手順の方針を定義する必要があります。たとえば、疑いのあるアクティビティが存在しているとは考えられない場合は、データベース監査を使用禁止にすることもできます。監査が必要な場合、セキュリティ管理者はデータベースの監査の詳細化レベルを決定する必要があります。通常、一般的なシステム監査では、不審なアクティビティを特定した後、さらに細かい監査を実行します。監査については、第 26 章「データベース使用の監査」を参照してください。

セキュリティ・チェックリスト

情報のセキュリティとプライバシー、および企業の資産とデータの保護は、どのようなビジネスにおいても非常に重要です。Oracle9*i* は、厳重なデータ保護、監査、スケーラブルなセキュリティ、安全なホスティング、データ交換などの最先端のセキュリティ機能を提供することにより、情報セキュリティのニーズに幅広く対応します。

Oracle9*i* データベース・サーバーは、セキュリティ機能において業界をリードしています。しかし、Oracle9*i* によって提供されるセキュリティ機能をどのようなビジネス環境でも最大限に活用するためには、Oracle9*i* 自体の保護が万全であることが必須条件です。また、セキュリティ機能を適切に使用し、基本的なセキュリティ・プラクティスに準拠することにより、データベースに関連した脅威や攻撃を防御でき、Oracle9*i* データベースの運用環境の安全性が高まります。

このセキュリティ・チェックリストは、運用データベースの配置に関する業界標準のベスト・セキュリティ・プラクティスに準拠し、それを推奨することにより、セキュリティで保護されたデータベースとして Oracle9*i* を構成するためのガイドラインを示しています。

データベース関連の特定のタスクおよびアクションに関する詳細は、Oracle のドキュメント・セット全体を通じて記載されています。

1. 必要なもののみをインストールする

Oracle9*i* の CD パックには、データベース・サーバーだけでなく、多くのオプションと製品が収録されています。追加の製品やオプションは必要に応じてインストールしてください。または、標準インストール（カスタム・インストールを行わない場合）に従ってインストールしてから、不要なオプションや製品を削除してください。追加の製品やオプションを使用しない場合は、それらをメンテナンスする必要はありません。追加の製品やオプションは、必要なときにいつでも適切かつ容易に再インストールできます。

2. デフォルトのユーザー・アカウントをロックし、期限切れにする

Oracle9*i* をインストールすると、データベース・サーバーのデフォルト（事前設定）のユーザー・アカウントがいくつか作成されます。Database Configuration Assistant（DBCA）ツールは、データベース・サーバーが正常にインストールされたときに、次のアカウントを除くデフォルトのデータベース・ユーザー・アカウントをすべて自動的にロックし、期限切れにします。

- `SYS`
- `SYSTEM`
- `SCOTT`
- `DBSNMP`

DBCA を利用せず、手動で Oracle9*i* をインストールした場合は、データベース・サーバーが正常にインストールされたときに、デフォルトのデータベース・ユーザーは 1 つもロックされません。これらのデータベース・ユーザーをデフォルト状態のままにしておくと、そのユーザー・アカウントを悪用してデータに不正にアクセスしたり、データベース操作を妨害できます。DBCA を利用せずに初期インストールを実行した場合は、インストール後に、SYS、SYSTEM、SCOTT および DBSNMP を除くデフォルトのデータベース・ユーザー・アカウントをすべてロックし、期限切れにしてください。Oracle9*i* では、この操作を行うために SQL が用意されています。

次の表は、DBCA を利用して「General Purpose」タイプのデータベースのインストールを行った後のデータベース・ユーザーを示しています。

USERNAME	ACCOUNT_STATUS
ANONYMOUS	EXPIRED & LOCKED
CTXSYS	EXPIRED & LOCKED
DBSNMP	OPEN
HR	EXPIRED & LOCKED
MDSYS	EXPIRED & LOCKED
ODM	EXPIRED & LOCKED
ODM_MTR	EXPIRED & LOCKED
OE	EXPIRED & LOCKED
OLAPSYS	EXPIRED & LOCKED
ORDPLUGINS	EXPIRED & LOCKED
ORDSYS	EXPIRED & LOCKED
OUTLN	EXPIRED & LOCKED
PM	EXPIRED & LOCKED
QS	EXPIRED & LOCKED
QS_ADM	EXPIRED & LOCKED
QS_CB	EXPIRED & LOCKED
QS_CBADM	EXPIRED & LOCKED
QS_CS	EXPIRED & LOCKED
QS_ES	EXPIRED & LOCKED
QS_OS	EXPIRED & LOCKED
QS_WS	EXPIRED & LOCKED

USERNAME	ACCOUNT_STATUS
RMAN	EXPIRED & LOCKED
SCOTT	OPEN
SH	EXPIRED & LOCKED
SYS	OPEN
SYSTEM	OPEN
WKPROXY	EXPIRED & LOCKED
WKSYS	EXPIRED & LOCKED
WMSYS	EXPIRED & LOCKED
XDB	EXPIRED & LOCKED

なんらかの理由で、この表で OPEN 以外の状態にあるデフォルト・データベース・サーバー・ユーザー・アカウントが必要な場合、DBA は単にそのアカウントのロックを解除し、意味のある新しいパスワードを設定してアカウントをアクティブにします。

3. デフォルト・ユーザーのパスワードを変更する

最も平凡な手段でありながら、Oracle9*i* を危険にさらす可能性があるのは、インストール後もデフォルトのパスワードを変更していないデフォルトのデータベース・サーバー・ユーザー・アカウントです。

a. 管理用ユーザーのデフォルト・パスワードを変更する

Oracle9*i* をインストールしたとき、SYS にはデフォルト・パスワードとして CHANGE_ON_INSTALL、SYSTEM にはデフォルト・パスワードとして MANAGER が設定されます。ユーザー SYS と SYSTEM のデフォルト・パスワードは、データベース・サーバーのインストール時にただちに変更してください。

b. すべてのユーザーのデフォルト・パスワードを変更する

Oracle9*i* をインストールしたとき、SCOTT にはデフォルト・パスワードとして TIGER が設定されます。他のアカウントのデフォルト・パスワードは、ユーザー・アカウントとまったく同じです（たとえば、ユーザー MDSYS のパスワードは MDSYS）。

ユーザー・アカウント SCOTT および DBSNMP のパスワードも同様に、データベース・サーバーのインストール時にただちに変更してください。インストール時にロックされ、期限切れになっている他のデフォルト・ユーザー・アカウントのいずれかをアクティブにする場合は、そのユーザー・アカウントに意味のある新しいパスワードを割り当てます。

このチェックリストではユーザー SCOTT のデフォルト・パスワードの変更を明示的に規定していますが、たとえそうでなくても、このユーザー・アカウントは、目的を持って使用する場合を除き、ロックすることをお薦めします。

c. パスワード管理を徹底する

データベースによって提供される基本的なパスワード管理ルール（パスワードの長さ、履歴、複雑度など）をすべてのユーザー・パスワードに適用し、すべてのユーザーに対して定期的なパスワードの変更を必須とすることをお薦めします。

また、可能であれば、Oracle Advanced Security（Oracle9*i* Enterprise Edition のオプション）を利用して、ネットワーク認証サービス（Kerberos など）、トークン・カード、スマート・カードまたは X.509 証明書を使用することもお薦めします。これらのサービスを使用すると、ユーザーの厳密認証が可能になるので、Oracle9*i* を不正アクセスから適切に保護できます。

4. データ・ディクショナリ保護の有効化

ANY システム権限を持つユーザーがデータ・ディクショナリに対してそれらの権限を使用しないように、データ・ディクショナリの保護を実装することをお薦めします。

ディクショナリ保護を有効にするには、次のように O7_DICTIONARY_ACCESSIBILITY 初期化パラメータを設定します。

```
O7_DICTIONARY_ACCESSIBILITY = FALSE
```

このパラメータを設定すると、DBA 権限付きで接続可能な（たとえば、CONNECT / AS SYSDBA）許可されたユーザーのみがデータ・ディクショナリに対する ANY システム権限を使用できます。このパラメータを前述の推奨値に設定しなければ、たとえば DROP ANY TABLE システム権限を持つユーザーであれば誰でも、悪意を持ってデータ・ディクショナリの一部を削除できます。

ただし、データ・ディクショナリへの参照アクセスが必要なユーザーに SELECT ANY DICTIONARY システム権限を付与することは許容されます。

Oracle9*i* では、デフォルトで O7_DICTIONARY_ACCESSIBILITY = FALSE に設定されています。Oracle8*i* ではこのパラメータがデフォルトで TRUE に設定されているので、このセキュリティ機能を有効にするには、明示的に FALSE に変更する必要があります。

5. 「最低限の権限」原則の実践

a. 必要な権限のみを付与する

データベース・ユーザーに必要以上の権限を付与しないでください。「最低限の権限」原則とは、言い換えれば、ユーザーに付与する権限を、各ユーザーが業務を効率よく容易に遂行するために実際に必要となる権限のみに限定するということです。

最低限の権限を実装するには、次のものを制限します。1) データベース・ユーザーに付与する SYSTEM 権限と OBJECT 権限の数、2) データベースで許可される SYS

権限付き接続の最大数。たとえば、一般には、DBA 権限を持たないユーザーに CREATE ANY TABLE を付与する必要はありません。

b. PUBLIC から不要な権限を取り消す

データベース・サーバーのユーザー・グループ PUBLIC から不要な権限およびロールをすべて削除してください。PUBLIC は、Oracle データベース内のすべてのユーザーに付与されるデフォルト・ロールです。PUBLIC に付与されている権限は、すべてのデータベース・ユーザーが行使できます。この種の権限には、たとえば、各種 PL/SQL パッケージの EXECUTE 権限があります。この権限があれば、最低限の権限しか持たないユーザーが、直接アクセスすることを許可されていないパッケージにアクセスし、実行できます。次の表に、悪用される可能性がある高機能のパッケージを示します。

パッケージ	説明
UTL_SMTP	このパッケージを使用すると、任意のユーザー間で任意のメール・メッセージを送信できます。このパッケージを PUBLIC に付与すると、メール・メッセージの不正な交換が可能になります。
UTL_TCP	このパッケージを使用すると、データベース・サーバーから任意の受信（待機）ネットワーク・サービスに対して、発信ネットワーク接続を確立できます。これにより、データベース・サーバーと待機ネットワーク・サービスの間で任意のデータを送信できます。
UTL_HTTP	このパッケージを使用すると、データベース・サーバーが HTTP を使用してデータを要求し、取得できます。このパッケージを PUBLIC に付与すると、HTML 形式を使用して悪意のある Web サイトにデータを送信できます。
UTL_FILE	このパッケージの構成が不適切な場合は、ホスト・オペレーティング・システムの任意のファイルにテキスト・レベルでアクセスできます。このパッケージは適切に構成していても、そのコール側アプリケーションを識別しないため、UTL_FILE にアクセスしたアプリケーションによって、別のアプリケーションが書き込んだのと同じ場所に任意のデータを書き込むことができます。
DBMS_RANDOM	このパッケージを使用すると、格納されているデータを暗号化できます。一般に、キーが安全に生成、格納および管理されていない場合、暗号化されたデータはリカバリできないことがあるため、ほとんどのユーザーにはデータを暗号化する権限は付与しません。

これらのパッケージは、その機能を必要とし、適切に構成して使用する一部のアプリケーションにとっては非常に有用です。それ以外のアプリケーションには適していません。したがって、どうしても必要でないかぎり、これらのパッケージは PUBLIC から削除してください。

c. ランタイム機能の権限を制限する

Oracle Java Virtual Machine（OJVM）など、データベース・サーバーのランタイム機能にすべての権限を割り当てないでください。データベース・サーバーの外部にあるファイルやパッケージを実行するランタイム機能については、特定の権限を明示的なドキュメント・ルート・ファイル・パスに設定してください。

無防備なランタイム・コールの例：

```
call dbms_java.grant_permission('SCOTT', 'SYS:java.io.FilePermission','<<ALL
FILES>>','read');
```

適切な（安全性の高い）ランタイム・コールの例：

```
call dbms_java.grant_permission('SCOTT',
'SYS:java.io.FilePermission','<<actual directory path>>','read');
```

6. アクセス制御を効果的に実行する

クライアントを適切に認証します。

リモート認証は、Oracle9*i* に組み込まれているセキュリティ機能の 1 つです。この機能を使用可能（TRUE）にすると、リモート・クライアントが Oracle データベースに接続するまでユーザーの認証が遅延されます。そのため、データベースはデータベース自体を適切に認証するために、暗黙的にすべてのクライアントを信頼します。一般に、オペレーティング・システム認証を適切に実行するために、PC などのクライアントは信頼されていません。したがって、この機能を使用可能にすることは、セキュリティの実施として適切ではありません。

この機能を使用禁止（FALSE）にした安全性の高い構成では、Oracle データベースに接続するクライアントがサーバーベースで適切に認証されます。

リモート認証を制限して、クライアントの信頼をデータベースに遅延するには、次のように REMOTE_OS_AUTHENT 初期化パラメータを設定します。

```
REMOTE_OS_AUTHENT = FALSE
```

7. オペレーティング・システムへのアクセスを制限する

オペレーティング・システム・ユーザーの数を制限します。

Oracle9*i* が稼働しているホスト（物理的なマシン）上のオペレーティング・システム・アカウントの権限（管理、ルート権限または DBA）を、ユーザーに必要最小限の権限に制限してください。

また、次のこともお薦めします。

- Oracle ホーム（インストール）ディレクトリやその内容について、デフォルトのファイルやディレクトリのアクセス権を変更できる権限を制限します。オラクル社から特に指示されないかぎり、権限を持つオペレーティング・システム・ユーザーと Oracle 所有者は、これらのアクセス権を変更しないでください。

- データベースへのパスやファイルを指定する場合には、そのファイルやパスのどの部分も、信頼のおけないユーザーが変更できないことを確認します。ファイルやパスのすべての構成要素は、DBA またはルートなどの信頼できるアカウントが所有する必要があります。この推奨事項は、データ・ファイル、ログ・ファイル、トレース・ファイル、外部表、BFILE など、あらゆるタイプのファイルに適用されます。

8. ネットワーク・アクセスを制限する

a. ファイアウォールを利用する

データベース・サーバーはファイアウォールの内側に配置してください。Oracle9*i* のネットワーク・インフラストラクチャである Oracle Net（旧称 Net8 および SQL*Net）は、様々なベンダーの各種ファイアウォールのサポートを提供しています。プロキシ対応のファイアウォールでは、Network Associates 社の Gauntlet と Axent 社の Raptor をサポートしています。パケット・フィルタ型のファイアウォールでは Cisco 社の PIX Firewall、ステートフル・インスペクション型のファイアウォール（より高機能のパケット・フィルタ型ファイアウォール）では CheckPoint 社の Firewall-1 をサポートしています。

b. ファイアウォールに穴を作らない

Oracle9*i* がファイアウォールの内側にある場合は、どのような環境でも、ファイアウォールに穴を作らないでください。たとえば、インターネットへの接続またはインターネットからの接続を確立するリスナーの 1521 番ポートを通過可能にしないでください。

ファイアウォールに穴があると、ファイアウォールを通過するポートの穴をさらに増やされたり、マルチスレッド・オペレーティング・システム・サーバーの問題点を攻撃されたり、ファイアウォールの内側にあるデータベースの重要な情報が漏洩するなど、セキュリティが著しく脆弱になります。また、パスワードを設定せずにリスナーを稼働していると、トレースやロギング情報、バナー情報、データベース記述子、サービス名など、リスナーがリスニングしているデータベースに関する重要な情報が入手されるおそれがあります。

このような多くの情報が入手され、不適切な構成のファイアウォールが利用されると、十分にターゲット・データベースへ不正な攻撃を仕掛けることが可能になります。

c. リスナーの不正な管理を防ぐ

リスナーのリモート構成を防ぐため、リスナーにはパスワードとして適切で意味のあるものを必ず設定してください。また、`listener.ora`（リスナーの制御ファイル）のセキュリティ構成パラメータを次のように設定してください。

```
ADMIN_RESTRICTIONS_listener_name = ON
```

これらを設定することにより、リスナーの不正な管理を防ぐことができます。

d. ネットワークの IP アドレスをチェックする

Oracle Net の「有効ノード・チェック」セキュリティ機能を利用すれば、指定の IP アドレスを持つネットワーク・クライアントから Oracle サーバー・プロセスへのアクセスを許可または拒否できます。この機能を使用するには、`protocol.ora`（Oracle Net の構成ファイル）のパラメータを次のように設定します。

```
tcp.validnode_checking = YES

tcp.excluded_nodes = {list of IP addresses}

tcp.invited_nodes = {list of IP addresses}
```

最初のパラメータは、有効ノード・チェック機能を使用可能にします。残り 2 つのパラメータは、リスナーに接続しようとする特定のクライアント IP アドレスをそれぞれ拒否または許可します。これにより、サービス拒否攻撃を可能なかぎり防御できます。

e. ネットワーク通信を暗号化する

可能であれば、Oracle Advanced Security を利用して、クライアント、データベースおよびアプリケーション・サーバーの間のネットワーク通信を暗号化してください（Oracle Advanced Security を利用できるのは、Oracle データベースの Enterprise Edition のみです）。

f. オペレーティング・システムを強化する

オペレーティング・システムの不要なサービスをすべて使用禁止にして、ホスト・オペレーティング・システムを強化してください。UNIX と Windows プラットフォームにはどちらも様々なオペレーティング・システム・サービスが組み込まれていますが、それらの大部分は、標準的なデータベース構成では必要ありません。この種のサービスには、FTP、TFTP、TELNET などがあります。使用禁止にしている各サービスの UDP ポートと TCP ポートは、両方とも必ず閉じてください。一方のポートを使用禁止にしていても、他方のポートが使用可能であれば、オペレーティング・システムの安全性が低下します。

9. セキュリティ・パッチと回避策をすべて適用する

Oracle9*i* を稼働しているオペレーティング・システム、Oracle9*i* 自体、およびインストール済みの Oracle9*i* のオプションとそのコンポーネントすべてについて、関連する最新のセキュリティ・パッチをすべて適用してください。

最新のセキュリティ情報の詳細および問題回避のための手段（もしくは個別パッチ）に関しては以下のサイトに掲載されている情報を参照してください。

`http://www.oracle.co.jp/news/security`

24

ユーザーとリソースの管理

この章では、Oracle データベースへのアクセスの制御方法を説明します。この章の内容は、次のとおりです。

- Oracle ユーザーの管理
- ユーザー認証方式
- プロファイルによるリソースの管理
- データベース・ユーザーとプロファイルに関する情報の表示

関連項目：

- 第 23 章「セキュリティ・ポリシーの設定」
- 第 25 章「ユーザー権限とロールの管理」

Oracle ユーザーの管理

各 Oracle データベースには、有効なデータベース・ユーザーのリストがあります。ユーザーがデータベースにアクセスするには、データベース・アプリケーションを稼働させた上で、データベースに定義されている有効なユーザー名を使用して、データベース・インスタンスに接続する必要があります。ここでは、データベース・ユーザーの管理方法について説明します。この項の内容は、次のとおりです。

- ユーザーの作成
- ユーザーの変更
- ユーザーの削除

関連項目： ユーザーの管理に使用する SQL 文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

ユーザーの作成

データベース・ユーザーを作成するには、CREATE USER 文を使用します。ユーザーを作成するには、CREATE USER システム権限が必要です。CREATE USER は強力なシステム権限のため、通常、この権限を持つユーザーは DBA またはセキュリティ管理者のみです。

次の例では、ユーザーを作成し、そのユーザーのパスワード、デフォルト表領域、一時セグメントが作成される一時表領域、表領域割当て制限およびプロファイルを指定しています。

```
CREATE USER jward
    IDENTIFIED BY aZ7bC2
    DEFAULT TABLESPACE data_ts
    QUOTA 100M ON test_ts
    QUOTA 500K ON data_ts
    TEMPORARY TABLESPACE temp_ts
    PROFILE clerk;
GRANT connect TO jward;
```

新しく作成したユーザーは、CREATE SESSION システム権限を付与するまで、データベースに接続できません。通常、新しく作成するユーザーには、事前定義済みのロール CONNECT（この例で使用されています）に似たロールを付与します。このロールは、データベースへのアクセスに必要な CREATE SESSION と他の基本的な権限を指定します。

ここでは、上の例を参照しながら、ユーザーを作成するための次の操作について説明します。

- 名前の指定
- ユーザー認証の設定
- デフォルト表領域の割当て
- 表領域割当て制限の割当て
- 一時表領域の割当て
- プロファイルの指定
- デフォルト・ロールの設定

関連項目： 25-11 ページ「システム権限とロールの付与」

名前の指定

各データベース内のユーザー名は、他のユーザー名およびロールと比較して一意であることが必要です。ユーザーとロールに同じ名前を付けることはできません。また、各ユーザーには対応するスキーマがあります。スキーマ内の各スキーマ・オブジェクトには、必ず一意の名前を指定する必要があります。

ユーザー認証の設定

前述の CREATE USER 文では、データベースを使用して新規ユーザーを認証しています。この場合は、接続に成功するために、接続ユーザーがデータベースに対して正しいパスワードを指定する必要があります。

ユーザー認証方式の選択と指定については、24-9 ページの「ユーザー認証方式」を参照してください。

デフォルト表領域の割当て

各ユーザーには、デフォルト表領域が必要です。ユーザーがスキーマ・オブジェクトを作成したときに、格納する表領域を指定しない場合、そのオブジェクトはユーザーのデフォルト表領域に格納されます。

ユーザーのデフォルト表領域のデフォルト設定は、SYSTEM 表領域です。ユーザーがオブジェクトを作成せず、オブジェクトを作成するための権限も持っていない場合は、デフォルト設定のままで問題ありません。しかし、ユーザーが任意のタイプのオブジェクトを作成できる場合は、そのユーザーにデフォルト表領域を明示的に割り当てる必要があります。SYSTEM 以外の表領域を使用すると、同じデータ・ファイルに対するデータ・ディクショナリ・オブジェクトとユーザー・オブジェクト間の競合が解消されます。一般に、ユーザー・データを SYSTEM 表領域に格納することはお薦めできません。

ユーザー作成時にユーザーのデフォルト表領域を設定しておき、作成後に ALTER USER 文を使用して変更できます。ユーザーのデフォルト表領域を変更すると、設定の変更後に作成されたオブジェクトのみがこの変更の影響を受けます。

ユーザーのデフォルト表領域を指定するときは、その表領域に対する割当て制限も併せて指定してください。

前述の CREATE USER 文では、jward のデフォルト表領域は data_ts であり、その表領域に対する jward の割当て制限は 500KB です。

表領域割当て制限の割当て

各ユーザーには、任意の表領域（一時表領域を除く）に対する表領域割当て制限を設定できます。割当て制限による影響は、次のとおりです。

- 特定のタイプのオブジェクト作成権限を持つユーザーは、指定した表領域内にオブジェクトを作成できます。
- 指定した表領域内にあるユーザーのオブジェクトの記憶域に対して割当て可能な領域が、割当て制限以内に制限されます。

デフォルトでは、ユーザーにはデータベース内の表領域に関する割当て制限はありません。ユーザーにスキーマ・オブジェクトを作成する権限がある場合は、必ず割当て制限を設定して、ユーザーがオブジェクトを作成できるようにします。少なくとも、デフォルト表領域に対する割当て制限をユーザーに割り当てます。さらに、オブジェクトを作成可能な他の表領域に対して追加の割当て制限を割り当てます。

ユーザーの割当て制限として、各表領域の一定量のディスク領域を個別に割り当てるか、またはすべての表領域のディスク領域を無制限に割り当てるかのどちらかを選択できます。一定量の割当て制限を設定すれば、ユーザーのオブジェクトがデータベースの領域を大幅に消費することを防ぐことができます。

ユーザーの表領域の割当て制限は、ユーザーの作成時に設定し、作成後に追加または変更できます。新しい割当て制限が古い制限値よりも小さい場合は、次の条件が成り立ちます。

- ユーザーがすでに新しい表領域割当て制限を超過している場合、これらのオブジェクトを合せた領域が新しい割当て制限を下回らないかぎり、その表領域のユーザー・オブジェクトに追加の領域を割り当てられません。
- ユーザーが新しい表領域割当て制限を超過していない場合、つまり表領域内でユーザーのオブジェクトが使用している領域が新しい表領域割当て制限よりも少ない場合は、そのユーザーのオブジェクトに新しい割当て制限までの領域を割り当てることができます。

ユーザーから表領域内にオブジェクトを作成する許可を取り消す方法 ユーザーが表領域内にオブジェクトを作成するための許可を取り消すには、そのユーザーの現行の割当て制限を 0（ゼロ）に変更します。割当て制限を 0（ゼロ）に変更した場合、表領域内のユーザーのオブジェクトはそのまま残りますが、新しいオブジェクトを作成することはできず、既存のオブジェクトに新たな領域を割り当てることもできません。

UNLIMITED TABLESPACE システム権限 データベース内の表領域をユーザーが無制限に使用することを許可するには、ユーザーに UNLIMITED TABLESPACE システム権限を付与します。これにより、ユーザーに対して明示的に指定されている表領域割当て制限がすべて上書きされます。この権限を後で取り消すと、明示的に指定された割当て制限が再び有効になります。この権限はロールに対してではなく、ユーザーに対してのみ付与できます。

UNLIMITED TABLESPACE システム権限を付与する前に、この方法のメリットとデメリットを考慮してください。

メリット

- データベースのすべての表領域に無制限にアクセスする権限を 1 つの文でユーザーに付与できます。

デメリット

- この権限によって、そのユーザーに対する明示的な表領域割当て制限がすべて置き換えられます。
- UNLIMITED TABLESPACE システム権限を持つユーザーから表領域へのアクセスを選択的に取り消すことはできません。その権限を取り消した後にかぎり、選択的にアクセスを付与できます。

一時表領域の割当て

各ユーザーには、一時表領域を割り当てる必要があります。ユーザーが一時セグメントを必要とする SQL 文を実行すると、このセグメントはユーザーの一時表領域に格納されます。これらの一時セグメントは、ソートまたは結合の実行時にシステムによって作成され、すべての表領域の RESOURCE 権限を持つ SYS の所有となります。

前述の CREATE USER 文では、jward の一時表領域は temp_ts です。これは一時セグメントのみを格納するために明示的に作成された表領域です。一時表領域は、CREATE TEMPORARY TABLESPACE 文を使用して作成します。

ユーザーの一時表領域を明示的に設定しない場合、そのユーザーには、データベースの作成時、またはその後 ALTER DATABASE 文によって指定されたデフォルト一時表領域が割り当てられます。一時表領域が存在しない場合、デフォルトは SYSTEM 表領域になります。ユーザー・データを SYSTEM 表領域に格納することはお薦めできません。また、一時表領域として使用する表領域を明示的に割り当てることにより、一時セグメントとそれ以外のタイプのセグメントとの間で発生するファイルの競合が解消されます。

注意： SYSTEM 表領域がローカル管理表領域の場合、ユーザーには特定のデフォルト（ローカル管理）一時表領域を割り当て、SYSTEM 表領域の使用をデフォルトで禁止する必要があります。これは、永続的なローカル管理表領域には、一時オブジェクトを格納できないためです。

ユーザー作成時にユーザーの一時表領域を設定しておき、作成後に ALTER USER 文を使用して変更できます。一時表領域には、割当て制限を設定しないでください。

関連項目：

- 11-11 ページ「一時表領域」
- 2-25 ページ「デフォルト一時表領域の作成」

プロファイルの指定

ユーザーの作成時にプロファイルも指定できます。プロファイルとは、データベース・リソースと、データベースへのパスワード・アクセスに関する制限のセットです。プロファイルを指定しなければ、そのユーザーにはデフォルト・プロファイルが割り当てられます。

関連項目：

- 24-18 ページ「プロファイルによるリソースの管理」
- 23-12 ページ「パスワード管理ポリシー」

デフォルト・ロールの設定

CREATE USER 文ではユーザーのデフォルト・ロールを設定できません。最初にユーザーを作成したときに、ユーザーのデフォルト・ロールは ALL に設定されます。これにより、その後ユーザーに付与されるロールはすべてデフォルト・ロールになります。ユーザーのデフォルト・ロールを変更するには、ALTER USER 文を使用します。

関連項目： 25-21 ページ「デフォルト・ロールの指定」

ユーザーの変更

ユーザーは自分のパスワードを変更できます。ただし、ユーザーのセキュリティ・ドメインの他のオプションを変更するには、ALTER USER システム権限が必要です。通常は、セキュリティ管理者のみがこのシステム権限を持ちます。これは、この権限があればすべてのユーザーのセキュリティ・ドメインを変更できるためです。この権限には、データベースの任意の表領域に対するユーザーの表領域割当て制限を設定する許可が含まれています。これは、変更を実行するユーザーが、指定した表領域に対する割当て制限を持っていなくても同じです。

ユーザーのセキュリティ設定を変更するには、ALTER USER 文を使用します。ユーザーのセキュリティ設定の変更は、現行セッションではなく、それ以降のセッションから反映されます。

次の文は、ユーザー avyrros のセキュリティ設定を変更します。

```
ALTER USER avyrros
    IDENTIFIED EXTERNALLY
    DEFAULT TABLESPACE data_ts
    TEMPORARY TABLESPACE temp_ts
    QUOTA 100M ON data_ts
    QUOTA 0 ON test_ts
    PROFILE clerk;
```

この ALTER USER 文によって、avyrros のセキュリティ設定は次のように変更されます。

- 認証方式として、avyrros のオペレーティング・システム・アカウントが使用されます。
- avyrros のデフォルト表領域と一時表領域が明示的に設定されます。
- data_ts 表領域に対する avyrros の割当て制限が 100MB になります。
- test_ts に対する avyrros の割当て制限が取り消されます。
- avyrros が clerk プロファイルに割り当てられます。

ユーザーの認証メカニズムの変更

DBA でなくてもほとんどのユーザーが、次のように ALTER USER 文を使用して自分のパスワードを変更できます。

```
ALTER USER andy
   IDENTIFIED BY swordfish;
```

ユーザーがパスワードを変更するには、特別な権限（データベースへの接続権限以外）は不要です。ユーザーには、自分のパスワードを頻繁に変更するように薦める必要があります。

ユーザーが認証方式を切り替えるには、ALTER USER 権限が必要です。通常、この権限を持つのは管理者のみです。

> **関連項目：** Oracle ユーザーが使用可能な認証方式の詳細は、24-9 ページの「ユーザー認証方式」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

デフォルト・ロールとは、ユーザーがセッションを作成したときに、自動的にそのユーザーに対して使用可能になるロールです。ユーザーには任意の数のデフォルト・ロールを割り当てることができます。まったく割り当てなくてもかまいません。

> **関連項目：** ユーザーのデフォルト・ロールの変更に関する詳細は、第 25 章「ユーザー権限とロールの管理」を参照してください。

ユーザーの削除

ユーザーを削除すると、そのユーザーおよび対応するスキーマがデータ・ディクショナリから削除されます。また、ユーザーのスキーマ内にスキーマ・オブジェクトがある場合は、そのすべてのオブジェクトがただちに削除されます。

注意： ユーザーのスキーマおよびそれに対応するオブジェクトは残したままで、ユーザーのデータベースへのアクセスを拒否する場合は、そのユーザーから CREATE SESSION 権限を取り消してください。

現在データベースに接続されているユーザーは削除できません。接続中のユーザーを削除するには、最初に KILL SESSION 句を指定した SQL 文 ALTER SYSTEM を使用して、そのユーザーのセッションを停止します。

データベースからユーザーを削除するには、DROP USER 文を使用します。ユーザーとそのユーザーのスキーマ・オブジェクト（ある場合）をすべて削除するには、DROP USER システム権限が必要です。DROP USER システム権限は非常に強力な権限なので、通常はセキュリティ管理者のみがこの権限を持ちます。

ユーザーのスキーマにスキーマ・オブジェクトが含まれている場合、ユーザー、対応付けられているすべてのオブジェクトおよびそのユーザーの表に依存している外部キーをすべて削除するには、CASCADE オプションを使用します。CASCADE を指定していない場合、ユーザーのスキーマにオブジェクトが含まれていると、エラー・メッセージが返され、ユーザーは削除されません。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、事前にどのオブジェクトがユーザーのスキーマに含まれているかを十分に調査し、削除による影響を確認しておく必要があります。CASCADE によって、事前に認識していなかった影響が表れることがあるので、注意してください。たとえば、表を所有するユーザーを削除する場合は、ビューまたはプロシージャがその表に依存していないかどうかを確認してください。

次の文は、ユーザー jones と、jones に対応付けられたすべてのオブジェクト、および jones が所有する表に依存するすべての外部キーを削除します。

```
DROP USER jones CASCADE;
```

関連項目： セッションの終了方法の詳細は、5-21 ページの「セッションの停止」を参照してください。

ユーザー認証方式

Oracle では、ユーザーにデータベース・セッションの作成を許可するために、次のような方法でユーザーを認証できます。

1. データベースでユーザーの識別と認証の両方が実行されるように、ユーザーを定義できます。これを**データベース認証**と呼びます。
2. オペレーティング・システムまたはネットワーク・サービスによって認証が実行されるように、ユーザーを定義できます。これを**外部認証**と呼びます。
3. Secure Sockets Layer（SSL）によって**グローバルに**認証されるように、ユーザーを定義できます。このようなユーザーを**グローバル・ユーザー**と呼びます。グローバル・ユーザーの場合は、エンタープライズ・ディレクトリを使用して、**グローバル・ロール**を介したデータベースへのアクセスを許可できます。
4. 中間層サーバーを介して接続が許可されるユーザーを指定できます。中間層サーバーはユーザーの識別情報を認証して引き継ぎ、ユーザーに対して特定のロールを使用可能にできます。これを**プロキシ認証**および認可と呼びます。

ここでは、次の認証方式について説明します。

- データベース認証
- 外部認証
- グローバル認証および認可
- プロキシ認証および認可

データベース認証

ユーザーに対してデータベース認証を選択すると、そのユーザーのユーザー・アカウントの管理は、認証を含めてすべて Oracle が行います。Oracle によってユーザーを認証するには、ユーザーを登録または変更するときに、そのユーザーのパスワードを指定します。ユーザーはいつでも自分のパスワードを変更できます。パスワードは暗号形式で格納されます。データベースがマルチバイト・キャラクタ・セットを使用している場合でも、各パスワードはシングルバイト・キャラクタで構成する必要があります。

注意： ユーザー名とパスワードのエンコーディングには、プラットフォームに応じて ASCII 文字または EBCDIC 文字のみを使用することをお薦めします。これにより、データベース・キャラクタ・セットに将来変更があった場合にも、それをサポートするための互換性が保たれます。

新規ターゲット・キャラクタ・セットへの移行時にサイズ拡張を伴う文字を使用してユーザー名またはパスワードが作成されている場合、移行後は、そのユーザーは認証失敗によりログインできなくなる可能性があります。これは、暗号化されてデータ・ディクショナリに格納されているユーザー名とパスワードは、新しいデータベース・キャラクタ・セットへの移行中に更新されないためです。

たとえば、現行のデータベース・キャラクタ・セットが WE8MSWIN1252 で、ターゲット・データベース・キャラクタ・セットが UTF8 の場合、ユーザー名 `scött`（ウムラウト付きの o）は UTF8 では 5 バイトから 6 バイトに変わります。ユーザー `scött` はログインできなくなります。

ユーザー名とパスワードが ASCII 文字または EBCDIC 文字に基づいていない場合は、新しいキャラクタ・セットに移行するときに、影響を受けるユーザー名とパスワードを再設定する必要があります。

データベース認証使用時のセキュリティを高めるために、アカウントのロック、パスワードのエージングと期限切れ、パスワードの履歴およびパスワードの複雑度の検証といったパスワード管理の使用をお薦めします。

関連項目： 23-12 ページ 「パスワード管理ポリシー」

データベースによって認証されるユーザーの作成

次の文は、Oracle によって識別および認証されるユーザーを作成します。ユーザー `scott` は、Oracle に接続するたびにパスワード `tiger` を指定する必要があります。

```
CREATE USER scott IDENTIFIED BY tiger;
```

関連項目： 有効なパスワードと、`CREATE USER` 文および `ALTER USER` 文に `IDENTIFIED BY` 句を指定する方法の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

データベース認証の利点

次に、データベース認証の利点を示します。

- ユーザー・アカウントとすべての認証がデータベースによって制御されます。データベースの外部のものには依存しません。
- Oracle には、データベース認証使用時のセキュリティを高めるために、強力なパスワード管理機能が組み込まれています。

- 小さいユーザー・コミュニティがある場合の管理が容易になります。

外部認証

ユーザーに対して外部認証を選択すると、ユーザー・アカウントは Oracle でメンテナンスされますが、パスワード管理とユーザー認証は外部サービスによって実行されます。この外部サービスは、オペレーティング・システムでも Oracle Net のようなネットワーク・サービスでもかまいません。

外部認証を使用すると、データベースはデータベース・アカウントへのアクセスの制限をその基礎となるオペレーティング・システムまたはネットワーク認証サービスに依存します。データベース・パスワードは、このタイプのログインには使用されません。オペレーティング・システムまたはネットワーク・サービスで許可されていれば、それらによってユーザーを認証できます。外部認証を行う場合は、初期化パラメータ OS_AUTHENT_PREFIX を設定し、Oracle ユーザー名にこの接頭辞を含めます。OS_AUTHENT_PREFIX パラメータは、全ユーザーのオペレーティング・システム・アカウント名の先頭に追加する接頭辞を定義します。Oracle は、ユーザーが接続しようとすると、その接頭辞付きのユーザー名をデータベース内の Oracle ユーザー名と比較します。

たとえば、OS_AUTHENT_PREFIX が次のように設定されている場合を想定します。

```
OS_AUTHENT_PREFIX=OPS$
```

注意： OS_AUTHENT_PREFIX 初期化パラメータに指定する文字列は、オペレーティング・システムによっては大 / 小文字が区別されるものがあります。この初期化パラメータの詳細は、オペレーティング・システム固有の Oracle マニュアルを参照してください。

オペレーティング・システム・アカウント名 tsmith を持つユーザーが、Oracle データベースに接続する際にオペレーティング・システムによって認証される場合、Oracle は対応するデータベース・ユーザー OPS$tsmith が存在しているかをチェックし、存在していれば接続を許可します。オペレーティング・システムによって認証されるユーザーへの参照には、OPS$tsmith のように、必ず接頭辞が含まれている必要があります。

このパラメータのデフォルト値は OPS$ であり、これによって古いバージョンの Oracle との下位互換性を維持しています。ただし、接頭辞の値には、他の文字列やヌル文字列（空の二重引用符 "" で指定）も設定できます。ヌル文字列を使用すると、オペレーティング・システム・アカウント名に接頭辞は追加されないので、Oracle ユーザー名とオペレーティング・システム・ユーザー名は完全に一致します。

OS_AUTHENT_PREFIX を設定すると、データベースの存続期間中は同じ設定が維持されます。接頭辞を変更した場合、古い接頭辞を含むデータベース・ユーザー名は、パスワード認証を使用するように変更しないかぎり、接続できません。

外部認証されるユーザーの作成

次の文は、Oracle によって識別され、オペレーティング・システムまたはネットワーク・サービスによって認証されるユーザーを作成します。この例では、OS_AUTHENT_PREFIX = "" の場合を想定しています。

```
CREATE USER scott IDENTIFIED EXTERNALLY;
```

オペレーティング・システムまたはネットワーク・サービスを介した認証を必要とするデータベース・アカウントを作成するには、CREATE USER...IDENTIFIED EXTERNALLY を使用します。Oracle は、特定のオペレーティング・システム・ユーザーが特定のデータベース・ユーザーへのアクセス権を持っていることを保証するために、この外部ログイン認証に依存します。

> **関連項目：** 外部認証の詳細は、『Oracle Advanced Security 管理者ガイド』を参照してください。

オペレーティング・システム認証

デフォルトでは、セキュリティで保護された接続を介し、オペレーティング・システムで認証されたログインのみが許可されます。したがって、オペレーティング・システムによってユーザーを認証する場合、デフォルトでは、そのユーザーは Oracle Net を介してデータベースに接続できません。これは、このユーザーが共有サーバー構成を使用して接続できないことを示します。共有サーバー構成の接続では、Oracle Net が使用されるためです。このデフォルトの制限により、リモート・ユーザーが、ネットワーク接続を介して別のオペレーティング・システムのユーザーになりすますことを防止できます。

リモート・ユーザーがネットワーク接続を介して別のオペレーティング・システムのユーザーになりすます心配がない場合は、データベースの初期化パラメータ・ファイルで REMOTE_OS_AUTHENT を TRUE に設定することにより（デフォルト設定は FALSE）、ネットワーク・クライアントのオペレーティング・システム・ユーザー認証を使用できます。初期化パラメータ REMOTE_OS_AUTHENT を TRUE に設定すると、セキュリティで保護されていない接続を介して受信したクライアントのオペレーティング・システムのユーザー名が RDBMS で受け入れられ、アカウント・アクセスに使用されます。この変更は、次にインスタンスを起動して、データベースをマウントするときに有効となります。

一般に、ホスト・オペレーティング・システムを使用したユーザー認証には、次のような利点があります。

- ユーザーは、個別のデータベース・ユーザー名やパスワードを指定せずに、Oracle に迅速かつ簡便に接続できます。
- データベースの監査証跡とオペレーティング・システムの監査証跡のユーザー・エントリが一致します。

ネットワーク認証

ネットワーク認証は、Oracle Advanced Security を使用して実行されます。Oracle Advanced Security は、Kerberos のようなサード・パーティのサービスを使用するように構成できます。Oracle Advanced Security を唯一の外部認証サービスとして使用する場合、パラメータ `REMOTE_OS_AUTHENT` の設定は関係ありません。これは、Oracle Advanced Security ではセキュリティで保護された接続のみが許可されるためです。

外部認証の利点

次に、外部認証の利点を示します。

- スマート・カード、指紋、Kerberos、オペレーティング・システムなど、使用可能な認証メカニズムの選択肢が増えます。
- Kerberos や DCE など、多数のネットワーク認証サービスがシングル・サインオンをサポートしています。つまり、ユーザーは多数のパスワードを覚えておく必要がありません。
- 前述の外部認証メカニズムをすでに使用している場合は、データベースで同じメカニズムを使用することにより、管理費用を節減できます。

グローバル認証および認可

Oracle Advanced Security を使用すると、認可などのユーザー関連情報を、Lightweight Directory Access Protocol（LDAP）ベースのディレクトリ・サービスで集中管理できます。ユーザーはデータベース内でグローバル・ユーザーとして識別できます。これは、そのユーザーが SSL によって認証され、集中管理されたディレクトリ・サービスによってデータベースの外部でこれらのユーザーが管理されることを意味します。グローバル・ロールはデータベース内で定義され、そのデータベースのみで認識されますが、そのロールに対する認可はディレクトリ・サービスによって行われます。

注意： SSL によって認証され、ディレクトリで認可が管理されないユーザーを作成することもできます。つまり、このユーザーはローカル・データベース・ロールのみを持つことになります。詳細は、『Oracle Advanced Security 管理者ガイド』を参照してください。

このような集中管理により、**エンタープライズ・ユーザー**と**エンタープライズ・ロール**の作成が可能になります。エンタープライズ・ユーザーの定義と管理は、ディレクトリ内で行います。エンタープライズ・ユーザーは企業内で一意の識別情報を持っており、複数データベースにまたがったアクセス権限を決定するエンタープライズ・ロールを割り当てることができます。エンタープライズ・ロールは 1 つ以上のグローバル・ロールから構成されているため、グローバル・ロールのコンテナとみなすことができます。

ディレクトリ・サービスによって認可されるユーザーの作成

ディレクトリ・サービスによって認可されるユーザーを指定するには、次のような方法があります。

グローバル・ユーザーの作成 次の文は、SSL によって認証され、エンタープライズ・ディレクトリ・サービスによって認可されるグローバル・ユーザーの作成方法を示しています。

```
CREATE USER scott
    IDENTIFIED GLOBALLY AS 'CN=scott,OU=division1,O=oracle,C=US';
```

AS 句で指定されている文字列は、エンタープライズ・ディレクトリにとって意味のある識別子（**識別名**、つまり **DN**）です。

この場合、scott は正当なグローバル・ユーザーです。ただし、ここでの欠点は、そのユーザー scott を、アクセスが必要なすべてのデータベース内とディレクトリ内で作成しなければならないことです。

スキーマに依存しないユーザーの作成 スキーマに依存しないユーザーを作成すると、複数のエンタープライズ・ユーザーがデータベース内の共有スキーマにアクセスできます。スキーマに依存しないユーザーは、次のような性質を持ちます。

- SSL またはパスワードによって認証されます。
- どのタイプの CREATE USER 文を使用しても、データベース内には作成されません。
- 権限がディレクトリ内で管理されます。
- 共有スキーマに接続します。

スキーマに依存しないユーザーを作成する手順は、次のとおりです。

1. 次のように、データベース内で共有スキーマを作成します。

   ```
   CREATE USER appschema INDENTIFIED GLOBALLY AS '';
   ```

2. ディレクトリ内で、複数のエンタープライズ・ユーザーとマッピング・オブジェクトを作成します。

   マッピング・オブジェクトは、データベースに対して、ユーザーの DN を共有スキーマにマップする方法を指定します。完全な DN マッピング（一意の DN 1 つに対して 1 つのディレクトリ・エントリが対応する）を作成できる他、たとえば、次の DN コンポーネントを含むすべてのユーザーを appschema にマップすることもできます。

   ```
   OU=division,O=Oracle,C=US
   ```

   これらのマッピングの詳細は、『Oracle Internet Directory 管理者ガイド』を参照してください。

ほとんどのユーザーは専用スキーマを必要としないため、スキーマに依存しないユーザーを実装することで、ユーザーをデータベースから切り離すことができます。データベース内で

同じスキーマを共有する複数のユーザーを作成すると、各ユーザーは他のデータベースの共有スキーマにもエンタープライズ・ユーザーとしてアクセスできます。

グローバル認証とグローバル認可の利点

グローバルなユーザー認証および認可には、次のような利点があります。

- SSL または Windows NT のシステム固有の認証を使用して、厳密認証が行われます。
- ユーザーと権限を全社規模で集中管理できます。
- 管理が容易です。ユーザーごとに、社内の各データベースにスキーマを作成する必要がありません。
- シングル・サインオンを容易に利用できます。ユーザーは一度サインオンすれば、複数のデータベースとサービスにアクセスできます。また、パスワードを使用するユーザーは、パスワード認証方式のエンタープライズ・ユーザーを受け入れるデータベースに、1 つのパスワードを使用してアクセスできます。
- パスワード・ベースのアクセスが提供されるため、以前に定義されたパスワード認証方式のデータベース・ユーザーを、集中管理されるディレクトリに（User Migration Utility を使用して）移行できます。これにより、以前のリリースの Oracle クライアントで使用可能だったグローバル認証と認可が引き続きサポートされます。
- `CURRENT_USER` データベース・リンクはグローバル・ユーザーとして接続します。ローカル・ユーザーは、ストアド・プロシージャのコンテキスト内ではグローバル・ユーザーとして接続できます。グローバル・ユーザーのパスワードをリンク定義に格納する必要はありません。

関連項目： グローバル認証と認可、エンタープライズ・ユーザーおよびエンタープライズ・ロールの詳細は、次のマニュアルを参照してください。

- 『Oracle Advanced Security 管理者ガイド』
- 『Oracle Internet Directory 管理者ガイド』

プロキシ認証および認可

プロキシ・クライアントに対する中間層サーバーを安全な方法で設計できます。

Oracle では、次の 3 つの形式のプロキシ認証が用意されています。

- 中間層サーバーは、データベース・サーバーを使用して中間層サーバー自体を認証します。クライアント（この場合はアプリケーション・ユーザーまたは別のアプリケーション）は、中間層サーバーを使用してクライアント自体を認証します。クライアントの識別情報は、データベースに到達するまで確実に保持されます。
- クライアント（この場合はデータベース・ユーザー）は、中間層サーバーによって認証されません。クライアントの識別情報とデータベース・パスワードが中間層サーバーを経由してデータベース・サーバーに渡され、そこで認証されます。
- クライアント（この場合はグローバル・ユーザー）は中間層サーバーによって認証され、中間層を介して次のいずれかを渡します。クライアントのユーザー名はそこから取得されます。
  - DN
  - 証明書

どの場合でも、中間層サーバーにクライアントの代理としての機能を与えるために、管理者は中間層サーバーを認可する必要があります。

中間層サーバーがクライアントのプロキシとして機能することを認可するには、`ALTER USER` 文の `GRANT CONNECT THROUGH` 句を使用します。また、クライアントとして接続したときに、中間層でアクティブにすることが可能なロールも指定できます。

中間層サーバーがクライアントのかわりに行う操作は、監査できます。

`PROXY_USERS` データ・ディクショナリ・ビューを問い合せると、現在中間層を介した接続が認可されているユーザーを表示できます。

プロキシ接続の認可を取り消すには、`ALTER USER` 文の `REVOKE CONNECT THROUGH` 句を使用します。

関連項目：

- プロキシ・ユーザーに対する中間層サーバーの設計の詳細は、『Oracle Call Interface プログラマーズ・ガイド』および『Oracle9*i* アプリケーション開発者ガイドー基礎編』を参照してください。
- `ALTER USER` の PROXY 句の詳細と構文については、『Oracle9*i* SQL リファレンス』を参照してください。
- ユーザーのかわりに中間層が行う操作の監査の詳細は、26-12 ページの「複数層環境での監査」を参照してください。

ユーザーのプロキシとして機能し、ユーザーを認証する中間層を認可する方法

次の文は、中間層サーバー appserve がユーザー bill として接続することを認可します。WITH ROLE 句の指定により、bill に対応付けられているロールのうち、payroll 以外のすべてのロールが appserve によってアクティブにされます。

```
ALTER USER bill
    GRANT CONNECT THROUGH appserve
    WITH ROLE ALL EXCEPT payroll;
```

中間層サーバー appserve に対して、ユーザー bill として接続するための認可を取り消すには、次の文を使用します。

```
ALTER USER bill REVOKE CONNECT THROUGH appserve;
```

他の方式で認証されたユーザーのプロキシとするために、中間層を認可する方法

中間層がプロキシとして機能するものの、中間層によって認証されないユーザーを認可するには、ALTER USER ... GRANT CONNECT THROUGH 文の AUTHENTICATED USING 句を使用します。現在サポートされている認証方式は、PASSWORD のみです。

次の文は、この認証方式の例を示しています。

```
ALTER USER mary
    GRANT CONNECT THROUGH midtier
    AUTHENTICATED USING PASSWORD;
```

この文では、中間層サーバー midtier が mary として接続することを認可しています。midtier は、認証を行うため、データベース・サーバーに mary のパスワードを渡す必要があります。

DN によって識別されたユーザーのプロキシとするために、中間層を認可する方法

次の文は、中間層サーバー WebDB がデータベース・サーバーにグローバル・ユーザー jeff の DN を渡すことを認可します。DN を使用して、ユーザー名が取得されます。ユーザー jeff は、中間層サーバー WebDB によってすでに認証されています。

```
ALTER USER jeff
     GRANT CONNECT THROUGH WebDB
     AUTHENTICATED USING DISTINGUISHED NAME;
```

必要であれば、中間層サーバーが証明書全体（DN が含まれている）を渡すことを認可できます。次の文は、この操作を示しています。

```
ALTER USER jeff
     GRANT CONNECT THROUGH WebDB
     AUTHENTICATED USING CERTIFICATE;
```

証明書全体を渡すと、認証に時間がかかります。しかし、アプリケーションによっては、証明書に含まれている他の情報を使用するものがあります。

プロファイルによるリソースの管理

プロファイルとは、リソース制限のセットに名前を付けたものです。ユーザーにプロファイルを指定すると、そのプロファイルの定義に従って、データベースの使用とインスタンスのリソースが制限されます。ユーザーごとにプロファイルを割り当てたり、固有のプロファイルを持たないすべてのユーザーに対してデフォルト・プロファイルを割り当てたりすることが可能です。プロファイルを有効にするには、データベース全体に対してリソース制限を使用可能にする必要があります。

ここでは、プロファイル管理について説明します。この項の内容は、次のとおりです。

- リソース制限を使用可能および使用禁止にする方法
- プロファイルの作成
- プロファイルの割当て
- プロファイルの変更
- コンポジット制限の使用
- プロファイルの削除

関連項目： プロファイルの管理に使用する SQL 文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

リソース制限を使用可能および使用禁止にする方法

プロファイルの作成、ユーザーへの割当て、変更および削除は、許可されている任意のデータベース・ユーザーによって随時実行できます。ただし、プロファイルに設定されたリソース制限が適用されるのは、対応するデータベースのリソース制限が使用可能な場合のみです。リソース制限を使用可能または使用禁止にするには、後続の 2 つの項で説明する方法のいずれかを使用します。

データベースがオープンしている状態でリソース制限の適用を変更するには、ALTER SYSTEM システム権限が必要です。

起動前にリソース制限を使用可能および使用禁止にする方法

データベースを一時的に停止できる場合は、データベースの初期化パラメータ・ファイル内の RESOURCE_LIMIT 初期化パラメータで、リソース制限の使用可能 / 使用禁止を設定できます。このパラメータの有効な値は、TRUE（リソース制限を使用可能にする）と FALSE です。デフォルトでは、このパラメータの値は FALSE に設定されます。初期化パラメータ・ファイルを編集した後、その変更を有効にするために、データベース・インスタンスを再起動します。インスタンスが起動されるたびに、新たなパラメータ値によってリソース制限が使用可能または使用禁止になります。

データベースのオープン中にリソース制限を使用可能および使用禁止にする方法

データベースを一時的に停止できない場合や、リソース制限機能を一時的に変更する必要がある場合は、SQL 文 ALTER SYSTEM を使用してリソース制限を使用可能または使用禁止にできます。インスタンスの起動後、ALTER SYSTEM 文を実行すると、RESOURCE_LIMIT 初期化パラメータで設定された値が置き換えられます。たとえば、次の文はデータベースに対するリソース制限を使用可能にします。

```
ALTER SYSTEM
   SET RESOURCE_LIMIT = TRUE;
```

注意： これは、初期化パラメータには適用されません。

ALTER SYSTEM 文は、リソース制限の適用を永続的に決定するわけではありません。データベースを停止または再起動すると、RESOURCE_LIMIT パラメータの設定値によってリソース制限の適用が決定されます。

プロファイルの作成

プロファイルを作成するには、CREATE PROFILE システム権限が必要です。プロファイルは、SQL 文 CREATE PROFILE を使用して作成します。このときに、特定のリソース制限を明示的に設定できます。

次の文は、プロファイル clerk を作成します。

```
CREATE PROFILE clerk LIMIT
    SESSIONS_PER_USER 2
    CPU_PER_SESSION unlimited
    CPU_PER_CALL 6000
    LOGICAL_READS_PER_SESSION unlimited
    LOGICAL_READS_PER_CALL 100
    IDLE_TIME 30
    CONNECT_TIME 480;
```

新しいプロファイルで指定されていないリソース制限はすべて、DEFAULT プロファイルによって制限値が設定されます。

各データベースには DEFAULT というプロファイルがあります。DEFAULT プロファイルの制限は、次の場合に使用されます。

- ユーザーにプロファイルが明示的に割り当てられていない場合、そのユーザーは DEFAULT プロファイル内のすべての制限に従います。
- プロファイルに制限が指定されていない場合は、DEFAULT プロファイルの対応する制限が使用されます。

最初は、DEFAULT プロファイルの制限はすべて UNLIMITED に設定されます。セキュリティ管理者は、DEFAULT プロファイルのユーザーがリソースを無制限に消費しないように、ALTER PROFILE 文を使用してデフォルト制限を変更する必要があります。

```
ALTER PROFILE default LIMIT
   ...;
```

ALTER PROFILE システム権限を持つユーザーであれば、DEFAULT プロファイル内の制限を調整できます。DEFAULT プロファイルは削除できません。

プロファイルの割当て

作成したプロファイルは、データベース・ユーザーに割り当てることができます。各ユーザーには、単一のプロファイルをいつでも割り当てることができます。すでにプロファイルが設定されているユーザーにプロファイルを割り当てると、新しいプロファイルの割当てによって前のプロファイルが置き換えられます。プロファイルの割当ては、現行セッションには影響しません。プロファイルは、ロールや他のプロファイルではなく、ユーザーにのみ割り当てられます。

プロファイルをユーザーに割り当てるには、CREATE USER 文または ALTER USER 文を使用します。

関連項目：

- 24-2 ページ「ユーザーの作成」
- 24-6 ページ「ユーザーの変更」

プロファイルの変更

プロファイルのリソース制限の設定を変更するには、SQL 文 ALTER PROFILE を使用します。プロファイルを変更するには、ALTER PROFILE システム権限が必要です。

プロファイルの制限を調整すると、そのプロファイルの調整前の設定値は上書きされます。DEFAULT の値で制限を調整すると、リソース制限はデータベースのデフォルトの制限値に戻ります。プロファイルの変更時に調整されなかったプロファイルは、すべてそれまでの設定値を維持します。プロファイルに対する変更は、現行セッションには影響しません。新しいプロファイルの設定値は、プロファイルの変更後に作成されたセッションに対してのみ使用されます。

次の文は、プロファイル clerk を変更します。

```
ALTER PROFILE clerk LIMIT
    CPU_PER_CALL default
    LOGICAL_READS_PER_SESSION 20000;
```

コンポジット制限の使用

CREATE PROFILE または ALTER PROFILE を使用して特定のリソースにリソース制限を割り当てることができますが、その他に、コンポジット制限を使用して、セッションのリソース・コストの合計を制限することもできます。コンポジット制限は、リソースの**サービス単位**で計算された加重合計として表されます。

プロファイルのコンポジット制限を設定するには、CREATE PROFILE 文または ALTER PROFILE 文の COMPOSITE_LIMIT 句を使用します。次の CREATE PROFILE 文は、COMPOSITE_LIMIT 句を指定しています。

```
CREATE PROFILE clerk LIMIT
    COMPOSITE_LIMIT 20000
    SESSIONS_PER_USER 2
    CPU_PER_CALL 1000;
```

1 つのプロファイル内には、明示的に指定したリソース制限とコンポジット制限を共存できます。最初にどちらかの制限に到達すると、セッションのアクティビティが停止します。コンポジット制限を使用すると、システム・リソースの使用を柔軟に制限できます。

コンポジット制限の値の決定

適正なコンポジット制限は、平均的なプロファイル・ユーザーによって使用されるリソースの合計によって決まります。明示的に指定されたリソース制限と同じく、標準的なプロファイル・ユーザーが使用するコンポジット・リソースの通常の範囲を決定するために、履歴情報を収集してください。

> **関連項目：** コンポジット制限の計算方法は、『Oracle9*i* SQL リファレンス』を参照してください。

リソース・コストの設定

それぞれの Oracle データベース・サーバー環境には、固有の特性があります。環境によっては、一部のシステム・リソースが非常に貴重であることがあります。Oracle では、次のリソースを重み付けして、リソース・コストの合計に占める度合いを制御できます。

- CPU_PER_SESSION
- LOGICAL_READS_PER_SESSION
- CONNECT_TIME
- PRIVATE_SGA

リソースを重み付けしない場合は、デフォルトで重みが 0（ゼロ）に設定され、そのリソースを使用してもリソース・コスト合計には加算されません。

Oracle は、リソース・コストの合計を計算する際、まずセッションで使用されている各リソースの量に重みを掛けてから、4 つのリソースすべての積を合計します。どのようなセッションでも、このコストは、ユーザー・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。積と合計コストは、どちらもサービス単位と呼ばれる単位で表されます。

リソースを重み付けするには、ALTER RESOURCE COST 文を使用します。そのためには、ALTER RESOURCE システム権限が必要です。次の例では、CPU_PER_SESSION および LOGICAL_READS_PER_SESSION リソースを重み付けしています。

```
ALTER RESOURCE COST
    CPU_PER_SESSION 1
    LOGICAL_READS_PER_SESSION 50;
```

この重みにより、セッションのコストは次の計算式で算出されます。

```
cost = (1 * CPU_PER_SESSION) + (50 * LOGICAL_READS_PER_SESSION)
```

この式で使用される CPU_PER_SESSION と LOGICAL_READS_PER_SESSION の値は、DEFAULT プロファイル内の値か、そのセッションを所有するユーザーのプロファイル内の値のいずれかです。

上の文では、リソース CONNECT_TIME と PRIVATE_SGA に重み付けしていないため、計算式にこれらのリソースは含まれていません。

関連項目：

リソース・コストの設定の詳細と推奨事項は、次のマニュアルを参照してください。

- 『Oracle9*i* SQL リファレンス』
- 使用しているオペレーティング・システム固有の Oracle マニュアル

プロファイルの削除

プロファイルを削除するには、DROP PROFILE システム権限が必要です。プロファイルを削除するには、SQL 文 DROP PROFILE を使用します。現在ユーザーに割り当てられているプロファイルを正常に削除するには、CASCADE オプションを使用します。

次の文は、プロファイル clerk を削除します。このプロファイルがユーザーに割り当てられていても削除されます。

```
DROP PROFILE clerk CASCADE;
```

削除するプロファイルに現在割り当てられているユーザーは、自動的に DEFAULT プロファイルに割り当てられます。DEFAULT プロファイルは削除できません。プロファイルを削除しても、現在アクティブなセッションには影響しません。プロファイルの削除後に作成されたセッションのみが、変更されたプロファイル割当てに従います。

データベース・ユーザーとプロファイルに関する情報の表示

次のデータ・ディクショナリ・ビューには、データベース・ユーザーとプロファイルに関する情報が含まれています。

ビュー	説明
DBA_USERS ALL_USERS USER_USERS	DBA ビューには、データベース内のすべてのユーザーが表示されます。ALL ビューには、現行ユーザーに対して表示可能なユーザーがリストされますが、それらの記述は表示されません。USER ビューには、現行ユーザーのみが表示されます。
DBA_TS_QUOTAS USER_TS_QUOTAS	ユーザーの表領域割当て制限が表示されます。
USER_PASSWORD_LIMITS	ユーザーに割り当てられているパスワード・プロファイル・パラメータが表示されます。
USER_RESOURCE_LIMITS	現行ユーザーのリソース制限が表示されます。
DBA_PROFILES	すべてのプロファイルとそれぞれの制限が表示されます。
RESOURCE_COST	各リソースのコストがリストされます。
V$SESSION	現行セッションごとにセッション情報が表示されます。この情報には、ユーザー名が含まれます。
V$SESSTAT	ユーザー・セッションの統計がリストされます。
V$STATNAME	V$SESSTAT ビューに表示される統計について、デコードされた統計名が表示されます。
PROXY_USERS	他のユーザーの識別情報を引き継ぐことができるユーザーが表示されます。

次の項では、これらのビューの使用例をいくつか示します。それらの使用例では、次の文が実行されたデータベースを使用していることを想定しています。

```
CREATE PROFILE clerk LIMIT
    SESSIONS_PER_USER 1
    IDLE_TIME 30
    CONNECT_TIME 600;

CREATE USER jfee
    IDENTIFIED BY wildcat
    DEFAULT TABLESPACE users
    TEMPORARY TABLESPACE temp_ts
    QUOTA 500K ON users
    PROFILE clerk;
```

```
CREATE USER dcranney
    IDENTIFIED BY bedrock
    DEFAULT TABLESPACE users
    TEMPORARY TABLESPACE temp_ts
    QUOTA unlimited ON users;

CREATE USER userscott
     IDENTIFIED BY scott1;
```

関連項目： データ・ディクショナリ・ビューと動的パフォーマンス・ビューの詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

すべてのユーザーとその関連情報のリスト

次の問合せは、データベースに定義されているユーザーとユーザーに関連する情報をリストします。

```
SELECT USERNAME, PROFILE, ACCOUNT_STATUS FROM DBA_USERS;

USERNAME        PROFILE         ACCOUNT_STATUS
--------------- --------------- ---------------
SYS             DEFAULT         OPEN
SYSTEM          DEFAULT         OPEN
USERSCOTT       DEFAULT         OPEN
JFEE            CLERK           OPEN
DCRANNEY        DEFAULT         OPEN
```

パスワードはすべてセキュリティ保護のために暗号化されています。ユーザーが PASSWORD 列を問い合せても、そのユーザーは他のユーザーのパスワードを判断できません。

すべての表領域割当て制限のリスト

次の問合せは、各ユーザーに明示的に割り当てられている表領域割当て制限をすべてリストします。

```
SELECT * FROM DBA_TS_QUOTAS;

TABLESPACE    USERNAME    BYTES     MAX_BYTES    BLOCKS    MAX_BLOCKS
----------    ---------  --------   ----------   -------   ----------
USERS         JFEE              0       512000         0          250
USERS         DCRANNEY          0           -1         0           -1
```

固有の割当て制限が割り当てられている場合は、正確な数値が MAX_BYTES 列に示されます。この数値は常にデータベース・ブロック・サイズの倍数となるため、倍数でない表領域割当て制限を指定すると、適切な値に切り上げられます。無制限割当ての場合は、-1 が表示されます。

すべてのプロファイルと割り当てられている制限のリスト

次の問合せは、データベース内のすべてのプロファイルと各プロファイル内の各制限に対応する設定をリストします。

```
SELECT * FROM DBA_PROFILES
   ORDER BY PROFILE;

PROFILE              RESOURCE_NAME              RESOURCE   LIMIT
-----------------    ---------------            ---------- --------------
CLERK                COMPOSITE_LIMIT            KERNEL     DEFAULT
CLERK                FAILED_LOGIN_ATTEMPTS      PASSWORD   DEFAULT
CLERK                PASSWORD_LIFE_TIME         PASSWORD   DEFAULT
CLERK                PASSWORD_REUSE_TIME        PASSWORD   DEFAULT
CLERK                PASSWORD_REUSE_MAX         PASSWORD   DEFAULT
CLERK                PASSWORD_VERIFY_FUNCTION   PASSWORD   DEFAULT
CLERK                PASSWORD_LOCK_TIME         PASSWORD   DEFAULT
CLERK                PASSWORD_GRACE_TIME        PASSWORD   DEFAULT
CLERK                PRIVATE_SGA                KERNEL     DEFAULT
CLERK                CONNECT_TIME               KERNEL     600
CLERK                IDLE_TIME                  KERNEL     30
CLERK                LOGICAL_READS_PER_CALL     KERNEL     DEFAULT
CLERK                LOGICAL_READS_PER_SESSION  KERNEL     DEFAULT
CLERK                CPU_PER_CALL               KERNEL     DEFAULT
CLERK                CPU_PER_SESSION            KERNEL     DEFAULT
CLERK                SESSIONS_PER_USER          KERNEL     1
DEFAULT              COMPOSITE_LIMIT            KERNEL     UNLIMITED
DEFAULT              PRIVATE_SGA                KERNEL     UNLIMITED
DEFAULT              SESSIONS_PER_USER          KERNEL     UNLIMITED
DEFAULT              CPU_PER_CALL               KERNEL     UNLIMITED
DEFAULT              LOGICAL_READS_PER_CALL     KERNEL     UNLIMITED
DEFAULT              CONNECT_TIME               KERNEL     UNLIMITED
DEFAULT              IDLE_TIME                  KERNEL     UNLIMITED
DEFAULT              LOGICAL_READS_PER_SESSION  KERNEL     UNLIMITED
DEFAULT              CPU_PER_SESSION            KERNEL     UNLIMITED
DEFAULT              FAILED_LOGIN_ATTEMPTS      PASSWORD   UNLIMITED
DEFAULT              PASSWORD_LIFE_TIME         PASSWORD   UNLIMITED
DEFAULT              PASSWORD_REUSE_MAX         PASSWORD   UNLIMITED
DEFAULT              PASSWORD_LOCK_TIME         PASSWORD   UNLIMITED
DEFAULT              PASSWORD_GRACE_TIME        PASSWORD   UNLIMITED
DEFAULT              PASSWORD_VERIFY_FUNCTION   PASSWORD   UNLIMITED
DEFAULT              PASSWORD_REUSE_TIME        PASSWORD   UNLIMITED
32 rows selected.
```

各ユーザー・セッションのメモリー使用の表示

次の問合せは、現行セッションをすべてリストし、各セッションの Oracle ユーザーとユーザー・グローバル領域（UGA）の現在のメモリー使用を示します。

```
SELECT USERNAME, VALUE || 'bytes' "Current UGA memory"
   FROM V$SESSION sess, V$SESSTAT stat, V$STATNAME name
WHERE sess.SID = stat.SID
   AND stat.STATISTIC# = name.STATISTIC#
   AND name.NAME = 'session uga memory';

USERNAME                       Current UGA memory
------------------------------ --------------------------------------------
                               18636bytes
                               17464bytes
                               19180bytes
                               18364bytes
                               39384bytes
                               35292bytes
                               17696bytes
                               15868bytes
USERSCOTT                      42244bytes
SYS                            98196bytes
SYSTEM                         30648bytes

11 rows selected.
```

インスタンス起動以降、各セッションに割り当てられた最大の UGA メモリーを表示するには、この問合せの `'session uga memory'` を `'session uga memory max'` に置き換えてください。

25

ユーザー権限とロールの管理

この章では、権限とロールを使用してスキーマ・オブジェクトへのアクセスを制御する方法と、システム操作の実行許可を制御する方法について説明します。この章の内容は、次のとおりです。

- ユーザー権限とロールの理解
- ユーザー・ロールの管理
- ユーザー権限とロールの付与
- ユーザー権限とロールの取消し
- ユーザー・グループ PUBLIC に対する付与と取消し
- 権限の付与と取消しが有効になるとき
- オペレーティング・システムまたはネットワークを使用したロールの付与
- 権限とロールに関する情報の表示

関連項目：

- データベースへのアクセス制御の詳細は、第 24 章「ユーザーとリソースの管理」を参照してください。
- 提案されている一般的なデータベース・セキュリティ・ポリシーの詳細は、第 23 章「セキュリティ・ポリシーの設定」を参照してください。

ユーザー権限とロールの理解

ユーザー**権限**とは、特定のタイプの SQL 文を実行するための権利、または他のユーザーのオブジェクトにアクセスするための権利です。権限のタイプは、Oracle により定義されます。

これに対して、**ロール**は、ユーザー（通常は管理者）が権限や他のロールをグループ化するために作成して使用します。ロールを使用すると、複数の権限またはロールをユーザーに容易に付与できます。

ここでは、Oracle のユーザー権限について説明します。この項の内容は、次のとおりです。

- システム権限
- オブジェクト権限
- ユーザー・ロール

関連項目： 権限とロールの詳細は、『Oracle9*i* データベース概要』を参照してください。

システム権限

システム権限は 100 種類以上あります。各システム権限を使用すると、ユーザーは特定のデータベース操作またはあるクラスのデータベース操作を実行できます。

注意： システム権限は非常に強力なので、この権限をデータベースのロールと信頼のあるユーザーに付与するのは必要な場合のみに限定してください。

関連項目： システム権限の全リストと詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

システム権限の制限

システム権限は非常に強力なので、通常のユーザー（DBA 以外のユーザー）がデータ・ディクショナリに対して ANY システム権限（UPDATE ANY TABLE など）を実行できないようにデータベースを構成することをお薦めします。データ・ディクショナリを保護するために、O7_DICTIONARY_ACCESSIBILITY 初期化パラメータを FALSE に設定してください。この機能を、ディクショナリ保護メカニズムと呼びます。

注意： O7_DICTIONARY_ACCESSIBILITY 初期化パラメータは、Oracle7 から Oracle8*i* 以上のリリースにアップグレードした場合の、システム権限に対する制限を制御します。このパラメータを TRUE に設定すると、SYS スキーマ内のオブジェクトへのアクセスが可能になります（Oracle7 の動作）。このパラメータを FALSE に設定すると、「任意のスキーマ」内にあるオブジェクトへのアクセスを許可するシステム権限では、SYS スキーマにあるオブジェクトへのアクセスは許可されません。O7_DICTIONARY_ACCESSIBILITY のデフォルトは FALSE です。

この初期化パラメータを FALSE に設定しない場合は、データ・ディクショナリに ANY 権限が適用されるため、ANY 権限を持つ悪意あるユーザーがデータ・ディクショナリ表にアクセスし、変更する危険性があります。

O7_DICTIONARY_ACCESSIBILITY 初期化パラメータの詳細は『Oracle9*i* データベース・リファレンス』、使用方法は『Oracle9*i* データベース移行ガイド』を参照してください。

ディクショナリ保護を有効（O7_DICTIONARY_ACCESSIBILITY = FALSE）にすると、SYS スキーマのオブジェクト（ディクショナリ・オブジェクト）へのアクセスが、SYS スキーマを持つユーザーのみに限定されます。これらのユーザーとは、SYS と、SYSDBA として接続するユーザーのことです。前述のユーザー以外は、他のスキーマのオブジェクトへのアクセスを提供するシステム権限があっても、SYS スキーマのオブジェクトにはアクセスできません。たとえば、SELECT ANY TABLE 権限を持つユーザーは、他のスキーマのビューや表にはアクセスできますが、ディクショナリ・オブジェクト（動的パフォーマンス・ビューのベース表、ビュー、パッケージおよびシノニム）は選択できません。ただし、これらのユーザーに明示的なオブジェクト権限を付与することで、SYS スキーマ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

明示的なオブジェクト権限のあるユーザーまたは管理権限のあるユーザー（SYSDBA）は、SYS スキーマ内のオブジェクトにアクセスできます。その他にも、ユーザーに次のロールを付与することによって、SYS スキーマ内のオブジェクトへのアクセスを許可できます。

- SELECT_CATALOG_ROLE

  このロールを付与されたユーザーには、すべてのデータ・ディクショナリ・ビューの SELECT 権限が与えられます。

- EXECUTE_CATALOG_ROLE

  このロールを付与されたユーザーには、データ・ディクショナリ内にあるパッケージとプロシージャの EXECUTE 権限が与えられます。

- DELETE_CATALOG_ROLE

  このロールを付与されたユーザーは、システム監査表（AUD$）からレコードを削除できます。

また、SYS スキーマ内に作成された表へのアクセスが必要なユーザーには、次のシステム権限を付与できます。

- SELECT ANY DICTIONARY

  このシステム権限を使用すると、SYS スキーマ内に作成された、表などの任意のオブジェクトに対して問合せを実行できます。このシステム権限は、権限を必要とする各ユーザーに個別に付与する必要があります。GRANT ALL PRIVILEGES には含まれておらず、ロールを介して付与することはできません。

注意： これらのロールおよび SELECT ANY DICTIONARY システム権限は、悪用することによってシステムの整合性が損なわれる危険があるため、付与する際には十分な注意が必要です。

オブジェクト権限

オブジェクトの各タイプには、異なるオブジェクト権限が対応付けられます。

ALL [PRIVILEGES] を指定すると、オブジェクトに対して使用可能なすべてのオブジェクト権限を付与するか、取り消すことができます。ALL は権限ではなくショートカット、つまり、GRANT 文および REVOKE 文において 1 語ですべてのオブジェクト権限を付与または取り消すための手段です。ALL を使用してすべてのオブジェクト権限を付与した場合でも、個別に権限を取り消すことができます。

同様に、ALL を指定して、個別に付与した権限をすべて取り消すこともできます。ただし、REVOKE ALL によって整合性制約が削除される場合（整合性制約は取り消そうとしている REFERENCES 権限に依存しているため）は、REVOKE 文に CASCADE CONSTRAINTS オプションを指定する必要があります。

関連項目： すべてのオブジェクト権限のリストは、『Oracle9*i* SQL リファレンス』を参照してください。

ユーザー・ロール

ロールは複数の権限やロールを 1 つにまとめるので、ユーザーに対して同時に権限を付与したり、取り消したりできます。ユーザーがロールを使用するには、そのユーザーに対してロールを使用可能にする必要があります。

Oracle には、データベース管理を容易にするために事前定義済みロールがいくつか用意されています。表 25-1 に示すこれらのロールは、データベース作成の一部である標準スクリプトの実行時に、Oracle データベースに対して自動的に定義されます。ユーザー定義のロールと同様に、これらの事前定義済みロールにも、権限とロールを付与したり、取り消したりできます。

表 25-1 事前定義済みロール

ロール名	作成するスクリプト	説明
CONNECT	SQL.BSQ	次のシステム権限が含まれます。ALTER SESSION、CREATE CLUSTER、CREATE DATABASE LINK、CREATE SEQUENCE、CREATE SESSION、CREATE SYNONYM、CREATE TABLE、CREATE VIEW
RESOURCE	SQL.BSQ	次のシステム権限が含まれます。CREATE CLUSTER、CREATE INDEXTYPE、CREATE OPERATOR、CREATE PROCEDURE、CREATE SEQUENCE、CREATE TABLE、CREATE TRIGGER、CREATE TYPE
DBA	SQL.BSQ	ADMIN OPTION 付きのすべてのシステム権限
注意：ここまでの 3 つのロールは、Oracle の旧リリースとの互換性を保つためのものであり、将来のリリースでは自動的に作成されない場合があります。これらのロールに依存するのではなく、データベース・セキュリティ用に独自のロールを設計することをお薦めします。		
EXP_FULL_DATABASE	CATEXP.SQL	全データベース・エクスポートと増分データベース・エクスポートを実行するための権限を付与します。含まれる権限は、SELECT ANY TABLE、BACKUP ANY TABLE、EXECUTE ANY PROCEDURE、EXECUTE ANY TYPE、ADMINISTER RESOURCE MANAGER と、表 SYS.INCVID、SYS.INCFIL および SYS.INCEXP に対する INSERT、DELETE および UPDATE です。また、ロール EXECUTE_CATALOG_ROLE および SELECT_CATALOG_ROLE も含まれます。

表 25-1 事前定義済みロール（続き）

ロール名	作成するスクリプト	説明
IMP_FULL_DATABASE	CATEXP.SQL	全データベース・インポートを実行するための権限を付与します。システム権限の詳細リスト（権限を表示するにはビュー DBA_SYS_PRIVS を使用する）と、ロール EXECUTE_CATALOG_ROLE および SELECT_CATALOG_ROLE が含まれます。
DELETE_CATALOG_ROLE	SQL.BSQ	システム監査表（AUD$）に対する DELETE 権限を付与します。
EXECUTE_CATALOG_ROLE	SQL.BSQ	データ・ディクショナリ内のオブジェクトに対する EXECUTE 権限を付与します。HS_ADMIN_ROLE も含まれます。
SELECT_CATALOG_ROLE	SQL.BSQ	データ・ディクショナリ内のオブジェクトに対する SELECT 権限を付与します。HS_ADMIN_ROLE も含まれます。
RECOVERY_CATALOG_OWNER	CATALOG.SQL	リカバリ・カタログの所有者用の権限を付与します。含まれる権限は、CREATE SESSION、ALTER SESSION、CREATE SYNONYM、CREATE VIEW、CREATE DATABASE LINK、CREATE TABLE、CREATE CLUSTER、CREATE SEQUENCE、CREATE TRIGGER および CREATE PROCEDURE です。
HS_ADMIN_ROLE	CATHS.SQL	異機種間サービス（HS）データ・ディクショナリ表（SELECT を付与）およびパッケージ（EXECUTE を付与）へのアクセスを保護するために使用します。一般的なデータ・ディクショナリへのアクセス権を持つユーザーでも HS データ・ディクショナリにアクセスできるように、SELECT_CATALOG_ROLE および EXECUTE_CATALOG_ROLE にこのロールが付与されています。
AQ_USER_ROLE	CATQUEUE.SQL	使用されなくなっていますが、主にリリース 8.0 との互換性のために残されています。DBMS_AQ および DBMS_AQIN の実行権限を付与します。
AQ_ADMINISTRATOR_ROLE	CATQUEUE.SQL	アドバンスト・キューイングを管理するための権限を付与します。ENQUEUE ANY QUEUE、DEQUEUE ANY QUEUE、MANAGE ANY QUEUE、AQ 表に対する SELECT 権限、および AQ パッケージに対する EXECUTE 権限が含まれます。

表 25-1 事前定義済みロール（続き）

ロール名	作成するスクリプト	説明
SNMPAGENT	CATSNMP.SQL	Enterprise Manager/Intelligent Agent によって使用されます。ANALYZE ANY 権限が含まれており、各種ビューに対する SELECT 権限を付与します。

他のオプションや製品をインストールした場合は、他の事前定義済みロールが作成されることがあります。

ユーザー・ロールの管理

ここでは、ロールの管理について説明します。この項の内容は、次のとおりです。

- ロールの作成
- ロール認可のタイプの指定
- ロールの削除

ロールの作成

ロールは CREATE ROLE 文を使用して作成できますが、そのためには CREATE ROLE システム権限が必要です。通常は、セキュリティ管理者のみがこのシステム権限を持っています。

注意： 作成した直後のロールには、権限は 1 つも対応付けられていません。新しいロールに権限を対応付けるには、新しいロールに権限や他のロールを付与する必要があります。

作成する各ロールには、データベースの既存のユーザー名やロール名とは異なる、一意の名前を与えることが必要です。ロールはどのユーザーのスキーマ内にも格納されません。マルチバイト・キャラクタ・セットを使用するデータベースでは、各ロール名に少なくとも 1 つのシングルバイト・キャラクタを含めることをお薦めします。ロール名がマルチバイト・キャラクタしか含まない場合、暗号化されたロール名およびパスワードの組合せの安全性が大幅に損なわれます。

次の文は、clerk ロールを作成します。clerk ロールは、パスワード bicentennial を使用して、データベースによって認可されます。

```
CREATE ROLE clerk IDENTIFIED BY bicentennial;
```

IDENTIFIED BY 句は、このロールを付与される特定のユーザーがロールを使用する前に、どの認可方式で認可される必要があるかを指定します。この句を指定しないか、または NOT

IDENTIFIED を指定すると、認可がなくてもロールが使用可能になります。ロールには、必要な認可方式として次の方式を指定できます。

- パスワードを使用したデータベースによる認可
- 指定のパッケージを使用したアプリケーションによる認可
- オペレーティング・システム、ネットワークまたはその他の外部ソースによる外部認可
- エンタープライズ・ディレクトリ・サービスによるグローバル認可

これらの認可については後述します。

後でロールの認可方式を設定または変更するには、ALTER ROLE 文を使用します。次の文は clerk ロールを変更し、ユーザーが外部ソースによって認可されている場合のみ、ロールを使用可能にすることを指定します。

```
ALTER ROLE clerk IDENTIFIED EXTERNALLY;
```

ロールの認可方式を変更するには、ALTER ANY ROLE システム権限を持っているか、または ADMIN OPTION 付きのロールが付与されている必要があります。

> **関連項目：** ロールと権限の管理に使用する SQL 文の構文、制限事項および認可情報は、『Oracle9*i* SQL リファレンス』を参照してください。

ロール認可のタイプの指定

ここでは、ロールの認可方式について説明します。ロールを使用するには、ユーザーに対してロールを使用可能にする必要があります。

> **関連項目：** ロールを使用可能にする方法の詳細は、25-21 ページの「権限の付与と取消しが有効になるとき」を参照してください。

データベースによるロールの認可

データベースによって認可されるロールの使用は、そのロールに対応付けられているパスワードによって保護されます。パスワード保護付きのロールが付与されている場合は、そのロールの適切なパスワードを SET ROLE 文で指定することにより、ロールを使用可能または使用禁止にすることができます。ただし、ロールがデフォルト・ロールであり、接続時に使用可能になる場合は、パスワードを入力する必要はありません。

次の文は、ロール manager を作成します。このロールを使用可能にするには、パスワード morework を入力する必要があります。

```
CREATE ROLE manager IDENTIFIED BY morework;
```

注意： マルチバイト・キャラクタ・セットを使用するデータベースの場合でも、ロールのパスワードにはシングルバイト・キャラクタのみを使用してください。マルチバイト・キャラクタのパスワードは受け入れられません。有効なパスワードの詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

アプリケーションによるロールの認可

INDENTIFIED USING *package_name* 句を使用すると、アプリケーション・ロールを作成できます。アプリケーション・ロールとは、認可パッケージを使用したアプリケーションによってのみ使用可能にできるロールのことです。アプリケーション開発者は、アプリケーション内部にパスワードを埋め込むことによってロールを保護する必要はありません。かわりに、アプリケーション・ロールを作成して、ロールを使用可能にすることを認可する PL/SQL パッケージを指定できます。

次の例は、ロール admin_role がアプリケーション・ロールであり、PL/SQL パッケージの hr.admin 内に定義されているモジュールによってのみ使用可能にできることを示しています。

```
CREATE ROLE admin_role IDENTIFIED USING hr.admin;
```

ログイン時に、ユーザーのプロファイルで指定されたデフォルト・ロールが使用可能になる場合、アプリケーション・ロールはチェックされません。

外部ソースによるロールの認可

次の文は accts_rec ロールを作成し、ユーザーが外部ソースによって認可されている場合のみ、ロールを使用可能にすることを指定します。

```
CREATE ROLE accts_rec IDENTIFIED EXTERNALLY;
```

オペレーティング・システムによるロールの認可 オペレーティング・システムによるロール認証は、オペレーティング・システムがオペレーティング・システムの権限をアプリケーションと動的にリンク可能なときのみ有効です。ユーザーがアプリケーションを開始すると、オペレーティング・システムはオペレーティング・システム権限をそのユーザーに付与します。付与されたオペレーティング・システム権限は、アプリケーションに対応付けられたロールと一致します。この時点で、アプリケーションはアプリケーション・ロールを使用可能にできます。アプリケーションが終了すると、先に付与されたオペレーティング・システム権限は、そのユーザーのオペレーティング・システム・アカウントから取り消されます。

ロールをオペレーティング・システムによって認可する場合は、オペレーティング・システム・レベルで各ユーザーの情報を構成する必要があります。この操作は、オペレーティング・システムによって異なります。

ロールがオペレーティング・システムによって付与されている場合は、オペレーティング・システムによってロールを認可する必要はありません。それは無駄な操作になります。

> **関連項目：** オペレーティング・システムによって付与されるロールの詳細は、25-22 ページの「オペレーティング・システムまたはネットワークを使用したロールの付与」を参照してください。

ロールの認可とネットワーク・クライアント ユーザーが Oracle Net を介してデータベースに接続する場合、デフォルトでは、ユーザーのロールはオペレーティング・システムによって認証できません。これには、共有サーバー構成による接続が含まれます。共有サーバー構成での接続は Oracle Net を必要とするためです。リモート・ユーザーはネットワーク接続を介して別のオペレーティング・システム・ユーザーになりすますおそれがあるため、デフォルトでこのような制限が課されています。

このようなセキュリティの危険性の心配がない場合は、データベースの初期化パラメータ・ファイル内にある初期化パラメータ `REMOTE_OS_ROLES` を `TRUE` に設定することにより、ネットワーク・クライアントでオペレーティング・システムによるロール認証を使用できます。この変更は、次にインスタンスを起動し、データベースをマウントするときに有効となります。このパラメータのデフォルト値は `FALSE` です。

エンタープライズ・ディレクトリ・サービスによるロールの認可

ロールをグローバル・ロールとして定義すると、そのロールを使用するために（グローバル）ユーザーはエンタープライズ・ディレクトリ・サービスによる認可を受ける必要があります。グローバル・ロールは、権限とロールを付与することによってデータベース内でローカルに定義しますが、グローバル・ロール自体をそのデータベース内のユーザーや他のロールに付与できません。グローバル・ユーザーがデータベースへの接続を試みると、エンタープライズ・ディレクトリへの問合せが実行され、そのユーザーに対応付けられたグローバル・ロールが取得されます。

次の文は、グローバル・ロールを作成します。

```
CREATE ROLE supervisor IDENTIFIED GLOBALLY;
```

グローバル・ロールは、エンタープライズ・ユーザー管理の構成要素の 1 つです。グローバル・ロールは 1 つのデータベースにのみ適用されますが、エンタープライズ・ディレクトリに定義されたエンタープライズ・ロールに付与できます。エンタープライズ・ロールは複数データベースのグローバル・ロールを含んだディレクトリ構造であり、エンタープライズ・ユーザーに付与できます。

ユーザーのグローバル認証および認可と、エンタープライズ・ユーザー管理におけるロールの概要については、24-13 ページの「グローバル認証および認可」を参照してください。

> **関連項目：** エンタープライズ・ユーザー管理とその実装方法の詳細は、『Oracle Advanced Security 管理者ガイド』および『Oracle Internet Directory 管理者ガイド』を参照してください。

ロールの削除

状況によっては、データベースからロールを削除した方がよい場合があります。削除したロールを付与されていたすべてのユーザーとロールのセキュリティ・ドメインは、削除したロールの権限がなくなったことを反映するために、ただちに変更されます。削除したロールによって間接的に付与されていたロールも、関連するセキュリティ・ドメインから削除されます。ロールを削除することによって、すべてのユーザーのデフォルト・ロール・リストからそのロールが自動的に削除されます。

オブジェクトの作成はロールを介して受け取った権限に依存しないため、ロールが削除されても、表や他のオブジェクトは削除されません。

ロールは、SQL 文 DROP ROLE を使用して削除します。ロールを削除するには、DROP ANY ROLE システム権限を持っているか、または ADMIN OPTION 付きのロールが付与されている必要があります。

次の文は、ロール clerk を削除します。

```
DROP ROLE clerk;
```

ユーザー権限とロールの付与

ここでは、権限とロールの付与について説明します。この項の内容は、次のとおりです。

- システム権限とロールの付与
- オブジェクト権限の付与
- 列に対する権限の付与

中間層またはプロキシを介して接続したユーザーにロールを付与することもできます。この操作については、24-16 ページの「プロキシ認証および認可」を参照してください。

システム権限とロールの付与

システム権限とロールを他のユーザーやロールに付与するには、GRANT 文を使用します。次の権限が必要です。

- システム権限を付与するには、ADMIN OPTION 付きのシステム権限を付与されているか、GRANT ANY PRIVILEGE システム権限を付与されている必要があります。
- ロールを付与するには、ADMIN OPTION 付きのロールを付与されているか、GRANT ANY ROLE システム権限を付与されている必要があります。

> **注意：** IDENTIFIED GLOBALLY を指定して作成されたロールは、他のロールやユーザーに付与できません。グローバル・ロールの付与（および取消し）は、エンタープライズ・ディレクトリ・サービスでのみ制御されます。

次の文は、システム権限 CREATE SESSION および accts_pay ロールをユーザー jward に付与します。

```
GRANT CREATE SESSION, accts_pay TO jward;
```

注意： オブジェクト権限は、同じ GRANT 文でシステム権限およびロールと同時に付与することはできません。

ADMIN OPTION の付与

WITH ADMIN OPTION 句を指定して権限またはロールを付与されたユーザーまたはロールは、次のように複数の拡張機能を使用できます。

- 付与されたユーザーは、データベース内の任意のユーザーまたは他のロールに対して、システム権限またはロールを付与または取消しができます。ただし、自分自身からロールを取り消すことはできません。
- 付与されたユーザーは、ADMIN OPTION 付きのシステム権限やロールを付与できます。
- ロールを付与されたユーザーは、そのロールを変更または削除できます。

次の文では、セキュリティ管理者が michael に new_dba ロールを付与しています。

```
GRANT new_dba TO michael WITH ADMIN OPTION;
```

ユーザー michael は、new_dba ロール内の権限をすべて暗黙的に使用できるだけでなく、必要に応じて new_dba ロールを付与、取消しまたは削除できます。このように ADMIN OPTION は非常に強力な機能なので、ADMIN OPTION 付きのシステム権限やロールを付与する際は、十分に注意してください。通常、このような権限はセキュリティ管理者用に用意されているため、システム内の他の管理者やユーザーに付与することはほとんどありません。

ユーザーがロールを作成すると、そのロールは自動的に ADMIN OPTION 付きでその作成ユーザーに付与されます。

GRANT 文を使用した新規ユーザーの作成

Oracle では、GRANT 文を使用して新しいユーザーを作成できます。IDENTIFIED BY 句を使用してパスワードを指定する場合に、ユーザー名またはパスワードがデータベースに存在しないと、そのユーザー名とパスワードを使用して新しいユーザーが作成されます。次の例では、新規ユーザーとして ssmith を作成し、CONNECT システム権限を付与しています。

```
GRANT CONNECT TO ssmith IDENTIFIED BY p1q2r3;
```

関連項目： 24-2 ページ「ユーザーの作成」

オブジェクト権限の付与

GRANT 文を使用して、ロールとユーザーにオブジェクト権限を付与することもできます。オブジェクト権限を付与するには、次のどちらかの条件を満たしている必要があります。

- 指定するオブジェクトを所有している。
- オブジェクト所有者のかわりに権限を付与したり取り消すことができる GRANT ANY OBJECT PRIVILEGE システム権限を付与されている。
- 所有者からオブジェクト権限を付与されるときに、WITH GRANT OPTION 句が指定されている。

注意： 同じ GRANT 文で、オブジェクト権限とともにシステム権限とロールを付与することはできません。

次の文は、emp 表のすべての列に対する SELECT、INSERT および DELETE のオブジェクト権限をユーザー jfee と tsmith に付与します。

```
GRANT SELECT, INSERT, DELETE ON emp TO jfee, tsmith;
```

salary ビューのすべてのオブジェクト権限をユーザー jfee に付与するには、次の例のように ALL キーワードを使用します。

```
GRANT ALL ON salary TO jfee;
```

GRANT OPTION の指定

WITH GRANT OPTION 句を指定すると、付与されたユーザーはオブジェクト権限を他のユーザーやロールに付与できます。スキーマ内にオブジェクトを格納しているユーザーには、GRANT OPTION 付きの関連するオブジェクト権限がすべて自動的に付与されます。この特別な権限によって、付与されたユーザーの権限は次のように拡張されます。

- 付与されたユーザーは、データベース内の任意のユーザーに対して、GRANT OPTION 付きまたは GRANT OPTION なしでオブジェクト権限を付与できます。また、データベース内の任意のロールに対してもオブジェクト権限を付与できます。
- 次の条件がどちらも成り立つ場合、権限を付与されたユーザーは表に対するビューを作成し、データベース内の任意のユーザーまたはロールに対して、ビューに関する対応する権限を付与できます。
  - 付与されたユーザーが、表に対する GRANT OPTION 付きのオブジェクト権限を受領している。
  - 付与されたユーザーが、CREATE VIEW または CREATE ANY VIEW システム権限を持っている。

オブジェクト権限をロールに付与する場合、GRANT OPTION は無効です。Oracle はロールによるオブジェクト権限を他に与えることを禁止しているため、ロールを付与されたユーザーがロールを介して与えられたオブジェクト権限を他に与えることはできません。

オブジェクト所有者にかわるオブジェクト権限の付与

GRANT ANY OBJECT PRIVILEGE システム権限を持つユーザーは、オブジェクト所有者のかわりに、すべてのオブジェクト権限を付与したり取り消すことができます。これにより、データベース管理者やアプリケーション管理者は、スキーマに接続せずに、スキーマ内のオブジェクトへのアクセス権を付与できます。また、スキーマ所有者のログイン資格証明をメンテナンスする必要がないため、オブジェクトへのアクセス権を付与でき、構成中に必要な接続数が減少します。

このシステム権限は Oracle が提供する DBA ロールに付属しているため、AS SYSDBA で接続するユーザー（ユーザー SYS）に（ADMIN OPTION 付きで）付与されます。他のシステム権限と同様に、GRANT ANY OBJECT PRIVILEGE システム権限を付与できるのは、ADMIN OPTION を所有しているユーザーのみです。

GRANT ANY OBJECT PRIVILEGE システム権限を使用してユーザーにオブジェクト権限を付与するときに、すでに GRANT OPTION 付きのオブジェクト権限を所有している場合、権限付与は通常の方法で実行されます。この場合は、権限所有者が権限を付与したユーザーとなります。権限を付与するユーザーがオブジェクト権限を所有していない場合は、GRANT ANY OBJECT PRIVILEGE システム権限を所有していても、オブジェクト所有者が権限を付与するユーザーとして実際の権限付与を実行します。

注意： GRANT 文により生成される監査レコードには、常に権限付与を実際に実行したユーザーが示されます。

たとえば、次のような場合を考えてみます。ユーザー adams が GRANT ANY OBJECT PRIVILEGE システム権限を持っているが、他の権限を付与する権限は持っていないとします。adams が次の文を発行します。

```
GRANT SELECT ON hr.employees TO blake WITH GRANT OPTION;
```

DBA_TAB_PRIVS ビューを調べると、権限を付与したユーザーとして hr が表示されることがわかります。

```
SQL> SELECT GRANTEE, OWNER, GRANTOR, PRIVILEGE, GRANTABLE
  2>     FROM DBA_TAB_PRIVS
  3>     WHERE TABLE_NAME = 'EMPLOYEES' and OWNER = 'HR';

GRANTEE  OWNER GRANTOR PRIVILEGE    GRANTABLE
-------- ----- ------- -----------  ----------
BLAKE    HR    HR       SELECT      YES
```

blake も GRANT ANY OBJECT PRIVILEGE システム権限を持っており、次の文を発行するとします。

```
GRANT SELECT ON hr.employees TO clark;
```

この場合は、DBA_TAB_PRIVS ビューを再び問い合せると、権限を付与したユーザーとして blake が表示されます。

```
GRANTEE  OWNER GRANTOR  PRIVILEGE    GRANTABLE
-------- ----- -------- --------     ----------
BLAKE    HR    HR       SELECT       YES
CLARK    HR    BLAKE    SELECT       NO
```

これは、blake がすでに hr.employees に対して GRANT OPTION 付きの SELECT 権限を持っているためです。

> **関連項目：** 25-17 ページ「オブジェクト所有者にかわるオブジェクト権限の取消し」

列に対する権限の付与

表の個々の列に対する INSERT、UPDATE または REFERENCES 権限を付与できます。

> **注意：** 列固有の INSERT 権限を付与する前に、NOT NULL 制約が定義されている列が表に含まれているかどうかを確認してください。INSERT できる列を選んで権限を付与したときに NOT NULL 列が抜けていると、ユーザーは表に行を挿入できません。このような状況を回避するために、各 NOT NULL 列が挿入可能であること、または NULL 以外のデフォルト値を持っていることを確認してください。そうでない場合、権限を付与されたユーザーは表に行を挿入できず、エラーが発生します。

次の文は、accounts 表の acct_no 列に対する INSERT 権限を scott に付与します。

```
GRANT INSERT (acct_no) ON accounts TO scott;
```

次の例では、emp 表の ename および job 列に対するオブジェクト権限が、ユーザー jfee および tsmith に付与されます。

```
GRANT INSERT(ename, job) ON emp TO jfee, tsmith;
```

ユーザー権限とロールの取消し

ここでは、ユーザー権限とロールの取消しについて説明します。この項の内容は、次のとおりです。

- システム権限とロールの取消し
- オブジェクト権限の取消し
- 権限の取消しによる連鎖的な影響

システム権限とロールの取消し

システム権限およびロールを取り消すには、SQL 文 REVOKE を使用します。

システム権限またはロールの ADMIN OPTION を持っているユーザーは、他のデータベース・ユーザーまたはロールから権限またはロールを取り消すことができます。取消しを行うユーザーは、権限またはロールを最初に付与されたユーザーでなくてもかまいません。GRANT ANY ROLE を持っているユーザーは、任意のロールを取り消すことができます。

次の文は、tsmith から CREATE TABLE システム権限と accts_rec ロールを取り消します。

```
REVOKE CREATE TABLE, accts_rec FROM tsmith;
```

注意： システム権限またはロールの ADMIN OPTION を選択的に取り消すことはできません。いったん権限またはロールを取り消してから、ADMIN OPTION を指定せずに同じ権限またはロールを再度付与してください。

オブジェクト権限の取消し

オブジェクト権限を取り消すには、REVOKE 文を使用します。オブジェクト権限を取り消すには、次のどちらかの条件を満たしている必要があります。

- 以前にユーザーまたはロールにオブジェクト権限を付与している。
- オブジェクト所有者のかわりに権限を付与したり取り消すことができる GRANT ANY OBJECT PRIVILEGE システム権限を付与されている。

取り消すことができるのは、権限を付与するユーザーとして直接認可した権限のみで、GRANT OPTION を付与された他のユーザーによる権限付与を取り消すことはできません。ただし、連鎖的な効果があります。権限を付与したユーザーのオブジェクト権限を取り消すと、GRANT OPTION を使用して伝播されたオブジェクト権限も取り消されます。

たとえば、最初に権限を付与したユーザー自身が次の文を発行して、ユーザー jfee と tsmith から emp 表の SELECT 権限と INSERT 権限を取り消します。

```
REVOKE SELECT, insert ON emp FROM jfee, tsmith;
```

次の文は、最初に human_resource ロールに付与した dept 表に対するすべてのオブジェクト権限を取り消します。

```
REVOKE ALL ON dept FROM human_resources;
```

注意： オブジェクト権限の GRANT OPTION を選択的に取り消すことはできません。いったんオブジェクト権限を取り消してから、GRANT OPTION を指定せずに同じオブジェクト権限を再度付与してください。ユーザーが自分自身からオブジェクト権限を取り消すことはできません。

オブジェクト所有者にかわるオブジェクト権限の取消し

GRANT ANY OBJECT PRIVILEGE システム権限を使用すると、オブジェクト所有者によって付与されたオブジェクト権限を指定して取り消すことができます。この操作を行うのは、オブジェクト権限がオブジェクト所有者によって付与されているか、GRANT ANY OBJECT PRIVILEGE システム権限を持つユーザーによって所有者のかわりに付与されている場合です。

オブジェクト権限がオブジェクト所有者と REVOKE 文を実行するユーザー（特定のオブジェクト権限と GRANT ANY OBJECT PRIVILEGE システム権限の両方を持つユーザー）によって付与されている場合は、REVOKE 文を発行したユーザーによって付与されたオブジェクト権限のみが取り消されます。この使用例については、25-14 ページの「オブジェクト所有者にかわるオブジェクト権限の付与」を参照してください。

今回は、blake が hr.employees に対する SELECT 権限を clark に付与しているとします。blake は GRANT ANY OBJECT PRIVILEGE システム権限のみでなく特定のオブジェクト権限も持っているため、この権限付与は blake に帰属します。hr も hr.employees に対する SELECT 権限を clark に付与するとします。DBA_TAB_PRIVS ビューの問合せでは、hr.employees 表について次の権限付与が有効であることが表示されます。

```
GRANTEE  OWNER GRANTOR PRIVILEGE    GRANTABLE
-------- ----- ------- -----------  ----------
BLAKE    HR    HR      SELECT       YES
CLARK    HR    BLAKE   SELECT       NO
CLARK    HR    HR      SELECT       NO
```

ユーザー blake が次の REVOKE 文を発行します。

```
REVOKE  SELECT ON hr.employees FROM clark;
```

blake が clark に付与したオブジェクト権限のみが削除されます。オブジェクト所有者 hr による権限付与はそのまま残ります。

```
GRANTEE  OWNER GRANTOR PRIVILEGE    GRANTABLE
-------- ----- ------- -----------  ----------
BLAKE    HR    HR       SELECT      YES
CLARK    HR    HR       SELECT      NO
```

blake が再度 REVOKE 文を発行すると、今度は hr によって付与されたオブジェクト権限が削除されます。

関連項目： 25-14 ページ「オブジェクト所有者にかわるオブジェクト権限の付与」

列固有のオブジェクト権限の取消し

表やビューの列を選択して INSERT、UPDATE および REFERENCES 権限を付与することはできますが、同じような REVOKE 文を使用して、列固有の権限を選択的に取り消すことはできません。かわりに、権限を付与するユーザーは表またはビューの全列に対するオブジェクト権限を最初に取り消し、次に残しておきたい列固有の権限を付与し直す必要があります。

たとえば、human_resources ロールに、dept 表の deptno 列および dname 列に対する UPDATE 権限が付与されているとします。その UPDATE 権限を deptno 列のみから取り消すには、次の 2 つの文を発行します。

```
REVOKE UPDATE ON dept FROM human_resources;
GRANT UPDATE (dname) ON dept TO human_resources;
```

この REVOKE 文によって、ロール human_resources から dept 表の全列に対する UPDATE 権限を取り消されます。GRANT 文によって、dname 列の UPDATE 権限が human_resources ロールに再度付与されます。

REFERENCES オブジェクト権限の取消し

REFERENCES オブジェクト権限を付与されたユーザーがその権限を使用して外部キー制約を作成し、その制約が現在も存在している場合、権限を付与したユーザーがその権限を取り消すには、REVOKE 文に必ず CASCADE CONSTRAINTS オプションを指定する必要があります。

```
REVOKE REFERENCES ON dept FROM jward CASCADE CONSTRAINTS;
```

CASCADE CONSTRAINTS 句を指定すると、取り消される REFERENCES 権限を使用して現在も定義されている外部キー制約が削除されます。

権限の取消しによる連鎖的な影響

権限のタイプによっては、権限の取消しによって連鎖的な影響が生じる場合があります。

システム権限

データ定義言語（DDL）操作に関連するシステム権限を取り消しても、影響は連鎖しません。これには、権限を付与したときに ADMIN OPTION を指定したかどうかは関係ありません。たとえば、次のような場合を考えてみます。

1. セキュリティ管理者が、ADMIN OPTION を使用して、jfee に CREATE TABLE システム権限を付与します。
2. ユーザー jfee が表を作成します。
3. ユーザー jfee が、tsmith に CREATE TABLE システム権限を付与します。
4. ユーザー tsmith が表を作成します。
5. セキュリティ管理者が、jfee から CREATE TABLE システム権限を取り消します。
6. ユーザー jfee の表はそのまま残ります。tsmith の表と CREATE TABLE システム権限はそのまま残ります。

連鎖的な影響は、データ操作言語（DML）操作に関連するシステム権限を取り消したときに生じる場合があります。ユーザーの SELECT ANY TABLE 権限を取り消すと、そのユーザーのスキーマ内に存在し、この権限に依存しているすべてのプロシージャは、権限が再度認可されないかぎり、正常に実行できません。

オブジェクト権限

オブジェクト権限を取り消すと、それに付随して連鎖的な影響が発生する場合があるため、REVOKE 文を発行する前に取消しの影響を調査する必要があります。

- DML オブジェクト権限を取り消すと、その DML オブジェクト権限に依存するオブジェクト定義にまで影響が及ぶ可能性があります。たとえば、test プロシージャのプロシージャ本体に、emp 表のデータを問い合せる SQL 文が記述されているとします。test プロシージャの所有者から emp 表の SELECT 権限を取り消すと、それ以降そのプロシージャを正常に実行できなくなります。
- 表に対する REFERENCES 権限をユーザーから取り消すと、そのユーザーが定義した外部キー整合性制約の中で、取り消された REFERENCES 権限を必要とするものが自動的に削除されます。たとえば、ユーザー jward に dept 表の deptno 列に対する REFERENCES 権限が付与されており、jward が emp 表の deptno 列に対して、deptno 列を参照する外部キーを作成したとします。dept 表の deptno 列に対する REFERENCES 権限を取り消すと、同じ操作で emp 表の deptno 列に対する外部キー制約も削除されます。

- 権限を付与したユーザーのオブジェクト権限を取り消すと、GRANT OPTION を使用して伝播されたオブジェクト権限も取り消されます。たとえば、user1 に GRANT OPTION 付きの SELECT オブジェクト権限を付与し、user1 が user2 に対して emp 表の SELECT 権限を付与したとします。その後、user1 から SELECT 権限を取り消します。この取消しは user2 にも連鎖します。user1 と user2 の取り消された SELECT 権限に依存するオブジェクトが、前述の箇条項目に記載されているような影響を受ける場合もあります。

ALTER または INDEX オブジェクト権限を取り消しても、ALTER および INDEX DDL の各オブジェクト権限を必要とするオブジェクト定義には影響は及びません。たとえば、別のユーザーの表に索引を作成したユーザーから INDEX 権限を取り消しても、その索引はそのまま残ります。

ユーザー・グループ PUBLIC に対する付与と取消し

ユーザー・グループ PUBLIC に対しても、権限とロールの付与および取消しが可能です。PUBLIC にはすべてのデータベース・ユーザーがアクセスできるため、PUBLIC に対して付与されたすべての権限またはロールには、すべてのデータベース・ユーザーがアクセスできます。

セキュリティ管理者とデータベース・ユーザーは、すべてのデータベース・ユーザーが権限またはロールを必要としている場合のみ、PUBLIC に権限またはロールを付与するようにします。これにより、各データベース・ユーザーは常に現在の作業を正常に実行するために必要な権限のみを持つという一般規則が保証されます。

PUBLIC から権限を取り消すと、かなりの規模で影響が連鎖する可能性があります。DML 操作に関連する権限（たとえば、SELECT ANY TABLE、UPDATE ON emp など）を PUBLIC から取り消した場合、ファンクションとパッケージを含むデータベース内のすべてのプロシージャは、再度使用する前に再認可する必要があります。そのため、DML に関連する権限を PUBLIC に付与または PUBLIC から取り消すときは、注意が必要です。

> **関連項目：** オブジェクト依存性の詳細は、21-23 ページの「オブジェクト依存性の管理」を参照してください。

権限の付与と取消しが有効になるとき

付与と取消しがいつ有効になるかは、付与または取り消す対象によって異なります。

- 任意の対象（ユーザー、ロールおよび PUBLIC）に対するシステム権限とオブジェクト権限の付与および取消しは、即時に有効になります。
- 任意の対象（ユーザー、他のロールおよび PUBLIC）に対するロールの付与 / 取消しが有効になるのは、付与 / 取消しの実行後、ロールを再度使用可能にするために現行のユーザー・セッションで SET ROLE 文を発行したとき、または付与 / 取消しを実行した後に新しくユーザー・セッションを作成したときです。

現在使用可能なロールは、SESSION_ROLES データ・ディクショナリ・ビューを問い合せることによって確認できます。

SET ROLE 文

ユーザーまたはアプリケーションは、セッション中に SET ROLE 文を何度でも使用して、現在そのセッションで使用可能になっているロールを変更できます。SET ROLE 文に指定するロールは、あらかじめ付与されている必要があります。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 によって制限されます。

次の例では、すでに付与されているロール clerk を使用可能にして、パスワードを指定します。

```
SET ROLE clerk IDENTIFIED BY bicentennial;
```

次の文を使用すると、すべてのロールを使用禁止にすることができます。

```
SET ROLE NONE;
```

デフォルト・ロールの指定

ユーザーがログインすると、そのユーザーに明示的に付与されている権限と、そのユーザーのデフォルト・ロールに含まれる権限が、すべて使用可能になります。

ユーザーのデフォルト・ロールのリストは、ALTER USER 文を使用して設定および変更できます。ALTER USER 文を使用すると、ユーザーがデータベースに接続するときに、ロールのパスワードを指定しなくても使用可能になるロールを指定できます。ユーザーには、そのロールが GRANT 文で直接付与されている必要があります。ディレクトリ・サービスを含む外部サービスによって管理されているロール（外部ロールまたはグローバル・ロール）は、デフォルト・ロールとして指定できません。

次の例では、ユーザー jane のデフォルト・ロールを設定しています。

```
ALTER USER jane DEFAULT ROLE payclerk, pettycash;
```

CREATE USER 文ではユーザーのデフォルト・ロールを設定できません。最初にユーザーを作成したときに、ユーザーのデフォルト・ロールは ALL に設定されます。これにより、その後ユーザーに付与されるロールはすべてデフォルト・ロールになります。ユーザーのデフォルト・ロールを制限するには、ALTER USER 文を使用します。

注意： ユーザー・ロール以外のロールを作成すると、ロールを作成したユーザーにそのロールが暗黙的に付与され、デフォルト・ロールとして追加されます。ロールの数が MAX_ENABLED_ROLES を超えている場合は、ログイン時にエラーが発生します。このエラーを回避するには、ユーザーのデフォルト・ロールの数を MAX_ENABLED_ROLES より少なくします。したがって、ユーザー・ロールを作成する前に、SYS と SYSTEM の DEFAULT ROLE の設定を変更する必要があります。

ユーザーが使用可能にできるロール数の制限

各ユーザーは、初期化パラメータ MAX_ENABLED_ROLES で指定された数だけロールを使用可能にできます。間接的に付与され、1 次ロールが使用可能になった結果として使用可能になるロールもすべてその数に含まれます。データベース管理者（DBA）は、このパラメータの値を変更することによって、この制限を変更できます。各ユーザー・セッションで同時に使用可能にするロール数を増やすには、大きい値を設定します。ただし、このパラメータの値を大きくすればするほど、各ユーザー・セッションにより多くのメモリー領域が必要になります。これは、ユーザー・セッションごとのプログラム・グローバル領域（PGA）サイズが影響を受け、ロール当たり 4 バイト必要となるためです。1 人のユーザーが同時に使用可能にできるロールの最大数を決定し、その値を MAX_ENABLED_ROLES パラメータに使用してください。

オペレーティング・システムまたはネットワークを使用したロールの付与

ここでは、オペレーティング・システムまたはネットワークを使用したロールの付与について説明します。この項の内容は、次のとおりです。

- オペレーティング・システムのロール識別機能の使用
- オペレーティング・システムのロール管理機能の使用
- OS_ROLES=TRUE の場合のロールの付与および取消し
- OS_ROLES=TRUE の場合にロールを使用可能および使用禁止にする方法
- オペレーティング・システムによるロール管理使用時のネットワーク接続の使用

セキュリティ管理者が GRANT 文と REVOKE 文を使用して、ユーザーに対し明示的にデータベース・ロールを付与または取り消すかわりに、Oracle が稼働しているオペレーティング・システムによって、接続時にユーザーにロールを付与できます。つまり、オペレーティング・システムを使用してロールを管理し、ユーザーのセッション作成時に Oracle にその

ロールを渡すことができます。このメカニズムの一部として、各ユーザーのデフォルト・ロールや、ADMIN OPTION 付きでユーザーに付与するロールを識別できます。オペレーティング・システムを使用してロールを使用するユーザーを認可する場合でも、必ずすべてのロールをデータベース内に作成し、GRANT 文を使用してそのロールに権限を割り当てる必要があります。

ロールは、ネットワーク・サービスを介しても付与できます。

ユーザーのデータベース・ロールを識別するためにオペレーティング・システムを使用する方法の利点は、Oracle データベースの権限管理をデータベースの外部で実施できることです。これにより、オペレーティング・システムに組み込まれているセキュリティ機能によって、ユーザーの権限が制御されます。また、このオプションを使用すると、大量のシステム・アクティビティに対してセキュリティを集中管理できるので、次のような状況に役立ちます。

- MVS 版 Oracle の管理者が、データベース・ユーザーのロールを識別するために RACF グループを使用する場合
- UNIX 版 Oracle の管理者が、データベース・ユーザーのロールを識別するために UNIX グループを使用する場合
- VMS 版 Oracle の管理者が、データベース・ユーザーのロールを識別するために権限識別子を使用する場合

ユーザーのデータベース・ロールを識別するためにオペレーティング・システムを使用する方法の主な欠点は、権限管理がロール・レベルでしか実施できないことです。個々の権限は、オペレーティング・システムを使用して付与することはできません。ただし、GRANT 文を使用してデータベース内で付与することは可能です。

この機能を使用する際の第 2 の欠点は、オペレーティング・システムがロールを管理している場合に、デフォルトではユーザーが共有サーバーまたはその他のネットワーク接続を介してデータベースに接続できないということです。ただし、このデフォルトは変更できます。25-26 ページの「オペレーティング・システムによるロール管理使用時のネットワーク接続の使用」を参照してください。

注意： この項で説明されている機能は、一部のオペレーティング・システムでしか使用できません。これらの機能が使用できるかどうかを確認するには、オペレーティング・システム固有の Oracle マニュアルを参照してください。

オペレーティング・システムのロール識別機能の使用

セッションの作成時にオペレーティング・システムを使用して各ユーザーのデータベース・ロールを識別するようにデータベースの運用を変更するには、初期化パラメータ OS_ROLES を TRUE に設定します（インスタンスが実行中の場合は再起動が必要）。ユーザーがセッションを作成しようとすると、Oracle はオペレーティング・システムによって識別されるデータベース・ロールを使用して、そのユーザーのセキュリティ・ドメインを初期化します。

ユーザーのデータベース・ロールを識別するために、各 Oracle ユーザーのオペレーティング・システム・アカウントは、そのユーザーが使用できるデータベース・ロールを示すオペレーティング・システム識別子を必ず持つ必要があります。この識別子は、グループ、権限識別子またはこれらと似たような名前で呼ばれています。ロールの指定では、ユーザーのデフォルト・ロールや ADMIN OPTION 付きのロールを示すこともできます。オペレーティング・システム・レベルでのロールの指定形式は、次のとおりです。この形式は、どのオペレーティング・システムを使用している場合でも同じです。

ora_*ID_ROLE*[_[d][a]]

各項目の意味は次のとおりです。

- ID の定義は、オペレーティング・システムによって異なります。たとえば、VMS では、ID はデータベースのインスタンス識別子です。MVS ではマシン・タイプ、UNIX ではシステム ID です。

  注意： ID は、ORACLE_SID と照合する際に大 / 小文字が区別されます。ROLE では、大 / 小文字は区別されません。

- ROLE は、データベース・ロールの名前です。
- d は、このロールがデータベース・ユーザーのデフォルト・ロールであることを示すオプション文字です。
- a は、このロールが ADMIN OPTION 付きでユーザーに付与されることを示すオプション文字です。このオプション文字を指定することによって、ユーザーはこのロールを他のロールにのみ付与できるようになります。オペレーティング・システムを使用してロールを管理している場合は、ユーザーにロールを付与できません。

  注意： 文字 d または a のどちらかを指定する場合は、その文字の直前にアンダースコアを指定してください。

たとえば、オペレーティング・システム・アカウントが、プロファイルで識別される次のロールを持っているとします。

```
ora_PAYROLL_ROLE1
ora_PAYROLL_ROLE2_a
ora_PAYROLL_ROLE3_d
ora_PAYROLL_ROLE4_da
```

対応するユーザーが Oracle の payroll インスタンスに接続すると、role3 と role4 がデフォルト・ロールになり、role2 と role4 が ADMIN OPTION 付きで付与されます。

オペレーティング・システムのロール管理機能の使用

オペレーティング・システムによって管理されているロールを使用する場合は、データベース・ロールがオペレーティング・システムのユーザーに付与されるということに注意してください。オペレーティング・システム・ユーザーが接続できるデータベース・ユーザーは、認可されたデータベース・ロールを使用できます。このため、OS_ROLES = TRUE を使用している場合は、権限が付与されているオペレーティング・システム・アカウントにデータベース・アカウントを対応付けるため、すべての Oracle ユーザーを IDENTIFIED EXTERNALLY として定義することを考慮してください。

OS_ROLES=TRUE の場合のロールの付与および取消し

OS_ROLES が TRUE に設定されている場合は、ユーザーに対するロールの付与と取消しがオペレーティング・システムによって完全に管理されます。それまでに GRANT 文によってユーザーに付与されたロールは、データ・ディクショナリ内には残っているものの、適用はされません。オペレーティング・システム・レベルでのユーザーへのロールの付与のみが適用されます。ユーザーは権限をロールとユーザーに付与できます。

注意： オペレーティング・システムによって ADMIN OPTION 付きでロールが付与された場合、ユーザーはそのロールを他のロールにのみ付与できます。

OS_ROLES=TRUE の場合にロールを使用可能および使用禁止にする方法

OS_ROLES が TRUE に設定されている場合、オペレーティング・システムによって付与されたロールは、SET ROLE 文を使用して動的に使用可能にできます。ロールがパスワードやオペレーティング・システムによる認可を必要とするように定義されていた場合でも、この文は適用されます。ただし、ユーザーのオペレーティング・システム・アカウントで識別されないロールは、SET ROLE 文に指定できません。これは、OS_ROLES = FALSE のときに GRANT 文を使用してロールを付与していた場合でも同じです。このようなロールを指定しても、無視されます。

OS_ROLES = TRUE の場合、ユーザーは初期化パラメータ MAX_ENABLED_ROLES で指定されている数までロールを使用可能にできます。

オペレーティング・システムによるロール管理使用時のネットワーク接続の使用

オペレーティング・システムでロールを管理する場合、デフォルトでは、ユーザーは共有サーバーを介してデータベースに接続できません。リモート・ユーザーはセキュリティで保護されていない接続を介して別のオペレーティング・システム・ユーザーになりすますおそれがあるため、デフォルトでこのような制限が課されています。

セキュリティの心配がない場合は、データベースの初期化パラメータ・ファイルで初期化パラメータ REMOTE_OS_ROLES を TRUE に設定することにより、共有サーバーまたはその他のネットワーク接続でオペレーティング・システムのロール管理を使用できます。この変更は、次にインスタンスを起動し、データベースをマウントするときに有効となります。このパラメータのデフォルト設定は FALSE です。

権限とロールに関する情報の表示

権限とロールの付与に関する情報にアクセスするには、次のデータ・ディクショナリ・ビューを問い合せます。

ビュー	説明
DBA_COL_PRIVS ALL_COL_PRIVS USER_COL_PRIVS	DBA ビューには、データベース内のすべての列オブジェクトが表示されます。ALL ビューには、オブジェクトの所有者、権限付与者または権限受領者が現行ユーザーまたは PUBLIC である列オブジェクトの権限付与がすべて表示されます。USER ビューには、オブジェクトの所有者、権限付与者または権限受領者が現行ユーザーである列オブジェクトの権限付与が表示されます。
ALL_COL_PRIVS_MADE USER_COL_PRIVS_MADE	ALL ビューには、オブジェクトの所有者または権限付与者が現行ユーザーである列オブジェクトの権限付与がリストされます。USER ビューには、権限付与者が現行ユーザーである列オブジェクトの権限付与が表示されます。
ALL_COL_PRIVS_RECD USER_COL_PRIVS_RECD	ALL ビューには、権限受領者が現行ユーザーまたは PUBLIC である列オブジェクトの権限付与が表示されます。USER ビューには、権限受領者が現行ユーザーである列オブジェクトの権限付与が表示されます。
DBA_TAB_PRIVS ALL_TAB_PRIVS USER_TAB_PRIVS	DBA ビューには、データベース内のすべてのオブジェクトに対するすべての権限付与がリストされます。ALL ビューには、権限受領者が現行ユーザーまたは PUBLIC であるオブジェクトの権限付与がリストされます。USER ビューには、権限受領者が現行ユーザーであるすべてのオブジェクトの権限付与がリストされます。

ビュー	説明
ALL_TAB_PRIVS_MADE USER_TAB_PRIVS_MADE	ALL ビューには、現行ユーザーが行ったオブジェクトの権限付与、または現行ユーザーが所有するオブジェクトに対する権限付与がすべてリストされます。USER ビューには、現行ユーザーが所有しているすべてのオブジェクトの権限付与がリストされます。
ALL_TAB_PRIVS_RECD USER_TAB_PRIVS_RECD	ALL ビューには、権限受領者が現行ユーザーまたは PUBLIC であるオブジェクトの権限付与がリストされます。USER ビューには、権限受領者が現行ユーザーであるオブジェクトの権限付与がリストされます。
DBA_ROLES	このビューには、データベース内に存在するすべてのロールがリストされます。
DBA_ROLE_PRIVS USER_ROLE_PRIVS	DBA ビューには、ユーザーとロールに付与されているロールがリストされます。USER ビューには、現行ユーザーに付与されているロールがリストされます。
DBA_SYS_PRIVS USER_SYS_PRIVS	DBA ビューには、ユーザーとロールに付与されているシステム権限がリストされます。USER ビューには、現行ユーザーに付与されているシステム権限がリストされます。
ROLE_ROLE_PRIVS	このビューには、他のロールに付与されているロールが表示されます。表示される情報は、ユーザーがアクセス可能なロールに関するもののみです。
ROLE_SYS_PRIVS	このビューには、ロールに付与されているシステム権限に関する情報が含まれています。表示される情報は、ユーザーがアクセス可能なロールに関するもののみです。
ROLE_TAB_PRIVS	このビューには、ロールに付与されているオブジェクト権限に関する情報が含まれています。表示される情報は、ユーザーがアクセス可能なロールに関するもののみです。
SESSION_PRIVS	このビューには、ユーザーに対して現在使用可能になっている権限がリストされます。
SESSION_ROLES	このビューには、ユーザーに対して現在使用可能になっているロールがリストされます。

次に、これらのビューの使用例をいくつか示します。各例では、次の文がすでに発行されていることを前提としています。

```
CREATE ROLE security_admin IDENTIFIED BY honcho;

GRANT CREATE PROFILE, ALTER PROFILE, DROP PROFILE,
    CREATE ROLE, DROP ANY ROLE, GRANT ANY ROLE, AUDIT ANY,
    AUDIT SYSTEM, CREATE USER, BECOME USER, ALTER USER, DROP USER
    TO security_admin WITH ADMIN OPTION;

GRANT SELECT, DELETE ON SYS.AUD$ TO security_admin;

GRANT security_admin, CREATE SESSION TO swilliams;
```

```
GRANT security_admin TO system_administrator;

GRANT CREATE SESSION TO jward;

GRANT SELECT, DELETE ON emp TO jward;

GRANT INSERT (ename, job) ON emp TO swilliams, jward;
```

関連項目： これらのデータ・ディクショナリ・ビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

付与されているすべてのシステム権限のリスト

次の問合せを実行すると、ロールとユーザーに対して付与されているシステム権限がすべて表示されます。

```
SELECT * FROM DBA_SYS_PRIVS;

GRANTEE            PRIVILEGE                        ADM
--------------     --------------------------------- ---
SECURITY_ADMIN     ALTER PROFILE                    YES
SECURITY_ADMIN     ALTER USER                       YES
SECURITY_ADMIN     AUDIT ANY                        YES
SECURITY_ADMIN     AUDIT SYSTEM                     YES
SECURITY_ADMIN     BECOME USER                      YES
SECURITY_ADMIN     CREATE PROFILE                   YES
SECURITY_ADMIN     CREATE ROLE                      YES
SECURITY_ADMIN     CREATE USER                      YES
SECURITY_ADMIN     DROP ANY ROLE                    YES
SECURITY_ADMIN     DROP PROFILE                     YES
SECURITY_ADMIN     DROP USER                        YES
SECURITY_ADMIN     GRANT ANY ROLE                   YES
SWILLIAMS          CREATE SESSION                   NO
JWARD              CREATE SESSION                   NO
```

付与されているすべてのロールのリスト

次の問合せを実行すると、ユーザーと他のロールに対して付与されているロールがすべて表示されます。

```
SELECT * FROM DBA_ROLE_PRIVS;

GRANTEE            GRANTED_ROLE                        ADM
------------------ ----------------------------------- ---
SWILLIAMS          SECURITY_ADMIN                      NO
```

ユーザーに付与されているオブジェクト権限のリスト

次の問合せを実行すると、特定のユーザーに対して付与されているオブジェクト権限（列固有の権限を除く）がすべて表示されます。

```
SELECT TABLE_NAME, PRIVILEGE, GRANTABLE FROM DBA_TAB_PRIVS
    WHERE GRANTEE = 'JWARD';

TABLE_NAME   PRIVILEGE    GRANTABLE
-----------  ------------ ----------
EMP          SELECT       NO
EMP          DELETE       NO
```

すでに付与されている列固有の権限をすべて表示するには、次の問合せを使用します。

```
SELECT GRANTEE, TABLE_NAME, COLUMN_NAME, PRIVILEGE
    FROM DBA_COL_PRIVS;

GRANTEE      TABLE_NAME     COLUMN_NAME      PRIVILEGE
-----------  ------------   -------------    --------------
SWILLIAMS    EMP            ENAME            INSERT
SWILLIAMS    EMP            JOB              INSERT
JWARD        EMP            NAME             INSERT
JWARD        EMP            JOB              INSERT
```

セッションの現在の権限ドメインのリスト

次の問合せを実行すると、発行者が現在使用できるロールがすべてリストされます。

```
SELECT * FROM SESSION_ROLES;
```

swilliams に対して security_admin ロールが使用可能になっている場合にこの問合せを実行すると、次の情報が表示されます。

```
ROLE
------------------------------
SECURITY_ADMIN
```

次の問合せを実行すると、発行者のセキュリティ・ドメインで現在使用可能なシステム権限がすべて表示されます。これには、明示的に付与されている権限と使用可能なロールから付与された権限がどちらも含まれています。

```
SELECT * FROM SESSION_PRIVS;
```

swilliams に対して security_admin ロールが使用可能になっている場合にこの問合せを実行すると、次の情報が表示されます。

```
PRIVILEGE
----------------------------------------
AUDIT SYSTEM
CREATE SESSION
CREATE USER
BECOME USER
ALTER USER
DROP USER
CREATE ROLE
DROP ANY ROLE
GRANT ANY ROLE
AUDIT ANY
CREATE PROFILE
ALTER PROFILE
DROP PROFILE
```

swilliams に対して security_admin ロールが使用禁止になっている場合、最初の問合せでは何も表示されず、2 番目の問合せでは CREATE SESSION 権限に関する行が 1 行のみ表示されます。

データベースのロールのリスト

データベースのすべてのロールと各ロールに対して使用されている認証を表示するには、DBA_ROLES データ・ディクショナリ・ビューを使用します。たとえば、次の問合せを実行すると、データベース内のすべてのロールが表示されます。

```
SELECT * FROM DBA_ROLES;

ROLE                  PASSWORD
----------------      --------
CONNECT               NO
RESOURCE              NO
DBA                   NO
SECURITY_ADMIN        YES
```

ロールの権限ドメイン情報のリスト

ROLE_ROLE_PRIVS、ROLE_SYS_PRIVS、ROLE_TAB_PRIVS の各データ・ディクショナリ・ビューには、ロールの権限ドメイン情報が含まれています。

たとえば、次の問合せは system_admin ロールに付与されているロールをすべてリストします。

```
SELECT GRANTED_ROLE, ADMIN_OPTION
   FROM ROLE_ROLE_PRIVS
   WHERE ROLE = 'SYSTEM_ADMIN';

GRANTED_ROLE              ADM
----------------          ----
SECURITY_ADMIN            NO
```

次の問合せを実行すると、security_admin ロールに付与されているシステム権限がすべて表示されます。

```
SELECT * FROM ROLE_SYS_PRIVS WHERE ROLE = 'SECURITY_ADMIN';

ROLE                     PRIVILEGE                     ADM
------------------------ ----------------------------  ---
SECURITY_ADMIN            ALTER PROFILE                YES
SECURITY_ADMIN            ALTER USER                   YES
SECURITY_ADMIN            AUDIT ANY                    YES
SECURITY_ADMIN            AUDIT SYSTEM                 YES
SECURITY_ADMIN            BECOME USER                  YES
SECURITY_ADMIN            CREATE PROFILE               YES
SECURITY_ADMIN            CREATE ROLE                  YES
SECURITY_ADMIN            CREATE USER                  YES
SECURITY_ADMIN            DROP ANY ROLE                YES
SECURITY_ADMIN            DROP PROFILE                 YES
SECURITY_ADMIN            DROP USER                    YES
SECURITY_ADMIN            GRANT ANY ROLE               YES
```

次の問合せを実行すると、security_admin ロールに付与されているオブジェクト権限がすべて表示されます。

```
SELECT TABLE_NAME, PRIVILEGE FROM ROLE_TAB_PRIVS
    WHERE ROLE = 'SECURITY_ADMIN';

TABLE_NAME                     PRIVILEGE
---------------------------    ----------------
AUD$                           DELETE
AUD$                           SELECT
```

26

データベース使用の監査

この章では、Oracle データベース・サーバーの監査機能の使用方法について説明します。この章の内容は、次のとおりです。

- 監査のガイドライン
- 監査証跡に記録される情報
- デフォルトで監査されるアクション
- 管理ユーザーの監査
- 監査証跡の管理
- ファイングレイン監査
- データベース監査証跡情報の表示

監査のガイドライン

ここでは、監査のガイドラインについて説明します。この項の内容は、次のとおりです。

- データベースまたはオペレーティング・システム監査証跡の使用の決定
- 監査済み情報の管理しやすい状態での維持
- 疑わしいデータベース・アクティビティの監査のガイドライン
- 通常のデータベース・アクティビティの監査のガイドライン

データベースまたはオペレーティング・システム監査証跡の使用の決定

すべてのデータベースのデータ・ディクショナリには SYS.AUD$ という名前の表があり、通常、これをデータベースの**監査証跡**と呼びます。この監査証跡は、データベースの文、権限またはスキーマ・オブジェクトの監査エントリを格納するように設計されています。

必要な場合は、データベース監査情報をオペレーティング・システム・ファイルに格納するように選択できます。オペレーティング・システムの監査機能によって生成された監査レコードがオペレーティング・システムの監査証跡に格納されており、Oracle がそこに書き込める場合は、データベース監査エントリをこのファイルに書き込むように選択できます。たとえば、Windows オペレーティング・システムでは、Oracle が監査レコードをアプリケーションのイベント・ログにイベントとして書き込むことができます。

データベース監査レコードの格納に、データベース監査証跡またはオペレーティング・システム監査証跡を使用する場合のメリットとデメリットを考えてみます。

データベース監査証跡を使用する利点は、次のとおりです。

- データ・ディクショナリ内に事前に定義された監査証跡ビューを使用して、選択した部分の監査証跡を表示できます。
- Oracle のツール製品（Oracle Reports など）を使用して、監査レポートを生成できます。

一方、オペレーティング・システム監査証跡では、Oracle やその他のアプリケーションを含む複数のソースからの監査レコードを整理統合できます。そのため、すべての監査レコードが 1 箇所にまとめられ、システム・アクティビティの調査をより効率的に行うことができます。

> **関連項目：** オペレーティング・システムの監査機能については、そのオペレーティング・システム固有のマニュアルを参照してください。

監査済み情報の管理しやすい状態での維持

監査は比較的コストがかかりませんが、監査するイベントの数は可能なかぎり制限する必要があります。これにより、監査される文を実行したときのパフォーマンスの影響を最小限に抑えるとともに、監査証跡のサイズも最小限にすることができます。

監査方針を作成する際は、次の一般的なガイドラインに従ってください。

- 監査目的を評価する

  監査の目的を明確にしておくと、適切な監査方針を打ち出せるため、不必要な監査をしなくて済みます。

  たとえば、不審なデータベース・アクティビティの調査のために監査すると仮定します。この情報のみでは不明確です。どのデータベース・アクティビティが疑わしい、または注意を要するといった具体的な情報が必要です。そのために、たとえば、データベース内の表から無許可にデータが削除されていないかを監査するというように、監査目的を絞り込みます。このような目的を設定すれば、監査の対象となるアクティビティの種類や、疑わしいアクティビティによって影響を受けるオブジェクトの種類を限定できます。

- 監査について十分理解する

  対象となる情報を取得するために必要な最小限の文、ユーザーまたはオブジェクトを監査します。これにより、不必要な監査情報のために重要な情報の識別が困難になることや、SYSTEM 表領域内の貴重な領域が無駄に消費されることがなくなります。収集が必要なセキュリティ情報の量と、その情報を格納および処理する能力とのバランスを保つ必要があります。

  たとえば、データベース・アクティビティに関する情報を収集するために監査する場合は、追跡するアクティビティの種類を正確に決定した上で、必要な情報を収集するために必要な期間にかぎり、目的のアクティビティのみを監査します。各セッションの論理 I/O 情報にのみ関心がある場合は、オブジェクトを監査しないでください。

疑わしいデータベース・アクティビティの監査のガイドライン

監査の目的が疑わしいデータベース・アクティビティを監視することである場合は、次のガイドラインに従ってください。

- 一般的な監査の後、特定の対象を監査する

  通常、疑わしいデータベース・アクティビティの監査を開始する時点では、対象のユーザーまたはスキーマ・オブジェクトを特定するために使用できる情報はあまりありません。このため、最初は一般的な監査オプションを設定する必要があります。準備的な監査情報の記録と分析を終えた後、一般的な監査オプションを停止し、特定の監査オプションを使用可能にします。この処理は、疑わしいデータベース・アクティビティの原因について具体的な結論が出せるだけの十分な裏付けが収集できるまで継続してください。

- 監査証跡を保護する

  疑わしいデータベース・アクティビティを監査する場合は、監査と無関係に監査情報が追加、変更または削除されないように、監査証跡を保護します。

  関連項目： 26-17 ページ 「監査証跡の保護」

通常のデータベース・アクティビティの監査のガイドライン

監査目的が特定のデータベース・アクティビティに関する履歴情報を収集することである場合は、次のガイドラインに従ってください。

- 関連のあるアクションのみを監査する

  役に立たない監査レコードのために重要な情報が識別できない事態を避け、監査証跡管理の量を削減するために、目的のデータベース・アクティビティのみを監査してください。

- 監査レコードをアーカイブし、監査証跡を削除する

  必要な情報を収集した後、目的の監査レコードをアーカイブし、この情報の監査証跡を削除します。

監査証跡に記録される情報

Oracle は、データベース監査証跡またはオペレーティング・システム・ファイル、あるいはその両方にレコードを書き込むことができます。この項では、この監査証跡情報の構成について説明します。

データベース監査証跡に格納される情報

データベース監査証跡は SYS.AUD$ 表に格納され、監査されるイベントや監査オプションのセットに応じて、異なるタイプの情報が記録されます。監査証跡レコードに必ず記録される情報は、次のとおりです。

- オペレーティング・システムのログイン・ユーザー名
- ユーザー名
- セッション識別子
- 端末識別子
- アクセスされたスキーマ・オブジェクトの名前
- 実行または試行された操作
- 操作の完了コード
- 日付と時刻のタイムスタンプ

監査証跡には、監査された文に関連するデータ値の情報は格納されません。たとえば、UPDATE 文を監査しているときに、更新された行の新旧のデータ値は格納されません。ただし、ファイングレイン監査方法を使用すれば、この特別なタイプの監査を実行できます。

> **関連項目：** ファイングレイン監査の詳細は、26-17 ページの「ファイングレイン監査」を参照してください。

オペレーティング・システム・ファイルに格納される情報

監査証跡を含むオペレーティング・システム・ファイルには、次の情報を記録できます。

- オペレーティング・システムによって生成された監査レコード
- データベース監査証跡レコード
- 常に監査されるデータベース関連のアクション
- 管理ユーザー（SYS）用の監査レコード

オペレーティング・システム監査証跡に書き込まれた監査証跡レコードにはエンコードされた情報が含まれていますが、この情報はデータ・ディクショナリ表とエラー・メッセージを使用して次のようにデコードできます。

エンコードされている情報	デコード方法
アクション・コード	このコードは、実行または試行された操作を示します。これらのコードとその説明のリストは、AUDIT_ACTIONS データ・ディクショナリ表に含まれています。
使用された権限	操作の実行に使用されたシステム権限を示します。これらのコードとその説明のリストは、SYSTEM_PRIVILEGE_MAP 表に含まれています。
完了コード	試行された操作の結果を示します。操作が正常に終了すると値 0（ゼロ）が戻り、失敗すると操作の失敗原因を示す Oracle エラー・コードが戻ります。これらのコードは、『Oracle9*i* データベース・エラー・メッセージ』を参照してください。

デフォルトで監査されるアクション

データベース監査が使用可能かどうかにかかわらず、Oracle は常にデータベースに関連する特定の操作を監査し、オペレーティング・システム監査ファイルに書き込みます。このような操作には次のものがあります。

- 管理者権限によるインスタンスへの接続

  SYSOPER または SYSDBA として Oracle に接続したオペレーティング・システム・ユーザーについて記述した監査レコードが生成されます。これにより、管理権限を持つユーザーの責任範囲が明確になります。このようなユーザーの完全監査は、26-6 ページの「管理ユーザーの監査」の説明に従って使用可能にすることができます。

- データベースの起動

  インスタンスを起動したオペレーティング・システム・ユーザー、そのユーザーの端末識別子、日付と時刻のタイムスタンプおよびデータベース監査が使用可能かどうかなどを記述した監査レコードが生成されます。データベース監査証跡は起動処理が正常に完了するまで使用できないため、この監査レコードはオペレーティング・システム監査証

跡に格納されます。起動時にデータベース監査の状態が記録されるので、データベース監査が使用禁止の状態でデータベースを再起動した場合がわかります。これにより、管理者は監査されずにアクションを実行できます。

- データベースの停止

  インスタンスを停止したオペレーティング・システム・ユーザー、そのユーザーの端末識別子および日付と時刻のタイムスタンプを記述した監査レコードが生成されます。

管理ユーザーの監査

SYS として接続したユーザー（SYSDBA または SYSOPER として接続したすべてのユーザーを含む）のセッションは、完全に監査できます。AUDIT_SYS_OPERATIONS 初期化パラメータを使用して、ユーザー SYS を監査するかどうかを指定します。たとえば、次の設定では、SYS を監査対象として指定しています。

```
AUDIT_SYS_OPERATIONS = TRUE
```

値がデフォルトである FALSE の場合、SYS の監査は使用禁止になります。

SYS に関するすべての監査レコードは、SYS.AUD$ ではなく、監査証跡を含むオペレーティング・システム・ファイルに書き込まれます。SYS が発行したすべての SQL 文が、AUDIT_TRAIL 初期化パラメータの設定に関係なく無差別に監査されます。

次の SYS セッションを考えてみます。

```
CONNECT / AS SYSDBA;
ALTER SYSTEM FLUSH SHARED_POOL;
UPDATE salary SET base=1000 WHERE name='myname';
```

SYS の監査が使用可能になっている場合は、ALTER SYSTEM 文と UPDATE 文の両方がオペレーティング・システム監査ファイルに次のように表示されます。

```
Thu Jan 24 12:58:00 2002
ACTION: 'CONNECT'
DATABASE USER: '/'
OSPRIV: SYSDBA
CLIENT USER: scott
CLIENT TERMINAL: pts/2
STATUS: 0

Thu Jan 24 12:58:00 2002
ACTION: 'alter system flush shared_pool'
DATABASE USER: ''
OSPRIV: SYSDBA
CLIENT USER: scott
CLIENT TERMINAL: pts/2
STATUS: 0
```

```
Thu Jan 24 12:58:00 2002
ACTION: 'update salary set base=1000 where name='myname''
DATABASE USER: ''
OSPRIV: SYSDBA
CLIENT USER: scott
CLIENT TERMINAL: pts/2
STATUS: 0
```

SYSDBA として接続したユーザーはスーパー・ユーザー権限を使用できるため、DBA にはこの接続を必要な場合にのみ使用することをお薦めします。日常的なメンテナンス・アクティビティは、DBA ロールが割り当てられている DBA が通常どおりに実行できます。

監査証跡の管理

ここでは、監査証跡情報の管理について説明します。内容は、次のとおりです。

- 監査の使用可能および使用禁止
- 監査オプションの設定
- 複数層環境での監査
- 監査オプションを使用禁止にする方法
- 監査証跡の増加とサイズの制御
- 監査証跡の保護

監査の使用可能および使用禁止

許可されたデータベース・ユーザーであれば、文、権限およびオブジェクト監査オプションをいつでも設定できますが、データベース監査が使用可能でないかぎり、Oracle はデータベース監査証跡用の監査情報を生成しません。通常は、セキュリティ管理者が監査の制御を担当します。

ここでは、監査を使用可能および使用禁止にする初期化パラメータについて説明します。

注意： 監査に影響する初期化パラメータは、すべて静的です。つまり、AUDIT_SYS_OPERATIONS、AUDIT_TRAIL および AUDIT_FILE_DEST 初期化パラメータの値を変更した場合は、新しい値が有効になるようにデータベースを停止して再起動する必要があります。

AUDIT_TRAIL 初期化パラメータの設定

データベース監査は、データベースの初期化パラメータ・ファイル内の AUDIT_TRAIL 初期化パラメータによって使用可能または使用禁止にします。このパラメータには、次の値を設定できます。

パラメータ値	意味
DB	データベース監査を使用可能にして、常にオペレーティング・システム監査証跡に書き込まれるレコードを除き、すべての監査レコードをデータベース監査証跡に書き込みます。
OS	データベース監査を使用可能にして、すべての監査レコードをオペレーティング・システム・ファイルに書き込みます。
NONE	監査を使用禁止にします。この値がデフォルトです。

AUDIT_FILE_DEST 初期化パラメータの設定

AUDIT_FILE_DEST 初期化パラメータでは、AUDIT_TRAIL=OS を指定した場合に監査証跡が書き込まれるオペレーティング・システム・ディレクトリを指定します。このディレクトリには、必須監査情報も書き込まれ、AUDIT_SYS_OPERATIONS 初期化パラメータで指定した場合は、ユーザー SYS の監査レコードも書き込まれます。

AUDIT_FILE_DEST パラメータを指定しない場合、デフォルト・ディレクトリは $ORACLE_HOME/rdbms/audit です。

注意：

- オペレーティング・システムで監査証跡がサポートされている場合、その書込み先はオペレーティング・システム固有です。たとえば、Windows オペレーティング・システムでは、監査レコードはアプリケーションのイベント・ログにイベントとして書き込まれます。これらのイベントは、イベント・ビューアを使用して表示し、管理できます。Windows プラットフォームには AUDIT_FILE_DEST 初期化パラメータを指定できません。詳細は、『Oracle9*i* Database for Windows 管理者ガイド』を参照してください。
- 一部のオペレーティング・システムでは、インスタンス接続とデータベース起動の監査レコードが、AUDIT_FILE_DEST の設定に関係なく常にデフォルトの $ORACLE_HOME/rdbms/audit に書き込まれます。これは、データベースがマウントされるまで、パラメータ設定が認識されないためです。

監査オプションの設定

監査オプションを指定するには、AUDIT 文を使用します。AUDIT 文では、次の 3 つのレベルで監査オプションを設定できます。

レベル	効果
文	特定の型のデータベース・オブジェクトに影響を与える特定の SQL 文または文のグループを監査します。たとえば、AUDIT TABLE 文を実行すると、CREATE TABLE、TRUNCATE TABLE、COMMENT ON TABLE および DELETE [FROM] TABLE 文が監査されます。
権限	特定のシステム権限によって許可される SQL 文を監査します。たとえば、AUDIT CREATE ANY TRIGGER 文を実行すると、CREATE ANY TRIGGER システム権限を使用して発行された文が監査されます。
オブジェクト	emp 表に対する ALTER TABLE 文など、特定のオブジェクトに対する特定の文を監査します。

AUDIT 文を使用して文オプションおよび権限オプションを設定するには、AUDIT SYSTEM 権限が必要です。AUDIT 文を使用してオブジェクト監査オプションを設定するには、監査対象のオブジェクトを所有しているか、または AUDIT ANY 権限を持っている必要があります。

文監査オプションおよび権限監査オプションを設定する監査文では、BY 句によってユーザーまたはアプリケーション・プロキシのリストを指定し、文監査オプションと権限監査オプションの有効範囲を限定できます。

監査オプションを設定するときは、次の監査条件も併せて指定できます。

- BY SESSION/BY ACCESS

  BY SESSION を指定すると、同じセッション内で発行された同じタイプの SQL 文すべてに対して単一のレコードが書き込まれます。BY ACCESS を指定すると、アクセスごとに 1 つのレコードが書き込まれます。

注意： 監査証跡にオペレーティング・システム・ファイルを使用する場合（AUDIT_FILE_DEST=OS）は、BY SESSION を指定しても、監査証跡に複数のレコードが書き込まれる場合があります。これは、Oracle がオペレーティング・システム・ファイルに書き込んでいる間は、そのファイルを読み込み、すでにアクションの監査エントリが書き込まれていることを検出できないためです。

- WHENEVER SUCCESSFUL/WHENEVER NOT SUCCESSFUL

  WHENEVER SUCCESSFUL を指定すると、文が成功した場合のみ監査レコードが記録されます。WHENEVER NOT SUCCESSFUL を指定すると、文が失敗した場合またはエラーが発生した場合のみ監査レコードが記録されます。

監査オプションの各選択肢が表す意味と AUDIT 文の句の指定については、後述します。

新しいデータベース・セッションが作成されると、このセッションはデータ・ディクショナリから監査オプションを取得します。これらの監査オプションは、データベースに接続している間は有効です。新しいシステム監査オプションまたはオブジェクト監査オプションを設定すると、それ以降のデータベース・セッションから新たに設定されたオプションが使用されます。既存のセッションでは、セッションの作成時に取得された監査オプションが引き続き使用されます。

注意： AUDIT 文を実行しても監査オプションが設定されるだけであり、監査機能全体が使用可能になるわけではありません。監査機能を使用可能にして、現在設定されている監査オプションに基づいて監査レコードを生成するかどうかを制御するには、26-7 ページの「監査の使用可能および使用禁止」の説明に従って、初期化パラメータ AUDIT_TRAIL を設定します。

関連項目： AUDIT 文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

文監査の指定

AUDIT 文と NOAUDIT 文に指定できる有効な文監査オプションの詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

ここでは、2 つの特別な文監査のケースについて説明します。

接続と切断の監査 SESSION 文オプションは、特定のタイプの文が発行されたときに監査レコードを生成するのではない点で他とは異なります。このオプションは、インスタンスへの接続によって作成されたセッションごとに、単一の監査レコードを生成します。監査レコードは、接続時に監査証跡に挿入され、切断時に更新されます。接続時刻、切断時刻、処理された論理 I/O や物理 I/O などのセッションに関する累積情報が、そのセッションに対応する単一の監査レコード内に格納されます。

ユーザーおよび正常終了 / 失敗の状態に関係なく、データベースとのすべての接続および切断について、SESSION（デフォルトであり、このオプションの唯一の値）ごとに監査するには、次の文を実行します。

```
AUDIT SESSION;
```

次の例に示すように、このオプションをユーザーごとに選択的に設定することもできます。

```
AUDIT SESSION
BY scott, lori;
```

オブジェクトが存在しないために失敗した文の監査 NOT EXISTS 文オプションは、ターゲット・オブジェクトが存在しないために失敗した SQL 文すべての監査を指定します。

権限監査の指定

権限監査オプションは、対応するシステム権限と正確に一致します。たとえば、DELETE ANY TABLE 権限の使用を監査するためのオプションは、DELETE ANY TABLE です。このオプションを使用可能にするには、次のような文を使用します。

```
AUDIT DELETE ANY TABLE
    BY ACCESS
    WHENEVER NOT SUCCESSFUL;
```

Oracle のシステム権限の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

DELETE ANY TABLE システム権限の正常な使用および失敗した使用をすべて監査するには、次の文を実行します。

```
AUDIT DELETE ANY TABLE;
```

すべての表に対する失敗した SELECT、INSERT および DELETE 文、および EXECUTE PROCEDURE システム権限の失敗した使用を、全データベース・ユーザーについて個々の監査文別にすべて監査するには、次の文を実行します。

```
AUDIT SELECT TABLE, INSERT TABLE, DELETE TABLE, EXECUTE PROCEDURE
      BY ACCESS
      WHENEVER NOT SUCCESSFUL;
```

文監査オプションまたは権限監査オプションを設定するには、AUDIT SYSTEM システム権限が必要です。通常、この権限を付与されているユーザーはセキュリティ管理者のみです。

オブジェクト監査の指定

有効なオブジェクト監査オプションと、各オプションが使用できるスキーマ・オブジェクトのタイプの詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

ユーザーは、自分のスキーマ内にあるオブジェクトのオブジェクト監査オプションを設定できます。他のユーザーのスキーマ内にあるオブジェクトのオブジェクト監査オプションを設定したり、デフォルトのオブジェクト監査オプションを設定するには、AUDIT ANY システム権限が必要です。通常、AUDIT ANY 権限を付与されているユーザーはセキュリティ管理者のみです。

scott.emp 表に対する正常終了および失敗したすべての DELETE 文をセッション別（BY SESSION、デフォルト）に監査するには、次の文を実行します。

```
AUDIT DELETE ON scott.emp;
```

ユーザー jward が所有する dept 表に対する正常終了したすべての SELECT、INSERT、DELETE の文をアクセス別（BY ACCESS）に監査するには、次の文を実行します。

```
AUDIT SELECT, INSERT, DELETE
     ON jward.dept
     BY ACCESS
     WHENEVER SUCCESSFUL;
```

失敗したすべての SELECT 文を監査するためのデフォルトのオブジェクト監査オプションをセッション別（BY SESSION、デフォルト）に設定するには、次の文を指定します。

```
AUDIT SELECT
     ON DEFAULT
     WHENEVER NOT SUCCESSFUL;
```

複数層環境での監査

複数層環境では、Oracle はすべての層を通じてクライアントの識別情報を保持します。これにより、クライアントのかわりに実行されたアクションの監査が可能になります。そのためには、AUDIT 文で BY *proxy* 句を指定します。

この句には、いくつかのオプションがあります。次の操作が可能です。

- 代理として指定したプロキシによって発行された SQL 文の監査
- 指定したユーザーのかわりに実行された文の監査
- 任意のユーザーのかわりに実行されたすべての文の監査

次の例では、プロキシ・アプリケーション・サーバー appserve によってクライアント jackson のかわりに発行された SELECT TABLE 文が監査されます。

```
AUDIT SELECT TABLE
    BY appserve ON BEHALF OF jackson;
```

関連項目： プロキシと複数層アプリケーションの詳細は、『Oracle9*i* データベース概要』および『Oracle9*i* アプリケーション開発者ガイドー基礎編』を参照してください。

監査オプションを使用禁止にする方法

Oracle の各種監査オプションを使用禁止にするには、NOAUDIT 文を使用します。このオプションは、文監査オプション、権限監査オプションおよびオブジェクト監査オプションの設定を解除するために使用します。文監査オプションおよび権限監査オプションを設定する NOAUDIT 文では、BY *user* または BY *proxy* オプションによってユーザーのリストを指定し、各監査オプションの適用範囲を限定できます。

NOAUDIT 文では、WHENEVER 句を使用して監査オプションを選択的に使用禁止にできます。この句を指定しなければ、正常終了と失敗のどちらの場合でも監査オプションは完全に使用禁止となります。

BY SESSION/BY ACCESS オプションのペアは、NOAUDIT 文ではサポートされていません。このため、監査オプションの設定方法に関係なく、監査オプションの設定は適切な NOAUDIT 文によって解除されます。

注意： NOAUDIT 文を実行しても監査オプションの設定が解除されるだけであり、監査機能全体が使用禁止になるわけではありません。監査機能を使用禁止にして、Oracle による監査レコードの生成を停止するには、26-7 ページの「監査の使用可能および使用禁止」の説明に従って、データベースの初期化パラメータ・ファイルに初期化パラメータ AUDIT_TRAIL を設定します。

関連項目： NOAUDIT 文の構文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

文監査および権限監査を使用禁止にする方法

対応する監査オプションを使用禁止にするには、次の文を実行します。

```
NOAUDIT session;
NOAUDIT session BY scott, lori;
NOAUDIT DELETE ANY TABLE;
NOAUDIT SELECT TABLE, INSERT TABLE, DELETE TABLE,
    EXECUTE PROCEDURE;
```

文監査オプションをすべて使用禁止にするには、次の文を実行します。

```
NOAUDIT ALL;
```

権限監査オプションをすべて使用禁止にするには、次の文を実行します。

```
NOAUDIT ALL PRIVILEGES;
```

文監査オプションまたは権限監査オプションを使用禁止にするには、AUDIT SYSTEM システム権限が必要です。

オブジェクト監査を使用禁止にする方法

対応する監査オプションを使用禁止にするには、次の文を実行します。

```
NOAUDIT DELETE
   ON emp;
NOAUDIT SELECT, INSERT, DELETE
   ON jward.dept;
```

また、emp 表に対するオブジェクト監査オプションをすべて使用禁止にするには、次の文を実行します。

```
NOAUDIT ALL
   ON emp;
```

デフォルトのオブジェクト監査オプションをすべて使用禁止にするには、次の文を実行します。

```
NOAUDIT ALL
   ON DEFAULT;
```

この NOAUDIT 文を発行する前に作成されたすべてのスキーマ・オブジェクトは、作成後に NOAUDIT 文を使用して明示的に設定が変更されていないかぎり、作成時に指定されたデフォルトのオブジェクト監査オプションが引き続き使用されます。

特定のオブジェクトのオブジェクト監査オプションを使用禁止にするには、そのスキーマ・オブジェクトの所有者であることが必要です。他のユーザーのスキーマ内にあるオブジェクトのオブジェクト監査オプションや、デフォルトのオブジェクト監査オプションを使用禁止にするには、AUDIT ANY システム権限が必要です。オブジェクトのオブジェクト監査オプションを使用禁止にする権限を持っているユーザーは、他のユーザーによって設定されたオプションを変更できます。

監査証跡の増加とサイズの制御

監査証跡がいっぱいになり、これ以上監査レコードを挿入できなくなった場合は、監査証跡を削除しないかぎり、監査対象の文を正常に実行できません。監査対象の文を発行するすべてのユーザーに対して警告が表示されます。このため、セキュリティ管理者は、監査証跡の増加とサイズを制御する必要があります。

監査を使用可能にして監査レコードが生成されているときは、次の 2 つの要因に従って監査証跡が増加します。

- 使用可能になっている監査オプションの数
- 監査対象の文の実行頻度

監査証跡の増加を制御するには、次の方法を使用します。

- データベース監査を使用可能および使用禁止にします。使用可能にすると、監査レコードが生成されて監査証跡に格納されます。使用禁止にすると、監査レコードは生成されません。
- 使用可能にする監査オプションを選択的に絞り込みます。選択的な監査を実施すると、不要な監査情報が生成されず、監査証跡に格納されません。
- オブジェクト監査を実行する許可を厳密に制御します。これには、2 種類の方法があります。
  - セキュリティ管理者がすべてのオブジェクトを所有し、AUDIT ANY システム権限を他のユーザーには付与しません。また、すべてのスキーマ・オブジェクトを、対応するユーザーが CREATE SESSION 権限を有していないスキーマに所属させることもできます。
  - すべてのオブジェクトを、実際のデータベース・ユーザーに対応していない（つまり、対応するユーザーに CREATE SESSION 権限が付与されていない）スキーマに格納しておき、セキュリティ管理者のみに AUDIT ANY システム権限を付与します。

  どちらの方法でも、セキュリティ管理者がオブジェクト監査を完全に制御できます。

データベース監査証跡（SYS.AUD$ 表）の最大サイズは、この表が格納されている SYSTEM 表領域のデフォルトの記憶域パラメータによって決まります。SYS.AUD$ は監査証跡の増加とサイズを制御する手段であるため、他の表領域に移動しないでください。ただし、SYS.AUD$ の記憶域パラメータは変更できます。

注意： SYS.AUD$ 表を SYSTEM 表領域の外へ移動する操作はサポートされていません。これは、Oracle コード内で、SYS.AUD$ など、アップグレードやバックアップ / リカバリの際に問題を引き起こす可能性のあるデータ・ディクショナリ表を暗黙的に想定しているためです。

関連項目： 監査レコードをオペレーティング・システムの監査証跡に書き込んでいる場合の、オペレーティング・システム監査証跡の管理方法の詳細は、使用しているオペレーティング・システム固有の Oracle マニュアルを参照してください。

監査証跡から監査レコードを削除する方法

セキュリティ管理者は、ある期間、監査機能を使用可能にした後、レコードをデータベース監査証跡から削除できます。これにより、監査証跡の領域が解放され、監査証跡の管理が容易になります。

たとえば、監査証跡からすべての監査レコードを削除するには、次の文を実行します。

```
DELETE FROM SYS.AUD$;
```

また、emp 表の監査の結果として生成された監査証跡からすべての監査レコードを削除するには、次の文を実行します。

```
DELETE FROM SYS.AUD$
     WHERE obj$name='EMP';
```

履歴の目的で監査証跡情報をアーカイブする必要がある場合、セキュリティ管理者は関連するレコードを通常のデータベース表にコピーしたり（INSERT INTO *table* SELECT ... FROM SYS.AUD$ ... などを使用）、監査証跡表をオペレーティング・システム・ファイルにエクスポートできます。

DELETE ANY TABLE 権限を持っている SYS ユーザー、または SYS によって SYS.AUD$ の DELETE 権限を付与されているユーザーのみがデータベース監査証跡からレコードを削除できます。

注意： 監査証跡が完全にいっぱいになっている場合で、接続が監査されているとき（つまり、SESSION オプションを設定している場合）は、その接続に対応する監査レコードを監査証跡に挿入できないため、通常のユーザーはデータベースに接続できません。この場合、セキュリティ管理者は必ず SYS で接続し（SYS による操作は監査されないため）、監査証跡で使用可能な領域を作成する必要があります。

関連項目： 表のエクスポートの詳細は、『Oracle9*i* データベース・ユーティリティ』を参照してください。

監査証跡のサイズの縮小

データベース表の場合と同様に、データベース監査証跡からレコードが削除されても、この表に割り当てられているエクステントはそのまま存在します。

データベース監査証跡に多数のエクステントが割り当てられていても、そのうちの大半が使用されていなければ、次の手順に従って、データベース監査証跡に割り当てられている領域を縮小できます。

1. 監査証跡の現在の情報を保存する場合は、その情報を別のデータベース表にコピーするか、または EXPORT ユーティリティを使用してエクスポートします。
2. 管理者権限を持つユーザーとして接続します。

3. TRUNCATE 文を使用して、SYS.AUD$ を切り捨てます。
4. 手順 1 で作成したアーカイブ済みの監査証跡レコードを再ロードします。

SYS.AUD$ の新しいバージョンには、現在の監査証跡レコードを記録するために必要な数だけエクステントが割り当てられます。

注意： 直接変更できる SYS オブジェクトは、SYS.AUD$ のみです。

監査証跡の保護

疑わしいデータベース・アクティビティを監査する場合は、監査証跡のレコードの整合性を保護することによって、監査情報が正確かつ完全であることを保証してください。

データベース監査証跡を不正な削除処理から保護するために、DELETE ANY TABLE システム権限はセキュリティ管理者に対してのみ付与します。

データベース監査証跡に対する変更を監査するには、次の文を使用します。

```
AUDIT INSERT, UPDATE, DELETE
    ON sys.aud$
    BY ACCESS;
```

SYS.AUD$ 表自体を不正な使用から保護している場合、SYS.AUD$ 表に対してオブジェクト監査オプションを設定した結果として生成された監査レコードを削除できるのは、管理者権限で接続しているユーザーのみです。

ファイングレイン監査

これまでに説明された監査方法では、監査証跡に固定セットのファクトが記録されていました。また、監査オプションを設定できるのは、オブジェクトのアクセスまたは権限を監視する場合のみでした。環境や問合せ結果に関する特定の情報を取得する方法や、監査の誤りを最小限に抑えるために監査条件を指定するメカニズムは、前述の方法ではサポートされていません。Oracle では、これらの目的を実現するために、ファイングレイン監査が用意されています。

ファイングレイン監査を使用すると、内容に基づいてデータのアクセスを監視できます。たとえば、中央税務当局では、従業員の詮索から保護するために、所得申告へのアクセスを追跡する必要があります。そのためには、特定のユーザーが特定の表に対して使用した SELECT 権限だけでなく、どのデータがアクセスされたかを判断できるほど詳細な情報が必要となります。ファイングレイン監査は、このような機能を提供します。

一般に、ファイングレイン監査の方針は、選択的監査の条件と同様に、表オブジェクトに対する単純なユーザー定義の SQL 述語に基づきます。フェッチによって返される行について方針条件が満たされた場合は、必ず問合せが監査されます。その後、Oracle は、自律型トラ

ンザクションを使用してユーザー定義のイベント・ハンドラを実行し、イベントを処理します。

ユーザー・アプリケーションにファイングレイン監査を実装するには、DBMS_FGA パッケージまたはデータベース・トリガーを使用します。

関連項目： ファイングレイン監査の詳細は、『Oracle9*i* アプリケーション開発者ガイド－基礎編』を参照してください。

データベース監査証跡情報の表示

データベース監査証跡（SYS.AUD$）は、各 Oracle データベースのデータ・ディクショナリ内にある単一の表です。この表に含まれている監査情報を意味のある形式で表示するために、いくつかの事前定義のビューが利用できます。これらのビューは管理者が作成します。監査を使用禁止にする場合は、後でそれらのビューを削除できます。

監査証跡ビューの作成

CATALOG.SQL および CATAUDIT.SQL スクリプトによって、次のビュー（STMT_AUDIT_OPTION_MAP を除く）が作成されます。

ビュー	説明
STMT_AUDIT_OPTION_MAP	監査オプション・タイプ・コードの情報が含まれます。CREATE DATABASE 時に SQL.BSQ スクリプトによって作成されます。
AUDIT_ACTIONS	監査証跡のアクション・タイプ・コードの記述が含まれます。
ALL_DEF_AUDIT_OPTS	オブジェクトの作成時に適用されるデフォルトのオブジェクト監査オプションが含まれます。
DBA_STMT_AUDIT_OPTS	システム全体の現行のシステム監査オプションがユーザー別に表示されます。
DBA_PRIV_AUDIT_OPTS	システム全体で監査対象となっている現行のシステム権限がユーザー別に表示されます。
DBA_OBJ_AUDIT_OPTS USER_OBJ_AUDIT_OPTS	すべてのオブジェクトの監査オプションが表示されます。USER ビューには、現行ユーザーが所有するすべてのオブジェクトの監査オプションが表示されます。
DBA_AUDIT_TRAIL USER_AUDIT_TRAIL	すべての監査証跡エントリがリストされます。USER ビューには、現行ユーザーに関連する監査証跡エントリが表示されます。

ビュー	説明
DBA_AUDIT_OBJECT USER_AUDIT_OBJECT	システム内にあるすべてのオブジェクトの監査証跡レコードが含まれます。USER ビューには、現行ユーザーがアクセス可能なオブジェクトに関する文の監査証跡レコードがリストされます。
DBA_AUDIT_SESSION USER_AUDIT_SESSION	CONNECT および DISCONNECT に関する監査証跡レコードがすべてリストされます。USER ビューには、現行ユーザーの接続および切断に関する監査証跡レコードがすべてリストされます。
DBA_AUDIT_STATEMENT USER_AUDIT_STATEMENT	データベース全体の GRANT、REVOKE、AUDIT、NOAUDIT および ALTER SYSTEM 文に関する監査証跡レコードがリストされます。USER ビューでは、ユーザーが発行したそれぞれの文に関する監査証跡レコードがリストされます。
DBA_AUDIT_EXISTS	BY AUDIT NOT EXISTS によって生成された監査証跡エントリがリストされます。
ファイングレイン監査では、次のビューが使用されます。	
DBA_AUDIT_POLICIES	システム上にあるすべての監査方針が表示されます。
DBA_FGA_AUDIT_TRAIL	値ベース監査の監査証跡レコードがリストされます。

関連項目： Oracle が提供する事前定義ビューの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

監査証跡ビューの削除

監査機能を使用禁止にしており、監査証跡ビューが不要な場合は、SYS としてデータベースへ接続して、スクリプト・ファイル CATNOAUD.SQL を実行することにより、ビューを削除できます。CATNOAUD.SQL スクリプトの名前と位置は、オペレーティング・システムによって異なります。

監査証跡ビューを使用した疑わしいアクティビティの調査

ここでは、監査証跡情報の検討方法および解釈方法の具体例を説明します。次の状況について考えます。

次のような疑わしいアクティビティについて、データベースを監査するとします。

- データベース・ユーザーのパスワード、表領域の設定および割当て制限が許可なく変更されている。
- おそらく排他的に表ロックを取得しているユーザーが原因で、デッドロックが頻繁に発生している。
- scott のスキーマ内にある emp 表から行が勝手に削除されている。

これらの不正なアクションのいくつかは、ユーザー jward と swilliams によって行われた疑いがあります。

調査を可能にするために、次の文を順序どおりに発行します。

```
AUDIT ALTER, INDEX, RENAME ON DEFAULT
    BY SESSION;
CREATE VIEW scott.employee AS SELECT * FROM scott.emp;
AUDIT SESSION BY jward, swilliams;
AUDIT ALTER USER;
AUDIT LOCK TABLE
    BY ACCESS
    WHENEVER SUCCESSFUL;
AUDIT DELETE ON scott.emp
    BY ACCESS
    WHENEVER SUCCESSFUL;
```

その後、ユーザー jward によって次の文が発行されました。

```
ALTER USER tsmith QUOTA 0 ON users;
DROP USER djones;
```

その後、ユーザー swilliams によって次の文が発行されました。

```
LOCK TABLE scott.emp IN EXCLUSIVE MODE;
DELETE FROM scott.emp WHERE mgr = 7698;
ALTER TABLE scott.emp ALLOCATE EXTENT (SIZE 100K);
CREATE INDEX scott.ename_index ON scott.emp (ename);
CREATE PROCEDURE scott.fire_employee (empid NUMBER) AS
  BEGIN
    DELETE FROM scott.emp WHERE empno = empid;
  END;
/

EXECUTE scott.fire_employee(7902);
```

次の項では、データ・ディクショナリ内の監査証跡ビューを使用して表示できる情報のうち、この調査に関連するものを示します。

- アクティブな文監査オプションのリスト
- アクティブな権限監査オプションのリスト
- 特定のオブジェクトに対するアクティブなオブジェクト監査オプションのリスト
- デフォルトのオブジェクト監査オプションのリスト
- 監査レコードのリスト
- AUDIT SESSION オプションの監査レコードのリスト

アクティブな文監査オプションのリスト

次の問合せを実行すると、設定されている文監査オプションがすべて表示されます。

```
SELECT * FROM DBA_STMT_AUDIT_OPTS;

USER_NAME                AUDIT_OPTION         SUCCESS         FAILURE
--------------------     -------------------  ----------      ---------
JWARD                    SESSION              BY SESSION      BY SESSION
SWILLIAMS                SESSION              BY SESSION      BY SESSION
                         LOCK TABLE           BY ACCESS       NOT SET
```

このビューでは、文監査オプションの設定、たとえば正常終了と失敗のどちら（あるいはその両方）に設定されているか、また BY SESSION と BY ACCESS のどちらに設定されているかといったことを確認できます。

アクティブな権限監査オプションのリスト

次の問合せを実行すると、設定されている権限監査オプションがすべて表示されます。

```
SELECT * FROM DBA_PRIV_AUDIT_OPTS;

USER_NAME            PRIVILEGE            SUCCESS      FAILURE
------------------- -------------------- ---------   ----------
ALTER USER           BY SESSION           BY SESSION
```

特定のオブジェクトに対するアクティブなオブジェクト監査オプションのリスト

次の問合せを実行すると、名前が emp という文字で始まり、かつ scott のスキーマ内に格納されているオブジェクトについて、監査オプションの設定がすべて表示されます。

```
SELECT * FROM DBA_OBJ_AUDIT_OPTS
    WHERE OWNER = 'SCOTT' AND OBJECT_NAME LIKE 'EMP%';

OWNER OBJECT_NAME OBJECT_TY ALT AUD COM DEL GRA IND INS LOC ...
----- ----------- --------- --- --- --- --- --- --- --- --- ...
SCOTT EMP         TABLE     S/S -/- -/- A/- -/- S/S -/- -/- ...
SCOTT EMPLOYEE    VIEW      -/- -/- -/- A/- -/- S/S -/- -/- ...
```

このビューでは、指定したオブジェクトに対するすべての監査オプションの情報が表示されます。このビューの情報は、次のように解釈します。

- 文字 - は、監査オプションが何も設定されていないことを示します。
- 文字 S は、監査オプションが BY SESSION に設定されていることを示します。
- 文字 A は、監査オプションが BY ACCESS に設定されていることを示します。
- 各監査オプションには WHENEVER SUCCESSFUL と WHENEVER NOT SUCCESSFUL の 2 つの設定があり、/ で区切られています。たとえば、scott.emp に対する DELETE 監査オプションは、正常終了した削除文に対して BY ACCESS が設定されています。失敗した削除文に対しては何も設定されていません。

デフォルトのオブジェクト監査オプションのリスト

次の問合せは、デフォルトのオブジェクト監査オプションをすべて返します。

```
SELECT * FROM ALL_DEF_AUDIT_OPTS;

ALT AUD COM DEL GRA IND INS LOC REN SEL UPD REF EXE
--- --- --- --- --- --- --- --- --- --- --- --- ---
S/S -/- -/- -/- -/- S/S -/- -/- S/S -/- -/- -/- -/-
```

このビューでは、USER_OBJ_AUDIT_OPTS と DBA_OBJ_AUDIT_OPTS の各ビューによく似た情報が表示されます（先の例を参照）。

監査レコードのリスト

次の問合せを実行すると、文監査オプションとオブジェクト監査オプションで生成された監査レコードがリストされます。

```
    SELECT * FROM DBA_AUDIT_OBJECT;
```

AUDIT SESSION オプションの監査レコードのリスト

次の問合せを実行すると、AUDIT SESSION 文監査オプションに対応する監査情報がリストされます。

```
SELECT USERNAME, LOGOFF_TIME, LOGOFF_LREAD, LOGOFF_PREAD,
    LOGOFF_LWRITE, LOGOFF_DLOCK
    FROM DBA_AUDIT_SESSION;

USERNAME   LOGOFF_TI LOGOFF_LRE LOGOFF_PRE LOGOFF_LWR LOGOFF_DLO
---------- --------- ---------- ---------- ---------- ----------
JWARD      02-AUG-91         53          2         24          0
SWILLIAMS  02-AUG-91       3337        256        630          0
```

第 V 部

データベース・リソースの管理

第 V 部では、データベース・リソースの管理について説明します。第 V 部の構成は、次のとおりです。

27

Database Resource Manager の使用

Oracle では、Database Resource Manager によるデータベース・リソースの管理機能が提供されます。この章では、その使用方法について説明します。

この章の内容は、次のとおりです。

- Database Resource Manager の概要
- Database Resource Manager の管理
- 単純なリソース・プランの作成
- 複雑なリソース・プランの作成
- リソース・コンシューマ・グループの管理
- Database Resource Manager を使用可能にする方法
- 各種の方法を組み合せた Database Resource Manager の例
- Database Resource Manager の監視とチューニング
- Database Resource Manager 情報の表示

注意： この章では、Oracle が提供する DBMS_RESOURCE_MANAGER および DBMS_RESOURCE_MANAGER_PRIVS パッケージを使用して、Database Resource Manager を管理する方法について説明します。Oracle Enterprise Manager（OEM）を使用すれば、一層容易に Database Resource Manager を管理できます。OEM は、Database Resource Manager を管理するために、操作性に優れたグラフィカル・インタフェースを備えています。

詳細は、OEM のドキュメント・セットを参照してください。

Database Resource Manager の概要

Database Resource Manager の主な目的は、Oracle データベース・サーバーがリソース管理の決定を厳密に制御することによって、非効率的なオペレーティング・システム管理から生じる問題を回避することです。

この項の内容は、次のとおりです。

- Database Resource Manager が対処する問題
- Database Resource Manager によるこれらの問題の対処方法
- Database Resource Manager の要素
- リソース・プランの理解

Database Resource Manager が対処する問題

データベース・リソースの割当てがオペレーティング・システムによって決定される場合は、次のような問題が生じることがあります。

- 過剰なオーバーヘッドの発生

  過剰なオーバーヘッドの原因は、サーバー・プロセスの数が多いときに、Oracle サーバー・プロセス間でオペレーティング・システムのコンテキストが切り替わることです。

- 非効率的なスケジューリング

  オペレーティング・システムは、Oracle データベース・サーバーがラッチを保持している間にデータベース・サーバーのスケジュールを解除しますが、これは非効率的です。

- 不適当なリソースの割当て

  オペレーティング・システムはタスク間の優先順位付けができないため、すべてのアクティブなプロセスの間で均等にリソースを分配します。

- パラレル実行サーバーやアクティブ・セッションなど、データベース固有のリソースを管理できない

Database Resource Manager によるこれらの問題の対処方法

Database Resource Manager は、マシン・リソースの割当て方法をデータベースで厳密に制御することによって、これらの問題を克服します。

Database Resource Manager を使用すると、次のことが可能になります。

- システムにかかる負荷やユーザー数に関係なく、特定のユーザーの処理リソースが最小量になることを保証します。
- 様々なユーザーおよびアプリケーションに、比率を指定して CPU タイムを割り当てて、使用可能な処理リソースを分配します。データ・ウェアハウスの場合は、バッチ・ジョブよりもリレーショナル・オンライン分析処理（ROLAP）アプリケーションへの割当て率を高くすることができます。
- ユーザー・グループのメンバーが実行する処理の並列度を制限します。
- **アクティブ・セッション・プール**を作成します。このプールは、ユーザー・グループ内で同時にアクティブにできる最大数のユーザー・セッションで構成されています。最大数を超える追加セッションは実行待ちのキューに送られますが、キューで待機しているジョブが終了するまでのタイムアウト周期を指定できます。
- 管理者が定義した基準に基づいて、グループ間でユーザーの**自動切替え**を行います。特定のユーザー・グループのメンバーが、指定された時間より長時間実行されるセッションを作成した場合、そのセッションをリソース要件が異なる他のユーザー・グループに自動的に切り替えることができます。
- ある操作の実行時間の見積りが事前定義の制限を超える場合、その操作は実行されません。
- **UNDO プール**を作成します。このプールは、ユーザー・グループで消費可能な量の UNDO 領域から構成されます。
- 特定のリソース割当て方法を使用するように、インスタンスを構成します。インスタンスを停止して再起動しなくても、日中の設定から夜間の設定へというように、リソース割当て方法を動的に変更できます。

Database Resource Manager の要素

次に、Oracle のデータベース・リソース管理の要素を示します。各要素は、Database Resource Manager パッケージを通じて定義します。

要素	説明
リソース・コンシューマ・グループ	リソースの処理要件に基づいてグループ化されたユーザー・セッション。
リソース・プラン	リソース・コンシューマ・グループへのリソースの割当て方法を指定するディレクティブを含みます。
リソース割当て方法	Database Resource Manager によって、リソース・コンシューマ・グループとリソース・プランに使用される特定のリソースを割り当てるための方法 / 方針。使用可能な割当て方法は Oracle が提供しますが、どの方法を使用するかは管理者が決めます。
リソース・プラン・ディレクティブ	管理者が、リソース・コンシューマ・グループを特定のプランに対応付け、リソース・コンシューマ・グループ間でリソースを割り当てるために使用するディレクティブ。

これらの要素の作成方法および使用方法については、後述します。

リソース・プランの理解

ここでは、リソース・プランの概要について簡単に説明します。単純なリソース・プランの例を図示します。より複雑なプランについては、Database Resource Manager の要素の作成およびメンテナンス方法を説明した後（27-26 ページの「各種の方法を組み合せた Database Resource Manager の例」）に示します。

リソース・プランでは、そのプランに属するリソース・コンシューマ・グループを指定し、これらのグループ間でのリソース割当て方法を示すディレクティブを含めます。`DBMS_RESOURCE_MANAGER` パッケージを使用して、Database Resource Manager の要素（リソース・プラン、リソース・コンシューマ・グループおよびリソース・プラン・ディレクティブ）を作成およびメンテナンスします。プラン情報は、データ・ディクショナリ内の表に格納されます。プラン・データの表示には、複数のビューを使用できます。

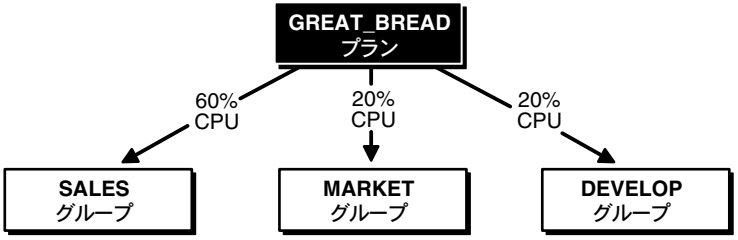
関連項目： Database Resource Manager の詳細は、『Oracle9*i* データベース概要』を参照してください。

27-33 ページ「Database Resource Manager 情報の表示」

単一レベルのリソース・プラン

図 27-1 は、単一レベルのプランを示しています。このプランでは、複数のリソース・コンシューマ・グループ間でリソースを割り当てています。Great Bread Company は、3 つのリソース・コンシューマ・グループ間で CPU リソースを割り当てるプラン great_bread を持っています。具体的には、sales に CPU タイムの 60%、market に 20%、develop に残りの 20% が割り当てられています。

図 27-1　単純なリソース管理プラン

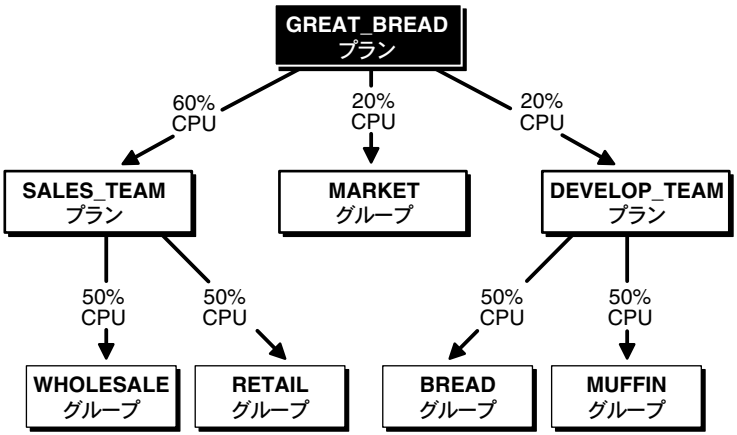


Oracle には、単純なリソース・プランを迅速に作成できるプロシージャ（CREATE_SIMPLE_PLAN）が用意されています。このプロシージャについては、27-11 ページの「単純なリソース・プランの作成」を参照してください。

複数レベルのリソース・プラン

プランにはリソース・コンシューマ・グループのみでなく、他のプランも含めることができます。これを**サブプラン**と呼びます。Great Bread Company では、CPU リソースを図 27-2 のように分割することもできます。

図 27-2　サブプランを含む複数レベルのプラン



このケースでは、great_bread プランは前述の例と同様にコンシューマ・グループ market に CPU リソースを割り当てていますが、その他にはサブプラン sales_team および develop_team に CPU リソースを割り当て、この 2 つのサブプランから各コンシューマ・グループに割り当てています。図 27-2 は、**トップレベルのプラン**（great_bread）とそのすべての子を含む**プラン・スキーマ**を示しています。

サブプランまたはコンシューマ・グループが複数の親（所有するプラン）を持つことはできますが、プラン・スキーマ内でループすることはできません。たとえば、Great Bread Company に夜間のプランと日中のプランがある場合、サブプランは複数の親を持つことになります。夜間のプランと日中のプランにはどちらも sales サブプランがメンバーとして含まれますが、各インスタンスの CPU リソース割当ては異なります。

注意： 前述のプランには、後で説明するように OTHER_GROUPS のプラン・ディレクティブも含める必要があります。ただし、図を単純化するために、このプラン・ディレクティブは省略されています。

リソース・コンシューマ・グループ

リソース・コンシューマ・グループとは、処理要件に基づいてグループ化されたユーザー（セッション）のグループです。次に説明するリソース・プラン・ディレクティブによって、プラン・スキーマ内のコンシューマ・グループおよびサブプラン間でのリソースの割当て方法を指定します。

リソース・プラン・ディレクティブ

リソース・コンシューマ・グループへのリソースの割当て方法は、リソース割当てディレクティブで指定します。Database Resource Manager には、いくつかのリソース割当て方法が用意されています。

CPU 方法 この方法では、コンシューマ・グループまたはサブプラン間での CPU リソースの割当て方法を指定します。複数レベル（最大 8 レベル）の CPU リソース割当てにより、プラン・スキーマ内で CPU 使用の優先順位を設定できます。レベル 2 は、レベル 1 でそのすべてのリソースを使用できなくなった場合のみリソースを取得します。複数レベルを使用すると、優先順位を設定できるだけでなく、すべての主リソースと残りのリソースの使用方法を明示的に指定できます。

キューイングを備えたアクティブ・セッション・プール コンシューマ・グループ内で同時にアクティブにできるセッションの最大数を制御できます。この最大数によって、アクティブ・セッション・プールが構成されます。プールがいっぱいのためにセッションを初期化できない場合は、セッションがキューに送られます。アクティブなセッションが終了すると、キュー内の最初のセッションが実行のためにスケジュールされます。実行キュー内で実行されるのを待機しているジョブがタイムアウトするまでのタイムアウト周期も指定できます。ジョブがタイムアウトすると、エラーが発生して終了します。

パラレル実行セッションは全体で 1 つのアクティブ・セッションとして数えられます。

並列度制限 並列度制限を指定すると、コンシューマ・グループ内の任意の操作における最大の並列度を制御できます。

コンシューマ・グループの自動切替え この方法では、他のコンシューマ・グループにセッションを自動的に切り替えるための基準を指定して、リソースを制御できます。コンシューマ・グループの切替えに使用できる基準は、次のとおりです。

- 切替えグループ。他の基準（以降の基準）が満たされた場合に、このセッションを切り替える先のコンシューマ・グループを指定します。
- 切替え時間。他のコンシューマ・グループに切り替えるまでにセッションが実行できる時間の長さを指定します。
- 見積りの使用。Oracle が独自に見積った操作の実行時間を使用するかどうかを指定します。

Database Resource Manager は、セッションがアクティブである時間が切替え時間で設定されている秒数を超えたときに、実行中のセッションを切替えグループに切り替えます。ここでのアクティブは、セッションが実行中でリソースを消費していることを意味します。ユーザー入力待ちのためにアイドルしているときや、CPU サイクル待ちは含まれません。切替え後のグループのアクティブ・セッション・プールがいっぱいの場合でも、セッションは引き続き実行することを許可されます。このような状況では、コンシューマ・グループは、アクティブ・セッション・プールで指定された数より多くのセッションを実行できます。セッションの処理が終了し、アイドルになると、元のグループに切り替えられます。

見積りの使用を TRUE に設定した場合、Database Resource Manager は、操作が完了するまでの時間の予測値を使用します。Oracle の予測値が切替え時間で指定された値よりも長い場合は、実行を開始する前に、セッションが切り替えられます。このパラメータを設定しない場合は、通常どおり操作が始まり、他の切替え基準が満たされた場合のみ、グループが切り替わります。

実行時間制限 ある操作に許可される最大の実行時間を指定できます。ある操作について Oracle が見積った実行時間が、指定された最大実行時間より長い場合、その操作はエラーを出力して終了します。このエラーはトラップ可能であり、操作は再スケジュールできます。

UNDO プール UNDO プールは、コンシューマ・グループごとに指定できます。UNDO プールによって、コンシューマ・グループが生成できる UNDO の合計量が制御されます。コンシューマ・グループが生成する UNDO の合計量がその UNDO 制限を超えると、REDO を生成している現行のデータ操作言語（DML）文が終了します。コンシューマ・グループの他のメンバーは、UNDO 領域がプールから解放されるまで、新たにデータ操作を実行できません。

> **関連項目：** Database Resource Manager の概念の詳細は、『Oracle9*i* データベース概要』を参照してください。

Database Resource Manager の管理

Database Resource Manager を管理するには、システム権限 ADMINISTER_RESOURCE_MANAGER が必要です。通常、データベース管理者（DBA）は、DBA ロール（またはそれと等価のロール）の一部として、ADMIN オプション付きでこの権限を持っています。

Database Resource Manager の管理者は、DBMS_RESOURCE_MANAGER パッケージ内のすべてのプロシージャを実行できます。次の表に、これらのプロシージャを示します。各プロシージャの使用方法については後述します。

プロシージャ	説明
CREATE_SIMPLE_PLAN	最大 8 個のコンシューマ・グループを含む単純なリソース・プランを 1 ステップで作成します。これは、このパッケージを初めて使用する際の最も迅速な方法です。

プロシージャ	説明
CREATE_PLAN	リソース・プランを作成し、その割当て方法を指定します。
UPDATE_PLAN	リソース・プランのコメント情報を更新します。
DELETE_PLAN	リソース・プランとそのディレクティブを削除します。
DELETE_PLAN_CASCADE	リソース・プランとそのすべての子を削除します。
CREATE_CONSUMER_GROUP	リソース・コンシューマ・グループを作成します。
UPDATE_CONSUMER_GROUP	コンシューマ・グループのコメント情報を更新します。
DELETE_CONSUMER_GROUP	コンシューマ・グループを削除します。
CREATE_PLAN_DIRECTIVE	プラン内のリソース・コンシューマ・グループ間、または複数レベルのプラン・スキーマ内のサブプラン間でリソースを割り当てるリソース・プラン・ディレクティブを指定します。
UPDATE_PLAN_DIRECTIVE	プラン・ディレクティブを更新します。
DELETE_PLAN_DIRECTIVE	プラン・ディレクティブを削除します。
CREATE_PENDING_AREA	プラン・スキーマを変更できるペンディング・エリア（スクラッチ領域）を作成します。
VALIDATE_PENDING_AREA	プラン・スキーマに対する保留中の変更の妥当性をチェックします。
CLEAR_PENDING_AREA	保留中のすべての変更をペンディング・エリアから消去します。
SUBMIT_PENDING_AREA	プラン・スキーマに対してすべての変更を発行します。
SET_INITIAL_CONSUMER_GROUP	ユーザーの初期コンシューマ・グループを設定します。
SWITCH_CONSUMER_GROUP_FOR_SESS	特定セッションのコンシューマ・グループを切り替えます。
SWITCH_CONSUMER_GROUP_FOR_USER	特定ユーザーに属しているすべてのセッションのコンシューマ・グループを切り替えます。

ADMIN オプションを持つ管理者は、必要に応じて管理権限を他のユーザーまたはロールに付与できます。そのためには、DBMS_RESOURCE_MANAGER_PRIVS パッケージを使用します。このパッケージには、次の表に示すプロシージャが含まれています。

プロシージャ	説明
GRANT_SYSTEM_PRIVILEGE	ユーザーまたはロールに、ADMINISTER_RESOURCE_MANAGER システム権限を付与します。
REVOKE_SYSTEM_PRIVILEGE	ユーザーまたはロールから、ADMINISTER_RESOURCE_MANAGER システム権限を取り消します。

プロシージャ	説明
GRANT_SWITCH_CONSUMER_GROUP	ユーザー、ロールまたは PUBLIC に、指定したリソース・コンシューマ・グループに切り替える許可を付与します。
REVOKE_SWITCH_CONSUMER_GROUP	ユーザー、ロールまたは PUBLIC から、指定したリソース・コンシューマ・グループに切り替える許可を取り消します。

次の例では、ユーザー scott に ADMIN オプションなしで管理権限を付与しています。したがって、scott は DBMS_RESOURCE_MANAGER パッケージ内のすべてのプロシージャを実行できますが、管理権限を他のユーザーに付与する GRANT_SYSTEM_PRIVILEGE プロシージャは使用できません。

```
EXEC DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SYSTEM_PRIVILEGE -
    (GRANTEE_NAME => 'scott', PRIVILEGE_NAME => 'ADMINISTER_RESOURCE_MANAGER', -
     ADMIN_OPTION => FALSE);
```

この権限は、REVOKE_SYSTEM_PRIVILEGE プロシージャを使用して取り消すことができます。

注意： ADMINISTER_RESOURCE_MANAGER システム権限を付与または取り消す方法は、DBMS_RESOURCE_MANAGER_PRIVS パッケージを使用する以外にありません。SQL の GRANT または REVOKE 文を使用して付与または取り消すことはできません。

DBMS_RESOURCE_MANAGER_PRIVS パッケージの他のプロシージャは、27-23 ページの「スイッチ特権の管理」を参照してください。

関連項目： Database Resource Manager のパッケージの詳細は、『Oracle9*i* 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。

- DBMS_RESOURCE_MANAGER
- DBMS_RESOURCE_MANAGER_PRIVS

単純なリソース・プランの作成

CREATE_SIMPLE_PLAN プロシージャを使用すれば、多くの状況に対応できる単純なリソース・プランを迅速に作成できます。このプロシージャでは、文を 1 回実行するだけで、コンシューマ・グループを作成し、それらにリソースを割り当てることができます。このプロシージャを使用する際は、ペンディング・エリアの作成、各コンシューマ・グループの個別作成およびリソース・プラン・ディレクティブの指定を行うために、後述のプロシージャをコールする必要はありません。

CREATE_SIMPLE_PLAN プロシージャには、次のパラメータを指定できます。

パラメータ	説明
SIMPLE_PLAN	プランの名前
CONSUMER_GROUP1	1 番目のグループのコンシューマ・グループ名
GROUP1_CPU	このグループに割り当てる CPU リソース
CONSUMER_GROUP2	2 番目のグループのコンシューマ・グループ名
GROUP2_CPU	このグループに割り当てる CPU リソース
CONSUMER_GROUP3	3 番目のグループのコンシューマ・グループ名
GROUP3_CPU	このグループに割り当てる CPU リソース
CONSUMER_GROUP4	4 番目のグループのコンシューマ・グループ名
GROUP4_CPU	このグループに割り当てる CPU リソース
CONSUMER_GROUP5	5 番目のグループのコンシューマ・グループ名
GROUP5_CPU	このグループに割り当てる CPU リソース
CONSUMER_GROUP6	6 番目のグループのコンシューマ・グループ名
GROUP6_CPU	このグループに割り当てる CPU リソース
CONSUMER_GROUP7	7 番目のグループのコンシューマ・グループ名
GROUP7_CPU	このグループに割り当てる CPU リソース
CONSUMER_GROUP8	8 番目のグループのコンシューマ・グループ名
GROUP8_CPU	このグループに割り当てる CPU リソース

このプロシージャでは、最大 8 個のコンシューマ・グループを指定できます。また、指定できるプラン・ディレクティブは、CPU に関するもののみです。プラン内に指定された各コンシューマ・グループには、レベル 2 の CPU パーセンテージが割り当てられます。また、プ

ランには、SYS_GROUP（ユーザー SYS と SYSTEM のために Oracle によって定義された初期コンシューマ・グループ）と OTHER_GROUPS も含まれます。

CREATE_SIMPLE_PLAN プロシージャの使用例

```
BEGIN
DBMS_RESOURCE_MANAGER.CREATE_SIMPLE_PLAN(SIMPLE_PLAN => 'simple_plan1',
   CONSUMER_GROUP1 => 'mygroup1', GROUP1_CPU => 80,
   CONSUMER_GROUP2 => 'mygroup2', GROUP2_CPU => 20);
END;
```

この文を実行すると、次のプランが作成されます。

コンシューマ・グループ	レベル 1	レベル 2	レベル 3
SYS_GROUP	100%	-	-
mygroup1	-	80%	-
mygroup2	-	20%	-
OTHER_GROUPS	-	-	100%

複雑なリソース・プランの作成

ここでは、より複雑なリソース・プランが必要な状況で使用できるアクションと DBMS_RESOURCE_MANAGER プロシージャについて説明します。この項の構成は、次のとおりです。

- ペンディング・エリアを使用したプラン・スキーマの作成
- リソース・プランの作成
- リソース・コンシューマ・グループの作成
- リソース・プラン・ディレクティブの指定

ペンディング・エリアを使用したプラン・スキーマの作成

プラン・スキーマを作成または変更する場合は、最初に**ペンディング・エリア**を作成する必要があります。これは、段階的な変更を加え、アクティブにする前に妥当性をチェックできるスクラッチ領域です。

ペンディング・エリアの作成

ペンディング・エリアを作成するには、次の文を使用します。

```
EXEC DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA;
```

実際には、プランの更新や新規プランの追加ができるように、ペンディング・エリアがアクティブになり、既存のプラン・スキーマ（アクティブなプラン・スキーマ）がすべてペンディング・エリアにロードされます。アクティブなプラン・スキーマとは、Database Resource Manager で使用できるように、すでにデータ・ディクショナリに格納されているスキーマです。先にペンディング・エリアをアクティブにしないで（つまり、作成せずに）、プランを更新または新規に追加しようとすると、ペンディング・エリアがアクティブでないことを示すエラー・メッセージが返されます。

ビューを使用すると、アクティブなすべてのリソース・プラン・スキーマと保留中のプラン・スキーマを表示できます。これらのビューについては、27-33 ページの「Database Resource Manager 情報の表示」を参照してください。

変更の妥当性チェック

ペンディング・エリアで変更を加えているときは、次の文によって、いつでも妥当性チェック・プロシージャをコールできます。

```
EXEC DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA;
```

このプロシージャは、加えられた変更が有効かどうかをチェックします。従う必要がある規則は、次のとおりです。これらの規則が妥当性チェック・プロシージャによってチェックされます。

1. プラン・スキーマにループを含めることはできません。
2. プラン・ディレクティブで参照されるプランやリソース・コンシューマ・グループは、すべて存在している必要があります。
3. すべてのプランに、プランまたはリソース・コンシューマ・グループを指すプラン・ディレクティブが必要です。
4. 特定のレベルの全パーセンテージの合計は、100 以下であることが必要です。
5. アクティブなインスタンスで現在トップレベルのプランとして使用されているプランは、削除できません。
6. 次のプラン・ディレクティブ・パラメータは、リソース・コンシューマ・グループを参照するプラン・ディレクティブでのみ使用できます。他のリソース・プランを参照するプラン・ディレクティブでは使用できません。
   - PARALLEL_DEGREE_LIMIT_P1
   - ACTIVE_SESS_POOL_P1
   - QUEUEING_P1
   - SWITCH_GROUP
   - SWITCH_TIME
   - SWITCH_ESTIMATE

- MAX_EST_EXEC_TIME
- UNDO_POOL

7. アクティブなプラン・スキーマに含まれるリソース・コンシューマ・グループ数は、32 以内にする必要があります。また、プランは最大 32 の子を持つことができます。トップレベルのプランのリーフはすべてリソース・コンシューマ・グループにする必要があります。つまり、プラン・スキーマの最下位レベルのプラン・ディレクティブは必ずコンシューマ・グループを参照するようにします。
8. プランとリソース・コンシューマ・グループには、異なる名前を使用する必要があります。
9. アクティブなプラン・スキーマ内のどこかに、OTHER_GROUPS のプラン・ディレクティブが存在する必要があります。これにより、現在のアクティブ・プラン内に含まれるコンシューマ・グループのいずれにも属していないセッションに、OTHER_GROUPS ディレクティブで指定したリソースが割り当てられることが保証されます。

これらの規則のいずれかに違反すると、エラー・メッセージが返されます。その場合は、変更を加えて問題を修正し、妥当性チェック・プロシージャを再度コールできます。

プラン・ディレクティブによって参照されない孤立したコンシューマ・グループを作成できます。これにより、当面は使用しないものの、将来的に実装するプランの一部として、コンシューマ・グループを作成できます。

変更の発行

変更の妥当性チェックを完了した後に、SUBMIT プロシージャをコールして変更をアクティブにします。

```
EXEC DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA;
```

SUBMIT プロシージャでは妥当性チェックも実行されるため、妥当性チェック・プロシージャを別個にコールする必要はありません。ただし、プラン・スキーマを大幅に変更している場合、通常は、変更の妥当性チェックを段階的に行う方が問題のデバッグ作業が容易になります。ペンディング・エリア内のすべての変更について妥当性チェックが成功するまで、変更は発行されません（つまり、アクティブになりません）。

SUBMIT_PENDING_AREA プロシージャは、変更の妥当性チェックとコミットに成功すると、ペンディング・エリアを消去（解除）します。

注意： VALIDATE_PENDING_AREA が正常終了しても、SUBMIT_PENDING_AREA のコールが失敗する場合があります。これが起こるのは、VALIDATE_PENDING_AREA をコールしてから SUBMIT_PENDING_AREA をコールするまでの間に、削除しようとしているプランがインスタンスによってロードされた場合などです。

ペンディング・エリアの消去

ペンディング・エリアを随時消去するためのプロシージャも用意されています。次の文は、すべての変更内容をペンディング・エリアから消去します。

```
EXEC DBMS_RESOURCE_MANAGER.CLEAR_PENDING_AREA;
```

再び変更を行うには、まず CREATE_PENDING_AREA プロシージャをコールする必要があります。

リソース・プランの作成

リソース・プランを作成するときは、次のパラメータを指定できます。

パラメータ	説明
PLAN	プランの名前。
COMMENT	任意のコメント。このフィールドはオプションです。
次のパラメータは、必要に応じて指定できます。それぞれのデフォルトは適切であり、その時点で許可されている唯一の値です。	
CPU_MTH	CPU のリソース割当て方法。デフォルトは EMPHASIS で、これはリソース・プラン・レベルで許可されている唯一の CPU 方法です。
ACTIVE_SESS_POOL_MTH	アクティブ・セッション・プールのリソース割当て方法。同時接続ユーザーの最大数を制御します。デフォルトは ACTIVE_SESS_POOL_ABSOLUTE で、これは使用可能な唯一の方法です。
PARALLEL_DEGREE_LIMIT_MTH	任意の操作について並列度の制限を指定するためのリソース割当て方法。デフォルトは PARALLEL_DEGREE_LIMIT_ABSOLUTE で、これは使用可能な唯一の方法です。
QUEUEING_MTH	キューイングのリソース割当て方法。キュー内のセッションの実行順序を制御します。デフォルトは FIFO_TIMEOUT で、これは使用可能な唯一の方法です。

Oracle には、単純な構造を含むリソース・プラン SYSTEM_PLAN が用意されています。環境によっては、このリソース・プランで十分です。このプランについては、27-30 ページの「Oracle が提供するプラン」を参照してください。

関連項目： リソース割当て方法の詳細は、『Oracle9*i* データベース概要』を参照してください。

プランの作成

プランを作成するには、CREATE_PLAN プロシージャを使用します。次の文は、great_bread プランを作成します。ここでは、デフォルトのリソース割当て方法を使用しています。

```
EXEC DBMS_RESOURCE_MANAGER.CREATE_PLAN(PLAN => 'great_bread', -
    COMMENT => 'great plan');
```

プランの更新

プラン情報を更新するには、UPDATE_PLAN プロシージャを使用します。引数を指定せずに UPDATE_PLAN プロシージャを実行すると、データ・ディクショナリ内のプラン情報は変更されません。次の文は、COMMENT パラメータを更新します。

```
EXEC DBMS_RESOURCE_MANAGER.UPDATE_PLAN(PLAN => 'great_bread', -
     NEW_COMMENT => 'great plan for great bread');
```

プランの削除

DELETE_PLAN プロシージャは、指定したプランと、それに対応付けられているすべてのプラン・ディレクティブを削除します。次の文は、great_bread プランとそのディレクティブを削除します。

```
EXEC DBMS_RESOURCE_MANAGER.DELETE_PLAN(PLAN => 'great_bread');
```

リソース・コンシューマ・グループ自体が削除されるのではなく、great_bread プランとの対応付けが解除されます。

DELETE_PLAN_CASCADE プロシージャは、指定したプランと、そのすべての子孫（プラン・ディレクティブ、サブプラン、リソース・コンシューマ・グループ）を削除します。DELETE_PLAN_CASCADE でエラーが発生するとロールバックされ、プラン・スキーマは変更されません。

リソース・コンシューマ・グループの作成

リソース・コンシューマ・グループを作成するときは、次のパラメータを指定できます。

パラメータ	説明
CONSUMER_GROUP	コンシューマ・グループの名前。
COMMENT	任意のコメント。
CPU_MTH	コンシューマ・グループの CPU リソース割当て方法。デフォルトは ROUND-ROBIN です。これは、現在リソース・コンシューマ・グループに使用可能な唯一の方法です。

次の 2 つの特別なコンシューマ・グループは、常にデータ・ディクショナリ内に存在します。これらのグループは変更または削除できません。

- DEFAULT_CONSUMER_GROUP

  これは、初期コンシューマ・グループが明示的に割り当てられていないすべてのユーザー / セッションの初期コンシューマ・グループです。DEFAULT_CONSUMER_GROUP は、PUBLIC に付与されたスイッチ特権を持っているため、すべてのユーザーにこのコンシューマ・グループのスイッチ特権が自動的に付与されます（27-23 ページの「スイッチ特権の管理」を参照）。

- OTHER_GROUPS

  このコンシューマ・グループは、明示的にユーザーに割り当てることはできません。OTHER_GROUPS は、任意のアクティブ・プランのスキーマ内に指定されたリソース・ディレクティブを必ず持っています。このグループは、DEFAULT_CONSUMER_GROUP を含め、現在アクティブなプラン・スキーマの一部でないコンシューマ・グループに属するすべてのセッションに選択的に適用されます。

また、他の 2 つのグループ SYS_GROUP および LOW_GROUP も、Oracle が提供する SYSTEM_PLAN の一部として提供されます。27-30 ページの「Oracle が提供するプラン」を参照してください。

コンシューマ・グループの作成

コンシューマ・グループを作成するには、CREATE_CONSUMER_GROUP プロシージャを使用します。次の文は、コンシューマ・グループ sales を作成します。この文を正常に実行するには、ペンディング・エリアをアクティブにする必要があります。

```
EXEC DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (CONSUMER_GROUP => 'sales', -
    COMMENT => 'retail and wholesale sales');
```

コンシューマ・グループの更新

コンシューマ・グループ情報を更新するには、UPDATE_CONSUMER_GROUP プロシージャを使用します。引数を指定せずに UPDATE_CONSUMER_GROUP プロシージャを実行すると、データ・ディクショナリ内のコンシューマ・グループ情報は変更されません。

コンシューマ・グループの削除

DELETE_CONSUMER_GROUP プロシージャは、指定されたコンシューマ・グループを削除します。コンシューマ・グループを削除すると、そのグループを初期コンシューマ・グループとして持っているすべてのユーザーには、初期コンシューマ・グループとして DEFAULT_CONSUMER_GROUP が割り当てられます。現在実行中のセッションのうち、削除されたコンシューマ・グループに属しているものはすべて、DEFAULT_CONSUMER_GROUP に切り替えられます。

リソース・プラン・ディレクティブの指定

リソース・プラン・ディレクティブは、リソース・プランにコンシューマ・グループを割り当てて、各リソース割当て方法のパラメータを指定します。リソース・プラン・ディレクティブを作成するときは、次のパラメータを指定できます。

パラメータ	説明
PLAN	リソース・プランの名前。
GROUP_OR_SUBPLAN	コンシューマ・グループまたはサブプランの名前。
COMMENT	任意のコメント。
CPU_P1	レベル 1 の CPU パーセンテージを指定します。CPU パラメータのデフォルトはすべて NULL です。
CPU_P2	レベル 2 の CPU パーセンテージを指定します。
CPU_P3	レベル 3 の CPU パーセンテージを指定します。
CPU_P4	レベル 4 の CPU パーセンテージを指定します。
CPU_P5	レベル 5 の CPU パーセンテージを指定します。
CPU_P6	レベル 6 の CPU パーセンテージを指定します。
CPU_P7	レベル 7 の CPU パーセンテージを指定します。
CPU_P8	レベル 8 の CPU パーセンテージを指定します。
ACTIVE_SESS_POOL_P1	コンシューマ・グループ内で同時にアクティブにできるセッションの最大数を指定します。デフォルトは UNLIMITED です。

パラメータ	説明
QUEUEING_P1	実行キュー内で実行されるのを待機しているジョブがタイムアウトするまでの時間を秒数で指定します。デフォルトは UNLIMITED です。
PARALLEL_DEGREE_LIMIT_P1	任意の操作について並列度の制限を指定します。デフォルトは UNLIMITED です。
SWITCH_GROUP	他の切替え基準が満たされた場合に、このセッションを切り替える先のコンシューマ・グループを指定します。デフォルトは NULL です。
SWITCH_TIME	他のコンシューマ・グループに切り替わるまでにセッションが実行できる時間を秒数で指定します。デフォルトは UNLIMITED です。
SWITCH_ESTIMATE	TRUE の場合、Oracle は実行時間の見積りを使用して、操作を開始する前にその操作のコンシューマ・グループを自動的に切り替えます。デフォルトは FALSE です。
MAX_EST_EXEC_TIME	セッションで許可される最大の実行時間を秒数で指定します。デフォルトは UNLIMITED です。
UNDO_POOL	コンシューマ・グループで生成される UNDO の合計量の最大値を KB で設定します。デフォルトは UNLIMITED です。

リソース・プラン・ディレクティブの作成

リソース・プラン・ディレクティブを作成するには、CREATE_PLAN_DIRECTIVE を使用します。次の文は、プラン great_bread のリソース・プラン・ディレクティブを作成します。

```
EXEC DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (PLAN => 'great_bread', -
     GROUP_OR_SUBPLAN => 'sales', COMMENT => 'sales group', -
     CPU_P1 => 60, PARALLEL_DEGREE_LIMIT_P1 => 4);
```

図 27-1 のようなプランを作成するには、次の文を実行します。

```
BEGIN
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (PLAN => 'great_bread',
     GROUP_OR_SUBPLAN => 'market', COMMENT => 'marketing group',
     CPU_P1 => 20);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (PLAN => 'great_bread',
     GROUP_OR_SUBPLAN => 'develop', COMMENT => 'development group',
     CPU_P1 => 20);
```

```
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (PLAN => 'great_bread',
     GROUP_OR_SUBPLAN => 'OTHER_GROUPS', COMMENT => 'this one is required',
     CPU_P1 => 0, CPU_P2 => 100);
END;
```

このプランでは、コンシューマ・グループ sales に対して操作の最大並列度が 4 に設定されていますが、他のコンシューマ・グループの並列度に制限はありません。また、レベル 1 の CPU リソースが残っている場合は、OTHER_GROUPS に（100%）割り当てられます。

リソース・プラン・ディレクティブの更新

プラン・ディレクティブを更新するには、UPDATE_PLAN_DIRECTIVE プロシージャを使用します。この例では、リソース・コンシューマ・グループ develop の CPU 割当てを変更しています。

```
EXEC DBMS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE (PLAN => 'great_bread', -
     GROUP_OR_SUBPLAN => 'develop', NEW_CPU_P1 => 15);
```

引数を指定せずに UPDATE_PLAN_DIRECTIVE プロシージャを実行すると、データ・ディクショナリ内のプラン・ディレクティブは変更されません。

リソース・プラン・ディレクティブの削除

リソース・プラン・ディレクティブを削除するには、DELETE_PLAN_DIRECTIVE プロシージャを使用します。

リソース・プラン・ディレクティブの相互作用

同じコンシューマ・グループを参照する複数のリソース・プラン・ディレクティブがある場合は、個々のケースに対して次の規則が適用されます。

1. コンシューマ・グループの並列度制限は、すべての入力値の最小値になります。
2. コンシューマ・グループのアクティブ・セッション・プールはすべての入力値の合計になり、キューのタイムアウトはすべての入力タイムアウト値の最小値になります。
3. 切替えグループと切替え時間がそれぞれ複数ある場合、Database Resource Manager は、すべての入力値の中で最も制限が大きいものを選択します。具体的には、次のようになります。
   - SWITCH_TIME = *min*（入力されるすべての over_switch_time 値）
   - SWITCH_ESTIMATE = FALSE よりも SWITCH_ESTIMATE = TRUE が優先する

注意： 両方のプラン・ディレクティブの切替え時間が同じで、切替えグループが異なる場合、切替え先のグループは、Database Resource Manager によって任意に決められた一方に固定されます。

4. 切替え時間を超えたためにセッションが別のコンシューマ・グループに切り替えられた場合は、切替え後のコンシューマ・グループのアクティブ・セッション・プールがいっぱいであっても、そのセッションは実行されます。
5. 見積り実行時間の最大値には、すべての入力値の中で最も制限の大きいものが選択されます。具体的には、次のようになります。
   - max_estimated_exec_time = min（入力されるすべての max_estimated_exec_time 値）

リソース・コンシューマ・グループの管理

Database Resource Manager を使用可能にする前に、ユーザーにリソース・コンシューマ・グループを割り当てる必要があります。DBMS_RESOURCE_MANAGER パッケージには、Database Resource Manager で使用される要素を作成、更新または削除するためのプロシージャだけでなく、ユーザーにリソース・コンシューマ・グループを割り当てるためのプロシージャも含まれています。また、ユーザー・セッションを一時的に別のコンシューマ・グループに切り替えるためのプロシージャも用意されています。

Database Resource Manager のシステム権限の付与に関連してすでに説明したように、DBMS_RESOURCE_MANAGER_PRIVS パッケージを使用して、別のユーザーにスイッチ特権を付与することもできます。この権限を付与されたユーザーは、自分のコンシューマ・グループを変更できます。

これから説明するプロシージャでは、ペンディング・エリアは使用しません。

初期リソース・コンシューマ・グループの割当て

ユーザーの初期コンシューマ・グループとは、そのユーザーによって作成されたセッションが最初に属するコンシューマ・グループです。ユーザーの初期コンシューマ・グループは、ユーザーを作成したときに、自動的に DEFAULT_CONSUMER_GROUP に設定されます。

あるコンシューマ・グループをユーザーの初期コンシューマ・グループにするには、そのコンシューマ・グループに切り替えるための権限をユーザー（または PUBLIC）に付与しておく必要があります。この権限をスイッチ特権と呼びます。スイッチ特権については、27-23 ページの「スイッチ特権の管理」を参照してください。初期コンシューマ・グループへのスイッチ特権は、そのユーザーに付与されているロールからは継承できません。

次の文は、ユーザーの初期コンシューマ・グループの設定方法を示しています。

```
EXEC DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SWITCH_CONSUMER_GROUP ('scott', 'sales',-
    TRUE);
EXEC DBMS_RESOURCE_MANAGER.SET_INITIAL_CONSUMER_GROUP('scott', 'sales');
```

リソース・コンシューマ・グループの変更

DBMS_RESOURCE_MANAGER パッケージには、管理者が実行中のセッションのリソース・コンシューマ・グループを変更できるように、2 つのプロシージャが含まれています。この 2 つのプロシージャでは、コーディネータのセッションに対応付けられたパラレル実行サーバー・セッションのコンシューマ・グループも変更できます。これらのプロシージャで行った変更は永続的ではなく、現行セッションのみに影響します。ユーザーの初期コンシューマ・グループも変更されません。

あるユーザーが CPU を過剰に使用している場合、管理者は、そのユーザーのセッションを停止するかわりに、ユーザーのコンシューマ・グループを CPU パーセンテージの低いグループに変更できます。この切替えは、コンシューマ・グループを自動的に切り替えるリソース・プラン・ディレクティブを使用して、自動的に実行できます。

セッションの切替え

SWITCH_CONSUMMER_GROUP_FOR_SESS は、指定したセッションを、指定したリソース・コンシューマ・グループに即時に移動します。この文を使用すれば、実質上、優先順位を変更できます。次の文は、特定セッションのリソース・コンシューマ・グループを、新しいコンシューマ・グループに変更します。セッション識別子（SID）が 17、セッション・シリアル番号（SERIAL#）が 12345 のセッションが、high_priority コンシューマ・グループに変更されます。

```
EXEC DBMS_RESOURCE_MANAGER.SWITCH_CONSUMER_GROUP_FOR_SESS ('17', '12345', -
   'high_priorty');
```

SID、セッション・シリアル番号、およびセッションの現行リソース・コンシューマ・グループは、V$SESSION データ・ディクショナリ・ビューを使用して確認できます。

ユーザー・セッションの切替え

SWITCH_CONSUMER_GROUP_FOR_USER プロシージャは、指定されたユーザー名を持つすべてのセッションのリソース・コンシューマ・グループを変更します。

```
EXEC DBMS_RESOURCE_MANAGER.SWITCH_CONSUMER_GROUP_FOR_USER ('scott', -
    'low_group');
```

スイッチ特権の管理

DBMS_RESOURCE_MANAGER_PRIVS パッケージを使用すると、ユーザー、ロールまたは PUBLIC にスイッチ特権を付与したり、取り消したりできます。スイッチ特権は、ユーザーに対して自分の現行リソース・コンシューマ・グループを指定のリソース・コンシューマ・グループに切り替える権限を与えます。このパッケージでは、スイッチ特権を取り消すこともできます。

実際の切替えは、DBMS_SESSION パッケージのプロシージャを実行することによって行われます。スイッチ特権が付与されているユーザー（またはそのユーザーが所有するプロシージャ）は、SWITCH_CURRENT_CONSUMER_GROUP プロシージャを使用して、別のリソース・コンシューマ・グループに切り替えることができます。切替え後のグループは、そのユーザーが明示的に切替えを許可されたグループである必要があります。

スイッチ特権の付与

次の例では、コンシューマ・グループに切り替える権限を付与しています。ユーザー scott には、コンシューマ・グループ bug_batch_group に切り替える権限が付与されます。

```
EXEC DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SWITCH_CONSUMER_GROUP ('scott', -
     'bug_batch_group', TRUE);
```

また、ユーザー scott には、bug_batch_group のスイッチ特権を他のユーザーに付与する許可も付与されます。

特定のコンシューマ・グループに切り替えるユーザー許可を付与すると、そのユーザーは自分の現行コンシューマ・グループを新しいコンシューマ・グループに切り替えることができます。

特定のリソース・コンシューマ・グループに切り替えるロール許可を付与すると、そのロールが付与されており、使用可能になっているユーザーは、自分の現行コンシューマ・グループを新しいコンシューマ・グループに即時に切り替えることができます。

特定のコンシューマ・グループに切り替える許可を PUBLIC に付与すると、だれでもそのグループに切り替えることができます。

GRANT_OPTION 引数が TRUE の場合、コンシューマ・グループのスイッチ特権が付与されたユーザーは、そのコンシューマ・グループのスイッチ特権を他のユーザーに付与することもできます。

スイッチ特権の取消し

次の例では、ユーザー scott から、コンシューマ・グループ bug_batch_group に切り替える権限を取り消しています。

```
EXEC DBMS_RESOURCE_MANAGER_PRIVS.REVOKE_SWITCH_CONSUMER_GROUP ('scott', -
     'bug_batch_group');
```

特定のコンシューマ・グループに対するユーザーのスイッチ特権を取り消した場合は、そのユーザーがそのコンシューマ・グループに切り替えようとすると失敗します。ユーザーの初期コンシューマ・グループを取り消すと、そのユーザーはログイン時に自動的に DEFAULT_CONSUMER_GROUP に割り当てられます。

コンシューマ・グループに対するロールのスイッチ特権を取り消すと、そのロールを介してのみコンシューマ・グループのスイッチ特権を与えられているユーザーは、そのコンシューマ・グループに切り替えることができなくなります。

コンシューマ・グループに対するスイッチ特権を PUBLIC から取り消した場合は、直接または PUBLIC を介してスイッチ特権を明示的に割り当てられているユーザーを除き、そのコンシューマ・グループに切り替えることができなくなります。

DBMS_SESSION パッケージを使用したコンシューマ・グループの切替え

スイッチ特権が付与されているユーザーは、DBMS_SESSION パッケージの SWITCH_CURRENT_CONSUMER_GROUP プロシージャを使用して、自分の現行コンシューマ・グループを切り替えることができます。

このプロシージャによって、ユーザーはスイッチ特権を持っているコンシューマ・グループに切り替えることができます。他のプロシージャからこのプロシージャがコールされた場合、ユーザーはコール側のプロシージャの所有者がスイッチ特権を持っているコンシューマ・グループに切り替えることができます。

このプロシージャのパラメータは、次のとおりです。

パラメータ	説明
NEW_CONSUMER_GROUP	切替え先のコンシューマ・グループ。
OLD_CONSUMER_GROUP	出力パラメータ。切替え元のコンシューマ・グループの名前が格納されます。このパラメータは、後で元のコンシューマ・グループに再び切り替えるときに使用できます。
INITIAL_GROUP_ON_ERROR	切替えエラー発生時の動作を制御します。 TRUE に設定すると、エラーが発生した場合にユーザーは初期コンシューマ・グループに切り替わります。 FALSE に設定すると、そのままエラーが出力されます。

次の例は、新しいコンシューマ・グループへの切替え方法を示しています。出力パラメータ old_group の値が出力されているので、変更前のコンシューマ・グループ名がどのように保存されているかがわかります。

```
SET serveroutput on
DECLARE
    old_group varchar2(30);
BEGIN
DBMS_SESSION.SWITCH_CURRENT_CONSUMER_GROUP('sales', old_group, FALSE);
DBMS_OUTPUT.PUT_LINE('OLD GROUP = ' || old_group);
END;
```

次の行が出力されます。

```
OLD GROUP = DEFAULT_CONSUMER_GROUP
```

DBMS_SESSION パッケージは、PL/SQL アプリケーション内部から使用できます。これにより、アプリケーションからコンシューマ・グループを変更できるので、実質上、優先順位の動的な変更が可能になります。

注意： Database Resource Manager は、一般的なデータベース・ユーザー名を使用してアプリケーションにログインしている環境でも機能します。DBMS_SESSION パッケージは、セッションの開始時、または特定のモジュールがコールされたときにセッションのコンシューマ・グループ割当てを切り替えるために使用できます。

関連項目： DBMS_SESSION パッケージのその他の例と詳細は、『Oracle9*i* 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。

Database Resource Manager を使用可能にする方法

Database Resource Manager を使用可能にするには、RESOURCE_MANAGER_PLAN 初期化パラメータを設定します。このパラメータには、トップレベルのプランを指定します。これにより、このインスタンスで使用するプラン・スキーマが識別されます。このパラメータでプランを指定しない場合、Database Resource Manager はアクティブになりません。次の例では、Database Resource Manager をアクティブにし、トップレベルのプランとして mydb_plan を指定しています。

```
RESOURCE_MANAGER_PLAN = mydb_plan
```

また、ALTER SYSTEM 文を使用して、Database Resource Manager のアクティブ化と解除、または現行のトップレベル・プランの変更が可能です。この例では、トップレベルのプランとして mydb_plan を指定しています。

```
ALTER SYSTEM SET RESOURCE _MANAGER_PLAN = mydb_plan;
```

指定したプランがデータ・ディクショナリに存在しない場合は、エラー・メッセージが返されます。

Database Resource Manager を解除するには、次の文を発行します。

```
ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = '';
```

各種の方法を組み合せた Database Resource Manager の例

ここでは、リソース・プラン・スキーマの例をいくつか示します。ここで取り上げる例は、次のとおりです。

- 複数レベルのスキーマの例
- 各種のリソース割当て方法を使用した例
- Oracle が提供するプラン

複数レベルのスキーマの例

次の文は、図 27-3 に図示されている複数レベルのスキーマを作成します。この例では、デフォルトのプランとリソース・コンシューマ・グループによる方法が使用されています。

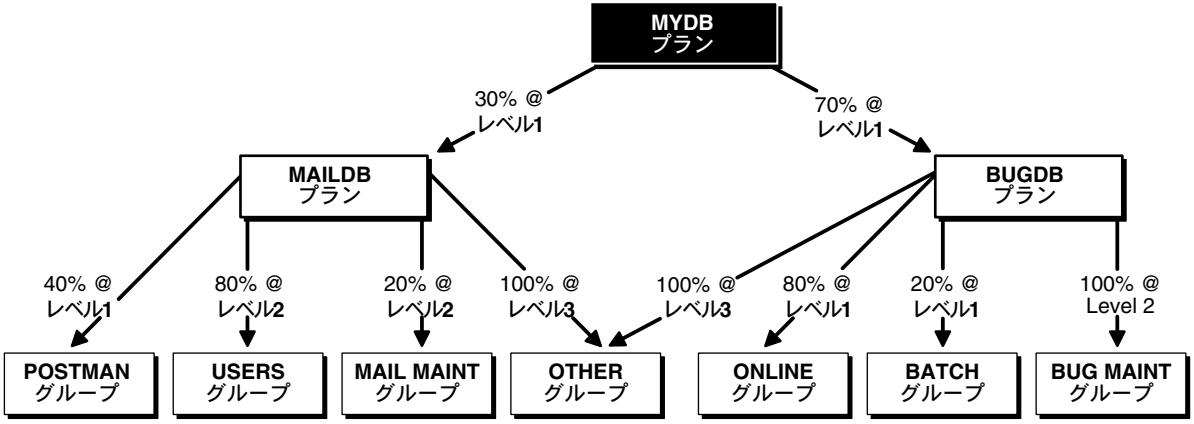
```
BEGIN
DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
DBMS_RESOURCE_MANAGER.CREATE_PLAN(PLAN => 'bugdb_plan',
   COMMENT => 'Resource plan/method for bug users sessions');
DBMS_RESOURCE_MANAGER.CREATE_PLAN(PLAN => 'maildb_plan',
   COMMENT => 'Resource plan/method for mail users sessions');
DBMS_RESOURCE_MANAGER.CREATE_PLAN(PLAN => 'mydb_plan',
   COMMENT => 'Resource plan/method for bug and mail users sessions');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Bug_Online_group',
```

```
   COMMENT => 'Resource consumer group/method for online bug users sessions');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Bug_Batch_group',
   COMMENT => 'Resource consumer group/method for batch job bug users sessions');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Bug_Maintenance_group',
   COMMENT => 'Resource consumer group/method for users sessions for bug db maint');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Mail_users_group',
   COMMENT => 'Resource consumer group/method for mail users sessions');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Mail_Postman_group',
   COMMENT => 'Resource consumer group/method for mail postman');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Mail_Maintenance_group',
   COMMENT => 'Resource consumer group/method for users sessions for mail db maint');
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'bugdb_plan',
   GROUP_OR_SUBPLAN => 'Bug_Online_group',
   COMMENT => 'online bug users sessions at level 1', CPU_P1 => 80, CPU_P2=> 0,
   PARALLEL_DEGREE_LIMIT_P1 => 8);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'bugdb_plan',
   GROUP_OR_SUBPLAN => 'Bug_Batch_group',
   COMMENT => 'batch bug users sessions at level 1', CPU_P1 => 20, CPU_P2 => 0,
   PARALLEL_DEGREE_LIMIT_P1 => 2);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'bugdb_plan',
   GROUP_OR_SUBPLAN => 'Bug_Maintenance_group',
   COMMENT => 'bug maintenance users sessions at level 2', CPU_P1 => 0, CPU_P2 => 100,
   PARALLEL_DEGREE_LIMIT_P1 => 3);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'bugdb_plan',
   GROUP_OR_SUBPLAN => 'OTHER_GROUPS',
   COMMENT => 'all other users sessions at level 3', CPU_P1 => 0, CPU_P2 => 0,
   CPU_P3 => 100);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'maildb_plan',
   GROUP_OR_SUBPLAN => 'Mail_Postman_group',
   COMMENT => 'mail postman at level 1', CPU_P1 => 40, CPU_P2 => 0,
   PARALLEL_DEGREE_LIMIT_P1 => 4);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'maildb_plan',
   GROUP_OR_SUBPLAN => 'Mail_users_group',
   COMMENT => 'mail users sessions at level 2', CPU_P1 => 0, CPU_P2 => 80,
   PARALLEL_DEGREE_LIMIT_P1 => 4);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'maildb_plan',
   GROUP_OR_SUBPLAN => 'Mail_Maintenance_group',
   COMMENT => 'mail maintenance users sessions at level 2', CPU_P1 => 0, CPU_P2 => 20,
   PARALLEL_DEGREE_LIMIT_P1 => 2);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'maildb_plan',
   GROUP_OR_SUBPLAN => 'OTHER_GROUPS',
   COMMENT => 'all other users sessions at level 3', CPU_P1 => 0, CPU_P2 => 0,
   CPU_P3 => 100);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'mydb_plan',
   GROUP_OR_SUBPLAN => 'maildb_plan',
   COMMENT=> 'all mail users sessions at level 1', CPU_P1 => 30);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'mydb_plan',
   GROUP_OR_SUBPLAN => 'bugdb_plan',
   COMMENT => 'all bug users sessions at level 1', CPU_P1 => 70);
```

```
DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();
DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
END;
```

妥当性チェックは SUBMIT_PENDING_AREA によって暗黙的に実行されるので、直前の VALIDATE_PENDING_AREA のコールはオプションです。

図 27-3　複数レベルのスキーマ



各種のリソース割当て方法を使用した例

ここで示す例は、パッケージ化された Enterprise Resource Planning（ERP）または Customer Relationship Management（CRM）をサポートするデータベース用のプランを表します。このような環境で必要な処理は多種多様です。大規模なパラレル問合せを含む長時間実行のバッチ・ジョブとともに、短いトランザクションや短い問合せが混在している場合があります。その目的は、バッチ・ジョブをパラレルに実行しながら、オンライン・トランザクション処理（OLTP）の適切な応答時間を得ることにあります。

次の表にプランがまとめられています。

グループ	CPU リソース割当て %	アクティブ・セッション・プール・パラメータ	自動切替えパラメータ	最大見積り実行時間	UNDO プール
oltp	レベル 1: 80%		切替え先グループ : batch 切替え時間 : 3 短い問合せ : TRUE		サイズ : 200KB

グループ	CPU リソース割当て %	アクティブ・セッション・プール・パラメータ	自動切替えパラメータ	最大見積り実行時間	UNDO プール
batch	レベル 2: 100%	プール・サイズ : 5 タイムアウト : 600		時間 : 3600	
OTHER_GROUPS	レベル 3: 100%				

次の文は、この表のプランを erp_plan という名前で作成します。

```
BEGIN
DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
DBMS_RESOURCE_MANAGER.CREATE_PLAN(PLAN => 'erp_plan',
  COMMENT => 'Resource plan/method for ERP Database');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'oltp',
  COMMENT => 'Resource consumer group/method for OLTP jobs');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'batch',
  COMMENT => 'Resource consumer group/method for BATCH jobs');
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'erp_plan',
  GROUP_OR_SUBPLAN => 'oltp', COMMENT => 'OLTP sessions', CPU_P1 => 80,
  SWITCH_GROUP => 'batch', SWITCH_TIME => 3,SWITCH_ESTIMATE => TRUE,
  UNDO_POOL => 200);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'erp_plan',
  GROUP_OR_SUBPLAN => 'batch', COMMENT => 'BATCH sessions', CPU_P2 => 100,
  ACTIVE_SESS_POOL_P1 => 5, QUEUEING_P1 => 600,
  MAX_EST_EXEC_TIME => 3600);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'erp_plan',
  GROUP_OR_SUBPLAN => 'OTHER_GROUPS', COMMENT => 'mandatory', CPU_P3 => 100);
DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();
DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
END;
```

Oracle が提供するプラン

Oracle には、システム・セッションの優先順位を設定するデフォルトのリソース・マネージャ・プラン、SYSTEM_PLAN が用意されています。SYSTEM_PLAN の定義は、次のとおりです。

	CPU リソース割当て		
リソース・コンシューマ・グループ	**レベル 1**	**レベル 2**	**レベル 3**
SYS_GROUP	100%	0%	0%
OTHER_GROUPS	0%	100%	0%
LOW_GROUP	0%	0%	100%

このプランで Oracle は次のグループを用意しています。

- SYS_GROUP は、ユーザー SYS および SYSTEM の初期コンシューマ・グループです。
- OTHER_GROUPS は、現在アクティブなプラン・スキーマの一部でないコンシューマ・グループに属するすべてのセッションに適用されます。
- LOW_GROUP は、このプランで SYS_GROUP および OTHER_GROUPS より優先順位の低いグループを表します。LOW_GROUP に含めるユーザー・セッションは、管理者が指定できます。このグループのスイッチ特権は、PUBLIC に付与されています。

これらのグループの使用は選択可能で、変更または削除することもできます。

環境にとって適切な場合は、Oracle が提供するこの単純なプランを使用できます。

Database Resource Manager の監視とチューニング

Database Resource Manager を効果的に監視およびチューニングするには、代理環境を設計する必要があります。Database Resource Manager は、システム使用率が高い大規模な本番環境で最も効果的に機能します。システムの負荷が不十分な状態でテストを行うと、測定された CPU 割当てとアクティブ・リソース・プランに指定する割当てが大きく異なる場合があります。

環境の作成

代理環境を作成するには、CPU リソースが不足するほど十分な負荷（CPU リソースの要求）が必要です。次の規則に従えば、テスト環境で生成される実際の（測定された）リソース割当てと、アクティブ・リソース・プランに指定するリソース割当てが一致します。

1. 十分な負荷を生み出すために必要な同時実行プロセスを最小限の数だけ作成します。この数には、次の数の大きい方を使用します。
   - コンシューマ・グループごとに 4 プロセス。
   - コンシューマ・グループごとに 1.5 ×（プロセッサ数）。結果が整数でない場合は切り上げます。
2. すべてのプロセスには、それぞれが動作するコンシューマ・グループに割り当てられた CPU リソースをすべて消費する能力が必要です。何が起こってもループし続けるリソース集約的なプログラムを作成してください。これは、次のような簡単なプログラムでかまいません。

```
BEGIN
DECLARE
    m NUMBER;
  BEGIN
    FOR i IN 1..100000 LOOP
      FOR j IN 1..100000 LOOP
        m := sqrt(4567);
      END LOOP;
    END LOOP;
  END;
END;
/
```

予測した結果を出す必要性

すべてのグループが要求どおりの CPU リソースを確保できる場合、最初に Database Resource Manager は、割り当てられたパーセンテージに従うのではなく、システム全体のスループットを最大限に高めようとします。たとえば、次のような条件を考えます。

- コンシューマ・グループごとに、実行可能なプロセスがただ 1 つのみある。
- 各プロセスは停止することなく動作し続ける。
- CPU は 4 つある。

この場合、各コンシューマ・グループで測定される CPU 割当ては、アクティブ・リソース・プランに指定された割当てとは関係なく 25% になります。

前述の 1 の計算を左右する別の要因があります。オペレーティング・システム・レベルでプロセッサ・アフィニティに従ってスケジューリングが実行された場合、負荷が十分でないシステムでは CPU 割当てにゆがみが生じることがあります。これについて、次の段落で説明します。

オペレーティング・システムの標準的なスケジューリング・アルゴリズムでは、同時実行プロセスの数が一定のレベルに達するまで、100% の使用は避けられます。Database Resource Manager は、実行プロセスの数を制限することによって、CPU の使用を制御します。また、どのプロセスにどれだけの時間実行を許可するかを決定することによって、CPU リソースの割当てを制御します。ある CPU のリソースが使用可能であり、他のプロセッサはすべて 100% 使用されている場合、即時にではありませんが、オペレーティング・システムは使用率の低いプロセッサにプロセスを移動します。

プロセッサ・アフィニティでは、オペレーティング・システムは、CPU 間でプロセスを強制的に移動するかわりに別の実行プロセスが CPU 使用を終えることを期待しながら、プロセスを移動するまで一時的に待機します。この方式は、負荷が 100% であり、多くのプロセスが待機しているシステムで機能します。大規模な本番環境では、現在の CPU キャッシュ・データを無効化して別の CPU キャッシュ・データを新たにロードするのはかなりの負担となるため、プロセッサ・アフィニティによってパフォーマンスを大幅に向上できます。ほとんどのプラットフォームで、プロセスはプロセッサ・アフィニティを持っているので、コンシューマ・グループごとに CPU の数より多くのプロセスを実行すべきです。そうでなければ、システムを 100% 使用することはできません。

監視結果

CPU の使用を監視するには、V$RSRC_CONSUMER_GROUP ビューを使用します。このビューには、各コンシューマ・グループのすべてのセッションで消費された CPU タイムの累積値が含まれています。その他にも、チューニングに役立つ測定値が含まれています。

```
SQL> SELECT NAME, CONSUMED_CPU_TIME FROM V$RSRC_CONSUMER_GROUP;

NAME                             CONSUMED_CPU_TIME
-------------------------------- -----------------
OTHER_GROUPS                                 14301
TEST_GROUP                                    8802
TEST_GROUP2                                      0

3 rows selected.
```

Database Resource Manager 情報の表示

Database Resource Manager に対応付けられているビューを次の表に示します。

ビュー	説明
DBA_RSRC_CONSUMER_GROUP_PRIVS USER_RSRC_CONSUMER_GROUP_PRIVS	DBA ビューには、すべてのリソース・コンシューマ・グループと、それが付与されているユーザーおよびロールがリストされます。USER ビューには、ユーザーに付与されたすべてのリソース・コンシューマ・グループがリストされます。
DBA_RSRC_CONSUMER_GROUPS	データベースに存在するすべてのリソース・コンシューマ・グループがリストされます。
DBA_RSRC_MANAGER_SYSTEM_PRIVS USER_RSRC_MANAGER_SYSTEM_PRIVS	DBA ビューには、Database Resource Manager のシステム権限が付与されているユーザーとロールがすべてリストされます。USER ビューには、DBMS_RESOURCE_MANAGER パッケージのシステム権限が付与されているユーザーがすべてリストされます。
DBA_RSRC_PLAN_DIRECTIVES	データベースに存在するすべてのリソース・プラン・ディレクティブがリストされます。
DBA_RSRC_PLANS	データベースに存在するすべてのリソース・プランがリストされます。
DBA_USERS USERS_USERS	DBA ビューには、データベース内のすべてのユーザーに関する情報が含まれます。Database Resource Manager に関連した情報としては、ユーザーの初期リソース・コンシューマ・グループが含まれます。USER ビューには、現行ユーザーに関する情報が含まれます。Database Resource Manager に関連した情報としては、現行ユーザーの初期リソース・コンシューマ・グループが含まれます。

ビュー	説明
V$ACTIVE_SESS_POOL_MTH	使用可能なアクティブ・セッション・プールによるリソース割当て方法がすべて表示されます。
V$PARALLEL_DEGREE_LIMIT_MTH	使用可能な並列度制限によるリソース割当て方法がすべて表示されます。
V$QUEUEING	使用可能なキューイングによるリソース割当て方法がすべて表示されます。
V$RSRC_CONSUMER_GROUP	アクティブなリソース・コンシューマ・グループに関する情報が表示されます。このビューは、チューニングに使用できます。
V$RSRC_CONSUMER_GROUP_CPU_MTH	リソース・コンシューマ・グループで使用可能な CPU リソース割当て方法がすべて表示されます。
V$RSRC_PLAN	現在のアクティブ・リソース・プランの名前がすべて表示されます。
V$RSRC_PLAN_CPU_MTH	リソース・プランで使用可能な CPU リソース割当て方法がすべて表示されます。
V$SESSION	現行セッションごとにセッション情報が表示されます。その中には、各現行セッションのリソース・コンシューマ・グループの名前が含まれます。

これらのビューを使用して、権限やプラン・スキーマを表示できます。また、これらのビューを監視して、Database Resource Manager のチューニングに使用する情報を収集することもできます。次に、これらのビューの使用例をいくつか示します。

関連項目： これらのビューの内容に関する詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

ユーザーまたはロールに権限付与されたコンシューマ・グループの表示

DBA_RSRC_CONSUMER_GROUP_PRIVS ビューには、ユーザーまたはロールに付与されたコンシューマ・グループが表示されます。具体的には、ユーザーまたはロールが属しているグループや、切替え先のグループが表示されます。たとえば、次に示すビューでは、ユーザー scott はコンシューマ・グループ market または sales に所属でき、sales グループに他のユーザーを割り当てる（付与する）権限を持っています。しかし、market グループに他のユーザーを割り当てる権限はありません。どちらのグループも、scott の初期コンシューマ・グループではありません。

```
SQL> SELECT * FROM DBA_RSRC_CONSUMER_GROUP_PRIVS;

GRANTEE                        GRANTED_GROUP                  GRA INI
------------------------------ ------------------------------ --- ---
PUBLIC                         DEFAULT_CONSUMER_GROUP         YES YES
PUBLIC                         LOW_GROUP                      NO  NO
```

```
SCOTT                          MARKET                         NO  NO
SCOTT                          SALES                          YES NO
SYSTEM                         SYS_GROUP                      NO  YES
```

scott は、DBMS_RESOURCE_MANAGER_PRIVS パッケージを使用してこれらのグループに切り替える権限をすでに付与されています。

プラン・スキーマ情報の表示

この例では、DBA_RSRC_PLANS ビューを使用して、データベース内に定義されているすべてのリソース・プランを表示する方法を示しています。表示されるプランはすべてアクティブです。つまり、ペンディング・エリアにステージングされていません。

```
SQL> SELECT PLAN,COMMENTS,STATUS FROM DBA_RSRC_PLANS;

PLAN         COMMENTS                                                STATUS
-----------  -------------------------------------------------------  ------
SYSTEM_PLAN  Plan to give system sessions priority                    ACTIVE
BUGDB_PLAN   Resource plan/method for bug users sessions              ACTIVE
MAILDB_PLAN  Resource plan/method for mail users sessions             ACTIVE
MYDB_PLAN    Resource plan/method for bug and mail users sessions     ACTIVE
GREAT_BREAD  Great plan for great bread                               ACTIVE
ERP_PLAN     Resource plan/method for ERP Database                    ACTIVE

6 rows selected.
```

セッションの現行コンシューマ・グループの表示

V$SESSION ビューを使用すれば、セッションに現在割り当てられているコンシューマ・グループを表示できます。

```
SQL> SELECT SID,SERIAL#,USERNAME,RESOURCE_CONSUMER_GROUP FROM V$SESSION;

SID    SERIAL#  USERNAME                  RESOURCE_CONSUMER_GROUP
-----  -------  ------------------------  --------------------------------
.
.
.
   11       136 SYS                       SYS_GROUP
   13     16570 SCOTT                     SALES

10 rows selected.
```

現在アクティブなプランの表示

この例では、mydb_plan（27-26 ページの「複数レベルのスキーマの例」の文で作成されたプラン）をトップレベルのプランとして設定しています。V$RSRC_PLAN ビューを問い合せると、現在のアクティブ・プランが表示されます。

```
SQL> ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = mydb_plan;

System altered.

SQL> SELECT * FROM V$RSRC_PLAN;

NAME
--------------------------------
MYDB_PLAN
MAILDB_PLAN
BUGDB_PLAN
```

第 VI 部

分散データベースの管理

第 VI 部では、分散データベース環境の管理について説明します。第 VI 部の構成は、次のとおりです。

28

分散データベースの概念

Oracle の分散データベース・アーキテクチャの基本概念と用語について説明します。この章の内容は、次のとおりです。

- 分散データベース・アーキテクチャ
- データベース・リンク
- 分散データベースの管理
- 分散システムでのトランザクション処理
- 分散データベース・アプリケーションの開発
- 分散環境でのキャラクタ・セットのサポート

分散データベース・アーキテクチャ

分散データベース・システムを使用すると、アプリケーションはローカル・データベースおよびリモート・データベースのデータにアクセスできます。**同機種間分散データベース・システム**では、個々のデータベースは Oracle データベースです。**異機種間分散データベース・システム**では、少なくともデータベースの 1 つは Oracle 以外のデータベースです。分散データベースは、**クライアント / サーバー・**アーキテクチャを使用して情報の要求を処理します。

この項の内容は、次のとおりです。

- 同機種間分散データベース・システム
- 異機種間分散データベース・システム
- クライアント / サーバー・データベース・アーキテクチャ

同機種間分散データベース・システム

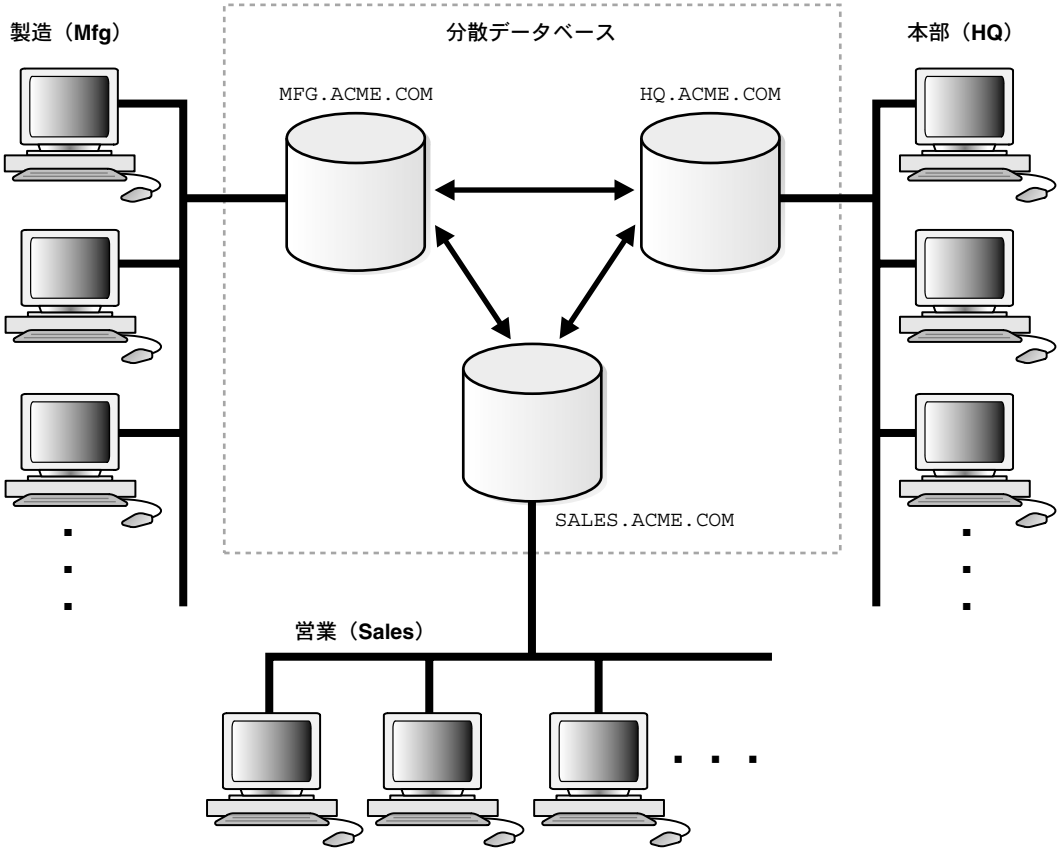
同機種間分散データベース・システムとは、1 台以上のマシンに存在する 2 つ以上の Oracle データベースのネットワークを表します。図 28-1 は、hq、mfg および sales の 3 つのデータベースを接続している分散システムを示しています。アプリケーションは、1 つの分散環境内にある複数のデータベースのデータに同時にアクセスしたり、データを同時に変更できます。たとえば、ローカル・データベース mfg の製造（Mfg）クライアントから発行する 1 回の問合せによって、ローカル・データベースの products 表のデータとリモート・データベース hq の dept 表のデータを結合したデータを取得できます。

クライアント・アプリケーションに対して、データベースの位置とプラットフォームは透過的です。また、分散システム内のリモート・オブジェクトに対してユーザーがローカル・オブジェクトと同じ構文を使用してアクセスできるように、リモート・オブジェクトの**シノニム**を作成することもできます。たとえば、mfg データベースに接続しているときにデータベース hq のデータにアクセスする場合は、リモートの dept 表に対するシノニムを mfg 上に作成することで、次の問合せを発行できます。

```
SELECT * FROM dept;
```

このように、分散システムではローカル・データベースと同様のデータ・アクセスが可能です。mfg のユーザーは、アクセスするデータがリモート・データベースに存在していることを知る必要はありません。

図 28-1　同機種間分散データベース



Oracle 分散データベース・システムは、異なるバージョンの Oracle データベースによって構成することが可能です。サポートされている Oracle のリリースは、すべて分散データベース・システムに参加できます。しかし、分散データベースを使用するアプリケーションでは、システムの各ノードで使用可能な機能を理解しておく必要があります。分散データベース・アプリケーションでは、Oracle9*i* でのみ使用可能な SQL の拡張を Oracle7 で実行できるかのような想定を行ってはなりません。

分散データベースと分散処理

分散データベースおよび**分散処理**という用語は密接に関連していますが、その意味は異なります。この 2 つの用語の定義は、次のとおりです。

- 分散データベース

  分散システムを構成する一連のデータベース。アプリケーションからは単一のデータ・ソースのように見えます。

- 分散処理

  アプリケーションが自身のタスクをネットワーク内の異なるコンピュータ間に分散するときに発生する操作。たとえば、データベース・アプリケーションは通常、フロントエンド側のプレゼンテーション・タスクをクライアント・コンピュータに配置し、バックエンド側のデータベース・サーバーでデータベースへの共有アクセスを管理できるようにします。このことから、分散データベース・アプリケーション処理システムは、一般にクライアント / サーバー・データベース・アプリケーション・システムとも呼ばれます。

Oracle 分散データベース・システムは、分散処理アーキテクチャを利用しています。たとえば、Oracle データベース・サーバーは、別の Oracle データベース・サーバーが管理しているデータを要求するときにクライアントとして動作します。

分散データベースとレプリケート・データベース

分散データベース・システムおよび**データベース・レプリケーション**という用語は関連していますが、その意味は異なります。**純粋な**（つまり、レプリケートされていない）分散データベースでは、すべてのデータとそれをサポートしているデータベース・オブジェクトの単一コピーが管理されています。通常、分散データベース・アプリケーションは、分散トランザクションを使用してローカルおよびリモートのデータにアクセスし、グローバル・データベースをリアルタイムで変更します。

注意： このマニュアルでは、純粋な分散データベースについてのみ説明しています。

レプリケーションという用語は、分散システムに属している複数のデータベースの間でデータベース・オブジェクトをコピーし、管理する操作を指します。レプリケーションは分散データベース・テクノロジに依存していますが、データベース・レプリケーションを使用すると、純粋な分散データベース環境では得られないような利点をアプリケーションが利用できます。

一般に、レプリケーションを使用すると代替のデータ・アクセス手段が提供されるので、ローカル・データベースのパフォーマンスが向上し、アプリケーションの可用性が保護されます。たとえば、アプリケーションは、ネットワークの通信量を最小限に抑えてパフォーマンスの最大化を図るために、リモート・サーバーではなくローカル・データベースにアクセ

スできます。また、ローカル・サーバーに障害が発生しても、レプリケートされたデータを持つ他のサーバーにアクセスできれば、アプリケーションは引き続き実行できます。

関連項目： Oracle のレプリケーション機能の詳細は、『Oracle9*i* レプリケーション』を参照してください。

異機種間分散データベース・システム

異機種間分散データベース・システムでは、少なくともデータベースの 1 つは Oracle 以外のシステムです。アプリケーションにとって、異機種間分散データベース・システムは 1 つのローカルな Oracle データベースのように見えます。ローカルの Oracle データベース・サーバーでは、データの分散と異機種性は隠されています。

Oracle データベース・サーバーは、Oracle 以外のシステムへのアクセスに Oracle 異機種間サービスと**エージェント**を組み合せて使用します。Oracle 以外のデータ・ストアに Oracle Transparent Gateway を使用してアクセスする場合、エージェントはシステム固有のアプリケーションになります。たとえば、Oracle 分散システムに Sybase データベースを含める場合は、分散システム内の Oracle データベースが Sybase データベースとやり取りできるように、Sybase 固有の Transparent Gateway を入手する必要があります。

Oracle 以外のシステムが ODBC または OLE DB プロトコルをサポートしている場合は、かわりに**一般接続**を使用して Oracle 以外のデータ・ストアにアクセスできます。

注意： このマニュアルで Oracle 異機種間サービスについて説明しているのは、この章の概要的な内容のみです。

異機種間サービスの詳細は、『Oracle9*i* Heterogeneous Connectivity Administrator's Guide』を参照してください。

異機種間サービス

異機種間サービスは Oracle データベース・サーバー内部に統合されたコンポーネントであり、現在の Oracle Transparent Gateway 製品群を実現するためのテクノロジです。異機種間サービスは、Oracle ゲートウェイ製品とその他の異機種間アクセス機能に対して共通のアーキテクチャと管理のメカニズムを提供します。また、以前にリリースされたほとんどの Oracle Transparent Gateway のユーザーに上位互換の機能を提供します。

Transparent Gateway エージェント

Oracle 以外の個々のシステムにアクセスする場合、異機種間サービスは Transparent Gateway エージェントを使用して、Oracle 以外の特定のシステムとのインタフェースの役目を果たします。エージェントは Oracle 以外のシステムに固有のものであり、システムのタイプごとに異なるエージェントが必要になります。

Transparent Gateway エージェントは、Oracle データベース・サーバー内の異機種間サービス・コンポーネントを使用して、Oracle データベースと Oracle 以外のデータベースとの間

のやり取りを容易にします。エージェントは Oracle データベース・サーバーのかわりに、Oracle 以外のシステムで SQL とトランザクション要求を実行します。

関連項目： Transparent Gateway の詳細は、オラクル社が提供するゲートウェイ固有のマニュアルを参照してください。

一般接続

一般接続を使用すると、異機種間サービス ODBC エージェントまたは異機種間サービス OLE DB エージェントのどちらかを使用して Oracle 以外のデータ・ストアに接続できます。これらはどちらも Oracle 製品に標準機能として組み込まれています。ODBC または OLE DB の規格と互換性があれば、どのようなデータ・ソースでも一般接続エージェントを使用してアクセスできます。

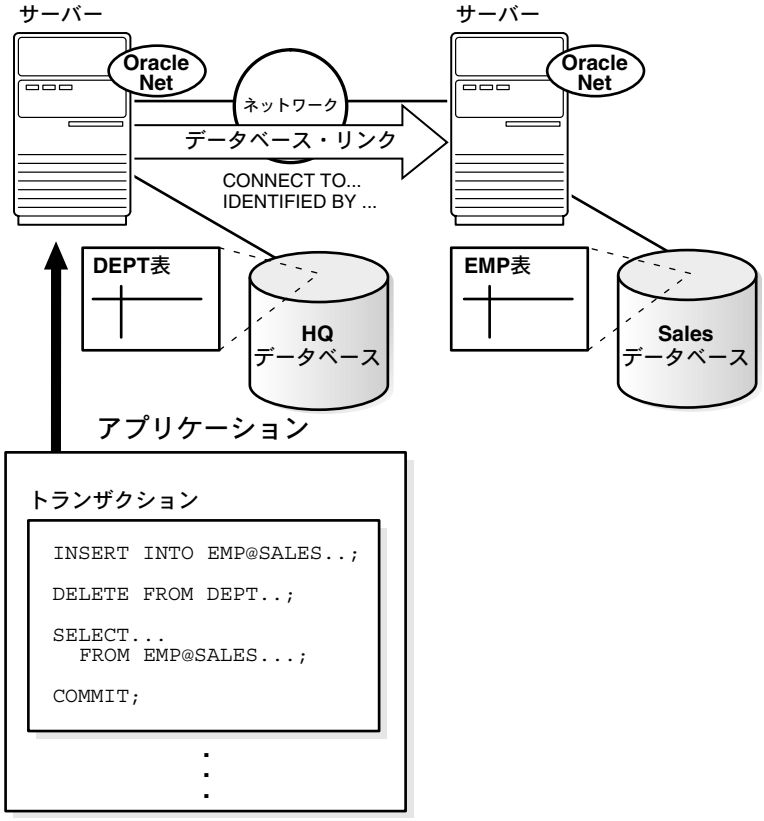
一般接続の利点は、必ずしもシステム固有のエージェントを別途購入して構成する必要がないことです。ODBC ドライバまたは OLE DB ドライバを使用して、エージェントとインタフェースできます。ただし、一部のデータ・アクセス機能は Transparent Gateway エージェントでしか利用できません。

クライアント / サーバー・データベース・アーキテクチャ

データベース・サーバーとはデータベースを管理する Oracle ソフトウェアのことであり、クライアントとはサーバーの情報を要求するアプリケーションのことです。ネットワーク内の各コンピュータはノードと呼ばれ、1 つ以上のデータベースのホストとなることができます。分散データベース・システム内の各ノードは、状況に応じてクライアント、サーバー、あるいはその両方として機能します。

図 28-2 の `hq` データベースのホストは、そのローカル・データに対して文が発行されたときはデータベース・サーバーとして機能します。たとえば、トランザクション内の 2 番目の文は、ローカルの `dept` 表に対して文を発行しています。しかし、リモート・データに対して文が発行されたときはクライアントとして機能します。たとえば、トランザクション内の最初の文は、`sales` データベース内のリモートの表 `emp` に対して発行されています。

図 28-2　Oracle 分散データベース・システム



クライアントは、データベース・サーバーに**直接**または**間接**に接続できます。直接接続となるのは、クライアントがサーバーに接続し、そのサーバー上に存在するデータベースの情報にアクセスするときです。たとえば、図 28-2 のように、hq データベースに接続してこのデータベースの dept 表にアクセスする場合は、次の文を発行できます。

```
SELECT * FROM dept;
```

この場合は、リモート・データベースのオブジェクトにアクセスしていないので、問合せは直接になります。

これに対して、間接接続となるのは、クライアントがサーバーに接続した後、別の異なるサーバー上のデータベースに格納されている情報にアクセスするときです。たとえば、図 28-2 のように、hq データベースに接続した後、リモートの sales データベースの emp 表にアクセスする場合は、次の文を発行できます。

```
SELECT * FROM emp@sales;
```

この場合は、アクセスしようとしているオブジェクトが直接接続しているデータベース上にないため、問合せは間接になります。

データベース・リンク

分散データベース・システムの中心的な概念となるのが、**データベース・リンク**です。データベース・リンクとは 2 つの物理的なデータベース・サーバー間の接続のことで、これにより、クライアントからそれらのサーバーに 1 つの論理データベースとしてアクセスできるようになります。

この項の内容は、次のとおりです。

- データベース・リンクの概要
- データベース・リンクを使用する理由
- データベース・リンク内のグローバル・データベース名
- データベース・リンクの名前
- データベース・リンクのタイプ
- データベース・リンクのユーザー
- データベース・リンクの作成 : 例
- スキーマ・オブジェクトとデータベース・リンク
- データベース・リンクの制限事項

データベース・リンクの概要

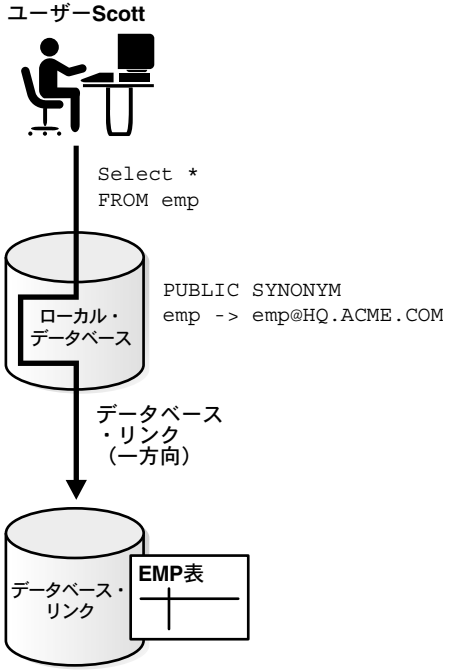
データベース・リンクは、ある Oracle データベース・サーバーから別のデータベース・サーバーへの一方向の通信経路を定義するポインタです。リンク・ポインタは、実際にはデータ・ディクショナリ表内のエントリとして定義されます。リンクにアクセスするには、データ・ディクショナリ・エントリのあるローカル・データベースに接続する必要があります。

データベース・リンク接続が一方向であるということは、たとえばローカル・データベース A に接続しているクライアントはデータベース A に格納されているリンクを使用してリモート・データベース B の情報にアクセスできるが、データベース B に接続しているユーザーは同じリンクを使用してデータベース A のデータにアクセスできないことを意味します。データベース B のローカル・ユーザーがデータベース A のデータにアクセスする場合には、データベース B のデータ・ディクショナリに格納されるリンクを定義する必要があります。

データベース・リンク接続を使用することで、ローカル・ユーザーはリモート・データベースのデータにアクセスできるようになります。この接続を実現するためには、分散システム内の各データベースがネットワーク・ドメイン内で一意の**グローバル・データベース名**を持っていることが必要です。グローバル・データベース名は、分散システム内のデータベース・サーバーを一意に識別します。

図 28-3 は、ユーザー scott がグローバル名 hq.acme.com を使用して、リモート・データベースの emp 表にアクセスしている例を示します。

図 28-3　データベース・リンク



データベース・リンクには、プライベートとパブリックの 2 種類があります。プライベートの場合は、そのリンクを作成したユーザーのみがアクセスできます。パブリックの場合は、すべてのデータベース・ユーザーがアクセスできます。

データベース・リンクにおける原理的な違いの 1 つに、リモート・データベースにどのように接続するかという点があります。ユーザーは、次のタイプのリンクによってリモート・データベースにアクセスします。

リンクのタイプ	説明
接続ユーザー・リンク	ユーザーはユーザー自身として接続します。つまり、ユーザーにはローカル・データベースのアカウントと同じユーザー名を持つリモート・データベースのアカウントが必要です。

リンクのタイプ	説明
固定ユーザー・リンク	ユーザーは、リンク内で参照されるユーザー名とパスワードを使用して接続します。たとえば、Jane がユーザー名およびパスワード scott/tiger を使用して hq データベースに接続する固定ユーザー・リンクを使用する場合、Jane は scott として接続し、scott に直接付与されている hq 内でのすべての権限と、hq データベースで scott に付与されているすべてのデフォルト・ロールを持ちます。
現行ユーザー・リンク	ユーザーは、グローバル・ユーザーとして接続します。ローカル・ユーザーは、ストアド・プロシージャのコンテキスト内ではグローバル・ユーザーとして接続できます。グローバル・ユーザーのパスワードをリンク定義に格納する必要はありません。たとえば、Jane は Scott が記述したプロシージャにアクセスでき、hq データベース上の Scott のアカウントと Scott のスキーマにアクセスできます。現行ユーザー・リンクは、Oracle Advanced Security の機能の 1 つです。

データベース・リンクを作成するには、`CREATE DATABASE LINK` 文を使用します。リンクを作成した後、SQL 文の中でリンクを使用してスキーマ・オブジェクトを指定できます。

関連項目：

- `CREATE DATABASE` 文の構文は、『Oracle9*i* SQL リファレンス』を参照してください。
- Oracle Advanced Security の詳細は、『Oracle Advanced Security 管理者ガイド』を参照してください。

共有データベース・リンクの概要

共有データベース・リンクとは、ローカル・サーバー・プロセスとリモート・データベースの間のリンクのことです。複数のクライアント・プロセスが同じリンクを同時に使用できるので、共有データベース・リンクと呼ばれます。

ローカル・データベースがデータベース・リンクを介してリモート・データベースに接続するとき、どちらのデータベースも専用サーバー・モードまたは共有サーバー・モードのいずれかで稼働しています。可能な組合せを次の表に示します。

ローカル・データベースのモード	リモート・データベースのモード
専用	専用
専用	共有サーバー
共有サーバー	専用
共有サーバー	共有サーバー

共有データベース・リンクは、これら 4 つのどの構成でも確立できます。共有リンクは、通常のデータベース・リンクと次の点で異なります。

- データベース・リンクを介して同じスキーマ・オブジェクトにアクセスする異なるユーザーの間で、ネットワーク接続を共有できます。
- ユーザーが特定のサーバー・プロセスからリモート・サーバーに接続を確立するときに、そのプロセスからリモート・サーバーに対してすでに確立されている接続を再利用できます。接続を再利用するには、その接続が同じサーバー・プロセスから同じデータベース・リンクを使用して確立されている必要があります。セッションは異なっていてもかまいません。非共有のデータベース・リンクでは、複数のセッション間で接続が共有されることはありません。
- 共有サーバー構成で共有データベース・リンクを使用すると、ネットワーク接続はローカル・サーバーの共有サーバー・プロセスの外部で直接に確立されます。ローカル共有サーバーでの非共有のデータベース・リンクでは、この接続はローカル・ディスパッチャを介して確立されるため、ローカル・ディスパッチャのためのコンテキスト・スイッチングが発生し、データがディスパッチャを経由することになります。

関連項目： 共有サーバーの詳細は、『Oracle9*i* Net Services 管理者ガイド』を参照してください。

データベース・リンクを使用する理由

データベース・リンクの大きな利点として、ユーザーがリモート・データベースにある他のユーザーのオブジェクトに対して、オブジェクトの所有者が持つ権限の制限内でアクセスできるという点があります。つまり、ローカル・ユーザーはリモート・データベースのユーザーにならなくても、リモート・データベースへのリンクにアクセスできます。

たとえば、従業員が経費精算書を買掛管理（A/P）アプリケーションに送信し、さらに A/P アプリケーションを使用するユーザーが hq データベースから従業員に関する情報を取得する必要があるとします。A/P のユーザーは、必要な情報を取得するために、hq データベースに接続してリモートの hq データベース内のストアド・プロシージャを実行できます。A/P のユーザーは、自分のジョブを実行するために hq データベースのユーザーになる必要はありません。プロシージャに記述されている制御された方法で hq の情報にアクセスできればそれで済みます。

データベース・リンクを使用すると、リモート・データベースへの制限されたアクセス権限をローカル・ユーザーに付与できます。現行ユーザー・リンクを使用すると、集中管理されたグローバル・ユーザーを作成できます。グローバル情報のパスワード情報は、管理者と管理者以外のユーザーのどちらからも参照できません。たとえば、A/P ユーザーは hq データベースに `scott` としてアクセスできますが、固定ユーザー・リンクとは異なり、`scott` の資格証明は格納されないので、データベース・ユーザーは資格証明を参照できません。

固定ユーザー・リンクを使用すると、非グローバル・ユーザーを作成できます。非グローバル・ユーザーのパスワード情報は、暗号化されない形式で `LINK$` データ・ディクショナリ表に格納されます。固定ユーザー・リンクは作成が簡単であり、必要なオーバーヘッドも少

なくて済みます。これは、SSL やディレクトリを必要としないためですが、パスワード情報がデータ・ディクショナリに格納されるので、セキュリティ上の危険性があります。

関連項目：

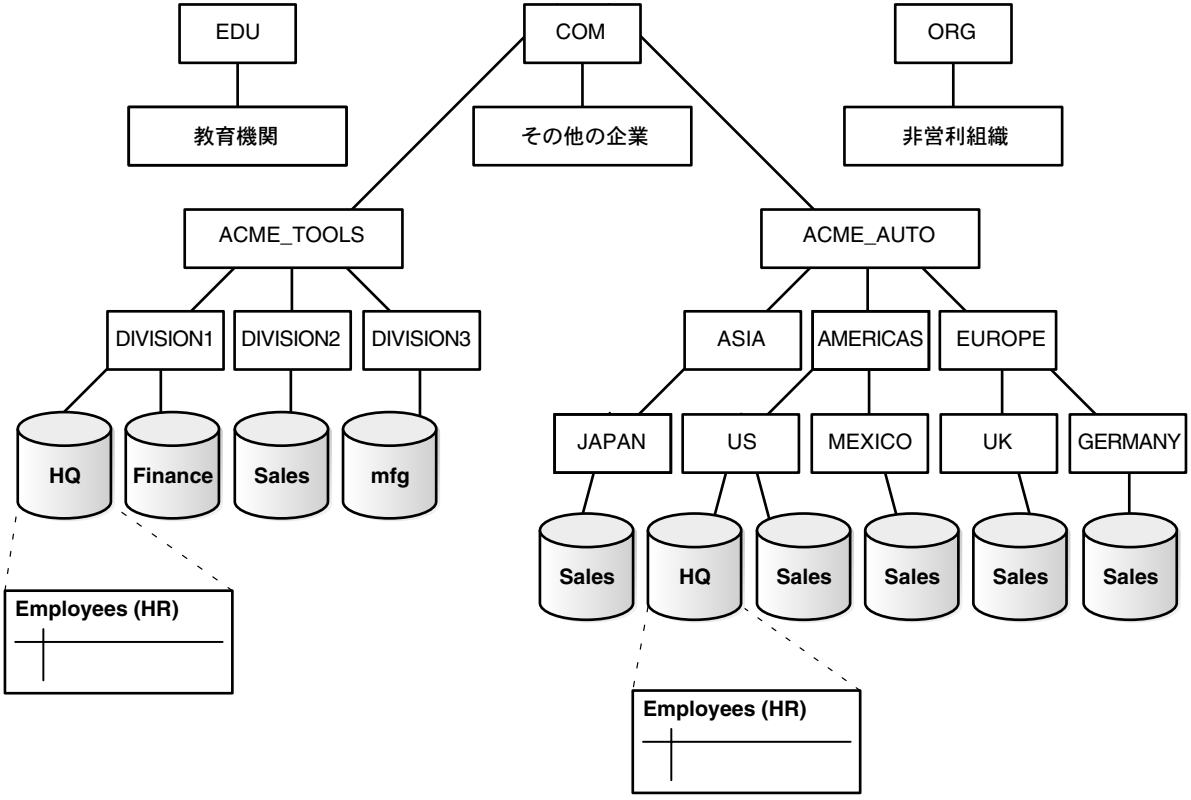
- データベース・リンク・ユーザーの詳細は、28-16 ページの「データベース・リンクのユーザー」を参照してください。
- パスワードを管理者以外のユーザーから隠す方法の詳細は、「データベース・リンク情報の表示」を参照してください。

データベース・リンク内のグローバル・データベース名

データベース・リンクの動作の仕組みを理解するには、まずグローバル・データベース名について理解する必要があります。分散データベース内の各データベースは、それぞれのグローバル・データベース名によって一意に識別されます。Oracle データベースのグローバル・データベース名は、データベースのネットワーク・ドメインの前に個々のデータベース名を連結して構成されます。データベースのネットワーク・ドメインは、データベースの作成時に DB_DOMAIN 初期化パラメータによって指定し、個々のデータベース名は DB_NAME 初期化パラメータによって指定します。

たとえば、図 28-4 では、ネットワーク全体のデータベースを階層的な配置で表しています。

図 28-4　ネットワーク化されたデータベースの階層的な配置



データベースの名前は、ツリーのリーフ（葉）から始まってルート（根）に至るまでの経路によって形成されます。たとえば、mfg データベースは、com ドメインの acme_tools ブランチの division3 にあります。mfg のグローバル・データベース名は、ツリー内のノードを次のように連結して作成されます。

- `mfg.division3.acme_tools.com`

複数のデータベースが 1 つの個別名を共有できますが、各データベースのグローバル・データベース名は一意にする必要があります。たとえば、ネットワーク・ドメイン us.americas.acme_auto.com と uk.europe.acme_auto.com には、どちらにも sales データベースがあります。グローバル・データベース名の命名体系では、americas 部門の sales データベースと europe 部門の sales データベースが次のように区別されます。

- sales.us.americas.acme_auto.com
- sales.uk.europe.acme_auto.com

関連項目： グローバル・データベース名を指定および変更する方法の詳細は、29-2 ページの「分散システムでのグローバル名の管理」を参照してください。

データベース・リンクの名前

通常、データベース・リンクの名前は、参照先のリモート・データベースのグローバル・データベース名と同じです。たとえば、データベースのグローバル・データベース名が sales.us.oracle.com であれば、データベース・リンクの名前も sales.us.oracle.com になります。

初期化パラメータ GLOBAL_NAMES を TRUE に設定すると、データベース・リンクの名前がリモート・データベースのグローバル・データベース名と同じになることが保証されます。たとえば、hq のグローバル・データベース名が hq.acme.com で、GLOBAL_NAMES が TRUE の場合は、リンク名も必ず hq.acme.com になります。Oracle は、初期化パラメータ・ファイル内の DB_DOMAIN の設定値ではなく、データ・ディクショナリに格納されているグローバル・データベース名のドメイン部分を調べます（29-4 ページの「グローバル・データベース名のドメインの変更」を参照）。

初期化パラメータ GLOBAL_NAMES を FALSE に設定した場合は、グローバル・ネーミングを使用する必要はありません。データベース・リンクに自由に名前を付けられます。たとえば、hq.acme.com へのデータベース・リンクに foo という名前を付けることができます。

注意： グローバル・ネーミングはレプリケーションなどの多数の機能で必要になるため、グローバル・ネーミングの使用をお薦めします。

グローバル・ネーミングを有効にすると、データベース・リンクの名前がリンクの参照先であるデータベースのグローバル名と同じになるため、データベース・リンクは本質的に分散データベースのユーザーに対して透過的になります。たとえば、次の文は、リモート・データベース sales へのデータベース・リンクをローカル・データベース内に作成します。

```
CREATE PUBLIC DATABASE LINK sales.division3.acme.com USING 'sales1';
```

関連項目： 初期化パラメータ GLOBAL_NAMES の指定の詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

データベース・リンクのタイプ

Oracle では、**プライベート**、**パブリック**および**グローバル**のデータベース・リンクを作成できます。これらの基本的なリンク・タイプは、どのユーザーに対してリモート・データベースへのアクセスが許可されているかによって異なります。

タイプ	所有者	説明
プライベート	リンクを作成したユーザー。次のビューによって所有権データを表示できます。 ■ DBA_DB_LINKS ■ ALL_DB_LINKS ■ USER_DB_LINKS	ローカル・データベースの特定のスキーマにリンクを作成します。プライベート・データベース・リンクの所有者またはスキーマ内の PL/SQL サブプログラムのみが、このリンクを使用して、対応するリモート・データベース内のデータベース・オブジェクトにアクセスできます。
パブリック	PUBLIC と呼ばれるユーザー。プライベートと同じビューによって所有権データを表示できます。	データベース全体にわたるリンクを作成します。データベース内のすべてのユーザーと PL/SQL サブプログラムが、パブリック・リンクを使用して、対応するリモート・データベース内のデータベース・オブジェクトにアクセスできます。
グローバル	PUBLIC と呼ばれるユーザー。プライベートと同じビューによって所有権データを表示できます。	ネットワーク全体にわたるリンクを作成します。Oracle ネットワークが Oracle Names を使用する場合は、システム内の名前サーバーによって、ネットワーク上に存在するすべての Oracle データベースのグローバル・データベース・リンクが自動的に作成および管理されます。どのデータベースのユーザーと PL/SQL サブプログラムでも、グローバル・リンクを使用して、対応するリモート・データベース内のオブジェクトにアクセスできます。

分散データベースで利用するデータベース・リンクのタイプは、システムを使用しているアプリケーションの仕様要件によって決まります。リンクの選択時には、次の機能を考慮してください。

リンクのタイプ	機能
プライベート・ データベース・リンク	このリンクは、パブリック・リンクやグローバル・リンクよりも安全です。その理由は、このリンクを使用してリモート・データベースにアクセスできるのが、プライベート・リンクの所有者と、同じスキーマ内のサブプログラムのみに限定されるためです。
パブリック・ データベース・リンク	多数のユーザーがリモートの Oracle データベースへのアクセス・パスを必要とするときは、データベース内のすべてのユーザーが使用できる 1 つのパブリック・データベース・リンクを作成できます。
グローバル・ データベース・リンク	Oracle ネットワークが Oracle Names を使用している場合、管理者はシステム内に存在するすべてのデータベースのグローバル・データベース・リンクを管理できるので便利です。データベース・リンクの管理が集中化され、単純になります。

関連項目：

- 各タイプのデータベース・リンクの作成方法の詳細は、29-9 ページの「リンク・タイプの指定」を参照してください。
- リンクに関する情報へのアクセス方法の詳細は、29-22 ページの「データベース・リンク情報の表示」を参照してください。

データベース・リンクのユーザー

リンクを作成するときは、どのユーザーがリモート・データベースに接続してデータにアクセスする必要があるかを決めます。次の表は、データベース・リンクに関係するユーザーのカテゴリについて、それらの違いを説明したものです。

ユーザー・ タイプ	意味	リンク作成構文の サンプル
接続ユーザー	固定のユーザー名およびパスワードが指定されていないデータベース・リンクにアクセスするローカル・ユーザー。SYSTEM が問合せ発行時にパブリック・リンクにアクセスする場合、接続ユーザーは SYSTEM であり、Oracle はリモート・データベースの SYSTEM スキーマに接続します。 **注意：**接続ユーザーは、必ずしもリンクを作成したユーザーである必要はありません。リンクにアクセスしようとしているすべてのユーザーが接続ユーザーになります。	`CREATE PUBLIC DATABASE LINK hq USING 'hq';`

ユーザー・タイプ	意味	リンク作成構文のサンプル
現行ユーザー	CURRENT_USER データベース・リンク内のグローバル・ユーザー。グローバル・ユーザーは、X.509 証明書（SSL 認証方式のエンタープライズ・ユーザー）またはパスワード（パスワード認証方式のエンタープライズ・ユーザー）によって認証される必要があり、リンクに関係する両方のデータベースのユーザーであることが必要です。現行ユーザー・リンクは、Oracle Advanced Security オプションの機能の 1 つです。 グローバル・セキュリティの詳細は、『Oracle Advanced Security 管理者ガイド』を参照してください。	CREATE PUBLIC DATABASE LINK hq CONNECT TO CURRENT_USER using 'hq';
固定ユーザー	ユーザー名 / パスワードがリンク定義の一部になっているユーザー。リンクに固定ユーザーが含まれている場合は、その固定ユーザーのユーザー名とパスワードがリモート・データベースへの接続に使用されます。	CREATE PUBLIC DATABASE LINK hq CONNECT TO jane IDENTIFIED BY doe USING 'hq';

関連項目： リンクを作成するユーザーを指定する方法の詳細は、29-11 ページの「リンク・ユーザーの指定」を参照してください。

接続ユーザー・データベース・リンク

接続ユーザー・リンクには、接続文字列が対応付けられていません。接続ユーザー・リンクの利点は、リンクを参照するユーザーが同じユーザーとしてリモート・データベースに接続することです。また、リンクに対応付けられた接続文字列がないため、データ・ディクショナリにパスワードが平文で格納されることもありません。

接続ユーザー・リンクには、いくつかの欠点があります。これらのリンクでは、ユーザーが接続先のリモート・データベースのアカウントと権限を持っている必要があるため、管理者の権限管理作業が増えます。また、必要以上の権限をユーザーに与えることは、セキュリティの基本概念である最低限の権限、つまり「ユーザーにはユーザー自身のジョブの実行に必要な権限のみを与える」に反することになります。

接続ユーザー・データベース・リンクの使用許可はいくつかの要因によって左右されますが、その中で最も重要なものは認証方式、つまりユーザーがパスワードを使用して Oracle によって認証されるのか、あるいはオペレーティング・システムまたはネットワーク認証サービスによって外部認証されるのかということです。ユーザーが外部認証される場合、接続ユーザー・データベース・リンクの使用許可は、リモート・データベースがユーザーのリモート認証を承認するかどうかという要因によっても左右されます。このリモート認証は、REMOTE_OS_AUTHENT 初期化パラメータによって設定します。

REMOTE_OS_AUTHENT パラメータは、次のように働きます。

REMOTE_OS_AUTHENT の値	動作
リモート・データベースに対して TRUE	外部認証方式のユーザーは、接続ユーザー・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できます。
リモート・データベースに対して FALSE	外部認証方式のユーザーは、Oracle Advanced Security オプションでサポートされているセキュリティ保護されたプロトコルまたはネットワーク認証サービスを使用しないかぎり、接続ユーザー・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ことはできません。

固定ユーザー・データベース・リンク

固定ユーザー・リンクの利点は、接続文字列で指定したユーザーのセキュリティ・コンテキストを使用して、プライマリ・データベースのユーザーをリモート・データベースに接続することです。たとえば、ローカル・ユーザー joe は、固定ユーザー scott とパスワード tiger を指定したパブリック・データベース・リンクを joe のスキーマ内に作成できます。jane がこの固定ユーザー・リンクを使用して問合せを発行すると、ローカル・データベースのユーザーである jane が、scott/tiger としてリモート・データベースに接続します。

固定ユーザー・リンクでは、接続文字列にユーザー名とパスワードが対応付けられています。ユーザー名とパスワードは、暗号化されない形式でデータ・ディクショナリの LINK$ 表に格納されます。

注意： ユーザー名とパスワードが暗号化されない形式でデータ・ディクショナリに格納されるということは、固定ユーザー・データベース・リンクにセキュリティ上の潜在的な弱点があることを意味しています。

O7_DICTIONARY_ACCESSIBILITY 初期化パラメータを TRUE に設定した場合、SELECT ANY TABLE システム権限を持つユーザーはデータ・ディクショナリにアクセスできるので、固定ユーザーの認証が危険にさらされます。

O7_DICTIONARY_ACCESSIBILITY 初期化パラメータのデフォルトは FALSE です。

このセキュリティ上の問題について、次のような例を考えてみます。jane は hq データベースに scott/tiger として接続するプライベート・リンクを使用する権限を持っていませんが、O7_DICTIONARY_ACCESSIBILITY 初期化パラメータが TRUE に設定されているデータベースに対して SELECT ANY TABLE 権限を持っているとします。jane は LINK$ に対して SELECT 文を実行し、hq への接続文字列が scott/tiger であることを知ります。jane が hq の存在しているホストのアカウントを持っている場合、jane はホストに接続し、パスワード tiger を使用して scott として hq に接続できます。jane がローカルに接続すると、scott のすべての権限を持つことになり、監査レコードには、jane があたかも scott であるかのように記録されます。

> **関連項目：** システム権限と O7_DICTIONARY_ACCESSIBILITY 初期化パラメータの詳細は、25-2 ページの「システム権限」を参照してください。

現行ユーザー・データベース・リンク

現行ユーザー・データベース・リンクは、グローバル・ユーザーを利用します。グローバル・ユーザーは、X.509 証明書またはパスワードによって認証される必要があり、リンクに関係する両方のデータベースのユーザーであることが必要です。

CURRENT_USER リンクを実行するユーザーは、必ずしもグローバル・ユーザーである必要はありません。たとえば、jane が買掛管理データベースへのパスワードによって（グローバル・ユーザーとしてではなく）認証される場合、jane はストアド・プロシージャにアクセスして、hq データベースからデータを取得できます。プロシージャは現行ユーザー・データベース・リンクを使用しますが、このリンクによって jane はグローバル・ユーザー scott として hq に接続します。ユーザー scott はグローバル・ユーザーであり、SSL 上の証明書を介して認証されますが、jane はグローバル・ユーザーではなく、SSL では認証されません。

現行ユーザー・データベース・リンクによって、次のような結果になることに注意してください。

- 現行ユーザー・データベース・リンクに対してストアド・オブジェクトの内部以外からアクセスしている場合、現行ユーザーは、リンクにアクセスしている接続ユーザーと同じになります。たとえば、scott が現行ユーザー・リンクを介して SELECT 文を発行した場合、現行ユーザーは scott になります。
- データベース・リンクにアクセスするプロシージャ、ビュー、トリガーなどのストアド・オブジェクトを実行すると、現行ユーザーは、そのストアド・オブジェクトをコールしたユーザーではなく、オブジェクトを所有しているユーザーになります。たとえば、jane がプロシージャ scott.p（scott によって作成されたプロシージャ）をコールし、コールされたプロシージャの内部から現行ユーザー・リンクが確立される場合、リンクの現行ユーザーは scott になります。
- ストアド・オブジェクトが実行者権限のファンクション、プロシージャまたはパッケージである場合は、実行者の認可 ID を使用してリモート・ユーザーとして接続します。たとえば、ユーザー jane がプロシージャ scott.p（scott によって作成された実行

者権限のプロシージャ）をコールし、プロシージャ scott.p の内部からリンクが確立される場合、リンクの現行ユーザーは jane になります。

- データベースにエンタープライズ・ユーザーとして接続して、共有のグローバル・スキーマ内に存在するストアド・プロシージャで現行ユーザー・リンクを使用することはできません。たとえば、ユーザー jane がデータベース hq 上の共有スキーマ guest 内のストアド・プロシージャにアクセスしている場合、jane は、このスキーマ内で現行ユーザー・リンクを使用してリモート・データベースにログインすることはできません。

関連項目：

- データベース・リンクに関連するセキュリティ上の問題の詳細は、28-25 ページの「分散データベースのセキュリティ」を参照してください。
- 『Oracle Advanced Security 管理者ガイド』

データベース・リンクの作成：例

データベース・リンクを作成するには、CREATE DATABASE LINK 文を使用します。次の表は、ローカル・データベース内でリモートの sales.us.americas.acme_auto.com データベースへのデータベース・リンクを作成する SQL 文の例です。

SQL 文	接続先のデータベース	接続時のユーザー	リンク・タイプ
CREATE DATABASE LINK sales.us.americas.acme_auto.com USING 'sales_us';	sales（ネット・サービス名 sales_us を使用）	接続ユーザー	プライベート接続ユーザー
CREATE DATABASE LINK foo CONNECT TO CURRENT_USER USING 'am_sls';	sales（サービス名 am_sls を使用）	現行グローバル・ユーザー	プライベート現行ユーザー
CREATE DATABASE LINK sales.us.americas.acme_auto.com CONNECT TO scott IDENTIFIED BY tiger USING 'sales_us';	sales（ネット・サービス名 sales_us を使用）	scott（パスワード tiger を使用）	プライベート固定ユーザー
CREATE PUBLIC DATABASE LINK sales CONNECT TO scott IDENTIFIED BY tiger USING 'rev';	sales（ネット・サービス名 rev を使用）	scott（パスワード tiger を使用）	パブリック固定ユーザー
CREATE SHARED PUBLIC DATABASE LINK sales.us.americas.acme_auto.com CONNECT TO scott IDENTIFIED BY tiger AUTHENTICATED BY anupam IDENTIFIED BY bhide USING 'sales';	sales（ネット・サービス名 sales を使用）	scott（パスワード tiger を使用。認証用としてユーザー名 anupam とパスワード bhide を使用）	共有パブリック固定ユーザー

関連項目：

- リンクの作成方法の詳細は、29-8 ページの「データベース・リンクの作成」を参照してください。
- CREATE DATABASE LINK 文の構文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

スキーマ・オブジェクトとデータベース・リンク

データベース・リンクを作成した後、リモート・データベースのオブジェクトにアクセスする SQL 文を実行できます。たとえば、データベース・リンク foo を使用してリモート・オブジェクト emp にアクセスするには、次の文を発行します。

```
SELECT * FROM emp@foo;
```

特定のリモート・オブジェクトにアクセスするには、リモート・データベース内での認可も必要です。

データベース・リンクを使用して適切な構成のオブジェクト名を作成することは、分散システムにおいて欠かせないデータ操作の 1 つです。

データベース・リンクを使用したスキーマ・オブジェクトの命名

Oracle は、グローバル・データベース名を使用してスキーマ・オブジェクトにグローバルな名前を付ける際、次のスキームを使用します。

schema.schema_object@global_database_name

各項目の意味は次のとおりです。

- *schema* は、データの論理構造（スキーマ・オブジェクト）の集まりです。スキーマはデータベース・ユーザーによって所有され、そのユーザーと同じ名前を持ちます。ユーザーはそれぞれ 1 つのスキーマを所有します。
- *schema_object* は、表、索引、ビュー、シノニム、プロシージャ、パッケージ、データベース・リンクなどの論理データ構造です。
- *global_database_name* は、リモート・データベースを一意に識別する名前です。この名前は、リモート・データベースの初期化パラメータ DB_NAME および DB_DOMAIN を連結したものと同じである必要があります。ただし、パラメータ GLOBAL_NAMES を FALSE に設定している場合は、任意の名前を使用できます。

たとえば、ユーザーまたはアプリケーションは、データベース sales.division3.acme.com へのデータベース・リンクを次のように使用して、リモート・データを参照できます。

```
SELECT * FROM scott.emp@sales.division3.acme.com;  # emp table in scott's schema
SELECT loc FROM scott.dept@sales.division3.acme.com;
```

GLOBAL_NAMES を FALSE に設定している場合は、sales.division3.acme.com へのリンクに対して任意の名前を使用できます。たとえば、リンク foo を次のようにコールして、リモート・データベースにアクセスできます。

```
SELECT name FROM scott.emp@foo;  # link name different from global name
```

リモート・スキーマ・オブジェクトへのアクセスに必要な認可

リモート・スキーマ・オブジェクトにアクセスするには、リモート・データベース内でそのオブジェクトへのアクセス権を付与される必要があります。また、リモート・オブジェクトの更新、挿入または削除を実行するには、オブジェクトに対する SELECT 権限と、UPDATE、INSERT または DELETE 権限を付与される必要があります。Oracle にはリモート記述機能がないため、ローカル・オブジェクトにアクセスする場合とは異なり、リモート・オブジェクトにアクセスするには SELECT 権限が必要です。Oracle は、構造を判断するためにリモート・オブジェクトの SELECT * を実行する必要があります。

スキーマ・オブジェクトのシノニム

Oracle では、シノニムを作成して、データベース・リンク名をユーザーから隠すことができます。シノニムを使用すると、ローカル・データベースの表にアクセスする場合と同じ構文を使用して、リモート・データベースの表にアクセスできます。たとえば、リモート・データベース内の表に対して次の問合せを発行するとします。

```
SELECT * FROM emp@hq.acme.com;
```

この場合、emp@hq.acme.com に対応するシノニム emp を作成すれば、同じデータにアクセスする際に次の問合せを発行できます。

```
SELECT * FROM emp;
```

> **関連項目：** データベース・リンクを使用して指定したオブジェクトのシノニムを作成する方法の詳細は、29-29 ページの「シノニムを使用した位置の透過性の作成」を参照してください。

スキーマ・オブジェクトの名前解決

スキーマ・オブジェクトへのアプリケーション参照を解決する（このプロセスを**名前解決**と呼びます）ために、Oracle はオブジェクト名を階層的に構成しています。たとえば、Oracle では、データベース内部の各スキーマは一意の名前を持ち、スキーマ内部では各オブジェクトが一意の名前を持つことが保証されています。そのため、スキーマ・オブジェクトの名前はデータベースの内部で常に一意になります。また、Oracle はオブジェクトのローカル名へのアプリケーション参照を解決します。

分散データベースでは、表などのスキーマ・オブジェクトに対してシステム内のすべてのアプリケーションからアクセスが可能です。Oracle は、グローバル・データベース名による階層ネーミング・モデルを拡張して**グローバル・オブジェクト名**を効果的に作成することにより、分散データベース・システム内のスキーマ・オブジェクトへの参照を解決します。たと

えば、問合せでリモートの表を参照する場合は、その完全修飾名（表が存在しているデータベースを含む）を指定します。

たとえば、ローカル・データベースにユーザー SYSTEM として接続するとします。

```
CONNECT SYSTEM/password@sales1
```

次に、データベース・リンク hq.acme.com を使用して次の文を発行し、リモート・データベース hq の scott スキーマおよび jane スキーマにあるオブジェクトにアクセスします。

```
SELECT * FROM scott.emp@hq.acme.com;
INSERT INTO jane.accounts@hq.acme.com (acc_no, acc_name, balance)
  VALUES (5001, 'BOWER', 2000);
UPDATE jane.accounts@hq.acme.com
  SET balance = balance + 500;
DELETE FROM jane.accounts@hq.acme.com
  WHERE acc_name = 'BOWER';
```

データベース・リンクの制限事項

データベース・リンクを使用して次の操作を実行することはできません。

- リモート・オブジェクトへの権限の付与。
- 一部のリモート・オブジェクトに対する DESCRIBE の実行。ただし、次のリモート・オブジェクトは DESCRIBE をサポートしています。
  - 表
  - ビュー
  - プロシージャ
  - ファンクション
- リモート・オブジェクトの分析。
- 参照整合性の定義または規定。
- リモート・データベース内のユーザーへのロールの付与。
- リモート・データベースにおけるデフォルト以外のロールの取得。たとえば、jane がローカル・データベースに接続し、scott として接続する固定ユーザー・リンクを使用したストアド・プロシージャを実行する場合、jane はリモート・データベースにおける scott のデフォルト・ロールを受け取ります。jane は、SET ROLE を発行してデフォルト以外のロールを取得することはできません。
- 共有サーバー接続を使用したハッシュ結合による問合せの実行。
- SSL、パスワードまたは Windows NT のシステム固有の認証によって認証されていない現行ユーザー・リンクの使用。

分散データベースの管理

ここでは、Oracle 分散データベース・システムにおけるデータベース管理について説明します。この項の内容は、次のとおりです。

- サイト自律性
- 分散データベースのセキュリティ
- データベース・リンクの監査
- 管理ツール

関連項目：

- 同機種システムの管理方法の詳細は、第 29 章「分散データベースの管理」を参照してください。
- 異機種間サービスの概念の詳細は、『Oracle9*i* Heterogeneous Connectivity Administrator's Guide』を参照してください。

サイト自律性

サイト自律性とは、分散データベース内の各サーバーが他のすべてのデータベースから独立して管理されることを意味します。複数のデータベースが協調して動作する場合もありますが、各データベースはそれぞれ別々のデータ・リポジトリであり、個別に管理されます。Oracle 分散データベースのサイト自律性には、次のような利点があります。

- 独立性を維持する必要がある個々の企業またはグループの論理的な組織構造を、システムのノード群として反映できます。
- ローカル管理者がそれぞれ対応するローカル・データを管理します。したがって、個々のデータベース管理者（DBA）の責任範囲が小さくなり、管理しやすくなります。
- 障害が単独で発生するため、分散データベースの他のノードに損傷を与える可能性が低くなります。1 つのデータベース障害によってすべての分散操作が停止したり、パフォーマンスのボトルネックが生じることはありません。
- 孤立したシステム障害が発生した際、管理者は、システム内の他のノードとは独立してリカバリできます。
- データ・ディクショナリがローカル・データベースごとに存在するため、ローカル・データへのアクセスにグローバル・カタログが必要になることがありません。
- 各ノードで独立してソフトウェアをアップグレードできます。

Oracle では分散データベース・システム内の各データベースを独立して管理できますが、システムのグローバルな要件を無視することのないようにしてください。たとえば、次のような作業が必要になることがあります。

- サーバー間接続を容易にするために作成したリンクをサポートするため、追加のユーザー・アカウントを各データベースに作成する。
- COMMIT_POINT_STRENGTH や OPEN_LINKS などの追加の初期化パラメータを設定する。

分散データベースのセキュリティ

Oracle では、非分散データベース環境で使用できる次のセキュリティ機能が、分散データベース・システムでもすべてサポートされています。

- ユーザーおよびロールのパスワード認証。
- ユーザーおよびロールに対する一部の外部認証タイプ。サポートされている外部認証タイプは、次のとおりです。
  - Kerberos バージョン 5（接続ユーザー・リンク用）
  - DCE（接続ユーザー・リンク用）
- クライアント / サーバー間およびサーバー間接続におけるログイン・パケットの暗号化。

次の項では、Oracle 分散データベース・システムの構成時に考慮すべき内容について説明します。

- データベース・リンクを介した認証
- パスワードなしの認証
- ユーザー・アカウントおよびロールのサポート
- ユーザーと権限の集中管理
- データの暗号化

関連項目： 外部認証の詳細は、『Oracle Advanced Security 管理者ガイド』を参照してください。

データベース・リンクを介した認証

データベース・リンクにはプライベートとパブリックの 2 種類があり、それぞれについて**認証ありの**場合と**認証なしの**場合があります。パブリック・リンクを作成するには、リンク作成文で PUBLIC キーワードを指定します。たとえば、次の文を発行できます。

```
CREATE PUBLIC DATABASE LINK foo USING 'sales';
```

認証ありのリンクを作成するには、データベース・リンク作成文で CONNECT TO 句、AUTHENTICATED BY 句、あるいはこれら両方の句を指定します。たとえば、次の文を発行できます。

```
CREATE DATABASE LINK sales CONNECT TO scott IDENTIFIED BY tiger USING 'sales';
CREATE SHARED PUBLIC DATABASE LINK sales CONNECT TO mick IDENTIFIED BY jagger
     AUTHENTICATED BY david IDENTIFIED BY bowie USING 'sales';
```

次の表は、ユーザーがリンクを介してリモート・データベースにアクセスする方法を示しています。

リンク・タイプ	認証	セキュリティ・アクセス
プライベート	認証なし	Oracle は、リモート・データベースに接続するときに、ローカル・セッションから取得したセキュリティ情報（ユーザー ID/ パスワード）を使用します。そのため、このリンクは接続ユーザー・データベース・リンクです。2 つのデータベース間で、パスワードが同期化されている必要があります。
プライベート	認証あり	ユーザー ID/ パスワードがローカル・セッションのコンテキストからではなく、リンク定義から取得されます。そのため、このリンクは固定ユーザー・データベース・リンクです。 この構成では、2 つのデータベース間で異なるパスワードを使用できますが、ローカル・データベース・リンクのパスワードはリモート・データベースのパスワードと一致している必要があります。パスワードはローカル・システム・カタログに平文で格納されるので、セキュリティの危険性が高まります。
パブリック	認証なし	動作はプライベートの非認証リンクとほぼ同じですが、すべてのユーザーがこのリモート・データベースへのポインタを参照できる点が異なります。
パブリック	認証あり	ローカル・データベースのすべてのユーザーがリモート・データベースにアクセスでき、すべてのユーザーが同じユーザー ID/ パスワードを使用して接続します。また、パスワードはローカル・カタログに平文で格納されるので、ローカル・データベースで必要な権限を持っていると、そのパスワードを参照できてしまいます。

パスワードなしの認証

接続ユーザーまたは現行ユーザーのデータベース・リンクを使用するときは、Kerberos などの外部認証ソースを使用して**エンド・トゥ・エンドのセキュリティ**を確立できます。エンド・トゥ・エンド認証では、資格証明がサーバー間で渡され、同じドメインに属するデータベース・サーバーによって資格証明を認証できます。たとえば、jane がローカル・データベースで外部認証され、接続ユーザー・リンクを使用して jane としてリモート・データベースに接続しようとする場合、ローカル・サーバーはリモート・データベースにセキュリティ・チケットを渡します。

ユーザー・アカウントおよびロールのサポート

分散データベース・システムでは、システムを使用するアプリケーションのサポートに必要なユーザー・アカウントおよびロールについて慎重に計画する必要があります。特に、次の点に注意してください。

- サーバー間接続の確立に必要なユーザー・アカウントは、分散データベース・システムのすべてのデータベースで使用可能である必要があります。
- 分散データベース・アプリケーションのユーザーに対してアプリケーション権限を使用可能にするために必要なロールは、分散データベース・システムのすべてのデータベースに存在する必要があります。

分散データベース・システム内のノードに対応するデータベース・リンクを作成する際は、それらのリンクを使用するサーバー間接続をサポートするために、各サイトでどのユーザー・アカウントとロールが必要になるかを決めてください。

通常、分散環境では、ユーザーは多数のネットワーク・サービスへのアクセスが必要になります。個々のユーザーが個々のネットワーク・サービスにアクセスするために個別の認証を構成する必要があるときは、特に大規模なシステムの場合にセキュリティの管理が難しくなることがあります。

> **関連項目：** 各種データベース・リンクをシステムでサポートするために使用可能にする必要があるユーザー・アカウントの詳細は、29-8 ページの「データベース・リンクの作成」を参照してください。

ユーザーと権限の集中管理

Oracle では、分散システムに関係するユーザーおよび権限を管理するための様々な手段が用意されています。たとえば、次のような方法があります。

- エンタープライズ・ユーザー管理。SSL を介して、またはパスワードを使用して認証されるグローバル・ユーザーを作成し、それらのユーザーと権限を独立したディレクトリ・サービスによってディレクトリ内で管理できます。
- ネットワーク認証サービス。この一般的な方法によって、分散環境のセキュリティ管理が容易になります。Oracle Advanced Security オプションを使用することで、Oracle Net と Oracle 分散データベース・システムのセキュリティを強化できます。Oracle 以

外の認証ソリューションの例としては、Windows NT のシステム固有の認証があります。

関連項目： グローバル・ユーザーのセキュリティの詳細は、『Oracle Advanced Security 管理者ガイド』を参照してください。

スキーマに依存するグローバル・ユーザー ユーザーと権限を集中管理する方法の 1 つとして、次のユーザーを作成できます。

- 中央ディレクトリ内のグローバル・ユーザー
- グローバル・ユーザーが接続する必要のあるすべてのデータベース内のユーザー

たとえば、次の SQL 文を使用して `fred` というグローバル・ユーザーを作成できます。

```
CREATE USER fred IDENTIFIED GLOBALLY AS 'CN=fred adams,O=Oracle,C=England';
```

この解決方法を使用すると、1 つのグローバル・ユーザーを中央ディレクトリによって認証できます。

スキーマに依存するグローバル・ユーザーによる解決方法では、`fred` というユーザーを、そのユーザーがアクセスする必要のあるすべてのデータベースに作成する必要があります。ユーザーのほとんどはアプリケーション・スキーマにアクセスするための権限は必要とするものの、自分自身のスキーマは必要としないため、すべてのグローバル・ユーザーに対し、各データベースにそれぞれのアカウントを作成するのは大変な手間となります。この問題を回避するため、Oracle ではスキーマに依存しないユーザーもサポートしています。このユーザーは、すべてのデータベース内にある単一の汎用スキーマにアクセスするグローバル・ユーザーです。

スキーマに依存しないグローバル・ユーザー Oracle では、グローバル・ユーザーをエンタープライズ・ディレクトリ・サービスによって集中管理する機能をサポートしています。このディレクトリ内で管理されるユーザーのことを、**エンタープライズ・ユーザー**と呼びます。このディレクトリには、次のことに関する情報が格納されています。

- エンタープライズ・ユーザーが分散システムのどのデータベースにアクセスできるのか。
- エンタープライズ・ユーザーが各データベースのどのロールを使用できるのか。
- エンタープライズ・ユーザーが各データベースのどのスキーマに接続できるのか。

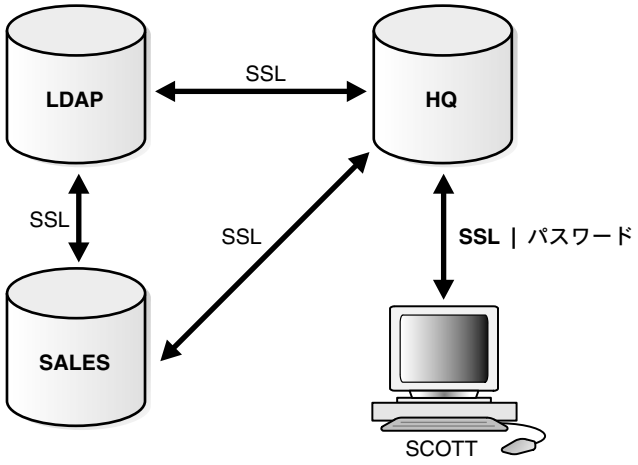
各データベースの管理者は、エンタープライズ・ユーザーが接続する必要のあるデータベースごとに各エンタープライズ・ユーザーのグローバル・ユーザー・アカウントを作成する必要はありません。そのかわりに、複数のエンタープライズ・ユーザーが**共有スキーマ**と呼ばれる同じデータベース・スキーマに接続できます。

注意： 共有スキーマ内の現行ユーザー・データベース・リンクにはアクセスできません。

たとえば、jane、bill および scott が全員、人事管理アプリケーションを使用しているとします。hq アプリケーション・オブジェクトはすべて、hq データベースの guest スキーマに格納されています。この場合、共有スキーマとして使用するローカルのグローバル・ユーザー・アカウントを作成できます。このグローバル・ユーザー名、つまり共有スキーマ名は guest です。jane、bill および scott はすべて、ディレクトリ・サービス内でエンタープライズ・ユーザーとして作成されます。また、3 名はディレクトリ内の guest スキーマにマップされ、hq アプリケーションの異なる認可が割り当てられることが可能です。

図 28-5 は、エンタープライズ・ディレクトリ・サービスを使用したグローバル・ユーザーのセキュリティの例を示しています。

図 28-5　グローバル・ユーザーのセキュリティ



エンタープライズ・ディレクトリ・サービスに、hq および sales のエンタープライズ・ユーザーに関する次の情報が格納されているとします。

データベース	ロール	スキーマ	エンタープライズ・ユーザー
hq	clerk1	guest	bill scott
sales	clerk2	guest	jane scott

また、hq および sales のローカル管理者が次の文を発行したとします。

データベース	CREATE 文
hq	CREATE USER guest IDENTIFIED GLOBALLY AS ''; CREATE ROLE clerk1 GRANT select ON emp; CREATE PUBLIC DATABASE LINK sales_link CONNECT AS CURRENT_USER USING 'sales';
sales	CREATE USER guest IDENTIFIED GLOBALLY AS ''; CREATE ROLE clerk2 GRANT select ON dept;

ここで、sales に関係する分散トランザクションを実行するために、エンタープライズ・ユーザー scott がローカル・データベース hq への接続を要求するとします。このとき、次の手順が実行されます（ただし、必ずしもこの順序どおりとはかぎりません）。

1. エンタープライズ・ユーザー scott が、SSL またはパスワードを使用して認証されます。
2. ユーザー scott が次の文を発行します。

   ```
   SELECT e.ename, d.loc
   FROM emp e, dept@sales_link d
   WHERE e.deptno=d.deptno;
   ```

3. データベース hq と sales が、SSL を使用して相互に認証します。
4. データベース hq はエンタープライズ・ディレクトリ・サービスを問い合せて、エンタープライズ・ユーザー scott が hq にアクセスできるかどうかを判断し、scott がロール clerk1 を使用してローカル・スキーマ guest にアクセスできることを確認します。
5. データベース sales はエンタープライズ・ディレクトリ・サービスを問い合せて、エンタープライズ・ユーザー scott が sales にアクセスできるかどうかを判断し、scott がロール clerk2 を使用してローカル・スキーマ guest にアクセスできることを確認します。
6. エンタープライズ・ユーザー scott は sales にログインし、ロール clerk2 を使用して guest スキーマにアクセスします。そこで SELECT を発行し、必要な情報を取得してその情報を hq に送信します。
7. データベース hq は要求されたデータを sales から受信し、それをクライアント scott に返します。

関連項目： エンタープライズ・ユーザーのセキュリティの詳細は、『Oracle Advanced Security 管理者ガイド』を参照してください。

データの暗号化

Oracle Advanced Security オプションでは、データが解読または改ざんされないように、Oracle Net とその関連製品でネットワーク・データの暗号化とチェックサムを使用できます。この機能では、RSA Data Security 社の RC4 またはデータ暗号化規格（DES）の暗号化アルゴリズムを使用することによって、データが不正に読み取られることを防ぎます。

データが転送中に改ざん、削除または再現されなかったことを保証するために、Oracle Advanced Security オプションのセキュリティ・サービスは、暗号的に安全なメッセージ・ダイジェストを生成し、ネットワーク上に送られる各パケットにそのダイジェストを含めることができます。

> **関連項目：** Oracle Advanced Security オプションのこれらの機能およびその他の機能の詳細は、『Oracle Advanced Security 管理者ガイド』を参照してください。

データベース・リンクの監査

操作の監査は、常にローカルで実行する必要があります。つまり、適切な監査オプションがそれぞれのデータベースで設定されている場合は、ユーザーがローカル・データベース内で操作を実行し、データベース・リンクを介してリモート・データベースにアクセスすると、ローカルの操作はローカル・データベースで監査され、リモートの操作はリモート・データベースで監査されます。

リモート・データベースでは、正常終了した接続要求とその後の SQL 文が別のサーバーからのものか、またはローカルに接続しているクライアントからのものかを判断することはできません。たとえば、次のような場合を考えてみます。

- 固定ユーザー・リンク `hq.acme.com` により、ローカル・ユーザー `jane` がリモート・ユーザー `scott` としてリモートの `hq` データベースに接続します。
- ユーザー `scott` は、リモート・データベースで監査されます。

リモート・データベース・セッション中に実行される操作は、あたかも `scott` が `hq` にローカルに接続し、そこで同じ操作を実行しているかのように監査されます。`jane` がリモート・データベースで何を実行しているのかを監査する場合は、リモート・データベースで監査オプションを設定し、リンクに埋め込まれているユーザー名（この場合は `hq` データベースの `scott`）の操作を捕捉する必要があります。

> **注意：** グローバル・ユーザーに対応するグローバル・ユーザー名を監査できます。

リモート・オブジェクトに対してローカルの監査オプションを設定することはできません。したがって、リモート・オブジェクトへのアクセスをリモート・データベースで監査することはできますが、データベース・リンクの使用は監査できません。

管理ツール

DBA は、Oracle 分散データベース・システムを管理する際にいくつかのツールを選択できます。

- Enterprise Manager
- サード・パーティ製管理ツール

Enterprise Manager

Enterprise Manager は、GUI を備えた Oracle のデータベース管理ツールです。操作性に優れたインタフェースを介して、分散データベースの管理機能を提供します。Enterprise Manager は、次の操作に使用できます。

- 複数のデータベースの管理。Enterprise Manager を使用して、1 つのデータベースを管理したり、複数のデータベースを同時に管理できます。
- データベース管理作業の集中化。世界中のあらゆる場所のあらゆる Oracle プラットフォームで稼働しているローカルおよびリモートの両方のデータベースを管理できます。また、Oracle Net がサポートしている任意のネットワーク・プロトコルでこれらの Oracle プラットフォームを接続できます。
- SQL、PL/SQL および Enterprise Manager コマンドの動的な実行。Enterprise Manager を使用して、文を入力、編集および実行できます。実行した文の履歴も保持されます。

  これにより、それらの文を再入力しなくても再実行できるので、分散データベース・システムで非常に長い文を繰り返し実行する必要がある場合に特に便利です。
- グローバル・ユーザー、グローバル・ロール、エンタープライズ・ディレクトリ・サービスなどのセキュリティ機能の管理。

サード・パーティ製管理ツール

現在、Oracle データベースおよびネットワークの管理に役立つ製品が、60 を超える企業から 150 以上出荷されており、真にオープンな環境を実現しています。

分散システムでのトランザクション処理

トランザクションとは、1 人のユーザーが実行する 1 つ以上の SQL 文によって構成された論理作業単位のことです。トランザクションは、ユーザーの最初に実行可能な SQL 文で開始し、そのユーザーによってコミットまたはロールバックされたときに終了します。

リモート・トランザクションは、1 つのリモート・ノードにアクセスする文のみで構成されます。**分散トランザクション**は、複数のノードにアクセスする文で構成されます。

次の項では、トランザクション処理における重要な概念を定義し、トランザクションが分散データベースのデータにアクセスする仕組みについて説明します。

- リモート SQL 文
- 分散 SQL 文
- リモート文と分散型の文の共有 SQL
- リモート・トランザクション
- 分散トランザクション
- 2 フェーズ・コミット・メカニズム
- データベース・リンクの名前解決
- スキーマ・オブジェクトの名前解決

リモート SQL 文

リモート問合せ文とは、そのすべてが同一のリモート・ノードに存在している 1 つ以上のリモート表から情報を選択する問合せのことです。たとえば、次の問合せは、リモートの sales データベースの scott スキーマにある dept 表のデータにアクセスします。

```
SELECT * FROM scott.dept@sales.us.americas.acme_auto.com;
```

リモート更新文とは、そのすべてが同一のリモート・ノードに存在している 1 つ以上の表のデータを変更する更新のことです。たとえば、次の問合せは、リモートの sales データベースの scott スキーマにある dept 表を更新します。

```
UPDATE scott.dept@mktng.us.americas.acme_auto.com
  SET loc = 'NEW YORK'
  WHERE deptno = 10;
```

注意： リモート更新には 1 つ以上のリモート・ノードからデータを取得する副問合せを含めることができますが、更新は 1 つのリモート・ノードでのみ発生するため、その文はリモート更新として分類されます。

分散 SQL 文

分散問合せ文は、複数のノードから情報を取得します。たとえば、次の問合せは、ローカル・データベースのデータと同時にリモートの sales データベースのデータにもアクセスします。

```
SELECT ename, dname
  FROM scott.emp e, scott.dept@sales.us.americas.acme_auto.com d
  WHERE e.deptno = d.deptno;
```

分散更新文は、複数のノードのデータを変更します。分散更新を行うには、プロシージャやトリガーなど、異なるノードのデータにアクセスする複数のリモート更新を含む PL/SQL サブプログラム・ユニットを使用します。たとえば、次の PL/SQL プログラム・ユニットは、ローカル・データベースとリモートの sales データベースにある表を更新します。

```
BEGIN
  UPDATE scott.dept@sales.us.americas.acme_auto.com
    SET loc = 'NEW YORK'
    WHERE deptno = 10;
  UPDATE scott.emp
    SET deptno = 11
    WHERE deptno = 10;
END;
COMMIT;
```

プログラム内の文がリモート・ノードに送られると、それらの文の実行はユニットとして成功または失敗します。

リモート文と分散型の文の共有 SQL

共有 SQL を使用したリモート文または分散型の文の仕組みは、本質的にはローカル文の仕組みと同じです。SQL テキストは必ず一致している必要があり、参照先のオブジェクトも必ず一致している必要があります。可能であれば、任意の文または分解された問合せをローカルまたはリモートで処理するために共有 SQL 領域を使用できます。

> **関連項目：** 共有 SQL の詳細は、『Oracle9*i* データベース概要』を参照してください。

リモート・トランザクション

リモート・トランザクションには 1 つ以上のリモート文があり、そのすべてが 1 つのリモート・ノードを参照しています。たとえば、次のトランザクションには 2 つの文があり、それぞれがリモートの sales データベースにアクセスします。

```
UPDATE scott.dept@sales.us.americas.acme_auto.com
  SET loc = 'NEW YORK'
  WHERE deptno = 10;
UPDATE scott.emp@sales.us.americas.acme_auto.com
  SET deptno = 11
  WHERE deptno = 10;
COMMIT;
```

分散トランザクション

分散トランザクションとは、1 つ以上の文からなり、それらが個別に、またはグループとして、分散データベースの複数のノードのデータを更新するようなトランザクションのことです。たとえば、このトランザクションは、ローカル・データベースとリモートの sales データベースを更新します。

```
UPDATE scott.dept@sales.us.americas.acme_auto.com
  SET loc = 'NEW YORK'
  WHERE deptno = 10;
UPDATE scott.emp
  SET deptno = 11
  WHERE deptno = 10;
COMMIT;
```

注意： トランザクションのすべての文が 1 つのリモート・ノードのみを参照している場合、そのトランザクションは分散トランザクションではなくリモート・トランザクションです。

2 フェーズ・コミット・メカニズム

データベースは、トランザクションが分散か非分散かにかかわらず、トランザクション内のすべての文がユニットとしてコミットまたはロールバックすることを保証します。実行中のトランザクションの結果は、すべてのノードの全トランザクションに対して不可視であり、このような透過性は、問合せや更新、リモート・プロシージャ・コールなど、あらゆるタイプの操作を含むトランザクションにおいて成り立つことが必要です。

非分散データベースにおけるトランザクション管理の一般的なメカニズムの詳細は、『Oracle9*i* データベース概要』を参照してください。分散データベースでは、Oracle はネットワーク上で同じ特性を持つトランザクション管理を連携させる必要があり、ネットワークやシステムに障害が発生しても、データ整合性を維持する必要があります。

Oracle の **2 フェーズ・コミット**・メカニズムは、分散トランザクションに参加しているすべてのデータベース・サーバーが、トランザクション内の文をすべてコミットするか、またはすべてロールバックすることを保証します。また、2 フェーズ・コミット・メカニズムにより、整合性制約、リモート・プロシージャ・コールおよびトリガーによって実行される暗黙的なデータ操作言語（DML）操作が保護されます。

> **関連項目：** Oracle の 2 フェーズ・コミット・メカニズムの詳細は、第 31 章「分散トランザクションの概念」を参照してください。

データベース・リンクの名前解決

グローバル・オブジェクト名とは、データベース・リンクを使用して指定されたオブジェクトのことです。グローバル・オブジェクト名の必須構成要素は、次のとおりです。

- オブジェクト名
- データベース名
- ドメイン

次の表は、明示的に指定されるグローバル・データベース・オブジェクト名の構成要素を示しています。

文	オブジェクト	データベース	ドメイン
`SELECT * FROM joan.dept@sales.acme.com`	`dept`	`sales`	`acme.com`
`SELECT * FROM emp@mktg.us.acme.com`	`emp`	`mktg`	`us.acme.com`

SQL 文にグローバル・オブジェクト名への参照が含まれていると、Oracle は必ずグローバル・オブジェクト名の中で指定されているデータベース名と一致する名前を持つデータベース・リンクを検索します。たとえば、次の文を発行した場合を考えます。

```
SELECT * FROM scott.emp@orders.us.acme.com;
```

この場合は、orders.us.acme.com というデータベース・リンクが検索されます。Oracle はこの操作を実行して、指定されたリモート・データベースへのパスを判断します。

一致するデータベース・リンクは、常に次の順序で検索されます。

1. SQL 文を発行したユーザーのスキーマ内のプライベート・データベース・リンク
2. ローカル・データベース内のパブリック・データベース・リンク
3. グローバル・データベース・リンク（Oracle Names Server が使用可能な場合のみ）

グローバル・データベース名が完全なときの名前解決

完全なグローバル・データベース名が指定された次の SQL 文を発行するとします。

```
SELECT * FROM emp@prod1.us.oracle.com;
```

この場合は、データベース名（prod1）とドメイン構成要素（us.oracle.com）の両方を指定しているため、プライベート、パブリックおよびグローバルのデータベース・リンクが検索されます。Oracle は、指定されたグローバル・データベース名に一致するリンクのみを検索します。

グローバル・データベース名が部分的なときの名前解決

ドメインの任意の一部を指定した場合は、完全なグローバル・データベース名を指定したものとみなされます。SQL 文で部分的なグローバル・データベース名を指定した場合（つまり、データベース構成要素のみを指定した場合）、Oracle は DB_DOMAIN 初期化パラメータ値を DB_NAME 初期化パラメータ値の後に追加し、完全な名前を構成します。たとえば、次の文を発行した場合を考えます。

```
CONNECT scott/tiger@locdb
SELECT * FROM scott.emp@orders;
```

locdb のネットワーク・ドメインが us.acme.com の場合、Oracle はこのドメインを orders の後に追加し、orders.us.acme.com という完全なグローバル・データベース名を構成します。続いて、構成されたグローバル名に一致するデータベース・リンクが検索されます。一致するリンクが見つからなければ、エラーが返され、SQL 文は実行できません。

グローバル・データベース名をまったく指定しないときの名前解決

グローバル・オブジェクト名がローカル・データベース内のオブジェクトを参照していて、データベース・リンク名がアットマーク（@）を使用して指定されていない場合、Oracle はオブジェクトがローカルであることを自動的に検出して検索を行わないか、またはデータベース・リンクを使用してオブジェクト参照を解決します。たとえば、次の文を発行した場合を考えます。

```
CONNECT scott/tiger@locdb
SELECT * from scott.emp;
```

2 番目の文ではデータベース・リンク接続文字列を使用してグローバル・データベース名を指定していないため、データベース・リンクは検索されません。

名前解決のための検索の終了

Oracle は、最初に一致したデータベース・リンクが見つかったときに、必ずしも一致するデータベース・リンクの検索を停止するとはかぎりません。リモート・データベースへの完全なパス（リモート・アカウントとサービス名の両方）がわかるまで、一致するプライベート、パブリックおよびネットワークのデータベース・リンクが検索されます。

最初に一致したデータベース・リンクが見つかると、次の表に従ってリモート・スキーマが決定されます。

操作	Oracle の処理	例
CONNECT 句が指定されていない場合	接続ユーザー・データベース・リンクを使用します。	CREATE DATABASE LINK k1 USING 'prod'
CONNECT TO ... IDENTIFIED BY 句が指定されている場合	固定ユーザー・データベース・リンクを使用します。	CREATE DATABASE LINK k2 CONNECT TO scott IDENTIFIED BY tiger USING 'prod'
CONNECT TO CURRENT_USER 句が指定されている場合	現行ユーザー・データベース・リンクを使用します。	CREATE DATABASE LINK k3 CONNECT TO CURRENT_USER USING 'prod'
USING 句が指定されていない場合	データベース文字列を指定するリンクが見つかるまで検索します。一致するデータベース・リンクが見つかっても文字列がまったく識別されない場合は、エラーを返します。	CREATE DATABASE LINK k4 CONNECT TO CURRENT_USER

完全なパスが決定した後、リモート・セッションが作成されます（同じローカル・セッションのために同一の接続がまだオープンになっていない場合）。セッションがすでに存在していれば、それが再利用されます。

スキーマ・オブジェクトの名前解決

ローカルの Oracle データベースが、SQL 文を発行したローカル・ユーザーのために指定のリモート・データベースに接続した後も、あたかもリモート・ユーザーが対応する SQL 文を発行したかのように、オブジェクトの解決処理が続きます。最初に一致したデータベース・リンクが見つかると、次の規則に従ってリモート・スキーマが決定されます。

使用するリンク	オブジェクト解決が処理されるスキーマ
固定ユーザー・データベース・リンク	リンク作成文で指定されたスキーマ
接続ユーザー・データベース・リンク	接続ユーザーのリモート・スキーマ
現行ユーザー・データベース・リンク	現行ユーザーのスキーマ

オブジェクトが見つからなかった場合、Oracle はリモート・データベースのパブリック・オブジェクトをチェックします。オブジェクトを解決できなくても確立されたリモート・セッションは残りますが、SQL 文は実行できず、エラーが返されます。

次に、分散データベース・システムにおけるグローバル・オブジェクトの名前解決の例を示します。

グローバル・オブジェクトの名前解決の例 : 完全なオブジェクト名

この例では、Oracle が完全なグローバル・オブジェクト名をどのように解決し、プライベートとパブリックの両方のデータベース・リンクを使用してリモート・データベースへの適切なパスをどのように決定するのかを示します。この例では、次のことを想定しています。

- リモート・データベースの名前は、`sales.division3.acme.com` です。
- ローカル・データベースの名前は、`hq.division3.acme.com` です。
- Oracle Names Server は使用できません。したがって、グローバル・データベース・リンクも使用できません。
- リモート表 `emp` は、スキーマ `tsmith` 内にあります。

次の文が `scott` によってローカル・データベースで発行された場合を考えます。

```
CONNECT scott/tiger@hq

CREATE PUBLIC DATABASE LINK sales.division3.acme.com
CONNECT TO guest IDENTIFIED BY network
  USING 'dbstring';
```

その後、Jward が接続して次の文を発行するとします。

```
CONNECT jward/bronco@hq

CREATE DATABASE LINK sales.division3.acme.com
  CONNECT TO tsmith IDENTIFIED BY radio;

UPDATE tsmith.emp@sales.division3.acme.com
  SET deptno = 40
  WHERE deptno = 10;
```

Oracle は、最後の文を次のように処理します。

1. Oracle は、jward の更新文の中で完全なグローバル・オブジェクト名が参照されていると判断します。したがって、一致する名前を持つデータベース・リンクの検索がローカル・データベース内で開始されます。
2. 一致するプライベート・データベース・リンクがスキーマ jward 内で見つかります。しかし、プライベート・データベース・リンク jward.sales.division3.acme.com は、リモートの sales データベースへの完全なパスを示しておらず、リモート・アカウントのみを示しています。そのため、続いて一致するパブリック・データベース・リンクが検索されます。
3. scott のスキーマ内でパブリック・データベース・リンクが見つかります。このパブリック・データベース・リンクからネット・サービス名 dbstring が取得されます。
4. Oracle は、一致するプライベートの固定ユーザー・データベース・リンクから取得したリモート・アカウントを結合して完全なパスを決定し、次に、ユーザー tsmith/radio としてリモートの sales データベースへ接続します。
5. これで、リモート・データベースは、emp 表へのオブジェクト参照を解決できます。Oracle は tsmith スキーマ内で検索を行い、参照先の emp 表を見つけます。
6. リモート・データベースは文の実行を完了し、その結果をローカル・データベースに返します。

グローバル・オブジェクトの名前解決の例 : 部分的なオブジェクト名

この例では、Oracle が部分的なグローバル・オブジェクト名をどのように解決し、プライベートとパブリックの両方のデータベース・リンクを使用してリモート・データベースへの適切なパスをどのように決定するのかを示します。

この例では、次のことを想定しています。

- リモート・データベースの名前は、sales.division3.acme.com です。
- ローカル・データベースの名前は、hq.division3.acme.com です。
- Oracle Names Server は使用できません。したがって、グローバル・データベース・リンクも使用できません。

- リモート・データベース sales の表 emp はスキーマ tsmith 内にあり、スキーマ scott 内にはありません。
- リモート・データベース sales に emp というパブリック・シノニムが存在します。このシノニムは、リモート・データベース sales の tsmith.emp を指しています。
- 28-39 ページの「グローバル・オブジェクトの名前解決の例 : 完全なオブジェクト名」で示したパブリック・データベース・リンクが、ローカル・データベース hq にすでに作成されています。

```
CREATE PUBLIC DATABASE LINK sales.division3.acme.com
  CONNECT TO guest IDENTIFIED BY network
  USING 'dbstring';
```

ローカル・データベース hq で次の文が発行された場合を考えます。

```
CONNECT scott/tiger@hq

CREATE DATABASE LINK sales.division3.acme.com;

DELETE FROM emp@sales
  WHERE empno = 4299;
```

Oracle は、最後の DELETE 文を次のように処理します。

1. Oracle は、scott の DELETE 文の中で部分的なグローバル・オブジェクト名が参照されていることを検出します。Oracle は、次のようにローカル・データベースのドメインを使用して、部分的なグローバル・オブジェクト名を完全なグローバル・オブジェクト名に拡張します。

   ```
   DELETE FROM emp@sales.division3.acme.com
     WHERE empno = 4299;
   ```

2. 一致する名前を持つデータベース・リンクがローカル・データベース内で検索されます。
3. スキーマ scott 内で、一致するプライベートの接続ユーザー・リンクが見つかりますが、そのプライベート・データベース・リンクにはパスがまったく示されていません。Oracle は、接続先のユーザー名 / パスワードをパスのリモート・アカウントとして使用し、一致するパブリックのデータベース・リンクを検索して見つけます。

   ```
   CREATE PUBLIC DATABASE LINK sales.division3.acme.com
     CONNECT TO guest IDENTIFIED BY network
     USING 'dbstring';
   ```

4. このパブリック・データベース・リンクからサービス名 dbstring が取得されます。この時点で、Oracle は完全なパスを決定しました。
5. Oracle はリモート・データベースに scott/tiger として接続し、検索を行いますが、スキーマ scott 内で emp というオブジェクトは見つかりません。
6. リモート・データベースは、emp というパブリック・シノニムを検索して見つけます。
7. リモート・データベースは文を実行し、その結果をローカル・データベースに返します。

ビュー、シノニムおよびプロシージャでのグローバル名前解決

ビュー、シノニムまたは PL/SQL プログラム・ユニット（プロシージャ、ファンクション、トリガーなど）は、グローバル・オブジェクト名によってリモートのスキーマ・オブジェクトを参照できます。グローバル・オブジェクト名が完全であれば、グローバル・オブジェクト名は拡張されずにオブジェクトの定義が格納されます。しかし、名前が部分的であれば、ローカル・データベース名のドメインを使用してその名前が拡張されます。

次の表は、ビュー、シノニムおよびプログラム・ユニットの部分的なグローバル・オブジェクト名を Oracle がいつ拡張するのかを示したものです。

操作	Oracle の処理
ビューの作成	部分的なグローバル名は拡張されません。定義内の問合せのテキストがそのままデータ・ディクショナリに格納されます。そのかわりに、ビューを使用する文が解析されるたびに、部分的なグローバル・オブジェクト名が拡張されます。
シノニムの作成	部分的なグローバル名が拡張されます。データ・ディクショナリに格納されるシノニムの定義には、拡張されたグローバル・オブジェクト名が含まれます。
プログラム・ユニットのコンパイル	部分的なグローバル名が拡張されます。

グローバル名を変更したときに起こる動作

グローバル名の変更は、部分的なグローバル・オブジェクト名を使用してリモート・データを参照するビュー、シノニムおよびプロシージャに影響を与える可能性があります。参照先データベースのグローバル名を変更すると、ビューおよびプロシージャは存在しないデータベースまたは正しくないデータベースを参照しようとすることがあります。一方、シノニムは実行時にデータベース・リンク名を拡張しないので、何も変わりません。

グローバル名の変更例

たとえば、sales.uk.acme.com および hq.uk.acme.com という 2 つのデータベースを考えます。また、sales データベースに次のビューとシノニムがあるとします。

```
CREATE VIEW employee_names AS
        SELECT ename FROM scott.emp@hr;

CREATE SYNONYM employee FOR scott.emp@hr;
```

Oracle は、employee シノニム定義を拡張して、次のように保存します。

```
scott.emp@hr.uk.acme.com
```

使用例 1: 両方のデータベース名が変更された場合 最初に、営業および人事の両部門が米国に配置替えされるという状況を考えます。その結果、対応するグローバル・データベース名はどちらも次のように変更されます。

古いグローバル名	新しいグローバル名
sales.uk.acme.com	sales.us.acme.com
hq.uk.acme.com	hq.us.acme.com

次の表は、グローバル名の変更前および変更後の問合せの拡張を示しています。

sales への問合せ	変更前の拡張	変更後の拡張
SELECT * FROM employee_names	SELECT * FROM scott.emp@hr.uk.acme.com	SELECT * FROM scott.emp@hr.us.acme.com
SELECT * FROM employee	SELECT * FROM scott.emp@hr.uk.acme.com	SELECT * FROM scott.emp@hr.uk.acme.com

使用例 2: 一方のデータベース名が変更された場合 この例では、営業部のみが米国に移動し、人事部は英国に留まるとします。その結果、対応するグローバル・データベース名はどちらも次のように変更されます。

古いグローバル名	新しいグローバル名
sales.uk.acme.com	sales.us.acme.com
hq.uk.acme.com	変更なし

次の表は、グローバル名の変更前および変更後の問合せの拡張を示しています。

sales への問合せ	変更前の拡張	変更後の拡張
SELECT * FROM employee_names	SELECT * FROM scott.emp@hr.uk.acme.com	SELECT * FROM scott.emp@hr.us.acme.com
SELECT * FROM employee	SELECT * FROM scott.emp@hr.uk.acme.com	SELECT * FROM scott.emp@hr.uk.acme.com

この場合、employee_names ビューを定義している問合せが、存在しないグローバル・データベース名に拡張されます。一方、employee シノニムは、引き続き正しいデータベースである hq.uk.acme.com を参照します。

分散データベース・アプリケーションの開発

分散システムにおけるアプリケーションの開発では、非分散システムには当てはまらない問題が起こります。ここでは、分散アプリケーションの開発について説明します。この項の内容は、次のとおりです。

- 分散データベース・システムにおける透過性
- リモート・プロシージャ・コール（RPC）
- 分散問合せの最適化

関連項目： 分散システム向けアプリケーションの開発方法の詳細は、第 30 章「分散データベース・システムのアプリケーション開発」を参照してください。

分散データベース・システムにおける透過性

データベース・システムを使用するユーザーに対して Oracle 分散データベース・システムを透過的にするアプリケーションを最小限の作業で開発できます。透過性の目的は、分散データベース・システムを単一の Oracle データベースであるかのように扱えるようにすることです。その結果、開発者およびシステムのユーザーは、分散データベースの複雑さから解放されます。透過性が備わっていなければ、この複雑さが原因で分散データベース・アプリケーションの開発は困難になり、ユーザーの生産性は大幅に低下することになります。

ここでは、分散データベース・システムにおける透過性についてさらに詳しく説明します。

位置の透過性

Oracle 分散データベース・システムには、アプリケーション開発者および管理者がデータベース・オブジェクトの物理的な位置をアプリケーションおよびユーザーから隠す機能があります。ユーザーが、アプリケーションの接続先ノードに関係なく、表などのデータベース・オブジェクトを一様に参照できるのには、**位置の透過性**が関係しています。位置の透過性には、次のようないくつかの利点があります。

- データベースのユーザーはデータベース・オブジェクトの物理的な位置を知る必要がないため、リモート・データへのアクセスが簡単になります。
- 管理者は、エンド・ユーザーや既存のデータベース・アプリケーションに影響を与えることなく、データベース・オブジェクトを移動できます。

通常、管理者および開発者は、アプリケーション・スキーマ内の表およびサポート・オブジェクトに対する位置の透過性を確立するために、シノニムを使用します。たとえば、次の文は、データベース内に別のリモート・データベース内の表に対応するシノニムを作成します。

```
CREATE PUBLIC SYNONYM emp
  FOR scott.emp@sales.us.americas.acme_auto.com;
CREATE PUBLIC SYNONYM dept
  FOR scott.dept@sales.us.americas.acme_auto.com;
```

これにより、リモート表にアクセスする際に次のような問合せを使用する必要がなくなります。

```
SELECT ename, dname
  FROM scott.emp@sales.us.americas.acme_auto.com e,
       scott.dept@sales.us.americas.acme_auto.com d
  WHERE e.deptno = d.deptno;
```

アプリケーションでは、リモート表の位置を考慮せずに、単純な問合せを発行できます。

```
SELECT ename, dname
  FROM emp e, dept d
  WHERE e.deptno = d.deptno;
```

シノニムの他にも、ビューおよびストアド・プロシージャを使用して、分散データベース・システムで動作するアプリケーションに対する位置の透過性を確立できます。

SQL および COMMIT の透過性

Oracle の分散データベース・アーキテクチャでは、問合せ、更新およびトランザクションの透過性も提供されています。たとえば、SELECT、INSERT、UPDATE、DELETE などの標準の SQL 文は、非分散データベース環境の場合とまったく同じように動作します。また、アプリケーションは、標準の SQL 文である COMMIT、SAVEPOINT および ROLLBACK を使用して、トランザクションを管理します。分散トランザクションを制御するために複雑なプログラミングやその他の特別な操作を行う必要はありません。

- 1 つのトランザクション内の文からは任意の数のローカル表またはリモート表を参照できます。
- Oracle では、分散トランザクションに関係するノードがすべて同じ動作をすることが保証されており、それらのノードすべてがトランザクションをコミットまたはロールバックします。
- 分散トランザクションのコミット中にネットワークまたはシステムの障害が発生した場合、トランザクションは自動的かつ透過的、およびグローバルに解決されます。具体的には、ネットワークまたはシステムがリストアされると、すべてのノードがトランザクションをコミットまたはロールバックします。

Oracle の内部では、コミットされた各トランザクションに対して、そのトランザクション内の文による変更を一意に識別するための**システム変更番号（SCN）**が対応付けられます。分散データベースでは、次のときに通信中のノードの SCN が調整されます。

- 1 つ以上のデータベース・リンクによって表されるパスを使用して接続が確立されるとき
- 分散 SQL 文が実行されるとき
- 分散トランザクションがコミットされるとき

特に、分散データベース・システムのノード間で SCN が調整されることにより、文とトランザクションの両方のレベルでグローバルな分散読込み一貫性が保証されることは大きな利点です。必要であれば、グローバルな時間ベースの分散リカバリを完了することもできます。

レプリケーションの透過性

Oracle はまた、システムのノード間でデータを透過的にレプリケートするための機能も数多く備えています。Oracle のレプリケーション機能の詳細は、『Oracle9*i* レプリケーション』を参照してください。

リモート・プロシージャ・コール（RPC）

開発者は、分散データベースで動作するアプリケーションをサポートするための PL/SQL パッケージおよびプロシージャをコーディングできます。アプリケーションでは、ローカル・データベースでの処理を実行するためにローカル・プロシージャ・コールを実行でき、リモート・データベースでの処理を実行するために **RPC** を実行できます。

プログラムがリモート・プロシージャをコールすると、ローカル・サーバーは、コールされたリモート・サーバーにすべてのプロシージャ・パラメータを渡します。たとえば、次の PL/SQL プログラム・ユニットは、リモートの `sales` データベースにあるパッケージ化されたプロシージャ `del_emp` をコールし、それにパラメータ 1257 を渡します。

```
BEGIN
 emp_mgmt.del_emp@sales.us.americas.acme_auto.com(1257);
END;
```

RPC を正常に実行するためには、コール側プロシージャがリモート・サイトに存在し、接続しようとするユーザーがそのプロシージャを実行するための適切な権限を持っている必要があります。

分散データベース・システム対応のパッケージおよびプロシージャを開発する際、開発者は、どのプログラム・ユニットをリモートの位置で実行し、その結果をどのようにコール側アプリケーションに返すのかについて理解した上で、コードを記述する必要があります。

分散問合せの最適化

Oracle の機能の 1 つである**分散問合せの最適化**は、トランザクションの分散 SQL 文内で参照されているリモート表からデータを取得するときに、サイト間で必要となるデータ転送の量を減らします。

分散問合せの最適化では、Oracle のコストベースの最適化を使用して、リモート表から必要なデータのみを抽出する SQL 式が検索または生成されます。抽出されたデータはリモート・サイト（場合によってはローカル・サイト）で処理され、その結果がローカル・サイトに送信されて、最終処理が行われます。このような操作により、表データのすべてをローカル・サイトに転送して処理する場合と比較して、必要なデータ転送の量が低減します。

`DRIVING_SITE`、`NO_MERGE`、`INDEX` など、各種のコストベース・オプティマイザ・ヒントを使用することにより、Oracle がデータを処理する場所とデータへのアクセス方法を制御できます。

> **関連項目：** コストベースの最適化の詳細は、30-5 ページの「コストベース最適化の使用」を参照してください。

分散環境でのキャラクタ・セットのサポート

Oracle は、クライアント、Oracle データベース・サーバーおよび Oracle 以外のサーバーが異なるキャラクタ・セットを使用する環境をサポートしています。Oracle では、異機種環境のために NCHAR のサポートが提供されています。各国語サポート（NLS）および異機種間サービス（HS）に関連する各種の環境変数および初期化パラメータを設定することにより、異なるキャラクタ・セット間でのデータ変換を制御できます。

キャラクタの設定は、次の NLS および HS パラメータで定義されます。

パラメータ	環境	定義の対象
NLS_LANG（環境変数）	クライアント / サーバー	クライアント
NLS_LANGUAGE NLS_CHARACTERSET NLS_TERRITORY	クライアント / サーバー 非異機種間分散 異機種間分散	Oracle データベース・サーバー
HS_LANGUAGE	異機種間分散	Oracle 以外のサーバー Transparent Gateway
NLS_NCHAR（環境変数） HS_NLS_NCHAR	異機種間分散	Oracle データベース・サーバー Transparent Gateway

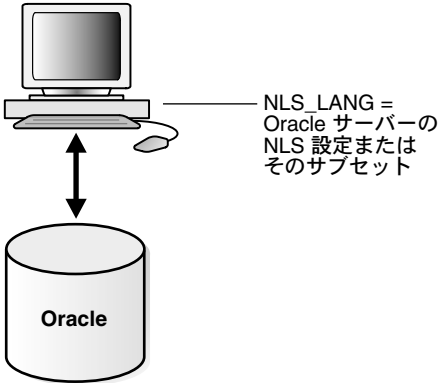
関連項目：

- NLS パラメータの詳細は、『Oracle9*i* Database グローバリゼーション・サポート・ガイド』を参照してください。
- HS パラメータの詳細は、『Oracle9*i* Heterogeneous Connectivity Administrator's Guide』を参照してください。

クライアント / サーバー環境

クライアント / サーバー環境では、図 28-6 のように、クライアント・キャラクタ・セットを Oracle データベース・サーバーのキャラクタ・セットと同じか、またはそのサブセットに設定します。

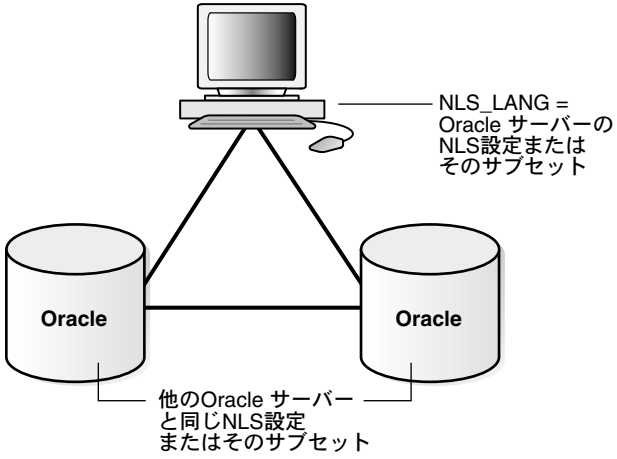
図 28-6　クライアント / サーバー環境における NLS パラメータの設定



同機種間分散環境

同機種環境では、図 28-7 のように、クライアントとサーバーのキャラクタ・セットを同じにするか、あるいはメイン・サーバーのキャラクタ・セットのサブセットにします。

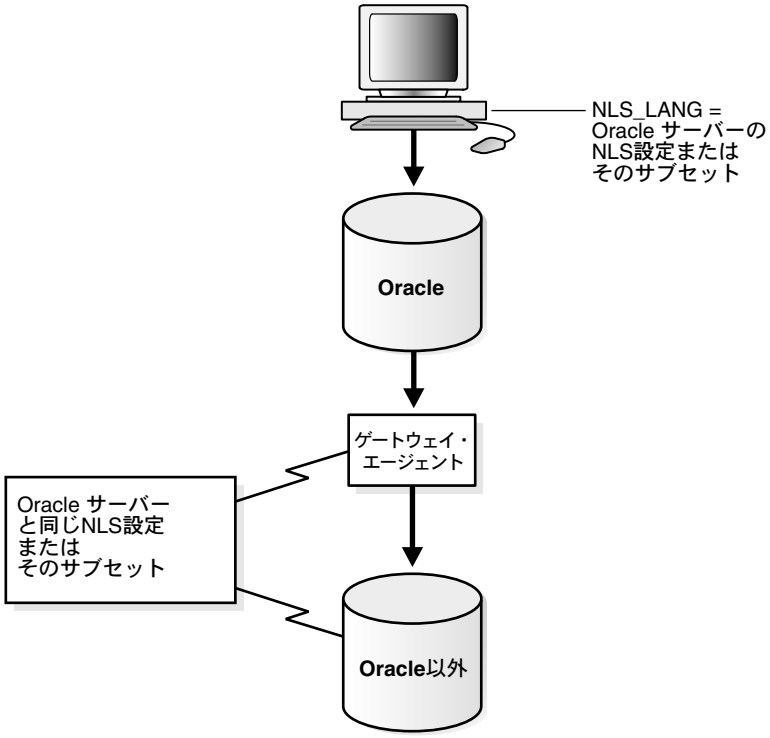
図 28-7　同機種環境における NLS パラメータの設定



異機種間分散環境

異機種環境では、図 28-8 のように、クライアント、Transparent Gateway および Oracle 以外のデータ・ソースの NLS 設定をすべて同じにするか、または Oracle データベース・サーバーのキャラクタ・セットのサブセットにします。Transparent Gateway は、グローバリゼーションを完全にサポートしています。

図 28-8　異機種環境における NLS パラメータの設定



異機種環境では、異機種間サービス・テクノロジによって構築された Transparent Gateway のみが完全な NCHAR 機能をサポートします。特定の Transparent Gateway が NCHAR をサポートしているかどうかは、対象となる Oracle 以外のデータ・ソースによって異なります。特定の Transparent Gateway による NCHAR サポートの処理方法の詳細は、システム固有の Transparent Gateway のマニュアルを参照してください。

> **関連項目：**　異機種間サービスの詳細は、『Oracle9*i* Heterogeneous Connectivity Administrator's Guide』を参照してください。

29

分散データベースの管理

この章では、分散データベース・システムの管理とメンテナンスの方法について説明します。この章の内容は、次のとおりです。

分散システムでのグローバル名の管理

分散データベース・システムでは、各データベースが一意の**グローバル・データベース名**を持ちます。グローバル・データベース名は、システム内のデータベースを一意に識別します。分散データベースでの主な管理作業として、グローバル・データベース名の作成と変更の管理があります。

この項の内容は、次のとおりです。

- グローバル・データベース名の書式の理解
- グローバル・ネーミング施行の判断
- グローバル・データベース名の参照
- グローバル・データベース名のドメインの変更
- グローバル・データベース名の変更 : 使用例

グローバル・データベース名の書式の理解

グローバル・データベース名は、データベース名とドメインの 2 つの構成要素から構成されます。データベース名とドメイン名は、データベースの作成時に次の初期化パラメータによって決まります。

構成要素	パラメータ	要件	例
データベース名	DB_NAME	文字数は必ず 8 文字以内です。	sales
データベースが存在するドメイン	DB_DOMAIN	必ず標準のインターネット表記規則に従います。ドメイン名の各レベルは必ずドットで区切ります。ドメイン名の順序はリーフ（葉）からルート（根）、左から右です。	us.acme.com

次に、有効なグローバル・データベース名の例を示します。

DB_NAME	DB_DOMAIN	グローバル・データベース名
sales	au.oracle.com	sales.au.oracle.com
sales	us.oracle.com	sales.us.oracle.com
mktg	us.oracle.com	mktg.us.oracle.com
payroll	nonprofit.org	payroll.nonprofit.org

DB_DOMAIN 初期化パラメータはデータベースの作成時にのみ重要であり、DB_NAME パラメータとともに使用してデータベースのグローバル名を構成します。データベースのグローバル名は、この時点でデータ・ディクショナリに格納されます。グローバル名を変更する際は、初期化パラメータ・ファイルの DB_DOMAIN パラメータを変更するのではなく、ALTER DATABASE 文を使用する必要があります。しかし、ドメインの変更が反映されるように、次回データベースを起動する前に DB_DOMAIN パラメータを変更しておくことには意味があります。

グローバル・ネーミング施行の判断

ローカル・データベースのリンクに付ける名前は、アクセスするリモート・データベースがグローバル・ネーミングを施行するかどうかによって異なります。リモート・データベースがグローバル・ネーミングを施行する場合は、リモート・データベースのグローバル・データベース名をリンクの名前として使用する必要があります。たとえば、ローカルの hq サーバーに接続していて、リモートの mfg データベースへのリンクを作成する場合、mfg がグローバル・ネーミングを施行していれば、mfg のグローバル・データベース名をリンク名として使用する必要があります。

データベース・リンク名の一部としてサービス名を使用することもできます。たとえば、サービス名 sn1 と sn2 を使用してデータベース hq.acme.com に接続している場合、hq がグローバル・ネーミングを施行していれば、hq へのリンク名を次のように作成できます。

- HQ.ACME.COM@SN1
- HQ.ACME.COM@SN2

関連項目： リンク名でのサービス名の使用の詳細は、29-13 ページの「リンク名に含まれるサービス名を指定するための接続修飾子の使用」を参照してください。

グローバル・ネーミングがデータベースで施行されているかどうかを判断するには、データベースの初期化パラメータ・ファイルを調べるか、または V$PARAMETER ビューを問い合せます。たとえば、mfg でグローバル・ネーミングが施行されているかどうかを判断するには、mfg のセッションを開始して、次の globalnames.sql スクリプトを作成し、実行できます（出力例も含まれています）。

```
COL NAME FORMAT A12
COL VALUE FORMAT A6
SELECT NAME, VALUE FROM V$PARAMETER
   WHERE NAME = 'global_names'
/

SQL> @globalnames
```

```
NAME         VALUE
------------ ------
global_names FALSE
```

グローバル・データベース名の参照

データベースのグローバル名を参照するには、データ・ディクショナリ・ビュー GLOBAL_NAME を使用します。たとえば、次の文を発行します。

```
SELECT * FROM GLOBAL_NAME;

GLOBAL_NAME
--------------------------------------------------------------------------------
SALES.AU.ORACLE.COM
```

グローバル・データベース名のドメインの変更

データベースのグローバル名のドメインを変更するには、ALTER DATABASE 文を使用します。データベースを作成した後に初期化パラメータ DB_DOMAIN を変更しても、グローバル・データベース名やデータベース・リンク名の解決には効果がありません。

次の例は、名前変更文の構文を示しています。ここで、*database* はデータベース名、*domain* はネットワーク・ドメインを表します。

```
ALTER DATABASE RENAME GLOBAL_NAME TO database.domain;
```

グローバル・データベース名のドメインを変更するには、次の手順を実行します。

1. 現行のグローバル・データベース名を確認します。たとえば、次の文を発行します。

   ```
   SELECT * FROM GLOBAL_NAME;

   GLOBAL_NAME
   --------------------------------------------------------------------------------
   SALES.AU.ORACLE.COM
   ```

2. ALTER DATABASE 文を使用して、グローバル・データベース名を変更します。たとえば、次のように入力します。

   ```
   ALTER DATABASE RENAME GLOBAL_NAME TO sales.us.oracle.com;
   ```

3. GLOBAL_NAME 表を問い合せて、新しい名前を確認します。たとえば、次のように入力します。

   ```
   SELECT * FROM GLOBAL_NAME;

   GLOBAL_NAME
   --------------------------------------------------------------------------------
   SALES.US.ORACLE.COM
   ```

グローバル・データベース名の変更 : 使用例

この例では、ローカル・データベースのグローバル・データベース名のドメイン部分を変更します。また、部分指定のグローバル名を使用してデータベース・リンクを作成し、Oracle がその名前をどのように解決するのかを調べます。Oracle は、初期化パラメータ DB_DOMAIN の値ではなく、ローカル・データベースの現行のグローバル・データベース名のドメイン部分を使用して部分名を解決することがわかります。

1. sales.us.acme.com に接続して GLOBAL_NAME データ・ディクショナリ・ビューを問い合せ、データベースの現行グローバル名を確認します。

```
CONNECT SYSTEM/password@sales.us.acme.com
SELECT * FROM GLOBAL_NAME;

GLOBAL_NAME
--------------------------------------------------------------------------
SALES.US.ACME.COM
```

2. V$PARAMETER ビューを問い合せ、DB_DOMAIN 初期化パラメータの現在の設定を調べます。

```
SELECT NAME, VALUE FROM V$PARAMETER WHERE NAME = 'db_domain';

NAME       VALUE
---------  -----------
db_domain  US.ACME.COM
```

3. 部分指定のグローバル名のみを使用して、hq データベースへのデータベース・リンクを作成します。

```
CREATE DATABASE LINK hq USING 'sales';
```

   Oracle は、ローカル・データベースのグローバル・データベース名のドメイン部分を、リンクで指定されたデータベースの名前の後に追加して、このリンクのグローバル・データベース名を拡張します。

4. USER_DB_LINKS を問い合せて、Oracle が部分指定のグローバル・データベース名を解決するためにどのドメイン名を使用したかを確認します。

```
SELECT DB_LINK FROM USER_DB_LINKS;

DB_LINK
------------------
HQ.US.ACME.COM
```

   この結果は、ローカル・データベースのグローバル・データベース名のドメイン部分が us.acme.com であることを示しています。Oracle は、データベース・リンク作成時の部分的なデータベース・リンク名を解決するために、このドメインを使用します。

5. sales データベースが日本に移動するという通知を受けたので、sales データベースを sales.jp.acme.com に変更します。

```
ALTER DATABASE RENAME GLOBAL_NAME TO sales.jp.acme.com;
SELECT * FROM GLOBAL_NAME;

GLOBAL_NAME
--------------------------------------------------------------------------
SALES.JP.ACME.COM
```

6. 再び V$PARAMETER を問い合せると、グローバル・データベース名のドメイン部分を変更しても、DB_DOMAIN の値は変更されていないことがわかります。

```
SELECT NAME, VALUE FROM V$PARAMETER
  WHERE NAME = 'db_domain';

NAME       VALUE
---------  -----------
db_domain  US.ACME.COM
```

この結果は、DB_DOMAIN 初期化パラメータの値が ALTER DATABASE RENAME GLOBAL_NAME 文とは関係がないことを示しています。ALTER DATABASE 文は、DB_DOMAIN 初期化パラメータではなく、グローバル・データベース名のドメインを決定します（それでも、新しいドメイン名が反映されるように DB_DOMAIN を変更することには意味があります）。

7. データベース supply への別のデータベース・リンクを作成してから、USER_DB_LINKS を問い合せて、Oracle が supply のグローバル・データベース名のドメイン部分をどのように解決するのかを確認します。

```
CREATE DATABASE LINK supply USING 'supply';
SELECT DB_LINK FROM USER_DB_LINKS;

DB_LINK
------------------
HQ.US.ACME.COM
SUPPLY.JP.ACME.COM
```

この結果は、Oracle がドメイン jp.acme.com を使用して、部分指定のリンク名を解決したことを示しています。このドメインは、リンクの作成時に使用されます。これは、このドメインがローカル・データベースのグローバル・データベース名のドメイン部分であるためです。Oracle は、部分的なリンク名を解決するときに、DB_DOMAIN 初期化パラメータの設定を使用しません。

8. 次に、前の情報が誤りで、sales は jp.acme.com ドメインではなく asia.jp.acme.com ドメインにあるという通知を受けました。そのため、グローバル・データベース名を次のように変更します。

```
ALTER DATABASE RENAME GLOBAL_NAME TO sales.asia.jp.acme.com;
SELECT * FROM GLOBAL_NAME;

GLOBAL_NAME
--------------------------------------------------------------------------
SALES.ASIA.JP.ACME.COM
```

9. 再び V$PARAMETER を問い合せて、パラメータ DB_DOMAIN の設定を確認します。

```
SELECT NAME, VALUE FROM V$PARAMETER
  WHERE NAME = 'db_domain';

NAME        VALUE
----------  -----------
db_domain   US.ACME.COM
```

この結果は、パラメータ・ファイルのドメイン設定が、2 つの ALTER DATABASE RENAME 文を発行する前と変わっていないことを示しています。

10. 最後に、warehouse データベースへのリンクを作成し、再度 USER_DB_LINKS を問い合せて、Oracle が部分指定のグローバル名をどのように解決するのかを調べます。

```
CREATE DATABASE LINK warehouse USING 'warehouse';
SELECT DB_LINK FROM USER_DB_LINKS;

DB_LINK
------------------
HQ.US.ACME.COM
SUPPLY.JP.ACME.COM
WAREHOUSE.ASIA.JP.ACME.COM
```

ここでも Oracle は、リンクの作成時にローカル・データベースのグローバル・データベース名のドメイン部分を使用して、部分的なリンク名を拡張していることがわかります。

注意： supply データベース・リンクを訂正するには、必ずこのデータベース・リンクを削除してから再作成してください。

関連項目： DB_NAME および DB_DOMAIN 初期化パラメータの指定の詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

データベース・リンクの作成

アプリケーションによるデータおよびスキーマ・オブジェクトへのアクセスを分散データベース・システム全体にわたってサポートするために、必要なデータベース・リンクをすべて作成する必要があります。この項の内容は、次のとおりです。

- データベース・リンクの作成に必要な権限の取得
- リンク・タイプの指定
- リンク・ユーザーの指定
- リンク名に含まれるサービス名を指定するための接続修飾子の使用

データベース・リンクの作成に必要な権限の取得

データベース・リンクは、リモート・データベースのオブジェクトへのアクセスを可能にするローカル・データベース内のポインタです。プライベート・データベース・リンクを作成するには、あらかじめ適切な権限が付与されている必要があります。次の表は、どのリンク・タイプのデータベースに対してどの権限が必要なのかを示しています。

権限	データベース	この権限を必要とする操作
CREATE DATABASE LINK	ローカル	プライベート・データベース・リンクの作成
CREATE PUBLIC DATABASE LINK	ローカル	パブリック・データベース・リンクの作成
CREATE SESSION	リモート	任意タイプのデータベース・リンクの作成

現在使用可能な権限を確認するには、ROLE_SYS_PRIVS を問い合せます。たとえば、次の privs.sql スクリプトを作成し、実行できます（出力例も含まれています）。

```
SELECT DISTINCT PRIVILEGE AS "Database Link Privileges"
FROM ROLE_SYS_PRIVS
WHERE PRIVILEGE IN ( 'CREATE SESSION','CREATE DATABASE LINK',
                     'CREATE PUBLIC DATABASE LINK')
/

SQL> @privs

Database Link Privileges
----------------------------------------
CREATE DATABASE LINK
CREATE PUBLIC DATABASE LINK
CREATE SESSION
```

リンク・タイプの指定

データベース・リンクを作成するときは、そのデータベース・リンクにアクセスするユーザーを決める必要があります。ここでは、3 種類の基本的なリンク・タイプの作成方法について説明します。

- プライベート・データベース・リンクの作成
- パブリック・データベース・リンクの作成
- グローバル・データベース・リンクの作成

プライベート・データベース・リンクの作成

プライベート・データベース・リンクを作成するには、次の文を指定します。ここで、*link_name* はグローバル・データベース名または任意のリンク名を表します。

```
CREATE DATABASE LINK link_name ...;
```

プライベート・データベース・リンクの例を次に示します。

SQL 文	作成されるプライベート・データベース・リンク
`CREATE DATABASE LINK supply.us.acme.com;`	リモートの supply データベースへの、グローバル・データベース名を使用したプライベート・リンク。 リンクは、接続ユーザーのユーザー ID/ パスワードを使用します。そのため、scott（パスワード：tiger）が問合せの中でこのリンクを使用すると、リモート・データベースへの接続が scott/tiger として確立されます。
`CREATE DATABASE LINK link_2 CONNECT TO jane IDENTIFIED BY doe USING 'us_supply';`	サービス名 us_supply を持つデータベースへの、link_2 という名前のプライベート固定ユーザー・リンク。このリンクは、接続ユーザーとは無関係に、jane/doe というユーザー ID/ パスワードでリモート・データベースに接続します。
`CREATE DATABASE LINK link_1 CONNECT TO CURRENT_USER USING 'us_supply';`	サービス名 us_supply を持つデータベースへの、link_1 という名前のプライベート・リンク。このリンクは、現行ユーザーのユーザー ID/ パスワードを使用してリモート・データベースにログインします。 **注意：**現行ユーザーと接続ユーザーは異なる場合があります。また、現行ユーザーは、リンクに関係している両方のデータベースのグローバル・ユーザーであることが必要です（28-16 ページの「データベース・リンクのユーザー」を参照）。現行ユーザー・リンクは、Oracle Advanced Security オプションの一部です。

関連項目： CREATE DATABASE LINK の構文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

パブリック・データベース・リンクの作成

パブリック・データベース・リンクを作成するには、キーワード PUBLIC を使用します。ここで、*link_name* はグローバル・データベース名または任意のリンク名を表します。

```
CREATE PUBLIC DATABASE LINK link_name ...;
```

パブリック・データベース・リンクの例を次に示します。

SQL 文	作成されるパブリック・データベース・リンク
CREATE PUBLIC DATABASE LINK supply.us.acme.com;	リモートの supply データベースへのパブリック・リンク。リンクは、接続ユーザーのユーザー ID/ パスワードを使用します。そのため、scott（パスワード : tiger）が問合せの中でこのリンクを使用すると、リモート・データベースへの接続が scott/tiger として確立されます。
CREATE PUBLIC DATABASE LINK pu_link CONNECT TO CURRENT_USER USING 'supply';	サービス名 supply を持つデータベースへの、pu_link という名前のパブリック・リンク。このリンクは、現行ユーザーのユーザー ID/ パスワードを使用してリモート・データベースにログインします。 **注意 :** 現行ユーザーと接続ユーザーは異なる場合があります。また、現行ユーザーは、リンクに関係している両方のデータベースのグローバル・ユーザーであることが必要です（28-16 ページの「データベース・リンクのユーザー」を参照）。
CREATE PUBLIC DATABASE LINK sales.us.acme.com CONNECT TO jane IDENTIFIED BY doe;	リモートの sales データベースへのパブリック固定ユーザー・リンク。このリンクは、jane/doe というユーザー ID/ パスワードでリモート・データベースに接続します。

関連項目： CREATE PUBLIC DATABASE LINK の構文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

グローバル・データベース・リンクの作成

グローバル・データベース・リンクは、Oracle Names Server で定義する必要があります。グローバル・データベース・リンクの作成方法の詳細は、『Oracle9*i* Net Services 管理者ガイド』を参照してください。

リンク・ユーザーの指定

データベース・リンクは、あるデータベースから別のデータベースへの通信経路を定義します。アプリケーションがデータベース・リンクを使用してリモート・データベースに接続するとき、Oracle はローカル・アプリケーションの要求にかわってリモート・データベース内でデータベース・セッションを確立します。

プライベートまたはパブリックのデータベース・リンクを作成する際、固定ユーザー、現行ユーザーおよび接続ユーザーのデータベース・リンクを作成することで、リモート・データベースのどのスキーマにリンクが接続を確立するのかを指定できます。

固定ユーザー・データベース・リンクの作成

固定ユーザー・データベース・リンクを作成するには、リモート・データベースへのアクセスに必要な資格証明（この場合はユーザー名とパスワード）をリンク定義に埋め込みます。

```
CREATE DATABASE LINK ... CONNECT TO username IDENTIFIED BY password ...;
```

固定ユーザー・データベース・リンクの例を次に示します。

SQL 文	作成される固定ユーザー・データベース・リンク
`CREATE PUBLIC DATABASE LINK supply.us.acme.com CONNECT TO scott AS tiger;`	リモートの supply データベースへの、グローバル・データベース名を使用したパブリック・リンク。このリンクは、scott/tiger というユーザー ID/ パスワードでリモート・データベースに接続します。
`CREATE DATABASE LINK foo CONNECT TO jane IDENTIFIED BY doe USING 'finance';`	サービス名 finance を持つデータベースへの、foo という名前のプライベート固定ユーザー・リンク。このリンクは、jane/doe というユーザー ID/ パスワードでリモート・データベースに接続します。

アプリケーションで固定ユーザー・データベース・リンクを使用するとき、ローカル・サーバーは必ずリモート・データベース内の固定リモート・スキーマへの接続を確立します。また、ローカル・サーバーはアプリケーションがリンクを使用してリモート・データベースにアクセスするときに、ネットワークを介して固定ユーザーの資格証明を送信します。

接続ユーザーおよび現行ユーザー・データベース・リンクの作成

接続ユーザーおよび現行ユーザー・データベース・リンクでは、リンク定義に資格証明が含まれません。リモート・データベースへの接続に使用する資格証明は、データベース・リンクを参照するユーザーや、アプリケーションが実行する操作によって異なります。

注意： 多くの分散アプリケーションでは、ユーザーがリモート・データベースでの権限を持つ必要はありません。これを実現する簡単な方法の 1 つは、固定ユーザーまたは現行ユーザーのデータベース・リンクを含むプロシージャを作成することです。この方法では、プロシージャにアクセスするユーザーに対して第三者の権限が一時的に付与されます。

接続ユーザーと現行ユーザー間の区別に関する概念の詳細は、28-16 ページの「データベース・リンクのユーザー」を参照してください。

接続ユーザー・データベース・リンクの作成 接続ユーザー・データベース・リンクを作成するには、CONNECT TO 句を省略します。次の構文は、接続ユーザー・データベース・リンクを作成します。ここで、*dblink* はリンクの名前、*net_service_name* はオプションの接続文字列を表します。

```
CREATE [SHARED] [PUBLIC] DATABASE LINK dblink ... [USING 'net_service_name'];
```

たとえば、接続ユーザー・データベース・リンクを作成するために、次の構文を使用します。

```
CREATE DATABASE LINK sales.division3.acme.com USING 'sales';
```

現行ユーザー・データベース・リンクの作成 現行ユーザー・データベース・リンクを作成するには、リンク作成文で CONNECT TO CURRENT_USER 句を使用します。現行ユーザー・リンクは、Oracle Advanced Security オプションでのみ使用できます。

次の構文は、現行ユーザー・データベース・リンクを作成します。ここで、*dblink* はリンクの名前、*net_service_name* はオプションの接続文字列を表します。

```
CREATE [SHARED] [PUBLIC] DATABASE LINK dblink CONNECT TO CURRENT_USER
[USING 'net_service_name'];
```

たとえば、sales データベースへの現行ユーザー・データベース・リンクを作成するには、次の構文を使用します。

```
CREATE DATABASE LINK sales CONNECT TO CURRENT_USER USING 'sales';
```

注意： 現行ユーザー・データベース・リンクを使用するには、リンクに関係している両方のデータベースで現行ユーザーがグローバル・ユーザーであることが必要です。

関連項目： データベース・リンクの作成に関する構文情報の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

リンク名に含まれるサービス名を指定するための接続修飾子の使用

場合によっては、同じリモート・データベースを指すものの、異なる通信経路を使用してリモート・データベースへの接続を確立する同じタイプ（パブリックなど）のデータベース・リンクを複数作成しなければならないことがあります。次のような場合に、この方法が役立ちます。

- リモート・データベースが Oracle Real Application Clusters 構成の一部であり、そのためリモート・データベースの特定のインスタンスに接続を確立できるようにローカル・ノードで複数のパブリック・データベース・リンクを定義する場合
- TCP/IP を使用して Oracle サーバーに接続するクライアントと、DECnet を使用して Oracle サーバーに接続するクライアントがある場合

このような機能を容易に利用できるように、Oracle ではデータベース・リンク名の中でオプションのサービス名を使用してデータベース・リンクを作成できます。データベース・リンクを作成するときは、サービス名をデータベース・リンク名の末尾部分に指定し、アットマーク（@）で区切ります。たとえば、`@sales` のようにします。この文字列のことを**接続修飾子**と呼びます。

たとえば、リモート・データベース `hq.acme.com` が Oracle Real Application Clusters 環境で管理されているとします。`hq` データベースには、`hq_1` および `hq_2` という名前の 2 つのインスタンスがあります。ローカル・データベースで次のパブリック・データベース・リンクを作成して、`hq` データベースのリモート・インスタンスへの経路を定義できます。

```
CREATE PUBLIC DATABASE LINK hq.acme.com@hq_1
  USING 'string_to_hq_1';
CREATE PUBLIC DATABASE LINK hq.acme.com@hq_2
  USING 'string_to_hq_2';
CREATE PUBLIC DATABASE LINK hq.acme.com
  USING 'string_to_hq';
```

最初の 2 つの例では、サービス名が単にデータベース・リンク名の一部であることに注意してください。サービス名のテキストは、必ずしも接続の確立方法を示す必要はありません。この情報は、`USING` 句のサービス名で指定します。また、3 番目の例ではサービス名がリンク名の一部として指定されていません。この場合は、サービス名をリンク名の一部として指定したときと同様に、`USING` 文字列によってインスタンスが決まります。

サービス名を使用して特定のインスタンスを指定するには、サービス名をグローバル・オブジェクト名の末尾に付けます。

```
SELECT * FROM scott.emp@hq.acme.com@hq_1
```

この例では、2 つのアットマーク（@）があることに注意してください。

共有データベース・リンクの作成

標準のデータベース・リンクを使用してリモート・サーバーを参照するアプリケーションはすべて、ローカル・データベースとリモート・データベースの間で接続を確立します。多くのユーザーがアプリケーションを同時に実行した場合、ローカル・データベースとリモート・データベースの間で多数の接続が発生する可能性があります。

共有データベース・リンクを使用すると、ローカル・サーバーとリモート・サーバーの間で必要なネットワーク接続の数を制限できます。

この項の内容は、次のとおりです。

- 共有データベース・リンクの使用の判断
- 共有データベース・リンクの作成
- 共有データベース・リンクの構成

関連項目： 共有データベース・リンクの概要の詳細は、28-10 ページの「共有データベース・リンクの概要」を参照してください。

共有データベース・リンクの使用の判断

アプリケーションおよび共有サーバーの構成を綿密に検討して、共有リンクを使用するかどうかを判断してください。簡単なガイドラインとして、データベース・リンクにアクセスするユーザー数がローカル・データベース内のサーバー・プロセス数よりもかなり多いと予測されるときは、共有データベース・リンクを使用します。

データベース・リンクに関して可能な 3 つの構成を次の表に示します。

リンク・タイプ	サーバー・モード	結果
非共有	専用 / 共有サーバー	アプリケーションで標準のパブリック・データベース・リンクを使用していて、100 人のユーザーが同時に接続を要求した場合は、リモート・データベースへの直接ネットワーク接続が 100 必要になります。
共有	共有サーバー	ローカルの共有サーバー・モード・データベースに 10 の共有サーバー・プロセスが存在している場合、同じデータベース・リンクを使用するユーザーが 100 人いるときに必要となるリモート・サーバーへのネットワーク接続数は 10 以下になります。ローカルの各共有サーバー・プロセスが必要とするリモート・サーバーへの接続は、それぞれ 1 つで済む場合があります。

リンク・タイプ	サーバー・モード	結果
共有	専用	10 のクライアントがローカルの専用サーバーに接続していて、各クライアントが同じ接続のセッションを 10 持っており（したがって全部で 100 のセッションを確立している）、各セッションで同じリモート・データベースを参照している場合、必要な接続は 10 で済みます。非共有データベース・リンクでは、100 の接続が必要になります。

共有データベース・リンクは、必ずしもすべての状況で役立つとはかぎりません。たとえば、リモート・サーバーに接続するユーザーが 1 人のみの場合を考えます。このユーザーが共有データベース・リンクを定義し、ローカル・データベースに 10 個の共有サーバー・プロセスが存在する場合、このユーザーはリモート・サーバーへのネットワーク接続を最大 10 個必要とする可能性があります。ユーザーは個々の共有サーバー・プロセスを使用できるため、各プロセスはそれぞれリモート・サーバーへの接続を確立できます。

非共有のデータベース・リンクでは 1 つのネットワーク接続しか必要としないため、このような状況では明らかに非共有のデータベース・リンクのほうが望ましいといえます。共有データベース・リンクでは、シングル・ユーザーの場合に多くのネットワーク接続が発生します。したがって、共有リンクは、多数のユーザーが同じリンクを使用する必要があるときのみ使用してください。通常、共有リンクはパブリック・データベース・リンクで使用しますが、多数のクライアントが同じローカル・スキーマ（したがって同じプライベート・データベース・リンク）にアクセスするときは、プライベート・データベース・リンクでも使用できます。

共有データベース・リンクの作成

共有データベース・リンクを作成するには、CREATE DATABASE LINK 文でキーワード SHARED を使用します。

```
CREATE SHARED DATABASE LINK dblink_name
[CONNECT TO username IDENTIFIED BY password] | [CONNECT TO CURRENT_USER]
AUTHENTICATED BY schema_name IDENTIFIED BY password
[USING 'service_name'];
```

次の例では、scott として接続し、keith として認証される、sales データベースへの固定ユーザー共有リンクを作成しています。

```
CREATE SHARED DATABASE LINK link2sales
CONNECT TO scott IDENTIFIED BY tiger
AUTHENTICATED BY keith IDENTIFIED BY richards
USING 'sales';
```

キーワード SHARED を使用するときは、必ず AUTHENTICATED BY 句が必要です。AUTHENTICATED BY 句で指定するスキーマはセキュリティ上の理由でのみ使用されるもので、ダミーのスキーマとみなすことができます。このスキーマは、共有データベース・リンクを使用するときに何も効果はなく、共有データベース・リンクのユーザーに影響を与えることもありません。AUTHENTICATED BY 句は、認可されていないクライアントがデータベース・リンクのユーザーになりすまして権限を要する情報にアクセスするのを防ぐために必要です。

関連項目： CREATE DATABASE LINK 文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

共有データベース・リンクの構成

共有データベース・リンクは、次の方法で構成できます。

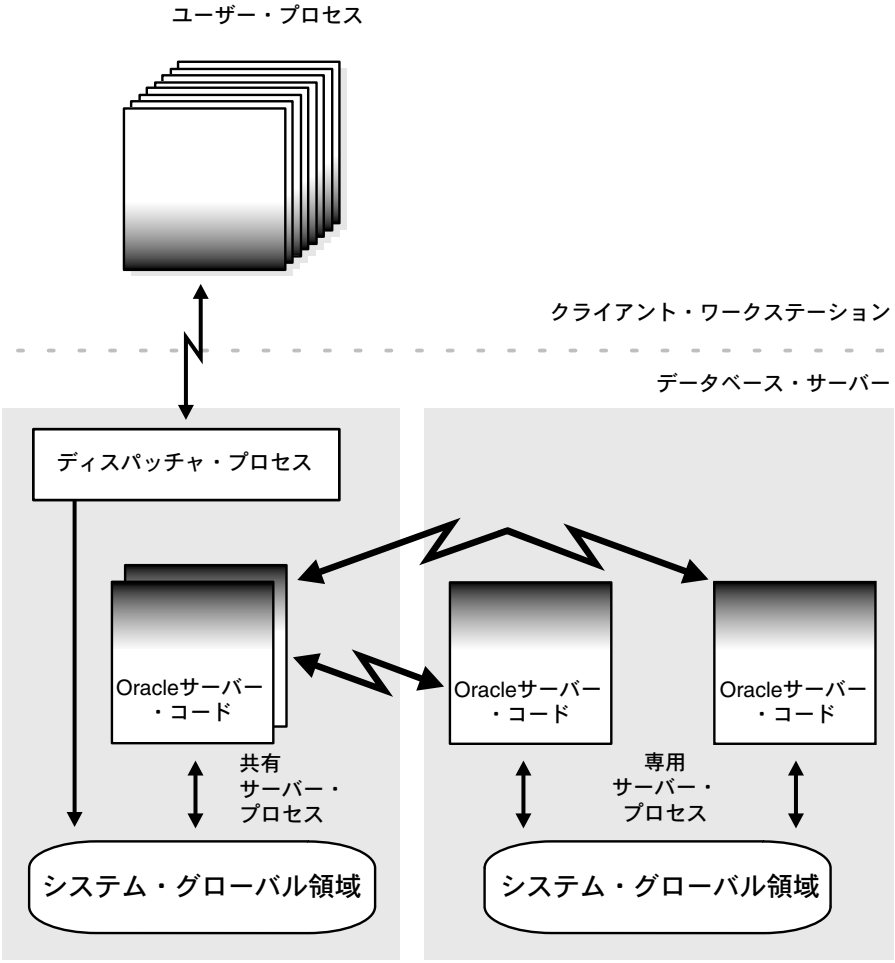
- 専用サーバーへの共有リンクの作成
- 共有サーバーへの共有リンクの作成

専用サーバーへの共有リンクの作成

図 29-1 に示す構成では、ローカル・サーバーの共有サーバー・プロセスは専用のリモート・サーバー・プロセスを所有しています。この構成の利点は、ローカルの共有サーバーとリモートの専用サーバーとの間に直接的なネットワーク・トランスポートが存在することです。ただし、余分なバックエンド・サーバー・プロセスが必要になるという欠点もあります。

注意： リモート・サーバーは、共有サーバーまたは専用サーバーのどちらか一方にできます。ローカル・サーバーとリモート・サーバーの間には専用の接続が存在します。リモート・サーバーが共有サーバーのときは、サービス名の定義に SERVER=DEDICATED 句を使用することで、専用サーバー接続を強制できます。

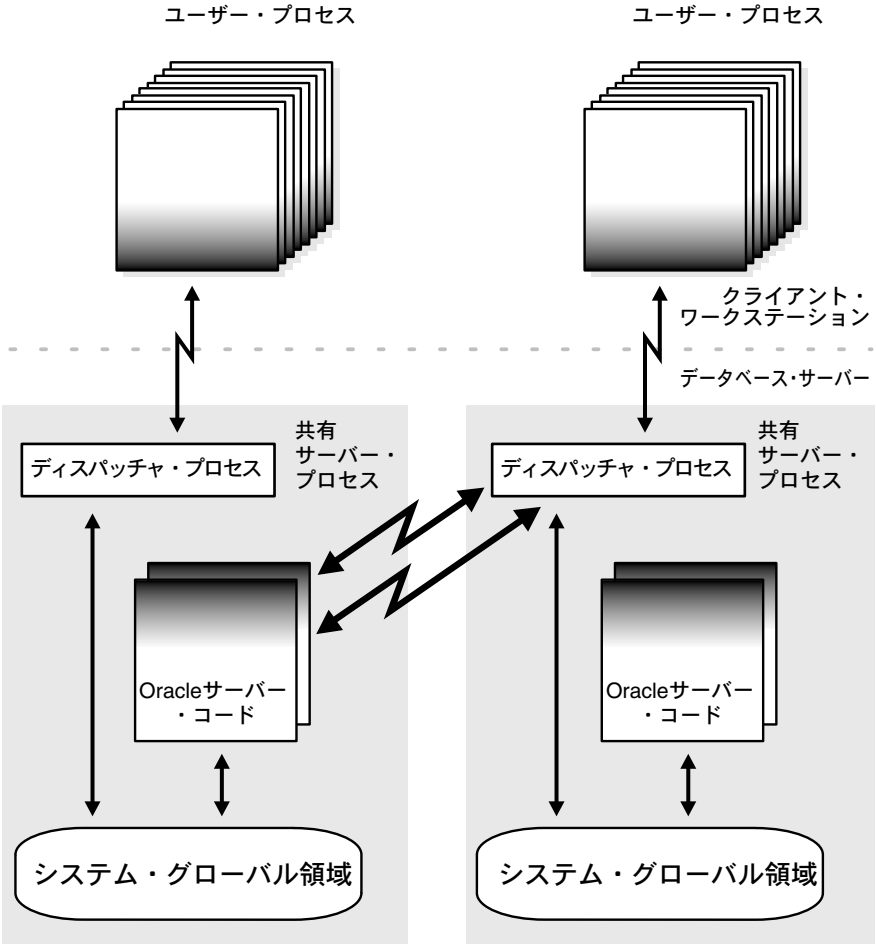
図 29-1　専用サーバー・プロセスへの共有データベース・リンク



共有サーバーへの共有リンクの作成

図 29-2 に示す構成では、リモート・サーバーで共有サーバー・プロセスを使用しています。この構成では、多数の専用サーバー・プロセスは必要ありませんが、リモート・サーバーのディスパッチャを経由する接続が必要になります。この場合、ローカル・サーバーとリモート・サーバーの両方を共有サーバーとして構成する必要があります。

図 29-2 共有サーバーへの共有データベース・リンク



関連項目： 共有サーバー・オプションの詳細は、『Oracle9*i* Net Services 管理者ガイド』を参照してください。

データベース・リンクの管理

この項の内容は、次のとおりです。

データベース・リンクのクローズ

あるセッションでデータベース・リンクにアクセスする場合、そのセッションをクローズするまでリンクはオープンしたままです。リンクがオープンしているということは、そのリンクを介してアクセスしている各リモート・データベースのプロセスがアクティブであることを意味します。この状況では、次のような結果が生じます。

- 20 人のユーザーがセッションをオープンしてローカル・データベースの同じパブリック・リンクにアクセスする場合、20 のデータベース・リンク接続がオープンします。
- 20 人のユーザーがセッションをオープンして各ユーザーがプライベート・リンクにアクセスする場合、20 のデータベース・リンク接続がオープンします。
- 1 人のユーザーがセッションを開始して 20 の異なるリンクにアクセスする場合、20 のデータベース・リンク接続がオープンします。

セッションをクローズした後、セッションでアクティブだったリンクは自動的にクローズされます。ユーザーがリンクを手動でクローズする場合もあります。たとえば、次のようなときにリンクをクローズします。

- リンクによって確立されたネットワーク接続がアプリケーションでそれほど頻繁に使用されないとき
- ユーザー・セッションを終了する必要があるとき

リンクをクローズする場合は、次の文を発行します。ここで、*linkname* はリンクの名前を表します。

```
ALTER SESSION CLOSE DATABASE LINK linkname;
```

この文は、現行セッションでアクティブなリンクのみをクローズします。

データベース・リンクの削除

データベース・リンクは、表やビューと同様に削除できます。リンクがプライベートの場合は、自分のスキーマ内に存在する必要があります。リンクがパブリックの場合は、DROP PUBLIC DATABASE LINK システム権限が必要です。

構文は次のとおりです。ここで、*dblink* はリンクの名前を表します。

```
DROP [PUBLIC] DATABASE LINK dblink;
```

プライベート・データベース・リンクの削除手順

1. SQL*Plus を使用して、ローカル・データベースに接続します。たとえば、次のように入力します。

   ```
   CONNECT scott/tiger@local_db
   ```

2. USER_DB_LINKS を問い合せて、所有しているリンクを確認します。たとえば、次のように入力します。

   ```
   SELECT DB_LINK FROM USER_DB_LINKS;

   DB_LINK
   -----------------------------------
   SALES.US.ORACLE.COM
   MKTG.US.ORACLE.COM
   2 rows selected.
   ```

3. DROP DATABASE LINK 文を使用して、目的のリンクを削除します。たとえば、次のように入力します。

   ```
   DROP DATABASE LINK sales.us.oracle.com;
   ```

パブリック・データベース・リンクの削除手順

1. ローカル・データベースに DROP PUBLIC DATABASE LINK 権限を持つユーザーとして接続します。たとえば、次のように入力します。

   ```
   CONNECT SYSTEM/password@local_db AS SYSDBA
   ```

2. DBA_DB_LINKS を問い合せて、パブリック・リンクを確認します。たとえば、次のように入力します。

   ```
   SELECT DB_LINK FROM USER_DB_LINKS
     WHERE OWNER = 'PUBLIC';

   DB_LINK
   -----------------------------------
   DBL1.US.ORACLE.COM
   SALES.US.ORACLE.COM
   INST2.US.ORACLE.COM
   ```

```
RMAN2.US.ORACLE.COM
4 rows selected.
```

3. DROP PUBLIC DATABASE LINK 文を使用して、目的のリンクを削除します。たとえば、次のように入力します。

```
DROP PUBLIC DATABASE LINK sales.us.oracle.com;
```

アクティブ・データベース・リンクの接続数の制限

静的な初期化パラメータ OPEN_LINKS を使用すると、ユーザー・プロセスからリモート・データベースへの接続数を制限できます。このパラメータは、分散トランザクションにおいて 1 つのユーザー・セッションが同時に使用できるリモート接続数を制御します。

このパラメータを設定する際は、次の点を考慮してください。

- 設定値は、複数のデータベースを参照する 1 つの SQL 文によって参照されるデータベースの数以上にします。
- 複数の分散データベースに継続してアクセスする場合は、設定値を大きくします。たとえば、3 つのデータベースに定期的にアクセスする場合は、OPEN_LINKS を 3 以上に設定します。
- OPEN_LINKS のデフォルト値は 4 です。OPEN_LINKS を 0（ゼロ）に設定した場合、分散トランザクションは許可されません。

関連項目： OPEN_LINKS の詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

データベース・リンク情報の表示

各データベースのデータ・ディクショナリには、そのデータベース内にあるデータベース・リンクの定義がすべて格納されています。データ・ディクショナリ表およびビューを使用して、リンクに関する情報を取得できます。この項の内容は、次のとおりです。

- データベース内のリンクの判断
- オープンしているリンク接続の判断

データベース内のリンクの判断

次のビューには、ローカル・データベースで定義され、データ・ディクショナリに格納されているデータベース・リンクが表示されます。

ビュー	用途
DBA_DB_LINKS	データベース内のデータベース・リンクがすべてリストされます。
ALL_DB_LINKS	接続ユーザーがアクセス可能なデータベース・リンクがすべてリストされます。
USER_DB_LINKS	接続ユーザーが所有しているデータベース・リンクがすべてリストされます。

これらのデータ・ディクショナリ・ビューには、データベース・リンクに関する同じ基本情報が含まれていますが、いくつか例外もあります。

列	対象となるビュー	説明
OWNER	USER_* を除くすべて	データベース・リンクを作成したユーザー。リンクがパブリックの場合、ユーザーは PUBLIC としてリストされます。
DB_LINK	すべて	データベース・リンクの名前。
USERNAME	すべて	リンク定義に固定ユーザーが含まれている場合、この列には固定ユーザーのユーザー名が表示されます。固定ユーザーが含まれていない場合、この列は NULL になります。
PASSWORD	USER_* のみ	リモート・データベースにログインするためのパスワード。
HOST	すべて	リモート・データベースへの接続に使用されるネット・サービス名。
CREATED	すべて	データベース・リンクの作成日時。

どのユーザーでも USER_DB_LINKS を問い合せることで、そのユーザーが使用できるデータベース・リンクを判断できます。ALL_DB_LINKS ビューまたは DBA_DB_LINKS ビューを使用できるのは、追加の権限を持つユーザーのみです。

次のスクリプトは、DBA_DB_LINKS ビューを問い合せて、リンク情報にアクセスします。

```
COL OWNER FORMAT a10
COL USERNAME FORMAT A8 HEADING "USER"
COL DB_LINK FORMAT A30
COL HOST FORMAT A7 HEADING "SERVICE"
SELECT * FROM DBA_DB_LINKS
/
```

ここでスクリプトが起動され、その出力結果が表示されます。

```
SQL>@link_script

OWNER      DB_LINK                        USER     SERVICE CREATED
---------- ------------------------------ -------- ------- ----------
SYS        TARGET.US.ACME.COM             SYS      inst1   23-JUN-99
PUBLIC     DBL1.UK.ACME.COM               BLAKE    ora51   23-JUN-99
PUBLIC     RMAN2.US.ACME.COM                       inst2   23-JUN-99
PUBLIC     DEPT.US.ACME.COM                        inst2   23-JUN-99
JANE       DBL.UK.ACME.COM                BLAKE    ora51   23-JUN-99
SCOTT      EMP.US.ACME.COM                SCOTT    inst2   23-JUN-99
6 rows selected.
```

パスワード情報表示の認可

パスワード情報の列があるのは、USER_DB_LINKS のみです。しかし、管理ユーザー（SYS、または AS SYSDBA として接続しているユーザー）であれば、LINK$ 表を問い合せることで、データベース内にあるすべてのリンクのパスワードを表示できます。管理ユーザー以外のユーザーは、次のどちらかの方法によって、LINK$ 表を問い合せるための権限を取得できます。

- LINK$ 表に対する特定のオブジェクト権限を付与される方法
- SELECT ANY DICTIONARY システム権限を付与される方法

関連項目： SYS スキーマのオブジェクトの表示に必要な権限の詳細は、25-2 ページの「ユーザー権限とロールの理解」を参照してください。

パスワード情報の表示

次のスクリプトを作成し、SQL*Plus で実行することで、パスワード情報を取得できます（出力例も含まれています）。

```
COL USERID FORMAT A10
COL PASSWORD FORMAT A10
SELECT USERID,PASSWORD
  FROM SYS.LINK$
  WHERE PASSWORD IS NOT NULL
/

SQL>@linkpwd

USERID     PASSWORD
---------- ----------
SYS        ORACLE
BLAKE      TYGER
SCOTT      TIGER
3 rows selected.
```

認証パスワードの表示

LINK$ 表を問い合せることにより、データベース内にあるすべてのリンクについて、AUTHENTICATED BY ... IDENTIFIED BY ... で指定されたユーザー名とパスワードを表示できます。次のスクリプトを作成し、SQL*Plus で実行することで、パスワード情報を取得できます（出力例も含まれています）。

```
COL AUTHUSR FORMAT A10
COL AUTHPWD FORMAT A10
SELECT AUTHUSR AS userid, AUTHPWD AS password
  FROM SYS.LINK$
  WHERE PASSWORD IS NOT NULL
/

SQL> @authpwd

USERID     PASSWORD
---------- ----------
ELLIE      MAY
1 row selected.
```

また、次のスクリプトを作成し、実行することで、リンクとパスワードの情報を結合し、まとめて表示できます（出力例も含まれています）。

```
COL OWNER FORMAT A8
COL DB_LINK FORMAT A15
COL USERNAME FORMAT A8 HEADING "CON_USER"
COL PASSWORD FORMAT A8 HEADING "CON_PWD"
COL AUTHUSR FORMAT A8 HEADING "AUTH_USER"
COL AUTHPWD FORMAT A8 HEADING "AUTH_PWD"
COL HOST FORMAT A7 HEADING "SERVICE"
COL CREATED FORMAT A10

SELECT DISTINCT d.OWNER,d.DB_LINK,d.USERNAME,l.PASSWORD,
                l.AUTHUSR,l.AUTHPWD,d.HOST,d.CREATED
FROM DBA_DB_LINKS d, SYS.LINK$ l
WHERE PASSWORD IS NOT NULL
AND d.USERNAME = l.USERID
/

SQL> @user_and_pwd

OWNER    DB_LINK         CON_USER CON_PWD  AUTH_USE AUTH_PWD SERVICE CREATED
-------- --------------- -------- -------- -------- -------- ------- ----------
JANE     DBL.ACME.COM    BLAKE    TYGER    ELLIE    MAY      ora51   23-JUN-99
PUBLIC   DBL1.ACME.COM   SCOTT    TIGER                      ora51   23-JUN-99
SYS      TARGET.ACME.COM SYS      ORACLE                     inst1   23-JUN-99
```

オープンしているリンク接続の判断

自分のセッションで現在オープンしているデータベース・リンク接続がわかれば役に立つ場合があります。しかし、SYSDBA として接続している場合には、ビューを問い合せてすべてのセッションでオープンしているすべてのリンクを判断することはできず、現在作業中のセッションのリンク情報にアクセスすることしかできません。

次のビューには、現行セッションで現在オープンしているデータベース・リンク接続が表示されます。

ビュー	用途
V$DBLINK	自分のセッションでオープンしているすべてのデータベース・リンク、つまり、IN_TRANSACTION 列が YES に設定されているすべてのデータベース・リンクがリストされます。
GV$DBLINK	自分のセッションでオープンしているすべてのデータベース・リンクと対応するインスタンスがリストされます。このビューは、Oracle Real Application Clusters 構成の使用時に役立ちます。

これらのデータ・ディクショナリ・ビューには、データベース・リンクに関する同じ基本情報が含まれていますが、1 つ例外があります。

列	対象となるビュー	説明
DB_LINK	すべて	データベース・リンクの名前
OWNER_ID	すべて	データベース・リンクの所有者
LOGGED_ON	すべて	データベース・リンクが現在ログインされているかどうか
HETEROGENEOUS	すべて	データベース・リンクが同機種間（NO）と異機種間（YES）のどちらであるか
PROTOCOL	すべて	データベース・リンクの通信プロトコル
OPEN_CURSORS	すべて	データベース・リンクでカーソルがオープンされているかどうか
IN_TRANSACTION	すべて	コミットまたはロールバックがまだ完了していないトランザクションで、データベース・リンクがアクセスされているかどうか
UPDATE_SENT	すべて	データベース・リンクで更新があったかどうか
COMMIT_POINT_STRENGTH	すべて	データベース・リンクを使用しているトランザクションのコミット・ポイント強度
INST_ID	GV$DBLINK のみ	ビュー情報が取得されたインスタンス

たとえば、次のスクリプトを作成し、実行することで、オープンされているリンクを判断できます（出力例も含まれています）。

```
COL DB_LINK FORMAT A25
COL OWNER_ID FORMAT 99999 HEADING "OWNID"
COL LOGGED_ON FORMAT A5 HEADING "LOGON"
COL HETEROGENEOUS FORMAT A5 HEADING "HETER"
COL PROTOCOL FORMAT A8
COL OPEN_CURSORS FORMAT 999 HEADING "OPN_CUR"
COL IN_TRANSACTION FORMAT A3 HEADING "TXN"
COL UPDATE_SENT FORMAT A6 HEADING "UPDATE"
COL COMMIT_POINT_STRENGTH FORMAT 99999 HEADING "C_P_S"

SELECT * FROM V$DBLINK
/

SQL> @dblink
```

```
DB_LINK                   OWNID LOGON HETER PROTOCOL OPN_CUR TXN UPDATE  C_P_S
------------------------- ------ ----- ----- -------- ------- --- ------ ------
INST2.ACME.COM                 0 YES   YES    UNKN          0 YES YES       255
```

位置の透過性の作成

必要なデータベース・リンクの構成が完了した後、各種のツールを使用してデータベース・システムの分散的な性質をユーザーから隠すことができます。言い換えれば、ユーザーはリモート・オブジェクトに対してあたかもローカル・オブジェクトであるかのようにアクセスできるようになります。ここでは、分散機能をユーザーから隠す方法について説明します。

- ビューを使用した位置の透過性の作成
- シノニムを使用した位置の透過性の作成
- プロシージャを使用した位置の透過性の作成

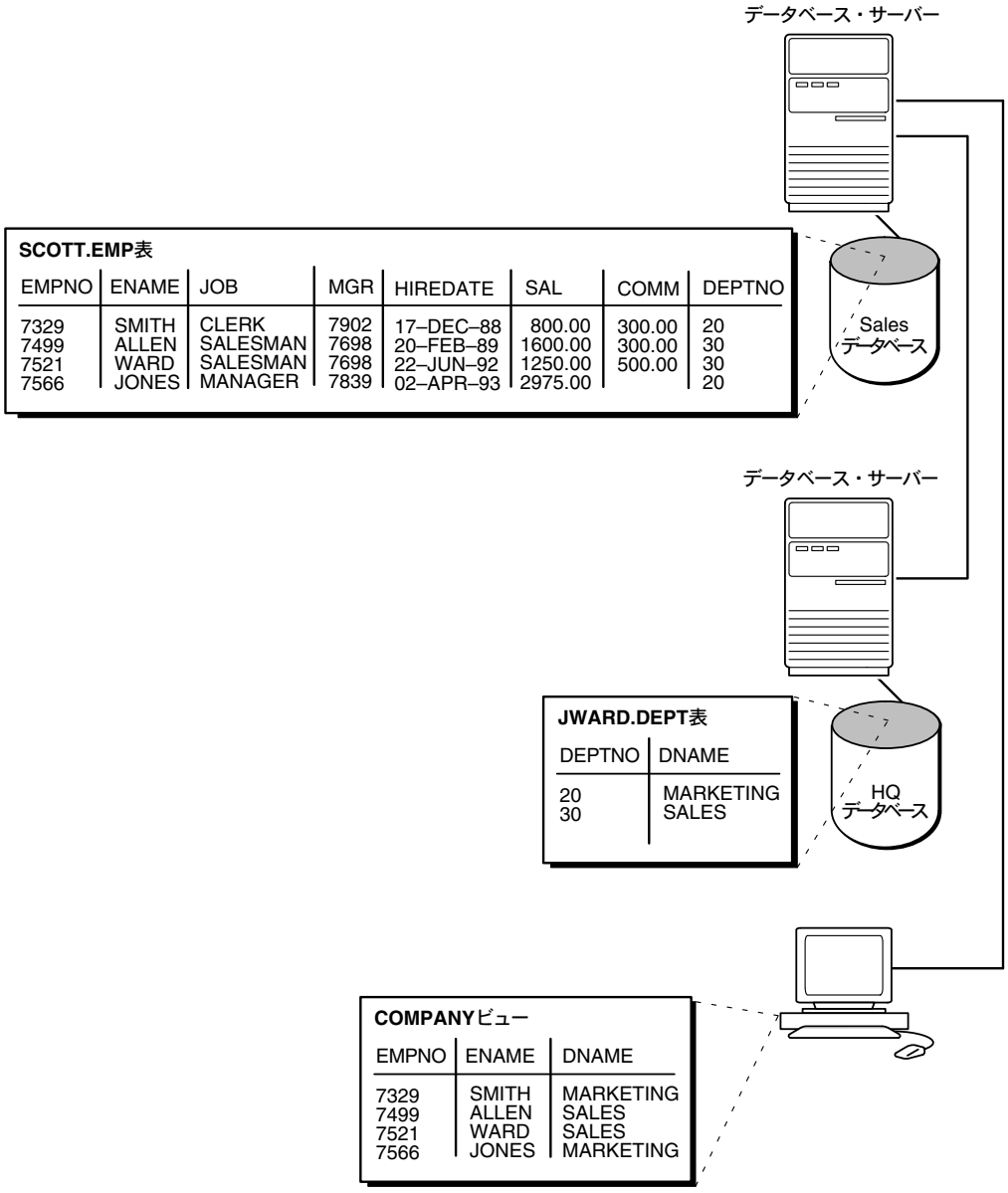
ビューを使用した位置の透過性の作成

ローカル・ビューは、分散データベース・システムにおいてローカルおよびリモートの表に対する位置の透過性を提供できます。

たとえば、emp 表がローカル・データベースに、dept 表がリモート・データベースに、それぞれ格納されているとします。システムのユーザーに対してこれらの表を透過的にするために、ローカル・データとリモート・データを結合するビューをローカル・データベースに作成できます。

```
CREATE VIEW company AS
   SELECT a.empno, a.ename, b.dname
   FROM scott.emp a, jward.dept@hq.acme.com b
   WHERE a.deptno = b.deptno;
```

図 29-3　ビューと位置の透過性



ユーザーはこのビューにアクセスするときに、データが物理的に格納されている場所や、複数の表のデータにアクセスしているかどうかについて知る必要はありません。したがって、ユーザーは必要な情報をより簡単に取得できます。たとえば、次の問合せは、ローカルおよびリモートの両方のデータベース表からのデータを提供します。

```
SELECT * FROM company;
```

ローカル・ビューの所有者は、リモート・ユーザーによってすでに付与されているローカル・ビューのオブジェクト権限のみを付与できます（リモート・ユーザーはデータベース・リンクのタイプによって示されます）。この仕組みは、ローカル・データを参照するビューの権限の管理と似ています。

シノニムを使用した位置の透過性の作成

シノニムは、分散データベース・システム内での位置など、基礎となるオブジェクトの個別情報を隠すので、分散環境と非分散環境の両方で役立ちます。基礎となるオブジェクトを名前変更または移動する必要がある場合でも、シノニムを再定義するだけで済み、シノニムをベースとするアプリケーションは引き続き通常どおり動作します。また、分散データベース・システムのユーザーが使用する SQL 文も、シノニムによって単純化されます。

シノニムの作成

次のもののシノニムを作成できます。

- 表
- 型
- ビュー
- 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
- パッケージ

シノニムはすべて、作成するデータベースのデータ・ディクショナリに格納されるスキーマ・オブジェクトです。シノニムでは、データベース・リンクを介したリモート表へのアクセスを簡単にするために、単一ワードでリモート・データにアクセスでき、それによって特定のオブジェクト名や位置をシノニムのユーザーから隠すことができます。

シノニムを作成するための構文は、次のとおりです。

```
CREATE [PUBLIC] synonym_name
FOR [schema.]object_name[@database_link_name];
```

各項目の意味は次のとおりです。

- PUBLIC は、すべてのユーザーがこのシノニムを使用できることを指定するキーワードです。このパラメータを省略するとシノニムがプライベートになり、作成者のみ使用可能になります。パブリック・シノニムは、CREATE PUBLIC SYNONYM システム権限を持つユーザーのみが作成できます。
- *synonym_name* には、ユーザーおよびアプリケーションが参照する代替オブジェクト名を指定します。
- *schema* には、*object_name* で指定されたオブジェクトのスキーマを指定します。このパラメータを省略すると、オブジェクトのスキーマとして作成者のスキーマが使用されます。
- *object_name* には、表、ビュー、順序、マテリアライズド・ビュー、型、プロシージャ、ファンクションまたはパッケージのいずれかを適宜指定します。
- *database_link_name* には、*object_name* で指定されたオブジェクトが存在するリモート・データベースおよびスキーマを識別するデータベース・リンクを指定します。

シノニムには、そのスキーマ内に存在するオブジェクトの中で一意の名前を付ける必要があります。スキーマにスキーマ・オブジェクトがあり、同じ名前のパブリック・シノニムが存在する場合、Oracle はスキーマを所有しているユーザーがその名前を参照するときに、常にスキーマ・オブジェクトの方を検索します。

例 : パブリック・シノニムの作成

分散データベース・システム内のすべてのデータベースにおいて、hq データベースに格納されている scott.emp 表のパブリック・シノニムが定義されているとします。

```
CREATE PUBLIC SYNONYM emp FOR scott.emp@hq.acme.com;
```

表 scott.emp@hq.acme.com の位置はパブリック・シノニムによって隠されているので、使用場所に依存しない従業員管理アプリケーションを設計できます。アプリケーション内の SQL 文は、パブリック・シノニム emp を参照して表にアクセスできます。

また、emp 表を hq データベースから hr データベースに移動する場合も、システムのノードでパブリック・シノニムを変更するだけで済みます。従業員管理アプリケーションは、すべてのノードで引き続き正常に動作します。

権限とシノニムの管理

シノニムは、実際のオブジェクトへの参照です。特定のスキーマ・オブジェクトのシノニムにアクセスするユーザーは、元のスキーマ・オブジェクトそのものに対する権限を持っている必要があります。たとえば、ユーザーがシノニムにアクセスしようとして、そのシノニムが識別する表に対してユーザーが権限を持っていなければ、表またはビューが存在しないというエラーが発生します。

scott がリモート・オブジェクト scott.emp@sales.acme.com の別名としてローカル・シノニム emp を作成したとします。この場合、scott は自分以外のローカル・ユーザーにシノニムのオブジェクト権限を付与することはできません。また、scott はシノニムのローカル権限を付与することもできません。この操作は最終的に sales データベースのリモートの emp 表の権限を付与することになりますが、そのような操作は許可されていないためです。この動作は、ローカルの表またはビューの別名であるシノニムの権限管理とは異なります。

したがって、シノニムを位置の透過性のために使用する場合、ローカル権限は管理できません。ベース・オブジェクトのセキュリティは、リモート・ノードですべて管理されます。たとえば、ユーザー admin は EMP_SYN シノニムのオブジェクト権限を付与することはできません。

ビューやプロシージャの定義で参照されるデータベース・リンクとは異なり、シノニムで参照されるデータベース・リンクを解決する際は、シノニムへの参照が解析される時点で有効なスキーマが所有しているプライベート・リンクが最初に検索されます。したがって、オブジェクトの適切な解決を保証するには、基礎となるオブジェクトのスキーマをシノニムの定義の中で指定することが特に重要になります。

プロシージャを使用した位置の透過性の作成

プロシージャと呼ばれる PL/SQL プログラム・ユニットは、位置の透過性を提供できます。それには、次の方法があります。

- ローカル・プロシージャを使用したリモート・データの参照
- ローカル・プロシージャを使用したリモート・プロシージャのコール
- ローカル・シノニムを使用したリモート・プロシージャの参照

ローカル・プロシージャを使用したリモート・データの参照

プロシージャおよびファンクションには、スタンドアロンとパッケージのどちらの場合でも、リモート・データを参照する SQL 文を含めることができます。たとえば、次の文で作成されるプロシージャを考えます。

```
CREATE PROCEDURE fire_emp (enum NUMBER) AS
BEGIN
  DELETE FROM emp@hq.acme.com
  WHERE empno = enum;
END;
```

ユーザーまたはアプリケーションが fire_emp プロシージャをコールするときに、リモート表が変更されようとしていることは見かけ上わかりません。

プロシージャ内の文でローカルのプロシージャ、ビューまたはシノニムを使用して間接的にリモート・データを参照するときは、2 層目の位置の透過性が可能です。たとえば、次の文はローカル・シノニムを定義します。

```
CREATE SYNONYM emp FOR emp@hq.acme.com;
```

このシノニムがあれば、次の文を使用して fire_emp プロシージャを作成できます。

```
CREATE PROCEDURE fire_emp (enum NUMBER) AS
BEGIN
  DELETE FROM emp WHERE empno = enum;
END;
```

表 emp@hq.acme.com を名前変更または移動した場合でも、表を参照するローカル・シノニムを変更するだけで済みます。このプロシージャをコールするプロシージャやアプリケーションを変更する必要はありません。

ローカル・プロシージャを使用したリモート・プロシージャのコール

ローカル・プロシージャを使用してリモート・プロシージャをコールできます。リモート・プロシージャでは、要求されたデータ操作言語（DML）を実行できます。たとえば、scott が local_db に接続して次のプロシージャを作成したとします。

```
CONNECT scott/tiger@local_db

CREATE PROCEDURE fire_emp (enum NUMBER)
AS
BEGIN
  EXECUTE term_emp@hq.acme.com;
END;
```

ここで、scott は、リモート・データベースに接続して次のリモート・プロシージャを作成するとします。

```
CONNECT scott/tiger@hq.acme.com

CREATE PROCEDURE term_emp (enum NUMBER)
AS
BEGIN
  DELETE FROM emp WHERE empno = enum;
END;
```

ユーザーまたはアプリケーションが local_db に接続して fire_emp プロシージャをコールすると、このプロシージャはさらに hq.acme.com にあるリモートの term_emp プロシージャをコールします。

ローカル・シノニムを使用したリモート・プロシージャの参照

たとえば、scott がローカルの sales.acme.com データベースに接続して次のプロシージャを作成したとします。

```
CREATE PROCEDURE fire_emp (enum NUMBER) AS
BEGIN
DELETE FROM emp@hq.acme.com
WHERE empno = enum;
END;
```

この後、ユーザー peggy が supply.acme.com データベースに接続し、scott がリモートの sales データベースで作成したプロシージャに対する次のシノニムを作成します。

```
SQL> CONNECT peggy/hill@supply
SQL> CREATE PUBLIC SYNONYM emp FOR scott.fire_emp@sales.acme.com;
```

supply のローカル・ユーザーは、このシノニムを使用して sales のプロシージャを実行できます。

プロシージャと権限の管理

ローカル・プロシージャに、リモートの表またはビューを参照する文が含まれているとします。ローカル・プロシージャの所有者は、EXECUTE 権限を任意のユーザーに付与することができます。これにより、そのユーザーには、プロシージャを実行する権限とリモート・データに間接的にアクセスする許可が与えられます。

一般に、プロシージャはセキュリティの面で役に立ちます。プロシージャの中で参照されるオブジェクトの権限を明示的にコール側のユーザーに付与する必要はありません。

文の透過性の管理

Oracle では、次に示す標準の DML 文でリモート表を参照することが可能です。

- SELECT（問合せ）
- INSERT
- UPDATE
- DELETE
- SELECT ... FOR UPDATE（異機種間システムでは常にサポートされるとはかぎらない）
- LOCK TABLE

結合、集計、副問合せおよび SELECT ... FOR UPDATE を含む問合せでは、ローカルおよびリモートの表およびビューをいくつでも参照できます。たとえば、次の問合せは 2 つのリモート表の情報を結合します。

```
SELECT e.empno, e.ename, d.dname
  FROM scott.emp@sales.division3.acme.com e, jward.dept@hq.acme.com d
  WHERE e.deptno = d.deptno;
```

UPDATE、INSERT、DELETE および LOCK TABLE 文は、ローカルおよびリモートの両方の表を参照できます。リモート・データを更新するためのプログラミングは不要です。たとえば、次の文はローカル・データベースの jward スキーマの emp 表から行を選択して、scott.sales スキーマのリモート表 emp に新しい行を挿入します。

```
INSERT INTO scott.emp@sales.division3.acme.com
  SELECT * FROM jward.emp;
```

制限事項：

文の透過性には、いくつかの制限事項が適用されます。

- 1 つの SQL 文において、参照されるすべての LONG および LONG RAW の列、順序、更新済みの表およびロック済みの表は、同じノードに存在する必要があります。
- Oracle では、同機種システムにおけるリモートのデータ定義言語（DDL）文（CREATE、ALTER、DROP など）は使用できません。ただし、次の例のように DBMS_SQL パッケージのプロシージャのリモート実行を使用する場合を除きます。

  ```
  DBMS_SQL.PARSE@link_name(crs, 'drop table emp', v7);
  ```

  異機種間システムでは、パススルー機能によって DDL の実行が可能です。
- ANALYZE 文の LIST CHAINED ROWS 句は、リモート表を参照できません。
- 分散データベース・システムでは、SYSDATE、USER、UID、USERENV などの環境に依存した SQL ファンクションは、その文（または文の一部）が実行される場所にかかわらず、常にローカル・サーバーについて評価されます。

注意： Oracle は、USERENV ファンクションを問合せの用途でのみサポートしています。

- リモート・オブジェクトのアクセスに関連して、次のようなパフォーマンス上の制限事項があります。
  - リモート・ビューは統計データを持ちません。
  - パーティション表に対する問合せは、最適化されない場合があります。
  - リモート表で考慮される索引の数は 20 以下です。
  - コンポジット索引で使用される列の数は 20 以下です。

    関連項目： DBMS_SQL パッケージの詳細は、『Oracle9*i* 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。

- Oracle の分散読込み一貫性の実装には制限があり、これによってあるノードが別のノードに対して古い状態になる可能性があります。問合せを実行したときに、読込み一貫性に従ってデータが取得されているにもかかわらず、そのデータが古い場合があります。この問題の管理方法の詳細は、32-26 ページの「読込み一貫性の管理」を参照してください。

分散データベースの管理 : 使用例

ここでは、データベース・リンクの管理に関する様々なタイプの文の例を示します。

- パブリック固定ユーザーのデータベース・リンクの作成
- パブリック固定ユーザーの共有データベース・リンクの作成
- パブリック接続ユーザーのデータベース・リンクの作成
- パブリック接続ユーザーの共有データベース・リンクの作成
- パブリック現行ユーザーのデータベース・リンクの作成

パブリック固定ユーザーのデータベース・リンクの作成

次の例では、ローカル・データベースに jane として接続し、データベース sales に対してパブリック固定ユーザー scott のデータベース・リンクを作成しています。データベースには、ネット・サービス名 sldb を介してアクセスします。

```
CONNECT jane/doe@local

CREATE PUBLIC DATABASE LINK sales.division3.acme.com
  CONNECT TO scott IDENTIFIED BY tiger
  USING 'sldb';
```

結果 :

ローカル・データベースに接続する任意のユーザーが、sales.division3.acme.com 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できます。各ユーザーは、リモート・データベースのスキーマ scott に接続します。

ユーザーは、次の SQL 問合せを発行して、scott のリモート・スキーマの表 emp にアクセスできます。

```
SELECT * FROM emp@sales.division3.acme.com;
```

各アプリケーションまたはユーザー・セッションは、サーバーの共通アカウントへの接続をそれぞれ別々に作成します。リモート・データベースへの接続は、アプリケーションの実行中またはユーザー・セッションの継続期間中オープンしたままです。

パブリック固定ユーザーの共有データベース・リンクの作成

次の例では、ローカル・データベースに dana として接続し、ネット・サービス名 sldb を使用して sales データベースへのパブリック・リンクを作成しています。このリンクによって、リモート・データベースに scott として接続でき、このユーザーが scott として認証されます。

```
CONNECT dana/sculley@local

CREATE SHARED PUBLIC DATABASE LINK sales.division3.acme.com
  CONNECT TO scott IDENTIFIED BY tiger
  AUTHENTICATED BY scott IDENTIFIED BY tiger
  USING 'sldb';
```

結果 :

ローカルの共有サーバーに接続する任意のユーザーがこのデータベース・リンクを使用し、共有サーバー・プロセスを介してリモートの sales データベースに接続できます。その後ユーザーは、scott スキーマの表を問い合せることができます。

上の例では、ローカルの共有サーバーはそれぞれリモート・サーバーへの接続を 1 つずつ確立できます。ローカルの共有サーバー・プロセスで sales.division3.acme.com データベース・リンクを介したリモート・サーバーへのアクセスが必要になると、そのたびにローカルの共有サーバー・プロセスが確立済みのネットワーク接続を再利用します。

パブリック接続ユーザーのデータベース・リンクの作成

次の例では、ローカル・データベースに larry として接続し、ネット・サービス名 sldb を使用してデータベースへのパブリック・リンクを作成しています。

```
CONNECT larry/oracle@local

CREATE PUBLIC DATABASE LINK redwood
  USING 'sldb';
```

結果 :

ローカル・データベースに接続する任意のユーザーが、redwood データベース・リンクを使用できます。データベース・リンクを使用するローカル・データベースの接続ユーザーによって、リモート・スキーマが決まります。

接続ユーザー scott がデータベース・リンクを使用した場合、データベース・リンクはリモート・スキーマ scott に接続します。接続ユーザー fox がデータベース・リンクを使用した場合、データベース・リンクはリモート・スキーマ fox に接続します。

リモート・スキーマ fox が emp スキーマ・オブジェクトを解決できない場合は、ローカル・データベースでローカル・ユーザー fox が次の文を実行すると失敗します。つまり、sales.division3.acme.com の fox スキーマに emp が表、ビューまたは（パブリック）シノニムとして存在しない場合は、エラーが返されます。

```
CONNECT fox/mulder@local

SELECT * FROM emp@redwood;
```

パブリック接続ユーザーの共有データベース・リンクの作成

次の例では、ローカル・データベースに neil として接続し、ネット・サービス名 sldb を使用して sales データベースへの共有パブリック・リンクを作成しています。ユーザーは、crazy/horse というユーザー ID/ パスワードによって認証されます。次の文は、パブリック接続ユーザーの共有データベース・リンクを作成します。

```
CONNECT neil/young@local

CREATE SHARED PUBLIC DATABASE LINK sales.division3.acme.com
  AUTHENTICATED BY crazy IDENTIFIED BY horse
  USING 'sldb';
```

結果 :

ローカル・サーバーに接続する各ユーザーは、この共有データベース・リンクを使用してリモート・データベースに接続し、対応するリモート・スキーマの表を問い合せることができます。

ローカルの共有サーバー・プロセスは、それぞれリモート・サーバーへの接続を 1 つずつ確立します。ローカルのサーバー・プロセスで sales.division3.acme.com データベース・リンクを介したリモート・サーバーへのアクセスが必要になると、たとえ接続ユーザーが異なるユーザーであっても、そのたびにローカルの共有サーバー・プロセスが確立済みのネットワーク接続を再利用します。

このデータベース・リンクが頻繁に使用されると、最終的にローカル・データベースのすべての共有サーバーがリモート接続を持つことになります。この時点で、新しいユーザーがこの共有データベース・リンクを使用しても、リモート・サーバーへの物理的な接続は新たに確立されません。

パブリック現行ユーザーのデータベース・リンクの作成

次の例では、ローカル・データベースに接続ユーザーとして接続し、ネット・サービス名 sldb を使用して sales データベースへのパブリック・リンクを作成しています。次の文は、パブリック現行ユーザーのデータベース・リンクを作成します。

```
CONNECT bart/simpson@local

CREATE PUBLIC DATABASE LINK sales.division3.acme.com
  CONNECT TO CURRENT_USER
  USING 'sldb';
```

注意 : このリンクを使用するには、現行ユーザーがグローバル・ユーザーであることが必要です。

結果 :

scott がリモートの emp 表から 1 行を削除するローカル・プロシージャ fire_emp を作成し、fire_emp の実行権限を ford に付与したとします。

```
CONNECT scott/tiger@local_db

CREATE PROCEDURE fire_emp (enum NUMBER)
AS
BEGIN
   DELETE FROM emp@sales.division3.acme.com
   WHERE empno=enum;
END;

GRANT EXECUTE ON fire_emp TO ford;
```

ここで、ford はローカル・データベースに接続して scott のプロシージャを実行したとします。

```
CONNECT ford/fairlane@local_db

EXECUTE PROCEDURE scott.fire_emp (enum 10345);
```

ford がプロシージャ scott.fire_emp を実行するとき、プロシージャは scott の権限のもとで実行されます。現行ユーザー・データベース・リンクが使用されているため、接続は ford のリモート・スキーマではなく、scott のリモート・スキーマに確立されます。scott はグローバル・ユーザーである必要がありますが、ford はグローバル・ユーザーでなくてもかまいません。

注意： かわりに接続ユーザー・データベース・リンクが使用された場合、接続は ford のリモート・スキーマに確立されます。実行者権限と権限の詳細は、『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。

---

scott のリモート・スキーマへの固定ユーザー・データベース・リンクを使用しても、同じ結果を得ることができます。ただし、固定ユーザー・データベース・リンクを使用した場合、データベース内にある scott のユーザー名とパスワードを読取り可能な形式で表示できるので、セキュリティが脅かされるおそれがあります。

30

分散データベース・システムのアプリケーション開発

この章では、分散データベース・システムで稼働するアプリケーションを開発する際に重要となる考慮事項について説明します。この章の内容は、次のとおりです。

- アプリケーションのデータの分散の管理
- データベース・リンクにより確立される接続の制御
- 分散システムの参照整合性の維持
- 分散問合せのチューニング
- リモート・プロシージャのエラー処理

関連項目： Oracle 環境でのアプリケーション開発の詳細は、『Oracle9*i* アプリケーション開発者ガイドー基礎編』を参照してください。

アプリケーションのデータの分散の管理

分散データベース環境では、データベース管理者（DBA）と共同でデータの最適な格納場所を決めます。その際、次の点について考慮します。

- 個々の場所から転送されるトランザクション数
- 各ノードで使用されるデータ（表の部分）の量
- パフォーマンス特性とネットワークの信頼性
- 各種ノードの速度とディスクの容量
- ノードまたはリンクが使用できないときの重要度
- 表間の参照整合性の必要性

データベース・リンクにより確立される接続の制御

SQL 文やリモート・プロシージャ・コールの中でグローバル・オブジェクト名が参照されると、ローカル・ユーザーにかわってデータベース・リンクがリモート・データベース内のセッションへの接続を確立します。リモートの接続とセッションは、それまでにローカルのユーザー・セッションに対して接続が確立されていない場合にのみ作成されます。

リモート・データベースとの間に確立された接続とセッションは、アプリケーションやユーザーによって明示的に終了されないかぎり、ローカル・ユーザーのセッションの間存続します。データベース・リンクを経由して SELECT 文を発行すると、ロールバック・セグメントにトランザクション・ロックが設定されます。セグメントを再び解放するには、COMMIT 文または ROLLBACK 文を発行する必要があります。

アプリケーションで不要になった高コストの接続を切断するには、データベース・リンクを使用して確立されたリモート接続を終了することが有効です。リモートの接続とセッションを終了するには、CLOSE DATABASE LINK 句を指定した ALTER SESSION 文を使用します。たとえば、次のトランザクションを発行した場合を考えます。

```
SELECT * FROM emp@sales;
COMMIT;
```

次の文は、sales データベース・リンクが指すリモート・データベース内のセッションを終了します。

```
ALTER SESSION CLOSE DATABASE LINK sales;
```

ユーザー・セッションのデータベース・リンク接続をクローズするには、ALTER SESSION システム権限が必要です。

注意： データベース・リンクをクローズする前に、まずそのリンクを使用しているカーソルをすべてクローズし、次にそのリンクを使用している現行トランザクションがあればそのトランザクションを終了してください。

関連項目： ALTER SESSION 文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

分散システムの参照整合性の維持

たとえば整合性制約違反など、分散型の文の一部が失敗した場合、Oracle はエラー番号 ORA-02055 を返します。以降の文またはプロシージャ・コールは、ロールバックまたはセーブポイントへのロールバックが発行されるまで、エラー番号 ORA-02067 を返します。

返されたエラー・メッセージをすべてチェックするようにアプリケーションを設計し、分散更新の一部が失敗したことを示すエラー・メッセージがないかを確認します。失敗を検出した場合は、トランザクション全体をロールバックしてからアプリケーションの処理を進めるようにしてください。

Oracle では、宣言参照整合性の制約を分散システムのノード間で定義することは許可されていません。つまり、1 つの表での宣言参照整合性の制約では、リモート表の主キーまたは一意キーを参照する外部キーを指定することはできません。ただし、トリガーを使用してノード間の親子の表関係を維持することは可能です。

トリガーを使用して分散データベースのノード間で参照整合性を定義する場合は、ネットワークの障害によって親表だけでなく子表へのアクセスも制限される可能性があります。たとえば、子表が sales データベースにあり、親表が hq データベースにあるとします。2 つのデータベース間のネットワーク接続が失敗した場合、子表に対する一部のデータ操作言語（DML）文（子表に行を挿入する文や子表内の外部キーの値を更新する文など）が実行を継続できない場合があります。これは、参照整合性トリガーが hq データベース内の親表にアクセスする必要があるためです。

関連項目： トリガーを使用した参照整合性の規定の詳細は、『Oracle9*i* アプリケーション開発者ガイドー基礎編』を参照してください。

分散問合せのチューニング

ローカルの Oracle データベース・サーバーは、分散問合せを要求と対応する数のリモート問合せに分割し、それらを実行するためにリモート・ノードに送信します。リモート・ノードは問合せを実行し、その結果をローカル・ノードに送り返します。ローカル・ノードは必要な後処理を実行し、その結果をユーザーまたはアプリケーションに返します。

問合せの処理が最適化されるようにアプリケーションを設計するには、いくつかの方法があります。この項の内容は、次のとおりです。

- 連結インライン・ビューの使用
- コストベース最適化の使用
- ヒントの使用
- 実行計画の分析

連結インライン・ビューの使用

分散問合せを最適化する最も効果的な方法は、リモート・データベースへのアクセスをできるだけ抑えて必要なデータのみを取得することです。

たとえば、分散問合せで 5 つのリモート表を 2 つの異なるリモート・データベースから参照し、複合フィルタ（`WHERE r1.salary + r2.salary > 50000` など）を使用するとします。この場合、リモート・データベースへのアクセスを 1 回にし、フィルタをリモート・サイトで適用するように問合せをリライトすることで、問合せのパフォーマンスを改善できます。このリライトにより、問合せを実行するサイトに転送されるデータの量が少なくなります。

リモート・データベースへのアクセスが 1 回になるように問合せをリライトするには、連結インライン・ビューを使用します。用語の定義は次のとおりです。

用語	定義
連結	同じデータベースにある 2 つ以上の表。

インライン・ビュー	親の SELECT 文の表に置き換えられる SELECT 文。次のかっこで囲まれた部分の埋込み SELECT 文がインライン・ビューの例です。
	`SELECT e.empno,e.ename,d.deptno,d.dname`
	`FROM (SELECT empno, ename from`
	`emp@orc1.world) e, dept d;`

連結インライン・ビュー	複数の表のデータを 1 つのデータベースのみから選択するインライン・ビュー。これにより、リモート・データベースへのアクセス回数が減り、分散問合せのパフォーマンスが向上します。

オラクル社では、連結インライン・ビューを使用して分散問合せを作成し、分散問合せのパフォーマンスを向上させることをお薦めします。Oracle のコストベース最適化を使用すると、分散問合せの多くが透過的にリライトされ、連結インライン・ビューがもたらすパフォーマンスの向上を活用できます。

コストベース最適化の使用

連結インライン・ビューによるクエリー・リライトに加えて、コストベース最適化の方法を使用すると、参照先の表から収集される統計とオプティマイザが実行する計算に従って分散問合せが最適化されます。

たとえば、コストベースのオプティマイザは、次の問合せを分析します。この例では、表統計が使用可能であることを前提としています。コストベースのオプティマイザは、CREATE TABLE 文の内部で問合せを分析します。

```
CREATE TABLE AS (
                 SELECT l.a, l.b, r1.c, r1.d, r1.e, r2.b, r2.c
                 FROM local l, remote1 r1, remote2 r2
                    WHERE l.c = r.c
                    AND r1.c = r2.c
                    AND r.e > 300
                );
```

この文は、次のようにリライトされます。

```
CREATE TABLE AS (
                 SELECT l.a, l.b, v.c, v.d, v.e
                 FROM (
                        SELECT r1.c, r1.d, r1.e, r2.b, r2.c
                         FROM remote1 r1, remote2 r2
                         WHERE r1.c = r2.c
                         AND r1.e > 300
                      ) v, local l
                 WHERE l.c = r1.c
                );
```

このリライトにより、別名 v がインライン・ビューに割り当てられます。このインライン・ビューは、上の SELECT 文内の表として参照できます。連結インライン・ビューを作成することで、リモート・サイトで実行される問合せの量が減り、それによって高コストのネットワーク通信量が減少します。

コストベース最適化の動作の仕組み

オプティマイザの主なタスクは、連結インライン・ビューを使用するように分散問合せをリライトすることです。この最適化は、次の 3 つの手順で実行されます。

1. マージ可能なビューがすべてマージされます。
2. オプティマイザが連結問合せブロックのテストを実行します。
3. オプティマイザが連結インライン・ビューを使用して問合せをリライトします。

問合せがリライトされた後、その問合せが実行され、データ・セットがユーザーに返されます。

コストベース最適化をユーザーに対して透過的に実行する場合は、複数の分散問合せのパフォーマンスを改善することはできません。特に、次のものが分散問合せに含まれている場合、コストベース最適化は効果がありません。

- 集計
- 副問合せ
- 複合 SQL

前述したものが 1 つでも分散問合せに含まれている場合は、問合せの修正方法と、分散問合せのパフォーマンスを改善するためのヒントの使用について、30-8 ページの「ヒントの使用」を参照してください。

コストベース最適化の設定

分散問合せのパフォーマンス改善のためにコストベース最適化を使用するようにシステムを設定すると、その処理がユーザーに対して透過的になります。つまり、問合せの発行時に自動的に最適化が実行されます。

Oracle のオプティマイザを利用するようにシステムを設定するには、次のタスクを完了する必要があります。

- 環境の設定
- 表の分析

環境の設定 コストベース最適化を使用可能にするには、OPTIMIZER_MODE 初期化パラメータを CHOOSE または COST に設定します。このパラメータは次の方法で設定できます。

- 初期化パラメータ・ファイルの OPTIMIZER_MODE パラメータを変更します。
- ALTER SESSION 文を発行してセッション・レベルで設定します。

OPTIMIZER_MODE 初期化パラメータをセッション・レベルで設定するには、次のどちらかの文を発行します。

```
ALTER SESSION OPTIMIZER_MODE = CHOOSE;
ALTER SESSION OPTIMIZER_MODE = COST;
```

関連項目： パラメータ・ファイルの OPTIMIZER_MODE 初期化パラメータの設定と、コストベース最適化方法を使用するためのシステムの構成の詳細は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。

表の分析 コストベース最適化で分散問合せのための最も効率的なパスが選択されるようにするには、問合せに関係する表の正確な統計を提供する必要があります。そのためには、DBMS_STATS パッケージまたは ANALYZE 文を使用します。

注意： DBMS_STATS プロシージャまたは ANALYZE 文を実行するには、表に対してローカルに接続する必要があります。たとえば、次の文は実行できません。

```
ANALYZE TABLE remote@remote.com COMPUTE STATISTICS;
```

この ANALYZE 文または等価の DBMS_STATS プロシージャを実行するには、先にリモート・サイトへの接続が必要になります。

次の DBMS_STATS プロシージャを使用すると、特定のクラスのオプティマイザ統計を収集できます。

- GATHER_INDEX_STATS
- GATHER_TABLE_STATS
- GATHER_SCHEMA_STATS
- GATHER_DATABASE_STATS

たとえば、分散トランザクションが日常的に scott.dept 表にアクセスするとします。コストベースのオプティマイザが引き続き最適な方法を確実に選択するように、次の文を実行します。

```
BEGIN
   DBMS_STATS.GATHER_TABLE_STATS ('scott', 'dept');
END;
```

関連項目：

- 統計収集の詳細は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。
- DBMS_STATS パッケージの使用の詳細は、『Oracle9*i* 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。

ヒントの使用

文が十分に最適化されない場合は、ヒントを使用してコストベース最適化の機能を拡張できます。特に、独自の問合せを記述して連結インライン・ビューを利用する場合は、分散問合せがリライトされないようにコストベース・オプティマイザに指示を与えます。

また、データベース環境に関する特別な情報（統計、負荷、ネットワークおよび CPU の制限事項、分散問合せなど）を持っている場合は、ヒントを指定してコストベース最適化を適切に誘導できます。たとえば、データベース環境の情報に基づく連結インライン・ビューを使用して、独自に最適化した問合せを記述した場合は、NO_MERGE ヒントを指定することにより、オプティマイザが問合せをリライトしないようにできます。

この手法は、分散問合せに集計、副問合せまたは複合 SQL が含まれている場合に特に役立ちます。このタイプの分散問合せはオプティマイザによってリライトできないので、NO_MERGE を指定して、オプティマイザが 30-6 ページの「コストベース最適化の動作の仕組み」で説明されている手順を省略するように指示します。

DRIVING_SITE ヒントを使用すると、リモート・サイトを問合せ実行サイトとして機能するように定義できます。この方法では、問合せがリモート・サイトで実行され、データがローカル・サイトに返されます。リモート・サイトにデータの大部分が格納されているときは、このヒントが特に役立ちます。

> **関連項目：** ヒントの使用の詳細は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。

NO_MERGE ヒントの使用

NO_MERGE ヒントは、連結されない可能性のある SQL 文にインライン・ビューがマージされるのを防ぎます（30-8 ページの「ヒントの使用」を参照）。このヒントは、SELECT 文に埋め込みます。インライン・ビューを使用する SELECT 文の先頭に引数として指定するか、またはインライン・ビューを定義する問合せブロック内に指定します。

```
/* with argument */

SELECT /*+NO_MERGE(v)*/ t1.x, v.avg_y
   FROM t1, (SELECT x, AVG(y) AS avg_y FROM t2 GROUP BY x) v,
   WHERE t1.x = v.x AND t1.y = 1;

/* in query block */

SELECT t1.x, v.avg_y
   FROM t1, (SELECT /*+NO_MERGE*/ x, AVG(y) AS avg_y FROM t2 GROUP BY x) v,
   WHERE t1.x = v.x AND t1.y = 1;
```

通常、このヒントは、データベース環境の情報に基づいて最適化した問合せを作成したときに使用します。

DRIVING_SITE ヒントの使用

DRIVING_SITE ヒントを使用すると、問合せを実行するサイトを指定できます。問合せを実行するサイトはコストベース最適化によって決定されるのが最適ですが、オプティマイザの判断を変更する方がよい場合は、実行サイトを手動で指定できます。

DRIVING_SITE ヒントを使用した SELECT 文の例を次に示します。

```
SELECT /*+DRIVING_SITE(dept)*/ * FROM emp, dept@remote.com
   WHERE emp.deptno = dept.deptno;
```

実行計画の分析

分散問合せのチューニングの際に重要なこととして、実行計画の分析があります。分析結果から得られるフィードバックは、データベースのテストと検証を行う上で重要な要素になります。計画を比較するときは、検証が特に重要になります。たとえば、コストベースのオプティマイザによって最適化された分散問合せの計画と、ヒントや連結インライン・ビューなどの手法を駆使して手動で最適化した問合せの計画を比較するときなどに重要です。

> **関連項目：** 実行計画、EXPLAIN PLAN 文およびその結果の解釈方法の詳細は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。

計画を格納するためのデータベースの準備

分散問合せの実行計画を表示できるようにするには、まずデータベース内に実行計画を格納する場所を準備します。そのために、スクリプトを実行します。次のスクリプトを実行し、データベース内に実行計画を格納する場所を準備します。

```
SQL> @UTLXPLAN.SQL
```

> **注意：** utlxplan.sql ファイルは、$ORACLE_HOME/rdbms/admin ディレクトリにあります。

utlxplan.sql を実行すると、現行スキーマ内に、実行計画を一時的に格納する PLAN_TABLE という表が作成されます。

実行計画の生成

データベース内に実行計画を格納する場所を準備すると、指定した問合せの計画を表示するための準備が完了します。SQL 文を直接実行するかわりに、EXPLAIN PLAN FOR 句に文を追加します。たとえば、次のような文を実行できます。

```
EXPLAIN PLAN FOR
   SELECT d.dname
   FROM dept d
      WHERE d.deptno
      IN (SELECT deptno
          FROM emp@orc2.world
          GROUP BY deptno
          HAVING COUNT (deptno) >3
          )
/
```

実行計画の表示

前述の SQL 文を実行すると、すでに作成した PLAN_TABLE に実行計画が一時的に格納されます。実行計画の結果を表示するには、次のスクリプトを実行します。

```
SQL> @UTLXPLS.SQL
```

> **注意：** utlxpls.sql ファイルは $ORACLE_HOME/rdbms/admin ディレクトリにあります。

utlxpls.sql スクリプトを実行すると、指定した SELECT 文の実行計画が表示されます。結果は、次のように書式化されます。

```
Plan Table
--------------------------------------------------------------------------------
| Operation                 |  Name    |  Rows | Bytes|  Cost  | Pstart| Pstop |
--------------------------------------------------------------------------------
| SELECT STATEMENT          |          |       |      |        |       |       |
|  NESTED LOOPS             |          |       |      |        |       |       |
|   VIEW                    |          |       |      |        |       |       |
|    REMOTE                 |          |       |      |        |       |       |
|   TABLE ACCESS BY INDEX RO|DEPT      |       |      |        |       |       |
|    INDEX UNIQUE SCAN      |PK_DEPT   |       |      |        |       |       |
--------------------------------------------------------------------------------
```

独自の連結インライン・ビューを記述するか、またはヒントを使用して、分散問合せを手動で最適化しようとする場合は、手動最適化の前および後に実行計画を生成するのが最適です。これら両方の実行計画を使用することで、手動最適化の効果を比較し、必要に応じて変更を加え、分散問合せのパフォーマンスを改善できます。

リモート・サイトで実行される SQL 文を表示するには、次の SELECT 文を実行します。

```
SELECT OTHER
FROM PLAN_TABLE
   WHERE operation = 'REMOTE';
```

出力例を次に示します。

```
SELECT DISTINCT "A1"."DEPTNO" FROM "EMP" "A1"
   GROUP BY "A1"."DEPTNO" HAVING COUNT("A1"."DEPTNO")>3
```

注意： OTHER 列の内容全体がうまく表示されない場合は、次の SQL*Plus コマンドを実行してください。

```
SET LONG 9999999
```

リモート・プロシージャのエラー処理

Oracle がプロシージャをローカルまたはリモートの位置で実行するときには、次の 4 種類の例外が発生する可能性があります。

- PL/SQL のユーザー定義例外。この例外は、EXCEPTION キーワードを使用して宣言する必要があります。
- PL/SQL の事前定義例外。NO_DATA_FOUND キーワードなど。
- SQL エラー。ORA-00900 や ORA-02015 など。
- RAISE_APPLICATION_ERROR() プロシージャを使用して生成されるアプリケーション例外。

ローカル・プロシージャを使用するときは、次のような例外ハンドラを記述して、これらのメッセージをトラップできます。

```
BEGIN
  ...
EXCEPTION
   WHEN ZERO_DIVIDE THEN
   /* ... handle the exception */
END;
```

WHEN 句には例外名が必要です。RAISE_APPLICATION_ERROR で生成された例外など、例外が名前を持っていない場合は、PRAGMA_EXCEPTION_INIT を使用して名前を割り当てることができます。次に例を示します。

```
DECLARE
  null_salary EXCEPTION;
  PRAGMA EXCEPTION_INIT(null_salary, -20101);
BEGIN
  ...
  RAISE_APPLICATION_ERROR(-20101, 'salary is missing');
...
EXCEPTION
  WHEN null_salary THEN
  ...
END;
```

リモート・プロシージャをコールするときは、例外をローカル・プロシージャの例外ハンドラで処理できます。リモート・プロシージャは、エラー番号をローカルのコール側プロシージャに返す必要があります。ローカルのコール側プロシージャは、受け取った例外を前の例で示したように処理します。PL/SQL のユーザー定義例外は、常に ORA-06510 をローカル・プロシージャに返します。

したがって、2 つの異なるユーザー定義例外をエラー番号で区別することはできません。その他のリモート例外はすべて、ローカル例外と同じ方法で処理できます。

> **関連項目：** PL/SQL プロシージャの詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。

31

分散トランザクションの概念

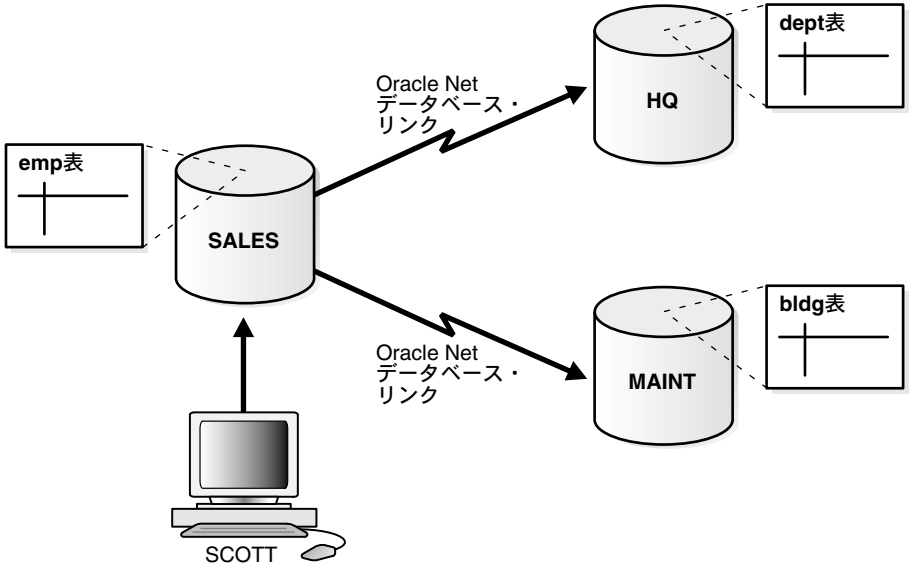
分散トランザクションの概要とその整合性を維持する Oracle の仕組みについて説明します。この章の内容は、次のとおりです。

- 分散トランザクションの概要
- 分散トランザクションのセッション・ツリー
- 2 フェーズ・コミット・メカニズム
- インダウト・トランザクション
- 分散トランザクション処理 : 事例

分散トランザクションの概要

分散トランザクションは 1 つ以上の文からなり、それらが個別に、またはグループとして、分散データベースの複数ノードのデータを更新します。たとえば、図 31-1 に示すデータベース構成を考えます。

図 31-1　分散システム



scott によって実行される次の分散トランザクションは、ローカルの sales データベース、リモートの hq データベース、およびリモートの maint データベースを更新します。

```
UPDATE scott.dept@hq.us.acme.com
  SET loc = 'REDWOOD SHORES'
  WHERE deptno = 10;
UPDATE scott.emp
  SET deptno = 11
  WHERE deptno = 10;
UPDATE scott.bldg@maint.us.acme.com
  SET room = 1225
  WHERE room = 1163;
COMMIT;
```

注意： トランザクションのすべての文が 1 つのリモート・ノードのみを参照している場合、そのトランザクションは分散トランザクションではなくリモート・トランザクションです。

分散トランザクションでは、次の 2 種類の操作が許可されます。

- DML および DDL トランザクション
- トランザクション制御文

DML および DDL トランザクション

分散トランザクションでサポートされているデータ操作言語（DML）およびデータ定義言語（DDL）操作は、次のとおりです。

- `CREATE TABLE AS SELECT`
- `DELETE`
- `INSERT`（デフォルトおよびダイレクト・ロード）
- `LOCK TABLE`
- `SELECT`
- `SELECT FOR UPDATE`

DML 文および DDL 文はパラレルに実行でき、ダイレクト・ロード・インサート文はシリアルに実行できます。ただし、次の制限に注意してください。

- リモート操作はすべて `SELECT` 文である必要があります。
- これらの文は、別の分散トランザクション内の句であってはいけません。
- `INSERT`、`UPDATE` または `DELETE` 文の *table_expression_clause* で参照される表がリモートの場合、実行はパラレルではなくシリアルになります。
- パラレル DML/DDL またはダイレクト・ロード・インサートの発行後にリモート操作を実行することはできません。
- XA または OCI を使用してトランザクションを開始した場合、そのトランザクションはシリアルに実行されます。
- パラレル操作の実行元であるトランザクションで、ループバック操作を実行することはできません。たとえば、実際はローカル・オブジェクトのシノニムであるリモート・オブジェクトを参照することはできません。
- トランザクションで `SELECT` 以外の分散操作を実行する場合、DML はパラレル化されません。

トランザクション制御文

サポートされているトランザクション制御文は、次のとおりです。

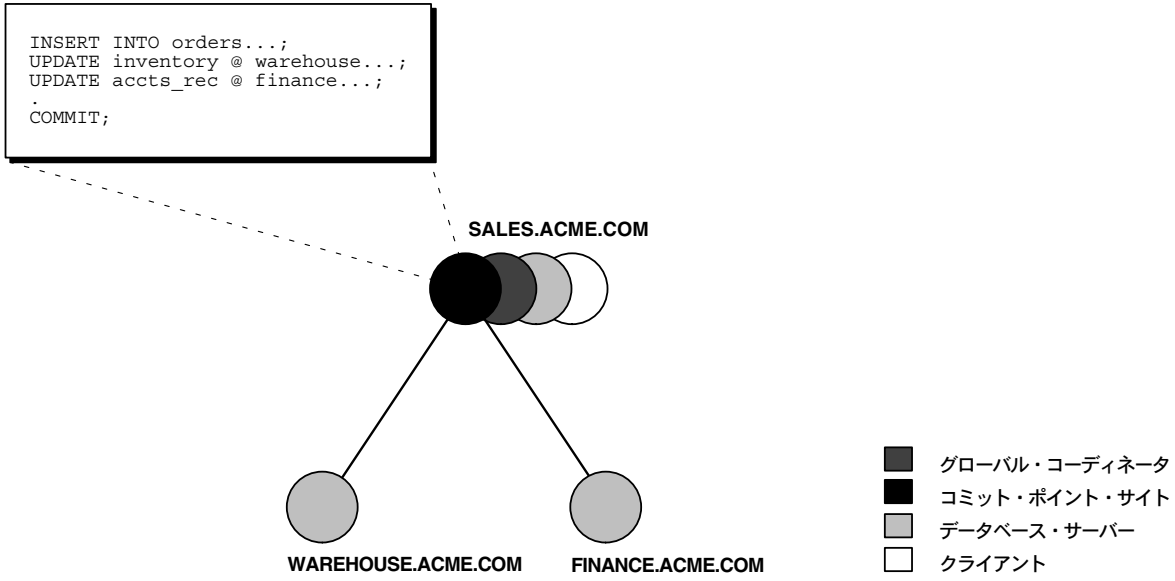
- COMMIT
- ROLLBACK
- SAVEPOINT

関連項目： これらの SQL 文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

分散トランザクションのセッション・ツリー

分散トランザクションで文が発行されると、Oracle はトランザクションに参加しているすべてのノードの**セッション・ツリー**を定義します。セッション・ツリーとは、セッション間の関係とセッションのロールを表す階層モデルです。セッション・ツリーの例を図 31-2 に示します。

図 31-2　セッション・ツリーの例



分散トランザクションのセッション・ツリーに参加しているすべてのノードは、次に示すロールを 1 つ以上持ちます。

ロール	説明
クライアント	異なるノードに属するデータベース内の情報を参照するノード。
データベース・サーバー	別のノードからの情報の要求を受け取るノード。
グローバル・コーディネータ	分散トランザクションの実行元ノード。
ローカル・コーディネータ	他のノードのデータを強制的に参照して、自身のトランザクション部分を完了するノード。
コミット・ポイント・サイト	グローバル・コーディネータの指示に従ってトランザクションをコミットまたはロールバックするノード。

分散トランザクションのノードが果たすロールは、次の条件によって決まります。

- トランザクションがローカルとリモートのどちらであるか。
- ノードの**コミット・ポイント強度**（31-7 ページの「コミット・ポイント・サイト」を参照）。
- 要求されたすべてのデータがノードで使用可能か、またはトランザクションを完了するために他のノードを参照する必要があるか。
- ノードが読取り専用かどうか。

クライアント

情報を別のノードのデータベースから参照するとき、ノードはクライアントとして機能します。参照先のノードはデータベース・サーバーです。図 31-2 のノード `sales` は、`warehouse` データベースおよび `finance` データベースが稼働しているノードのクライアントです。

データベース・サーバー

データベース・サーバーは、クライアントがデータを要求する要求先データベースが稼働しているノードです。

図 31-2 では、`sales` ノードのアプリケーションは、`warehouse` ノードおよび `finance` ノードのデータにアクセスする分散トランザクションを開始します。したがって、`sales.acme.com` はクライアント・ノードのロールを持ち、`warehouse` および `finance` はどちらもデータベース・サーバーのロールを持ちます。この例では、`sales` はデータベース・サーバーとクライアントを兼務しています。これは、アプリケーションが `sales` データベースのデータも変更するためです。

ローカル・コーディネータ

自身の分散トランザクション部分を完了するために他のノードのデータを参照する必要があるノードのことを、ローカル・コーディネータと呼びます。図 31-2 では、sales は自身が直接参照している warehouse ノードおよび finance ノードを調整するので、ローカル・コーディネータになります。ノード sales はまた、トランザクションに関係するすべてのノードを調整することから、グローバル・コーディネータにもなっています。

ローカル・コーディネータは、自身が直接やり取りするノードの間で次のようにトランザクションを調整する役目を果たします。

- これらのノード間でトランザクションのステータス情報を受け渡します。
- これらのノードに問合せを渡します。
- これらのノードから問合せを受け取り、他のノードに渡します。
- 問合せの結果を開始元のノードに返します。

グローバル・コーディネータ

分散トランザクションの実行元であるノードのことを、グローバル・コーディネータと呼びます。分散トランザクションを発行するデータベース・アプリケーションは、グローバル・コーディネータとして機能しているノードに直接接続します。たとえば、図 31-2 では、ノード sales で発行されたトランザクションは、データベース・サーバー warehouse および finance の情報を参照します。したがって、sales.acme.com は、この分散トランザクションのグローバル・コーディネータになります。

グローバル・コーディネータは、セッション・ツリーの親またはルートになります。グローバル・コーディネータは、分散トランザクション処理中に次の操作を実行します。

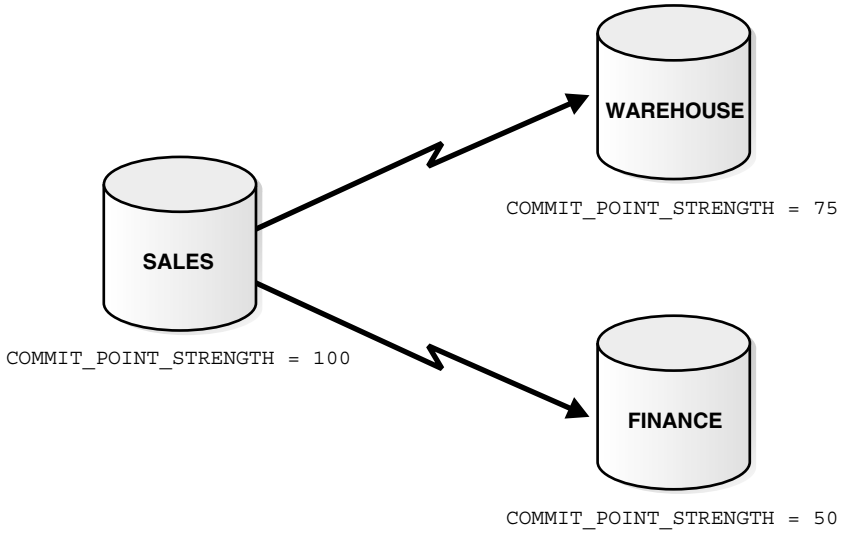
- 分散トランザクションのすべての SQL 文やリモート・プロシージャ・コールなどを参照先のノードに直接送り、それによってセッション・ツリーを構成します。
- コミット・ポイント・サイト以外のすべての直接参照先ノードに対して、トランザクションの準備をするように指示します。
- すべてのノードの準備が正常に完了した場合に、トランザクションのグローバル・コミットを開始するようにコミット・ポイント・サイトに対して指示します。
- ノードから異常終了の応答があった場合に、トランザクションのグローバル・ロールバックを開始するようにすべてのノードに対して指示します。

コミット・ポイント・サイト

コミット・ポイント・サイトの役割は、グローバル・コーディネータの指示に従ってコミットまたはロールバックの操作を開始することです。システム管理者は、すべてのノードにコミット・ポイント強度を割り当てることで、セッション・ツリー内のノードの 1 つをコミット・ポイント・サイトとして必ず指定します。コミット・ポイント・サイトには、最も重要なデータを格納するノードを選択してください。

図 31-3 は、sales がコミット・ポイント・サイトとして機能している分散システムの例です。

図 31-3 コミット・ポイント・サイト



コミット・ポイント・サイトは、分散トランザクションに関係する他のすべてのノードと次の点で区別されます。

- コミット・ポイント・サイトが準備完了状態に入ることはありません。そのため、コミット・ポイント・サイトに最も重要なデータが格納されている場合は、たとえ障害が起きたとしても、データがインダウトのままになることはありません。障害が発生すると、障害を起こしたノードは準備完了状態のままになり、インダウト・トランザクションが解決されるまで必要なロックが保持されます。

- コミット・ポイント・サイトは、トランザクションに関係している他のノードよりも先にコミットします。実際は、コミット・ポイント・サイトでの分散トランザクションの結果によって、すべてのノードでのトランザクションがコミットされるかロールバックされるかが決まり、他のノードはコミット・ポイント・サイトの指示に従います。グローバル・コーディネータは、すべてのノードでコミット・ポイント・サイトと同様にトランザクションが完了することを保証します。

分散トランザクションのコミットの仕組み

分散トランザクションは、コミット・ポイント以外のすべてのサイトで準備が完了した後、コミットされたとみなされますが、実際には、トランザクションはコミット・ポイント・サイトで先にコミットされています。コミット・ポイント・サイトのオンライン REDO ログは、このノードで分散トランザクションがコミットされるとただちに更新されます。

コミット・ポイント・ログにはコミットの記録があります。そのため、たとえ参加中のノードの一部がまだ準備完了状態であり、それらのノードで実際にトランザクションがコミットされていない場合であっても、トランザクションはコミットされたとみなされます。同様の意味で、コミット・ポイント・サイトでコミットがまだログに記録されていない場合には、分散トランザクションはコミットされていないとみなされます。

コミット・ポイント強度

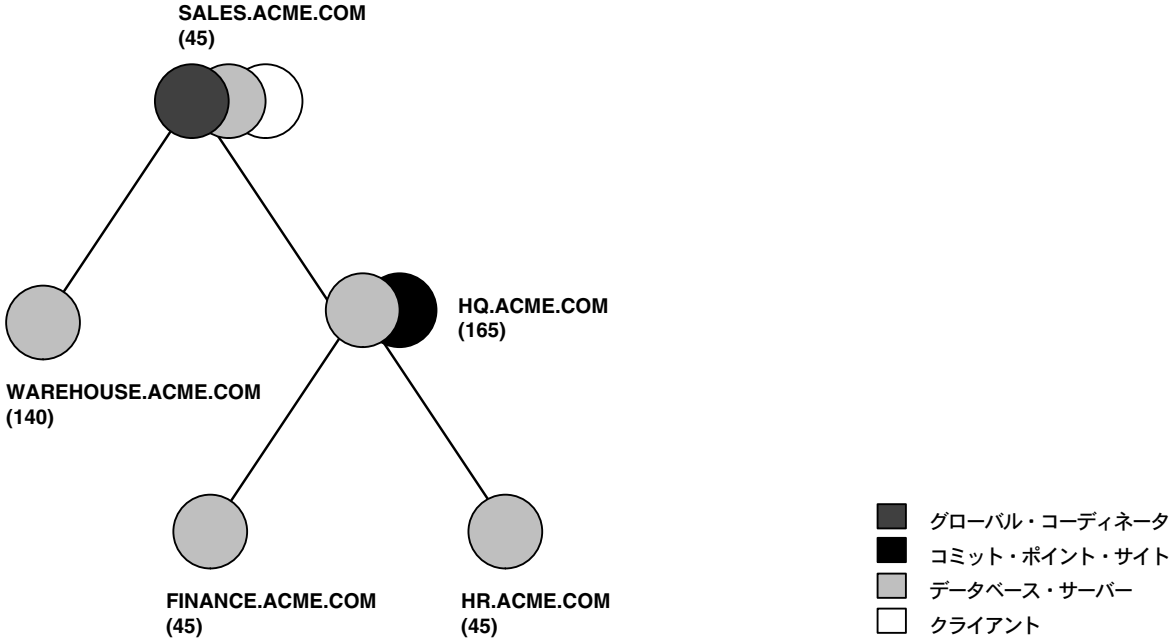
データベース・サーバーには、必ずコミット・ポイント強度を割り当てる必要があります。データベース・サーバーが分散トランザクション内で参照される場合、そのコミット・ポイント強度の値によって、2 フェーズ・コミットにおける役割が決まります。具体的には、このコミット・ポイント強度によって、どのノードが分散トランザクションのコミット・ポイント・サイトになり、他のすべてのノードより前にコミットするかが決まります。この値を指定するには、初期化パラメータ `COMMIT_POINT_STRENGTH` を使用します。ここでは、Oracle がコミット・ポイント・サイトを決定する仕組みについて説明します。

準備フェーズの冒頭で決定されるコミット・ポイント・サイトは、トランザクションに参加しているノードの中からのみ選択されます。次に示す一連のイベントが発生します。

1. Oracle は、グローバル・コーディネータが直接参照しているノードの中で、最も高いコミット・ポイント強度を持つノードをコミット・ポイント・サイトとして選択します。
2. 最初に選択されたノードは、このトランザクションの情報を取得する必要のあるノードの中で、自身よりも高いコミット・ポイント強度を持っているノードがないかを判断します。
3. トランザクションで直接参照しているノードの中で最も高いコミット・ポイント強度を持つノードか、またはそのノードのサーバーの中でより高いコミット・ポイント強度を持つもののどちらかが、コミット・ポイント・サイトになります。
4. 最終的なコミット・ポイント・サイトが決定した後、グローバル・コーディネータは、トランザクションに参加しているすべてのノードに準備応答を送ります。

図 31-4 は、各ノードのコミット・ポイント強度（かっこ内の値）を示したセッション・ツリーの例です。また、コミット・ポイント・サイトとして選択されたノードも示しています。

図 31-4　コミット・ポイント強度とコミット・ポイント・サイトの決定



コミット・ポイント・サイトを決定するときは、次の条件が適用されます。

- 読取り専用ノードは、コミット・ポイント・サイトにはなれません。
- グローバル・コーディネータが直接参照している複数のノードが同じコミット・ポイント強度を持っている場合、Oracle はそれらのうちの 1 つをコミット・ポイント・サイトとして指定します。
- 分散トランザクションがロールバックで終了する場合は、準備フェーズとコミット・フェーズは不要です。そのため、コミット・ポイント・サイトは決定されません。そのかわりに、グローバル・コーディネータは ROLLBACK 文をすべてのノードに送り、分散トランザクションの処理を終了します。

図 31-4 のように、コミット・ポイント・サイトとグローバル・コーディネータがセッション・ツリーの異なるノードになることもあります。各ノードのコミット・ポイント強度は、最初に接続が確立されたときにコーディネータに渡されます。コーディネータは、2 フェーズ・コミット時のコミット・ポイント・サイトを効率的に選択するために、自身が直接やり取りしている各ノードのコミット・ポイント強度を保持しています。そのため、コミットが発生するたびにコーディネータとノード間でコミット・ポイント強度を交換する必要がありません。

関連項目：

- ノードのコミット・ポイント強度の設定方法の詳細は、32-2 ページの「ノードのコミット・ポイント強度の指定」を参照してください。
- 初期化パラメータ COMMIT_POINT_STRENGTH の詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

2 フェーズ・コミット・メカニズム

ローカル・データベースのトランザクションとは異なり、分散トランザクションには複数のデータベースでのデータの変更が伴います。そのため、Oracle はトランザクションの変更のコミットまたはロールバックを自己完結単位として調整する必要があり、分散トランザクションの処理はより複雑になります。言い換えれば、トランザクション全体がコミットするか、またはトランザクション全体がロールバックするかのどちらかの結果になります。

Oracle は、**2 フェーズ・コミット・メカニズム**を使用することで、分散トランザクションにおけるデータの整合性を保証します。**準備フェーズ**では、トランザクション内の開始ノードが他の参加ノードに対して、トランザクションをコミットまたはロールバックすることを確約するように要求します。**コミット・フェーズ**では、開始ノードがすべての参加ノードに対して、トランザクションをコミットするように要求します。この結果が達成できない場合、すべてのノードはロールバックするように要求されます。

分散トランザクションに参加しているすべてのノードは、必ず同じ動作を実行します。つまり、ノードすべてがトランザクションをコミットするか、またはノードすべてがトランザクションをロールバックします。Oracle は、分散トランザクションのコミットまたはロールバックを自動的に制御および監視しており、2 フェーズ・コミット・メカニズムを使用して**グローバル・データベース**（トランザクションに参加しているデータベースの集まり）の整合性を維持します。このメカニズムは完全に透過的であり、ユーザーやアプリケーション開発者の側でプログラミングを行う必要はまったくありません。

コミット・メカニズムは、次の各フェーズからなります。ユーザーが分散トランザクションをコミットすると、必ずこれらのフェーズが自動的に実行されます。

フェーズ	説明
準備フェーズ	**グローバル・コーディネータ**と呼ばれる開始ノードは、コミット・ポイント・サイト以外の参加ノードに対して、たとえ障害が起きた場合でもトランザクションをコミットまたはロールバックすることを確約するように要求します。準備ができないノードがある場合、トランザクションはロールバックされます。
コミット・フェーズ	すべての参加ノードが準備完了の応答をコーディネータに伝えると、コーディネータは、コミット・ポイント・サイトにコミットを要求します。コミット・ポイント・サイトのコミット後、コーディネータは、トランザクションをコミットするように他のすべてのノードに要求します。
情報消去フェーズ	グローバル・コーディネータは、トランザクションに関する情報を消去します。

この項の内容は、次のとおりです。

- 準備フェーズ
- コミット・フェーズ
- 情報消去フェーズ

準備フェーズ

準備フェーズは、分散トランザクションのコミットにおける 1 番目のフェーズです。このフェーズでは、Oracle がトランザクションを実際にコミットまたはロールバックすることはありません。ここでは、分散トランザクションで参照されているすべてのノード（31-7 ページの「コミット・ポイント・サイト」で説明されているコミット・ポイント・サイトを除く）がコミットを準備するように指示されます。ノードは、準備を完了するために次の処理を実行します。

- オンライン REDO ログの情報を記録し、それ以降障害が発生してもトランザクションをコミットまたはロールバックできるようにします。
- 変更済みの表に分散ロックを設定し、読取りを防ぎます。

各ノードは、コミットの準備が完了したという応答をグローバル・コーディネータに伝えることにより、その後トランザクションをコミットまたはロールバックすることを確約します。しかし、トランザクションをコミットまたはロールバックするという決定を各ノードが一方的に下すわけではありません。この確約の意味は、この時点でインスタンス障害が発生した場合、ノードはオンライン・ログの REDO レコードを使用してデータベースをリカバリし、準備フェーズに戻すことができるということです。

注意： ノードの準備完了後に発行した問合せは、すべてのフェーズが完了するまで、関連するロック済みデータにアクセスできません。この時間は、障害が発生しないかぎり問題にはなりません（32-8 ページの「インダウト・トランザクションの処理方法の決定」を参照）。

準備フェーズでの応答のタイプ

準備を指示されたノードは、次の方法で応答できます。

応答	意味
準備完了	ノードのデータの変更が分散トランザクション内の文によって完了しており、ノードの準備が正常に完了しています。
読取り専用	ノードで変更されるデータがない（問合せのみ）か、または変更できないので、準備は不要です。
異常終了	ノードが正常に準備できません。

準備応答 ノードの準備が正常に完了すると、ノードは**準備完了メッセージ**を発行します。このメッセージは、ノードの変更レコードがオンライン・ログに格納されており、ノードでコミットまたはロールバックのどちらかを実行できる準備が整っていることを示します。また、このメッセージによって、トランザクションに対して保持されているロックが障害発生時にも残ることが保証されます。

読取り専用応答 ノードが準備を要求されたときに、データベースにアクセスする SQL 文がノードのデータを変更しない場合、ノードは**読取り専用メッセージ**で応答します。このメッセージは、ノードがコミット・フェーズに参加しないことを示します。

分散トランザクションの全部または一部が読取り専用になるケースとして、次の 3 つがあります。

ケース	条件	結果
一部読取り専用	次のいずれかが発生した場合 ■ 1 つ以上のノードで問合せのみが発行された。 ■ データが変更されない。 ■ トリガーの起動または制約違反のために変更がロールバックされた。	読取り専用ノードは、準備を要求されたときに自身のステータスを認識します。読取り専用ノードは、読取り専用応答をローカル・コーディネータに伝えます。この結果、Oracle は読取り専用ノードを以降の処理から除外するので、コミット・フェーズがより高速に完了します。
準備フェーズでの完全な読取り専用	次のすべてが発生した場合 ■ データが変更されない。 ■ トランザクションが SET TRANSACTION READ ONLY 文で始まっていない。	準備フェーズ時にすべてのノードが読取り専用であることを認識するので、コミット・フェーズは不要になります。グローバル・コーディネータにはすべてのノードが読取り専用かどうかわからないため、グローバル・コーディネータは引き続き準備フェーズを実行する必要があります。
2 フェーズ・コミットなしの完全な読取り専用	次のすべてが発生した場合 ■ データが変更されない。 ■ トランザクションが SET TRANSACTION READ ONLY 文で始まっている。	このトランザクションでは問合せのみが許可されるため、グローバル・コーディネータは、2 フェーズ・コミットを実行する必要がありません。また、システム変更番号（SCN）の調整がノード間でグローバルに行われるため、他のトランザクションによる変更によってグローバル・トランザクション・レベルの読込み一貫性が損なわれることはありません。このトランザクションでは、ロールバック・セグメントは使用されません。

分散トランザクションが読取り専用に設定されている場合、そのトランザクションでロールバック・セグメントは使用されません。多数のユーザーがデータベースに接続していて、ユーザーのトランザクションが READ ONLY に設定されていない場合は、それらのトランザクションが問合せを実行するのみであっても、ロールバック領域が割り当てられます。

異常終了時のエラー ノードは、正常に準備できないときに次の処理を実行します。

1. トランザクションが現在保持しているリソースを解放し、トランザクションのローカル部分をロールバックします。
2. 分散トランザクション内で自身を参照しているノードに対し、異常終了メッセージで応答します。

これらの処理は、分散トランザクションに関係している他のノードに伝播します。これにより、他のノードはトランザクションをロールバックできるので、グローバル・データベースのデータの整合性が保証されます。この応答によって、「トランザクションに関係しているすべてのノードが同じ論理時間ですべてコミットするかまたはすべてロールバックする」という分散トランザクションの基本原則が守られます。

準備フェーズの手順

準備フェーズを完了するために、コミット・ポイント・サイトを除く各ノードは次の手順を実行します。

1. ノードは、自身の**子**（以降参照する各ノード）に対して、コミットの準備をするように要求します。
2. ノードは、トランザクションによって自分自身のデータまたは子のデータが変更されるかどうかをチェックします。データが変更されない場合、ノードは残りの手順を省略し、読取り専用応答を返します（31-12 ページの「読取り専用応答」を参照）。
3. データが変更される場合、ノードはトランザクションのコミットに必要なリソースを割り当てます。
4. ノードは、トランザクションによる変更に対応する REDO レコードをオンライン REDO ログに保存します。
5. ノードは、トランザクションに対して保持されているロックが障害発生時にも残ることを保証します。
6. ノードは、準備応答を開始ノードに伝えます（31-12 ページの「準備応答」を参照）。あるいは、自身またはその子のいずれかが準備の試行に失敗した場合は、異常終了応答を伝えます（31-14 ページの「異常終了時のエラー」を参照）。

これらの処理によって、ノードが後で自身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ることが保証されます。この後、準備完了ノードは、グローバル・コーディネータから COMMIT または ROLLBACK 要求を受け取るまで待機します。

各ノードの準備が完了した後、分散トランザクションは**インダウト**と呼ばれる状態になります（31-16 ページの「インダウト・トランザクション」を参照）。分散トランザクションは、すべての変更がコミットまたはロールバックされるまで、インダウト状態のままです。

コミット・フェーズ

コミット・フェーズは、分散トランザクションのコミットにおける 2 番目のフェーズです。このフェーズになる前に、分散トランザクションで参照されるコミット・ポイント・サイト以外のすべてのノードが準備を完了していること、つまり、コミット・ポイント・サイト以外のすべてのノードがトランザクションのコミットに必要なリソースを確保していることが保証されます。

コミット・フェーズの手順

コミット・フェーズは次の手順で構成されています。

1. グローバル・コーディネータは、コミット・ポイント・サイトにコミットを指示します。
2. コミット・ポイント・サイトは、コミットを実行します。
3. コミット・ポイント・サイトは、コミットが完了したことをグローバル・コーディネータに伝えます。
4. グローバル・コーディネータおよびローカル・コーディネータは、すべてのノードに対してトランザクションをコミットするように指示するメッセージを送ります。
5. 各ノードで分散トランザクションのローカル部分がコミットされて、ロックが解放されます。
6. 各ノードで、トランザクションのコミットが完了したことを示す追加の REDO エントリがローカル REDO ログに記録されます。
7. 各参加ノードは、自身のコミットが完了したことをグローバル・コーディネータに伝えます。

コミット・フェーズが完了するときは、分散システムの全ノードのデータについて一貫性が保たれています。

グローバル・データベースの一貫性の保証

コミットされた各トランザクションには、そのトランザクション内部の SQL 文によって行われた変更を一意に識別するための SCN が対応付けられます。SCN は、データベースのコミット済みバージョンを一意に識別する Oracle 内部のタイムスタンプの役割を持ちます。

分散システムでは、次の処理がすべて発生したときに、通信中のノードの SCN が調整されます。

- 1 つ以上のデータベース・リンクによって表されるパスを使用した接続の確立
- 分散 SQL 文の実行
- 分散トランザクションのコミット

特に、分散システムのノード間で SCN が調整されることにより、文とトランザクションの両方のレベルでグローバルな読込み一貫性が保証されるという利点があります。必要であれば、グローバルな時間ベースのリカバリを実行することもできます。

準備フェーズ時に、Oracle は、トランザクションに関係しているすべてのノードで最も高い SCN を判断します。次に、コミット・ポイント・サイトにおいて最も高い SCN でトランザクションをコミットします。さらに、すべての準備完了ノードに対して、コミットの指示とともにコミット SCN を送ります。

関連項目： 読込み一貫性におけるタイム・ラグ問題の管理の詳細は、32-26 ページの「読込み一貫性の管理」を参照してください。

情報消去フェーズ

参加ノードが自身のコミットの完了をコミット・ポイント・サイトに通知した後、コミット・ポイント・サイトは、トランザクションに関する情報を消去できます。次の手順が発生します。

1. すべてのノードのコミットが完了したことをグローバル・コーディネータから通知された後、コミット・ポイント・サイトは、このトランザクションに関するステータス情報を消去します。
2. コミット・ポイント・サイトは、ステータス情報を消去したことをグローバル・コーディネータに伝えます。
3. グローバル・コーディネータは、トランザクションに関する自分自身の情報を消去します。

インダウト・トランザクション

2 フェーズ・コミット・メカニズムは、すべてのノードがコミットされるかまたはロールバックを一斉に実行することを保証します。システムやネットワークのエラーのために、3 つのフェーズのいずれかが失敗した場合はどうなるのでしょうか。この場合、トランザクションはインダウトになります。

分散トランザクションは、次の要因によってインダウトになる可能性があります。

- Oracle ソフトウェアを実行しているサーバー・マシンがクラッシュした。
- 分散処理に関係している複数の Oracle データベース間のネットワーク接続が切断された。
- 未処理のソフトウェア・エラーが発生した。

マシン、ネットワークまたはソフトウェアの問題が解決されると、RECO プロセスによってインダウト・トランザクションが自動的に解決されます。RECO がトランザクションを解決できるまで、データは読取りおよび書込みの両方についてロックされます。読取りをブロックするのは、Oracle が問合せに対してどのバージョンのデータを表示すればよいかを判断できないためです。

この項の内容は、次のとおりです。

- インダウト・トランザクションの自動解決
- インダウト・トランザクションの手動解決
- インダウト・トランザクションの SCN の関連性

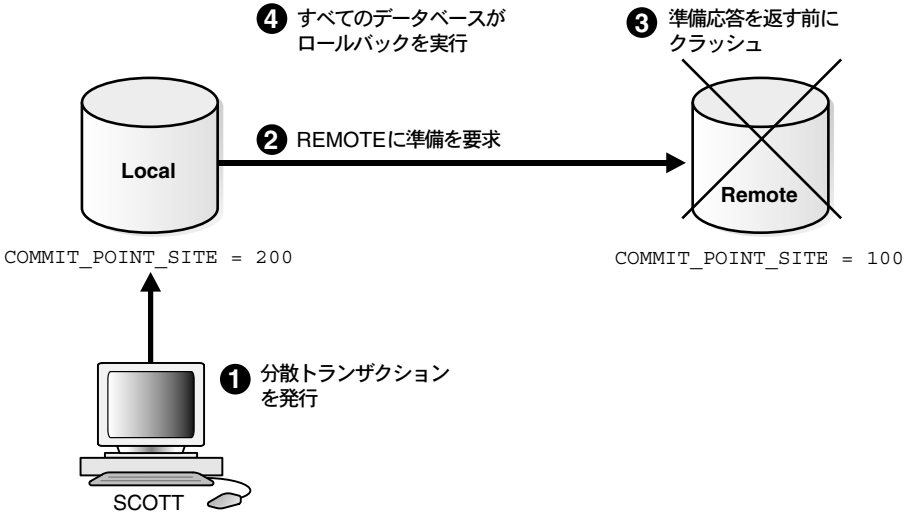
インダウト・トランザクションの自動解決

Oracle では、多くの場合、インダウト・トランザクションは自動的に解決されます。たとえば、次の使用例において `local` と `remote` の 2 つのノードがあるとします。ローカル・ノードはコミット・ポイント・サイトです。ユーザー `scott` は、`local` に接続して `local` と `remote` を更新する分散トランザクションを実行し、コミットします。

準備フェーズ中の障害

図 31-5 は、分散トランザクションの準備フェーズ中に障害が発生したときの一連のイベントを示しています。

図 31-5　準備フェーズ中の障害



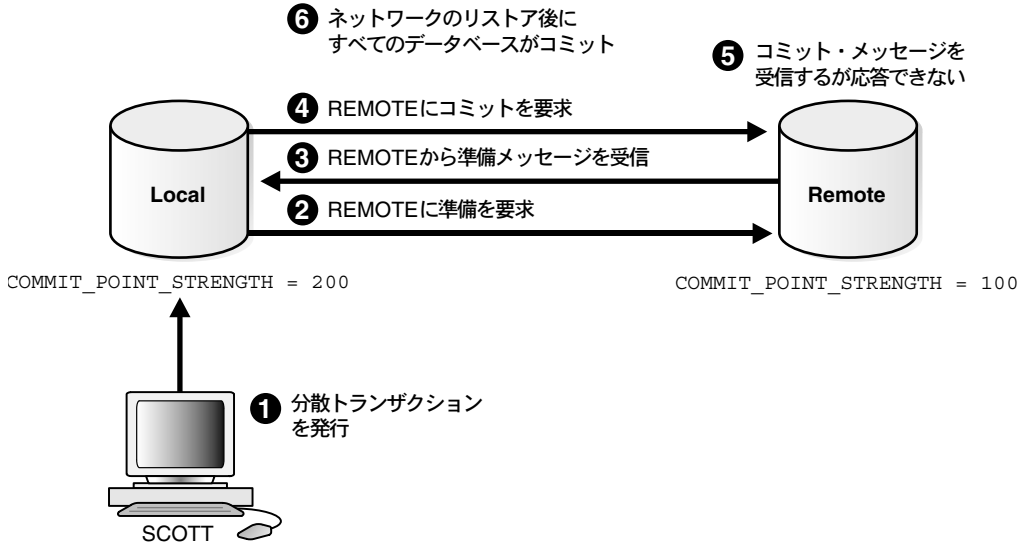
次の手順が発生します。

1. ユーザー Scott は local に接続して分散トランザクションを実行します。
2. グローバル・コーディネータ（この例ではコミット・ポイント・サイトを兼務）は、コミット・ポイント・サイト以外のすべてのデータベースに対して、コミットまたはロールバックの指示があったときにその動作を実行することを確約するように要求します。
3. remote データベースが、local に準備応答を発行する前にクラッシュします。
4. トランザクションは、リモート・サイトのリストア時に、RECO プロセスによって各データベースで最終的にロールバックされます。

コミット・フェーズ中の障害

図 31-6 は、分散トランザクションのコミット・フェーズ中に障害が発生したときの一連のイベントを示しています。

図 31-6　コミット・フェーズ中の障害



次の手順が発生します。

1. ユーザー Scott は local に接続して分散トランザクションを実行します。
2. グローバル・コーディネータ（この場合はコミット・ポイント・サイトを兼務）は、コミット・ポイント・サイト以外のすべてのデータベースに対して、コミットまたはロールバックの指示があったときにその動作を実行することを確約するように要求します。
3. コミット・ポイント・サイトは、コミットの確約を示す準備完了メッセージを remote から受け取ります。
4. コミット・ポイント・サイトはトランザクションをローカルにコミットし、それからコミット・メッセージを remote に送ってコミットを要求します。
5. remote データベースはコミット・メッセージを受け取りましたが、ネットワーク障害のために応答できません。
6. トランザクションは、ネットワークがリストアされた後、RECO プロセスによってリモート・データベースで最終的にコミットされます。

関連項目： 障害状況の説明と、2 フェーズ・コミット中に障害が発生した場合の Oracle の解決方法の詳細は、32-8 ページの「インダウト・トランザクションの処理方法の決定」を参照してください。

インダウト・トランザクションの手動解決

次の場合のみ、インダウト・トランザクションを手動で解決する必要があります。

- インダウト・トランザクションが重要なデータまたはロールバック・セグメントをロックしている場合
- マシン、ネットワークまたはソフトウェアの障害の原因をすぐに修復できない場合

インダウト・トランザクションの解決が複雑になる場合があります。その場合は、次の手順を実行する必要があります。

- インダウト・トランザクションのトランザクション識別番号を特定します。
- DBA_2PC_PENDING ビューおよび DBA_2PC_NEIGHBORS ビューを問い合せ、トランザクションに関係しているデータベースのコミットが完了しているかどうかを確認します。
- 必要であれば、COMMIT FORCE 文を使用してコミットを強制実行するか、または ROLLBACK FORCE 文を使用してロールバックを強制実行します。

関連項目： インダウト・トランザクションの解決方法の詳細は、次の項を参照してください。

- 32-8 ページ「インダウト・トランザクションの処理方法の決定」
- 32-11 ページ「インダウト・トランザクションの手動上書き」

インダウト・トランザクションの SCN の関連性

SCN は、コミット済みバージョンのデータベースの内部的なタイムスタンプです。Oracle データベース・サーバーは、SCN クロック値を使用することでトランザクションの一貫性を保証します。たとえば、ユーザーがトランザクションをコミットするとき、Oracle はそのコミットの SCN をオンライン REDO ログに記録します。

Oracle は、SCN を使用して、異なるデータベース間での分散トランザクションを調整します。たとえば、Oracle は、次のときに SCN を使用します。

1. アプリケーションがデータベース・リンクを使用して接続を確立するとき
2. 分散トランザクションが、それに関係しているすべてのデータベースの中で最も高いグローバル SCN でコミットされるとき
3. コミット・グローバル SCN が、トランザクションに関係しているすべてのデータベースに送られるとき

SCN は、トランザクションが失敗した場合も含め、トランザクションの同期化されたコミット・タイムスタンプとして機能するので、分散トランザクションにとって非常に重要な存在です。トランザクションがインダウトになった場合、管理者はこの SCN を使用して、グローバル・データベースへの変更を調整できます。トランザクション・コミットのグローバル SCN は、分散リカバリの実行時など、後でトランザクションの識別に使用することもできます。

分散トランザクション処理 : 事例

この使用例では、ある会社が `sales.acme.com` および `warehouse.acme.com` という異なる Oracle データベース・サーバーを持っています。ユーザーが売上レコードを `sales` データベースに挿入すると、対応付けられたレコードが `warehouse` データベースで更新されます。

この分散処理の事例では、次のことを示します。

- セッション・ツリーの定義。
- コミット・ポイント・サイトがどのように決定されるか。
- 準備メッセージがいつ送られるか。
- トランザクションがいつ実際にコミットされるか。
- トランザクションに関するどのような情報がローカルに格納されるか。

第 1 段階 : クライアント・アプリケーションによる DML 文の発行

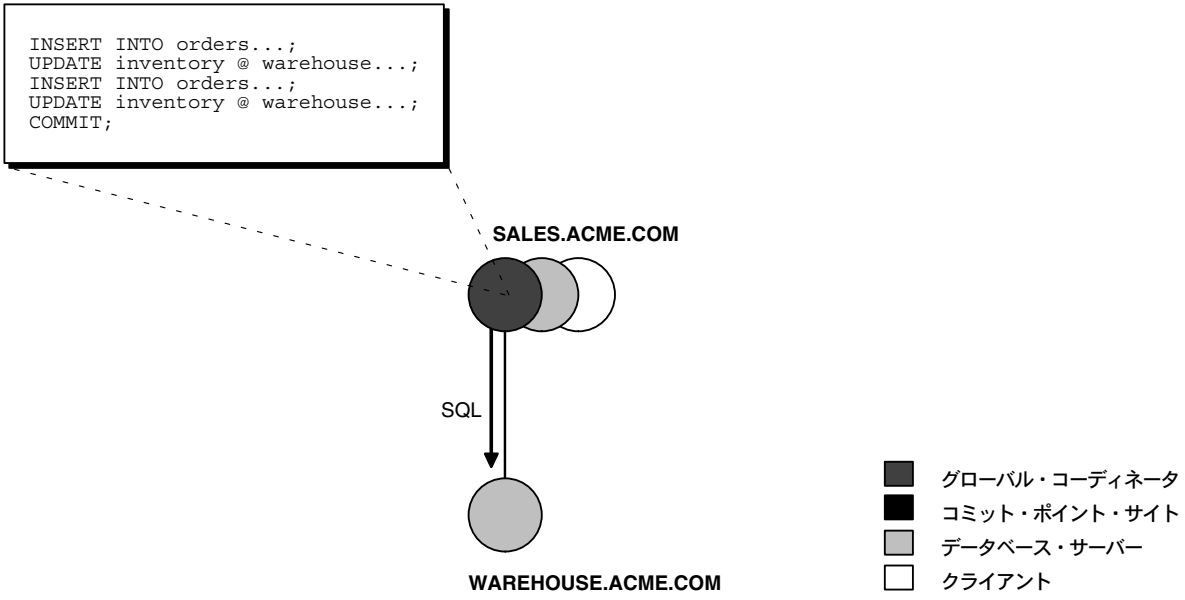
営業部門で、営業担当が SQL*Plus を使用して売上注文を入力し、コミットします。アプリケーションは次のような SQL 文を発行し、sales データベースに注文を入力して、warehouse データベースの inventory 表を更新します。

```
CONNECT scott/tiger@sales.acme.com ...;
INSERT INTO orders ...;
UPDATE inventory@warehouse.acme.com ...;
INSERT INTO orders ...;
UPDATE inventory@warehouse.acme.com ...;
COMMIT;
```

これらの SQL 文は単一の分散トランザクションの一部であり、発行されるすべての SQL 文がユニットとして成功または失敗することが保証されています。文をユニットとして扱うことにより、注文が設定されているにもかかわらず、注文を反映するように在庫が更新されていないという事態を防ぐことができます。実際は、トランザクションによってグローバル・データベースのデータの一貫性が保証されています。

トランザクションの各 SQL 文が実行されると、図 31-7 のようにセッション・ツリーが定義されます。

図 31-7　セッション・ツリーの定義



トランザクションの次の点に注意してください。

- トランザクションを開始するのは、sales データベースで実行されている注文入力アプリケーションです。したがって、分散トランザクションのグローバル・コーディネータは、sales.acme.com になります。
- 注文入力アプリケーションは、新しい売上レコードを sales データベースに挿入し、warehouse データベースの inventory 表を更新します。したがって、ノード sales.acme.com と warehouse.acme.com はどちらもデータベース・サーバーになります。
- sales.acme.com は inventory 表を更新するため、warehouse.acme.com のクライアントになります。

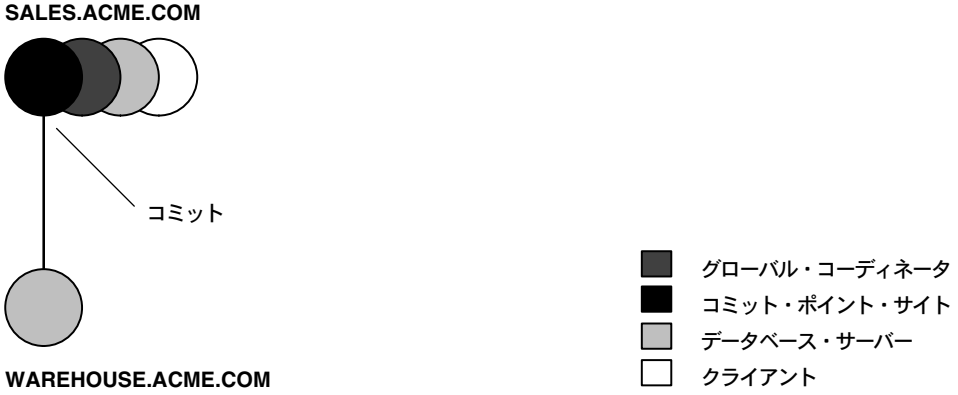
この段階では、この分散トランザクションのセッション・ツリーの定義が完了します。ツリー内の各ノードは、必要なデータ・ロックを獲得して、ローカル・データを参照する SQL 文を実行します。これらのロックは、SQL 文の実行が完了した後も、2 フェーズ・コミットが完了するまで保持されます。

第 2 段階 : Oracle によるコミット・ポイント・サイトの判別

Oracle は、COMMIT 文の直後にコミット・ポイント・サイトを判別します。図 31-8 のように、グローバル・コーディネータの sales.acme.com がコミット・ポイント・サイトとして判別されます。

> **関連項目 :** コミット・ポイント・サイトの判別の詳細は、31-8 ページの「コミット・ポイント強度」を参照してください。

図 31-8 コミット・ポイント・サイトの判別



第 3 段階 : グローバル・コーディネータによる準備応答の送信

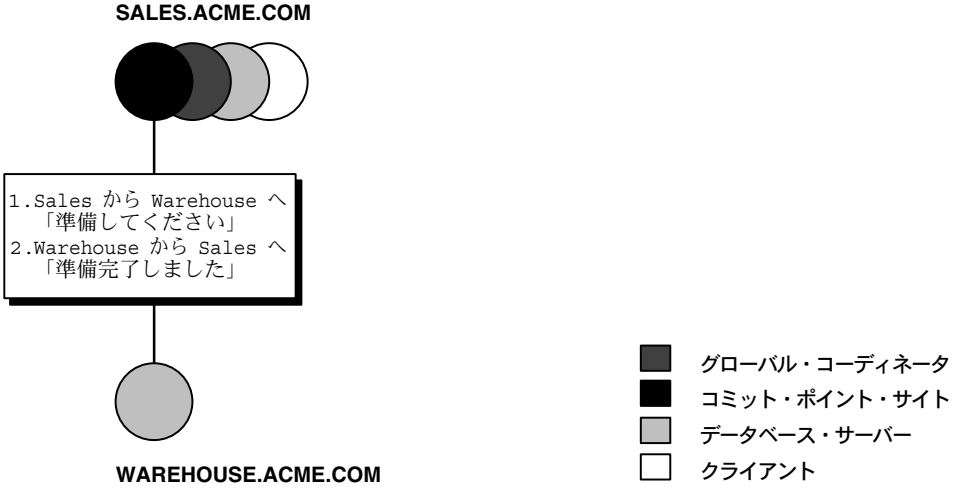
準備段階では、次の手順が実行されます。

1. Oracle がコミット・ポイント・サイトを判別した後、グローバル・コーディネータは、コミット・ポイント・サイトを除くセッション・ツリーの直接参照先のノードすべてに準備メッセージを送ります。この例では、準備を要求されるノードは warehouse.acme.com のみです。
2. ノード warehouse.acme.com は準備を試みます。トランザクション内のローカルに独立した部分をコミットし、自身のローカル REDO ログにコミット情報を記録できることをノードが保証できれば、そのノードは正常に準備できます。この例では、sales.acme.com がコミット・ポイント・サイトなので、warehouse.acme.com のみが準備メッセージを受け取ります。
3. ノード warehouse.acme.com は、準備完了メッセージで sales.acme.com に応答します。

各ノードが準備を終えると、準備を要求した側のノードに応答メッセージが送り返されます。応答に応じて、次のいずれかの動作が発生します。

- 準備を要求されたノードのうち、異常終了メッセージでグローバル・コーディネータに応答したノードが 1 つでもある場合、グローバル・コーディネータはトランザクションをロールバックするようにすべてのノードに指示し、操作は完了します。
- 準備を要求されたノードのすべてが準備完了または読取り専用のメッセージでグローバル・コーディネータに応答した場合、つまり、すべてのノードの準備が正常に完了した場合、グローバル・コーディネータはトランザクションをコミットするようにコミット・ポイント・サイトに要求します。

図 31-9　準備メッセージの送信と確認



第 4 段階 : コミット・ポイント・サイトによるコミット

コミット・ポイント・サイトによるトランザクションのコミットでは、次の手順が実行されます。

1. ノード sales.acme.com は、warehouse.acme.com の準備が完了しているという確認を受け取り、トランザクションをコミットするようにコミット・ポイント・サイトに指示します。
2. この時点で、コミット・ポイント・サイトはトランザクションをローカルにコミットし、この操作を自身のローカル REDO ログに記録します。

warehouse.acme.com がまだコミットを完了していない場合でも、このトランザクションの結果は事前に決まっています。つまり、対象となるノードのコミットが遅れた場合でも、トランザクションはすべてのノードでコミットされます。

第 5 段階 : コミット・ポイント・サイトによるグローバル・コーディネータへのコミットの通知

この段階では、次の手順が実行されます。

1. コミット・ポイント・サイトは、トランザクションが完了したことをグローバル・コーディネータに伝えます。この例では、コミット・ポイント・サイトとグローバル・コーディネータが同じノードなので、必要な操作はありません。コミット・ポイント・サイトでは、トランザクションがコミットされたことがわかっています。これは、トランザクションがコミットされたことが自身のオンライン・ログに記録されているためです。
2. グローバル・コーディネータは、分散トランザクションに関係している他のすべてのノードでトランザクションがコミットされたことを確認します。

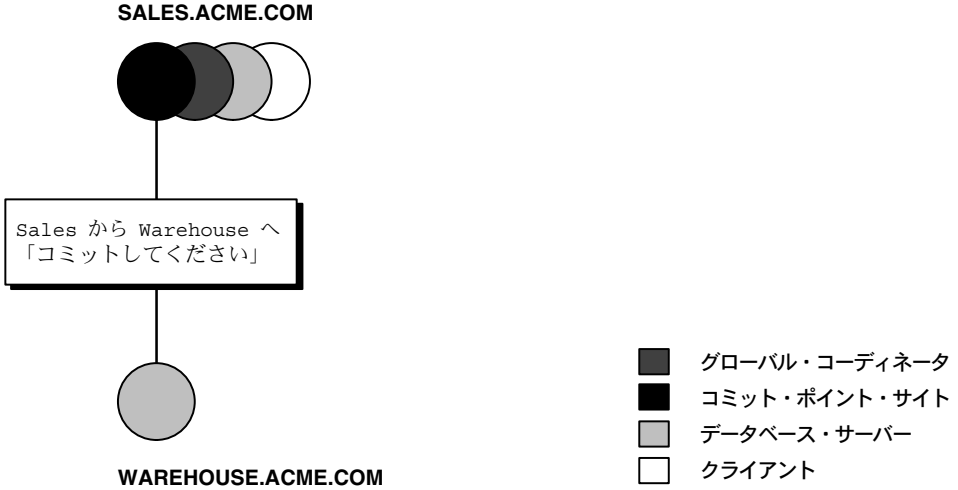
第 6 段階 : グローバルおよびローカル・コーディネータによる全ノードへのコミットの要求

トランザクション内のすべてのノードによるトランザクションのコミットでは、次の手順が実行されます。

1. コミット・ポイント・サイトでのコミットがグローバル・コーディネータに通知された後、グローバル・コーディネータは、直接参照している他のすべてのノードに対してコミットを指示します。
2. コミットの指示を受けたローカル・コーディネータは、次に自身のサーバーに対してコミットを指示し、以下同様にコミットの指示が伝播されます。
3. グローバル・コーディネータを含む各ノードは、トランザクションをコミットし、該当する REDO ログ・エントリをローカルに記録します。各ノードのコミットが完了すると、そのトランザクションのためにローカルに保持されていたリソース・ロックが解放されます。

図 31-10 では、コミット・ポイント・サイトとグローバル・コーディネータを兼務している `sales.acme.com` が、トランザクションのローカルのコミットをすでに完了しています。この時点で、`sales` は `warehouse.acme.com` に対してトランザクションのコミットを指示します。

図 31-10　ノードへのコミットの指示



第 7 段階 : グローバル・コーディネータとコミット・ポイント・サイトによるコミットの完了

トランザクションのコミットの完了は、次の手順で実行されます。

1. すべての参照先ノードとグローバル・コーディネータがトランザクションのコミットを完了した後、グローバル・コーディネータはこれをコミット・ポイント・サイトに通知します。
2. このメッセージを待っていたコミット・ポイント・サイトは、この分散トランザクションに関するステータス情報を消去します。
3. コミット・ポイント・サイトは、消去が完了したことをグローバル・コーディネータに伝えます。つまり、コミット・ポイント・サイトは、分散トランザクションのコミットに関する情報をまったく保持しません。2 フェーズ・コミットに関係しているすべてのノードでトランザクションのコミットが正常に完了すると、それらのノードで今後ノード自身のステータスを判断する必要はありません。したがって、このような処理が許可されます。
4. グローバル・コーディネータは、トランザクション自体に関する情報を消去することでトランザクションを完了します。

COMMIT フェーズの完了後、分散トランザクションそのものが完了します。これまで説明した手順は、1 秒以内に自動的に実行されます。

32

分散トランザクションの管理

分散トランザクションの管理とトラブルシューティングの方法について説明します。この章の内容は、次のとおりです。

ノードのコミット・ポイント強度の指定

分散トランザクションでどのノードが最初にコミットされるかは、最も高いコミット・ポイント強度を持つデータベースによって決まります。各ノードのコミット・ポイント強度を指定するときは、準備フェーズまたはコミット・フェーズ中に障害が発生した場合に最も重要なサーバーがブロックされないようにしてください。ノードのコミット・ポイント強度は、COMMIT_POINT_STRENGTH 初期化パラメータによって指定します。

デフォルト値は、オペレーティング・システムによって異なります。値の範囲は、0（ゼロ）から 255 までの任意の整数です。たとえば、データベースのコミット・ポイント強度を 200 に設定するには、そのデータベースの初期化パラメータ・ファイルに次の行を追加します。

```
COMMIT_POINT_STRENGTH = 200
```

コミット・ポイント強度は、分散トランザクションでコミット・ポイント・サイトを決定する際にのみ使用されます。

データベースのコミット・ポイント強度を設定するときは、次の点を考慮してください。

- コミット・ポイント・サイトは、トランザクションのステータスに関する情報を格納します。そのため、他のノードがトランザクションのステータスに関する情報を必要とする場合に備えて、信頼性が低下したり、使用できなくなったりする頻度が高いノードをコミット・ポイント・サイトにしないでください。
- データベースのコミット・ポイント強度は、データベース内の重要な共有データの量に比例するように設定してください。たとえば、メインフレーム・コンピュータのデータベースは、一般に PC のデータベースよりも多くのデータをユーザー間で共有しています。したがって、メインフレームのコミット・ポイント強度は、PC のコミット・ポイント強度よりも高い値に設定します。

関連項目： コミット・ポイントの概念の詳細は、31-7 ページの「コミット・ポイント・サイト」を参照してください。

トランザクションの命名

Oracle9*i* から、トランザクションの命名が可能になりました。この機能は、特定の分散トランザクションを識別する際に便利であり、同じ目的の COMMIT COMMENT 文にかわるものです。

トランザクションに命名するには SET TRANSACTION ... NAME 文を使用します。次に例を示します。

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
     NAME 'update inventory checkpoint 0';
```

この例は、SERIALIZABLE に等しい分離レベルを持つ新しいトランザクションをユーザーが開始し、それに 'update inventory checkpoint 0' という名前を付けたことを示しています。

分散トランザクションでは、トランザクションがコミットされるときに、名前が参加サイトに送られます。トランザクション名が存在する場合は、COMMIT COMMENT があっても無視されます。

トランザクション名は、V$TRANSACTION ビューの NAME 列に表示されます。

分散トランザクション情報の表示

各データベースのデータ・ディクショナリには、オープンしているすべての分散トランザクションに関する情報が格納されています。データ・ディクショナリの表とビューを使用すると、トランザクションに関する情報を取得できます。この項の内容は、次のとおりです。

- 準備完了トランザクションの ID 番号と状態の判断
- インダウト・トランザクションのセッション・ツリーのトレース

準備完了トランザクションの ID 番号と状態の判断

次のビューには、ローカル・データベースで定義され、データ・ディクショナリに格納されているデータベース・リンクが表示されます。

ビュー	用途
DBA_2PC_PENDING	インダウト分散トランザクションがすべてリストされます。このビューには、インダウト・トランザクションが移入されるまで何もデータが入っていません。トランザクションが解決された後、ビューはパージされます。

このビューは、特定のトランザクション ID のグローバル・コミット番号を判断するために使用します。このグローバル・コミット番号は、インダウト・トランザクションを手動で解決するときに使用できます。

関連性の最も高い列を次の表に示します（ビューのすべての列の詳細は、『Oracle9*i* データベース・リファレンス』を参照してください）。

表 32-1 DBA_2PC_PENDING

列	説明
LOCAL_TRAN_ID	「*integer.integer.integer*」という書式のローカル・トランザクション識別子。 **注意：**接続の LOCAL_TRAN_ID と GLOBAL_TRAN_ID が同じ場合、そのノードはトランザクションのグローバル・コーディネータです。
GLOBAL_TRAN_ID	「*global_db_name.db_hex_id.local_tran_id*」という書式のグローバル・データベース識別子。ここで、*db_hex_id* は、データベースを一意に識別するために使用される 8 文字の 16 進値です。この共通のトランザクション ID は、1 つの分散トランザクションに関係するすべてのノードで同じです。 **注意：**接続の LOCAL_TRAN_ID と GLOBAL_TRAN_ID が同じ場合、そのノードはトランザクションのグローバル・コーディネータです。

表 32-1 DBA_2PC_PENDING（続き）

列	説明
STATE	STATE に可能な値は、次のとおりです。 ■ Collecting 通常、このカテゴリは、グローバル・コーディネータまたはローカル・コーディネータにのみ適用されます。ノードは現在他のデータベース・サーバーから情報を収集中であり、その後ノード自身が準備できるかどうかが判断されます。 ■ Prepared ノードは準備を完了していますが、そのことをローカル・コーディネータに準備完了メッセージで通知しているかどうかは不明です。しかし、コミット要求はまだ受け取っていません。ノードは準備完了状態にあり、トランザクションのコミットに必要なローカル・リソースのロックをすべて保持しています。 ■ Commited ノード（任意のタイプ）はトランザクションのコミットを完了していますが、トランザクションに関係している他のノードが同じように完了していない可能性があります。つまり、トランザクションは 1 つ以上のノードでまだ保留しています。 ■ Forced Commit ペンディング・トランザクションは、データベース管理者（DBA）の判断で強制的にコミットできます。このエントリは、ローカル・ノードでトランザクションが手動でコミットされた場合に表示されます。 ■ Forced termination (rollback) ペンディング・トランザクションは、DBA の判断で強制的にロールバックできます。このエントリは、ローカル・ノードでこのトランザクションが手動でロールバックされた場合に表示されます。
MIXED	YES は、トランザクションの一部があるノードではコミットされ、別のノードではロールバックされたことを示します。
TRAN_COMMENT	トランザクションがコミットされると、トランザクション・コメントまたはトランザクション名（トランザクションに命名している場合）がこの列に設定されます。
HOST	ホスト・マシンの名前。
COMMIT#	コミットされたトランザクションのグローバル・コミット番号。

DBA_2PC_PENDING の関連情報を問い合せるには、次のスクリプト pending_txn_script を実行します（出力例も含まれています）。

```
COL LOCAL_TRAN_ID FORMAT A13
COL GLOBAL_TRAN_ID FORMAT A30
COL STATE FORMAT A8
COL MIXED FORMAT A3
COL HOST FORMAT A10
COL COMMIT# FORMAT A10

SELECT LOCAL_TRAN_ID, GLOBAL_TRAN_ID, STATE, MIXED, HOST, COMMIT#
FROM DBA_2PC_PENDING
/

SQL> @pending_txn_script

LOCAL_TRAN_ID GLOBAL_TRAN_ID                  STATE    MIX HOST       COMMIT#
------------- ------------------------------- -------- --- ---------- ----------
1.15.870      HQ.ACME.COM.ef192da4.1.15.870  commit   no  dlsun183   115499
```

この出力は、ローカル・トランザクション 1.15.870 がこのノードではコミットを完了しているものの、他の 1 つ以上のノードで保留している可能性があることを示しています。LOCAL_TRAN_ID と GLOBAL_TRAN_ID のローカル部分が同じなので、このノードはトランザクションのグローバル・コーディネータです。

インダウト・トランザクションのセッション・ツリーのトレース

次のビューには、リモート・クライアントから受信されるインダウト・トランザクションと、リモート・サーバーに送信されるインダウト・トランザクションが表示されます。

ビュー	用途
DBA_2PC_NEIGHBORS	インダウト分散トランザクションについて、リモート・クライアントから受信されるものとリモート・サーバーに送信されるものがすべてリストされます。また、ローカル・ノードがトランザクションのコミット・ポイント・サイトかどうかも示されます。 このビューには、インダウト・トランザクションが移入されるまで何もデータが入っていません。トランザクションが解決された後、ビューはパージされます。

トランザクションがインダウトのときは、セッション・ツリー内のどのノードがどのロールを実行しているのかを判断することが必要な場合があります。このビューを使用すると、次のことが判断できます。

- 特定のトランザクションのすべての受信接続と送信接続。
- ノードが特定のトランザクションのコミット・ポイント・サイトかどうか。
- ノードが特定のトランザクションのグローバル・コーディネータかどうか（ノードのローカル・トランザクション ID とグローバル・トランザクション ID が同じかどうかで判断する）。

関連性の最も高い列を次の表に示します（ビューのすべての列の詳細は、『Oracle9*i* データベース・リファレンス』を参照してください）。

表 32-2 DBA_2PC_NEIGHBORS

列	説明
LOCAL_TRAN_ID	「*integer.integer.integer*」という書式のローカル・トランザクション識別子。 **注意：**接続の LOCAL_TRAN_ID と GLOBAL_TRAN_ID.DBA_2PC_PENDING が同じ場合、そのノードはトランザクションのグローバル・コーディネータです。
IN_OUT	受信トランザクションの場合は IN、送信トランザクションの場合は OUT です。
DATABASE	受信トランザクションの場合は、このローカル・ノードから情報を要求したクライアント・データベースの名前です。送信トランザクションの場合は、リモート・サーバーの情報へのアクセスに使用されたデータベース・リンクの名前です。
DBUSER_OWNER	受信トランザクションの場合は、リモート・データベース・リンクによる接続で使用されるローカル・アカウントです。送信トランザクションの場合は、データベース・リンクの所有者です。
INTERFACE	C はコミット・メッセージ、N は準備完了状態を示すメッセージまたは読取り専用コミットの要求のどちらかです。 IN_OUT が OUT の場合、C は、接続のリモート側の子がコミット・ポイント・サイトであり、コミットと終了のどちらを実行するかを知っていることを意味します。N は、ローカル・ノードの準備が完了したことをリモート・ノードに通知中であることを意味します。 IN_OUT が IN の場合、C は、送信接続のリモート側のローカル・ノードまたはデータベースがコミット・ポイント・サイトであることを表します。N は、リモート・ノードの準備が完了したことをローカル・ノードに通知中であることを意味します。

DBA_2PC_PENDING の関連情報を問い合せるには、次のスクリプト neighbors_script を実行します（出力例も含まれています）。

```
COL LOCAL_TRAN_ID FORMAT A13
COL IN_OUT FORMAT A6
COL DATABASE FORMAT A25
COL DBUSER_OWNER FORMAT A15
COL INTERFACE FORMAT A3
SELECT LOCAL_TRAN_ID, IN_OUT, DATABASE, DBUSER_OWNER, INTERFACE
FROM DBA_2PC_NEIGHBORS
/

SQL> CONNECT SYS/password@hq.acme.com
SQL> @neighbors_script

LOCAL_TRAN_ID IN_OUT DATABASE                  DBUSER_OWNER    INT
------------- ------ ------------------------- --------------- ---
1.15.870      out    SALES.ACME.COM            SYS             C
```

この出力は、トランザクション 1.15.870 をコミットするための送信要求をローカル・ノードがリモート・サーバー sales に送ったことを示します。sales がトランザクションをコミットしたものの、他のノードがコミットしなかった場合は、sales がコミット・ポイント・サイトであることが判明します。コミット・ポイント・サイトは、常に最初にコミットされるためです。

インダウト・トランザクションの処理方法の決定

2 フェーズ・コミットの間に障害が発生すると、トランザクションはインダウトになります。分散トランザクションがインダウトになる要因は、次のとおりです。

- Oracle ソフトウェアを実行しているサーバー・マシンがクラッシュした。
- 分散処理に関係している複数の Oracle データベース間のネットワーク接続が切断された。
- 未処理のソフトウェア・エラーが発生した。

> **関連項目：** インダウト・トランザクションの概念の詳細は、31-16 ページの「インダウト・トランザクション」を参照してください。

ローカルのインダウト分散トランザクションは、手動で強制的にコミットまたはロールバックできます。この操作では一貫性の問題が生じる可能性があるため、特定の条件が成り立つとき以外は実行しないでください。

この項の内容は、次のとおりです。

- 2 フェーズ・コミットに関する問題の検出
- 手動上書きを実行するかどうかの判断
- トランザクション・データの分析

2 フェーズ・コミットに関する問題の検出

分散トランザクションをコミットするユーザー・アプリケーションには、次のいずれかのエラー・メッセージによって問題が通知されます。

```
ORA-02050: トランザクション ID はロールバックされました。
           いくつかのリモート・データベースはインダウトの可能性があります。
ORA-02053: トランザクション ID はコミットしました。
           いくつかのリモート・データベースはインダウトの可能性があります。
ORA-02054: トランザクション ID はインダウトです
```

アプリケーションでこれらのエラーを受け取った場合は、トランザクションに関する情報を保存するようにしてください。この情報は、後で分散トランザクションの手動リカバリが必要になった場合に使用できます。

ネットワークまたはシステムの障害のために、任意のノードでインダウト分散トランザクションが 1 つ以上発生した場合でも、そのノードの管理者がなんらかの処理を実行する必要はありません。ネットワークやシステムの障害が解決した後、Oracle の自動リカバリ機能によって、セッション・ツリーのすべてのノードが同じ結果になるように（つまり、すべてコミットされるかすべてロールバックされる）、すべてのインダウト・トランザクションの処理が透過的に完了します。

ただし、障害が長期にわたる場合には、トランザクションを強制的にコミットまたはロールバックすることで、ロックされたデータをすべて解放できます。アプリケーションは、この操作を想定しておく必要があります。

手動上書きを実行するかどうかの判断

次の条件が成り立つときのみ、特定のインダウト・トランザクションを手動で上書きしてください。

- インダウト・トランザクションが他のトランザクションに必要なデータをロックしている場合。この状況は、ORA-01591 エラー・メッセージによってユーザーのトランザクションが妨げられたときに起こります。
- インダウト・トランザクションによって、他のトランザクションがロールバック・セグメントのエクステントを使用できなくなった場合。インダウト分散トランザクションのローカル・トランザクション ID は、その最初の部分がロールバック・セグメントの ID に対応しています。この ID は、データ・ディクショナリ・ビュー DBA_2PC_PENDING および DBA_ROLLBACK_SEGS にリストされます。

- 2 フェーズ・コミットの各フェーズの完了を妨げている障害を許容される時間範囲内で訂正できない場合。このような例としては、通信ネットワークやデータベースの損傷など、リカバリに長い時間を要する障害があります。

通常は、他の場所の管理者と相談した上で、インダウト分散トランザクションを強制的にローカルに処理するかどうかを決めてください。判断を誤った場合は、トレースの困難なデータベースの非一貫性が生じる可能性があります。この非一貫性は、手動で訂正する必要があります。

前述の条件が当てはまらない場合は、必ず Oracle の自動リカバリ機能によってトランザクションを完了するようにしてください。しかし、前述のいずれかの条件に当てはまる場合には、インダウト・トランザクションのローカルでの修正を検討してください。

トランザクション・データの分析

トランザクションの強制完了を決めた場合は、次の目的に留意しながら、使用可能な情報を分析します。

コミットまたはロールバックされたノードの特定

DBA_2PC_PENDING ビューを使用し、トランザクションをコミットまたはロールバックしたノードを探します。トランザクションをすでに解決しているノードが見つかった場合は、そのノードで実行された処理に従うことができます。

トランザクション・コメントの確認

DBA_2PC_PENDING の TRAN_COMMENT 列に、対象の分散トランザクションに関するなんらかの情報が与えられていないかを確認します。コメントは、COMMIT 文の COMMENT 句で指定されます。また、トランザクションに命名している場合は、トランザクションがコミットされたときに、トランザクション名が TRAN_COMMENT フィールドに設定されます。

たとえば、インダウト分散トランザクションのコメントで、トランザクションの開始元とタイプを示すことができます。

```
COMMIT COMMENT 'Finance/Accts_pay/Trans_type 10B';
```

また、この情報をトランザクション名として提供するために、SET TRANSACTION ... NAME 文を使用することも可能です。

関連項目： 32-3 ページ「トランザクションの命名」

トランザクション・アドバイスの確認

DBA_2PC_PENDING の ADVICE 列に、対象の分散トランザクションに関するなんらかの情報が与えられていないかを確認します。アプリケーションで ALTER SESSION 文の ADVISE 句を使用すれば、分散トランザクションの別々の部分を強制的にコミットまたはロールバックする際のアドバイスを規定できます。

準備フェーズ中に各ノードに送られたアドバイスは、現行トランザクション内でそのデータベースに対し最新の DML 文が実行されたときに有効なアドバイスです。

たとえば、あるノードの emp 表から別のノードの emp 表に従業員レコードを移動する分散トランザクションを考えます。次の一連の SQL 文を追加することにより、たとえ管理者が各ノードで個別にインダウト・トランザクションを強制的に完了したとしても、トランザクションでレコードを保護できます。

```
ALTER SESSION ADVISE COMMIT;
INSERT INTO emp@hq ... ;    /*advice to commit at HQ */
ALTER SESSION ADVISE ROLLBACK;
DELETE FROM emp@sales ... ; /*advice to roll back at SALES*/

ALTER SESSION ADVISE NOTHING;
```

与えられたアドバイスに従ってインダウト・トランザクションを手動で強制的に完了すると、最悪の場合、各ノードに従業員レコードがコピーされ、消去できなくなる可能性があります。

インダウト・トランザクションの手動上書き

COMMIT 文または ROLLBACK 文に、FORCE オプションと、コミットまたはロールバックするインダウト・トランザクションのローカル・トランザクション ID またはグローバル・トランザクション ID を示すテキスト文字列を指定します。

注意： 以降のすべての例で、トランザクションはローカル・ノードでコミットまたはロールバックされます。ローカルのペンディング・トランザクション表では、このトランザクションの行の STATE 列に強制コミットまたは強制終了の値が記録されます。

この項の内容は、次のとおりです。

- インダウト・トランザクションの手動コミット
- インダウト・トランザクションの手動ロールバック

インダウト・トランザクションの手動コミット

トランザクションのコミットを試みる前に、適正な権限を持っていることを確認してください。必要な権限は次のとおりです。

トランザクションをコミットした実行者	必要な権限
管理者	FORCE TRANSACTION
別のユーザー	FORCE ANY TRANSACTION

トランザクション ID のみを使用したコミット

次の SQL 文は、インダウト・トランザクションをコミットします。

```
COMMIT FORCE 'transaction_id';
```

変数 *transaction_id* は、DBA_2PC_PENDING データ・ディクショナリ・ビューの LOCAL_TRAN_ID 列または GLOBAL_TRAN_ID 列のどちらかで指定されているトランザクション識別子です。

たとえば、DBA_2PC_PENDING を問い合せて、分散トランザクションの LOCAL_TRAN_ID が 1:45.13 であることを確認するとします。

そして、次の SQL 文を発行し、このインダウト・トランザクションのコミットを強制実行します。

```
COMMIT FORCE '1.45.13';
```

システム変更番号（SCN）を使用したコミット

必要であれば、トランザクションを強制的にコミットするときに、トランザクションの SCN を指定することもできます。この機能を使用すると、他のノードでトランザクションがコミットされたときに割り当てられた SCN を使用して、インダウト・トランザクションをコミットできます。

これにより、障害が発生した場合でも、分散トランザクションのコミット時間の同期を保つことができます。SCN は、別のノードですでにコミットされた同じトランザクションの SCN が判別できるときのみ指定してください。

たとえば、次のグローバル・トランザクション ID を使用してトランザクションを手動でコミットするとします。

```
SALES.ACME.COM.55d1c563.1.93.29
```

まず、対象のトランザクションに関係しているリモート・データベースの DBA_2PC_PENDING ビューを問い合せます。次に、そのノードでトランザクションのコミットに使用された SCN をメモします。そして、ローカル・ノードでトランザクションをコミットする

ときにこの SCN を指定します。たとえば、SCN が 829381993 の場合は、次の文を発行します。

```
COMMIT FORCE 'SALES.ACME.COM.55d1c563.1.93.29', 829381993;
```

関連項目： COMMIT 文の使用の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

インダウト・トランザクションの手動ロールバック

インダウト分散トランザクションのロールバックを試みる前に、適正な権限を持っていることを確認してください。必要な権限は次のとおりです。

トランザクションをコミットした実行者	必要な権限
管理者	FORCE TRANSACTION
別のユーザー	FORCE ANY TRANSACTION

次の SQL 文は、インダウト・トランザクションをロールバックします。

```
ROLLBACK FORCE 'transaction_id';
```

変数 *transaction_id* は、DBA_2PC_PENDING データ・ディクショナリ・ビューの LOCAL_TRAN_ID 列または GLOBAL_TRAN_ID 列のどちらかで指定されているトランザクション識別子です。

たとえば、ローカル・トランザクション ID が 2.9.4 のインダウト・トランザクションをロールバックするには、次の文を使用します。

```
ROLLBACK FORCE '2.9.4';
```

注意： インダウト・トランザクションをセーブポイントまでロールバックすることはできません。

関連項目： ROLLBACK 文の使用の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

データ・ディクショナリからの保留行のパージ

インダウト・トランザクションは、RECO（リカバラ・プロセス）によってリカバリされる前に、DBA_2PC_PENDING.STATE 列に COLLECTING、COMMITTED または PREPARED として表示されます。COMMIT FORCE または ROLLBACK FORCE を使用してインダウト・トランザクションを強制的に完了すると、FORCED COMMIT または FORCED ROLLBACK の状態になることがあります。

自動リカバリでは通常、これらの状態にあるエントリが削除されます。唯一の例外として、リカバリ時に、トランザクション内の他のサイトと一貫性のない状態にある強制トランザクションが検出されるときがあります。この場合は、エントリが表内に残る可能性があり、DBA_2PC_PENDING の MIXED 列の値が YES になります。これらのエントリは、DBMS_TRANSACTION.PURGE_MIXED プロシージャを使用してクリーンアップできます。

リモート・データベースが永続的に失われたために自動リカバリが不可能な場合は、データベースを再作成してもリカバリではそのデータベースを識別できません。これは、再作成時に新しいデータベース ID が割り当てられるためです。この場合は、DBMS_TRANSACTION パッケージの PURGE_LOST_DB_ENTRY プロシージャを使用して、エントリをクリーンアップする必要があります。このエントリによってデータベース・リソースが保持されることはないため、エントリのクリーンアップを急ぐ必要はありません。

PURGE_LOST_DB_ENTRY プロシージャの実行

エントリをデータ・ディクショナリから手動で削除するには、次の構文を使用します（ここで、*trans_id* はトランザクションの識別子を表します）。

```
DBMS_TRANSACTION.PURGE_LOST_DB_ENTRY('trans_id');
```

たとえば、保留中の分散トランザクション 1.44.99 をパージするには、SQL*Plus で次の文を入力します。

```
EXECUTE DBMS_TRANSACTION.PURGE_LOST_DB_ENTRY('1.44.99');
```

このプロシージャは、自動リカバリによってトランザクションを解決できないほどの大幅な再構成を行った場合にのみ実行してください。たとえば、次のような場合です。

- リモート・データベースが完全に失われた場合
- ソフトウェアを再構成した結果、2 フェーズ・コミットの機能がなくなった場合
- TPMonitor などの外部トランザクション・コーディネータからの情報が失われた場合

関連項目： DBMS_TRANSACTION パッケージの詳細は、『Oracle9*i* 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。

DBMS_TRANSACTION を使用する時期の判断

次の表は、分散トランザクションに関する各種の状態と、それに対して管理者が実行すべき処理を示したものです。

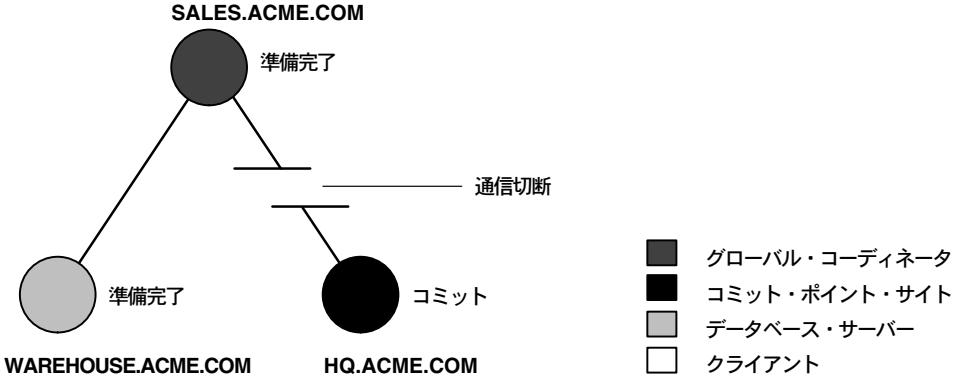
STATE 列	グローバル・トランザクションの状態	ローカル・トランザクションの状態	通常の処理	代替処理
Collecting	Rolled back	Rolled back	なし	PURGE_LOST_DB_ENTRY（自動リカバリでトランザクションを解決できない場合のみ）
Commited	Commited	Commited	なし	PURGE_LOST_DB_ENTRY（自動リカバリでトランザクションを解決できない場合のみ）
Prepared	Unknown	Prepared	なし	強制コミットまたは強制ロールバック
Forced commit	Unknown	Commited	なし	PURGE_LOST_DB_ENTRY（自動リカバリでトランザクションを解決できない場合のみ）
Forced rollback	Unknown	Rolled back	なし	PURGE_LOST_DB_ENTRY（自動リカバリでトランザクションを解決できない場合のみ）
Forced commit	Mixed	Commited	非一貫性部分を手動で削除してから PURGE_MIXED を使用する	-
Forced rollback	Mixed	Rolled back	非一貫性部分を手動で削除してから PURGE_MIXED を使用する	-

関連項目： DBMS_TRANSACTION パッケージの詳細は、『Oracle9*i* 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。

インダウト・トランザクションの手動コミット : 例

図 32-1 は、分散トランザクションのコミット中の障害を示しています。この障害例では、準備フェーズは完了しています。しかし、コミット・フェーズ時に、コミット・ポイント・サイトがトランザクションをコミットしていても、コミット・ポイント・サイトのコミット確認がグローバル・コーディネータに到達しません。インダウト・トランザクションは他のトランザクションにとっても重要なものであるため、在庫データはロックされており、アクセスできません。また、インダウト・トランザクションがコミットまたはロールバックされるまで、ロックが必ず保持されます。

図 32-1　インダウト分散トランザクションの例



次の手順に従って、インダウト・トランザクションのローカル部分を手動で強制完了できます。詳細は、以降の項を参照してください。

手順 1: ユーザーからのフィードバックの記録

手順 2: DBA_2PC_PENDING の問合せ

手順 3: ローカル・ノードでの DBA_2PC_NEIGHBORS の問合せ

手順 4: 全ノードでのデータ・ディクショナリ・ビューの問合せ

手順 5: インダウト・トランザクションのコミット

手順 6: DBA_2PC_PENDING を使用した MIXED 結果のチェック

手順 1: ユーザーからのフィードバックの記録

インダウト・トランザクションのロックと競合を起こしているローカル・データベース・システムのユーザーは、次のエラー・メッセージを受け取ります。

```
ORA-01591: インダウト分散トランザクション 1.21.17 がロックを保持しています。
```

この場合、1.21.17 はインダウト分散トランザクションのローカル・トランザクション ID です。どのインダウト・トランザクションを強制処理すればよいかを特定するために、問題を報告したユーザーからこの ID 番号を聞き、記録しておきます。

手順 2: DBA_2PC_PENDING の問合せ

SQL*Plus を使用して warehouse に接続した後、ローカルの DBA_2PC_PENDING データ・ディクショナリ・ビューを問い合せて、インダウト・トランザクションに関する情報を取得します。

```
CONNECT SYS/password@warehouse.acme.com
SELECT * FROM DBA_2PC_PENDING WHERE LOCAL_TRAN_ID = '1.21.17';
```

次の情報が表示されます。

```
Column Name            Value
---------------------- -------------------------------------
LOCAL_TRAN_ID          1.21.17
GLOBAL_TRAN_ID         SALES.ACME.COM.55d1c563.1.93.29
STATE                  prepared
MIXED                  no
ADVICE
TRAN_COMMENT           Sales/New Order/Trans_type 10B
FAIL_TIME              31-MAY-91
FORCE_TIME
RETRY_TIME             31-MAY-91
OS_USER                SWILLIAMS
OS_TERMINAL            TWA139:
HOST                   system1
DB_USER                SWILLIAMS
COMMIT#
```

グローバル・トランザクション ID の判断

グローバル・トランザクション ID は、分散トランザクションのすべてのノードで同一である共通のトランザクション ID です。次のような書式で記録されています。

global_database_name.hhhhhhhh.local_transaction_id

各項目の意味は次のとおりです。

- *global_database_name* は、グローバル・コーディネータのデータベース名です。
- *hhhhhhhh* は、グローバル・コーディネータの内部データベース識別子（16 進値）です。
- *local_transaction_id* は、グローバル・コーディネータで割り当てられた、対応するローカル・トランザクション ID です。

グローバル・コーディネータでは、グローバル・トランザクション ID の最後の部分とローカル・トランザクション ID が一致します。この例ではこれらの番号が一致しないので、warehouse はグローバル・コーディネータでないことがわかります。

```
LOCAL_TRAN_ID          1.21.17
GLOBAL_TRAN_ID         ... 1.93.29
```

トランザクションの状態の判断

このノードのトランザクションは、準備完了状態です。

```
STATE          prepared
```

したがって、warehouse は、自身のコーディネータからコミット要求またはロールバック要求が送られてくるのを待っています。

コメントまたはアドバイスの確認

トランザクションのコメントまたはアドバイスに、このトランザクションに関する情報が含まれている可能性があります。トランザクションに関する情報が含まれていれば、そのコメントを利用します。この例では、トランザクションのコメントに開始元とトランザクション・タイプが含まれています。

```
TRAN_COMMENT           Sales/New Order/Trans_type 10B
```

また、SET TRANSACTION ... NAME 文によって、トランザクションに関する情報がトランザクション名として提供されている可能性もあります。

この情報から、トランザクションのローカル部分をコミットすべきか、またはロールバックすべきかを決める際に役立つ情報を得ることができます。インダウト・トランザクションに有益なコメントが付けられていない場合は、その他の管理作業を実行し、セッション・ツリーをトレースして、トランザクションをすでに解決しているノードを探す必要があります。

手順 3: ローカル・ノードでの DBA_2PC_NEIGHBORS の問合せ

この手順の目的は、セッション・ツリーを探索してコーディネータを見つけ、最終的にグローバル・コーディネータに到達することです。それと同時に、トランザクションをすでに解決しているコーディネータが見つかる場合もあります。トランザクションをすでに解決しているコーディネータが見つからない場合は、最終的にコミット・ポイント・サイトを探し出します。コミット・ポイント・サイトでは、常にインダウト・トランザクションが解決されています。セッション・ツリーをトレースするには、各ノードの DBA_2PC_NEIGHBORS ビューを問い合せます。

この例では、warehouse データベースでこのビューを問い合せます。

```
CONNECT SYS/password@warehouse.acme.com
SELECT * FROM DBA_2PC_NEIGHBORS
  WHERE LOCAL_TRAN_ID = '1.21.17'
  ORDER BY SESS#, IN_OUT;

Column Name            Value
---------------------- -------------------------------------
LOCAL_TRAN_ID          1.21.17
IN_OUT                 in
DATABASE               SALES.ACME.COM
DBUSER_OWNER           SWILLIAMS
INTERFACE              N
DBID                   000003F4
SESS#                  1
BRANCH                 0100
```

データベースのロールとデータベース・リンク情報の取得

DBA_2PC_NEIGHBORS ビューは、インダウト・トランザクションに対応付けられた接続に関する情報を提供します。情報は、接続が**受信**（IN_OUT = in）か**送信**（IN_OUT = out）かによって異なります。

IN_OUT	意味	DATABASE	DBUSER_OWNER
in	このノードは別のノードのサーバーです。	このノードに接続しているクライアント・データベースの名前がリストされます。	インダウト・トランザクションに対応するデータベース・リンク接続のローカル・アカウントがリストされます。
out	このノードは他のサーバーのクライアントです。	リモート・ノードに接続しているデータベース・リンクの名前がリストされます。	インダウト・トランザクションのデータベース・リンクの所有者がリストされます。

この例では、IN_OUT 列によって、warehouse データベースが sales クライアント（DATABASE 列に示されている）のサーバーであることがわかります。

```
IN_OUT                  in
DATABASE                SALES.ACME.COM
```

warehouse への接続は、swilliams アカウント（DBUSER_OWNER 列に示されている）からのデータベース・リンクを介して確立されています。

```
DBUSER_OWNER            SWILLIAMS
```

コミット・ポイント・サイトの判別

この他、INTERFACE 列によって、ローカル・ノードまたは下位ノードがコミット・ポイント・サイトなのかどうかがわかります。

```
INTERFACE               N
```

INTERFACE 列が示すように、warehouse もその子もコミット・ポイント・サイトではありません。

手順 4: 全ノードでのデータ・ディクショナリ・ビューの問合せ

この時点で、各設置ノードの管理者に連絡を取り、グローバル・トランザクション ID を使用して手順 2 および 3 を繰り返すように各管理者に依頼できます。

注意： これらのノードに別のネットワークを介して直接接続できる場合は、手順 2 および 3 を独力で実行できます。

たとえば、sales および hq で手順 2 および 3 を実行すると、次の結果が返されます。

sales でのペンディング・トランザクションの状態の確認

この段階では、sales の管理者が DBA_2PC_PENDING データ・ディクショナリ・ビューを問い合せます。

```
SQL> CONNECT SYS/password@sales.acme.com
SQL> SELECT * FROM DBA_2PC_PENDING
   > WHERE GLOBAL_TRAN_ID = 'SALES.ACME.COM.55d1c563.1.93.29';

Column Name            Value
---------------------- -------------------------------------
LOCAL_TRAN_ID          1.93.29
GLOBAL_TRAN_ID         SALES.ACME.COM.55d1c563.1.93.29
STATE                  prepared
MIXED                  no
```

```
ADVICE
TRAN_COMMENT            Sales/New Order/Trans_type 10B
FAIL_TIME               31-MAY-91
FORCE_TIME
RETRY_TIME              31-MAY-91
OS_USER                 SWILLIAMS
OS_TERMINAL             TWA139:
HOST                    system1
DB_USER                 SWILLIAMS
COMMIT#
```

sales でのコーディネータとコミット・ポイント・サイトの判別

次に、sales の管理者は DBA_2PC_NEIGHBORS を問い合せて、グローバル・コーディネータ、ローカル・コーディネータおよびコミット・ポイント・サイトを判別します。

```
SELECT * FROM DBA_2PC_NEIGHBORS
   WHERE GLOBAL_TRAN_ID = 'SALES.ACME.COM.55d1c563.1.93.29'
   ORDER BY SESS#, IN_OUT;
```

この問合せは、次の 3 行を返します。

- warehouse への接続
- hq への接続
- ユーザーが確立した接続

warehouse 接続の行に対応する情報を、書式を変えて示すと次のようになります。

```
Column Name             Value
----------------------- -------------------------------------
LOCAL_TRAN_ID           1.93.29
IN_OUT                  OUT
DATABASE                WAREHOUSE.ACME.COM
DBUSER_OWNER            SWILLIAMS
INTERFACE               N
DBID                    55d1c563
SESS#                   1
BRANCH                  1
```

hq 接続の行に対応する情報を、書式を変えて示すと次のようになります。

```
Column Name            Value
---------------------- -------------------------------------
LOCAL_TRAN_ID          1.93.29
IN_OUT                 OUT
DATABASE               HQ.ACME.COM
DBUSER_OWNER           ALLEN
INTERFACE              C
DBID                   00000390
SESS#                  1
BRANCH                 1
```

前の問合せから得られた情報によって、次のことがわかります。

- sales のローカル・トランザクション ID とグローバル・トランザクション ID が一致しているので、sales はグローバル・コーディネータです。
- このノードからは送信接続が 2 つ確立されていますが、受信接続は確立されていません。したがって、sales は別のノードのサーバーではありません。
- コミット・ポイント・サイトは、hq またはそのサーバーの 1 つです。

HQ でのペンディング・トランザクションの状態の確認

この段階では、hq の管理者が DBA_2PC_PENDING データ・ディクショナリ・ビューを問い合せます。

```
SELECT * FROM DBA_2PC_PENDING@hq.acme.com
   WHERE GLOBAL_TRAN_ID = 'SALES.ACME.COM.55d1c563.1.93.29';

Column Name            Value
---------------------- -------------------------------------
LOCAL_TRAN_ID          1.45.13
GLOBAL_TRAN_ID         SALES.ACME.COM.55d1c563.1.93.29
STATE                  COMMIT
MIXED                  NO
ACTION
TRAN_COMMENT           Sales/New Order/Trans_type 10B
FAIL_TIME              31-MAY-91
FORCE_TIME
RETRY_TIME             31-MAY-91
OS_USER                SWILLIAMS
OS_TERMINAL            TWA139:
HOST                   SYSTEM1
DB_USER                SWILLIAMS
COMMIT#                129314
```

この時点で、トランザクションを解決しているノードが見つかりました。ビューが示しているように、このノードではコミットが完了しており、コミット ID 番号が割り当てられています。

```
STATE                   COMMIT
COMMIT#                 129314
```

したがって、自分のローカル・データベースでインダウト・トランザクションを強制的にコミットできます。この調査結果を他の管理者に伝えれば、他の場所での解決にも役立つ可能性があります。

手順 5: インダウト・トランザクションのコミット

sales データベースの管理者と連絡を取り、グローバル ID を使用してインダウト・トランザクションを手動でコミットしてもらうように依頼します。

```
SQL> CONNECT SYS/password@sales.acme.com
SQL> COMMIT FORCE 'SALES.ACME.COM.55d1c563.1.93.29';
```

同時に、warehouse データベースの管理者として、グローバル ID を使用してインダウト・トランザクションを手動でコミットします。

```
SQL> CONNECT SYS/password@warehouse.acme.com
SQL> COMMIT FORCE 'SALES.ACME.COM.55d1c563.1.93.29';
```

手順 6: DBA_2PC_PENDING を使用した MIXED 結果のチェック

トランザクションを手動で強制的にコミットまたはロールバックした後も、ペンディング・トランザクション表の対応する行はそのまま残っています。トランザクションの状態は、トランザクションをどのように強制完了したかに応じて変わります。

Oracle データベースには必ず**ペンディング・トランザクション表**があります。これは、2 フェーズ・コミットの各フェーズの進行に従って分散トランザクションに関する情報を格納する特別な表です。データベースのペンディング・トランザクション表には、DBA_2PC_PENDING データ・ディクショナリ・ビューを介して問い合せることができます（詳細は表 32-1 を参照）。

また、ペンディング・トランザクション表で特に重要なものとして、DBA_2PC_PENDING.MIXED に示される MIXED 結果フラグがあります。ペンディング・トランザクションを強制的にコミットまたはロールバックする場合は、選択を誤る可能性があります。たとえば、ローカル管理者がトランザクションをロールバックしたにもかかわらず、他のノードではそのトランザクションをコミットしてしまうといったことが考えられます。このような誤った判断は自動的に検出され、対応するペンディング・トランザクションのレコードに損傷フラグが設定されます（MIXED=yes）。

RECO バックグラウンド・プロセスは、ペンディング・トランザクション表の情報を使用して、インダウト・トランザクションの状態を最終決定します。また、管理者がペンディング・トランザクション表の情報を使用して、保留中の分散トランザクションの自動リカバリ手順を手動で上書きすることもできます。

RECO によって自動的に解決されたトランザクションは、すべてペンディング・トランザクション表から削除されます。また、管理者によって正しく解決されたインダウト・トランザクションに関する情報も、RECO が通信を再確立したときにチェックされて、すべてペンディング・トランザクション表から自動的に削除されます。ただし、管理者が解決したことによってノード間にまたがって MIXED の結果が生じた行は、`DBMS_TRANSACTIONS.PURGE_MIXED` を使用して手動で削除しないかぎり、関係しているすべてのノードのペンディング・トランザクション表にそのまま残ります。

ロックによるデータ・アクセスの障害

SQL 文を発行すると、Oracle は文を正常に実行するために必要なリソースのロックを試みます。しかし、要求されたデータが、コミットされていない他のトランザクションの文によって現在保持されていて、長時間ロックされたままになっていると、タイムアウトが発生します。

データ・アクセスの障害に関しては、次の事例について考慮してください。

- トランザクションのタイムアウト
- インダウト・トランザクションによるロック

トランザクションのタイムアウト

リモート・データベースのロックを必要とする DML 文は、要求したデータのロックを別のトランザクションが所有している場合にブロックされる可能性があります。このようなロックによって SQL 文の要求がブロックされ続けると、次の一連のイベントが発生します。

1. タイムアウトが発生します。
2. 文がロールバックされます。
3. ユーザーに次のエラー・メッセージが返されます。

   ```
   ORA-02049: タイムアウト : 分散トランザクションがロックを待機しています。
   ```

トランザクションによってデータは変更されていないので、タイムアウトの結果として必要な処理はありません。アプリケーションでは、デッドロックが発生したものとして処理を継続してください。文を実行したユーザーは、後で同じ文の再実行を試みることができます。それでもロックが続く場合には、ユーザーは管理者に連絡して問題を報告してください。

インダウト・トランザクションによるロック

ローカル・データベースのロックを必要とする問合せまたは DML 文は、インダウト分散トランザクションによるリソースのロックのために無期限にブロックされる可能性があります。この場合は、次のエラー・メッセージが発行されます。

```
ORA-01591: インダウト分散トランザクション ID がロックを保持しています。
```

この場合は、SQL 文がただちにロールバックされます。文を実行したユーザーは、後で同じ文の再実行を試みることができます。それでもロックが続く場合には、ユーザーは管理者に連絡して問題を報告してください。このとき、インダウト分散トランザクションの ID も併せて報告してください。

2 フェーズ・コミットの重要な部分の処理中に障害が発生する確率は低いため、このような状況が起こる可能性はほとんどありません。仮にこのような障害が発生した場合でも、ネットワーク障害やシステム障害から迅速にリカバリできれば、手動で介入しなくても問題は自動的に解決されます。したがって、この種の問題は、ユーザーや DBA に検出される前に解決されるのが普通です。

分散トランザクション障害のシミュレーション

分散トランザクションの障害は、次のような理由で強制的に発生させることができます。

- RECO がトランザクションのローカル部分を自動的に解決する様子を観察するため
- インダウト分散トランザクションの手動解決の演習を行い、その結果を観察するため

Oracle インスタンスの RECO バックグラウンド・プロセスは、分散トランザクションに伴う障害を自動的に解決します。ノードの RECO バックグラウンド・プロセスは、時間間隔を指数関数的に広げながら、インダウト分散トランザクションのローカル部分のリカバリを試みます。

RECO は、障害を起こしたトランザクションに関係している他のノードに対して、既存の接続を使用するか、または新しい接続を確立できます。接続が確立されると、RECO はすべてのインダウト・トランザクションを自動的に解決します。各データベースのペンディング・トランザクション表内にインダウト・トランザクションに対応している行があれば、自動的に削除されます。

`ENABLE/DISABLE DISTRIBUTED RECOVERY` オプションを指定して `ALTER SYSTEM` 文を使用すると、RECO を使用可能または使用禁止にできます。たとえば、RECO を一時的に使用禁止にして 2 フェーズ・コミットの障害を強制的に発生させ、インダウト・トランザクションを手動で解決できます。

次の文は、RECO を使用禁止にします。

```
ALTER SYSTEM DISABLE DISTRIBUTED RECOVERY;
```

一方、次の文は RECO を使用可能にし、インダウト・トランザクションが自動的に解決されるようにします。

```
ALTER SYSTEM ENABLE DISTRIBUTED RECOVERY;
```

注意： シングル・プロセス・インスタンス（MS-DOS が稼働している PC など）では、インスタンス以外のバックグラウンド・プロセスは稼働しておらず、したがって RECO プロセスはありません。そのため、分散システムに参加するシングル・プロセス・インスタンスを起動したときは、上の文を使用して手動で分散リカバリを使用可能にする必要があります。

関連項目： シングル・プロセス・インスタンスにおける分散トランザクション・リカバリの詳細は、オペレーティング・システム固有の Oracle マニュアルを参照してください。

読込み一貫性の管理

Oracle の分散読込み一貫性の実装には、重要な制限事項があります。この問題は、各システムが独自の SCN を持っていることに起因します。SCN は、データベースの内部タイムスタンプとして表示できます。Oracle データベース・サーバーは、SCN を使用して、問合せにどのバージョンのデータを返せばよいかを判断します。

分散トランザクションの SCN は、リモート SQL 文の最後と、各トランザクションの最初および最後で同期化されます。2 つのノード間の通信量が多く、特に分散更新がある場合には、この同期化が頻繁に実行されます。しかし、分散システムの SCN の絶対的な同期を保つための実用的な手段は存在しません。つまり、あるノードの SCN が別のノードの SCN と比べて幾分古くなるというような状況が常に存在します。

このような SCN の食い違いから、実行した問合せがわずかに古いスナップショットを使用する可能性があります。その問合せ結果には、リモート・データベースに対する最新の変更が含まれていません。そのため、問合せを実行したときに、読込み一貫性に従ってデータが取得されているにもかかわらず、そのデータが古い場合があります。そのような問合せによって取得したデータは、すべて古い SCN に基づいています。したがって、ローカルに実行した更新トランザクションによってリモート・ノードの 2 つの表が更新された場合、次回のリモート・アクセスで両方の表から選択したデータには、更新前のデータが含まれることになります。

SCN が食い違っているために起こりうる結果の 1 つとして、`SELECT` 文が 2 つ連続している場合に、それら 2 つの文の間で DML をまったく実行していなくても各文で異なるデータが取得されることがあります。たとえば、`UPDATE` 文を発行して、リモート・データベースで更新をコミットしたとします。このリモート表に基づくビューに対して `SELECT` 文を発行すると、ビューには更新された行が表示されません。次に `SELECT` 文を発行するときは、更新された行が表示されます。

次の手法を使用すると、問合せの直前に 2 つのマシンの SCN が必ず同期化されます。

- SCN はリモート問合せの最後に同期化されるので、各リモート問合せの前に、同じサイトで `SELECT * FROM DUAL@REMOTE` というようなダミーのリモート問合せを実行します。
- SCN はすべてのリモート・トランザクションの最初に同期化されるので、リモート問合せを発行する前に、現行トランザクションをコミットまたはロールバックします。

索引

数字

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

あ

い

え

お

か

き

く

け

こ

さ

し

す

せ

た

ち

つ

て

と

な

に

ね

は

ひ

ふ

へ

ほ

み

め

や

ゆ

よ

り

れ

ろ

わ