

Oracle8i

管理者ガイド

リリース 8.1

ORACLE™

Oracle8i 管理者ガイド リリース 8.1

部品番号 : A62758-1

第 1 版 1999 年 5 月 (第 1 刷)

原本名 : Oracle8i Administrator's Guide, Release 8.1.5

原本部品番号 : A67772-01

原本著者 : Joyce Fee

原本協力者: Alex Tsukerman, Andre Kruglikov, Ann Rhee, Ashwini Surpur, Bhaskar Himatsingka, Harvey Eneman, Jags Srinivasan, Lois Price, Robert Jenkins, Sophia Yeung, Vinay Srihari, Wei Huang, Jonathan Klein, Mike Hartstein, Bill Lee, Diana Lorentz, Lance Ashdown, Phil Locke, Ekrem Soylemez, Connie Dialaris, Steven Wertheimer, Val Kane, Mary Rhodes, Archana Kalra, Nina Lewis

Copyright © 1996, 1999, Oracle Corporation. All rights reserved.

Printed in Japan.

制限付権利の説明

プログラムの使用、複製または開示は、オラクル社との契約に記された制約条件に従うものとします。著作権、特許権およびその他の知的財産権に関する法律により保護されています。

当ソフトウェア (プログラム) のリバース・エンジニアリングは禁止されています。

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

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

危険な用途への使用について

オラクル社製品は、原子力、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションを用途として開発されていません。オラクル社製品を上述のようなアプリケーションに使用することについての安全確保は、顧客各位の責任と費用により行ってください。万一かかる用途での使用によりクレームや損害が発生いたしましても、日本オラクル株式会社と開発元である Oracle Corporation (米国オラクル) およびその関連会社は一切責任を負いかねます。当プログラムを米国国防総省の米国政府機関に提供する際には、『Restricted Rights』と共に提供してください。この場合次の Legend が適用されます。

Restricted Rights Legend

Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication and disclosure of the Programs shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-14, Rights in Data -- General, including Alternate III (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

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

目次

はじめに	xxi
------------	-----

第 I 部 基本データベース管理

1 Oracle データベース管理者

Oracle ユーザーのタイプ	1-2
データベース管理者	1-2
セキュリティ管理者	1-3
アプリケーション開発者	1-3
アプリケーション管理者	1-3
データベース・ユーザー	1-3
ネットワーク管理者	1-3
データベース管理者のセキュリティと権限	1-4
データベース管理者のオペレーティング・システム・アカウント	1-4
データベース管理者のユーザー名	1-4
DBA ロール	1-5
データベース管理者の認証	1-5
認証方法の選択	1-6
オペレーティング・システム認証の使用	1-7
OSOPER と OSDBA	1-7
認証パスワード・ファイルの使用	1-8
パスワード・ファイル管理	1-9
ORAPWD の使用	1-9
REMOTE_LOGIN_PASSWORDFILE の設定	1-11
パスワード・ファイルへのユーザーの追加	1-11

管理者権限での接続	1-13
パスワード・ファイルのメンテナンス	1-14
データベース管理者ユーティリティ	1-16
SQL*Loader	1-16
エクスポートとインポート	1-16
データベース管理者の優先順位	1-17
ステップ 1: Oracle ソフトウェアのインストール	1-17
ステップ 2: データベース・サーバー・ハードウェアの評価	1-18
ステップ 3: データベースの計画	1-18
ステップ 4: データベースの作成とオープン	1-19
ステップ 5: データベース設計のインプリメント	1-19
ステップ 6: データベースのバックアップ	1-19
ステップ 7: システム・ユーザーの登録	1-19
ステップ 8: データベースのパフォーマンス調整	1-20
Oracle ソフトウェア・リリースの識別	1-20
リリース番号の形式	1-20
他の Oracle ソフトウェアのバージョン	1-21
現行のリリース番号のチェック	1-21

2 Oracle データベースの作成

データベースを作成する前の考慮点	2-2
作成の前提条件	2-3
初期データベースの使用	2-3
旧リリースのデータベースの移行	2-3
Oracle データベースの作成	2-3
Oracle データベースを作成する手順	2-4
データベースの作成例	2-7
データベース作成の問題解決	2-8
データベースの削除	2-8
パラメータ	2-8
DB_NAME および DB_DOMAIN	2-9
CONTROL_FILES	2-9
DB_BLOCK_SIZE	2-10
DB_BLOCK_BUFFERS	2-11
PROCESSES	2-11

ROLLBACK_SEGMENTS	2-11
ライセンスに関するパラメータ	2-12
LICENSE_MAX_SESSIONS と LICENSE_SESSIONS_WARNING	2-12
LICENSE_MAX_USERS	2-13
データベースを作成した後の考慮点	2-13
初期のチューニング・ガイドライン	2-14
ロールバック・セグメントの割当て	2-14
DB_BLOCK_LRU_LATCHES の数値の選択	2-14
I/O の分散	2-15

3 起動と停止

データベースの起動	3-2
インスタンス起動の準備	3-2
インスタンスの起動方法	3-3
データベースの可用性の変更	3-7
インスタンスにデータベースをマウントする方法	3-7
クローズされたデータベースをオープンする方法	3-7
データベースを読取り専用モードでオープンする方法	3-8
オープン状態のデータベースへのアクセスを制限する方法	3-8
データベースの停止方法	3-9
NORMAL オプションによる停止	3-10
IMMEDIATE オプションによる停止	3-11
TRANSACTIONAL オプションによる停止	3-11
ABORT オプションによる停止	3-12
データベースを中断して再開する方法	3-12
パラメータ・ファイルの使用	3-13
サンプル・パラメータ・ファイル	3-14
パラメータ・ファイルの数	3-14
分散環境におけるパラメータ・ファイルの位置	3-14

第 II 部 Oracle Server の構成

4 Oracle プロセスの管理

サーバー・プロセスのセットアップ	4-2
専用サーバー・プロセスに接続する場合	4-2

マルチスレッド・サーバー・アーキテクチャ用の Oracle の構成	4-3
MTS_DISPATCHERS 初期ディスパッチャ数の設定 (必須)	4-5
サーバー・プロセスの変更	4-6
共有サーバー・プロセスの最小数の変更	4-6
ディスパッチャ・プロセスの追加と削除	4-7
Oracle プロセスの追跡	4-7
Oracle インスタンスのプロセスの監視	4-8
トレース・ファイル、ALERT ファイル、バックグラウンド・プロセス	4-10
チェックポイント・プロセスの起動	4-12
パラレル問合せオプションのプロセスの管理	4-12
問合せサーバーの管理	4-12
問合せサーバー・プロセスの数の変化	4-13
外部プロシージャのプロセスの管理	4-13
セッションの停止	4-15
停止するセッションの識別	4-15
アクティブなセッションの停止	4-16
アクティブでないセッションの停止	4-16

5 制御ファイルの管理

制御ファイルのガイドライン	5-2
制御ファイルの命名	5-2
異なるディスク上での制御ファイルの多重化	5-2
制御ファイルの適切な配置	5-3
制御ファイルのサイズ管理	5-3
制御ファイルの作成	5-3
初期制御ファイルの作成	5-4
制御ファイルの追加コピーの作成と制御ファイルの改名 / 再配置	5-4
新しい制御ファイル	5-5
新しい制御ファイルの作成	5-6
制御ファイルの作成後の問題解決	5-8
欠落したファイルや余分なファイルのチェック	5-8
CREATE CONTROLFILE でのエラーの処理	5-9
制御ファイルの削除	5-9

6 オンライン REDO ログの管理

オンライン REDO ログ	6-2
REDO スレッド	6-2
オンライン REDO ログの内容	6-2
Oracle によるオンライン REDO ログの書込み	6-3
オンライン REDO ログの計画	6-5
オンライン REDO ログ・ファイルの多重化	6-5
オンライン REDO ログ・メンバーを別々のディスクに配置	6-9
オンライン REDO ログ・メンバーのサイズ設定	6-9
適切なオンライン REDO ログ・ファイル数の選択	6-9
オンライン REDO ログ・グループおよびメンバーの作成	6-11
オンライン REDO ログ・グループの作成	6-11
オンライン REDO ログ・メンバーの作成	6-11
オンライン REDO ログ・メンバーの改名および再配置	6-12
オンライン REDO ログ・グループおよびメンバーの削除	6-14
ログ・グループの削除	6-14
オンライン REDO ログ・メンバーの削除	6-15
ログ・スイッチの強制	6-16
REDO ログ・ファイル内のブロックの検証	6-16
オンライン REDO ログ・ファイルの初期化	6-17
制限	6-17
オンライン REDO ログに関する情報のリスト	6-18

7 アーカイブ済み REDO ログの管理

アーカイブ済み REDO ログ	7-2
NOARCHIVELOG モードと ARCHIVELOG モードの選択	7-4
NOARCHIVELOG モードによるデータベースの実行	7-4
ARCHIVELOG モードによるデータベースの実行	7-4
アーカイブの切替え	7-6
初期データベース・アーカイブ・モードの設定	7-6
データベース・アーカイブ・モードの変更	7-6
自動アーカイブの使用可能化	7-7
自動アーカイブの使用禁止	7-8
手動アーカイブの実行	7-9
アーカイブ先の指定	7-10

アーカイブ先の指定	7-10
アーカイブ先の状態の理解	7-12
ログ送信モードの指定	7-13
通常送信モード	7-14
スタンバイ送信モード	7-14
アーカイブ先の障害管理	7-15
正常なアーカイブ先の最小数の指定	7-16
障害アーカイブ先への再アーカイブ	7-18
アーカイブ・パフォーマンスの調整	7-19
複数の ARCn プロセスの指定	7-19
アーカイブ・バッファ・パラメータの設定	7-21
アーカイブ済み REDO ログ情報の表示	7-22
LogMiner を使用したオンラインおよびアーカイブ済み REDO	
ログの分析	7-24
LogMiner の用途	7-24
制限事項	7-25
ディクショナリ・ファイルの作成	7-25
分析する REDO ログ・ファイルの指定	7-27
LogMiner の使用	7-28
LogMiner の使用 : 使用例	7-30

8 ジョブ・キューの管理

SNP バックグラウンド・プロセス	8-2
複数の SNP プロセス	8-3
SNP プロセスの起動	8-3
ジョブ・キューの管理	8-3
DBMS_JOB パッケージ	8-4
ジョブをジョブ・キューに送る方法	8-4
ジョブの実行方法	8-9
ジョブ・キューからのジョブの削除	8-11
ジョブの変更	8-11
中断されたジョブ	8-12
ジョブの強制的な実行	8-14
ジョブの終了	8-14
ジョブ・キューについての情報の表示	8-15

第 III 部 データベース記憶域

9 表領域の管理

表領域を管理するためのガイドライン	9-2
複数の表領域の使用	9-2
表領域の記憶領域パラメータの指定	9-3
ユーザーに表領域割当て制限を割り当てる方法	9-3
表領域の作成	9-3
ローカルに管理される表領域の作成	9-5
一時表領域の作成	9-5
表領域割当ての管理	9-8
表領域に対する記憶領域設定の変更	9-8
空き領域を合わせる方法	9-8
表領域の可用性の変更	9-10
表領域をオンラインにする方法	9-10
表領域をオフラインにする方法	9-10
表領域を読取り専用にする方法	9-12
前提条件	9-13
読取り専用表領域を書込み可能にする方法	9-13
WORM デバイスの読取り専用表領域の作成	9-14
表領域の削除	9-14
DBMS_SPACE_ADMIN パッケージの使用	9-15
使用例 1	9-16
使用例 2	9-16
使用例 3	9-17
使用例 4	9-17
データベース間での表領域のトランスポート	9-17
トランスポートابل・テーブルスペースの概要	9-18
現行の制限事項	9-19
ステップ 1: 自己完結型の表領域セットの選択	9-20
ステップ 2: トランスポートابل・テーブルスペースセットの生成	9-21
ステップ 3: 表領域セットのトランスポート	9-22
ステップ 4: 表領域セットのプラグイン	9-22
オブジェクトの動作	9-24

データ・ウェアハウジングのためのパーティションのトランスポートと 連結例	9-26
構造化データの CD への発行	9-28
複数データベースの同じ表領域の読取り専用でのマウント	9-28
トランスポータブル・テーブルスペースを介した履歴データのアーカイブ	9-29
トランスポータブル・テーブルスペースを使用した TSPITR の実行	9-29
表領域についての情報の表示	9-29

10 データ・ファイルの管理

データ・ファイルを管理するためのガイドライン	10-2
データ・ファイル数の判断	10-2
データ・ファイルのサイズ設定	10-4
適切なデータ・ファイルの配置	10-4
REDO ログ・ファイルと分離したデータ・ファイルの格納	10-4
データ・ファイルの作成および表領域への追加	10-5
データ・ファイルのサイズを変更	10-5
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	10-5
手動によるデータ・ファイルのサイズ変更	10-6
データ・ファイルの可用性の変更	10-7
ARCHIVELOG モードでデータ・ファイルをオンラインにする方法	10-8
NOARCHIVELOG モードでデータ・ファイルをオフラインにする方法	10-8
データ・ファイルの改名および再配置	10-9
単一の表領域のデータ・ファイルの改名および再配置	10-9
複数の表領域のデータ・ファイルの改名および再配置	10-10
データ・ファイル内のデータ・ブロックの検証	10-12
データ・ファイルの情報の表示	10-13

11 データベース・リソース・マネージャの使用

概要	11-2
データベース・リソース・マネージャ・パッケージの使用	11-3
DBMS_RESOURCE_MANAGER パッケージの使用	11-3
DBMS_RESOURCE_MANAGER_PRIVS パッケージ	11-9
DBMS_SESSION パッケージを使用したユーザーのリソース・コンシューマ・グループの変更	11-10
データベース・リソース・マネージャのビュー	11-12

12 スキーマ・オブジェクトを管理するためのガイドライン

データ・ブロックの領域管理	12-2
PCTFREE パラメータ	12-2
PCTUSED パラメータ	12-4
関連する PCTUSED と PCTFREE の値の選択	12-6
記憶領域パラメータの設定	12-7
指定可能な記憶領域パラメータ	12-7
INITTRANS と MAXTRANS の設定	12-9
表領域内のセグメントのデフォルト記憶領域パラメータの設定	12-10
データ・セグメントの記憶領域パラメータの設定	12-10
索引セグメントの記憶領域パラメータの設定	12-10
LOB セグメントの記憶領域パラメータの設定	12-10
記憶領域パラメータの値の変更	12-11
記憶領域パラメータの優先順位の理解	12-11
領域の割当て解除	12-13
高水位標の表示	12-13
領域割当て解除文の発行	12-13
データ型の使用領域の理解	12-16
Oracle のデータ型のまとめ	12-19

13 パーティション表と索引の管理

パーティション表と索引	13-2
パーティション化の方法	13-2
レンジ・パーティション化方法の使用	13-3
ハッシュ・パーティション化方法の使用	13-3
コンポジット・パーティション化方法の使用	13-5
パーティションの作成	13-8
パーティションのメンテナンス	13-9
パーティションの移動	13-10
パーティションの追加	13-11
パーティションの削除	13-11
パーティションを合わせる方法	13-14
パーティションのデフォルト属性の変更	13-14
パーティションの切捨て	13-14
パーティションの分割	13-16

パーティションのマージ	13-17
表パーティションの交換	13-17
索引パーティションの再構築	13-19
履歴表での時間枠の移動	13-19
複数ステップのメンテナンス操作中のアプリケーション静止	13-20

14 表の管理

表を管理するためのガイドライン	14-2
作成前の表の設計	14-2
データ・ブロック領域の使用方法の指定	14-2
トランザクション・エントリ・パラメータの指定	14-3
各表の位置の指定	14-3
表作成の並行化	14-3
回復不能表作成に関する考慮事項	14-4
表のサイズの見積りと記憶領域パラメータの設定	14-4
大規模な表の計画	14-5
表の制限	14-5
表の作成	14-9
表の変更	14-10
表の記憶領域の手動割当て	14-11
表の削除	14-11
列の削除	14-12
索引構成表	14-13
索引構成表とは何か	14-13
索引構成表の作成	14-15
索引構成表のメンテナンス	14-18
索引構成表の分析	14-20
索引構成表での ORDER BY 句の使用	14-20
索引構成表の標準的な表への変換	14-21

15 ビュー、順序およびシノニムの管理

ビューの管理	15-2
ビューの作成	15-2
結合ビューの変更	15-4
ビューの置換	15-8

ビューの削除	15-9
順序の管理	15-9
順序の作成	15-9
順序の変更	15-10
順序に影響を及ぼす初期化パラメータ	15-10
順序の削除	15-10
シノニムの管理	15-11
シノニムの作成	15-11
シノニムの削除	15-11

16 索引の管理

索引を管理するためのガイドライン	16-2
表データ挿入後の索引の作成	16-3
表あたりの索引の数の制限	16-3
トランザクション・エントリ・パラメータの指定	16-3
索引ブロックの領域使用の指定	16-4
各索引の表領域の指定	16-4
索引作成の平行化	16-4
NOLOGGING を指定した索引作成に関する考慮事項	16-5
索引サイズの見積りと記憶領域パラメータの設定	16-5
制約を使用禁止または削除する前の検討	16-7
索引の作成	16-7
制約に対応付けられる索引の作成	16-7
索引の明示的な作成	16-8
索引のオンラインでの作成	16-8
ファンクション・ベース索引の作成	16-9
既存の索引の再作成	16-11
キー圧縮型索引の作成	16-11
索引の変更	16-12
索引の領域使用を監視	16-13
索引の削除	16-13

17 クラスタの管理

クラスタを管理するためのガイドライン	17-2
クラスタに対する適切な表の選択	17-4

クラスタ・キーに対する適切な列の選択	17-4
データ・ブロック領域使用の指定	17-5
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	17-5
各クラスタとクラスタ索引の位置の指定	17-5
クラスタ・サイズの見積りおよび記憶領域パラメータの設定	17-6
クラスタの作成	17-6
クラスタ化された表の作成	17-7
クラスタ索引の作成	17-7
クラスタの変更	17-8
クラスタ表とクラスタ索引の変更	17-9
クラスタの削除	17-9
クラスタ化した表の削除	17-10
クラスタ索引の削除	17-10

18 ハッシュ・クラスタの管理

ハッシュ・クラスタを管理するためのガイドライン	18-2
ハッシングの長所	18-2
ハッシングの短所	18-3
ハッシュ・クラスタが必要とするサイズの見積りおよび記憶領域パラメータの設定	18-4
ハッシュ・クラスタの作成	18-4
ハッシュ・クラスタ内の領域使用の制御	18-5
ハッシュ・クラスタの変更	18-8
ハッシュ・クラスタの削除	18-9

19 データ・ブロック破損の検出と修復

DBMS_REPAIR パッケージの内容	19-2
ステップ 1: 破損を検出してレポート	19-2
DBMS_REPAIR: check_object および admin_tables プロシージャの使用	19-3
DB_VERIFY: オフライン・データベース・チェックの実行	19-3
ANALYZE: 破損のレポート	19-3
DB_BLOCK_CHECKING: ブロック・チェック初期化パラメータ	19-3
ステップ 2: DBMS_REPAIR を使用してコストと利点を評価	19-4
ステップ 3: オブジェクトの使用可能化	19-5
破損の修復 : fix_corrupt_blocks および skip_corrupt_blocks プロシージャの使用	19-5
破損ブロックをスキップする操作の含意	19-5

ステップ 4: 破損を修復して失われたデータの再構築	19-6
dump_orphan_keys プロシージャを使用したデータの回復	19-6
rebuild_freelists プロシージャを使用した空きリストの修復	19-6
制約および制限事項	19-6
DBMS_REPAIR のプロシージャ	19-7
check_object	19-7
fix_corrupt_blocks	19-8
dump_orphan_keys	19-9
rebuild_freelists	19-10
skip_corrupt_blocks	19-11
admin_tables	19-12
DBMS_REPAIR の例外	19-13

20 スキーマ・オブジェクトの一般的な管理

一度の操作で複数の表やビューを作成	20-2
スキーマ・オブジェクトの改名	20-2
表、索引およびクラスタの分析	20-3
表、索引およびクラスタの統計の使用	20-4
表、索引およびクラスタの妥当性の検査	20-8
表とクラスタの連鎖された行のリスト	20-8
表とクラスタの削除	20-9
トリガーの使用可能および使用禁止	20-11
トリガーを使用可能にする方法	20-11
トリガーを使用禁止にする方法	20-12
整合性制約の管理	20-12
整合性制約の状態	20-13
制約チェックの延期	20-15
対応付けられた索引をもつ制約の管理	20-17
定義するときの整合性制約の設定	20-17
既存の整合性制約を変更する方法	20-19
整合性制約を削除する方法	20-20
制約例外のレポート	20-20
オブジェクトの依存性の管理	20-22
ビューを手動で再コンパイルする方法	20-23
プロシージャとファンクションの手動による再コンパイル	20-23

パッケージを手動で再コンパイルする方法	20-24
オブジェクト名の名前の変換の管理	20-24
データ・ディクショナリの記憶領域パラメータの変更	20-25
データ・ディクショナリの構造	20-26
データ・ディクショナリの記憶領域の変更が必要なエラー	20-27
スキーマ・オブジェクトについての情報を表示	20-28
Oracle ディクショナリ記憶領域パッケージ	20-29
例 1: スキーマ・オブジェクトのタイプ別表示	20-29
例 2: 列情報の表示	20-30
例 3: ビューとシノニムの依存性の表示	20-30
例 4: グローバル・セグメント情報の表示	20-31
例 5: グローバル・エクステント情報の表示	20-31
例 6: データベースの空き領域（エクステント）の表示	20-31
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	20-32

21 ロールバック・セグメントの管理

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	21-2
複数ロールバック・セグメントの使用	21-2
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	21-3
自動的に取得するロールバック・セグメントの指定	21-3
ロールバック・セグメント・サイズの設定	21-4
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	21-5
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	21-5
ロールバック・セグメントに対する記憶位置の設定	21-7
ロールバック・セグメントの作成	21-7
新しいロールバック・セグメントをオンラインにする方法	21-8
ロールバック・セグメントの記憶領域パラメータを指定	21-8
ロールバック・セグメントを作成するときに記憶領域パラメータを設定 する方法 21-8	
ロールバック・セグメントの記憶領域パラメータを変更する方法	21-9
ロールバック・セグメントのフォーマットを変更する方法	21-9
ロールバック・セグメントを手動で縮小する方法	21-10
ロールバック・セグメントのオンライン化とオフライン化	21-10
ロールバック・セグメントをオンラインにする方法	21-11
ロールバック・セグメントをオフラインにする方法	21-12

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 割り当てる方法	21-13
ロールバック・セグメントの削除	21-14
ロールバック・セグメント情報を監視	21-15
ロールバック・セグメント情報を表示する方法	21-15

第 IV 部 データベース・セキュリティ

22 セキュリティ方針の設定

システム・セキュリティ方針	22-2
データベース・ユーザー管理	22-2
ユーザー認証	22-2
オペレーティング・システムのセキュリティ	22-3
データ・セキュリティ方針	22-3
ユーザー・セキュリティ方針	22-4
一般的なユーザー・セキュリティ	22-4
エンド・ユーザーのセキュリティ	22-5
管理者のセキュリティ	22-7
アプリケーションの開発者のセキュリティ	22-8
アプリケーション管理者のセキュリティ	22-10
パスワード管理方針	22-11
アカウント・ロック	22-11
パスワード・エイジングおよび時間切れ	22-12
パスワード履歴	22-13
パスワード複雑度の検証	22-13
監査方針	22-17

23 ユーザーとリソースの管理

セッションとユーザーのライセンス	23-2
同時使用ライセンス	23-2
接続権限	23-3
セッションの最大数の設定	23-4
セッション警告制限の設定	23-4
データベースの稼働中の同時使用制限の変更	23-4
名前付きユーザー制限	23-5

ライセンス制限と現行値の参照	23-6
ユーザー認証	23-7
データベース認証	23-7
外部認証	23-8
企業認証	23-10
Oracle ユーザー	23-10
ユーザーの作成	23-10
ユーザーの変更	23-14
ユーザーの削除	23-15
プロファイルによるリソースの管理	23-16
プロファイルの作成	23-17
プロファイルの割当て	23-18
プロファイルの変更	23-18
複合制限の使用	23-18
プロファイルの削除	23-20
リソース制限を使用可能、使用禁止にする方法	23-20
データベース・ユーザーとプロファイルに関する情報のリスト	23-21
ユーザーとプロファイルに関する情報の記述例	23-22
例	23-24

24 ユーザー権限とロールの管理

ユーザー権限の識別	24-2
システム権限	24-2
オブジェクト権限	24-4
ユーザー・ロールの管理	24-4
ロールの作成	24-4
事前定義済みロール	24-6
ロールの認可	24-7
ロールの削除	24-10
ユーザー権限とロールの付与	24-10
システム権限とロールの付与	24-10
オブジェクト権限とロールの付与	24-11
列への権限の付与	24-12
ユーザー権限とロールの取消し	24-13
システム権限とロールの取消し	24-13

オブジェクト権限とロールの取消し	24-13
権限の取消しによる影響	24-15
ユーザー・グループ PUBLIC に対する付与と取消し	24-16
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てロールを付与	24-17
オペレーティング・システムのロール識別機能の使用	24-18
オペレーティング・システムのロール管理機能の使用	24-19
OS_ROLES=TRUE の場合のロールの付与および取消し	24-19
OS_ROLES=TRUE の場合にロールを使用可能、使用禁止にする方法	24-19
オペレーティング・システムによるロール管理でのネットワーク接続の使用	24-20
権限とロールの情報の記述	24-20
権限およびロール情報の記述例	24-20

25 データベース使用の監査

監査機能のガイドライン	25-2
データベースまたはオペレーティング・システムによる監査	25-2
監査済み情報の管理しやすい状態での維持	25-2
データベース監査証跡ビューの作成および削除	25-4
監査証跡ビューの作成	25-4
監査証跡ビューの削除	25-5
監査証跡情報の管理	25-5
デフォルトで監査されるイベント	25-6
監査オプションの設定	25-7
データベース監査を使用可能、使用禁止にする方法	25-12
監査証跡の成長とサイズを制御する方法	25-13
監査証跡を保護する方法	25-15
データベース監査証跡情報の参照	25-16
アクティブな文監査オプションの記述	25-17
アクティブな権限監査オプションの記述	25-17
特定のオブジェクトのアクティブなオブジェクト監査オプションの記述	25-17
デフォルトのオブジェクト監査オプションの記述	25-18
監査レコードの記述	25-18
AUDIT SESSION オプションの監査レコードの記述	25-19
データベース・トリガーによる監査	25-19

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・システムの円滑な運用を保証し、その使用状況を監視する責任を負うものとみなされます。データベース管理者の責務については、第 1 章で説明します。

注意： このマニュアルは、Oracle8i と Oracle8i Enterprise Edition 製品の特徴と機能を説明しています。Oracle8i と Oracle8i Enterprise Edition の基本機能は同じです。ただし、最新の機能のいくつかは、Enterprise 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、自動の表領域 Point-in-Time 回復 (Recovery Manager を使用して) を実行するには、Enterprise Edition が必要です。

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に精通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・システム環境にも精通している必要があります。

インストレーションと移行に関する情報が必要な読者

管理者は、Oracle Server ソフトウェアのインストールや、既存の Oracle データベースの新しい形式への移行作業（たとえば、バージョン 7 データベースを Oracle8i 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、インストレーションやシステムの移行については説明しません。

主に必要な情報がインストレーションに関する情報である場合は、使用しているオペレーティング・システム固有の Oracle マニュアルを参照してください。

主に必要な情報がデータベースまたはアプリケーションの移行に関する情報である場合は、『Oracle8i 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、管理者だけでなく、Oracle に精通した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効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、『Oracle8i アプリケーション開発者ガイド 基礎編』と、Oracle データベース・アプリケーションを開発するため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。

このマニュアルの使用方法

このマニュアルを読む前に、必ず『Oracle8i 概要』の第 1 章を読んでください。そこに記述された Oracle に関する概念と用語の概要は、このマニュアルの詳細な情報の基礎となるものです。『Oracle8i 概要』の他の章では、Oracle のアーキテクチャと各機能について説明し、各機能の動作について詳しく述べています。

このマニュアルの構成

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 章の「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を説明します。

第 2 章の「Oracle データベースの作成」

データベースの作成について、最も考慮すべき点について説明します。データベースを計画するときに参照してください。

第 3 章の「起動と停止」

データベースの起動、その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 Server の構成

第 4 章の「Oracle プロセスの管理」

専用サーバー・プロセスおよびマルチスレッド・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 5 章の「制御ファイルの管理」

制御ファイルの管理のすべての作業（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 6 章の「オンライン REDO ログの管理」

オンライン REDO ログの管理のすべての作業（オンライン REDO ログ・ファイルの計画、作成、改名、削除および消去）について説明します。

第 7 章の「アーカイブ済み REDO ログの管理」

アーカイブ・モード、アーカイブのチューニングおよび表示方法について説明します。

第 8 章の「ジョブ・キューの管理」

ジョブ・キューを扱う前に参照してください。ジョブ・キューの発行、削除、変更および調整などすべての作業について説明しています。

第 III 部 : データベース記憶領域

第 9 章の「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 10 章の「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・ファイルに関する情報の作成、変更、改名および表示の方法を説明します。

第 11 章の「データベース・リソース・マネージャの使用」

データベース・リソース・マネージャを使用してリソースを割り当てる方法を説明します。

第 12 章の「スキーマ・オブジェクトを管理するためのガイドライン」

記憶領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 13 章の「パーティション表と索引の管理」	パーティション表（および索引）について、およびその作成方法と管理方法を説明します。
第 14 章の「表の管理」	一般的な表の管理のガイドライン、および表の作成、変更、メンテナンス、削除についても説明します。
第 15 章の「ビュー、順序およびシノニムの管理」	ビュー、順序、シノニムの管理のすべての作業について説明します。
第 16 章の「索引の管理」	作成、変更、監視および削除など、索引についての一般的なガイドラインを示します。
第 17 章の「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 18 章の「ハッシュ・クラスタの管理」	ハッシュ・クラスタの変更または削除についての一般的なガイドラインを示します。
第 19 章の「データ・ブロック破損の検出と修復」	DBMS_REPAIR パッケージ内のプロシージャを使用してデータ・ブロックの破損を検出し、訂正する方法を説明します。
第 20 章の「スキーマ・オブジェクトの一般的な管理」	第 12 章で説明したスキーマ管理のうち、表の分析、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など、特定の事項をさらに詳しく説明します。いくつかの例も記載されています。
第 21 章の「ロールバック・セグメントの管理」	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を示します。

第 IV 部：データベース・セキュリティ

第 22 章の「セキュリティ方針の設定」	パスワードの管理に関連した特定の作業だけでなく、システム、データとユーザーのセキュリティ方針などのデータベース・セキュリティのすべての作業について説明します。
第 23 章の「ユーザーとリソースの管理」	セッションとユーザーのライセンス、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作業の特定の例を提供します。
第 24 章の「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ついて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の方法について説明します。
第 25 章の「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。

このマニュアルで使用する表記規則

ここでは、このマニュアルで使用する表記上の規則について説明します。

- 本文
- 構文図および表記法
- コードの例

本文

ここでは、このマニュアルの本文で使用する表記規則について説明します。

大文字

アルファベットの大文字は、コマンドのキーワード、オブジェクト名、パラメータ、ファイル名などを明示するために使用されています。次に例を示します。

「プライベート・ロールバック・セグメントを作る場合、ロールバック・セグメントの名前は、パラメータ・ファイルの ROLLBACK_SEGMENTS パラメータに指定する必要があります。」

構文図および表記法

このマニュアルの構文図と表記法では、SQL コマンドの構文、関数、ヒントおよびその他の要素を示しています。ここでは、それらに基づき、構文図と例の読み方および SQL 文の書き方について説明します。

キーワード

キーワードは、SQL 言語で特別な意味を持つワードです。このマニュアルの構文図では、キーワードを大文字で示します。大文字と小文字のどちらでも使用できる場合以外は、構文図が示すとおり SQL 文にキーワードを入力する必要があります。たとえば、CREATE TABLE 文を開始するには、CREATE TABLE 構文図が示すように CREATE キーワードを使用する必要があります。

パラメータ

パラメータは、構文図でプレースホルダの役割をします。パラメータは小文字で示されます。パラメータは、通常、データベース・オブジェクトの名前、オラクルのデータ型名または式です。構文図にパラメータがある場合、SQL 文では、オブジェクトまたは式の部分に適切な語句を代入します。たとえば、CREATE TABLE 文を記述するには、構文図の *table* パラメータのかわりに、EMP のように作成する表の名前を使用します。(パラメータ名が本文内でイタリックで示されていることに注意してください。)

次のリストでは、このマニュアルの構文図にあるパラメータと文中でその部分に代入される値の例を示します。

パラメータ	説明	例
<i>table</i>	代入値は、パラメータに指定されたタイプのオブジェクト名にする必要があります。	emp
<i>'text'</i>	代入値は、引用符でくくられた文字リテラルにする必要があります。	'Employee Records'
<i>condition</i>	代入値は、TRUE または FALSE を評価する条件にする必要があります。	ename > 'A'
<i>date</i> <i>d</i>	代入値は、日付定数または日付データ型の式にする必要があります。	TO_DATE ('01-Jan-1996', DD-MON-YYYY)
<i>expr</i>	代入値は、どのデータ型の式でも使用できます。	sal + 1000
<i>integer</i>	代入値は整数にする必要があります。	72
<i>rowid</i>	代入値は、データ型 ROWID の式にする必要があります。	00000462.0001.0001
<i>subquery</i>	代入値は、他の SQL 文に含まれる SELECT 文にする必要があります。	SELECT ename FROM emp
<i>statement_name</i> <i>block_name</i>	代入値は、SQL 文または PL/SQL ブロックの識別子にする必要があります。	s1 b1

コードの例

SQL、SQL*Plus の各コマンドや文は、次のようにクーリエ・フォントで示され、本文のパラグラフとは区別して表記されています。

```
INSERT INTO emp (empno, ename) VALUES (1000, 'JFEE');
ALTER TABLESPACE users ADD DATAFILE 'users2.ora' SIZE 50K;
```

例文には、カンマや引用符などの句読点が含まれている場合があります。例文に示した句読点はすべて必須です。例文はすべて、セミコロン (;) で終了します。使用するアプリケーションによっては、文を終了する際にセミコロンやその他の終了記号が必要である場合とそうでない場合があります。

例文では、Oracle SQL におけるキーワードを大文字で明示します。ただし、文を発行するときはキーワードが大 / 小文字区別を行わないことに注意してください。

例文では、例文のコンテキストにだけ適用されたワードを小文字で示します。たとえば、小文字のワードは表、列またはファイルの名前を示す場合があります。

第Ⅰ部

基本データベース管理

Oracle データベース管理者

この章では、Oracle Server を管理する人、つまりデータベース管理者の責任について説明します。

この章のトピックは、次のとおりです。

- Oracle ユーザーのタイプ
- データベース管理者のセキュリティと権限
- データベース管理者の認証
- パスワード・ファイル管理
- データベース管理者ユーティリティ
- データベース管理者の優先順位
- Oracle ソフトウェア・リリースの識別

Oracle ユーザーのタイプ

ユーザーのタイプやその責任は、サイトによって異なることがあります。たとえば、大規模なサイトの場合は、データベース管理者が果たす役割を何人かで分担することがあります。

この項のトピックは、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](#)
- [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](#)
- [ネットワーク管理者](#)

データベース管理者

Oracle データベース・システムは、非常に大規模で多数のユーザーがいるため、システムの管理は何人かの担当者または複数の担当者グループで行う必要があります。このような管理者をデータベース管理者（DBA）と呼びます。どのデータベースでも、管理の義務を果たす担当者が少なくとも 1 名は必要です。

データベース管理者が担当する作業は、次のとおりです。

- Oracle Server とアプリケーション Tools 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、初期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、初期オブジェクト（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い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保証します。
- データベースに対するユーザー・アクセスを制御、監視します。
- データベースのパフォーマンスを監視、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、復元します。

- 技術サポートについてオラクル社に連絡します。

セキュリティ管理者

場合によっては、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、主にユーザーの登録、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に関与します。したがって、サイトごとにセキュリティ管理者がいる場合、データベース管理者はこれらの業務に対する責任はありません。

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、インプリメントします。アプリケーション開発者が担当する作業は、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。
- データベース管理者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調整します。
- 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。

アプリケーション管理者

Oracle では、サイトごとに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。アプリケーション管理者は、特定のアプリケーションの管理上の要求に責任があります。

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。一般ユーザーの責任で行われる作業は、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。
- データの報告書を作成します。

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存在することがあります。ネットワーク管理者は、Net8 などの Oracle ネットワーク製品の管理を担当することがあります。

関連項目：『Oracle8i 分散システム』の「ネットワーク管理」

データベース管理者のセキュリティと権限

Oracle で管理業務を遂行するには、データベースとデータベースの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。データベース管理者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。

この項のトピックは、次のとおりです。

- [データベース管理者のオペレーティング・システム・アカウント](#)
- [データベース管理者のユーザー名](#)
- [DBA ロール](#)

データベース管理者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上の多くの業務を果たすには、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・ユーザーが Oracle ソフトウェアのインストールを実行するなどの場合に比べて、より大きなオペレーティング・システム権限やアクセス権が必要になります。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。

また、Enterprise Manager では、オペレーティング・システム・アカウントまたは ID をなんらかの方法で識別して、オペレーティング・システムによって権限が与えられた Enterprise Manager コマンドを使用できるようにする必要があります。

関連項目：データベース管理者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。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。

データベース管理者のユーザー名

データベースとともに 2 つのユーザー・アカウントが自動的に作成され、DBA ロールを付与されます。このユーザー・アカウントは次の 2 つです。

- SYS (初期パスワード: CHANGE_ON_INSTALL)
- SYSTEM (初期パスワード: MANAGER)

これらの 2 つのユーザー名について、次に説明します。

注意： データ・ディクショナリ表への不当なアクセスを防ぐため、Oracle データベースを作成した直後に、SYS および SYSTEM のユーザー名に対するパスワードを変更する必要があります。

日常的な管理業務を遂行するときに使用する管理者ユーザー名を最低 1 つ追加で作成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 (パスワード CHANGE_ON_INSTALL で識別される) が自動的に登録され、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、スキーマ SYS 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、SYS スキーマ内の表は Oracle でのみ操作されます。つまりユーザーやデータベース管理者は修正してはならず、ユーザー SYS のスキーマ内に表を作成してはなりません。(ただし、必要に応じてデータ・ディクショナリ設定の記憶領域パラメータを変更することはできます)。

ほとんどのデータベース・ユーザーに対し、SYS アカウントによる接続を禁止する必要があります。管理者はこのアカウントを使用してデータベースに接続できますが、その場合も Oracle データベース管理者やマニュアルによって指示されたときだけに限定してください。

SYSTEM

データベースの作成時には、ユーザー SYSTEM (パスワード MANAGER によって識別される) も自動的に作成され、データベースに関するシステム権限がすべて付与されます。

ユーザー名 SYSTEM により、管理情報を表示する追加表とビュー、および Oracle Tools 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

事前定義済みの "DBA" という名前のロールは、あらゆる 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、すべてのデータベース・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り、専任のデータベース管理者だけに付与してください。

データベース管理者の認証

データベース管理者は、データベースの起動や停止などの特殊な操作を実行する必要があります。このような操作は、通常のデータベース・ユーザーは実行すべきでないため、データベース管理者のユーザー名には安全性の高い認証方式が必要です。

この項のトピックは、次のとおりです。

- [認証方法の選択](#)
- [オペレーティング・システム認証の使用](#)
- [OSOPER と OSDBA](#)
- [認証パスワード・ファイルの使用](#)

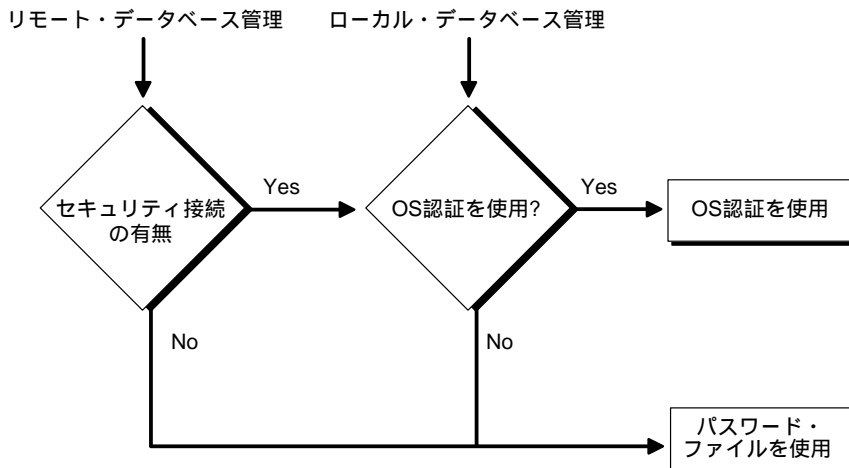
認証方法の選択

次のデータベース管理者認証方法は、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。

- オペレーティング・システム認証
- パスワード・ファイル

データベースが存在する同一の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、オペレーティング・システム認証か、またはデータベース管理者を認証するためのパスワード・ファイルのどちらかを選択できます。[図 1-1](#) は、データベース管理者用に選択できる認証方式を示しています。

図 1-1 データベース管理者オウフクの認証方式



ほとんどのオペレーティング・システムでは、データベース管理者の OS 認証のため、データベース管理者の OS ユーザー名を特別なグループ（UNIX システムでは DBA グループ）に入れたり、その OS ユーザー名に特別な処理を実行する権限を与えたりします。

データベースは、パスワード・ファイルを使用して、管理者権限を付与されているデータベース・ユーザー名を追跡管理します。

関連項目：『Oracle8i 概要』の「ユーザー認証」

オペレーティング・システム認証の使用

この方法を選択すると、データベース管理操作を実行するユーザーをオペレーティング・システムによって認証させることができます。

1. オペレーティング・システムが認証するユーザーを設定します。
2. 初期化パラメータ `REMOTE_LOGIN_PASSWORDFILE` が `NONE` に設定されていることを確認します。これが、このパラメータのデフォルト値です。
3. 認証されたユーザーは、次のどちらかのコマンドを入力すると、ローカル・データベースに接続したり、セキュリティのある接続によってリモート・データベースに接続できます。

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

旧リリースの Oracle で `INTERNAL` として正常に接続できる場合は、ステップ 3 に示す新しい構文を使用することにより、引き続き正常に接続できます。

注意： OS 認証を使用して `SYSOPER` または `SYSDBA` として接続するときは、`SYSOPER` または `SYSDBA` のシステム権限を付与されている必要はありません。オペレーティング・システム・レベルで適切な `OSDBA` または `OSOPER` ロールを付与されているかどうかをサーバーが検証します。

OSOPER と OSDBA

オペレーティング・システム認証を使用するときは、`OSOPER` と `OSDBA` という 2 つの特殊なオペレーティング・システム・ロールによってデータベース管理者のログインが制御されます。

OSOPER	ユーザーは、STARTUP、SHUTDOWN、ALTER DATABASE OPEN/MOUNT、ALTER DATABASE BACKUP、ARCHIVE LOG および RECOVER の実行を許可されます。このロールは RESTRICTED SESSION 権限を持っています。
OSDBA	ADMIN OPTION 付きのすべてのシステム権限と OSOPER ロールを持っています。CREATE DATABASE と時間ベースの回復を許可します。

OSOPER と OSDBA の名前と機能は、使用しているオペレーティング・システムによって異なる可能性があります。

OSOPER ロールと OSDBA ロールは、オペレーティング・システムからのみユーザーに付与されます。これらのロールは、GRANT 文では付与できません。また、これらのロールは取消しも削除もできません。ユーザーが管理者権限でログインするとき、REMOTE_LOGIN_PASSWORDFILE が NONE に設定されていると、Oracle は、オペレーティング・システムと通信して最初に OSDBA を使用可能にしようとし、それが失敗した場合は次に OSOPER を使用可能にしようします。そして両方とも成功しない場合、その接続は失敗に終わります。オペレーティング・システムからこれらの権限を付与する方法は、オペレーティング・システムによって異なります。

リモート・データベース管理の場合は、Net8 マニュアルを参照して、セキュリティ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、Net8 のどのバージョンを使用してもセキュリティ接続を使用することはできません。

関連項目：データベース管理者のオペレーティング・システム認証の詳細は、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。

認証パスワード・ファイルの使用

ユーザーがデータベース管理を実行することをパスワード・ファイルを使用して認証する必要があると判断した場合は、次に示すステップを実行する必要があります。各ステップについては、この章の後の項で詳しく説明します。

1. 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
```

2. REMOTE_LOGIN_PASSWORDFILE 初期化パラメータを EXCLUSIVE に設定します。
3. SQL を使用してパスワード・ファイルにユーザーを追加し、次の例のようにデータベース管理を実行する必要がある各ユーザーに適切な権限を付与します。

```
GRANT SYSDBA TO scott;  
GRANT SYSOPER TO scott;
```

SYSDBA 権限は、OSDBA と同じ操作の実行を許可します。同様に、SYSOPER 権限は OSOPER と同じ操作の実行を許可します。

4. 権限を付与されたユーザーは、次のコマンドを使用してデータベースに接続できます。

```
CONNECT scott/tiger@acct.hq.com AS SYSDBA
```

パスワード・ファイル管理

パスワード・ファイルは、パスワード・ファイル作成ユーティリティ ORAPWD を使用して作成します。特定のオペレーティング・システムでは、このファイルを標準インストール作業の一部として作成できます。

この項のトピックは、次のとおりです。

- [ORAPWD の使用](#)
- [REMOTE_LOGIN_PASSWORDFILE の設定](#)
- [パスワード・ファイルへのユーザーの追加](#)
- [管理者権限での接続](#)
- [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・ユーティリティでパスワード・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

パラメータをまったく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 and INTERNAL (mand),
entries - maximum number of distinct DBAs and OPERs (opt),
There are no spaces around the equal-to (=) character.
```

たとえば、次のコマンドを使用すると、ACCTPWD という名前のパスワード・ファイルが作成され、30 人までのユーザーに権限を与えて、それぞれ異なるパスワードが使用できます。このファイルは、最初はパスワード SECRET で、SYSOPER または SYSDBA として接続するユーザー用に作成されます。

```
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。ファイルには完全な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、ユーザーには読めないファイルです。これは必須パラメータです。

パスワード・ファイルで許されているファイル名のタイプは、オペレーティング・システムにより異なります。一部のプラットフォームでは、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プラットフォームもあります。プラットフォームで許されている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。

Oracle Parallel Server を使用して Oracle の複数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なければなりません。

警告： 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可能性を潜在的に持っています。

PASSWORD

このパラメータは、SYSOPER および SYSDBA 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 コマンドを発行して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。INTERNAL ユーザーは、下位互換性を保つ目的でのみサポートされています。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・ファイルに指定できるエントリの最大数を設定します。これは、データベースに SYSDBA または SYSOPER として接続できる異なるユーザーの最大数に対応します。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きに、このエントリは繰り返し使用されます。このパスワード・ファイルをいずれ EXCLUSIVE にする必要がある場合には、このパラメータは必須です。

警告： この制限を超える必要がある場合は、新しいパスワード・ファイルを作成しなければなりません。必要になるとされる数よりも大きい数を選択するのが最も安全です。

関連項目：パスワード・ファイルの正確な名前、またはオペレーティング・システムにこの名前を指定する環境変数の名前は、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。

REMOTE_LOGIN_PASSWORDFILE の設定

パスワード・ファイルの作成に加えて、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。認識される値を次に示します。

注意： インスタンスまたはデータベースを起動する場合は、必ず Enterprise Manager を使用してください。データベース名とパラメータ・ファイルを指定して、インスタンスの設定を初期化します。Net8 を使用して、リモート・データベースの完全修飾名を指定します。ただし、初期化パラメータ・ファイルと構成ファイルなどの関連ファイルが必ずクライアント・マシン上に必要です。つまり、Enterprise Manager を稼働しているマシン上にパラメータ・ファイルが必要です。

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動作をします。つまり、セキュリティのない接続では権限付きの接続は実現できません。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。EXCLUSIVE ファイルだけに SYSOPER と SYSDBA 以外のユーザー名を登録できます。EXCLUSIVE パスワード・ファイルを使用すると、個々のユーザーにシステム権限 SYSDBA と SYSOPER を付与し、SYSDBA、SYSOPER 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・ファイルによって認識されるユーザーは SYSDBA と SYSOPER のみです。SHARED パスワード・ファイルでは、ユーザーの追加はできません。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 と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 人のデータベース管理者が複数のデータベースを管理する場合に役に立ちます。

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・ファイルの作成後、ただちに REMOTE_LOGIN_PASSWORDFILE ファイルの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。サーバーに 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 か

SHARED である場合は、これらの権限を付与しようとする、エラー・メッセージが表示されます。

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。ユーザーのこの 2 つの権限のうちの最後の 1 つを取り消すと、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. パスワード・ファイルの作成手順に従って操作します。
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/change_on_install 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。必要であれば、データベース管理者自身および適切なユーザーに SYSOPER 権限または SYSDBA 権限を付与します。
6. これで、ユーザーはパスワード・ファイルに追加され、(SYS ではなく) ユーザー名とパスワードを使用して SYSOPER または SYSDBA としてデータベースに接続できます。OS 認証されたユーザーが OS 認証の基準を満たしていれば、パスワード・ファイルを使用することによって OS 認証されたユーザーの接続が妨げられることはありません。

SYSOPER 権限と SYSDBA 権限の付与と取消し

サーバーで EXCLUSIVE パスワード・ファイルを使用している場合は、GRANT コマンドを使用して、次の例に示すようにユーザーに SYSDBA または SYSOPER のシステム権限を付与してください。

```
GRANT SYSDBA TO scott;
```

ユーザーのシステム権限 SYSDBA または SYSOPER を取り消すには、次の例のように REVOKE コマンドを使用します。

```
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 (または INTERNAL) として接続しているユーザーだけが、別のユーザーにシステム権限 SYSDBA または SYSOPER を付与できます。これは、REVOKE にも当てはまります。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の起動後までは使用可能にならないためです。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 24 章の「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・メンバーのリスト

VSPWFILE_USERS ビューは、1 つのデータベースについてどのユーザーにシステム権限 SYSDBA および SYSOPER が付与されているかを判断するために使用します。このビューで表示される列は、次のとおりです。

USERNAME

パスワード・ファイルで認識されるユーザーの名前。

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。

管理者権限での接続

ユーザー名とパスワードを使用して SYSOPER または SYSDBA 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 SYS で接続していることになります。

管理者権限での接続例

たとえば、ユーザー SCOTT が次のコマンドを発行したとします。

```
CONNECT scott/tiger
CREATE TABLE scott_test(name VARCHAR2(20));
```

後で、SCOTT が次のコマンドを発行するとします。

```
CONNECT scott/tiger AS SYSDBA
SELECT * FROM scott_test;
```

この場合、SCOTT_TEST が存在しないというエラーを受け取ります。このエラーは、表は SCOTT スキーマ内に作成されていますが、SCOTT が現在デフォルトで SYS スキーマを参照しているために起こります。

セキュリティのないリモート接続

セキュリティの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、次の条件を満たす必要があります。

- 接続するサーバーにパスワード・ファイルがあること。
- システム権限 SYSOPER または SYSDBA を付与されていること。

- ユーザー名とパスワードを使用して接続すること。

ローカル接続およびセキュリティのあるリモート接続

ローカル接続またはセキュリティのあるリモート接続で、権限を持つユーザーとして Oracle に接続するには、次のどちらかの条件を満たす必要があります。

- 前述のセキュリティのない接続の項で概説した基準を満たす場合は、パスワード・ファイルを使用して接続する必要があります。
- サーバーがパスワード・ファイルを使わない場合、あるいは SYSOPER 権限または SYSDBA 権限を付与されていないため、これらがパスワード・ファイルにない場合は、権限付き接続のために、オペレーティング・システムによってオペレーティング・システム・ユーザー名を認証する必要があります。この認証の形式は、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・システム認証の詳細は、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。

関連項目 : 1-9 ページの「[パスワード・ファイル管理](#)」

パスワード・ファイルのメンテナンス

ここでは、パスワード・ファイルの拡張、再配置および削除と、パスワード・ファイルの状態変更の防止方法について説明します。

パスワード・ファイルのユーザー数の追加

ユーザーにシステム権限 SYSDBA または SYSOPER を付与しようとしたときに、ファイルが満杯 (ORA-1996) というエラーが発行された場合は、よりサイズの大きいパスワード・ファイルを作成し、ユーザーにもう 1 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. VSPWFILE_USERS ビューに問い合せて、どのユーザーが SYSDBA 権限または 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. 1-9 ページの「[ORAPWD の使用](#)」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 パラメータは、必ず十分な大きさの数値に設定してください。
5. 1-11 ページの「[パスワード・ファイルへのユーザーの追加](#)」の指示に従って操作します。

パスワード・ファイルの再配置

パスワード・ファイルを作成した後、希望の位置に再配置できます。パスワード・ファイルを再配置した後、該当する環境変数を新しいパス名に再設定してください。オペレーティング・システムがあらかじめ定義されたパス名を使用する場合は、パスワード・ファイルの位置は変更できません。

パスワード・ファイルの削除

ユーザーの認証にパスワード・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しかデータベース管理操作ができません。

警告： 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・ファイルを削除したり修正したりしないでください。そのようにすると、パスワード・ファイルを使用してリモートに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・ファイルは使用することができません。

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・ファイルの状態は、REMOTE_LOGIN_PASSWORDFILE パラメータを設定すると変更できます。STARTUP 文で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とが比較されます。これらの値が一致しない場合は、ファイル内の値が上書きされます。

警告： 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の REMOTE_LOGIN_PASSWORDFILE パラメータの値が一致する必要があります。それ以外の場合は、パスワード・ファイルの状態は、インスタンスを起動した場所によって変わります。

データベース管理者ユーティリティ

Oracle Server には、そのメンテナンスと制御に役立つ便利なユーティリティがいくつか用意されています。

この項のトピックは、次のとおりです。

- [SQL*Loader](#)
- [エクスポートとインポート](#)

SQL*Loader

SQL*Loader は、Oracle のデータベース管理者とユーザーの両方が使用します。SQL*Loader は、オペレーティング・システムの標準ファイル（テキストまたは C データ形式のファイル）から Oracle データベース表にデータをロードします。

関連項目：『Oracle8i ユーティリティ・ガイド』

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・ユーティリティにより、Oracle データベースとの間で Oracle 形式の既存のデータを移動できます。たとえば、エクスポート・ファイルを使用して、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異なる Oracle データベース間でデータを移動できます。

関連項目：『Oracle8i ユーティリティ・ガイド』

データベース管理者の優先順位

一般に、データベース管理者は、データベース・システムを編成し、稼働させるための一連の手順を実行し、メンテナンスする必要があります。

どのタイプのコンピュータ・システムでも、Oracle Server およびデータベースを構成するには、次に示すステップを実行する必要があります。次の項で、各ステップについて説明します。

Oracle Server の構成手順

- ステップ 1: Oracle ソフトウェアのインストール
- ステップ 2: データベース・サーバー・ハードウェアの評価
- ステップ 3: データベースの計画
- ステップ 4: データベースの作成とオープン
- ステップ 5: データベース設計のインプリメント
- ステップ 6: データベースのバックアップ
- ステップ 7: システム・ユーザーの登録
- ステップ 8: データベースのパフォーマンス調整

注意： 新しいリリースへ移行する場合は、既存の本番データベースのバックアップをとってからインストールしてください。このようなデータベースの保存方法の詳細は、『Oracle8i 移行ガイド』を参照してください。

ステップ 1: Oracle ソフトウェアのインストール

データベース管理者は、Oracle Server ソフトウェアと、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する必要があります。分散処理環境のインストレーションによっては、データベースが中央のコンピュータによって制御され、データベースのツールとアプリケーションはリモート・マシン上で実行されることがあります。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 Net8 ドライバもインストールする必要があります。

関連項目： 詳細は、1-20 ページの「[Oracle ソフトウェア・リリースの識別](#)」を参照してください。

インストレーションの特定の要件や指示の詳細は、使用しているオペレーティング・システム固有の Oracle マニュアルと、フロント・エンド・ツールおよび Net8 ドライバのインストレーション・ガイドを参照してください。

ステップ 2: データベース・サーバー・ハードウェアの評価

インストール後、使用可能なコンピュータ・リソースを、Oracle とそのアプリケーションが最大限に活用するにはどうしたらよいかを評価します。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）

ステップ 3: データベースの計画

データベース管理者は、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構造
- 全体のデータベース設計
- 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構造が、システムのパフォーマンスと種々の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、いくつのデータ・ファイルで表領域を構成するのか、データ・ファイルが物理的にどこ（どのディスク・ドライブ）に格納されるのか、どのような種類の情報がそれぞれの表領域に格納されるのかについて、表領域を作成する前に把握しておくべきです。データベース全体の論理記憶構造を計画するとき、非常に重要なことは、実際にデータベースが作成され、稼働したときに、この構造によって生じる影響を考慮しておくことです。次の項目について、データベースの論理記憶構造が及ぼす影響を検討してください。

- 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

また、データベースのオブジェクト間のリレーショナル設計と、各オブジェクトの記憶特性を計画してください。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に直接的に影響を与えます。また、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す可能性があります。

また、この計画段階で、データベースのバックアップも計画してください。この計画を作成した後で、バックアップの効率を向上させるために、計画したデータベースの論理記憶構造、またはデータベース設計を変更すべきことに気付くこともあります。

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扱っていません。このような設計上の問題の詳細は、これらの問題を説明している、業界標準として一般に認められている資料を参照してください。

関連項目：データベースの論理記憶構造、オブジェクトおよび整合性制約の作成の詳細は、第 9 章～第 21 章を参照してください。

ステップ 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、通常の使用のためにデータベースを作成し、オープンできます。オペレーティング・システムによっては、Oracle のインストール・プロシージャの中で初期データベースがすでに作成されていることがあります。その場合には、インスタンスを起動し、初期データベースをマウントしてからオープンするのみです。

Oracle のインストール中にオペレーティング・システムが初期データベースを作成するかどうかを判断するには、インストール・ガイドまたはユーザズ・ガイドを参照してください。インストール時にデータベースが作成されない場合、または追加のデータベースを作成する場合は、この手順について第 2 章を参照してください。

関連項目：データベースとインスタンスの起動および停止手順の詳細は、第 3 章を参照してください。

ステップ 5: データベース設計のインプリメント

データベースの作成および起動の後に、必要なすべてのロールバック・セグメントと表領域を作成することにより、計画したデータベースの論理構造を作成できます。これが作成されると、データベースにオブジェクトを作成できます。

関連項目：データベースの論理記憶構造とオブジェクトの作成手順については、第 9 章～第 21 章を参照してください。

ステップ 6: 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に追加の REDO ログ・ファイルを作成し、最初の全体データベース・バックアップ（オンラインまたはオフライン）を実行し、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することによって、データベースのバックアップ計画を完了します。

関連項目：バックアップ操作をカスタマイズする方法と回復手順の実行方法の詳細は、『Oracle8i バックアップおよびリカバリ・ガイド』を参照してください。

ステップ 7: 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完了後に、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、登録したユーザーのロールを作成し、各ユーザーに適切なロールを付与できます。

関連項目：ユーザー・アカウントやロールの作成手順およびライセンス契約の遵守の詳細は、第 22 章～第 24 章を参照してください。

ステップ 8: データベースのパフォーマンス調整

データベース・システムのパフォーマンスを最適化することは、データベース管理者の日常的な責任の 1 つです。

関連項目：データベースとアプリケーションの調整の詳細は、『Oracle8i チューニング』を参照してください。

Oracle ソフトウェア・リリースの識別

Oracle 製品の開発と変更は常に進行しているため、ある時点でユーザーに提供されている製品のリリース番号が複数存在する可能性があります。ソフトウェア製品を完全に見分けるには、5 つの番号が必要となります。

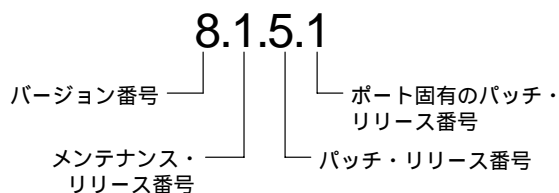
この項のトピックは、次のとおりです。

- [リリース番号の形式](#)
- [他の Oracle ソフトウェアのバージョン](#)
- [現行のリリース番号のチェック](#)

リリース番号の形式

たとえば、Oracle Server 配布テープに「リリース 8.1.5.1」というラベルが付いているとします。この番号の各部の意味は次のとおりです。

図 1-2 Oracle のリリース番号の例



バージョン番号

8 のようなバージョン番号が最も一般的な識別子です。バージョンは、通常は重要な新しい機能が含まれる、ソフトウェアの主な新規エディションを示すものです。

メンテナンス・リリース番号

メンテナンス・リリース番号は、一般的なバージョンの異なるリリース番号を示し、バージョン 8.0 でもそうであるように、0 から始まります。メンテナンス・リリース番号は、既存のプログラムのバグ修正や新しい機能が使用可能になるたびに大きくなります。この例では、8.1 はバージョン 8 の最初のメンテナンス・リリースであることを示します。

パッチ・リリース番号

パッチ・リリース番号は、8.1.5 など、特定レベルのオブジェクト・コードを示します。パッチ・リリースには、次のメンテナンス・リリースまで待てないほど重大なバグの修正が含まれています。メンテナンス・リリースの最初の提供版のパッチ番号は、常に 0 です。

ポート固有のパッチ・リリース番号

4 番目（および、場合によっては 5 番目）の番号では、8.1.5.1. や 8.1.5.1.3 など、そのオペレーティング・システム上のソフトウェア製品の特定の緊急パッチ・リリースを識別できます。通常、緊急パッチは広く配布することを意図したものではなく、特定の重大な問題を修正または回避するためのものです。

リリース番号の例

Oracle8i のリリース番号の例を次に示します。

8.0.0	Oracle8i の初回配布分
8.1.0	Oracle8i の初回メンテナンス・リリース
8.2.0	Oracle8i の 2 番目のメンテナンス・リリース（全体では 3 番目のリリース）
8.2.2	2 番目のメンテナンス・リリース後の 2 番目のパッチ・リリース

他の Oracle ソフトウェアのバージョン

オラクル社は、新製品を発表したり、既存製品の機能を拡張しているため、個々の製品のバージョン番号は別々に増えます。したがって、Oracle Server リリース 8.1.5.1 システムを、Oracle Forms バージョン 4.0.3、SQL*Plus バージョン 3.1.9 および Pro*FORTRAN バージョン 1.5.2 とともに稼働させることもあります（これらの番号は、あくまでも一例です）。

現行のリリース番号のチェック

現在使用している Oracle とそのコンポーネントのリリースを確認するには、次のようにして、データ・ディクショナリ・ビュー PRODUCT_COMPONENT_VERSION を問い合わせます（この情報は、オラクル社カスタマ・サポートに連絡する必要があるときに役立ちます）。

```
SELECT * FROM product_component_version;
```

PRODUCT	VERSION	STATUS
-----	-----	-----
CORE	3.4.1.0.0	Production
NLSRTL	3.1.3.0.0	Production
Oracle8i Server	8.1.4.0.0	Beta Release
PL/SQL	2.2.1.0.0	Beta
TNS for SunOS:	2.1.4.0.0	Production
5 rows selected		

Oracle データベースの作成

この章では、Oracle データベースを作成するために必要な手順を説明します。この章のトピックは、次のとおりです。

- データベースを作成する前の考慮点
- Oracle データベースの作成
- パラメータ
- データベースを作成した後の考慮点
- 初期のチューニング・ガイドライン

データベースを作成する前の考慮点

この項のトピックは、次のとおりです。

- [作成の前提条件](#)
- [初期データベースの使用](#)
- [旧リリースのデータベースの移行](#)

データベース作成では、いくつかのオペレーティング・システム・ファイルを準備して、それらが Oracle データベースとして作動するようにします。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、1 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も作成できます。

データベース作成では、次のような操作を実行します。

- 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。
- 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（データ・ディクショナリ）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。

データベースを作成する前に、次の点を考慮してください。

- 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。
- オンライン REDO ログとアーカイブ済み REDO ログの構成（そしてこれらが必要とする領域）を含め、新規データベースを保護する方法を計画し、バックアップ計画を立てます。
- データベースのキャラクタ・セットを選択します。データベースの作成時には、データベースのキャラクタ・セットを指定する必要があります。データベースを作成した後でキャラクタ・セットを変更するには、再度作成する必要があります。そのため、使用するキャラクタ・セットの選択は十分注意して行うことが重要です。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・セットに格納されます。ユーザーが別のキャラクタ・セットを使用してデータベースにアクセスする場合、データベースのキャラクタ・セットは、ユーザーの使用しているすべてのキャラクタ・セットと同じか、またはそのスーパーセットである必要があります。

また、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、パラメータ・ファイルの使用に関する原理とオプションについても理解しておく必要があります。

関連項目：『Oracle8i NLS ガイド』

表、索引および領域管理の詳細は、第 12 ~ 19 章を参照してください。

オンライン REDO ログおよびアーカイブ REDO ログの詳細は、それぞれ [第 6 章「オンライン REDO ログの管理」](#) と [第 7 章「アーカイブ済み REDO ログの管理」](#) を参照してください。

作成の前提条件

新しいデータベースを作成するには、次の項目が必要です。

- データベース管理者があらゆる操作を実行できるようなオペレーティング・システム権限
- Oracle インスタンスを起動するために十分なメモリー
- 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領域

初期データベースの使用

使用しているオペレーティング・システムによっては、Oracle のインストール・プロシージャの一部として初期データベースがすでに自動的に作成されていることがあります。この初期データベースを使用して情報管理要件に一致するようにカスタマイズしたり、この初期データベースを廃棄し、1 つ以上の新規データベースを作成して置き換えることができます。

旧リリースのデータベースの移行

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だけです。それ以外の場合は、旧バージョンの Oracle が管理している既存の Oracle データベースを移行して、新しいバージョンの Oracle ソフトウェアで使用できます。

関連項目：既存のデータベースの移行方法の詳細は、『Oracle8i 移行ガイド』を参照してください。

既存のデータベースの移行方法の詳細は、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。

Oracle データベースの作成

この項のトピックは、次のとおりです。

- [Oracle データベースを作成する手順](#)
- [データベースの作成例](#)
- [データベース作成の問題解決](#)
- [データベースの削除](#)

Oracle データベースを作成する手順

次に示す手順は、Oracle データベースを作成する方法を示したもので、ここに記述されている順序で必ず実行してください。

新しいデータベースを作成し、システムで使用可能にする手順

1. 既存のデータベースのバックアップを作成します。
2. パラメータ・ファイルを作成します。
3. 新しいパラメータ・ファイルを編集します。
4. システムのインスタンス識別子を確認します。
5. SQL*Plus を起動し、SYSDBA として Oracle に接続します。
6. インスタンスを起動します。
7. データベースを作成します。
8. データベースのバックアップを作成します。

関連項目：これらの手順では、すべてのオペレーティング・システム上でのデータベース作成作業全般について説明しています。使用しているプラットフォーム上でのデータベース作成の詳細は、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。

ステップ 1: 既存データベースのバックアップ作成。データベース作成は既存ファイルに影響を及ぼす可能性があるため、新しいデータベースを作成する前に、すべての既存データベースの完全バックアップを作成することを強くお勧めします。バックアップには、パラメータ・ファイル、データ・ファイル、REDO ログ・ファイル、制御ファイルを含めなければなりません。

ステップ 2: 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（システム・グローバル領域とバックグラウンド・プロセス）は、パラメータ・ファイルを使用して起動されます。

システム上の各データベースには、そのデータベースにのみ対応する、カスタマイズしたパラメータ・ファイルを少なくとも 1 つ用意してください。複数のデータベースで同一ファイルを使用しないでください。

これから作成するデータベースのパラメータ・ファイルを作成するには、オペレーティング・システムを使用して、オラクル社が配布メディア上に提供しているパラメータ・ファイルのコピーを作成してください。このコピーに新しいファイル名を付けます。その後、新しいデータベース用にこの新しいファイルを編集したり、カスタマイズできます。

関連項目：パラメータ・ファイルのコピーの詳細は、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。

注意： 分散処理環境では、Enterprise Manager がネットワーク内のクライアント・マシンから実行されます。クライアント・マシンを使用して Enterprise Manager を実行し、新しいデータベースを作成している場合は、新しいパラメータ・ファイル（この時点では Oracle を稼働しているコンピュータ上にある）をクライアント・ワークステーションにコピーする必要があります。この手順は、オペレーティング・システムによって異なります。ネットワーク内のコンピュータ間でファイルをコピーする方法の詳細は、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。

ステップ 3: 新しいパラメータ・ファイルの編集。データベースを新しく作成するには、新しいパラメータ・ファイルの中で次のパラメータを必ず検査し、編集してください。

パラメータ	参照先
DB_NAME	2-9 ページ
DB_DOMAIN	2-9 ページ
CONTROL_FILES	2-9 ページ
DB_BLOCK_SIZE	2-10 ページ
DB_BLOCK_BUFFERS	2-11 ページ
PROCESSES	2-11 ページ
ROLLBACK_SEGMENTS	2-11 ページ

該当するライセンス・パラメータも必ず編集してください。

パラメータ	参照先
LICENSE_MAX_SESSIONS	2-12 ページ
LICENSE_SESSION_WARNING	2-12 ページ
LICENSE_MAX_USERS	2-13 ページ

ステップ 4: システムのインスタンス識別子の確認。他にデータベースがある場合、Oracle インスタンス識別子をチェックしてください。システムで同時に稼働している他の Oracle インスタンスとの混同を避けるため、Oracle のインスタンス識別子は、データベース名（DB_NAME の値）と一致する必要があります。

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。

ステップ 5: SQL*Plus の起動と、SYSDBA としての Oracle への接続。 データベースに SYSDBA として接続します。

```
$ SQLPLUS /nolog
connect username/password as sysdba
```

ステップ 6: インスタンスの起動。 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;
```

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

ステップ 7: データベースの作成。 新しいデータベースを作成するには、SQL の CREATE DATABASE 文を使用します。この文には、データベースの指定、ファイルの最大数の設定、ファイルの指定およびそのサイズの設定など、オプションでパラメータを設定できます。

CREATE DATABASE 文を実行すると、Oracle は次の操作を実行します。

- データベースにデータ・ファイルを作成します。
- データベースに制御ファイルを作成します。
- データベースに REDO ログ・ファイルを作成します。
- SYSTEM 表領域と SYSTEM ロールバック・セグメントを作成します。
- データ・ディクショナリを作成します。
- ユーザー SYS と SYSTEM を作成します。
- データベースへのデータの格納に使用するキャラクタ・セットを指定します。
- データベースをマウントし、オープンします。

警告： 指定したデータ・ファイルおよび REDO ログ・ファイルの名前が、別のデータベースの各ファイルと矛盾しないかどうかを確認してください。

関連項目： ローカルに管理される SYSTEM 表領域を持つデータベースの作成もできます。詳細は、9-5 ページの「[ローカルに管理される SYSTEM 表領域を持つデータベースの作成](#)」を参照してください。

ステップ 8: データベースのバックアップ作成。 メディア障害が発生した場合に回復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、データベースの完全バックアップを作成する必要があります。

関連項目：『Oracle8i バックアップおよびリカバリ・ガイド』

パラメータ・ファイルの詳細は、3-13 ページの「[パラメータ・ファイルの使用](#)」を参照してください。

CREATE DATABASE 文、キャラクタ・セット、データベース作成の詳細は、『Oracle8i SQL リファレンス』を参照してください。

データベースの作成例

次に、CREATE DATABASE 文の例を示します。

```
CREATE DATABASE test
  DATAFILE 'test_system' SIZE 10M
  LOGFILE GROUP 1 ('test_log1a', 'test_log1b') SIZE 500K,
  GROUP 2 ('test_log2a', 'test_log2b') SIZE 500K;
```

この例で、MAXLOGFILES、MAXLOGMEMBERS、MAXDATAFILES、MAXLOGHISTORY および MAXINSTANCES の各オプションに対する値は、オペレーティング・システムに固有のデフォルト値を想定しています。データベースは、デフォルト・モードの NOARCHIVELOG と EXCLUSIVE でマウントされ、オープンされます。

上の文の項目と情報では、次のような特性を持つデータベースが作成されます。

- 新しいデータベースには TEST という名前が付けられます。
- 新しいデータベースの SYSTEM 表領域は、10MB の TEST_SYSTEM という名前の 1 つのデータ・ファイルから構成されます。
- 新しいデータベースは、それぞれ 500KB のメンバーを 2 つ含む、2 つのオンライン REDO ログ・グループを持っています。
- 新しいデータベースは、パラメータ・ファイルで指定されている既存の制御ファイルを上書きしません。

注意： データベースを作成するとき、制限をいくつか設定できます。また、これらの制限のいくつかは、オペレーティング・システムの制限にかかわる条件となり、お互いに影響を及ぼす可能性があります。たとえば、MAXDATAFILES を設定した場合、最初はデータベースにデータ・ファイルが 1 つしかなくても、Oracle は MAXDATAFILES だけのファイル名を格納するために十分な領域を制御ファイル内に割り当てます。制御ファイルの最大サイズには制限があり、オペレーティング・システムによって異なりますから、CREATE DATABASE のパラメータすべてをその理論的な最大値に設定できない可能性があります。

関連項目：データベースの作成時に制限を設定する方法の詳細は、『Oracle8i SQL リファレンス』を参照してください。

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。

データベース作成の問題解決

なんらかの理由でデータベース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。

データベース作成の失敗の原因となったエラーを訂正した後で、「[データベースの作成例](#)」に戻ってください。

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、パラメータ・ファイルおよびアーカイブ済みログ・ファイル）すべてを削除します。

データベースのデータ・ファイルと REDO ログ・ファイルの名前を表示するには、データ・ディクショナリ・ビュー V\$DATAFILE と V\$LOGFILE を問い合わせます。

関連項目：これらのビューの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

パラメータ

「[Oracle データベースの作成](#)」のステップ 3 で説明するように、オラクル社では最小のパラメータ・セットを変更することを提案しています。これらのパラメータについては、次の項で説明します。

- [DB_NAME](#) および [DB_DOMAIN](#)
- [CONTROL_FILES](#)
- [DB_BLOCK_SIZE](#)
- [PROCESSES](#)
- [ROLLBACK_SEGMENTS](#)
- [ライセンスに関するパラメータ](#)
- [DB_BLOCK_BUFFERS](#)
- [LICENSE_MAX_SESSIONS](#) と [LICENSE_SESSIONS_WARNING](#)
- [LICENSE_MAX_USERS](#)

DB_NAME および DB_DOMAIN

データベースのグローバル・データベース名（ネットワーク構造内の名前と位置）は、データベースを作成する前に、DB_NAME と DB_DOMAIN の両パラメータを設定することによって作成されます。作成した後では、データベースの名前は簡単には変更できません。DB_DOMAIN パラメータはネットワーク構造内のドメイン（論理的な位置）を示し、DB_NAME パラメータはデータベース名のローカル名構成要素を決定します。これら 2 つのパラメータの設定を組み合わせ、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、次のように新しいパラメータ・ファイルのパラメータを編集します。

```
DB_NAME = TEST
DB_DOMAIN = US.ACME.COM
```

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。データベースの作成時に、DB_NAME に指定した名前は、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されません。

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列です。通常、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、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、『Oracle8i 分散システム』を参照してください。

CONTROL_FILES

新しいパラメータ・ファイルに CONTROL_FILES パラメータを指定し、新しいデータベースで使用する制御ファイル名（リスト）にその値を設定してください。データベース用の制御ファイルを作成するときに Oracle が新しいオペレーティング・システム・ファイルを作成するには、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 Oracle が既存のファイルを再使用、または上書きするには、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。

警告： このオプションを選択する場合には十分に注意してください。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。

CONTROL_FILES パラメータにファイル名を記述しないと、デフォルト・ファイル名が使用されます。

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オラクル社はお勧めします。したがって、新しいパラメータ・ファイルの CONTROL_FILES パラメータを指定するときには、次のガイドラインに従ってください。

- CONTROL_FILES パラメータに少なくとも 2 つのファイル名を記述してください。
- ファイル名ごとに、異なるディスク・ドライブを参照するファイル名を完全に指定することによって、別々の物理ディスク・ドライブ上に各制御ファイルを配置してください。

注意： 制御ファイルのファイル指定は、オペレーティング・システムによって異なります。使用しているオペレーティング・システムとは関係なく、制御ファイルには常にファイル名を省略せずに指定してください。

CREATE DATABASE 文を実行する（ステップ 7）場合、パラメータ・ファイルの CONTROL_FILES パラメータで表示されている制御ファイルが作成されます。

関連項目： CONTROL_FILES パラメータ用の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。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。

DB_BLOCK_SIZE

Oracle server のデータ・ブロックのデフォルト・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。標準では、Oracle のデータ・ブロック・サイズは 2KB か 4KB のどちらかです。一般に、デフォルトのデータ・ブロック・サイズで十分です。ただし、場合によってはデータ・ブロック・サイズを大きくして、ディスクとメモリーの I/O（データのアクセスと記憶）の効率を向上させることができます。それは次のような場合です。

- 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 を稼働させるオペレーティング・システムが小さなオペレーティング・システムのブロック・サイズを使用する場合。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、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ために、データベース・ブロックは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。

各データベースのブロック・サイズは、初期化パラメータ DB_BLOCK_SIZE によってデータベース作成時に設定されます。データベースを作成した後は、データベースを作成し直す以

外にブロック・サイズの変更はできません。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、データベースのブロック・サイズは、オペレーティング・システムのブロック・サイズの倍数である必要があります。

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、DB_BLOCK_SIZE 初期化パラメータに対して次のように設定すると有効です。

```
DB_BLOCK_SIZE=4096
```

また、初期化パラメータ DB_BLOCK_SIZE は、システム・グローバル領域 (SGA) のバッファ・キャッシュ内のデータベース・バッファのサイズも決定します。

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。

DB_BLOCK_BUFFERS

このパラメータは、システム・グローバル領域 (SGA) 内のバッファ・キャッシュのバッファ数を決定します。このバッファ数がキャッシュのパフォーマンスに影響を及ぼします。キャッシュ・サイズを大きくすると、修正したデータをディスクに書き込む回数が少なくなります。ただし、キャッシュを大きくすることによって、メモリーを使用しすぎて、ページングやスワッピングが発生する可能性もあります。

表、索引およびロールバック・セグメントを含めて、アプリケーションが最も頻繁にアクセスするデータ・ブロックの数を見積ってください。この見積りが、およそのキャッシュが保持すべきバッファの最小数となります。一般に、バッファ数の実際の最小値は 1000 ~ 2000 です。

関連項目 : バッファ・キャッシュの調整方法の詳細は、『Oracle8i チューニング』を参照してください。

PROCESSES

このパラメータは、Oracle へ同時に接続できるオペレーティング・システム・プロセスの最大数を決定します。このパラメータには、バックグラウンド・プロセスのための値を 5、ユーザー・プロセスごとの値を 1 として、それらを加算した値を指定する必要があります。たとえば、50 の同時実行ユーザーを計画している場合、このパラメータを最低でも 55 に設定してください。

ROLLBACK_SEGMENTS

このパラメータは、Oracle インスタンスがデータベースの起動時に獲得するロールバック・セグメントのリストです。このパラメータの値として、ロールバック・セグメントを記述してください。

注意： インストールした後、必ず SYSTEM ロールバック・セグメントの他に SYSTEM 表領域の中に 1 つ以上のロールバック・セグメントを作成してからスキーマ・オブジェクトを作成してください。

関連項目： 必要なロールバック・セグメント数の詳細は、『Oracle8i チューニング』を参照してください。

ライセンスに関するパラメータ

Oracle は、サイトが Oracle のライセンス契約に従っていることを保証する手助けをします。サイトが同時ユーザーのライセンスを受けている場合、インスタンスに同時に接続するセッション数を追跡し、制限できます。サイトが端末ユーザーによってライセンスされている場合、データベース内に作成できる端末ユーザー数を制限できます。この機能を使用するには、サイトが持っているライセンス契約の種類とセッションまたは端末ユーザーの最大数を知っている必要があります。サイトは、どちらかの種類のライセンス（セッション・ライセンスまたは端末ユーザー・ライセンス）を使用し、両方を使用することはありません。

関連項目： ライセンス管理の詳細は、23-2 ページの「[セッションとユーザーのライセンス](#)」を参照してください。

LICENSE_MAX_SESSIONS と LICENSE_SESSIONS_WARNING

指定したコンピュータ上のデータベースに接続できる同時実行セッションの数に対して、制限を設定できます。インスタンスの同時実行セッションの最大数を設定するには、次の例に示すようにインスタンスを起動するパラメータ・ファイルに LICENSE_MAX_SESSIONS パラメータを設定します。

```
LICENSE_MAX_SESSIONS = 80
```

セッションの最大数の設定に加えて、同時実行セッションの数に対して警告制限を設定できます。この制限に達しても、追加のユーザーは（最大制限まで）接続できますが、Oracle は接続中の各ユーザーに警告を送信します。インスタンスに対して警告制限を設定するには、パラメータ LICENSE_SESSIONS_WARNING を設定します。警告制限は、LICENSE_MAX_SESSIONS よりも小さな値に設定してください。

Parallel Server で実行しているインスタンスの場合、各インスタンスには、固有の同時使用制限と警告制限を設定できます。ただし、それらのインスタンスの制限の合計は、サイトのセッション・ライセンスを超えないでください。

関連項目： Parallel Server を使用するときこれらの制限を設定する方法の詳細は、『Oracle8i Parallel Server 概要および管理』を参照してください。

LICENSE_MAX_USERS

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、だれも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、端末ユーザー・ライセンスは、Oracle のライセンス契約に従っていることを確認する手助けができるように、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パラメータ・ファイルに LICENSE_MAX_USERS パラメータを設定します。

```
LICENSE_MAX_USERS = 200
```

Parallel Server で稼働しているインスタンスの場合、同一のデータベースに接続しているインスタンスはすべて同じ端末のユーザー制限を持っています。

関連項目： Parallel Server を使用するときこれらの制限を設定する方法の詳細は、『Oracle8i Parallel Server 概要および管理』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

データベースを作成した後は、インスタンスは稼働したままであり、データベースは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。データベース・システムに 1 つ以上のデータベースが存在する場合、以降のデータベース起動時に、使用するパラメータ・ファイルを指定してください。

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、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。その他の製品の詳細は、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。通常、データベースのデータ・ディクショナリにこれらの表を作成し、データをロードするためのコマンド・ファイルが提供されます。

Oracle Server 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 ファイルを使用すると、システムで試しに操作したり、SQL について学んだり、その他の表、ビュー、シノニムを作成したりできます。

新たに作成したデータベースには、SYS および SYSTEM の 2 人のユーザーしか存在しません。この 2 つのユーザー名のパスワードは、データベースを作成した直後に変更してください。

関連項目： ユーザー SYS および SYSTEM の詳細は、1-4 ページの「[データベース管理者のユーザー名](#)」を参照してください。

ユーザーのパスワードの変更方法の詳細は、23-14 ページの「[ユーザーの変更](#)」を参照してください。

初期のチューニング・ガイドライン

インストールに続いてただちに、2、3 の重要なチューニング上の変更を Oracle に加えることができます。次の指示に従うと、Oracle を稼働時にチューニングする必要が少なくなります。ここでは、次のインストール項目に対する推奨事項を示します。

- [ロールバック・セグメントの割当て](#)
- [DB_BLOCK_LRU_LATCHES の数値の選択](#)
- [I/O の分散](#)

ロールバック・セグメントの割当て

ロールバック・セグメントを適切に割り当てると、データベースのパフォーマンスが最適化されます。最適のパフォーマンスを得るために必要なロールバック・セグメントのサイズと数は、アプリケーションによって異なります。Oracle サーバー上で同時に実行されるトランザクション数に基づいてロールバック・セグメントの割当て数を選択するための全般的なガイドラインについては、『Oracle8i チューニング』を参照してください。ほとんどのアプリケーションでは、このガイドラインが当てはまります。

ロールバック・セグメントを作成するには、CREATE ROLLBACK SEGMENT 文を使用してください。

関連項目：CREATE ROLLBACK SEGMENT 文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

ロールバック・セグメントのサイズの選択

ロールバック・セグメントのサイズもパフォーマンスに影響を及ぼします。ロールバック・セグメントのサイズは、CREATE ROLLBACK SEGMENT 文の STORAGE パラメータによって決定されます。ロールバック・セグメントは、トランザクションのロールバック・エントリを保持できるように十分大きくする必要があります。

関連項目：ロールバック・セグメントのサイズの選択の詳細は、『Oracle8i チューニング』を参照してください。

DB_BLOCK_LRU_LATCHES の数値の選択

LRU (least recently used: 最低使用頻度) ラッチに対する競合は、多数の CPU を持つ対称型マルチプロセッサ (SMP) マシン上ではパフォーマンスを低下させる可能性があります。LRU ラッチは、バッファ・キャッシュ内のバッファの置換を制御します。SMP システムの場合、Oracle は自動的に LRU ラッチの数をシステム上の CPU の数の半分に設定します。SMP 以外のシステムの場合は、LRU ラッチは 1 つで十分です。

システム上の LRU ラッチの数は、初期化パラメータ DB_BLOCK_LRU_LATCHES で指定できます。このパラメータは、必要な LRU ラッチ数の最大値を設定します。各 LRU ラッチによって一連のバッファが制御され、Oracle ではバッファ間で置換バッファの割当てが均衡化されます。

関連項目 : LRU ラッチの詳細は、『Oracle8i チューニング』を参照してください。

I/O の分散

I/O を適切に分散すると、データベースのパフォーマンスをかなり改善できます。I/O は、Oracle のインストール時に分散できます。インストール時に I/O を分散させると、Oracle の稼働時に I/O を分散させる必要を少なくできます。

Oracle をインストールするときに I/O を分散させる方法は、次のようにいくつかあります。

- REDO ログ・ファイルの配置
- データ・ファイルの配置
- 表と索引の分離
- データの密度（データ・ブロックあたりの行）

関連項目 : I/O を分散する方法の詳細は、『Oracle8i チューニング』を参照してください。

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。この章のトピックは、次のとおりです。

- データベースの起動
- データベースの可用性の変更
- データベースの停止方法
- データベースを中断して再開する方法
- パラメータ・ファイルの使用

データベースの起動

この項のトピックは、次のとおりです。

- [インスタンス起動の準備](#)
- [インスタンスの起動方法](#)

データベースやインスタンスをコマンド行から起動するには、SQL*Plus を使用して管理者権限で Oracle に接続し、STARTUP コマンドを発行します。また、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。Enterprise Manager の GUI を使用してコマンド行を使用しない場合の手順は、『Oracle Enterprise Manager 管理者ガイド』を参照してください。

次のように様々な方法でインスタンスとデータベースを起動できます。

- インスタンスを起動し、データベースはマウントしない方法。
- 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方法。
- インスタンスを起動し、データベースのマウントを次のどちらかのモードで行い、オープンする方法。
 - 非制限モード（すべてのユーザーに対して使用可能）
 - 制限モード（データベース管理者のみが使用可能）

注意： マルチスレッド・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、データベースのインスタンスを起動できません。

さらに、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回復を開始したりできます。使用しているオペレーティング・システムが Oracle Parallel Server をサポートしている場合には、インスタンスを起動してから、排他モードまたは共有モードでデータベースをマウントします。

関連項目： OPS 環境でデータベースを起動する方法の詳細は、『Oracle8i Parallel Server 概要および管理』を参照してください。

SQL*Plus コマンド構文の詳細は、『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。

Recovery Manager のコマンドの詳細は、『Oracle8i 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。

インスタンス起動の準備

インスタンスを起動する前に、いくつかの作業を実行する必要があります。

1. 次のように入力し、データベースに接続しないで 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
```

マルチスレッド・サーバー経由では接続できないので注意してください。

3. STARTUP コマンドを入力するときには、次のようにデータベース名とパラメータ・ファイルのフルパスを指定します。

```
STARTUP database_name PFILE=myinit.ora
```

PFILE オプションを指定しなければ、標準パラメータ・ファイルが使用されます。データベース名を指定しなければ、インスタンスを起動するパラメータ・ファイル内の DB_NAME の値が使用されます。

関連項目：ファイル名の使用は、オペレーティング・システムに応じて異なります。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。

DB_NAME パラメータの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

インスタンスの起動方法

次に、インスタンスを起動するいくつかの方法を説明します。

注意： 制御ファイルまたはデータベース・ファイル、REDO ログ・ファイルを使用できない場合、インスタンスの起動で問題が発生することがあります。CONTROL_FILES 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 つまたは複数のデータ・ファイルや 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。

インスタンスを起動し、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のみの方法

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、特定のメンテナンス操作を実行するときにオープンできます。たとえば、次のような操作の実行中は、データベースをマウントしても、オープンしてはなりません。

- データ・ファイルを改名する操作。
- REDO ログ・ファイルを追加、削除または改名する操作。
- REDO ログ・アーカイブ・オプションを使用可能、または使用禁止にする操作。
- 全データベース回復を実行する操作。

STARTUP コマンドで MOUNT オプションを指定して、インスタンスを起動し、データベースをマウントし、クローズしたままにしておきます。

```
STARTUP MOUNT;
```

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、インスタンスが起動されており、データベースがマウントされてオープンされていることを意味します。これにより、有効なユーザーはその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るようになります。

オプションを指定せずに STARTUP コマンドを使用して、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します。

```
STARTUP;
```

起動時にデータベースへのアクセスを制限する方法

データベースを管理担当者のみ使用することができ、データベースの一般ユーザーには使用できないようにするために、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し、オープンします。以下のいずれかの作業を実行するときは、このデータベース起動モードを使用してください。

- 索引の再構築など、構造上のメンテナンスを実行する場合。
- 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。
- データ・ロードを実行する場合（SQL*Loader を使用）。
- 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。

一般に、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にデータベース・アクセスを許可します。したがって、RESTRICTED SESSION システム権限は、データベース管理者のみが持つようにしてください。

STARTUP コマンドで RESTRICT オプションを指定し、制限モードでインスタンスを起動（必要であればデータベースをマウントしてオープン）します。

```
STARTUP RESTRICT;
```

後で ALTER SYSTEM 文を使用して RESTRICTED SESSION 機能を使用禁止にします。

関連項目 : ALTER SYSTEM 文の詳細は、『Oracle8i 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

データベース・インスタンスを起動しようとして、問題が発生することがまれにあります。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。

- 現行のインスタンスを 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（および、必要であればデータベースをマウントし、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は、STARTUP FORCE を使用すると、ABORT モードで停止してから再起動します。

インスタンスを起動し、データベースをマウントし、完全メディア回復を開始する方法

メディア回復が必要な場合には、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、回復処理を自動的に開始できます。

```
STARTUP OPEN RECOVER;
```

不要な場合に回復を実行しようすると、エラー・メッセージが表示されます。

排他モードまたはパラレル・モードで起動する方法

Oracle Server で、複数インスタンスが同時に 1 つのデータベースにアクセスできる場合、データベースを排他的にマウントするかパラレルにマウントするかを選択してください。たとえば、パラレル・モードでオープンする場合は、次のコマンドを発行できます。

```
STARTUP OPEN sales PFILE=INITSALE.ORA PARALLEL;
```

インスタンスとデータベースの起動例

次の例では、INITSALE.ORA という名前のパラメータ・ファイルを使用してインスタンスを起動し、sales というデータベースを排他モードでマウント、オープンし、アクセスを管理担当者に制限しています。データベース管理者は、すでに管理者権限で接続しています。

```
STARTUP OPEN sales PFILE=INITSALE.ORA EXCLUSIVE RESTRICT;
```

オペレーティング・システム起動時のデータベース自動起動の方法

システム起動に続いて、すぐに 1 つ以上の Oracle インスタンスとデータベースの自動起動を使用可能にするための手順が、多くのサイトで使用されます。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。

リモート・インスタンスを起動する方法

ローカルの Oracle Server 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する必要がある場合があります。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・システムに依存し、かなり異なります。

関連項目：権限を持たないユーザーにデータベースを使用可能にする方法の詳細は、3-8 ページの「[オープン状態の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。

制御ファイル、データベース・ファイルおよび REDO ログの回復の詳細は、[第 6 章「オンライン REDO ログの管理」](#)および[第 7 章「アーカイブ済み REDO ログの管理」](#)を参照してください。

現行のインスタンスでの異常終了の影響の詳細は、3-12 ページの「[ABORT オプションによる停止](#)」を参照してください。

排他モードまたはパラレル・モードでの起動方法の詳細は、『Oracle8i Parallel Server 概要および管理』を参照してください。

STARTUP 文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、『Oracle8i SQL リファレンス』を参照してください。

自動起動の方法の詳細は、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。

データベースの可用性の変更

インスタンスを起動し、マウントされたデータベースをオープンして、データベースを部分的に使用可能にできます。次に、データベースの可用性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](#)
- [クローズされたデータベースをオープンする方法](#)
- [データベースを読み取り専用モードでオープンする方法](#)
- [オープン状態の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

特定の管理操作の必要があるとき、データベースは、起動され、インスタンスにマウントされていても、クローズされている必要があります。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。

データベースをマウントするときに、データベースをそのインスタンスだけに排他的にマウントするのか、同時に他のインスタンスにもマウントするのかを指示します。

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、SQL 文 ALTER DATABASE で MOUNT オプションを指定します。データベースを排他モードでマウントするには、次の文を使用します。

```
ALTER DATABASE MOUNT;
```

関連項目 : データベースがマウントされ、クローズされている必要のある操作（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、3-4 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のみの方法](#)」を参照してください。

クローズされた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、マウント済みのクローズされ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。マウント済みのデータベースをオープンするには、SQL の 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 システム権限を与えられた正当な Oracle ユーザーであれば、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、オープン状態のデータベースを問い合わせることができ、その間もデータの内容がオンラインで変更される恐れはありません。これにより、データ・ファイルと REDO ログ・ファイルに書き込まれないことが保証され、制限されずにデータベース回復や REDO を生成しない「状態」変更を行うことができます。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、このような切替が可能です。

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回復モードの間で交互に切り替えているときにオープンするのが理想です。この 2 つは相互排他モード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、データベースは次のように読取り書き込みモードでオープンすることもできます。

```
ALTER DATABASE OPEN READ WRITE;
```

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、『Oracle8i SQL リファレンス』を参照してください。

データベースを読取り専用モードでオープンする概念の詳細は、『Oracle8i 概要』を参照してください。

オープン状態のデータベースへのアクセスを制限する方法

正常な条件下では、CREATE SESSION システム権限を持つすべてのユーザーがインスタンスに接続できます。ただし、インスタンスは制限モード、または非制限モードにできます。インスタンスが制限モードになっていると、CREATE SESSION システム権限と RESTRICTED SESSION システム権限を持っているユーザーのみがインスタンスに接続できます。通常、管理者のみが RESTRICTED SESSION システム権限を持っています。

次の作業の必要があるときに制限モードが有効です。

- 索引の再構築など、構造上のメンテナンスを実行する場合。
- 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。
- データ・ロードを実行する場合（SQL*Loader を使用）。

- 管理者以外のユーザーがデータを一時的に使用できないようにする場合。

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 オプションを指定します。インスタンスを制限モードにした後、管理作業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（KILL）する場合があります。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：データベース・インスタンスの起動と、制限モードでのデータベースのマウントおよびオープンの詳細は、3-4 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。

ALTER SYSTEM 文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

データベースの停止方法

ここでは、次の様々な手順について詳しく説明します。

- [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](#)

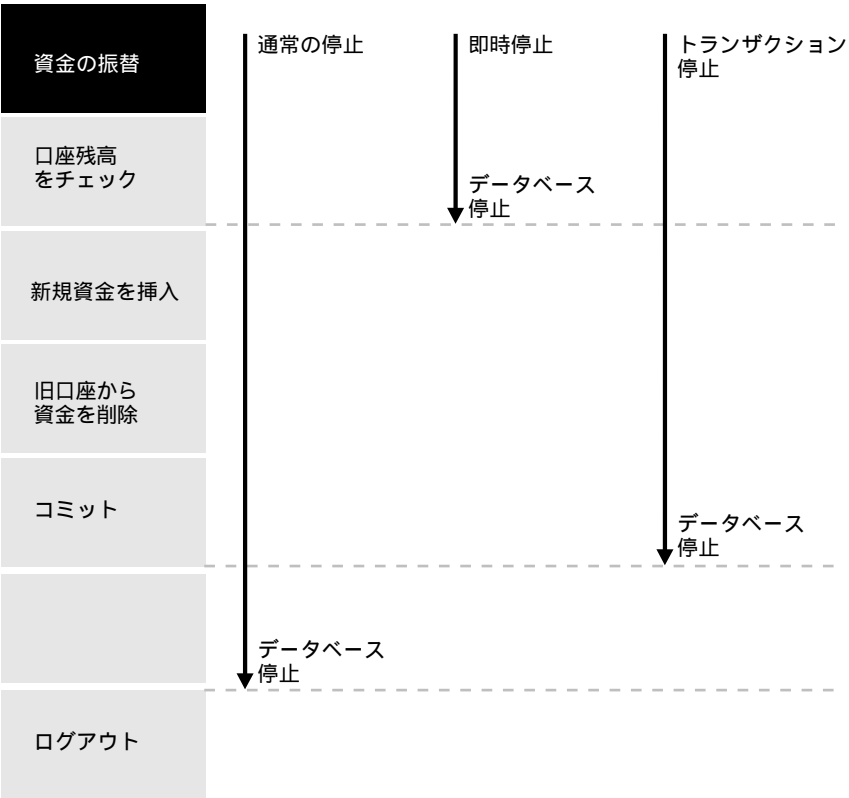
データベースを停止するには、SQL*Plus の SHUTDOWN コマンドを使います。停止が完了するまで、データベース停止を開始するセッションに制御が戻りません。停止処理の進行中に接続しようとするユーザーは、次のようなメッセージを受け取ります。

ORA-01090: shutdown in progress - connection is not permitted

注意： マルチスレッド・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 で接続します。図 3-1 は、銀行口座間の資金の振替え時に様々な SHUTDOWN コマンドが入力された場合に発生するイベントの順序を示しています。

図 3-1 各種の SHUTDOWN 時のイベントの順序



NORMAL オプションによる停止

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、どのような新しい接続も許しません。
- 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。

- データベースの次の起動では、どのようなインスタンス回復手順も要求しません。

データベースを通常の状況で停止するには、SHUTDOWN コマンドで NORMAL オプションを指定します。

```
SHUTDOWN NORMAL;
```

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だけ使用します。

- 電源が間もなく停止する場合。
- データベースまたはデータベース・アプリケーションの 1 つが異常に作動している場合。
このデータベース停止では、次のように処理が進みます。
- 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、この停止方法は、その名前に反してただちに成されない可能性があります。)
- Oracle は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。Oracle はアクティブ・トランザクションを暗黙のうちにロールバックしてから、接続しているユーザーをすべて切断します。

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;
```

注意： SHUTDOWN IMMEDIATE 文は既存のアイドル接続をすべて切断し、データベースを停止します。ただし、結果待ちになっているプロセス（挿入、検索または更新など）を送った場合、SHUTDOWN TRANSACTIONAL 文を使用するとそのプロセスの結果を返してから切断されます。

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先に完了できるようにして、予定どおりのインスタンスの停止を行う場合は、SHUTDOWN コマンドで TRANSACTIONAL オプションを指定します。

```
SHUTDOWN TRANSACTIONAL;
```

この文を送った後、クライアントはこの特定のインスタンスで新しいトランザクションを起動できません。クライアントが新しいトランザクションを起動しようとすると、切断されます。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。このとき、インスタンスは SHUTDOWN IMMEDIATE 文を送る場合のように停止します。

トランザクションのシャットダウンはクライアントが作業を失わないようにすると同時に、すべてのユーザーがログオフする必要はありません。

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中断することによって、データベースをただちに停止できます。ただし、このタイプの停止は、できるだけ次のような状況でのみ実行するようにしてください。

- データベースまたはデータベース・アプリケーションの 1 つが異常に作動していて、かつ、他のタイプの停止がどれも作動しない場合。
- 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）。
- 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。

インスタンスの中断によって、データベースは次のように停止されます。

- Oracle によって処理されている現行のクライアント SQL 文をただちに終了させます。
- コミットされていないトランザクションをロールバックしません。
- Oracle は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。Oracle は暗黙のうちに接続しているユーザーをすべて切断します。

通常の停止オプションと即時停止オプションのどちらも作動しない場合は、SHUTDOWN コマンドで ABORT オプションを指定して、現行のデータベース・インスタンスをただちに中断させます。

```
SHUTDOWN ABORT;
```

データベースを中断して再開する方法

ALTER SYSTEM SUSPEND 文では、すべてのインスタンス内ですべての I/O（データ・ファイル、制御ファイルおよびファイル・ヘッダー）と問合せが中断され、進行中のトランザクションを処理しないでデータベースのコピーを作成できます。新しいインスタンスは中断されないの、あるインスタンスの中断中は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 文を使います。

中断および再開機能は、ディスクやファイルをミラー化し、そのミラーを分割できるシステムで役立ちます。書き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、この中断および再開機能を使用すると容易に分割できます。ただし、中断され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、中断 / 再開機能は通常の停止操作の簡易版ではありません。

SUSPEND 文と RESUME 文は、異なるインスタンスから発行できるので注意してください。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 SUSPEND 文を発

行した場合、インスタンス 1、2 または 3 から同様に RESUME 文を発行することができます。

SUSPEND と RESUME を使用してミラーの分割を容易にする手順

1. ALTER TABLESPACE BEGIN BACKUP 文を使用して、データベース表領域をホット・バックアップ・モードにします。
2. ミラー化システムでディスク書き込み中にミラー分割エラーが発生した場合は、ALTER SYSTEM SUSPEND 文を発行します。
3. ミラーを分割します。
4. ALTER SYSTEM RESUME 文を発行して通常のデータベース操作を再開します。
5. ALTER TABLESPACE END BACKUP 文を使用して、表領域をホット・バックアップ・モードから元のモードに戻します。
6. バックアップのために、通常どおり制御ファイルをコピーし、オンライン REDO ログをアーカイブします。

警告： 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 SUSPEND 文を使用しないでください。

関連項目： ALTER SYSTEM SUSPEND/RESUME 文と ALTER TABLESPACE 文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

パラメータ・ファイルの使用

次に、パラメータ・ファイルの使用方法について説明します。

- サンプル・パラメータ・ファイル
- パラメータ・ファイルの数
- 分散環境におけるパラメータ・ファイルの位置

インスタンスを起動するためには、Oracle はインスタンス構成パラメータのリストが入っているテキスト・ファイルである、「パラメータ・ファイル」を読み込む必要があります。多くの場合、このファイルの名前は INIT.ORA または INITsid.ORA です。sid はオペレーティング・システム固有です。

注意： Oracle Enterprise Manager を使用する場合、初期化パラメータ・ファイルのかわりに格納された構成を使用する方法の詳細は、『Oracle Enterprise Manager 管理者ガイド』を参照してください。

パラメータ・ファイル内のパラメータ値は基本的なテキスト・エディタで編集できますが、編集方法はオペレーティング・システムによって異なります。初期化パラメータの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

ファイル内の各国語サポート (NLS) パラメータに定義されている文字列リテラルは、データベースのキャラクタ・セットであるかのように扱われます。

関連項目: 初期化パラメータ・ファイルの詳細は、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。

サンプル・パラメータ・ファイル

サンプルのパラメータ・ファイル (INIT.ORA または INITSid.ORA) が、Oracle の配布セットに含まれています。このサンプル・ファイルのパラメータは、Oracle データベースの初期インストールに適合しています。システムが稼働してから、Oracle を使用した後で、パラメータ値のいくつかを変更する必要があるかもしれません。

関連項目: パラメータ・ファイルを使用したデータベースのパフォーマンスの最適化の詳細は、『Oracle8i チューニング』を参照してください。

パラメータ・ファイルの数

Oracle データベースには、そのデータベースにのみ対応するパラメータ・ファイルが 1 つ以上あります。このように、与えられたファイル内のデータベース固有パラメータ (DB_NAME、CONTROL_FILES など) は、常に特定のデータベースに関係があります。1 つのデータベースに複数の異なるパラメータ・ファイルを用意できます。たとえば、いろいろな状況でデータベースのパフォーマンスを最適化するために、1 つのデータベースに複数の異なるパラメータ・ファイルを用意できます。

分散環境におけるパラメータ・ファイルの位置

データベースへのアクセスに使用するクライアントでは、データベースのパラメータ・ファイルを読み取ってデータベースのインスタンスを起動する必要があります。したがって、データベースのパラメータ・ファイルは、必ずクライアントを稼働するコンピュータに格納してください。

分散処理構成でない場合は、同一のコンピュータで Oracle とクライアントが実行されます。このコンピュータには、すでにパラメータ・ファイルがディスク・ドライブのうちの 1 つに格納されています。ただし、分散処理構成では、リモート・マシンに格納されたデータベースをローカルのクライアント・ワークステーションで管理できます。このタイプの構成では、ローカルのクライアント・マシンには、対応するデータベースの各パラメータ・ファイルのコピーを格納する必要があります。

関連項目: 分散環境で管理用の Oracle を使用方法の詳細は、『Oracle8i 分散システム』を参照してください。

第 II 部

Oracle Server の構成

Oracle プロセスの管理

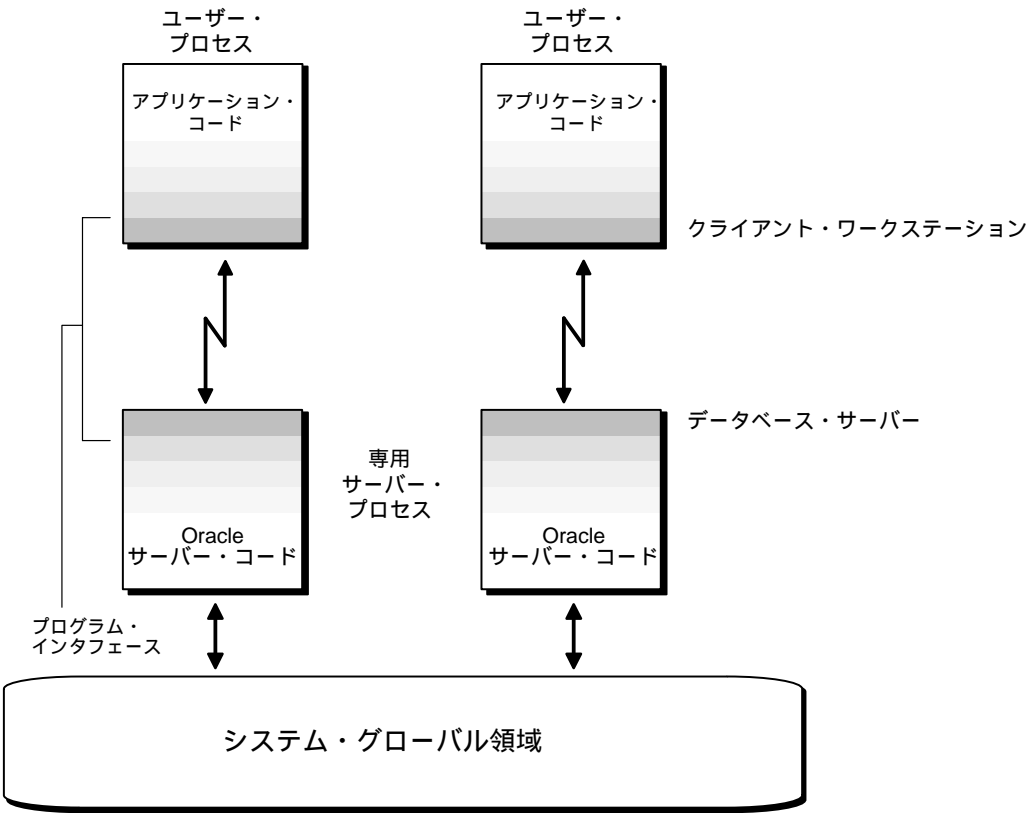
この章では、Oracle インスタンスのプロセスを管理する方法について説明します。この章のトピックは、次のとおりです。

- [サーバー・プロセスのセットアップ](#)
- [マルチスレッド・サーバー・アーキテクチャ用の Oracle の構成](#)
- [サーバー・プロセスの変更](#)
- [Oracle プロセスの追跡](#)
- [パラレル問合せオプションのプロセスの管理](#)
- [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・プロセスのセットアップ

あるユーザー・プロセスによってデータベース・アプリケーションが実行され、別個のサーバー・プロセスによって各ユーザーのために対応付けられた Oracle サーバーが実行される場合に、その別個のサーバー・プロセスを「専用サーバー・プロセス」といいます（[図 4-1](#)を参照）。Oracle はこの構成用に自動的にインストールされます。オペレーティング・システムがこの構成で Oracle をサポートできる場合は、マルチスレッド・サーバー・プロセスをサポートすることも可能です。

図 4-1 Oracle 専用サーバー・プロセス



専用サーバー・プロセスに接続する場合

可能であれば、ディスパッチャを介してインスタンスに接続してください。これにより、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ます。ただし、次の状況

では、ユーザーとデータベース管理者は、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しなければなりません。

- 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）。
- Enterprise Manager を使用してデータベースを起動または停止するか、データベース上でメディア回復を実行する場合。
- Recovery Manager を使用してデータベースをバックアップ、復元または回復する場合。

専用サーバー接続を要求するには、Net8 TNS 接続文字列に SERVER=DEDICATED 句の指定が必要です。

関連項目 : Net8 接続文字列の構文の詳細は、オペレーティング・システム固有の Oracle マニュアルおよび『Oracle8i Net8 管理者ガイド』を参照してください。

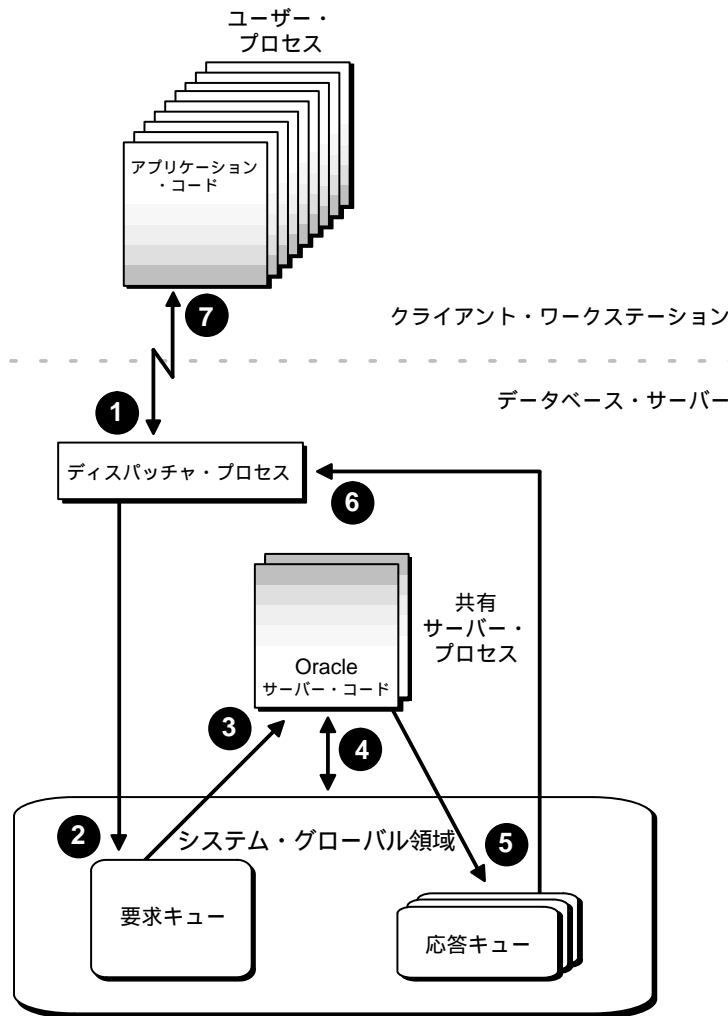
初期化パラメータとパラメータ・ファイルの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

マルチスレッド・サーバー・アーキテクチャ用の Oracle の構成

専用サーバー・プロセスを持つ注文入力システムを例にとつて考えてみます。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。大部分のトランザクションでは、担当者は顧客と電話中で、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。ほとんどのトランザクション中は、サーバー・プロセスは必要とされず、アイドル状態のサーバー・プロセスはシステム・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場合にシステムは低速になります。

マルチスレッド・サーバー・アーキテクチャでは、接続ごとの専用サーバー・プロセスは必要ありません（[図 4-2](#) を参照）。少数の共有サーバー・プロセスで多数の専用サーバー・プロセスと同じ量の処理を実行できます。また、各ユーザーに必要なメモリー量が比較的少なく、わずかなメモリーおよびプロセス管理作業で済むので、多数のユーザーをサポートできます。

図 4-2 Oracle のマルチスレッド・サーバー・プロセス



マルチスレッド・サーバー構成でシステムをセットアップするには、ネットワーク・リスナー・プロセスを起動し、MTS_DISPATCHERS パラメータ（初期ディスパッチャ数を設定する必須パラメータ）を設定します。

この初期化パラメータを設定した後でインスタンスを再起動すると、その時点でマルチスレッド・サーバー構成が使用されます。マルチスレッド・サーバー・アーキテクチャでは、Net8 が必要です。マルチスレッド・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 Net8 を介して接続してください。

関連項目: ネットワーク・リスナー・プロセスの開始と管理の詳細は、『Oracle8i 分散システム』および『Oracle8i Net8 管理者ガイド』を参照してください。

MTS_DISPATCHERS 初期ディスパッチャ数の設定（必須）

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、パラメータ MTS_DISPATCHERS によって制御されます。インスタンスを起動する前に、各ネットワーク・プロトコルに対して、起動するディスパッチャの数を見積ってください。

MTS_DISPATCHERS パラメータを設定するときには、有効なプロトコルをすべて対象とすることができます。

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、オペレーティング・システムによって異なるプロセスあたりの接続数に対するホスト・オペレーティング・システムの制限、およびネットワーク・プロトコルあたりに必要とされる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に対応した同数の接続を提供できなければなりません。インスタンスの起動後には、必要に応じてさらにディスパッチャ・プロセスを開始できます。

関連項目: ディスパッチャ・プロセスの詳細は、4-7 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。

初期ディスパッチャ・プロセス数の計算

使用中のオペレーティング・システムについて、プロセスあたり可能な接続数がわかったら、インスタンス起動時に作成するディスパッチャ・プロセスの最初の数を、ネットワーク・プロトコルごとに、次の計算式を使用して計算してください。

$$\text{ディスパッチャ数} = \text{CEIL} \left(\frac{\text{同時接続セッション数の最大値}}{\text{ディスパッチャごとの接続数}} \right)$$

注意: この場合、ディスパッチャごとの接続数の値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、TCP/IP を介して 900 人のユーザーが同時に接続し、SPX を介して 600 人のユーザーが接続し、プロセスあたり 255 の接続をサポートするシステムがあるとします。この場合、MTS_DISPATCHERS パラメータは、次のように設定します。

```
MTS_DISPATCHERS = "(PROTOCOL=TCP) (DISPATCHERS=4)"
MTS_DISPATCHERS = "(PROTOCOL=SPX) (DISPATCHERS=3)"
```

例

例 1 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。

```
MTS_DISPATCHERS="(ADDRESS=(PARTIAL=TRUE)(PROTOCOL=TCP)\n(HOST=144.25.16.201))(DISPATCHERS=2)"
```

これにより、2 つのディスパッチャが起動され、HOST=144.25.16.201 でリスニングしますが、それらは、ディスパッチャにアクセス可能なカードであることが必要です。

例 2 ディスパッチャの正確な位置を知るためには、次のように PORT を追加します。

```
MTS_DISPATCHERS="(ADDRESS=(PARTIAL=TRUE)(PROTOCOL=TCP)\n(HOST=144.25.16.201)(PORT=5000))(DISPATCHERS=1)"\nMTS_DISPATCHERS="(ADDRESS=(PARTIAL=TRUE)(PROTOCOL=TCP)\n(HOST=144.25.16.201)(PORT=5001))(DISPATCHERS=1)"
```

注意： 複数の MTS_DISPATCHERS を INIT.ORA ファイルに指定できませんが、それらは互いに隣接していることが必要です。また、MTS_DISPATCHERS はデフォルトで 1 に設定されます。

サーバー・プロセスの変更

ここでは、インスタンスの起動後にできる変更について説明します。この項のトピックは、次のとおりです。

- [共有サーバー・プロセスの最小数の変更](#)
- [ディスパッチャ・プロセスの追加と削除](#)

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 コマンド 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。

Oracle は最終的に、共有サーバー数が指定した最小限度を超えると、アイドル状態のサーバーを終了させます。

MTS_SERVERS を 0 に設定した場合、Oracle は現行のサーバーすべてがアイドル状態になると、これらのサーバーをすべて終了させ、MTS_SERVERS の値を大きくしない限り新しくサーバーを起動しません。つまり、MTS_SERVERS を 0 に設定することによって、マルチスレッド・サーバーを効率的に使用禁止にすることができます。

共有サーバー・プロセスの最小数を制御するには、ALTER SYSTEM 権限を持っていることが必要です。

次の文は、共有サーバー・プロセスの数を 2 に設定しています。

```
ALTER SYSTEM SET MTS_SERVERS = 2
```

ディスパッチャ・プロセスの追加と削除

インスタンスの中のディスパッチャ・プロセスの数を制御できます。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を V\$QUEUE、V\$DISPATCHER および V\$DISPATCHER_RATE ビューが示している場合は、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。ディスパッチャの数が MTS_MAX_DISPATCHERS に等しくなるまで、新しいディスパッチャを起動できます。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と、パフォーマンスを改善できます。

ディスパッチャ・プロセスの数を変更するには、SQL コマンド ALTER SYSTEM を使用します。特定のプロトコルに対するディスパッチャ数を変更しても、他のプロトコルのディスパッチャに影響は及びません。

MTS_DISPATCHERS パラメータに指定したプロトコル用に新しいディスパッチャ・プロセスを開始する方法と、新しい MTS_DISPATCHERS 構成を追加する方法があります。したがって、ディスパッチャがあるプロトコル用にはディスパッチャを追加し、現在はディスパッチャがないプロトコルの場合はディスパッチャを起動できます。

特定のプロトコル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。最終的には、MTS_DISPATCHERS に指定した制限に達するまでディスパッチャを終了します。

ディスパッチャ・プロセスの数を制御するには、ALTER SYSTEM 権限を持っていることが必要です。

次の例は、SPX プロトコル用のディスパッチャ・プロセスを追加する方法を示しています（ここでは、これまで MTS_DISPATCHER 構成が 1 つしかなかったものとします）。

```
ALTER SYSTEM  
  SET MTS_DISPATCHERS = '(INDEX=1) (PROTOCOL=SPX)';
```

関連項目：マルチスレッド・サーバーのチューニングの詳細は、『Oracle8i チューニング』を参照してください。

Oracle プロセスの追跡

Oracle のインスタンスでは、多数のバックグラウンド・プロセスを実行できます。可能であれば、これらのプロセスを追跡するようにしてください。ここではこれらのプロセスの追跡方法を説明します。この項のトピックは、次のとおりです。

- Oracle インスタンスのプロセスの監視
- トレース・ファイル、ALERT ファイル、バックグラウンド・プロセス
- チェックポイント・プロセスの起動

関連項目 : Oracle プロセスのチューニングの詳細は、『Oracle8i チューニング』を参照してください。

Oracle インスタンスのプロセスの監視

モニターを使用して、データベース・アクティビティおよびリソースの使用率を追跡できます。同時に複数のモニターを操作できます。表 4-1 は、Oracle プロセスの追跡に役立つ Enterprise Manager モニターを示しています。

表 4-1 Enterprise Manager モニター

モニター名	説明
プロセス	プロセス・モニターは、クライアント / サーバー・プロセス、ユーザー・プロセス、サーバー・プロセス、バックグラウンド・プロセスなど、その時点で現行のデータベース・インスタンスを介してデータベースにアクセスしているすべての Oracle プロセスに関する情報を要約します。
セッション	セッション・モニターは、接続されている各 Oracle セッションのセッション ID とその状態を示します。

ロックの監視

表 4-2 に、インスタンス内で進行中のトランザクションについてのロック情報を監視する 2 つの方法を示します。

表 4-2 Oracle の監視機能

モニター名	説明
Enterprise Manager モニター	Enterprise Manager/GUI のモニター機能では、インスタンスのロック情報を表示する 2 つのモニター（ロック・モニターとラッチ・モニター）を提供します。
UTLLOCKT.SQL	UTLLOCKT.SQL スクリプトは、ツリー構造方式で簡単なロック待機グラフを表示します。スクリプトは、Enterprise Manager や SQL*Plus など、非定型の問合せツールを使用して、システム内のロックを待機しているセッションとロックを保持しているセッションを出力します。このスクリプト・ファイルの位置は、オペレーティング・システムによって違います。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 CATBLOCK.SQL を実行する必要があります。）

動的パフォーマンス表の監視

次のビューは、動的パフォーマンス表に対して作成されるものであり、Oracle インスタンス・プロセスの監視に役立ちます。

ビュー (モニター) 名	説明
V\$CIRCUIT	ディスパッチャおよびサーバーを介したユーザー接続である、仮想回路についての情報が含まれています。
V\$QUEUE	マルチスレッド・メッセージ・キューについての情報が含まれています。
V\$DISPATCHER	ディスパッチャ・プロセスについての情報が含まれています。
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\$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\$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\$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\$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。
V\$SYSSTAT	システム統計情報が含まれています。

次に示すのは、動的パフォーマンス表の 1 つである V\$DISPATCHER の一般的な問合せです。出力には、システム内の各ディスパッチャ・プロセスに対する処理負荷が表示されます。

```
SELECT (busy/(busy + idle)) * 100 "% OF TIME BUSY"
       FROM v$dispatcher;
```

オペレーティング・システム・バックグラウンド・プロセスからの Oracle バックグラウンド・プロセスの識別

Oracle には、1 台のコンピュータ上で多くの Oracle データベースを同時に実行する場合に、インスタンスのプロセスを命名するメカニズムがあります。バックグラウンド・プロセス名には、インスタンス識別子によって接頭辞が付けられ、各インスタンスに対するプロセスの集合を識別します。

たとえば、TEST というインスタンスには、次の名前のバックグラウンド・プロセスが存在します。

- ORA_TEST_DBWR
- ORA_TEST_LGWR
- ORA_TEST_SMON
- ORA_TEST_PMON
- ORA_TEST_RECO
- ORA_TEST_LCK0
- ORA_TEST_ARCH
- ORA_TEST_D000
- ORA_TEST_S000
- ORA_TEST_S001

関連項目：ビューと動的パフォーマンス表の詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

インスタンス識別子および Oracle のプロセス名の書式の詳細は、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、ALERT ファイル、バックグラウンド・プロセス

各サーバー・プロセスとバックグラウンド・プロセスは、対応付けられたトレース・ファイルに書き込むことができます。内部エラーがプロセスによって検出されると、エラーに関する情報がトレース・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、データベース管理者用であり、その他の情報はオラクル社カスタマ・サポート用です。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調整にも使用されます。

ALERT ファイルは特殊なトレース・ファイルです。データベースの ALERT ファイルは、次のものを含むメッセージとエラーの履歴ログです。

- 発生したすべての内部エラー（ORA-600）、ブロック障害エラー（ORA-1578）、デッドロック・エラー（ORA-60）
- 管理操作、たとえば、CREATE/ALTER/DROP DATABASE/TABLESPACE/ROLLBACK SEGMENT の各 SQL 文と STARTUP、SHUTDOWN、ARCHIVE LOG
- 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いくつかのメッセージとエラー
- スナップショットの自動リフレッシュ中に発生したエラー
- データベースとインスタンスの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・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）、これらの特別な操作のログを保持するために、ALERT ファイルを使用します。操作が成功すると、タイムスタンプとともに「completed（完了）」というメッセージが、ALERT ファイルに書き込まれます。

トレース・ファイルの使用

ALERT ファイルとインスタンスのその他のトレース・ファイルを定期的にチェックして、バックグラウンド・プロセスでエラーを発見したかどうかを確認できます。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 ALERT 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、ただちに解決すべきメディアまたは I/O の問題が発生しています。

他の重要な統計に加えて初期化パラメータの値も ALERT ファイルに書き込まれます。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（ただし、中断はしない）、Oracle は、インスタンスが起動してから、インスタンスに同時に接続されたセッションの最高数を ALERT ファイルに書き込みます。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・ファイルの位置指定

バックグラウンド・プロセスに対するすべてのトレース・ファイルと ALERT 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宛先に書き込まれます。サーバー・プロセスに対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宛先に書き込まれます。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、ファイルを書き込んでいるプロセスの名前が含まれます（LGWR、RECO など）。

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（ALERT 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、いくつかのオペレーティング・システム・ブロックとして設定されます。ALERT ファイルのサイズを制御するために、不要になったファイルは手動で削除してください。そうしないと、Oracle はファイルを追加し続けます。インスタンスの実行中に ALERT ファイルを削除しても問題はありませんが、最初に ALERT ファイルのアーカイブ・コピーを作成しておくといでしょう。

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。ただし、初期化パラメータ SQL_TRACE が TRUE に設定されている場合にのみ、トレース・ファイルはサーバー・プロセスのために書き込まれます（内部エラー中にも書き込まれます）。

SQL_TRACE の現行値にかかわらず、各セッションは、SQL コマンド ALTER SESSION に SET SQL_TRACE パラメータを指定することにより、対応付けられたサーバー・プロセスに対しトレース・ロギングを使用可能にしたり、使用禁止にしたりできます。

```
ALTER SESSION SET SQL_TRACE TRUE;
```

マルチスレッド・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・プロセスへのルートが指定されています。また、そのセッションがトレースを使用可能にしている場合（または、エラーが見つかった場合）に限り、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に対する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。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す可能性があるため、統計を収集するときのみ、この機能を使用可能にしてください。

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。

ALTER SESSION コマンドの詳細は、『Oracle8i SQL リファレンス』を参照してください。

チェックポイント・プロセスの起動

チェックポイント・プロセス（CKPT）が使用可能でない場合には、最新のチェックポイントを反映するために、ログ・ライター・プロセス（LGWR）で、すべての制御ファイルとデータ・ファイルのヘッダーを更新する必要があります。チェックポイントを完了するまでの所要時間を短縮するには（特にデータベースが多くのデータ・ファイルから構成されているとき）、データベースのパラメータ・ファイル内で CHECKPOINT_PROCESS パラメータを TRUE に設定して、CKPT バックグラウンド・プロセスを使用可能にしてください。（デフォルト値は FALSE です。）

パラレル問合せオプションのプロセスの管理

ここでは、パラレル問合せオプションによって Oracle がどのようにパラレル処理を実行するかを説明します。この構成では、Oracle によって特定のタイプの SQL 文の処理作業を複数の問合せサーバー・プロセスに配分できます。この章の構成は、次のとおりです。

- [問合せサーバーの管理](#)
- [問合せサーバー・プロセスの数の変化](#)

関連項目：パラレル問合せオプションの詳細は、『Oracle8i チューニング』を参照してください。

問合せサーバーの管理

インスタンスを起動すると、Oracle Server はどの問合せコーディネータでも使用できる問合せサーバー・プロセスのプールを作成します。Oracle Server がインスタンスの起動時に作成する問合せサーバー・プロセスの数は、初期化パラメータ PARALLEL_MIN_SERVERS で指定します。

問合せサーバー・プロセスは、その実行フェーズを通して 1 つの文に対応付けられた状態になっています。文が完全に処理されると、その文の問合せサーバー・プロセスが他の文を処理できるようになります。問合せコーディネータ・プロセスは、結果として生成されるデータを、その文を発行したユーザー・プロセスに戻します。

問合せサーバー・プロセスの数の変化

インスタンスによって同時に処理される SQL 文のボリュームの変化が非常に大きい場合は、Oracle Server はこのボリューム変化に対応するようにプール中の問合せサーバー・プロセスの数を自動的に変更します。

このボリュームが増えると、Oracle Server は自動的に問合せサーバー・プロセスを追加作成して、入力文を処理します。インスタンス用の問合せサーバー・プロセスの最大数は、初期化パラメータ `PARALLEL_MAX_SERVERS` で指定します。

後でこのボリュームが減ると、Oracle Server は初期化パラメータ `PARALLEL_SERVER_IDLE_TIME` で指定した時間間隔のアイドル状態になった場合、問合せサーバー・プロセスを終了させます。問合せサーバー・プロセスがアイドル状態となっていた時間に関係なく、Oracle Server がプールのサイズを `PARALLEL_MIN_SERVERS` の値より小さくすることはありません。

プール中のすべての問合せサーバーが使用され、最大数の問合せサーバーが起動されると、結果として、問合せコーディネータ・プロセスが文を処理します。

関連項目： インスタンスの問合せサーバーのプールの監視方法と、初期化パラメータの適切な値を決める方法の詳細は、『Oracle8i チューニング』を参照してください。

外部プロシージャのプロセスの管理

C 関数の共有ライブラリを Oracle データベースからコールしたい場合もあります。ここでは、それらの外部プロシージャをコールするための環境を設定する方法について説明します。

注意： 必須ではありませんが、これらの作業はできるだけインストール時に行ってください。

データベース管理者が適切なライブラリについての実行権限をアプリケーションの開発者に付与し、そのアプリケーションの開発者が外部プロシージャを作成し、特定の外部プロシージャについての実行権限を他のユーザーに付与します。

外部プロシージャをコールするための環境を設定する手順

1. `tnsnames.ora` ファイルを編集して、リスナー・プロセス（および、その後 `EXTPROC` プロセス）に接続するための項目を追加します。

2. *listener.ora* ファイルを編集して、「外部プロシージャ・リスナー」のための項目を追加します。
3. 外部プロシージャを排他的に処理するための別個のリスナー・プロセスを開始します。
4. リスナーによって生成された EXTPROC プロセスはリスナーのオペレーティング・システム権限を継承するため、別個のリスナー・プロセスの権限を制限することをお勧めします。そのプロセスに対して、データベース・ファイルまたは Oracle Server のアドレス・スペースへの読み込みまたは書き込みの許可を与えないようにしてください。

また、この別個のリスナー・プロセスの所有者は「oracle」（サーバーの実行可能ファイルおよびデータベース・ファイルのデフォルト所有者）にしないでください。
5. まだインストールされていない場合は、**extproc** 実行可能ファイルを \$ORACLE_HOME/bin に格納します。

外部ライブラリ（DLL ファイル）は、必ず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）から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変換されず、外部プロシージャが失敗する原因となる場合があります。

tnsnames.ora のサンプル・エントリ

tnsnames.ora に記述する、外部プロシージャ・リスナーのためのエントリのサンプルを次に示します。

```
extproc_connection_data = (DESCRIPTION =
    (ADDRESS = (PROTOCOL=IPC)
        (KEY=extproc_key)
    )
    (CONNECT_DATA = (SID = extproc_agent)
    )
)
```

この例およびすべての外部プロシージャのコールにおいて、エントリ名 **extproc_connection_data** は変更できません。ここに示されているとおりに入力してください。指定するキー（この場合は **extproc_key**）は、*listener.ora* ファイルで指定した KEY と一致するようにします。また、指定する SID 名（この場合は **extproc_agent**）は、*listener.ora* ファイルで指定した SID_NAME エントリと一致するようにします。

listener.ora のサンプル・エントリ

listener.ora に記述する、外部プロシージャのためのエントリのサンプルを次に示します。

```
EXTERNAL_PROCEDURE_LISTENER =

(ADDRESS_LIST =
    (ADDRESS = (PROTOCOL=ipc)
        (KEY=extproc_key)
    )
)
...
```

```

SID_LIST_EXTERNAL_PROCEDURE_LISTENER =

(SID_LIST =
  (SID_DESC = (SID_NAME=extproc_agent)
               (ORACLE_HOME=/oracle)
               (PROGRAM=extproc)
            )
)

```

この例で、PROGRAM は **extproc** でなければならず、変更はできません。ここに示されているとおりに入力してください。SID_NAME は、*tnsnames.ora* ファイル内の SID 名と一致させます。ORACLE_HOME は、Oracle ソフトウェアがインストールされているディレクトリに設定する必要があります。実行可能ファイル **extproc** は、\$ORACLE_HOME/bin に存在するようにします。

関連項目 : 外部プロシージャの詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。

セッションの停止

状況によっては、現行のユーザー・セッションを停止する場合があります。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終了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。この項のトピックは、次のとおりです。

- [停止するセッションの識別](#)
- [アクティブなセッションの停止](#)
- [アクティブでないセッションの停止](#)

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）がただちに解除され、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、SID 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。セッションの索引（SID）とシリアル番号を識別するために、V\$SESSION 動的パフォーマンス表を問い合せてください。

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#
FROM v$session
WHERE username = 'JWARD';
```

SID	SERIAL#	STATUS
7	15	ACTIVE
12	63	INACTIVE

セッションは、Oracle に対して SQL コールを実行しているときは ACTIVE です。セッションは、Oracle に対して SQL コールを実行していないときは INACTIVE です。

関連項目：セッションの状態値の詳細は、『Oracle8i チューニング』を参照してください。

アクティブなセッションの停止

停止時にユーザー・セッションが Oracle へ SQL コールを実行している（ACTIVE である）場合、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: your session has been killed
```

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戻します。

```
ORA-01012: not logged on
```

アクティブ・セッションを中断できない（たとえば、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）場合、その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、停止するまで、セッションはすべてのリソースを保持します。さらに、セッションを停止するために ALTER SYSTEM 文を発行するセッションは、そのセッションが停止されるまで 60 秒待機します。中断できなかった操作が 1 分間続くと、ALTER SYSTEM 文の発行者は、セッションが停止したことを示す「マーク付」を伝えるメッセージを受け取ります。そのようなマーク付きのセッションでは、V\$SESSION の状態（STATUS）に "KILLED" が表示され、サーバー（SERVER）には "PSEUDO" 以外の値が示されます。

アクティブでないセッションの停止

セッションが停止時に、Oracle に対して SQL コールを実行していない（INACTIVE である）場合、ORA-00028 メッセージはただちには戻されません。このメッセージは、ユーザーが後で停止されたセッションを使用するときまで戻されません。

アクティブでない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。停止され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する V\$SESSION から削除され、ORA-00028 メッセージが表示されます。

次の例では、管理者がアクティブでないセッションを停止する場合を示します。

```
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.

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。この章のトピックは、次のとおりです。

- [制御ファイルのガイドライン](#)
- [制御ファイルの作成](#)
- [制御ファイルの作成後の問題解決](#)
- [制御ファイルの削除](#)

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。この項のトピックは、次のとおりです。

- [制御ファイルの命名](#)
- [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](#)
- [制御ファイルのサイズ管理](#)

制御ファイルの命名

データベースのパラメータ・ファイル内の `CONTROL_FILES` 初期化パラメータによって、制御ファイルの名前を割り当てます。`CONTROL_FILES` には、1 つ以上の制御ファイルの名前をカンマで区切って指定します。インスタンス起動時に、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制御ファイルをすべてメンテナンスします。

Oracle は、データベースの稼働中に `CONTROL_FILES` パラメータに指定されたすべてのファイルに書き込みます。

異なるディスク上での制御ファイルの多重化

どの Oracle のデータベースでも、制御ファイルは複数にし、それぞれを別々のディスクに格納してください。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停止させてください。ディスク・ドライブが修復されると、破損した制御ファイルは完全な制御ファイルのコピーを使用して復元され、インスタンスが再起動できるようになります。つまり、メディア回復は必要ありません。

多重制御ファイルの動作

次に示す項目は、多重制御ファイルの動作について説明したものです。

- 2 つ以上のファイル名をデータベースのパラメータ・ファイルの初期化パラメータ `CONTROL_FILES` に指定する。
- `CONTROL_FILES` パラメータ・リストの最初のファイルのみが、データベースの稼働中に Oracle Server によって読み込まれる。
- データベースの稼働中に制御ファイルのどれかが使用不可能になった場合、インスタンスは作動できなくなり、異常終了する。

複数の制御ファイルを持つことの唯一の欠点は、制御ファイルを更新するすべての操作（データ・ファイルの追加やデータベースのチェックポイントの実行など）に要する時間が少し長くなることです。ただし、この違いは通常それほど重要ではなく（特に複数の同時書込みが可能なオペレーティング・システムの場合）、単一制御ファイルの方がよいという裏付けにはなりません。

注意： オラクル社では、各種ディスク上にデータベースの 2 つ以上の制御ファイルを作成しておくことを強くお勧めしています。

制御ファイルの適切な配置

制御ファイルのコピーはそれぞれ異なるディスク・ドライブに格納する必要があります。さらに、オンライン REDO ログを多重化している場合は、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しなければなりません。このような配置をすることによって、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を少なくします。

制御ファイルのサイズ管理

制御ファイルのサイズは、主に、対応するデータベースを作成した CREATE DATABASE 文のパラメータ MAXDATAFILES、MAXLOGFILES、MAXLOGMEMBERS、MAXLOGHISTORY、MAXINSTANCES に設定された値によって決まります。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は、オペレーティング・システムによって異なります。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。

制御ファイルの作成

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するもので、次のような内容が含まれています。

- データベース名
- 対応付けられたデータベースおよびオンライン 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

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、制御ファイルのコピーは、データベースの作成時に少なくとも 1 つ作成されることになっています。一部のオペレーティング・システムでは、Oracle が複数のコピーを作成するものもあります。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勧めします。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したりする場合には、制御ファイルを作成する必要があります。

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。この項のトピックは、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と制御ファイルの改名 / 再配置](#)
- [新しい制御ファイル](#)
- [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、データベース作成時に使用されるパラメータ・ファイル内の `CONTROL_FILES` パラメータに、1 つ以上の制御ファイル名を指定することによって作成されます。`CONTROL_FILES` パラメータのファイル名は省略しないで指定してください。ファイル名の仕様は、オペレーティング・システムによって異なります。

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` 文に `CONTROLFILE REUSE` パラメータを指定しなければなりません。そうしないとエラーが発生します。なお、古い制御ファイルと新しい制御ファイルのサイズが異なる場合には、`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 のリリース間で異なることがあります。また、制御ファイルで指定したファイル数に変更がある場合も異なります。制御ファイルのサイズには、`MAXLOGFILES`、`MAXLOGMEMBERS`、`MAXLOGHISTORY`、`MAXDATAFILES` および `MAXINSTANCES` などの構成パラメータが影響します。

データベースの作成前に `CONTROL_FILES` にファイル名を指定しなかった場合、Oracle はデフォルトのファイル名を使用します。デフォルトのファイル名もオペレーティング・システムによって異なります。

`CONTROL_FILES` 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。

制御ファイルの追加コピーの作成と制御ファイルの改名 / 再配置

既存の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することによって、新しい制御ファイルを追加できます。

同じように、既存の制御ファイルを新しい名前や位置にコピーし、制御ファイルのリスト内でそのファイルの名前を変更することによって、制御ファイルを改名できます。

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. データベースのパラメータ・ファイルの CONTROL_FILES パラメータを編集して、新たな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

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。これは、次のような状況で実行します。

- 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。
- データベースの名前、MAXLOGFILES および MAXLOGMEMBERS、MAXLOGHISTORY、MAXDATAFILES、MAXINSTANCES など、CREATE DATABASE 文で最初に指定した永久的なデータベース設定の 1 つを変更する場合。

たとえば、分散環境において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、その名前の変更が必要になることがあります。または、前述のパラメータの 1 つで、最初の設定が小さすぎる場合にもその値を変更する必要があります。

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 ('logfile1A', 'logfile1B') SIZE 50K,
GROUP 2 ('logfile2A', 'logfile2B') SIZE 50K
NORESETLOGS
DATAFILE 'datafile1' SIZE 3M, 'datafile2' SIZE 5M
MAXLOGFILES 50
MAXLOGMEMBERS 3
MAXDATAFILES 200
MAXINSTANCES 6
ARCHIVELOG;
```

警告： CREATE CONTROLFILE 文は、指定したデータ・ファイルとオンライン REDO ログ・ファイルに損傷を与える可能性があります。ファイル名を指定しないと、そのファイル内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こともあります。このコマンドを使用するときには十分注意し、必ずこの後の部分の手順に従ってください。

関連項目： CREATE CONTROLFILE 文の詳細は、『Oracle8i 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

ここでは、新しい制御ファイルを作成する手順について示します。

新しい制御ファイルを作成する手順

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。

データベース・バックアップの推奨事項に従っていれば、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あるはずです。

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、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・ファイルのすべてを検出してください。新しい制御ファイルが作成されると、ステップ 5 で指定したファイル以外は回復できません。さらに、SYSTEM 表領域を構成するファイルを 1 つでも指定しないと、データベースを回復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、できる限り通常の手順でデータベースを停止してください。最後の手段としてだけ、IMMEDIATE オプションまたは ABORT オプションを使用してください。

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。
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には、新しい制御ファイルの作成時に RESETLOGS オプションを選択してください。この場合には、失った REDO ログ（ステップ 8）から回復する必要があります。データベースを

改名している場合は、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。

7. データベースのパラメータ・ファイルを編集します。

データベースのパラメータ・ファイルを編集して、ステップ 5 とステップ 6 で作成した制御ファイルすべて（バックアップの制御ファイルを除く）を、CONTROL_FILES パラメータに指定してください。

8. 必要に応じて、データベースを回復させます。

回復の一部として制御ファイルを作成している場合には、データベースを回復してください。新しい制御ファイルを NORESETLOGS オプションを使用して作成した場合（ステップ 5）、完全な、クローズ状態のデータベース回復操作によってデータベースを回復できます。

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済みの REDO ログ、またはデータ・ファイルが失われた場合は、これらのファイルの回復手順に従ってください。

9. データベースをオープンします。

次の方法の 1 つを使用して、データベースをオープンしてください。

- 回復を実行しなかった場合、通常の手順でデータベースをオープンします。
- ステップ 8 でクローズ状態のデータベースの完全な回復操作を実行した場合は、データベースを起動します。
- 制御ファイルの作成時に RESETLOGS を指定した場合は、ALTER DATABASE 文に RESETLOGS を指定して使用します。

これでデータベースはオープンされ、使用可能になっています。

関連項目：次のトピックの詳細は、『Oracle8i 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・ファイルの回復
- クローズ状態のデータベースの回復

制御ファイルの作成後の問題解決

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。この項のトピックは、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 でのエラーの処理](#)

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、ALERT ログをチェックして、Oracle がデータ・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、制御ファイルには示されていないなど）を検出したかどうかを調べてください。

データ・ディクショナリの中にはデータ・ファイルが存在していて、新しい制御ファイルには示されていない場合、Oracle は MISSINGnnnn（この場合、nnnn は 10 進数のファイル番号）で制御ファイル内にプレースホルダ・エントリを作成します。MISSINGnnnn では、オフラインであること、およびメディア回復を必要とすることを示すフラグが制御ファイル内で立てられます。

次の 2 つの場合に限っては、MISSINGnnnn に対応する実際のデータ・ファイルを指すように MISSINGnnnn を改名することによって、実際のデータ・ファイルにアクセスできるようになります。

ケース 1: NORESETLOGS オプションを指定した CREATE CONTROLFILE 文を使用して新しい制御ファイルを作成することにより、RESETLOGS オプションを使用しないでデータベースをオープンすることが可能な場合。これができるのは、すべてのオンライン REDO ログが使用可能な場合だけです。

ケース 2: CREATE CONTROLFILE 文で RESETLOGS オプションを使用する必要がある、RESETLOGS オプションを使用してデータベースを強制的にオープンしたが、MISSINGnnnn に対応する実際のデータ・ファイルは読込み専用または通常のアフラインであった場合。

一方、RESETLOGS オプションを使用してデータベースをオープンする必要があったときに、MISSINGnnnn が読込み専用または通常のアフラインでないデータ・ファイルに対応している場合は、改名操作を使用してもデータ・ファイルへのアクセスを可能にはできません（データ・ファイルに必要なメディア回復が RESETLOGS の結果により不可能であるため）。この場合、そのデータ・ファイルが入っている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。どちらの場合も、Oracle は ALERT ファイルにどんな状態が検出されたかを通知するメッセージを書き込みます。

CREATE CONTROLFILE でのエラーの処理

新しい制御ファイルを作成後に、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、Oracle がエラー（通常、ORA-01173、ORA-01176、ORA-01177、ORA-01215、ORA-01216 のどれか）を送信した場合、最も可能性の高い原因は、CREATE CONTROLFILE 文に指定し忘れたファイルがあるか、またはリストに示されていないファイルを指定したということです。この場合、ステップ 3 でバックアップしたファイルを復元し、正しいファイル名を使用してステップ 4 から手順を実行し直してください。

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には、その制御ファイルを削除する必要があります。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しなければなりません。

1. データベースを停止します。
2. データベースのパラメータ・ファイルの CONTROL_FILES パラメータを編集して、古い制御ファイルの名前を削除します。
3. データベースを再起動します。

警告： この操作では、不要な制御ファイルがディスクから物理的に削除されるわけではありません。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。

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。この章のトピックは、次のとおりです。

- オンライン REDO ログ
- 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・メンバーの改名および再配置
- 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 ログに関する情報のリスト

関連項目 : Oracle Parallel Server 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については、『Oracle8i Parallel Server 概要および管理』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回復に及ぼす影響については、『Oracle8i チューニング』を参照してください。

オンライン REDO ログ

オンライン REDO ログは、回復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、その発生時に格納する 2 つ以上の事前割当てファイルからなっています。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。

注意： オンライン REDO ログのバックアップを作成することはお薦めできません。

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 つのみなので、スレッドは 1 つしか存在しません。ただし、Oracle Parallel Server 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、Oracle Parallel Server を使用していないときに、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、コマンド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。

関連項目： Oracle Parallel Server とともにオンライン REDO ログを構成する方法の詳細は、『Oracle8i Parallel Server 概要および管理』を参照してください。

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、それぞれが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である変更ベクトル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 レコードを生成します。

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回復させるときには、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 SGA の REDO ログ・バッファに入れられ、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 では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・バッファが満杯になるか、別のトランザクションがコミットされると、一部の REDO レコードは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。必要であれば、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

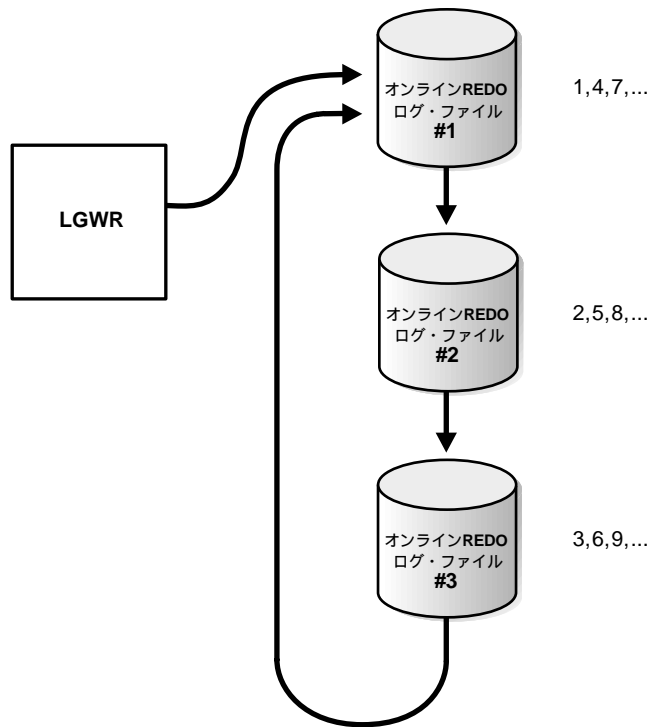
データベースのオンライン REDO ログは、2 つ以上のオンライン REDO ログ・ファイルからなっています。Oracle は、一方のファイルのアーカイブ中にも (ARCHIVELOG モード時)、他方のファイルが常に書込み可能であることを保証するために、最低 2 つのファイルが必要とします。

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・ファイルが満杯になると、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・ファイルが満杯になると、LGWR は最初のオンライン REDO ログ・ファイルに戻って書き込みを行い、再び循環を開始します。[図 6-1](#) 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。

満杯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 で再使用できます。

- アーカイブが使用禁止になっている場合 (NOARCHIVELOG モード)、満杯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)、満杯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 6-1 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（現行）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 つだけ使用して、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 が書き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。

インスタンス回復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。インスタンス回復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、ARCn の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・ファイルの再使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、最後のオンライン REDO ログ・ファイルが満杯になると、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き込みが継続します。

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・ファイルが完全に満杯になり、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、常にログ・スイッチが発生します。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。

オンラインまたはアーカイブされた各 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回復中に、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・ファイルを昇順で正しく適用します。

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を説明します。この項のトピックは、次のとおりです。

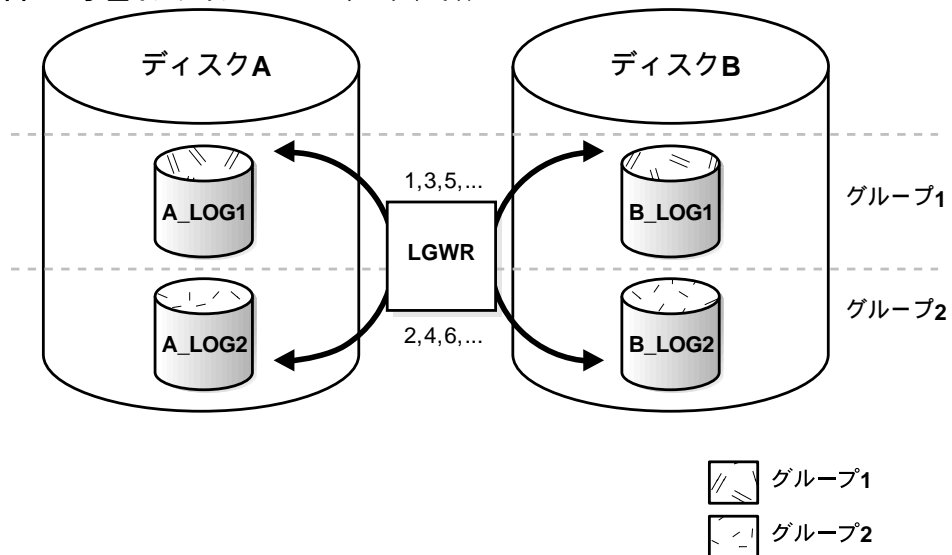
- オンライン REDO ログ・ファイルの多重化
- オンライン REDO ログ・メンバーを別々のディスクに配置
- オンライン REDO ログ・メンバーのサイズ設定
- 適切なオンライン REDO ログ・ファイル数の選択

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、一箇所の REDO ログの障害は問題になりません。

注意： 回復が必要になった場合に、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恐れがあるため、REDO ログ・ファイルを多重化することをお勧めします。

図 6-2 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 6-2 で、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、まったく同一でなければなりません。

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ることに注目してください。図 6-2 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「stale (無効)」を示すマークを付け、アクセス不能ファイルの問題を示すエラー・メッセージを、LGWR トレース・ファイルとデータベースのアラート・ファイルに書き込みます。特定のオンライン REDO ログ・メンバーが使用不能な場合、LGWR の動作はその原因に応じて異なります。

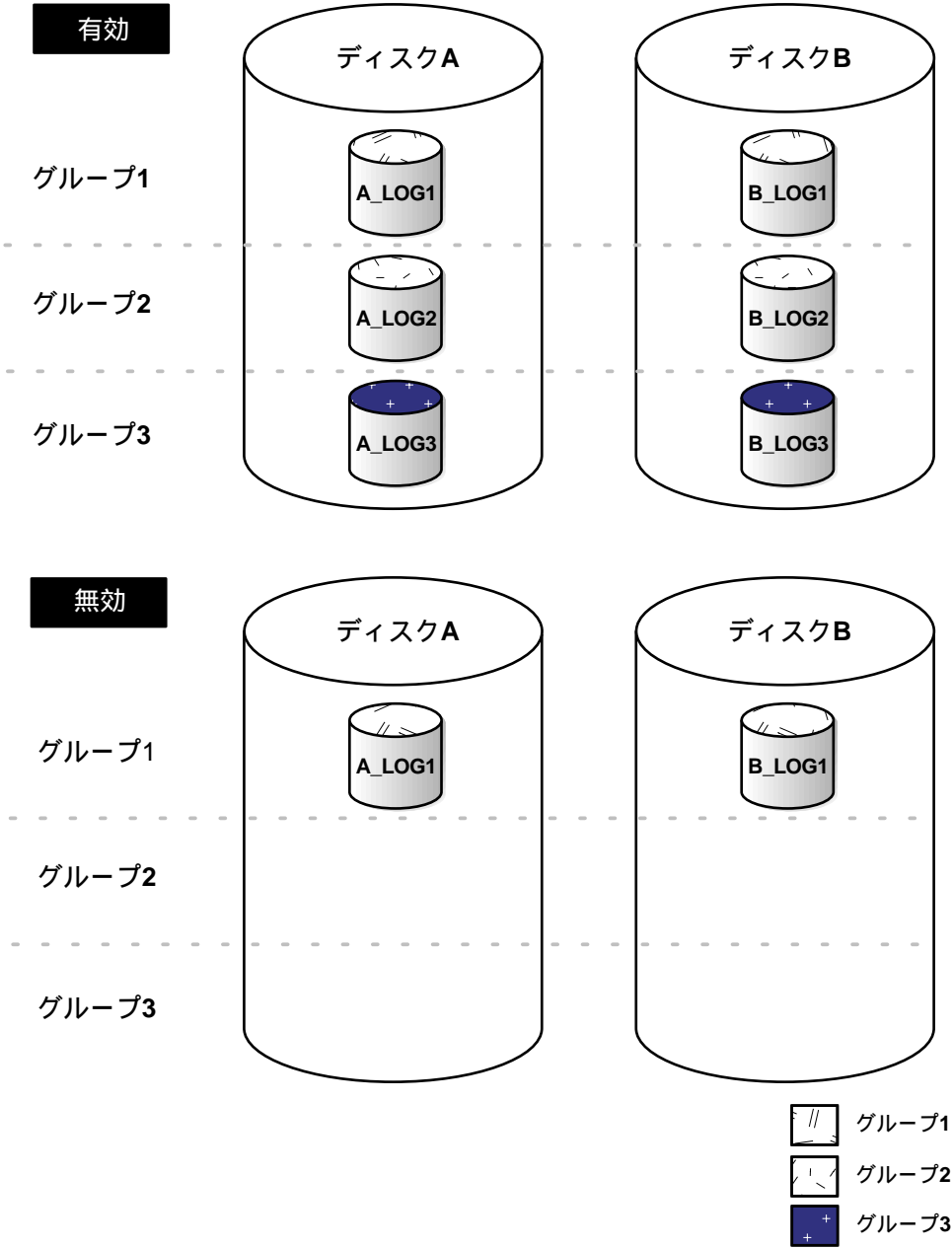
状況	動作
LGWR がグループ内の最低 1 つのメンバーに正常に書き込める場合	通常どおり書き込みが進行します。LGWR はグループのうち使用可能なメンバーに書き込むのみで、使用不能なメンバーは無視します。
グループをアーカイブする必要があるため、LGWR がログ・スイッチの発生時に次のグループにアクセスできない場合	データベース操作は、グループが使用可能になるか、グループがアーカイブされるまで一時停止します。
メディア障害のため、ログ・スイッチの発生時に、LGWR が次のグループのどのメンバーにもアクセスできない場合	Oracle はエラーを戻し、データベース・インスタンスは停止します。この場合は、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回復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損失した REDO ログ（この例では、現行でないログ）を回避した場合、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、メディア回復の必要はありません。単にアクセスできない REDO ログ・グループを削除するのみですみます。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でなければ、ログを削除できません。
LGWR が書き込み中に、突然グループのすべてのメンバーにアクセスできなくなった場合	Oracle はエラーを戻し、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、メディア回復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、メディア回復は不要です。この場合は、ドライブをオンに戻して、Oracle にインスタンス回復を実行させるのみですみます。

有効な構成と無効な構成

一箇所にあるオンライン REDO ログの障害に備えて、多重オンライン REDO ログを対称型にしておくのが理想的な方法です。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、多重オンライン REDO ログ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に備えるものです。

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 6-3 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 つしかないので無効です。

図 6-3 多重オンライン REDO ログの有効な構成と無効な構成



オンライン REDO ログ・メンバーを別々のディスクに配置

多重オンライン REDO ログを設定する場合、グループのメンバーを異なるディスク上に配置します。このようにすれば、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は引き続き LGWR にアクセスでき、インスタンスは機能し続けることができます。

REDO ログをアーカイブする場合には、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配置してください。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。このようにすれば、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、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必要があります。

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、『Oracle8i 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。

オンライン REDO ログ・メンバーのサイズ設定

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、満杯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てください。たとえば、テープに満杯のオンライン REDO ログ・グループを1つだけアーカイブし、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とよいでしょう。

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でなければなりません。異なるグループのメンバーは異なるサイズを持つことができます。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には、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：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。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。

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、異なる構成をテストすることです。最適の構成では、グループ数は LGWR が REDO ログ情報を書き込むのを妨げない最小の値です。

データベース・インスタンスに必要なグループが2つだけの場合もあります。また、LGWR が使用可能な再生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、現在のオンライン 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。チェックポイントが完了していなかったり、グループがアーカイブされていないために、LGWR が頻繁に待機する必要があるというメッセージが示された場合は、グループを追加してください。

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3つのパラメータは、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。これは、データベースあたりのオンライン REDO ログ・ファイルの最大グループ数を決めるものです。グループの値は 1 ~ MAXLOGFILES です。この上限値を置き換える唯一の方法は、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検討することが重要です。CREATE DATABASE 文に MAXLOGFILES パラメータが指定されていない場合には、Oracle はオペレーティング・システムのデフォルト値を使用します。
- LOG_FILES 初期化パラメータ（パラメータ・ファイル内）。このパラメータを使用すると、現行インスタンスの存続中、オンライン REDO ログ・ファイルのグループの最大数を一時的に小さくできます。ただし、LOG_FILES パラメータを使用して MAXLOGFILES を書き換えて最大数を大きくすることはできません。データベースのパラメータ・ファイルに LOG_FILES パラメータが指定されていない場合、Oracle はオペレーティング・システム固有のデフォルト値を使用します。
- CREATE DATABASE 文の MAXLOGMEMBERS パラメータ。これはグループに含まれるメンバーの最大値を決めるものです。MAXLOGFILES と同じように、この上限値を置き換える唯一の方法は、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、この上限値を検討することが重要です。CREATE DATABASE 文に MAXLOGMEMBERS パラメータが指定されていない場合には、Oracle はオペレーティング・システムのデフォルト値を使用します。

関連項目：MAXLOGFILES パラメータ、MAXLOGMEMBERS パラメータおよび LOG_FILES 初期化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。

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・ファイルのグループとメンバーは、データベースの作成時と作成後の両方で作成できます。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。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 システム権限を持っている必要があります。

場合によっては、オンライン REDO ログ・ファイルのグループまたはメンバーの追加作成が必要なことがあります。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。データベースは、MAXLOGFILES グループまで持つことができます。

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 パラメータを指定します。

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： 新しいログ・メンバーのファイル名は、オペレーティング・システム・ファイルを作成すべき位置を含め、省略しないで指定します。そうしないと、ファイルはオペレーティング・システムによって異なるデータベース・サーバーのデフォルトのディレクトリに作成されます。なお、既存のオペレーティング・システム・ファイルを再使用する場合、ファイル・サイズを指定する必要はありません。

ALTER DATABASE 文で ADD LOGFILE オプションとともに GROUP オプションを使用すると、グループを識別する番号を指定できます。

```
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、グループ番号は 1 ~ MAXLOGFILES の数値でなければなりません。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けれども、（ディスク障害などにより）1 つ以上のグ

ループのメンバーが削除されたために完全ではない場合などです。このような場合、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 LOGFILE 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;
```

なお、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 コマンドを使用するときには、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： 新しいログ・メンバーのファイル名は、オペレーティング・システム・ファイルを作成すべき位置を含め、省略しないで指定します。そうしないと、ファイルはデータベース・サーバーのデフォルトのディレクトリに作成されます。

オンライン REDO ログ・メンバーの改名および再配置

オンライン REDO ログ・メンバーを改名して、その位置を変更できます。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べき場合に、この手順が必要となります。

オンライン REDO ログ・メンバーを改名するには、ALTER DATABASE システム権限が必要です。さらに、ファイルを希望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。

オンライン REDO ログ・メンバーを改名する前に、新たなオンライン REDO ログ・ファイルがすでに存在することを確認してください。

警告： 次に示す手順では、データベースの制御ファイル内の内部ファイル・ポインタが修正されるだけで、オペレーティング・システム・ファイルを物理的に改名したり、作成するわけではありません。オペレーティング・システムを使用して、既存のオンライン REDO ログ・ファイルを新しい位置にコピーしてください。

オンライン REDO ログ・メンバーを改名する手順

1. データベースのバックアップを作成します。

オンライン REDO ログ・メンバーの改名や再配置のように、データベースの構造を変更する場合は、その操作の実行時に発生する可能性のある問題に備えて、事前にデータベース（制御ファイルを含む）のバックアップを作成します。

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・メンバーのようなオペレーティング・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・システムのマニュアルを参照してください。

注意： オペレーティング・システムのコマンドを実行すると、既存の SQL*Plus を終了しないでファイルをコピーできます。HOST コマンドを使用してください。

3. オンライン REDO ログ・メンバーを改名します。

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・ファイルを改名します。

4. 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、次にデータベースがオープンされたときに有効となります。データベースをオープンするには、現行インスタンスを停止する必要がある場合（データベースが現行インスタンスによって事前にオープンされていた場合）と、現行インスタンスを使用してデータベースをオープンするだけでよい場合があります。

5. 制御ファイルのバックアップを作成します。

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・ファイルを改名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。

次に示すのは、オンライン REDO ログ・メンバーを改名する例です。ただし、次のことを前提とします。

- インスタンスに対して、データベースは現在マウントされ、クローズされていること。
- ログ・ファイルが、diska および diskb という 2 つのディスク上にあること。
- オンライン REDO ログが多重化されていること。この場合、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ること。
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。

`diska` 上のファイル `/diska/logs/log1a.rdo` および `/diska/logs/log2a.rdo` を、
`diskc` 上の新しいファイル `/diskc/logs/log1c.rdo` および `/diskc/logs/log2c.rdo` にコピーする必要があります。

```
ALTER DATABASE RENAME FILE '/diska/logs/log1a.rdo', '/diska/logs/log2a.rdo'  
TO '/diskc/logs/log1c.rdo', '/diskc/logs/log2c.rdo';
```

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する必要があります。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必要がある場合などです。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があります。たとえば、ディスク障害が発生した場合、アクセスできないファイルに書込みがなされないように、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する必要があります。これ以外にも、適切ではない位置にファイルを格納した場合など、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。

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 システム権限を持っていなければなりません。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の数にかかわらず、インスタンスには、少なくともオンライン REDO ログ・ファイルのグループが 2 つ必要です。(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、アクティブ・グループでない場合にのみ削除できます。アクティブ・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。6-16 ページの「ログ・スイッチの強制」を参照してください。
- 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、SQL*Plus の ARCHIVE LOG 文に LIST パラメータを指定して実行します。

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・グループを削除しても、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。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、データベースに関連する制御ファイルが更新されます。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てください。

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 ログが一時的に非対称になっても問題はないか検討してください。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 ログに対して単一点障害の発生する可能性をなくしてください。
- 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、他のメンバーが有効になるまで、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 STALE になります。この使用されなくなったログ・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、もう1度有効になります。
- 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、SQL*Plus の ARCHIVE LOG コマンドに LIST パラメータを指定して実行します。

アクティブでない特定のオンライン REDO ログ・メンバーを削除するには、SQL の ALTER DATABASE 文で DROP LOGFILE MEMBER 句を指定して実行します。

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に、そのオペレーティング・システム・ファイルがディスクから削除されるわけではありません。つまり、データベース構造からメンバーを削除するために、対応するデータベースの制御ファイルが更新されます。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てください。

関連項目：アクティブ・グループのメンバー削除の詳細は、6-16 ページの「ログ・スイッチの強制」を参照してください。

SQL*Plus のコマンド構文の詳細は、『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。

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・グループへの書き込みを停止して、別のログ・グループへの書き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・グループが満杯になると、ログ・スイッチが自動的に発生します。

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アクティブでない状態にし、オンライン REDO ログのメンテナンス操作に使用できます。たとえば、現在アクティブなグループを削除する場合、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ないことがあります。また、現在アクティブなグループのメンバーが完全に満杯になる前に、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、満杯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・ファイルを使用して構成する場合に有効です。

ログ・スイッチを強制するには、ALTER SYSTEM 権限を持っている必要があります。また、SQL 文 ALTER SYSTEM で SWITCH LOGFILE オプションを指定します。

次の文は、ログ・スイッチを強制します。

```
ALTER SYSTEM SWITCH LOGFILE;
```

関連項目：Oracle Parallel Server でログ・スイッチを強制する方法については、『Oracle8i Parallel Server 概要および管理』を参照してください。

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。REDO ログ・ブロックがチェックできるようにするには、初期化パラメータ LOG_BLOCK_CHECKSUM を TRUE に設定します。LOG_BLOCK_CHECKSUM のデフォルト値は、FALSE です。

REDO ログ・ブロックをチェックできるようにすると、Oracle は現行のログに書き込まれた各 REDO ログ・ブロックのチェックサムを算出します。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・ブロック内の破損データを検出します。Oracle は、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、およびブロックが回復処理中にアーカイブ済みログから読み込まれるときに、そのブロックを検証しようとしています。

Oracle は、REDO ログ・ブロックをアーカイブしようとしているときにブロック内で破損データを検出すると、グループの中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、アーカイブ処理ができません。

オンライン REDO ログ・ファイルの初期化

REDO ログ・ブロックのチェックができる場合、Oracle は各ブロックをアーカイブする前に検証します。特定の REDO ログ・ブロックがグループのすべてのメンバーで破損している場合は、アーカイブが停止します。最終的にはすべての REDO ログが埋め込まれ、アーカイブが再開できるまで、データベース・アクティビティが停止します。

この状況では、破損した REDO ログがアーカイブされないように、SQL 文 ALTER DATABASE ... CLEAR LOGFILE を使用してその REDO ログを初期化してください。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。

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 UNARCHIVED LOGFILE GROUP 3;
```

制限

REDO ログ・ファイルは、アーカイブされていてもアーカイブされていなくても初期化できます。ただし、アーカイブされていない場合は、ALTER DATABASE CLEAR LOGFILE 文にキーワード UNARCHIVED を含める必要があります。

バックアップの回復に必要なログ・ファイルを初期化すると、そのバックアップからの回復処理ができなくなります。Oracle は、警告ログに、そのバックアップからの回復処理ができないことを記述する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。

オフラインの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 文に UNRECOVERABLE DATAFILE 句を指定して使用してください。

オフラインの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なくなります。表領域を削除するか、不完全回復を実行する必要があります。表領域が正常にオフラインになる場合、回復は不要なので注意してください。

関連項目 : ALTER DATABASE 文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

オンライン REDO ログに関する情報のリスト

データベースのオンライン REDO ログに関する情報を参照するには、ビュー V\$LOG、V\$LOGFILE および V\$THREAD を使用してください。V\$THREAD ビューは Parallel Server 管理者にとって特に有効です。

次の問合せでは、Parallel Server を使用しないデータベースのオンライン REDO ログに関する情報が戻ります。

```
SELECT group#, bytes, members FROM sys.v$log;
```

GROUP#	BYTES	MEMBERS
-----	-----	-----
1	81920	2
2	81920	2

グループのすべてのメンバーの名前を表示するには、次のような問合せを使用してください。

```
SELECT * FROM sys.v$logfile
WHERE group# = 2;
```

GROUP#	STATUS	MEMBER
-----	-----	-----
2		LOG2A
2	STALE	LOG2B
2		LOG2C

メンバーの STATUS が空白の場合、そのファイルは使用中です。

アーカイブ済み 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。この章のトピックは、次のとおりです。

- [アーカイブ済み REDO ログ](#)
- [NOARCHIVELOG モードと ARCHIVELOG モードの選択](#)
- [アーカイブの切替え](#)
- [アーカイブ先の指定](#)
- [ログ送信モードの指定](#)
- [アーカイブ先の障害管理](#)
- [アーカイブ・パフォーマンスの調整](#)
- [アーカイブ済み REDO ログ情報の表示](#)
- [LogMiner を使用したオンラインおよびアーカイブ済み REDO ログの分析](#)

関連項目 : Oracle を Parallel Server で使用している環境でのアーカイブ方法の詳細は、『Oracle8i Parallel Server 概要および管理』を参照してください。

アーカイブ済み REDO ログ

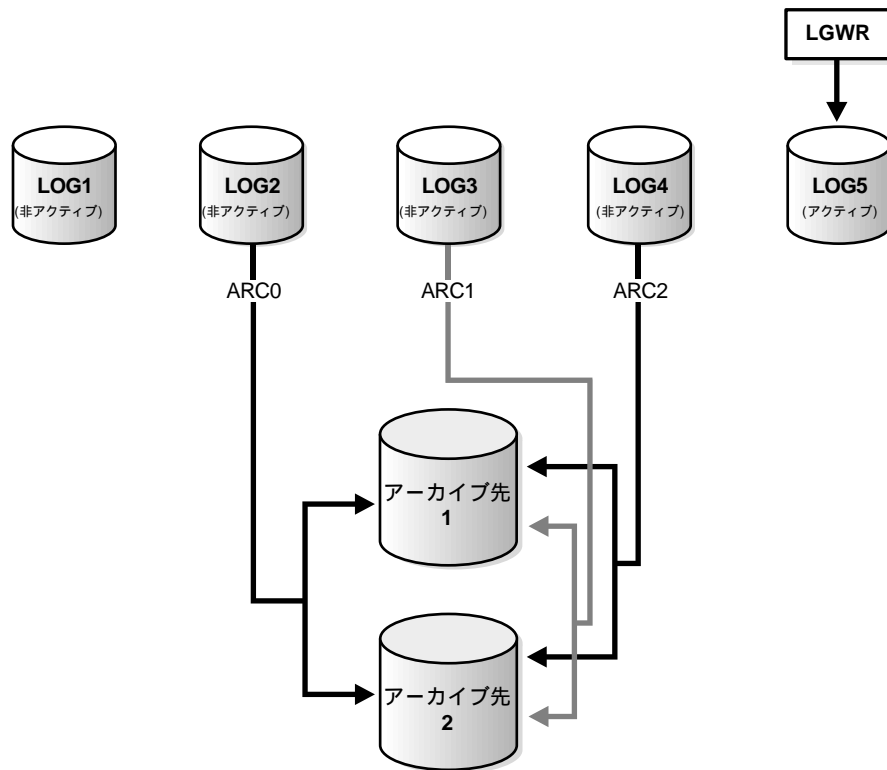
Oracle では、満杯になったオンライン REDO ログ・ファイルのグループを「アーカイブ済み REDO ログ」として、1 つ以上のオフライン接続先に保存できます。「アーカイブ」とは、オンライン REDO ログをアーカイブ済み REDO ログに変更するプロセスです。アーカイブ操作は、バックグラウンド・プロセス ARC_n によって自動化されます。アーカイブ・ログは、次の操作に使用できます。

- データベースを回復する操作。
- スタンバイ・データベースを更新する操作。
- LogMiner ユーティリティを介してデータベースの履歴情報を取得する操作。

アーカイブ済み REDO ログ・ファイルは、オンライン REDO ログ・グループのうち満杯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、グループの同一メンバーに存在する 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保たれます。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、 ARC_n では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 が破損した場合も、 ARC_n ではそれと同一の B_LOG1 をアーカイブできます。

アーカイブを使用可能にしている場合、LGWR では、オンライン REDO ログ・グループがアーカイブされるまで再利用つまり上書きすることができません。したがって、アーカイブ済み 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。[図 7-1](#) は、 ARC_n による REDO ログのアーカイブ方法を示しています。

図 7-1 オンライン REDO ログのアーカイブ



警告： 現行のオンライン・ログはコピーしないでください。コピーして復元したオンライン・ログは、REDO スレッドの最後に現われます。スレッド内で追加の REDO が生成されている可能性があるため、REDO ログのコピーを使用して回復しようとすると、誤って REDO スレッドの終わりが検出され、途中で処理が終了し、データベースが破損する恐れがあります。現行のオンライン・ログの内容のバックアップを作成する最善の方法は、必ずアーカイブし、アーカイブ・ログのバックアップを作成することです。

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。この項のトピックは、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 ログのアーカイブは使用禁止になります。データベースの制御ファイルは、満杯のグループをアーカイブする必要がないことを示します。したがって、満杯のグループがログ・スイッチの発生後に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。

満杯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中の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。満杯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、管理作業が増えるので注意してください。

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。オンライン REDO ログのグループに格納されている、データベースへの変更のうち、最新の変更だけがインスタンスの回復に有効です。つまり、NOARCHIVELOG モードでは、最後に全体データベース・バックアップしたときの状態にしかデータベースを復元（回復ではなく）できません。それ以降のトランザクションは回復できません。

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復元に使用できるのは、データベースがクローズされているときに作成された全体データベース・バックアップだけです。したがって、NOARCHIVELOG モードでデータベースを操作する場合は、全体データベース・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、オンライン REDO ログのアーカイブが使用可能になります。データベースの制御ファイルは、満杯のオンライン 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。満杯になったグループは、REDO 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。

満杯になったグループをアーカイブすると、次のような長所が得られます。

分散データベース回復 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、調整式分散データベース回復を実行できます。ただし、分散データベース内のデータベースのどれかが 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回復は、NOARCHIVELOG モードで操作しているデータベースの最新の全体バックアップによって制限されます。

関連項目 : REDO ログ・ブロックをアーカイブするときに、これらのブロックを検証するように Oracle を構成することもできます。詳細は、6-16 ページの「[REDO ログ・ファイル内のブロックの検証](#)」を参照してください。

アーカイブの切替え

ここでは、さまざまなアーカイブ作業について説明します。この項のトピックは、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化](#)
- [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 : 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。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。

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 ログ情報を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。

データベース・アーカイブ・モードの変更

アーカイブ・モードを NOARCHIVELOG モードまたは ARCHIVELOG モードに切り替えるには、ARCHIVELOG オプションまたは NOARCHIVELOG オプションを指定して SQL 文 ALTER DATABASE を使用します。次の文は、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。

```
ALTER DATABASE ARCHIVELOG;
```

データベースのアーカイブ・モードを切り替える前に、次の操作を実行します。

1. データベース・インスタンスを停止します。

データベースがオープンされている場合、データベースのアーカイブ・モードを切り替える前にクローズしてディスマウントし、対応するインスタンスを停止する必要があります。メディア回復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。

データベースを大幅に変更する前に、データベース・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。

3. 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。

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。

注意： Oracle Parallel Server を使用している場合、データベースのアーカイブ・モードを切り替えるには、1 つのインスタンスを使用してデータベースを排他的にマウントしなければなりません。

4. データベースのアーカイブ・モードを切り替えます。

ALTER DATABASE コマンドを使用してデータベースのアーカイブ・モードを切り替えた後、データベースをオープンして通常の操作を実行できます。ARCHIVELOG モードに切り替えた場合、アーカイブ・オプションも設定する必要があります。満杯のオンライン REDO ログ・ファイル・グループが Oracle により自動アーカイブされるようにするかどうかを決めてください。

オペレーティング・システムによっては、満杯のグループをアーカイブするために、さらにいくつかの手順を実行しなければならない場合があります。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。

関連項目： Oracle Parallel Server を使用している場合のアーカイブ・モードの切替えの詳細は、『Oracle8i Parallel Server 概要および管理』を参照してください。

自動アーカイブの使用可能化

オペレーティング・システムが許可する場合には、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。このオプションを使用すると、グループが満杯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。このような便利さから、自動アーカイブはアーカイブ方法の選択肢となっています。

インスタンスの起動の前後に自動アーカイブを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、管理者権限を使用して Oracle に接続してください。

警告： Oracle は、データベースも ARCHIVELOG モードでなければ、ログ・ファイルを自動的にアーカイブしません。

関連項目： 自動アーカイブを使用可能にするときは、アーカイブ済みの REDO ログのアーカイブ先とファイル名の形式を必ず指定します。7-10 ページの「[アーカイブ先の指定](#)」を参照してください。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。7-9 ページの「[手動アーカイブの実行](#)」を参照してください。

インスタンス起動時の自動アーカイブの使用可能化

インスタンスを起動するたびに、満杯の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化

現行インスタンスを停止しないで、満杯のオンライン 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 を指定した SQL 文 ALTER SYSTEM パラメータを使用します。この場合、アーカイブ先を任意に指定できます。

```
ALTER SYSTEM ARCHIVE LOG START;
```

ALTER SYSTEM を使用すると、自動アーカイブを使用可能にするためにインスタンスを停止させる必要がなくなります。ただし、自動アーカイブが使用可能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。この設定では、自動アーカイブは使用可能あるいは使用不能場合があります。

自動アーカイブの使用禁止

いつでもオンライン REDO ログ・グループの自動アーカイブを使用禁止にできます。ただし、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しなければなりません。データベースが ARCHIVELOG モードで操作され、自動アーカイブが使用禁止になっている場合、オンライン REDO ログ・ファイルのグループがすべて満杯になり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使用して REDO ログ・エントリの書込みを続行できなくなります。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されます。

インスタンスの起動時または起動後に、自動アーカイブを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、管理者権限および ALTER SYSTEM 権限で接続している必要があります。

インスタンス起動時の自動アーカイブの使用禁止

データベース・インスタンスが起動するたびに、満杯のオンライン 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、次のようにデータベースのパラメータ・ファイルの LOG_ARCHIVE_START パラメータを FALSE に設定します。

```
LOG_ARCHIVE_START=FALSE
```

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禁止

現行インスタンスを停止せずに満杯のオンライン 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 は現行のグループのアーカイブを完了しますが、次の満杯のオンライン 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。この設定では、自動アーカイブは使用可能あるいは使用不能場合があります。

手動アーカイブの実行

データベースを ARCHIVELOG モードで操作する場合は、満杯になったオンライン REDO ログ・ファイルのアクティブでないグループをアーカイブする必要があります。自動アーカイブの設定にかかわらず、オンライン REDO ログのグループを手動でアーカイブできます。

- 自動アーカイブが使用可能でない場合、満杯のオンライン 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。オンライン REDO ログ・グループがすべて満杯になってアーカイブされない場合、LGWR はオンライン REDO ログ・メンバーの非アクティブなグループを再使用して、REDO ログ・エントリの書込みを続行できません。したがって、必要なアーカイブが実行されるまでデータベース操作は一時的に停止されます。
- 自動アーカイブは使用可能になっているが、満杯のオンライン REDO ログ・メンバーのアクティブでないグループを別の位置にもう一度アーカイブする場合、手動アーカイブを使用できます。ただし、手動アーカイブの終了前に、REDO ログ・グループを再使用してファイルを上書きするように、インスタンスによって決められることがあるので注

意してください。このような場合、Oracle は ALERT ファイルにエラー・メッセージを入れます。

満杯のオンライン 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、SQL 文 ALTER SYSTEM 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;
```

関連項目 : Oracle Parallel Server を使用している場合を除き、手動または自動アーカイブを実行する際にスレッドを指定する必要はありません。詳細は、『Oracle8i Parallel Server 概要および管理』を参照してください。

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ます。さまざまなアーカイブ先の状態と、固定ビューを使用してアーカイブ情報にアクセスする操作に慣れておく必要があります。

アーカイブ先の指定

ログ用に単一のアーカイブ先を作成するか、多重化するか、つまり複数のアーカイブ先を指定するかどうかを決定する必要があります。

次の初期化パラメータを設定し、プライマリ・データベースのアーカイブ・ログに使用するロケーション数を指定します。

パラメータ	ホスト	例
LOG_ARCHIVE_DEST_ <i>n</i> (<i>n</i> は 1 ~ 5 の整数)	リモート または ローカル	LOG_ARCHIVE_DEST_1 = 'LOCATION = / disk1/arc' LOG_ARCHIVE_DEST_2 = 'SERVICE = standby1'
LOG_ARCHIVE_DEST および LOG_ARCHIVE_DUPLEX_DEST	ローカル のみ	LOG_ARCHIVE_DEST = /oracle/arc LOG_ARCHIVE_DUPLEX_DEST = /bak

最初の方法では、LOG_ARCHIVE_DEST_ *n* パラメータ（この場合、*n* は 1 ~ 5 の整数）を使用して、1 ~ 5 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では、特定のアーカイブ先を個々に識別します。たとえば、LOG_ARCHIVE_DEST_1、LOG_ARCHIVE_DEST_2 となります。

LOG_ARCHIVE_DEST_ *n* のロケーションは、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ロケーション	LOG_ARCHIVE_DEST_1= 'LOCATION=/arc'
SERVICE	Net8 サービス名を介したりモート・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てください。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するように、サービス名には対応するデータベース SID が必要であることに注意してください。

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、必要であれば 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。

LOG_ARCHIVE_DEST_n を使用してアーカイブ済み REDO ログのアーカイブ先を設定する手順

1. SQL*Plus を使用してデータベースを停止します。

```
SHUTDOWN IMMEDIATE;
```

2. LOG_ARCHIVE_DEST_n パラメータを編集し、1 ~ 5 のアーカイブ先を指定します。LOCATION キーワードでは、OS 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 ファイルから有効なネット・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 パラメータを編集します。ファイル名にログ順序番号を挿入するには、%s を使用し、スレッド番号を挿入するには、%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
```

たとえば、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
```

LOG_ARCHIVE_DEST および LOG_ARCHIVE_DUPLEX_DEST を使用してアーカイブ済み REDO ログのアーカイブ先を設定する手順

1. SQL*Plus を使用してデータベースを停止します。

```
SHUTDOWN IMMEDIATE;
```

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 コマンドを使用して、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挿入するには、%s を使用し、スレッド番号を挿入するには、%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_100.arc
```

関連項目：スタンバイ・データベースへのアーカイブ方法の詳細は、『Oracle8i 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。

アーカイブ先の状態の理解

LOG_ARCHIVE_DEST_ *n* パラメータでは、指定されたアーカイブ先の状態が識別されます（この場合、*n* は 1 ~ 5 の整数です）。アーカイブ先パラメータには、ENABLE および DEFER という 2 つの値があります。ENABLE は Oracle がアーカイブ先を使用できることを示し、DEFER は使用できないことを示します。

各アーカイブ先は、次の 3 つの変動特性を持っています。

- **Valid/Invalid。** ディスクのロケーションまたはサービス名情報が指定されているかどうかを示します。

- *Enabled/Disabled*。Oracle がアーカイブ先情報を使用する必要があ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。

アーカイブ先には複数の状態が考えられます。各インスタンスの現行のアーカイブ先の状態情報を取得するには、V\$ARCHIVE_DEST ビューを問い合わせます。最後に入力したパラメータで指定したアーカイブ先にアクセスしますが、このアーカイブ先に完全なアーカイブ先データが含まれているとは限りません。

表 7-1 に、このビューに表示される状態情報を示します。

表 7-1 アーカイブ先の状態

VALID	ENABLED	ACTIVE	状態	意味
FALSE	N/A	N/A	INACTIVE	ユーザーはアーカイブ先情報を指定していない、または削除しました。
TRUE	TRUE	TRUE	VALID	ユーザーは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。
TRUE	TRUE	FALSE	ERROR	アーカイブ先ファイルの作成または書き込み中にエラーが発生しました。エラー・データを参照。
TRUE	FALSE	TRUE	DEFERRED	ユーザーはアーカイブ先を手動で一時的に使用禁止にしています。
TRUE	FALSE	FALSE	DISABLED	ユーザーは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。
N/A	N/A	N/A	BAD PARAM	パラメータ・エラーが発生した。エラー・データを参照。通常、この状態は LOG_ARCHIVE_START が設定されていない場合にのみ発生します。

関連項目 : V\$ARCHIVE_DEST およびアーカイブ先パラメータの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

ログ送信モードの指定

アーカイブ・ログをアーカイブ先に送信する場合、「通常アーカイブ送信」および「スタンバイ送信」という 2 つのモードがあります。通常送信では、ファイルはローカル・ディスク

に送信されます。スタンバイ送信では、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・データベースに送信されます。

通常送信モード

通常送信モードでは、アーカイブ先はデータベース・サーバーの別のディスク・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、短時間で完了するので、グループは LGWR に使用可能になります。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。

できれば、アーカイブ済み REDO ログ・ファイルとそれに対応するデータベース・バックアップを、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域メディアに永久的に移動しておきます。アーカイブ・ログは主としてデータベース回復に使用されるので、プライマリ・データベースに障害が発生した場合も、これらのログが安全かどうかを確認する必要があります。

スタンバイ送信モード

スタンバイ送信モードでは、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。

警告： スタンバイ・データベースはローカル・ディスク上でメンテナンスできますが、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、最大限の災害対策を講じることをお勧めします。

スタンバイ・データベースを「管理された回復モード」で操作している場合は、送信されたアーカイブ・ログを自動的に適用し、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保つことができます。

ファイルをスタンバイ・データベースに正常に送信するには、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・ロケーションを認識します。
- リモート・ファイル・サーバー（RFS）プロセスを使用してアーカイブ・ログを送信します。

各 ARC*n* プロセスは、スタンバイ・アーカイブ先ごとに対応する RFS を作成します。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。

Net8 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・ロケーションに送信できます。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 Net8 サービス名を指定します。SERVICE_NAME パラメータを使用してサービス名を設定すると、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

データベースの制御ファイルのログ履歴を正しく更新するように、サービス名には対応するデータベース SID が必要です。ご注意ください。

RFS プロセスは接続先ノード上で実行され、ARCn 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、ARCn は情報を RFS にプッシュし、RFS がそれをスタンバイ・データベースに送信します。

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。

- ARCn プロセスからのネットワーク I/O を消費します。
- STANDBY_ARCHIVE_DEST パラメータを使用して、スタンバイ・データベース上でファイル名を作成します。
- ログ・ファイルをリモート・サイトに移入します。
- スタンバイ・データベースの制御ファイルを (Recovery Manager で回復に使用できるように) 更新します。

アーカイブ済み REDO ログは、オリジナルのデータベースの完全な複製であるスタンバイ・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済み REDO ログで自動的に更新されます。

関連項目：スタンバイ・データベースの詳細は、『Oracle8i バックアップおよびリカバリ・ガイド』の関連する章を参照してください。

Net8 の詳細は、『Oracle8i Net8 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

アーカイブ先が障害を起こし、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。アーカイブ先の障害に関連する問題を最小限度に抑えるために、Oracle8i では次のことを指定できます。

- Oracle で正常にアーカイブしなければならないアーカイブ先の最小数
- ARCn が障害アーカイブ先への再アーカイブを試みる時期と頻度

正常なアーカイブ先の最小数の指定

オプション・パラメータ `LOG_ARCHIVE_MIN_SUCCEED_DEST=n` (n は 1 ~ 5 の整数) によって、Oracle がオンライン・ログ・ファイルを再使用するまでに REDO ログ・グループを正常にアーカイブしなければならない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を使用して、LGWR がオンライン・ログを上書きできるかどうか判断されます。

パラメータの設定を決めるときには、次の点に注意してください。

- アーカイブ先に `MANDATORY` を指定しなければ、`OPTIONAL` を指定したことになります。
- ローカル・アーカイブ先を最低 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、最低 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` パラメータの関連付けが失われます。
- `LOG_ARCHIVE_MIN_SUCCEED_DEST` には、アーカイブ先を超える値や、`MANDATORY` のアーカイブ先数と `OPTIONAL` のローカル・アーカイブ先数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` を指定した場合に、アーカイブ・ログがスタンバイ・サイトに送信されないままオンライン・ログが上書きされる場合は、ログを手動でスタンバイに送信する必要があります。

必要であれば、`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` を介して宣言されたアーカイブ先は必須です。
- `LOG_ARCHIVE_DUPLEX_DEST` を介し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

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。例 1 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 7-2 に、この例の LOG_ARCHIVE_MIN_SUCCEED_DEST=*n* に考えられる値を示します。

表 7-2 例 1 の LOG_ARCHIVE_MIN_SUCCEED_DEST の値

値	意味
1	OPTIONAL のアーカイブ先のうち最低 1 つへのアーカイブに成功した場合にのみ、Oracle はログ・ファイルを再使用できます。
2	OPTIONAL のアーカイブ先のうち最低 2 つへのアーカイブに成功した場合にのみ、Oracle はログ・ファイルを再使用できます。
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・ファイルを再使用できます。
4	エラー：値がアーカイブ先数を超えています。
5	エラー：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 MANDATORY に明示的に設定しない場合も、LOG_ARCHIVE_MIN_SUCCEED_DEST が 1、2、または 3 に設定されていれば、Oracle はこれらのロケーションに正常にアーカイブする必要があることを示しています。

例 2 では、次のような状況を考えます。

- MANDATORY のアーカイブ先を 2 つ指定する場合。
- OPTIONAL のアーカイブ先を 2 つ指定する場合。
- 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場合。

表 7-3 に、LOG_ARCHIVE_MIN_SUCCEED_DEST=*n* に考えられる値を示します。

表 7-3 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成功しなくても、ログ・ファイルを再使用できます。
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、ログを再使用できます。
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、ログを再使用できます。
5	エラー：値がアーカイブ先数を超過しています。

この例は、LOG_ARCHIVE_MIN_SUCCEED_DEST をアーカイブ先が少数になるように設定するかどうかに関係なく、MANDATORY として指定するアーカイブ先にアーカイブする必要があることを示しています。

関連項目：LOG_ARCHIVE_MIN_SUCCEED_DEST=*n* またはアーカイブ関連の他のパラメータについての追加情報は、『Oracle8i リファレンス・マニュアル』を参照してください。

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 エラーだけでなく、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リ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 0 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ければ、ARC*n* はエラー発生後にアーカイブ先をリオープンしません。

REOPEN は、再接続とアーカイブ・ログ送信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 は成功または失敗を試行し、その場合に 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、Oracle は正常にアーカイブできない場合に本番データベースを停止します。この使用例では、次の操作が必要です。

- 障害アーカイブ先に手動でアーカイブする操作。

- アーカイブ先を遅延させ、オプションとして指定するか、サービスを変更して、アーカイブ先を変更する操作。
- アーカイブ先を削除する操作。

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。

- $ARCn$ 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リオープンします。現行の操作中にリオープンすることはありません。常に先頭からログ・コピーを再試行します。
- REOPEN 時刻を指定するか、またはデフォルト値を受け入れた場合、 $ARCn$ 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、 $ARCn$ 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正常に反映されます。VALID および ENABLED 状態は変化しません。

アーカイブ・パフォーマンスの調整

ほとんどのデータベースでは、 $ARCn$ 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可能性もあります。 $ARCn$ が非常に高速な処理を実行すると、CPU サイクルがアーカイブに使われるため、 $ARCn$ の実行中は全体のシステム・パフォーマンスが低下する可能性があります。もう一方では、 $ARCn$ が極端に低速で実行すると、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。それらがアーカイブされるのを待機するために、すべての REDO ログ・グループが使用不可能になると、ボトルネックになる可能性があります。

アーカイブ操作を調整するには、次の方法を使用します。

- [複数の \$ARCn\$ プロセスの指定](#)
- [アーカイブ・バッファ・パラメータの設定](#)

関連項目：データベースのチューニングの詳細は、『Oracle8i チューニング』を参照してください。

複数の $ARCn$ プロセスの指定

データベース・インスタンスごとに、 $ARCn$ プロセスを最大 10 個まで指定します。起動時または実行時にパラメータ LOG_ARCHIVE_MAX_PROCESSES= n (n は 1 ~ 10 の整数) を設定し、複数処理機能を使用可能にしてください。デフォルトでは、このパラメータは 0 に設定されます。

LGWR は、現行の $ARCn$ 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ければ自動的に増やすので、このパラメータは初期の $ARCn$ プロセス数を指定したり、または現行の数を増減させるためのものです。

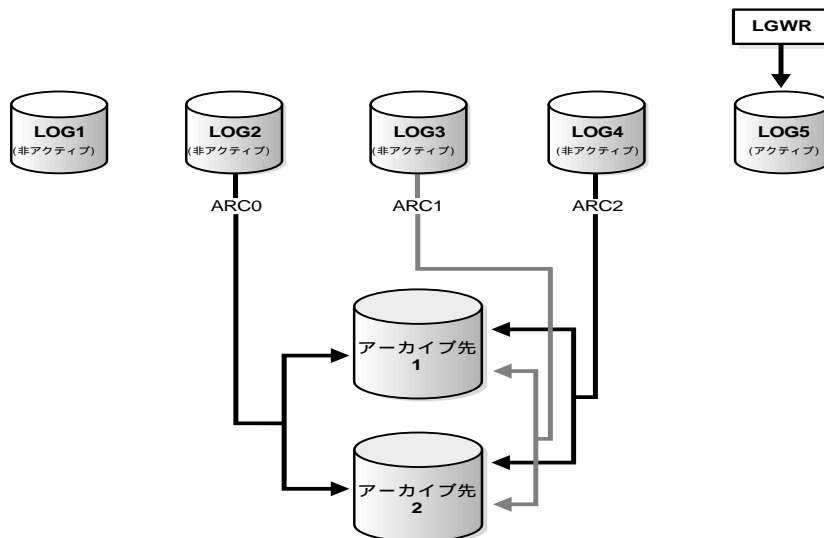
次の場合には、複数プロセスを作成すると特に有効です。

- 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

LGWR が複数のオンライン REDO ログ間で切り替える速度が、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速度より高速の場合は、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されます。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せず、指定された各アーカイブ先にアーカイブする必要があるので注意してください。

たとえば、5 個のオンライン REDO ログ・ファイルをメンテナンスする場合は、3 個の ARC_n プロセスを使用してインスタンスを起動することができます。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 プロセスは各種のアーカイブ先に同時に最高 3 個の非アクティブなログ・ファイルをアーカイブできます。図 7-3 のように、ARC_n の各インスタンスはログ・ファイルを 1 つずつ受け持ち、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 7-3 複数の ARCH プロセスの使用



アーカイブ・バッファ・パラメータの設定

ここでは、アーカイブ・バッファ初期化パラメータを調整に使用する作業について説明します。この項のトピックは、次のとおりです。

- システム・パフォーマンスへの影響を最小にする方法
- アーカイブを高速にする方法

アーカイブ操作は、ボトルネックとならずに可能な限り低速でアーカイブされるように、または実質的にシステム・パフォーマンスを低下させずに可能な限り高速でアーカイブされるように調整できます。そのためには、初期化パラメータ LOG_ARCHIVE_BUFFERS（アーカイブに割り当てられるバッファ数）と LOG_ARCHIVE_BUFFER_SIZE（各バッファのサイズ）を調整します。

注意： LOG_ARCHIVE_BUFFERS または LOG_ARCHIVE_BUFFER_SIZE の値を変更すると、新しい値は次回インスタンスを起動するときに有効になります。

システム・パフォーマンスへの影響を最小にする方法

システムが REDO ログを待機させずに、できる限り ARCn を低速にするには、最初にアーカイブ・バッファの数（LOG_ARCHIVE_BUFFERS）を 1 に設定し、各バッファのサイズ（LOG_ARCHIVE_BUFFER_SIZE）を可能な最大値に設定します。

ARCn の作動中にシステムのパフォーマンスが著しく低下する場合には、ARCn の実行時にシステム・パフォーマンスが低下しなくなるまで、LOG_ARCHIVE_BUFFER_SIZE を小さく調整してください。

注意： アーカイブをなるべく低速に設定するが、REDO ログ・ファイルを再使用できるようになる前にそれらがアーカイブされるまで Oracle が多く待たなければならないときは、REDO ログ・ファイル・グループを追加作成できます。追加のグループによって、グループを常に Oracle に対して使用可能にすることができます。

アーカイブを高速にする方法

アーカイブ操作のパフォーマンスを改善するには、複数のアーカイブ・バッファを使用して、ARCn プロセスに出力ログの書込みと同時にアーカイブ・ログを強制的に読み込ませます。LOG_ARCHIVE_BUFFERS を 2 に設定できますが、非常に高速なテープ・ドライブに対しては、3 以上に設定してもかまいません。アーカイブ・バッファのサイズは適当な数に設定し、それからシステム・パフォーマンスを損なわずに、アーカイブが望ましい速度になるまで大きくしてください。

関連項目：この最大値は、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 マニュアルを参照してください。LOG_ARCHIVE パラメータの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

アーカイブ済み REDO ログ情報の表示

アーカイブ済み REDO ログに関して役立つ情報を含む複数の固定ビューがあります。

固定ビュー	説明
V\$DATABASE	データベースが ARCHIVELOG モードであるか、NOARCHIVELOG モードであるかを識別します。
V\$ARCHIVED_LOG	制御ファイルからのアーカイブ・ログ履歴情報を表示します。リカバリ・カタログを使用すると、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、これらのアーカイブ先の現行の値、モードおよび状態の記述が表示されます。
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・カタログを使用すると、RC_BACKUP_REDOLOG ビューにも同様の情報が含まれます。
V\$LOG	データベースのすべてのオンライン REDO ログ・グループと、そのうちのどれをアーカイブする必要があるかが表示されます。
V\$LOG_HISTORY	アーカイブ済みのログや、各アーカイブ・ログの SCN 範囲など、ログ履歴情報が含まれます。

たとえば、次の問合せでは、どのオンライン REDO ログ・グループをアーカイブする必要があるかが表示されます。

```
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 ビューを問い合わせます。

```
SELECT log_mode FROM sys.v$database;
```

```
LOG_MODE
-----
NOARCHIVELOG
```

SQL 文 ARCHIVE LOG LIST でも、接続されているインスタンスのアーカイブ情報が表示されます。

```
ARCHIVE LOG LIST;
```

Database log mode	ARCHIVELOG
Automatic archival	ENABLED
Archive destination	<i>destination</i>
Oldest online log sequence	30
Next log sequence to archive	32
Current log sequence number	33

この表示は、現行インスタンスのアーカイブ済み 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になっています。
- アーカイブ済み REDO ログのアーカイブ先（オペレーティング・システム固有）。
- 満杯の最も古いオンライン REDO ログ・グループの順序番号は 30 です。
- 次にアーカイブされる満杯のオンライン REDO ログ・グループの順序番号は 32 です。
- 現行のオンライン REDO ログ・ファイルの順序番号は 33 です。

「Next log sequence to archive」以上、「Current log sequence number」未満の順序番号を持つ REDO ログ・グループを、すべてアーカイブする必要があります。この例は、順序番号が 32 のオンライン REDO ログ・グループをアーカイブする必要があることを示しています。

関連項目：データ・ディクショナリ・ビューの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

LogMiner を使用したオンラインおよびアーカイブ済み REDO ログの分析

Oracle ユーティリティの LogMiner を使用すると、オンライン REDO ログとアーカイブ済み REDO ログに含まれる情報を、選択条件に基づいて読むことができます。LogMiner のリレーショナル SQL インタフェースにより、データベースの完全な履歴ビューに直接アクセスでき、アーカイブ済み REDO ログ・ファイルを復元する必要がありません。

この項のトピックは、次のとおりです。

- [LogMiner の用途](#)
- [制限事項](#)
- [ディクショナリ・ファイルの作成](#)
- [分析する REDO ログ・ファイルの指定](#)
- [LogMiner の使用](#)
- [LogMiner の使用 : 使用例](#)

LogMiner の用途

LogMiner は、論理的な破損の識別とやり直しに特に役立ちます。LogMiner 処理では、REDO ログ・ファイルの内容が、データベースに対して実行された論理操作を表す SQL 文に変換されます。次に、V\$LOGMNR_CONTENTS ビューに、元の操作を表すように再度組み立てられた SQL 文 (SQL_REDO 列) と、操作をやり直すための対応する SQL 文 (SQL_UNDO 列) が表示されます。SQL_UNDO 文を適用し、データベースに対する元の変更をロールバックしてください。

また、V\$LOGMNR_CONTENTS ビューは次の目的で使用できます。

- 不完全回復の実行が必要となる時刻または SCN を特定し、データベースの論理的な破損が始まったと思われる時刻を判断する目的。
- 特定の表に対する変更の追跡。
- 特定のユーザーが行った変更の追跡。
- データ・アクセスのパターンのマップ。
- アーカイブされたデータを調整と容量計画に使用する目的。

関連項目 : LogMiner データ・ディクショナリ・ビューの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

制限事項

LogMiner の使用方法と互換性に関する要件は、次のとおりです。LogMiner では、次の操作のみが実行されます。

- Oracle バージョン 8.1 以上で動作します。
- 分析対象となるインスタンスと同じデータベース・キャラクタ・セットを使用し、同じハードウェア・プラットフォーム上で稼働しているバージョン 8.0 以上のデータベースからの REDO ログ・ファイルを分析します。
- PL/SQL パッケージによって作成されたディクショナリを使用して、REDO ログ・ファイルの内容を完全に分析します。このディクショナリにより、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す。
- 従来型の表に対する DML 操作情報を取得します。次の操作対象はサポートされていません。
- 索引構成表
 - クラスタ化表 / 索引
 - 非スカラー型
 - 連鎖行

ディクショナリ・ファイルの作成

LogMiner は、データベースがマウントされているかどうかに関係なく、Oracle インスタンス内で実行されます。LogMiner は、作成されたデータベースとファイル作成時刻を含む特殊なファイル、「ディクショナリ・ファイル」を使用します。このディクショナリ・ファイルは、必須ではありませんが、作成することをお勧めします。

ディクショナリ・ファイルがなければ、それと同等の SQL 文ではオブジェクト名に Oracle 内部オブジェクト ID が使用され、列値は 16 進データとして表されます。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'));
```

ディクショナリ・ファイルを作成するには、データベースをマウントし、ディクショナリ情報を外部ファイルに抽出します。ディクショナリ・ファイルは、分析するログ・ファイルを生成したのと同じデータベースから作成する必要があります。作成後は、そのディクショナリ・ファイルを使用して REDO ログを分析できます。

ディクショナリの作成時に指定する内容は、次のとおりです。

- ディクショナリ・ファイル名を指定するための `DICTIONARY_FILENAME`
- ファイル位置を指定するための `DICTIONARY_LOCATION`

Oracle8i データベースのディクショナリ・ファイルを作成する手順

1. `init.ora` パラメータ `UTL_FILE_DIR` を設定し、PL/SQL プロシージャで使用するディレクトリを必ず指定します。このパラメータを参照しなければ、プロシージャは失敗します。たとえば、`/oracle/logs` を使用するには次のように設定します。

```
UTL_FILE_DIR = /oracle/logs
```

2. SQL*Plus を使用して、ファイルの分析対象となるデータベースをマウントし、オープンします。たとえば、次のように入力します。

```
STARTUP
```

3. 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・パス名の両方を指定してください。このプロシージャによって、ログ・ファイルの分析に使用すべきディクショナリ・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/logs` 内でファイル `dictionary.ora` を作成します。

```
EXECUTE dbms_logmnr_d.build(  
  dictionary_filename => 'dictionary.ora',  
  dictionary_location => '/oracle/logs');
```

Oracle8 データベースのディクショナリ・ファイルを作成する手順

LogMiner はリリース 8.1 以上のデータベースでしか動作しませんが、リリース 8.0 データベースからの REDO ログの分析に使用できます。

1. O/S ユーティリティを使用して、Oracle8i データベースの `$ORACLE_HOME/rdbms/admin` ディレクトリにある `dbmslogmnrd.sql` スクリプトを、Oracle8 データベース内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/dbmslogmnrd.sql /8.0/oracle/rdbms/admin/dbmslogmnrd.sql
```

2. SQL*Plus を使用して、ファイルの分析対象となるデータベースをマウントし、オープンします。たとえば、次のように入力します。

```
STARTUP
```

3. コピーした `dbmslogmnrd.sql` スクリプトを 8.0 データベース上で実行し、`DBMS_LOGMNR_D` パッケージを作成します。たとえば、次のように入力します。

```
@dbmslogmnrd.sql
```

4. `init.ora` パラメータ `UTL_FILE_DIR` を設定し、PL/SQL パッケージで使用するディレクトリを指定します。このパラメータを参照しなければ、プロシージャは失敗します。たとえば、`/8.0/oracle/logs` を使用するには次のように設定します。

```
UTL_FILE_DIR = /8.0/oracle/logs
```

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・パス名の両方を指定してください。このプロシージャによって、ログ・ファイルの分析に使用すべきディクショナリ・ファイルが作成されます。たとえば、次のように入力して /8.0/oracle/logs 内でファイル dictionary.ora を作成します。

```
EXECUTE dbms_logmnr_d.build(  
dictionary_filename => 'dictionary.ora',  
dictionary_location => '/8.0/oracle/logs');
```

関連項目 : DBMS_LOGMNR_D の詳細は、『Oracle8i パッケージ・プロシージャ リファレンス』を参照してください。

分析する REDO ログ・ファイルの指定

ディクショナリ・ファイルの作成後に、REDO ログの分析を開始できます。最初のステップでは、ADD_LOGFILE プロシージャを使用して、分析するログ・ファイルを指定します。次の定数を使用します。

- NEW。新規リストを作成します。
- ADDFILE。リストに REDO ログを追加します。
- REMOVEFILE。リストから REDO ログを削除します。

LogMiner を使用する手順

1. データベースがマウントされているかどうかに関係なく、SQL*Plus を使用して Oracle インスタンスを起動します。たとえば、次のように入力します。

```
startup
```

2. DBMS_LOGMNR.ADD_LOGFILE プロシージャの実行時に NEW オプションを指定して、ログのリストを作成します。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。

```
execute dbms_logmnr.add_logfile(  
LogFileName => '/oracle/logs/log1.f',  
Options => dbms_logmnr.NEW);
```

3. 必要であれば、ADDFILE オプションを指定して、さらにログを追加します。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。

```
execute dbms_logmnr.add_logfile(  
LogFileName => '/oracle/logs/log2.f',  
Options => dbms_logmnr.ADDFILE);
```

4. 必要であれば、REMOVEFILE オプションを指定してログを削除します。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。

```
execute dbms_logmnr.add_logfile(  
  LogFileName => '/oracle/logs/log2.f',  
  Options => dbms_logmnr.REMOVEFILE);
```

関連品目 : DBMS_LOGMNR の詳細は、『Oracle8i パッケージ・プロシージャ リファレンス』を参照してください。

LogMiner の使用

ディレクトリ・ファイルを作成し、分析するログ・ファイルを指定した後は、LogMiner を起動して分析を開始できます。次のオプションを使用して、開始時に検索範囲を限定してください。

使用するオプション	指定する内容
StartScn	SCN 範囲の開始
EndScn	SCN 範囲の終了
StartTime	時間間隔の開始
EndTime	時間間隔の終了
DictFileName	ディクショナリ・ファイル名

LogMiner を起動後は、次のデータ・ディクショナリ・ビューを分析に使用できます。

ビュー	表示される情報
V\$LOGMNR_DICTIONARY	使用中のディクショナリ・ファイル
V\$LOGMNR_PARAMETERS	LogMiner の現行のパラメータ設定
V\$LOGMNR_FILES	分析対象の REDO ログ・ファイル
V\$LOGMNR_CONTENTS	分析対象の REDO ログ・ファイルの内容

LogMiner を使用する手順

1. DBMS_LOGMNR.START_LOGMNR プロシージャを発行して LogMiner ユーティリティを起動します。たとえば、 /oracle/dictionary.ora を使用して LogMiner を起動するには、次のプロシージャを発行します。

```
execute dbms_logmnr.start_logmnr(  
  DictFileName => '/oracle/dictionary.ora');
```

必要であれば、StartTime および EndTime パラメータを設定して、データに時刻のフィルタをかけます。プロシージャは、日付値を必要とするので注意してください。この例のように、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。

```
execute dbms_logmnr.start_logmnr(
  DictFileName => '/oracle/dictionary.ora',
  StartTime => to_date('01-Jan-98 08:30:00', 'DD-MON-YYYY HH:MI:SS')
  EndTime => to_date('01-Jan-1998 08:45:00', 'DD-MON-YYYY HH:MI:SS'));
```

次のように、StartScn および EndScn パラメータを使用して、データに SCN でフィルタをかけます。

```
execute dbms_logmnr.start_logmnr(
  DictFileName => '/oracle/dictionary.ora',
  StartScn => 100,
  EndScn => 150);
```

2. V\$LOGMNR_CONTENTS 表を介して出力を表示します。LogMiner では、すべての行が SCN 順に戻されます。これは、メディア回復に適用されるのと同じ順序です。たとえば、次の問合せでは操作情報のリストが表示されます。

```
SELECT operation, sql_redo FROM v$logmnr_contents;
OPERATION SQL_REDO
-----
INTERNAL
INTERNAL
START      set transaction read write;
UPDATE     update SYS.UNDO$ set NAME = 'RS0', USER# = 1, FILE# = 1, BLOCK# = 2450, SCNBAS =
COMMIT     commit;
START      set transaction read write;
UPDATE     update SYS.UNDO$ set NAME = 'RS0', USER# = 1, FILE# = 1, BLOCK# = 2450, SCNBAS =
COMMIT     commit;
START      set transaction read write;
UPDATE     update SYS.UNDO$ set NAME = 'RS0', USER# = 1, FILE# = 1, BLOCK# = 2450, SCNBAS =
COMMIT     commit;
11 rows selected.
```

関連項目 : DBMS_LOGMNR の詳細は、『Oracle8i パッケージ・プロシージャ リファレンス』を参照してください。

LogMiner のデータ・ディクショナリ・ビューの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

他のデータベースからのアーカイブ済み REDO ログ・ファイルの分析 あるデータベースからの REDO ログ・ファイルの分析中に、他のデータベースのインスタンスに対して LogMiner を実行できます。他のデータベースからのアーカイブ済み REDO ログ・ファイルを分析するための、LogMiner の要件は次のとおりです。

- REDO ログ・ファイルと同じデータベースから同じデータベース・キャラクタ・セットを使用して作成されたディクショナリ・ファイルにアクセスする必要があります。
- ログ・ファイルが生成されたのと同じハードウェア・プラットフォーム上で動作する必要があります。これは、同じシステム上でなくてもかまいません。
- Oracle バージョン 8.0 以上からの回復に適用できる REDO ログ・ファイルを使用する必要があります。

LogMiner の使用 : 使用例

ここでは、LogMiner の次の使用例について説明します。

- [ユーザーの追跡](#)
- [表アクセス統計の計算](#)

ユーザーの追跡

この例では、特定範囲の時間中にユーザーの 1 人である JOEDEVO がデータベースに対して行ったすべての変更を表示しようとしています。この操作は次の手順で実行します。

- [ステップ 1: ディクショナリの作成](#)
- [ステップ 2: ログの追加と検索範囲の限定](#)
- [ステップ 3: LogMiner の起動とデータの分析](#)

ステップ 1: ディクショナリの作成 LogMiner を使用して JOEDEVO のデータを分析するには、LogMiner を起動する前にディクショナリ・ファイルを作成する必要があります。

次の操作を実行します。

- ディクショナリ・ファイル `orcldict.ora` をコールします。
- このディクショナリをディレクトリ `/user/local/dbs` に格納します。
- 初期化パラメータ `UTL_FILE_DIR` を `/user/local/dbs` に設定します。

```
# Set the initialization parameter UTL_FILE_DIR in the init.ora file
UTL_FILE_DIR = /user/local/dbs
```

```
# Start SQL*Plus and then connect to the database
connect system/manager
```

```
# Open the database to create the dictionary file
startup
```

```
# Create the dictionary file
execute dbms_logmnr_d.build(
```

```
dictionary_filename => 'orcldict.ora',
dictionary_location => '/usr/local/dbs');
```

```
# The dictionary has been created and can be used later
shutdown;
```

ステップ 2: ログの追加と検索範囲の限定 これでディクショナリが作成されたので、特定の時刻に発生した変更を表示することにしします。次の操作を実行します。

- 使用するログ・ファイルのリストを作成し、ログ log1orcl.ora を指定します。
- リストにログ log2orcl.ora を追加します。
- LogMiner を起動し、検索範囲を 1998 年 1 月 1 日の午前 8:30 ~ 8:45 に限定します。

```
# Start SQL*Plus, connect as SYSTEM, then start the instance
connect system/manager
startup nomount
```

```
# Supply the list of logfiles to the reader. The Options flag is set to indicate this is a
# new list.
```

```
execute dbms_logmnr.add_logfile(Options => dbms_logmnr.NEW,
LogFileName => 'log1orcl.ora');
```

```
# Add a file to the existing list. The Options flag is clear to indicate that you are
# adding a file to the existing list
```

```
execute dbms_logmnr.add_logfile(Options => dbms_logmnr.ADDFILE,
LogFileName => 'log2orcl.ora');
```

ステップ 3: LogMiner の起動とデータの分析 この時点で、V\$LOGMNR_CONTENTS 表を問合せに使用可能です。ユーザー JOEDEVO が給与表に対して行ったすべての変更を検索することにしします。その結果、JOEDEVO は次の 2 つの操作を要求したことが判明します。つまり、自分の過去の給与を削除し、前より高額な給与を新規に挿入しています。これで、この操作を取り消す（および、JOEDEVO の解雇を正当化する）ために必要なデータが得られました。

```
# Start the LogMiner. Limit the search to the specified time range.
execute dbms_logmnr.start_logmnr(
DictFileName => 'orcldict.ora',
StartTime => to_date('01-Jan-98 08:30:00', 'DD-MON-YYYY HH:MI:SS')
EndTime => to_date('01-Jan-1998 08:45:00', 'DD-MON-YYYY HH:MI:SS'));
```

```
SELECT sql_redo, sql_undo FROM v$logmnr_contents
WHERE username = 'JOEDEVO' AND tablename = 'SALARY';
```

```
# The following data is displayed (properly formatted)
```

SQL_REDO

delete * from SALARY

where EMPNO = 12345

and ROWID = 'AAABOOAABAAEPCABA';

insert into SALARY(NAME, EMPNO, SAL)

values('JOEDEVO',12345,2500)

2 rows selected

SQL_UNDO

insert into SALARY(NAME,EMPNO, SAL)

values ('JOEDEVO', 12345,500)

delete * from SALARY

where EMPNO = 12345

and ROWID = 'AAABOOAABAAEPCABA';

表アクセス統計の計算

Oracle RDBMS によって生成される REDO ログ・ファイルには、データベースに対するすべての変更の履歴が含まれています。したがって、REDO ログを調べれば、データベースの調整に役立つ情報が得られます。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。

最初に、LogMiner を起動して時間の範囲を指定します。

```
execute dbms_logmnr.start_logmnr(
  StartTime => '07-Aug-98',
  EndTime => '15-Aug-98',
  DictFileName => '/usr/local/dict.ora');
```

次に、V\$LOGMNR_CONTENTS を問い合せて、指定した時間範囲内に変更があった表を判別します。

```
select seg_owner, seg_name, count(*) as Hits from
V$LOGMNR_CONTENTS where seg_name not like '%$' group by
seg_owner, seg_name;
```

SEG_OWNER	SEG_NAME	Hits
-----	-----	----
CUST	ACCOUNT	384
SCOTT	EMP	12
SYS	DONOR	12
UNIV	DONOR	234
UNIV	EXECDONOR	325
UNIV	MEGADONOR	32

関連項目：V\$LOGMNR_CONTENTS、または LogMiner のビューや初期化パラメータの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

DBMS_LOGMNR.ADD_LOGFILE や他の PL/SQL パッケージの詳細は、『Oracle8i パッケージ・プロシージャ リファレンス』を参照してください。

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 PL/SQL コードを定期的に実行するためのスケジューリング方法について説明します。
この章のトピックは、次のとおりです。

- SNP バックグラウンド・プロセス
- ジョブ・キューの管理
- ジョブ・キューについての情報の表示

SNP バックグラウンド・プロセス

ここでは、SNP バックグラウンド・プロセスと、ジョブ・キューの管理に果たす役割について説明します。この項の構成は、次のとおりです。

- 複数の SNP プロセス
- SNP プロセスの起動

ジョブ・キューを使用すると、ルーチンが定期的に行われるようにスケジュールできます。このルーチンとは PL/SQL コードのことです。ジョブをスケジュールするには、ジョブをジョブ・キューに入れて、ジョブを実行する頻度を指定します。キューに入れられたジョブは、変更、使用禁止または削除することができます。

パフォーマンスを最大にし、多くのユーザーから使用できるようにするために、マルチプロセス Oracle システムでは「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ます。バックグラウンド・プロセスは、各ユーザー・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ようなファンクションを統合します。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 プロセスを監視し、並列性を高め、パフォーマンスと信頼性を向上させます。

SNP バックグラウンド・プロセスでは、ジョブ・キューを実行します。SNP プロセスでは、キューに入っている実行予定のジョブを定期的起動し、実行します。キューに入っているジョブをバックグラウンドで実行するためには、1 つ以上の SNP プロセスが稼働していなければなりません。

SNP バックグラウンド・プロセスは、Oracle の他のバックグラウンド・プロセスとは異なり、SNP プロセスで障害が発生しても、インスタンスには影響を与えません。SNP プロセスで障害が発生すると、Oracle がこのプロセスを再起動させます。

システムが制限モードで開始された場合、SNP バックグラウンド・プロセスは、ジョブを実行しません。ただし、ALTER SYSTEM コマンドを使用すると、次のように制限モードのオンとオフを切り替えることができます。

```
ALTER SYSTEM ENABLE RESTRICTED SESSION;  
ALTER SYSTEM DISABLE RESTRICTED SESSION;
```

制限セッションを使用可能にすると、SNP バックグラウンド・プロセスはジョブを実行しますが、使用禁止にすると実行します。

関連項目 : SNP バックグラウンド・プロセスの詳細は、『Oracle8i 概要』を参照してください。

複数の SNP プロセス

1 つのインスタンスの SNP プロセスの数は、最大 36 個です (SNP0 ~ SNP9、および SNPA ~ SNPZ の名前)。1 つのインスタンスに複数の SNP プロセスがあると、キューに入っているジョブを実行するタスクをこの複数のプロセス間で共有できるため、パフォーマンスが向上します。ただし、各ジョブはどの時点でも 1 つのプロセスによってのみ実行されることに注意してください。1 つのジョブを複数の SNP プロセスで同時には共有できません。

SNP プロセスの起動

ジョブ・キューの初期化パラメータを使用すると、SNP バックグラウンド・プロセスの動作を制御できます。インスタンスの初期化パラメータ・ファイルで次のパラメータを設定すると、設定したパラメータは、次にインスタンスを起動したときに有効になります。

表 8-1 に、ジョブ・キューの初期化パラメータを示します。

表 8-1 ジョブ・キュー初期化パラメータ

パラメータ名	説明
JOB_QUEUE_PROCESSES	デフォルト: 0 値の範囲: 0 ~ 36 複数インスタンス: 異なる値を設定可能 インスタンスごとの SNP バックグラウンド・プロセスの数を設定します。
JOB_QUEUE_INTERVAL	デフォルト: 60 (秒) 値の範囲: 1 ~ 3600 (秒) 複数インスタンス: 異なる値を設定可能 インスタンスの SNP バックグラウンド・プロセスの起動間隔を設定します。

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。この項のトピックは、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](#)
- [WHAT の構文](#)

- 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・キュー内のジョブのスケジューリングおよび管理は、DBMS_JOB パッケージのプロシージャを使用して実行します。ジョブ・キューの使用に関係付けられた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、だれでもジョブ・キューを使用できます。表 8-2 に、DBMS_JOB パッケージ内のジョブ・キュー・プロシージャを示します。

表 8-2 DBMS_JOB パッケージのプロシージャ

プロシージャ	説明	参照先
SUBMIT	ジョブをジョブ・キューに送ります。	8-4 ページ
REMOVE	ジョブ・キューから指定したジョブを削除します。	8-11 ページ
CHANGE	指定した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。	8-12 ページ
WHAT	指定したジョブのジョブの定義を変更します。	8-12 ページ
NEXT_DATE	指定したジョブの次の実行時刻を変更します。	8-12 ページ
INTERVAL	指定したジョブの実行間隔を変更します。	8-12 ページ
BROKEN	ジョブの実行を禁止にします。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ません。	8-12 ページ
RUN	指定したジョブを強制的に実行させます。	8-14 ページ

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。

```
DBMS_JOB.SUBMIT(  job          OUT  BINARY_INTEGER,
                  what          IN   VARCHAR2,
```

```
next_date      IN      DATE DEFAULT SYSDATE,
interval       IN      VARCHAR2 DEFAULT 'null',
no_parse       IN      BOOLEAN DEFAULT FALSE)
```

SUBMIT プロシージャは、送られたジョブの番号を戻します。表 8-3 に、このプロシージャのパラメータを示します。

表 8-3 DBMS_JOB.SUBMIT のパラメータ

パラメータ	説明
job	作成した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、必ずジョブ番号を使用する必要があります。 ジョブ番号の詳細は、8-7 ページの「ジョブ番号」を参照。
what	実行する PL/SQL コード。 ジョブの定義方法の詳細は、8-7 ページの「ジョブ定義」を参照。
next_date	ジョブを次に実行する日付。デフォルト値は SYSDATE。
interval	次にジョブを実行する時刻を計算する日付ファンクション。デフォルト値は NULL。INTERVAL は将来のある時点または NULL に評価される必要があります。 実行間隔の指定方法の詳細は、8-8 ページの「ジョブの実行間隔」を参照。
no_parse	フラグ。デフォルト値は FALSE。 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、ジョブに対応付けられている表を作る前に、そのジョブを送る場合は、NO_PARSE を TRUE に設定します。

例として、新しいジョブをジョブ・キューに送ります。ジョブはプロシージャ DBMS_DDL.ANALYZE_OBJECT をコールして、DQUON.ACCOUNTS 表のオブティマイザ統計を生成します。この統計は、ACCOUNTS 表の行数の半分のサンプルに基づいています。このジョブは、24 時間ごとに実行されます。

```
VARIABLE jobno number;
begin
    2>          DBMS_JOB.SUBMIT(:jobno,
    3>          'dbms_ddl.analyze_object(''TABLE'',
    4>          ''DQUON'', ''ACCOUNTS'',
    5>          ''ESTIMATE'', NULL, 50);'
    6>          SYSDATE, 'SYSDATE + 1');
    7>          commit;
    8> end;
    9> /
Statement processed.
```

```
print jobno  
JOBNO  
-----  
14144
```

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
- 現行のスキーマ
- MAC 権限（該当する場合）

次の NLS パラメータも記録します。

- 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再格納されます。NLS_LANGUAGE パラメータと NLS_TERRITORY パラメータは、NLS パラメータが指定されなかった場合のデフォルトです。

DBMS_SQL パッケージと ALTER SESSION コマンドを使用して、ジョブの環境を変更できます。

ジョブとインポート/エクスポート

ジョブは、インポートしたりエクスポートしたりできます。したがって、1つの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。ジョブのインポートでもエクスポートでも、ジョブ番号、環境および定義は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の既存の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、ジョブの所有者だけです。

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、SYS.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したりエクスポートしたりしても、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ってください。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。

Oracle がジョブ定義の中で認識する特殊なパラメータ値がいくつかあります。表 8-4 にこれらのパラメータを示します。

表 8-4 ジョブ定義の特殊なパラメータの値

パラメータ	モード	説明
job	IN	現行ジョブの番号
next_date	IN/OUT	ジョブが次に実行される日付。デフォルト値は SYSDATE。
broken	IN/OUT	ジョブが中断されているか中断されていないかの状態。IN の値は FALSE。

次に、有効なジョブ定義の例を示します。

```
'myproc(''10-JAN-82'', next_date, broken);'  
'scott.emppackage.give_raise(''JFEE'', 3000.00);'  
'dms_job.remove(job);'
```

ジョブの実行間隔

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 から計算された日付が新しい NEXT_DATE になります。INTERVAL 日付関数が NULL に評価され、ジョブが正常に実行されると、そのジョブはキューから削除されます。

設定した間隔でジョブを定期的に行うには、INTERVAL パラメータで 'SYSDATE + 7' のような日付式を使用してください。たとえば、月曜日に実行間隔を 'SYSDATE + 7' と設定したときに、なんらかの理由で（たとえば、ネットワーク障害など）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。

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。

表 8-5 に、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ます。

表 8-5 一般的なジョブの実行間隔

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。

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が入っていなければなりません。名前のないデータベース・リンクは正常に機能しません。

関連項目： DBMS_SQL パッケージの詳細は、『Oracle8i パッケージ・プロシージャ リファレンス』を参照してください。

ジョブの実行方法

各ジョブは、SNP バックグラウンド・プロセスで実行されます。ジョブを実行するため、SNP バックグラウンド・プロセスによりジョブを実行するセッションが作成されます。

SNP 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。

DBMS_JOB.RUN プロシージャを使用してジョブを強制的に実行させる場合、ジョブはユーザー・プロセスで実行されます。ユーザー・プロセスでジョブが実行される場合、ジョブは直接付与されている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。

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 1 度に 1 つのセッションで実行されるようにします。ジョブの実行中は、そのセッションはジョブに対するジョブ・キュー (JQ) をロックします。

JQ ロックについての情報の解釈 Enterprise Manager の「Lock Monitor」またはデータ・ディクショナリのロッキング・ビューによって、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.

このキューで、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 ロック識別子は常に 0 です。ID2 ロック識別子は、セッションで実行中のジョブのジョブ番号です。

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として、次のものがあります。

- ジョブを実行するための SNP バックグラウンド・プロセスがない場合
- 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

ジョブの失敗と実行時刻 Oracle がジョブの実行中にそのジョブがエラーを戻した場合、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。再試行の間隔が実行間隔を超えると、Oracle は標準の実行間隔でジョブの再試行を続けます。ただし、ジョブの失敗の回数が 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。

したがって、ジョブの実行失敗の回数が 16 回になる前に、そのジョブの実行を妨げる問題を訂正できれば、最終的に Oracle はそのジョブをもう 1 度実行します。

関連項目 : ロック・ビューの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

ロックの詳細は、『Oracle8i 概要』を参照してください。

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。

```
DBMS_JOB.REMOVE(job IN BINARY_INTEGER)
```

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。

```
DBMS_JOB.REMOVE(14144);
```

制限

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に、現行の実行は完了します。

削除できるのは、自分が所有しているジョブだけです。自分が所有していないジョブを削除しようすると、そのジョブはジョブ・キューに存在しないジョブである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、INTERVAL のいずれかのプロシージャを使用します。

次の例では、14144 で識別されるジョブを 3 日ごとに実行するように変更します。

```
DBMS_JOB.CHANGE(14144, null, null, 'SYSDATE + 3');
```

制限

変更できるのは、自分の所有しているジョブだけです。自分の所有していないジョブを変更しようすると、そのジョブはジョブ・キューにないことを示すメッセージが表示されます。

CHANGE の構文

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。このプロシージャのパラメータについては、[表 8-3](#) を参照してください。

DBMS_JOB.CHANGE(job	IN BINARY_INTEGER,
what	IN VARCHAR2,
next_date	IN DATE,
interval	IN VARCHAR2)

- 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

ジョブに中断または非中断のマークを付けるには、DBMS_JOB パッケージの BROKEN プロシージャを使用します。このプロシージャのパラメータについては、[表 8-4](#) を参照してください。

```
DBMS_JOB.BROKEN( job           IN BINARY_INTEGER,  
                 broken        IN BOOLEAN,  
                 next_date     IN DATE DEFAULT SYSDATE)
```

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ます。

```
DBMS_JOB.BROKEN(14144, FALSE, NEXT_DAY(SYSDATE, 'MONDAY'));
```

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させない限り、Oracle により実行されません。

制限

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だけです。自分が所有していないジョブにマークを付けようとすると、そのジョブはジョブ・キューに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は、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方法。
- 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つ方法。

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リセットします。

ジョブの中断フラグを（RUN または BROKEN をコールして）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したい場合があります。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することによってただちにテストしたい場合です。

ジョブをただちに強制実行するには、DBMS_JOB パッケージの RUN プロシージャを使用します。このプロシージャを使用すると、Oracle はジョブに中断のマークが付いていても、そのジョブを実行しようとします。

```
DBMS_JOB.RUN( job IN BINARY_INTEGER)
```

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もう一度計算します。たとえば、'SYSDATE' の NEXT_DATE 値と 'SYSDATE + 7' の INTERVAL 値を使用してジョブを月曜日に作成すると、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。

注意： ジョブを強制的に実行すると、そのジョブは現行の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

実行できるのは、自分が所有しているジョブだけです。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・キューにないことを示すメッセージが表示されます。

次の文で、ジョブ 14144 は現行のセッションで実行し、次の実行の日付をもう一度計算します。

```
DBMS_JOB.RUN(14144);
```

RUN プロシージャには、暗黙の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、ロールバックはできません。

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 がその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。

ジョブを実行中のセッションを（V\$SESSION を介して）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。

関連項目： ジョブとセッションについての情報の表示例は、「[ジョブ・キューについての情報の表示](#)」を参照してください。

V\$SESSION の詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

ジョブ・キューについての情報の表示

表 8-6 のデータ・ディクショナリ・ビューによって、ジョブ・キュー内のジョブについての情報を見ることができます。

表 8-6 ジョブ・キュー内の情報を示すビュー

ビュー	説明
DBA_JOBS	データベース内のすべてのジョブをリストします。
USER_JOBS	ユーザーが所有するすべてのジョブをリストします。
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。このビューは V\$LOCK と JOBS を結合します。

たとえば、ジョブの状態および失敗した実行についての情報を表示できます。次の問合せの例では、キューに送られている各ジョブのジョブ番号および次の実行時刻、失敗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 user_jobs;
```

```
JOB          NEXT_DATE NEXT_SEC FAILURES  B
-----
9125         01-NOV-94 00:00:00      4      N
14144        24-OCT-94 16:35:35      0      N
41762        01-JAN-00 00:00:00     16      Y
3 rows selected.
```

また、現在実行中のジョブについての情報も表示できます。次の問合せの例では、現在実行中のすべてのジョブについて、セッション識別子および、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         JFEE              24-OCT-94  17:21:24
25           8536         SCOTT             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、『Oracle8i リファレンス・マニュアル』を参照してください。

第 III 部

データベース記憶域

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。この章のトピックは、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](#)
- [表領域割当ての管理](#)
- [表領域の可用性の変更](#)
- [表領域を読取り専用にする方法](#)
- [表領域の削除](#)
- [DBMS_SPACE_ADMIN パッケージの使用](#)
- [データベース間での表領域のトランスポート](#)
- [表領域についての情報の表示](#)

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、次に説明するガイドラインを理解してください。

- [複数の表領域の使用](#)
- [表領域の記憶領域パラメータの指定](#)
- [ユーザーに表領域割当て制限を割り当てる方法](#)

複数の表領域の使用

データベース操作を実行しているとき、複数の表領域を使用するとシステムはより柔軟性に富んだものとなります。たとえば、データベースに複数の表領域があるときには、次のことができます。

- データ・ディクショナリのデータからユーザー・データを分離すること。
- 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すること。
- 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こと。
- 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こと。
- 別の表領域をオンライン状態に保持しながら、個々の表領域をオフラインにすること。
- 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と。
- 個々の表領域をバックアップすること。

オペレーティング・システムによっては、同時にオープンできるファイル数に制限が設定されていることがあります。つまり、これらの制限は、同時にオンラインにできる表領域の数に間接的な影響を与えます。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してください。それとともに、できる限りファイル数が少なくなるように表領域を作成してください。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。

これらの利点を考慮に入れてデータを見直し、データベース設計に必要な表領域の数を決定してください。

表領域の記憶領域パラメータの指定

新しい表領域を作成するとき、次のルールに従って、その表領域に作成されるオブジェクトに対して、デフォルトの記憶領域パラメータを指定できます。オブジェクトを作成するときに指定された記憶領域パラメータによって、そのオブジェクトが含まれている表領域のデフォルトの記憶領域パラメータが上書きされます。ただし、オブジェクトを作成するときに記憶領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表領域に対して自動的にデフォルトの記憶領域パラメータを使用します。

表領域のデフォルトの記憶領域パラメータは、その表領域が収容する代表的なオブジェクトのサイズを計算して設定してください（サイズを見積ってください）。例外的なオブジェクトに対する記憶領域パラメータは、そのオブジェクトを作成するときに指定できます。

注意： 新しい表領域にデフォルトの記憶領域パラメータを指定しないと、Oracle のデフォルトの記憶領域パラメータが、その表領域のデフォルトの記憶領域パラメータとなります。

関連項目： オブジェクトのサイズの見積りについては、第 11 ~ 17 章を参照してください。

ユーザーに表領域割当て制限を割り当てる方法

表、クラスタ、スナップショット、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保有する予定の表領域内の割当て制限（領域の許容または制限）を与えます。セキュリティ管理者には、オブジェクトを作成するの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責任があります。

関連項目： 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方法については、23-12 ページの「[表領域の割当て制限の割当て](#)」を参照してください。

表領域の作成

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。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は、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。それぞれの場合に、Oracle は、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。しかし、オペレーティング・システムによっては、インストールの前にデータ・ファイル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、データベースを作成するときには、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。

次のような場合には、新しい表領域を作成できます。

- 対応するデータベースに、より多くのディスク記憶領域を割り当て、データベースを大きくする場合。
- 特定のタイプのデータを、他のデータベース・データから分離して格納するために、論理記憶構造を作成する必要がある場合。

データベースの合計サイズを大きくするために、新しい表領域を追加するのではなく、既存の表領域にデータ・ファイルを追加できます。

注意： 現在のインスタンスに対して少なくとも 2 つのロールバック・セグメントが (SYSTEM ロールバック・セグメントを含めて) 取得されるまで、どのようなデータも新しい表領域に挿入できません。

新しい表領域を作成するには、SQL 文 CREATE 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。

次の例では、(データベースのロールバック・セグメントを保持するために) 次のような特性を持つ表領域 RB_SEGS 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。
- この表領域に作成されるすべてのセグメントに対して、デフォルト記憶領域パラメータが明示的に設定されます。
- 表領域は、作成後にオフラインになります。

次の文では、表領域 RB_SEGS が作成されます。

```
CREATE TABLESPACE rb_segs
  DATAFILE 'datafilers_1' SIZE 50M
  DEFAULT STORAGE (
    INITIAL 50K
    NEXT 50K
    MINEXTENTS 2
    MAXEXTENTS 50
    PCTINCREASE 0)
  OFFLINE;
```

表領域の作成時にファイル名を完全に指定しなければ、それに対応するデータ・ファイルは ORACLE_HOME/dbfs ディレクトリに作成されます。

関連項目： 表領域の初期作成の詳細は、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。

データ・ファイルの追加方法の詳細は、10-5 ページの「[データ・ファイルの作成および表領域への追加](#)」を参照してください。

CREATE TABLESPACE 文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

ローカルに管理される表領域の作成

通常、表領域は「ディクショナリにマップ」されています。これは、この種の表領域では、SQL ディクショナリ表を使用して領域の使用率が追跡されることを意味します。これに対して、「ローカルに管理される表領域」では、使用済み領域と空き領域の追跡に（SQL ディクショナリ表のかわりに）ビットマップが使用されます。

ローカルに管理される表領域を作成して使用すると、次のような利点があります。

- 領域操作の同時発生が改善されます。
ビット値を（割り当てる場合は 0 を 1 に、割当て解除の場合は 1 を 0 に）変更すると、領域を割り当てたり、割当てを解除できます。
- 領域管理操作中の再帰が排除されます。
- スタンバイ・データベースで一時表領域の管理がサポートされます。
- データ・ディクショナリに対するユーザーの依存度が低減します。

必要な情報は、セグメント・ヘッダーとビットマップ・ブロックに格納されます。

次の文では、ローカルに管理される表領域 TBS_1 が作成され、各エクステンツは 128KB で、ビットマップ内の各ビットでは 64 個のブロックが記述されます。

```
CREATE TABLESPACE tbs_1 DATAFILE 'file_1.f'  
    BITMAP ALLOCATION UNIFORM SIZE 128K;
```

関連項目：ローカルに管理される表領域の構文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

ローカルに管理される SYSTEM 表領域を持つデータベースの作成

ローカルに管理される SYSTEM 表領域を持ったデータベースを作成できます。ただし、このデータベースのロールバック・セグメントも、均一かつローカルに管理される表領域内で作成する必要があります。ローカルに管理される SYSTEM 表領域は、常にシステムによって管理されます。また、後でバージョン 8.1 以前の Oracle の状態に戻すことはできません。

関連項目：ローカルに管理される SYSTEM 表領域を持つデータベースを作成する方法の詳細は、『Oracle8i SQL リファレンス』を参照してください。

一時表領域の作成

複数のソート操作の並行性の改善、オーバーヘッドの減少または Oracle の領域管理操作の完全な回避を実現する場合には、一時表領域を作成できます。

一時表領域では、あるインスタンスと表領域のソート操作は、すべて 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。一時表領域には永続オブジェクトを格納できません。

VSSORT_SEGMENT 表を介して、一時表領域のソート・セグメント内の領域の割当てと、割当て解除を表示できます。

表領域を作成するときにその表領域を一時表領域として指定するには、次の文を発行します。

```
CREATE TABLESPACE tablespace TEMPORARY;
```

既存の表領域で表領域を一時表領域として指定するには、次の文を発行します。

```
ALTER TABLESPACE tablespace TEMPORARY;
```

注意： 一時表領域をオフラインにできます。一時表領域をオンラインに戻しても、一時的状態には影響しません。

関連項目： CREATE TABLESPACE 文および ALTER TABLESPACE の詳細は、『Oracle8i SQL リファレンス』を参照してください。

VSSORT_SEGMENT の詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

Oracle の領域管理の詳細は、『Oracle8i 概要』を参照してください。

一時データ・ファイル

一時データ・ファイルは、恒久データ・ファイルと違って、DBA_DATA_FILES ビューには表示されません。かわりに、DBA_TEMP_FILES ビューに表示されます。このビューは DBA_DATA_FILES ビューに似ていますが、一時データ・ファイルに関する情報が含まれています。SQL では、一時表領域に属しているファイルも、DATAFILES ではなく TEMPFILES として識別されます。

関連項目： 一時データ・ファイルと DBA_TEMP_FILES の詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

ローカルに管理される一時表領域の作成

データベース内でセッションの継続時間中にスキーマ・オブジェクトを入れることができる領域を割り当てる場合は、ローカルに管理される一時表領域を作成できます。

ローカルに管理される一時表領域を作成するには、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。デフォルトのデータベース・ブロック・サイズは 2KB で、マップ内の各ビットは 1 エクステントを表すので、各ビットで 8,000 個のブロックがマップされます。

```
CREATE TEMPORARY TABLESPACE tbs_1 TEMPFILE 'file_1.f'  
    BITMAP ALLOCATION UNIFORM SIZE 16M;
```

関連項目：ローカルに管理される一時表領域を持つデータベースを作成する方法の詳細は、『Oracle8i SQL リファレンス』を参照してください。

ローカルに管理される一時表領域の変更

ローカルに管理される一時表領域を変更したり、データ・ファイル（つまり一時ファイル）を追加できます。

次の文では、ローカルに管理される一時表領域にファイルが追加されます。

```
ALTER TABLESPACE tbs_1  
    ADD TEMPFILE 'file_1.f';
```

次の文では、一時ファイルがオフラインにされてからオンラインに戻されます。

```
ALTER DATABASE TEMPFILE 'temp_file_1.f' OFFLINE;  
ALTER DATABASE TEMPFILE 'temp_file_1.f' ONLINE;
```

次の文では、一時ファイル TEMP_FILE_1.F が 12KB にサイズ変更されます。

```
ALTER DATABASE TEMPFILE 'temp_file_1.f' RESIZE 12K;
```

次の文では、一時ファイルが削除されます。

```
ALTER DATABASE TEMPFILE 'temp_file_1.f' DROP;
```

関連項目：ローカルに管理される一時表領域を変更するための文の詳細と制限は、『Oracle8i SQL リファレンス』を参照してください。

表領域割当ての管理

ここでは、表領域の割当ての管理について説明します。この項のトピックは、次のとおりです。

- [表領域に対する記憶領域設定の変更](#)
- [空き領域を合わせる方法](#)

表領域に対する記憶領域設定の変更

表領域のデフォルトの記憶領域パラメータを変更して、将来その表領域に作成されるオブジェクトのデフォルトの指定を変更できます。後から表領域に作成されるオブジェクトのデフォルトの記憶領域パラメータを変更するには、SQL 文 ALTER TABLESPACE を使用します。また、表領域のデフォルト記憶領域パラメータの変更には、ALTER TABLESPACE システム権限が必要です。

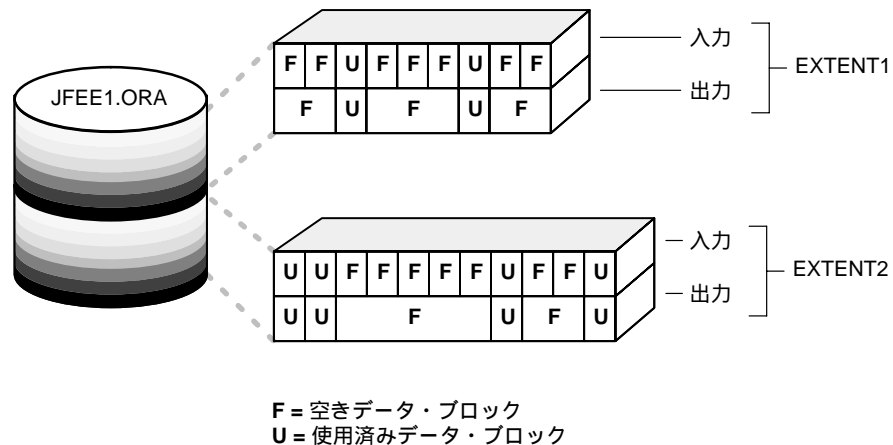
```
ALTER TABLESPACE users
  DEFAULT STORAGE (
    INITIAL 50K
    NEXT 50K
    MINEXTENTS 2
    MAXEXTENTS 20
    PCTINCREASE 50);
```

表領域のデフォルト記憶領域パラメータの新しい値は、その表領域内のセグメントに対して割り当てられる将来のエクステントにだけ影響します。

空き領域を合わせる方法

表領域セグメントの領域は、特定の数の連続するデータ・ブロックから構成されるエクステントを使用して管理されます。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われます。したがって、大きな使用可能エクステントを分けたり、連続する小さな使用可能エクステントを合わせて1つの大きな使用可能エクステントにすることができます（[図 9-1](#) を参照）。ただし、空き領域の割当ておよび割当て解除を連続して実行すると、表領域が分けられて、大きなエクステントの割当てが困難になります。デフォルトでは、SMON（システム監視）プロセスが表領域の使用可能エクステントをバックグラウンドで合わせていきます。必要に応じて、SMON が使用可能エクステントを合わせないようにすることもできます。

図 9-1 空き領域を合わせる



領域の断片化が激しい（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）場合には、1つの領域トランザクションで空き領域を合わせることができます。8回合わせるたびに領域トランザクションはコミットし、別のトランザクションが領域の割当てや割当て解除を実行できるようになります。表領域を合わせるには、ALTER TABLESPACE 権限が必要です。次のコマンドを使用すると、表領域ごとに、表領域内の使用可能なすべての空き領域のエクステントを合わせて、連続する大きなエクステントにすることができます。

```
ALTER TABLESPACE tablespace COALESCE;
```

また、このコマンドを使用して、SMON およびエクステント割当てを合わせる機能を補うことによって、激しく断片化した表領域での領域割当てのパフォーマンスを向上させることもできます。このコマンドを発行しても、その表領域にアクセスしている他のユーザーのパフォーマンスには影響ありません。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。

表領域についての情報の表示

表領域について、合わせることのできるエクステントについての統計を表示するには、DBA_FREE_SPACE_COALESCED ビューを表示させます。特定の表領域で領域を合わせる必要があるかどうかを判断する場合に、このビューを問い合わせることができます。

関連項目：DBA_FREE_SPACE_COALESCED の内容の詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

表領域の可用性の変更

オフライン状態の表領域をオンラインにして、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ができます。また、データベースをオープンしたままオンライン状態の表領域をオフラインにして、データベースのその部分だけを一時的に一般用途では使用できないようにし、残りの部分をオープンのまま使用可能にしておくこともできます。この項のトピックは、次のとおりです。

- [表領域をオンラインにする方法](#)
- [表領域をオフラインにする方法](#)

表領域をオンラインにする方法

Oracle データベースがオープンされていれば、いつでもデータベース内の任意の表領域をオンラインにすることができます。ただし、データ・ディクショナリは常に Oracle で使用できる状態である必要がありますので、SYSTEM 表領域は常にオンラインにしてください。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。

データベースがオープンされているときにオフライン表領域をオンラインにするには、SQL 文 ALTER TABLESPACE を使用します。表領域をオンラインにするには、MANAGE TABLESPACE システム権限が必要です。

注意： オンラインにしようとする表領域が、「明らかに」(ALTER TABLESPACE OFFLINE 文の NORMAL オプションを使用して) オフラインになっていない場合、最初にメディア回復をしない限りオンラインにできません。メディア回復をしないと、エラーが戻されて表領域はオフラインのままになります。

次の文は、USERS 表領域をオンラインにします。

```
ALTER TABLESPACE users ONLINE;
```

表領域をオフラインにする方法

表領域は、次のような理由の場合はオフラインにすることができます。

- データベースの一部だけ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。
- オフラインの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はバックアップ可能）。
- アプリケーションの更新時またはメンテナンス時には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。

データベースがオープンされているときにオンライン表領域をオフラインにするには、SQL コマンド ALTER TABLESPACE を使用します。表領域をオフラインにするには、MANAGE TABLESPACE システム権限が必要です。

表領域をオフラインにするときには、次の優先順位のどれかを指定できます。

通常のオフライン	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にすることができます。書き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にすることができません。通常のオフラインの優先順位では、Oracle は表領域のすべてのデータ・ファイルのチェックポイントを取って、データ・ファイルをオフラインにします。
一時オフライン	<p>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にすることができます。一時オフラインの優先順位では、Oracle はまだオフラインになっていないデータ・ファイルのチェックポイントを取って、これらのファイルをオフラインにします。</p> <p>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にする場合は、表領域をオンラインに戻す前にメディア回復をする必要はありません。ただし、表領域の 1 つまたは複数のファイルが書き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にする場合は、表領域をオンラインに戻す前に回復をする必要があります。</p>
即時オフライン	表領域を即時にオフラインにする場合は、Oracle がデータ・ファイルのチェックポイントを取る必要はありません。即時オフラインの優先順位では、表領域をオンラインに戻す前に、表領域のメディア回復が必要となります。データベースを NOARCHIVELOG モードで運用している場合、表領域を即時にオフラインにはできません。

警告： 表領域をオフラインにしなければならない場合、できれば通常のオプション（デフォルト）を使用します。これにより、表領域をオフラインにしてからオンラインに戻すまでに、REDO ログ順序が（不完全メディア回復後に ALTER DATABASE OPEN RESETLOGS 文を使用して）リセットされたとしても、表領域が通常どおりにオフラインであれば、表領域の回復を必要とすることなく、オンラインにできることが保証されます。

通常どおりにオフラインにできない場合に限り、表領域を一時的にオフラインにします。この場合、エラーが原因でオフラインにされたファイルだけを、表領域をオンラインにする前に回復する必要があります。通常オプションと一時オプションの両方でオフラインにできなかった後に限り、即時オプションで表領域をオフラインにしてください。

次の例は、USERS 表領域を通常の方法でオフラインにします。

```
ALTER TABLESPACE users OFFLINE NORMAL;
```

関連項目：オンラインの表領域をオフラインにする前に、その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てください。詳細は、21-12 ページの「[ロールバック・セグメントをオフラインにする方法](#)」を参照してください。

表領域を読取り専用にする方法

ここでは、表領域を読取り専用にする方法について説明します。この項のトピックは、次のとおりです。

- [前提条件](#)
- [読取り専用表領域を書込み可能にする方法](#)
- [WORM デバイスの読取り専用表領域の作成](#)

表領域を読取り専用にすると、それ以降は表領域のデータ・ファイルに対して書込み操作ができなくなります。表領域を読取り専用にしてから、その表領域をバックアップしてください。

SQL 文 ALTER TABLESPACE を使用して、表領域を読取り専用に変更します。表領域を読取り専用にするには、ALTER TABLESPACE システム権限が必要です。次の文は、FLIGHTS 表領域を読取り専用にします。

```
ALTER TABLESPACE flights READ ONLY
```

表領域を読取り専用にした後は、そのファイルを読取り専用メディアにコピーできます。次に、SQL 文 ALTER DATABASE RENAME を使用して制御ファイル内のデータ・ファイルを改名し、新しい位置を指し示すようにしてください。

読取り専用の表領域はオンラインでもオフラインでもありません。ONLINE または OFFLINE オプションを指定して ALTER TABLESPACE 文を発行しても、表領域の読取り専用の状態は変更されません。このコマンドを実行すると、表領域内のすべてのデータ・ファイルがオンラインまたはオフラインになります。

ALTER TABLESPACE...READ ONLY 文は、アクティブなトランザクションが完了するまで待ってから読取り専用操作を実行します。したがって、トランザクションが完了するまで待たなくても、表領域を読取り専用にすることができます。

前提条件

表領域を読み取り専用にするには、あらかじめ次の条件を満たしてください。これらの制限条件を満たすには、この機能を制限モードで実行して、RESTRICTED SESSION システム権限を持つユーザーだけがログインできるようにするのが最も簡単な方法です。

- 表領域がオンラインになっていること。
表領域に適用する必要がある取消し情報がないことを確認する必要があります。
- 表領域にはアクティブなロールバック・セグメントが入っていないこと。
したがって、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、SYSTEM 表領域は読み取り専用にはできません。さらに、読み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、表領域を読み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ことをお勧めします。
- オンライン・バックアップは、終了時に表領域内のすべてのデータ・ファイルのヘッダー・ファイルを更新するため、現時点で表領域がオンライン・バックアップに含まれていてはなりません。
- 初期化パラメータ COMPATIBLE は 7.1.0 以上に設定する必要があります。

読み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には、表領域を読み取り専用にする直前に、表領域の表のすべてのブロックにアクセスする問合せを発行しておくといよいでしょう。SELECT COUNT (*) などの単一の問合せを各表に対して実行しておく、それ以降、表領域のデータ・ブロックに最も効果的にアクセスできるようになります。つまり、これにより、最後にブロックを修正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。

警告： 読み取り専用表領域のデータ・ファイルの改名とサイズ変更はできません。

関連項目： 読み取り専用表領域の詳細は、『Oracle8i 概要』を参照してください。

読み取り専用表領域を書込み可能にする方法

表領域は、作成時には常に読み書き可能になっています。読み取り専用の表領域を読み書き可能な表領域に戻すには、SQL コマンド ALTER TABLESPACE を使用します。読み取り専用の表領域を読み書き可能に変更するには、ALTER TABLESPACE システム権限が必要です。次のコマンドは、FLIGHTS 表領域を書込み可能にします。

```
ALTER TABLESPACE flights READ WRITE;
```

読み取り専用の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が更新されるので、読み取り専用バージョンのデータ・ファイルを回復の開始点として使用できます。

前提条件

このコマンドを発行するには、表領域内のすべてのデータ・ファイルをオンラインにしてください。データ・ファイルをオンラインにするには、ALTER DATABASE コマンドの DATAFILE ONLINE オプションを使用します。V\$DATAFILE ビューはデータ・ファイルの現行の状態をリストします。

WORM デバイスの読取り専用表領域の作成

更新する必要のない読取り専用ファイルがあるときは、WORM（追記型）デバイスに読取り専用表領域を作成するとよいでしょう。

1. 別のデバイスに書き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. READ ONLY オプションを指定した ALTER TABLESPACE コマンドを発行して、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にします。
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改名します。データ・ファイルを改名すると、制御ファイルにあ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にします。

表領域の削除

表領域とその内容が必要なくなつたと判断される場合には、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 のデータベースの表領域はすべて削除できます（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。

警告： 表領域が削除されると、表領域のデータを回復できません。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、将来的に必要な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前後では、ただちにデータベースを完全にバックアップする必要があります。表領域を間違つて削除した場合、または表領域が削除された後にデータベースが将来問題を起こした場合、データベースを回復できるように、バックアップを作成することを強くお勧めします。

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・ポインタだけが削除されます。削除された表領域を構成していたデータ・ファイルは削除されません。使われていたディスク領域を解放するには、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、削除した表領域のデータ・ファイルを物理的に削除します。

削除しようとする表領域に、アクティブ・セグメントは存在できません。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・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。簡単にするために、削除する前に表領域をオフラインにしてください。

表領域が削除された後、表領域のエントリは、データ・ディクショナリ（たとえば、DBA_TABLESPACES ビュー）に残っていますが、表領域の状態は INVALID（無効）に変更されます。

表領域を削除するには、SQL コマンド DROP TABLESPACE 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（つまり、表、ビューなど何も含まない）の場合、INCLUDING CONTENTS オプションを指定する必要はありません。表領域に、他の表領域内の表の外部キーによって参照される主キーまたは一意キーを持つ表が含まれている場合、子表の FOREIGN KEY 制約の削除をカスケードする場合には、DROP TABLESPACE 文の CASCADE CONSTRAINTS オプションを使用してください。

関連項目：表領域をオフラインにする方法の詳細は、9-10 ページの「[表領域をオフラインにする方法](#)」を参照してください。

DROP TABLESPACE 文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

DBMS_SPACE_ADMIN パッケージの使用

DBMS_SPACE_ADMIN パッケージは、管理者に欠陥の診断および修復機能を提供します。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。

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。

- SEGMENT_VERIFY
- SEGMENT_CORRUPT
- SEGMENT_DROP_CORRUPT
- SEGMENT_DUMP
- TABLESPACE_VERIFY
- TABLESPACE_REBUILD_BITMAPS
- TABLESPACE_FIX_BITMAPS

- TABLESPACE_MIGRATE_TO_BITMAP
- TABLESPACE_MIGRATE_FROM_BITMAP

関連項目：これらのプロシージャの詳細は、『Oracle8i パッケージ・プロシージャ リファレンス』を参照してください。

使用例 1

TABLESPACE_VERIFY プロシージャで、セグメントにビットマップ内で「空き」マークが付いているブロックを割り当てたが、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る場合。

この使用例では、次のタスクを実行してください。

- SEGMENT_EXTENT_MAP_DUMP プロシージャをコールし、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。
- 範囲ごとに、TABLESPACE_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。

使用例 2

ビットマップには「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。

- SEGMENT_CHECK_ALL 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、次のタスクを実行します。
 - SEGMENT_EXTENT_MAP_DUMP プロシージャをコールし、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。
 - 範囲ごとに、TABLESPACE_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。
 - SEGMENT_DROP_CORRUPT プロシージャをコールして SEG\$ エントリを削除します。

使用例 3

TABSPACE_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。一部の実データを、前の内部エラーに基づいて犠牲にする必要があります。

表 T1 など、犠牲にするオブジェクトを選択し、次のタスクを実行します。

- 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。
- 表 T1 を削除します。必要であれば、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。
- 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。必要であれば、TABSPACE_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。
- TABSPACE_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・ブロックの集合にメディア障害がある場合。

この使用例では、次のタスクを実行してください。

- 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、TABSPACE_REBUILD_MAPS プロシージャをコールします。
- TABSPACE_VERIFY プロシージャをコールして、ビットマップが一貫しているかどうかを検証します。

関連項目 : DBMS_SPACE_ADMIN パッケージの詳細は、『Oracle8i パッケージ・プロシージャ リファレンス』を参照してください。

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。この項のトピックは、次のとおりです。

- [トランスポータブル・テーブルスペースの概要](#)
- [現行の制限事項](#)
- [ステップ 1: 自己完結型の表領域セットの選択](#)
- [ステップ 2: トランスポータブル・テーブルスペースセットの生成](#)
- [ステップ 3: 表領域セットのトランスポート](#)
- [ステップ 4: 表領域セットのプラグイン](#)
- [オブジェクトの動作](#)

- [データ・ウェアハウジングのためのパーティションのトランスポートと 連結例](#)
- [構造化データの CD への発行](#)
- [複数データベースの同じ表領域の読取り専用でのマウント](#)
- [トランスポートابل・テーブルスペースを介した履歴データのアーカイブ](#)
- [トランスポートابل・テーブルスペースを使用した TSPITR の実行](#)

トランスポートابل・テーブルスペースの概要

注意： トランスポートابل・テーブルスペースセットを生成するには、Oracle8i Enterprise Edition が必要です。ただし、トランスポートابل・テーブルスペースセットを Oracle データベースにプラグインするには、Oracle の任意のエディション（Enterprise、Work group または Personal Oracle8i）を使用できます。

「トランスポートابل・テーブルスペース」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。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送る場合。
- 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。
- 内部と外部のカスタマーへのデータ・パブリッシングを行う場合。

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すむので、同じデータをインポート / エクスポートしたりアンロード / ロードするより、トランスポートابل・テーブルスペースを介してデータを移動する方が、はるかに高速です。また、トランスポートابل・テーブルスペースを使用して索引データを移動し、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。

表領域セットを移動またはコピーするには、次のタスクを実行する必要があります。

- [ステップ 1: 自己完結型の表領域セットの選択](#)
- [ステップ 2: トランスポートابل・テーブルスペースセットの生成](#)

「トランスポートابل・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。

■ ステップ 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（O/S コピー・ユーティリティ、ftp または CD 上でのパブリッシングなど）を使用できます。

■ ステップ 4: 表領域セットのプラグイン

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。

関連項目: トランスポートابل・テーブルスペースと、データ・マートおよびデータ・ウェアハウジングでの使用の詳細は、『Oracle8i 概要』を参照してください。

トランスポートابل・テーブルスペースを使用してメディア回復を実行する方法については、『Oracle8i 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。

トランスポートابل・テーブルスペースの互換性（異なる Oracle リリース間）に関する問題については、『Oracle8i 移行ガイド』を参照してください。

現行の制限事項

トランスポートابل・テーブルスペースの計画を作成して使用する場合は、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してください。たとえば、Sun Solaris の Oracle データベース間、または 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 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・データベースは、データベース・ブロック・サイズを同じにしてください。
- ソース・データベースとターゲット・データベースでは、同じキャラクタ・セットを使用してください。
- 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。
- 現在、トランスポートابل・テーブルスペースは次のものをサポートしていません。
 - スナップショット / レプリケーション
 - ファンクション・ベースの索引
 - 有効範囲付き REF

- ドメイン・インデックス（拡張可能な索引作成機能が提供する新しいタイプの索引）
- 8.0 互換の複数受信者を持つアドバンスト・キュー

ステップ 1: 自己完結型の表領域セットの選択

トランスポートできるのは、自己完結型の表領域セットだけです。この場合、「自己完結型」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。たとえば、表領域セット内に、その外部にある表の索引がある場合、そのセットは自己完結型ではありません。

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。

表領域セットをトランスポートするときには、参照の整合性制約を含めるかどうかを選択できます。ただし、その場合は、表領域セットが自己完結型かどうかを判断しなければなりません。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。次に、自己完結型の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。
- 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。
- 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、表領域名のリストを指定し、参照の整合性制約をトランスポートするように指定して、組込み PL/SQL プロシージャを起動できます。たとえば、表領域 `ts1` および `ts2` が自己完結型かどうかを判断する場合を考えます（制約を考慮に含めます）。次のコマンドを発行できます。

```
execute dbms_tts.transport_set_check('ts1,ts2', TRUE)
```

この場合、`transport_set_check` は PL/SQL パッケージ `DBMS_TTS` 内の PL/SQL ルーチンで、次のプロトタイプを持ちます。

```
PROCEDURE transport_set_check(ts_list IN varchar2, incl_constraints IN boolean)
```

`ts_list` - コンマによって区切られる表領域名のリスト

`incl_constraints` - 制約を考慮に含める場合は `TRUE` を、含めない場合は `FALSE` を指定する。

この PL/SQL ルーチン呼び出すと、`TRANSPORT_SET_VIOLATIONS` ビューから選択してすべての違反を表示できます。表領域セットが自己完結型であれば、このビューは空になります。自己完結型でなければ、このビューにはすべての違反が表示されます。たとえば、2 つの違反がある場合を考えます。1 つ目は表領域セットの境界をまたがる外部キー定数 `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 `sales` です。`TRANSPORT_SET_VIOLATIONS` を問い合わせると、次の結果が戻されます。

```
select * from transport_set_violations;
```

VIOLATIONS

 Constraint DEPT_FK between table JIM.EMP in tablespace FOO and table JIM.DEPT in
 tablespace OTHER
 Partitioned table JIM.SALES is partially contained in the transportable set

表領域セットを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 の後の問合せではユーザー・エラーを示
 します。

関連項目：REF の詳細は、『Oracle8i 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。

ステップ 2: トランスポートابل・テーブルスペースセットの生成

トランスポートする自己完結型の表領域セットを識別した後に、次のタスクを実行してトラ
 ンスポートابل・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み取り専用にします。もちろん、表領域がすで
 に読み取り専用になっていれば、このステップを実行する必要はありません。

```
ALTER TABLESPACE sales READ ONLY;
```

2. エクスポート・ユーティリティを起動し、トランスポートابل・セットに含める表領域
 を次のように指定します。

```
EXP TRANSPORT_TABLESPACE=y TABLESPACES=sales_1,sales_2  

  TRIGGERS=y/n CONSTRAINTS=y/n GRANTS=y/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、エクスポートさ
 れるのはデータ・ディクショナリの構造情報だけです。したがって、この
 操作は大型の表領域の場合にも高速です。

プロンプトが表示されたら、"sys as sysdba" として接続します。

必ず TABLESPACES を指定してください。FILE パラメータでは、作成する構造情報エ
 クスポート・ファイルの名前を指定します。

TRIGGERS=n に設定すると、トリガーはエクスポートされません。TRIGGERS=y に設
 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、後続のインポート中にコンパイル・エラーになります。

GRANTS=y に設定すると、エクスポートされる表に対するすべての権限付与もエクス
 ポートされますが、それ以外の場合はすべての GRANTS が無視されます。

CONSTRAINTS=y に設定すると、参照の整合性制約がエクスポートされ、それ以外の場合はこの制約は無視されます。

この3つのオプションのデフォルト設定は、いずれも 'y' です。

3. データ・ファイルを別の記憶領域またはターゲット・データベースにコピーします。
4. 必要であれば、コピーしたセット内の表領域を、次のように読み書きモードに戻します。

```
ALTER TABLESPACE sales_1 READ WRITE;
```

トランスポートする表領域セットが自己完結型でなければ、エクスポートは失敗し、トランスポートابل・セットが自己完結型でないことを示します。その場合は、ステップ1に戻ってすべての違反を解決する必要があります。

ステップ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（O/S コピー・ユーティリティ、ftp または CD 上でのパブリッシングなど）を使用できます。

ステップ4: 表領域セットのプラグイン

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. コピーした表領域セットのデータ・ファイルを、ターゲット・データベースからアクセスできる位置に配置します。
2. 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 DATAFILES='/db/sales_jan','/db/sales_feb',...fn
TABLESPACES=sales_1,sales_2,... TTS_OWNERS=dcranney,jfee
FROMUSER=dcranney,jfee TOUSER=smith,williams FILE=expdat.dmp
```

プロンプトが表示されたら、"sys as sysdba" として接続します。

次に2つの例を示します。

```
IMP TRANSPORT_TABLESPACE=y DATAFILES='(/db/staging1.f,/db/staging2.f)'
```

```
IMP TRANSPORT_TABLESPACE=y DATAFILES='/db/staging.f' TABLESPACES=jan OWNERS=smith
```

必ず DATAFILES を指定してください。

TABLESPACES、TTS_OWNERS、FROMUSER および TOUSER はオプションです。FILE パラメータでは、構造情報エクスポート・ファイルの名前を指定します。

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、エラーが戻されます。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 では、表領域セット内のデータを所有するユーザー全員がリストされます。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、エラーが戻されます。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。

FROMUSER と TOUSER を指定しなければ、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する必要があります。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戻されます。

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。たとえば、FROMUSER=dcranney,jfee TOUSER=smith, williams と指定した場合、ソース・データベース内で dcranney に所有され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 に所有され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・データベースには、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み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する必要があります。この時点で ALTER TABLESPACE...READ WRITE 文を発行し、新しい表領域を読み書き専用モードにすることができます。

多数のデータ・ファイルを扱う場合、コマンド行でデータ・ファイル名のリストを指定するのは煩雑だけでなく、コマンド行の制限を超えることもあります。この場合は、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、このステップのコマンドの 1 つが次のようになっている場合を考えます。

```
IMP PARFILE='par.f'
```

ファイル par.f の内容は、次のとおりです。

```
TRANSPORT_TABLESPACE=y
DATAFILES=/db/staging.f
TABLESPACES=jan
TT_OWNERS=smith
```

データベース間で表領域をトランスポートするには、ソース・データベースとターゲット・データベースの両方を Oracle8i で稼働させ、init.ora の互換性パラメータを 8.1 に設定する必要があります。

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
- アドバンスド・キュー
- 索引
- トリガー
- スナップショット / 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）プラグインされている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 が表領域セットは自己完結型であると判断した場合、REF はチェックされません。その結果、プラグインされている表領域には、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 の後の問合せでは、ユーザー・エラーが戻されます。

権限

エクスポート時に GRANTS=y と指定すると、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ければ、そのパーティション表はトランスポートブル・テーブルスペースを介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れば、表を元の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る場合は、変換操作が成功することは保証されています。まれに、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戻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戻される場合は、ALTER TABLE MOVE PARTITION 文を使用してパーティションの方を移動できます。その後で、変換操作を再試行してください。

変換文の 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戻されます。ただし、パーティション内のデータはコピーできるので、パーティションの移動が低速になることがあります。パーティション表の使用例については、9-26 ページの「[データ・ウェアハウジングのためのパーティションのトランスポートと連結例](#)」を参照してください。

オブジェクト

トランспортаブル・テーブルスペースには、次のオブジェクトを含めることができます。

- 表
- 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 互換キューでなければ、これらのキューはトランспортаブル・テーブルスペースを使用して移動またはコピーできます。キューをターゲット・データベースにトランスポートすると、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み込み / 書き込みにしてから、組込み式の PL/SQL ルーチン `dbms_aqadm.start_queue()` を介して起動すると、キューを使用可能にすることができます。

索引

通常の索引とビットマップ索引をトランスポートできます。トランスポートابل・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・ベースの索引とドメイン索引は、サポートされません。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トابل・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。

スナップショット / レプリケーション

スナップショット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、サポートされません。トランスポートする表領域内で表がレプリケートされている場合は、表領域をトランスポートする前に、そのレプリケーションの構造情報を削除し、表を通常の表に変換する必要があります。

データ・ウェアハウジングのためのパーティションのトランスポートと連結例

標準的な企業のデータ・ウェアハウスには、1 つ以上の大きい事実表が含まれています。これらの事実表は、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、スター問合せをスピードアップできます。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すむように、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」という大型の事実表があり、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、ローカルの非同一次索引を作成します。

```
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ものとします。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、ローカルの非同一次索引にもう1つのパーティションも作成されます。

```
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すむので、結果は即時に戻されます。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、この文では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、通常、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戻されることがあります。

```
ORA-19728: data object number conflict between table JUL_SALES and partition JAN98 in  
table SALES
```

この競合を解決するには、次の文を発行して違反元の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）。したがって、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となります。

構造化データの CD への発行

トランスポートابل・テーブルスペースを使用すると、構造化データを CD に発行できます。データ提供者は、発行するデータを持つ表領域をロードし、トランスポートابل・セットを生成し、トランスポートابل・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、NT マシンの D: ドライブが CD ドライブであるものとします。次のように、データ・ファイル `catalog.f` とエクスポート・ファイル `expdat.dmp` を持つトランスポートابل・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。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戻されます。ただし、このデータ・ファイル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、表領域は再び読み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ないことと、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わせる時にのみ、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久的に連結されない限り、常に `READ_ONLY_OPEN_DELAYED` 初期化パラメータを `TRUE` に設定する必要があります。

複数データベースの同じ表領域の読取り専用でのマウント

トランスポートابل・テーブルスペースを使用すると、表領域を複数のデータベースに読取り専用でマウントできます。これにより、個々のデータベースがデータを別々のディスクに複製するかわりに、ディスク上の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。

同じ表領域を複数のデータベースに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。この表領域を読み取り専用にしてトランスポータブル・セットを生成し、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。Oracle Parallel Server で要求されるように、クラスタ化されたファイル・システムまたはロー・ディスクを使用できます。Oracle は共有ディスク上でこの種のデータ・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わせると、NFS 操作がタイムアウトになるまでデータベースがハングすることがあります。

後で、一部のデータベースから読み取り専用表領域を削除できます。削除しても、表領域のデータ・ファイルは変更されないで、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、表領域は読み書き可能にしないでください。

トランスポータブル・テーブルスペースを介した履歴データのアーカイブ

トランスポータブル・テーブルスペースセットは、任意の Oracle データベースにプラグインできる自己完結型のファイル・セットなので、この章で説明するトランスポータブル・テーブルスペースの手順を介して、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、『Oracle8i 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。

トランスポータブル・テーブルスペースを使用した TSPITR の実行

トランスポータブル・テーブルスペースを使用して、表領域の Point-in-Time 回復 (TSPITR) を実行できます。

関連項目 : トランスポータブル・テーブルスペースを使用して TSPITR を実行する方法については、『Oracle8i 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。

表領域についての情報の表示

次のデータ・ディクショナリ・ビューは、データベースの表領域に関して有益な情報を提供します。

- USER_EXTENTS、DBA_EXTENTS
- USER_SEGMENTS、DBA_SEGMENTS
- USER_FREE_SPACE、DBA_FREE_SPACE
- DBA_USERS
- DBA_TS_QUOTAS
- USER_TABLESPACES、DBA_TABLESPACES
- DBA_DATA_FILES
- V\$DATAFILE

このマニュアルの他の章には記述されていないビューの使用方法を次の例で説明します。データベースには、2つの表領域 SYSTEM と USERS が含まれているものとします。USERS は2つのファイル、FILE1（100MB）と FILE2（200MB）から構成されます。表領域は通常のアフライン状態になっています。

表領域とデフォルト記憶領域パラメータの記述例

データベース内のすべての表領域の名前とデフォルト記憶領域パラメータをすべて記述するには、DBA_TABLESPACES ビューに対して次の問合せを使用してください。

```
SELECT tablespace_name "TABLESPACE",
       initial_extent "INITIAL_EXT",
       next_extent "NEXT_EXT",
       min_extents "MIN_EXT",
       max_extents "MAX_EXT",
       pct_increase
FROM sys.dba_tablespaces;
```

TABLESPACE	INITIAL_EXT	NEXT_EXT	MIN_EXT	MAX_EXT	PCT_INCREASE
SYSTEM	10240000	10240000	1	99	50
USERS	10240000	10240000	1	99	50

データ・ファイルとデータベースの対応する表領域の記述例

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記述するには、DBA_DATA_FILES ビューに対して次の問合せを入力してください。

```
SELECT file_name, bytes, tablespace_name
FROM sys.dba_data_files;
```

FILE_NAME	BYTES	TABLESPACE_NAME
filename1	10240000	SYSTEM
filename2	10240000	USERS

filename3 20480000 USERS

各表領域の空き領域（エクステンツ）の記述例

データベース内の各表領域の使用可能エクステンツ内で利用可能な領域の容量に関する情報を記述するには、次の問合せを入力してください。

```
SELECT tablespace_name "TABLESPACE", file_id,
COUNT(*)             "PIECES",
MAX(blocks)            "MAXIMUM",
MIN(blocks)            "MINIMUM",
AVG(blocks)            "AVERAGE",
SUM(blocks)            "TOTAL"
FROM sys.dba_free_space
WHERE tablespace_name = 'SYSTEM'
GROUP BY tablespace_name, file_id;
```

TABLESPACE	FILE_ID	PIECES	MAXIMUM	MINIMUM	AVERAGE	TOTAL
SYSTEM	1	2	2928	115	1521.5	3043

TOTAL は各表領域の使用可能領域の総量、PIECES は表領域のデータ・ファイル内の断片化の総量、MAXIMUM は最大の連続領域を示します。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たいときに、この問合せを使用します。

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。この章のトピックは、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・ファイルのサイズを変更
- データ・ファイルの可用性の変更
- データ・ファイルの改名および再配置
- データ・ファイル内のデータ・ブロックの検証
- データ・ファイルの情報の表示

関連項目：データ・ファイルは、メディア障害時にデータベース回復処理の一部として作成されることもあります。詳細は、『Oracle8i 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・ファイルを管理するためのガイドライン

ここでは、データ・ファイルの管理について説明します。この項のトピックは、次のとおりです。

- データ・ファイル数の判断
- 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・ファイルと分離したデータ・ファイルの格納

すべてのデータ・ファイルには、「絶対ファイル番号」と「相対ファイル番号」という2つのファイル番号が対応付けられています。

絶対ファイル番号は、データベース内のデータ・ファイルを固有に識別するものです。Oracle8 より前のバージョンでは、絶対ファイル番号が単に「ファイル番号」と呼ばれていました。

相対ファイル番号は、データ・ファイルを表領域内で固有に識別するものです。多くの場合、小規模および中規模のサイズのデータベースでは、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）を超えると、相対ファイル番号と絶対ファイル番号が違ってきます。相対ファイル番号は、多くのデータ・ディクショナリ・ビューで使用されています。

データ・ファイル数の判断

データベースの SYSTEM 表領域に対しては、データ・ファイルが最低1つ必要です。小規模システムでは、データ・ファイルを1つだけ持つこともあります。一般に、小さいデータ・ファイルを多数作成するよりも、大きいデータ・ファイルを少数だけ作成し、同時にオープンする必要があるファイルを少なくすることを推奨します。

オペレーティング・システム固有の制限に応じて、表領域にデータ・ファイルを追加することもできます。

オペレーティング・システムの制限	各オペレーティング・システムには、プロセスあたりのオープン・ファイルの最大数に制限があります。他の制限にかかわらず、オープンしている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、データ・ファイルを作成することができません。
------------------	---

Oracle システムの制限	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。この制限はポートによって異なります。
----------------	--

制御ファイルの上限

CREATE DATABASE 文または CREATE CONTROLFILE 文を発行するとき、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が指定されます。あとでファイルを追加したとき、そのファイルの番号が MAXDATAFILES より大きく、かつ DB_FILES 以下なら、制御ファイルは、データ・ファイル部分にもっと多くのファイルが入るように自動的に拡張されます。

インスタンスまたは SGA の上限

Oracle8 インスタンスを起動すると、データベースのパラメータ・ファイルがデータ・ファイルの情報用に予約されている SGA 領域の容量を示します。つまり、データ・ファイルの最大数は DB_FILES パラメータによって制御されます。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だけです。

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。

Oracle Parallel Server では、すべてのインスタンスのデータ・ファイルの上限は同じ値に設定されている必要があります。

DB_FILES の値を決めるときは、次のことを考慮してください。

- DB_FILES の値が低すぎると、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なくなります。
- DB_FILES の値が多すぎると、メモリーが不必要に消費されてしまいます。

理論上、Oracle データベースが処理できるデータ・ファイルの数に制限はありません。しかし、データ・ファイルの数を決めるときは、次のことを考慮してください。

- 小さいデータ・ファイルを多数処理するより、データ・ファイルの数を少なくする方がパフォーマンスが向上します。さらに、ファイルが大きいほうが回復可能単位をきめ細かく設定できます。
- ほとんどの場合、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。Oracle の DBW0 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。さらに、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もあります。

Oracle のデータベース内のデータ・ファイルの数は、オペレーティング・システムで定義されている制限より大きくすることができます。その場合、パフォーマンスは落ちることがあります。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。

1 つの表領域の中のデータ・ファイルの最大数に関するオペレーティング・システム固有の制限は、ほとんどの場合、1023 ファイルです。

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。

Parallel Server のオペレーティング・システムの制限の詳細は、『Oracle8i Parallel Server 概要および管理』を参照してください。

MAXDATAFILES の詳細は、『Oracle8i 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（オリジナルの SYSTEM 表領域内の）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 7MB が必要です。他の Oracle 製品をインストールする場合は、これらの製品には SYSTEM 表領域内の追加領域が（たとえば、オンライン・ヘルプ用の領域など）さらに必要となります。これらの製品の詳細は、インストレーションの指示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定されます。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に利用できるディスク・ドライブがいくつかある場合には、1 つのディスク・ドライブ上の表領域に表データを格納し、別のディスク・ドライブ上の表領域に索引データを格納すると有効です。このようにユーザーが表情報を問い合わせることにより、両方のディスク・ドライブが同時に作動して、表データと索引データの両方を同時に検索し、データベース・システムのパフォーマンスを改善できます。

REDO ログ・ファイルと分離したデータ・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回復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）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。

表領域を作るときにデータベース管理者がデータベース・オブジェクトの潜在的なサイズを見積って、十分なファイルまたはデバイスを追加しておくことが理想です。そのようにすれば、データがすべてのデバイスに均等に分散します。

表領域にデータ・ファイルを追加するには、Oracle Enterprise Manager/GUI の「Add Datafile」ダイアログ・ボックスを使用するか、または SQL コマンド ALTER TABLESPACE を実行します。データ・ファイルを表領域に追加するには、ALTER TABLESPACE システム権限が必要です。

次の文では、RB_SEGS 表領域の新しいデータ・ファイルが作成されます。

```
ALTER TABLESPACE rb_segs  
  ADD DATAFILE 'filename1' SIZE 1M;
```

表領域に新しいデータ・ファイルを追加して、ファイル名を完全に指定しないと、データ・ファイルはデータベース・サーバーのデフォルト・ディレクトリに作成されます。既存のファイルを再使用しない場合は、必ず他のファイルと重複しないファイル名を新しく指定してください。そうしないと、前に削除した旧ファイルが上書きされます。

データ・ファイルのサイズを変更

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。この項のトピックは、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

データベースに領域がさらに必要になった場合に自動的にサイズを大きくできるようにデータ・ファイルを作成したり、既存のデータ・ファイルを変更できます。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、指定の増分値で大きくなります。

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、次のような利点があります。

- 表領域が足りなくなった場合に即時中断をする必要性が減ります。
- エクステンツ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を知るには、DBA_DATA_FILES ビューに問合せ、AUTOEXTENSIBLE 列を調べます。

次の SQL コマンドを使用して、データ・ファイルを作成するときに自動ファイル拡張機能を指定できます。

- 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、SQL 文 ALTER DATABASE を使用します。

次の例は、USERS 表領域に追加されたデータ・ファイル FILENAME2 の自動拡張機能を使用可能にします。

```
ALTER TABLESPACE users
  ADD DATAFILE 'filename2' 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 FILENAME2 の自動拡張機能を使用禁止にします。

```
ALTER DATABASE DATAFILE 'filename2'
  AUTOEXTEND OFF;
```

関連項目: データ・ファイルの作成または変更のための SQL 文の詳細は、『Oracle8i 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 コマンドを使用します。

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。データベース内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、このコマンドが有効です。

また、データ・ファイルのサイズを手動で小さくして、データベース内の未使用の領域の再生もできます。これは必要な領域の容量見積りを誤った場合にこれを訂正するときに有効です。

この例では、データ・ファイル FILENAME2 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小さくできます。

次のコマンドは、データ・ファイル FILENAME2 のサイズを小さくします。

```
ALTER DATABASE DATAFILE 'filename2'  
    RESIZE 100M;
```

注意： 常にファイルのサイズを指定した値まで小さくできるわけではありません。

関連項目： ダウングレードに対してリサイジング・ファイルが持つ意味の詳細は、『Oracle8i 移行ガイド』を参照してください。

ALTER DATABASE 文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

データ・ファイルの可用性の変更

ここでは、データ・ファイルの可用性を変更するさまざまな方法について説明します。この項のトピックは、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にする方法
- NOARCHIVELOG モードでデータ・ファイルをオフラインにする方法

非常にまれな状況において、特定のデータ・ファイルをオンライン（利用できる状態）にしたり、特定のファイルをオフライン（利用できない状態）にする必要があるかもしれません。たとえば、データ・ファイルへの書込みに問題がある場合、Oracle はそのデータ・ファイルを自動的にオフラインにすることができます。このような場合メディア回復の前に、破損したデータ・ファイルを手動でオンラインにしたり、オフラインにする必要があるかもしれません。

注意： 表領域をオフラインにすることによって、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以外）を一時的に使用不能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、表領域内のファイルはそのままにしておく必要があります。

オフライン・データ・ファイルにはアクセスできません。読取り専用表領域のデータ・ファイルをオンラインにすると、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み書き可能な状態に戻らない限り、このファイルには書き込めません。読取り専用表領域のファイルを個別にオンラインまたはオフラインにするには、ALTER DATABASE コマンドの DATAFILE オプションを使用します。

データ・ファイルをオンラインにしたり、オフラインにしたりするには、どちらのアーカイブ・モードであっても、ALTER DATABASE システム権限が必要です。データベースが排他モードでオープンされている場合に限り、これらの操作を実行できます。

ARCHIVELOG モードでデータ・ファイルをオンラインにする方法

個々のデータ・ファイルをオンラインにするには、SQL 文 ALTER DATABASE を発行し、DATAFILE パラメータを入力します。

注意： ALTER DATABASE 文のこのオプションを使用するには、データベースを ARCHIVELOG モードにしてください。一方、NOARCHIVELOG モードでデータ・ファイルをオフラインにするとファイルが失われる可能性があります。ARCHIVELOG モードで、データ・ファイルの不慮の損失を防止できます。

次の文は、指定したデータ・ファイルをオンラインにします。

```
ALTER DATABASE DATAFILE 'filename' ONLINE;
```

関連項目：メディア回復時にデータ・ファイルをオンラインにする方法の詳細は、『Oracle8i 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。

NOARCHIVELOG モードでデータ・ファイルをオフラインにする方法

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にするには、DATAFILE パラメータと OFFLINE DROP オプション付きの ALTER DATABASE コマンドを使用します。これによって、ただちにデータ・ファイルをオフラインにし、削除できます。たとえば、データベースは NOARCHIVELOG モードで運用されていて、データ・ファイルが、一時セグメントからのデータだけを含み、バックアップされていない場合に便利です。

次の文は、指定したデータ・ファイルをオフラインにします。

```
ALTER DATABASE DATAFILE 'filename' OFFLINE DROP;
```

データ・ファイルの改名および再配置

ここでは、データ・ファイルの改名と再配置について説明します。この項のトピックは、次のとおりです。

- [単一の表領域のデータ・ファイルの改名および再配置](#)
- [複数の表領域のデータ・ファイルの改名および再配置](#)

データ・ファイルを改名して、それらの名前や位置を変更できます。Oracle には、次のような変更のためのオプションが用意されています。

- データベースの残りの部分をオープンした状態での、単一のオフライン表領域内のデータ・ファイル（たとえば、TBSP1 内の FILENAME1 と FILENAME2）の改名、再配置。
- データベースをマウントしクローズしたままの状態での、複数の表領域内のデータ・ファイル（たとえば、TBSP1 内の FILE1 と TBSP2 内の FILE2）の同時改名、同時再配置。

注意： SYSTEM 表領域のデータ・ファイルを改名または再配置する場合、SYSTEM 表領域はオフラインにできないため、前述の后者のオプションを使用する必要があります。

これらの手順によるデータ・ファイルの改名と再配置は、データベースの制御ファイルに記録される、データ・ファイルへのポインタのみを変更します。オペレーティング・システム・ファイルを物理的に改名したり、オペレーティング・システム・レベルでファイルをコピーするわけではありません。そのために、データ・ファイルの改名と再配置には、複数のステップが必要になります。この手順を実行する前に、ステップと例題をよく理解してください。

単一表領域のデータ・ファイルを改名するには、ALTER TABLESPACE システム権限が必要です。

単一の表領域のデータ・ファイルの改名および再配置

単一の表領域のデータ・ファイルを改名または再配置する手順

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にします。
2. オペレーティング・システム・コマンドを使用して、データ・ファイルを新しい位置または新しい名前にコピーします。
3. 指定された新しい完全なファイル名が、古いファイル名と異なることを確認します。
4. SQL 文 ALTER TABLESPACE で RENAME DATAFILE オプションを使用して、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 FILENAME1 と FILENAME2 を FILENAME3 と FILENAME4 にそれぞれ改名します。

```
ALTER TABLESPACE users
  RENAME DATAFILE 'filename1', 'filename2'
  TO 'filename3', 'filename4';
```

新しいファイルがすでに存在している必要があります。このコマンドではファイルは作られません。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。

複数の表領域のデータ・ファイルの改名および再配置

1 つ以上の表領域のデータ・ファイルは、RENAME FILE オプションを指定した SQL コマンド ALTER DATABASE を使用して、改名、再配置できます。一回の操作で、複数の表領域のデータ・ファイルを改名、再配置する場合、または SYSTEM 表領域のデータ・ファイルを改名、再配置する場合、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かなければならない場合、前ページで説明した手順を検討してください。

一回の操作で、複数の表領域のデータ・ファイルを改名する、または SYSTEM 表領域のデータ・ファイルを改名するには、ALTER DATABASE システム権限が必要です。

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・コマンドで新しい位置と名前を指定して、改名するデータ・ファイルをコピーします。
3. 新しくコピーしたデータ・ファイル名が、現在使用しているデータ・ファイルと違う、指定された完全なファイル名になっていることを確認します。
4. SQL 文 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・ポインタを改名します。

たとえば、次の文は、データ・ファイル FILENAME1 と FILENAME2 を FILENAME3 と FILENAME4 にそれぞれ改名します。

```
ALTER DATABASE
  RENAME FILE 'filename1', 'filename2'
  TO 'filename3', 'filename4';
```

新しいファイルがすでに存在している必要があります。このコマンド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。

データ・ファイルの再配置例

次のような条件を想定します。

- オープンしているデータベースには、USERS という表領域が存在し、コンピュータの同じディスク上に位置する 1 つ以上のデータ・ファイルによって構成されています。
- USERS 表領域のデータ・ファイルが別のディスク・ドライブに再配置されます。
- 現在 Enterprise Manager を使用しており、管理者権限でオープン・データベースに接続しています。

データ・ファイルを再配置する手順

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 を問い合わせると、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）が記述されます。

```
SELECT file_name, bytes FROM sys.dba_data_files
       WHERE tablespace_name = 'USERS';
FILE_NAME          BYTES
-----
FILENAME1          102400000
FILENAME2          102400000
```

FILENAME1 と FILENAME2 は完全に指定した 2 つのファイル名であり、それぞれのサイズは 1MB です。

2. データベースのバックアップを作成します。
1 つ以上の表領域のデータ・ファイルを改名、再配置するなど、データベースに対する構造上の変更を実行する前に、この操作中なんらかの問題が発生するのに備えて、データベースを完全にバックアップしておくことが非常に重要です。
3. データ・ファイルが入っている表領域をオフラインにするか、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。
4. オペレーティング・システム・コマンドを使用して、データ・ファイルを新しい位置にコピーします。たとえば、既存のファイル FILENAME1 と FILENAME2 を、それぞれ FILENAME3 と FILENAME4 にコピーします。

注意： オペレーティング・システムの HOST コマンドを実行すると、ファイルをコピーできます。

5. Oracle 内のデータ・ファイルを改名します。

USERS 表領域を構成するファイルに対するデータ・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 FILENAME1 と FILENAME2 から FILENAME3 と FILENAME4 にそれぞれ変更してください。

表領域がオフラインになっても、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。

6. 表領域をオンラインにするか、データベースを停止してから、再起動します。

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、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たが、クローズしている場合は、データベースをオープンしてください。

7. データベースのバックアップを作成します。データベースの構造を変更した後は、データベースの即時完全バックアップを実行してください。

関連項目: DBA_DATA_FILES データ・ディクショナリ・ビューの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

表領域をオフラインにする方法の詳細は、9-10 ページの「[表領域をオフラインにする方法](#)」を参照してください。

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、動的に変更でき、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。

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。

DBW0 プロセスは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・ロード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、ブロックの破損を検出するためにチェックサムを使用します。破損が検出されると、メッセージ ORA-01578 が戻され、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。

警告: 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・サポートの指示があった場合のみにしてください。

データ・ファイルの情報の表示

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・ファイルに関して有益な情報を提供します。

- USER_EXTENTS、DBA_EXTENTS
- USER_SEGMENTS、DBA_SEGMENTS
- USER_FREE_SPACE、DBA_FREE_SPACE
- DBA_USERS
- DBA_TS_QUOTAS
- USER_TABLESPACES、DBA_TABLESPACES
- DBA_DATA_FILES
- V\$DATAFILE

次の例では、このマニュアルの他の章でまだ説明されていないビューの使用方法を説明します。データベースには、2つの表領域 SYSTEM と USERS が含まれているとします。USERS は2つのファイル、FILE1(100MB) と FILE2(200MB) から構成されます。表領域は通常のオンライン状態になっています。ここで、データベースのデータ・ファイルに関する状態情報を表示するには、V\$DATAFILE を問い合わせます。

```
SELECT name,
       file#,
       status,
       checkpoint_change# "CHECKPOINT" FROM v$datafile;
```

NAME	FILE#	STATUS	CHECKPOINT
-----	----	-----	-----
filename1	1	SYSTEM	3839
filename2	2	OFFLINE	3782
filename3	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られる、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、ファイルが回復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になっている場合、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になっている場合、このファイルの STATUS は OFFLINE または RECOVER になります。CHECKPOINT は、データ・ファイルの最も最近のチェックポイントに対して書き込まれた最後の SCN を示します。

データベース・リソース・マネージャの使用

この章では、データベース・リソース・マネージャの使用方法について説明します。この章のトピックは、次のとおりです。

- [データベース・リソース・マネージャ・パッケージの使用](#)
- [データベース・リソース・マネージャのビュー](#)

概要

通常、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（OS）に左右される場合は、次のような問題が発生することがあります。

- 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、サーバー数が多い場合に Oracle サーバー間で OS コンテキストが切り替わることです。
- スケジューリングが十分でない問題。
OS は、ラッチを保持している間に Oracle サーバーのスケジュールを解除するため、非効率的です。
- リソースの不十分なパーティション化
OS は、重要度の異なるタスク間では CPU リソースを適切にパーティション化できません。
- パラレル・スレーブやアクティブ・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問題。

Oracle のデータベース・リソース・マネージャは、データベース管理者が指定したリソース・プランに基づいてリソースを割り当てます。また、最終的には、データベース・リソース・マネージャによってリソース管理上の意思決定を厳密に管理し、不十分な OS スケジューリングによる問題に対処できます。

管理者は、[表 11-1](#) に示すデータベース・リソース・マネージャの基本要素を使用します。

表 11-1 データベース・リソース・マネージャの要素

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・グループに割り当てるリソースを指定するダイレクティブを含みます。
リソース・アロケーション・メソッド	データベース・リソース・マネージャで、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。
リソース・プラン・ダイレクティブ	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パーティション化するために使用するダイレクティブ。

関連項目: データベース・リソース・マネージャの概念の詳細は、『Oracle8i 概要』を参照してください。

データベース・リソース・マネージャ・パッケージの使用

リソース・プランとリソース・コンシューマ・グループを作成するには、次のパッケージを使用します。

- DBMS_RESOURCE_MANAGER
- DBMS_RESOURCE_MANAGER_PRIVS

DBMS_RESOURCE_MANAGER パッケージの使用

DBMS_RESOURCE_MANAGER パッケージを使用して、リソース・プラン、リソース・コンシューマ・グループおよびプラン・ダイレクティブをメンテナンスします。また、このパッケージを使用し、プラン・スキーマの変更をまとめてグループ化することもできます。

データベース・リソース・マネージャを管理するには、SYSTEM 権限が必要です。通常、管理者は、この SYSTEM 権限と ADMIN オプションを持っています。この SYSTEM 権限の付与と取消しに使用するプロシージャは、次のとおりです。

```
grant_system_privilege(grantee_name in varchar2,admin_option in boolean)
```

```
revoke_system_privilege (revokee_name in varchar2)
```

リソース・プランの管理

注意: リソース・マネージャ・オブジェクトを作成する前に、ペンディング・エリアを作成する必要があります。詳細は、11-5 ページの「[ペンディング・エリアの作成と管理](#)」を参照してください。

次のプロシージャを使用して、リソース・プランを作成、更新または削除できます。

```
create_plan(plan in varchar2, comment in varchar2,  
  cpu_mth in varchar2 DEFAULT 'EMPHASIS',  
  max_active_sess_target_mth in varchar2 DEFAULT  
  'MAX_ACTIVE_SESS_ABSOLUTE',  
  parallel_degree_limit_mth in varchar2 DEFAULT  
  'PARALLEL_DEGREE_LIMIT_ABSOLUTE')  
update_plan(plan in varchar2, new_comment in varchar2)  
  DEFAULT NULL, new_cpu_mth in varchar2  
  DEFAULT NULL, new_max_active_sess_target_mth in  
  varchar2 DEFAULT NULL,  
  new_parallel_degree_limit_mth in varchar2
```

```
DEFAULT NULL)  
delete_plan(plan in varchar2)  
delete_plan_cascade(plan in varchar2)
```

`delete_plan` プロシージャでは、指定したプランと、それが参照するすべてのプラン・ダイレクティブが削除されます。`delete_plan_cascade` プロシージャでは、指定したプランと、そのすべての子孫（プラン・ダイレクティブ、サブプラン、リソース・コンシューマ・グループ）が削除されます。`delete_plan_cascade` にエラーが発生するとロールバックされ、プラン・スキーマは未変更のまま残ります。

`update_plan` プロシージャに引数を指定しなければ、データ・ディクショナリにそのまま残ります。

デフォルトのリソース・アロケーション・メソッドを使用する場合、プランの作成または更新時に指定する必要はありません。デフォルトの方法では、次のように設定されます。

- `cpu_method = 'EMPHASIS'`
- `parallel_degree_limit_mth = 'PARALLEL_DEGREE_LIMIT_ABSOLUTE'`

リソース・コンシューマ・グループの管理

次のプロシージャを使用して、リソース・コンシューマ・グループを作成、更新または削除できます。

```
create_consumer_group(consumer_group in varchar2,  
    comment in varchar2, cpu_mth in varchar2  
    DEFAULT 'ROUND-ROBIN')  
update_consumer_group(consumer_group in varchar2,  
    new_comment in varchar2 DEFAULT NULL,  
    new_cpu_mth in varchar2 DEFAULT NULL)  
delete_consumer_group(consumer_group in varchar2)
```

デフォルトの CPU アロケーション・メソッドである ROUND-ROBIN を使用する場合、`cpu_mth` パラメータを指定する必要はありません。

`update_consumer_group` プロシージャに引数を指定しなければ、データ・ディクショナリにそのまま残ります。

リソース・プラン・ダイレクティブの管理

次のプロシージャを使用して、リソース・プラン・ダイレクティブを作成、更新または削除できます。

```
create_plan_directive(plan in varchar2, group_or_subplan  
    in varchar2, comment in varchar2, cpu_pl in number  
    DEFAULT NULL, cpu_p2 in number  
    DEFAULT NULL, cpu_p3 in number  
    DEFAULT NULL, cpu_p4 in number  
    DEFAULT NULL, cpu_p5 in number)
```

```

DEFAULT NULL, cpu_p6 in number
DEFAULT NULL, cpu_p7 in number
DEFAULT NULL, cpu_p8 in number
DEFAULT NULL, max_active_sess_target_p1 in number
DEFAULT NULL, parallel_degree_limit_p1 in number DEFAULT NULL)
update_plan_directive(plan in varchar2, group_or_subplan
    in varchar2, new_comment in varchar2
DEFAULT NULL, new_cpu_p1 in number
DEFAULT NULL, new_cpu_p2 in number
DEFAULT NULL, new_cpu_p3 in number DEFAULT NULL, new_cpu_p4 in number
DEFAULT NULL, new_cpu_p5 in number DEFAULT NULL, new_cpu_p6 in number
DEFAULT NULL, new_cpu_p7 in number DEFAULT NULL, new_cpu_p8 in number
DEFAULT NULL, max_active_sess_target_p1 in number
DEFAULT NULL, new_parallel_degree_limit_p1 in number
DEFAULT NULL)
delete_plan_directive(plan in varchar2, group_or_subplan
    in varchar2)

```

どのパラメータも、デフォルト値は NULL です。

`update_plan_directive` プロシージャに引数を指定しなければ、データ・ディクショナリにそのまま残ります。

ペンディング・エリアの作成と管理

プラン・スキーマに対するすべての変更は、プラン・スキーマ変更用の「スクラッチ」であるペンディング・エリア内で実行できます。このペンディング・エリアを作成し、必要に応じて変更して、変更結果を送る必要があります（妥当性チェックはオプションです）。

次のプロシージャを使用し、データベース・リソース・マネージャを作成し、保留中の変更の妥当性をチェックして送ることができます。

```

dbms_resource_manager.create_pending_area

dbms_resource_manager.validate_pending_area

dbms_resource_manager.clear_pending_area

dbms_resource_manager.submit_pending_area

```

注意： 変更が有効になり、アクティブになるのは、
`submit_pending_area` プロシージャが正常終了した場合だけです。

また、ペンディング・エリアがアクティブになっている間に適切なユーザー・ビューから選択して、変更を含む現行のスキーマを表示することもできます。現行の変更は、ペンディン

グ・エリアを消去すればいつでも中止できます。変更が有効かどうかをチェックするには、`validate` プロシージャをコールします。

ペンディング・エリアでは、次の規則に従って変更を加える必要があります。

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・ダイ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・ダイレクティブが必要です。
4. EMPHASIS リソース・アロケーション・メソッドの場合、特定のレベルのすべてのパーセンテージの合計が 100 を超えないようにします。
5. アクティブなインスタンスで現在最上位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. プラン・ダイレクティブ・パラメータ `parallel_degree_limit_pl` は、リソース・コンシューマ・グループ（他のリソース・プランではなく）を参照するプラン・ダイレクティブでしか使用できません。
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。最上位プランのすべてのリーフをリソース・コンシューマ・グループにし、プラン・スキーマの最下位レベルのプラン・ダイレクティブはコンシューマ・グループを参照する必要があります。
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・ダイレクティブが存在する必要があります。これにより、現在アクティブになっているプランでカバーされないセッションには、OTHER_GROUPS ダイ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。

現在は使用されていないが、将来は使用する予定のリソース・コンシューマ・グループの作成を必要とする場合があるので、データベース・リソース・マネージャでは、「親なし」リソース・コンシューマ・グループ（それを参照するプラン・ダイレクティブがないリソース・コンシューマ・グループ）が許されます。

`validate` または `submit` プロシージャによるチェック中に、前述のいずれかの規則に違反していると、エラー・メッセージが表示されます。その場合は、変更を加えて問題を解決し、`validate` または `submit` プロシージャを再発行します。`submit_pending_area` で

は、変更の妥当性がチェックされ、コミットされた（有効な場合）後に、ペンディング・エリアが消去されます。

注意： `validate_pending_area` が正常終了しても、`submit_pending_area` のコールが異常終了することがあります。これが発生するのは、`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る場合などです。

次のコマンドでは、複数レベルのスキーマが作成され、[図 11-1](#) に示すデフォルトのプランとリソース・コンシューマ・グループの方法が使用されます。

```
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 bug users' sessions who run batch jobs');
dbms_resource_manager.create_consumer_group(consumer_group => 'Bug_Maintenance_group',
    comment => 'Resource consumer group/method for users' sessions who maintain
    the bug db');
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 who maintain the mail
    db');
dbms_resource_manager.create_plan_directive(plan => 'BUGDB_PLAN', group_or_subplan =>
'Bug_Online_group',
    comment => 'online bug users' sessions at level 0', 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 0', 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 1', cpu_p1 => 0, cpu_p2 => 100,
    parallel_degree_limit_p1 => 3);
dbms_resource_manager.create_plan_directive(plan => 'MAILDB_PLAN', group_or_subplan =>
```

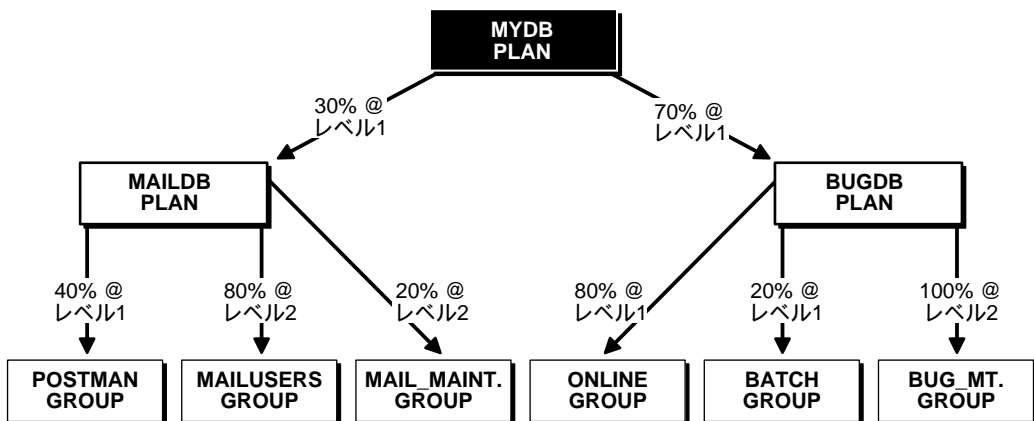
```

'Mail_Postman_group',
  comment => 'mail postman at level 0', 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 1', 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 1', cpu_p1 => 0, cpu_p2 => 20,
  parallel_degree_limit_p1 => 2);
dbms_resource_manager.create_plan_directive(plan => 'MYDB_PLAN', group_or_subplan =>
'MAILDB_PLAN',
  comment=> 'all mail users' sessions at level 0', cpu_p1 => 30);
dbms_resource_manager.create_plan_directive(plan => 'MYDB_PLAN', group_or_subplan =>
'BUGDB_PLAN',
  comment => 'all bug users' sessions at level 0', 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 で暗黙に実行されるので、直前の validate_pending_area のコールはオプションです。

図 11-1 複数レベルのスキーマ



ユーザーへのリソース・コンシューマ・グループの割当て

DATABASE_RESOURCE_MANAGER パッケージには、リソース・プランとリソース・コンシューマ・グループをメンテナンスするための前述のプロシージャだけでなく、リソース・

コンシューマ・グループをユーザーに割り当てるためのプロシージャも含まれています。次のプロシージャでは、ユーザーの初期コンシューマ・グループが設定されます。

```
set_initial_consumer_group(user in varchar2, consumer_group in varchar2)
```

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・グループにする前に、そのユーザーまたは PUBLIC にスイッチ特権を直接付与する必要があります。初期コンシューマ・グループのスイッチ特権は、そのユーザーに付与されたロールから取り込むことはできません（ALTER USER DEFAULT ROLE の場合と同じです）。

ユーザーの初期コンシューマ・グループを設定しなければ、そのユーザーの初期コンシューマ・グループは自動的にコンシューマ・グループ DEFAULT_CONSUMER_GROUP となります。DEFAULT_CONSUMER_GROUP には、PUBLIC に付与されたスイッチ特権があるので、ユーザー全員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。

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は、自分の初期コンシューマ・グループとして DEFAULT_CONSUMER_GROUP を持つこととなります。削除されたコンシューマ・グループに属するすべてのセッションは、DEFAULT_CONSUMER_GROUP に切り替えられます。

リソース・コンシューマ・グループの変更

次のプロシージャを使用すると、特定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ます。

```
switch_consumer_group_for_sess(session_id in number, session_serial  
in number, consumer_group in varchar2)
```

次のプロシージャを使用すると、特定のユーザー ID を持つすべてのセッションのリソース・コンシューマ・グループを変更できます。

```
switch_consumer_group_for_user(user in varchar2, class in varchar2)
```

さらに、どちらのプロシージャでも、最上位ユーザー・セッションに関連する任意の（PQ）スレーブ・セッションのリソース・コンシューマ・グループが変更されます。

関連項目：データベース・リソース・マネージャに対応付けられているビューの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

DBMS_RESOURCE_MANAGER_PRIVS パッケージ

DBMS_RESOURCE_MANAGER_PRIVS パッケージを使用して、リソース・コンシューマ・グループに対応付けられている権限をメンテナンスします。このパッケージ内のプロシージャは、コール側の権限で実行されます。

スイッチ特権の付与

コンシューマ・グループに切り替える権限を付与するには、次のプロシーダを使用します。

```
grant_switch_consumer_group(grantee_name in varchar2, consumer_group in varchar2,  
    grant_option in boolean)
```

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、そのロールが使用可能になっているユーザーは、自分の現行のコンシューマ・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 であれば、コンシューマ・グループのスイッチ特権が付与されているユーザーは、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

リソース・コンシューマ・グループに切り替える権限を取り消すには、次のプロシーダを使用します。

```
revoke_switch_consumer_group(revokee_name in varchar2, consumer_group in varchar2)
```

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後は、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しても失敗します。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 の一部になります。

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 PUBLIC のスイッチ特権を取り消した後は、これまで PUBLIC を介してのみコンシューマ・グループを使用していたユーザーは、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。

DBMS_SESSION パッケージを使用したユーザーのリソース・コンシューマ・グループの変更

DBMS_SESSION パッケージ内の次のプロシーダをコールすると、現行のリソース・コンシューマ・グループを変更できます。

```
switch_current_consumer_group(new_consumer_group in varchar2,  
    old_consumer_group out varchar2, initial_group_on_error in boolean)
```

このプロシージャにより、ユーザーはスイッチ特権を持っているコンシューマ・グループに切替えできるようになります。コール側が他のプロシージャの場合、このプロシージャにより、ユーザーはそ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・グループに切替えできるようになります。また、このプロシージャではユーザーに旧コンシューマ・グループが戻され、後からそのコンシューマ・グループに切り替えることができます。

`initial_group_on_error` パラメータでは、エラー発生時のプロシージャの動作を制御します。このパラメータが `TRUE` に設定されているときにエラーが発生すると、起動者のコンシューマ・グループがその初期コンシューマ・グループに設定されます。

データベース・リソース・マネージャのビュー

データベース・リソース・マネージャには、次の動的パフォーマンス表ビューが対応付けられています。

- V\$SESSION
- V\$MYSESSION
- V\$RSRC_CONSUMER_GROUP
- V\$RSRC_PLAN
- V\$RSRC_CONSUMER_GROUP_CPU_MTH
- V\$RSRC_PLAN_CPU_MTH
- V\$PARALLEL_DEGREE_LIMIT_MTH

データベース・リソース・マネージャには、次の静的データ・ディクショナリ・ビューが対応付けられています。

- DBA_RSRC_CONSUMER_GROUP_PRIVS
- DBA_RSRC_MANAGER_SYSTEM_PRIVS
- DBA_RSRC_CONSUMER_GROUPS
- DBA_RSRC_PLAN_DIRECTIVES
- DBA_RSRC_PLANS
- USER_RSRC_CONSUMER_GROUP_PRIVS
- USER_RSRC_MANAGER_SYSTEM_PRIVS
- DBA_USERS
- USERS_USERS

関連項目：これらの各ビューの内容の詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

スキーマ・オブジェクトを管理するための ガイドライン

この章では、スキーマ・オブジェクトを管理するためのガイドラインについて説明します。
この章のトピックは、次のとおりです。

- データ・ブロックの領域管理
- 記憶領域パラメータの設定
- 領域の割当て解除
- データ型の使用領域の理解

第 13 ~ 18 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。この項のトピックは、次のとおりです。

- [PCTFREE パラメータ](#)
- [PCTUSED パラメータ](#)
- [関連する PCTUSED と PCTFREE の値の選択](#)

PCTFREE パラメータと PCTUSED パラメータを使用して、次のような変更ができます。

- データの書き込みと検索のパフォーマンスを向上させます。
- データ・ブロック内の未使用領域を少なくします。
- データ・ブロック間の行連鎖を少なくします。

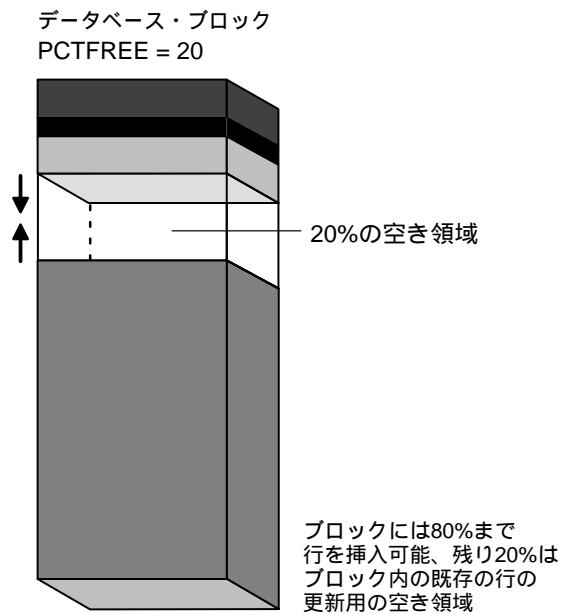
PCTFREE パラメータ

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 12-1](#) は PCTFREE を示しています。

図 12-1 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 の指定

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 の値を小さくすると、次のような影響が現れます。

- 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満杯になります。
- 表または索引の全データが格納されるブロックが少なくなる（ブロックあたり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 の値は小さい方が適しています。

PCTFREE の値を大きくすると、次のような影響が現れます。

- 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（ブロックあたり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ので、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 の値が大きい方が適しています。

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場合があります。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。

クラスタ化されていない表の PCTFREE クラスタ化されていない表の行のデータ・サイズが大きくなりそうな場合、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 クラスタ化されていない表についての説明は、クラスタ化された表にもあてはまります。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は、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 PCTFREE を指定できるのは、初めて索引を作成するときだけです。

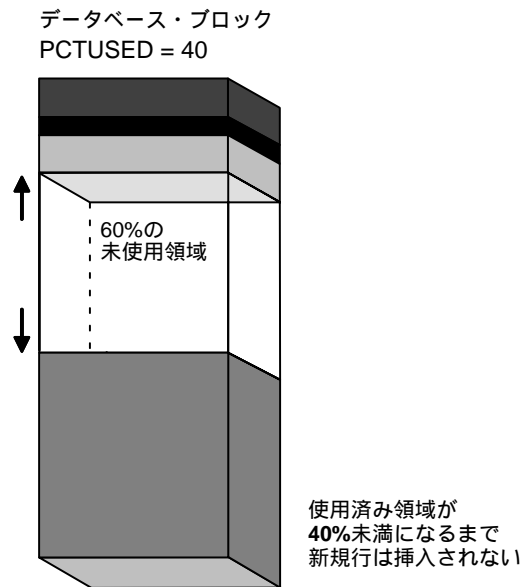
PCTUSED パラメータ

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 12-2 は、この状態を示しています。

図 12-2 PCTUSED



PCTUSED の指定

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。そのパーセント値は、合計領域からオーバーヘッドが差し引かれた後、データに使え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 の値を小さくすると、次のような影響が現れます。

- 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 文に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 の値を大きくすると、次のような影響が現れます。

- 領域効率が改善されます。
- INSERT と UPDATE での処理コストが増大します。

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 になる場合、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。
- ブロック・オーバーヘッドは、PCTUSED または PCTFREE の計算に含まれません。
- PCTUSED が 75 で PCTFREE が 20 の場合のように、PCTFREE と PCTUSED の和と 100 との差が小さいほど、パフォーマンス・コストで領域使用の効率がよくなります。

PCTFREE と PCTUSED の値を選択する例

次の例では、PCTFREE と PCTUSED の値を表に指定する方法とその理由を示します。

例 1	使用例：	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。
	設定値：	PCTFREE = 20 PCTUSED = 40
例 2	使用例：	多くのアクティビティに、処理の対象となる行のサイズを大きくしない INSERT 文や DELETE 文、および UPDATE 文が含まれています。
	設定値：	PCTFREE = 5 PCTUSED = 60
	説明：	ほとんどの UPDATE 文では行サイズは増加しないので、PCTFREE は 5 に設定されています。PCTUSED は、DELETE 文で解放される領域がまもなく使用され、処理は最小限度になるように 60 に設定されています。
例 3	使用例：	表がきわめて大きいいため記憶領域が主な問題になります。多くのアクティビティに、読み込み専用トランザクションが含まれています。
	設定値：	PCTFREE = 5 PCTUSED = 40
	説明：	この表は大きく、各ブロックを完全に満杯にするのが望ましいので、PCTFREE を 5 に設定します。

記憶領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領域パラメータについて説明します。
この項のトピックは、次のとおりです。

- [指定可能な記憶領域パラメータ](#)
- [INITTRANS と MAXTRANS の設定](#)
- [表領域内のセグメントのデフォルト記憶領域パラメータの設定](#)
- [データ・セグメントの記憶領域パラメータの設定](#)
- [索引セグメントの記憶領域パラメータの設定](#)
- [LOB セグメントの記憶領域パラメータの設定](#)
- [記憶領域パラメータの値の変更](#)
- [記憶領域パラメータの優先順位の理解](#)

記憶領域パラメータは、次のタイプの論理記憶構造に対して設定できます。

- 表領域（表領域内の任意のセグメントの大半のデフォルト）
- 表、クラスタ、スナップショット、スナップショット・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

指定可能な記憶領域パラメータ

すべてのデータベースには、記憶領域パラメータのデフォルト値があります。システムのデフォルトのかわりに、その表領域で作成されるオブジェクトだけに適用される表領域のデフォルトを指定できます。さらに、個々のオブジェクトに対して記憶領域設定を指定できます。設定できる記憶領域パラメータを次に示します。

INITIAL

セグメントが作成されるときに、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。

デフォルト: 5 データ・ブロック

最小値: 2 データ・ブロック（切上げ）

最大値: オペレーティング・システム固有

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になります。それ以降、NEXTは1つ前のNEXTのサイズに $(1 + \text{PCTINCREASE}/100)$ を掛けたサイズに設定されます。

デフォルト: 5 データ・ブロック
最小値: 1 データ・ブロック
最大値: オペレーティング・システム固有

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全体のエクステント数。

デフォルト: データ・ブロック・サイズとオペレーティング・システムによって異なります。
最小値: 1 (エクステント)
最大値: 無制限

MINEXTENTS

セグメントの作成時に割り当てられる全体のエクステント数。これによって、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。

デフォルト: 1 (エクステント)
最小値: 1 (エクステント)
最大値: 無制限

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掛けることによって、各増分エクステントが大きくなる割合 (百分率)。PCTINCREASE が 0 である場合、各増分エクステントのサイズは一定となります。ただし、PCTINCREASE が 0 (ゼロ) より大きいと、NEXT は計算されるたびに PCTINCREASE だけ大きくなります。PCTINCREASE が負の値になることはありません。

新しい NEXT は、 $1 + \text{PCTINCREASE}/100$ に、最後の増分エクステントのサイズ (古い NEXT) を掛けて、ブロック・サイズの次の倍数に切り上げられます。

デフォルト: 50 (%)
最小値: 0 (%)
最大値: オペレーティング・システム固有

INITRANS

データ・ブロック・ヘッダー内で領域を最初に確保すべき DML トランザクション・エントリ数を指定します。この領域は、対応するデータ・セグメントまたは索引セグメント内のすべてのデータ・ブロックのヘッダー内に確保されます。

デフォルト値は表については 1、クラスタと索引については 2 です。

MAXTRANS

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、この領域は効果よくブロック・ヘッダーの永久部分となります。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、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ができる空き領域を、MAXTRANS を使用して制限できます。

デフォルト値はオペレーティング・システム固有のブロック・サイズの間数であり、255 を超えません。

関連項目：記憶領域パラメータの詳細は、『Oracle8i SQL リファレンス』を参照してください。

デフォルト値にはオペレーティング・システムによって違うものであるため、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。

INITRANS と MAXTRANS の設定

表またはクラスタ、索引に割り当てられたデータ・ブロックに対するトランザクション・エントリの設定は、次の基準に基づいて、オブジェクトごとに個別に設定してください。

- データベース・データに確保する領域と比較した、トランザクション・エントリに確保する領域。
- 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のであれば、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、データベースで領域が特に貴重な場合には、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があります。この場合、INITRANS を大きく設定し（オブジェクトの使用中に必要となる、トランザクション・エントリ領域を割り当てるオーバーヘッドをなくすため）ユーザーが必要なデータ・ブロックをアクセスするのを待たないように MAXTRANS を大きく設定して、事前に割り当てるトランザクション・エントリ領域を検討してください。

表領域内のセグメントのデフォルト記憶領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領域パラメータを設定できます。表領域内のセグメントの作成時または作成後に変更するときに、明示的に記憶領域パラメータを設定しないと、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領域パラメータに自動的に設定されます。

パーティション表を使用して、表レベルでデフォルト記憶領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は、パーティション表からデフォルト記憶領域パラメータが継承されます（個々のパーティションに対して指定していない場合）。パーティション表の記憶領域パラメータが指定されていなければ、表領域から継承されます。

表領域レベルで MINEXTENTS を指定すると、表領域に割り当てられた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。基本的には、エクステント数はブロック数の倍数です。

データ・セグメントの記憶領域パラメータの設定

クラスタ化されていない表、スナップショットまたはスナップショット・ログのデータ・セグメントに対する記憶領域パラメータは、表またはスナップショット、スナップショット・ログの CREATE 文や ALTER 文の STORAGE 句を使用して設定できます。

対照的に、クラスタのデータ・セグメントに対する記憶領域パラメータは、CREATE CLUSTER コマンドまたは ALTER CLUSTER コマンドの STORAGE 句を使って設定します。クラスタ内に表やスナップショットを設定する個々の CREATE コマンドや ALTER コマンドは使用しません。クラスタ化された表やスナップショットの作成または変更時に指定された記憶領域パラメータは、無視されます。クラスタに設定された記憶領域パラメータは、表の記憶領域パラメータを上書きします。

索引セグメントの記憶領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領域パラメータは、CREATE INDEX コマンドまたは ALTER INDEX コマンドの STORAGE 句を使用して設定できます。主キー制約または一意キー制約を施行するために使用される索引に対して作成する索引セグメントの記憶領域パラメータは、CREATE TABLE コマンド、ALTER TABLE コマンドの ENABLE 句または ALTER INDEX コマンドの STORAGE 句に設定できます。

索引に PCTFREE を設定すると、索引の作成時にのみ効果があります。索引セグメントには PCTUSED を指定できません。

LOB セグメントの記憶領域パラメータの設定

CREATE TABLE 文の NOCACHE、NOLOGGING、PCTVERSION LOB の各記憶領域パラメータを使用して LOB セグメントに記憶領域パラメータを設定できます。

関連項目 : LOB 文の記憶領域パラメータの詳細リストは、『Oracle8i SQL リファレンス』を参照してください。

記憶領域パラメータの値の変更

現行の設定が不適切であると判断した場合、表領域に対するデフォルト記憶領域パラメータと、個々のセグメントに対する特定の記憶領域パラメータを変更できます。デフォルトの記憶領域パラメータはすべて表領域に再設定できます。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 記憶領域パラメータと MINEXTENTS 記憶領域パラメータは変更できません。セグメントの NEXT だけを変更すると、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 の両方を変更する場合、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。

記憶領域パラメータの優先順位の理解

ある時点で有効な記憶領域パラメータは、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。

1. ALTER TABLE/CLUSTER/SNAPSHOT/SNAPSHOT LOG/INDEX/ROLLBACK SEGMENT 文
2. CREATE TABLE/CLUSTER/SNAPSHOT/SNAPSHOT LOG/CREATE 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領域パラメータはどれも、表領域レベルで設定された対応するオプションに置き換わります。記憶領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ければ、表領域レベルでデフォルトが適用されます。記憶領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 システム・デフォルトが適用されます。記憶領域パラメータが変更されると、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だけ適用されます。

注意： 一時セグメントの記憶領域パラメータは、対応する表領域のために設定されたデフォルト記憶領域パラメータを常に使用します。

記憶領域パラメータの例

次の文が実行されていると想定します。

```
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 12-1 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	255 ブロックまたは 522240 バイト

NEXT または PCTINCREASE 記憶領域パラメータが ALTER 文（ALTER TABLE など）で変更されると、その指定された値が、データ・ディクショナリに格納されている現行値に置き換わります。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 記憶領域パラメータを修正します。

```
ALTER TABLE test_storage STORAGE (NEXT 500K);
```

その結果、割り当てるときに 3 番目のエクステントは 500KB、4 番目のエクステントは (500KB*1.5) = 750KB となり、それ以降は同じように計算されます。

領域の割当て解除

ここでは、未使用の領域の割当て解除について説明します。この項のトピックは、次のとおりです。

- [高水位標の表示](#)
- [領域割当て解除文の発行](#)

セグメントに領域を割り当てた後で、その領域が使用されていないとわかることがあります。たとえば、PCTINCREASE を高い値に設定し、それによって、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るかもしれません。また、ALTER TABLE...ALLOCATE EXTENT 文を発行して、領域を明示的に過大割当てする可能性もあります。未使用の領域または過大割当て領域があることがわかった場合は、その領域を解放して、その未使用領域を他のセグメントから使用できるようにすることができます。

高水位標の表示

割当てを解除する前に、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戻すプロシージャ (UNUSED_SPACE) が入っている DBMS_SPACE パッケージを使用できます。

セグメント内で、高水位標は使用された領域の量を示します。(割当てを解除する領域にデータがまったくない場合でも) 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。

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して変更されます。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して INITIAL の値が変更されます。

KEEP 句を指定しないと、初期エクステントのサイズと MINEXTENTS が保たれる限り、すべての未使用領域 (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、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。

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、『Oracle8i SQL リファレンス』を参照してください。

DBA_FREE_SPACE ビューを見ると、割当て解除によって解放された領域を確認できます。このビューの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

領域の割当ての解除例

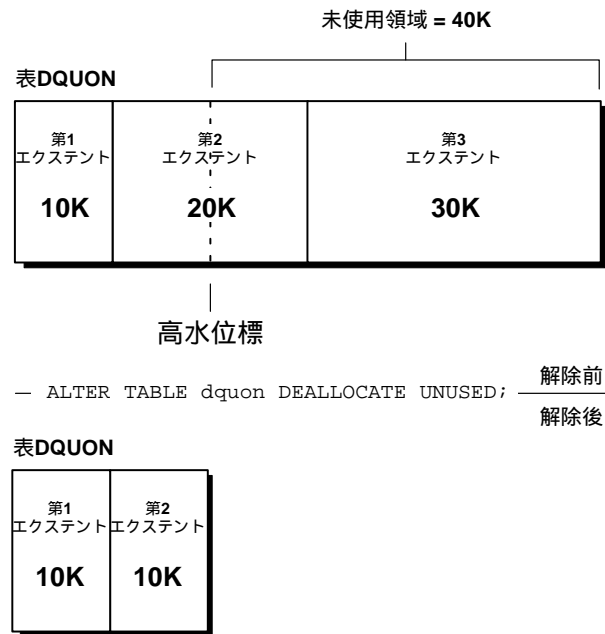
ここでは、領域の割当てを解除するさまざまな使用例を示します。あらかじめ、『Oracle8i リファレンス・マニュアル』の ALTER...DEALLOCATE UNUSED 文について理解しておく必要があります。

例 1

表 DQUON は、3 つのエクステントから構成されています（図 12-3 を参照してください）。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。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。次の文は、未使用領域の割当てをすべて解除し、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。

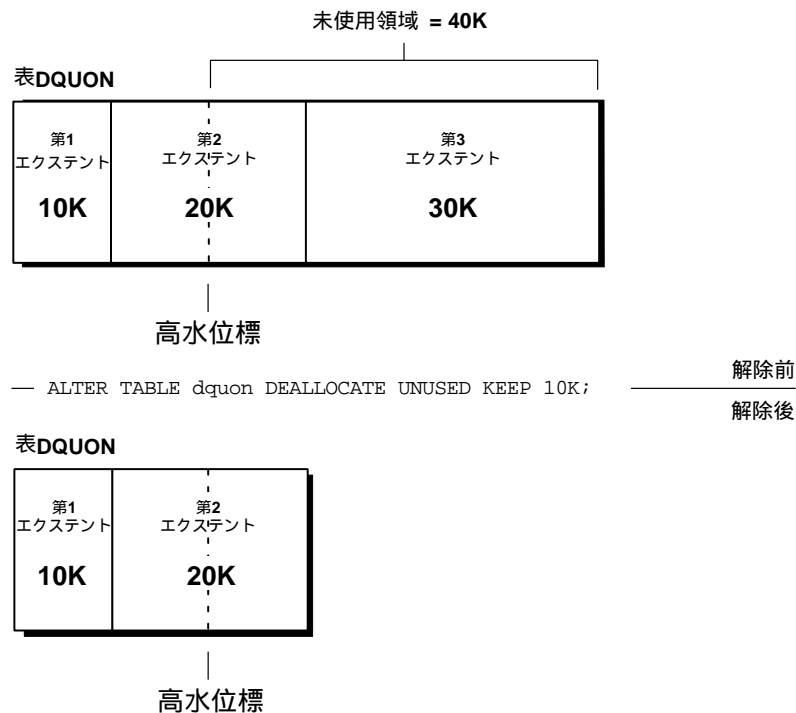
```
ALTER TABLE dquon DEALLOCATE UNUSED;
```

図 12-3 すべての未使用領域の割当てを解除する



DQUON のすべての未使用領域の割当てを解除して KEEP 10K を指定すると（図 12-4 を参照）、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 12-4 未使用領域の割当てを解除し、KEEP 10K を指定する



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;
```

例 2

ALTER TABLE DQUON DEALLOCATE UNUSED 文を発行すると、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。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、この例では 30KB に設定されるので注意してください。ただし、次のエクステントのサイズを明示的に設定できる場合は、ALTER...STORAGE [NEXT] 文を使用します。

例 3

MINEXTENTS に設定した数のエクステントを保持するため、DEALLOCATE では、そのセグメントにもともと割り当てられていたエクステントの割当てを解除する一方で、もともとインスタンスに割り当てられたエクステント（高水位標より下に追加されたもの）を保持できます。

たとえば、表 DQUON の MINEXTENTS の値が 2 だと仮定します。例 1 と例 2 では結果は同じです。ただし、MINEXTENTS の値が 3 の場合は、ALTER TABLE DQUON DEALLOCATE UNUSED 文は有効ではなく、ALTER TABLE DQUON DEALLOCATE UNUSED KEEP 10K 文では第 3 エクステントが削除され、MINEXTENTS の値は 2 に変更されます。

データ型の使用領域の理解

表またはその他のデータ構造を作る際には、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。次に説明するように、データ型ごとに異なる領域要件があります。

文字データ型 CHAR データ型と VARCHAR2 データ型では、Oracle を稼働させるハードウェアが使用するキャラクタ・セットに応じて、英数字データを ASCII（アメリカ合衆国標準情報交換用符号）または EBCDIC（拡張 2 進化 10 進コード）の文字列で格納します。また、Oracle の各国語サポート（NLS）機能によってサポートされるキャラクタ・セットを使用して、データも格納できます。

CHAR データ型は、固定長文字列を格納します。CHAR 列を含む表の作成時には、CHAR 列の列長を 1 ~ 255 の間で指定できます（単位は文字ではなくバイト）。デフォルトは 1 バイトです。列の値がその列長より小さい場合、列の残りの領域にはブランクが埋められます。

VARCHAR2 データ型は、可変長文字列を格納します。VARCHAR2 列を含む表の作成時には、VARCHAR2 列の最大列長を 1 ~ 4000 の間で指定できます（単位は文字ではなくバイト）。各行では、列の中のそれぞれの値が可変長フィールドとして格納されます。列の残りの領域には、ブランクが追加されません。

**NUMBER
データ型**

NUMBER データ型には、固定小数点数および浮動小数点数を格納します。つまり、 $1 \times 10^{-130} \sim 9.99...9 \times 10^{125}$ （最大有効桁数は 38）の範囲の正数、負数を格納します。

NUMBER データ型の列を定義するときには、オプションとして精度（桁数の合計）と位取り（小数点以下の桁数）も指定できます。Oracle では、38 桁以内の精度を持つ数値の移植性が保証されます。精度を指定しないで位取りのみを指定することもできます。

つまり、 $-1 \times 10^{-130} \sim -9.99...9 \times 10^{125}$ および 0（ゼロ）を指定できます（有効桁数 38 まで）。精度を指定しないと、列は値をそのとおりに格納します。また、位取りも精度も指定しなくてもかまいません。

```
column_name NUMBER (*, scale)
```

この場合、精度は 38 で、指定した位取りが保たれます。

**DATE
データ型**

DATE データ型は、日付や時刻など、ある時点を表す値を格納します。日付データは、それぞれ 7 バイトの固定長フィールドに格納されます。

**LONG
データ型**

LONG として定義される列は、2GB までの情報を持つ可変長文字データを格納します。LONG データは、テキスト・データであり、異なるキャラクタ・セットに移動するときには適切に変換されます。LONG データに索引を付けることはできません。

注意: LONG データ型のデータを、より制限の少ない LOB データ型のデータに変換できます。詳細は、『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』を参照してください。

**RAW および
LONG RAW
データ型**

RAW は VARCHAR2 文字データ型に似た可変長のデータ型です。ただし、RAW データと LONG RAW データの送信時には、Net8（ユーザー・セッションをインスタンスに接続する）およびインポート・ユーティリティとエクスポート・ユーティリティで文字が変換されない点が異なります。これとは対照的に、データベースのキャラクタ・セットとユーザー・セッションのキャラクタ・セットが違う場合、CHAR データおよび VARCHAR2 データ、LONG データに関し、この 2 つのキャラクタ・セットの間の変換が Net8 とエクスポート / インポートによって自動的に実行されます。

RAW データに索引を付けることはできますが、LONG RAW データに索引を付けることはできません。

ROWID 疑似列および ROWID データ型

Oracle データベースのクラスタ化されてない表のあらゆる行に、行の行断片（複数の行断片の間で連鎖される行の場合には最初の行断片）の物理アドレスに対応する、一意の ROWID が割り当てられます。

Oracle データベースの各表の内部には、ROWID という疑似列があります。この疑似列は、SELECT 文（SQL*Plus を使用している場合は DESCRIBE 文）を実行して、表の構造を記述したときは表示されませんが、SQL 問合せで予約語 ROWID を列名に使用して検索できます。

ROWID は選択された行ごとの物理アドレスを 2 進数を使用して表します。ROWID の VARCHAR2 を 16 進数で表す場合は、block.slot.file の 3 つの部分に分かれます。block はファイルのうちその行を含むデータ・ブロックであり、データ・ファイルに対する相対値です。row はそのブロック内の行です。file はその行を含むデータ・ファイルです。通常、行に割り当てられた ROWID は変更されません。インポート・ユーティリティとエクスポート・ユーティリティを使用して、行がインポートまたはエクスポートされたときには例外が発生します。行が表から削除されると（また、それを含むトランザクションがコミットされると）削除された行に対応付けられていた ROWID は、後続のトランザクションで挿入する行に割り当てることができます。

関連項目：NLS と各種キャラクタ・セットのサポートの詳細は、『Oracle8i NLS ガイド』を参照してください。

データ型の詳細は、『Oracle8i SQL リファレンス』を参照してください。

Oracle のデータ型のまとめ

表 12-2 に、Oracle の各データ型についての重要な情報をまとめます。

表 12-2 Oracle のデータ型についての情報のまとめ

データ型	説明	列長 (バイト)
CHAR(<i>size</i>)	長さが <i>size</i> バイトの固定長文字データ。	表の各行ごとに固定 (後続空白あり)。最大サイズは行あたり 2000 バイト、デフォルト・サイズは行あたり 1 バイト。 <i>size</i> を設定する前に、使うキャラクタ・セットの検討が必要です。(1 バイトと 2 バイトのどちらのキャラクタ・セットを使用するか)。
NCHAR (<i>size</i>)	長さが <i>size</i> 文字またはバイトの固定長文字データ (各国文字セットの選択によります)。	最大 <i>size</i> は、各文字の格納に必要なバイト数で判断されます。上限値は 2000 バイト。 <i>size</i> のデフォルトおよび最小サイズは、1 文字または 1 バイト (キャラクタ・セットによります)。
VARCHAR2 (<i>size</i>)	最大長が <i>size</i> バイトの可変長文字列。最大サイズを指定する必要があります。	最大 <i>size</i> は 4000 で、最小値は 1。VARCHAR2 の <i>size</i> は指定する必要があります。
NVARCHAR2 (<i>size</i>)	最大長 <i>size</i> 文字またはバイトの可変長文字列 (各国文字セットの選択によります)。	最大 <i>size</i> は、各文字の格納に必要なバイト数で判断されます。上限値は 4000 バイト。NVARCHAR2 には <i>size</i> を指定する必要があります。
NUMBER (<i>p</i> , <i>s</i>)	精度が <i>p</i> で位取りが <i>s</i> の数値。精度 <i>p</i> は 1 ~ 38、位取り <i>s</i> は -84 ~ 127。	行ごとに可変。列に必要な最大領域は、行あたり 21 バイト。
DATE	固定長の日付時間データ。紀元前 4712 年 1 月 1 日 ~ 西暦 4712 年 12 月 31 日の日付。デフォルトの書式は、DD-MON-YY。	表の各行に対して 7 バイトに固定。
LONG	可変長文字データ。	表の行ごとに可変、行あたり最大 $2^{31}-1$ バイト (2GB)。
RAW (<i>size</i>)	長さ <i>size</i> バイトのロー・バイナリ・データ。	表の行ごとに可変、行あたり最大 2000 バイト。RAW 値の <i>size</i> を指定する必要があります。
LONG RAW	可変長のローバイナリ・データ。	表の行ごとに可変、行あたり最大 2GB。

表 12-2 Oracle のデータ型についての情報のまとめ (続き)

データ型	説明	列長 (バイト)
ROWID	一意行アドレスを表す 16 進データ。このデータ型は、主として ROWID 疑似列から戻される値に使用されます。	表の行ごとに 6 バイトに固定。
UROWID [(size)]	索引構成表の 1 行の論理アドレスを表す 16 進文字列。	オプションの size は、データ型 UROWID の列のサイズ。最大サイズとデフォルトは 4000 バイト。
CHAR (size)	長さが size バイトの固定長文字データ。	最大 size は 2000 バイト。デフォルトおよび最小 size は 1 バイト。

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。この章のトピックは、次のとおりです。

- [パーティション表と索引](#)
- [パーティション化の方法](#)
- [パーティションの作成](#)
- [パーティションのメンテナンス](#)

パーティション表と索引

注意： パーティション表または索引の作成またはパーティションに関するメンテナンス操作を実行する前に、『Oracle8i 概要』のパーティションの情報を参照してください。

企業では、数百 GB におよびデータ（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、パーティション表と索引を作成し、それを使用することです。パーティション表または索引は、同じ論理属性を持つ多数の「サブパーティション」に分けられています。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）が同じ索引オプションを共有しています。各パーティション（またはサブパーティション）は別々のセグメントに格納され、異なる物理属性（PCTFREE、PCTUSED、INTRANS、MAXTRANS、TABLESPACE および STORAGE 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をそれぞれ別個の表領域に入れる必要はありませんが、それには利点があります。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くします。
- 各パーティションを個別にバックアップおよび回復します。
- 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します（I/O ロードのバランス調整に重要）。
- 管理作業を軽減し、可用性とパフォーマンスを改善します。

関連項目： パーティションの概念および利点の詳細は、『Oracle8i 概要』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 3 つの方法があります。

- レンジ・パーティション化
- ハッシュ・パーティション化
- コンポジット・パーティション化

この項では、この 3 つの使用方法について説明します。

レンジ・パーティション化方法の使用

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。レンジ・パーティション化は、表または索引のパーティション化仕様、および個々の各パーティションのパーティション化仕様によって定義されます。

次の例は、4つのパーティション（四半期の売上ごとに1つずつ）の表を示しています。この場合、SALE_YEAR=1998、SALE_MONTH=8 および SALE_DAY=18 の行のパーティション化キーは (1998, 8, 18) であり、第3パーティションに属し、表領域 TSC に格納されます。SALE_YEAR=1998、SALE_MONTH=8 および SALE_DAY=1 の行のパーティション化キーは (1998, 8, 1) であり、これも第3パーティションに属し、表領域 TSC 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 (1998, 04, 01)
  TABLESPACE tsa,
  PARTITION sales_q2 VALUES LESS THAN (1998, 07, 01)
  TABLESPACE tsb,
  PARTITION sales_q3 VALUES LESS THAN (1998, 10, 01)
  TABLESPACE tsc,
  PARTITION sales_q4 VALUES LESS THAN (1999, 01, 01)
  TABLESPACE tsd );
```

レンジ・パーティション化のメンテナンス

レンジ・パーティション化方法を使用して作成したパーティションに対して実行するメンテナンス操作は、パーティション化のマージのみです。ALTER TABLE...MERGE PARTITIONS コマンドを使用すると、2つの隣接するレンジ・パーティションの内容をマージして1パーティションにまとめることができます。履歴データをオンラインで大きなパーティションにしておく場合は、この操作を行います。たとえば、日付別のパーティションがあり、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することができます。

関連項目：レンジ・パーティション化の詳細は、『Oracle8i 概要』を参照してください。

CREATE TABLE...PARTITION 構文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

ハッシュ・パーティション化方法の使用

ハッシュ・パーティション化では、固定数のパーティション間でデータの物理位置が制御されます。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、データ位置をきめ細かく調整できます。

次の例は、パーティションのすべての記憶域属性を表レベルで指定する方法を示しています。

```
CREATE TABLE scubagear(  
    id NUMBER,  
    name VARCHAR2 (60))  
TABLESPACE ocean  
STORAGE (INITIAL 19k)  
    PARTITION BY HASH (id)  
    PARTITIONS 4;
```

ハッシュ・パーティションは、次の文が示すように特定の表領域に格納できます。

```
CREATE TABLE scubagear (...)  
    STORAGE (INITIAL 10k)  
    PARTITION BY HASH (id) PARTITIONS 16  
    STORE IN (hlto4, h4to8, h8to12, hl2to16);
```

また、各ハッシュ・パーティションに名前を付与して特定の表領域に格納することもできます。

```
CREATE TABLE product(...)  
    STORAGE (INITIAL 10k)  
    PARTITION BY HASH (id)  
    (PARTITION p1 TABLESPACE hl,  
     PARTITION p2 TABLESPACE h2);
```

ハッシュ・パーティション化索引用に、パーティション・レベルの表領域を指定することも可能です。

```
CREATE INDEX bcd_type ON scubagear(id) LOCAL  
PARTITIONS 4 STORE IN (ix1, ix2);  
  
CREATE INDEX bcd_type ON scubagear(id) LOCAL  
(PARTITION p1 TABLESPACE ix1, PARTITION p2 TABLESPACE ix2,  
 PARTITION p3 TABLESPACE ix3, PARTITION p4 TABLESPACE ix4);
```

ハッシュ・パーティションのメンテナンス

ハッシュ・パーティションの場合、次のものを除き、現行のレンジ・パーティションのメンテナンス操作がすべてサポートされます。

- ALTER TABLE...SPLIT PARTITION
- ALTER TABLE...DROP PARTITION
- ALTER TABLE...MERGE PARTITIONS

また、ハッシュ・パーティション化方法で作成したパーティション用に、特別に2つのメンテナンス操作があります。

- ハッシュ・パーティションを合わせる操作
- ハッシュ・パーティションを追加する操作

ハッシュ・パーティションを合わせる操作 単一のハッシュ・パーティションを削除してデータを他のパーティションへ再配分するには、次の文を使用します。

```
ALTER TABLE scubagear COALESCE PARTITION;
```

合わせるパーティションは、ハッシュ方式で判断されるので注意してください。また、ハッシュ・パーティションを合わせてデータを再配分する場合、ローカル索引はメンテナンスされません。ハッシュ・パーティションは、並行に合わせるすることができます。行を吸収したパーティションに対応する論理索引パーティションは、既存のパーティションから再構築する必要があります。

ハッシュ・パーティションを追加する操作 単一のハッシュ・パーティションを追加してデータを再配分するには、次の文を 1 つだけ使用します。

```
ALTER TABLE scubagear ADD PARTITION;  
ALTER TABLE scubagear  
ADD PARTITION p3 TABLESPACE t3;
```

ハッシュ・パーティションを追加する場合、ローカル索引はメンテナンスされません。また、ハッシュ・パーティションはパラレルで追加できます。

関連項目 : CREATE TABLE PARTITION...BY HASH および ALTER TABLE 文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

ハッシュ・パーティション化の詳細は、『Oracle8i 概要』を参照してください。

コンポジット・パーティション化方法の使用

コンポジット・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ハッシュ方式でサブパーティション化されます。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を管理しやすくするだけでなく、ハッシュ・パーティション化による並列性の長所も得られます。

コンポジット・パーティションの作成時には、次のことを指定します。

- パーティション化の方法（レンジ）
- パーティション化キー
- パーティション記述（パーティションバウンドを含む）
- 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

■ パーティション当り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

また、STORE IN 句を使用すると、各表パーティションのサブパーティションを分散させる表領域を指定できます。

次の文では、コンボジット・パーティション表が作成されます。

```
CREATE TABLE scubagear (equipno NUMBER, equipname VARCHAR(32), price NUMBER)
PARTITION BY RANGE (equipno) SUBPARTITION BY HASH (equipname)
SUBPARTITIONS 8 STORE IN (ts1, ts3, ts5, ts7)
(PARTITION p1 VALUES LESS THAN (1000),
PARTITION p2 VALUES LESS THAN (2000),
PARTITION p3 VALUES LESS THAN (MAXVALUE));
```

次の文は、サブパーティション名と、サブパーティションを格納する表領域の名前を指定できることを示しています。

```
CREATE TABLE scubagear (equipno NUMBER, equipname VARCHAR(32), price NUMBER)
PARTITION BY RANGE (equipno) SUBPARTITION BY HASH (equipname)
SUBPARTITIONS 8 STORE IN (ts1, 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));
```

コンボジット・パーティションのメンテナンス

コンボジット・パーティション表または索引には、レンジ・パーティションのすべてのメンテナンス操作を実行し、表パーティションのデフォルト属性を変更できます。

コンボジット・サブパーティションのメンテナンス

この項では、サブパーティションの特定のメンテナンス操作について説明します。内容は次のとおりです。

- [サブパーティションの変更](#)（表と索引）
- [サブパーティションの再構築](#)（索引のみ）
- [サブパーティションの改名](#)（表と索引）
- [サブパーティションの交換](#)（表のみ）
- [サブパーティションの追加](#)（表のみ）
- [サブパーティションを合わせる方法](#)（表のみ）
- [サブパーティションの移動](#)（表のみ）
- [サブパーティションの切捨て](#)（表のみ）

サブパーティションの変更 使用不可のマークが付いているパーティション表上で、ローカル索引のサブパーティションに次のようにマークを付けることができます。

```
ALTER INDEX scuba
  MODIFY SUBPARTITION bcd_types UNUSABLE;
```

また、MODIFY SUBPARTITION 句を使用して、表または索引のサブパーティションに領域を割り当てたり解除することもできます。

サブパーティションの再構築 サブパーティションを再構築し、索引サブパーティション内のデータを再生成できます。次の文では、表のローカル索引のサブパーティションが再構築されます。

```
ALTER INDEX scuba
  REBUILD SUBPARTITION bcd_types
  TABLESPACE tbs23 PARALLEL (DEGREE 2);
```

この例では、索引は異なる表領域に再構築されるので注意してください。

サブパーティションの改名 表または索引のサブパーティションに、新しい名前を割り当てることができます。次の文は、表のローカル索引のサブパーティションに新しい名前を割り当てる方法を示しています。

```
ALTER INDEX scuba RENAME SUBPARTITION bcd_types TO bcd_brands;
```

次の文は、単に、基礎となる表にパーティションが追加されたため、システムによって生成された名前を持つサブパーティションを改名する方法を示しています。

```
ALTER INDEX scuba RENAME SUBPARTITION sys_subp3254 TO bcd_types;
```

表のサブパーティションに新しい名前を割り当てることもできます。

```
ALTER TABLE diving RENAME SUBPARTITION locations_us
  TO us_monterey;
```

サブパーティションの交換 次の文は、表のサブパーティションを非パーティション表に変換する方法を示しています。

```
ALTER TABLE diving
  EXCHANGE SUBPARTITION locations_us
  WITH TABLE us_ca INCLUDING INDEXES;
```

サブパーティションの追加 次の文は、表のパーティションにサブパーティションを追加する方法を示しています。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。

```
ALTER TABLE diving MODIFY PARTITION locations_us
```

```
ADD SUBPARTITION us_monterey TABLESPACE us1;
```

サブパーティションを合わせる方法 次の文は、表のうち指定したパーティションのサブパーティション（RDBMS によって選択）の内容を、同じパーティションの他の 1 つ以上のサブパーティション（ハッシュ関数で決定）に分配し、選択したサブパーティションを破棄する方法を示しています。基本的に、この操作の結果は、ALTER TABLE MODIFY PARTITION ADD SUBPARTITION 文と逆になります。

```
ALTER TABLE diving MODIFY PARTITION us_locations  
COALESCE PARTITION;
```

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にあるデータを移動する方法を示しています。

```
ALTER TABLE scuba_gear MOVE SUBPARTITION bcd_types  
TABLESPACE tbs23 PARALLEL (DEGREE 2);
```

サブパーティションの切捨て 次の文は、表のサブパーティションにあるデータを切り捨てる方法を示しています。

```
ALTER TABLE diving  
TRUNCATE SUBPARTITION us_locations  
DROP STORAGE;
```

関連項目：この項で説明した文の構文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

パーティションの作成

パーティション表を作成する操作は、表や索引の作成時とほとんど同じです。PARTITION BY 句を指定して CREATE TABLE 文を使用する必要があります。また、各パーティションの表領域名を指定してください。

次の例は、パーティション 4 個を含む CREATE TABLE 文であり、四半期の売上ごとにパーティションを使用します。SALE_YEAR=1998、SALE_MONTH=7 および SALE_DAY=18 の行のパーティション化キーは (1998, 7, 18) であり、第 3 パーティション（表領域 TSC）になります。SALE_YEAR=1998、SALE_MONTH=7 および SALE_DAY=1 の行のパーティション化キーは (1998, 7, 1) であり、これも第 3 パーティションです。

```
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 ( 1998, 04, 01 )
  TABLESPACE tsa,
PARTITION sales_q2 VALUES LESS THAN ( 1998, 07, 01 )
  TABLESPACE tsb,
PARTITION sales_q3 VALUES LESS THAN ( 1998, 10, 01 )
  TABLESPACE tsc,
PARTITION sales_q4 VALUES LESS THAN ( 1999, 01, 01 )
  TABLESPACE tsd);
```

関連項目 : CREATE TABLE 文と PARTITION 句の詳細は、『Oracle8i SQL リファレンス』を参照してください。

パーティション化キー、パーティション名、バウンド、同一レベル・パーティション表および索引の詳細は、『Oracle8i 概要』を参照してください。

パーティションのメンテナンス

ここでは、パーティションの特定のメンテナンス操作について説明します。内容は次のとおりです。

- [パーティションの移動](#)
- [パーティションの追加](#)
- [パーティションの削除](#)
- [パーティションを合わせる方法](#)
- [パーティションのデフォルト属性の変更](#)
- [パーティションの切捨て](#)
- [パーティションの分割](#)
- [パーティションのマージ](#)
- [表パーティションの交換](#)
- [索引パーティションの再構築](#)
- [履歴表での時間枠の移動](#)
- [複数ステップのメンテナンス操作中のアプリケーション静止](#)

関連項目 : DDL 文の SQL 構文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

パーティション表と索引を記述しているカタログ・ビュー、およびパーティション表または索引のパーティションの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

インポート (Import)、エクスポート (Export) およびパーティションの詳細は、『Oracle8i ユーティリティ・ガイド』を参照してください。

パーティション化の一般情報は、『Oracle8i 概要』を参照してください。

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。

- データをクラスタ化しなおして断片化を少なくします。
- パーティションを別の表領域に移動します。
- 作成時間属性を修正します。

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領域属性を 1 ステップで変更できます。しかし、MODIFY PARTITION では変更できない物理属性もあります (TABLESPACE など)。そのような場合は、MOVE PARTITION 句を使用します。

表パーティションの移動

MOVE PARTITION 句を使用すると、パーティションを移動できます。たとえば、DBA が I/O のバランスを調整するため、最もアクティブなパーティションを自分のディスク上の表領域に移動させたいとします。その場合、DBA は次の文を発行します。

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

この文は、常にパーティションの旧セグメントを削除し、新しい表領域を指定しなくても新しいセグメントが作成されます。

移動するパーティションにデータが含まれているとき、MOVE PARTITION は、それぞれのローカルな索引の中のそれに一致するパーティションとグローバル索引のすべてのパーティションとを使用不可にします。MOVE PARTITION を発行した後で、それらの索引のパーティションを再構築する必要があります。

索引のパーティションの移動

MOVE PARTITION や DROP TABLE PARTITION など、一部の操作では、グローバル索引のすべてのパーティションに使用不可のマークが設定されます。ALTER INDEX REBUILD PARTITION 文を使って個々に各パーティションを再構築することによって、索引全体を再構築できます。そのような再構築は、同時実行できます。

また、単に索引を削除して作成しなおすという方法もあります。

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、パーティションをローカル索引に追加する方法を説明します。

表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を「高位」側の最後（既存の最後のパーティションの次の位置）に追加できます。パーティションを表の先頭または中央に追加する場合、あるいは最高位のパーティションのパーティション・バウンドが MAXVALUE である場合は、かわりに SPLIT PARTITION 文を使用する必要があります。

最高位のパーティションのパーティション・バウンドが MAXVALUE 以外のときは、ALTER TABLE...ADD PARTITION 文を使用してパーティションを追加できます。

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、DBA は 1 月用のパーティションを追加します。

```
ALTER TABLE sales
  ADD PARTITION jan96 VALUES LESS THAN ( '01-FEB-1999' )
  TABLESPACE tsx;
```

表にローカル索引が定義されていて、ALTER TABLE...ADD PARTITION 文を発行すると、一致するパーティションは各ローカル索引にも追加されます。新しい索引の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領域属性が割り当てられるため、ADD 操作が完了したら、それらを改名または変更するとよいでしょう。

索引のパーティションの追加

パーティションは、明示的にはローカル索引に追加できません。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である表に追加するときのみです。

最高位のパーティションのパーティション・バウンドが 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。

パーティションの削除

ここでは、ALTER TABLE DROP PARTITION 文を使用して表および索引のパーティションおよびそれらのデータを削除する方法を説明します。

表パーティションの削除

ALTER TABLE DROP PARTITION 文を使用して、表パーティションを削除できます。

この表にローカルな索引が定義されている場合は、ALTER TABLE DROP PARTITION により、各ローカル索引からも一致するパーティションが削除されます。

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。

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削除 ただし、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は、表パーティションの削除には次のどちらかの方法を使用してください。

1. グローバル索引は、ALTER TABLE...DROP PARTITION 文中の正しい位置にあるようにしておきます。この状況では、DROP PARTITION により、グローバル索引のすべてのパーティションに使用不可になるため、後でそれらを再構築してください。

注意： ALTER TABLE...DROP PARTITION 文は、すべてのグローバル索引パーティションを使用不可とマーク設定するだけでなく、すべての非パーティション索引を使用禁止状態にします。1 つの文でパーティション索引全体を再構築することはできません。パーティション索引を再構築する場合は、パーティション索引内のパーティションごとに別個の REBUILD 文を記述する必要があります。ここでは、`sal1` は非パーティション索引です。

```
ALTER TABLE sales DROP PARTITION dec94;  
ALTER INDEX sales_area_ix REBUILD sal1;
```

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。

2. ALTER TABLE...DROP PARTITION 文を発行する前に、DELETE コマンドを発行し、パーティションからすべての行を削除します。DELETE コマンド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 および UNDO ログが生成されます。

注意： すべてのパーティションの行を削除する前に、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。

たとえば、DBA がパーティションバウンド 10000 の最初のパーティションを削除するとします。この場合、DBA は次の文を発行します。

```
DELETE FROM sales WHERE TRANSID < 10000;  
ALTER TABLE sales DROP PARTITION dec94;
```

これは、小さい表の場合、または削除され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だけが含まれるような大規模な表の場合には最適な方法です。

データおよび参照の整合性制約を含む表の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、表の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。

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 cname_sales1;
ALTER TABLE sales DROP PARTITION dec94;
ALTER TABLE sales
  ENABLE CONSTRAINT cname_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。

2. ALTER TABLE...DROP PARTITION 文を発行する前に、DELETE コマンドを発行し、パーティションからすべての行を削除します。DELETE コマンドで参照整合性制約が施行され、さらにトリガーが起動され、REDO 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、または削除されるパーティションにその表の全データのごく一部分だけが含まれるような大規模な表の場合には最適な方法です。

索引の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。

グローバル索引のパーティションが空の場合は、ALTER INDEX...DROP PARTITION 文の発行によって明示的に削除できます。

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が使用不可となります。たとえば、DBA が索引のパーティション 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、DBA は、次の文を発行しなければなりません。

```
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。

パーティションを合わせる方法

ハッシュ方式でパーティション化した表のパーティション（RDBMS によって選択）の内容を、ハッシュ方式で判別された 1 つ以上のパーティションに分配し、選択したパーティションを破棄できます。

次の文では、最後のパーティションを合わせて、表のパーティション数を 1 だけ減らします。

```
ALTER TABLE ouu1
  COALESCE PARTITION;
```

パーティションのデフォルト属性の変更

コンボジット方式で作成された表のローカル索引のパーティション（または、コンボジット表のパーティション）のデフォルト属性を変更できます。

次の文では、パーティション表にあるローカル索引のパーティションの PCTFREE 属性（パーティション・レベルのデフォルト）を変更します。

```
ALTER INDEX scuba_1
  MODIFY DEFAULT ATTRIBUTES FOR PARTITION bcd_1998
  PCTFREE 25;
```

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削除するときは、ALTER TABLE...TRUNCATE PARTITION 文を使用します。索引パーティションの切捨てはできません。しかし、ALTER TABLE TRUNCATE PARTITION 文により、各ローカル索引の中の一致するパーティションは切捨てられます。

パーティション表の切捨て

ALTER TABLE...TRUNCATE PARTITION 文を使用して、空白を再生したり再生せずに、表のパーティションからすべての行を削除することができます。この表にローカル索引が定義されている場合は、ALTER TABLE...TRUNCATE PARTITION により、各ローカル索引からも一致するパーティションが切捨てられます。

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 ただし、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捨てるには次のいずれかの方法を使用します。

1. グローバル索引は、ALTER TABLE TRUNCATE PARTITION 文中の正しい位置におきます。

注意： ALTER TABLE...TRUNCATE PARTITION 文は、すべてのグローバル索引パーティションを使用不可とマーク設定するだけでなく、すべての非パーティション索引を使用禁止状態にします。1つの文でパーティション索引全体を再構築することはできません。パーティション索引を再構築する場合は、パーティション索引内のパーティションごとに別個の REBUILD 文を記述する必要があります。ここでは、sal1 は非パーティション索引です。

```
ALTER TABLE sales TRUNCATE PARTITION dec94;
ALTER INDEX sales_area_ix REBUILD sal1;
```

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。

2. ALTER TABLE...TRUNCATE PARTITION 文を発行する前に、DELETE コマンドを発行し、パーティションからすべての行を削除します。DELETE コマンド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 および UNDO ログが生成されます。

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だけが含まれるような大規模な表の場合には最適な方法です。

データおよび参照整合性制約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。

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、整合性制約を使用可能にします。

```
ALTER TABLE sales
  DISABLE CONSTRAINT dname_sales1;
ALTER TABLE sales TRUNCATE PARTITION 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。

2. ALTER TABLE...TRUNCATE PARTITION 文を発行する前に、DELETE コマンドを発行し、パーティションからすべての行を削除します。DELETE コマンドで参照整合性制約が適用され、さらにトリガーが起動され、REDO および UNDO ログが生成されます。

注意： すべてのパーティションの行を削除する前に、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だけが含まれるような大規模な表の場合には最適な方法です。

パーティションの分割

ALTER TABLE/INDEX のこの形式は、1 つのパーティションを 2 つのパーティションに分割します。パーティションが大規模になりすぎて、バックアップ操作、リカバリ操作またはメンテナンス操作に時間がかかる原因となっている場合には、SPLIT PARTITION 句を使用できます。また、SPLIT PARTITION 句を使用して I/O ロードを再分配することもできます。この句は、ハッシュ・パーティションには使用できないので注意してください。

表パーティションの分割

表のパーティションを分割するには、ALTER TABLE...SPLIT PARTITION 文を発行します。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、各ローカル索引の中の一致するパーティションも分割されます。新しい索引のパーティションには、Oracle によってシステム生成名とデフォルトの記憶領域属性が割り当てられるため、索引の分割後には、それらの索引のパーティションを改名または変更しても構いません。

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、ALTER TABLE...SPLIT PARTITION 文により、各ローカル索引の中の一致するパーティション (2 個) と、グローバル索引のすべてのパーティションが使用不可にされます。これらの索引のパーティションは、ALTER TABLE...SPLIT PARTITION 文を発行した後で、再構築する必要があります。

表パーティションの分割：使用例 このシナリオでは、"VET_cats" という表に "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。また、この表には VET というグローバル索引があります。VET には、VET_parta と VET_partb という 2 つのパーティションが含まれています。

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構築するために DBA が発行する文は、次のとおりです。

```
ALTER TABLE vet_cats SPLIT PARTITION  
    fee_katy at (100) INTO ( PARTITION
```

```
fee_katy1 ..., PARTITION fee_katy2 ...);  
ALTER INDEX JAF1 REBUILD PARTITION SYS_P00067;  
ALTER INDEX JAF1 REBUILD PARTITION SYS_P00068;  
ALTER INDEX VET REBUILD PARTITION VET_parta;  
ALTER INDEX VET REBUILD PARTITION VET_partb;
```

注意： 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調べるには、データ・ディクショナリを調べる必要があります。この使用例の場合は、SYS_P00067 と SYS_P00068 です。改名もできます。また、すでにパーティション fee_katy1 および fee_katy2 が JAF1 に含まれていないかぎり、この分割で生成されたローカル索引パーティションに割り当てられた名前は対応する実表パーティションの名前に合致します。

索引のパーティションの分割

ローカルな索引のパーティションは、明示的には分割できません。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。

次の文は、データを含むグローバル索引のパーティション QUON1 を分割します。

```
ALTER INDEX quon1 SPLIT  
    PARTITION canada AT VALUES LESS THAN ( 100 ) INTO  
    PARTITION canada1 ..., PARTITION canada2 ...);  
ALTER INDEX quon1 REBUILD PARTITION canada1;  
ALTER INDEX quon1 REBUILD PARTITION canada2;
```

再構築が必要になるのは、分割する索引パーティションが使用不可になっていた場合のみです。

パーティションのマージ

レンジ・パーティション表またはコンポジット・パーティション表のうち、2 つの隣接するパーティションの内容を 1 つにマージできます。1 つにマージされたパーティションは、マージ前のパーティションの 1 つ上位のパウンドを継承します。

次の文は、レンジ・パーティション表のうち、2 つの隣接するパーティションをマージします。

```
ALTER TABLE diving  
    MERGE PARTITIONS bcd1, bcd2 INTO PARTITION bcd1bcd2;
```

表パーティションの交換

表およびパーティションのデータ（および索引）セグメントを交換することによって、パーティションを非パーティション表に変換したり、表をパーティション表のパーティションに変換できます。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

ションがあって、そのような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便利です。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・ビューをパーティション表に変換する使用例

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）をパーティション表に変換する方法を説明します。パーティション・ビューは、次のように定義されます。

```
CREATE VIEW accounts
  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

1. 最初に、パーティション表を作成し、最後の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つのブロックのセグメントを（ブレースホルダとして）を取得します。

```
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 コマンドを使用して、表をそれに対応するパーティションに移行します。

```
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;
```

これで、NOV98 および DEC98 パーティションに対応付けられたブレースホルダ・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
  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。これらは、もともと NOV98 パーティションと DEC98 パーティションに付加されていたプレースホルダ・セグメントの所有者です。
5. UNIONALL ビュー内のすべての表がパーティションに変換された後に、ビューを削除しパーティション表をそのビューの名前に改名します。

```

DROP VIEW accounts;
RENAME accounts_new TO accounts;

```

関連項目：ここで説明したそれぞれの文の構文と使用方法の詳細は、『Oracle8i SQL リファレンス』を参照してください。

索引パーティションの再構築

ALTER TABLE...DROP PARTITION など、いくつかの操作では、グローバル索引のすべてのパーティションが使用不可になります。グローバル索引のパーティションを再構築するには、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構築します（再構築は同時実行できます）。
2. 一度索引を削除し、再作成します。

注意： この方法は表を一回走査するだけなので、この方法のほうが効率的です。

履歴表での時間枠の移動

履歴表は、ある期間にわたる企業の業務上の取引を記述するものです。履歴表は、売上、小切手、注文などの基礎情報を含むもので、実表になります。履歴表は、まとめ表、つまり、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たサマリー情報を取り込むものともなります。

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。DBA は、定期的に最も古いトランザクションを記述する一連の行を削除し、最近のトランザクションを記述する一連の行に領域を割り当てます。たとえば、1995 年 4 月 30 日の業務の締めで、DBA は 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。

履歴表で時間枠を移動する手順 ここで、特定の例を考えてみます。今月分の注文と 1 年分の履歴データを合わせて、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。月ごとに 1 つずつパーティションがあり、各パーティションにはそれが格納されている表領域と同様に ORDER_yymm という名前が付いています。

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、注文番号に関するローカルな一意（UNIQUE）の同一キー索引です。もう 1 つは ORDER_IX_SUPP で、これは業者番号に関するローカルな非同一次元キー索引です。ローカル索引のパーティションの名前には、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。1994 年 10 月 31 日には、次のようにして ORDER の時間枠を変更することになります。

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. 最近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. グローバル索引を削除し、再作成します。

```
ALTER INDEX ORDER DROP PARTITION ORDER_IX_CUST;  
ALTER INDEX REBUILD PARTITION ORDER_IX_CUST;
```

複数ステップのメンテナンス操作中のアプリケーション静止

通常、Oracle は、ALTER TABLE...DROP PARTITION など、個別の DDL 文を妨げる操作（DML、DDL、ユーティリティ）がないようにするために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ステップが必要な場合、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）が、進行中の複数ステップの操作を妨げないようにしてください。

たとえば、表 ORDER に参照整合性制約があり、パーティションを削除するために、それを使用禁止にしない場合は、かわりに前項からのステップ 2 を次のように置き換えます。

```
DELETE FROM ORDER WHERE ODATE < TO_DATE( '01-NOV-93' );  
ALTER TABLE ORDER DROP PARTITION ORDER_9310;
```

すべてのアプリケーションで使用されている APPLICATION ロールからアクセス権限を取り消すことによって、DELETE ステップと複数の DROP PARTITION ステップの間で新しい行を ORDER に挿入できないようにできます。毎晩または毎週末に、正しく定義されたバッチ枠の中で、ユーザー・レベルのアプリケーションをすべて終了させることもできます。

この章では、表を管理する方法について説明します。この章のトピックは、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](#)
- [表の記憶領域の手動割当て](#)
- [表の削除](#)
- [索引構成表](#)

この章で説明する作業を実行する前に、[第 12 章「スキーマ・オブジェクトを管理するためのガイドライン」](#)の内容をよく理解しておいてください。

表を管理するためのガイドライン

ここでは、表の管理のガイドラインについて説明します。この項のトピックは、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的指定](#)
- [トランザクション・エントリ・パラメータの指定](#)
- [各表の位置の指定](#)
- [表作成の並行化](#)
- [回復不能表作成に関する考慮事項](#)
- [表のサイズの見積りと記憶領域パラメータの設定](#)
- [大規模な表の計画](#)
- [表の制限](#)

表の管理をできる限り簡単にするために、ここに示すガイドラインに従ってください。

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、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、記憶領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。

表が次の要素を含むように、アプリケーション開発者とともに慎重に各表を作成してください。

- 表を正規化すること。
- 各列は適当なデータ型を持っていること。
- 記憶領域を節約するために、NULL を許す列が最後に定義されること。
- 記憶領域を節約し、SQL 文のパフォーマンスを最適化するために、適切であればいつでも表をクラスタ化すること。

データ・ブロック領域の使用方法的指定

各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・ブロック内の領域使用の効率、および現行データ更新時の予約域を調整できます。

関連項目：[PCTFREE と PCTUSED の指定方法の詳細は、12-2 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。](#)

トランザクション・エントリ・パラメータの指定

各表を作成するときに INITRANS パラメータと MAXTRANS 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・ブロック内のトランザクション・エントリに、最初に割り当てられる領域とその後割り当てられる領域を調整できます。

関連項目 : INITRANS と MAXTRANS の指定方法の詳細は、12-7 ページの「[記憶領域パラメータの設定](#)」を参照してください。

各表の位置の指定

適当な権限と表領域割当て制限があれば、現在、オンライン状態の表領域内に新しい表を作成できます。そのため、新しい表を格納する予定の表領域を識別するために、CREATE TABLE 文に TABLESPACE オプションを指定すべきです。

CREATE TABLE 文で表領域を指定しない場合、自分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、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。各表を作成するときに表領域を適切に指定することによって、次のようなことができます。

- データベース・システムのパフォーマンスを向上させます。
- データベース管理に必要な時間を短縮します。

次のように、スキーマ・オブジェクトの記憶領域の位置を誤ると、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可能性があります。
- 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、データベース管理者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。

関連項目 : 表領域の指定方法の詳細は、9-3 ページの「[ユーザーに表領域割当て制限を割り当てる方法](#)」を参照してください。

表作成の並行化

表の作成を CREATE TABLE コマンドの副問合せと並行化できます。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。

関連項目 : パラレル表作成の詳細は、『Oracle8i Parallel Server 概要および管理』を参照してください。

CREATE TABLE コマンドの詳細は、『Oracle8i SQL リファレンス』を参照してください。

回復不能表作成に関する考慮事項

表を回復不能として作成すると、表をアーカイブ済みログから回復できません（回復不能の表作成では、必要な REDO ログ・レコードが生成されないため）。したがって、表が失われると問題がある場合は、表を作成された後にバックアップする必要があります。一時的に使用するために作成する表など、注意が必要ない場合もあります。

CREATE TABLE...AS SELECT 文の副問合せを使用して表を作成するときに UNRECOVERABLE を指定すると、表を回復不能として作成できます。ただし、それ以降に挿入された行は回復できます。実際、その文の実行が完了した後は、それ以降に実行される文はすべて完全に回復できます。

表を回復不能として作成すると、次の利点があります。

- 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。
- 大規模な表の平行作成のパフォーマンスが向上します。

一般に、表を回復不能として作成したとき、小規模な表の場合より大規模な表の方が相対的にパフォーマンスの向上が大きくなります。小規模な表を回復不能として作成しても、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、特に表作成を並行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領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。

- 索引、ロールバック・セグメントおよび REDO ログ・ファイルの見積りと合わせた表の見積りサイズを使用して、作成するデータベースを保持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と、表が使用するディスク領域をよりよく管理できます。表を作成するとき、適切な記憶領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ことができる場合を想定します。表を作成するときに記憶領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、この表を呼び出すディスク 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もるかどうにかかわらず、クラスタ化されていない表を作成するときは記憶領域パラメータを明示的に設定できます。（クラスタ化した表はクラスタの記憶領域パラメータを自動的に使用します。）表を作成するとき、または表を変更するときに記憶領域パラメータを明示的に設定しなければ、その表が常駐する表領域に設定されたデフォルト記憶領域パラメータが自動的に使用されます。

表のデータ・セグメントのエクステントに記憶領域パラメータを明示的に設定する場合、エクステントを小さくしてその数を増やすよりも、エクステントを大きくしてその数を減らすように、表のデータを格納すべきであることに注意してください。

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簡単にし、無駄な領域や断片化を少なくして、領域の再使用率を向上させることができます。ただし、Oracle にはエクステントの数に制限はありませんが、表の中のエクステントの数が増えすぎると、その表を必要とする操作を実行するときのパフォーマンスに影響するので注意してください。

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。

データベース内にこのような表がある場合は、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。かなり大規模な表に対して索引を削除して作成しなおす必要がある場合（制約を使用可能、使用禁止にしたり、表を作成しなおすときなど）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 かなり大規模な表のデータを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であり、ユーザーがこの領域にアクセスできることを確認してください（なお、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

表を作成する前に、次の制限を理解しておいてください。

- 新しいオブジェクト型を含む表は Oracle8 以前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、『Oracle8i SQL リファレンス』を参照してください）。ユーザー定義のデータ型を含む表を作成すると、ユーザー定義型の列はそれを格納するリレーショナル列にマップされます。これらの「非表示」のリレーショナル列は、DESCRIBE 表の文では非表示であり、SELECT * 文では戻されません。したがって、Oracle は非表示列を作成してユーザー定義型のデータを格納するので、オブジェクト表や、REF 列、VARRAY、ネストした表またはオブジェクト型を持つ関係表の作成時には、その表に対して実際に作成される列の合計数が指定した値よりも多くなることがあります。次の公式によって、ユーザー定義型のデータを持つ表に対して作成される列の合計数が決定されます。

オブジェクト表の列数：

```
num_columns(object_table) =
    num_columns(object_identifier)
  + num_columns(row_type)
  + number of top-level object columns in the object type of table
  + num_columns(object_type)
```

関係表の列数：

```
num_columns(relational_table) =
    number of scalar columns in the table
  + number of object columns in the table
  + SUM [num_columns(object_type(i))]   i= 1 -> n
  + SUM [num_columns(nested_table(j))]   j= 1 -> m
  + SUM [num_columns(varray(k))]         k= 1 -> p
  + SUM [num_columns(REF(l))]            l= 1 -> q
```

where in the given relational table

object_type(i) is the ith object type column and

n is the total number of such object type columns

nested_table(j) is the jth nested_table column and

m is the total number of such nested table columns

varray(k) is the kth varray column and

p is the total number of such varray columns,

REF(l) is the lth REF column and

q is the total number of such REF columns.

```
num_columns(object identifier) = 1
num_columns(row_type)           = 1
num_columns(REF)                 = 1, if REF is unscoped
                                = 1, if the REF is scoped and the object identifier
                                is system generated and the REF has no
                                referential constraint
                                = 2, if the REF is scoped and the object identifier
                                is system generated and the REF has a
                                referential constraint
                                = 1 + number of columns in the primary key,
```

```

                                if the object identifier is primary key based
num_columns(nested_table)      = 2
num_columns(varray)            = 1
num_columns(object_type)       = number of scalar attributes in the object type
                                + SUM[num_columns(object_type(i))]      i= 1 -> n
                                + SUM[num_columns(nested_table(j))]      j= 1 -> m
                                + SUM[num_columns(varray(k))]            k= 1 -> p
                                + SUM[num_columns(REF(l))]                l= 1 -> q

```

特定のオブジェクト型の場合：

object_type(i) is an embedded object type attribute and
 n is the total number of such object type attributes,
 nested_table(j) is an embedded nested_table attribute and
 m is the total number of such nested table attributes,
 varray(k) is an embedded varray attribute and
 p is the total number of such varray attributes,
 REF(l) is an embedded REF attribute and
 q is the total number of such REF attributes.

例 1:

```

CREATE TYPE physical_address_type AS OBJECT
    (no CHAR(4), street CHAR(31), city CHAR(5), state CHAR(3));
CREATE TYPE phone_type AS VARRAY(5) OF CHAR(15);
CREATE TYPE electronic_address_type AS OBJECT
    (phones phone_type, fax CHAR(12), email CHAR(31));
CREATE TYPE contact_info_type AS OBJECT
    (physical_address physical_address_type,
     electronic_address electronic_address_type);
CREATE TYPE employee_type AS OBJECT
    (eno NUMBER, ename CHAR(60),
     contact_info contact_info_type);

CREATE TABLE employee_object_table OF employee_type;

```

従業員オブジェクト表の列数を計算するには、最初に employee_type に必要な列数を計算する必要があります。

```

num_columns(physical_address_type) =
    number of scalar attributes = 4
num_columns(phone_type) =
    num_columns(varray) = 1
num_columns(electronic_address_type) =
    number of scalar attributes
    + num_columns(phone_type)
    = 2 + 1 = 3
num_columns(contact_info_type) =
    num_columns(physical_address_type)
    + num_columns(electronic_address_type)

```

```
= 3 + 4 = 7
num_columns(employee_type) =
    number of scalar attributes
+ num_columns(contact_info_type)
= 2 + 7 = 9

num_columns (employee_object_table) =
    num_columns(object_identifier)
+ num_columns(row_type)
+ number of top level object columns in employee_type
+ num_columns(employee_type)
= 1 + 1 + 1 + 9 = 12
```

例 2:

```
CREATE TABLE employee_relational_table (einfo employee_type);

num_columns (employee_relational_table) =
    number of object columns in table
+ num_columns(employee_type)
= 1 + 9 = 10
```

例 3:

```
CREATE TYPE project_type AS OBJECT (pno NUMBER, pname CHAR(30), budget NUMBER);

CREATE TYPE project_set_type AS TABLE OF project_type;

CREATE TABLE department
    (dno NUMBER, dname CHAR(30),
    mgr REF employee_type REFERENCES employee_object_table,
    project_set project_set_type)
NESTED TABLE project_set STORE AS project_set_nt;

num_columns(department) =
    number of scalar columns
+ num_columns(mgr)
+ num_columns(project_set)
= 2 + 2 + 2 = 6
```

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。さらに、表の所有者は、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。たとえば、ユーザー SCOTT が次の文を発行すると、クラスタ化されていない表 EMP が SCOTT のスキーマに作成され、表領域 USERS に格納されます。

```
CREATE TABLE      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 dept_fkey REFERENCES dept)
PCTFREE 10
PCTUSED 40
TABLESPACE users
STORAGE ( INITIAL 50K
          NEXT 50K
          MAXEXTENTS 10
          PCTINCREASE 25 );
```

整合性制約が表のいくつかの列に定義され、いくつかの記憶設定が表に明示的に指定されていることに注目してください。

関連項目 : システム権限の詳細は、[第 24 章「ユーザー権限とロールの管理」](#)を参照してください。表領域割当て制限の詳細は、[第 23 章「ユーザーとリソースの管理」](#)を参照してください。

表の変更

表を変更するためには、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 システム権限のどちらかが必要です。

Oracle データベース内の表は、次の理由で変更できます。

- 表の 1 つ以上の列を追加または削除する場合。
- 表の整合性制約を追加または変更する場合。
- 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。
- データ・ブロック領域使用パラメータ（PCTFREE、PCTUSED）を修正する場合。
- トランザクション・エントリ設定（INITRANS、MAXTRANS）を変更する場合。
- 記憶領域パラメータ（NEXT、PCTINCREASE）を変更する場合。
- 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。
- 表に対応付けられた整合性制約を削除する場合。

既存の列の長さを拡張できます。ただし、表の中に行がない場合を除いて、列の長さは縮小できません。また、データ型 CHAR の列長を拡張するために表を修正している場合、特に表内の行数が多いのであれば、この操作は時間を浪費する可能性があり、さらに相当な追加記憶領域を必要とする可能性があることに注意してください。これは、各行の CHAR 値には空白を埋めて、新しい列長に合わせなければならないためです。

表のデータ・ブロック領域使用パラメータ（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るので注意してください。しかし、すでに割り当てられているブロックは、領域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。

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、MAXTRANS の新しい設定が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）に適用される一方、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るので注意してください。

記憶領域パラメータ INITIAL と MINEXTENTS は変更できません。他の記憶領域パラメータ（たとえば NEXT、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、前の値に基づいて決まるわけではありません。

表は SQL コマンド ALTER TABLE を使用して変更します。次の文は、EMP 表を変更します。

```
ALTER TABLE emp  
  PCTFREE 30  
  PCTUSED 60;
```

警告： 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。

新しい列を表に追加すると、列は NULL となります。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ときのみです。

ビューまたは PL/SQL プログラム・ユニットが実表に依存する場合、実表を変更すると、依存するオブジェクトに影響が及ぶ可能性があります。

関連項目： Oracle による依存性管理の詳細は、20-22 ページの「[オブジェクトの依存性の管理](#)」を参照してください。

表の記憶領域の手動割当て

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 Parallel Server を使用しているとき、表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。

新しいエクステントは、ALLOCATE EXTENT オプションを指定した SQL コマンド ALTER TABLE を使用して表に割り当てることができます。

関連項目： ALLOCATE EXTENT オプションの詳細は、『Oracle8i Parallel Server 概要および管理』を参照してください。

表の削除

表を削除するためには、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。

SQL コマンド DROP TABLE を使用して、必要ではなくなった表を削除できます。次の文は、EMP 表を削除します。

```
DROP TABLE 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 制約を削除するのであれば、次のように DROP TABLE コマンドに CASCADE オプションを指定してください。

```
DROP TABLE emp CASCADE CONSTRAINTS;
```

警告： 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せん。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。
 - 削除する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、無効になります（使用できません）。この種の依存性を管理する方法の詳細は、20-22 ページの「[オブジェクトの依存性の管理](#)」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るが、使用するとエラーが戻されます。
 - クラスタ化されてない表を削除する場合、それに割り当てられているエクステン트는表領域の空き領域にすべて戻され、新しいエクステンツ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使用されます。クラスタ化した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。
-

列の削除

Oracle では、表の行から列を削除して、領域を必要とする未使用の列をクリーン・アップし、索引と制約を再作成できます。データのエクスポート / インポートは不要です。

不要になった列を削除する方法と、将来システム・リソースに対する需要が小さくなった時点で削除するように、列にマークを付ける方法があります。

次の文では、未使用の列が表 t1 から削除されます。

```
ALTER TABLE t1 DROP UNUSED COLUMNS;
```

制限事項

列削除操作には、次の制限が適用されます。

- オブジェクト型の表からは列を削除できません。
- ネストした表からは列を削除できません。
- 表のすべての列を削除することはできません。

- パーティション化キー列は削除できません。
- SYS が所有する表からは列を削除できません。
- 親キー列は削除できません。

関連項目 : 表からの列削除に使用する構文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

索引構成表

ここでは、索引構成表の管理について説明します。この項のトピックは、次のとおりです。

- [索引構成表とは何か](#)
- [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](#)

索引構成表とは何か

索引構成表は、主キーに従ってグループ化されたデータ行を持つ表です。このクラスタ化は、B* ツリー索引を使用して行います。B* ツリー索引は、主キーと非キー列の両方を格納しているという点で、正規の表の B ツリー索引とは異なる特殊なタイプの索引ツリーです。索引構成表の属性は、索引の物理データ構造体内にすべて格納されています。

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呼び出す問合せに対して、表データへのより高速なキー・ベースのアクセスができます。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、別の表記憶領域がないために索引構造の更新のみが実行されます。

また、キー列が表と索引の両方で重複しないため、記憶領域要件も少なくなります。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しなければならないアプリケーションを使用しているときに有効です。また、索引構成表は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反転した索引が必要です。

関連項目 : 索引構成表の詳細は、『Oracle8i 概要』を参照してください。

索引構成表と標準的な表との違い

索引構成表は、1 つ以上の列に主キー索引を持つ標準的な表に似ています。ただし、索引構成表は表と B* ツリーの索引のための 2 つの別の記憶領域をメンテナンスせずに、表の主キーとその他の列の値を含む 1 つの B* ツリー索引だけをメンテナンスします。

図 14-1 標準的な表と索引構成表の構造

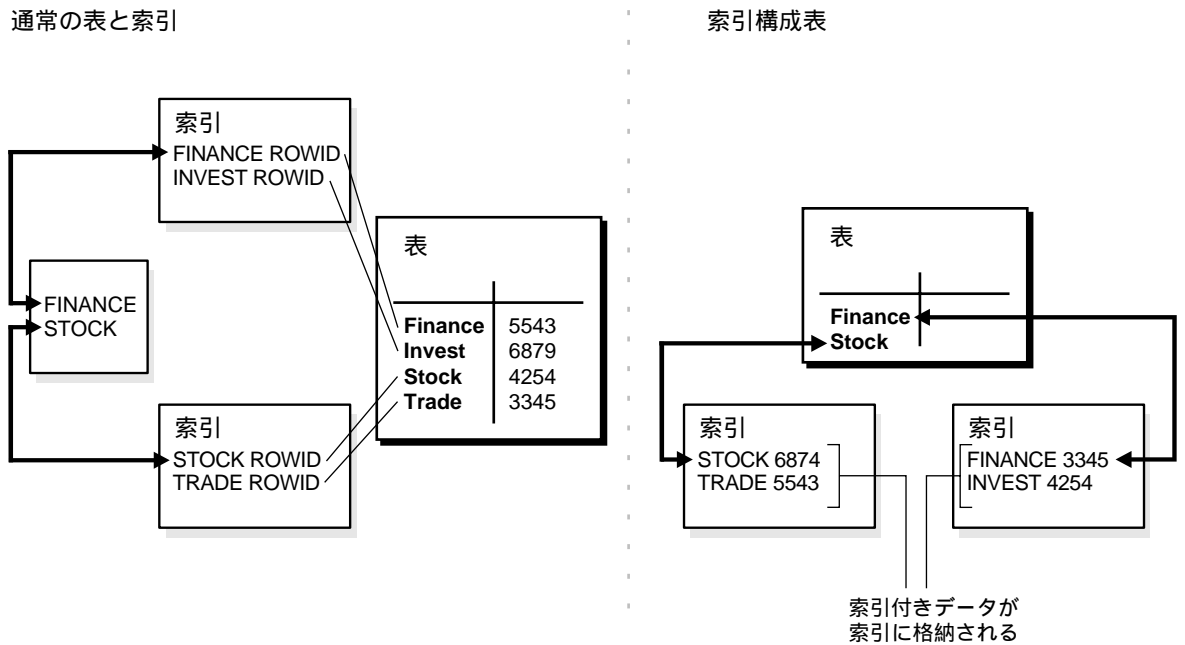


表 14-1 索引構成表と通常の表の比較 (続き)

通常の表	索引構成表
2 次索引には物理データが格納されます。	2 次索引には主キー・ベースの論理 ROWID が格納されます。
ハッシュ・クラスタまたは索引クラスタに格納できます。	ハッシュ・クラスタまたは索引クラスタには格納できません。

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 (索引構成表であることを示すもの)
- 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。
- オプションの行オーバーフロー仕様句は、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。

行オーバーフロー表領域は、ブロック・サイズの割合として定義されます。行のサイズが指定したしきい値 (PCTTHRESHOLD) より大きい場合、非キー列値はオーバーフロー表領域に格納されます。つまり、行は、列の境界で先頭の断片と後尾の断片などの 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。

- 次の例は、索引構成表を作成するときと与える情報を示したものです。

```
CREATE TABLE docindex(
    token char(20),
    doc_id NUMBER,
    token_frequency NUMBER,
    token_offsets VARCHAR2(512),
    CONSTRAINT pk_docindex PRIMARY KEY (token, doc_id))
ORGANIZATION INDEX TABLESPACE ind_tbs
PCTTHRESHOLD 20
OVERFLOW TABLESPACE ovf_tbs;
```

この例では、ORGANIZATION INDEX 修飾子で索引構成表を指定することを示しており、キー列と非キー列は表の主キー (token, doc_id) を指す列に対して定義される索引の中にあります。

索引構成表は、オブジェクト型を格納できます。たとえば、次のように、この例の目的であるオブジェクト型 mytype の列を含む索引構成表を作成できます。

```
CREATE TABLE iot (c1 NUMBER primary key, c2 mytype)
    ORGANIZATION INDEX;
```

ただし、オブジェクト型の索引構成表は作成できません。たとえば、次の文は無効です。

```
CREATE TABLE iot of mytype ORGANIZATION INDEX;
```

関連項目：CREATE INDEX 文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

AS 副問合せを使用する

AS 副問合せを使用して索引構成表を作成できます。この方法で作成した索引構成表は、PARALLEL オプションを使用してパラレルにロードできます。

次の文は、従来型の表 `rt` 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 iot(i primary key, j) ORGANIZATION INDEX PARALLEL (DEGREE 2)
    AS SELECT * FROM rt;
```

関連項目：索引構成表を作成する構文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

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、表領域 OVF_TBS に格納されるデータ・セグメントに入れられることを示しています。このキー列は、指定したしきい値に収まる必要があります。

非キー列を更新したために行のサイズが小さくなる場合、その更新が適用できる行断片（先頭または後尾）が識別され、その断片が書き直されます。

非キー列を更新したために行のサイズが大きくなる場合、その更新が適用できる行断片（先頭または後尾）が識別され、その断片が書き直されます。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、この断片が再度 2 つに分かれ、指定したしきい値以下に行のサイズを保つことに注意してください。

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ている列は、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個別性を判断できます。

関連項目：ANALYZE 文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

INCLUDING 句の使用 PCTTHRESHOLD を指定する他、INCLUDING <column_name> 句を使用してキー列でどの非キー列を格納するかを制御できます。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とします。INCLUDING 句で指定した列を超える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。

たとえば、索引構成表を作成した直前の例を変更して、その表の token_offsets 列値が常にオーバーフロー領域に格納できます。

```
CREATE TABLE docindex(
  token CHAR(20),
  doc_id NUMBER,
  token_frequency NUMBER,
  token_offsets VARCHAR2(512),
  CONSTRAINT pk_docindex PRIMARY KEY (token, doc_id))
ORGANIZATION INDEX TABLESPACE ind_tbs
PCTTHRESHOLD 20
INCLUDING token_frequency
OVERFLOW TABLESPACE ovf_tbs;
```

ここには、索引リーフ・ブロック内のキー列値で token_frequency までの非キー列だけ（この場合は 1 つの列だけ）を格納します。

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、キー列の接頭辞値の反復的な発生を排除できます。

キー圧縮によって、索引キーは接頭辞およびサフィクス・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サフィクス・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、索引ブロック当り格納できるキー数が増え、パフォーマンスが改善されます。

キー圧縮を使用可能にするには、次の操作に 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、キー列が接頭辞およびサフィクス・エントリに分割されるときの方法を識別する接頭辞の長さを（キー列数として）指定できます。

```
CREATE TABLE iot(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 iot(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上書きすることもできます。

```
CREATE TABLE iot(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 A MOVE NOCOMPRESS;
```

関連項目：キー圧縮の詳細は、『Oracle8i 概要』および『Oracle8i SQL リファレンス』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とは、物理的な構成のみが異なります。論理的には、索引構成表は同じように処理されます。INSERT 文および SELECT 文、DELETE 文、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 キーワードの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 キーの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、主キー索引セグメントの INITRANS を 4 に、データ・セグメントのオーバーフローの INITRANS を 6 に設定できます。

```
ALTER TABLE docindex INITRANS 4 OVERFLOW INITRANS 6;
```

また、PCTTHRESHOLD および INCLUDING 列の値も変更できます。新しい設定は、後続の操作中に行を先頭部分とオーバーフローの最後の部分に分けるために使用されます。たとえば、docindex 表の PCTTHRESHOLD および INCLUDING 列の値を次のように変更できます。

```
ALTER TABLE docindex PCTTHRESHOLD 15 INCLUDING doc_id;
```

INCLUDING 列を doc_id に設定すると、doc_id 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 句を使用してこの種のセグメントを追加できます。たとえば、docindex 表にオーバーフロー・セグメントがなければ、次のように追加できます。

```
ALTER TABLE docindex ADD OVERFLOW TABLESPACE ovf_tbs;
```

関連項目：ALTER TABLE 文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

索引構成表の移動（再構築）

索引構成表は主として B* ツリー索引に格納されるので、増分更新の結果として断片化を生じることがあります。ただし、ALTER TABLE...MOVE 文を使用すると、索引を再構築して、このような断片化を低減できます。

次の文は、索引構成表 docindex の INITTRANS を 10 に設定した後で再構築します。

```
ALTER TABLE docindex MOVE INITTRANS10;
```

オーバーフロー・データ・セグメントを持たない索引構成表は、ONLINE オプションを使用してオンラインで移動できます。たとえば、docindex 表にオーバーフロー・データ・セグメントがなければ、次のように移動をオンラインで実行できます。

```
ALTER TABLE docindex MOVE ONLINE INITTRANS 10;
```

次の文は、索引構成表 docindex をオーバーフロー・データ・セグメントとともに再構築します。

```
ALTER TABLE docindex MOVE TABLESPACE ix_tbs OVERFLOW TABLESPACE ov_tbs;
```

また、この最後の文では、索引構成表 iot は、C2 の LOB 索引とデータ・セグメントの再構築中に移動されます。

```
ALTER TABLE iot MOVE LOB (C2) STORE AS (TABLESPACE lob_ts);
```

関連項目：MOVE オプションの詳細は、『Oracle8i SQL リファレンス』を参照してください。

使用例：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ために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、論理上、dept_id=20 と e_id=10 の従業員行が削除され、dept_id=23 と e_id=10 の従業員行が挿入されることを表わしています。

```
UPDATE employees
SET dept_id=23
WHERE dept_id=20 and e_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。

```
ANALYZE TABLE docindex COMPUTE STATISTICS;
```

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できます。
- 主キー索引セグメントの物理統計を問い合わせるには、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 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 * FROM DBA_INDEXES WHERE INDEX_NAME= 'PK_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わせるには、USER_TABLES、ALL_TABLES または DBA_TABLES を使用します。IOT_TYPE = 'IOT_OVERFLOW' を検索すると、オーバーフロー・エントリを識別できます。たとえば、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 * FROM DBA_TABLES WHERE IOT_TYPE='IOT_OVERFLOW' and IOT_NAME= '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第 1 列だけを参照する場合、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。

たとえば、次の表を作成します。

```
CREATE TABLE employees (dept_id INTEGER, e_id INTEGER, e_name  
    VARCHAR2, PRIMARY KEY (dept_id, e_id)) ORGANIZATION INDEX;
```

データはすでに主キーについて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 employees ORDER BY (dept_id, e_id);  
SELECT * FROM employees ORDER BY (dept_id);
```

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、さらにソートする必要があります (他の 2 次索引が定義されていない場合)。

```
SELECT * FROM employees ORDER BY (e_id);  
SELECT * FROM employees ORDER BY (e_name);
```

索引構成表の標準的な表への変換

索引構成表から標準的な表に変換するには、Oracle の IMPORT/EXPORT ユーティリティまたは 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

- 索引構成表のデータを従来型パスを使用して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。
- 索引構成表のデータを、IGNORE=y (オブジェクト存在エラーを無視する)を確認してインポートします。

注意： 索引構成表を標準的な表に変換する前に、Oracle8 以前の Export ユーティリティで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： IMPORT/EXPORT の使用方法の詳細は、『Oracle8i ユーティリティ・ガイド』を参照してください。

ビュー、順序およびシノニムの管理

この章では、ビューの管理について説明します。この章のトピックは、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](#)

この章で説明する作業を実行する前に、[第 12 章の「スキーマ・オブジェクトを管理するためのガイドライン」](#)の内容をよく理解しておいてください。

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の専用表示であり、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、「格納された問合せ」と考えることも、「仮想表」と考えることもできます。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を説明します。この項のトピックは、次のとおりです。

- [ビューの作成](#)
- [結合ビューの変更](#)
- [ビューの置換](#)
- [ビューの削除](#)

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。
- ビューの所有者は、ビュー定義で参照するすべてのオブジェクトにアクセスするための権限を明示的に保持する必要があります。所有者は、ロールによってはそれらの権限を取得できません。また、ビューの機能はビューの所有者の権限に依存します。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、基盤となるオブジェクトの 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。

ビューは、SQL コマンド CREATE VIEW を使用して作成します。ビューはそれぞれ、表またはスナップショット、その他のビューを参照する問合せによって定義されます。ビューを定義する問合せには、FOR UPDATE 句を指定できません。たとえば、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 10 の行のみを参照します。また、CHECK OPTION は、そのビューが選択することが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を使用して、ビューを作成します。たとえば、次の INSERT 文では、SALES_STAFF ビューによって EMP 表に行が正常に挿入されます。それらの行はすべて部門番号が 10 の行です。

```
INSERT INTO sales_staff VALUES (7584, 'OSTER', 10);
```

ただし、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するため、その文はロールバックされ、エラーが戻されます。

```
INSERT INTO sales_staff VALUES (7591, 'WILLIAMS', 30);
```

次の文は、EMP 表と DEPT 表のデータを結合する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;
```

DIVISION1_STAFF ビューは、EMP 表と DEPT 表の情報を結合するものです。このビューの CREATE VIEW 文には CHECK OPTION が指定されていません。

ビュー作成時の問合せ定義の展開

ANSI/ISO 規格に従って、ビューが作成されるときに、Oracle はトップレベル・ビュー問合せのワイルドカードを列リストに展開し、結果の問合せをデータ・ディクショナリに格納します。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、基盤となる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すために引用符で囲まれています。

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、Oracle は定義の問合せのワイルドカードを展開します。

エラー付ビューの作成

CREATE VIEW 文に構文エラーがない場合、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。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にも、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 コマンドの 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、ビューはエラー付きで作成されません。ビューがエラー付きで作成されると、そのことを示すメッセージが戻されます。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を実行できるようになると、ビューは再コンパイルされ、有効（使用可能）になります。

関連項目：条件の変更とそれがビューに及ぼす影響の詳細は、20-22 ページの「[オブジェクトの依存性の管理](#)」を参照してください。

結合ビューの変更

変更可能な結合ビューとは、SELECT 文の上位の FROM 句に複数の表を含んでいて、かつ次のいずれも含まないビューのことです。

- DISTINCT 演算子
- 集約関数：AVG、COUNT、GLB、MAX、MIN、STDDEV、SUM または VARIANCE
- 集合演算：UNION、UNION、ALL、INTERSECT、MINUS
- GROUP BY 句または HAVING 句
- START WITH 句または CONNECT BY 句
- ROWNUM 疑似列

制限がいくつかありますが、結合を伴うビューを変更できます。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が上位のビューにマージできなければなりません。

次の項の例では、EMP 表と DEPT 表を使用します。これらの例は、この 2 つの表に主キーと外部キーを明示的に定義するか、一意の索引を定義する場合にのみ当てはまります。次に示すのは、制約に適切に従った EMP と DEPT の表定義です。

```
CREATE TABLE dept (  
    deptno      NUMBER(4) PRIMARY KEY,  
    dname       VARCHAR2(14),  
    loc         VARCHAR2(13));
```

```
CREATE TABLE emp (  
    empno       NUMBER(4) PRIMARY KEY,  
    ename       VARCHAR2(14),  
    job         VARCHAR2(14),  
    mgr         NUMBER(4) REFERENCES emp (empno),  
    sal         NUMBER(8,2),  
    comm        NUMBER(8,2);
```

```

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、前述の主キーと外部キーの制約を省略し、DEPT (DEPTNO) に UNIQUE INDEX を作成して、次の例を適用できます。

関連項目：マージ可能なビューの詳細は、『Oracle8i チューニング』を参照してください。

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なものです。表のすべてのキーが結合の結果のキーでもある場合、その表はキー保存です。つまり、キー保存表は、結合後もそのキーを保存します。

注意： 表の 1 つまたは複数のキーをキー保存として選択する必要はありません。1 つまたは複数のキーが選択され、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あって、表内のデータの特性ではありません。たとえば、EMP 表で各部門に最高でも 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 DEPT.DEPTNO は一意ですが、DEPT はキー保存表ではありません。

EMP_DEPT_VIEW から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 は表 EMP 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からです。

DML 文および結合ビュー

結合ビューにおいて、どのような UPDATE、INSERT または DELETE 文でも、基礎を形成する実表 1 つのみを変更できます。

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';
```

この文は基礎を形成する DEPT 表を変更しようとしませんが、DEPT 表は EMP_DEPT ビューに保存されたキーではないため、ORA-01779 エラー「複数表にマップする列を変更できません」が発生し、文を実行できません。

一般に、結合ビューの変更可能な列はすべて、キー保存表の列にマップしなければなりません。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変更できません。

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 ビューを対象にしたものです。

```
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、この文は妥当です。

次のビューでは 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、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内部結合を伴うので、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 文は実行されます。

```
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 (したがって、EMP 表に対する FOREIGN KEY 整合性制約を満たすので) この文は実行されます。

次の INSERT 文は、実 EMP 表に対する UPDATE が失敗するのと同じ理由で失敗します。つまり、EMP 表の FOREIGN KEY 整合性制約に違反しています。

```
INSERT INTO emp_dept (ename, empno, deptno)
  VALUES ('KURODA', 9010, 77);
```

次の INSERT 文は ORA-01776 エラー「結合ビューを介して複数の実表を変更できません」によって失敗します。

```
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

表 15-1 で説明するビューは、結合ビューを変更する際に使用すると便利です。

表 15-1 UPDATABLE_COLUMNS ビュー

ビュー名	説明
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列をすべて表示します。
DBA_UPDATABLE_COLUMNS	DBA スキーマ内で変更可能なすべての表とビューの列をすべて表示します。
ALL_UPDATABLE_VIEWS	変更可能なすべての表とビューの列をすべて表示します。

ビューの置換

ビューを置き換えるには、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必要がある場合は、そのビューを置き換える必要があります。これは、ビューの定義を直接変更することが出来ないことを意味します。次の方法でビューを置き換えることができます。

- ビューを削除してから再作成します。

警告： 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。ビューを作成しなおしてから、権限を再付与してください。

- 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き換え、現行のセキュリティ認可を保存します。たとえば、先の例で示されたように、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する場合を想定します。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必要があります。次の文によって、SALES_STAFF ビューの現行バージョンを置き換えることができます。

```
CREATE OR REPLACE VIEW sales_staff AS
  SELECT empno, ename, deptno
  FROM emp
  WHERE deptno = 30
  WITH CHECK OPTION CONSTRAINT sales_staff_cnst;
```

ビューを置き換える前に、次の影響を検討してください。

- ビューを置き換えることによって、データ・ディクショナリ内のビュー定義が置き換えられます。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。

- 以前 CHECK OPTION に制約が定義されていたが、新しいビュー定義には指定されない場合、その制約は削除されます。
- 置き換えら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（使用不可能）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、20-22 ページの「[オブジェクトの依存性の管理](#)」を参照してください。

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のビューを削除するには、DROP ANY VIEW システム権限が必要です。ビューは SQL コマンド DROP VIEW を使用して削除します。たとえば、次の文は SALES_STAFF ビューを削除します。

```
DROP VIEW sales_staff;
```

順序の管理

ここでは、順序を管理する方法について説明します。この項のトピックは、次のとおりです。

- [順序の作成](#)
- [順序の変更](#)
- [順序に影響を及ぼす初期化パラメータ](#)
- [順序の削除](#)

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に順序を作成するには、CREATE ANY SEQUENCE 権限が必要です。順序は、SQL コマンド CREATE SEQUENCE を使用して作成します。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をより速く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、順序番号がスキップされる可能性があります。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、使用されたにもかかわらず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、『Oracle8i ユーティリティ・ガイド』を参照してください。

関連項目：Oracle Parallel Server がキャッシュされた順序番号に与える影響の詳細は、『Oracle8i Parallel Server 概要および管理』を参照してください。

順序番号をキャッシュに格納する際のパフォーマンスの詳細は、『Oracle8i チューニング』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 システム権限が必要です。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。順序の開始点を変更するには、順序を削除してから、作成しなおしてください。順序番号で DDL を実行すると、キャッシュ値が削除されます。

順序は SQL コマンド ALTER SEQUENCE を使用して変更します。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;
```

順序に影響を及ぼす初期化パラメータ

初期化パラメータ SEQUENCE_CACHE_ENTRIES は、キャッシュされる順序の数を設定します。監査が使用可能になっている場合は、監査セッション番号を識別するために、追加の順序を 1 つ用意できます。

SEQUENCE_CACHE_ENTRIES の値が小さすぎると、次の例のように Oracle が順序値をスキップすることがあります。キャッシュされた 5 つの順序を使用しており、キャッシュが満杯で、SEQUENCE_CACHE_ENTRIES = 4 になっている場合を考えます。現在キャッシュに 4 つの順序が入っている場合、5 番目の順序によってキャッシュ内で最後に使用された順序が置き換えられ、追い出された順序の中に残っていた値はすべて（最後にキャッシュに格納された順序番号まで）失われます。

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。必要なくなった順序は、SQL コマンド

DROP SEQUENCE 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;
```

順序が削除されると、その定義がデータ・ディクショナリから削除されます。シノニムはそのまま残りますが、参照時にエラーが戻されます。

シノニムの管理

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

シノニムの作成

自分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 SYNONYM 権限が必要です。また、別のユーザーのスキーマにプライベート・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。

シノニムは、SQL コマンド CREATE SYNONYM を使用して作成します。たとえば、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のプライベート・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。またパブリック・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。

SQL コマンド DROP SYNONYM を使用して、必要なくなったシノニムを削除できます。プライベート・シノニムを削除するには、PUBLIC キーワードを指定しないでください。またパブリック・シノニムを削除するには、PUBLIC キーワードを指定してください。

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;
```

シノニムが削除されると、その定義がデータ・ディクショナリから削除されます。削除されたシノニムを参照するオブジェクトはすべてそのまま残ります。ただしそれらは無効（使用不可能）になります。

関連項目：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、20-22 ページの「[オブジェクトの依存性の管理](#)」を参照してください。

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。この章のトピックは、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](#)
- [索引の領域使用を監視](#)
- [索引の削除](#)

この章で説明する作業を実行する前に、[第 12 章「スキーマ・オブジェクトを管理するためのガイドライン」](#)の内容をよく理解しておいてください。

索引を管理するためのガイドライン

ここでは、索引の管理のガイドラインについて説明します。この項のトピックは、次のとおりです。

- 表データ挿入後の索引の作成
- 表あたりの索引の数の制限
- 各索引の表領域の指定
- トランザクション・エントリ・パラメータの指定
- 索引ブロックの領域使用の指定
- 索引作成の平行化
- NOLOGGING を指定した索引作成に関する考慮事項
- 索引サイズの見積りと記憶領域パラメータの設定

索引とは、表とクラスタに対応付けられたオプションの構造体であり、明示的に作成することによって、表に対する SQL 文の実行速度を上げることができます。このマニュアルに索引が付いていることによって情報を速く検索できるのと同じように、Oracle の索引は表データに対する高速アクセス経路を提供します。

索引の有無によって SQL 文を変更する必要はありません。索引はデータに対する高速アクセス経路を提供するだけであり、実行の速度にだけ影響します。データ値に索引が付いている場合、その索引によって、その値を含む行の位置が直接指示されます。

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、実表またはその他の索引に影響を与えることなく、いつでも作成または削除できます。索引を削除してもアプリケーションはすべて実行を続けますが、それまで索引が付いていたデータのアクセスは遅くなります。索引は、独立した構造体であり、記憶領域を必要とします。

索引の作成後は、Oracle は自動的に索引をメンテナンスし、使用します。新しい行の追加、行の更新または行の削除といったデータに対する変更は、関連するすべての索引に自動的に反映され、ユーザーは何もする必要はありません。

関連項目：索引の作成のパフォーマンスの意味に関する詳細は、『Oracle8i チューニング』を参照してください。

索引の詳細は、『Oracle8i 概要』を参照してください。

Oracle Objects オプションがある場合は、ドメイン固有の演算子と索引作成スキーマを定義し、それをサーバーに統合できます。詳細は、『Oracle8i 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

表の索引は、(SQL*Loader または Import ユーティリティを使用して) データを表に挿入またはロードした後で作成します。索引のない表にデータ行を挿入してから、それ以降のアクセスのために索引を作成するほうが効率的です。表データをロードする前に索引を作成すると、表に行が挿入されるたびに索引をすべて更新しなければなりません。また、クラスタの索引は、そのクラスタにデータを挿入する前に作成してください。

すでにデータを持っている表に索引を作成するには、Oracle はソート領域を使用しなければなりません。Oracle は索引の作成者に割り当てられたメモリー内のソート領域（初期化パラメータ SORT_AREA_SIZE で決まるユーザーあたりの容量）を使用しますが、索引作成のために割り当てられた一時セグメントとソート情報もスワップしなければなりません。

索引が極端に大きい場合は、次の手順を実行すると便利です。

大きな索引を管理する手順

1. 一時セグメント表領域を新しく作成します。
2. 索引作成者の一時セグメント表領域を変更します。
3. 索引を作成します。
4. 一時セグメント表領域を削除し、必要であれば、作成者の一時セグメント表領域を再指定します。

関連項目：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できるので、データをロードしながら索引を作成できます。詳細は、『Oracle8i ユーティリティ・ガイド』を参照してください。

表あたりの索引の数の制限

表は、かなり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、表を修正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。

したがって、表からデータを検索する速度とその表を更新する速度の間に妥協点があります。たとえば、表が主に読取り専用であれば、索引を増やすと有効ですが、表がかなり頻繁に更新されるのであれば、索引を少なくするほうが望ましいでしょう。

トランザクション・エントリ・パラメータの指定

各索引を作成するときに INITRANS パラメータと MAXTRANS パラメータを指定することによって、索引のセグメントのデータ・ブロック内のトランザクション・エントリに、最初に割り当てられる領域とその後割り当てられる領域を調整できます。また、更新の余地を残しておき、これらの長期的な（索引の使用期間中など）設定値は後から識別する必要があります。

関連項目：これらのパラメータ設定の詳細は、12-7 ページの「[記憶領域パラメータの設定](#)」を参照してください。

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を使用して PCTFREE まで満たされます。索引ブロック用に PCTFREE によって確保されている領域は、表に新しい行が挿入されたため、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）に入れなければならない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は、索引の値は（字句設定順に基づいて）別の索引ブロックに入れられます。そのため、索引を付けられ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、あまり挿入されず、表が相対的に静的である場合、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 を小さくできます。

関連項目：PCTUSED は索引には指定できません。PCTFREE パラメータの詳細は、12-2 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。

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、または異なる表領域に作成できます。

表とその索引に対して同じ表領域を使用すると、データベースのメンテナンス（表領域バックアップ、ファイル・バックアップまたはアプリケーションの可用性や更新など）が容易になるかもしれません。この場合、関連するすべてのデータが常にまとめてオンラインになります。

表とその索引に対して異なる表領域（異なるディスク上）を使用すると、表と索引を同じ表領域に格納するよりもディスクの競合が低減するために、よりよいパフォーマンスが得られます。

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作動する保証はありません。

索引作成の平行化

索引作成は平行化できます。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 Oracle の索引作成の速度が速くなります。

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領域パラメータが使用されます。したがって、INITIAL 値が 5M、並行度が 12 で作成された索引は、索引を作成するときに 60MB 以上の記憶領域を使用します。

関連項目：索引作成の並行化の詳細は、『Oracle8i チューニング』を参照してください。

NOLOGGING を指定した索引作成に関する考慮事項

CREATE INDEX 文で NOLOGGING を指定すると、索引を作成して最小限の REDO ログ・レコードを生成できます。

注意： LOGGING 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する必要があります。

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。
- 索引を作成するのに要する時間が削減できます。
- 大規模な索引の平行化作成のパフォーマンスが向上します。

一般に、LOGGING を指定しないで作成した場合、小規模な索引より大規模な索引の方が相対的にパフォーマンスの向上が大きくなります。小規模な索引をログギングなしとして作成しても、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、特に索引作成を並行化したときに、パフォーマンスが著しく向上します。

索引サイズの見積りと記憶領域パラメータの設定

次のような理由で、索引を作成する前に索引のサイズを見積ると有効です。

- 表、ロールバック・セグメント、REDO ログ・ファイルの見積りと合わせた索引の見積りサイズを使用して、作成するデータベースを保持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。
- 個々の索引の見積もりサイズを使用すると、索引が使用するディスク領域をよりよく管理できます。索引を作成するときに、適切な記憶領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 パフォーマンスを向上させることができます。

たとえば、索引を作成する前に、その最大サイズを見積る場合を想定します。索引の作成時に記憶領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ディスク領域のうち比較的連続した部分に格納されます。これによって、この索引を呼び出すディスク 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 程度になります。表と同じように、索引を作成するときにも、記憶領域パラメータを明示的に設定できます。

関連項目：記憶領域パラメータの詳細は、12-7 ページの「[記憶領域パラメータの設定](#)」を参照してください。

索引を合わせる方法

索引断片化（不適切なサイズ設定やサイズの拡大による）が発生した場合は、索引を再構築するか、合わせることができます。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、索引を再構築する場合と合わせる場合のコストと利点を示しています。

表 16-1 再構築する方法と合わせる方法の比較

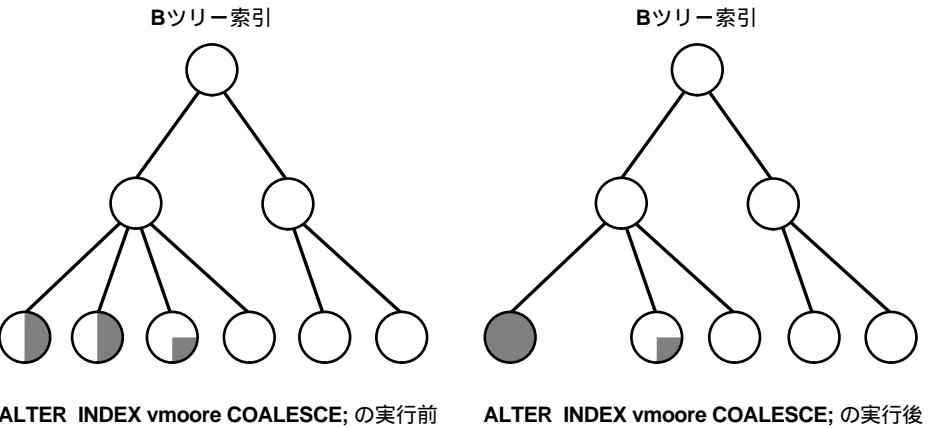
再構築する場合	合わせる場合
索引を別の表領域に高速で移動する場合。	索引は別の表領域には移動できない場合。
コストが大きい場合。ディスク所要量が大きい場合。	コストが小さい場合。ディスク所要量が小さい場合。
該当する場合は、新しいツリーを作成して高さを縮小する場合。	ツリーのうち同じブランチ内のリーフ・ブロックを合わせる場合。
オリジナルの索引を削除しないで、記憶領域パラメータと表領域パラメータを迅速に変更できます。	索引のリーフ・ブロックを迅速に解放できます。

再使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、そのようなリーフ・ブロックを次の文を使用してマージできます。

```
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。この操作を実行する前は、最初の 2 つのリーフ・ブロックは 50% 埋まっており、最初のブロックの断片化を解消して完全に満杯にして、2 番目のブロックを解放できることを意味します（この例では、PCTFREE=0 を想定しています）。

図 16-1 索引を合わせる



制約を使用禁止または削除する前の検討

一意キーと主キーには対応する索引があるため、UNIQUE キー制約や PRIMARY KEY 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、UNIQUE キー制約や PRIMARY KEY 制約に対する索引がかなり大きい場合には、その索引を削除して作成しなおすよりも、その制約を使用可能な状態のままにして時間を節約できます。

索引の作成

ここでは、索引を作成する方法について説明します。この項のトピックは、次のとおりです。

- [制約に対応付けられる索引の作成](#)
- [索引の明示的な作成](#)
- [ファンクション・ベース索引の作成](#)
- [既存の索引の再作成](#)
- [キー圧縮型索引の作成](#)

UNIQUE キーまたは PRIMARY KEY（対応付けられる索引を作成）を使用可能にするには、表の所有者に、索引を収録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。

LOBS 列、LONG 列および LONG RAW 列に索引を付けることはできません。

Oracle は、一意キーまたは主キーに一意索引を作成することによって、UNIQUE キー整合性制約または PRIMARY KEY 整合性制約を施行します。制約を使用可能にすると、この索引は Oracle によって自動的に作成されます。つまり、CREATE TABLE 文や ALTER TABLE 文を発行するユーザーは、索引を作成するために何もする必要はありません。制約を定義し使用可能にすると、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ときの両方であてはまります。

一般に、一意性を施行するには、CREATE UNIQUE INDEX 構文を使用するよりも、制約を作成するほうがよいでしょう。制約に対応する索引は、常にその制約の名前を想定しています。つまり、制約索引に固有の名前は指定できません。

索引に記憶領域オプション（INITIAL と NEXT など）を指定しないと、ホスト表領域のデフォルトの記憶領域オプションが自動的に使用されます。

制約に対応付けられる索引の作成

USING INDEX オプションを指定した ENABLE 句を使用すると、UNIQUE キー制約または PRIMARY KEY 制約に対応する索引の記憶領域オプションを指定できます。次の文は、PRIMARY KEY 制約を定義し、対応する索引の記憶領域オプションを指定します。

```
CREATE TABLE emp (  
    empno NUMBER(5) PRIMARY KEY, . . . )  
    ENABLE PRIMARY KEY USING INDEX  
    TABLESPACE users  
    PCTFREE 0;
```

索引の明示的な作成

SQL コマンド CREATE INDEX を使用して、索引を明示的に（整合性制約以外に）作成します。次の文は、EMP 表の ENAME 列に索引 EMP_ENAME 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、複数の記憶設定が明示的に指定されているので注意してください。

索引のオンラインでの作成

従来、表の索引作成時には、索引構築作業中にその表には常に DML S-lock がありました。これは、構築中は実表に対して DML 操作を実行できないことを意味していました。

このリリースからは、表のサイズが絶えず大きくなり、連続操作が必要になっている状況で、索引をオンラインで作成し、再構築できます。つまり、実表の更新中に、その表の索引を構築または再構築できます。ただし、まだ DML SS-lock があり、オンライン索引構築中に他の DDL 操作は実行できないので注意してください。

つぎの文は、オンライン索引構築操作を実行します。

```
ALTER INDEX emp_name REBUILD ONLINE;  
  
CREATE INDEX emp_name ON emp (mgr, emp1, emp2, emp3) ONLINE;
```

注意： オンライン索引構築中に DML 操作を実行できますが、このプロシージャでは重要 / 大規模な DML 操作を実行しないことをお勧めします。たとえば、既存の表のサイズのうち合計 30% を占める行をロードする場合は、オンライン索引構築の前に実行する必要があります。

関連項目：索引をオンラインで作成するための構文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

ファンクション・ベース索引の作成

ファンクション・ベース索引により、関数や式から戻される値を修飾する問い合わせが可能です。関数や式の値は、事前に計算されて索引に格納されます。

ファンクション・ベースの索引作成には、言語ソート・キー（照合）に基づく言語ソートのサポート、SQL 文の効率的な言語照合、関数を伴う予測評価のための効率的なメカニズムなどの利点があります。

次の文は、Area(geo) の索引を定義します。

```
CREATE INDEX area_index ON rivers (Area(geo)) DESC;
```

area_index は、関数 area(geo) で定義されます。

```
SELECT      id, geo Area(geo), desc
FROM        rivers r
WHERE       Area(geo) >5000;
```

この SQL 文では、Area(geo) が WHERE 句で参照されると、オプティマイザでは索引 area_index を使用するものと見なされます。

注意： 表所有者は、ファンクション・ベース索引で使用される関数に対して EXECUTE 権限を持つ必要があります。また、ファンクション・ベース索引は使用する関数に依存するので、関数の変更時に無効にできます。

関連項目： ファンクション・ベース索引の概念情報は、『Oracle8i 概要』を参照してください。

ファンクション・ベースの索引作成とアプリケーション開発の詳細は、『Oracle8i アプリケーション開発者ガイド - 基礎編』を参照してください。

例 1

次の文は、表 emp のファンクション・ベース索引 idx を作成します。

```
CREATE INDEX idx ON emp (UPPER(emp_name));
```

ここで、SELECT 文は UPPER(emp_name) のファンクション・ベース索引を使用して、KEYCOL などの名前を持つすべての従業員を取り出します。

```
SELECT * FROM emp WHERE UPPER(emp_name) like :KEYCOL;
```

SELECT 文では、索引範囲スキャン（式は索引の接頭辞）または全索引走査（索引で高度な並行性が指定されている場合に望ましい）を使用できます。

```
CREATE INDEX idx ON t (a + b * (c - 1), a, b);
SELECT a FROM t WHERE a + b * (c - 1) < 100;
```

例 2

ファンクション・ベース索引を使用して、NLS ソート索引をサポートすることもできます。NLSSORT は、文字列が与えられているソート・キーを戻す関数です。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 nls_index を作成します。

ここで、NLS_SORT から選択するために、次の文を発行します。

```
SELECT * FROM t_table ORDER BY name;
```

行の順序は、German の照合順を使用して決定されます。

例 3

ファンクション・ベース索引のもう 1 つの用途は、大 / 小文字区別なしの検索を実行することです。

```
CREATE INDEX case_insensitive_idx ON emp_table (UPPER(empname));
```

この新しい索引の問合せは、次のようになります。

```
SELECT * FROM emp_table WHERE UPPER(empname) = 'JOE';
```

例 4

この例も、ファンクション・ベース索引作成の一般的な用途を示しています。つまり、大 / 小文字区別なしのソートと言語ソートです。

```
CREATE INDEX emp1 ON emp  
  (UPPER(ename), NLSSORT(ename));
```

この場合、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るので、NLS_SORT の仕様部は NLSSORT 引数に含まれません。セッション設定で指定されたものと異なる言語を使用する場合は、前述の例の NLSSORT(ename) を次のように置き換えてください。

```
NLSSORT(ename, NLS_SORT='German')
```

この行では、ソートに German 言語ソート・キーを使用するように指示しています。

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 回復を実行する場合、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ければ、その索引には無効を示すマークが設定されます。ANALYZE VALIDATE INDEX 文を使用して、この索引を検証する必要があります。

関連項目： ファンクション・ベースの索引作成の詳細は、『Oracle8i 概要』および『Oracle8i SQL リファレンス』を参照してください。

既存の索引の再作成

既存の索引を再作成または再構築する前に、16-6 ページの表 16-1 の説明に従って両者に伴うコストと利点を比較してください。

既存の索引をデータ・ソースとして使用して索引を作成できます。このようにして索引を作成すると、記憶特性の変更や新たな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作成しなおすと、ブロック内の断片化もなくなります。実際には、索引を削除して CREATE INDEX コマンドを使用するより、既存の索引を作成しなおしたほうがパフォーマンスはよくなります。

既存の索引を作成しなおすには、次の文を発行します。

```
ALTER INDEX index_name REBUILD;
```

REBUILD 句は索引名の直後、他のオプションの前に必要です。また、REBUILD 句は DEALLOCATE UNUSED 句とは併用できません。

関連項目： ALTER INDEX コマンドとオプションの句の詳細は、『Oracle8i SQL リファレンス』を参照してください。

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、キー列の接頭辞値の反復的な発生を排除できます。

キー圧縮によって、索引キーは接頭辞およびサフィクス・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サフィクス・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、索引ブロック当り格納できるキー数が増え、パフォーマンスが改善されます。

キー圧縮は、次のような状況で役立ちます。

- キーを一意にするために ROWID が追加される一意でない索引がある場合。このような状況でキー圧縮を使用すると、重複キーは ROWID なしの索引ブロックに接頭辞エン

リとして格納されます。残りの行は、ROWID のみからなるサフィクス・エントリとなります。

- 一意の複数列索引がある場合。

COMPRESS 句を使用してキー圧縮を使用可能にすることもできます。キー列が接頭辞およびサフィクス・エントリに分割されるときの方法を識別する接頭辞の長さを（キー列数として）指定できます。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 (ename)
  TABLESPACE users
  COMPRESS 1
```

再構築中に COMPRESS 句を指定することもできます。たとえば、再構築中に、次のようにして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 : CREATE INDEX 文の詳細は、『Oracle8i SQL リファレンス』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 システム権限が必要です。索引は、トランザクション・エントリ・パラメータを変更したり、記憶領域パラメータを変更する場合以外は変更できません。索引の列構造は変更できません。

主キーと一意キーの整合性制約を適用するために、Oracle によって作成される索引も含めて、どの索引の記憶領域パラメータも、SQL コマンド ALTER INDEX 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
  INITRANS 5
  MAXTRANS 10
  STORAGE (PCTINCREASE 50);
```

索引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、MAXTRANS の新しい設定が索引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）に適用される一方、INITRANS の新しい設定はその後割り当てられるブロックだけに適用されます。

記憶領域パラメータ INITIAL と MINEXTENTS は変更できません。他の記憶領域パラメータの新しい設定はすべて、その後索引に割り当てられるエクステンツにしか影響しません。

整合性制約をインプリメントする索引では、USING INDEX オプションを指定した ENABLE 句を含む ALTER TABLE 文を発行することによって、記憶領域パラメータの調整もできます。たとえば、次の文は先に定義した索引の記憶領域オプションを変更します。

```
ALTER TABLE emp
  ENABLE PRIMARY KEY USING INDEX
  PCTFREE 5;
```

索引の領域使用を監視

索引のキー値を頻繁に挿入、更新および削除する場合、索引がその取得した領域を効果的に何度も使用するかどうかはわかりません。そこで、最初に索引の構造を分析することによって、定期的な間隔で索引の領域使用の効率を監視した後、次のように INDEX_STATS ビューを問い合わせるとよいでしょう。

```
SELECT pct_used FROM sys.index_stats WHERE name = 'indexname';
```

索引の領域使用の割合は、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、索引の平均的な領域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
- PCT_USED のチェック
- 索引の削除および再作成（または結合）

索引の領域使用がその平均を下回るときは、索引を削除してから再構築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：索引の構造分析の詳細は、20-3 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。

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 システム権限が必要です。

索引を削除する理由には、次のようなものがあります。

- 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改善をもたらしていない場合。（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が、索引エントリが非常に少ないなど）。
- 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、再構築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、再構築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、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に作成したかという索引の作成方法に依存します。

注意： 表が削除されると、対応する索引はすべて自動的に削除されます。

使用可能になっている UNIQUE キー制約や PRIMARY KEY 制約に対応付けられた索引だけの削除はできません。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、または削除してください。

```
DROP INDEX emp_ename;
```

関連項目： 索引分析の詳細は、20-3 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。

制約に対応する索引の削除の詳細は、20-12 ページの「[整合性制約の管理](#)」を参照してください。

クラスタの管理

この章では、クラスタ（クラスタ化された表と索引を含む）を管理する方法について説明します。この章のトピックは、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](#)

この章で説明する作業を実行する前に、[第 12 章「スキーマ・オブジェクトを管理するためのガイドライン」](#)の内容をよく理解しておいてください。

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オプションとして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おり、そのデータ・ブロックは共通の列を共有し、一緒に使用されることが多いためにグループ化されています。たとえば、EMP 表および DEPT 表は DEPTNO 列を共有しています。EMP 表および DEPT 表をクラスタ化する場合（[図 17-1](#) 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。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。

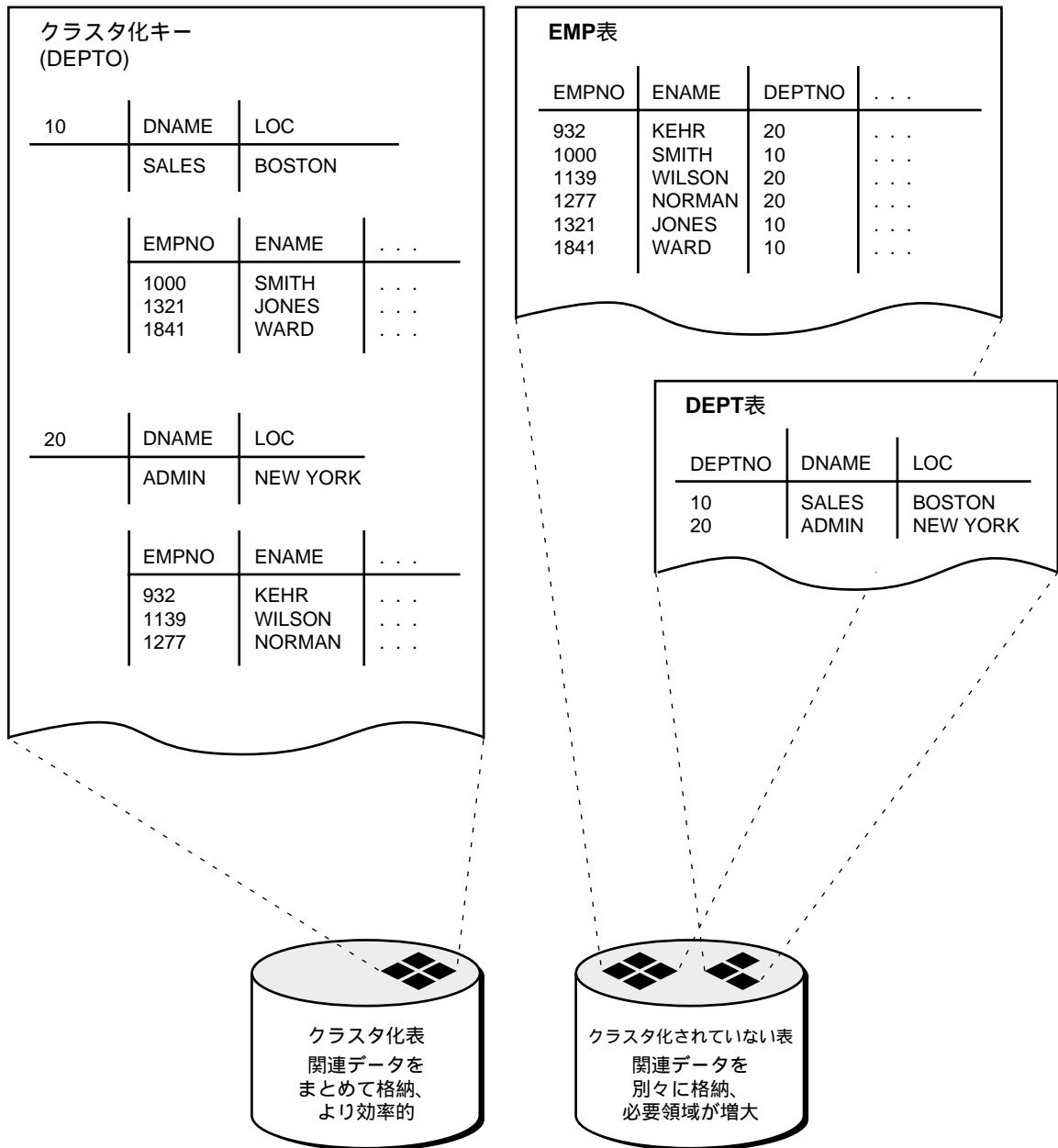
クラスタは別々の表の関連する行を同じデータ・ブロックに格納するため、クラスタを正しく使用すると利点として主に次の 2 つがあります。

- クラスタ化した表の結合のため、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、クラスタ化した表が共通に持ってい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、異なる表のどれだけ多くの行にその値が含まれているかに関係なく、クラスタとクラスタ索引のそれぞれに 1 度だけ格納されます。

このため、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領域は、クラスタ化されてない表形式の場合に比べて少なくて済みます。これは、各クラスタ・キー（各 DEPTNO）が EMP 表および DEPT 表の両方の中の同じ値を含んでいる多数の行について 1 度だけ格納されるということを考えればよくわかります。

クラスタを作成した後で、そのクラスタ内に表を作成できます。ただし、クラスタ化した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、クラスタ化した表に対してさらに索引を作成することには影響しません。通常の方法で作成および削除できます。

図 17-1 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を説明します。この項のトピックは、次のとおりです。

- [クラスタに対する適切な表の選択](#)
- [クラスタ・キーに対する適切な列の選択](#)
- [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位置の指定](#)
- [クラスタ・サイズの見積りおよび記憶領域パラメータの設定](#)

関連項目：クラスタの削除の詳細は、『Oracle8i 概要』を参照してください。

クラスタに対する適切な表の選択

問合せを主とした（挿入または更新をあまりしない）1 つ以上の表を格納するため、その問合せが頻繁にクラスタ内の複数の表のデータを結合したり、またはその単一の表から関連したデータを検索したりする場合に、クラスタを使用してください。

クラスタ・キーに対する適切な列の選択

クラスタ・キー列を慎重に選択してください。表を結合する問合せで、複数の列が使用される場合、クラスタ・キーを複合キーにしてください。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。

各キー値に対応している行のグループがおおよそ 1 つのデータ・ブロックを満杯にするように、よいクラスタ・キーは十分な一意の値を持ちます。クラスタ・キー値あたりの行が少なすぎると、領域を浪費し、パフォーマンスが低下する結果となります。少しの行のみが共通の値を共有するような特性のクラスタ・キーは、クラスタを作成するときに小さなサイズを指定しない限り、ブロック内の領域を浪費する可能性があります（下記参照）。

クラスタ・キー値あたりの行が多すぎると、そのキーに対する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。あまりに一般的な値（たとえば性別）に対するクラスタ・キーは、過度の検索を必要とし、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可能性があります。

クラスタ索引は一意にできません。また、LONG として定義した列を含むことはできません。

関連項目：適切な索引の特性の詳細は、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。

データ・ブロック領域使用の指定

クラスタを作成するときに、PCTFREE パラメータと PCTUSED パラメータを指定することによって、領域使用率、および現在行への更新のために、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変えることができます。クラスタ内の表に設定された PCTFREE パラメータと PCTUSED パラメータは、無視されることに注意してください。つまり、クラスタ化した表はクラスタの設定を自動的に使用します。

関連項目 : PCTFREE と PCTUSED の指定の詳細は、12-2 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 コマンド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に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・ブロック内に置かれるクラスタ・キーの数を制限する場合。これによって、クラスタ内のキーの記憶域効率を最大にします。

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かでも使用できます。

デフォルトでは、Oracle は 1 つのクラスタ・キーとそれに対応付けられている行をそのクラスタのデータ・セグメントの各データ・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・システムによって違いますが、クラスタ化した表が他のマシン上の他のデータベースにインポートされるときにも、ブロックあたり 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、ブロックはまとめて連鎖され、キーによるすべての値へのアクセス速度を向上させます。クラスタ索引は、各クラスタ・キー値に対応する行を内容とする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位置の指定

適切な権限と表領域割当て制限を持っていると、現在、オンライン状態の表領域内に新しいクラスタを作成できます。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ためには、CREATE CLUSTER/INDEX 文に TABLESPACE オプションを必ず指定してください。

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し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・サイズの見積りおよび記憶領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。

- 索引、ロールバック・セグメントおよび REDO ログ・ファイルの見積りと合わせたクラスタの見積りサイズを使用して、特定のデータベースを保持するために必要なディスク容量を決定できる。この見積りを利用して適切なハードウェアの購入とその他の意思決定ができます。
- 個々のクラスタの見積りサイズを使用すると、クラスタが使用するディスク領域を適切に管理できます。クラスタを作成するときに、適切な記憶領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、クラスタ化されてない表を作成するときは記憶領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領域パラメータを設定しないと、その表が常駐する表領域に設定されたデフォルト記憶領域パラメータが自動的に使用されます。クラスタ化した表はクラスタの記憶領域パラメータも自動的に使用します。

クラスタの作成

ここでは、クラスタの作成方法について説明します。この項のトピックは、次のとおりです。

- [クラスタ化された表の作成](#)
- [クラスタ索引の作成](#)

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含む予定の表領域に対する割当て量を持っているか、または UNLIMITED TABLESPACE システム権限が必要です。

別のユーザーのスキーマにクラスタを作成するには CREATE ANY CLUSTER システム権限が必要です。さらに、所有者はそのクラスタを含む予定の表領域に対する割当て量を持っているか、または UNLIMITED TABLESPACE システム権限を持つ必要があります。

SQL 文 CREATE CLUSTER を使用して、クラスタを作成します。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);
```

クラスタ化された表の作成

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には、表領域割当て制限または UNLIMITED TABLESPACE システム権限を必要としません。

CLUSTER オプションを指定した SQL 文 CREATE TABLE を使用して、クラスタに表を作成できます。たとえば、次の文を使用して、EMP 表と DEPT 表を EMP_DEPT クラスタに作成できます。

```
CREATE TABLE dept (  
    deptno NUMBER(3) PRIMARY KEY, . . . )  
    CLUSTER emp_dept (deptno);
```

```
CREATE TABLE emp (  
    empno NUMBER(5) PRIMARY KEY,  
    ename VARCHAR2(15) NOT NULL,  
    . . .  
    deptno NUMBER(3) REFERENCES dept)  
    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、クラスタを含むスキーマと異なるスキーマ内にあってもかまいません。また、列名は一致しなくてもかまいませんが、その構造は同じにする必要があります。

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 1 つが真でなければなりません。

- 自分のスキーマ内にクラスタが含まれ、さらに CREATE INDEX システム権限を持っていること。
- CREATE ANY INDEX システム権限を持っていること。

どちらの場合も、クラスタ索引を収録する表領域に対する割当て量、または UNLIMITED TABLESPACE システム権限が必要です。

クラスタ化した表に行を挿入する前に、クラスタ索引を作成してください。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;
```

いくつかの記憶設定が、クラスタとクラスタ索引に対して明示的に指定されています。

関連項目：システム権限の詳細は、[第 24 章「ユーザー権限とロールの管理」](#)を、表領域割当て制限の詳細は、[第 23 章「ユーザーとリソースの管理」](#)を参照してください。

クラスタの変更

既存のクラスタを変更して、次の設定を変更できます。

- データ・ブロック領域使用パラメータ (PCTFREE、PCTUSED)
- 平均のクラスタ・キー・サイズ (SIZE)
- トランザクション・エントリの設定 (INITRANS、MAXTRANS)
- 記憶領域パラメータ (NEXT、PCTINCREASE)

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 システム権限が必要です。

クラスタのデータ・ブロック領域使用パラメータ (PCTFREE と PCTUSED) またはクラスタ・サイズ・パラメータ (SIZE) を変更するときには、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、必要なときに (ただちにではなく) 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、MAXTRANS) を変更するときには、MAXTRANS の新しい設定が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) に適用される一方、INITRANS 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。

記憶領域パラメータ INITIAL と MINEXTENTS は変更できません。他の記憶領域パラメータの新しい設定はすべて、その後クラスタに割り当てられるエクステンツにしか影響しません。

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;
```

クラスタ表とクラスタ索引の変更

SQL の ALTER TABLE 文を使用して、クラスタ表を変更できます。ただし、クラスタ化した表に対して ALTER TABLE 文でどのようなデータ・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領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。」) が出されます。これは、Oracle はそのクラスタのパラメータをすべてのクラスタ化した表に対して使用するからです。そのため、ALTER TABLE コマンドは、クラスタ化した表に対して、列の追加や修正、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にするためにしか使用できません。

注意： クラスタ索引のサイズを見積るときには、索引は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。各キーは索引内に 1 度しか現れません。

クラスタに記憶領域を手動で割り当てる方法

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、Oracle Parallel Server を使用しているとき、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。

新しいエクステントは、ALLOCATE EXTENT オプションを指定した ALTER CLUSTER 文を使用して、クラスタに割り当てることができます。

関連項目：表の変更の詳細は、14-10 ページの「[表の変更](#)」を参照してください。

クラスタ索引は、他の索引と同じように変更します。詳細は、16-12 ページの「[索引の変更](#)」を参照してください。

ALTER CLUSTER 文の CLUSTER パラメータの詳細は、『Oracle8i Parallel Server 概要および管理』を参照してください。

クラスタの削除

ここでは、クラスタの削除について説明します。この項のトピックは、次のとおりです。

- [クラスタ化した表の削除](#)
- [クラスタ索引の削除](#)

クラスタ内の表が不要になった場合、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利用できるようになります。

クラスタ化した表の削除

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 システム権限が必要です。クラスタ化した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ありません。

クラスタ化した表は、その表のクラスタ、他のクラスタ化した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。クラスタ化した表は、クラスタ化されて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。

注意： 単一の表をクラスタから削除するとき、Oracle は表の各行を個々に削除します。クラスタ全体を削除する効率を最大にするには、INCLUDING TABLES オプション付きの DROP CLUSTER 文を使用して、表も含めて、そのクラスタを削除してください。クラスタの残りの部分をそのままにしておくときのみ、(DROP TABLE 文を使用して) クラスタから個々の表を削除してください。

関連項目： 表を削除する方法の詳細は、14-11 ページの「[表の削除](#)」を参照してください。

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した表に影響を及ぼさずに削除できます。ただし、クラスタ索引が存在しないと、クラスタ化した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、クラスタ索引を作成してください。断片化したクラスタ索引を再構築する手順の一部として、クラスタ索引が削除されることがあります。

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、SQL の 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された表が含まれ、その表も削除する場合は、DROP CLUSTER 文の INCLUDING TABLES オプションを次のように追加してください。

```
DROP CLUSTER emp_dept INCLUDING TABLES;
```

INCLUDING TABLES オプションを指定していないのに、クラスタに表が含まれているとエラーが戻されます。

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限り、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のに、CASCADE CONSTRAINTS オプションを使用しないと、Oracle はエラーを戻します。

関連項目 : 索引の削除の詳細は、16-13 ページの「[索引の削除](#)」を参照してください。

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。この章のトピックは、次のとおりです。

- [ハッシュ・クラスタを管理するためのガイドライン](#)
- [ハッシュ・クラスタの変更](#)
- [ハッシュ・クラスタの削除](#)

関連項目 : この章で説明する作業を実行する前に、[第 12 章「スキーマ・オブジェクトを管理するためのガイドライン」](#)の内容をよく理解しておいてください。

ハッシュ・クラスタを管理するためのガイドライン

ここでは、ハッシュ・クラスタの管理を実行する前に考慮すべきガイドラインについて説明します。この項のトピックは、次のとおりです。

- [ハッシングの長所](#)
- [ハッシングの短所](#)
- [ハッシュ・クラスタが必要とするサイズの見積りおよび記憶領域パラメータの設定](#)

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、索引を持つクラスタ化しない表または索引クラスタに対して選択肢を提供します。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・クラスタを作り、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します。

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複合キー（複数列キー）でもかまいません。ハッシュ・クラスタ内の行の検索または格納を行う場合、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、以後、Oracle は、発行された文のためにその読み込みや書き込みをします。

索引付きの表またはクラスタ内の行の検索または格納のためには、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、索引内のキー値の検索または格納の実行
- 別のI/Oによる、表またはクラスタ内の行の読み取りまたは書き込みの実行

一方、Oracle はハッシュ関数を使用してハッシュ・クラスタ内の行の位置を突き止めるので、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み込みや書き込みに必要とされるのは最小限のI/O操作だけになります。

関連項目：ハッシュ・クラスタの詳細は、『Oracle8i 概要』を参照してください。

ハッシングの長所

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。

ハッシングは、次のような状況で最も有効です。

- 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、等価条件を満たすクラスタ・キーがハッシュされ、対応するハッシュ・キーは通常一度の読み込みで見つかります。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見つけなければならず（通常複数回の読み込み）、その後で行が表から読み込まれます（別の読み込み）。

- ハッシュ・クラスタ内の表のサイズが初めに固定されていて、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可能性があります。

ハッシングの短所

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、全表走査、または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は使用できません。これにかわって、全表走査と同等の機能を実行して問合せの行を取り出す必要があります。

```
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、絶えず大きくなる場合。表が無制限に大きくなる場合、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できません。
- アプリケーションが表に対して頻繁に全表走査を実行し、表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、処理に必要な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。

関連項目：ハッシュ・クラスタの作成とハッシュ関数の指定の詳細は、『Oracle8i SQL リファレンス』を参照してください。

ハッシュ関数とユーザー定義のハッシュ関数の指定の詳細は、『Oracle8i 概要』を参照してください。

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が付いている可能性があります。追加推奨事項については、『Oracle8i 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。

ハッシュ・クラスタが必要とするサイズの見積りおよび記憶領域パラメータの設定

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領域を見積ることは重要です。

Oracle は、領域の初期割当てが、設定 SIZE と HASHKEYS を使用すればハッシュ表を格納するのに十分であることを保証します。記憶領域パラメータ INITIAL、NEXT および MINEXTENTS の設定がハッシュ表サイズを考慮していない場合、少なくとも $\text{SIZE} \times \text{HASHKEYS}$ 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、利用可能なデータ領域がブロックあたりおおよそ 1900 バイト（データ・ブロック・サイズからオーバーヘッドを引く）と想定し、CREATE CLUSTER コマンドに STORAGE パラメータと HASH パラメータを指定すると次のようになります。

```
STORAGE (INITIAL 100K
          NEXT 150K
          MINEXTENTS 1
          PCTINCREASE 0)
SIZE 1500
HASHKEYS 100
```

この例では、データ・ブロックあたり、ハッシュ・キーを 1 つだけ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（ $100 \times 2\text{KB}$ ）です。記憶領域パラメータの設定はこの要件を考慮していません。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るとします。

```
SIZE 500 HASHKEYS 100
```

この場合、データ・ブロックあたり、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（ $34 \times 2\text{KB}$ ）です。記憶領域パラメータの初期設定は、この要件を満たしています（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・クラスタの作成

ハッシュ・クラスタの作成後に、そのクラスタ内に表を作成できます。ハッシュ・クラスタは、SQL コマンド CREATE CLUSTER を使用して作成できます。次の文は TRIAL 表を格納するクラスタ TRIAL_CLUSTER を作り、TRIALNO 列によってクラスタ化します。

```
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);
```

次の部分では、ハッシュ・クラスタ固有の CREATE CLUSTER コマンドのパラメータの設定について説明します。

関連項目 : クラスタ内に表を作成する方法、CREATE CLUSTER コマンドの他のパラメータの設定のガイドラインおよびハッシュ・クラスタを作成するために必要な権限の詳細は、17-6 ページの「[クラスタの作成](#)」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でなければなりません。つまり、ハッシュ・キーとデータ行の間には、1 対 1 のマッピングが必要です。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;
```

HASHKEY 値は近似の素数に丸められるので、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： この単一表オプションは、ハッシュ・クラスタにのみ有効です。
また、HASHKEYS も指定する必要があります。

関連項目 : CREATE CLUSTER 文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

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、クラスタ化した表に対して最もよく発行される問合せのタイプに依存しています。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、EMPNO 列をクラスタ・キーにするとういでしょう。問合せが部門番号によって行を頻繁に選択する場合には、DEPTNO 列をクラスタ・キーにしてください。単一の表を含むハッシュ・クラスタの場合、通常クラスタ・キーをその含まれる表の主キー全体にします。

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複合キー（複数列キー）でもかまいません。複合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、その内容が均一に分布した整数である場合だけにしてください。この条件が当てはまる場合、それぞれの一意のクラスタ・キー値が衝突なしで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布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しないで内部ハッシュ関数を使用してください。

SIZE を設定する

SIZE は、ハッシュ・キーに対して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てください。そのため、SIZE を適切に決定するには、自分のデータの特徴をよく理解する必要があります。

- ハッシュ・クラスタが表を 1 つだけ収録し、その表の行のハッシュ・キー値が一意（値あたり 1 行）であれば、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のであれば、SIZE は代表ハッシュ値に対応付けられているすべての行を保持するために必要領域の平均サイズに設定できます。
- ハッシュ・クラスタが、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、見積りのとおりに SIZE を設定できます。この場合、衝突は発生せず、領域は可能な限り効果的に使用されます。
- 挿入に関して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。衝突が頻繁な場合にブロックのオーバーフローの可能性を軽減し、パフォーマンスを最大にするには、表 18-1 に従って SIZE を大きくする必要があります。

表 18-1 SIZE の増加チャート

ブロック当りの使用可能領域 / 算出されたサイズ	SIZE の設定
1	算出された SIZE
2	算出された SIZE +15%
3	算出された SIZE +12%
4	算出された SIZE + 8%
>4	算出された SIZE

ただし、SIZE の値を過大に見積ると、クラスタ内の未使用の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であれば、上の調整を無視して SIZE に見積りの値を使用してください。

HASHKEYS を設定する

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 値を近似の素数に丸めます。

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。すべての例について、データ・ブロック・サイズは 2KB であり、平均して各ブロックの 1950 バイトが利用可能なデータ領域です（ブロック・サイズからオーバーヘッドを引く）。

例 1

ハッシュ・クラスタに EMP 表をロードします。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。常に EMP 表の最大行数は 10000 であり、平均の行サイズが 55 バイトとして見積られています。

この場合は、EMPNO をクラスタ・キーにします。この列は一意的な整数を持っているので、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。データ・ブロックあたり、34 のハッシュ・キーが割り当てられることに注目してください。HASHKEYS は、表の行数（10000）より大きい最初の素数（10007）に切り上げたものに設定できます。

```
CREATE CLUSTER emp_cluster (empno  
NUMBER)  
.  
.  
.  
SIZE 55  
HASH IS empno HASHKEYS 10007;
```

例 2

先の例と同様の条件を考えます。ただし、この場合、行は通常部門番号によって検索されます。部門あたり平均 10 名の従業員を持つ部門が、最大で 1000 部門存在します。部門番号が 10 ずつ増加する (0、10、20、30、...) ことに注目してください。

この場合は、DEPTNO をクラスタ・キーにします。この列は一樣に分布する整数を持っているので、内部ハッシュ関数は無視できます。事に見積った SIZE (部門あたり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、データ・ブロックあたり、3 つのハッシュ・キーだけを割り当てることができます。いくらかの衝突を予想し、データ検索の最大パフォーマンスが必要であれば、オーバーフロー・ブロックを必要とすることによ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。SIZE を 12% だけ調整して 620 バイトにすることによって (SIZE の設定に関する前項を参照してください)、データ・ブロックあたり 3 つのハッシュ・キーのみが割り当てられ、予想される衝突の行のためにより多くの領域を残します。

HASHKEYS は、一意の部門番号の数 (1000) より大きい最初の素数 (1009) に切り上げたものに設定できます。

```
CREATE CLUSTER emp_cluster (deptno NUMBER)  
.  
.  
.  
SIZE 620  
HASH IS deptno HASHKEYS 1009;
```

ハッシュ・クラスタの変更

ハッシュ・クラスタは、SQL の ALTER CLUSTER 文を使用して変更します。

```
ALTER CLUSTER emp_dept . . . ;
```

ハッシュ・クラスタの変更に関係する問題は、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。クラスタを作りなおしてこれらのパラメータを変更し、元のクラスタからデータをコピーする必要があります。

関連項目 : 索引クラスタの変更の詳細は、17-8 ページの「[クラスタの変更](#)」を参照してください。

ハッシュ・クラスタの削除

ハッシュ・クラスタは、SQL の DROP CLUSTER 文を使用して削除できます。

```
DROP CLUSTER emp_dept;
```

ハッシュ・クラスタ内の表が、SQL 文 DROP TABLE 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の問題は、索引クラスタの場合と同じです。

関連項目 : クラスタの削除の詳細は、17-9 ページの「[クラスタの削除](#)」を参照してください。

データ・ブロック破損の検出と修復

Oracle には、データ・ブロックの破損を検出して修正できるように、さまざまな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が、この方法が常に可能または望ましいとは限りません。データ・ブロックの破損が行のサブセットに限られている場合は、破損した行を除くすべてのデータを選択して表を再構築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、可能であれば破損に対処し、再構築または修復中もオブジェクトを引き続き使用できます。DBMS_REPAIR では、破損に対処するために次のアプローチが使用されます。

- ステップ 1: 破損を検出してレポート
- ステップ 2: DBMS_REPAIR を使用してコストと利点を評価
- ステップ 3: オブジェクトの使用可能化
- ステップ 4: 破損を修復して失われたデータの再構築

注意： データの損害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。したがって、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適切かどうかを判断する必要があります。修復の性質によっては、データが失われ、論理の整合性が損なわれることがあるので、DBMS_REPAIR の使用に伴う利害得失を検討してください。

DBMS_REPAIR パッケージの内容

表 19-1 は、DBMS_REPAIR パッケージを構成するプロシージャを示しています。

表 19-1 DBMS_REPAIR のプロシージャ

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	ブロックに破損を示すマークを付けます（従来は、check_object プロシージャで識別）。
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レポートします。
rebuild_freelists	オブジェクトの空きリストを再構築します。
skip_corrupt_blocks	使用時には、表と索引の走査中に破損マークが付いたブロックは無視されます。使用しなければ、破損マークが付いたブロックの検出時にエラー ORA-1578 が戻されます。
admin_tables	DBMS_REPAIR の Repair Table および Orphan Key Table 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。

ステップ 1: 破損を検出してレポート

DBMS_REPAIR を使用する前に、破損を検出してレポートする必要があります。レポートはブロック内の問題を示すだけでなく、それに対応する修復ダイレクティブも識別されます。DBMS_REPAIR の他にも、破損検出用に複数のオプションがあります。表 19-2 は、各種の検出方法を示しています。

表 19-2 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。 Repair Table 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行ユーティリティ。
ANALYZE	VALIDATE STRUCTURE オプションと併用すると、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チェックまたは検証されます。
DB_BLOCK_CHECKING	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、ブロック・チェックは索引とデータ・ブロックに対してそれぞれ実行されます。

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 を実行した場合に発生する修正も識別されます。この情報は Repair Table の移入によって使用可能になるので、最初に admin_tables プロシージャで Repair Table 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、Repair Table 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ダイ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、DB_VERIFY をオフライン診断ユーティリティとして使用します。

関連項目 : DB_VERIFY の詳細は、『Oracle8i ユーティリティ・ガイド』を参照してください。

ANALYZE: 破損のレポート

ANALYZE TABLE...VALIDATE STRUCTURE 文は、分析されるオブジェクトの構造を検証します。構造が正常に検証される場合は、それを確認するメッセージが戻されます。オブジェクトの構造内で破損が検出されると、エラー・メッセージが戻されます。この場合は、オブジェクトを削除して再作成してください。

関連項目 : ANALYZE 文の詳細は、『Oracle8i 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

インスタンスのブロック・チェックは、DB_BLOCK_CHECKING パラメータを介して設定できます（デフォルト値は TRUE）。これにより、データ・ブロックと索引ブロックは変更時にチェックされます。DB_BLOCK_CHECKING は、ALTER SYSTEM SET 文で変更可能な動的パラメータです。

ステップ 2: DBMS_REPAIR を使用してコストと利点を評価

DBMS_REPAIR を使用する前に、使用した場合の利点と問題点を検討する必要があります。また、破損オブジェクトに使用可能な他の選択肢も検証する必要があります。

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。

破損および修復アクションの有無を判断するには、`check_object` プロシージャを実行して Repair Table を問い合わせます。

2. ブロック破損の取扱いに使用可能な他のオプションがありますか。

データを他のソースから使用可能であれば、そのオブジェクトを削除し、再作成して移入し直します。もう 1 つの方法は、破損表から `CREATE TABLE...AS SELECT` 文を発行して新しい表を作成することです。

`SELECT` 文から破損行を除外して、破損を無視できます。

メディア回復を実行してください。

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、どんな論理的な誤りや副作用を生じるか。これらの問題に対処できるか。そのためにはどんな作業が必要ありますか。

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。ただし、まだ正常にアクセスできる行があっても、ブロックに破損マークが付いている可能性もあります。

ブロックに破損マークが付いている場合は、参照整合性制約に対する違反が考えられます。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、制約を正常に再使用できます。

表にトリガーが定義されている場合は、誤りを生じることがあります。たとえば、行が再度挿入されるときに、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のトリガーとその使用を理解する必要があります。

空きリスト・ブロックがアクセス不能になることがあります。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初期化しなおされます。空きリストにないはずのブロックが含まれている場合があります。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。この問題には、最初に `dump_orphan_keys` プロシージャを実行すれば（破損データの再構築中に役立つキーから情報を取得するため）対処できます。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。`dump_orphan_keys` プロシージャを使用すると、この情報を取り出すことができます。もちろん、この方法でデータを取り出せるかどうかは、索引と表にどの程度の冗長性があるかによって異なります。

ステップ 3: オブジェクトの使用可能化

このステップで、`DBMS_REPAIR` は表と索引の走査中に破損を無視して、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残っている破損をスキップする環境を設定し、破損オブジェクトを使用可能にします。

破損がデータ・ブロック内の不良行のようなデータの損害を伴う場合、この種のすべてのブロックには `fix_corrupt_blocks` プロシージャによって破損マークが設定されます。そこで、`skip_corrupt_blocks` プロシージャを実行し、オブジェクトについて破損マークが付いているブロックをスキップできます。スキップするように設定すると、表および索引の走査は破損マーク付きのすべてのブロックをスキップします。これは、メディアとソフトウェアの両方の破損ブロックに適用されます。

破損ブロックをスキップする操作の含意

索引と表が同期化されていない場合、ある問合せでは索引のみをブロープし、後続の問合せでは索引と表の両方をブロープするような状況では、`SET TRANSACTION READ ONLY` トランザクションの一貫性がなくなることがあります。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戻すため、読取り専用トランザクションの規則違反となります。この場合のアプローチの 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、同じ行を問い合わせても、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。

ステップ 4: 破損を修復して失われたデータの再構築

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回復

`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。この種のすべての索引エントリは、破損のキーと行 ID を格納する Orphan Key Table に挿入されます。

索引エントリ情報が取り出された後に、`ALTER INDEX REBUILD ONLINE` 文を使用して索引を再構築できます。

rebuild_freelists プロシージャを使用した空きリストの修復

「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、その空きリストが初期化しなおされ、エラーが戻されます。これにより破損ブロックは空きリストから取り出されますが、破損ブロックに続くすべてのブロックには、空きリストからアクセスできなくなります。

`rebuild_freelists` プロシージャを使用して、空きリストを初期化しなおすことができます。オブジェクトが走査され、ブロックを空きリストに入れることが適切であれば、マスター空きリストに追加されます。複数の空きリスト・グループがある時は、一度に 1 ブロックずつ公正に割り振られます。再構築中、オブジェクト内で破損マーク付きのブロックは無視されます。

制約および制限事項

DBMS_REPAIR プロシージャには、次の制約があります。

- LOBS 付き表、ネスト表および VARRAYS を持つ表はサポートされますが、行外の列は無視されます。
- クラスタは、`skip_corrupt_blocks` および `rebuild_freelist` 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 の索引は、サポートされません。
- `dump_orphan_keys` プロシージャは、ビットマップ索引やファンクション・ベース索引には機能しません。
- `dump_orphan_keys` プロシージャでは、長さ 3,950 バイト以内のキーが処理されます。

DBMS_REPAIR のプロシージャ

ここでは、DBMS_REPAIR の各プロシージャについて詳しく説明します。

check_object

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、破損および修復ダイレクティブに関する情報を Repair Table に移入します。検証は、オブジェクト内のすべてのブロックをチェックするブロックからなっています。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。

```
procedure check_object(schema_name IN varchar2,  
    object_name IN varchar2,  
    partition_name IN varchar2 DEFAULT NULL,  
    object_type IN binary_integer DEFAULT TABLE_OBJECT,  
    repair_table_name IN varchar2 DEFAULT 'REPAIR_TABLE',  
    flags IN binary_integer DEFAULT NULL,  
    relative_fno IN binary_integer DEFAULT NULL,  
    block_start IN binary_integer DEFAULT NULL,  
    block_end IN binary_integer DEFAULT NULL,  
    corrupt_count OUT binary_integer)
```

表 19-3 check_object プロシージャ

引数	説明
schema_name	チェックするオブジェクトのスキーマ名。
object_name	チェックする表または索引の名前。
partition_name (オプション)	チェックするパーティションまたはサブパーティションの名前。パーティション・オブジェクトに対し、partition_name を指定していなければ、すべてのパーティションとサブパーティションがチェックされます。パーティション・オブジェクトで、指定したパーティションにサブパーティションが含まれている場合は、すべてのサブパーティションがチェックされます。
object_type (オプション)	処理するオブジェクトの型。TABLE_OBJECT または INDEX_OBJECT でなければなりません。デフォルトは TABLE_OBJECT です。
repair_table_name (オプション)	移入する Repair Table の名前。表は SYS スキーマに存在する必要があります。admin_tables プロシージャを使用して Repair Table を作成します。デフォルト名は 'REPAIR_TABLE' です。
flags (オプション)	将来のために確保されています。
relative_fno (オプション)	相対ファイル番号。ブロックの範囲を指定するときに使用します。
block_start (オプション)	ブロックの範囲を指定した場合に最初に処理するブロック。オブジェクトが単一の表、パーティションまたはサブパーティションの場合にのみ指定できます。
block_end (オプション)	ブロックの範囲を指定した場合に最後に処理するブロック。オブジェクトが単一の表、パーティションまたはサブパーティションの場合にのみ指定できます。 block_start と block_end の一方しか指定しなければ、他方はそれぞれデフォルトでファイルの先頭ブロックまたは最終ブロックに設定されます。
corrupt_count	レポートされる破損の数。

fix_corrupt_blocks

このプロシージャを使用し、check_object プロシージャによって前に生成された Repair Table 内の情報に基づいて、指定したブロック内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、そのブロックをチェックしてまだ破損しているかどうかを確認する必要があります。破損ブロックは、ブロックに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。修復が有効になると、Repair Table 内で対応付けられている行が修正タイムスタンプで更新されます。

```
procedure fix_corrupt_blocks(
    schema_name IN varchar2,
    object_name IN varchar2,
    partition_name IN varchar2 DEFAULT NULL,
    object_type IN binary_integer DEFAULT TABLE_OBJECT,
    repair_table_name IN varchar2 DEFAULT 'REPAIR_TABLE',
    flags IN boolean DEFAULT NULL,
    fix_count OUT binary_integer)
```

表 19-4 fix_corrupt_blocks プロシージャ

引数	説明
schema_name	スキーマ名。
object_name	修正対象の破損ブロックを持つオブジェクトの名前。
partition_name (オプション)	処理するパーティションまたはサブパーティションの名前。パーティション・オブジェクトに対し、partition_name を指定していなければ、すべてのパーティションとサブパーティションが処理されます。パーティション・オブジェクトで、指定したパーティションにサブパーティションが含まれている場合は、すべてのサブパーティションが処理されます。
object_type (オプション)	処理するオブジェクトの型。TABLE_OBJECT または INDEX_OBJECT である必要があります。デフォルトは TABLE_OBJECT です。
repair_table_name (オプション)	修復ダイレクティブを持つ Repair Table の名前。SYS スキーマに存在する必要があります。
flags (オプション)	将来のために確保されています。
fix_count	修正されたブロック数。

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 Orphan Key Table に 1 行ずつ挿入されます。

Repair Table を指定すると、実表に対応付けられている破損ブロックと、ソフトウェア破損マークが付いているすべてのデータ・ブロックが処理されます。それ以外の場合は、破損マークが付いているブロックのみが処理されます。

この情報は、表内で失われた行を再構築する場合や診断に使用します。

```
procedure dump_orphan_keys(
  schema_name IN varchar2,
  object_name IN varchar2,
  partition_name IN varchar2 DEFAULT NULL,
  object_type IN binary_integer DEFAULT INDEX_OBJECT,
  repair_table_name IN varchar2 DEFAULT 'REPAIR_TABLE',
  orphan_table_name IN varchar2 DEFAULT 'ORPHAN_KEY_TABLE',
  key_count OUT binary_integer)
```

表 19-5 dump_orphan_keys プロシージャ

引数	説明
schema_name	スキーマ名。
object_name	オブジェクト名。
partition_name (オプション)	処理するパーティションまたはサブパーティションの名前。 パーティション・オブジェクトに対し、partition_name を指 定していなければ、すべてのパーティションとサブパーティ ションが処理される。パーティションオブジェクトで、指定し たパーティションにサブパーティションが含まれている場合は、 すべてのサブパーティションが処理されます。
object_type (オプション)	処理するオブジェクトの型。デフォルトは INDEX_OBJECT で す。
repair_table_name (オプション)	実表内の破損ブロックに関する情報を含む Repair Table の名前。 指定する表は、SYS スキーマに存在する必要があります。表を 作成するには、admin_tables プロシージャを使用します。
orphan_table_name (オプション)	破損データ・ブロック内の 1 行を参照する各索引エントリの情 報が移入される Orphan Key Table の名前。指定する表は、SYS スキーマに存在する必要があります。表を作成するには、 admin_tables プロシージャを使用します。
key_count	処理する索引エントリの数。

rebuild_freelists

指定されたオブジェクトの空きリストを再構築します。すべての空きブロックは、マスター
空きリストに入れられます。他のすべての空きリストはゼロになります。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散されます。

```
procedure rebuild_freelists(
  schema_name IN varchar2,
  object_name IN varchar2,
  partition_name IN varchar2 DEFAULT NULL,
```

```
object_type IN binary_integer DEFAULT TABLE_OBJECT);
```

表 19-6 rebuild_freelists プロシージャ

引数	説明
schema_name	スキーマ名。
object_name	空きリストを再構築するオブジェクトの名前。
partition_name (オプション)	空きリストを再構築するパーティションまたはサブパーティションの名前。パーティション・オブジェクトに対し、partition_name を指定していなければ、すべてのパーティションとサブパーティションが処理されます。パーティション・オブジェクトで、指定したパーティションにサブパーティションが含まれている場合は、すべてのサブパーティションが処理されます。
object_type (オプション)	処理するオブジェクトの型。TABLE_OBJECT または INDEX_OBJECT でなければなりません。デフォルトは TABLE_OBJECT です。

skip_corrupt_blocks

指定されたオブジェクトの索引および表走査中に、破損ブロックのスキップを可能にするか、または禁止します。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、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。

```
procedure skip_corrupt_blocks(
    schema_name IN varchar2,
    object_name IN varchar2,
    partition_name IN varchar2 DEFAULT NULL,
    object_type IN binary_integer DEFAULT TABLE_OBJECT,
    flags IN boolean DEFAULT SKIP_FLAG);
```

表 19-7 skip_corrupt_blocks プロシージャ

引数	説明
schema_name	処理するオブジェクトのスキーマ名。
object_name	オブジェクト名。
partition_name (オプション)	処理するパーティションまたはサブパーティションの名前。パーティション・オブジェクトに対し、partition_name を指定していなければ、すべてのパーティションとサブパーティションが処理されます。パーティション・オブジェクトで、指定したパーティションにサブパーティションが含まれている場合は、すべてのサブパーティションが処理されます。
object_type (オプション)	処理するオブジェクトの型。TABLE_OBJECT または CLUSTER_OBJECT でなければなりません。デフォルトは TABLE_OBJECT です。
flags (オプション)	SKIP_FLAG を指定すると、索引と表の走査中にオブジェクトのソフトウェア破損ブロックのスキップがオンになります。NOSKIP_FLAG を指定した場合は、走査中にソフトウェア破損ブロックが検出されると ORA-1578 が戻されます。

admin_tables

Repair Table および Orphan Key Table の管理機能を提供します。

```
procedure admin_tables(  
    table_name IN varchar2,  
    table_type IN binary_integer,  
    action IN binary_integer,  
    tablespace IN varchar2 DEFAULT NULL);
```

表 19-8 admin_tables プロシージャ

引数	説明
table_name	処理する表の名前。指定した table_type に基づいてデフォルトで ORPHAN_KEY_TABLE または REPAIR_TABLE に設定されます。この引数を指定する場合、表名には適切な接頭辞 ORPHAN_ または REPAIR_ が必要です。
table_type	表のタイプ。ORPHAN_TABLE または REPAIR_TABLE でなければなりません。
action	実行する管理アクションを示します。CREATE_ACTION、PURGE_ACTION または DROP_ACTION でなければなりません。表がすでに存在している場合に CREATE_ACTION を指定すると、エラーが戻されます。 PURGE_ACTION は、存在しないオブジェクトに対応付けられている表内のすべての行を削除することを示します。表が存在しない場合に DROP_ACTION を指定すると、エラーが戻されます。 CREATE_ACTION と DROP_ACTION を指定すると、対応付けられているビュー DBA_<table_name> がそれぞれ作成され、削除されます。このビューは、存在しないオブジェクトに対応付けられた行が排除されるように定義されます。 SYS スキーマ内で作成されます。
tablespace (オプション)	表の作成時に使用する表領域を示します。デフォルトでは、SYS のデフォルト表領域が使用されます。表領域を指定した場合に、アクションが CREATE_ACTION でなければ、エラーが戻されます。

DBMS_REPAIR の例外

- 942 Repair Table が存在しません。
- 1418 指定された索引が存在しません。
- 24120 無効なパラメータ。
- 24121 CASCADE_FLAG とブロック範囲は指定できません。
- 24122 無効なブロック範囲。
- 24124 無効なアクション・パラメータが指定されています。

- 24126 CASCADE_FLAG が指定されましたが、オブジェクトは表ではありません。
- 24127 表領域が指定されましたが、アクションは CREATE_ACTION ではありません。
- 24128 非パーティション・オブジェクトに対してパーティションが指定されています。
- 24129 無効な Orphan Key Table 名。ORPHAN_ 接頭辞が付いていません。
- 24129 指定された Repair Table には REPAIR_ 接頭辞が付いていません。
- 24131 Repair Table の列が正しくありません。
- 24132 Repair Table 名が長すぎます。

スキーマ・オブジェクトの一般的な管理

この章では、第 11 ～ 19 章で扱わなかった、スキーマ・オブジェクトの一般的な管理について説明します。この章のトピックは、次のとおりです。

- 一度の操作で複数の表やビューを作成
- スキーマ・オブジェクトの改名
- 表、索引およびクラスタの分析
- 表とクラスタの削除
- トリガーの使用可能および使用禁止
- 整合性制約の管理
- オブジェクトの依存性の管理
- オブジェクト名の名前の変換の管理
- データ・ディクショナリの記憶領域パラメータの変更
- スキーマ・オブジェクトについての情報を表示

一度の操作で複数の表やビューを作成

スキーマ・オブジェクトを作成するには、それに伴う操作に必要な権限が必要です。たとえば、CREATE SCHEMA コマンドを使用して複数の表を作成するには、表を作成するために必要な権限が必要です。

SQL 文 CREATE SCHEMA を使用して、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。一度の操作で複数の表やビューを確実に作成して権限を付与するには、CREATE SCHEMA 文が便利です。個々の表やビューの作成が失敗したり、権限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。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、ANSI の CREATE TABLE コマンドと CREATE VIEW コマンドを拡張した Oracle 独自の機能をサポートしていません。これには、STORAGE 句が含まれません。

スキーマ・オブジェクトの改名

オブジェクトを改名するには、そのオブジェクトを所有する必要があります。スキーマ・オブジェクトは、次のどちらかの方法で改名できます。

- オブジェクトを削除して作成しなおします。
- SQL 文 RENAME を使用してオブジェクトを改名します。

オブジェクトを削除して作成しなおす場合、そのオブジェクトの権限付与はすべて失われます。オブジェクトを作成しなおすときに、権限を再付与してください。一方、RENAME 文を使用して、表、ビュー、順序、またはそのプライベート・シノニムを改名できます。RENAME 文を使用すると、そのオブジェクトの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 ビューを改名します。

```
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改名できません。そのようなオブジェクトを改名するには、削除してから作成しなおします。

スキーマ・オブジェクトを改名する前に、次のような影響を理解する必要があります。

- 改名されたオブジェクトに依存しているすべてのビューと PL/SQL プログラム・ユニットは、無効になる次の使用の前に再コンパイルする必要があります。
- 改名されたオブジェクトのすべてのシノニムは、使用されるとエラーを戻します。

関連項目： Oracle によるオブジェクト依存性の管理の詳細は、20-22 ページの「[オブジェクトの依存性の管理](#)」を参照してください。

表、索引およびクラスタの分析

ここでは、表、索引およびクラスタを分析する方法について説明します。トピックは次のとおりです。

- [表、索引およびクラスタの統計の使用](#)
- [表、索引およびクラスタの妥当性の検査](#)
- [表とクラスタの連鎖された行のリスト](#)

表、索引またはクラスタを分析して、それについてのデータを収集したり、その記憶形式の妥当性を検証したりできます。表、クラスタまたは索引を分析するには、その表、クラスタまたは索引を所有しているか、ANALYZE ANY システム権限が必要です。

これらのスキーマ・オブジェクトを分析して、特定のオブジェクトについての統計の収集や更新もできます。DML 文が発行されると、参照されるオブジェクトの統計を使用して、その文の最も効率的な実行計画が決定されます。この最適化を「コストベースの最適化」と呼びます。統計はデータ・ディクショナリ内に格納されます。

表、索引またはクラスタを分析して、オブジェクトの構造の妥当性を検査できます。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。索引の妥当性検査をするとき、索引の中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ことを確かめることができます。スキーマ・オブジェクトが破損した場合、そのオブジェクトを削除して再作成できます。

表やクラスタの連鎖された行に関する情報を収集するために、その表やクラスタを分析できます。この結果は、行の更新のための領域が十分であるかどうかを判断する上で役に立ちます。たとえば、この情報によって、PCTFREE が表やクラスタに対して適切に設定されているかどうかを知ることができます。

関連項目：パフォーマンスの統計表示とオプティマイザの表、索引およびクラスタの詳細は、『Oracle8i チューニング』を参照してください。

索引構成表の分析の詳細は、[第 14 章「表の管理」](#)を参照してください。

表、索引およびクラスタの統計の使用

STATISTICS オプションを指定した SQL 文 ANALYZE を使用して、表、索引またはクラスタの物理記憶領域特性の統計を収集し、データ・ディクショナリに格納できます。コストベースの最適化によって、分析したオブジェクトにアクセスする SQL 文の最も効率的な実行計画を選択するときに、Oracle はこのような統計を使用できます。また、このコマンドによって生成される統計を使用して、分析されたオブジェクトにアクセスする効率的な SQL 文を書くこともできます。

COMPUTE STATISTICS オプションまたは ESTIMATE STATISTICS オプションを指定した ANALYZE 文を使用して、統計の計算や見積りができます。

COMPUTE STATISTICS	統計を計算するとき、オブジェクト全体が、オブジェクトに関するデータを収集するために走査されます。Oracle では、オブジェクトに関する厳密な統計値を計算するために、このデータを使用します。計算された統計では、オブジェクトにわたってわずかな変動がみられます。計算された統計の情報を収集するために、オブジェクト全体が走査されるので、オブジェクトのサイズが大きいほど、必要な情報を収集するために必要な作業が多くなります。
ESTIMATE STATISTICS	統計を見積るとき、Oracle はオブジェクトの一部から代表的な情報を収集します。この情報のサブセットにより、オブジェクトに関して合理的に統計を見積ることができます。統計を見積る際の正確さは、Oracle がサンプリングの代表値をどのように使用するかに依存します。見積る統計の情報の収集ではオブジェクトの一部だけが走査されるので、オブジェクトを高速に分析できます。Oracle が見積る上で使用するべき行の数や割合は、オプションで指定できます。

注意： 表やクラスタの統計を計算するとき、その計算に必要な一時領域の容量は指定された行の数に関連します。COMPUTE STATISTICS の場合、表全体を格納し、ソートするために十分な一時領域に、行ごとに多少のオーバーヘッドを加えた領域が必要となります。また、ESTIMATE STATISTICS の場合は、要求されたサンプル行を格納し、ソートするために十分な一時領域と行ごとの多少のオーバーヘッドを加えた領域が必要となります。索引の場合は、分析用に一時領域は必要ありません。

関連項目：SQL 文 ANALYZE の詳細は、『Oracle8i SQL リファレンス』を参照してください。
統計を含むデータ・ディクショナリ・ビューの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

オブジェクトの統計の表示

オブジェクトの統計が計算されたもの、または見積られたものでも、統計はデータ・ディクショナリ内に格納されます。次のデータ・ディクショナリ・ビューを使用して、統計を問い合わせることができます。

- USER_INDEXES、ALL_INDEXES、DBA_INDEXES
- USER_TABLES、ALL_TABLES、DBA_TABLES
- USER_TAB_COLUMNS、ALL_TAB_COLUMNS、DBA_TAB_COLUMNS

注意： これらのビューの中の行では、統計が収集された索引、表およびクラスタについてのみ、統計表示列にエントリが入ります。オブジェクトを ANALYZE するたびにそのオブジェクトに対するエントリが更新されます。

表の統計

表に関して、次に示す統計を収集できます。

注意： * 記号は、統計の計算時に数値が必ず正確な値であることを示しています。

- 行数
- 使用されたブロック数 *

- 使用されていないブロック数
- 利用可能な平均空き領域
- 連鎖行の数
- 平均行長
- 列あたりの重複しない値の数
- 列あたりの 2 番目に小さい値 *
- 列あたりの 2 番目に大きい値 *

注意： 表が分析されるときに、表に対応付けられたすべての索引の統計が自動的に収集されます。

索引統計 索引に関して、次に示す統計を収集できます。

- 索引レベル *
- リーフ・ブロックの数
- 重複しないキーの数
- 平均のリーフ・ブロック / キーの数
- 平均のデータ・ブロック / キーの数
- クラスタ化の係数

クラスタ統計 クラスタに関して収集可能な唯一の統計は平均のクラスタ・キーの連鎖長であり、この統計は見積りまたは計算ができます。クラスタ内の表とクラスタ化した表に対応付けた（クラスタ・キー索引を含む）すべての索引に対する統計は、クラスタが統計に対して分析されるときに自動的に収集されます。

注意： ANALYZE 文が発行されるとき、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、新しい統計がデータ・ディクショナリ内の古い統計に置き換わります。

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、1064 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;
```

どちらの場合も、50 より大きな百分率や、そのオブジェクトの中の行または索引値の数の 50% を超える行または索引値の数を指定すると、Oracle は見積りではなく、正確な統計を計算します。

スキーマ・オブジェクトの統計の削除

表、索引またはクラスタの統計は、DELETE STATISTICS オプションを指定した ANALYZE コマンドを使用して、データ・ディクショナリから削除できます。たとえば、オブジェクトに関する文に対して、コストベースの最適化を使わない場合、そのオブジェクトの統計も削除できます。次の文は、EMP 表の統計をデータ・ディクショナリから削除します。

```
ANALYZE TABLE emp DELETE STATISTICS;
```

共有 SQL と統計分析

表、クラスタまたは索引を分析すると、現行の（共有プール内にある）共有 SQL 文に影響が及びます。統計を更新、削除するためにオブジェクトが分析されるときは、文の次の実行が新しい統計を利用できるように、分析されたオブジェクトを参照するすべての共有 SQL 文はメモリーからフラッシュされます。

次のプロシージャをコールできます。

DBMS_UTILITY.-ANALYZE_SCHEMA()

このプロシージャには 2 つの引数、つまりスキーマの名前と分析方法 (COMPUTE、ESTIMATE または DELETE) が必要です。そのスキーマ内の全オブジェクトに関する統計が収集されます。

DBMS_DDL.-ANALYZE_OBJECTS()

このプロシージャには、4 つの引数、つまりオブジェクトの型 (CLUSTER、TABLE または INDEX)、およびオブジェクトのスキーマ、オブジェクトの名前、分析方法 (COMPUTE、ESTIMATE または DELETE) が必要です。そのオブジェクトに関する統計が収集されます。

これらのプロシージャを定期的にコールして、統計を更新するとよいでしょう。

表、索引およびクラスタの妥当性の検査

表、索引、クラスタまたはスナップショットの構造の整合性を確認するためには、`VALIDATE STRUCTURE` オプションを指定した `ANALYZE` コマンドを使用します。構造が妥当である場合、エラーは戻されません。しかし、構造が破損しているとエラー・メッセージが戻されます。表、索引またはクラスタが破損した場合には、それを削除して作成しなおしてください。スナップショットが破損した場合、完全リフレッシュを実行し、それで問題が解決したことを確認してください。問題が解決していなければ、スナップショットを削除し、作成しなおします。

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、および関連するすべてのオブジェクトの妥当性を検査できます。次の文は、EMP 表と対応付けられているすべての索引の妥当性を検査します。

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

表とクラスタの連鎖された行のリスト

表またはクラスタの連鎖された行と移行された行は、`LIST CHAINED ROWS` オプションを指定した `ANALYZE` コマンドを使用して検出できます。このコマンドの結果は、`LIST CHAINED ROWS` オプションによって戻される情報を受け取るものとして明示的に作成して指定した表に格納されます。

`ANALYZE...LIST CHAINED ROWS` 文によって戻されるデータを受け取るための適切な表を作成するには、Oracle とともに提供される `UTLCHAIN.SQL` スクリプトを使用してください。 `UTLCHAIN.SQL` スクリプトは、このスクリプトを実行するユーザーのスキーマ内に表 `CHAINED_ROWS` を作成します。

`CHAINED_ROWS` 表が作成されたら、`ANALYZE` 文を使用するときにその表を指定できます。たとえば、次の文は `CHAINED_ROWS` 表に `EMP_DEPT` クラスタ内の連鎖された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： `UTLCHAIN.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。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。

表とクラスタ内の連鎖行と移行行の数を減らす方法の詳細は、『Oracle8i チューニング』を参照してください。

表とクラスタの削除

表（またはクラスタ）は存在したままで、完全に空になるように、表のすべての行、またはクラスタ化した表のグループのすべての行を削除できます。たとえば、毎月のデータを含む表があり、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、3つのオプションがあります。

1. DELETE 文を使用します。

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;
```

2. DROP コマンドと CREATE 文を使用します。

表を削除してから表を作成しなおします。たとえば、次の文は EMP 表を削除した後で作成しなおします。

```
DROP TABLE emp;  
CREATE TABLE emp ( . . . );
```

3. TRUNCATE を使用します。

SQL 文 TRUNCATE 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 TRUNCATE します。

```
TRUNCATE TABLE emp;
```

DELETE の使用方法

DELETE コマンドを使用するときに、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かなりのシステム・リソースが消費されます。たとえば、その表と対応付けられた索引の CPU 時間、REDO ログ領域およびロールバック・セグメント領域はリソースを必要とします。また、各行が削除されるときに、トリガーが起動される可能性があります。その結果空になる表またはクラスタに前もって割り当てられた領域は、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 の使用方法

表やクラスタを削除してから作成しなおすと、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、削除される表またはクラスタ化した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、削除される表またはクラスタ化した表に対する権限付与もすべて削除されます。

TRUNCATE の使用方法

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しないで、すぐにコミットします。つまり、それは DDL 文であり、ロールバックできません。TRUNCATE 文は、TRUNCATE される表に対応付けられている構造（制約およびトリガー）または許可に影響を及ぼしません。また、TRUNCATE 文は、TRUNCATE の後で、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します。

ユーザーの対応するスキーマ内の表またはクラスタを TRUNCATE できます。また、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 TRUNCATE できます。

親キーを含む表またはクラスタ化表を TRUNCATE する前に、別の表の中の参照しているすべての外部キーを使用禁止にしてください。自己参照制約を使用禁止にする必要はありません。

TRUNCATE 文が表から行を削除する場合、表に対応付けたトリガーは起動されません。また、TRUNCATE 文は、監査が使用可能であっても、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、単一の監査レコードが、発行された TRUNCATE 文に対して生成されます。

ハッシュ・クラスタは TRUNCATE では削除できません。また、ハッシュ・クラスタまたは索引クラスタ内の表を個々に TRUNCATE できません。索引クラスタを TRUNCATE すると、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。個々のクラスタ化した表からすべての行を削除する必要がある場合には、DELETE コマンドを使用するか、または表を削除してから作成しなおしてください。

TRUNCATE コマンドの REUSE STORAGE オプション、または DROP STORAGE オプションは、TRUNCATE の後に、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、結果表に割り当てられているエクステントの数を少なくして、MINEXTENTS の元の設定にします。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 オプションは、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を、それらに割り当てたままにすることを指定します。たとえば、次の文は EMP_DEPT クラスタを TRUNCATE し、クラスタに対して前から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 オプションまたは DROP STORAGE オプションは、対応付けられたどの索引にも適用されます。表またはクラスタが TRUNCATE されると、対応付けられた索引もすべて TRUNCATE されます。また、TRUNCATE された表またはクラスタ、対応付けられた索引に対する記憶領域パラメータは、TRUNCATE の結果として変化しません。

関連項目：監査の詳細は、[第 25 章「データベース使用の監査」](#)を参照してください。

トリガーの使用可能および使用禁止

データベース・トリガーは、データベースに格納され、表に行を追加するなど、ユーザーが特定の変更を加えるときにアクティブ化される（「起動される」）プロシージャです。トリガーを使用すると、高度にカスタマイズされたデータベース管理システムを提供する Oracle の標準機能を補足できます。たとえば、通常の営業時間中に発行された文しか許可されないように、表に対する DML 操作を制限するトリガーを作成できます。

データベース・トリガーは、対応付けられている表に対して次のいずれかの文が発行されると、暗黙的に実行されます。

- INSERT
- UPDATE
- DELETE
- STARTUP
- SHUTDOWN
- LOGON

トリガーには、次の 2 つの異なるモードがあります。

使用可能	トリガー文を発行し、トリガー制限が TRUE と評価された場合、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。トリガーを作成した当初は、デフォルトによって使用可能に設定されます。
使用禁止	トリガー文を発行し、トリガー制限が 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、表の 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 システム権限が必要です。

関連項目：トリガーの詳細は、『Oracle8i 概要』を参照してください。

トリガーの作成方法の詳細は、『Oracle8i SQL リファレンス』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 表の REORDER という名前の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;
```

ALTER TABLE 文の ENABLE ALL TRIGGERS オプションを使用すれば、特定の表に定義してあ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 1 つが真である場合には、一時的にトリガーを使用禁止にできます。

- トリガーの参照するオブジェクトが使用可能になっていないこと。
- 大規模なデータ・ロードを実行する必要がある、トリガーを実行しないでデータ・ロードを迅速に処理すること。
- トリガーが適用される表にデータをロードしていること。

トリガーを使用禁止にするには ALTER TRIGGER 文の DISABLE オプションを使用します。たとえば、INVENTORY 表のトリガー REORDER を使用禁止にするには、次の文を入力します。

```
ALTER TRIGGER reorder DISABLE;
```

ALTER TABLE 文の DISABLE ALL TRIGGERS オプションを使用すれば、表に関連するトリガーをすべて同時に使用禁止にすることができます。たとえば、INVENTORY 表の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、データベース内のデータに関するルールまたは文です。使用可能になっている制約は、データベース内で入力または更新されたときにデータをチェックし、制約のルールに従わないデータが入力されないようにします。妥当性検査済みの制約では、一意性、マスター / ディテール関係、式への準拠または NULL が存在しないことが保証されます。

それらのルールや文は、制約が使用可能になっていて検証済みのときは常に真です。ただし、制約を使用禁止にする（または「使用可能であることが確認できない」と、整合性制約に違反したデータがデータベース内に存在する可能性があるため、この文が真であるかどうか分かりません。ここでは、整合性制約の管理のメカニズムと手順について説明します。

- [整合性制約の状態](#)
- [制約チェックの延期](#)

- 対応付けられた索引をもつ制約の管理
- 定義するときの整合性制約の設定
- 既存の整合性制約を変更する方法
- 整合性制約を削除する方法
- 制約例外のレポート

関連項目 : 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。20-20 ページの「[制約例外のレポート](#)」を参照してください。

整合性制約の一般情報は、『Oracle8i 概要』を参照してください。

整合性制約の状態

表で定義する整合性制約は、次の 4 つの状態の 1 つとすることができます。

妥当性検査なしで 使用禁止	<p>制約が妥当性検査なしで使用禁止の場合、制約により定義したルールは、制約で指定した列のデータ値に適用されません。ただし、制約の定義は、データ・ディクショナリ内に格納されます。</p> <p>このモードは、データ・ウェアハウス・ロールアップまたはロードを実行していて、ロード処理のスピードを速くする場合に役立ちます。</p>
妥当性検査なしで 使用可能	<p>妥当性検査なしで使用可能な制約がある表には、無効なデータが入っていることがあります。無効なデータの新たな追加はできません。</p> <p>妥当性検査ありの使用可能を使用して表の中のデータの妥当性検査を行う前に中間的な状態として有効です。このため、新しいデータが制約に違反することはできず、妥当性検査なしの使用可能から妥当性検査ありの使用可能までの制約をとるときにロックは保持されません。</p> <p>このモードは、例外に関してチェックする制約を使用可能にしない場合に役立ちます（たとえばデータ・ウェアハウス・ロードの後など）。</p>
使用可能で妥当性 検査済み	<p>使用可能な制約が適用され、有効（表のデータの有効性がチェックされる）として認識されます。制約の定義は、データ・ディクショナリ内に格納されます。</p> <p>これは、制約処理の標準的な運用状態です。このモードは、正規の OLTP 処理中の無効なデータ入力を防ぐのに使用します。</p>

使用禁止で妥当性
検査済み

索引なしで一意の制約を与えることができます。この状態になっている表は更新できません。

EXCHANGE PARTITION 文を使用してパーティション化されていないデータをパーティション表にロードすることを可能にします。また、データ・ウェアハウジング用の表があり、領域の使用量を最小限度に抑える必要がある場合にも使用します。

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適用するには、その制約を常に使用可能にしてください。しかし、状況によっては、次の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が望ましい場合があります。

- 表に大量のデータをロードする場合
- 表に対して大規模な変更を加えるバッチ操作を実施する場合（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）
- 一度に 1 つの表をインポートまたはエクスポートする場合

上記の 3 つの場合には、整合性制約を一時的に使用禁止にして操作のパフォーマンスを改善できます（特にデータ・ウェアハウス構成において）。

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、上記に示した操作を終了した後は制約を使用可能にする必要があります。

妥当性検査なしで制約を使用可能にする方法

制約が妥当性検査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、制約に適合するかどうかチェックされます。ただし、表の中の既存のデータはチェックされません。適用された制約がある表には、無効なデータが入っていることがあります。無効なデータの新たな追加はできません。妥当性検査なしの状態では制約を使用可能にすることは、有効な 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便利です。

制約を使用可能にする場合、妥当性検査は不要です。妥当性検査なしで制約を使用可能にするほうが、妥当検査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を妥当性検査する場合、妥当性検査中の DML ロックは不要です（すでに使用禁止にした制約を妥当性検査する場合とは違う）。施行によって、妥当性検査中の違反が持ち込まれないことが保証されます。したがって、妥当性検査なしで使用可能にすれば、制約を使用可能にすることに一般に関連するダウンタイムを少なくすることができます。

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には、制約に違反する行は表に挿入されません。しかし、制約が使用禁止になっていると、そのような行も挿入できます。その行は制約に対する例外と呼ばれます。制約が妥当性検査なしの使用可能状態にある場合、制約が使用禁止になっていた間に入力したデータから生ずる違反が残ります。制約に違反する行は、制約を検査済みの状態にするために更新または削除しなければなりません。

制約に違反するすべての行は、EXCEPTIONS 表で調べることができます。

関連項目 : EXCEPTIONS 表の詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

整合性制約の状態（プロシージャと利点）

整合性制約の状態を次の順序で使用すると、最高の利点が得られます。

1. 使用禁止状態。
2. 操作（ロード、エクスポート、インポート）を実行します。
3. 妥当性検査なしで使用可能にします。
4. 使用可能状態

制約をこの順序で使用する利点は、次のとおりです。

- ロックが保持されない
- すべての制約が同時に使用可能状態にすることができる
- 制約を使用可能にする操作がパラレルで行われる
- 表で同時のアクティビティが許される

制約チェックの延期

Oracle が制約をチェックして、制約が満たされていない場合はエラーのシグナルが出されます。トランザクションが終わるまで、制約の有効性のチェックを延期できます。

SET CONSTRAINTS 文を発行すると、SET CONSTRAINTS モードはトランザクションの期間中、または別の SET CONSTRAINTS 文によりモードが設定しなおされるまで続きます。

注意： SET CONSTRAINT 文は、トリガーの内部では発行できません。

関連項目 : SET CONSTRAINTS 文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

制約の一般情報は、『Oracle8i 概要』を参照してください。

制約チェックを延期する方法

適切なデータの選択 処理中のデータが次のような特性のものである場合、UNIQUE キーと FOREIGN キーの制約チェックを延期した方がよいかもしれません。

- 表がスナップショットである場合。
- 表に入っている大量のデータが別のアプリケーションによって操作されており、そのアプリケーションは同じ順序でデータを戻すかどうかわからない場合。
- FOREIGN キーでカスケード操作を更新する場合。

外部のアプリケーションで操作中の大量のデータを処理する場合、トランザクションの終わりでまで有効性に関する制約のチェックを延期できます。

制約の作成が延期可能かどうかを確認する 適切な表を識別し選択した後、表の FOREIGN、UNIQUE および PRIMARY キーの制約が延期可能として作成されるようにしてください。そのためには、次のような文を発行します。

```
CREATE TABLE dept (
    deptno NUMBER PRIMARY KEY,
    dname VARCHAR2 (30)
);
CREATE TABLE emp (
    empno NUMBER,
    ename VARCHAR2 (30),
    deptno NUMBER REFERENCES (dept),
    CONSTRAINT epk PRIMARY KEY (empno) DEFERRABLE,
    CONSTRAINT efk FOREIGN KEY (deptno)
        REFERENCES (dept. deptno) DEFERRABLE);
INSERT INTO dept VALUES (10, 'Accounting');
INSERT INTO dept VALUES (20, 'SALES');
INSERT INTO emp VALUES (1, 'Corleone', 10);
INSERT INTO emp VALUES (2, 'Costanza', 20);
COMMIT;

SET CONSTRAINT efk DEFERRED;
UPDATE dept SET deptno = deptno + 10
    WHERE deptno = 20;

SELECT * from emp ORDER BY deptno;
EMPNO    ENAME          DEPTNO
-----
1        Corleone         10
2        Costanza         20
UPDATE emp SET deptno = deptno + 10
    WHERE deptno = 20;
SELECT * FROM emp ORDER BY deptno;

EMPNO    ENAME          DEPTNO
```

```

-----
      1      Corleone      10
      2      Costanza     30
COMMIT;

```

すべての制約を延期に設定する データ操作に使用されるアプリケーション内では、実際にデータの処理を始める前にすべての制約を延期に設定する必要があります。すべての制約を延期可能に設定するには、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では、その制約が存在する限りそのまま残ります。ALTER SESSION SET CONSTRAINTS 文は、現行のセッションにしか適用されません。

コミットをチェックする（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って、制約違反をチェックできます。制約になにか問題がある場合、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・メッセージが表示されます。

対応付けられた索引をもつ制約の管理

UNIQUE キーまたは PRIMARY キーを作成するときに、Oracle はその制約を適用するのに既存の索引を使用できるかどうかを調べます。そのような索引がない場合、Oracle は索引を作成します。

Oracle が一意索引を使用して制約を適用しており、一意索引に対応付けられた制約が削除されるか、使用禁止になると、その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 キーか UNIQUE キーの制約または索引を削除したり使用禁止にすることはできません。

注意： 延期できる UNIQUE キーと PRIMARY キーはすべて、一意でない索引を使用する必要があります。

定義するとき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、制約の定義に次の句を入れることにより、その制約を使用可能、使用禁止、妥当性検査あり、または妥当性検査なしにすることができます。

- ENABLE
- DISABLE
- ENABLE [VALIDATE]
- DISABLE [NOVALIDATE]
- ENABLE NOVALIDATE
- DISABLE VALIDATE

制約定義にこれらの句のどれも指定しない場合、その制約は自動的に使用可能にされ、妥当性検査されます。

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、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、異常終了することはありません。制約の定義は、そのルールが適用されていないので許可されます。

関連項目 : 制約例外の詳細は、20-20 ページの「[制約例外のレポート](#)」を参照してください。

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、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、使用可能にしようとする ALTER TABLE 文は、表の行が整合性制約に違反している場合、異常終了する可能性があります。その場合、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されません。

対応する索引を作成する UNIQUE キーまたは PRIMARY KEY を使用可能にするには、表の所有者は索引を収録する表領域の割当て制限、または UNLIMITED TABLESPACE システム権限が必要です。

既存の整合性制約を変更する方法

ALTER TABLE 文を使用して制約を使用可能または使用禁止にしたり、変更できます。

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。

```
ALTER TABLE dept
    DISABLE CONSTRAINT dname_ukey;
ALTER TABLE dept
    DISABLE PRIMARY KEY,
    DISABLE UNIQUE (dname, loc);
```

次の文は、使用禁止にした整合性制約を妥当性検査なしで使用可能にします。

```
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、または検査します。

```
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 キー制約または PRIMARY KEY 制約、およびすべての依存する FOREIGN KEY 制約を一度で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てください。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;
```

整合性制約を削除する方法

整合性制約は、適用するルールが真でなくなった場合、またはその制約が必要でなくなった場合に削除できます。整合性制約は、ALTER TABLE 文に DROP 句を指定して削除します。次の 2 つの文は、整合性制約を削除します。

```
ALTER TABLE dept
    DROP UNIQUE (dname, loc);
ALTER TABLE emp
    DROP PRIMARY KEY,
    DROP CONSTRAINT dept_fkey;
```

UNIQUE キー制約と PRIMARY KEY 制約を削除すると、対応する一意索引が削除されます。また、FOREIGN KEY が UNIQUE キーまたは PRIMARY KEY を参照している場合、DROP 文に CASCADE CONSTRAINTS 句を指定しなければ、制約を削除できません。

制約例外のレポート

制約の妥当性検査時に例外が存在する場合は、エラーが戻され、整合性制約は妥当性検査なしの状態のままです。整合性制約の例外が存在しているため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には、制約に対するすべての例外を更新または削除するまで、制約の妥当性検査はできません。

CREATE TABLE 文を使用して、どの行が違反しているかを判断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ためには、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、すべての例外となる行の ROWID、表所有者、表名および制約名はある特定の表に設定されます。

注意： 制約を使用可能にする前に、ENABLE 句の EXCEPTIONS オプションから情報を受け取る、適切な例外レポート表を作成してください。例外の表は、スクリプト UTLEXCPT.SQL を送ることによって作成でき、それにより EXCEPTIONS という名前の表が作成されます。また、スクリプトを修正し、再実行することによって、新たに別の名前で例外表を作成できます。

次の文は DEPT 表の PRIMARY KEY を検査しようとします。例外が存在した場合は、EXCEPTIONS という名前の表に情報が挿入されます。

```
ALTER TABLE dept ENABLE PRIMARY KEY EXCEPTIONS INTO exceptions;
```

重複する主キー値が DEPT 表に存在し、DEPT の PRIMARY KEY 制約の名前が SYS_C00610 である場合には、前の文によって次の行が EXCEPTIONS 表の中に入れられることがあります。

```
SELECT * FROM exceptions;
```

ROWID	OWNER	TABLE_NAME	CONSTRAINT
AAAAZ9AABAAABvgAAB	SCOTT	DEPT	SYS_C00610
AAAAZ9AABAAABvgAAG	SCOTT	DEPT	SYS_C00610

問合せ情報により、例外レポート表とマスター表の行を結合し、次のように特定の制約に違反している実際の行を記述します。

```
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

制約が指定された表から、制約に違反しているすべての行を更新または削除してください。例外を更新する場合には、制約に違反する値を、制約または NULL と一致する値に変更してください。マスター表の行を更新または削除した後で、例外レポート表の例外に対応する行は、他の例外レポートとの混同を避けるために削除してください。また、マスター表と例外レポート表を更新する文は、必ず同一トランザクション上で指定し、トランザクション一貫性を保証してください。

前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;
```

例外を管理する上での最終的な目的は、例外レポート表の例外をすべて排除することにあります。

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場合は、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、例外の排除を行う前に制約を妥当性検査なしで使用可能にします。

関連項目： UTLEXCPT.SQL スクリプトの正確な名前と位置は、オペレーティング・システムによって異なります。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。

オブジェクトの依存性の管理

ここでは、様々なオブジェクトの依存性について説明します。トピックは次のとおりです。

- [ビューを手動で再コンパイルする方法](#)
- [プロシージャとファンクションの手動による再コンパイル](#)
- [パッケージを手動で再コンパイルする方法](#)

最初に、[表 20-1](#) を見てください。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内の変更によってどのような影響を受けるかを示したものです。

表 20-1 オブジェクトの状態に影響を与える操作

操作	結果としてのオブジェクトの状態	結果としての依存オブジェクトの状態
CREATE 表、順序、シノニム	VALID (エラーがない場合)	変化なし ¹
ALTER 表 (ADD 列 MODIFY 列) RENAME 表、順序、シノニム、ビュー	VALID (エラーがない場合)	INVALID
DROP 表、順序、シノニム、ビュー、プロシージャ、ファンクション、パッケージ	変化なし。オブジェクトは削除されます	INVALID
CREATE ビュー、プロシージャ ²	VALID (エラーがない場合); INVALID (構文エラー、認可エラーの場合)	変化なし ¹
CREATE OR REPLACE ビューまたはプロシージャ ²	VALID (エラーがない場合); INVALID (構文エラー、認可エラーの場合)	INVALID
REVOKE オブジェクト権限 ³ ON オブジェクト TO/FROM ユーザー	変化なし	INVALID (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) ³
REVOKE オブジェクトの権限 ³ ON オブジェクト TO/FROM PUBLIC	変化なし	INVALID (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) ³

表 20-1 オブジェクトの状態に影響を与える操作 (続き)

操作	結果としてのオブジェクトの状態	結果としての依存オブジェクトの状態
REVOKE システム権限 ⁴ TO/FROM ユーザー	変化なし	INVALID (すべてのユーザーのオブジェクト) ⁴
REVOKE システム権限 ⁴ TO/FROM PUBLIC	変化なし	INVALID (データベース内のオブジェクト) ⁴
¹ 以前にオブジェクトが存在しなかった場合は、依存オブジェクトを INVALID にすることがあります。 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 および EXECUTE などの DML オブジェクト権限のみ。妥当性の再検査での再コンパイルは必要ありません。 ⁴ SELECT、INSERT、UPDATE、DELETE ANY TABLE および EXECUTE ANY PROCEDURE などの DML システム権限のみ。妥当性の再検査での再コンパイルは必要ありません。		

Oracle は、無効なビューまたは PL/SQL プログラム・ユニットを、それが次に使用されるときに自動的に再コンパイルします。さらに、適切な SQL コマンドに COMPILE パラメータを指定して、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、依存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指定するのが最も一般的です。この場合、ビューまたはプログラム・ユニットが実行されない限り、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_/ALL_/DBA_OBJECTS を問い合わせてください。

ビューを手動で再コンパイルする方法

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内にあるか、または ALTER ANY TABLE システム権限が必要です。ビューを再コンパイルするには、COMPILE パラメータを指定した ALTER VIEW コマンドを使用します。次の文は、自分のスキーマ内の EMP_DEPT ビューを再コンパイルします。

```
ALTER VIEW emp_dept COMPILE;
```

プロシージャとファンクションの手動による再コンパイル

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内にあるか、または ALTER ANY PROCEDURE システム権限が必要です。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 パラ

メータを指定した ALTER PROCEDURE/FUNCTION 文を使用します。次の文は、自分のスキーマ内のストアド・プロシージャ UPDATE_SALARY を再コンパイルします。

```
ALTER PROCEDURE update_salary COMPILE;
```

パッケージを手動で再コンパイルする方法

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内にあるか、または ALTER ANY PROCEDURE システム権限が必要です。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 パラメータを指定した ALTER PACKAGE 文を使用します。次の文はそれぞれ、パッケージ ACCT_MGMT の本体だけ、および本体と仕様部の両方を再コンパイルします。

```
ALTER PACKAGE acct_mgmt COMPILE BODY;  
ALTER PACKAGE acct_mgmt COMPILE PACKAGE;
```

オブジェクト名の名前の変換の管理

ここでは、Oracle でオブジェクト名を分解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 では SCOTT が最初の断片です。1 つの断片だけが存在する場合、1 つの断片は、最初の断片と見なされます。
 - a. Oracle は、現行のスキーマで、オブジェクト名の最初の断片に一致するオブジェクトを検索します。そのようなオブジェクトが見つからない場合には、ステップ b を続けます。
 - b. 現行のスキーマでスキーマ・オブジェクトが見つからない場合、Oracle は、オブジェクト名の最初の断片に一致するパブリック・シノニムを検索します。そのようなシノニムが見つからない場合には、ステップ c を続けます。
 - c. パブリック・シノニムも見つからない場合、Oracle は、オブジェクト名の最初の断片に一致するスキーマを検索します。スキーマが見つかったら、ステップ b へ戻り、識別されたスキーマ内で見つけるオブジェクトとして名前の 2 番目の断片を使用します。2 番目の断片が前に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には、エラーが戻されます。
- ステップ c でスキーマが見つからない場合、そのオブジェクトは識別できず、エラーが戻されます。

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（ある場合）は、見付けられたオブジェクトの妥当な部分に一致する必要があります。たとえば、SCOTT.EMP.DEPTNO が名前、SCOTT がスキーマとして識別され、EMP が表として識別される場合には、（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。

明示的、またはシノニム内で間接的に分散データベースで、グローバル・オブジェクト名が使用されるとき、ローカル Oracle はその参照をローカルに分解します。たとえば、シノニムをリモート表のグローバル・オブジェクト名に分解します。部分的に分解された文はリモート・データベースに転送され、リモート Oracle が、ここで説明したオブジェクトの分解を完了します。

データ・ディクショナリの記憶領域パラメータの変更

ここでは、データ・ディクショナリの記憶領域パラメータについて説明します。この項の構成は、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領域の変更が必要なエラー](#)

データベースが非常に大規模であったり、著しい数のオブジェクト、表の列、制約定義、ユーザーまたはその他の定義を含んでいる場合、データ・ディクショナリを構成する表が、どこかの時点で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、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。このような状況を改善するために、ユーザーの作成したセグメントの記憶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領域パラメータを変更し、より多くのエクステントが割り当てられるようにできます。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。

警告： データ・ディクショナリのオブジェクトの記憶領域の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、データベース全体を作成しなおす必要があるおそれがあります。たとえば、データ・ディクショナリ表 USERS の PCTINCREASE を 0、NEXT を 2K に設定すると、その表はすぐにセグメントの最大エクステントに達し、データベース全体をエクスポートし、作成しなおし、インポートしなければ、ユーザーまたはロールをそれ以上作成できなくなります。

データ・ディクショナリの構造

次の表とクラスタは、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を含みます。

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（クラスタ化した表を含む）。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ツ。
UET\$	セグメントに割り当てられているエクステンツ。
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（クラスタ化した表を含む）。I_TAB1 による索引付き。
CLUS	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（複合索引内の各列の個々のエントリを含む）。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。
CONS	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\$	CONS 内の制約の定義。I_CDEF1、I_CDEF2 および I_CDEF3 による索引付き。

CCOLS	制約を定義されている列（複合キー内の各列の個々のエントリを含む）、I_CCOL1 による索引付き。
USERS	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。
TSQS	ユーザーに対する表領域の割当て制限（各ユーザーに定義されている各表領域の割当て制限ごとに 1 エントリを含みます）。
C_OBJ#	TABS、CLUS、ICOLS、INDS および COLS を含むクラスタ。I_OBJ# による索引付き。
C_TS#	FETS、TSS および FILES を含むクラスタ。I_TS# による索引付き。
C_USER#	USERS と TSQS を含むクラスタ。I_USER# による索引付き。
C_COBJ#	CDEFs と CCOLS 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、変更する可能性があるセグメントは次のとおりです。

C_TS#	データベース内の空き領域がかなり断片化している場合
C_OBJ#	表に多くの索引または多くの列がある場合
CONS, 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

クラスタ化した表については、その表の記憶設定ではなく、クラスタの記憶設定を変更しなければなりません。

データ・ディクショナリの記憶領域の変更が必要なエラー

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なければならないような新しいオブジェクトをユーザーが作成しようとして、Oracle がエクステントを割り当てることができないと、エラーが戻されます。エラー・メッセージ ORA-1653 「表領域 'name' 内でサイズ *num* のエクステントの割当てエラーが発生しました。」は、このような問題を示します。

このエラー・メッセージを受け取り、変更しようとしていたセグメント（表またはロールバック・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、その定義を含むオブジェクトの記憶領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て、Oracle が作成しなければならないキーの索引のために十分な領域が存在しているにもかかわらず、ORA-1547を受け取った場合、CONS または C_COBJ# を別のエクステントに割り当てることができるかどうかをチェックしてください。このために、DBA_SEGMENTS を問い合わせ、CONS または C_COBJ# の記憶領域パラメータを変更することを検討してください。

関連項目：詳細は、20-32 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・オブジェクトについての情報を表示

データ・ディクショナリには、このマニュアルで説明したスキーマ・オブジェクトに関する多くのビューがあります。次に、スキーマ・オブジェクトに係するビューをまとめます。

- ALL_OBJECTS、USER_OBJECTS、DBA_OBJECTS
- ALL_CATALOG、USER_CATALOG、DBA_CATALOG
- ALL_TABLES、USER_TABLES、DBA_TABLES
- ALL_TAB_COLUMNS、USER_TAB_COLUMNS、DBA_TAB_COLUMNS
- ALL_TAB_COMMENTS、USER_TAB_COMMENTS
- ALL_COL_COMMENTS、USER_COL_COMMENTS、DBA_COL_COMMENTS
- ALL_VIEWS、USER_VIEWS、DBA_VIEWS
- ALL_INDEXES、USER_INDEXES、DBA_INDEXES
- ALL_IND_COLUMNS、USER_IND_COLUMNS、DBA_IND_COLUMNS
- USER_CLUSTERS、DBA_CLUSTERS
- USER_CLU_COLUMNS、DBA_CLU_COLUMNS
- ALL_SEQUENCES、USER_SEQUENCES、DBA_SEQUENCES
- ALL_SYNONYMS、USER_SYNONYMS、DBA_SYNONYMS
- ALL_DEPENDENCIES、USER_DEPENDENCIES、DBA_DEPENDENCIES

データベースのセグメントに関する情報は、次のデータ・ディクショナリ・ビューに含まれています。

- USER_SEGMENTS
- DBA_SEGMENTS

データベースのエクステントに関する情報は、次のデータ・ディクショナリ・ビューに含まれています。

- USER_EXTENTS
- DBA_EXTENTS
- USER_FREE_SPACE
- DBA_FREE_SPACE

Oracle デイクショナリ記憶領域パッケージ

表 20-2 は、PL/SQL で一部の SQL 機能にアクセスできるようにするか、またはデータベースの機能を拡張するために、Oracle で提供されるパッケージについて説明します。

表 20-2 提供されるパッケージ：追加機能

プロシージャ	説明
dbms_space.unused_space	オブジェクト（表、索引またはクラスタ）の中の使われていない領域に関する情報を戻します。
dbms_space.free_blocks	オブジェクト（表、索引またはクラスタ）の中の空きブロックに関する情報を戻します。
dbms_session.free_unused_user_memory	大容量の（100KB より大きい）メモリーを必要とする操作を実行したあと、未使用のメモリーを再生させるためのプロシージャ。このプロシージャは、メモリーに余裕のない場合にのみ使用します。

次の例では、様々なスキーマ・オブジェクトを表示する方法を示します。

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によって所有されるオブジェクトのすべてをリストします。

```
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 1 つを使用して、名前、データ型、長さ、精度、位取りおよびデフォルト・データ値などの列情報を記述できます。たとえば、次の問合せは、EMP 表と DEPT 表のデフォルト列値のすべてをリストします。

```
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	

列はすべてユーザー指定デフォルトを持っていないことに注目してください。これらの列は、デフォルトとして NULL を想定します。

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。ALL/USER/DBA_DEPENDENCIES データ・ディクショナリ・ビューを使用してビューの依存性を明らかにすることができ、ALL/USER/DBA_SYNONYMS データ・ディクショナリ・ビューは、シノニムのベース・オブジェクトをリストするために使用できます。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 sys.dba_synonyms
WHERE owner = 'JWARD';
```

TABLE_OWNER	TABLE_NAME	SYNONYM_NAME
SCOTT	DEPT	DEPT
SCOTT	EMP	EMP

例 4: グローバル・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、各ロールバック・セグメントのサイズを戻します。

```
SELECT segment_name, tablespace_name, bytes, blocks, extents
FROM sys.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: グローバル・エクステント情報の表示

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たエクステントとそれらのエクステントのサイズがそれぞれ識別されます。

```
SELECT segment_name, bytes, blocks
FROM sys.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、50KB の 3 つの等しいサイズ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）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によって利用できる空き領域を示します。

```
SELECT tablespace_name, file_id, bytes, blocks
FROM sys.dba_free_space;
```

TABSPACE_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ができないセグメントの表示

また、DBA_FREE_SPACE を、ビュー DBA_SEGMENTS、DBA_TABLES、DBA_CLUSTERS、DBA_INDEXES、DBA_ROLLBACK_SEGS と組み合わせて使用して、他の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かどうかを判断できます。

セグメントは、次のどちらかの理由でエクステントを割り当てることができない可能性があります。

- 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。
- セグメントに、データ・ディクショナリに記録されている、最大数のエクステントが存在する (SEG.MAX_EXTENTS)。
- セグメントに、データ・ブロック・サイズによって許可される、最大数のエクステントが存在する。これは、オペレーティング・システム固有の値です。

注意： MAXEXTENTS の STORAGE 句の値を UNLIMITED にできますが、データ・ディクショナリ表は許容されるブロックの最大値より大きな 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なしのフォーマットに変換できません。

次の問合せは、上記の基準のどちらかに適合するすべてのセグメントの名前、所有者および表領域を戻します。

```
SELECT seg.owner, seg.segment_name,
       seg.segment_type, seg.tablespace_name,
       DECODE(seg.segment_type,
              'TABLE', t.next_extent,
              'CLUSTER', c.next_extent,
              'INDEX', i.next_extent,
              'ROLLBACK', r.next_extent)
FROM sys.dba_segments seg,
     sys.dba_tables t,
     sys.dba_clusters c,
     sys.dba_indexes i,
     sys.dba_rollback_segs r

WHERE ((seg.segment_type = 'TABLE'
       AND seg.segment_name = t.table_name
       AND seg.owner = t.owner
```

```

AND NOT EXISTS (SELECT tablespace_name
                  FROM dba_free_space free
                  WHERE free.tablespace_name = t.tablespace_name
                  AND free.bytes >= t.next_extent))
OR (seg.segment_type = 'CLUSTER'
    AND seg.segment_name = c.cluster_name
    AND seg.owner = c.owner
    AND NOT EXISTS (SELECT tablespace_name
                        FROM dba_free_space free
                        WHERE free.tablespace_name = c.tablespace_name
                        AND free.bytes >= c.next_extent))
OR (seg.segment_type = 'INDEX'
    AND seg.segment_name = i.index_name
    AND seg.owner = i.owner
    AND NOT EXISTS (SELECT tablespace_name
                        FROM dba_free_space free
                        WHERE free.tablespace_name = i.tablespace_name
                        AND free.bytes >= i.next_extent))
OR (seg.segment_type = 'ROLLBACK'
    AND seg.segment_name = r.segment_name
    AND seg.owner = r.owner
    AND NOT EXISTS (SELECT tablespace_name
                        FROM dba_free_space free
                        WHERE free.tablespace_name = r.tablespace_name
                        AND free.bytes >= r.next_extent)))
OR seg.extents = seg.max_extents OR seg.extents = data_block_size;

```

注意： この問合せを使用するときには、data_block_size をシステムのデータ・ブロック・サイズで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すれば、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。

- 表領域が満杯であれば、表領域にデータ・ファイルを追加する。
- セグメントに多くのエクステントが存在し、セグメントの MAXEXTENTS を大きくできない場合は、次の手順を実行する。最初に、セグメント内のデータをエクスポートします。次に、多くのエクステントを割り当てる必要がないように INITIAL の設定を大きくして、セグメントを削除して作成しなおします。最後に、データをセグメントにインポートします。

ロールバック・セグメントの管理

この章では、ロールバック・セグメントを管理する方法について説明します。この項のトピックは、次のとおりです。

- [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・セグメントの記憶領域パラメータを指定](#)
- [ロールバック・セグメントのオンライン化とオフライン化](#)
- [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る方法](#)
- [ロールバック・セグメントの削除](#)
- [ロールバック・セグメント情報を監視](#)

関連項目 : Oracle Parallel Server オプションを使用する場合は、『Oracle8i Parallel Server 概要および管理』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

ここでは、データベースのロールバック・セグメントを作成したり管理したりする前に考慮すべきガイドラインについて説明します。この項のトピックは、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・サイズの設定
- 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
- ロールバック・セグメントに対する記憶位置の設定

すべてのデータベースには、1 つ以上のロールバック・セグメントが含まれています。ロールバック・セグメントとは、トランザクションがロールバックされるイベントでのトランザクションのアクションが記録されるデータベースの部分です。ロールバック・セグメントを使用して、読み込みの一貫性を実現し、トランザクションのロールバック、データベースの回復ができます。

関連項目：ロールバック・セグメントの詳細は、『Oracle8i 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、多くのセグメントにわたるロールバック・セグメントの競合が分散され、システム・パフォーマンスが改善されます。複数のロールバック・セグメントは、次の状況で必要とされます。

- 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 表領域に作成される。SYSTEM 表領域以外にオブジェクトを作成できますが、その表領域 (SYSTEM 以外のオブジェクト用) に少なくとも 1 つの追加のロールバック・セグメントを作成してオンラインにするまでは、それらに書き込むことはできません。
- 多くのトランザクションが同時に処理されているときには、より多くのロールバック情報が同時に生成される。インスタンスに対して用意する同時実行のトランザクション数はパラメータ TRANSACTIONS で指定し、各ロールバック・セグメントが処理すべきトランザクション数はパラメータ TRANSACTIONS_PER_ROLLBACK_SEGMENT で指定します。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 $\text{TRANSACTIONS} / \text{TRANSACTIONS_PER_ROLLBACK_SEGMENT}$ で算出される数のロールバック・セグメントを取得して、最大量のトランザクションを処理しようとします。したがって、パラメータを設定した後で、TRANSACTIONS / TRANSACTIONS_PER_ROLLBACK_SEGMENT で算出される数のロールバック・セグメントを作成してください。

関連項目：Oracle Parallel Server 環境では、インスタンスを起動するためには、各インスタンスは SYSTEM ロールバック・セグメントの他にそれ自身のロールバック・セグメントにアクセスする必要があります。詳細は、『Oracle8i Parallel Server 概要および管理』を参照してください。

SYSTEM 表領域へのロールバック・セグメントの追加

インスタンスは、必要な他のロールバック・セグメントに加えて、常に SYSTEM ロールバック・セグメントを取得します。ただし、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 SYSTEM ロールバック・セグメントを使用して、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ようとします。SYSTEM 以外のロールバック・セグメントに対するトランザクションが多すぎると、Oracle は SYSTEM セグメントを使用します。

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、インスタンスがデータベースをオープンするときに、インスタンスによって明示的に取得されます。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えるロールバック・セグメントのプールを形成します。

データベースに Parallel Server オプションがない場合には、パブリック・ロールバック・セグメントとプライベート・ロールバック・セグメントは同一です。したがって、すべてをパブリック・ロールバック・セグメントとして作成できます。また、セグメント数が十分で、データベースをオープンする各インスタンスが、SYSTEM ロールバック・セグメントに加えて、ロールバック・セグメントを少なくとも 1 つ取得できる限り、Parallel Server オプション付きのデータベースでも、パブリック・セグメントだけにできます。Oracle Parallel Server を使用している場合には、プライベート・ロールバック・セグメントも使用できません。

関連項目：Parallel Server オプションとロールバック・セグメントの詳細は、『Oracle8i Parallel Server 概要および管理』を参照してください。

パブリック・ロールバック・セグメントとプライベート・ロールバック・セグメントの詳細は、『Oracle8i 概要』を参照してください。

自動的に取得するロールバック・セグメントの指定

インスタンスが起動すると、デフォルトではインスタンスは TRANSACTIONS/TRANSACTIONS_PER_ROLLBACK_SEGMENT で算出される数のロールバック・セグメントを取得します。特定のサイズまたは特定の表領域を持つ特定のロールバック・セグメントをインスタンスが取得するようにする場合は、インスタンスのパラメータ・ファイル内のパラメータ ROLLBACK_SEGMENTS に、そのロールバック・セグメントの名前を指定してください。

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、インスタンスはこの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・セグメントは、プライベートでもパブリックでもかまいません。

ロールバック・セグメント・サイズの設定

ロールバック・セグメントの全体のサイズは、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べきです。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ときにパフォーマンスが向上します。多くの場合、ロールバック・セグメントはどのようなサイズのトランザクションであっても処理できます。ただし、トランザクションが極端に短かったり、長かったりするような極端な場合には、適切なサイズのロールバック・セグメントも使用してください。

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、主メモリ内に常時キャッシュされるように小さくすべきです。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 SGA 内にキャッシュされる可能性が大きくなり、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、長い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る」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すと、読み込み一貫性を実現するのに必要なロールバック・エントリが上書きされるために発生します。アプリケーションのトランザクションを設計する際にはこのことを考慮して、トランザクションを短い最小作業単位にして、この問題が発生しないようにしてください。

対照的に、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・セグメントの事前に割り当てられたエクステンツに適合できるので、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・セグメントでは効率よく作動します。

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、トランザクションが、トランザクション / ロールバック・セグメント・サイズを基準にしたロールバック・セグメントに明示的に割り当てられた場合に、パフォーマンスを最適化できます。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ツ割当てと切捨てを最小限に抑えることができます。これは、多くのシステムに対して必要なわけではなく、極端に大きい、または小さいトランザクションだけを対象にします。

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、各トランザクションのタイプ（小型、中型、大型など）に対するロールバック・セグメントを適切なサイズにしてください。ロールバック・セグメントのほとんどは、一般トランザクションに対応し、例外トランザクション用のロールバック・セグメントの数は少ないはずです。このように少ない方のロールバック・セグメントに OPTIMAL を設定することにより、セグメント・サイズは必要に応じて目的のレベルに戻ります。

ロールバック・セグメントは、トランザクションの種類によって使い分ける必要があることをユーザーに徹底させてください。多くの場合、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利点はありません。ただし、不規則なトランザクションに対して作られた適切なロールバック・セグメントに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合がそれほど多くなければ、ほとんどの SQL 文が表の 10% 以下にしか影響を及ぼさないため、それぞれのロールバック・セグメントは、データベースの最大の表のサイズの 10% にする必要があります。したがって、このサイズのロールバック・セグメントは、多くの SQL 文によって実行されるアクションを格納するのに十分な大きさになるはずです。

一般的には、ロールバック・セグメントに対して、大きな MAXEXTENTS を設定するとよいでしょう。これによって、ロールバック・セグメントは、必要なだけ、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、インスタンスに対する各ロールバック・セグメントに、等しいサイズのエクステントが 10 から 20 ある場合に、ロールバック I/O パフォーマンスは最適になります。

初期ロールバック・セグメントの合計のサイズとセグメントに対する初期エクステントの数を決定した後、ロールバック・セグメントの各エクステントのサイズを算出するために、次の公式を使用してください。

$$T / n = s$$

ここで

T = 初期ロールバック・セグメントの合計のサイズ (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 = 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)

s が計算された後、ロールバック・セグメントを作成し、記憶領域パラメータ INITIAL と NEXT を s 、MINEXTENTS を n に指定してください。PCTINCREASE はロールバック・セグメントに対しては指定できないので、0 にデフォルト設定されます。また、エクステントのサイズ s がデータ・ブロック・サイズの倍数でない場合、次の倍数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

それぞれロールバック・セグメントの OPTIMAL パラメータを設定するときには、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 が頻繁に、エクステントの縮小と割当てを実行する必要がないように、OPTIMAL を大きくする必要があります。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る」という問題を避けるために OPTIMAL を大きくする必要があります。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、OPTIMAL をより小さく設定して、メモリー内にキャッシュされるようにロールバック・セグメントを十分小さくしてください。これにより、システムのパフォーマンスが向上します。

ロールバック・セグメントに対する OPTIMAL の値を推測し、その有効性を監視するには、MONITOR ROLLBACK 文を使用します。ロールバック・セグメントごとに、次の統計が表示されます。

Size, High Water	ロールバック・セグメントに割り当てられた領域の最大値 (単位はバイト)
Size, Optimal	ロールバック・セグメントの OPTIMAL サイズ (単位はバイト)
Occurrences, Wraps	トランザクションがロールバック・セグメント内のあるエクステントから別の既存エクステントに書き込みを継続した累積回数
Occurrences, Extends	新しいエクステントがロールバック・セグメントに割り当てられた累積回数
Shrinks	Oracle がロールバック・セグメントからエクステントを切り捨てた累積回数
Average Size, Shrunk	Oracle がロールバック・セグメントから切り捨てた領域の平均サイズ (単位はバイト)
Average Size, Active	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 (時間計測)

インスタンスが、同等のサイズのエクステントを持つ同じサイズのロールバック・セグメントを持っている場合、ロールバック・セグメントの OPTIMAL パラメータは、Average Sizes, Active より少し大きく設定する必要があります。表 21-1 は、このモニターに表示される統計の解釈に関する追加情報を示しています。

表 21-1 現在の OPTIMAL 設定の有効性を分析する

Shrinks	Average Sizes, Shrunk	分析と推奨事項
Low	Low	<i>Average Sizes, active</i> が <i>Sizes, Optimal</i> に近い場合、OPTIMAL の設定は正しいです。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 は理想的に設定されています。
High	Low	OPTIMAL が小さすぎます。縮小が必要以上に多く実行されています。
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。 <i>Shrinks</i> 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。

ロールバック・セグメントに対する記憶位置の設定

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表領域を 1 つ特別に作成してください。このようにすれば、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って、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、オフラインにできます。
- エクステンツ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・セグメントがある表領域では、使用可能エクステンツ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。データベース用に追加のロールバック・セグメントを作成するには、SQL 文 CREATE ROLLBACK SEGMENT を使用します。新しいロールバック・セグメントが入る表領域はオンラインにしてください。

次の文は、USERS 表領域のデフォルトの記憶領域パラメータを使用して、USERS 表領域内にパブリック・ロールバック・セグメント USERS_RS を作成します。

```
CREATE PUBLIC ROLLBACK SEGMENT users_rs TABLESPACE users;
```

新しいロールバック・セグメントをオンラインにする方法

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、そのロールバック・セグメントの名前をデータベースのパラメータ・ファイル内の ROLLBACK_SEGMENTS パラメータに追加してください。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 RS1 と RS2 が作成される場合、パラメータ・ファイルの ROLLBACK_SEGMENTS パラメータは、次のようになります。

```
ROLLBACK_SEGMENTS= (RS1, RS2)
```

関連項目：作られたロールバック・セグメントは、オンラインになるまで、どのインスタンスのトランザクションにも使えません。詳細は、21-10 ページの「[ロールバック・セグメントのオンライン化とオフライン化](#)」を参照してください。

ロールバック・セグメントの記憶領域パラメータを指定

ここでは、ロールバック・セグメントの記憶領域パラメータの設定について説明します。この項のトピックは、次のとおりです。

- [ロールバック・セグメントを作成するときに記憶領域パラメータを設定する方法](#)
- [ロールバック・セグメントの記憶領域パラメータを変更する方法](#)
- [ロールバック・セグメントのフォーマットを変更する方法](#)
- [ロールバック・セグメントを手動で縮小する方法](#)

ロールバック・セグメントを作成するときに記憶領域パラメータを設定する方法

次のような記憶領域パラメータと最適サイズを持つパブリック・ロールバック・セグメント DATA1_RS を作成する場合を想定します。

- ロールバック・セグメントには、50KB の初期エクステントが割り当てられる場合。
- ロールバック・セグメントには、50KB の第 2 エクステントが割り当てられる場合。
- ロールバック・セグメントの最適サイズは 750KB である場合。
- 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は 15 である場合。

- 初期エクステントを含めて、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は 100 である場合。

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 PUBLIC ROLLBACK SEGMENT datal_rs
    TABLESPACE users
    STORAGE (
        INITIAL 50K
        NEXT 50K
        OPTIMAL 750K
        MINEXTENTS 15
        MAXEXTENTS 100);
```

ロールバック・セグメントの記憶領域パラメータを変更する方法

ロールバック・セグメントの記憶領域パラメータは、作成後に変更することもできます。ただし、ロールバック・セグメントに対して現在割り当てられているエクステントのサイズは変更できません。つまり、将来のエクステントだけに影響します。

SQL 文 ALTER ROLLBACK SEGMENT を使用して、ロールバック・セグメントの記憶領域パラメータを変更します。

次の文を使用して、DATA1_RS ロールバック・セグメントが割り当てることのできるエクステントの最大数を変更します。

```
ALTER PUBLIC ROLLBACK SEGMENT datal_rs
    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、OPTIMAL パラメータを含む、SYSTEM ロールバック・セグメントの設定を変更できます。

関連項目：ロールバック・セグメントのサイズと記憶領域パラメータ（OPTIMAL を含む）の設定に関するガイドラインについては、21-2 ページの「[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](#)」を参照してください。

ロールバック・セグメントのフォーマットを変更する方法

ロールバック・セグメントを変更するには、ALTER ROLLBACK SEGMENT システム権限が必要です。

ロールバック・セグメントについては、制限付きフォーマットまたは制限なしフォーマットを定義できます。制限付きまたは制限なしのフォーマットに変換するときは、そのロールバック・セグメントをオフラインにしてください。ロールバック・セグメントのフォーマットが制限なしの場合は、そのセグメントのエクステントには最小限 4 つのデータ・ブロックが必要です。このため、データ・ブロックが 4 つ未満のエクステントがある場合には、制限付きフォーマットのロールバック・セグメントを制限なしフォーマットに変換できません。制限付きフォーマットから制限なしフォーマットに変換し、エクステント内のデータ・ブロック数を 4 未満にする場合は、そのロールバック・セグメントを削除して作成しなおすしかありません。

ロールバック・セグメントを手動で縮小する方法

ロールバック・セグメントを縮小するには、ALTER ROLLBACK SEGMENT システム権限が必要です。

SQL 文 ALTER ROLLBACK SEGMENT 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にしてください。

次の文は、ロールバック・セグメント RBS1 を 100KB まで縮小します。

```
ALTER ROLLBACK SEGMENT rbs1 SHRINK TO 100K;
```

関連項目：ALTER ROLLBACK SEGMENT コマンドの詳細は、『Oracle8i SQL リファレンス』を参照してください。

ロールバック・セグメントのオンライン化とオフライン化

ここでは、ロールバック・セグメントをオンラインおよびオフラインにする方法について説明します。この項のトピックは、次のとおりです。

- ロールバック・セグメントをオンラインにする方法
- ロールバック・セグメントをオフラインにする方法

ロールバック・セグメントは、オンラインでトランザクションに利用できるか、またはオフラインでトランザクションに利用できないかのどちらかの状態です。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、トランザクションによる使用に対して利用可能な状態です。

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にすると効果的です。

- 表領域をオフラインする必要があるが、その表領域にロールバック・セグメントが入っている場合。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・セグメントを、表領域が含んでいる場合には、その表領域をオフラインにできません。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にする前にロールバック・セグメントをオフラインにします。

- トランザクションによって、現在ロールバック・セグメントが使用されていると、そのロールバック・セグメントを削除できません。ロールバック・セグメントが使用されないように、削除する前にロールバック・セグメントをオフラインにします。

注意： SYSTEM ロールバック・セグメントはオフラインにできません。

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にして、トランザクションで使用できます。作成直後、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・セグメントは、インスタンスのトランザクションで使用する前に、明示的にオンラインにしてください。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介して、オフラインのロールバック・セグメントをオンラインにできます。

ロールバック・セグメントをオンラインにする方法

現在の状態（DBA_ROLLBACK_SEGS データ・ディクショナリ・ビューに示される）が OFFLINE または PARTLY AVAILABLE であるロールバック・セグメントだけをオンラインにできます。オフライン・ロールバック・セグメントをオンラインにするには、ONLINE オプションを指定した SQL 文 ALTER ROLLBACK SEGMENT を使用します。

PARTLY AVAILABLE ロールバック・セグメントをオンラインにする方法

PARTLY AVAILABLE 状態のロールバック・セグメントには、インダウトまたは回復した分散トランザクションのデータが入っています。その状態は、データ・ディクショナリ・ビュー DBA_ROLLBACK_SEGS では PARTLY AVAILABLE となっています。ロールバック・セグメントは、そのトランザクションが RECO によって自動的に、または DBA によって手動で解決されるまで、通常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 になっている場合もあります。そのような場合には、前述の方法で PARTLY AVAILABLE セグメントをオンラインにできます。

ロールバック・セグメントがインダウト・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、ロールバック・セグメントは大きくする必要があるかもしれません。

PARTLY AVAILABLE ロールバック・セグメントをオンラインにするかわりに、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簡単なこともあります。

ロールバック・セグメントを自動的にオンラインにする方法

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にする場合は、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。

ロールバック・セグメントをオンラインにする例

次の例は、ロールバック・セグメント USER_RS_2 をオンラインにします。

```
ALTER ROLLBACK SEGMENT user_rs_2 ONLINE;
```

オンラインにした後、データ・ディクショナリ・ビュー DBA_ROLLBACK_SEGS でのロールバック・セグメントの状態は ONLINE です。

関連項目 : ROLLBACK_SEGMENTS パラメータと DBA_ROLLBACK_SEGS パラメータの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

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、21-15 ページの「[ロールバック・セグメント情報を表示する方法](#)」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にする方法

オンライン・ロールバック・セグメントをオフラインにするには、OFFLINE オプションを指定した ALTER ROLLBACK SEGMENT コマンドを使用します。DBA_ROLLBACK_SEGS のロールバック・セグメントの状態は ONLINE でなければならず、そのロールバック・セグメントは現行インスタンスが取得してください。

たとえば、ロールバック・セグメント USER_RS_2 がオフラインになります。

```
ALTER ROLLBACK SEGMENT user_rs_2 OFFLINE;
```

アクティブ・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にしようとすると、Oracle はただちにそのセグメントをオフラインにし、その状態を OFFLINE に変更します。

対照的に、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トランザクションがすべて完了すると、オフラインにします。トランザクションが完了するまで、オフラインに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にできません。この間、ビュー DBA_ROLLBACK_SEGS 内のロールバック・セグメントの状態は、ONLINE になっていません。ただし、ビュー V\$ROLLSTAT 内のロールバック・セグメントの状態は、PENDING OFFLINE になっています。

ロールバック・セグメントをオフラインにしようとして、PENDING OFFLINE を引き起こ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せます。ロールバック・セグメントは、オンラインに戻されても、通常どおり機能します。

パブリックとプライベートのロールバック・セグメントをオフラインにする方法

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にすると、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。

関連項目：ロールバック・セグメントの状態の表示については、21-15 ページの「[ロールバック・セグメント情報を表示する方法](#)」を参照してください。

ビュー DBA_ROLLBACK_SEGS と V\$ROLLSTAT の詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

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る方法

トランザクションは、USE ROLLBACK SEGMENT 句を指定した SET TRANSACTION 文を使用して、特定のロールバック・セグメントに対して明示的に割り当てることができます。次のような目的で、トランザクションはロールバック・セグメントに明示的に割り当てられます。

- トランザクションによって生成されるロールバック情報の予想量を、割り当てられたロールバック・セグメントの現在のエクステントに合わせることができます。
- 付加的なエクステントをロールバック・セグメントに動的に割り当てる（そして切り捨てる）必要はありません。もしそうにすると、システム全体のパフォーマンスが低下します。

トランザクションをロールバック・セグメントに明示的に割り当てるには、ロールバック・セグメントが現行のインスタンスに対してオンラインにしてください。さらに、SET TRANSACTION USE ROLLBACK SEGMENT 文はトランザクションの最初の文にしてください。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、エラーが戻されます。

たとえば、大量（多くのトランザクションよりもさらに多く）の作業を含むトランザクションを開始する場合には、次の文で、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_rsl;
```

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限り、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

セグメントのエクステントがディスク上で頻繁に断片化した場合や、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。ロールバック・セグメントが、ONLINE、PARTLY AVAILABLE、NEEDS RECOVERY または INVALID になっている場合、そのロールバック・セグメントを削除できません。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。削除する前に、ロールバック・セグメントをオフラインにしてください。

ロールバック・セグメントを削除するには、DROP ROLLBACK SEGMENT システム権限が必要です。

ロールバック・セグメントがオフライン状態の場合は、SQL 文 DROP ROLLBACK SEGMENT を使用して削除できます。

次の文は、DATA1_RS ロールバック・セグメントを削除します。

```
DROP PUBLIC ROLLBACK SEGMENT data1_rs;
```

DROP ROLLBACK SEGMENT 文を使用する場合、削除するロールバック・セグメントのタイプ（パブリックまたはプライベート）を、PUBLIC キーワードの有無によって正しく指定してください。

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、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。このステップ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。ただし、次回、ロールバック・セグメントが作成されると、利用できるのであれば、そのロールバック・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た行を使用します。これによって、削除されたロールバック・セグメントの行が、DBA_ROLLBACK_SEGS ビューに示されることはありません。

関連項目： ビュー DBA_ROLLBACK_SEGS パラメータの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

ロールバック・セグメント情報を監視

対応する操作で MONITOR を使用方法の詳細は、21-5 ページの「[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](#)」を参照してください。

ロールバック・セグメント情報を表示する方法

DBA_ROLLBACK_SEGS データ・ディクショナリ・ビューは、データベースのロールバック・セグメントに関する情報を格納します。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 sys.dba_rollback_segs;
```

SEGMENT_NAME	TABLESPACE_NAME	STATUS
-----	-----	-----
SYSTEM	SYSTEM	ONLINE
PUBLIC_RS	SYSTEM	ONLINE
USERS_RS	USERS	ONLINE

さらに、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

すべてのロールバック・セグメントを表示する方法

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戻します。

```
SELECT segment_name, tablespace_name, bytes, blocks, extents
       FROM sys.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

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にする場合、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。オフラインにしようとしてから実際にオフラインになるまでの間、DBA_ROLLBACK_SEGS のその状態が ONLINE になっていても、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てください。

```
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 Parallel Server 構成の一部である場合、この問合せは現行インスタンスだけのロールバック・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。

遅延ロールバック・セグメントを表示する方法

次の問合せは、どのロールバック・セグメントがプライベートで、どのセグメントがパブリックかを表示します。なお、現行インスタンスに対して現在オンラインになっているロールバック・セグメントに関する情報のみが表示されることに注意してください。

```
SELECT segment_name, tablespace_name, owner
      FROM sys.dba_rollback_segs;
```

SEGMENT_NAME	TABLESPACE_NAME	OWNER
-----	-----	-----
SYSTEM	SYSTEM	SYS
PUBLIC_RS	SYSTEM	PUBLIC
USERS_RS	USERS	SYS

遅延ロールバック・セグメントをすべて表示する方法

次の問合せは、すべての遅延ロールバック・セグメント（表領域がオンラインに戻されるまでオフラインにされた表領域のロールバック・エントリを保持するために作られたロールバック・セグメント）を表示します。

```
SELECT segment_name, segment_type, tablespace_name
      FROM sys.dba_segments
 WHERE segment_type = 'DEFERRED ROLLBACK';
```

SEGMENT_NAME	SEGMENT_TYPE	TABLESPACE_NAME
-----	-----	-----
USERS_RS	DEFERRED ROLLBACK	USERS

第Ⅳ部

データベース・セキュリティ

セキュリティ方針の設定

この章では、セキュリティ方針を作成するためのガイドラインについて、次の分野のデータベース操作ごとに解説します。

- システム・セキュリティ方針
- データ・セキュリティ方針
- ユーザー・セキュリティ方針
- パスワード管理方針
- 監査方針

システム・セキュリティ方針

ここでは、システム・セキュリティ方針について説明します。この項のトピックは、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、多岐にわたるセキュリティ方針のメンテナンスに責任を負う 1 人以上の管理者、つまりセキュリティ管理者が必要です。小規模なデータベース・システムの場合、データベース管理者がセキュリティ管理も兼務できます。しかし、大規模なデータベース・システムの場合、特別な人物または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とよいでしょう。

システムのセキュリティ管理担当者の決定後、あらゆるデータベースに対してセキュリティ方針を設定してください。データベースのセキュリティ方針には、次の部分で説明する、いくつかの詳細方針を設定してください。

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にセキュリティを保守してください。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応じて、セキュリティ管理者は、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唯一のユーザーの場合があります。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可能性もあります。どちらにしても、信頼のおける担当者の一人に、データベース・ユーザーの管理に必要な強力な権限を付与しなければなりません。

ユーザー認証

Oracle では、ホスト・オペレーティング・システム、ネットワーク・サービスまたはデータベースを使用して、データベース・ユーザーを認証する（正しい人物であることを確認すること）ことができます。多くの場合、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の方が、次の理由により望ましいでしょう。

- ユーザー名やパスワードの指定なしに、ユーザーは Oracle に迅速かつ簡便に接続できます。
- ユーザー認証はオペレーティング・システム内で集中管理されます。Oracle では、オペレーティング・システムとデータベースが対応していれば、ユーザー・パスワードとユーザー名を格納または管理する必要はありません。
- 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・エントリは一致します。

通常、データベースによるユーザー認証は、ホスト・オペレーティング・システムでユーザー認証がサポートできない場合に使用されます。

関連項目：ネットワーク認証の詳細は、『Oracle8i 分散システム』を参照してください。

ユーザー認証の詳細は、23-10 ページの「[ユーザーの作成](#)」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

状況に応じて、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、次のようなセキュリティを考慮する必要があります。

- データベース管理者は、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・システム権限を必ず持つ必要があります。
- 通常のデータベース・ユーザーは、データベースに関連付けられた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・システム権限を持たないでください。
- 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定義域を修正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：Oracle データベースに関するオペレーティング・システムのセキュリティの問題点の詳細は、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。

データ・セキュリティ方針

データ・セキュリティでは、オブジェクト・レベルでの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制御するメカニズムを採用しています。ファイン・グレイン・アクセス・コントロールでは、よりきめ細かいレベルへのデータ・アクセスも制限できます。データ・セキュリティ方針によって、特定のスキーマ・オブジェクトへのアクセス権限を所有するユーザーと、そのオブジェクトに対して許可される、ユーザーごとの特定の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・セキュリティ方針では、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）を定義してください。

データ・セキュリティ方針は、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、任意のユーザーが任意のスキーマ・オブジェクトを作成できるようにする場合、またはユーザーにシステム内の他のユーザーのオブジェクトへのアクセス権限を付与する場合は、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、データベース管理者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たり、オブジェクトのアクセス権限をロールおよびユーザーへ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。

データ・セキュリティ全般はデータの機密性に基づいています。機密性の低い情報は、データ・セキュリティ方針を厳しく設定する必要はありません。しかし、機密性の高いデータの場合には、セキュリティ方針を慎重に設定し、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。

ユーザー・セキュリティ方針

ここでは、ユーザー・セキュリティ方針について説明します。この項のトピックは、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](#)
- [アプリケーションの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・セキュリティ事項を考えてみます。

- [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、データベース・アクセス・セキュリティを保守するために、パスワード・セキュリティ方針を設定しなければなりません。たとえば、各データベース・ユーザーは、一定の間隔およびパスワードが別のユーザーに漏れた場合にパスワードを変更する必要があります。このような場合に、ユーザーに強制的にパスワードを修正するように要求すれば、許可されていないデータベースへのアクセスの可能性を減らせます。

暗号化されたパスワードによる接続の保護

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 / サーバーまたはサーバー / サーバー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。

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 `ORA_ENCRYPT_LOGIN` 環境変数を `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。

パスワードはクライアントとサーバーの両方で使用可能になっても、ネットワーク内をそのままの形で送信されるのではなく、修正 DES（データ暗号化規格）アルゴリズムを使用して暗号化されます。

DBLINK_ENCRYPT_LOGIN パラメータは 2 つの Oracle Server を接続するとき使用されます（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、ORA_ENCRYPT_LOGIN 環境変数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、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。接続が失敗して監査が使用可能になっていた場合、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、監査ログに以前記録された失敗が接続にかわり、接続が維持されます。不正なユーザーが暗号化されていないパスワードを使用して、Oracle に接続を強制的に再試行させることがないように、適切な（パラメータの）値を 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ことを考慮する必要があります。たとえば、多数のユーザー名を扱うデータベースで、ユーザーが使用できる権限を管理するには、ロール（ユーザーまたは他のロールに付与する、関連する権限のグループ）を使用すると便利な場合があります。ただし、少数のユーザー名を扱うデータベースでは、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簡単です。

セキュリティ管理者は、多数のユーザーまたはアプリケーション、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。ロールは非常に複雑な環境における権限管理のタスクを簡単にします。

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の方針も定義する必要があります。多数のユーザーがアクセスする大規模なデータベースの場合、セキュリティ管理者は、ユーザー・グループのカテゴリの決定およびそのユーザー・グループに対するユーザー・ロールの作成、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・ロールの付与および、ユーザーへのユーザー・ロールの割当てができます。例外を考慮して、セキュリティ管理者は必ず個々のユーザーに対して明示的に付与しなければならない権限もあわ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。

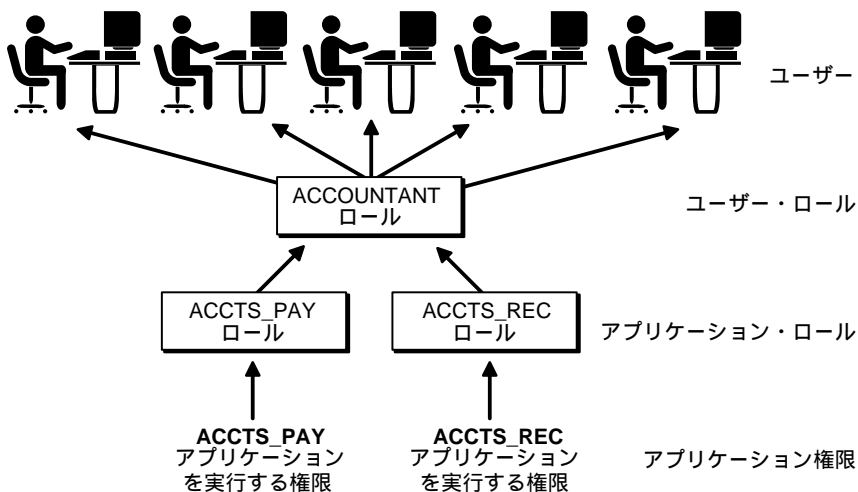
ある会社の経理部門のユーザー全員がデータベース・アプリケーション ACCTS_RECEIVABLE および ACCTS_PAYABLE を実行するための権限を必要としている状況を考えてみます。ロールは前述の両アプリケーションに対応付けられ、それらのアプリケーションを実行するために必要なオブジェクト権限を含んでいます。

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、データベース管理者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 および 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 22-1 に示します。

図 22-1 ユーザーのロール



このモデルは、次のような状況に対処します。

- その後に経理部門が新しいデータベース・アプリケーションに対してロールを必要とする場合には、そのアプリケーションのロールが ACCOUNTANT ロールに付与され、その結果、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付与されます。毎回、アプリケーションを使用する必要があ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、その経理部門で特定のアプリケーションがなくなった場合も、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る場合、新しい権限をそのアプリケーションの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定義域や ACCOUNTANT ロールを付与されたすべてのユーザーにその権限の修正が自動的に反映されます。
- このように、ユーザーに直接割り当てるロールが 1 つのみで済みます。

できれば、すべての状況でロールを使用して、エンド・ユーザーの権限管理を効率化し、かつ簡素化してください。

管理者のセキュリティ

セキュリティ管理者には、管理者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各種のデータベース管理者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するようにするとよいでしょう。これにより、管理ロールを適切な管理者ユーザーに付与できます。ただし、少量のデータベースでさらに管理者が数人しかいない場合、1 つの管理ロールを作成し、その権限を管理者全員に付与するのが便利です。

SYS や SYSTEM としての接続に対する保護

データベースの作成後、ただちに管理用の SYS ユーザー名と SYSTEM ユーザー名のパスワードを変更して、データベースに対する無許可のアクセスを防止してください。SYS および SYSTEM として接続すると、多数の方法でデータベースを変更する強力な権限がユーザーに与えられます。したがって、これらのユーザー名の権限は非常に機密性があり、データベース管理者を選定するときのみ使用してください。

関連項目：これらのアカウントに対応するパスワードの変更の詳細は、23-14 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、データベース管理者のみでなければなりません。SYSDBA として接続すると、データベースに関する処理（起動、停止および回復など）またはデータベース内のオブジェクトに関する処理（作成、削除など）を実行するための無制限な権限がユーザーに付与されます。データ・ディクショナリ表を変更できるのは、SYS 権限で接続しているユーザーのみです（SYSDBA/SYSOPER としての接続など）。

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な、強力なシステム権限やロールに制限を与える最も簡単な方法です。

大規模なインストール作業におけるデータベース管理者の責任を、何人かのデータベース管理者で分担し、各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シナリオを考えてみます。

- オブジェクトの作成と保守を担当する管理者

- データベースの調整とパフォーマンスを担当する管理者
- 新規ユーザーの作成と、データベース・ユーザーに対するロールや権限の付与を担当するセキュリティ管理者
- ルーチン・データベース操作（起動、停止、バックアップなど）を担当するデータベース管理者
- データベース回復などの緊急事態を担当する管理者
- 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データベース管理者

このシナリオでは、セキュリティ管理者は管理者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、それぞれのタイプの業務を遂行するために必要な権限をその中に明確に盛り込んでください（たとえば、DBA_OBJECTS、DBA_TUNE、DBA_SECURITY、DBA_MAINTAIN、DBA_RECOV、DBA_NEW）。
2. 各ロールには適切な権限を付与します。
3. 各タイプのデータベース管理者ごとに対応するロールを付与できます。

この計画では、次のような方法で今後問題が発生する可能性が抑えられます。

- データベース管理者のジョブの記述を変更して、さらに責任担当部分を付加する場合、そのデータベース管理者には新しい担当に対応する他の管理ロールを付与できます。
- データベース管理者のジョブの記述からその責任担当部分を減らすように変更する場合、そのデータベース管理者は適切な管理ロールを取り消すことができます。
- データ・ディクショナリは必ず各ロールと各ユーザーの情報を格納しているので、こうした情報によって各管理者のタスクを明らかにする目的で適用できます。

アプリケーションの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殊なセキュリティ方針を定義してください。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することがあります。そのかわりに、開発者からオブジェクト作成要求を受け取ったデータベース管理者にのみ、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、開発作業を遂行するために特殊な権限グループを要求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、CREATE PROCEDUREなどの各種システム権限を必要とします。ただし、データベースにおける機能全般を制限するために、開発者には唯一の特定システム権限を付与する必要があります。

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者は、データベースをテストすることのみに制限されており、本番データベースに関して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。

アプリケーションを開発し徹底的にテストをした時に、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、その本番データベースの適切なエンド・ユーザーに適用します。

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対して付与すべき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れば、このアプリケーションが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、プロシージャなどすべて、アプリケーション開発者の要求どおりに、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れば、データベース管理者がデータベースの領域の用途やデータベース内での情報のアクセス経路を完全に制御できます。

データベース・システムによっては、この2つのオプションのうち1つのみしか使用しないものもありますが、2つのオプションを混用できるシステムもあります。たとえば、アプリケーション開発者はストアド・プロシージャやパッケージの新規作成を許可して、表や索引の作成を許可しないようにもできます。セキュリティ管理者は、次の状況に応じてこの事項を決定します。

- データベースの領域の使用全般に必要な制御。
- スキーマ・オブジェクトに対するアクセス経路全般に必要な制御。
- 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望ましいです。

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、通常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 という名前の付いた通常のロールには、CREATE TABLE、CREATE VIEW および 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場合は、次の点に考慮してください。

- CREATE システム権限は、通常、アプリケーション開発者に権限付与されるので、開発者独自のオブジェクトを作成できます。ただし、CREATE ANY システム権限は、任意のユーザーのスキーマでのオブジェクトの作成を許可するためのものなので、通常、開発者には付与されません。この制限によって、その開発者のユーザー・アカウントのみが、新しいオブジェクトを作成できます。
- オブジェクト権限はアプリケーション開発者が使用するロールに付与されることはほとんどありません。ただし、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（主として、ビューおよびストアド・プロシージャ）を作成するときの有用性が制限されるので、これはあまり実用的ではありません。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せられる領域の制限

一般的に、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るのに対して、セキュリティ管理者は必ず各アプリケーション開発者で使用するデータベースの領域の限定とサイズの制限を保守する必要があります。たとえば、セキュリティ管理者として、個々のアプリケーション開発者ごと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
- 開発者にとってアクセス可能な各表領域としての制限割当て

関連項目：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。詳細は、23-14 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

大量のデータベース・アプリケーション（たとえば、プリコンパイラ、Forms アプリケーションなど）を使用する大規模データベース・システムにおいては、アプリケーション管理者が必要となります。アプリケーション管理者は次のタイプの業務に責任があります。

- 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 プロシージャの保守と更新

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、これらのジョブを必ずしも開発担当者に任せる必要はありません。データベース・アプリケーションに詳しい別の担当者を指名することもできます。

パスワード管理方針

データベース・セキュリティ・システムは、どんな時にもパスワードが機密になっていることに依存しています。それでもパスワードは、盗難、偽造および悪用などには弱い場合があります。データベース・セキュリティの制御をさらに強化するために、Oracle のパスワード管理方針が DBA によって制御されます。

ここでは、Oracle のパスワード管理の次の点を説明します。

- [アカウント・ロック](#)
- [パスワード・エイジングおよび時間切れ](#)
- [パスワード履歴](#)
- [パスワード複雑度の検証](#)

アカウント・ロック

特定のユーザーが、指定された回数以上の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可される回数を指定します。また、DBA はアカウントがロックされる時間の長さも指定します。

次の例では、ユーザー ASHWINI に許可されているログイン失敗の最大回数は 4 で、アカウントがロックされる時間の長さは 30 日です。アカウントは、30 日がすぎると自動的にロック解除されます。

```
CREATE PROFILE prof LIMIT
  FAILED_LOGIN_ATTEMPTS 4
  PASSWORD_LOCK_TIME 30;
ALTER USER ashwini PROFILE prof;
```

DBA がアカウントのロック解除に関して時間間隔を指定しないなら、ACCOUNT_LOCK_TIME はデフォルト値になります。DBA が ACCOUNT_LOCK_TIME を UNLIMITED と指定した場合、システム・セキュリティ管理担当者は明示的にそのアカウントのロックを解除する必要があります。したがって、アカウントがロックされている時間の長さは、ユーザーに割り当てられているリソース・プロファイルが DBA が構成する方法によって違います。

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カウントは、ゼロにリセットされます。

セキュリティ管理担当者は、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、アカウントは自動的にロック解除されません。セキュリティ管理担当者がアカウントのロックを解除することが必要です。

関連項目 : CREATE PROFILE 文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

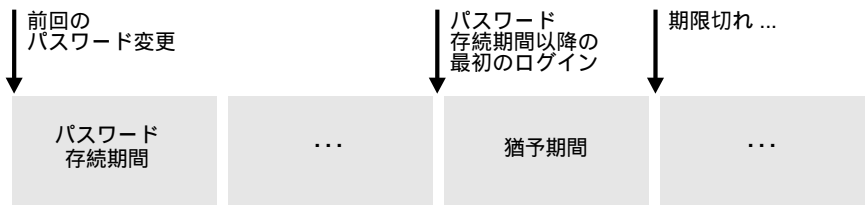
パスワード・エイジングおよび時間切れ

DBA は、CREATE PROFILE 文を使用してパスワードの最大存続期間を指定します。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。次の文の場合、ASHWINI は、同じパスワードを 90 日間使用でき、90 日間経過すると期限が切れます。

```
CREATE PROFILE prof LIMIT
  FAILED_LOGIN_ATTEMPTS 4
  PASSWORD_LOCK_TIME 30
  PASSWORD_LIFE_TIME 90;
ALTER USER ashwini PROFILE prof;
```

DBA は、CREATE PROFILE 文を使用して、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、データベース・アカウントに最初にログインしようとするとき、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しなければなりません。パスワードが猶予期間内に変更されなければ、アカウントの期限が切れ、パスワードが変更されるまでそのアカウントにはログインできません。

図 22-2 パスワードの存続期間と猶予期間の記録



たとえば、パスワードの存続期間が 60 日間で、猶予期間が 3 日間だとします。ユーザーが 60 日目（70 日目、100 日目などでもよく、要はそれがパスワードの存続期間以降の試行の最初のログインであること）以降の任意の日にログインを試行する場合、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そのユーザーに表示されます。ユーザーが猶予期間の 1 ～ 3 日の間にパスワードを変更しなければ、そのユーザーのアカウントの期限が切れます。次の文は、ユーザーがパスワードを期限切れ 3 日以内に変更する必要があることを指定しています。

```
CREATE PROFILE prof LIMIT
  FAILED_LOGIN_ATTEMPTS 4
  ACCOUNT_LOCK_TIME 30
  PASSWORD_GRACE_TIME 3;
ALTER USER ashwini PROFILE prof;
```

セキュリティ管理担当者は、明示的にアカウントを期限切れにすることもできます。これは、特に新しいアカウントの場合に役立ちます。

関連項目 : CREATE PROFILE 文の詳細は、『Oracle8iSQL リファレンス』を参照してください。

パスワード履歴

DBA は、CREATE PROFILE 文を使用して、ユーザーがパスワードを再使用できない時間間隔を指定します。

次の文では、DBA はユーザーが現在の自分のパスワードを 60 日間再使用できないことを指定しています。

```
CREATE PROFILE prof LIMIT
  PASSWORD_REUSE_TIME 60
  PASSWORD_REUSE_MAX UNLIMITED;
```

次の文は、ユーザーが現在のパスワードを再使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、次のことを調べます。

- パスワードの長さが 4 以上であること。
- パスワードがユーザー ID と同じではないこと。
- 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、1 つの句読点が含まれていること。
- パスワードが、welcome、account、database または user などの簡単な単語ではないこと。

- 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 文ではパスワード検証機能を十分サポートしないので、その文でパスワードを変更しないようにお勧めします。かわりに、パスワード変更には OCIPasswordChange() を使用してください。

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

DBA は、パスワードの既存の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できます。

DBA が作成した PL/SQL コールは、次のフォーマットにしてください。

```
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に、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る必要があります。

```
CREATE/ALTER PROFILE profile_name LIMIT  
PASSWORD_VERIFY_FUNCTION routine_name
```

パスワード検証ルーチンは、SYS が所有してください。

パスワード検証ルーチン：サンプル・スクリプト 次のサンプル・スクリプトは、デフォルトのパスワード・リソースの制限を設定し、パスワードの複雑度について最低限のチェックを行います。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場合は、このサンプル・スクリプトをモデルとして使用できます。

このスクリプトは、デフォルトのパスワード・リソース・パラメータを設定しますが、パスワード機能を使用可能にするときには必ず実行してください。ただし、必要に応じてデフォルトのリソース・パラメータを変更できます。

デフォルトのパスワード複雑度チェック機能は、次の最小複雑度チェックを実行します。

- このパスワードが最小の長さ要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。必要に応じて、この機能を変更できます。

この機能は SYS スキーマ内で作成しますが、スクリプトを実行する前に *sys/<password>* を *sysdba* として接続する必要があります。

```
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プロシージャ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、データベース監査の使用禁止を決定してもかまいません。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レベルを詳細に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、疑いのある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。

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。この章のトピックは、次のとおりです。

- [セッションとユーザーのライセンス](#)
- [ユーザー認証](#)
- [Oracle ユーザー](#)
- [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リスト](#)

関連項目 : ユーザーおよびプロファイルのセキュリティ方針の設定に関するガイドラインは、[第 22 章「セキュリティ方針の設定」](#)を参照してください。

権限およびロールは、ユーザーのデータベースに対するアクセスやデータベース中のスキーマ・オブジェクトに対するアクセスを制御します。権限とロールの詳細は、[第 24 章「ユーザー権限とロールの管理」](#)を参照してください。

セッションとユーザーのライセンス

ここでは、セッションおよびユーザー・ライセンスについて説明します。この項のトピックは、次のとおりです。

- [同時使用ライセンス](#)
- [接続権限](#)
- [セッションの最大数の設定](#)
- [セッション警告制限の設定](#)
- [データベースの稼働中の同時使用制限の変更](#)
- [名前付きユーザー制限](#)
- [ライセンス制限と現行値の参照](#)

Oracle は、サイトが Oracle Server のライセンス契約に従っていることを保証する手助けをします。サイトが同時使用のライセンスを受けている場合、データベースに同時に接続するセッション数を追跡し、制限できます。サイトが名前付きユーザー別にライセンスを受けている場合、データベース内に作成できる名前付きユーザー数を制限できます。どちらの場合も、ライセンス機能を制御し、その機能を使用可能にし、適切な制限を設定してください。

ライセンス機能を使用するには、サイトのライセンス契約のタイプ、およびセッションまたは名前付きユーザーの最大数を確認する必要があります。サイトはどちらかのタイプのライセンス（同時使用または名前付きユーザー）を使用し、両方を使用することはありません。

注意： 場合によっては、同時使用ライセンスまたは名前付きユーザー・ライセンスではなく、無制限のライセンスを保持しているサイトがあります。この場合に限り、ライセンス・メカニズムを使用禁止にしておきます。つまり、パラメータ・ファイル内の `LICENSE_MAX_SESSIONS`、`LICENSE_SESSIONS_WARNING` および `LICENSE_MAX_USERS` を指定しないか、またはこの 3 つのパラメータをすべて 0 に設定します。

同時使用ライセンス

同時使用ライセンスは、指定したコンピュータ上のデータベースへ同時に接続できるセッションを制限します。インスタンスを起動する前に、同時実行のセッション数に対して制限を設定できます。実際には、初期インストール手順の一部として、この制限を設定しておいてください。また、データベースの稼働中に同時実行セッションの最大数も変更できます。

関連項目： 初期インストール手順の詳細は、[第 2 章「Oracle データベースの作成」](#)を参照してください。

接続権限

インスタンスのセッションの制限に達すると、その後データベースに接続できるのは、RESTRICTED SESSION 権限を持つユーザー（通常、DBA）のみです。RESTRICTED SESSION 権限を持つユーザーがデータベースに接続すると、最大制限数に達していることを示すメッセージがこのユーザーに送信され、このメッセージが ALERT ファイルに書き込まれます。最大数に到達しているときには、不要なプロセスを停止する場合だけ、接続してください。Oracle のライセンス契約をアップグレードしていない限り、ライセンス制限を大きくしないでください。

最大同時実行セッションの設定に加えて、同時実行セッションの数に警告制限を設定できます。この制限に到達すると、追加のユーザーは引き続き接続できますが（最大制限まで）、Oracle は各接続で ALERT ファイルに適切なメッセージを書き込み、RESTRICTED SESSION 権限を持っている接続中の各ユーザーに、最大値に近づいていることを示す警告を送信します。

ユーザーが管理者権限で接続している場合も、この制限は適用されます。ただし、Oracle が制限を適用するのは、ユーザーが実行する最初の文の後になります。

同時使用制限の適用に加えて、Oracle はインスタンスごとに同時実行セッションの最大数を追跡記録します。この「高水位標」を使用できます。

関連項目：セッションの終了方法は、4-15 ページの「[セッションの停止](#)」を参照してください。

Oracle ライセンス制限のアップグレードの詳細は、23-6 ページの「[ライセンス制限と現行値の参照](#)」を参照してください。

Parallel Server の同時使用制限

Parallel Server で実行しているインスタンスの場合、各インスタンスには、固有の同時使用制限と警告制限を設定できます。ただし、それらのインスタンスの制限の合計は、サイトの同時使用ライセンスを超えてはなりません。

警告： 多重化ソフトウェアやハードウェア（TP モニターなど）を通じて Oracle に接続するセッションは、それぞれ個別に同時使用制限に寄与します。ただし、Oracle のライセンス・メカニズムは、この方法で接続されるセッション数を区別できません。サイトが多重化ソフトウェアやハードウェアを使用する場合、そのことを考慮し、多重化されたセッションを計算に入れるために、最大同時使用制限をより小さく設定してください。

関連項目：パラレル・サーバー環境で制限を設定および変更する方法の詳細は、『Oracle8i Parallel Server 概要および管理』を参照してください。

セッションの最大数の設定

インスタンスに対して同時実行セッションの最大数を設定するには、パラメータ `LICENSE_MAX_SESSIONS` を次のように設定してください。

```
LICENSE_MAX_SESSIONS = 80
```

この制限を設定する場合、警告制限の設定（`LICENSE_SESSIONS_WARNING`）が必要とされるわけではありません。ただし、警告制限を使用すると、サイトが最大使用に近づいていることが事前に通知されるため、最大制限の管理が容易になります。

セッション警告制限の設定

インスタンスに警告制限を設定するには、インスタンスの起動に使用するパラメータ・ファイルにパラメータ `LICENSE_SESSIONS_WARNING` を設定します。

セッション警告には、現行の同時実行最大制限（`LICENSE_MAX_SESSIONS`）よりも小さい値を設定してください。

データベースの稼働中の同時使用制限の変更

データベースの稼働中に同時使用制限の最大数または警告制限を変更するには、適切なオプションを指定した `ALTER SYSTEM` コマンドを使用します。次の文は、同時実行セッションの最大制限数を 100 に変更します。

```
ALTER SYSTEM SET LICENSE_MAX_SESSIONS = 100;
```

次の文は、警告制限と最大制限数の両方を変更します。

```
ALTER SYSTEM  
  SET LICENSE_MAX_SESSIONS = 64  
  LICENSE_SESSIONS_WARNING = 54;
```

どちらかの制限を現行のセッション数よりも小さい値に変更すると、現行セッションはそのままです。ただし、新しい設定は、インスタンスが停止されるまで、その後のすべての接続に適用されます。制限を永久的に変更するには、パラメータ・ファイル内の適切なパラメータの値を変更してください。

データベースの稼働中に同時使用制限を変更するには、`ALTER SYSTEM` 権限が必要です。また、最大制限に到達しているインスタンスに接続するには、`RESTRICTED SESSION` 権限が必要です。

警告： Oracle Server ライセンスを適切にアップグレードしていない限り、同時使用制限を大きくしないでください。詳細は、オラクル社カスタマ・サポートまで連絡してください。

名前付きユーザー制限

名前付きユーザー・ライセンスでは、指定したコンピュータ上で Oracle を使用することが許可されるユーザー数を制限します。このライセンスを適用するため、インスタンス起動前に、データベースに作成するユーザー数に対して制限を設定できます。また、インスタンスの稼働中に最大ユーザー数を変更したり、制限を使用禁止にしたりできます。この制限に達すると、それ以上ユーザーを作成することはできません。制限を超えてユーザーを作成しようとすると、すでに最大数のユーザーが作成されているというエラーが戻され、このメッセージが ALERT ファイルに書き込まれます。

このメカニズムは、データベースにアクセスしているユーザーがそれぞれ一意のユーザー名を持っており、共有されているユーザー名がないことを前提として機能します。複数のユーザーが同一ユーザー名を使用してデータベースにアクセスしないようにしてください。

関連項目 : Parallel Server で稼働しているインスタンスの場合、同一のデータベースに接続しているインスタンスは、すべて同一の名前付きユーザー制限が必要です。詳細は、『Oracle8i Parallel Server 概要および管理』を参照してください。

ユーザー制限の設定

データベースに対して作成されるユーザー数を制限するには、そのデータベースのパラメータ・ファイル内の LICENSE_MAX_USERS パラメータを設定します。次の例では、最大ユーザー数が 200 に設定されます。

```
LICENSE_MAX_USERS = 200
```

データベースを起動するときに、LICENSE_MAX_USERS よりも多くのユーザーがデータベース内に存在すると、Oracle は警告を戻し、ALERT ファイルに適切なメッセージを書き込みます。ユーザー数が制限を下回るまで、ユーザーを削除するか、または、Oracle ライセンスをアップグレードするまで、追加のユーザーを作成することはできません。

ユーザー制限の変更

名前付きユーザーの最大制限数を変更するには、LICENSE_MAX_USERS オプションを指定した ALTER SYSTEM コマンドを使用します。次の文は、定義されるユーザーの最大数を 300 に変更します。

```
ALTER SYSTEM SET LICENSE_MAX_USERS = 300;
```

制限を現行のユーザー数よりも小さな値に変更しようとすると、Oracle はエラーを戻し、引き続き古い制限を使用します。制限の変更に成功すると、新しい制限はインスタンスを停止するまで効果があります。制限を半永久的に変更するには、パラメータ・ファイル内の LICENSE_MAX_USERS の値を変更してください。

名前付きユーザー制限の最大値を変更するには、ALTER SYSTEM 権限が必要です。

警告： Oracle ライセンスを適切にアップグレードしない限り、名前付きユーザー制限を大きくしないでください。詳細は、オラクル社カスタマ・サポートまで連絡してください。

ライセンス制限と現行値の参照

V\$LICENSE データ・ディクショナリ・ビューを問い合わせることによって、すべてのライセンス設定の現行制限、セッションの現行数およびインスタンスに対する現行セッションの最大数を参照できます。この情報を使用して、より多くの現行セッションまたは名前付きユーザーを許可するために、Oracle 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判断できます。

```
SELECT sessions_max s_max,
       sessions_warning s_warning,
       sessions_current s_current,
       sessions_highwater s_high,
       users_max
FROM v$license;
```

S_MAX	S_WARNING	S_CURRENT	S_HIGH	USERS_MAX
100	80	65	82	50

さらに、データベースを停止するときにセッションの高水位標がデータベースの ALERT ファイルに書き込まれるため、このファイルで高水位標を確認できます。

データベースに定義されている現行の名前付きユーザー数を確認するには、次の問合せを使用してください。

```
SELECT COUNT(*) FROM dba_users;
```

```
COUNT(*)
-----
174
```

ユーザー認証

ここでは、ユーザーの認証について説明します。この項のトピックは、次のとおりです。

- [データベース認証](#)
- [外部認証](#)
- [企業認証](#)

ユーザーを定義する方法には、データベース・セッションを作成する許可を与える前にユーザー ID を認証する方法に応じて、次の 3 つの方法があります。

1. ユーザーの識別と認証の両方をするように、Oracle を構成できます。これをデータベース認証と呼びます。
2. ユーザーの識別だけをする（認証はオペレーティング・システムまたはネットワーク・サービスです）ように、Oracle を構成できます。これを外部認証と呼びます。
3. ユーザーの識別のみを実行するように、Oracle を構成できます。これを企業認証と呼びます。

データベース認証

ユーザーのデータベース認証を選択すると、ユーザー・アカウント、パスワード管理およびそのユーザーの認証の管理全体を Oracle が行います。Oracle にユーザーを認証させるには、ユーザーの登録または変更のときに、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も、各パスワードはシングルバイト・キャラクタによって構成される必要があります。

データベース認証を使用するとき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さの検証など、パスワード管理の使用をお勧めします。

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。

```
CREATE USER scott IDENTIFIED BY tiger;
```

関連項目 : CREATE USER 文と ALTER USER 文の詳細は、『Oracle8i SQL リファレンス』を参照してください。

有効なパスワードの詳細は、『Oracle8i SQL リファレンス』を参照してください。

Oracle パスワード管理の詳細は、[第 22 章「セキュリティ方針の設定」](#)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は、データベースで制御される。データベースの外部のものに依存しません。
- Oracle は、データベース認証の使用時に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を提供しています。
- 小さいユーザー・コミュニティの管理が容易になります。

外部認証

ユーザーの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り行われます。この外部サービスは、オペレーティング・システムでも Net8 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、そのオペレーティング・システムにユーザーを認証させることができます。そのようにする場合、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使用してください。このパラメータは、あらゆる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、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=OP$
```

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しようとし、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 `OP$TSMITH` の存在を調べ、存在していれば接続を許可します。オペレーティング・システムによって認証されるユーザーへのすべての参照には、`OP$TSMITH` のように、接頭辞が含まれていなければなりません。

このパラメータのデフォルト値は `OP$` であり、以前のバージョンの Oracle と下位互換です。ただし、接頭辞の値は他の文字列や `NULL` 文字列（2 つの二重引用符 "" で指定）も設定できます。`NULL` 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ため、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中は同じ設定のまま維持されます。接頭辞を変更する場合、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限り、そのユーザー名では接続を確立できません。

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の作成には、次のコマンドを使用します。

```
CREATE USER scott IDENTIFIED EXTERNALLY;
```

CREATE USER IDENTIFIED EXTERNALLY を使用すると、データベース管理者は、オペレーティング・システムまたはネットワーク・サービスによる認証が必要なデータベース・アカウントを作成できます。ただし、パスワードを使用しても認証できません。

関連項目 : OS_AUTHENT_PREFIX パラメータのテキストは、オペレーティング・システムにより大 / 小文字の区別が必要になります。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・システム認証

デフォルトでは、オペレーティング・システムで認証されており、安全な接続を介したログインだけが許可されます。したがって、オペレーティング・システムにユーザーを認証させる場合、デフォルトでは、そのユーザーは Net8 を介してデータベースに接続できません。つまり、この接続では Net8 を使用するため、マルチスレッド・サーバーでは接続できません。このデフォルトの制限により、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・システムのユーザーのふりをすることができません。

ネットワーク接続を介してリモート・ユーザーが別のオペレーティング・システムのユーザーのふりをするについて心配がなく、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・ユーザー認証を使用する場合は、データベースのパラメータ・ファイルで REMOTE_OS_AUTHENT (デフォルト設定は FALSE) を TRUE に設定してください。初期化パラメータ REMOTE_OS_AUTHENT を TRUE に設定すると、RDBMS はセキュリティの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を受け入れてアカウント・アクセスに使用できます。この変更は、次回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。

ネットワーク認証

ネットワーク認証は Net8 を介して実行されます。Net8 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。Net8 を唯一の外部認証サービスとして利用する場合、Net8 では安全な接続しか行えないので、パラメータ REMOTE_OS_AUTHENT の設定は無関係です。

関連項目 : ネットワーク認証の詳細は、『Oracle8i 分散システム』を参照してください。

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、フィンガープリント、Kerberos、オペレーティング・システムなどの認証メカニズムの選択肢も使用できます。
- すでに上記のようなある種の外部認証メカニズムを使用している場合、データベースでそのメカニズムを使用すると管理費用も低減できます。

企業認証

ユーザーの企業認証を選択すると、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 Oracle Security Service (OSS) により行われます。この認証サービスは複数の Oracle データベース・サーバーの間で共有でき、このサービスを使用するとユーザーの認証と許可の情報を中央で管理できます。

Oracle によって識別され、Oracle Security Service によって 認証されるユーザー（グローバルなユーザー）の作成には、次のコマンドを使います。

```
CREATE USER scott IDENTIFIED GLOBALLY as '<external name>';
```

関連項目：<EXTERNAL NAME> 文字列の内容の詳細は、『Oracle8i 分散システム』を参照してください。

企業認証の利点

次に、企業認証の利点を示します。

- 多数のデータベースを持つ大きなユーザー・コミュニティの管理が容易になります。
- 業界標準の公開鍵証明書を使用すると、相互操作の機会が増えます。

関連項目：企業認証の詳細は、『Oracle8i 分散システム』を参照してください。

Oracle ユーザー

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。データベースに対してアクセスするには、ユーザーは必ずデータベース・アプリケーションを稼働させた上で、データベース内で定義した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しなければなりません。ここでは、データベースに対するユーザーの管理方法と、次のトピックについて説明します。

- [ユーザーの作成](#)
- [ユーザーの変更](#)
- [ユーザーの削除](#)

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 システム権限が必要です。新規ユーザーの作成時には、その作成者が指定する表領域の割当て制限をもっていなくても、データベース上の任意の表領域を表領域割当て制限として指定できます。CREATE USER は強力なシステム権限なので、通常、この権限を持つユーザーはセキュリティ管理者だけです。

ユーザーを作成するには、SQL 文 CREATE USER を使用します。どちらかのオプションを使用して、新たにユーザーのデフォルト、一時セグメント表領域、表領域割当て制限およびプロファイルも指定できます。

```
CREATE USER OPS$jward
  IDENTIFIED EXTERNALLY
  DEFAULT TABLESPACE data_ts
  TEMPORARY TABLESPACE temp_ts
  QUOTA 100M ON test_ts
  QUOTA 500K ON data_ts
  PROFILE clerk;
```

関連項目 : 新たに作成されたユーザーは、CREATE SESSION システム権限が付与されるまで、データベースに接続できません。24-10 ページの「[システム権限とロールの付与](#)」を参照してください。

名前の指定

各データベース内では、ユーザー名は他のユーザー名とロールに対して一意にしてください。ユーザーとロールは同じ名前を持つことができません。さらに、各ユーザーには対応するスキーマがあります。スキーマ内では、各スキーマ・オブジェクトに必ず一意の名前を指定してください。

ユーザー認証の設定

前の CREATE USER 文では、新規ユーザーをオペレーティング・システムを使用して認証しました。ユーザー名には、デフォルトの接頭辞 "OPSS" が含まれています。OS_AUTHENT_PREFIX パラメータを異なるものに設定する場合（接頭辞を指定しなかったり、別の接頭辞を指定したりした場合）必要に応じて接頭辞を省略または適切な接頭辞に置き換えることでユーザー名を修正できます。

また、次のようにデータベースとパスワードを使用して認証されるユーザーも作成できます。

```
CREATE USER jward
  IDENTIFIED BY airplane
  . . . ;
```

この場合、接続に成功するために、接続ユーザーはデータベースに対して正しい接続パスワードを指定してください。

マルチバイト・キャラクタ・セットを使用する場合のユーザー・パスワード 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、パスワードには必ずシングルバイト・キャラクタだけを使用してください。パスワードでは、マルチバイト・キャラクタは受け入れられません。

関連項目 : 有効なパスワードの詳細は、『Oracle8i SQL リファレンス』を参照してください。

デフォルト表領域の割当て

各ユーザーはそれぞれデフォルト表領域を持っています。ユーザーがスキーマ・オブジェクトを作成して、このオブジェクトを格納する表領域を割り当てないと、こ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。

すべてのユーザーのデフォルトの表領域に対するデフォルトの設定値は、SYSTEM 表領域です。ユーザーがオブジェクトを作成しない場合には、デフォルトの設定値が最適です。ただし、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、そのユーザーのデフォルトの表領域を個別に設定するように考慮してください。ユーザー作成中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、設定値の変更後に作成されたオブジェクトだけがこの変更の影響を受けます。

指定する表領域を決定する際、次の項目を考慮します。

- ユーザーが任意のオブジェクト（表、ビューおよびクスタなど）を作成する権限を持っている場合に限り、ユーザーのデフォルト表領域を設定します。
- ユーザーが割当て制限を有している表領域には、ユーザーのデフォルトの表領域を設定します。
- 可能な場合には、データ・ディクショナリ・オブジェクトとユーザー・オブジェクトとの間の競合を削減するため、SYSTEM 表領域以外の表領域にユーザーのデフォルト表領域を設定します。

前の例の CREATE USER 文では、JWARD のデフォルトの表領域は DATA_TS になります。

一時表領域の割当て

各ユーザーは、一時表領域も持っています。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。

ユーザーの一時表領域を明示的に設定しなければ、デフォルトは SYSTEM 表領域です。ただし、各ユーザーが一時表領域を設定することによって、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競合が減少します。ユーザーの一時表領域は、ユーザーの作成中に設定し、作成後に変更できます。

前の例の 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だけを格納するために明示的に作成された表領域です。

表領域の割当て制限の割当て

任意の表領域に対する表領域割当て制限を、各ユーザーに割り当てられます。割当て制限による影響は次のとおりです。

- ユーザーになんらかのオブジェクトを作成する権限がある場合に、ユーザーは指定した表領域中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り当てられる領域が、割当て制限以内に設定されます。

デフォルトでは、ユーザーにはデータベース中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には、必ずオブジェクトを作成できる割当て制限を割り当ててください。最低限、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、さらにオブジェクトを作成する際に他の表領域に対する割当て制限を追加で割り当てます。

ユーザーに対して、各表領域の固有のディスク領域に対して割当て制限を個々に割り当てるか、またはすべての表領域中のディスク領域に対して無制限に割り当てるかのどちらかを選択できます。個々に割当て制限を設定することによって、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未然に防げます。

ユーザーの表領域の割当て制限は、ユーザーの作成中に設定し、作成後に追加または変更できます。新たな割当て制限が古い制限値よりも小さければ、次の条件の場合に真となります。

- 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わせた領域が新しい割当て制限を下回らない限り、その表領域のユーザー・オブジェクトにさらに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、または表領域上のユーザーのオブジェクトで使用する領域が新しい表領域割当て制限よりも少ない場合、そのユーザーのオブジェクトに新しい割当て制限の領域を割り当てることができます。

表領域アクセスを取り消す ユーザーの表領域アクセスを取り消すには、そのユーザーの現行の割当て制限をゼロに変更します。いったん、割当て制限 0 を割り当てておけば、取り消された表領域中のユーザーのオブジェクトはそのまま残りますが、そのオブジェクトを新たな領域に割り当ててすることはできません。

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。なお、この権限はロールに対してではなく、ユーザーに限り付与できます。

UNLIMITED TABLESPACE システム権限を付与する前に、その措置の利点と欠点を考慮してください。

利点

- 1 文で、データベースのすべての表領域を無制限にアクセスする権限をユーザーに付与できます。

欠点

- 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** システム権限もつユーザーからは、表領域アクセスを選択的に取り消すことができません。その権限を取り消した後に限り、選択的にアクセスを付与できます。

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するときに、そのユーザーのデフォルト・ロールは ALL に設定されます。これによって、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 コマンドを使用してください。

警告： ユーザー・ロール以外のロールを作成する場合は、そのロールを暗黙に付与し、デフォルトのロールとして追加します。複数の MAX_ENABLED_ROLES がある場合、ログインでエラーになります。このエラーを回避するには、ユーザーのデフォルトのロールを MAX_ENABLED_ROLES より小さな値に変更します。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、唯一このシステム権限を持つユーザーのみがセキュリティ管理者なのは、その権限によりユーザーのセキュリティ定義域を修正できるからです。なお、修正を実行するユーザーが表領域に対する割当て制限を持たなくても、データベースの任意の表領域上のユーザーのために表領域割当て制限を設定する機能が、この権限には含まれています。

SQL 文 ALTER USER を使用すると、ユーザーのセキュリティ設定を変更できます。ユーザー・セキュリティの設定値の変更は、現在のセッションにではなく、それ以降のセッションに反映されます。

次の文は、ユーザー AVYRROS のセキュリティの設定値を変更します。

```
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 のセキュリティの設定値は次のように変更されます。

- AVYRROS のオペレーティング・システム・アカウントを使用して認証を変更する。

- AVYRROS のデフォルトと一時表領域を明示的に設定します。
- AVYRROS には、DATA_TS 表領域として 100MB 分の割当て制限を指定します。
- TEST_TS に対する AVYRROS の割当て制限を取り消します。
- AVYRROS を CLERK プロファイルに割り当てます。

ユーザーの認証メカニズムの変更

DBA 以外のほとんどのユーザーも、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;
```

特別な権限（データベースに接続するための権限以外の権限）がなくても、ユーザーは誰でもこの方法で自分のパスワードを変更できます。ユーザーは自分のパスワードを必要に応じて変更するとよいでしょう。

Oracle データベース認証、オペレーティング・システム認証および企業認証の間を変更するには、ALTER USER 権限が必要です。通常、この権限を持つのは DBA だけです。

マルチバイト・キャラクタ・セットを使用する場合のパスワード 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、パスワードには必ずシングルバイト・キャラクタだけを使用してください。パスワードでは、マルチバイト・キャラクタは受け入れられません。

関連項目：有効なパスワードの詳細は、『Oracle8i SQL リファレンス』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となるように設定されたものです。ユーザーにはゼロ以上のデフォルト・ロールを割り当てることができます。

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 24 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

ユーザーが削除されると、そのユーザーと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、さらに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、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、ユーザーからデータベースへのアクセスを拒否しなければならない場合、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てください。

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 権限が必要です。DROP USER システム権限は非常に強力な権限であるため、通常セキュリティ管理者がこの権限を持つ唯一のユーザーになります。

SQL 文 DROP USER を使用すると、データベースからユーザーを削除できます。

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・メッセージが戻され、ユーザーは削除されません。スキーマがオブジェクトを含んでいるユーザーを削除する場合には、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、削除による影響を事前に調査しておく必要があります。カスケードによる未知の影響に注意してください。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。

```
DROP USER jones CASCADE;
```

関連項目： セッションの停止の詳細は、4-15 ページの「[セッションの停止](#)」を参照してください。

プロファイルによるリソースの管理

プロファイルは一連の命名されたリソース制限です。リソース制限が起動されると、データベースおよびインスタンス・リソースの使用が、そのプロファイルに指定されたとおりに制限されます。各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ることができます。プロファイルを実施するには、リソース制限をデータベース全体に起動してください。

ここでは、プロファイル管理について説明します。この項のトピックは、次のとおりです。

- [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](#)

- 複合制限の使用
- プロファイルの削除
- リソース制限を使用可能、使用禁止にする方法

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、CLERK プロファイルを作成します。

```
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に対してリソース制限をまったく指定していない場合には、DEFAULT プロファイルによって制限値が設定されます。また、DEFAULT プロファイルに対しても制限を指定できます。

DEFAULT プロファイルの使用

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、ユーザーは DEFAULT プロファイルすべての制限に準拠します。
- プロファイルにまったく制限を指定していない場合、DEFAULT プロファイルに対応する制限を使用します。

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。ただし、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、セキュリティ管理者は 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 システム権限を有するユーザーは、DEFAULT プロファイル中でそ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、任意に単一のプロファイルを割り当てることができます。すでにプロファイルを有しているユーザーにプロファイルを割り当てると、最新のプロファイルの割当てが前回割り当てたプロファイルを置き換えます。プロファイルの割当ては現在のセッションにはまったく影響しません。プロファイルは、ロールや他のプロファイルではなく、ユーザーに限り、割り当てられます。

プロファイルは、SQL 文 CREATE USER または ALTER USER を使用してユーザーに割り当てることができます。

関連項目 : ユーザーのプロファイルの割当ての詳細は、23-10 ページの「[ユーザーの作成](#)」および 23-14 ページの「[ユーザーの変更](#)」を参照してください。

プロファイルの変更

プロファイルのうち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ことによって、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。プロファイルを変更する際に調整しなかったプロファイルは、すべて前回の設定値を維持します。プロファイルに対してなされた変更は、現行の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だけ使用されます。

次の文は、CLERK プロファイルを変更します。

```
ALTER PROFILE clerk LIMIT
  CPU_PER_CALL default
  LOGICAL_READS_PER_SESSION 20000;
```

関連項目 : デフォルトのプロファイルの詳細は、23-17 ページの「[DEFAULT プロファイルの使用](#)」を参照してください。

複合制限の使用

複合制限によりセッションに対するリソース・コストの合計を制限できます。プロファイルに対する特定のリソース制限を明示的に設定するだけでなく、単一複合制限では、プロファイル上のリソース制限に対するアカウントを設定できます。プロファイルの複合制限は、SQL 文 CREATE PROFILE または ALTER PROFILE の COMPOSITE_LIMIT パラメータで設定できます。複合制限は、各リソースの加重値であるサービス単位を介して設定されます。

次の CREATE PROFILE 文は、COMPOSITE_LIMIT パラメータを使用して定義されています。

```
CREATE PROFILE clerk LIMIT
  COMPOSITE_LIMIT 20000
  SESSIONS_PER_USER 2
  CPU_PER_CALL 1000;
```

明示的に指定したリソース制限と複合制限は、プロファイルごとに共存できます。最初に適合した制限は、セッションのアクティビティを停止します。複合制限は、システム・リソースの使用を制限する際に、柔軟性のある制限を実現します。

複合制限の値の判断

適正な複合制限は、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総合計に応じて異なります。各固有のリソース制限と同様に、履歴情報は、平均的プロファイル・ユーザーのための複合リソースの通常範囲を決定するために、必ず収集してください。

関連項目：複合制限の計算方法は、『Oracle8i 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

システムにはそれぞれ独自の特徴があります。中には他よりも価値があるシステム・リソースもあります。Oracle では、各システム・リソースにコストを設定できます。コストは、データベース・レベルで各システム・リソースに重み付けをします。コストは、プロファイルの複合制限にのみ適用されます。ただし、個々のリソース制限を明示的に設定する場合には、コストは適用されません。

リソース・コストを設定するには、ALTER RESOURCE システム権限が必要です。

コストを設定できるのは、CPU_PER_SESSION、LOGICAL_READS_PER_SESSION、CONNECT_TIME、PRIVATE_SGA など、特定のリソースのみです。データベースのコストは、SQL コマンド ALTER RESOURCE COST を使用して設定します。

```
ALTER RESOURCE COST
  CPU_PER_SESSION 1
  LOGICAL_READS_PER_SESSION 50;
```

多額なコストとはリソースが非常に高価であることを意味し、少額のコストとはリソースが高価でないことを意味しています。デフォルトでは、最初に各リソースにコスト 0 が設定されます。コスト 0 は、リソースを複合制限で考慮してはならない（すなわち、このリソースを使用するコストはかからない）ことを意味します。なお、リソースをコスト NULL に指定できません。

関連項目：リソース・コストの設定に関する追加情報と推奨事項は、オペレーティング・システム固有の Oracle マニュアルと『Oracle8i SQL リファレンス』を参照してください。

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。

次の文では、CLERK プロファイルを削除します（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;
```

削除され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 のプロファイルに割り当てられます。また、DEFAULT プロファイルは削除できません。プロファイルが削除されていても、その削除は現在のアクティブ・セッションには影響しません。プロファイルが削除された後に作成されたセッションに限り、修正済みのプロファイルの割当てに従います。

リソース制限を使用可能、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されます。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。リソース制限の強制を使用可能にするかどうかは、次の部分で説明する 2 種類の方法で設定できます。

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強制を変更するには、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、使用禁止にする

データベースを一時的に停止する場合には、データベースのパラメータ・ファイル中の RESOURCE_LIMIT の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。パラメータとして有効な値は、TRUE（強制可）と FALSE です。特に何も指定しなければ、パラメータの値は FALSE に設定されます。一度、パラメータ・ファイルを編集した場合、データベース・インスタンスを再起動して、その編集を反映してください。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の強制の使用可能または使用禁止が設定されます。

データベースのオープン中にリソース制限を使用可能、使用禁止にする

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には、SQL コマンド ALTER SYSTEM を使用してリソース制限の強制の可否のいずれかを選択してください。一度、インスタンスを起動すると、ALTER SYSTEM 文は RESOURCE_LIMIT パラメータで値の設定を置き換えます。たとえば、次の指定文はデータベースに対するリソース制限を強制的に設定します。

```
ALTER SYSTEM
  SET RESOURCE_LIMIT = TRUE;
```

注意： これは、パスワード・リソースに適用されません。

ALTER SYSTEM 文はリソース制限の強制を永続的に決定するわけではありません。データベースが停止または再起動されると、RESOURCE_LIMIT パラメータに設定されている値によってリソース制限の強制が決定します。

データベース・ユーザーとプロファイルに関する情報のリスト

データ・ディクショナリは、すべてのユーザーとプロファイルに関する情報を格納しています。次の情報が含まれています。

- データベース内のすべてのユーザー
- 表、クラスタおよび索引に対する各ユーザーのデフォルト表領域
- 一時セグメントのための各ユーザーの表領域
- 各ユーザーの領域割当て制限（ある場合）
- 各ユーザーに割り当てられているプロファイルとリソース制限
- 適用可能な各システム・リソースに割り当てられているコスト
- 各現行セッションのメモリー使用

次のデータ・ディクショナリ・ビューは、データベース・ユーザーおよびプロファイルに係るものです。

- ALL_USERS
- USER_USERS
- DBA_USERS
- USER_TS_QUOTAS
- DBA_TS_QUOTAS
- USER_PASSWORD_LIMITS
- USER_RESOURCE_LIMITS
- DBA_PROFILES
- RESOURCE_COST
- V\$SESSION
- V\$SESSTAT
- VSSTATNAME

関連項目：各ビューの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

ユーザーとプロファイルに関する情報の記述例

この部分の例は、次の文を実行したデータベースを想定しています。

```
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"
    PASSWORD_LIFETIME 60
    PASSWORD_GRACE_TIME 10;
```

すべてのユーザーとその関連情報の記述

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示したものです。

```
SELECT username, profile, account_status from dba_users;
-----
```

USERNAME	PROFILE	ACCOUNT_STATUS
SYS	DEFAULT	OPEN
SYSTEM	DEFAULT	OPEN
BLAKE	DEFAULT	OPEN
SCOTT	DEFAULT	OPEN
ADAMS	DEFAULT	OPEN
JFEE	DEFAULT	OPEN
DCRANNEY	DEFAULT	OPEN
JONES	DEFAULT	OPEN
CLARK	DEFAULT	OPEN
U	DEFAULT	LOCKED

パスワードはすべてセキュリティ維持のために暗号化されています。

表領域割当て制限の記述

次の問合せは、各ユーザーごとに割り当てられている表領域割当て制限をすべて示すものです。

```
SELECT * FROM sys.dba_ts_quotas;
```

TABLESPACE	USERNAME	BYTES	MAX_BYTES	BLOCKS	MAX_BLOCKS
SYSTEM	SYSTEM	0	0	0	0
SYSTEM	JFEE	0	512000	0	250
SYSTEM	DCRANNEY	0	-1	0	-1

特定の割当て制限を設定すると、正確な数値が MAX_BYTES 列に示されます。制限のない割当ては "-1" で示されます。

すべてのプロファイルと割り当てられている制限の記述

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記述します。

```
SELECT * FROM sys.dba_profiles
ORDER BY profile;
```

PROFILE	RESOURCE_NAME	RESOURCE	LIMIT
DEFAULT	COMPOSITE_LIMIT	KERNEL	UNLIMITED
DEFAULT	SESSIONS_PER_USER	KERNEL	1
DEFAULT	CPU_PER_CALL	KERNEL	UNLIMITED
DEFAULT	LOGICAL_READS_PER_CALL	KERNEL	UNLIMITED
DEFAULT	CONNECT_TIME	KERNEL	30
DEFAULT	IDLE_TIME	KERNEL	600
DEFAULT	LOGICAL_READS_PER_SESSION	KERNEL	UNLIMITED
DEFAULT	CPU_PER_SESSION	KERNEL	UNLIMITED
DEFAULT	PRIVATE_SGA	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
PROF	COMPOSITE_LIMIT	KERNEL	DEFAULT
PROF	PRIVATE_SGA	KERNEL	DEFAULT
PROF	CONNECT_TIME	KERNEL	DEFAULT
PROF	IDLE_TIME	KERNEL	DEFAULT
PROF	LOGICAL_READS_PER_CALL	KERNEL	DEFAULT
PROF	LOGICAL_READS_PER_SESSION	KERNEL	DEFAULT

```
PROF          SESSIONS_PER_USER      KERNEL      DEFAULT
PROF          CPU_PER_CALL            KERNEL      DEFAULT
PROF          CPU_PER_SESSION         KERNEL      DEFAULT
PROF          FAILED_LOGIN_ATTEMPTS   PASSWORD    5
PROF          PASSWORD_LIFE_TIME      PASSWORD    60
PROF          PASSWORD_REUSE_MAX      PASSWORD    UNLIMITED
PROF          PASSWORD_LOCK_TIME      PASSWORD    1
PROF          PASSWORD_GRACE_TIME     PASSWORD    10
PROF          PASSWORD_VERIFY_FUNCTION PASSWORD    UNLIMITED
PROF          PASSWORD_REUSE_TIME     PASSWORD    60
32 rows selected.
```

ユーザー・セッションあたりのメモリーの使用状況の確認

次の問合せは、すべての現行セッションを記述し、セッションあたりの Oracle ユーザーと現在のメモリー使用を示します。

```
SELECT username, value || 'bytes' "Current session 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_MEMORY';
```

"Current session memory" に示される領域は、マルチスレッド・サーバーを通じて接続している各セッションの共有ブール内に割り当てられています。PRIVATE_SGA リソース制限によって、ユーザーあたりに割り当てられるメモリー容量を制限できます。

インスタンス起動後に、各セッションに割り当てられた最大メモリーを確認するには、この問合せの 'session memory' を 'max session memory' に置き換えてください。

例

ここでは、この章全体で説明しているファンクションを使用する例を示します。

- 1. 次の文は、プロファイル prof を作成します。

```
CREATE PROFILE prof limit
  FAILED_LOGIN_ATTEMPTS 5
  PASSWORD_LIFE_TIME 60
  PASSWORD_REUSE_MAX 60
  PASSWORD_REUSE_TIME UNLIMITED
  PASSWORD_VERIFY_FUNCTION verify_function
  PASSWORD_LOCK_TIME 1
  PASSWORD_GRACE_TIME 10;
```

2. 次の文は、プロファイル prof でユーザー名と同じパスワードを持つユーザーを作成します。

```
CREATE USER userscott IDENTIFIED BY userscott PROFILE prof;
ORA-28003: Password verification for the specified password failed
ORA-20001: Password same as user
```

3. 次の文は、プロファイル prof により "scott%" で識別されるユーザー userscott を作成します。

```
CREATE USER userscott IDENTIFIED BY "scott%" PROFILE prof;
```

4. 次の文は、ユーザーのパスワードを "scott%" にもう一度変更し、エラーを戻します。

```
ALTER USER userscott IDENTIFIED BY "scott%";
ORA-28007: The password cannot be reused
```

5. 次の文は、ユーザー・アカウントをロックします。

```
ALTER USER userscott ACCOUNT LOCK;
```

6. 次の文は、ユーザー・アカウントの状態をチェックします。

```
SELECT username, user_id, account_status, lock_date
FROM dba_users
WHERE username='USERSCOTT';
```

7. 次の文は、パスワードを期限切れにします。

```
ALTER USER userscott PASSWORD EXPIRE;
```

8. 次の文は、ユーザー・アカウントの状態をチェックします。

```
SELECT username, user_id, account_status, expiry_date
FROM dba_users
WHERE username='USERSCOTT';
```

9. 次の文は、ユーザーをアンロックします。

```
ALTER USER userscott ACCOUNT UNLOCK;
```

10. 次の文は、アカウントの状態をチェックします。

```
SELECT username, user_id, account_status, expiry_date
FROM dba_users
WHERE username='USERSCOTT';
```

ユーザー権限とロールの管理

この章では、権限およびロールを使用して、システム操作を実行する機能と、スキーマ・オブジェクトへのアクセスを制御する方法について説明します。この章のトピックは、次のとおりです。

- [ユーザー権限の識別](#)
- [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](#)
- [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てロールを付与](#)
- [権限とロールの情報の記述](#)

関連項目 : データベースへのアクセスの制御の詳細は、[第 23 章](#)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・セキュリティ方針の詳細は、[第 22 章](#)を参照してください。

ユーザー権限の識別

ここでは、Oracle ユーザー権限について説明します。この項のトピックは、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へアクセスするための権利です。また、通常まとめて付与される権限または取り消される権限をグループ化するショートカットも用意されています。

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 1 つまたは複数の特定のデータベース操作を実行できるようになります。

セキュリティ上の理由により、システム権限ではユーザーからデータ・ディクショナリにアクセスできません。したがって、ANY 権限（UPDATE ANY TABLE、SELECT ANY TABLE または CREATE ANY INDEX など）のユーザーは、PUBLIC に付与されていないディクショナリ表やビューにはアクセスできません。

警告： 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トラステッド・ユーザーに付与する場合には十分に注意してください。ANY 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ません。

関連項目：すべてのシステム権限のリストと詳細は、『Oracle8i 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

ディクショナリ保護のメカニズムによって、権限のないユーザーがディクショナリ・オブジェクトにアクセスできないようになっています。

ディクショナリ・オブジェクトへのアクセスは、ユーザー SYSDBA と SYSOPER に制限されています。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、ディクショナリ・オブジェクトにはアクセスできません。たとえば、SELECT ANY TABLE 権限によって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・ビューの実表、ビュー、パッケージおよびシノニム）は選択できません。

また、SQL*Plus コマンド `connect SYS/password` で接続しようとしても失敗に終わります。ただし、次の 2 つの SQL*Plus コマンドは有効です。

```
connect SYS/password as SYSDBA
connect SYS/password as SYSOPER
```

頻繁に使用するディクショナリへのアクセス

明示的なオブジェクト権限のあるユーザーと SYSDBA は、ディクショナリ・オブジェクトにアクセスできます。ただし、ディクショナリ・オブジェクトにアクセスする必要があるのに明示的なオブジェクト権限がない場合は、次のロールを付与されるようにできます。

- SELECT_CATALOG_ROLE

ユーザーは、このロールに付与されたエクスポート済みのすべてのカタログ・ビューと表を SELECT で選択できます。このロールを、データ・ディクショナリのすべてのエクスポート済みビューと表にアクセスする必要があるユーザーに付与してください。

- EXECUTE_CATALOG_ROLE

ディクショナリのエクスポート済みパッケージに対する EXECUTE 権限を提供します。

- DELETE_CATALOG_ROLE

ユーザーは、AUD\$ 表からレコードを削除できます。

これらのロールによって、データベース管理者は、ディクショナリのセキュリティをメンテナンスするときにディクショナリ中の特定のオブジェクトにアクセスできます。

注意： SYSDBA は、ディクショナリ内でエクスポートされていないオブジェクトに関しては、どのユーザーにもオブジェクト権限を付与しないでください。そのようにすると、データベースの整合性が失われることがあります。

関連項目： エクスポート済みの表またはビューの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

オブジェクト権限

オブジェクトの種類ごとに異なるオブジェクト権限が対応付けられます。オブジェクトとそれに対応付けられる権限の詳細リストは、『Oracle8i SQL リファレンス』を参照してください。

オブジェクト権限のショートカット

ALL および ALL PRIVILEGES ショートカットを使用すると、あるオブジェクトに対して使用可能なオブジェクト権限がすべて付与されるか、またはすべて取り消されます。このショートカットは権限ではなく、GRANT 文および REVOKE 文の中の 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にも、個々に権限を取り消せます。

同様に、ALL を使用して、個々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 を使用し、その取消し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）、REVOKE 文に CASCADE CONSTRAINTS オプションを指定してください。

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。この項のトピックは、次のとおりです。

- [ロールの作成](#)
- [事前定義済みロール](#)

ロールは複数の権限やロールをまとめて指定するので、ユーザーに対して同時に権限を付与したり、それを取り消すことができます。ロールはユーザーごとに使用可能にしたり、使用禁止にすることができます。

関連項目：ロールの詳細は、『Oracle8i 概要』を参照してください。

ロールの作成

SQL 文 CREATE ROLE を使用して、ロールを作成します。

ロールを作成するには、CREATE ROLE システム権限が必要になります。通常、セキュリティ管理者だけがこのシステム権限を持っています。

注意： ロールを作成した直後には、ロールに対応付けられている権限はありません。新しいロールに対応付けるには、新しいロールに権限や他のロールを付与してください。

次の文は、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る CLERK ロールを作成します。

```
CREATE ROLE clerk  
IDENTIFIED BY bicentennial;
```

ロール名

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、一意の名前を与えてください。ロールはどのユーザーのスキーマ内にも指定できません。

マルチバイト・キャラクタ・セットのロール名

オラクル社は、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、各ロール名にシングルバイト・キャラクタを少なくとも 1 つ含めることをお勧めします。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み合わせの安全性が大幅に損なわれます。

事前定義済みロール

表 24-1 のロールは、Oracle データベースに対して自動的に定義されます。これらのロールによって、以前のバージョンの Oracle との下位互換性が維持されます。これらの事前定義済みのロールに、ロールに定義できる権限とロールを付与し、取り消すことができます。

表 24-1 事前定義済みロール

ロール名	ロールに付与される権限
CONNECT ¹	ALTER SESSION、CREATE CLUSTER、 CREATE DATABASE LINK、CREATE SEQUENCE、CREATE SESSION、CREATE SYNONYM、CREATE TABLE、CREATE VIEW
CREATE TYPE ⁷	CREATE TYPE、EXECUTE、EXECUTE ANY TYPE
RESOURCE ^{1、2}	CREATE CLUSTER、CREATE INDEXTYPE、 CREATE OPERATOR、CREATE PROCEDURE、CREATE SEQUENCE、 CREATE TABLE、CREATE TRIGGER、 CREATE TYPE
DBA ^{1、3、4}	ADMIN OPTION 付きのすべてのシステム権限
EXP_FULL_DATABASE ⁵	表 SYS.INCVID、SYS.INCFIL、SYS.INCEXP に 対する SELECT ANY TABLE、BACKUP ANY TABLE、INSERT、DELETE、UPDATE
IMP_FULL_DATABASE ⁵	BECOME USER
DELETE_CATALOG_ROLE ⁶	このロールに関するすべてのディクショナリ・ パッケージに対する DELETE 権限
EXECUTE_CATALOG_ROLE ⁶	このロールに関するすべてのディクショナリ・ パッケージに対する EXECUTE 権限
SELECT_CATALOG_ROLE ⁶	このロールに関するすべてのカタログ表および ビューに対する SELECT 権限
RECOVERY_CATALOG_OWNER ⁸	DROP ROLE RECOVERY_CATALOG_OWNER、CREATE ROLE RECOVERY_CATALOG_OWNER、 CREATE TRIGGER、CREATE PROCEDURE TO RECOVERY_CATALOG_OWNER
HS_ADMIN_ROLE ⁹	HS_EXTERNAL_OBJECT、 HS_EXTERNAL_USER
AQ_USER_ROLE ¹⁰	

表 24-1 事前定義済みロール (続き)

ロール名	ロールに付与される権限
AQ_ADMINISTRATOR_ROLE ¹⁰	
SNMPAGENT ¹¹	
¹ SQL.BSQ によって作成されます。下位互換性を確保するためのものです。使用しないようにお勧めします。 ² RESOURCE ロールの権限受領者は、UNLIMITED TABLESPACE システム権限を (RESOURCE ロールの一部としてではなく) 明示的に付与されるものとして受領します。下位互換性を確保するためのものです。使用しないようにお勧めします。 ³ DBA ロールの権限受領者は、ADMIN オプションを持つ UNLIMITED TABLESPACE システム権限を (DBA ロールの一部としてではなく) 明示的に付与されるものとして受領します。したがって、DBA ロールが取り消されると、明示的に付与された UNLIMITED TABLESPACE もすべて取り消されます。 ⁴ CATEXPSQL が実行されている場合は、EXP_FULL_DATABASE ロールと IMP_FULL_DATABASE ロールも含まれます。 ⁵ CATEXPSQL によって作成されます。 ⁶ DBA ロールはないが、データ・ディクショナリ内のビューと表へのアクセスが必要なユーザーには、このロールを付与してください。 ⁷ Oracle オブジェクトのオプションがユーザーのデータベース・サーバーにインストールされている場合にのみ、CREATE TYPE コマンドを使用できます。 ⁸ CAT.SQL によって作成されます。 ⁹ CATQUEUE.SQL によって作成されます。 ¹⁰ Advanced Queuing オプションがある場合にのみ付与されます。 ¹¹ Intelligent Agents オプションがある場合にのみ付与されます。	

ロールの認可

ユーザーがデータベース・ロールを使用可能にするとき、そのロールはオプションで認可を要求できます。ロールの認可は、データベース (パスワードを使用) オペレーティング・システムまたはネットワーク・サービスによってメンテナンスされます。

ロールの認可方法を変更するには、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 : ネットワーク・ロールの詳細は、『Oracle8i 分散システム』を参照してください。

データベースによるロールの認可

ロールの使用は、こ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る場合、そのロールの適切なパスワードを SET ROLE 文指定した場合のみ、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。

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、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。パスワードでは、マルチバイト・キャラクタは受け入れられません。

関連項目： 有効なパスワードの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

オペレーティング・システムによるロールの認可

次の文は、ACCTS_REC という名前のロールを作成します。このロールを使用するための認可をオペレーティング・システムから受けることが必要です。

```
CREATE ROLE role IDENTIFIED EXTERNALLY;
```

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。

ロールがオペレーティング・システムによって認可される場合には、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してください。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、オペレーティング・システムにその認可を求め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、24-17 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てロールを付与](#)」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 Net8 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可できません。マルチスレッド・サーバーを介する接続には Net8 が必要であるためです。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・ユーザーとして不正に接続する恐れがあるため、この制限がデフォルトになっています。

このようなセキュリティの危険性の心配がなく、ネットワーク・クライアントに対してオペレーティング・システムによるロール認証を使用する場合は、データベースのパラメータ・ファイルにあるパラメータ REMOTE_OS_ROLES を TRUE に設定してください。この変更は、次回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（パラメータのデフォルト設定は FALSE です）。

認可の保留

認可を指定しないでロールを作成することもできます。保護も設定しないでロールを作成すると、どの権限受領者でもその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。

ロールの認可の変更

ロールの認可方法は、SQL 文 ALTER ROLE を使用して設定し、変更します。

次の文は、CLERK ロールを外部的に認可されるように変更します。

```
ALTER ROLE clerk  
IDENTIFIED EXTERNALLY;
```

ユーザーのデフォルト・ロールの変更

ユーザーのデフォルト・ロール・リストは、SQL 文 ALTER USER を使用して設定し、変更します。

関連項目：これらのオプションの詳細は、23-14 ページの「[ユーザーの変更](#)」を参照してください。

MAX_ENABLED_ROLES パラメータの使用 初期化パラメータ MAX_ENABLED_ROLES に指定したロール数だけロールを使用できます。最初のロールが使用可能になった結果、使用可能となる間接的に付与され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はこのパラメータの値を修正することによって、この制限を変更できます。同時に使用可能にするロール数を増やすには、大きな値を各ユーザー・セッションに設定します。ただし、このパラメータの値が大きくなると、ユーザー・セッションごとに大量のメモリー領域が必要になります。これは、ユーザー・セッションごとに PGA サイズが影響を受け、ロールあたり 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。

ロールの削除

状況によっては、ロールをデータベースから削除した方が適切な場合があります。削除したロールを付与されていたユーザーとロールのセキュリティ・ドメインは、削除したロール権限がなくなったことを反映するためただちに更新されます。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、影響を受けていたセキュリティ・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は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。

SQL 文 DROP ROLE を使用して、ロールを作成します。

次の文では、CLERK が削除されます。

```
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。この項のトピックは、次のとおりです。

- [システム権限とロールの付与](#)
- [オブジェクト権限とロールの付与](#)
- [列への権限の付与](#)

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ロールやユーザーに付与するには、SQL 文 GRANT を使用します。

システム権限またはロールを付与するには、付与するすべてのシステム権限およびロールの ADMIN OPTION が必要です。また、GRANT ANY ROLE システム権限を持っているユーザーは、データベース内で任意のロールを付与できます。

次の文は、システム権限および 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を同時に付与できません。

ADMIN オプション

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。ADMIN オプションを持つ権限受領者は、権限の有効範囲が次のように拡大されます。

- 権限受領者は、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するシステム権限またはロールを付与したり取り消すことができます。(ただし、ユーザーは自分自身からロールを取り消すことはできません。)
- 権限受領者は、さらに ADMIN OPTION 付きのシステム権限やロールを付与できます。
- ロールの権限受領者は、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 NEW_DBA ロールを MICHAEL に付与します。

```
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であるため、これを使用してシステム権限やロールを付与する際には、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確保されているため、システム内の他の管理者やユーザーに付与されることはほとんどありません。

オブジェクト権限とロールの付与

ロールとユーザーにオブジェクト権限を付与するには SQL コマンド GRANT を使用します。

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なければなりません。

- 指定するオブジェクトを所有していること。
- GRANT OPTION で付与しているオブジェクト権限を付与されていること。

次の文では、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がユーザー JFEE と TSMITH に付与されます。

```
GRANT select, insert, delete ON emp TO jfee, tsmith;
```

ユーザー JFEE と TSMITH に、EMP 表の ENAME 列と JOB 列の INSERT オブジェクト権限だけを付与するには、次のように入力します。

```
GRANT insert(ename, job) 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 ショートカットを使用します。

```
GRANT ALL ON salary TO jfee;
```

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できません。

GRANT OPTION

スキーマにオブジェクトを含んでいるユーザーには、GRANT OPTION に対応するすべてのオブジェクト権限が付与されます。この特別な権限によって、権限受領者の権限が拡張されます。

- 権限受領者は、GRANT OPTION の有無に関係なく、データベース内の任意のユーザーやロールにオブジェクト権限を付与できます。
- 権限受領者が GRANT OPTION 付きで表のオブジェクト権限を受け取り、さらにシステム権限 CREATE VIEW または CREATE ANY VIEW を持っている場合、この権限受領者はその表に対してビューを作成し、データベース内の任意のユーザーとロールにそのビューの対応する権限を付与できます。

オブジェクト権限をロールに付与する場合には、GRANT OPTION は無効です。Oracle はロールによるオブジェクト権限の波及を未然に防ぐため、ロールの権限受領者はロールを介して受領したオブジェクト権限を反映できません。

列への権限の付与

表の個々の列に INSERT、UPDATE、REFERENCES 権限を付与できます。

警告： 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判断します。NOT NULL 列を含めずに選択的な挿入機能を付与すると、ユーザーは行を表に挿入できません。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か、NULL 以外のデフォルト値を持っているかを確認してください。それ以外は、権限受領者は行を表に挿入できず、エラーが発生します。

ACCOUNTS 表の ACCT_NO 列の INSERT 権限を SCOTT に付与します。

```
GRANT INSERT (acct_no)
ON accounts TO scott;
```

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。この項のトピックは、次のとおりです。

- システム権限とロールの取消し
- オブジェクト権限とロールの取消し

システム権限とロールの取消し

システム権限や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・ユーザーや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。ロールを取り消すユーザーは、その権限またはロールを最初に付与したユーザーでなくてもかまいません。また、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 は、選択的に取り消すことはできません。権限またはロールを取り消してから、ADMIN OPTION を指定せずにその権限またはロールを再度付与してください。

オブジェクト権限とロールの取消し

オブジェクト権限は、SQL コマンド REVOKE を使用して取り消すことができます。

オブジェクト権限を取り消すユーザーは、取消しの対象となるオブジェクト権限の権限付与者である必要があります。

たとえば、ユーザー JFEE と TSMITH から EMP 表の INSERT 権限を取り消す場合は、この権限付与者自身が次の文を発行します。

```
REVOKE select, insert ON emp  
FROM jfee, tsmith;
```

次の文では、DEPT 表からすべての権限（HUMAN_RESOURCE ロールに最初に付与された権限すべて）が取り消されます。

```
REVOKE ALL ON dept FROM human_resources;
```

注意： この文は、他のユーザーが行った付与を取り消すのではなく、権限付与者が許可した権限のみを取り消します。オブジェクト権限の GRANT OPTION は、選択的に取り消せません。オブジェクト権限を取り消してから、GRANT OPTION を使わずに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。

列に選択的に付与されていたオブジェクト権限の取消し

INSERT、UPDATE、REFERENCES 権限を表やビューの列に選択的に付与できますが、それに類似する 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、権限付与者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、さらに残しておきたい列固有の権限を選択的に再付与してください。

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ためには、次の 2 つの文を入力します。

```
REVOKE UPDATE ON dept FROM human_resources;  
GRANT UPDATE (dname) ON dept TO human_resources;
```

この REVOKE 文が HUMAN_RESOURCES ロールから DEPT 表の全列に対する UPDATE 権限を取り消します。GRANT 文で、DNAME 列の UPDATE 権限を HUMAN_RESOURCES ロールに再度付与します。

REFERENCES オブジェクト権限の取消し

REFERENCES オブジェクト権限の権限受領者が外部キーの制約（現行の制約）を設定するための権限を使用している場合、権限付与者は REVOKE 文に CASCADE CONSTRAINTS オプションを指定することによって、唯一その権限を取り消せます。

```
REVOKE REFERENCES ON dept FROM jward CASCADE CONSTRAINTS;
```

CASCADE CONSTRAINTS オプションを指定すると、取消し REFERENCES 権限を使用して、現在定義されている外部キーの制約が削除されます。

権限の取消しによる影響

権限のタイプによっては、権限を取り消す際に連鎖的な影響が生じる場合があります。

システム権限

DDL 操作に関連したシステム権限を取り消す際には、カスケードする影響はまったくありません。このとき、ADMIN OPTION の使用や、権限付与の有無にはまったく関係ありません。ここでは、次のような条件を想定します。

1. セキュリティ管理者は、ADMIN OPTION を使用して、JFEE に対して CREATE TABLE システム権限を付与します。
2. JFEE は表を作成します。
3. JFEE は、TSMITH に CREATE TABLE システム権限を付与します。
4. TSMITH は表を作成します。
5. セキュリティ管理者は、JFEE から CREATE TABLE システム権限を取り消します。
6. JFEE の表は引き続き存在します。TSMITH は引き続き表と CREATE TABLE システム権限を持っています。

カスケードする影響は、DML 操作に関連するシステム権限の削除の際に発生しています。たとえば、SELECT ANY TABLE をユーザーに付与する際に、そのユーザーがプロシージャを事前に作成してある場合、ユーザーのスキーマに含まれているすべてのプロシージャは、再度使用可能となる前に必ず許可し直してください。

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ので、REVOKE 文を指定する前に必ず十分に検討してください。

- 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を及ぼす可能性があります。たとえば、TEST プロシージャの中に、EMP 表のデータを問合せる SQL 文を指定したとします。EMP 表の SELECT 権限が TEST プロシージャの所有者から削除されると、それ以降そのプロシージャを正常に実行できません。
- ALTER または INDEX オブジェクト権限を取り消しても、ALTER と INDEX DDL の各オブジェクト権限が必要となるオブジェクト定義には影響しません。たとえば、別のユーザーの表に索引を作ったユーザーから、INDEX 権限を取り消しても、その索引は権限が取り消された後も存在します。

- 表に対する REFERENCES 権限をユーザーから取り消すと、その取り消された REFERENCES 権限を必要とするユーザーが定義した外部キーの整合性制約が自動的に削除されます。たとえば、ユーザーの JWARD には DEPT 表の DEPTNO 列に対する REFERENCES 権限が付与されているときに、その DEPTNO 列を参照する EMP 表の DEPTNO 列上で外部キーを作成するとします。EMP 表の DEPTNO 列の REFERENCES 権限を取り消すと、その EMP 表の DEPTNO 列上の外部キーの制約まで、同一操作上で削除されます。
- GRANT OPTION を使用したことの影響としてのオブジェクト権限付与は、権限付与者のオブジェクト権限が取り消されると、取り消されます。たとえば、GRANT OPTION を使用して、SELECT オブジェクト権限が USER1 に付与されているときに、USER2 に対し EMP 表の SELECT 権限を付与するとします。次に SELECT 権限は USER1 から取り消されます。この取消しは USER2 にも同様に適用されます。USER1 と USER2 の取り消された SELECT 権限に従属するオブジェクトは、前述のような影響を受ける可能性もあります（前の説明箇所を参照）。

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、権限とロールの付与と取消しができます。PUBLIC はすべてのデータベース・ユーザーにアクセスできるため、PUBLIC に対して付与された権限やロールには、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にのみ、権限またはロールを PUBLIC に付与してください。これにより、いつでも各データベース・ユーザーは、現行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的な規則が保証されます。

PUBLIC から権限を取り消すと、波及する深刻な影響が発生する可能性があります。DML 操作に関連する権限を PUBLIC から取り消す場合（たとえば、SELECT ANY TABLE、UPDATE ON emp など）、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。したがって、DML に関連する権限を PUBLIC に付与するときには注意が必要です。

関連項目：オブジェクト依存性の詳細は、20-22 ページの「[オブジェクトの依存性の管理](#)」を参照してください。

権限付与とその取消しの実施時期

権限付与とその取消しの処理内容に応じて、それぞれの処理の実施時期は異なります。

- 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、即座に実施されます。
- 任意の対象（ユーザー、ロールおよび PUBLIC）に対する、ロールの付与 / 取消しは、現在のセッション・ユーザーがその付与 / 取消しを実施した後でロールを再度使用できるように、SET ROLE 文を発行するとき、またはその付与 / 取消しを実施した後で新規ユーザー・セッションが作られるときに限り、実施されます。

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てロールを付与

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。この項のトピックは、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、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で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・ロールを付与するかわりに、Oracle を操作するオペレーティング・システムを基にして、接続時にユーザーに対しロールを付与できます。このため、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そのロールを Oracle に対して渡せます。その一部として、各ユーザーのデフォルト・ロールや、ADMIN OPTION を使用してユーザーに対して付与した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に対するユーザーを許可する場合でも、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対し権限を割り当ててください。

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。ネットワーク・ロールの詳細は、『Oracle8i 分散システム』を参照してください。

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・ロールを識別する際に得られる利点は、Oracle データベースに対する権限管理を外部的に実施できることです。つまり、オペレーティング・システムによって提供されるセキュリティ機能がユーザーの権限を制御します。さらに、このオプションを使用すれば、大量のシステム・アクティビティに対するセキュリティの集中管理による各種の利点が生れます。たとえば、MVS Oracle の管理者は RACF グループを、UNIX Oracle の管理者は UNIX グループを、また VMS Oracle の管理者は権限の識別子をそれぞれ使用してデータベース・ユーザーのロールを識別させることができます。

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ことの主な欠点は、権限管理がロール・レベルでしか実行できないことです。つまり、個々の権限は、オペレーティング・システムを使用しては付与できませんが、GRANT 文を使用してデータベース内で付与できます。

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マルチスレッド・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することができます。24-20 ページの「[オペレーティング・システムによるロール管理でのネットワーク接続の使用](#)」を参照してください。

関連項目：ここで説明した機能は、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。オペレーティング・システムによって異なりますので、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

データベースを操作するために、オペレーティング・システムを使用して、セッションの作成時に各ユーザーのデータベース・ロールを識別できるように、初期化パラメータ `OS_ROLES` を `TRUE` に設定してください（そして、インスタンスが実行中であれば、再起動してください）。ユーザーがセッションを作成しようとする、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。

ユーザーに対するデータベース・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（この識別子とは、グループ、権限識別子またはその他の類似する名前です）。ロールの指定により、ユーザーのデフォルトのロールであるのか、`ADMIN OPTION` を使用可能なロールであるのかということまで把握できます。使用しているオペレーティング・システムには関係なく、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。

```
ORA_<ID>_<ROLE>[_][D][A]]
where:
```

ID

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 上では、ID はデータベースのインスタンス識別子です。MVS 上ではマシン・タイプ、UNIX 上ではシステム ID です。

D

このオプション文字は、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します。

A

このオプション文字は、ロールが `ADMIN OPTION` 付きでユーザーに付与されることを示します。これによって、ユーザーはこのロールを他のロールにだけ付与できます（オペレーティング・システムを使用してロールを管理している場合は、ロールをユーザーに付与できません）。

注意： 文字 D または A のどちらかを指定する場合には、アンダースコアをその文字の直前に指定してください。

たとえば、オペレーティング・システム・アカウントは、プロファイルで識別される次のロールを持っている可能性があります。

```
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き、ROLE3 と ROLE4 はデフォルトであり、ROLE2 と ROLE4 は ADMIN OPTION で適用できます。

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。OS 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 OS 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に、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には、オペレーティング・システムはユーザーに対するロールの付与と取消しを完全に管理しています。以前になされた GRANT 文によるユーザーへのロールの付与は適用されません。ただし、それらはデータ・ディクショナリには記述されたままです。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だけが適用されます。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限り付与できます。

OS_ROLES=TRUE の場合にロールを使用可能、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 コマンド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・システム認証を必要とするように定義されていた場合、それは適用されます。ただし、OS_ROLES = FALSE のときに GRANT 文を使用してロールが付与されていても、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（このようなロールを指定しても Oracle では無視されます）。

OS_ROLES = TRUE のとき、ユーザーはパラメータ MAX_ENABLED_ROLES で指定されている数の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でのネットワーク接続の使用

オペレーティング・システムにロールを管理させる場合、デフォルトでは、ユーザーはマルチスレッド・サーバーを通じ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のない接続を介して別のオペレーティング・システム・ユーザーとして不正に接続する恐れがあるため、この制限がデフォルトになっています。

セキュリティの心配がなく、マルチスレッド・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する場合、データベースのパラメータ・ファイル内のパラメータ `REMOTE_OS_ROLES` を `TRUE` に設定してください。この変更は、次回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)。

権限とロールの情報の記述

オブジェクトに対応する権限付与をリスト表示するために、次のデータ・ディクショナリ・ビューを問い合わせることができます。

- `ALL_COL_PRIVS`, `USER_COL_PRIVS`, `DBA_COL_PRIVS`
- `ALL_COL_PRIVS_MADE`, `USER_COL_PRIVS_MADE`
- `ALL_COL_PRIVS_RECD`, `USER_COL_PRIVS_RECD`
- `ALL_TAB_PRIVS`, `USER_TAB_PRIVS`, `DBA_TAB_PRIVS`
- `ALL_TAB_PRIVS_MADE`, `USER_TAB_PRIVS_MADE`
- `ALL_TAB_PRIVS_RECD`, `USER_TAB_PRIVS_RECD`
- `DBA_ROLES`
- `USER_ROLE_PRIVS`, `DBA_ROLE_PRIVS`
- `USER_SYS_PRIVS`, `DBA_SYS_PRIVS`
- `COLUMN_PRIVILEGES`
- `ROLE_ROLE_PRIVS`, `ROLE_SYS_PRIVS`, `ROLE_TAB_PRIVS`
- `SESSION_PRIVS`, `SESSION_ROLES`

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、『Oracle8i リファレンス・マニュアル』を参照してください。

権限およびロール情報の記述例

次の例は、次の指定文の発行が前提条件となっています。

```
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;
```

付与されているすべてのシステム権限の記述

次の問合せの結果、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 sys.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記述

次の問合せの結果、ユーザーと他のロールに対して付与されたロールがすべて戻されます。

```
SELECT * FROM sys.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記述

次の問合せの結果、特定のユーザーに対して付与されたオブジェクト権限（列に固有な権限は含まれていません）がすべて戻されます。

```
SELECT table_name, privilege, grantable FROM sys.dba_tab_privs
      WHERE grantee = 'JWARD';

TABLE_NAME  PRIVILEGE  GRANTABLE
-----
EMP         SELECT    NO
EMP         DELETE    NO
```

あらかじめ付与されている列固有の権限をすべて表示するためには、次の問合せを使用してください。

```
SELECT grantee, table_name, column_name, privilege
      FROM sys.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
```

セッションの現行の権限ドメインの記述

次の問合せの結果、発行者が現在使用できるすべてのロールが表示されます。

```
SELECT * FROM session_roles;

SWILLIAMS が SECURITY_ADMIN ロールを使用可能にしてあり、さらに上記の問合せを
発行すると、次の情報が戻されます。

ROLE
-----
SECURITY_ADMIN
```

次の問合せの結果、この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（明示的に付与されている権限や使用可能なロールはいずれも含まれています）。

```
SELECT * FROM session_privs;
```

SWILLIAMS が SECURITY_ADMIN ロールを使用可能にしてあり、さらに上記の問合せを発行する場合に、次の結果が戻されます。

```
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
```

SECURITY_ADMIN ロールが SWILLIAMS に対して使用禁止となっている場合、最初の問合せではまったく行が戻されませんが、2 番目の問合せで CREATE SESSION 権限に対して 1 行だけが戻されます。

データベースのロールをリストする方法

DBA_ROLES データ・ディクショナリ・ビューを使用して、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が表示されます。たとえば、次の問合せの結果、データベース内のすべてのロールが表示されます。

```
SELECT * FROM sys.dba_roles;
```

ROLE	PASSWORD
-----	-----
CONNECT	NO
RESOURCE	NO
DBA	NO
SECURITY_ADMIN	YES

ロールの権限ドメインの情報の記述

ROLE_ROLE_PRIVS、ROLE_SYS_PRIVS、ROLE_TAB_PRIVS の各データ・ディクショナリ・ビューは、ロール権限のドメインを含むことができます。

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
```

次の問合せの結果、SECURITY_ADMIN ロールに対して付与されたシステム権限すべてが表示されます。

```
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

次の問合せの結果、SECURITY_ADMIN ロールに対して付与された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

データベース使用の監査

この章では、Oracle 監査機能の使用方法について説明します。トピックは、次のとおりです。

- [監査機能のガイドライン](#)
- [データベース監査証跡ビューの作成および削除](#)
- [監査証跡情報の管理](#)
- [データベース監査証跡情報の参照](#)
- [データベース・トリガーによる監査](#)

監査機能のガイドライン

ここでは、監査機能のガイドラインについて説明します。トピックは、次のとおりです。

- データベースまたはオペレーティング・システムによる監査
- 監査済み情報の管理しやすい状態での維持

データベースまたはオペレーティング・システムによる監査

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言います。

データベース監査証跡またはオペレーティング・システム監査証跡のどちらかで、文監査、権限監査またはオブジェクト監査の結果として生成された監査レコードをすべて格納できます。

なお、オペレーティング・システムによっては、オペレーティング・システム監査証跡のためのデータベース監査がサポートされているものと、そうでないものがある可能性があります。このオプションが適用可能な場合には、データベース監査レコードを格納するために、データベース監査証跡またはオペレーティング・システム監査証跡のどちらかを適用するに当たり、それぞれの長所および短所を考慮しておく必要があります。

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・ディクショナリ中で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参照できます。
- Oracle Tools (Oracle Reports など) を使用すると、監査レポートを生成できます。

また、オペレーティング・システム監査証跡によって、Oracle やその他のアプリケーションなどの複数のソースから監査レコードを整理統合できます。したがって、監査レコードがすべて 1 か所にまとめられるため、システム・アクティビティの検証がより効率的に実行されます。

関連項目：オペレーティング・システムには、オペレーティング・システムの監査機能で生成される監査レコードを格納する監査証跡が含まれていることもあります。ただし、この機能はオペレーティング・システムによって異なります。使用しているオペレーティング・システム固有 Oracle 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

監査には比較的成本がかかりますが、可能な限り監査するイベントの回数を制限しておく必要があります。制限することによって、監査される文の実行に対するパフォーマンスの影響を最小限に抑え、監査証跡のサイズも最小限に抑えられます。

次の一般ガイドラインに従って、監査方針を作成してください。

- 監査目的の評価。

監査の目的を明確にしておく、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も済みます。

たとえば、疑わしいデータベース・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、または気になるのか、といった具体的な情報が必要です。たとえば、焦点を絞り込んだ監査目的というのは、データベース内の表から許可されていない削除の監査である可能性もあります。この目的であれば、監査の対象となるアクティビティのタイプや、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タイプを制限できます。

- 十分に理解した上での監査。

対象となる情報の入手に必要な最小数の文、ユーザーまたはオブジェクトを監査します。こうすれば、不必要な監査情報が重要な情報を混乱させたり、SYSTEM 表領域内の貴重な領域を消費しません。十分なセキュリティ情報収集の必要性和、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には、追跡するアクティビティのタイプを正確に決定し、目的のアクティビティのみを監査し、必要な情報を収集するために必要な時間数のみを監査します。たとえば、各セッションの論理 I/O 情報だけに關心があれば、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

監査の目的がデータベース・アクティビティを監視することである場合には、次のガイドラインを考慮に入れてください。

- 全体的に監査してから特定の対象を監査。

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、対象とする特定のユーザーまたはスキーマ・オブジェクトに対して使用可能な情報はあまりありません。このため、通常は、最初に全体的な監査オプションを設定する必要があります。一度予備的な監査情報が記録され分析されると、この全体的な監査オプションの使用を終了し、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対する具体的な結論が出るまで（十分な裏付けが収集できるまで）は、必ず続行しなければなりません。

- 監査証拠の保護。

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、監査証拠を保護して、監査情報が監査されていない状態では追加、変更または削除されないようにします。

関連項目：監査証拠の詳細は、25-15 ページの「[監査証拠を保護する方法](#)」を参照してください。

通常のデータベース・アクティビティの監査

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には、次のガイドラインを考慮に入れてください。

- 関連性のあるアクションのみを監査すること。
重要な情報の中に役に立たない監査レコードが入っていることによる混乱を避け、監査証跡の管理操作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、監査証跡を消去すること。
一度必要な情報を収集すると、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消去します。

データベース監査証跡ビューの作成および削除

ここでは、データベース監査証跡ビューの作成および削除方法について説明します。トピックは、次のとおりです。

- [監査証跡ビューの作成](#)
- [監査証跡ビューの削除](#)

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。この表の中には、重要な監査情報の参照に役立つように、事前定義のビューがいくつか用意されています。監査機能を使用するには、監査証跡を作成する必要があります。監査機能を使用しなくなった場合は、後から監査証跡を削除できます。

スクリプト CATALOG.SQL を実行すると、監査証跡ビューが自動的に作成されます。

監査証跡ビューの作成

監査機能を適用する場合は、SYS として接続し CATAUDIT.SQL スクリプトを実行します。このスクリプトにより、次のビューが作成されます。

- STMT_AUDIT_OPTION_MAP
- AUDIT_ACTIONS
- ALL_DEF_AUDIT_OPTS
- DBA_STMT_AUDIT_OPTS
- USER_OBJ_AUDIT_OPTS、DBA_OBJ_AUDIT_OPTS
- USER_AUDIT_TRAIL、DBA_AUDIT_TRAIL
- USER_AUDIT_SESSION、DBA_AUDIT_SESSION
- USER_AUDIT_STATEMENT、DBA_AUDIT_STATEMENT

- USER_AUDIT_OBJECT、DBA_AUDIT_OBJECT
- DBA_AUDIT_EXISTS
- USER_AUDIT_SESSION、DBA_AUDIT_SESSION
- USER_TAB_AUDIT_OPTS

関連項目：これらのビューの詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

監査情報の解釈例の詳細は、25-16 ページの「[データベース監査証跡情報の参照](#)」を参照してください。

監査証跡ビューの削除

監査を使用禁止にし、その監査証跡ビューが不要になった場合、SYS としてデータベースへ接続し、スクリプト・ファイル CATNOAUD.SQL を実行して、ビューを削除します。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。

監査証跡情報の管理

ここでは、監査証跡情報の管理について説明します。トピックは、次のとおりです。

- [デフォルトで監査されるイベント](#)
- [監査オプションの設定](#)
- [データベース監査を使用可能、使用禁止にする方法](#)
- [監査証跡の成長とサイズを制御する方法](#)
- [監査証跡を保護する方法](#)

監査されるイベントや設定する監査オプションによっては、監査証跡レコードに複数の異なるタイプの情報が含まれます。次の情報は、特定の監査アクションに対して意味のある場合に、常に各監査証跡レコードに含まれます。

- 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

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、判読不可能なコードが含まれています。これらのコードは次のように解読できます。

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記述しています。AUDIT_ACTIONS データ・ディクショナリ表に、これらのコードとその説明のリストが含まれています。

使用された権限

操作の実行に使用された権限を記述しています。SYSTEM_PRIVILEGE_MAP 表に、これらのコードとその説明がすべて記述されています。

完了コード

試行した操作の結果を記述しています。操作が正常に終了すると値 0 が戻り、失敗すると操作が失敗した原因を記述した Oracle エラー・コードが戻ります。

デフォルトで監査されるイベント

データベース監査が使用可能かどうかにかかわらず、Oracle では常にデータベースに関連する特定のアクションを監査し、オペレーティング・システム監査証跡に記録します。このようなイベントは次のとおりです。

インスタンスの起動	インスタンスを起動した OS 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は使用できないため、この監査レコードは OS 監査証跡に格納されます。起動時のデータベース監査の状態を記録しておくことで、未監査のアクションを管理者が実行できるように、データベース監査が使用禁止の状態のデータベースを再起動することを防止します。
インスタンス停止	インスタンスを停止した OS 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。
管理者権限によるデータベースへの接続	SYSOPER または SYSDBA として Oracle に接続した OS 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。SYSOPER または SYSDBA により、ユーザーのアカウントにシステム権限が付与されます。

Oracle が OS の監査証跡にアクセスできないオペレーティング・システムでは、これらの監査証跡レコードはバックグラウンド・プロセス・トレース・ファイルと同じディレクトリにある Oracle の監査証跡ファイルに記録されます。

監査オプションの設定

監査オプションの設定に応じて、監査レコードにさまざまなタイプの情報を記録できます。ただし、すべての監査オプションにより、次の情報が生成されます。

- 監査文を実行したユーザーに関する情報
- ユーザーによって実行された監査文を示すアクション・コード（コード番号）に関する情報
- 監査文で参照されたオブジェクトに関する情報
- 監査文が実行された日時に関する情報

監査証跡は、監査された文に関連するデータ値に関する情報は格納しません。たとえば、UPDATE 文の監査時には、更新した行の新旧のデータ値は格納されません。ただし、この特殊な監査機能は、データベース・トリガーを使用して、表に関連する DML 文で実行できます。

Oracle 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。

文	使用された SQL 文のタイプに基づいて監査します。たとえば、表に関する SQL 文を監査します（CREATE TABLE 文、TRUNCATE TABLE 文、DROP TABLE 文をそれぞれ記録します）。
権限	特定のシステム権限の使用を監査します。たとえば、CREATE TABLE を監査します。
オブジェクト	特定のオブジェクトに関する特定の文を監査します。たとえば、EMP 表に関する ALTER TABLE を監査します。

関連項目：この特殊な監査機能を実行するためのトリガーの使用方法的例は、25-19 ページの「[データベース・トリガーによる監査](#)」を参照してください。

文監査オプション

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、『Oracle8i SQL リファレンス』を参照してください。

文監査オプションのショートカット ショートカットは、1 語で複数の関連する文オプションを指定するために用意されています。

ショートカットとは、文オプション自体ではなく、AUDIT 文および NOAUDIT 文に、関連する複数の文オプションを 1 語で指定するためのものです。システム権限と文オプションのショートカットの詳細は、『Oracle8i SQL リファレンス』を参照してください。

接続と切断の監査

SESSION 文オプション（および CONNECT ショートカット）は、特殊なタイプの文の発行時には監査レコードを生成しないため、一意となります。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、論理 I/O や物理 I/O の処理などのセッションに関する蓄積情報は、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。

関連項目：ショートカットでカバーされないその他の監査オプションについては、『Oracle8i SQL リファレンス』を参照してください。

権限監査オプション

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。このオプションを有効にするには、次のような文を使用してください。

```
AUDIT DELETE ANY TABLE
  BY ACCESS
  WHENEVER NOT SUCCESSFUL;
```

Oracle のシステム権限のリストの詳細は、24-2 ページの「[システム権限](#)」を参照してください。

オブジェクト監査オプション

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、『Oracle8i SQL リファレンス』を参照してください。

オブジェクト監査オプションのショートカット ALL ショートカットを使用して、スキーマ・オブジェクトに対して指定可能なすべてのオブジェクト監査オプションを指定できます。このショートカットは、オプション自体ではなく、AUDIT 文および NOAUDIT 文ですべてのオブジェクト監査オプションを 1 語で指定する手段です。

AUDIT オプションを使用可能にする方法

SQL 文 AUDIT 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を使用可能にします。この文を使用して文オプションと権限オプションを設定するには、AUDIT SYSTEM 権限を持つ必要があります。この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 権限を持つ必要があります。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には、BY 句を含めて、ユーザーまたはアプリケーション代理のリストを指定し、文監査オプションと権限監査オプションの有効範囲を制限できます。

任意の監査オプションを設定し、監査機能として次の条件を指定できます。

- WHENEVER SUCCESSFUL/WHENEVER NOT SUCCESSFUL
- BY SESSION/BY ACCESS

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・ディクショナリの監査オプションを使用します。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・セッションでは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指定された監査オプションが引き続き使用されます。

警告： AUDIT 文を実行しても、監査オプションを指定するだけで、監査機能の全体が使用可能になるわけではありません。監査オプションを使用可能にして、現在設定されている監査オプションに基づいた、Oracle による監査レコードの生成を制御するには、データベースのパラメータ・ファイルにパラメータ AUDIT_TRAIL を設定します。

関連項目：AUDIT コマンドの詳細は、『Oracle8i SQL リファレンス』を参照してください。

監査を使用可能および使用禁止にすることの詳細は、25-12 ページの「[データベース監査を使用可能、使用禁止にする方法](#)」を参照してください。

文権限の監査を使用可能にする方法 ユーザーに関係なく、正常終了および異常終了したデータベースとのすべての接続と、正常終了および異常終了したデータベースからのすべての切断について、BY SESSION (このオプションのデフォルト値かつ唯一の値) を指定して監査するには、次の文を指定します。

```
AUDIT SESSION;
```

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。たとえば、次の例に示すとおりです。

```
AUDIT SESSION  
BY scott, lori;
```

DELETE ANY TABLE システム権限の正常な使用および異常終了した使用をすべて監査するためには、次の文を指定します。

```
AUDIT DELETE ANY TABLE;
```

すべての表でデータベース・ユーザーが異常終了した SELECT、INSERT および DELETE 文、および個々の監査文による EXECUTE PROCEDURE システム権限の異常終了を監査するには、次の文を指定します。

```
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の設定には、AUDIT SYSTEM システム権限が必要です。通常、セキュリティ管理者のみがこの権限を付与されている唯一のユーザーです。

オブジェクト監査を使用可能にする方法 BY SESSION (デフォルト値) によって、SCOTT.EMP 表に関して正常終了および異常終了した DELETE 文をすべて監査するには、次の文を指定します。

```
AUDIT DELETE ON scott.emp;
```

ユーザー JWARD が所有する DEPT 表に関して正常終了した SELECT、INSERT、DELETE の文をすべて監査するには (BY ACCESS)、次の文を指定します。

```
AUDIT SELECT, INSERT, DELETE  
ON jward.dept  
BY ACCESS  
WHENEVER SUCCESSFUL;
```

BY SESSION (デフォルト値) によって、異常終了した全 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設定するには、次の文を指定します。

```
AUDIT SELECT  
ON DEFAULT  
WHENEVER NOT SUCCESSFUL;
```

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、このシステム権限はセキュリティ管理者にのみ付与されています。

AUDIT オプションを使用禁止にする方法

NOAUDIT コマンドを使用して、Oracle の各種監査オプションの設定を解除します。文監査オプション、権限監査オプション、オブジェクト監査オプションの設定を解除するために、この監査オプションを使用します。文監査オプションや権限監査オプションを設定する NOAUDIT 文には、ユーザー・リストを指定する BY USER オプションを使用して、各監査オプションの範囲を制限できます。

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せん。この句を指定しなければ、正常終了と異常終了のどちらの場合に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 オプションは、NOAUDIT コマンドに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、監査オプションは適切な NOAUDIT 文によってその設定が解除されます。

警告： NOAUDIT 文を実行しても、監査オプションを指定するだけでは、監査機能の全体を使用禁止にすることはできません。監査オプションを使用禁止にして、現在監査オプションが設定されている状態で、Oracle による監査レコードの生成を停止するには、データベースのパラメータ・ファイルにパラメータ AUDIT_TRAIL を設定します。

関連項目：NOAUDIT コマンドの詳細は、『Oracle8i SQL リファレンス』を参照してください。

また、25-12 ページの「[データベース監査を使用可能、使用禁止にする方法](#)」も参照してください。

文監査と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、次の文を指定します。

```
NOAUDIT session;  
NOAUDIT session BY scott, lori;  
NOAUDIT DELETE ANY TABLE;  
NOAUDIT SELECT TABLE, INSERT TABLE, DELETE TABLE,  
EXECUTE PROCEDURE;
```

すべての文（システム）監査オプションおよび権限監査オプションを使用禁止にするには、次の文を指定します。

```
NOAUDIT ALL;  
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オフにするには、次の文を指定します。

```
NOAUDIT DELETE  
ON emp;  
NOAUDIT SELECT, INSERT, DELETE  
ON jward.dept;
```

さらに、EMP 表に関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、次の文を入力します。

```
NOAUDIT ALL
ON emp;
```

デフォルトのオブジェクト監査オプションを使用禁止にする方法 デフォルトのオブジェクト監査オプションをすべてオフにするには、次の文を入力します。

```
NOAUDIT ALL
ON DEFAULT;
```

なお、この NOAUDIT 文を発行する前に作成したすべてのスキーマ・オブジェクトは、そのスキーマ・オブジェクトの作成後に明示的に NOAUDIT 文で上書き済みでない限り、作成時に実施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なければなりません。なお、別のユーザーのスキーマ内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したり、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ためには、AUDIT ANY システム権限を持っている必要があります。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置き換えることができます。

データベース監査を使用可能、使用禁止にする方法

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ブジェクトを任意で設定できますが、データベース監査が使用可能な状態でなければ、Oracle は監査証跡に監査レコードを生成および格納しません。通常、セキュリティ管理者がこの操作を担当します。

データベース監査は、データベースのパラメータ・ファイル内の AUDIT_TRAIL 初期化パラメータで使用可能にしたり、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。

DB	データベース監査を使用可能にして、データベース監査証跡のためのすべての監査レコードを管理します。
OS	データベース監査を使用可能にして、オペレーティング・システム監査証跡のためにすべての監査レコードを管理します。
NONE	監査を使用禁止にします（この値はデフォルトです）。

パラメータ・ファイルを編集した後は、データベース監査を使用可能または使用禁止にするために、データベース・インスタンスを再起動してください。

関連項目：パラメータ・ファイルの編集の詳細は、『Oracle8i リファレンス・マニュアル』を参照してください。

監査証跡の成長とサイズを制御する方法

監査証跡が完全に満杯になり、これ以上監査レコードを挿入できなくなった場合は、証跡が消去されない限り、監査された文を正常に実行できません。監査された文を発行するユーザー全員に対して警告が戻されます。このため、セキュリティ管理者は必ず監査証跡の成長とサイズを制御しなければなりません。

監査を使用可能にして監査レコードを生成すると、次の 2 つの要因に応じ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
- 監査文の実行頻度

監査証跡の成長を制御するためには、次の方法を使用できます。

- データベース監査を使用可能および使用禁止にする方法。使用可能にすると、監査レコードが生成され監査証跡に格納されます。使用禁止の場合、監査レコードが生成されません。
- 使用可能となっている監査オプションを任意に選択可能にする方法。選択的な監査を実施すると、不要な監査情報は生成されず、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機能を厳密に制御する方法。これには、2 種類の方法があります。
 - セキュリティ管理者はすべてのオブジェクトを所有し、その AUDIT ANY システム権限を他のユーザーには付与しない。そのかわりに、すべてのスキーマ・オブジェクトは、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属するようにできます。
 - 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、CREATE SESSION 権限が対応するユーザーに付与されていない）スキーマに格納しておき、セキュリティ管理者のみが、AUDIT ANY システム権限を付与された唯一のユーザーとなる。

どちらの方法でも、オブジェクト監査についてはセキュリティ管理者が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、データベース作成中に事前に定義されます。デフォルトで、この表に各 10K のエクステンツを最高 99 個まで割り当てることができます。

監査証跡の成長とサイズを制御する手段として、SYS.AUD\$ を他の表領域に移動することはできません。しかし、SYS.AUD\$ 内のデフォルトの記憶領域パラメータ（INITIAL を除く）は修正できます。

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出力する場合の、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。

SYS.AUD\$ 記憶領域パラメータの詳細は、『Oracle8i リファレンス・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消去する方法

監査を使用可能にした後でも、セキュリティ管理者はレコードをデータベース監査証跡から削除して、監査証跡の領域を解放し、監査証跡を容易に管理できます。

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、次の文を指定します。

```
DELETE FROM sys.aud$;
```

また、EMP 表の監査の結果、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、次の文を指定します。

```
DELETE FROM sys.aud$  
WHERE obj$name='EMP';
```

監査証跡情報を履歴の目的でアーカイブする必要がある場合には、セキュリティ管理者は通常のデータベース表へ関連するレコードをコピーしたり ("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 が SYS.AUD\$ の DELETE 権限を付与しているユーザーに限り、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： 監査証跡が完全に満杯の状態で接続が監査中の場合（つまり、SESSION オプションを設定している場合）通常のユーザー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、データベースに接続できません。この場合には、セキュリティ管理者は必ず SYS として接続し（SYS による操作は監査されません）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：表エクスポートの詳細は、『Oracle8i ユーティリティ・ガイド』を参照してください。

監査証跡のサイズを削減する方法

データベース監査証跡からレコードが削除された後に、データベース表を使用すると、この表に割り当てられた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が表に対して多数のエクステンツを割り当てていても、そのうちの大半が使用されていなければ、データベース監査証跡用に割り当ててある領域は、次の手順に従って、削減できます。

1. 現在、監査証跡内に情報を保存する場合には、別のデータベース表にその情報をコピーするか、または EXPORT ユーティリティを使用してその情報をエクスポートします。
2. 管理者権限を使用して接続します。
3. TRUNCATE コマンドを使用して、SYS.AUD\$ を切り捨てます。
4. 手順 1 によって生成され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は、現在の監査証跡レコードを記録するのに必要なエクステンツの数だけ割り当てられます。

注意： 直接修正できる SYS オブジェクトは SYS.AUD\$ のみです。

監査証跡を保護する方法

なんらかの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、正確で完全な監査情報を保証してください。

データベースの監査証跡を未許可の削除処理から保護するため、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してなされた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\$ 表に対してオブジェクト監査オプションを設定した結果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。なお、この表自体は許可されていない使用に対して保護されています。監査証跡の最終的な保護基準として、管理者権限で接続しているときに実行された操作は、使用可能であれば、オペレーティング・システム監査証跡で監査されます。

関連項目： オペレーティング・システム監査証跡の可用性とその使用方法の詳細は、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。

データベース監査証跡情報の参照

ここでは、監査証跡情報の検査および解釈方法の具体例を説明します。トピックは、次のとおりです。

- アクティブな文監査オプションの記述
- アクティブな権限監査オプションの記述
- 特定のオブジェクトのアクティブなオブジェクト監査オプションの記述
- デフォルトのオブジェクト監査オプションの記述
- 監査レコードの記述
- AUDIT SESSION オプションの監査レコードの記述

次のような疑わしいアクティビティが存在する場合は、データベースを監査する必要があります。

- 許可なく変更されているデータベース・ユーザーに対するパスワード、表領域の設定および割当て制限。
- 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、頻発に発生しているデッドロックの回数。
- SCOTT のスキーマ内の EMP 表から削除されている行。

たとえば、ユーザー JWARD および SWILLIAMS がこの種の不正なアクションを行ったと疑われる場合、データベース管理者は次の文を（順番に）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;
```

ユーザー JWARD によって次の文が発行されます。

```
ALTER USER tsmith QUOTA 0 ON users;
DROP USER djones;
```

ユーザー SWILLIAMS によって次の文が発行されます。

```
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);
```

次に、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を示します。

アクティブな文監査オプションの記述

次の問合せの結果、設定されている文監査オプションがすべて戻されます。

```
SELECT * FROM sys.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

このビューにより、文監査オプションが正常終了または異常終了（あるいはその両方）のどちらかに設定され、それぞれ BY SESSION または BY ACCESS のどちらかに設定されているかなどの設定が明らかになります。

アクティブな権限監査オプションの記述

次の問合せの結果、設定されている権限がすべて戻されます。

```
SELECT * FROM sys.dba_priv_audit_opts;
```

USER_NAME	PRIVILEGE	SUCCESS	FAILURE
ALTER USER	BY SESSION	BY SESSION	

特定のオブジェクトのアクティブなオブジェクト監査オプションの記述

次の問合せの結果、SCOTT のスキーマ内に収録されたオブジェクトに対する監査オプションがすべて戻されます。

```
SELECT * FROM sys.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	-/-	-/-	...

このビューにより、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戻されます。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、"/" で区切られた WHENEVER SUCCESSFUL と WHENEVER NOT SUCCESSFUL の 2 つの設定があります。たとえば、SCOTT.EMP に対する DELETE 監査オプションは、正常終了した削除に対して BY ACCESS を設定し、異常終了した削除文に対しては何も設定していません。

デフォルトのオブジェクト監査オプションの記述

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戻します。

```
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 -/- -/- -/- -/-
```

このビューにより、USER_OBJ_AUDIT_OPTS と DBA_OBJ_AUDIT_OPTS の各ビューに類似する情報が戻されます（先の例を参照してください）。

監査レコードの記述

次の問合せの結果、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一覧で表示されます。

```
SELECT * FROM sys.dba_audit_object;
```

AUDIT SESSION オプションの監査レコードの記述

次の問合せの結果、AUDIT SESSION 文監査オプションに対応する監査オプションが一覧で表示されます。

```
SELECT username, logoff_time, logoff_lread, logoff_pread,
       logoff_lwrite, logoff_dlock
FROM sys.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

データベース・トリガーによる監査

トリガーを使用して、Oracle の組み込まれた監査機能を補足できます。トリガーを使用して、AUDIT コマンドで書き込んだ情報と同じようなレコード情報を書き込むことができますが、これはより詳細な監査情報が必要な場合にのみ使用します。たとえば、トリガーを使用して、表の行単位に値ベースで監査できます。

注意： いくつかの分野では、Oracle の AUDIT コマンドはセキュリティ監査機能と見なされますが、トリガーは財務監査機能を提供できます。

データベース・アクティビティを監査するトリガーを作成するかどうかを決めるときには、トリガーによる監査機能と比較し、標準 Oracle データベースの監査機能による利点を考慮してください。

- 標準監査オプションは、すべてのタイプのスキーマ・オブジェクトと構造に関する DML と DDL 文の監査が可能です。
- データベースの監査情報はすべて中央に記録され、Oracle の監査機能に自動的に使用されます。
- 標準 Oracle の機能を使用して使用可能となる監査機能は、宣言と保守が簡単であり、トリガーが定義する監査機能と比較しても、エラーは発生しません。
- 既存の監査オプションの変更を監査して、不当なデータベース・アクティビティを防止できます。
- データベースの監査機能を使用して、監査文が発行されるたび (BY ACCESS)、または監査文を発行するセッションごとに (BY SESSION)、レコードを作成できます。トリガーはセッションごとには監査できません。トリガー監査表が参照されるたびに、監査レコードが作成されます。

- データベース監査では、失敗したデータ・アクセスを監査できます。ただし、トリガー文がロールバックされると、トリガーが生成した監査情報もロールバックされます。
- 標準データベース監査機能を使用して、接続と切断、およびセッション・アクティビティ（物理 I/O、論理 I/O およびデッドロックなど）を記録できます。

トリガーを使用して高度な監査を実行するには、通常 AFTER トリガーを使用します。AFTER トリガーによって、整合性制約に従うトリガー文の後に、監査情報を記録できます。監査処理において、整合性制約の例外を生成する文の無意味な実行を防止します。

AFTER 行と AFTER 文トリガーは、監査情報に応じて使い分ける必要があります。たとえば、表の中の行の値を行単位で監査するには、行トリガーが便利です。トリガーを使用して、監査済み SQL 文を発行して「理由コード」を出力できます。これは、行と文レベルの監査状況の両方に便利です。

次は、EMP 表に対する修正を行単位で監査するトリガーの例です。この例では、更新前に、「理由コード」をグローバル・パッケージ変数に格納する必要があります。このトリガーでは、次のことが例示されます。

- トリガーを使用して値ベースの監査を実行する方法
- パブリック・パッケージ変数の使用方法

コード内のコメントには、トリガーの機能が記述されています。

```
CREATE TRIGGER audit_employee
AFTER INSERT OR DELETE OR UPDATE ON emp
FOR EACH ROW
BEGIN
/* AUDITPACKAGE is a package with a public package
   variable REASON. REASON could be set by the
   application by a command such as EXECUTE
   AUDITPACKAGE.SET_REASON(reason_string). Note that a
   package variable has state for the duration of a
   session and that each session has a separate copy of
   all package variables. */
IF auditpackage.reason IS NULL THEN
   raise_application_error(-20201,'Must specify reason with ',
   'AUDITPACKAGE.SET_REASON(reason_string)');
END IF;

/* If the above conditional evaluates to TRUE, the
   user-specified error number and message is raised,
   the trigger stops execution, and the effects of the
   triggering statement are rolled back. Otherwise, a
   new row is inserted into the pre-defined auditing
   table named AUDIT_EMPLOYEE containing the existing
   and new values of the EMP table and the reason code
   defined by the REASON variable of AUDITPACKAGE. Note
   that the "old" values are NULL if triggering
   statement is an INSERT and the "new" values are NULL
```

```
        if the triggering statement is a DELETE. */
INSERT INTO audit_employee VALUES
    (:old.ssn, :old.name, :old.job_classification, :old.sal,
     :new.ssn, :new.name, :new.job_classification, :new.sal,
     auditpackage.reason, user, sysdate );
END;
```

更新ごとに理由コードを強制的に設定する場合、任意に理由コードを NULL に設定し直すこともできます。次の AFTER 文トリガーは、トリガー文の実行後に、理由コードを NULL に設定し直します。

```
CREATE TRIGGER audit_employee_reset
    AFTER INSERT OR DELETE OR UPDATE ON emp
BEGIN
    auditpackage.set_reason(NULL);
END;
```

上記のトリガーは2つとも同じタイプの SQL 文によって実行されます。ただし、トリガー文の実行が終了している場合は、AFTER 文トリガーは一度しか実行されないのに対し、AFTER 行トリガーは、トリガー文が処理対象とする表の各行ごとに一度実行されます。

A

- ADD LOGFILE MEMBER オプション
 - ALTER DATABASE コマンド, 6-12
- ADD LOGFILE オプション
 - ALTER DATABASE コマンド, 6-11
- ADD PARTITION 句
 - ALTER TABLE コマンド, 13-11
- ADMIN OPTION
 - 説明, 24-11
 - 取消し, 24-13
- admin_tables プロシージャ, 19-3, 19-12
- AFTER トリガー
 - 監査, 25-20
- ALERT ファイル
 - 位置, 4-11
 - 書き込む時期, 4-11
 - サイズ, 4-11
 - 使用, 4-10
 - セッション高水位標, 23-6
 - 説明, 4-10
- ALL_INDEXES ビュー
 - データを入れる, 20-5
- ALL_TAB_COLUMNS ビュー
 - データを入れる, 20-5
- ALL_TABLES ビュー
 - データを入れる, 20-5
- ALTER CLUSTER コマンド
 - ALLOCATE EXTENT オプション, 17-9
 - MAXTRANS オプション, 12-9
 - 索引クラスタに対して使用, 17-8
 - ハッシュ・クラスタに使用, 18-8
- ALTER DATABASE コマンド
 - DROP LOGFILE MEMBER オプション, 6-12
 - ADD LOGFILE オプション, 6-11
 - ARCHIVELOG オプション, 7-6
 - CLEAR LOGFILE オプション, 6-17
 - CLEAR UNARCHIVED LOGFILE オプション, 6-7
 - DATAFILE...OFFLINE DROP オプション, 10-8
 - DROP LOGFILE MEMBER オプション, 6-15
 - DROP LOGFILE オプション, 6-14
 - MOUNT オプション, 3-7
 - NOARCHIVELOG オプション, 7-6
 - OPEN オプション, 3-7
 - RENAME FILE オプション
 - 複数の表領域のデータ・ファイル, 10-10
 - UNRECOVERABLE DATAFILE オプション, 6-17
 - ユーザーが部分的に使用可能なデータベース, 3-7
- ALTER FUNCTION コマンド
 - COMPILE オプション, 20-24
- ALTER INDEX COALESCE, 16-6
- ALTER INDEX コマンド, 13-17
 - MAXTRANS オプション, 12-9
- MOVE PARTITION 句, 13-10
- REBUILD PARTITION 句, 13-10, 13-19
- 説明, 16-12
- ALTER PACKAGE コマンド
 - COMPILE オプション, 20-24
- ALTER PROCEDURE コマンド
 - COMPILE オプション, 20-24
- ALTER PROFILE コマンド
 - COMPOSITE_LIMIT オプション, 23-18
 - リソース制限の変更, 23-18
- ALTER RESOURCE COST コマンド, 23-19
- ALTER ROLE コマンド
 - 認可方法の変更, 24-9
- ALTER ROLLBACK SEGMENT コマンド
 - OFFLINE オプション, 21-12
 - ONLINE オプション, 21-11, 21-12
 - PUBLIC オプション, 21-9

- STORAGE 句, 21-9
 - STORAGE パラメータの変更, 21-9
- ALTER SEQUENCE コマンド, 15-10
- ALTER SESSION コマンド
 - SET SQL_TRACE パラメータ, 4-10
- ALTER SYSTEM RESUME, 3-12
- ALTER SYSTEM SUSPEND, 3-8
- ALTER SYSTEM コマンド
 - ARCHIVE LOG ALL オプション, 7-10
 - ARCHIVE LOG オプション, 7-9
 - ENABLE RESTRICTED SESSION オプション, 3-9
 - SET LICENSE_MAX_SESSIONS オプション, 23-4
 - SET LICENSE_MAX_USERS オプション, 23-5
 - SET LICENSE_SESSIONS_WARNING オプション, 23-4
 - SET MTS_DISPATCHERS オプション, 4-7
 - SET MTS_SERVERS オプション, 4-6
 - SET RESOURCE_LIMIT オプション, 23-20
 - SWITCH LOGFILE オプション, 6-16
- ALTER TABLESPACE コマンド
 - ADD DATAFILE パラメータ, 10-5
 - ONLINE オプション
 - 例, 9-10
 - READ ONLY オプション, 9-12
 - READ WRITE オプション, 9-13
 - RENAME DATA FILE オプション, 10-9
- ALTER TABLE コマンド
 - ADD PARTITION 句, 13-11
 - ALLOCATE EXTENT オプション, 14-11
 - DISABLE ALL TRIGGERS オプション, 20-12
 - DISABLE 整合性制約オプション, 20-19
 - DROP PARTITION 句, 13-11
 - DROP 整合性制約オプション, 20-20
 - ENABLE ALL TRIGGERS オプション, 20-12
 - ENABLE 整合性制約オプション, 20-19
 - MAXTRANS オプション, 12-9
 - MODIFY PARTITION 句, 13-10
 - SPLIT PARTITION 句, 13-11, 13-16
 - TRUNCATE PARTITION 句, 13-14
 - 例, 14-11
- ALTER TRIGGER コマンド
 - DISABLE オプション, 20-12
 - ENABLE オプション, 20-11
- ALTER USER 権限, 23-14
- ALTER VIEW コマンド
 - COMPILE オプション, 20-23
- ANALYZE TABLE VALIDATE STRUCTURE, 19-3

- ANALYZE コマンド
 - CASCADE オプション, 20-8
 - COMPUTE STATISTICS オプション, 20-6
 - ESTIMATE STATISTICS SAMPLE オプション, 20-7
 - LIST CHAINED ROWS オプション, 20-8
 - STATISTICS オプション, 20-4
 - VALIDATE STRUCTURE オプション, 20-8
 - 共有 SQL, 20-7
- ARCHIVE LOG オプション
 - ALTER SYSTEM コマンド, 7-9
- ARCHIVE LOG コマンド
 - LIST オプション, 6-14
- ARCHIVELOG モード, 7-4, 7-6
 - アーカイブ, 7-4
 - 切替え, 7-6
 - 手動アーカイブ, 7-5
 - 使用可能, 7-6
 - 実行, 7-4
 - 自動アーカイブ, 7-5
 - 長所, 7-4
 - 定義, 7-4
 - データ・ファイルのオフラインとオンラインの切替え, 10-8
 - 分散データベース, 7-6
- ARCH プロセス
 - 複数プロセスの指定, 7-19
- AUDIT_TRAIL パラメータ
 - 設定, 25-12
- AUDIT コマンド, 25-8
 - システム権限, 25-9
 - スキーマ・オブジェクト, 25-10
 - 文監査, 25-9

B

- BACKGROUND_DUMP_DEST パラメータ, 4-11

C

- CASCADE オプション
 - 一意キーまたは主キーの削除時, 20-19
 - 整合性制約, 17-10
- CATAUDIT.SQL
 - 実行, 25-4
- CATNOAUD.SQL
 - 実行, 25-5
- CHAR データ型

- 使用領域, 12-16
- 列の長さの拡張, 14-10
- check_object プロシージャ, 19-3, 19-7
- CHECKPOINT_PROCESS パラメータ
 - 設定, 4-12
- CHECK 制約, 20-18
- CKPT, 4-12
- CLEAR LOGFILE オプション
 - ALTER DATABASE コマンド, 6-17
- COMPUTE STATISTICS オプション, 20-6
- CONNECT ロール, 24-7
- CONTROL_FILES パラメータ
 - 既存の制御ファイルの上書き, 2-9
 - 設定
 - データベース作成前, 2-9, 5-4
 - 名前, 5-2
- CREATE CLUSTER コマンド
 - HASH IS オプション, 18-6
 - HASHKEYS オプション, 18-7
 - SIZE オプション, 18-6
 - ハッシュ・クラスタ, 18-4
 - 例, 17-6
- CREATE CONTROLFILE コマンド
 - NORESETLOGS オプション, 5-7
 - RESETLOGS オプション, 5-7
 - 不整合の検出, 5-8
 - 説明, 5-5
- CREATE DATABASE コマンド
 - CONTROLFILE REUSE オプション, 5-4
 - MAXLOGFILES オプション, 6-10
 - MAXLOGMEMBERS オプション, 6-10
 - 例, 2-7
- CREATE INDEX コマンド
 - ON CLUSTER オプション, 17-7
 - 回復不能 (UNRECOVERABLE), 16-5
 - 制約の指定, 16-7
 - 明示的, 16-8
- CREATE PROFILE コマンド
 - COMPOSITE_LIMIT オプション, 23-18
 - 説明, 23-17
- CREATE ROLE コマンド
 - IDENTIFIED BY オプション, 24-8
 - IDENTIFIED EXTERNALLY オプション, 24-8
- CREATE ROLLBACK SEGMENT コマンド
 - 説明, 21-7
 - チューニングのガイドライン, 2-14
- CREATE SCHEMA コマンド

- 必要な権限, 20-2
- 複数の表とビュー, 20-2
- CREATE SEQUENCE コマンド, 15-9
- CREATE SYNONYM コマンド, 15-11
- CREATE TABLESPACE コマンド
 - 例, 9-4
 - データ・ファイル名, 9-4
- CREATE TABLE コマンド
 - CLUSTER オプション, 17-7
 - PARTITION 句, 13-8
 - 回復不能 (UNRECOVERABLE), 14-4
 - 説明, 14-9
- CREATE USER コマンド
 - IDENTIFIED BY オプション, 23-11
 - IDENTIFIED EXTERNALLY オプション, 23-11
- CREATE VIEW コマンド
 - OR REPLACE オプション, 15-8
 - WITH CHECK OPTION, 15-3
 - 説明, 15-2

D

- DATE データ型, 12-17
- DB_BLOCK_BUFFERS パラメータ
 - データベース作成前の設定, 2-11
- DB_BLOCK_CHECKING パラメータ, 19-3
- DB_BLOCK_CHECKSUM, 10-12
- DB_BLOCK_SIZE パラメータ
 - 作成前の設定, 2-10
 - データベース・バッファのキャッシュ・サイズ, 2-11
- DB_DOMAIN パラメータ
 - データベース作成前の設定, 2-9
- DB_NAME パラメータ
 - データベース作成前の設定, 2-9
- DB_VERIFY ユーティリティ, 19-3
- DBA, 1-2
- DBA_DATA_FILES, 9-30, 10-13
- DBA_EXTENTS, 10-13
- DBA_FREE_SPACE, 9-30, 10-13
- DBA_FREE_SPACE_COALESCED ビュー, 9-9
- DBA_INDEXES ビュー
 - データを入れる, 20-5
- DBA_ROLLBACK_SEGS ビュー, 21-14, 21-15
- DBA_SEGMENTS, 9-30, 10-13
- DBA_TAB_COLUMNS ビュー
 - データを入れる, 20-5

DBA_TABLESPACES, 9-30, 10-13
DBA_TABLESPACES ビュー, 9-15
DBA_TABLES ビュー
 データを入れる, 20-5
DBA_TS_QUOTAS, 9-30, 10-13
DBA_USERS, 9-30, 10-13
DBA ロール, 1-5, 24-7
DBMS_JOB パッケージ
 REMOVE プロシージャ, 8-11
 ジョブ・キュー, 8-3
 ジョブの強制的な実行, 8-14
 ジョブの変更, 8-11
 ジョブを送る方法, 8-4
DBMS_LOGMNR.D.BUILD パッケージ, 7-26, 7-27
DBMS_LOGMNR.ADD_LOGFILE パッケージ
 LogMiner, 7-27
DBMS_LOGMNR.START_LOGMNR パッケージ
 LogMiner, 7-28
DBMS_REPAIR パッケージ, 19-1
DBMS_RESOURCE_MANAGER_PRIVS パッケージ,
 11-9
DBMS_RESOURCE_MANAGER パッケージ, 11-3
DBMS_SESSION パッケージ, 11-10
DBMS_UTILITY.ANALYZE_SCHEMA()
 実行, 20-7
DEFAULT_CONSUMER_GROUP, 11-9
DROP CLUSTER コマンド
 CASCADE CONSTRAINTS オプション, 17-10
 INCLUDING TABLES オプション, 17-10
 削除
 ハッシュ・クラスタ, 18-9
 表を含まないクラスタ, 17-10
DROP LOGFILE MEMBER オプション
 ALTER DATABASE コマンド, 6-15
DROP LOGFILE オプション
 ALTER DATABASE コマンド, 6-14
DROP PARTITION 句
 ALTER TABLE コマンド, 13-11
DROP PROFILE コマンド, 23-20
DROP ROLE コマンド, 24-10
DROP ROLLBACK SEGMENT コマンド, 21-14
DROP SYNONYM コマンド, 15-11
DROP TABLESPACE コマンド, 9-15
DROP TABLE コマンド
 CASCADE CONSTRAINTS オプション, 14-12
 クラスタ化された表のための, 17-10
 説明, 14-11

DROP USER 権限, 23-16
DROP USER コマンド, 23-16
dump_orphan_keys プロシージャ, 19-6, 19-9

E

Enterprise Manager
 オペレーティング・システム・アカウント, 1-4
ESTIMATE STATISTICS オプション, 20-6
EXP_FULL_DATABASE ロール, 24-7

F

fix_corrupt_blocks プロシージャ, 19-5, 19-7
FOREIGN KEY 制約
 使用可能, 20-18

G

GRANT OPTION
 説明, 24-12
 取消し, 24-14
GRANT コマンド
 ADMIN オプション, 24-11
 GRANT オプション, 24-12
 SYSOPER/SYSDBA 権限, 1-12
 オブジェクト権限, 24-11
 システム権限とロール, 24-10
 実施されるとき, 24-16

H

HOST コマンド
 SQL*Plus, 6-13

I

IMP_FULL_DATABASE ロール, 24-7
INITIAL 記憶領域パラメータ, 12-7
 変更, 14-10
INITRANS 記憶領域パラメータ
 設定するためのガイドライン, 12-9
 デフォルト, 12-8
 トランザクション・エントリ, 12-8
 変更, 14-10
INSERT 権限
 取消し, 24-14

付与, 24-12
INTERNAL
 OSOPER と OSDBA, 1-8
 セキュリティ, 22-7
 代替, 1-8
 停止のための接続, 3-10
INTERNAL 日付関数
 ジョブの実行, 8-8
I/O
 分散, 2-15
I/O の分散, 2-15

J

JQ ロック, 8-9

L

LGWR, 4-11
LICENSE_MAX_SESSIONS パラメータ
 インスタンスの稼働中に変更, 23-4
 設定, 23-4
 データベース作成前の設定, 2-12
LICENSE_MAX_USERS パラメータ
 設定, 23-5
 データベース作成前の設定, 2-12
 データベースの稼働中に変更, 23-5
LICENSE_SESSION_WARNING パラメータ
 データベース作成前の設定, 2-12
LICENSE_SESSIONS_WARNING パラメータ
 インスタンスの稼働中に変更, 23-4
 設定, 23-4
LIST CHAINED ROWS オプション, 20-8
LOG_ARCHIVE_BUFFER_SIZE 初期化パラメータ,
 7-21
LOG_ARCHIVE_BUFFERS パラメータ
 設定, 7-21
LOG_ARCHIVE_BUFFERS 初期化パラメータ, 7-21
LOG_ARCHIVE_DEST_n 初期化パラメータ, 7-10
 REOPEN オプション, 7-18
LOG_ARCHIVE_DEST 初期化パラメータ
 アーカイブ先の指定に使用, 7-10
LOG_ARCHIVE_DUPLEX_DEST 初期化パラメータ
 アーカイブ先の指定に使用, 7-10
LOG_ARCHIVE_MAX_PROCESSES 初期化パラメータ,
 7-19
LOG_ARCHIVE_MIN_SUCCEED_DEST 初期化パラ

メータ, 7-16
LOG_ARCHIVE_START 初期化パラメータ, 7-8
 アーカイブ先のパラメータ・エラー状態, 7-13
 設定, 7-9
LOG_BLOCK_CHECKSUM 初期化パラメータ
 REDO ブロックのチェックの使用可能化, 6-16
LOG_FILES 初期化パラメータ
 ログ・ファイル数, 6-10
LogMiner, 7-24
LogMiner ユーティリティ, 7-24, 7-30
 アーカイブ済み REDO ログの分析に使用, 7-24,
 7-29
 使用, 7-27, 7-28
 ディクショナリ・ファイル, 7-25
LONG データ型, 12-17

M

MAX_DUMP_FILE_SIZE パラメータ, 4-11
MAX_ENABLED_ROLES パラメータ
 デフォルト・ロール, 24-9
 ロールを使用可能にする, 24-9
MAXDATAFILES パラメータ
 変更, 5-5
MAXEXTENTS 記憶領域パラメータ
 説明, 12-8
 データ・ディクショナリに合わせて設定, 20-27
MAXINSTANCES パラメータ
 変更, 5-5
MAXLOGFILES オプション
 CREATE DATABASE コマンド, 6-10
MAXLOGFILES パラメータ
 変更, 5-5
MAXLOGHISTORY
 変更, 5-5
MAXLOGMEMBERS オプション
 CREATE DATABASE コマンド, 6-10
MAXLOGMEMBERS パラメータ
 変更, 5-5
MAXTRANS 記憶領域パラメータ
 設定するためのガイドライン, 12-9
 デフォルト, 12-9
 トランザクション・エントリ, 12-9
 変更, 14-10
MINEXTENTS 記憶領域パラメータ
 説明, 12-8
 変更, 14-10

MODIFY PARTITION 句
 ALTER TABLE コマンド, 13-10
MOVE PARTITION 句
 ALTER TABLE コマンド, 13-10
MTS_DISPATCHERS パラメータ
 初期設定, 4-5

N

Net8
 アーカイブ・ログの送信に使用, 7-15
 サービス名, 7-15
NEXT 記憶領域パラメータ, 12-8
 データ・ディクショナリに合わせて設定, 20-27
NOARCHIVELOG モード
 アーカイブ, 7-4
 切替え, 7-6
 実行, 7-4
 定義, 7-4
 データ・ファイルのオフライン化, 10-8
 ホット・バックアップなし, 7-4
 メディア障害, 7-4
NOAUDIT コマンド
 監査オプションを使用禁止にする, 25-10
 権限, 25-11
 スキーマ・オブジェクト, 25-11
 文, 25-11
NOT NULL 制約, 20-18
NUMBER データ型, 12-17

O

OPTIMAL 記憶領域パラメータ, 21-6
Oracle8i
 インストール, 1-17
Oracle8i Server
 プロセス
 オペレーティング・システム名, 4-9
 監視, 4-8
 チェックポイント (CKPT), 4-12
 トレース・ファイル fpr, 4-10
 ライセンス契約の遵守, 23-2
 リリースの識別, 1-20
Oracle8i Server プロセス
 プロセス
 識別と管理, 4-7
 専用サーバー・プロセス, 4-2

Oracle ブロック, 2-10
ORAPWD ユーティリティ, 1-9
OS_ROLES パラメータ
 REMOTE_OS_ROLES, 24-20
 オペレーティング・システム認可, 24-8
 使用, 24-18
OS 認証, 1-7

P

Parallel Server
 ALTER CLUSTER..ALLOCATE EXTENT, 17-9
 V\$THREAD ビュー, 6-18
 アーカイブ用のスレッド指定, 7-10
 インスタンスのデータ・ファイル数の上限, 10-3
 順序番号, 15-10
 セッションと警告制限, 23-3
 独自のロールバック・セグメント, 21-3
 名前付きユーザー, 2-13
 名前付きユーザーでの制限, 23-5
 ライセンス・セッションの制限, 2-12
 オンライン REDO ログのスレッド, 6-2
PARALLEL_MAX_SERVERS パラメータ, 4-13
PARALLEL_MIN_SERVERS パラメータ, 4-13
PARALLEL_SERVER_IDLE_TIME パラメータ, 4-13
PARTITION 句
 CREATE TABLE コマンド, 13-8
PCTFREE 記憶領域パラメータ
 クラスタ化されていない表, 12-4
 PCTUSED, 12-6
 機能, 12-2
 クラスタ化された表, 12-4
 索引, 12-4
 設定するためのガイドライン, 12-3
 デフォルト, 12-3
 ブロック・オーバーヘッド, 12-6
 変更, 14-10
PCTINCREASE 記憶領域パラメータ
 説明, 12-8
 変更, 12-11
 データ・ディクショナリに合わせて設定, 20-27
PCTUSED 記憶領域パラメータ
 PCTFREE, 12-6
 機能, 12-4
 設定するためのガイドライン, 12-5
 デフォルト, 12-5
 ブロック・オーバーヘッド, 12-6

- 変更, 14-10
- PL/SQL プログラム・ユニット
 - 置き換えられたビュー, 15-9
 - 削除した表, 14-12
- PRIMARY KEY 制約
 - 削除時の外部キー参照, 20-19
 - 作成時に使用可能にする, 16-7
 - 使用可能, 20-18
 - 使用禁止, 20-18
 - 対応付けられる索引, 16-7
 - 対応付けられる索引の記憶領域, 16-7
 - 対応する索引の削除, 16-14
- PROCESSES パラメータ
 - データベース作成前の設定, 2-11
- PUBLIC_DEFAULT プロファイル
 - 使用, 23-17
 - プロファイルの削除, 23-20
- PUBLIC ユーザー・グループ
 - プロシージャ, 24-16
 - 権限の付与と取消し, 24-16

R

- REBUILD PARTITION 句
 - ALTER INDEX コマンド, 13-10, 13-19
- rebuild_freelists プロシージャ, 19-6, 19-10
- REDO エントリ
 - REDO レコードを参照
 - 内容, 6-2
- REDO レコード, 6-2
- REDO ログ・バッファ
 - 書込み, 6-3
- REDO ログ・ファイル
 - LGWR, 6-3
 - REDO エントリ, 6-2
 - REDO ログ内の数, 6-9
 - アーカイブされた
 - 長所, 7-2
 - 内容, 7-2
 - ログ・スイッチ, 6-5
 - アーカイブ済み REDO ログ, 7-4
 - アーカイブ済み REDO ログ・ファイル, 7-6
 - アクティブ (現行), 6-4
 - オンライン, 6-2
 - 2 つは必要, 6-3
 - 回復時の使用, 6-2
 - スレッド, 6-2

- オンライン REDO ログ, 6-1
- グループ, 6-6
 - LOG_FILES 初期化パラメータ, 6-10
 - 数の削減, 6-10
 - 削除, 6-14
 - 作成, 6-11
 - スレッド, 6-2
 - メンバー, 6-6
- 計画, 6-5 ~ 6-10
- 権限
 - グループとメンバーの追加, 6-11
- 作成
 - グループとメンバー, 6-11
- 初期化, 6-7, 6-17
 - 制限, 6-17
- 使用可能, 6-3
- 循環使用, 6-3
- 多重化, 6-5
 - 一部のメンバーがアクセス不能の場合, 6-7
 - グループ, 6-6
 - すべてがアクセス不能な場合, 6-7
 - 図, 6-6
- 内容, 6-2
- 非アクティブ, 6-4
- 表示, 2-8
- ブロックの検証, 6-16
- 分散トランザクション情報, 6-3
- ミラー化
 - ログ・スイッチ, 6-7
- メンバー, 6-6
 - 最大数, 6-10
 - 削除, 6-14
 - 作成, 6-11
- メンバーの作成, 6-12
- 有効な構成と無効な構成, 6-7
- 要件, 6-7
- ログ順序番号, 6-5
- ログ・スイッチ, 6-5
- REDO ログ・ファイルの初期化, 6-7, 6-17
 - 制限, 6-17
- REFERENCES 権限
 - CASCADE CONSTRAINTS オプション, 24-14
 - 取消し, 24-14
- REMOTE_LOGIN_PASSWORDFILE パラメータ, 1-11
- REMOTE_OS_AUTHENT パラメータ
 - 設定, 23-9
- REMOTE_OS_ROLES パラメータ

- 設定, 24-9, 24-20
- RENAME コマンド, 20-2
- REOPEN オプション
 - LOG_ARCHIVE_DEST_n 初期化パラメータ, 7-18
- RESOURCE_LIMIT パラメータ
 - 制限を使用可能および使用禁止にする, 23-20
- RESOURCE ロール, 24-7
- RESTRICTED SESSION 権限
 - 制限モード, 3-4
 - 制限モードのインスタンス, 3-8
 - セッション制限, 23-3
- REVOKE コマンド, 24-13
 - 実施されるとき, 24-16
- ROLLBACK_SEGMENTS パラメータ
 - データベース作成前の設定, 2-12
 - ロールバック・セグメントの追加, 21-8

S

- SCN, 10-13
- SEQUENCE_CACHE_ENTRIES パラメータ, 15-10
- SET ROLE コマンド
 - オペレーティング・システム・ロールを使用するとき, 24-19
 - パスワードの設定方法, 24-8
- SET TRANSACTION コマンド
 - USE ROLLBACK SEGMENT オプション, 21-13
- SGA
 - キャッシュのバッファ数の決定, 2-11
- SHUTDOWN コマンド
 - ABORT オプション, 3-12
 - IMMEDIATE オプション, 3-11
 - NORMAL オプション, 3-11
- skip_corrupt_blocks プロシージャ, 19-5, 19-11
- SNP バックグラウンド・プロセス
 - 説明, 8-2
- SORT_AREA_SIZE パラメータ
 - 索引作成, 16-3
- SPLIT PARTITION 句, 13-17
 - ALTER INDEX コマンド, 13-17
 - ALTER TABLE コマンド, 13-11, 13-16
- SQL*Loader
 - 索引, 16-3
 - 説明, 1-16
- SQL*Plus コマンド
 - コマンド、SQL*Plus を参照
- SQL_TRACE パラメータ

- トレース・ファイル, 4-10
- SQL トレース機能
 - 使用可能にするとき, 4-12
- SQL 文
 - 監査オプションを使用可能にする, 25-9
 - 監査オプションを使用禁止にする方法, 25-11
- STALE 状態
 - REDO ログ・メンバー, 6-15
- STARTUP コマンド, 3-2
 - FORCE オプション, 3-5
 - MOUNT オプション, 3-4
 - NOMOUNT オプション, 2-6, 3-3
 - RECOVER オプション, 3-5
 - データベース名の指定, 3-3
- SWITCH LOGFILE オプション
 - ALTER SYSTEM コマンド, 6-16
- SYS
 - 権限, 1-5
 - 初期パスワード, 1-4
 - 所有オブジェクト, 1-5
 - 保護の方針, 22-7
 - ユーザー, 1-5
- SYS.AUD\$
 - 監査証跡, 25-2
 - 作成と削除, 25-4
- SYSOPER/SYSDBA 権限
 - 権限所有者の判別, 1-13
 - 接続, 1-13
 - パスワード・ファイルへのユーザーの追加, 1-11
 - 付与と取消し, 1-12
- SYSTEM
 - 初期パスワード, 1-4
 - 所有オブジェクト, 1-5
 - 保護の方針, 22-7
 - ユーザー, 1-5
- SYSTEM 表領域
 - オフラインにすることの制限, 10-7
 - 削除できない, 9-14
 - 作成されるとき, 9-3
 - 初期ロールバック・セグメント, 21-2
 - 非データ・ディクショナリ表, 14-3
- SYSTEM ロールバック・セグメント
 - 記憶領域パラメータの変更, 21-9

T

- TNSNAMES.ORA ファイル, 7-11

TRANSACTIONS パラメータ
 使用, 21-2
TRANSACTIONS_PER_ROLLBACK_SEGMENT パラメータ
 使用, 21-2
TRUNCATE PARTITION 句
 ALTER TABLE コマンド, 13-14
TRUNCATE コマンド, 20-9
 DROP STORAGE オプション, 20-10
 REUSE STORAGE オプション, 20-10

U

UNIQUE キー制約
 削除時の外部キー参照, 20-19
 作成時に使用可能にする, 16-7
 使用可能, 20-18
 使用禁止, 20-18
 対応付けられる索引, 16-7
 対応付けられる索引の記憶領域, 16-7
 対応する索引の削除, 16-14
UNLIMITED TABLESPACE 権限, 23-13
UNRECOVERABLE DATAFILE オプション
 ALTER DATABASE コマンド, 6-17
UPDATE 権限
 取消し, 24-14
USER_DUMP_DEST パラメータ, 4-11
USER_EXTENTS, 10-13
USER_FREE, 9-30, 10-13
USER_INDEXES ビュー
 データを入れる, 20-5
USER_SEGMENTS, 9-30, 10-13
USER_TAB_COLUMNS ビュー
 データを入れる, 20-5
USER_TABLESPACES, 9-30, 10-13
USER_TABLES ビュー
 データを入れる, 20-5
ユーザー
 エンド・ユーザーのセキュリティ方針, 22-5
UTLCHAIN.SQL, 20-8
UTLLOCKT.SQL スクリプト, 4-8

V

VSARCHIVE_DEST ビュー
 アーカイブ先の状態の取得, 7-13
VSARCHIVE ビュー, 7-22

VSDATABASE ビュー, 7-22
VSDATAFILE, 9-30, 10-13
VSDBFILE ビュー, 2-8
VSDISPATCHER ビュー
 ディスパッチャ・プロセス・ロードの制御, 4-7
VSLICENSE ビュー, 23-6
VSLOG ビュー
 アーカイブ状態の表示, 7-22
VSLOGFILE ビュー, 2-8
 REDO データの表示, 6-18
 ログファイルの状態, 6-15
VSLOGMNR_CONTENTS ビュー, 7-29
 アーカイブ済み REDO ログの分析に使用, 7-24
VSLOG ビュー, 7-22
 REDO データの表示, 6-18
 オンライン REDO ログ, 6-18
VSPWFILE_USERS ビュー, 1-13
VSQUEUE ビュー
 ディスパッチャ・プロセス・ロードの制御, 4-7
VSROLLNAME
 PENDING OFFLINE セグメントの検索, 21-16
VSROLLSTAT
 PENDING OFFLINE セグメントの検索, 21-16
VSSSESSION, 8-14
VSSSESSION ビュー, 4-16
VSTHREAD ビュー, 6-18
 REDO データの表示, 6-18
VALIDATE STRUCTURE オプション, 20-8
VARCHAR2 データ型, 12-16
 使用領域, 12-16

W

WORM デバイス
 読み取り専用表領域, 9-14

あ

アーカイブ
 アーカイブ先
 障害, 7-15
 アーカイブ先の状態, 7-12
 アクティブ / 非アクティブ, 7-13
 使用可能 / 使用禁止, 7-13
 有効 / 無効, 7-12
 アーカイブ・バッファ・パラメータの設定, 7-21
 アーカイブ・モードの変更, 7-6

- 権限
 - 手動アーカイブ, 7-10
 - 使用可能, 7-7
 - 使用禁止, 7-9
- 高速化, 7-21
- システム・パフォーマンスへの影響を最小にする, 7-21
- 手動, 7-9
- 障害アーカイブ先へ, 7-18
- 初期モードの設定, 7-6
- 使用可能, 7-6, 7-8
- 使用禁止, 7-6
- 自動
 - インスタンス起動後の使用可能化, 7-8
 - インスタンス起動時の使用可能化, 7-8
 - 起動時の使用禁止, 7-9
 - 使用可能, 7-7
 - 使用禁止, 7-8
- 情報の表示, 7-22
- 短所, 7-4
- 長所, 7-4
- 調整, 7-19
- 複数の ARCH プロセス, 7-19
- アーカイブ先
 - アーカイブ済み REDO ログ
 - 使用例, 7-17
- アーカイブ先の指定
 - アーカイブ済み REDO ログ, 7-10
- アーカイブ先の遅延状態, 7-13
- アーカイブ先のパラメータ・エラー状態, 7-13
- アーカイブ済み REDO ログ, 7-2
- アーカイブ先
 - 障害アーカイブ先への再アーカイブ, 7-18
 - 使用例, 7-17
- アーカイブ先の指定, 7-10
- アーカイブ先の状態, 7-12
 - アクティブ / 非アクティブ, 7-13
 - 使用可能 / 使用禁止, 7-13
 - 遅延, 7-13
 - パラメータ・エラー, 7-13
 - 有効 / 無効, 7-12
- アーカイブ・モード, 7-6
- 障害アーカイブ先, 7-15
- 自動アーカイブ, 7-8
- 自動アーカイブの使用可能化, 7-7
- 状態情報, 7-22
- スタンバイ送信, 7-14
- 送信, 7-14
- 多重化, 7-10
- 調整, 7-19
- 通常送信, 7-14
- 分析, 7-24
- アーカイブ済み REDO ログのアーカイブ先の状態, 7-12
- アーカイブ済み REDO ログの送信, 7-14
 - スタンバイ送信モード, 7-14
 - 通常送信モード, 7-14
- アーカイブ済み REDO ログの分析, 7-24
- アーカイブ・バッファ・パラメータ, 7-21
- アーカイブ・バッファ・パラメータの設定, 7-21
- アカウント
 - オペレーティング・システム
 - データベース管理者, 1-4
 - ロール識別機能, 24-18
 - ユーザー
 - SYS と SYSTEM, 1-4
- 空き領域
 - 結合, 9-8
 - 使用可能エクステンツのリスト表示, 20-31
 - 表領域, 9-31
- アクセス
 - オブジェクト
 - 権限のタイプ, 24-4
 - 権限の取消し, 24-13
 - 権限の付与, 24-11
 - データ
 - 管理, 24-1
 - システム権限, 24-2
 - データベース
 - 権限の取消し, 24-13
 - 権限の付与, 24-10
 - 制御, 23-1
 - 制限, 3-4
 - データベース管理者のアカウント, 1-4
- アクティブなアーカイブ先の状態
 - アーカイブ済み REDO ログ, 7-13
- アプリケーション
 - メンテナンス操作時の休止, 13-20
- アプリケーション開発
 - セキュリティ, 22-9
- アプリケーションの開発者
 - ロール, 22-9
- アプリケーション開発者
 - 権限, 22-8

- アプリケーション管理者, 1-3
 - データベース管理者との関係, 22-10
- 暗号化
 - Oracle パスワード, 23-7

い

- 移行
 - データベースの移行, 2-3
- 依存性
 - 表示, 20-30
- 位置
 - ロールバック・セグメント, 21-7
- 一時セグメント
 - 索引作成, 16-3
- 一時領域
 - 割当て, 14-5
- 移動
 - 再配置, 10-9
 - 索引パーティション, 13-10
 - 制御ファイル, 5-4
 - 表パーティション, 13-10
- インスタンス
 - 起動, 3-2
 - 即時停止, 3-11
 - 中断, 3-12
 - データベース作成前に起動, 2-6
- インスタンスの起動
 - 一般手順, 3-2
 - 回復, 3-5
 - 強制的, 3-5
 - システム起動時に自動起動, 3-6
 - 自動アーカイブの使用可能化, 7-8
 - 制限モード, 3-4
 - 通常の方法, 3-4
 - ディスパッチャ・プロセス, 4-5
 - データベースのクローズとマウント, 3-4
 - データベースの作成時, 3-3
 - データベースのマウントとオープン, 3-4
 - データベース名の競合, 2-9
 - データベースをマウントしない, 3-3
 - 排他モード, 3-5
 - 発生する問題, 3-5
 - パラレル・モード, 3-5
 - マルチスレッド・サーバー, 3-2
 - マルチスレッド・サーバーを使用, 4-4
 - リモート・インスタンスの起動, 3-6

- 例, 3-6
- インスタンスの構成
 - 専用サーバー・プロセス, 4-2
- インスタンスの停止
 - INTERNAL として接続, 3-10
 - インスタンスの中断, 3-12
 - 接続, 3-9
 - 即時, 3-11
 - 通常の方法, 3-10
 - 例, 3-11
- インストール
 - Oracle8i, 1-17
 - チューニングに関する推奨事項, 2-14
 - データベースの作成, 2-3
- インダウト・トランザクション
 - ロールバック・セグメント, 21-11
- インポート
 - ジョブ, 8-7
- インポート・ユーティリティ
 - 制限モード, 3-4
 - 説明, 1-16

え

- 英数字データ型, 12-16
- エクステント
 - 削除した表, 14-12
 - 使用可能エクステントの表示, 20-31
 - 情報の表示, 20-31
 - データ・ディクショナリ・ビュー, 20-29
 - 割当て
 - クラスタ, 17-9
 - 索引作成, 16-5
 - 表, 14-11
- エクスポート
 - モード, 7-13, 7-17, 7-18
- エクスポート・ユーティリティ
 - 制限モード, 3-4
 - 説明, 1-16
- エラー
 - ALERT ファイル, 4-10
 - ORA-00028, 4-16
 - ORA-01090, 3-9
 - ORA-01173, 5-9
 - ORA-01176, 5-9
 - ORA-01177, 5-9
 - ORA-1215, 5-9

- ORA-1216 , 5-9
- ORA-1547 , 20-27
- ORA-1628 ~ 1630 , 20-27
- インスタンス起動時 , 3-5
- 制御ファイル作成時 , 5-9
- データベース作成時 , 2-8
- トレース・ファイル , 4-10
- 古すぎるスナップショット , 21-6

お

置換え

- ビュー , 15-8
- オフライン化
 - 表領域 , 9-10
- オフライン表領域
 - 変更 , 9-10
 - 優先順位 , 9-10
 - ロールバック・セグメント , 21-10
- オフライン・ロールバック・セグメント
 - オンライン化 , 21-11
 - 説明 , 21-10
- オブジェクト・スキーマ
 - 削除されたユーザーが所有する , 23-16
 - 取り消された表領域 , 23-13
 - 権限 , 24-4
 - 権限の取消し , 24-13
 - 権限の付与 , 24-11
 - デフォルト表領域 , 23-12
 - 取消しでの連鎖的影響 , 24-15
- オブジェクトの分析
 - 権限 , 20-3
 - 説明 , 20-3
- オペレーティング・システム
 - Oracle8i プロセス名 , 4-9
 - アカウント , 24-18
 - オープン・ファイル数の制限 , 10-2
 - 監査に使用 , 25-2
 - データ・ファイルの削除 , 9-15
 - セキュリティ , 22-3
 - データベース管理者の要件 , 1-4
 - 認証 , 24-17
 - ファイルの改名と再配置 , 10-9
 - ロール識別機能 , 24-18
 - ロール , 24-17
 - ロールを使用可能および使用禁止にする , 24-19
- オンライン REDO ログ , 6-2

- INVALID メンバー , 6-15
- STALE メンバー , 6-15
- 位置 , 6-9
- 管理 , 6-1
- グループの削除 , 6-14
- 権限
 - グループの削除 , 6-14
 - グループの追加 , 6-11
 - メンバーの削除 , 6-15
 - ログ・スイッチの強制 , 6-16
- 構成のガイドライン , 6-5
- 最適の構成 , 6-9
- 作成
 - グループとメンバー , 6-11
- 情報の表示 , 6-18
- データ・ファイルと分離した格納 , 10-4
- データベースのオープン中に使用不可能 , 3-3
- バックアップを作成しない , 7-3
- ファイル数 , 6-9
- ファイルの移動 , 6-13
- ファイルの改名 , 6-13
- メンバーの改名 , 6-12
- メンバーの削除 , 6-14
- メンバーの作成 , 6-12
- ログ・スイッチの強制 , 6-16
- オンライン化
 - 表領域 , 9-10
- オンライン索引 , 16-6
- オンライン表領域
 - 変更 , 9-10
- オフライン・ロールバック・セグメント
 - 使用するとき , 21-10
- オンライン・ロールバック・セグメント
 - オフライン化 , 21-12
 - 新規作成時 , 21-8
 - 説明 , 21-10
 - ロールバック・セグメントのオンライン化 , 21-11

か

- 開発者、アプリケーション , 22-8
- 回復
 - 新しい制御ファイルの作成 , 5-5
 - 自動回復を実行する起動 , 3-5
- 回復不能
 - 表 , 14-4
- 回復不能索引

- 索引, 16-5
 - 改名
 - オンライン REDO ログ・メンバー, 6-12
 - スキーマ・オブジェクト, 20-2
 - 制御ファイル, 5-4
 - 単一の表領域のデータ・ファイル, 10-9
 - データ・ファイル, 10-9, 10-10
 - 書き込み可能表領域, 9-13
 - 監査, 25-2
 - AUDIT コマンド, 25-8
 - 疑わしいアクティビティ, 25-3
 - オブジェクト監査に必要な権限, 25-10
 - オブジェクト監査のためのショートカット, 25-8
 - オプションを使用可能にする, 25-8, 25-12
 - オプションを使用可能にすることとの関係, 25-9
 - オプションを使用禁止にする, 25-10, 25-11, 25-12
 - オプションを使用禁止にすることとの関係, 25-11
 - オペレーティング・システム監査証跡, 25-6
 - 監査オプションのレベル, 25-7
 - 監査証跡の管理, 25-4
 - 監査証跡レコード, 25-5
 - ガイドライン, 25-2
 - 権限監査オプション, 25-8
 - システム監査に必要な権限, 25-10
 - システム権限, 25-9
 - システムのショートカット, 25-7
 - 情報を管理しやすい状態に維持, 25-2
 - スキーマ・オブジェクト, 25-10
 - セッション・レベル, 25-8
 - データベースを使用, 25-2
 - デフォルトのオプション, 25-10
 - デフォルトのオプションを使用禁止にする方法, 25-12
 - トリガー, 25-19
 - 表示
 - アクティブなオブジェクト・オプション, 25-17
 - アクティブな権限オプション, 25-17
 - アクティブな文オプション, 25-17
 - デフォルトのオブジェクト・オプション, 25-18
 - ビュー, 25-4
 - 文, 25-9
 - 文レベル, 25-7
 - 方針, 22-17
 - 履歴情報, 25-4
 - 監査証跡, 25-13
 - アーカイブ, 25-14
 - 解釈, 25-16
 - サイズの縮小, 25-14
 - サイズの制御, 25-13
 - 最大サイズ, 25-13
 - 作成と削除, 25-4
 - 整合性の保護, 25-15
 - 内のレコード, 25-7
 - ビュー, 25-4
 - ビューの削除, 25-5
 - 変更の監査, 25-15
 - 変更の記録, 25-15
 - 保持する表, 25-2
 - レコードの消去, 25-14
 - 監視
 - インスタンスのプロセス, 4-8
 - データ・ファイル, 10-13
 - パフォーマンス表, 4-9
 - 表領域, 10-13
 - ロールバック・セグメント, 21-6
 - ロック, 4-8
 - 管理
 - オブジェクト依存性, 20-23
 - 監査, 25-1
 - クラスタ, 17-1
 - クラスタ化された表, 17-1
 - クラスタ索引, 17-1
 - 索引, 16-1, 16-13
 - シノニム, 15-11
 - 順序, 15-9
 - ジョブ, 8-3
 - 表, 14-1
 - ビュー, 15-1, 15-9
 - プロファイル, 23-16
 - ユーザー, 23-10
 - ロール, 24-4
 - ロールバック・セグメント, 21-1
 - ガイドライン
 - ロールバック・セグメントの管理, 21-2
- ## き
-
- キー
 - クラスタ, 17-2
 - キー保存表
 - 結合ビュー, 15-5
 - 記憶領域
 - 表領域の変更, 9-8
 - 表領域の取消し, 23-13

- 無制限の割当て制限, 23-13
- 割当て制限, 23-13
- 記憶領域パラメータ
 - INITIAL, 12-7, 14-10
 - INTRANS, 12-9, 14-10
 - MAXEXTENTS, 12-8
 - MAXTRANS, 12-9, 14-10
 - MINEXTENTS, 12-8, 14-10
 - NEXT, 12-8
 - OPTIMAL (ロールバック・セグメントでの), 21-6
 - PCTINCREASE, 12-8
 - SYSTEM ロールバック・セグメント, 21-9
 - 一時セグメント, 12-11
 - 設定値の変更, 12-11
 - 適用できるオブジェクト, 12-7
 - データ・ディクショナリ, 20-25, 20-27
 - デフォルト, 12-7
 - 優先順位, 12-11
 - ロールバック・セグメント, 21-8
- 記憶領域パラメータの優先順位, 12-11
- 記憶領域パラメータ
 - PCTFREE, 14-10
 - PCTUSED, 14-10
- 機密性
 - セキュリティ, 22-3
- キャラクタ・セット
 - Oracle によるサポート, 12-16
 - データベース作成時の指定, 2-2
 - パラメータ・ファイル, 3-14
 - マルチバイト・キャラクタ
 - ユーザー・パスワード, 23-11
 - ロール・パスワード, 24-8
 - ロール名, 24-5
- キューに送られたジョブの所有者, 8-7
- 共有 SQL 領域
 - ANALYZE コマンド, 20-7
- 共有サーバー・プロセス
 - 数を変更する権限, 4-6
 - 最小数の変更, 4-6
 - トレース・ファイル, 4-10
- 共有プール
 - ANALYZE コマンド, 20-7
- 共有モード
 - ロールバック・セグメント, 21-3
- 切捨て
 - クラスタ, 20-9
 - 権限, 20-10

- パーティション・オブジェクト, 13-14
- 表, 20-9

行

- 整合性制約違反, 20-14
- ブロック間での連鎖, 12-4, 20-8

<

クラスタ

- PCTFREE の指定, 12-4
- 位置, 17-5
- エクステントの割当て, 17-9
- 管理, 17-1
- 管理のガイドライン, 17-4
- 概要, 17-2
- キー, 17-2
- 記憶領域パラメータ, 12-10
- 切捨て, 20-9
- クラスタ・キーの列, 17-4
- 権限
 - 削除するための, 17-10
 - 作成のための, 17-6
- 構造の妥当性検査, 20-8
- 索引, 16-2
 - ハッシュと対比, 18-2
- 索引作成, 17-7
- 削除, 17-9
- 削除した表, 14-13
- 作成, 17-6
- データの選択, 17-4
- 統計分析, 20-3
- ハッシュ
 - 索引と対比, 18-2
- ハッシュ・クラスタ, 18-1
- 変更, 17-8
- 領域の見積り, 17-5, 17-6

クラスタ化された表, 17-10

クラスタ・キー

- SIZE パラメータ, 17-5
- 列, 17-4

グループ

- REDO ログ・ファイル
 - LOG_FILES 初期化パラメータ, 6-10

グローバル索引

- パーティションの削除, 13-12, 13-14
- パーティションの分割, 13-17

グローバルなデータベース名, 2-9

グローバルなユーザー, 23-10

け

計画

- データベース, 1-18
- データベース作成, 2-2
- リレーショナル設計, 1-18

警告

- CONTROL_FILES パラメータの設定, 2-9
- 監査オプションを使用禁止にする, 25-11
- 監査を使用可能にする, 25-9
- データ・ディクショナリの記憶領域パラメータの変更, 20-25
- ミラー化された制御ファイルの使用, 5-2
- ロールバック・セグメントの作成, 2-12

結合ビュー, 15-4

- DELETE 文, 15-6
- キー保存表, 15-5
- 変更
 - 規則, 15-6
- 変更可能なとき, 15-4
- マージ可能, 15-5

権限, 24-2, 24-4

- CREATE SCHEMA コマンド, 20-2
- REDO ログ・グループの追加, 6-11
- RESTRICTED SESSION システム権限, 3-4, 3-8
- アプリケーション開発者, 22-8
- オブジェクト, 24-4
- オブジェクト権限の取消し, 24-13
- オブジェクトの監査, 25-10
- オブジェクトの分析, 20-3
- オペレーティング・システム
 - データベース管理者に必要な権限, 1-4

改名

- REDO ログ・メンバー, 6-12
- オブジェクト, 20-2
- 表領域のデータ・ファイル, 10-9
- 複数の表領域のデータ・ファイル, 10-10

管理の方針, 22-5

切捨て, 20-10

クラスタ作成, 17-6

権限付与のリスト, 24-21

個々の権限名, 24-2

削除

- REDO ログ・グループ, 6-14
- オンライン REDO ログ・メンバー, 6-15

クラスタ, 17-10

索引, 16-13

シノニム, 15-11

順序, 15-11

表, 14-11

ビュー, 15-9

ロール, 24-10

ロールバック・セグメント, 21-14

作成

- シノニム, 15-11
- 順序, 15-9
- 表, 14-9
- 表領域, 9-4
- ビュー, 15-2
- ユーザー, 23-10
- ロール, 24-4
- ロールバック・セグメント, 21-7

システム, 24-2

システムの監査, 25-10

使用の監査, 25-8

自動アーカイブの使用可能化, 7-7

自動アーカイブの使用禁止, 7-9

ジョブ・キュー, 8-4

プロシージャの再コンパイル, 20-24

セッション制限の変更, 23-4

選択した列, 24-14

データ・ファイルのオフラインとオンラインの切替え, 10-7

データベース管理者, 1-4

トリガーの使用可能および使用禁止, 20-11

取消し, 24-13

オブジェクト権限, 24-15

システム権限, 24-13

パッケージの再コンパイル, 20-24

表領域のオフライン化, 9-10

表領域のオンライン化, 9-10

表領域の結合, 9-9

表領域へのデータ・ファイルの追加, 10-5

ビューの置換え, 15-8

ビューの再コンパイル, 20-23

付与

オブジェクト権限, 24-11

システム権限, 24-10

説明, 24-10

必要な権限, 24-11

プロファイルの削除, 23-20

変更

- 索引, 16-12
- 順序, 15-9
- ディスパッチャ権限, 4-7
- デフォルトの記憶領域パラメータ, 9-8
- 名前付きユーザーの制限, 23-6
- パスワード, 23-15
- 表, 14-10
- ユーザー, 23-14
- ロール認証, 24-7
- ロールバック・セグメント, 21-9
- リソース・コストの設定, 23-19
- リソース制限を使用可能および使用禁止にする方法, 23-20
- 列, 24-12
- 連鎖的な取消し, 24-15
- ロールで分類, 24-4
- ログ・スイッチの強制, 6-16
- 手動アーカイブ, 7-10
- 権限とロールの取消し
 - REVOKE コマンド, 24-13
 - オブジェクト権限のショートカット, 24-4
 - 選択した列, 24-14
- 取消し、権限とロール
 - オペレーティング・システム・ロールを使用するとき, 24-19
- 権限とロールの付与
 - SYSOPER/SYSDBA 権限, 1-12
 - オブジェクト権限のショートカット, 24-4
 - 権限付与のリスト, 24-20

こ

- 高水位標
 - セッション, 23-3
- コスト
 - リソース制限, 23-19
- コマンド、SQL
 - CREATE DATABASE, 6-10
- コマンド、SQL*Plus
 - ARCHIVE LOG, 6-14
 - HOST, 6-13

さ

- サーバー
 - 専用
 - マルチスレッドと対比, 4-3

- マルチスレッド
 - 専用と対比, 4-3
- サーバー単位
 - 複合制限, 23-18
- 再コンパイル
 - 自動的, 20-23
 - パッケージ, 20-24
 - ビュー, 20-23
 - ファンクション, 20-24
 - プロシージャ, 20-24
- サイズ
 - データ・ファイル, 10-4
 - ハッシュ・クラスタ, 18-4
 - ロールバック・セグメント, 21-4
- サイズの見積り
 - ハッシュ・クラスタ, 18-4
 - 表, 14-4
- 再配置
 - 制御ファイル, 5-4
 - データ・ファイル, 10-9, 10-10
- 索引
 - INITRANS, 16-3
 - MAXTRANS, 16-3
 - PCTFREE, 16-4
 - PCTUSED, 16-4
 - SQL*Loader, 16-3
 - 一時セグメント, 16-3
 - エクステント割当て, 16-5
 - 管理, 16-1, 16-13
 - 管理のガイドライン, 16-2
 - 概要, 16-2
 - 記憶領域パラメータ, 12-10
 - 記憶領域パラメータの設定, 16-5
 - クラスタ
 - 管理, 17-1
 - 削除, 17-9
 - 作成, 17-6
 - 変更, 17-9
 - 権限
 - 削除するための, 16-13
 - 変更のための, 16-12
 - 構造の妥当性検査, 20-8
 - サイズの見積り, 16-5
 - 索引作成の平行化, 16-4
 - 削除, 16-13
 - 削除した表, 14-12
 - 作成

- 回復不能, 16-5
- 表データ挿入後, 16-3
- 明示的, 16-8
- 制約の使用禁止および削除, 16-7
- 正しい表および列, 16-7
- 統計分析, 20-3
- パーティションの追加, 13-11
- 表との分離, 14-5
- 表領域, 16-4
- 表あたりの制限, 16-3
- 変更, 16-12
- 領域使用の監視, 16-13
- 索引構成表, 14-13
- 索引パーティション
 - 移動, 13-10
 - 再構築, 13-19
 - 削除, 13-13
 - 分割, 13-17
- 削除
 - オンライン REDO ログ・グループ, 6-14
 - オンライン REDO ログ・メンバー, 6-14
 - 監査証跡, 25-4
 - クラスタ, 17-9
 - クラスタ索引, 17-9
 - 索引, 16-13
 - 索引パーティション, 13-13
 - シノニム, 15-11
 - 順序, 15-11
 - 制御ファイル, 5-9
 - 整合性制約
 - 索引への影響, 16-7
 - 説明, 20-20
 - データ・ファイル, 9-14
 - データベース, 2-8
 - ハッシュ・クラスタ, 18-9
 - 表, 14-11
 - 表の統計, 20-4
 - 表パーティション, 13-11
 - 表領域
 - 説明, 9-14
 - 必要な権限, 9-14
 - ビュー, 15-9
 - プロファイル, 23-20
 - ユーザー, 23-16
 - ロール, 24-10
 - ロールバック・セグメント, 21-11, 21-14
- 作成

- REDO ログ・メンバー, 6-12
- オンライン REDO ログ・グループ, 6-11
- 監査証跡, 25-4
- クラスタ, 17-6
- クラスタ化された表, 17-6
- クラスタ索引, 17-6
- 索引
 - 明示的, 16-8
- シノニム, 15-11
- 順序, 15-9
- 制御ファイル, 5-3
- データ・ファイル, 9-3, 10-5
- データベース, 1-19, 2-1, 7-6
 - CREATE DATABASE の実行, 2-6
 - 新しいデータベースのバックアップ, 2-6
 - インストール中, 2-3
 - 異なるバージョンからの移行, 2-3
 - 準備, 2-2
 - 前提条件, 2-3
 - 発生する問題, 2-8
- ハッシュ・クラスタ, 18-4
- ハッシュ・クラスタ表, 18-4
- パーティション・オブジェクト, 13-8
- パーティション表, 13-8
- パラメータ・ファイル, 2-4
- 表, 14-9
- 表領域, 9-3
 - 必要なロールバック・セグメント, 9-4
- ビュー, 15-2
- 複数のオブジェクト, 20-2
- プロファイル, 23-17
- ロールバック・セグメント
 - 記憶領域パラメータの指定, 21-8
 - 説明, 21-7
- 参照整合性制約
 - 表パーティションの切捨て, 13-15
 - 表パーティションの削除, 13-13

し

- 識別
 - ユーザー, 23-7
- システム・グローバル領域, 2-11
- システム・グローバル領域 (SGA), 2-11
- システム権限, 24-2
- システム変更番号 (SCN)
 - 決定される時期, 6-3

データ・ファイルのチェック, 10-13

シノニム

依存性の表示, 20-30

管理, 15-11

削除, 15-11

削除した表, 14-12

削除するための権限, 15-11

作成, 15-11

作成するための権限, 15-11

パブリック, 15-11

プライベート, 15-11

手動アーカイブ

ARCHIVELOG モード, 7-9

障害

メディア

多重オンライン REDO ログ, 6-5

ショートカット

オブジェクト監査, 25-8

オブジェクト権限, 24-4

文レベル監査オプション, 25-7

CONNECT、監査のため, 25-8

初期

SYS と SYSTEM のパスワード, 1-4

初期化パラメータ

LOG_ARCHIVE_BUFFER_SIZE, 7-21

LOG_ARCHIVE_BUFFERS, 7-21

LOG_ARCHIVE_DEST_n, 7-10

LOG_ARCHIVE_DEST_STATE_n, 7-12

LOG_ARCHIVE_MAX_PROCESSES, 7-19

LOG_ARCHIVE_MIN_SUCCEED_DEST, 7-16

LOG_ARCHIVE_START, 7-8, 7-9, 7-13

LOG_BLOCK_CHECKSUM, 6-16

LOG_FILES, 6-10

順序に影響する, 15-10

マルチスレッド・サーバー, 4-4

使用可能

アーカイブ, 7-6

監査オプション

権限, 25-12

説明, 25-8

整合性制約

違反が存在するとき, 20-14

作成時, 20-17

例, 20-18

例外のレポート, 20-20

トリガー, 20-11

リソース制限, 23-20

使用可能なアーカイブ先の状態

アーカイブ済み REDO ログ, 7-13

使用禁止

アーカイブ, 7-6, 7-8

監査, 25-12

監査オプション, 25-10, 25-11

整合性制約, 20-17

索引への影響, 16-7

トリガー, 20-12

リソース制限, 23-20

使用禁止のアーカイブ先の状態

アーカイブ済み REDO ログ, 7-13

時間枠

履歴表内での移動, 13-19

事前定義済みのロール, 1-5

自動アーカイブ

アーカイブ・ログのアーカイブ先, 7-8

順序

Parallel Server, 15-10

管理, 15-9

削除, 15-11

削除するための権限, 15-11

作成, 15-9

作成するための権限, 15-9

初期化パラメータ, 15-10

変更, 15-10

変更するための権限, 15-9

ジョブ

INTERNAL 日付関数, 8-8

インポート, 8-7

エクスポート, 8-7

管理, 8-3

強制的な実行, 8-14

所有者, 8-7

実行, 8-9

ジョブ・キューからの削除, 8-11

ジョブ・キューに送る方法, 8-4

ジョブ定義, 8-7

ジョブ番号, 8-7

スケジューリング, 8-3

中止, 8-14

中断された, 8-12

中断されたジョブの実行, 8-13

中断されたジョブをマークする, 8-13

データベース・リンク, 8-9

トレース・ファイル, 8-10

変更, 8-11

- 問題解決, 8-10
- ジョブ・キュー, 8-2, 8-3
 - 使用する権限, 8-4
 - ジョブの削除, 8-11
 - ジョブのスケジュールリング, 8-3
 - 中のジョブの実行, 8-9
 - 表示, 8-15
 - ロック, 8-9
- ジョブのエクスポート, 8-7
- ジョブの環境, 8-6

す

- スキーマ・オブジェクト
 - アクセスするための権限, 24-4
 - オブジェクト間の依存性, 20-23
 - 改名, 20-2, 20-3
 - 改名する権限, 20-2
 - 監査オプションを使用可能にする, 25-10
 - 監査オプションを使用禁止にする方法, 25-11
 - 情報のリスト表示, 20-28
 - タイプ別のリスト, 20-29
 - デフォルトの監査オプション, 25-10
 - 複数のオブジェクトを作成する, 20-2
- スタンバイ送信モード
 - Net8, 7-15
 - RFS プロセス, 7-14
 - 定義, 7-14
- ストアド・プロシージャ
 - PUBLIC に付与された権限の使用, 24-16
 - 再コンパイルのための権限, 20-24
- ストリーム
 - テープ・ドライブ, 7-21
- スナップショット
 - 記憶領域パラメータ, 12-10
 - 古すぎる
 - OPTIMAL 記憶領域パラメータ, 21-6
- スナップショット・ログ
 - 記憶領域パラメータ, 12-10
- スレッド
 - オンライン REDO ログ, 6-2

せ

- 制御ファイル
 - 1 つは必要, 5-3
 - 位置, 5-3

- 移動, 5-4
- 改名, 5-4
- 数, 5-3
- 管理, 5-1
- ガイドライン, 5-2
- 既存のファイルの上書き, 2-9
- 起動時に使用不可能, 3-3
- サイズ, 5-3
- サイズの変更, 5-4
- 再配置, 5-4
- 削除, 5-9
- 作成
 - 新しいファイル, 5-5
 - 初期, 5-4
 - 説明, 5-3
 - 追加の制御ファイル, 5-4
- 作成中のエラー, 5-9
- 追加, 5-4
- データ・ディクショナリとの不一致, 5-8
- データベース作成前に名前を指定, 2-9
- デフォルト名, 2-10, 5-4
- 名前, 5-2
- ミラー化, 2-10
- ミラー化の重要性, 5-2
- ログ順序番号, 6-5
- 制限
 - セッション、高水位標, 23-3
 - 同時使用, 23-2
 - 複合制限, 23-18
 - リソース制限, 23-18
- 整合性制約
 - 違反, 20-14
 - 違反が存在する場合に使用可能にする, 20-14
 - 管理, 20-15
 - 削除, 20-20
 - 削除および使用禁止, 16-7
 - 作成時に使用可能にする, 20-17
 - 作成時に使用禁止にする, 20-17
 - 使用可能, 20-13
 - 使用禁止, 20-13, 20-18
 - 使用禁止にするととき, 20-14
 - 表領域の削除, 9-15
 - 例外, 20-20
- 整合性制約違反, 20-14
- 責任
 - データベース管理者, 1-2
 - データベース・ユーザー, 1-3

セキュリティ

- REMOTE_OS_ROLES パラメータ , 24-20
- アプリケーション開発者 , 22-8
- 一般のユーザー , 22-4
- オペレーティング・システムのセキュリティとデータベース , 22-3
- 監査証跡の保護 , 25-15
- 監査方針 , 22-17
- 管理者 , 22-2
- 機密性 , 22-3
- 権限 , 22-2
- 権限管理の方針 , 22-5
- セキュリティ管理者 , 1-3
- セキュリティを強制するロール , 22-5
- データ , 22-3
- データベース管理者の方針 , 22-7
- データベース・セキュリティ , 22-2
- データベースへのアクセス , 22-2
- データベース・ユーザー , 22-2
- 方針の設定 , 22-1
- マルチバイト・キャラクタ
 - ユーザー・パスワード , 23-11
 - ロール・パスワード , 24-8
 - ロール名 , 24-5
- ユーザーの認証 , 22-2

セグメント

- 一時記憶領域パラメータ , 12-11
- 監視 , 21-15
- 情報の表示 , 20-31
- データ・ディクショナリ , 20-27
- データと索引
 - デフォルトの記憶領域パラメータ , 12-10
- ロールバック , 21-1

セッション

- Parallel Server のセッション制限 , 2-12
- インスタンスごとの制限 , 23-2
- インスタンスでの最大数の設定 , 23-4
- インスタンスの警告制限の設定 , 23-4
- 権限ドメインのリスト , 24-22
- 現行数と高水位標の参照 , 23-6
- 接続と切断の監査 , 25-8
- 同時実行セッションの数 , 2-12
- メモリー使用の表示 , 23-24

セッション、ユーザー

- アクティブ , 4-16
- アクティブでない , 4-16
- 停止 , 4-15

停止したセッションのビュー , 4-16

停止のマークを設定 , 4-16

セッションの制限、ライセンス

初期設定 , 2-12

セッションの停止

- アクティブでないセッション , 4-16
- アクティブでないセッション、例 , 4-17
- アクティブなセッション , 4-16
- セッションの識別 , 4-15

セッション・モニター , 4-8

接続

- 監査 , 25-8
- 管理者権限 , 3-10
- 専用サーバー , 4-3
- 停止処理中 , 3-9

切断

監査 , 25-8

専用サーバー

マルチスレッド・サーバーと対比 , 4-3

専用サーバー・プロセス

- 構成 , 4-2
- 接続 , 4-3
- トレース・ファイル , 4-10

前提条件

データベースの作成 , 2-3

そ

ソフトウェア・バージョン , 1-20

た

多重化

- REDO ログ・ファイル , 6-5
- グループ , 6-6
- アーカイブ済み REDO ログ , 7-10

ち

チェックサム

- REDO ログ・ブロック , 6-16
- データ・ブロック , 10-12

チェックポイント・プロセス (CKPT)

起動 , 4-12

中止

ジョブ , 8-14

中断

インスタンスの停止, 3-12

中断されたジョブ

実行, 8-13

説明, 8-12

マークする, 8-13

チューニング

初期, 2-14

調整

アーカイブ, 7-19

データベース, 1-20

つ

通常送信モード

定義, 7-14

て

停止

ユーザー・セッション, 4-15

テープ・ドライブ

アーカイブのためのストリーミング, 7-21

テスト

データベースのセキュリティ, 22-9

ディクショナリ・ファイル

LogMiner, 7-25

ディスパッチャ・プロセス

開始する数, 4-5

数の設定, 4-7

数を変更する権限, 4-7

削除, 4-7

新規作成, 4-7

データ

セキュリティ, 22-3

データ型

DATE, 12-17

LONG, 12-17

NUMBER, 12-17

個々の型名, 12-16

使用領域, 12-18

まとめ, 12-19

文字, 12-16

データ・ディクショナリ

VSDBFILE ビュー, 2-8

VSDISPATCHER ビュー, 4-7

VSLOGFILE ビュー, 2-8

VSQUEUE ビュー, 4-7

記憶領域パラメータの設定, 20-25

記憶領域パラメータの変更, 20-27

削除した表, 14-12

スキーマ・オブジェクト・ビュー, 20-28

制御ファイルとの不一致, 5-8

セグメント, 20-27

データの整合性, 20-18

整合性制約, 20-18

データ・ファイル

MISSING, 5-8

REDO ログ・ファイルと分離した格納, 10-4

位置, 10-4

オフラインにする権限, 10-7

オンライン, 10-8

オンラインとオフラインの切替え, 10-7

改名, 10-9, 10-10

改名する権限, 10-9

監視, 10-13

管理, 10-1

最小数, 10-2

再使用, 10-5

サイズ, 10-4

最大数, 10-2

再配置, 10-9, 10-10

再配置の例, 10-11

削除, 9-14

NOARCHIVELOG モード, 10-8

作成, 9-3

対応付けられた表領域のチェック, 9-30

単一の表領域の場合の改名, 10-9

データ・ブロックの検証, 10-12

データベース管理者によるアクセス, 1-4

データベースのオープン中に使用不可能, 3-3

デフォルト・ディレクトリ, 10-5

表示

VSDBFILE および VSLOGFILE ビュー, 2-8

一般的な状態, 10-13

表領域への追加, 10-5

ファイル名の識別, 10-11

ファイル名を完全に指定, 10-5

データ・ブロック

PCTFREE 記憶領域パラメータ, 12-3

PCTUSED 記憶領域パラメータ, 12-5

オペレーティング・システムのブロックとの比較,
2-11

クラスタで共有される, 17-2

検証, 10-12

- サイズ, 2-10
- サイズの変更, 2-11
- 使用領域の管理, 12-2
- 領域使用方法の管理, 12-2
- データベース
 - CREATE DATABASE コマンド, 2-7
 - アクセスの制限, 3-4, 3-8
 - 移行, 2-3
 - インスタンスへのマウント, 3-7
 - オープン
 - クローズされたデータベース, 3-7
 - 改名, 5-5
 - 可用性, 3-7
 - 監査, 25-1
 - 管理, 1-1
 - サイズ, 10-1
 - 起動
 - アクセスの制限, 3-4
 - 一般手順, 3-2
 - データベース作成前, 2-6
 - グローバルなデータベース名
 - 説明, 2-9
 - 分散システム, 2-9
 - 計画, 1-18
 - 構造
 - 分散データベース, 1-18
 - 削除, 2-8
 - 作成
 - オープン, 1-19
 - 問題の解決, 2-8
 - 制御ファイル, 5-2
 - 制御ファイルの指定, 2-9
 - 設計
 - インプリメント, 1-19
 - 調整
 - 責任, 1-20
 - 大規模データベースのアーカイブ, 7-19
 - テスト, 22-9
 - データ・ファイルと REDO ログ・ファイルの表示, 2-8
 - データベースのマウント, 3-4
 - 名前
 - 競合, 2-9
 - 説明, 2-9
 - ハードウェア評価, 1-18
 - 排他モード, 3-5
 - バックアップ
 - 作成後, 1-19
 - 全体バックアップ, 2-6
 - パラレル・モード, 3-5
 - 物理構造, 1-18
 - 本番, 22-9, 22-10
 - ユーザーの責任, 1-3
 - 論理構造, 1-18
- データベース管理者, 1-2
 - アプリケーション管理者との関係, 22-10
 - オペレーティング・システム・アカウント, 1-4
 - 責任, 1-2
 - セキュリティ, 22-7
 - セキュリティ管理者との関係, 1-3, 22-2
 - セキュリティと権限, 1-4
 - パスワード・ファイル, 1-7
 - ユーザー名, 1-4
 - 優先順位, 1-17
 - ユーティリティ, 1-16
 - ロール
 - セキュリティ, 22-7
 - 説明, 1-5
- データベース設計のインプリメント, 1-19
- データベースのオープン
 - 作成後, 1-19
 - マウントされたデータベース, 3-7
- データベースの起動
 - 一般手順, 3-2
 - 説明, 3-1
- データベースの停止, 3-1
- データベースの物理構造, 1-18
- データベースのマウント, 3-4
 - 排他モード, 3-5
 - パラレル・モード, 3-5
- データベースの論理構造, 1-18
- データベースへのアクセス制限
 - インスタンスの起動, 3-4
- データベース・リソース・マネージャ, 11-1
- データベース・リンク
 - ジョブ・キュー, 8-9
- デフォルト
 - 一時表領域, 23-12
 - 監査オプション, 25-10
 - 使用禁止, 25-12
 - 表領域割当て制限, 23-13
 - プロファイル, 23-17
 - ユーザー表領域, 23-12
 - ロール, 23-15

と

問合せサーバー・プロセス

説明, 4-12

統計

更新, 20-4

登録

データベース・ユーザー, 1-19

トランザクション

コミット

REDO ログ・バッファの書込み, 6-3

特定のロールバック・セグメントへの割当て,
21-13

ロールバック・セグメント, 21-13

トランザクション・エントリ

記憶領域のガイドライン, 12-9

トランザクションのコミット

REDO ログ・バッファの書込み, 6-3

トランスポートابل・テーブルスペース, 9-18

トリガー

監査, 25-19

削除した表, 14-12

使用可能, 20-11

使用可能および使用禁止にする権限, 20-11

使用禁止, 20-12

例, 25-19

取消し

権限とロール

SYSOPER/DBA 権限, 1-12

トレース・ファイル

位置, 4-11

書き込む時期, 4-11

サイズ, 4-11

使用, 4-10, 4-11

ジョブの失敗, 8-10

ログ・ライター, 4-11

ログ・ライター・プロセス, 6-6

動的パフォーマンス表

使用, 4-9

な

名前付きユーザーの制限, 23-5

初期設定, 2-13

に

認可

オペレーティング・システムのロール管理, 24-8

ロール

説明, 24-7

マルチスレッド・サーバー, 24-8

ロールに対して省略, 24-9

ロールに対する変更, 24-9

認証

オペレーティング・システム, 1-7

データベースで管理される, 23-7

パスワード・ファイル, 1-8

パスワード方針, 22-4

ユーザー, 22-2, 23-7, 23-9

ユーザー作成時の指定, 23-11

ね

ネットワーク・プロトコル

それぞれのディスパッチャ, 4-5

は

ハードウェア

評価, 1-18

排他モード

データベース, 3-5

残りのユーザー・セッションの停止, 4-15

ロールバック・セグメント, 21-3

ハッシュ・クラスタ

管理, 18-1

キーの選択, 18-6

記憶領域の見積り, 18-4

クラスタ, 18-1

削除, 18-9

作成, 18-4

使用方法, 18-2

変更, 18-8

領域使用の制御, 18-5

例, 18-7

バージョン, 1-20

他の Oracle ソフトウェア, 1-21

バグ修正, 1-20

バックアップ

アーカイブの影響, 7-4

データベース作成前, 2-4

- データベースの新規作成後
 - ガイドライン, 1-19
 - 全体バックアップ, 2-6
- バックグラウンド・プロセス
 - Oracle8i プロセス, 4-9
- バッファ
 - SGA のバッファ・キャッシュ, 2-11
- パーティション
 - 索引からの削除, 13-13
 - 索引への追加, 13-11
- パーティション・オブジェクト, 13-1 ~ 13-21
 - 移動, 13-10
 - 切捨て, 13-14
 - 作成, 13-8
 - 追加, 13-11
 - 定義, 13-2
 - パーティションの分割, 13-16
 - マージ, 13-17
 - メンテナンス, 13-9 ~ 13-21
 - メンテナンス時のアプリケーションの休止, 13-20
- 索引のパーティション化
 - パーティションの再構築, 13-19
- パーティション表
 - パーティションの追加, 13-11
 - パーティションの分割, 13-16
 - 非パーティション化状態への変換, 13-18
- パーティション・ビュー
 - パーティション表への変換, 13-18
- パスワード
 - REMOTE_LOGIN_PASSWORD パラメータの設定, 1-11
 - SYS と SYSTEM の初期パスワード, 1-4
 - 認証ファイル, 1-8
 - パスワード・ファイル, 1-11
 - OS 認証, 1-7
 - 再配置, 1-15
 - 削除, 1-15
 - 作成, 1-9
 - 状態, 1-15
 - 変更するための権限, 23-14
 - ユーザー認証, 23-7
 - ユーザーのセキュリティ方針, 22-4
 - ロール, 24-8
 - ロールに対する変更, 24-9
 - ロールを変更するための権限, 24-7
- パッケージ
 - DBMS_LOGMNR.D.BUILD, 7-26, 7-27

- DBMS_LOGMNR.ADD_LOGFILE, 7-27
- DBMS_LOGMNR.START_LOGMNR, 7-28
- 再コンパイル, 20-24
- 再コンパイルのための権限, 20-24
- パッチ・リリース番号, 1-21
- パフォーマンス
 - アーカイブの調整, 7-19
 - データ・ファイルの位置, 10-4
- パフォーマンス表
 - 動的パフォーマンス表, 4-9
- パブリック
 - シノニム, 15-11
- パブリック・ロールバック・セグメント
 - オフライン化, 21-13
 - 使用可能にする, 21-10
- パラメータ・ファイル
 - 位置, 3-14
 - 数, 3-14
 - キャラクタ・セット, 3-14
 - 個々のパラメータ名, 2-9
 - 最小セット, 2-8
 - サンプル, 3-14
 - データベース作成前に編集, 2-5
 - データベース作成用に作成, 2-4
- パラレル問合せオプション
 - サーバー・プロセスの数, 4-12
 - 索引作成のパラレル化, 16-4
 - 問合せサーバー, 4-12
 - 表作成の並行化, 14-3
- パラレル・モード
 - データベース, 3-5

ひ

- 非アクティブなアーカイブ先の状態
 - アーカイブ済み REDO ログ, 7-13
- 表
 - PCTFREE の指定, 12-4
 - SYSTEM 表領域, 14-3
 - 位置, 14-3, 14-9
 - 一時領域, 14-5
 - エクステントの割当て, 14-11
 - 回復不能 (UNRECOVERABLE), 14-4
 - 管理, 14-1
 - 管理のガイドライン, 14-1, 14-5
 - キー保存, 15-5
 - 記憶領域パラメータ, 12-10

- 切捨て, 20-9
- クラスタ化された, 17-2
- クラスタ化された表
 - 管理, 17-1
 - 削除, 17-9
 - 削除するための権限, 17-10
 - 作成, 17-6
 - 変更, 17-9
- クラスタ化のスキーマ, 17-7
- 構造の妥当性検査, 20-8
- サイズの見積り, 14-4
- 索引, 16-2
- 索引との分離, 14-5
- 索引の制限, 16-3
- 削除, 14-11
- 削除するための権限, 14-11
- 作成, 14-9
- 作成するための権限, 14-9
- 作成の並行化, 14-3
- 作成前の設計, 14-2
- 統計分析, 20-3
- トランザクション・パラメータ, 14-3
- ハッシュ・クラスタ
 - 管理, 18-1
 - 作成, 18-4
- パーティションの追加, 13-11
- 表領域の指定, 14-3, 14-9
- 変更, 14-10, 14-11
- 変更するための権限, 14-10
- 列の長さの拡張, 14-10
- 評価
 - Oracle8i 用のハードウェア, 1-18
- 表パーティション
 - 切捨て, 13-14
 - グローバル索引を含む, 13-12
 - 交換, 13-18
 - 削除, 13-11
 - 作成, 13-8
 - 分割する, 13-16
- 表領域
 - SYSTEM 表領域, 9-3
 - 空き領域のリスト, 9-31
 - 位置, 10-4
 - 一時, 23-12
 - 一時オフライン化, 9-11
 - オフラインにする権限, 9-10
 - オンライン化, 9-10
 - 書込み可能, 9-13
 - 可用性の変更, 9-10
 - 監視, 10-13
 - 管理, 10-1
 - 管理のガイドライン, 9-2
 - 記憶領域設定の変更, 9-8
 - 削除
 - 説明, 9-14
 - 必要な権限, 9-14
 - 作成, 9-3
 - 作成するための権限, 9-4
 - 追加の作成, 9-4
 - 通常のアフライン化, 9-10
 - データ・ファイルの追加, 10-5
 - デフォルト記憶領域パラメータのチェック, 9-30
 - デフォルトの一時, 23-12
 - デフォルトの記憶領域パラメータ, 12-10
 - デフォルトの記憶領域パラメータの設定, 9-3
 - デフォルトの割当て制限, 23-13
 - 必要なロールバック・セグメント, 9-4
 - ファイルのリスト, 9-30
 - 複数を使用, 9-2
 - 無制限の割当て制限, 23-13
 - ユーザーからの取消し, 23-13
 - ユーザーの割当て制限, 23-12
 - ユーザーへのデフォルトの割当て, 23-12
 - ユーザー割当て制限の割当て, 9-3
 - 読取り専用, 9-12
 - 割当て制限
 - 割当て, 9-3
 - 割当て制限の表示, 23-23
- 表領域結合, 9-8
- 表領域セット, 9-20
- ヒント
 - オブジェクト監査のためのショートカット, 25-8
 - オブジェクト権限のショートカット, 24-4
 - 文監査ショートカット, 25-7
- ビットマップ化された表領域, 9-5
- ビュー
 - FOR UPDATE 句, 15-3
 - ORDER BY 句, 15-3
 - VSARCHIVE, 7-22
 - VSARCHIVE_DEST, 7-13
 - VSDATABASE, 7-22
 - VSLOG, 6-18, 7-22
 - VSLOGFILE, 6-15, 6-18
 - VSLOGMNR_CONTENTS, 7-24, 7-29

VSTHREAD, 6-18
WITH CHECK OPTION, 15-3
依存性の表示, 20-30
エラー付きで作成, 15-4
置換え, 15-8
置き換えるための権限, 15-8
管理, 15-1, 15-9
権限, 15-2
再コンパイル, 20-23
再コンパイルのための権限, 20-23
削除, 15-9
削除した表, 14-12
削除するための権限, 15-9
作成, 15-2
パーティション化
 パーティション表への変換, 13-18
ワイルドカード, 15-3

ふ

ファイル
 OS によるオープン数の制限, 9-2
ファンクション
 再コンパイル, 20-24
ファンクション・ベース索引, 16-9
複合制限, 23-18
 サービス単位, 23-18
複数の ARCH プロセスの指定, 7-19
ブロックの検証
 REDO ログ・ファイル, 6-16
分散処理
 パラメータ・ファイルの位置, 3-14
分散データベース
 ARCHIVELOG モードで実行, 7-6
 NOARCHIVELOG モードで実行, 7-6
 リモート・インスタンスの起動, 3-6
プライベート
 シノニム, 15-11
 ロールバック・セグメント, 21-8
 オフライン化, 21-13
プログラム・グローバル領域 (PGA)
 MAX_ENABLED_ROLES の効果, 24-9
プロシージャ
 再コンパイル, 20-24
プロセス, 4-1
 SNP バックグラウンド・プロセス, 8-2
プロファイル

PUBLIC_DEFAULT, 23-17
管理, 23-16
削除, 23-20
削除するための権限, 23-20
作成, 23-17
制限を NULL に設定, 23-18
デフォルト, 23-17
表示, 23-23
複合制限, 23-18
変更, 23-18
変更するための権限, 23-18
ユーザーへの割当て, 23-18
リスト, 23-21
リソース・コストを設定するための権限, 23-19
リソース制限を使用可能にする, 23-20
リソース制限を使用禁止にする, 23-20

へ

変更

記憶領域パラメータ, 14-10
クラスタ, 17-8
クラスタ化された表, 17-9
クラスタ索引, 17-9
結合ビュー, 15-4
索引, 16-12
順序, 15-10
データベース状態, 3-7
ハッシュ・クラスタ, 18-8
パブリック・ロールバック・セグメント, 21-9
表, 14-10, 14-11
表領域に対する記憶領域, 9-8
ユーザー, 23-14
ロールバック・セグメント記憶領域パラメータ,
 21-9
変更可能な結合ビュー
 定義, 15-4
変更ベクトル, 6-2
ベンディング・エリア, 11-5

ま

マークの付けられたユーザー・セッション, 4-16
マルチスレッド・サーバー
 OS ロール管理制限, 24-20
 OS ロール認可での制限, 24-8
 起動, 4-4

- 使用可能と使用禁止, 4-6
- 専用サーバーと対比, 4-3
- ディスパッチャの構成, 4-5
- データベースの起動, 3-2

み

- ミラー化
 - 制御ファイル, 2-10
- ミラー化された制御ファイル
 - 重要性, 5-2
- ミラー化したファイル
 - オンライン REDO ログ, 6-6
 - 位置, 6-9
 - サイズ, 6-9

む

- 無効なアーカイブ先の状態
 - アーカイブ済み REDO ログ, 7-12

め

- メディア回復
 - アーカイブの影響, 7-4
- メモリー
 - ユーザーごとの表示, 23-24
- メンテナンス・リリース番号, 1-21

も

- モード
 - 制限, 3-4, 3-8
 - 排他, 3-5
 - パラレル, 3-5

ゆ

- 有効なアーカイブ先の状態
 - アーカイブ済み REDO ログ, 7-12
- ユーザー
 - PUBLIC グループ, 24-16
 - 一般のユーザーのセキュリティ, 22-4
 - 数の制限, 2-13
 - 管理, 23-10
 - 権限の管理に関する方針, 22-5
 - 削除, 23-16

- 削除後のオブジェクト, 23-16
- 削除するための権限, 23-16
- 作成するための権限, 23-10
- 識別, 23-7
- 新規作成したデータベース, 2-13
- 情報の表示, 23-22
- セキュリティ, 22-2
- セッション、停止, 4-16
- 他と重複しないユーザー名, 2-13, 23-5
- デフォルト表領域, 23-12
- デフォルト・ロールの変更, 23-15
- 登録, 1-19
- 認証
 - 説明, 22-2, 23-7
 - データベース認証, 23-7
- パスワード・セキュリティ, 22-4
- パスワードを変更するための権限, 23-14
- 表領域割当て制限, 23-12
- 表領域割当て制限の表示, 23-23
- 表領域割当て制限の割当て, 9-3
- 複合制限, 23-18
- 付与される権限のリスト, 24-21
- 付与されるロールのリスト, 24-21
- プロファイルの削除, 23-20
- プロファイルの割当て, 23-18
- 変更, 23-14
- マルチバイト・キャラクタ
 - パスワード, 23-11
- 無制限の割当て制限の割当て, 23-13
- メモリー使用の表示, 23-24
- ユーザー名の指定, 23-11
- リスト, 23-21
- ロールの削除, 24-10
- ユーザー名
 - SYS と SYSTEM, 1-4
- ユーティリティ
 - SQL*Loader, 1-16
 - インポート, 1-16
 - エクスポート, 1-16
 - データベース管理者用, 1-16

よ

- 読取り専用のデータベースのオープン, 3-8
- 読取り専用表領域
 - WORM デバイス上, 9-14
 - 書込み可能に変更, 9-13

作成, 9-12
データ・ファイル, 10-7

ら

ライセンス

制限の参照, 23-6
セッション制限を変更するための権限, 23-4
セッションをベースとした, 23-2
同時使用, 23-2
同時実行セッションの数, 2-12
名前付きユーザー, 23-2, 23-5
名前付きユーザー制限を変更するための権限, 23-6
ライセンス契約の遵守, 2-12, 23-2

り

リソース

プロファイル, 23-16

リソース制限

使用可能, 23-20

リソース・アロケーション・メソッド, 11-2

リソース・コンシューマ・グループ, 11-2

リソース制限

NULL に設定, 23-18
PUBLIC_DEFAULT プロファイル, 23-17
コスト, 23-19
コストを設定するための権限, 23-19
サービス単位, 23-18
使用可能および使用禁止にするための権限, 23-20
使用禁止, 23-20
複合制限, 23-18
プロファイル, 23-16
プロファイルでの変更, 23-18
プロファイルでの割当て, 23-18
プロファイルの作成, 23-17

複合制限

コスト, 23-19

リソース・プラン, 11-2

リソース・プラン・ダイレクティブ, 11-2

リモート接続, 1-15

SYSOPER/SYSDBA として接続, 1-13
パスワード・ファイル, 1-9

領域

索引による使用, 16-13
データベースに追加, 9-4

領域管理

PCTFREE, 12-2
PCTUSED, 12-4

リリース

Oracle8i の識別, 1-20
他の Oracle ソフトウェアのバージョン, 1-21
パッチ・リリース番号, 1-21
ポート固有のリリース番号, 1-21
メンテナンス・リリース番号, 1-21
リリース番号のチェック, 1-21

リレーショナル設計

計画, 1-18

履歴表

時間枠の移動, 13-19

れ

例

索引の変更, 16-12
制約の作成, 20-18

例外

整合性制約, 20-20

列

INSERT 権限, 24-12
権限, 24-12
権限の取消し, 24-14
権限の付与, 24-12
情報の表示, 20-30
選択した列についての権限の付与, 24-11
長さの拡張, 14-10
付与されるユーザーのリスト, 24-22
連鎖的な取消し, 24-15

ろ

ロール

ADMIN OPTION, 24-11
ADMIN OPTION の取消し, 24-13
CONNECT ロール, 24-7
DBA ロール, 1-5, 24-7
EXP_FULL_DATABASE, 24-7
GRANT OPTION, 24-12
GRANT コマンド, 24-19
IMP_FULL_DATABASE, 24-7
OS 管理とマルチスレッド・サーバー, 24-20
RESOURCE ロール, 24-7
REVOKE コマンド, 24-19
SET ROLE コマンド, 24-19

- アプリケーションの開発者, 22-9
- 一意の名前, 24-5
- オペレーティング・システムでの付与, 24-18, 24-19
- オペレーティング・システムを使用した管理, 24-17
- オペレーティング・システム認可, 24-8
- 下位互換性, 24-7
- 管理, 24-4
- 権限
 - 削除するための, 24-10
 - 作成のための, 24-4
 - システム権限またはロールの付与, 24-10
 - 認可方法の変更, 24-7
 - パスワードの変更, 24-7
- 権限とロールのリスト, 24-23
- 権限付与のリスト, 24-21
- 削除, 24-10
- 使用可能にするためのパスワード, 24-8
- 事前定義済み, 1-5, 24-7
- セキュリティ, 22-5
- データベース認可, 24-8
- デフォルト, 23-15
- 取消し, 24-13
- 認可, 24-7
- 認可なし, 24-9
- 認可の変更, 24-9
- パスワードの変更, 24-9
- パスワードのマルチバイト・キャラクタ, 24-8
- 付与
 - 説明, 24-10
- マルチスレッド・サーバー, 24-8
- マルチバイト・キャラクタ
 - 名前, 24-5
 - リスト, 24-23
- 権限
 - 取消し
 - ADMIN OPTION, 24-13
 - GRANT OPTION, 24-14
- ロール
 - ロールで分類, 24-4
- ロールバック・セグメント
 - AVAILABLE, 21-11
 - OFFLINE, 21-11
 - PARTLY AVAILABLE, 21-11
 - PENDING OFFLINE, 21-12
 - 位置, 21-7
- エクステンツのリスト表示, 20-31
- オフライン化, 21-12
- オフラインかどうかの検査, 21-12
- オフライン状態, 21-12
- オフライン・ロールバック・セグメント, 21-10
- オンライン状態, 21-12
- オンライン・ロールバック・セグメント, 21-10
- 数の選択, 2-14
- 監視, 21-6
- 管理, 21-1
- 管理のガイドライン, 21-2
- 記憶領域パラメータ, 21-8
- 記憶領域パラメータの変更, 21-9
- 起動時に取得, 2-12
- 権限
 - 削除するための, 21-14
 - 作成に必要, 21-7
 - 変更するために必要, 21-9
- サイズの縮小, 21-10
- サイズの設定, 21-4
- サイズの選択, 2-14
- 削除, 21-14
- 削除の状態, 21-14
- 作成, 21-7
- 初期, 21-2
- 使用可能にする, 21-10
- 自動的に取得, 21-3, 21-11
- 状態, 21-11
- 状態変更
 - PARTLY AVAILABLE セグメントのオンライン化, 21-11
 - オンライン, 21-11
 - 新規作成時のオンライン化, 21-8
 - 自動的にオンライン化, 21-11
- すべての名前の表示, 21-15
- 遅延, 21-16
- データベース作成後に作成, 21-3
- 等サイズのエクステンツ, 21-5
- トランザクション, 21-13
- トランザクションの明示的割当て, 21-13
- パブリックとプライベートの作成, 21-3
- パブリックの変更, 21-9
- 表示
 - PENDING OFFLINE セグメント, 21-16
 - 情報, 21-15
 - すべての遅延ロールバック・セグメント, 21-16
 - 遅延ロールバック・セグメント, 21-16

- 表領域のオフライン化, 9-12
- 複数を使用, 21-2
- 無効な状態, 21-14
- 割当て, 2-14
- ログ順序番号
 - 制御ファイル, 6-5
- ログ・スイッチ
 - アーカイブ完了待ち, 6-7
 - 強制, 6-16
 - 権限, 6-16
 - 説明, 6-5
 - 多重 REDO ログ・ファイル, 6-7
 - ログ順序番号, 6-5
- ログ・スイッチの強制, 6-16
 - ALTER SYSTEM コマンド, 6-16
- ログ・ライター・プロセス (LGWR)
 - オンライン REDO ログ・ファイルへの書込み, 6-3
 - 使用可能なオンライン REDO ログ, 6-3
 - 多重 REDO ログ・ファイル, 6-6
 - トレース・ファイル, 6-6
 - トレース・ファイルの監視, 4-11
- ロック
 - 監視, 4-8
 - ジョブ・キュー, 8-9

わ

- ワイルドカード
 - ビュー, 15-3
- 割当て
 - 一時領域, 14-5
 - エクステント, 14-11
 - クラスタのエクステント, 17-9
 - ロールバック・セグメントのエクステントの最小化, 21-13
- 割当て制限
 - 一時セグメント, 23-13
 - ゼロに設定, 23-13
 - 表示, 23-23
 - 表領域, 23-12
 - 表領域割当て制限, 9-3
 - 無制限, 23-13
 - ユーザーからの取消し, 23-13
 - リスト, 23-21